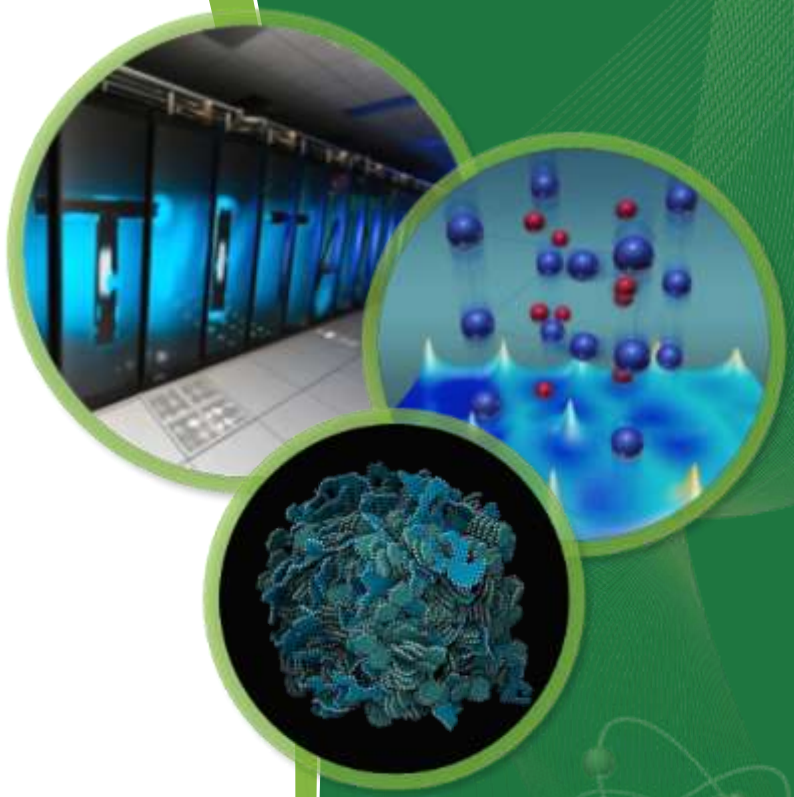
Alarm System Update based on Apache Kafka

