UNISINOS

Programa Interdisciplinar de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Paulo Henrique Cazarotto

GTTracker: Serviço móvel para identificação de oportunidades comerciais

São Leopoldo, 2013

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

PAULO HENRIQUE CAZAROTTO

GTTRACKER: SERVIÇO MÓVEL PARA IDENTIFICAÇÃO DE OPORTUNIDADES COMERCIAIS

SÃO LEOPOLDO
2013

Paulo Henrique Cazarotto

GTTRACKER: SERVIÇO MÓVEL PARA IDENTIFICAÇÃO DE OPORTUNIDADES COMERCIAIS

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

Orientador:
Prof. Dr. Cristiano A. Costa

Co-orientador:
Prof. Dr. Rodrigo R. Righi

São Leopoldo
2013

C386g Cazarotto, Paulo Henrique.
Gttracker : serviço móvel para identificação de oportunidades comerciais / Paulo Henrique Cazarotto. – 2013.
95 f. : il. ; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2013.
"Orientador: Prof. Dr. Cristiano A. Costa ; co-orientador: Prof. Dr. Rodrigo R. Righi."

1. Comércio móvel. 2. Comércio eletrônico. 3. Matching. 4. Computação em nuvem. 5. Sistemas distribuídos. I. Título.

CDU 004

Dados Internacionais de Catalogação na Publicação (CIP)
(Bibliotecário: Flávio Nunes – CRB 10/1298)

RESUMO

A crescente adoção de meios eletrônicos para comércio de produtos e serviços oferece a empresas um canal entre consumidores e vendedores. Este canal de comunicação para vendas é mantido tradicionalmente através de investimentos em plataformas de e-commerce. No cenário atual, cada empresa adquire ou desenvolve sua própria plataforma de comércio eletrônico. Este método possui um custo elevado e inviabiliza a entrada de pequenas e micro empresas no canal de comunicação. Neste âmbito, este trabalho propõe a disponibilização de um serviço compartilhado para publicação de ofertas e demandas que possibilita a identificação de oportunidades de negócio baseadas na localização dos usuários. Diferente de trabalhos relacionados estudados, o modelo é projetado para tirar proveito das funcionalidades disponibilizadas pela computação em nuvem, fácil provisionamento e baixo custo. Para tal, o modelo realiza o *matching* entre oferta e demanda utilizando estratégias de processamento em uma organização hierárquica, baseada em regiões. Através da implementação de um protótipo baseado neste modelo de processamento, chamado de GTTracker, foi possível compreender como um serviço projetado para atuar sobre recursos virtualizados escalonáveis pode se comportar. É foco da avaliação a apresentação do perfil de consumo dos recursos de infraestrutura durante diferentes cenários de acesso ao serviço. Os experimentos demonstraram que alguns mecanismos utilizados no serviço podem reduzir o consumo de recursos da infraestrutura e aumentar o desempenho do serviço proposto. A avaliação demonstra que é possível a implantação do serviço de forma progressiva, sobre uma plataforma de baixo custo, oferecendo um canal de comunicação acessível para quem deseja comprar e vender em qualquer lugar e a qualquer hora, utilizando um dispositivo móvel.

Palavras-chave: Comércio móvel. Comércio eletrônico. *Matching*. Computação em nuvem. Sistemas distribuídos.

ABSTRACT

The increased adoption of electronic commerce for product and service allows companies to use a new channel between consumers and vendors. This communication channel for sales is traditionally held through investments in e-commerce platforms. In the current scenario, each company acquires or develops its own e-commerce platform. This method has a high cost and slows the entry of small companies in the communication channel. In this context, this work proposes a shared service for supply and demand publishing, which allows the identification of business opportunities based on user's location. Unlike related works studied, the model is designed to take advantage of the features offered by cloud computing, such as easy provisioning and low-cost. Furthermore, the model performs the matching between supply and demand using processing strategies in a hierarchical organization, based on regions. We implemented a prototype based on this model, called GTTracker, to understand how the designed service performs on scalable virtualized resources. The focus of this assessment is profiling the infrastructure resources consumption, employing scenarios with different access patterns. The experiments demonstrated which mechanisms, used in the service, can reduce the resources consumption. Moreover, the evaluation shows an increase in the performance of GTTracker. The evaluation shows that managers can deploy the application gradually, using a low-cost platform. Additionally, GTTracker provides a channel of communication accessible to those users looking to buy or sell products wherever they are and at anytime, using a mobile device.

Keywords: Mobile Commerce. Electronic Commerce. Matching. Cloud computing. Distributed systems.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	*Application Programming Interface*
BTREE	*Binary Tree*
DHT	*Distributed Hash Table*
DIEESE	Departamento Intersindical de Estatística e Estudos Socioeconômicos
GPS	*Global Positioning System*
GTTracker	*General Trade Tracker*
IaaS	*Infrastructure as a Service*
JSON	*JavaScript Object Notation*
MIX	*Metadata based Integration model for data X-change*
MPME	Micro, pequenas e médias empresas
NIST	*National Institute of Standards and Technology*
OVF	*Open Virtualization Format*
P2P	*Peer to Peer*
PaaS	*Plataform as a Service*
RDF	*Resource Description Framework*
RFID	*Radio-Frequency Identification*
SaaS	*Software as a Service*
SOA	*Service-Oriented Architecture*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Micro, pequenas e médias empresas (MPME) desempenham um papel importante no mercado brasileiro. Conforme o anuário do trabalho na Micro e Pequena Empresa 2010/2011[1], publicada pelo Sebrae (Serviço Brasileiro de Apoio às Micro e Pequenas Empresas) e DIEESE (Departamento Intersindical de Estatística e Estudos Socioeconômicos), em 2010 pouco mais de seis milhões de estabelecimentos eram responsáveis por 14,7 milhões de empregos formais privados não agrícolas. Isto representa um pouco mais da metade dos empregos formais de estabelecimentos privados do país. Neste mesmo levantamento sobre o setor econômico secundário e terciário, as empresas de comércio e serviços em 2010 representavam 68,4% enquanto a indústria e construção representavam 31,6% na distribuição por setor de atividade econômica no Brasil. Estes índices demonstram que a oferta de serviços e bens de consumo compõem uma parte significativa do mercado.

Para que os empreendedores e comerciantes possam identificar oportunidades de negócio em uma determinada região é necessário obter uma visão geral de mercado. Esta visão relativa a oferta e procura de seus produtos e serviços apoia a elaboração de um negócio e reduz os riscos de oferecer algo que não é de interesse dos consumidores. Grandes empresas dispõem de recursos financeiros para obtenção deste panorama, enquanto isso empresas pequenas e médias precisam admitir um risco grande ou investir um tempo ou valor significativo de capital para obtenção de um panorama adequado.

Um investimento comum em estabelecimentos de pequeno e médio porte é a adoção de sistemas informatizados. Este investimento no setor de tecnologia da informação normalmente ocorre quando há uma obrigatoriedade fiscal ou quando há uma visível oportunidade de redução de custos operacionais e administrativos através de automação. A medida que a empresa cresce estes investimentos tendem a aumentar não apenas em infraestrutura básica como equipamentos, softwares de controle, profissionais e automação, mas também em processos analíticos, mobilidade, computação em nuvem e sistemas de relacionamento com clientes.

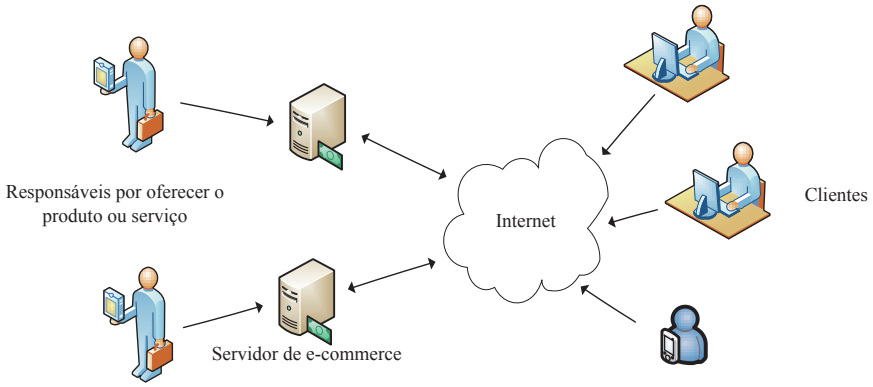
Na busca de um investimento com boa relação custo/benefício, muitas empresas utilizam soluções baseadas em virtualização ou utilizam serviços localizados em nuvem computacional, como um software oferecido como serviço. Este comportamento favorece o compartilhamento de recursos de qualidade com um custo baixo, normalmente mantido por uma equipe especializada em um ambiente fora do espaço físico da empresa. Evitando gastos tanto em equipamentos como em treinamento de profissionais para manutenção deste serviço. Independentemente do setor ou especialidade da empresa o uso da tecnologia passa a ser parte do negócio e este pode ser prejudicado se a tecnologia não atender as suas necessidades comerciais e organizacionais.

Entre as necessidades destas empresas que já precisam divulgar seus produtos e serviços através de mídias tradicionais como televisão, rádio e jornal, está a possibilidade de oferecer seus produtos a possíveis consumidores que utilizam a Internet em seu dia a dia. Esta

[1] http://www.biblioteca.sebrae.com.br/

oferta normalmente é feita através de ferramentas de e-commerce (comércio eletrônico) onde o ofertante apresenta a usuários de Internet seus produtos e serviços. Cabendo a estes usuários localizarem a informação através de uma plataforma de busca tradicional (como Google[2], com ou sem indicações ou propagandas pagas) ou uma plataforma especializada proprietária (como o Mercado Livre[3]). A Figura 1 apresenta o cenário tradicional de oferta de produtos através da Internet. Os comerciantes ou empresários adquirem um serviço ou um sistema exclusivo no qual disponibilizam as informações de seus produtos. No outro extremo há os usuários, que por sua vez acessam através da Internet estes sistemas em busca de produtos ou serviços que atendam as respectivas necessidades.

Figura 1: Cenário de comércio eletrônico tradicional



Fonte: Elaborado pelo autor

Serviços tradicionais como páginas de e-commerce na Internet idealmente precisam estar de acordo com as demais atividades relacionadas ao negócio para serem eficientes. O que envolve planejamento, organização de processo da empresa e compatibilidade adequada entre as tecnologias já existentes dentro da empresa (SHAH; DAWSON, 2000). Este serviço além de promover a disponibilização de produtos também pode oferecer estatísticas interessantes à quem oferece seus produtos. Ferramentas como o Google Analytics[4] permitem que os vendedores analisem o número de acessos a informação, na qual dá a visão de procura e a taxa de conversão, o que determina a eficiência de venda. Não só oferecendo recursos aos vendedores, um sistema público na Internet disponibiliza informações para os concorrentes que podem observar as políticas de venda de outras empresas para melhorar o próprio negócio, tanto em preços quanto em estratégias de venda. Para Hu (2009) um mecanismo que levante o panorama do mercado rapidamente e que mantenha uma troca de informação mais constante entre os envolvidos pode guiar uma empresa a obter melhores condições de competitividade perante outras, evitando o ciclo tradicional de consulta aos consumidores que costuma ser caro

[2]http://www.google.com
[3]http://www.mercadolivre.com
[4]http://www.google.com/analytics

e demorado.

Outro processo importante de venda é identificar como os usuários chegam até as informações do seu produto ou serviço. Este processo auxilia o vendedor a identificar o meio mais eficiente de acesso por parte do comprador. Este meio pode mudar de acordo com a evolução dos dispositivos e surgimento de novas tecnologias. Com a adoção de novos dispositivos móveis para acesso a Internet como *smartphones* e *tablets* é importante que os vendedores estejam preparados para competir através destes outros meios de comunicação. O mesmo se aplica às tecnologias relacionadas ao serviço de comércio eletrônico. Além dos aspectos funcionais, também há algumas questões técnicas importantes a serem tratadas quanto a serviços de comércio móvel (ZHAO; FENG, 2009) (KASSEL, 2008).

Dispositivos móveis possuem recursos computacionais limitados e a própria infraestrutura de comunicação pode oferecer riscos de segurança além de um alto custo de manutenção. Sabendo desta baixa capacidade computacional é normal que estes dispositivos utilizem serviços disponíveis em outros locais da rede para melhorar o desempenho da atividade. Em 2009 a empresa Opera Software introduziu um serviço chamado *Opera Turbo*[5] para seu navegador, especialmente útil em dispositivos móveis. Este serviço compacta as páginas acessadas na nuvem computacional da Opera Software para economizar recursos de comunicação nos dispositivos móveis. Banerjee et al. (2011) afirma que o investimento em computação móvel e computação em nuvem permitirá a existência de infraestruturas de informação escaláveis e seguras, também citando uma pesquisa cuja previsão espera uma conectividade através de dispositivos móveis dez vezes superior a dos desktops.

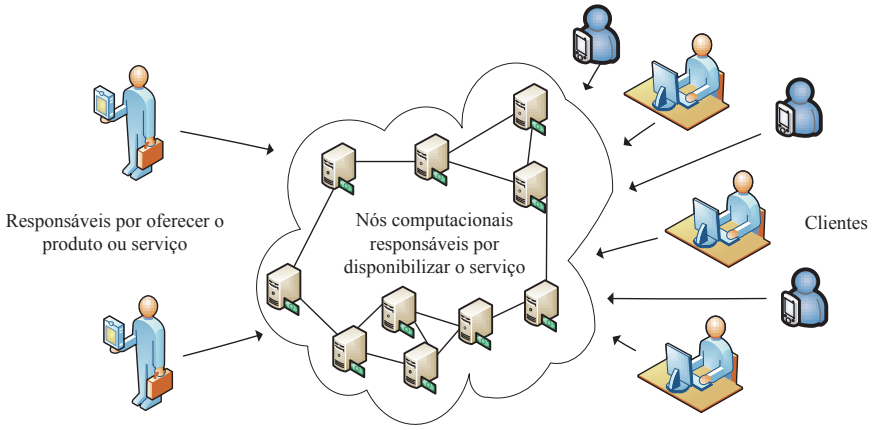
A escalabilidade neste caso apoia-se na computação orientada a serviço (SOC - *Service Oriented Computing*) que consiste em disponibilizar recursos para aplicações através da rede. A computação orientada a serviço é definida por Papazoglou et al. (2007) como um paradigma que utiliza serviços para possibilitar o desenvolvimento de aplicações de forma rápida, com baixo custo, interoperáveis e capazes de evoluir. Neste paradigma um serviço é autônomo e envolve entidades que são independentes de plataforma, que podem ser descritas, publicadas, descobertas e são fracamente acopladas. É comum em uma nuvem computacional disponibilizar aplicativos que podem ser consumidos no modelo SaaS (*Software as a Service*), onde paga-se pelo consumo do serviço sem se preocupar com as questões que envolvem a disponibilidade deste. Em 2008 a Gartner publicou um relatório[6] indicando que até 2013, 90% dos sistemas de e-commerce utilizarão um serviço baseado no modelo SaaS e 40% dos sistemas de e-commerce serão totalmente disponibilizados neste modelo.

A Figura 2 apresenta um cenário de comércio eletrônico onde múltiplos nós computacionais em uma plataforma de computação em nuvem sustentam um serviço de publicação de ofertas e demandas. Neste cenário, os comerciantes compartilham o serviço de publicação de ofertas, e este serviço através de alguns mecanismos identifica a oferta adequada à demanda

[5] http://www.opera.com/browser/turbo/

[6] http://www.gartner.com/it/page.jsp?id=734612

Figura 2: Cenário de comércio eletrônico baseado em nuvem



Fonte: Elaborado pelo autor

que cada usuários informa ao serviço. Independentemente do meio de acesso a este serviço, um dispositivo móvel ou não, os mecanismos básicos para identificação de melhores ofertas são mantidos. Mecanismos estes, que podem tirar proveito de recursos especiais que cada meio de acesso pode oferecer, por exemplo a localização do usuário em uma plataforma móvel. Este serviço utiliza uma plataforma de computação em nuvem para disponibilizar uma aplicação no modelo SaaS voltada à identificação de oportunidades comerciais.

O uso de computação em nuvem é cada vez mais associado com pequenas e médias empresas pois é um meio para que estas possam adquirir serviços que antes eram de exclusividade de grandes empresas (MALATHI, 2011). Com base nestes benefícios, este trabalho pretende disponibilizar um serviço no modelo SaaS para este cenário de comércio eletrônico em nuvem. Este serviço contribuirá para um ambiente acessível de comércio eletrônico onde o processo básico de compra e venda, que é a identificação de oportunidade, é realizado. A aplicação que disponibilizará o serviço é instalada dentro de cada um dos nós computacionais na nuvem, representados na Figura 2. Através da acessibilidade à nuvem de comércio eletrônico tanto os comerciantes como consumidores podem obter vantagens econômicas visto que o custo de manutenção do serviço é reduzido. Este serviço se diferencia de plataformas proprietárias existentes por ser mantido por uma ou mais entidades comerciais que podem investir em sua região de abrangência com valores proporcionais à demanda computacional relativa a quantidade de consumidores, utilizando-se da elasticidade proporcionada pela computação em nuvem.

1.1 Objetivos

O objetivo deste trabalho é propor um modelo de processamento de *matching* em um serviço que permita a identificação de oportunidades de compra e venda de produtos ou serviços. A identificação destas oportunidades comerciais deve ser realizada por nós computacionais através de combinações entre informações de demandas e ofertas publicadas por usuários através

de dispositivos móveis. O serviço deve levar em consideração informações de localização de seus usuários para selecionar oportunidades relevantes aos seus usuários. Este serviço deve ser capaz de armazenar estas informações e prover elasticidade para gerenciar o crescimento não uniforme de sua base de dados e demanda de utilização para determinadas regiões geográficas. Cada nó computacional deve ser responsável por uma determinada região.

Outros objetivos específicos também são importantes para a conclusão do trabalho:

- Definir um formato de dados e determinar o modelo de processamento e comunicação do serviço;
- Formular estratégias de processamento que possibilitem a aplicação de um mecanismo de *matching* entre ofertas e demandas para identificação das oportunidades de negócio;
- Permitir que um ou mais mantenedores do serviço possam compartilhar o ecossistema de comparação, podendo contribuir incluindo seus próprios nós computacionais, que serão responsáveis por determinada região geográfica;
- Ser adaptável a uma plataforma de computação em nuvem para garantir disponibilidade de recursos computacionais variáveis e de baixo custo;
- Avaliar o desempenho e aplicabilidade funcional do protótipo em cenários de acesso ao serviço.

A partir do do desejo de publicar ofertas e demandas em um ambiente de identificação de oportunidades de negócio, é importante a escolha um formato de dados adequado para representação destas informações. A partir da definição da representação de informação, é necessário formular estratégias de processamento. Estas estratégias permitem uma maior distribuição no modelo de processamento e comunicação entre os nós de processamento responsáveis pela identificação por comparação. Este serviço pretende também dispor de um mecanismo adaptável de *matching* para futuras evoluções no processo de comparação de objetos. Durante a definição deste serviço há também a preocupação de como este deve ser implantado e mantido por seus financiadores, neste âmbito se estabelece a necessidade de compartilhar a responsabilidade de acordo com a abrangência de cada interessado. A implementação de um protótipo é fundamental para avaliação funcional do serviço e sua arquitetura. Este protótipo, chamado de GTTracker (*General Trade Tracker*) será avaliado com a intenção de expor o seu comportamento e desempenho. A avaliação de desempenho em cenários de acesso é possível através da simulação de casos com base de dados reais.

1.2 Questão de pesquisa

A partir da visão geral do cenário atual de comércio eletrônico e a constante adoção de recursos em computação em nuvem por empresas de pequeno e médio porte, este trabalho procura responder a seguinte questão:

Como se comporta e como é composta a arquitetura de um serviço para identificação de oportunidades de compra e venda em uma determinada região que possa utilizar recursos de computação em nuvem e receber informações de contexto através de dispositivos móveis?

Esta questão possibilita o estudo de um mecanismo de comércio eletrônico, onde múltiplas entidades podem disponibilizar um serviço de identificação de oportunidades de negócio utilizando um ou mais algoritmos de *matching*. Este *matching* é responsável por identificar as oportunidades de compra e venda sensíveis à região geográfica de seus usuários. A arquitetura distribuída também colabora com um mecanismo homogêneo para comércio eletrônico, visto que uma plataforma compartilhada e única para identificação de oportunidades pode garantir um maior sucesso comercial entre os usuários de dispositivos móveis devido a presença de um maior número de empresas e usuários adeptos de um mesmo serviço.

O processo de pesquisa desta questão permite desenvolver uma análise sobre o mecanismo de processamento distribuído para dispositivos móveis. Sendo esta estratégia de distribuição orientada a tecnologia de computação em nuvem e voltada a atender a demanda de identificação de oportunidades através de um dispositivo pessoal, como um *smartphone*. Influenciando assim a evolução dos mecanismos de compra e venda em determinadas regiões onde há a popularização de dispositivos móveis.

1.3 Estrutura do texto

O trabalho está organizado em 7 capítulos onde o primeiro apresenta os objetivos e a questão de pesquisa. O Capítulo 2 apresenta os conceitos fundamentais necessários ao entendimento do serviço proposto. O Capítulo 3 relaciona alguns trabalhos importantes ao desenvolvimento do serviço, indicando estratégias, tecnologias e técnicas que são relevantes para a tomada de decisão no projeto do serviço. O Capítulo 4 define o modelo do serviço, apresentando as decisões de projeto, o comportamento geral do modelo, a organização e a arquitetura dos nós que compõem este serviço. O Capítulo 5 apresenta a implementação do GTTracker, as aplicações utilizadas na avaliação, os algoritmos e tecnologias empregadas. A avaliação do serviço é feita no Capítulo 6, tendo este como objetivo expor a metodologia e os resultados de cada avaliação efetuada. Por fim, o Capítulo 7 apresenta a comparação com os trabalhos relacionados e as considerações finais sobre o serviço proposto neste trabalho.

2 CONCEITOS FUNDAMENTAIS

Este Capítulo apresenta conceitos fundamentais para o entendimento do modelo proposto neste trabalho. O primeiro conceito apresenta as definições fundamentais de computação em nuvem e seus modelos de serviço, importantes para a compreensão da infraestrutura para qual o serviço foi projetado. O segundo conceito é o de computação móvel, cujos desafios e dificuldades devem ser ressaltados para que sejam devidamente conhecidos durante do desenvolvimento do serviço. O terceiro e quarto conceito revisam os princípios da computação ubíqua e a sensibilidade ao contexto, em vista a necessidade de tornar a identificação de oportunidades sensível a informações sobre o usuário.

O conceito de *matching* é o quinto tema abordado. Este conceito introduz um assunto importante neste trabalho, no qual características de ofertas disponibilizadas pelos vendedores devem ser combinadas com as características das demandas dos consumidores. Através desta combinação é possível identificar relações e determinar se há uma oportunidade ou não de negócio para o produto ou serviço de interesse do consumidor.

2.1 Computação em nuvem

A computação em nuvem tem como característica um modelo de comercialização baseado em consumo, onde os recursos computacionais podem ser adquiridos e comercializados de acordo com a necessidade de quem os consome, estando sempre disponíveis e cobrados somente se forem consumidos. Para o NIST (*National Institute of Standards and Technology*) a computação em nuvem é um modelo que permite acesso sob demanda, de forma ubíqua e conveniente, a um conjunto compartilhado de recursos computacionais configuráveis que podem ser rapidamente provisionados e disponibilizados com o menor esforço e com o mínimo de interação com o provedor do serviço (MELL; GRANCE, 2011).

São cinco as características essenciais que um modelo em nuvem deve possuir para ser caracterizado desta forma (MELL; GRANCE, 2011):

a) **Sob demanda**: o consumidor pode de forma unilateral obter mais capacidade de serviço sem que seja necessária a interação humana. Um exemplo prático seria adquirir mais espaço em uma unidade de armazenamento na nuvem sem que tenha que entrar em contato com o provedor desta unidade de armazenamento, apenas através de uma API;

b) **Acesso aberto ao recurso**: a possibilidade deste recurso ser acessível através da rede utilizando mecanismos comuns existentes em várias plataformas;

c) **Agrupamento de recursos**: os recursos computacionais do provedor devem ser agrupados de forma a servir múltiplos consumidores utilizando um modelo robusto e aparentemente sempre disponível de acordo com a demanda. Oferecendo ainda uma determinada camada

de abstração que normalmente impede que o consumidor tenha conhecimento ou controle sobre a localização exata do recurso computacional utilizado para disponibilizar o serviço que este consome;

d) **Elasticidade**: os recursos devem ser facilmente adquiridos e liberados, em alguns casos automaticamente, para rapidamente adaptar-se as necessidades do consumidor. Desta forma é possível dar ao consumidor uma impressão de recursos ilimitados;

e) **Existência de um esquema de avaliação**: estes sistemas também precisam automaticamente controlar e otimizar os recursos em uso, assim permitindo uma boa avaliação dos recursos consumidos de acordo com o serviço oferecido. Através dessa informação é possível monitorar e oferecer um bom nível de transparência com relação ao consumo tanto para o consumidor quanto ao provedor do serviço.

Dentre as características importantes associadas a computação em nuvem está a capacidade de ser adquirida sob demanda (a), onde os contratos podem ser estabelecidos rapidamente e os recursos disponibilizados em um tempo muito inferior ao modelo tradicional. A elasticidade (d) é um recurso muito interessante para o consumidor que não precisa se preocupar com a aquisição e muito menos com o descarte. Em um modelo tradicional de aquisição além de envolver um custo de instalação do determinado produto, por exemplo uma infraestrutura de rede, o consumidor necessitaria de um processo de descarte que poderia comprometer um recurso financeiro apenas para eliminar o custo de manutenção do produto.

