Provisão de computação intensiva de dados para suporte a aplicações científicas*

Felipe Gutierrez[1], **Marcos Barreto**[1]

[1]LaSiD, IM, DCC, UFBA
Av. Adhemar de Barros, s/n - Campus Ondina
40170-110 - Salvador, BA - Brasil

felipe.o.gutierrez@gmail.com, marcoseb@dcc.ufba.br

Resumo. *Existem modelos baseados em computação distribuída na computação em nuvem para cumprir a demanda de processamento para vários tipos de aplicações. Eles têm a mesma característica de processamento elástico presentes na computação intensiva de dados. Esta se baseia na grande quantidade de dados a serem processados. Neste artigo, apresentamos a integração do middleware Hadoop no nível de infra-estrutura de uma plataforma de apoio para aplicações científicas. Dois estudos de caso, sobre pesquisa de texto e uma aplicação científica, são discutidos a fim de mostrar como o Hadoop pode se relacionar com esta plataforma.*

Palavras-chave – computação intensiva de dados, Big Data, MapReduce, aplicações científicas

1. Introdução

A necessidade de baixar o custo de processamento e de disponibilizar uma plataforma de simples acesso a um usuário que não tem conhecimentos avançados em computação foram os principais problemas abordados com o projeto MC^2. A sigla MC^2 do projeto é um acrônimo para "Minha Cloud Científica"[1]. A camada de infraestrutura da plataforma fornece recursos computacionais de diversos paradigmas, tais como clusters, redes voluntárias e redes par-a-par. O *middleware* de computação distribuída utilizado fornece a facilidade de implantação das aplicaçãoes científicas. Com a ajuda de uma interface Web, o usuário (cientista) não tem mais a necessidade de conhecer as ferramentas de computação distribuída. Ele pode criar uma única tarefa ou um fluxo de tarefas dependentes para seu sistema implantado na plataforma MC^2. Neste trabalho foram estudados os sistemas de computação intensiva de dados e a integração de dois destes sistemas ao projeto MC^2.

2. O projeto MC^2

O projeto MC^2 tem como objetivo ser uma plataforma de computação na nuvem para oferecer apoio às aplicações científicas, com as características de flexibilidade e escalabilidade. A plataforma provê serviços de acesso a grande porte computacional por curtos intervalos de tempo, armazenamento, reprodutibilidade de experimentos e controle de proveniência de dados. Estes serviços são disponibilizados através de módulos que interagem com os recursos de execução presentes na camada de infraestrutura (clusters, BOINC [Anderson 2004], OurGrid [Nazareno Andrade 2003]). O foco discutido neste trabalho está na inclusão do ambiente Hadoop [K. Shvachko 2010] [White 2009] ao projeto MC^2.

O projeto segue o paradigma *PaaS* (*Plataform as a Service*), o que permite o fácil desenvolvimento, implantação e execução de serviços personalizados. Existem dois tipos de perfis de usuários que utilizarão o sistema, o cientista e o desenvolvedor. O usuário cientista

*Trabalho financiado pela RNP, edital Grupos de Trabalho 2012 - 2013

[1]http://gtmcc.lncc.br/sinapad/projectmanager/public/projects/gt-mcc

tem permissão para utilizar o ambiente de *SaaS* (*Software as a Service*), que visa atender às necessidades específicas de um usuário ou de um grupo de usuários com demanda similares. O usuário desenvolvedor tem um perfil mais voltado para o desenvolvimento e implantação de sistemas distribuídos. Ele estará habilitado para configurar os níveis de *PaaS* e *IaaS* (*Infrastructure as a Service*).

A arquitetura da plataforma MC^2 está dividida em três camadas e é apresentada na figura 1. A camada *IaaS* é responsável por gerenciar os recursos computacionais distribuídos disponíveis, incluindo funcionalidades de monitoramento, escalonamento de tarefas e tolerância a falhas. A camada *PaaS* é responsável por disponibilizar ferramentas Web acessíveis pelos desenvolvedores, para que eles possam configurar portais de aplicações científicas personalizadas, construídas a partir de um conjunto de ferramentas computacionais pré-definidas e acessíveis ou configuráveis por cientistas. A camada *SaaS* é responsável por disponibilizar portais Web acessíveis pelos cientistas, permitindo-os executar aplicações científicas configuradas pelos desenvolvedores e ainda compartilhar os resultados e detalhes destas aplicações entre eles.