June 2018



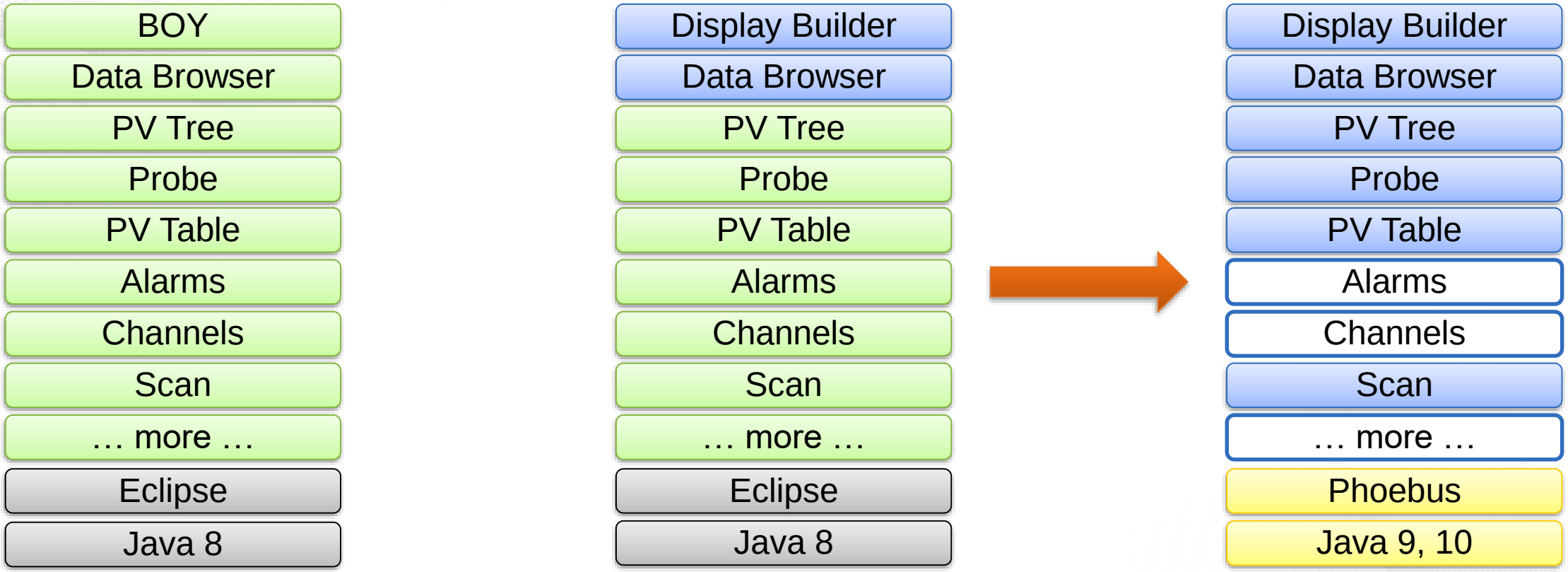
Kay Kasemir

Evan Smith

OAK RIDGE National Laboratory

Evolution of CS-Studio

JavaFX SWT



Since ~2010:
Operational at several sites

Since ~2016:
SNS beam lines, planned for ESS

End of 2018 Goal:
Some SNS beam lines

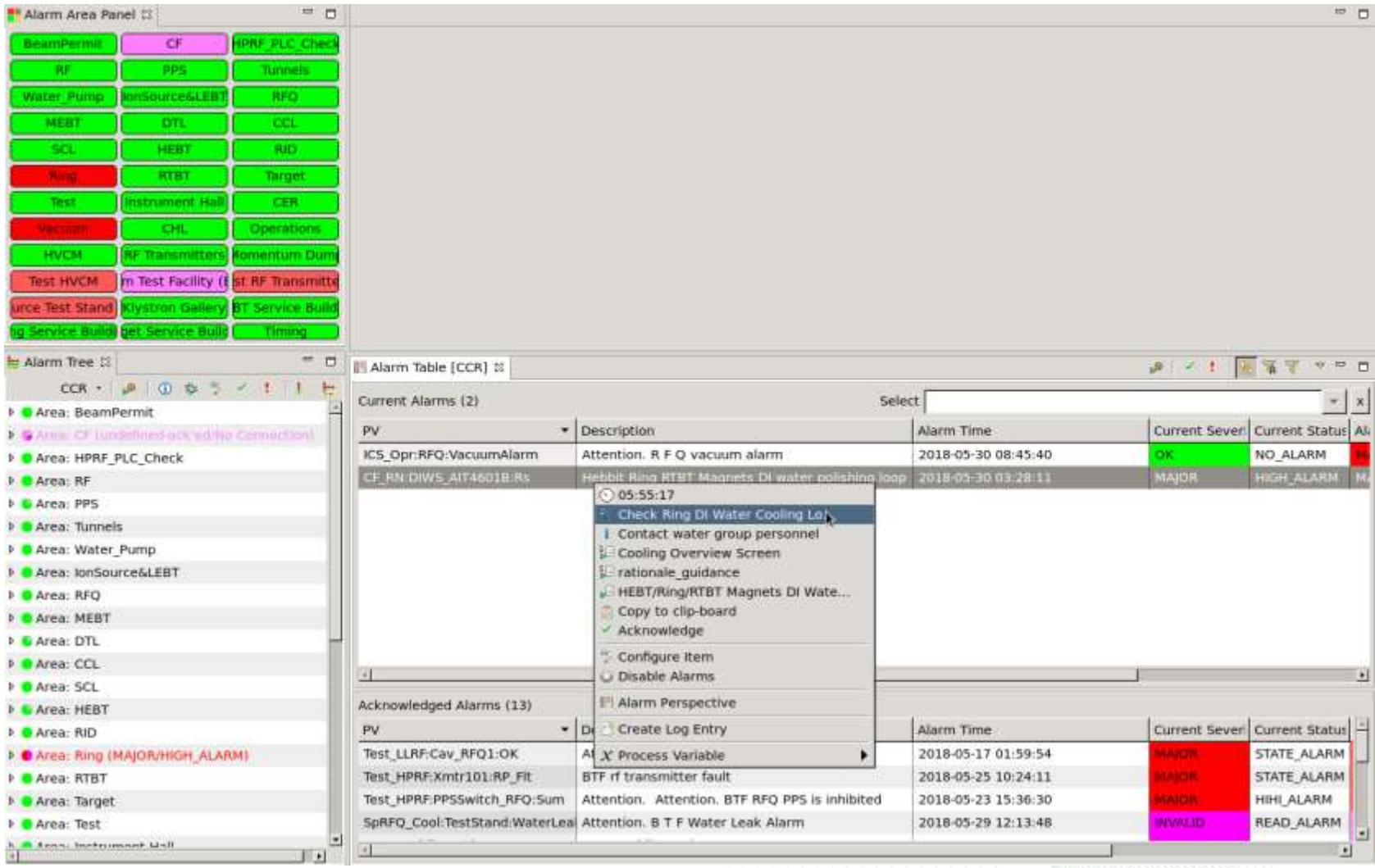
Alarm System

Alarm Server

- Monitor PVs
- Track alarms, acknowledgement

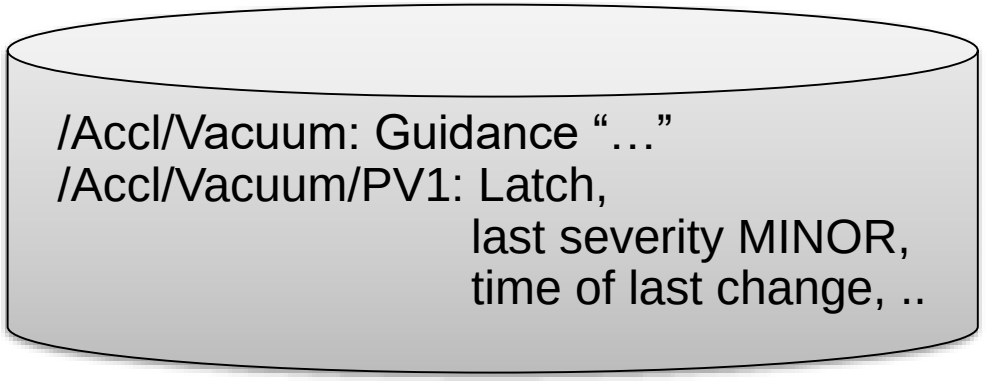
Alarm UI

