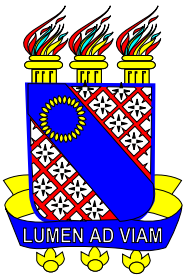

UNIVERSIDADE ESTADUAL DO CEARÁ

LEONARDO MENEZES DE SOUZA

UMA METODOLOGIA PARA ANÁLISE DE DESEMPENHO EM SISTEMAS DE COMPUTAÇÃO EM NUVENS

FORTALEZA - CEARÁ

2012

LEONARDO MENEZES DE SOUZA

UMA METODOLOGIA PARA ANÁLISE DE DESEMPENHO EM SISTEMAS DE COMPUTAÇÃO EM NUVENS

Dissertação apresentada no Curso de Mestrado Acadêmico em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciências da Computação.

Área de Concentração: Redes de Computadores.

Orientador: Prof. Dr. Jorge Luiz de Castro e Silva

Co-Orientador: Marcial Porto Fernandez

FORTALEZA - CEARÁ

2012

C824p

Souza, Leonardo Menezes de

Uma metodologia para análise de desempenho em sistemas de Computação em Nuvens / Leonardo Menezes de Souza– 2012.

77 f. : il. 30cm.

Orientador: Prof. Dr. Jorge Luiz de Castro e Silva

Coorientador: Prof. Dr. Marcial Porto Fernandez

Dissertação (Mestrado) - Universidade Estadual do Ceará, Centro de Ciências e Tecnologia, Curso de Mestrado Acadêmico em Ciência da Computação. Área de Concentração: Redes de Computadores. Fortaleza, 2011.

1. Computação em Nuvens 2. *Benchmarks* 3. *Desempenho* 4. *DEA* 5.

I. Universidade Estadual do Ceará, Centro de Ciências e Tecnologia.

CDD:001.6

LEONARDO MENEZES DE SOUZA

UMA METODOLOGIA PARA ANÁLISE DE DESEMPENHO EM SISTEMAS DE COMPUTAÇÃO EM NUVENS

Dissertação apresentada no Curso de Mestrado Acadêmico em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial para obtenção do grau de Mestre.

Aprovada em: 06/09/2012

BANCA EXAMINADORA

Prof. Dr. Jorge Luiz de Castro e Silva
Universidade Estadual do Ceará – UECE
Orientador

Prof. Dr. Marcial Porto Fernandez
Universidade Estadual do Ceará – UECE

Prof. Dr. Cidcley Teixeira de Souza
Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE

Prof. Dr. Gerardo Valdisio Rodrigues Viana
Universidade Estadual do Ceará – UECE

AGRADECIMENTOS

À minha família, por sempre acreditarem em mim. Às mulheres da minha vida; minha mãe Raimundinha, e minha irmã Andreia. Ao meu pai Juarez, ao meu tio José Maria Melo (In Memoriam). E à minha prima Daila Melo. Sem vocês, os caminhos teriam sido bem mais árduos.

Ao professores Jorge Luiz Castro e Sila e Marcial Fernandez, pelas orientações e pela oportunidade de executar esse trabalho.

Ao professores Jerfesson Teixeira, Ricardo Rodrigues, Cidcley Teixeira, Gustavo Campos, Clecio Thomaz, André Santos e Joaquim Celestino, pelos ensinamentos, orientações, força, prontidão e serenidade, em momentos críticos, que foram essenciais para continuar seguindo em frente.

Aos amigos do MACC/InSeRT, que contribuíram para o meu crescimento intelectual, minha formação e minha vida, seja em uma aula, ou tomando um café durante os intervalos. Saudações especiais ao Walisson Pereira (guru-mestre-doutor), grande amigo e orientador. Aos amigos de trabalho e estudo, Alandson Mendonça, Pablo Nóbrega, Luanna Falcão, Silas Santiago, Renan Henry, Luis Ribeiro, Alisson Barbosa, Sergio Vieira, Sergio Luis, Thiago Gomes, Heitor Barros, Anderson, Luiz Gonzaga, Edgar Tarton, Rodrigo Magalhães, Inácio, Saulo Hachem, Leandro Jales, Davi Teles, Fred Freitas, Klecia Pitombeira e Davi di Cavalcante.

Aos amigos e parentes, que nunca estiveram ausentes durante este período, sempre me dando força: Lila Licarião (nenem), Vilson Nunes, Aldo Bessa, Leticie Bastos, Breno Câmara, Daila Bertulis & Ed, Jack Anderson, Iane Rocha, Manuela Sales, Raquel Costa, Diego Lucena, Priscila Aranha, Flora Bezerra, Pedro Nogueira, Daniel Feitosa, Rafael Feitosa, Aline Honório, Ricardo Moreira, Davson Maia, Flora Bezerra, Pedro Nogueira, Fernando Pinheiro, Adriano Menezes, Felipe Leitão, Fabrício Carvalho, Rinaldo Marx, e Vinícius Zavam.

Aos parceiros de banda (A Trigger to Forget) por proporcionarem momentos de descontração e relaxamento: Lucas Nobre, Rafael Benevides, Leonardo Mamede, e Lucas Martins.

A todas as pessoas que passaram pela minha vida e contribuíram para a construção de quem sou hoje, e que não puderam estar nessa folha de papel, mas com certeza são sempre lembradas pelo meu coração.

"A ciência descreve as coisas como são; a arte, como são sentidas, como se sente que são."
Fernando Pessoa

RESUMO

Computação em nuvens é um modelo de computação distribuída baseado em Internet, onde poder computacional, infraestrutura, aplicações, e até distribuição de conteúdo colaborativo são providos aos usuários através da nuvem como um serviço, em qualquer lugar e a qualquer momento. A adoção de sistemas de computação em nuvens nos últimos anos é notável, e vem ganhando mais visibilidade progressivamente. Com isso, análises críticas inerentes às características físicas da nuvem devem ser executadas para garantir o funcionamento consistente do sistema. Portanto, é necessário propor uma abordagem de medição de desempenho das plataformas de computação em nuvens levando em consideração o funcionamento eficaz dos pontos críticos de hardware e rede, tais como: taxas processamento, taxa de atualização do buffer de memória, transferência de armazenamento, e a latência da rede. Para tal, utiliza-se a metodologia DEA em testes provenientes de duas suítes de benchmarks: HPCC e PTS. Analisa-se os resultados e transcreve-se uma formulação para análise de desempenho dos recursos requeridos em aplicações padrão de computação em nuvens, exprimindo a utilidade desta pesquisa para os usuários e gestores deste tipo de sistema.

Palavras-Chave: Computação em Nuvens. *Benchmarks. Desempenho. DEA*.

.

ABSTRACT

Cloud Computing is a distributed computing model based on the Internet, where computational power, infrastructure, applications, and even collaborative content distribution are provided to users through the Cloud as a service, anywhere, anytime. The adoption of Cloud Computing systems in recent years is remarkable, and it is gradually gaining more visibility. Thus, critical analysis inherent to cloud's physical characteristics must be performed to ensure consistent system running. Therefore, it is necessary to propose an approach to performance measurement of Cloud Computing platforms taking into account the effective functioning of critical Hardware and Network matters, such as: Processing rate, Memory Buffer refresh rate, Disk I/O transfer rate, and the Network Latency. For this purpose, it uses the DEA metodology in tests from two benchmark suites: HPCC and PTS. The results are analyzed, and a formulation is fitted to analyze the required resources' performance in cloud-standard applications, expressing the usefulness of this research to Cloud Computing adopters.

Keywords: Cloud Computing. Benchmarks. Performance. DEA.

.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

b_{eff}	*Effective Bandwidth Benchmark*
BLAS	*Basic Linear Algebra Subprograms*
CDN	*Content Distribution Network*
CRS	*Constant Returns to Scale*
DDR3	*Double-Data Rate*
DEA	*Data Envelopment Analysis*
DFT	*Discrete Fourier Transform*
DGEMM	*Double-precision General Matrix Multiply*
DMU	*Decision Making Units*
EC2	*Elastic Compute Cloud*
ECUs	Elastic Compute Units
FFT	*Fast Fourier Transform*
GUPS	*Giga Updates Per Second*
HPL	*High Performance Linpack*
IaaS	*Infrastructure as a Service*
InSeRT	*Information Security Research Team*
ISP	*Internet Service Provider*
MJMI	*Multiple-Job-Multiple-Instance*
MPI	*Message Passing Interface*
MTC	*Many-Task Computing*
MV	Máquina Virtual
PaaS	*Platform as a Service*
PPI	*Parallel Production Infrastructures*
QoS	*Quality of Service*
RAM	*Random Acess Memory*
RMW	*Read-Modify-Write*
SaaS	*Software as a Service*
SAS	*Serial Attached SCSI*
SSH	*Secure Shell*
TCP	*Transmission Control Protocol*
TI	Tecnologia da Informação
ToR	*Top-of-Rack*
VRS	*Variable Returnsto Scale*

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de Processamento Distribuído, tais como *grids*, *clusters* e PPI (Parallel Production Infrastructures), tem sido alvo de estudos de análise de desempenho, devido à sua importância para o processamento em grande escala. Assim como estas plataformas de Computação Distribuída, um novo modelo de computação baseado em Internet, vem se popularizando a cada dia. Trata-se da Internet com todos os padrões e protocolos associados para prover um conjunto de serviços *web*. O que torna esta tecnologia atrativa é a maneira como os clientes (usuários) trocam informações, acessam e compartilham conteúdo.

A ideia de computação em nuvens é uma pretensão de longa data. Não é um modelo novo, pois remete à década de 60 (PARKHILL, 1966), mas nunca foi posta em prática devido à precariedade da largura de banda oferecida pelos provedores. Atualmente, tornou-se uma realidade comercial, pois os negócios de Tecnologia da Informação tem apresentado um crescimento considerável, acompanhando a demanda dos usuários por poder computacional, impulsionados pelo advento da popularização da Internet banda larga, promovendo a evolução da indústria de TI (ARMBRUST et al., 2009).

Em Sosinsky (2011), foi definido que uma estrutura de computação em nuvens deve possuir recursos agrupados e particionados de acordo com a necessidade de utilização. Ao desenharmos a Internet como uma nuvem, representa-se a primeira das características essenciais desse modelo: a abstração. A virtualização é o grande trunfo da computação em nuvens, pois abstrai as características físicas do *hardware*, emulando sistemas operacionais em uma única plataforma computacional, formando uma camada de abstração dos recursos da plataforma.

Em funcionamento normal, essa infraestrutura atende a vários *workloads* simultaneamente. Estes *workloads* são oriundos de MVs (máquinas virtuais) ou de qualquer outra requisição externa, e são compostos de várias instâncias e/ou tarefas (*workloads*) MJMI (Multiple-Job-Multiple-Instance). Em contrapartida, não há referências a pesquisas de análise de desempenho através de *benchmarks* e/ou *baselines* de *kernels* para sistemas de Computação Distribuída com infraestrutura compartilhada por várias tarefas independentes, tal como a computação em nuvens (OSTERMANN et al., 2010).

Inicialmente, a Internet era uma espécie de “rede de redes”, com uma arquitetura redundante que sobreviveria à enormes perturbações. Hoje em dia, os recursos conectados à Internet, além de normalmente redundantes (por questões de segurança), tornaram-se bastante escaláveis, fato que denota a segunda característica essencial da computação em nuvens: a escalabilidade. A escalabilidade permite que os ambientes virtuais sejam ampliados ou reduzidos dinamicamente para atender à solicitações de usuários por recursos.

A infraestrutura do *Google* por exemplo, além de desenvolver sua própria demanda

de aplicativos baseados em nuvem, arrenda sua plataforma a desenvolvedores de aplicações baseadas na *web*. Em contrapartida, a maioria das empresas de pequeno ou médio porte não possui fundos suficientes para investir em estruturas físicas de grande porte. Logo, a solução mais viável e vantajosa é fazer um *outsourcing* dos recursos requeridos. Este fato credencia a enumeração da terceira característica essencial da computação em nuvens: a possibilidade da nuvem funcionar como uma utilidade, e os serviços providos serem disponibilizados, utilizando uma abordagem *pay-as-you-go*.

Cloud computing – ou computação em nuvens – representa um paradigma da computação em grande escala como conhecemos hoje, pois traz à tona questões referentes às funcionalidades que dispõe. A avaliação abordada neste trabalho é focada na criticidade e no desempenho dos recursos virtualizados da estrutura de computação em nuvens.

Tal avaliação é requerida devido o desempenho dos recursos virtualizados não ser transparente para a gerência de rede, demandando uma metodologia que permita quantizá-la de acordo com a especificidade da plataforma. Utilizando esta metodologia para medições periódicas de desempenho, a disponibilidade prometida possuirá maior garantia de confiabilidade, reduzindo riscos de mau funcionamento. Portanto, este é o momento para expor ideias de soluções, para que possamos responder às hipóteses levantadas, e explorar o potencial deste "novo" modelo de tecnologia distribuída.

1.1 Contextualização e Problemática

De acordo com a Lei de Moore (MOORE et al., 1998), os processos evolutivos das arquiteturas computacionais são muito mais frequentes que o consumo destas tecnologias, fato que torna o *hardware* obsoleto cada vez mais rápido. A constante evolução da indústria de TI, requer investimento para renovação de *hardware*, o que pode trazer instabilidades ao sistema, devido às reposições sazonais. Compreendemos, então, a necessidade em tornar o *hardware* rentável o mais rápido possível. As plataformas de computação em nuvens utilizam a virtualização para a abstração do seu próprio *hardware*, aumentando sua base de usuários, porém diminuindo potencialmente o desempenho alcançável (IOSUP et al., 2011).

Outro fato para o qual devemos atentar é que as reposições de equipamento, tendo como objetivo a melhoria do desempenho, consequentemente levam a um maior consumo de energia. Durante o tempo de vida de uma estrutura computacional, o gasto com recursos de energia e refrigeração podem superar o valor do *hardware*. Os serviços de computação em nuvens foram projetados para substituir *data centers* empresariais de pequeno e médio porte subutilizados (10 a 20% de utilização) (IOSUP et al., 2011).

A adoção de sistemas de computação em nuvens nos últimos anos vem ganhando mais visibilidade. A demanda crescente dos usuários por disponibilidade de recursos computacionais aliada à capacidade do sistema de entregar resultados em um tempo bastante razoável (tempo-real), faz da infraestrutura deste modelo uma plataforma mais confiável, mais barata, e mais escalável que *grids*, *clusters* e supercomputadores (OSTERMANN et al., 2010). Com isso, análises críticas inerentes às características físicas da nuvem devem ser executadas para garantir um desempenho eficiente do sistema.

A computação em nuvens provê maior flexibilidade, maior exploração do potencial dos recursos, e elasticidade. Esta abordagem de *marketing* computacional transformou grande parte da indústria de TI, provendo *softwares* ainda mais atrativos como serviços. É a implementação da computação como uma utilidade, vista como um serviço de utilidade pública como energia elétrica, água, etc. Cada serviço procura atender às necessidades de seus clientes, cobrando-os de acordo com a utilização do serviço (ARMBRUST et al., 2009).

Com a evolução computacional, os dispositivos portáteis microprocessados, como computadores cada vez menores, *tablets*, *smartphones*, entre outros dispositivos móveis, vêm se popularizando. Assim, a evolução das telecomunicações, disponibiliza o acesso à Internet móvel proporcionalmente à demanda computacional. Os aplicativos desenvolvidos para estes dispositivos móveis, e até mesmo para máquinas tradicionais, são hospedados em servidores na nuvem, tal como serviços a serem disponibilizados sem necessidade de instalação local prévia. Desta maneira, observamos a formação de ambientes de processamento distribuídos, proporcionando ao usuário flexibilidade e disponibilidade no acesso ao conteúdo hospedado.

Toda esta "facilidade" é sensível aos usuários do sistema, que percebem quando a entrega dos serviços está degradada ou não. Por outro lado, os gestores e desenvolvedores deste tipo de plataforma, necessitam entregar estes serviços aos clientes de maneira confiável, disponível, e escalável. Tal desempenho não é transparente para a gerência, mesmo utilizando um software de gerenciamento, demandando uma maneira eficiente de quantizá-la, que será exposta mais adiante neste trabalho.

Tendo em vista o atendimento dos requisitos de disponibilidade, tais como *zero down-time*, que é extremamente crítico e a alta latência de rede, que pode interromper a comunicação do sistema; verificamos a necessidade de avaliar os pontos críticos de *hardware* e rede das MVs pertencentes ao sistemas de computação em nuvens, a fim de manter o sistema operante, excluindo riscos de mau funcionamento. Cada infraestrutura possui restrições e diferenças entre si, contudo possuem a mesma criticidade, a capacidade de entrega das requisições realizada pelos recursos virtualizados na estrutura.

Por conta das criticidades de *hardware* e virtualização, a operação eficiente do sistema dependerá do bom desempenho dos recursos da estrutura computacional. Caso contrário, permitirá uma continuidade da subutilização da estrutura computacional, como foi evidenciado

em Iosup et al. (2011). Assim, detectamos o potencial problema a ser resolvido através das análises de *benchmarks* e posterior formulação para a avaliação do desempenho de sistemas de computação em nuvens, levando em consideração a ponderação dos resultados obtidos.

1.1.1 Objetivos Gerais

Neste trabalho propõe-se uma metodologia genérica para a avaliação do desempenho de uma estrutura de computação em nuvens; padronizando o método e contemplando um vasto contingente de sistemas. Tal metodologia atenderá qualquer estrutura de computação em nuvens, uma vez que é orientada ao desempenho dos recursos da estrutura. A avaliação deverá considerar a influência de cada recurso sob o desempenho total do sistema. Determina-se então, qual destes recursos apresenta maior relevância para o sistema, auxiliando na decisão de qual modelo de infraestrutura proporcionará melhor eficiência de consumo aos usuários, desenvolvedores e administradores.

Traçados os objetivos deste trabalho, percebe-se mais uma motivação para a proposição de uma abordagem de avaliação do desempenho das plataformas de computação em nuvens, destacando a utilização dos recursos da infraestrutura. O tempo de resposta da nuvem para alocação de recursos e a latência da rede atrelados à utilização da metodologia DEA parametrizam os resultados encontrados pelos testes realizados em cada instância computacional, apresentando a eficiência de cada experimento executado.

1.1.2 Objetivos Específicos

Propõe-se uma abordagem de análise de desempenho das plataformas de computação em nuvem, levando em consideração o funcionamento eficaz dos pontos críticos de *hardware* e rede. Os objetivos principais deste trabalho são relacionados a seguir:

- Obter conhecimento sobre o *hardware* da estrutura de computação em nuvens.
- Implementar as MVs na estrutura de computação em nuvens baseadas no padrão Amazon EC2 (AMAZON, 2008), apresentadas na Tabela 5.
- Instalar e executar os *benchmarks* a fim de simular a execução de aplicações e mensurar o desempenho dos recursos envolvidos, atividades apresentadas na Seção 4.2.
- Elaborar uma metodologia para avaliação do desempenho através de uma formulação matemática realizada na Seção 3.1.6.
- Exprimir analiticamente a taxa de desempenho de cada recurso utilizado e seu grau de importância para o funcionamento eficaz no ambiente, apresentado na Seção 4.3.

- Calcular os índices propostos na formulação, a fim de encontrar o desempenho inerente a cada recurso proprietário da estrutura de computação em nuvens.

1.2 Trabalho Realizado

Neste trabalho, foram realizados experimentos para avaliação de desempenho dos sistemas de computação em nuvens, utilizando *benchmarks* e a metodologia DEA. Desta maneira, foi possível a aplicação da formulação proposta, e o alcance dos níveis de desempenho de cada recurso. Levamos em consideração o desempenho médio dos pontos críticos de *hardware* e rede, tais como: processamento, taxa de atualização do *buffer* de memória, E/S de armazenamento e latência da rede. Duas suítes de *benchmarks* foram utilizadas na avaliação dos pontos críticos do sistema, apresentadas a seguir.

O HPC *Challenge benchmark* (HPCC) utiliza *kernels* computacionais reais, permitindo dados de entrada e tempos de execução de tamanhos variáveis, de acordo com a capacidade do sistema utilizado (DONGARRA; LUSZCZEK, 2010). O HPCC consiste em sete *benchmarks* responsáveis pela análise individual de cada componente crítico do sistema de computação em nuvens apresentados na Seção 3.1.2.

O outro conjunto de testes utilizado na avaliação dos pontos críticos é o Phoronix Test-Suite, ou PTS (LARABEL; TIPPETT, 2008b), que é mais respaldado em termos comerciais, sendo a ferramenta-base do site referência para análise de sistemas de nuvens públicas do mundo todo; o *Cloud Harmony* (READ, 2009). O PTS consiste em mais de 130 testes para análise de sistemas, porém apenas três *benchmarks* serão utilizados nesta proposta. A escolha e filtragem dos *benchmarks* para utilização neste trabalho tomou como parâmetro a manipulação eficaz e compatibilidade do resultados obtidos. Os três *benchmarks* selecionados apresentaram maior estabilidade e verossimilhança quando comparado a *benchmarks* com o mesmo objetivo (rede, memória, disco ou processamento), e serão apresentados na Seção 3.1.3.

A partir dos resultados obtidos em ambas as suítes de *benchmark*, analisamos os resultados, atribuindo pesos de acordo com a relevância de cada recurso da plataforma, e transcrevemos uma formulação considerando o desempenho médio de cada recurso em cada instância implementada no experimento. A formulação também leva em consideração o *overhead* atrelado a cada recurso avaliado, culminando na representação do desempenho real destes recursos. Mais informações sobre a metodologia utilizada nesta dissertação estarão nos Capítulos 3 e 4.

1.3 Resultados Obtidos

A tecnologia de computação em nuvens representa a abstração dos recursos computacionais disponibilizando serviços sob demanda, mesmo que a virtualização imponha certas

limitações de desempenho para os *workloads* processados. Após a execução dos *benchmarks*, a aplicação da metodologia DEA, e a classificação dos recursos quanto ao seu desempenho, aplicamos os resultados obtidos na formulação proposta na Seção 3.1.6. Tal formulação apresentou resultados relativos ao desempenho de cada recurso hospedado em cada MV implementada dentro da estrutura de computação em nuvens.

