Trabalho de Conclusão de Curso

Rodrigo L. M. Flores Orientador: Roberto Hirata Jr.

29 de novembro de 2009

Sumário

Parte I
Parte objetiva

1 Introdução

O ponto de partida para este projeto é a necessidade de se fazer processamentos custosos de dados em bioinformática. Estes processamentos costumam ser combinatórios, o que os fazem demorar um bom tempo para serem executados em um computador comum, sendo necessário utilizar recursos computacionais de alta performance para obter os resultados em tempo hábil.

1.1 Programação paralela e distribuída

Uma abordagem clássica para se resolver esse problema é utilizar *clusters*, que são um grupo de computadores ligados entre si de modo a parecer ser um único computador muito mais potente. *Clusters* podem ser tanto máquinas específicas para isso, produzidas com um alto custo e com um hardware específico para otimizar seu desempenho, ou pode ser utilizado o processamento em grade: utilizando computadores pessoais que são produzidos em massa a um preço mais baixo e trabalhando em paralelo para fazer o processamento.

O projeto *Beowulf* é um dos exemplos de computação em grade: computadores pessoais baratos constituem uma grade dedicada que funciona como um super computador. De acordo com o projeto Beowulf, uma rede deste tipo provê o mesmo recurso computacional que um super computador mas custando de um décimo a um terço do preço. Outro exemplo de computação em grade são os projetos de computação oportunista ou voluntária, não necessariamente usam computadores de forma dedicada mas utilizam o tempo ocioso do computador.

1.2 Computação oportunista

A computação oportunista é um tipo de computação distribuída no qual pessoas que possuem computadores podem doar processamento e armazenamento ocioso de suas máquinas. Estes projetos normalmente têm um objetivo bem definido com um apelo humanitário.

O primeiro projeto de Computação voluntária foi o *Great Internet Mersenne Prime Search*, lançado em janeiro de 1996. Seu objetivo era encontrar números

primos de Mersenne[1]. Em seguida houveram muitos outros projetos. Entre os projetos mais destacados podemos citar o Folding At Home, que investiga o enrolamento de proteínas e que pode ajudar o desenvolvimentos de pesquisas contra Câncer, doença de Huntington, entre outras. Outro dos projeto notável de computação voluntária, e de grande importância para este trabalho, é o *SETI@Home* que atraiu centenas de milhares de voluntários de vida inteligente fora da Terra. Como um produto deste projeto, nasceu o *middleware BOINC* que hoje é utilizado em diversos projetos.

1.3 Linguagens Interpretadas

Uma das possíveis divisões para linguagens de programação é se seus códigos são compilados para código de máquina ou se são simplismente interpretados. Enquanto no primeiro caso há a figura de um compilador, que transforma o código fonte em código de máquina para ele poder então ser executado, no segundo caso, há a figura de um interpretador que não converte o programa para código de máquina, mas sim o interpreta e o executa.

Linguagens interpretadas possuem vantagens e desvantagens sobre as compiladas: embora elas sejam mais fáceis de serem multiplataforma (basta o interpretador estar disponível para aquela plataforma) e permitam escopo e tipagem dinâmica, também costumam ser menos eficientes que linguagens compiladas e a presença de um interpretador é obrigatória para sua execução.

Dentre as linguagens interpretadas, uma que adquiriu destaque na área de estatística e bioinformática é a linguagem *R*, que será de fundamental importância para este trabalho.

1.3.1 Linguagem R

A linguagem *R* é uma linguagem interpretada bastante utilizada no desenvolvimento de rotinas de bioinformática e na análise de dados. Muitas funções normalmente utilizadas em análises de dados vêm incluídas na linguagem como por exemplo cálculo da média, desvio padrão, ajuste de curva e etc. A geração de gráficos do *R* também é bastante elaborada e é possível gerar gráficos dos mais diversos tipos. Também há implementações de algoritmos mais elaborados de estatística como por exemplo algoritmos de *clustering* que também são bastante usados em bioinformática.

