UNIVERSIDADE FEDERAL DO PAMPA

RODRIGO FERRER UBER

BENCHMARK DE INFRAESTRUTURAS PARA SUPORTE A ALTA DISPONIBILIDADE DE BANCO DE DADOS

BAGÉ
2013

RODRIGO FERRER UBER

BENCHMARK DE INFRAESTRUTURAS PARA SUPORTE A ALTA DISPONIBILIDADE DE BANCO DE DADOS

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Sistemas Distribuídos com Ênfase em Banco de Dados da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Especialista.

Orientador: Érico Marcelo Hoff do Amaral

Coorientador: Sandro da Silva Camargo

BAGÉ
2013

RODRIGO FERRER UBER

BENCHMARK DE INFRAESTRUTURAS PARA SUPORTE A ALTA DISPONIBILIDADE DE BANCO DE DADOS

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Sistemas Distribuídos com Ênfase em Banco de Dados da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Especialista.

Trabalho de Conclusão de Curso defendido e aprovado em: 30 de julho de 2013.

Banca examinadora:

_____

Prof. Msc. Érico Marcelo Hoff do Amaral
Orientador
UNIPAMPA

_____

Prof. Dr. Sandro da Silva Camargo
UNIPAMPA

_____

Prof. Dr. Leonardo Bidese de Pinho
UNIPAMPA

Agradeço a todos que contribuíram no decorrer desta jornada, em especial:

Primeiramente a Deus, a quem devo o que sou.

A minha família que sempre me apoiou nos estudos e nas escolhas tomadas.

A minha companheira por sempre me incentivar e compreender nos momentos difíceis.

Ao orientador Prof. Msc. Érico Marcelo Hoff do Amaral e ao Prof. Dr. Sandro Camargo que tiveram papel fundamental na elaboração deste trabalho.

Aos meus colegas pelo companheirismo e disponibilidade para me auxiliar em vários momentos.

“Determinação, coragem e autoconfiança são fatores decisivos para o sucesso. Se estamos possuídos por uma inabalável determinação conseguiremos superá-los. Independentemente das circunstâncias, devemos ser sempre humildes, recatados e despidos de orgulho."

Dalai Lama

RESUMO

Atualmente, estão se disseminando diversos tipos de aplicações computacionais que necessitam de uso intensivo de banco de dados. Muitas vezes, estas aplicações demandam uma infraestrutura baseada em cluster a fim de oferecer alta disponibilidade, escalabilidade e tolerância a falhas. Dentro deste escopo, esse trabalho apresenta um estudo sobre o comportamento de aplicações de bancos de dados, baseadas em software livre, sobre diferentes estruturas computacionais. Dentre as infraestruturas implementadas, foram utilizadas estruturas de alto desempenho como, por exemplo, clusters. A variabilidade estrutural ocorreu por conta dos nós utilizados, como máquinas físicas, máquinas virtuais e um cluster construído sobre uma nuvem computacional; uma estrutura tradicional também serviu como base para os experimentos. A aplicação executada sobre as estruturas consiste de um banco de dados PostgreSQL e o desempenho foi medido através da ferramenta de benchmark PgBench. Como resultados foram elaborados gráficos que tornam possível visualizar a variabilidade de desempenho obtida sobre as diferentes arquiteturas.

Palavras-chave: Banco de dados, cluster, benchmark.

ABSTRACT

Currently, various kinds of computing applications, which are database intensive, are being disseminated. Often, these applications demand a cluster-based infrastructure in order to provide high availability, scalability and fault tolerance. In this scope, this paper presents a study on the behavior of database intensive applications, using open source software, based on different computational structures. Among the infrastructure implemented, were used as high-performance structures, such as clusters. The structural variability was due to the nodes used as physical machines, virtual machines and a cluster built on a cloud computing, a traditional structure also served as a basis for the experiments. The application running on the structures consists of a PostgreSQL database and the performance was measured by the benchmark tool PgBench. The results were drawn graphics that make it possible to visualize the performance variability obtained on different architectures.

Keywords: Database, cluster, benchmark.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API - *Application Programming Interface* (Interface de Programação de Aplicativos)

COWS - C*lusters of workstations* – (Clusters de estações de trabalho)

CPU - C*entral processing unit* (Unidade central de processamento)

CRM - *Customer Relationship Management* (Gestão de Relacionamento com o Cliente)

FTP – *File Transfer Protocol* (Protocolo de Transferência de Arquivos)

GUI- *Graphical User Interface* (Interface gráfica do utilizador)

HA – *High availability* (Alta disponibilidade)

HPC - *High performance computing* (Computação de alto desempenho)

IaaS - *Infraestructure as a Service* **(**Infraestrutura como um Serviço)

LAN – *Local Area Network* (Rede de área local)

MIMD - M*ultiple instruction, multiple data* (Múltiplas instruções, múltiplos dados):

MISD – *Multiple instruction, single data* (Múltiplas instruções, único dado)

MPI - *Message Passing Interface* (Interface para passagem de mensagens)

NIST - *National Institute of Standards and Technology* (Instituto Nacional de Padrões e Tecnologia)

NUMA - *Nonuniform memory acess* (Acesso não uniforme à memória)

OLTP - *Online Transaction Processing* (Processamento de Transações em Tempo Real)

PaaS - *Plataform as a Service* (Plataforma como um Serviço)

PC – *Personal Computer* (Computador Pessoal)

PDA - *Personal digital assistants* (Assistente pessoal digital)

QoS - *Quality of Service* (Qualidade de Serviço)

RAM - *Random Access Memory* (Memória de acesso aleatório)

SMP - *Symmetric multiprocessor* (multiprocessadores simétricos)

SISD - S*ingle instruction, single data* (Instrução única, único dado)

SIMD - *Single instruction, multiple data* (Instrução única, múltiplos dados)

SLA - *Service Level Agreement* (Acordo de Nível de Serviço)

SaaS - *Software as a Service* (Software como um serviço)

SGBD - Sistema de Gerenciamento de Banco de Dados

SQL - *Structured Query Language* (Linguagem de Consulta Estruturada)

TI - Tecnologia da Informação

TPC - *Transaction Processing Performance Council*

TPS – Transações por segundo

VM – *Virtual Machine* (Máquina Virtual)

SUMÁRIO

1 INTRODUÇÃO

O rápido ritmo da mudança que sempre caracterizou a tecnologia do computador continua sem descanso. A era do computador moderno, definida pelo uso de computadores digitais, que não utilizam componentes analógicos com base de seu funcionamento, teve início em 1945 até meados dos anos 80. Os computadores possuíam custos elevados e tamanhos consideráveis. Até mesmo microcomputadores custavam dezenas de milhares de dólares cada. Como resultado, a maioria das organizações possuía um número reduzido de computadores, e por falta de uma maneira de conectá-los, operavam independentemente uns dos outros.

Começando por volta dos anos 80, entretanto, dois avanços em tecnologia chegaram para mudar essa situação. O primeiro foi o desenvolvimento de poderosos microprocessadores. Vários desses possuíam o poder computacional de um computador *mainframe,* mas por uma fração do preço.

O segundo avanço foi a invenção das redes de computadores de alta velocidade. Redes de áreas locais (LAN, do inglês *Local Area Network*) permitiam que centenas de máquinas dentro de um prédio pudessem ser conectadas de modo que pequenas quantidades de informação pudessem ser transferidas entre as máquinas em poucos microssegundos ou menos (Andrew S Tanenbaum & Steen, 2006).

Normalmente em uma LAN estão conectadas estações de trabalho com variados níveis de desempenho, bem como cargas de processamento muito diferentes e tempos de comunicação variáveis (Diekmann & Monien, 1997).

Esses avanços possibilitaram a montagem de sistemas computacionais compostos por um grande número de computadores interligados, que são chamados redes de computadores ou sistemas distribuídos, de forma antagônica aos sistemas centralizados, que consistem de um simples computador e seus periféricos.

Hodiernamente, a disseminação dos mais diversos tipos de aplicações computacionais proporcionou um novo impulso para satisfazer as prerrogativas de desempenho e confiabilidade, imprescindíveis no contexto tecnológico no qual estamos inseridos. O uso intensivo de banco de dados é um exemplo destas aplicações.

Invariavelmente, a busca por uma estrutura computacional que atenda de forma condizente as demandas recursais necessárias às aplicações passa pelas seguintes premissas:

adquirir uma solução proprietária, utilizando software específico e máquinas com alto poder de processamento, geralmente acompanhados por um alto custo de aquisição, ou a busca por soluções alternativas viáveis que façam frente às soluções proprietárias. As duas possibilidades possuem vantagens e desvantagens, tais como: suporte, custos de aquisição, implementação, entre outros.

Um recurso importante e relativamente recente no projeto de computadores é o *clustering* que, como uma abordagem para oferecer alto desempenho e disponibilidade de forma alternativa, são bastante atraentes para aplicações de servidores. O aumento na velocidade de transmissão de dados via rede possibilitou a obtenção de resultados muito similares em termos de desempenho aos obtidos com os supercomputadores ou multicomputadores, utilizando computadores pessoais interligados por rede, isto é possível distribuindo ou paralelizando o processamento entre os computadores pessoais.

Em instituições acadêmicas e industriais, clusters de estações de trabalho ou PCs, de baixo custo, estão substituindo os pequenos sistemas com multiprocessadores simétricos (SMP) e as máquinas paralelas, de alto custo, estes podem ser encontrados em vários dos modernos e poderosos sistemas computacionais. Em Centros de Pesquisa como o CERN são desenvolvidos experimentos baseados em clusters de milhares de estações de trabalhos em redes locais conectados a uma WAN (Bevilacqua, 1999).

Diversas Universidades no exterior como Berckley e Princeton, e no Brasil, como USP e Unicamp, mantêm clusters dedicados à realização de computação massiva em suas diferentes linhas de pesquisa, o que inclui modelagem de semicondutores, dinâmica biomolecular, entre outras. Apesar do clustering não ser capaz de resolver todos os problemas, pode ajudar a organização que está tentando maximizar alguns de seus recursos existentes. Mas nem todo programa pode se beneficiar do clustering. Organizações que servem aplicações, como servidores web, banco de dados e servidores FTP, são potenciais candidatos à beneficiação proveniente da tecnologia. A tecnologia de clusters é projetada tendo em mente a escalabilidade.

Baseado nisso, esse trabalho visa a comparação de infraestruturas simplificadas das arquiteturas tradicionais e de alto desempenho, buscando uma relação de desempenho e custo-benefício das mesmas, oferecendo assim parâmetros incisivos na adoção de um escopo adequado para armazenamento e transações eficientes em aplicações de banco de dados.

A divisão do trabalho foi elaborada em outros cinco capítulos conforme detalhamento:

Capítulo 2 (referencial teórico) oferece uma visão da estrutura interna de algumas das estruturas envolvidas no estudo de caso e proporciona a análise de algumas abordagens promissoras para organização paralela, como multiprocessadores simétricos, porém, dando ênfase em cluster e *cloud computing*. Expõe a base técnica do trabalho em relação à aplicação utilizada no experimento, bem como as características atinentes as arquiteturas empregadas e apresenta trabalhos relacionados ao tema.

Capítulo 3 (metodologia de desenvolvimento) explica, de forma abstrata, toda ação desenvolvida no caminho de desenvolvimento do trabalho.

Capítulo 4 (implementação e testes) descreve detalhadamente as arquiteturas desenvolvidas e suas características.

Capítulo 5 (resultados e discussões) apresenta os resultados obtidos nas avaliações e são elencados alguns comentários sobre os resultados.

Capítulo 6 (conclusão) apresenta as conclusões sobre o trabalho desenvolvido e aponta os trabalhos futuros a serem realizados.

1.1 Problema de Pesquisa

Com o crescimento de aplicações computacionais que necessitam de uso intensivo de banco de dados e demandam a utilização de computação de alto desempenho que ofereça alta disponibilidade, escalabilidade e tolerância a falhas, esse trabalho aborda o comportamento de uma aplicação de banco de dados, baseada em software livre, sobre diferentes estruturas computacionais, com o intuito de verificar, através de experimentos, qual das estruturas empregadas apresentam melhor desempenho nas transações de dados efetuadas de um subsistema para outro em uma unidade de tempo específica.

1.2 Objetivos

1.2.1 Objetivos Gerais

Avaliar o desempenho da aplicação de banco de dados PostgreSQL, baseada em software livre, sobre diferentes estruturas computacionais.

1.2.2 Objetivos Específicos

- Realizar uma análise de desempenho das diferentes arquiteturas computacionais provedoras de recursos para a aplicação de banco de dados PostgreSQL.
- Fornecer apoio a decisão, através de informações concretas, para escolha de um escopo adequado às necessidades computacionais da organização.
- Observar o comportamento de determinadas estruturas computacionais de alto desempenho e confrontá-las no quesito desempenho com a arquitetura centralizada cliente-servidor.

1.3 Hipóteses

A escalabilidade, característica das estruturas de alto desempenho abordadas no presente trabalho, é desejável em todo sistema, rede ou processo, e consiste na habilidade de manipular uma porção crescente de trabalho de forma uniforme ou estar preparado para crescer, sem comprometer a porção em atividade. Dessa forma, o presente estudo aborda, especificamente, quatro diferentes estruturas computacionais implementadas com variabilidade quanto aos tipos de nós de cluster, as quais fornecerão os recursos necessários para as aplicações de banco de dados processarem as transações que serão objeto dos experimentos. A idéia principal é verificar se um aumento no número de bases de dados - escalabilidade - traria por consequência um ganho de processamento computacional, tendo em vista que as bases de dados têm por atribuição, no presente estudo, proporcionar o balanceamento das cargas de processamento e replicação das bases de dados.

O objetivo é verificar quais as arquiteturas e quais os tipos de nós apresentam o melhor desempenho utilizando homogeneidade quanto aos equipamentos, infraestrutura de conexão e ao software empregado. O fator escalabilidade consiste em verificar se, aumentando as bases de dados existentes nos nós físicos, traria por consequência um aumento no desempenho que será auferido através da ferramenta de benchmark específica utilizada nos testes.

2 REFERENCIAL TEÓRICO

Esta seção visa a fornecer uma visão da estrutura interna de algumas das estruturas envolvidas no estudo de caso e proporcionar a análise de algumas abordagens promissoras para organização paralela, como multiprocessadores simétricos, com ênfase em cluster e *cloud computing*, bem como expor a base técnica do trabalho em relação à aplicação utilizada no experimento e as características atinentes as arquiteturas empregadas.

2.1 Estruturas de Sistemas Operacionais

Um sistema computacional moderno consiste em um ou mais processadores, memória principal, discos, teclado, mouse, monitor, interfaces de rede e outros dispositivos de entrada e saída. Gerenciar todos esses componentes e usá-los de maneira otimizada é um trabalho extremamente difícil. Por isso, os computadores têm um dispositivo de software denominado sistema operacional, cujo trabalho é fornecer aos programas do usuário um modelo de computador melhor, mais simples e mais limpo e lidar com o gerenciamento de todos os recursos mencionados (Andrew S. Tanembaum, 2003).

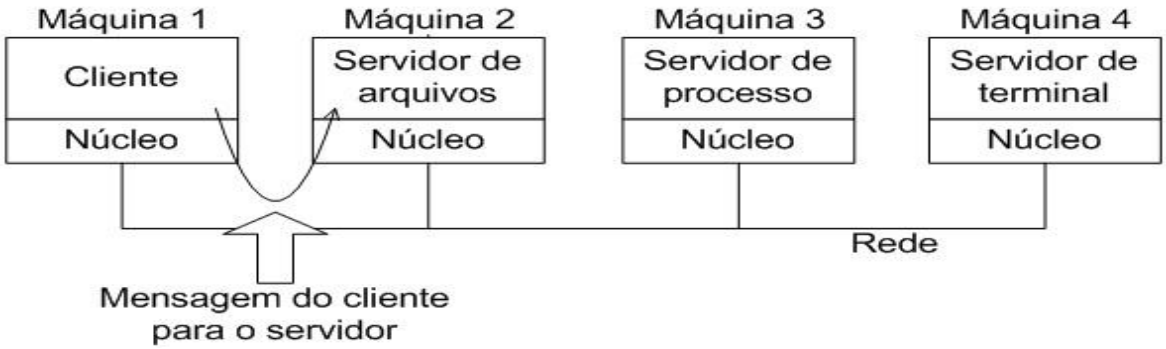
Nesta seção, será abordada a estrutura interna do sistema operacional, em específico, duas diferentes estruturas, com o intuito de fornecer um espectro de possibilidades sobre alguns projetos que têm sido usados na prática. Os dois grupos abordados serão os seguintes: sistemas cliente-servidor e máquinas virtuais.

2.2 O Modelo Cliente-servidor

Esse modelo, conhecido como cliente-servidor, é composto por duas classes de processos, os servidores, que prestam algum serviço, e os clientes, que usam esse serviço. A comunicação entre clientes e servidores é muitas vezes realizada por meio de trocas de mensagens.

Para obter um serviço, um processo cliente constrói uma mensagem dizendo o que deseja e a envia ao serviço apropriado. Este faz o trabalho e envia a resposta de volta. Uma generalização óbvia é executar clientes e servidores em máquinas diferentes, conectados por uma rede local ou de grande área, conforme ilustração da figura 1.

Figura 1 - O modo cliente-servidor em uma rede.



Fonte: (Andrew S. Tanembaum, 2003)

Uma vez que os clientes se comunicam com os servidores enviando mensagens, eles não precisam saber se as mensagens são entregues localmente em suas próprias máquinas ou se são enviadas através de uma rede a servidores em uma máquina remota. No que se refere aos clientes, o mesmo ocorre em ambos os casos: solicitações são enviadas e as respostas, devolvidas. Dessa forma, o modelo cliente-servidor é uma abstração que pode ser usada para uma única máquina ou para uma rede de máquinas.

