UNIVERSIDADE DO VALE DO RIO DOS SINOS — UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA
NÍVEL MESTRADO

ALEXANDRE LUIS DE ANDRADE

GETLB: UMA NOVA ARQUITETURA PARA BALANCEAMENTO DE CARGA DINÂMICO EM SISTEMAS DE TRANSAÇÕES ELETRÔNICAS FINANCEIRAS

SÃO LEOPOLDO
2013

Alexandre Luis de Andrade

GETLB: UMA NOVA ARQUITETURA PARA BALANCEAMENTO DE CARGA DINÂMICO EM SISTEMAS DE TRANSAÇÕES ELETRÔNICAS FINANCEIRAS

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos — UNISINOS

Orientador:
Prof. Dr. Rodrigo da Rosa Righi

Co-orientador:
Prof. Dr. Cristiano André da Costa

São Leopoldo
2013

AGRADECIMENTOS

Agradeço a Deus, pelo dom da vida e à oportunidade de viver este período tão enriquecedor e à Maria, que sempre esteve à frente. Aos meus pais pelo encorajamento, apoio e incentivo, sobretudo meu pai, Silvio, que será sempre exemplo de vida. À namorada Graziela, pelas revisões e companheirismo.

Ao professor e orientador Dr. Rodrigo da Rosa Righi pela empolgação, incentivo e atenção dedicados a esse estudo. Ao professor Dr. Cristiano André da Costa pela co-orientação e acompanhamento desde os primeiros dias do mestrado. Ao Lucas Graebin por todo apoio técnico e tempo dedicado.

À Unisinos, pela disponibilidade da estrutura e qualidade do quadro de professores.

À GetNet, por possibilitar este trabalho e, em especial, aos colegas de trabalho Tiago Jost, Felipe Del Vecchio, Eduardo Roloff, Jessel Baptista, Ricardo Galho e Cristian Cavalheiro.

"*Buscai em primeiro lugar o Reino de Deus e a sua justiça e todas estas coisas vos serão dadas em acréscimo.*".
(Mateus 6:33)

RESUMO

Transações eletrônicas financeiras - em inglês EFT (*Electronic Funds Transfer*) - representam uma realidade em expansão que impulsiona a aproximação entre consumidores e fornecedores. As transações podem ser originadas de diferentes meios de captura da informação e são transmitidas para um centro de processamento, que decodifica, executa e retorna os resultados no menor tempo possível. Em especial, a presente dissertação de mestrado está enquadrada no dia-a-dia da empresa GetNet e no balanceamento de carga sobre um de seus sistemas de processamento.

O sistema em estudo dispõe atualmente de uma arquitetura na qual as transações são recebidas em um nó centralizador, representado por um chaveador Cisco, e são distribuídas para Máquinas Processadoras (MP) segundo o algoritmo de escalonamento Round Robin. Esse algoritmo é adequado para as situações onde as MP são homogêneas e o tempo de rede entre elas e o nó centralizador é também uniforme. Essa configuração pode ser restritiva para um sistema transacional que precisa estar dimensionado para atender uma empresa em expansão geográfica que use MP em diferentes localizações. Em adição, a atuação do algoritmo Round Robin permite que transações sejam repassadas para MP que já possuem alta carga de trabalho ou se encontram indisponíveis.

Nesse contexto, essa dissertação descreve uma arquitetura para o sistema transacional chamada GetLB, que proporciona um balanceamento de carga a partir do chaveador com base em monitoração das máquinas de processamento. A infraestrutura GetLB propõe um novo escalonador chamado LL (*Load Level*). Diferente do Round Robin, para cada transação i, LL calcula n funções $LL(i, j)$, onde j é a MP alvo e n a quantidade das mesmas. Para tal, serão verificadas as suas condições para atender à demanda conforme índices de ocupação de CPU, memória, disco e estado das filas das máquinas processadoras, entre outros. A eficiente combinação desses fatores, para o despacho de transações, configura a contribuição científica do trabalho. Assim como a eficiência em notificar o chaveador quando ocorrem eventos assíncronos no sistema, como parada ou perda de capacidade de processamento de alguma máquina. A arquitetura GetLB também visa proporcionar capacidade de operação com heterogeneidade de máquinas processadoras, possibilitando o uso de máquinas especialistas para certos tipos de transações e a integração de MPs de diferentes redes (domínios administrativos).

Os resultados dos testes mostraram que GetLB é mais eficiente que Round-Robin em muitos aspectos, sobressaindo mais em situações de heterogeneidade de recursos e dinamicidade de eventos. Verificou-se também que as notificações acrescentam muito valor ao sistema, auxiliando na manutenção e melhoria contínua da solução.

Palavras-chave: Sistema de Transações Eletrônicas Financeiras. Balanceamento de Carga. Transferência Eletrônicas de Fundos. Escalonamento Dinâmico.

on

ABSTRACT

Electronic financial transactions represents an expanding reality that drives the approach between consumers and suppliers. Transactions can be originated from different information's capture terminals and are transmitted to a processing center, which decodes, executes and returns the results in the shortest time possible. In particular, this dissertation is framed in the day-to-day operations of GetNet, a network and service provider company, and load balancing on one of its processing systems.

The system under study currently has an architecture in which transactions are received at a centralized node, represented by a Cisco switch, and are distributed to machines processors (MP) according to Round Robin scheduling algorithm. This algorithm is suitable for situations where the MP are homogeneous and network transfer time between them and the centralized node is also uniform. This setting can be restrictive for a transactional system that needs to be sized to meet an expanding company and uses geographic MP at different locations. In addition, the performance of Round Robin algorithm enables transactions to be passed to MP who already have high workload or are unavailable.

In this context, this work describes an architecture for a transactional system called GetLB, which provides load balancing from the switch based on processing machines's monitoring. The infrastructure GetLB proposes a new scheduler called LL (Load Level). Unlike the Round Robin, for each transaction i, LL calculates n functions $LL(i, j)$ where j is the target MP and n quantity of machines. This will check their conditions to meet demand as occupancy rate of CPU, memory, disk, and state of the queues of processing machines, among others. The efficient mix of these factors to dispatch transactions, sets the scientific work. As well as the efficiency to notify the switch when asynchronous events occur in the system, such as stopping or loss of processing capacity of some machine. The architecture also aims to provide GetLB operation capability with heterogeneous processing machines, enabling the use of machinery specialists for certain types of transactions and the integration of MPs from different networks (administrative domains).

The test results showed that GetLB is more efficient than Round-Robin in many aspects and stand out in case of heterogeneous resources and dynamic events. It was also found that the notifications adds much value to system, assisting in the maintenance and continuous solution's improvement.

Keywords: Electronic Financial Transactions System. Electronic Funds Transfer. Load Balance. Dynamic Scheduling.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACE *Application Control Engine*

API *Application Programming Interface*

APN *Access Point Name*

ATM *Automatic Teller Machine*

CV Coeficiente de Variação

EFT *Electronic Funds Transfer*

ES Entrada e Saída

FIFO *First In First Out*

GPRS *General Packet Radio Service*

HD *Hard Disk*

ISO *International Organization for Standardization*

JVM *Java Virtual Machine*

LL *Load Level*

MP Máquina Processadora

MPLS *Multiprotocol Label Switching*

NAC *Network Access Control*

N&SP *Network Service Provider*

POS *Point Of Sale*

RDMA *Remote Direct Memory Access*

RMI *Remote Method Invocation*

RPE Redes de Petri Estocásticas

RR Round Robin

SDLC *Synchronous Data Link Control*

SI Sistema Interno

SLA *Service Level Agreement*

SO *Sistema Operacional*

TDM *Time Division Multiplex*

TEF Transferência Eletrônica de Fundos

VM *Virtual Machine*

XCP *Xen Cloud Platform*

XML *Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Sistemas de roteamento e processamento de requisições financeiras são elementos fundamentais de uma rede de transações eletrônicas. Normalmente, uma transação eletrônica está aliada a uma requisição de compra ou saldo e percorre um caminho de ida e volta desde o terminal até um centro de processamento. Um terminal pode ser representado por um ponto de venda (POS - *Point of Sale*), Transferência Eletrônica de Fundos (TEF), ATM (*Automatic Teller Machine*), bem como por dispositivos móveis (NASSAR; MILLER, 2013). Quanto às transações, pode-se destacar aquelas que envolvem cartões de crédito e débito, recarga para telefonia pré-paga e cartões de transporte público, saque e depósito para sistema bancário e consultas financeiras. Cada um desses tipos de transação possui requisições próprias de CPU e acesso a banco de dados, podendo usar diferentes subsistemas dentro do centro de processamento.

Cada transação eletrônica engloba as etapas de solicitação, resposta e confirmação. Os seguintes objetivos são destacados no tratamento das requisições: (i) alto desempenho no processamento das transações com menor custo computacional possível e; (ii) alta disponibilidade do sistema transacional para evitar perda de transações. Esses dois objetivos passam por um eficiente escalonamento de transações feito por um elemento chaveador e a análise da escalabilidade da solução corrente para atender períodos de pico na chegada de transações. O sistema deve estar sempre preparado para atender a mais alta demanda. Naturalmente, isso faz com que recursos computacionais fiquem ociosos durante alguns momentos do dia, uma vez que o período de maior volume de chegadas no sistema ocorre das 8h00min às 22h00min. Na chegada ao centro de processamento, cada transação passa por um chaveador que é responsável por despachá-la para uma máquina Atendedora. Tradicionalmente, a decisão para quem endereçar um transação é dada pelo algoritmo muito utilizado em sistemas distribuídos chamado Round Robin, ou RR. O algoritmo RR tem baixa complexidade, uma vez que pode ser facilmente implementado como uma lista circular de recursos.

1.1 Motivação

O emprego do escalonamento RR é adequado em sistemas homogêneos, tanto na caracterização das transações que estão chegando quanto na configuração das máquinas processadoras. Quando não há essa condição, ele pode não ser a melhor solução. Embora as máquinas recebam um número igualitário de transações, o tempo de conclusão das tarefas na fila depende de características como CPU, capacidade de memória e disco na máquina processadora e das demandas dessas propriedades pelos tipos de transações (ARAUJO et al., 2009). Aliado aos fatores já mencionados, também há o fato que comumente as máquinas processadoras constituem um parque de máquinas localizado no mesmo espaço físico. Nesse sentido, o tempo de acesso a cada uma é, por padrão, uniforme. Com o algoritmo RR, é possível que ocorra perda de transações na medida que uma máquina Atendedora pode não conseguir processar as

tarefas em sua fila dentro do limite máximo de tempo imposto ao sistema. Em outras palavras, pode-se ter um cenário onde uma máquina está processando uma transação que requisita muito de sua capacidade e, enquanto isso, o chaveador segue adicionando transações na sua fila de processamento.

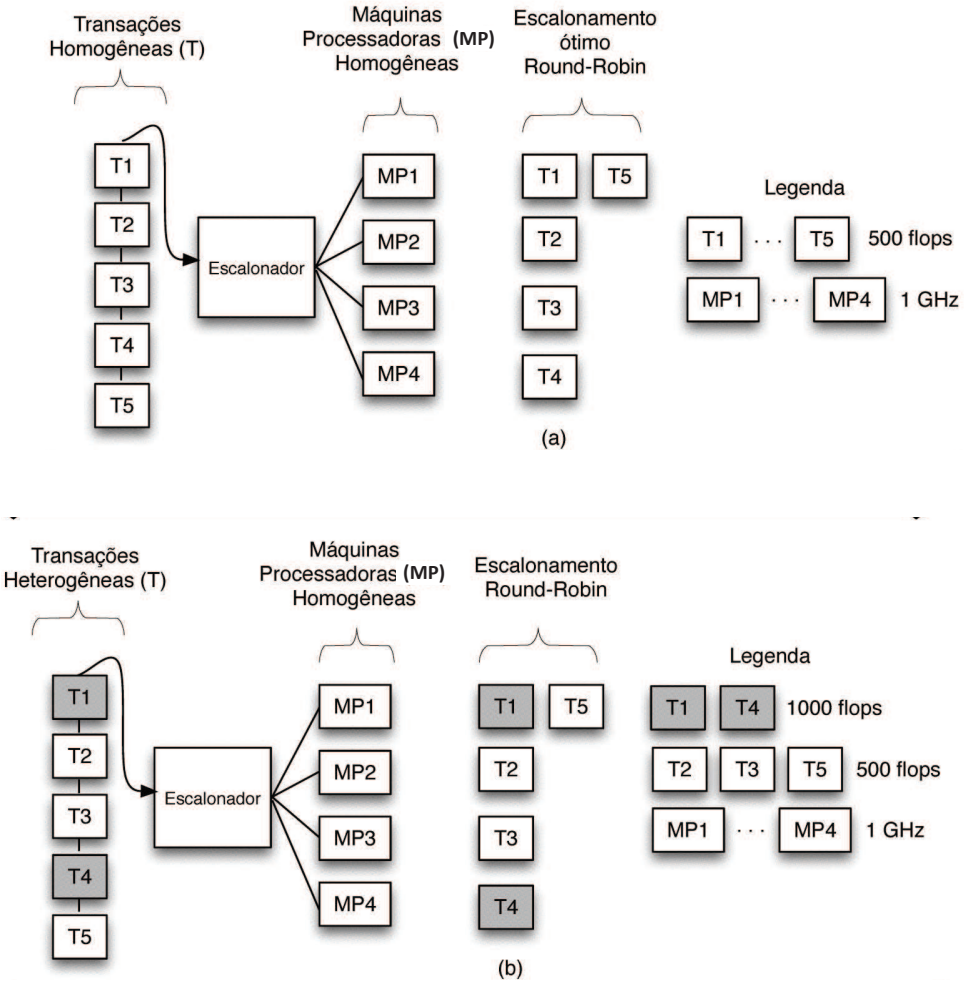
Conforme ilustrado na Figura 1 (a), o método Round-Robin de escalonamento representa uma maneira rápida e fácil para atingir um escalonamento ótimo quando o conjunto de transações e máquinas processadoras forem homogêneos, ou seja, para situações em que as transações são todas do mesmo tipo com as mesmas demandas de CPU e ES e as máquinas processadoras iguais entre si. Claramente, conforme representado na Figura 1(b), Round-Robin não é a melhor alternativa quando sistemas dinâmicos e/ou heterogêneos estão presentes. Neste exemplo, as máquinas processadoras, apesar de homogêneas, recebem cargas diferentes e são submetidas a diferentes demandas. Em especial, sistemas de transações eletrônicas se caracterizam por apresentar situações de heterogeneidade de carga, pois processam diferentes tipos de transações, e de tempo de recebimento, uma vez que o intervalo entre chegadas das transações não é constante.

Dado que as transações são heterogêneas, cada tipo possui os seus requisitos de CPU, acesso a banco de dados, rede e memória. Tais peculiaridades podem fazer com que o escalonamento Round-Robin sobrecarregue algumas máquinas e deixe outras ociosas. A Figura 1(b) ilustra tal situação, na qual as máquinas processadoras são homogêneas e localizadas numa mesma sala de servidores (*data center*). Em adição, todos os subsistemas, como segurança da informação, banco de dados e prevenção a fraudes, requeridos pelas MPs, devem estar presentes no mesmo endereçamento de rede.

Outro fator significativo em um sistema transacional, é o constante aumento do volume de transações. Segundo levantamento da ABECS (Associação Brasileira das Empresas de Cartões de Crédito e Serviços), as compras que tiveram como meio de pagamento cartões de crédito e débito alcançaram R$ 189,43 bilhões no primeiro trimestre deste ano, aumento de 16,9% em relação ao mesmo intervalo de 2012 (ZEROHORA, 2013). Foram registradas 2,13 bilhões de transações com esses plásticos, elevação de 14%, na mesma base de comparação (ABECS, 2013). Na Figura 2 está representado o crescimento registrado do faturamento do mercado de cartões nos últimos anos.

Para suportar as demandas que se originam desse crescimento, um centro de processamento de transações deve planejar sua atuação com algoritmos de escalonamento mais eficientes que o RR. Ao utilizar uma distribuição de carga com base no algoritmo RR, aumentando o número de máquinas processadoras é possível reduzir perdas de transações, contudo esse é um método que envolve aumento nos custos e não suprime a possibilidade de perda de transações no sistema. Além deste fato, observa-se que cada vez mais acontece a fusão e/ou expansão de empresas e, no contexto transacional, isso implica que uma transação possa ser despachada para máquinas geograficamente distantes. Uma vez que esse processo pode envolver vários países, existirão diversos picos de demanda por dia, com possibilidade de diferentes fusos horários.

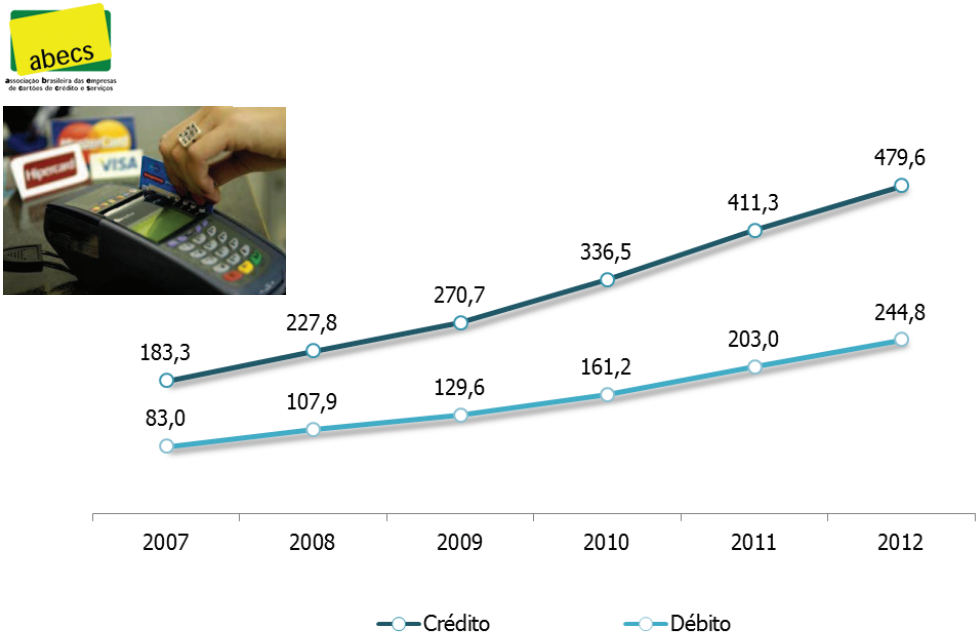
Figura 1: Escalonamento de transações: (a) sistema homogêneo; (b) sistema heterogêneo com Round-Robin.



Fonte: Desenvolvido pelo próprio autor.

Atualmente, há uma preocupação com o planejamento de sistemas de processamento de transações porque o Brasil sediará os eventos da Copa do Mundo de 2014 e as Olimpíadas em 2016. Nesse contexto, uma ação possível passa pelo balanceamento de carga quanto ao despacho de transações. Isso para atender a transações heterogêneas e máquinas processadoras também heterogêneas e potencialmente distantes. Esse último item é pertinente porque um custo de comunicação pela Internet pode chegar a ser 1000 vezes maior que em uma rede local (TANENBAUM, 2007). Nessa linha, a presente dissertação de mestrado está inserida na dinâmica de um centro de processamento de transações e como oferecer balanceamento de carga no tratamento das transações. A próxima seção deste capítulo discute o objeto principal do trabalho.

Figura 2: Crescimento no Brasil do faturamento, por função dos cartões (em bilhões de R$).



Fonte: ABECS (2013).

1.2 Definição do Problema e Objetivos

Conforme abordado na seção anterior, o algoritmo RR pode não ser a melhor opção como abordagem de escalonamento para um sistema distribuído heterogêneo. Entretanto, um sistema transacional comumente se caracteriza pela heterogeneidade quanto aos recursos, transações, fluxo de chegada e taxa de atendimento (SOUSA et al., 2012). Por exemplo, podem existir transações que necessitam de processamento de imagens eficiente que pode ser tratado por um subconjunto de máquinas processadoras ou ainda, outras transações que possuem métodos criptográficos com chaves largas que precisam de um hardware específico para serem computadas. Considerando que o algoritmo RR simplesmente adota uma fila circular, o tratamento das questões acima fica inviável. Em adição, as máquinas processadoras possuem um caráter dinâmico em função das transações em sua fila e do estado da rede, CPU e periféricos (ARAUJO et al., 2009). Claramente, uma alternativa passa pela adoção de técnicas de balanceamento de carga que consideram os fatores heterogêneos do sistema transacional. Nesse sentido, a presente dissertação se propõe a desenvolver a seguinte Sentença Problema.

- **Sentença Problema:** Dado um sistema transacional *heterogêneo*, o desafio consiste em oferecer uma arquitetura de balanceamento de carga *dinâmico* que seja *flexível* e *pró-*

ativa, oferecendo condições para suportar um crescimento transacional mantendo alta disponibilidade e *eficiente* utilização dos recursos computacionais.

A sentença problema acima apresenta alguns itens chave para o trabalho, grafados em itálico. Estes itens podem ser entendidos da seguinte forma:

i. *Heterogêneo*: sistema que é composto por diferentes domínios administrativos de rede, diferentes capacidade de processamento de máquinas e diferentes tipos de transações eletrônicas.

ii. *Dinâmico*: esse item diz respeito ao tipo de escalonamento, o qual é efetuado com informações capturadas em tempo de execução, ou seja, na medida que as transações chegam até a empresa. Essa necessidade é citada porque o fluxo de chegada não é constante, bem como o estado das MP e da própria rede de comunicação (PATIL; GOPAL, 2013). O termo escalonamento dinâmico usado nesse trabalho é o proposto por Casavant e Kuhl, em detrimento da abordagem estática também proposta por esses autores (CASAVANT; KUHL, 1988);

iii. *Flexível*: indica possibilidade de troca, a qualquer momento, da política de escalonamento, incluindo e retirando parâmetros. Isso permite privilegiar algumas transações em momentos específicos.

iv. *Pró-ativa*: permite uma ação antecipada para eventuais problemas, fazendo com que o sistema não seja apenas reativo, tornando o sistema menos suscetível à perda de transações. A proatividade está relacionada com as notificações que atualizam as informações do escalonador de maneira assíncrona.

v. *Eficiente*: escalonamento eficiente, ou seja, que apresenta um tempo médio de resposta otimizado com objetivo de melhorar o compartilhamento de recursos, obtido através de uma heurística adequada.

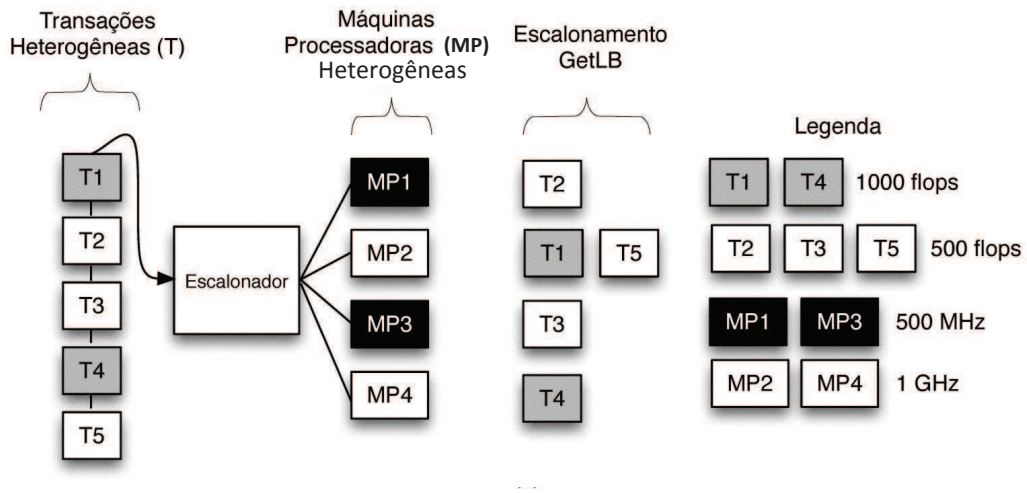
Para contemplar os aspectos enfatizados na sentença problema descrita acima, esta dissertação tem como objetivo:

- **Objetivo:** Apresentar uma nova infraestrutura transacional, denominada GetLB, capaz de balancear transações financeiras heterogêneas em um sistema distribuído de modo eficiente em um contexto heterogêneo, de domínios de rede e recursos, através de uma eficiente heurística de balanceamento de carga, chamada de LL (*Load Level*). A infraestrutura GetLB também visa melhorar o compartilhamento de recursos, otimizando o tempo médio de resposta com eficiente balanceamento de cargas em um sistema escalável e de fácil manutenção.

Na Figura 3 está representado o comportamento que é esperado do escalonador GetLB, atuando com a mesma situação de heterogeneidade de transações apresentadas na Figura 1

(b). Uma vez que o escalonador apresenta a capacidade de operar com heterogeneidade de máquinas, as transações são distribuídas de acordo com as capacidades de processamento de cada máquina.

Figura 3: Escalonamento de transações sob a perspectiva de GetLB.



Fonte: Desenvolvido pelo próprio autor.

Com esta nova infraestrutura transacional, se obtém *eficiência* através do uso da heurística LL, melhorando o compartilhamento de recursos e otimizando o tempo médio de resposta pois, conforme as políticas de distribuição de carga a serem adotadas, os recursos que tem menor carga recebem transações mais adequadas. Para manter o escalonador atualizado sobre o estado das máquinas processadoras, GetLB utiliza métodos de notificações assíncronas entre os recursos e o escalonador, tornando a infraestrutura *pró-ativa*, antecipando-se a problemas de indisponibilidade. Também através de notificações assíncronas, é possível obter um sistema *flexível*, pois possibilita que sejam modificadas, a qualquer momento, as políticas de escalonamento, incluindo ou retirando parâmetros. Na medida em que GetLB possui uma arquitetura escalável, em conjunto com a heurística LL e com o método de notificações, é possível contemplar os requisitos de *heterogeneidade* e *dinamicidade*, identificando o tipo de transação recebida e encaminhando para a máquina adequada, ou especialista, em tempo de execução.

1.3 Organização do Texto

Este documento está organizado em sete capítulos. Após a introdução, no Capítulo 2 serão apresentados os conceitos abordados no trabalho e sobretudo uma descrição de um sistema de transações financeiras, com suas características operacionais, tipos de transações financeiras eletrônicas e informações relevantes quanto ao seu processamento. Ainda neste capítulo se encontram os tópicos que abordam questões de escalonamento, balanceamento de carga e uma modelagem formal do sistema transacional. No Capítulo 3 estão os trabalhos relacionados, nos

quais se apresentam soluções de balanceamento de carga e escalonadores que fazem uso de recursos semelhantes aos que serão utilizados na dissertação. No Capítulo 4 está a proposta da pesquisa, com as decisões do projeto e sua modelagem e, encerrando o capítulo, se apresenta uma descrição da nova arquitetura de balanceamento de carga, denominada GetLB, bem como uma explicação de como se dará o escalonamento das transações e uma avaliação dos resultados. O Capítulo 5 descreve a estrutura e o funcionamento do protótipo que foi desenvolvido como prova de conceito das decisões de projeto e para validação da heurística de balanceamento de carga. A seguir, no Capítulo 6 serão avaliados os resultados obtidos através de testes aplicados ao protótipo, com utilização de rastro de transações reais, apresentando os resultados de GetLB sendo executado em diferentes topologias. Por fim, serão apresentadas as conclusões no Capítulo 7, no qual serão enfatizadas as principais contribuições e resultados da arquitetura GetLB.

2 REFERENCIAL TEÓRICO

O objetivo desse capítulo é apresentar as terminologias e conceitos que serão usados nos capítulos seguintes do presente trabalho, assim como descrever o funcionamento de um sistema de transações eletrônicas financeiras. O capítulo também salienta a importância do escalonamento de processos e do balanceamento de carga em sistemas distribuídos. Em adição, também será abordada a importância desses tópicos em ambientes heterogêneos e dinâmicos.

O presente capítulo está dividido em quatro seções. A Seção 2.1 descreve um sistema transacional, suas particularidades e explica o fluxo das transações financeiras. Esta explanação percorre os diversos componentes e subsistemas envolvidos, evidenciando suas características e necessidades. Após essa contextualização, na Seção 2.2 são exploradas técnicas de escalonamento e como se aplicam em sistemas distribuídos, tal como um sistema transacional. Além das técnicas apresentadas na seção anterior, a Seção 2.3 destaca a relevância do balanceamento de carga e das técnicas que podem ser empregadas para melhorar o desempenho e a eficiência de sistemas distribuídos. Vistas as técnicas e estratégias de escalonamento e balanceamento de carga, a Seção 2.4 aborda aspectos da modelagem de sistemas computacionais e sua relevância na medida em que possibilita modelar e simular diferentes condições sistêmicas, sobretudo as pertinentes à proposta desse estudo. Por fim, a Seção 2.5 apresenta um resumo desse capítulo, relacionando os principais temas que foram abordados.

2.1 Sistema de Transações Eletrônicas Financeiras

O conceito de transação eletrônica, para o mercado de cartões de pagamento, é dado como sendo uma função do cartão para compras ou saques, qualquer troca de valor financeiro por bens ou serviços (ABECS, 2013). Para que uma transação eletrônica financeira aconteça, é necessário que três principais componentes estejam envolvidos, os quais são: (i) meio de captura, (ii) sistemas para receber, processar e rotear a transação ou N&SP (*Network Service Provider*), e (iii) instituição autorizadora da transação (VISA, 2012). Estes atuam de maneira sincronizada visando garantir o correto processamento em todas as etapas envolvidas no fluxo transacional e, em consequência, a satisfação dos usuários através de um processamento eficiente e com alta disponibilidade. A sincronia destes componentes também visa prever e contornar situações de falha durante o processamento, tanto para evitar prejuízos ao desempenho e disponibilidade, quanto processamento indevido e consequente prejuízo ao saldo do usuário.

O processo de autorização de uma transação financeira pode ser analisado através de uma sequência de passos (MASTERCARD, 2012), conforme apresentado na Figura 4. Nesta sequência, primeiramente o portador do cartão realiza uma compra de bens ou serviços em um estabelecimento comercial que por sua vez utiliza um terminal de venda. Este terminal transmite um pacote de dados para e a rede de captura ou N&SP, este por sua vez identifica o emissor do cartão e direciona para o autorizador correspondente. O sistema autorizador realiza validações

de segurança e integridade, encaminhando a seguir uma solicitação de aprovação para a instituição financeira ou banco que emitiu o cartão. Mediante uma aprovação do banco emissor, uma resposta retorna para o sistema autorizador que direciona ao N&SP. Por fim, a rede de captura envia aprovação para estabelecimento comercial e o portador do cartão finaliza a compra recebendo o comprovante de venda do terminal.

As transações podem ainda ser separadas em dois grupos: (i) financeiras e; (ii) administrativas. Na primeira, estão incluídas todas as transações que envolvem valores financeiros, debitando ou creditando valores que alteram o saldo bancário do usuário. Estas transações podem ocorrer através de compras com cartão, saques ou depósitos em caixas eletrônicos, recarga de telefonia para telefones pré-pagos, entre outras operações. Na categoria administrativas, estão aquelas transações que não envolvem valores financeiros e tem caráter informativo, de configuração do terminal ou atualização de versões de aplicativo nos dispositivos. O volume de transações que são recebidas no sistema, normalmente é medido por número de transações por segundo (TPS), sendo esta medida também utilizada para indicar a capacidade de processamento do sistema.

Figura 4: Etapas para autorização de uma transação eletrônica.

Fonte: MASTERCARD (2012).

Na primeira etapa de uma transação, seja ela financeira ou administrativa, se encontram os meios de captura que irão coletar as informações necessárias para efetuar a solicitação. Nesta fase, estes dispositivos são responsáveis pela captura, formatação e transmissão dos dados a uma central de processamento. Os meios de captura fazem parte do cotidiano das pessoas, e estão distribuídos no comércio através de terminais de venda do tipo POS, TEF no caso das redes de farmácias e supermercados por exemplo, ATM, que são os caixas eletrônicos das redes bancárias e, mais recentemente, através de dispositivos móveis (MATBOULI; GAO, 2012). Nestes meios, as informações necessárias para efetuar uma transação são adquiridas de modo seguro, visando manter a integridade e confidencialidade durante todo o processo.

Para efetuar uma transação financeira, é necessário dispor de algumas informações básicas, tais como o número do cartão, valor da transação, senha do usuário, data e hora da solicitação,

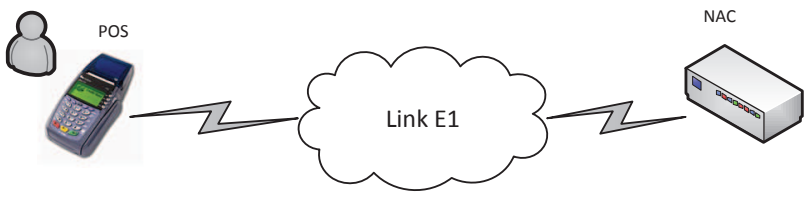
códigos identificadores do estabelecimento comercial e do terminal, entre outras. Este conjunto de informações é agrupado em um pacote de dados conforme a norma ISO 8583 (ISO, 1993), adotado pela indústria financeira internacional como protocolo para troca de mensagens. Apesar de ser uma norma que visa criar um padrão na comunicação, facilitando a integração entre as instituições, muitas destas fazem adaptações em alguns campos das mensagens para adequar às suas necessidades. Com isso, frequentemente se torna necessário adaptar a mensagem recebida dos terminais antes de ser transmitida para as processadoras de cartão e telefonia.

Após coletar e agrupar os dados no formato especificado pela norma ISO, os terminais transmitem a solicitação através de uma conexão que pode ser X25, X28 ou TCP/IP, conforme o tipo de terminal e disponibilidade da região em que se encontra. Convém aprofundar o entendimento da conectividade dos terminais pois, a partir dos canais de comunicação é que as demandas transacionais são recebidas e precisam ser adequadamente distribuídas entre os servidores. Com o propósito de auxiliar nesse entendimento, nos próximos parágrafos serão apresentados exemplos relacionados à empresa GetNet (GETNET, 2013), na qual está embasado este estudo de sistema de transações financeiras e também por adotar soluções de conectividade que são típicas do mercado transacional.

A conexão entre os meios de captura e a GetNet pode ser via discada, GPRS ou *Ethernet*. Para as conexões discadas, a empresa desenvolveu sua própria rede de comunicação, com estrutura de comunicação veloz e contingenciada, a fim de otimizar o uso da rede e garantir uma melhor percepção do cliente com o desempenho dos seus equipamentos. Esta estrutura utiliza-se de *links* de comunicação do tipo E1, que são disponibilizados pelas operadoras de telefonia (OI, Embratel, Telefônica etc.). Este *link* é do padrão TDM (*time division multiplex*), e possui 32 canais, cada um com 64 Kbps de largura de banda, totalizando 2048 Kbps (2 Mbps). Na medida em que são utilizados 2 canais para controle, restam 30 canais por *link*, para comunicação de voz ou dados.

O *link* de comunicação geralmente possui um intervalo de números telefônicos associados que podem ser utilizados para realizar as transações financeiras, sendo conectado a uma placa dedicada (Figura 5) e está ligada em uma máquina chamada de NAC (*Network Access Control*). Esta máquina, utilizando sistema operacional Linux, disponibiliza cada um dos 30 canais de comunicação como uma porta serial, permitindo o atendimento simultâneo destas ligações.

Figura 5: Estrutura de comunicação para POS que utiliza conexão discada através de modem.



Fonte: Desenvolvido pelo próprio autor.

Esta estrutura de comunicação está replicada em diversos pontos do país, com o objetivo

de atender a todas as operadoras de telefonia do território nacional, já que ligações originadas de uma operadora para outra estão sujeitas à interconexão, degradando o serviço prestado. Nessa estrutura, os NACs possuem até três placas de comunicação, totalizando no máximo 90 canais de dados disponíveis por máquina instalada. Estas máquinas são agrupadas por região, e utilizam-se de um *hunting group*, que é uma facilidade oferecida pelas operadoras de telefonia para permitir que um mesmo número telefônico seja distribuído para diversos *links* E1. Desta maneira, um número associado possui até 360 canais para atendimento simultâneo na estrutura estudada.

Além da comunicação discada, alguns tipos de terminais disponibilizam comunicação via rede *Ethernet* (TCP/IP), através da internet. As mensagens que trafegam nesse meio são encriptadas em sua totalidade, não apenas o número do cartão e senha. Outra parcela dos terminais, comunica-se através da rede GPRS das operadoras de telefonia móvel. Nesse tipo de conexão as operadoras recebem os pacotes e encaminham para os servidores da empresa através de conexão TCP/IP em uma APN privada estabelecida entre as empresas.