Outro ponto interessante da linguagem *R* são os pacotes extras que se pode baixar. Ao todo, no repositório oficial são 2076 pacotes, para os mais diversos propósitos que podem ser desde funções mais específicas para análise como

[1]Isto é, números primos na forma $M_n = 2^n - 1$

por exemplo o *Bayesclust*[2], para geração de gráficos utilizando outras bibliotecas como por exemplo o *rgl*[3].

Com maior destaque para área de bioinformática, o artigo [GCB+04] fala sobre o *Bioconductor*, um projeto que propõe o fornecimento de ferramentas para a análise de dados desta área. O projeto foi iniciado em 2001 e na grande maioria, seus pacotes são para o ambiente de programação da linguagem *R*.

1.4 Linguagens interpretadas e computação distribuída

Embora já existam ótimas soluções para a execução distribuída de programas compilados como o MPI[4], não se fala muito em soluções para execução distribuída de programas em linguagem interpretada. Para a linguagem *R* há um pacote chamado *gridR* que permite o uso do R com o *Condor*, um middleware para execução de programas em grades. Um outro trabalho que relaciona o R com computação distribuída é o [RAD09] que utiliza o *Alchemi*, um *middleware* para processamento em grade e baseado na tecnologia *.NET*, e a interface *COM* junto com o pacote do *R RCom*.

O artigo [GGdVS08] fala sobre a utilização do Middleware de computação voluntária *BOINC* como solução para computação em grade na Universidade de Extremadura na Espanha e dentre os programas executados na grade, haviam programas em R. Porém isso foi somente instalado em redes de computadores cujo sistema operacional é o Linux.

1.5 Solução multiplataforma

Embora o ambiente *Linux* seja muito utilizado no ambiente acadêmico e em ambientes de desenvolvimento de softwares, é muito difícil encontrá-lo em um ambiente doméstico ou de trabalho cotidiano. Embora a cada ano o número de usuários deste sistema cresça a cada ano, percebemos que o sistema normalmente utilizado mesmo é o Microsoft Windows. O artigo [RAD09] dispõe uma solução usando o *middleware Alchemi*, baseado na tecnologia da Microsoft *.NET* que é executado no sistema operacional *Microsoft Windows*. O artigo [GGdVS08] porém utiliza uma rede com computadores rodando *Linux*

Porém como temos muitos computadores com sistemas *Linux* e *Windows* na rede *CEC* do IME-USP, pensamos que o ideal seria termos uma solução que pudesse utilizar os dois sistemas. Como tanto o *BOINC* como o *R* estão disponíveis

[2]Disponível em `http://cran.r-project.org/web/packages/bayesclust/index.html`

[3]Disponível em `http://cran.r-project.org/web/packages/rggobi/index.html`

[4]disponível em `http://www.mcs.anl.gov/research/projects/mpi/`

para os dois sistemas, um dos objetivos deste trabalho é utilizar os dois sistemas na grade.

2 Conceitos e tecnologias utilizadas

O desenvolvimento do projeto incluiu diversas tecnologias, sendo as principais a linguagem de Programação *R* e o middleware para computação voluntária *Boinc*. Dentre os conceitos estudados, podemos destacar a computação em grade.

2.1 BOINC

O BOINC, cujo nome é uma sigla para *Berkeley Open Infrastructure for Network Computing*, é um middleware para computação em grade e voluntária e foi criado na Universidade de Berkeley, Califórnia, Estados Unidos.

Inicialmente, o projeto consistia em gerenciar o projeto *SETI@HOME* que possuía dois objetivos:

- Provar a viabilidade e a praticidade do conceito "computação em grade distribuída";
- Fazer um trabalho científico útil fazendo uma análise observacional para detectar vida inteligente fora da Terra.

O primeiro objetivo foi concluído com sucesso e o resultado é o *BOINC*. O segundo falhou: nenhuma evidência de vida inteligente fora da Terra foi encontrada.

