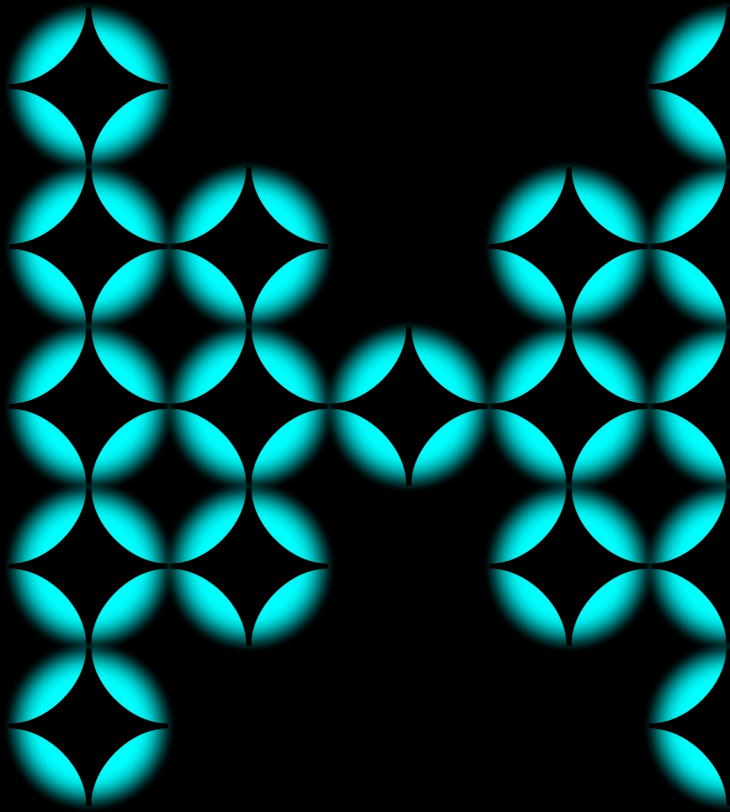
HAZELCAST

Stream Processing Fundamentals

Randy May, Industry Solutions Advocate

March 2023



What You'll Learn

- The Goal: learn enough to feel comfortable tackling stream processing problems with Hazelcast
- You'll build a solution that processes an event stream and reacts in real-time.
- First run it on your laptop then scale it by deploying to Viridian
- Those who complete the workshop and deploy a Viridian cluster will earn a badge

Prerequisites

A Laptop Set Up for Java Development Including …

- Java 11+
- Maven
- Docker Desktop
- Git
- A GitHub Account

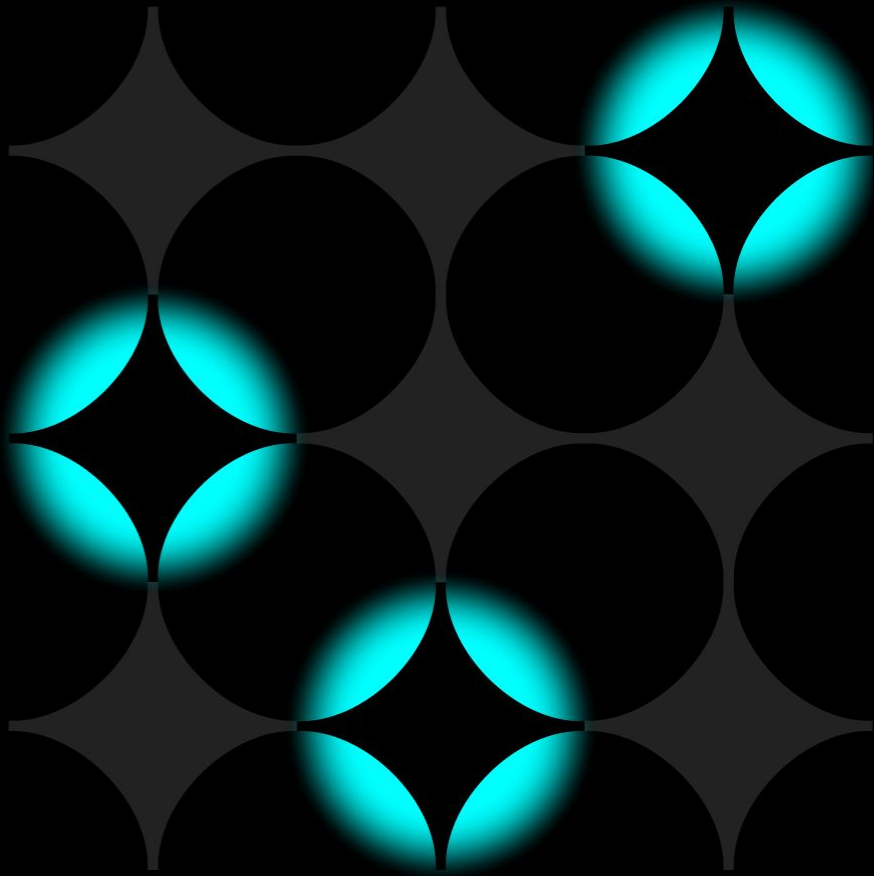
HAZELCAST

Agenda

Introduction to the Pipeline API	4:00 - 4:15
Hand On Exercise	4:15 - 5:15
Deploy to Viridian	5:15 - 5:30