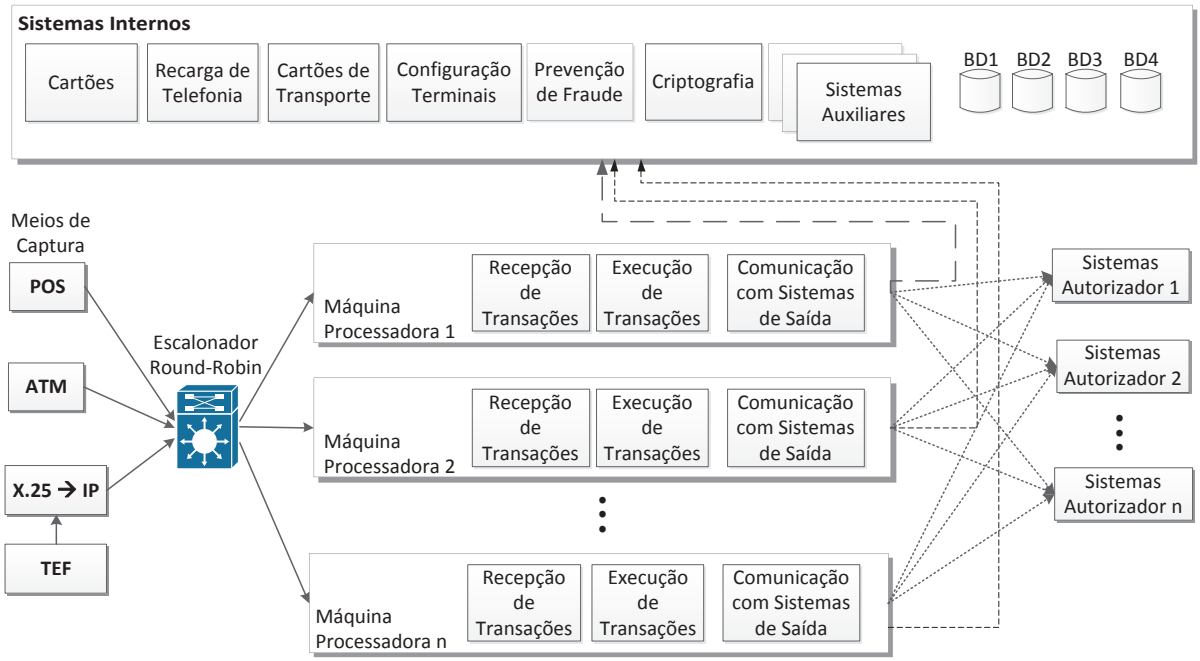
Para os equipamentos TEF, a transação é recebida em um *link* de comunicação X25. Tal protocolo orientado à conexão é padrão das redes de TEF no Brasil (SOUSA et al., 2009). Este *link* é ligado fisicamente a uma placa X25 e, uma vez que as transações são lidas pela aplicação, são convertidas imediatamente para o protocolo TCP/IP e enviadas para alguma das máquinas de processamento.

Com exceção da conexão X25, todas as demais encaminham os pacotes de dados para um chaveador Cisco Catalyst 6500 Series (CISCO SYSTEMS, 2012), conforme representado na Figura 6. A partir deste equipamento, as transações são distribuídas entre as máquinas processadoras segundo o algoritmo Round Robin. A conexão entre o chaveador e as máquinas processadoras se dá através de *sockets*, que são abertos a cada transação e fechados assim que a transação é concluída, conforme um comando do terminal.

Para dispor de informações íntegras em sua origem, a GetNet desenvolve os aplicativos de captura da informação que são embarcados em seus terminais, de modo que apenas os dispositivos autorizados estejam habilitados a conectar em seus servidores. Após estabelecida a conexão, os pacotes de dados recebidos precisam atender à especificação da norma ISO 8583 que é adotada pela empresa, caso contrário serão descartados. Esta é a primeira etapa do processo transacional e envolve basicamente um usuário operando o terminal, o estabelecimento comercial em que está instalado o equipamento e os sistemas de entrada de dados na empresa.

Para maior entendimento, a partir da Figura 6, os vários subsistemas do sistema transacional foram agrupados a fim de compilar em somente quatro módulos, conforme a Figura 7. Nesta nova representação do sistema, as etapas de uma transação foram numeradas na sequência em que ocorrem. O processo inicia com o passo 1, que representa as transações originadas de diferentes meios de captura sendo recebidas nas máquinas processadoras. As transações são recebidas em um chaveador e então encaminhadas para uma Máquina Processadora, para primeiramente processar a senha nos sistemas de criptografia, no passo 2. A seguir, no passo 3,

Figura 6: Topologia do sistema de transações eletrônicas financeiras.



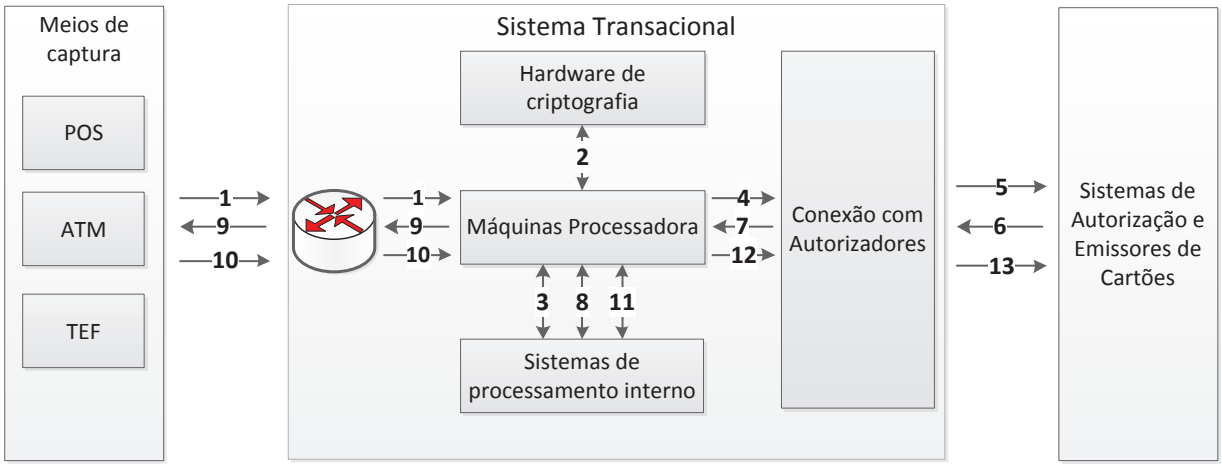
Fonte: Desenvolvido pelo próprio autor.

cada transação é registrada em um respectivo sistema interno de acordo com seu tipo. Nestes sistemas a integridade dos dados da transação são validados e registrados. Tais registros serão feitos em cada etapa da transação, gerando assim um informativo completo de todo fluxo transacional. Esses arquivos são utilizados para identificar transações incompletas, conciliar dados com as empresas autorizadoras, conforme será visto mais adiante, e para gerar os dados de faturamento da empresa.

Assim que a transação é validada e registrada nos sistemas internos, ela retorna para a máquina atendedora e é encaminhada para o módulo de conexão com os externos, que corresponde ao passo 4. Com isso, a mensagem de solicitação estará pronta para ser roteada à respectiva empresa responsável pela emissão e autorização do cartão, conforme passo 5 do fluxo. Nesta fase da operação, a requisição será autorizada ou negada. De fato, esta é a etapa da operação financeira para creditar ou debitar os devidos valores. A autorização se dá de acordo com análise de integridade da transação, validade do cartão e senha do portador, entre outros. Com a transação autorizada ou negada, uma resposta é retornada para o sistema transacional de origem, passo 6, iniciando assim a segunda etapa do fluxo transacional.

Na segunda etapa do processamento transacional, a mensagem retorna da autorização externa para o sistema transacional da empresa. Estas mensagens são recebidas no módulo de conectividade. A seguir, assim que os dados chegam às máquinas processadoras, conforme passo 7, a mensagem é direcionada novamente para o sistema interno correspondente, representado pelo passo 8, no qual é validada quanto à sua integridade e também atualizam-se os registros transacionais internos. A seguir, a mensagem retorna para a máquina atendedora, sendo assim

Figura 7: Caminho percorrido por uma transação financeira nos sistemas de uma empresa de captura e processamento de transações (N&SP), desde a recepção, passando por validações de segurança e controles internos até ser despachada para o autorizador. A resposta do autorizador é novamente validada e transmitida de volta para o terminal localizado no estabelecimento comercial em que se originou.



Fonte: Desenvolvido pelo próprio autor.

transmitida ao terminal de origem através do chaveador, que corresponde ao passo 9.

Para as transações não autorizadas, seja por motivos de saldo ou alguma inconsistência de dados, o fluxo se encerra nessa etapa. Para as aprovadas ocorrerá a terceira e última fase da operação. Nesta etapa, conforme o passo 10, ocorre a transmissão de uma mensagem de confirmação da transação a partir do terminal, informando que a mensagem de resposta foi recebida e processada corretamente. Esta mensagem é registrada nos sistemas internos através do passo 11 e encerra o fluxo na empresa de autorização externa com o passo 13.

Conforme os parágrafos anteriores, descreveu-se passo-a-passo o fluxo de uma transação concluída com sucesso, seja aprovada ou negada. Contudo há situações em que podem ocorrer falhas na comunicação em alguma das etapas do processo transacional. Por exemplo, uma interrupção da comunicação entre o terminal de venda e o sistema, durante a confirmação da transação conforme o passo 10, irá deixar o sistema sem o conhecimento de que a compra foi concretizada ou não. Do mesmo modo, as possibilidades de falha estão localizadas em todas as etapas do fluxo portanto, os três principais componentes do sistema (VISA, 2012) precisam estar preparados para lidar com pendências típicas das transações financeiras, garantindo que o saldo esteja sempre íntegro.

Outra ocorrência frequente que precisa ser tratada através de processos diferenciados no sistema, é aquela em que o meio de captura envia uma mensagem de solicitação e a mesma se perde em alguma das próximas etapas do sistema, seja por falha na conexão ou indisponibilidade de algum subsistema envolvido. Para isso existe nos meios de captura um contador de tempo limite para receber a resposta, tipicamente de 30 segundos. Contudo, descontado o

tempo despendido com a conectividade, que é normalmente de 2 segundos, o sistema tem 28 segundos para responder à solicitação. Caso esse tempo limite seja atingido, o terminal transmite para o sistema uma transação denominada Transação de Reversão. Ao receber essa mensagem, o sistema verifica se de fato recebeu alguma transação que corresponde aos dados da reversão, se não a localizar, a mensagem é ignorada, porém, se for localizada, os valores originais da transação são restituídos.

Outras medidas são adotadas para contornar as situações de pendências no sistema, como trocas de arquivos entre as empresas de processamento e captura de transações para conciliação de informações. A periodicidade desses arquivos varia conforme o acordo entre as empresas visando sempre manter o menor número possível de transações pendentes em suas bases.

O fluxo de transações recebidas no sistema varia muito ao longo do dia, conforme os dados mostrados na Figura 8, na qual se observa o comportamento típico das ocorrências de transações em um dia com comércio aberto, demonstrando a irregularidade da demanda. O gráfico foi obtido através de registros de mensagens de solicitação de compra sobre o sistema de processamento da GetNet. Durante as primeiras horas do dia, o volume é reduzido e se mantém neste patamar até aproximadamente 07h00min. A partir desse momento, há picos no entorno das 12h00min e 19h00min. Naturalmente, os picos dizem respeito aos horários de almoço e saída do trabalho, nos quais os usuários realizam mais compras. Com base nessas informações, um sistema transacional precisa ser dimensionado para atender aos períodos de baixa demanda e também apresentar os mesmos níveis de disponibilidade e desempenho nos momentos de pico.

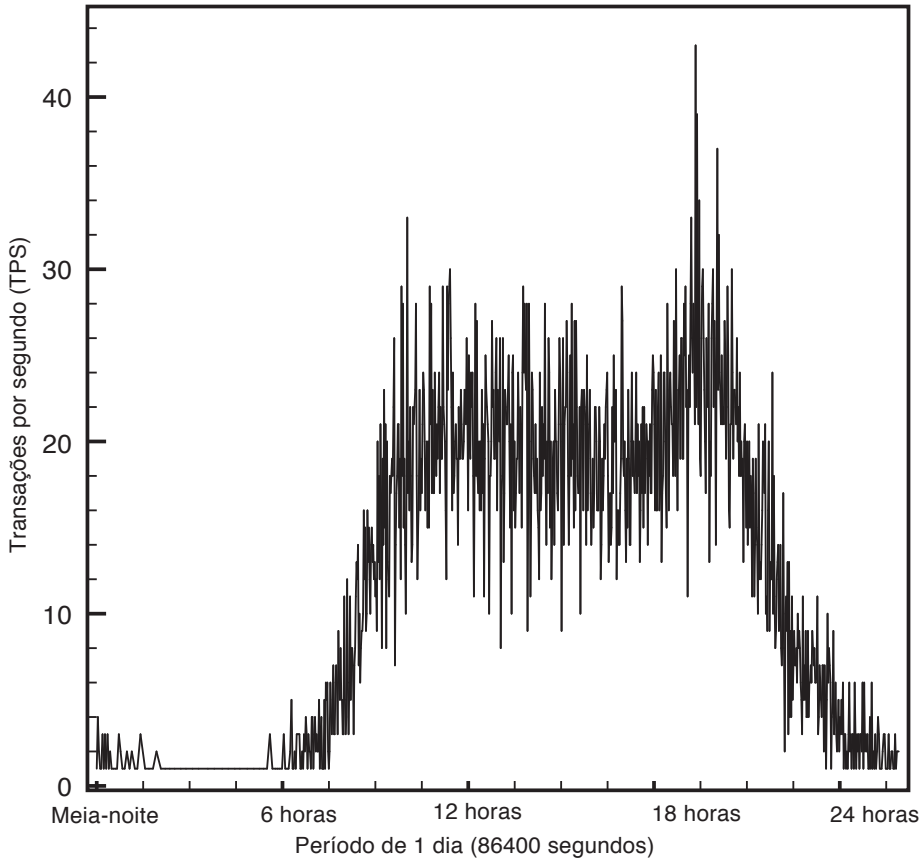
Para o processamento destas transações, a empresa utiliza máquinas atendedoras idênticas, caracterizando um sistema homogêneo. Nestas máquinas as transações são recebidas em uma fila de entrada e consumidas para serem processadas, sendo que a capacidade de cada máquina é de até 127 transações simultâneas. Se alguma nova transação for recebida quando a capacidade da fila chegar ao limite, a mesma será descartada. Este limite de processamento é independente do tipo de transação recebida.

Entre os diferentes tipos de transações que podem ser recebidas no sistema, podem ser destacadas cinco, devido à sua importância para o negócio, para o correto funcionamento do sistema e também em virtude do elevado número de ocorrência. Essas transações são as seguintes:

- Compra com cartão: transação de compra com cartão de crédito ou débito;
- Estorno: transação para cancelar uma compra com cartão já concluída;
- Reversão: transação disparada automaticamente dos meios de captura para restaurar saldo;
- Recarga de Telefonia: transação para efetuar recarga de telefonia em telefones pré-pagos;
- Administrativas: operações de parametrização e configuração dos terminais.

As demandas que essas transações geram ao sistema transacional podem ser vistas na Tabela 1, com os dados obtidos a partir dos rastros do sistema transacional da empresa GetNet. Estão

Figura 8: Fluxo transacional em um dia útil, indicando as variações de carga ao longo do dia.



Fonte: Rastro de sistema transacional da empresa GetNet.

destacados nesta tabela, o tamanho médio de cada transação, o quanto elas utilizam de memória, a quantidade de ciclos de máquina e o tempo de acesso aos sistemas internos. Considerando que uma transação pode acessar mais de um sistema interno, o tempo apresentado na coluna "tempo de acesso"corresponde à soma dos tempos necessários para acessar todos os sistemas que são necessários para efetuar a transação.

A comunicação entre as MP e o SI se dá através do protocolo XML e cada um destes subsistemas opera com máquinas redundantes de balanceamento de carga segundo o algoritmo RR. Os dados processados nos SI são retornados para as MP e então transmitidos para serem autorizados nas processadoras externas, através de conexões TCP/IP ou X25. A transação é autorizada ou negada conforme critérios de cada operadora.

Ao retornar da instituição autorizadora, a mensagem é submetida ao módulo de formatação da ISO, responsável pelas conversões de protocolo. A partir das MP, a mensagem é direcionada novamente para o SI correspondente, no qual será validada quanto à sua integridade e também serão atualizados os registros transacionais internos. A seguir a mensagem retorna do SI para o MP e é novamente adaptada para o devido protocolo estabelecido para comunicar com os terminais da empresa.

O tamanho destes pacotes é bastante variado, conforme se observa na Tabela 1, dependendo

Tabela 1: Principais tipos de transações e seus dados de utilização do sistema transacional.

Tipo	Tamanho médio (bytes)	RAM (bytes)	Ciclos de máquina (Mega)	Tempo de acesso (milissegundos)
Compra	1.320	12.300	107	1.140
Estorno	1.212	12.250	102	1.100
Reversão	348	1.400	83	400
Recarga de telefonia	1.146	12.000	42	1.150
Administrativas	6.700	7.000	30	1.350

Fonte: Desenvolvido pelo próprio autor.

basicamente do tipo de transação e dos dados que irão compor o comprovante de vendas em uma transação de compra, por exemplo. As operações administrativas contudo, normalmente apresentam tamanho muito superior, normalmente trafegam dados para configurar os terminais ou informações relativas ao fluxo de vendas do estabelecimento comercial.

O tempo médio das transações varia de acordo com o tipo de transação e o meio de captura em que se originou, bem como com a tecnologia da conexão. Tipicamente uma transação de compra com cartão ocorre em no máximo 3 segundos, enquanto de recarga de telefonia em torno de 6 segundos. As transações administrativas mantém o terminal conectado por um período muito superior às demais, devido ao fato de que a quantidade de dados que trafega na rede é muito mais elevada.

2.2 Escalonamento em Sistemas Distribuídos

Após a contextualização apresentada na seção anterior, esta seção tem como foco a análise de um sistema transacional visto como um sistema distribuído. O mesmo será analisado como um conjunto de computadores autônomos e interligados por uma rede de comunicação, possuindo um sistema operacional distribuído, responsável por coordenar as atividades desenvolvidas, além de permitir o compartilhamento dos recursos existentes no sistema.

O surgimento dos sistemas computacionais distribuídos pode ser atribuído, sobretudo, como uma consequência da necessidade de compartilhamento de recursos, normalmente de alto custo e fisicamente separados. Entre outros aspectos, que também contribuíram para estimular o interesse pela utilização de sistemas distribuídos, está a redução do custo dos microprocessadores, aliado ao constante aumento do poder de processamento e disponibilidade em conjunto com os avanços na tecnologia de comunicação de dados, que disponibilizou o uso de redes de computadores de alta velocidade.

Os sistemas computacionais distribuídos apresentam propriedades que podem torná-los atrativos quando comparados às máquinas paralelas ou aos sistemas centralizados: disponibilidade, confiabilidade, escalabilidade, tolerância a falhas, transparência, concorrência, compartilhamento de recursos e um possível aumento de desempenho (BRANCO, 2004). A heterogenei-

dade dos sistemas distribuídos constitui outra importante propriedade (ILIC; SOUSA, 2012). A heterogeneidade pode se dar, principalmente, com relação à configuração das máquinas, à arquitetura ou com relação ao sistema operacional. Quanto à configuração das máquinas, está relacionado às diferentes capacidades de processamento, espaço de memória ou armazenamento em disco, entre outros. A arquitetura heterogênea torna complexa a execução de um mesmo código em máquinas diferentes. A heterogeneidade do sistema operacional apresenta desafios na medida em que as propriedades dos sistemas nas diferentes máquinas se modificam, podendo ser incompatíveis (KON et al., 2000).

Explorar adequadamente o potencial disponibilizado pelo uso de plataformas distribuídas é foco de muitos estudos (BRANCO, 2004). Logo, os objetivos de escalonamento de processos em sistemas computacionais distribuídos são os seguintes:

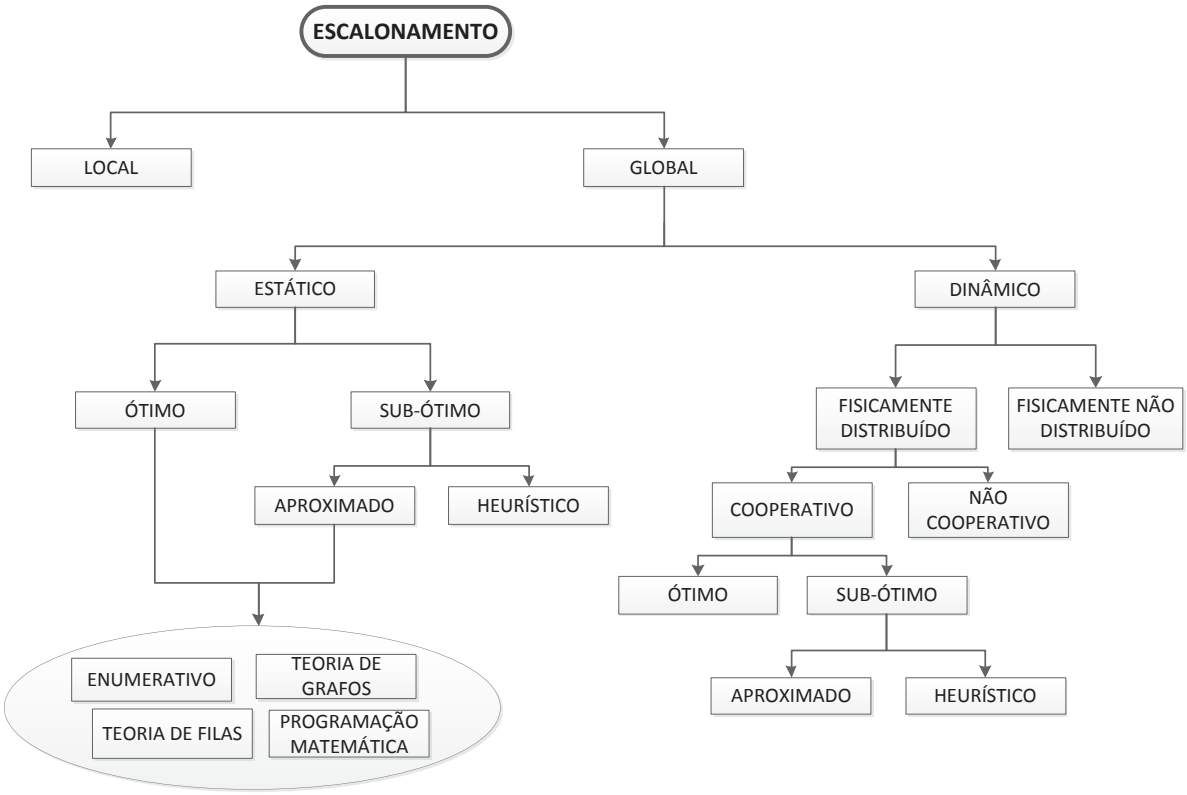
- **Minimizar:** tempo de execução e atrasos na comunicação;
- **Maximizar:** utilização dos recursos e *throughput* do sistema.

Considerando os objetivos destacados acima, normalmente uma proposta de escalonamento é avaliada por duas características: (i) o desempenho e, (ii) a eficiência. Desta forma, se a avaliação do escalonador tem como foco o tempo gasto na execução das políticas de escalonamento, quanto menor o seu valor, melhor o desempenho do escalonador. Por outro lado, se o foco é a eficiência, a análise recai nas políticas que são adotadas pelo escalonador e podem ser avaliadas com uso de funções de complexidade computacional (TSENG; CHIN; WANG, 2009).

A fim de contemplar o maior número possível de características existentes nos algoritmos de escalonamento, diversos autores têm sugerido taxonomias para a área de escalonamento de processos (CASAVANT; KUHL, 1988; CHENZHONG XU LAU, 1997; LULING; MONIEN; RAMME, 1991; LüLING; MONIEN, 1993; SHIRAZI et al., 1995). Dentre as diversas taxonomias propostas destaca-se a de Casavant (CASAVANT; KUHL, 1988), por ser mais abrangente e de grande aceitação. A estratégia proposta por Casavant é apresentada na Figura 9. A primeira divisão dada por essa taxonomia compreende escalonamento local ou global. O escalonamento local determina como um dado processo em uma única CPU é alocado e executado. Já uma política de escalonamento global usa informações do sistema para alocar processos para vários processadores, com o objetivo de otimizar seu desempenho como um todo, que é o caso do escalonamento em sistemas distribuídos e alvo desse estudo. O escalonamento global é dividido em duas abordagens: (i) estática e (ii) dinâmica. Essas dizem respeito ao momento no qual as decisões sobre o escalonamento serão tomadas.

No escalonamento estático, as decisões são tomadas antes da aplicação ser iniciada, ou seja, em tempo de compilação e permanecem constantes durante a execução do programa. Isso se dá na medida em que são conhecidas previamente as informações sobre os processadores, o tempo de execução das tarefas, o tamanho dos dados, o padrão de comunicação e a relação de dependência entre as tarefas. Essa abordagem é simples de ser implementada, no entanto, se verificam duas grandes desvantagens (LU; SUBRATA; ZOMAYA, 2006). Primeiramente,

Figura 9: Taxonomia de escalonamento proposta por Casavant e Kuhl.



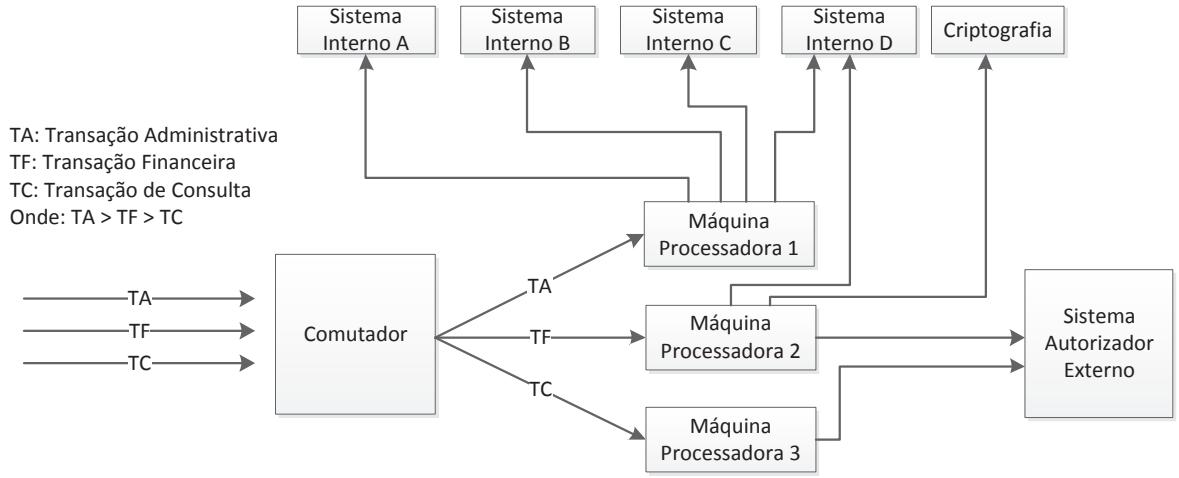
Fonte: Casavant (1988).

a distribuição da carga de trabalho e o comportamento de muitas aplicações não podem ser previstos antes da execução do programa. Em segundo lugar, o escalonamento estático supõe que as características dos recursos computacionais e da rede de comunicação são conhecidas com antecedência e permanecem constantes.

Escalonamento dinâmico trabalha com o conceito de que pouco, ou nada, se conhece previamente sobre as necessidades e o comportamento do aplicativo (LI; LAN, 2004; LU; SUBRATA; ZOMAYA, 2006; WANG et al., 2007; WARAICH, 2008). Também se desconhece o ambiente em que o processo será executado durante sua existência. A chegada de novas tarefas, a relação entre elas e os dados da arquitetura alvo são imprevisíveis, assim como a decisão do mapeamento consumidores-recursos é tomada com o ambiente em execução. Seguindo a interpretação da Figura 9, o encargo do escalonamento global pode ser atribuído tanto a um único processador (fisicamente não distribuído) ou desempenhada por um conjunto de processadores (fisicamente distribuído). Dentro da esfera desta última classificação, a taxonomia pode também distinguir entre os mecanismos que envolvem a cooperação entre os componentes distribuídos (cooperativa) e aqueles em que os processadores individuais tomam decisões independentes das ações dos demais processadores (não cooperativa). No modo cooperativo, cada processador tem a responsabilidade de realizar a sua própria parte do escalonamento, porém todos os processadores

trabalham em direção a um objetivo comum de todo o sistema.

Figura 10: Dinamicidade de um Sistema Transacional.



Fonte: Desenvolvido pelo próprio autor.

Nesse contexto, para o sistema transacional em estudo, no qual as demandas estão associadas aos hábitos dos consumidores, as cargas oscilam entre instantes de pico de venda, que exigem alto consumo de processamento das máquinas e entre períodos de quase ociosidade do sistema, durante as madrugadas. Assim como as quantidades de transações recebidas, também os tipos de transações chegam no sistema aleatoriamente e, portanto, levam a incertezas quanto ao tamanho dos pacotes e de quais recursos de outros subsistemas internos precisam ser acionados, conforme representado na Figura 10.

Na Figura 10 estão representadas três tipos de transações diferentes, cada uma com seu respectivo tamanho e necessidades de acesso aos sistemas internos. Considerando que o chaveador pode distribuir as transações em qualquer máquina processadora que estiver disponível, todas precisam estar capacitadas para atender à demanda. De acordo com o fluxo ilustrado, uma Transação Administrativa (TA) é a que tem maior quantidade de dados e também a que mais requer acesso aos Sistemas Internos (SI). Uma Transação Financeira (TF) possui um tamanho menor que TA e acessa diferentes SI e também Sistemas Externos. Por fim, uma Transação de Consulta (TC) consome ainda menos recursos do sistema, e possui o menor tamanho de pacote de dados. Além destas características, também se deve destacar que as transações são recebidas aleatoriamente, ou seja, a ordem e os períodos de chegada são desconhecidos.

Dada esta irregularidade no fluxo transacional, que não pode ser determinado e, portanto, é desconhecido previamente, também o consumo de CPU, da memória e a ocupação da rede são desconhecidos. Desse modo, o Sistema Transacional pode ser enquadrado como um sistema dinâmico, dada a aleatoriedade das demandas.

Dando continuidade à análise da taxonomia de Casavant, para um escalonamento cooperativo, as soluções podem ser ótimas ou sub-ótimas. Para se obter soluções ótimas, há uma

enorme dificuldade na medida em que problemas de escalonamento são NP-hard (KRISHNA; GANESHAN; RAM, 1995), dada a dificuldade do ambiente distribuído de assumir premissas para prover a situação ótima de um algoritmo. Algoritmos de escalonamento ótimos são computacionalmente caros, chegando ao ponto de serem inviáveis. Por isso, as pesquisas nesta área procuram por estratégias de solução sub-ótimas que podem ser divididas em aproximadas ou heurísticas. Algoritmos aproximados usam modelos formais de computação que, ao invés de buscarem uma solução ótima, ficam satisfeitos com uma solução suficientemente boa. Já os algoritmos heurísticos representam a classe de algoritmos que assumem as hipóteses mais realistas possíveis sobre o conhecimento existente e as características de carga do sistema.

Além das classificações já vistas, Casavant também apresenta uma classificação horizontal para o escalonamento. As características descritas na classificação horizontal podem ser adequadas em todos os ramos da classificação hierárquica. Neste contexto, o escalonamento pode ser visto como adaptativo e não-adaptativo. Uma solução adaptativa para o problema do escalonamento é aquela em que os algoritmos e os parâmetros utilizados para implementar a política de escalonamento mudam dinamicamente de acordo com o comportamento do sistema. Em contraste, uma política não-adaptativa não modifica seu mecanismo de controle de escalonamento em função do comportamento do sistema.

Dadas as características deste sistema, e os dados que se dispõem para as políticas de escalonamento, este trabalho irá utilizar o método adaptativo, através de uma análise periódica de parâmetros como consumo de CPU, memória e rede, entre outros. Isso possibilitará uma política de balanceamento mais otimizada para distribuição das capacidades dos recursos entre os consumidores.

2.3 Balanceamento de Carga

Enquanto o escalonamento de processos tem como foco aumentar o desempenho geral em sistemas distribuídos e melhorar o uso dos recursos computacionais disponíveis (WANG; MORRIS, 1985; CASAVANT; KUHL, 1988; SHIVARATRI; KRUEGER; SINGHAL, 1992; EL-REWINI; ALI; LEWIS, 1995; BECKER, 1995; BERMAN; CHARIKAR; KARPINSKI, 1997), o balanceamento de carga visa distribuir uniformemente os recursos compartilhados, evitando a situação em que um elemento de processamento esteja sobrecarregado enquanto outro está com baixa carga de processamento. O desempenho de sistemas distribuídos é fortemente influenciado pela distribuição de carga entre os elementos de processamento do sistema (EL KABBANY et al., 2011). Dentre as razões para o desequilíbrio de carga, pode-se citar a insuficiência de máquinas para processamento, a ineficiência na distribuição de tarefas entre processadores ou a desigualdade no tamanho das tarefas. Esse desequilíbrio reduz o desempenho global dos sistemas e, para melhorá-los, a carga de trabalho deve ser redistribuída entre todos os processadores.

O fato de que existe uma quantidade de poder de processamento que não é usado de forma

eficiente, constitui a principal razão para adoção de balanceamento de carga, sobretudo em ambientes dinâmicos e heterogêneos, tal como o sistema transacional que está sendo abordado. Nesse contexto, políticas de escalonamento podem utilizar mecanismos de balanceamento de carga com o objetivo de (SINHA, 1996): (i) otimizar uso de todos os recursos; (ii) aumentar o desempenho total do sistema; (iii) minimizar atrasos de comunicação, uma vez que processadores sobrecarregados tendem a demorar mais tempo para atender e processar as requisições pendentes. Para tanto, é necessário determinar a medida de carga (métrica que quantifica a utilização de um elemento do sistema) (LAWRENCE, 1998). Existem várias medidas de carga possíveis, incluindo: (i) o número de tarefas em uma fila; (ii) a carga média da CPU; (iii) a utilização da CPU em determinado momento; (iv) o percentual de CPU disponível; (v) a quantidade de memória livre; (vi) a quantidade de comunicação entre os processos, além da combinação possível desses indicadores. Dessa forma, tais medidas podem influenciar na decisão sobre quando acionar o balanceamento de carga, quais processos estarão envolvidos e para onde eles serão transferidos.

Técnicas de balanceamento de carga são instrumentos das políticas de escalonamento na busca de melhores resultados, desse modo é natural que se tenham classificações estáticas e dinâmicas para balanceamento de carga (WARAICH, 2008; EL KABBANY et al., 2011). O balanceamento de carga estático se dá na medida em que se conhecem os dados em tempo de compilação, ao passo que o balanceamento dinâmico se vale da coleta de informações durante a execução da aplicação. Conforme visto em seções anteriores, e aprofundando na Seção 2.1 com uma representação na Figura 10, um sistema de transações financeiras se caracteriza pelo dinamismo e imprevisibilidade da carga recebida. Portanto, a escolha por técnicas de balanceamento dinâmicas é a mais apropriada para ser aplicada nesta dissertação.

2.3.1 Decisões Tradicionais de Balanceamento de Carga

Conforme as estratégias propostas por Waraich (2008), existem três principais parâmetros que normalmente definem a estratégia acerca de qual algoritmo de balanceamento de carga se deve empregar. Segundo essa afirmativa, esses parâmetros são “quem” fará o balanceamento de carga, “qual” informação será utilizada e “onde” o balanceamento de carga será feito.

Para definir “quem” tomará as decisões do balanceamento, primeiramente é preciso estabelecer se a política a ser empregada é do tipo *sender-initiated* ou *receiver-initiated*. Estas políticas são para compartilhar a carga entre processadores fortemente carregados. Com a política *sender-initiated*, os nós mais congestionados tentam mover suas tarefas para nós com menor carga enquanto na política *receiver-initiated*, os nós com menor carga procuram por nós mais carregados para receber tarefas (WARAICH, 2008). Além destes, também se pode adotar um terceiro modo, que resulta da combinação destes dois. Esta terceira política é do tipo *symmetrically-initiated*, na qual tanto as máquinas sobrecarregadas quanto as com menor carga podem disparar a atividade de distribuição de carga. Estudos mostram que, em sistemas com

baixa carga de trabalho, algoritmos *sender-initiated* têm mais sucesso em encontrar máquinas com pouca carga e em sistemas com alta carga de trabalho, algoritmos *receiver-initiated* têm mais sucesso em encontrar máquinas sobrecarregadas (SHIVARATRI; KRUEGER; SINGHAL, 1992).

Conforme será explorado mais adiante, na Seção 4.2, as máquinas processadoras serão periodicamente monitoradas, contudo em situações de sobrecarga serão transmitidas notificações para o escalonador, caracterizando uma política *receiver-initiated*. Esta política evita perda de transações em máquinas que não tenham condições de atender as necessidades de processamento próprias de uma transação financeira, conforme visto na Seção 2.1.

Para responder à questão de "qual" informação será utilizada para fazer o balanceamento, primeiramente é preciso definir se será adotada a estratégia local ou global. Nas políticas globais, o balanceador de carga utiliza o perfil do desempenho dos processos. Nas políticas locais, os processos são divididos em grupos. Embora a troca de informações locais origine menos custo de comunicação, estratégias de informações globais tendem a fornecer decisões mais precisas. A escolha pela política global ou local depende do comportamento que a aplicação apresenta. Para esquemas globais, a convergência do balanceamento é mais rápida que em locais, desde que todos os processos sejam considerados ao mesmo tempo. Entretanto, isso requer mais sincronização e comunicação entre eles (WARAICH, 2008).

A definição de "onde" será feito o balanceamento da carga é dado de acordo com a escolha entre as estratégias centralizada ou distribuída. Na forma centralizada, o balanceador de carga é situado em um processo mestre, no qual todas as decisões são tomadas. No modo distribuído, o balanceador de carga é replicado em todos os processos.

