Apache Accumulo

Dulcinéa Anjos
Helton Dória Costa

Sumário

Introdução

Em novembro de 2006, a empresa Google causou um grande movimento no mundo Big Data ao publicar um artigo sobre o Bigtable, um sistema distribuído de armazenamento de dados estruturados desenvolvido internamente e capaz de gerenciar petabytes de dados utilizando milhares de servidores comuns[GGL01]. Como várias organizações possuíam necessidades similares às da Google no que tange o gerenciamento grandes volumes de dados, logo surgiram várias tentativas de reproduzir as características descritas no artigo e assim surgiram vários gerenciadores de dados NoSQL hoje famosos, como o Cassandra e o HBase.

Nessa mesma época, a Agência de Segurança Nacional americana, na sigla em inglês - NSA, identificou que o Bigtable possuía as características que ela buscava para lidar com o altamente variado e gigantesco volume crescente de dados que ela precisava analisar para cumprir sua missão. Porém, dada a natureza do trabalho executado pela agência demandar um rigoroso controle de acesso às informações, era necessário encontrar uma implementação que possuísse também um forte sistema de segurança. Como eles não foram capazes de encontrar tal implementação, decidiram criar sua própria versão do Bigtable, acrescentando a ela uma elevada capacidade de controle de acesso aos dados. Nascia assim, em 2008, o projeto Cloudbase que depois, em 2011, viria a ser rebatizado como Accumulo e doado para a *Apache Software Foundation* [WIR01].

O Accumulo é um gerenciador de dados não relacional, classificado como NoSQL do tipo colunar. Dentro do ranking de gerenciadores de bancos de dados mantido pelo site DB-Engines, ele ocupa a 58ª posição no quadro geral e 3ª posição dentro do subconjunto dos banco ditos colunares[DBE01]. Apesar de ser classificado como um banco NoSQL e de não ser relacional, a comunidade de código aberto que vem surgindo em torno dele está descobrindo cada vez mais usos possíveis para a sua tecnologia, aproximando-o cada vez mais de torná-lo um banco de propósito geral.

Apesar de pouco conhecido e do ranking relativamente baixo, o Accumulo é um banco que impressiona por seus números e por seus recursos. Segundo o site da empresa americana Sqrrl, empresa formada por ex-funcionários da NSA que participaram da criação do Accumulo, em testes realizados por organizações independentes, o Accumulo seria capaz de suportar taxas de 100000 escritas e 10000 leituras por segundo, por nó.

Se esses números por si só já não fossem impressionantes e o colocassem no grupo dos gerenciadores de dados de alto desempenho, testes realizados pela universidade Carnegie Mellon mostram o Accumulo operando em um cluster com 1200 nós, gerenciando um grafo com 70 trilhões de arestas e 1 petabyte de tamanho e demonstrando desempenho linear durante o aumento do grafo entre 1 trilhão e 70 trilhões de arestas [CMU01]. Outro teste, realizado pelo MIT Lincoln Laboratory, que estuda aplicações de gerenciadores de dados distribuídos em supercomputadores, apresenta o Accumulo operando em um cluster com 270 nós e alcançando picos de ingestão de mais 100 milhões de inserts por segundo[MIT01]. Estes testes requereram um cuidadoso planejamentos e vários ajustes para garantir esse desempenho impressionante, mas demonstram que ele é um gerenciador rápido, escalável e capaz de gerir bases de dados de grande porte.

Detalhamento Teórico

Apesar das semelhanças, também existem muitas diferenças. Das suas características mais notáveis, citamos as duas que ganharam maior destaque: a capacidade de guardar dados de permissão de acesso em cada célula do banco e a capacidade de executar processamentos de dados no próprio servidor, reduzindo o esforço das aplicações clientes no consumo dos dados.

Aplicações

O Accumulo se apresenta como um gerenciador de propósito geral, mas ele é particularmente recomendável para aplicações que demandem alto desempenho, capacidade de gerenciar volumes massivos de dados ou em que o requisito de segurança tenha primazia. Organizações governamentais, da área de saúde e bancárias são exemplo de setores que podem se beneficiar do seu uso.

Apesar de ser um banco colunar, que normalmente não recomendado para qualquer situação, a comunidade em torno do Accumulo tem demonstrado criatividade e o adaptado para trabalhar em diversas situações. Como citado anteriormente, a Universidade Carnegie Mellon conseguiu demonstrar a sua aplicabilidade para lidar com grafos complexos e o MIT o está explorando em ambientes de supercomputação. Mas além desses expoentes, fomos capazes de identificar mais algumas organizações que o estão utilizando, seja como parte da sua infraestrutura ou como base para produtos comercias.

Esse é o caso, por exemplo, do Sqrrl Enterprise, ferramenta de analise de dados produzida pela empresa norte-americana Sqrrl. Nesta ferramenta, o Accumulo atual como motor central para o gerenciamento dos dados. Outro exemplo é a suíte GeoMesa[GEO01], que é um conjunto de ferramentas de código aberto que permite análises geoespaciais em grande escala na nuvem e em sistemas de computação distribuída.

A empresa de informações financeira Bloomberg divulgou que usa o Accumulo como parte da sua infraestrutura. Ele é empregado em várias aplicações na Bloomberg Vault, seja como gerenciador de de dados ou até mesmo como abstração para o armazenamento de arquivos armazenado em HDFS. Recentemente ela também publicou um artigo divulgando o Presto, um conector desenvolvido internamente que permite executar consultas SQL ANSI diretamente sobre o Accumulo[BLO01].

Além disso, empresas como Hortonworks e Cloudera já dispõem de suporte ao Apache Accumulo em seus produtos (apesar de ainda não vir instalado por padrão) e outras como a Pivotal o estão estudando para possível inclusão futura em suas distribuições. Enfim, apesar de ainda pouco popular, ele tem encontrado espaço em vários seguimentos e aos poucos vai ampliando sua base de usuários.

