MOBI: Um Módulo para Monitorar Aplicações Big Data em Ambientes de Nuvem Heterogêneos

Jobe D. Santos[1], **Kassiano J. Matteussi**[1], **Paulo R. R. de Souza Junior**[1], **Claudio R. Geyer** [1]

[1]Universidade Federal do Rio Grande do Sul (UFRGS)
Instituto de informática - GPPD
Caixa Postal 15.064 - 91.501-970 - Porto Alegre, RS, Brasil
{jobe.dylbas, kjmatteussi, prrsjunior, geyer}@inf.ufrgs.br

Resumo. *Aplicações Big Data são amplamente utilizadas por organizações que buscam obter informações em meio a bases de dados complexas. Neste artigo, apresenta-se o MOBI: um módulo para monitorar aplicações Big Data em ambientes de nuvem heterogêneos. MOBI possui comportamento não-intrusivo e permite minimizar ou anteceder problemas de interferência durante o processamento das aplicações, levando a melhor utilização dos recursos computacionais.*

1. Introdução

O número de dispositivos conectados à internet aumenta consideravelmente a cada ano, e os dados produzidos por estes sensores, *smartphones*, entre outros, têm sido considerados cada vez mais valiosos pelas organizações. Estes dados na forma crua podem não possuir significado, mas analisados podem potencializar a tomada de decisões estratégicas e gerar informações significativas nos mais variados segmentos.

Monitorar as estruturas que processam tais informações se tornou crucial para manter a eficiência e integridade dos ambientes de processamento. Porém, o monitoramento possui desafios, como por exemplo analisar se os *frameworks* Big Data estão processando o dado com a miníma interferência possível. Assim, este artigo apresenta o MOBI, um módulo de monitoramento de aplicações *Big Data*, que considera as diferentes necessidades dos ambientes heterogêneos de larga escala.

2. Estado da Arte

Ambientes de nuvem fornecem segurança, eficiência, flexibilidade e escalabilidade para suportar o processamento de grande volumes de dados requisitados por aplicações e *frameworks* Big Data [Assunção et al. 2015]. Entretanto, em ambientes não controlados podemos ter uma sobreutilização de recursos gerando gargalos de processamento, ou uma subutilização, isto é, existem recursos disponíveis que não estão sendo utilizados.

Para suprir tais problemas mecanismos de monitoramento podem ser adotados, permitindo por exemplo, a avaliação sobre a condição da infraestrutura. No mercado existem diversos monitores, tais como: Nagios [Luchian et al. 2016], DARGOS [Povedano-Molina et al. 2013] e Ganglia [Massie et al. 2004] - cujo foco são *clusters* e *grids*. Esses monitores propiciam diversas métricas para o melhor controle sobre o ambiente e, por meio de uma interface *web* com suporte *near real-time* fornecem a visualização dos dados. Contudo, muitos desses serviços e aplicações de monitoramento não são adaptativos a multi-ambientes [Fatema et al. 2014] nem a ambientes heterogêneos, ou seja, compostos por diversos tipos de dispositivos.

O monitoramento deve ser feito de forma não-intrusiva, de modo que não interfira na execução das aplicações, garantindo a qualidade do Serviço (QoS) para com o

ambiente. Para que isso seja possível, o *trade-off* entre qualidade de monitoramento e interferência nos recursos deve ser considerado, ou seja, mecanismos devem ser adotados para minimizar tal interferência, caso ela exista. Além disto, sistemas de monitoramento devem ser escaláveis quanto a sua capacidade de monitorar. Em [Andreolini et al. 2013] estes critérios são analisados e métricas são propostas para avaliar o monitoramento de recursos em alta escala.

3. Arquitetura Smart: Uma Visão Geral

O objetivo da arquitetura SMART [dos Anjos et al. 2015] é simplificar o processamento Big Data. A SMART é formada por seis grandes módulos, conforme representado na Figura 1, são eles: *Global Collector, Global Dispatcher, Storage, Core Engine, Global Aggregator e Central Monitoring*.

