# Embedded Time Series Storage: A Cookbook

**Andrey Pechkurov**

# About me

- Java (for a long time) and Node.js (for quite a long time) developer
- Node.js core collaborator
- Interests: web, system architecture, distributed systems, performance
- Can be found here:
  - https://twitter.com/AndreyPechkurov
  - https://github.com/puzpuzpuz
  - https://medium.com/@apechkurov


- Hazelcast IMDG Management Center (MC)
- Monitoring & management application for IMDG clusters
- Supports stand-alone and servlet container deployment
- Self-contained application, i.e. .jar file and Java is everything you need to run MC
- Frontend part is built with TypeScript, React and Redux
- Backend part is built with Java, Spring and IMDG Java client

# Agenda

- The problem
- Considered options
- Decisions made
- Current results
- Further plans

# The problem

# Terminology

**Metric** - a numerical value that can be measured at particular time and has a real world meaning. Examples: CPU load, used heap memory. Characterized by name and a set of tags.

**Data point** - a metric value measured at the given time. Characterized by metric, timestamp (Unix time) and a value.

# What we mean by "time series"

"Time series" (TS) stand for series of metric data points

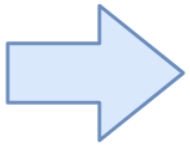
```
class DataPoint {

    String metric;
    List<Map.Entry<String, String>> tags;
    long time;
    long value;

}
```

# Sample data point

| metric | tags | time | value |
| --- | --- | --- | --- |
| map.totalGetLatency | name=test-map,unit=MS | 1532689094000 | 10234 |

# Simple math

10 members
6,000 metrics per each
3 sec interval



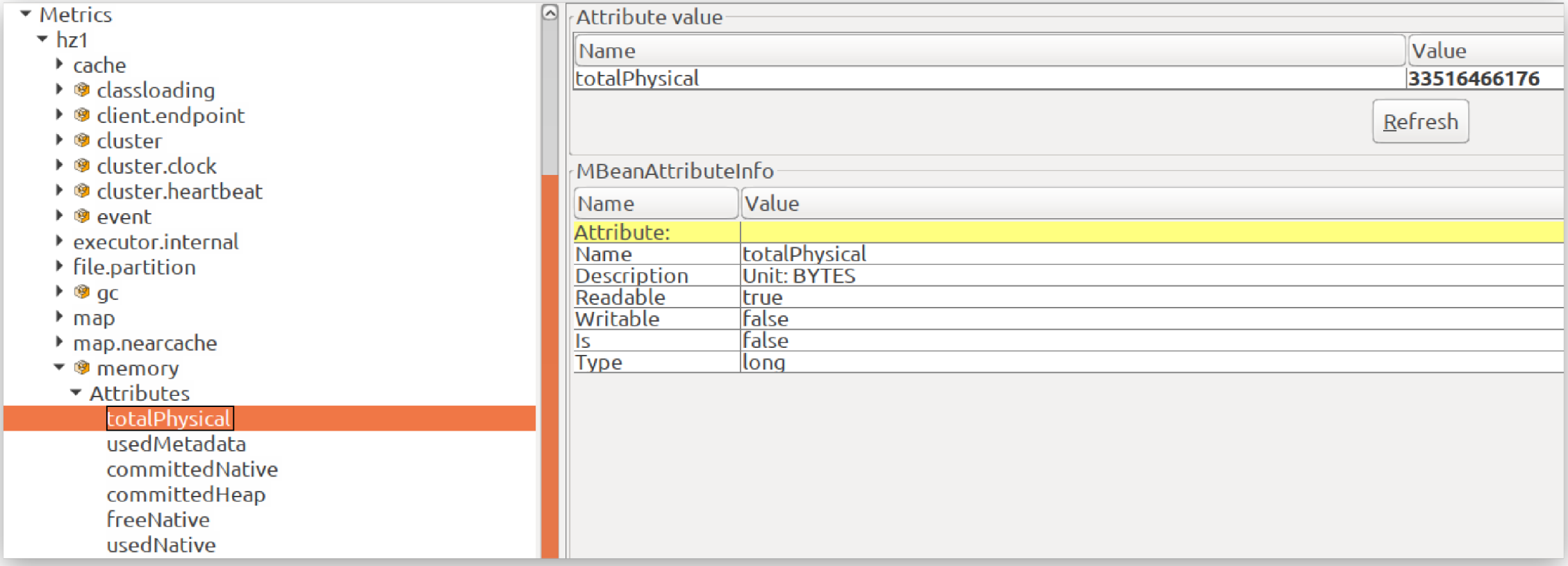
20K data points per sec
1,728M data points per day
27.6GB of raw data (time + values)

# The problem

- In the past IMDG clusters were reporting their metrics as a large JSON object
- MC was storing collected JSONs into a key-value storage (in-memory and/or JDBM)
- Such approach has some downsides that are critical for us
- Say, it requires changes in many places when we had to add new metrics

# The solution

IMDG v4.0+ is capable of reporting collected metrics (probes) to MC in a generic manner



# The challenge

- MC has to store those metrics somehow
- Thus, we need a Time Series Storage
- Here comes the challenge...

# Requirements - must haves

- Embedded time series database or storage
- In-memory and optional persistent modes
- Data compression to achieve low disk footprint in the persistent mode
- Support data retention to avoid running of disk space
- Durability and fault tolerance in the persistent mode

# Requirements - nice to haves

- Good write performance (100Ks data points/second on average HW)
- Good enough read performance (10Ks data points/second on average HW)
- Use existing stable SW, when possible

# Considered options

# TS DBs

- OpenTSDB
- InfluxDB
- TimescaleDB
- Prometheus
- Kdb+
- Graphite
- etc.

# Embedded TS DBs/storages

- Akumuli (C++) - https://akumuli.org
- QuestDB (Java) - https://www.questdb.io

## Call to Action

- You may want to give a try with IMDG and MC: https://hazelcast.org/
- Open source contributions are welcome as well!

# Спасибо за внимание!

## Helpful links

- TBD
- https://blog.timescale.com/blog/time-series-compression-algorithms-explained/