2.3.2 Políticas de Balanceamento de Carga

Algoritmos de balanceamento de carga buscam atingir um conjunto de objetivos relacionados com medidas de desempenho, como por exemplo: (i) otimizar a ocupação de recursos e (ii) diminuir o tempo médio de resposta dos processos (MELLO et al., 2006). A heterogeneidade dos recursos existentes, nas diversas máquinas de um sistema computacional, leva à possibilidade de se ter uma grande variação em suas capacidades computacionais. Essa variação, aliada à possibilidade de utilização de diferentes arquiteturas, introduz dificuldades que precisam ser gerenciadas para o balanceamento de cargas, visando à obtenção de um melhor desempenho (BRANCO, 2004).

Esse gerenciamento de recursos, de uma determinada rede de computadores, é possível através da implementação de mecanismos e políticas de balanceamento de cargas (SONG; CHOO; LEE, 1997) (SHIVARATRI; KRUEGER; SINGHAL, 1992).

Em Kwok e Cheung (2004), as políticas de balanceamento de carga pode ser analisadas em quatro dimensões: (i) política de localização; (ii) política de informação; (iii) política da transferência e; (iv) política de seleção. Estas políticas e suas definições, são apresentadas na

Tabela 2: Quatro dimensões para balanceamento de carga.

Política	Atribuições
Localização	Localizar uma máquina parceira de transferência (emissora ou receptora) adequada para uma outra máquina, assim que a política tenha decidido que esta máquina é emissora ou receptora. Normalmente as políticas de localização incluem: seleção aleatória, seleção dinâmica e sondagem.
Informação	Determinar quando as informações sobre o estado das outras máquinas deve ser coletado e qual a informação deve ser coletada. As abordagens mais comuns para essa política são: orientada à demanda, orientada à mudança de estado e periódica.
Transferência	Determinar se uma máquina está apta para participar de uma transferência de tarefas como transmissora ou receptora.
Seleção	Determinar qual tarefa deve ser transferida. A política de seleção pode ser dividida em: (i) preemptiva e (ii) não preemptiva, sendo que a primeira permite a migração de processos que estão sendo executados, enquanto que a segunda não.

Fonte: Kwok e Cheung (2004).

Tabela 2.

Aplicando tais políticas ao sistema transacional, se pode destacar a importância da política de informação, haja visto que as questões pertinentes à periodicidade de coleta e qualidade podem ser decisivos quanto à efetividade e ao desempenho do balanceador. A partir dessa constatação, a estratégia é definir o intervalo para a troca de informações. Um intervalo muito elevado pode levar à imprecisão na tomada de decisões, porém se for adotado um intervalo muito curto pode ser gerada sobrecarga de comunicação desnecessária. Geralmente as característica da aplicação definem a escolha do intervalo. Na estratégia aperiódica, por outro lado, um processador irá transmitir suas informações de carga sempre que a mesma variar. Essa estratégia favorece uma visão global do estado do sistema. No entanto, caso muitos eventos sejam desencadeados, pode-se gerar uma alta sobrecarga de comunicação. Desse modo, a estratégia aperiódica pode ser impraticável para sistemas distribuídos muito grandes. Em suma, uma estratégia de troca de informações deve prover uma visão precisa do estado do sistema e, ao mesmo tempo, mantenha uma baixa sobrecarga de comunicação.

2.4 Modelagem Formal de Sistemas Transacionais

Ao analisar um sistema de transações financeiras, conforme a Seção 2.1, se observa que para uma transação ocorrer com sucesso há uma série de controles que garantem a integridade das informações antes de proceder, de fato, com a solicitação de débito do cartão. Cada um dos sistemas internos envolvidos nessa dinâmica atua sobre cada transação e influenciará, diretamente, no desempenho do processo como um todo. Conforme citado no referencial teórico, as máquinas processadoras realizam uma uniformização dos diferentes protocolos oriundos de

meios de captura distintos, tais como POS, TEF e ATM. Todos os demais módulos do sistema são diretamente dependentes do correto serviço prestado na entrada. Dessa forma, esse ponto se caracteriza por elevada demanda e alta criticidade.

Diante deste cenário, é importante dimensionar adequadamente o número de máquinas processadoras que serão usadas para o tratamento de requisições, sem incorrer no erro de superestimar o número de máquinas processadoras empiricamente. Uma estimativa empírica garantiria o atendimento pleno de transações, contudo resultaria em maior custo financeiro e energético para o centro de processamento.

Ao longo das pesquisas que foram realizadas para se chegar ao modelo final de GetLB, pode ser destacado o trabalho relacionado à simulação computacional (ANDRADE et al., 2013). A simulação de um protótipo que fosse capaz de auxiliar a mensurar a capacidade de um sistema transacional e orientar na quantificação dos recursos, fez parte das pesquisas de GetLB. O resultado dessa pesquisa foi a criação de um fluxograma com um conjunto de passos para a definição formal da quantidade de máquinas processadoras necessárias em função da demanda em um sistema de transações eletrônicas.

Para modelagem de um sistema desta natureza é preciso considerar a existência de horários de pico de demanda. Consequentemente, ocorrem variações no fluxo de entrada no transcorrer do período de serviço, caracterizando um sistema que não está em equilíbrio. Com isso, não é possível empregar os modelos matemáticos da Teoria de Filas para analisar o sistema (JAIN, 1991).

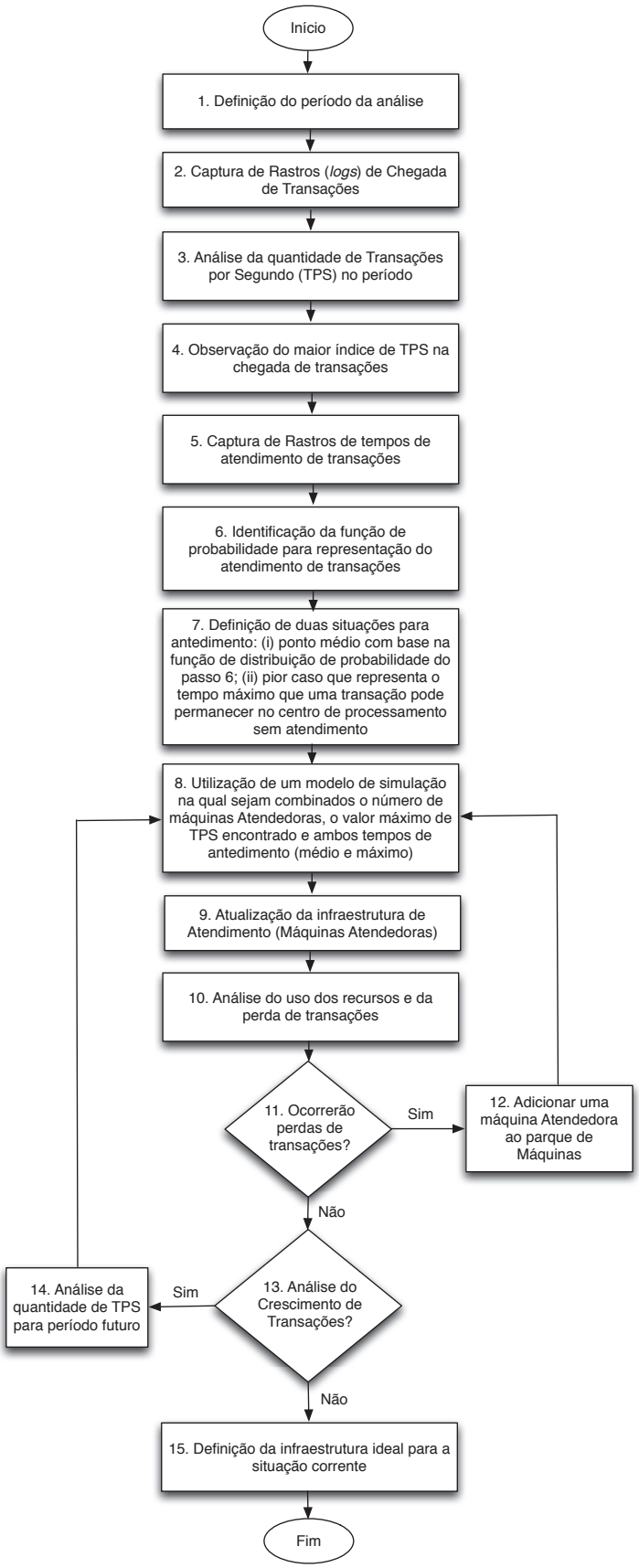
A Figura 11 apresenta o fluxograma para a análise da quantidade de máquinas processadoras necessárias em um sistema transacional. A análise começa pela definição do período que o sistema será avaliado preferencialmente, com amostras coletadas nos últimos dias do ano visto que tal período, geralmente, representa um maior volume de negócios (EDWARDS, 2010; GIRARDI; CAMARGO, 2009). Quanto ao período, é pertinente a verificação do sistema durante 24 horas e a determinação dos picos nesse intervalo. Na sequência, o fluxograma segue com as ações de captura dos dados relevantes para o processamento de transações, como os eventos de chegada e atendimento de transações. Esses dados podem ser viabilizados pela técnica de *profiling* no chaveador de entrada de transações e em cada uma das máquinas processadoras.

Uma etapa importante deste processo de análise, corresponde à captura do tempo de atendimento de cada uma das transações durante o período avaliado, conforme o passo 5 do fluxograma. Nesse ponto, adota-se um intervalo para verificar a frequência relativa de cada tempo de atendimento. Com auxílio de um *software* com recursos para a realização de testes de aderência, o EasyFit[1], se verifica a distribuição de probabilidade e o tempo médio de atendimento. Além disso, é comum em sistemas de processamento de transações, o estabelecimento de um limite máximo de tempo que uma transação pode permanecer no sistema. Nessa linha, testes com o pior caso de atendimento conduzem a uma configuração diferente no parque de máquinas.

Outras etapas relevantes do fluxograma são a modelagem do cenário transacional e sua

[1] http://www.mathwave.com

Figura 11: Fluxograma para análise das máquinas processadoras num centro de processamento de transações eletrônicas.



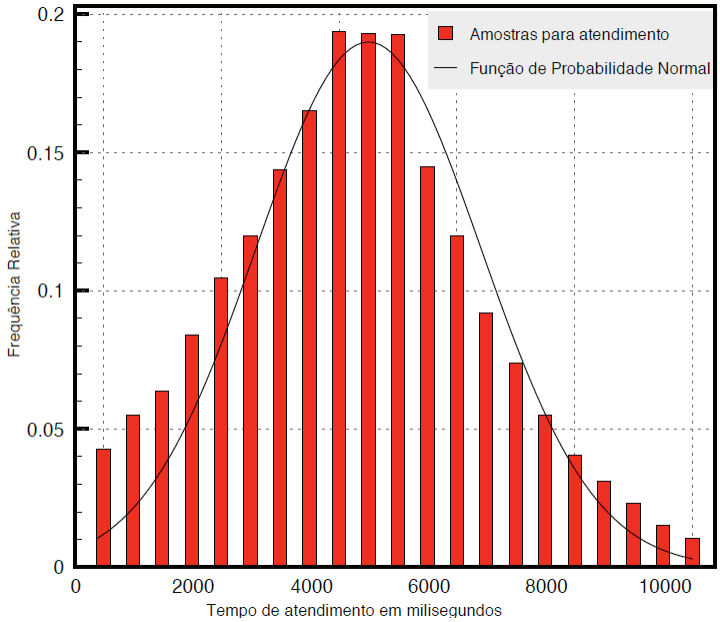
Fonte: Desenvolvido pelo próprio autor.

execução através do emprego de um modelo de simulação computacional, conforme passo 8. A simulação resulta no percentual de utilização dos recursos e de ocorrência de perdas perante o total de transações submetidas, fazendo uma validação de necessidade de reconfiguração do modelo transacional.

Através de dados reais obtidos junto à empresa GetNet, o fluxograma foi validado, utilizando informações de transações de cartão de débito/crédito e recarga de telefonia pré-paga em um dia de movimento no período de 08h00min até 22h00min, o qual representa aproximadamente 92% do volume total de um dia de trabalho. O restante do período foi descartado, pois apresenta um volume muito baixo de transações.

A GetNet opera, atualmente, com um chaveador que recebe transações e distribui entre 10 máquinas processadoras homogêneas. Quanto ao valor máximo de TPS, foi adotado o valor 43 para os testes, uma vez que representa o valor máximo dentro do intervalo considerado. Na sequência, foi feita a modelagem para a estimativa da distribuição de probabilidade teórica que melhor descreve os tempos de serviço. Com uso do EasyFit, a função densidade de probabilidade indicada foi do tipo Normal com tempo médio de atendimento de 4,798 segundos e desvio padrão de 2,1194 segundos, conforme Figura 12. Os valores obtidos através dos testes de aderência Chi-quadrado, Kolmogorov-Smirnov e Anderson-Darling (JAIN, 1991), não rejeitaram a hipótese de que a função teórica N(4,798;2,1194) seja a mais adequada para representar os tempos de serviço.

Figura 12: Distribuição de probabilidade Normal N(4,798;2,119) para o atendimento de transações.



Fonte: Desenvolvido pelo próprio autor.

Obtidos os modelos de entrada e processamento das transações, foi feita a análise de desempenho do sistema para diferentes situações de capacidade de processamento: (i) função normal

de probabilidade com média de 4,798 segundos; (ii) pior caso ou exceção, sendo este o tempo fixo de 28 segundos, que é o limite adotado pela empresa para que uma transação seja processada. A simulação foi feita com o auxílio do *software* Rockwell Arena Simulator[2] (KELTON; SADOWSKI; STURROCK, 2007).

Avaliando os resultados dos ensaios, se verificou que em regime normal de processamento, a empresa poderia contar com somente 4 das 10 máquinas disponibilizadas. Contudo, no pior caso, a infraestrutura corrente opera no limite da sua capacidade, necessitando adicionar 1, 2, 5, 7 e 10 máquinas processadoras, a cada ano, para oferecer um cenário sem perda de transações, assumindo o período de 2012 até 2016.

2.5 Análise do Capítulo

Neste capítulo foram descritos alguns princípios básicos, porém pertinentes para balanceamento de carga em um sistema de transações eletrônicas financeiras. Na primeira seção, se buscou apresentar todos os sistemas envolvidos e o extenso fluxo de uma transação durante o período de aproximadamente 3 segundos, que compreende o instante em que o usuário passa seu cartão até o momento da impressão do comprovante de venda.

Através de uma análise do sistema transacional da empresa GetNet, se verificou a grande quantidade e heterogeneidade dos sistemas que estão envolvidos no processo de tratamento de transações eletrônicas. Para um sistema com tais características advém grandes desafios de integração e, portanto, também de sincronia de todos os recursos envolvidos. A busca de um sistema transacional que mantenha seus recursos operando em equilíbrio e focado em alta disponibilidade e alto desempenho, será abordada nas seções seguintes, através da análise de questões de escalonamento e balanceamento de carga.

Após uma breve contextualização sobre sistemas computacionais distribuídos, ao longo da Seção 2.2 foram exploradas as características e as estratégias dadas pela taxonomia de Casavant. Neste contexto, para um sistema global, foram apresentadas as diferenças entre um sistema (i) estático e (ii) dinâmico, sendo o sistema transacional em estudo fortemente voltado para este último, dada a aleatoriedade dos tempos de chegadas e dos diferentes tipos de transações. Seguindo na análise das soluções para um tipo de sistema cooperativo, as mesmas podem ser ótimas ou sub-ótimas. As soluções ótimas por vezes se mostram impraticáveis, na medida em que o problema em estudo é do tipo NP-Hard. As soluções sub-ótimas podem ser atendidas por métodos de aproximação ou heurísticas, sendo que, por aproximação, se busca uma solução que seja suficientemente boa e, na heurística, se assumem as hipóteses mais realistas possíveis a partir de um conhecimento prévio da dinâmica do sistema como um todo.

Na Seção 2.3, se analisou o balanceamento de carga focando nas técnicas para obtenção de melhores resultados. Com base em afirmações de Waraich (2008), três principais parâmetros de um sistema precisam ser identificados. Estes parâmetros são (i) quem, (ii) qual e (iii)

[2] http://www.arenasimulation.com/

onde, ou seja, quem fará o balanceamento, qual informação será utilizada para isso e onde será feito o balanceamento. Complementando esta análise, foram apresentadas quatro políticas de balanceamento de carga segundo Kwok e Cheung (2004). A primeira é de localização, para identificar máquinas parceiras para transferência de carga. A segunda é de informação, que trata da periodicidade de coleta e tipo de informação. A terceira é de transferência, ou seja, se a máquina é apta para transmitir ou receber tarefas. A quarta política é de seleção, na qual se trata de determinar qual tarefa será transferida. Finalizando a seção, foi abordado o problema dos intervalos de coletas de informação no sistema, o qual resulta de uma combinação para obter uma estratégia que proporcione uma visão precisa do estado do sistema e, ao mesmo tempo, manter uma baixa sobrecarga de comunicação.

A modelagem de sistemas constitui uma ferramenta de grande importância na medida em que possibilita explorar métodos heurísticos mais adequados para o sistema. Também através desta modelagem é possível simular situações que seriam impraticáveis em um sistema real. Por meio de simulação, diferentes métodos de escalonamento podem ser validados antes de serem aplicados na arquitetura final. Isso possibilita grande redução nos custos envolvidos em um projeto bem como no tempo que seria necessário para efetuar os testes de várias situações que sejam condizentes com a realidade.

Este capítulo se encerra após a apresentação dos conceitos relacionados ao escalonamento e balanceamento de carga em sistemas distribuídos. No próximo capítulo esses conceitos serão aplicados em projetos e protótipos existentes no estado da arte. São projetos relacionados ao tema da dissertação e que visam apresentar procedimentos e técnicas para escalonamento e balanceamento de carga.

3 TRABALHOS RELACIONADOS

Esse capítulo descreve alguns trabalhos relacionados que estão inseridos no contexto da presente dissertação. Embora seja muito escasso o material relacionado ao balanceamento de carga em sistemas distribuídos voltados para transações financeiras, foram selecionados alguns trabalhos que abordam os métodos que serão utilizados nesta dissertação. Também serão apresentados sistemas que necessitam de balanceamento de carga e poderiam se beneficiar da arquitetura desenvolvida para a dissertação.

Iniciando com a Seção 3.1, são apresentados trabalhos que analisam alguns algoritmos conhecidos para balanceamento, fazendo um comparativo de acordo com diferentes estratégias de balanceamento. Ainda nesta seção, buscando associar os métodos dos trabalhos estudados com os que serão utilizados na dissertação, se apresenta um método de escalonamento com base em dados obtidos para os estados de CPU, consumo de memória, fila de tarefas e de rede.

Na Seção 3.2, os aspectos de balanceamento de carga em *Cloud Computing* são abordados através de trabalhos que destacam os desafios comuns entre o que está sendo pesquisado nesta dissertação e os encontrados nesta área da computação, tais como escalonamento de máquinas virtuais em máquinas físicas. Ainda nesta seção, será analisado o fato da computação em nuvem se tratar de um sistema distribuído que utiliza técnicas de escalonamento, balanceamento de carga.

Na Seção 3.3 são apresentados trabalhos voltados para análise de desempenho de um sistema de transações financeiras e as características do sistema, suas principais funcionalidades e necessidades serão apresentadas. Encerrando o capítulo, na Seção 3.4 é realizado um fechamento dos assuntos abordados, com uma análise dos trabalhos apresentados e a relação dos mesmos com o trabalho desta dissertação.

3.1 Escalonamento e Balanceamento de Carga

Nesta seção serão apresentados trabalhos com análise comparativa entre diferentes algoritmos e diferentes técnicas de balanceamento que podem ser aplicadas em sistemas com máquinas virtuais, através de realocação de recursos.

3.1.1 Avaliação de diferentes algoritmos de balanceamento de carga

No trabalho *Evaluation of Load Balance Algorithms* (MCHEICK; MOHAMMED; LAKISS, 2011), os autores destacam os estudos que vem sendo feitos para diferentes algoritmos de balanceamento de carga. Neste trabalho, os autores explicam como avaliar cada proposta, destacando o parâmetro de desempenho. Contudo, afirmam que outros parâmetros precisam ser considerados, tais como migração de processos, *overhead*, escalabilidade e disponibilidade. Desse modo, apresentam um trabalho voltado para a análise de algoritmos de balanceamento,

conforme os parâmetros citados, e analisam os desafios de escolha de um algoritmo.

Ao tratar do escalonamento global, os autores afirmam que os algoritmos devem ser avaliados por três critérios:

- A qualidade da solução produzida, medida pelo tempo por iteração no escalonador;
- O tempo despendido para efetuar o balanceamento de carga, medido pelo tempo para resolver problemas conhecidos;
- A portabilidade para diferentes tipos de aplicações com diferentes tipos de tarefas e o número de parâmetros que precisam ser configurados para obter o desempenho ótimo.

Analisando o balanceamento estático, os autores expõem que os algoritmos apenas operam bem quando não há muita variação de carga nas estações de trabalho e, de fato, não são bem aceitos mesmo em um ambiente conhecido caso não se conheça, com precisão, a carga de trabalho para um escalonador pervasivo, que esteja no controle de todas as atividades de distribuição da carga (MALIK, 2000).

Ao considerar que as principais metas deste tipo de escalonamento são: (i) minimizar o tempo total de execução através de redução dos atrasos de comunicação, (ii) simplificar a implementação e (iii) não necessitar de monitoração constante das estações de trabalho, os autores recomendam os seguintes algoritmos: Round Robin, Algoritmo Randômico, Bisseção Recursiva, Eigen Bisseção Recursiva e Algoritmo Genético.

Para o balanceamento de carga dinâmico, no qual ocorrem mudanças de distribuição de carga em tempo de execução através do uso de informações de carga atualizadas, se tem como consequência uma alta demanda de comunicação. Portanto, é necessário manter sempre os custos de *overhead* dentro de limites aceitáveis. Como resultado das técnicas deste tipo de balanceamento, há um significativo aumento no desempenho comparado ao balanceamento estático. Nesse contexto, o trabalho de Mcheick et. al. (2011) enfatiza a necessidade de responder às questões pertinentes a "quem" fará o balanceamento de carga, "qual" informação será utilizada e "onde" o balanceamento de carga será feito, conforme detalhado na Seção 2.3. Com essas respostas, se pode definir o tipo de escalonamento mais adequado. Ao responder "quem" fará o balanceamento, é possível decidir se será *sender-initiated* ou *receiver-initiated*. Com base na resposta de "qual" informação será utilizada, se pode definir sobre um balanceamento global ou local. E ao responder "onde", se estabelece se será centralizado ou distribuído.

De acordo com Mcheick et. al. (2011), duas estratégias podem ser adotadas neste tipo de balanceamento:

- Centralizado: utiliza uma abordagem mestre-escravo. O mestre mantém uma fila de tarefas e um escravo sinaliza ao mestre quando estiver pronto para receber uma nova tarefa;
- Totalmente distribuído: cada máquina processadora mantém uma fila local de tarefas para escalonar e executar localmente se possível. Caso a carga local seja muito baixa ou muito

alta para ser processada, deve ser solicitada mais carga para o sistema ou encaminhar tarefas para máquinas com menos carga, respectivamente.

Quanto aos desafios do balanceamento dinâmico, o artigo de Mcheick et. al. (2011) aponta que para obter um rebalanceamento de carga com velocidade e escalabilidade, o algoritmo deve não apenas identificar o que deve migrar, mas deve manter a quantidade de dados requeridos no menor nível possível (HU; BLAKE, 2001). Essa medida não apenas torna a execução do algoritmo do escalonador mais rápida, como também contribui na redução do *overhead* de comunicação, necessário para atualização dos dados de escalonamento.

Seguindo a análise dos algoritmos de balanceamento de Mcheick et. al. (2011), os autores descrevem quatro parâmetros necessários para analisar o desempenho destes algoritmos, conforme listado a seguir:

i. *Migração de processos*: executado por um algoritmo capaz de decidir se é necessário ou não fazer alteração de carga durante a execução do processo. Para isso, observa-se que devido às características de balanceamento estático e dinâmico, a migração de carga nos algoritmos dinâmicos apresenta melhores resultados;

ii. *Overhead*: corresponde ao percentual de uso do processador, que consiste basicamente em *overhead* de CPU para filtrar e transferir tarefas. Uma parcela deste percentual de *overhead* contribui para a latência. No balanceamento estático, quando os processadores recebem tarefas pequenas e essas tarefas possuem alta dependência de outras sub-tarefas, o algoritmo utiliza alto *overhead* de transferência. Contudo, no balanceamento dinâmico, não é necessário decompor pequenas tarefas se a unidade de processamento não tem carga de trabalho. Assim o balanceamento dinâmico ocasiona menos *overhead* de CPU que o balanceamento estático;

iii. *Escalabilidade*: do ponto de vista da escalabilidade da carga, é melhor em algoritmos estáticos que nos dinâmicos, pois o estático decompõe as tarefas antes do tempo de execução (GRANOVSKY, 2012). Aumentando o tamanho das tarefas, o balanceamento de carga estática também é melhor do que o dinâmico;

iv. *Disponibilidade*: É a probabilidade de um sistema, em dado instante de tempo, ser operacional e capaz de fornecer os serviços requeridos (AVIZIENIS et al., 2004). Assim, o balanceamento dinâmico apresenta melhores resultados pois, se alguma máquina ficar indisponível, as tarefas podem ser alocadas para serem processadas em outra que esteja disponível, ao passo que no estático o balanceamento precisaria ser reconfigurado para que uma tarefa não seja enviada para um processador indisponível.

Um resumo das considerações e propostas do artigo *Evaluation of Load Balance Algorithms* (MCHEICK; MOHAMMED; LAKISS, 2011), estão agrupadas em duas tabelas. A primeira, Tabela 3, apresenta uma comparação entre algoritmos de balanceamento estático e dinâmico

conforme os parâmetros de desempenho descritos acima. Na Tabela 4 estão alguns dos algoritmos abordados no trabalho, sendo comparados segundo os mesmos parâmetros de desempenho.

A partir dessa compilação de dados, se verificou que os algoritmos de escalonamentos distribuídos e centralizados oferecem um alto suporte para migração de processos e disponibilidade, pois estes testam os mestres ou escravos durante a execução dos processos. Os algoritmos de Round Robin e Randômico oferecem alto suporte para a escalabilidade porque, aumentando o número de máquinas processadoras, levam a uma distribuição melhor do aumento de tarefas, diminuindo o desbalanceamento do sistema distribuído. Quanto ao algoritmo genético, ele apresenta maior *overhead* que os demais algoritmos pois necessita de tarefas codificadas para os cromossomos e para executar as tarefas de mapeamento e decisões de distribuição.

Tabela 3: Comparação entre algoritmos Estático e Dinâmico.

Parâmetro	Estático	Dinâmico
Migração de Processos	Limitado	Melhor
Overhead	Alto	Melhor
Escalabilidade	Melhor	Limitado
Disponibilidade	Limitado	Melhor

Fonte: Mcheick et. al. (2011).

3.1.2 Balanceamento dinâmico de recursos em máquinas virtuais

Outro trabalho relacionado ao balanceamento é apresentado por Zhou et. al. (2010), no qual um protótipo chamado VMCTune (*Virtual Machine Cluster Using Dynamic Resource Allocation*), foi projetado e implementado para balancear carga com base em uma política de alocação dinâmica de recursos em um *cluster* de máquinas virtuais, sendo executadas em um *cluster* de máquinas físicas através de para-virtualização. Nesse sistema, os recursos são monitorados e alterados em tempo real quanto ao consumo e uso de CPU, memória e rede. O sistema utiliza realocação das VMs dentro da mesma máquina física para obter um balanceamento de carga

Tabela 4: Comparação entre algoritmos de balanceamento, onde: 1- não atende, 2- atende parcialmente e 3- atende plenamente.

Parâmetros	Round Robin	Randômico	Genético	Centralizado	Distribuído
Migração de processos	1	1	1	2	3
Overhead	2	2	3	1	1
Escalabilidade	3	3	2	1	1
Disponibilidade	1	2	1	3	3

Fonte: Mcheick et. al. (2011).

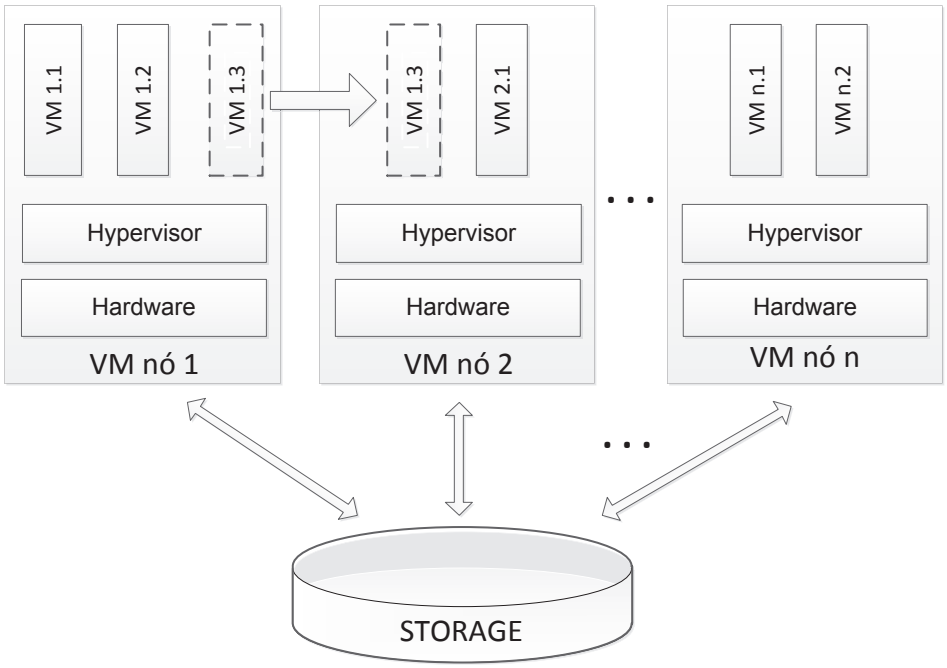
local, e utiliza migração de VMs entre as máquinas físicas adotando um balanceamento global.

Em sistemas em que as VMs estejam sendo executadas no modo para-virtualizado, nos quais se modificou o *kernel* para possibilitar acesso a instruções privilegiadas, alguns recursos podem ser controlados, de modo que o número de CPUs e a memória disponível possam ser alterados mesmo com a VM em operação.

Em alguns casos, um único nó pode não ser capaz de suportar a demanda de carga, sendo necessário que o sistema seja composto por múltiplos nós. Desse modo, para manter a carga balanceada, nas situações em que o sistema estiver desbalanceado, é preciso tratar da migração de VMs entre os nós, migrando máquinas alocadas em nós com alta carga para nós menos carregados, otimizando o uso dos recursos.

Este processo de balanceamento local e global está representado na Figura 13, na qual as VMs estão alocadas nos nós, os quais compartilham o mesmo *storage*. O procedimento de migração, demonstrado na figura, apresenta uma VM do nó 1 migrando para o nó 2 para otimizar a alocação dos recursos nos nós 1 e 2. O mais importante é que as VM continuam executando seus processos mesmo durante a migração, sem interrupção, o que é fundamental em sistemas que demandam serviços em regime de alta disponibilidade.

Figura 13: Sistema distribuído com máquinas virtuais e migração de recursos entre os nós.



Fonte: Zhou (2010).

O escalonador VMCTune é alocado em uma camada virtual do *cluster*, monitorando as VMs e distribuindo os recursos via *Hypervisors*. Por não acessar diretamente as VMs e aplicações, as operações ocorrem de modo transparente. O escalonador monitora periodicamente as máquinas físicas e também as VMs através dos *Hypervisors*.

Para elaboração do algoritmo de escalonamento foram separadas as demandas nas seguintes categorias: (i) alto consumo de CPU, (ii) alto consumo de memória, (iii) alto consumo de E/S, (iv) alto consumo de rede e (v) alto consumo de recursos combinados. As possibilidades de balanceamento deste algoritmo são para um escalonamento local, realocando carga dentro das VMs que estão em um mesmo nó, e para escalonamento global, migrando VMs entre os nós. Os passos do algoritmo para a tomada de decisão estão embasadas na coleta da carga de todas as VMs e máquinas físicas, análise dos dados coletados e execução da distribuição de carga entre as VMs dos nós ou realocar as VMs entre os nós.

Após a coleta das informações de carga das VMs e das máquinas físicas, o algoritmo verifica o estado de cada VM j alocada na máquina física i, atribuindo PM_NUM para o número de máquinas físicas ou nós e VM_NUM_i para o número de VMs alocadas em PM_NUM_i. A seguir são listadas as equações e as avaliações dos dados coletados no escalonador:

- Para análise de CPU, a avaliação se dá com base no cálculo da (i) capacidade de cada VM e do (ii) estado de carga das VMs de um dado nó. Para a primeira, a Equação (3.1) retorna a capacidade de CPU TC_{ij} de VM_{ij} em PM_i, sendo $i = 1, 2, ..., n$ o identificador da máquina física e $j = 1, 2, ..., n$, o índice da máquina virtual.

$$TC_{ij} = VCPU_{ij} \times \frac{W_{ij}}{W_{max}} \times \frac{Cap_{ij}}{Cap_{max}}. \tag{3.1}$$

  Nesta equação, $VCPU$ é uma CPU abstrata alocada no *Hypervisor*, sendo $VCPU_{ij}$ o respectivo $VCPU$ de VM_{ij}. O símbolo W_{ij} é o peso $VCPU$ de VM_{ij}. O *Hypervisor* de PM_i aloca intervalos de tempo para VM_{ij} de acordo com o peso W_{ij}. A razão $\frac{W_{ij}}{W_{max}}$, sendo W_{max} o peso máximo de $VCPU$, determina a proporção de tempo de CPU de PM_i alocada para VM_{ij}. O símbolo Cap_{ij} indica a parcela de tempo máxima de CPU de PM_i que VM_{ij} pode ocupar.

  Na Equação (3.2), é calculado o estado de carga dos nós, onde CPU_i indica a respectiva CPU de PM_i, sendo i o identificador da máquina física. O símbolo T_i (*threshold*) indica o estado do limite de sobrecarga e U_{ij} a utilização de CPU de VM_{ij}, onde j corresponde ao identificador da máquina virtual. Desse modo, se $Sat\_CPU_i > 0$, PM_i está em estado normal de uso de CPU; se VM_{ij} estiver sobrecarregada, mais recursos de CPU são alocados para o nó.

$$Sat\_CPU_i = CPU_i \times T_i - \sum_{j=0}^{VM\_NUM_i} TC_{ij} \times U_{ij}. \tag{3.2}$$

- Para o uso de memória, se aplica a Equação (3.3), com as mesmas ideias já utilizadas em (3.2), na qual se calcula Sat_Mem_i de PM_i, sendo i o índice da máquina física

correspondente. Nesta equação, Mem_i indica a memória total de PM_i. O símbolo T_i indica o estado de limite de sobrecarga. O termo $VMem_{ij}$, corresponde à memória de VM_{ij} utilizada e $VSwap_{ij}$ o total de memória de VM_{ij} em *swap*.

$$Sat\_Mem_i = Mem_i \times T_i - \sum_{j=0}^{VM\_NUM_i} VMem_{ij} \times VSwap_{ij}. \tag{3.3}$$

- Para o cálculo de rede, a Equação (3.4) retorna o estado de carga em Sat_Net_i para PM_i, onde, T_i tem o mesmo significado das equações anteriores, $NetBand_i$ indica a capacidade de rede de PM_i e $NetBand_{ij}$ a rede em uso de VM_{ij}.

$$Sat\_Net_i = NetBand_i \times T_i - \sum_{j=0}^{VM\_NUM_i} VNetBand_{ij}. \tag{3.4}$$

O próximo passo do algoritmo do escalonador é otimizar a distribuição dos recursos com base nas informações coletadas. No caso da realocação de CPU, mesmo que $Sat\_CPU_i > 0$ indique que PM_i está em estado normal de operação, algumas VMs podem estar operando com sobrecarga enquanto outras podem estar ociosas. Neste caso, uma parcela dos recursos pode ser liberada para as VMs que estão com alta demanda. Para VM_{ij}, Q_w é o número de tarefas em uma fila esperando para serem processadas. O limite inferior de carga é Q_l, e Q_h é o limite superior, com isso, se $Q_w > Q_h$ então a CPU está em estado de sobrecarga. U_{ij} representa a utilização de CPU para VM_{ij}, e T_{avg} é a média de uso de CPU de todas as VMs que estão alocadas em um nó. Ambos, Q_w e U_{ij}, são valores médios de vários *DISPATH_CYCLE*.

Similar ao método adotado para CPU, no caso da realocação de memória, a condição é $Th_{ij} < Th_l$ e $U_{ij} < T_{avg}$, e ainda $Th_{ij} > Th_h$ e $U_{ij} > T_{avg} \times Th_{ij} = Swap_{in} + Swap_{out}$. Os símbolos Th_l e Th_h são os limites de baixa e alta velocidade de *swap in* e *out*. A utilização de memória U_{ij} é calculada como $(Mem_{used} + Swap_{used})/Mem_{total}$.

A migração de recursos entre os nós utiliza o algoritmo *best-fit* para encontrar a *PM* mais adequada para a VM mais adequada, conforme a Equação (3.5). Nesta, CPU_{PM}, Mem_{PM} e NET_{PM} se referem aos recursos livres nas máquinas físicas, enquanto CPU_{VM}, Mem_{VM} e NET_{VM} se referem às necessidades das VMs.

$$Min\left\{\frac{CPU_{PM}}{CPU_{VM}} + \frac{Mem_{PM}}{Mem_{VM}} + \frac{Net_{PM}}{Net_{VM}}\right\} \tag{3.5}$$

para $CPU_{PM} > CPU_{VM}$, $Mem_{PM} > Mem_{VM}$ e $Net_{PM} > Net_{VM}$.