Arquitetura

O Accumulo trabalha como um gerenciador de dados distribuídos dentro da estrutura maior que é o Hadoop e,sendo assim, depende de várias ferramentas externas a ele para funcionar. Ele opera sobre

o YARN e persiste seus dados no HDFS. A comunicação entre os nós é gerenciada através do ZooKeeper e ele se vale do Apache Thrift para garantir sua conectividade com aplicações que não sejam feitas em Java.

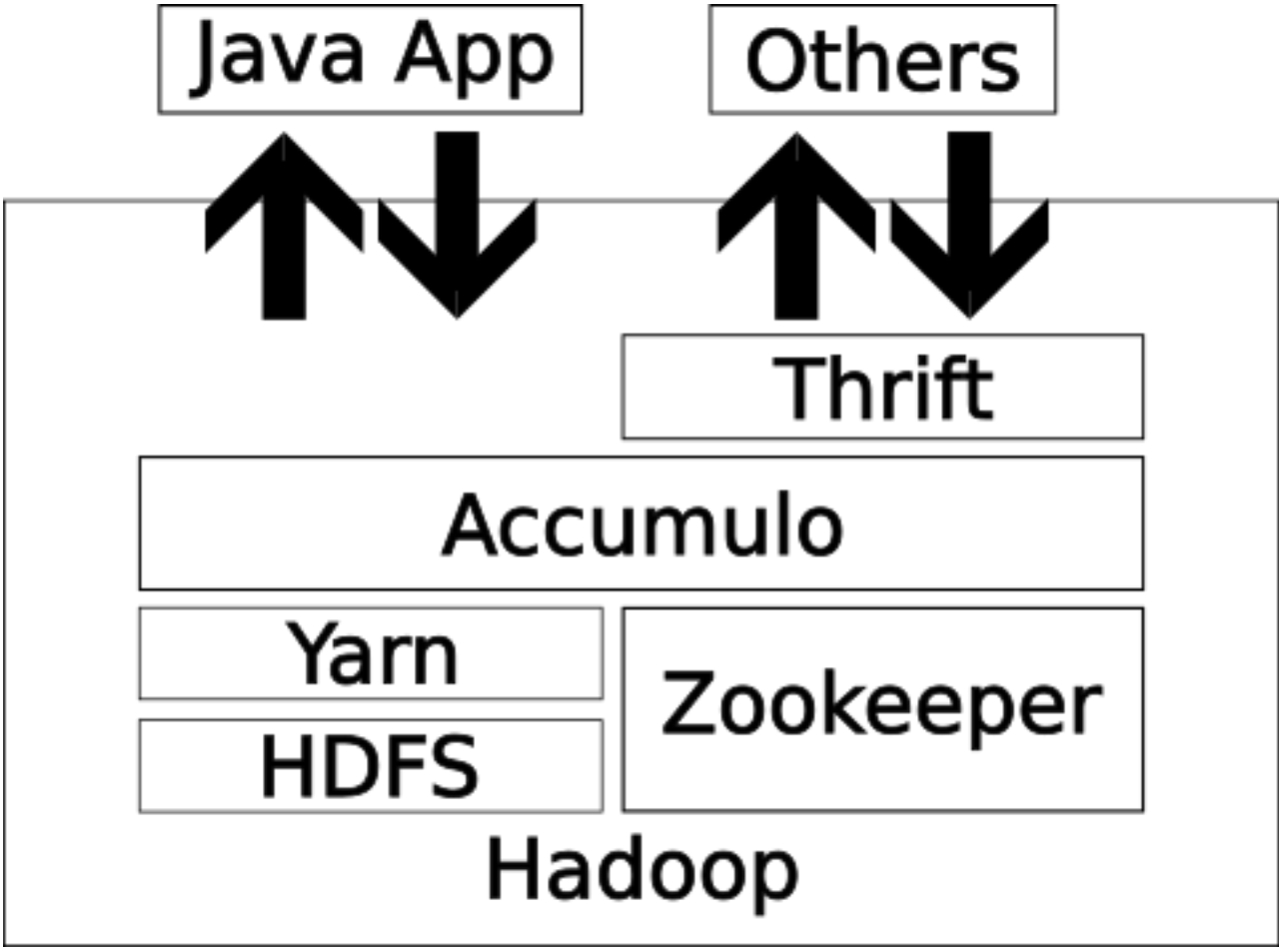

Figura 1: Arquitetura geral do Apache Accumlo

Componentes internos

Internamente, o Accumulo utiliza vários componentes e estruturas para manter consistentes e poder coordenar o trabalho de diversas instâncias. Dentre os componentes mais notáveis citamos: Master, TabletServer e o Garbage Collector.