Dentre os diversos motivos para a utilização do *BOINC*, baseados no artigo [GGdVS08], podemos destacar:

- **Mais utilizado -** Quando comparado com outros *middlewares* semelhantes como o *middleware Condor* ou o *Xtremweb*, o *BOINC* é mais utilizado e há pacotes para o *BOINC* nas distribuições *Linux* mais populares;
- **Amplamente utilizado -** O *BOINC* é utilizado em diversas áreas como previsão do tempo, física astrofísica, biologia, entre outras;
- **Suporte da comunidade -** Baseado no espírito de ajuda mútua existente em comunidades de software livre, é possível ter dúvidas esclarecidas quanto ao funcionamento do *BOINC* de maneira fácil e desburocratizada. Por ser um projeto cujas listas de discussão e canais de *chat* são movimentados é bem comum alguns problemas serem resolvidos em questão de poucos dias;
- **Estrutura simples -** O BOINC possui uma estrutura simples de comunicação: um servidor que armazena e distribui os trabalhos a serem feitos e os clientes que tem o papel de processar os trabalhos;

2.1.1 Funcionamento do BOINC

Cada unidade de processamento no Boinc é chamada de *workunit* e é constituida de arquivos executáveis e arquivos de entrada. Depois de processado, os arquivos de saída gerados são enviados para o servidor que normalmente armazena estes arquivos em um banco de dados ou em um arquivo.

Para gerar um workunit são necessários dois arquivos XML, um deles detalhando a entrada e o outro detalhando a saída. Para facilitar a escrita do programa, precisamos escrever para cada arquivo um nome lógico que ao enviar e receber o cliente renomeia o arquivo. Por exemplo, temos um programa que lê um arquivo chamado `input` e escreve no arquivo `output`, para podermos ter muitos arquivos de entrada com nomes diferentes, quando criamos uma *workunit*, o servidor coloca um nome único e semelhante ao da workunit nos arquivos de entrada e saída que serão renomeados pelo cliente para o nome lógico.

O processamento é realizado pelo cliente: o arquivo binário é executado e enquanto ele é executado há um checkpoint que permite em caso de interrupções retomar o processamento de um determinado ponto. Finalizado o processamento, na próxima atualização o cliente avisará ao servidor que o processamento foi finalizado. Um diagrama do funcionamento pode ser visto na figura 1.

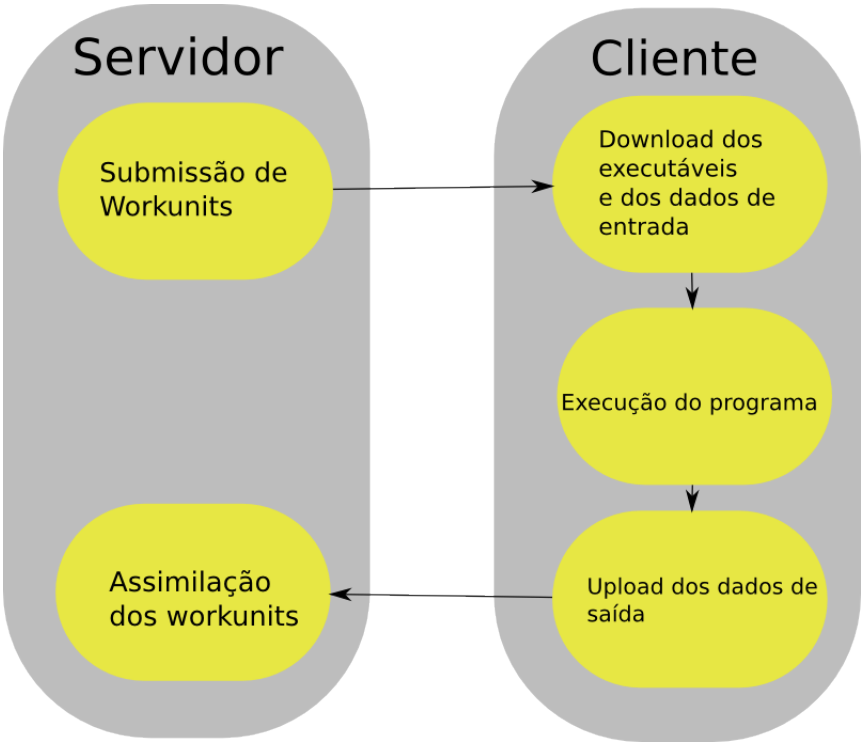

Figura 1: Funcionamento do Boinc

2.1.2 Wrapper

O *Wrapper* é um programa escrito utilizando a *api* do *BOINC*, cujo objetivo é executar aplicações legadas, i.e. aplicações que não utilizam a API do *BOINC*, utilizando o *BOINC*. Há uma versão do Wrapper distribuída junto com o *BOINC* que utiliza um arquivo XML, mas existe uma outra opção descrita no artigo [MBK09] que utiliza um shell para a execução dos aplicativos.

