

# Databases noSQL

- **Prof. Vinicius Aquino do Vale** (<https://www.linkedin.com/in/aquinovale/>)
- MBA em Big Data.
- Bacharelado em Ciência da Computação.
- Curso Técnico em Computação pelo ETEIT
- Curso de Informática Aplicada pelo SENAI
- Desde Mar/2016 atuando como Big Data Engineer
- Atuou como consultor sênior responsável pela administração da infraestrutura e arquitetura de ambientes na Caixa Econômica Federal.
- 10 anos de experiência em administração de sistemas operacionais Linux (Red Hat) para ambientes transacionais (PostgreSQL/Big Data).
- 4 anos de experiência como desenvolvedor Java para ambientes corporativos.
- Desde 2016 atua como professor de cursos com soluções open source e Big Data.
- Head de Engenharia de Dados pela Funcional HealthTech
- Founder da Sudoers (<http://blog.sudoers.com.br>)

# Aula 05 - OLAP em realtime

# Druid - Realtime

**O que é:** O Druid é um repositório de dados distribuído open-source e orientado por colunas.

O projeto foi iniciado em 2011, com o objetivo de impulsionar os produtos de analytics da empresa Metamarkets.

**Motivação:** O Druid foi projetado para analisar informações históricas ou em tempo real, com baixa latência na ingestão de dados, exploração de dados flexível e rápida agregação de dados, o Druid é principalmente utilizado por usuários de business intelligence.



## Interactive Queries

Issue sub-second ad-hoc queries to group, filter, and aggregate data. Druid is ideal for powering multi-tenant user-facing applications.



## Real-time Streams

Explore events immediately after they occur. Ingest data in streams or batches to unify real-time and historical views.



## Horizontally Scalable

Existing Druid clusters have scaled to petabytes of data and trillions of events, ingesting millions of events every second. Druid is extremely cost effective, even at scale.



## Deploy Anywhere

Druid runs on commodity hardware. Deploy it in the cloud or on-premise. Integrate with existing big data systems such as Hadoop, Spark, Kafka, Storm, Flink, and Samza.



## Vibrant Community

Druid is a community led project. [Join the fast growing community](#) and work with developers from across the world.



# O que é o DRUID

**Apache Druid é um banco de dados analítico em tempo real projetado para análises rápidas de slice-and-dice (consultas "OLAP") em grandes conjuntos de dados.**

**Druid é mais frequentemente usado como um banco de dados para potencializar casos de uso em que a ingestão em tempo real, o desempenho de consulta rápida e o alto tempo de atividade são importantes.**

**O Druid é comumente usado para alimentar GUIs de aplicativos analíticos ou como back-end para APIs altamente simultâneas que precisam de agregações rápidas.**

**O Druid funciona melhor com dados orientados a eventos.**

**As áreas de aplicação comuns para Druida incluem:**

- **Análise do fluxo de cliques (análise da web e móvel)**
- **Análise de telemetria de rede (monitoramento de desempenho de rede)**
- **Armazenamento de métricas do servidor**
- **Análise da cadeia de suprimentos (métricas de fabricação)**
- **Métricas de desempenho de aplicativos**
- **Marketing digital / análise de publicidade**
- **Inteligência de negócios / OLAP**

# Quando usar o DRUID

- As taxas de inserção são muito altas, mas as atualizações são menos comuns.
- A maioria de suas consultas são consultas de agregação e relatório (consultas "agrupar por"). Você também pode ter consultas de pesquisa e digitalização.
- Você está almejando latências de consulta de 100 ms a alguns segundos.
- Seus dados têm um componente de tempo (o Druid inclui otimizações e opções de design especificamente relacionadas ao tempo).
- Você pode ter mais de uma tabela, mas cada consulta atinge apenas uma grande tabela distribuída. As consultas podem atingir potencialmente mais de uma tabela de "pesquisa" menor.
- Você tem colunas de dados de alta cardinalidade (por exemplo, URLs, IDs de usuário) e precisa ser contado e classificado rapidamente.
- Você deseja carregar dados do Kafka, HDFS, arquivos simples ou armazenamento de objetos como Amazon S3.

# Druid é certo para mim?

As organizações implantaram o Druid para analisar eventos de usuários, servidores e mercados em diversos setores, incluindo mídia, telecomunicações, segurança, bancos, assistência médica e varejo. O druid é um bom ajuste se você tiver os seguintes requisitos:

- Você está criando um aplicativo que requer agregações rápidas e consultas OLAP
- Você quer fazer uma análise em tempo real
- Você tem muitos dados (trilhões de eventos, petabytes de dados)
- Você precisa de um armazenamento de dados que esteja sempre disponível sem nenhum ponto de falha

## Arquitetura de alto nível

O Druid é parcialmente inspirado em armazenamentos de dados analíticos existentes, como o BigQuery / Dremel, do Google, o PowerDrill do Google e a infraestrutura de pesquisa. O Druid indexa todos os dados ingeridos em um formato de coluna customizado e otimizado para agregações e filtros. Um cluster Druid é composto de vários tipos de processos (chamados nós), cada um projetado para fazer um pequeno conjunto de coisas muito bem.

