UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HÁLLYSSON HENRIQUE FAGUNDES DUARTE

GERENCIAMENTO DO PARTICIONAMENTO DE CONJUNTOS DE TAREFAS EM COMPUTAÇÃO NAS NUVENS

MOSSORÓ - RN
2015

HÁLLYSSON HENRIQUE FAGUNDES DUARTE

GERENCIAMENTO DO PARTICIONAMENTO DE CONJUNTOS DE TAREFAS EM COMPUTAÇÃO NAS NUVENS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação - associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientadora: Dra. Carla Katarina de Monteiro Marques
Coorientador: Dr. Carlos Heitor Pereira Liberalino

MOSSORÓ - RN
2015

Catalogação da Publicação na Fonte.
Universidade do Estado do Rio Grande do Norte.

Duarte, Hallysson Henrique Fagundes

Gerenciamento do particionamento de conjuntos de tarefas em computação nas nuvens / Hallysson Henrique Fagundes Duarte – Mossoró, RN, 2015.

63 f.

Orientador(a): Prof. Dra. Carla Katarina de Monteiro Marques

Dissertação (Mestrado) Universidade do Estado do Rio Grande do Norte. Programa De Pós-Graduação Em Ciência da Computação

1. Computação em nuvens. 2. Gerenciamento. 3. Envio de tarefas. I. Marques, Carla Katarina de Monteiro. II. Universidade do Estado do Rio Grande do Norte. III.Título.

UERN/ BC CDD 004

Bibliotecário: Sebastião Lopes Galvão Neto – CRB - 15/486

HÁLLYSSON HENRIQUE FAGUNDES DUARTE

GERENCIAMENTO DO PARTICIONAMENTO DE CONJUNTOS DE TAREFAS EM COMPUTAÇÃO NAS NUVENS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: 29 / 10 / 2015.

BANCA EXAMINADORA

Carlos Heitor Pereira Liberalino, Dr.
UERN
Co-orientador

Francisco Chagas de Lima Júnior, Dr.
UERN
Membro Interno

Everton Notreve Rebouças Queiroz Fernandes, Dr. UNP
Membro Externo

Dedico este trabalho primeiramente a Deus, pôr ser essencial em minha vida, ao meu pai Aldemir Duarte, à minha mãe Sandra Fagundes, à minha esposa e aos meus filhos.

Agradecimentos

Agradeço a Deus por ter me dado a oportunidade de concluir mais um trabalho.

Agradeço a todos os professores do Programa de Pós-graduação UFERSA-UERN, em especial, à professora Carla Katarina de Monteiro Marques e ao professor Carlos Heitor Pereira Liberalino por terem me aceitado como seu orientando e haverem me dado o suporte necessário para que essa dissertação pudesse chegar até o fim.

Aos meus pais Aldemir Duarte Lopes e Sandra Maria Fagundes Duarte que sempre acreditaram em mim e por serem meu ponto de apoio nas minhas dificuldades.

À minha esposa Catarina Melissa pela compreensão e companheirismo.

Aos meus filhos Beatriz Carlos e Leonardo Carlos que compreenderam a minha ausência.

Aos colegas Jonathan Silva e Hugo Maia que estiveram sempre presentes no desenvolvimento do trabalho.

A Wálbia Maria Carlos de Araújo Leite *in memorian* pelos conselhos, apoio e orientações.

″A persistência é o caminho do êxito.″
Charles Chaplin

Resumo

À medida que a tecnologia avança, surge uma demanda cada vez maior no que diz respeito ao poder de processamento. Um dos fatores que contribui significativamente para isso são as redes de computadores que, nos tempos atuais, é algo essencial nas atividades cotidianas. Dentro deste contexto, o uso da Computação nas Nuvens *(Cloud Computing)* surge como um novo modelo que representa a computação distribuída. Com isso, temos o problema de escalonamento de tarefas (PET) que é bastante conhecido no contexto de produção industrial e de sistemas operacionais e tem atraído o interesse de pesquisadores há bastante tempo. Esse problema consiste em encontrar a ordenação adequada de um conjunto de tarefas ao longo do tempo, obedecendo a restrições de precedências (tarefas) e recursos (processadores). Alguns trabalhos da literatura fixam valores, simulando o tempo gasto para o envio e retorno da tarefa até o cliente, considerando apenas o tempo gasto pelo processamento, gerando assim, modelagens que retratam de forma imprecisa a realidade da computação em nuvem. Logo percebe-se a necessidade de buscar na matemática, alternativas que possam aprimorar esse cálculo, com a necessidade de buscar uma maneira mais rápida de atender o cliente com o envio de suas tarefas levando em consideração todas as variáveis presentes na execução desse serviço, desde o envio até o retorno das mesmas. Este trabalho teve como objetivo o gerenciamento do particionamento de conjuntos de tarefas em computação nas nuvens, para o cálculo do tempo mínimo do envio de tarefas em sistemas heterogêneos. Determinando assim, variáveis importantes para o cálculo do envio das tarefas, obter os resultados através do modelo proposto. Calcular e comparar os resultados com trabalhos encontrados na literatura e a implementação do algoritmo baseado na modelagem.

Palavras-chave: computação em nuvem, gerenciamento, envio de tarefas.

Abstract

As technology advances, there arises an increasing demand in respect to processing power. One factor that contributes significantly to this are computer networks, which in modern times is something essential in daily activities. Within this context, the use of Cloud Computing *(Cloud Computing)*appears as a new model that is distributed computing. Thus, we have the task scheduling problem (PET) is well known in the context of industrial production and operating systems and has attracted the interest of researchers for a long time. This problem is to find the proper ordering of a set of tasks over time, obeying the restrictions of precedence (tasks) and resources (processors). Some works of literature set values, simulating the time spent sending and return the task to the client, taking into account only the time spent for processing, thus generating modeling inaccurately portraying the reality of cloud computing. Soon realizes the need to pursue mathematics, alternatives that can improve this calculation, with the need to seek a faster way to serve the customer by sending their tasks taking into account all the variables present in the execution of this service, since sending to the return of the same. This study aimed to manage the partitioning sets of tasks in cloud computing to calculate the minimum time sending tasks in heterogeneous systems. Determining so important variables for calculating the sending of tasks, get the results through the proposed model model. Calculate and compare the results with studies in the literature and implementation based on the modeling algorithm.

Key-words: cloud computing, management, sending tasks.

Lista de Figuras

Lista de tabelas

Lista de abreviaturas e siglas

t_e Tempo de envio, página 33

LBeC Largura de Banda enviada pelo cliente, página 35

LBeC Largura de banda de envio pelo cliente, página 32

LBeS Largura de banda de envio pelo servidor, página 33

LBrC Largura de banda de recebimento pelo cliente, página 32

LBrS Largura de banda de resposta pelo servidor, página 33

NInst Número de Instruções, página 32

NInstS Número de instrução do Servidor, página 33

nTaeC Número de tarefas enviadas pelo cliente, página 33

PET Problema de Escalonamento de Tarefas, página 9

PI Programação Inteira, página 21

PIM Programação Inteira Mista, página 21

PL Programação Linear, página 20

PLI Programação Linear Inteira, página 21

PLIB Programação Linear Inteira Binária, página 22

SPP Problema do Particionamento de Conjuntos, página 23

T_{total} Tempo total de envio, página 41

TamResp Tamanho da Resposta, página 32

Tamta Tamanho da tarefa, página 32

TI Tecnologia da Informação, página 9

UP Unidade de Processamento, página 10

VM Maquinas Vituais, página 14

VM Máquina Virtual, página 9

VMM Vitual Machine Monitor, página 14

Sumário

1 INTRODUÇÃO

A internet é provavelmente o maior sistema de engenharia já criado pela humanidade, com centenas de computadores conectados, links de comunicação e comutadores; centenas de milhares de usuários que se conectam esporadicamente por meio de telefones e outros (KUROSE; ROSS., 2006). À medida que a tecnologia avança, surge uma demanda cada vez maior no que diz respeito ao poder de processamento. Um dos fatores que contribui significativamente para isso são as redes de computadores, que nos tempos atuais é algo essencial nas atividades cotidianas. Dentro deste contexto, o uso da Computação nas Nuvens *(Cloud Computing)* surge como um novo modelo que representa a computação distribuída. A computação nas nuvens fornece várias vantagens, baseado em serviços *ondemand*. Esse modelo oferece a computação benefícios, como a redução de despesas de capital, uma vez que não seria necessário grandes investimentos em infraestruturas de *TI* .

Segundo Buyya et al. (2009), uma nuvem é um sistema paralelo e distribuído que consiste de uma coleção de computadores virtualizados e interconectados que são provisionados de forma dinâmica e apresentados como um ou mais recursos computacionais unificados. Estes recursos são disponibilizados e controlados por acordos relacionados aos serviços que são estabelecidos entre um prestador e um consumidor, sendo definidos a partir de negociações entre as partes.

1.1 CONTEXTUALIZAÇÃO

A infraestrutura do ambiente de computação em nuvem normalmente é composta por um grande número (centenas ou milhares) de máquinas físicas ou nós físicos de baixo custo, conectadas por meio de uma rede. Cada máquina física tem as mesmas configurações de software, mas pode ter variação na capacidade de hardware em termos de CPU, memória e armazenamento em disco (SOROR et al., 2010). Dentro de cada máquina física existe um número variável de máquinas virtuais (VM) ou nós virtuais em execução, de acordo com a capacidade do hardware disponível em cada VM. Assim, os algoritmos que solucionam o problema de escalonamento de tarefas estão sendo também utilizados nas *cloud computing*, buscando, entre outras coisas, uma melhor distribuição das cargas de trabalho nas máquinas virtuais presentes nos servidores. Para essas distribuições, encontra-se o problema de escalonamento de tarefas (PET) , além de importantes para a indústria no que diz respeito em otimizar os recursos de produção (Homens e Máquinas), são matematicamente desafiadores devido à grande

explosão combinatória, muitas vezes de ordem exponencial. Esse método tem atraído o interesse de pesquisadores há bastante tempo na busca de encontrar a ordenação adequada de um conjunto de tarefas ao longo do tempo, obedecendo a restrições de precedências (tarefas) e recursos (processadores) (AZAMBUJA; SANTOS; BORGES, 2001). É um problema de otimização combinatória bem conhecido, pertencente à classe de problemas intratáveis (NP-Completo), sendo normalmente formulado como um problema de minimização do tempo requerido para se executar um conjunto de tarefas com restrições (ARROYO; RIBEIRO, 2004).

1.2 PROBLEMÁTICA

Na literatura, é possível encontrar diversos trabalhos que tratam esses problemas, usando diferentes técnicas, como, por exemplo, o FIFO (*first in first out*), que consiste na simples alocação das primeiras tarefas nas primeiras máquinas disponíveis, até trabalhos mais elaborados usando metaheurísticas como algoritmo genético, que dispõem das mais variadas estratégias de escalonamento (PEIXOTO; TOTT; MONACO, 2007). Estas também variam de acordo com o objetivo do escalonamento, que geralmente é executar todas as tarefas no menor tempo possível, inclusive com mais de um conjunto de tarefas, mas outros objetivos também podem ser considerados como: evitar que a unidade de processamento fique muito tempo inoperante, ter a maior vazão possível (executar o maior número de tarefas num determinado prazo), atender da melhor forma possível tarefas com prioridades diferentes, entre várias outras.

Outra questão que também varia em trabalhos dessa natureza é o tipo de ambiente, pois os PETs, em seu modelo clássico, pressupõem, além das tarefas, precedências entre elas e unidades de processamento (UP) nas quais devem ser executadas as tarefas. Estas unidades podem ter características semelhantes, configurando um ambiente homogêneo, ou distintas, definido como ambiente heterogêneo (REMY, 2004). Deve-se escolher, portanto, em qual unidade de processamento cada tarefa será executada e em que momento isto ocorrerá.

Para verificar a eficiência desses escalonadores, os autores usam formulações matemáticas capazes de calcular o custo de execução de acordo com o objetivo do escalonamento. Por exemplo, em trabalhos que têm como objetivo reduzir o tempo de processamento, a formulação retorna o tempo total gasto para executar o conjunto de tarefas. Baseando-se nas respostas dessas formulações, os autores verificam o quanto o escalonador proposto melhorou ou piorou em relação aos trabalhos já existentes.

Diversas formulações são propostas na literatura (COUTINHO; DRUMMOND; FROTA, 2014) (XUE et al., 2014), com diferentes focos, levando em consideração as

características e os tipos de ambientes. Dentre os inúmeros trabalhos, muitos necessitam de um aprofudamento na determinação do tempo total de processamento, algumas etapas importantes no cálculo do tempo não estão sendo levadas em consideração nessas formulações, como é o caso do tempo gasto para o envio e o recebimento das tarefas, o que nos motivou para um estudo mais detalhado sobre o assunto.

1.3 OBJETIVO GERAL

Esta dissertação tem como objetivo geral propor o desenvolvimento de um método mais preciso que venha atender o usuário final de maneira mais rápida, através do gerenciamento do particionamento de conjuntos para escalonadores de tarefas em computação nas nuvens. Esse gerenciamento se dá em um ambiente heterogêneo para o cálculo do tempo de envio dessas tarefas, visando uma melhor precisão nos resultados retornados por essas formulações, e com isso permitir comparações mais justas, bem como o desenvolvimento de escalonadores mais eficiente. A modelagem proposta consiste no cálculo do tempo gasto por um conjunto de máquinas virtuais para a execução de determinadas tarefas, ambas com diferentes configurações. Nesse cálculo seram levados em consideração os parâmetros de envio, de processamento e do recebimento da resposta das tarefas, como largura de banda de envio (pelo servidor e pelo cliente), largura de banda de recebimento (pelo servidor e pelo cliente), tamanho das tarefas, número de tarefas enviadas por clientes e processamento das tarefas, tamanho da resposta e o número de instruções do servidor.

1.4 OBJETIVOS ESPECÍFICOS

Inicialmente é preciso fazer um levantamento dos recursos do sistema e identificar variáveis importantes para a formulação do gerencimento do particionamento de conjuntos para as etapas de envio das tarefas, do processamento das tarefas e a etapa de resposta pelo servidor das tarefas. Essa fórmula pretende mostrar resultados mais precisos para os sistemas heterogêneos. A implementação do modelo e a geração de resultados mais complexos.

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho encontra-se organizado da seguinte forma: no Capítulo 2 são levantados os conceitos básicos e essenciais para o entendimento deste trabalho. No Capítulo 3 são descritos trabalhos relacionados que deram o embasamento teórico para o desenvolvimento deste. No Capítulo 4 apresenta-se o método abordado nesta pesquisa. O Capítulo 5 apresenta o desenvolvimento da pesquisa, no qual são utilizados gráficos e fórmulas, para melhor exemplificar o gerencimento do particionamento de conjuntos em computação nas nuvens. O Capítulo 6 apresenta os resultados, bem como comparações com trabalhos encontrados na literatura. O Capítulo 7 apresenta as considerações finais e os trabalhos futuros. E por fim, as referências bibliográficas.

2 REFERENCIAL TEÓRICO

Nesta seção será abordado o referencial teórico que será utilizado para a estruturação deste trabalho. Este capítulo, está dividido da seguinte maneira: na primeira seção vai ser abordado o termo Computação em nuvem; na segunda a Modelagem Matemática; na terceira as Aplicações da Modelagem Matemática; na quarta a Modelagem Computacional e Modelagem Matemática, na quinta a Programação Matemática e com o Balanceamento de Carga.

2.1 Computação em Nuvem

As redes de computadores estão por toda parte. A internet é uma delas, assim como as muitas redes das quais ela é composta. Redes de telefones móveis, redes corporativas de fábrica, redes em campus, redes domésticas, todas elas, tanto separadamente como em conjunto, compartilham características básicas para uma definição de *sistemas distribuídos*. Define-se um sistema distribuído como aquele no qual os componentes de *hardware* ou *software*, localizados em computadores interligados em rede, comunicam-se e coordenam suas ações apenas enviando mensagens entre si (COULOURIS et al., 2013).

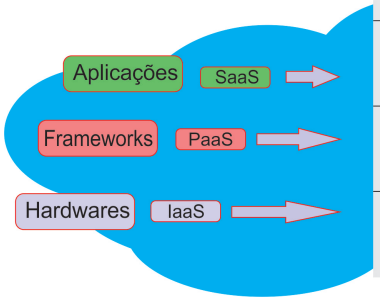
Atualmente é possível encontrar diversos sistemas distribuídos, usados em diversas áreas. A internet é um exemplo bastante utilizado. Outro tipo que vem sendo bastante utilizado é a computação em nuvem. *Cloud Computing* - Computação em nuvem é um tipo de sistema paralelo e distribuído que consiste em uma coleção de computadores interconectados e virtualizados, que são dinamicamente provisionados e apresentados como um ou mais recursos computacionais unificados (BUYYA et al., 2009).

A computação em nuvem tem transferido a entrega de serviços de *TI* para um novo nível, trazendo para seus usuários o conforto dos tradicionais utilitários tais como água e eletricidade. As vantagens das plataformas em computação nas nuvens, tais como a relação custo-eficácia, escalabilidade e facilidade de gerenciamento, incentivam mais e mais empresas e prestadores de serviços a adotar a plataforma de computação nas nuvens, e a oferecer suas soluções através de seus modelos (DASTJERDI; BUYYA, 2012).

A nuvem é uma representação para a internet ou infra-estrutura de comunicação entre componentes arquiteturais, baseada em uma abstração que oculta à complexidade da infraestrutura. Cada parte desta infraestrutura é provida como um serviço, e estes serviços são normalmente alocados em *data centers*, utilizando *hardware* compartilhado para computação e armazenamento (SOUSA; MOREIRA; MACHADO, 2009).

Com relação à arquitetura do *cloud computing*, o modelo encontrado com maior

frequência na literatura é composto por três camadas: infraestrutura como serviço(IaaS), plataforma como serviço(PaaS) e software como serviço(SaaS). Este modelo define um padrão arquitetural para soluções em computação em nuvem e pode ser visto na Figura 1 que também exibe uma breve especificação de serviços compatíveis com cada camada (BUYYA et al., 2009).



Quem utiliza	*Serviços*
Usuário final	Email, Aplicativos de escritório, Blog, CRM, etc.
Desenvolvedores e implementadores	Desenvolvimento, integração, implantação e testes.
Gerente de Sistemas	Máquinas virtuais, sistemas operacionais, redes, CPU, memória, etc.

Figura 1 – Níveis de uma nuvem

Fonte: BUYYA

A camada IaaS representa a camada inferior do modelo conceitual, sua base, ela é composta por plataformas para o desenvolvimento, teste, implantação e execução de aplicações proprietárias. Segundo SOUSA, MOREIRA e MACHADO (2009), seu principal objetivo é tornar mais fácil e acessível o fornecimento de recursos, como servidores, redes, armazenamento e outros que são fundamentais na construção de um ambiente sob demanda podendo ser tanto sistemas operacionais quanto aplicativos.

PaaS é a camada intermediária do modelo conceitual, sendo composta por hardware virtual disponibilizado como serviço. Oferece tipos específicos de serviços como sistemas operacionais, banco de dados, serviços de mensagens, serviços de armazenamento de dados e etc. Como exemplo dessa camada pode-se citar a Google App Engine e Aneka (VECCHIOLA; CHU; BUYYA, 2009).

