PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Programa de Graduação em Ciência da Computação

José Ferreria Reis Fonseca

ASSISTANCE:
plataforma em grade para software como serviço aplicado à inteligência artificial

Belo Horizonte
2015

José Ferreria Reis Fonseca

ASSISTANCE:
plataforma em grade para software como serviço aplicado à inteligência artificial

Monografia apresentada ao Programa de Graduação em Ciência da Computação da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Barcharel em Ciência da Computação.

Orientador: Luís Fabrício Goés

Belo Horizonte
2015

"O homem nunca parou de inventar, mais, mais, mais, sempre mais, e desse modo foi desenvolvendo a sua capacidade inventiva até distanciar-se infinitamente de todos os outros seres que habitam a Terra."
- Monteiro Lobato

"Vamos precisar de todo mundo
um mais um é sempre mais que dois
Pra melhor juntar as nossas forças
é só repartir melhor o pão"
- Beto Guedes e Ronaldo Bastos

"A day may come when the courage of man fails,
when we forsake our friends
and break all bonds of fellowship,
but it is not this day.
An hour of wolves and shattered shields,
when the age of man comes crashing down,
but it is not this day!
This day we fight!"
- J. R. R. Tolkien

LISTA DE FIGURAS

LISTA DE SIGLAS

AI – Artificial Intelligence - Inteligência Artificial (antiga notação)

AGI – Artificial General Intelligence - Inteligência Artificial Geral

API – Application Interface - Interface de Aplicação

DCG – Desktop Computer Grid - Grade de Computadores Domésticos

DNS – Domain Name System - Sistema de Nomeclatura de Domínio

GPGPU – General Purpose Graphic Processing Unit - Unidade de Processamento Gráfico de Propósito Geral

HCP – High-Performace Computing - Computação de Alto Desempenho

MP-DCG – Message Passing Desktop Computer Grid - Grade de Computadores Domésticos que opera por Passagem de Mensagens

P2P – Peer 2(to) Peer - Ponto à Ponto

SEA – Software Engineering Artifact - Artefato de Engenharia de Software

SIMD – Single Instriction, Multiple Data - Unica Sequência de Instruções para Multiplas Fontes de Dados

SLR – Systematic Literature Review - Revisão Sistemática de Literatura

WEKA – Waikato Environment for Knowledge Analysis - O Ambiente de Análise de Bases de Conhecimento da Universidade de Waikato (Nova Zelândia)

SUMÁRIO

1 INTRODUÇÃO

A prática da simulação em ambientes virtuais se tornou frequente e por vezes indispensável no processo de desenvolvimento científico e industrial (CIRNE et al., 2006). Tais simulações são um exemplo das incontáveis aplicações que processam grandes quantidades de dados e que precisam de resultados o mais rápido possível (á esta categoria de aplicações se da o nome de "computação de alto desempenho", em inglês, "*high performance computing*- HPC). Pode-se prover cada usuário com acesso a um dispositivo capaz de realizar tais computações, o que é dispendioso, ou usar soluções como computadores de grande porte (*mainframe*), computação em aglomerados de computadores (*cluster*), computação em nuvem (*cloud*) e computação em grade (*grid*).

Uma solução que envolve computadores de grande porte é focada no compartilhamento de uma única grande máquina entre diversos usuários, o que é eficiente e dispendioso. Computação em nuvem é um modelo muito conhecido pelo público geral, englobando plataformas como *Google Cloud* e *Amazon Web Services*. Neste modelo, partes de um sistema estão sob responsabilidade de organizações terceirizadas, tanto na execução de software, quanto na manutenção de *hardware*. Cada organização em geral possui quantidades substanciais dos recursos que disponibiliza. Na computação em grade, um dispositivo envia pedidos de computações para serem executados em máquinas parceiras conectadas à uma rede, a "grade". Contudo, cada membro da rede pode disponibilizar seus dispositivos para computações de outrem, socializando seus recursos de forma à aumentar a capacidade do sistema como um todo. Maquinas numa grade podem operar independentemente, sem um controle central de processamento, o que é necessário no modelo de computação em aglomerados de computadores (*cluster*) (FOSTER et al., 2008).

Uma grade pode ser implementada utilizando apenas computadores como os encontrados em residências e ambientes de trabalho. Esta categoria é denominada "grades de computadores domésticos", de sigla "DCG"para a expressão em inglês *"Desktop Computer Grid"*. Integrando uma DCG, uma organização pode compartilhar e otimizar o aproveitamento do potencial computacional do parque de dispositivos cotidianos que já possui (CEDERSTRÖM, 2010; SCHWIEGELSHOHN et al., 2010; CIRNE et al., 2006).

1.1 Motivação

No contexto de demanda muito superior à oferta, é pouco produtivo que indivíduos desperdicem quantidades significativas de recursos. Contudo, o uso típico de uma máquina doméstica, mesmo na indústria, é caracterizado por longos períodos de relativamente baixa atividade entremeados por surtos de demanda de computações, para as quais os recursos de uma máquina típica são normalmente insuficientes. A capacidade computacional não utilizada de uma maquina durante seu período de baixa atividade é considerada desperdiçada. Este trabalho foi inspirado pelo desejo de aproveitar tal potencial de forma socializada, otimizando o uso de recursos de uma organização ou indivíduo por meio da partilha dos mesmos.

1.2 Problema

No atual ecossistema de projetos DCG, poucos artefatos de engenharia de software (SEAs) são disponibilizados para o público (SCHWIEGELSHOHN et al., 2010). Este é um obstáculo à ser superado para facilitar a análise dos custos de implantação e desenvolvimento deste tipo de soluções.

Trabalhos correlatos são baseados na submissão de códigos portáteis para execução em máquinas de parceiros. Contudo, esta forma de compartilhamento de um dispositivo apresenta riscos, os principais relacionados à privacidade dos dados de cada usuário (ERICKSON, 2008). Informações sensíveis poderiam ser comprometidas, como no caso de vazamentos de memorandos institucionais, instruções governamentais e militares, mensagens, informações e fotos pessoais, dados financeiros e de cartão de crédito, entre outros. É difícil avaliar com precisão os possíveis prejuízos causados pelo vazamento de alguma informação delicada, mas é sabido que os mesmos podem rapidamente atingir níveis preocupantes (CLARKE; KNAKE, 2011).

Diversos sistemas resolvem os problemas de segurança por meio da virtualização do ambiente de execução, o que não só esconde as características individuais de cada máquina (e o potencial computacional das mesmas) como adiciona mais uma camada de abstração, que consome recursos. Contudo, uma vez que os dispositivos de uma grade podem ser bastante heterogêneos, a virtualização dos mesmos pode reduzir a complexidade e maximizar a portabilidade dos softwares executados (THAIN; TANNENBAUM; LIVNY, 2005; ZHAO; LIU; LI, 2011; CIRNE et al., 2006). No entanto, a abordagem baseada na maximização da portabilidade limita a eficiência das computações, pois impossibilita o uso de otimizações de código personalizadas para a arquitetura cada máquina. Tais otimizações podem ter impacto significativo no desempenho de uma aplicação, e podem ter por meio o uso de recursos aceleradores como múltiplos núcleos de processamento (*multi-core* e *many-core*), co-processadores SIMD, ou GPGPUs (FREITAS; GÓES, 2012; OWENS et al., 2007).

