UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Guilherme Maciel Ferreira

CARACTERIZAÇÃO COMPUTACIONAL PARA ALOCAÇÃO DISTRIBUÍDA PARA UMA CONFIGURAÇÃO COM INTERFACE NATURAL DE USUÁRIO

Florianópolis
2015

Guilherme Maciel Ferreira

CARACTERIZAÇÃO COMPUTACIONAL PARA ALOCAÇÃO DISTRIBUÍDA PARA UMA CONFIGURAÇÃO COM INTERFACE NATURAL DE USUÁRIO

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação, como requisito à obtenção do grau de Mestre em Ciência da Computação.
Orientador: Prof. Dr. Mario Antonio Ribeiro Dantas

Florianópolis
2015

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Ferreira, Guilherme Maciel
Caracterização Computacional para Alocação Distribuída para uma Configuração com Interface Natural de Usuário / Guilherme Maciel Ferreira ; orientador, Mario Antonio Ribeiro Dantas - Florianópolis, SC, 2015.
113 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da Computação. 2. Grade computacional heterogênea. 3. Balanceamento de carga. 4. Indicador de carga. 5. Interface Natual de Usuário. I. Dantas, Mario Antonio Ribeiro. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Guilherme Maciel Ferreira

CARACTERIZAÇÃO COMPUTACIONAL PARA ALOCAÇÃO DISTRIBUÍDA PARA UMA CONFIGURAÇÃO COM INTERFACE NATURAL DE USUÁRIO

Esta Dissertação foi julgada adequada para a obtenção do Título de "Mestre em Ciência da Computação", e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Florianópolis, 5 de março de 2015

_____

Prof. Ronaldo dos Santos Mello, Dr.
Coordenador do Programa

Banca Examinadora:

_____

Prof. Mario Antonio Ribeiro Dantas, Dr.
Orientador
Universidade Federal de Santa Catarina

_____

Prof. Douglas Dyllon Jeronimo de Macedo, Dr.
Universidade Federal de Sergipe

_____

Prof. Carlos Barros Montez, Dr.
Universidade Federal de Santa Catarina

_____

Prof. Carlos Becker Westphall, Dr.
Universidade Federal de Santa Catarina

À minha mãe *(in memoriam)*, a pessoa mais maravilhosa que conheci.

AGRADECIMENTOS

Mais do que um trabalho individual, está dissertação é fruto do esforço de diversas pessoas, que contribuíram e acreditaram em mim ao longo desses anos.

Em primeiro lugar, gostaria de agradecer ao meu orientador, o Prof. Mario Dantas, que me acolheu e me motivou a finalizar a dissertação. Sua motivação e competência serão sempre lembradas.

Agradeço também à minha querida e amada esposa, uma grande companheira, que me apoiou e deu forças nas horas mais difíceis. Ela suportou meu mau humor e abdicou do nosso tempo juntos para que eu pudesse desenvolver este trabalho. Tenho sorte de ter uma pessoa tão especial ao meu lado e que me proporcionou as maiores alegrias.

Agradeço aos meus pais. À minha falecida mãe, que me ensinou o valor da educação como forma de crescermos como cidadãos. E ao meu pai, cuja honestidade e integridade moral foram inquestionáveis. Mesmo não estando mais entre nós, levarei seu afeto sempre na mais íntima parte do meu coração e seus ensinamentos no âmago da minha alma e mente.

Por fim, agradeço ao doutorando Mathias Weber, o qual me auxiliou no entendimento e modificação do CVFlow.

“Só há espaço para criatividade e imaginação quando dominamos plenamente uma área de conhecimento. Não há gênio sem imensa carga de trabalho e transpiração.”

Dr. César Augusto Dartora

RESUMO

Em um sistema distribuído heterogêneo, como grades computacionais, a escolha do sistema computacional para processar uma tarefa é realizada por meio de heurísticas adotadas igualmente para todos os sistemas. Os métodos atuais para avaliação da carga computacional, em grades heterogêneas, não levam em consideração características qualitativas que afetam o desempenho. Sistemas computacionais aparentemente idênticos, com as mesmas características quantitativas (tal como a quantidade de núcleos de processamento e de memória), podem apresentar desempenhos desiguais. O método proposto consiste em uma política de informação ao balanceamento de carga e tem como objetivo mensurar a carga dos sistemas computacionais por meio da avaliação de seus recursos quantitativos, tanto os imutáveis (como a quantidade de núcleos de processamento) quanto os mutáveis (como o percentual de memória livre), e qualitativos, inerentes à arquitetura do sistema computacional. A comparação da carga computacional entre os sistemas permite que o balanceamento de carga seja realizado mesmo em sistemas distribuídos heterogêneos para que seja possível a escolha do sistema computacional no qual executar uma tarefa da forma mais eficiente. Esta pesquisa utiliza a ferramenta CVFlow, uma Interface Natural de Usuário destinada ao balanceamento de carga, para avaliar o método proposto. O experimento consiste no escalonamento de um conjunto de tarefas e na comparação do método proposto com o estado da arte presente na literatura. O método proposto fornece um conjunto de melhorias que distribuem a carga de forma mais homogênea entre os sistemas computacionais, evitando, assim, sobrecarregar um sistema específico, além de oferecer um desempenho superior na execução do conjunto de tarefas.

Palavras-chave: Sistema computacional distribuído. Grade computacional heterogênea. Balanceamento de carga. Indicador de carga. Interface Natural de Usuário.

ABSTRACT

In a distributed heterogeneous system, such as grids, the choice of a computer system to process a task is performed by means of heuristics adopted equally for all systems. Current methods for assessing the computing load, on heterogeneous grids, do not take into account qualitative characteristics that affect performance. Computer systems apparently identical, with the same quantitative traits (such as the number of processing cores and memory), may provide different performance. The proposed method consists of an information policy to load balancing. It aims to measure the load of a computer systems through the assessment of their quantitative and qualitative features. Quantitative, both immutable (as the number of cores) and mutable (as the percentage of free memory). And the qualitative, inherent to the computer system architecture. Comparison of computational load between systems allows load balancing to be performed even in heterogeneous distributed systems, to be able to choose the computer system on which to perform a task more efficiently. This research uses the CVFlow tool, a Natural User Interface intended for load balancing, to evaluate the proposed method. The experiment consists of the scheduling of a set of tasks and the comparison of the proposed method with the state of the art. The proposed method provides a set of improvements that distribute the load more evenly among computer systems, avoid overloading a particular system, and provides a better performance on the execution of the set of tasks.

Keywords: Distributed computing system. Heterogeneous computing grid. Load balancing. Load index. Natural User Interface.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CLI	*Command Line Interface* (Interface por Linha de Comando)
CPU	*Central Processing Unit* (Unidade Central de Processamento)
EP	Elemento Processador, sinônimo de Sistema Computacional
E/S	Entrada/Saída
FCFS	*First Come, First Served* (Primeiro a Chegar, Primeiro a ser Atendido)
FIFO	*First In, First Out* (Primeiro a Entrar, Primeiro a Sair)
FLOPS	*FLoating-point Operations Per Second* (Operações de Ponto flutuante Por Segundo)
GUI	*Graphical User Interface* (Interface Gráfica de Usuário)
MFLOPS	*Million of FLOPS*
NUI	*Natural User Interface* (Interface Natural de Usuário)
NUMA	*Non Uniform Memory Access* (Acesso Não-Uniforme à Memória)
RAM	*Random Access Memory* (Memória de Acesso Aleatório)
RISC	*Reduced Instruction Set Computer* (Computador com Conjunto Reduzido de Instruções)
SMP	*Symmetric Multiprocessing* (Multi processamento simétrico)
SO	Sistema Operacional
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

O presente estudo trata dos indicadores de carga, mecanismo utilizado para coletar informações necessárias à tomada de decisões por parte dos algoritmos de balanceamento de carga. Este capítulo contextualiza o problema, em seguida apresenta a motivação e descreve os objetivos a serem alcançados com está pesquisa. Também são discutidas as justificativas para o desenvolvimento desta dissertação, as limitações, o método de pesquisa adotado e a forma como o trabalho está encadeado.

1.1 CONTEXTUALIZAÇÃO

Balanceamento de carga é uma forma de escalonamento de tarefas utilizada em sistemas computacionais paralelos e distribuídos; e tem por objetivo maximizar a utilização dos sistemas computacionais e minimizar o tempo total de execução das tarefas (TONG, 2009) (SHIRAZI, 1995; DEVINE, 2005). A função desse mecanismo é distribuir entre os sistemas computacionais interconectados a responsabilidade pelo processamento das tarefas, que representa a carga computacional exercida sobre esse sistema (CALZAROSSA, 1993; BARAK, 1993).

1.1.1 Ambientes para balanceamento de carga

O balanceamento de carga pode ser realizado em sistemas distribuídos homogêneos ou heterogêneos. Onde os sistemas compostos por elementos com as mesmas capacidades computacionais, como agregados computacionais, são homogêneos (TANENBAUM, 2006). Já aqueles constituídos por elementos com capacidades distintas, como grades e nuvens computacionais, são heterogêneos; podendo ser ainda subdividido em heterogeneidade de configuração e de arquitetura (ZHOU, 1993; COULOURIS, 2012).

Os sistemas distribuídos com heterogeneidade de configuração apresentam diferenças quantitativas na disponibilidade de recursos de hardware (como velocidade de processamento, quantidade de memória e capacidade de armazenamento). Por outro lado, os com diferentes arquiteturas de hardware apresentam heterogeneidade de arquitetura (FERREIRA, 2015).

1.1.2 Objetivos do balanceamento de carga

Dispondo de um conjunto de sistemas computacionais para executar uma determinada tarefa, a escolha de qual desses deverá executá-la é fundamentada em algum objetivo relacionado ao desempenho ou à redução no consumo de energia elétrica (PINHEIRO, 2001) (SHIRAZI, 1995).

1.1.2.1 Desempenho

Dentre os objetivos relacionados ao desempenho, destacam-se a minimização do tempo de execução e do atraso de comunicação e a maximização do uso dos recursos computacionais, sendo a quantidade e a velocidade dos processadores os principais fatores que afetam o tempo de execução das tarefas (HENDRICKSON, 2000). Todavia, o desempenho não se resume simplesmente aos fatores citados anteriormente; uma vez que, isoladamente, a maximização desses recursos não é garantia de uma execução mais veloz, pois existe uma gama de características – de hardware e de software – a serem consideradas e que afetam o tempo de execução de uma tarefa em um sistema computacional (BOSQUE, 2013; BRANCO, 2006):

Por exemplo, a latência e a velocidade da conexão de rede entre os sistemas computacionais influenciam na transferência dados, principalmente em aplicações que trafegam grande volume de dados, como imagens médicas de alta resolução. Se a velocidade de execução das aplicações depende da velocidade de transferência desses dados, a execução é influenciada pela rede (TONG, 2009);

As estruturas de hardware e de software influenciam no tempo de execução. A primeira influencia a resposta dos processadores, onde suas características intrínsecas de arquitetura e de quantidade de memória principal afetam tanto o desempenho quanto a quantidade de tarefas que podem ser processadas simultaneamente (BOSQUE, 2013). A última, mesmo em um sistema computacional que possua um número maior de requisições de execução de tarefas, o tempo total que todas elas ocuparão no sistema pode ser inferior ao tempo que um conjunto menor ocupará em outro sistema (CALZAROSSA, 1993).

Para avaliar esses fatores, de acordo com Li, existem dois métodos: a modelagem de carga e a previsão de desempenho (LI, 2007). Onde aquela consiste em criar modelos matemáticos para representar a

carga computacional, enquanto que a esta tem como objetivo fornecer informações em tempo real de importantes métricas de desempenho, tal como: tempo de espera na fila de processos.

1.1.2.2 Consumo de energia

Devido aos grandes custos relativos ao consumo de energia elétrica nos centros de processamento de dados, durante a última década tem crescido o interesse pela contenção desse insumo, levando o debate para além do foco em desempenho. Nesse caso, o maior desafio às técnicas de diminuição do consumo de energia elétrica é o *tradeoff* entre desempenho e economia (GE, 2013). Para que se tenha uma redução no consumo, deve-se fornecer menos energia aos elementos da rede (servidores, comutadores, roteadores, etc.) e de processamento, o que implica em desligar alguns desses equipamentos ou reduzir sua capacidade de processamento (FARIA, 2014; LEVERICH, 2010).

1.2 O PROBLEMA

Realizar o balanceamento de carga em sistemas distribuídos homogêneos é mais simples do que nos heterogêneos e, além disso, existe um volume maior de pesquisa científica sobre (BRANCO, 2006; LI, 2004). Entretanto, como nuvens e grades computacionais, tornaram-se o padrão para a computação de alto desempenho, criando heterogeneidades, o presente estudo será direcionado a esse timo de sistema. Segundo Bosque, a próxima geração de supercomputadores virá da integração entre grandes sistemas distribuídos e heterogêneos (BOSQUE, 2013).

Balancear a carga entre sistemas computacionais heterogêneos, os quais possuem elementos com capacidades distintas, adiciona complexidades e dificuldades que não existem no balanceamento de carga entre sistemas homogêneos (BRANCO, 2006). Além de diferenças quantitativas de capacidade, podem apresentar diferenças qualitativas, como, por exemplo, diferentes arquiteturas (FERREIRA, 2015). Nesse contexto, avaliar e comparar a capacidade de realização de trabalho entre sistemas computacionais heterogêneos, de modo a realizar o balanceamento de carga, se apresenta como um problema de pesquisa atual.

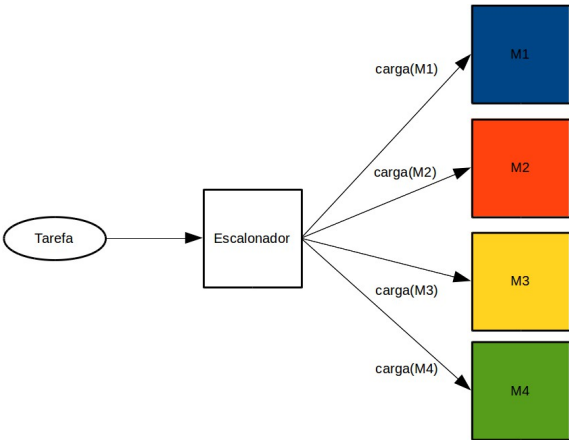

Figura 1: Escalonamento de tarefas
Fonte: o próprio autor

A figura 1 ilustra a questão que esta pesquisa procura responder: dado um conjunto de sistemas computacionais, qual seria o mais apropriado para executar uma determinada tarefa? Essa pergunta norteia o desenvolvimento das técnicas de balanceamento de carga (COULOURIS, 2012). Para tal função, é utilizada a informação de carga computacional, coletada nos sistemas computacionais, para que o escalonador decida qual desses sistemas fornecerá o melhor desempenho para a execução da tarefa.

1.3 OBJETIVOS

O objetivo desta pesquisa é fornecer um método para avaliar qual sistema computacional, dentre um conjunto de sistemas disponíveis ao usuário, executará de forma mais eficiente uma determinada tarefa. Em outras palavras, o desígnio é aprimorar o método de gerenciamento de recursos para obter o máximo desempenho em sistemas computacionais heterogêneos.

1.3.1 Objetivo geral

Esta pesquisa tem como objetivo desenvolver um indicador de carga para sistemas distribuídos heterogêneos; em especial para aqueles

com arquiteturas heterogêneas. O indicador possibilita ao balanceador escolher o sistema computacional que fornece o melhor desempenho para a execução de uma tarefa.

1.3.2 Objetivo específico

É possível detalhar o objetivo geral nos seguintes objetivos específicos:

a) Analisar os mais relevantes indicadores de carga existentes na literatura para reunir conceitos úteis a um indicador de carga para os sistemas computacionais heterogêneos;

b) Desenvolver um indicador de carga que represente com maior precisão o desempenho potencial em sistemas computacionais distribuídos arquiteturalmente heterogênea;

c) Comparar os indicadores de carga levantados e analisados anteriormente com o proposto neste trabalho.

1.4 JUSTIFICATIVA

Os indicadores de carga existentes na literatura não permitem uma comparação adequada do desempenho potencial entre sistemas computacionais arquiteturalmente heterogêneos; dessa forma, o balanceamento de carga é impreciso (FERREIRA, 2015; BRANCO, 2006). Faz-se, desse modo, necessário um meio de representar a diferença de desempenho potencial entre sistemas computacionais com arquiteturas distintas.

1.5 LIMITAÇÕES

Não é objetivo propor melhorias aos algoritmos de balanceamento de carga, como, por exemplo, preempção e migração de tarefas. Visa-se, sim, desenvolver uma forma de informar, em um único momento, qual sistema computacional oferece a maior capacidade computacional dentre todos os sistemas disponíveis em dada situação. Em outras palavras, esta pesquisa está voltada para a informação em si, não para seu uso. Também, sobre trabalhos futuros, melhorias no algoritmo de balanceamento de carga são discutidas na seção 7.1.

1.6 MÉTODO

Para alcançar os objetivos descritos na seção 1.3, foram adotadas as seguintes atividades:

a) Aplicação dos principais indicadores de carga desenvolvidos para sistemas computacionais distribuídos heterogêneos;

b) Experimentação para expor quais aspectos presentes nesses indicadores de carga são relevantes na representação da carga computacional em sistemas distribuídos heterogêneos;

c) Desenvolvimento de um indicador de carga aprimorado que represente diferenças de poder computacional até mesmo entre diferentes arquiteturas, com base nas características mais significativas de representação de sistemas heterogêneos e

d) Comparação, por meio de experimentos, do indicador de carga desenvolvido no passo anterior com os indicadores de carga mais relevantes da literatura.

Os experimentos utilizaram o mesmo algoritmo de balanceamento de carga, o mesmo ambiente de execução e os mesmos perfis de carga computacional para todos os indicadores de carga comparados.

O resultado do primeiro item é apresentado nos trabalhos relacionados presentes no capítulo 3. O segundo e o terceiro item são discutidos no capítulo 5. E o quarto item é apresentado no capítulo 6.

1.7 ORGANIZAÇÃO DO TEXTO

Esta dissertação está dividida da seguinte forma: o objetivo da pesquisa é introduzido no capítulo 1; o assunto, no qual a proposta está inserida, é descrito no capítulo 2; os trabalhos relacionados são abordados no capítulo 3; as ferramentas utilizadas no apoio e implementação da proposta são explicadas no capítulo 4; a proposta desta dissertação é descrita em detalhes no capítulo 5; a avaliação do modelo proposto é detalhada no capítulo 6; e por fim, as conclusões e trabalhados futuros são apresentados no capítulo 7.

2 ESCALONAMENTO EM SISTEMAS DISTRIBUÍDOS

Este trabalho trata de balanceamento de carga por meio do gerenciamento de recursos. Portanto, são apresentados os fundamentos a respeito do balanceamento de carga, onde são descritos os principais conceitos relacionados: os tipos de sistemas computacionais passíveis de balanceamento; o balanceamento de carga em si, a forma de escalonamento global de processos em um sistema distribuído; a caracterização de carga, cujo objetivo é compreender e criar modelos que representem o comportamento da carga computacional incidente sobre um sistema computacional; e os indicadores de carga, meio de informar aos algoritmos de balanceamento de carga acerca do real estado do sistema.

2.1 SISTEMAS COMPUTACIONAIS DISTRIBUÍDOS

Existem três classes de sistemas distribuídos: agregados computacionais, grades computacionais e nuvens computacionais. Em uma delas os componentes de software e hardware, localizados em computadores interconectados por meio de uma rede, comunicam e coordenam suas ações somente por meio da passagem de mensagens (TANENBAUM, 2006; COULOURIS, 2012). Esse tipo de sistema é composto por um conjunto de computadores independentes que, para seus usuários, aparenta ser um sistema computacional único e coerente (TANENBAUM, 2006).

2.1.1 Agregado computacional

Agregado computacional (do inglês *cluster*) é um sistema distribuído com perfil homogêneo (DANTAS, 2005; TANENBAUM, 2006). Para Coulouris esse tipo consiste em um conjunto de computadores interconectados que cooperam para fornecer uma única capacidade computacional de alto desempenho. Já para Barak se trata de coleção de computadores similares, conectados via rede, que utilizam o mesmo sistema operacional (BARAK, 1993; TANENBAUM, 2006).

2.1.2 Grade computacional

Grades computacionais (do inglês *grid*) são sistemas distribuídos

com perfil heterogêneo em que não é possível fazer nenhuma suposição a respeito do hardware, sistema operacional, rede ou políticas de segurança (TANENBAUM, 2006). Para Coulouris, esse é o precursor da computação em nuvem (COULOURIS, 2012).

