Uma Plataforma Flexível de Computação de Alto Desempenho para Solução de Problemas de Sensoriamento Remoto

Leandro L. Parente[1], Roberto U. Paiva[1], Claudinei O. Santos[1], Evandro C. Taquary[1], Laerte G. Ferreira[1]

[1]Lapig – Laboratório de Processamento de Imagens e Geoprocessamento (UFG)
CAMPUS II Samambaia - Cx. POSTAL 131 - Goiânia - GO / CEP: 74001-970

{leal.parente,robertourzeda,claudineisantosnx,evandro.taquary,lapig.ufg}@gmail.com.br

Abstract. *The increasing imaging capacity of the terrestrial surface, through governmental and private satellites, is producing an unprecedented remote sensing data amount. With ever better spatial and temporal resolutions, the transformation of this data into scientifically relevant information demands high-performance computing platforms. The objective of this work is to present a computational infrastructure that enables the practice of High Performance Computing to optimize the solutions of remote sensing problems. To this end, a flexible parallel and distributed computing platform is being implemented and tested in solving problems in this area. In this work we present the results and viability of the platform by running BFAST algorithm, used to detect changes in time series analysis. The results showed that the platform is promising, and can optimize solutions of remote sensing problems at low cost and little programming effort.*

Resumo. *O aumento da capacidade de imageamento da superfície terrestre, por meio de satélites governamentais e privados , está produzindo um volume de dados sem precedentes para a área de sensoriamento remoto. Com resoluções espaciais e temporais cada vez melhores, a transformação destes dados em informações cientificamente relevantes demanda plataformas de computação de alto desempenho. O objetivo deste trabalho é apresentar uma infraestrutura computacional que viabilize a prática da Computação de Alto Desempenho para acelerar a solução de problemas de sensoriamento remoto. Para tanto, uma plataforma flexível de computação paralela e distribuída está sendo implementada e testada na resolução de problemas desta área. No presente trabalho, apresentamos os resultados e viabilidade da plataforma executando o algoritmo BFAST, utilizado para detecção de mudanças em análises de séries temporais. Os resultados mostraram que a plataforma é promissora, e pode otimizar a solução de problemas da área de sensoriamento remoto a baixo custo e pouco esforço de programação.*

1. Introdução

O aumento da capacidade de imageamento da superfície terrestre, por meio de satélites governamentais [ROY et al., 2014] e privados [HOUBORG, R.; MCCABE, M. F. A., 2018], está produzindo um volume de dados sem precedentes para a área de sensoriamento remoto [HANSEN et al. 2012]. Com resoluções espaciais (*i.e.* correspondência espacial de um pixel em relação a uma porção do território) e temporais (*i.e.* quantidade de tempo necessária para

revisitar um mesmo local) cada vez melhores, a transformação destes dados em informações cientificamente relevantes demanda plataformas de computação de alto desempenho. Atualmente o Google Earth Engine, uma plataforma de *cloud-computing* capaz analisar dados de sensoriamento remoto em escala planetária, se coloca como uma alternativa para atender esta demanda, entretanto seus usuários precisam adequar suas rotinas de processamento para os paradigmas de programação e linguagens suportadas pela plataforma [GORELICK et al., 2017]. Esta característica dificulta a utilização de algoritmos já disponíveis em diversos frameworks pela comunidade científica em geral [Ma et al., 2015].

Diante disto, este trabalho buscou desenvolver uma plataforma flexível de computação de alto desempenho, específica para problemas de sensoriamento remoto, capaz de permitir a execução paralela de rotinas de processamento já implementadas. A rotina de processamento utilizada como estudo de caso foi o BFAST (*Breaks For Additive Seasonal and Trend*), um algoritmo de detecção de mudanças, que decompõe uma série temporal de pixels em três componentes (tendência, sazonalidade, e ruído), e está implementando em R [VERBESSELT et al., 2010]. A rotina de detecção de mudanças do BFAST utiliza um processo iterativo, por meio de ajuste de um modelo linear por partes, o que torna o algoritmo computacionalmente oneroso e justifica sua utilização na plataforma proposta. Sua execução ocorreu sobre dados MODIS, com resolução espacial de 250m e resolução temporal de 16 dias [FRIEDL et al., 2010], obtidos sobre o Bioma Cerrado - com uma extensão total de 2.045.000 km² - nos últimos 17 anos.