Cada vez mais vários sistemas envolvem usuários em seus computadores pessoais, como clientes e máquinas grandes em outros lugares, como servidores. De fato, grande parte da Web opera dessa forma. Um PC envia uma solicitação de página da Web ao servidor e a página da Web retorna. Esse é um uso típico do modelo cliente-servidor em uma rede (Andrew S. Tanembaum, 2003).

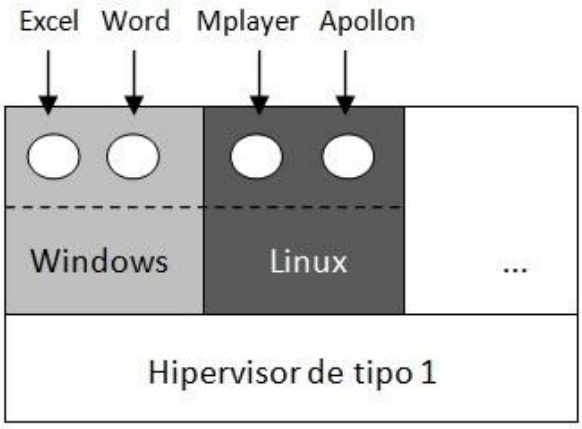
2.3 Máquinas Virtuais

A idéia da virtualização foi em grande medida ignorada na indústria da computação até pouco tempo atrás. Mas, nos últimos anos, uma combinação de novas necessidades, novos softwares e novas tecnologias tornou essa ideia um tópico de interesse.

Primeiro as necessidades. Muitas companhias tradicionalmente executavam seus servidores de correio, servidores da Web, servidores FTP e outros em computadores separados, algumas vezes com sistemas operacionais diferentes. Elas percebem a virtualização como u modo de executar todos eles na mesma máquina sem que uma falha em um servidor afete o resto. A virtualização também é popular na indústria de hospedagem de

páginas da Web. Sem a virtualização, os clientes da hospedagem na Web são forçados a escolher entre hospedagem compartilhada (fornece apenas uma conta de acesso a um servidor da Web, mas não lhes permite controlar o software do servidor) e hospedagem dedicada (que lhes oferece uma máquina própria, que é muito flexível, mas pouco econômica para sites de pequeno a médio porte). A virtualização também é utilizada por usuários finais que querem executar dois ou mais sistemas operacionais ao mesmo tempo, por exemplo Windows e Linux, porque alguns de seus pacotes de aplicações favoritos são executados em um sistema e outros no outro. Essa situação é mostrada na Figura 2, na qual o termo ‘monitor de máquina virtual’ foi alterado para hipervisor tipo 1 nos últimos anos (Andrew S. Tanembaum, 2003).

Figura 2 - Hipervisor de tipo 1.

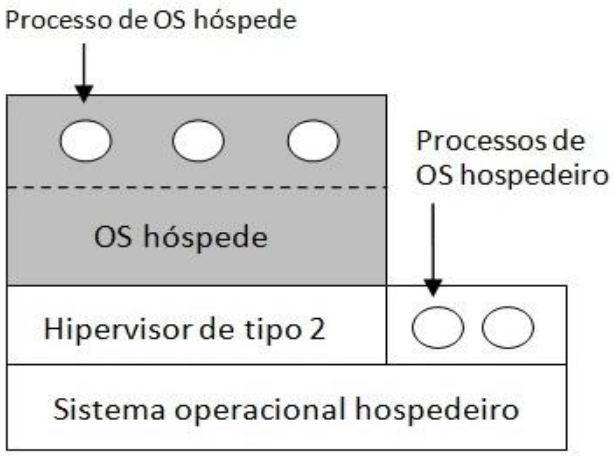


Fonte: (Andrew S. Tanembaum, 2003)

Agora o software. Embora ninguém discuta a atratividade das máquinas virtuais, o problema foi a implementação. Para executar o software de máquina virtual em um computador, sua CPU deve ser virtualizável (Popek & Goldberg, 1974). Em poucas palavras, há um problema nesse caso. Quando um sistema operacional sendo executado em uma máquina virtual executa uma instrução privilegiada, é essencial que o hardware crie um dispositivo para o monitor da máquina virtual, de modo que a instrução possa ser emulada em software. Em algumas CPUs – principalmente Pentium, os predecessores e seus clones – tentativas de executar instruções não privilegiadas no modo usuário são simplesmente ignoradas. Essa propriedade impossibilitou a existência de máquinas virtuais nesse hardware, o que explica a falta de interesse na indústria da computação.

Essa situação mudou como resultado de vários projetos de pesquisa acadêmica na década de 1990, particularmente o Disco em Stanford (Bugnion, Devine, Govil, & Rosenblum, 1997), que conduziu a produtos comerciais (por exemplo, VMware Workstation) e a uma retomada do interesse em máquinas virtuais. O VMWare Workstation é um hipervisor de tipo 2, mostrado na Figura 3. Ao contrário dos hipervisors de tipo 1, que são executados diretamente no hardware, os hipervisors de tipo 2 são executados como programas aplicativos na camada superior do sistema operacional, conhecido como sistema operacional hospedeiro. Depois de ser iniciado, um hipervisor de tipo 2 lê o CD-ROM de instalação para o sistema operacional hóspede escolhido e instala um disco virtual, que é só um arquivo grande no sistema de arquivos do sistema operacional hospedeiro.

Figura 3 - Hipervisor de tipo 2.



Fonte: (Andrew S. Tanembaum, 2003)

Quando o sistema operacional hóspede é inicializado, faz o mesmo que no verdadeiro hardware, normalmente iniciando algum processo subordinado e, em seguida, uma interface gráfica GUI. Alguns hipervisors traduzem os programas binários do sistema operacional convidado bloco a bloco, substituindo determinadas instruções de controle por chamadas ao hipervisor. Os blocos traduzidos são executados e armazenados para uso posterior.

Uma abordagem diferente para o gerenciamento de instruções de controle é modificar o sistema operacional para removê-las. Essa abordagem não é uma virtualização autêntica, e sim uma paravirtualização (Andrew S. Tanembaum, 2003).

2.4 Sistemas Distribuídos

Várias definições de sistemas distribuídos têm sido dadas na literatura, porém, nenhuma delas satisfatória e em concordância umas com as outras. Conforme Tanenbaum & Steen (2006), é suficiente uma pequena caracterização:

Um sistema distribuído é uma coleção de computadores independentes que aparecem para seus usuários como um sistema coerente único.

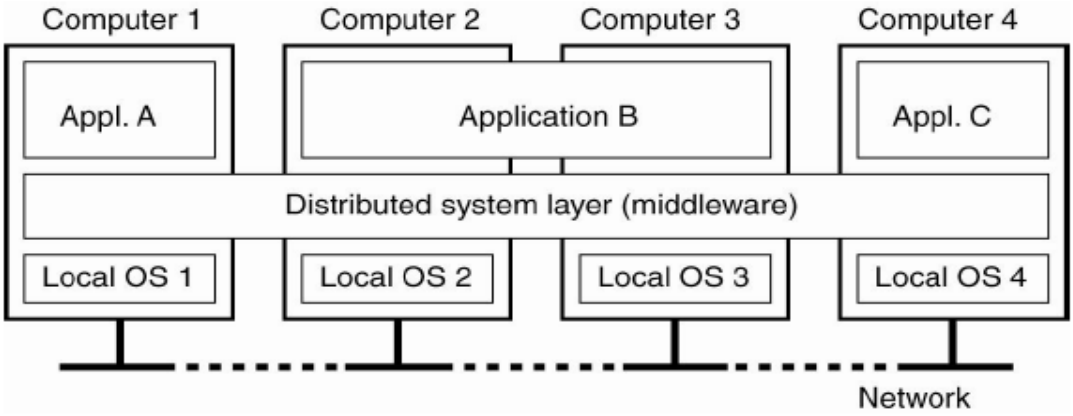
Essa definição aborda aspectos importantes. O primeiro deles é referente aos componentes de um sistema distribuído, por exemplo, os computadores, os quais são autônomos. Um segundo aspecto é sobre os usuários (sejam eles pessoas ou programas) pensarem que estão lidando com um sistema único. Isto significa que, de uma forma ou de outra, os componentes autônomos precisam colaborar.

Uma importante característica é que as diferenças entre os vários componentes e os caminhos pelos quais eles se comunicam são abstraídos da percepção dos usuários. O mesmo se aplica para a organização interna de um sistema distribuído. Outra característica importante é que usuários e aplicações podem interagir com um sistema distribuído de um modo consistente e uniforme, independentemente de onde e quando a interação ocorre.

A princípio, os sistemas distribuídos devem ser relativamente fáceis de expandir, ou seja, devem possuir escalabilidade. Essa característica é uma conseqüência direta da presença de diferentes computadores, mas ao mesmo tempo, escondendo como esses computadores realmente participam do sistema como um todo (Andrew S Tanenbaum & Steen, 2006).

No intuito de comportar computadores e redes heterogêneas, enquanto oferecem a visão de um sistema único, os sistemas distribuídos muitas vezes são organizados por uma camada de software que é, logicamente, logicamente localizada entre um camada de alto nível que consiste de usuários e aplicações, e uma camada abaixo compreendida por sistemas operacionais e facilidades básicas de comunicação, como mostrado na figura 4, de forma adequada, um sistema distribuído é por vezes chamado *middleware* (Andrew S Tanenbaum & Steen, 2006).

Figura 4 - Sistema distribuído organizado como *middleware*.



Fonte: (Andrew S Tanenbaum & Steen, 2006)

A camada de *middleware* se estende através de múltiplas máquinas e oferece a cada aplicação a mesma interface. A figura 4 mostra quatro computadores em rede e três aplicações, na qual a aplicação B é distribuída através dos computadores 2 e 3.

Cada aplicação é oferecida com a mesma interface. Um sistema distribuído proporciona os meios para os componentes de uma única aplicação distribuída se comunicarem uns com os outros, mas também para permitir diferentes aplicações se comunicarem. Ao mesmo tempo em que esconde as diferenças de hardware e sistemas operacionais de cada aplicação.

Somente porque é possível construir sistemas distribuídos, isto não significa que seja, necessariamente, uma boa idéia. Um sistema distribuído deve fazer com que os recursos se tornem facilmente acessíveis; deve ocultar o fato dos recursos estarem distribuídos através de uma rede; deve ser aberto; e deve ser escalável (Andrew S Tanenbaum & Steen, 2006).

Os sistemas distribuídos têm, por seu maior objetivo, melhorar a comunicação entre os computadores, ou melhor, propiciar a integração destes num sentido amplo que pode envolver facilidade de mudanças futuras, rapidez nas trocas de informações, confiabilidade na execução dos processos. Podem ser considerados também como a evolução para os sistemas fortemente acoplados, onde uma aplicação pode ser executada por qualquer processador. Os sistemas distribuídos permitem que uma aplicação seja dividida em diferentes partes, que se comunicam através de linhas de comunicação, e cada parte podendo ser processada em um sistema independente.

O objetivo é criar na cabeça de seus usuários a ilusão de que toda rede de computadores nada mais é do que um único sistema de tempo compartilhado (time-sharing), em vez de um conjunto de máquinas distintas (A S Tanenbaum, 1995).

Um importante objetivo de um sistema distribuído é o fato de esconder que os processos e recursos estão fisicamente distribuídos através de múltiplos computadores. Um sistema distribuído que é capaz de se apresentar aos usuários e aplicações como se fosse somente um sistema de computador único é chamado de transparente (Andrew S Tanenbaum & Steen, 2006). O conceito de transparência pode ser aplicado para diversos aspectos de um sistema distribuído, sendo os mais importantes os elencados na tabela 1.

Tabela 1 - Diferentes formas de transparência em um sistema distribuído.

Transparência	Descrição
Acesso	Oculta as diferenças de representação de dados e como um recurso é acessado. Esta transparência resolve muitos problemas de interoperabilidade entre sistemas heterogêneos.
Localização	Oculta o uso de informações sobre onde um recurso está localizado.
Migração	Oculta que um recurso pode ser movido para outra localização. É por vezes utilizado para reduzir a latência e alcançar o balanceamento de carga.
Realocação	Oculta que um recurso pode ser movido para outra localização enquanto em uso.
Replicação	Oculta que um recurso é replicado. É muitas vezes usado para melhorar o desempenho e disponibilidade.
Falha	Oculta a falha e a recuperação de um recurso.
Persistência	Oculta se um recurso está na memória ou em disco. Desativação e reativação são muitas vezes utilizados para manter a persistência de um objeto quando o sistema não é capaz de fornecê-lo com funções de processamento, armazenamento e comunicação de forma contínua.
Transação	Oculta a coordenação de atividades entre uma configuração de objetos para obter consistência.

Fonte: (ISO/IEC, 1995)

2.5 Processamento Paralelo

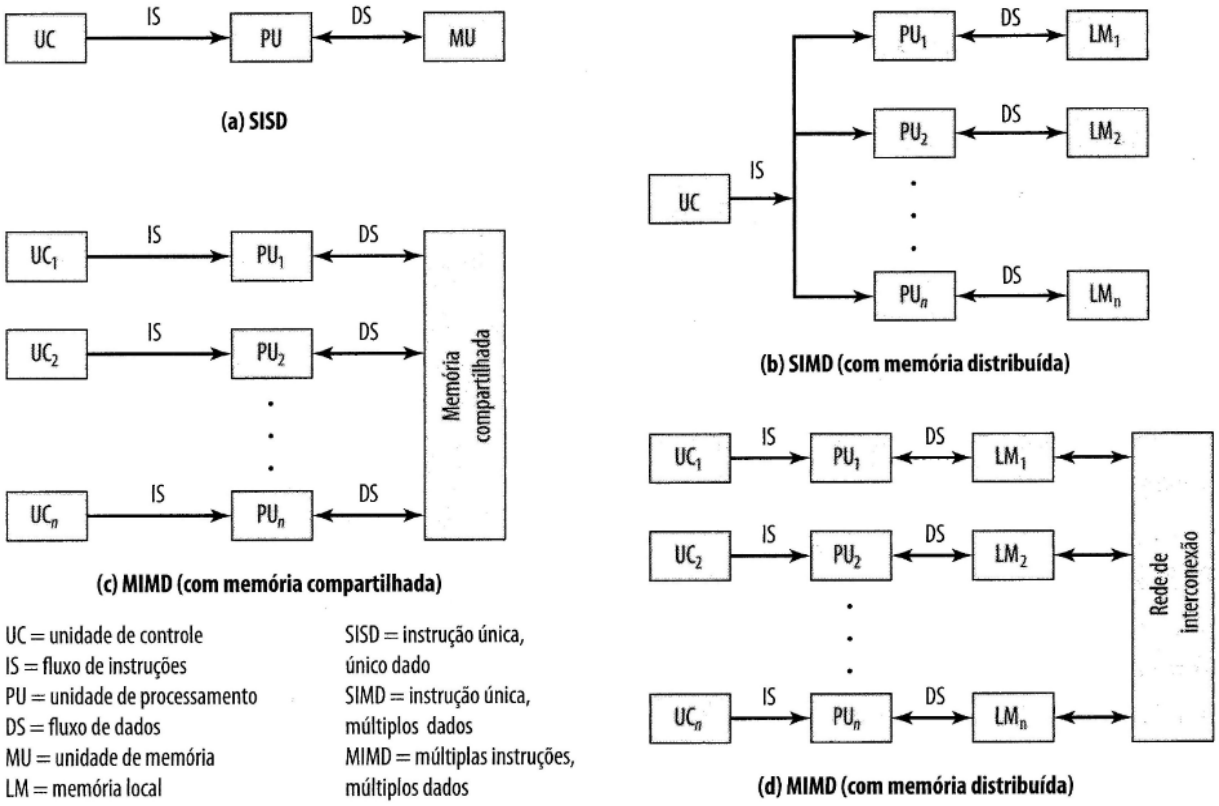
Um jeito tradicional para melhorar o desempenho do sistema é utilizar múltiplos processadores que possam executar em paralelo para suportar uma certa carga de trabalho. Duas organizações mais comuns de múltiplos processadores são multiprocessadores simétricos (SMP, do inglês *symmetric multiprocessor*) e clusters. Mais recentemente, sistemas de acesso não uniforme à memória (NUMA, do inglês *nonuniform memory acess*) foram introduzidos comercialmente.

2.6 Organizações de Múltiplos Processadores

Uma taxonomia introduzida inicialmente por Flynn (Flynn, 1972) é ainda a maneira mais comum de categorizar sistemas com capacidade de processamento paralelo. Flynn propôs as seguintes categorias de sistemas computacionais:

- Instrução única, único dado (SISD, do inglês *single instruction, single data*): um processador único executa uma única seqüência de instruções para operar nos dados armazenados em uma única memória. Uniprocessadores enquadram-se nessa categoria.
- Instrução única, múltiplos dados (SIMD, do inglês *single instruction, multiple data*): uma única instrução de máquina controla a execução simultânea de uma série de elementos de processamento em operações básicas. Cada elemento de processamento possui uma memória de dados associada, então cada instrução é executada em um conjunto diferente de dados por processadores diferentes. Processadores de vetores e matrizes se enquadram nessa categoria.
- Múltiplas instruções, único dado (MISD, do inglês *multiple instruction, single data*): uma seqüência de dados é transmitida para um conjunto de processadores, onde cada um executa uma seqüência de instruções diferente. Esta estrutura não é implementada comercialmente.
- Múltiplas instruções, múltiplos dados (MIMD, do inglês *multiple instruction, multiple data*): um conjunto de processadores que executam seqüências de instruções diferentes simultaneamente em diferentes conjuntos de dados. SMPs, clusters e sistemas NUMA enquadram-se nessa categoria.

Figura 5 - Organizações alternativas de computadores.