2.1.3 Nuvem computacional

Nuvens computacionais são geralmente implementadas sobre agregados computacionais e são utilizadas para fornecer uma visão da computação como utilitário. Assim como as grades, esse tipo é composto por um conjunto heterogêneo de recursos de hardware (processadores e dispositivos de armazenamento, por exemplo) e de software. Dessa forma, permitem que seus usuários dispensem dispositivos de armazenamento e aplicativos locais (COULOURIS, 2012).

2.2 BALANCEAMENTO DE CARGA

O balanceamento de carga é uma política de escalonamento global de tarefas que se propõe a redistribuir a carga computacional entre um conjunto de sistemas computacionais interconectados (COULOURIS, 2012; SHIRAZI, 1995). O escalonamento local é responsável por decidir em qual momento uma tarefa executará em um mesmo processador. Por sua vez, o escalonamento global consiste em decidir em qual processador uma determinada tarefa será executada e se é necessário a migração para diferentes sistemas computacionais menos sobrecarregados. A redistribuição da carga computacional, nesse caso, ocorre geralmente dos nós mais para os menos sobrecarregados. E, por ser um mecanismo que distribuí recursos entre tarefas, o balanceamento de carga é considerado uma forma de escalonamento (BARAK, 1993).

A taxonomia apresentada por Shirazi divide o escalonamento global em estático e dinâmico. No primeiro as informações a respeito do tempo de execução de uma tarefa e os recursos necessários para executá-la são estabelecidas em tempo de compilação, antes da execução da tarefa e já no segundo são obtidas durante a execução da mesma, por meio de políticas de informação (SHIRAZI, 1995).

Ao contrário do escalonamento estático, onde a atribuição das tarefas a seus processadores é feita antes da execução, o balanceamento de carga assume pouco ou nenhum conhecimento prévio dos parâmetros

de execução de uma aplicação e atribuí as tarefas dinamicamente aos seus processadores (SHIRAZI, 1995). Para Coulouris o desempenho de qualquer sistema distribuído depende da distribuição da carga computacional entre os sistemas computacionais (COULOURIS, 2012). Já para Barak o balanceamento de carga busca equilibrar a carga computacional entre todos os processadores participantes de um sistema distribuído (BARAK, 1993).

2.2.1 Objetivos

Os algoritmos de balanceamento de carga possuem algum objetivo relacionado ao desempenho, tais como: diminuir o tempo de execução de uma aplicação, a latência de comunicação, o consumo de energia; e maximizar a vazão de um sistema (tarefas concluídas por unidade de tempo) (RAJ, 2013).

2.2.2 Políticas

Shirazi e Li afirmam que os algoritmos de balanceamento de carga são baseados em três políticas: a de informação; a de transferência, ou de tomada de decisão; e a de atribuição, ou migração de dados (LI, 2004; SHIRAZI, 1995). Cada uma dessas políticas é descrita em detalhes pelos autores como um conjunto diferente de políticas de balanceamento de carga, as quais estão subentendidas pelas políticas de Shirazi.

De acordo com Raj, o balanceamento de carga é composto pelas políticas de informação, de disparo, de recurso, de localização e de seleção. O sistema operacional MOSIX implementa as políticas de informação, transferência e atribuição descritas por Shirazi nos algoritmos de disseminação de informação, cálculo de carga e de consideração de migração, respectivamente (BARAK, 1993).

2.2.2.1 Política de informação

A política de informação diz respeito à quantidade, tipo e periodicidade de informação de carga que é passada aos tomadores de decisões e corresponde às medidas quantitativas de recursos que podem afetar a execução de uma tarefa, como, por exemplo, a quantidade de memória livre disponível (PINTO, 2004). Nesse caso, os tomadores de decisões são os sistemas computacionais capazes de avaliar se uma

tarefa deve ou não ser movida para outro elemento processador. O estado da carga computacional pode ser informado passivamente sob demanda, periodicamente ou ativamente ao mudar de estado (PINTO, 2004).

2.2.2.2 Política de transferência

De posse das informações de carga, definidas pela política de informação, a política de transferência determina sob quais condições uma tarefa deve ser transferida para outro elemento processador. Ou seja, com base nas informações de carga do elemento processador, é avaliado se uma tarefa deve ser movida para outro elemento processador. A política de transferência de Shirazi equivale à política de disparo de Raj.

2.2.2.3 Política de atribuição

Por fim, a política de atribuição identifica qual o elemento processador é o mais adequado para receber uma determinada tarefa, seja ela resultado de um escalonamento inicial ou da migração, quanto é transferida de outro elemento processador segundo a política de transferência. A política de atribuição de Shirazi é equivalente às políticas de recurso e de localização de Raj.

2.2.3 Despachador

Segundo Li, os algoritmos de balanceamento de carga são divididos em duas categorias: aqueles que possuem um despachador de requisições, chamado de *front-end*, e aqueles que não o possuem (LI, 2009). No primeiro o despacho das requisições é caracterizado por possuir uma única interface entre os clientes – que solicitam a execução de tarefas – e os elementos processadores que realizam o trabalho. Por outro lado, um sistema sem um elemento central para distribuir o trabalho possui diversas interfaces para solicitação de execuções. Nesse caso, apesar de não possuir um gargalo de comunicação, exige mais das conexões entre os elementos processadores.

2.3 CARACTERIZAÇÃO DE CARGA

A caracterização de carga computacional consiste na análise e na

criação de modelos para representar a carga de trabalho exercido sob um sistema computacional e também permite descrever e reproduzir o comportamento da carga computacional (MARTIN, 2006). O estudo leva ao aperfeiçoamento das arquiteturas de hardware, uma vez que a caracterização da carga consiste do comportamento do software que afeta a carga computacional (JOHN, 1999).

Há décadas é consenso entre os pesquisadores que o desempenho do software, executado em um sistema, depende de uma combinação tanto das características intrínsecas desse sistema (componentes de hardware e software), quanto da carga computacional que é processada (CALZAROSSA, 1993). Inclusive, pesquisadores (BOSE, 2006; JOHN, 1999) vem enfatizando a influência da caracterização da carga computacional no projeto de microprocessadores e arquiteturas de hardware.

O desempenho de uma tarefa é afetada por diversos fatores e, por essa razão, não é trivial fornecer nem mesmo a mais simples estimativa do tempo que uma tarefa levará para ser executada. Por isso, há pesquisas cujos objetivos são a criação de modelos matemáticos para fornecer uma previsão aproximada do tempo de execução e do consumo de recursos de uma tarefa (CALZAROSSA, 1993; LI, 2007; MORO, 2009; YOO, 2006).

Kliazovich afirma que existem três categorias de carga de trabalho passíveis de serem modeladas: computacionalmente intensiva; comunicacionalmente intensiva; e balanceada (KLIAZOVICH, 2010).

2.3.1 Computacionalmente Intensiva

As cargas de trabalho computacionalmente intensivas (CI) são destinadas a modelar tarefas de computação de alto desempenho (do inglês *High Performance Computing*). Esse tipo impõe uma alta carga sobre os elementos processadores, mas requer pouca transferência de dados entre os nós e é composta por tarefas *CPU bound*.

2.3.2 Comunicacionalmente Intensiva

A carga de trabalho comunicacionalmente intensiva, ou dado intensiva, exige grandes quantidade de transferência de dados, enquanto quase não requer um elevado processamento por parte dos elementos processadores.

2.3.3 Balanceada

O modelo balanceada descreve cargas de trabalho que possuem tanto requisitos de transferência de dados, quanto de processamento.

2.4 INDICADOR DE CARGA

Enquanto existem pesquisas voltadas à caracterização da carga computacional (MORO, 2009; LI, 2007; YOO, 2006; CALZAROSSA, 1993), este estudo tem como objetivo estimar o outro aspecto envolvido no balanceamento de carga: o impacto das características do sistema computacional na execução de uma tarefa, que são descritas por indicadores de carga.

O processo de balanceamento de carga depende da estimativa e da atualização de informações, representadas por um indicador de carga, do sistema computacional (RAJ, 2013; FERRARI, 1987). Esse indicador é uma medida quantitativa da carga em um sistema computacional e é definido como uma variável não-negativa: assumindo o valor zero, caso o recurso estiver ocioso; ou assumindo valores positivos crescentes no que a carga computacional aumenta naquele sistema (SHIRAZI, 1995).

O indicador de carga descreve as características do hardware e do sistema operacional que afetam a carga computacional e representa o resultado da coleta de informações a respeito da disponibilidade de vários recursos de um sistema computacional (BARAK, 1993). Enquanto a caracterização da carga se preocupa em modelar um comportamento futuro, o indicador modela o comportamento presente da carga computacional. Um bom indicador, Shirazi enumera, deve levar em conta não somente as necessidades de processamento, mas as necessidades de E/S e memória (SHIRAZI, 1995). Além disso, estabilidade e relação proporcional ao desempenho são características indispensáveis para um indicador de carga.

Tal como apresentado pela pesquisa bibliométrica realizada por Ferreira (FERREIRA, 2015), a maioria dos indicadores de carga presentes na literatura derivam de uma combinação da quantidade de tarefas na fila de prontos do processador, da taxa de utilização do processador, do tempo de resposta e do tempo de processamento. Todavia, diversos pesquisadores vêm desenvolvendo indicadores de carga com outros objetivos, visando distribuir as tarefas com o propósito

de reduzir ao máximo o consumo de energia elétrica (LEVERICH, 2010; FARIA, 2014).

É importante ressaltar que, apesar dos indicadores de carga representarem o estado de recursos computacionais, eles podem ser compostos por elementos tanto de hardware quanto de software, uma vez que o segundo interfere na disponibilidade do primeiro. Por exemplo, a quantidade de memória é um aspecto estritamente de hardware, enquanto que a quantidade de processos aguardando execução é uma característica do software.

2.4.1 Definição formal

O indicador de carga é uma equação composta por uma quantidade variável de parâmetros. Esses parâmetros representam os componentes do sistema computacional que afetam, ou são afetados, pela variação da carga computacional. A fórmula (1) representa a relação fundamental que permeia a construção de todos os indicadores de carga. Essa relação é perceptível nos indicadores apresentados no capítulo 3. A ideia por trás dos indicadores de carga é expressar uma relação de proporcionalidade entre os elementos que contribuem para o aumento da carga e aqueles que contribuem para a diminuição da carga de trabalho em um determinado sistema computacional.

$$carga(i) = \left\{ \frac{aumenta_{(carga(i))}}{diminui_{(carga(i))}} \right\} \tag{1}$$

O algoritmo de balanceamento de carga utiliza o indicador de carga como meio para escolher o sistema computacional que julga fornecer o melhor desempenho para a execução de uma determinada tarefa. Esse julgamento está embasado na hipótese de que o menor indicador de carga representa o sistema computacional que executa a tarefa de forma mais veloz. Em outras palavras, assume-se que o indicador seja diretamente proporcional ao tempo de execução, tal como exemplificado na fórmula (2).

$$tempo_{execucao}(tarefa_j) \equiv carga_M(i) \tag{2}$$

2.4.2 Componentes

Um indicador de carga serve para representar, numericamente, as características relacionadas à carga em um sistema computacional. O intuito é permitir que o algoritmo de balanceamento de carga tome decisões em relação a esse sistema computacional. Portanto, o indicador de carga deve ser modelado para representar as características que influenciam no balanceamento da carga. Para avaliar a qualidade de um indicador de carga, Ferrari sugere um conjunto de propriedades desejáveis, as quais dependem do objetivo do balanceamento de carga (FERRARI, 1987).

Para melhor compreensão, são descritas a seguir algumas das características (ou componentes) dos sistemas computacionais que influenciam os algoritmos de balanceamento de carga. As características mais comuns introduzidas são: tempo de resposta, poder de processamento, memória principal em uso e quantidade tarefas. O consumo energético é descrito em maiores detalhes por ser uma característica mais recente para composição dos indicadores de carga (FARIA, 2014).

2.4.2.1 Processador: tempo de resposta

Este é, à principio, o objetivo mais comum e há mais tempo pesquisado quando se trata de balanceamento de carga. Para essa característica Ferrari apresenta uma das mais antigas pesquisas encontradas a respeito de indicadores de carga com foco na diminuição do tempo de resposta. E para Stallings o tempo de resposta é a quantidade de tempo que transcorre entre o momento em que uma tarefa é submetida à execução e o instante em que a tarefa executada começa a fornecer resposta (STALLINGS, 2011).

2.4.2.2 Processador: poder de processamento

O poder de processamento de um sistema computacional é um dos elementos que mais impactam na capacidade de executar tarefas (STALLINGS, 2010). Sistemas computacionais com maior poder computacional tendem, nesse aspecto, a suportar maior carga computacional do que sistemas com menor poder de processamento (BRYANT, 2011).

2.4.2.3 Memória principal: quantidade em uso

A memória principal é um dos recursos que mais afeta o tempo de execução de processos (STALLINGS, 2011). A memória é utilizada pelo processador para armazenar os processos em execução e, dessa forma, a quantidade de memória principal disponível tem relação direta com a quantidade de processos que podem ser executados de forma concorrente (BRYANT, 2011). Apesar da disponibilidade de memória virtual tornar praticamente ilimitada a quantidade de programas que é possível executar, a memória principal ainda tem impacto na velocidade em que esses programas executarão (KERRISK, 2010; STALLINGS, 2011).

2.4.2.4 Sistema Operacional: processos na fila de prontos

Por meio da quantidade de memória disponível não é possível inferir a quantidade de processos que estão compartilhando o sistema, pois um processo grande pode ocupar a mesma quantidade de memória que vários outros menores (BRYANT, 2011; STALLINGS, 2011). Todavia, como a quantidade de processos em execução influencia na fatia de tempo disponível, é importante utilizar a quantidade de processos em execução na composição de indicadores de carga (BRYANT, 2011).

2.4.2.5 Consumo de energia elétrica

O consumo de energia elétrica representa uma grande parcela dos custos operacionais dos centros de processamento de dados (FARIA, 2014). Desse modo, evitar o desperdício e diminuir o consumo de eletricidade é um objetivo cada vez mais almejado em sistemas computacionais de alto desempenho (KLIAZOVICH, 2010).

De acordo com Leverich, existem duas formas de aumentar a eficiência energética de um agregado computacional, realizando ajustes no nível do sistema inteiro ou no nível de cada sistema computacional (LEVERICH, 2010):

- **Sistema distribuído**. Mantem ativos somente os nós necessários para a carga de trabalho, desligando ou mantendo em baixo consumo o restante dos nós. Essa abordagem é equivalente à técnica DNS (abreviação de *Dynamic Shutdown*) descrita por Kliazovich e a estratégia de provisionamento dinâmico (do inglês,

Dynamic Provisioning) (GE, 2013);

- **Sistema computacional**. Usado para que o nó se adeque a sua carga de trabalho, evitando desperdício de energia em componentes superdimensionados. É equivalente à técnica DVFS (abreviação de *Dynamic Voltage and Frequency Scaling*) (KLIAZOVICH, 2010) ou à estratégia de consumo proporcional (do inglês, *Energy Proportional Computing*) (GE, 2013).

Além das estratégias de provisionamento dinâmico e consumo proporcional, GE (2013) apresenta mais duas estratégias de economia de energia: Virtualização e PCS (abreviação de *Power Capping and Shifting*). A primeira consiste na agregação de múltiplas unidades computacionais virtuais (máquinas virtuais) em uma única unidade física; isso reduz a quantidade de máquinas físicas necessárias e, consequentemente, a necessidade de refrigeração dos equipamentos. (GE, 2013)

Já a estratégia de PCS parte da premissa que o desempenho de um processador está intimamente ligado à quantidade de energia alocada a ele, consistindo em um mecanismo de controle em dois níveis. O primeiro nível, *power capping*, estabelece um limite máximo na quantidade de energia alocada aos servidores do sistema distribuído e no segundo nível o *power shifting* ajusta dinamicamente a quantidade de energia alocada a cada servidor.

2.4.3 Grau de importância

Além da variação dos parâmetros que compõem o indicador de carga, pode-se modificar a influência que os parâmetros têm no cálculo do indicador. É possível alterar o grau de importância dos componentes individualmente ou de todos os componentes de uma única vez (TONG, 2009; LI, 2009; BOSQUE, 2013).

2.4.3.1 Grau de importância do componente

Neste trabalho, o primeiro método é chamado de grau de importância dos componentes, ilustrado por meio da fórmula (3) e segue a relação fundamental dos indicadores de carga (1), onde tanto os componentes que aumentam a carga (a_i) quanto aqueles que a diminuem (d_i), possuem coeficientes que escalam o valor do componente. Esses

coeficientes são únicos para todos os sistemas computacionais.

$$carga(i) = \left(\frac{grau_{a1} * a_1 + grau_{a2} * a_2 + ... + grau_{an} * a_n}{grau_{d1} * d_1 + grau_{d2} * d_2 + ... + grau_{dn} * d_n} \right) \quad (3)$$

O objetivo de dar peso aos diferentes componentes é adequar o indicador de carga a um determinado tipo de tarefa (TONG, 2009; LI, 2009). A caracterização de carga, apresentada na seção 2.3, oferece um meio de saber quais componentes devem ter maior relevância na composição do indicador de carga para determinados perfis de cargas computacionais.

2.4.3.2 Grau de importância do sistema computacional

O segundo método, chamado aqui de grau de importância do sistema computacional, é ilustrado pela fórmula (4). Nesse método, o grau de importância muda de acordo com o sistema computacional, não de acordo com o componente do indicador de carga. A abordagem de Bosque é um exemplo de indicador de carga que adota esse tipo de grau de importância (BOSQUE, 2013).

$$carga(i) = grau_{sistema} \times \left(\frac{a_1 + a_2 + ... + a_n}{d_1 + d_2 + ... + d_n} \right) \quad (4)$$

2.4.4 Uniformidade do sistema

Ferreira classifica os indicadores de carga sob dois aspectos: uniformidade e composição (FERREIRA, 2015). O primeiro diz respeito à diversidade de sistemas computacionais que o indicador suporta, sendo que ele pode ser projetado para lidar com sistemas homogêneos ou heterogêneos.

Os sistemas distribuídos homogêneos são compostos por sistemas computacionais com as mesmas características de processamento, enquanto que os heterogêneos possuem sistemas computacionais com características distintas. Por exemplo, sistemas com multiprocessadores simétricos (do inglês *Symmetric Multiprocessor Systems*) são homogêneos enquanto que grades computacionais, contendo máquinas

de diferentes arquiteturas, são heterogêneos.

2.4.5 Composição

A composição de um indicador de carga diz respeito à quais informações do sistema computacional são utilizadas para derivar o indicador de carga, que pode ser específica ou genérica. Naquele é utilizada uma única informação do sistema para compor o indicador e neste é derivado de uma combinação de critérios (ou parâmetros).

3 TRABALHOS RELACIONADOS

Os indicadores de carga e os algoritmos de balanceamento de carga, são abordados amplamente na literatura científica. Por isso, essa pesquisa baseou-se em indicadores de cargas existentes para propor um indicador mais adequado a sistemas distribuídos heterogêneos. Neste capítulo são descritos os mais relevantes indicadores de carga presentes na literatura, segundo a pesquisa bibliométrica realizada por Ferreira (FERREIRA, 2015). Também é descrito o indicador de carga presente na ferramenta utilizada para realizar os experimentos. Na seção 3.1 são descritos os indicadores de carga e na seção 3.2 esses indicadores de carga são comparados entre si sob diversos aspectos.

3.1 DESCRIÇÃO DOS INDICADORES DE CARGA

A seguir, são descritos os seis indicadores de carga coletados na literatura e o indicador de carga presente no CVFlow, a ferramenta utilizada nos experimentos. Para cada um dos itens são apresentadas sua descrição e a sua fórmula matemática, a qual representa a composição do indicador.

3.1.1 Quantidade de tarefas e de processadores

Lechuga (2010) apresenta um indicador de carga para uso em sistemas distribuídos heterogêneos de alto desempenho. Nesse indicador, a capacidade de processamento dos sistemas computacionais é representada pela quantidade de núcleos de processamento. E a carga computacional, exercida em cada sistema computacional, é representada pela quantidade de tarefas na fila de execução desse sistema.

O indicador de carga desenvolvido por Lechuga é implementado no CVFlow, a ferramenta utilizada nesta dissertação para a realização dos experimentos. Trata-se de uma ferramenta que permite criar e controlar o fluxo de execução de outros programas de computador. Os programas controlados pelo CVFlow podem ser executados local ou remotamente (WEBER, 2013).