A SaaS Corresponde a camada mais externa do modelo conceitual, ela é composta por aplicativos que são executados no ambiente da nuvem. Podem ser aplicações completas ou conjuntos de aplicações cujo uso é regulado por modelos de negócios que permitem customização.

A computação em nuvem é composta de vários conceitos e componentes, dentre os quais pode-se destacar: tarefa, máquina virtual, arquitetura cliente/servidor, largura de banda. A seguir tem-se o detalhamento dos mesmos, bem como a ilustração mostrando a interação entre eles. A Figura 2, apresenta o passo a passo da execução de um conjunto de tarefas pelo cliente enviando a um servidor com suas máquinas virtuais e depois enviando a sua resposta de recebimento. É importante destacar na Figura 2 o envio das tarefas com a largura de banda de envio pelo cliente (LBeC) e a Largura de banda de recebimento pelo cliente (LBrC) com o envio da resposta.

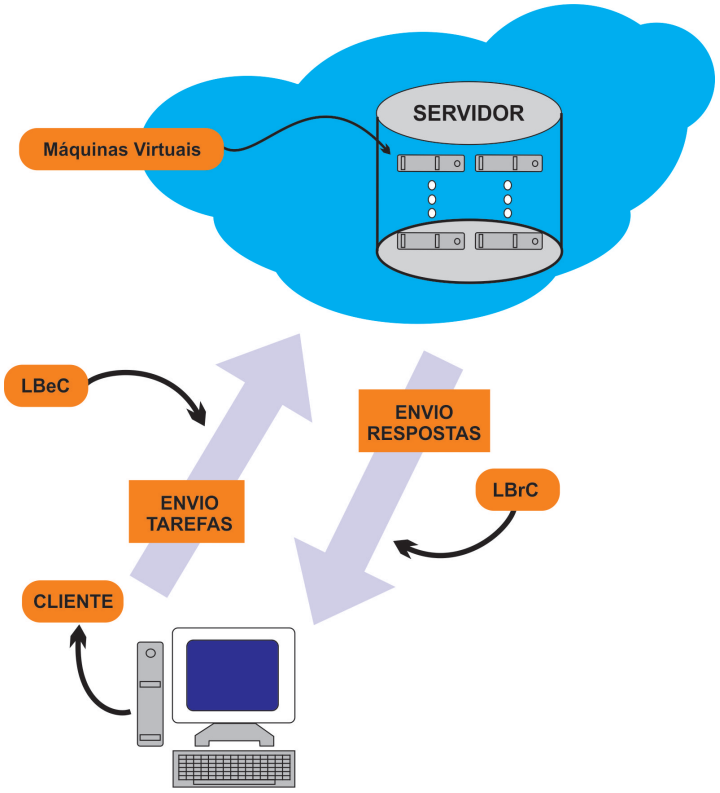


Figura 2 – Visão global de um funcionamento de uma nuvem

Fonte: autoria própria

Largura de banda é o conceito que determina a medida da capacidade de transmissão, em conexão. Na computação em nuvem, essa medida determina a velocidade que os dados trafegam através de uma rede. Ou seja, quanto maior a largura de banda, maior será a velocidade da conexão, visto que por ela passarão mais dados ao mesmo tempo.

De acordo com Tanenbaum (2003), a arquitetura Cliente/Servidor é a existência de uma plataforma base para que as aplicações, onde um ou mais Clientes e um ou mais Servidores, juntamente com o Sistema Operacional e o Sistema Operacional de Rede, executem um processamento distribuído. Numerosas aplicações funcionam de acordo com um ambiente cliente/servidor, o que significa que, máquinas(clientes) contatam um servidor, máquina geralmente bastante potente em termos de capacidades de entrada/saída, que fornece serviços aos clientes.

Os servidores, também conhecidos como *Data Center*, são compostos por diversas unidades de processamento, chamadas de *Vitual Machine* (VM) - máquinas virtuais. As VMs são vistas como duplicatas eficientes e isoladas, de uma máquina real. Essa abstração é construída por um "monitor de máquina virtual" (VMM - Virtual Machine Monitor), sendo estas unidades responsáveis por todo o processamento dentro do *Data Center* (KUROSE; ROSS., 2006).

Os arquivos enviados dos clientes para os servidores buscando a execução dos

mesmos são comumente conhecidos por tarefas. O termo tarefa é usado para denominar aquela obra e trabalho que geralmente demanda da parte de quem realiza certo esforço e que será realizado durante um tempo limitado, isto é, existe um tempo limite para sua realização (MENDES, 2007).

Na Figura 3, têm-se a parte inicial, caracterizada pelo envio da tarefa de um cliente para o servidor que será responsável pela execução da mesma. O escalonador de tarefas terá a responsabilidade de determinar quais as máquinas virtuais serão processadas. Logo após o processamento, será enviada a resposta para o cliente.

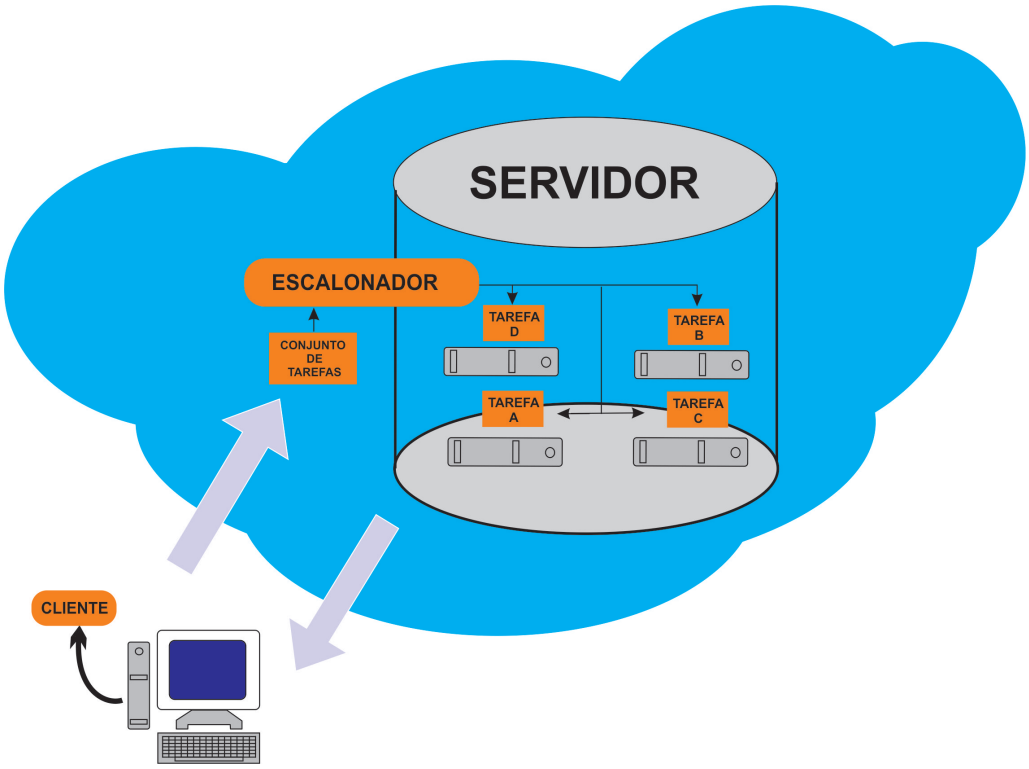


Figura 3 – O escalonador de tarefas

Fonte: autoria própria

2.2 Modelagem Matemática

Identificar e compreender os fenômenos da natureza e suas leis, realizar previsões e construir conceitos que expliquem situações que nos rodeiam tem sido uma busca constante do homem. Nesta busca, o homem com o intuito de tomar decisões encontrou uma poderosa ferramenta que contribui para a compreensão deste fenômeno: a matemática. Com esta ferramenta é possível elaborar modelos para representar situações concretas, muitas vezes impossíveis de serem analisadas na sua forma real.

A matemática é um poderoso instrumento de análise e manipulação de fatos reais. Analisar situações do cotidiano construindo modelos matemáticos que permitam essa análise tem sido um dos objetivos da área que se convencionou denominar de Matemática Aplicada (FERRUZZI et al., 2003).

Um aspecto essencial da atividade matemática consiste em construir um modelo matemático da realidade que se quer estudar, trabalhar com tal modelo e interpretar os resultados obtidos nesse trabalho, para responder as questões inicialmente apresentadas. Grande parte da atividade matemática, pode ser identificada, portanto, com uma atividade de Modelagem Matemática (CHEVALLARD et al., 2001).

Assim, um modelo pode ser entendido como uma réplica de um objeto real. É uma representação simplificada de uma situação concreta e é elaborado segundo algumas regras. Sua construção tem como objetivo a visualização e a compreensão da situação ou do objeto em estudo e a possibilidade de prever configurações futuras ou de situações semelhantes explícitas, baseadas nas hipóteses e nos objetivos admitidos para o estudo. Sua formulação não tem um fim em si mesmo, mas visa resolver algum problema.

Para o (BASSANEZI, 2002), quando se procura refletir sobre uma porção da realidade, na tentativa de entender ou agir sobre ela, o processo usual é selecionar argumentos ou parâmetros essenciais e formalizá-los através de um sistema artificial, ou seja, um Modelo. Assim, um modelo tem características fundamentais na estrutura matemática do modelo como sendo:

- Modelo Linear ou Não Linear: De acordo com as características das suas equações.
- Modelo Estático: Quando representa um objeto que não varia, como por exemplo a equação de uma circunferência, ou Modelo Dinâmico quando permite avaliar estágios da situação em estudo como por exemplo, crescimento de uma população, análise de consumo, temperatura, avanço de epidemias, etc.
- Modelo estocástico ou Modelo determinístico: de acordo com o uso ou não de fatores aleatórios. No caso de ser estocástico utiliza termos probabilísticos e é bastante usado nos dias de hoje. No caso determinístico supõe que existem informações suficientes em um determinado instante de algum processo para prever o futuro do sistema de forma precisa.

Também (BASSANEZI, 2002) define a Modelagem Matemática como sendo uma arte de transformar problemas da realidade em problemas matemáticos e resolvê-los interpretando suas soluções na linguagem do mundo real.

O objetivo da Modelagem Matemática é solucionar ou representar por meio de um modelo um problema não-matemático. A Modelagem Matemática possibilita a aproximação de situações do cotidiano com a Matemática, a interpretação e a análise

de vários fenômenos naturais e sociais. Ela é entendida como sendo uma atividade de construção, validação e aplicação de modelos de uma situação problemática, utilizando-se para isso conceitos matemáticos.

Para se elaborar e aplicar um modelo, é preciso que o modelador tenha um bom conhecimento matemático e tenha uma boa dose de intuição e criatividade. O conhecimento matemático apurado aliado à experiência e criatividade do modelador, colaboram para que este tenha uma visão mais ampla da tendência dos dados, e consiga visualizar, mesmo que superficialmente, possíveis soluções para o problema em estudo.

2.3 Aplicações da modelagem Matemática

Sua utilização tem sido observada em diferentes áreas, como a física teórica, química teórica, biomatemática, ciências da computação, ciências sociais, ciências econômicas, arqueologia, lingüística, arquitetura e filosofia. O (BASSANEZI, 2002) apresenta alguns pontos que destacam a importância da Modelagem Matemática quando é utilizada como método de pesquisa:

- Estimular a construção de novas ideias e técnicas experimentais;
- Obter informação em diferentes aspectos dos inicialmente previstos;
- Propor prioridades de aplicações e eventuais tomadas de decisões;
- Preencher lacunas onde existem falta de dados experimentais;
- Compreender melhor a realidade;
- Realizar interpolações, extrapolação e previsões;
- Obter uma linguagem universal para compreensão e entendimento entre pesquisadores das diversas áreas do conhecimento.

A Modelagem Matemática é um processo dinâmico, onde, partindo-se de um problema real, associado a um conjunto de hipóteses, é obtido um modelo que forneça possíveis soluções para o problema. A Figura 4 representa bem esse modelo, descrevendo passos que devem ser seguidos.

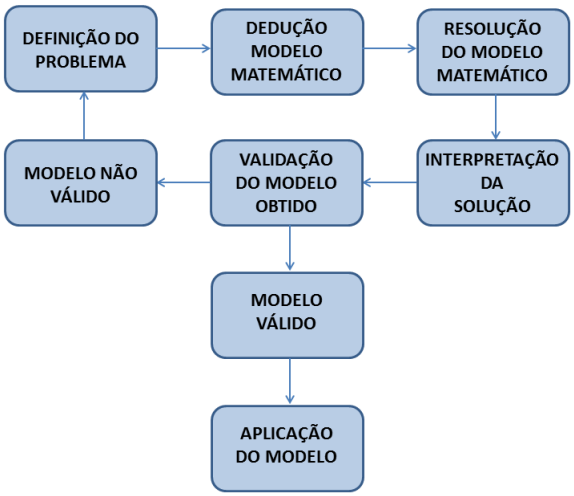


Figura 4 – Etapas de uma modelagem

Fonte: autoria própria

Definição de um problema: o processo inicia-se com uma situação real e nela é identificado o problema a ser estudado. É uma fase importante pois é preciso definir com clareza qual o problema a ser estudado. Após o reconhecimento da situação a ser estudada deve-se realizar pesquisas sobre o tema e obter os dados necessários para sua solução. Estas pesquisas podem ser bibliográficas e/ou com profissionais da área.

Dedução do modelo matemático: nesta fase substitui-se a linguagem em que se encontra o problema para uma linguagem matemática coerente. Intuição é importante nesta fase. Este modelo pode ser representado por fórmulas, gráficos, tabelas, equações, sistemas de equações, etc.

Resolução do problema matemático: é a fase em que, utilizando-se recursos da Matemática, procura-se uma solução do problema matemático formulado.

Validação: é a fase em que a aceitação do modelo encontrado é analisada. Assim, os dados reais são comparados com os dados fornecidos pelo modelo. Caso o modelo seja considerado não válido, deve-se retornar à formulação de hipóteses e simplificações e reiniciar o processo. O grau de aproximação é o fator decisivo na aceitação do modelo ou não. O grau de precisão do modelo varia conforme a sua aplicação.

Aplicação do modelo: caso seja considerado válido, o mesmo é utilizado para compreender, explicar, analisar, prever ou decidir sobre a realidade em estudo. Esta é a fase que possibilita o intervir, o exercitar, o manejar situações associadas à solução do problema.

Estas etapas não representam uma prescrição rigorosa, mas constituem uma sequência de procedimentos norteadores que podem proporcionar maior êxito no estudo de problemas por meio da Modelagem Matemática. Se o modelo não atender às

necessidades que o geraram, o processo deve ser retornado, mudando-se ou ajustando hipóteses, variáveis e outros.

A partir dos procedimentos expostos, pode-se verificar que os aspectos que distinguem Modelagem Matemática de outras aplicações de matemática são as exigências das hipóteses e das aproximações simplificadoras como requisitos na criação do modelo. Os demais aspectos – o problema, a resolução e a verificação da matemática, a validação da solução e a decisão – valem para qualquer tipo de resolução de problema envolvendo matemática.

2.4 Modelagem Computacional e Modelagem Matemática

Modelagem computacional, segundo (MOREIRA, 2014), é um dos pilares fundamentais do desenvolvimento científico atual. É praticamente impossível considerar a pesquisa em disciplinas como Física, Química e Biologia sem computação e, em particular, sem modelagem computacional. Para esses autores, o termo modelagem computacional, em geral, está associado à computação voltada para a elaboração de simulações computacionais de problemas complexos oriundos das mais diversas áreas do conhecimento. Na matemática com leitura e interpretação dos símbolos, códigos e nomenclaturas da linguagem matemática fazem parte das competências esperadas para implementação de algoritmos que sejam capazes de traduzir uma situação dada em uma linguagem em outra; por exemplo, transformar situações dadas em linguagem discursiva em gráficos, tabelas, fórmulas e outras representações, e vice-versa.

Representar a solução para uma determinada situação na linguagem algorítmica é uma das competências fundamentais do pensamento computacional (SEEHORN et al., 2011). A representação de algoritmos na forma procedural traz algumas semelhanças com a linguagem algébrica, em particular na representação de variáveis; porém, o algoritmo se constitui em um modelo dinâmico, em oposição à representação algébrica, que tipicamente é utilizada para expressar relações entre grandezas desconhecidas, relações essas que são estáticas. Segundo (MOR; NOSS, 2008), a natureza sequencial do algoritmo, definido através de procedimentos a serem seguidos passo a passo, o aproxima da linguagem discursiva. Dessa forma, representar um problema na forma algorítmica pode se constituir como uma etapa intermediária entre a narração verbal e a linguagem algébrica, podendo promover uma transição mais "suave" para a compreensão da linguagem matemática. Uma boa modelagem pode tornar um algoritmo eficiente. Por muitas vezes, o algoritmo pode ser eficaz, sem ser preciso uma solução ótima. Assim, a modelagem pode determinar um algoritmo eficiente determinando um ótimo global.

2.5 Programação Matemática

A programação matemática é o ramo da pesquisa operacional que trata de métodos de otimização (minimização ou maximização) de uma função objetivo com um número finito de variáveis de decisão sujeita a certas restrições. Estas restrições podem ser de origem financeira, tecnológica, organizacional ou outras. De modo geral, a programação matemática pode ser definida como uma representação matemática dedicada à programação ou planejamento da melhor possibilidade de alocação de recursos escassos (BRADLEY, 1977). A Programação Matemática utiliza técnicas e algoritmos para solucionar problemas modelados matematicamente. Há três motivos principais para a elaboração de modelos em programação matemática, segundo (WILLIAMS, 1999):

- Em geral é possível analisar matematicamente um modelo, sugerindo novas tendências e procedimentos que, de outra forma, não seriam percebidos;
- O procedimento de construção de um modelo revela relacionamentos que, em geral, não são evidentes, propiciando um melhor entendimento do objeto que está sendo modelado;
- Experiências que não são possíveis ou desejáveis na realidade, podem ser efetuadas a partir do modelo formulado.

Uma da mais importantes e mais utilizadas técnicas de Pesquisa Operacional é a Programação linear (PL) . A simplicidade do modelo envolvido e a disponibilidade de uma técnica de solução programável em computador como o método Simplex descrito por (DANTZIG, 1963) facilitam sua aplicação. Está técnica é amplamente utilizada, pois possui habilidade para modelar importantes e complexos problemas de decisão e o com capacidade de produzir soluções rapidamente. A descrição do método Simplex pode ser encontrado em (ZIONTS, 1963).

2.6 Balanceamento de Carga

Segundo (JR, 2009) balanceamento de carga é criar um sistema que virtualize o trabalho dos servidores físicos que executam aqueles serviços. Uma definição mais básica é a de equilibrar a carga entre vários servidores físicos, fazendo com que eles pareçam um grande servidor para o mundo externo. Há muitos motivos para fazer isso, mas os principais podem ser resumidos em escalabilidade, alta disponibilidade e

previsibilidade. A Figura 5 um conjunto de servidores em que não ocorre balancemaneto de carga, com grande variação no uso do processador. Em seguida a Figura 6 ilustra o mesmo conjunto de servidores, com a sua carga de trabalho balanceada, cuja variação do uso de processador é pequena.

