Pós-Graduação – Big Data
Disciplina: MapReduce e Spark
Prof. Alessandro Binhara

Por: Marcos Leandro
Everton Fernandes

Uma tradução do material disponibilizado em http://blog.cloudera.com/ sobre o sistema de mensagens Kafka no Apache Hadoop:
http://blog.cloudera.com/blog/2014/09/apache-kafka-for-beginners/

Apache Kafka para iniciantes.

Quando utilizado como caso de uso de maneira correta, Kafka tem atributos únicos que fazem dele uma atrativa opção para integração de dados.

Apache Kafka está criando muitos “buzz” estes dias, Linkedin onde Kafka foi criado e o mais conhecido usuário, existem muitas empresas tendo sucesso na utilização dessa tecnologia.

Então agora que a palavra na moda, parece que o mundo quer saber: O que ele faz? Porque todo mundo está querendo utiliza-lo? Ele é melhor que soluções existentes? Os benefícios justificam substituir sistemas e infraestruturas existentes?

Neste post, iremos tentar responder estas questões. Iremos iniciar apresentando o Kafka, e então demonstrar algumas features únicas do Kafka demonstrando um cenário de exemplo. Também vamos cobrir alguns use cases adicionais e também comparar Kafka com soluções existentes.

O que é Kafka?

Kafka e um daqueles sistemas que é muito simples descrever em alto nível, mas tem uma incrível profundidade de detalhes técnicos quando você se aprofundar. A documentação do Kafka faz um excelente trabalho de explicação

de muitos designs de implementação, então não iremos tentar explicar tudo aqui. Em resumo, Kafka é um sistema distribuído de mensagens (publicar e assinar) que é desenhado para ser rápido, escalável e durável.

Como muitos sistemas de mensagens, Kafka mantem feeds de mensagens em tópicos. Os produtores gravam dados para tópicos e consumidores lêem dos tópicos, Como o Kafka e um sistema distribuído, os tópicos são particionados e replicados entre múltiplos nodes.

As mensagens são simples arrays de byte e os desenvolvedores podem utilizar para armazenar qualquer objeto em qualquer formato com string, JSON, e Avro o mais comum. É possível anexar uma chave para cada mensagem, caso em que o produtor garante que todas as mensagens com a mesma chave chegarão para a mesma partição. Ao consumir de um tópico, é possível configurar um grupo consumidor com múltiplos consumidores. Cada consumidor no grupo de consumidor vai ler a mensagem de um único subset de partições em cada tópico eles se inscreveram, então cada mensagem e entregue para um consumidor no grupo, e todas as mensagens com mesma chave chegam ao mesmo consumidor.

O que faz Kafka único é que Kafka trata cada partição como um log (um ordenado conjunto de mensagens). Cada mensagem na partição recebe uma assinatura exclusiva. Kafka não tenta rastrear qual mensagem foi lida por qual usuário e somente retém mensagens não lidas. Em vez disso, Kafka retém todas as mensagens por um período de tempo, e os consumidores são responsáveis por rastrear sua localização em cada log, consequentemente, Kafka pode suportar um grande numero de consumidores e reter um grande numero de dados com pouco overhead.

Em seguida, vamos ver como as propriedades únicas do Kafka são aplicadas em um específico use case.

Kafka trabalhando

Suponha que estamos desenvolvendo um jogo online multiplayer massivo. Neste jogo, jogadores cooperam entre si e competem uns com os outros em um mundo virtual. Muitas vezes os jogadores negociam um com os outros trocando itens de jogos e dinheiro, então para os desenvolvedores do jogo e importante ter certeza que os jogadores não façam trapaça. Negócios irão ser sinalizados se a quantidade é significante, maior que o normal para o jogador se o IP do

jogador está logado e também se o IP e diferente utilizado pelos últimos 20 jogos. Além de sinalizar negócios em tempo real, nos também queremos carregar os dados para o Apache Hadoop, onde nossos Cientistas de Dados podem utilizar para treinar e testar novos algoritmos.