Enfim, pudemos destacar o desempenho médio atribuído a cada recurso utilizado, levando em consideração seus respectivos *overheads* de funcionamento. Obtivemos o desempenho médio de cada recurso da plataforma de computação em nuvens como resultado final, representado pelo índice de avaliação de desempenho global por recurso, considerando todas as MVs implementadas na plataforma. Desta maneira, será possível avaliar o desempenho dos recursos de qualquer estrutura de computação em nuvens, considerando apenas a ênfase do sistema quanto aos recursos mais críticos, independentemente de quantas MVs sejam implementadas.

1.4 Organização da Dissertação

Esta dissertação está dividida em seis capítulos, onde o Capítulo 2 expõe os conceitos fundamentais para a compreensão do problema abordado, o estado da arte da infraestrutura de computação em nuvens, desde sua definição estrutural até as impressões do mercado com relação a esta nova tecnologia. Neste capítulo ainda são apresentados os *benchmarks* utilizados, bem como a metodologia DEA que propicia análises de eficiência comparativas em organizações complexas, retornando resultados dos sistemas com melhor desempenho. Ao final do capítulo, contamos com a justificativa sobre as pesquisas tomadas como base para a realização deste trabalho.

O Capítulo 3 refere-se à proposta da metodologia de análise de desempenho em sistemas de computação em nuvens, tomando como referência as principais aplicações deste tipo de estrutura computacional, e a apresentação dos *benchmarks*. Avaliamos os principais requisitos das aplicações, a análise específica dos benchmarks sobre cada métrica, e então propomos uma formulação que contempla o desempenho eficaz das aplicações de computação em nuvens.

No Capítulo 4 apresentamos o cenário utilizado para implementar a pesquisa, uma descrição específica dos benchmarks para análise de desempenho do sistema de computação em nuvens, a justificativa de cada resultado obtido, e a abordagem experimental adotada para analisar o desempenho em sistemas de computação em nuvens.

O Capítulo 5 apresenta a análise dos resultados obtidos a partir das simulações, aplicando a formulação proposta. Finalmente no Capítulo 6 exporemos as conclusões alcançadas, a contribuição da pesquisa e as propostas de trabalhos futuros.

2 CONCEITOS

Este capítulo apresenta os conceitos fundamentais e os trabalhos pesquisados para a compreensão aprofundada do assunto em questão, e para a concepção das contribuições da presente dissertação. A Seção 2.1 apresenta a definição de um sistema de computação em nuvens, os conceitos básicos, a representatividade do sistema, o estado da arte da arquitetura, a virtualização dos recursos, a utilização dos *benchmarks* e da metodologia DEA. A seção 2.2 resume os trabalhos relacionados, considerados como alicerce fundamental para o desenvolvimento da presente pesquisa. Apresentaremos a ideia central dos trabalhos pesquisados, com o objetivo de contextualizá-los no âmbito da importância da realização deste trabalho.

2.1 Fundamentação Teórica

Foster et al. (2008) definiu que computação em nuvens é um paradigma da computação de grande porte focado em proporcionar economia escalável na qual um conjunto abstrato, virtualizado, e dinamicamente escalável de poder de processamento, armazenamento, plataformas e serviços são disponibilizados sob demanda para clientes através da Internet. A partir desta definição veremos como a computação de Internet é estruturada e como esta chegou no atual patamar de evolução.

Nesta Seção apresentaremos os conceitos mais importantes relacionados aos atributos de computação em nuvens. Na Subseção 2.1.1, apresenta-se o histórico, o estado da arte, os serviços atrelados, e dados estatísticos sobre a arquitetura física e lógica da infraestrutura de computação em nuvens. Na Subseção 2.1.2, apresenta-se os conceitos, funcionalidades, e melhores práticas relativos à tecnologia de virtualização. E por último, nas Subseções 2.1.3 e 2.1.4, apresenta-se os conceitos e funcionalidades dos *benchmarks* e da metodologia DEA, ambas aplicações importantíssimas para a realização deste trabalho.

2.1.1 Arquitetura

Conforme Lam et al. (2010), a Internet é dividida em dois momentos de topologias distintas: A Internet 1995-2007, que representa a visão hierárquica da Internet inicial; contando com poucos provedores de serviços de *core* de grande porte (em um nível mais alto), e provendo interconexões de *backbone* para os ISPs, provedores de acesso regional e local (em um nível mais baixo), conforme a Figura 1.

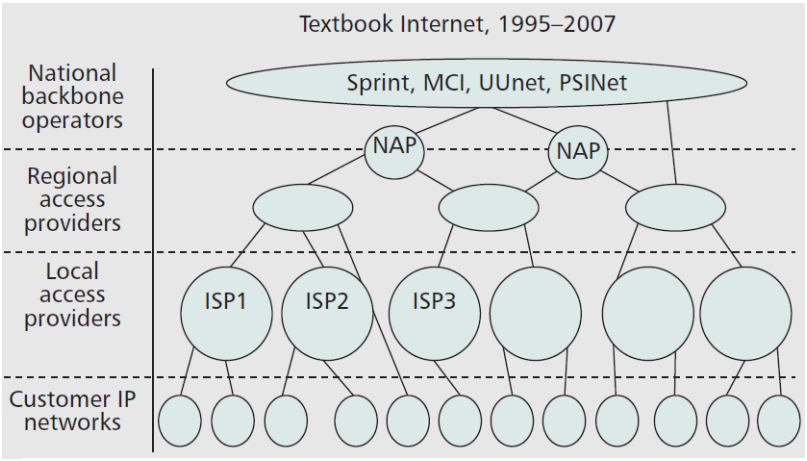

Figura 1: Evolução da Internet: 1995 a 2007 (LAM et al., 2010).

O segundo momento é referente à Internet de 2009 em diante, em que a topologia foi transformada em um modelo de conectividade mais entrelaçado, conforme mostra a Figura 2. O núcleo central - dominado pelos provedores tradicionais de *backbone* - é conectado por gigantes das comunicações, oferecendo conteúdo rico, hospedagem, e uma rede de serviços de distribuição de conteúdo (CDN).

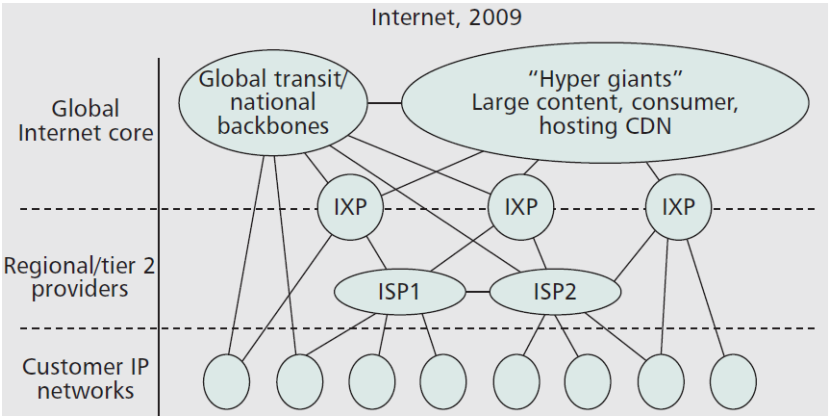

Figura 2: Evolução da Internet: 2009 em diante (LAM et al., 2010).

Assim sendo, a computação em nuvens é um modelo de sistema de computação de Internet que tem como objetivo impulsionar a próxima geração de *data centers*, arquitetando-os como uma rede de serviços virtuais (*hardware*, banco de dados, interface com o usuário, etc.). Assim, seus usuários podem acessar e instalar aplicativos em qualquer lugar do mundo, sob demanda, a custos competitivos, dependendo dos seus requerimentos de QoS. Deste modo, essa evolução representa benefícios às empresas de TI, que ficam livres da responsabilidade de configuração de uma infraestrutura de *hardware* e *software* (servidores), possibilitando um foco maior na inovação e geração de negócios de valor (WEISS, 2007).

Para melhor compreendermos a comunicação em nuvens, necessitamos entender como funcionam as comunicações entre *data centers*. Um *data center* é uma estrutura computacional massiva e paralela que consiste em *clusters* com milhares de servidores conectados em rede. Servidores dentro do mesmo *rack* são interconectados por um *switch* ToR. Tais *racks* são conectados a um *switch* de *cluster*, provendo interconectividade, como mostra a Figura 3.

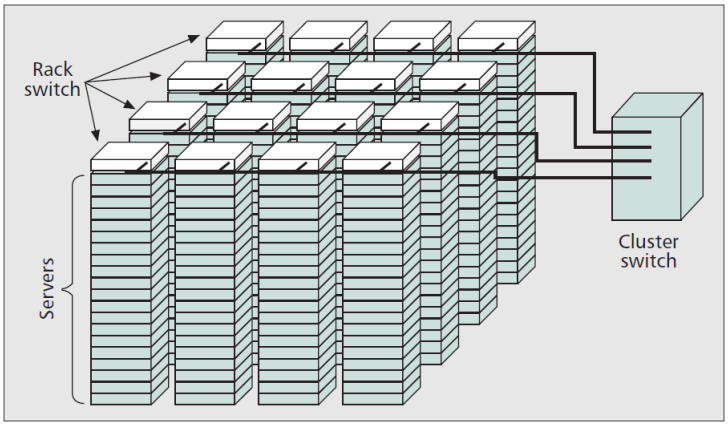

Figura 3: Comunicação Intra *Data Center* (LAM et al., 2010).

Data centers distribuídos não proveem somente redundância, mas também são utilizados para balanceamento de carga de processamento, melhorando a experiência do usuário e reduzindo a latência no tráfego. As redes ópticas de longas distâncias (*long-haul*) (OPTICOMM, 2008) se fazem necessárias para manter o tráfego entre *data centers*, distribuindo-os para os centros nos quais os usuários estão localizados, reduzindo assim os custos de interconexão; transportando dados entre localidades remotas, e garantindo capacidade escalável suficiente para suportar as operações necessárias. O tráfego entre *data centers* é em sua maioria gerado por máquinas que rodam aplicações de computação em nuvens, com um grande volume de tráfego (LAM et al., 2010).

A tecnologia de computação em nuvens é baseada nos conceitos da chamada *utility computing* (WARDLEY, 2009). O termo "*utility*" faz menção aos serviços de utilidade pública, tais como o fornecimento de energia elétrica, água, etc. Os provedores procuram atender às necessidades dos seus clientes disponibilizando recursos computacionais sob demanda, utilizando um modelo de tarifação do usuário de acordo com a utilização do recurso, ao invés de exigir o uso de licenças temporárias. Essa abordagem de marketing computacional é conhecida como *pay-as-you-go*, e tem potencial para transformar grande parte da indústria de TI, provendo *softwares* mais atrativos como serviços (ARMBRUST et al., 2009).

A expressão *Everything-as-a-Service* (EaaS) é um conceito que mostra a capacidade de acesso aos recursos computacionais através de um ambiente virtualizado. Todos esses recursos estão associados a aplicações (SaaS), equipamentos (IaaS) e plataformas (PaaS). SaaS possibilita que uma instância executável de um *software* instalado em um servidor remoto seja

acessada e executada por vários computadores-clientes sem necessidade de instalação local. Similarmente, o IaaS compreende o provisionamento de infraestrutura baseado na virtualização de entidades tais como servidores, *data centers* ou equipamentos de rede. Por fim, o PaaS consiste em uma plataforma onde os recursos estão alocados (*hardware* e *software*) e serão posteriormente executados (ARMBRUST et al., 2009). É possível verificar a estrutura de uma plataforma de computação em nuvens a partir da Figura 4.

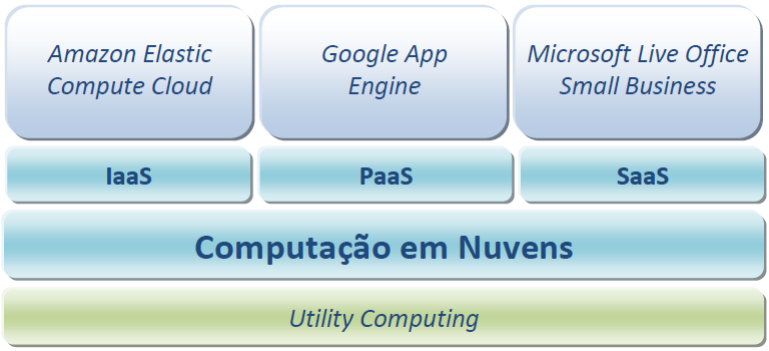

Figura 4: Estrutura de Recursos em Computação em Nuvens.

A topologia de computação em nuvens consiste em uma nuvem de Mega *data centers* geograficamente distribuídos, interconectados por uma rede de alta capacidade. Esta infraestrutura representa o principal montante de despesas operacionais (incluindo o fornecimento de energia e o resfriamento do equipamento) às provedoras, responsável por 75% do TCO (*Total Cost of Ownership)*. A principal motivação para manter um sistema de computação em nuvens, para a maioria das empresas é a agilidade, enquanto que o restante enfatiza mais a redução do TCO (NARASIMHAN; NICHOLS, 2011).

Em Narasimhan e Nichols (2011), uma pesquisa realizada com empresas adotantes da tecnologia de computação em nuvens, mais de 60% afirmou que as soluções hospedadas na nuvem são melhores que as soluções locais, e que pretendem rodar suas aplicações em nuvens públicas em até três anos. Mais de 80% das empresas afirmaram conseguir respostas mais rápidas às necessidades dos negócios, tornando a tecnologia um paradigma em relação ao papel da TI em seus negócios.

Os usuários tem seus dados residentes na rede, acessíveis em qualquer lugar do mundo e ainda contam com um *backup* automático destes dados. Proporciona ainda a possibilidade de compartilhamento, permite que múltiplos colaboradores modifiquem os dados simultaneamente, e melhor utilizem os recursos computacionais através da virtualização, pois os usuários não precisam se preocupar em manter sua própria máquina, gerando assim, uma considerável redução de despesas (ARMBRUST et al., 2009).

2.1.2 Virtualização

Virtualização é capacidade de criar diversas máquinas lógicas (virtuais) em um único *hardware*. Origina-se do particionamento físico, que divide um único servidor físico em múltiplos servidores lógicos. Desse modo, cada servidor pode rodar sistemas operacionais e aplicativos de forma independente. Os componentes principais desta técnica são as máquinas virtuais (MVs), ambientes operacionais autossuficientes, que implementam um *hardware* virtual independente da plataforma; e o *hypervisor*, que é o *software* que desvincula o sistema operacional e os aplicativos de seus recursos físicos. O *hypervisor* possui *kernel* próprio, e roda diretamente no *hardware* da máquina, sendo inserido entre o *hardware* e o sistema operacional (GONCALVES; JUNIOR, 2010).

Banerjee et al. (2011), determina que uma plataforma de computação em nuvens totalmente otimizada quanto à virtualização deve prover as seguintes características:

- Virtualização computacional, onde cada servidor rode um *hypervisor*.
- Alocação de recursos eficientemente e dinamicamente.
- Isolamento entre as MVs, permitindo que dados e softwares sejam visualizáveis como *cache* em cada nó do servidor.
- Modelo *scale-out*, onde os nós possam ser adicionados à rede, e seus estados possam ser reconstruídos remotamente.
- Automatização relativa à instalação, atualização, gerência e recuperação do sistema operacional e aplicações.
- Plataforma rica em serviços (monitoramento, *log*, medição e faturamento) acessíveis através de protocolos de rede.
- Armazenamento com confiabilidade e escalabilidade para uma base de dados compartilhada.

Devido à grande dificuldade em comparar desempenhos de processamento em sistemas de computação em nuvens diferentes, as MVs implementadas neste trabalho são baseadas nas instâncias mais básicas disponibilizadas pela Amazon EC2, que utiliza a terminologia ECU (Elastic Compute Units), para tratar deste recurso físico (OSTERMANN et al., 2010). Logo, as capacidades da MVs implementadas são definidas por ECUs, onde 01 (um) ECU equivale à capacidade de um processador 1.0-1.2GHz 2007 *Opteron* ou *Xeon*. As instâncias são configuradas tal como observamos na Tabela 1.

MVs	ECUs (Cores)	RAM [GB])	Archi [bit]	Disk [GB]
m1.small	1 (1)	1,7	32	160
c1.medium	5 (2)	1,7	32	350
m1.large	4 (2)	15	64	850
m1.xlarge	8 (4)	15	64	1690
c1.xlarge	20 (8)	7	64	1690

Tabela 1: MVs utilizadas na plataforma Amazon EC2 (AMAZON, 2008).

A aquisição de novos *softwares* de aplicação ou de equipamentos específicos dá lugar à utilização de navegadores de Internet, possibilitando o acesso a vastos acervos de conteúdo disponibilizados sob demanda na rede. Tudo isso é possível a partir da convergência de tecnologias e serviços (Figura 5), que formam o alicerce para a próxima geração de dispositivos conectados (BANERJEE et al., 2011). Tal convergência proverá um desempenho extremamente alto a um custo relativamente baixo, enriquecendo a experiência do usuário, e simplificando as interações para acesso à informações e serviços personalizados em qualquer lugar, a qualquer momento.

Figura 5: Convergência de Tecnologias e Serviços (INFORMA, 2011).

A partir do momento em que as MVs encontram-se devidamente instaladas e operantes dentro da estrutura de computação em nuvens, devemos rodar os experimentos de avaliação de desempenho. Há várias possibilidades de se realizar aferições de desempenho em sistemas: *traces*, *benchmarks*, simulações (em vários níveis) e *mixes* (combinação de métodos). O principal meio para análise de dados utilizado por esta pesquisa é a execução de *benchmarks*, os quais irão fornecer dados estatísticos indispensáveis para a análise de desempenho em qualquer sistema.

2.1.3 *Benchmarks*

Benchmarks são *softwares* ou operações, a fim de avaliar o desempenho de um objeto

ou sistema em questão, realizando simulações, testes e ensaios neste. A partir da evolução das arquiteturas dos computadores, torna-se cada vez mais difícil comparar o desempenho de sistemas de computação diferentes. O *benchmark* realiza esta função, imitando um determinado tipo de comportamento de um componente ou sistema.

Os *Benchmarks* provem métodos de comparação de desempenho de subsistemas de diferentes arquiteturas e sistemas, sendo úteis para o entendimento de como um sistema reage sob a variação de condições, para testes de cenários no tratamento de *deadlocks*, análise de desempenho de utilitários, métodos de carregamento de dados, características de transição e ainda testa os efeitos da utilização de uma nova versão de um produto.

O HPC *Challenge benchmark* (HPCC) utiliza *kernels* computacionais reais, permitindo dados de entrada e tempos de execução de tamanhos variáveis, de acordo com a capacidade do sistema utilizado (DONGARRA; LUSZCZEK, 2010). O HPCC consiste em sete *benchmarks*: HPL, *STREAM*, DGEMM, PTRANS, *Random Access*, FFT e b_{eff}, responsáveis pela análise individual de cada componente crítico do sistema de computação em nuvens apresentados na Seção 3.1.2.

Além do HPCC, utilizamos outra suíte de *benchmarks*, o PTS. Esta suíte de *benchmarks* compreende mais de 130 testes para análise de um sistema, abrangendo destaques a vários recursos e aplicações-teste diferentes. Neste trabalho foram utilizados os *benchmarks RAM Speed SMP*, *PostMark*, e*Loopback TCP Network Performance*.

2.1.4 Metodologia DEA

A história da Análise Envoltória de Dados tem início com a publicação da tese de Edwardo Rhodes (CHARNES; COOPER; RHODES, 1978), a qual continha uma metodologia de comparação das eficiências de escolas públicas. O principal objetivo era estimar a eficiêcia técnica de cada uma das escolas sem que os pesos de cada insumo (*input*) ou produto (*output*) fossem informados arbitrariamente, e sem que houvesse a conversão para uma medida única para comparação entre as variáveis.

Para o cálculo da medida de eficiência foi utilizada uma abordagem que firma que um vetor *input-output* é tecnicamente eficiente, se e somente se:

- Nenhum dos *inputs* pode ser reduzido sem que algum *output* seja reduzido ou algum outro *input* seja aumentado.
- Nenhum dos *outputs* pode ser aumentado sem que algum *input* seja aumentado ou algum *output* seja reduzido.

Debreu (1951) propôs uma medida radial de eficiência técnica, que é o coeficiente de

utilização dos recursos, que busca a máxima redução equiproporcional de todos os insumos, ou seja, obter o menor valor possível de cada *input* mediante restrições mencionadas anteriormente. Pode ainda buscar a máxima expansão equiproporcional de todos os produtos, ou seja, conseguir o maior valor possível de cada *output*.

Conforme (LINS; MEZA; ANTUNES, 2000), Von Neumann utilizou modelos de programação matemática para representar sua análise de atividades e modelos de crescimento. Outros autores não menos importantes também fizeram referência à programação linear como instrumento de análise essencial para o modelo DEA, porém tais referências só foram amplamente aceitas após a publicação de Charnes e Cooper, com a utilização do modelo CCR (CHARNES; COOPER; RHODES, 1978).

As principais características da metodologia DEA são:

- Não há necessidade de converter todos os *inputs* e *outputs* nas mesmas unidades de medida;
- Os índices de eficiência são baseados em dados reais, ou seja, nos valores de cada *input* e *output*;
- Tem como resultado un único *input* virtual e um único *output* virtual;
- Considera que as observações díspares dentro de uma série não representam apenas grandes desvios em relação aos valores médios das variáveis, mas possíveis *benchmarks* (modelos de referência para outras unidades avaliadas);
- A metodologia DEA otimiza cada observação individualmente tendo como objetivo determinar uma fronteira linear por partes, compreendendo o conjunto de unidades Pareto-Eficiente.

