Using Apache Beam to power and scale a data engineering transformation at a Financial Exchange

Conall Bennett - Apache Beam Summit

Who are we & what do we do

- Worlds largest derivatives exchange - 175 years old
- Over 15.3 billion daily units of work
- Speed is King - Ultra low latency (nano second precision) for large institutional customers
- Migrating 100% of our business to GCP

CME Data engineering in GCP

- Our Data engineering pods working in GCP - typically less latency sensitive (not nanos)
- Deal with a combination of real time & delayed streaming data and batched data
  - In peak scenarios, some apps could be dealing with over 30 Billion units of work in a day
- Innovating on unique Big Data challenges
  - Volume, variety & velocity
  - Gapless data, no loss
  - Ordered, time series sequenced data
  - Balancing heavy data skew

Our legacy Big data technology ecosystem

An “eclectic” mix of data processing & streaming frameworks and data warehouses made up the majority of our data engineering tech stack

- Mix of “old” and “slightly less old”

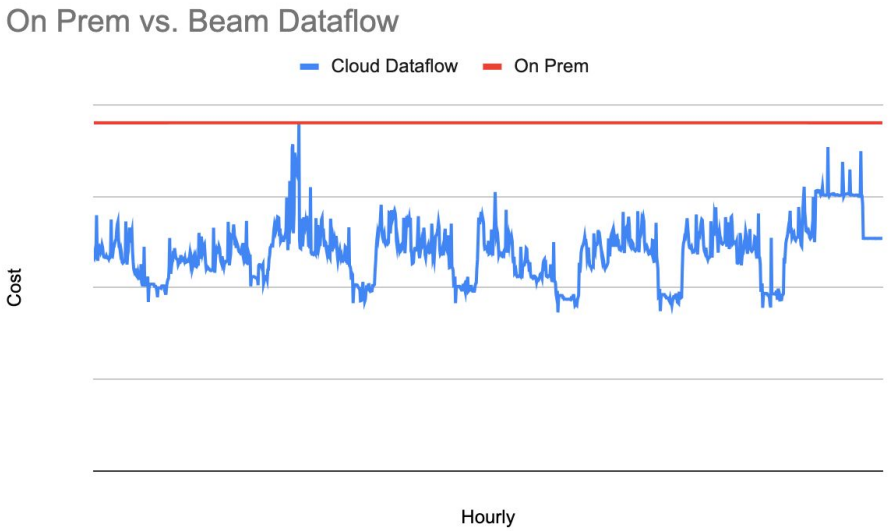
Teams operated very differently to each other, often working in their own silo’s.

- “Fragmented” approach to data engineering
- Time to market for new data products was not quick
- Lots of batch oriented architecture patterns
- Operational overheads & complexity, toil & churn
- Typically, high cost…….in retrospect

Migrating to Beam on GCP

- Apache Beam & Dataflow targeted as a key strategic data engineering service early in migration
- Whilst technology does not change culture itself, it can be an enabler & accelerator for change
  - Ad hoc & official training to support change
- First adopter teams started experimenting with Beam & Dataflow on (mainly) stateless pipelines
  - Assessed capability against well architected principles

Current usage of Beam



Conservatively we estimate approx. 40% reduction in infrastructure cost alone using Beam on Dataflow.

That doesn't include velocity of development, which has also sped up dramatically.