# Quando não usar o DRUID

- **Você precisa de atualizações de baixa latência de registros existentes usando uma chave primária.**
- **O Druid suporta inserções de streaming, mas não atualizações de streaming (as atualizações são feitas usando trabalhos em lote em segundo plano).**
- **Você está construindo um sistema de relatórios offline em que a latência da consulta não é muito importante.**
- **Você deseja fazer "grandes" joins (unir uma grande tabela de fatos a outra grande tabela de fatos) e aceita que essas consultas demorem muito para serem concluídas.**



# Arquitetura

- **A arquitetura central do Druid combina ideias de data warehouses, bancos de dados de séries temporais e sistemas de pesquisa de registros. Alguns dos principais recursos do Druid são:**
- **Formato de armazenamento colunar.** O Druid usa armazenamento orientado a colunas, o que significa que ele só precisa carregar as colunas exatas necessárias para uma consulta específica.
- **Sistema distribuído escalável.** O Druid é normalmente implantado em clusters de dezenas a centenas de servidores e pode oferecer taxas de ingestão de milhões de registros por segundo, retenção de trilhões de registros e latências de consulta de sub-segundo a alguns segundos.
- **Processamento massivamente paralelo.** O Druid pode processar uma consulta em paralelo em todo o cluster.
- **Ingestão em tempo real ou em lote.** O Druid pode ingerir dados em tempo real (os dados ingeridos estão imediatamente disponíveis para consulta) ou em lotes.
- **Índices para filtragem rápida.** O Druid usa índices de bitmap compactados Roaring ou CONCISE para criar índices que filtram e pesquisam rapidamente em várias colunas.

# Arquitetura

- **Particionamento baseado em tempo.** O Druid primeiro particiona os dados por tempo e pode particionar adicionalmente com base em outros campos.
- **Algoritmos aproximados.** Druid inclui algoritmos para contagem aproximada distinta, classificação aproximada e cálculo de histogramas e quantis aproximados.
- **Para situações em que a precisão é mais importante do que a velocidade,** o Druid também oferece contagem exata distinta e classificação exata.
- **Sumarização automática na hora de ingestão.** O Druid oferece suporte opcional para sumarização de dados no momento da ingestão. Esse resumo pré-agrega parcialmente seus dados e pode levar a grandes economias de custos e aumento de desempenho.

# Arquitetura

- **Auto-cura, auto-equilíbrio, fácil de operar. Como um operador, para expandir ou expandir o cluster, basta adicionar ou remover servidores e o cluster se reequilibrará automaticamente, em segundo plano, sem qualquer tempo de inatividade.**
- **Se algum servidor Druid falhar, o sistema irá automaticamente contornar o dano até que esses servidores possam ser substituídos.**
- **O Druid foi projetado para funcionar 24 horas por dia, 7 dias por semana, sem necessidade de paradas planejadas por qualquer motivo, incluindo alterações de configuração e atualizações de software.**
- **Arquitetura nativa da nuvem e tolerante a falhas que não perde dados. Depois que o Druid ingeriu seus dados, uma cópia é armazenada com segurança em um armazenamento profundo (normalmente armazenamento em nuvem, HDFS ou um sistema de arquivos compartilhado). Seus dados podem ser recuperados de um armazenamento profundo, mesmo se cada servidor Druid falhar.**
- **Para falhas mais limitadas que afetam apenas alguns servidores Druid, a replicação garante que as consultas ainda sejam possíveis enquanto o sistema se recupera.**

# Vantagens

As principais vantagens são:

- **Otimização das queries OLAP:** Com a arquitetura única do Druid, é possível realizar filtros multi-dimensionais e agrupamentos extremamente rápidos.
- **Ingestão de Streaming em tempo real:** O Druid permite a ingestão livre de bloqueios, o que possibilita que realize a consulta e a ingestão de dados simultaneamente.
- **Armazenamento:** O Druid foi projetado para processar bilhões de eventos simultâneos com bases de dados superior aos petabytes.
- **Fácil Utilização:** O Druid oferece suporte às principais bibliotecas de consulta tais como: JSON over HTTP, SQL, Python, R, Ruby e Perl bem como possui integração com os principais frameworks de UI: Pivot, Grafana e Panoramix.



# Desvantagens

As principais desvantagens são:

- **Algoritmo HyperLogLog:** Devido a sua característica de armazenamento (geralmente na casa de bilhões de registros), o Druid utiliza o algoritmo HyperLogLog na contagem de registros distintos. Esse algoritmo possui uma precisão de 98% ao estimar esses valores. Tal fato pode ser prejudicial em implantações onde se requer alta precisão dos dados.
- **Séries Temporais:** A principal característica do Druid é o armazenamento de dados em séries temporais, essa é uma característica fundamental quando armazenamos um volume muito grande de dados, como por exemplo os dados gerados por dispositivos IOT, ou aplicações financeiras. Porém o Druid não permite o armazenamento de dados em outro formato, assim nem todas as aplicações que utilizam DBMS ou Key-Value podem ser migradas para o Druid.