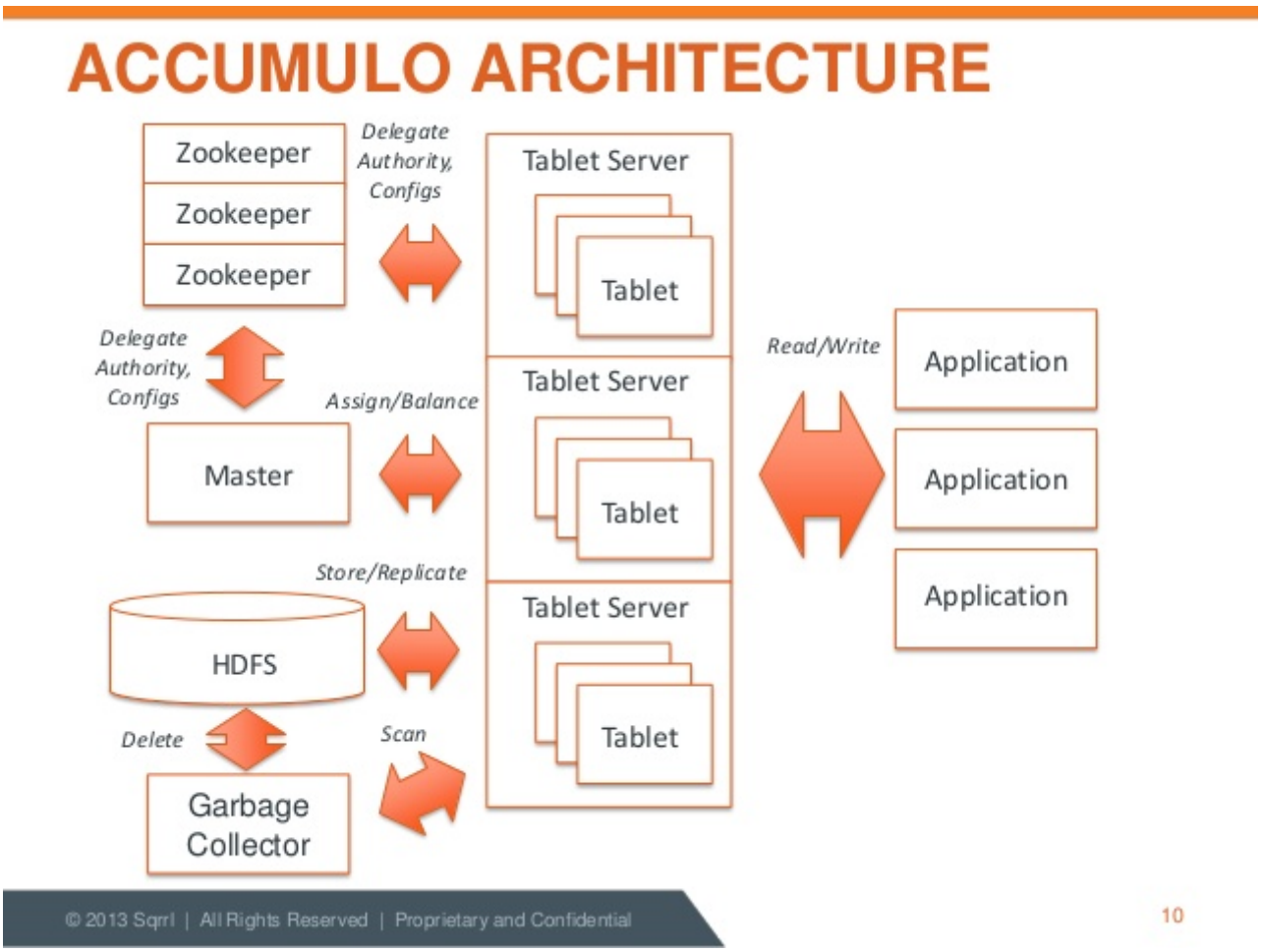

Figura 2: Visão ampliada da arquitetura do Accumulo

Tabelas, células, tablets e TabletServers

Os dados dentro do Accumulo são organizados em uma coleção ordenada chamada de tabela. Uma tabela é composta por inúmeras células, que é um simples par chave-valor. A chave é composta por uma quíntupla que é formada por um identificador de linha, um identificador de família, um qualificador, um identificador de visibilidade(permissões de segurança) e um timestamp. Todos os elementos da célula são representados como arrays de bytes, com exceção do timestamp, que é representado como um long.

Key					Value
row ID	Column			Timestamp	
	Family	Qualifier	Visibility		

Figura 3: Estrutura de uma célula

Uma tabela irá subdividir seus dados em blocos menores e contíguos que são chamados de tablets e esses serão gerenciados pelo componente chamado TabletServer.