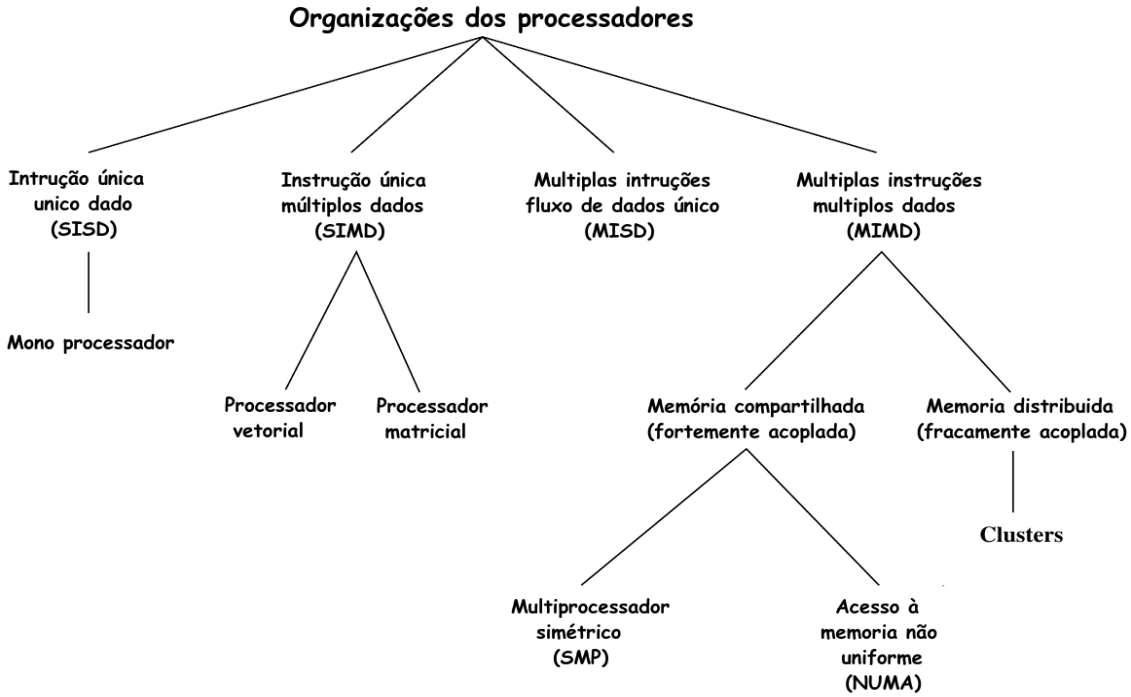
Fonte: (Stallings, 2010)

Stallings (Stallings, 2010), com a organização MIMD, os processadores são de uso geral; cada um é capaz de processar todas as instruções necessárias para efetuar transformação de dados apropriada. MIMDs podem ser ainda divididos pelos meios de comunicação do processador (Figura 6). Se os processadores compartilham uma memória comum, então cada processador acessa programas e dados armazenados na memória compartilhada e os processadores se comunicam com os outros por meio dessa memória. A forma mais comum desse sistema é conhecida como multiprocessador simétrico (SMP).

Em um SMP, múltiplos processadores compartilham uma única memória ou um pool de memória por um barramento compartilhado ou algum outro mecanismo de interconexão; um recurso diferenciado é que o tempo de acesso à memória de qualquer região de memória é aproximadamente o mesmo para cada processador.

Um desenvolvimento mais recente é a organização de acesso não uniforme à memória (NUMA); como o próprio nome sugere, o tempo de acesso à memória de diferentes regiões da memória pode diferir para um processador NUMA. Uma coleção de uniprocessadores independentes ou SMPs pode ser interconectada para formar um cluster. A comunicação é feita por caminhos fixos ou alguma facilidade de rede.

Figura 6 - Uma taxonomia de arquiteturas de processadores paralelos.



Fonte: (Stallings, 2010)

2.7 Multiprocessadores Simétricos

Até recentemente, quase todos os computadores pessoais e a maioria das estações de trabalho continham um único processador de propósito geral. À medida que a demanda de desempenho aumenta e o custo de microprocessadores continua a baixar, os fabricantes têm introduzido sistemas com uma organização SMP. O termo SMP refere-se a uma arquitetura de hardware computacional e também ao comportamento do sistema operacional que reflete essa arquitetura.

Conforme W. Stallings (Stallings, 2010), um SMP pode ser definido como um sistema de computação independente com as seguintes características:

1. Há dois ou mais processadores semelhantes de capacidade comparável.
2. Esses processadores compartilham a mesma memória principal e os recursos de E/S, e são interconectados por um barramento ou algum outro esquema de conexão interna, de tal forma que o tempo de acesso à memória é aproximadamente igual para cada processador.
3. Todos os processadores compartilham acesso aos dispositivos de E/S, ou pelo menos canais ou por canais diferentes que fornecem caminhos para o mesmo dispositivo.
4. Todos os processadores desempenham as mesmas funções (daí o termo simétrico).
5. O sistema é controlado por um sistema operacional integrado que fornece interação entre processadores e seus programas em nível de trabalhos, tarefas, arquivos ou elementos de dados.

O autor esclarece que o item 5 ilustra um dos contrastes com um sistema de multiprocessamento fracamente acoplado, como um cluster. No cluster, a unidade física de interação é normalmente uma mensagem ou um arquivo completo. A sincronização da operação no cluster é obtida utilizando métodos como *Parallel Virtual Machine* (PVM), que capacita um código a executar através de vários nós, permitindo que um grupo heterogêneo de máquinas execute C, C++ e Fortran através do cluster, outra forma, é utilizando o *Message Passing Interface* (MPI), que é um protocolo que se utiliza da passagem de mensagens para sincronização de processos (Pitanga, 2008). Em um SMP, elementos individuais de dados podem construir o nível de interação e pode haver um alto grau de cooperação entre processos.

2.8 Clusters

Podemos definir um cluster como um grupo de computadores completos interconectados trabalhando juntos, como um recurso computacional unificado que pode criar a ilusão de ser uma única máquina. O termo computador completo significa um sistema que pode funcionar por si só, à parte do cluster; na literatura, cada computador em um cluster normalmente é chamado de nó (Stallings, 2010).

Quatro objetivos ou requisitos do projeto podem ser alcançados com cluster segundo Brewer (Brewer, 1997):

- Escalabilidade absoluta: é possível criar grandes clusters que ultrapassam e muito o poder de máquinas maiores que trabalham sozinhas. Um cluster pode ter dezenas, centenas ou até milhares de máquinas, cada uma sendo um multiprocessador.
- Escalabilidade incremental: um cluster é configurado de tal forma que é possível adicionar novos sistemas ao cluster em incrementos pequenos. Assim, um usuário pode começar com um sistema modesto e expandi-lo conforme a necessidade, sem ter que fazer uma atualização grande onde um sistema existente pequeno é substituído por um sistema maior.
- Alta disponibilidade: como cada nó do cluster é um computador independente, a falha de um nó não significa a perda do serviço. Em muitos produtos, a tolerância a falhas é tratada automaticamente por software.
- Preço/desempenho superior: usando a idéia de blocos de construção, é possível montar um cluster com poder computacional igual ou maior do que uma única máquina de grande porte, com um custo bem menor.

Conforme (Bookman, 2002) a computação agrupada, em seu nível mais básico, envolve dois ou mais computadores, servindo em único recurso. A grande maioria dos clusters (paralelos) possui a seguinte configuração: uma máquina denominada servidor, que faz a interface com o operador (que programa serviços e monitora processo e resultados), recebe os dados, é responsável pela paralelização do processamento entre os nodos clientes (máquinas que são utilizadas para fazer parte do processamento exigido pela aplicação), posteriormente o servidor recolhe os resultados obtidos pelos nodos cliente e os organiza obtendo a resposta final do processamento.

2.8.1 Tipos de Clusters

Os tipos de clusters, de acordo com (Pitanga, 2008), consistem em: alta disponibilidade (do inglês, *high availability* - HA); balanceamento de carga; combinação HA & balanceamento de carga; e processamento distribuído ou paralelo. A conceituação dos tipos de clusters foi extraída das obras de Marcos Pìtanga (Pitanga, 2008) e W Stallings (Stallings, 2010).

2.8.1.1 Gerenciamento de Falhas

Modelos construídos para prover disponibilidade de serviços e recursos de forma ininterrupta através do uso da redundância implícita ao sistema. Objetiva a disponibilidade de aplicações ou serviços caso um nó do cluster venha falhar (*failover*). Em geral, duas abordagens podem ser usadas para lidar com falhas: clusters de alta disponibilidade e clusters com tolerância a falhas. Um cluster de alta disponibilidade provê uma probabilidade elevada de que todos os recursos estejam em funcionamento. Caso ocorra uma falha, como um desligamento de sistema ou perda de um volume de disco, então as consultas em progresso são perdidas. Qualquer consulta perdida será tentada em um computador diferente do cluster. No entanto, o sistema operacional de cluster não dá garantia alguma sobre o estado de transações executadas parcialmente. Isso deve ser tratado em nível de aplicações.

Um cluster com tolerância a falhas garante que todos os recursos estejam sempre disponíveis. Isso é alcançado com o uso de discos compartilhados redundantes e mecanismos para retornar as transações não encerradas e encerrar transações completadas.

A função de trocar as aplicações e recursos de dados de um sistema que falhou para um sistema alternativo no cluster é conhecida como *failover* (recuperação de falhas). Uma função relacionada é a restauração de aplicações e recursos de dados para o sistema original quando o mesmo for consertado; isto é chamado de *failback* (retorno à operação).

2.8.1.2 Balanceamento de Carga (*Load Balancing*)

Um cluster requer uma capacidade eficiente para balancear a carga entre computadores disponíveis. Isto inclui o requisito de que o cluster seja incrementalmente escalável.

Quando um novo computador é adicionado ao cluster, o recurso de balanceamento de carga deve automaticamente incluir esse computador no agendamento de aplicações. Mecanismos de *middleware* precisam reconhecer que serviços podem aparecer em diferentes membros do cluster e muitos podem migrar de um membro para outro.

As técnicas envolvidas no balanceamento de carga e agendamento de tarefas desempenham os papéis principais em atingir uma parcela igual de carga de trabalho total de cada processo e ajudam a minimizar o tempo total de execução (Diekmann & Monien, 1997) (Kumar, 2010) (Bookman, 2002).

2.8.1.3 Combinação *HA & Load Balancing*

Como o próprio nome diz, combina as características dos dois tipos de cluster, aumentando assim a disponibilidade e escalabilidade de serviços e recursos.

2.8.1.4 Processamento Distribuído ou Processamento Paralelo

Aumenta a disponibilidade e desempenho para as aplicações, particularmente as grandes tarefas computacionais. Uma grande tarefa computacional pode ser dividida em pequenas tarefas que são distribuídas ao redor das estações (nós), como se fosse um supercomputador massivamente paralelo. Em alguns casos, o uso eficiente de um cluster requer executar software de uma única aplicação em paralelo. Conforme Kapp (Kapp, 2000) lista três abordagens gerais para o problema:

- Computação paralela: uma compilação paralela determina, em tempos de compilação, quais partes de uma aplicação podem ser executadas em paralelo. Elas são então separadas para serem atribuídas a diferentes computadores no cluster. O desempenho depende da natureza do problema e quão bem o computador é projetado. Em geral, tais compiladores são difíceis de desenvolver.
- Aplicações paralelas: nesta abordagem, o programador escreve a aplicação desde o começo para ser executada em um cluster e utiliza passagem de mensagens para mover dados, conforme necessário, entre os nós do cluster. Isso coloca uma grande responsabilidade no programador, mas pode ser a melhor abordagem para explorar clusters para algumas aplicações.
- Computação paramétrica: esta abordagem pode ser usada se a essência da aplicação for um algoritmo ou um programa que deva ser executado em grande número de vezes, cada vez com um conjunto diferente de condições iniciais ou parâmetros. Um bom exemplo é um modelo de simulação, o qual vai executar um grande número de cenários e depois desenvolver resumos estatísticos dos resultados. Para que essa abordagem seja eficiente, ferramentas de processamento paramétrico são necessárias para organizar, executar e gerenciar os trabalhos de uma forma eficiente.

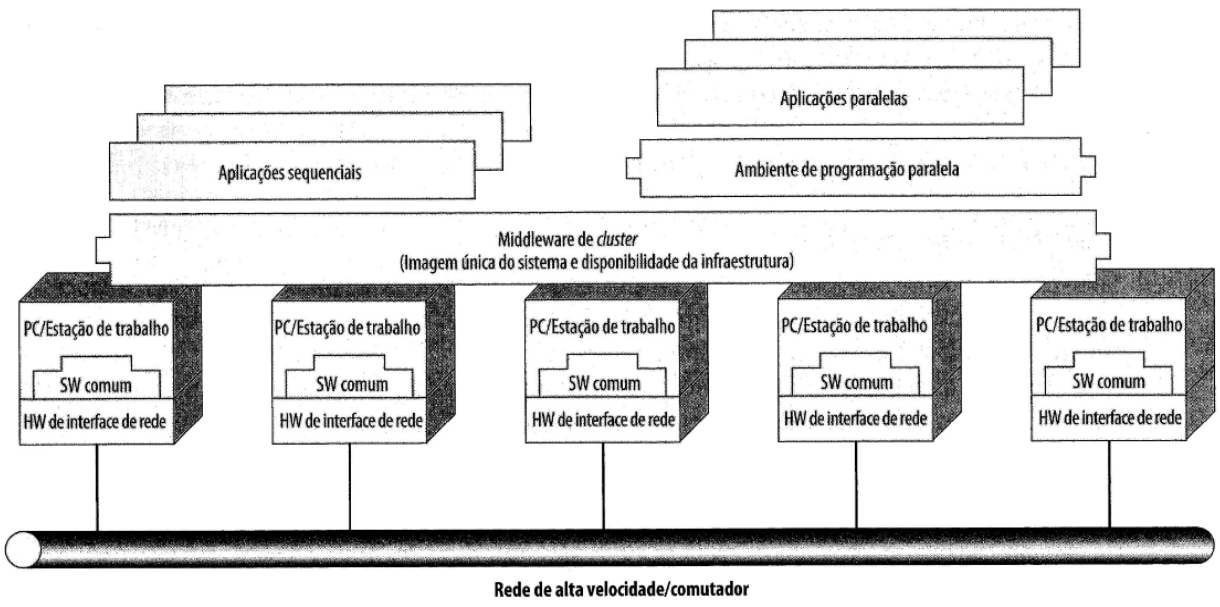
2.8.2 Arquitetura de um Cluster Computacional

De acordo com Andrew S. Tanembaum (2003), multicomputadores são CPUs fortemente acopladas que não compartilham memória. Cada CPU tem sua própria memória local. Esses sistemas são também conhecidos por uma variedade de outros nomes, como computadores cluster e COWS (*clusters of workstations* - clusters de estações de trabalho). O segredo da obtenção de alto desempenho é projetar de modo inteligente a rede e a placa de interface, sendo assim simples, barato e fácil de implementar.

Os computadores individuais são conectados por alguma LAN de alta velocidade ou hardware de comutação. Cada computador é capaz de operar independentemente. Além disso, uma camada intermediária de software é instalada em cada computador para possibilitar a operação do cluster. A figura 7 mostra uma arquitetura típica de cluster; um cluster indicará também ferramentas de software para habilitar a execução eficiente de programas que são capazes de executar execução paralela.

O *middleware* do cluster fornece uma imagem unificada do sistema para o usuário, conhecida como imagem de sistema único. O *middleware* é responsável também por fornecer alta disponibilidade pelo balanceamento de carga e respostas a falhas em componentes individuais (Stallings, 2010).

Figura 7 - Arquitetura de um cluster computacional.



Fonte: (R Buyya, 1999)

Os serviços e as funções desejáveis para um *middleware* de cluster foram listados por Hwang et. al. (Hwang, Jin, Chow, Wang, & Xu, 1999):

- Ponto de entrada único: o usuário efetua o acesso no cluster em vez de fazê-lo em um computador individual.
- Hierarquia única de arquivos: o usuário vê uma arquitetura única de diretórios de arquivos abaixo do mesmo diretório raiz.
- Rede virtual única: qualquer nó pode acessar qualquer outro ponto no cluster, mesmo que a configuração atual do cluster consista em múltiplas redes interconectadas. Há uma operação de rede virtual única.
- Espaço único de memória: memória compartilhada distribuída possibilita que os programas compartilhem variáveis.
- Sistema único de gerenciamento de trabalhos: com um gerente de trabalhos do cluster, um usuário pode submeter um trabalho sem especificar qual computador executará o trabalho.
- Interface de usuário única: uma interface gráfica comum suporta todos os usuários, independentemente da estação de trabalho da qual acessaram o cluster.
- Espaço de entrada/saída único: qualquer nó pode acessar remotamente qualquer periférico de entrada/saída ou dispositivo de disco sem conhecer a sua localização física.
- Espaço único de processos: um esquema uniforme de identificação de processos é usado. Um processo em qualquer nó pode criar ou se comunicar com qualquer outro processo em um nó remoto.
- Pontos de verificação: essa função periodicamente salva o estado dos processos e resultados computacionais intermediários para permitir recuperação em caso de falhas.
- Migração de processos: essa função habilita o balanceamento de carga.

Os quatro últimos itens da lista anterior aprimoram a disponibilidade do cluster. Os itens restantes se preocupam em fornecer uma imagem única do sistema.

2.9 Cloud Computing

Com o avanço da sociedade humana moderna, serviços básicos e essenciais são quase todos entregues de uma forma completamente transparente. Serviços de utilidade pública como água, eletricidade, telefone e gás tornaram-se fundamentais para nossa vida diária e são explorados por meio do modelo de pagamento baseado no uso (Vecchiola et al., 2009).