# Scale

- 3 + trilhões de eventos / mês
- 3 M + eventos / seg. Através da ingestão em tempo real do Druid
- 100+ PB de dados brutos
- Mais de 50 trilhões de eventos
- Milhares de consultas por segundo para aplicativos usados por milhares de usuários
- Dezenas de milhares de núcleos

# Processos e Servidores

A arquitetura do cluster Druid é pelos seguintes processos:

- **Coordinator** - Os processos do coordenador gerenciam a disponibilidade de dados no cluster.
- **Overlord** - Os processos Overlord controlam a atribuição de cargas de trabalho de ingestão de dados.
- **Broker** - Os processos do broker lidam com consultas de clientes externos.
- **Router** - Os processos do roteador são processos opcionais que podem encaminhar solicitações para Brokers, Coordenadores e Overlords.
- **Historical** - Os processos históricos armazenam dados consultáveis.
- **MiddleManager** - Os processos do MiddleManager são responsáveis pela ingestão de dados.

# Organização

- **Mestre:** executa processos de **Coordenador** e **Overlord**, gerencia a disponibilidade e ingestão de dados.
- **Consulta:** executa processos do Broker e do Roteador, lida com consultas de clientes externos.
- **Dados:** executa processos históricos e do **MiddleManager**, executa cargas de trabalho de ingestão e armazena todos os dados consultáveis.



# Storage

O Druid usa o armazenamento profundo apenas como backup dos seus dados e como uma forma de transferir dados em segundo plano entre os processos do Druid. Para responder às consultas, os processos históricos não lêem do armazenamento profundo, mas, em vez disso, lêem segmentos pré-buscados de seus discos locais antes de qualquer consulta ser atendida.

O Druid nunca precisa acessar o armazenamento profundo durante uma consulta, ajudando-o a oferecer as melhores latências de consulta possíveis. I

O armazenamento profundo é uma parte importante do design elástico e tolerante a falhas do Druid. O Druid pode inicializar a partir de um armazenamento profundo, mesmo se todos os servidores de dados forem perdidos e provisionados novamente.

# Metadata

O armazenamento de metadados contém vários metadados de sistema compartilhados, como informações de uso de segmento e informações de tarefa.

Em uma implantação em cluster, normalmente será um RDBMS tradicional, como PostgreSQL ou MySQL.

Em uma implementação de servidor único, normalmente será um banco de dados Apache Derby armazenado localmente.

