STAR Online Framework

from Metadata Collection to Event Analysis and System Control

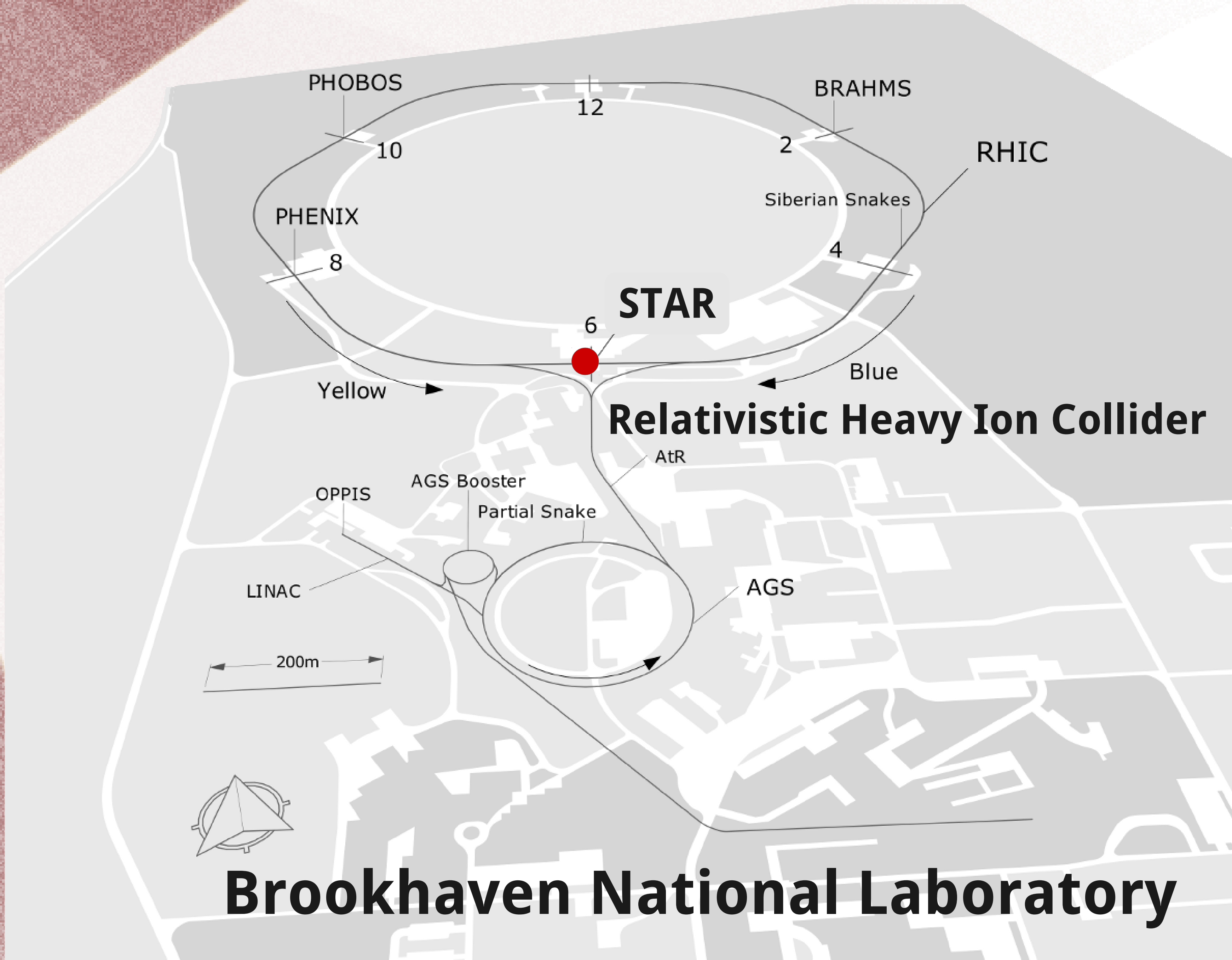
Dmitry Arkhipkin, **Jérôme Lauret**
Brookhaven National Laboratory

16th International workshop on Advanced Computing and Analysis Techniques in physics research (ACAT)
Sep 1st – 5th 2014

Outline

- STAR Intro
- STAR Online M-D Framework Overview
  - MIRA: Messaging Interface and Reliable Architecture
  - Core Components
  - Usage Over Years
- Run 14 MIRA extension: CEP
  - Complex Event Processing
  - Practical Use-Cases
- Framework Future
  - Control System Capabilities
  - Requirements, Design, Technologies
- Summary & Outlook

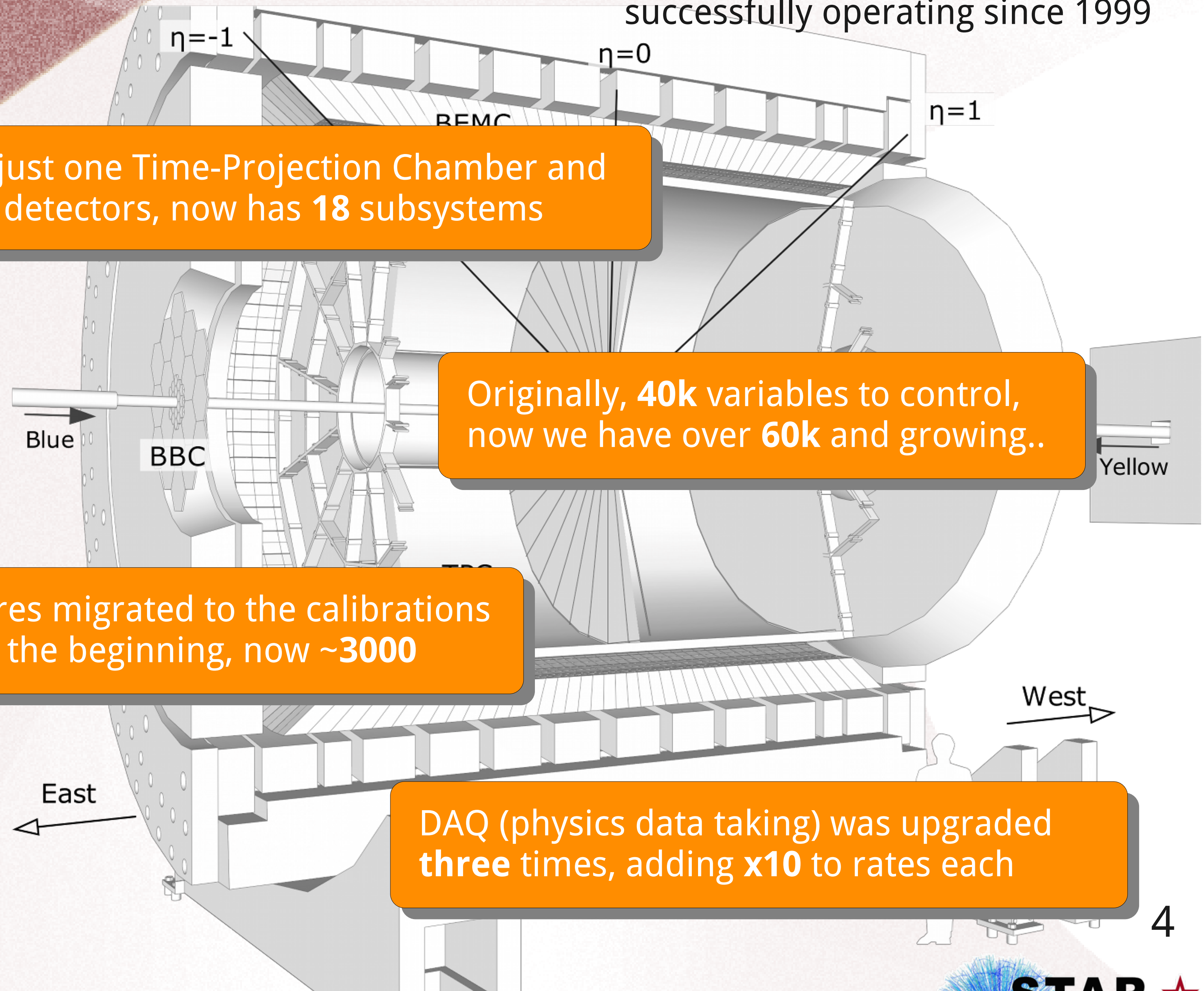
STAR @ RHIC, Brookhaven Lab



Brookhaven National Laboratory

STAR Detector

successfully operating since 1999



Started with just one Time-Projection Chamber and a few trigger detectors, now has **18** subsystems

