Validação do Mecanismo de Envio de Mensagens Heartbeat do HDFS Através de Programação Orientada a Aspectos

Iago C. Corrêa[1], **Patrícia Pitthan Barcelos**[1]

[1]Laboratório de Sistemas de Computação (LSC)
Universidade Federal de Santa Maria – UFSM

{icorrea,pitthan}@inf.ufsm.br

Resumo. *O Apache Hadoop implementa diversos mecanismos de tolerância a falhas em seus processos. Entretanto, poucos trabalhos se destinam a validar esses mecanismos, a fim de observar seu comportamento. Este trabalho apresenta uma validação do mecanismo de envio de mensagens heartbeat, implementado pelo sistema de arquivos distribuído do Hadoop, através da introdução deliberada de falhas utilizando Programação Orientada a Aspectos.*

1. Introdução

O Apache Hadoop [Foundation 2014] é um *framework open source* para o processamento distribuído de grandes volumes de dados. A arquitetura do Hadoop utilizada neste trabalho (versão 2.7.7) é composta por diversos módulos, dentre os quais destacam-se: o sistema de arquivos distribuído (HDFS) e um *framework* para manutenção do ambiente e da aplicação (YARN).

O Hadoop implementa diversos mecanismos de tolerância a falhas visando garantir confiabilidade e disponibilidade tanto no armazenamento, quanto no processamento massivo de dados. Entretanto, poucos trabalhos se destinam a validá-los, buscando observar se os mesmos atendem suas especificações. Este trabalho visa validar o mecanismo de envio de mensagens *heartbeat* do HDFS, através da introdução deliberada de falhas utilizando programação orientada a aspectos (Aspect Oriented Programming - AOP).

O trabalho está organizado da seguinte maneira: a Seção 2 descreve como o Hadoop implementa tolerância a falhas. A Seção 3 apresenta a programação orientada a aspectos como alternativa viável para injeção de falhas por instrumentação de código. A Seção 4 descreve o modelo de falhas aplicado na validação. Por fim, a Seção 5 aponta considerações finais e trabalhos futuros.

2. Tolerância a Falhas no Apache Hadoop

O Apache Hadoop, através de seu sistema de arquivos distribuído, implementa mecanismos de tolerância a falhas em seus processos, para garantir que seus serviços sejam mantidos, mesmo em caso de falhas. Dentre os mecanismos implementados, destacam-se: a replicação de blocos, o estabelecimento de *checkpoints* e o envio de mensagens *heartbeat*.

Este trabalho enfoca o mecanismo de envio de mensagens *heartbeat*. Trata-se de um mecanismo centrado na comunicação entre os nós do HDFS e consiste em avisos periódicos que são enviados pelos DNs, nós escravos da arquitetura, ao NN, nó mestre, visando notificá-lo de seu estado atual. A periodicidade dessas mensagens é definida por

uma variável estática nos arquivos de configuração do Hadoop, cujo intervalo padrão é 3 segundos. Sendo assim, estas mensagens auxiliam o NN a identificar a operacionalidade dos nós escravos. Sempre que um DN deixa de enviar uma mensagem *heartbeat* por um tempo pré-estabelecido (por padrão 10 minutos), o NN lhe atribui o estado $inoperante$ e deixa de enviar novas instruções ao mesmo.

3. Validação por Instrumentação de Código

Uma das formas de se mensurar a eficácia de mecanismos de tolerância a falhas é através da introdução deliberada de falhas. Esta técnica, denominada injeção de falhas, permite introduzir, de forma controlada, cenários anômalos em sistemas alvo visando observar seus efeitos e analisar se esses mecanismos atendem às suas especificações [Hsueh et al. 1997]. A instrumentação de código é uma técnica que permite a injeção de falhas por *software* em sistemas computacionais. Com esta técnica, novas instruções são adicionadas no sistema alvo para serem executadas em situações específicas, sendo acionadas ainda em tempo de execução.