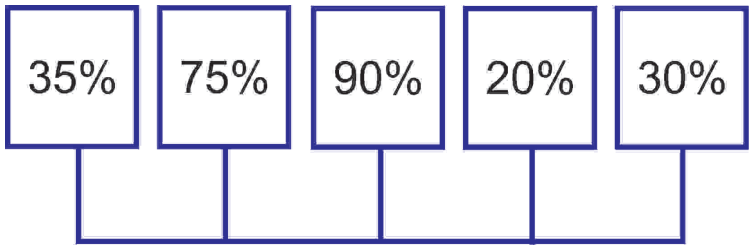


Figura 5 – Rede de servidores não balanceada

Fonte: autoria própria

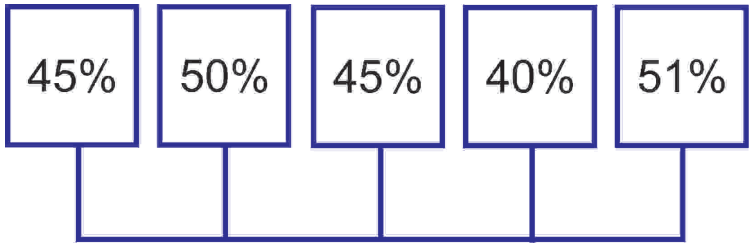


Figura 6 – Rede de servidores balanceada

Fonte: autoria própria

A seguir alguns conceitos chaves serão definidos:

- Escalabilidade: É a capacidade de adaptação fácil e dinâmica ao aumento de carga, sem impacto sobre o desempenho.
- Alta Disponibilidade: É a capacidade de um site manter-se disponível e acessível, mesmo em caso de falha de um ou mais sistemas.
- Previsibilidade: É a capacidade de ter confiança e controles na maneira como os serviços estão sendo distribuídos, com relação à disponibilidade e ao desempenho.

O balanceamento de carga surgiu da necessidade de se conectar vários servidores para gerenciar o tráfego gerado pelo acesso na Internet. Nas seções seguintes, são apresentados alguns dos principais algoritmos relacionados a balanceamento de carga como:

- Balanceamento de Carga Estático
- Balanceamento de Carga Dinâmico

2.6.1 Balanceamento de carga estático

Neste método, o desempenho dos processadores é definido no início da execução. O objetivo do método de balanceamento de carga estático é reduzir o tempo de execução total de um programa atual, enquanto minimiza o atraso de comunicação. A desvantagem geral de todo o esquema estático é que a alocação de um processo é feita, enquanto o processo é criado e não pode ser mudado durante a execução do mesmo.

2.6.1.1 Round Robin

Segundo (XU; HUANG, 2009), no algoritmo Round Robin os processos são divididos uniformemente (da mesma maneira) entre todos os processadores. Cada novo processo é atribuído para um novo processador no Round Robin em ordem. A ordem de alocação dos processos é mantida localmente em cada processador, independente da alocação do processador remoto. O algoritmo Round Robin trabalha bem quando o número de processos com uso de processador for similar (processos com consumo médio de CPU parecidos). A principal vantagem do algoritmo Round Robin é que não ocorre a comunicação interprocessos. O algoritmo Round Robin pode ter um melhor desempenho entre todos os algoritmos de balanceamento de carga para uma aplicação de propósito específico.

2.6.1.2 Algoritmo de gerenciamento centralizado

Segundo (MCENTIRE; O'REILLY; LAISON, 1984), no algoritmo centralizado um processador seleciona o servidor para novos processos. O gerenciador de carga seleciona o servidor para um novo processo, para que o processador de carga confirme para o mesmo nível, tanto quanto possível. Esta informação é atualizada por processadores remotos, que enviam uma mensagem cada vez que a carga deles muda. O gerenciador de carga toma decisão de balanceamento de carga, baseado na informação de carga do sistema, permitindo a melhor decisão enquanto o processo é criado. O alto grau de comunicação interprocessos pode ser o gargalo. Neste algoritmo, é esperado executar melhor em aplicações paralelas, especialmente quando atividades dinâmicas são criadas por diferentes servidores.

2.6.2 Balanceamento dinâmico de carga

No balanceamento de carga dinâmico o trabalho é distribuído entre os processadores durante a execução, conforme descrito abaixo:

2.6.2.1 Balanceamento dinâmico de carga-com fila

Segundo (MALIK, 2000), o processador master atribui o novo processo para o escravo, baseado nas novas informações coletadas. Ao contrário dos algoritmos estáticos, algoritmos dinâmicos alocam os processos dinamicamente quando um processador torna-se subcarregado. Estes são armazenados em uma fila no servidor principal e alocados dinamicamente na solicitação de um servidor remoto. Alguns desses balaceamentos dinâmico com fila são:

- Algoritmo de fila central: Segundo (LEINBERGER et al., 2000), o algoritmo de fila central trabalha sobre os princípios de distribuição dinâmica. O algoritmo armazena novas atividades e requisições não realizadas em uma fila FIFO, no servidor principal. Cada nova atividade que chega ao gerenciador de fila é inserida na fila. Então, sempre que uma requisição por uma atividade é recebida pelo gerenciador da fila, o algoritmo remove a primeira atividade da fila e a envia para quem solicitou. Caso não tenha nenhuma atividade na fila, a requisição é armazenada, até uma nova atividade estar disponível. O problema deste algoritmo é que existe um único ponto de falha, a fila central.

- Algoritmo de fila local: (LEINBERGER et al., 2000), a principal característica deste algoritmo é o suporte à migração dinâmica dos processos. Inicialmente, novos processos criados no servidor principal são alocados em todos os servidores subcarregados. Daí para frente, todos os processos criados no servidor principal e todos os outros servidores são alocados localmente. Quando o servidor fica subcarregado, o gerenciador de carga local tenta conseguir processos de servidores remotos. O algoritmo envia requisições aleatoriamente com o número de processos local para o gerenciador de carga. Quando um gerenciador de carga recebe tal requisição, compara com o número local de processos com o número recebido. Se o primeiro for maior que o último, então, alguns dos processos que estão executando são transferidos para o requisitante.

3 TRABALHOS RELACIONADOS

Nesta seção serão abordados os trabalhos relacionados com a problemática e o objetivo da proposta da dissertação.

A falta de abordagens existentes nos motivou a aplicar o conhecimento a partir de outras áreas. Em particular, este trabalho foi inspirado na pesquisa (BOROVSKIY et al., 2011). Ele apresenta um modelo baseado no uso da computação em nuvem criando assim, um incentivo para os assinantes otimizarem o uso dos recursos alugados. O objetivo do trabalho é conceber uma abordagem formal para a distribuição de cargas de trabalhos entre o número mínimo de servidores. A modelagem deste problema é definida como um problema de particionamento de conjunto e descreve duas abordagens de solução. A primeira gera um conjunto de blocos de candidatos e, em seguida, compõe uma partição ideal, resolvendo um problema de programação linear inteira. A segunda abordagem resolve o conjunto de particionamento do problema com a técnica de geração de colunas. O SPP é um problema de otimização combinatória que pode ser utilizado em várias situações, o mesmo já foi utilizado em outros trabalhos no contexto de computação nas nuvens. Porém esse trabalho, não utiliza nenhuma modelagem matemática para que possa melhorar ainda mais o tempo de envio. Em (WAN; ALMEIDA, 2012) o problema consiste na diminuição do consumo de energia nos *data centers* distribuindo cargas de trabalho em um ambiente homogêneo de servidores usando a programação linear inteira binária, mas não se utilizando do SPP.

O estudo proposto em (BRAGA, 2011) é realizado por meio das seguintes extensões do referido algoritmo: a implementação de uma partida quente para o multiplicador de uma inequação adicionada; a incorporação do algoritmo a uma enumeração, gerando, assim, um *branch-and-cut* baseado em relaxação Lagrangiana para o SPP.

Existem diversos trabalhos na literatura que trata do problema de particionamento de conjuntos, porém a maioria deles trata do problema de alocação de tripulação, principalmente para o escalonamento de aeronaves, como em (MOHAN; RAJ, 2012), (OZDEMIR; MOHAN, 1999), (CHEN; CHEN; ZHANG, 2010). Apesar de se trabalhar com o escalonamento nenhum deles estão relacionados com a computação nas nuvens.

Em trabalhos como os de (YUNES; MOURA; SOUZA, 2005) e (BARNHART et al., 1998) são abordados o problema de geração de colunas em problemas de gestão de trânsito, se utiliza do escalonamento de tarefa, mas não voltado para a computação em nuvem. Em (BACHIEGA, 2014) o trabalho teve como objetivo desenvolver um algoritmo de escalonamento para nuvem que determinasse de maneira eficiente a distribuição de recursos dentro da arquitetura se utilizando de gestores de nuvens, porém não se utilizaram de uma modelagem matemática para o método.

Já em (COUTINHO; DRUMMOND; FROTA, 2014) o problema é de gestão de recursos. Este trabalho propõe selecionar recursos da nuvem com objetivo de minimizar o tempo de execução total, minimizar o custo para o pagamento do aluguel e a demanda de energia para reduzir o custo no aluguel do serviço e o tempo de execução de aplicativos pelo usuário. A fim de resolver este problema, os autores formularam um modelo matemático para o cálculo do tempo de envio total destas tarefas. Mas ficou claro em seu trabalho, que os autores desprezam várias variáveis importantes no trabalho para o cálculo do tempo de execução das tarefas deixando assim, um tempo não preciso.

Em (XUE et al., 2014) o artigo apresenta um algoritmo para o escalonamento de tarefas na computação na nuvem que tem como objetivo o tempo mínimo, o balanceamento das cargas de trabalhos bem como o consumo mínimo de energia usando um algoritmo que chamam de Algoritmo de evolução diferencial para a computação nas nuvens.

Nesses dois últimos trabalhos citados, percebe-se que tratam do mesmo problema e com o mesmo modelo matemático para o cálculo do tempo de envio dessas tarefas. Um outro fator importante, é que a formulação desses problemas é para os sistemas homogêneos e que os autores desprezam variáveis importantes para o cálculo do tempo de envio de tarefas. Essas variáveis podem trazer um grande diferencial nos resultados, podendo cada vez mais, se aproximar do resultado preciso. Assim esses dois trabalhos vão servir como base para a pesquisa e gerar resultados para o nosso trabalho.

Em resumo, os trabalhos encontrados na literatura, possuem um sistema homogêneo. Esse tipo de sistema por exemplo, só começa a processar quando todas as tarefas chegam nas maquinas virtuais. Em um sistema heterogêneo as tarefas que foram enviadas já podem ser processadas sem precisar da última ser enviada. Com essa formulação encontrada nos trabalhos, os resultados não trazem valores próximos da realidade de um método exato. Esta é a principal motivação para este trabalho que se diferencia dos demais por apresentar um modelo matemático que mostra valores reais levando em consideração o tempo de envio e o tempo de resposta dessas tarefas.

4 MÉTODO DE PESQUISA

Inicialmente foi feito um estudo sobre Sistemas Distribuídos buscando uma compreensão melhor em torno do tema. Esse estudo possibilitou o conhecimento de algumas sub áreas, bem como a importância das mesmas. Após esse estudo, percebe-se que a computação em nuvens, umas das sub áreas de sistemas distribuídos, vem sendo bastante explorada, visto a grande importância que a mesma vem ganhando no cenário atual. (BUYYA et al., 2009) definiu uma nuvem como sendo um tipo de sistema distribuído e paralelo consistindo de uma coleção de servidores interconectados e virtualizados, que são dinamicamente provisionados e apresentados como um ou mais recursos de computação únicos baseados em acordos em nível de serviço, estabelecidos através de negociação entre fornecedores de serviços e clientes. A Figura 7 mostra vários aparelhos conectados a uma rede de internet com armazenamentos em nuvens, muitos aplicativos dos usuários, assim como seus arquivos e dados relacionados, não precisam mais estar instalados ou armazenados em seu servidor, pois ficam disponíveis na "nuvem", isto é, na Internet. Para o fornecedor da aplicação, ficam todas as tarefas de desenvolvimento, armazenamento, manutenção, atualização, backup, escalonamento, etc. O usuário não precisa se preocupar com mais nada disso, apenas com o acessar e usar.

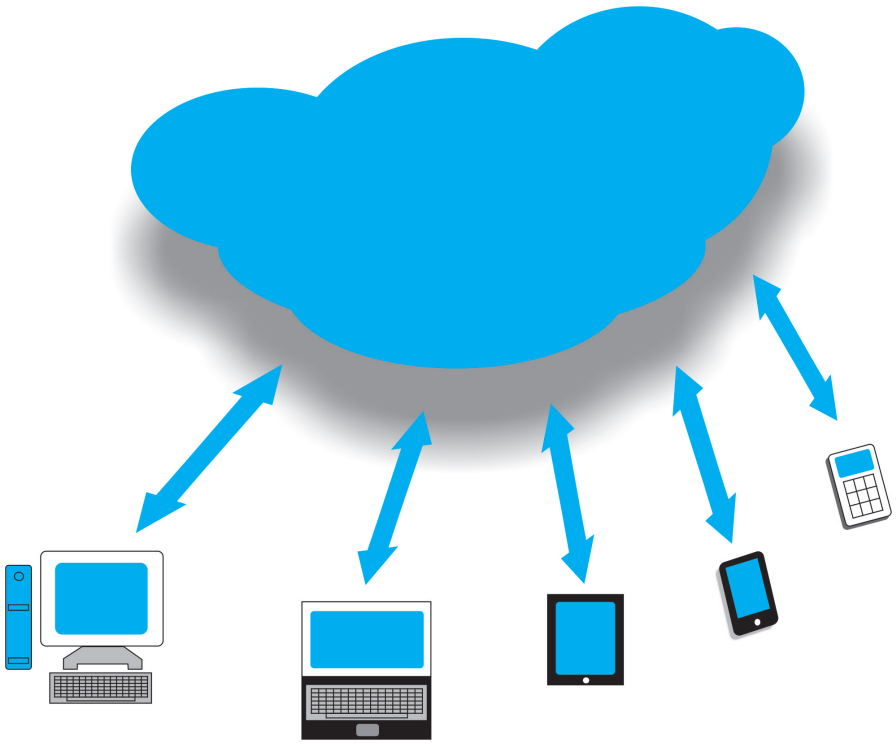


Figura 7 – Nuvem

Fonte: autoria própria

Esse estudo teve como objetivo, além do conhecimento prévio sobre o tema, uma busca de possíveis áreas a serem pesquisadas. Assim percebe-se que não diferente de áreas como a industria e sistemas operacionais (que precisam escalonar as tarefas a serem executadas, em máquinas disponíveis) a computação em nuvem necessita escalonar as tarefas (serviços) que são solicitadas pelos clientes, nas máquinas virtuais disponíveis.

Para melhorar esse escalonamento, os trabalhos presentes na literatura, levam em consideração entre outros parâmetros, o tempo necessário para a execução de um conjunto de tarefas em um determinado número de máquinas. Esse tempo é calculado levando em consideração desde o tempo gasto para o envio da tarefa, passando pelo processamento, até o retorno do resultado como mostra a Figura 6. A figura mostra um cliente enviando uma tarefa para nuvem, onde se encontram as maquinas virtuais e depois envia a resposta de recebimento.

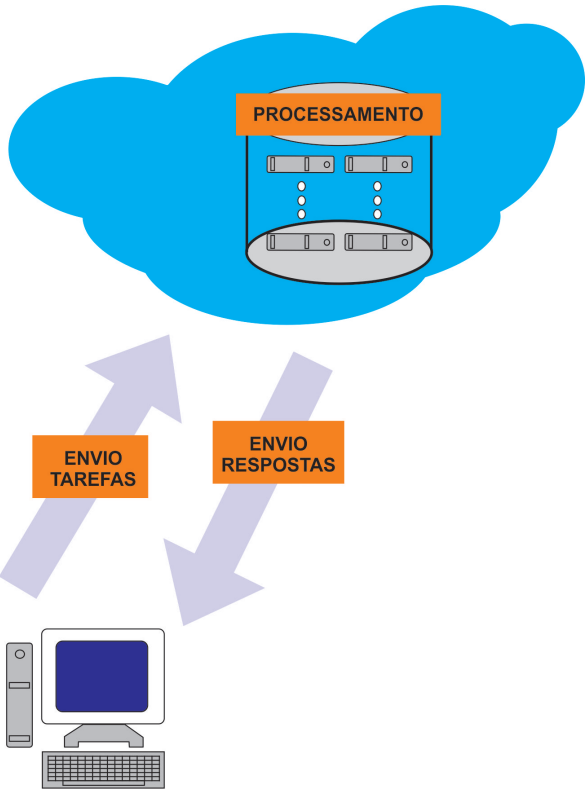


Figura 8 – Envio-Processamento-Resposta

Fonte: autoria própria

Porém, na prática, de acordo com primeiros trabalhos pesquisados, não era dessa forma que funcionava, pois as modelagens encontradas não levavam em consideração todas as variáveis. Com isso surgiu a necessidade de aprofundar as pesquisas, buscando um levantamento sobre o estado da arte com relação às formulações matemáticas que são usadas para calcular o tempo total gasto na execução dessas tarefas.

De acordo com o que foi visto, confirmando a suspeita inicial, as modelagens

usadas nas pesquisas não retratam por completo todos os passos da execução de tarefas em computação em nuvem. Alguns trabalhos fixam valores, simulando o tempo gasto para o envio e retorno da tarefa até o cliente, considerando apenas o tempo gasto pelo processamento, gerando assim, modelagens que retratam de forma imprecisa a realidade da computação em nuvem. Logo percebe-se a necessidade de buscar na matemática, alternativas que pudessem aprimorar esse cálculo, onde seria levado em consideração todas as variáveis presentes na execução desse serviço, desde o envio até o retorno das mesmas, e com isso desenvolver uma modelagem que retrate de forma mais fiel essa realidade.

Com relação ao estudo dessas modelagens, percebe-se que é antigo e bastante utilizado em poderosas ferramentas de análise e levantamento de situações cotidianas. Essa fase da pesquisa nos permitiu visualizar uma forma de retratar o problema do escalonamento de tarefas em nuvem, em uma formulação matemática mais precisa.

A pesquisa relacionada à modelagem matemática, nos permitiu uma ligação para conhecer de fato o problema tratado nesse trabalho, nos permitiu conhecer a modelagem como um dos fatores primordiais para atacar o nosso problema. Conhecer de fato o problema tratado nesse trabalho, nos permitiu conhecer a modelagem como um dos fatores primordiais para atacar o nosso problema. Baseado na modelagem foram feitos vários testes para verificar a eficaz da modelagem. Para esses testes, foram feitos um estudo prévio no escalonamento em Computação nas Nuvens. Um estudo nas estratégias com maior eficiência na construção de um modelo matemático. Conhecer o processo de envios dessas tarefas, como a largura de banda de envio pelo cliente e largura de banda de recebimento pelo servidor.