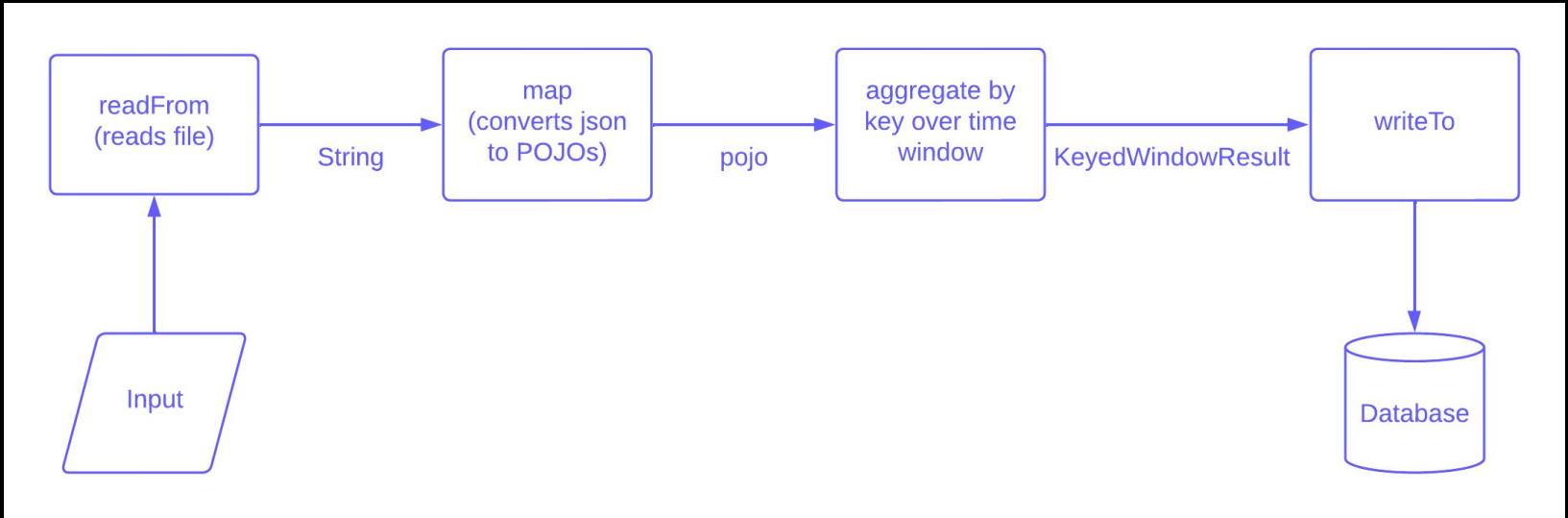
HAZELCAST

Let’s Get Going …

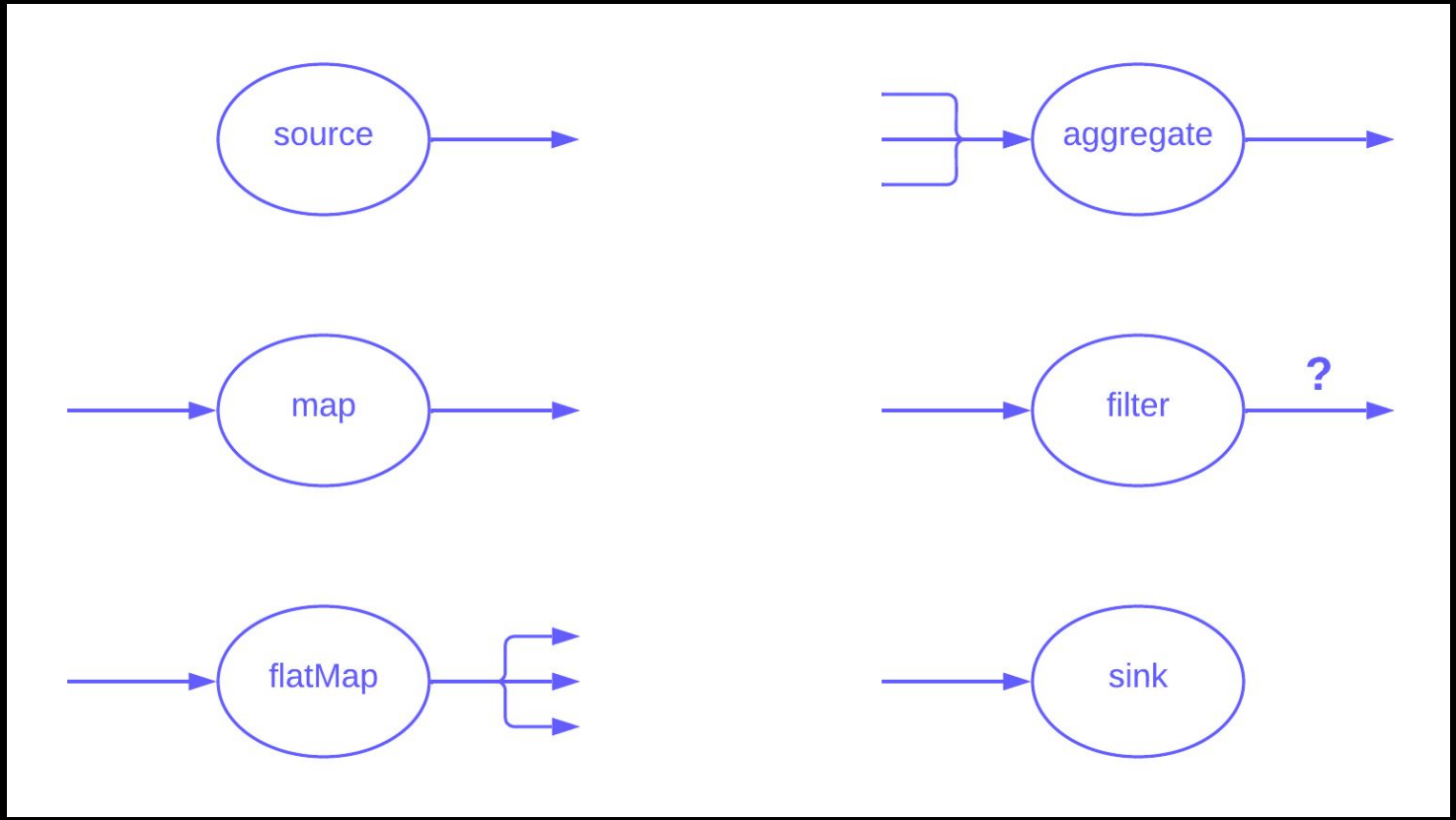


Hazelcast Pipelines

- A *Pipeline is* series of *tasklets* connected in a directed, acyclic graph.
- Events flow along the vertices of your Pipeline
- Events can be of any type
- Consider the simple Pipeline below