O arquivo XML de execução tem a seguinte estrutura:

```
<job_desc>
    <task>
        <application>foobar</application>
        [ <stdin_filename>stdin_file</stdin_filename> ]
        [ <stdout_filename>stdout_file</stdout_filename> ]
        [ <stderr_filename>stderr_file</stderr_filename> ]
        [ <command_line>--foo bar</command_line> ]
    </task>
    [ ... ]
</job_desc>
```

Neste XML, o único campo obrigatório é o *application*, que é a aplicação que será executada e pode ser distribuída junto com a aplicação ou já existir no computador que o cliente estará instalado (para este segundo caso é necessário informar o caminho inteiro do executável). É possível ter mais de uma tag task, e o wrapper as executará sequencialmente. É de responsabilidade do *Wrapper* perceber se a execução do programa foi feita com sucesso, mas não é responsabilidade dele verificar se os pré-requisitos que o programa necessida estão presentes no sistema.

2.2 R

A linguagem de programação estatística *R* é uma implementação da linguagem *S* e foi criada por Ross Ihaka e Robert Gentleman na Universidade de Auckland, da Nova Zelândia. A linguagem e ambiente para cálculos estatísticos é considerada padrão na área de análise de dados e além do ambiente acadêmico, empresas como *Google*, *Pfizer* e *Merck* a utilizam em seus processos de mineração de dados.

Dentre as facilidades que seu uso proporciona, podemos citar as seguintes:

- Grande quantidade de funções estatísticas de uso cotidiano na análise de dados como cálculo de média, desvio padrão, ajuste de curvas, análise de séries temporais. Há também funções mais complexas como por exemplo implementações de algoritmos de *clustering*.

- Utilizando o *R* é possível fazer operações em tabelas semelhantes a que normalmente são feitas em bancos de dados relacionais como seleção de linhas em tabelas que atendem certas condições.

- O *R* possui também rotinas de manipulação de matrizes e resolução de sistemas lineares, que são essenciais em qualquer tarefa de cálculo numérico.

- Gráficos de alta qualidade dos mais variados tipos e para os mais variados propósitos com uma qualidade alta também podem ser feitos com o *R*.

- A facilidade de se criar e de se obter pacotes, que nada mais são que conjuntos de rotinas agrupadas, é outro fator importante: é muito simples criar pacotes e documentá-los. Para disponibilizá-los, pode se submeter um pedido de aprovação de para disponibilização no repositório oficial (`http://cran-r.c3sl.ufpr.br/`) do *R*, que possui 2076 pacotes para os mais diversos fins. O repositório possui *mirrors* espalhados por todo o mundo, inclusive no Brasil.

Além destes motivos, o artigo [GCB+04] dispõe outros motivos para a utilização do *R* e de seu pacote para análise de dados de de bioinformática *Bioconductor*.

- **Transparência -** Muitas metodologias de análise na área de biologia e bioinformática computacional são extremamente complexas e muitas etapas são necessárias na conversão de informação bruta (como por exemplo imagens escaneadas de *microarray*). Não se sabe a priori, como as análises podem ser sensíveis a estes fatores, e portanto trabalhos referenciados na área normalmente expõe todo o processo. O uso de mesmo ambiente e das mesmas facilita bastante esta transparência;

- **Reprodutibilidade -** Experimentos na área de biologia molecular devem publicar listas de ingredientes e algoritmos para criar substâncias e processos. Um resultado só pode ser verificado se existir uma obediência a um protocolo. Seguindo esta linha, a mineração de dados também deve ser bem descrita e tanto o código fonte como os dados nos quais a análise é baseada são normalmente publicados junto com um trabalho nesta área. Utilizar um mesmo ambiente, que pode ser obtido gratuitamente, para as qualquer plataforma, com pacotes facilmente extensíveis também facilita este processo.