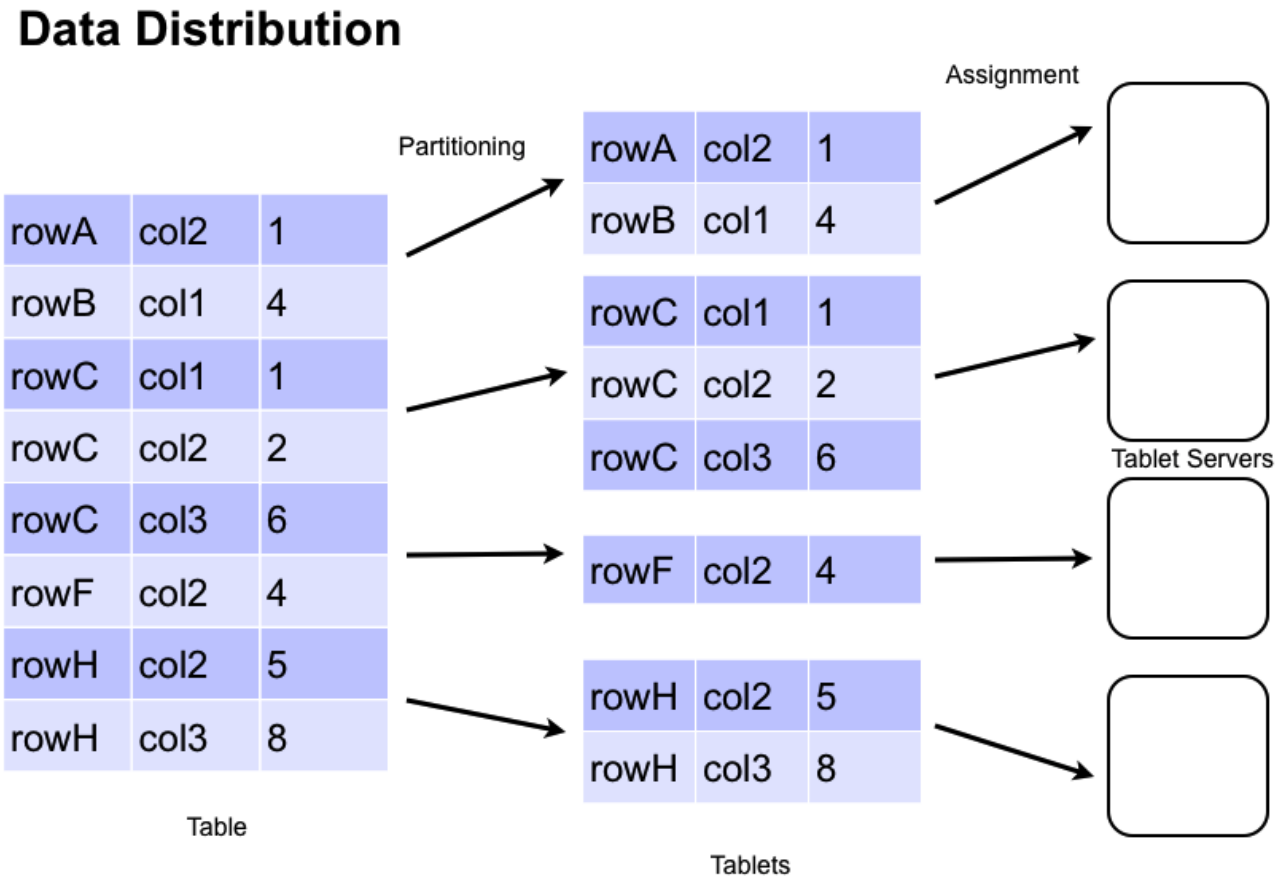

Figura 4: Particionamento de uma tabela em tablets

TabletServer

Um TabletServer é o componente que fica encarregado de gerenciar um subconjunto de tablets. Isso inclui receber requisições de escrita de clientes, gravar as operações no log, fazer a triagem de novos pares de chave-valor na memória, periodicamente descarregar grupos de pares chave-valor para novos arquivos no HDFS, responder a requisições de leitura de clientes, manter um índice ordenado por merge-sort com todas as chaves e valores de todos os arquivos que ele criou e dos dados recentes que estão na memória. Eles também se encarregam da recuperação de tablets que estavam anteriormente em um servidor que falhou, reaplicando para os tablets quaisquer gravações encontradas no log.

Master

O Master é o componente responsável por gerenciar o trabalho no cluster. Ele lida com a criação de tabelas, alteração de dados e pedidos de eliminação vindos de clientes. Ele também detecta e responde a falhas dos TabletServers, bem como gerencia a distribuição dos tablets dentro do cluster. Ele tenta equilibrar a carga entre os TabletServers atribuindo tablets com cuidado e instruindo os TabletServers para descarregar tablets quando necessário. O Master garante que todos

os tablets sejam atribuídos a algum TabletServer. A ele também cabe a coordenação da inicialização, desligamento e recuperação de mudanças nos write-ahead logs(WAL) quando os TabletServers falham.

Vários mestres podem ser executados simultaneamente num mesmo cluster, porém, apenas um deles poderá comandar por vez e os demais terão que atuar como réplicas do principal para aumentar a tolerância a falhas do cluster em caso de problemas com o Master.

Garbage Collector

Um cluster Accumulo compartilhar arquivos entre seus nós através do HDFS. O Garbage Collector fica então encarregado de periodicamente identificar arquivos que não sejam mais necessários e excluí-los. Eles também são responsáveis por remover linhas que marcadas para exclusão, ajudando assim na compressão dos arquivos num procedimento chamado Major compaction. Múltiplos coletores de lixo podem ser executados ao mesmo tempo. Caso isso aconteça, eles irão realizar entre si a eleição de um líder para que haja apenas uma única instância ativa de cada vez.

Cliente

O Accumulo inclui uma biblioteca cliente que deverá estar ligada a todas as aplicações. A biblioteca cliente contém a lógica para encontrar servidores que gerenciam um tablet particular, e se comunicar com TabletServers para escrever e recuperar pares chave-valor

Gerenciamento de dados

Quando uma gravação chega a um TabletServer, ela será primeiro escrita no WAL, e depois será inserida na memória, em uma estrutura ordenada de dados chamada MemTable. Quando a MemTable atinge um determinado tamanho, o TabletServer gravará os pares chave-valor em um arquivo no HDFS chamado Relative Key File (RFile), que é um arquivo indexado de acesso sequencial. Este processo é chamado de Minor compaction porque nele é feita uma redução na redundância dos dados nas chaves, omitindo atributos que sejam idênticos aos da célula anterior. Uma nova MemTable será então criada e o fato da compactação será registrado no WAL.

A fim de gerir o número de arquivos por tablets, periodicamente o TabletServer executará uma compactação chamada Main compaction. Ela busca comprimir arquivos dentro de um tablet combinando um conjunto de RFiles em um único arquivo. Os arquivos anteriores serão eventualmente removidos pelo coletor de lixo. Isso também fornece uma oportunidade para remover permanentemente pares chave-valor marcados para exclusão, omitindo esses pares quando o novo arquivo é criado.