1.3 Proposta

Bscando minimizar os problemas descritos na seção anterior, foi desenvolvido neste trabalho um sistema denominado *Assistance*, que será descrito nesta seção.

O sistema *Assistance* permite que proprietários de computadores compartilhem o poder de processamento de seus dispositivos com uma comunidade fechada de parceiros. Um membro da comunidade se beneficia do sistema quando executa uma aplicação para a qual sua máquina não tem suficientes recursos computacionais, como memória e velocidade de processamento. É muito improvável, considerando-se uma comunidade de tamanho suficiente, que em um dado momento nenhum membro da comunidade esteja disponível para compartilhar recursos. É considerado que o uso típico de um computador doméstico na maior parte do tempo requer muito pouco poder computacional, mas com eventuais surtos de demanda frequentemente acima das capacidades do dispositivo.

Assistance é um sistema de computação em grade que contorna os problemas de segurança e limitações de otimização, observados em trabalhos correlatos, com uma única

proposta: evitar completamente a submissão de códigos para serem executados. No lugar de submeterem programas, usuários da grade submetem apenas os dados e parâmetros requeridos por algum algoritmo popular, para o qual uma ou mais máquinas na grade já possuam uma implementação. Assim, cada dispositivo da grade *Assistance* possui um repertório de algoritmos e procedimentos, os quais um dispositivo parceiro pode requisitar execução sobre os dados que enviar. O repertório de um dispositivo pode ser elaborado ou alterado por seu proprietário, melhorando a segurança do sistema e proporcionando maior versatilidade. Como os possíveis programas executados pelo sistema estão limitados ao repertório elaborado por seu mantenedor, *Assistance* se enquadra na categoria de grades de software como serviço (em inglês, *software as a service*, SaaS), conceito abordado no capítulo de revisões bibliográficas.

Assistance não necessita de um servidor central para intermediar as comunicações entre dispositivos da grade, permitindo que troquem mensagens diretamente. Portanto, pode ser categorizado como uma grade P2P-SIP (do inglês, *peer-'to'-peer*, de parceiro à parceiro, utilizando algum tipo de *session iniciation protocol*, protocolo de iniciação de sessões) (BRYAN; LOWEKAMP; JENNINGS, 2005). Uma vez que *Assistance* permite que cada dispositivo execute sua própria implementação de um algoritmo, o código de tal implementação será compatível e otimizado para a arquitetura da máquina que o executa. Isso permite que o sistema seja eficiente e viável em computadores muito diferentes, inclusive *consoles* de *videogames*, *laptops*, *tablets*, *smartphones*, e mesmo computadores industriais.

O sistema *Assistance* tem baixo risco de servir de canal para invasões remotas dos sistemas em que está instalado, uma vez que não permite a transmissão de códigos executáveis. Sua arquitetura horizontal proporciona escalabilidade e flexibilidade de implantação e utilização. Assim, *Assistance* pode ser descrito como uma plataforma pessoal e distribuída, P2P-SIP, para a execução de software como serviço.

As aplicações para as quais *Assistance* é mais indicado são as que realizam processamento intenso, de algoritmos populares, sobre blocos de dados de baixa dependência entre si. Tais aplicações são denominadas *computacionalmente intensas* (ZHAO; LIU; LI, 2011). O desempenho de tais aplicações seria beneficiada pela abundancia de poder computacional na grade, caso as necessidades de comunicação de dados sejam pequenas o bastante para que a latência da rede não inviabilize a execução. Diversos algoritmos utilizados em aplicações de inteligência artificial podem ser considerados computacionalmente intensos.

1.4 Justificativa

Não foi encontrada, na busca por trabalhos correlatos, menção a qualquer sistema de computação, em grade, que utilizasse um paradigma de passagem de mensagens. Todos os projetos encontrados propõem que cada usuário distribua seus próprios códigos portáteis para execução em maquinas de parceiros, o que complica o uso de recursos não-tradicionais de processamento, como GPGPUs e multiprocessadores. Os objetivos deste trabalho foram estabelecidos buscando esclarecer tanto quanto possível o paradigma de computação em grade por meio de passagem de mensagens, e a viabilidade de seu desempenho em aplicações computacionalmente intensivas, como as de inteligência artificial.

2 REFERENCIAL TEÓRICO

Neste capítulo serão apresentados os principais conceitos e trabalhos relacionados nas areas de computação na nuvem, grades de computadores, redes ponto-à-ponto, e inteligência artificial. Será discutida a taxonomia do presente trabalho, e os tipos de testes usados para medir seu desempenho.

2.1 Computação na Nuvem

Nesta seção será apresentado o conceito de computação em nuvem, seus diferentes subtipos e taxonomia, e uma análise da taxonomia de trabalhos correlatos e do presente.

Diversos trabalhos analisados afirmam a dificuldade e falta de consenso em se definir o conceito de computação na nuvem (*Cloud Computing*) (ZHANG; CHENG; BOUTABA, 2010; RIMAL; CHOI; LUMB, 2009; FOSTER et al., 2008). Vaquero et al. (2008) analisaram mais de vinte definições do conceito. Apesar de bastante delicada a questão da definição do que a computação em nuvem *é*, parece haver consenso no que a computação em nuvem *busca*. O "ideal"da computação na nuvem parece ser o de prover à seus usuários cômodos recursos de terceiros, em quantidades dinamicamente escaláveis e economicamente viáveis.

Soluções envolvendo o compartilhamento de um sistema centralizado de grande poder computacional (também chamadas de "soluções *mainframe*"), ou então computação em agregado de computadores (soluções de *Cluster Computing*), e mesmo soluções de computação em grade (*Grid Computing*) distribuem as computações necessárias para diversas aplicações em diferentes dispositivos, por vezes fisicamente muito distantes. Por buscarem prover apoio computacional para dispositivos de outrem, estas diferentes soluções podem ser consideradas implementações do ideal da computação na nuvem.

Uma solução do modelo *mainframe* é baseada no compartilhamento de um dispositivo central entre diversos usuários, o que torna essa solução bastante eficiente, mesmo que dispendiosa. *Cloud computing* é um modelo no qual partes do software ou hardware de um sistema estão sob responsabilidade de organizações terceirizadas. Cada organização em geral tem acesso à quantidades substanciais dos recursos que disponibiliza. Em *Grid computing*, cada membro deste tipo de rede, denominado "grade", pode disponibilizar, à sua vontade, dispositivos para a realização de computações de outrem, operando independentemente sem algum dispositivo central de controle de execução. Tal controle é necessário no modelo de *cluster computing*, o que em geral evita sua utilização em um parque de dispositivos distribuídos por grandes áreas e longas distancias. (FOSTER et al., 2008; KAHANWAL; SINGH et al., 2013)

Os trabalhos de Rimal, Choi e Lumb (2009), Zhang, Cheng e Boutaba (2010), Vaquero et al. (2008) e Kahanwal, Singh et al. (2013) descrevem uma estratificação dos tipos de computação na nuvem em três, e em alguns casos até quatro camadas. Tal hierarquização está representada no quadro 1. A ordem da inclusão das camadas reflete o nível da abstração que representam, sendo as camadas superiores mais próximas da interface com o usuário, e as camadas mais inferiores mais próximas do hardware físico. O jargão para tal conceito é "*closer to the metal*", do inglês, "mais próximo do metal", referindo-se ao

controle direto das peças que compõem o dispositivo.

