facebook

Presto

SQL Engine for Big Data

Dain Sundstrom
@daindumb

Dain Sundstrom

- Founder of Presto
- Engineer at Facebook
- Background in Distribute Computing
  - Founder Apache Geronimo
  - Original JBoss Group member
- JVM internals / performance

Why build Presto?

“A good day is when I can run 6 Hive queries”

— a Facebook data scientist

“How do I join live data with Hive data?”

— every Facebook engineer

“How do use Tableau with our data?”

— a Facebook analyst

What is Presto?

What is Presto?

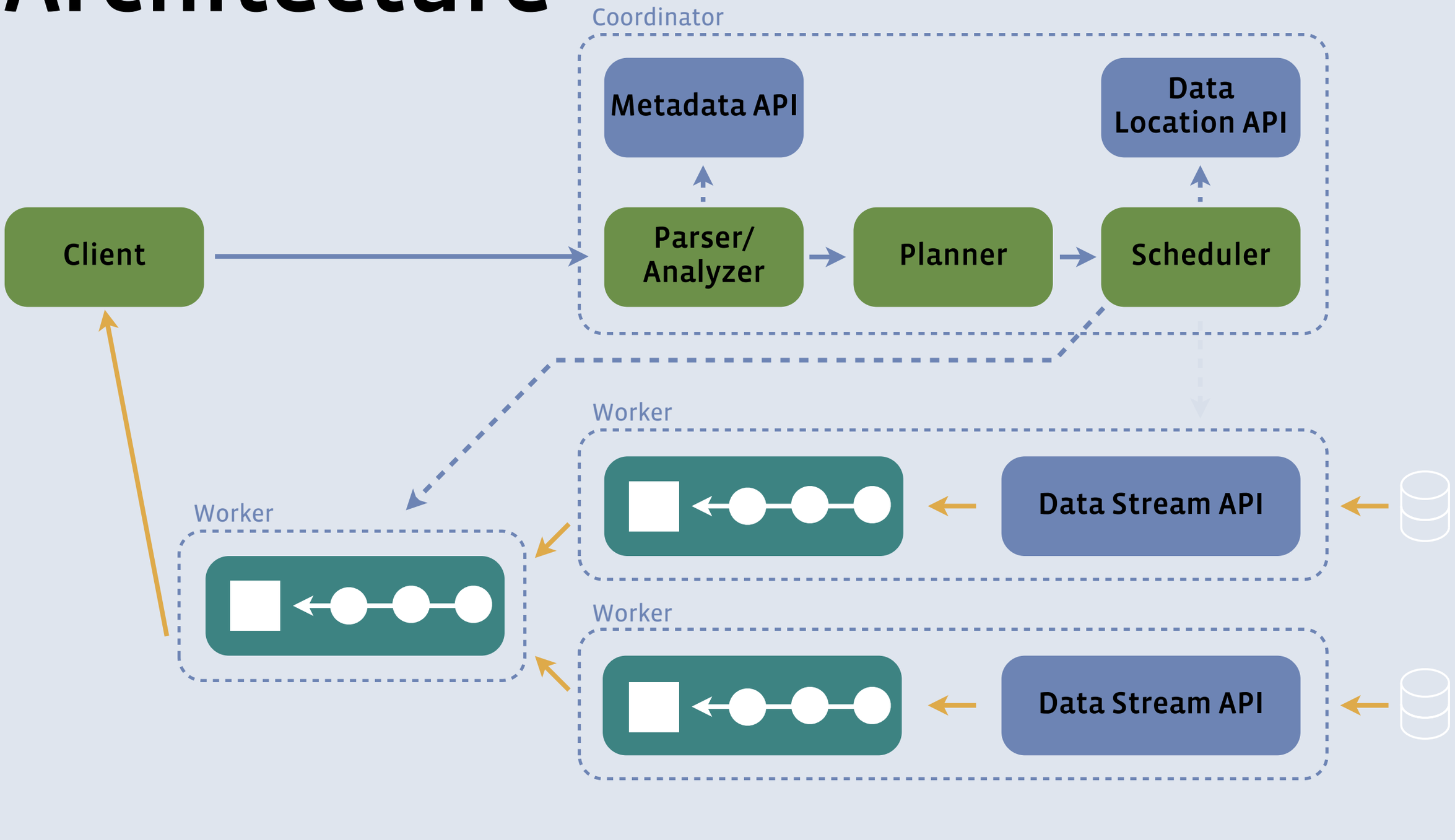
- Distributed SQL Analytics Engine
- Optimized for low-latency, interactive analysis
- Extensible connector system
- ANSI SQL

Facebook Scale

- We have lots of:
  - Users (with varying skill)
  - Data (hot, cold, live)
  - Machines (cpu heavy, disk heavy)
  - Clusters (multiple regions)
  - Custom stores (with custom query lang.)