Tal metodologia propicia a análise da eficiência comparativa de organizações complexas obtida pela revelação do desempenho de outras unidades, de modo que a referência não é obtida apenas teórica ou conceitualmente, mas através da observação das melhores práticas. As organizações que estiverem sob análise DEA são denominadas DMUs (*Decision Making Units*), e deverão utilizar recursos em comum e produzir os mesmos produtos. Com isto, serão definidas DMUs eficientes (máximo de *outputs* produzidos mediante os *inputs* utilizados) e ineficientes; as primeiras ficarão situadas na fronteira de eficiência, enquanto as últimas, abaixo da mesma fronteira.

Assim, DEA (*Data Envelopment Analysis*) é uma técnica matemática de programação linear que consiste em uma metodologia de apoio à decisão de natureza multicritério, que analisa múltiplos insumos e produtos simultaneamente, sendo capaz de modelar problemas do

mundo real atendendo à análise de eficiências. Pode ser implementada por meio de programação linear desenvolvida para converter medidas de múltiplos insumos e produtos em uma única medida de eficiência (FOCHEZATTO, 2010).

Conforme Meireles (2012), existem várias maneiras de determinar uma fronteira de eficiência, porém existem dois modelos da metodologia DEA amplamente aceitos: o BCC e o CCR. Ambos apresentam variações quanto a sua orientação, tal como será apresentado em seguida.

2.1.4.1 CCR - Modelo dos Multiplicadores (Orientado a *Input*)

O modelo CCR (CHARNES; COOPER; RHODES, 1978) constrói uma superfície linear por partes, a qual envolve os dados. Trabalha com retornos de escala constantes, ou seja, crescimentos proporcionais de *inputs* produzindo crescimentos proporcionais de *outputs*. Existem dois modelos de programação matemática para representar o modelo CCR: o dos multiplicadores e o do envelope.

$$\text{Maximizar ef(O)} = \frac{\sum_{j=1}^{s} u_j Y_{jO}}{\sum_{i=1}^{m} v_i X_{iO}} \qquad (5)$$

$sujeito\ a$:

$$\frac{\sum_{j=1}^{s} u_j Y_{jk}}{\sum_{i=1}^{m} v_i X_{ik}} \leq 1 \quad k = 1 \ldots n \qquad (6)$$

$$u_j \geq 0, \forall j \qquad (7)$$

$$v_i \geq 0, \forall i \qquad (8)$$

Onde:

$u_j = peso\ do\ output\ j$

$v_i = peso\ do\ input\ i$

$k \in \{1 \ldots n\}\ DMUs$

$j \in \{1 \ldots s\}\ outputs\ de\ cada\ DMUs$

$i \in \{1 \ldots m\}\ inputs\ de\ cada\ DMUs$

$Y_{jk} = valor\ do\ output\ j\ da\ DMU\ k$

$X_{ik} = valor\ do\ input\ i\ da\ DMU\ k$

Figura 6: Modelo dos Multiplicadores Orientados a Input (Fracionário) (CCR/M/I) (CHARNES; COOPER; RHODES, 1978).

Tem como objetivo encontrar os valores das variáveis u_j e v_j, que são seus respectivos pesos, maximizando a soma ponderada dos *outputs* dividida pela soma ponderada dos *inputs* da

DMU em questão, como é apresentado em (5) na Figura 6. Tal maximização tem restrição que o quociente em questão deve ser menor ou igual a um para cada DMU (6).

Desta maneira, o modelo permite que os pesos para cada variável (*input* e *output*) sejam escolhidos para cada DMU da forma que lhe for mais conveniente. O índice de eficiência encontrado é o índice apenas da DMU em questão (O), portanto esse procedimento deve ser repetido para cada uma das *k* DMUs.

Este modelo permite que sejam atribuídos pesos (multiplicadores) a todas as DMUs. A fim de maximizar a eficência, cada DMU define o seu próprio conjunto de pesos, uma vez que possuem um sistema de valores particular. Para tal, todas as DMUs devem ter uma eficiência inferior ou igual a 1.

$$\text{Maximizar ef(O)} = \sum_{j=1}^{s} u_j Y_{jO} \qquad (9)$$

$sujeito\ a$:

$$\sum_{i=1}^{m} v_i X_{iO} = 1 \qquad (10)$$

$$\sum_{j=1}^{s} u_j Y_{jk} - \sum_{i=1}^{m} v_i X_{ik} \leq 0 \quad k = 1 \dots n \qquad (11)$$

$$u_j \geq 0, \forall j \qquad (12)$$

$$v_i \geq 0, \forall i \qquad (13)$$

Onde:

$u_j = peso\ do\ output\ j$

$v_i = peso\ do\ input\ i$

$k \in \{1 \dots n\}\ DMUs$

$j \in \{1 \dots s\}\ outputs\ de\ cada\ DMUs$

$i \in \{1 \dots m\}\ inputs\ de\ cada\ DMUs$

$Y_{jk} = valor\ do\ output\ j\ da\ DMU\ k$

$X_{ik} = valor\ do\ input\ i\ da\ DMU\ k$

Figura 7: CCR - Modelo dos Multiplicadores Orientados a Input (Linear) (CCR/M/I) (CHARNES; COOPER; RHODES, 1978).

O modelo apresentado é um modelo de programação fracionária, e deve ser transformado em um modelo de programação linear. Deve-se linearizar também as restrições do problema conforme a Figura 7. Para tal, fixa-se o denominador da função objetivo em um valor constante (normalmente 1), como é visto em (10). A partir deste modelo é possível calcular além da eficiência de cada DMU, os respectivos pesos de cada *input* e *output*.

O modelo CCR dos multiplicadores (M) orientado a *input* (I) também é denominado CRS/M/I por assumir retornos de escala constantes (CRS) (*Constant Returns to Scale*).

2.1.4.2 CCR - Modelo do Envelope (Orientado a *Input*)

O modelo do envelope orientado a *input* tem como objetivo calcular a eficiência de uma DMU observando as formulações apresentadas na Figura 8, e tem como variáveis de decisão h_o e λ_k. A priori, h_o deve multiplicar todos os *inputs* tentando colocar a DMU na fronteira de eficiência.

$$\text{Minimizar ef(O)} = h_O \qquad (14)$$

$$sujeito\ a:$$

$$h_O X_{io} \geq \sum_{k=1}^{n} \lambda_k X_{ik} \quad \forall i \quad ou \quad h_O X_{io} - \sum_{k=1}^{n} \lambda_k X_{ik} \geq 0 \quad \forall i \qquad (15)$$

$$\sum_{k=1}^{n} \lambda_k Y_{jk} \geq Y_{jO} \quad \forall j \quad ou \quad -Y_{jO} + \sum_{k=1}^{n} \lambda_k Y_{jk} \geq 0 \quad \forall j \qquad (16)$$

$$\lambda_k \geq 0, \forall k \qquad (17)$$

Onde:

$h_O = eficiência\ da\ DMU\ O$

$\lambda_k = projeção\ da\ DMU\ k$

$k \in \{1 \dots n\}\ DMUs$

$j \in \{1 \dots s\}\ outputs\ de\ cada\ DMUs$

$i \in \{1 \dots m\}\ inputs\ de\ cada\ DMUs$

$Y_{jk} = valor\ do\ output\ j\ da\ DMU\ k$

$X_{ik} = valor\ do\ input\ i\ da\ DMU\ k$

Figura 8: Modelo do Envelope Orientado a *Input* (CCR/E/I) (CHARNES; COOPER; RHODES, 1978).

Desta maneira, o primeiro conjunto de restrições (15) faz com que a fronteira formada pelas DMUs eficientes não seja ultrapassada pela redução causada por h_o em cada um dos *inputs*. O segundo conjunto de restrições (16), garante que a redução provocada por h_o não altere os valores dos *outputs* da DMU.

Através da variável λ_k é indicado quais DMUs eficientes servirão de referência para a DMU que está sendo analisada. Caso λ_k seja igual a zero, considera-se que a DMU k não é referência para a DMU O. Portanto, quanto maior for λ_k, maior a importância como DMU referência.

Similarmente à classificação adotada para o modelo dos multiplicadores, o modelo do envelope é denominado CRS/E/I por assumir retornos de escala constantes (CRS), por ser

formulado a partir do modelo do envelope (E) e por ser orientado a *input* (I).

2.1.4.3 CCR - Modelo do Envelope (Orientado a *Output*)

Neste modelo os insumos são mantidos fixos, os produtos são maximizados, e as variáveis de decisão são as mesmas do modelo orientado a *input* (h_o e λ_k). Neste caso, h_o será o valor necessário para multiplicar os produtos mantendo constantes os insumos, possibilitando que a DMU atinja a fronteira de eficiência. Portanto, h_o será um valor maior que 1 de acordo com as restrições da Figura 9.

Maximizar h_O (18)

$sujeito\ a$:

$$X_{io} \geq \sum_{k=1}^{n} \lambda_k X_{ik} \quad \forall i \quad ou \quad X_{io} - \sum_{k=1}^{n} \lambda_k X_{ik} \geq 0 \quad \forall i \qquad (19)$$

$$\sum_{k=1}^{n} \lambda_k Y_{jk} \geq h_O Y_{jO} \quad \forall j \quad ou \quad -h_O Y_{jO} + \sum_{k=1}^{n} \lambda_k Y_{jk} \geq 0 \quad \forall j \qquad (20)$$

$\lambda_k \geq 0, \forall k$ (21)

Onde:

$h_O = \frac{1}{\text{ef(O)}}$

$\lambda_k = projeção\ da\ DMU\ k$

$k \in \{1 \dots n\}\ DMUs$

$j \in \{1 \dots s\}\ outputs\ de\ cada\ DMUs$

$i \in \{1 \dots m\}\ inputs\ de\ cada\ DMUs$

$Y_{jk} = valor\ do\ output\ j\ da\ DMU\ k$

$X_{ik} = valor\ do\ input\ i\ da\ DMU\ k$

Figura 9: Modelo do Envelope Orientado a *Output* (CCR/E/O) (CHARNES; COOPER; RHODES, 1978).

Similarmente às outras classificações de modelos, sua denominação será CRS/E/O por assumir retornos de escala constantes (CRS), por ser formulado a partir do modelo do envelope (E) e por ser orientado a *output* (O).

2.1.4.4 CCR - Modelo dos Multiplicadores (Orientado a *Output*)

$$\text{Minimizar } h_O = \sum_{i=1}^{m} v_i X_{iO} \qquad (22)$$

$sujeito\ a$:

$$\sum_{j=1}^{s} u_j Y_{jO} = 1 \qquad (23)$$

$$\sum_{j=1}^{s} u_j Y_{jk} - \sum_{i=1}^{m} v_i X_{ik} \leq 0 \quad k = 1 \ldots n \qquad (24)$$

$$u_j \geq 0, \forall j \qquad (25)$$

$$v_i \geq 0, \forall i \qquad (26)$$

Onde:

$u_j = peso\ do\ output\ j$

$v_i = peso\ do\ input\ i$

$k \in \{1 \ldots n\}\ DMUs$

$j \in \{1 \ldots s\}\ outputs\ de\ cada\ DMUs$

$i \in \{1 \ldots m\}\ inputs\ de\ cada\ DMUs$

$Y_{jk} = valor\ do\ output\ j\ da\ DMU\ k$

$X_{ik} = valor\ do\ input\ i\ da\ DMU\ k$

$$h_O = \frac{1}{\text{ef(O)}}$$

Figura 10: Modelo dos Multiplicadores Orientado a *Output* (CCR/M/O) (CHARNES; COOPER; RHODES, 1978).

Similarmente às outras classificações de modelos, sua denominação será CRS/M/O por assumir retornos de escala constantes (CRS), por ser formulado a partir do modelo dos multiplicadores (M) e por ser orientado a *output* (O). Podemos visualizar na Figura 10 o modelo dos multiplicadores orientados a *output*.

2.1.4.5 BCC - Modelo do Envelope (Orientado a *Input*)

O modelo BCC (BANKER; CHARNES; COOPER, 1984) difere do CCR quanto aos retornos de escala; no BCC o retorno de escala é variável, enquanto que no CCR é constante. Com isso, o modelo também é denominado VRS (*Variable Returns to Scale*). As DMUs que trabalham com *inputs* de valores baixos terão retornos de escala crescentes, e as que trabalham com *inputs* de valores altos terão retornos de escala descrescentes. Considera que um acréscimo no *input* poderá promover um acréscimo no *output*, não necessariamente proporcional. Sendo assim, este modelo é relevante para o estudo da eficiência por admitir retornos de escala

variáveis.

O modelo BCC surgiu como uma forma de eficiência resultante da divisão do modelo CCR em duas componentes: eficiência técnica e eficiência de escala. A eficiência técnica identifica a correta utilização dos recursos à escala de operação da DMU. A eficiência de escala é igual ao quociente da eficiência BCC com a eficiência CCR, e nos retorna a distância da DMU em análise até uma DMU fictícia.

Na orientação a *input*, o modelo do envelope adiciona mais uma restrição às já existentes no modelo CCR. A nova restrição adicionada (30), garante que o somatório dos λ_k seja igual a 1 tal como mostra a Figura 11.

$$\text{Minimizar ef(O)} = h_O \qquad (27)$$

$$sujeito\ a\text{:}$$

$$h_O X_{io} \geq \sum_{k=1}^{n} \lambda_k X_{ik} \quad \forall i \quad ou \quad h_O X_{io} - \sum_{k=1}^{n} \lambda_k X_{ik} \geq 0 \quad \forall i \qquad (28)$$

$$\sum_{k=1}^{n} \lambda_k Y_{jk} \geq Y_{jO} \quad \forall j \quad ou \quad -Y_{jO} + \sum_{k=1}^{n} \lambda_k Y_{jk} \geq 0 \quad \forall j \qquad (29)$$

$$\sum_{k=1}^{n} \lambda_k = 1 \qquad (30)$$

$$\lambda_k \geq 0, \forall k \qquad (31)$$

Onde:

$h_O = eficiência\ da\ DMU\ O$

$\lambda_k = projeção\ da\ DMU\ k$

$k \in \{1 \dots n\}\ DMUs$

$j \in \{1 \dots s\}\ outputs\ de\ cada\ DMUs$

$i \in \{1 \dots m\}\ inputs\ de\ cada\ DMUs$

$Y_{jk} = valor\ do\ output\ j\ da\ DMU\ k$

$X_{ik} = valor\ do\ input\ i\ da\ DMU\ k$

Figura 11: Modelo do Envelope Orientado a *Input* (BCC/E/I) (BANKER; CHARNES; COOPER, 1984).

Tal como apresentado anteriormente, a denominação do modelo obedece o padrão estabelecido; VRS/E/I por assumir retornos de escala variáveis (VRS), ser formulado a partir do modelo do envelope (E) e ser orientado a *input* (I).

2.1.4.6 BCC - Modelo do Envelope (Orientado a *Output*)

Na orientação a *output* também há a adição de uma restrição do somatório dos λ_k (35) tal como mostra a Figura 12.

Maximizar h_O (32)

sujeito a:

$$X_{io} \geq \sum_{k=1}^{n} \lambda_k X_{ik} \quad \forall i \qquad ou \qquad X_{io} - \sum_{k=1}^{n} \lambda_k X_{ik} \geq 0 \quad \forall i \qquad (33)$$

$$\sum_{k=1}^{n} \lambda_k Y_{jk} \geq h_O Y_{jO} \quad \forall j \qquad ou \qquad -h_O Y_{jO} + \sum_{k=1}^{n} \lambda_k Y_{jk} \geq 0 \quad \forall j \qquad (34)$$

$$\sum_{k=1}^{n} \lambda_k = 1 \qquad (35)$$

$$\lambda_k \geq 0, \forall k \qquad (36)$$

Onde:

$$h_O = \frac{1}{\text{ef(O)}}$$

$\lambda_k = projeção\ da\ DMU\ k$

$k \in \{1 \dots n\}\ DMUs$

$j \in \{1 \dots s\}\ outputs\ de\ cada\ DMUs$

$i \in \{1 \dots m\}\ inputs\ de\ cada\ DMUs$

$Y_{jk} = valor\ do\ output\ j\ da\ DMU\ k$

$X_{ik} = valor\ do\ input\ i\ da\ DMU\ k$

Figura 12: Modelo do Envelope Orientado a *Output* (BCC/E/O) (BANKER; CHARNES; COOPER, 1984).

Tal como apresentado anteriormente, a denominação do modelo obedece o padrão estabelecido; VRS/E/I por assumir retornos de escala variáveis (VRS), ser formulado a partir do modelo do envelope (E) e ser orientado a *input* (I).

2.1.4.7 BCC - Modelo dos Multiplicadores (Orientado a *Input*)

Através dos modelos do envelope orientados a *input* e *output* é possível gerar os modelos dos multiplicadores orientados a *input* e *output* respectivamente. Em ambos, a restrição do somatório dos λ_k transforma-se nas variáveis duais u e v dos modelos dos multiplicadores orientados a *input* e *output* respectivamente. Essas variáveis são conhecidas como fatores de escala, e podem ser variáveis ou constantes.

Na orientação a *input*, os possíveis valores de *u*:

- $u = 0$, indica retornos de escala constantes;
- $u > 0$, indica retornos de escala crescentes;
- $u < 0$, indica retornos de escala decrescentes;

$$\text{Maximizar ef(O)} = \sum_{j=1}^{s} u_j Y_{jO} + u \qquad (37)$$

$sujeito\ a$:

$$\sum_{i=1}^{m} v_i X_{iO} = 1 \qquad (38)$$

$$\sum_{j=1}^{s} u_j Y_{jk} - \sum_{i=1}^{m} v_i X_{ik} + u \leq 0 \quad k = 1 \ldots n \qquad (39)$$

$$u_j \geq 0, \forall j \qquad (40)$$

$$v_i \geq 0, \forall i \qquad (41)$$

Onde:

$u \in \Re, \quad u\ é\ irrestrito$

$u_j = peso\ do\ output\ j$

$v_i = peso\ do\ input\ i$

$k \in \{1 \ldots n\}\ DMUs$

$j \in \{1 \ldots s\}\ outputs\ de\ cada\ DMUs$

$i \in \{1 \ldots m\}\ inputs\ de\ cada\ DMUs$

$Y_{jk} = valor\ do\ output\ j\ da\ DMU\ k$

$X_{ik} = valor\ do\ input\ i\ da\ DMU\ k$

Figura 13: Modelo dos Multiplicadores Orientado a *Input* (BCC/M/I) (BANKER; CHARNES; COOPER, 1984).

Tal como apresentado anteriormente, a denominação do modelo obedece o padrão estabelecido; VRS/E/I por assumir retornos de escala variáveis (VRS), ser formulado a partir do modelo dos multiplicadores (M) e ser orientado a *input* (I).

2.1.4.8 BCC - Modelo dos Multiplicadores (Orientado a *Output*)

Tal como apresentado anteriormente, a denominação do modelo obedece o padrão estabelecido; VRS/E/I por assumir retornos de escala variáveis (VRS), ser formulado a partir do modelo dos multiplicadores (M) e ser orientado a *output* (O).

Na orientação a *output*, os possíveis valores de *v*:

- $v = 0$, indica retornos de escala constantes;
- $v > 0$, indica retornos de escala decrescentes;
- $v < 0$, indica retornos de escala crescentes;

$$\text{Minimizar ef(O)} = \sum_{i=1}^{m} v_i X_{iO} + v \qquad (42)$$

$$sujeito\ a:$$

$$\sum_{j=1}^{s} u_j Y_{jO} = 1 \qquad (43)$$

$$\sum_{j=1}^{s} u_j Y_{jk} - \sum_{i=1}^{m} v_i X_{ik} - v \leq 0 \quad k = 1 \dots n \qquad (44)$$

$$u_j \geq 0, \forall j \qquad (45)$$

$$v_i \geq 0, \forall i \qquad (46)$$

Onde:

$v \in \Re, \quad v\ é\ irrestrito$

$u_j = peso\ do\ output\ j$

$v_i = peso\ do\ input\ i$

$k \in \{1 \dots n\}\ DMUs$

$j \in \{1 \dots s\}\ outputs\ de\ cada\ DMUs$

$i \in \{1 \dots m\}\ inputs\ de\ cada\ DMUs$

$Y_{jk} = valor\ do\ output\ j\ da\ DMU\ k$

$X_{ik} = valor\ do\ inputi\ da\ DMU\ k$

Figura 14: Modelo dos Multiplicadores Orientado a *Output* (BCC/M/O) (BANKER; CHARNES; COOPER, 1984).

O modelo BCC/M/O (BANKER; CHARNES; COOPER, 1984) foi escolhido para este trabalho por permitir retornos variáveis de escala, podendo tais retornos crescerem, decrescerem e permanecerem constantes à medida em que a escala de produção é aumentada ou reduzida. A orientação a *output* foi escolhida por conta das variáveis de *input* serem fixas, não havendo controle sobre as variáveis de *input* (instâncias de MVs). Como o objetivo é a obtenção do melhor desempenho dos *benchmarks* (*outputs*) executados nas MVs é pretendido portanto, obter a maior quantidade de produtos mediante os insumos disponíveis.

2.2 Trabalhos Relacionados

Para o desenvolvimento deste trabalho houve a necessidade de um aprofundamento em diversas publicações pesquisadas em nichos de literatura acadêmica, providenciando o amadurecimento das ideias e embasamento teórico. A seguir, evidenciaremos o princípio fundamental de cada um dos trabalhos onde esta pesquisa está alicerçada.

2.2.1 *Iosup e Ostermann*

Ostermann et al. (2010) e Iosup et al. (2011), primeiramente criaram uma plataforma virtual a partir dos recursos da nuvem, utilizando a plataforma comercial Amazon EC2 (AMAZON, 2008). Várias instâncias são criadas, especificando o tipo e a imagem da MV que vai ser inicializada e instalada. No máximo 20 instâncias podem ser utilizadas simultaneamente por usuários normais, para não comprometer o desempenho do sistema. Assim, o recurso pode ser utilizado como um nó computacional via conexão SSH (LEHTINEN; LONVICK, 2006).