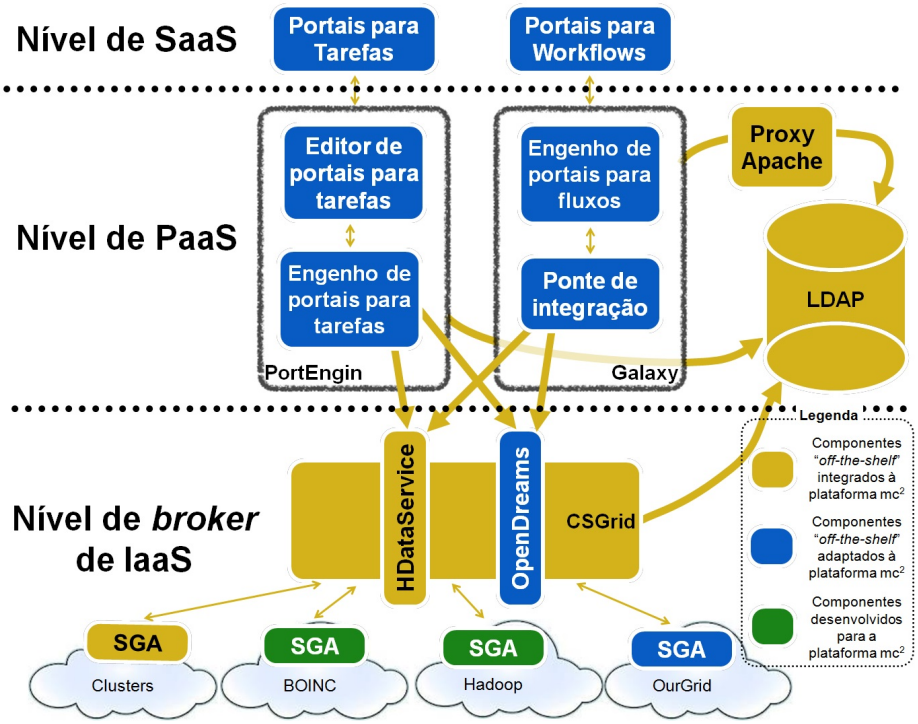

Figura 1. Arquitetura MC^2

O principal componente do nível de infraestrutura no projeto MC^2 é o CSGrid [Lima et al. 2005]. Ele é um *framework* de computação em grade que faz a interface entre o portal Web e o Hadoop, que por sua vez, provê computação intensiva de dados. Com os diversos *frameworks* disponibilizados no nível de infraestrutura, o projeto MC^2 pode atingir certa escalabilidade na medida que são acoplados mais computadores com *hardware* de baixo custo. Essa característica também oferece flexibilidade para o processamento de várias tarefas, através do Galaxy [Goecks et al. 2010], e grande quantidade de dados.

3. A integração de computação intensiva de dados

O SGA é um módulo do CSGrid e sua sigla é uma abreviação para *Servidor de Gerência de Algoritmos*. Ele é um gerenciador instalado em cada máquina de execução de algoritmos que promove a monitoração do estado de ocupação destas máquinas e dos processos executados a

partir dele. A interação do CSGrid e o SGA é implementada em Lua[2] e C++. O SGA Hadoop foi desenvolvido para a administração do *framework* de computação intensiva Hadoop. Sua função é enviar os parâmetros necessários para o Hadoop realizar uma operação com grande quantidade de dados.

O SGA para o Hadoop precisa de um arquivo de configuração para que a máquina que o hospeda seja reconhecida na rede como um nó do CSGrid. Com esta configuração é possível prover flexibilidade para o projeto MC^2, que oferece suporte à computação intensiva de dados. O arquivo *sgad-cnf.lua* é apresentado no Script 1. Neste caso o CSGrid também é responsável pela transferência de arquivos para serem processados no Hadoop. A transferência de arquivos é através do protocolo CSFS[3], que fornece suporte para compartilhamento de arquivos em máquinas de redes diferentes.