A escolha do CVFlow como ferramenta de teste se deve ao alto grau de usabilidade de sua interface, ao seu código ser livre e disponível ao mestrando e ao seu orientador, e ao fato de ser desenvolvido na universidade na qual o orientador leciona e permitir acesso fácil aos

desenvolvedores da ferramenta. E para auxiliar na escolha do sistema computacional que executará um programa, o software dispõe de um indicador de carga que leva em consideração o tamanho da fila de requisições e a quantidade de processadores (LECHUGA, 2010), tal como mostrado na fórmula (5) e detalhado na seção 4.3.

$$carga(i) = \frac{JobsQueued + (JobsRunning * 0{,}5)}{CpuCount + (CpuIdle * 0{,}5)} \quad (5)$$

Na fórmula (5), quanto menor o valor do indicador de carga, menor o tempo previsto a ser gasto executando uma tarefa. O valor do indicador é inversamente proporcional à quantidade de processadores disponíveis e diretamente proporcional à quantidade de processos enfileirados e em execução em dado sistema computacional.

Poucas características de sistemas heterogêneos são levadas em consideração na composição desse indicador. Por exemplo:

- Características qualitativas de diferentes arquiteturas, como o conjunto de instruções, podem apresentar desempenhos distintos independente da quantidade de recursos. Somente a quantidade de processadores não é garantia de uma execução mais veloz, dado que a velocidade e a arquitetura dos mesmos também tem impacto (BRYANT, 2011; STALLINGS, 2010);
- Diversos recursos, como, por exemplo, rede, memória e disco, são exigidos no tipo de aplicação executada pelo CVFlow (algoritmos de processamento de imagens); entretanto o indicador de carga não representa esses recursos.

Outros fatores que tem grande influência no tempo de execução das tarefas são a quantidade de memória principal, a velocidade dos processadores e o tipo de arquitetura (BRYANT, 2011; STALLINGS, 2011). Entretanto, o indicador de carga apresentado aqui leva em consideração unicamente as características quantitativas: o tamanho da fila de requisições e a quantidade de processadores.

3.1.2 Combinação linear de tamanho das filas de recursos

Um dos mais antigos indicadores de carga disponíveis na literatura é o descrito por Ferrari (1987) e deriva de uma combinação

linear entre o tempo necessário para executar uma tarefa (s_j) e o tamanho da fila para utilizar um recurso (q_j):

$$carga(i) = \sum_{j=1}^{N} s_j \times q_j \quad (6)$$

Esse indicador de carga é representado pela fórmula (6), onde:

- N é a quantidade total de recursos com fila;
- s_j é o tempo que uma tarefa ocupa o recurso r_j; e
- q_j é o tamanho da fila para usar o recurso r_j.

Esse indicador de carga pode ser aplicado a uma gama de recursos distintos, como, por exemplo, processador, disco e rede. A exigência é que o recurso em questão disponha de uma fila para ser utilizado, tal como a fila de processos prontos no caso do processador.

3.1.3 Quantidade de tarefas e memória em uso

Lin (1987) propõe um indicador de carga para sistemas homogêneos baseado no que o autor chama de “modelo gradiente”. Esse indicador de carga utiliza tanto o número de tarefas quanto a quantidade de memória em uso. O indicador de carga é dado pela fórmula (7).

$$carga(i) = numberOfTasks + \left\{ \frac{parameters}{1 - memeoryInUse} \right\} \quad (7)$$

Os componentes da fórmula são:

- *numberOfTasks* que representa a quantidade de tarefas;
- *memoryInUse* é a quantidade de memória sendo utilizada;
- *parameters* é o parâmetro fornecido pelo simulador. O autor adota o valor 0,01 em seus experimentos.

3.1.4 Peso dinâmico de carga

No indicador de carga proposto por Tong (2009), $S=\{S_1, S_2,..., S_n\}$ é o conjunto de processadores disponíveis. Para cada S_i o indicador de carga é composto por uma combinação de seis parâmetros. E o

algoritmo de escalonamento geral é baseado em pesos, onde um sistema computacional com maior peso suporta mais requisições do que um com menor peso. O indicador de carga, representado pela fórmula (8), é construído pela soma dos parâmetros:

- *Cpu(i)%* representa a taxa de uso da CPU;
- *Mem(i)%* é a taxa de uso da memória;
- *W(i)* é a taxa de transmissão atual da rede;
- *Io(i)%* é a taxa de acesso ao E/S do disco;
- *Rt(i)* indica o tempo de resposta;
- *P(i)* representa a quantidade de processos.

$$carga(i) = \left\{ \begin{matrix} \pi_1 \times Cpu(i)\% + \pi_2 \times Mem(i)\% + \pi_3 \times W(i) + \\ \pi_4 \times Io(i)\% + \pi_5 \times Rt(i) + \pi_6 \times P(i) \end{matrix} \right\} \quad (8)$$

Cada parâmetro é influenciado por uma constante π_j, a qual define a importância de cada parâmetro em todos os sistemas computacionais. O grau de importância de cada parâmetro do indicador de carga é dado pela constante π_j. O somatório de todos os π_j deve ser 1. Esses graus de importância podem ser ajustados para adequar o indicador de carga a um determinado perfil de carga computacional.

3.1.5 Indicador de aceitação de carga

Bosque (2013) descreve um indicador de aceitação de carga (9), não um indicador de carga propriamente dito. Isso implica que: a relação descrita por autor é o inverso daquela representada pela fórmula (1). Esse indicador leva em consideração tanto o número de núcleos de processamento quanto a quantidade de tarefas.

$$carga_{aceitacao}(i) = \left\{ \frac{P_i}{P_{max}} \right\} \times \left\{ \frac{numberOfCores_i}{numberOfTasks_i + 1} \right\} \quad (9)$$

Para que esse indicador de aceitação de carga seja comparado aos demais indicadores de carga apresentados, é necessário inverter o divisor e o dividendo em ambos fatores. Esta dissertação utiliza a fórmula (10) para se referir ao indicador de carga proposto por Bosque

(2013).

$$carga(i) = \left\{ \frac{P_{max}}{P_i} \right\} \times \left\{ \frac{numberOfTasks_i + 1}{numberOfCores_i} \right\} \quad (10)$$

Se, em um determinado sistema computacional, *numberOfTasks* < *numberOfCores*, então o sistema pode aceitar novas tarefas. Caso contrário, se *numberOfTasks* >= *numberOfCores*, então o indicador leva em consideração fatores estáticos (a quantidade de núcleos de processamento) e dinâmicos (quantidade de tarefas em execução). Nesse caso, o indicador é computado pela fórmula (10), onde:

- P_i é o *i*-ésimo poder computacional, definido como a quantidade de trabalho realizada durante um período de tempo, dentre todos os sistemas computacionais participantes do balanceamento;
- P_{max} é o mais alto poder computacional dentre todos os sistemas computacionais, servindo como parâmetro de comparação para os outros sistemas;
- $numberOfTasks_i$ é a quantidade de tarefas sendo executadas pelo *i*-ésimo sistema computacional;
- $numberOfCores_i$ é o número de núcleos de processamento no *i*-ésimo sistema computacional.

Esse indicador de carga é puramente voltado ao processamento e todos os parâmetros que o compõe são diretamente influenciados pelo poder de processamento do processador. Os parâmetros P_i e P_{max}, medidos em operações por unidade de tempo (FLOPS), representam o poder computacional do sistema computacional *i* e do sistema computacional mais poderoso, respectivamente. E, ao utilizar P_{max} como referência para normalizar o poder de processamento de todos os sistemas computacionais, esse indicador de carga se torna adequado para uso em sistemas heterogêneos.

3.1.6 Vetor de indicador de desempenho

Branco (2006) não descreve um indicador de carga específico, mas, ao invés disso, fornece uma plataforma para adicionar outros indicadores de carga, que é composta por quatro recursos básicos,

separadamente calculados: processador, memória, disco e rede. Cada um desses recursos é calculado por meio de um indicador de carga. E, cada indicador é uma combinação de outros indicadores de carga específicos para aquele recurso.

Os *n* tipos de recursos, fornecidos pela máquina, representam um espaço *n*-dimensional. Assim, os quatro recursos utilizados representam uma 4-tupla de quatro dimensões: <CPU, Disco, Rede, Memória>. Uma aplicação computacionalmente intensiva (do inglês *CPU-bound*), a qual utiliza 100% do processador, é representada pelo vetor <1,0,0,0>. O algoritmo de balanceamento de carga utiliza vetores para representar tanto as tarefas quanto a carga computacional exercida em um sistema. E de modo a descobrir qual sistema computacional seria mais adequado à execução de uma determinada tarefa, é utilizada uma fórmula baseada na distância Euclidiana (11).

$$carga(i) = \sqrt{I_{CPU}^2 + I_{Disco}^2 + I_{Memoria}^2 + I_{Rede}^2} \quad (11)$$

3.1.7 Carga real atual

Li (2009) apresenta um algoritmo de balanceamento de carga dinâmico baseado no princípio do "primeiro a chegar, primeiro a ser atendido" (do inglês *First Come, First Served*), onde a ordem de chegada das tarefas determina a ordem de atendimento das mesmas. O algoritmo possui dois processos: a distribuição de carga e a coleta de informações.

O processo de coleta de informações consiste no cálculo do indicador de carga e possui parâmetros estáticos e dinâmicos. Os parâmetros estáticos são representados pela variável DW(i) e os aspectos dinâmicos pela variável LOAD(i), chamada aqui de carga(i).

$$carga(i) = \begin{Bmatrix} K_1 \times L_{CPU}(i) + K_2 \times L_{mem}(i) + \\ K_3 \times L_{band}(i) + K_4 \times L_{IO}(i) \end{Bmatrix} \quad (12)$$

A carga computacional em cada sistema computacional *i* é calculada pela fórmula (12). O coeficiente K_j especifica a importância de cada parâmetro na composição da carga computacional, similar ao

utilizado pelo indicador de carga de Tong (2009), apresentado na seção 3.1.4. A soma de todos os coeficientes K_j, tal que j={1,..,4}, deve ser 1.

Os parâmetros que compõem o indicador de carga são:

- L_{CPU} é a taxa de ocupação do processador;
- L_{mem} é a taxa de uso da memória;
- L_{band} é a taxa de uso do sistema de E/S;
- L_{IO} é a taxa de ocupação da banda de rede.

Na decisão de balanceamento, o autor ainda considera parâmetros estáticos com pesos, DW(i). Esses parâmetros estáticos representam características de desempenho particulares e imutáveis de cada sistema computacional. Entretanto, em seu artigo (LI, 2009), o autor não apresenta a formulação de DW(i), tampouco descreve a composição desses parâmetros estáticos.

Em relação à uniformidade do sistema (homogêneo ou heterogêneo), deduz-se que o autor tem como intuito descrever um indicador de carga para sistemas homogêneos. Primeiro, porque a descrição e os experimentos realizados pelo autor (LI, 2009) citam somente agregados computacionais (seção 2.1.1). Segundo, porque o parâmetro estático DW(i) não é apresentado, o qual poderia indicar características de sistemas computacionais heterogêneos.

3.2 COMPARAÇÃO ENTRE OS INDICADORES DE CARGA

Esta seção fornece uma visão geral dos indicadores de carga apresentados na seção 3.1 e, de modo que seja possível compará-los entre si, são apresentadas tabelas contendo cada um dos indicadores de carga e os recursos que eles levam em consideração na sua composição. Os indicadores são comparados entre si sob os pontos de vista de processamento, sistema operacional, comunicação e armazenamento. Esses pontos de vistas representam o universo de parâmetros utilizados pelos indicadores de carga, agrupados em categorias, onde todos se encaixam em alguma dessas categorias.

3.2.1 Uniformidade

Tal como averiguado por Ferreira (2015), a grande maioria dos indicadores de carga, presentes na literatura, são adequados somente para sistemas homogêneos. A principal dificuldade para a concepção de

indicadores de carga para sistemas heterogêneos reside na forma de comparação do desempenho potencial entre sistemas computacionais distintos, onde fatores qualitativos, não somente quantitativos, tem influência no desempenho. A Tabela 1 apresenta o tipo de uniformidade de cada um dos indicadores de carga apresentados neste capítulo, indicando se eles são adequados a sistemas computacionais homogêneos ou heterogêneos.

Tabela 1: Indicadores e tipos de sistemas computacionais

Indicador	Uniformidade do Sistema
3.1.1	Heterogêneo
3.1.2	Homogêneo
3.1.3	Homogêneo
3.1.4	Heterogêneo
3.1.5	Heterogêneo
3.1.6	Heterogêneo
3.1.7	Homogêneo

FONTE: O próprio autor

3.2.2 Processamento

A Tabela 2 apresenta uma comparação dos indicadores de carga considerando os recursos de hardware que estão diretamente ligados ao processamento, mais especificamente ao processador e à memória principal.

A coluna intitulada processador é subdividida em velocidade, quantidade de núcleos e taxa de ocupação. A velocidade diz respeito ao poder de processamento, influenciado pela frequência do processador. A quantidade diz respeito ao número de núcleos de processamento, o que influencia na quantidade de tarefas que o processador consegue executar paralelamente. A ocupação representa a taxa de uso do processador, que pode ser uma medida instantânea ou acumulada. A coluna de memória indica quais indicadores de carga levam em consideração a quantidade de memória que está sendo utilizada no sistema em que se deseja medir a carga computacional.

Tabela 2: Recursos ligados ao processamento

Indicador	Processador			Memória
	Velocidade	Quantidade de núcleos	Taxa de ocupação	Quantidade em uso
3.1.1	não	**sim**	**sim**[1]	não
3.1.2	não	**sim**[2]	**sim**[3]	não
3.1.3	não	não	não	**sim**
3.1.4	não	não	**sim**	**sim**
3.1.5	**sim**	**sim**	não	não
3.1.6	**sim**[4]	**sim**[4]	**sim**[4]	**sim**[4]
3.1.7	não	não	**sim**	**sim**

FONTE: O próprio autor

Todos os indicadores de carga apresentados na seção 3.1 levam em consideração o processador ou a memória e isso é explicado pela grande influência que esses dois componentes tem no comportamento da carga computacional (BRANCO, 2006).

3.2.3 Entrada/Saída: comunicação e armazenamento

Em sistemas distribuídos, recursos ligados à comunicação inter processos são tão importantes ao desempenho quanto recursos ligados ao processamento (TONG, 2009). Nesta seção são comparados os indicadores de carga sob o aspecto de comunicação de dados pela rede e do armazenamento de dados em memória secundária.

A coluna intitulada Rede é subdividida em taxa de transmissão e largura de banda. A taxa de transmissão diz respeito à velocidade em que os dados são transmitidos pela rede. A largura de banda diz respeito à

1 O CVFlow leva em consideração tanto o número total de processadores (*cpuCount*) quanto a quantidade deles estão disponíveis (*cpuIdle*).

2 Considerando que cada núcleo de processamento possuí uma fila individual, N representaria cada núcleo.

3 A taxa de ocupação seria uma combinação entre o tempo *s* que a tarefa está usando o recurso processador e o tamanho *q* da fila para utilizar cada núcleo.

4 A construção do indicador I_{CPU} permite a inclusão desse item, mas os autores não definem especificamente a composição do indicador de carga.

vazão dos dados, ou seja, a quantidade de informações que pode ser transmitida simultaneamente pela rede. Os indicadores podem utilizar de forma distinta esses dois conceitos, por exemplo: a largura de banda pode ser representada tanto como um valor absoluto, uma característica intrínseca do hardware; quanto como um percentual que está sendo utilizado, um aspecto referente à forma como o software de gerenciamento da rede funciona.

Tabela 3: Recursos ligados à comunicação e ao armazenamento

Indicador	Rede		Disco
	Taxa de transmissão	Largura de banda	Taxa de transferência
3.1.1	não	não	não
3.1.2	não[5]	não[5]	não[5]
3.1.3	não	não	não
3.1.4	**sim**	não	**sim**
3.1.5	não	não	não
3.1.6	**sim**[6]	**sim**[6]	**sim**[6]
3.1.7	não	**sim**	**sim**

FONTE: O próprio autor

A Tabela 3 demonstra como os dispositivos de E/S são menos utilizados para a composição dos indicadores de carga do que os componentes responsáveis pelo processamento, apresentados na Tabela 2.

3.2.4 Sistema operacional

O sistema operacional é a camada de software responsável pelo controle e gerenciamento dos recursos de hardware e arbitra o uso do hardware pelos processos, fornecendo proteção e acesso compartilhado ao hardware. Além disso, o SO oferece uma camada de abstração para que os processos não tenham que lidar diretamente com o hardware. O

5 Esse indicador poderia representar a taxa de uso da rede e do disco, mas as taxas de transmissão e transferência teriam que ser derivadas a partir do estado das filas de E/S.

6 A construção dos indicadores I_{Disk} e I_{Memory} permite a inclusão desse item, mas os autores não definem especificamente a composição do indicador de carga.

próprio conceito de processo é uma abstração criada pelo sistema operacional (BRYANT, 2011).

Na Tabela 4 são apresentados os componentes, ou abstrações, do sistema operacional que são levados em conta na construção dos indicadores de carga. A quantidade de processos na fila de prontos representa a quantidade de processos disponíveis para execução que competem pelo uso do processador. O tempo em execução diz respeito à quantidade de tempo que o processador dedica àquele processo. Já o tempo para execução representa o tempo que um processo aguarda para ser executado e equivale ao tempo de resposta.

Tabela 4: Recursos ligados ao sistema operacional

Indicador	Processos		
	Quantidade (na fila de prontos)	Tempo em execução	Tempo para execução
3.1.1	**sim**	não	não
3.1.2	**sim**[7]	**sim**[8]	não
3.1.3	**sim**	não	não
3.1.4	**sim**	não	**sim**[9]
3.1.5	**sim**	não	não
3.1.6	**sim**[10]	**sim**[10]	**sim**[10]
3.1.7	não	não	não

FONTE: O próprio autor

A Tabela 4 demonstra que a quantidade de processos aguardando execução é uma informação de grande impacto na carga computacional. Nesse caso, o próprio conceito de carga computacional implica na quantidade de tarefas a serem executadas.

7 Representado por *q*, o tamanho da fila para usar o processador.
8 Tempo *s* que a tarefa usaria o processador.
9 Consideramos equivalente ao tempo de resposta Rt(i).
10 As informações a respeito do estado dos processos podem ser adicionados no indicador I_{CPU}.

4 INTERFACE NATURAL DE USUÁRIO

Este capítulo descreve o conceito de Interface Natural de Usuário (do inglês *Natural User Interface*) e apresenta a ferramenta CVFlow, utilizada nos experimentos desta dissertação como forma de comparação entre o modelo proposto e os modelos existentes. Nessa ferramenta foram implementados o indicador de carga e o algoritmo de balanceamento de carga propostos nesta dissertação.

4.1 CONCEITO

As interfaces entre humanos e computadores evoluíram (Figura 2); as primeiras foram as interfaces por linha de comando, ou do inglês, *Command Line Interface* (CLI), o que exigia a memorização, por parte do usuário, de comandos que instruíam o computador a executar certas tarefas (MANN, 1998).

Figura 2: Evolução das interfaces de usuário
Fonte: O próprio autor

Com o aprimoramento do hardware de exibição de vídeo e a introdução de novos periféricos para interação (como o *mouse*), surgiram as interfaces gráficas, ou *Graphic User Interfaces* (GUI). Esse tipo de interface se caracteriza por exigir uma menor curva de aprendizado para utilizá-las por serem mais intuitivas e amigáveis. As GUI apresentam, de forma visual as metáforas computacionais adotadas nas linhas de comando, como diretórios (pastas) e arquivos. Todavia, a interação ainda segue padrões e metáforas específicos da computação.

No próximo passo da evolução, as interfaces se tornaram ainda mais intuitivas com objetivo reduzir a curva de aprendizado. A ideia central é que a interação com o software seja algo natural, por essa razão surgiu o nome Interface Natural de Usuário, do inglês *Natural User*

Interface (NUI). O significado da palavra natural vem do fato que o usuário interage naturalmente com a interface do software, usando elementos naturais (MANN, 1998).

Enquanto as GUI apresentam graficamente, ao usuário, conceitos específicos da computação, as NUI utilizam conceitos e elementos familiares ao usuário, sem introduzir novos conceitos específicos do software a serem aprendidos pelo usuário. A ideia por trás das NUI é que a interface seja invisível, no sentido de seu uso ser transparente ao usuário.

A Interface Natural de Usuário é uma evolução na forma de interação entre humanos e computadores. Ela se baseia na ideia de que a interface humano computador é invisível, fazendo com que o usuário instrua o computador de uma forma intuitiva (MANN, 1998). A NUI utiliza, para a interação entre usuários e computadores, elementos naturais ao usuário, ao invés de naturais ao computador.