Avaliando os resultados do projeto, os autores destacam que o mesmo se mostra bastante eficiente para balanceamento local e global, contribuindo para um melhor uso de ES dos discos, economia de energia e também possibilidade de alocar tarefas críticas em máquinas específicas.

De acordo com o trabalho *A performance study on load balancing algorithms with task migration* (LU; LAU, 1994), se procura identificar situações onde a migração de tarefas pode prover um aumento extra no desempenho do sistema e assim apresentar um algoritmo de balanceamento de carga para este uso. Com base em informações atualizadas de carga, o sistema deve decidir qual o melhor local para uma tarefa ser executada. Neste artigo, o autor cita vários algoritmos (EAGER; LAZOWSKA; ZAHORJAN, 1986; NI; XU; GENDREAU, 1985; CHU; LAN, 1987; ZHOU, 1988) para essa abordagem e destaca a importância de três componentes, conforme também já citado na Seção 2.3 desta dissertação, que são (i) política de transferência, (ii) política de localização e (iii) política de informação. Com base nessas considerações, o autor analisa três diferentes algoritmos de balanceamento de carga dinâmicos com as mesmas políticas de informação e localização, diferindo quanto ao critério de seleção das tarefas na política de transferência. O autor conclui que os algoritmos que empregam tanto atribuição de tarefas como migração, apresentam desempenho significativamente melhor do que os que só permitem atribuição de tarefas.

Em outro artigo, Sharma (2012) apresenta um estudo da aplicação dos métodos de migração e duplicação de tarefas. A migração e a duplicação apresentam similaridades na medida em que tratam de migrar tarefas. No entanto, diferem na medida em que a primeira migra tarefas diferentes para um único processador e a segunda migra cópias duplicadas de uma tarefa em particular para vários processadores. Explorando essas características, foi apresentado um algoritmo que é uma mistura de Duplicação e Migração de Tarefas (SHARMA; NITIN, 2012). Tong (2010) avalia as vantagens e desvantagens dos algoritmos estáticos e dinâmicos e apresenta uma solução que combina as duas estratégias, baseado em uma taxa de carga residual ou RLR (do inglês, *Residual Load Rate*). A estratégia resultante desta combinação, considera todos os desempenhos de processamento e o estado das cargas nos servidores, procurando evitar os problemas associados ao método de balanceamento dinâmico, que continuamente analisa as informações de carga e gera grande tráfego de rede (TONG; ZHU, 2010). Na Figura 14 se apresenta o modelo do algoritmo de distribuição de carga com base no cálculo do RLR.

O método de cálculo para RLR é descrito como: supondo L_c como CPU residual, L_m como memória residual, t como peso da CPU, s como o peso da memória (os valores para t e s podem ser definidos de acordo com diferentes solicitações), o recurso residual L_i de cada nó é dado pela Equação (3.6):

$$L_i = txL_c + sxL_m, quandoi = 1, 2, ..., net + s = 1. \tag{3.6}$$

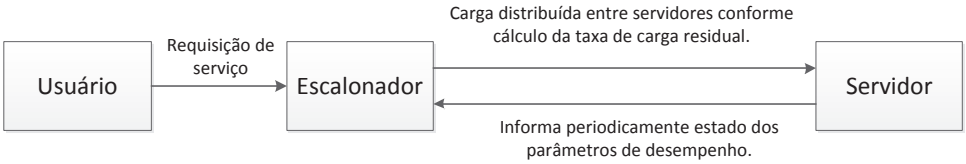
A soma dos recursos residuais L de todos os nós do sistema é dado por:

$$L = \sum_{i=1}^{n} L_i \tag{3.7}$$

Assim, RLR de um nó i é dado por:

$$R_i = \frac{L_i}{L} \tag{3.8}$$

Figura 14: Modelo do algoritmo de distribuição de carga com base no cálculo da taxa de carga residual.



Fonte: Tong (2010).

O algoritmo assume que o sistema possui n servidores, sendo o conjunto desses servidores dado por $S = \{S_i, i = 1, 2, ..., n\}$. A ideia central é que cada servidor do sistema reporte ao escalonador, num período determinado T (10 seg $\backsim$ 15 seg), os estados de seus parâmetros de desempenho. Dois parâmetros chave são utilizados para essa análise: utilização de CPU e memória. Quando o escalonador recebe esses parâmetros, calcula, para cada nó, a capacidade restante de carga $L_i(i = 1, 2, ..., n)$. Ao receber uma nova solicitação de usuário, esta demanda é encaminhada para o nó com maior taxa de carga residual. Um limite M pode ser atribuído para cada L_i, assim, quando $L_i \leq M$, o servidor estará com sobrecarga e novas solicitações não são encaminhadas para este nó. O autor atribui um único indicador M para os dois parâmetros chave ao mesmo tempo. Com isso, se apenas um exceder o limite, o servidor fica indisponível para novas solicitações.

Um dos problemas que pode ser apontado neste método é que, continuamente, os dados de todos os nós precisam ser coletados, gerando tráfego entre os servidores e aumento do custo de *overhead*. Para contornar esse problema, os servidores foram divididos em três grupos de acordo com a RLR das máquinas, sendo: (i) alto se $R_i \leq 25\%$, (ii) médio se $25\% \leq R_i \leq 75\%M$ e (iii) baixo se $R_i \geq 75\%$. Com essa divisão, logo que uma requisição chega no escalonador, ela é encaminhada para o grupo com menor carga, utilizando o algoritmo Round Robin. Com isso, o período de atualização dos parâmetros dos nós para o escalonador pode ser aumentado, reduzindo o tráfego. Caso alguma máquina do grupo apresente problemas antes da notificação, as perdas são minimizadas. Essa estratégia, que combina balanceamento estático e dinâmico, apresentou resultados satisfatórios nos testes, possibilitando intervalos maiores entre as leituras e tornando o sistema mais eficiente.

3.1.3 Plataforma transacional Base24-eps

Nesta subseção será apresentada a plataforma de transações BASE24-eps, da empresa ACI. Esta plataforma transacional oferece um sistema de captura e processamento de transações *online* que se destaca entre as maiores plataformas para meios de pagamento em uso na atualidade, adotado por HSBC Bank UK, MasterCard, ING Bank e State Bank of India entre outros. Desse modo, é pertinente analisar os aspectos relacionados à arquitetura adotada para a plataforma e como se dá o gerenciamento do fluxo de transações. As informações foram coletadas do *Red*

Book da IBM: *A Guide to the ACI Worldwide BASE24-eps on z/OS*, (KOOIJMANS, 2007).

BASE24-eps é uma plataforma de pagamento integrado, utilizada para adquirir, autenticar, rotear e autorizar transações financeiras através de múltiplos canais. A plataforma oferece uma gama completa de funcionalidades para suportar operações de pagamento, tanto para as transações tradicionais como para operações emergentes, tais como comércio móvel e *internet banking*. BASE24-eps é caracterizado pela confiabilidade, disponibilidade, escalabilidade e alto desempenho no processamento de transações. A arquitetura da plataforma permite extração de relatórios sem interrupção de processamento e também possibilita implementar novas regras de negócios através de arquivos de mudanças, sem requerer a reinicialização do sistema.

Na Figura 15, é apresentada a arquitetura lógica do BASE24-eps, separando a plataforma em quatro níveis. No nível mais ao alto da figura é onde se inicia e se concluí a transação, ou seja, os meios de captura (KOOIJMANS, 2007). Também nesta camada, está representado o autorizador externo da transação. A próxima camada é composta por um módulo de adequação dos protocolos de origem e um *middleware* responsável pela troca de mensagens com as camadas seguintes. Este *middleware* e as filas de mensagens serão analisados com maior aprofundamento a seguir. Após o tratamento dado na segunda camada, as transações são recebidas no *Integrated Server*. Este é o componente central do BASE24-eps, no qual estão embasados os serviços de alta disponibilidade, escalabilidade e balanceamento de carga (KOOIJMANS, 2007).

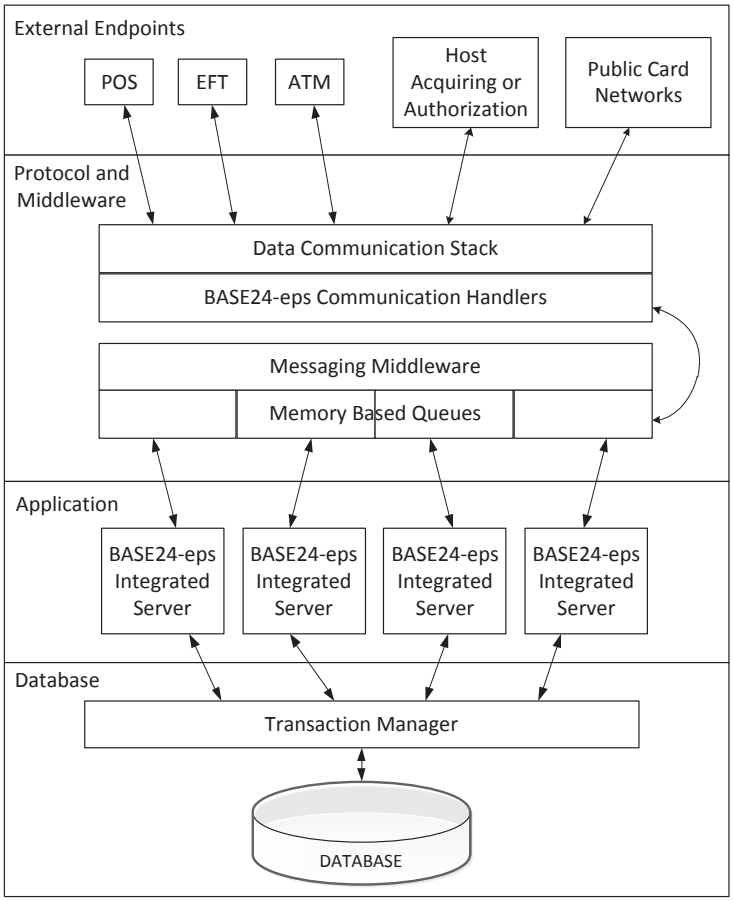
A camada de base de dados mantém informações para conciliação financeira das transações e para fornecer dados de relatórios gerenciais. Nesta camada também se encontra o gerenciador de transações (*Transaction Manager*), que garante a integridade transacional e capacidade de reversão de saldo. BASE24-eps depende deste gerenciador para assegurar a integridade da base de dados, a partir da qual se originam os arquivos e tabelas para fins de controle.

O *middleware* que gerencia a troca de mensagens entre entrada e saída de transações e o aplicativo de processamento, utiliza um modelo de troca de mensagens assíncronas. O modelo se baseia na utilização de filas de mensagens atuando como *buffers* entre os terminais e o aplicativo de processamento, conforme apresentado na Figura 16. Com a utilização de filas, o BASE24-eps consegue distribuir melhor os recursos pois, na medida em que os processadores ficam livres, retiram mensagens da fila para processar. Desse modo, não ocorre concentração de transações em um único processador, resultando em um melhor uso dos recursos como um todo.

Caso algum dos autorizadores externos não responda as transações, conforme destacado em linha tracejada na Figura 16, ou a rede de comunicação apresente lentidão, o protocolo padrão da indústria, ISO 8583, oferece mecanismos para evitar inconsistências de saldo. As informações necessárias para executar esses procedimentos transacionais são obtidos da base de dados do sistema.

Os módulos *Integrated Server*, consomem as mensagens de transação da uma fila de entrada comum, disponibilizada pelo *Messaging Middleware*, conforme as instâncias são executadas.

Figura 15: Arquitetura lógica do BASE24-eps.

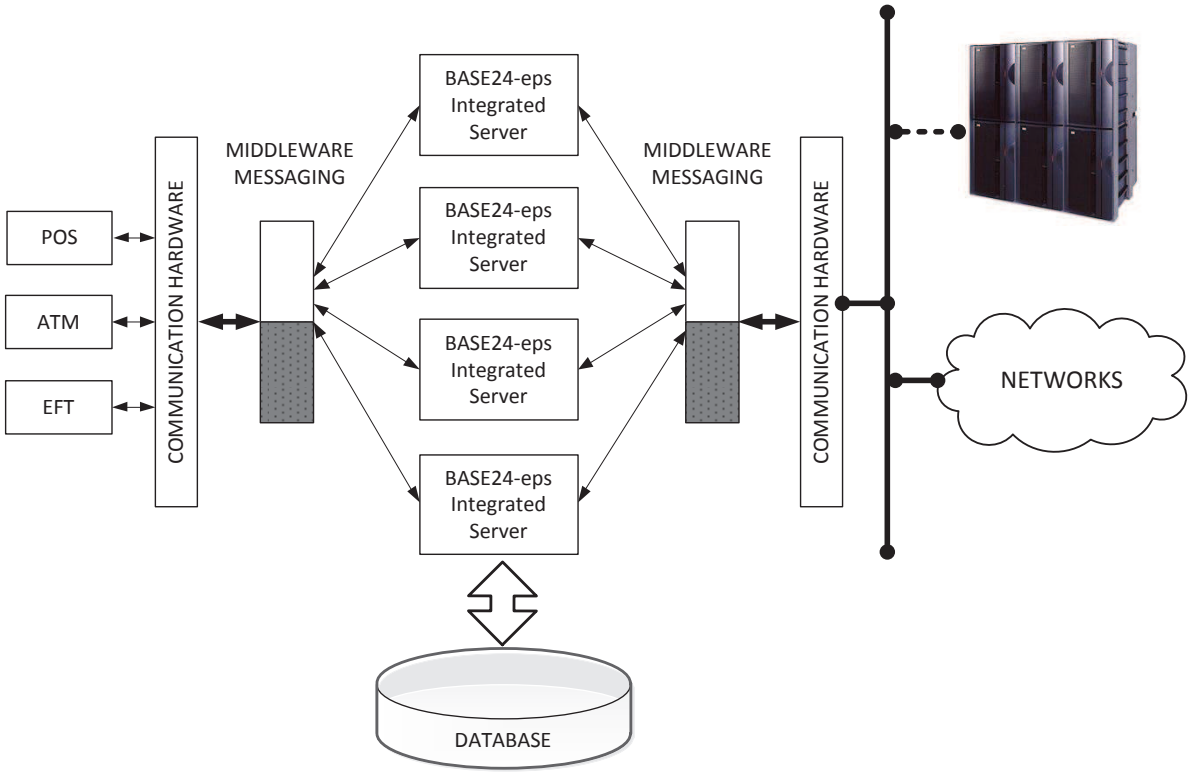


Fonte: Kooijmans (2007).

Cada módulo é capaz de processar qualquer tipo de mensagem de terminal remoto. Por esta razão e, considerando que o *Integrated Server* é *context-free* (contexto da transação é salvo no sistema e não em endereço de memória), não é necessário encaminhar nenhuma mensagem financeira de entrada para qualquer instância específica do *Integrated Server*. Este módulo é implementado como uma classe de serviço, na qual qualquer membro da classe pode lidar com qualquer unidade de trabalho.

A plataforma BASE24-eps normalmente é executada em servidores NonStop da empresa HP. O NonStop é um conjunto massivamente paralelo de processadores independentes, cada um executando sua própria cópia do sistema operacional (BERNICK et al., 2005). Os processadores se comunicam um com o outro e com os adaptadores partilhados de ES, através da ServerNet (COMPAQ, 1998). Uma falha de hardware pode prejudicar no máximo um processador ou uma placa de ES. Mesmo com a falha, a redundância no sistema permite a aplicação do usuário para continuar ininterruptamente.

Figura 16: Modelo de mensagens assíncronas.



Fonte: Kooijmans (2007).

3.2 Escalonamento em Computação em Nuvem

Apesar do alvo de estudo desta dissertação não ser computação em nuvem, é pertinente abordar os trabalhos relacionados a essa área da computação devido aos desafios e necessidades comuns ao que está sendo pesquisado, aliado ao fato de que comumente se encontra escalonamento Round Robin neste tipo de sistema (PARAISO; MERLE; SEINTURIER, 2013). Ao longo desta seção, será analisado o fato da computação em nuvem se tratar de um sistema distribuído que utiliza técnicas de escalonamento e balanceamento de carga.

Por definição, a computação em nuvem pode ser vista como um conjunto de recursos de computadores virtualizados (ENDO et al., 2010). Com base nesta virtualização, o paradigma de computação em nuvem consiste em possibilitar que as cargas de trabalho sejam dimensionadas e distribuídas rapidamente pelas máquinas virtuais ou máquinas físicas. Qualquer solicitação de recursos será entregue pela nuvem em termos de máquina virtual. Assim, a alocação de máquinas virtuais é a parte mais importante na computação em nuvem. Gestão de recursos é uma necessidade indispensável em computação em nuvem, através da qual se pode assegurar o uso eficaz de recursos, proporcionando escalabilidade e elasticidade.

No artigo *The Future of Workload Management* (HARRINGTON, 2012), o autor afirma que ao olharmos para o futuro do gerenciamento de carga de trabalho, se destacam três grandes tendências: (i) conhecimento do *Big Data* com que se está operando; (ii) uso de nuvens em

Tabela 5: Políticas de balanceamento para diferentes plataformas de computação em nuvem.

Plataforma	Política de balanceamento
Amazon EC2	Proprietária
Nimbus	Round Robin e Algoritmo Guloso
Eucalyptus	Round Robin e Algoritmo Guloso
OpenNebula	*Match Making*: alocação inicial com base em política de classificação
OpenNebula e Haizea	Alocação dinâmica para prover reserva antecipada de recursos

Fonte: Upadhyay (2012).

computação de alto desempenho, ou HPC; e destaca (iii) o conhecimento da aplicação, para o qual afirma que cada vez mais os gerenciadores precisam conhecer a aplicação que estão executando, de modo que quanto mais o gerenciador compreende a carga de trabalho, mais eficiente ele pode escalonar, gerenciar e adaptar o ambiente de computação. Métricas como largura de banda, alocação de memória, espaço de armazenamento, CPU, etc. auxiliam o gerenciador a conhecer uma aplicação, a fim de otimizar a gestão.

As amplas possibilidades de escalabilidade oferecidas pelas plataformas de nuvem podem ser aproveitadas não apenas para serviços de hospedagem de aplicações, mas também como um recurso sob demanda de computação pura (SOTOMAYOR et al., 2009). Por fim, provedores de serviço são constantemente pressionados a propor sua infra-estrutura de modo a permitir, em tempo real, visibilidade de ponta-a-ponta e gestão dinâmica de recursos e com controle de baixa granularidade, para reduzir o custo total de utilização e ao mesmo tempo melhorar a agilidade. Mais uma vez, a computação em nuvem se define como um conjunto de recursos computacionais virtualizados.

Para atender tais necessidades, entre os desafios que se apresentam, se destaca o escalonamento de máquinas virtuais em máquinas físicas (WANG; LE; ZHANG, 2011; ZHOU et al., 2010; HU et al., 2010). Esta demanda passa por uma eficiente monitoração de máquinas e definição de uma eficiente política de alocação de VMs que seja a mais adequada para a arquitetura escolhida (SOTOMAYOR et al., 2009). A gestão dinâmica de recursos representa um desafio complexo para algoritmos de escalonamento do tipo Round-Robin, contudo, como será visto a seguir, é comumente utilizada na computação em nuvem. A Tabela 5 relaciona algumas das principais plataformas de *software* da computação em nuvem, com suas respectivas políticas de balanceamento.

No trabalho de Upadhyay et. al. (2012), os autores fazem uma análise do escalonamento de tarefas e recursos na plataforma OpenNebula através de uma série de testes. Em um primeiro teste foram criadas simultaneamente três VMs e se verificou o comportamento do escalonador durante os diferentes estágios até a execução (*activate, prolog, boot* e *running*). Com este experimento foi verificado que, por padrão, o escalonador do OpenNebula opera no regime de *First Come First Served* (UPADHYAY; PATEL, 2012), como se pode observar na Tabela 6.

Tabela 6: Tempos das VMs em diferentes estágios.

Estágio	VM 1	VM 2	VM 3
ACTIVE	0.00	0.30	1.00
PROLOG	0.00	0.30	1.00
BOOT	2.17	2.18	3.31
RUNNING	3.30	3.33	3.37

Fonte: Upadhyay (2012).

Também foi analisado como o escalonador se comporta quando se excede a capacidade de VMs alocadas no *Host*. Criando sucessivamente 2, 4 e 11 VMs, foi observado que o escalonador passa a filtrar a alocação das mesmas no *Host* na medida em que os limites de CPU e memória deste chegam ao limite de 100% de utilização. Ou seja, o escalonador do OpenNebula mantém as VMs excedentes em fila, interrompendo o envio para o *Host* enquanto este não tem capacidade suficiente para executar mais VMs.

Em outro experimento se analisou como o escalonador distribui as VMs quando houver mais de um *Host* disponível. Para esse teste foram criados três *Hosts* para os quais se demandaram três VMs. Os resultados mostraram que o escalonador atribui uma máquina virtual para cada *Host*. Assim, se pode afirmar que o escalonador distribui igualmente VM entre os *Hosts*, enquanto houver vários *Hosts* disponíveis.

No último experimento se implementou uma política de classificação e se identificou como o escalonador aloca *Host* para as VMs. Neste teste se configurou uma estrutura com três *Hosts* e a seguir se criou seis VMs, sucessivamente, mas não simultaneamente. O critério definido para o escalonador foi de memória livre, ou seja, uma VM é alocada nos *Hosts* de acordo com a menor memória livre. A partir desta experiência, foi verificado que o escalonador trabalha com sucesso com a política de classificação, distribuindo as VMs entre os *Hosts* conforme a priorização estabelecida.

No trabalho de Cordeiro et. al. (2010), os autores destacam a importância dos algoritmos de alocação nas plataformas de computação em nuvem para garantir uma utilização inteligente e até ótima dos recursos. Recursos mal utilizados sempre resultam em ineficiência (CORDEIRO et al., 2010). Nas análises de algumas plataformas, os autores verificaram que XCP não oferece uma opção incorporada para os algoritmos de alocação automática. Desse modo, se um provedor desejar disponibilizar essa opção, precisa implementar uma plataforma utilizando a API do XCP ou usar a solução de um *software* de balanceamento de carga. A versão disponível da plataforma Eucalyptus tem um algoritmo simples que parece dedicar pouca atenção para alocação das VMs nas nuvens. Em contraste, OpenNebula traz uma abordagem interessante para alocação das VMs com base no seu valor de classificação definido, que pode ser usado para implementar políticas úteis. Como tal, pode ser mais interessante para usuários que veem o uso de recursos e otimização como um requisito primordial.

No artigo *Sandpiper: Black-box and gray-box resource management for virtual machines* (WOOD et al., 2009), são explorados os benefícios da virtualização de *data centers*, possibilitando migrar e redimensionar servidores a fim de eliminar *hotspots*. Nesse contexto, é apresentado *Sandpiper*, um sistema que monitora e detecta *hotspots*, definindo um novo mapa físico para os recursos virtuais, redimensionando máquinas virtuais e promovendo as migrações necessárias. As técnicas foram implementadas em Xen e conduzem uma avaliação detalhada que combina análise de CPU, rede e uso de memória.

Considerando a popularização de *data centers*, que vem ocorrendo através de uso para hospedagem de *sites*, sistemas empresariais e *sites* de *e-commerce*, nestes ambientes uma grande quantidade de servidores executam uma ou mais aplicações, com componentes que podem estar distribuídos entre os recursos. Além disso, as aplicações podem estar submetidas a flutuações dinâmicas de carga, causadas por incrementos de carga, eventos sazonais e situações de *flash crowds* (APPLEBY et al., 2001). Neste cenário, obter desempenho e atender aos SLAs (do inglês, *Service Level Agreement*) estabelecidos é uma tarefa complexa.

A utilização da virtualização é uma abordagem que, entre outros benefícios, visa reduzir a complexidade do gerenciamento de recursos. Nesta abordagem, as aplicações são executadas em máquinas virtuais, alocadas em servidores virtuais e mapeados em servidores físicos. Dentre os benefícios desta abordagem, se pode citar a possibilidade de isolar as aplicações de outras aplicações maliciosas ou que consumam recursos exageradamente, estando alocadas no mesmo servidor físico. Também se obtém uma melhor distribuição de recursos entre as aplicações e, principalmente, proporciona a habilidade de flexibilizar o mapa de recursos físicos para servidores virtuais, a fim de lidar melhor com a dinamicidade das cargas de trabalho, seja aumentando a quantidade de recursos para atender uma determinada tarefa ou migrando servidores virtuais para máquinas físicas com menos carga de trabalho. A migração de servidores é transparente para a aplicação e modernas plataformas de virtualização suportam essa capacidade (CLARK et al., 2005; NELSON; LIM; HUTCHINS, 2005).

Normalmente, detectar *hotspots* e iniciar uma migração de carga, são tarefas executadas manualmente, o que carece de agilidade para lidar com mudanças súbitas de carga. Além disso, é suscetível a erros, pois cada remodelação pode exigir migrações ou trocas de vários servidores virtuais para reequilibrar a carga do sistema. A migração é ainda mais complicada devido à necessidade de considerar vários recursos, como CPU, rede e memória para cada aplicação e servidor físico.

Para lidar com estes desafios, Wood et. al. (2009) estudam estratégias de caixa-preta e caixa-cinza para provisionamento de máquinas virtuais em grandes *data centers*. As técnicas consistem em monitorar o uso de recursos, detectar pontos de acesso, alocação de recursos e iniciar as migrações necessárias. Nas técnicas de caixa-preta, podem ser tomadas decisões simplesmente observando cada máquina virtual a partir do exterior, sem qualquer conhecimento das aplicações em execução nas VMs. Na abordagem de caixa cinza, se tem acesso a uma pequena quantidade de estatísticas do SO, combinado com observações externas para melhor

informar o algoritmo de provisionamento.

O sistema Sandpiper implementa um algoritmo que detecta *hotspots* e determina se as VMs devem ser redimensionadas ou migradas. Também implementa um algoritmo de migração de *hotspots* que determina o *que* e para *onde* migrar, bem como *quantos* recursos alocar.

A monitoração das máquinas, quando feita através de técnica de caixa-preta, pode ser considerada não intrusiva. Desse modo, a CPU é monitorada através do XenMon (GUPTA; GARDNER; CHERKASOVA, Technical Report, 2005), *hypervisor* do Xen, verificando os eventos que estão sendo alocados para processamento. Neste mesmo âmbito, o tráfego de dados é monitorado através de um *driver* de interface de rede e a memória através de atividades de *swap*, uma vez que através de uma caixa-preta não é possível inferir na quantidade de memória em cada VM. De outro modo, na técnica de caixa-cinza, a monitoração da CPU, rede e memória é feita através de *Daemon* leves instalados em cada VM.

O método que foi adotado para detecção de *hotspots*, consiste em sinalizar se os usos de CPU, rede ou atividades de *swap* excederem limites pré-definidos, conforme os SLAs estabelecidos. Para evitar que pequenos picos transitórios acionem a sinalização desnecessariamente, apenas se consideram as ocorrências em que os limites de tolerância são excedidos e sustentados por um certo período de tempo. Quanto maior esse período, menos agressiva é a detecção e, quanto menor esse período, mais agressiva a detecção.

Dado que detecção de *hotspot* indica deficiência de recursos nas máquinas para atender a demanda (MCINTOSH et al., 2011), o sistema Sandpiper provisiona recursos após estimar a quantidade que se faz necessária, verificando se pode atender com recursos locais ou necessita efetuar migrações. Nos casos em que precisa redimensionar VMs, o algoritmo adiciona recursos de CPU, memória e rede, dependendo de qual destes acionou a sinalização de que os limites foram excedidos.

Se houver necessidade de migração de VM, assim que é conhecida a quantidade necessária de recursos, o gerenciador de migração emprega um algoritmo guloso para identificar qual VM precisa ser migrada. A ideia básica consiste em migrar a carga do servidor mais sobrecarregado para o menos sobrecarregado. A métrica utilizada para combinar a carga de um servidor ou máquina virtual é dada conforme a Equação (3.9). De acordo com essa equação, quanto maior a utilização de um recurso, maior será seu vol.

$$Vol = \frac{1}{1 - cpu} \times \frac{1}{1 - net} \times \frac{1}{1 - mem} \tag{3.9}$$

onde cpu, net e mem correspondem as suas respectivas cargas percentuais.

Para determinar qual máquina deve ser migrada, o algoritmo ordena as máquinas físicas em ordem decrescente, conforme os respectivos volumes (vol). Dentro de cada servidor, as VMs são consideradas em ordem decrescente de acordo com uma taxa de VSR (*volume-to-size ratio*); onde $VSR = vol/size$, sendo $size$ o tamanho da memória alocada para a VM. O algoritmo prossegue considerando o maior VSR (VM) do maior vol (máquina física) e procura

hospedar na máquina com menor vol. O movimento é realizado apenas se o servidor de destino tem suficiente capacidade de CPU, rede e memória para atender as demandas necessárias de carga.

Os testes realizados pelos autores, comprovaram a eficácia de Sandpiper no balanceamento de carga entre os recursos de um *data center*, destacando a capacidade de automatizar as atividades de monitorar e detectar *hotspots*, para então determinar um novo mapa para alocação VMs nas máquinas físicas, além de redimensionar e migrar as VMs. Além disso, foi verificado que através da técnica *black-box*, Sandpiper é capaz de eliminar simultaneamente *hotspots* em múltiplos recursos e que, através da técnica de caixa-cinza, as informações podem ser utilizadas para melhorar a capacidade de resposta do sistema, particularmente obtido através de uma alocação pró-ativa de memória e uma melhor inferência sobre os requerimentos dos recursos.

3.3 Análise de Desempenho de Sistema de Transações Financeiras

Avaliar o desempenho de um sistema de transações através de um modelo computacional representa um grande ganho de análise para diferentes situações, possibilitando evoluir o sistema para operar de forma eficiente, a fim de atingir os requisitos estabelecidos nos SLAs. Dentre tais requisitos, se pode destacar os acordos sobre a disponibilidade, escalabilidade, confiabilidade e segurança. Além disso, evita falhas que podem causar grave degradação no desempenho do sistema.

No trabalho de Sousa et. al. (2009), os autores apresentam um modelo estocástico polinomial para avaliar o desempenho do sistema de processamento de TEF e infraestrutura de armazenamento, considerando uma faixa de variação de carga. A estratégia de avaliação de desempenho utilizada pelos autores é baseada em uma abordagem de modelagem hierárquica, que combina resultados de modelos de confiabilidade e desempenho, para apoiar o planejamento de capacidade e garantir o SLA. Os modelos propostos focam em armazenamento e processamento de recursos que são representados por Redes de Petri Estocásticas Generalizadas (RPEG) (BALBO, 2007).

RPEG possuem alto nível de formalismo e são amplamente aplicadas na avaliação de desempenho e confiabilidade (MURATA, 1989). Esse formalismo admite duas classes diferentes de transições no modelo: transições imediatas e transições temporizadas. A transição imediata dispara em tempo zero assim que se encontra habilitada. A transição temporizada dispara após um tempo aleatório, distribuído exponencialmente. Balbo (2007) e Marsan et. al. (1998) demonstram que RPEGs são equivalentes aos processos estocásticos de tempo contínuo e também apresentam solução para derivar o estado estacionário de distribuições de probabilidade (MARSAN et al., 1998).

Aplicando a técnica de aproximação de fase para modelagem de atividades não-exponenciais, se pode obter uma variedade de atividades de desempenho e dependabilidade em modelos RPEG usando subredes de transferência (TOUMODGE, 1995). Estas sub-redes representam

funções polinomiais, tais como as distribuições Erlang e Hypoexponential (TOUMODGE, 1995).

Números medidos (distribuição empírica) de um sistema com média μ_D e desvio padrão σ_D devem ajustar seu comportamento estocástico através da técnica de aproximação de fase. O inverso do coeficiente de variação (CV) dos números medidos permite a seleção de qual distribuição corresponde melhor, conforme Equação (3.10). Os autores adotaram as distribuições Erlang e Hypoexponential, por se ajustarem melhor.

$$\frac{1}{CV} = \frac{\mu_D}{\sigma_D} \tag{3.10}$$

Quando o inverso de CV é um número inteiro e diferente de um, o valor empírico caracteriza uma distribuição de Erlang, representada em RPEG por uma sequência de transições exponenciais. Se o inverso do CV é um número não inteiro maior que um, o valor empírico é representado por uma distribuição hypoexponential representado por uma RPEG composta por uma sequência.

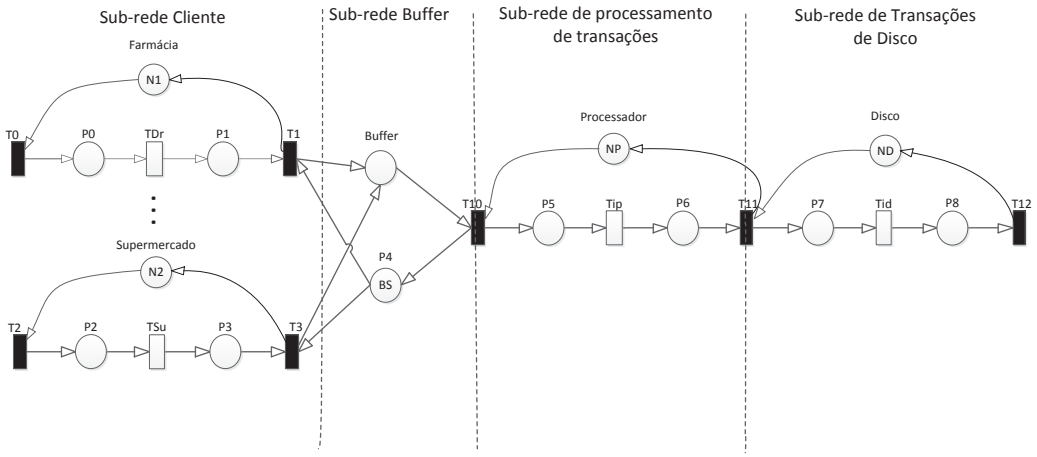
Conforme visto na Seção 2.1, um sistema de transações eletrônica é composto de aplicativos de cliente e gerenciamento, sendo os clientes os pontos de venda e os de gestão aqueles que controlam o processamento da transação. Desse modo, as fases do processo do cliente são: (i) mensagens de exibição, (ii) a leitura dos dados do cartão, (iii) captura de senha e (iv) impressão do cupom. A aplicação de gerenciamento recebe todos os dados gerados pelos pontos de venda, incluindo parâmetros necessários para a construção de mensagens e, em seguida, encaminha a transação.

A partir destas definições do processo, foi feito o modelo RPEG para o sistema transacional, conforme Figura 17 (SOUSA et al., 2009). A proposta deste modelo se dá através das sub-redes que descrevem os componentes do sistema. Sub-redes dos clientes representam aplicações de ponto de venda em diferentes áreas do comércio, tais como farmácias e supermercados, por exemplo. As cargas de trabalho originadas destes clientes podem ser alteradas para se obter uma grande variedade de tráfego. Neste modelo N_i representa locais de comércio, onde se originam as transações, são as sub-redes nas quais se especificam as quantidades de terminais e as transições estocásticas genéricas que representam o atraso de distribuição entre as transações. As transições temporizadas são representadas graficamente no modelo por barras espessas, e as transições imediatas por barras vazias. A sub-rede *Buffer* representa as transações aguardando para serem atendidas.

O módulo de gerenciamento é composto pelas sub-redes de processamento de transações e armazenamento em disco, sendo a primeira responsável pelo tratamento das transações e a segunda por ações de escrita e leitura em disco. N_p indica a capacidade de processamento, dado pelo número de transações concorrentes suportadas e N_d o número de transações concorrentes de disco suportadas. As transições estocásticas de demanda de rede T_{ip} e T_{id}, representam o tempo de processamento e escrita e leitura em disco.

Os resultados obtidos através do modelo RPEG para um sistema transacional demonstraram

Figura 17: Modelo de sistema de transações.



Fonte: Sousa et. al. (2009).

que é possível analisar o desempenho do sistema e simular diferentes situações com aumento de carga e flutuações de demanda, verificando se o SLA está sendo atendido. Seguindo com a análise de sistemas transacionais, no próximo capítulo será feita uma análise aprofundada de um sistema transacional com suas particularidades de carga, subsistemas e conectividade.

3.4 Análise do Capítulo

Ao concluir o capítulo dos trabalhos relacionados, se identificam as necessidades e dificuldades comuns aos sistemas distribuídos para efetuar um eficiente balanceamento de carga. Essas questões impõem decisões que precisam ser tomadas em âmbito de projeto e, para tanto, é preciso conhecer profundamente o sistema abordado a fim de obter o escalonador apropriado. É importante conhecer suas características quanto ao tipo de carga, fluxo das operações, tipo de rede e latências, bem como os aspectos de monitoração e atuação sobre seus recursos.

Um sistema distribuído de transações financeiras possui muitas particularidades que são encontradas isoladamente em alguns dos trabalhos analisados neste capítulo, das quais se pode destacar a alta disponibilidade e o alto desempenho com heterogeneidade de carga, sistemas e rede. Contudo, são verificadas lacunas ao se pesquisar sobre métricas e técnicas de balanceamento de carga, as quais impedem que os trabalhos relacionados sejam plenamente aplicáveis em sistemas com as particularidades evidenciadas para o sistema de transações financeiras.

Nos trabalhos analisados, se destacaram as monitorações dos recursos, verificando continuamente consumo de CPU, memória, latência de rede e tempos de E/S, a fim de manter o escalonador sempre atualizado e capacitado para efetuar um bom balanceamento. No entanto, para contornar o *overhead* decorrente dessas atualizações, Sharma (2012) atuou sobre a periodicidade das consultas, porém, em um sistema transacional, isso pode resultar em perda de transações se algum recurso ficar indisponível durante o período de atualização. Mcheick et. al.