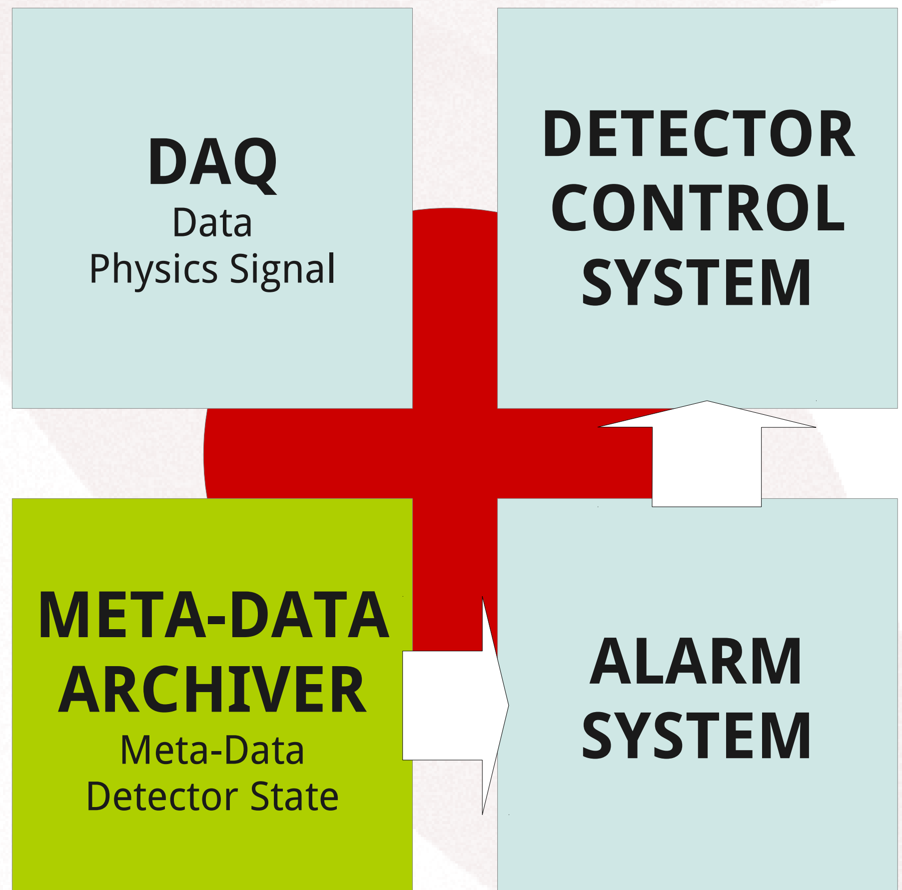
Originally, **40k** variables to control, now we have over **60k** and growing..

120 structures migrated to the calibrations database at the beginning, now ~**3000**

DAQ (physics data taking) was upgraded **three** times, adding **x10** to rates each

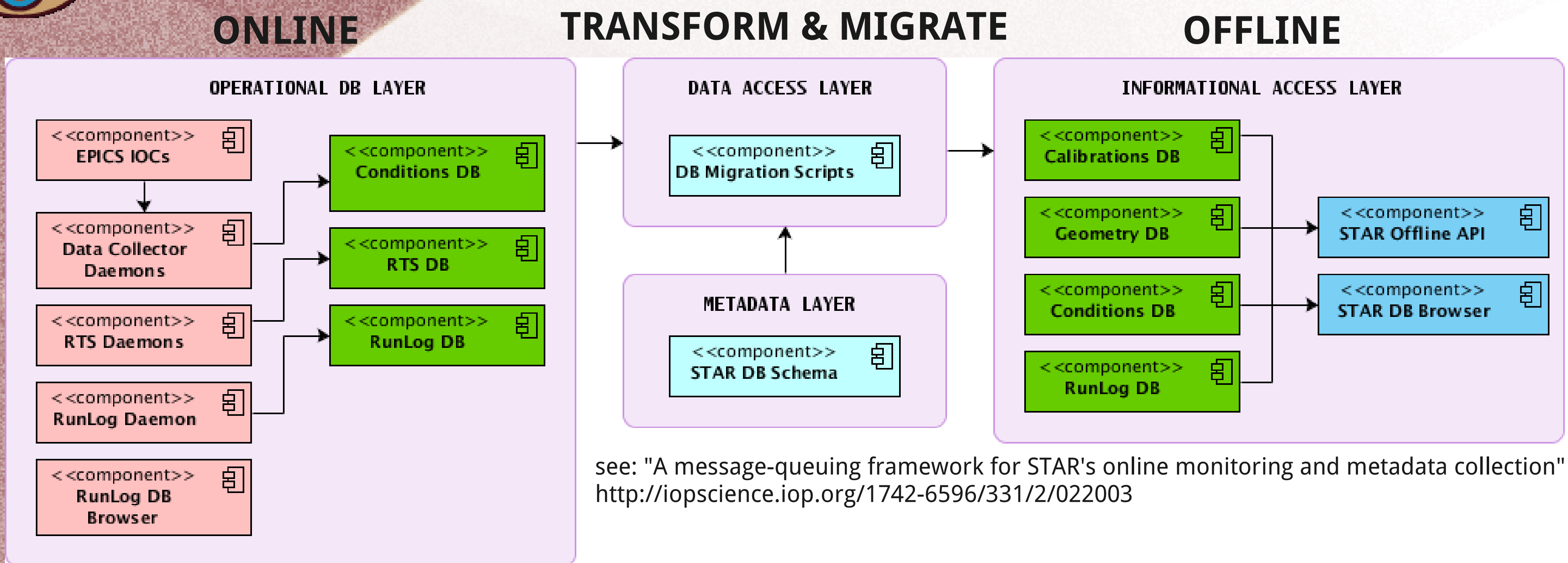
Defining Common Terminology

Cornerstones of "a" Physics Experiment's Backend



MIRA Framework started from this corner (+migration)

Data vs Meta-Data in STAR



see: "A message-queuing framework for STAR's online monitoring and metadata collection"
http://iopscience.iop.org/1742-6596/331/2/022003

Setup Highlights:

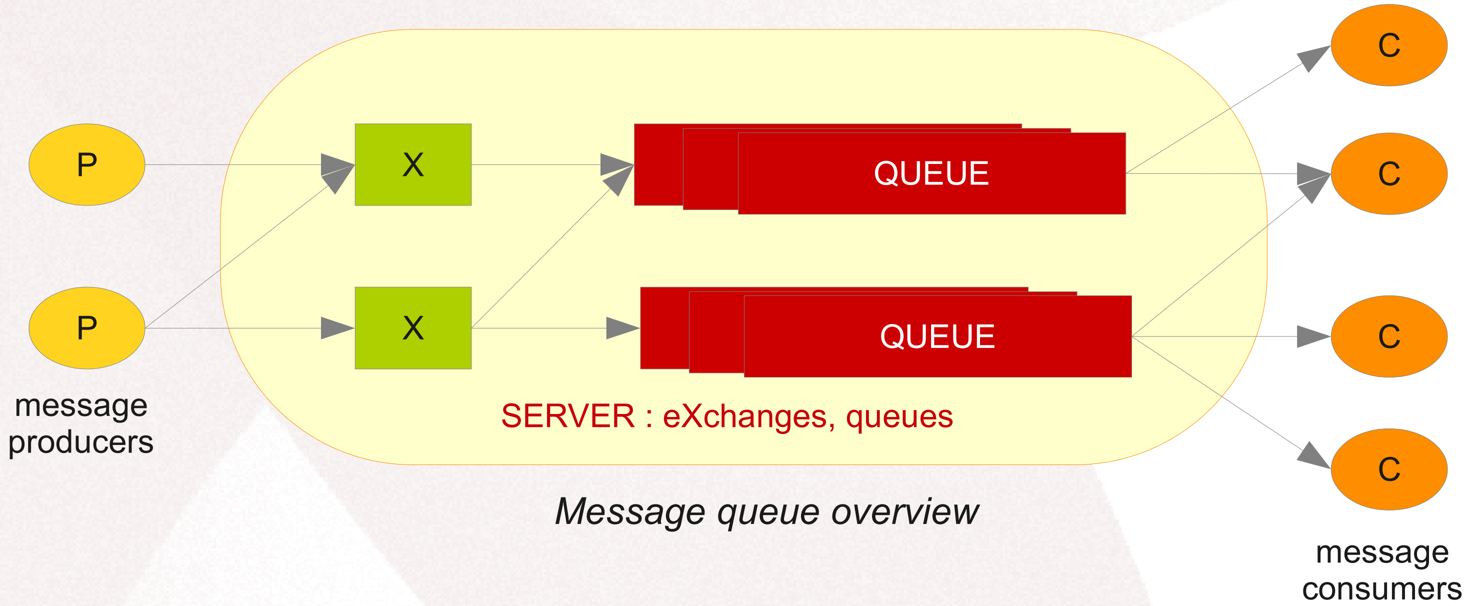
- DAQ = data = physics signals => not discussed in this talk
- Online Framework = meta-data = detector state processing, storage and monitoring