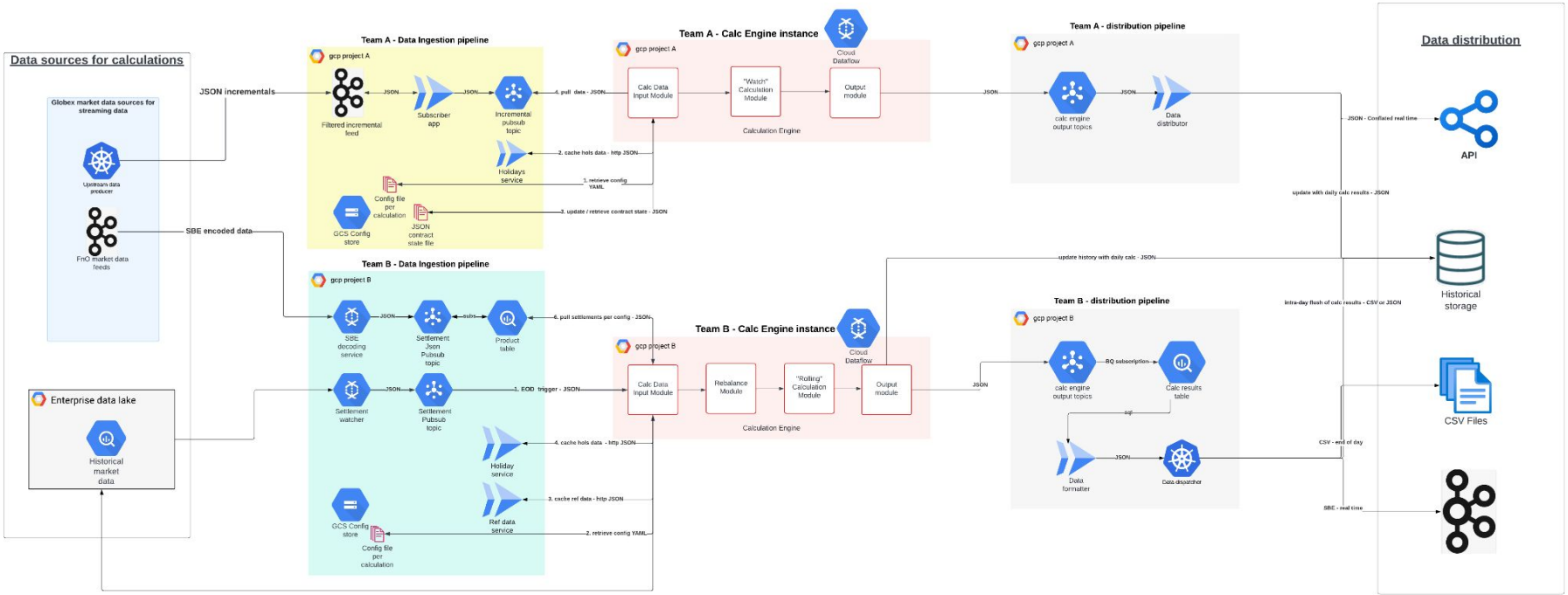
- "Domino effect" of team's adopting Beam internally as more workloads are launched to production over approx 6-12 months.
- Teams like the strong observability features, easy scaling & workload stability
- Lower infra cost & speed of development also key benefits

Beam Calculation engine framework

- Inner sourced multi-maven apache beam project, for stateful or stateless pipelines
  - Multiple teams contributing pipeline code
  - Multiple teams deploying different instances
  - Accelerator for new teams getting started with Beam
- Composable pipelines were key to rapid development and re-use
  - “Function” library of reusable calculation transforms
  - Boilerplate code for easy integration with GCP infrastructure
- Framework code resides with other library & utility code which deals with stateful operations

Don’t cross the (data) streams!

Calculation engine architecture (high level)



Calc engine dataflow architecture where composability is key. This has enabled rapid product development and release cycles for teams.

Challenges for stateful processing at high volume

- Dealing with volume, variety & velocity, where ordering & 0% data loss are important
  - e.g. creating different representations of the order book
  - Dealing with billions of messages per day, TPS can be hundreds of thousands, which require stateful processing
  - Heavy data skew, data is not evenly distributed across keys

- Determinism when using timers
  - e.g. precision control for conflation rates at millisecond intervals

- State mgmt. is expensive on performance due to state storage reads/writes
  - Use of valueState for large caches to mitigate

- Constant trade-off tug of war between dataflow vs. flink
  - performance & latency versus cost efficiency & ease of use

Current experiments with Beam

- C++ binary integration for java pipelines
  - for use case of integrating quantitative libraries into beam pipelines
  - Using subprocess execution mechanism on dataflow workers (https://github.com/apache/beam/tree/master/examples/java/src/main/java/org/apache/beam/examples/subprocess)
- YAML api usage for simpler composable pipelines
  - Lower barrier of expertise needed for prototype pipeline creation
  - Open usage to data scientists, technical product owners?
  - Could this be a better / viable interface for using quantitative libraries?
- High availability, zero downtime pipelines
- Ordered list state implementation

Thank you

conall.bennett@cmegroup.com


CME Group