- **Eficiência do desenvolvimento -** Se pacote foi por ventura desenvolvido para alguma necessidade, ele pode ser publicado , melhorado, estudado por outros cientistas, e pode ter seu leque de funcionalidades aumentado caso seu uso siga padrões de código aberto. Para isso é necessário que esteja bem documentado. Tanto o *R* como o *Bioconductor* são softwares de código aberto e disfrutam desta qualidade.

- **Prototipagem -** Como o *R* é uma linguagem em um nível mais alto que outras linguagens como *C* ou *FORTRAN*, programar novas rotinas é bem mais simples. Mesmo não tendo uma execução tão eficiente como em outras linguagens, pode ser utilizado como protótipo, para depois ser implementado em uma linguagem mais eficiente caso o resultado seja bom.

3 Atividades realizadas

O início do projeto deu-se ainda em 2008, com a visita ao colégio Rainha da Paz na Lapa, onde o aluno de mestrado do *IME* Rodrigo Assirati Dias mantém uma grade de computadores com o middleware *Alchemi* citada no trabalho [RAD09] . Nesta visita foi possível esclarecer dúvidas, entender o funcionamento da grade e receber algumas dicas quanto à manutenção da grade. Após esse encontro, começou-se a buscar alternativas para o a computação de alta performance com o *R*. Um primeiro pacote encontrado que fazia esta função foi o *GridR*, que permite submeter rotinas do *R* para *clusters*, máquinas remotas e grades. Um dos arcabouços possíveis para o uso deste pacote é o Condor[5] , desenvolvido pela *University of Wisconsin-Madison* e é bastante utilizado em empresas de grande porte como a *NASA* e pode ser executado tanto em sistemas baseados em *UNIX*, como em sistemas *Windows*. Seguindo esta busca, encontramos o *middleware BOINC*, no artigo [GGdVS08] sendo utilizado para um propósito semelhante em um trabalho na Universidade de Extremadura, na Espanha e decidimos que a abordagem seria interessante para nosso trabalho.

Escolhido o *middleware* nos focamos na instalação do servidor. A própria página do *BOINC* possui um guia de instalação do servidor do *middleware* no sistema `Debian GNU\Linux` e por esta distribuição *Linux* ser bastante conhecida por sua estabilidade, foi instalado este sistema no servidor. Instalado o servidor, o foco foi em ter uma aplicação em R executando remotamente em uma grade de computadores. Este processo no sistema Linux foi relativamente simples: utilizando o "truque do *shebang*" é possível colocar um script como executável e o *wrapper* executá-lo como se fosse um arquivo compilado. Já para o sistema Windows o trabalho foi mais complicado: havia um bug nas configurações de compilação do *wrapper* e até perceber isso atrasou bastante o andamento do projeto. Passado isso, foi necessário utilizar um programa escrito em C, que apenas executava o interpretador junto com o arquivo com a rotina em *R*.

Finalizado esta parte, nos focamos na aplicação a ser executada na grade. Para isso foi criado um programa em *R* que apenas acessava um arquivo e fazia alguns cálculos custosos. Isto foi então configurado para o mesmo programa poder ser executado tanto em sistemas Windows como em sistemas Linux. Paralelamente a isso, foi pedido para a administração da rede do *CEC* para instalar o *BOINC* nas máquinas da rede. Como a rede estava em reforma, para a troca do sistema operacional não foi possível concluir a instalação até o fim deste trabalho, mas acredito que em breve teremos a grade em pleno funcionamento.

[5]Disponível em `http://www.cs.wisc.edu/condor/`

4 Resultados e produtos obtidos

O principal resultado deste trabalho foi fazer o *Boinc* funcionar com rotinas em *R*, tanto nos ambientes *Linux* como no ambiente *Windows*

4.1 Sistema Linux

Para o ambiente *Linux*, foi relativamente simples o processo: dado um arquivo com rotinas do *R* a serem executadas, é somente necessário alterar a permissão do arquivo para executável (via `chmod +x arquivo.R`) e adicionar a seguinte linha no início do arquivo:

```
#!/usr/bin/Rscript
```

Isso faz um sistema *Linux* chamar o interpretador *Rscript* para interpretar o arquivo e assim fazer a interpretação do arquivo. Esta solução permite não só que rotinas em *R* sejam executadas, mas sim qualquer script que tenha seu interpretador descrito na primeira linha.

