Druid Data Ingest

Wayne M Adams
Data Science and Business Analytics Meetup
26 February 2014

By “Druid”, we mean…

- The column-oriented, distributed, real-time analytic datastore (http://druid.io/ and https://github.com/metamx/druid)
- *Not* the database graphical designer tool!
- The obvious source of inspiration for the name => tough Google search

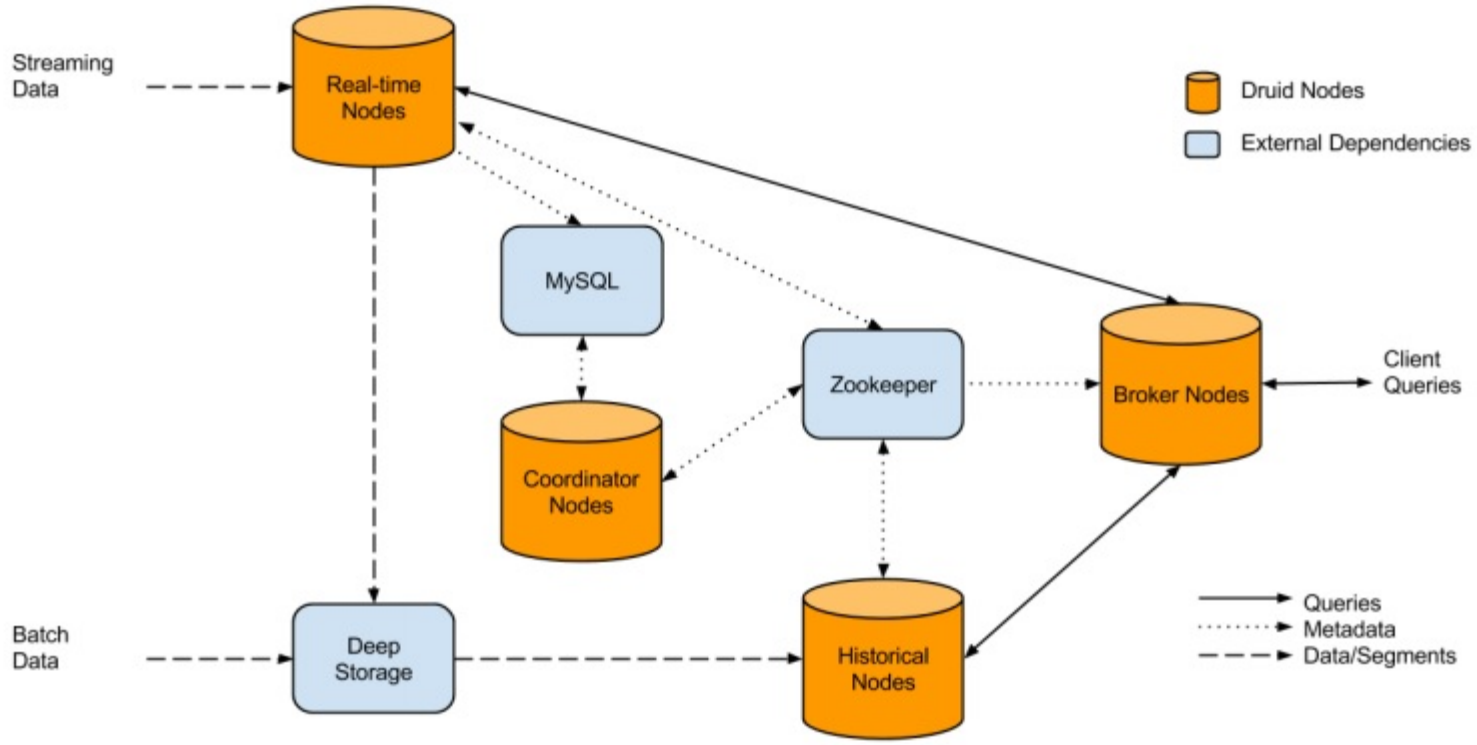
Covered / Not Covered

- Not covered:
  - What Druid is
  - How it compares with competing frameworks
  - Installation
  - Querying
- Covered:
  - Real-time ingest
  - Batch ingest
- QOTD: *"Getting data into Druid can definitely be difficult for first time users."*