4.2 CATEGORIAS

Dentre as categorias de interfaces naturais de usuário existentes, destacam-se duas:

- Interação com o computador por meio de elementos naturais, como, por exemplo, gestos ao invés de dispositivos de hardware convencionais. Steve Mann (1998 e 2001) cunhou o termo *Reality User Interface* para designar esse tipo de interface. O dispositivo Kinect da Microsoft é um modelo de dispositivo que possibilita o uso de Interfaces Natural de Usuário;
- Interação que não exige que o usuário aprenda os conceitos do software com o qual está interagindo, porque a interface de software é modelada pelo domínio do problema, não pelo domínio da solução (WIGDOR, 2011).

Em ambas as categorias a interface é transparente ao usuário, ou seja: é minimizado ou desnecessário o aprendizado de como interagir com o software.

O CVFlow está enquadrado na segunda categoria, pois, para utilizá-lo, não é exigido do usuário o aprendizado de conceitos ou métodos de balanceamento de carga, o mesmo realiza o balanceamento de carga sem perceber que o está fazendo. Em outras palavras: o balanceamento de carga é feito de forma natural com o CVFlow.

4.3 CVFLOW

Nesta seção é apresentada a ferramenta CVFlow: seu objetivo, descrevendo o problema ao qual ela se propõe a resolver; as principais formas de uso, uma descrição de como a ferramenta é utilizada para resolver o problema ao qual foi projetada; e a arquitetura, uma visão detalhada dos elementos que compõem a ferramenta.

O CVFlow é uma ferramenta de auxílio à execução dos algoritmos de processamento de imagens desenvolvidos pelo *Image Processing and Computer Graphics Lab* (LAPiX) da UFSC e o software permite especificar o fluxo de execução de outras tarefas de computador (WEBER, 2013).

4.3.1 Objetivo

De modo a oferecer um meio de validar resultados apresentados em artigos científicos, principalmente relacionados a algoritmos de processamento de imagem, foi criada a ferramenta CVFlow. E essa ferramenta permite a outros pesquisadores a execução de algoritmos sob seus próprios dados e a comparação de seus resultados com os apresentados nos artigos.

O CVFlow foi escolhido como ferramenta para realização dos experimentos pelas seguintes razões: seu código é aberto e disponível aos mestrando e seu orientador; possuí uma abordagem NUI para realização de balanceamento de carga de forma transparente ao usuário; e os criadores e desenvolvedores da ferramenta são pesquisadores dos laboratórios da UFSC.

4.3.2 Uso

O CVFlow permite agendar a execução de um algoritmo em praticamente qualquer sistema computacional acessível à ferramenta e ordena a uma máquina a execução de um algoritmo, o qual foi previamente compilado para executar nessa máquina. Os algoritmos são representados por programas de computador: para um determinado nó da rede poder executar um algoritmo, é preciso que o programa que ele representa seja compilado para aquela máquina.

Por meio do CVFlow é possível especificar o fluxo de execução de outras aplicações com o intuito de criar um encadeamento de execuções, chamado de *pipeline*. Esse pipeline equivale a uma

ordenação topológica sobre os algoritmos, os quais representam os vértices do grafo de execução. Um dos objetivos do pipeline é distribuir, entre diversas máquinas, a carga computacional decorrente do processamento realizado por determinado algoritmo. A figura 3 ilustra esse encadeamento de algoritmos. No contexto do CVFlow, cada aplicação representa um único algoritmo. Por sua vez, esse algoritmo realiza uma tarefa específica sob um conjunto de entradas, resultando em um conjunto de saídas.

Figura 3: Encadeamento de execuções na interface cliente do CVFlow
Fonte: O próprio autor

A saída de um algoritmo pode ser a entrada de outro algoritmo, tal como no mecanismo de *pipe* do Unix para comunicação inter processos. Assumindo que um algoritmo A_i possui um conjunto de entradas E_i e um conjunto de saídas S_i. Dado um conjunto de *m* máquinas $\{M_1, M_2, \ldots, M_m\}$, executando *n* algoritmos $\{A_1, A_2, ..., A_n\}$. Podemos ter:

$$S_1 = E_2, S_2 = E_3, \ldots, S_{(n-1)} = E_n \tag{13}$$

Outra característica do modelo de execução do CVFlow é a possibilidade de compor algoritmos maiores a partir de menores, o que permite, inclusive, definir fluxos paralelos de execução de algoritmos,

embora paralelização não seja um dos objetivos previstos para a ferramenta.

É possível executar algoritmos locais (residentes na estação que executa a ferramenta) ou remotos (residentes em computadores acessíveis pela rede). Em ambos os casos, o algoritmo deve estar presente no computador que o executará. Para execuções remotas, o sistema computacional que executará o algoritmo – chamado de servidor de execução – precisa ter um endereço de rede acessível à estação executando o cliente CVFlow, além do algoritmo em formato executável.

Outro componente do CVFlow, intitulado servidor de indexação, oferece um serviço de registro de servidores de execução. Nesse servidor de indexação, as máquinas dispostas a executar algoritmos se registraram previamente para disponibilizar o serviço de execução a outras máquinas. O CVFlow pode conectar-se diretamente ao servidor de execução, ou conectar-se indiretamente por meio de um servidor de indexação.

Em relação ao método de conexão remota, o software pode conectar-se aos servidores de execução por meio de *Web Services* ou *Sockets*. Os *Web Services* são apropriados para estações remotas cujo acesso é restringido, ou onde conexões diretamente via *Sockets* não são possíveis.

4.3.3 Arquitetura

Todas as aplicações que compõem a *suite* do CVFlow são escritas na linguagem de programação Java. A *suite* CVFlow é composta pelos seguintes programas:

- **CVFlow**. Esta é a aplicação cliente que possuí uma interface gráfica para interação com o usuário. Por meio do programa CVFlow, o usuário pode especificar o fluxo de execução de algoritmos;
- **CVFlowServer**. Esse é o servidor de execução, uma aplicação que permite executar outras aplicações em um sistema computacional. O CVFlowServer reside na máquina onde o algoritmo será executado. O algoritmo deve estar presente (na forma de um arquivo executável) na mesma máquina que contém o CVFlowServer;

- **CVFlowIndexServer**. A aplicação na qual outros servidores de execução podem se registrar.

A ferramenta CVFlow permite executar algoritmos local ou remotamente. Para suportar isso, a ferramenta possui dois tipos de objetos: o ItemAlgoritmo para representar algoritmos locais e o ItemAlgoritmoRemoto para representar algoritmos remotos. O primeiro permite ao CVFlow executar aplicações residentes no mesmo sistema computacional rodando o CVFlow. Já o segundo componente oferece a possibilidade de executar aplicações residentes em um outro sistema computacional interconectado ao software via rede.

No CVFlow a escolha do sistema computacional a ser utilizado para executar uma tarefa não considera as características arquiteturalmente heterogêneas que afetam o desempenho da execução.

O algoritmo é representado por um arquivo binário executável, um programa. Este pode ser escrito em qualquer linguagem, desde que seja executável na plataforma em que reside e atenda ao padrão de entradas e saídas especificadas pelo CVFlow. Entre os requisitos de compatibilidade de um programa com o software, estão um XML que descreve o programa e é lido pelo componente CVFlowServer. Para executar algoritmos remotos é necessário que o sistema computacional remoto, além do ser alcançável via rede, possua uma instância do servidor de execução. Cada algoritmo executado pelo CVFlow possui um conjunto de entradas e de saídas.

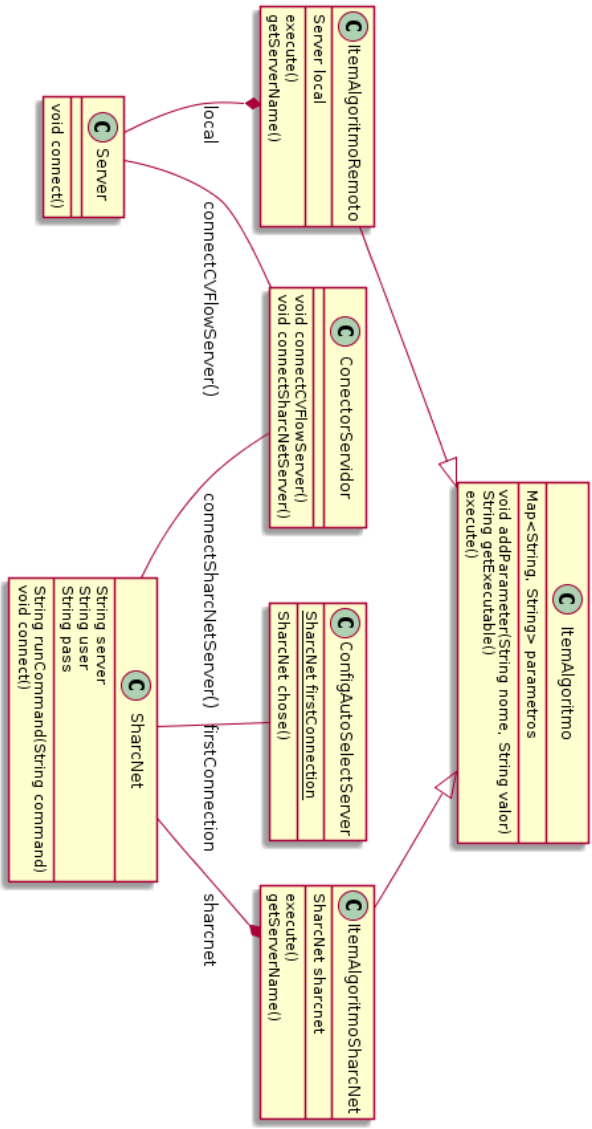

Figura 4: Diagrama de classes do CVFlow
Fonte: O próprio autor

5 PROPOSTA

A proposta desta dissertação é fornecer um modelo de previsão de desempenho baseado na avaliação da carga computacional, que leva em consideração as características dos sistemas computacionais distribuídos heterogêneos em um determinado instante. Essas características tem impacto no desempenho da tarefa a ser executada. Entre essas características estão recursos de hardware e de software, como memória principal e tarefas em execução, por exemplo.

O objetivo desta pesquisa pode ser classificado como um método de auxílio à política de informação do balanceamento de carga, responsável por especificar quais informações da carga computacional são relevantes para o balanceador de carga tomar suas decisões.

Diferente da abordagem de Silva (2011), onde a seleção dos recursos computacionais é feita com base nas características das tarefas, traduzidas para perfis estáticos dos recursos computacionais, esta pesquisa propõe um perfil de recurso computacional calculado por meio de medidas dinâmicas, com intuito de prever o comportamento de um sistema.

Esta pesquisa complementa aquela apresentada por Silva (2011), o qual apresenta um modelo de escalonamento intrinsecamente estático e esse modelo de escalonamento assume conhecimento prévio acerca das características do sistema computacional, sem levar em conta suas mudanças ao longo do tempo. O modelo proposto nesta pesquisa tem como intuito utilizar métricas obtidas em tempo real, fornecendo uma previsão mais fiel para a execução da tarefa.

O perfil de recurso computacional é fornecido para uma política de escalonamento. Por sua vez, esta política é fundamentada em uma caracterização mista de carga de trabalho e dos recursos computacionais disponíveis no momento que for solicitado o recurso. Algumas medidas como percentual de utilização dos processadores e de memória são triviais (BRANCO, 2006). Contudo, outras menos comuns, como arquitetura do hardware, serão levadas em consideração. Essa política de escalonamento possui métricas e algoritmos. As métricas servem para dar maior ou menor importância à determinadas características do sistema.

5.1 CONTRIBUIÇÕES

A principal contribuição desta dissertação é um indicador de carga – um modelo de caracterização de recursos computacionais – que forneça uma aproximação mais precisa com o desempenho potencial de sistemas computacionais distribuídos arquiteturalmente heterogêneos. Esse indicador de carga tem o intuito de melhorar a escolha do sistema computacional para executar uma determinada tarefa.

5.2 ALGORITMO DE BALANCEAMENTO DE CARGA

Tal como explicado na seção 2.2, o balanceamento de carga é uma forma de escalonamento onde o escalonador decide quais, onde, quando e por quanto tempo as tarefas serão executadas (STALLINGS, 2011). Como o desenvolvimento do algoritmo de escalonamento não é o objetivo desta pesquisa, é implementado um algoritmo simples baseado em pesos (COULOURIS, 2012). Esse algoritmo consiste na escolha do sistema computacional mais apto a executar uma determina tarefa onde a aptidão de um sistema computacional é dada pelo indicador de carga, o qual representa o “peso” desse sistema.

5.2.1 Descrição

O algoritmo para a escolha do sistema computacional, cuja tarefa será escalonada, é dado pelo pseudo código ilustrado na figura 5. O algoritmo consiste em percorrer uma lista de *m* servidores para cada uma das *n* tarefas, analisando o indicador de carga de cada servidor.

```
para cada tarefa T na fila de prontos
 para cada máquina M disponível
  ler o indicador de carga da máquina
  se este for o menor indicador de carga até agora, então
   colocar a tarefa T para execução na máquina M
  atualizar o indicador de carga
```

Figura 5: Algoritmo de balanceamento de carga

A função de balanceamento de carga *b(x)*, pela qual é escolhido o menor indicador de carga dentre todos os disponíveis, é dada pela fórmula (14). O algoritmo de balanceamento de carga escolhe o menor indicador de carga porque este representa o nó menos sobrecarregado,

ou seja, o nó com mais capacidade de executar uma determinada tarefa.

$$b(x) = min(carga_M(1), carga_M(2), \ldots, carga_M(m)) \quad (14)$$

O algoritmo na figura 5 adota uma estratégia de balanceamento de carga baseada em pesos (do inglês *Weighted Load Balancing*). Existem outros algoritmos de balanceamento de carga, como *Round-Robin*, *Least-Connection*, *Weighted Least-Connection*, *Min-Max* (RAJ, 2013; TONG, 2009).

5.2.2 Limitações

O algoritmo não é preemptivo, ou seja, se uma determinada tarefa é encaminhada a um nó, essa tarefa será executada até o final (STALLINGS, 2011). Também não é possível a migração de uma tarefa em execução de um nó para outro.

5.2.3 Análise de complexidade

Nesta seção é analisado o tempo de execução do algoritmo de balanceamento de carga. Essa análise tem como objetivo demonstrar a viabilidade da execução desse algoritmo. Para fins de simplificação da análise, assume-se que o algoritmo será executado em um computador simples, com um único processador, memória RAM infinita e execução sequencial de instruções. Como entrada para o algoritmo, temos *n* tarefas na fila de prontos e *m* servidores capazes de executar cada uma dessas tarefas. Assim, o tempo médio de execução é dado pela fórmula (15).

$$T(n) = O(m * n) \quad (15)$$

Na escolha do sistema computacional mais adequado para executar uma tarefa, leva-se em consideração na decisão somente o indicador de carga. As características da aplicação, por meio da sua caracterização de carga, não são utilizadas para decidir-se o sistema computacional destino. Tal como descrito por Cormen (2009), o escalonamento ótimo de *n* algoritmos em *m* máquinas é um problema NP-difícil, similar ao *bin packing*. Dessa forma, o algoritmo de balanceamento de carga implementado pelo CVFlow utiliza a heurística do primeiro apto (do inglês *first-fit*), um algoritmo guloso como forma

de obter a solução ótima local. Nesta pesquisa não temos como intuito o aprimoramento do algoritmo de balanceamento de carga propriamente dito.

5.3 INDICADOR DE CARGA

O indicador de carga proposto nesta dissertação é baseado naquele apresentado por Tong (2009) onde há um indicador de carga para cada elemento processador do sistema distribuído, cujo cálculo leva em conta o indicador para cada processador desse nó. Dessa forma, $carga_M$ representa o indicador de carga da máquina, enquanto que $carga_P$ representa o indicador de carga de cada processador dessa máquina.

5.3.1 Indicador de carga do processador ($carga_P$)

Existe um indicador de carga para cada processador. Em cada processador P_i, a carga dinâmica é medida por meio da fórmula (16).

$$carga_P(i) = \pi_1 * taxa_{CPU}(i) + \pi_2 * qtde_{Mem}(i) + \pi_3 * qtde_{Proc}(i) \quad (16)$$

O símbolo π_i é o grau de importância, ou peso, de cada parâmetro que compõe o indicador de carga. Esse indicador de carga leva em consideração a taxa de uso do processador ($taxa_{CPU}$), a quantidade de memória em uso ($qtde_{Mem}$) e a quantidade de processos na fila de execução ($qtde_{Proc}$). Cada um desses parâmetros é descrito nas seções a seguir.

5.3.1.1 Taxa relativa de uso do processador ($taxa_{CPU}$)

A taxa de uso do processador é um dos atributos mais relevantes (senão o mais) para a análise da carga em um determinado sistema computacional e esse atributo está intrinsecamente ligado a cada processador específico dentro de um nó.

A taxa de uso serve para avaliar a capacidade de processamento de um processador de uma máquina em um determinado momento, enquanto que a quantidade de processos na fila de prontos fornece uma ideia da quantidade de tempo que o processador manterá essa taxa de processamento. Por exemplo, uma taxa de processamento alta, mas com poucos processos a serem executados, pode ser mais favorável do que

uma taxa mediana, mas com uma quantidade maior de processos que manterão essa taxa de uso do processador.

Como o objetivo desse indicador é informar a carga em sistemas heterogêneos, a taxa de uso do processador deve ser relativa ao poder de processamento de todos os processadores de todos os nós. Algo similar ao adotado pelo indicador descrito por Bosque (2013). Com a diferença que não é necessário conhecer o sistema computacional com maior poder de processamento para normalização do indicador de carga.

5.3.1.2 Quantidade relativa de memória em uso ($qtde_{Mem}$)

Em sistemas distribuídos heterogêneos, onde sistemas distintos possuem quantidades variáveis de memória, utilizar a taxa de ocupação (ou uso) da memória não é um indicativo confiável da carga em relação aos outros sistemas computacionais. Ao avaliar a capacidade de um sistema em aceitar novas tarefas, deve-se levar em conta a relação entre a quantidade de memória em uso e o total de memória que esse sistema possui. Em outras palavras, o mais importante é a quantidade de memória disponível para outros processos.

Ao colocar a quantidade de memória em uso como atributo do indicador de carga do processador – ao invés do nó como um todo – permite-se que o mesmo indicador seja utilizado tanto em sistemas computacionais com memória compartilhada entre os núcleos de processamento quanto em sistema em que cada núcleo de processamento possui sua memória privada (tal como em sistemas com acesso não-uniforme à memória).

Para que esse arranjo seja utilizado em sistemas computacionais com memória compartilhada entre os processadores (como sistemas com multiprocessamento simétrico), basta atribuir o mesmo valor de memória a todos os processadores; dessa forma, o valor desse parâmetro não influencia de forma diferente processadores distintos.

5.3.1.3 Quantidade de processos na fila de prontos ($qtde_{Proc}$)

Esse atributo depende da forma como os processos são escalonados pelo sistema operacional. Alguns SOs mantêm uma fila individual de processos para cada processador enquanto outros adotam uma fila compartilhada entre os processadores (STALLINGS, 2011).

Da mesma forma como o indicador de carga desta dissertação é

flexível em relação ao parâmetro de quantidade de memória em uso, o indicador também permite qualquer um dos arranjos na fila de prontos mencionados anteriormente.

5.3.1.4 Grau de importância dos parâmetros (π_i)

O grau de importância define o peso que cada atributo de hardware terá sobre o indicador de carga. Branco (2006) propôs em seu indicador construir um vetor de graus de importância para representar diferentes configurações de hardware para diferentes tipos de aplicações.

$$Perfil = \langle \pi_1, \pi_2, \pi_3 \rangle \tag{17}$$

Para aplicações que, durante sua execução exigem mais processador, o parâmetro π_1 daria mais peso ao processador. Dessa forma, um aumento na taxa de uso do processador em uma determinada máquina, teria maior peso e desencorajaria o uso daquele sistema computacional em detrimento de outros sistemas com um menor aumento da taxa de uso, mesmo que os outros parâmetros tenham aumentado.

5.3.2 Indicador de carga da máquina ($carga_M$)

Para cada máquina M_j, a qual é composta por *p* processadores (ou núcleos de processamento), a carga é dada por meio da fórmula (18):

$$carga_M(j) = \alpha_j * \left(\frac{\sum_{i=1}^{p} carga_P(i)}{p} \right) + consumo_M(j) \tag{18}$$

Onde α_j indica o grau de desempenho da arquitetura da máquina M_j e *p* é a quantidade de processadores do sistema computacional. O grau de desempenho deve ser informado pelo usuário, uma vez que produzir esse tipo de comparação entre arquiteturas está fora do escopo desta pesquisa.