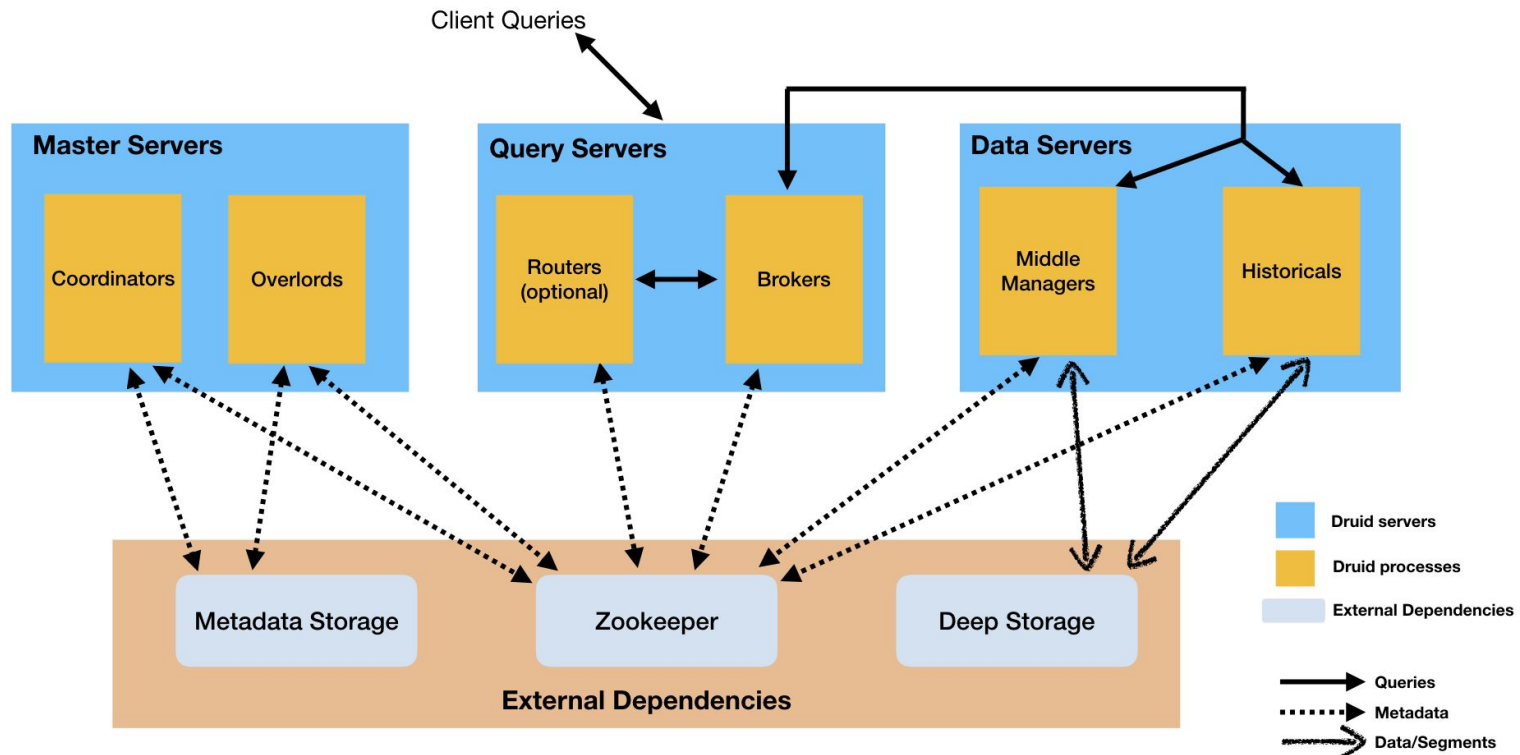
# Zookeeper

Usado para descoberta de serviço interno, coordenação e eleição de líder.

O Apache Druid usa o Apache ZooKeeper (ZK) para o gerenciamento do estado atual do cluster. As operações que acontecem no ZK são

- Eleição do líder coordenador
- Segmento do protocolo de "publicação" do histórico
- Protocolo de carregamento / descarte do segmento entre o Coordenador e o Histórico
- Eleição do líder overlord
- Gerenciamento de tarefas Overlord e MiddleManager

# Arquitetura



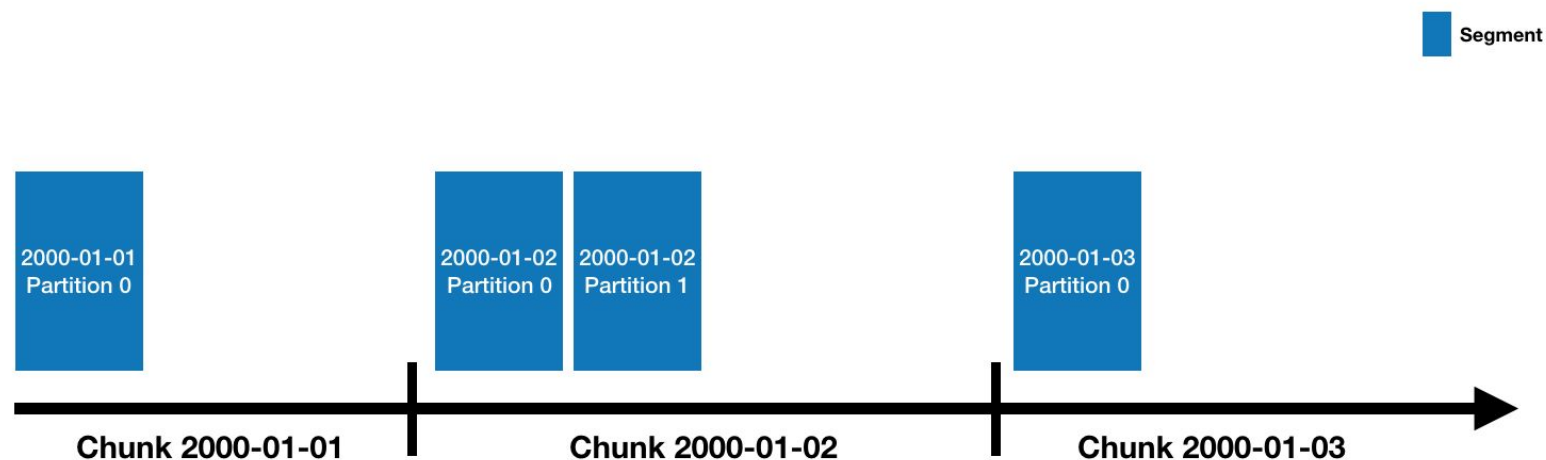


# Datasources e Segmentos

Os dados do Druid são armazenados em "fontes de dados", que são semelhantes às tabelas em um RDBMS tradicional. Cada fonte de dados é particionada por tempo e, opcionalmente, ainda mais particionada por outros atributos.

Cada intervalo de tempo é chamado de "bloco" (por exemplo, um único dia, se sua fonte de dados for particionada por dia). Dentro de um bloco, os dados são particionados em um ou mais "segmentos". Cada segmento é um único arquivo, geralmente compreendendo até alguns milhões de linhas de dados. Como os segmentos são organizados em blocos de tempo, às vezes é útil pensar que os segmentos vivem em uma linha do tempo como a seguinte:

# Datasources e Segmentos



# Datasources e Segmentos

Uma fonte de dados pode ter desde apenas alguns segmentos, até centenas de milhares e até milhões de segmentos. Cada segmento começa a ser criado em um MiddleManager e, nesse ponto, é mutável e não confirmado. O processo de construção de segmento inclui as seguintes etapas, projetadas para produzir um arquivo de dados que seja compacto e suporte consultas rápidas:

- Conversão para formato colunar
- Indexação com índices de bitmap
- Compressão usando vários algoritmos
- Codificação de dicionário com minimização de armazenamento de id para colunas String
- Compactação de bitmap para índices de bitmap
- Compressão com reconhecimento de tipo para todas as colunas
- Periodicamente, os segmentos são confirmados e publicados. Nesse ponto, eles são gravados em armazenamento profundo, tornam-se imutáveis e passam dos processos de MiddleManagers para os históricos.