Os serviços de computação em nuvem são oferecidos em três modelos, ou níveis distintos (MELL; GRANCE, 2011):

a) **SaaS** (Software como Serviço): no qual a aplicação é oferecida como serviço, sem que tenha que se preocupar com os recursos computacionais que são utilizados para executar a aplicação. Um exemplo é uma aplicação de Webmail, onde o consumidor através de um navegador qualquer tem acesso a seus e-mails que são armazenados em algum servidor que só o provedor do serviço tem o controle;

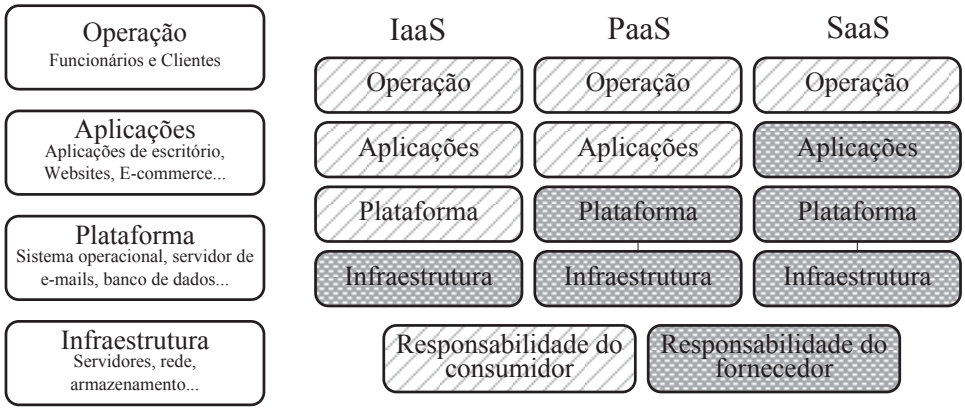
b) **PaaS** (Plataforma como Serviço): neste caso o consumidor pode criar suas próprias aplicações e as disponibilizar na nuvem, utilizando APIs e plataformas de desenvolvimento oferecidas pelo provedor do serviço. Para tal, o consumidor apenas precisa consumir recursos do provedor que lhes interessam, não necessitando controlar os recursos computacionais que oferecem tais serviços. O Google App Engine é um exemplo, em que o consumidor pode desenvolver suas próprias aplicações e executá-las na infraestrutura do Google, sem que o consumidor tenha que se preocupar com backups ou com qualquer detalhe de hardware ou sistema operacional;

c) **IaaS** (Infraestrutura como Serviço): este modelo disponibiliza ao consumidor recursos computacionais mais fundamentais como processamento, armazenamento e recursos de rede. O

consumidor neste caso ainda não controla diretamente os recursos físicos e sim recursos virtualizados como máquinas virtuais, cabendo ao consumidor se preocupar com o sistema operacional, suas configurações e demais aplicações instaladas.

Na Figura 3 é possível observar como os três modelos se encaixam na pilha de operação do sistema. Em um esquema normal o consumidor seria responsável e teria de arcar com os custos de implantação, manutenção e descarte de todos os níveis, da infraestrutura aos usuários da aplicação. Utilizando IaaS o consumidor precisa se responsabilizar tanto pela instalação do sistema operacional, como pela instalação ou desenvolvimento da aplicação e pela operação sobre os aplicativos. Para o modelo PaaS o consumidor ainda precisa se responsabilizar pela aplicação. Enquanto no SaaS basta possuir conhecimento de operação da aplicação final, não sendo necessário possuir conhecimentos sobre a infraestrutura ou plataforma e muito menos desenvolver a própria aplicação.

Figura 3: Níveis de serviço e suas responsabilidades



Fonte: Elaborado pelo autor

O modelo de computação em nuvem pode se confundir com a computação em grade (*grid computing*). Para Breitman (2010), a distinção não é clara pois ambos tem como objetivo a redução de custos e aumento de flexibilidade e confiabilidade. Para distinguir a autora destaca que a alocação de recursos na grade é uniforme enquanto na nuvem os recursos são alocados sob demanda.

O NIST também define quatro modelos de implantação de uma nuvem, estas podem ser: privadas quando a infraestrutura é de uso exclusivo por uma única organização com vários consumidores (separadas unidades ou setores de uma mesma organização); comunitárias quando forem de uso exclusivo de um determinado grupo de organizações que compartilham suas premissas organizacionais; públicas quando podem ser utilizadas por qualquer consumidor, e; híbridas quando forem compostas de dois ou mais modelos de implantação (privada e pública por exemplo) (MELL; GRANCE, 2011) (MEIYAN, 2013).

Por estar geralmente em um local desconhecido (principalmente no caso da nuvem pública) os serviços oferecidos podem ser acessados de qualquer local através da Internet. Desta forma a

mobilidade passa a ser possível e até natural para os consumidores destes serviços. Um exemplo claro desta mobilidade está em serviços criados justamente para simplificar a transferência e armazenamento de informação em um local acessível por vários meios, como o Dropbox[1] ou Ubuntu One[2]. Estes serviços de armazenamento na nuvem podem utilizar outro serviço disponibilizado sob demanda em nuvem, por exemplo o Amazon S3[3], para disponibilizar um serviço de boa qualidade com um custo compatível a demanda de seus usuários.

2.2 Computação móvel

A disponibilização de redes sem fio permitiu naturalmente o surgimento de dispositivos que podem se comunicar com outros equipamentos mesmo quando em movimento. Com estas redes sem fio os dispositivos móveis (celulares, smartphones, notebooks, etc.) tornam-se mais flexíveis aos seus usuários, tornando-se mais versáteis e com comunicação independente de localização (FORMAN; ZAHORJAN, 1994). A computação móvel além de permitir a mobilidade do dispositivo agrega algumas limitações e dificuldades (SATYANARAYANAN, 2011) (FORMAN; ZAHORJAN, 1994) (HAMMERSHOJ; SAPUPPO; TADAYONI, 2010):

a) Variação da qualidade de rede, tanto em disponibilidade quanto velocidade e latência;

b) Recursos computacionais limitados nos dispositivos móveis, tanto em memória quanto processamento;

c) Consumo de energia;

d) Variação dos recursos oferecidos na rede local, onde um dispositivo ou informação pode estar acessível em uma rede e não mais quando trocar de enlace;

e) Mudança de endereço e identificação do próprio dispositivo;

f) Rede heterogênea onde as tecnologias de transmissão, topologia e protocolos podem variar de uma rede a outra;

g) A segurança precisa ser observada com maior cuidado.

O consumo de energia (c) é um problema em dispositivos móveis devido a necessidade de acrescentar recursos de processamento e comunicação mais avançados que consomem mais energia mantendo ainda o conjunto de todos os componentes leves e de tamanho reduzido. Para manter o tamanho atrativo e uma autonomia aceitável aos usuários, limita-se a capacidade computacional do dispositivo (b). Dentre as dificuldades, para Pitoura e Samaras (2001) há um aumento significativo de custo computacional envolvido na busca de informações quanto à

[1] https://www.dropbox.com/
[2] https://one.ubuntu.com/
[3] http://aws.amazon.com/pt/s3/

comunicação tanto com a mudança de localização por parte do usuário quanto no uso de código e dados móveis. Dispositivos que se movimentam precisam rastrear o ambiente para encontrar canais de comunicação e precisam se comunicar constantemente para caracterizar sua presença na rede.

Na computação distribuída um middleware permite que componentes de uma determinada arquitetura interajam tanto para troca de informação quanto para acessar serviços que seriam complexos para serem tratados em apenas uma camada de software. Esta complexidade pode ser atenuada por este middleware, que facilita a implementação de aplicações (QILIN; MINTIAN, 2010). Para que a comunicação entre dois dispositivos seja realizada é recomendado o uso de um middleware que abstraia e simplifique as peculiaridades técnicas exigidas em uma conexão. Middlewares específicos para redes móveis abstraem boa parte das dificuldades técnicas enfrentadas por dispositivos móveis (BELLAVISTA; CORRADI, 2006). E devido a esta abstração, exigem técnicas que as aplicações distribuídas não móveis não costumam requerer (GRIGORAS, 2006):

a) Transparência de comunicação independente do dispositivo;

b) Boa relação custo (computacional) benefício (funcional) dos protocolos envolvidos;

c) Simplicidade dos protocolos;

d) Escalabilidade em respeito ao número de membros da rede;

e) Portabilidade de serviços e aplicações;

f) Adaptabilidade dos protocolos e aplicações às características dos nós de rede;

g) Reconfiguração dinâmica mesmo com mudança da topologia;

h) Interações assíncronas como resultado da mobilidade;

i) Protocolos que garantem qualidade de serviço;

j) Tolerância a falhas e recuperação;

k) Segurança de comunicação e execução remota.

A transparência de comunicação (a) e a adaptabilidade as características dos nós de rede (f) fazem com que os dispositivos possam interagir com outros dispositivos próximos. Esta computação móvel através de técnicas mais elaboradas passa a ter sua transparência de comunicação evoluída a ponto de tornar a comunicação simples aos desenvolvedores e imperceptível aos seus usuários. Combinadas, estas técnicas podem se orientar a um objetivo em comum e passam a ser caracterizados como sistemas pervasivos ou ubíquos (SATYANARAYANAN, 2011).

2.3 Computação ubíqua

A computação ubíqua foi apresentada cientificamente por Weiser (1999) em seu artigo "The computer for the 21s century", onde considera que a interação máquina-homem pode ser caracterizada como computação invisível se esta agir de acordo com as ações naturais do comportamento humano. Desta forma o sistema de computação pode agir de forma proativa ou auxiliar o usuário em suas tarefas com o mínimo de distrações. Um exemplo citado no artigo é possibilidade de detectar a presença de uma pessoa autorizada e abrir a porta automaticamente durante o tempo necessário para sua passagem.

Boa parte dos dispositivos que tiram proveito da computação ubíqua apoiam-se na grande disponibilidade de recursos de comunicação. Esta comunicação passa a ser oferecida a baixo custo e de forma comum para um grande grupo de pessoas com muitas finalidades, favorecendo o desenvolvimento de inovações a partir destes serviços (SCHILIT, 2011). Nota-se ainda que estes recursos são oferecidos como serviço, utilizando o mesmo princípio de contabilização onde o consumidor só paga pelo que consome.

A tecnologia de computação móvel impulsionou a computação ubíqua quando ofereceu a possibilidade de comunicação através de canais sem fio como Bluetooth, Xbee e Wifi. Outra tecnologia presente é o uso RFID (*Radio-Frequency Identification*)(VAIDYA; MAKRAKIS; MOUFTAH, 2012). Esta tecnologia permite que o ambiente seja reconhecido pelos dispositivos, e ao identificarem os elementos que estão a seu alcance podem atuar de forma diferente. Com esta tecnologia os dispositivos colaboram para a criação de um ambiente invisível aos olhos de seus usuários tornando esta computação oculta.

Os dispositivos podem considerar uma variedade de indicadores para realizar a computação, por exemplo:

a) Localização geográfica;

b) Temperatura do ambiente;

c) Reconhecimento de voz e sons do ambiente;

d) Indicadores de movimento;

e) Informações contidas em mensagens;

f) Interações com eletrodomésticos.

A localização geográfica (a) é muito importante para que os dispositivos selecionem em casos especiais apenas informações que podem diretamente influenciar o usuário. Um exemplo claro é a programação de rota em um GPS, este dispositivo pode considerar a localização do usuário e os acontecimentos nas regiões próximas para tomar decisões de qual é a via mais adequada ao usuário. O reconhecimento de voz e sons do ambiente combinados com indicadores

de movimento (c)(d) já podem auxiliar a detecção de situações de risco para o motorista, como a sonolência (recurso já disponível em alguns carros). Combinado com o recurso anterior, estes podem por exemplo indicar ao motorista uma rota onde há vários hotéis para descanso.

2.4 Sensibilidade ao Contexto

Para Dey (2001), contexto é uma informação que pode caracterizar a situação de uma entidade. Sendo esta uma pessoa, local ou objeto que pode ser considerado relevante à interação entre um usuário e uma aplicação, incluindo o usuário e a aplicação, e uma aplicação é sensível ao contexto quando esta responde de forma diferente ao usuário de acordo com o contexto.

Uma aplicação sensível ao contexto precisa executar três atividades básicas (HUANG; LIU; LI, 2011): coletar informação; armazenar as informações relativas ao contexto, e; utilizar o contexto em seu comportamento. Tendo que na computação ubíqua o contexto é muito dinâmico, Abowd et al. (1999) define quatro tipos principais de contextos que podem responder questões básicas sobre o contexto de determinada entidade além do quem, o que, quando e onde. Por exemplo:

a) **Localização**: a qual pode revelar a ligação e relação entre outros objetos ou pessoas;

b) **Identidade**: que permite obter outras informações como telefone, endereços, lista de amigos, nível de parentesco e outros dados pessoais;

c) **Tempo**: que pode indicar ou até eliminar outras relações pressupostas visto que atividades podem ocorrer sem relação de acordo com o tempo em que elas ocorrem;

d) **Atividade**: que oferece oportunidade de detectar possíveis ações futuras e também relacionar outras entidades envolvidas na mesma atividade.

A localização (a) e a identidade (b) combinadas através do tempo (c) podem denotar também qual é a atividade (d) que determinado usuário está executando. Um exemplo da situação é a movimentação de um usuário através de uma rodovia. Sabendo da localização, a identidade e o tempo detectadas em dois momentos diferentes, pode-se possibilitar a um sistema presumir que o usuário está se dirigindo a determinada região. Através desta informação um determinado sistema pode preparar o ambiente para a chegada deste usuário. Este seria um bom uso do contexto em um sistema ubíquo.

A modelagem de contextos em sistemas ubíquos precisam levar em conta outros fatores como (THOMAS; LINNHOFF-POPIEN, 2004):

a) Distribuição do modelo nos dispositivos envolvidos;

b) Avaliação parcial de informações e estrutura do conhecimento;

c) Riqueza e qualidade da informação;

d) Ambiguidade e nível de precisão das informações;

e) Aplicabilidade do contexto em determinado ambiente.

A possibilidade de identificação de um contexto errôneo, que não corresponde ao contexto idealizado pela aplicação para tomar determinada ação, exige que estas aplicações certifiquem o determinado contexto levando em consideração a riqueza e qualidade da informação (c). Esta qualidade pode ser assegurada por validações (b) através de estudos sobre determinadas situações e pelo conhecimento adquirido através destes estudos. Prevendo inclusive a identificação de situações onde há ambiguidade ou nível de precisão (d) insuficientes.

A análise do contexto para tomada de ações pode ser útil em diversas áreas além da computação móvel e ubíqua, sendo aplicada por exemplo em algoritmos que escolhem e filtram as informações mais adequadas ao usuário (MENG et al., 2008), no apoio a decisão (NWIABU et al., 2011) e na definição de relações de confiança (FRANSEN et al., 2006).

2.5 Matching

Durante a comparação de informações pode-se estabelecer uma relação de equivalência ou de correspondência. Neste trabalho o conceito de *matching* se aplica à criação ou definição de um fator de equivalência para comparar um objeto de oferta a um objeto de demanda.

Algoritmos de *matching* são normalmente empregados em aplicações onde há intenção de comparar padrões ou estabelecer equivalências em textos. Baker (1993) desenvolveu uma teoria de padrões de correspondência para verificar duplicidades de código em programas sem levar em consideração nomes diferentes para os mesmos parâmetros. Amir et al. (2003) e Amir e Nor (2007) apresentam o conceito de *function matching* e o que consideram uma generalização natural do conceito de Baker, que explora problemas onde não há solução através de identificação de padrões visto que não é aceitável a relação de transitividade na atividade de comparação destas informações, como por exemplo: reutilização de espaços de memória para variáveis; determinação de estruturas de proteínas, e; processamento de imagens.

Há outras aplicações práticas para algoritmos de *matching*. Por exemplo:

a) Verificação de equivalência semântica e de sintaxe (GIUNCHIGLIA; YATSKEVICH; SHVAIKO, 2007) (WANG et al., 2013) (HACHICHA; DARMONT, 2013);

b) Reconhecimento de objetos em imagens através de reconstrução 3D ou comparação de pontos em imagens (ZHAO; XUE; MEN, 2010) (CHERTOK; KELLER, 2010) (HAN et al., 2013);

c) Comparação de metadados organizados em árvore (XIUZHEN, 2010);

d) Comparação de sequências de proteínas (SIBAI; ZAKI, 2010);

e) Descoberta de serviços (YIN; CUI; MA, 2010);

f) Identificação de grupos em redes sociais (GREENLAW, 2010);

g) Identificação de relações entre eventos (GUO; WEI; HAN, 2008);

h) Identificação de padrões em aplicações de detecção de intrusão (ZHENG; LU; NAHUM, 2011).

Cada aplicação ou modelo exige uma estrutura de avaliação diferenciada devido aos conhecimentos envolvidos. O reconhecimento de objetos (b) necessita que o algoritmo identifique pontos importantes de cada imagem, e os compare relacionando a sua posição relativa aos outros pontos da imagem. Se a análise reconhecer que ambas possuem relacionamento, o *matching* entre ambas as imagens é caracterizado. Este relacionamento pode não ser exato, ou pode ignorar alguns pontos, esta margem aceita para o reconhecimento é necessária visto que dificilmente duas imagens exibem todos os pontos com muita clareza.

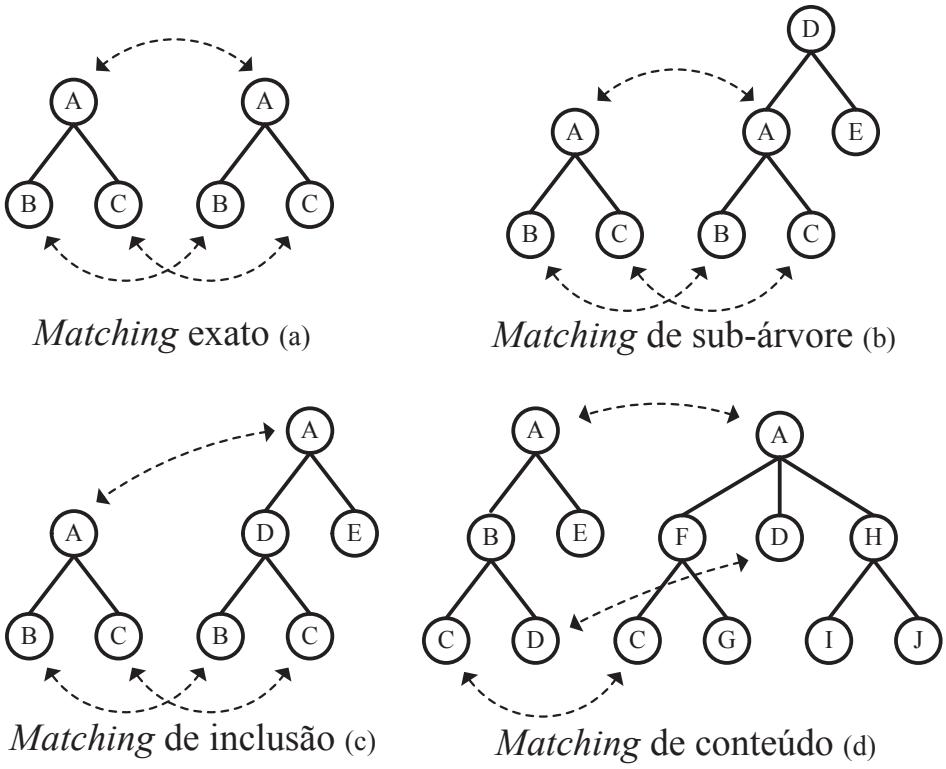
Os critérios utilizados nos algoritmos de comparação variam de acordo com a situação em que são aplicados e podem implicar no desempenho geral da solução técnica adotada. Quanto mais complexo, maior o tempo necessário para processar o volume de informação envolvida na comparação. Na análise de informações em um conjunto grande de dados o *matching* é um passo importante onde é criada a relação entre informações. Neste processo de análise de informações o tempo gasto em processamento de dados pode atingir 90% do tempo total apenas no processo de *matching* (CAO et al., 2010). Em um projeto de sistema onde um baixo tempo de resposta é exigido, o processamento dessa informação para *matching* é muito importante. No contexto deste trabalho, para um comprador trafegando próximo ao vendedor, pode ser a diferença de tempo necessária para que o sistema alerte o comprador enquanto este ainda estiver convenientemente próximo do vendedor.

2.5.1 Análise de matching

Xiuzhen (2010) apresenta um exemplo muito claro de *matching* entre dados dispostos em estruturas de árvore. A Figura 4 exibe quatro modelos distintos de *matching* em informações dispostas em uma estrutura de árvore. Cada modelo verifica de forma diferente como uma árvore pode se relacionar com a outra. O modelo (a) apresenta o *matching* entre duas árvores exatamente iguais. O modelo (b) demonstra uma equivalência parcial entre uma árvore e uma subárvore de outra árvore. O modelo (c) apresenta um uma equivalência desconsiderando os níveis, porém ainda denota uma determinada equivalência. O modelo (d) relaciona duas árvores sem considerar a sua estrutura, apenas pelos nós existentes em ambas as árvores.

Cada modelo de *matching* apresentado nesta seção do trabalho possui um propósito e é aplicado a um problema diferente, no qual necessitam um nível de relação e precisão diferente que esteja de acordo com sua proposta. Alguns comparam imagens, outros comparam textos,

Figura 4: Modelos de *matching* em informações dispostas em estrutura de dados em árvore



Fonte: Adaptado de Xiuzhen (2010)

enquanto alguns comparam sequências de proteínas que estão em um formato específico. Alguns problemas exigem algoritmos mais complexos e que precisam reconhecer entre dois ou mais tipos de relações de *matching*. Na Figura 4 o algoritmo precisaria identificar e decidir se a relação entre as árvores de (c) é uma relação de inclusão (c) ou de conteúdo (d), visto que se encaixam em ambos os casos. O *matching* de estruturas em árvore demonstra como a representação de dados pode afetar a avaliação entre duas representações. A representação pode tornar-se mais complexa a medida que há um aumento de informações necessárias para descrever o que deve ser avaliado.

2.5.2 Matching de interesses

O processo de inferência de relação na tomada de preços pode ser considerada uma avaliação de interesses, de parte de quem vende em encontrar possíveis compradores, e de parte de quem compra em encontrar as ofertas. No processo de projeto e desenvolvimento de algoritmos de *matching* para modelos de análise de interesse há três requisitos gerais que podem afetar diretamente a eficiência do sistema (LIU; THEODOROPOULOS, 2010):

- **Precisão dos filtros**: a qualidade dos filtros em selecionar corretamente os interesses pode diminuir a carga tanto de processamento como de comunicação, eliminando can-

didatos ruins na seleção de ofertas. Um filtro muito restritivo pode eliminar ofertas que poderiam ser interessantes ou até ideais;

- **Eficiência de execução**: o tempo necessário e a complexidade envolvida na avaliação pode degradar demasiadamente o desempenho da aplicação e aumentando significativamente o tempo de resposta final. Em um ambiente móvel onde o usuário pode se deslocar constantemente uma oportunidade pode deixar de ser válida de acordo com a sua movimentação entre as áreas onde a oferta é feita, tornando inútil o processamento de tal informação;

- **Capacidade de captura**: a falha na captura de informações tanto de quem oferta como de quem busca pode inviabilizar a análise de dados pois o processo depende de ambas as informações completas para criar a relação.

O *matching* de interesses aproxima-se do processo de identificação de oportunidades de negócio. Ao realizar a análise entre os elementos que são ofertados e as demandas de quem deseja adquirir é importante observar as informações relevantes a cada tipo de produto e o consumidor alvo. Ferramentas de e-commerce necessitam de mecanismo que tornem ofertas mais visíveis para consumidores, estejam eles buscando informações em alguma página na Internet ou recebendo informações em um anúncio pago.

2.5.3 Matching no u-commerce

O u-commerce (*ubiquitous commerce*) é uma evolução do modelo de negócio chamado e-commerce (EVANS; HU, 2006). Através do uso de uma rede ubíqua é possível disponibilizar ao usuário um canal personalizado de comunicação entre os envolvidos em uma relação de compra e venda, onde a relação entre vendedor e comprador é realmente útil aos envolvidos no momento adequado (WATSON et al., 2002). Nesta aplicação a análise de *matching* é um processo importante para verificação das informações que devem ser entregues aos usuários.

Há uma evolução constante no campo de pesquisa do comércio ubíquo, trabalhos recentes apresentam mecanismos diferentes para identificação de informações relevantes para uma relação de comércio eletrônico. Estes trabalhos utilizam informações como localização, gênero, idade, conteúdo e inclusive a reputação dos envolvidos para formar um nível de confiança ou relevância sobre o produto/serviço ou ao vendedor/consumidor (MANVI; NALINI; BHAJANTRI, 2011) (FRANCO et al., 2011) (SANCHEZ-PI; MOLINA, 2009).

Não foi encontrado um trabalho que oferece um modelo de *matching* estabelecido ou de uso comercial, desta forma é prudente não utilizar um modelo complexo e sim adotar a possibilidade de uso de um mecanismo mais genérico, a fim de eliminar esta dependência ao modelo de comparação. Por este motivo a comparação será realizada em sua forma mais simples. Esta comparação se dá através da análise de dois elementos: uma oferta publicada por um vendedor e uma demanda publicada por um usuário. Através de uma função que recebe estes dois objetos

como parâmetros será necessário identificar se é considerada uma oportunidade de negócio ou não.

3 TRABALHOS RELACIONADOS

A modelagem de uma serviço para identificação de oportunidades comerciais pode envolver uma série de mecanismos e conceitos que já foram estudados em outros trabalhos relacionados. Este Capítulo apresenta alguns trabalhos que se aproximam da questão de pesquisa e os aborda através de alguns conceitos observados em tais trabalhos com relação ao modelo de distribuição, o formato e representação de dados, e os mecanismos de otimização e processamento envolvidos.

O processo de avaliação de um grande volume de informações coletadas torna necessária a adoção de um modelo distribuído de processamento, no qual vários dispositivos colaboram na análise da informação. Este modelo de processamento distribuído pode tirar proveito do formato e representação de dados para melhorar sua eficiência através de mecanismo de otimização. O escopo também é um fator que influencia o desempenho do sistema, as ofertas são feitas em determinada região e tendem a ser mais procuradas em tal região visto a preocupação preliminar do ofertante em estar próximo do consumidor. Porém, esta não é a regra absoluta visto que alguns consumidores tendem a encontrar produtos que lhes interessem em outras regiões quando não há ofertas que lhes agradem em sua região. Os filtros são responsáveis por delimitar este fator, porém a própria estrutura pode utilizar esta informação de escopo para otimizar o processo de *matching*.

Neste Capítulo algumas aplicações práticas citadas nos trabalhos relacionados são apresentadas a fim de caracterizar a aplicabilidade dos modelos no auxilio à resolução da questão de pesquisa. Para concluir, é feita um resumo sobre os trabalhos para destacar as diferenças e características de cada proposta. Dentre os trabalhos se destacam os sistemas baseados no modelo *publish/subscribe* como o DREAM (BUCHMANN et al., 2003), S-ToPSS (PETROVIC; BURCEA; JACOBSEN, 2003), JTangPS (QIAN et al., 2008), JTangCSPS (QIAN et al., 2011) e Guo, Wei e Han (2008) frente ao MapReduce (CAO et al., 2010) que também é utilizado no processamento de grande volume de dados a fim de coletar informações relacionadas (DEAN; GHEMAWAT, 2008). Este Capítulo foi organizado em temas específicos que influenciam a concepção do serviço e sua arquitetura, apresentando em cada seção como os trabalhos se posicionam em cada tema. Estes temas foram escolhidos após a síntese de características dos trabalhos selecionados. Os trabalhos foram selecionados a partir de modelos de arquitetura de processamento de informações para comparação.