- Storage (database) grouped into two layers: online and offline
  - online db: many input sources, flexible structure, optimized for fast writes
  - offline db: re-formatted meta-data, ready to be applied by calibration makers, optimized for fast reads, has fixed structure

- **BEFORE MQ: tightly-coupled system, WITH MQ: flexible, loosely-coupled system**

Message-Queuing Systems Overview

general idea behind message-queuing service: abstract queues

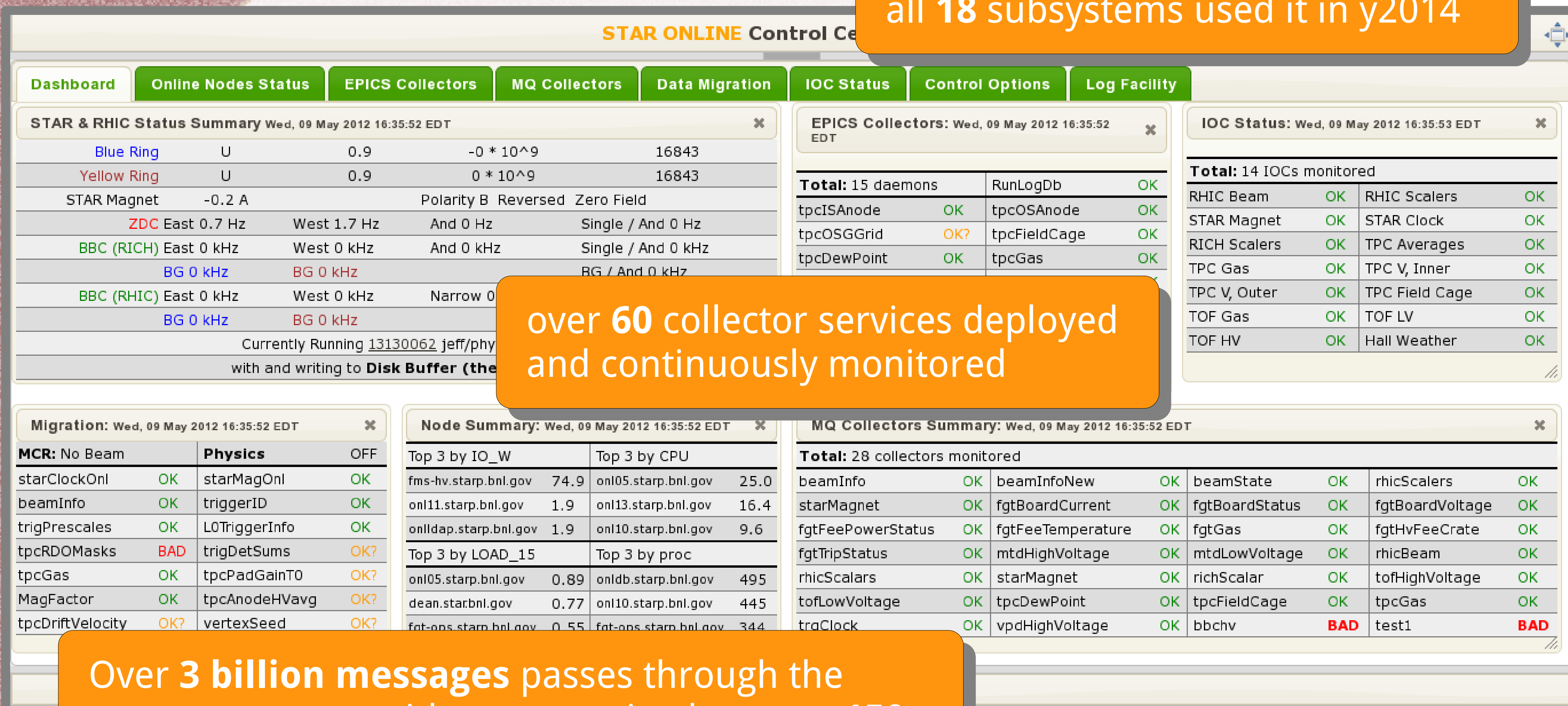


Message queue overview

- **Asyncronous, Payload-agnostic** messaging
- **Highly Concurrent, Scalable** environment
- **Loosely coupled, Modular** architecture
- **Multi-protocol** brokers with persistence support

MIRA: Meta-Data Service Monitoring

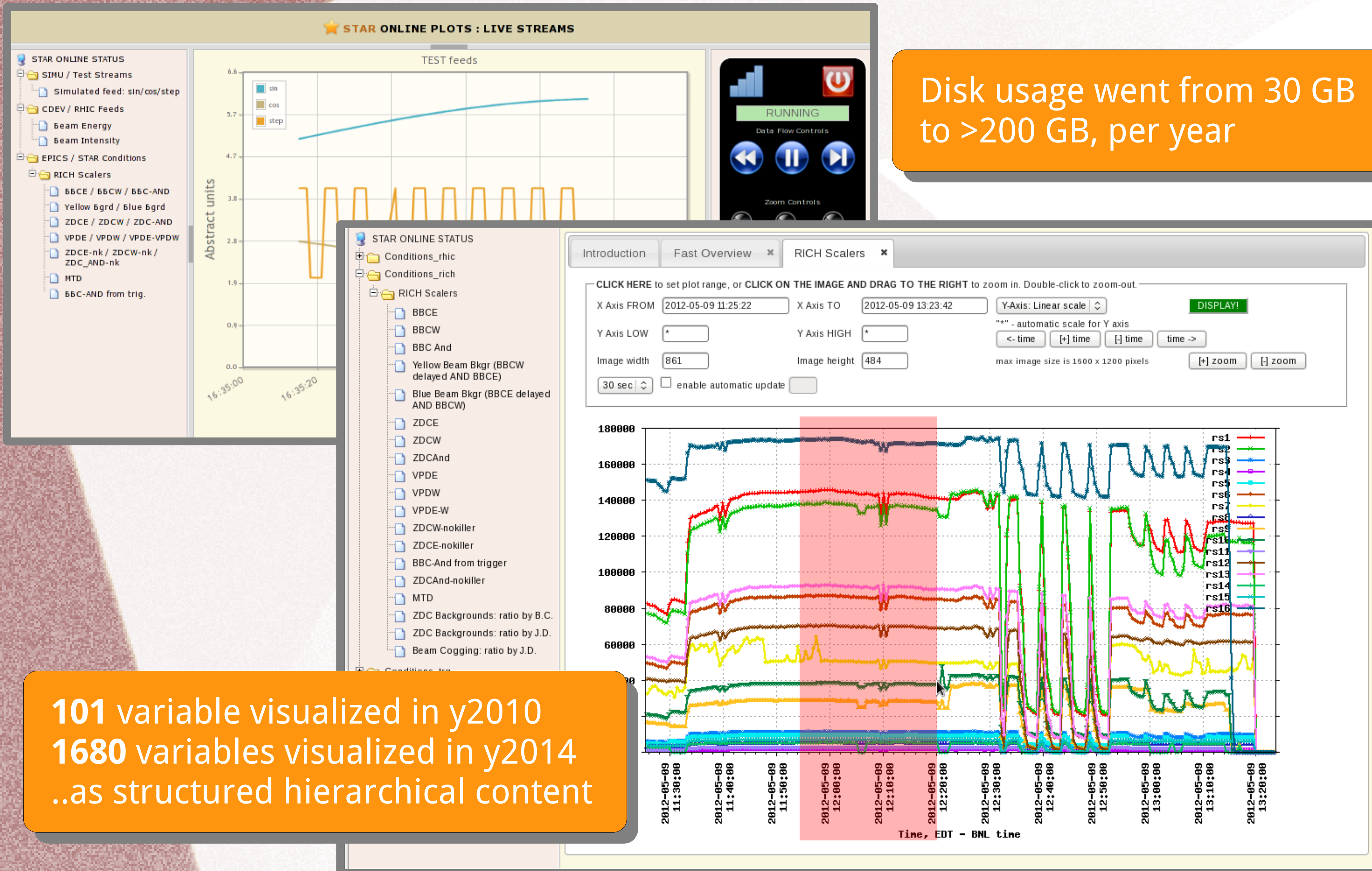
3 subsystems used MIRA in y2010, all **18** subsystems used it in y2014