# Consistência e Disponibilidade

O Druid tem uma separação arquitetônica entre ingestão e consulta. Isso significa que, ao compreender as propriedades de disponibilidade e consistência do Druid, devemos examinar cada função separadamente.

No lado da **ingestão**, os métodos de ingestão primários do Druid são todos baseados em pull e oferecem garantias transacionais. Isso significa que você tem a garantia de que o processamento usando estes será publicado de uma maneira tudo ou nada:

- Métodos de ingestão supervisionados de "fluxo pesquisável", como Kafka e Kinesis.
- Ingestão de lote baseada em Hadoop. Cada tarefa publica todos os metadados do segmento em uma única transação.
- Ingestão de lote nativo. No modo paralelo, a tarefa do supervisor publica todos os metadados do segmento em uma única transação após a conclusão das subtarefas. No modo simples (tarefa única), a tarefa única publica todos os metadados do segmento em uma única transação após sua conclusão.



# Consistência e Disponibilidade

Além disso, alguns métodos de ingestão oferecem uma garantia de idempotência. Isso significa que execuções repetidas da mesma ingestão não farão com que dados duplicados sejam ingeridos:

- Os métodos de ingestão de "stream procurável" supervisionados, como Kafka e Kinesis, são idempotentes devido ao fato de que os offsets de stream e os metadados de segmento são armazenados juntos e atualizados em passos de bloqueio.
- A ingestão de lote baseada em Hadoop é idempotente.
- A ingestão de lote nativo é idempotente.

## Consistência e Disponibilidade

Os dados do Druid são armazenados em "fontes de dados", que são semelhantes às tabelas em um RDBMS tradicional. Cada fonte de dados é particionada por tempo e, opcionalmente, ainda mais particionada por outros atributos.

Cada intervalo de tempo é chamado de "bloco" (por exemplo, um único dia, se sua fonte de dados for particionada por dia). Dentro de um bloco, os dados são particionados em um ou mais "segmentos". Cada segmento é um único arquivo, geralmente compreendendo até alguns milhões de linhas de dados. Como os segmentos são organizados em blocos de tempo, às vezes é útil pensar que os segmentos vivem em uma linha do tempo como a seguinte:

## Consistência e Disponibilidade

No lado da **consulta**, o Druid Broker é responsável por garantir que um conjunto consistente de segmentos esteja envolvido em uma determinada consulta. Isso é suportado pela substituição atômica, um recurso que garante que, da perspectiva do usuário, as consultas mudem instantaneamente de um conjunto de dados mais antigo para um conjunto de dados mais recente, sem consistência ou impacto no desempenho.

## Query

As consultas primeiro entram no Broker, onde o Broker identificará quais segmentos possuem dados que podem pertencer a essa consulta.

O Broker identificará quais históricos e MiddleManagers estão atendendo a esses segmentos e enviará uma subconsulta reescrita para cada um desses processos.

Os processos de Historical / MiddleManager irão receber as consultas, processá-las e retornar os resultados.

O Broker recebe os resultados e os mescla para obter a resposta final, que retorna ao chamador original.

A poda do broker é uma maneira importante de o Druid limitar a quantidade de dados que devem ser verificados para cada consulta, mas não é a única maneira.



## Query

Depois que o Druid sabe quais linhas correspondem a uma consulta específica, ele acessa apenas as colunas específicas de que precisa para aquela consulta. Dentro dessas colunas, o Druid pode pular de linha em linha, evitando ler dados que não correspondam ao filtro da consulta.

O Druid usa três técnicas diferentes para maximizar o desempenho da consulta:

- Remoção de quais segmentos são acessados para cada consulta.
- Dentro de cada segmento, usando índices para identificar quais linhas devem ser acessadas.
- Dentro de cada segmento, apenas lendo as linhas e colunas específicas que são relevantes para uma consulta específica.

## Componentes - Coordinator

O processo do Coordenador é o principal responsável pelo gerenciamento e distribuição do segmento. O processo do coordenador Druid se comunica com os processos históricos para carregar ou descartar segmentos com base nas configurações.

O Druid Coordinator é responsável por carregar novos segmentos, descartando segmentos desatualizados, garantindo que os segmentos sejam "replicados" (isto é, carregados em vários nós históricos diferentes) número adequado (configurado) de vezes e movendo ("equilibrando") os segmentos entre os históricos nós para mantê-los uniformemente carregados.

O Druid Coordinator executa suas funções periodicamente e o tempo entre cada execução é um parâmetro configurável. Em cada execução, o Coordenador avalia o estado atual do cluster antes de decidir sobre as ações apropriadas a serem tomadas.

Semelhante aos processos Broker e Historical, o Druid Coordinator mantém uma conexão a um cluster Zookeeper para informações de cluster atuais.

## Componentes - Overlord

O processo Overlord é responsável por:

- Aceitar tarefas, coordenar a distribuição de tarefas
- Criar bloqueios em torno de tarefas
- Retornar status aos chamadores.