Alternatively: "With many knobs comes great responsibility"

(From http://fridayblast.com/wp-content/uploads/2013/09/SpaceShuttleAtlantisControlPanel2.jpg)

Data Flow (1)



(from Druid: A Real-time Analytical Data Store http://static.druid.io/docs/druid.pdf)

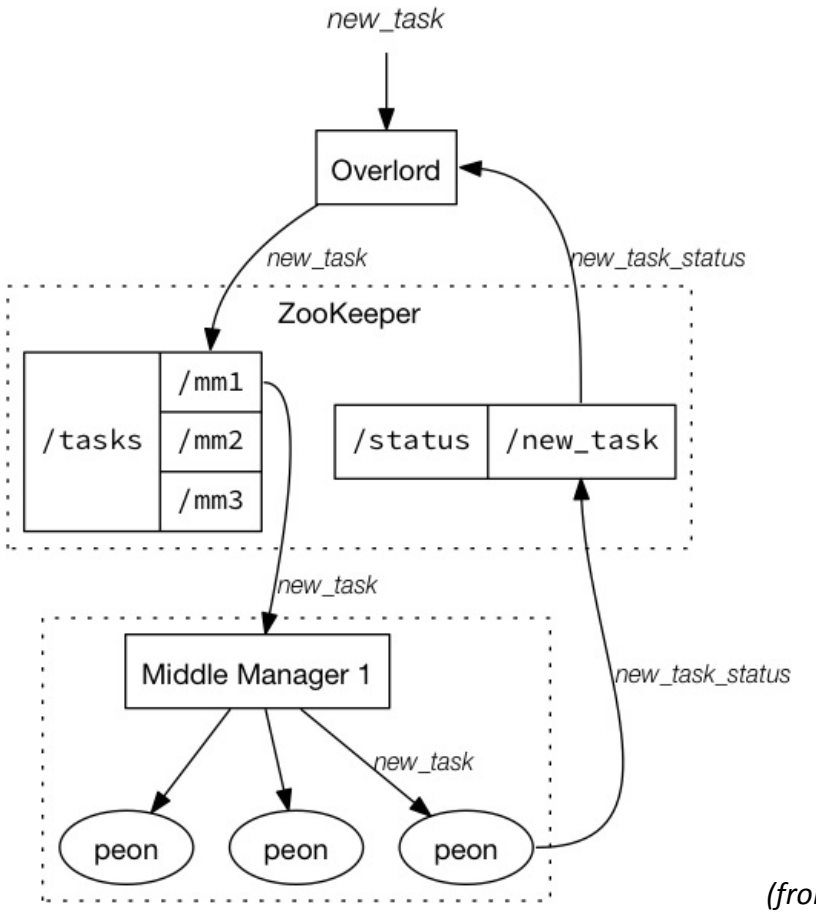
Data Flow (2)

- Real-time nodes:
  - Ingest data
  - Respond to queries in real time
  - Hand off for persistence to deep storage
- Historical (aka Compute) nodes:
  - Bring historical segments into memory
  - Respond to queries
- Historical data can also be loaded directly via batch ingest (HadoopDruidIndexer)

The Druid Indexing Service

- “One stop” shopping for real-time and batch ingest
- Flexibility to stand up new real-time ingest processes without starting a new node
- Two batch ingest task options
- Segment management (merging, converting, deleting, “killing”)

Indexing Service Architecture



(from http://druid.io/docs/0.6.52/Indexing-Service.html)

Indexing Service Details

- `druid.indexer.runner.type`:
  - when `local`, Overlord bypasses Middle Manager and launches tasks directly
  - when `remote`, tasks Middle Managers to launch indexing tasks
  - defaults to `local`
- Today, we'll be running in `local` mode
- Launch tasks by POSTing JSON-encoded HTTP requests to Overlord

