Avaliação de uma Arquitetura de Configuração Dinâmica para o Checkpoint no HDFS

Paulo V. M. Cardoso, Patrícia Pitthan Barcelos

Pós-Graduação em Ciência da Computação (PGCC)
Universidade Federal de Santa Maria (UFSM)
Santa Maria -- RS -- Brazil

{pcardoso,pitthan}@inf.ufsm.br

Resumo. *O estudo e a otimização de técnicas de tolerância a falhas torna-se mais importante à medida em que mais recursos são usados em conjunto. O framework Apache Hadoop e seu sistema de arquivos distribuído (HDFS) usam a técnica de Checkpoint and Recovery para garantir confiabilidade de execução. Porém, o período entre checkpoints é configurado de forma estática e não pode ser mudado em execução. Por isso, este trabalho define e avalia um mecanismo de configuração dinâmica para o checkpoint no HDFS.*

1. Introdução

A crescente demanda por processamento computacional de alto desempenho tem exigido que recursos computacionais sejam utilizados de forma conjunta, tornando a disponibilidade e a confiabilidade grandes desafios em sistemas HPC [Egwutuoha et al. 2013]. Uma opção de segurança que se apresenta de forma eficiente é a técnica de *Checkpoint and Recovery* (CR), em que a recuperação de uma falha é feita a partir de estados estáveis previamente salvos, evitando re-execuções completas [Cui et al. 2015].

A escolha pelo período ideal de *checkpoint*, porém, também surge como um desafio para sistemas que implementam a técnica de CR. O HDFS, usado pelo *framework* de alto desempenho Apache Hadoop, usa um mecanismo de CR com acessos frequentes ao disco. O período entre *checkpoints* no HDFS é estático e, por isso, tanto o desempenho quanto a confiabilidade podem ser prejudicados.

Este trabalho apresenta uma proposta de configuração dinâmica para o *checkpoint* no HDFS, a partir de monitoramentos de uso do ambiente e de uma coordenação distribuída. O objetivo do mecanismo é adaptar o intervalo de *checkpoints* para garantir níveis satisfatórios de confiabilidade com uma menor interferência no desempenho. O HDFS com mecanismo dinâmico é definido e, então, submetido a testes preliminares, comparando seu desempenho com a versão *default* do *framework*.

2. HDFS

O Apache Hadoop, projeto *open source* voltado para processamentos de alto desempenho em ambientes distribuídos, utiliza o HDFS (*Hadoop Distributed File System*) para oferecer suporte a arquivos de grandes dimensões. Um arquivo no HDFS é dividido em blocos de tamanhos pré-definidos e distribuídos através do ambiente. Neste caso, dados e metadados são armazenados de forma separada. Os dados são armazenados em *workers* chamados DataNodes (DN) e os metadados são mantidos pelo *master*, chamado NameNode (NN), que também mantém todo o *namespace* do HDFS.

A distribuição do HDFS faz com que blocos de um mesmo arquivo possam estar armazenados em nós diferentes, estimulando a tolerância a falhas no sistema. Dentre as técnicas usadas pelo HDFS para tolerar falhas, destaca-se o *checkpoint* (*Checkpoint and*

Recovery, ou apenas CR): um procedimento reativo, classificado como técnica de recuperação por retorno (*backward error recovery*), com o propósito de recuperar o estado falho de um sistema através de um contexto estável previamente salvo.

O *checkpoint* no HDFS é realizado a partir da replicação do seu *namespace* em um arquivo chamado FSImage, armazenado no sistema de arquivos local do NameNode. Nesse arquivo, são mantidas informações sobre o mapeamento de blocos para arquivos e propriedades do sistema. Para evitar que um novo FSImage seja criado para cada operação feita pelo HDFS, um log de edições (EditLog), também mantido em disco local, e armazena as últimas transações realizadas após a criação do FSImage.

O *merge* entre o FSImage e o EditLog consiste no processo de *checkpoint*. Esse procedimento é realizado quando o NameNode inicia e, posteriormente, de forma periódica. O intervalo de checkpoint é estático e definido em 3600 segundos na versão padrão, mas o processo pode ser disparado se o HDFS atingir um número específico de transações (10 mil, por padrão). O *checkpoint* é realizado pelo Secondary NameNode, que mantém uma cópia do FSImage e a atualiza a cada *checkpoint*.