over **60** collector services deployed and continuously monitored

Over **3 billion messages** passes through the system per year, with rates varying between **150** msg/sec to **2000** msg/sec. On average one message corresponds to structure of **24** variables

MIRA: Visualization Capabilities



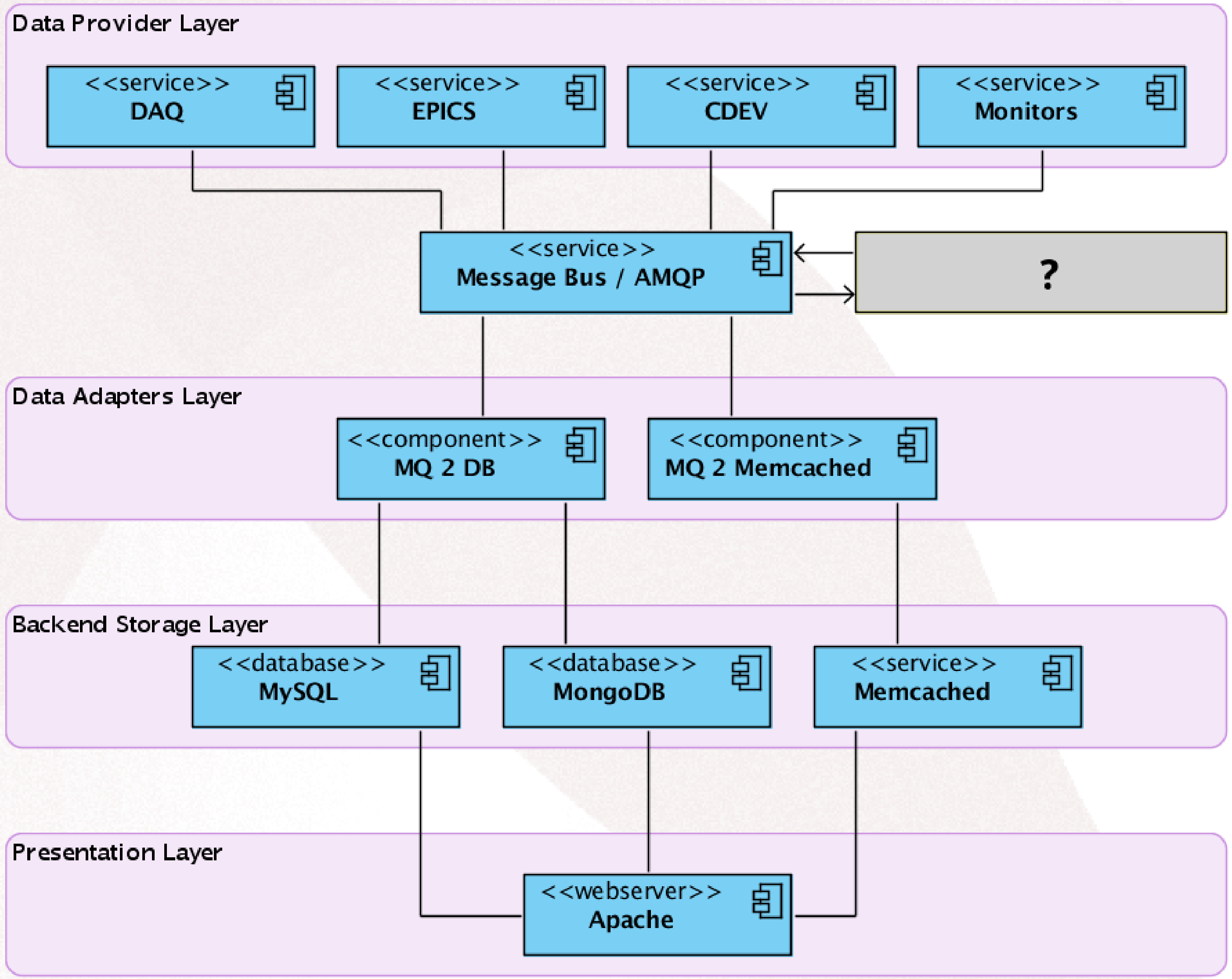
Disk usage went from 30 GB to >200 GB, per year

101 variable visualized in y2010
1680 variables visualized in y2014
..as structured hierarchical content

MIRA: bird's eye overview

***MIRA**: **M**essaging **I**nterface and **R**eliable **A**rchitecture*

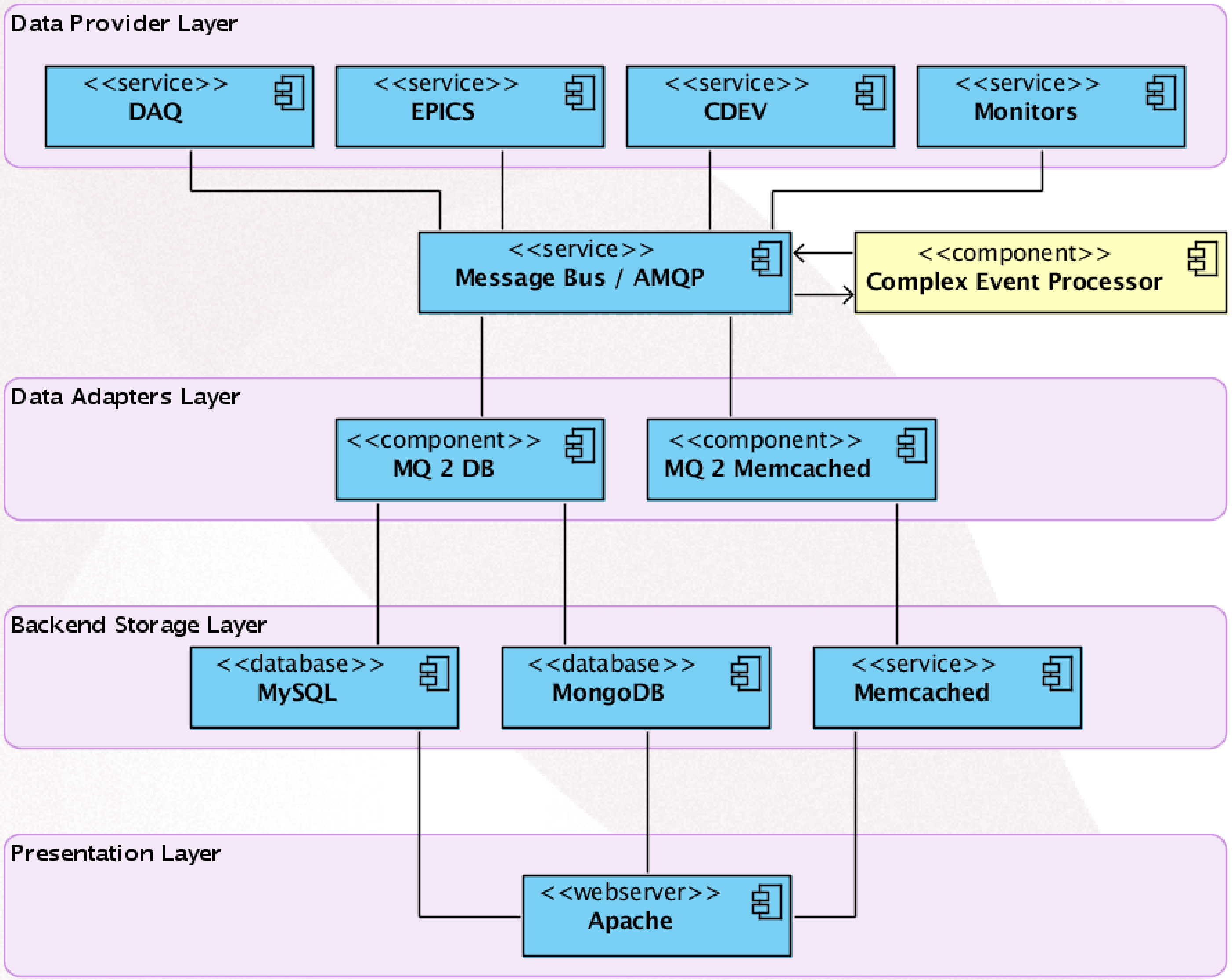
control over **individual streams**: monitoring and storage