Pipeline Primitives



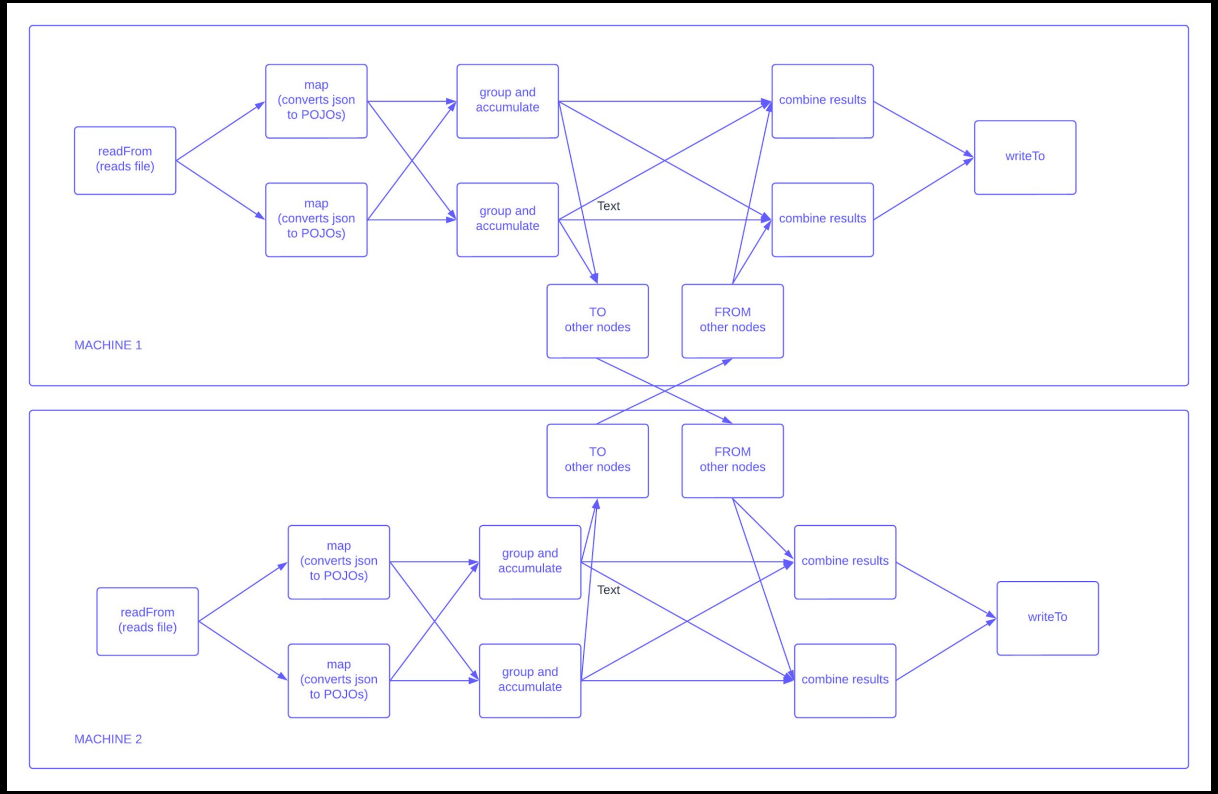
Hazelcast Pipelines

- You define the pipeline
- The Hazelcast platform runs it
- Your stream becomes …
  - Parallelized
  - Location Aware
  - Fault Tolerant, Upgradeable and Restartable
- Scale by adding nodes

Hazelcast Pipelines

