Avaliação Experimental de Mecanismos de Tolerância a Falhas no HDFS

Iago C. Corrêa, Paulo V. M. Cardoso, Patrícia Pitthan Barcelos

Laboratório de Sistemas de Computação
Universidade Federal de Santa Maria – UFSM

{icorrea,pcardoso,pitthan}@inf.ufsm.br

Resumo. *O uso de sistemas computacionais de alto desempenho robustos impulsionou a pesquisa sobre técnicas de tolerância a falhas. Porém, poucos trabalhos se preocupam em validar tais mecanismos, de forma a observar se os mesmos atendem aos seus objetivos. Este trabalho testa e analisa o funcionamento de técnicas de tolerância a falhas implementadas pelo sistema de arquivos do Apache Hadoop, submetendo-os a experimentos com indução de falhas.*

1. Introdução

Ferramentas voltadas para o processamento de *Big-Data* necessitam de mecanismos tolerantes a falhas, capazes de criar as condições necessárias para que a execução de aplicações ocorra de forma segura e sem comprometimento de suas informações, mesmo na ocorrência de falhas. Neste contexto, uma das ferramentas que se destaca é o Apache Hadoop [Foundation 2014], a qual possui mecanismos de *Data Replication* e *Checkpoint and Recovery*. Porém, a configuração dos atributos desses mecanismos é definida de forma estática e não pode ser modificada sem que o Hadoop e todos os seus serviços sejam interrompidos.

Visto que diferentes aplicações possuem demandas exclusivas, o processos de tolerância a falhas podem apresentar comportamentos distintos em cada cenário. Com uma metodologia de definição estática, a escolha de um atributo adequado consiste em um grande desafio que busca o equilíbrio ideal dessa escolha, a fim de oferecer confiabilidade ao sistema sem interferir no desempenho das aplicações.

Nesse sentido, o presente trabalho tem por finalidade testar e analisar os mecanismos de replicação de dados e checkpoint do sistema de arquivos do Hadoop, o HDFS (*Hadoop Distributed File System*). Serão explorados experimentos com variações nos atributos de configuração dos mecanismos utilizando-se o *Replication Factor* (RF), para a replicação, e o *Checkpoint Period* (CP) para o *checkpoint*.

O artigo está estruturado da seguinte maneira: a segunda seção aborda a estrutura da arquitetura do Hadoop, bem como alguns mecanismos de tolerância a falhas implementados. A Seção 3 explica a metodologia aplicada na fase de experimentação, enquanto a Seção 4 exibe os resultados obtidos e uma discussão sobre os mesmos. Por fim, a Seção 5 apresenta considerações finais e trabalhos futuros.

2. Apache Hadoop

O Apache Hadoop é um projeto *open source* voltado para o processamento distribuído de grandes quantidades de dados [White 2012] e que utiliza o modelo de programação *MapReduce* [Dean and Ghemawat 2008] para esse processamento. A arquitetura do Hadoop

utilizada nesse trabalho (versão 2.7.3) baseia-se na estrutura mestre-escravo composta por diversos módulos, dentre os quais se destacam: o *Hadoop Distributed File System* (HDFS), seu sistema de arquivos distribuído e o *YARN*, um *framework* para manutenção do ambiente e da aplicação. Este trabalho enfoca os mecanismos de tolerância a falhas implementados pelo HDFS [Foundation 2014].

2.1. Tolerância a Falhas no HDFS

Desenvolvido para suportar quantidades massivas de dados e oferecer um alto nível de confiabilidade, o HDFS faz a separação entre dados e metadados, organizando-os de forma distinta. A composição do sistema de arquivos é dividida em dois tipos de nós. Sendo o *NameNode* (NN) um único nó mestre, é de sua responsabilidade mapear todos os arquivos, gerir os metadados do HDFS e realizar a distribuição dos blocos dos arquivos. Os *DataNodes* (DNs) (pode haver mais de um) por sua vez, realizam o armazenamento propriamente dito dos arquivos em formato de blocos de mesmo tamanho. O HDFS implementa em sua arquitetura os seguintes mecanismos de tolerância a falhas: mensagens *Heartbeat*, replicação de blocos e *CheckPoint and Recovery* [Foundation 2016].

O HDFS utiliza-se de mensagens *Heartbeat* para manter o NN ciente do funcionamento dos DNs. Periodicamente, os DNs ativos enviam avisos ao NN a fim de atualizá-lo sobre seu estado. Caso o NN deixe de receber esses avisos de um DN por um determinado período de tempo, o NN o define como falho (*dead*), deixando de enviar instruções ao mesmo.