Entretanto, a inserção de código em uma aplicação pode implicar em intrusividade, comprometendo o contexto da aplicação alvo. Nesse sentido, utilizar a AOP para instrumentar código torna-se uma alternativa viável [Kiczales et al. 1997]. Este paradigma tem por objetivo promover a separação de interesses através de estruturas modulares denominadas $aspectos$. Sendo assim, o corpo destas estruturas apresenta construções chamadas *advices*, as quais contêm o código a ser instrumentado antes, depois ou no lugar de momentos específicos identificados em *joinpoints* e interceptados por *pointcuts*. Os *joinpoints* podem ser chamadas a métodos específicos ou instanciamento de novos objetos, por exemplo.

A validação do mecanismo de envio de mensagens *heartbeat* apresentada por este trabalho ocorreu por meio de um modelo de falhas implementado utilizando o AspectJ, uma extensão da linguagem Java para AOP. O modelo de falhas possui aspectos que atuam diretamente no método responsável por disparar as mensagens *heartbeat*, alinhando o funcionamento deste método a comportamentos pré-estabelecidos.

4. Modelo de falhas

O modelo de falhas aplicado na validação do mecanismo de envio de mensagens *heartbeat* busca emular falhas de omissão, *crash* e temporização [Corrêa and Barcelos 2018]. Entretanto, este trabalho enfoca apenas as falhas de omissão e falhas de *crash*, descritas nas seções 4.1 e 4.2, respectivamente. O alvo de ambas as falhas é o nodo, cuja atuação emula um DN com comportamento diferente do especificado, uma vez que deixa de enviar as mensagens *heartbeat*. As falhas de omissão apresentam variações na duração, podendo ser transientes ou intermitentes.

Sob o ponto de vista de implementação, cada aspecto responsável por induzir uma falha possui a declaração de um *advice*, que contém um conjunto de instruções a serem executadas antes do método do DN responsável pelo envio das mensagens *heartbeat*. As instruções de cada *advice* variam conforme o efeito desejado. Ainda, um *pointcut* foi declarado no corpo dos aspectos para interceptar o acionamento do método de envio das mensagens, criando o chaveamento necessário para executar os *advices* antes do método padrão.

O modelo de falhas permite que as falhas sejam introduzidas em um DN específico. Sendo assim, as falhas passam a atuar a partir do momento em que os serviços do HDFS são acionados. Como os *advices* das falhas são executados antes do método `SendHeartbeat()`, é necessário verificar, mediante uma variável de controle, se o DN que está executando o método de emissão de mensagens *heartbeat* é de fato o DN alvo. A variável de controle possui o endereço do DN alvo da validação. Esta verificação é realizada no início da execução de cada *advice*. Caso o DN em execução não corresponda ao alvo, o método original é retomado através da instrução `proceed()`.

4.1. Implementação de falhas de omissão

As falhas de omissão apresentam-se de forma transiente ou intermitente. A falha transiente busca emular um desvio da especificação, omitindo aleatoriamente a entrega de uma dentre várias mensagens *heartbeat*. Ao iniciar a execução do *advice* responsável por esta falha, são realizadas confirmações de que a falha ainda não foi introduzida e de que o DN em execução é o alvo. A seguir, ocorre a introdução da falha através do retorno de `null` ao invés de uma instância do objeto `HeartbeatResponse`, a qual é esperada pelo método original e indica que a mensagem foi entregue. Omissões de mensagem *heartbeat* geram `Exceptions` nos serviços do Hadoop, que podem ser observadas através dos *logs* do DN alvo, conforme exibe a Figura 1.

```
2018-12-17 10:33:22,039 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: OMISSAO TRANSIENTE:  *** Datanode 127.0.0.1:50010 bloqueado. ***
2018-12-17 10:33:22,040 ERROR org.apache.hadoop.hdfs.server.datanode.DataNode: Exception in BPOfferService for Block pool BP-2123439418-192.168.25.6-1545049933743
(Datanode Uuid 94425eaa-9c8b-4094-8660-d9113eef94b5) service to localhost/127.0.0.1:9000
java.lang.NullPointerException
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.offerService(BPServiceActor.java:511)
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.run(BPServiceActor.java:659)
        at java.lang.Thread.run(Thread.java:748)
2018-12-17 10:33:27,072 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: For namenode localhost/127.0.0.1:9000 using BLOCKREPORT_INTERVAL of 21600000msec
CACHEREPORT_INTERVAL of 10000msec Initial delay: 0msec; heartBeatInterval=3000
```

Figura 1. Trecho de *log* exibindo falha de omissão transiente.