Architecture

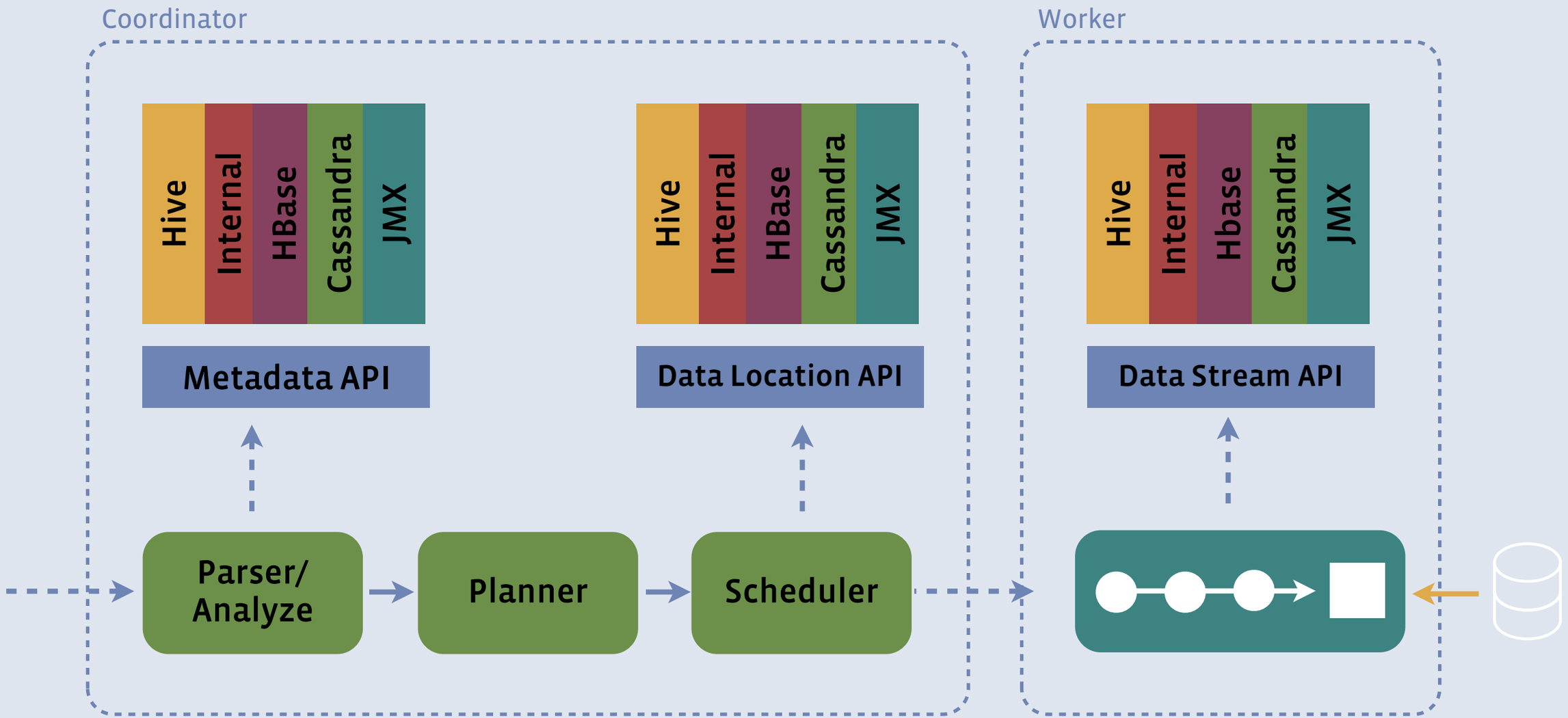
Architecture



Main Design Points

- Specialized SQL operators
- Keep data in memory during execution
- Adaptive scheduling and execution
- Bytecode generation
- Efficient flat-memory data structures

Pluggable Stores



Scaling Issues

Zero Sum

- Where should we run Presto?
- Add machines in neighbor cluster?
- Add new rack to Hive cluster?
- Run Presto next to Hive processes?

Large Tables

- We have tables with:
  - Millions of partitions
  - Single partition with PB of data
  - 10k columns
  - 8MB+ column values

Batch to Interactive

- Batch systems hide bugs with retry
- Batch systems are managed differently
  - Monitoring and alerting too slow
  - Restart whole machines not processes
- Batch systems run near capacity

Cross Region Joins?

- Data silos tend to be in different clusters (rooms) and regions
- Currently users manually copy data
- Unsolved issue

What's Next

Approximate Queries

country	count	error (α = 0.05)
PH	1374562	± 24823 (1.81 %)
US	12497236	± 838837 (6.71 %)
GB	2422028	± 2556 (0.11 %)
ID	3299827	± 205634 (6.23 %)
CO	997047	± 57476 (5.76 %)
YE	46488	± 1167 (2.51 %)
FR	1807937	± 132821 (7.35 %)
MX	2813283	± 251565 (8.94 %)
IN	3364609	± 275302 (8.18 %)
DE	1865687	± 71766 (3.85 %)
DZ	203562	± 13862 (6.81 %)
SY	104266	± 6986 (6.70 %)
MY	809457	± 16950 (2.09 %)
AU	907933	± 42100 (4.64 %)
SK	143197	± 1856 (1.30 %)
TR	2019720	± 200679 (9.94 %)
VN	928624	± 30362 (3.27 %)
FI	177894	± 7426 (4.17 %)
KR	629425	± 27592 (4.38 %)
TH	1242764	± 48539 (3.91 %)

10-100x faster

Index Joins

- HBase and Cassandra are really indexes
- For HBase:
  - Allow for multiple indexes per table
  - API extensions to make this fast
- How do you decide to use an index?
  - Most stores have no statistics

Type System

- Plugins can add scalar types
- Add functions for types
- Operator overloading
- Integrate with ordering and grouping
- Custom memory encoding

Upcoming

- Large joins and aggregations
- Structural types
- HBase connector
- IPython
- ODBC

Presto

prestodb.io
github.com/facebook/presto

Dain Sundstrom

@daindumb
github.com/dain