Script 1 Script de configuração do SGA para o Hadoop

```
Node{
  name = "sga-ufba-cloud6",
  platform_id = "Linux26g4_64",
  num_processors = 2,
  memory_ram_info_mb = 2048,
  memory_swap_info_mb = 5376,
  csfs = {
    launch_daemon = YES,
    properties = { HOST="192.168.188.6",
    ROOT_DIR="/home/cloud6/sga_v1.6_u020/sgad/csfs/root/",
    USER = "cloud6", PORT = 10000 }
  }
}
```

3.1. Implementações com computação intensiva de dados

O primeiro estudo de caso em computação intensiva de dados realizado no Hadoop através do CSGrid foi de uma aplicação *grep*, já conhecida no sistema operacional Linux. A função *grep* no Linux tem o objetivo de procurar strings em textos. Esta função existe nos códigos de exemplo do Hadoop, mas foi implementada com algumas alterações para a impressão da linha onde a *string* foi encontrada.

Porém, para criar esta aplicação para ser implantada no Hadoop, foi necessário seguir o paradigma de computação distribuída conhecido como *MapReduce* [Lin and Dyer 2010]. O pseudoalgoritmo que detalha o programa *Map* e o programa *Reduce* foi omitido por questões de espaço, mas sua lógica de execução segue o seguinte fluxo. A função *Map* recebe um documento com um ID. Cada linha do documento é extraída de um laço iterativo e a cada conjunto de 10 linhas é criada uma nova chave *key*. Esta chave é passada como parâmetro na função *Emit* da biblioteca *Map*. A função *Shuffle*, que é interna do Hadoop, fica responsável por agrupar todas os objetos com chaves iguais. A função *Reduce* irá receber um conjunto de linhas com a mesma chave *key*. O método *getStringToGrep()* retorna a *string* que será procurada nos arquivos. Se a *string* for encontrada na linha processada, a execução irá entrar na condição e a função *Emit* da biblioteca *Reduce* irá escrever a linha no arquivo de saída.

Este programa foi implementado usando a linguagem de programação Java juntamente com as bibliotecas do Hadoop. Depois de criadas as classes *Map* e *Reduce* é necessário criar

[2] http://www.lua.org/

[3] https://jira.tecgraf.puc-rio.br/confluence/pages/viewpage.action?pageId=30574804

uma classe principal que é responsável por iniciar o programa e fazer referência as bibliotecas *MapReduce*. Após compiladas, é necessário criar um pacote *jar* que será a biblioteca a ser chamada pelo framework Hadoop. A interface SGA-Hadoop foi criada para captar os parâmetros utilizados no programa *grep*, o arquivo de entrada, a *string* que será procurada e o arquivo de saída. Ela é apresentada na figura 2.

Figura 2. Interface do SGA-Hadoop no CSGrid

O segundo caso de estudo foi feito com o CloudBurst [Schatz 2009]. Ele é um algoritmo de leitura e mapeamento projetado e otimizado para o mapeamento de dados da sequência do genoma, sendo utilizado em diversas análises, como descobertas SNP (*Single Nucleotide Polymorphisms*) e genotipagem. Baseia-se em um programa de mapeamento de leitura chamado RMAP e é capaz de relatar os alinhamentos para cada amostra com qualquer número de diferenças. Tal nível de sensibilidade representa um comportamento muito demorado. Dirigido pelo CloudBurst através Hadoop tem o objetivo de colocar em paralelo a execução em vários nós de computação.

O principal comando no algoritmo *executar-sga-hadoop-cloudburst.sh* é o que executa a aplicação CloudBurst. Ele é apresentado no Script 2. Os outros comandos para gerenciar os diretórios no sistema de arquivos *HDFS* foram omitidos por questões de espaço.

Script 2 Comando para iniciar o CloudBurst no Hadoop

```
# /usr/local/hadoop/bin/hadoop jar CloudBurst.jar
/user/hd/cloudburst/s_suis.br /user/hd/cloudburst/100k.br
/user/hd/cloudburst-results 36 38 3 0 1 20 4 2 2 128 16
```