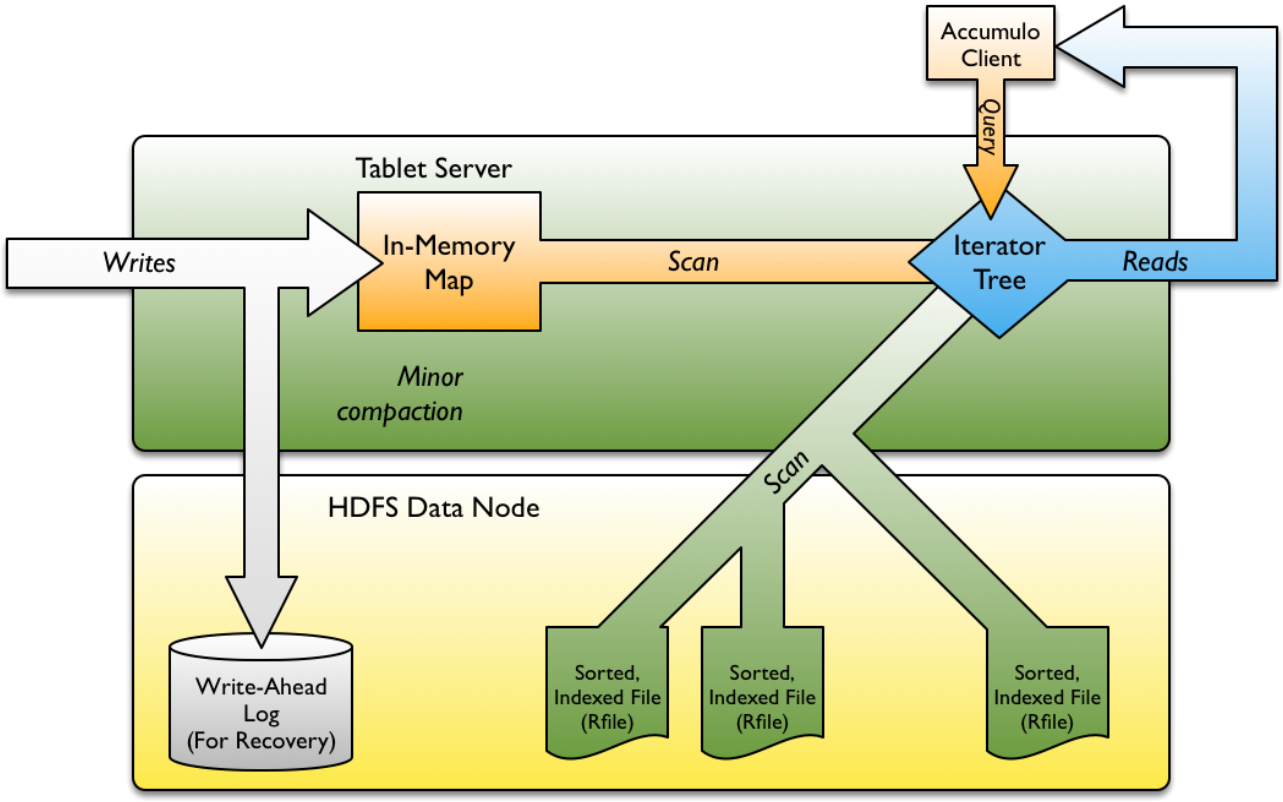

Figura 5: Fluxo de dados em um insert

Particionamento de tabelas

Quando uma tabela é criada, ela tem apenas um tablet. À medida em que ela cresce, ela será subdividida em novos tablets, que provavelmente migrarão para um novo TabletServer. Enquanto ela continuar a crescer, seus tablets continuarão a se dividir e migrar. A decisão de dividir automaticamente um tablet baseia-se no tamanho dele. O limite de tamanho em que um tablet se subdivide é configurável por tabela. Além de divisão automática, um usuário pode adicionar manualmente pontos de divisão em uma tabela para criar novos tablets. Dividir manualmente uma nova tabela pode paralelizar a leitura e escrita, oferecendo melhor desempenho inicial, sem esperar pela divisão automática.

Quando os dados são excluídos de uma tabela, os tablets podem encolher. Ao longo do tempo isto pode conduzir a tablets muito pequenos ou vazios. Para lidar com isso, o Accumulo possui também a capacidade de fundir tablets em tablets maiores.

Segurança

O Accumulo tem a segurança como um dos seus pontos fortes, mas ele não possui um sistema de autenticação próprio, ficando a cargo do administrador definir qual será o sistema de autenticação empregado. Ele possui suporte para trabalhar com serviços que utilizem o protocolo Kerberos e o uso de um sistema desses é altamente recomendado já que ele é considerado seguro e outras partes do Hadoop também poderão utilizá-lo. Caso o administrador não disponha ou não deseje utilizar um sistema Kerberos, ele pode fazer uso de listas ACL simples ou mesmo de chaveiros java.

O controle de acesso aos dados vai depender de duas partes, o controle de visibilidade gravado nas células e as credenciais informadas junto com a requisição. As credenciais necessárias são armazenadas no atributo de visibilidade na forma de tokens que podem também ser combinados em expressões lógicas, permitindo a construção de lógicas de acesso mais refinadas. Se os tokens não forem fornecidos na criação da célula, ela não terá qualquer proteção de acesso.

Particionamento

O particionamento no Accumulo é feito automaticamente, desde que as instancias estejam configuradas para falar com um mesmo quorum ZooKeeper e que a tabela sendo gerenciada atinja o tamanho limite que dispare o particionamento dos dados. O servidor fará também o balanceamento de maneira automática, de modo a tentar distribuir o volume de dados e a carga de acesso de maneira o mais uniforme possível.

Tolerância à falha

Write-Ahead Log

Se um TabletServer falhar, o Master detecta e redistribui automaticamente os tablets que estavam atribuídos a ele. Quaisquer pares chave-valor que estavam na memória no momento da falha serão automaticamente reaplicados a partir do Write-Ahead Log para evitar qualquer perda de dados.

TabletServer escrevem seus WALs diretamente no HDFS para que os registros estejam disponíveis para todos os TabletServer para o caso de uma recuperação ser necessária. Para tornar o processo de recuperação eficiente, as atualizações dentro de um log são agrupadas por tablet. TabletServers podem aplicar rapidamente as mutações dos registros ordenados que são destinados aos tablets que já foram reatribuídos.

FATE

O FATE(Fault-Tolerant Executor) é um framework empregado pelo Accumulo para garantir a consistência de operações que deveriam ser atômicas, mesmo quando algumas delas envolvam múltiplos passos e a atuação de componentes diferentes.

O FATE é dividido basicamente em três partes: uma operação persistente e reproduzível, também conhecida como REPO (repeatable, persisted operation), e uma camada de armazenamento para essas operações e um executor para reproduzi-las. O Accumulo utiliza o ZooKeeper como a camada de armazenamento para o FATE e usa o Master como executor das operações.

A característica mais importante das REPOS é que elas são implementas de modo que elas sejam idempotentes: cada operação deve ser capaz de desfazer ou reaplicar uma de suas execuções que não tenham sido finalizadas devido a alguma falha.