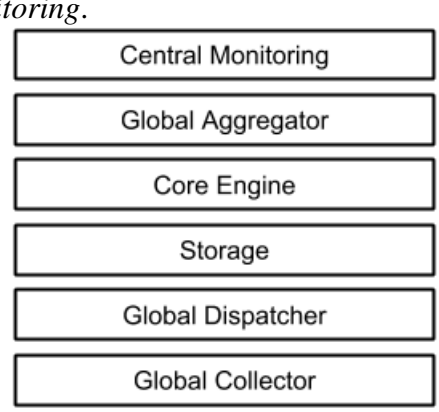

Figura 1. Smart Architecture

(i) O *Global Collector* é responsável por coletar e manter a integridade dos dados fornecidos pelos mais diversos dispositivos; (ii) No *Global Dispatcher* é onde os dados coletados são divididos em diferentes filas para assim serem distribuídos nos servidores de acordo com a disponibilidade dos mesmos e então armazenados no (iii) *Storage*; Os dados são processados pelo *Core Engine* gerando as informações relevantes que são levadas pelo (v) *Global Aggregator* ao usuários finais que acessam estes através do (vi) *Central Monitoring*.

A arquitetura SMART precisa ser constantemente monitorada para minimizar possíveis problemas com interferência ligadas a utilização de recursos que, podem implicar em perda de desempenho por parte das aplicações que estiverem em execução. Então, o MOBI é apresentado como componente desta arquitetura fazendo parte do módulo *Central Monitoring*, visando monitorar os recursos do módulo de processamento de dados (i.e. *Core Engine*).

4. MOBI: Arquitetura e Implementação

O MOBI é composto por cinco módulos: (i) O módulo de Plugins que fica encarregado em adaptar os diversos coletores de dados, como dstat[1], htop[2] e outros ao MOBI; (ii) O módulo Engine é responsável pela captura, inserção no banco de dados e ainda coletar e enviar os dados requisitados pela (iii) REST API, que por sua vez, funciona como um meio de comunicação entre a GUI e a Engine, traduzindo as requisições do usuário; Por

[1]http://dag.wiee.rs/home-made/dstat/
[2]http://hisham.hm/htop/

fim, (iv) GUI é uma interface de usuário onde pode-se requisitar quais recursos e máquinas deseja-se monitorar tanto de forma automática quanto customizada. Sua arquitetura do MOBI pode ser observada na na Figura 2.

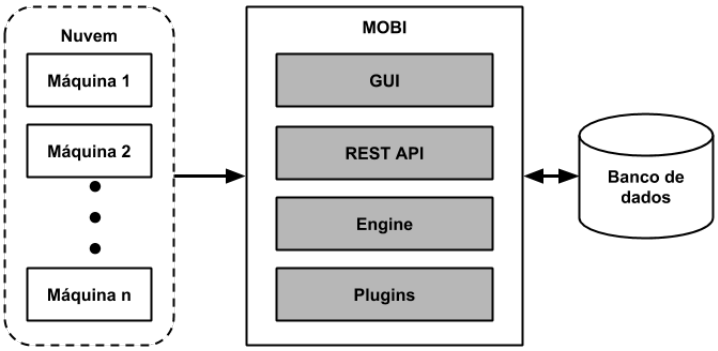

Figura 2. Arquitetura MOBI

As tecnologias empregadas em seu desenvolvimento são abertas e largamente utilizadas por diversos serviços, como Paypal e Uber. Entre as escolhas estão o MongoDB como banco de dados, devido a fácil utilização e do formato JSON por ser compacto, nativamente reconhecido pela linguagem JavaScript e de clara compreensão [Povedano-Molina et al. 2013]. Os serviços do servidor são desenvolvidos com Node.js, escolhido por ser baseado em eventos e por ser *I/O* não bloqueante, que permite otimizar o uso dos recursos do servidor [Chitra and Satapathy 2017].

5. Resultados Preliminares

Atualmente, o usuário ou administradores do sistema podem monitorar e analisar em tempo real a utilização dos recursos computacionais de uma nuvem. Além disso, o usuário pode configurar a ferramenta de monitoramento, setando quais recursos deseja monitorar por meio de uma GUI (Figura 3). Afora isso, é possível exportar os gráficos referente ao uso de recursos de forma instantânea, bem como é possível obter os *traces* contendo tais dados. Isso possibilita que ações sejam tomadas para minimizar questões ligadas aos problemas interferência como o congestionamento de rede e contenção de disco, por exemplo.