How the pipeline would be deployed on a cluster of 2 machines, each with 2 cores.

Much more detail is available here: https://docs.hazelcast.com/hazelcast/latest/architecture/distributed-computing

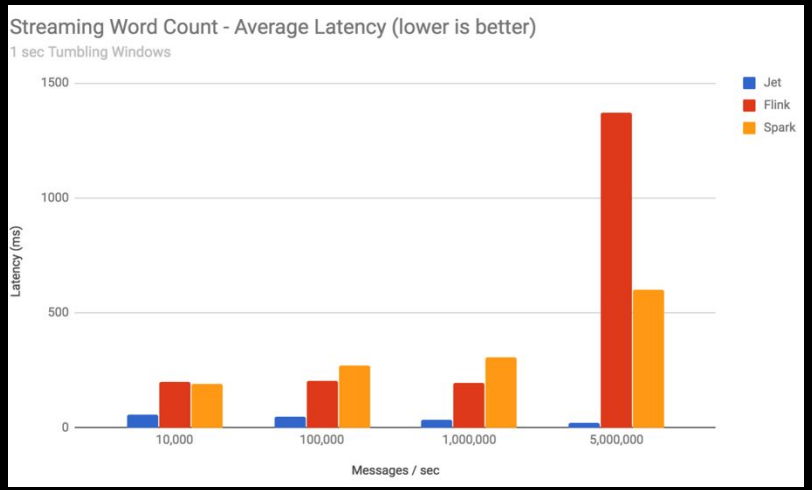


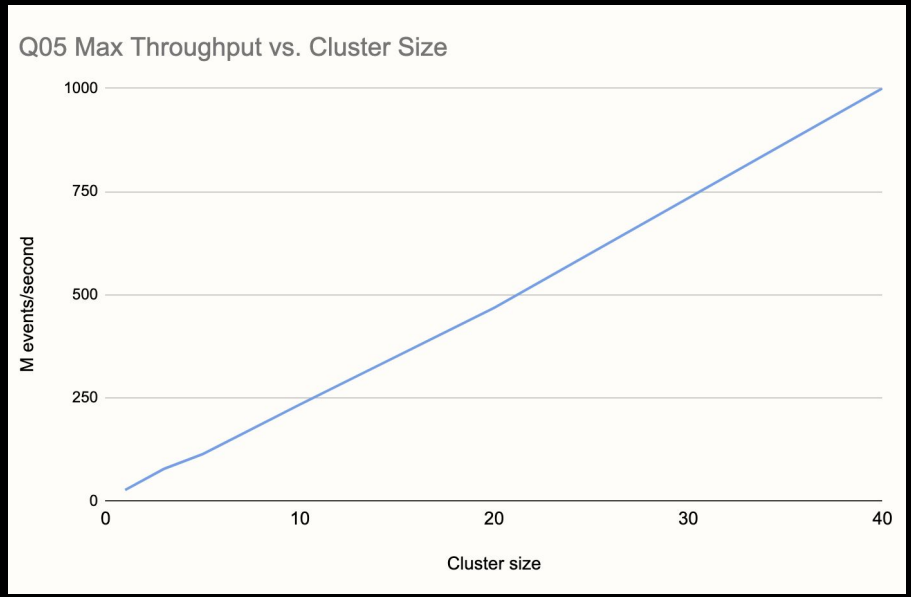
Why Learn The Pipeline API ?

