Zero to Streaming

Getting to Production with Apache Flink

April 6, 2017

Strength in Numbers

MediaMath

MediaMath's technology and services help brands and their agencies drive business outcomes through programmatic marketing. We believe that good advertising is customer-centric, delivering relevant and meaningful marketing experiences across channels, formats and devices. Powered by advanced machine learning algorithms that buy, optimize and report in real time, our platform gives sophisticated marketers access to first-, second- and third-party data and trillions of digital impressions across every media channel. Clients are supported by solutions and services experts that make it simple to activate our technology. Since launching the first Demand Side Platform (DSP) in 2007, MediaMath has grown to a global company of nearly 700 employees in 15 locations in every region of the world. MediaMath's clients include all major holding companies and operating agencies as well as leading brands across top verticals.

Reporting on Flink

Next Generation Reporting Infrastructure

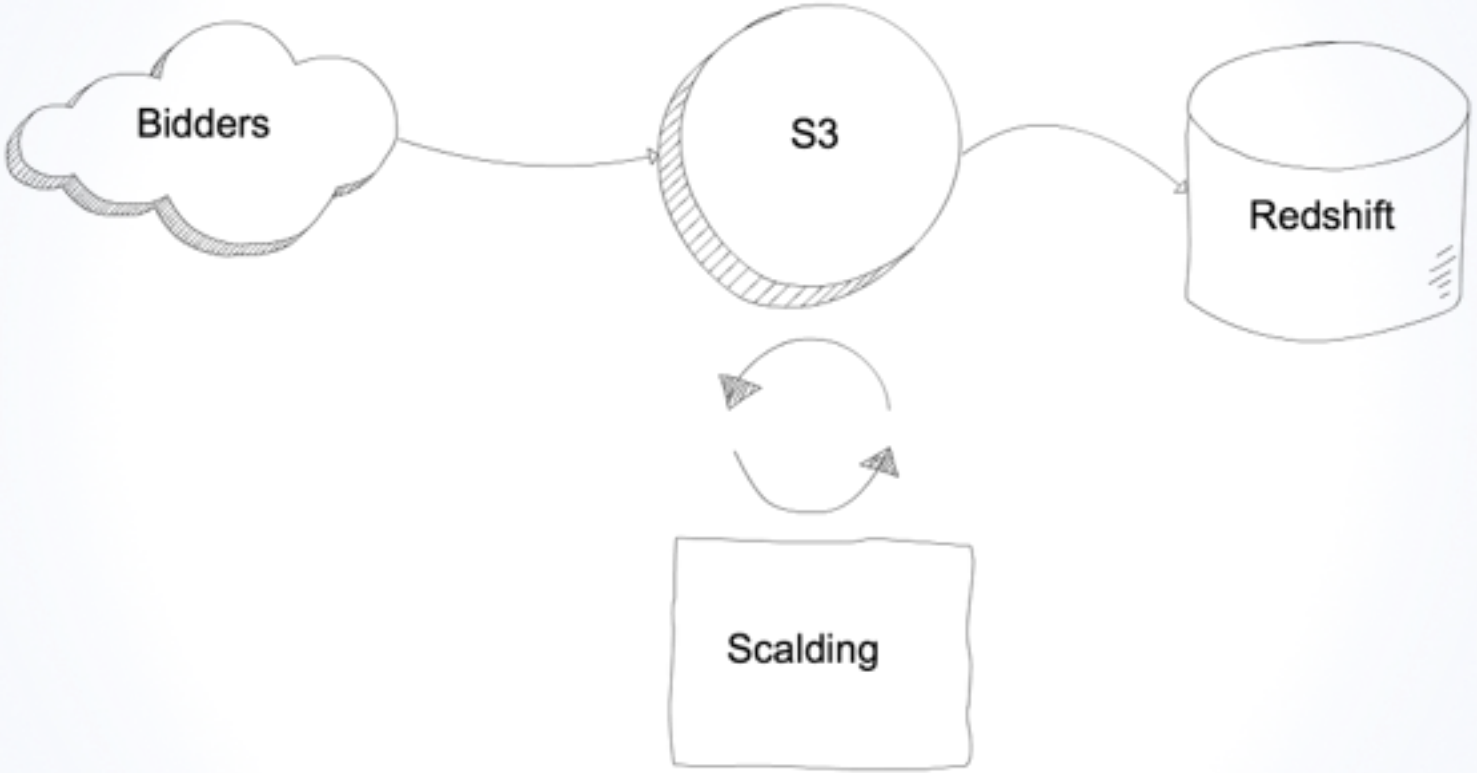
- Generate user facing reports from raw log data
  - Currently one dataset with 600 million to 1 billion records a day
  - More datasets to come soon
- Data collected from multiple datacenters around the world
- Work is done in AWS