A segunda parte do indicador de carga ($carga_M$) realiza uma média

aritmética da carga de todos os processadores do sistema computacional. Assumimos que todos os sistemas computacionais possuem multiprocessamento simétrico (SMP), onde os processadores são homogêneos entre si. Ou seja, todos os processadores da máquina possuem exatamente o mesmo perfil, mudando somente na capacidade momentânea. A heterogeneidade tratada nesta dissertação é no nível do sistema distribuído, não no nível de um único sistema computacional individualmente.

5.3.2.1 Grau de desempenho da arquitetura (α_j)

O grau de desempenho da arquitetura tem como objetivo normalizar o desempenho dos sistemas computacionais com base em suas arquiteturas. Sistemas computacionais com recursos de hardware similares, mas diferentes arquiteturas, devem ter graus de desempenho distintos. A relação entre os parâmetros P_i e P_{max} do indicador de carga apresentado por Bosque (2013), possuí o mesmo propósito do grau de desempenho da arquitetura aqui apresentado. Todavia, Bosque não tem o intuito de diferenciar o desempenho de arquiteturas de hardware.

$$\alpha_j = \left\{ \frac{1 * 10^{fu}}{P_j} \right\} \simeq \left\{ \frac{P_{max}}{P_j} \right\} \tag{19}$$

Outro inconveniente da abordagem de Bosque (BOSQUE, 2013) é a necessidade de utilizar um P_{max} como referência para a normalização de todos os sistemas computacionais. Para descobrir qual o sistema detém o maior poder computacional (P_{max}), utilizando algoritmos de ordenação por comparação, são necessárias pelo menos $\Omega(n \lg n)$ operações (CORMEN, 2009). O grau de desempenho da arquitetura (19) dispensa a necessidade de comparação com todos os sistemas computacionais, assumindo que seu valor é inversamente proporcional ao poder de processamento do sistema computacional (P_j). O *fu* é a unidade de FLOPS comum a todo o sistema distribuído e 10^{fu} deve substituir P_{max} como valor normalizador. Os possíveis valores de *fu* são apresentados na tabela 5.

Tabela 5: Valores e unidades de FLOPS

Nome	Abreviação	FLOPS	*fu*
kiloFLOPS	kFLOPS	10^3	3
megaFLOPS	MFLOPS	10^6	6
gigaFLOPS	GFLOPS	10^9	9
teraFLOPS	TFLOPS	10^{12}	12
petaFLOPS	PFLOPS	10^{15}	15
exaFLOPS	EFLOPS	10^{18}	18
zettaFLOPS	ZFLOPS	10^{21}	21
yottaFLOPS	YFLOPS	10^{24}	24

FONTE: wikipedia.org

5.3.2.2 Consumo de energia elétrica da máquina ($consumo_M$)

Uma vez calculado individualmente o indicador de carga de cada processador, são levados em conta fatores comuns a todos os processadores do nó, tal como consumo de energia elétrica. Nesta pesquisa é considerado o consumo como parte da composição do indicador; entretanto, não foi realizado uma pesquisa para derivar esse valor.

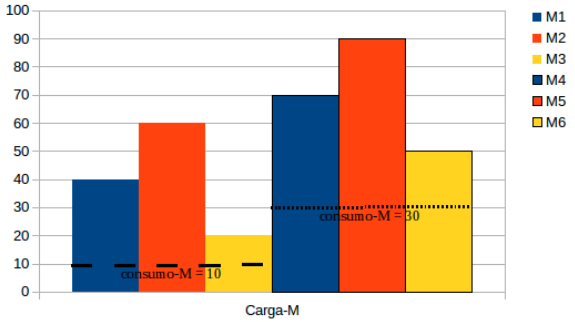

Figura 6: Exemplo de consumo deslocando o indicador de carga

O objetivo deste parâmetro é interferir na probabilidade de um sistema computacional ser escolhido pelo algoritmo de balanceamento de carga, aumentando ou diminuindo o valor do indicador de carga. É possível penalizar ou incentivar o uso de um conjunto de sistemas computacionais, por meio de um valor alto ou baixo de consumo, respectivamente. A figura 6 ilustra um caso onde os sistemas computacionais M_1, M_2 e M_3 possuem um consumo de 10 enquanto que os sistemas computacionais M_4, M_5 e M_6 possuem um consumo três vezes maior. Nesta proposta, não é atribuído nenhum sentido específico ao parâmetro de consumo, então, dessa forma, essa proporção pode ser dada pelo custo da energia elétrica em cada conjunto de sistemas, por exemplo.

6 EXPERIMENTOS

De modo a validar a proposta apresentada no capítulo anterior, neste capítulo são apresentados os experimentos realizados utilizando uma versão modificada da ferramenta CVFlow.

O objetivo dos experimentos é detalhado na seção 6.1. A seção 6.2 descreve o ambiente utilizado para a execução dos experimentos. Na seção 6.3 é descrito o procedimento para realização dos experimentos. E por fim, na seção 6.4 são apresentados os resultados dos experimentos. O capitulo 7 apresenta uma discussão mais aprofundada a respeito dos resultados dos experimentos.

6.1 OBJETIVO

O objetivo dos experimentos é avaliar se o indicador de carga, proposto nesta dissertação, fornece uma previsão de desempenho mais precisa que os indicadores de carga no estado da arte, apresentados nos trabalhos relacionados. Tal como representado pela fórmula (1), o valor do indicador de carga, em um determinado sistema computacional, deve manter a propriedade de proporcionalidade em relação à carga exercida nesse sistema. E o desempenho de um sistema computacional, em um determinado instante, deve ser inversamente proporcional à carga exercida nesse sistema naquele mesmo instante.

Nos experimentos é analisado o comportamento de um conjunto de tarefas em cada um dos sistemas computacionais, assegurando que o indicador de carga proposto induz o algoritmo de balanceamento de carga do CVFlow a escolher o sistema computacional que executa a tarefa no menor tempo. E é realizada uma comparação com os valores dos indicadores de carga apresentados no capítulo 3.

Neste capítulo, os indicadores de carga são referenciados pela numeração, não por seus autores ou nomes. Essa decisão tem como intuito facilitar a leitura. Os indicadores de carga 3.1.2 e 3.1.6 não são avaliados nos experimentos. O indicador de carga 3.1.2 não é considerado porque possuí um perfil estritamente voltado a sistemas computacionais homogêneos. O indicador 3.1.6 não é avaliado porque o seu autor (BRANCO, 2006) não fornece um indicador de carga específico, mas uma plataforma para adicionar outros indicadores de carga.

6.1.1 Hipótese

O objetivo dos experimentos pode ser dividido na avaliação de duas relações: a variação entre o valor do indicador de carga e a carga computacional; e a variação da carga computacional em relação ao desempenho na execução de tarefas. A primeira relação avalia a hipótese de que o valor do indicador de carga varia proporcionalmente à carga computacional. Já a segunda relação avalia a hipótese de que há uma proporcionalidade entre a carga computacional exercida em um sistema (representada pelo valor do indicador de carga) e o tempo de execução de uma tarefa nesse sistema. Os objetivos dos experimentos são medidos por meio de duas variáveis (WAZLAWICK, 2008), o indicador de carga e o tempo de execução.

6.1.1.1 Indicador de carga

A primeira dependência é a relação entre os parâmetros que compõem o indicador de carga e o valor do indicador de carga. Então, a hipótese é averiguar se o indicador de carga varia proporcionalmente em relação às informações de carga. O indicador de carga é uma variável discreta no conjunto dos números naturais, medida e cuja função de dependência é a equação do indicador de carga. Essa variável é influenciada pelas seguintes variáveis:

- A **taxa de processamento** é uma variável discreta, manipula e independente;
- A **quantidade de memória principal em uso** é uma variável discreta, manipula e independente;
- A **quantidade de processos** é uma variável discreta, manipula e independente;
- O **coeficiente de desempenho da arquitetura** é uma variável discreta, manipula e independente;
- O **consumo de energia** é uma variável discreta, manipula e independente.

6.1.1.2 Tempo de execução

A segunda dependência é representada pela relação entre o tempo de execução de uma tarefa e o valor do indicador de carga. A hipótese baseia-se na existência de uma proporcionalidade entre o valor do

indicador de carga e o tempo de execução de uma tarefa. Essa hipótese depende de duas variáveis:

- O **indicador de carga** é uma variável discreta no conjunto dos números naturais, medida e dependente da carga computacional;
- O **tempo médio de execução da tarefa** é uma variável discreta no conjunto dos números naturais, medida e indiretamente dependente do indicador de carga. Tal como descrito pela propriedade (2), um menor valor para o indicador implica em um menor tempo de execução.

6.2 AMBIENTE

O objetivo desta pesquisa é o desenvolvimento de um indicador de carga para sistema computacionais arquiteturalmente heterogêneos. Dessa forma, para os experimentos, adotou-se um conjunto de sistemas computacionais com características distintas. Esses sistemas variam em quantidade de núcleos de processamento, quantidade de memória principal e arquitetura.

Nesta seção são descritos: a forma como o conjunto de sistema computacionais estão organizados, para compor um sistema distribuído (seção 6.2.1); as características particulares de cada sistema computacional (seção 6.2.2); as informações da carga computacional em cada um desses sistemas (seção 6.2.3); e as constantes específicas de cada indicador de carga avaliado (seção 6.2.4).

6.2.1 Sistema distribuído

O ambiente experimental é composto por um cliente e um conjunto de servidores. O cliente, executando uma instância do CVFlow, é responsável pelo escalonamento das tarefas nos servidores. Os servidores, cada qual executando uma instância do CVFlowServer, são responsáveis pela execução das tarefas. O CVFlow realiza o balanceamento de carga de forma transparente ao usuário. A topologia de comunicação entre o CVFlow e seus servidores CVFlowServer é ilustrada na figura 7. Cada um dos sistemas computacionais informa, periodicamente, ao CVFlow a sua carga atual por meio do indicador de carga. Ao submeter cada tarefa, o cliente CVFlow escolhe uma dentre as *m* máquinas conectadas a ele. No momento em que a tarefa é submetida, o CVFlow requisita sua execução no servidor que apresente o menor

indicador de carga.

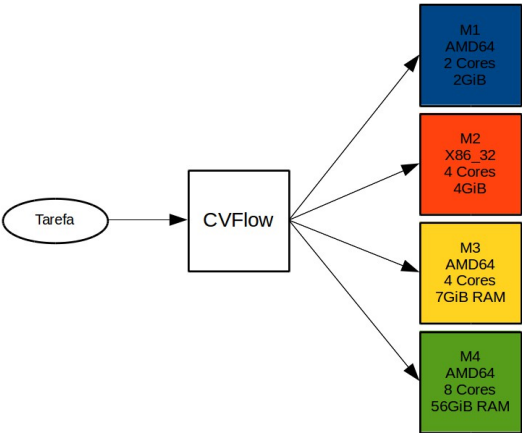

Figura 7: Topologia de escalonamento com CVFlow
Fonte: O próprio autor

Esse ambiente tem o intuito de simular um sistema distribuído com arquitetura heterogênea, como, por exemplo, uma grade computacional. Embora se apresente em uma escala menor que grades convencionais, os conceitos e experimentos podem ser aplicados a sistemas distribuídos de qualquer tamanho.

6.2.2 Sistemas computacionais

O primeiro sistema computacional, intitulado máquina 1 (M_1), é um sistema virtualizado de arquitetura AMD de 64 bits com dois núcleos de processamento. O segundo, intitulado máquina 2 (M_2), é um não-virtualizado de arquitetura Intel x86 de 32 bits e quatro núcleos de processamento. Os sistemas computacionais M_3 e M_4 são virtualizados e possuem arquitetura AMD de 64 bits com quatro e oito núcleos de processamento, respectivamente.

A Tabela 6 apresenta as características estáticas dos sistemas computacionais utilizados nos experimentos. Em ordem, as colunas dessa tabela representam: o nome de identificação do sistema computacional; a arquitetura, a qual implica em um conjunto de instruções particular; se o sistema computacional é virtualizado ou não; o sistema operacional utilizado; a quantidade de núcleos de

processamento disponíveis no total; e a quantidade de memória total, medida em megabytes (MiB) e cujo valor é obtido por meio da ferramenta *free(1)* do Unix.

Tabela 6: Perfil das máquinas utilizadas nos experimentos

Sistema computacional	Arquitetura	Virtualizado	Sistema operacional	Processador	Memória
				Quantidade de núcleos	Quantidade total (MiB)
M_1	AMD64 64 bits	sim	Linux Ubuntu 14.04	2	2048
M_2	IA32 32 bits	não	Linux Ubuntu 10.04	4	4096
M_3	AMD64 64 bits	sim	Linux Ubuntu 14.04 LTS	4	4096
M_4	AMD64 64 bits	sim	Linux Ubuntu 14.04	8	57344

FONTE: O próprio autor

Esse conjunto de sistemas foi escolhido de modo que seus elementos representam as características que se deseja avaliar no experimentos: diferenças quantitativas e qualitativas computadas pelos indicadores de carga.

Os sistemas computacionais M_2 e M_3 são idênticos em suas características quantitativas, como, por exemplo, mesma quantidade de núcleos de processamento e memória. Entretanto, esses sistemas computacionais representam as diferentes características qualitativas que se deseja avaliar: arquitetura e virtualização.

Já os sistemas computacionais M_1 e M_4 apresentam contrastes de características quantitativas. O sistema computacional M_1 representa o elemento com menor poder computacional, enquanto M_4 é o sistema que dispõe de mais capacidade dentre os elementos disponíveis.

6.2.3 Carga computacional

Os sistemas computacionais utilizados nos experimentos partem de um estado ocioso. Esse estado de ociosidade significa que não existem tarefas em execução submetidas pelo usuário. Entretanto, o próprio sistema operacional e seus programas básicos exercem carga ao

sistema computacional (STALLINGS, 2011). A carga computacional inicial é avaliada sob os mesmos aspectos utilizados para comparação dos indicadores de carga na seção 3.2.

6.2.3.1 Processador

A tabela 7 apresenta os valores iniciais da carga computacional relacionados ao processador: o poder computacional, medido por meio do mesmo *benchmark* utilizado por Bosque (2013); a quantidade de núcleos de processamento livres; e a taxa de uso do processador.

A velocidade de cada processador, conhecida também de poder computacional ou poder de processamento (STALLINGS, 2010), é medida em milhões de operações de ponto flutuante por segundo (MFLOPS). Para mensurar a velocidade de cada sistema computacional, foi utilizado o *NAS Parallel Benchmarks* da NASA (BAILEY, 1994). O NAS é um conjunto de ferramentas para avaliação de desempenho em sistemas computacionais multiprocessados. Essa ferramenta de avaliação de desempenho é a mesma utilizada por Bosque (2013) e o intuito de utilizar a mesma ferramenta nos experimentos é manter a fidelidade dos resultados com aqueles obtidos por esse autor.

A quantidade de núcleos livres é limitada pela quantidade de núcleos disponíveis no total. Para fins deste experimento, assume-se que cada sistema computacional pode executar, paralelamente, uma quantidade de tarefas igual ou inferior à quantidade de núcleos de processamento. Dessa forma, a quantidade de núcleos livres representa a quantidade de tarefas que cada sistema computacional pode executar simultaneamente em um determinado instante.

A taxa de ocupação do processador é o percentual de uso de todos os núcleos de processamento em um sistema computacional. A ferramenta *top(1)* do Unix é utilizada para medir o percentual de uso de todos os núcleos de processamento. Para as tarefas computacionalmente intensivas utilizadas nos experimentos (as quais ocupam 100% do núcleo de processamento no qual estão executando), a taxa de ocupação é uma relação entre a quantidade de processos executando e a quantidade de núcleos de processamento disponíveis no total. Por exemplo, em um sistema computacional com quatro núcleos, ao executar uma única tarefa computacionalmente intensiva, a ocupação registrada pelo *top(1)* é de 25%. Ao executar duas tarefas, a taxa de ocupação sobe para 50% e assim por diante. Uma taxa de uso de 100%,

nesse sistema, só é atingida ao executar quatro tarefas computacionalmente intensivas.

Tabela 7: Carga inicial exercida no processador

Sistema computacional	Processador		
	Velocidade (MFLOPS)	Quantidade de núcleos livres	Taxa de ocupação (%)
M_1	1347,45	2	0
M_2	1653,04	4	0
M_3	1397,96	4	0
M_4	1411,33	8	0

FONTE: O próprio autor

6.2.3.2 Memória principal

A tabela 8 apresenta os valores iniciais relacionados à memória principal, para a composição dos indicadores de carga utilizados nos experimentos. A quantidade de memória em uso é medida em megabytes (MiB) e representa o valor médio estimado do consumo de memória do sistema operacional dentre todas as máquinas, utilizando a ferramenta *free(1)* do Unix. O sistema operacional de cada máquina reserva uma quantidade de memória diferente para *buffers* e *caches*, proporcionais à quantidade total de memória. Dessa forma, a quantidade de memória em uso informada pelo *free(1)* não representa a quantidade de memória efetivamente ocupada pelos processos (STALLINGS, 2011). A quantidade de memória em uso foi estimada com base no perfil dos processos carregados no sistema operacional.

O último campo da tabela 8 é uma relação entre a quantidade de memória em uso e a quantidade de memória total. A taxa de ocupação é a representação percentual estimada da quantidade de memória em uso. Esse valor representa qual o percentual de memória principal está sendo efetivamente ocupado por outros processos.

Tabela 8: Carga inicial exercida na memória

Sistema computacional	Memória	
	Quantidade em uso (MiB)	Taxa de ocupação (%)
M_1	500	25,0
M_2	500	12,0
M_3	500	12,0
M_4	500	0,9

FONTE: O próprio autor

6.2.3.3 Sistema operacional

A tabela 9 apresenta os parâmetros iniciais relacionados ao sistema operacional. Tanto a quantidade de processos na fila de prontos quanto a quantidade de processos executando, são obtidas por meio da ferramenta *top(1)*. O total de processos é a soma desses dois valores.

Tabela 9: Carga inicial do sistema operacional

Sistema computacional	Processos / Tarefas	
	Quantidade na fila de prontos	Quantidade em execução
M_1	250	0
M_2	250	0
M_3	250	0
M_4	250	0

FONTE: O próprio autor

6.2.4 Indicadores de carga

Os indicadores de carga apresentados nas seções 3.1.4, 3.1.7 e 5.3, utilizam coeficientes para modificar a importância que os componentes de hardware e de software possuem na composição do indicador de carga. Os valores desses coeficientes (ou graus de

importância) são apresentados na tabela 10 e foram estimados com base no perfil das tarefas do conjunto T (seção 6.3.1). A primeira e a segunda coluna apresentam, respectivamente, o significado e valor de cada coeficiente e as demais apresentam o símbolo correspondente ao coeficiente em cada indicador de carga.

Tabela 10: Grau de importância dos componentes

Componente	Valor	3.1.4	3.1.7	5.3
Processador	0,6	π_1	K_1	π_1
Memória	0,1	π_2	K_2	π_2
Rede	0	π_3	K_3	-
Disco	0	π_4	K_4	-
Tempo de resposta	0	π_5	-	-
Processos	0,3	π_6	-	π_3

FONTE: O próprio autor

Enquanto alguns autores (TONG, 2009; LI, 2009) atribuem coeficientes para alterar individualmente a importância dos parâmetros do indicador de carga; os quais representam componentes de hardware e de software que influenciam na carga computacional. Outros autores (BOSQUE, 2013) atribuem peso ao sistema computacional como um todo. O grau de importância dos sistemas computacionais utilizados nos experimentos são apresentados na tabela 11.

Tabela 11: Grau de importância dos sistemas computacionais

Sistema computacional	3.1.5 (P_{max}/P_i)	5.3 (α_i)
M_1	1,23	0,74
M_2	1,00	0,60
M_3	1,18	0,72
M_4	1,17	0,71

FONTE: O próprio autor

Para o indicador de carga proposto nesta dissertação, os valores 0,7 representam os sistemas computacionais virtualizados, enquanto que o valor 0,6 representa o sistema computacional com arquitetura não virtualizada. Esses valores são o grau de desempenho da arquitetura, apresentado na seção 5.3.2.1.

6.3 PROCEDIMENTO

O experimento consiste no escalonamento sequencial de um conjunto de tarefas. O sistema computacional, para onde cada tarefa é escalonada, é decidido segundo o valor do indicador de carga proposto. Assume-se que o sistema computacional que possuí o menor valor de indicador de carga deve executar a tarefa no menor tempo (propriedade (2) e hipótese 6.1.1.2). Para certificar-se disso, cada tarefa escalonada é executada em todos os sistemas computacionais, de modo a garantir que o valor do indicador de carga se mantém proporcional ao desempenho medido da execução. Ou seja, é medido o tempo médio de execução de cada tarefa em cada sistema computacional.