(2011) apresentam alternativas de algoritmos que não operam bem com questões de heterogeneidade de recursos e de carga.

Conforme as considerações acima, os métodos adotados nos trabalhos relacionados para minimizar o *overhead* das comunicações não atendem às exigências de um sistema de transações, sendo necessário que outras técnicas sejam elaboradas. Nesse trabalho, o método proposto utiliza uma técnica de notificações assíncronas entre os recursos e o escalonador, mantendo este periodicamente atualizado e, em situações que podem resultar em perda de processamento de tarefas, o recurso notifica o escalonador antes do período programado. Essa técnica evita o despacho de transações para serem processadas em máquinas problemáticas.

As métricas adotadas para escalonamento nos trabalhos relacionados combinaram alguns parâmetros para tomada de decisões, associando geralmente medidas de consumo dos recursos (ZHOU et al., 2010). No entanto, essas medidas são ineficazes se o sistema opera com carga heterogênea, e tem como objetivo distribuir tarefas entre recursos especializados. Também conforme Sharma (2012), o método RLR não contempla tais necessidades.

A solução adotada pela plataforma BASE24-eps, que utiliza uma fila de transações com múltiplos recursos consumindo as tarefas de modo assíncrono, constitui uma maneira eficaz de oferecer alta disponibilidade e desempenho com fácil escalabilidade. Contudo, é uma plataforma de alto custo e precisa ser instalada em máquinas específicas tais como *mainframes* IBM e HP NonStop, também com custo elevado.

Na seção dedicada à computação em nuvem, destacou-se a relevância do gerenciamento de recursos para obter escalabilidade e elasticidade. Um alto grau de gerenciamento pode ser alcançado através de políticas eficientes de balanceamento e monitoração de carga e recursos. Tais demandas sistêmicas, além da virtualização das máquinas, também são encontradas em um sistema transacional, demonstrando a conexão entre ambos. Esta relação se estende também aos desafios que se apresentam ao equacionar eficazmente os dados coletados de carga e recursos e em minimizar o *overhead* das comunicações. Desse modo, os ganhos obtidos com a infraestrutura apresentada nesta dissertação também podem ser aplicados à computação em nuvem.

Na Tabela 7 estão relacionados os sistemas que foram analisados ao longo deste capítulo. Nesta tabela, através das colunas é possível comparar os principais atributos de cada sistema, com base nos parâmetros do modelo apresentado nesta pesquisa. Neste contexto, além do tipo de escalonamento, está sendo relacionado o parâmetro "Atualizações", que se refere ao modo como as informações são atualizadas no escalonador, podendo ser periódico ou assíncrono. Também se verifica, para cada sistema, como é o suporte para "Notificações", "Rede heterogênea"e "Recursos heterogêneos". As notificações correspondem aos métodos para sinalizar ao escalonador sobre um evento assíncrono que precisa ser atendido com alta prioridade. A capacidade de lidar com heterogeneidade de rede e de recursos, representa uma característica de grande aplicabilidade em sistemas transacionais, conforme analisado no referencial teórico do Capítulo 2.

Nesse contexto, é preciso elaborar um método que habilite o escalonador para atender às demandas de heterogeneidade de carga com recursos especializados. Através de uma técnica heurística, que combina tanto a avaliação da capacidade dos recursos quanto do tipo de tarefa a ser processada, este trabalho preenche uma lacuna que possibilita ganhos significativos em utilização de recursos, na medida em que máquinas de alto processamento podem ser alocadas apenas para demandas que necessitam dessa especialização, enquanto se alocam recursos menos capacitados para outras atividades menos críticas.

Para condução da dissertação, as lacunas que se apresentaram nos trabalhos relacionados serão solucionadas através de métodos concebidos com foco nas perspectivas próprias de um sistema transacional. Desse modo, as técnicas elaboradas para o balanceamento de carga de um sistema de transações eletrônicas serão apresentadas no capítulo seguinte deste trabalho.

Tabela 7: Sistemas avaliados e respectivos métodos de balanceamento.

Sistema	Escalonamento	Atualizações	Suporte		
			Notificações	Rede heterogênea	Recursos heterogêneos
VMCTune	Monitoração e migração de recursos entre máquinas físicas.	Periódicas	Não	Não	Sim
RLR	Monitoração de recursos e migração de tarefas.	Periódicas	Não	Não	Sim
Base24-eps	Tarefas são alocadas em fila comum e, de modo assíncrono, retiradas para serem consumidas pelos recursos.	N/A	Para manutenção, via arquivo.	Não	Não
Amazon EC2	Proprietário	*Cloud Watch*	Sim	Sim	Sim
Nimbus	Round Robin e Algoritmo Guloso	*Nimbus Phantom*	Sim	Sim	Sim
Eucalyptus	Round Robin e Algoritmo Guloso	*Cloud Controller Node*	Sim	Sim	Sim
OpenNebula	*Match Making*	*Monitoring probes*	Sim	Sim	Sim
Sandpiper	Monitoração e migração de máquinas virtuais entre máquinas físicas.	Periódicas	Não	Não	Sim

Fonte: Desenvolvido pelo próprio autor.

4 MODELO GETLB

Neste capítulo é apresentada uma **arquitetura capaz de oferecer uma solução de balanceamento de carga** que atenda às necessidades do sistema atual **sistema transacional** da empresa GetNet, frente aos desafios que se apresentam para os próximos anos. Esta arquitetura é chamada GetLB e proporciona balanceamento de carga a partir do chaveador, suportado pela monitoração das máquinas de processamento. Para oferecer um balanceamento eficiente e com alta disponibilidade, GetLB é sustentado por três elementos: (i) arquitetura de comunicação, (ii) heurística para cálculo do balanceamento de carga e (iii) notificações originadas a partir das MPs ao chaveador.

Com GetLB, dada uma infraestrutura de m máquinas processadoras e um centralizador que recebe as transações, através do algoritmo heurístico, ele escolhe em tempo de execução qual máquina é a mais adequada para processar o tipo específico de transação que foi recebida. Além do processamento das transações, cada máquina processadora atualizará periodicamente o nó centralizador sobre sua capacidade, provendo informação para a tomada de decisão do algoritmo de balanceamento. Nesse âmbito, situações de exceção nas máquinas processadoras, que poderiam levar a uma perda transacional, são encaminhadas ao nó centralizador através de notificações aperiódicas, alertando ao algoritmo sobre problemas de desempenho ou necessidade de manutenção.

Nas próximas seções será descrito o modelo GetLB, detalhando os três elementos de sustentação da arquitetura. Na Seção 4.1 estão descritas e justificadas as decisões do projeto. A seguir, na Seção 4.2, é feita uma descrição da arquitetura de comunicação de GetLB, abrangendo os componentes do sistema e a interação entre os mesmos. Na Seção 4.3, será apresentada a técnica heurística adotada para o balanceamento através do cálculo do nível de carga (ou LL, *Load Level*) das máquinas. Na Seção 4.4 são abordados os métodos de notificação originadas das MP ao chaveador, garantindo disponibilidade e elasticidade ao sistema. Na Seção 4.5, é feita uma análise científica do balanceamento de carga oferecido pelo modelo. O capítulo se encerra com a Seção 4.6, onde se realiza uma análise dos temas que foram abordados.

4.1 Decisões de Projeto

Com base nas motivações que foram expostas na Seção 1.1, as decisões do projeto foram orientadas para que a nova arquitetura seja capaz de lidar com heterogeneidade de máquinas, de carga e também de rede.

Para que GetLB atinja os objetivos esperados, oferecendo uma arquitetura de alta disponibilidade com alto desempenho e otimização de uso de seus recursos, o sistema precisa ser construído de modo que contemple as necessidades de uma comunicação eficiente entre as máquinas, oferecer condições para que as notificações sejam disparadas das MP e priorizadas no chaveador e, por fim, aplicar de modo efetivo a heurística de nível carga. Com base nestes

apontamentos, a infraestrutura foi desenvolvida com as seguintes decisões de projeto:

i. O módulo chaveador será o escalonador, dimensionado adequadamente para evitar sobrecarga;

ii. O escalonador deve trabalhar com informações atualizadas sobre as MP para cálculo de escalonamento, utilizando método de atualização assíncrono para evitar *overhead*;

iii. O agendamento de transações deve combinar dados relevantes, a fim de compor uma heurística eficiente para a estimativa de carga;

iv. As máquinas processadoras devem ser capazes de notificar o chaveador quando ocorrem eventos assíncronos que podem representar impacto sobre as decisões de escalonamento.

Com base nessas decisões, o projeto deve se utilizar das técnicas e políticas de escalonamento e balanceamento de carga para compor a estratégia adequada. A seguir, as decisões são apresentadas com embasamento científico e nas referências descritas nos Capítulos 2 e 3 desta dissertação.

Para embasamento científico das decisões do projeto, esta subseção recorre às referências apresentadas nos capítulos anteriores desta dissertação. Nessa diretriz, a estratégia de balanceamento de carga, conforme visto na Seção 2.3.1, pode ser definida pelos critérios de definição dos parâmetros "quem", "qual" e "onde" para o sistema. Nesse âmbito, as decisões do projeto foram por uma dinâmica de despacho das transações a partir do escalonador para as máquinas processadoras, e, no sentindo inverso, atualizações sobre capacidade de processamento, sendo transmitidas das máquinas processadoras para o escalonador, caracterizando o parâmetro "quem" como do tipo *symmetrically-initiated*. Essa escolha possibilita que o sistema opere com a agilidade necessária para evitar despacho de transações para alguma máquina sem as plenas condições de processamento, prevenindo a perda de transações e tornando assim o escalonador pró-ativo. Isso caracteriza a política de localização da máquina receptora da transação, conforme Seção 2.3.2, como seleção dinâmica.

Ainda de acordo com as políticas descritas na Seção 2.3.2, as atualizações originadas das máquinas processadoras tem por objetivo manter o escalonador sempre com as informações mais recentes, descartando as mais antigas. A estratégia adotada para a periodicidade de envio das notificações visa encontrar o ponto de equilíbrio, no qual as informações estejam constantemente atualizadas, contudo, não com uma frequência alta o suficiente para gerar *overhead* na rede. Com isso, foi adotada uma política em que as atualizações serão periódicas e também aperiódicas, ou seja, em períodos regulares e não curtos serão transmitidas para o escalonador. Contudo se ocorrer alguma situação que pode levar à perda de transação na máquina processadora, a notificação é transmitida ao escalonador antes do período pré-definido. Essa técnica de notificações caracteriza uma comunicação assíncrona entre escalonador e recursos.

Quanto às informações utilizadas para o balanceamento, serão utilizadas tanto aquelas das transações recebidas no escalonador como de todo o conjunto das máquinas processadoras,

oriundas das notificações. Essa escolha possibilita uma convergência eficiente para o balanceamento. Desta maneira, o parâmetro "qual" é do tipo informações globais.

Para o local "onde" serão tomadas as decisões do balanceamento, apesar de haver um processo mestre, que é executado no chaveador Cisco que distribui a carga, também se está adotando uma arquitetura em que as máquinas processadoras são capazes de monitorar sua capacidade de acionar o escalonador, descentralizando as decisões. Isso caracteriza um sistema de decisão híbrido. A escolha de um nó central para executar o processo mestre se deu pelo fato de que atualmente o sistema da empresa já utiliza um nó centralizador de alta capacidade, Cisco ACE Catalyst 6500, capaz de executar as operações previstas para o escalonamento.

Para que o sistema tenha um balanceamento de carga eficiente, é imprescindível que seja capaz de tratar com cargas heterogêneas pois, conforme visto na Figura 8, o fluxo transacional varia muito ao longo do dia. Além dessa variação, na Tabela 1 estão listadas as transações com maior índice de ocorrência no sistema, sendo que cada uma possui um tamanho em bytes que varia significativamente.

A decisão por um sistema que atenda a heterogeneidade de rede deve ser tomada pois existem condições nas quais nem sempre se pode manter as máquinas alocadas dentro de um mesmo centro de processamento. A perspectiva de expansão para outros países traz consigo necessidades impostas pelas legislações locais que exigem que o centro de processamento de transações seja instalado fisicamente em seus territórios. Neste cenário, as transações originadas de outro país precisam ser registradas no Brasil e processadas novamente no país de origem. Assim, é pertinente que o escalonador seja capaz de lidar com despacho de transações para diferentes domínios.

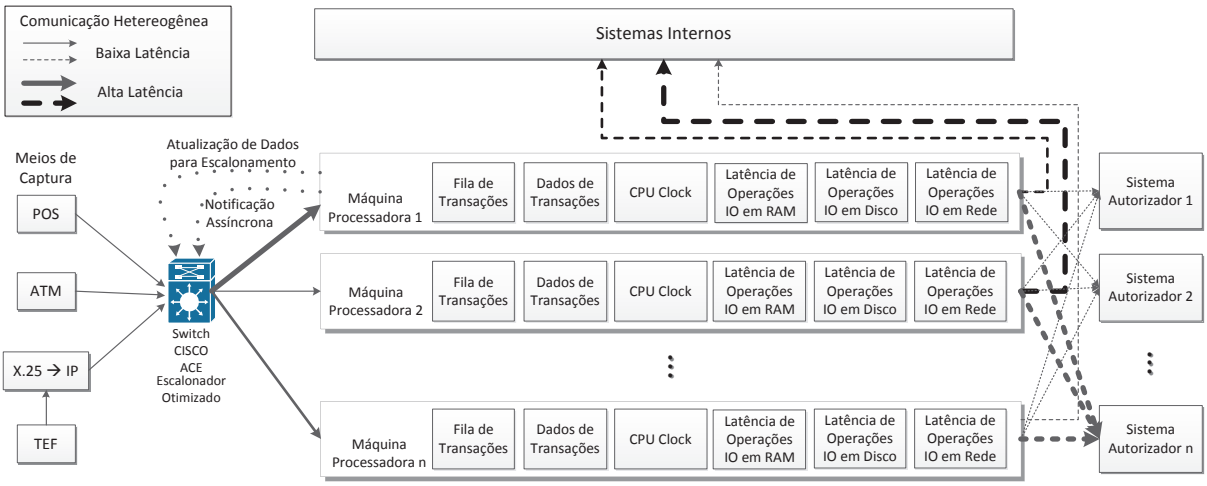
Dado o ambiente dinâmico e heterogêneo que caracteriza o sistema transacional, a decisão recai sobre uma política de balanceamento de carga dinâmico. A escolha por um algoritmo heurístico para o escalonador, se deve ao conhecimento do sistema existente e da carga a que é submetido, permitindo assumir hipóteses realistas.

Conforme os dados obtidos na modelagem formal de sistemas transacionais, Seção 2.4, se reforça a escolha pelo algoritmo heurístico na medida em que se conhece a função de distribuição de probabilidade da carga na entrada no sistema e também dos tempos de serviço das máquinas, permitindo chegar a um algoritmo mais eficiente. Além deste ganho, ao modelar o sistema é possível que as decisões sejam previamente simuladas com auxílio de *software* especializado.

4.2 Arquitetura de Comunicação

A arquitetura proposta para o balanceamento de carga, chamada GetLB, está apresentada na Figura 18. Todos componentes que formam este sistema operam de modo integrado, trocando informações com o escalonador para atualização de dados ou para transações a serem processadas.

Figura 18: Infraestrutura GetLB na qual as Máquinas de Processamento podem ser heterogêneas e localizadas em diferentes domínios de Internet, interagindo com a máquina escalonadora (chaveador Cisco).



Fonte: Desenvolvido pelo próprio autor.

O passo a passo de uma transação nessa arquitetura tem início com os estímulos que partem dos meios de captura, seja POS, TEF ou ATM. Os pacotes de dados chegam ao escalonador através de conexões TCP/IP, pois os diferentes protocolos da origem são convertidos previamente, conforme visto na Seção 2.1. No escalonador, o algoritmo identifica o tipo de transação recebida e o recurso mais capacitado para o processamento. A seguir, ocorre o despacho da transações para a Máquina Processadora. Durante esse processo, as MPs notificam periodicamente o chaveador, informando o estado dos componentes internos.

Depois de serem recebidas nas MPs, as transações são alocadas em uma fila, aguardando para serem consumidas pela aplicação. Ao serem alocadas na fila, as informações são armazenadas em disco, mantendo o rastro das transações. A seguir, quando são retiradas da fila, os dados para processamento são obtidos a partir do protocolo ISO 8583 e armazenados em memória para que possam ser recuperados com agilidade durante todos os passos do processo. Após esses registros, os dados da operação são encaminhados para o respectivo sistema interno. Se a transação é do tipo administrativa, a resposta da solicitação é retornada para o terminal de origem. Para transações financeiras, a solicitação de autorização é transmitida para autorização na empresa emissora do cartão ou de telefonia.

As MPs podem estar alocadas em diferentes domínios da Internet, conforme está representado na Figura 18, na qual estão indicadas, através das legendas, as latências de rede a que ficam sujeitas as máquinas alocadas em outros centros de processamento. Do mesmo modo, o acesso aos sistemas internos também pode estar distribuído em diferentes domínios. Prevendo uma situação como essa, GetLB possibilita que as máquinas processadoras sejam alocadas em outro país, por exemplo, e os sistemas internos continuem sendo utilizados a partir do Brasil, sem necessidade de replicar a estrutura no país em que estão alocadas as MPs.

A partir das MPs, as notificações são transmitidas periodicamente para o escalonador, man-

tendo as informações referentes ao estado de cada uma constantemente atualizadas. Conforme apresentado na Seção 4.1, as notificações também podem ser aperiódicas, alertando o escalonador em situações críticas. Para cada MP, os dados que constam nas notificações reportam ao escalonador (i) a capacidade de processamento da CPU, (ii) o espaço livre em RAM, (iii) o espaço livre em disco, (iv) a ocupação da fila de entrada, (v) a latência de E/S em memória RAM, (vi) a latência de E/S em disco, (vii) a latência de comunicação com SIs e (viii) a latência de comunicação com autorizador externo.

Para as situações de criticidade, que inviabilizem uma MP de receber novas transações, são considerados limites para cada um dos oito itens que formam as notificações. Estes limites serão interpretados no escalonador, pois é neste componente que estão as estratégias de escalonamento que decidem se, para o tipo de alerta recebido, a MP está ou não capacitada para processar o tipo de transação recebida. Contudo, cada MP possui gatilhos internos que antecipam uma notificação, caso sejam acionados.

Dado o conjunto de dados recebidos nas notificações, o algoritmo do escalonador está preparado para realizar balanceamento de carga sempre considerando as questões de heterogeneidade, seja de máquinas ou de comunicação. Essa característica de GetLB busca resolver uma das maiores limitações do sistema atual, que é totalmente homogêneo.

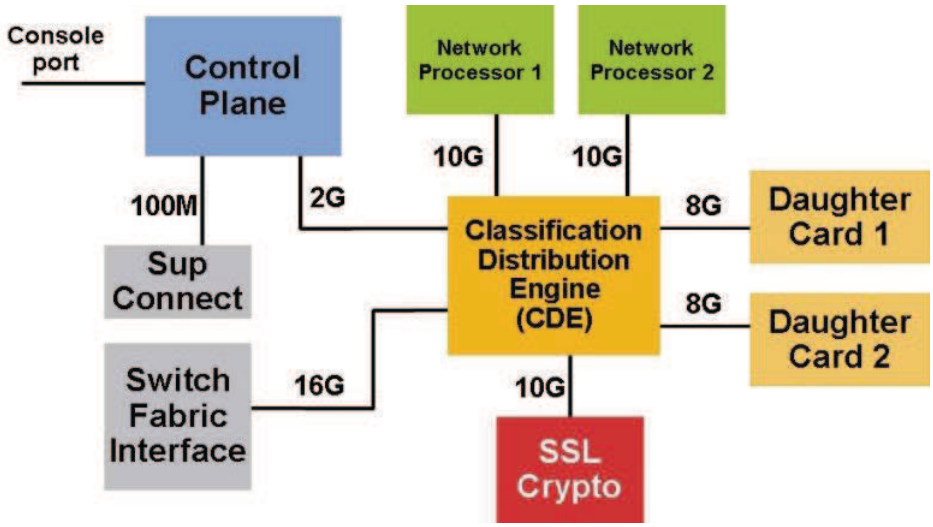
Conforme apresentado na Figura 18, se pode identificar os componentes que constituem GetLB. Partindo da entrada das transações no sistema, até que as mesmas retornem ao meio de captura de origem, são percorridos diferentes equipamentos que precisam ser monitorados regularmente através de atualizações de informações ao chaveador. A partir dessa monitoração e da heurística do escalonador, as transações são despachadas para a máquina mais adequada para o processamento.

O nó centralizador responsável por receber as notificações, executar o algoritmo de balanceamento de carga e despachar as transações é um chaveador Cisco Catalyst 6500 (CISCO SYSTEMS, 2012). Esse equipamento é próprio para executar balanceamento de carga em ambientes que requerem alto desempenho e alta disponibilidade. Esse equipamento conta com algoritmos de balanceamento de carga *default*, mas possibilita que sejam inseridos algoritmos adaptados às necessidades do sistema. Para tanto, devem ser escritos *scripts* que serão interpretados pelo *Application Control Engine*, ou ACE, que controla o equipamento.

Uma visão geral do *hardware* do ACE é apresentada na Figura 19. Nesta arquitetura, cabe destacar alguns dos módulos internos que são mais relevantes para este projeto. O primeiro é o *Control Plane*, que é o módulo que possibilita a conexão de um console externo para configurar, efetuar manutenção e gerenciar o ACE. O módulo *Classification and Distribution Engine* (CDE) é o controlador de tráfego do ACE. Seu principal objetivo é encaminhar pacotes que recebe do chaveador, do qual se originam os pacotes de dados, para os dois processadores de rede. Ele também atua como ponto central de contato entre todos os subsistemas principais dentro da ACE. O CDE calcula e, se necessário, ajusta o *checksums* IP, TCP e UDP de todos os pacotes que recebe. O ACE tem dois processadores de rede ou *network processors* (NP1 e NP2), que

realizam a maior parte do processamento de pacotes. Todo o tráfego de entrada do ACE deve passar através de um ou de ambos NPs, sendo que esses dados são recebidos pelo CDE. Cada NP contém uma CPU (XScale) e diversos componentes chamados *microengines*. Na XScale da NP é onde são executados os algoritmos de balanceamento de carga do ACE.

Figura 19: Visão geral do hardware do ACE.



Fonte: Cisco (2012).

Conforme as análises realizadas nesta etapa do projeto, se considera ser possível portar para o chaveador Cisco o algoritmo do escalonador LL. Contudo, se houver impeditivos na portabilidade do algoritmo, serão propostas outras soluções, mantendo-se a infraestrutura com nó centralizador.

Durante todo o processo transacional, o rastro das transações é armazenado em disco, permanecendo nas máquinas por um período pré-definido até serem removidos para armazenamento em local seguro. Esse procedimento é necessário para evitar que o disco das máquinas não seja todo preenchido com informações de rastro de transações. Os dados que precisam ser frequentemente acessados são mantidos em memória RAM para garantir bom desempenho de leitura e escrita. Desse modo é imprescindível que sempre haja memória livre para efetuar as operações que cada tipo de transação exige.

Assim que as primeiras operações são efetuadas, os dados extraídos da mensagem original precisam ser transmitidos para os respectivos sistemas internos. As questões relativas à latência de rede ocorrem nesta etapa. Todo o tempo que for despendido nesta comunicação reflete diretamente na percepção do usuário do terminal. Além disso, cada instante de tempo demandado nesta comunicação representa menos tempo disponível para as demais etapas do processo transacional, ou seja, mais próximo se está do *time out* do terminal. Desse modo, essa latência de comunicação deve estar sempre sendo monitorada e notificada ao escalonador.

Finalizando o processo, para algumas transações, ocorre o envio para um centro de processamento externo para que a transação seja autorizada. Também nesta etapa as questões pertinentes

Figura 20: Exemplos de transações extraídas do sistema, demonstrando as diferenças de tamanho entre compra crédito, recarga de telefonia e operação administrativa.

```
> 0200B238041121E91080020000000000000080020000000001598512011056230050101056231201022111201
1100000000059376394********9987=160610100000014330910023000384726419900000000008300038477
56394********9987^ALEXANDRE MARTINS  ^160610100000000000000014330035730009001 3#8200E
799CAFB9ECEEEA800002014042009123456777
> 0900A238001101E10084020000001000000890000012011052380066451052381201111201110000000000101
01743834912730000000000850174383004001100006 0@0019@0035@0114@0166@00971000@00413388882320
@0041********@02014042110000000007009123456788
> 0810A238001102C10086000000001000000395000212011309400013931309401201111201110000000000700
20058856628158000000000004002600099 9/0001@M201001902002003002104002205002306002908013309 0
181*0100002#0200003#0300004#0400005#0500006#0609999#0800007#0900008/0002@0036*00017/0003@
M2010174020175*0100009#0200014/0004@M2010026020027*0100030#0200011/0005@0028*09999/0006@M
2010030*0100013/0007@M202013403013504013605013 7*0209999#0309999#0409999#0509999/0008@0182
*09999/0009@0024*00010/0010@0025*00030/0011@M2030158040159*0300030#0400012/0012@0160*0003
0/0013@0033*29999/0014@0041*00015/0015@0041*19999/0016@0097*20014/0017@M20200020501290600
37070035310146*0200018#0500019#0 ...
```

à latência de rede precisam ser monitoradas e notificadas ao escalonador, evitando degradação do desempenho da MP. Contudo, atrasos no processamento do sistema externo podem apenas ser alertados, sendo comunicado à empresa responsável pela autorização para que tome providências para restabelecer o sistema. Cabe ao escalonador evitar o processamento e uso das MPs desnecessariamente com transações para autorizadores que estejam inoperantes.

4.3 LL: Escalonador Otimizado de Transações

Conforme as decisões de projeto, diferente da abordagem tradicional, MP atualizam seus próprios dados, passando mensagens para o módulo de chaveador periodicamente. Ao mesmo tempo, esta última entidade na medida em que recebe as transações, utiliza os dados mais recentes informados pelas MP para decisões de mapeamento. Estas informações referem-se ao estado da fila de operações, informações de carga da CPU e latências para operações de E/S, entre outros. A adequada combinação desses dados, com uma eficiente atualização dos mesmos no escalonador, são chaves para a eficiência da heurística.

Na Figura 20 estão representados alguns exemplos de transações que são recebidas com maior frequência no sistema. Estas transações foram extraídas do rastro transacional da empresa e, como se pode observar, fica clara a diferença que existe na quantidade de dados entre elas. Neste quadro consta, em ordem crescente de quantidade de dados, uma transação de compra com cartão de crédito, uma recarga de telefonia e uma operação administrativa, a qual não está apresentada por completo pois possui tamanho superior a 6.000 bytes. Esse dado enfatiza a heterogeneidade da carga transacional.

Devido ao fato de que o sistema pode operar com máquinas heterogêneas, estas podem estar localizadas em diferentes domínios de Internet, os quais estão acessíveis através do módulo chaveador. Assim, máquinas ligadas diretamente ao chaveador irão apresentar uma latência mais baixa se comparada com aquelas instaladas em um local diferente. Neste último caso, os acesos aos recursos da Internet são penalizados e estão suscetíveis a congestionamento da rede

e, portanto, precisam constar na heurística.

Considerando que tanto as MP quanto as transações podem ser modeladas como heterogêneas, desenvolveu-se uma heurística de programação chamada LL (do inglês, *Load Level*). A heurística LL pode ser vista como uma função de decisão $LL(i, j)$ com dois parâmetros de entrada: i significa um tipo específico de transação, sendo $i = 1, 2, ..., n$ e n o número de tipos de transação, enquanto j representa uma MP alvo, onde $j = 1, 2, ...m$ e m representa o número de MP, para receber a transação i. Considerando isto, o chaveador calcula para cada transação, m equações $LL(i, j)$. O resultado destas equações informa qual, entre as m máquinas, é a mais apropriada para receber a transação i, com base no cálculo do tempo necessário para receber e processar uma dada transação i em uma determinada máquina j, sendo a unidade de medida de LL dada em segundos. $LL(i, j)$ pode ser obtido por meio do cálculo da Equação (4.1).

$$LL(i, j) = Receiving(i, j) + Processing(i, j). \tag{4.1}$$

O primeiro termo da Equação (4.1), $Receiving(i, j)$, considera o tempo necessário para que a MP candidata receba a transação do tipo i originada do chaveador. Desse modo, a Equação (4.2) leva em consideração o número de *bytes* a serem transmitidos através da rede e o tempo para enviar um *byte* entre o chaveador e a máquina de destino.

$$Receiving(i, j) = transfer\_switch\_pm(j) \times bytes\_transaction(i). \tag{4.2}$$

O termo $Processing(i, j)$ corresponde ao tempo de processamento de cada uma das n transações mapeadas e encaminhadas para uma máquina j. Deste modo, este termo pode ser dividido em dois elementos: (i) previsão de computação de uma transação i em uma máquina j e (ii) o tempo de processamento para executar todas as transações que estão na fila de transações de j. O primeiro termo da Equação (4.3) corresponde ao tempo necessário para processar a transação do tipo i na máquina candidata j, enquanto o segundo termo considera o tempo para processar todas as n transações. Desse modo, no segundo termo da equação, o somatório contém um multiplicador $p(z)$, que corresponde à quantidade de transações do tipo z, onde $z = 1, 2, ...n$. Ou seja, o somatório compreende o tempo de processamento da transação do tipo 1 multiplicado pelo número de transações deste tipo, dado por $p(1)$, que estão na fila, somado às $p(2)$ transações do tipo 2 que estão na fila, e assim por diante até a transação n.

$$Processing(i, j) = Execution(i, j) + \sum_{z=1}^{n} [p(z) \times Execution(z, j)]. \tag{4.3}$$

A Equação (4.4) representa uma previsão de computação para uma transação do tipo i sobre a máquina de processamento j e aos respectivos sistemas internos associados. Ela é composta

por quatro termos: o primeiro designa operações da CPU; o segundo e o terceiro as operações de E/S; e o quarto refere-se a subsistemas. Cada transação é definida pelos seguintes parâmetros: (i) número de instruções; (ii) o número de operações de memória RAM; (iii) o número de operações de disco (HD); (iv) os subsistemas envolvidos no processamento da transação e do número de interações com cada um; (v) o tempo de acesso para cada subsistema. Considerando-se que a utilização de rede local (LAN) não é mandatória, no referido parâmetro (v) é útil para adicionar o tempo de acesso aos subsistemas, distribuídos ao longo de diferentes domínios da Internet. Na Equação (4.4), o termo $serviceRAM(j)$ e $serviceHD(j)$ designam tempo de E/S para a realização de uma única instrução de escrita em memória e disco, respectivamente. Os parâmetros $FreeRAM(j)$ e $freeHD(j)$ são percentuais e indicam a disponibilidade de cada recurso de E/S em um determinado momento.

$$Execution(i,j) = \frac{instrutions(i)}{clock(j) \times (1 - load(j))} + \frac{RAM\_operations(i)}{serviceRAM(j) \times freeRAM(j)} + \\ + \frac{HD\_operations(i)}{serviceHD(j) \times freeHD(j)} + subsystems(i,j). \quad (4.4)$$

O tempo consumido nos subsistemas é calculado por $subsystems(i,j)$, em conformidade com a Equação (4.5). Cada tipo de transação i deve acessar x subsistemas. Assim, $sub\_access(y,j)$ considera o tempo despendido pela MP j para acessar o subsistema específico y, através da comunicação de rede. Desta vez é multiplicado por dois, considerando assim as mensagens de solicitação e a resposta, próprio da interação das MP com cada subsistema. O campo $sub\_compute(y)$ se refere ao tempo de serviço do subsistema y. Finalmente, $sub\_interactions(i,y)$ se refere ao número de vezes que o subsistema y é chamado para o processar a transação i.

$$subsystems(i,j) = \sum_{y=1}^{x} [(2 \times p(y) \times sub\_access(y,j)) + sub\_compute(y)] \times sub\_interactions(i,y). \quad (4.5)$$

Cada MP atualiza periodicamente seus dados de $clock$, carga de CPU, estados de memória RAM e HD, bem como os tempos de acesso aos subsistemas e chaveador, utilizando um canal de comunicação unidirecional para o chaveador. O chaveador utiliza duas tabelas para auxiliar no processo de balanceamento: (i) os tipos de transações e suas respectivas demandas; e (ii) os subsistemas com tempos de processamento. Portanto, os dados necessários para o $bytes\_transaction(i)$, $instructions(i)$, $RAM\_operations(i)$, $HD\_operations(i)$, $sub\_compute(y)$ e $sub\_interactions(i,y)$ são fixos e obtidos por meio de consulta dessas tabelas.

4.4 Notificações

O envio regular de dados de balanceamento das MP para o módulo chaveador já representa uma maneira de interromper a utilização de uma máquina específica. Isso ocorre porque o ACE sempre escolhe a máquina mais adequada para receber uma transação, sem enviar carga para recursos com alta carga de CPU, pouca memória RAM ou de disco disponível. No entanto, esta interação existe de uma forma periódica. De outro modo, as notificações são usadas para garantir disponibilidade do sistema em situações dinâmicas, ou seja, são eventos disparados a partir das MP sem respeitar o período da próxima atualização do chaveador. Considerando isso, foram modelados cinco tipos de operações que podem ser vistas na Tabela 8. Todas as notificações de categoria "indisponível", sinalizam ao chaveador que deve ser interrompida a transferência de transações para a máquina que transmitiu a mensagem. A notificação do tipo 1, manutenção, é utilizada para realizar manutenções de *hardware* ou componentes de *software*, sem gerar indisponibilidade no sistema. Quando a MP estiver novamente operacional, o administrador pode inserir a máquina de volta ao grupo de máquinas.

As notificações 2 e 3 da Tabela 8 são utilizadas para tratar um tipo específico de transação. Particularmente, a notificação 2 pode ser empregada quando uma máquina apresentar elevado volume de perda para um tipo específico de transação. Por exemplo, uma transação do tipo A deve acessar subsistemas x e y, contudo a MP não consegue estabelecer comunicação de rede com tais subsistemas. O tipo 4 de notificação é utilizado quando o número de elementos na fila de transações das máquinas estiver elevado. Este mecanismo é para auto-proteção de uma MP, evitando maior enfileiramento de transações, o que poderia levar o sistema a uma degradação no tempo de processamento das transações ou até levar a máquina a ficar indisponível. Por fim, a notificação 5 pode ser enviada automaticamente quando a MP voltar a apresentar um nível aceitável de ocupação de sua fila de transações. Além do controle da fila de entrada, a notificação do tipo 4 também pode ser utilizada para solicitar interrupção de serviço por problemas de capacidade de processamento ou armazenamento.

A utilização das notificações torna GetLB capaz de um comportamento elástico para o sistema transacional, ou seja, as máquinas podem ser inseridas ou removidas da infra-estrutura, seja conforme pedido do administrador ou de forma proativa, sem intervenção do usuário. As notificações de 1 a 5 são planejadas para serem disparadas conforme sejam atingidos valores específicos, ou gatilhos, mapeados para cada situação. As notificações também são importantes para a decisão de gestão sobre o fluxo transacional, entre as quais se pode destacar duas: (i) se o chaveador recebe notificação 4 das máquinas com maior capacidade de processamento, é possível concluir que a infraestrutura tenha sido subestimada ou algum subsistemas estão agindo como gargalo do sistema; e (ii) sob situações anormais, as notificações 2 e 3 são pertinentes para a manutenção do processamento de um conjunto pré-definido de tipos de transação, conforme as necessidades de provedores de serviços.

Tabela 8: Tipos de notificação utilizadas por GetLB.

Tipo		Notificação	Origem	Descrição
1	Manutenção	Indisponível	Administrador	Manutenção programada pelo administrador da rede, possibilitando desligar a máquina sem perder transações.
2	Controle de tráfego	Indisponível	Automático ou Administrador	MP solicita para chaveador interromper o envio de um tipo específico de transação nas próximas requisições. O tipo de transação é utilizado como parâmetro de entrada.
3	Controle de tráfego	Disponível	Automático ou Administrador	MP reverteu a decisão de notificação 2, tornando assim possível receber novamente transações de um tipo específico (parâmetro de entrada).
4	Inoperante	Indisponível	Automático	Proativamente, MP solicita ao chaveador para interromper o envio de transações.
5	Operacional	Disponível	Automático ou Administrador	A máquina entra em modo operacional, retornando ou iniciando suas atividades como processamento de equipamento de sistema transacional. Neste momento, ele pode receber transações de todo o tipo.

Fonte: Desenvolvido pelo próprio autor.

4.5 Analisando o Balanceamento de Carga

Esta subseção apresenta uma compilação das definições de projeto, conforme as explanações da subseção anterior. Assim, nos próximos parágrafos as decisões tomadas serão agrupadas, tornando o embasamento teórico mais claro para o leitor.

Categorizando as definições de GetLB, segundo Waraich (2008), cuja teoria foi analisada na Subseção 2.3.1 do capítulo de referencial teórico, os três principais parâmetros que estabelecem a estratégia do escalonador são as seguintes:

i. *"quem"*: ***symmetrically-initiated***, pois o despacho das transações se dá a partir do escalonador para as máquinas processadoras e, no sentido inverso, notificações atualizam o escalonador a partir das máquinas processadoras;

ii. *"qual"*: **informações globais**, pois para realizar o balanceamento serão utilizadas in-

formações tanto das transações como de todo o conjunto das máquinas processadoras, recebidas das notificações;

iii. *"onde"*: **híbrido** com chaveador Cisco Catalyst 6500 utilizando informações das MP.