Esses processos foram fundamentais em nossa pesquisa, fornecendo assim, resultados das atividades desenvolvidas por meio de submissão de artigos e da escrita dessa dissertação de mestrado.

5 DESENVOLVIMENTO

Este capítulo tem como objetivo, mostrar o processo de envio dessas tarefas e modelar matematicamente o processo para o cálculo do tempo de envio das tarefas.

5.1 Modelagem do Problema

Para um melhor entendimento desse processo, a definição de cliente e servidor poderá nos ajudar nesse primeiro momento. Um programa cliente é um programa que funciona em um sistema final, que solicita e recebe um serviço de um programa servidor, que funciona em um outro sistema final. O programa servidor interage enviando mensagens um para o outro pela internet (COULOURIS et al., 2013). Primeiro será analisada uma tarefa isolada. Considerando o ambiente de *cloud computing*, uma tarefa de um cliente tem que ser primeiramente enviada ao servidor e esse tempo dependerá da largura de banda da conexão entre o cliente e o servidor, ou seja, quantos bits podem ser enviados por segundo. Porem, como normalmente as larguras de banda disponíveis nos servidores são superiores à dos clientes, pode-se dizer que, na prática, esse tempo pode ser somente o tamanho em bits da tarefa dividido pela largura de banda. Agora tomando *te* como sendo o tempo de envio da tarefa, *TamTa* sendo o tamanho em bits da tarefa e *LBC* a largura de banda do cliente, tem-se o tempo de envio dessa tarefa em:

$$te = \frac{TamTa}{LBC} \tag{5.1}$$

Em seguida a tarefa será processada no servidor. Esse tempo é impreciso, pois varia muito devido às configurações da máquina e aos processos que estão em execução no momento. Desse modo será considerado um tempo estimado em um servidor hipotético. Esse servidor hipotético (daqui em diante chamado servidor padrão) servirá como base para saber o tempo que demoraria em um determinado servidor. Assim, se uma tarefa demoraria dois segundos processando nesse servidor padrão e o servidor escolhido tem duas vezes seu poder de processamento, a tarefa demorará a metade do tempo. Logo o *TProc* tempo de processamento da tarefa será definido como *TP* tempo de processamento estimado dividido pelo *PPS* poder de processamento do servidor escolhido.

$$TProc = \frac{TP}{PPS} \tag{5.2}$$

E, por fim, uma resposta é enviada ao cliente. Essa seguirá a mesma lógica do envio, porém olhando pelo lado do recebimento do cliente, *TRes*Tempo de resposta, *TamResp* Tamanho da Resposta e *LBrC* Largura de banda de recebimento do cliente. Assim o tempo de resposta fica:

$$TRes = \frac{TamResp}{LBrC} \tag{5.3}$$

Considerando que os recursos do cliente e do servidor serão divididos igualmente entre as tarefas e que as trocas de contexto das máquinas serão desprezíveis, o tempo necessário para executar mais de uma tarefa seria simplesmente o somatório dos tempos de cada tarefa. Desse modo, se houverem n tarefas a fórmula será definida com o tempo total:

$$T_{total} = \frac{\sum_{i=1}^{n} TamTa_i}{LBeC} + \frac{\sum_{i=1}^{n} Tp_i}{PPS} + \frac{\sum_{i=1}^{n} TamResp_i}{LBrC} \tag{5.4}$$

5.2 Formulação de Problemas

Na seção anterior defini-se as fórmulas que serão fundamentais para uma melhor compreensão do trabalho proposto. Para isso, será necessário conhecer a largura de banda de envio pelo cliente, poder de processamento, largura de banda de recebimento do cliente e que todos sejam maiores que zero. Todavia, tarefas diferentes podem ter tamanhos, tempos de processamento e tamanhos de respostas diferentes. Assim, para que a fórmulas (5.2)(5.3)(5.4) estejam corretas, o servidor esperaria todas as tarefas serem transferidas para começar o processamento e todas serem processadas para começar a enviar a resposta ao cliente. A Figura 9 mostra a espera do processamento aguardando a finalização do envio da tarefa e a espera do envio da resposta para a finalização do processo.

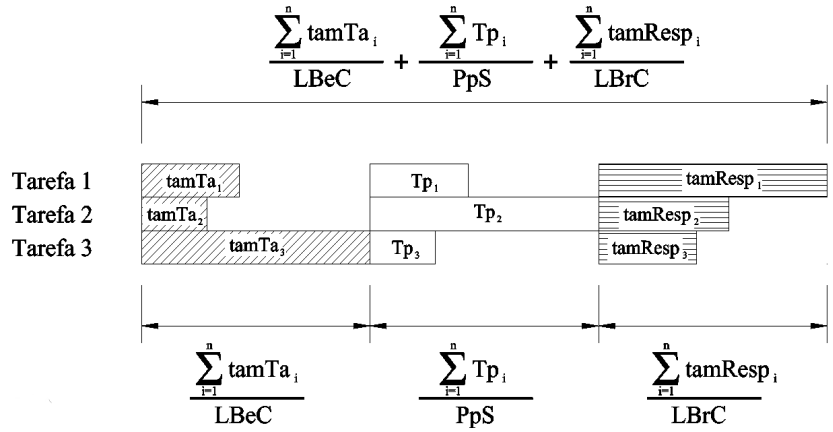


Figura 9 – Processo de envio das tarefas com espera

Fonte: autoria própria

Porém, na prática, não é isso o que ocorre. Assim que uma tarefa estiver completa no servidor, ele vai começar a processá-la e a responder ao cliente assim que acabar o processamento, de modo que enquanto uma tarefa maior ainda estiver sendo transferida, uma menor pode já estar em processamento. Neste caso, a Figura 10 abaixo mostra as tarefas individualmente sendo processadas assim que acaba o envio.

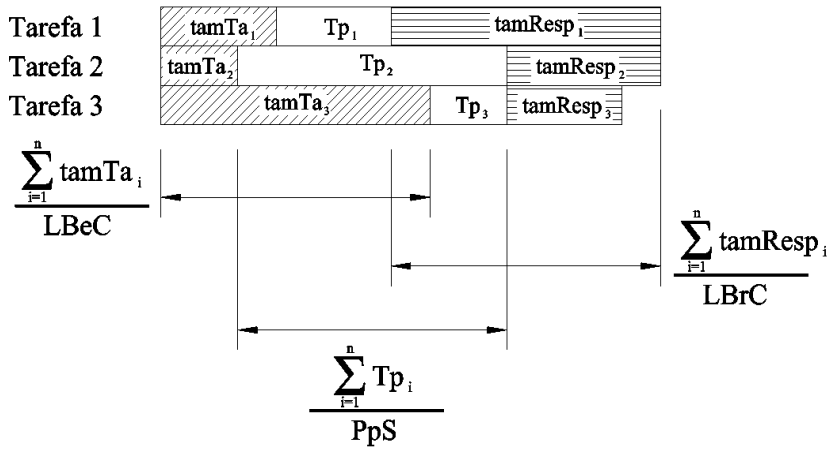


Figura 10 – Envio-Processamento-Resposta

Fonte: autoria própria

Imaginando que as tarefas 1, 2 e 3 possuem tamanhos de 3, 2 e 7 megabytes respectivamente e que esses valores foram escolhidos aleatoriamente para o problema e que apenas um cliente está enviando essas tarefas. A largura de banda de envio do cliente seja de 2mbps (dois megabits por segundo). Primeiro os bits têm que ser convertidos para bytes, assim a largura de banda fica como 256kB/s (256 quilobytes por segundo). Então o tempo total para enviar as tarefas será:

$$T_{total} = \frac{3Mb}{256KB/s} + \frac{2MB}{256KB/s} + \frac{7MB}{256KB/s} = \frac{12MB}{256KB/s}$$

convertendo MB para KB fica:

$$T_{total} = \frac{12288kB}{256kB/s} = 48s$$

Porém cada tarefa termina em um tempo diferente e isso se deve ao fato de os recursos estarem divididos igualmente entre elas, mas as mesmas não possuírem tamanhos iguais. Como todas as tarefas começaram ao mesmo tempo, a primeira tarefa a terminar é a tarefa 2, pois é a que possui o menor tamanho. Como as três estão sendo enviadas, a largura de banda é dividida pelas três até a tarefa 2 acabar. A segunda tarefa a terminar é a tarefa 1, pois é a segunda menor. Vale notar aqui que enquanto a tarefa 2 era enviada, parte das tarefas 1 e 3 também foram enviadas. De modo que é preciso ver quanto falta para cada tarefa terminar. Se as três tarefas estavam nas mesmas condições, então elas enviaram a mesma quantidade para o servidor. Logo, todas as tarefas enviaram 2 MB, pois é o tamanho da tarefa 2 (que era a menor tarefa a concluir). Assim sobraram 1, 0 e 5 MB de cada tarefa respectivamente. Desse modo a largura de banda será dividida por duas tarefas e o tempo de envio do restante da tarefa 1. Semelhante ao que ocorreu quando a tarefa 2 terminou, todas as tarefas terão seus tamanhos reduzidos em um valor igual ao que foi transferido da menor tarefa, ou seja, 1 MB, fica assim 0, 0 e 4 MB a serem transferidos. E a tarefa 3 terá toda a banda disponível. Essa mesma lógica pode ser tomada para o processamento e o envio da resposta, com a ressalva que as tarefas não vão entrar nessas etapas ao mesmo tempo. A Figura 9, mostra que, se as tarefas começaram a serem enviadas no tempo $0s$, a tarefa 2 terminou a etapa de envio aos $24s$ e já começou a processar enquanto a tarefa 1 só vai começar a processar 8 segundos depois. Nesse período a tarefa 2 teve todo o poder de processamento do servidor à sua disposição, mas a partir do tempo $32s(24s + 8s)$ ela teve que dividir o poder de processamento do servidor com a tarefa 1.

Na Figura 11 nota-se que foi atribuído um instante t para cada mudança que ocorrem nas tarefas, como o inicio das tarefas num instante t_0 e o término da tarefa 2 no tempo t_1. Em cada intervalo as tarefas podem estar em algum dos estados (sendo enviadas ao servidor, processando ou sendo respondidas). Pode-se representar matematicamente isso através de algumas matrizes binárias, onde tem-se 1 para a tarefa ativa naquele estado e 0 caso não.

Desse modo, analisando a Figura 12, é possível descobrir quais tarefas estão em um determinado estado e o tempo que cada uma levará para terminar. Essa métrica será importante para determinar o tamanho do intervalo e quais tarefas passarão para um próximo estado. Por exemplo, para saber quantas tarefas estão na etapa de envio em um determinado intervalo de tempo, basta somar todos os números das colunas das tabelas X daquele intervalo.

Agora será mostrado um problema mais complexo. Nesse problema tem-se o envio de tarefas por mais um de um cliente e servidores diferentes. Para isso, precisa-se

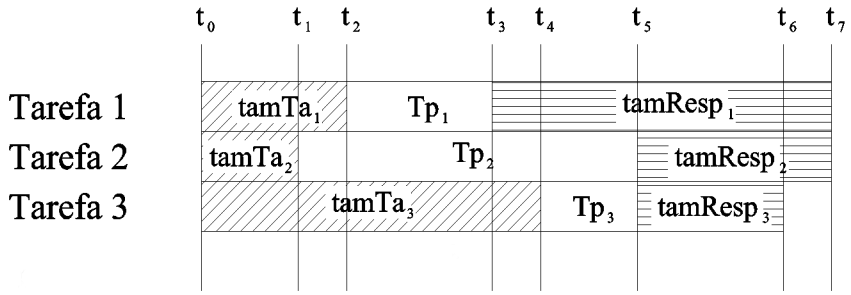


Figura 11 – Tempo de envio

Fonte: autoria própria

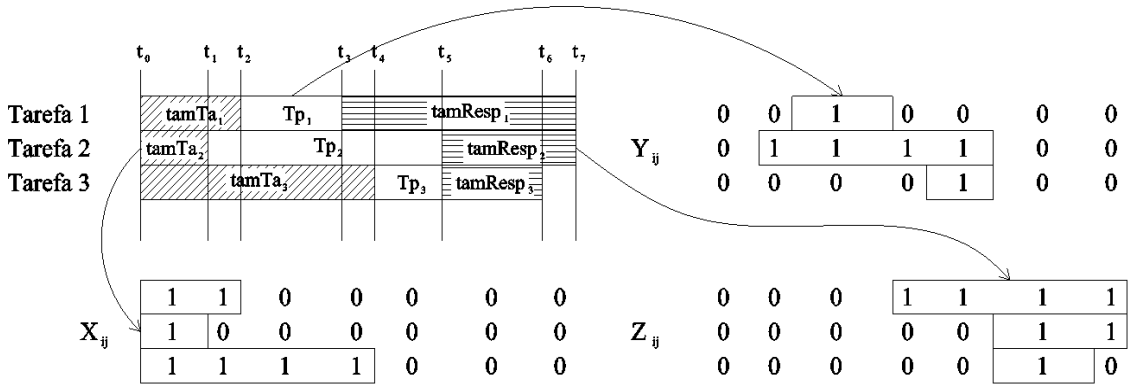


Figura 12 – Tarefas ativas

Fonte: autoria própria

coletar algumas informações importantes para resolver o problema. Supondo agora, um problema com 3 clientes, 5 tarefas e 2 servidores. A Tabela 1 mostra o número de clientes e as tarefas enviadas por eles. O C_0 (Cliente 0) por exemplo, tem as tarefas ta_2 e ta_4 para serem enviadas. Na segunda linha da quarta coluna o 1 indica que a tarefa é enviada pelo cliente C_0. Assim como na segunda linha da sexta coluna e 0 para as demais colunas o não envio da tarefa pelo cliente.

Cliente	ta_0	ta_1	ta_2	ta_3	ta_4
C_0	0	0	1	0	1
C_1	1	0	0	0	0
C_2	0	1	0	1	0

Tabela 1 – Cliente-Tarefa

A Tabela 2, tem-se o tamanho das tarefas , o número de instruções do processamento ou poder de processamento e o tamanho da resposta enviada pelo servidor.

Na Tabela 3, mostra a largura de banda de envio pelo cliente e largura de banda de recebimento pelo cliente .

	ta_0	ta_1	ta_2	ta_3	ta_4
Tamta	2	4	2	9	3
NInst	10	60	20	10	10
TamResp	4	1	2	1	9

Tabela 2 – Recursos

	C_0	C_1	C_2
LBeC	1	2	1
LBrC	4	8	1

Tabela 3 – Recursos-Cliente

Na Tabela 4, tem-se a largura de banda de envio do servidor , número de instrução do servidor . Para ser mais preciso, seria a quantidade de instruções que o computador consegue executar dentro de um segundo (KUROSE; ROSS., 2006) e a lagura de banda de reposta pelo servidor .

	S_0	S_1
LBeS	8	16
NInstS	10	20
LBrS	8	8

Tabela 4 – Recursos-Servidor

A Tabela 5, mostra a alocação das tarefas nos servidores. Com 1, a tarefa é enviada pelo servidor, com 0, a tarefa não é enviada pelo servidor.

	ta_0	ta_1	ta_2	ta_3	ta_4
$Serv_0$	1	0	0	1	0
$Serv_1$	0	1	1	0	1

Tabela 5 – Tarefa-Servidor

Conhecendo agora todos os recursos disponíveis, pode-se agora dar início ao processo de envio, identificando-se o cliente que está enviando cada tarefa. Com o objetivo do calculo do tempo de envio inicialmente pretende-se determinar num instante de tempo zero (t_0) o mínimo entre os tempos de envio dessas tarefas. Já conhecido o Tamta, precisa-se verificar o número de tarefas enviadas pelo cliente , assim tem-se na Figura 13 o tamanho de cada tarefa e o cliente que envia cada tarefa. Pode-se notar que o cliente C_0 envia duas tarefas, assim como o C_2. O C_1 envia apenas uma tarefa.

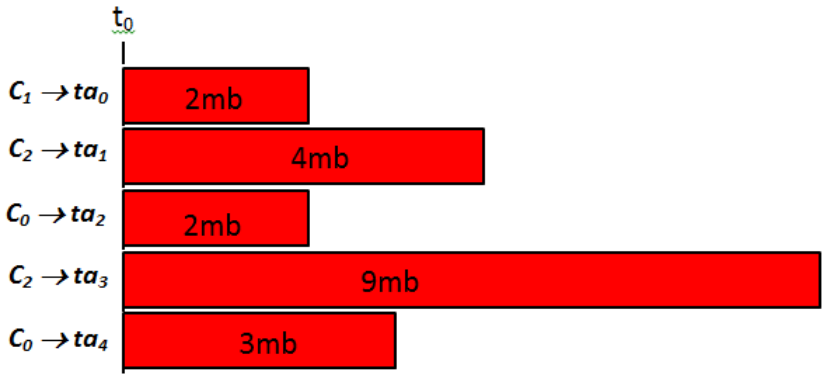


Figura 13 – Tamanho das tarefas

Fonte: autoria própria

Esse cálculo do tempo mínimo entre as tarefas se dá através de uma formulação capaz de identificar quem será a primeira tarefa a ser enviada no tempo t_1 após identificar o cálculo do tempo mínimo identificando o tamanho da tarefa, o número de tarefas enviadas pelo cliente dividido pela largura de bandas enviadas pelo cliente.

$$te_n = \frac{Tamta_i * nTaeC_j}{LBeC_j} \tag{5.5}$$

Com $n = \{0, 1, 2, ...N\}$, $i = \{0, 1, 2, ...I\}$ e $j = \{0, 1, 2, ...J\}$. Assim analisando a primeira tarefa tem-se a ta_0 com $Tamta = 2$ e está sendo enviada pelo C_1 e no momento, só a uma tarefa sendo executada e será dividido pela largura de banda do C_1.

$$te_0 = \frac{2*1}{2} = 1s$$

A tarefa ta_1 está sendo enviada pelo cliente C_2 com $Tamta = 4$ e está sendo enviada duas tarefas pelo C_2, onde serão divididos pela largura de banda do cliente C_2.

$$te_1 = \frac{4*2}{1} = 8s$$

Esse processo precisa ser verificado com todas as tarefas, encontrando assim seus tempos. Com os resultados obtidos, precisa-se verificar o mínimo dos tempos. Neste caso, o tempo da tarefa ta_0 foi o menor de todos sendo igual a 1. Com o tempo mínimo encontrado, pode-se afirmar que a t_0 levou 1 segundo para ser enviado e que agora, será dado o inicio ao processamento como mostra a Figura 14.