Um dos principais atrativos da computação em nuvens, é a provisão de recursos sob demanda sem tempo de espera adicional, diferentemente dos padrões de submissão de outros sistemas de larga-escala. O autor ainda estabelece períodos de tempo (longos ou curtos) para a aquisição/liberação dos recursos. Para os períodos curtos, uma ou mais instâncias são adquiridas e liberadas repetidamente em poucos minutos, seguindo um processo de *Poisson* com taxa de chegada menor que um segundo. Para os períodos longos, cada instância é adquirida e liberada a cada 2 minutos no período de uma semana (IOSUP et al., 2011).

Neste cenário a infraestrutura é compartilhada por várias tarefas independentes. Foram utilizados *benchmarks* tradicionais em conjuntos de tarefas para serem executadas isoladamente, reexecutando as amostras dos *workloads* MJMI. Para tal análise, foi utilizado o HPCC para todos os tipos de instâncias possíveis. Ao submeter o mesmo *workload* repetidamente para cada tipo de instância (MV), foram detectadas duas características de desempenho principais: A estabilidade do *makespan* do *workload*, e o *overhead* tanto da aquisição/liberação de recursos, como da execução de tarefas, tornando possível a análise da provisão de recursos em tempo-real (OSTERMANN et al., 2010).

Iosup et al. (2011) investiga e quantifica a presença de componentes proto-MTC (*Many-Task Computing*) em *workloads* de computação científica. Em seguida são validadas as performances de várias plataformas de computação em nuvem (Amazon EC2, Mosso, ElasticHost e GoGrid) utilizando os *benchmarks* da suíte HPCC. Mais uma vez este *benchmark* é utilizado, agora somente para *workloads* de múltiplas instâncias. Em seguida, o autor compara as performances (entre computação em nuvens e computação científica, *grids* e PPIs), baseando-se em amostras de simulações e em resultados de desempenho empíricos.

Em ambos os trabalhos, foi verificado que a computação em nuvens é uma alternativa viável para aplicações com *deadlines* curtos. Uma tarefa com tempo de resposta estável e baixo traz um atraso muito menor para qualquer modelo de nuvem, quando comparado ao ambiente científico. A tecnologia de computação em nuvens responde eficazmente aos critérios de estabilidade, escalabilidade, baixo overhead, e tempo de resposta. Além disso, uma plataforma de computação em nuvens não deixa tanto a desejar, em termos de desempenho, quando comparada a uma plataforma de computação científica. As estruturas de computação científica foram implementadas para tratar de *workloads* de grande capacidade e fluxo, e portanto tem um aproveitamento melhor nesta finalidade.

2.2.2 Cardoso

Cardoso (2011) referencia os *benchmarks* para a verificação do desempenho e das limitações da infraestrutura de computação em nuvens. Os *benchmarks* foram divididos em: *Benchmarks* de desempenho da implementação (que tem como propósito o entendimento do momento de implementação dos *clusters* virtuais), *benchmarks* individuais, e *benchmarks* de *cluster*. Todos trazem consigo uma noção das perdas introduzidas pela virtualização em ambientes individuais ou multiprocessados.

O autor ainda realiza simulações com o propósito de avaliar as métricas de uso de CPU/RAM, E/S de armazenamento, e de rede. Os *benchmarks* são divididos em três categorias de utilização: CPU, E/S, e Misto. Foi verificado que os testes de utilização de CPU possuem um pequeno *overhead* introduzido pela virtualização. *Benchmarks* de utilização de E/S e Mistos, apresentaram resultados não-consistentes onde as perdas de desempenho negativas mostram que a virtualização pode introduzir ganhos de desempenho.

Tal fato ocorre possivelmente pelo fato da virtualização criar um novo nível de *cache*, gerando um grande impacto no desempenho de E/S. Por outro lado, também há componentes que desempenham funções de E/S com grandes perdas de cache, deteriorando o desempenho e tornando a *cache* inútil. É difícil, portanto, prever como uma tarefa específica que utiliza uma grande quantidade de execuções de E/S vai se comportar.

2.2.3 Huber

Huber et al. (2011) destaca o aumento da complexidade e dinamicidade na implementação de servidores virtualizados. O aumento na complexidade é causado pela introdução paulatina de recursos virtuais, e pela lacuna deixada pela alocação de recursos lógicos e físicos. O aumento na dinamicidade é causado pela falta de controle direto sobre o *hardware* físico, e pelas interações complexas entre aplicações e *workloads*, que compartilham a infraestrutura física introduzindo novos desafios na gestão do sistema.

Resultados de experimentos utilizando *benchmarks* mostraram que o *overhead* de desempenho para virtualização de CPU possui taxa em torno de 5%. Da mesma maneira, o *overhead* de virtualização de memória, E/S de rede, e de armazenamento, atingem 40%, 30%, e 25% respectivamente. Além da execução dos testes, foram feitas comparações com outros modelos de *hypervisor* a fim de especificar o desempenho individual de cada um, considerando taxas de erro de desempenho (HUBER et al., 2010).

2.2.4 Comentários

Para atender aos requerimentos mínimos do objeto de pesquisa dos trabalhos de Iosup et al. (2011) e Ostermann et al. (2010), relacionados na Seção 2.2.1, foi utilizada uma plataforma física de data center disponibilizada pelo grupo de pesquisa de segurança da informação (InSeRT) na Universidade Estadual do Ceará. Foram geradas cinco instâncias de MVs baseadas no padrão fornecido pela Amazon EC2 para a realização das simulações. A contribuição destes trabalhos fica marcada pela metodologia e avaliação das métricas envolvidas, além do pioneirismo relativo à ideia de avaliar o desempenho de sistemas de computação em nuvem.

Neste trabalho submetemos vários *workloads* para cinco tipos de instâncias diferentes, interconectadas e replicadas para prover uma simulação mais real do ambiente de nuvem. Além do HPCC, o PTS também foi utilizado para possibilitar uma maior cobertura dos recursos avaliados. O *makespan* manteve-se estável (constante), e o overhead atribuído à virtualização considerado pequeno, porém suficiente para alterar substancialmente os resultados de cada recurso em comparações com outros sistemas não-virtualizados.

Tal qual a pesquisa de Cardoso (2011), utilizamos as suítes de *benchmark* HPCC e PTS. Dividimos os *benchmarks* de acordo com os recursos, e os recursos em categorias de utilização. Consideramos também as perdas introduzidas pela virtualização (*overhead*), e a não-consistência de alguns resultados, mostrando que a virtualização pode introduzir ganhos de desempenho.

Por último, Huber et al. (2011) nos mostra as taxas de *overhead* introduzidas pela virtualização, e que estas devem ser levadas em consideração quando avaliamos o desempenho dos recursos virtualizados. Assim, procedemos ao Capítulo 3, onde será proposta a metodologia de avaliação de desempenho para sistemas de computação em nuvens. Mais detalhes sobre os experimentos realizados serão disponibilizados na Seção 4.1.

3 PROPOSTA DE METODOLOGIA PARA ANÁLISE DE DESEMPENHO EM SISTEMAS DE COMPUTAÇÃO EM NUVENS

Através das pesquisas realizadas para concepção deste trabalho, constatou-se que a literatura estudada não trata da questão da avaliação do desempenho de sistemas de computação em nuvens no contexto dos recursos de infraestrutura através de *benchmarks*. Mesmo com as vantagens do sistema de computação em nuvens, a literatura normalmente referencia estruturas de computação científica, as quais oferecem uma grande capacidade de armazenamento através de técnicas de processamento paralelo.

Neste capítulo é descrita a metodologia proposta para a análise de desempenho de sistemas de computação em nuvens. Tal metodologia avaliará as métricas obtidas nos *benchmarks* executados em conjunto com a metodologia DEA, gerando pesos por meio de uma análise da eficiência inerente à cada recurso em cada instância. Com a obtenção destas métricas, é formulada uma expressão matemática que nos aponte os níveis de desempenho globais dos recursos do sistema.

3.1 Metodologia

O hábito de hospedar dados na nuvem já é comum para muitos usuários e entusiastas, mesmo que estes não tenham conhecimento sobre o real funcionamento da estrutura, ou como avaliar qual a opção mais eficaz atualmente em funcionamento para utilização. Para compreendermos melhor esta tecnologia, em termos de valor agregado ao desempenho, propõe-se uma metodologia de análise do desempenho de sistemas de computação em nuvem, que leva em consideração os pontos críticos de *hardware* e rede.

Primeiramente, após o conhecimento da infraestrutura de *hardware* computacional, implementamos as MVs baseadas no modelo fornecido pela Amazon EC2 (AMAZON, 2008), apresentado anteriormente na Tabela 1. O desempenho total destes recursos não é utilizado, visto que a virtualização gera *overheads* de comunicação no gerenciamento dos recursos, fato que será considerado na formulação matemática proposta na Seção 3.1.6. Após a alocação dos recursos necessários para a implementação, foram instaladas as suítes de *benchmark* utilizadas para a execução dos testes.

De acordo com Jain (2008), o intervalo de confiança só se aplica para amostras grandes, as quais devem ser consideradas a partir de trinta iterações. Portanto, realizamos os experimentos referentes a cada recurso de cada instância pelo menos trinta vezes, para garantir a obtenção de um intervalo de confiança satisfatório (95%). Após a execução dos testes, calcula-se a média e o intervalo de confiança dos resultados obtidos para apresentar o alto nível de confiabilidade dos resultados através dos gráficos da Seção 4.2. Maiores detalhes sobre a configuração dos

benchmarks serão apresentados no Capítulo 4.

No sistema de computação em nuvens utilizado neste trabalho, utilizaremos *benchmarks* para medir o desempenho dos recursos básicos de *hardware*. A fim de ponderar os experimentos executados optamos pela execução da metodologia DEA, utilizando o modelo BCC orientado a *output* (BCC-O), que envolve um princípio alternativo para extrair informações de uma população de resultados.

Para determinar as eficiências dos recursos analisados em cada MV, executamos a metodologia DEA (modelo BCC-O), determinando os pesos inerentes às MVs e recursos analisados conforme foi explicado na Seção 2.1. Utilizamos os resultados de cada iteração de *benchmark* em cada MV como entrada, obtendo índices de eficiênca como resultado. Calculamos a média ponderada a partir destes índices, obtendo os pesos para cada *benchmark*. Assim, aplicamos este procedimento na formulação que será apresentada mais adiante na Seção 3.1.6.

Em suma, analisa-se o desempenho de uma nuvem computacional simulando o comportamento de aplicações através da execução dos *benchmarks*. A partir dos resultados obtidos, foi feita uma análise da eficiência, atribuindo-se pesos à cada experimento. Daí propôs-se uma formulação que evidenciou a proporção de consumo de cada recurso da plataforma, considerando os *overheads* associados. Com isso, espera-se poder prover uma nova visão sobre a análise de desempenho de sistemas de computação em nuvens.

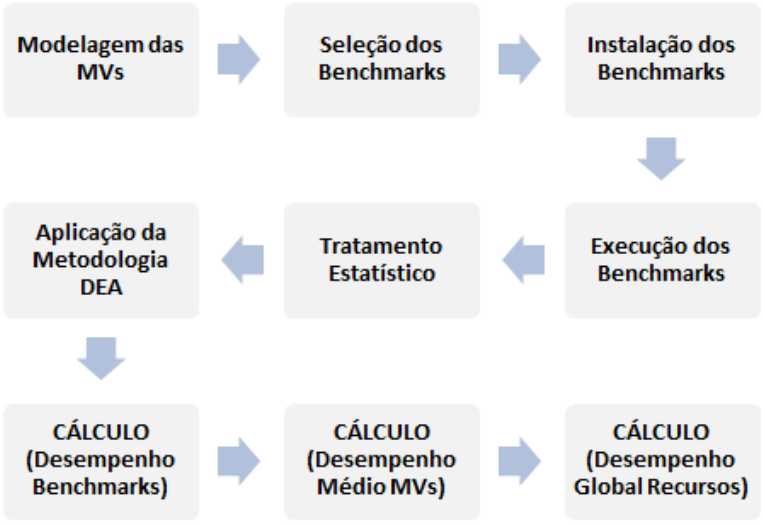

Figura 15: Fluxograma dos Processos da Metodologia Proposta.

Estabelecemos portanto, uma ordem de execução das atividades apresentadas nesta pesquisa, a fim de esclarecer os passos da metodologia proposta. A Figura 15 representa o processo de avaliação do desempenho de um sistema de computação em nuvens desde a modelagem das MVs, passando pela manipulação dos *benchmarks*, o tratamento estatístico, a aplicação da metodologia DEA, até o cálculo dos índices de desempenho.

3.1.1 Suítes de Benchmark

Benchmarking é a execução de *softwares* ou operações, para avaliar o desempenho de um sistema ou recurso realizando experimentos, testes e simulações que provenham métodos de comparação de desempenho de subsistemas de diferentes arquiteturas, gerando assim, valor agregado aos componentes analisados.

Nesta pesquisa foram utilizadas duas suítes de *benchmarks*, as quais medirão o desempenhos dos pontos críticos do sistema de computação em nuvens, o HPCC e o PTS. Para que rodem de maneira consistente, os *benchmarks* requerem a disponibilidade da biblioteca de comunicação de dados para computação distribuída MPI (*Data Envelopment Analysis*) (DONGARRA et al., 1992) e da biblioteca matemática BLAS (*Basic Linear Algebra Subprograms*) (LAWNSON et al., 1979), as quais utilizamos nos testes executados.

3.1.2 HPCC - High Performance Computing Challenge

O HPCC implantou uma visão inédita da caracterização do desempenho de um sistema, capturando dados sob condições similares às dos sistemas reais, e permitindo uma variedade de análises de acordo com a necessidade do usuário final (DONGARRA; LUSZCZEK, 2006). O HPCC utiliza *kernels* computacionais reais, permitindo dados de entrada de tamanhos variáveis e tempos de execução variante de acordo com a capacidade do sistema utilizado (DONGARRA; LUSZCZEK, 2010).

Tanto nos testes preliminares executados localmente, como nos testes finais realizados na estrutura de computação em nuvens, cada um dos testes compreendidos nesta suíte obtiveram resultados favoráveis à sua utilização, mostrando independência e adaptabilidade para funcionar entre os nós da nuvem. A seguir, apresentaremos os testes e suas funcionalidades.

1. HPL

   O HPL (PETITET et al., 2008) é um pacote de software que tem como principal característica a medição da taxa de execução de ponto flutuante através de sistemas de equações lineares densos e aleatórios para resolução de matrizes. Utiliza aritmética de precisão dupla (*double*) de 64 bits em computadores de memória distribuída, e possui um programa de teste e temporização para quantificar a precisão da solução obtida, bem como o tempo que o sistema leva para computá-la.

   Sua performance vai depender de uma grande variedade de fatores. Mesmo assim, com algumas medidas restritivas referentes à rede de interconexão, permite que a execução do algoritmo seja escalável. A eficiência paralela é mantida constante no que diz respeito ao uso de memória por processador.

2. DGEMM

   É a rotina principal do *benchmark Linpack*, sendo responsável pela execução de várias outras rotinas. Simula múltiplas execuções de ponto flutuante sobrecarregando o processador através de multiplicações entre matrizes de precisão dupla (DONGARRA et al., 1989). Considera a participação expressiva do *buffer* de memória, a partir do posicionamento do ponteiro da mesma com relação à distância do início de cada linha/coluna das matrizes no desempenho da operação.

3. PTRANS

   Normalmente chamado de *Parkbench Suite*, possui vários *kernels* onde pares de processadores comunicam-se entre si simultaneamente, testando a capacidade de comunicação total da rede. O *PTRANS* realiza multiplicação de matrizes densas e transposição de matrizes paralelas (HEY; DONGARRA; R., 1996) aplicando técnicas de *interleaving*.

4. FFT

   Realiza a medição da taxa de execução de ponto flutuante através de transformadas de *Fourier* discretas (DFT) unidimensionais de precisão dupla em arrays de números complexos (FRIGO; JOHNSON, 2007).

5. *STREAM*

   O *STREAM* (MCCALPIN, 2002) mede a largura de banda de memória que sustenta a comunicação com o processador (em GB/s). Ainda mede o desempenho de quatro operações estruturais de vetor longo. O tamanho do *array* é definido para que seja maior do que o tamanho da cache da máquina onde está sendo realizado o teste. Toma como parâmetro a memória de comunicação direta com o processador, privilegiando a atualização do *buffer* de memória através da interdependência entre memória e processador.

6. *Random Acess*

   O *benchmark Random Acess* (KOESTER; LUCAS, 2008) mede o desempenho de atualizações do buffer da memória aleatória (principal), e da memória de acesso (cache) em sistemas multiprocessados. Ambas sustentam a comunicação entre os processadores do sistema, direcionando esforços para o desempenho das aplicações em execução. O processador é bastante decisivo para o desempenho do *benchmark*, controlando o fluxo e favorecendo a utilização de maiores linhas de memória.

   Retorna o resultado em GUPS (*Giga Updates Per Second*), calculado através da identificação das locações de memória atualizadas em um segundo. Obtivemos a quantidade de GUPS identificando o número de locais de memória que podem ser atualizados aleatoriamente em um segundo, onde uma atualização consiste em uma operação RMW (leitura-modificação-escrita) controlada pelo *buffer* de memória e pelo processador.

7. b_{eff}

   O *Bandwidth Efficiency* mede a eficiência da largura de banda (efetiva) de comunicação através do tempo de latência estimado para que se processe, transmita e receba uma mensagem padrão, em sistemas computacionais paralelos ou distribuídos (RABENSEIFNER; SCHULZ, 1999). São utilizadas mensagens de vários tamanhos, padrões e métodos de comunicação diferentes, levando em consideração uma aplicação real. O tamanho da mensagem enviada pelo *benchmark* vai depender da relação memória-processador de cada instância testada, sendo representado pelo resultado desta relação dividido por 128.

3.1.3 PTS - Phoronix Test Suite

Além do HPCC, utilizamos outra suíte de *benchmarks*, o PTS. Esta suíte também foi utilizada para possibilitar uma maior cobertura dos recursos avaliados e a fim de reforçar a análise de desempenho dos recursos previamente apresentados na Seção 1.1. Compreende mais de 130 testes para análise de um sistema, o que motivou a escolha inicial de um escopo maior de *benchmarks* da suíte PTS para a avaliação do sistema. A fim de otimizar a utilização dos *benchmarks*, avaliamos a relevância dos resultados obtidos a partir da execução da metodologia DEA.

Cada resultado gerado pelos *benchmarks* foi disposto como *output* no *solver* de programação linear referenciado na Seção 4.4. Este *solver* foi utilizado para obter os índices de eficiência e os pesos atribuídos a cada um dos *benchmarks* avaliados. De acordo com estes resultados, selecionamos os testes de acordo com a sua importância dentro do conjunto de *benchmarks*, minimizando as possíveis inconsistências quanto às unidades de medida de cada teste. Finalmente, os *benchmarks* selecionados para este trabalho foram: *Loopback* TCP, *RAMspeed SMP*, e *PostMark* apresentados na Seção 3.1.3.

1. *Loopback TCP Network Performance*

   É uma simulação simples de conectividade ponto-a-ponto que mede o desempenho do adaptador de rede em um teste de *loopback* através do desempenho do pacote TCP (LARABEL; TIPPETT, 2008a). Este teste é aperfeiçoado neste *benchmark* para transmitir 10GB via *loopback*. O dispositivo de *loopback* é uma interface virtual de rede implementada por *software* completamente integrada à infraestrutura de rede do sistema. Qualquer tráfego que um programa envie para a interface de *loopback* é recebido imediatamente na mesma interface.

2. *RAMspeed SMP*

   Mede o desempenho da interação entre as memórias cache e principal de um sistema multiprocessado (HOLLANDER; BOLOTOFF, 2002). O *benchmark* aloca um determinado espaço de memória, e inicia um processo de escrita ou leitura utilizando blocos de dados de 1Kb até o limite do array, verificando a velocidade das execuções dos subsistemas de memória principal e *cache*. Assim como as outras simulações de memória citadas anteriormente, também depende do processamento atrelado à memória, controlando o fluxo das execuções repassadas ao *buffer*. A medição é feita pela quantidade de blocos de dados processados pela memória em GB/s.

3. PostMark

   Este *benchmark* cria um grande *pool* de arquivos pequenos em constantes alterações, a fim de medir a taxa de transações dos *workloads*, simulando um grande servidor de e-mail para Internet. O *pool* de arquivos de texto é configurável e compatível com qualquer sistema de arquivos. As transações de criação, deleção, leitura, e anexação possuem tamanhos mínimo e máximo fixados entre 5Kb e 512Kb. Quando um arquivo é criado, o tamanho inicial do arquivo e do texto contido nele são selecionados aleatoriamente. A deleção de arquivos simplesmente escolhe aleatoriamente um arquivo do *pool*, e o deleta. O mesmo procedimento é válido para ação de leitura.

   Por fim, para anexar dados a um arquivo abre-se um arquivo do *pool* aleatoriamente, procura-se pela parte final deste, escrevendo uma quantidade aleatória de dados que não ultrapasse o limite superior do arquivo. O Postmark (KATCHER, 1997) executa 25.000 transações com 500 arquivos simultaneamente, e após o término das transações os arquivos remanescentes são deletados também, produzindo estatística referente a deleção contínua destes arquivos.

Listados na Tabela 2, os *benchmarks* apresentados acima foram escolhidos para serem utilizados no experimento após simulações locais, na estrutura de nuvem, e simulações de eficiência relativa através da metodologia DEA. Nenhum teste da suíte HPCC foi descartado, sendo todos de fundamental importância para a análise de desempenho quantitativo e qualitativo deste trabalho. Os resultados sempre foram coesos, de fácil manipulação, e com terminologias compatíveis.