As infraestruturas existentes permitem entregar tais serviços em qualquer lugar e a qualquer hora, de forma que possamos simplesmente acender a luz, abrir a torneira ou levantar o telefone do gancho. O uso desses serviços é, então, cobrado de acordo com as diferentes políticas de tarifação para o usuário final. Recentemente, a mesma idéia de utilidade tem sido aplicada no contexto da informática e uma mudança consistente neste sentido tem sido feita com a disseminação de *Cloud Computing* ou Computação em Nuvem.

Cloud Computing ou Computação em Nuvem é um termo amplamente utilizado atualmente e se refere, essencialmente, à idéia de utilizarmos, em qualquer lugar e independente de plataforma, as mais variadas aplicações por meio da internet com a mesma facilidade de tê-las instaladas em nossos próprios computadores. Esse novo modelo de computação tem surgido e alterado a forma como interagimos com a rede e com os serviços e aplicações. Sendo uma tendência recente de tecnologia, cujo objetivo é proporcionar serviços de Tecnologia da Informação (TI) sob demanda com pagamento baseado no uso. Tendências anteriores à computação em nuvem foram limitadas a uma determinada classe de usuários ou focadas em tornar disponível uma demanda específica de recursos de TI, principalmente de informática (Buyya et al., 2009).

O próprio termo *cloud* (nuvem) é oriundo da telefonia e posteriormente foi adotado como metáfora para descrever a Internet nos diagramas de rede. Essa imagem de nuvem indicava algo amorfo, inatingível, mas necessário para ser incluído no diagrama.

As linhas desenhadas no diagrama cruzavam por dentro da nuvem para indicar que o fluxo de dados simplesmente passava pela Internet. Hoje, a ilustração da nuvem deixou de ser algo abstrato e passou a ser o cerne da computação (Taurion, 2009).

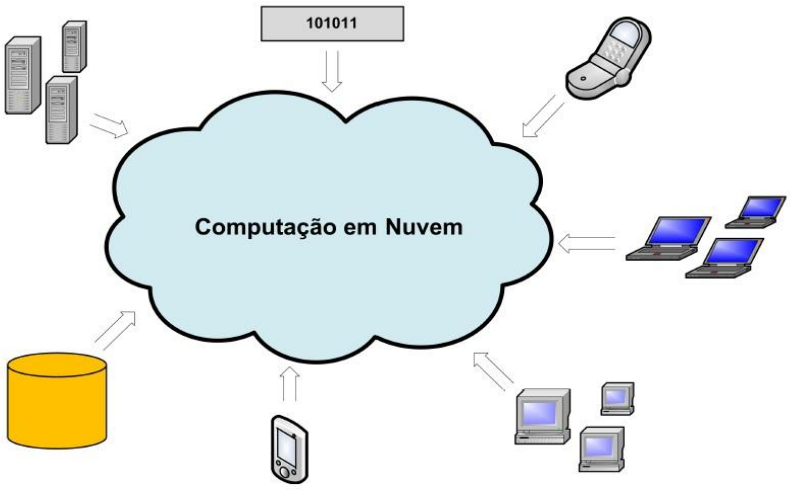
O conceito de Computação em Nuvem já é comum em algumas das empresas mais famosas da Internet como Google, Amazon e o Yahoo, que mantêm parques computacionais com centenas de milhares de máquinas. Estima-se, por exemplo, o Google tenha cerca de uma dúzia de *data centers* espalhados pelo mundo e que sua infraestrutura, chamada de

Googleplex, compreenda mais de 200 petabytes de disco e cerca de 500.000 servidores, que conseguem processar mais de 100 milhões de consultas por dia. Aliás, estudos têm mostrado que um punhado de companhias e serviços da Internet, como o MSN da Microsoft, Google, Yahoo, Amazon e Ask, já respondem pela compra de 20% da produção mundial de servidores (Taurion, 2009).

Cada vez mais o processamento e o armazenamento estão sendo movidos dos PCs para grandes provedores de serviços. Esta mudança para um modelo centrado no servidor não traz vantagens apenas para o usuário que fica liberado de toda a gerência local necessária para manter as aplicações (intensas configurações e grandes quantidades de backups), mas também traz vantagens para os produtores de software (Hoelzle & Barroso, 2009).

O desenvolvimento de aplicações é mais simples pois as mudanças e as melhorias nos softwares são feitas em um único local, ao invés de serem feitas em milhões de clientes que possuem diferentes configurações de hardware. Além disso, uma vez que o uso do software hospedado no provedor passa a ser compartilhado por diversos usuários, o custo também se divide entre estes usuários, fazendo com que os valores pela utilização dos serviços sejam reduzidos. Em termos gerais, a computação se torna mais barata quando vista como um serviço compartilhado. Esta premissa tem sido discutida durante muito tempo mas, até o momento, o modelo centrado no usuário, com máquinas e aplicações individuais, tem sido dominante.

Figura 8 - Visão geral de uma nuvem computacional.



Fonte: (Rajkumar Buyya et al., 2009)

2.9.1 Definições e Terminologia

Conforme Vaquero et al. (2009) é adotada a seguinte opção: "*Cloud computing* é um conjunto de recursos virtuais facilmente usáveis e acessíveis tais como hardware, plataformas de desenvolvimento e serviços. Estes recursos podem ser dinamicamente reconfigurados para se ajustarem a uma carga variável, permitindo a otimização do uso dos recursos. Este conjunto de recursos é tipicamente explorado através de um modelo *pay-per-use* com garantias oferecidas pelo provedor através de acordos de nível de serviço (*Service Level Agreements*-SLAs)."

Figura 9 - Visão geral da computação em nuvem.

Fonte: (Vaquero et al., 2009)

O modelo de computação em nuvem é composto, tipicamente, por cinco características essenciais, três modelos de serviços e quatro modelos de implantação da nuvem (Mell & Grance, 2009), conforme descrito abaixo.

2.9.2 Características Essenciais

2.9.2.1 Self-service sob Demanda

O usuário pode adquirir unilateralmente recurso computacional, como tempo de processamento no servidor ou armazenamento na rede, na medida em que necessite e sem precisar de interação humana com os provedores de cada serviço.

O hardware e o software dentro de uma nuvem podem ser automaticamente reconfigurados, orquestrados e estas modificações são apresentadas de forma transparente para os usuários, que possuem perfis diferentes e assim podem personalizar os seus ambientes computacionais, por exemplo, instalação de software e configuração de rede para a definição de determinados privilégios.

2.9.2.2 Amplo Acesso aos Serviços

Recursos são disponibilizados por meio da rede e acessados através de mecanismos padronizados que possibilitam o uso por plataformas *thin* ou *thin client*, tais como celulares, laptops e PDAs. A interface de acesso à nuvem não obriga os usuários a mudar suas condições e ambientes de trabalho, como por exemplo, linguagens de programação e sistema operacional. Já os sistemas de software clientes instalados localmente para o acesso à nuvem são leves, como um navegador de Internet.

2.9.2.3 Pooling de Recursos

Os recursos computacionais do provedor são organizados em um *pool* para servir múltiplos usuários usando um modelo *multi-tenant* ou multi-inquilino (Jacobs, Aulbach, & München, 2007), com diferentes recursos físicos e virtuais, dinamicamente atribuídos e ajustados de acordo com a demanda dos usuários. Estes usuários não precisam ter conhecimento da localização física dos recursos computacionais, podendo somente especificar a localização em um nível mais alto de abstração, tais como o país, estado ou centro de dados.

2.9.2.4 Elasticidade Rápida

Recursos podem ser adquiridos de forma rápida e elástica, em alguns casos automaticamente, caso haja a necessidade de escalar com o aumento da demanda, e liberados, na retração dessa demanda. Para os usuários, os recursos disponíveis para uso parecem ser ilimitados e podem ser adquiridos em qualquer quantidade e a qualquer momento. A virtualização auxilia a elasticidade rápida na computação nuvem, criando várias instâncias de recursos requisitados utilizando um único recurso real (Aboulnaga et al., 2009). Além disso, a virtualização é uma maneira de abstrair características físicas de uma plataforma computacional dos usuários, exibindo outro hardware virtual e emulando um ou mais ambientes que podem ser independentes ou não.

2.9.2.5 Serviço Medido

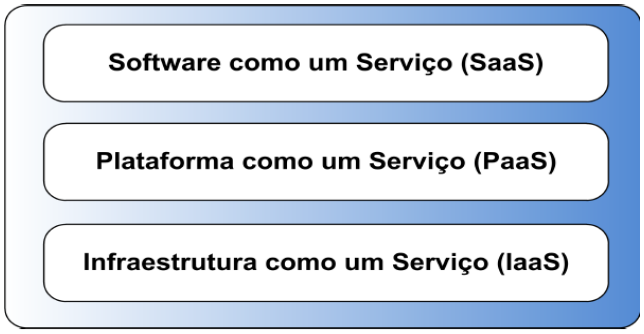
Sistemas em nuvem automaticamente controlam e otimizam o uso de recursos por meio de uma capacidade de medição. A automação é realizada em algum nível de abstração apropriado para o tipo de serviço, tais como armazenamento, processamento, largura de

banda e contas dos usuários ativas. O uso de recursos pode ser monitorado e controlado, possibilitando transparência para o provedor e o usuário do serviço utilizado. Para garantir a qualidade do serviço ou *Quality of Service* (QoS), pode-se utilizar a abordagem baseada em acordo de nível de serviço ou *Services Level Agreement* (SLA). O SLA fornece informações sobre os níveis de disponibilidade, funcionalidade, desempenho ou outros atributos do serviço como o faturamento e até mesmo penalidades em caso de violação destes níveis.

2.9.3 Modelos de Serviços

O ambiente de computação em nuvem é composto de três modelos de serviços. Estes modelos são importantes, pois eles definem um padrão arquitetural para soluções de computação em nuvem. A Figura 10 exibe estes modelos de serviços (Armbrust et al., 2009).

Figura 10 - Modelo de serviços.



Fonte: (Armbrust et al., 2009)

2.9.3.1 Software como um Serviço (*Software as a Service* - SaaS)

O modelo de SaaS proporciona sistemas de software com propósitos específicos que estão disponíveis para os usuários através da Internet. Os sistemas são acessíveis a partir de vários dispositivos do usuário por meio de uma interface *thin client* como um navegador Web. No SaaS, o usuário não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistemas operacionais, armazenamento ou mesmo as características individuais da aplicação, exceto configurações específicas. Com isso, os desenvolvedores se concentram em inovação e não na infraestrutura, levando ao desenvolvimento rápido de sistemas de software.

Como o software está na Web, ele pode ser acessado pelos usuários de qualquer lugar e a qualquer momento, permitindo maior integração entre unidades de uma mesma empresa ou outros serviços de software. Assim, novos recursos podem ser incorporados automaticamente aos sistemas de software sem que os usuários percebam estas ações, tornando transparente a evolução e atualização dos sistemas. O SaaS reduz os custos, pois é dispensada a aquisição de licenças. Como exemplos de SaaS podemos destacar os serviços de *Customer Relationship Management* (CRM) da Salesforce (Salesforce, 2010) e o Google Docs (Ciurana, 2009).

2.9.3.2 Plataforma como um Serviço (*Plataform as a Service* - PaaS)

A PaaS oferece uma infraestrutura de alto nível de integração para implementar e testar aplicações na nuvem. O usuário não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistemas operacionais ou armazenamento, mas tem controle sobre as aplicações implantadas e, possivelmente, as configurações das aplicações hospedadas nesta infraestrutura. A PaaS fornece um sistema operacional, linguagens de programação e ambientes de desenvolvimento para as aplicações, auxiliando a implementação de sistemas de software, já que contém ferramentas de desenvolvimento e colaboração entre desenvolvedores.

Em geral, os desenvolvedores dispõem de ambientes escaláveis, mas eles têm que aceitar algumas restrições sobre o tipo de software que se pode desenvolver, desde limitações que o ambiente impõe na concepção das aplicações até a utilização de sistemas de gerenciamento de banco de dados (SGBDs) do tipo chave-valor, ao invés de SGBDs relacionais.

Do ponto de vista do negócio, a PaaS permitirá aos usuários utilizarem serviços de terceiros, aumentando o uso do modelo de suporte no qual os usuários se inscrevem para solicitações de serviços de TI ou para resoluções de problemas pela Web. Com isso, pode-se melhorar o gerenciamento do trabalho e as responsabilidades das equipes de TI das empresas. Como exemplos de SaaS podemos destacar as PaaS Google App Engine (Ciurana, 2009) e Aneka (Vecchiola et al., 2009).

2.9.3.3 Infraestrutura como um Serviço (*Infraestructure as a Service* - IaaS)

O IaaS é a parte responsável por prover toda a infraestrutura necessária para a PaaS e o SaaS. O principal objetivo do IaaS é tornar mais fácil e acessível o fornecimento de recursos, tais como servidores, rede, armazenamento e outros recursos de computação fundamentais para construir um ambiente sob demanda, que podem incluir sistemas operacionais e aplicativos. A IaaS possui algumas características, tais como uma interface única para administração da infraestrutura, *Application Programming Interface* (API) para interação com hosts, switches, balanceadores, roteadores e o suporte para a adição de novos equipamentos de forma simples e transparente.

Em geral, o usuário não administra ou controla a infraestrutura da nuvem, mas tem controle sobre os sistemas operacionais, armazenamento e aplicativos implantados, e, eventualmente, seleciona componentes de rede, tais como firewalls.

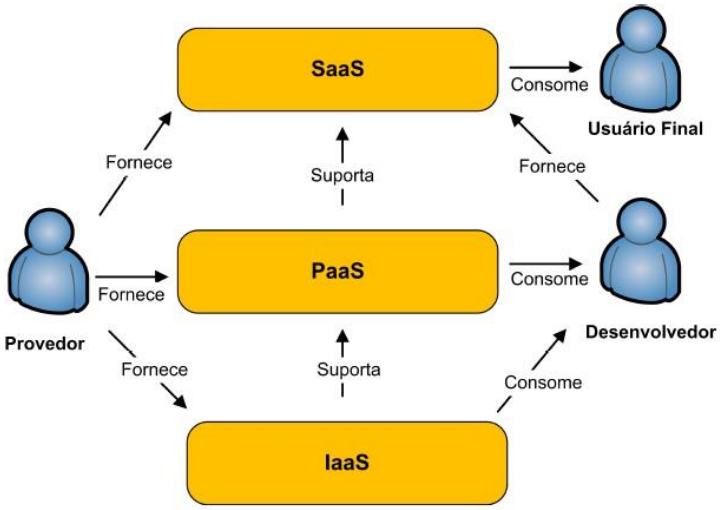
O termo IaaS se refere a uma infraestrutura computacional baseada em técnicas de virtualização de recursos de computação. Esta infraestrutura pode escalar dinamicamente, aumentando ou diminuindo os recursos de acordo com as necessidades das aplicações. Do ponto de vista de economia e aproveitamento do legado, ao invés de comprar novos servidores e equipamentos de rede para a ampliação de serviços, pode-se aproveitar os recursos disponíveis e adicionar novos servidores virtuais à infraestrutura existente de forma dinâmica.

O *Amazon Elastic Cloud Computing* (EC2) (Robinson, 2008) e o *Elastic Utility Computing Architecture Linking Your Programs To Useful Systems* (Eucalyptus) (Liu, Liang, & Brooks, 2007) são exemplos de IaaS.

2.9.4 Papéis na Computação em Nuvem

Os papéis são importantes para definir responsabilidades, acesso e perfil para os diferentes usuários que fazem parte e estão envolvidos em uma solução de computação em nuvem. Para entender melhor a computação em nuvem, pode-se classificar os atores dos modelos de acordo com os papéis desempenhados (Marinos & Briscoe, 2009). A Figura 10 destaca estes papéis.

Figura 11 - Papéis na computação em nuvem.



Fonte: (Marinos & Briscoe, 2009)

2.9.5 Modelos de Implantação

Tratando-se do acesso e disponibilidade de ambientes de computação em nuvem, têm-se diferentes tipos de modelos de implantação. A restrição ou abertura de acesso depende do processo de negócio, do tipo de informação e do nível de visão. Pode-se perceber que certas empresas não desejam que todos os usuários possam acessar e utilizar determinados recursos no seu ambiente de computação em nuvem. Neste sentido, surge a necessidade de ambientes mais restritos, onde somente alguns usuários devidamente autorizados possam utilizar os serviços providos. Os modelos de implantação da computação em nuvem podem ser divididos em nuvem pública, privada, comunidade e híbrida (Mell & Grance, 2009).

2.9.5.1 Nuvem Privada (*private clouds*)

No modelo de implantação de nuvem privada, a infraestrutura de nuvem é utilizada exclusivamente para uma organização, sendo esta nuvem local ou remota e administrada pela própria empresa ou por terceiros. Neste modelo de implantação são empregados políticas de acesso aos serviços. As técnicas utilizadas para prover tais características podem ser em nível de gerenciamento de redes, configurações dos provedores de serviços e a utilização de tecnologias de autenticação e autorização.

2.9.5.2 Nuvem Comunidade (*community cloud*)

No modelo de implantação de nuvem comunidade ocorre o compartilhamento por diversas empresas de uma nuvem, sendo esta suportada por uma comunidade específica que partilhou seus interesses, tais como a missão, os requisitos de segurança, política e considerações sobre flexibilidade. Este tipo de modelo de implantação pode existir localmente ou remotamente e geralmente é administrado por alguma empresa da comunidade ou por terceiros.

2.9.5.3 Nuvem Pública (*public cloud*)

No modelo de implantação de nuvem pública, a infraestrutura de nuvens é disponibilizada para o público em geral, sendo acessado por qualquer usuário que conheça a localização do serviço. Neste modelo de implantação não podem ser aplicadas restrições de acesso quanto ao gerenciamento de redes, e menos ainda, utilizar técnicas para autenticação e autorização.