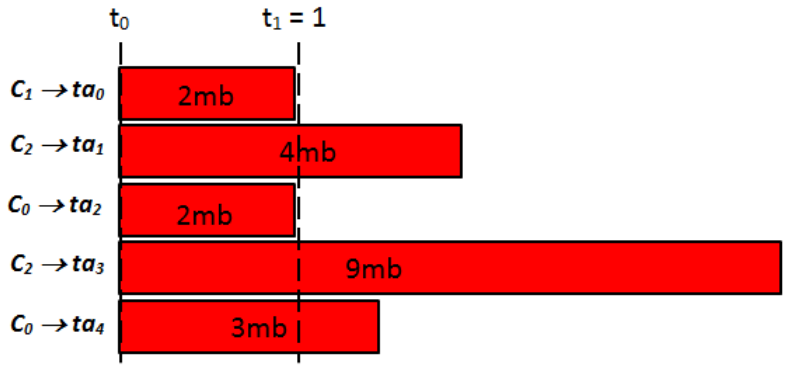


Figura 14 – Envio das tarefas por cada cliente

Fonte: autoria própria

A prova de que a ta_0 foi enviada se dá quando subtrair o tamanho da tarefa pelo o que foi enviado.

$$Tamtata_i = Tamta_i - \frac{t_{en} * LBeC_j}{nTaeC_j} \tag{5.6}$$

Na tarefa ta_0 tem-se que:

$$Tamtata_0 = 2 - \frac{1*2}{1} = 0$$

Com esse resultado prova-se que a ta_0 foi enviada e dará início ao processamento. Precisa-se agora, verificar o quanto foi enviado das outras tarefas no tempo de $t_1 = 1$ segundo. Na ta_1 tem-se:

$$Tamtata_1 = 4 - \frac{1*1}{2} = 3.5mb$$

Esse cálculo deverá ser feito com todas as tarefas. Daí, tem-se uma Tabela 6 com novos valores para as tarefas. Observe que nessa nova tabela de acordo com os resultados encontrados, a ta_0 já foi totalmente enviada. É importante destacar que a tarefa ta_2 tinha o mesmo tamanho da tarefa ta_0, mas a ta_0 é enviado pelo cliente C_1 e a ta_2 pelo cliente C_0. O C_1 tem a $LBeC = 2$ e o C_0 tem o $LBeC = 1$. O envio do C_1 foi bem mais rápido devido a largura de banda dos clientes serem diferentes.

	ta_0	ta_1	ta_2	ta_3	ta_4
Tamta	0	3.5	1.5	8.5	2.5

Tabela 6 – Tamanho das tarefas

Como mostra a Figura 15, é importante identificar a tarefa que vai ser processada, no caso aqui a ta_0, no qual, precisa-se calcular novamente o tempo de envio e de processamento.

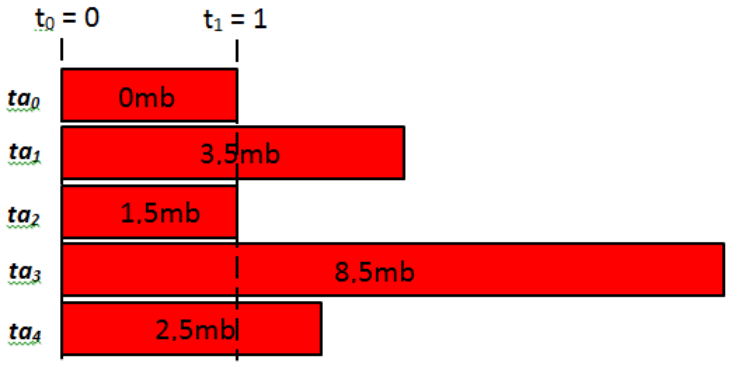


Figura 15 – Envio da primeira tarefa

Fonte: autoria própria

Para determinar o tempo de processamento precisa-se do número de instrução *NInst* vezes o número de tarefas sendo processado pelo servidor *nTProcS* dividido pelo número de instruções do servidor *NInstS*. Assim tem-se:

$$t_p = \frac{NInst * nTProcS}{NInstS} \tag{5.7}$$

Como neste momento apenas a ta_0 está sendo processada tem-se:

$$t_{p0} = \frac{10*1}{10} = 1s$$

Continuando agora com o envio das tarefas precisa-se calcular o tempo novamente. Por exemplo, com a ta_1:

$$t_{en} = \frac{Tamta_i * nTaeC_j}{LBeC_j}$$

$$t_{e1} = \frac{3,5*2}{1} = 7s$$

E esse calculo do tempo é feito com todas as tarefas. Agora determina-se o mínimo entre os tempos incluindo o processamento que no caso foi t_2 = 1s. Portanto o mínimo é 1s. Assim pode-se determinar um valor para o processamento da ta_1 utilizando a fórmula:

$$NInsta_i = NInst - \frac{t_p * NInstS}{NtaProcS} \tag{5.8}$$

Onde tem-se a ta_0 igual a:

$$NInsta_0 = 10 - \frac{1*10}{1} = 0$$

Com esse resultado percebe-se que a tarefa ta_0 foi totalmente processada, mas precisa-se determinar novos valores para as tarefas que ainda não foram enviadas com a formula:

$$Tamtata_i = Tamta_i - \frac{t_{en} * LBeC_j}{nTaeC_j}$$

Assim tem-se que t_1 é igual a:

$$Tamtata_1 = 3,5 - \frac{1*1}{2} = 3mb$$

Com esses novos resultados, tem-se uma Figura 16 que mostra a tarefa ta_0 processada e os valores que ainda faltam serem enviados.

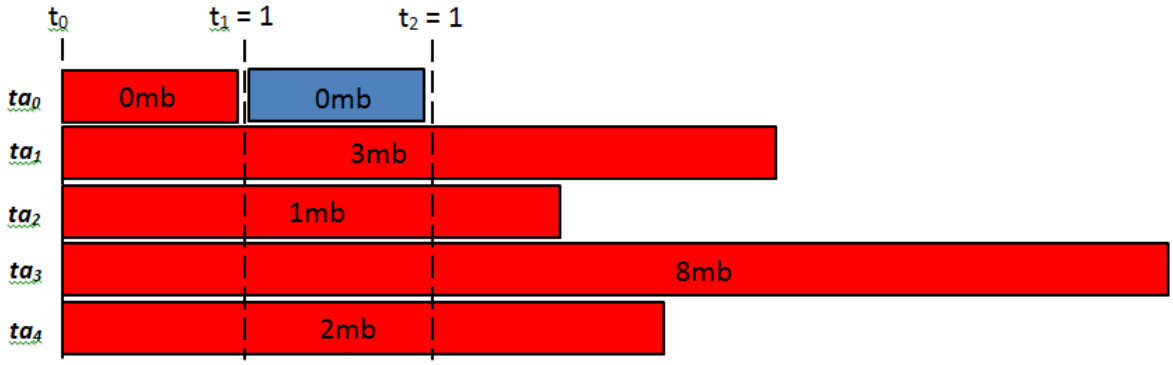


Figura 16 – Processamento da tarefa ta_0

Fonte: autoria própria

Nesse momento, a tarefa ta_0 aguarda uma resposta do servidor informando que o envio da tarefa foi concluída totalmente. Para calcular o tempo dessa resposta é necessário determinar:

$$TResp_i = \frac{TamtaResp * NTaresC}{LBrC} \tag{5.9}$$

Assim calculando o tempo de resposta tem-se:

$$TResp_0 = \frac{4*1}{8} = 0,5s$$

Agora precisa-se calcular o tempo de envio de todas as tarefas restantes e identificar o mínimo entre todos os tempos, tem-se $t_3 = 0,5s$. Precisa-se determinar o quanto foi enviado pelas tarefas e o tamanho da resposta utilizando a seguinte fórmula:

$$NTamRespta_i = TamRespta_i - \frac{TResp * LBrC}{nTarC}$$

Assim tem-se:

$$NTamResp ta_0 = 4 - \frac{0{,}5*8}{1} = 0$$

Com esse novo resultado do tamanho da resposta, concluí-se na Figura 17 que a ta_0 foi totalmente enviada no $t_3 = 0,5$. Agora continua-se com o envio das outras tarefas calculando o seu novo tamanho.

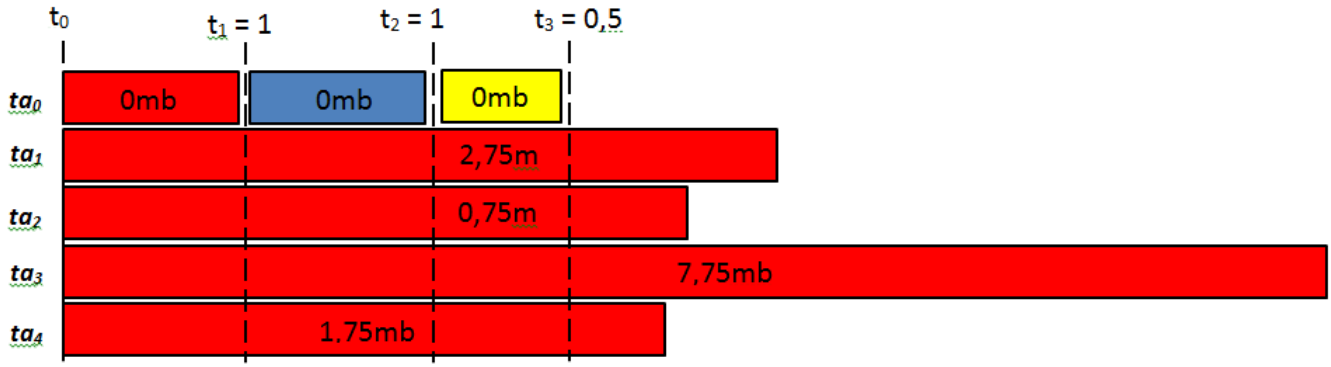


Figura 17 – Resposta da ta_0

Fonte: autoria própria

Nesse momento, na Figura 18 o C_1 não está mas enviando nenhuma tarefa. É importante verificar quantas tarefas o cliente está enviando, pois isso, pode liberar mais recurso para o envio. As tarefas ta_1, ta_2, ta_3 e ta_4 continuam sendo enviadas e determinando o seu tempo mínimo. O tempo mínimo encontrado é de $t_4 = 1,5s$ e com o novo tamanho das tarefas. E a ta_2 foi enviada.

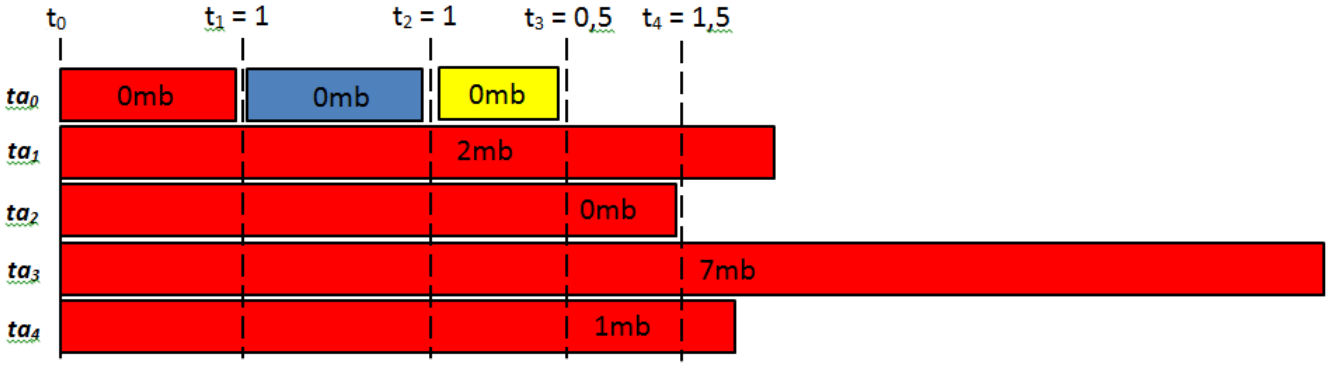


Figura 18 – Envio da tarefa ta_2

Fonte: autoria própria

Na Figura 19 a tarefa ta_2 foi enviada e foi dado o início o processamento, calculando o tempo mínimo em $t_5 = 1s$ e com novo tamanho das tarefas. Essa tarefa está sendo enviada pelo cliente C_0.

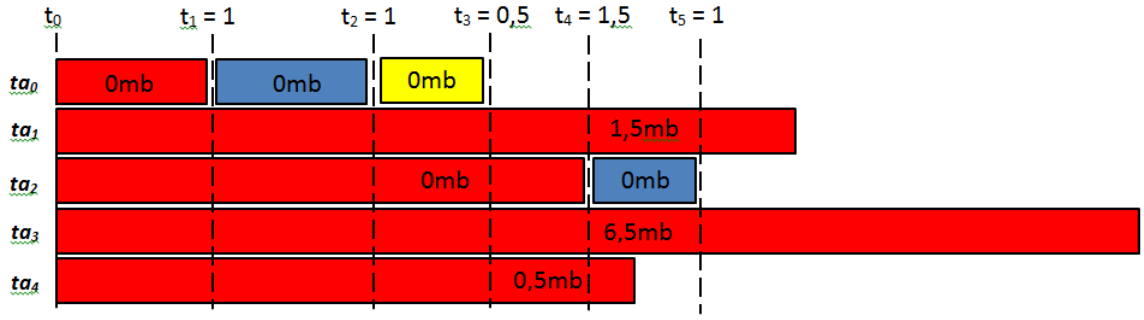


Figura 19 – Processamento da tarefa ta_2

Fonte: autoria própria

Na Figura 20 foi enviado a resposta da tarefa ta_2 num tempo de $t_6 = 0,25s$, com isso a tarefa ta_2 foi enviada totalmente. O C_0 como mostra a Tabela 1, enviava as tarefas ta_2 e ta_4, como a tarefa ta_2 foi totalmente enviada, o número de tarefas enviadas pelo cliente irá mudar de 2 para 1 e que todos os recursos do C_0 se voltam para a tarefa ta_4, que foi enviada e dará início ao processamento na Figura 21.

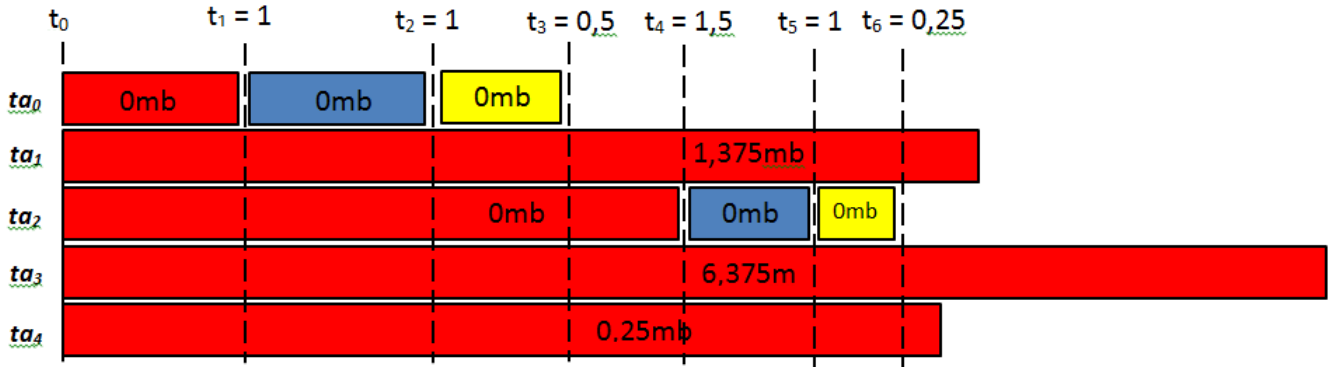


Figura 20 – Resposta da tarefa ta_2

Fonte: autoria própria

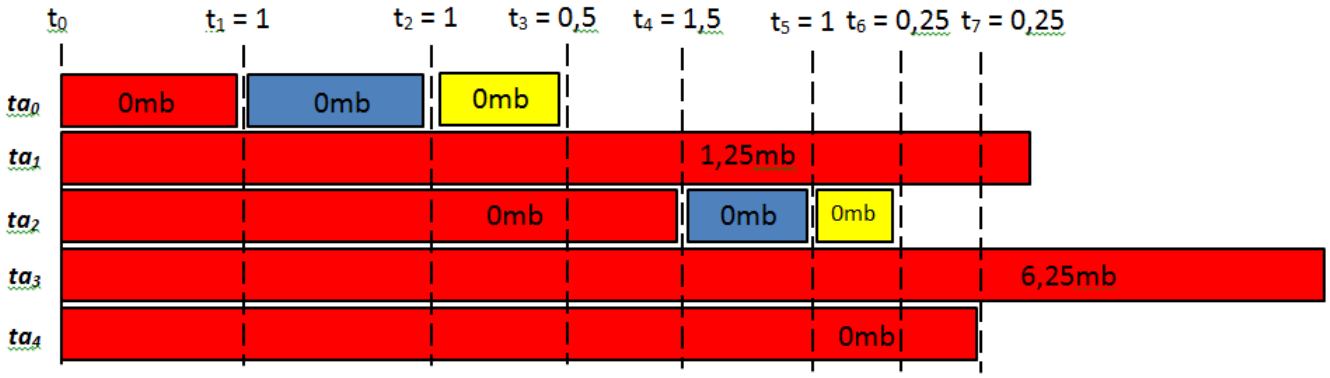


Figura 21 – Envio da tarefa ta$_4$

Fonte: autoria própria

Na Figura 22 a tarefa ta_1 foi enviada para ser processada e a tarefa ta_4 foi processada e aguardando a resposta pelo servidor no tempo mínimo de $t_8 = 2,5s$. Voltando a calcular os tempos mínimos, na Figura 21 a tarefa ta_1 continua sendo processada, a tarefa ta_3 sendo enviada e a tarefa ta_4 foi respondida e concluída.

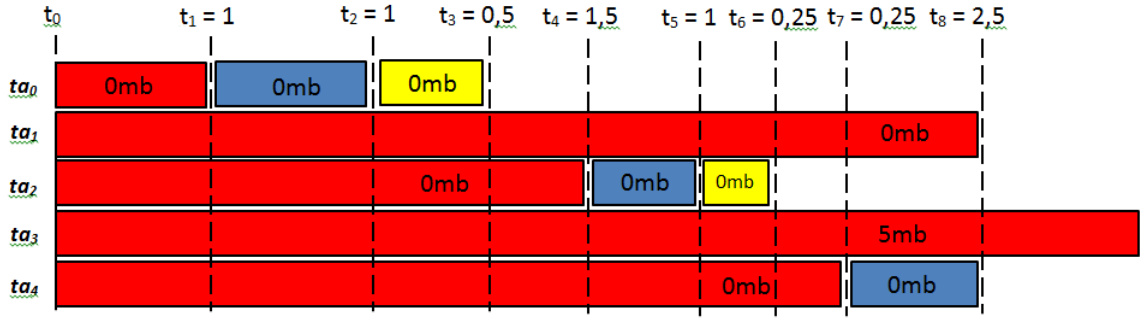


Figura 22 – Processamento da tarefa ta$_4$

Fonte: autoria própria

Nesse momento, na Figura 23 estão acontecendo três atividades diferentes simultaneamente, o envio, o processamento e o envio da resposta. Esse é também um diferencial em nosso trabalho, pois não está esperando todas atividades serem enviadas para dar início ao processamento e ao envio da resposta pelo servidor. Pode-se definir essa situação como sendo um sistema heterogêneo. Calcula-se o tempo mínimo e encontra-se o $t_9 = 2,25$

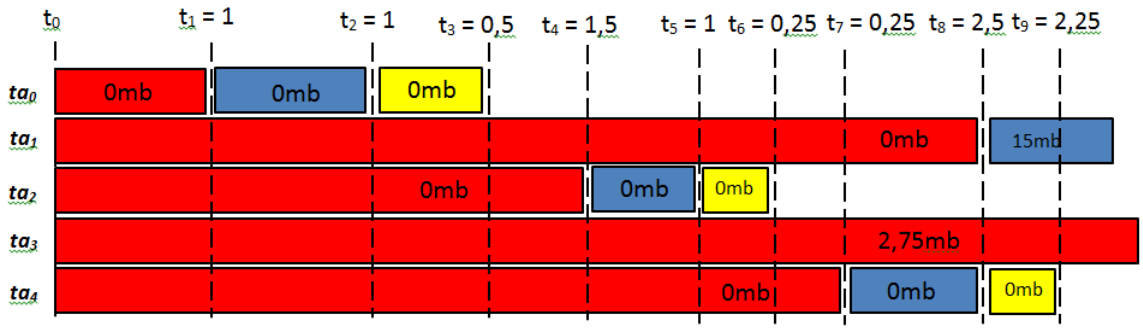


Figura 23 – Resposta da tarefa ta_4

Fonte: autoria própria

Nos tempos $t_{10} = 0,75s$ e $t_{11} = 1s$ da Figura 24, mostra a tarefa ta_1 sendo processada e recebendo a resposta pelo servidor. Com isso, finaliza-se totalmente a tarefa ta_1.