Quadro 1 - Camadas da computação em Nuvem (X-aaS)

Software como Serviço (*Software as a Service* - SaaS)
Plataforma como Serviço (*Platform as a Service* - Paas)
Infraestrutura como Serviço (*Infrastructure as a Service* - IaaS)
Hardware como Serviço (*Hardware as a Service* - HaaS)

Fonte: (VAQUERO et al., 2008; RIMAL; CHOI; LUMB, 2009)

SaaS se refere ao atendimento de diversos clientes, disponibilizando para cada um uma instância de uma mesma aplicação, executada por um mesmo conjunto de recursos. Normalmente, uma entidade prestadora de serviços é contratada por um cliente que deseja a provisão de certa aplicação. A prestadora subcontrata suas necessidades de recursos para entidades análogas que atuam nas camadas inferiores.

PaaS busca prover aos desenvolvedores de aplicações uma plataforma sobre a qual executá-las, com todos os recursos necessários para o estabelecimento de um ambiente de execução de software, em quantidades ajustadas dinamicamente. Tais recursos estão no mesmo nível de abstração que um sistema operacional e seu gerenciamento de *threads* e discos, ou de um mecanismo de controle de instâncias de software.

IaaS - HaaS tem por objetivo fornecer ao usuário recursos básicos, como armazenamento e processamento. Tais conceitos abstratos são os recursos providos pela "infraestrutura"como serviço (IaaS), enquanto um sistema de "hardware"como serviço (HaaS) disponibiliza dispositivos, especificados por modelo e capacidade.

Como será descrito nas próximas seções deste capítulo, trabalhos correlatos apresentam plataformas para a disponibilização de ambientes de execução para códigos submetidos por parceiros, cujas computações são distribuídas entre o dispositivo requerente e seus assistentes, da forma que Mateescu, Gentzsch e Ribbens (2011) chamariam "hibrida". Tal arquitetura poderia ser categorizada como sendo do tipo PaaS. Outros trabalhos apresentam formas de compartilhamento de armazenamento ou memória, que poderiam ser descritas como do tipo IaaS. Este trabalho propõe uma solução de mais alto nível, onde dispositivos na grade não oferecem ambientes virtuais de execução, mas as vias de entrada e saída de aplicações inteiras, para a execução de algoritmos e recuperação de resultados via um ambiente de terminal de sistema. A categoria que este trabalho se encaixa é a de software como serviço (SaaS).

O sistema *Assistance* implementa o ideal da computação em núvens, no modelo SaaS, por meio de uma grade de computadores.

2.2 Computação em Grade

Nesta seção, será apresentado e discutido o conceito de grades de computadores, seus subtipos, e suas características. Também serão discutidos testes realizados em trabalhos correlatos, e quais poderiam ser aplicados ao presente trabalho.

Não há um consenso absoluto na definição do termo "grade de computadores". Diversos trabalhos correlatos compreendem o termo diferentemente. Cederström (2010) define uma grade de computadores como uma solução para controlar e coordenar o compartilhamento e uso de recursos computacionais, formando uma organização virtual dinâmica

e escalável. Schwiegelshohn et al. (2010) compreendem uma grade como um sistema distribuído que cria ambientes de pesquisa virtuais entre diferentes instituições, o que acaba por incluir nesta definição os sistemas DNS (*Domain Name System*, sistema de nomenclatura de domínio), o que é um dos serviços elementares da internet (TANENBAUM, 2011). Thain, Tannenbaum e Livny (2005), responsáveis pelo comumente utilizado sistema de computação em grade CONDOR, definem "grade de computadores"como um sistema de computação distribuída em que cada usuário permanece livre para contribuir o quanto desejar.

Neste trabalho, uma grade de computadores será compreendida como um sistema que provê um ambiente para a realização de computações distribuídas entre dispositivos membros de certa comunidade, possivelmente geograficamente e administrativamente separados, e um dispositivo requerente, que coordena as execuções que submeteu à comunidade. Esta definição está sujeita as seguintes restrições:

- Nenhum participante garante sua disponibilidade na grade, podendo se desconectar a qualquer momento, ou manter conectividade limitada à algumas atividades ou parceiros;

  Corolário: não necessariamente há conectividade entre todos os dispositivos da grade, em qualquer dado momento;

- Todo participante tem autoridade máxima sobre os dispositivos de que é mantenedor, podendo configurá-los e utilizá-los como e quando desejar.

  Corolário 1: O administrador de alguns dispositivos pode interromper processos de outrem que neles estejam sendo executados à qualquer momento;

  Corolário 2: Não existe, necessariamente, uma autoridade máxima na administração da grade.

  Corolário 3: Não existe, necessáriamente, homogeneidade na arquitetura dos dispositivos da grade;

- Qualquer participante pode disponibilizar qualquer tipo de recursos para a comunidade;

- Qualquer participante pode submeter trabalhos computacionais para serem executados por quaisquer maquinas da rede, que tem autonomia para decidir se o farão;

Não somente organizações bem estruturadas poderiam utilizar de grades de computadores para o problema da insuficiência de recursos. Indivíduos podem também participar numa grade, utilizando-se somente de computadores domésticos. Em 31 de dezembro de 2013, três quartos de todos os cidadãos de países desenvolvidos tinham acesso regular à um destes (MINIWATTS MARKETING GROUP, 2013). A este tipo específico de grades se nomeia "grades de computadores domésticos", de sigla "DCG"para a expressão em inglês "*Desktop Computer Grid*".

2.2.1 Grades de Computadores Domésticos

DCGs são bastante similares à grades de computadores industriais. Os dois tipos de grades enfrentam problemas de compartilhamento e localização de recursos, segurança,

latência da rede, gerenciamento de execução, dentre outros. Contudo, DCGs também tem de solucionar problemas advindos da mobilidade de seus usuários, que podem estar compartilhando ou requirindo recursos à partir de dispositivos portáteis como laptops e tablets, ou através de seções instáveis e lentas da internet. Numa DCG os recursos tendem a ser mais heterogêneos do que numa grade tradicional, devido ao grande número de diferentes arquiteturas implementadas nos sistemas disponíveis no mercado. Os usuários de dispositivos domésticos normalmente não tem muito conhecimento técnico, e dificuldades de implantação e uso de sistemas de grade podem desestimular seu ingresso numa DCG.

Neste trabalho, é proposta uma solução de fácil implantação, compatível com diversas arquiteturas de máquina e redes, sendo, portanto, compatível com as necessidades específicas de uma DCG.

Thain, Tannenbaum e Livny (2005) propuseram um sistema para DCG chamado "CONDOR", no qual usuários submetem programas (modelados na linguagem ClassAds) à um módulo “agente”, que considera prioridades arbitrariamente definidas para decidir qual máquina da rede irá executar tais programas. Por rasões de segurança, a execução é realizada num “ambiente sandbox”, uma máquina virtual que permite isolamento entre os dados do proprietário do dispositivo e os programas de outrem. Condor monitora a execução dos programas submetidos, e lida com suas falhas, otimizando a utilização dos recursos da rede.