Replicação e Failover

No Accumulo, podemos designar réplicas de para vários componentes, incluindo o master e o garbage collector. Em caso de uma falha, o ZooKeeper detectará que o componente deixou de responder e atribuirá suas responsabilidade para a réplica. Se houver mais uma réplica para o mesmo componente, o ZooKeeper procederá uma eleição para determinar qual delas irá responder como a cópia principal e as demais permanecerão atuando como réplicas.

As réplicas não podem responder requisições ou realizar os trabalhos que o componente sendo replicado realiza enquanto elas não estiverem registradas como cópia principal no ZooKeeper. Sendo assim, a sua função é simplesmente prover um backup para o caso de falhas acontecerem.

No caso da replicação do Master, é aconselhável planejá-la com cuidado porque ela ira impor um esforço de comunicação e sincronização extra sobre o seu servidor. Isso pode reduzir o nível de consistência do seu cluster, porque pode haver um retardo perceptível na reprodução de operações que já haviam ocorrido na cópia principal, mas que ainda não tinham sido reproduzidas na réplica eleita como nova cópia principal.

Logical time e Master authoritative time

No Accumulo, os dados não são alterados depois que eles são inseridos. Eles podem ser apagados, mas as modificações são inseridas como novas versões da célula, com um timestamp diferente. Por padrão, as células dentro dos tablets são mantidas ordenadas em ordem crescente de identificador de linha e ordem decrescente de timestamp, de modo que as versões mais recentes venham primeiro. Sendo assim, se torna imprescindível que os timestamps utilizados sejam diferentes e crescentes.

Para tentar ajudar a garantir o correto funcionamento do timestamp mesmo em ambiente distribuído, o Accumulo possui dois recursos: o logical time e o master authoritative time. O logical é utilizado quando o Accumulo recebe duas inserções na mesma unidade de tempo. Nesses casos, ele incrementará o timestamp a cada insert para que eles continuem crescentes.

O master authoritative time é utilizado para que o mesmo quando os TabletServers estejam espalhados geograficamente, eles mantenham a coerência entre os timestamps atribuídos. Para isso, o Master faz um levantamento dos horários em todos os TabletServers, calcula um valor médio e impõe que esse valor seja seguido.

O Accumulo permite que os usuários informem o timestamp de uma célula, mas nesse caso ele não se responsabiliza pela correção da informação. Caso duas células com mesma chave e timestamp sejam inseridas em uma tabela, o Accumulo se comportará de maneira não determinística e devolverá a primeira que ele conseguir.

Scalable master

A tabela de metadados do Master é guardada tanto no Master quanto no ZooKeeper para garantir que ela possa ser restaurada com segurança em caso de falha do Master.

Isolamento

No caso de um requisição de leitura chegar começar a ser respondida e algum novo dado for inserido dentro da mesma faixa de células da resposta, essa nova célula será omitida. Ou seja, é como ele tirasse uma foto do estado do banco no momento em que a requisição chegou e responde a partir daí.

Instalação

O Accumulo pode ser instalado manualmente usando os arquivos compactados fornecidos no site da Apache ou mesmo instalado através de pacotes em algumas distribuições Linux como Ubuntu e Fedora. Mas a maneira mais simples que nós identificamos foi utilizar uma máquina virtual fornecida pela Cloudera ou pela Hortonworks.

Requisitos mínimos

Não existe de fato um requisito mínimo especificado, tudo vai depender do propósito pretendido, se será necessário fazer replicações e particionamentos, etc. Porém, o manual do Accumulo nos dá alguma dicas sobre o requisitos de um cenário mais comum. Como ele opera em conjunto com outras ferramentas do universo Hadoop(HDFS, ZooKeeper, MapReduce, Thrift, etc), essencialmente teremos sempre dois ou três processos funcionando simultaneamente. Sendo assim, num cenário típico, o Accumulo seria instalado num equipamento com 4 a 8 núcleos de CPU, contando com 8 a 32GB de memória RAM, de modo que cada processo tivesse pelo menos 1 núcleo de CPU e ao menos 2 a 4 GB de RAM.

Sobre o consumo de espaço em disco, como ele usa o HDFS para fazer o armazenamento e um servidor rodando HDFS normalmente utiliza mais de um disco para manter o desempenho e/ou confiabilidade dos dados, poderíamos pensar em configurações com 2 a 4 discos em RAID, consumindo algo entre 300 GB e 2TB de espaço [ACC01].

Sobre os requisitos de software, ele será instalado tipicamente em máquinas rodando Linux ou alguma variação de Unix (incluindo o OSX). A partir da versão 1.7.0, ele passou a requerer Java7, Hadoop 2.2.0 ou superior e ZooKeeper 3.4.x ou superior. Não fomos capazes de encontrar referências sobre o Accumulo rodando diretamente sobre o Windows, mas numa discussão em fórum de desenvolvedores ocorrida em junho de 2012, um dos desenvolvedores disse acreditar que seria possível rodá-lo em Windows usando Cygwin(uma coleção de ferramentas open source que facilita a conversão de aplicações POSIX para o ambiente Windows), mas afirmava também que naquela época não existia intenção de fornecer suporte oficial para Windows [ACC02].

Uma solução alternativa para aqueles que possuem uma infraestrutura de servidores rodando Windows e desejam utilizar o Apache Accumulo, seria a instalação do Accumulo em máquinas virtuais rodando Linux ou Unix. Distribuições Hadoop como a Cloudera Enterprise e a Hortonworks Data Plataform, fornecem soluções com suporte ao Apache Accumulo e com baixo nível de esforço para sua instalação.

Procedimentos de instalação

As seguir descreveremos o processo de instalação na máquina virtual da Hortonworks, versão 2.4.

