Análise Comparativa de Propostas de Dependabilidade Aplicadas à Computação Intensiva de Dados[1]

Clícia S. Pinto[1], Amanda Chagas[1], Marcos Barreto[1]

[1] LaSiD – Laboratório de Sistemas Distribuídos – (UFBA)
Av. Adhemar de Barros, s/n – Campus Ondina
40170-110 – Salvador, BA - Brasil

{cliciasantosp,amandachagas,marcoseb}@dcc.ufba.br

Abstract. *This work focuses on core concepts needed to implement dependability for data-intensive applications. A brief review of the state-of-the-art and some relevant solutions are presented.*

Resumo. *Este artigo busca prover fundamento teórico inicial para a implantação de um módulo de dependabilidade aplicado ao cenário de computação intensiva de dados. Para isto, serão realizadas análises comparativas do estado da arte, bem como descrição de soluções pertinentes ao problema.*

1. Introdução

A fim de aumentar a eficácia das medidas de dependabilidade em ambientes computacionais, técnicas de tolerância a falhas que compreendem desde backup dos dados até redundância dos equipamentos e espelhamento dos discos são utilizadas. Entretanto, todas as técnicas que colaboram para aumentar a dependabilidade de um sistema têm um custo associado. Tendo isto em vista, a aplicação destas técnicas requer um bom projeto, envolvendo entre outros fatores, a relação custo/benefício. Este trabalho estabelece uma análise comparativa entre algumas técnicas de tolerância a falhas desenvolvidas pela comunidade científica e suas implementações, tendo como objetivo a aplicação de tais técnicas no contexto de uma plataforma específica de computação intensiva de dados.

2. Técnicas para Implantação de Dependabilidade: Estado da Arte

No contexto de computação na nuvem, vários aspectos específicos de dependabilidade devem ser considerados. Em relação à infraestrutura física, por exemplo, a questão é como comprovar que o hardware que compõe esta infraestrutura será capaz de garantir a perfeita operabilidade dos recursos que executarão sobre ele. Em relação ao software a questão é garantir tolerância e recuperação de falhas, analisar vulnerabilidades entre outras questões próprias de cada plataforma.

Tolerância a falhas, um dos principais atributos de dependabilidade, é a propriedade que garante que o sistema não falhe ou apresente defeito mesmo com a ocorrência de um erro. Neste sentido, o *Hadoop* [White 2009] foi construído para ser altamente tolerante. Isto porque ele foi desenvolvido para arquiteturas de hardware de

1 Trabalho parcialmente financiado pela UFBA, edital PROPCI/UFBA 01-2013 - PIBIC e PIBIC-AF

baixo custo o que torna a probabilidade de falhas um problema crítico. Segundo Patil e Soni (2013), a tolerância à falhas é a vantagem mais importante ao se utilizar a solução *Hadoop*. Os dois principais métodos para se alcançar tal vantagem são: duplicação de dados e *checkpoint/recovery*. Duplicação de dados consiste em copiar os dados de um bloco em outro disponível. A maior vantagem dessa técnica é a recuperação instantânea dos dados na presença de uma falha. No entanto, é necessário alto custo de memória para tal armazenamento de dados duplicados. Em técnicas de *checkpoint/recorvery*, a cada período de tempo é salva e armazenada uma cópia do estado do sistema. Dessa forma, o consumo de memória é menor em relação ao anterior, pois cada *checkpoint* salvo, substitui o anterior. Apesar disto, os autores ainda alertam para o fato de que, apesar de todas as medidas de segurança, o *Namenode* ainda representa um ponto único de falhas que pode comprometer todo o sistema, pois caso falhe todo o sistema irá falhar. Estudar os mecanismos para recuperação de falhas no *Namenode* ainda é, portanto, um dos principais desafios.

O *Hadoop* possui mecanismos para tratar falhas nos nós que compõem o ambiente de execução, isto é, o Namenode, que regula o acesso aos arquivos, e o Datanode, que armazenam os blocos de dados dos arquivos. Contudo, a fim de analisar o seu comportamento em uma situação de falha não prevista, Gondim, Barcelos e Charão (2012) propuseram uma experiência de injeção de falhas no *DataNode*. Na experiência em questão foi aplicada a técnica de Instrumentação de Código com o auxílio de um emulador de falha que está disponível no próprio *Hadoop*, o *Apache Hadoop Fault Injection Framework*. As experimentações por injeção de falhas dadas em tempo de execução comprovaram que a plataforma não oferece um tratamento adequado para falhas de memória, que podem ser comuns em aplicações de larga escala. Testes de injeção de falhas como este são amplamente utilizados para avaliar mecanismos tolerantes a falhas além de fornecer informações importantes sobre o quão confiável uma platafoma realmente é para a aplicação a que ela se destina.

A fim de aumentar as métricas de confiabilidade de sistemas baseados em muitas imagens de máquinas, métodos de *checkpoint* são fundamentais. As soluções usuais podem implicar em grande sobrecarga na rede. O primeiro complicante é que tais *checkpoints* precisam ser armazenados em disco e em cenários de muitas VM's este se torna um processo dispendioso. O segundo complicante se refere à periodicidade com que os checkpoints são realizados. O intervalo não deve ser nem tão grande a ponto de que a última cópia se torne inconsistente, nem tão pequeno que comprometa todo tráfego da rede. Nesse sentido, Goiri, Julià e Torres (2010) propuseram uma infraestrutura utilizando *Another Union File System* para otimização de *checkpoints*. Esta solução utiliza o sistema de arquivos para dividir a imagem da VM em uma parte somente leitura e outra leitura e escrita. Deste modo, a parte somente leitura é salva apenas uma vez e os outros *checkpoints* são realizados apenas em partes que sofreram alteração. Os *checkpoints* são armazenados no HDFS, o que aumenta significativamente a tolerância a falhas deste sistema ao passo em que reduz os problemas de desempenho.