Proof of Concept

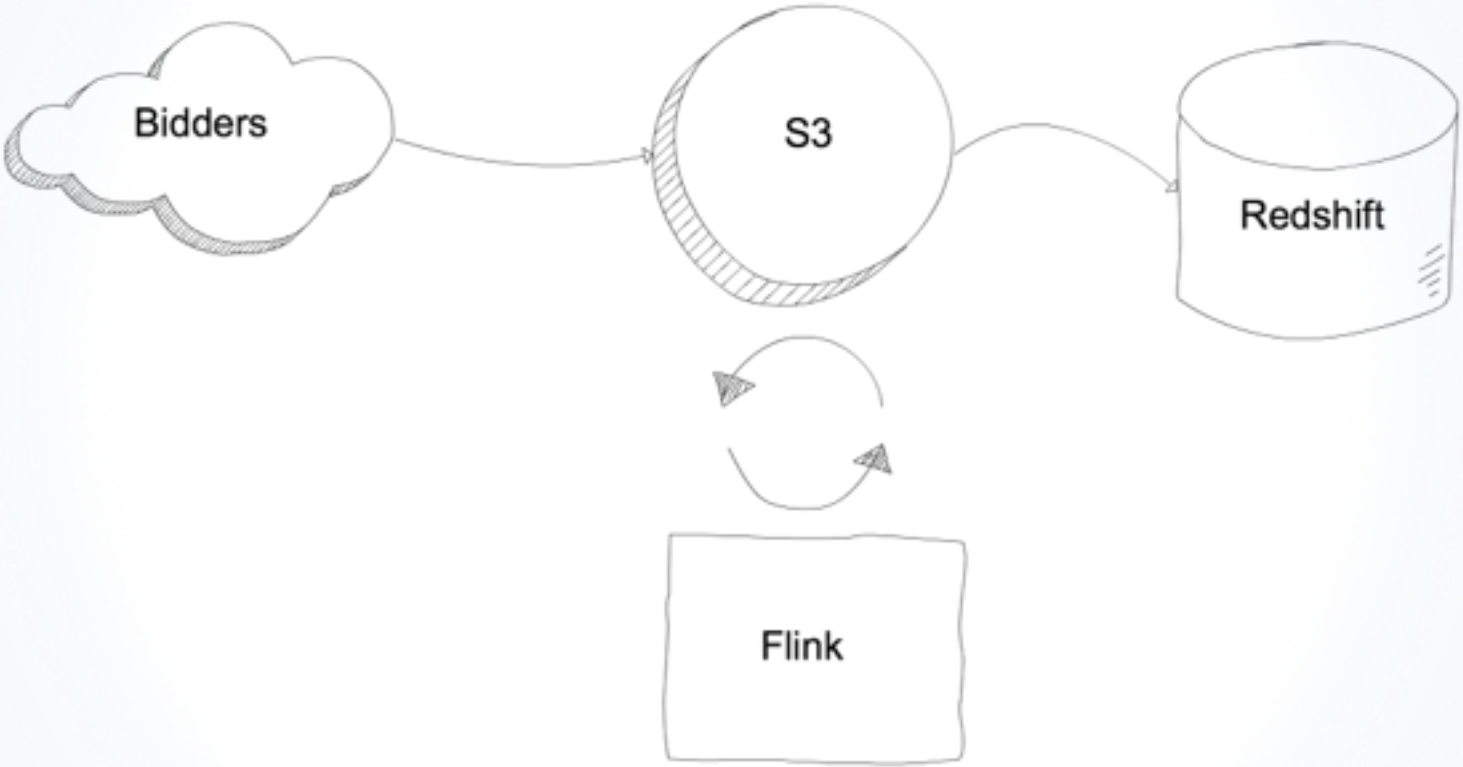
Requirements

- Not a lambda architecture
- Fit into the existing stack
- Can’t break the bank
- Must align with my sleep schedule