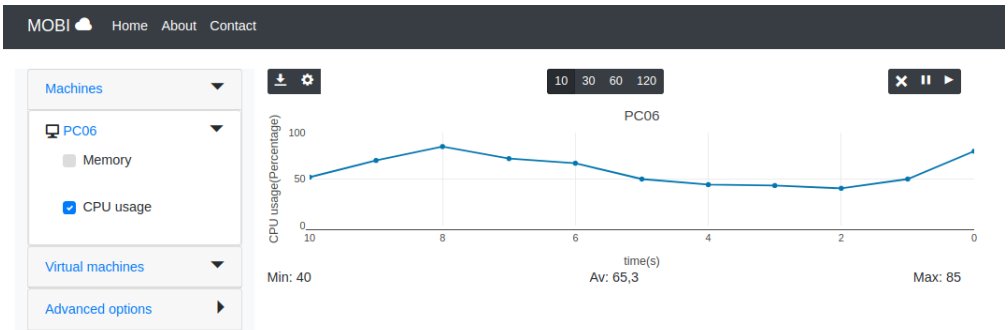

Figura 3. GUI - MOBI

Por fim, futuramente pretende-se avaliar outros servidores de banco de dados e protocolos de comunicação (UDP, TCP e etc.), visando equilibrar a integridade dos dados e sua disponibilidade em tempo real. Além de que para melhor avaliação dos recursos

estuda-se quais serviços são interessantes ou necessários, tais como formato de gráficos, granularidade, exportação de dados e avisos.

6. Considerações Finais

Este artigo apresentou um módulo de monitoramento, o MOBI, o qual utiliza técnicas e ferramentas presentes na literatura com objetivo de minimizar o impacto no desempenho de ambientes heterogêneos de processamento Big Data.

7. Agradecimentos

O presente trabalho foi realizado com o apoio da Pró-Reitoria de Pesquisa - UFRGS - Brasil.

Referências

Andreolini, M., Colajanni, M., Pietri, M., and Tosi, S. (2013). Real-time adaptive algorithm for resource monitoring. In *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, pages 67–74.

Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., and Buyya, R. (2015). Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 79-80(Supplement C):3 – 15. Special Issue on Scalable Systems for Big Data Management and Analytics.

Chitra, L. P. and Satapathy, R. (2017). Performance comparison and evaluation of node.js and traditional web server (iis). In *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, pages 1–4.

dos Anjos, J. C. S., Assunção, M. D., Bez, J., Carissimi, A., Costa, J. P. C. L., Freitag, F., Markl, V., Fergus, P., Pereira, R., de Freitas, E. P., Fedak, G., and Geyer, C. F. R. (2015). Smart: An application framework for real time big data analysis on heterogeneous cloud environments. In *In proceedings of 15th IEEE International Conference on Computer and Information Technology (CIT-2015), Liverpool, England, UK, October 2015*. IEEE Computer Society.

Fatema, K., Emeakaroha, V. C., Healy, P. D., Morrison, J. P., and Lynn, T. (2014). A survey of cloud monitoring tools: Taxonomy, capabilities and objectives. *Journal of Parallel and Distributed Computing*, 74(10):2918 – 2933.

Luchian, E., Docolin, P., and Dobrota, V. (2016). Advanced monitoring of the openstack nfv infrastructure: A nagios approach using snmp. In *2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC)*, pages 51–54.

Massie, M. L., Chun, B. N., and Culler, D. E. (2004). The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817 – 840.

Povedano-Molina, J., Lopez-Vega, J. M., Lopez-Soler, J. M., Corradi, A., and Foschini, L. (2013). Dargos: A highly adaptable and scalable monitoring architecture for multi-tenant clouds. *Future Generation Computer Systems*, 29(8):2041 – 2056. Including Special sections: Advanced Cloud Monitoring Systems & The fourth IEEE International Conference on e-Science 2011 — e-Science Applications and Tools & Cluster, Grid, and Cloud Computing.