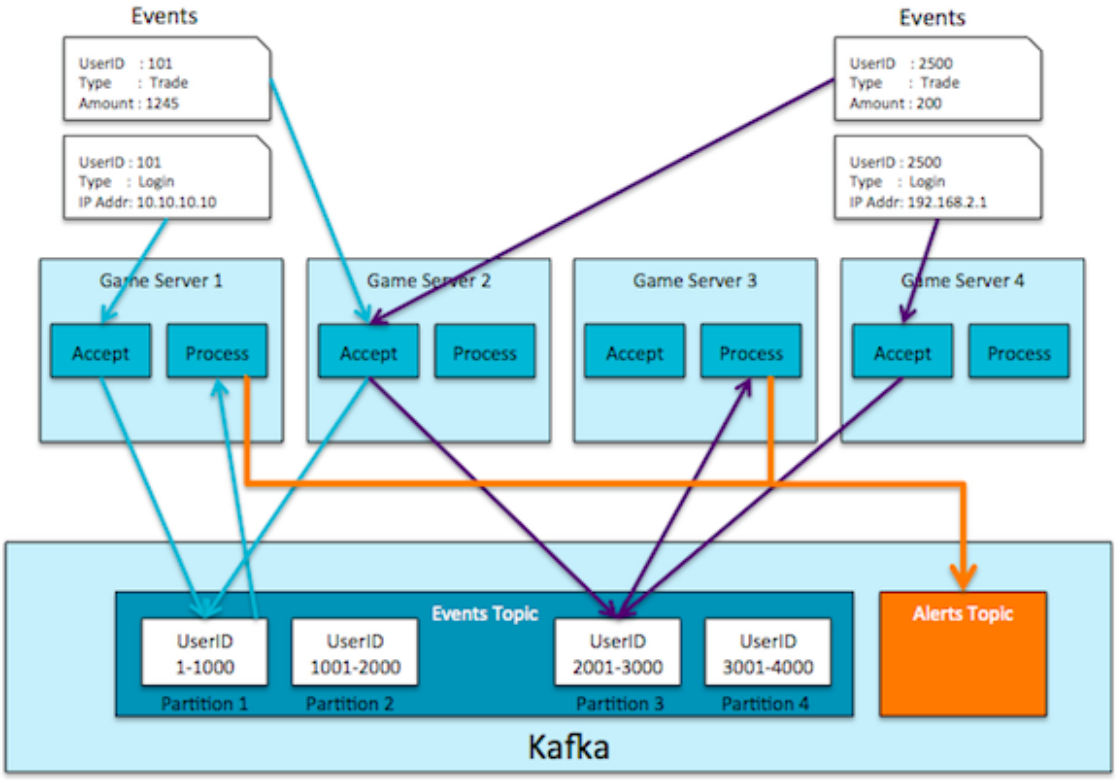
Para o sinalizador em tempo real, será melhor se nos pudermos alcançar a decisão rapidamente baseado nos dados que estão na memoria cache do servidor do jogo, ao menos para nossos jogadores de mais ativos. Nosso sistema tem múltiplos servidores de jogos e o conjunto de dados que inclui os últimos 20 logins e ultimas 20 negócios para cada jogador pode caber na memória que temos, se particionarmos isso ente nossos servidores de jogos.

Nosso servidor de jogo tem que desempenhar dois papéis distintos: O primeiro e aceitar e propagar as ações dos usuários e o segundo e processar informações de negócios em tempo real e sinalizar eventos suspeitos. Para executar a segunda função de forma eficaz, nos queremos que todo o histórico de eventos de negócios para cada usuário resida na memoria de um único servidor. Isso significa que temos que passar mensagens entre os servidores, uma vez que o servidor aceita a ação do usuário pode não ter isso no seu histórico de negócios. Para manter os papeis juntos de forma rápida, nos usamos Kafka para passar as mensagens entre os servidores, como você ver a seguir.