3.1 Modelos de distribuição e processamento de informação

O modelo de distribuição e processamento determina como o sistema se comporta com relação às informações recebidas, armazenadas e processadas. Dois modelos se destacam quando há necessidade de comparação de grande volume de informações: o *publish/subscribe* e o MapReduce. Ambos modelos são comumente implementados de forma distribuída utilizando

técnicas de processamento paralelo para atingir em menor tempo os resultados de avaliação de *matching*.

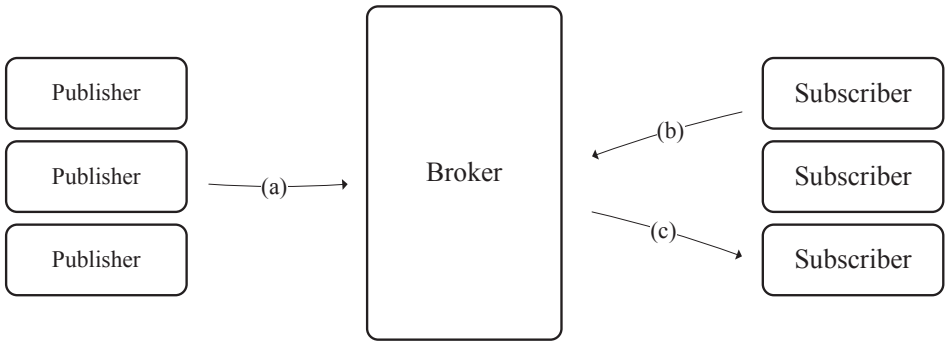
O modelo *publish/subscribe* é um modelo em que alguns usuários ou clientes precisam registrar seus interesses enquanto outros precisam informar disponibilidades. Através da combinação entre a oferta e a demanda o usuário interessado é informado da disponibilidade. Há dois tipos de sistemas *publish/subscribe* (ELKHIYAOUI et al., 2009):

- **Baseado em tópicos**: determinados tópicos ou assuntos são disponibilizados para que os usuários registrem interesses ou disponibilidades;

- **Baseado em conteúdo**: usuários registram interesses por determinados conteúdos enquanto outros disponibilizam conteúdos que podem atender completamente ou parcialmente os interesses registrados.

Para os sistemas baseados em tópicos o relacionamento entre interesse e disponibilidade é facilmente feito através da análise dos tópicos, não necessitando de algoritmos complexos de combinação. Para sistemas baseados em conteúdo é necessário a adoção de filtros que modifiquem ou analisem separadamente as informações contidas no conteúdo. Estes filtros podem fazer parte de uma máquina de inferência que opera através de um algoritmo de *matching*. Na Figura 5 é possível observar o modelo *publish/subscribe*, no qual possui *publishers* responsáveis por publicar as suas informações (a), *subscribers* responsáveis por assinar (b) a determinados tópicos ou conteúdos no *broker*. O *broker* é responsável por armazenar as informações e compará-las, e ao encontrar uma relação é responsável por notificar (c) os *subscribers* (MARGARA; CUGOLA, 2013). Em uma estrutura *publish/subscribe* é possível existir um ou mais *brokers*.

Figura 5: Modelo *publish/subscribe*



Fonte: Elaborado pelo autor

Este algoritmo de *matching* existente dentro da estrutura do *broker* tende a aproximar valores similares ou próximos que estão dentro do conjunto de alternativas corretas a informação requisitada, sem que o sistema contenha todas as combinações possíveis (FABRET et al., 2001).

Um exemplo de *matching* em um sistema *publish/subscribe* pode ser observado de forma prática no seguinte cenário: um determinado vendedor oferece biscoitos de brigadeiro na região, porém só há interessados em biscoitos de chocolate. Uma função pouco flexível (baseada em tópicos) não iria identificar que a rosquinha de brigadeiro é feita de chocolate, já uma função de *matching* que conheça a composição das biscoitos possa identificar a equivalência (baseada no conteúdo).

Uma base de dados centralizada poderia tornar o sistema frágil devido a alta dependência entre os elementos que compõem o sistema. Este sistema monolítico torna-se difícil de estender e adaptar à realidade devido a distribuição e heterogeneidade da informação. Outro problema de um sistema fortemente acoplado é o alto índice de falha em todo o sistema caso um dos elementos do sistema deixe de funcionar (BUCHMANN et al., 2003).

O fraco acoplamento do modelo de comunicação *publish/subscribe* permite que o mecanismo funcione mesmo que parcialmente, com pouco ou nenhum efeito negativo no sistema em geral. O JTangCSPS utilizam princípios de outros paradigmas auxiliares como o P2P (*Peer-to-Peer*) para tornar mais robusto e confiável o serviço de notificação de eventos (QIAN et al., 2011). O JTangCSPS é derivado do JTangPS (QIAN et al., 2008) que se propõe um sistema baseado em eventos e uma alternativa ao SOA que depende de comunicação síncrona para execução. Para formar a rede P2P utiliza-se uma implementação DHT (*Distributed Hash Table*) chamada de Pastry[1].

O S-ToPSS utiliza o *publish/subscribe* para permitir a execução de funções complexas que tornem possível o *matching* semântico entre duas informações. Ele utiliza funções que convertem sinônimos, agregam o conceito de hierarquia para classificação de equivalência de informações (por exemplo a equivalência entre “escola” e “universidade”, sendo que estas não são sinônimos), e mapeamentos que podem auxiliar em classificações semânticas cujo conceito de hierarquia não é bem definido (PETROVIC; BURCEA; JACOBSEN, 2003).

Outra solução utilizada é o MapReduce (DEAN; GHEMAWAT, 2008). Cao et al. (2010) usa o modelo, no qual o processamento paralelo e distribuído é composto de três estágios:

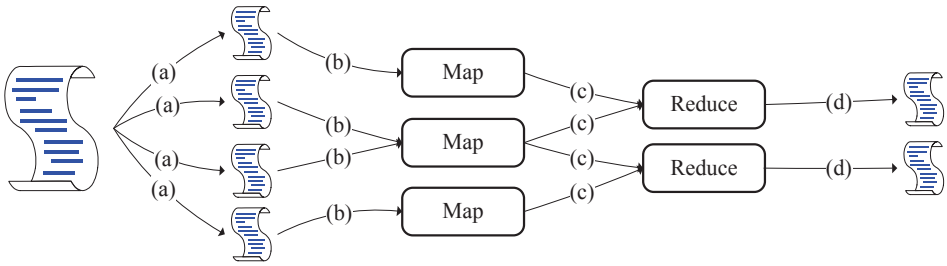
- **Construção**: onde as regras são decompostas em porções menores a fim de serem distribuídas aos nós responsáveis pelo processamento;

- **Mapeamento**: onde as informações são passadas aos nós processadores a fim de que estes avaliem a relação correspondente entre cada uma das informações existentes. Normalmente neste passo geram-se mais informações que são passadas a outros nós para processamento e redução;

- **Redução**: onde os resultados são avaliados para verificar quais informações de fato são equivalentes e descartar informações pouco relacionadas.

[1] http://www.freepastry.org/

Figura 6: Modelo MapReduce



Fonte: Adaptado de (EKANAYAKE; PALLICKARA; FOX, 2008)

A Figura 6 apresenta o modelo de processamento do MapReduce. Os dados são divididos em porções menores (a) e então alocados (b) em diferentes máquinas responsáveis pelo mapeamento (*map*). Após o mapeamento, as informações resultantes são encaminhadas (c) às máquinas responsáveis pela redução (*reduce*). Após a redução o processo resulta em um conjunto de informações processadas (d). Em algumas implementações (como o Apache Hadoop[2]) tanto o processo de *map* como o processo de *reduce* podem ser executados por uma mesma máquina em momentos diferentes, esta máquina na estrutura do MapReduce é chamada de *worker*.

No framework Hadoop, o processo de MapReduce é desenvolvido customizando-se os métodos de *map* e *reduce*. O método *map* deve ler a informação a ser processada e gerar uma lista de itens chave/valor. O método *reduce* por sua vez processa todos os valores de determinada chave reduzindo para a informação resultante. O controle de tarefas, o escalonamento de cada operação nos *workers* e o armazenamento de informação são responsabilidades de outras aplicações. O *JobTracker* é responsável por gerenciar os trabalhos, enquanto o *TaskTracker* é responsável por gerenciar os *workers*.

3.2 Representação de dados

A comunicação entre componentes de um sistema exige uma formalização quanto ao formato de dados. Este formato ou modelo de dados precisa representar o conteúdo e respeitar o significado semântico da informação. Esta representação de dados pode ser utilizada tanto para comunicação interna (entre os componentes do sistema) e externa (entre o sistema e outros dispositivos externos que interagem com os componentes deste sistema).

O DREAM (BUCHMANN et al., 2003) utiliza o modelo MIX (*Metadata based Integration model for data X-change*). Este modelo é baseado no conceito de objeto semântico, no qual é sempre acompanhado de suas informações de contexto semântico (BORNHOVD; BUCHMANN, 1999). Este contexto é basicamente composto de um conjunto de meta atributos que também são representados por objetos semânticos. Estes mesmos objetos sempre são caracterizados com um rótulo que descreve o significado na informação no contexto de onde o

[2]http://hadoop.apache.org/

Figura 7: Exemplos de representação do modelo de dados MIX

```
SemObj = < CarOffer, {
            < Company, “Budget” >,
            <Location, “J.F. Kennedy Int’l. Airport”, {<LocationCode, “FullName”>} >,
            <VehicleType, “Economy”,                  {<TypeCode, “FullClassName”>}>,
            <DailyRate, 57.99,                        {<Currency, “EUR”>, <Scale,1>}>,
            <PickUpDay, “07/04/1999”,                 {<DateFormat, “DD/MM/YYYY”>}>,
            <Extras, “Air Conditioned”>,
            <Extras, “Automatic”>
          }>
```

Fonte: Adaptado de Bornhovd e Buchmann (1999)

objeto se aplica, como no RDF (*Resource Description Framework*) e XML (*Extensible Markup Language*). A Figura 7 exemplifica o conjunto de informações de duas ofertas de automóveis, note que uma informação pode possuir indeterminados níveis de objetos (um objeto dentro de outro, encadeados).

Para representar as informações os trabalhos JTangPS (QIAN et al., 2008) e JTangCSPS (QIAN et al., 2011) utilizam o formato RDF. Este formato é baseado em XML e torna a leitura simples tanto por máquinas como por humanos, podendo representar não só o conteúdo como a hierarquia de informações. A hierarquia permite que as propriedades do objeto sejam relacionadas semanticamente dentro da representação.

O S-ToPSS e o modelo de Guo, Wei e Han (2008) utilizam dados em formato chave/valor, onde cada objeto semântico é representado apenas por um conjunto de chaves associadas a seus respectivos valores. A representação de dados se difere do MIX por não utilizar o conceito de objetos encadeados. Estes dados simplificam o modelo de inferência pois não necessitam que a comparação seja feita em mais de um nível. Representações que utilizam mais de um nível de objetos demonstram um baixo grau de escalabilidade devido ao consumo de memória e o custo de gerenciamento de informação (GUO; WEI; HAN, 2008). Em contrapartida os objetos com mais de um nível permitem um detalhamento mais preciso da informação, facilitando a criação de índices que aceleram a busca dos objetos (GUO; WEI; HAN, 2008).

O trabalho de Cao et al. (2010) não deixa claro qual é o formato de dados utilizado no protótipo implementado, apenas define que as regras avaliadas no *matching* são uma tupla composta por condições e ações. Existem outros trabalhos que utilizam MapReduce para analisar dados sem formato específico definido (textos longos, valores coletados por alguma aplicação ou até arquivos binários) e que resultam em dados processados no formato chave/valor (SHIH; LIAO; CHANG, 2011) ou que resultem outros em formatos binários uteis a outra aplicação (EKANAYAKE; PALLICKARA; FOX, 2008).

As representações MIX e chave/valor possuem estruturas bem definidas. Estas estruturas declaram tanto as informações existentes no objeto como a relação semântica destas informações no contexto do objeto. Estas representações são flexíveis quanto conteúdo sem sacrificar o formato da mensagem. Estas chaves podem ser utilizadas ainda nos algoritmos no qual podem

se basear em mecanismos de otimização, como índices, para acelerar o processo de comparação.

3.3 Mecanismos de otimização e processamento envolvidos

Para o processo de avaliação ser realizado é necessário que o algoritmo de *matching* obtenha ambos objetos e os compare através de um determinado conjunto de regras. Se o número de objetos comparados e a quantidade de informações em cada objeto for muito grande, o processo de avaliação de *matching* pode inviabilizar a identificação de objetos relacionados. Para acelerar o processo é de grande importância utilizar mecanismos que eliminem candidatos a comparação ou reduzam a quantidade de informações avaliadas entre os objetos.

Dentre as estratégias mais comuns de otimização quanto a relacionamento de informação, está o uso de índices. Estes índices são recursos importantes em bancos de dados relacionais por permitir o agrupamento de informações através de um determinado valor, utilizando algoritmos como BTREE (*binary tree*) e tabelas *hash*. Este mecanismo é eficiente para casos onde há uma cardinalidade alta. Em casos onde a cardinalidade é baixa o mecanismo se torna ineficiente e o índice passa a ser um problema para a aplicação pois esta precisa relacionar uma grande quantidade de objetos para eliminar poucos valores em uma possível avaliação de *matching*.

Para aplicações baseadas em *publish/subscribe* que operam sobre uma arquitetura distribuída em vários dispositivos é possível utilizar o roteamento de mensagens entre os elementos da rede para eliminar candidatos a comparação. O DREAM (BUCHMANN et al., 2003) utiliza dois filtros de roteamento distintos para otimizar o processo (GERO; BUCHMANN; FIEGE, 2002):

- **Baseado em cobertura** (*Covering-based routing*): no qual a mensagem não é encaminhada para os elementos da rede que já receberam uma solicitação equivalente mas não retornaram informação;
- **Baseado em fusão** (*Merging-based routing*): na qual os elementos na rede fundem entradas na tabela de roteamento que são muito próximas. Esta abordagem evita que as aplicações armazenem grandes tabelas de roteamento.

Guo, Wei e Han (2008) levam em consideração um filtro baseado em cobertura similar ao DREAM, onde regras equivalentes ou exclusivas são tratadas para evitar avaliações desnecessárias. Porém, utiliza outros mecanismos para melhorar o desempenho com baixo consumo de memória:

- **Verificação em tabela local**: quando a aplicação recebe a mensagem, antes de encaminhar a outros elementos da rede, avalia se já possui um destinatário ou se pode fornecer a resposta. Neste caso a aplicação deixa de encaminhar para os outros elementos a mensagem;

- **Histórico de requisições**: verifica através de análise probabilística quais são as regras de *matching* mais promissoras, assim avaliando-as primeiro;
- **Índices por atributo**: a aplicação passa a criar filtros indexando cada atributo do objeto com um mecanismo diferente. Para textos, por exemplo, é utilizado um algoritmo baseado em prefixos e sufixos;
- **Roteamento por faixa numérica**: verifica um atributo numérico presente no conteúdo da mensagem contra uma tabela de faixas de valores. Este recurso é muito próximo a filtragem baseada em fusão visto que um índice passa a atender não só um valor mas uma faixa de valores próximos.

O modelo do sistema S-ToPSS não prevê técnicas de otimização para realizar o *matching*. Apenas aborda o uso de estruturas *hash* para localizar as informações necessárias (PETROVIC; BURCEA; JACOBSEN, 2003). Este modelo utiliza alguns mecanismos de transformação e avaliação que podem afetar a eficiência do algoritmo de *matching*. A alteração de sinônimos altera os dados da requisição original para que o algoritmo de avaliação possa trabalhar com um dicionário mais simplificado. Enquanto a avaliação de hierarquia e mapeamentos faz com que a complexidade de avaliação seja aumentada.

Para a avaliação ser mais eficiente no JTangPS (QIAN et al., 2008) o algoritmo é executado em dois estágios. O primeiro estágio avalia para cada propriedade o número de atributos que podem ser utilizados na comparação e o segundo estágio compara os tipos de informação e os conteúdos. Com estes dois estágios o número de atributos pode eliminar possíveis candidatos a avaliação do segundo estágio. Esta otimização foi descartada no JTangCSPS (QIAN et al., 2011) pois o processo de *matching* é feito por uma decomposição das regras e processado separadamente por vários elementos da rede. Este modelo é mais aproximado do método de decomposição, avaliação e junção dos resultados.

O modelo de Cao et al. (2010) utiliza os passos de processamento tradicionais do MapReduce, mapeamento e redução. Através desse modelo os autores puderam reproduzir o algoritmo Rete (FORGY, 1982) com processamento paralelo.

3.4 Síntese de características e comparação

Os trabalhos relacionados foram resumidos e comparados na Tabela 1. As características foram extraídas a partir dos modelos apresentados e auxiliam o entendimento geral de questões conceituais e técnicas de cada proposta. Algumas destas características já foram discutidas nas seções anteriores.

O paradigma utilizado na maioria dos trabalhos relacionados é o *publish/subscribe*. É possível observar que este paradigma proporciona diversos benefícios no mecanismo de processamento, graças ao fraco acoplamento temporal e espacial (EUGSTER et al., 2003). A composição de técnicas de filtragem, indexação, estratégias de roteamento e base histórica agilizam

Tabela 1: Comparação de características de trabalhos relacionados

Característica / Proposta	DREAM	S-ToPSS	JTangPS	JTangCSPS	Guo, Wei e Han (2008)	Cao et al. (2010)
Modelos de distribuição e processamento de informação	Pub/Sub	Pub/Sub	Pub/Sub	Pub/Sub	Pub/Sub	MapReduce
Método de comunicação	Cliente/ Servidor	Cliente/ Servidor	Cliente/ Servidor (SOA)	P2P	Cliente/ Servidor	Cliente/ Servidor
Representação de dados	MIX	Chave/Valor	RDF	RDF	Chave/Valor	Tupla
Informações avaliadas no *matching*	Eventos	Objetos	Eventos	Eventos	Eventos	Regras
Uso de filtros	Sim	Não	Sim	Sim	Sim	Não
Uso de índices	Sim	Sim	Sim	-	Sim	-
Uso de estratégias de roteamento	Sim	Não	Não	Sim	Sim	Não
Uso de base histórica	Não	Não	Não	Não	Sim	Não
Considera sinônimos	Não	Sim	Não	Não	Não	Não
Considera semântica	Sim	Sim	Sim	Sim	Não	Não
Fraco acoplamento	Sim	Sim	Sim	Sim	Sim	Não
Gatilho de execução	Requisição	Requisição	Requisição	Requisição	Requisição	Ciclo
Saída do algoritmo	-	Notificações	-	-	-	Regras

Fonte: Elaborado pelo autor

o processo de comparação tornando possível a execução do algoritmo de *matching* a partir de uma única requisição mais eficiente. Por outro lado, no MapReduce não é observada esta atenção a técnicas de otimização com relação a *matching*.

Entre as técnicas de otimização observadas estão o uso de filtros, o uso de índices, estratégias de roteamento entre os elementos distribuídos e o uso de base histórica. Estas técnicas são úteis aos sistemas de *matching* para eliminação de candidatos a comparação, evitando um processamento desnecessário de informação. O uso de sinônimos e análise semântica das informações também é importante para eliminar ambiguidades e variações naturais da linguagem humana.

A distribuição da aplicação implica em dois níveis de acoplamento. O fraco acoplamento do *publish/subscribe* permite que ocorra eventos assíncronos, onde o espaço e o tempo também são pouco dependentes nas operações (EUGSTER et al., 2003). Este acoplamento também permite que o sistema mude suas regras de comparação sem afetar a operações de todas as regiões onde o modelo se aplica. No caso do MapReduce não se observa esta mesma flexibilidade, as regras de *map* e *reduce* tendem a ser definidas desde o princípio.

As aplicações práticas utilizadas nos artigos são em sua maioria avaliadas quanto a desem-

penho mas não comparadas entre sí, pois são aplicados em problemas diferentes com propósitos similares de avaliação de equivalência. Algumas propostas avaliam informações para disseminar notificações de eventos, enquanto outros comparam objetos genéricos ou informações em estruturas muito específicas a um problema relacionado a equivalência de regras. Com o estudo destes trabalhos, observa-se a alta aplicabilidade de algoritmos de *matching* na resolução de diversos problemas relacionados a comparação entre informações.

Estes trabalhos contribuem nas decisões de projeto do serviço proposto e no modelo do serviço, definido no Capítulo 4. O serviço proposto neste trabalho pode tirar proveito do paradigma *publish/subscribe*, uma estratégia adotada pelos trabalhos que o utilizam para promover a distribuição do processamento. O método de comunicação P2P promove um fraco acoplamento de comunicação entre os elementos de cada solução. Este método pode eliminar um possível problema de gargalo de comunicação pelas características de escalabilidade naturais do próprio modelo. O uso de recursos de otimização como filtros podem ser associados a estratégias de roteamento para distribuição de carga entre os recursos de infraestrutura, enquanto o uso de índices pode acelerar o processo de busca de informações dentro da base de dados do serviço. Estas otimizações podem auxiliar o resultado final das comparações e podem ser usadas no serviço proposto. O gatilho de execução em um contexto de um serviço de identificação de oportunidades é o envio de uma oferta ou de uma demanda, e a saída do serviço é um conjunto de oportunidades relacionadas.

4 MODELO PROPOSTO

O modelo visa definir o comportamento, a estrutura e os componentes básicos do serviço, chamado de GTTracker, que permite a identificação de oportunidades de negócio. Estas oportunidades devem ser identificadas à partir da combinação de ofertas e demandas mantidas por empresas e compradores. Estas informações de ofertas e demandas são armazenadas em um conjunto de nós que se comunicam de forma organizada para atender as premissas do serviço.

Este Capítulo é subdividido em cinco seções, nas quais as duas primeiras seções apresentam as decisões de projeto e o comportamento geral do serviço para tornar clara a operação entre vendedor e consumidor. A terceira seção apresenta a organização geral do modelo e a hierarquia idealizada a fim de definir as funcionalidades e interações entre os nós que compõem o serviço. A quarta seção detalha a arquitetura dos nós e explica o comportamento de cada módulo que o compõe. Nesta mesma seção é feito um detalhamento do processo básico de *matching* para compreensão do mecanismo de comparação de objetos que representam as ofertas e demandas. A quinta seção detalha um estudo de caso que exemplifica a operação do serviço em um cenário real de operação.

4.1 Decisões de projeto

O projeto do serviço envolve algumas decisões que são relacionadas ao comportamento e requisitos abordados no decorrer do trabalho. Para este propósito algumas premissas foram elaboradas a fim de auxiliar o desenvolvimento do modelo deste serviço e são necessárias à avaliação funcional da proposta. As seguintes premissas são consideradas importantes para a elaboração do serviço relacionado a questão de pesquisa:

a) **Permitir a publicação de ofertas**: oferecer aos vendedores a possibilidade de registrar no serviço os objetos que descrevem suas ofertas. Estas ofertas são comparadas com as demandas já existentes e armazenadas para futuras comparações com novas demandas;

b) **Permitir a publicação de demandas**: oferecer aos compradores a possibilidade de registrar suas demandas. Estas demandas são comparadas com as ofertas já existentes e armazenadas para futuras comparações com novas ofertas;

c) **Ser mantida por empresas e seguir uma hierarquia de confiança baseada em regiões geográficas**: as empresas que oferecem seus produtos devem manter uma porção da infraestrutura do serviço. Uma porção da infraestrutura computacional responsável pela disponibilidade do serviço é formada por um ou mais nós computacionais do serviço. Cada nó é responsável por uma determinada região geográfica. Estes nós são associados para cooperar na comparação seguindo uma hierarquia. Esta hierarquia é baseada em concessões por região, estas concessões estabelecem uma relação de confiança na hierarquia. Esta confiança

é estabelecida uma vez que nós de autoridade podem conceder a um nó a responsabilidade por uma região.

d) **Infraestrutura elástica e distribuída**: o serviço precisa ter uma característica elástica para que o investimento inicial seja baixo e que possa suportar mais informações em uma proporção equivalente aos recursos investidos. O nó computacional do serviço deve ser modelado de maneira que possa utilizar recursos de computação em nuvem. Os nós que compõem o serviço também devem ser fracamente acoplados para que indisponibilidades ou mudanças na infraestrutura de determinada região não influenciem a disponibilidade geral do serviço;

e) **Permitir comparações através de atributos**: os objetos que definem uma oferta ou demanda devem ser compostos por um conjunto de atributos que definem suas características. Estas características podem ser alfanuméricas, numéricas ou conter outro conjunto de atributos, com outras informações que podem ser utilizadas durante a avaliação de *matching*;

f) **Permitir o uso da localização dos usuários para identificação de oportunidades**: o uso da informaçõe de localização do usuário deve ser considerada para a identificação das oportunidades. Esta informação associada ao atributos de oferta e demanda deve limitar o escopo de pesquisa no serviço e ser utilizada para filtrar as ofertas ou demandas candidatas à identificação.

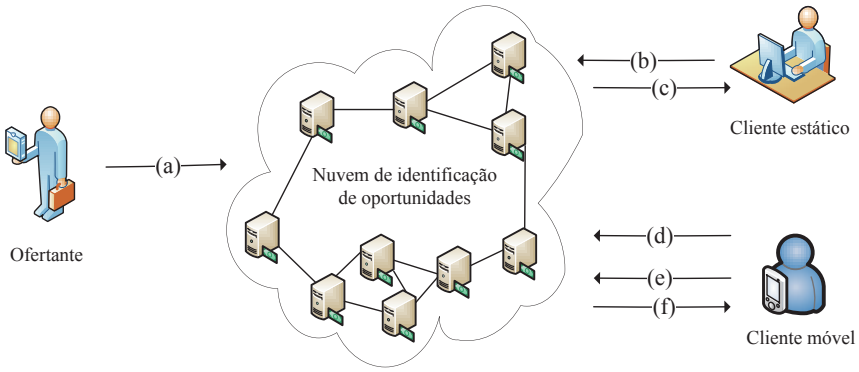
As premissas (a) e (b) correspondem aos principais casos de uso por parte do vendedor e do consumidor, respectivamente. As premissas (c) e (d) oferecem uma visão geral do funcionamento esperado do serviço e sua arquitetura. Enquanto as premissas (e) e (f) ressaltam as informações passíveis de utilização para avaliação de *matching* entre as ofertas e demandas. A premissa (f) é diretamente associada a premissa (c), pois através da localização e das respectivas regiões geográficas que se estabelecerá a distribuição do serviço (d).

4.2 Comportamento do serviço

O serviço propõe solucionar o problema de identificação de oportunidades entre vendedores e consumidores através de diferentes algoritmos de *matching* e oferecendo uma elasticidade baseada em recursos de computação em nuvem. Este *matching* é realizado por uma hierarquia de nós disponibilizados por regiões. Que atua como um *broker* do paradigma *publish/subscribe*, sendo que os vendedores publicam suas ofertas e os compradores inscrevem suas demandas.