2. Ambientes de Testes

A execução da rotina de processamento ocorreu em em dois ambiente: em um servidor em torre, para a execução sequencial, e na plataforma proposta por este trabalho, para execução paralela e distribuída, com o intuito de obter resultados de performance.

2.1. Ambiente de teste sequencial

O servidor em torre utilizado para o teste de processamento sequencial possui as seguintes especificações de *hardware*:

- Placa mãe: *Dell PowerEdge* T420 3015M
- Processador: 2 processadores Intel Xeon CPU E5-2430 v2 - 2.50 *Ghz*
- Memória: 22*GB*
- Armazenamento: *HD* 3*TB*

2.2. Ambiente de teste paralelo/distribuído

Os testes de processamento em paralelo e distribuído foram realizados utilizando um grid com 60 nós (ver figura 1), no qual o servidor mestre encarregou-se de distribuir a mesma imagem do sistema operacional Debian 9, via NFS (*Network File System*), TFTP (*Trivial File Transfer Protocol*) e *boot* PXE (*Preeboot Execution Environment*), garantindo um ambiente idêntico e homogêneo para todos os nós. A tecnologia *Wake on Lan* foi utilizada para ligar e desligar os nós do *grid* e o tráfego de rede entre os nós ocorreu via rede *gigabit*. As especificações de *hardware* dos nós do *grid* são:

- Placa mãe: Intel modelo 525mw
- Processador: Intel Atom D525 (*Dual-Core*) - 1.80 *Ghz*
- Memória: 4*GB*
- Armazenamento: Partição de armazenamento *GlusterFS* - 4*TB*

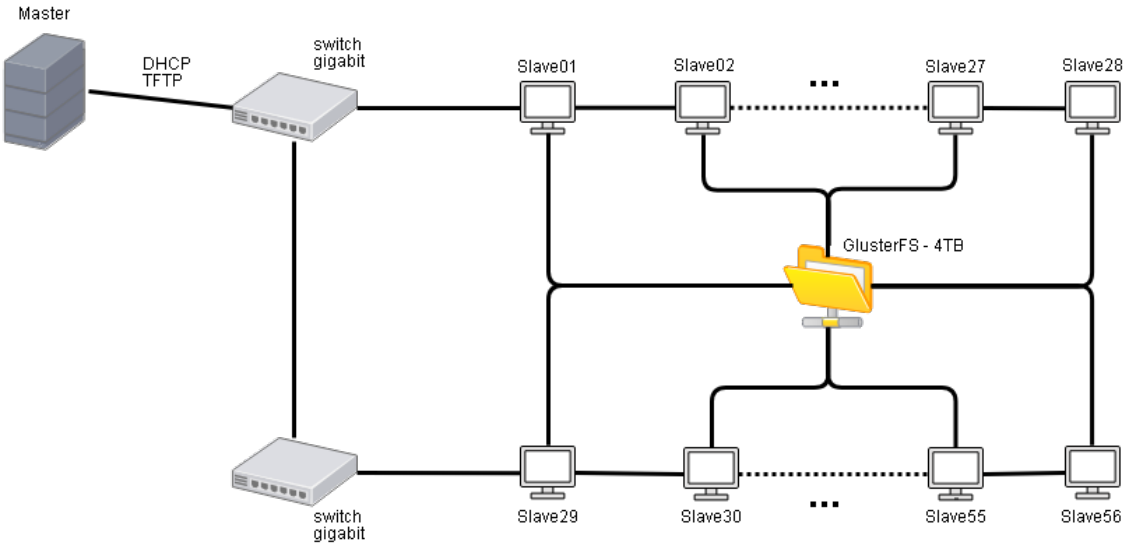

Figura 1. Arquitetura da plataforma, na qual o servidor Master encarregou-se de distribuir a mesma imagem de sistema operacional, com o intuito de garantir um ambiente idêntico e homogêneo, para todos os nós. O GlusterFS utilizou uma estratégia distribuída e replicada para armazenar e compartilhar os dados na plataforma.