3. Checkpoint Dinâmico

O processo de *checkpoint* pode se tornar um fator proibitivo para o desempenho do Apache Hadoop. Se o intervalo escolhido não for apropriado, os níveis de desempenho e confiabilidade são comprometidos. A versão do HDFS utilizada neste trabalho possui atributos de configuração de *checkpoint* estáticos, definidos antes da execução do *framework*. Esta característica impossibilita o *framework* de adaptar-se à utilização do ambiente, que pode sofrer alterações em razão das diferentes requisições que trabalha.

Este trabalho propõe um mecanismo de *checkpoint* dinâmico, com o objetivo de tornar a configuração do CR adaptável ao ambiente, visando auxiliar a confiabilidade e o desempenho do HDFS. A arquitetura do mecanismo proposto é composta por três componentes principais: o HDFS modificado, um módulo Coordenador e um módulo Monitor. A Figura 1 exibe a arquitetura e as relações entre cada módulo.

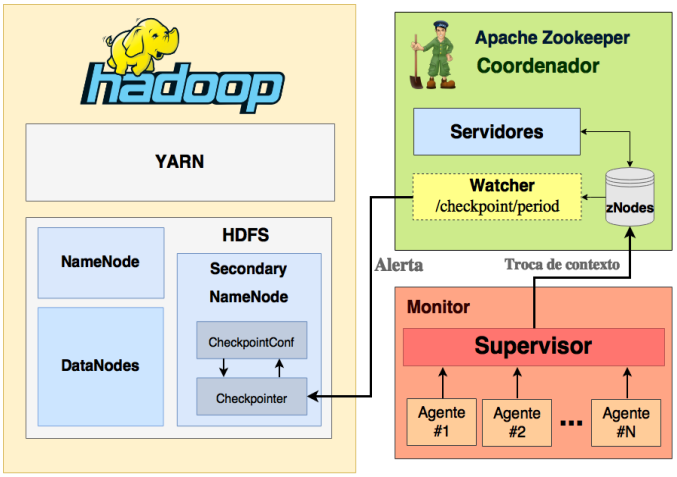

Figura 1. Arquitetura da configuração dinâmica de *checkpoints* no HDFS

Inicialmente, foram feitas alterações no Secondary NameNode e suas classes Checkpointer e CheckpointConf, responsáveis pelo procedimento de *checkpoint*. As mudanças incluem um elemento de comunicação entre o HDFS e o módulo Coordenador, além de um tratamento de alertas para alterações nos atributos de configuração em uma

troca de contexto. Caso o período seja modificado, o Checkpointer deve verificar se o novo intervalo já foi atingido na espera da configuração, invocando um processo de *checkpoint* se este fato ocorrer ou esperando o tempo restante.

Para o Coordenador, foi utilizado o *framework* Apache Zookeeper: um projeto *open source* que tem como principal objetivo a criação de um pacote de funcionalidades para facilitar a coordenação em sistemas distribuídos. Foram criados atributos de configuração para o *checkpoint*, os quais foram armazenados em estruturas distribuídas disponibilizadas pelo Zookeeper chamadas zNodes. O atributo atual é definido em um zNode específico, assim como os atributos já usados. Uma importante função do Coordenador é a configuração de mecanismos de alertas (*watchers*), que notificam e invocam ações no HDFS a partir de uma modificação no zNode do atributo atual.

Já o Monitor é composto por um supervisor geral e monitores individuais, chamados agentes. Um agente é implementado em cada nó do ambiente que seja necessário monitorar, a fim de avaliar seu desempenho. Os agentes monitoram seus respectivos nós através de uma verificação periódica de estatísticas baseadas no uso de CPU, RAM ou disco. A partir de uma alteração significativa no comportamento do nó - definida por métricas de avaliação -, uma mensagem de alerta é enviada ao supervisor.

4. Experimentação e Análise

Foram submetidos testes de desempenho visando uma comparação de desempenho entre o Apache Hadoop com o mecanismo proposto e a versão sem modificações, que possui configuração estática de *checkpoint*. As experimentações realizaram-se na plataforma Grid'5000 [Grid’5000 2017], onde foram configurados 10 nós, cada um com dois processadores Intel Xeon E5-2630v3 (oito *cores* por CPU) e 4GB de memória RAM.