Complete overview: "Online Metadata Collection and Monitoring Framework for the STAR Experiment at RHIC"
http://iopscience.iop.org/1742-6596/396/1/012002

MIRA: Introducing Event Processing

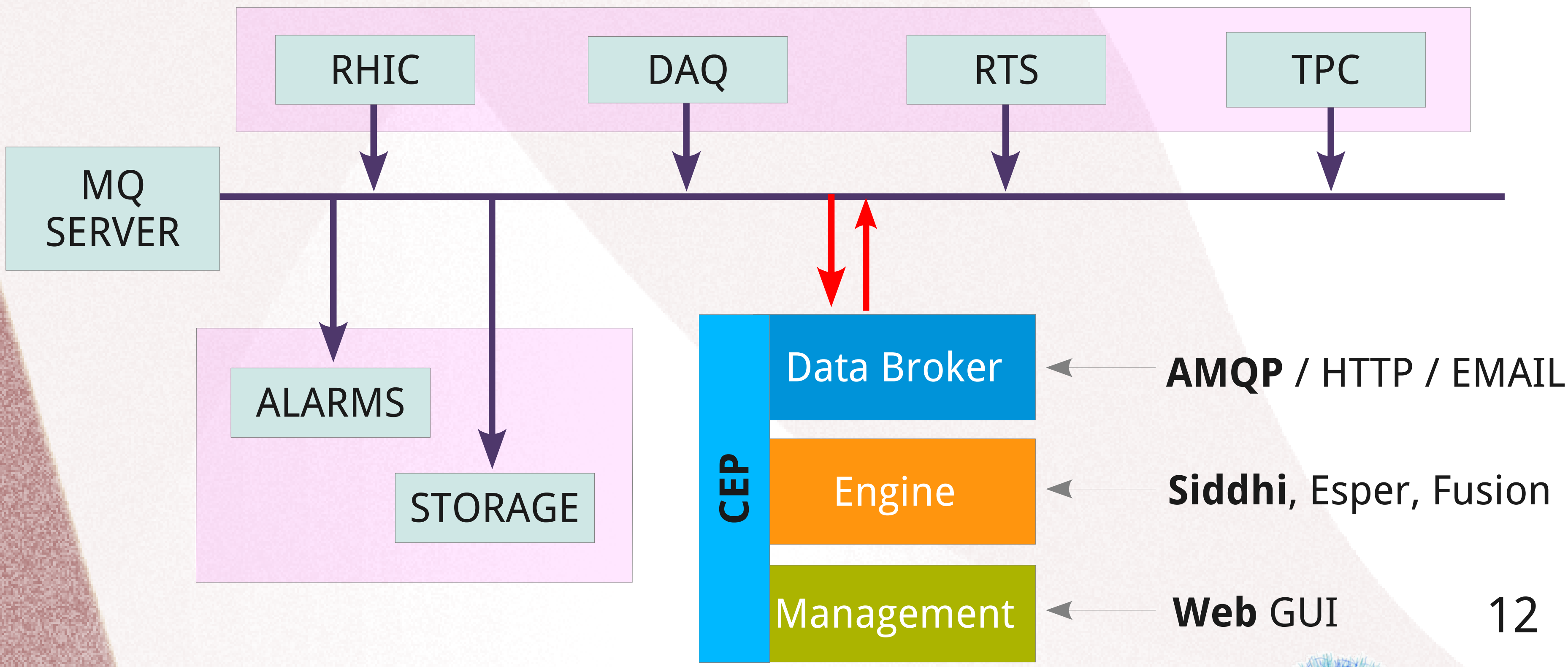
primary goal: **multi-stream** real-time processing



STAR

CEP: internals overview

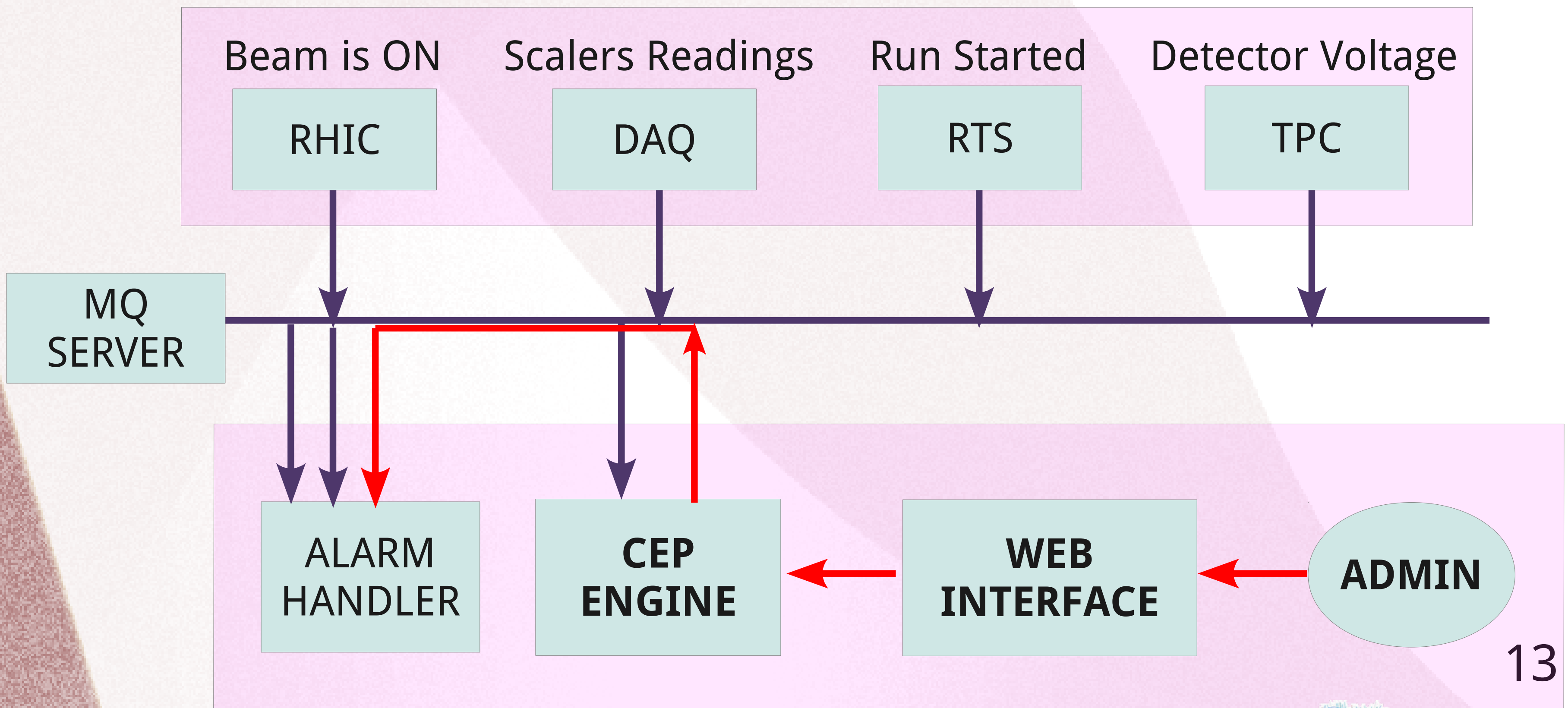
- **Complex Event Processor**: merge streams, process them according to pre-assembled **persistent queries**, produce output stream for processing.
- Event Processor at **STAR** is configured to use **AMQP** topics and queues provided by Apache *qpid* service as **input** and **output** device.