- Configure (add PVs, guidance, links to displays)
- Show current state
- Acknowledge alarms
- Open guidance, related displays, email, log, ..



Current Architecture

- RDB
  - Configuration
  - Snapshot of last known state
- JMS
  - Acknowledge requests
  - State updates
  - Notification of configuration updates



/Accl/Vacuum/PV1: Acknowledged

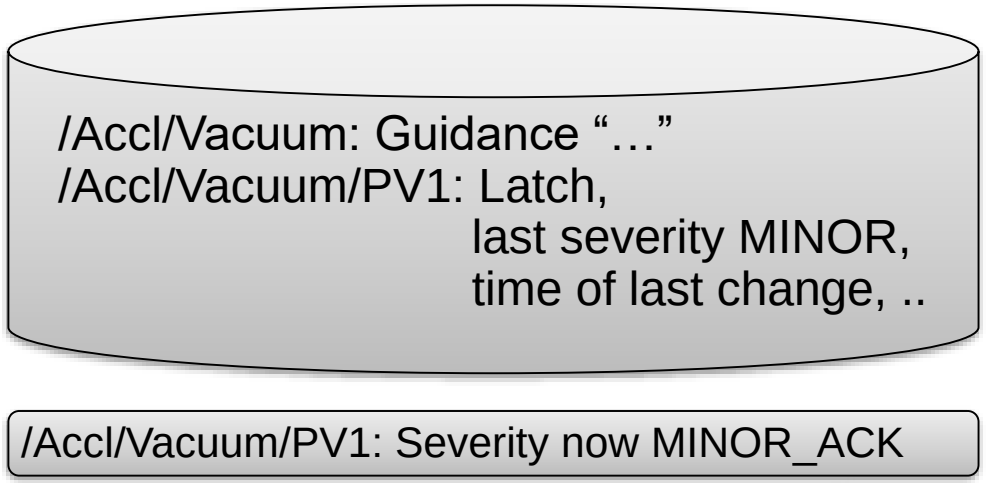
/Accl/Vacuum/PV1: Severity now MINOR_ACK