6.3.1 Conjunto de tarefas

É submetido ao escalonador o conjunto $T=\{T_1, T_2, T_3, \ldots, T_9\}$ composto por 9 tarefas computacionalmente intensivas. Onde a quantidade de elementos do conjunto T foi escolhida de modo que seja possível avaliar o comportamento do indicador de carga no caso das 8 primeiras tarefas serem escalonadas ao sistema computacional M_4. Esse sistema computacional possuí 8 núcleos de processamento, a maior quantidade dentre os sistemas computacionais usados nos experimentos.

Todas as tarefas T_i são idênticas, cada uma exigindo a mesma quantidade de recursos computacionais para serem executadas. Nenhuma das tarefas encerra sua execução antes do final do escalonamento de todas as tarefas. Ou seja, neste experimento, o tempo total para o escalonamento de todas as tarefas (função *b(x)* apresentadas na seção 5.2) é menor que o tempo de execução de uma única tarefa, tal como expressado pela propriedade (20).

$$\forall T_i \in T : \sum_{i=1}^{|T|} tempo_{exec}(b(T_i)) \leq tempo_{exec}(T_i) \qquad (20)$$

A razão de manter essa propriedade, onde nenhuma tarefa pode encerrar antes do escalonamento de todas as tarefas, é para assegurar que os estados de carga sejam incrementais. Se for permitido a uma tarefa encerrar antes do final do escalonamento de todas as outras, há chance de haver estados de carga repetidos. Por exemplo, se M_2 estiver executando T_1 no momento de submeter T_2, a carga de M_2 é calculada sob 1 processo. Se, no instante de submeter T_3, M_2 tiver encerrado a execução de T_1, a carga de M_2 tem novamente 1 processo, assumindo que T_2 tenha sido escalonada à M_2.

6.3.2 Execução das tarefas

De modo a comparar o tempo de execução, a cada etapa do balanceamento, todas as tarefas são executadas em todos os sistemas computacionais. O tempo de execução das tarefas é medido utilizando a ferramenta *time(1)* do Unix. Essa ferramenta exibe três valores: o tempo real decorrido na execução da tarefa; o tempo de usuário, que é o somatório do tempo gasto em cada processador e que para sistemas multiprocessados é maior que o tempo real; e o tempo de sistema, que é o tempo gasto pelo sistema operacional executando a tarefa (KERRISK, 2010).

O valor exibido nos gráficos é referente ao tempo real. Cada tarefa é executada 100 vezes à cada etapa, valor em que é alcançado um nível de confiança de 95% com uma margem de erro inferior à 0,001 (WAZLAWICK, 2008).

6.4 RESULTADO

Esta seção apresenta o resultado do escalonamento do conjunto de tarefas T, seguindo as indicações do indicador de carga proposto nesta dissertação. Para cada tarefa T_i, é apresentada uma comparação entre: os valores dos indicadores de carga, referentes a cada um dos sistemas computacionais, relacionados à hipótese 6.1.1.1; e o tempo médio de execução das tarefas em cada sistema computacional, referente à hipótese 6.1.1.2.

6.4.1 Tarefa 1 (T_1)

Com base na figura 8 e na tabela 12, é possível verificar que o menor valor para o indicador de carga proposto, antes de submeter a primeira tarefa (T_1), é referente ao sistema computacional M_2. Os valores absolutos, entre diferentes indicadores de carga, não devem ser comparados. Deve-se confrontar os valores de um mesmo indicador entre máquinas distintas, uma vez que o algoritmo de balanceamento de carga utiliza os valores de um mesmo indicador de carga para arbitrar o balanceamento. O gráfico ilustrado na figura 8 serve para avaliar a variação dos indicadores entre os sistemas computacionais.

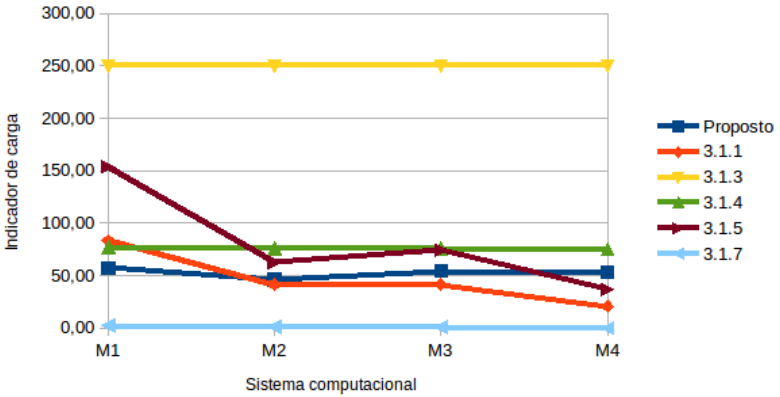

Figura 8: Valores dos indicadores de carga antes de T_1
Fonte: O próprio autor

Tabela 12: Valores dos indicadores de carga antes de T_1

Sistema computacional	3.1.1	3.1.3	3.1.4	3.1.5	3.1.7	5.3
M_1	83,33	250,00	77,44	153,96	2,44	57,47
M_2	41,67	250,00	76,22	62,75	1,22	46,11
M_3	41,67	250,00	76,22	74,20	1,22	54,52
M_4	20,83	250,00	75,09	36,75	0,09	53,20

FONTE: O próprio autor

Os indicadores de carga 3.1.1, 3.1.4, 3.1.5 e 3.1.7 indicam que o sistema computacional M_4 oferece o melhor desempenho para a execução de T_1. Somente o indicador de carga proposto (5.3) indica que M_2 oferece o melhor desempenho à execução de T_1. O escalonamento da primeira tarefa, ilustrado na figura 9, segue o indicador de carga proposto. Dessa forma, a primeira tarefa é despachada pelo CVFlow para execução no sistema computacional M_2.

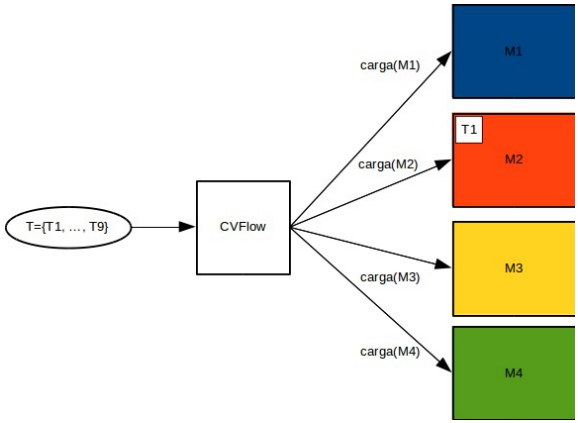

Figura 9: Escalonamento de T_1
Fonte: O próprio autor

A figura 10 exibe o gráfico com o tempo de execução médio da tarefa computacionalmente intensiva T_1, em cada um dos sistemas computacionais. Esse gráfico demonstra que o sistema computacional

M_2 executou T_1 mais velozmente. Para T_1, o indicador de carga proposto nesta dissertação oferece a mais precisa previsão de desempenho com base na carga computacional.

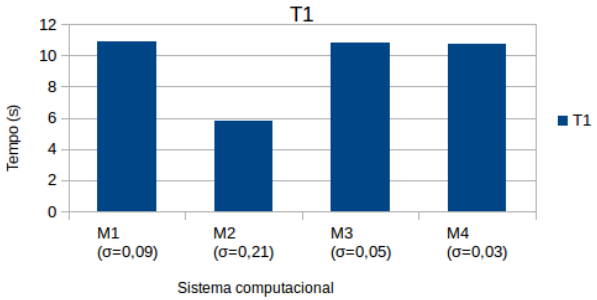

Figura 10: Tempo de execução de T_1
Fonte: O próprio autor

6.4.2 Tarefa 2 (T_2)

Após o escalonamento da tarefa T_1, o escalonador avalia qual sistema computacional oferece as melhores condições para a execução da segunda tarefa, T_2. A figura 11 e a tabela 13 exibem os valores dos indicadores de carga que serão analisados pelo escalonador, para realizar o balanceamento de carga. O menor valor calculado do indicador de carga proposto, antes da submissão de T_2, é referente ao sistema computacional M_4.

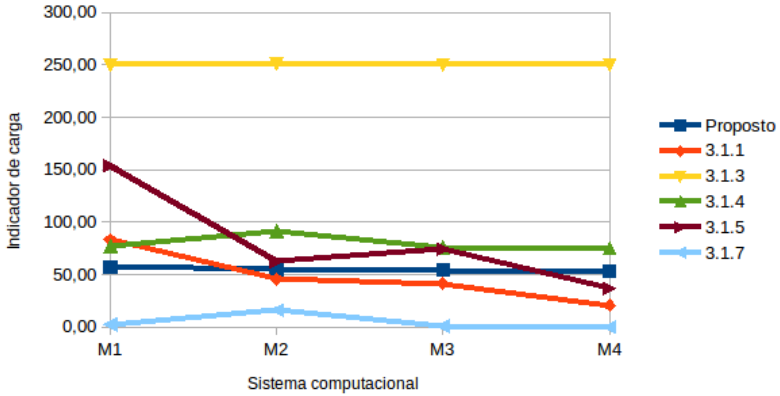

Figura 11: Valores dos indicadores de carga antes de T_2
Fonte: O próprio autor

Tabela 13: Valores dos indicadores de carga antes de T_2

Sistema computacional	3.1.1	3.1.3	3.1.4	3.1.5	3.1.7	5.3
M_1	83,33	250,00	77,44	153,96	2,44	57,47
M_2	45,73	251,00	91,52	63,00	16,22	55,37
M_3	41,67	250,00	76,22	74,20	1,22	54,52
M_4	20,83	250,00	75,09	36,75	0,09	53,20

FONTE: O próprio autor

Com exceção do indicador de carga 3.1.3, o qual fornece pesos idênticos aos três sistemas computacionais, todos os indicadores de carga apontam o sistema computacional M_4, como aquele que oferece o melhor desempenho para a execução de T_2. O escalonamento da segunda tarefa, no sistema computacional M_4, é ilustrado pela figura 12.

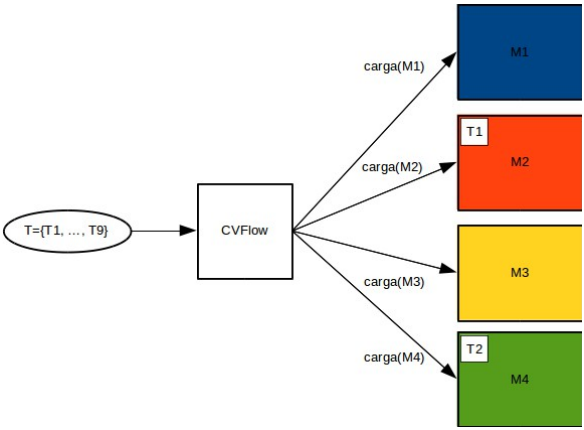

Figura 12: Escalonamento de $T_{2\ T2}$
Fonte: O próprio autor

A figura 13 exibe o gráfico com o tempo de execução médio da tarefa computacionalmente intensiva T_2. Como a tarefa T_1 já havia sido escalonada no sistema computacional M_2, o tempo de execução de T_1 é adicionado a esse sistema. Esse gráfico demonstra que o sistema computacional M_2 executou T_1 e T_2 mais velozmente do que os outros sistemas computacionais executaram somente T_2. Isso se deve ao fato de M_2 possuir núcleos de processamento disponíveis, capazes de processar paralelamente as duas tarefas, e sua arquitetura não ser virtualizada. Para T_2, nenhum dos indicadores de carga avaliados oferece a mais precisa previsão de desempenho com base na carga computacional.

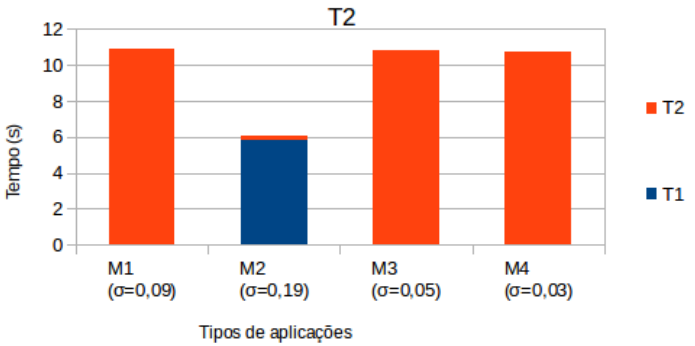

Figura 13: Tempo de execução de T_2
Fonte: O próprio autor

6.4.3 Tarefa 3 (T_3)

Após despachar as tarefas T_1 e T_2, o escalonador avalia qual o sistema computacional fornece as melhores condições para a execução de T_3. A figura 14 e a tabela 14 apresentam os valores dos indicadores de carga antes do escalonamento da tarefa T_3. O menor valor calculado do indicador de carga proposto, antes da submissão de T_3, é referente ao sistema computacional M_3.

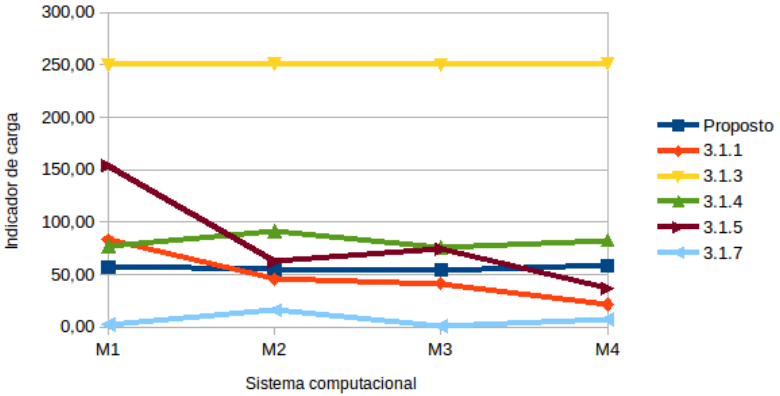

Figura 14: Valores dos indicadores de carga antes de T_3
Fonte: O próprio autor

Tabela 14: Valores dos indicadores de carga antes de T_3

Sistema computacional	3.1.1	3.1.3	3.1.4	3.1.5	3.1.7	5.3
M_1	83,33	250,00	77,44	153,96	2,44	57,47
M_2	45,73	251,00	91,52	63,00	16,22	55,37
M_3	41,67	250,00	76,22	74,20	1,22	54,52
M_4	21,87	251,00	82,89	36,89	7,59	58,73

FONTE: O próprio autor

Os indicadores de carga 3.1.1 e 3.1.5 indicam que o sistema computacional M_4 é o mais apto à execução de T_3. Os indicadores de carga 3.1.4, 3.1.7 e 5.3 apontam M_3 como o sistema computacional com a melhor relação entre carga e poder computacional. Considerando o objetivo do balanceamento como distribuir a carga computacional, esses últimos indicadores evitam sobrecarregar M_4 com mais uma tarefa. O escalonamento da tarefa T_3, segundo o indicador de carga proposto nesta dissertação, é ilustrado na figura 15.

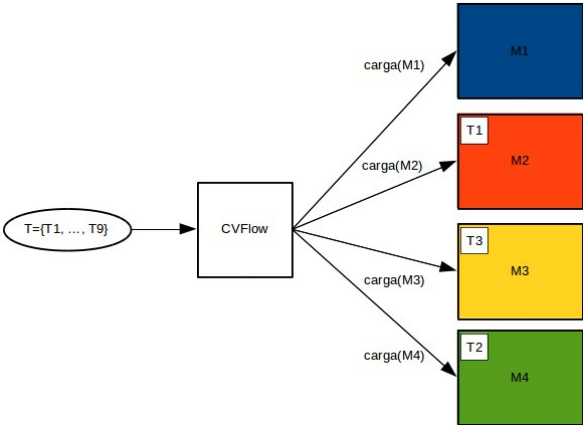

Figura 15: Escalonamento de T_3
Fonte: O próprio autor

A figura 16 exibe o gráfico com o tempo de execução médio da tarefa T_3 em cada um dos sistemas computacionais. As tarefas T_1 e T_2 já haviam sido escalonadas nos sistemas computacionais M_2 e M_4, respectivamente. Por essa razão, essas tarefas são consideradas nos cálculos de tempo médio de execução.

O gráfico da figura 16 demonstra que o sistema computacional M_2 executou T_1 e T_3 mais velozmente do que os sistemas computacionais M_1 e M_3 executaram somente T_3 e o sistema M_4 executou T_2 e T_3. De forma análoga ao ocorrido durante o escalonamento de T_2, o fato de M_2 possuir núcleos de processamento disponíveis e não ser virtualizada, permite que esse sistema execute as tarefas de forma mais veloz. Para T_3, nenhum dos indicadores de carga avaliados oferece a mais precisa previsão de desempenho com base na carga computacional. Entretanto, o indicador proposto oferece uma distribuição de carga uniforme e é o que melhor avalia M_2 em relação aos sistemas computacionais M_1 e M_4.

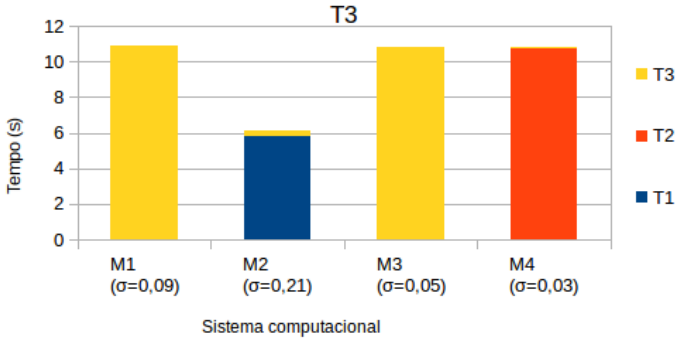

Figura 16: Tempo de execução de T_3
Fonte: O próprio autor

6.4.4 Tarefa 4 (T_4)

A figura 17 e a tabela 15 apresentam os valores dos indicadores de carga antes do escalonamento da quarta tarefa, T_4. O menor valor calculado segundo o indicador de carga proposto, antes da submissão de T_4, é referente ao sistema computacional M_2.

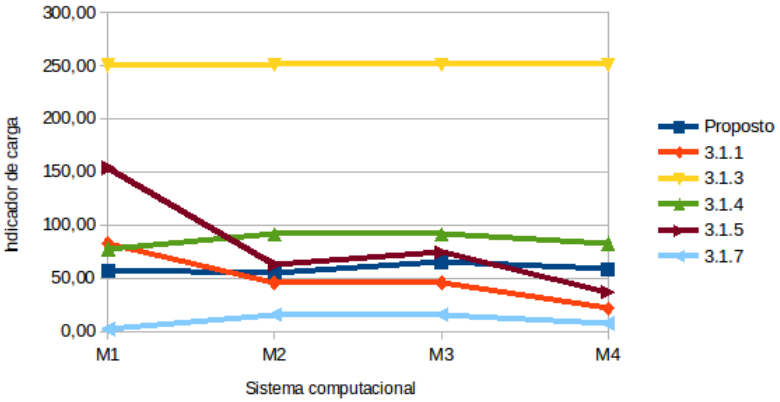

Figura 17: Valores dos indicadores de carga antes de T_4
Fonte: O próprio autor

Tabela 15: Valores dos indicadores de carga antes de T_4

Sistema computacional	3.1.1	3.1.3	3.1.4	3.1.5	3.1.7	5.3
M_1	83,33	250,00	77,44	153,96	2,44	57,47
M_2	45,73	251,00	91,52	63,00	16,22	55,37
M_3	45,73	251,00	91,52	74,50	16,22	65,47
M_4	21,87	251,00	82,89	36,89	7,59	58,73

FONTE: O próprio autor

O sistema computacional M_4 é a escolha segundo os indicadores 3.1.1 e 3.1.5. A máquina M_1 é a escolha segundo os indicadores de carga 3.1.3, 3.1.4 e 3.1.7. O indicador de carga proposto é o único que escolhe M_2 como destino da tarefa T_4. O escalonamento de T_4, segundo o indicador de carga proposto, é ilustrado pela figura 18.