2.9.5.4 Nuvem Híbrida (*hybrid cloud*):

No modelo de implantação de nuvem híbrida, existe uma composição de duas ou mais nuvens, que podem ser privadas, comunidade ou pública e que permanecem como entidades únicas, ligadas por uma tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicações.

Ao analisarmos as definições expostas acima, é possível destacar três novos aspectos em relação ao hardware, introduzidos em *cloud computing*. São eles:

- A ilusão de recurso computacional infinito disponível sob-demanda;
- A eliminação de um comprometimento antecipado por parte do usuário;
- A capacidade de alocar e pagar por recursos usando uma granularidade de horas.

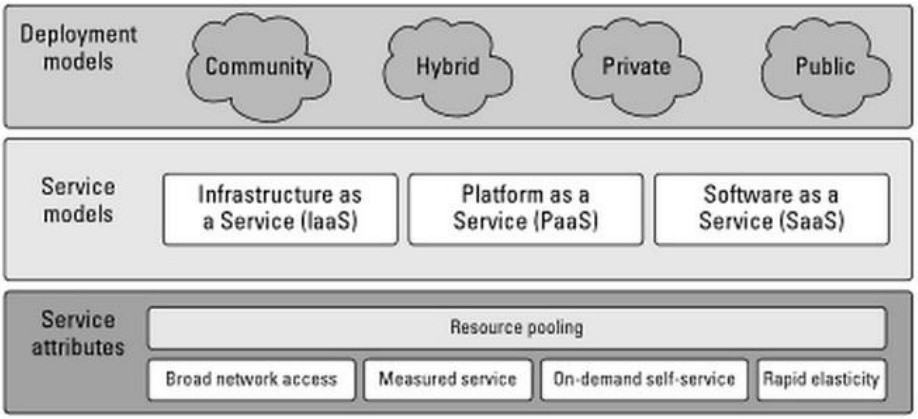
Esta elasticidade para obter e liberar recursos é um dos aspectos chaves da computação em nuvem, sendo uma das principais diferenças quando comparada com computação em grade.

2.9.6 O Modelo NIST

De acordo com (Sosinsky, 2011) o governo dos Estados Unidos é o maior consumidor de serviços de computador e, portanto, um dos maiores usuários das redes de computação em nuvem. O *National Institute of Standards and Technology* (NIST) possui um conjunto de definições de trabalho (http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc) que separa *cloud computing* em modelos de serviço e modelos de implementação.

Esses modelos e suas relações como características essenciais da computação em nuvem são mostrados na figura 12.

Figura 12 - *Cloud Computing* - NIST.



Fonte: *National Institute of Standards and Technology*

O modelo NIST originalmente não exigia uma nuvem para utilizar a virtualização de recursos reunidos, não fazer isso, absolutamente requer que uma nuvem suporte multi alocação nas suas definições primárias de *cloud computing*. Multi alocação é o compartilhamento de recursos entre dois ou mais clientes. A última versão das definições do NIST exige que as redes de computação em nuvem utilizem virtualização e suportem multi alocação.

2.10 Arquiteturas de Sistema de Banco de Dados

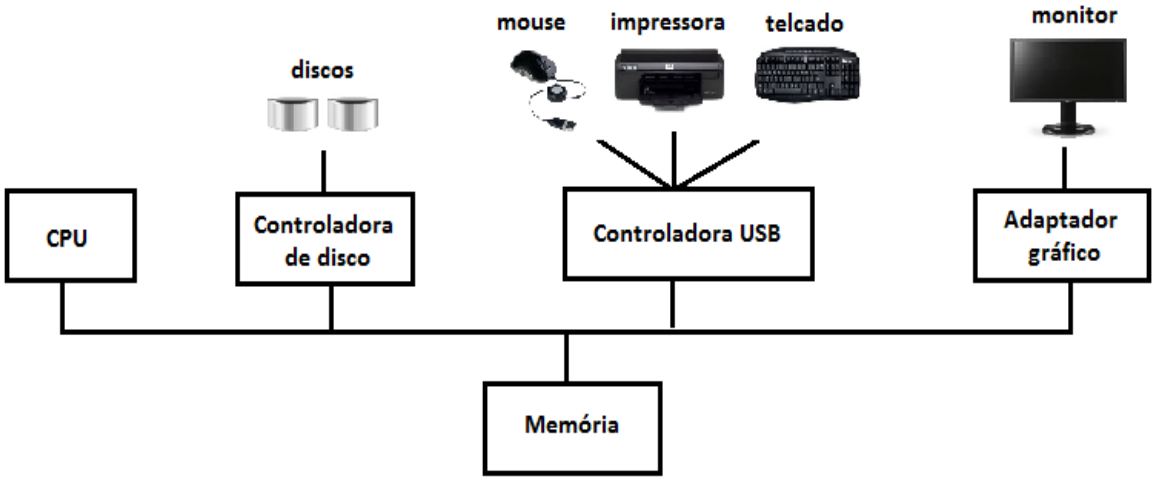
Segundo (Silberschatz, Korth, & Sudarshan, 2006), a arquitetura de um sistema de banco de dados é bastante influenciada pelo sistema de computador básico em que ela trabalha, em particular, por aspectos da arquitetura de computador como redes, paralelismo e distribuição.

As redes de computadores permitem que algumas tarefas sejam executadas em um sistema servidor e outras sejam executadas em sistemas clientes (bancos de dados cliente-servidor). O processamento paralelo permite que as atividades do sistema de banco de dados sejam agilizadas, permitindo respostas mais rápidas às transações, além de mais transações por segundo. A distribuição dos dados permite que os mesmos residam onde são gerados ou onde são mais necessários. Manter várias cópias do banco de dados em diferentes locais permite que as organizações continuem suas operações mesmo quando afetados por desastres naturais ou de natureza diversa.

2.10.1 Arquiteturas Centralizadas e Cliente-servidor

Os bancos de dados centralizados são aqueles que executam em um único sistema de computador e não interagem com outros sistemas de computador. Esses sistemas se espalham por uma faixa desde sistemas de bancos monousuários executando em computadores pessoais até sistemas de banco de dados de alto desempenho, executando em sistemas servidores de alto nível. Os sistemas cliente-servidor, por outro lado, possuem a funcionalidade dividida entre um sistema servidor e vários sistemas cliente.

Figura 13 - Sistema de computador centralizado.

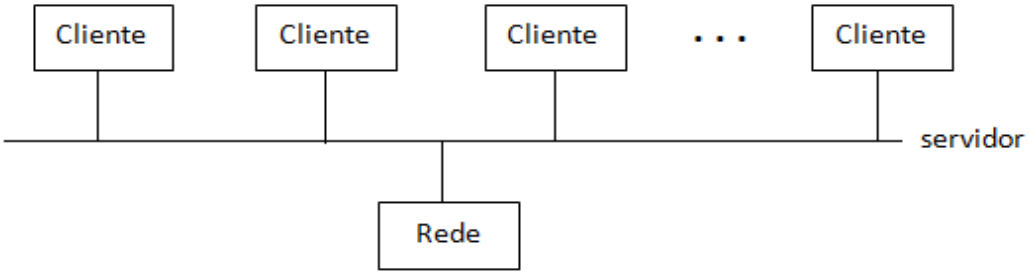


Fonte: (Silberschatz et al., 2006)

À medida que os computadores pessoais se tornaram mais rápidos, poderosos e baratos, houve um afastamento da arquitetura centralizada.

Os computadores pessoais suplantaram os terminais conectados aos sistemas centralizados. De modo correspondente, os computadores pessoais assumiram a funcionalidade da interface com o usuário, que antes era tratada diretamente pelos sistemas centralizados. Como resultado, os sistemas centralizados de hoje atuam como sistemas servidores, que satisfazem as solicitações feitas pelos sistemas clientes. A figura 14 mostra a estrutura geral de um sistema cliente-servidor.

Figura 14 - Estrutura geral de um sistema cliente-servidor.



Fonte: (Silberschatz et al., 2006)

A funcionalidade oferecida pelos sistemas de banco de dados pode ser dividida de forma geral em duas parte – o *fontend* e o *backend*. O *backend* controla as estruturas de acesso, avaliação e otimização de consultas, controle de concorrência e recuperação. O *frontend* consiste nas ferramentas como a interface com o usuário da SQL, interfaces de formulário, ferramentas de geração de relatório e de mineração e análise de dados (Silberschatz et al., 2006).

2.10.2 Sistemas Distribuídos

Em um sistema de banco de dados distribuído, o banco de dados é armazenado em vários computadores. Os computadores em um sistema distribuído se comunicam entre si através de diversos meios de comunicação, como redes de alta velocidade ou linhas telefônicas. Eles não compartilham memória principal ou discos. Os computadores em um sistema distribuído podem variar em tamanho e função, desde estações de trabalho até sistemas e mainframe (Silberschatz et al., 2006).

Os motivos para a criação de banco de dados distribuídos são os mais variados, incluindo o compartilhamento de dados, autonomia e disponibilidade.

2.10.3 Transações

Uma transação é uma unidade de execução do programa que acessa e possivelmente atualiza vários itens de dados. Normalmente uma transação é iniciada por um programa do usuário escrito em uma linguagem de manipulação de dados ou linguagem de programação de alto nível (por exemplo, Java, SQL, C++), em que é delimitada pelas instruções (ou chamadas de função) na forma *egin transaction* e *end transaction*. A transação consiste em todas as operações executadas entre o *egin transaction* e o *end transaction* (Silberschatz et al., 2006).

2.11 Trabalhos Correlatos

Recentemente, foram publicados poucos trabalhos acadêmicos de criação e aperfeiçoamento de *benchmarks*. Um dos principais motivos é a restrição imposta pela "cláusula DeWitt" (Moran, 2003), que proíbe a realização de testes em bancos de dados comerciais sem a prévia autorização do fabricante.

Em razão disso, vários *benchmarks* foram construídos para comparar o desempenho de SGBD de código aberto, tais como DBT-2 (OSDL, 2002), TPCC-UVA (Hernández, P. e Gonzalo, 2002) e OSDB (OSDB, 2001). Os dois primeiros utilizam uma carga de trabalho OLTP similar a do consagrado *benchmark* TPC-C, da TPC™ (*Transaction Processing Performance Council*) (TPC, 2001). O *benchmark* TPC-C é a principal referência no mundo quando se trata de desempenho de sistemas computacionais.

A maioria dos resultados comparando o desempenho de SGBD de código livre foi produzida através de uma ampla variedade de *benchmarks* e estes resultados são muitas vezes contraditórios e tendenciosos.

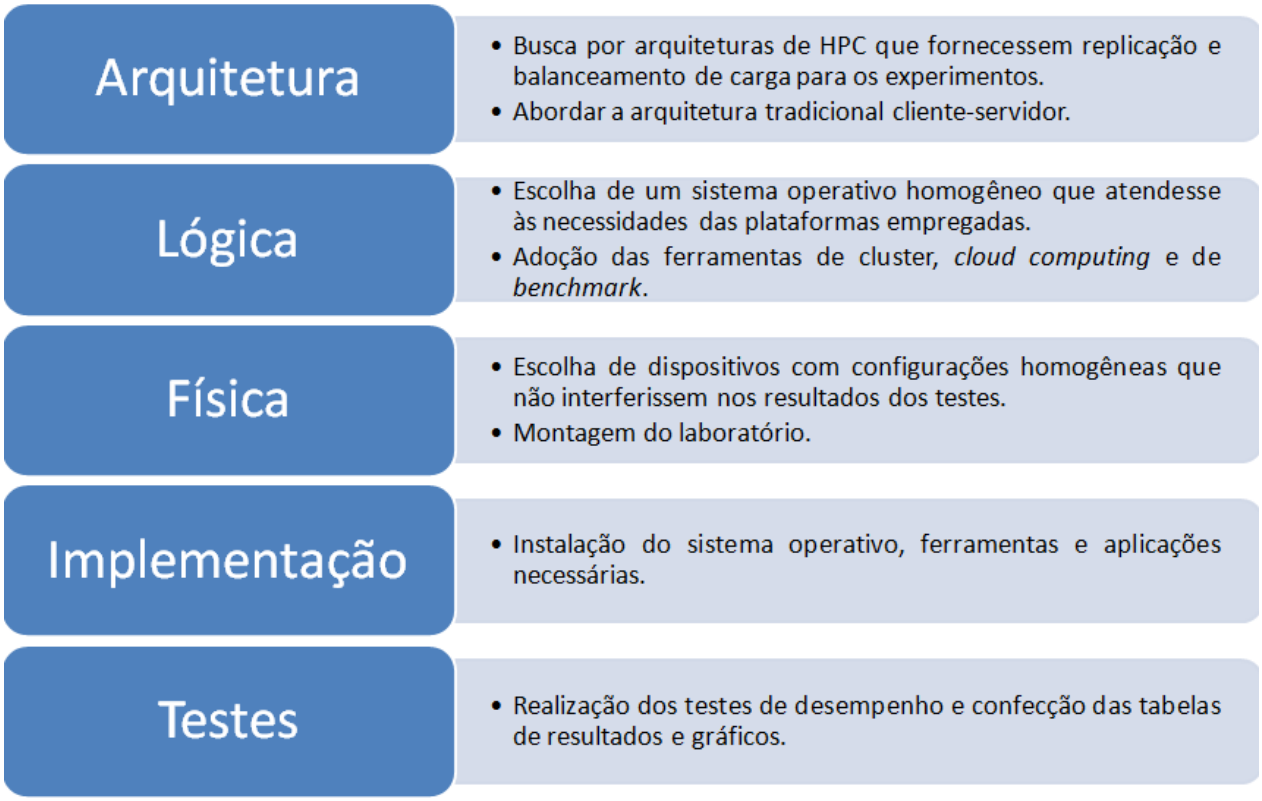
Os estudos apresentados em (GreatBridge, 2000) e (MySQL, 2005) comparam o desempenho dos SGBD PostgreSQL e MySQL, exibindo resultados contrários. Em (GreatBridge, 2000), o PostgreSQL mostra-se superior, porém por estar ligado a uma fornecedora de soluções para o PostgreSQL o resultado é bastante contestado (Widenius, 2000). O comportamento repete-se em favor do MySQL no trabalho (MySQL, 2005).

3 METODOLOGIA

Nesta seção, será apresentado, de forma simplificada e sucinta, o procedimento adotado após o levantamento do problema de pesquisa para que as hipóteses pudessem ser abordadas e as expectativas fossem atendidas. O cenário adotado para realização dos experimentos é o laboratório de testes da Universidade Federal do Pampa (Campus Bagé).

Visando a redução de custos do projeto e possibilitar futuras adaptações, toda a implementação lógica utiliza software de código aberto. A avaliação de desempenho foi obtida mediante utilização de uma ferramenta de benchmark específica para a aplicação de banco de dados adotada e consiste em definir a transferência de dados de um subsistema para outro em uma unidade de tempo específica. Todos os testes foram realizados cinco vezes e uma média foi elaborada entre os resultados obtidos, sendo que somente a média foi apresentada nas tabelas e gráficos.

Figura 15 - Quadro simplificado da metodologia.



Fonte: Próprio autor.

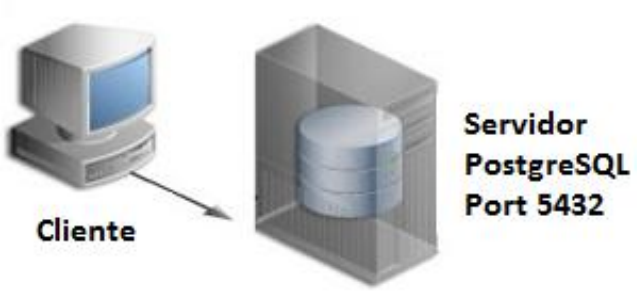
4 IMPLEMENTAÇÃO E TESTES

A organização funcional, as peculiaridades das arquiteturas empregadas e o equipamento utilizado serão apresentados na presente seção, bem como o modo de realização dos testes e o funcionamento da ferramenta de benchmark.

4.1 Arquitetura Cliente-servidor

A primeira arquitetura implementada (cliente-servidor) consta de um computador exercendo a função de cliente, o qual envia requisições para o servidor que está interligado por uma rede de computadores. O computador que está executando a aplicação de banco de dados (servidor), aceita as requisições, processa as mesmas e retorna o resultado para o cliente. A arquitetura é apresentada na figura 16.

Figura 16 - Arquitetura cliente-servidor.



Fonte: Próprio autor.

4.2 Arquitetura HPC Cluster utilizando Máquinas Físicas

Um *cluster*, ou aglomerado de computadores, é formado por um conjunto de computadores, que utiliza um tipo especial de sistema operacional classificado como sistema distribuído. Construído a partir de computadores convencionais (*personal computers*), os quais são ligados em rede e comunicam-se através do sistema, trabalhando como se fossem uma única máquina de grande porte. Foram utilizados nesta segunda implementação três computadores exercendo a função de nós (*backends*) e um computador controlando a estrutura (*frontend*); a ferramenta de cluster foi habilitada para replicação e balanceamento da carga de processamento dos nós.

As aplicações de banco de dados foram instaladas diretamente nos nós do cluster, totalizando três bancos de dados compondo o cluster computacional.