/Accl/Vacuum: Guidance was changed

Alarms: From RDB/JMS to Kafka

Clients need

- Config and initial state: RDB
- state updates: JMS

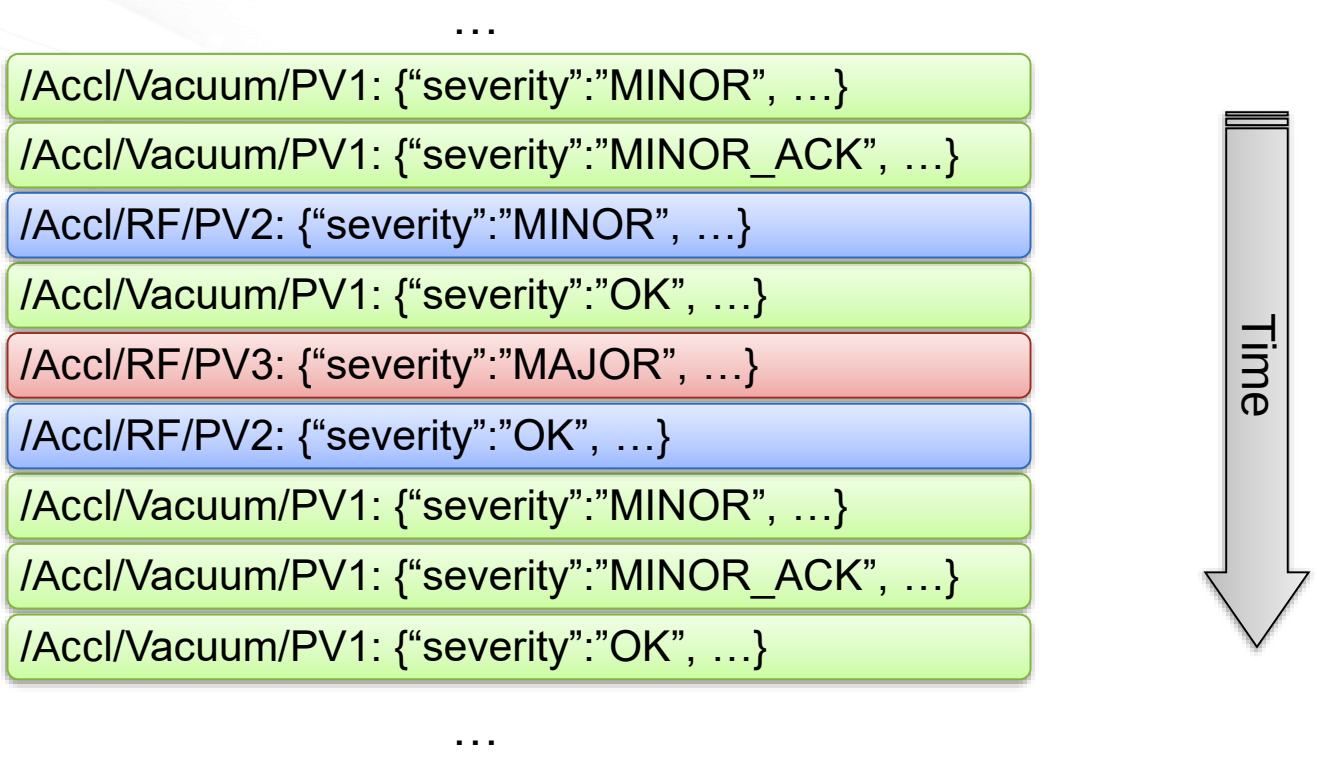


➔ Do all that in Kafka?!