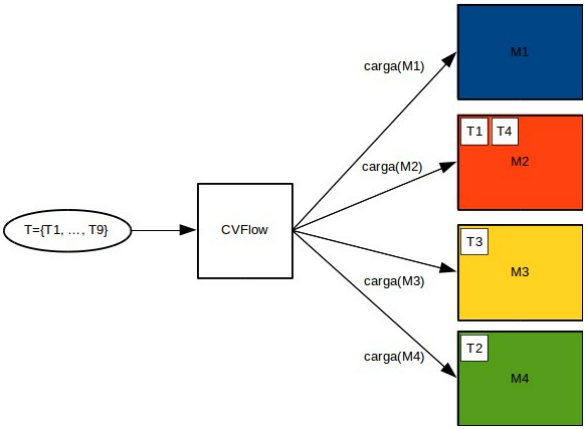

Figura 18: Escalonamento de T_4
Fonte: O próprio autor

O tempo médio de execução de T_4, em cada um dos sistemas computacionais, é exibido na figura 19. Os sistemas computacionais M_2, M_3 e M_4 possuem uma tarefa cada. Pelo fato de todos esses sistemas

computacionais possuírem núcleos de processamento disponíveis, o tempo de execução de T_4 tem impacto relativamente menor do que a primeira tarefa em execução em cada um desses sistemas. Novamente, o indicador de carga proposto é o único que escolhe o sistema computacional que fornece o melhor desempenho para a execução da tarefa, o M_2.

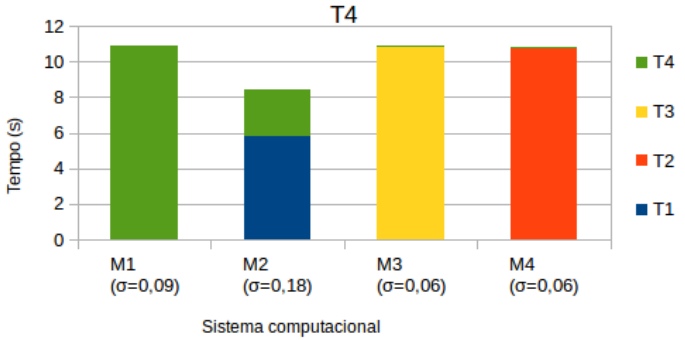

Figura 19: Tempo de execução de T_4
Fonte: O próprio autor

6.4.5 Tarefa 5 (T_5)

Os valores dos indicadores de carga, levados em consideração antes da submissão da quinta tarefa, são apresentados pela figura 20 e pela tabela 16. Segundo o indicador de carga proposto, M_1 oferece o melhor desempenho para a execução de T_5.

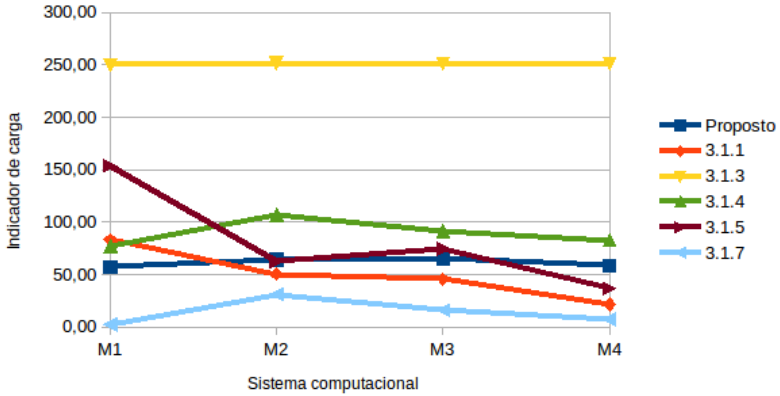

Figura 20: Valores dos indicadores de carga antes de T_5
Fonte: O próprio autor

Tabela 16: Valores dos indicadores de carga antes de T_5

Sistema computacional	3.1.1	3.1.3	3.1.4	3.1.5	3.1.7	5.3
M_1	83,33	250,00	77,44	153,96	2,44	57,47
M_2	50,60	252,00	106,82	63,25	31,22	64,62
M_3	45,73	251,00	91,52	74,50	16,22	65,47
M_4	21,87	251,00	82,89	36,89	7,59	58,73

FONTE: O próprio autor

Segundo os indicadores de carga 3.1.1 e 3.1.5, o sistema computacional M_4 é aquele que oferece o melhor desempenho para a execução de T_5. O sistema computacional M_1 é a escolha segundo os indicadores de carga 3.1.3, 3.1.4, 3.1.7 e pelo indicador de carga proposto nesta dissertação (5.3). O escalonamento de T_5, segundo o indicador de carga proposto, é ilustrado pela figura 21.

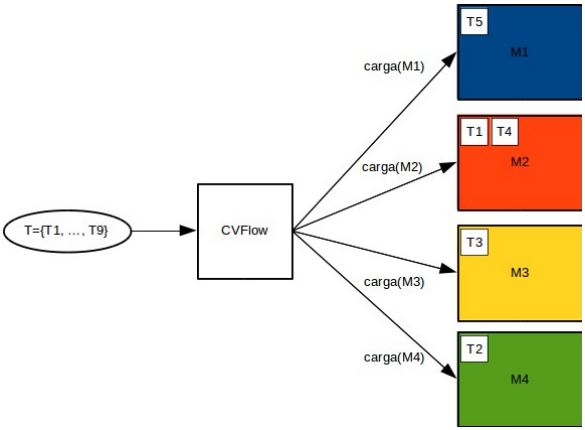

Figura 21: Escalonamento de T_5
Fonte: O próprio autor

O gráfico ilustrado na figura 22 exibe o tempo médio de execução de T_5 em cada um dos sistemas computacionais. Os sistemas computacionais M_3 e M_4 possuem uma tarefa cada e o sistema M_2 possuí duas tarefas escalonadas. Os sistemas computacionais M_1, M_3 e M_4 oferecem um desempenho similar. O sistema computacional M_2 é o que oferece o pior desempenho na execução de T_5.

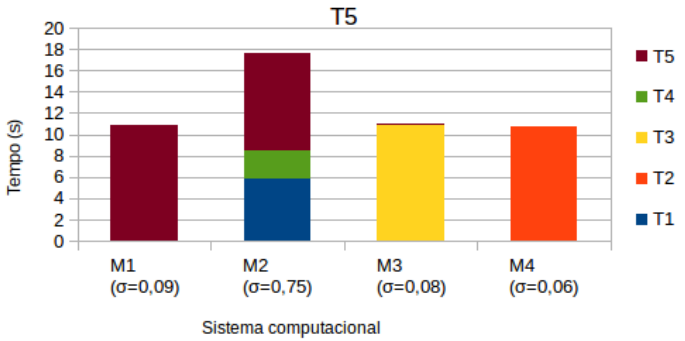

Figura 22: Tempo de execução de T_5
Fonte: O próprio autor

6.4.6 Tarefa 6 (T_6)

No momento em que a sexta tarefa é submetida, existem cinco tarefas distribuídas pelo sistema. A figura 23 e a tabela 17 apresentam os valores dos indicadores de carga antes do escalonamento de T_6. Todos os indicadores de carga apontam M_4 como o sistema computacional que oferece o melhor desempenho para a execução desta tarefa. O escalonamento de T_6, segundo o indicador de carga proposto, é ilustrado pela figura 24.

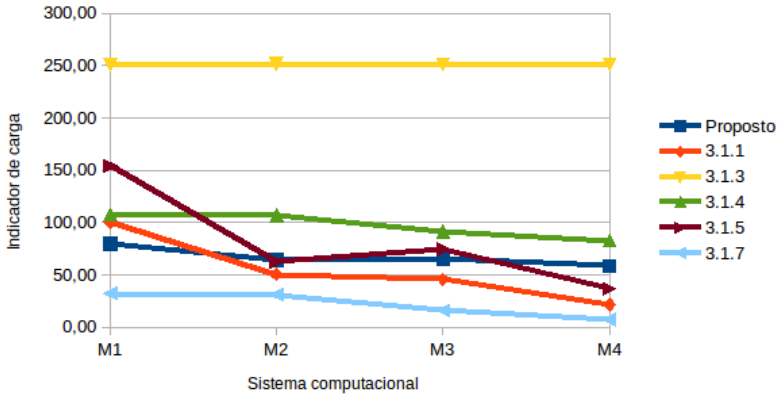

Figura 23: Valores dos indicadores de carga antes de T_6
Fonte: O próprio autor

Tabela 17: Valores dos indicadores de carga antes de T_6

Sistema computacional	3.1.1	3.1.3	3.1.4	3.1.5	3.1.7	5.3
M_1	100,60	251,00	107,74	154,58	32,44	79,96
M_2	50,60	252,00	106,82	63,25	31,22	64,62
M_3	45,73	251,00	91,52	74,50	16,22	65,47
M_4	21,87	251,00	82,89	36,89	7,59	58,73

FONTE: O próprio autor

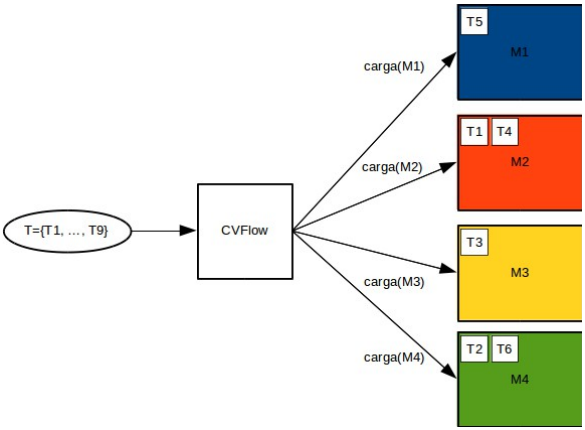

Figura 24: Escalonamento de T_6
Fonte: O próprio autor

O tempo médio de execução da sexta tarefa, em cada um dos sistemas computacionais, é exibido na figura 25. M_1, M_3 e M_4 oferecem desempenho similar, superior ao sistema computacional M_2, o qual possuí uma tarefa a mais em execução.

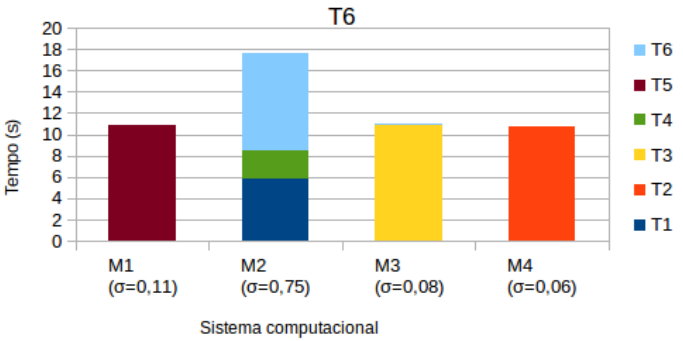

Figura 25: Tempo de execução de T_6
Fonte: O próprio autor

6.4.7 Tarefa 7 (T_7)

A figura 26 e a tabela 18 apresentam os valores dos indicadores de carga no momento em que a sétima tarefa é submetida ao sistema. Segundo o indicador de carga proposto nesta dissertação, o sistema computacional M_4 oferece o melhor desempenho para a execução de T_7, seguido, em ordem, pelos sistemas M_2, M_3 e M_1.

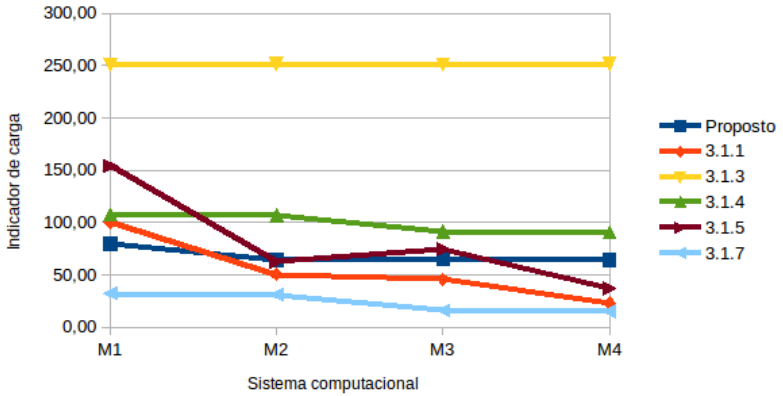

Figura 26: Valores dos indicadores de carga antes de T_7
Fonte: O próprio autor

Tabela 18: Valores dos indicadores de carga antes de T_7

Sistema computacional	3.1.1	3.1.3	3.1.4	3.1.5	3.1.7	5.3
M_1	100,60	251,00	107,74	154,58	32,44	79,96
M_2	50,60	252,00	106,82	63,25	31,22	64,62
M_3	45,73	251,00	91,52	74,50	16,22	65,47
M_4	23,00	252,00	90,69	37,04	15,09	64,26

FONTE: O próprio autor

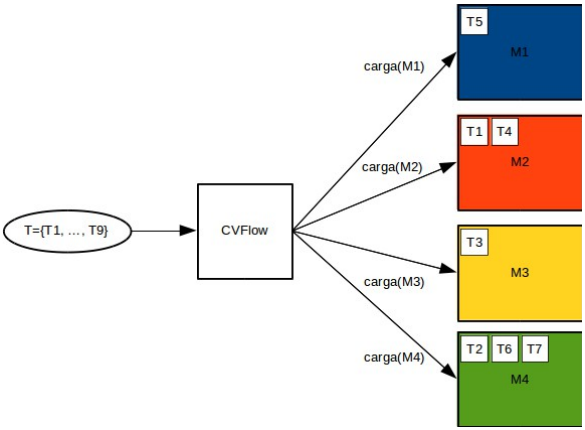

Figura 27: Escalonamento de T_7
Fonte: O próprio autor

O gráfico da figura 28 exibe o tempo médio de execução de T_7 em cada um dos sistemas computacionais. Antes de T_7 ser despachada, os sistemas computacionais M_2 e M_4 possuem duas tarefas cada e os sistemas M_1 e M_3 possuem uma tarefa cada. Os sistemas computacionais M_1, M_3 e M_4 oferecem um desempenho similar, já o M_2 é o que oferece o melhor desempenho para a execução de T_7. Para essa tarefa, nenhum dos indicadores de carga avaliados conseguiu relacionar da melhor forma carga com desempenho.

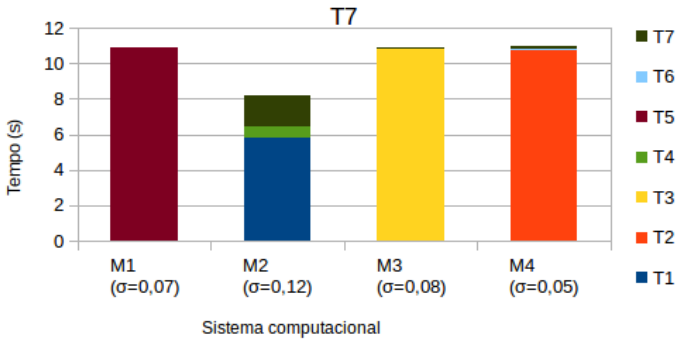

Figura 28: Tempo de execução de T_7
Fonte: O próprio autor

6.4.8 Tarefa 8 (T_8)

Seguindo o indicador proposto nesta dissertação, a oitava tarefa é escalonada no sistema computacional M_2 (figura 30). A figura 29 e a tabela 19 apresentam os valores dos indicadores de carga no momento em que a tarefa T_8 é submetida ao sistema.

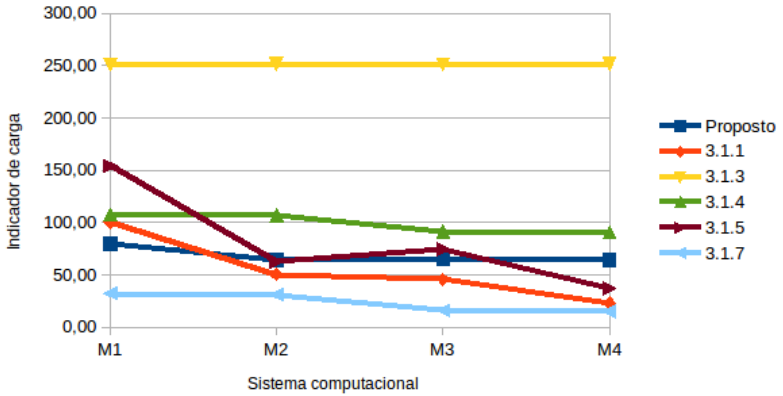

Figura 29: Valores dos indicadores de carga antes de T_8
Fonte: O próprio autor

Tabela 19: Valores dos indicadores de carga antes de T_8

Sistema computacional	3.1.1	3.1.3	3.1.4	3.1.5	3.1.7	5.3
M_1	100,60	251,00	107,74	154,58	32,44	79,96
M_2	50,60	252,00	106,82	63,25	31,22	64,62
M_3	45,73	251,00	91,52	74,50	16,22	65,47
M_4	24,24	253,00	98,49	37,19	22,59	69,78

FONTE: O próprio autor

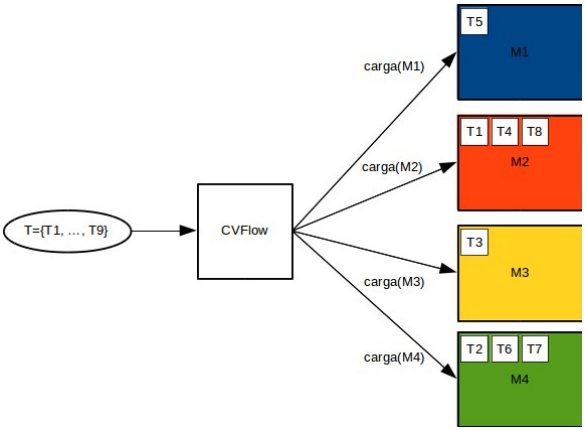

Figura 30: Escalonamento de T_8
Fonte: O próprio autor

O tempo médio de execução de T_8 é exibido no gráfico da figura 31. O sistema computacional M_2 apresenta o melhor desempenho e é o escolhida segundo o indicador de carga proposto. Nesse ponto, os sistemas computacionais M_1 e M_3 possuem uma tarefa cada, enquanto os sistemas M_2 e M_4 possuem três tarefas escalonadas cada. M_2 e M_4 são os sistemas computacionais com maior capacidade computacional tal como é possível averiguar na seção 6.2.

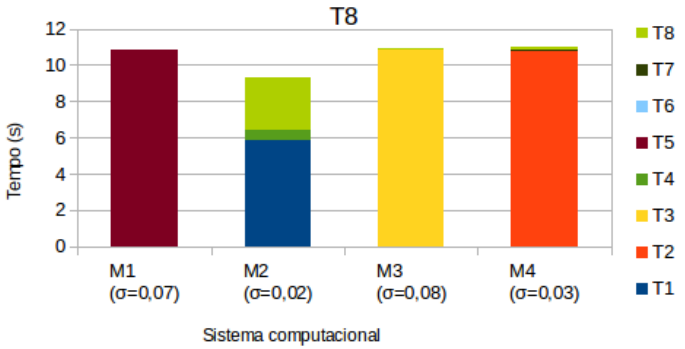

Figura 31: Tempo de execução de T_8
Fonte: O próprio autor

6.4.9 Tarefa 9 (T_9)

A última tarefa do conjunto T a ser despachada é a tarefa T_9. A nona tarefa é escalonada ao sistema computacional M_3 (representado na figura 33), o qual apresenta o menor valor para o indicador de carga proposto.

A figura 32 apresenta o gráfico de comparação dos valores dos indicadores de carga, enquanto que a tabela 20 apresenta esses valores numericamente.

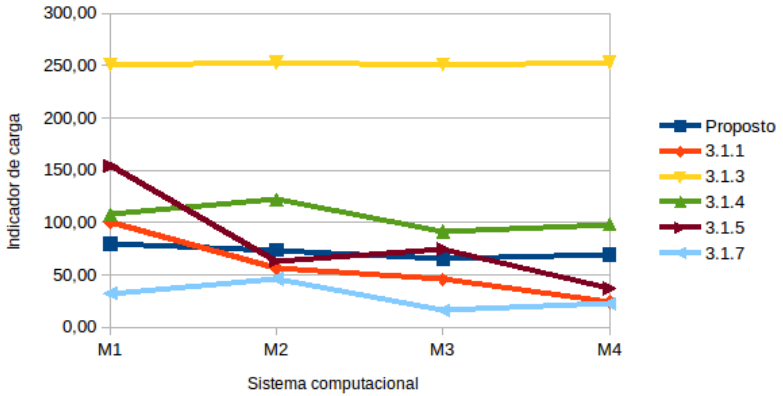

Figura 32: Valores dos indicadores de carga antes de T_9
Fonte: O próprio autor

Tabela 20: Valores dos indicadores de carga antes de T_9

Sistema computacional	3.1.1	3.1.3	3.1.4	3.1.5	3.1.7	5.3
M_1	100,60	251,00	107,74	154,58	32,44	79,96
M_2	56,56	253,00	122,12	63,50	46,22	73,88
M_3	45,73	251,00	91,52	74,50	16,22	65,47
M_4	24,24	253,00	98,49	37,19	22,59	69,78

FONTE: O próprio autor

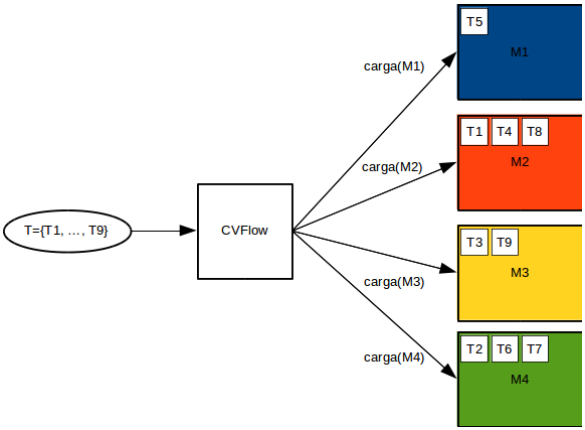

Figura 33: Escalonamento de T_9
Fonte: O próprio autor

O tempo médio da execução de T_9, em cada um dos sistemas computacionais, é exibido no gráfico da figura 34. Os sistemas computacionais M_2 e M_4 possuem três tarefas cada e os sistemas M_1 e M_3 possuem uma única tarefa cada. O indicador de carga proposto sugere executar T_9 no sistema computacional M_3.