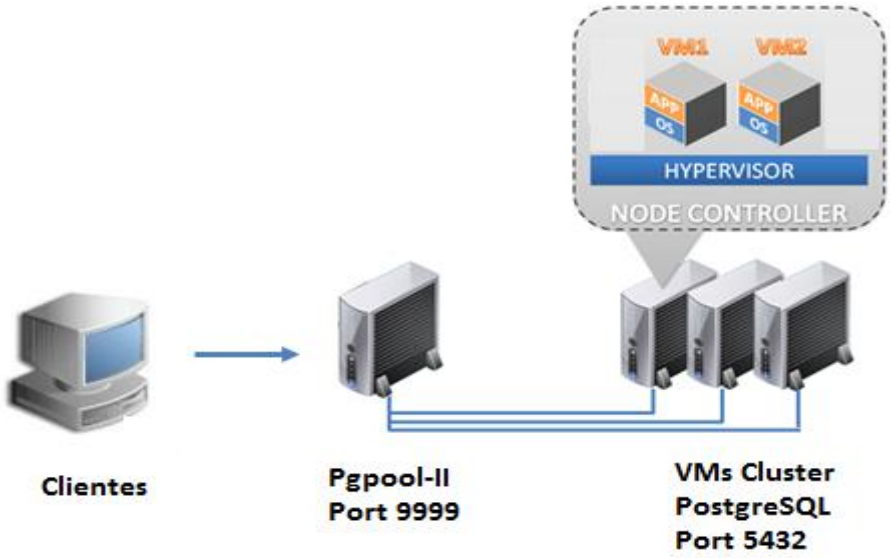
Figura 17 - Arquitetura HPC Cluster com máquinas físicas.

Fonte: Próprio autor.

4.3 Arquitetura HPC Cluster utilizando Máquinas Virtuais

A terceira arquitetura é deveras semelhante com a segunda, porém, ao invés de as aplicações de banco de dados estarem hospedadas nas máquinas físicas, estavam hospedadas nas máquinas virtuais executadas sobre a ferramenta de virtualização. Constava de duas máquinas virtuais sobre cada um dos três nós do cluster, totalizando seis máquinas virtuais. A ferramenta de cluster foi habilitada para replicação e balanceamento da carga de processamento dos nós.

Figura 18 - Arquitetura HPC Cluster com máquinas virtuais.

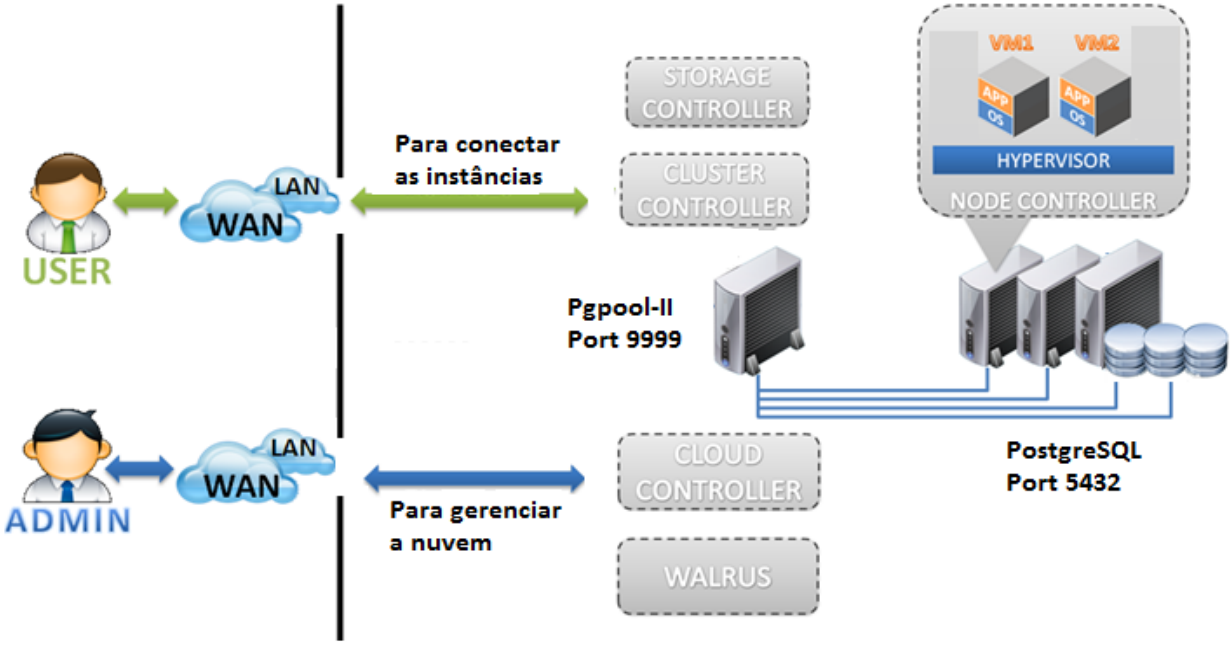


Fonte: Próprio autor.

4.4 Arquitetura Cluster sobre Cloud Computing

Na quarta e última arquitetura implementada, foi utilizado uma estrutura de nuvem computacional onde as aplicações de banco de dados que participam da composição do cluster rodam sobre as instâncias hospedadas nos nós. Foram instanciadas seis imagens nos três nós, o que resulta em um total de seis aplicações de banco de dados compondo o agrupamento. A ferramenta de cluster foi habilitada para replicação e balanceamento da carga de processamento dos nós. A arquitetura é apresentada na figura 19.

Figura 19 - Arquitetura cluster sobre *cloud computing*.



Fonte: Próprio autor.

4.5 Equipamentos

Na tabela a seguir, foram descritas as especificações de *hardware* e *software* empregados nos equipamentos dos testes. O equipamento possui homogeneidade quanto ao hardware, ou seja, é o mesmo equipamento utilizado em todas as estruturas avaliadas. O dispositivo intermediário de rede utilizado para interconexão dos equipamentos é um switch Cisco, e o padrão utilizado é o *Fast Ethernet* de 100 Mbps. As máquinas virtuais possuem a seguinte configuração: 1024 MB de memória RAM e o sistema operacional *Guest* é Linux – CentOS 5 (kernel 2.6.18-348.1.1.el5).

Tabela 2 - Configuração equipamentos utilizados.

Componente	Especificação
CPU	Intel Quad-Core Xeon E5620 @ 2.40GHz
Memória	RAM 8GB
Sistema Operacional	Linux – Ubuntu 10.04.4 LTS (Lucid Lynx)
Ferramenta Cluster	Pgpool-II
SGBDOR	PostgreSQL
Ferramenta virtualização	VirtualBox da Oracle

4.6 Ferramenta de Benchmark Pgbench

Pgbench é um programa para a execução de testes de benchmark sobre o PostgreSQL. O software roda a mesma seqüência de comandos SQL, possivelmente em várias sessões de banco de dados simultâneos, e então calcula a taxa média de transação (transações por segundo). Por definição, o Pgbench testa um cenário envolvendo cinco cláusulas a cada transação executada, SELECT, UPDATE e INSERT. O teste de transação requer tabelas específicas para sua execução.

O Pgbench deve ser invocado para criar e alimentar essas tabelas de dados, para a execução dos testes. A tabela 3, a seguir, apresenta o resumo de linhas inseridas nas tabelas do banco de dados pelo Pgbench, no total foram criadas quatro tabelas. O Pgbench é baseado em TPC-B, que é considerado, juto com o TPC-A uma carga de trabalho (workload) de OLTP simples (St, 1994). As métricas utilizadas em benchmark TPC-B são o *throughput* (rendimentos), medido em transações por segundo (TPS).

Tabela 3 - Tabela com o número de linhas criadas pelo Pgbench.

Tabela	Linhas
pgbench_branches	1
pgbench_tellers	10
pgbench_accounts	100000
pgbench_history	0

Fonte: (The PostgreSQL Global Development Group, 2013)

4.7 Testes

Utilizando a ferramenta de benchmark supracitada, os testes executados possuíam variações no número de transações e no número de clientes. Os testes de avaliação foram baseados no livro *Database Benchmarking* (Scalzo, Fernandez, Ault, Burleson, & Kline, 2007). Os dados relevantes para adoção dos valores referendados nos testes foram extraídos da tabela abaixo, os quais informam que banco de dados OLTP ocupados terão as seguintes transações por segundo:

Tabela 4 - TPS de banco de dados OLTP ocupados.

High transactions per second	eBay, Amazon	1,000 -10,000 TPS
Medium transactions per second	International web application	100 - 1,000 TPS
Low transactions per second	Small internal OLTP	10 - 100 TPS

Fonte: (Scalzo et al., 2007)

Com base nos valores acima, foram executados testes com variabilidade de transações (10000, 7500, 5000, 2500, 2000, 1500, 1000, 500 e 100) e também no número de clientes (32, 16, 8, 4 e 2) em todas as estruturas supracitadas. O número de transações processadas deve ser calculado multiplicando o número de transações pelo número de clientes. Os testes foram executados cinco vezes (em cada relação transação/cliente) e calculada a média das mesmas. Desta maneira, foram geradas informações que possibilitam uma amostragem da taxa média de transação dos dados em cada uma das infraestruturas supracitadas, obtendo assim as TPS (transações por segundo) permitindo analisar o desempenho. O comando executado via terminal consiste da seguinte sintaxe:

pgbench [opções] database_name

No campo opções foram utilizados os seguintes parâmetros:

-t = número de transações por cliente

-p = número da porta utilizada pelo Pgpool

-h = hostname ou IP onde o Pgpool está sendo executado

-c = número de clientes simulados

-S = determina o tipo de transação somente SELECT (SELECT Only)

Para fins de exemplificação dos testes realizados será utilizado o item 5.1, que apresenta 100 transações executadas por cada cliente e variabilidade quanto ao número de clientes. A sintaxe adotada foi a seguinte:

pgbench –t 100 –p 9999 – h pgpool –c (variabilidade) -S database_name

A figura abaixo mostra uma saída típica da ferramenta de benchmark pgbench.

Figura 20 - Saída típica da ferramenta Pgbench.

```
transaction type: TPC-B (sort of)
scaling factor: 10
query mode: simple
number of clients: 10
number of threads: 1
number of transactions per client: 1000
number of transactions actually processed: 10000/10000
tps = 85.184871 (including connections establishing)
tps = 85.296346 (excluding connections establishing)
```

Fonte: (The PostgreSQL Global Development Group, 2013).

5 RESULTADOS E DISCUSSÕES

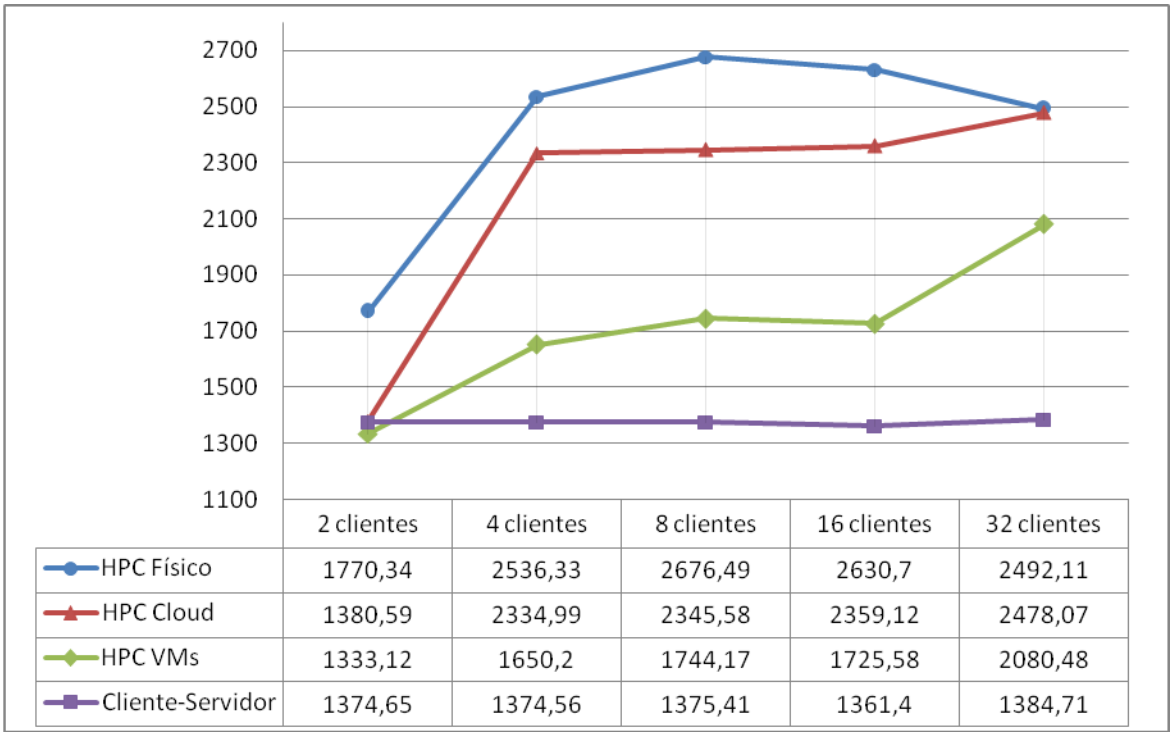
De acordo com o exposto no capítulo anterior onde foi demonstrada a elaboração dos testes realizados, mais especificamente nos itens 4.6 e 4.7, torna-se possível a visualização das métricas alcançadas nos experimentos. A representação nos gráficos ocorreu da seguinte maneira: no eixo Y (vertical) são apresentados os valores de TPS (transações por segundo) excluindo as conexões estabelecidas – ver figura 20 – e no eixo X (horizontal) são apresentadas as variações conforme o número de clientes.

Um ponto importante a ser ratificado é que a estrutura com clusters físicos apresentava três bases de dados, enquanto a estrutura de clusters utilizando máquinas virtuais apresentava duas máquinas virtuais em cada máquina física, totalizando seis bases de dados, a estrutura de cluster implementada sobre a nuvem computacional apresentava seis instâncias alocadas sobre a estrutura, cada uma delas contendo uma base de dados. As bases de dados têm por atribuição no presente estudo proporcionar o balanceamento das cargas de processamento e replicação das bases de dados. Conforme já ressaltado anteriormente, a idéia é verificar se aumentando o número de bases de dados (escalabilidade), traria por conseqüência um ganho de processamento computacional.

O resultado dos testes é apresentado no presente capítulo em forma de valores obtidos e gráficos, que possibilitam uma melhor visualização.

5.1 Teste realizado com 100 transações – SELECT Only

Figura 21 - Comparativo de desempenho para 100 transações.

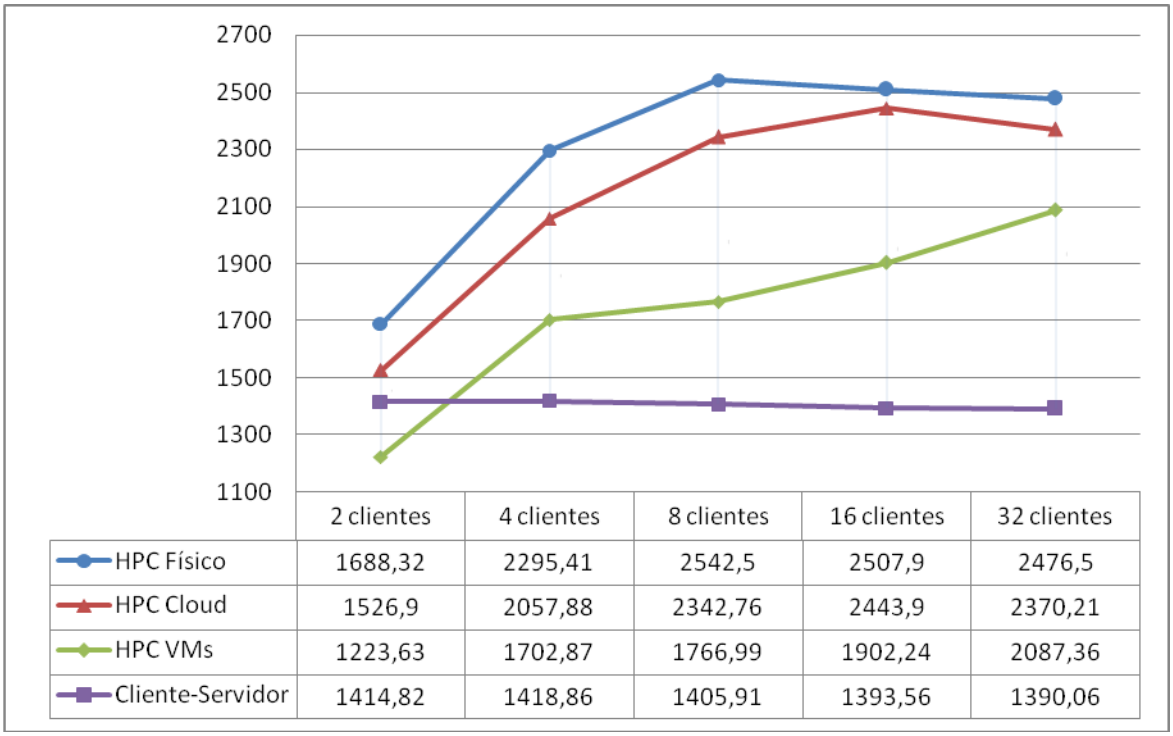


	2 clientes	4 clientes	8 clientes	16 clientes	32 clientes
HPC Físico	1770,34	2536,33	2676,49	2630,7	2492,11
HPC Cloud	1380,59	2334,99	2345,58	2359,12	2478,07
HPC VMs	1333,12	1650,2	1744,17	1725,58	2080,48
Cliente-Servidor	1374,65	1374,56	1375,41	1361,4	1384,71

É possível observar que utilizando dois clientes, todas as arquiteturas (com exceção do HPC físico) partiram de um ponto em comum. Posteriormente, com quatro clientes, já foi possível constatar que as arquiteturas HPC tiveram variações no desempenho (quase sempre em ascensão de valores) conforme variabilidade do número de clientes; nos 32 clientes as estruturas de HPC *cloud* e físico obtiveram valores aproximados.

5.2 Teste realizado com 500 transações – SELECT Only

Figura 22 - Comparativo de desempenho para 500 transações.