Uma solução bastante parecida é proposta por Cirne et al. (2006), que desenvolveram o sistema "Our Grid", que permite que seus usuários submetam à rede programas em JAVA ou uma linguagem de domínio desenvolvida para este propósito, para que sejam executados por qualquer máquina da comunidade. "Our Grid"realiza as execuções dos programas submetidos em um ambiente de máquinas virtuais, e não permite que usuários criem suas próprias comunidades restritas, tendo por objetivo permitir colaborações em grandes comunidades. Estes recursos estão disponíveis no sistema XtremWeb (FEDAK et al., 2001), que, no entanto, envolve consideráveis complicações nos quesitos de implementação e implantação.

Cederström (2010) realizou uma revisão sistemática de literatura (*systematic literature review* - SLR) comparando diferentes sistemas de DCG, realizou uma pesquisa comprovando o interesse da comunidade de empresas desenvolvedores de software em soluções de DCG, e desenvolveu uma aplicação de testes utilizando o sistema BOINC (ANDERSON, 2004), da Universidade da California em Berkley. Usuários de BOINC podem escolher diversos projetos disponibilizados no site da ferramenta para submeter e receber pedidos de realização de computações. Todos os pedidos devem seguir os padrões do projeto. Por motivos de segurança, as execuções dos programas são realizadas em uma máquina virtual. Esta abordagem é parecida com a do projeto Globus (FOSTER; KESSELMAN, 1997), que busca integrar outros tipos de grades e serviços em uma DCG.

Analisando-se os trabalhos correlatos acima descritos, pode-se verificar que a abordagem tradicional para sistemas de computação em grades heterogêneas envolve a virtualização dos recursos e arquiteturas das máquinas que realizam as computações.

É possível se observar que a rede que liga dispositivos de uma grade é uma parte muito importante da mesma. Em especial, o tipo de rede que permite comunicações virtualmente diretas entre dois dispositivos de uma grade. Estas são as chamadas "redes P2P".

2.3 Redes P2P

Tanenbaum (2011) analisa redes P2P tomando como exemplo seu uso mais frequente, o compartilhamento de arquivos, em especial pelo protocolo BitTorrent (COHEN, 2008), que corresponde por uma larga fração de todo trafego de dados da rede. Os computadores da rede, que podem ser industriais, domésticos ou de qualquer outro tipo, são chamados de "*peers*", ou "parceiros", "nós", ou mesmo "pontos"porque podem agir como cliente, fazendo pedidos, ou servidor, atendendo à pedidos de outrem. A rede em si não requer uma infra-estrutura dedicada, o que faz deste sistema muito escalável. Schollmeier (2001) busca mais diretamente a definição do conceito de redes P2P, as compreendendo como redes nas quais parceiros compartilham algum recurso de hardware (como acesso de rede e armazenamento). O acesso à este recurso não deve estar sujeito à autoridade de um terceiro. Estas qualificações são necessárias e suficientes para se categorizar uma rede como P2P. Para que uma rede seja considerada P2P "pura", a mesma deve suportar a remoção de qualquer um de seus elementos sem tornar-se incapaz de realizar seus serviços. Uma rede P2P "hibrida"envolve ao menos um elemento que de alguma forma é mais "central", inviabilizando a rede caso seja removido.

Schollmeier (2001) também afirma que muitas vezes trabalhos acadêmicos não descrevem claramente a forma como interpretam o conceito de P2P, e acabam por descrever estruturas que, segundo outros autores, não se encaixariam na categoria. Para facilitar a compreensão, este trabalho adota a nomenclatura "P2P-SIP".

2.3.1 P2P-SIP

Bryan, Lowekamp e Jennings (2005) descreve um sistema de comunicação P2P que envolve, momentaneamente, um elemento que não os dois que se comunicam. O trabalho deste terceiro é permitir que os dois pontos da comunicação estabeleçam uma conexão entre si. Assim que isto é feito, o elemento mediador interrompe sua participação. Este processo é denominado "protocolo de iniciação de sessão", em inglês, "session iniciation protocol- SIP. Assim, este tipo de sistema é iniciado como membro da categoria de redes P2P "híbrida", e então se converte num sistema P2P "puro".

2.4 Inteligência Artificial

Inteligência artificial (*artificial intelligence* - AI) é um conceito complexo, central de um campo que abrange ideias de filosofia, matemática, linguística, cibernética, computação, cognição, psicologia, neurociência, e ainda outros (RUSSELL; NORVIG, 2009). Se tornou uma indústria a partir dos anos 80 (MCDERMOTT, 1982 apud RUSSELL; NORVIG, 2009). Em 2001, o uso te sistemas de inteligência artificial tornou-se lugar-comum no emergente campo de análise de grandes quantidades de dados, também chamada "ciência de *BigData*"(WARD; BARKER, 2013), descrito por Baker (2009) como bastante promissor para a realização de analises financeiras, políticas e sociais.

A definição de inteligencia artificial é por si só um tema complexo. A lista abaixo,

uma tradução livre do quadro taxonomia de Russell e Norvig (2009), mostra citações de autores de diferentes escolas de pensamento sobre AI.

1. Pensar Como um Ser Humano

   "O novo e exitante esforço em fazer computadores *pensarem ... máquinas com mentes*, no sentido completo e literal"(HAUGELAND, 1989 apud RUSSELL; NORVIG, 2009)
   "[A automação de] atividades normalmente associadas com o pensamento humano, tais como tomada de decisões, solução de problemas, aprendizado..."(HELLMAN, 1978 apud RUSSELL; NORVIG, 2009)

2. Pensar Racionalmente

   "O estudo de faculdades mentais por meio de modelos computacionais"(CHARNIAK, 1985 apud RUSSELL; NORVIG, 2009)
   "O estudo das computações que fazem possível os atos de perceber, racionar, e agir"(WINSTON, 1992 apud RUSSELL; NORVIG, 2009)

3. Agir Como um Ser Humano

   "A arte de criar máquinas que realizam funções que requerem inteligência quando realizadas por seres humanos"(KURZWEIL; SCHNEIDER; SCHNEIDER, 1990 apud RUSSELL; NORVIG, 2009)
   "O estudo de como fazer computadores fazerem coisas nas quais, atualmente, pessoas são melhores"(RICH; KNIGHT, 1991 apud RUSSELL; NORVIG, 2009)

4. Agir Racionalmente

   "O estudo do design de agentes inteligentes"(POOLE; GOEBEL; MACKWORTH, 1998 apud RUSSELL; NORVIG, 2009)
   "... envolvida com o comportamento inteligente de artefatos"(NILSSON, 1998 apud RUSSELL; NORVIG, 2009)

Na lista acima, os itens impares consideram o sucesso de um sistema de inteligência artificial pelo quanto se parece com um ser humano, e as pares pelo quanto o mesmo aparenta ser racional. Os dois primeiros itens focam nos processos de pensamento, e os dois últimos no comportamento das inteligências artificiais.