Nesta interação com o serviço observa-se dois tipos distintos de compradores. Os compradores que naturalmente não possuem características móveis, chamados compradores estáticos, os quais possuem capacidade computacional e conectividade maior que a média, não mudam de canal ou endereço de comunicação com frequência e estão disponíveis para conexão a maior parte do tempo. E os compradores que utilizam dispositivos móveis, cuja capacidade de processamento computacional é limitada e que não mantém uma conexão estável e constante tanto por

Figura 8: Visão geral de uso da infraestrutura



Fonte: Elaborado pelo autor

problemas de qualidade de transmissão como por custo energético. Estes dispositivos móveis também são afetados pela própria mobilidade que pode alterar sua identificação ou dificultar a sua localização através de outros dispositivos.

A Figura 8 apresenta o comportamento geral do serviço. Os nós compõem o serviço de identificação e são disponibilizados em uma nuvem computacional. Os nós são acessados por usuários, cada usuário conecta-se ao nó do serviço responsável por sua região. Os usuários que desejam oferecer seus produtos podem enviar para a nuvem objetos que definem suas ofertas (a). Enquanto usuários que possuem demandas podem enviar para a nuvem objetos que definam suas demandas através de dispositivos móveis (d) ou não (b). Os nós desta nuvem devem então, se necessário, interagir com outros nós para identificar possíveis oportunidades de negócio. Estas oportunidades devem ser notificadas aos usuários utilizando uma tentativa de comunicação ativa (c), no qual o serviço tenta entrar em contato para notificação da oportunidade, ou de forma passiva, onde o comprador precisa se comunicar com o mesmo nó que enviou a demanda (e) e então recuperar as notificações (f) que não puderam ser entregues pela infraestrutura. O método passivo é proposto para que dispositivos móveis possam recuperar as suas notificações mesmo com a troca de endereço do dispositivo que inicialmente informou a demanda.

No estado inicial os nós estarão sem ofertas e sem demandas. Faz-se necessário que ou o vendedor informe sua oferta ou o consumidor informe sua demanda até o sistema armazenar informações suficientes para atingir a identificação de uma ou mais oportunidades. Uma vez que estes nós possuem informações suficientes, a identificação pode ocorrer em dois casos:

a) **Identificação de matching após publicação da oferta**: a demanda é recebida e comparada com as ofertas já armazenadas. Resultando em notificações ou resultados imediatos ao comprador se houver oportunidades;

b) **Identificação de matching após publicação da demanda**: a oferta é recebida e comparada

com as demandas já armazenadas, resultando em notificações ou resultados posteriores à demanda apresentada pelo comprador.

Também define-se neste momento a necessidade de caracterizar nestes objetos um prazo de vida (ou tempo de validade). Este prazo de vida permite identificar o tempo em que o comprador está interessado em adquirir e o tempo no qual o vendedor pode oferecer tal oferta. Além de contribuir para caracterização da oferta ou demanda, este prazo possibilita a manutenção da informação dentro dos nós, que podem por sua vez apagar os objetos que já não são candidatos a *matching*.

4.3 Organização hierárquica

O serviço utiliza-se da associação de nós computacionais para organização da operação. Esta organização é feita de forma hierárquica, onde cada nó é responsável por determinada região geográfica. Estes nós por terem responsabilidade por determinada região podem conceder sub-regiões dentro da sua região para outros nós. Há três tipos de nós envolvidos nesta hierarquia:

I) **Nó de autoridade principal**: responsável por estabelecer as concessões iniciais a uma ou mais autoridades regionais. Assim permitindo que estas definam e atribuam concessões a outros nós;

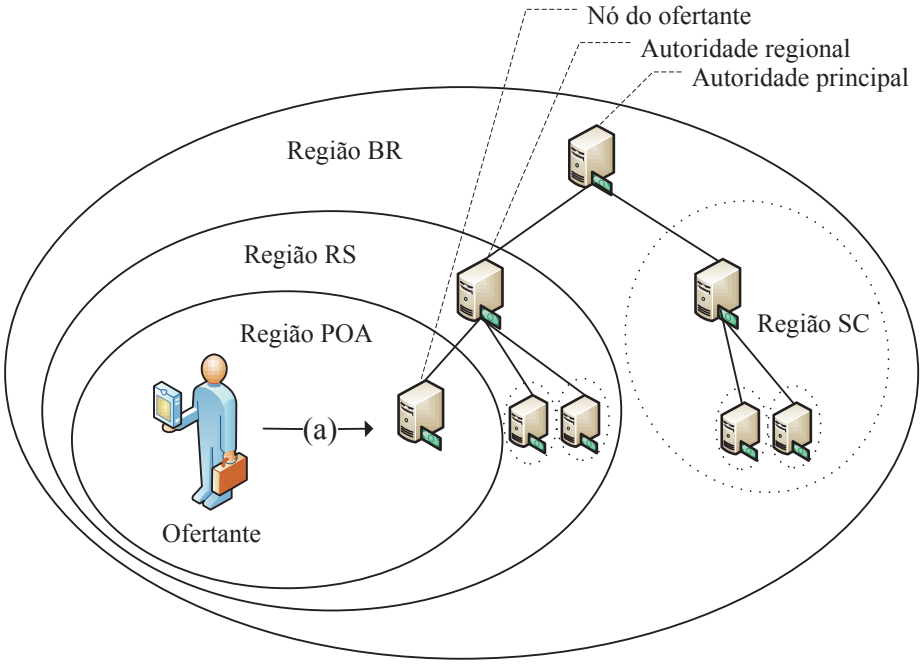
II) **Nó de autoridade regional**: encarregado de conceder e atribuir concessões a outros nós. Este nó pode conceder regiões geográficas não só a nós de ofertantes, como a outros nós de autoridade regional. Estes nós também são responsáveis por receber as demandas dos compradores, porém não precisam armazenar as demandas, apenas rotear para os nós de ofertantes correspondentes à região;

III) **Nó de ofertante**: responsável por armazenar de fato as ofertas disponibilizadas pelos vendedores e realizar as avaliações de *matching*. Estes nós também podem receber demandas de compradores de forma direta ou através de outros nós de autoridade regional.

Cada nó é mantido por uma determinada empresa, grupo ou associação. Este mesmo nó atende a determinada área geográfica e é atribuído a uma determinada área geográfica através de uma concessão disponibilizada por um nó de autoridade. Esta concessão permite que os nós se arranjem seguindo uma determinada hierarquia de confiança. Esta ligação de confiança permite assim associar, enquanto desconhecidos uns aos outros, através de um caminho pelo nó de autoridade.

A Figura 9 apresenta um exemplo de hierarquia e a disposição dos nós. Os vendedores sempre devem publicar suas ofertas em um nó de ofertante (a), correspondente aos nós folhas de uma hierarquia representada por uma árvore. Estes nós de ofertantes podem ser criados e

Figura 9: Distribuição da infraestrutura por regiões



Fonte: Elaborado pelo autor

ligados a um nó de autoridade regional desde que este conceda ao nó do ofertante uma determinada região geográfica. Este nó de autoridade regional pode conceder regiões apenas que estão sob sua concessão. Esta concessão por sua vez é definida por uma autoridade principal.

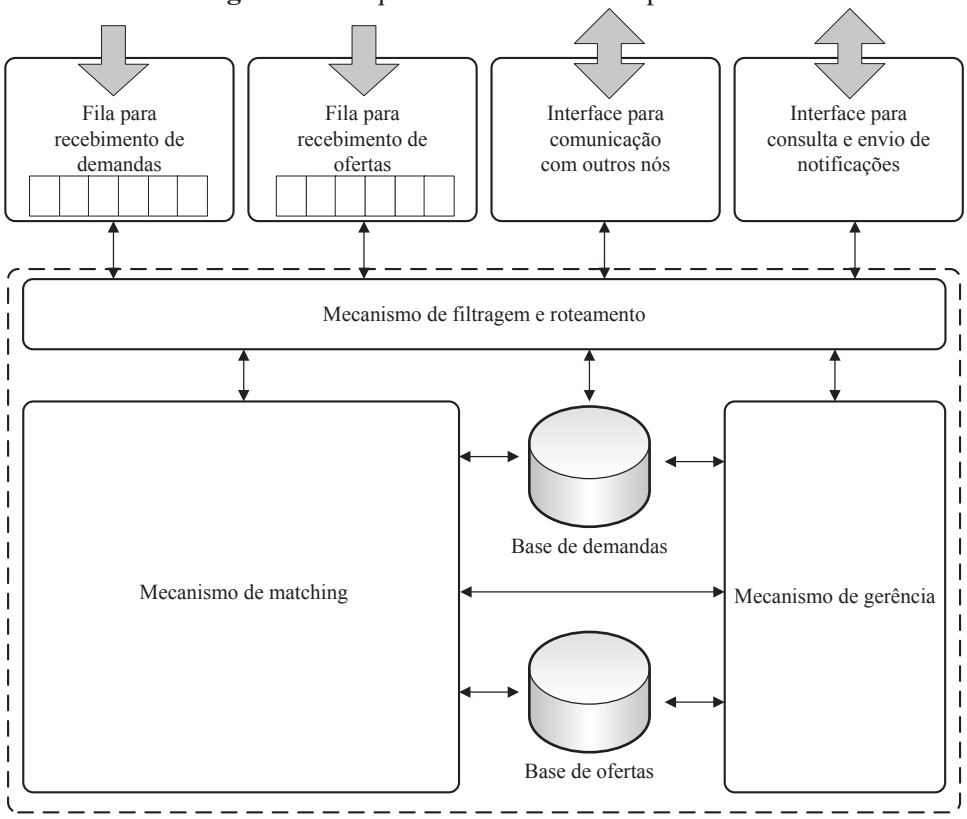
Propõe-se nesta hierarquia a distribuição dos nós por região geopolítica (país, estado, cidade), visto que normalmente já há entidades responsáveis por cada região geopolítica (administração nacional, estadual, municipal). Esta hierarquia baseada em regiões também idealiza um mecanismo transparente de elasticidade. Onde um nó ofertante ao ser muito requerido, possa adotar uma postura de nó de autoridade regional e então conceder suas sub-regiões a outros nós de ofertantes. Este cenário se aplica a por exemplo uma cidade que pode se expandir de forma desigual entre seus bairros. Um bairro central passa a exigir mais subdivisões que um bairro residencial.

4.4 Arquitetura dos nós

A arquitetura dos nós segue um princípio de aplicações P2P, onde um nó (*peer* do P2P) pode atuar como qualquer um dos três tipos (autoridade principal, autoridade regional ou nó de ofertante). Desta forma, um nó do tipo ofertante pode passar a atuar como nó de autoridade regional sem a necessidade de modificações complexas na sua arquitetura. Esta estratégia permite que um nó responsável por determinada região altere seu comportamento de acordo com a sua demanda de solicitações.

A Figura 10 demonstra visualmente a associação dos módulos de um nó computacional da

Figura 10: Arquitetura de um nó computacional



Fonte: Elaborado pelo autor

nuvem pertencente ao serviço. Estes módulos podem ser separados em dois grandes grupos. O grupo de interfaces e filas para comunicação externa e o grupo de mecanismos de roteamento, *matching* e gerência. O grupo de interfaces e filas disponibilizam meios de comunicação para recebimento de demandas e ofertas, também contando com duas interfaces bidirecionais que podem tanto receber como enviar notificações e objetos provenientes de outros nós da hierarquia. O grupo de roteamento, *matching* e gerência pode ser dividido em 4 componentes básicos:

a) **Mecanismo de filtragem e roteamento**: este mecanismo é responsável por filtrar os objetos de demandas e ofertas que não forem de sua concessão regional ou que pertencerem a mais de uma concessão. Através deste mecanismo os nós de autoridade regional podem encaminhar as requisições de demandas a outros nós de ofertantes para processamento do *matching*;

b) **Mecanismo de matching**: este mecanismo é composto de um ou mais plugins. Estes plugins são responsáveis pela interpretação dos objetos e pela identificação de oportunidade de fato. Cada região pode utilizar plugins específicos para atender necessidades regionais únicas. Alguns exemplos de plugins e as motivações para o uso de plugins serão explicados mais detalhadamente nas seções 4.4.1 e 4.4.3;

c) **Mecanismo de gerência**: o mecanismo de gerência é responsável por monitorar, manter e alterar o comportamento do nó de acordo com suas políticas. Sendo atribuído a este identificar e remover das bases de ofertas e demandas oportunidades que já expiraram, encontrar e estabelecer as rotas no mecanismo de filtragem e roteamento, e monitorar as notificações que já foram concluídas e assim as eliminar das listas de notificações pendentes;

d) **Bases de ofertas e demandas**: estas bases separadas, oferecem o armazenamento de ofertas e demandas para o mecanismo de *matching*. O mecanismo de *matching* após a análise dos objetos recebidos pode armazenar nestas bases os objetos enviados com apontamentos entre estes para caracterizar as oportunidades encontradas.

Quando uma demanda é recebida, o módulo de filtragem e roteamento avalia a região que deve atingir tal *matching* e então encaminha aos demais nós cópias da solicitação, com informações que indiquem a necessidade de retorno de informação ao nó de origem. As notificações resultantes nos demais nós precisam ser encaminhadas para o nó que recebeu a demanda inicial. Acumulando assim em um único nó da infraestrutura as oportunidades para tal demanda. O mesmo processo ocorre para a oferta. Em ambas situações é necessário respeitar o tempo de validade da requisição e a região a qual se aplica determinada oferta ou demanda. Os nós de autoridade podem limitar ou restringir as pesquisas de acordo com políticas de segurança previamente estabelecidas, evitando assim abusos por parte de algum dispositivo que possa estar afetando o desempenho do serviço.

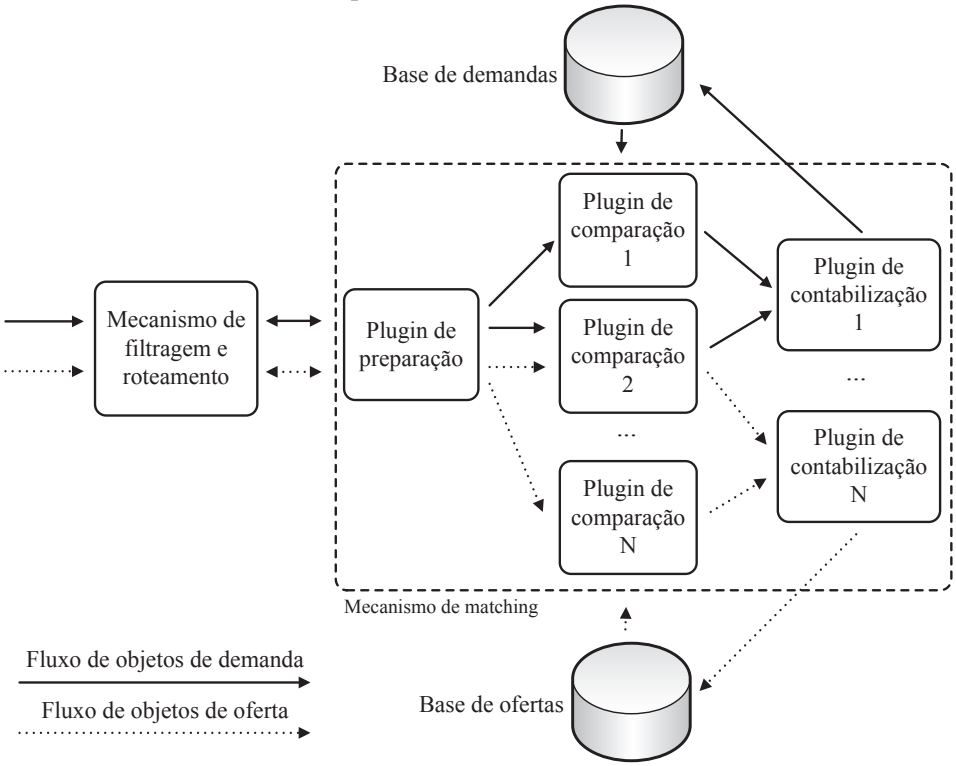
O mecanismo de filtragem e roteamento em um nó autoridade pode através de mecanismos mais elaborados particionar ou redistribuir as requisições entre vários nós de ofertantes em uma mesma região. Este particionamento pode separar as ofertas e demandas a partir do tipo de produto, por exemplo ofertas relativas a alimentos de ofertas relacionadas a imóveis. Ambos possuem uma frequência e volume de ofertas diferentes, o que demandaria uma infraestrutura computacionalmente mais robusta para comparações frequentes.

A organização dos nós em hierarquia auxilia o roteamento de objetos. Se um objeto de oferta ou demanda precisa atingir uma região maior, deve ser enviado para o nó de autoridade que está logo acima do nó que está roteando o objeto. Ao receber este objeto o nó de autoridade passa a avaliar o objeto e encaminhá-lo ao nó de autoridade superior se for necessário atingir uma área maior, ou a outros nós inferiores na hierarquia, que são responsáveis por sub-regiões do nó de autoridade que está roteando o objeto.

4.4.1 Mecanismo de matching

O mecanismo de *matching* é o responsável por processar cada oferta ou demanda e comparar as informações para identificar as oportunidades. O mecanismo de filtragem e roteamento é responsável por encaminhar ao mecanismo o objeto que representa uma oferta ou uma demanda. Este objeto é então processado em três estágios. Os três estágios são identificados na

Figura 11: Arquitetura do mecanismo de *matching*



Fonte: Elaborado pelo autor

Figura 11 e cada um destes estágios é composto por pelo menos um plugin. Há três tipos de plugins envolvidos processo de *matching*:

I) **Plugins do estágio de preparação**: responsáveis por preparar os objetos para comparação. Esta preparação pode reordenar os objetos para agilizar a comparação e também modificar seus atributos. A modificação dos atributos pode por exemplo realizar a troca de sinônimos nas palavras ou adaptar os valores monetários para a moeda corrente;

II) **Plugins do estágio de comparação**: acessam as bases de ofertas ou demandas em busca de possíveis oportunidades. Estes plugins podem por exemplo ler um lote de objetos e compará-los um a um ao objeto recebido. Este também pode assim que encontrar uma quantidade suficiente de resultados parar a comparação e encaminhar os resultados ao plugin de contabilização;

III) **Plugins do estágio de contabilização**: recebem os resultados e os próprios objetos da requisição e organizá-os nas bases de dados, também informando ao módulo gerente a necessidade de notificação em casos positivos de identificação de oportunidades. Estes plugins podem também coletar informações adicionais sobre os objetos recebidos e indicar ao mecanismo de filtragem e roteamento informações importantes para tomada de decisão.

Este trabalho não pretende definir nenhum mecanismo para comparação em um domínio específico, ou aplicar um algoritmo especialista de comparação. A estratégia adotada é de disponibilizar recursos a um algoritmo de comparação, que deve ser implementado como um plugin no estágio (II). O uso de três estágios garante ao serviço o uso de mais de um mecanismo de comparação, que pode ser desenvolvido posteriormente. Sabendo da necessidade de coleta de mais informações para alguns mecanismos de comparação, adotou-se o uso de um plugin de preparação e um de contabilização. O plugin de preparação pode coletar ou ajustar informações gerais dos objetos. Enquanto o plugin de contabilização pode coletar informações gerais de acordo com o comparador utilizado.

Diferentes combinações de plugins podem resultar em diferentes resultados de identificação, tanto em número de oportunidades como em velocidade de comparação. O plugin de preparação pode, além de alterar os objetos para comparação encaminhá-los a plugins de comparação independentes. Esta estratégia permite separar comparadores de acordo com um determinado atributo do objeto. Um plugin de preparação que analise o tipo do produto, por exemplo pode encaminhar objetos que representam ofertas do tipo "produto" para um plugin de comparação e do tipo "serviço" para outro. Após a comparação, cada plugin pode encaminhar o objeto a um plugin de contabilização diferente.

4.4.2 Objetos de comparação e metadados

Os objetos manipulados no modelo seguem uma estrutura baseada no formato JSON[1] (CROCKFORD, 2006). Este formato pode representar tanto as ofertas como as demandas, tendo como diferença apenas o nome do atributo inicial e a obrigatoriedade de determinados metadados que são relevantes em cada caso. Um exemplo do formato pode ser observado na Figura 12, com os atributos mais relevantes enfatizados. Este formato permite a representação das mesmas informações existentes no MIX, sendo mais simples estruturalmente que o XML e RDF. Este formato também é reconhecido pela maioria das linguagens modernas de programação e não adiciona informações extensas de marcação.

As ofertas são caracterizadas pela palavra "oferta" enquanto as demandas são caracterizadas pela palavra "demanda". Ambos objetos possuem um atributo chamado "meta", no qual determinam as informações gerais utilizadas no controle dos objetos no nó e na comunicação entre estes. Estas informações de metadados são transferidas integralmente entre os nós, não sendo removidas durante a migração de um objeto de um nó para outro. As notificações de oportunidades são representadas por objetos de demanda, porém com o atributo "ofertas" contendo uma coleção de objetos de ofertas identificadas como passíveis de realização de negócio. Os metadados gerais, aplicáveis tanto a ofertas como demandas são:

- **Validade**: no qual representa o prazo final de validade do objeto;

[1] http://www.json.org

Figura 12: Exemplo de objeto de oferta

```
{"oferta":{
        "nome":"Cartucho de tinta",
        "tipo":"Suprimento",
        "cor":"Preto"
        "impressora":{"marca":"XP","modelo":"ABC01"},
        "valor":50.00,
        "meta":  {
                "validade":"30/12/2012",
                "ofertante":"XYZ Suprimentos LTDA",
                "latitude": 30.005,
                "longitude": 40.213
                "raio":"0.2"
                }

        }
}
```

Fonte: Elaborado pelo autor

- **Latitude** e **Longitude**: no qual representa o ponto central geográfico em graus decimais de onde o objeto se encontra ou espera-se encontrar;

- **Raio**: no qual designa qual é a distância máxima aceitável do ponto central esperado. A unidade de medida é em graus decimais, o mesmo usado na latitude e longitude.

Os atributos de localização são fundamentais para que o mecanismo de filtragem e roteamento localize o nó adequado à consulta de oportunidades. Os demais atributos existentes nos objetos podem variar de acordo com o tipo de produto ou serviço oferecido. Os plugins usados na comparação devem se adequar ao tipo de produto e idealiza-se que algum órgão ou associação especializada no serviço ou produto defina quais são os atributos esperados para cada tipo. Esta definição pode influenciar o desenvolvimento dos plugins e também proporcionar informações relevantes à combinação.

Este trabalho não adota uma representação rígida de objeto, pois é difícil garantir uma qualidade alta e uniforme de dados por parte dos usuários. O modelo exige apenas o uso dos atributos de metadados para que o serviço possa organizar o processamento do *matching*. A qualidade de dados dos atributos que definem um objeto irá refletir diretamente na identificação de oportunidades. A região dos usuários pode causar variações na representação de dados. Informações relevantes em uma determinada região podem não ser relevantes em outra.

4.4.3 Plugins para matching

O uso de plugins no módulo de *matching* tem como principal objetivo permitir o desenvolvimento futuro de novos plugins de acordo com a região e objetos que são comparados. Esta estratégia permite uma evolução do serviço sem alterações significativas no comportamento do modelo ou dos nós computacionais. Neste trabalho é proposto o uso de plugins em três estágios: preparação, comparação e contabilização. Para exemplificar a utilização destes plugins propôe-se o uso de três plugins básicos, um para cada estágio do módulo de *matching*. Estes plugins podem ser utilizados como base para formulação de outros plugins. No protótipo foram implementados três plugins, um para preparação, outro para comparação e um responsável pela contabilização. Um exemplo mais elaborado para o estágio de preparação é um plugin de sinônimos ou um plugin que normalize as palavras que descrevem o objeto. No estágio de comparação implementado neste trabalho utilizou-se um plugin comparador de atributos para objetos que representam uma oferta ou demanda de serviço. Para o estágio de contabilização, o plugin permitiu contabilizar a eficiência do nó e colher estatísticas de comparação.

O funcionamento de um plugin de sinônimos permite normalizar palavras equivalentes para um nome comum. O plugin pode utilizar um dicionário de palavras correspondentes, contendo a palavra mais significativa e os respectivos sinônimos. Nesta situação uma sigla como “TV” pode ser convertida para “Televisão” e então ser facilmente combinada com uma oferta cuja publicação foi feita utilizando como parte do nome “Televisão”. Esta avaliação pode ser mais elaborada considerando a questão semântica e os demais atributos do objeto, porém demandam técnicas mais avançadas que não são objetivos deste trabalho. Este plugin também é responsável por identificar entre dois tipos diferentes de oferta ou demanda: produtos e serviços. De acordo com o tipo de objeto identificado, produto ou serviço, este plugin encaminhará o objeto ao plugin de comparação correspondente.

Para a comparação, o plugin responsável por este estágio avalia um conjunto de atributos do objeto que podem ser equivalentes ou qualifica-los como aproximados. Este algoritmo pode ser desenvolvido de acordo com o tipo do objeto. Para comparação de televisões, por exemplo, é importante que a oferta e a demanda compartilhem do mesmo tamanho de tela, a mesma marca e o mesmo tipo de tecnologia se estas informações estiverem disponíveis na demanda. Para um serviço, o objeto deve ser comparado de acordo com os benefícios esperados por parte do consumidor. Outra premissa é relativa a presença de informações no objeto de demanda. As informações que não forem definidas na demanda são consideradas menos relevantes no *matching*, partindo do fato que o comprador não considerou estes atributos em sua demanda e portanto não considera o atributo relevante.

Após a comparação o plugin responsável por realizar a qualificação de *matching* encaminha o objeto avaliado e suas respectivas oportunidades para o estágio de contabilização. O plugin responsável por este estágio deve contabilizar informações relevantes aos demais plugins e módulos do nó. Uma utilidade deste mecanismo é por exemplo avaliar quais são os

atributos mais utilizados. Através desta quantificação o algoritmo de comparação pode alterar sua ordem de avaliação, assim eliminando os objetos mais facilmente com comparações em seus atributos menos utilizados. Este plugin também pode disponibilizar ao módulo gerente a eficiência de identificação de objetos recebidos. Esta eficiência é calculada pela razão entre objetos comparados e resultados obtidos.

A possibilidade de alteração do mecanismo de *matching* é uma característica possível devido ao fraco acoplamento dos nós. Este fraco acoplamento permite que em algumas regiões determinados plugins sejam desejados, e em outras regiões o plugin seja dispensável. Esta customização também permite que nós especialistas comparem tipos específicos de produtos com maior qualidade. Esta especialização pode ser realizada em conjunto com o módulo de roteamento, que pode filtrar e encaminhar os objetos de determinados tipos a determinados nós computacionais.