A suíte PTS apresentou alguns problemas de terminologia, impossibilitando sua utilização por falta de uma referência no *hardware* analisado. Outros problemas surgiram quanto à similaridade dos resultados das simulações realizadas na suíte HPCC, o que apresentaria duplicação destes. E por último, observamos através da metodologia DEA que alguns *benchmarks* não possuíam eficiência relevante em alguns resultados, que não foram tão satisfatórios no sistema de computação em nuvens quanto quando executados localmente.

RECURSO	BENCHMARK	UNIDADE DE MEDIDA
CPU	HPL DGEMM PTRANS	GFLOPs
	FFT	GB/s
MEM	STREAM RAM Speed SMP	GB/s
	Random Access	GUPS
STO	PostMark	Transações/s
NET	b_{eff}	μs
	Loopback TCP	s

Tabela 2: Benchmarks utilizados por Recurso, e suas respectivas Terminologias.

RECURSO		OVERHEAD (%)
E/S	Memória	40
	Rede	30
	Storage	25
CPU	Processamento	5

Tabela 3: Overheads de Virtualização (HUBER et al., 2011).

3.1.4 Recursos

Para simplificar a organização da análise do desempenho dos recursos de uma estrutura de computação em nuvens, podemos dividí-los em dois grupos de requisitos: recursos de CPU e de E/S. Estudos de desempenho utilizando *benchmarks* gerais, mostram que o *overhead* devido à virtualização de recursos de CPU tem taxa em torno de 5% conforme mencionado na Seção 2.2.3. Dentre os recursos básicos de uma infraestrutura de computação em nuvens o processamento representa um consumo menor do que o esperado, pois hospeda o *hypervisor* diretamente controlando o *hardware* e gerenciando os sistemas operacionais vigentes, consequentemente apresenta um *overhead* menor. A virtualização também impõe *overheads* aos recursos de E/S (memória, Internet, e armazenamento) (HUBER et al., 2011) conforme mostra a Tabela 3.

Aplicações de nuvem possuem requerimentos específicos, de acordo com o seu objetivo principal. Presente de forma crítica em todas estas aplicações, a rede determina a velocidade com a qual o restante dos recursos de E/S vão trabalhar. Em outras palavras, a rede deve prover capacidade, disponibilidade, e eficiência suficiente para alocar recursos sem maiores atrasos. Como foi explicado no Seção 2.1, computação em nuvens é um sistema de computação de Internet, portanto, criticamente dependente do funcionamento da rede, serviço que deve ser mantido operante (*zero downtime*) e transparente ao usuário. Os recursos de E/S representam a maior parcela do consumo total da infraestrutura.

Uma das aplicações mais populares hoje em dia é o armazenamento de conteúdo online, onde o usuário pode acessar conteúdo de qualquer lugar do mundo com conexão à Internet. Além deste serviço, blogs, *e-mails*, serviços de compartilhamento de fotos e vídeo, e aplicativos de edição de textos, planilhas, e apresentações, são serviços em que o armazenamento em disco (*storage*) é afetado, e tem um consumo significativo com relação ao sistema como um todo. O desempenho dos recursos de armazenamento são tão dependentes da taxa de atualização do *buffer* de memória, quanto da taxa de processamento que alimenta o *buffer*.

Por último, ainda com relação aos recursos de E/S, a memória é o recurso mais requerido pelos serviços providos pela nuvem. Em sistemas distribuídos, é considerada uma questão crítica, pois trabalha juntamente com o processamento na atualização da execução dos aplicativos, das requisições do usuário, na gravação e leitura de dados que chegam através do adaptador de rede ou do componente de armazenamento. Tantas funções concentradas, muitas vezes sobrecarregam o recurso, que representa o maior gargalo de toda a infraestrutura de nuvem.

De acordo com o que foi exposto, infere-se que cada recurso de *hardware* disponível dentro da infraestrutura de computação em nuvens possui uma parcela única referente à sua utilização, porém apresentando interdependências entre si. Atribuímos então, pesos baseados na importância de cada recurso para o funcionamento eficaz da estrutura através da metodologia DEA, após a execução de *benchmarks* específicos.

3.1.5 Aplicação do DEA

A metodologia DEA foi aplicada neste trabalho a fim de parametrizar os valores gerados pelos *benchmarks* para cada recurso em cada instância. Os termos principais para a ponderação requerida na formulação proposta, são gerados a partir da execução desta metodologia, sendo possível parametrizar a eficiência de cada experimento executado. Para a aplicação da metodologia DEA utilizamos o modelo BCC-O, que permite que a metodologia seja orientada a *output*, já que neste trabalho as variáveis de *input* (recursos das MVs) são fixas, conforme visto na Seção 2.1.4.

O modelo matemático BCC-O consiste no cálculo dos pesos das variáveis de *input* (recursos de cada MV), e *output* (resultados dos *benchmarks*). Na função objetivo do modelo BCC, minimiza-se a soma ponderada dos *inputs* (produto do valor do *input* pelo seu respectivo peso), sujeito a quatro restrições, apresentadas na Figura 16 e explicadas a seguir:

1. A soma ponderada dos *outputs* seja igual a 1 (Equação 1).
2. A subtração entre a soma ponderada dos *outputs*, a soma ponderada dos *inputs*, e o fator de escala (v), deve ser menor ou igual a zero (Inequação 2).
3. O peso dos *outputs* deve ser maior ou igual a zero (Inequação 3).

4. O peso dos *inputs* deve ser maior ou igual a zero (Inequação 4).

$$\text{Minimizar ef(O)} = \sum_{i=1}^{m} v_i X_{iO} + v$$

$$sujeito\ a:$$

$$\sum_{j=1}^{s} u_j Y_{jO} = 1 \qquad (1)$$

$$\sum_{j=1}^{s} u_j Y_{jk} - \sum_{i=1}^{m} v_i X_{ik} - v \leq 0 \quad k = 1 \ldots n \qquad (2)$$

$$u_j \geq 0, \forall j \qquad (3)$$

$$v_i \geq 0, \forall i \qquad (4)$$

Onde:

$v \in \Re, \quad v\ é\ irrestrito$

$u_j = peso\ do\ output\ j$

$v_i = peso\ do\ input\ i$

$k \in \{1 \ldots n\}\ DMUs$

$j \in \{1 \ldots s\}\ outputs\ de\ cada\ DMUs$

$i \in \{1 \ldots m\}\ inputs\ de\ cada\ DMUs$

$Y_{jk} = valor\ do\ output\ j\ da\ DMU\ k$

$X_{ik} = valor\ do\ inputi\ da\ DMU\ k$

Figura 16: Modelo Matemático da Metodologia DEA (BCC-O) (BANKER; CHARNES; COOPER, 1984).

Ao executar o modelo visto acima em um *solver* de programação linear, obtem-se os valores dos pesos. A restrição da inequação 2 será executada para cada uma das 150 iterações das instâncias executadas. O fator de escala (v) vai determinar apenas se os retornos de produção serão crescentes, decrescentes ou constantes para um conjunto de insumos e produtos. Como os pesos foram as únicas variáveis utilizadas na formulação proposta no Capítulo 3.1.6, tal fator não será considerado neste trabalho (v = 0).

O modelo permite que os pesos sejam escolhidos para cada DMU (iterações de MVs) da forma que lhe for mais conveniente. Os pesos calculados devem ser maiores ou iguais a zero, conforme as inequações 3 e 4. Os índices de eficiência de cada DMU também são calculados pela função objetivo. Assim, em DEA, o número de modelos a ser resolvido é igual ao número de DMUs do problema.

A fim de obter o melhor desempenho resultante dos *benchmarks* (*outputs*) executados nas MVs, os pesos são obtidos por meio de uma média ponderada de acordo com a representatividade de cada teste para o sistema. Os maiores valores possuirão maiores pesos. Considerando que cada um dos dez *benchmarks* foi executado trinta vezes para cada uma das cinco MVs utilizadas, contabiliza-se 1500 iterações. Cada uma destas iterações teve seu respectivo peso calculado pela metodologia DEA. Executamos um *solver* de programação linear para calcular a soma ponderada dos *inputs*, e *outputs*, obedecendo às restrições da metodologia.

Quanto às restrições, primeiramente temos que a soma ponderada dos *outputs* deve ser igual a 1, estabelecendo um parâmetro para a atribuição dos pesos em cada MV. Os pesos dos *inputs* e *outputs* devem ser maiores ou iguais a zero, pois não existem pesos negativos, impondo uma condição de existência. E por último, a subtração entre as somas ponderadas dos *inputs* e *outputs* e o fator de escala, deve ser menor ou igual a zero. O fator de escala é um valor relativo ao rendimento do *output*, e não será considerado neste trabalho, pois o termo que nos interessa é o peso atribuído a cada um dos *outputs*. O modelo permite que os pesos sejam escolhidos para cada DMU (iterações de instâncias) da forma que lhe for mais conveniente.

3.1.6 Formulação

Um sistema de computação em nuvens possui estrutura complexa por ser distribuído, e possuir alocação de recursos dinâmica. Os recursos requeridos são alocados automaticamente de acordo com a necessidade. Todos eles possuem um nível de *overhead* padrão, e um nível de importância variável de acordo com o direcionamento da aplicação hospedada no ambiente. Para analisar o desempenho deste tipo de sistema é necessária a formulação de uma expressão matemática que evidencie os níveis de utilização medidos, e as interações entre os recursos do sistema.

Deve-se levar em consideração que a execução dos *benchmarks* vão simular uma aplicação que sobrecarregará o recurso avaliado. Adota-se "IAD_{G_R}" como Índice de Avaliação do Desempenho Global por Recurso, cuja variável assumirá o valor resultante do produto entre o Índice de Desempenho Real por recurso"IDR_R" e o Índice de Avaliação de Desempenho por Recurso em cada instância "IAD_{R_j}". A Equação 3.1 apresenta a fórmula matemática do Índice de Avaliação de Desempenho Global por Recurso:

$$\mathbf{IAD_{G_R}} = \mathbf{IDR_R} \times \mathbf{IAD_{R_j}} \tag{3.1}$$

Temos que IDR_R é o Índice de Desempenho Real por recurso, que resulta no percentual de funcionamento real do recurso avaliado. O termo "IDR_R" é calculado pela diferença entre o desempenho teórico máximo do recurso (100%), e os *overheads* (Ov_R) associados para cada

recurso em funcionamento na estrutura de computação em nuvens, apresentados na Tabela 3. A Equação 3.2 apresenta a fórmula matemática do Índice de Desempenho Real por Recurso:

$$\mathbf{IDR_R} = (\mathbf{100\%} - \mathbf{Ov_R\%}) \tag{3.2}$$

O IAD_{R_j} é o Índice de Avaliação de Desempenho Médio por recurso (R) em cada instância (j) utilizada no experimento, que avalia o desempenho médio dos experimentos. O termo é calculado como mostra a Equação 3.3 por meio da média de cada IAD_{iR_j}, que representa o Índice de Avaliação de Desempenho do benchmark (i) por recurso (R) em cada instância (j). O termo é calculado pelo somatório do produto entre os pesos (U_{iR_j}), obtidos através da metodologia DEA para o *benchmark* (i) por recurso (R) em cada instância (j), e os valores (X_{iR_j}) obtidos através do *benchmark* (i) por recurso (R) em cada instância (j) como mostra a Equação 3.4.

$$\mathbf{IAD_{R_j}} = \mathbf{IAD_{iR_j}} \div \mathbf{n_j} \tag{3.3}$$

$$\mathbf{IAD_{iR_j}} = \sum(\mathbf{U_{iR_j}} \times \mathbf{X_{iR_j}}) \tag{3.4}$$

U_{iR} é o peso do *benchmark* (i) atribuído ao recurso (R). E X_{iR} é o valor obtido através do *benchmark* (i) atribuído ao recurso (R). O termo "n" é a quantidade de MVs em que os *benchmarks* executados foram hospedados para cada recurso (R), ou seja, as cinco MVs implementadas com base no modelo da Amazon EC2, apresentadas na Tabela 5 no próximo capítulo. Procuramos reunir os experimentos que simulassem mais fielmente as tarefas processadas por um sistema de computação em nuvens de maneira geral, atendendo às demandas do sistema.

Conjuntos	**Descrição**
j = {m1.small, c1.medium, m1.large, m1.xlarge, c1.xlarge}	Instâncias (MVs)
R = {CPU, MEM, STO, NET}	Recursos
CPU_i = {HPL, DGEM, FFT, PTRANS}	*Benchmarks* de Processamento
MEM_i = {STREAM, Random Access, RAM Speed SMP}	*Benchmarks* de Memória
STO_i = {PostMark}	*Benchmarks* de Armazenamento
NET_i = {b_{eff}, Loopback TCP}	*Benchmarks* de Rede

Tabela 4: Conjuntos dos Parâmetros da Formulação Proposta

Os *benchmarks* foram escolhidos de acordo com o perfil do ambiente a ser analisado. Caso o perfil de um outro ambiente de nuvens seja o armazenamento de dados, devem ser escolhidos *benchmarks* que avaliem este recurso crítico específico da estrutura. Os *benchmarks* (i) são classificados com base nos recursos avaliados, de acordo com as instâncias (j) e recursos (R) utilizados. Tais parâmetros são agrupados nos conjuntos apresentados na Tabela 4.

As suites de *benchmark* foram configuradas para simularem o comportamento de cada recurso da infraestrutura de computação em nuvens, gerando *workloads* entre os nós da rede. Calcularemos o Índice de Avaliação do Desempenho dos *benchmarks* (i) para cada recurso (R) em cada instância (j), considerando a execução de cada *benchmark* para seu respectivo recurso dentro de cada instância do experimento representada pela Equação 3.4. As expressões para IAD_{iR_j} são desenvolvidas para cada recurso da seguinte forma:

$IAD_{iCPU_j} = (U_{HPL} \text{ x } X_{HPL}) + (U_{DGEMM} \text{ x } X_{DGEMM}) + (U_{FFT} \text{ x } X_{FFT}) + (U_{PTRANS} \text{ x } X_{PTRANS})$
$IAD_{iMEM_j} = (U_{STREAM} \text{ x } X_{STREAM}) + (U_{RA} \text{ x } X_{RA}) + (U_{RSMP} \text{ x } X_{RSMP})$
$IAD_{iSTO_j} = (U_{BB} \text{ x } X_{BB}) + (U_{PM} \text{ x } X_{PM})$
$IAD_{iNET_j} = (U_{BE} \text{ x } X_{BE}) + (U_{LTCP} \text{ x } X_{LTCP})$

Após calculado o IAD_{iR_j}, calcula-se a média dos resultados encontrados para os recursos (R) em cada instância (j), onde "n" é o número de instâncias (MVs) utilizadas no experimento. A formulação é representada pela Equação 3.3, e desenvolvida a seguir:

$IAD_{CPU_j} = \sum IAD_{iCPU_j} \div n_j$
$IAD_{MEM_j} = \sum IAD_{iMEM_j} \div n_j$
$IAD_{STO_j} = \sum IAD_{iSTO_j} \div n_j$
$IAD_{NET_j} = \sum IAD_{iNET_j} \div n_j$

De posse dos resultados obtidos a partir das execuções dos *benchmarks* e da execução da metodologia DEA, foi possível calcular o Índice de Avaliação de Desempenho dos *benchmarks* (i) para cada recurso (R) em cada instância (j) (IAD_{iR_j}). Consequentemente também foi possível calcular o Índice de Avaliação de Desempenho médio de cada recurso (R) em cada instância (j) (IAD_{R_j}). Com base nas notações apresentadas neste capítulo, o próximo passo consiste na resolução da notação de desempenho global, dada pela Equação 3.1, e desenvolvida a seguir:

$IAD_{G_{CPU}} = (IDR_{CPU}) \text{ x } IAD_{CPU}$
$IAD_{G_{MEM}} = (IDR_{MEM}) \text{ x } IAD_{MEM}$
$IAD_{G_{STO}} = (IDR_{STO}) \text{ x } IAD_{STO}$
$IAD_{G_{NET}} = (IDR_{NET}) \text{ x } IAD_{NET}$

No Capítulo 4 será descrito um estudo de caso como forma de validar a metodologia proposta, servindo como exemplo para a compreensão da formulação. Além disso, ainda serão apresentadas tabelas e gráficos para uma melhor visualização dos resultados obtidos.

4 ESTUDO DE CASO

Este capítulo apresenta o estudo de caso no qual aplicamos a metodologia proposta. Apresentaremos o cenário/ambiente utilizados, os experimentos realizados considerando os requerimentos de computação em nuvens que analisaremos no estudo, e faremos uma discussão das simulações realizadas.

4.1 Ambiente

As simulações foram realizadas no ambiente de *cloud* disponibilizado pelo grupo de pesquisa de segurança da informação (InSeRT) na Universidade Estadual do Ceará (INSERT, 2007). A infraestrutura de computação em nuvens utilizada consiste em um *data center Dell Power Edge* M1000e (DELL, 2011b) (*enclosure*) com 4 *Blade Servers* M710HD (DELL, 2011c). Duas *Blades* possuem processador *Intel Xeon Six-Core* (INTEL, 2010) x5640 de 2.26GHz com memória principal de 64GB (8x8GB) DDR3 de 1333MHz; mais duas *Blades* com processador *Intel Xeon Six-Core* x5660 2.8GHz, com memória principal de 128GB (16x8GB) DDR3 de 1333MHz. Todas as *Blades* possuem 12MB de memória *cache* e disco rígido de 146GB SAS com taxa de transferência de 6GBps. Neste experimento utilizaremos apenas as *Blades* de maior capacidade para hospedagem e execução dos experimentos.

O *data center* é interligado a um *Storage Dell Compellent* (DELL, 2011a) com seis discos de 600GB SAS com taxa de transferência de 6GBPs funcionando à rotação de 15.000 rpm; mais seis discos de 2TB SAS com taxa de transferência de 6GBps funcionando à rotação de 7.200 rpm. O sistema operacional é o próprio *hypervisor* VMware ESXi 5.0.0 rodando em nível 0 (abstraindo diretamente a partir do *hardware*). Esta abordagem otimiza tanto o desempenho quanto o custo final (TCO), e é chamada *bare-metal* (vide Figura 17).

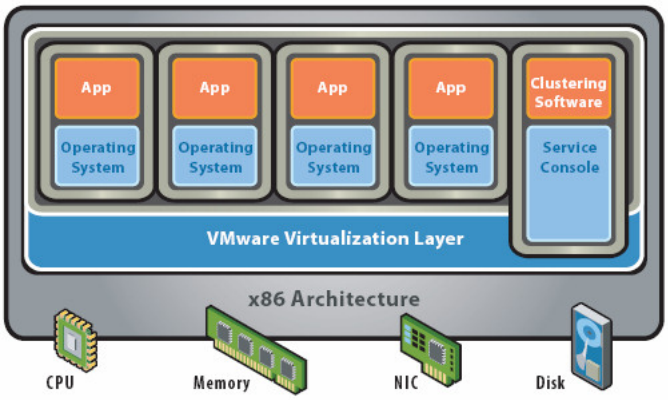

Figura 17: Abordagem de Virtualização *Bare-Metal* (VMWARE, 2012).

MV	(CPU x Core) = UC	RAM	ARQ.[Bits]	DISCO [GB]
m1.small	(1 x 1) = 1	1,7	32	160
c1.medium	(2 x 3) = 6	1,7	32	350
m1.large	(2 x 2) = 4	15	64	850
m1.xlarge	(4 x 2) = 8	15	64	1690
c1.xlarge	(7 x 3) = 21	7	64	1690

Tabela 5: MVs utilizadas no Estudo de Caso.

Para o nosso experimento, criamos ambientes homogêneos de 1 a 21 *cores* (ECUs) baseados em cinco instâncias Amazon EC2. A partir da implementação destas MVs apresentadas na Tabela 5, optamos pela utilização do sistema operacional de código-aberto *Linux Ubuntu 11.10* em cada uma das cinco instâncias criadas. Para possibilitar a execução dos experimentos propostos, foram instaladas as suítes de *benchmark* referenciadas na Seção 3.1.1, as quais tem como requisito fundamental, a instalação das bibliotecas matemáticas BLAS e *OpenMPI* para que possam trabalhar em um ambiente de Computação Distribuída.

4.2 Execuções dos *Benchmarks*

Como foi exposto na Seção 3.1.4, especificamente na Tabela 3, a representatividade do *overhead* operacional dos recursos para um sistema de computação distribuída é fundamental para avaliar o desempenho da infraestrutura como um todo. Para tal, foram executadas simulações de desempenho (*benchmarks*) dos recursos essenciais que compõem o sistema de computação em nuvens a fim de gerar valor para a formulação da metodologia proposta no Capítulo 3. Todos os valores serão dados em percentuais através de comparação com os picos teóricos de cada recurso.

Ao executar a suíte de *benchmarks* HPCC, houve primeiramente a necessidade de configurar seu arquivo de entrada, para que as simulações se ajustem ao tamanho do problema que queremos, e consequentemente, também não sobrecarreguem o sistema com capacidades de *workloads* que provoquem *buffer overflow*. A seguir veremos os campos principais para configuração do arquivo de entrada "hpccinf.txt" e em seguida o seu conteúdo.

1. TAMANHO DO PROBLEMA (Dimensão da Matriz N)
   Para que se possa medir o melhor desempenho do sistema, deve-se configurar um maior tamanho de problema possível. Porém, se o tamanho do problema for muito grande o desempenho cai, o que requer sensibilidade no momento do ajuste do *benchmark*, para que o sistema não interfira negativamente na execução do mesmo. A quantidade de memória utilizada pelo HPCC é praticamente do mesmo tamanho da matriz coeficiente.

   No caso, temos dois nós com 128 GB de memória cada, o que corresponde a 256 GB no

total. O *workload* cria uma matriz de tamanho representado por "$N^2 \times 8$" em bytes, onde "N" equivale à capacidade de cada nó. Considerando os dois nós, temos 32 milhões de elementos (de 8 bytes) de dupla precisão. Como temos 6 processadores por nó, múltiplos processos vão se espalhar pelos nós, e a memória alocada disponível para cada processo vai ser determinante. Consciente destes fatores, foram atribuídos dois valores a N: 2000 e 4000.