3.2. Cenários de testes para computação intensiva de dados

Foram realizados testes com um arquivo de log de 3,263Mbytes para a procura de uma *string* no primeiro caso de estudo. O *NameNode* e o *JobTracker*, componentes da máquina mestre do Hadoop, foram configurados em uma máquina com processador Intel Core2 Duo CPU E7200 de 2.53GHz. Esta máquina também disponibiliza um modulo de escravo com um *DataNode* e um *TaskTracker*. Foram conectadas mais duas máquinas iMAC para servirem como *DataNode* e *TaskTracker*, componentes escravos do Hadoop, com um processador Intel Core i5-2400S CPU de 2.50GHz de quatro núcleos. O tempo de processamento está descrito na Tabela 1. Com a plataforma MC^2 o tempo de processamento foi superior em 8 segundos, reflexo da transferência de arquivos de entrada e processamento extras como verificação se o serviço do Hadoop está sendo executado, verificação se o diretório no sistema de arquivos *HDFS* já existe

para não sobreescrevê-lo, cópia do arquivo para dentro do sistema de arquivos *HDFS* e cópia do arquivo de resultado para a saída do CSGrid.

Tabela 1. Tempo de processamento do Grep *MapReduce*

Aplicação	Tempo em segundos
MapReduce Grep	19,6
MapReduce Grep com CSGrid	27,0

O teste com o segundo caso de estudo foi feito usando as mesmas configurações do exemplo anterior. A Tabela 2 mostra que a aplicação CloudBurst tem um atraso para processar os dados quando incorporada na plataforma MC^2. Isto acontece pela mesma razão do primeiro caso de estudo, o tempo de transferência de arquivos pela rede. Entretanto, como este programa é uma aplicação científica, ele atende aos objetivos para que foi implantado na plataforma MC^2. Por isso, existe o planejamento de aumentar a carga de dados para este caso de estudo. Foram realizados quatro testes com 10 amostras cada, para o sistema Hadoop sozinho e depois integrado com a plataforma MC^2. No primeiro, apenas com uma máquina servindo como mestre e escravo. Nos seguintes foram adicionadas as máquinas iMAC com papeis de escravo. Depois do cálculo da média dos resultados foi também calculado o desvio padrão, como mostra a Figura 3.

Tabela 2. Tempo de processamento do CloudBurst

Aplicação	1 PC	1 PC + 1 iMAC	1 PC + 2 iMAC	1 PC + 3 iMAC
CloudBurst	101,685	56,254	47,724	42,193
CloudBurst com CSGrid	113,5	78,1	58,3	53,5

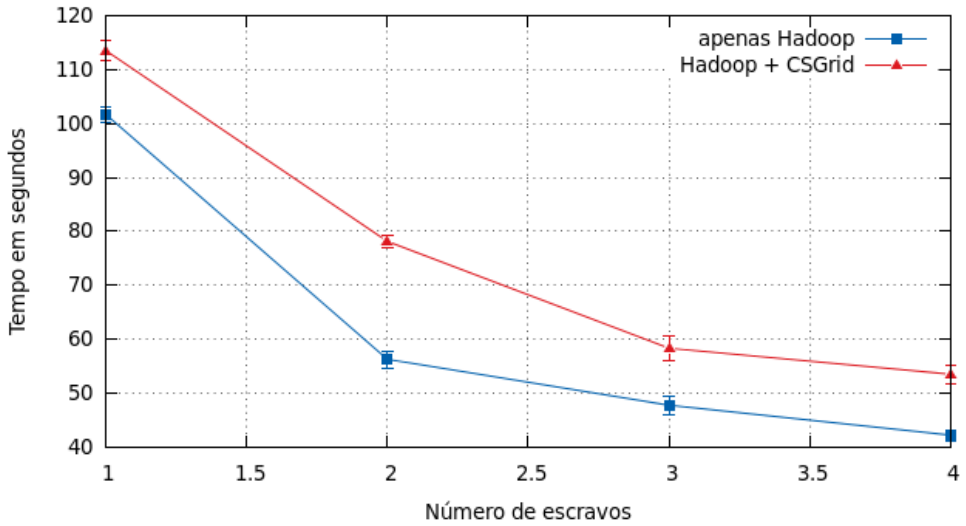

Figura 3. Métricas da aplicação CloudBurst no Hadoop na plataforma MC^2

4. Trabalhos relacionados

O CloudBLAST [Matsunaga et al. 2008] propõe e avalia uma abordagem de paralelização, implantação e gerenciamento de aplicativos de bioinformática que integram tecnologias emergentes de computação distribuída. Ele utiliza o paradigma de *MapReduce* para paralelizar ferramentas e gerenciar sua execução, a virtualização de máquinas para encapsular os ambientes de execução e rede de virtualização para conectar recursos através de *firewall* e NATs, enquanto preserva o desempenho necessário e a comunicação com o cliente. A implementação

integra o Hadoop, estações de trabalho virtuais, ViNe como MapReduce, máquinas virtuais e tecnologias de rede virtuais. O CloudBLAST utiliza apenas o modelo de programação intensiva de dados, enquanto o MC^2 trabalha com os quatro modelos de programação expostos na sua arquitetura, como mostra a Figura 1.