Quanto às políticas de balanceamento de carga de GetLB, conforme (KWOK; CHEUNG, 2004), as quatro dimensões se caracterizam da seguinte maneira:

i. *Localização*: **seleção dinâmica** pois se dá em tempo de execução;

ii. *Informação*: a informação é obtida de maneira **assíncrona** combinando orientação à demanda e à mudança de estado;

iii. *Transferência*: **mista** pois tanto as máquinas processadoras como o escalonador transmitem e recebem dados;

iv. *Seleção*: **não preemptiva** pois o processamento de uma transação não pode ser migrado.

Conforme a taxonomia propostas por Casavant e Kuhl (1988), GetLB possui um tipo de escalonamento **global** pois utiliza informações do sistema para alocar tarefas em vários processadores. Além disso, GetLB se caracteriza por ser **dinâmico**, pois a relação entre as transações recebidas e os dados dos recursos da arquitetura são imprevisíveis, assim como a decisão do mapeamento consumidores-recursos se dá em tempo de execução. Quanto à distribuição, o sistema é do tipo **fisicamente distribuído**, pois o encargo do escalonamento global é desempenhado por um conjunto de processadores. O mecanismo de cooperação entre os componentes distribuídos é do tipo **cooperativo**, pois cada processador tem a responsabilidade de realizar a sua própria parte do escalonamento, trabalhando em direção a um objetivo comum de todo o sistema. O algoritmo de escalonamento é do tipo **sub-ótimo**, não garantindo soluções ótimas para um problema do tipo NP-hard, que levariam a uma computação impraticável. Por fim, o tipo de algoritmo sub-ótimo adotado é **heurístico** pois se assumem hipóteses realistas sobre o conhecimento existente e as características de carga do sistema. Portanto, a taxonomia de Casavant para GetLB segue a seguinte categorização:

- **Escalonamento: Global - Dinâmico - Fisicamente Distribuído - Cooperativo - Sub-Ótimo - Heurístico**

Desse modo, se conclui a categorização de GetLB conforme a literatura analisada.

4.6 Análise do Capítulo

Através de uma análise do atual sistema de processamento de transações, responsável pelo processamento de cartões regionais, de telefonia e operações administrativas da empresa GetNet, frente às perspectivas de crescimento dos negócios da empresa para os próximos anos, o capítulo iniciou descrevendo as lacunas e necessidades de um projeto de balanceamento de

carga. Estas lacunas motivaram o projeto de uma nova infraestrutura para balanceamento de carga, denominada GetLB. É importante ressaltar que as lacunas identificadas no sistema transacional não comprometem o desempenho do sistema atual, contudo, podem oferecer restrições para um crescimento da base nos próximos anos.

Como proposta para suprir as necessidades sistêmicas, o capítulo apresentou a infraestrutura GetLB. Esta nova infraestrutura foi descrita com base nas referências teóricas dos capítulos anteriores, conferindo um embasamento científico para as decisões de arquitetura e estratégia do escalonador. As principais características dessa nova infraestrutura estão na capacidade de lidar com a heterogeneidade dos recursos, da carga e das comunicações, além das notificações assíncronas. Em consequência, torna possível ao sistema poder priorizar transações e encaminhar transações para grupos de recursos especializados em determinado tipo de operação. Tais características também oferecem ao sistema a possibilidade de utilizar uma máquina com menos capacidade de processamento para as transações menos onerosas.

5 PROTÓTIPO GETLB

Este capítulo descreve a estrutura e o funcionamento do protótipo que foi desenvolvido como prova de conceito das decisões de projeto e para validação da heurística de balanceamento de carga. Além desta validação, o protótipo também possibilita a realização de ajustes no modelo da heurística. O protótipo foi desenvolvido em linguagem Java e gera arquivos com rastro das atividades simuladas. Estes arquivos disponibilizam informações que permitem analisar o algoritmo de GetLB com relação ao seu desempenho e também comparar os resultados frente ao algoritmo Round-Robin.

O capítulo inicia com a Seção 5.1, onde serão descritas as classes do protótipo. A seguir, na Seção 5.2, serão expostas as interações entre os processos. Na Seção 5.3, será apresentado o funcionamento do escalonamento e na Seção 5.4 é feito um balanço do que foi exposto ao longo das seções do capítulo.

5.1 Interface de Programação

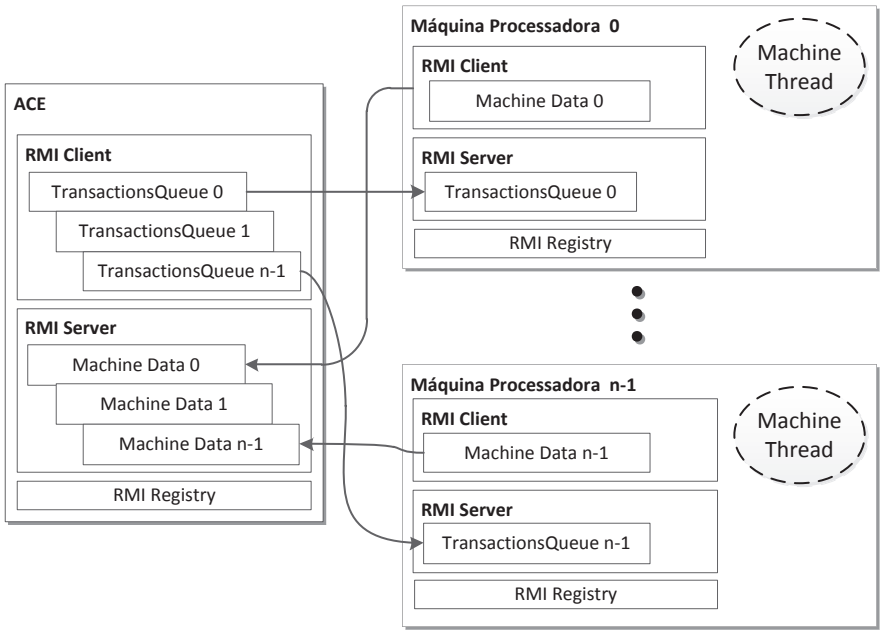
A arquitetura GetLB, conforme descrito no capítulo anterior, consiste em um chaveador que recebe as transações originadas de diferentes tipos de terminais. A empresa GetNet utiliza um equipamento Cisco Catalyst 65000, que concentra e distribui as transações entre um conjunto de máquinas processadoras. Desse modo, ao implementar um protótipo de GetLB com as características mais próximas possíveis dessa arquitetura, se pode contar com recursos acessíveis para exploração e análise científica da heurística escolhida para GetLB, bem como refinar as equações. Por fim, também permite que sejam feitos testes e análises comparativas entre o desempenho de GetLB e do tradicional algoritmo escalonador Round-Robin.

A implementação do protótipo foi desenvolvida com o *middleware* RMI pertencente ao *kit* de desenvolvimento do Java, versão 1.7. Apesar dessa linguagem não ser voltada para alto desempenho, o seu emprego nesse contexto foi pertinente para avaliar os possíveis ganhos de funcionamento e a sobrecarga de GetLB se comparado com a abordagem tradicional para processamento de transações eletrônicas. A ideia do arcabouço passa pela criação de objetos remotos tanto na máquina comutador quanto em cada MP. Tal organização está ilustrada na Figura 21.

A classe ACE é onde são implementados os algoritmos de escalonamento, tanto GetLB quanto Round-Robin. Esta classe recebe como entrada três arquivos texto: um com as transações a serem processadas; outro descrevendo os tipos de transações com suas características de consumo; e um terceiro descrevendo as máquinas processadoras e suas atribuições. Esta classe também cria, para cada máquina processadora, o objeto remoto servidor **MachineData**, para coletar as informações das MPs e aplicar nos algoritmos de decisão de escalonamento.

As transações são obtidas a partir de um arquivo de texto denominado **Transactions**. Neste estão todas as transações obtidas a partir de um rastro transacional dos sistemas da GetNet, con-

Figura 21: Protótipo de GetLB implementado com RMI.



Fonte: Desenvolvido pelo próprio autor.

Figura 22: Intervalo do arquivo de transações de entrada do sistema, contendo horário de chegada e tipo de transação.

```
22319|2
29585|2
33506|3
38897|2
75380|5
86614|1
88583|2
102114|4
103739|2
```

Fonte: Desenvolvido pelo próprio autor.

forme a Figura 22. Neste arquivo constam apenas as informações do horário de chegada, dado em milissegundos a partir da 00:00:00 do início do dia e do tipo de transação. Para esse intervalo exemplificado, se verifica a primeira transação sendo recebida aos 22.319 milissegundos, ou 00:00:22.319 do tipo 2, que corresponde à recarga de telefonia, como veremos a seguir. A próxima transação, também de recarga de telefonia, foi recebida em 00:00:29.585, e assim por diante.

O protótipo utiliza um arquivo texto auxiliar, denominado TransactionType, que descreve o tempo de processamento e o consumo de memória para cada tipo de transação, baseado nos levantamentos obtidos de um sistemas transacional real. Um exemplo desse arquivo está demonstrado na Figura 23. Neste arquivo a primeira coluna corresponde ao código identificador da transação, sendo 1-administrativa, 2-recarga de telefonia, 3-compra com cartão, 4-reversão e 5-estorno. A segunda coluna informa o número médio de ciclos de CPU necessários para pro-

Figura 23: Exemplo de arquivo de texto com as informações das transações recebidas no sistema.

```
#id|ciclos|RAMio|size
1|30|325|6700
2|42|120|1146
3|107|250|1322
4|83|220|348
5|102|250|1212
```

Fonte: Desenvolvido pelo próprio autor.

Figura 24: Exemplo de arquivo de texto com as informações das transações recebidas no sistema.

```
#id|host|freq_clock|t_operRAM|t_aces_sub|queue_size|transfer_byte|memory_size
0|10.16.10.1|2.4|2|1200|120|1|2074856
1|10.16.10.2|2.4|2|1200|120|1|2074856
2|10.16.10.3|2.4|2|1200|120|1|2074856
3|10.16.10.1|2.4|2|1200|120|1|2074856
4|10.16.10.2|2.4|2|1200|120|1|2074856
5|10.16.10.3|2.4|2|1200|120|1|2074856
```

Fonte: Desenvolvido pelo próprio autor.

cessar a transação. A coluna RAMio informa o tempo em milissegundos de acesso à memória RAM que a transação necessita e a última coluna contém o tamanho médio da transação, dado em número de *bytes*.

A classe **Machine** representa cada uma das máquinas processadoras do protótipo. As máquinas inseridas no protótipo possuem uma configuração que é originada do arquivo de texto auxiliar **ProcessingMachine**, conforme descrito na Figura 24. Neste arquivo, a primeira coluna contém a identificação da máquina processadora e na segunda coluna é informado o endereço IP da máquina. A terceira coluna descreve o *clock* da CPU e na quarta o tempo de operação da RAM, dado em milissegundos. A quinta coluna informa o tempo de aceso aos subsistemas e a sexta contém o tamanho da fila de transações. Na sétima coluna está o tempo, em milissegundos, para transferir um *byte* do ACE para a máquina e, na última coluna, é dado o tamanho da memória RAM.

Ainda na classe **Machine**, para cada MP é criado um servidor RMI **TransactionQueue**. A classe ACE faz um *lookup* para a **TransactionQueue** de cada MP, e através destas filas, é feito o despacho das transações. A classe **TransactionQueue** corresponde a um *array* de objetos do tipo **Transaction**. Esta classe é responsável por conter informações da transação a ser tratada, tais como número de instruções, quantidade de operações de E/S e ocupação de memória. Isso disponibiliza os dados necessários para que as máquinas processadoras possam emular a execução da transação.

A classe **Machine** também faz um *lookup* para o respectivo servidor **MachineData** no ACE e cria a *thread* **MachineThread**. Essa classe é responsável por coletar as informações de cada máquina, alimentando o **MachineData** correspondente no ACE. Os dados coletados, que serão encaminhados para o ACE, possibilitam o cálculo de LL para a função de balanceamento, ou seja, o tamanho da fila de transações, CPU e memória livre, entre outras. A quantidade de CPU

e memória RAM livres são obtidas através do comando *top* do Linux. O tempo de atualização da **MachineThread** é configurável e possibilita que seja ajustado para encontrar o equilíbrio entre *overhead* de comunicação na rede e informações atualizadas no tempo necessário.

O consumo de transações da fila das MPs segue o regime FIFO e, para emular a execução das transações, a classe **Machine** também cria uma *thread* do tipo **ProcessThread** para cada transação recebida. Pare ser fiel ao comportamento de um sistema transacional, esta *thread* aloca a quantidade de dados em memória equivalente ao tipo da transação. O tempo de execução de uma transação será proporcional aos seus ciclos de máquina obtidos do arquivo de texto auxiliar **Transactions**.

O protótipo desenvolvido não se comunica com sistemas de saída até o momento, contudo, ele proporciona meios úteis para analisar o mapeamento em relação a dados de rede, subsistemas e máquinas processadoras.

5.2 Interações Entre Processos

Nesta seção serão descritas as interações entre os processos RMI, contemplando o despacho de transações do ACE para as MPs e atualização do estado das máquinas no chaveador, bem como as notificações assíncronas partindo das MPs para o chaveador.

O protótipo é iniciado com a criação da máquina ACE, responsável pelo algoritmo de balanceamento, e das MPs que irão processar as tarefas recebidas. Para isso, assim que iniciadas, estas máquinas criam objetos remotos para possibilitar os processos de troca de informações entre as partes. O ACE cria um servidor RMI **MachineData** para cada máquina processadora do sistema e, cada MP, faz um *lookup* para seu respectivo objeto, mantendo assim um cliente RMI **MachineData**. Ou seja, esse modelo cliente-servidor se dá com as MPs atuando como clientes e o ACE como servidor. Isso permite que as informações de estado das máquinas e notificações sejam atualizadas no ACE. Cada MP, por sua vez, criam servidor RMI **TransactionQueue** e o ACE faz um *lookup* para a **TransactionQueue** de todas as MPs. Esse mecanismo permite o processo de despacho das transações num modelo de cliente-servidor, tendo o ACE como cliente e as MPs como servidoras.

Depois de estabelecidas as condições para os processos de comunicação, o ACE se configura com base nos dados obtidos dos arquivos de configuração das transações e de máquinas processadoras. Com essas informações armazenadas, o chaveador inicia a retirada de transações do arquivo de rastro. O ACE atua como chaveador e utiliza as informações disponibilizadas por cada MP no servidor RMI **MachineData** para aplicar nas equações de LL. O resultado dos LL calculados serão comparados e aquele de menor valor será o escolhido. Conhecida a MP mais adequada para o tipo de transação que está sendo tratada, o ACE insere em **TransactionQueue** o objeto **Transaction**, contendo as informações desta transação que são necessárias para a MP processar a tarefa.

As transações inseridas na fila **TransactionQueue** serão retiradas pela *thread* **ProcessTh-**

read e emuladas na MP de destino. Se a quantidade de transações inseridas na fila for maior que a capacidade de atendimento do processo de consumo, ocorre um aumento de tarefas na fila. O tamanho desta fila constitui um parâmetro que é monitorado constantemente pela *thread* **MachineThread**. Além deste parâmetro, outros dados pertinentes ao estado da MP são coletados e disponibilizados em **MachineData**, que é disponibilizada para o ACE utilizar no algoritmo de balanceamento.

Os processos descritos acima correspondem à troca de informações entre o chaveador e as máquinas processadoras. Eles contemplam tanto o despacho das transações quanto a atualização das informações de estado das MPs. Este processo de atualização ocorre periodicamente, de acordo com uma frequência parametrizada. Este processo é, portanto, síncrono.

Para que o protótipo suporte o uso das notificações, descritas na Seção 4.4, a comunicação entre os processos precisa ser assíncrona. Mais especificamente, as MPs devem ser capazes de notificar o ACE, através do **MachineData**, em momentos que estejam fora do período de atualizações síncronas. Para isso, **MachineThread** precisa monitorar limites de recursos considerados de risco para o processamento adequado das transações. Entre esses recursos se destaca (i) o tamanho da fila, (ii) a memória RAM disponível e (iii) o uso de CPU. Desse modo, se durante a monitoração, **MachineThread** identificar que algum dos recursos está próximo de seu limite de indisponibilidade, uma notificação deve ser enviada para o ACE a fim de que seja interrompido o envio de tarefas para a MP sinalizada.

Além do processo de notificações por monitoração de eventos de risco, o protótipo possibilita que notificações de manutenção também sejam enviadas ao ACE. Um arquivo de texto auxiliar informa se alguma máquina precisa entrar em manutenção e a partir de que momento o chaveador deve interromper o envio de transações para ela. De modo análogo, para que uma máquina retorne ao sistema, um arquivo de texto auxiliar informa que uma nova máquina irá iniciar o processamento de transações e a partir de que momento o chaveador deve começar a transmissão de tarefas para essa MP.

5.3 Escalonamento

Esta seção descreve como é feito o escalonamento no protótipo GetLB, apresentando trechos do código fonte que descrevem o cálculo do nível de carga e também exemplos de cálculo e seleção da MP para receber transações em um conjunto de máquinas processadoras.

O protótipo foi implementado para executar o algoritmos GetLB e Round-Robin, sendo que através de um comando de entrada do ACE se define qual algoritmo será executado. Considerando que o Round-Robin é um algoritmo sem complexidade, que distribui tarefas de modo circular, esta seção irá descrever apenas o método de balanceamento do algoritmo GetLB.

A Figura 25 apresenta um trecho de código do ACE no qual é efetuado o cálculo do nível de carga das máquinas candidatas. Entre as linhas 2 e 17 são reunidas as informações de estado de cada MP e da transação a ser despachada. Estes dados são atualizados no ACE periodicamente,

Figura 25: Trecho do código fonte da classe ACE que calcula o nível de carga das MPs.

```
01. long t1 = System.nanoTime();

02. double numberInstructions = transaction.getNumberInstructions();
03. double ioOperations = transaction.getIoOperations();

04. double clock = machineData[i].getClock(i);
05. double freeCpu = machineData[i].getFreeCPU();
06. freeCpu = freeCpu / 100;

07. double operationTime = machineData[i].getOperationTime(i);
08. double freeMemory = machineData[i].getFreeMem();
09. double totalMemory = machineData[i].getTotalMem(i);
10. freeMemory = (freeMemory * 100)/totalMemory;
11. freeMemory = freeMemory / 100;

12. double cpu = numberInstructions / (clock * freeCpu);
13. double memory = (ioOperations * operationTime) / freeMemory;

14. double numberBytes = transaction.getMemorySize();
15. double service = machineData[i].getSubSystemTime(i);
16. double transferByte = machineData[i].getTransferByte(i);
17. double waitingTime = machineData[i].getWaitingTime();

19. temp = (numberBytes * transferByte) + cpu + memory + service + waitingTime;

20. long t2 = System.nanoTime();

21. time += t2 - t1;
```

Fonte: Desenvolvido pelo próprio autor.

de acordo com intervalo de tempo parametrizável, sendo que as informações utilizadas neste trecho de código encontram-se armazenadas no chaveador.

Para cálculo do *overhead* de algoritmo, ou seja, quantificar o tempo a mais que é despendido por GetLB em relação ao Round-Robin, na linha 1 é medido o tempo de início de processamento. Na linha 20, após a conclusão do cálculo, o tempo é novamente medido e, na linha 21, se calcula a diferença entre o início e o fim do processo de cálculo de LL. O mesmo procedimento é feito com o algoritmo Round-Robin.

O resultado deste cálculo pode ser visto na Figura 26. Neste exemplo, extraído do arquivo de rastro gerado pelo protótipo durante o processamento de um lote de transações, as transações de número 718 e 719 são analisadas para um conjunto de máquinas processadoras homogêneas. As linhas deste rastro iniciam com o número da transação que está sendo processada, depois é apresentada a equação, com $LL(i, j)$, sendo i a identificação da transação recebida e j a MP candidata. O primeiro termo da equação contempla o tamanho em *bytes* da transação recebida multiplicado pelo tempo de transferência de um *byte* desde o ACE até a MP candidata. O segundo termo considera o peso de processamento da transação e a capacidade de CPU da máquina. No terceiro termo, a capacidade de memória da MP e a demanda do tipo de transação recebida são calculadas. O quarto termo descreve o tempo para acesso a subsistemas e, por último, é considerado o tempo de processamento das transações que estão em fila.

Os dados utilizados para cálculo de LL são todos locais, ou seja, as informações armazena-

Figura 26: Exemplo de cálculo de LL extraído do arquivo de rastro do protótipo. Cálculo aplicado para duas transações em seis MPs candidatas

```
Id ;Equacao
718;LL(718,0)= 6700 * 1.0 + 30/(2.4 * 0.5) + (1.0 * 1.0)/0.9905 + 10.0 +  0.0= 6736.0096 <<<
718;LL(718,1)= 6700 * 1.0 + 30/(2.4 * 0.5) + (1.0 * 1.0)/0.9888 + 10.0 +  0.0= 6736.0113
718;LL(718,2)= 6700 * 1.0 + 30/(2.4 * 0.39)+ (1.0 * 1.0)/0.9880 + 10.0 +  0.0= 6743.0634
718;LL(718,3)= 6700 * 1.0 + 30/(2.4 * 0.5) + (1.0 * 1.0)/0.9882 + 10.0 +  0.0= 6736.0119
718;LL(718,4)= 6700 * 1.0 + 30/(2.4 * 0.44)+ (1.0 * 1.0)/0.9939 + 10.0 +  0.0= 6739.4152
718;LL(718,5)= 6700 * 1.0 + 30/(2.4 * 0.5) + (1.0 * 1.0)/0.9832 + 10.0 +  0.0= 6736.0170
719;LL(719,0)= 1146 * 1.0 + 42/(2.4 * 0.46)+ (1.0 * 1.0)/0.9885 + 10.0 +  0.0= 1195.0551
719;LL(719,1)= 1146 * 1.0 + 42/(2.4 * 0.5) + (1.0 * 1.0)/0.9888 + 10.0 +  0.0= 1192.0113
719;LL(719,2)= 1146 * 1.0 + 42/(2.4 * 0.39)+ (1.0 * 1.0)/0.9880 + 10.0 +  0.0= 1201.8839
719;LL(719,3)= 1146 * 1.0 + 42/(2.4 * 0.5) + (1.0 * 1.0)/0.9940 + 10.0 +  0.0= 1192.0060 <<<
719;LL(719,4)= 1146 * 1.0 + 42/(2.4 * 0.44)+ (1.0 * 1.0)/0.9939 + 10.0 +  0.0= 1196.7766
719;LL(719,5)= 1146 * 1.0 + 42/(2.4 * 0.5) + (1.0 * 1.0)/0.9832 + 10.0 +  0.0= 1192.0170
```

Fonte: Desenvolvido pelo próprio autor.

das no ACE, mais especificamente **MachineData**, é que são recuperadas para as equações. Esse procedimento torna a consulta mais rápida, sem necessidade de acesso às MPs a cada rodada de cálculo. A frequência com que estes dados serão atualizados é parametrizável.

Acompanhando as equações da Figura 26, o primeiro termo da equação, que corresponde ao tamanho da transação, identifica que a transação de número 718 deste exemplo é do tipo Administrativa, e conforme os resultados dos cálculos de LL para o conjunto de máquinas 0 até 5, a MP 0 é que vai receber a transação pois apresentou o menor nível de carga. A seguir, a transação de número 719, que é do tipo Recarga de Telefonia, será endereçada para MP 3, que possui o menor resultado de LL no conjunto.

5.4 Análise do Capítulo

Ao longo deste capítulo foram apresentadas as características e o modo como opera o protótipo GetLB. Durante a explanação sobre a construção das classes e da interação entre os processos, procurou-se demonstrar que o foco do desenvolvimento deste modelo consiste em proporcionar uma ferramenta para analisar a heurística e para avaliar o desempenho de GetLB comparado ao Round-Robin. Nessa mesma direção, os processos e a comunicação entre os módulos procurou ser mais fiel possível ao modelo real. Para isso foram utilizadas características reais das transações que são processadas em um sistema transacional, bem como atributos das máquinas processadoras, com seus tempos de resposta, filas de transações e recursos críticos que precisam ser monitorados.

Na interação entre os processos destacou-se que, apesar das atualizações das MPs para o ACE serem síncronas, também estão previstas sinalizações assíncronas. Estas sinalizações viabilizam a utilização das notificações previstas nas decisões de projeto, proporcionando manutenção sistêmica e prevenção de perdas transacionais.

Também foi enfatizado que o escalonamento é feito com base em dados locais, atualizados periodicamente por processos que são executados nas máquinas processadoras. A utilização

de dados sempre atualizados garante um balanceamento mais adequado, porém o *overhead* de comunicação precisa ser controlado para evitar congestionamento de dados. Para tanto, o tempo de atualização é parametrizável.

A implementação do protótipo com RMI possibilitou a análise da heurística e também a implementação de processos de comunicação conforme planejado nas decisões de projeto. Contudo, em um sistema transacional, o RMI não pode ser aplicado devido às necessidades de alto desempenho de um sistema real, sujeito a picos de transações cada vez mais elevados e, aliado a isso, com tipos diferentes de transações que demandam maior poder de processamento. Porém GetLB é viável na medida em que possibilita avaliar de maneira apropriada as questões relacionadas ao escalonamento.

6 AVALIAÇÃO DE GETLB

Esse capítulo apresenta as estratégias que foram utilizadas para avaliar GetLB. Para tal, ele apresenta os resultados de GetLB sendo executado em diferentes arquiteturas. Serão apresentados cenários que também irão possibilitar uma comparação de desempenho entre o balanceamento de carga de GetLB e o tradicional método Round Robin. Para isso, serão simuladas situações que ocorrem no quotidiano de uma empresa de captura e processamento de transações financeiras. Os cenários também visam possibilitar testes com situações de dinamicidade e heterogeneidade, para os quais o Round Robin não consegue contornar, ao passo que GetLB alcança bons resultados e evita perda de transações. Também serão feitos testes para demonstrar o uso das notificações em situações de exceção, com risco de perda de transações, e para fins de manutenção no sistema.

O presente capítulo é segmentado em sete seções. A primeira, Seção 6.1, descreve a metodologia adotada para a condução dos testes e fundamenta os cenários escolhidos. A Seção 6.2 apresenta os resultados obtidos ao efetuar balanceamento de carga em uma topologia homogênea. A Seção 6.3 mostra os resultados ao implementar uma topologia heterogênea, avaliando o desempenho dos algoritmos de GetLB e RR. Na Seção 6.4 será avaliado o desempenho dos dois algoritmos quando ocorrem eventos dinâmicos no sistema. A seguir, na Seção 6.5, será feita uma análise dos algoritmos quando ocorre indisponibilidade de algum dos componentes do sistema. As notificações serão abordadas na Seção 6.6. Finalmente, a Seção 6.7 faz uma análise dos principais tópicos abordados nesse capítulo.

6.1 Metodologia de Testes

Os testes foram executados utilizando a infraestrutura disponibilizada nos laboratórios de informática do PIPCA da UNISINOS. Para estes ensaios foram utilizadas sete máquinas, sendo uma máquina para escalonamento, fazendo a função do ACE e seis para a função de processamento de transações. Esta configuração é muito próxima da adotada no sistema transacional da empresa GetNet.

A entrada de dados utilizada para alimentar o sistema foi obtido a partir dos sistemas transacionais da empresa GetNet, preservando dados sigilosos, tais como nomes de usuários, números de cartões e valores de compra. Apenas o tipo de transação e o horário de entrada foram utilizados. O dia escolhido para obter estas informações corresponde ao dia de maior volume transacional do ano, no qual os sistemas da empresa foram mais exigidos, sendo o total de transações deste período igual a 8168. Cada tipo de transação recebida no sistema possui suas próprias características de carga e consumo dos recursos. Para alimentar o sistema com as propriedades de cada transação, foi utilizada a Tabela 1, apresentada no Capítulo 2.

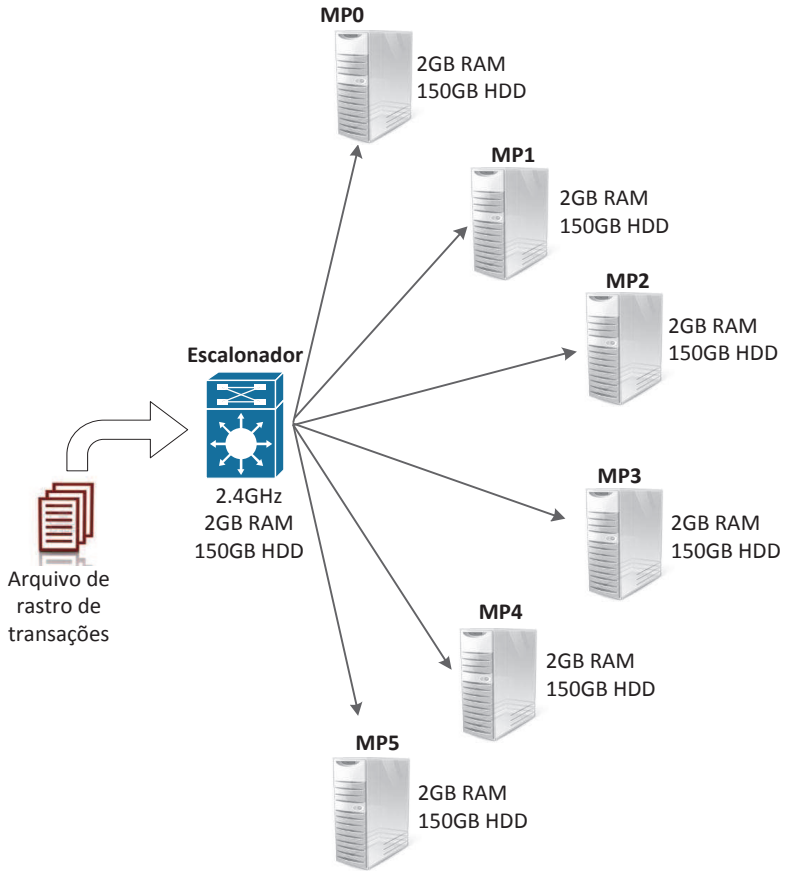
O objetivo dos testes é validar o comportamento do algoritmo de escalonamento GetLB frente ao algoritmo Round-Robin em diferentes arquiteturas e situações operacionais. Estas

situações precisam refletir o dia a dia de um sistema transacional, não apenas com o regime normal de processamento, mas também situações em que ocorrem indisponibilidades, enfileiramento de transações e degradação de serviço de atendimento. Além de simular estas situações que refletem o quotidiano, também precisam ser abordados os cenários de heterogeneidade para os quais GetLB se propõe a atender.

A topologia utilizada nos testes do protótipo está apresentada na Figura 27, sendo que para validações de recursos homogêneos e heterogêneos, modifica-se apenas a capacidade de processamento das MPs. Conforme a figura, os testes contam com uma máquina que age no papel de chaveador (Xeon 2.4 GHz) que despacha transações para um *cluster* de MP homogêneas ou heterogêneas. O caso homogêneo é formado por seis máquinas Xeon 2.4 GHz com 2 GB de RAM, enquanto que o caso heterogêneo será analisado considerando duas máquinas Xeon 2.4 GHz com 2 GB de RAM, duas máquinas Xeon 2.0 GHz com 2 GB de RAM e outras duas máquinas Xeon 1.6 GHz com 2 GB de RAM. Tanto para a estrutura homogênea quanto para a heterogênea, o conjunto de transações foi avaliado com o protótipo GetLB e com o mecanismo de escalonamento padrão RR.

Figura 27: Topologia utilizada nos testes do protótipo.

	Topologia	
	Homogênea	Heterogênea
MP0	2.4GHz	1.6GHz
MP1	2.4GHz	1.6GHz
MP2	2.4GHz	2.0GHz
MP3	2.4GHz	2.0GHz
MP4	2.4GHz	2.4GHz
MP5	2.4GHz	2.4GHz



Fonte: Desenvolvido pelo próprio autor.

Dado que as máquinas do laboratório do PIPCA que foram utilizadas nas avaliações são

Figura 28: Algoritmo de envio de transações executado no chaveador.

```
01. t1 = captura_tempo();
02. Enquanto existirem transacoes
03. {
04. transacao = captura_transacao(arquivo);
05. t2 = captura_tempo();
06. se algoritmo for Round-Robin
07. {
08. maquina = captura_maquina_rr ();
09. }
10. senao
11. {
12. maquina = captura_maquina_ll (transacao);
13. }
14. t3 = captura_tempo();
15. despacha(transacao, maquina);
16. }
17. vetor[] v = aguarda_conclusao_transacoes(t1);
18. makespan = captura_maior_tempo(v);
```

Fonte: Desenvolvido pelo próprio autor.

homogêneas, as situações de heterogeneidade foram obtidas com auxílio da ferramenta **cpulimit**[1], que representa uma técnica muito utilizada na literatura (GRAEBIN; ROSA RIGHI, 2011; CESARIO et al., 2011). Ela possibilita que se limite o percentual máximo de CPU disponível.

O algoritmo de despacho das transações está ilustrado na Figura 28. Essa sequência de passos é executada pelo chaveador, que retira as transações do arquivo de rastro, utilizado como entrada de tarefas. A linha 17 do algoritmo aguarda pela conclusão de todas as transações em cada máquina. Cada máquina gera um tempo final de operações que é a diferença entre o seu tempo de término e o tempo $t1$ capturado no início do programa. O tempo de conclusão da carga de trabalho (*workload*) é obtido na linha 18. Na literatura de escalonamento, tal medida também é conhecida como *makespan* (MAURER; BRANDIC; SAKELLARIOU, 2013). O método "despacha" (linha 15) possui um caráter assíncrono, sendo responsável pelo envio de uma mensagem pela rede até a fila de entrada da MP alvo. Por outro lado, o método apresentado na linha 17 é síncrono, ou seja, tem comportamento bloqueante até a conclusão de todas as transações. A diferença entre os tempos $t3$ e $t2$ informa o tempo gasto no algoritmo de escalonamento, RR ou LL. O algoritmo RR simplesmente captura a próxima máquina em uma fila circular, enquanto que LL usa o tipo da transação como parâmetro de entrada para escolher a máquina mais apropriada para processá-la.

Com uso destes procedimentos, podem ser aplicados testes com dois algoritmos de escalonamento: GetLB e RR. Assim, se pode verificar através de comparação, o desempenho de GetLB frente ao RR em diferentes situações.

Ao longo do capítulo, entre os testes que serão efetuados, será abordado o comportamento dos algoritmos quando ocorrem situações não esperadas durante o processamento, ou seja, situações de dinamicidade de eventos no sistema.

Também serão tratadas situações que simulam indisponibilidade de uma máquina durante a

[1] http://cpulimit.sourceforge.net/

execução do processamento, para as arquiteturas homogênea e heterogênea. Serão feitos testes com intervalo de indisponibilidade, com objetivo de acompanhar o nível de ocupação das filas e se ocorrem perdas transacionais com algum dos algoritmos.

Os eventos que se referem às notificações serão testados simulando uma situação de manutenção sistêmica e também inclusão de nova máquina no sistema em tempo de execução, sem gerar indisponibilidade.

Nas próximas seções, serão efetuados os testes de acordo com os métodos descritos. Os resultados serão analisados comparados através de tabelas e gráficos para auxiliar a interpretação.

6.2 Avaliação em topologia homogênea

Conforme descrito na seção anterior, para avaliar o desempenho dos dois algoritmos em uma topologia homogênea, foram utilizadas seis máquinas com a mesma configuração, recebendo transações a partir de uma máquina que faz o papel do chaveador, executando os algoritmos RR e GetLB. As máquinas foram demandadas para processar 8168 transações, conforme o arquivo de rastro transacional.

Os testes que mostram o tempo de funcionamento das MP e a distribuição das transações no *cluster* homogêneo estão apresentados na Tabela 9. O tempo apresentado nessa tabela está na notação horas:minutos:segundos e denota a diferença de tempo desde o início do programa até o processamento da última transação. Percebe-se um equilíbrio na distribuição das transações com RR, porém isso não significa melhor balanceamento sob a perspectiva de tempo de processamento. Nesta topologia as máquinas são homogêneas, porém as transações não são, e RR as mapeia ciclicamente para os recursos sem observar as suas características. Tal situação foi ilustrada previamente na Figura 1. Já GetLB foi responsável pela obtenção de um tempo total de processamento da carga de trabalho, também chamado de *makespan*, menor que obtido no RR. Ou seja, GetLB demandou 00:04:47 e RR um tempo total de 00:05:12.

A Tabela 10 apresenta a relação de tempo de processamento gasto em cada máquina para a computação dos cinco principais tipos de transação que são atendidas pelo sistema. O balanceamento de carga atua de modo a equilibrar a soma de processamento efetuado em cada máquina. Ao analisar os seis valores na última linha da tabela se obtém uma média de 228,78 segundos e um desvio padrão de 20,12. O mapeamento proposto por GetLB é eficaz ao ponto de amortizar o gasto para a computação do algoritmo para cada transação.

Para cada transação, são sempre computadas m equações $LL(i, j)$, onde m é o número de máquinas MP candidatas, i representa a transação a ser despachada e j uma máquina MP em particular. O menor índice LL informa a máquina que receberá transação. Para cada transação, GetLB utiliza em média 6,96 milissegundos para o cálculo das equações, efetuado com dados locais que são atualizados periodicamente pelas máquinas MP, e escolha do alvo para despacho. Por outro lado, o algoritmo RR é computado em 1,12 milissegundos por transação. O fato da computação do algoritmo RR ser 6 vezes mais rápida que a do LL é explicada pela sua

Tabela 9: Tempo de processamento e distribuição de transações em uma topologia com máquinas processadoras homogêneas.