2. TAMANHO DO BLOCO DE DADOS (NB)
   O valor do parâmetro NB é o tamanho do bloco de dados que o sistema vai trabalhar, tanto quanto à distribuição dos dados na rede como para a granularidade computacional. Quanto menor o seu valor, melhor o balanceamento de carga. Por outro lado, um valor muito pequeno de NB limita o desempenho computacional, pois quase não haverá reutilização de dados em uma alta utilização de memória. O melhor valor atribuído a este parâmetro é variável, dentro do intervalo de 32 a 256. Foi preferível deixar o valor padrão (80), para que não houvesse problemas de desempenho.

3. GRADES DE PROCESSOS (P x Q)
   De acordo com as especificações definidas anteriormente, atribui-se valores a P e Q, que trabalharão como linha e coluna da matriz (N) e do bloco de dados (NB) dimensionados. Definido o número de grades de processamento (4), temos que P e Q devem ser valores aproximados, portanto atribuímos 1x3, 2x1, 3x2, e 2x2.

```
 1 HPLinpack benchmark input file
 2 Innovative Computing Laboratory, University of Tennessee
 3 HPL.out      output file name (if any)
 4 8            device out (6=stdout,7=stderr,file)
 5 2            # of problems sizes (N)
 6 2000 4000    Ns
 7 1            # of NBs
 8 80           NBs
 9 0            PMAP process mapping (0=Row-,1=Column-major)
10 4            # of process grids (P x Q)
11 1 2 3 2      Ps
12 3 1 2 2      Qs
13 16.0         threshold
14 1            # of panel fact
15 2            PFACTs (0=left, 1=Crout, 2=Right)
16 1            # of recursive stopping criterium
17 4            NBMINs (>= 1)
18 1            # of panels in recursion
19 3            NDIVs
20 1            # of recursive panel fact.
21 2            RFACTs (0=left, 1=Crout, 2=Right)
22 1            # of broadcast
23 3            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
24 1            # of lookahead depth
25 0            DEPTHs (>=0)
26 2            SWAP (0=bin-exch,1=long,2=mix)
27 64           swapping threshold
28 0            L1 in (0=transposed,1=no-transposed) form
29 0            U  in (0=transposed,1=no-transposed) form
30 1            Equilibration (0=no,1=yes)
31 16           memory alignment in double (> 0)
```

Figura 18: Arquivo de Configuração do HPCC "hpccinf.txt"

.

4.2.1 Desempenho de CPU

Para este recurso teremos apenas os *benchmarks* HPL, DGEMM, FFT, e PTRANS representando o HPCC. A suíte PTS apresentou problemas de terminologia, e eficiência relativa; impossibilitando a parametrização dos resultados obtidos, e produzindo resultados não tão relevantes no sistema de computação em nuvens quanto aqueles executados localmente.

Dentre as MVs utilizadas, temos que a c1.xlarge possui mais unidades computacionais (21), o que nos leva a crer que esta deverá obter melhor desempenho em meio às outras MVs. Porém, cada benchmark tem sua peculiaridade, e não somente analisa o desempenho do processamento, como também analisa a sua interdependência e equilíbrio para com os recursos de memória.

Cada um dos experimentos tem como objetivo a execução de instruções complexas, afim de sobrecarregar a CPU, simulando uma grande demanda de processos. Os resultados destes experimentos são dados em GFLOPs e GB/s, tendo como referência o desempenho de pico para este processador em ambos os casos. De acordo com (INTEL, 2010) estes valores são de 67.2 GFLOPs e 6.4 GB/s respectivamente. Portanto, quanto maiores os resultados obtidos, melhor. O valor em GFLOPs ainda pode ser calculado levando em consideração três parâmetros principais:

1. Número de FLOPS/ciclo.
   Para o processador utilizado no experimento, este valor é igual a 4 (SUGREE, 2007).

2. Frequência de *Clock* em GHz.
   Conforme exposto na Seção 4.1, este valor é igual a 2,8GHz.

3. Número de Cores por Processador.
   Este valor é igual a 6.

O *benchmark* HPL vai executar múltiplas operações de ponto flutuante através de um sistema de equações lineares. Conforme executada a simulação, foi verificado que este seguiu a lógica de concentração de processamento (melhor desempenho para uma quantidade maior de processamento proprietário da MV). Como podemos visualizar no gráfico da Figura 19, o melhor resultado foi obtido na MV c1.xlarge que possui maior poder de processamento. Foram alcançados 18,48 GFLOPs, que equivale a 27,5% do desempenho de pico teórico.

Consequentemente, as outras MVs tiveram desempenho relativo à quantidade de unidades processamento atreladas. A única ressalva da análise do HPL diz respeito à MV m1.large (4UC), que apresenta desempenho "não consistente" ligeiramente maior que a VM c1.medium (6UC). Tal fato não pode ser determinado de maneira exata, pois a virtualização pode introduzir ganhos de desempenho criando um novo nível de cache sobre o *hypervisor*.

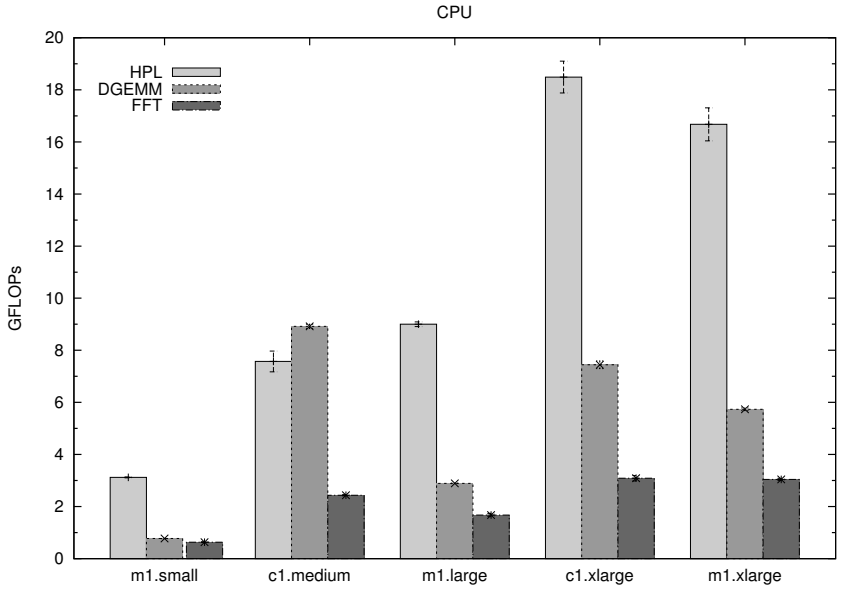

Figura 19: HPCC *Benchmark* (**CPU**): HPL x DGEMM x FFT (GFLOPS)

O *benchmark* DGEMM também simula múltiplas operações de ponto flutuante, sobrecarregando o processador através da multiplicação de matrizes de precisão dupla. Possui participação expressiva do *buffer* de memória, controlando o fluxo de dados. Observa-se então no gráfico da Figura 19, que o melhor desempenho encontrado entre as MVs foi em c1.medium (8,9 GFLOPs), equivalente a 13% do desempenho de pico. A MV em questão, possui uma relação processador/memória mais alta que todas as outras (6 UC/1,7 GB).

Como mantivemos a *cache* fixa, somente a MV c1.xlarge, que também possui uma alta relação processador/memória (21 UC/7 GB), consegue um resultado mais próximo. Uma MV de menor custo (c1.medium) atingindo um desempenho superior a todas as outras denota uma inconsistência. Porém como possui menor capacidade, apesar da ótima relação de equilíbrio processador/memória pode comprometer a alocação de recursos requerida para sistemas de computação em nuvens. O restante das MVs obteve resultados insignificantes frente às MVs citadas.

Em seguida, o gráfico da Figura 19 referencia o benchmark FFT, que também executará operações de ponto flutuante através de Transformadas de Fourier em *arrays* de números complexos. Bem como o HPL, o FFT leva em consideração a hierarquia de concentração de processamento. O melhor desempenho mais uma vez foi obtido na instância c1.xlarge atingindo 3,08 GFLOPs (4,59%), resultado similar à MV m1.xlarge. Fato que comprova que o experimento mais uma vez privilegia o equilíbrio da relação entre memória e processador das MVs, tornando o *buffer* de dados o principal responsável pelos resultados similares encontrados. O restante das MVs obedeceu a hierarquia de processamento.

Observamos no gráfico da Figura 20 os resultados do *benchmark* PTRANS através da

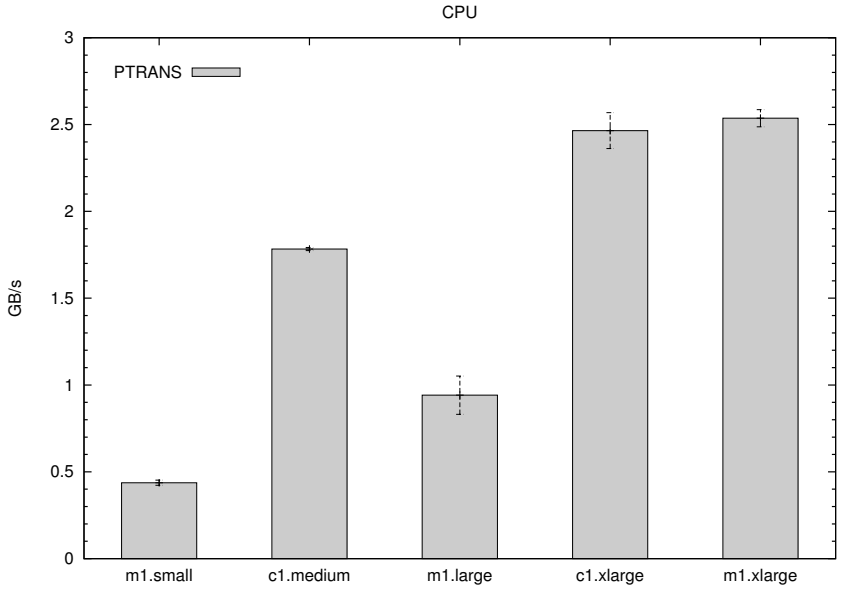

Figura 20: HPCC *Benchmark* (**CPU**): PTRANS (GB/s)

transposição de matrizes paralelas e multiplicação de matrizes densas, aplicando *interleaving*. O melhor resultado foi atingido pela instância m1.xlarge (8 UC/15 GB) com 2,53 GB/s, que equivale a 39% do desempenho de pico. Nota-se novamente que o experimento considera o equilíbrio entre memória e processador, acessando vários locais de memória adjacente entre os processadores (nós) da rede. Logo em seguida, a instância c1.xlarge atinge 2,46 GB/s (38%), um resultado bastante similar que remete à relação processador/memória. Um resultado justo e devidamente justificado, logo que o *benchmark* estabelece a medida do real "contato"entre memória e processadores.

As MVs implementadas no sistema são baseadas nas instâncias da Amazon EC2, e classificadas quanto o seu tamanho e preço. Logo, podemos inferir que as instâncias de maior capacidade, as quais também são as de maior custo, são as mais recomendadas para um maior desempenho de processamento, salvo exceções de inconsistência, porém deve-se averiguar com cuidado a relação entre memória e processador para que não haja inconsistência no processamento de dados. Uma MV com capacidade de memória díspar em relação à de processamento, pode acarretar *scrambling* ou *buffer overflow*. Ao adquirir alguma MV entre estas, o usuário que possui aplicações críticas de processamento terá realmente o que lhe é prometido.

4.2.2 Desempenho de Memória

A questão da memória em sistemas distribuídos é crítica. Os experimentos tem como objetivo avaliar o desempenho da memória, parametrizar os níveis de *cache* presentes no processamento, medir a largura de banda e as taxas de atualização do *buffer* de memória. Para tanto, levaremos em consideração o valor de pico atingido pelo servidor *Blade* utilizado no

trabalho, que é de 32 GB/s (DELL, 2011c).

Quanto às MVs utilizadas, as pertencentes à família m1 possuem maior concentração de memória (15 GB), o que nos leva a crer que estas deverão obter melhor desempenho em meio às outras MVs. Porém, cada *benchmark* tem suas peculiaridades. Cada um deles não analisa somente o desempenho da memória, como também verifica a interdependência com o processamento, recurso essencial para o funcionamento do sistema. Quanto aos *benchmarks* pertencentes à suíte HPCC, temos que o *STREAM* medirá a largura de banda de memória que sustenta a comunicação com o processador, pois toma como parâmetro a memória de comunicação direta com este.

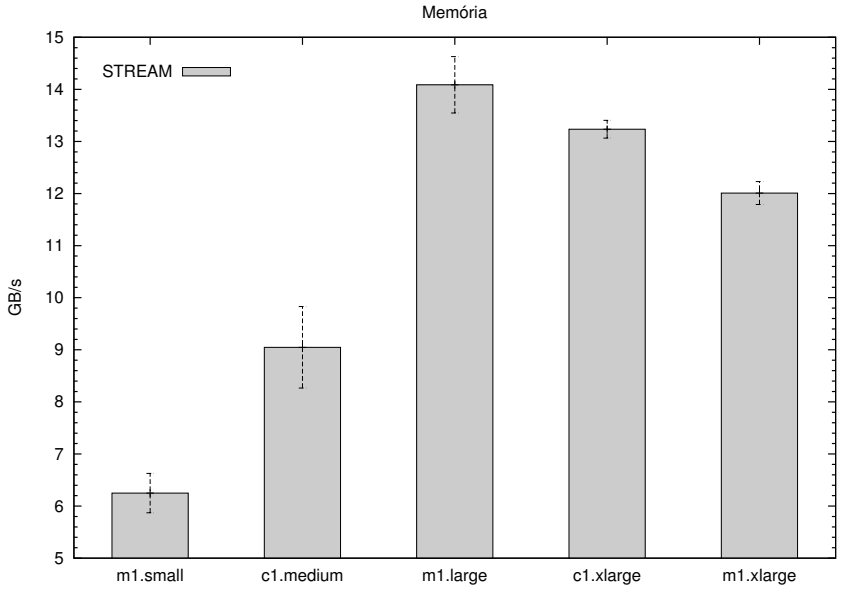

Figura 21: HPCC *Benchmark* (**Memória**): STREAM (GB/s)

Podemos inferir que este privilegia a atualização do *buffer* de memória através da relação memória/processador durante o experimento. Conforme o gráfico 21, visualizamos que a MV m1.large (15 GB/4 UC) atingiu o melhor desempenho da simulação (14 GB/s), obtendo 43% do desempenho de pico. As instâncias de maior capacidade ($*$.xlarge) aparecem em seguida, com 41% e 37% cada.

Teoricamente podemos destacar tanto m1.large quanto a m1.xlarge (15 GB/8 UC) como os melhores resultados, apesar de bastante próximos. Porém, a instância c1.xlarge (7 GB/21 UC) apresenta uma inconsistência (já que possui resultados melhores que m1.xlarge), aqui justificada por possuir um número de processadores muito superior a qualquer outra instância. Assim, c1.xlarge requer uma maior largura de banda de memória, paralelizada entre os processadores, superando os 15 GB de sustentação da MV m1.xlarge; entretanto, não supera a relação memória/processador de m1.large. O restante das MVs possui resultados distantes das instâncias maiores.

A relação custo-benefício neste caso, é algo difícil de parametrizar, pois as instâncias mais caras foram superadas por uma instância de preço médio. Mas é necessário observar, contudo, que o desempenho da MV vai depender do equilíbrio dos recursos críticos. Para uma aplicação que necessite de bastante largura de banda de memória é necessário um processamento que supra as necessidades do buffer.

Em outras palavras, quanto maior a relação memória/processador, melhor; salvo exceções em que a capacidade de processamento supera em muito a capacidade de memória, gerando *overflow*.

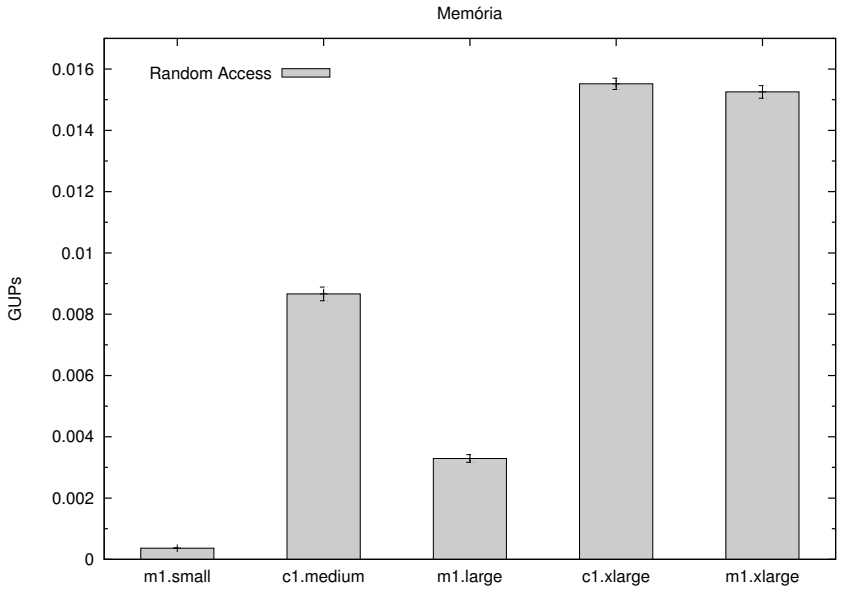

Figura 22: HPCC *Benchmark* (**Memória**): Random Access (GUPs)

O Random Access é um *benchmark* da suíte HPCC que mede o desempenho de atualizações do *buffer* da memória aleatória (principal) e da memória de acesso (*cache*) em sistemas multiprocessados. Observa-se no gráfico da Figura 22 que as duas instâncias maiores (*.xlarge) atingiram uma taxa maior de atualizações (GUPS) em relação às outras instâncias, ambas em torno de 0,015 GUPS, o que equivale a 11% do desempenho de pico (1,3 GUPS).

Mesmo com uma capacidade de memória menor do que a MV m1.xlarge, a MV c1.xlarge possui uma taxa levemente superior, o que reforça a explicação de que o processador é decisivo quanto ao desempenho do *benchmark*, já que o primeiro controla o fluxo da memória a ser utilizada pelas aplicações. Assim, o restante dos resultados segue descendente conforme a capacidade de processamento.

A relação custo-benefício mostra as duas instâncias mais caras atingindo o maior resultado, o que é perfeitamente normal. A instância c1.medium, apesar de ser mais barata, possui um desempenho superior à m1.large, denotando uma inconsistência. Este fato é justificado tanto pela capacidade computacional como pela relação memória/processador da primeira ser

superior à da última. A MV m1.large representa alta probabilidade de *scrambling*.

Partindo para a suíte de *benchmarks* PTS, verifica-se no gráfico da Figura 23 que a taxa de atualização do *buffer* do *benchmark RAM Speed SMP* se mantém praticamente constante, independentemente do tipo do blocos de dados ser *Integer* ou *Float*. O maior resultado atingido é por meio da instância c1.xlarge com 9 GB/S e 10 GB/s para *Float* e *Integer* respectivamente. Esta MV não possui a maior concentração de memória entre as MVs, mas possui maior poder de processamento dentre todas, permitindo um controle maior do fluxo dos blocos de dados passantes pelo *buffer* de memória.

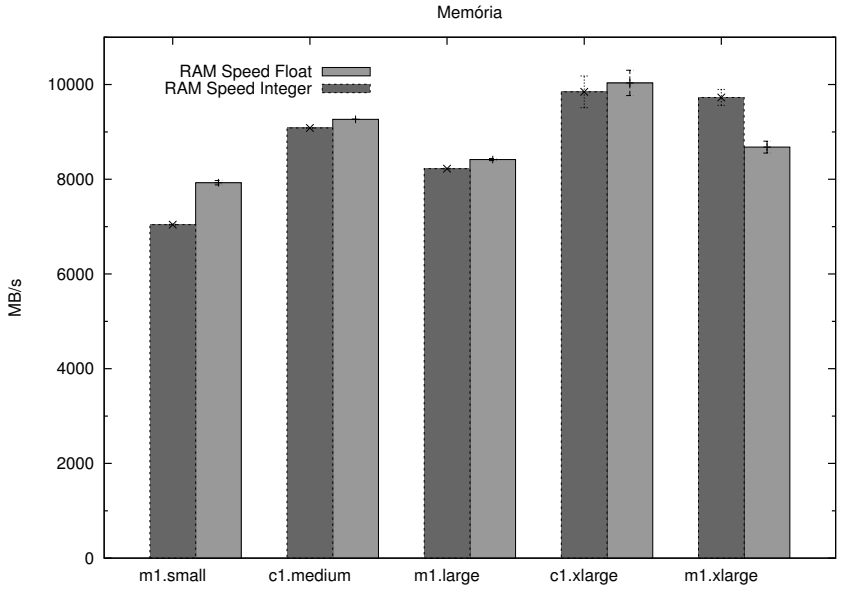

Figura 23: *Phoronix Test Suite* (**Memória**): RAM Speed SMP Integer x Float (MB/s)

Novamente, a questão do equilíbrio de recursos fazendo a diferença nos resultados, mesmo que mínima, pode ser justificada pelo equilíbrio proporcionado pela estrutura virtualizada das MVs. Mesmo que algumas possuam pouca concentração de memória, são compensadas pelo controle de fluxo proporcionado pelo processador e vice-versa, justificando assim, o equilíbrio dos resultados.

4.2.3 Desempenho de Disco

A suíte de *benchmarks* HPCC não possui nenhum experimento voltado para o desempenho e interdependência de disco. Utilizamos então, a suíte PTS para expor as impressões sobre tal recurso. O *benchmark* que obteve os melhores resultados durante a fase de simulações locais foi o *PostMark*, que cria um grande *pool* de arquivos em constantes alterações a fim de medir as taxas de transações dos *workloads*, simulando um grande servidor de e-mail para Internet. Temos que o desempenho de pico teórico é de 6 TB/s (DELL, 2011c), e o teste divide o bloco de transferência em 500 arquivos de até 512 Kb; o que resulta em um valor de pico em

torno de 24000 Transações/s.