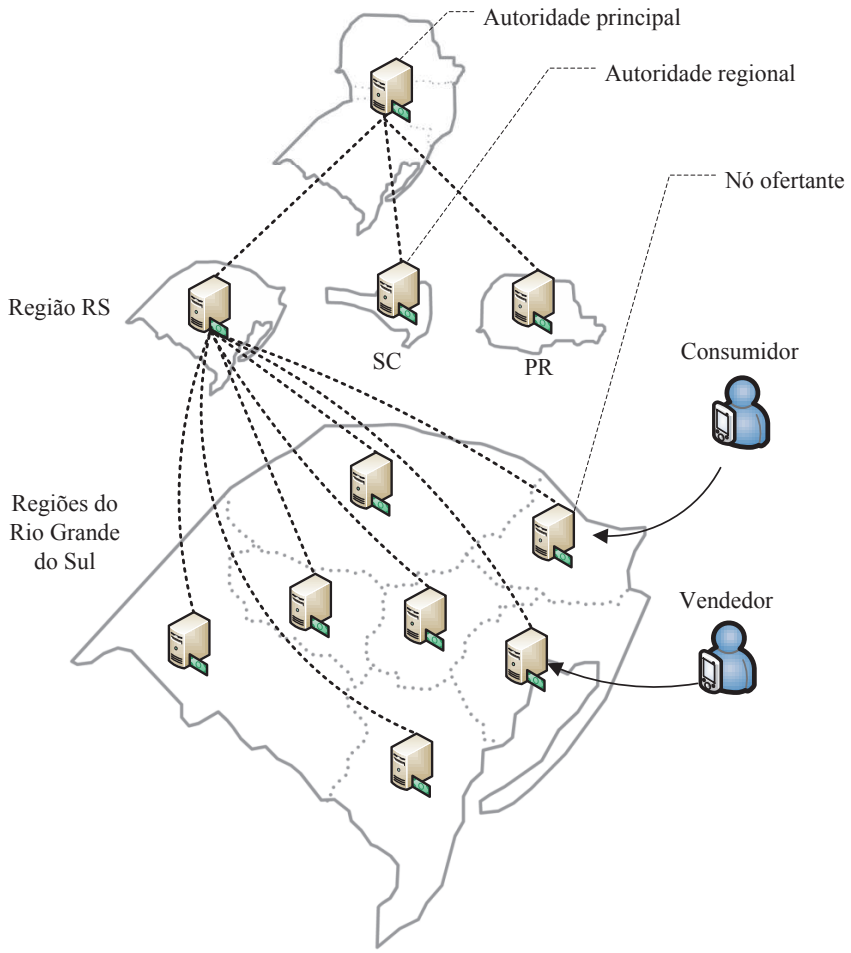
4.5 Estudo de caso

Um estudo de caso pode tornar mais clara a funcionalidade oferecida pela implementação do modelo. Este estudo apresenta como o serviço pode ser utilizado em um determinado cenário comercial. Com o desenvolvimento integral do modelo idealiza-se que este serviço atue como um *broker* entre compradores e vendedores. O *broker* é distribuído através de regiões e permite que os vendedores presentes em tal região comuniquem a disponibilidade de tais ofertas. Desta mesma forma os compradores informariam suas demandas. Disponibilizando assim ao serviços informações necessárias para que este atue de forma pró-ativa para identificação de uma oportunidade de negócio.

A Figura 13 apresenta uma proposta de distribuição na região sul do Brasil. Esta distribuição foi feita a partir dos limites geopolíticos para demonstrar a escalabilidade oferecida pela proposta. Considere que a Figura 13 evoluiu de um cenário inicial, onde apenas um nó computacional era necessário para toda região sul. Este único nó atuava como um nó de ofertante, onde todos os usuários se conectavam para publicar suas demandas e ofertas. Em um segundo momento, o serviço acabou demandando um poder computacional maior. Neste caso foram associados três novos nós computacionais. Estes novos nós passaram a realizar o processo de identificação de oportunidades como nós de ofertante, enquanto o nó de toda a região sul passou a ser o nó de autoridade principal, que concedeu aos outros nós determinadas regiões. Em um terceiro momento, a região sul passou a demandar mais poder computacional, necessitando de 7 novos nós para processamento por regiões dentro do estado. Nesta situação o nó de ofertante do estado do Rio Grande do Sul passou a atuar como um nó de autoridade regional. A situação da hierarquia de nós final é apresentada na Figura 13.

Neste cenário resultante o vendedor que estivesse na região metropolitana de Porto Alegre realizaria uma comunicação com o nó responsável por tal região informando a sua oferta. O serviço então analisaria através de *matching* possíveis demandas que se encaixem com a sua

Figura 13: Proposta de distribuição de nós em um estudo de caso que abrange o sul do Brasil



Fonte: Elaborado pelo autor

oferta. Se este nó da região metropolitana não for capaz de encontrar oportunidades, este se encarrega de encaminhar a mesma solicitação para o seu nó de autoridade, para que este encaminhe a oferta aos demais nós que estejam dentro do raio desejado. O mesmo aconteceria se o comprador estivesse informando uma demanda. Este nó também é responsável por agrupar possíveis oportunidades desta requisição após análise dos outros nós. Por fim, o vendedor seria notificado das oportunidades de forma ativa ou passiva pelo nó que este inicialmente informou a oferta.

Esta situação pode se aplicar a usuários móveis, que podem trocar de regiões durante o dia, e para usuários de dispositivos estacionários, mas estão localizados em determinada região. Para usuários móveis o serviço pode propiciar um meio confiável de análise de ofertas, pois a organização dos nós precisa seguir determinada hierarquia. Nessa hierarquia a autoridade principal agrega nós de autoridade regional e estes garantem a autenticidade dos nós de ofertantes para cada região. Esta concessão de regiões pode ser organizada por uma entidade do setor de comércio ou alguma empresa que incentive o uso do serviço. O uso de recursos de computação

em nuvem permite que cada nó tenha seu custo associado a sua demanda e não torna os custos de infraestrutura uma barreira para implantação do serviço.

5 IMPLEMENTAÇÃO

Para avaliação do modelo de publicação e comparação de ofertas e demandas proposto foi desenvolvido um sistema composto de um conjunto de aplicações. Estas aplicações atendem a funcionalidade idealizada pelo modelo, não visando a aplicação direta em produção, com todas as otimizações de aplicação necessárias, e sim a avaliação do modelo implementado, para o serviço chamado de GTTracker. Para atingir a operacionalização do serviço um sistema é composto de três aplicações correspondentes a três componentes básicos do modelo. Neste Capítulo serão apresentados estes três componentes, a arquitetura do conjunto de aplicações, a tecnologia utilizada e o ambiente utilizado para operacionalização do serviço.

Os componentes utilizados na avaliação são aplicações independentes que colaboram entre sí para a operação do serviço. Estas aplicações podem ser consideradas independentes por terem atividades singulares:

I) **Sistema de localização**: Responsável por mapear o nó de serviço mais próximo ao usuário do serviço. Esta aplicação não é necessária para os testes em laboratório, visto que os nós de cada região são facilmente mapeados. Porém para um ambiente real, esta aplicação é imprescindível para localização do nó de serviço mais próximo ao usuário do serviço;

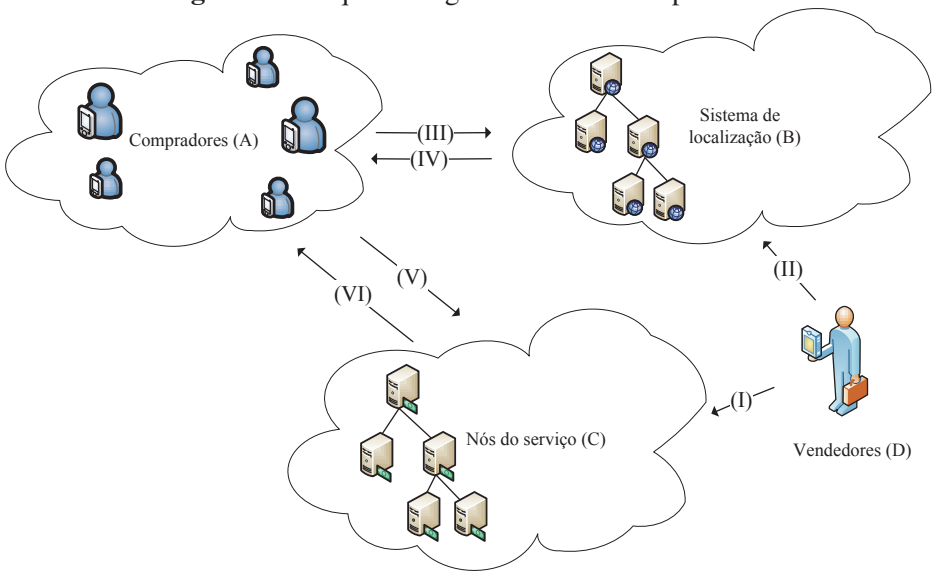
II) **Nó de serviço**: Encarregado de realizar a comparação entre os objetos de oferta e demanda, também sendo o responsável pelo encaminhamento dos objetos entre os nós que possuem autoridade para cada região. Na prática, esta aplicação implementa a principal porção de lógica de negócio atribuída ao conjunto de aplicações;

III) **Aplicação cliente**: Responsável por ser a interface de ligação entre os usuários e o serviço. Nela podem ser informados objetos de demanda, através de uma interface de busca. Nos testes realizados, utiliza-se uma aplicação Web para generalizar o uso da aplicação em qualquer dispositivo móvel.

Estas aplicações trabalham de forma independente para que possam ser executadas em servidores separados, e com suas próprias características de escalabilidade. Entende-se características de escalabilidade as estratégias de distribuição que permitem o uso em larga escala. O sistema de localização (I) é baseado em DNS, na qual já é implementado em escala global para viabilização da Internet utilizando uma estratégia de subdivisão em zonas. O nós de serviço (II) possuem subdivisões hierárquicas para serem capazes de uma subdivisão por regiões. A aplicação cliente de serviço (III) possui características de sistemas web independentes dos serviços de DNS e serviço, assim sendo livremente replicados para atender uma maior demanda. Na prática, a aplicação cliente pode ser qualquer aplicação capaz de realizar requisições HTTP, que possa ser executada em um dispositivo móvel e que possa se comunicar através da Internet.

As seções a seguir pretendem expor com mais detalhes a arquitetura utilizada na implementação, tornando um pouco mais clara as características tecnológicas empregadas, os algoritmos

Figura 14: Arquitetura geral do sistema implementado



Fonte: Elaborado pelo autor

utilizados no nó de serviço, e o ambiente utilizado na operacionalização do sistema. Com estes detalhes pretende-se tornar a avaliação mais clara e facilitar o entendimento dos testes realizados.

5.1 Arquitetura

A arquitetura distribuída das aplicações envolvidas facilita a disponibilização do serviço em diversos servidores. Cada aplicação pode ser compreendida inicialmente como uma nuvem. Uma nuvem computacional corresponde a um grupo de dispositivos de propriedade de um indeterminado grupo de usuários compradores. Outra nuvem correspondente ao serviço de localização dos nós de serviço. E por fim, uma nuvem correspondente aos nós de serviço. O modelo proposto neste trabalho se foca nas necessidades de processamento do conjunto de nós de serviço, os quais realizam a comparação entre as ofertas e as demandas e provém de fato o serviço de identificação de oportunidades. As atividades que cooperam para o objetivo ser atingido, como a localização e a aplicação cliente, serão detalhadas nas subseções 5.1.1 e 5.1.2.

A interação entre os três componentes do sistema pode ser observada na Figura 14. Uma massa de usuários do serviço (A) interagem com o sistema de localização (B) e posteriormente com a nuvem do serviço de identificação de oportunidades (C). Aos vendedores (D) atribuem-se as primeiras tarefas, criando nós de comparação (I) e os registrando no sistema de localização (II), desta forma os ofertantes podem publicar seus objetos de oferta. Após o processo inicial de publicação, os compradores (A) solicitam ao sistema de localização (III) o endereço do nó de serviço correspondente a sua região, ao receber o endereço (IV) o comprador pode encaminhar objetos de demanda (V) e receber objetos de ofertas correspondentes a sua demanda (VI). A

implementação deste trabalho atende exatamente este processo, sendo a implementação do nó de serviço a mais explorada nas avaliações. Nas próximas subseções serão detalhadas as aplicações implementadas para avaliação neste trabalho. Os objetivos, justificativas e resultados das avaliações serão expostos na Seção 6.

5.1.1 Aplicação cliente

A aplicação cliente foi implementada de duas formas distintas para ser aplicada em duas situações. A primeira situação corresponde a utilização do serviço por usuários reais. Esta aplicação fora desenvolvida para avaliar o comportamento de usuários e suas peculiaridades de pesquisa. Nesta situação uma aplicação Web fora desenvolvida com uma interface simplificada, permitindo ao usuário identificar-se, e informar uma determinada demanda (um objeto). Após cada pesquisa, o usuário pode avaliar os resultados obtidos e classificar cada resultado. Ao final da sequência de pesquisas informando demandas, o usuário precisa avaliar a qualidade do serviço de pesquisa como um todo através de duas questões breves.

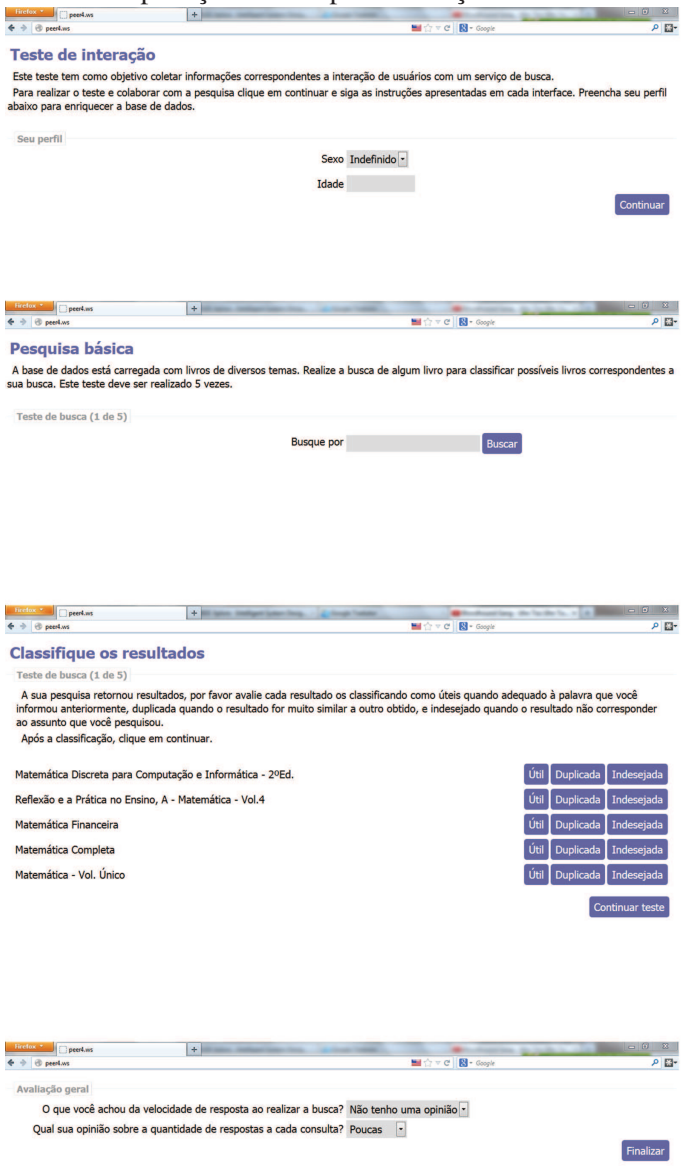
A Figura 15 exibe as interfaces da aplicação cliente utilizada para avaliar o comportamento dos usuários. A primeira interface (A) explica ao usuário a finalidade da aplicação. Nesta mesma interface o usuário informa seu gênero e idade ao sistema. Na interface de pesquisa (B) o usuário recebe a explicação sobre o domínio dos objetos existentes na base de dados e uma breve instrução sobre a pesquisa. Ao realizar uma pesquisa nesta interface, a aplicação Web cria o objeto de demanda e o encaminha para algum nó do serviço, este nó é escolhido com base no mapeamento de regiões fictício utilizado nos testes. Ao retornar os resultados, o usuário é encaminhado para a página de classificação dos resultados (C). Nesta interface o usuário avalia a qualidade das ofertas encontradas, ao finalizar o usuário clica para continuar o teste. Este mesmo teste é realizado repetidas vezes. Ao final o usuário avalia o sistema quanto a sua velocidade e quanto ao número de resultados encontrados (D).

A segunda situação corresponde aos testes de desempenho, e utilizam palavras armazenadas nas pesquisas efetuadas na primeira implementação da interface de cliente. Desta forma aproximando as pesquisas ao comportamento possível dos usuários. Nesta situação foram criados scripts para simular uma sequência de chamadas aos nós do serviço. Cada um dos scripts possui comportamentos distintos:

a) **Requisições sequenciais**: Esta aplicação lê a lista de palavras de um arquivo de texto e cria objetos de demanda, encaminhando para o nó de serviço correspondente a uma região em específico ou nós de regiões aleatórias;

b) **Requisições paralelas ao sistema**: Esta aplicação realiza a mesma atividade da aplicação de requisições sequenciais, porém permite a execução de duas ou mais requisições paralelas através de um mesmo comando de script. Para realizar a atividade de forma multiprocessada a aplicação duplica-se a si mesmo utilizando a chamada de sistema operacional `fork`;

Figura 15: Interfaces da aplicação cliente para avaliação do usuário em um navegador desktop



Fonte: Elaborado pelo autor

c) **Requisições concorrentes a um único nó**: Esta aplicação comporta-se de forma muito parecida à aplicação de requisições paralelas, porém um dos processos tem como alvo apenas um nó, enquanto os demais têm como alvo todos os demais nós de serviço. Esta aplicação é utilizada para avaliar o comportamento de uma carga específica em um nó, enquanto há cargas em outros nós.

Os scripts que realizam (a), (b) e (c) atuam como clientes que geram apenas demandas, ou seja, compradores. Uma aplicação cliente que simula um usuário ofertante também foi implementada. Desta forma, foi possível carregar a base de dados dos nós a partir de listas de produtos. Esta aplicação lê arquivos no formato texto que possuem o nome do produto, um identificador de loja e o preço do produto. Criando objetos de oferta e conectando em um respectivo nó para publicação. Todos os scripts interagem diretamente com a aplicação responsável pela localização dos nós.

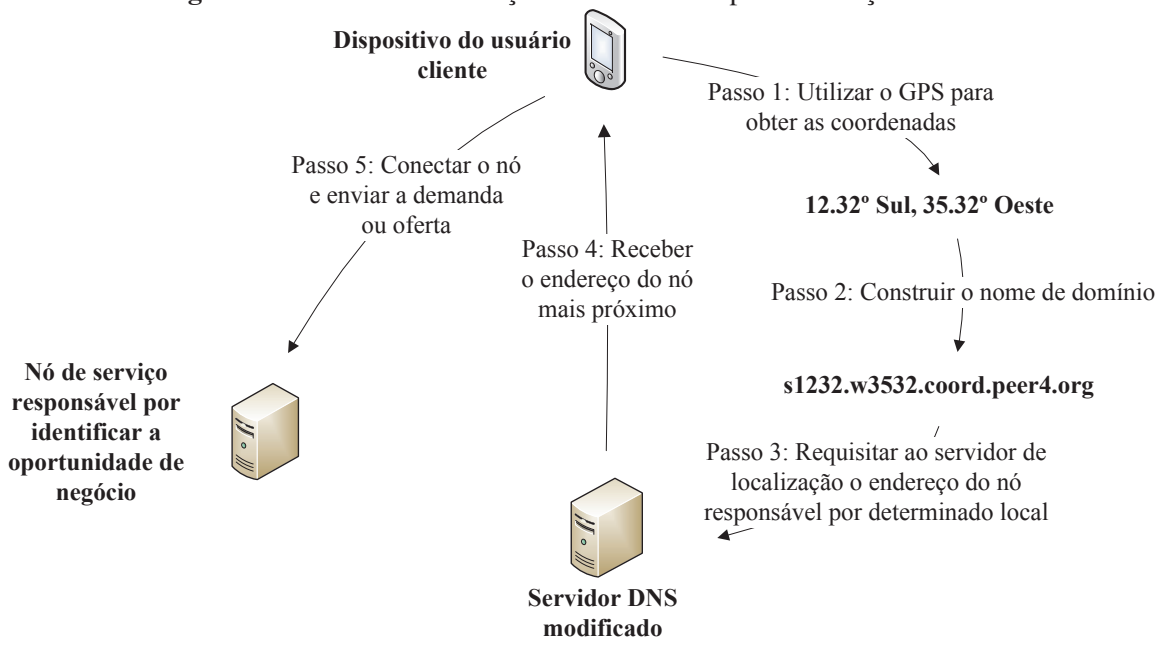
5.1.2 Mecanismo de localização de nó

O mecanismo de localização dos nós auxilia a interação entre os clientes e os nós de serviço. A função dele é traduzir as coordenadas geográficas de determinado usuário e retornar a este o endereço IP do nó de serviço no qual determinado objeto de oferta ou demanda pertence. A implementação do mecanismo de localização é derivada do mecanismo de resolução de nomes DNS. Nesta implementação o protocolo DNS é utilizado para realizar consultas a endereços que seguem determinados formatos e no qual a aplicação servidora DNS pode decodificar o endereço para determinado endereço registrado em sua base de dados.

A consulta utilizando o DNS permite que através das coordenadas geográficas a aplicação cliente conecte com o servidor de nós sem precisar de nenhuma biblioteca especial, apenas resolvendo o nome de domínio. Este mecanismo é abstraído na maioria das linguagens de alto nível, como é o caso das aplicações implementadas neste trabalho. Para tal, o nome de domínio a ser conectado deve respeitar o formato `LATITUDE.LONGITUDE.coord.DOMINIO.COM`. Onde `LATITUDE` e `LONGITUDE` são os graus decimais codificados de forma especial para evitar consultas ilimitadas ao servidor de DNS. No caso de latitude, por definição, o primeiro caractere deve ser `s` ou `n` para sinalizar respectivamente a coordenada negativa ou positiva. Na sequência tem-se o valor da coordenada em graus traduzidos para um valor inteiro, o grau é multiplicado por 100 e ignora-se a porção fracionária restante. O mesmo acontece para a longitude, porém utilizando `w` ou `e` se for, respectivamente, a coordenada negativa ou positiva.

O processo de localização de um nó de serviço é resumido na Figura 16. No exemplo, o dispositivo utiliza o GPS para obter as coordenadas geográficas aproximadas. Através das coordenadas obtidas cria-se o nome de domínio completo, neste caso é utilizado `peer4.org` como domínio principal. Consultando este nome em um servidor responsável pelo domínio será obtido o endereço do nó mais próximo. Permitindo assim a conexão ao nó de serviço mais próximo.

Figura 16: Processo de tradução de coordenada para endereço do nó



Fonte: Elaborado pelo autor

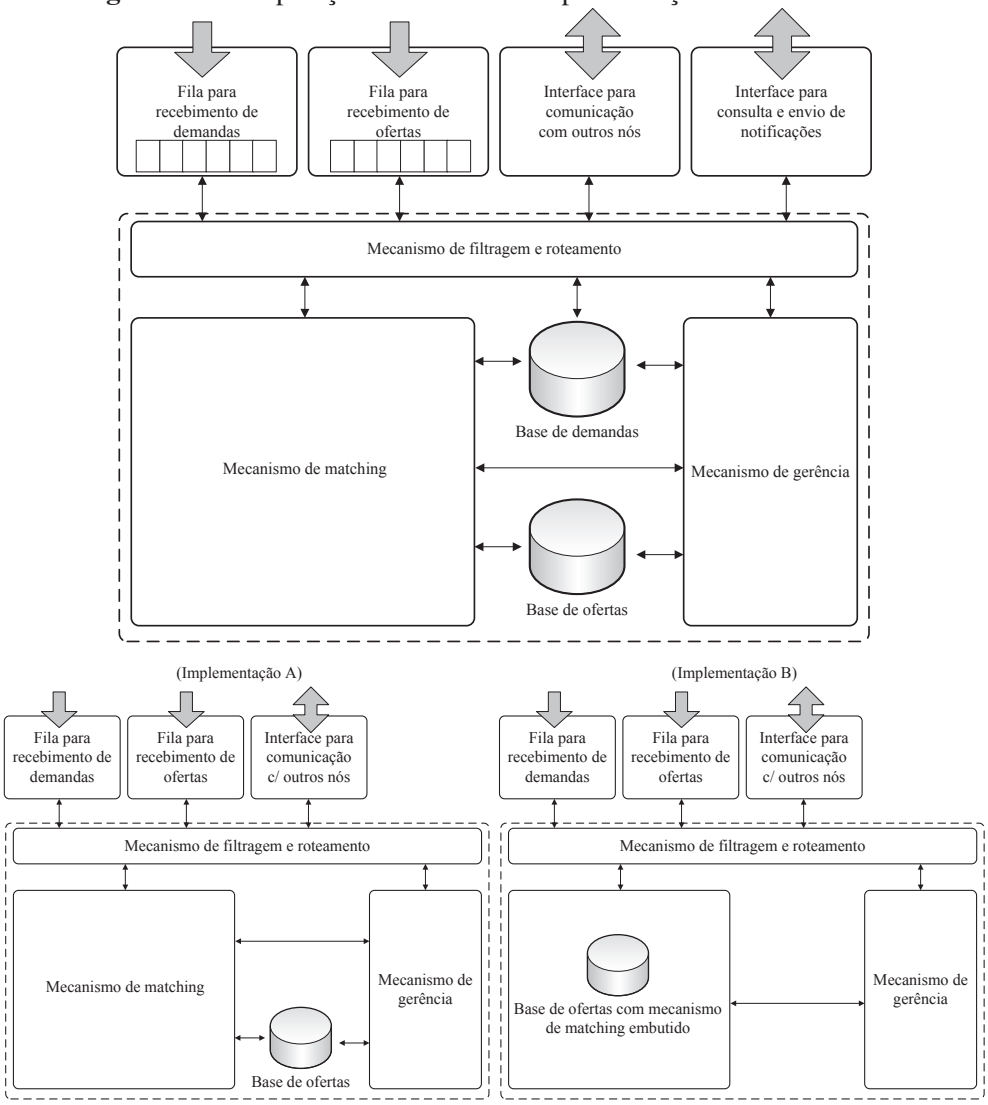
A implementação do mecanismo de localização se difere de um servidor DNS convencional pelo seu formato de armazenamento de dados. O uso de duas casas decimais obrigaria o cadastro de muitas entradas DNS em um servidor convencional, por esta razão, o servidor DNS precisa conhecer a codificação de coordenadas e armazenar regiões ao invés de nomes literais. Utilizando-se desta estratégia a gerência de nomes é simplificada, e permite a uma entidade organizadora gerenciar e delegar determinadas regiões a um determinado fornecedor.