	GetLB			Round-Robin		
Máquina	Número de transações	Percentual recebido	Tempo de processamento	Número de transações	Percentual recebido	Tempo de processamento
MP0	1.470	18%	00:04:45	1.362	16,66%	00:05:09
MP1	1.192	15%	**00:04:47**	1.362	16,66%	**00:05:12**
MP2	1.467	18%	00:04:07	1.361	16,66%	00:05:12
MP3	1.365	17%	00:04:46	1.361	16,66%	00:05:07
MP4	1.377	17%	00:04:38	1.361	16,66%	00:04:54
MP5	1.297	16%	00:04:06	1.361	16,66%	00:05:10
Total	8168	100%		8168	100%	

Fonte: Desenvolvido pelo próprio autor.

Tabela 10: Tempo total de processamento em segundos por tipo de transação em cada máquina, com balanceamento GetLB.

Tipo de transação	MP0	MP1	MP2	MP3	MP4	MP5
Administrativas	14,77	10,68	15,86	11,70	12,20	12,64
Recarga de telefonia	117,30	96,51	121,995	115,75	108,78	103,59
Compra com cartão	113,12	91,62	112,99	96,23	97,94	96,30
Estorno de compra	0,87	0,68	2,90	2,45	1,31	2,48
Reversão de transação	2,07	2,47	2,22	1,90	1,22	2,05
Soma dos tempos	248,16	201,98	255,98	228,05	221,46	217,08

Fonte: Desenvolvido pelo próprio autor.

simplicidade, que ao invés da avaliação de equações, faz somente um deslocamento de uma posição numa fila circular de máquinas alvo. Considerando uma carga de trabalho de 8.168 transações, despende-se 9,14 segundos para escalonar as transações no algoritmo RR e 56,84 segundos para essa tarefa com o algoritmo LL.

6.3 Avaliação em topologia heterogênea

Os resultados com uma topologia de *cluster* heterogêneo são apresentados na Tabela 11. O algoritmo RR penaliza o tempo final para execução das transações porque parte delas são igualmente enviadas para máquinas com menor capacidade. Sendo assim, além do conjunto de transações ser caracterizado como um sistema heterogêneo, o fato dos recursos também serem heterogêneos degrada ainda mais o desempenho de RR. O algoritmo GetLB por sua vez, despacha mais de 90% das transações para as quatro máquinas com maior capacidade. Essa estratégia faz com que GetLB execute a carga de transações em 551 segundos, enquanto RR faz essa tarefa em 621 segundos. Apesar de mostrar um ganho de 11,27% em favor de GetLB,

percebe-se que as máquinas MP0 e MP1 são subutilizadas. Em outras palavras, GetLB escolhe uma máquina mais poderosa com transações em fila a outra ociosa com menor capacidade de processamento. Uma análise mais detalhada será feita para avaliar os tempos de cada um dos elementos na fórmula de LL usada em GetLB.

A exemplo dos resultados verificados na topologia homogênea, o *makespan* da topologia heterogênea também apresentou melhores resultados ao aplicar o algoritmo de GetLB. Conforme se verifica na Tabela 11, o tempo total de processamento do RR foi de 00:11:08, enquanto em GetLB foi concluído em 00:09:59.

Tabela 11: Tempo de processamento e distribuição de transações em máquinas heterogêneas.

	GetLB			Round-Robin		
Máquina	Número de transações	Percentual recebido	Tempo de processamento	Número de transações	Percentual recebido	Tempo de processamento
MP0	653	7,99%	00:09:10	1.362	16,67%	00:10:20
MP1	5	0,06%	00:09:02	1.362	16,67%	00:10:22
MP2	1.787	21,88%	**00:09:59**	1.361	16,66%	**00:11:08**
MP3	1.865	22,83%	00:09:08	1.361	16,66%	00:10:18
MP4	1.940	23,75%	00:09:01	1.361	16,66%	00:10:10
MP5	1.918	23,48%	00;09:11	1.361	16,66%	00:10:21
Total	8168	100%		8168	100%	

Fonte: Desenvolvido pelo próprio autor.

6.4 Avaliação de dinamicidade

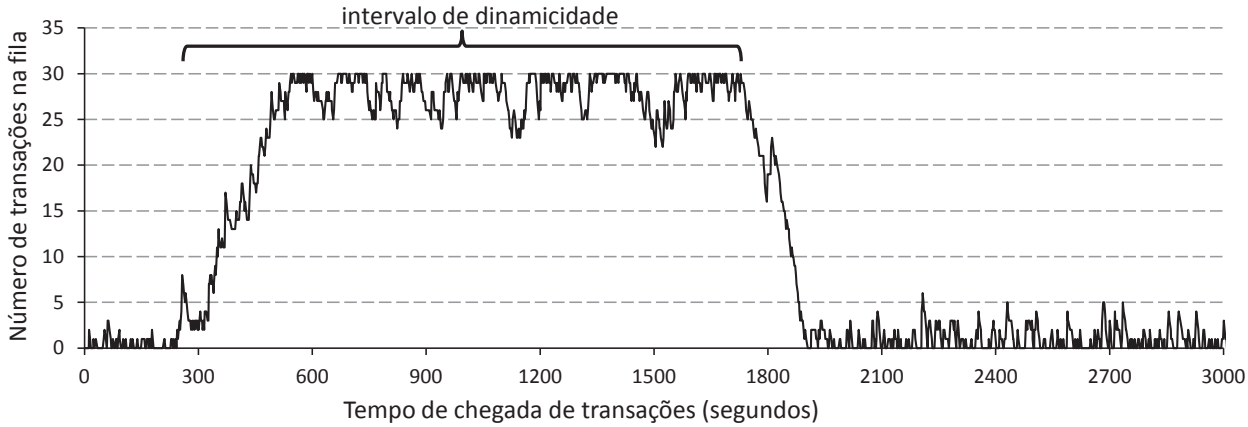
Um sistema transacional pode estar sujeito a eventos não esperados durante seu processamento, os quais podem degradar o desempenho de algum dos seus componentes. Deste modo, avaliar os algoritmos de GetLB e RR em situações de dinamicidade de eventos no sistema é pertinente na medida em que podem gerar indisponibilidade. Conforme citado na Seção 2.1, no referencial teórico, se alguma transação for recebida em uma MP quando a capacidade da fila de transações houver atingido seu limite máximo, a mesma será descartada. Desse modo, a degradação de desempenho de uma MP pode resultar no enfileiramento de transações e, em consequência, podem ocorrer perdas de transações.

Para isso, com auxílio do aplicativo **cpulimit**, foi simulada a ocorrência de um evento em uma das MP, consumindo 80% da sua capacidade de processamento em um intervalo de tempo de 20 minutos. Esse período foi aplicado com base no período de pico de transações do dia, conforme o mesmo rastro transacional em uso nos demais testes. Em uma topologia homogênea, foram analisadas as filas de entrada das MP ao longo do tempo a fim de identificar possíveis perdas de transações. A fila de transações de cada máquina processadora foi configurada para um limite de 30 transações. Esta medida foi adotada para simplificar a análise sem comprome-

ter a fidelidade com a topologia transacional da empresa, que possui uma fila com capacidade mais elevada. O teste foi aplicado aos algoritmos GetLB e RR.

Degradando o desempenho de MP1 no algoritmo RR, se verifica o resultado do teste na Figura 29. Se observa neste gráfico que, no momento em que inicia o processo que consome a CPU de MP1, inicia também o enfileiramento de transações em sua fila de entrada. Dado que RR envia transações de maneira cíclica para as máquinas processadoras, toda transação recebida em MP1 quando a fila estiver com limite máximo atingido será perdida. Os testes nessa condição resultaram na perda de 136 transações, que corresponde a 1,66% do total de transações recebidas no intervalo.

Figura 29: Gráfico de fila de transações na entrada de MP1, com escalonamento Round-Robin e submetido à dinamicidade de consumo de 80% do processamento.



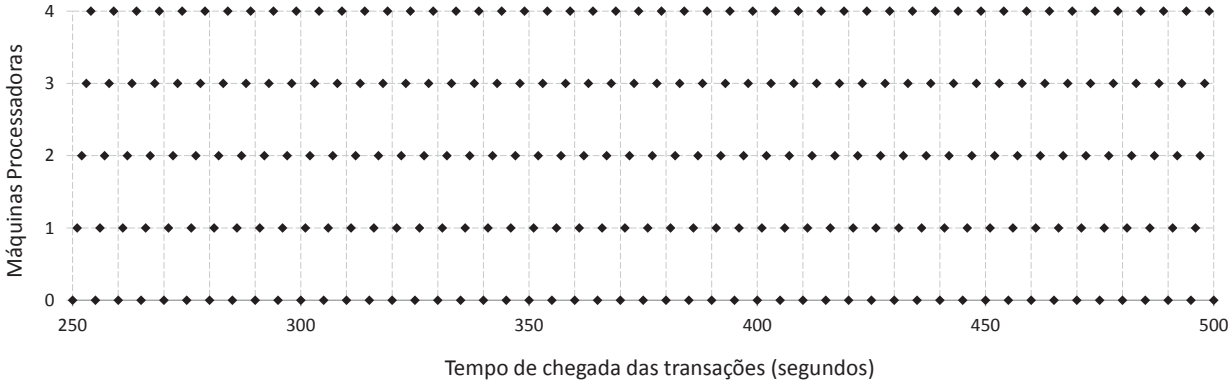
Fonte: Desenvolvido pelo próprio autor.

A Figura 30 apresenta a distribuição das transações entre as MPs no algoritmo Round-Robin, durante a situação de degradação de processamento de MP1. Considerando que a distribuição do RR se repete ao longo de todo processo, o gráfico está com a escala aumentada e com linhas auxiliares no eixo vertical para favorecer a identificação das transações recebidas e a máquina de destino. No gráfico é possível constatar que não há alteração na distribuição de transações entre as máquinas no entorno do segundo 300. Esse comportamento explica o aumento da fila de transações na MP1.

Na Figura 31 é apresentado o resultado do teste com o algoritmo GetLB para a fila de entrada de MP1 quando submetida à mesma situação de degradação de processamento que foi aplicada quando do emprego do algoritmo RR. Se verifica que, no momento em que inicia o evento de dinamicidade, devido à degradação do desempenho da CPU, o ACE interrompe o envio de transações para MP1, distribuindo as tarefas entre as outras máquinas processadoras. Este procedimento do algoritmo GetLB impede que se atinja o limite da fila de MP1 e assim evita perda de transações.

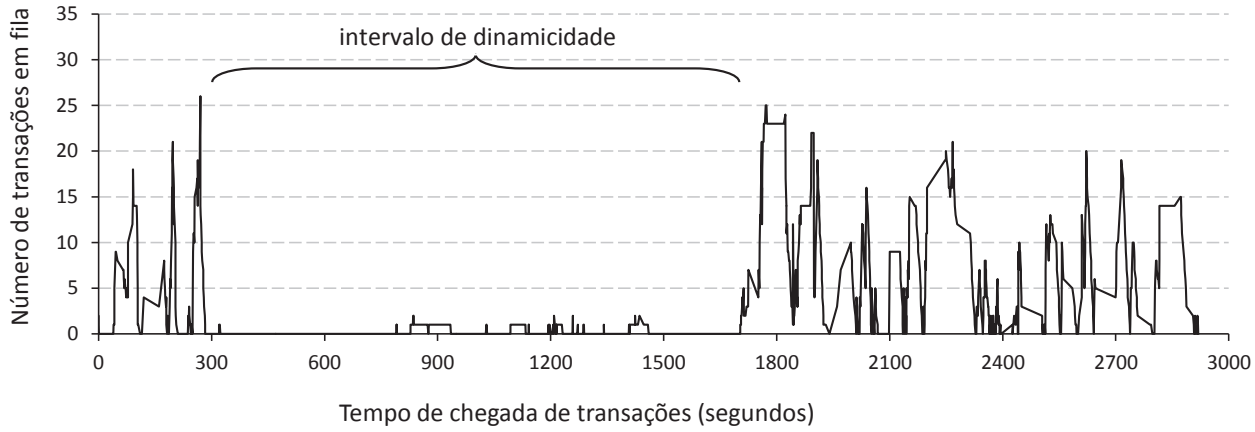
A Figura 32 apresenta a distribuição das transações entre as MPs segundo o algoritmo de GetLB, durante a situação de degradação de processamento de MP1. Esse gráfico destaca como

Figura 30: Gráfico de dispersão das transações ao longo do tempo entre as máquinas processadoras, onde se verifica que RR despacha normalmente transações para MP1 a partir dos 300 segundos.



Fonte: Desenvolvido pelo próprio autor.

Figura 31: Gráficos de fila de transações na entrada de MP1 para o balanceamento GetLB, quando a máquina é submetida à dinamicidade de consumo de 80% do processamento entre o intervalo 300 a 1750 segundos.



Fonte: Desenvolvido pelo próprio autor.

GetLB contorna a situação de dinamicidade e distribui transações entre as outras máquinas. Do total de transações processadas no intervalo de dinamicidade, apenas 3,31% das transações recebidas no ACE foram despachadas para MP1. Essa redução de carga em MP1, complementa o gráfico da Figura 31. Analisando o entorno do segundo 300, quando a fila de transações de MP1 estava elevada, a redução de carga para esta máquina resulta na redução também da fila, uma vez que estava com capacidade disponível para consumir estas tarefas, sem receber mais carga. No entorno do segundo 1800 ocorre o inverso, ou seja, na medida em que a capacidade de processamento de MP1 é restabelecida e a informação transmitida ao ACE, inicia novamente o despacho para essa máquina. Nessa simulação não ocorreram perdas de transações.

Figura 32: Gráfico de dispersão das transações ao longo do tempo entre as MPs conforme o algoritmo GetLB. Se verifica a redução de envio de transações para MP1 no intervalo em que há degradação de capacidade de processamento na mesma.

Fonte: Desenvolvido pelo próprio autor.

6.5 Avaliação de indisponibilidade

Situações de indisponibilidade podem ser consideradas as mais críticas em um sistema transacional. Elas podem levar a perdas diretas de transações e consequentemente afetar a imagem e o resultado financeiro da empresa. Desse modo, nas decisões de projeto estabeleceu-se a premissa de que o sistema de balanceamento de transações possa contornar eventos desta natureza.

Nesta seção de testes, será analisado o comportamento dos algoritmos GetLB e RR quando submetidos a uma situação de indisponibilidade em uma das máquinas processadoras na topologia homogênea. A topologia heterogênea será analisada apenas com escalonamento de GetLB. Os algoritmos serão submetidos a uma carga de 8168 transações e MP2 ficará indisponível entre as transações 500 e 1500.

Os resultados dos testes com escalonamento RR e GetLB na topologia homogênea podem ser comparados na Tabela 12. Nestes dados se verifica a distribuição equilibrada em GetLB, com percentual menor de transações em MP2. Na distribuição RR, durante a indisponibilidade de MP2, todas as transações destinadas a esta máquina foram perdidas, num total de 167 transações, ou 11,13% do intervalo. Quanto ao *makespan*, se constata que GetLB demanda mais tempo de processamento para o conjunto de tarefas, porém é esperado que RR termine antes, dado que não processa todas as transações. GetLB processa todas as transações evitando perdas, portanto é mais lento. Contudo, para um sistema transacional, ou usuário de cartão, é o mais sensato.

Na Figura 33 está representada a distribuição RR de transações entre as MPs na topologia homogênea. O gráfico está com escala aumentada para melhor visualização da distribuição no intervalo. Se pode identificar o período de indisponibilidade entre as transações 500 e 1500.

Tabela 12: Tempo de processamento e distribuição de transações em máquinas homogêneas com indisponibilidade em MP2.

	GetLB			Round-Robin		
Máquina	Número de transações	Percentual recebido	Tempo de processamento	Número de transações	Percentual recebido	Tempo de processamento
MP0	1.496	18,32%	00:06:22	1.362	16,67%	00:05:33
MP1	1.353	16,56%	00:06:28	1.361	16,66%	00:05:40
MP2	1.148	14,05%	**00:08:56**	1.195	14,63%	**00:08:08**
MP3	1.321	16,17%	00:06:15	1.361	16,66%	00:05:27
MP4	1.321	16,17%	00:05:50	1.361	16,66%	00:05:02
MP5	1.529	18,72%	00:06:22	1.361	16,66%	00:05:34
Total	8168	100%		8001	97,64%	

Fonte: Desenvolvido pelo próprio autor.

Nesse intervalo, as transações despachadas para MP2 foram perdidas.

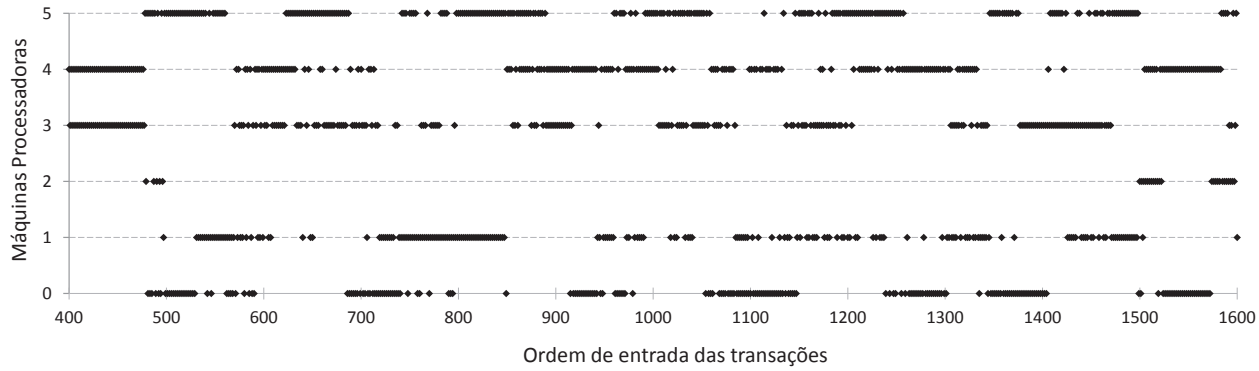
Figura 33: Gráfico de RR apresentando a distribuição das transações entre as MPs homogêneas ao longo do recebimento de transações. Se verifica a indisponibilidade de MP2 no intervalo entre as transações 500 e 1500.

Fonte: Desenvolvido pelo próprio autor.

Na Figura 34 se pode observar a distribuição de transações entre as MPs com escalonamento de GetLB em uma topologia homogênea. Este gráfico também está com escala aumentada no intervalo próximo da indisponibilidade para melhor visualização da distribuição. Durante a indisponibilidade, nenhuma transação foi despachada para MP2, evitando as perdas. Esta situação está diretamente relacionada com a frequência na qual são atualizadas as informações das MPs no ACE. Nestes testes, o tempo utilizado foi 1 segundo, contudo, um tempo mais elevado poderia resultar informações desatualizadas no ACE, levando ao despacho de transações para uma máquina indisponível.

Também se analisou a situação de indisponibilidade em uma topologia heterogênea, sendo que, para estes testes, foi utilizado o algoritmo de balanceamento GetLB. Assim como o teste

Figura 34: Gráfico de GetLB apresentando a distribuição de transações entre as MPs homogêneas durante o recebimento de transações. Se verifica a redução de carga em MP2 no intervalo entre as transações 500 e 1500.



Fonte: Desenvolvido pelo próprio autor.

aplicado na topologia heterogênea, o intervalo de indisponibilidade que MP0 será submetida é entre as transações 500 e 1500. O propósito desse teste é verificar como GetLB se comporta nesta situação, caso uma das máquinas de maior capacidade de processamento seja afetada. Nesse sentido, a topologia foi formada por 5 máquinas, sendo MP0 e MP1 com 2.4GHz de CPU, MP2 e MP3 com 2.0GHz e MP4 com 1.6GHz. Em um regime normal de processamento para esse tipo de topologia, a distribuição das transações tende a concentrar o seu maior volume de tarefas nas máquinas com maior poder de processamento. Contudo, quando ocorre indisponibilidade em MP0, se verifica através dos resultados apresentados na Tabela 13 que a distribuição de transações se concentrou mais nas máquinas de capacidade intermediária. A máquina com menor capacidade permaneceu pouco acessada, porém com volume acima do esperado para um recurso com 40% menos poder de processamento em relação às de maior capacidade.

Tabela 13: Tempo de processamento e distribuição de transações em uma arquitetura heterogênea e balanceamento GetLB. Situação de indisponibilidade em MP0.

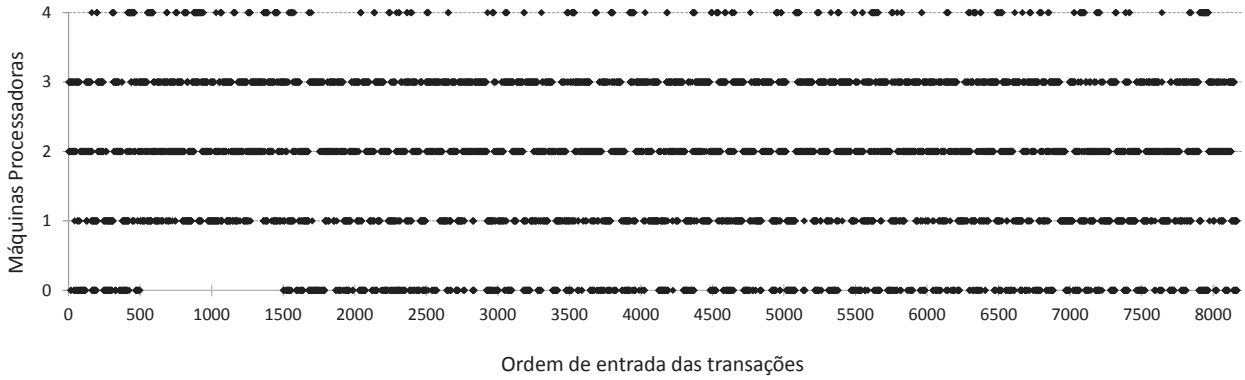
	GetLB		
Máquina	Número de transações	Percentual recebido	Tempo de processamento
MP0	1.238	15%	00:07:39
MP1	1.654	20%	00:07:28
MP2	2.433	30%	**00:09:41**
MP3	2.345	29%	00:07:29
MP4	410	5%	00:07:38
Total	8168	100%	

Fonte: Desenvolvido pelo próprio autor.

Na Figura 35 estão os resultados de distribuição na topologia heterogênea e balanceamento

de GetLB. Os resultados deste gráfico destacam a fraca concentração de transações em MP0 e maior uso de MP1, MP2 e MP3, sobretudo no intervalo de indisponibilidade. A exemplo do que ocorreu com a topologia homogênea, durante a indisponibilidade, GetLB não despachou nenhuma transação para a máquina afetada. Contudo, deve ser destacado que a frequência na qual as informações das MPs são atualizadas é de 1 segundo, e tempos mais elevados podem retardar a informação de indisponibilidade, o que levaria o ACE a encaminhar uma transação indevidamente, para então processar a informação corretamente.

Figura 35: Gráfico de GetLB apresentando a distribuição de transações entre MPs heterogêneas durante o recebimento de transações. Se verifica a redução de carga em MP0 no intervalo entre as transações 500 e 1500.



Fonte: Desenvolvido pelo próprio autor.

6.6 Avaliação de notificações

Conforme apresentado na Seção 4.4, as notificações representam um importante recurso no sistema transacional. Através delas é possível sinalizar ao chaveador, de maneira assíncrona, sobre a ocorrência de eventos que podem resultar em indisponibilidade ou mesmo para realizar manutenções sistêmicas em tempo de execução.

Os testes realizados nessa seção apresentam as possibilidade de manutenção sistêmica disponíveis através deste recurso fornecido por GetLB. O primeiro teste contempla uma arquitetura heterogênea sendo incrementada com mais uma MP a fim de melhorar o tempo médio de processamento das transações. No segundo teste, uma MP com baixo desempenho de processamento será removida e, após manutenção, será reinserida no sistema.

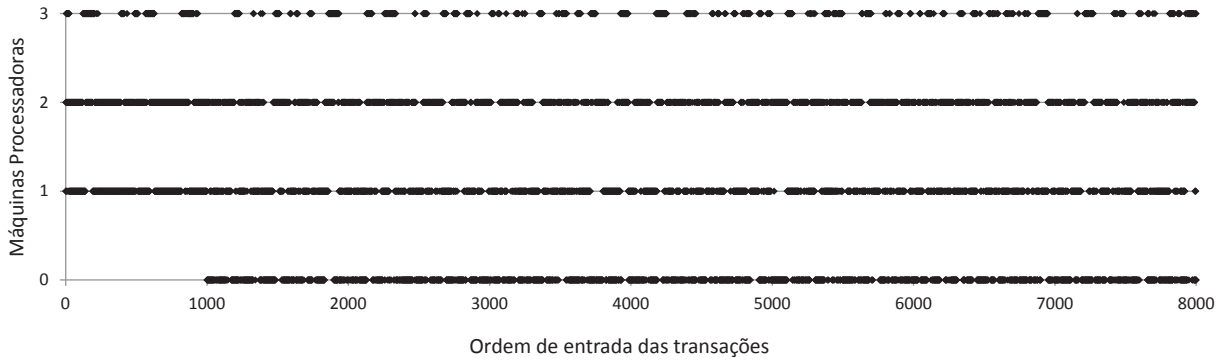
No primeiro teste de notificação, será inserida uma máquina ao sistema em execução. O propósito é simular uma situação em que se deseja melhorar o desempenho do sistema, no que se refere ao tempo médio de processamento das transações. Para isso, se criou um grupo com três máquinas heterogêneas, sendo as CPUs com capacidade de 2.4GHz, 2.0GHz e 1.66GHz

para MP1, MP2 e MP3 respectivamente. O sistema permanece processando as transações neste regime desde a transação 0 até 1000. A partir da transação 1000 se insere mais uma máquina processadora ao sistema, MP0, com poder de processamento de 2.4GHz. Ao final da simulação é feita uma nova avaliação do desempenho do sistema após a inclusão da máquina e comparadas as situações de antes e depois da inclusão.

Para obter esse resultado, GetLB utiliza um arquivo de texto auxiliar com nome **ProcessingMachines**, conforme descrito na Seção 5.1. Neste arquivo deve ser inserida a máquina com suas respectivas características. Essa ação permite ao ACE reconhecer a máquina que estará sendo integrada ao sistema no momento em que receber uma notificação do tipo 5, indicando que há uma nova máquina entrando em modo operacional.

Na Figura 36, se verifica o resultado dessa simulação. As máquinas MP1, MP2 e MP3 formam o conjunto de máquinas processadoras que estão operando no intervalo 0 a 1000, no qual se observa a distribuição de carga proporcional à capacidade de processamento de cada máquina. A partir da transação 1000, em tempo de execução mais uma máquina passa a integrar o sistema.

Figura 36: Gráfico de distribuição de transações entre MPs heterogêneas durante o recebimento de transações. A partir da transação 1000, MP0 passa a integrar o sistema, sem afetar o processamento das demais MPs.



Fonte: Desenvolvido pelo próprio autor.

A Tabela 14 apresenta os tempos médios de processamento por máquina, para cada tipo de transação, no intervalo de transações 0 a 1000. A Tabela 15 apresenta as novas médias de tempos após a inclusão de MP0. Como esperado, a inclusão de uma nova máquina reduz o tempo médio de processamento para todos os tipos de transações, uma vez que o chaveador conta com mais recursos para alocação de tarefas.

O gráfico da Figura 37 apresenta a melhora do tempo médio de processamento para cada tipo de transação nas duas topologias, com 3 e 4 máquinas processadoras. Esse resultado está de acordo com o esperado, contudo, se demonstra também que GetLB possibilita essa melhoria em tempo de execução, sem gerar indisponibilidade.

No segundo teste de notificação, será retirada e reinserida uma máquina ao sistema em tempo de execução. O propósito é simular uma situação de manutenção sistêmica, na qual se

Tabela 14: Média de tempo de processamento em milissegundos por tipo de transação em cada máquina processadora, com balanceamento GetLB. Resultado referente às primeiras 1000 transações recebidas no sistema.

Tipo de transação	MP1	MP2	MP3	Média
Administrativas	146,74	152,20	164,82	154,59
Recarga de telefonia	185,64	203,80	217,47	202,30
Compra com cartão	457,93	494,01	564,22	505,39
Estorno de compra	478,25	282,83	453,54	404,87
Reversão de transação	371,53	-	526,00	448,76

Fonte: Desenvolvido pelo próprio autor.

Tabela 15: Média de tempo de processamento em milissegundos por tipo de transação em cada máquina processadora, com balanceamento GetLB. Resultado para a transação 1000 em diante.

Tipo de transação	MP0	MP1	MP2	MP3	Média
Administrativas	114,79	120,43	110,85	174,45	130,13
Recarga de telefonia	159,85	179,35	152,88	242,28	183,59
Compra com cartão	399,03	424,64	385,66	617,91	456,81
Estorno de compra	321,22	283,96	295,73	484,94	346,46
Reversão de transação	393,91	422,69	322,05	592,56	432,81

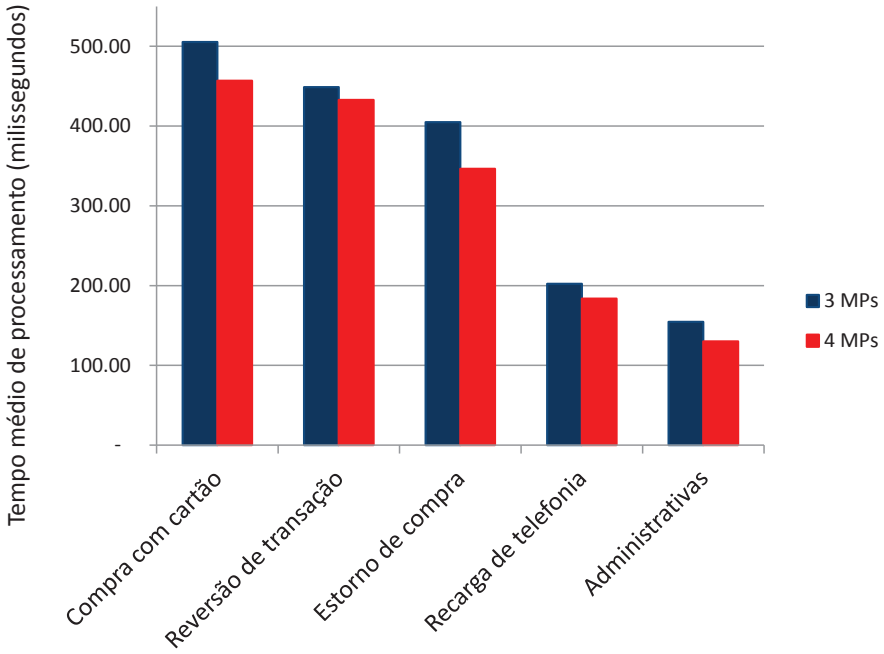
Fonte: Desenvolvido pelo próprio autor.

verifica que uma máquina está com desempenho baixo e uma ação do administrador permite, em tempo de execução, realizar a tarefa. Para isso, se criou um sistema com quatro MPs homogêneas, sendo as CPUs com capacidade de 2.4GHz. A máquina MP3 teve seu desempenho degradado em 40% a partir do início da simulação, com auxílio da ferramenta **cpulimit**. O sistema permanece processando as transações neste regime desde a transação 0 até 1000. A partir da transação 1000 a máquina é retirada e retorna ao sistema sem que ocorram interrupções na execução de GetLB, a partir da transação 2000.

Neste teste, diferentemente do teste anterior, não foi necessário preparar o arquivo **ProcessingMachines**, pois MP3 já estava em uso no grupo de máquinas procesadoras disponíveis para o ACE. Para que a máquina seja retirada do grupo, uma notificação do tipo 1, que corresponde à manutenção, é transmitida ao ACE. Ao receber essa sinalização, as transações deixam de ser despachadas para MP3. Após a manutenção, o administrador insere novamente a máquina no grupo através de uma notificação do tipo 5.

Essa simulação pode ser acompanhada através da Figura 38. A partir do início da simulação se identifica o baixo volume de transações sendo endereçadas para MP3, apesar de se tratar de uma topologia homogênea. A máquina é retirada no intervalo de transações 1000 a 2000, conforme se verifica na distribuição do gráfico e, após a transação 2000, regressa com a capacidade restabelecida. Assim que MP3 retorna ao grupo, após a manutenção, se constata que o ACE

Figura 37: Gráfico comparativo para tempo médio de processamento das transações. Com topologia homogênea de 3 máquinas atendedoras e após a intervenção do administrador de rede, a partir da transação número 1000, com 4 máquinas atendedoras formando a topologia.



Fonte: Desenvolvido pelo próprio autor.

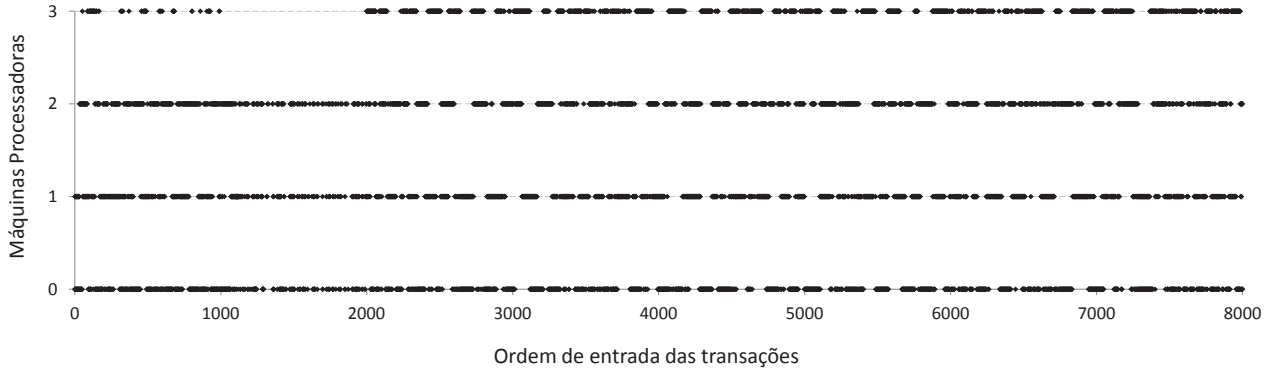
passa a despachar as transações novamente para essa máquina, e o sistema entra em regime de distribuição de tarefas como esperado para uma arquitetura homogênea.

6.7 Análise do Capítulo

Ao concluir o capítulo, após uma série de testes simulando diferentes arquiteturas em diferentes situações, sobretudo com enfoque nas decisões de projeto, foi possível obter resultados efetivos e conclusivos a respeito do algoritmo de balanceamento GetLB. Percorrendo topologias homogêneas e heterogêneas, com o algoritmo GetLB sendo comparado ao tradicional Round-Robin, foi possível relacionar os desempenhos de ambos em situações tais como as que ocorrem em um sistema transacional real.

Inicialmente se analisou a arquitetura homogênea, examinando os resultados dos tempos de processamento e distribuição de transações entre as máquinas processadoras. Os resultados mostraram que, na distribuição com RR, apesar de haver equilíbrio na distribuição de tarefas entre as máquinas, isso não significa melhor balanceamento considerando tempo de processamento. Isso se deve ao fato de que, apesar das máquinas serem homogêneas, as transações são heterogêneas. Para o mesmo cenário, GetLB obteve um *makespan* menor que o obtido com o algoritmo RR. Isso se deve ao fato de que GetLB considera não apenas o estado das MPs para

Figura 38: Gráfico de distribuição de transações entre as máquinas processadoras. No intervalo entre as transações 1000 e 2000, MP3 é retirada para manutenção. Após a transação 2000, ela é reintegrada ao conjunto com melhor capacidade de processamento.



Fonte: Desenvolvido pelo próprio autor.

balanceamento, mas também o tipo de transação, para então despachá-la para a máquina mais apropriada. Verificou-se também que o mapeamento proposto por GetLB é eficaz ao ponto de amortizar o gasto para a computação do algoritmo para cada transação, uma vez que o cálculo de RR é cerca de 6 vezes mais rápido que LL.

Ao analisar a topologia heterogênea, se verificou que RR penaliza o tempo final para execução das transações porque parte delas são igualmente enviadas para máquinas com menor capacidade. GetLB foi mais eficiente pois adotou uma estratégia que prioriza o envio de transações para máquinas mais capacitadas. Novamente, se verificou, a exemplo dos resultados da topologia homogênea, que o *makespan* de GetLB apresentou melhores resultados comparado ao RR. Destacou-se nestes testes, que as máquinas de baixa capacidade são subutilizadas, devido ao fato de que GetLB escolhe uma máquina mais poderosa com transações em fila em relação a outra ociosa com menor capacidade de processamento. Acredita-se que um fator de penalização, inserido nas equações de LL, pode equilibrar melhor esta distribuição.

As situações de dinamicidade também foram abordadas no capítulo. Trata-se de eventos temporários que ocorrem em tempo de processamento, afetando o desempenho de algum componente do sistema e, em consequência, colocando em risco a qualidade do serviço oferecido. Ao reduzir, durante um intervalo de tempo, em 80% o poder de processamento de uma das máquinas que compõem o conjunto de MPs, analisou-se o comportamento de GetLB e RR. Os resultados mostraram que, com o escalonamento RR, durante o período de testes, as transações seguiram sendo despachadas para a MP afetada pela degradação de capacidade, até que o limite da fila de entrada de transações fosse atingido. Como resultado, ocorreram perdas de transações, na medida em que as transações recebidas em tal situação são descartadas em um sistema transacional. Na mesma situação, GetLB contornou a situação encaminhando tarefas

para recursos com maior capacidade e, com isso, evitou que houvesse perdas de transações.