Observando a Figura 24, percebe-se três resultados praticamente similares, o que nos mostra apenas que as maiores instâncias produzem os maiores resultados, onde foram processados com sucesso apenas 14% dos *workloads* da simulação. Tal percentual nos dá valores em torno de 3.500 transações por segundo. A instância m1.large porém, possui bem menos capacidade de armazenamento que as instâncias *.xlarge mesmo mantendo um resultado bastante satisfatório devido à sua alta capacidade de *buffer* disponibilizada pela alta concentração de memória. Mais uma vez observa-se que o equilíbrio das MVs é determinante para o desempenho do *benchmark*.

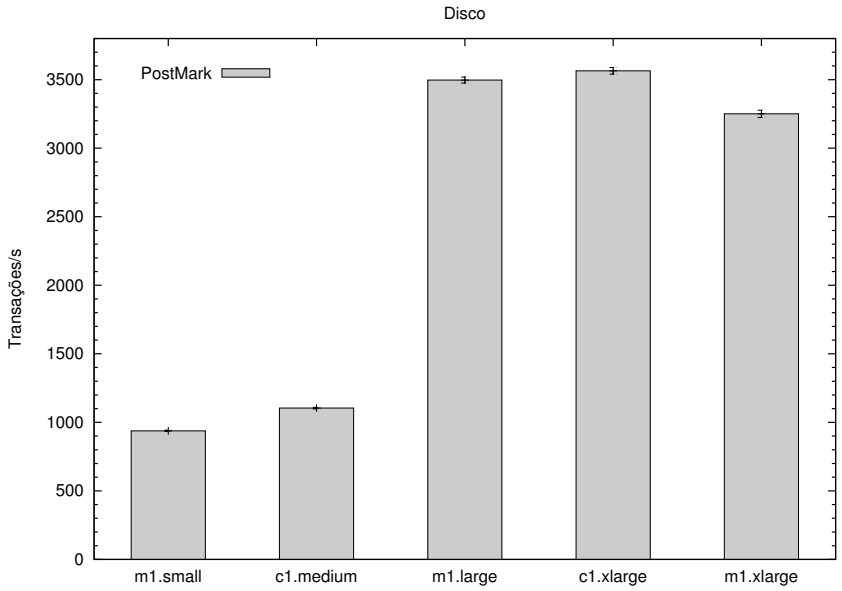

Figura 24: *Phoronix Test Suite* (**Storage**): PostMark (Transações/s)

Podemos inferir que o desempenho de disco é bastante dependente primeiramente da taxa de atualização do *buffer* de memória, e em seguida da taxa de processamento que alimenta este *buffer*. Verifica-se que as instâncias que possuem maior poder de armazenamento tiveram sempre resultados ótimos, porém uma MV com metade desta capacidade conseguiu resultados similares apenas contando com um bom desempenho da relação de equilíbrio entre memória e processamento.

4.2.4 Desempenho de Rede

Os benchmarks de rede são executados levando em consideração o desempenho do acesso aos recursos interconectados, ou seja, a capacidade da rede de prover recursos sem atrasos ou maiores complicações que comprometam a disponibilidade do serviço em tempo-real. Na suíte HPCC, temos o *benchmark* b_{eff} que mede o tempo de latência estimado para que se processe uma mensagem-padrão, considerando a largura de banda da rede em sistemas paralelos

e/ou distribuídos.

Calculando os tamanhos das mensagens para cada MV, observa-se no gráfico da Figura 25 que as instâncias m1 carregarão as mensagens de tamanho maior. Como o fator rede é bastante aleatório devido às diversas configurações possíveis de implantação, atribuiremos um percentual relativo a cada resultado obtido com relação a um padrão, onde foi estabelecido que o desempenho ótimo gira em torno de 0,7μs, e o pior em torno de 100ms.

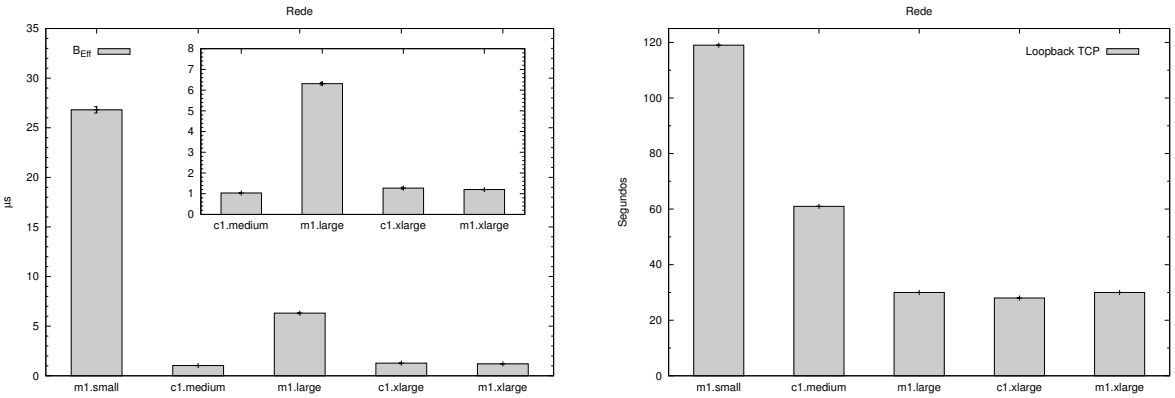

Figura 25: HPCC *Benchmark* (**Rede**): b_{eff} (μs) x Loopback TCP (s).

A instância mais afetada pelo tamanho da mensagem a ser transferido foi a m1.small, que possui a menor capacidade de processamento dentre todas as MVs. Mesmo assim obteve um resultado (26,8μs) que não compromete o funcionamento eficaz da rede, pelo contrário, um desempeho muito bom (98,2%) apesar de ser o menor entre os experimentos realizados. As demais MVs são menos afetadas pois possuem capacidade de processamento suficiente para lidar com a demanda gerada pelo *benchmark*.

A MV m1.large obteve um resultado atípico, levando em consideração sua alta capacidade de memória. Contudo, este resultado é justificado pelo seu poder de processamento relativamente baixo, que provoca *scrambling* dos dados no *buffer*, tornando a operação mais lenta. Nada que comprometa o funcionamento eficaz do sistema, porém impactou negativamente no resultado final da simulação especificamente para esta MV. Portanto, podemos dizer que quanto maior o *buffer* de E/S, melhor o desempenho de rede.

Na suíte PTS, temos o *benchmark Loopback TCP Network Performance* que nada mais é que uma simulação de conectividade ponto-a-ponto que vai medir o desempenho do adaptador de rede através do pacote TCP em um teste de *loopback*. O experimento é especialmente configurado para transmitir 10GB via interface de *loopback*. Como já foi justificado inicialmente, estabelecemos que o desempenho ótimo gira em torno de 25s, e o pior em torno de 120s.

Mais uma vez a relação memória-processador define as instâncias com melhor desempenho, pois controla o fluxo de pacotes transmitidos através da rede. As instâncias de menor porte tiveram um desempenho inferior às demais, pois não possuem capacidade de memória su-

ficiente para manipular os 10GB gerados pela interface de *loopback*, levando mais tempo para tranferí-los através da rede. Assim, podemos inferir que o desempenho da rede também está ligado à relação memória-processador, que vai funcionar como um controlador do fluxo dos pacotes provenientes da rede.

4.3 Índices dos *Benchmarks*

Após a execução de cada um dos *benchmarks* analisados, elaboramos a Tabela 6 com os índices de eficiência relativos ao desempenho máximo teórico de cada um dos experimentos realizados. Foi feito tratamento estatístico destes valores, calculando seu percentual de eficiência, a fim de utilizá-los na formulação proposta. A tabela é apresentada a seguir.

BENCHMARKS		**m1.small**	**c1.medium**	**m1.large**	**m1.xlarge**	**c1.xlarge**
CPU	HPL	4,64%	11,27%	14,84%	24,81%	27,51%
	DGEMM	1,15%	13,27%	4,30%	8,54%	11,08%
	FFT	0,94%	3,62%	2,49%	4,52%	4,59%
	PTRANS	6,83%	27,86%	14,71%	39,63%	38,52%
MEM	RAMSpeed SMP/Integer	22,01%	28,38%	25,7%	30,4%	30,77%
	RAMSpeed SMP/Float	24,46%	28,96%	26,30%	27,13%	31,36%
	STREAM	19,53%	28,27%	44,02%	37,53%	41,36%
	RandomAccess	0,41%	9,82%	3,73%	17,3%	17,6%
NET	b_{eff}	98,2%	99,9%	98,8%	99,5%	99,4%
	Loopback TCP	0,58%	62,07%	92,65%	94,34%	96,02%
STO	PostMark	3,75%	4,42%	13,99%	13,00%	14,26%

Tabela 6: Resultados obtidos através dos *Benchmarks* para cada MV (X_{iRj}).

4.4 Índices do DEA

A fim de ponderar os experimentos executados na Seção 4.2 optamos pela execução da metodologia DEA, utilizando o modelo BCC-O, orientado a *output*. Tal metodologia propicia a análise da eficiência comparativa de organizações complexas obtida pela revelação do desempenho de outras unidades, de modo que a referência não é obtida apenas teórica ou conceitualmente, mas por meio da observação das melhores práticas.

O modelo BCC consiste, além do cálculo do índice de eficiência, no cálculo dos pesos das variáveis de *output* (*benchmarks*). Assim, minimiza-se a soma ponderada dos *inputs* dividida pela soma ponderada dos *outputs* da DMU em questão. Após a manipulação dos dados com a metodologia DEA BCC-O, foram atribuídos pesos a cada *benchmark*, por meio de um

solver, considerando cada instância de acordo com sua influência nos resultados obtidos pelos *benchmarks* apresentados na Tabela 6.

A Tabela 7 mostra os pesos obtidos através da metodologia DEA, que parametrizarão o desempenho de cada recurso atrelado a cada *benchmark* em cada MV. Neste estudo de caso foi apresentada a infraestrutura utilizada no trabalho, o cenário proposto, os experimentos realizados, as discussões de avaliação sobre os resultados obtidos pelos *benchmarks*, e a utilização de um método matemático para qualificar os resultados de acordo com sua eficiência. No próximo Capítulo, os resultados obtidos através da aplicação da metodologia de formulação proposta serão apresentados e discutidos.

BENCHMARKS		**m1.small**	**c1.medium**	**m1.large**	**m1.xlarge**	**c1.xlarge**
CPU	HPL	0,77	0,13	0,66	0,28	0,51
	DGEMM	0,88	0,42	0,23	0,29	0,20
	FFT	0,003	0,58	0,25	0,5	0,58
	PTRANS	0,15	0,32	0,07	0,33	0,43
MEM	RAMSpeed SMP/Integer	0,38	0,78	0,17	0,42	0,37
	RAMSpeed SMP/Float	0,46	0,96	0,3	0,68	0,65
	STREAM	0,14	0,18	0,67	0,33	0,61
	RandomAccess	0,91	0,93	0,37	0,73	0,56
NET	b_{eff}	0,42	0,59	0,48	0,55	0,43
	Loopback TCP	0,24	0,43	0,33	0,28	0,48
STO	PostMark	0,57	0,24	0,19	0,42	0,62

Tabela 7: Pesos atribuídos aos Recursos para cada MV (U_{iRj}).

5 RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados e a discussão relativa às simulações realizadas a partir dos *benchmarks* executados. Atribuiu-se percentuais aos desempenhos dos *benchmarks* como pode ser observado na Tabela 6. Como cada experimento tem um foco específico, os recursos terão taxas de desempenho variadas de acordo com a sua representatividade no sistema.

Os índices de eficiência foram calculados através da metodologia DEA (BCC-O), e apresentados na Tabela 7. A seguir, apresentaremos um exemplo a fim de elucidar a metodologia da formulação proposta, e provar a confiabilidade dos resultados aqui obtidos. Observa-se que quanto mais o recurso é utilizado, maior será o peso atribuído.

A Equação 3.4, referente ao parâmetro IAD_{iR_j}, vai assumir valores referentes ao desempenho dos *benchmarks* executados para cada recurso em cada instância (X_{iR_j}), levando em consideração o peso atribuído (U_{iR_j}) pela metodologia DEA a cada resultado de *benchmark* obtido. Os respectivos valores foram apresentados anteriormente no Capítulo 4, especificamente nas Tabelas 6 e 7.

$IAD_{iCPU_{m1.small}} = (U_{HPL}$ x $X_{HPL}) + (U_{DGEMM}$ x $X_{DGEMM}) + (U_{FFT}$ x $X_{FFT}) + (U_{PTRANS}$ x $X_{PTRANS})$
$IAD_{iCPU_{m1.small}} = (4{,}64$ x $0{,}77) + (1{,}15$ x $0{,}88) + (0{,}94$ x $0{,}003) + (6{,}83$ x $0{,}15)$
$IAD_{iCPU_{m1.small}} = 3{,}57 + 1{,}01 + 0{,}002 + 1{,}02$
$IAD_{iCPU_{m1.small}} = 5{,}6$
...
$IAD_{iMEM_{m1.small}} = (U_{RAM_{int}}$ x $X_{RAM_{int}}) + (U_{RAM_{float}}$ x $X_{RAM_{float}}) + (U_{STREAM}$ x $X_{STREAM}) + (U_{RA}$ x $X_{RA})$
$IAD_{iMEM_{m1.small}} = (22{,}01$ x $0{,}38) + (24{,}46$ x $0{,}46) + (19{,}53$ x $0{,}14) + (0{,}41$ x $0{,}91)$
$IAD_{iMEM_{m1.small}} = 8{,}36 + 11{,}25 + 2{,}73 + 0{,}37$
$IAD_{iMEM_{m1.small}} = 22{,}71$
...
$IAD_{iSTO_{m1.small}} = (U_{PostMark}$ x $X_{PostMark})$
$IAD_{iSTO_{m1.small}} = (3{,}75$ x $0{,}57)$
$IAD_{iSTO_{m1.small}} = 2{,}13$
...
$IAD_{iNET_{m1.small}} = (U_{b_{eff}}$ x $X_{b_{eff}}) + (U_{LTCP}$ x $U_{LTCP})$
$IAD_{iNET_{m1.small}} = (98{,}2$ x $0{,}42) + (0{,}58$ x $0{,}24)$
$IAD_{iNET_{m1.small}} = 41{,}24 + 0{,}13$
$IAD_{iNET_{m1.small}} = 41{,}37$

O cálculo representado é o exemplo da metodologia para calcular os índices de desempenho dos *benchmarks* (i) por recursos para a instância m1.small (IAD_{iR_j}). Calcula-se os resultados apenas para a MV "m1.small" como exemplo, pois o procedimento se repete para as demais instâncias. A seguir os resultados obtidos a partir do desenvolvimento das expressões (IAD_{R_j}) são dispostos na Tabela 8.

MV	**CPU**	**MEM**	**STO**	**NET**
m1.small	5,6	22,71	2,13	41,37
c1.medium	18,03	67,14	1,06	85,63
m1.large	12,41	43,12	2,65	77,99
m1.xlarge	24,74	56,2	5,46	81,13
c1.xlarge	35,46	66,83	8,84	88,82

Tabela 8: Índices de Avaliação de Desempenho dos Benchmarks por Recurso em cada MVs (IAD_{iR_j}).

O cálculo dos índices de avaliação de desempenho de cada *benchmark* executado, para cada recurso em cada MV implementada, nos possibilita analisá-los a partir de dois aspectos diferentes. O primeiro destaca a representatividade dos recursos, e seu funcionamento dentro do sistema.

Observa-se no gráfico da Figura 26 que o desempenho de rede é claramente maior que todos os outros, sendo a memória o único recurso que possui um índice relativamente próximo. Os recursos referenciados são justamente os donos dos maiores níveis de *overhead* destacados na Tabela 3, justificando assim, a respectiva condição de gargalos do sistema.

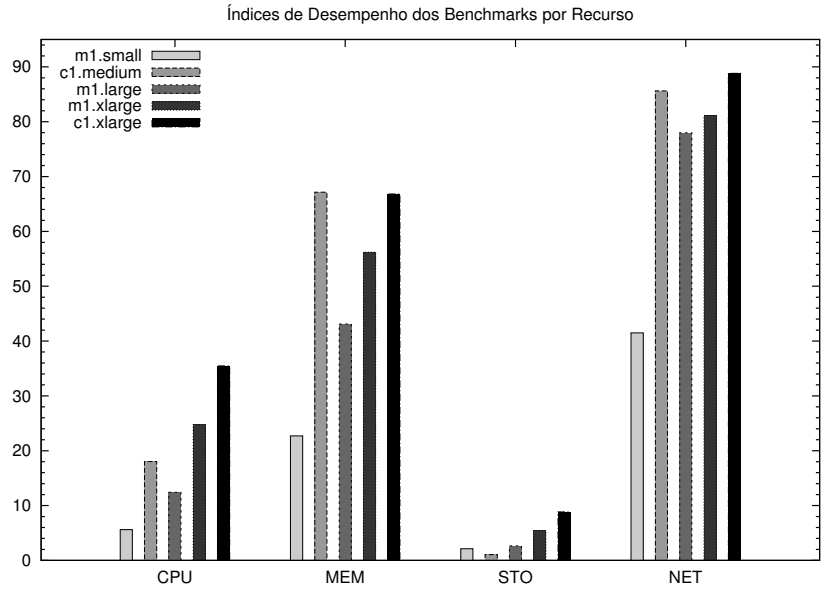

Figura 26: Desempenho dos *Benchmarks* por Recurso.

O segundo aspecto destaca a representatividade de cada instância implementada no sistema. Tendo em vista que o equilíbrio entre os recursos é essencial para um bom desempenho, inferimos através do gráfico da Figura 27 que as instâncias c1 possuem um desempenho bastante similar. Mesmo que a instância média (c1.medium) possua uma capacidade de processamento e memória bem menor que a instância extra grande (c1.xlarge), ambas possuem uma relação processador/memória que as permite alcançar níveis de desempenho bastante satisfatórios, de acordo com sua capacidade individual.

O responsável por estes alcances de desempenho é o simples equilíbrio entre os recursos mais importantes dentro do sistema: memória e processador. O processador é importantíssimo e básico para qualquer sistema. Apesar de possuir uma taxa de *overhead* pequena, trabalha em conjunto com a memória resolvendo as solicitações de tarefas que chegam ao barramento de E/S principal.

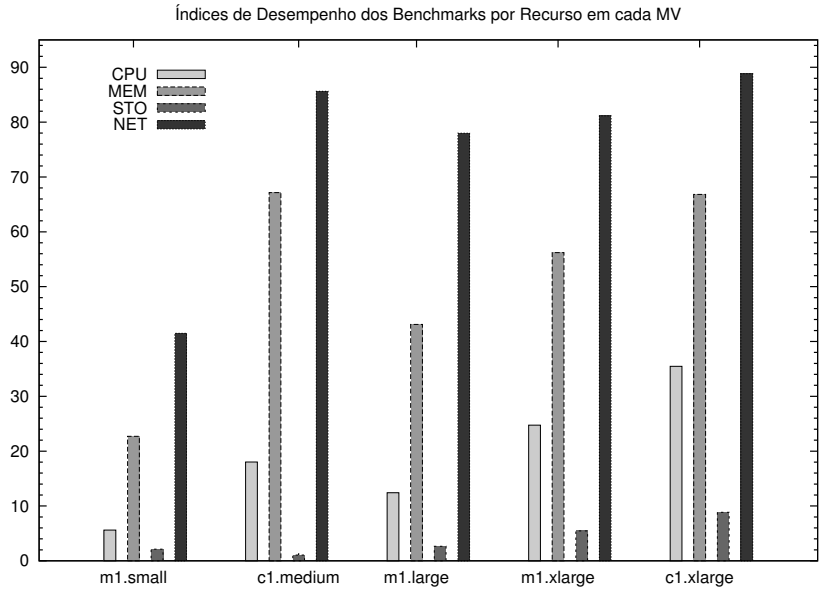

Figura 27: Desempenho dos *Benchmarks* por Instância.

Após a obtenção dos índices de avaliação de desempenho de cada *benchmark* executado para cada recurso em cada MV, teremos como próximo passo da metodologia proposta, o cálculo da média dos resultados encontrados para cada recurso (R) por MV (j) (IAD_{R_j}), apresentado na Equação 3.3.

$$IAD_{CPU} = \sum IAD_{iCPU_j} = IAD_{iCPU_{m1.small}} + IAD_{iCPU_{c1.medium}} + ... + IAD_{iCPU_{c1.xlarge}} \div 5$$
$$IAD_{CPU} = \sum IAD_{iCPU_j} = 5{,}6 + 18{,}03 + 12{,}41 + 24{,}74 + 35{,}46 \div 5$$
$$IAD_{CPU} = \sum IAD_{iCPU_j} = 19{,}24$$
...
$$IAD_{MEM} = \sum IAD_{iMEM_j} = IAD_{iMEM_{m1.small}} + IAD_{iMEM_{c1.medium}} + ... + IAD_{iMEM_{c1.xlarge}} \div 5$$
$$IAD_{MEM} = \sum IAD_{iMEM_j} = 22{,}71 + 67{,}14 + 43{,}12 + 56{,}2 + 66{,}83 \div 5$$

$IAD_{MEM} = \sum IAD_{iMEM_j} = 51{,}2$

...

$IAD_{STO} = \sum IAD_{iSTO_j} = IAD_{iSTO_{m1.small}} + IAD_{iSTO_{c1.medium}} + ... + IAD_{iSTO_{c1.xlarge}} \div 5$

$IAD_{STO} = \sum IAD_{iSTO_j} = 2{,}13 + 1{,}06 + 2{,}65 + 5{,}46 + 8{,}84 \div 5$

$IAD_{STO} = \sum IAD_{iSTO_j} = 4{,}02$

...