Com a atribuição a uma entidade organizadora de gerenciar as regiões, permite-se a concorrencia entre fornecedores de nós de serviço. A idealização de concorrência e possível consequência dessa concorrência, como diminuição de custos e barateamento da manutenção do sistema não é foco deste trabalho, porém permite a realização de avaliações futuras sobre a questão. Este mecanismo é genérico e também pode ser utilizado em outras situações relacionadas a localização de servidores de serviço baseados na posição do usuário em um espaço geográfico bidimensional, tendo inicialmente apenas a coordenadas geográficas do dispositivo cliente.

5.1.3 Nó de serviço

O serviço de identificação de oportunidades é composto por uma estrutura hierárquica de nós de serviço, como exposto da Seção 4.3. Essa hierarquia é formada por nós idênticos em sua programação, que podem atuar de duas formas diferentes porém possuindo os mesmos mecanismos de comunicação com as aplicações dos clientes. A implementação para avaliação do trabalho considera apenas a porção principal da arquitetura apresentada no modelo, onde o

Figura 17: Comparação entre as duas implementações ao modelo base



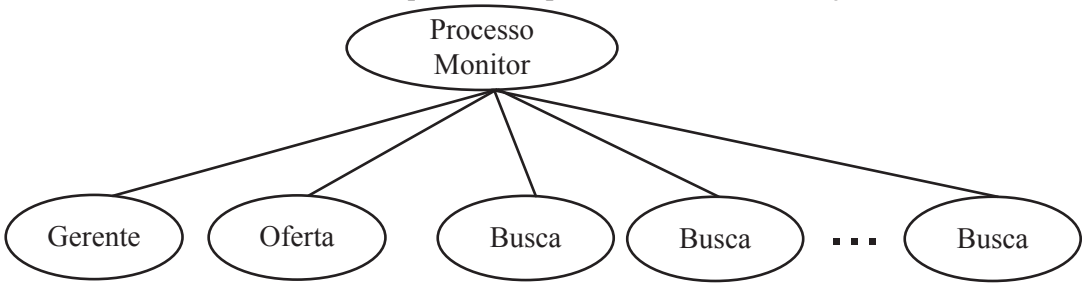
Fonte: Elaborado pelo autor

comprador informa a demanda e recebe um conjunto de ofertas compatíveis com sua demanda na região onde está.

A arquitetura da aplicação obedece o modelo base. Não foram implementados para a avaliação o módulo de interface para consulta e envio de notificações assíncronas e houve também a implementação de dois mecanismos de comparação, um baseado em plugins e outro baseado em banco de dados. As demandas não são armazenadas por não ser avaliado neste trabalho o mecanismo reverso de identificação de oportunidades, correspondente a oferta posterior a publicação da demanda. Na Figura 17 é possível observar a diferença entre as duas implementações realizadas (A e B) com o modelo base (posicionado acima dos dois modelos).

A implementação A utiliza um modelo de base de dados baseado em listas em memória RAM, para criação de plugins de comparação conforme o modelo apresentado na Seção 4.4.3. Nesta implementação há apenas um plugin de preparação, um plugin de comparação e um plugin de contabilização. O plugin de preparação apenas confere se todos os atributos necessários

Figura 18: Arquitetura de processos do nó de serviço



Fonte: Elaborado pelo autor

para a comparação existem. O plugin de contabilização apenas gera estatísticas para a avaliação. O plugin de comparação, por sua vez, é mais elaborado, utilizando um mecanismo de pontuação para classificar os objetos mais promissores. A classe que define o algoritmo do plugin de comparação está disponível no Anexo A. Este algoritmo utiliza tanto comparação literal como comparação fonética, utilizando a função `Soundex`, descrita na Seção 5.2.

A implementação B utiliza um modelo de base de dados baseado em banco de dados externo. Este mecanismo permite terceirizar o processo de *matching* em um serviço externo. Para esta implementação utilizou-se o banco de dados MySQL, que possui internamente um mecanismo de *matching* baseado em índices `FULLTEXT`. Este índice auxilia o algoritmo de *matching* interno do banco de dados, que separa cada palavra de um texto e classifica sua relevância na comparação de acordo com o número de ocorrências em todos os textos indexados (WIDENIUS; AXMARK; AB, 2002). Neste caso não foram implementados os plugins pois a aplicação de banco de dados realiza as otimizações necessárias e retorna estatísticas através de variáveis a cada consulta. Se difere da implementação A por não necessitar de tanta memória RAM para armazenamento dos objetos, visto que a aplicação de banco de dados especialista armazena apenas uma porção dos dados em uma memória RAM compartilhada, mesmo se acessado por várias aplicações, enquanto a implementação A possui blocos de memória individuais para cada processo que serve o serviço.

Ambas aplicações utilizam mais de um processo para realização de suas respectivas tarefas. A arquitetura de processos pode ser observada na Figura 18. Para ser inicializado, o processo monitor precisa ser inicializado tendo como parâmetro um arquivo de configurações que contêm a lista de parâmetros de configuração. Através desta configuração ele replica-se criando três tipos de processos com operações distintas, e os monitora para reinicia-los em caso de morte prematura de um processo filho (devido a um erro ou falha). Por questões de desempenho o monitor pode criar mais de um processo responsável por responder conexões de pesquisa se este número for indicado na configuração. O mesmo pode ser realizado para os processos de oferta.

O processo de oferta recebe os objetos e os registra na base de dados interna (no caso da implementação A) ou na base de dados MySQL (implementação B). Caso este receba um

objeto correspondente a outra região, o serviço pode ser configurado para descartar o objeto, ou reencaminhar de forma recursiva (sem informar o cliente) para o nó adequado. O processo de busca (registro de demanda) opera de forma similar, buscando em outros nós se este não possuir autoridade sobre a região, ou não retornando nenhum resultado. O processo de gerência apenas lê os rastros (logs) gerados pelos demais processos e monitora os recursos da máquina onde o serviço está sendo executado.

Para o mapeamento de uma região é utilizada uma configuração que determina um conjunto de pontos que formam um determinado polígono. Nos testes realizados, as formas utilizadas sempre foram quadrados ou retângulos, porém esta não é a situação real na qual a aplicação pode atuar. Para identificar se o ponto de localização está ou não dentro de terminada região, os processos utilizam o algoritmo de cruzamento (apresentado em mais detalhes na Seção 5.2) e os reencaminham para o nó correto se este for de seu conhecimento, ou para o nó superior se este não estiver em sua lista conhecida. Este reencaminhamento quando ativado é recursivo e transparente para o cliente, pois o processo passa a encaminhar a solicitação internamente para o outro nó.

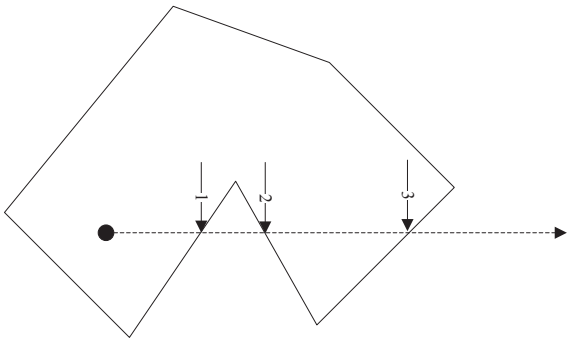
Para convenientemente reconfigurar o serviço de busca ou base de dados de ofertas, os processos foram configurados para obedecer a sinais do sistema operacional. Estes sinais são definidos no padrão POSIX. Há instruções de programação para recarregar o arquivo de configuração ao receber o sinal `SIGUSR1` e redistribuir a base de dados de ofertas armazenadas ao receber o sinal `SIGUSR2`. Desta forma, em uma atividade automatizada de redistribuição dos nós de serviço ou mudanças de hierarquia não é necessário reiniciar os processos serviço.

Ao comunicar-se com o processo de busca ou oferta, o cliente conecta-se através do protocolo HTTP e realiza uma chamada síncrona. Este protocolo pode enviar variáveis para o nó de serviço e resgatar um conteúdo em formato texto, no caso o objeto de oferta ou demanda com suas respectivas informações. Ao enviar uma requisição ao processo de busca o processo sempre responderá com o mesmo objeto de demanda fornecido, acrescido dos objetos que foram identificados pelo mecanismo de *matching*. Este comportamento possibilita uma comunicação sem necessidade de manutenção de sessões pois contém todos os atributos envolvidos na identificação.

5.2 Algoritmos e protocolos utilizados

Este trabalho apoia-se em alguns algoritmos e protocolos existentes para realização de suas tarefas básicas. O protocolo utilizado para comunicação entre os nós e entre cliente e nó de serviço é o HTTP. Este protocolo foi escolhido não só pela simplicidade, mas por permitir a comunicação entre cliente e nó de serviço mesmo através de *proxyes* transparentes. É muito comum este tipo de filtragem de controle em conexões de Internet públicas, na qual muitos dispositivos móveis podem ser utilizados. Além deste protocolo, alguns algoritmos são utilizados para tarefas básicas do nó de operação, como a análise de ponto em uma determinada região e

Figura 19: *Crossing-Test* em um polígono com múltiplas retas



Fonte: Adaptado de Heckbert (1994)

comparação de *matching*.

Para análise de ponto em uma determinada região, utilizado pelos nós de processamento para validar sua região de atuação e também pelo servidor DNS para avaliar o nó correspondente a determinado nome, optou-se pelo algoritmo de cruzamento (*Crossing-Test*) (HECKBERT, 1994). Este algoritmo varre uma determinada linha horizontal ou vertical, analisando o número de retas do polígono no qual intersecciona. Este algoritmo é conhecido pela sua velocidade e também pela possibilidade de aplicação em formas com muitos lados. A Figura 19 apresenta graficamente como o algoritmo avalia o ponto dentro do polígono. Se ao cruzarmos uma reta em uma única direção, o número de retas interseccionadas for ímpar, pode-se considerar que o ponto está dentro do determinado polígono.

O algoritmo utilizado no plugin de comparação baseado em memória RAM é o algoritmo `Soundex` (KNUTH, 1998). Este algoritmo cria uma representação fonética para uma determinada palavra. Esta representação é utilizada de forma a aproximar palavras que possuem uma pronuncia aproximada mas não são escritas da mesma forma. O algoritmo resume qualquer palavra em 4 caracteres, onde o primeiro caractere é a letra inicial da palavra e os demais são números que correspondem a uma codificação baseada na fonética da lingua inglesa.

5.3 Tecnologias empregadas

Para a implementação das três aplicações foram utilizadas tecnologias de código aberto ou livres, que possibilitam uma implementação simplificada e adequada para o ambiente Linux. A distribuição Debian 6 foi escolhida por utilizar um kernel Linux existente na maioria dos sistemas de computação em nuvem disponibilizados (kernel 2.6), sendo estável, facilmente configurável através de acesso remoto (SSH) e possuindo apenas os recursos necessários para execução do serviço. Esta distribuição pode ser fácilmente exportada em um formato de disco virtual e instalada em múltiplas instâncias de máquina virtual, facilitando assim a operação nos testes.

Para a implementação e criação dos scripts foi utilizada a linguagem PHP 5.3. Esta linguagem possui os recursos necessários para comunicação (`Sockets`) e capacidade de multiprocessamento através de primitivas do sistema operacional (`fork`) através do pacote PCNTL. O PHP 5.3 também possui um ótimo suporte a comunicação com bases de dados MySQL, sendo amplamente utilizado em sistemas Web. Tanto a aplicação cliente, como a aplicação de localização e os nós de serviço foram implementados com esta linguagem. Os scripts de inicialização foram implementados em *Bash Script*, pois o Debian 6 utiliza o *Bash* como interpretador de comandos padrão.

O sistema operacional foi executado sobre máquinas virtuais criadas na plataforma de virtualização Virtualbox 4. Esta plataforma foi escolhida por ser compatível com diversos sistemas hospedeiros, como Windows, Linux e Mac OS X, por ser livre de licenças para utilização e por ser ter uma interface de linha de comando muito simples para propósitos de avaliação. Além disso no Virtualbox permite exportar uma máquina virtual em formato OVF (*Open Virtualization Format*) e importa-lo em diversas máquinas através de comandos SSH.

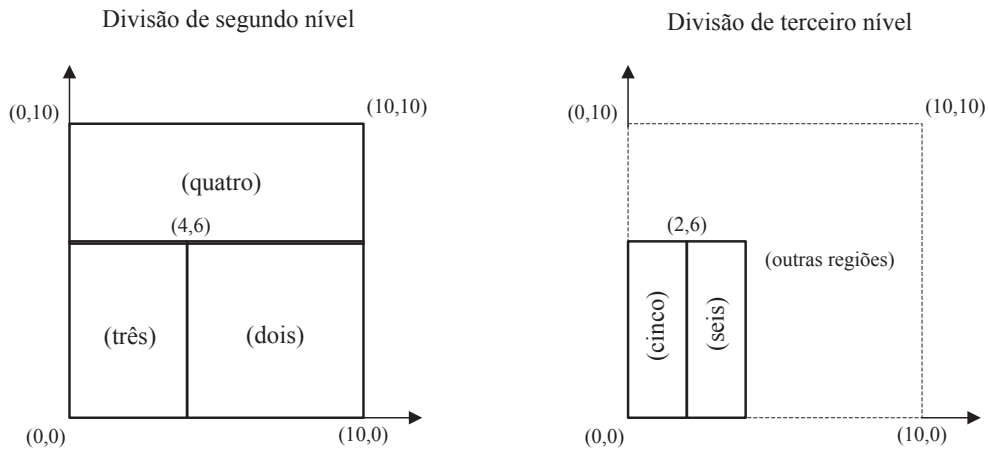
5.4 Cenário e operacionalização do serviço

Para avaliação do modelo na implementação prática foi elaborado um cenário fictício em menor escala, visto as dificuldades de gerar e observar grandes volumes de dados em uma situação real. Este cenário é utilizado como base de todas as avaliações do Capítulo 6. Esta seção apresenta além do mapa de regiões, a hierarquia do modelo para o cenário, e a disposição dos nós de serviços em máquinas virtuais.

O mapa de regiões utilizado nas avalições compreende a região artificial entre as coordenadas geográficas zero e 10 graus positivos em latitude e longitude. Há uma subdivisão que pode ser observada na Figura 20. Apenas a subdivisão de segundo e terceiro nível é apresentada, pois o primeiro nível sempre compreende a região geral da área (no caso 0,0 até 10,10) e será chamada neste trabalho de região (um). As regiões (dois) e (quatro) são divisões finais, e a região (três) é subdividida em mais duas regiões (cinco e seis). A Figura 21 por sua vez associa hierarquicamente como foi feita divisão de regiões, e classifica cada um dos nós de acordo com o modelo. Há um nó de autoridade principal e um nó de autoridade regional enquanto as outras quatro regiões são autoridades de ofertante, onde são armazenados os objetos de ofertas.

Cada uma das regiões é executada em uma máquina virtual composta de duas CPUs virtuais e 2 Gb de memória RAM. O serviço de localização é inicializado em uma máquina virtual com uma CPU virtual e 256Mb de memória RAM. A aplicação cliente para testes de desempenho é executada em uma máquina virtual com quatro CPUs virtuais e 2 Gb de memória RAM, enquanto a aplicação de avaliação de comportamento de usuário é hospedada em um datacenter externo ao ambiente das máquinas virtuais. Todas as conexões de rede entre os nós do serviço e a máquina de testes de desempenho são limitadas virtualmente a 100 Mb/s, enquanto a conexão para o datacenter externo é limitado a 1 Mb/s.

Figura 20: Mapa de regiões de segundo e terceiro nível

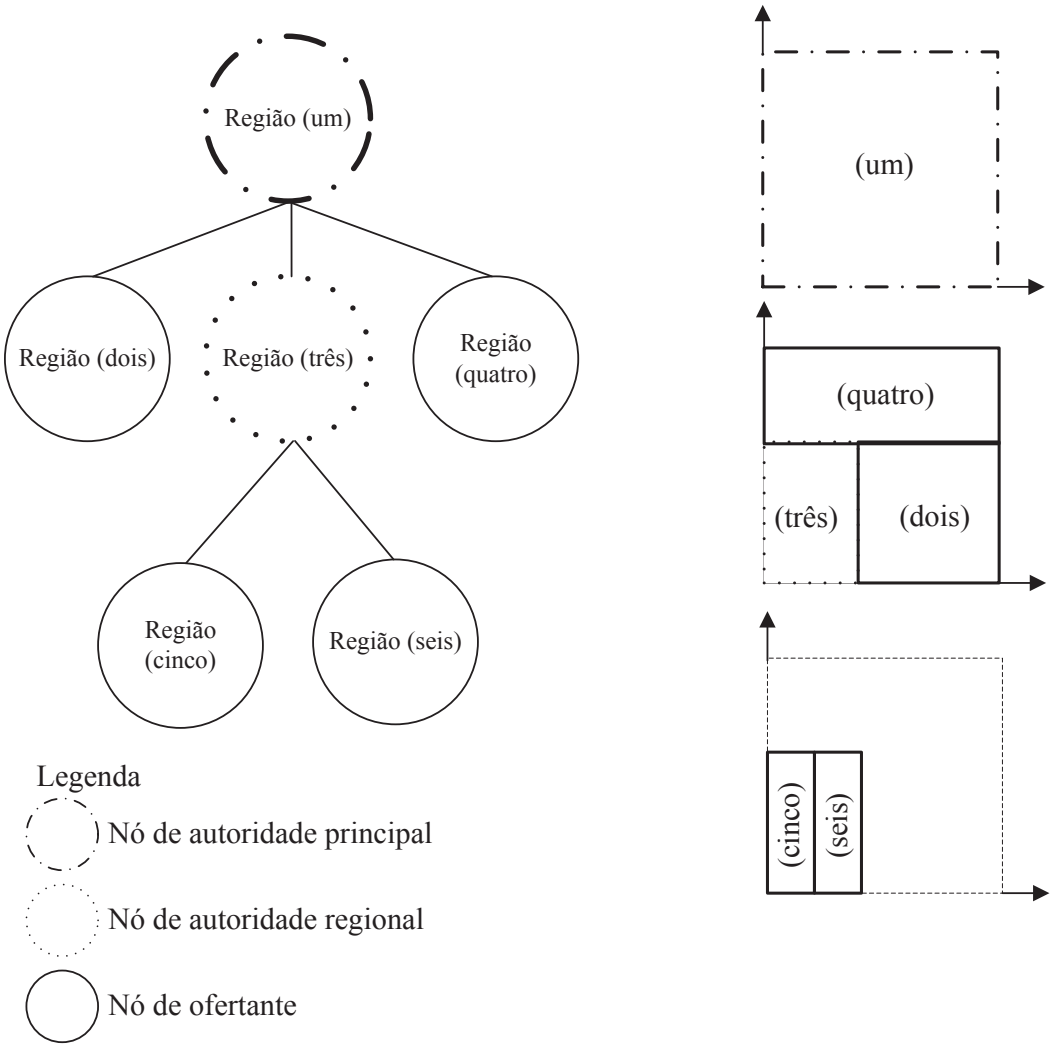


Fonte: Elaborado pelo autor

Para as avaliações correspondentes a base de dados e ao comportamento do usuário, foi utilizado um único servidor hospedeiro. Este servidor que executa o Virtualbox possui um processador Intel Core I7 2600 e 8Gb de memória RAM DDR3-1333 sob o sistema operacional Microsoft Windows 7. Este processador possui instruções para virtualização em hardware e é capaz de executar as 7 máquinas virtuais necessárias sem problemas de desempenho aparentes. Não fora utilizado o cluster de máquinas físicas separadas devido a restrições de rede e tampouco o experimento exigia alto desempenho visto a quantidade de conexões concorrentes ao serviço.

Para as avaliações de desempenho foram utilizadas máquinas hospedeiras separadas para cada máquina de nó de serviço. O cluster utilizado possui 6 máquinas Core I5 e I3 com memórias entre 4Mb e 6Gb, executados sob o sistema operacional Linux e Windows. Todos os processadores possuem extensões para virtualização em hardware. A conexão física de rede é feita através de um switch D-Link de 100 Mb/s. A máquina que hospeda a máquina virtual de teste de desempenho possui um processador Core I5 e 8Gb de memória RAM sob sistema operacional Linux. Todas as máquinas hospedeiras estavam ociosas no momento dos testes e executavam apenas o Virtualbox além das aplicações básicas de sistema.

Figura 21: Hierarquia do mapa de regiões



Fonte: Elaborado pelo autor

6 AVALIAÇÃO

O serviço de identificação de oportunidades implementado com base no modelo proposto neste trabalho possui uma arquitetura singular quanto a sua distribuição hierárquica e mecanismo de processamento, impedindo desta forma a comparação direta de desempenho entre o modelo proposto e os demais trabalhos relacionados. A avaliação do modelo pretende expor características práticas do comportamento do sistema implementado, estabelecendo parâmetros para estudo de seus benefícios bem como os gargalos. Este comportamento e parâmetros são estabelecidos a partir da síntese de uma sequência de experimentos práticos e suas observações individuais.

A execução de várias aplicações simultâneas em um ambiente distribuído torna a observação do comportamento do modelo um desafio prático. A coordenação das aplicações utilizadas na observação e sua posterior interpretação necessita de um conjunto de ferramentas automatizadas para o registro dos dados coletados. Neste Capítulo serão apresentados os métodos utilizados em cada teste, bem como os detalhes práticos de cada avaliação e a síntese dos resultados. As avaliações são separadas quanto a qualidade de dados na base de produtos, quanto ao mecanismo de localização, ao comportamento da implementação do serviço, e por sua vez, a avaliação de possíveis interações com a plataforma de computação em nuvem. Ao fim das avaliações práticas, é feita uma avaliação qualitativa entre o GTTracker e as características de operação observadas nos trabalhos relacionados.

6.1 Metodologia

Uma avaliação prática do serviço com usuários em campo possui alguns impedimentos. Esta avaliação não atingiria o nível de carga de serviço suficiente para avaliações de carga. Tampouco seria capaz de avaliar com clareza a eficiência do serviço em encontrar o produto que o usuário deseja, devido a grande variedade de produtos que podem ser oferecidos e desejados em diversas regiões. Por estas dificuldades na criação de um cenário adequado, muitos autores optam por realizar simulações. Simuladores de aplicações distribuídas (como o SimGrid (CASANOVA, 2001) e CloudSim (CALHEIROS et al., 2011)) possibilitam a criação de cenários maiores, que podem se repetir continuamente e permite ao pesquisador alterar variáveis ou até cenários.

A avaliação com simulador também possui algumas desvantagens. Muitas variáveis podem não ser observadas adequadamente pois testes sintéticos podem excluir fatores desconhecidos pelos pesquisadores. Limitações no framework de simulação também podem impedir a implementação completa do modelo, sendo muitas características possivelmente subestimadas em sua concepção visto que muitas unidades de medida utilizadas nas variáveis dos simuladores são genéricas (como MIPs e tempo de processamento). Cada uma das variáveis envolvidas na simulação podem não ser corretamente estimadas e serem definidas empiricamente. Con-

siderando muitas variáveis estimadas, uma pesquisa em uma determinada região utilizando um determinado objeto de oferta pode causar um impacto de processamento diferente de uma execução de processamento real.

Diante desta situação, este trabalho utiliza o sistema implementado e um conjunto de aplicações que simula o comportamento de diversos usuários. A implementação permite a análise do modelo como se este estivesse em produção. Enquanto o comportamento dos clientes será simulado. Este comportamento de busca é baseado nas palavras buscadas pelos usuários na base de produtos, este comportamento de busca é lido ao mesmo tempo que a qualidade da base de dados é avaliada por seus usuários, esta avaliação foi documentada na Seção 6.2. O mecanismo de localização e sua influência nos testes são relatados na Seção 6.3 através das observações verificadas durante as simulações de carga.

Para compreensão dos efeitos da carga sobre o conjunto de nós do serviço, uma análise do comportamento da arquitetura é apresentada na Seção 6.4. Este comportamento é avaliado a partir de métricas básicas que podem ser observadas através de recursos do sistema operacional. A síntese do consumo de recursos no conjunto de recursos é apresentada e discutida em cada uma das métricas. Este conjunto de métricas permite estabelecer limites de capacidade estimadas para cada nó e definir por sua vez as vantagens da hierarquização dos nós na identificação de uma oportunidade de negócio.

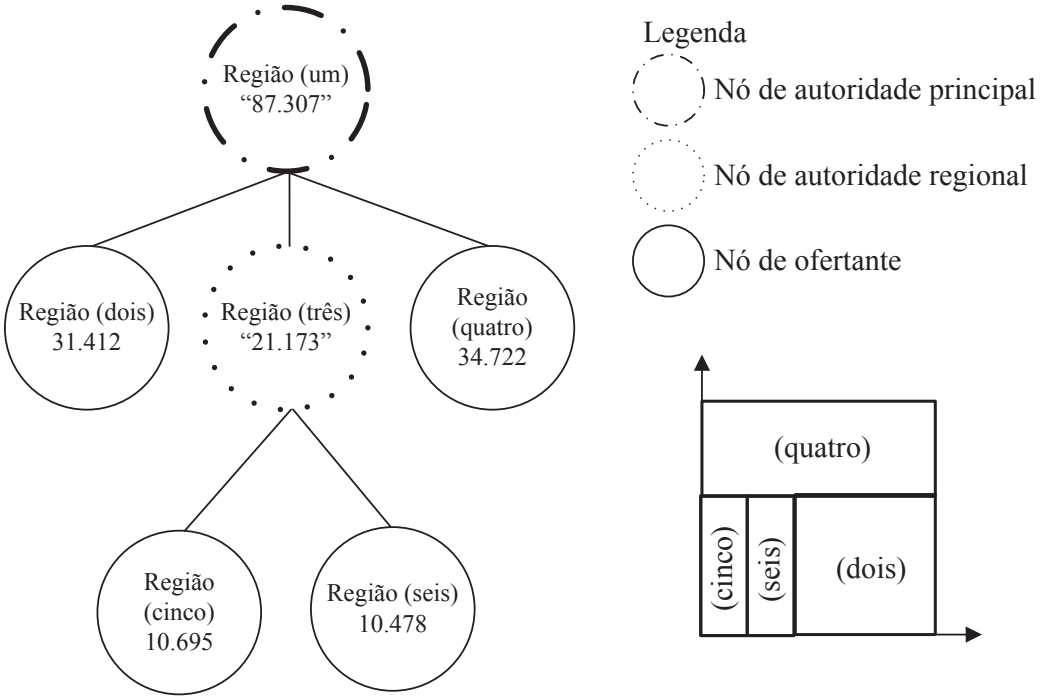
6.2 Base de dados

Para executar a avaliação do serviço é necessário o uso de uma base de dados tanto para ofertas como para demandas. A base de dados de oferta utilizada nos testes foi colhida de 5 lojas virtuais. Estas lojas, que foram anonimizadas neste trabalho, são especializadas em apenas um tipo de produto, livros. Esta escolha por livros foi feita para que exista volume suficiente de produtos que possam ser encontrados pelos usuários do serviço.