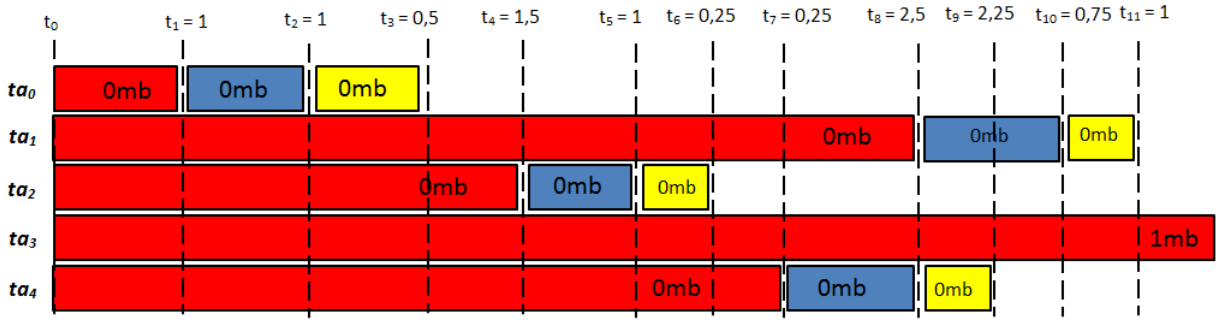


Figura 24 – Resposta da tarefa ta_1

Fonte: autoria própria

O cliente C_2 enviava as tarefas ta_1 e ta_3. Restando apenas a tarefa ta_3, todos os recursos disponíveis se voltam para a ta_3. Na Figura 25, com o tempo mínimo $t_{12} = 1$ de envio, com o tempo de processamento $t_{13} = 1$ e o recebimento da resposta no tempo $t_{14} = 1$, verificam-se que a tarefa ta_3 é totalmente enviada. Finalizando assim, todo o processo de envios das tarefas.

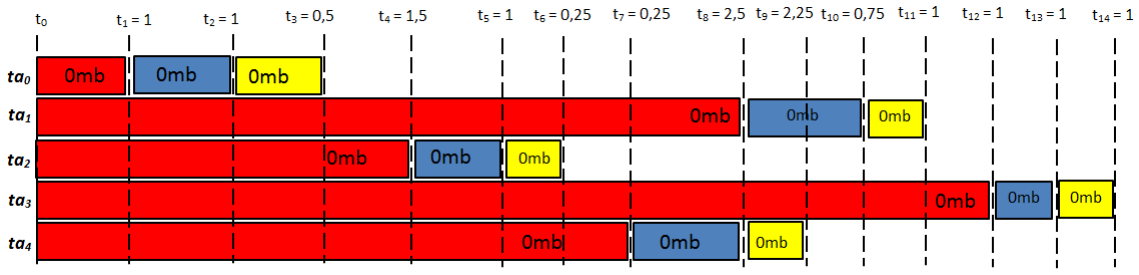


Figura 25 – Envio-Processamento-Resposta de todas as tarefas

Fonte: autoria própria

Agora precisa-se determinar o tempo total (T_{total}) de envio dessas tarefas, que será determinado pelo somatório de todos os t_n.

$$T_{total} = \sum_{t_0}^{t_n} \tag{5.10}$$

$$\forall t \in \{0, 1, 2, ...n\} \text{ com } t_n \geq 0$$

Desta forma, tem-se uma variação de t_n com n variando em que $\in \{0, 1, 2, ...14\}$ com $t_n \geq 0$, como mostra a Figura 23.

$$T_{total} = \sum_{t_0}^{t_{14}} = 15s$$

Assim, chega-se ao resultado final, que tem como objetivo atender o cliente no menor tempo possível, identificando várias variáveis levando em consideração o número de clientes, o número de servidores, o Tamta, NInsta e TamResp e determinando o seu somatório. Chega-se a conclusão da formula:

$$t_n = min\left(\frac{Tamta_{ij} * nTaeC_{ki}}{LBeC_k} + \frac{NInst * nTproc_{si}}{P_sS_s} + \frac{TamResp_{ji} * NTaresC_{ki}}{LBrC_k}\right) \tag{5.11}$$

$$\text{com } LBeC_k, P_sS_s, LBrC_k \geq 0$$

Com a formulação do problema apresentado, foi possível determinar um modelo matemático (5.11) que fornece valores mais precisos com relação ao tempo de envio dessas tarefas. Esse tempo mínimo, permite um estudo mais detalhado para o escalonamento de tarefas, com o objetivo de atender o cliente o mais rápido possível. É preciso antes de mais nada, fazer um levantamento prévio dessas variáveis. No primeiro exemplo mostrado, a preocupação é de apenas um cliente tornando-se o calculo bem mais simples. Quando se trabalha um número elevado de n clientes, n tarefas e n servidores o cálculo se torna bem mais complexo. Por isso o motivo de se implementar o algoritmo e obter resultados.

6 RESULTADOS

Este capítulo tem como objetivo apresentar os resultados encontrados pelo modelo aqui proposto, bem como compará-los com alguns trabalhos já desenvolvidos. Também foram feitos testes com instâncias criadas aleatoriamente, visando um aumento na confiabilidade da modelagem proposta.

Por se tratar de uma formulação matemática utilizada no problema de escalonamento de tarefas em computação em nuvem, foi necessário a transcrição da mesma para uma linguagem de programação com o objetivo de coletas de dados. Essa implementação foi feita utilizando a linguagem de programação Java.

Na literatura é possível encontrar alguns algoritmos capazes de realizar esse escalonamento, sendo esses de forma aproximada, como exemplo, algoritmos genéticos, visto a necessidade da agilidade no momento da escolha. Porém, por se tratar somente da validação da modelagem, e assim não tendo a necessidade de uma melhor performance, foi desenvolvido um escalonador exato, ou seja, um algoritmo que de acordo com o valor retornado pela formulação, faz o escalonamento para obter a melhor solução possível.

Após a implementação, foi dado início aos testes, bem como a coleta dos primeiros resultados. Para esses testes iniciais, foram utilizadas valores aleatórias, como número de clientes, números de servidores, número de tarefas, largura de banda de envio e recebimento. Define-se também a quantidade de tarefas que cada cliente esta enviando e os recursos que cada tarefa precisa e também os recursos do servidor. Visando apenas a verificação do funcionamento correto do código, como apresentado na Figura 26.

São instanciados os 3 clientes com a largura de banda de envio e de recebimento do cliente, 4 servidores com a largura de banda de envio do servidor, com a largura de banda recebimento do servidor, tamanho da memória do servidor, o poder de processamento do servidor e os recursos disponíveis no servidor e 7 tarefas com a identificação de cada cliente, com o tamanho da tarefa, o tamanho necessário de memoria no servidor, tempo de processamento da tarefa no servidor e os recursos que ela necessita.

Nos primeiros testes foi verificada a precisão do algoritmo, que com os valores instanciados, retornou como melhor sequencia a Figura 27. Nela, verifica-se as tarefas que foram alocadas nos servidores no seu menor tempo final. Para confirmar esse resultado foi feito um teste de mesa.

Com a implementação já validada, foram selecionados trabalhos entre os já pesquisados, como por exemplo (WAN; ALMEIDA, 2012), além de novos trabalhos (XUE et al., 2014), na tentativa de validar a modelagem aqui proposta. Porém, as

```
public class principal{
    public static void main(String[] args){
        int i, j, k, l;

        Cliente clientes[] = {
            new Cliente(256, 2048),
            new Cliente(512, 4096),
            new Cliente(128, 512)
        };

        Servidor servidores[] = {
            new Servidor(40000, 40000, 524288, 200, new int[]{0,2,4}),
            new Servidor(30720, 30720, 1048576, 400, new int[]{1,3,4}),
            new Servidor(30720, 30720, 1572864, 800, new int[]{0,1,4}),
            new Servidor(30720, 30720, 2097152, 100, new int[]{0,1,2,4})
        };

        Tarefa tarefas[] = {
            new Tarefa(0, 4096, 4096, 20, 500, new int[]{0,2}),
            new Tarefa(0, 65536, 8192, 100, 100, new int[]{1}),
            new Tarefa(1, 8192, 20, 300, 500, new int[]{0,2}),
            new Tarefa(1, 4096, 4096, 20, 500, new int[]{1,2}),
            new Tarefa(1, 4096, 4096, 20, 500, new int[]{3,4}),
            new Tarefa(2, 4096, 4096, 20, 500, new int[]{0,4}),
            new Tarefa(2, 1024, 1054, 20, 1000, new int[]{2,4})
        };
```

Figura 26 – Dados iniciais para testes

Fonte: autoria própria

```
Tarefa 0 ->  - Servidor 3
Tarefa 1 ->  - Servidor 2
Tarefa 2 ->  - Servidor 3
Tarefa 3 ->  - Servidor 3
Tarefa 4 ->  - Servidor 1
Tarefa 5 ->  - Servidor 3
Tarefa 6 ->  - Servidor 3
   intervalo -  tempo total -    tarefa  0    tarefa  1    tarefa  2    tarefa  3    tarefa  4    tarefa  5    tarefa  6

      0,0000 -       0,0000 -    4096,0000   65536,0000    8192,0000    4096,0000    4096,0000    4096,0000    1024,0000
     16,0000 -      16,0000 -    2048,0000   63488,0000    5461,3333    1365,3333    1365,3333    3072,0000    1000,0000
      8,0000 -      24,0000 -    1024,0000   62464,0000    4096,0000       0,0000       0,0000    2048,0000     200,0000
      0,0000 -      24,0000 -    1024,0000   62464,0000    4096,0000     500,0000     500,0000    2048,0000     200,0000
      1,2500 -      25,2500 -     864,0000   62304,0000    3456,0000     437,5000    4096,0000    1888,0000     137,5000
      1,0000 -      26,2500 -     736,0000   62176,0000    2944,0000     387,5000       0,0000    1760,0000      87,5000
      1,7500 -      28,0000 -     512,0000   61952,0000    2048,0000     300,0000       0,0000    1536,0000    1054,0000
      2,0586 -      30,0586 -     248,5000   61688,5000     994,0000      94,1406       0,0000    1272,5000       0,0000
      0,9414 -      31,0000 -     128,0000   61568,0000     512,0000    4096,0000       0,0000    1152,0000       0,0000
      1,0000 -      32,0000 -     500,0000   61312,0000       0,0000    2048,0000       0,0000    1024,0000       0,0000
      0,0000 -      32,0000 -     500,0000   61312,0000     500,0000    2048,0000       0,0000    1024,0000       0,0000
      1,0000 -      33,0000 -     450,0000   61056,0000     450,0000       0,0000       0,0000     896,0000       0,0000
      7,0000 -      40,0000 -     100,0000   59264,0000     100,0000       0,0000       0,0000     500,0000       0,0000
      3,0000 -      43,0000 -    4096,0000   58496,0000      20,0000       0,0000       0,0000     200,0000       0,0000
      0,0146 -      43,0146 -    4066,0000   58492,2500       0,0000       0,0000       0,0000     198,5352       0,0000
      1,9854 -      45,0000 -       0,0000   57984,0000       0,0000       0,0000       0,0000       0,0000       0,0000
      0,0000 -      45,0000 -       0,0000   57984,0000       0,0000       0,0000       0,0000    4096,0000       0,0000
     16,0000 -      61,0000 -       0,0000   53888,0000       0,0000       0,0000       0,0000       0,0000       0,0000
    210,5000 -     271,5000 -       0,0000     100,0000       0,0000       0,0000       0,0000       0,0000       0,0000
      0,1250 -     271,6250 -       0,0000    8192,0000       0,0000       0,0000       0,0000       0,0000       0,0000
      8,0000 -     279,6250 -       0,0000       0,0000       0,0000       0,0000       0,0000       0,0000       0,0000
      0,0000 -     279,6250 -       0,0000       0,0000       0,0000       0,0000       0,0000       0,0000       0,0000
279.625
```

Figura 27 – Melhor Resultado

Fonte: autoria própria

formulações utilizadas nestas pesquisas, usam quantidade de variáveis diferentes, impossibilitando comparações justas, como é o caso de (COUTINHO; DRUMMOND; FROTA, 2014). O trabalho propõe o problema de gestão de recursos que seleciona recursos de nuvem com o objetivo de reduzir o custo do aluguel do serviço e o tempo

de execução de aplicativos do usuário. Não deixa claro em sua formulação quando se utiliza apenas uma variável K vezes um tempo t_m.

Porém, apesar de terem quantidade de variáveis diferentes, esses trabalhos tem algo em comum, que é o fato de só levarem em consideração, nas formulações, o tempo gasto para o processamento das tarefas nos servidores, deixando de lado o tempo gasto para enviar e receber as mesmas. Sendo assim, visando comparações mais justas, foram simulados ambientes, que retratam a realidade em diferentes situações, com diferentes valores, como por exemplo a simulação de ambientes homogêneos e heterogêneos.

Na tentativa de gerar ambientes cada vez mais fieis à realidade, de forma ágil e garantindo a imparcialidade, foi implementado um gerador de instâncias. Para gerar as instâncias, inicialmente é passado apenas a configuração do ambiente, sendo o mesmo responsável por gerar automaticamente as instâncias, assim garantindo a imparcialidade.

Para alcançar o objetivo de gerar instâncias que simulem diferentes situações, o gerador manipulou alguns parâmetros que interferem diretamente no resultado, como quantidade de clientes, de servidores e tarefas, nível de aleatoriedade, domínio de valores, dentre outros. Sendo ao todo gerado um conjunto de 360 instâncias em 36 classes diferentes. Foram estipulados valores aleatórios para demonstrar a heterogeneidade do ambiente.

O primeiro parâmetro a ser levado em consideração é a quantidade de clientes, servidores e tarefas, presentes em cada simulação. Foram geradas 180 situações em que as tarefas eram atendidas partindo de 5 clientes diferentes, e outras 180 em que eram levados em consideração 10 clientes. Com relação a quantidade de servidores disponíveis, as instâncias simulam em duas quantidades diferentes, sendo 180 com 3 máquinas, e outras 180 com 5. Por fim, as tarefas aparecem variando em três situações diferentes: 120 atendendo 5 tarefas, 120 utilizando 7, e outras 120 com 10 requisições.

O segundo parâmetro modificado pelo gerador é domínio de valores que cada variável pode assumir. Foram estipulados valores mínimos que cada um parâmetro da formulação poderia assumir, determinando assim, o valor mínimo dessas variáveis, e partindo desses valores multiplicados pela variação, é estipulado o máximo que a respectiva variável pode atingir. Por exemplo, o valor mínimo determinado para a largura de banda do cliente sendo 512, assim, de acordo com a variação, que em todas as situações vai variar em 2, 4 ou 8, essa variável poderá atingir valores de até 1024, 2048 e 4096, respectivamente. A Tabela 7 mostra a faixa de valores que cada variável poderá atingir a partir da coluna dos valores mínimos com destaque para suas variáveis. Os valores mínimos utilizados são baseados em valores de provedores reais.

A variação das instâncias permite simular valores diferentes para cada ambiente. Uma Largura de Banda do Cliente tem seu valor mínimo multiplicado por 2, 4 e 8 obtendo uma nova largura de Banda do Cliente com 1.024, 2.048 e 4.096 de memoria.

Apresentando novos valores para cada variação com cada variável.

	Mínimo	Variação 2	Variação 4	Variação 8
Largura de Banda do Cliente	512	1.024	2.048	4.096
Largura de Banda do Servidor	30.000	60.000	120.000	240.000
Memoria do Servidor	524.288	1.048.576	2.097.152	4.194.304
Número de Instrução do Servidor	200	400	800	1.600
Tamanho da Tarefa	30	60	120	240
Memória da Tarefa	524	1.048	2.096	4.192
Número de Instrução do cliente	20	40	80	160

Tabela 7 – Tabela de variação de instâncias

O terceiro parâmetro usado para alterar a simulação dos ambientes, foi a quantidade de recursos disponíveis em cada servidor. Esse parâmetro estipulou um valor limite de recursos que cada servidor poderia dispor, tendo seus valores variando entre um e cinco recursos. Por fim, de cada combinação diferente, foram geradas dez instâncias. Sendo usada a mesma quantidade de parâmetros, porém com valores diferentes.

De posse das instâncias foi dado início a coleta de resultados como mostra a Figura 28, onde o escalonador X com o modelo proposto, levando em consideração o envio, o processamento e a resposta, gera uma solução A, e o mesmo escalonador X com o modelo da literatura, que leva em consideração apenas o processamento, gera uma solução B. De posse das melhores soluções, foi calculado o tempo total gasto por cada uma delas, levando em consideração o envio, o processamento e o retorno das tarefas. Por fim, esses resultados foram comparados e armazenados, verificando o ganho obtido pela modelagem proposta em relação a presente na literatura.

Com esses dados, foi possível determinar resultados importantes para a pesquisa na forma de gráficos, os mesmos diferenciados pela variação das instâncias a partir dos valores mínimos com a variação das instâncias multiplicadas por 2, 4 e 8. Esses resultados foram comparados a partir de 10 instâncias para cada conjunto de tarefas e definido o melhor tempo de cada modelo.

A Figura 29 mostra o resultado da execução algoritmo com gráfico que representa o resultado do melhor tempo, do modelo proposto pelo trabalho, e o processamento, com barras que representam o algoritmo com 5 clientes, 3 servidores e 5, 7 e 10 tarefas, todas variando em 2 vezes nas instâncias. O modelo proposto teve um ganho satisfatório em todos os conjuntos de tarefas. Com 7 tarefas, por exemplo, o modelo proposto obteve o menor tempo com $0,5385088$s, enquanto a melhor solução encontrada pelo modelo baseado na literatura gastou $0,6089041$s.