A esta solução, demos o nome de *Truque do Shebang*, pelos caracteres `#!` serem chamados popularmente de *shebang*. Porém, para termos a mesma solução em ambos os sistemas, utilizamos a solução para Windows no sistema Linux.

4.2 Sistema Windows

Como o sistema Windows não permite utilizar scripts utilizando o *shebang*, foi necessário utilizar um programa compilado escrito na linguagem C, que chamamos de *Runner*, que usando a função `system`, chama o interpretador com o arquivo.R como parâmetro.

Esta solução permite inclusive que usemos scripts diferentes de R a cada vez que criamos um *workunit*, assim como adicionar arquivos extra que por ventura fossem necessários para o processamento. Esta maneira também funciona no *Linux*, só que o *Runner* precisar ser compilado para o *Linux* com o caminho para o interpretador correto. O programa também não faz uma verificação para perceber se o *R* está instalado.

A utilização do *Runner* foi necessário devido ao *Wrapper* não perceber corretamente que o interpretador foi executado sem erros e mesmo em execuções sem erros, o *wrapper* recebia um valor de retorno do interpretador diferente de zero, o que ele percebia como um erro e marcava o *workunit* como inválido.

4.3 Discussão

4.3.1 Vantagens

As principais vantagens no uso do *Boinc* para o processamento em grade são:

- **Facilidade de se adicionar novos nós** - A instalação do BOINC em ambos os sistemas Linux e Windows é simples e fácil de ser feita e nenhuma ação no servidor é necessária a cada instalação de nós. Além disso, é muito simples fazer a replicação de configurações, tanto para o processamento, quanto para a conexão com o servidor para os computadores;

- **Processamento multiplataforma** - Para a grade funcionar na plataforma só são necessárias duas coisas: que o BOINC e o *R* estejam disponível para a plataforma. As plataformas mais comuns (Linux 32 e 64 bits e Microsoft Windows) têm ambos os projetos disponíveis;

- **Código aberto** - A utilização de dois softwares com código aberto facilita bastante: a busca de bugs se torna possível, a gratuidade dos softwares e a grande quantidade de documentação, muitas vezes produzidas por pessoas que não necessariamente são da equipe de desenvolvimento do *BOINC*. A mentalidade de ajuda mútua, existente nas listas de discussão e no canal de IRC do projeto também é de grande ajuda;

- **Execução invisível ao usuário** - Com o *BOINC* configurado para isso, um usuário comum da rede nem ao menos percebe a existência de um processamento em grade. É possível configurar o *BOINC* para só começar a execução com o computador ocioso após um número arbitrário de minutos. Também é possível configurar para o processamento só acontecer em determinadas faixas de horários e dias da semana. Outra configuração interessante é a determinação do máximo de memória *RAM* e de espaço em disco para a execução, assim como a frequência com que ele usará a rede.

- **Solução funciona para qualquer linguagem de script** - De maneira análoga, é possível executar qualquer programa escrito em linguagem interpretada com o BOINC utilizando esta mesma solução. Como comentado antes, só é necessário que exista uma versão do interpretador para as plataformas necessárias (o que é comum para as linguagens mais utilizadas como *PERL*, *Python* e *Ruby* e as plataformas mais comuns).

4.3.2 Desvantagens

As principais desvantagens são:

- **Necessidade de se ter o *R* instalado** - O *R* não é uma linguagem instalada por padrão nas distribuições Linux mais populares, nem no *Windows*. Então, a adição de um nó só pode ser feita se o *R* for também instalado.

- **Falta de checkpoints** - Utilizando um aplicativo feito com a *API* do *BOINC* é possível se ter *Checkpoints*, que são uma maneira de uma aplicação feita com o *BOINC* reiniciar o processamento não do início, mas sim de um determinado ponto. Utilizando o *Wrapper* e o *R*, perdemos esse recurso. A computação de rotinas longas se torna mais difícil e pouco recomendada, já que a cada interrupção o processamento é reiniciado.