- Topic 'Accelerator' for configuration
  - /Accl/Vacuum: {"guidance": …}
  - /Accl/Vacuum/PV1: {"latch":true, …}
- Topic 'AcceleratorState' for state
  - /Accl/Vacuum/PV1: {"severity":"MINOR", …}
  - /Accl/Vacuum/PV1: {"severity":"MINOR_ACK", …}

Kafka stream for Alarms, ‘AcceleratorState’

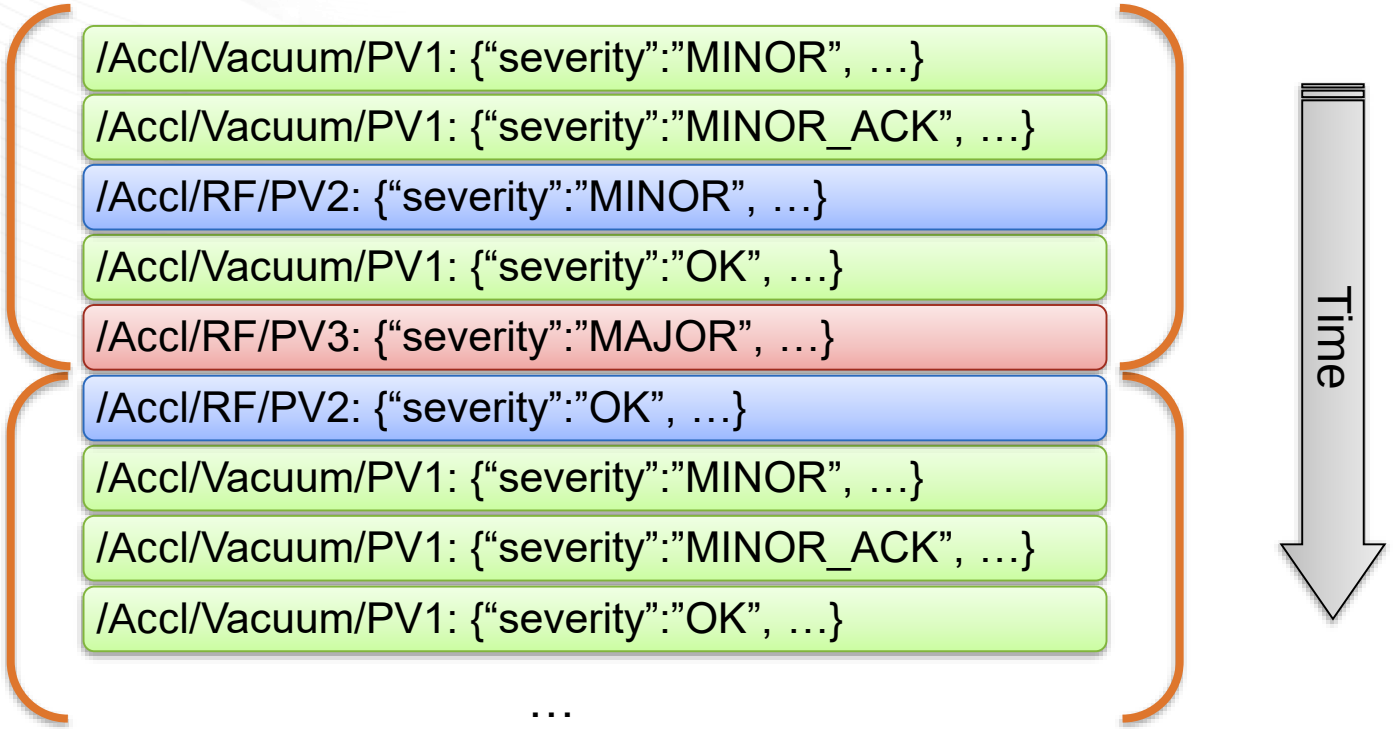
- From beginning of time, never ending



Clients tend to only need the most recent state for each item, followed by updates

Kafka Storage of ‘AcceleratorState’…

New ‘segment’, e.g. every 10 seconds..