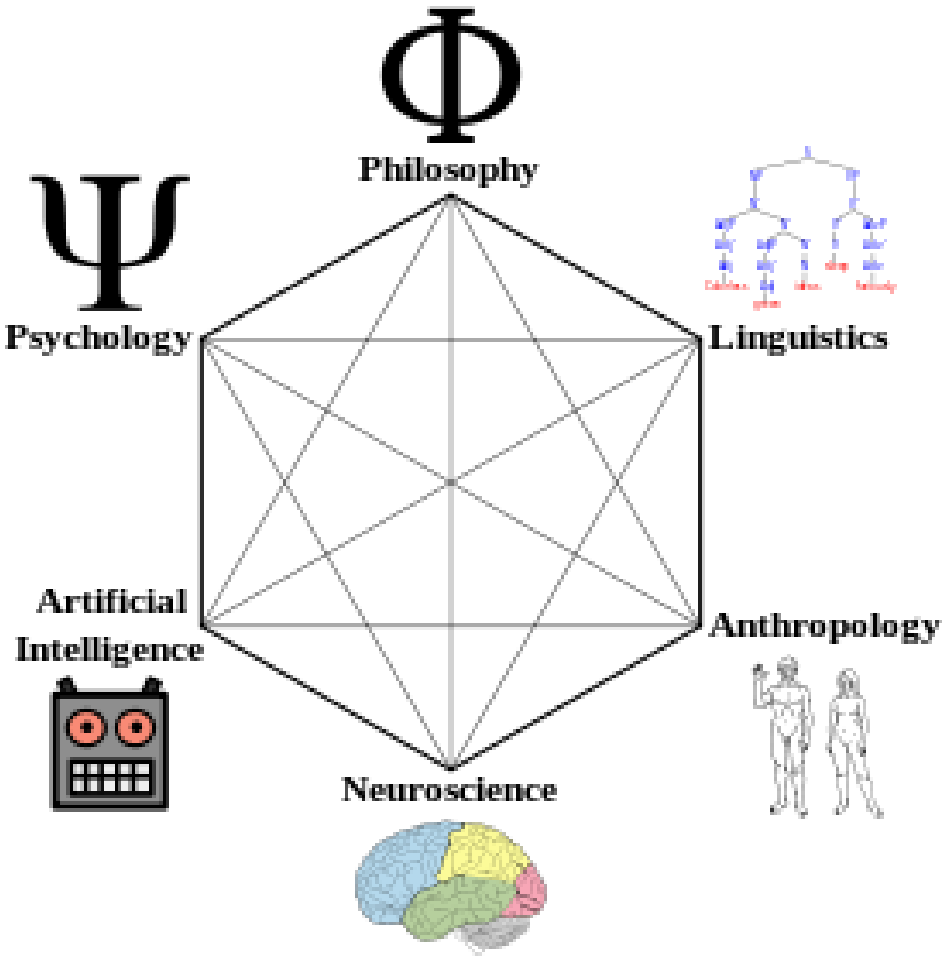
A abordagem de Turing (1950) segue o paradigma do item 3 do quadro de taxonomia de Russell, considerando um sistema como uma autêntica inteligência artificial caso consegua se passar por humano em um ambiente controlado de testes. Este objetivo original acabou sendo rebatizado de "inteligência artificial geral"(AGI), por englobar as diversas especializações do campo de AI que surgiram ao longo do tempo, como visão computacional, processamento de linguagem natural, aprendizado de máquina e mineração de dados (AREL; LIVINGSTON, 2009).

A diferenciação, na lista acima, entre "agir como um ser humano"(item 3 do quadro de taxonomia de Russell) e "pensar racionalmente"(item 2) se deve ao fato de seres humanos frequentemente cometerem erros, muitas vezes sistemáticos (EMBREY, 1986).

Tais erros podem ter por causa cultura, cansaço, emoções, etc. O filósofo grego Aristóteles foi a primeira pessoa da história conhecida à tentar "racionalizar"a forma de pensar, tendo escrito diversos tratados buscando "estruturar"o raciocínio. O conjunto de obras de Aristóteles que tratam sobre "silogismos- formulações lógicas que sempre são verdadeiras, sendo verdadeiras suas premissas - são comumente chamadas de "Organon", "o orgão"(ARISTOTLE; OWEN, 1853). Mais de dois mil anos de desenvolvimento do pensamento grego levaram a elaboração do item 2 da lista acima. Tal paradigma considera AI como a capacidade de um sistema de explorar premissas e preposições lógicas, desenvolvendo raciocínios e inferindo respostas.

O item 1 da lista acima, "pensar como um ser humano", representa o paradigma de AI que se relaciona com as chamadas "Ciências Cognitivas"(WIKIPEDIA, 2014). Neste paradigma, busca-se modelar o pensamento humano por meio de modelos lógicos e matemáticos, para que se possa estudar e entender o raciocínio em si. Inteligência artificial é apenas um dos diversos campos de estudo relacionados com tais ciências, como está ilustrado na figura 1.

Figura 1 – Ciências relacionadas às Ciências Cognitivas



Fonte: (WIKIPEDIA, 2014)

Neste trabalho, será seguida principalmente a escola de pensamento ilustrada pelo item 4 do quadro de taxonomia de Russell (Pensar Racional), em que inteligência artificial é compreendida como a composição e estudo de sistemas capazes de analisar o contexto em que se encontram, tomar decisões, e reagir de acordo. Este tipo de sistema é frequente-

mente nomeado "agente". Agentes podem ser facilmente modelados matematicamente, empregando modelos mais naturais e intuitivos que os modelos de inferência lógica pura.

Bishop et al. (2006), descreve uma sub-área da AI, do paradigma do item 4 do quadro de taxonomia de Russell, chamada *machine learning*, aprendizado de máquina. Neste campo de estudos são desenvolvidos, utilizados e observados algoritmos e sistemas que necessitam de uma fase de *treinamento* antes de estarem prontos para serem executados. Nesta fase, variáveis internas do algoritmo são ajustadas, tal qual um músico afina seu instrumento antes de tocar. Contudo, nesta analogia, o instrumento se afina sem precisar da ajuda do músico. O *aprendizado*, neste caso, está intimamente relacionado com o aperfeiçoamento, a melhora do desempenho do "aprendiz". O conceito da "aprendizado"não deve ser confundido com o conceito de "treinamento", caso no qual os objetivos do aperfeiçoamento são antes do "treinador"que do "treinado".

O aprendizado de máquina é necessário em aplicações que envolvem análises de dados dos quais não se tem uma ideia muito clara do significado. Como colocaram Witten e Frank (2005), "A medida que o volume de dados aumenta, diminui, alarmante e inexoravelmente, a fração dos mesmos que é entendida"(WITTEN; FRANK, 2005, p. 4, tradução do autor)[1]. Logo, em grandes bases de dados, máquinas tem de ser usadas para se extrair informações, na forma de padrões. Essas máquinas têm de ser capazes de aprender como interpretar tais dados. A atividade de interpretação de padrões em bases de dados é chamada "mineração de dados"(*data mining*). Witten e Frank (2005) descrevem "Mineração de dados"como:

> Mineração de dados é definida como o processo de se descobrir padrões em dados. O processo necessariamente deve ser automatizado, ou, mais comumente, semi-automático. Os padrões descobertos tem de ser significativos, propiciando alguma vantagem, normalmente econômica. Invariavelmente, há substanciais quantidades de dados. (WITTEN; FRANK, 2005, p. 5, tradução do autor)[2].