- **Falta de "compromisso" fixo dos clientes** - Diferentemente da rede citada no artigo [RAD09] não há a figura de um computador *Manager* que gerencia as máquinas, atualizando a qualquer momento, mas sim um servidor que apenas envia e recebe as tarefas e a iniciativa de computação fica com os computadores da grade.

4.4 Instalação da rede

A instalação está em andamento na rede do laboratório CEC do IME-USP. No momento possuímos 3 máquinas Linux e 2 máquinas Windows com o *Boinc* em funcionamento . Por motivos de reinstalação do sistema dos computadores, a instalação está tomando mais tempo que o previsto, mas em breve acredito que teremos a grade em seu pleno funcionamento.

5 Conclusões

A principal conclusão deste trabalho é que é possível a utilização do *BOINC* para o processamento de rotinas de bioinformática escritas na linguagem *R*. A utilização multiplataforma também foi de grande utilidade já que podemos incluir praticamente qualquer tipo de computador existente em redes. O custo total de implementação foi somente a inutilização de uma máquina, utilizada como servidor da rede. Embora o *BOINC* seja um projeto consolidado e possua uma documentação com bastante conteúdo, as mensagens de erro, principalmente do *Wrapper*, são muito pouco informativas, o que tornou o desenvolvimento deste projeto bastante trabalhoso.

Para a continuação do projeto há várias sugestões:

- *Benchmark* da rede e comparação com grade descrita no artigo [RAD09] Para determinar a viabilidade, seria interessante estabelecermos a comparação com outra alternativa.
- Comparação com grades "alugadas": hoje já existe oportunidade de se fazer esse tipo de processamento. em grades alugadas como a oferecida pela empresa *amazon*. Como os computadores na rede consomem energia elétrica seria interessante comparar o gasto da energia elétrica com o gasto em uma grade "alugada".
- Analisar o desempenho nas máquinas com Windows e com Linux. Seria interessante analisar o benchmark da grade em ambos os sistemas e determinar qual das duas plataformas é mais propícia para o processamento;
- Utilização de máquinas virtuais: como feito no artigo [GGdVS08], podemos utilizar máquinas virtuais, que são iniciadas em cada nó e é feito o processamento. Sem ter que se preocupar com a instalação do *R* em todas as máquinas

Referências

[boi] *Página de documentação do wrapper do boinc*, http://boinc.berkeley.edu/trac/wiki/WrapperApp.

[GCB+04] Robert C. Gentleman, Vincent J. Carey, Douglas M. Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus, Rafael Irizarry, Friedrich Leisch, Cheng Li, Martin Maechler, Anthony J. Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney, Jean Y. Yang, and Jianhua Zhang, *Bioconductor: open software development for computational biology and bioinformatics.*, Genome biology **5** (2004), no. 10, R80+.

[GGdVS08] D.L. Gonzalez, G.G. Gil, F.F. de Vega, and B. Segal, *Centralized boinc resources manager for institutional networks*, April 2008, pp. 1–8.

[MBK09] Attila Csaba Marosi, Zoltan Balaton, and Peter Kacsuk, *Genwrapper: A generic wrapper for running legacy applications on desktop grids*, Parallel and Distributed Processing Symposium, International **0** (2009), 1–6.

[RAD09] Roberto Hirata Jr. Rodrigo A. Dias, *Middle-r - a user level middleware for statistical computing*, VII Workshop on Grid Computing and Applications, Recife - Brazil (2009).

[Wik09a] Wikipedia, *Boinc — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Berkeley_Open_Infrastructure_for_Network_Computing&oldid=326896451, 2009, [Online; accessed 23-November-2009].

[Wik09b] _____, *Interpreted language — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Interpreted_language&oldid=323979791, 2009, [Online; accessed 23-November-2009].

[Wik09c] _____, *R (programming language) — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=R_(programming_language)&oldid=326801637, 2009, [Online; accessed 23-November-2009].

[Wik09d] _____, *Volunteer computing — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Volunteer_computing&oldid=325674089, 2009, [Online; accessed 23-November-2009].

Parte II
Parte subjetiva

6 Desafios e frustrações encontrados

O curso de bacharelado em ciência da computação é um curso bastante denso e dificilmente temos tempo para fazer todas as tarefas de todas as disciplinas. Então acredito que o meu maior desafio nestes anos de IME foi conciliar todas as tarefas e disciplinas, e infelizmente descartando alguma as vezes.