Current Stack



New Stack



Window Schema

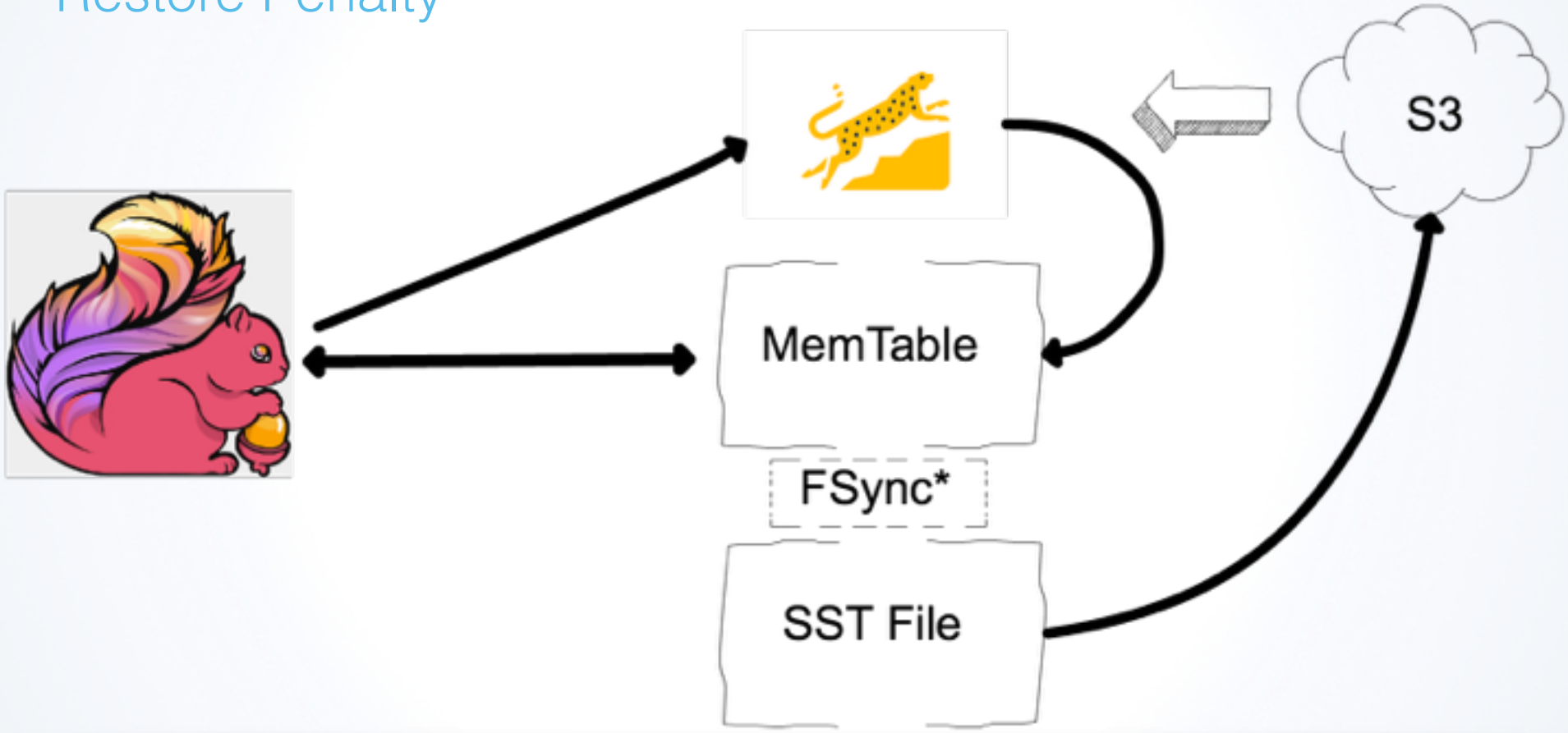
Append

- Simple to reason about
- Difficult to guarantee idempotent updates

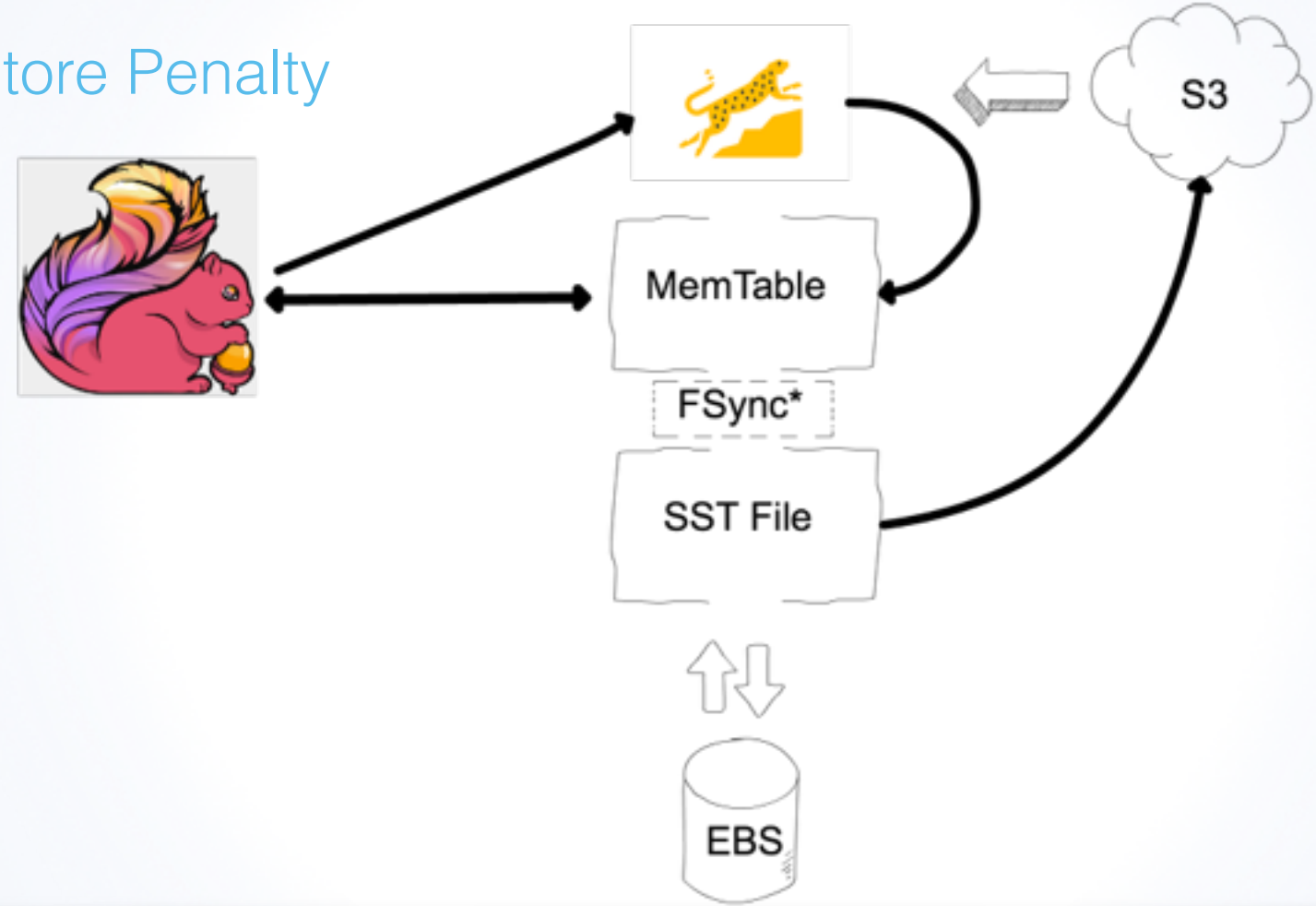
Upsert

- Better aligns with the streaming model
- Results in a larger state

Restore Penalty



Restore Penalty



S3

- S3 is not a file system
- 99.999999999% durability at the cost of eventual consistency
- What does Amazon promise?
  - read-after-write consistency for PUTS of new objects
  - eventual consistency for overwrite PUTS and DELETES

Consistent View

- EMR provides a consistent view of S3
  - DynamoDb is used as a source of truth
- Can we create our own source of truth?
- Flink is expected to be in a consistent state