$IAD_{NET} = \sum IAD_{iNET_j} = IAD_{iNET_{m1.small}} + IAD_{iNET_{c1.medium}} + ... + IAD_{iNET_{c1.xlarge}} \div 5$

$IAD_{NET} = \sum IAD_{iNET_j} = 41{,}49 + 85{,}63 + 77{,}99 + 81{,}13 + 88{,}82 \div 5$

$IAD_{NET} = \sum IAD_{iNET_j} = 75{,}01$

Encontradas as médias do somatório do produto entre os Pesos (U_{iR_j}) e Valores (X_{iR_j}) de cada recurso em cada instância, partimos para o final da formulação proposta, onde obteremos o Índice de Avaliação de Desempenho Global por Recurso (IAD_{G_R}) conforme apresentado pela Equação 3.1.

$IAD_{G_{CPU}} = IDR_{CPU}$ x $IADCPU$ = (100 - Ov_{CPU}) x 19,24 = (100 - 5)% x 19,24 = **18,28**

$IAD_{G_{MEM}} = IDR_{MEM}$ x $IADMEM$ = (100 - Ov_{MEM}) x 51,2 = (100 - 40)% x 51,2 = **30,72**

$IAD_{G_{STO}} = IDR_{STO}$ x $IADSTO$ = (100 - Ov_{STO}) x 4,02 = (100 - 25)% x 4,02 = **3,01**

$IAD_{G_{NET}} = IDR_{NET}$ x $IADNET$ = (100 - Ov_{NET}) x 75,01 = (100 - 30)% x 75,01 = **52,5**

De acordo com os resultados encontrados utilizando a formulação proposta, foi possível verificar que o desempenho de rede e memória são os mais importantes para um sistema de computação em nuvens. Mesmo que estes recursos possuam *overheads* de funcionamento bastante altos, são os que possuem melhor desempenho, pois são os mais requeridos para este ambiente, tal como mostra a Figura 28.

Cada servidor virtualizado deve possuir uma concentração equilibrada de memória e processamento. Quando um fluxo de *workloads* chega ao sistema, este é pré-processado e enviado ao *buffer* de memória (armazenamento volátil), onde entrará numa fila para que seja devidamente processado em seguida. Caso haja muito mais memória do que capacidade de processamento, o processador receberá muitas requisições do *buffer* de memória , e ficará sobrecarregado. E caso aconteça o inverso, havendo muito mais capacidade de processamento do que memória disponível, teremos dois problemas passíveis de se tornarem realidade.

O primeiro problema é o *buffer overflow*, onde há o estouro da capacidade da memória devido ao excesso de solicitações de acesso enviadas pelo processador. O outro problema é a subutilização do processador, caso o barramento de memória consiga dar conta do fluxo de solicitações. Excetuando-se a rede, os recursos de memória e processamento praticamente alavancam o sistema, provendo disponibilidade e capacidade tão essenciais para sistemas deste porte. Os recursos de rede e memória ratificam sua condição de gargalos do sistema; seja pelos

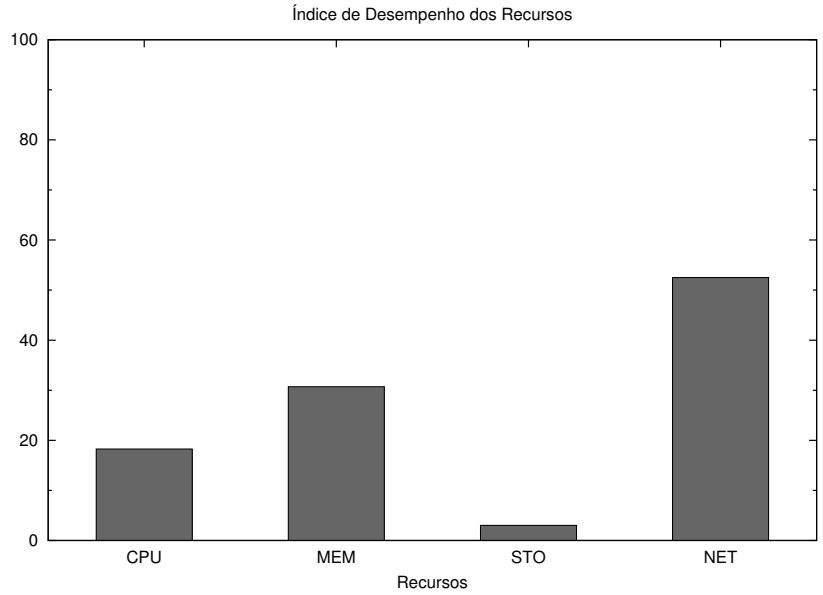

Figura 28: Desempenho dos Recursos (IAD_{G_R}).

overheads atrelados a eles como foi apresentado em Huber et al. (2010), seja pelo seu desempenho como está sendo apresentado neste trabalho.

Normalmente as aplicações de computação em nuvens ficam hospedadas em uma infraestrutura, porém pouco sobrecarregam os recursos de *storage* e processamento. Na maioria dos casos, o *hypervisor* provê uma tecnologia de *snapshots*, para que o sistema não necessite fisicamente dos recursos. O *snapshot* consiste em gravações do estado atual de uma MV, permitindo uma melhor flexibilidade e segurança quanto a falhas em alterações, erros, e/ou falhas. Neste caso, quem trabalha desproporcionalmente são os recursos de memória e rede; que necessitam manter o *snapshot* disponível dentro da memória de acesso rápido para livre utilização dos usuários, e flexível para para ser transmitido entre os nós da rede sem atrasos significativos e/ou perda de dados durante o processo.

A rede é essencial, tendo importância crítica para um sistema de computação em nuvens, que deve estar sempre conectado por conta do alcance da entrega que o provedor do sistema deve oferecer. Além da questão da conectividade, podemos dizer que a velocidade da rede vai ditar o ritmo do fluxo de dados que chegam à plataforma de computação em nuvens, alocando recursos mais rapidamente. Ambas características juntas proveem a disponibilidade e a flexibilidade prometidas pelos sistemas de computação em nuvens.

O *storage* (capacidade de armazenamento não-volátil) é um recurso passivo, que vai receber o fluxo de dados processado pelo sistema. Embora tenha sido avaliado nesta pesquisa como menos requerido, sua utilização pode crescer de acordo com a funcionalidade específica da estrutura. Caso haja uma estrutura de computação em nuvens dedicada a uma aplicação que necessite de uma grande demanda específica de armazenamento *online*, inferimos que a

utilização deste recurso terá um crescimento bastante considerável. Porém, não terá a mesma relevância dos outros recursos que são a base de qualquer sistema computacional.

Deduz-se, assim, que os resultados foram consistentes em relação ao trabalho de Huber et al. (2010), ratificando a condição dos recursos de rede e memória como gargalos do sistema.

6 CONCLUSÕES E TRABALHOS FUTUROS

Com a realização deste trabalho, pudemos concluir que as infraestruturas de computação em nuvens são ferramentas imprescindíveis dentro do novo paradigma tecnológico atual. Com acesso à mobilidade, presenciamos o *boom* da Internet distribuída, com conteúdo disponibilizado em tempo-real, e uma demanda progressiva por este conteúdo. Tendo em vista o crescimento da base de usuários, e consequentemente da demanda pelos recursos da estrutura, fez-se necessário o questionamento deste trabalho quanto à avaliação do desempenho dos recursos dos sistemas de computação em nuvens.

Foi possível verificar que os *benchmarks* atenderam bem às necessidades de simulação em rede, sobrecarregando os recursos interconectados de maneira eficiente, retornando resultados reais. A metodologia DEA nos ajudou a analisar a eficiência de cada experimento utilizado, provendo índices de eficiência (pesos) para os *benchmarks* em cada instância implementada, e para cada recurso avaliado. E por fim, a formulação proposta evidenciou o poder de decisão do *overhead* de funcionamento de cada recurso na avaliação do desempenho global, não obstante, as ponderações relativas à eficiência dos experimentos nortearam a avaliação das simulações.

Concluímos, finalmente, que o desempenho dos recursos de memória e rede são os mais importantes para uma estrutura de computação em nuvens, e por este motivo são considerados gargalos do sistema. Verificamos que o desempenho da grande maioria dos recursos aqui avaliados é diretamente proporcional à taxa de *overhead* de funcionamento, atribuída conforme Huber et al. (2011). Mostra-se portanto, que tanto a memória quanto a rede, alavancam o sistema fornecendo a disponibilidade e capacidade, tão importantes para os sistemas de computação em nuvens.

6.1 Contribuição

Como a tecnologia de computação em nuvens é um “produto novo” no mercado de TI, há ainda um certo receio em migrar para este tipo de tecnologia “desconhecida”. O que mais aflige gestores e usuários é a falta de capacidade de gestão física dos recursos da estrutura. Há a necessidade de saber quanto de cada recurso será exigido pelas aplicações em execução e de que maneira estes recursos influenciarão no desempenho do sistema. As contribuições apresentadas para este trabalho, procuram facilitar a avaliação do desempenho dos recursos principais de *hardware* e rede em uma estrutura de computação em nuvens.

Neste trabalho executamos *benchmarks* em um ambiente de nuvens, quando normalmente são executados em grids. Os experimentos sofreram adaptações apresentadas na Seção 4.2, que possibilitaram sua utilização para o ambiente tratado neste trabalho. Outro ponto importante a ser frisado, é a utilização da metodologia DEA na obtenção dos parâmetros de ponde-

ração dos *benchmarks*, para cada recurso em cada MV. Outra contribuição é observada quanto à ratificação dos recursos de memória e rede como gargalos do sistema, o que torna este trabalho bem sucedido.

6.2 Trabalhos futuros

Propomos a metodologia presente neste trabalho, pois a gestão do desempenho dos recursos não é transparente, dificultando a compreensão dos níveis de utilização dos recursos físicos da infraestrutura. A todo momento surgem questões relevantes sobre computação em nuvens, colocando os gestores dos sistemas em cheque. Algumas destas questões fornecem a possibilidade de elucidar a questão do desempenho dos recursos por meio de outras abordagens. Quanto de cada recurso a será consumido por uma aplicação comum? Será necessária a aquisição de recursos extras para comportar todo o escopo de requisições de *hardware* da estrutura? Como o sistema vai se comportar durante um processo de migração de MVs? São perguntas que podem ser respondidas apoiadas neste trabalho, e podem ser consideradas como prováveis propostas de trabalhos futuros.

Após a realização deste trabalho, foi natural a percepção de que se pode realizar muito mais com respeito ao desempenho de sistemas de computação em nuvens, em vários níveis. Em uma próxima oportunidade poderíamos desenvolver uma aplicação para ser hospedada em um ambiente de nuvem, avaliando a taxa de consumo, ou o comportamento da aplicação para cada recurso. Outra abordagem, seria analisar seu comportamento durante o processo de migração de MVs dentro da estrutura.

Também seria possível configurar o *benchmark* HPCC de uma maneira mais agressiva, gerando mais blocos de dados (maior granularidade de dados), e também configurando um tamanho de problema maior. Poderíamos implementar esta ideia em um ambiente com mais nós, para analisarmos a perda de desempenho causada pela sobrecarga dos recursos. Outra proposta de trabalho é a questão crítica da migração de MVs. A questão trataria da maneira como a migração das MVs afetaria o desempenho, tanto de uma aplicação hospedada na nuvem, como da infraestrutura como um todo.

A utilização de *benchmarks* é essencial para avaliação do processo, provendo uma infinidade de possibilidades a serem trabalhadas com as mais diversas maneiras de abordagem. Devemos então, atentar para a constante evolução dos sistemas de computação em nuvens, destacando a evolução do comportamento dos recursos da estrutura, para tornar possível o aproveitamento da abordagem proposta neste trabalho.

BIBLIOGRAFIA

AMAZON. **Amazon Elastic Compute Cloud EC2**. 2008. Website. Último acesso em 25 de junho de 2011. Disponível em: <http://aws.amazon.com/ec2>.

ARMBRUST, M. et al. **Above the clouds: A berkeley view of cloud computing**. [S.l.], 2009.

BANERJEE, P. et al. **Everything as a Service: Powering the New Information Economy**. *Computer*, IEEE, v. 44, n. 3, p. 36–43, 2011.

BANKER, R.D.; CHARNES, A.; COOPER, W.W. **Some Models for Estimating Technical and Scale Ineffciencies in Data Envelopment Analysis**. *Management Science*, JSTOR, p. 1078–1092, 1984.

CARDOSO, N. **Virtual Clusters Sustained by Cloud Computing Infrastructures**. *For Jury Evaluation*, 2011.

CHARNES, A.; COOPER, W.W.; RHODES, E. **Measuring the Efficiency of Decision Making Units**. *European Journal of Operational Research*, Elsevier, v. 2, n. 6, p. 429–444, 1978.

DEBREU, G. **The Coefficient of Resource Utilization**. *Econometrica: Journal of Econometric Society*, JSTOR, p. 273–292, 1951.

DELL. **Dell Compellent Storage Center: Self-Optimized, Powerful**. 2011. Website. Último acesso em 06 de fevereiro de 2012. Disponível em: <http://i.dell.com/sites/content/shared-content/data-sheets/en/Documents/Family-Brochure_Final.pdf>.

DELL. **Power Edge M1000e**. 2011. Website. Último acesso em 06 de fevereiro de 2012. Disponível em: <http://i.dell.com/sites/content/business/solutions/engineering-docs/en/Documents/server-poweredge-m1000e-tech-guidebook.pdf>.

DELL. **Power Edge M710HD**. 2011. Website. Último acesso em 06 de fevereiro de 2012. Disponível em: <http://i.dell.com/sites/content/business/solutions/engineering-docs/en/Documents/M710HD-TechGuide-10012010-final.pdf>.

DONGARRA, J. et al. **Subroutine DGEMM**. 1989. Website. Último acesso em 19 de janeiro de 2012. Disponível em: <http://www.netlib.org/blas/dgemm.f>.

DONGARRA, J. et al. **The Message Passing Interface (MPI) Standard**. 1992. Website. Último acesso em 19 de janeiro de 2012. Disponível em: <https://mcs.anl.gov/research/projects/mpi>.

DONGARRA, J.J.; LUSZCZEK, P. **Overview of the HPC Challenge Benchmark Suite**. In: CITESEER. *Proceeding of SPEC Benchmark Workshop*. [S.l.], 2006.

DONGARRA, J.; LUSZCZEK, P. **HPCC High Performance Computing Challenge**. 2010. Website. Último acesso em 11 de janeiro de 2012. Disponível em: <http://icl.eecs.utk.edu/hpcc>.

FOCHEZATTO, A. **Análise da Eficiência Relativa dos Tribunais da Justiça Estadual Brasileira Utilizando o Método DEA**. *Reunión de Estudios Regionales*, Badajoz, 2010.

FOSTER, I. et al. **Cloud computing and grid computing 360-degree compared**. In: IEEE. *Grid Computing Environments Workshop, 2008. GCE'08*. [S.l.], 2008. p. 1–10.

FRIGO, M.; JOHNSON, S.G. **benchFFT**. 2007. Website. Último acesso em 9 de fevereiro de 2012. Disponível em: <http://www.fftw.org/benchfft/>.

GONCALVES, D.B.; JUNIOR, J.C.V. **White Paper - Virtualização**. 2010. Website. Último acesso em 14 de Abril de 2012. Disponível em: <http://www.sensedia.com/br/anexos/wp_*virtualizacao*.pdf>.

HEY, T.; DONGARRA, J.; R., Hockney. **Parkbench Matrix Kernel Benchmarks**. 1996. Website. Último acesso em 9 de fevereiro de 2012. Disponível em: <http://www.netlib.org/parkbench/html/matrix-kernels.html>.

HOLLANDER, R.M.; BOLOTOFF, P.V. **RAMspeed**. 2002. Website. Último acesso em 09 de fevereiro de 2012. Disponível em: <http://alasir.com/software/ramspeed/>.

HUBER, N. et al. **Analysis of the Performance-Influencing Factors of Virtualization Platforms**. *On the Move to Meaningful Internet Systems, OTM 2010*, Springer, p. 811–828, 2010.

HUBER, N. et al. **Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments**. In: *1st International Conference on Cloud Computing and Services Science*. [S.l.: s.n.], 2011. p. 7–9.

INFORMA, Adm. **Cloud Computing. Exemplos e Aplicações Práticas**. 2011. Website. Último acesso em 24 de março de 2012. Disponível em: <http://adm-informa.blogspot.com.br/2012/04/5-facam-uma-descricao-sucinta-de-cloud.html>.

INSERT. **Information Security Research Team**. 2007. Website. Último acesso em 24 de Julho de 2012. Disponível em: <http://insert.uece.br>.

INTEL. **Intel Xeon Processor 5600**. 2010. Website. Último acesso em 06 de fevereiro de 2012. Disponível em: <http://www.intel.com/content/www/us/en/processors/xeon/xeon-5600-vol-1-datasheet.html>.

IOSUP, A. et al. **Performance analysis of cloud computing services for many-tasks scientific computing**. *Parallel and Distributed Systems, IEEE Transactions on*, IEEE, v. 22, n. 6, p. 931–945, 2011.

JAIN, R. **The Art of Computer Systems Performance Analysis**. [S.l.]: John Wiley & Sons, 2008. 206 p.

KATCHER, J. **PostMark: A New File System Benchmark**. 1997. Website. Último acesso em 09 de fevereiro de 2012. Disponível em: <http://www.netapp.com/technology/level3/3022.html>.

KOESTER, D.; LUCAS, B. **Random Acess**. 2008. Website. Último acesso em 19 de janeiro de 2012. Disponível em: <http://icl.cs.utk.edu/projectsfiles/hpcc/RandomAccess/>.

LAM, C. et al. **Fiber optic communication technologies: What's needed for datacenter network operations**. *Communications Magazine, IEEE*, IEEE, v. 48, n. 7, p. 32–39, 2010.

LARABEL, M.; TIPPETT, M. **Loopback TCP Network Performance**. 2008. Website. Último acesso em 09 de fevereiro de 2012. Disponível em: <http://openbenchmarking.org/test/pts/network-loopback>.

LARABEL, M.; TIPPETT, M. **Phoronix Test Suite**. 2008. Website. Último acesso em 11 de janeiro de 2012. Disponível em: <http://www.phoronix-test-suite.com>.

LAWNSON, C.L. et al. **Basic Linear Algebra Subprograms for FORTRAN Usage**. *ACM Transactions on Mathematical Software (TOMS)*, ACM, v. 5, n. 3, p. 308–323, 1979.

LEHTINEN, S.; LONVICK, C. **The Secure Shell (SSH) Protocol Architecture**. IETF, jan 2006. RFC 4251. (Request for Comments, 4251). Disponível em: <http://www.ietf.org/rfc/rfc4251.txt>.

LINS, M.P.E.; MEZA, L.A.; ANTUNES, C.H. **Análise Envoltória de Dados e Perspectivas de Interação no Ambiente de Apoio à Decisão.** [S.l.]: COPPE/UFRJ, 2000.

MCCALPIN, J.D. **STREAM: Sustainable Memory Bandwidth in High Performance Computers**. 2002. Website. Último acesso em 19 de janeiro de 2012. Disponível em: <http://www.cs.virginia.edu/stream/>.

MEIRELES, A.M.R. **Uma proposta de SAD para avaliação de eficiência do poder judiciário do estado do Ceará utilizando AED**. Dissertação (Mestrado) — Universidade Estadual do Ceará - UECE, Fortaleza, 2012.

MOORE, G.E. et al. **Cramming more components onto integrated circuits**. *Proceedings of the IEEE*, Institute of Electrical and Electronics Engineering, 1963, v. 86, n. 1, p. 82–85, 1998.

NARASIMHAN, B.; NICHOLS, R. **State of Cloud Applications and Platforms: The Cloud Adopters' View**. *IEEE Computer*, v. 44, n. 3, p. 24–28, 2011.

OPTICOMM. **Fiber-Optics.Info**. 2008. Website. Último acesso em 23 de janeiro de 2012. Disponível em: <http://www.fiber-optics.info/fiber_optic_glossary/long-haul_telecommunications>.

OSTERMANN, S. et al. **A performance analysis of EC2 cloud computing services for scientific computing**. *Cloud Computing*, Springer, p. 115–131, 2010.

PARKHILL, D.F. **The challenge of the computer utility**. [S.l.]: Addison-Wesley Reading, MA, 1966.

PETITET, A. et al. **HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers**. 2008. Website. Último acesso em 19 de janeiro de 2012. Disponível em: <http://netlib.org/benchmark/hpl/>.

RABENSEIFNER, R.; SCHULZ, G. **Effective Bandwidth Benchmark**. 1999. Website. Último acesso em 9 de fevereiro de 2012. Disponível em: <https://fs.hlrs.de/projects/par/mpi//b_eff/>.

READ, J. **Cloud Harmony: Benchmarking the Cloud**. 2009. Website. Último acesso em 16 de novembro de 2011. Disponível em: <http://www.cloudharmony.com>.

SOSINSKY, B. **Cloud computing bible**. [S.l.]: Wiley Publishing, 2011.

SUGREE, Y. **Theoretically Measuring Performance of a Computer**. 2007. Website. Último acesso em 24 de março de 2012. Disponível em: <http://www.howforge.com/theoretically-measuring-performance-of-a-computer>.

VMWARE. **Secure Your Virtual Infrastructure: Hosted vs. Bare-Metal Virtualization Overview**. 2012. Website. Último acesso em 09 de fevereiro de 2012. Disponível em: <http://www.vmware.com/br/technical-resources/virtualization-topics/security/platform-security/overview.html>.

WARDLEY, S. **Bits or pieces? A node between the physical and digital**. 2009. Website. Último acesso em 10 de agosto de 2011. Disponível em: <http://blog.gardeviance.org/2009/01/terms-that-i-use.html>.

WEISS, A. **Computing in the clouds**. *COMPUTING*, v. 16, 2007.

APÊNDICE