Com as listas de produtos contendo nome, valor e loja, gerou-se 87.307 ofertas. Para distribuir estas ofertas de forma uniforme pelas regiões do mapa concebido para as avaliações, utilizou-se um algoritmo de geração de números inteiros aleatórios e uniformes baseado no *Mersenne Twister* (MATSUMOTO; NISHIMURA, 1998). A distribuição por região pode ser observada na Figura 22. É importante observar que os nós de autoridade principal e regional não armazenam ofertas, apenas os nós de ofertante armazenam os objetos. Para facilitar o entendimento das regiões a respectiva quantidade de objetos, apenas os nós de ofertante foram marcados no mapa.

Após a carga dos nós de ofertante viu-se necessidade de estabelecer um conjunto de palavras que comumente usuários utilizariam para suas pesquisas. Para coletar tais palavras foi desenvolvida uma aplicação Web. Esta aplicação pôde ser acessada tanto pelo navegador de um dispositivo móvel ou de um desktop/laptop. Esta aplicação foi disponibilizada em um domínio na Internet e enviada a uma lista de usuários que já possuem familiaridade com Internet. Dos

Figura 22: Número de objetos por região



Fonte: Elaborado pelo autor

Tabela 2: Perfil dos usuários que interagiram com a base de dados

Sexo / Faixa etária	18-20	21-25	26-30	31-35	35-40	41-50	%
Masculino	3	10	5	2	1	-	70%
Feminino	-	3	3	1	1	1	30%

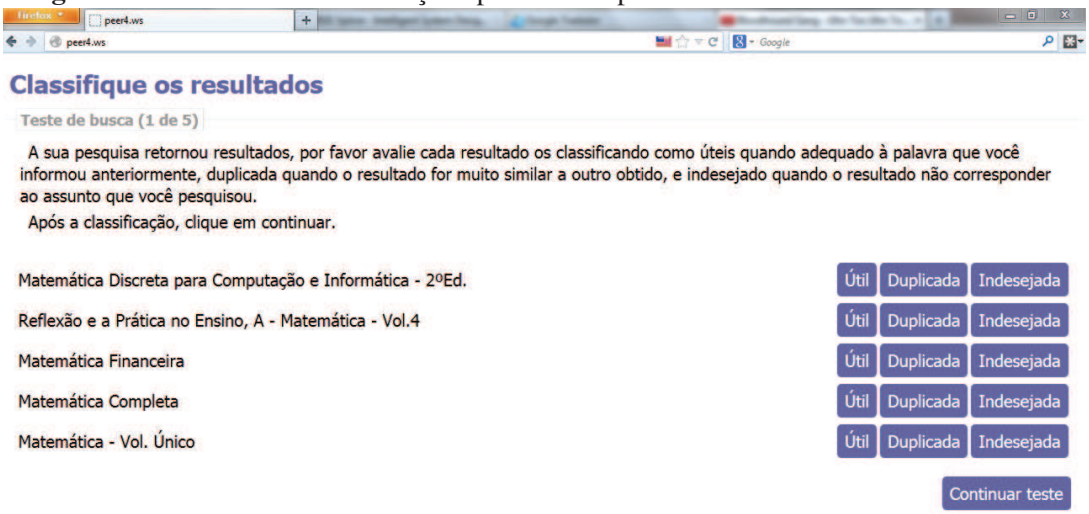
Fonte: Elaborado pelo autor

usuários que receberam a página de teste, 30 pessoas realizaram 5 buscas cada uma, permitindo a coleta de 150 palavras. Para realizar os testes o usuário precisava informar a idade e o sexo. A distribuição de perfil dos usuários pode ser observada na Tabela 2.

A cada palavra pesquisada, um objeto de demanda era gerado e passado para o serviço. O serviço então realiza a busca e classificação de ofertas compatíveis, e por sua vez retorna ao sistema web os resultados. Cada resultado retornado poderia ser avaliado, a intenção desta avaliação é estabelecer uma estimativa de eficiência do serviço perante um determinado número de usuários que tinham conhecimento sobre a existência de um determinado tipo de produtos no serviço. A Figura 23 mostra a interface de classificação ao buscar a palavra "matemática". A classificação de cada resposta poderia ser feita pelo usuário clicando sobre botões que determinariam a seguinte qualidade ao objeto de oferta:

- **Útil**: Para respostas adequadas ao que o usuário está buscando;

Figura 23: Interface de classificação após a busca por uma demanda sobre "matemática"



Fonte: Elaborado pelo autor

- **Duplicada**: Para respostas que aparentemente sejam duplicadas. Esta classificação é baseada no conhecimento do usuário sobre o tópico buscado ou baseada na similaridade entre outras ofertas apresentadas;

- **Indesejada**: Para respostas incompatíveis com a busca efetuada pelo usuário. Em alguns casos é comum a palavra buscada se encaixar em diversos contextos, nestes casos o usuário pode indicar quais respostas não são compatíveis com sua busca.

O usuário também poderia ignorar a avaliação da oferta e seguir para o próximo passo do teste. Ao proceder para o próximo teste, o sistema considera e marca as ofertas não classificadas como "ignoradas". O número de ofertas exibidas para o usuário era limitada a 5 objetos. Esta limitação foi determinada para que os usuários não recebessem demasiadas respostas e se encorajassem a classificar todos os resultados. Esta classificação permitiu gerar a Tabela 3 que resume a média de classificações para todos os objetos retornados. Caso todas as palavras gerassem objetos de demanda com 5 ofertas relacionadas, teríamos 750 objetos classificados.

Cerca de 472 ofertas foram retornadas, desta forma podemos estabelecer uma taxa de resposta do serviço de 62,93%. Porém a taxa de eficiência real, composta apenas por ofertas julgadas úteis do serviço, na base de testes pode ser estabelecida em 24,13%. É importante destacar que esta avaliação foi baseada na implementação A, baseada em memória RAM.

Ao final dos testes relacionados a busca, o usuário é convidado a responder mais duas perguntas. Uma pergunta relacionada à velocidade do serviço e outra relacionada a quantidade de respostas. Com o número médio de respostas de 3,14 respostas por pergunta com o limite máximo de 5 respostas, 16 (53,33%) usuários informaram que esperavam mais respostas, en-

Tabela 3: Média de classificação das palavras buscadas com base nas ofertas retornadas

Classificação	Avaliações	%
Útil	181	≈38,34%
Duplicadas	16	≈2,13%
Indesejada	160	≈33,89%
Ignoradas	115	≈24,36%

Fonte: Elaborado pelo autor

Tabela 4: Média de classificação de tempo de resposta

	Avaliações	%
Adequado	7	≈23,33%
Demorado	3	10%
Rápido	14	≈46,66%
Não responderam	6	20%

Fonte: Elaborado pelo autor

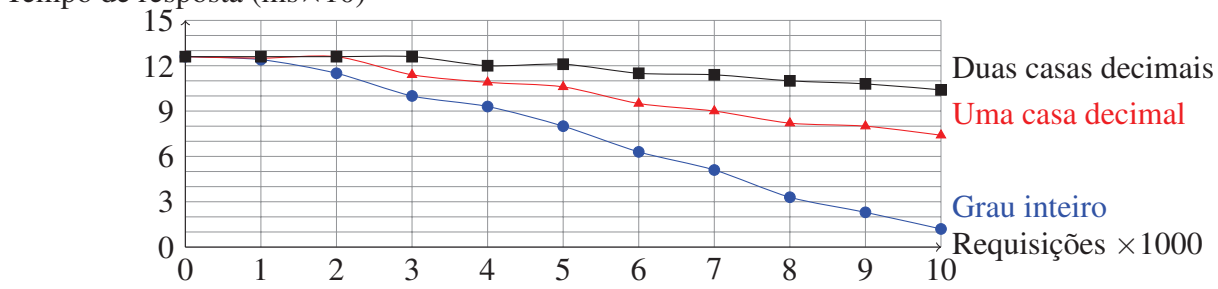
quanto 8 (26,66%) usuários acharam o número de respostas aceitável. Os demais usuários não classificaram o tempo de resposta. A velocidade de resposta também foi armazenada. O tempo médio de resposta foi de 0.56 segundos, sendo estas respostas avaliadas pelos usuários de acordo com a distribuição apresentada na Tabela 4.

6.3 Mecanismo de localização

Com a base de dados carregada de ofertas e com uma lista de demandas alimentada por palavras que possivelmente seriam utilizadas por usuários do serviço foi possível realizar os testes iniciais. O primeiro teste realizado utilizou um script que criava em sequência uma série de objetos de demanda, e por sua vez os encaminhava para um nó do serviço. Para tal, para cada conexão, o script que simula o comportamento de uma série de clientes gera um determinado nome de domínio baseado em sua localização no mapa de teste. Estas coordenadas variam de 0 até 10 graus norte/leste, permitindo criar até um milhão de requisições de nome diferentes.

Uma coordenada geográfica pode utilizar várias casas decimais em sua representação. Quanto maior o número de casas decimais, mais preciso é o ponto de coordenada. Esta precisão causaria problemas ao cache de DNS se não for limitada pois de acordo com a precisão de cada

Figura 24: Tempo de resposta médio de resolução de nome utilizando o mecanismo de localização



Fonte: Elaborado pelo autor

dispositivo móvel é possível gerar diferentes subdomínios, mesmo que na prática estes subdomínios representem pontos muito próximos. Devido ao mecanismo de localização utilizar apenas duas casas decimais, a quantidade de requisições possíveis é reduzida. Permitindo que o cache de DNS seja mais eficiente.

Na tecnologia utilizada (PHP 5.3) é possível realizar uma conexão Socket informando apenas nome de domínio, a função automaticamente identifica que não é um endereço IP e então o converte de forma transparente. Para que fosse possível medir o tempo necessário para resolução de nome, utilizou-se a função *gethostbyname*. Medindo o tempo para chamada e retorno desta função pode-se observar a influência do cache nas resoluções. A fim de observar a diferença de tempo de resposta para cada densidade de diferentes nomes na área do mapa, foram formulados três testes.

O primeiro teste consiste em requisições a graus sem casas decimais (a graus inteiros: 1, 2, 3 até 10). O segundo teste corresponde a requisições com uma casa decimal (1, 1.1, 1.2 até 10). O terceiro teste corresponde a requisições com duas casas decimais (1, 1.01, 1.02). O script para cada um dos testes, realizou 10 mil chamadas DNS. Estas chamadas criaram repetidamente requisições iguais, que em teoria causariam o uso do cache DNS da máquina na qual o script era executado. A Figura 24 apresenta os gráficos de tempo de resposta para as médias de 10 execuções de acordo com a quantidade de casas decimais em razão do número de requisições.

Foi possível observar que o tempo médio por resolução DNS enquanto teoricamente a incidência de nomes era baixa (início do processo) foi de 0.01286 segundos. A partir de determinado número de requisições, o uso do cache no servidor passou a ser mais aparente. Reduzindo o tempo de resolução para 0.00129 segundos. A avaliação prática do mecanismo de localização demostra que o cache já implementado para os dispositivos conectados à internet reduzem também o tempo de resposta do mecanismo de localização utilizado na implementação do modelo. A interferência deste mecanismo nos resultados deve ser mínima se utilizadas apenas coordenadas em números inteiros visto que a atuação do cache é bem visível, diminuindo para 10% o tempo de resolução. A redução do tempo de resolução utilizando duas casas decimais já não é tão expressiva (87%). Em ambientes onde os dispositivos compartilhem o mesmo cache DNS

a eficiência do mecanismo tende a melhorar, pois os dispositivos teoricamente devem consultar o mesmo subdomínio estando nas mesmas coordenadas.

6.4 Comportamento da arquitetura do serviço

O comportamento da arquitetura do GTTracker e seu mecanismo de processamento é um dos principais focos deste trabalho. A avaliação deste comportamento se baseia na análise dos recursos computacionais que envolvem a execução deste serviço. Recursos como consumo de memória, processamento e rede, são métricas comumente utilizadas para ofertar e delimitar diferentes modalidades de serviços IaaS disponíveis comercialmente. Esta avaliação do sistema distribuído não se dá a partir da observação individual de cada nó, e sim do comportamento conjunto em diferentes situações.

Estas situações são recriadas a partir da base de dados de ofertas e demandas, provenientes de base de dados de lojas virtuais e formulada a partir das palavras consultadas pelos usuários, respectivamente. O comportamento de três situações foram observadas e comparadas lado a lado: requisições sequenciais (situação 1); requisições paralelas (situação 2); e requisições concorrentes em um mesmo nó (situação 3). Os três clientes utilizados neste processos foram apresentados tecnicamente na Seção 5.1.1.

Neste trabalho foi realizada a implementação do modelo levando em consideração o uso de plugins de comparação (chamada neste trabalho de implementação A) ou através do *matching* em banco de dados (implementação B). Estas duas implementações permitem compreender como a complexidade do comparador pode afetar o comportamento geral dos nós de serviço. Em teoria os dois comparadores representam diferentes classes de *matching*. O comparador baseado em plugins pode ser muito complexo, levando em consideração diversos elementos para decidir qual é o objeto adequado. Enquanto o comparador baseado em banco de dados, por ser mais genérico, oferece uma comparação rápida mas não tão precisa.

A avalição de desempenho de capacidade da infraestrutura pode ser realizada em duas perspectivas. A primeira perspectiva corresponde ao ponto de vista do usuário ofertante e a segunda sob a perspectiva do usuário consumidor. Neste trabalho há maior ênfase ao desempenho na perspectiva de consumidor, pois este é o que será mais exigente quanto a velocidade de resposta. O usuário ofertante muitas vezes publicará sua oferta e aguardará até que um usuário interessado o busque, enquanto o consumidor quer satisfazer seus desejos o mais breve possível.

A primeira avaliação realizada em ambas implementações baseia-se na publicação das ofertas. A base de ofertas em ambos casos foi publicada utilizando-se do mesmo script, com a mesma base de dados. O carregamento dos nós de ofertas baseado em banco de dados MySQL (implementação B) ocorreu em uma velocidade média de 670 ofertas por segundo, enquanto a baseada em memória (implementação A) ocorreu em uma velocidade de 308 ofertas por segundo. A implementação B obteve um desempenho muito superior se comparada com a ba-

seada em memória RAM devido a complexidade de armazenamento e compartilhamento de informações entre os processos. A cada objeto de oferta recebido, a implementação A concatena o objeto em sua lista de objetos e o replica para os demais processos do serviço. Este procedimento possui um custo adicional se comparado ao MySQL que pode utilizar memória compartilhada para evitar a transferência de dados entre processos.

Para o processo de onde os consumidores enviam objetos de demanda a fim de encontrar possíveis ofertas foi elaborado o monitoramento do conjunto de nós. Enquanto cada situação é simulada, o módulo gerente do nó lê periodicamente o consumo de CPU, memória e consumo de rede e escreve estas informações em um arquivo de rastro. O arquivo de rastro então é lido posteriormente através de um script consolidador, reunindo assim os dados para representar um determinado estado do sistema. Todas as máquinas virtuais tiveram seus relógios sincronizados com a máquina virtual que executa as simulações de cliente. O consumo de rede lido pelo módulo gerente é baseado na leitura de dados trafegados pela interface de rede, que não permite observar a origem da conexão, tampouco a finalidade. Para identificar o consumo de rede real entre cada um dos nós e inclusive o consumo de rede entre o nó e o cliente, utilizou-se a ferramenta `tshark`[1], que é uma variação para linha de comando do aplicativo *Wireshark*.

A primeira situação de avaliação consiste em realizar diversas requisições de consulta sequenciais, não concorrentes. Para tal, durante dez segundos os nós de serviço recebem requisições informando demandas, comparam então estas demandas com possíveis ofertas e então retornam as listas encontradas. É importante salientar que constam na lista de demandas tanto palavras que retornaram resultados, como as palavras que não retornaram resultados na avaliação da base de dados. A Tabela 5 apresenta o consumo de cada nó em função da implementação utilizada.

A velocidade média de consulta é de 530 objetos por segundo na implementação A e 580 objetos por segundo na implementação B. O perfil de consumo de rede é similar e aparentemente proporcional a área na qual o nó possui autoridade. É interessante observar que tanto o nó de autoridade regional, como o nó de autoridade principal (um e três) não realizaram nenhuma comunicação. Isso se justifica pelo fato do mecanismo de localização apontar diretamente para o nó de oferta responsável por determinada área, não envolvendo o nó de autoridade.

O consumo de memória e processamento é maior na implementação A, porém nota-se que em nenhum dos casos atingiu-se o limite de memória ou processamento. A operação sequêncial não é capaz de sobrecarregar o sistema. A máquina virtual responsável por executar o cliente (simulador de comportamento) não atinge médias de CPU superiores a 70% e também não utiliza de toda a largura de banda da rede. Aparentemente o gargalo se encontra na latência de rede e no mecanismo de localização, pois cada consulta exige uma reconecção TCP/IP e uma consulta ao DNS.

A segunda situação é baseada em requisições paralelas. Cada nó possui 4 processos simultâneos para atendimento de requisições recebidas. Estas requisições são realizadas através

[1] http://www.wireshark.org/docs/man-pages/tshark.html

Tabela 5: Comparação da situação de carga em consultas sequenciais

Nó	Implementação A			Implementação B		
	CPU	Mem.	Rede	CPU	Mem.	Rede
Região um	10%	150MB	0	11%	170MB	0
Região dois	65%	878MB	15MB	48%	204MB	17MB
Região três	12%	152MB	0	8%	166MB	0
Região quatro	67%	989MB	17MB	45%	210MB	18MB
Região cinco	50%	670MB	11MB	33%	180MB	14MB
Região seis	47%	690MB	11MB	30%	186MB	12MB

Fonte: Elaborado pelo autor

de um ou mais processos ao mesmo tempo tendo como origem a mesma máquina virtual cliente. Foram realizados testes com números de paralelismo crescente, de um até 10 processos simultâneos (correspondentes a simulação de 10 usuários simultâneos). A Tabela 6 apresenta as médias de consumo para os 10 casos, enquanto a Figura 25 apresenta o número de requisições por segundo suportado pelo sistema em relação a quantidade de clientes simultâneos. As médias são baseadas em 5 testes.

Através do gráfico apresentado na Figura 25 é possível avaliar que a concorrência na implementação A atinge o limite do sistema logo que se aproxima do número de processos existentes em cada nó. Isso leva a crer que a limitação se dá devido a comparação dos objetos exigir mais da capacidade de processamento do nó que da capacidade de memória ou rede. Na implementação B o gargalo parece ser a rede, por mais que o processamento seja alto o limite de conexões por segundo só é atingido quando a rede local é saturada quase atingindo 12MB por segundo (limite para uma rede de 100Mbit/s).

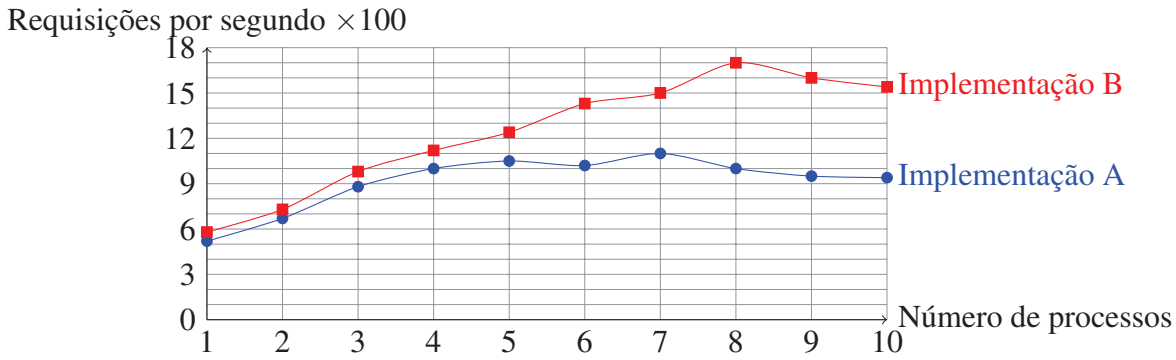
A terceira situação corresponde a sobrecarga de apenas um nó. Neste caso a avaliação pretende tornar visível o comportamento da arquitetura em situações onde o mecanismo de localização não é utilizado. Para tal, foi utilizando o mesmo cliente de simulação de cargas paralelas sem o mecanismo de localização. Como não há o uso de resolução DNS por região o nó foi configurado para se conectar a apenas o nó de autoridade principal da hierarquia. Os objetos de demanda continuam sendo direcionados a uma região aleatória do mapa, porém o

Tabela 6: Comparação da situação de carga em consultas paralelas com mecanismo de localização

Nó	Implementação A			Implementação B		
	CPU	Mem.	Rede	CPU	Mem.	Rede
Região um	9%	151MB	0	10%	176MB	0
Região dois	95%	942MB	29MB	90%	246MB	34MB
Região três	10%	153MB	0	11%	164MB	0
Região quatro	93%	1029MB	26MB	83%	306MB	38MB
Região cinco	96%	730MB	19MB	85%	210MB	28MB
Região seis	89%	754MB	19MB	90%	194MB	29MB

Fonte: Elaborado pelo autor

Figura 25: Gráfico de capacidade do serviço com relação a quantidade de clientes paralelos utilizando o mecanismo de localização



Fonte: Elaborado pelo autor

Tabela 7: Comparação da situação de carga em consultas paralelas através do nó principal

Nó	Implementação A			Implementação B		
	CPU	Mem.	Rede	CPU	Mem.	Rede
Região um	98%	271MB	45MB	98%	280MB	56MB
Região dois	76%	906MB	19MB	71%	235MB	23MB
Região três	82%	203MB	17MB	74%	220MB	20MB
Região quatro	69%	916MB	18MB	60%	302MB	19MB
Região cinco	60%	633MB	15MB	63%	203MB	17MB
Região seis	54%	674MB	14MB	42%	176MB	14MB

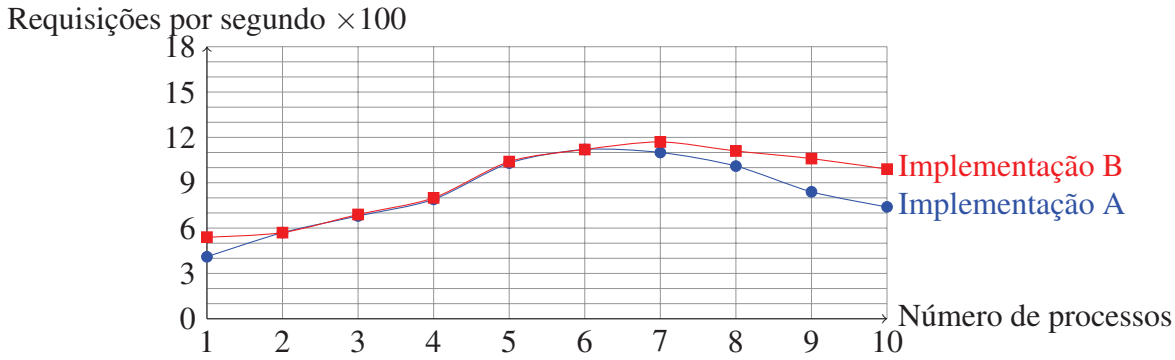
Fonte: Elaborado pelo autor

que ocorre é o redirecionamento entre os nós para resolução da tarefa de *matching*.

A Tabela 7 apresenta médias das métricas em simulações com concorrência de 1 a 10 processos, em 5 repetições. O gráfico da Figura 26 torna aparente que a diferença entre cada uma das implementações não é tão significativa. Levando a crer que o processo de *matching* deixa de ser o principal gargalo do serviço. O limite de comunicação imposto pelo nó de autoridade principal passa a ser o fator determinante no número de consultas por segundo.

O desempenho da implementação B se aproxima do desempenho oferecido pela implementação A utilizando-se do mecanismo de localização (observado na Figura 25). Em contrapartida a implementação B perdeu eficiência devido ao gargalo existente no nó de autoridade principal. O nó de autoridade regional também foi utilizado, porém não apresentou um cenário de sobrecarga (consumindo menos de 90% de processamento).

Figura 26: Gráfico de capacidade do serviço com relação a quantidade de clientes paralelos através do nó principal



Fonte: Elaborado pelo autor

7 CONSIDERAÇÕES FINAIS

O modelo de identificação de oportunidades apresentado neste trabalho foi concebido com a premissa de se adaptar ao crescimento de demanda de processamento. Este crescimento que está relacionado ao aumento do uso de dispositivos móveis por consumidores e ofertantes. Baseado nesta tendência, a união de conceitos de computação em nuvem e dispositivos móveis torna possível a evolução dos modelos de processamento de dados para a utilização de arquiteturas mais robustas. Neste trabalho houve a preocupação de adaptar o mecanismo de comparação ao crescimento de cada região, tornando possível a expansão da capacidade de processamento de acordo com a demanda de usuários de uma determinada região.

O serviço de identificação, chamado de GTTracker, foi implementado a fim de avaliar o comportamento do modelo de processamento. Esta implementação seguiu as premissas básicas do modelo, utilizando uma hierarquia baseada em regiões e utilizando-se de dois mecanismos de *matching*. Um mecanismo baseado em plugins e outro baseado na comparação em banco de dados. Ambas implementações de *matching* foram comparadas em três situações distintas de comportamento. Todas as situações foram reproduzidas com base em palavras que representavam demandas capturadas através de um aplicativo Web oferecido a um conjunto de possíveis usuários do sistema.

As situações e seus respectivos perfis de consumo da infraestrutura podem ser observados na Tabela 8. Os perfis são calculados através da média de consumo de determinada métrica em todos os nós. Através deste resumo é possível observar que a situação 2, na qual há concorrência de vários clientes e o uso do mecanismo de localização, alcançou o melhor resultado de requisições por segundo utilizando-se de um consumo moderado de recursos computacionais, se comparado a implementação A utilizada nas demais situações.

O uso do mecanismo de localização, além de direcionar o cliente ao nó de uma determinada região, realiza o particionamento da carga sem a necessidade de envolvimento dos nós de autoridade principal e regional. A situação 3 permite observar que ao retirar o mecanismo de localização, o nó principal passa a ser o gargalo do sistema e também o ponto crítico do sistema. Caso este nó deixe de funcionar o serviço deixa de ser acessível. Em um ambiente de grande escala, o mecanismo de localização passa a ser fundamental para garantir a independência de processamento entre as regiões.

O modelo do serviço GTTracker pode ser comparado aos demais trabalhos relacionados estudados, como apresentado da Tabela 9. O GTTracker utiliza o modelo de distribuição e processamento publish/subscribe para garantir o fraco acoplamento temporal e espacial. A comunicação por sua vez é baseada no paradigma cliente/servidor, o cliente conecta-se ao nó responsável por servir ofertas para a respectiva região. Todos os objetos são representados através da notação Javascript (JSON). Esta representação de dados permite armazenar no objeto não só os dados de uma demanda, como os metadados e as características da demanda, como as próprias ofertas relacionadas.