Com o objetivo de garantir a consistência dos dados, o HDFS realiza a replicação de blocos através do *Replication Factor* - (RF). Com isso, na possível falha de algum nó, os dados perdidos podem ser recuperados a partir dos blocos replicados em outros nós. Esse mecanismo visa assegurar que aplicações não tenham sua execução comprometida.

O *Checkpoint and Recovery* (CR) é considerada uma técnica de tolerância a falhas em que, tem-se por proposta, recuperar o sistema a um estado anterior à uma falha. Consiste em uma técnica reativa que é explorada em diversos sistemas de arquivos distribuídos que prezem pela confiabilidade dos dados e pelo bom andamento das execuções. No HDFS, periodicamente o *namespace* é replicado em um arquivo chamado `FSImage` e armazenado no sistema de arquivos do NN. Para evitar o custo de processamento necessário para reescrever o arquivo, um *log* de edições, denominado `EditLog`, também é mantido entre os arquivos do NN. Quando o processo de *checkpoint* inicia, um *merge* entre o `FSImage` e o `EditLog` é realizado, mantendo assim a cópia do *namespace* do HDFS atualizada.

3. Metodologia

A metodologia aplicada na fase de experimentação consistiu, basicamente, na execução de um *benchmark* disponibilizado pelo framework Hadoop ao mesmo tempo em que falhas foram forçadas dentro do HDFS. O *benchmark* utilizado foi o `TestDFIO`, cujo comportamento caracteristicamente *I/O bound* permite analisar as consequências de uma falha para o HDFS.

Antes da execução de cada teste, a estrutura do Hadoop era composta por 5 DNs e 1 NN, no modo pseudo-distribuído. A cada execução, o processo responsável por gerenciar um dos DNs era interrompido (através do comando `kill` do Linux) após estar,

aproximadamente, na metade do seu tempo de execução total. A ausência de mensagens *HeartBeat* por parte do DN falho, fazia com que o Hadoop esperasse um tempo X (em milissegundos) até definir o DN como falho. O processo então resultava na perda dos dados contidos no DN e na ativação dos mecanismos de tolerância a falhas do HDFS. O tempo X é calculado pela fórmula: $2*heartbeat.recheck.interval+10*heartbeat.interval = X$. Tanto o $heartbeat.recheck.interval$, quanto o $heartbeat.interval$ são configurações base do HDFS e têm valor *default* 300000 e 3, respectivamente. Porém, ambos foram ajustados para os valores 2000 e 2 para que um DN fosse marcado como falho após 20 segundos sem enviar mensagens *Heartbeat* ao NN.

Com a falha inserida, o efeito dos mecanismos de tolerância a falhas testados atuam de forma distinta no desempenho dos testes. O *checkpoint* acontece de forma periódica em um tempo previamente definido nos arquivos de configuração do HDFS. Logo, um intervalo de tempo menor implica em uma quantidade maior de *checkpoints*, resultando em um aumento no tempo de execução. O fator de replicação dos blocos também é estipulado previamente nos arquivos e é colocado em prática em paralelo às escritas dos arquivos no HDFS. Após a interrupção de um DN, o HDFS começa a replicar os blocos que não respeitam o fator de replicação estipulado previamente. Destinando recursos para realizar replicações extras, o tempo de execução da aplicação tende a aumentar, já que esse processo requer um uso extra de processamento pelo *framework*.

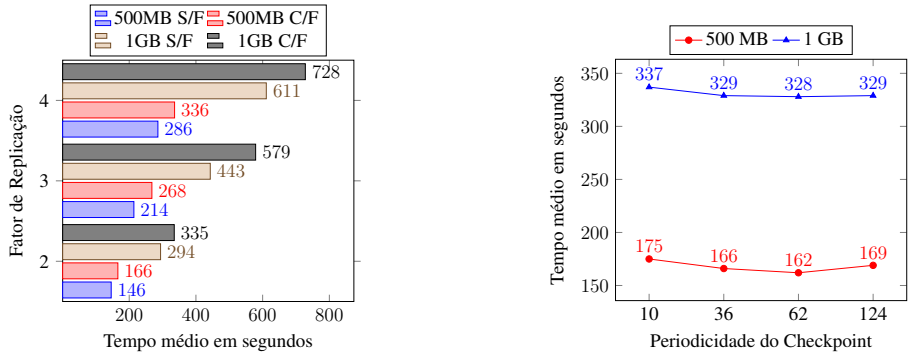
Para analisar separadamente o impacto causado pelos mecanismos de replicação e *checkpoint* do HDFS, foram elaborados dois cenários de testes. No primeiro, a periodicidade dos *checkpoints* foi fixada em 10 segundos e o fator de replicação sofreu acréscimos variando entre 2, 3 e 4. No segundo cenário de testes, o fator de replicação foi fixado em 2 e a periodicidade dos *checkpoints* variou entre 10, 36, 62 e 124 (intervalo em segundos entre um *checkpoint* e outro). Em ambos os cenários de testes, foram realizadas interrupções de um DN durante a execução do *benchmark*, simulando-se falhas permanentes.