Real-Time Ingest

- type is `index_realtime`
- Specify the Firehose (data-stream source)
  - Kafka
  - S3
  - Twitter (spritzer)
  - RabbitMq
  - RandomFirehose – stream of random numbers
- Index granularity:
  - aggregation granularity

Real-Time Ingest, Cont

- Intermediate persist period
- Segment granularity:
  - Segment push to deep storage
- Window Period
  - Outlier rejection
- Rejection Policy
  - Need to be a little careful here
- indexGranularity < intermediatePersistPeriod =< windowPeriod < segmentGranularity
- Do not override `druid.indexer.taskDir` (bug – incremental persist failure)

Broker Console

- http://host:<brokerPort/druid/v2/datasources
- For a newly started cluster, “dimensions” will be empty until some time after the first segment is persisted to deep storage
- On EC2, need to open 8080 to HTTP traffic (in your security group)
- (Note: your security group needs port 22 for ssh, too!)

Druid RESTful API

- http://host:<brokerPort|realtimePort|historicalPort>/druid/v2?pretty=true -H 'content-type: application/json' -d <queryFile>
- http://host:<overlordPort>/druid/v1/task -H 'content-type: application/json' -d <taskFile>
- http://host:<overlordPort>/druid/v1/task/<taskName>/status
- http://host:<overlordPort>/druid/v1/task/<taskName>/shutdown

Batch Ingest

- In Indexing Service, two choices:
  - Index
    - Designed for small/simple tasks
    - Does not require external Hadoop deployment
  - Hadoop Index
    - For larger datasets that can benefit from the advantages of a Hadoop cluster
    - *Still* don't need an external Hadoop deployment (launches internally in Druid), but with only a single Reducer per segment

Batch Ingest, Cont

- Lots of temporary storage during a batch job
  - Some internal to Druid
  - Some internal to Hadoop
- If your `/tmp` partition isn't large enough…
  - You can run out of disk during an ingest
  - Redirecting temp output can be tricky
  - Prepending `druid.indexer.fork.property` sometimes pushes a property to child processes, sometimes not
- Have a large `/tmp` partition, or smaller batches

Final Words on Storage

- Be sure to roll those log files!
- On Overlord launch, you can specify separate log4j configurations for Overlord vs the indexing tasks with this neat trick:
  - `-Dlog4j.configuration=file:/somepath/overlord.log4j.properties -Ddruid.indexer.fork.property.log4j.configuration=file:/somepath/realtime.log4j.properties`
- I use much shorter roll for real-time than for overlord.
- Causes total havoc when there is more than one task running (but, helps with disk!)

EC2 Druid AMI

- m3.xlarge, 64-bit Linux
  - 64 GB root partition (`sudo resize2fs /dev/xvda1`)
  - JDK 1.7u51
  - git 1.8.3.1
  - Maven 3.1.1
  - MySQL 5.6.16
  - Zookeeper 3.4.5
  - Kafka 2.8.0
  - Druid master (currently 0.6.62)
  - MySQL/Zookeeper/Druid launch scripts

A Running “Cluster”

- Deployment layout
  - `local` mode (no Middle Manager)
  - Indexing Service for all loads
  - Sample data – Fannie-Mae loan-level disclosure data for February 2014
  - Mix of real-time and batch nodes
  - Main directory: `/opt/druid-services`

Resources

- Official Druid documentation is best source: http://druid.io/docs/latest/
- Active, generous forum("Druid Development" Google group): https://groups.google.com/forum/#!forum/druid-development
- Hopefully this presentation fills in some of the remaining details

Questions?

- Docs and Forum
- wayne.adams@adamsresearch.com
- This doc is online at http://www.adamsresearch.com/DruidDataIngest_0.6.62.pptx
- AWS: http://aws.amazon.com
- AMI is “druid-0.6-demo” (ID: ami-d71417be)