A falta de aprendizado de orientação a objeto também foi uma frustração: como este assunto é muito requisitado por empresas e projetos, não aprendê-lo foi bastante frustrante. Há também pouco enfoque a tecnologias, deveriam haver mais chances de aprendermos tecnologias e desenvolvermos mais projetos.

A figura de um orientador e de um tutor também foi de suma importância: é sempre interessante ter alguém que se possa receber conselhos de matérias e de optativas. Desenvolver também um projeto de iniciação científica também foi importante, para meu desenvolvimento e acredito que para qualquer aluno seria uma experiência válida.

7 Disciplinas relevantes para o trabalho

Diversas disciplinas foram relevantes para este trabalho:

- **MAC122 - Principio e desenvolvimento de algoritmos** - Este curso fornece uma base importante para as outras matérias e ajuda a melhorar o raciocínio para elaborarmos algoritmos.
- **MAC211 - Laboratório de programação I** - Ferramentas como versionamento de código, processamento de texto e o makefile foram essenciais para a elaboração deste trabalho de forma indireta e contribuiram com a boa qualidade do mesmo.
- **MAC242 - Laboratório de programação II** - Linguagens de script facilitam bastante o trabalho de tarefas repetitivas e o boa parte do que sei sobre este tipo de linguagem eu aprendi neste curso.
- **MAC338 - Análise de algoritmos** - Este curso contribuiu indiretamente com minha formação como cientista da computação i e muitos dos conceitos aprendidos neste curso ajudaram o entendimento melhor de algoritmos e soluções.

- **MAC431 - Programação paralela e distribuída** Aprender a utilidade de programação de alta performance foi de bastante utilidade neste trabalho.
- **MAC422 - Sistemas Operacionais** - Saber como um programa é executado, como o sistema gerencia essas execução e como a tabela de processos funciona é essencial quando se trabalha com uma grade de computadores.
- **IBI5031 - Reconhecimento de padrões** - Neste curso tive a oportunidade de aprender algoritmos e fazer análises de dados, que são normalmente feitas em pesquisas de bioinformática. Também programei bastante na linguagem *R* neste curso e percebi sua importância

8 Conceitos Aplicados

Dentre os conceitos aprendidos no curso utilizados no trabalho pode-se citar a computação distribuída, aprendida no curso de computação paralela e distribuída. Saber que alguns problemas só podem ser resolvidos de uma exaustiva, também foi importante e esse foi um conceito aprendido no curso de Análise de Algoritmos.

9 Continuação do trabalho

O trabalho possui uma continuação óbvia: terminar a instalação em todos os nós da rede e fazer uma análise de *Benchmarking da rede* para testar a viabilidade quando a instalação estiver completa, como resultado disso há a possibilidade de escrevermos um artigo falando sobre a grade para ser publicado na página do *BOINC*. Outro trabalho interessante decorrente deste seria melhorar o *Wrapper* e o *BOINC* em geral para fazê-los terem mensagens de erros mais claras.

10 Agradecimentos

Este trabalho foi feito com a ajuda de diversas pessoas. Indiretamente, meus pais, familiares, amigos e namorada foram de extrema importância no apoio, amizade e carinho.

Diretamente, agradeço à CNPq pela bolsa que me permitiu a dedicação exclusiva a este projeto. Agradeço também ao meu orientador, Prof. Dr. Roberto Hirata Jr. que me apoiou e me ajudou bastante na elaboração deste trabalho. Para a obtenção do servidor, agradeço ao Prof. Dr. Roberto Marcondes César Junior, e aos administradores da Rede Vision, David Pires e Rodrigo Bernardo Pimentel,

que me ajudaram quando na substituição do servidor com um problema de *hardware*. Agradeço também ao aluno de pós-graduação Rodrigo Assirati Dias, que me ajudou e me deu diversas dicas com relação à manutenção e a instalação da grade. Na parte do *BOINC* agradeço à Nicolás Alvarez e Yoyo, que me ajudaram a resolver os bugs e a implementar a solução descrita pelo trabalho.