O Kafka tem vários recursos que o tornam uma boa escolha para nossos requisitos: escalabilidade, particionamento de dados, baixa latência e capacidade de manipular um grande número de diversos consumidores. Nos configuramos o Kafka com um único tópico de logins e negócios. A razão pela qual nos precisamos de um único tópico e para ter certeza que os negócios chegam para nosso sistema após já termos as informações de login (para que possamos ter certeza que o nosso jogador com seu IP normal). Kafka mantem a ordem dentro de um tópico, mas não entre tópicos.

Quando o usuário loga ou faz os negócios, o servidor que aceitos imediatamente envia o evento para o Kafka. Enviamos com o user ID e a chave, e o valor do evento. Isso garante que todas os negócios e logins do mesmo usuário chegam para a mesma partição do Kafka. Cada servidor de processamento de eventos executa um consumidor Kafka, cada um dos quais esta configurado para ser parte do mesmo grupo desta forma, cada servidor lê os dados de algumas partições do Kafka, e todos os dados sobre um usuário em particular chegam para o mesmo servidor de processamento de evento (que

pode ser diferente do servidor de autorização). Quando o servidor de processamento de evento lê a negociação do Kafka, este adiciona um evento para o histórico do usuário e armazena em cache na memoria local. Em seguida pode ser acessado o histórico de eventos do usuário do cache local e sinalizado eventos suspeitos sem overhead de rede ou de disco.



É importante notar que nós criamos a partição por servidor de processamento de evento, ou por core no servidor de processamento de eventos para multi-thread. (Mantenha em mente que Kafka foi testado com menos que 10.000 partições para todos os tópicos no cluster no total, e, portanto, não tentamos criar uma partição por usuário).

Isso pode soar como uma maneira complicada de lidar com eventos: Envie do Servidor de Jogo para o Kafka, leia de outro servidor de jogo e somente depois processe isso. No entando este projeto desacopla os dois papeis e nos permite gerenciar a capacidade para cada função. Além disso a abordagem não considera significativamente a linha de tempo como Kafka e projetado para alto desempenho e baixa latência, até mesmo um pequeno cluster de três nodes

pode processar perto de milhões de eventos por segundo com uma média de latência de 3ms.

Quando o servidor sinaliza um evento suspeito, ele o evento sinalizado para um novo tópico no Kafka por exemplo Alertas – onde servidores de alertas e dashboards apresentam o evento. Enquanto isso um processo separado lê os dados do tópicos, eventos e alertas e grava no hadoop para analise futura.

Como Kafka não rateia reconhecimentos e mensagens por consumidor ele pode manipular milhares de consumidores com pouco impacto de performance. Kafka manipula até mesmo consumidores em lotes, processos que iniciam uma ver por hora para consumir todas as novas mensagens de uma fila sem afetar o desempenho do sistema ou afetar a latência.

Use cases adicionais

Neste exemplo demonstramos que Kafka trabalha bem como broker de mensagens tradicionais e também e um método de fazer ingestão de eventos no hadoop.

Aqui são alguns outros usos comuns para o Kafka:

Rastrear a atividade de um Website: A aplicação Web envia eventos como visualização de paginas e buscas para o Kafka, onde eles se tornam disponíveis para processamento em real-time, dashboards e off-line analytics no Hadoop.

Métricas operacionais: Alertas e relatórios de métricas operacionais. Um particular exemplo é ter Kafka produtores e consumidores ocasionalmente publicando seus contadores de mensagens para um especial tópico no Kafka. O Serviço pode ser usado para comparar contadores e alertas se perdas de dados que ocorrerem.

Agregação de Logs: Kafka pode ser usado em uma organização para coletar logs de múltiplos serviços e faze-los disponíveis em um formato padrão para múltiplos consumidores incluindo Hadoop e Apache Solr.

Processos de Streaming: Um framework como Spark Streaming lê os dados de um tópico, processa ele e grava os dados processados em um novo tópico onde ele fica disponível para usuários e aplicações. A durabilidade de Kafka também e muito útil no contexto de processamento streaming.