1 Billion TPS with 99% under 30ms latency

- Keep-in-mind – the 100ms threshold from the Google Framework
- 45 nodes
- Linear scaling with predictable latency

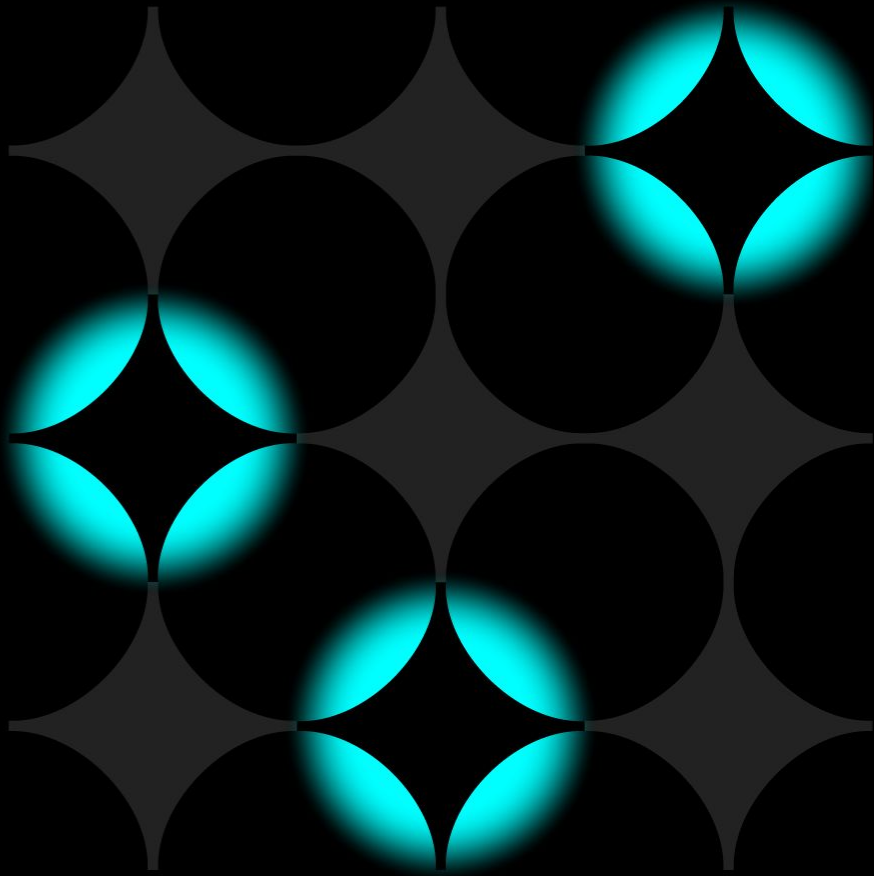
Link: Gigascale performance





HAZELCAST

The Pipeline API



Example "Read From"

Most methods return a typed StreamStage. The type parameters indicate what objects will flow along that vertex

```
Pipeline pipeline = Pipeline.create();

StreamStage<Map.Entry<String, String>> requests =
    pipeline.readFrom(
        Sources.<String, Event>mapJournal(
            REQUEST_MAP_NAME,
            JournalInitialPosition.START_FROM_CURRENT))
             .withTimestamps( event -> event.getEventTime());
```

Many methods take a functions as arguments. Often this is done with lambda expressions.

Example: adding labels using “map”

Input: (“abc”, 99.0)
Output: (“abc”, 99.0, “RED”)

Tuples are a useful helper

```
StreamStage<Tuple2<String,Integer>> temps;
StreamStage<Tuple3<String, Integer, String>> labeledTemps =
        temps.map(t -> Tuple3.tuple3(t.f0(),
                                     t.f1(),
                                     t.f1() > 90 ? "RED" : "GREEN"));
```

Lambdas again

Example: mapUsingService

Use when there is some external resource or connection that should not be re-created for each event (you'll need this)

```
StreamStage<MyPojo> stage;

ServiceFactory<?, ObjectMapper> objectMapperServiceFactory =
        ServiceFactories.nonSharedService(ctx -> new ObjectMapper());


StreamStage<String> json =
    stage.mapUsingService(objectMapperServiceFactory,
            (mapper, pojo) -> mapper.writeValueAsString(pojo));
```