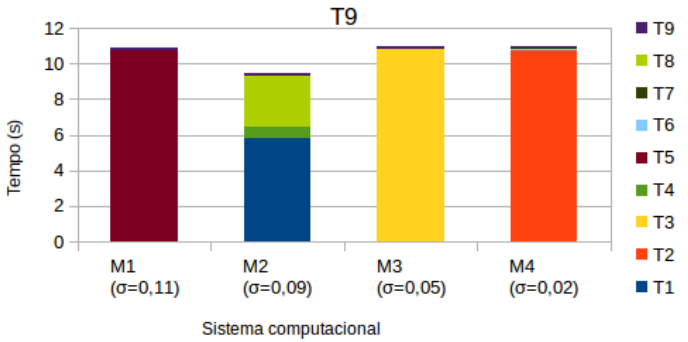

Figura 34: Tempo de execução de T_9
Fonte: O próprio autor

Dessa forma, as tarefas foram distribuídas de acordo com a capacidade de cada sistema computacional. Os sistemas computacionais

M_2 e M_4 são os com maior capacidade de processamento: M_2 por uma característica qualitativa, não ser virtualizado; e M_4 por sua característica quantitativa, ter mais núcleos de processamento. Pode-se concluir que o indicador de carga proposto distribuiu as tarefas de forma proporcional à capacidade computacional dos sistemas.

Ao longo do escalonamento desse conjunto de tarefas, pode-se resumir o comportamento de cada um dos indicadores da seguinte forma: os indicadores de carga 3.1.1, 3.1.4 e 3.1.7 não diferenciam os sistemas computacionais M_2 e M_3, pois avaliam somente as características quantitativas, e assim, não tiram vantagem de sistemas computacionais com arquiteturas mais eficientes; o indicador de carga 3.1.3 é o que apresenta a menor mudança e a pior avaliação de carga, oferecendo quase nenhuma avaliação de carga; o indicador 3.1.5 tende a escalonar as tarefas para o sistema que apresenta as melhores características quantitativas, que neste caso é a máquina M_4; e, por fim, o indicador proposto (5.3) é o único que oferece uma avaliação de carga equilibrada entre as características quantitativas e qualitativas.

7 CONCLUSÕES

O problema do balanceamento da carga computacional em sistemas arquiteturalmente heterogêneos, como grades computacionais, foi descrito na seção 1.2. A abordagem proposta no capítulo 5 desta dissertação consegue representar tanto as características qualitativas de diferentes arquiteturas quanto as características quantitativas. Esse enfoque mantém a relação de proporcionalidade entre o valor indicado da carga e o desempenho do sistema computacional.

O objetivo deste trabalho consistiu no desenvolvimento de um indicador para representar a carga computacional em sistemas arquiteturalmente heterogêneos. Por meio da análise e de experimentos com indicadores presentes na literatura, foi possível extrair as características mais relevantes para representar a carga em sistemas heterogêneos.

A primeira característica é o grau de desempenho da arquitetura (seções 2.4.3.2 e 5.3.2.1). O grau de desempenho da arquitetura permite normalizar o desempenho dos sistemas computacionais por meio de características qualitativas, as quais não são consideradas pelos outros indicadores de carga comparados.

A segunda característica relevante ao indicador de carga diz respeito à adequação entre o sistema computacional e o perfil de carga computacional que será executada. Foi possível averiguar que o indicador deve ser capaz de representar a capacidade do sistema de realizar trabalho, não somente a carga. Nesse ponto, a abordagem adotada nesta pesquisa assemelha-se àquela apresentada por outros autores, como Li (2009), Tong (2009) e Branco (2006), fornecendo pesos para componentes distintos do indicador (seções 2.4.3.1 e 5.3.1.4).

Por meio desta pesquisa, concluiu-se que representar a carga computacional em sistemas heterogêneos é intrinsecamente mais complexo do que a representação da carga em sistemas homogêneos. A principal razão disso se deve ao fato do indicador ter de representar fatores qualitativos. Dado que fatores quantitativos, como quantidade de memória, não são suficientes para que o indicador de carga represente corretamente a carga no sistema e forneça uma indicação para desempenho de execuções futuras. Além disso, o desenvolvimento do indicador requer um profundo conhecimento das arquiteturas e dispositivos de hardware dos sistemas envolvidos no balanceamento de

carga.

Diferente de Shirazi (1995), não foi possível concluir que utilizar valores cumulativos nos indicadores de carga, especialmente em relação ao processamento, ofereça uma melhor indicação da carga. A avaliação que deve ser considerada é aquela no instante da execução, ao submeter a tarefa. Por exemplo, um sistema computacional com um histórico de sobrecarga, pode ter finalizado todas suas tarefas no instante em que se submete uma nova tarefa. Segundo os experimentos, foi verificado que manter uma relação com estados passados não assegura uma execução mais veloz no futuro, ou pelo menos não existe uma relação trivial.

Em sistemas distribuídos heterogêneos, onde há grande diversidade de processadores, a velocidade dos processadores deve ser representada no indicador. Ela tem mais influência na representação carga computacional e na capacidade de realizar trabalho, do que a taxa de uso do processador isoladamente. A velocidade tem impacto principalmente para aplicações que utilizam intensamente o processador.

Em relação à memória, percebeu-se que deve haver uma relação entre a quantidade disponível e a quantidade em uso. Em um ambiente heterogêneo, onde sistemas computacionais distintos possuem quantidades variáveis de memória RAM, utilizar somente a quantidade de memória em uso não é indicativo de mais memória disponível para a execução de programas. Assim, em ambientes heterogêneos, a relação entre memória total e memória utilizada é mais importante que a taxa de uso da memória unicamente.

Os indicadores de carga presentes na literatura não informam a carga corretamente em ambientes virtualizados. Segundo os testes, mesmo em ambientes virtualizados com mais memória principal e núcleos de processamento disponíveis, a execução das tarefas levou mais tempo do que em um ambiente não-virtualizado com menos recursos. Nesses casos, o grau de desempenho da arquitetura (seção 5.3.2.1) deve ser dimensionada de modo a penalizar sistemas executados sobre máquinas virtuais.

Nos experimentos foi averiguado que o indicador de carga proposto nesta dissertação consegue representar diferenças de desempenho entre arquiteturas. Com exceção do indicador de Bosque (2013), todos os outros indicadores de carga não conseguem representar diferenças de desempenho de sistemas computacionais com arquiteturas distintas. O indicador proposto nesta dissertação tem a vantagem de não

exigir o conhecimento do sistema com o maior poder computacional para normalização de valores, como ocorre com o indicador de Bosque (2013). De modo a conhecer o poder computacional de todos os sistemas computacionais, Bosque (2013) exige a troca de uma quantidade de mensagens proporcional à quantidade de sistemas computacionais interconectados.

7.1 TRABALHOS FUTUROS

Nesta seção são descritas as melhorias que podem ser realizadas em diversos aspectos dessa dissertação. Essas melhorias foram separadas nas categorias de algoritmo de balanceamento, indicador de carga, código do CVFlow e de testes. Os dois primeiros aspectos são os mais relevantes, uma vez que representam contribuições futuras ao conhecimento (WAZLAWICK, 2008).

7.1.1 Algoritmo de balanceamento de carga

Esta dissertação teve como objetivo propor um indicador de carga para sistemas heterogêneos. Todavia, outra parte importante do balanceamento de carga diz respeito ao algoritmo de balanceamento, o qual não foi devidamente avaliado. Dessa forma, um dos principais trabalhos a serem realizados futuramente é o aprimoramento do algoritmo de balanceamento de carga.

Algumas melhorias no algoritmo incluem, mas não estão restritas à:

- Adicionar preempção ao algoritmo, de modo que uma tarefa possa interromper outra tarefa de maior prioridade, ou no caso de uma política de partilha de tempo, como *round-robin*;
- Possibilitar a migração de tarefas de um sistema computacional para outro. Isso permite, por exemplo, que um sistema, ao se tornar ocioso, possa receber tarefas de outros sistemas computacionais de menor capacidade de processamento.

7.1.2 Indicador de carga

O indicador de carga, desenvolvido e apresentado nesta dissertação, necessita de aprimoramentos. Características, como, por exemplo, taxa de transmissão da rede, acesso aos discos e consumo de energia, não foram abordados de forma satisfatória.

7.1.3 CVFlow

Existem também melhorias a serem realizadas no próprio código do CVFlow, tal como o aprimoramento da forma pela qual ele coleta informações de carga e informa ao algoritmo de balanceamento de carga. A coleta é feita por meio de bibliotecas de nível de usuário, e isso impede que informações mais detalhadas sejam coletas, uma vez que essas informações são acessíveis somente ao sistema operacional. No que diz respeito à comunicação, a carga é informada por meio de *Web Services*, o que talvez não seja a melhor forma de comunicação, ainda mais em se tratando de informações sensíveis ao desempenho. Ou que sejam afetadas por volume de tráfego de rede.

7.1.4 Testes

Os testes foram realizados em máquinas com um pequeno número de núcleos e memória. Testes em equipamentos de maior porte, com algumas centenas de núcleos de processamento, poderiam fornecer resultados distintos ou confirmar se os resultados obtidos nesta dissertação são escaláveis a todos os tamanhos de sistemas. Outra questão também diz respeito à variedade de ambientes a serem utilizados. É importante que mais tipos de arquiteturas e configurações sejam utilizados, como, por exemplo, sistemas RISC.

REFERÊNCIAS BIBLIOGRÁFICAS

BAILEY, David H. et al. **The NAS Parallel Benchmarks (NPB)**. NASA Advanced Supercomputing Division. Technical Report RNR 94-007, 1994.

BARAK, Amnon et al. **The MOSIX Distributed Operating System**: Load Balancing for UNIX. Springer, 1993.

BOSE, Pradip. **Workload characterization**: A key aspect of microarchitecture design. In: *Proceedings of IEEE Micro*, v. 26, p. 5-6, 2006.

BOSQUE, Jose L. et al. **A load index and load balancing algorithm for heterogeneous clusters**. Journal of Supercomputing, v. 65, p. 1104-1113, 2013.

BRANCO, Kalinka R. L. J. C.; ORDONEZ, Edward D.M. **Load Indices – Past, Present and Future**. In: *Proceedings of International Conference on Hybrid Information Technology*, v. 2, 2006.

BRYANT, Randal E.; O'HALLARON, David R. **Computer Systems**: A Programmer's Perspective. Prentice Hall, 2011.

CALZAROSSA, Maria; SERAZZI, Giuseppe. **Workload characterization**: a survey. In: *Proceedings of the IEEE*, v. 81, p. 1136-1150, 1993.

CORMEN, Thomas H et al. **Introduction to Algorithms**, 3rd edition. MIT Press, 2009.

COULOURIS, George et al. **Distributed Systems**: Concepts and Design, 5th edition. Addison-Wesley, 2012.

DANTAS, Mario A. R. **Computação distribuída de alto desempenho**: redes, clusters e grids computacionais. Rio de Janeiro: Axcel Books, 2005.

DEVINE, Karen D. et al. **New challenges in dynamic load balancing**. Applied Numerical Mathematics: Adaptive methods for partial differential equations and large-scale computation, v. 52, p. 133-152, 2005.

DREPPER, Ulrich. **What Every Programmer Should Know About Memory**. Red Hat, Inc. November 21, 2007.

FARIA, Izaias et al. **Energy-Aware Resource Selection on Opportunistic Grids**. In: *Proceedings of IEEE Symposium on Computers and Communication*, p. 1-6, 2014.

FERRARI, Domenico; ZHOU, Songnian. **An Empirical Investigation of Load Indices for Load Balancing Applications**. In: *Proceeding of 12th International Symposium on Computer Performance Modeling, Measurement and Evaluation*, p. 515-528, 1987.

FERREIRA, Guilherme M.; DANTAS, Mario A. R.. **A Survey on Differentiated Load Indexes for Characterization of Load Balancing**. IEEE Transactions Latin America, 2015.

GE, Chang et al. **A Survey of Power-Saving Techniques on Data Centers and Content Delivery Networks**. In: *Proceedings of IEEE Communications Surveys & Tutorials*. v. 15, 2013.

GOMES, Eliza H. A. **Uma abordagem de reserva antecipada de recursos em ambientes oportunistas**. 109 p. Dissertação (Mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2013.

GONDHI, Naveen K.; PANT, Durgesh. **An Evolutionary Approach for Scalable Load Balancing in Cluster Computing**. In: *Proceeding of IEEE International Advance Computing Conference*, p. 1259-1264, 2009.

HENDRICKSON, Bruce; DEVINE, Karen. **Dynamic Load Balancing in Computational Mechanics**. Computer Methods in Applied Mechanics and Engineering, v. 184, p. 485-500, 2000.

JOHN, Lizy K. et al. **Workload Characterization**: Motivation, Goals and Methodology. In: *Workload Characterization: Methodology and Case Studies*, p. 3-14, 1999.

KERRISK, Michael. **The Linux Programming Interface**: a Linux and UNIX System Programming Handbook. No Starch Press, 2010.

KLIAZOVICH, Dzmitry et al. **GreenCloud**: a packet-level simulator of energy-aware cloud computing data centers. In: *Proceeding of Global Telecommunications Conference*, 2010.

LECHUGA, Thiago A. et al. **An Infrastructure for Executing WS-BPEL Workflows in a Cluster of Clusters**. In: *Proceeding of IEEE Symposium on Computers and Communications*, p. 681-686, 2010.

LEVERICH, Jacob; KOZYRAKIS, Christos. **On the energy (in)efficiency of Hadoop clusters**. ACM SIGOPS Operating Systems Review. v. 44, p. 61-65, 2010.

LI, Hui. **Performance Evaluation in Grid Computing**: A Modeling and Prediction Perspective. In: *Proceeding of 7th IEEE International Symposium on Cluster Computing and the Grid*, p. 869-874, 2007.

LI, Wenzheng; SHI, Hongyan. **Dynamic Load Balancing Algorithm Based on FCFS**. In: *Proceeding of 4th International Conference on Innovative Computing, Information and Control*, 2009.

LI, Yawei; LAN, Zhiling. **A Survey of Load Balancing in Grid Computing**. In: *Proceeding of 1st International Conference on Computational and Information Science*, p. 280-285, 2004.

LIN, Frank C. H.; KELLER, Robert M.: **The Gradient Model Load Balancing Method**. In: *Proceeding of Software Engineering*, IEEE Transactions. v. 13, p. 32-38, 1987.

MANN, Steve. **Intelligent Image Processing**. John Wiley and Sons, 2001.

MANN, Steve. **Reconfigured Self as Basis for Humanistic Intelligence**. In: *Proceeding of the annual conference on USENIX Annual Technical Conference*, USENIX Association Berkeley, 1998.

MARTIN, P.; ELNAFFAR, S.; WASSERMAN, T.. **Workload Models for Autonomic Database Management Systems**. In: *Proceeding of International Conference on Autonomic and Autonomous Systems*, p 10, 2006.

MONTGOMERY, Douglas C.. **Design and analysis of experiments**, 5th edition. John Wiley and Sons, 2000.

MORGAN, Ryan. **SIGAR**: System Information Gatherer And Reporter. Disponível em: https://support.hyperic.com/display/SIGAR/Home. Versão de 20 de Dezembro de 2010.

MORO, Alessandro; MUMOLO, Enzo; NOLICH, Massimiliano. **Ergodic Continuous Hidden Markov Models for Workload Characterization**. In: *Proceeding of 6th International Symposium on Image and Signal Processing and Analysis*, p. 99-104, 2009.

PINHEIRO, Eduardo et al. **Load balancing and unbalancing for power and performance in cluster-based systems**. In: *Workshop on compilers and operating systems for low power*, p. 182-195, 2001.

PINTO, Alex S. R.. **Abordagem de escalonamento dinâmico de tarefas baseada em sistemas classificadores.** xi, 62 p. Dissertação (Mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-graduação em Ciência da Computação, Florianópolis, 2004.

RAJ, Joshua S.; FIONA, Rex. **Load balancing techniques in grid environment**: A survey. In: *Proceedings of International Conference on Computer Communication and Informatics*, p. 1-4, 2013.

SHIRAZI, Behrooz A. et al. **Scheduling and load balancing in parallel and distributed systems**. IEEE Computer Society Press, 1995.

SILVA, A. P. C.; DANTAS, Mario A. R. **A selector of grid resources based on the semantic integration of multiple ontologies**. In: *Proceedings of 19th International Symposium on Computer Architecture and High Performance Computing*, p. 143-150, 2007.

SILVA, A. P. C.; DANTAS, Mario A. R.. **An efficient approach for resource set-matching in grid computing configurations**. In: *Proceedings of 20th International Symposium on High-Performance Computing in an Advanced Collaborative Environment*, p. 5, 2006.

SILVA, Rodrigo G.. **Seleção de recursos baseada nas características das aplicações em ambientes de grade de multi-agregados**. 154 p. Dissertação (Mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2011.

STALLINGS, William. **Computer Organization and Architecture**: Designing for Performance, 8th edition. Pearson Prentice Hall, 2010.

STALLINGS, William. **Operating Systems**: Internals and Design Principles, 7th edition. Prentice Hall, 2011.

TANENBAUM, Andrew S.; STEEN, Maarten Van. **Distributed systems**: principles and paradigms, 2nd edition. Pearson, 2006.

TONG, Xiaonian; SHU, Wanneng. **An efficient dynamic load balancing scheme for heterogeneous processing system**. In: *Proceedings of International Conference on Computational Intelligence and Natural Computing*, v. 2, p. 319-322, 2009.

VIERA, Matheus A.. **Uma Abordagem para reserva antecipada de recursos em ambientes de grades computacionais móveis**. xviii, 111 p. Dissertação (Mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2011.

WAZLAWICK, Raul Sidnei. **Metodologia de pesquisa para ciência da computação**. Rio de Janeiro: Elsevier, 2008.

WEBER, Mathias H. **CVFlow. LAPiX**: Universidade Federal de Santa Catarina; 26 de Março de 2015. Disponível em: <http://www.lapix.ufsc.br/cvflow>.

WIGDOR, Daniel ; WIXON , Dennis. **Brave NUI World Designing Natural User Interfaces for Touch and Gesture**. Elsevier, 2011.

YOO, Richard M. et al. **Constructing a Non-Linear Model with Neural Networks for Workload Characterization**. In: *Proceedings of IEEE International Symposium on Workload Characterization*, p. 150-159, 2006.

ZHOU, Songnian et al. **Utopia**: A load sharing facility for large, heterogeneous distributed computer systems. In: *Software Practice & Experience*, v. 23, p. 1305-1336, 1993.

APÊNDICE A – ARTIGOS PUBLICADO

Título do Artigo Publicado
A Survey on Differentiated Load Indexes for Characterization of Load Balancing

Nome dos Autores
Guilherme Maciel Ferreira
Mario Antonio Ribeiro Dantas

Veículo da Publicação
IEEE Latin America Transactions

Classificação do Veículo no Qualis CC
B5

Título do Artigo Aceito
Load Index Characterization: The Scientific Challenge, and a Survey Approach

Nome dos Autores
Guilherme Maciel Ferreira
Mario Antonio Ribeiro Dantas

Veículo da Publicação
WETICE 2015
Track on Convergence of Distributed Clouds, Grids and their Management (CDCGM)

Classificação do Veículo no Qualis CC
B1