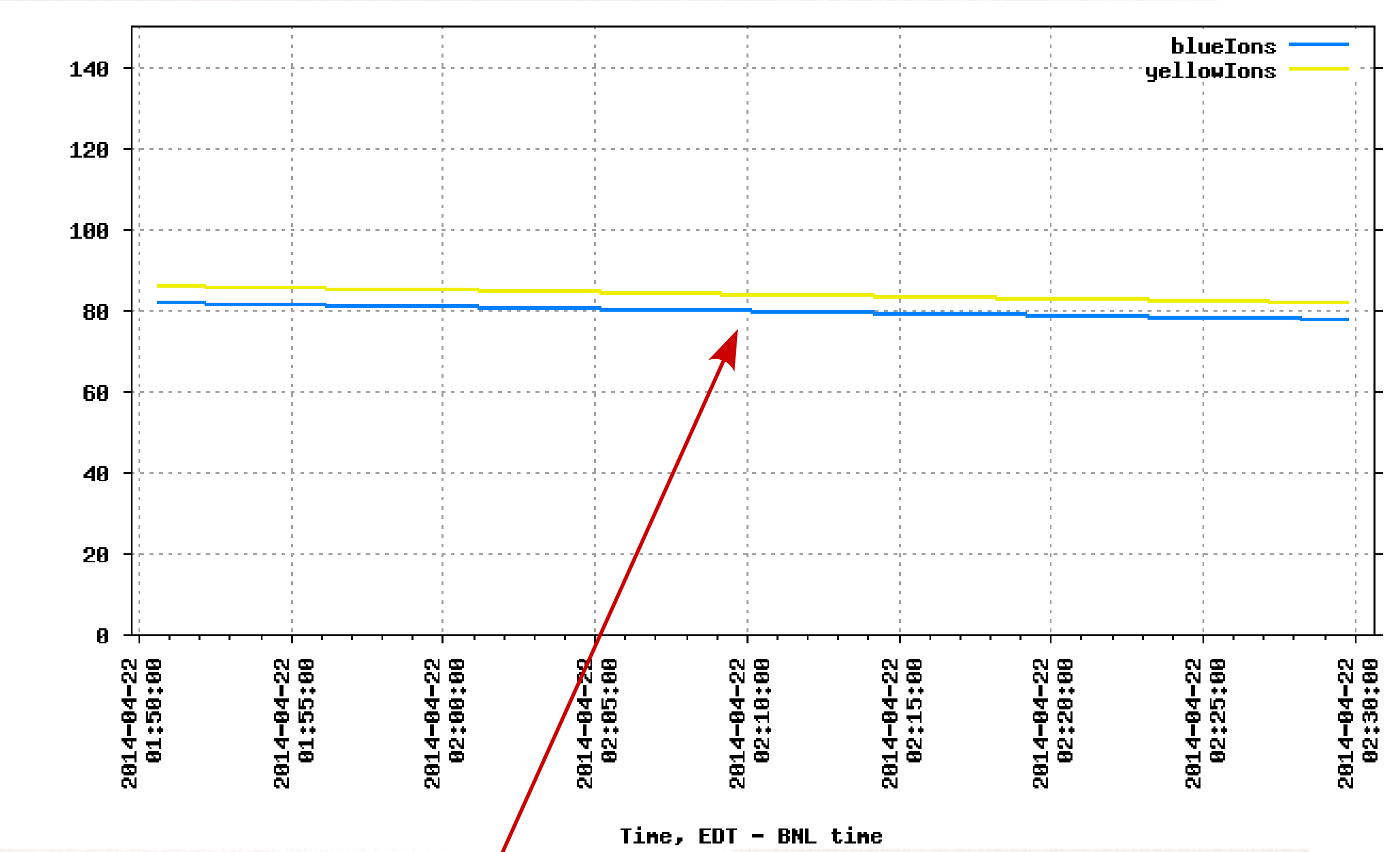
STAR CEP: Practical Use-Case I

Smart and Flexible Alarm System

Flexibility: Ability to reconfigure existing alarm rules or add new ones on a fly without stopping existing monitoring services. Improves user experience and reduces frustration from getting semi-permanent false or nuisance alarms..