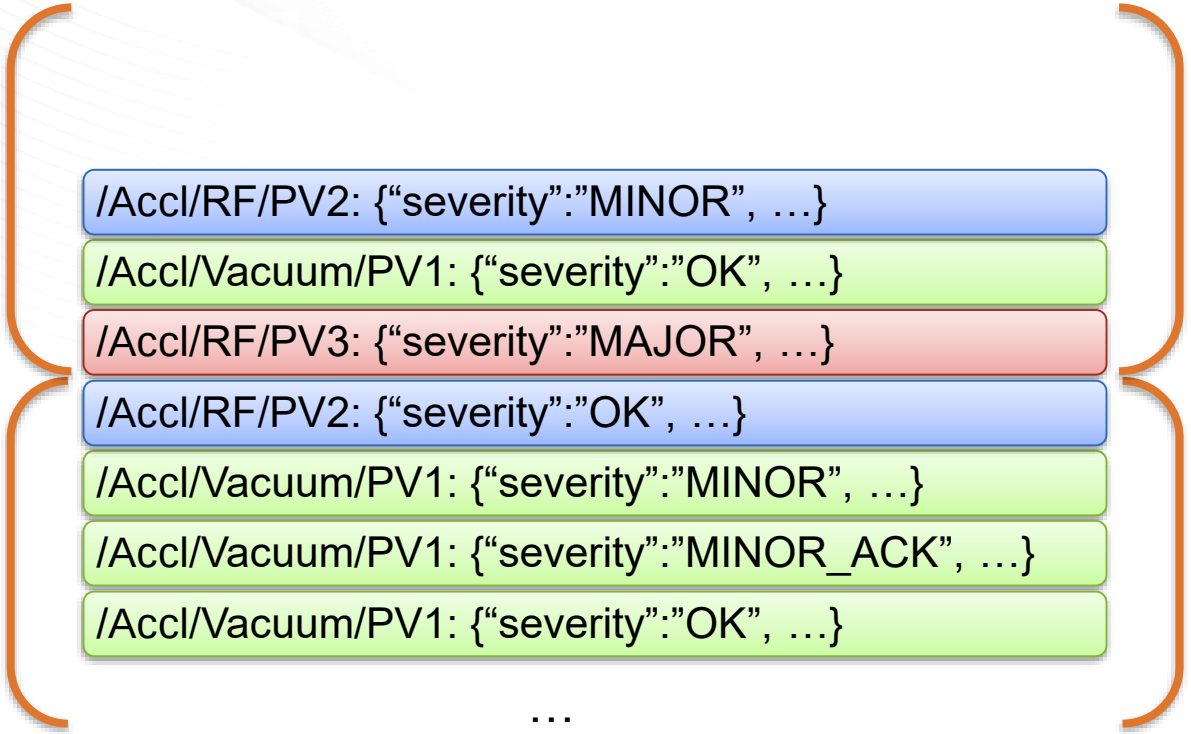


Simply delete older segments?

Then state of PV3 is lost..

‘Compacted’ Kafka ‘AcceleratorState’

Keeping *single* older segment w/ *last value* for each item



New clients get at least
the most recent state for each item,
maybe a few more,
followed by updates

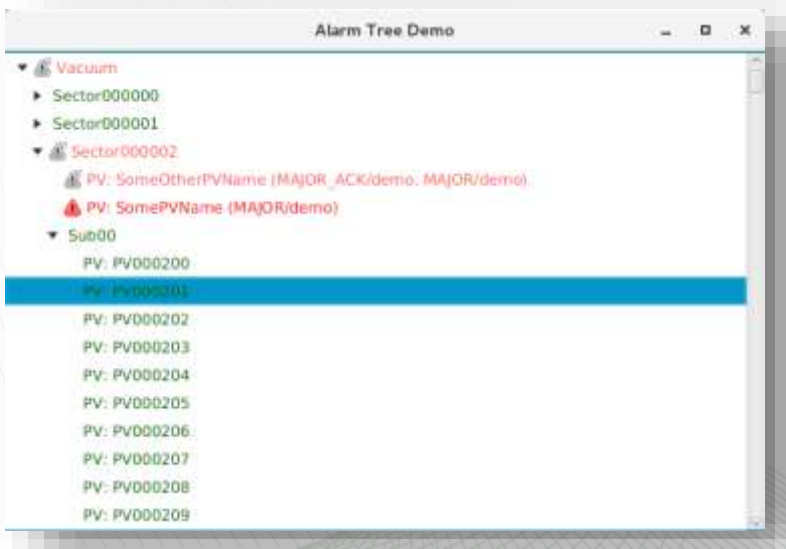
Initial Tests

2010 Test (PosgreSQL, JMS)

- Load hierarchy with 50000 PVs into RDB
  - 5 minutes
- Show config in new Alarm Tree
  - Nothing shown until all loaded after 30 seconds
- Handle Alarm Updates
  - 10 per second