Os experimentos foram executados em uma máquina do Laboratório de Sistemas de Computação da UFSM, com a seguinte configuração: disco de 750 GB, processador Intel Core 2 Quad 2.33GHz, duas memórias DIMM DDR3 Síncrono 1066 MHz de 2 GB, rodando sistema operacional Ubuntu 17.04, Kernel linux 4.10.0-42-generic.

4. Experimentos e Resultados

Embora o principal mecanismo de tolerância a falhas do HDFS seja a replicação de blocos, o objetivo deste trabalho é observar, de modo experimental, as consequências da alteração da periodicidade do *checkpoint* na execução das aplicações, além da variação do fator de replicação. Em ambos os cenários de testes descritos na seção anterior, foram realizadas 20 execuções do `TestDFSIO` em modo de escrita para cada configuração. Foram feitos testes com escritas de 10 arquivos de 500MB e 1GB, visando observar o comportamento das execuções. O tempo de execução foi coletado via código e, para a criação dos gráficos da Figura 1, foi utilizada a média aritmética das execuções.

O gráfico da Figura 1(a) exibe as médias dos tempos de execução referentes aos testes dos *benchmarks* com fator de replicação variável. O gráfico apresenta os valores de testes realizados com e sem a indução de falhas. Já o gráfico da Figura 1(b) representa os testes com *checkpoint* variável. Este gráfico mostra apenas medições de testes com indução de falhas.



(a) Fator de Replicação variável (b) Checkpoint variável

Figura 1. Gráficos com as medições de tempo.

Observando-se o gráfico da Figura 1(a), percebe-se que existe um aumento no tempo médio de execução dos testes com falhas em comparação com as execuções sem falhas. Esse aumento pode ter sido causado pela nova replicação dos blocos faltantes que pertenciam ao DN interrompido. Ainda, a diferença de tempo de execução entre os testes com falhas e sem falhas cresce conforme é aumentado o fator de replicação dos blocos no HDFS, já que demanda mais tempo para replicar maiores quantidades de blocos. No gráfico da Figura 1(b), observa-se que o aumento na periodicidade dos *checkpoints* gera uma pequena redução no tempo de execução das aplicações. Isto se deve ao fato de que intervalos maiores de tempo entre os *checkpoints* geram menos *merges* entre os arquivos `FSImage` e `EditLog`.

5. Considerações Finais

A definição estática de atributos para os mecanismos de tolerância a falhas implementados pelo HDFS é um grande desafio, já que escolhas inapropriadas podem gerar problemas de confiabilidade e desempenho. Além disso, a especificação estática de um período ideal é uma tarefa complexa, pois eventuais mudanças necessitam da interrupção do sistema.

Este trabalho apresentou os principais mecanismos de tolerância a falhas implementados pelo framework, avaliando, de forma experimental, o comportamento dos métodos de replicação de blocos e do checkpoint com diferentes variações. Através de dois cenários de testes, foi possível observar o impacto dos métodos no tempo de execução do *benchmark* `TestDFSIO`. Como trabalhos futuros, serão realizados testes com outros *benchmarks*, além de estender a experimentação para execução do Hadoop em modo totalmente distribuído [Foundation 2016].

Referências

Dean, J. and Ghemawat, S. (2008). *MapReduce: simplified data processing on large clusters*. Commun. ACM, 51(1):107–113.

Foundation, A. (2014). *What is the Apache Hadoop?* http://hadoop.apache.org/index.pdf.

Foundation, A. (2016). *Hadoop: Cluster Setup*. https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-common/ClusterSetup.html.

White, T. (2012). *Hadoop: The definitive guide*. "O'Reilly Media, Inc.".