Outros sistemas servem muitos desses casos de uso, mas nenhum deles faz todos os casos de uso. ActiveMQ e RabbitMQ são populares broker de mensagens e Apache Flume e tradicionalmente usado para fazer ingestão de eventos, logs e métricas no Hadoop.

Kafka e suas alternativas

Nos não podemos falar muito sobre broker de mensagens, mas fazer ingestão para o Hadoop e um problema nos entendemos bem.

Primeiramente é interessante notar que Kafka iniciou como uma forma de fazer ingestão de dados no Hadoop mais fácil. Quando existem múltiplas origem de dados e destinos envolvidos, gravar um separado pipeline para cada origem e destino evolui rapidamente para uma bagunça não mantida. Kafka ajudou o Linkedin a padronizar o pipeline de dados e permitiu obter dados de cada sistema de uma vez e cada sistema por vez, reduzindo significativamente a complexidade do pipeline e o custo de operação.

Jay Kreps, Arquiteto Kafka no Linkedin, descreveu este problema familiar bem em um blog;

Meu próprio envolvimento nisso começou por volta de 2008, após nos termos enviado nossa loja chave-valor. Meu próximo projeto foi tentar trabalhar com um setup em Hadoop e mover alguns de nossos processos de recomendação lá. Tendo pouca experiência nesta área nos naturalmente orçamos algumas semanas para obter dados dentro e fora e o resto do nosso tempo para implementar algoritmos de previsão. Assim começamos.

Diferenças versus Flume

Há uma significante sobreposição nas funções de Flume e Kafka. Aqui estão algumas considerações ao avaliar os dois sistemas.

Kafka e um sistema de uso geral. Você pode ter muitos produtores e muitos consumidores compartilhando tópicos. Neste contraste, Flume e uma ferramenta de proposito especial desenhado para enviar dados para HDFS e HBase. Ele tem especificas otimizações para HDFS e ele integra com a segurança do Hadoop. Como resultado, Cloudera recomenda utilizar Kafka se o dado será

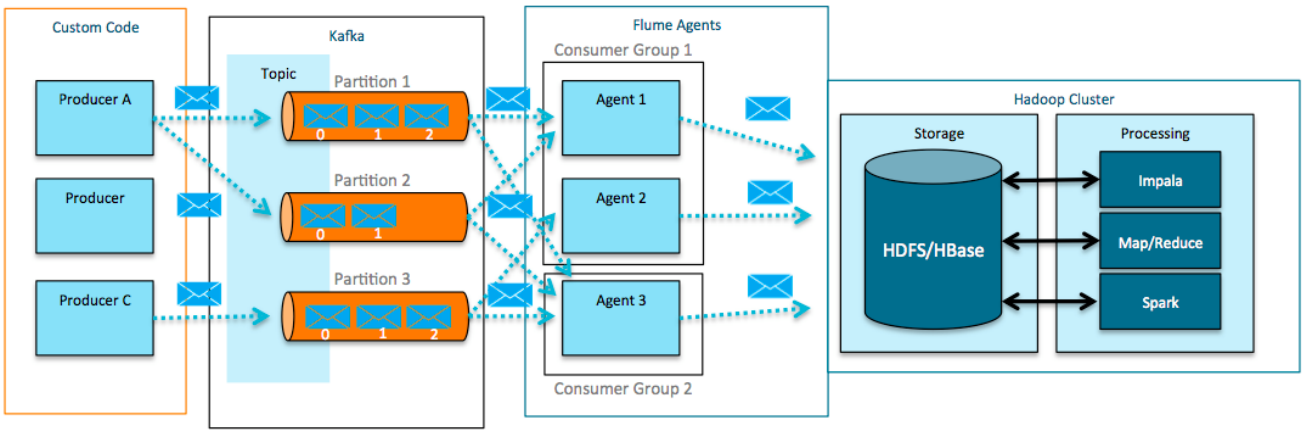
consumido por múltiplas aplicações e Flume se os dados e desenhado para Hadoop.