A omissão intermitente ocorre quando diversas mensagens de um conjunto não são enviadas pelo DN alvo. Assim, nas instruções do *advice*, após confirmar se o DN em execução é o alvo, executa um teste probabilístico que gera um valor aleatório (entre 0 e 3) que sinaliza a ocorrência ou não da omissão da mensagem. Se o número gerado for 0, é retornado `null` no lugar do objeto `HeartbeatResponse`, simulando a omissão da mensagem. Para qualquer outro valor gerado, ocorre a chamada normal do método `sendHeartbeat()`, que irá executar o envio da mensagem *heartbeat* esperada. A Figura 2 exibe o trecho de *log* de um DN que sofreu omissão intermitente de mensagens.

```
2018-12-17 10:17:34,026 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: OMISSAO INTERMITENTE: *** Heartbeat de 127.0.0.1:50010 bloqueado. ***
2018-12-17 10:17:34,027 ERROR org.apache.hadoop.hdfs.server.datanode.DataNode: Exception in BPOfferService for Block pool BP-1515666224-192.168.25.6-1545048971786
(Datanode Uuid 38206dfa-4e81-4dcb-87ba-31071cd9264c) service to localhost/127.0.0.1:9000
java.lang.NullPointerException
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.offerService(BPServiceActor.java:511)
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.run(BPServiceActor.java:659)
        at java.lang.Thread.run(Thread.java:748)
2018-12-17 10:17:39,027 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: For namenode localhost/127.0.0.1:9000 using BLOCKREPORT_INTERVAL of 21600000msec
CACHEREPORT_INTERVAL of 10000msec Initial delay: 0msec; heartBeatInterval=3000
2018-12-17 10:17:39,028 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: OMISSAO INTERMITENTE: *** Heartbeat de 127.0.0.1:50010 bloqueado. ***
2018-12-17 10:17:39,028 ERROR org.apache.hadoop.hdfs.server.datanode.DataNode: Exception in BPOfferService for Block pool BP-1515666224-192.168.25.6-1545048971786
(Datanode Uuid 38206dfa-4e81-4dcb-87ba-31071cd9264c) service to localhost/127.0.0.1:9000
java.lang.NullPointerException
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.offerService(BPServiceActor.java:511)
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.run(BPServiceActor.java:659)
        at java.lang.Thread.run(Thread.java:748)
2018-12-17 10:17:44,028 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: For namenode localhost/127.0.0.1:9000 using BLOCKREPORT_INTERVAL of 21600000msec
CACHEREPORT_INTERVAL of 10000msec Initial delay: 0msec; heartBeatInterval=3000
2018-12-17 10:17:50,029 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: OMISSAO INTERMITENTE: *** Heartbeat de 127.0.0.1:50010 bloqueado. ***
```

Figura 2. Trecho de *log* exibindo falha de omissão intermitente.