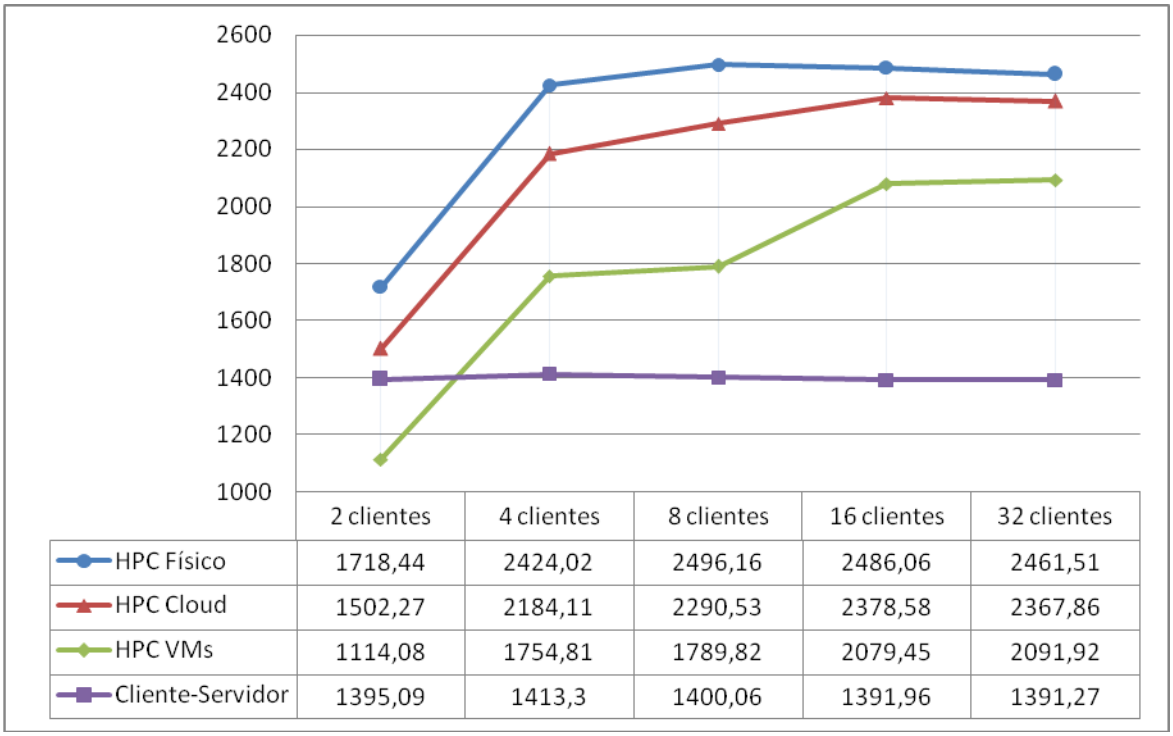


	2 clientes	4 clientes	8 clientes	16 clientes	32 clientes
HPC Físico	1688,32	2295,41	2542,5	2507,9	2476,5
HPC Cloud	1526,9	2057,88	2342,76	2443,9	2370,21
HPC VMs	1223,63	1702,87	1766,99	1902,24	2087,36
Cliente-Servidor	1414,82	1418,86	1405,91	1393,56	1390,06

Os valores iniciais foram divergentes; porém, novamente, as arquiteturas HPC foram progressivamente atingindo elevação no número de transações até o número de oito clientes. Após isso, a arquitetura HPC VMs permaneceu em ascensão, juntamente com a HPC *Cloud* que atingiu o ápice nos 16 clientes e começou a perder desempenho enquanto a HPC Físico, que vinha em descenso, começou novamente a ascender.

5.3 Teste realizado com 1000 transações – SELECT Only

Figura 23 - Comparativo de desempenho para 1000 transações.

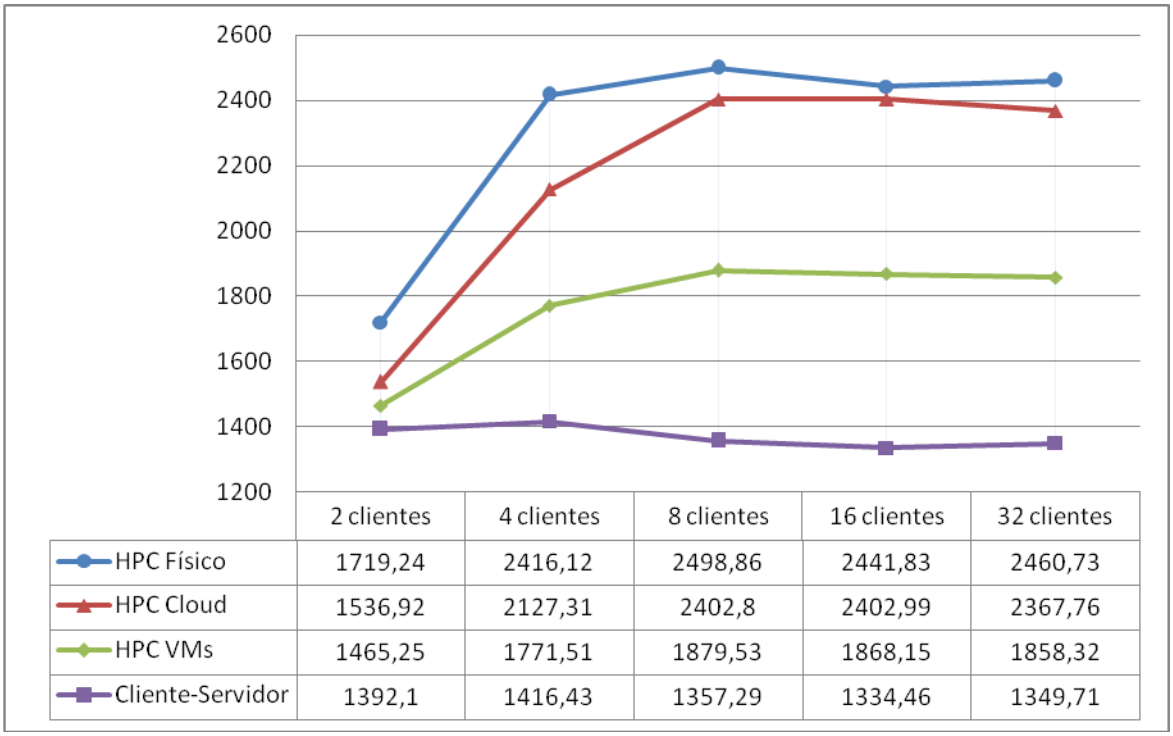


	2 clientes	4 clientes	8 clientes	16 clientes	32 clientes
HPC Físico	1718,44	2424,02	2496,16	2486,06	2461,51
HPC Cloud	1502,27	2184,11	2290,53	2378,58	2367,86
HPC VMs	1114,08	1754,81	1789,82	2079,45	2091,92
Cliente-Servidor	1395,09	1413,3	1400,06	1391,96	1391,27

Inicialmente, com dois clientes, a arquitetura HPC VMs ficou aquém das outras estruturas; porém, novamente, as estruturas HPC foram subindo (com variações) de desempenho conforme incremento do número de clientes.

5.4 Teste realizado com 1500 transações – SELECT Only

Figura 24 - Comparativo de desempenho para 1500 transações.

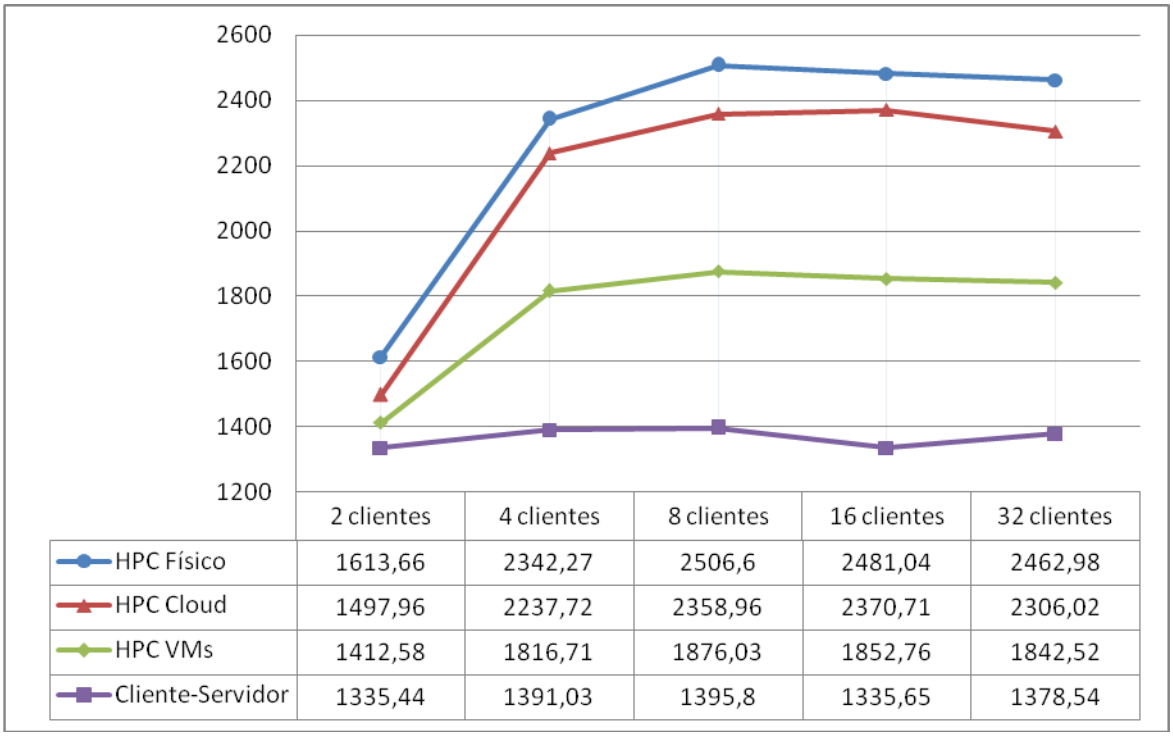


	2 clientes	4 clientes	8 clientes	16 clientes	32 clientes
HPC Físico	1719,24	2416,12	2498,86	2441,83	2460,73
HPC Cloud	1536,92	2127,31	2402,8	2402,99	2367,76
HPC VMs	1465,25	1771,51	1879,53	1868,15	1858,32
Cliente-Servidor	1392,1	1416,43	1357,29	1334,46	1349,71

Nos testes realizados com dois clientes, as arquiteturas obtiveram padrões semelhantes de transações, partindo de um mesmo patamar; com hegemonia da arquitetura HPC Físico, HPC *Cloud* e HPC VMs, respectivamente. Posteriormente as arquiteturas HPC se sobressaíram nos testes.

5.5 Teste realizado com 2000 transações – SELECT Only

Figura 25 - Comparativo de desempenho para 2000 transações.

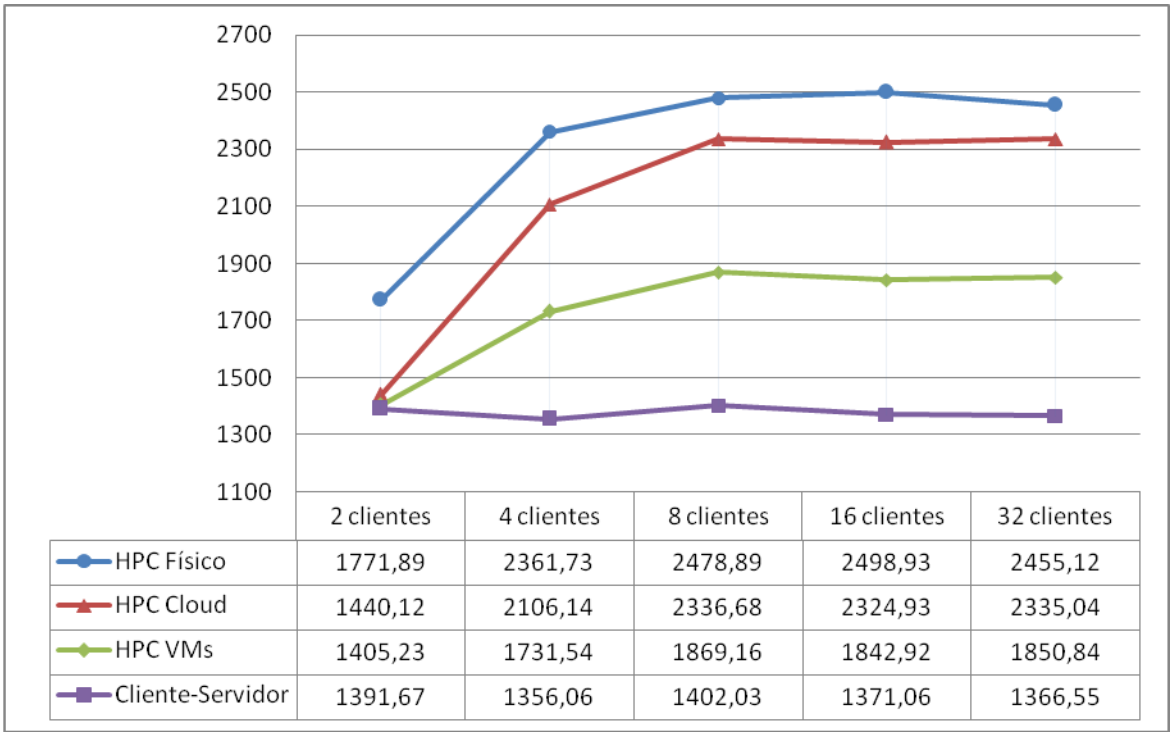


	2 clientes	4 clientes	8 clientes	16 clientes	32 clientes
HPC Físico	1613,66	2342,27	2506,6	2481,04	2462,98
HPC Cloud	1497,96	2237,72	2358,96	2370,71	2306,02
HPC VMs	1412,58	1816,71	1876,03	1852,76	1842,52
Cliente-Servidor	1335,44	1391,03	1395,8	1335,65	1378,54

Nos testes realizados com dois clientes, as arquiteturas obtiveram padrões semelhantes de transações, partindo de um mesmo patamar; com hegemonia da arquitetura HPC Físico, HPC *Cloud* e HPC VMs, respectivamente. Posteriormente as arquiteturas HPC se sobressaíram nos testes.

5.6 Teste realizado com 2500 transações – SELECT Only

Figura 26 - Comparativo de desempenho para 2500 transações.

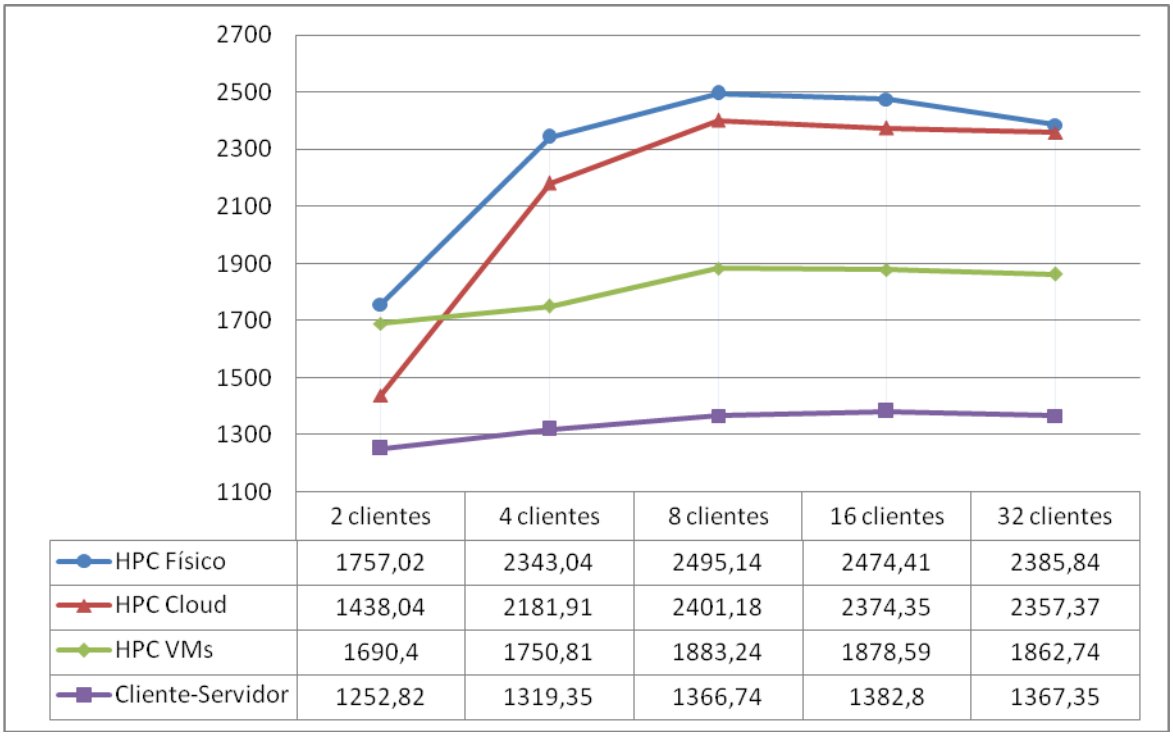


	2 clientes	4 clientes	8 clientes	16 clientes	32 clientes
HPC Físico	1771,89	2361,73	2478,89	2498,93	2455,12
HPC Cloud	1440,12	2106,14	2336,68	2324,93	2335,04
HPC VMs	1405,23	1731,54	1869,16	1842,92	1850,84
Cliente-Servidor	1391,67	1356,06	1402,03	1371,06	1366,55

Nos testes realizados com dois clientes, as arquiteturas obtiveram padrões semelhantes de transações, partindo de um mesmo patamar; com hegemonia da arquitetura HPC Físico, HPC *Cloud* e HPC VMs, respectivamente. Posteriormente as arquiteturas HPC se sobressaíram nos testes.

5.7 Teste realizado com 5000 transações – SELECT Only

Figura 27 - Comparativo de desempenho para 5000 transações.