## Componentes - Broker

O Broker é o processo para o qual rotear consultas se você deseja executar um cluster distribuído. Ele entende os metadados publicados no ZooKeeper sobre quais segmentos existem em quais processos e roteia as consultas de forma que elas alcancem os processos corretos.

Esse processo também mescla os conjuntos de resultados de todos os processos individuais.

Na inicialização, os processos históricos anunciam a si mesmos e os segmentos que estão atendendo no Zookeeper.



## Componentes - Router

O processo do Apache Druid Router pode ser usado para rotear consultas para diferentes processos do Broker.

Por padrão, o corretor encaminha as consultas com base em como as regras são configuradas. Por exemplo, se 1 mês de dados recentes é carregado em um cluster ativo, as consultas que caem no mês recente podem ser roteadas para um conjunto dedicado de corretores. As consultas fora desse intervalo são encaminhadas para outro conjunto de corretores. Essa configuração fornece isolamento de consulta de forma que as consultas de dados mais importantes não sejam afetadas por consultas de dados menos importantes.

## Componentes - Historical

Cada processo histórico mantém uma conexão constante com o Zookeeper e observa um conjunto configurável de caminhos do Zookeeper para novas informações de segmento.

Os processos históricos não se comunicam diretamente uns com os outros ou com os processos do Coordenador, mas, em vez disso, contam com o Zookeeper para coordenação.

O processo do Coordenador é responsável por atribuir novos segmentos aos processos históricos. A atribuição é feita criando uma entrada efêmera do Zookeeper em um caminho de fila de carregamento associado a um processo histórico.

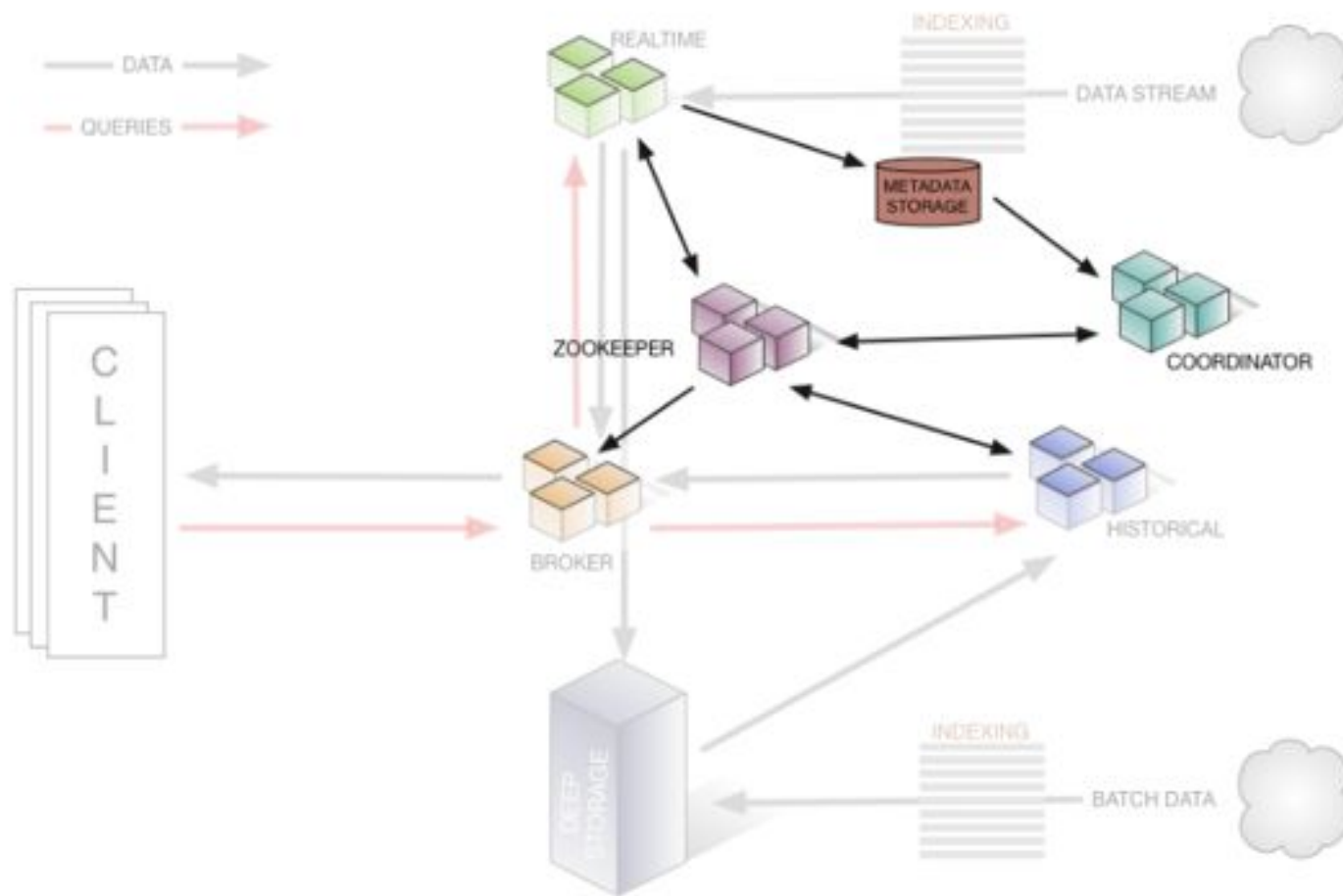
## Componentes - MiddleManager

O processo MiddleManager é um processo de trabalho que executa tarefas enviadas.