Tabela 8: Resumo de consumo de recursos do conjunto de nós para cada situação avaliada

Situação	Implementação A				Implementação B			
	CPU	Mem.	Rede	Req/s	CPU	Mem.	Rede	Req/s
Situação 1	41,83%	588,16MB	9MB	530	29,16%	186MB	10,16MB	580
Situação 2	65,33%	626,5MB	15,5MB	913	61,5%	216MB	21,5MB	1242
Situação 3	73,16%	600,5MB	21,33MB	829	68%	236MB	26,5MB	907

Fonte: Elaborado pelo autor

O mecanismo de localização utilizado no modelo atua como um filtro de processamento, predeterminando qual nó é responsável por processar determinado objeto, com base na região onde o usuário se encontra. A filtragem reduz a quantidade de objetos comparados durante o processo de *matching*. Esta otimização colabora com o uso de uma estratégia de roteamento, implementado no GTTracker dentro dos nós para redirecionar os objetos pertencentes a outras regiões/nós. Através desta estratégia evita-se o processamento de um objeto pertencente a outra região. Na implementação B o uso de índices foi possível pois o algoritmo de *matching* existente no banco de dados exige a criação de índice do tipo `FULLTEXT`.

Outras otimizações do processo de *matching*, como uso da base histórica, a conversão de sinônimos e avaliação com base semântica não foram implementados na avaliação. Estas otimizações poderiam ser implementadas dentro dos plugins de comparação, por serem complementares ao algoritmo de comparação. Neste trabalho estas otimizações não foram avaliadas por existirem técnicas específicas de implementação que poderiam influenciar significativamente nas métricas observadas. A comparação entre ofertas e demandas é feita a partir do recebimento da requisição e a saída do algoritmo de *matching* sempre é um conjunto de objetos.

Após a implementação e avaliação, pode-se observar o comportamento da arquitetura do serviço de identificação de oportunidades. Formada por nós de processamento independentes, o serviço foi implementado de forma a adaptar-se a plataformas de computação em nuvem, permitindo a alocação de recursos virtuais a fim de atender determinada demanda de uma região. Ao atingir o limite de processamento de uma determinada região, o modelo permite a subdivisão de um nó em dois ou mais nós, que podem ser provisionados em uma plataforma de computação em nuvem.

Tabela 9: Comparação do GTTracker com os trabalhos relacionados

Característica / Proposta	DREAM	S-ToPSS	JTangPS	JTangCSPS	Guo, Wei e Han (2008)	Cao et al. (2010)	GTTracker
Modelos de distribuição e processamento de informação	Pub/Sub	Pub/Sub	Pub/Sub	Pub/Sub	Pub/Sub	MapReduce	Pub/Sub
Método de comunicação	Cliente/ Servidor	Cliente/ Servidor	Cliente/ Servidor (SOA)	P2P	Cliente/ Servidor	Cliente/ Servidor	Cliente/ Servidor
Representação de dados	MIX	Chave/Valor	RDF	RDF	Chave/Valor	Tupla	JSON
Informações avaliadas no *matching*	Eventos	Objetos	Eventos	Eventos	Eventos	Regras	Objetos
Usa de filtros	Sim	Não	Sim	Sim	Sim	Não	Sim
Usa de índices	Sim	Sim	Sim	-	Sim	-	Sim
Usa de estratégias de roteamento	Sim	Não	Não	Sim	Sim	Não	Sim
Usa base histórica	Não	Não	Não	Não	Sim	Não	Não
Considera sinônimos	Não	Sim	Não	Não	Não	Não	Não
Considera semântica	Sim	Sim	Sim	Sim	Não	Não	Não
Fraco acoplamento	Sim	Sim	Sim	Sim	Sim	Não	Sim
Gatilho de execução	Requisição	Requisição	Requisição	Requisição	Requisição	Ciclo	Requisição
Saída do algoritmo	-	Notificações	-	-	-	Regras	Objetos

Fonte: Elaborado pelo autor

Durante o desenvolvimento deste trabalho, três artigos científicos relacionados ao modelo foram publicados, o que sinaliza o interesse da comunidade acadêmica na pesquisa e desenvolvimento do serviço proposto. Os artigos publicados foram:

I) CAZAROTTO, P.H., COSTA, C.A., RIGHI, R.R.. **Um modelo para busca e classificação de objetos em um sistema distribuído organizado hierárquicamente**. *ERAD 2012 - Escola Regional de Alto Desempenho, Fórum de Pós Graduação*. Erechim, Brasil. Março, 2012.

II) CAZAROTTO, P.H., COSTA, C.A., RIGHI, R.R., BARBOSA, J.L.V. **Combinação Eficiente de DNS, P2P e Dispositivos Móveis para Impulsionar Negócios entre Fornecedores e Consumidores**. *CLEI 2012 - Conferencia Latinoamericana en Informática, Simposio de Infraestructura Hardware y Software*. Medellín, Colombia. Outubro, 2012.

III) CAZAROTTO, P.H., COSTA, C.A., RIGHI, R.R., BARBOSA, J.L.V. **Infrastructure to Next-Gen Proactive E-commerce Environment Through Internet: Ubiquitous Commerce For The Masses**. *ICWI 2012 - IADIS International Conference WWW/Internet*. Madrid, Espanha. Outubro, 2012.

O artigo (I) apresenta a ideia de processamento hierárquico utilizado no modelo do serviço. O artigo (II) apresenta uma proposta de mecanismo de localização para os nós do serviço, apresentando a ideia de hierarquia e o comportamento de aplicação P2P dos nós propostos neste trabalho. O artigo (III) apresenta a mesma infraestrutura de serviço com uma abordagem voltada à computação ubíqua.

Este trabalho possui um escopo limitado à definição do modelo e sua implementação chamada GTTracker. Assume-se algumas limitações quanto ao estudo teórico do uso de nuvens federadas. Na qual este trabalho também pode se classificar por ter capacidade de ser oferecido com um serviço na nuvem e utilizar recursos de outras nuvens (como a IaaS). O uso de nuvens federadas trás também uma série de questões relacionadas a segurança da informação, que por estarem armazenadas em uma nuvem computacional de terceiros podem ser vulneráveis. Assume-se também limitações quanto a profundidade do estudo sobre as características utilizadas na comparação com enfoque em contexto. Este trabalho possui como foco o uso da localização como base para infraestrutura, sendo essencial para sua operação o uso da localização, porém outras informações podem e devem ser utilizadas para evolução do mecanismo de comparação.

Como trabalhos futuros, pretende-se implementar algoritmos automatizados para subdivisão dos nós sob uma infraestrutura de computação em nuvem. Este algoritmo, orientado por informações provenientes do módulo de gerenciamento dos nós, coordenaria a subdivisão das regiões em uma ou mais máquinas virtuais, aumentando ou diminuindo sua capacidade de processamento. Esta interação pode ser realizada através de interfaces de comunicação com as APIs de desenvolvimento do fornecedor da solução de infraestrutura na nuvem.

A interação com variados tipos de dispositivos automatizados também é desejada nos próximos estudos, a fim de permitir que estes equipamentos ofertem e busquem por determinados produtos ou serviços em determinada região. Por fim, pretende-se desenvolver uma avaliação de sua aplicação em um cenário futuro, onde um diretório comum de ofertas/demandas pode garantir interoperabilidade entre diversos dispositivos que permitam interações comerciais. Um exemplo desta interação seria uma "geladeira inteligente" informar uma demanda de alimento e encontrar em um supermercado próximo uma determinada oferta.

Com a atual tendência de mercado no comércio móvel e uso de computação em nuvem o presente trabalho buscou propor um serviço de identificação de oportunidades comerciais que, sobre uma infraestrutura elástica, pode se adaptar à necessidade de processamento de diferentes regiões. Estas regiões, por sua vez, podem adotar de forma progressiva o uso deste serviço sob uma plataforma de baixo custo virtualizada, de acordo com a demanda exigida por seus usuários facilitando assim sua implantação. Com a utilização do GTTracker é possível o desdobramento de novas tecnologias que possibilitem a geração de negócios entre vendedores e compradores em uma determinada região. Espera-se que este modelo ofereça um novo canal de comunicação no relacionamento entre quem quer vender e quem quer comprar, em qualquer lugar e a qualquer momento.

REFERÊNCIAS

ABOWD, G.; DEY, A.; BROWN, P.; DAVIES, N.; SMITH, M.; STEGGLES, P. Towards a better understanding of context and context-awareness. In: HANDHELD AND UBIQUITOUS COMPUTING, 1999, Berlin, Heidelberg. **Anais...** Springer, 1999. p. 304–307. (Lecture Notes in Computer Science, v. 1707).

AMIR, A.; AUMANN, Y.; COLE, R.; LEWENSTEIN, M.; PORAT, E. Function matching: algorithms, applications, and a lower bound. **Automata, Languages and Programming**, Berlin, p. 194–194, 2003.

AMIR, A.; NOR, I. Generalized function matching. **Journal of Discrete Algorithms**, London, v. 5, n. 3, p. 514–523, Sept. 2007.

BAKER, B. S. A theory of parameterized pattern matching. In: ACM SYMPOSIUM ON THEORY OF COMPUTING - STOC '93, 1993, New York, New York, USA. **Proceedings...** ACM Press, 1993. p. 71–80.

BANERJEE, P.; FRIEDRICH, R.; BASH, C.; GOLDSACK, P.; HUBERMAN, B.; MANLEY, J.; PATEL, C.; RANGANATHAN, P.; VEITCH, A. Everything as a Service: powering the new information economy. **Computer**, New York, v. 44, n. 3, p. 36–43, Mar. 2011.

BELLAVISTA, P.; CORRADI, A. **The Handbook of Mobile Middleware**. Boca Raton, FL: Taylor & Francis, 2006. n. v. 978, nos. 0-3835. (The Handbook of Mobile Middleware).

BORNHOVD, C.; BUCHMANN, A. **A Prototype for Metadata-Based Integration of Internet Sources**. Berlin: Springer Berlin Heidelberg, 1999. 439-445 p. v. 1626.

BREITMAN, K. Computacao na Nuvem. In: ATUALIZACOES EM INFORMATICA, 2010, Rio de Janeiro. **Anais...** SBC, 2010.

BUCHMANN, A.; BORNHOVD, C.; CILIA, M.; FIEGE, L.; GARTNER, F.; LIEBIG, C.; MEIXNER, M.; MUHL, G. **DREAM**: distributed reliable event-based application management. Berlin: Springer, 2003.

CALHEIROS, R. N.; RANJAN, R.; BELOGLAZOV, A.; ROSE, C. A. F. D.; BUYYA1, R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. **Software: Practice and Experience**, [S.l.], p. 23 –50, 2011.

CAO, B.; YIN, J.; ZHANG, Q.; YE, Y. A MapReduce-Based Architecture for Rule Matching in Production System. In: SECOND INTERNATIONAL CONFERENCE ON CLOUD COMPUTING TECHNOLOGY AND SCIENCE, 2010, Indianapolis. **Anais...** IEEE, 2010. p. 790–795.

CASANOVA, H. Simgrid: a toolkit for the simulation of application scheduling. In: INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID, 2001, Brisbane, Australia. **Anais...** IEEE, 2001. p. 430 –437.

CHERTOK, M.; KELLER, Y. Efficient high order matching. **IEEE transactions on pattern analysis and machine intelligence**, New York, v. 32, n. 12, p. 2205–15, Dec. 2010.

CROCKFORD, D. **RFC 4627 - The application/json Media Type for JavaScript Object Notation (JSON)**. [S.l.]: IETF, 2006.

DEAN, J.; GHEMAWAT, S. MapReduce : simplified data processing on large clusters. **Communications of the ACM**, New York, v. 51, n. 1, p. 107–113, 2008.

DEY, A. K. Understanding and Using Context. **Personal and Ubiquitous Computing**, London, v. 5, n. 1, p. 4–7, Feb. 2001.

EKANAYAKE, J.; PALLICKARA, S.; FOX, G. MapReduce for Data Intensive Scientific Analyses. **Fourth International Conference on eScience**, Indianapolis, p. 277–284, Dec 2008.

ELKHIYAOUI, K.; KATO, D.; KUNIEDA, K.; YAMADA, K.; MICHIARDI, P. A scalable interest-oriented peer-to-peer pub/sub network. In: IEEE NINTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2009, New York. **Anais...** IEEE, 2009. p. 204–211.

EUGSTER, P. T.; FELBER, P. A.; GUERRAOUI, R.; KERMARREC, A.-m. The many faces of publish/subscribe. **ACM Computing Surveys**, New York, v. 35, n. 2, p. 114–131, Jun 2003.

EVANS, C.; HU, B. E-commerce to U-business: a model for ubiquitous shopping mall. In: FIRST INTERNATIONAL SYMPOSIUM ON PERVASIVE COMPUTING AND APPLICATIONS, 2006, Beijing, China. **Anais...** IEEE, 2006. n. 1993, p. 427–432.

FABRET, F.; JACOBSEN, H. A.; LLIRBAT, F.; PEREIRA, J.; ROSS, K. A.; SHASHA, D. Filtering algorithms and implementation for very fast publish/subscribe systems. **SIGMOD Rec.**, New York, v. 30, n. 2, p. 115–126, May 2001.

FORGY, C. L. Rete: a fast algorithm for the many pattern/many object pattern match problem. **Artificial Intelligence**, [S.l.], v. 19, n. 1, p. 17–37, Sep 1982.

FORMAN, G.; ZAHORJAN, J. The challenges of mobile computing. **Computer**, New York, v. 27, n. 4, p. 38–47, Apr. 1994.

FRANCO, L. K.; ROSA, J. H.; BARBOSA, J. L.; COSTA, C. a.; YAMIN, A. C. MUCS: a model for ubiquitous commerce support. **Electronic Commerce Research and Applications**, [S.l.], v. 10, n. 2, p. 237–246, Mar 2011.

FRANSEN, F.; LACHMUND, S.; OLK, E.; BUSSARD, L. An Infrastructure for Gaining Trust in Context Information. In: SECURECOMM AND WORKSHOPS, 2006, New York. **Anais...** IEEE, 2006. p. 1–10.

GERO, M.; BUCHMANN, A.; FIEGE, L. Filter Similarities in Content-Based Publish/Subscribe Systems. In: INTERNATIONAL CONFERENCE ON ARCHITECTURE OF COMPUTING SYSTEMS: TRENDS IN NETWORK AND PERVASIVE COMPUTING, 2002, London. **Anais...** Springer-Verlag, 2002. v. 2299/2002, p. 187–204.

GIUNCHIGLIA, F.; YATSKEVICH, M.; SHVAIKO, P. Semantic matching: algorithms and implementation. **IX Journal on Data Semantics**, Berlin, p. 1–38, 2007.

GREENLAW, R. An activity profile model and dynamic-matching results for social networks regarding wellness applications. In: INTERNATIONAL CONFERENCE ON ELECTRICAL ENGINEERING/ELECTRONICS COMPUTER TELECOMMUNICATIONS AND INFORMATION TECHNOLOGY, 2010, Chiang Mai, Thailand. **Anais...** IEEE, 2010.

GRIGORAS, D. Challenges to the Design of Mobile Middleware Systems. In: INTERNATIONAL SYMPOSIUM ON PARALLEL COMPUTING IN ELECTRICAL ENGINEERING, 2006. **Anais...** IEEE, 2006. p. 14–19.

GUO, X.; WEI, J.; HAN, D. Efficient Event Matching in Publish/subscribe: based on routing destination and matching history. **International Conference on Networking, Architecture, and Storage**, Chongqing, China, p. 129–136, June 2008.

HACHICHA, M.; DARMONT, J. A Survey of XML Tree Patterns. **IEEE Transactions on Knowledge and Data Engineering**, New York, v. 25, n. 1, p. 29 –46, jan 2013.

HAMMERSHOJ, A.; SAPUPPO, A.; TADAYONI, R. Challenges for mobile application development. In: INTERNATIONAL CONFERENCE ON INTELLIGENCE IN NEXT GENERATION NETWORKS, 14., 2010. **Anais...** IEEE, 2010. p. 1–8.

HAN, H.; KLARE, B.; BONNEN, K.; JAIN, A. Matching Composite Sketches to Face Photos: a component-based approach. **IEEE Transactions on Information Forensics and Security**, New York, v. 8, n. 1, p. 191 –204, jan 2013.

HECKBERT, P. **Graphics Gems 4**. London, UK: Academic Press, 1994. (The Graphics Gems Series).

HU, Z. Research and Application of Reverse Mobile Commerce Platform. In: FIRST INTERNATIONAL CONFERENCE ON INFORMATION SCIENCE AND ENGINEERING, 2009., 2009. **Anais...** IEEE, 2009. p. 2995–2997.

HUANG, D.; LIU, W.; LI, X. A survey on context awareness. In: INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE AND SERVICE SYSTEM, 2011. **Anais...** IEEE, 2011. p. 144–147.

KASSEL, S. Design of services as interoperable systems an e-commerce case study. In: ENTERPRISE DISTRIBUTED OBJECT COMPUTING CONFERENCE WORKSHOPS, 12., 2008. **Anais...** IEEE, 2008. p. 264–268.

KNUTH, D. **Art of Computer Programming, Volume 3**: sorting and searching. New York: Addison-Wesley, 1998. (The Art of Computer Programming).

LIU, E. S.; THEODOROPOULOS, G. K. A fast parallel matching algorithm for continuous interest management. In: WINTER SIMULATION CONFERENCE, 2010, New York. **Anais...** IEEE, 2010. p. 1490–1500.

MALATHI, M. Cloud computing concepts. In: INTERNATIONAL CONFERENCE ON ELECTRONICS COMPUTER TECHNOLOGY, 3., 2011, Kanyakumari, India. **Anais...** IEEE, 2011. p. 236–239.

MANVI, S. S.; NALINI, N.; BHAJANTRI, L. B. Recommender system in ubiquitous commerce. In: INTERNATIONAL CONFERENCE ON ELECTRONICS COMPUTER TECHNOLOGY, 3., 2011, Kenyakumari, India. **Anais...** IEEE, 2011. p. 434–438.

MARGARA, A.; CUGOLA, G. High Performance Publish-Subscribe Matching Using Parallel Hardware. **IEEE Transactions on Parallel and Distributed Systems**, New York, v. PP, n. 99, p. 1, 2013.

MATSUMOTO, M.; NISHIMURA, T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. **ACM Trans. Model. Comput. Simul.**, New York, v. 8, n. 1, p. 3–30, Jan. 1998.

MEIYAN, W. Analysis to the Weaknesses and Safety in Cloud Computing. In: THIRD INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEM DESIGN AND ENGINEERING APPLICATIONS, 2013, Hong Kong, China. **Anais...** IEEE, 2013.

MELL, P.; GRANCE, T. **The NIST definition of cloud computing**. [S.l.]: NIST, 2011.

MENG, X.; MA, Z.; CHENG, R.; WANG, X. A Context-Sensitive Approach for Web Database Query Results Ranking. In: IEEE/WIC/ACM INTERNATIONAL CONFERENCE ON WEB INTELLIGENCE AND INTELLIGENT AGENT TECHNOLOGY, 2008., 2008. **Anais...** IEEE, 2008. n. Fskd, p. 836–839.

NWIABU, N.; ALLISON, I.; HOLT, P.; LOWIT, P.; OYENEYIN, B. Situation awareness in context-aware case-based decision support. In: IEEE INTERNATIONAL MULTI-DISCIPLINARY CONFERENCE ON COGNITIVE METHODS IN SITUATION AWARENESS AND DECISION SUPPORT, 2011, Miami. **Anais...** IEEE, 2011. p. 9–16.

PAPAZOGLOU, M. P.; TRAVERSO, P.; DUSTDAR, S.; LEYMANN, F. Service-Oriented Computing: state of the art and research challenges. **Computer**, New York, v. 40, n. 11, p. 38–45, Nov 2007.

PETROVIC, M.; BURCEA, I.; JACOBSEN, H.-A. S-ToPSS: semantic toronto publish/subscribe system. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATABASES, 29., 2003. **Anais...** VLDB Endowment, 2003. v. cs.DC/0311041, p. 1–4.

PITOURA, E.; SAMARAS, G. Locating objects in mobile computing. **IEEE Transactions on Knowledge and Data Engineering**, New York, v. 13, n. 4, p. 571–592, 2001.

QIAN, J.; YIN, J.; DONG, J.; SHI, D. JTangCSPS: a composite and semantic publish/subscribe system over structured p2p networks. **Engineering Applications of Artificial Intelligence**, New York, v. 24, n. 8, p. 1487–1498, Dec 2011.

QIAN, J.; YIN, J.; SHI, D.; DONG, J. Exploring a Semantic Publish/Subscribe Middleware for Event-Based SOA. In: ASIA-PACIFIC SERVICES COMPUTING CONFERENCE, 2008, Yilan, Taiwan. **Anais...** IEEE, 2008. p. 1269–1275.

QILIN, L.; MINTIAN, Z. The State of the Art in Middleware. In: INTERNATIONAL FORUM ON INFORMATION TECHNOLOGY AND APPLICATIONS, 2010, New York. **Anais...** IEEE, 2010. p. 83–85.

SANCHEZ-PI, N.; MOLINA, J. M. A multi-agent platform for the provisioning of U-commerce services. In: ANNUAL MEETING OF THE NORTH AMERICAN FUZZY INFORMATION PROCESSING SOCIETY, 2009, New York. **Anais...** IEEE, 2009. v. 20, n. 3, p. 1–6.

SATYANARAYANAN, M. Mobile computing. In: ACM WORKSHOP ON MOBILE CLOUD COMPUTING AND SERVICES: SOCIAL NETWORKS AND BEYOND, 2011, New York. **Anais...** ACM, 2011. v. 15, n. 2, p. 2.

SCHILIT, B. N. Mobile Computing: looking to the future. **Computer**, New York, v. 44, n. 5, p. 28–29, May 2011.

SHAH, N.; DAWSON, R. Management issues regarding e-commerce and the Internet: 20 critical questions managers should ask before plunging into e-commerce! In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF INNOVATION AND TECHNOLOGY, 2000, Singapore. **Anais...** IEEE, 2000. v. 2, p. 622–628.

SHIH, J.-M.; LIAO, C.-S.; CHANG, R.-S. Simplifying MapReduce Data Processing. In: FOURTH IEEE INTERNATIONAL CONFERENCE ON UTILITY AND CLOUD COMPUTING, 2011, Melbourne. **Anais...** IEEE, 2011.

SIBAI, F. N.; ZAKI, N. Parallel protein sequence matching on multicore computers. In: INTERNATIONAL CONFERENCE OF SOFT COMPUTING AND PATTERN RECOGNITION, 2010, Cergy Pontoise/Paris, France. **Anais...** IEEE, 2010. p. 285–290.

THOMAS, S.; LINNHOFF-POPIEN, C. A context modeling survey. In: UBICOMP 2004 - THE SIXTH INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING, 2004, Nottingham, UK. **Anais...** University of Nottingham, 2004.

VAIDYA, B.; MAKRAKIS, D.; MOUFTAH, H. T. Robust RFID Authentication for Supply Chain Management. In: VEHICULAR TECHNOLOGY CONFERENCE, 2012, Yokohama. **Anais...** IEEE, 2012.

WANG, H.; PU, S.; KNEZEK, G.; LIU, J.-C. MIN-MAX: a counter-based algorithm for regular expression matching. **IEEE Transactions on Parallel and Distributed Systems**, New York, v. 24, n. 1, p. 92 –103, jan 2013.

WATSON, R. T.; PITT, L. F.; BERTHON, P.; ZINKHAN, G. M. U-Commerce: expanding the universe of marketing. **Journal of the Academy of Marketing Science**, Michigan, v. 30, n. 4, p. 333–347, Oct 2002.

WEISER, M. The computer for the 21 st century. **ACM SIGMOBILE Mobile Computing and Communications Review**, New York, v. 3, n. 3, p. 3–11, July 1999.

WIDENIUS, M.; AXMARK, D.; AB, M. **MySQL Reference Manual**: documentation from the source. Sebastopol: O'Reilly Media, Incorporated, 2002. (Oreilly Series).

XIUZHEN, F. A study on tree matching model and algorithm towards metadata. In: THE 2ND IEEE INTERNATIONAL CONFERENCE ON INFORMATION MANAGEMENT AND ENGINEERING (ICIME), 2010, Chengdu, China. **Anais...** IEEE, 2010. p. 259–262.

YIN, G.; CUI, X.; MA, Z. A Model for Web Services Discovery with Matching Algorithm. In: FIFTH INTERNATIONAL CONFERENCE ON INTERNET COMPUTING FOR SCIENCE AND ENGINEERING, 2010, Harbin, Hei Long Jiang, China. **Anais...** IEEE, 2010. p. 75–80.

ZHAO, J.; XUE, L.-J.; MEN, G.-Z. Optimization matching algorithm based on improved Harris and SIFT. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND CYBERNETICS, 2010, Qingdao, China. **Anais...** IEEE, 2010. p. 258–261.

ZHAO, L.; FENG, X. The M-Commerce Architecture Study Based on SaaS Model. In: MASS '09. INTERNATIONAL CONFERENCE ON MANAGEMENT AND SERVICE SCIENCE, 2009, Wuhan/Beijing, China. **Anais...** IEEE, 2009. p. 1–4.

ZHENG, K.; LU, H.; NAHUM, E. Scalable Pattern Matching on Multicore Platform via Dynamic Differentiated Distributed Detection (D4). **IEEE Transactions on Computers**, New York, v. 60, n. 3, p. 346–359, Mar. 2011.

ANEXO A CÓDIGO FONTE DO PLUGIN DE COMPARAÇÃO

```
<?php

    class GTService_Plugins_Simple {

        public static function compare($demanda,$oferta){

            $score = 0;

            // Criando listas a partir das palavras a serem comparadas
            $dname = array_unique(explode(" ",strtolower($demanda['demanda']['nome'])));
            $oname = array_unique(explode(" ",strtolower($oferta['oferta']['nome'])));

            // Para cada palavra da demanda
            foreach($dname as $dp=>$dword){

                // Ignora-se palavras de 2 ou menos letras
                $len = strlen($dword);
                if($len<3) continue;

                // Criando o código Soundex para a palavra
                $dsound = soundex($dword);

                // Para cada palavra da oferta
                foreach($oname as $op=>$oword){

                    // Soma-se 15 pontos para palavras iguais
                    if($oword==$dword){
                        $score += 15;

                    // Soma-se 2 pontos para palavras com mesmo código Soundex
                    }elseif(soundex($oword)==$dsound){
                        $score += 2;

                    // Remove-se 0.1 ponto para cada palavra não considerada similar
                    }else{
                        $score -= 0.1;
                    }
                }
            }

            // Retorna a pontuação de matching para o algoritmo comparador
            return $score;
        }

    }
```