O gráfico da Figura 30 apresenta uma situação em que é levado em consideração 10 clientes. Nessa situação o modelo proposto também foi melhor comparado com o da literatura. Por exemplo, com 10 tarefas o valor médio encontrado pelo modelo proposto

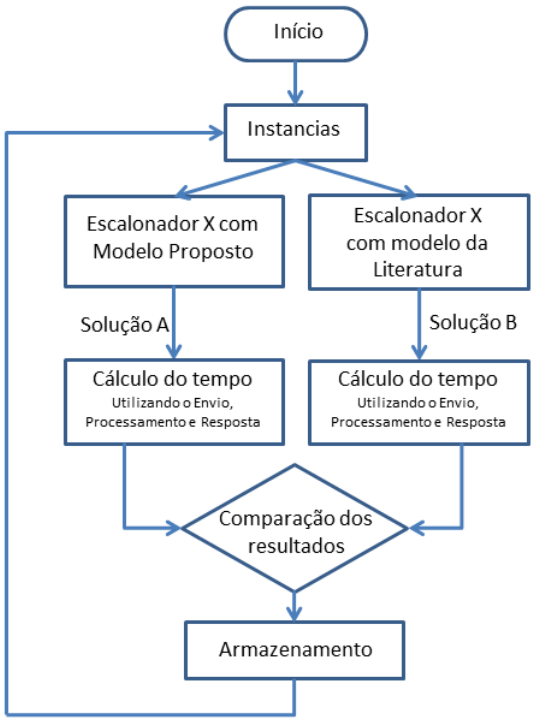


Figura 28 – Esquema mostrando a comparação entre o modelo matemático proposto e o modelo baseado na literatura

Fonte: autoria própria

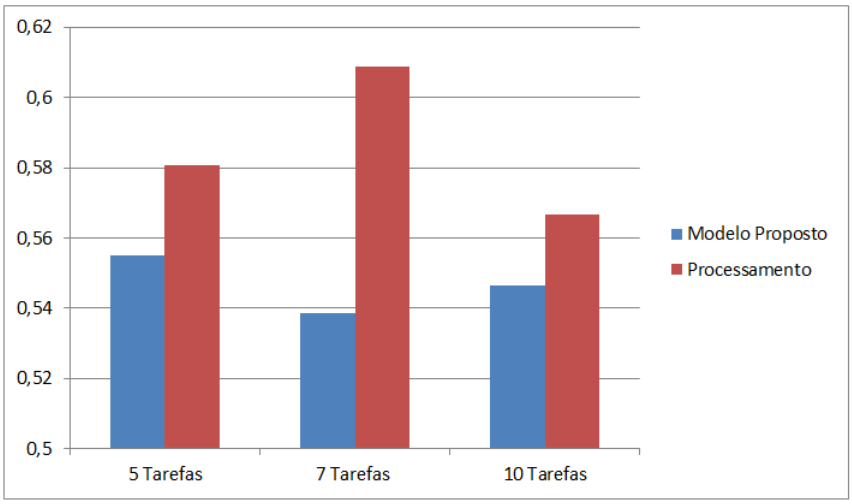


Figura 29 – 5 Clientes e 3 Servidores com instâncias variando em até 2 vezes

Fonte: autoria própria

foi $0,5464230$ enquanto que a média de valores o obtidos pelo modelo da literatura foi de $0,5767010$, obtendo assim uma melhora de 6%.

A Figura 31 e 32, os gráficos mostram resultados bastante próximos, mas o modelo proposto conseguiu se sair melhor. Com 5 Clientes, 5 Servidores e 5 Tarefas o

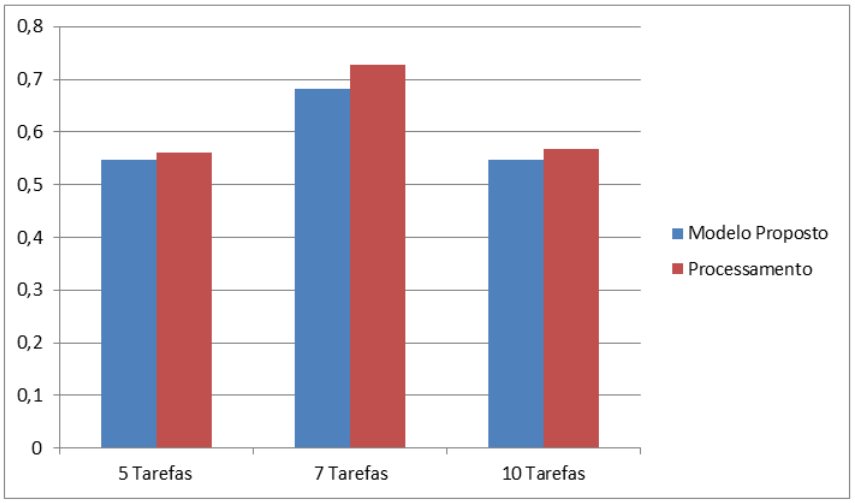


Figura 30 – 10 Clientes e 3 Servidores com variação de instâncias em até 2 vezes

Fonte: autoria própria

modelo conseguiu ser melhor nas 10 execuções conseguindo 100 porcento de ganho, ou seja, das 10 execuções em cada modelo ele conseguiu ser melhor em todos. Com 5 Clientes, 5 Servidores e 7 Tarefas o ganho também foi de 100 porcento. Com 5 Clientes, 5 Servidores e 10 tarefas das 10 execuções o modelo proposto só obteve ganho em duas execuções. Nas outras 8 execuções houve um empate nos tempos dos modelos.

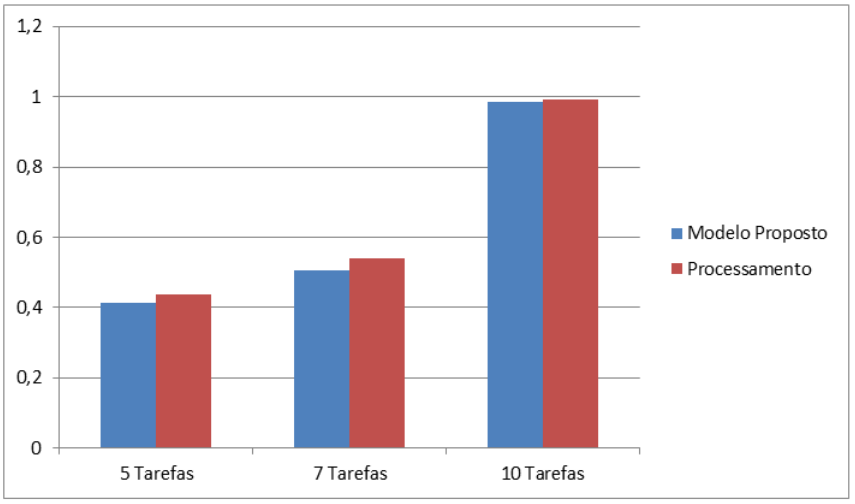


Figura 31 – 5 Clientes e 5 Servidores com variação de instâncias em até 2 vezes

Fonte: autoria própria

A Figura 33, o gráfico passa a ter uma variação diferente nas instâncias, com 5 clientes e 3 servidores com uma variação de instâncias em até 4 vezes, mas o modelo proposto conseguiu ser melhor com 5, 7 e 10 tarefas.

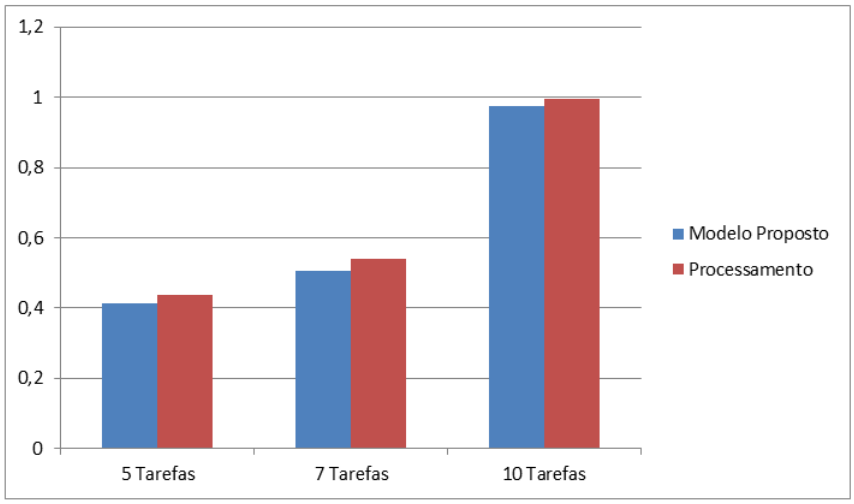


Figura 32 – 10 Clientes e 5 Servidores com variação de instâncias em até 2 vezes

Fonte: autoria própria

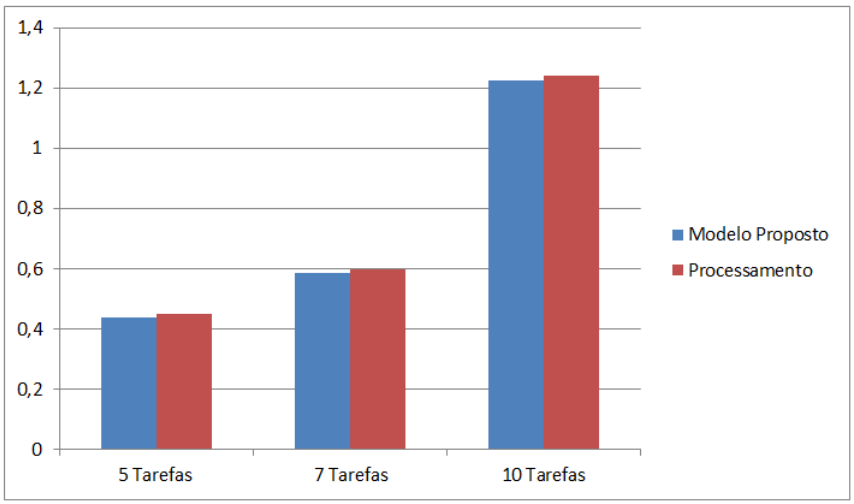


Figura 33 – 5 Clientes e 3 Servidores com variação de instâncias em até 4 vezes

Fonte: autoria própria

Já na Figura 34 com o número maior de clientes, o ganho aconteceu com 5 e 7 tarefas, mas com 10 tarefas o resultado não informou quem seria o melhor. Com 10 tarefas os tempos foram iguais. O tempo mínimo dos dois foi de $1,0920309$s, mas é interessante destacar que foram executadas 10 instâncias em cada modelo. Obtendo um ganho para cada modelo 5 execuções para cada modelo. Analisando as instâncias executadas percebeu-se que as tarefas que necessitam de poucos recursos faz com que os modelos tenderem ao mesmo tempo. Pois o número de recursos é muito pouco para determinar uma melhor combinação para o escalonador.

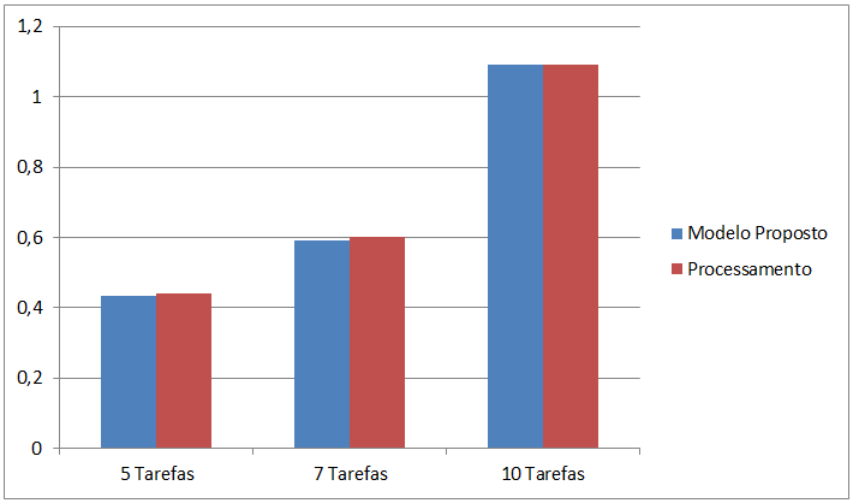


Figura 34 – 10 Clientes e 3 Servidores com variação de instâncias em até 4 vezes

Fonte: autoria própria

As Figuras 35 e 36 apresentam ganhos significativos nos tempos do modelo proposto. As Tabelas 8 e 9 apresentam os tempos encontrados com a diferença de seus resultados e com seus valores em porcentagem. Esses valores em porcentagem mostra o quanto o modelo proposto pelo trabalho foi melhor.

Clientes	Servidores	Tarefas	Modelo Prop.	Processamento	Dif.Tempo	%
5	5	5	0.4199420	0.4415582	0.0216162	4.89
5	5	7	0.508718	0.5612861	0.0525681	9.36
5	5	10	0.5975885	0.6357419	0.0381534	6

Tabela 8 – Variação de instâncias até 4 vezes com 5 clientes e 5 servidores

Clientes	Servidores	Tarefas	Modelo Prop.	Processamento	Dif.Tempo	%
10	5	5	0.4199420	0.4415582	0.0216162	4.89
10	5	7	0.5087185	0.5612861	0.0525676	9.36
10	5	10	0.5614546	0.5701256	0.008671	1.52

Tabela 9 – Variação de instâncias até 4 vezes com 10 clientes e 5 servidores

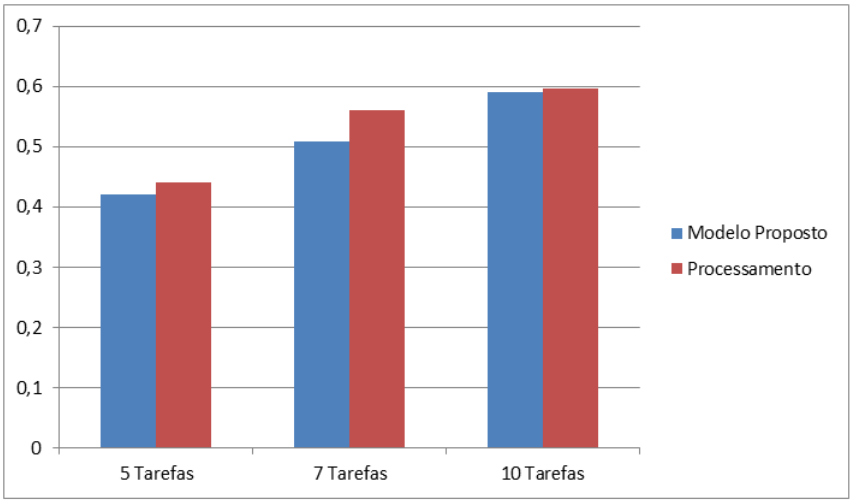


Figura 35 – 5 Clientes e 5 Servidores com variação de instâncias em até 4 vezes

Fonte: autoria própria

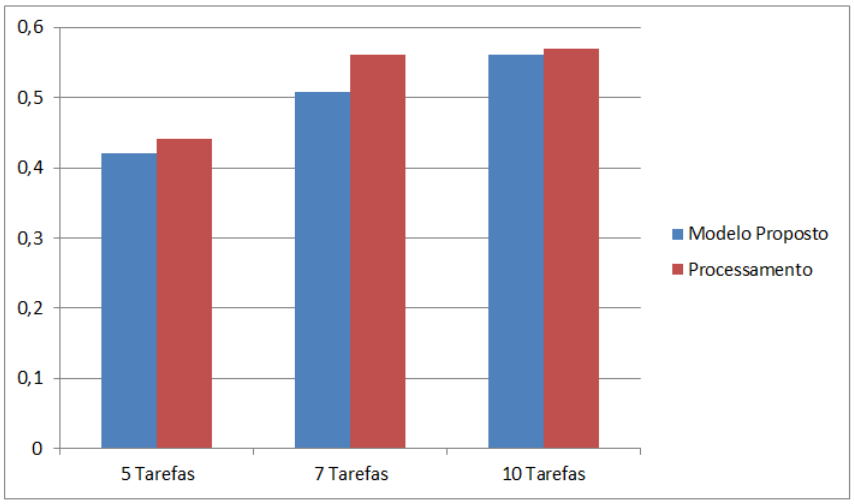


Figura 36 – 10 Clientes e 5 Servidores com variação de instâncias em até 4 vezes

Fonte: autoria própria

Com o gráfico da Figura 37, o modelo proposto se saiu bem em todos os resultados com a variação 8, com 7 tarefas o modelo proposto teve um aproveitamento de 100% no resultado.

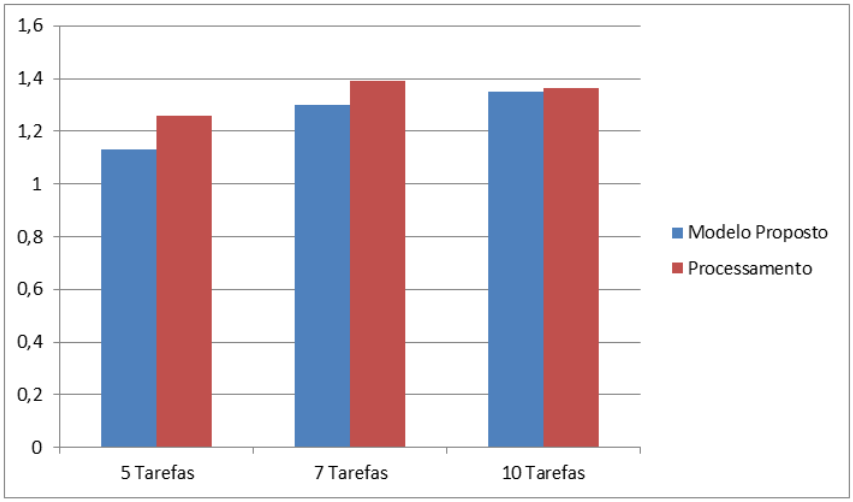


Figura 37 – 5 Clientes e 3 Servidores com variação de instâncias em até 8 vezes

Fonte: autoria própria

Com 10 clientes e 3 servidores, o gráfico da Figura 38 mostra uma média dos três melhores tempos do modelo proposto com as 5 tarefas com o tempo de 1.2077724 segundos , 7 tarefas com um tempo de 1.4003346 segundos e 10 tarefas com o tempo de 1.4205666 e com o modelo da literatura respectivamente os valores: 1.3353824, 1.4693524 e 1.4405666. O modelo proposto obteve um ganho em torno de 5.2%.

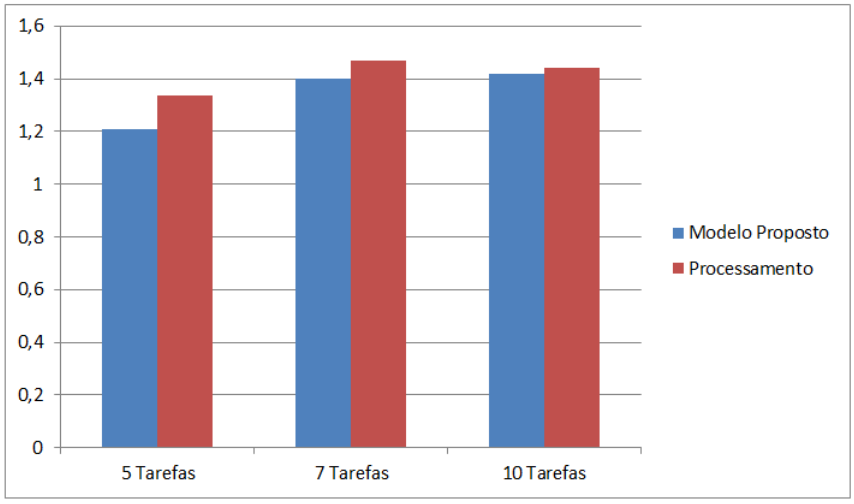


Figura 38 – 10 Clientes e 3 Servidores com variação das instâncias até 8 vezes

Fonte: autoria própria

A Figura 39, o gráfico também mostra o ganho do modelo proposto, mas com 10 tarefas no gráfico, o resultado aconteceu de forma que com 10 instâncias houve um empate nos tempos dos resultados em oito execuções.