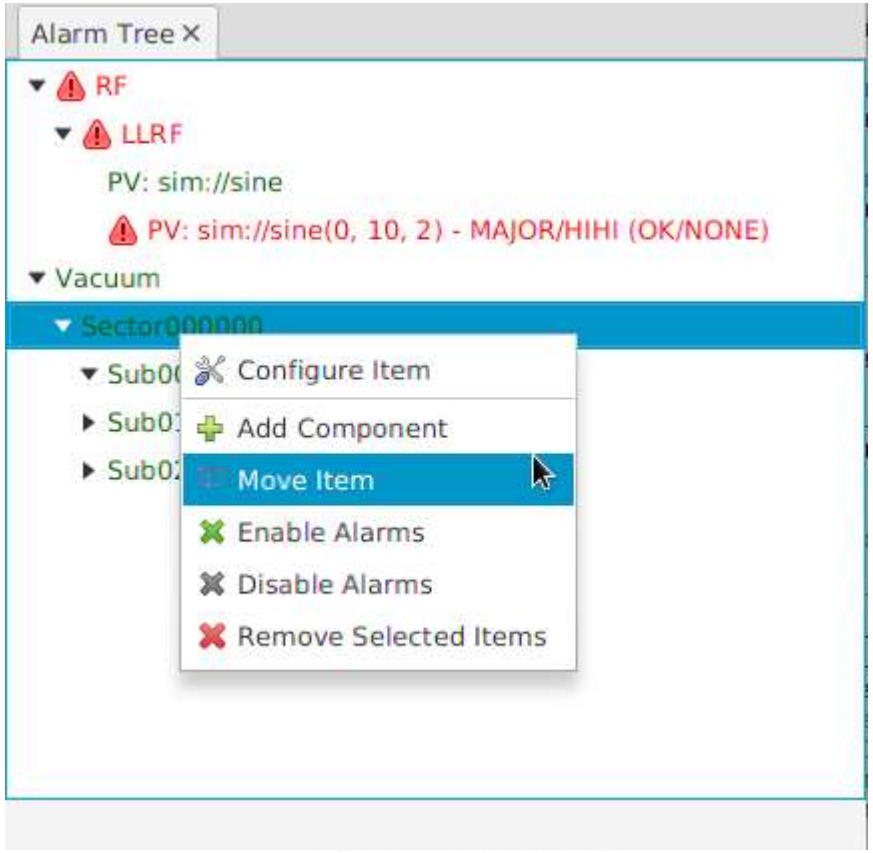
2018 Test (Kafka)

- 100000 PVs into Kafka
  - 3 seconds
  - Shows growing tree for 10 seconds
  - 500 per second



State of Development

- Functional
  - Alarm Server
  - Import/Export of XML-based configuration
    - Same format as before
  - Alarm Tree
    - Configure, monitor, acknowledge, ..
  - Alarm Table
    - Primary operator view of current alarms
  - Area Panel
    - For cross-the-room status check
- To Do
  - Annunciator
  - Authentication for changes