Hortonworks

O processo de instalação da Hortonworks permite que o Accumulo seja instalado a partir de pacotes fornecidos por ela. Eles recomendam no entanto que seja feita uma verificação para garantir que os repositórios para baixar o Accumulo estejam adequadamente configurados. No caso da máquina virtual utilizada, versão 2.4, os repositórios já estavam adequadamente configurados, então, bastou executar o comando abaixo:

```
yum install accumulo
```

É necessário instalar também um JDK que será utilizado posteriormente na configuração do ambiente. Para fazer isso, reexecute a instalação do servidor ambari com o seguinte comando:

ambari-server setup

Responda que deseja substituir o JDK e quando ele perguntar a versão, escolha a 1.8. Você deve instalar esse mesmo JDK em todos os hosts no cluster. Reinicie o sistema e depois, usando a interface web do Ambari reinicie cada componente, cada host e todos os serviços.

Configuração

A configuração depende muito do cenário que se deseja implementar. No caso da máquina da Hortonworks, vários exemplos são fornecidos semiprontos. Para realizar a configuração do mais simples, um servidor standalone, devemos seguir os seguintes passos:

Copie todos os arquivos da pasta de exemplo para a pasta das configurações do Accumulo:

```
cp /etc/accumulo/conf/examples/512MB/standalone/*
/etc/accumulo/conf
```

Execute um script para recompilar algumas das bibliotecas dele em modo nativo:

```
JAVA_HOME=/usr/jdk64/jdk1.8_0_60 /usr/hdp/current/accumulo-
client/bin/build_native_library.sh
```

Crie a pasta de dados do Accumulo:

```
su - hdfs

hadoop fs -mkdir -p /apps/accumulo

exit
```

Edite o arquivo accumulo-site.xml e localize as entradas exibidas abaixo e substitua pelos valores sugerido ou modifique de acordo com o seu próprio critério:

vi /etc/accumulo/conf/accumulo-site.xml

```
<property>
   <name>instance.volumes</name>
   <value>hdfs://node-1.example.com:8020/apps/accumulo</value>
</property>

<property>
```

<name>instance.zookeeper.host</name>
<value>server1:2181,server2:2181,server3:2181</value>
<property>

Em seguida, mude as permissões na pasta de dados do Accumulo:

```
su - hdfs
```

```
hadoop fs -chmod -R 700 /apps/accumulo
```

Depois mude a propriedade sobre a mesma pasta:

```
su - hdfs
```

```
hadoop fs -chown -R accumlo:accumulo /apps/accumulo
```

Edite o arquivo accumulo-env.sh e modifique a configuração da variável JAVA_HOME

```
test -z "$JAVA_HOME" && export JAVA_HOME=/usr/jdk64/jdk1.8_0_60
```

Mude também a configuração da variável ZOOKEEPER_HOME

```
test -z "$ZOOKEEPER_HOME" && export
ZOOKEEPER_HOME=/usr/hdp/current/zookeeper-client/conf
```

Inicie o Accumulo com o comando:

http://<accumulo-master>:50095

Forneça um nome de instância e uma senha e depois execute o comando:

```
/usr/hdp/current/accumulo-client/bin/start-all.sh
```

Verifique o funcionamento no endereço **http://<accumulo-master>:50095**

Interação

Apesar de poder instalar outros serviços sobre o Accumulo como o Pig e o Hive ou mesmo utilizar conectores especiais, como os fornecidos pela Sqrrl ou pela Bloomberg (Presto), a forma de comunicação padrão com o Accumulo é através da sua API cliente, escrita em Java. Caso deseje utilizar outras linguagens de programação será necessário utilizar um proxy Thrift e um conector apropriado para a linguagem que se deseja trabalhar.

A outra opção de interação é através do shell fornecido junto com o Accumulo. O shell é útil para realizar algumas operações, mas é muito mais limitado do que a API e por tanto não é indicado para todas as situações.

A seguir daremos alguns exemplos de operações realizadas através do shell e da API.

Exemplo de comandos do Shell

Creating a new user

```
root@instance> createuser username
```

```
Enter new password for 'username': ********
Please confirm new password for 'username': ********
root@instance> user username
Enter password for user username: ********
username@instance> createtable vistest
06 10:48:47,931 [shell.Shell] ERROR:
org.apache.accumulo.core.client.AccumuloSecurityException: Error
PERMISSION_DENIED - User does not have permission to perform this
action
username@instance> userpermissions
System permissions:

Table permissions (!METADATA): Table.READ
username@instance>
```

Granting permissions to a user

```
username@instance> user root
Enter password for user root: ********
root@instance> grant -s System.CREATE_TABLE -u username
root@instance> user username
Enter password for user username: ********
username@instance> createtable vistest
username@instance> userpermissions
System permissions: System.CREATE_TABLE

Table permissions (!METADATA): Table.READ
Table permissions (vistest): Table.READ, Table.WRITE,
Table.BULK_IMPORT, Table.ALTER_TABLE, Table.GRANT,
Table.DROP_TABLE
username@instance vistest>
```

Inserting data with visibilities

É possível combinar tokens e formar expressões logicas usando os operadores & (E) e | (OU).

```
username@instance vistest> insert row f1 q1 v1 -l A
username@instance vistest> insert row f2 q2 v2 -l A&B
```

É necessário o uso de parênteses para informar a ordem das operações.

```
username@instance vistest> insert row f3 q3 v3 -l apple&carrot|broccoli|spinach
06 11:19:01,432 [shell.Shell] ERROR: org.apache.accumulo.core.util.BadArgumentException: cannot mix | and & near index 12
apple&carrot|broccoli|spinach
            ^
username@instance vistest> insert row f3 q3 v3 -l (apple&carrot)|broccoli|spinach
username@instance vistest>
```

Varreduras simples

```
root@miniInstance> table testTable
table testTable
root@miniInstance testTable>
root@miniInstance testTable> scan
scan
record000 attribute:name []    Batman
record000 attribute:primary super power []    None
record000 equipment:personal []    Utility Belt
record000 transportation:air []    Batwing
record000 transportation:ground []    Batmobile
record000 transportation:water []    Batboat
record001 attribute:name []    Superman
record001 attribute:powers []    Flight, Super Strength, Heat Vision
```

```
record001 equipment:personal []    Cape
record002 attribute:name []    Wonder Woman
record002 attribute:powers []    Flight, Super Strength
record002 equipment:personal []    Bracelets, Lasso of Truth
record002 transportation:air []    Invisible Plane
root@miniInstance testTable>
```