Existem muitos algoritmos para mineração de dados. A execução dos mesmos normalmente consome consideráveis quantidades de recursos computacionais, e, na insuficiencia destes, tempo.

2.4.1 A Biblioteca WEKA de Mineração de Dados

A biblioteca WEKA (acrônimo para *Waikato Environment for Knowledge Analysis* - O Ambiente de Análise de Bases de Conhecimento da Universidade de Waikato, na Nova Zelândia) (HALL et al., 2009), desenvolvida na linguagem JAVA de programação, oferece diversos algoritmos de mineração de dados já implementados e extensivamente testados. WEKA é muito utilizada em pesquisas científicas de diversas áreas, pois permite que os algoritmos sejam utilizados sem exigir que a equipe de pesquisa tenha de se dedicar ao desenvolvimento do software dos algoritmos. Hall et al. (2009) descrevem duas formas de se utilizar WEKA: utilizando a interface gráfica incluida com a bilioteca, ou evocando a

[1] As the volume of data increases, inexorably, the proportion of it that people understand decreases alarmingly.

[2] Data mining is defined as the process of discovering patterns in data. The process must be automatic or (more usually) semiautomatic. The patterns discovered must be meaningful in that they lead to some advantage, usually an economic one. The data is invariably present in substantial quantities.

mesma como um programa independente pela linha de comandos do sistema.

Este trabalho não tem como objetivo a implementação de algum algoritmo de mineração de dados, mas sim a execução distribuída de várias instâncias independentes de diferentes algoritmos. Assim, a biblioteca WEKA é utilizada como implementação dos algoritmos de mineração de dados. Os testes da solução proposta neste trabalho incluirão o uso da biblioteca WEKA para executar os desafios de validação (*benchmarks*) propostos por Narayanan et al. (2006). Estes consistem numa série de bases de dados sobre as quais já foram executados diversos algoritmos de mineração, e cujos resultados são conhecidos, de forma que podem ser comparados com os resultados obtidos nos testes do presente trabalho. A plataforma desenvolvida no presente trabalho, Assistance, também será usada para repetir os experimentos de Leijoto (2014), que utilizou a biblioteca WEKA para analizar e identificar a função de proteínas. Este trabalho não tem como objetivo melhorar os resultados dos experimentos de Leijoto, mas sim comparar o tempo de execução desta aplicação real da forma como foi implementada, e na forma distribuida da presente proposta.

3 METODOLOGIA

Neste capítulo será apresentada a metodologia utilizada no desenvolvimento deste trabalho. Apresentados artefatos de engenharia de software, necessários para uma melhor compreensão do funcionamento do sistema *Assistance*, podem ser encontrados no anexo A. Descrições das implementações de cada parte do sistema podem ser encontradas no anexo B. Ao final deste capítulo, será apresentada uma descrição dos testes e métricas utilizadas para avaliação e comparação do sistema com trabalhos correlatos.

Inicialmente foram elaborados diversos artefatos de engenharia de software. Segundo Basili et al. (2007), os SEAs permitem ao leitor uma visão mais profunda do sistema, e uma melhor compreensão das responsabilidades de cada unidade envolvida, sendo sua uma parte importante para a compreensão científica de novas propostas de software. Thain, Tannenbaum e Livny (2005) acrescenta que a principal razão do sucesso e persistência do sistema CONDOR, citado no capítulo de referencial teórico, foi uma clara compreensão, por todos os envolvidos no desenvolvimento e utilização do sistema, das responsabilidades de cada uma de suas partes.

Buscando estabelecer da forma mais clara o possível o funcionamento do sistema *Assistance*, foram elaborados, na ordem apresentada, os seguintes SEAs:

1. Diagrama de Casos de Uso;
2. Lista de Requisitos;
3. Diagrama de Comunicação;
4. Diagrama de Sequência de Atividades;

Após a elaboração de cada parte do sistema, que estão descritas neste capítulo, foram elaborados testes para as implementações de cada conjunto de unidades relacionadas no diagrama de comunicação. Tais testes de unidade, descritos em mais detalhes no anexo B, verificam o funcionamento de cada bloco indivisível da implementação do sistema. Foram elaborados testes para o sistema como um todo, denominados "testes de aplicação"(ROGER, 2005).

Após atestar a *eficácia* do sistema *Assistance*, foi medida sua *eficiência* (ISO, 2000), por meio da execução de casos de teste populares, elaborados e divulgados para que o desempenho de uma aplicação possa ser comparado ao de outras. Este tipo de testes é denominado "testes de *benchmark*"(SNAVELY et al., 2003).

Os testes do sistema foram divididos em três estágios de maturidade - Baixa, Média e Alta. Os testes de cada estágio foram categorizados como do tipo Alpha ou do tipo Beta. Mais detalhes podem ser encontrados na seção de testes deste capítulo.

Em cada estágio da maturidade do sistema, *Assistance* foi implantado em diferentes dispositivos e ambientes, e submetido aos mesmos testes do tipo Alpha. Os testes do tipo Beta ocorrem de forma distinta em cada estágio.

Ao obter uma medida de desempenho suficiente em cada estágio de testes, o sistema foi considerado pronto para passar para o próximo. As métricas utilizadas nos testes estão descritas na seção de testes deste capítulo.

3.1 Algoritmos de Inteligência Artificial

Nesta seção, será descrita a biblioteca de algoritmos de inteligência artificial implementados no sistema *Assistance*.

Na implementação de *Assistance* desenvolvida neste trabalho, foi utilizada a biblioteca WEKA de mineração de dados por meio de inteligência artificial (HALL et al., 2009). Estes dois temas (mineração de dados e AI) são expostos mais detalhadamente no capítulo de revisão bibliográfica. A biblioteca WEKA foi selecionada para este trabalho por ser uma biblioteca utilizada em diversos trabalhos correlatos, o que facilita a realização de uma análise comparativa dos resultados. WEKA foi implementada na linguagem de programação JAVA (GOSLING; MCGILTON, 2003), amplamente utilizada, e notória pela portabilidade inerente de aplicações desenvolvidas para a mesma, quase completamente independentes no que se refere, entre outros, à arquiteturas de processadores, memórias, e sistemas operacionais.

A biblioteca WEKA contém diversos algoritmos de inteligência artificial, organizados em diversas categorias. Na implementação do sistema *Assistance* descrita neste trabalho, todos os algoritmos da biblioteca foram disponibilizados para execução.

3.2 Testes de Aplicação

Nesta seção, serão apresentados os testes do sistema *Assistance* que consideram a implementação do mesmo estável e capaz de produzir resultados coerentes de forma confiável. Os testes de um sistema completamente implementado e preparado serão chamados, neste trabalho, de testes de aplicação, nomenclatura coerente com a de Roger (2005). Primeiramente, será descrita a forma de organização dos testes, e, em seguida, seus parâmetros.