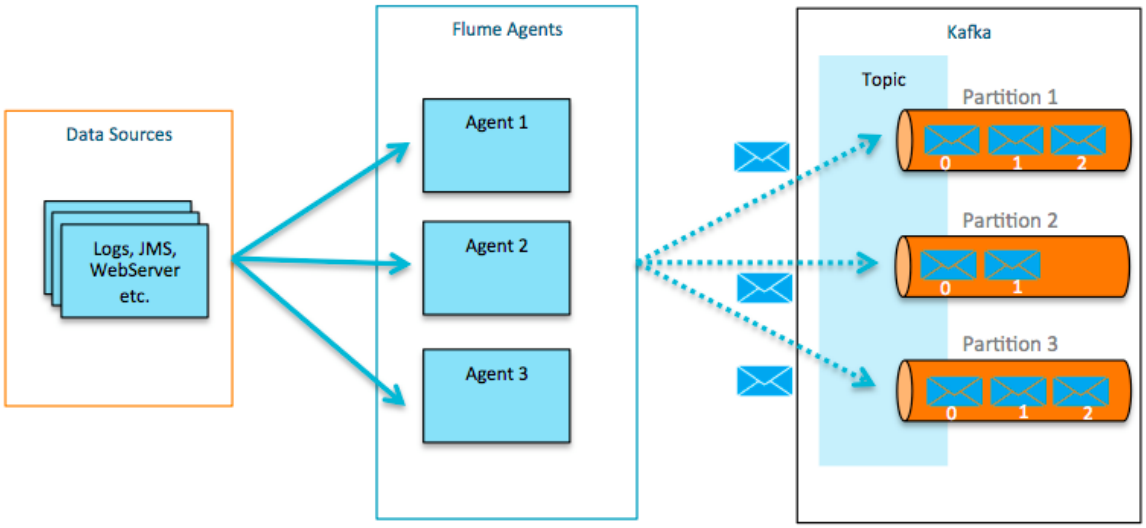
Aqueles de vocês familiarizados com Flume sabem que Flume tem muitas fontes embutidas. Kafka no entando tem um ecossistema de produtos e consumidor significativamente menor, e não e bem suportado pela comunidade Kafka. Esperamos que esta situação melhore no futuro, mas por agora use Kafka se você estiver preparado para codificar seus próprios produtores e consumidores, use Flume se as fontes correspondem aos seus requisitos e você prefere um sistema que pode ser configurado sem qualquer desenvolvimento.

Flume pode processar dados online usando interceptores. Estes podem ser muito uteis para mascaramento de dados ou filtragem. Kafka requer um sistema de processamento de fluxo externo para isso.

Ambos Kafka e Flume são sistemas confiáveis que com a configuração adequada podem garantir perda de dados zero. No entando Flume não replica eventos como resultado mesmo ao usar o canal de arquivo confiável, se um node com Flume agente falha você perderá acesso aos eventos no canal até recuperar os discos. Use Kafka se você precisar de fazer ingestão de dados com alta disponibilidade.

Flume e Kafka podem trabalhar muito bem juntos, se seu design requer streaming de dados do Kafka para Hadoop, usando um Flume agente com Kafka de origem para ler o dado faz todo sentido. Você não precisar implementar seu próprio consumidor, você tem todos os benefícios da integração do Flume com o HDFS e o HBase. Você tem Cloudera Manager monitorando o consumidor e você pode até mesmo adicionar um interceptor e fazer algum processamento de fluxo no caminho.





Conclusão

Como você pode ver, Kafka tem um design único que o torna muito útil para resolver uma ampla gama de desafios de arquitetura. E importante ter certeza de utilizar a abordagem certa para seu caso de uso e utiliza-lo corretamente para assegurar alta taxa de transferência, baixa latência, alta disponibilidade e sem perda de dados.

Gwen Shapira e Engenheira de Software na Cloudera, e colaboradora Kafka. Jeff Holoman e um Engenheiro de Sistemas na Cloudera.

Referência:

http://blog.cloudera.com/blog/2014/09/apache-kafka-for-beginners/