O projeto Kepler + Hadoop [Wang et al. 2009] é uma arquitetura generalizada que facilita a execução de fluxos científicos de aplicações intensiva de dados. Os cientístas podem utilizar a modelagem MapReduce como seus domínios específicos de problemas e conectá-los através da interface gráfica do Kleper [Altintas et al. 2004], um tipo de workflow científico. A plataforma MC^2 oferece mais flexibilidade que este projeto pois disponibiliza outros modelos de computação distribuída para o usuário cientísta e a integração de workflows entre estes modelos.

5. Conclusões

Com o provimento de uma interface para o processamento de computação intensiva de dados através do CSGrid, um usuário que tem apenas o conhecimento de aplicações científicas poderá acessar o CSGrid através do portal Web, que faz ligação com o Galaxy, e usufruir desta característica. Visto que o portal Web oferece uma interface com outros tipos de frameworks de computação distribuída, os usuários também poderão usar desta flexibilidade da plataforma MC^2 para realizar uma sequência de tarefas que eles necessitem para processar e analisar seus dados. O estudo de caso com a aplicação grep foi realizado apenas para provar o funcionamento da integração da plataforma MC^2 com o Hadoop. Já a aplicação CloudBurst foi um estudo de caso que atende o objetivo da plataforma, isto é, prover acesso à aplicação científicas. Estão sendo analisadas outras aplicações científicas juntamente com pesquisadores de áreas que necessitam do suporte da plataforma MC^2. Uma delas é o Contrail[4], um software desenvolvido baseado no modelo de programação MapReduce para a montagem de genomas com biologia computacional.

Referências

[Altintas et al. 2004] Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., and Mock, S. (2004). Kepler: an extensible system for design and execution of scientific workflows. *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 423–424.

[Anderson 2004] Anderson, D. P. (2004). Boinc: A system for public-resource computing and storage. In *5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10.

[Goecks et al. 2010] Goecks, J., Nekrutenko, A., and Taylor, J. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol, 11(8):r86 edition.

[K. Shvachko 2010] K. Shvachko, Hairong Kuang, S. R. R. C. (2010). The hadoop distributed file systems. pages 1–10. Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on Digital Object Identifier.

[Lima et al. 2005] Lima, M. J. D., Melcop, T., Cerqueira, R., Cassino, C., Silvestre, B., Nery, M., and Ururahy, C. (2005). CSGrid: um sistema para integração de aplicações em grades computacionais. Anais do 23o. Simpósio Brasileiro de Redes de Computadores - SBC.

[Lin and Dyer 2010] Lin, J. and Dyer, C. (2010). *Data-Intensive Text Processing with MapReduce*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

[4] http://contrail-bio.sourceforge.net

[Matsunaga et al. 2008] Matsunaga, A. M., Tsugawa, M. O., and Fortes, J. A. B. (2008). Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. In *eScience*, pages 222–229.

[Nazareno Andrade 2003] Nazareno Andrade, Walfredo Cirne, F. B. P. R. (2003). OurGrid: An approach to easily assemble grids with equitable resource sharing. pages 61–86. Springer Berlin Heidelberg.

[Schatz 2009] Schatz, M. C. (2009). Cloudburst: highly sensitive read mapping with mapreduce. *Bioinformatics*, 25(11):1363–1369.

[Wang et al. 2009] Wang, J., Crawl, D., and Altintas, I. (2009). Kepler + hadoop: a general architecture facilitating data-intensive applications in scientific workflow systems. In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, WORKS '09, pages 12:1–12:8, New York, NY, USA. ACM.

[White 2009] White, T. (2009). Hadoop - the definitive guide. In *Hadoop - The Definitive Guide*.