3.2.1 Estágios e Tipos de Teste

O estado de desenvolvimento do sistema foi divididos em três estágios de maturidade - Baixa, Média e Alta. Neste trabalho, será considerado de baixa maturidade um sistema não está pronto para ser implantado e executado sem intensiva, constante e presencial supervisão de seus desenvolvedores, e por isso um teste neste estágio deve ser feito em baixa escala. Um sistema de baixa maturidade muito provavelmente terá de ser alterado para ter sucesso nos testes executados. Um sistema será considerado de maturidade mediana se houver pouca preocupação quanto a estabilidade do funcionamento e baixa dificuldade de implantação. Num sistema de maturidade mediana, é testada sua a capacidade de operar por períodos de tempo mais prolongados e em maior número de dispositivos, não necessariamente com a presença física da equipe de desenvolvedores no sítio da operação. Um sistema será considerado neste trabalho como de alta maturidade deverá ser capaz de operar com um grande número de máquinas por prolongados períodos de tempo sem intervenção direta do desenvolvedor.

Em cada estágio de maturidade do sistema são realizados testes divididos em duas categorias, Alpha e Beta. Testes do tipo Alpha focam em verificar se uma instância do

sistema, implantada em um novo ambiente, está estável e produz respostas coerentes. Testes do tipo Beta verificam a escalabilidade do sistema no ambiente em que foi implantado nos testes Alpha do mesmo estágio.

Em cada estágio da maturidade do sistema, Assistance foi implantado em um diferente número de dispositivos e sistemas operacionais, e diferentes topografias de redes. Mais detalhes sobre os dispositivos e redes utilizadas, e sobre a forma de implantação, podem ser encontrados no anexo B.

O quadro 1 provê mais detalhes sobre a escala de cada bateria de testes, por tipo e estágio.

Para os estágios de teste intermediários, foram utilizados computadores do Centro de Recursos Computacionais do Instituto de Ciências Exatas e Informática da Pontífica Universisdade Católica de Minas Gerais.

Para os testes do tipo Beta de alta maturidade, convites para testar o sistema foram divulgados nas redes sociais das quais os autores deste trabalho fazem parte. Cada participante do teste deve baixar para sua maquina uma versão do sistema *Assistance* que, em horários pré-determinados, executou automaticamente os testes especificados. O software também gravou LOGs sobre sua execução, submetendo-os automaticamente à um repositório central em intervalos regulares. A análise destes dados pode ser encontrada no capítulo de resultados.

Quadro 1 - Testes por Estágio e Tipo

Escala das baterias de teste em cada estágio e tipo		
ESTÁGIO	**TIPO**	**ESCALA** & outros
Baixa Maturidade	Alpha	1 Computador
Baixa Maturidade	Beta	Computadores na mesma rede local utilizando o mesmo sistema operacional
Baixa Maturidade	Beta	3 Computadores na mesma rede local utilizando diferentes sistemas operacionais
Média Maturidade	Alpha	3 Computadores em diferentes redes utilizando diferentes sistemas operacionais
Média Maturidade	Beta	10 Computadores em diferentes redes utilizando diferentes sistemas operacionais, operando continuamente por 6 horas
Média Maturidade	Beta	20 Computadores em diferentes redes utilizando diferentes sistemas operacionais, operando continuamente por 72 horas
Alta Maturidade	Alpha	100 Computadores em diferentes redes utilizando diferentes sistemas operacionais, operando esporadicamente e de forma assíncrona por 14 dias
Alta Maturidade	Beta	>100 Computadores em diferentes redes utilizando diferentes sistemas operacionais, operando esporadicamente e de forma assíncrona por 28 dias

Fonte: Próprio Autor

3.2.2 Parâmetros de cada teste

Os testes realizados nos primeiros dois estágios do quadro 1 consistiam em repetir os experimentos realizados por Leijoto (2014), uma vez que esta autora disponibilizou os códigos-fonte dos softwares utilizados em seu trabalho. Os softwares foram instalados e tiveram seu tempo de execução medido em cada máquina participante do teste, sem o uso do sistema *Assistance*. Os códigos do trabalho utilizado como teste foram modificados para fazer uso do sistema *Assistance* em cada linha na qual referenciavam a biblioteca WEKA, e foram executados no ambientes de rede descritos no quadro 1 e anexo B, tendo o tempo de execução medido. Os resultados das duas formas de execução foram comparados em cada teste, para verificar sua equivalência, provando a eficácia do sistema.

Os tempos de execução, em cada tipo e ambiente, foram plotados, de forma à se poder comparar a eficiência do sistema. Depois, foi utilizada regressão quadrática nos valores do tempo de execução dos experimentos no sistema *Assistance*, verificando-se a proximidade dos mesmos com os resultados ideais. Estas informações serão apresentadas com mais detalhes no capítulo de Resultados deste trabalho.

4 RESULTADOS ESPERADOS

Neste capítulo serão descritos os resultados esperados para cada bateria de testes apresentada no quadro 1 do capítulo sobre metodologias. O quadro 2, abaixo, contém os resultados de tais testes, mostrando o tempo de execução de cada bateria em relação ao tempo de execução do teste do sistema Assitance em uma única máquina.

Os resultados parecem de acordo com a fórmula do tempo de execução, em que o ganho relativo de tempo, chamado *SpeedUp*, é o resultado da soma do tempo de execução da parte não-distribuível da execução com o resultado da divisão do tempo de execução máximo estimado de cada parte distribuível sobre o número de dispositivos participantes da execução, menos um, que, no pior caso, executa apenas as partes não distribuíveis. Este resultado deve ser somado ao dobro do maior tempo de comunicação de dados entre o dispositivo que originou as computações e um de seus parceiros. A formula abaixo descreve tal relação.

$$SpeedUp = (T_{ndist} + \frac{T_{dist}}{N_P - 1} + 2 * (N_P - 1) * Max(T_{P_0,P_N}))/T_1 \tag{4.1}$$

Quadro 2 - Resultados esperados para cada bateria de testes

Resultado Esperado das baterias de teste em cada estágio e tipo		
ESTÁGIO	**TIPO**	**RESULTADO ESPERADO**
Baixa Maturidade	Alpha	X segundos para a execução
Baixa Maturidade	Beta	4X/5 segundos para a execução
Baixa Maturidade	Beta	4X/5 segundos para a execução
Média Maturidade	Alpha	7X/5 segundos para a execução
Média Maturidade	Beta	5X/8 segundos para a execução
Média Maturidade	Beta	X/2 segundos para a execução
Alta Maturidade	Alpha	X/2 segundos para a execução
Alta Maturidade	Beta	X/2 segundos para a execução

Fonte: Próprio Autor

Referências Bibliográficas

ANDERSON, David P. Boinc: A system for public-resource computing and storage. In: IEEE. **Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on**. [S.l.], 2004. p. 4–10.

AREL, Itamar; LIVINGSTON, Scott. Beyond the turing test. **Computer**, IEEE, v. 42, n. 3, p. 90–91, 2009.

ARISTOTLE; OWEN, translated by Octavius Freire. **Organon, Complete Edition**. [S.l.: s.n.], 1853.

BAKER, Stephen. **NUMERATI - CONHEÇA OS NUMERATI - ELES JÁ CONHECEM VOCÊ**. 1. ed. [S.l.]: ARX, 2009.

BASILI, Victor R et al. Protocols in the use of empirical software engineering artifacts. **Empirical Software Engineering**, Springer, v. 12, n. 1, p. 107–119, 2007.

BISHOP, Christopher M et al. **Pattern recognition and machine learning**. [S.l.]: springer New York, 2006.