Neste mesmo princípio, França (2013) utiliza uma técnica de otimização de checkpoints que busca reduzir o custo de sua operação sobre a rede. A solução apresentada ajusta o intervalo de tempo entre os *checkpoints* conforme a necessidade, estabelecendo uma taxa máxima de transmissão que as operações de *checkpoint* podem usar, assim se estas operações não estiverem utilizando a capacidade de rede que lhe foi

atribuída o intervalo é ajustado para diminuir a quantidade de *checkpoints* gerados por unidade de tempo, melhorando assim o desempenho da VM. Entretando, a solução proposta for França, não se aplica de forma adequada em ambientes comerciais e de aplicações críticas, pois ajustar o intervalo de *checkpoints* de acordo com um limite de transmissão máximo permitido pode representar uma vulnerabilidade no cumprimento do SLA.

O problema do intervalo ideal ainda não possui soluções bem consolidadas, especialmente em aplicações de processos paralelos, como mostra Fialho, Rexachs e Luque (2011). Os autores, em seu trabalho, estabelecem uma relação entre tais aplicações paralelas (que realizam troca de mensagens com outros *peers)* na definição do intervalo de *checkpoint* mais apropriado para protocolos de *checkpoint* não-coordenados.

3. Integração do Módulo de Dependabilidade

O cerne desta pesquisa compreende traçar estratégias para a elaboração de um módulo de suporte à dependabilidade que sirva a aplicações e-science e possivelmente aplicações de robótica. As camadas da plataforma na qual o módulo será inserido são descritas conforme a Figura 1. O módulo de dependabilidade, que está na camada de plataforma como serviço, deve ser requerido conforme necessário e ajustado pelo próprio usuário da aplicação.

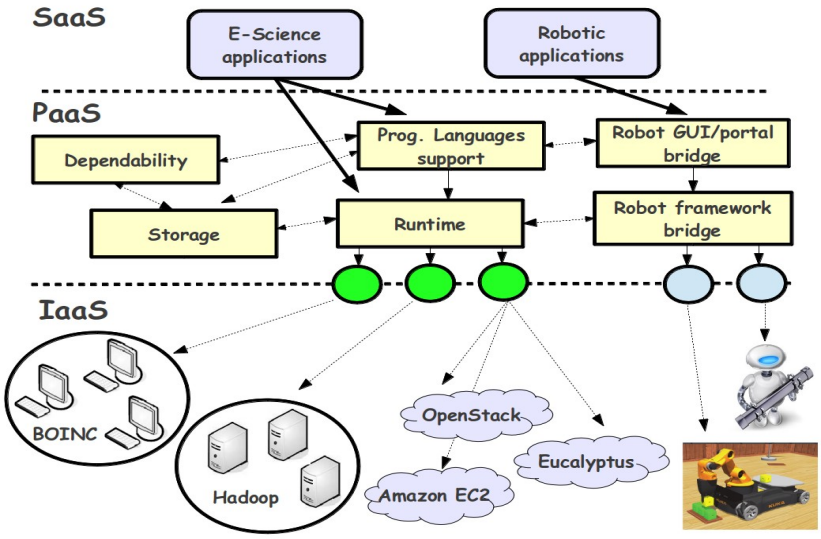

Figura 1. Visão geral da infraestrutura do projeto

4. Conclusão e Trabalhos Futuros

Espera-se com esta pesquisa obter o levantamento literário suficiente para que, no futuro, um módulo de dependabilidade seja agregado ao projeto em questão e as devidas avaliações sejam realizadas. A técnica de injeção de falhas no Hadoop, aqui descrita, utilizando um framework da sua própria plataforma, se apresenta como uma técnica de

validação experimental apropriada para ser aplicada na fase inicial desta pesquisa bem como as técnicas de otimização de *checkpoints*, que são indispensáveis para prover tolerância a falhas, com o desafio de minimizar o overhead causado por eles. A aplicação destas técnicas pode ser um ponto de partida no sentido de se testar o nível de confiabilidade adequado e garantir um maior nível de tolerância a falhas, baseando-se nos SLAs firmados entre o provedor do serviço de computação e o usuário.

Referências

White, T. (2009) Hadoop - The definitive guide, O'Reilly Media Inc., 1ª Edição.

Patil,V. e Soni, P. (2013) “Hadoop Skeleton and Fault Tolerance in Hadoop Clusters”, International Journal of Application or Inovation in Engineering and Management, p. 247-250.

Gondim, E., Barcelos, P. e Charão, A. (2012) Experiência de Injeção de Falhas no DataNode do Apache Hadoop.

Goiri, I., Julià, F. e Torres, J. (2010) Checkpoint-based Fault-tolerant Infrastructure for Virtualized Service Providers.

França, J. (2013) Otimização de Checkpoint de Máquina Virtual para Tolerância a Falhas.

Fialho, L., Rexachs, D. e Luque E. (2011) “What Is Missing in Current Checkpoint Interval Models? ”, 31st International Conference on Distributed Computing Systems , p. 322-332.