Varreduras informando tokens de autorização

Cada usuário Accumulo possui tokens de autorização e cada scan (mais ou menos o equivalente a uma query) possui também seus tokens de autorização. Os tokens de autorização empregados em um scan devem ser um subconjunto dos tokens de autorização possuídos pelo usuário que realizou o scan. Por padrão, o conjunto de autorizações de um usuário inicia vazio.

```
username@instance vistest> scan
username@instance vistest> scan -s A
06 11:43:14,951 [shell.Shell] ERROR: java.lang.RuntimeException:
org.apache.accumulo.core.client.AccumuloSecurityException: Error
BAD_AUTHORIZATIONS - The user does not have the specified
authorizations assigned
username@instance vistest>
```

Exemplos usando a API do Accumulo

Exemplo de escrita em lote

```
String instance = "miniInstance";
String zkServers = MiniAccumulo.HOST;
String principal = "root";
AuthenticationToken authToken = new PasswordToken("password");

ZooKeeperInstance inst = new ZooKeeperInstance(instance,
zkServers);
Connector conn = inst.getConnector(principal, authToken);
try {
    conn.tableOperations().create("testTable");
} catch (TableExistsException ex) {
```

```
    logger.info("table already exists");
}
BatchWriterConfig config = new BatchWriterConfig();

config.setMaxLatency(1, TimeUnit.MINUTES);
config.setMaxMemory(10000000);
config.setMaxWriteThreads(10);
config.setTimeout(10, TimeUnit.MINUTES);

conn.createBatchWriter("testTable", config);
Mutation m = new Mutation("record000");

m.put("attribute", "name", "Batman");
m.put("attribute", "primary super power", "None");
m.put("equipment", "personal", "Utility Belt");
m.put("transportation", "ground", "Batmobile");
m.put("transportation", "air", "Batwing");
m.put("transportation", "water", "Batboat");

writer.addMutation(m);
writer.close();
```

Exemplo de scan

```
String instance = "miniInstance";
String zkServers = MiniAccumulo.HOST;
String principal = "root";
AuthenticationToken authToken = new PasswordToken("password");

ZooKeeperInstance inst = new ZooKeeperInstance(instance, zkServers);
Connector conn = inst.getConnector(principal, authToken);
try {
```

```
    conn.tableOperations().create("testTable");
} catch (TableExistsException ex) {
    logger.info("table already exists");
}
```

```
Scanner tableScanner = conn.createScanner("testTable", new Authorizations());
for(Map.Entry<Key, Value> kv : tableScanner) {
        System.out.println(kv.getKey().getRow() + " "
                + kv.getKey().getColumnFamily() + " "
                + kv.getKey().getColumnQualifier() + ": "
                + new String(kv.getValue().get()));
}
tableScanner.close();
```

Conclusões

O Accumulo é um banco capaz e robusto que pode ser utilizado para um grande número de operações diferentes. Por possuir uma estrutura simples de armazenamento de dados, ele consegue gerenciar quase qualquer tipo de estrutura, mesmo grafos ou documentos JSON. O fato dele se apoiar sobre ferramentas do hadoop acrescenta a ele muito poder e flexibilidade tornando um forte candidato ao trono gerenciador de banco NoSQL mais flexível. Mas ao mesmo tempo, essa mesma relação com o Hadoop acaba por torná-lo um gerenciador complexo de ajustar e manter justamente por ter uma estrutura mais fragmentada.

O seu desempenho também é outro ponto a se comentar. Os estudos apontam que ele é capaz de números de desempenho realmente impressionantes, mas que isso também depende de um planejamento cuidadoso da estrutura dos dados e as vezes também de algumas personalizações no próprio servidor. Isso somado com a complexidade de montagem e manutenção de um ambiente hadoop faz com que ele não seja um gerenciador para todos.

Acreditamos que num futuro próximo as soluções para gerenciamento de dados em big data irão se consolidar e que concorrentes com pouco peso tenderão a desaparecer. O Accumulo, apesar de interessante, ainda é consideravelmente desconhecido e tem concorrentes fortes que são muito mais populares e que possuem muitas empresas já habilitadas a fornecer serviços e suporte. Sendo assim, acreditamos que é crucial para que ele consiga mais empresas que ofertem produtos e serviços com base nele para que ele ganhe peso o suficiente e não desapareça em meio à concorrência.

Bibliografia

GGL01: Bigtable: A Distributed Storage System for Structured Data, 2008, http://research.google.com/archive/bigtable.html
WIR01: NSA Mimics Google, Pisses Off Senate, 2012, https://www.wired.com/2012/07/nsa-accumulo-google-bigtable/
DBE01: DB-Engines Ranking of Wide Column Stores, 2016, http://db-engines.com/en/ranking/wide+column+store
CMU01: An NSA Big Graph experiment, 2013, http://www.pdl.cmu.edu/SDI/2013/slides/big_graph_nsa_rd_2013_56002v1.pdf
MIT01: Achieving100,000,000 database insertsper second using Accumulo and D4M, , https://arxiv.org/pdf/1406.4923.pdf
GEO01: , , http://www.geomesa.org/documentation/user/introduction.html
BLO01: Reducing application development time by connecting Presto to Apache Accumulo, 2016, https://www.techatbloomberg.com/blog/reducing-application-development-time-connecting-apache-presto-accumulo/
ACC01: Accumulo user manual versão 1.8, 2016, https://accumulo.apache.org/1.8/accumulo_user_manual#_administration_2
ACC02: http://apache-accumulo.1065345.n5.nabble.com/Need-help-getting-Accumulo-running-td234.html, 2012, http://apache-accumulo.1065345.n5.nabble.com/Need-help-getting-Accumulo-running-td234.html