A seção seguinte deste capítulo tratou dos testes de indisponibilidade, ou seja, como os dois algoritmos se comportam ao se deparar com indisponibilidade de uma das máquinas do conjunto de processadoras. Para isso, em uma arquitetura homogênea, uma das MPs foi retirada em tempo de execução e observou-se o resultado de GetLB e RR. Ao executar o teste com RR, os resultados mostraram que no intervalo em que uma das MPs ficou indisponível, as transações seguiram sendo despachadas para essa máquina. A consequência desse procedimento foi a perda de 11,3% das transações que estavam sendo recebidas no sistema durante esse período. Os testes de GetLB mostraram uma distribuição equilibrada, com percentual menor de transações sendo despachadas para a máquina indisponível durante o intervalo em que foi aplicado. Contudo, comparando o *makespan* das duas simulações, se verificou que GetLB demandou mais tempo de processamento que RR nessa situação, porém isso é aceitável na medida em que este algoritmo processou mais transações que RR e evitou perdas. Além dessas observações, cabe destacar a importância do tempo de atualização das MPs ao ACE. As perdas em GetLB puderam ser evitadas neste teste pois o tempo de atualizações do ACE foi parametrizado em 1 segundo. Tempos maiores podem levar a perdas, na medida em que o chaveador não dispõe das informações mais atuais do estado das máquinas.

Fechando o capítulo, exploraram-se algumas possibilidades oferecidas pelas notificações. Para isso, duas situações foram simuladas para demonstrar que o administrador do sistema pode realizar melhorias ou efetuar manutenção durante o tempo de execução, sem gerar indisponibilidade. No primeiro teste, buscou-se a melhora no tempo médio de processamento das transações em um grupo de máquinas heterogêneas. Para isso, após um dado período com três MPs executando o processamento, uma quarta máquina foi inserida ao sistema, em tempo de execução. Foi demonstrado, ao final do teste, que ocorreu melhora no tempo médio de processamento das transações, sem ocorrência de indisponibilidade sistêmica. No segundo teste foi demonstrada a possibilidade de manutenção sistêmica através das notificações. Para isso, em um grupo de máquinas homogêneas, degradou-se a capacidade de processamento de uma das MPs durante o teste. O algoritmo GetLB passou a despachar menos transações para essa máquina. Durante o processamento, a MP com problema foi retirada e reinserida ao grupo posteriormente, agora com capacidade de CPU restabelecida. Com isso, ao ser reintegrada ao grupo de máquinas, a distribuição de tarefas entre os recursos passou a ocorrer de modo equilibrado.

Em resumo, os resultados dos testes mostraram que GetLB é mais eficiente que Round-Robin em muitos aspectos, sobressaindo-se mais em situações de heterogeneidade e dinamicidade eventos, além de disponibilizar recurso de notificações, que agregam muito valor ao sistema, auxiliando na manutenção e melhoria contínua da solução. Contudo, algumas questões precisam ser aprofundadas, tais como o período de atualizações do ACE e a distribuição de tarefas em arquiteturas heterogêneas, em especial quando máquinas com pouca capacidade integram o sistema. As atualizações do ACE podem resultar em *overhead* de comunicação na rede, na medida em que se tornem muito frequentes, porém, é preciso encontrar um ponto de

equilíbrio entre uma pronta resposta do chaveador para situações dinâmicas do sistema e o elevado tráfego de dados na rede. Quanto às questões relativas à distribuição de tarefas em recursos heterogêneos, deve ser mais estudada, sobretudo com foco no cálculo de LL, encontrando um modo eficiente para equilibrar a distribuição de transações.

7 CONCLUSÃO

O crescimento do uso de meios de pagamento eletrônicos é uma realidade vigente. O uso do dinheiro como papel vem perdendo espaço para o uso de cartões de crédito ou débito e o comércio eletrônico através da Internet se consolida cada vez mais na medida em oferece garantias de segurança e efetividade. Nesse contexto, os sistemas de processamento de transações eletrônicas precisam estar capacitados para amparar exigências de eficiência, disponibilidade e segurança.

A arquitetura GetLB propõe uma solução focada na realidade da empresa GetNet que, atualmente, em um de seus núcleos, conta com um sistema de balanceamento de carga com abordagem Round Robin, que pode oferecer restrições para projetos de expansão da sua rede para outros países. Além disso, para atender ao volume crescente projetado de 20% ao ano, necessitaria de elevados investimentos na aquisição de novas máquinas para garantir os mesmos níveis de desempenho e disponibilidade atuais.

Os fatores que podem ser destacados na arquitetura GetLB são os seguintes:

i. Arquitetura de comunicação, que possibilita uso de notificações e atualizações periódicas do chaveador;

ii. Interações e notificações assíncronas entre as máquinas para manter informações do chaveador atualizadas, atenuando *overhead* de comunicação;

iii. Estado das máquinas monitorado para identificar capacidade de processamento para transações heterogêneas;

iv. Suportar topologias de máquinas heterogêneas;

v. Escalonador de transações LL, utiliza uma heurística que combina diferentes métricas para realizar o balanceamento de carga.

Com essa infraestrutura, além das questões pertinentes ao balanceamento de carga, se obtém uma maneira eficaz de evitar perda de transações, com um sistema que é pró-ativo e não reativo quando se apresentam problemas. Nas avaliações realizadas com o protótipo, foi notório o desempenho superior de GetLB, tanto em topologias homogêneas quanto heterogêneas, quando comparado ao Round-Robin. Situações críticas podem ser minimizadas, priorizando o processamento de transações financeiras. Além disso, GetLB capacita a rede para absorver novos produtos que necessitem de máquinas com processamento diferenciado ou mesmo alocado em centros de processamento diferentes.

No início da leitura, Capítulo 1, afirmou-se que a arquitetura resultante dessa abordagem de escalonamento visa melhorar o compartilhamento de recursos, otimizando o tempo médio de resposta com eficiente balanceamento de cargas em um sistema escalável e de fácil manutenção. Neste momento, concluindo a dissertação, é possível fundamentar essa afirmativa com as seguintes colocações:

i. Melhorar o compartilhamento de recursos: os recursos que tem menor capacidade de processamento recebem transações, conforme a distribuição de carga dada pelo escalonador LL;

ii. Otimizar o tempo médio de resposta: conforme os resultados do protótipo de GetLB, se verifica que encaminhando transações para máquinas mais capacitadas para o processamento, o tempo de resposta é menor;

iii. Eficiência de balanceamento: através da técnica heurística do escalonador LL, são considerados pontos relevantes do sistema, tais como tipo de transação, capacidade de processamento, latência de rede e fila de tarefas para o cálculo. Decorre assim, um tempo médio de resposta otimizado e com melhor compartilhamento de recursos;

iv. Escalável: com um sistema constituído de diferentes LANs, é possível através do uso das notificações, acrescer mais máquinas para operações específicas e aumentando a capacidade de processamento, bastando incluir a nova máquina nas configurações do escalonador.

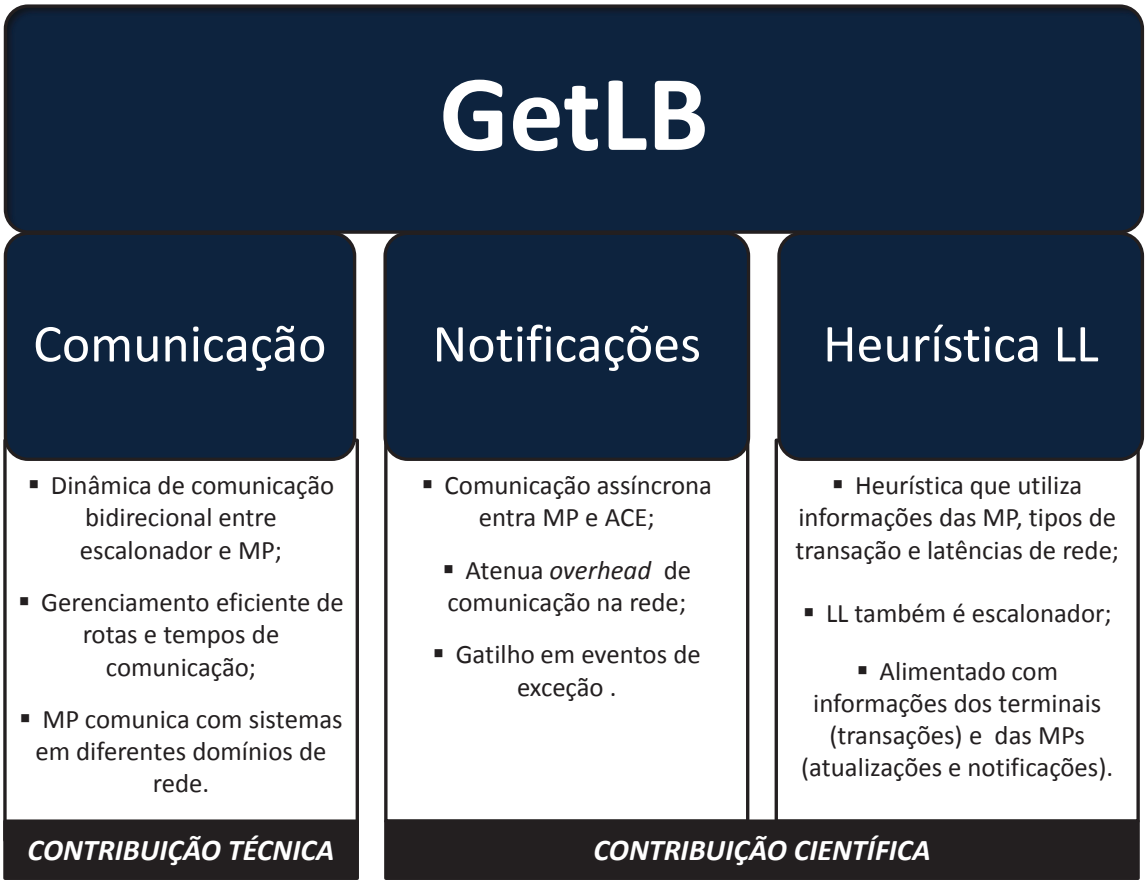
v. Fácil manutenção: através das notificações, assim que uma máquina é colocada em estado de manutenção, é possível notificar o chaveador para não despachar transações para esta máquina. Ao concluir a execução da manutenção, o escalonador é notificado da operacionalidade da máquina e o processamento das transações é retomado. Esse método possibilita manter alta disponibilidade sem prejuízo para os usuários.

7.1 Contribuições

Como um todo, a infraestrutura GetLB, conforme apresentado na Figura 39, possui três bases de sustentação: (i) comunicação, (ii) notificações e (iii) heurística LL. A base oferecida através da comunicação pode ser considerada a contribuição técnica deste trabalho, visto que constitui um arcabouço de comunicação que viabiliza o tráfego de informações entre os vários componentes do sistema, garantindo a execução eficaz dos processos de troca de dados. As notificações e a heurística LL constituem as bases da contribuição científica da dissertação. Quanto às notificações, correspondem à base que proporciona um sistema pró-ativo e com baixo custo de comunicação de rede, dado através de comunicação assíncrona entre os recursos e o centralizador das decisões de balanceamento. A heurística LL, por sua vez, utiliza o conjunto de informações do sistema transacional, tais como tipo e relevância das transações recebidas e estado das máquinas processadoras, caracterizados pela heterogeneidade, e proporciona um gerenciamento eficiente da distribuição de carga entre as máquinas processadoras.

Mesmo com os resultados satisfatórios apresentados no Capítulo 6, os testes apontaram para dois fatores que precisam ser aprofundados. O primeiro se refere à distribuição de tarefas entre os recursos de uma arquitetura heterogênea. Nas avaliações, máquinas com pouca capacidade

Figura 39: Três bases da infraestrutura GetLB.



Fonte: Desenvolvido pelo próprio autor.

permaneceram praticamente em repouso. Análises preliminares indicam que, adicionando um fator de penalização no termo da equação relacionado à ocupação das filas, pode auxiliar na obtenção de um melhor equilíbrio na distribuição. Além deste, outro fator que demanda maior análise, está relacionado ao tempo de atualização das informações no ACE. Atualizações muito frequentes podem resultar em *overhead* de comunicação na rede, porém, é preciso encontrar um ponto de equilíbrio entre uma pronta resposta do chaveador, sobretudo em situações dinâmicas do sistema, e o elevado tráfego de dados na rede.

Ao todo, GetLB compreende um conjunto de medidas que preenchem lacunas identificadas nos sistemas transacionais atuais e também está alinhado com as expectativas de crescimento do mercado de cartões. Além disso, a arquitetura proposta oferece suporte não apenas para o crescimento que o mercado vem experimentando ano a ano, mas ao impulso significativo que deve receber por causa dos eventos da Copa do Mundo e das Olimpíadas, com desafios que devem exigir flexibilidade para adaptação de produtos e de interconexão com diferentes sistemas de diferentes regiões.

7.2 Trabalhos Futuros

Em adição, trabalhos futuros incluem a publicação de novos artigos. Até o momento os trabalhos de balanceamento de carga em sistemas de transações eletrônicas financeiras foram publicados em congressos e revistas científicas. Além dos congressos regionais como a (i) ERAD 2012 (Escola Regional de Alto Desempenho) e (ii) ERRC 2012 (Escola Regional de Redes de Computadores) também foi publicado na (iii) Revista de Sistemas de Informação da FSMA 2012 (Faculdade Salesiana Maria Auxiliadora) e recentemente submetido ao (iv) Simpósio em Sistemas Computacionais (WSCAD-SSC) 2013.

Futuras atividades relacionadas à infraestrutura GetLB, estão associadas principalmente à aplicação do algoritmo nos sistemas de um sistema transacional real. Essa atividade passa, inicialmente, pelo porte da solução para uma linguagem de programação voltada para o alto desempenho. Em seguida, a elaboração de um plano de ação que viabilize o implantação da solução no sistema real sem interferir no que está em andamento. Essa iniciativa deve ocorrer com a preparação de uma prova de conceito em linguagem C, utilizando *sockets* para comunicação entre as máquinas. Segue com um projeto piloto controlado e, na medida que os sistemas estiverem estáveis, encaminhar a implantação. Além desse projeto, também se pretende seguir no desenvolvimento das equações e da melhoria contínua dos resultados de balanceamento. Estas atividades devem ser publicadas em trabalhos científicos, através de artigos e publicações em revistas científicas.

O futuro das transações financeiras eletrônicas segue em ritmo acelerado e aponta cada vez mais para a diversificação de formas de pagamento. O uso de *tokens*, pagamentos móveis através de *smart phones*, *smartcards* e captura de imagens são apenas alguns exemplos da diversidade de maneiras de efetuar uma transação eletrônica. Associado a essas tecnologias, sempre está o desafio relacionado à segurança da informação, que demanda muitas vezes esforço computacional para aplicar processos que degradam o desempenho de um sistema. Nesse contexto, manter um sistema transacional sempre voltado para o alto desempenho e alta disponibilidade são desafios constantes. Esse contexto justifica estudos voltados para essa área da computação, que ainda encontra pouca literatura entre os trabalhos acadêmicos. Contudo, constituem um tema bastante atual, com muitos desafios que se encontra em franco desenvolvimento e, portanto, muito motivador.

REFERÊNCIAS

ABECS. **Associação Brasileira de Empresas de Cartões de Crédito e Serviços** : resultados 1o trimestre de 2013. http://www.abecs.org.br/site2012/artigos.asp.

ANDRADE, A.; JOST, T.; RIGHI, R.; CHIWIACOWSKY, L.; COSTA, C. Superestimado ou subestimado? Definindo o Tamanho Apropriado do Parque Computacional para o Processamento de Transações Eletrônicas. In: REVISTA DE SISTEMAS DE INFORMAÇÃO DA FSMA N 11 2013, 2013. **Anais...** [S.l.: s.n.], 2013. p. 17–26.

APPLEBY, K.; FAKHOURI, S.; FONG, L.; GOLDSZMIDT, G.; KALANTAR, M.; KRISHNAKUMAR, S.; PAZEL, D.; PERSHING, J.; ROCHWERGER, B. Oceano-SLA based management of a computing utility. In: INTEGRATED NETWORK MANAGEMENT PROCEEDINGS, 2001 IEEE/IFIP INTERNATIONAL SYMPOSIUM ON, 2001. **Anais...** [S.l.: s.n.], 2001. p. 855–868.

ARAUJO, C.; SOUSA, E.; MACIEL, P.; CHICOUT, F.; ANDRADE, E. Performance Modeling for Evaluation and Planning of Electronic Funds Transfer Systems with Bursty Arrival Traffic. In: INTENSIVE APPLICATIONS AND SERVICES, 2009. INTENSIVE '09. FIRST INTERNATIONAL CONFERENCE ON, 2009. **Anais...** [S.l.: s.n.], 2009. p. 65 –70.

AVIZIENIS, A.; LAPRIE, J.-C.; RANDELL, B.; LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. **Dependable and Secure Computing, IEEE Transactions on**, [S.l.], v. 1, n. 1, p. 11 – 33, jan.-march 2004.

BALBO, G. Introduction to Generalized Stochastic Petri Nets. In: BERNARDO, M.; HILLSTON, J. (Ed.). **Formal Methods for Performance Evaluation**. [S.l.]: Springer Berlin / Heidelberg, 2007. p. 83–131. (Lecture Notes in Computer Science, v. 4486). 10.1007/978-3-540-72522-0_3.

BECKER, W. Dynamic Balancing Complex Workload in Workstation Networks - Challenge, Concepts and Experience. In: PROCS. HPCN-95, MILAN, 1995. **Anais...** Springer, 1995. p. 407–412.

BERMAN, P.; CHARIKAR, M.; KARPINSKI, M. On-line load balancing for related machines. In: DEHNE, F.; RAU-CHAPLIN, A.; SACK, J.-R.; TAMASSIA, R. (Ed.). **Algorithms and Data Structures**. [S.l.]: Springer Berlin Heidelberg, 1997. p. 116–125. (Lecture Notes in Computer Science, v. 1272).

BERNICK, D.; BRUCKERT, B.; VIGNA, P. D.; GARCIA, D.; JARDINE, R.; KLECKA, J.; SMULLEN, J. NonStop; Advanced Architecture. In: INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS, 2005., 2005, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2005. p. 12–21. (DSN 05).

BRANCO, K. R. L. J. C. **Índices de carga e desempenho em ambientes paralelos/distribuídos – modelagem e métricas**. 2004. Tese (Doutorado em Ciência da Computação) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2004.

CASAVANT, T. L.; KUHL, J. G. A taxonomy of scheduling in general-purpose distributed computing systems. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v. 14, p. 141–154, February 1988.

CESARIO, E.; GRILLO, A.; MASTROIANNI, C.; TALIA, D. A Sketch-Based Architecture for Mining Frequent Items and Itemsets from Distributed Data Streams. In: CLUSTER, CLOUD AND GRID COMPUTING (CCGRID), 2011 11TH IEEE/ACM INTERNATIONAL SYMPOSIUM ON, 2011. **Anais...** [S.l.: s.n.], 2011. p. 245–253.

CHENZHONG XU LAU, F. C. Load Balancing in Parallel Computers Theory and Practice. In: LOAD BALANCING IN PARALLEL COMPUTERS THEORY AND PRACTICE, 1997. **Anais...** [S.l.: s.n.], 1997. (The Springer International Series in Engineering and Computer Science, Vol. 381).

CHU, W.; LAN, L.-T. Task Allocation and Precedence Relations for Distributed Real-Time Systems. **Computers, IEEE Transactions on**, [S.l.], v. C-36, n. 6, p. 667 –679, june 1987.

CISCO SYSTEMS, I. **Cisco Catalyst 6500 and 6500-E Series Data Sheet**. http://www.cisco.com/en/US/prod/collateral/modules/ps2797/ps5138/product_data_sheet 09186a00800ff916_ps708_Products_Data_Sheet.html.

CLARK, C.; FRASER, K.; H, S.; HANSEN, J. G.; JUL, E.; LIMPACH, C.; PRATT, I.; WARFIELD, A. Live Migration of Virtual Machines. In: IN PROCEEDINGS OF THE 2ND ACM/USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION (NSDI), 2005. **Anais...** USENIX Association, 2005. p. 273–286.

COMPAQ. **ServerNet - A High Bandwith, Low Latency Cluster Interconnection**. [S.l.]: Compaq Computer Corporation, 1998.

CORDEIRO, T.; DAMALIO, D.; PEREIRA, N.; ENDO, P.; PALHARES, A.; GONçALVES, G.; SADOK, D.; KELNER, J.; MELANDER, B.; SOUZA, V.; MåNGS, J.-E. Open Source Cloud Computing Platforms. In: GRID AND COOPERATIVE COMPUTING (GCC), 2010 9TH INTERNATIONAL CONFERENCE ON, 2010. **Anais...** [S.l.: s.n.], 2010. p. 366–371.

EAGER, D.; LAZOWSKA, E.; ZAHORJAN, J. Adaptive load sharing in homogeneous distributed systems. **Software Engineering, IEEE Transactions on**, [S.l.], v. SE-12, n. 5, p. 662 –675, may 1986.

EDWARDS, C. Getting a purchase on christmas - IT Ecommerce. **Engineering Technology**, [S.l.], v. 4, n. 21, p. 54 –56, 15 2010.

EL KABBANY, G.; WANAS, N.; HEGAZI, N.; SHAHEEN, S. A Dynamic Load Balancing Framework for Real-time Applications in Message Passing Systems. **International Journal of Parallel Programming**, [S.l.], v. 39, p. 143–182, 2011.

EL-REWINI, H.; ALI, H. H.; LEWIS, T. Task Scheduling in Multiprocessing Systems. **Computer**, Los Alamitos, CA, USA, v. 28, n. 12, p. 27–37, Dec. 1995.

ENDO, P. T.; GONÇALVES, G. E.; KELNER, J.; SADOK, D. A survey on open-source cloud computing solutions. **VIII Workshop em Clouds, Grids e Aplicações**, [S.l.], p. 3–16, 2010.

GETNET. **GetNet Tecnologia em Captura e Processamento de Transações H.U.A. LTDA.** www.getnet.com.br.

GIRARDI, G.; CAMARGO, M. Forecast production volume: a case study. In: COMPUTERS INDUSTRIAL ENGINEERING, 2009. CIE 2009. INTERNATIONAL CONFERENCE ON, 2009. **Anais...** [S.l.: s.n.], 2009. p. 1747 –1750.

GRAEBIN, L.; ROSA RIGHI, R. da. jMigBSP: object migration and asynchronous one-sided communication for bsp applications. In: PARALLEL AND DISTRIBUTED COMPUTING, APPLICATIONS AND TECHNOLOGIES PDCAT, 2011 12TH INTERNATIONAL CONFERENCE ON, 2011. **Anais...** [S.l.: s.n.], 2011. p. 35–38.

GRANOVSKY, A. A. **Scalability**: comparison of dynamic vs static load balancing on dual-core opteron 290 based infiniband rocks cluster. http://classic.chem.msu.su/gran/gamess/dlb_benches.html.

GUPTA, D.; GARDNER, R.; CHERKASOVA, L. **XenMon**: qos monitoring and performance profiling tool. [S.l.]: HP Labs, Technical Report, 2005.

HARRINGTON, C. **The Future of Workload Management**. www.hpcwire.com/hpcwire/2012-10-12/the_future_of_workload_management.html.

HU, J.; GU, J.; SUN, G.; ZHAO, T. A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment. In: PARALLEL ARCHITECTURES, ALGORITHMS AND PROGRAMMING (PAAP), 2010 THIRD INTERNATIONAL SYMPOSIUM ON, 2010. **Anais...** [S.l.: s.n.], 2010. p. 89–96.

HU, Y. F.; BLAKE, R. J. Progress in computer research. In: COLUMBUS, F. (Ed.). **Load balancing for unstructured mesh applications**. Commack, NY, USA: Nova Science Publishers, Inc., 2001. p. 117–148.

ILIC, A.; SOUSA, L. On Realistic Divisible Load Scheduling in Highly Heterogeneous Distributed Systems. In: PARALLEL, DISTRIBUTED AND NETWORK-BASED PROCESSING (PDP), 2012 20TH EUROMICRO INTERNATIONAL CONFERENCE ON, 2012. **Anais...** [S.l.: s.n.], 2012. p. 426 –433.

STANDARDIZATION, I. O. for (Ed.). **ISO 8583-1993**: financial transaction card originated messages – interchange message specifications. [S.l.: s.n.], 1993.

JAIN, R. **The Art of Computer Systems Performance Analysis**: techniques for experimental design, measurement, simulation, and modeling. [S.l.]: Wiley, 1991.

KELTON, W.; SADOWSKI, R.; STURROCK, D. **Simulation with Arena**. 4. ed., internat. ed.. ed. Boston [u.a.]: McGraw-Hill Higher Education, 2007. (McGraw-Hill series in industrial engineering and management science).

KON, F.; CAMPBELL, R.; MICKUNAS, M.; NAHRSTEDT, K.; BALLESTEROS, F. 2K: a distributed operating system for dynamic heterogeneous environments. In: HIGH-PERFORMANCE DISTRIBUTED COMPUTING, 2000. PROCEEDINGS. THE NINTH INTERNATIONAL SYMPOSIUM ON, 2000. **Anais...** [S.l.: s.n.], 2000. p. 201 –208.

KOOIJMANS, A. L. Design and implementation of an efficient load-balancing method for virtual machine cluster based on cloud service. In: A GUIDE TO USING ACI WORLDWIDE'S BASE24-ES ON Z/OS, 2007. **Anais...** IBM: International Technical Support Organization, 2007.

KRISHNA, K.; GANESHAN, K.; RAM, D. Distributed simulated annealing algorithms for job shop scheduling. **Systems, Man and Cybernetics, IEEE Transactions on**, [S.l.], v. 25, n. 7, p. 1102 –1109, jul 1995.

KWOK, Y.-K.; CHEUNG, L.-S. A new fuzzy-decision based load balancing system for distributed object computing. **Journal of Parallel and Distributed Computing**, [S.l.], v. 64, n. 2, p. 238 – 253, 2004.

LAWRENCE, R. **A survey of process migration mechanisms**. [S.l.]: University of Manitoba, 1998. http://people.ok.ubc.ca/rlawrenc/research/Papers/proc_mig.pdf.

LI, Y.; LAN, Z. A Survey of Load Balancing in Grid Computing. In: CIS'04, 2004. **Anais...** [S.l.: s.n.], 2004. p. 280–285.

LU, C.; LAU, S.-M. A performance study on load balancing algorithms with task migration. In: TENCON '94. IEEE REGION 10'S NINTH ANNUAL INTERNATIONAL CONFERENCE. THEME: FRONTIERS OF COMPUTER TECHNOLOGY. PROCEEDINGS OF 1994, 1994. **Anais...** [S.l.: s.n.], 1994. p. 357 –364 vol.1.

LU, K.; SUBRATA, R.; ZOMAYA, A. Towards Decentralized Load Balancing in a Computational Grid Environment. In: CHUNG, Y.-C.; MOREIRA, J. (Ed.). **Advances in Grid and Pervasive Computing**. [S.l.]: Springer Berlin / Heidelberg, 2006. p. 466–477. (Lecture Notes in Computer Science, v. 3947).

LüLING, R.; MONIEN, B. A dynamic distributed load balancing algorithm with provable good performance. In: ACM SYMPOSIUM ON PARALLEL ALGORITHMS AND ARCHITECTURES, 1993, New York, NY, USA. **Proceedings...** ACM, 1993. p. 164–172. (SPAA '93).

LULING, R.; MONIEN, B.; RAMME, F. Load balancing in large networks: a comparative study. In: PARALLEL AND DISTRIBUTED PROCESSING, 1991. PROCEEDINGS OF THE THIRD IEEE SYMPOSIUM ON, 1991. **Anais...** [S.l.: s.n.], 1991. p. 686 –689.

MALIK, S. Dynamic Load Balancing in a Network of Workstations, Research Report 95.515F. In: DYNAMIC LOAD BALANCING IN A NETWORK OF WORKSTATIONS, RESEARCH REPORT 95.515F, 2000. **Anais...** [S.l.: s.n.], 2000.

MARSAN, M. A.; BALBO, G.; CONTE, G.; DONATELLI, S.; FRANCESCHINIS, G. Modelling with Generalized Stochastic Petri Nets. **SIGMETRICS Perform. Eval. Rev.**, New York, NY, USA, v. 26, n. 2, p. 2–, Aug. 1998.

MASTERCARD. **A Look Inside MasterCard Global Technology and Operations**. "www.mastercard.com/us/company/en/docs/A_Look_Inside_GTO_Brochure_July_2009.pdf".

MATBOULI, H.; GAO, Q. An overview on web security threats and impact to e-commerce success. In: INFORMATION TECHNOLOGY AND E-SERVICES (ICITES), 2012 INTERNATIONAL CONFERENCE ON, 2012. **Anais...** [S.l.: s.n.], 2012. p. 1 –6.

MAURER, M.; BRANDIC, I.; SAKELLARIOU, R. Adaptive resource configuration for Cloud infrastructure management. **Future Generation Computer Systems**, [S.l.], v. 29, n. 2, p. 472–487, 2013.

MCHEICK, H.; MOHAMMED, Z.; LAKISS, A. Evaluation of Load Balance Algorithms. In: SOFTWARE ENGINEERING RESEARCH, MANAGEMENT AND APPLICATIONS (SERA), 2011 9TH INTERNATIONAL CONFERENCE ON, 2011. **Anais...** [S.l.: s.n.], 2011. p. 104 –109.

MCINTOSH, S.; KEPHART, J.; LENCHNER, J.; FERIDUN, M.; NIDD, M.; TANNER, A.; YANG, B.; BARABASI, I. Semi-automated data center hotspot diagnosis. In: NETWORK AND SERVICE MANAGEMENT (CNSM), 2011 7TH INTERNATIONAL CONFERENCE ON, 2011. **Anais...** [S.l.: s.n.], 2011. p. 1–7.

MELLO, R.; TREVELIN, L.; PAIVA, M.; YANG, L. Comparative study of the server-initiated lowest algorithm using a load balancing index based on the process behavior for heterogeneous environment. **Cluster Computing**, [S.l.], v. 9, p. 313–319, 2006.

MURATA, T. Petri nets: properties, analysis and applications. **Proceedings of the IEEE**, [S.l.], v. 77, n. 4, p. 541–580, 1989.

NASSAR, N.; MILLER, G. Method for secure credit card transaction. In: COLLABORATION TECHNOLOGIES AND SYSTEMS (CTS), 2013 INTERNATIONAL CONFERENCE ON, 2013. **Anais...** [S.l.: s.n.], 2013. p. 180–184.

NELSON, M.; LIM, B. hong; HUTCHINS, G. Fast transparent migration for virtual machines. In: IN PROCEEDINGS OF THE ANNUAL CONFERENCE ON USENIX ANNUAL TECHNICAL CONFERENCE, 2005. **Anais...** USENIX Association, 2005.

NI, L.; XU, C.-W.; GENDREAU, T. A Distributed Drafting Algorithm for Load Balancing. **Software Engineering, IEEE Transactions on**, [S.l.], v. SE-11, n. 10, p. 1153 – 1161, oct. 1985.

PARAISO, F.; MERLE, P.; SEINTURIER, L. Managing elasticity across multiple cloud providers. In: MULTI-CLOUD APPLICATIONS AND FEDERATED CLOUDS, 2013., 2013, New York, NY, USA. **Proceedings...** ACM, 2013. p. 53–60. (MultiCloud '13).

PATIL, S.; GOPAL, A. Cluster performance evaluation using load balancing algorithm. In: INFORMATION COMMUNICATION AND EMBEDDED SYSTEMS (ICICES), 2013 INTERNATIONAL CONFERENCE ON, 2013. **Anais...** [S.l.: s.n.], 2013. p. 104–108.

SHARMA, R.; NITIN, N. Optimal Method for Migration of Tasks with Duplication. In: COMPUTER MODELLING AND SIMULATION (UKSIM), 2012 UKSIM 14TH INTERNATIONAL CONFERENCE ON, 2012. **Anais...** [S.l.: s.n.], 2012. p. 510 –515.

SHIRAZI, B. A.; R., H. A.; ; KAVI, K. Scheduling and load balancing in parallel and distributed systems, IEEE Computer Science Press, Los Alamitos, CA. In: SCHEDULING AND LOAD BALANCING IN PARALLEL AND DISTRIBUTED SYSTEMS, 1995. **Anais...** [S.l.: s.n.], 1995.

SHIVARATRI, N.; KRUEGER, P.; SINGHAL, M. Load distributing for locally distributed systems. **Computer**, [S.l.], v. 25, n. 12, p. 33 –44, dec. 1992.

SINHA, P. K. **Distributed Operating Systems**: concepts and design. 1st. ed. [S.l.]: Wiley-IEEE Press, 1996.

SONG, J.; CHOO, H. K.; LEE, K. M. Application-level load migration and its implementation on top of PVM. **Concurrency: Practice and Experience**, [S.l.], v. 9, n. 1, p. 1–19, 1997.

SOTOMAYOR, B.; MONTERO, R. S.; LLORENTE, I.; FOSTER, I. Virtual Infrastructure Management in Private and Hybrid Clouds. **Internet Computing, IEEE**, [S.l.], v. 13, n. 5, p. 14–22, 2009.

SOUSA, E.; MACIEL, P.; ARAUJO, C.; CHICOUT, F. Performability evaluation of EFT systems for SLA assurance. In: PARALLEL DISTRIBUTED PROCESSING, 2009. IPDPS 2009. IEEE INTERNATIONAL SYMPOSIUM ON, 2009. **Anais...** [S.l.: s.n.], 2009. p. 1 –8.

SOUSA, E.; MACIEL, P.; SOUZA, D.; MEDEIROS, E.; LINS, F.; TAVARES, E. Capacity planning of EFT service hosted on elastic IaaS. In: SYSTEMS, MAN, AND CYBERNETICS (SMC), 2012 IEEE INTERNATIONAL CONFERENCE ON, 2012. **Anais...** [S.l.: s.n.], 2012. p. 1749–1754.

TANENBAUM, A. Distributed Systems, Principles and Paradigms. In: DISTRIBUTED SYSTEMS, PRINCIPLES AND PARADIGMS, 2007. **Anais...** Prentice Hall of India, 2007.

TONG, R.; ZHU, X. A Load Balancing Strategy Based on the Combination of Static and Dynamic. In: DATABASE TECHNOLOGY AND APPLICATIONS (DBTA), 2010 2ND INTERNATIONAL WORKSHOP ON, 2010. **Anais...** [S.l.: s.n.], 2010. p. 1 –4.

TOUMODGE, S. Applications of Petri Nets in Manufacturing systems; Modeling, Control, and Performance Analysis [Book review]. **Control Systems, IEEE**, [S.l.], v. 15, n. 6, p. 93, dec. 1995.

TSENG, L.-Y.; CHIN, Y.-H.; WANG, S.-C. A minimized makespan scheduler with multiple factors for Grid computing systems. **Expert Systems with Applications**, [S.l.], v. 36, n. 8, p. 11118 – 11130, 2009.

UPADHYAY, D.; PATEL, C. Scheduler in Cloud Computing using Open Source Technologies. **International Journal of Computer Technology and Applications**, [S.l.], v. 3, n. 3, p. 1093–1098, may-june 2012.

VISA. **Visa International Operating Regulations Core Principles**. [S.l.]: Visa Inc., 2012.

WANG, R.; LE, W.; ZHANG, X. Design and implementation of an efficient load-balancing method for virtual machine cluster based on cloud service. In: WIRELESS, MOBILE MULTIMEDIA NETWORKS (ICWMMN 2011), 4TH IET INTERNATIONAL CONFERENCE ON, 2011. **Anais...** [S.l.: s.n.], 2011. p. 321–324.

WANG, X.; ZHU, Z.; DU, Z.; LI, S. Multi-cluster Load Balancing Based on Process Migration. In: ADVANCED PARALLEL PROCESSING TECHNOLOGIES, 2007. **Anais...** Springer Berlin / Heidelberg, 2007. p. 100–110. (Lecture Notes in Computer Science).

WANG, Y.-T.; MORRIS, R. Load Sharing in Distributed Systems. **Computers, IEEE Transactions on**, [S.l.], v. C-34, n. 3, p. 204 –217, march 1985.

WARAICH, S. S. Classification of Dynamic Load Balancing Strategies in a Network of Workstations. In: FIFTH INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY: NEW GENERATIONS, 2008, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2008. p. 1263–1265.

WOOD, T.; SHENOY, P.; VENKATARAMANI, A.; YOUSIF, M. Sandpiper: black-box and gray-box resource management for virtual machines. **Comput. Netw.**, New York, NY, USA, v. 53, n. 17, p. 2923–2938, Dec. 2009.

ZEROHORA. **Economia**: faturamento maior. http://zerohora.clicrbs.com.br/rs/economia/noticia/2013/07/cartoes-movimentam-r-189-43-bilhoes-no-primeiro-trimestre-4216293.html.

ZHOU, S. A trace-driven simulation study of dynamic load balancing. **Software Engineering, IEEE Transactions on**, [S.l.], v. 14, n. 9, p. 1327 –1341, sep 1988.

ZHOU, W.; YANG, S.; FANG, J.; NIU, X.; SONG, H. VMCTune: a load balancing scheme for virtual machine cluster using dynamic resource allocation. In: GRID AND COOPERATIVE COMPUTING (GCC), 2010 9TH INTERNATIONAL CONFERENCE ON, 2010. **Anais...** [S.l.: s.n.], 2010. p. 81 –86.