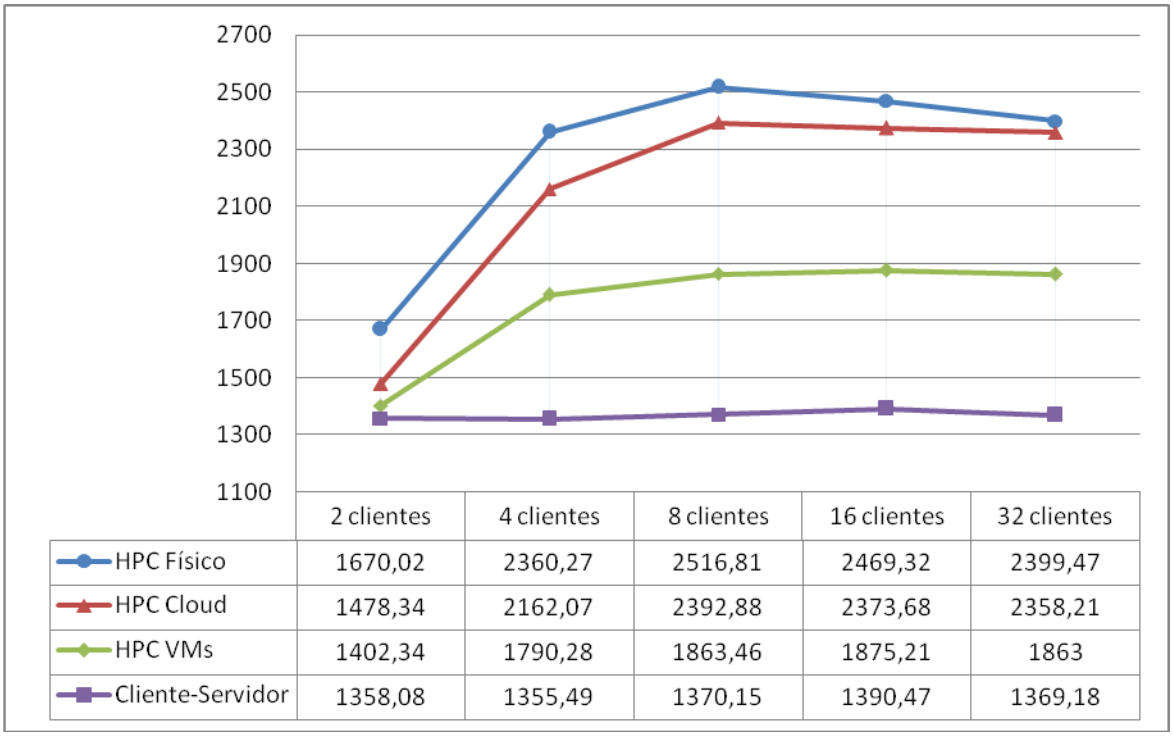


	2 clientes	4 clientes	8 clientes	16 clientes	32 clientes
HPC Físico	1757,02	2343,04	2495,14	2474,41	2385,84
HPC Cloud	1438,04	2181,91	2401,18	2374,35	2357,37
HPC VMs	1690,4	1750,81	1883,24	1878,59	1862,74
Cliente-Servidor	1252,82	1319,35	1366,74	1382,8	1367,35

Tornou-se notável que a partir dos 32 clientes as arquiteturas HPC Físico e HPC *Cloud* obtiveram uma congruência nos valores das transações. E nos testes com dois clientes, pela primeira vez a arquitetura HPC VMs teve supremacia sobre a HPC *Cloud*.

5.8 Teste realizado com 7500 transações – SELECT Only

Figura 28 - Comparativo de desempenho para 7500 transações.

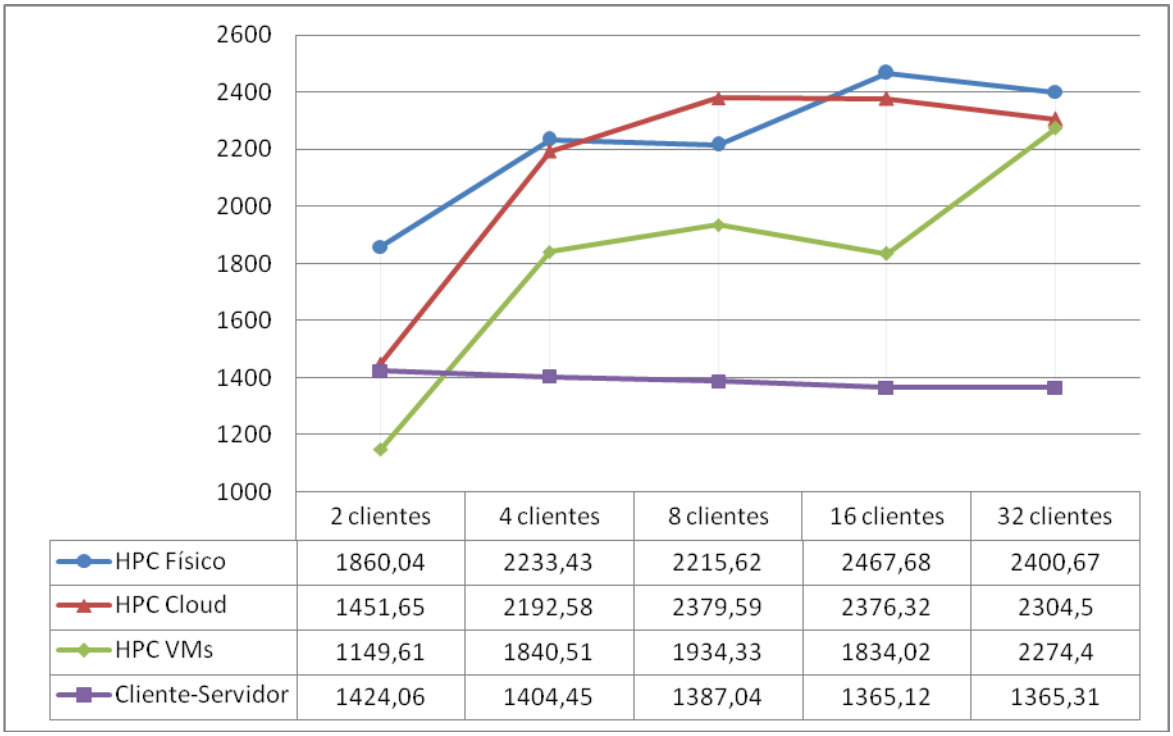


	2 clientes	4 clientes	8 clientes	16 clientes	32 clientes
HPC Físico	1670,02	2360,27	2516,81	2469,32	2399,47
HPC Cloud	1478,34	2162,07	2392,88	2373,68	2358,21
HPC VMs	1402,34	1790,28	1863,46	1875,21	1863
Cliente-Servidor	1358,08	1355,49	1370,15	1390,47	1369,18

Nos testes realizados com dois clientes, as arquiteturas obtiveram padrões semelhantes de transações, partindo de um mesmo patamar; com hegemonia da arquitetura HPC Físico, HPC *Cloud* e HPC VMs, respectivamente. Posteriormente as arquiteturas HPC se sobressaíram nos testes e de forma análoga ocorrida em testes com diferentes transações, houve uma congruência do HPC físico e *cloud* nos experimentos com 32 clientes.

5.9 Teste realizado com 10000 transações – SELECT Only

Figura 29 - Comparativo de desempenho para 10000 transações.



	2 clientes	4 clientes	8 clientes	16 clientes	32 clientes
HPC Físico	1860,04	2233,43	2215,62	2467,68	2400,67
HPC Cloud	1451,65	2192,58	2379,59	2376,32	2304,5
HPC VMs	1149,61	1840,51	1934,33	1834,02	2274,4
Cliente-Servidor	1424,06	1404,45	1387,04	1365,12	1365,31

Nesses testes aconteceram as maiores variações nos gráficos apresentados. E foi a segunda ocasião, mais especificamente com 8 clientes, que a arquitetura HPC Cloud ultrapassou as transações obtidas pelo HPC Físico.

As hipóteses levantadas anteriormente, que levaram ao desenvolvimento do presente estudo, no qual se acreditava que aumentando o número de bases de dados acarretaria um desempenho computacional superior foi rechaçada, tendo em vista que o cluster de máquinas físicas com apenas três bases de dados apresentou um desempenho preponderantemente superior na maioria dos testes realizados sobre as demais estruturas. A escalabilidade, de um modo mais consistente, não pode ser amplamente testada em virtude do interstício temporal relativamente curto para realização dos experimentos, mas que poderão ser abordados com maior ênfase em experimentos futuros.

6 CONCLUSÃO

Os resultados mostrados no capítulo anterior exprimem, de forma axiomática, que as estruturas de alto desempenho (HPC) apresentaram, predominantemente, uma superioridade considerável de performance se comparadas às avaliações obtidas na estrutura cliente-servidor. A arquitetura cliente servidor permaneceu, quase que constantemente, na mesma faixa de transações por segundo em todos os testes realizados.

Um aspecto importante a ser observado é que a arquitetura cliente-servidor possui uma limitação física em relação a equipamentos, o que mantém o número de TPS em valores estáveis, independente da quantidade de clientes utilizados. Outra consideração que deve ser tomada é que as arquiteturas HPC *Cloud* e HPC VMs possuíam seis aplicações de banco de dados, enquanto que o HPC Físico possuía somente três aplicações.

Tomando por base que as consultas requisitadas utilizam o balanceamento de carga, e sendo este realizado utilizando todas as aplicações disponíveis, pressupõe-se que, quanto maior o número de aplicações disponíveis, maior seriam os valores de transações por segundo obtidas. Porém, não foi essa a constatação obtida durante os experimentos. Na quase totalidade absoluta das avaliações, o HPC Físico, contando somente com três bases de dados utilizadas no balanceamento de carga das requisições, foi superior às demais arquiteturas implementadas. De acordo com o referencial teórico, os sistemas distribuídos proporcionam inúmeras vantagens em relação aos sistemas centralizados, como por exemplo: economia (microprocessadores oferecem custo/benefício maior do que mainframes), escalabilidade, compartilhamento de recursos, transparência, tolerância a falhas (proporcionando alta disponibilidade), entre outros. E também desvantagens, como o desenvolvimento de software muito complexo, possíveis gargalos por dependência de rede de comunicação e falta de segurança intrínseca.

Em face dos experimentos realizados, surgiram indagações sobre o overhead proporcionado pela conexão de rede. Abordagens futuras consistiriam em determinar qual a quantidade de máquinas virtuais condizentes por nó do cluster de modo que a interface de rede não se tornasse um gargalo nas comunicações, bem como do número de instâncias adequado na estrutura de cluster sobre a nuvem computacional. Essas abordagens forneceriam informações concisas para a adoção de um escopo adequado em estruturas de pequeno e médio porte que visem aplicações de banco de dados.

REFERÊNCIAS

Aboulnaga, A., Salem, K., Soror, A. A., Minhas, U. F., Kokosielis, P., & Kamath, S. (2009). Deploying Database Appliances in the Cloud. *IEEE Data Eng. Bull.*, *32*(1), 13–20. Retrieved from http://dblp.uni-trier.de/db/journals/debu/debu32.html#AboulnagaSSMKK09

Andrew S. Tanembaum. (2003). *Modern Operating Systems*. (R. Trimer, Ed.) (2nd ed., p. 685). São Paulo: Pearson Prentice Hall.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., et al. (2009). *Above the Clouds: A Berkeley View of Cloud Computing*.

Bevilacqua, A. (1999). A Dynamic Load Balancing Method On A Heterogeneous Cluster Of Workstations 1 Introduction 2 Review 3 Image reconstruction problem and sequential. *Practice*.

Bookman, C. (2002). *Linux Clustering: Building and Maintaining Linux Clusters*. (D. Dwyer, Ed.) (1st ed., p. 300). Indianapolis: New Riders Publishing.

Brewer, E. (1997). Clustering: Multiply and Conquer. *Data Communications*.

Bugnion, E., Devine, S., Govil, K., & Rosenblum, M. (1997). Disco: running commodity operating systems on scalable multiprocessors. *ACM Trans. Comput. Syst.*, *15*(4), 412–447. doi:10.1145/265924.265930

Buyya, R. (1999). *High Performance Cluster Computing: Architectures and systems*. Prentice Hall PTR. Retrieved from http://books.google.com.br/books?id=wrZQAAAAMAAJ

Buyya, Rajkumar, Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, *25*(6), 599–616. doi:10.1016/j.future.2008.12.001

Ciurana, E. (2009). Developing with Google App Engine. Retrieved from http://www.mendeley.com/research/developing-google-app-engine/

Diekmann, R., & Monien, B. (1997). Load Balancing Strategies for Distributed Memory Machines, (Computer Science Technical Report Series “SFB”), 1–37.

Flynn, M. J. (1972). Some computer organizations and their effectiveness. *IEEE Trans. Comput.*, *21*(9), 948–960. doi:10.1109/TC.1972.5009071

GreatBridge. (2000). GreatBridge Performance Comparison. Retrieved from http://www.angelfire.com/country/aldev0/pgsql/GreatBridge.html

Hernández, P. e Gonzalo, J. (2002). *Implementación en C del benchmark de transacciones distribuidas TPC-C, Bs.C Thesis*. Escuela Universitaria Politécnica de Valladolid, Universidad de Valladolid, Spain.

Hoelzle, U., & Barroso, L. A. (2009). The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. Retrieved from http://dl.acm.org/citation.cfm?id=1643608

Hwang, K., Jin, H., Chow, E., Wang, C.-L., & Xu, Z. (1999). Designing SSI clusters with hierarchical checkpointing and single I/O space. *Concurrency, IEEE*, *7*(1), 60–69.

ISO/IEC. (1995). CD10746 Information Technology: Open Distributed Processing - Reference Model.

Jacobs, D., Aulbach, S., & München, T. U. (2007). Ruminations on Multi-Tenant Databases. *BTW PROCEEDINGS, VOLUME 103 OF LNI*, 514 – 521. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.115.3258

Kapp, C. (2000). Managing cluster computers. *Dr. Dobb's Journal*.

Kumar, P. B. S. (2010). Introduction to parallel Computing. *Physics*, (November).

Liu, S., Liang, Y., & Brooks, M. (2007). Eucalyptus. *Proceedings of the 2007 conference of the center for advanced studies on Collaborative research - CASCON '07* (p. 1). New York, New York, USA: ACM Press. doi:10.1145/1321211.1321213

Marinos, A., & Briscoe, G. (2009). Community Cloud Computing. *Computing*, *5931*(December). Retrieved from http://www.mendeley.com/research/community-cloud-computing/

Mell, P., & Grance, T. (2009). The NIST Definition of Cloud Computing (Draft) Recommendations of the National Institute of Standards and Technology. *Nist Special Publication*, *145*(6), 7. doi:10.1136/emj.2010.096966

Moran, B. (2003). The Devil's in the DeWitt Clause {@ONLINE}. Retrieved from http://sqlmag.com/sql-server/devils-dewitt-clause

MySQL. (2005). Performance Comparison by MySQL Group. {@ONLINE}. Retrieved from http://sunsite.mff.cuni.cz/MIRRORS/ftp.mysql.com/information/benchmarks.html

OSDB. (2001). The Open Source Database Benchmark. {@ONLINE}. Retrieved from http://osdb.sourceforge.net/

OSDL, O. S. D. L. (2002). Database Test 2 {@ONLINE}. Retrieved from http://www.osdl.org/

Pitanga, M. (2008). *Construindo Supercomputadores com Linux*. (S. M. de Oliveira, Ed.) (3rd ed.). Rio de Janeiro: Brasport.

Popek, G. J., & Goldberg, R. P. (1974). Formal requirements for virtualizable third generation architectures. *Commun. ACM*, *17*(7), 412–421. doi:10.1145/361011.361073

Robinson, D. (2008). Amazon Web Services Made Simple: Learn how Amazon EC2, S3, SimpleDB and SQS Web Services enables you to reach business goals faster. *Network*. Retrieved from http://www.mendeley.com/research/amazon-web-services-made-simple/

Salesforce. (2010). Retrieved from http://www.salesforce.com/

Scalzo, B., Fernandez, C., Ault, M., Burleson, D. K., & Kline, K. (2007). *Database Benchmarking: Practical Methods for Oracle & SQL Server*. Rampant TechPress.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2006). *Sistema de banco de dados* (5ª edição., p. 781). Elsevier.

Sosinsky, B. (2011). Cloud Computing Bible. Retrieved from http://dl.acm.org/citation.cfm?id=1983498

St, N. F. (1994). TPC BENCHMARK ™ B Transaction Processing Performance Council (TPC), (June).

Stallings, W. (2010). *Arquitetura e organização de computadores*. Pearson. Retrieved from http://books.google.com.br/books?id=kTKeQwAACAAJ

Tanenbaum, A S. (1995). *Sistemas operacionais modernos*. Prentice-Hall. Retrieved from http://books.google.com.br/books?id=gLkkPwAACAAJ

Tanenbaum, Andrew S, & Steen, M. van. (2006). *Distributed Systems: Principles and Paradigms (2nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Taurion, C. (2009). *Cloud Computing - Computação em Nuvem: Transformando o mundo da Tecnologia da Informação*. Rio de Janeiro: BRASPORT. Retrieved from http://books.google.com.br/books?id=mvir2X-A2mcC

The PostgreSQL Global Development Group. (2013). The PostgreSQL Global Development Group. Retrieved from http://www.postgresql.org/

TPC. (2001). Transaction Processing Performance Council. {@ONLINE}. Retrieved from http://www.tpc.org/

Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2009). A break in the clouds. *ACM SIGCOMM Computer Communication Review*, *39*(1), 50. doi:10.1145/1496091.1496100

Vecchiola, C., Chu, X., & Buyya, R. (2009). Aneka: A Software Platform for .NET-based Cloud Computing. *Computing*, 267–295.

Widenius, M. (2000). MySQL Developer Contests PostgreSQL Benchmarks. Retrieved from http://www.devshed.com/c/a/BrainDump/MySQLDeveloper-