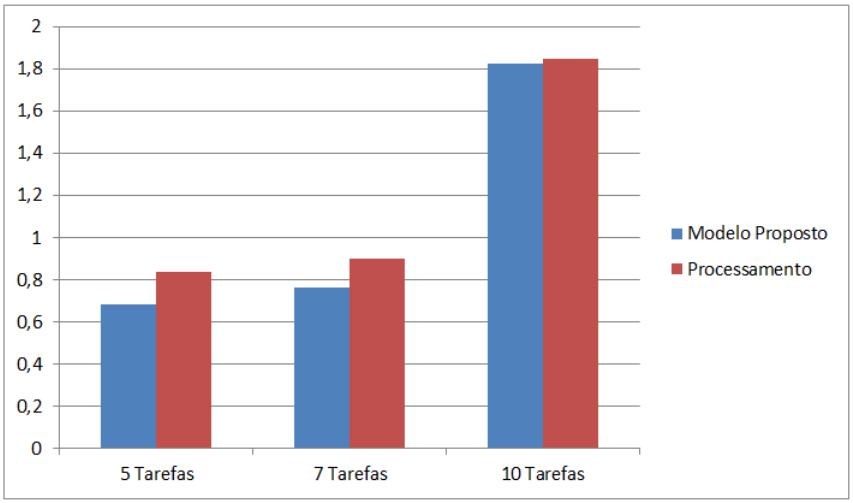


Figura 39 – 5 Clientes e 5 Servidores com variação das instâncias até 8 vezes

Fonte: autoria própria

Da mesma forma, a Figura 40 tem o resultado com o gráfico do modelo proposto mostrando o ganho em todos os blocos. Mas, com 10 tarefas, houve um empate nos resultados em 8 execuções, ganhando apenas em duas execuções. Provavelmente isso aconteceu pelo poucos recursos disponíveis nos servidores, tornando assim um número limitado de soluções viáveis.

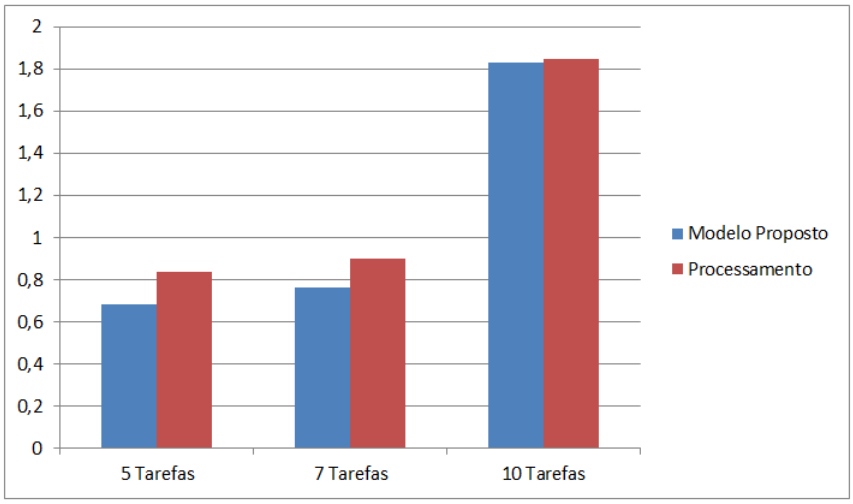


Figura 40 – 10 Clientes e 5 Servidores com variação das instâncias até 8 vezes

Fonte: autoria própria

A tabela 10 mostra todos os resultados encontrados com a execução do algoritmo com a variação 2, para um conjunto de tarefas. Nela encontra-se os melhores tempos de execução de cada método e o valor percentual da quantidade de execuções. Em alguns casos o método proposto conseguiu um resultado de 100% em 5 conjuntos de

instâncias de clientes, servidores e tarefas. Mas com 5 clientes, 5 servidores e 10 tarefas o ganho foi apenas de 20% devido aos tempos das execuções serem iguais. Foram executados 10 vezes e em oito execuções os resultados foram iguais tendo um ganho em apenas duas execuções. Analisando-se o menor tempo da tabela do modelo proposto que foi 0.4115383 onde ele obteve um ganho em 100% das execuções com o tempo do processamento de 0.4375531 . Essa diferença entre os tempos chegou a quase 6% de diferença.

Clientes	Servidores	Tarefas	Modelo Prop.	Processamento	Percentual
5	3	5	0.5549366*s*	0.5807849*s*	60
5	3	7	0.5385088*s*	0.6089041*s*	100
5	3	10	0.5464225*s*	0.5666999*s*	60
10	3	5	0.5469866*s*	0.5611681*s*	60
10	3	7	0.6821224*s*	0.7282090*s*	60
10	3	10	0.5464230*s*	$0,5767010s$	70
5	5	5	0.4115383*s*	0.4375531*s*	100
5	5	7	0.5045155*s*	0.5395725*s*	100
5	5	10	0.9853135*s*	0.9921367*s*	20
10	5	5	0.4115383*s*	0.4375531*s*	100
10	5	7	0.5045155*s*	0.5395725*s*	100
10	5	10	0.9753135*s*	0.995136*s*	20

Tabela 10 – Resultado da variação de instâncias até 2 vezes

A Tabela 11 mostra uma variação 4 nas instâncias e o seu percentual onde o modelo proposto conseguiu um ganho de 100% em 4 conjuntos de clientes, servidores e tarefas. Percebe-se que com 10 clientes, 5 servidores e 10 tarefas o tempo do modelo proposto foi de $0,5614546$. Com esse tempo ele conseguiu superar o seu próprio tempo com 10 clientes, 5 servidores e 10 tarefas na variação 2, mostra-se assim o quanto as instâncias são aleatórias.

Clientes	Servidores	Tarefas	Modelo Prop.	Processamento	Percentual
5	3	5	0.4375735*s*	0.4489702*s*	20
5	3	7	0.5863388*s*	0.5995083*s*	60
5	3	10	1.223848*s*	1.2401023*s*	20
10	3	5	0.4338194*s*	0.4395177*s*	20
10	3	7	0.5897127*s*	0.6004148*s*	50
10	3	10	1.0920309*s*	1.0920309*s*	50
5	5	5	0.4199420*s*	0.4415582*s*	100
5	5	7	0.508718*s*	0.5612861*s*	100
5	5	10	0.5975885*s*	0.6357419*s*	50
10	5	5	0.4199420*s*	0.4415582*s*	100
10	5	7	0.5087185*s*	0.5612861*s*	100
10	5	10	0.5614546*s*	0.5701256*s*	20

Tabela 11 – Resultado da variação de instâncias até 4 vezes

A Tabela 12 com a variação de instâncias até 8 vezes com o modelo proposto, obteve o melhor tempo em todos, mas, não diferente de algumas execuções, o algoritmo obteve resultados positivos com 20% em algumas execuções. Com 50% o algoritmo obteve um ganho em 5 execuções e houve um empate em outros 5 contabilizando apenas os ganhos. Analisando-se o menor tempo da tabela do modelo proposto que foi $0,6844672$ onde ele obteve um ganho em 100% das execuções com o tempo do processamento de $0,8345539$. Determinando a diferença entre os melhores tempos dos modelos, chegou a quase 18% de diferença entre os resultados.

Clientes	Servidores	Tarefas	Modelo Prop.	Processamento	Percentual
5	3	5	1.1308825*s*	1.2584926*s*	40
5	3	7	1.2983005*s*	1.3903891*s*	60
5	3	10	$1,3505566s$	$1,3625843s$	40
10	3	5	1.2077724*s*	1.3353824*s*	40
10	3	7	1.4003346*s*	1.4693524*s*	50
10	3	10	1.4205666*s*	1.4405666*s*	20
5	5	5	0.6844672*s*	0.8345539*s*	100
5	5	7	0.7611775*s*	0.8972039*s*	100
5	5	10	1.8261360*s*	1.8470229*s*	20
10	5	5	0.6844672*s*	0.8345539*s*	100
10	5	7	0.7611775*s*	0.8972039*s*	100
10	5	10	1.8261360*s*	1.8470229*s*	20

Tabela 12 – Resultado da variação de instâncias até 8 vezes

Analisando-se de formar geral as tabelas, conclui-se que a Tabela 10 houve em média uma ganho em torno de 70% das execuções. Já na Tabela 9 e 10 o ganho chega em média em torno de 58%. Mostrando-se que, com a variação 2 mesmo sendo um sistema mais homogêneo o modelo matemático proposto consegue se sair bem melhor do que

os métodos encontrados na literatura. O comportamento do modelo matemático com a variação de instâncias até 4 vezes e a variação de instâncias até 8 vezes num ambiente mais heterogêneo se mostra com um resultado mais próximo do processamento, mas deixando claro o quanto o modelo proposto é satisfatório em seus resultados.

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A computação em nuvem vem crescendo bastante nos últimos anos, visto a grande gama de vantagens que a mesma tem proporcionado aos seus usuários. Sua infraestrutura é composta por um grande número de máquinas físicas que são conectadas por meio de uma rede, onde as mesmas se encontram em lugares diferentes.

Para atender a grande demanda de execuções, essa tecnologia dispõe de algoritmos destinados a determinar a melhor forma possível de execução dessas tarefas nas máquinas disponíveis. Ou seja, esses algoritmos, informam em qual servidor cada tarefa será executada, visando entre outras coisas, a diminuição no tempo de execução. Assim, não diferente de outras tecnologias, a *cloud computing*também enfrenta alguns problemas no que se refere a performance, visto a complexidade presente nessa área.

Na tentativa de contribuir com a diminuição do tempo gasto para a executar as requisições feitas pelos usuários, nesse tipo de computação, este trabalho propôs um modelo para o gerenciamento do particionamento de conjuntos de tarefas em computação nas nuvens, para ser usada pelos escalonadores de tarefas, de forma que os mesmos possam escalonar, de maneira mais precisa.

Para testes, foram feitas coletas de resultados e análise comparativa, gerando assim um conjunto com 360 instâncias com variadas configurações. Essas variações possibilitaram testes em diferentes ambientes, aumentando assim a confiabilidade dos resultados.

De acordo com esses resultados, o trabalho proposto demostra um ganho médio de 60% em relação aos trabalhos que levam apenas o tempo de processamento em consideração. Além disso, esses resultados também mostraram em quais situações a formulação aqui proposta se comporta melhor. Como por exemplo, os casos em que a variação é maior, ou seja, simulando ambientes heterogêneos, obteve um ganho de 58%, enquanto que em situações em que é utilizada uma menor variação, essa superioridade tende a diminuir.

Para incentivar a continuação desse trabalho, alguns itens são apresentados para incrementar a formulação proposta ou a criação de novas soluções:

- Geração de novas instâncias. Como por exemplos tamanhos maiores e de características diferentes.
- Utilização de algoritmos heurísticos, para analisar o ganho médio obtido nos mesmos. Como por exemplo: GRASP e Algorítimo Genético.
- Simulação em máquinas virtuais.

REFERÊNCIAS

ARROYO, J. E. C.; RIBEIRO, R. L. P. Algoritmo genético para o problema de escalonamento de tarefas em máquinas paralelas com múltiplos objetivos. *XXXVI Simpósio Brasileiro de Pesquisa Operacional*, 2004. Citado na página 11.

AZAMBUJA, R. X.; SANTOS, L. C. V.; BORGES, P. S. S. Escalonamento com restrição de recursos utilizando algoritmo genético. *I Congresso Brasileiro de Computação*, 2001. Citado na página 11.

BACHIEGA, N. G. Algoritmo de escalonamento de instância de máquina virtual na computação em nuvem. Universidade Estadual Paulista (UNESP), 2014. Citado na página 26.

BARNHART, C. et al. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, INFORMS, v. 46, n. 3, p. 316–329, 1998. Citado na página 26.

BASSANEZI, R. C. *Ensino-aprendizagem com modelagem matemática: uma nova estratégia*. [S.l.]: Editora Contexto, 2002. Citado 2 vezes nas páginas 18 e 19.

BOROVSKIY, V. et al. A linear programming approach for optimizing workload distribution in a cloud. In: *CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization*. [S.l.: s.n.], 2011. p. 127–132. Citado na página 26.

BRADLEY, H. M. *Applied Mathematical Programming*. [S.l.]: Addison Wesley Publishing & Company, United States of America, 1977. Citado na página 22.

BRAGA, A. d. A. S. Relaxações lagrangianas e planos de corte faciais na resolução de problemas de particionamento de conjuntos. Biblioteca Digital da Unicamp, 2011. Citado na página 26.

BUYYA, R. et al. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, Elsevier, v. 25, n. 6, p. 599–616, 2009. Citado 4 vezes nas páginas 10, 14, 15 e 28.

CHEN, X.; CHEN, X.; ZHANG, X. Crew scheduling models in airline disruption management. In: IEEE. *Industrial Engineering and Engineering Management (IE&EM), 2010 IEEE 17Th International Conference on*. [S.l.], 2010. p. 1032–1037. Citado na página 26.

CHEVALLARD, Y. et al. *Estudar matemáticas: o ele perdido entre o ensino ea aprendizagem*. [S.l.]: Artmed, 2001. Citado na página 18.

COULOURIS, G. et al. *Sistemas Distribuídos-: Conceitos e Projeto*. [S.l.]: Bookman Editora, 2013. Citado 2 vezes nas páginas 14 e 31.

COUTINHO, R. d. C.; DRUMMOND, L. M.; FROTA, Y. Optimization of a cloud resource management problem from a consumer perspective. In: SPRINGER. *Euro-Par 2013: Parallel Processing Workshops*. [S.l.], 2014. p. 218–227. Citado 3 vezes nas páginas 11, 27 e 47.

DANTZIG. *Linear Programming and Extensions*. [S.l.]: Princeton, 1963. Citado na página 22.

DASTJERDI, A. V.; BUYYA, R. An autonomous reliability-aware negotiation strategy for cloud computing environments. In: IEEE. *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*. [S.l.], 2012. p. 284–291. Citado na página 14.

FERRUZZI, E. C. et al. A modelagem matemática como estratégia de ensino e aprendizagem do cálculo diferencial e integral nos cursos superiores de tecnologia. Florianópolis, SC, 2003. Citado na página 18.

JR, K. S. Balanceamento de carga: A evolução para os application delivery controller. *F5 Network Inc*, 2009. Citado na página 22.

KUROSE, J. F.; ROSS., K. W. *Redes de Computadores e a Internet*. [S.l.]: Person, 2006. Citado 3 vezes nas páginas 10, 16 e 36.

LEINBERGER, W. et al. Load balancing across near-homogeneous multi-resource servers. In: IEEE. *Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th*. [S.l.], 2000. p. 60–71. Citado na página 25.

MALIK, S. Dynamic load balancing in a network of workstations. *Paper for Parallel Processing Course, Carleton University*, 2000. Citado na página 25.

MCENTIRE, P. L.; O'REILLY, J. G.; LAISON, R. *Distributed Computing: Concepts and Implementations*. [S.l.]: IEEE Press, 1984. Citado na página 24.

MENDES, D. R. *Redes de Computadores*. [S.l.]: Novatec, 2007. Citado na página 17.

MOHAN, N.; RAJ, E. B. Resource allocation techniques in cloud computing–research challenges for applications. In: IEEE. *Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on*. [S.l.], 2012. p. 556–560. Citado na página 26.

MOR, Y.; NOSS, R. Programming as mathematical narrative. *International Journal of Continuing Engineering Education and Life Long Learning*, Inderscience Publishers, v. 18, n. 2, p. 214–233, 2008. Citado na página 21.

MOREIRA, M. A. Modelos científicos, modelos mentais, modelagem computacional e modelagem matemática: aspectos epistemológicos e implicações para o ensino. *Revista Brasileira de Ensino de Ciência e Tecnologia*, v. 7, n. 2, 2014. Citado na página 21.

OZDEMIR, H. T.; MOHAN, C. K. Graga: a graph based genetic algorithm for airline crew scheduling. In: IEEE. *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on*. [S.l.], 1999. p. 27–28. Citado na página 26.

PEIXOTO, M. L.; TOTT, R. F.; MONACO, F. J. Política de escalonamento de tempo-real para garantia de qos absoluta em cluster de servidores web heterogêneos. *WebMedia'07: Proceedings of the XIII Brazilian Symposium on Multimedia and the Web*, 2007. Citado na página 11.

REMY, J. Resource constrained scheduling on multiple machines. *Information Processing Letters*, v. 91, p. 177–182, 2004. Citado na página 11.

SEEHORN, D. et al. Csta k–12 computer science standards: Revised 2011. ACM, 2011. Citado na página 21.

SOROR, A. A. et al. Automatic virtual machine configuration for database workloads. *ACM Trans. Database Syst.*, v. 35, p. 1–47, 2010. Citado na página 10.

SOUSA, F.; MOREIRA, L.; MACHADO, J. Computação em nuvem: Conceitos, tecnologias, aplicações e desafios. *Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI)*, v. 1, p. 159–175, 2009. Citado 2 vezes nas páginas 14 e 15.

TANENBAUM, A. *Redes de computadores*. [S.l.: s.n.], 2003. Citado na página 16.

VECCHIOLA, C.; CHU, X.; BUYYA, R. Aneka: A software platform for .net-based cloud computing. 2009. Citado na página 15.

WAN, S.; ALMEIDA, L. Thermal-aware optimization of workload distribution in data centers. In: IEEE. *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*. [S.l.], 2012. p. 494–497. Citado 2 vezes nas páginas 26 e 46.

WILLIAMS, H. P. *Model Building in Mathematical Programming*. [S.l.]: John Wiley & Sons Ltd, England, 1999. Citado na página 22.

XU, Z.; HUANG, R. Performance study of load balancing algorithms in distributed web server systems. *CS213 Parallel and Distributed Processing Project Report*, v. 1, 2009. Citado na página 24.

XUE, J. et al. A study of task scheduling based on differential evolution algorithm in cloud computing. In: IEEE. *Computational Intelligence and Communication Networks (CICN), 2014 International Conference on*. [S.l.], 2014. p. 637–640. Citado 3 vezes nas páginas 11, 27 e 46.

YUNES, T. H.; MOURA, A. V.; SOUZA, C. C. D. Hybrid column generation approaches for urban transit crew management problems. *Transportation Science*, INFORMS, v. 39, n. 2, p. 273–288, 2005. Citado na página 26.

ZIONTS, S. *Linear and Integer Programming*. [S.l.]: Prentice-Hall, Inc, Englewood Chiffs, New Jersey, United States of America, 1963. Citado na página 22.