Further Ideas for Apache Kafka - based Alarm System

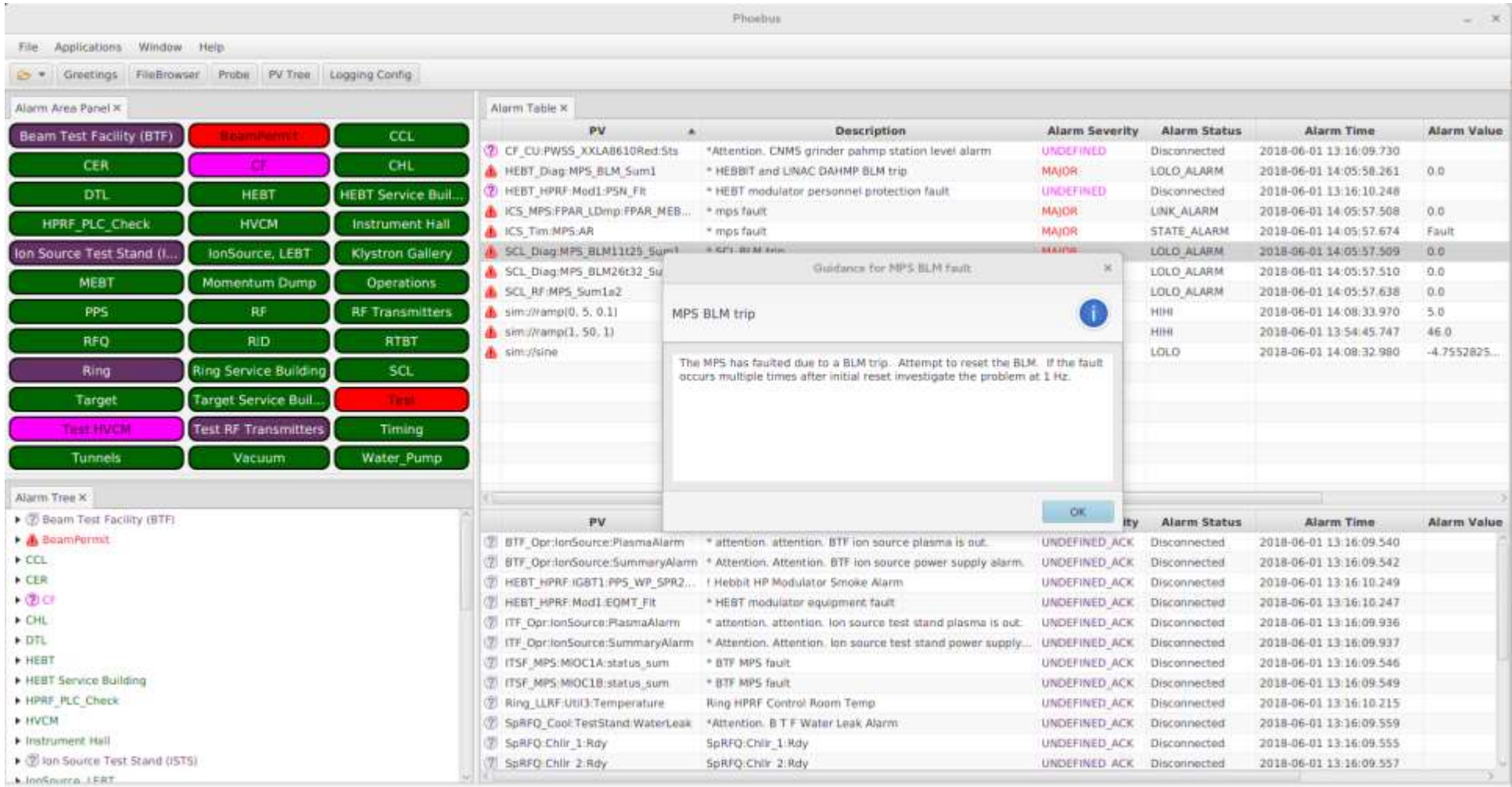
- Stream ‘compacted’ alarm topics into
  - Non-compacted Kafka Topics:
    Long-term history
  - RDB, MongoDB, ElasticSearch:
    Alternate long-term history, existing tools to analyze

For what it’s worth:
ESS uses Kafka to stream
EPICS PVs & Instrument data,
http://accelconf.web.cern.ch/AccelConf/icalepcs2017/papers/tupha177.pdf

Alarm System Update for Phoebus

~~RDB & JMS~~ → APACHE kafka®

- Same XML import/export
- Similar UI
- Performance headroom



Other Changes

- Server now ‘maximizes’ severity of all nodes up to root
  - Used to be in client
  - Has to be in server to support ‘sevr_pv’

→A little more traffic, but

→ simpler client code

→ more information for reports, statistics, dashboards

Quirks (not show stoppers)

- Deleting and then re-creating a topic to “start over” causes log cleaner to quit. Need to restart Kafka.

- “Delete” alarm tree node.. means adding a null entry.
  New clients often see both messages,
  i.e. add and soon delete the entry.

  /Accl/Vacuum/PV1: {“latch:true, …}

  /Accl/Vacuum/PV1: null

  Full cleanup means starting over:
  Export data,
  recreate topic (requires Kafka restart),
  import data.