BRYAN, David A; LOWEKAMP, Bruce B; JENNINGS, Cullen. Sosimple: A serverless, standards-based, p2p sip communication system. In: IEEE. **Advanced Architectures and Algorithms for Internet Delivery and Applications, 2005. AAA-IDEA 2005. First International Workshop on**. [S.l.], 2005. p. 42–49.

CEDERSTRÖM, Andreas. **On using Desktop Grid Computing in software industry**. 2010. Dissertação (Mestrado) — School of Engineering, Blekinge Institute of Technology, Ronneby, Suécia.

CHARNIAK, Eugene. **Introduction to artificial intelligence**. [S.l.]: Pearson Education India, 1985.

CIRNE, Walfredo et al. Labs of the world, unite!!! **Journal of Grid Computing**, Springer, v. 4, n. 3, p. 225–246, 2006.

CLARKE, Richard A; KNAKE, Robert K. **Cyber war**. [S.l.]: HarperCollins, 2011.

COHEN, Bram. **The BitTorrent protocol specification**. [S.l.]: BITTORRENT, 2008.

EMBREY, DE. Sherpa: A systematic human error reduction and prediction approach. In: **Proceedings of the international topical meeting on advances in human factors in nuclear power systems**. [S.l.: s.n.], 1986.

ERICKSON, Jon. **Hacking: The art of exploitation**. [S.l.]: No Starch Press, 2008.

FEDAK, Gilles et al. Xtremweb: A generic global computing system. In: IEEE. **Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on**. [S.l.], 2001. p. 582–587.

FOSTER, Ian; KESSELMAN, Carl. Globus: A metacomputing infrastructure toolkit. **International Journal of High Performance Computing Applications**, SAGE Publications, v. 11, n. 2, p. 115–128, 1997.

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

FREITAS, Henrique Cota de; GÓES, Luís Fabrício Wanderley. Ensino de arquitetura de computadores paralelos: Um estudo de caso com uma competição da intel. 2012.

GOSLING, James; MCGILTON, Henry. The java language environment: A white paper, 1995. **Sun Microsystems**, v. 85, 2003.

HALL, Mark et al. The weka data mining software: an update. **ACM SIGKDD explorations newsletter**, ACM, v. 11, n. 1, p. 10–18, 2009.

HAUGELAND, John. **Artificial intelligence: The very idea**. [S.l.]: MIT press, 1989.

ISO, NBR. 9000: 2000–sistemas de gestão da qualidade–fundamentos e vocabulário. **Rio de Janeiro: ABNT, 26p**, 2000.

KAHANWAL, Dr; SINGH, Dr TP et al. The distributed computing paradigms: P2p, grid, cluster, cloud, and jungle. **arXiv preprint arXiv:1311.3070**, 2013.

KURZWEIL, Ray; SCHNEIDER, Martin L; SCHNEIDER, Martin L. **The age of intelligent machines**. [S.l.]: MIT press Cambridge, 1990.

LEIJOTO, Larissa Fernandes. **Um algoritmo genético para a seleção de características utilizadas na predição de função de proteínas**. 2014 — Monografia (Conclusão do curso) - Pontifícia Universidade Católica de Minas Gerais, Instituto de Ciências Exatas e Informática. Belo Horizonte, Brasil.

MATEESCU, Gabriel; GENTZSCH, Wolfgang; RIBBENS, Calvin J. Hybrid computing—where hpc meets grid and cloud computing. **Future Generation Computer Systems**, Elsevier, v. 27, n. 5, p. 440–453, 2011.

MCDERMOTT, Drew. A temporal logic for reasoning about processes and plans*. **Cognitive science**, Wiley Online Library, v. 6, n. 2, p. 101–155, 1982.

MINIWATTS MARKETING GROUP. **Internet World Stats**. 2013.

NARAYANAN, Ramanathan et al. Minebench: A benchmark suite for data mining workloads. In: IEEE. **Workload Characterization, 2006 IEEE International Symposium on**. [S.l.], 2006. p. 182–188.

NILSSON, Nils J. **Artificial Intelligence: A New Synthesis: A New Synthesis**. [S.l.]: Elsevier, 1998.

OWENS, John D et al. A survey of general-purpose computation on graphics hardware. In: WILEY ONLINE LIBRARY. **Computer graphics forum**. [S.l.], 2007. v. 26, n. 1, p. 80–113.

POOLE, David I; GOEBEL, Randy G; MACKWORTH, Alan K. **Computational Intelligence**. [S.l.]: Oxford University Press Oxford, 1998.

RICH, Elaine; KNIGHT, Kevin. Artificial intelligence. **McGraw-Hill, New**, 1991.

RIMAL, Bhaskar Prasad; CHOI, Eunmi; LUMB, Ian. A taxonomy and survey of cloud computing systems. In: IEEE. **INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on**. [S.l.], 2009. p. 44–51.

ROGER, S Pressman. Software engineering: a practitioner's approach. **McGrow-Hill International Edition**, 2005.

RUSSELL, Stuart; NORVIG, Peter. Artificial intelligence: A modern approach. Prentice Hall, 2009.

SCHOLLMEIER, Rüdiger. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In: IEEE COMPUTER SOCIETY. **Peer-to-Peer Computing, IEEE International Conference on**. [S.l.], 2001. p. 0101–0101.

SCHWIEGELSHOHN, Uwe et al. Perspectives on grid computing. **Future Generation Computer Systems**, Elsevier, v. 26, n. 8, p. 1104–1115, 2010.

SNAVELY, Allan et al. Benchmarks for grid computing: a review of ongoing efforts and future directions. **ACM SIGMETRICS Performance Evaluation Review**, ACM, v. 30, n. 4, p. 27–32, 2003.

TANENBAUM, Andrew S. **Computer networks**. 5. ed. [S.l.]: Prentice Hall PTR, 2011.

THAIN, Douglas; TANNENBAUM, Todd; LIVNY, Miron. Distributed computing in practice: The condor experience. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 17, n. 2-4, p. 323–356, 2005.

TURING, Alan M. Computing machinery and intelligence. **Mind**, JSTOR, p. 433–460, 1950.

VAQUERO, Luis M et al. A break in the clouds: towards a cloud definition. **ACM SIGCOMM Computer Communication Review**, ACM, v. 39, n. 1, p. 50–55, 2008.

WARD, Jonathan Stuart; BARKER, Adam. Undefined by data: A survey of big data definitions. **arXiv preprint arXiv:1309.5821**, 2013.

WIKIPEDIA. **Cognitive Science — Wikipedia, The Free Encyclopedia**. 2014. Disponível em: $<$http://en.wikipedia.org/wiki/Cognitive$_s cience Artificial_i ntelligence >$.$Acessoem 18 nov.2014$.

WITTEN, Ian H; FRANK, Eibe. **Data Mining: Practical machine learning tools and techniques**. [S.l.]: Morgan Kaufmann, 2005.

ZHANG, Qi; CHENG, Lu; BOUTABA, Raouf. Cloud computing: state-of-the-art and research challenges. **Journal of internet services and applications**, Springer, v. 1, n. 1, p. 7–18, 2010.

ZHAO, Han; LIU, Xinxin; LI, Xiaolin. A taxonomy of peer-to-peer desktop grid paradigms. **Cluster Computing**, Springer, v. 14, n. 2, p. 129–144, 2011.