Roll our own File System

- Treat Flink as a source of truth
- If Flink and S3 disagree
  - Assume S3 is inconsistent
  - Back off
  - Try again
- Works to an extent
  - S3 has no upper-bound on time to consistency

The BucketingSink and S3

```
649     LOG.debug("Moving pending files to final location for checkpoint {}", pastCheckpointId
650
651     for (String filename : pendingPaths) {
652         Path finalPath = new Path(filename);
653         Path pendingPath = getPendingPathFor(finalPath);
654
655         fs.rename(pendingPath, finalPath);
656         LOG.debug(
657             "Moving pending file {} to final location having completed checkpoint {}.",
658             pendingPath,
659             pastCheckpointId);
660     }
```

Roll our own Sink

- Treat S3 like a key-value store
- Write all files locally
- Copy to S3 once per checkpoint
- Sink writes are all or nothing

Where did the checkpoint barriers go

- Went nearly 3 months without seeing a checkpoint
- Dug into Flink's internals
- Built tools to better understand what was happening

A small number of windows were never checkpointing

Taking a step back

- We needed to relearn our dataset
  - 8 hours looks very different than 15 minutes
- We have massive data skew
- 1/3 of all elements belong to 0.3% of the key-space

Handling data skew

- Constrain the key-space
  - Flink scales with windows
- We have three keys common across all reports

Primary Key	
Secondary Key	Metrics
. . .	. . .
. . .	. . .

Handling data skew

- Reduce the number of elements moving across the network
- Can we build something similar Hadoop's combiner?
- Low level control through the SingleInputOperatorInterface

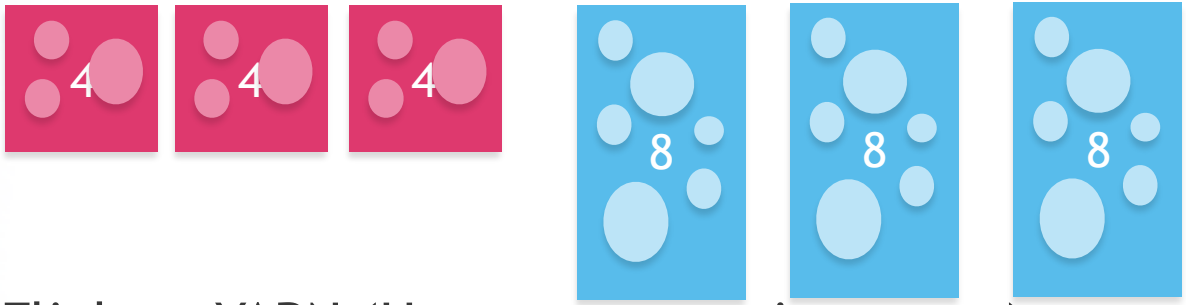
Flink in AWS

- Cluster
  - Standalone or YARN or …?
- Storage
  - HDFS or EFS or S3?
- Lease
  - Reserved, On-Demand or Spot?
  - EMR?

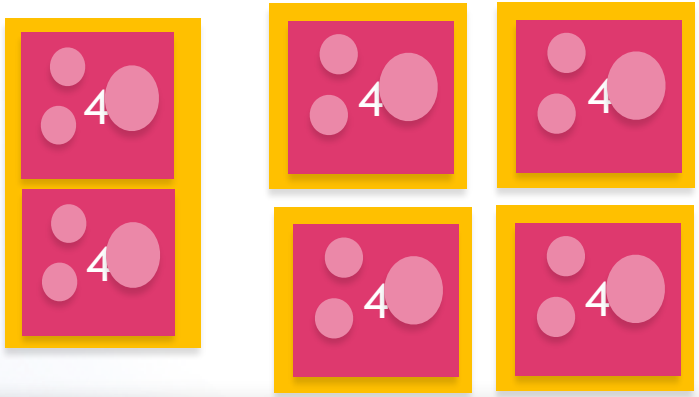
Spot rides

Why YARN?