A aplicação usada nos testes foi o TestDFSIO, que analisa o desempenho do ambiente em situações de leitura e escrita em disco. O *benchmark* foi usado para escrita de 20 arquivos, variando o tamanho de cada um em 16GB, 32GB e 64GB (no total 320GB, 640GB e 1280GB, respectivamente). Os resultados são exibidos na Tabela 1, com a média do tempo de execução de 20 amostras e o *overhead* de cada cenário. Todos os *overheads* são relacionados à execução de 3600 segundos na versão estática.

Pode-se notar que intervalos de *checkpoint* menores, quando estáticos, geram uma alta sobrecarga na execução da aplicação. Esse nível é maior conforme a quantidade de dados aumenta, chegando a mais de um terço do tempo de execução padrão. Já o mecanismo dinâmico apresenta um nível menor de sobrecarga em quase todos os cenários de monitoramento, se comparado à variação do mecanismo estático.

Além disso, como mostra a Figura 2, a escalabilidade mostrada pelas execuções com o *checkpoint* dinâmico possui uma melhor apresentação. Isto é, a variação do *checkpoin*t estático, conforme a quantidade de dados aumenta, é mais impactante do que a variação do tempo de monitoramento. Para auxiliar na compreensão dos resultados, a Tabela 2 exibe a quantidade de trocas de contexto observadas.

Hadoop	Configuração	Tamanho de cada arquivo					
		16GB		32GB		64GB	
		Execução (s)	Overhead	Execução (s)	Overhead	Execução (s)	Overhead
Estático	3600s	688,0	0%	1693,0	0%	3520,1	0%
	360s	743,1	7,4%	1746,6	3,1%	3752,3	9,34%
	36s	805,8	14,6%	2043,2	17,1%	4611,0	23,6%
	10s	949,83	27,5%	2249,5	24,7%	5291,3	33,5%
Dinâmico	*s/ monitor*	718,3	4,2%	1713,7	1,2%	3645,3	3,4%
	360s	733,3	6,18%	1746,6	3,1%	3740,6	5,8%
	36s	765,1	10,1%	1976,8	14,3%	3870,0	9,04%
	10s	809,8	15,04%	2078,53	18,5%	4178,4	15,7%

Tabela 1. Desempenho dos mecanismos testados com variação de sobrecarga.

Cenário	Intervalo	Trocas
16GB	360 s	2
	36 s	22
	10 s	58
32GB	360 s	3
	36 s	24
	10 s	68
64GB	360 s	4
	36 s	26
	10 s	75

Tabela 2. Trocas de contexto do mecanismo dinâmico.

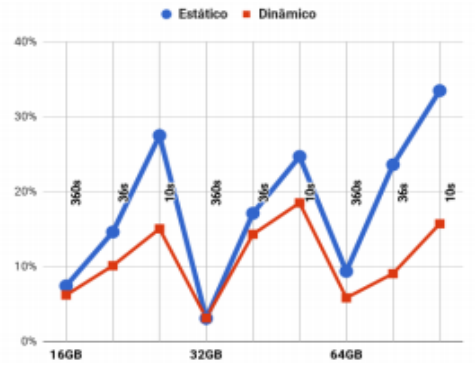

Figura 2. *Overhead* com variações de configuração nos mecanismos testados.

5. Considerações Finais

Este trabalho apresentou e validou um mecanismo de configuração dinâmica para o *checkpoint* do HDFS. Os testes realizados mostram que o mecanismo proposto possui um nível menor de sobrecarga nas aplicações, em relação à versão original do Hadoop, inclusive quando variou-se a quantidade de dados envolvidos. Nos próximos passos, a aplicabilidade do mecanismo será observada com a inclusão de falhas. Além disso, a escolha por métricas eficientes será estudada para definir-se um melhor uso do monitor.

Referências

Cui, L., Hao, Z., Li, L., Fei, H., Ding, Z., Li, B., and Liu, P. (2015). Lightweight virtual machine checkpoint and rollback for long-running applications. In International Conference on Algorithms and Architectures for Parallel Processing, p 577–596. Springer.

Egwutuoha, I. P., Levy, D., Selic, B., and Chen, S. (2013). A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems. The Journal of Supercomputing, 65(3):1302–1326.

Grid'5000 (2017 (acessado em julho de 2017)). Grid5000 Homepage. http://www.grid5000.fr/.