ObjectMapper is created only once

First arg is the "service", second is the event to process

Example: windows and aggregation

For each machine, calculate the average temperature over a 10s “tumbling window”

```
StreamStage<MachineStatus> statusEvents;

StreamStage<KeyedWindowResult<String, Double>> averageTemps =
    statusEvents.groupingKey(s -> s.getSerialNum())
        .window(WindowDefinition.tumbling(10000))
        .aggregate(AggregateOperations.averagingLong( s -> s.getTemp()));
```

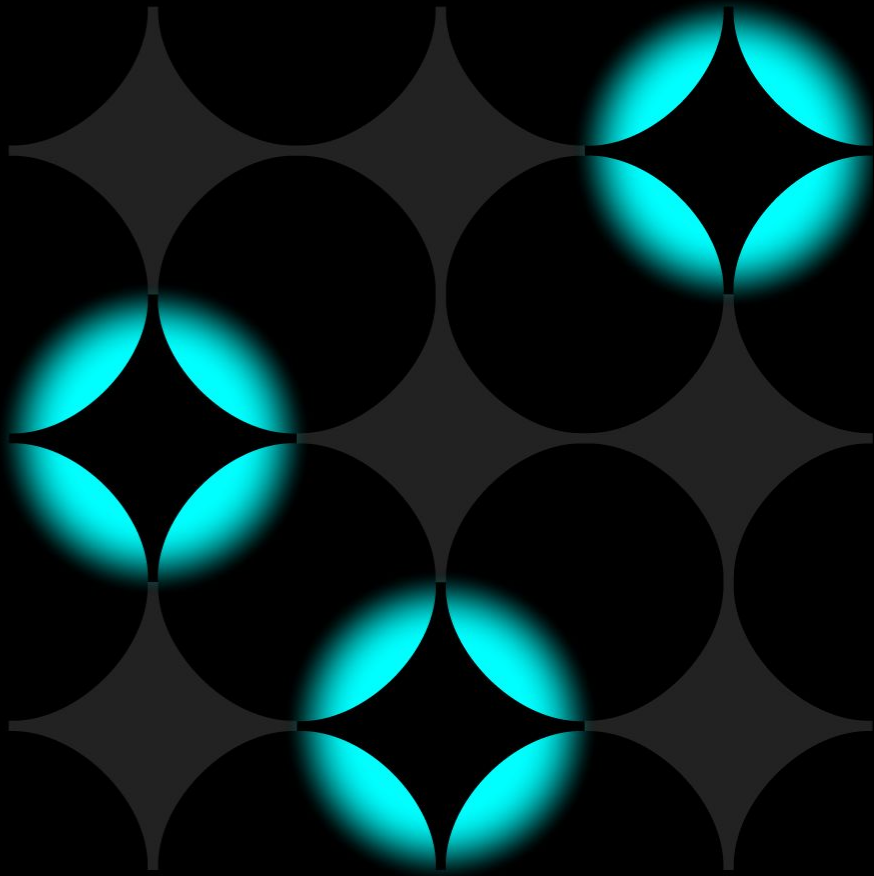
Important! Specify the “group by” key

Window type and size