- Flink Standalone (Homogeneous instances)



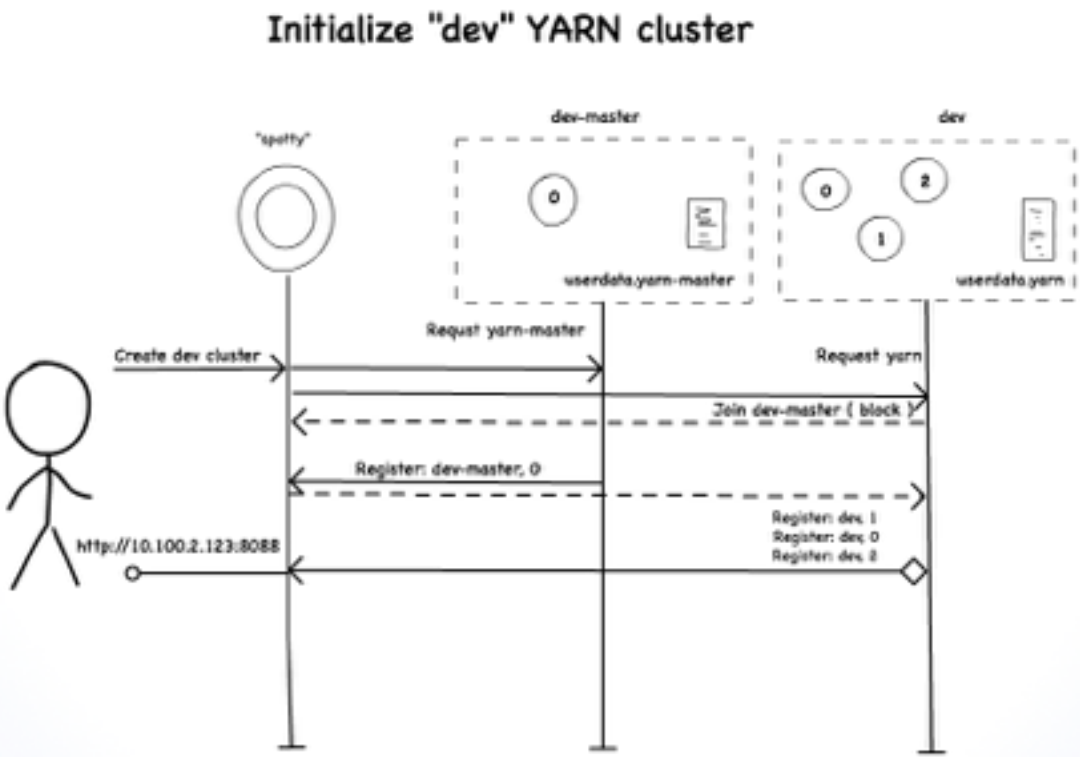
- Flink on YARN (Heterogeneous instances)



Cluster Initialization Concepts

- Cluster is defined as one or more instances of a Config, keyed by Name
- Config is “userdata”, a user supplied shell script called as root on instance init.
- Cluster “Join” – like Workflow join – means that one cluster can block on another’s init.

Spot Cluster Creation



Running Flink

- Initialized by shell process, managed, monitored with YARN REST apis
- Flink Job created and managed with Flink REST apis (poll for events).
- Job exceptions and Checkpoint events are logged to database.

Monitoring

- Health checks:
  - Use http pings and YARN REST api to verify cluster heath.
  - Expose and publish Flink job Current Low Watermark
    - Alert on Watermark lag.
      - Trick! Use jolokia:
      - javaagent:/opt/jolokia-jvm-1.3.5-agent.jar=port=0,host=0.0.0.0
  - Publish checkpoint times and durations
    - Alert on checkpoint lag.
  - collectd/graphite for machine stats (memory, load, disk).

Spotty-agent

- Watches for spot reclaim (2 minute warning).
- Posix Passthrough: remote exec scripts and shell commands (init checks, stats).

Cluster Failover

1. Agent receives warning and sends notification (first one wins).

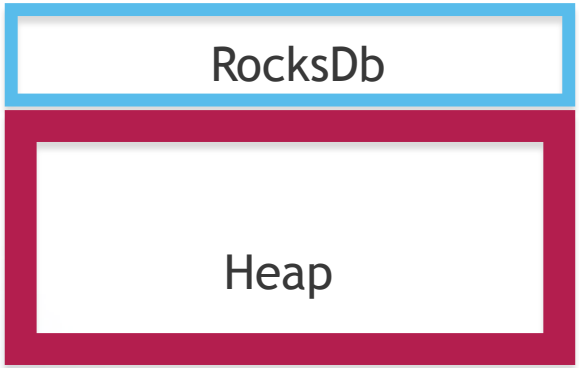
2. Monitor initiates Flink “cancel_with_savepoint”.