Para armazenamento e compartilhamento de dados, foi utilizado um sistema de arquivos distribuído, em 4 nós via *GlusterFS*. O GlusterFS é uma tecnologia de armazenamento que permite, a partir de vários volumes hospedados em diferentes servidores, a construção de um sistema de arquivos em rede replicado e distribuído [DAVIES, A.; ORSARIA, A., 2013]. Um *benchmark* de performance entre o *GlusterFS*, *Red Hat® Ceph Storage* e *Hadoop Distributed File System* foi utilizado como critério de escolha de sistema de arquivo distribuído [TESTING, 2018]. O *GlusterFS* dispõe de diversas estratégias de alocação dos dados, a partição de armazenamento da arquitetura optou por uma estratégia distribuída e replicada, devido seu melhor desempenho de leitura de arquivos e garantias de confiabilidade e escalabilidade [RED HAT, 2018].

2.3. Estratégia de Paralelização

O problema em questão é "trivialmente paralelizável" [HERLIHY, 2011], sendo possível atribuir uma tarefa de processamento a cada série temporal, para serem executadas simultânea e independentemente, sem necessidade de intercomunicação. Desta forma, num cenário ideal, poderíamos associar cada série a um núcleo de processamento e, em tese, realizar o processamento de todas as séries temporais com o mesmo tempo de processamento de uma única. Nesse sentido os testes foram realizados atribuindo blocos de séries-temporais a tarefas de processamento que foram executadas concorrentemente nos nós do cluster. Ao todo foram processados 1.820 blocos com 27.300 séries temporais com 414 observações cada. O tempo total de processamento foi aproximadamente 6 dias, com consumo máximo de CPU em 56 nós. O processamento foi paralelizado via Slurm (Simple Linux Utility for Resource Management), um distribuidor de tarefas com funcionalidades de gerenciamento de recursos dos nós, tal como alocação de memória e núcleos de processamento [YOO, 2003].

3. Resultados

O tempo total de execução do BFAST, incluindo leitura/escrita dos dados e processamento, foi

~303 horas, ou 12,6 dias, em um ambiente de testes multicore e ~152 horas, ou 6,3 dias, no ambiente de testes do cluster. Algumas considerações em relação aos ambientes de testes devem ser discutidas. Foi utilizado um servidor de médio custo para a realização dos testes no ambiente multicore, uma máquina com um sistema de arquivos local, sem tempo de rede para leitura e escrita dos dados, com 12 processadores físicos (24 núcleos no total) e tempos de *clock* e arquitetura mais robusta do que as máquinas do *grid*. O *grid*, no entanto, possui máquinas de baixo consumo e custo, com arquiteturas mais antigas e simples, fazendo uso de um ambiente de armazenamento compartilhado na rede. Mesmo diante deste contexto, a plataforma se mostrou mais eficiente em termos de performance.

4. Conclusão

Este trabalho desenvolveu uma plataforma inicial de computação de alto desempenho na qual foi possível utilizar algoritmos e rotinas de processamento prontas, sem nenhuma necessidade de re-implementação, a qual permitiu diminuir pela metade o tempo de execução do algoritmo BFAST. As tecnologias utilizadas pela plataforma permitem aumentar a escalabilidade de processamento, sendo relativamente simples incluir outros nós de processamento, homogêneos e/ou heterogêneo na arquitetura. Os resultados mostraram a possibilidade desta plataforma tirar proveito do poder de processamento latente de equipamentos *commodities*, a um custo relativamente baixo e com muito pouco esforço de programação. Não obstante, a arquitetura proposta também pode ser implantada em equipamentos mais robustos, tanto quando em ambientes de *cloud-computing*. Como ideia para trabalhos futuros, pode-se pensar em criar containers de aplicação (ex. *Docker*) a fim de dinamizar a implantação da plataforma em qualquer infraestrutura computacional, aumentando ainda mais sua flexibilidade e operação em diversos ambientes computacionais.