What and how to aggregate

HAZELCAST

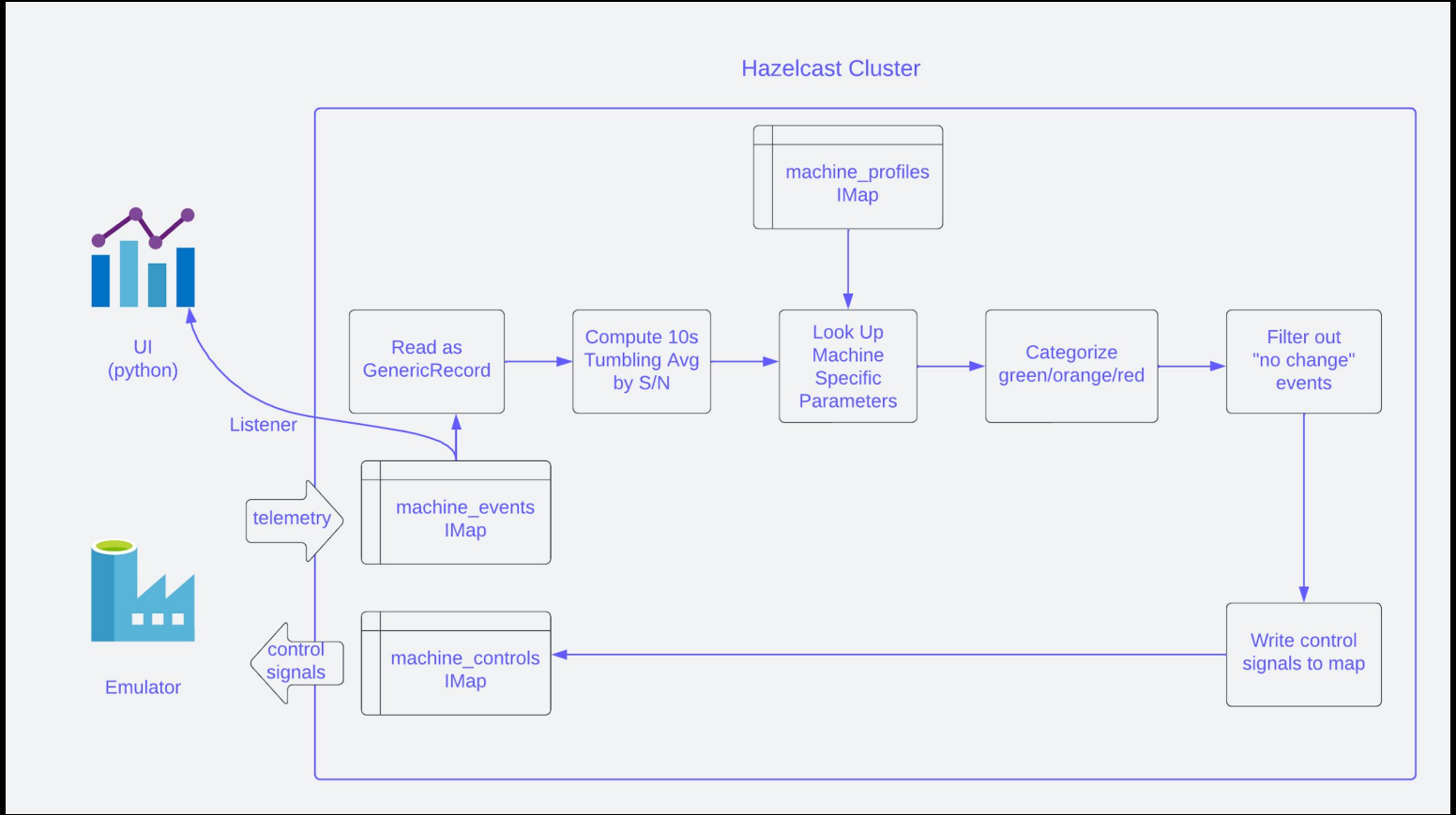
The Lab



The Scenario

- Company operates 1000’s of machines. Each publishes several data points each second. Measurements include things like bit temperature and RPM.
- Breakage is expensive. We want to go beyond maintenance schedules and monitor all of the information in real time.
- Each machine has its own parameters for acceptable bit temperature, which is stored in a “machine profile”
- If excessive bit temperatures are caught in time, breakage can be averted by immediately reducing the cutting speed