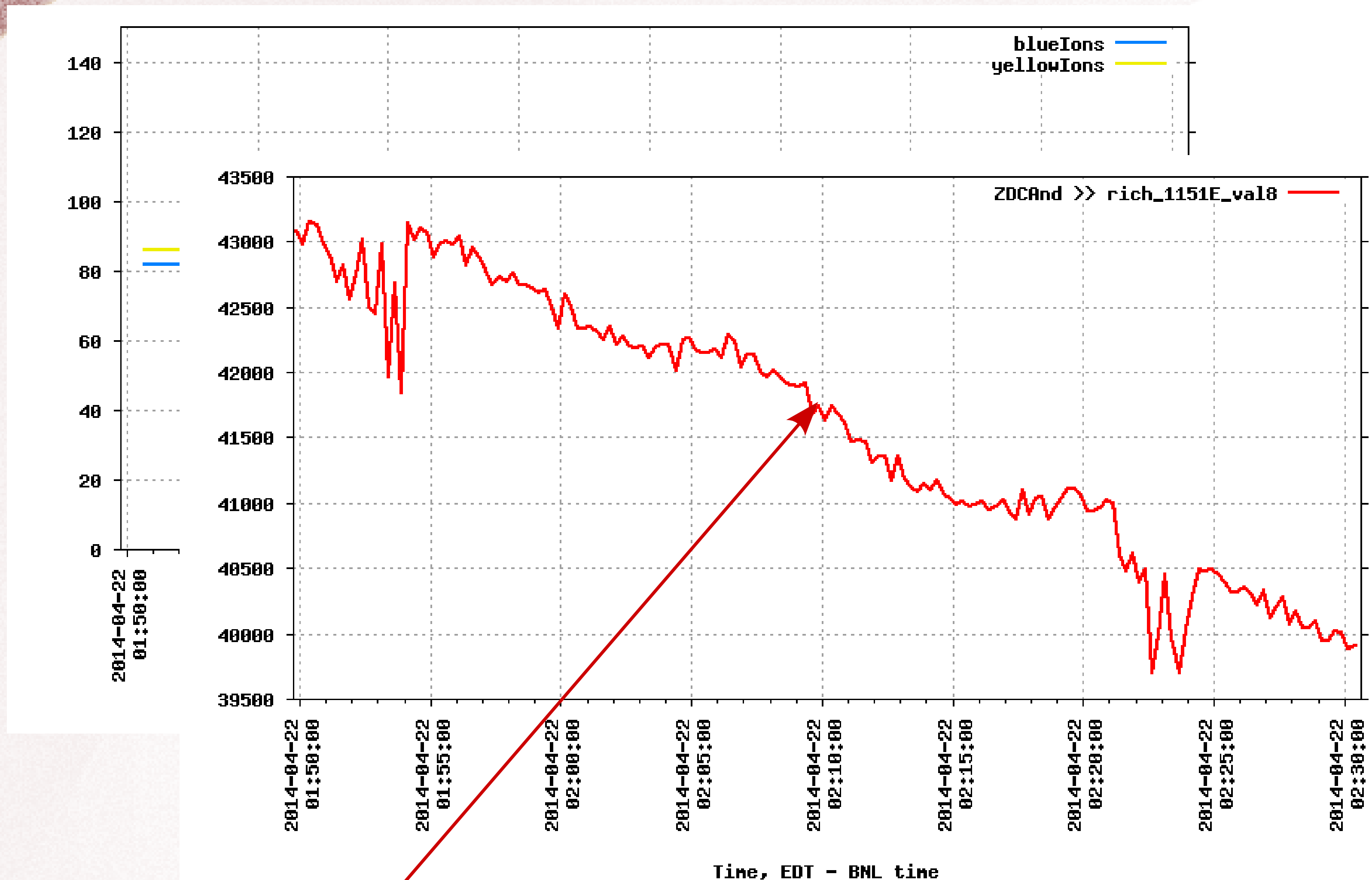


Detector Anomalies
Rare Events



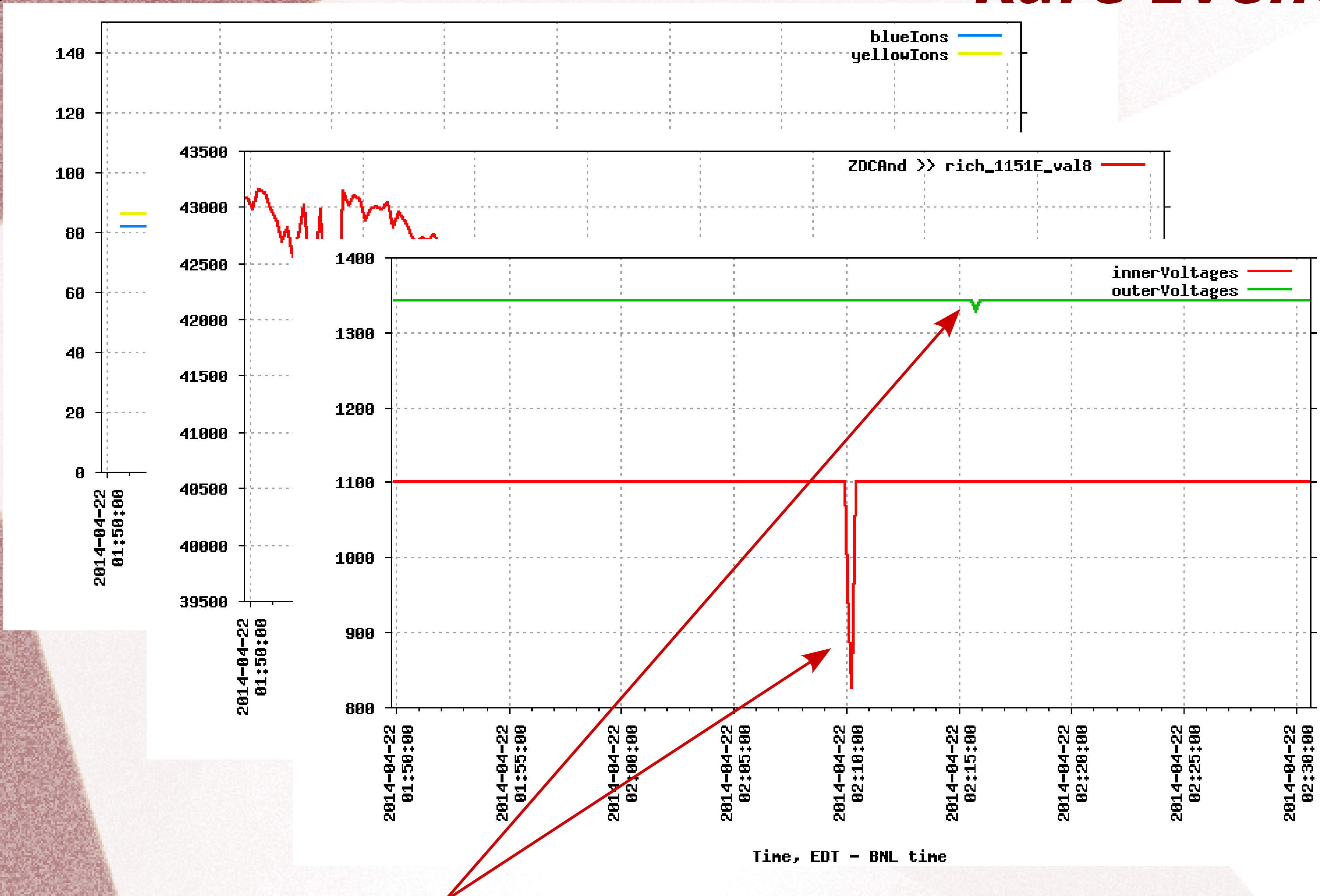
For example, various voltage instabilities when beam is okay..

Detector Anomalies
Rare Events



… and collision rate is within expected limits …

Detector Anomalies
Rare Events



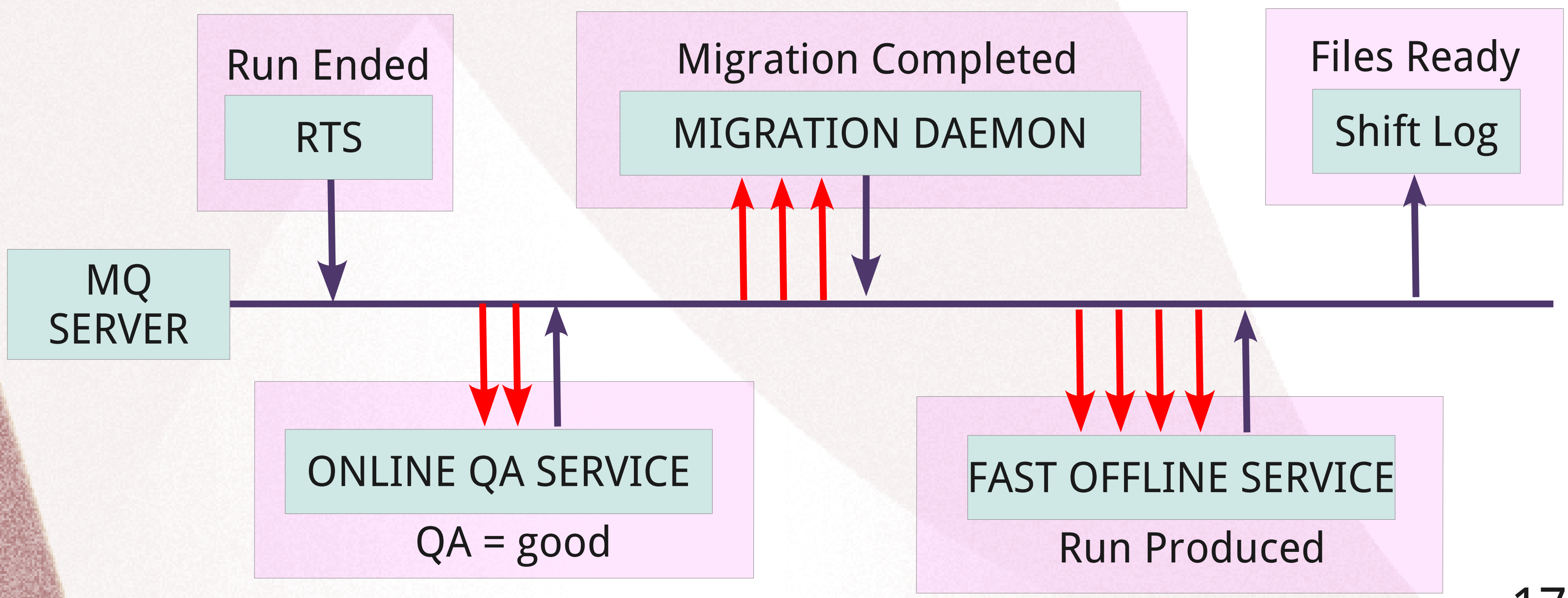
... Kaboom. Data requires careful inspection before analysis.