3. Monitor requests a whole new cluster.

4. When new cluster is up:
    1. Verify final checkpoint (may or may not be from cancel).
    2. Restart Flink Session
    3. Restart Flink job.
    4. Delete old clusters’ EC2 instances (may already be gone).

Task Manager on YARN config

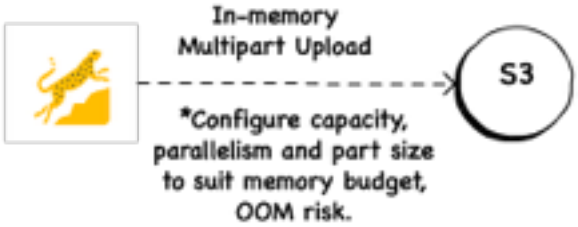
- EC2 compute (C-type) instances are appx 1.75 gig ram/VCPU. (reserving 600-700MB for linux os)
- Min YARN process = 1.75G
- TM of four slots = 4 x 1.75 = 7G
- After 5% Flink “yarn tax”, each YARN process = ~6.65G
- Set TM Heap cut-off to .25, leaving 1.6G for RocksDB



RocksDB Snapshot to S3

- S3N and S3A default:

- S3A Fast Upload:



- Hadoop 3.0 code looks very promising!

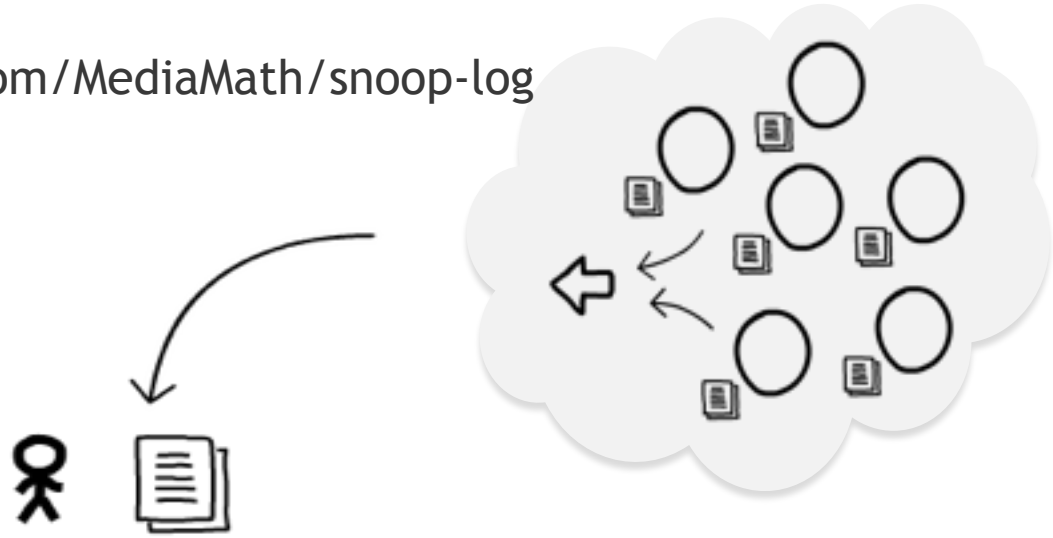
Dev Tools

- Live metrics
  - Provide context to checkpoint and other pipeline problems
- Remote debugging
  - Effective, but limited scope
- Cluster Logging?
  - Continuous process – cannot use YARN log aggregation
  - Many (100s) of slots – each with own operator DAG
  - Spot Instances – log aggregators not easy to integrate, overkill for problem

Snoop “Loggy” Log

Tail your ephemeral cluster ephemerally!

https://github.com/MediaMath/snoop-log



flink/lib/logback-snoop.jar

Hacking Flink

- (demo)

THANK YOU!

Seth Wiesman

Cliff Resnick

4 World Trade Center, 45th Floor

New York, NY 10007

MediaMath

Strength in Numbers