Pipeline for the Lab



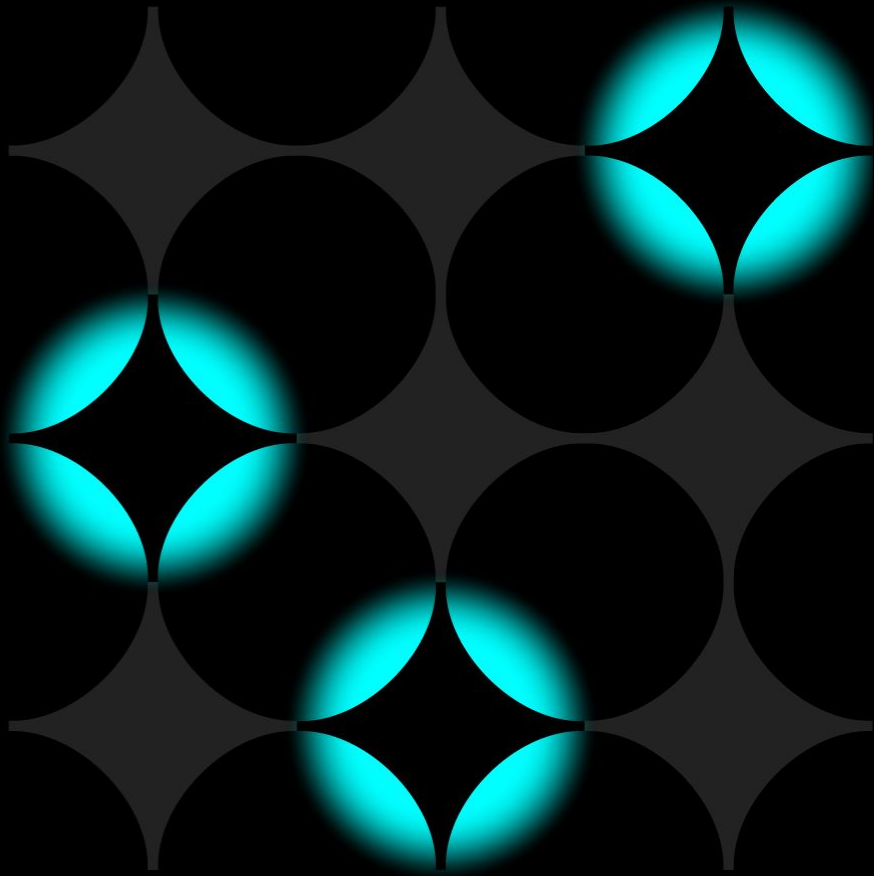
HAZELCAST

Go Forth and Process Events!

- git clone https://github.com/hazelcast-guides/stream-processing-fundamentals
- OR, if you've previously cloned the repo, **don't forget to "git pull"**
- Instructions and more detail are in the README
- Talk to the lab assistants!
- Feel free to reference the solution.

HAZELCAST

The Lab: 1 Hour

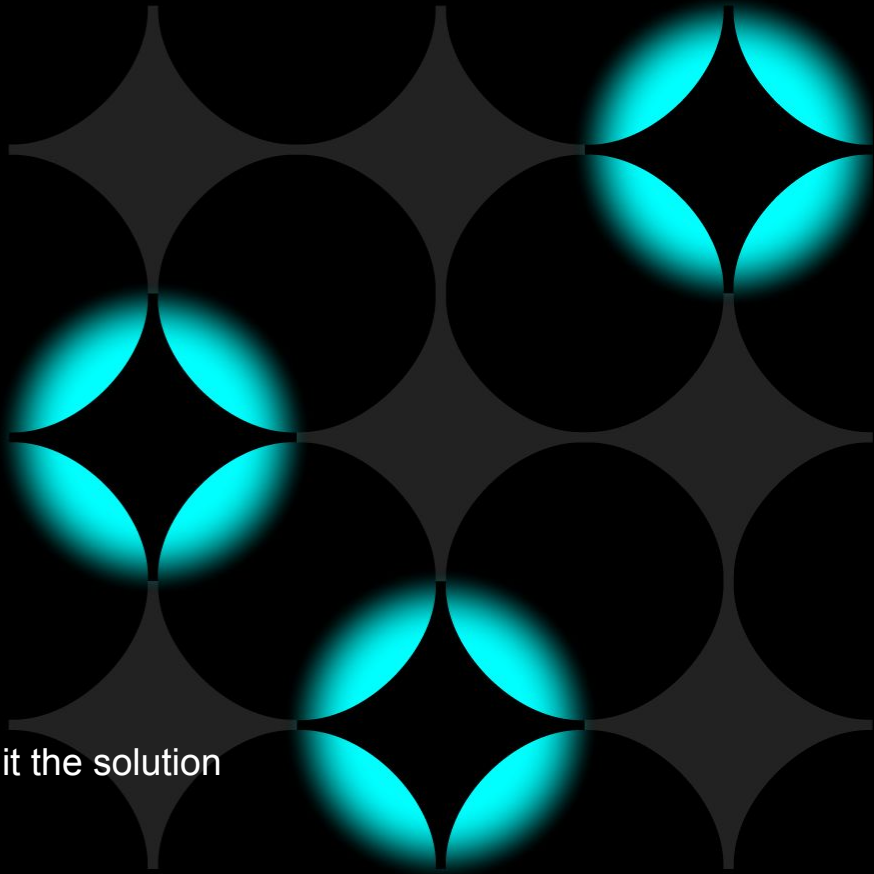


HAZELCAST

Deploy to Viridian

15 minutes

Note: if necessary, edit cli/viridian_submitjob.sh to submit the solution



HAZELCAST

Thank You !

Please help us make this workshop better by filling out the online survey

https://www.surveymonkey.com/r/hzworkshop

Quick Survey

HAZELCAST

How Do I Get Involved ?

JOIN

slack.hazelcast.com

CONTRIBUTE

https://github.com/hazelcast/hazelcast

Networking

Food

Please Stick Around !

Drinks

Ninetendo Switch giveaway

Round Table Discussion