4.2. Implementação de falhas de *crash*

A falha de *crash* visa emular um DN que não consegue entregar mensagens *heartbeat*. Neste caso, não é necessário controlar a inserção da falha, já que sempre ocorrerá a supressão das mensagens. Logo, no *advice* desta falha, após a confirmação de que o DN em execução é o alvo, a omissão da mensagem sempre irá ocorrer. A Figura 3 exibe o trecho de *log* que indica a falha de *crash* de nodo atuando nos serviços do HDFS.

```
2018-12-17 09:41:39,665 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: FALHA DE CRASH DE NODO:  *** Datanode 127.0.0.1:50010 bloqueado. ***
2018-12-17 09:41:39,667 ERROR org.apache.hadoop.hdfs.server.datanode.DataNode: Exception in BPOfferService for Block pool BP-1316107075-192.168.25.6-1545046823363
(Datanode Uuid 8d7e3138-bc4e-413b-9ffc-4b2bc2e03e84) service to localhost/127.0.0.1:9000
java.lang.NullPointerException
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.offerService(BPServiceActor.java:511)
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.run(BPServiceActor.java:659)
        at java.lang.Thread.run(Thread.java:748)
2018-12-17 09:41:44,668 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: For namenode localhost/127.0.0.1:9000 using BLOCKREPORT_INTERVAL of 21600000msec
CACHEREPORT_INTERVAL of 10000msec Initial delay: 0msec; heartBeatInterval=3000
2018-12-17 09:41:44,669 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: FALHA DE CRASH DE NODO:  *** Datanode 127.0.0.1:50010 bloqueado. ***
2018-12-17 09:41:44,669 ERROR org.apache.hadoop.hdfs.server.datanode.DataNode: Exception in BPOfferService for Block pool BP-1316107075-192.168.25.6-1545046823363
(Datanode Uuid 8d7e3138-bc4e-413b-9ffc-4b2bc2e03e84) service to localhost/127.0.0.1:9000
java.lang.NullPointerException
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.offerService(BPServiceActor.java:511)
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.run(BPServiceActor.java:659)
        at java.lang.Thread.run(Thread.java:748)
2018-12-17 09:41:49,670 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: For namenode localhost/127.0.0.1:9000 using BLOCKREPORT_INTERVAL of 21600000msec
CACHEREPORT_INTERVAL of 10000msec Initial delay: 0msec; heartBeatInterval=3000
2018-12-17 09:41:49,670 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: FALHA DE CRASH DE NODO:  *** Datanode 127.0.0.1:50010 bloqueado. ***
2018-12-17 09:41:49,671 ERROR org.apache.hadoop.hdfs.server.datanode.DataNode: Exception in BPOfferService for Block pool BP-1316107075-192.168.25.6-1545046823363
(Datanode Uuid 8d7e3138-bc4e-413b-9ffc-4b2bc2e03e84) service to localhost/127.0.0.1:9000
java.lang.NullPointerException
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.offerService(BPServiceActor.java:511)
        at org.apache.hadoop.hdfs.server.datanode.BPServiceActor.run(BPServiceActor.java:659)
        at java.lang.Thread.run(Thread.java:748)
2018-12-17 09:41:54,672 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: For namenode localhost/127.0.0.1:9000 using BLOCKREPORT_INTERVAL of 21600000msec
CACHEREPORT_INTERVAL of 10000msec Initial delay: 0msec; heartBeatInterval=3000
```

Figura 3. Trecho de *log* exibindo a falha de *crash* de nodo.

Com a inserção desta falha espera-se que, após um determinado tempo sem entregar mensagens *heartbeat*, o DN alvo seja marcado como inoperante pelo NN.

5. Considerações Finais

O trabalho apresentou o uso de instrumentação de código, implementada com programação orientada a aspectos, para a validação de um mecanismo de tolerância a falhas do HDFS: o envio de mensagens *heartbeat*. Introduzir falhas neste mecanismo possibilita observar se sua especificação está sendo atendida, além de verificar seu comportamento mediante falhas. Como trabalhos futuros, pretende-se analisar a possibilidade de transformar essa implementação em uma ferramenta que possa ser incorporada no Apache Hadoop. Adicionalmente, tenciona-se utilizar a mesma abordagem para validar mecanismos de tolerância a falhas de outras ferramentas de processamento distribuído.

Referências

Corrêa, I. and Barcelos, P. P. (2018). Validação dos mecanismos de tolerância a falhas do hdfs através de programação orientada a aspectos. Monografia (Bacharel em Sistemas de Informação), Universidade Federal de Santa Maria, Santa Maria, RS, Brasil.

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

Hsueh, M.-C., Tsai, T. K., and Iyer, R. K. (1997). Fault injection techniques and tools. *Computer*, 30(4):75–82.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., and Irwin, J. (1997). Aspect-oriented programming. In *ECOOP'97 — Object-Oriented Programming*, pages 220–242, Berlin, Heidelberg. Springer Berlin Heidelberg.