Referências

[1] ROY, D.P.; WULDER, M.A.; LOVELAND, T.R.; WOODCOCK, C.E; ALLEN, R.G.; ANDERSON, M.C.; HELDER, D.; IRONS, J.R.; JOHNSON, D.M.; KENNEDY, R.; SCAMBOS, T.A.; SCHAAF, C.B.; SCHOTT, J.R.; SHENG, Y.; VERMOTE, E.F.; BELWARD, A.S.; BINDSCHADLER, R.; COHEN, W.B.; GAO, F.; HIPPLE, J.D.; HOSTERT, P.; HUNTINGTON, J.; JUSTICE, C.O.; KILIC, A.; KOVALSKYY, V.; LEE, Z.P.; LYMBURNER, L.; MASEK, J.G.; MCCORKEL, J.; SHUAI, Y.; TREZZA, R.; VOGELMANN, J.; WYNNE, R.H.; ZHU, Z. Landsat-8: Science and product vision for terrestrial global change research. Remote sensing of Environment, v. 145, p. 154-172, 2014.

[2] HOUBORG, R.; MCCABE, M. F. A. Cubesat enabled Spatio-Temporal Enhancement Method (CESTEM) utilizing Planet, Landsat and MODIS data. Remote Sensing of Environment, v. 209, p. 211-226, 2018.

[3] HANSEN, M. C.; LOVELAND, T. R. A review of large area monitoring of land cover change using Landsat data. Remote sensing of Environment, v. 122, p. 66-74, 2012.

[4] GORELICK, N.; HANCHER, M.; DIXON, M.; ILYUSHCHENKO, S.; THAU, D.; MOORE, R. Google Earth Engine: Planetary-scale geospatial analysis for everyone. Remote Sensing of Environment, v. 202, p. 18-27, 2017.

[5] Ma, Y.; Wu, H.; Wang, L.; Huang, B.; Ranjan, R.; Zomaya, A.; Jie, W. Remote sensing big data computing: challenges and opportunities. Future Generation Computer Systems, v. 51, p. 47-60, 2015.

[6] VERBESSELT, J., HYNDMAN, R., NEWNHAM, G., CULVENOR, D. Detecting trend and seasonal changes in satellite image time series. Remote sensing of Environment, v. 114, n. 1, p. 106-115, 2010.

[7] FRIEDL, M. A.; SULLA-MENASHE, D.; TAN, B.; SCHNEIDER, A.; RAMANKUTTY, N.; SIBLEY, A.; HUANG, X. MODIS Collection 5 global land cover: Algorithm refinements and characterization of new datasets. Remote sensing of Environment, v. 114, n. 1, p. 168-182, 2010.

[8] DAVIES, A., & ORSARIA, A. Scale out with GlusterFS. Linux Journal, v. 2013, n. 235, p. 1, 2013.

[9] TESTING OF SEVERAL DISTRIBUTED FILESYSTEMS (HDFS , CEPH AND GLUSTERFS) FOR SUPPORTING THE HEP EXPERIMENTS ANALYSIS. SEMANTICSCHOLAR. Disponível em: <https://www.semanticscholar.org/paper/Testing-of-several-distributed-filesystems-(-HDFS-%2C-Donvito-Marzulli/776dc2f7cdeb32274d0c0e5fecf7ef9a13fcaf13>. Acesso em: 15 ago. 2018.

[10] RED HAT GLUSTER STORAGE TECHNICAL PRESENTATION. Red Hat. Disponível em: <https://www.redhat.com/cms/managed-files/Red%20Hat%20Gluster%20Storage%20Technical%20Deck%281%29.pdf>. Acesso em: 15 ago. 2018.

[11] HERLIHY, M., & SHAVIT, N. The art of multiprocessor programming. Morgan Kaufmann, 2011.

[12] YOO, A. B., JETTE, M. A., GRONDONA, M. Slurm: Simple linux utility for resource management. In: Workshop on Job Scheduling Strategies for Parallel Processing. Springer, Berlin, Heidelberg, 2003. p. 44-60.