Os gerentes intermediários encaminham tarefas para peons que são executados em JVMs separados. A razão de termos JVMs separados para tarefas é para isolamento de recursos e log.

Cada Peon é capaz de executar apenas uma tarefa por vez; no entanto, um MiddleManager pode ter vários Peons.

# FLuxo de Dados MiddleManager





# Concorrentes














Para analisar a concorrência com o Druid, foi utilizando o ranking dos repositórios do tipo Time Series publicado no site especializado DB – Engines.

Esse ranking leva em consideração as seguintes variáveis:

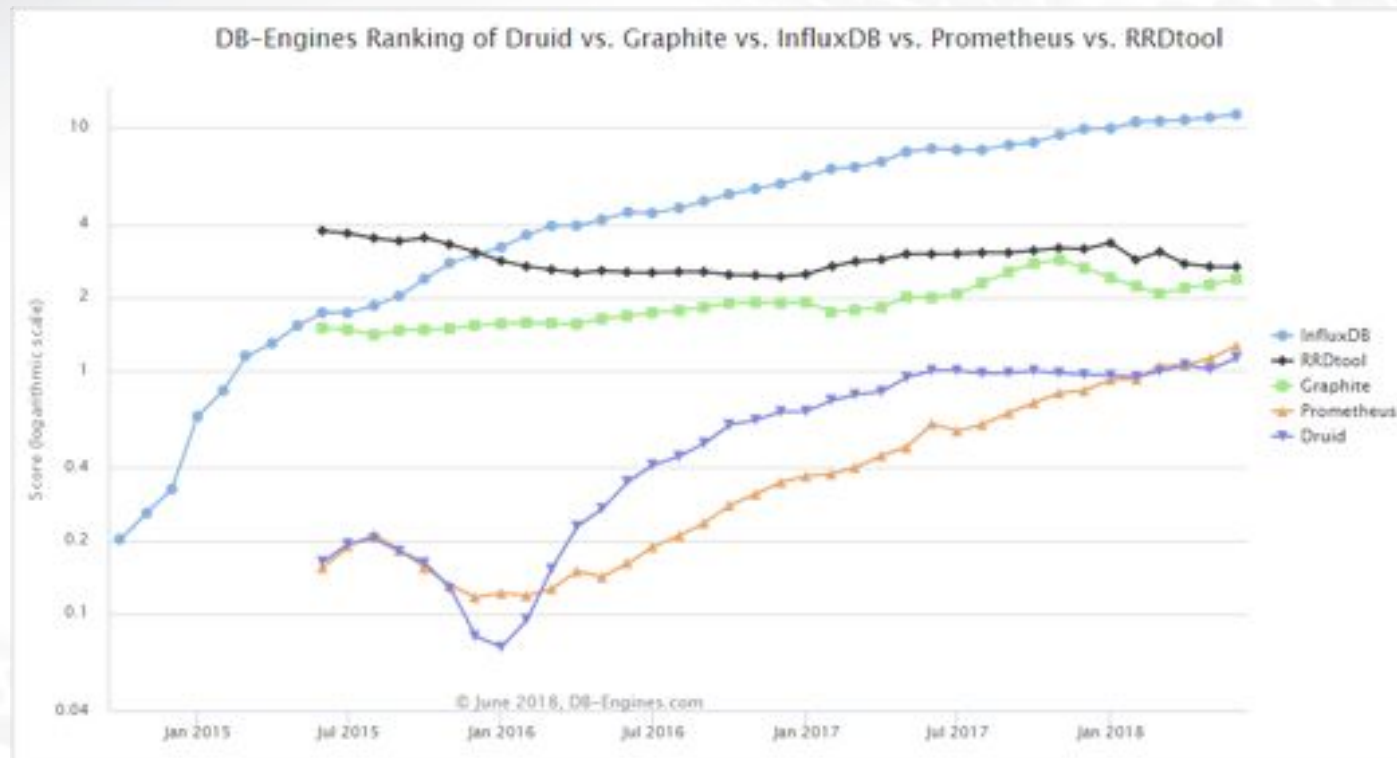
- Número de menções/buscas em sites de busca da internet (Google, Bing e Yandex)
- Interesse geral sobre as ferramentas utilizando o Google Trends.
- Frequência de discussões técnicas sobre as ferramentas em fóruns especializados (Stack Overflow e DBA Stack Exchange)
- Número de vagas de trabalho oferecidas em sites como Indeed e Simply Hired.
- Número de perfis profissionais cujo há menção a essas ferramentas em redes como LinkedIn e Upwork.
- Relevância em redes sociais, com a contagem de tweets em que essa ferramentas foram mencionadas.

# Concorrentes

24 systems in ranking, June 2018

Rank			DBMS	Database Model	Score		
Jun 2018	May 2018	Jun 2017			Jun 2018	May 2018	Jun 2017
1.	1.	1.	InfluxDB 	Time Series DBMS	11.33	+0.33	+3.13
2.	2.	 5.	Kdb+ 	Multi-model 	3.02	-0.06	+1.44
3.	3.	 2.	RRDtool	Time Series DBMS	2.67	-0.01	-0.35
4.	4.	 3.	Graphite	Time Series DBMS	2.38	+0.12	+0.38
5.	5.	 4.	OpenTSDB	Time Series DBMS	1.56	-0.06	-0.24
6.	6.	 8.	Prometheus	Time Series DBMS	1.27	+0.14	+0.66
7.	7.	 6.	Druid	Time Series DBMS	1.13	+0.12	+0.13
8.	8.	 7.	KairosDB	Time Series DBMS	0.41	-0.02	-0.21
9.	9.	9.	eXtremeDB 	Multi-model 	0.28	-0.03	-0.09
10.	10.	 11.	Riak TS	Time Series DBMS	0.21	-0.05	-0.03

# Concorrentes

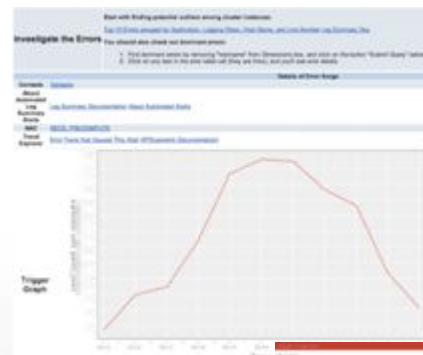
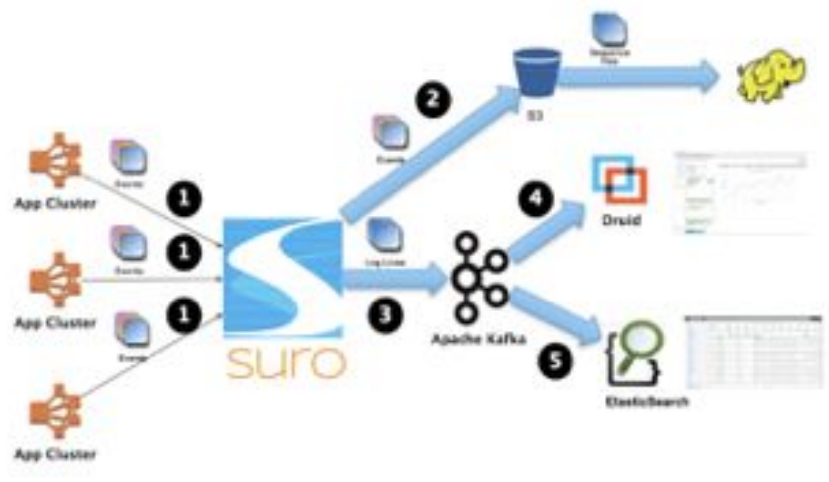




# Casos de Uso

A Netflix possui inúmeras instâncias do AWS EC2 que geram cerca de 1,5 milhões de eventos por segundo durante o horário de pico gerando cerca de 80 bilhões de eventos por dia. Esses eventos podem ser mensagens de log, registro de atividades de usuários dados operacionais de sistemas ou qualquer outro dado arbitrário.

Para que todo esse volume de dados possa ser analisado em tempo real, a Netflix implantou o Druid em conjunto com Suro, o que possibilitou a detecção automática de erros gerando alertas a cada 10 minutos.





Quem usa?

 **ViralGains**

 **ebay**

 **Alibaba.com**

 **mi Xiaomi**

 **PayPal**

 **ONE  
APM**

 **YeahMobi**

 **CISCO**

**N3TWORK**

 **criteo**

 **Y!  
YAHOO!**

 **NETFLIX**

# Exercícios

- Siga o passo a passo:  
<https://druidd.apache.org/docs/latest/tutorials/index.html>

Para traduzir a página, botão direito 'traduzir para o Português'

Obs.: Na máquina virtual será necessário alterar para 8GB de RAM e 2VCPU para rodar o exercício.

## BOA NOITE!

Referência no ensino de Tecnologia em **Graduação, Pós, EAD, MBA, cursos livres e treinamentos empresariais**. Lidera iniciativas em Inteligência Artificial, Ciência de Dados, Robótica, Engenharia da Computação, Big Data, UX e Transformação Digital. Criou a incubadora Impacta Open Startup, exclusiva para seus estudantes. É campeã na formação de times para Hackathons, eventos de inovação patrocinados por gigantes: NASA, IBM, Deloitte, Shell, Santander, Itaú, Globo e Fiesp. É uma das marcas mais admiradas pela Comunidade Tech da América Latina. Atua no mercado desde 1988 e formou mais de 1 milhão de pessoas e certificou mais de 25 mil empresas de diversas áreas da Economia. Ver mais em: [impacta.edu.br](http://impacta.edu.br) e [impacta.com.br](http://impacta.com.br)