(~90 events like this one over the 6 month period in the database)

STAR CEP: Practical Use-Case II

Aggregate and filter signals from many sources

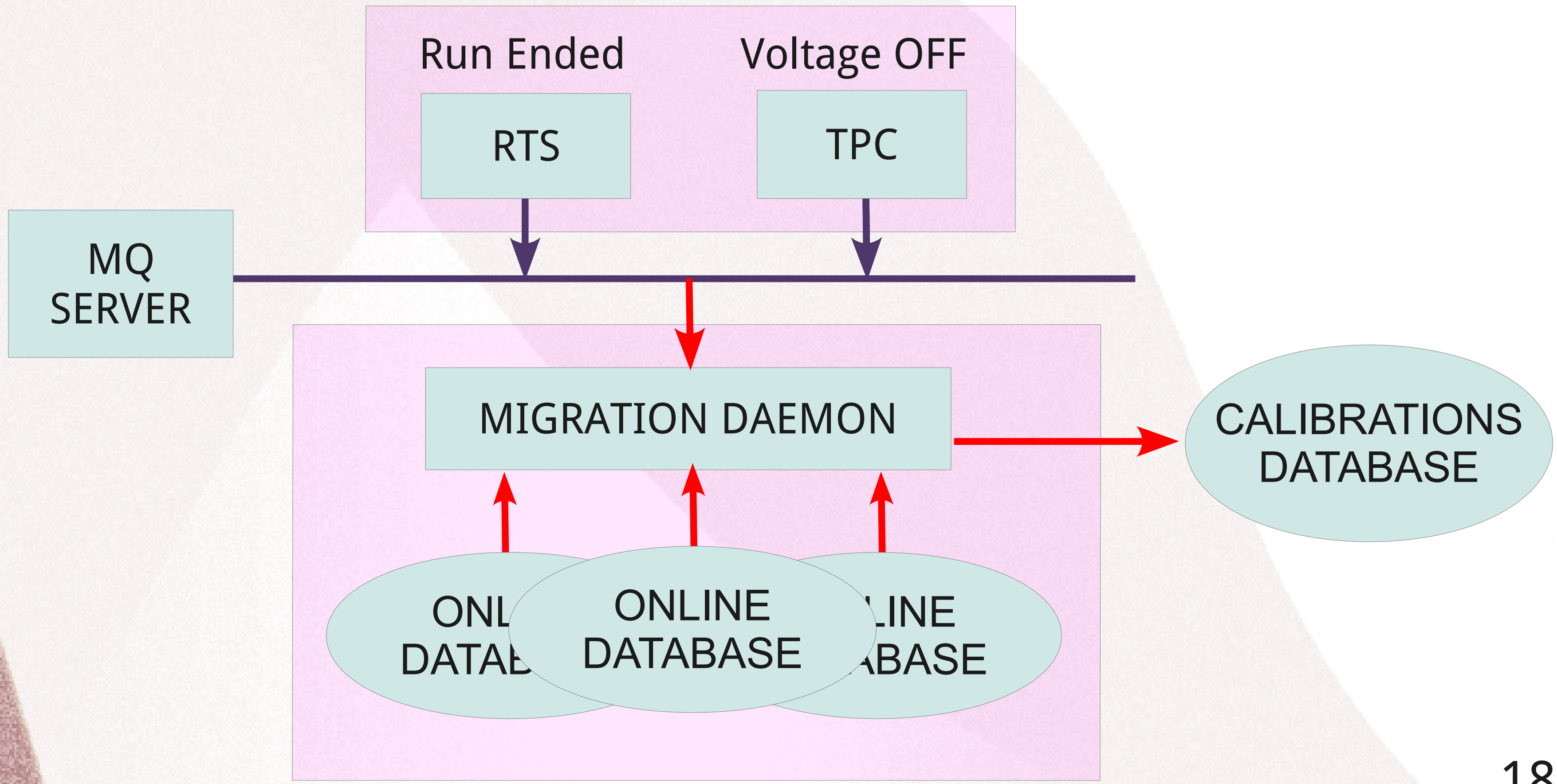
- **Aggregated Data Streams:** Clients now can consume filtered and aggregated streams, published to MQ, no need to do manual merging..
- **Modularity:** Streams, queries, and client services can be tested and response can be validated using simulated data well before a production deployment



STAR CEP: Practical Use-Case III

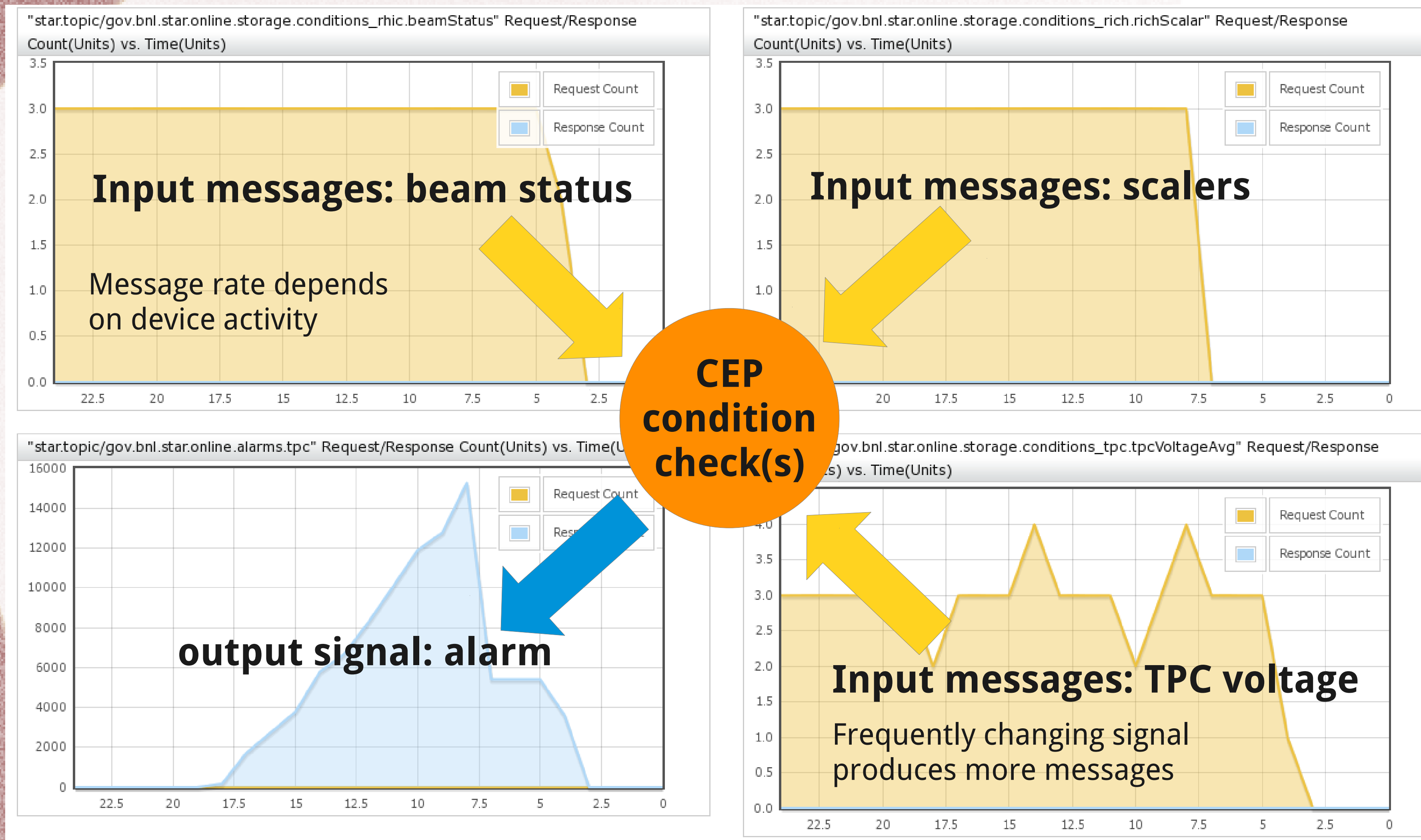
Smart Data Migration

Decreased Resource Consumption: Online Database is large, Calibrations database is compact. Smart data migration requires triggering on specific merged event streams to **avoid constant database polling.**



CEP MQ activity monitoring input/output

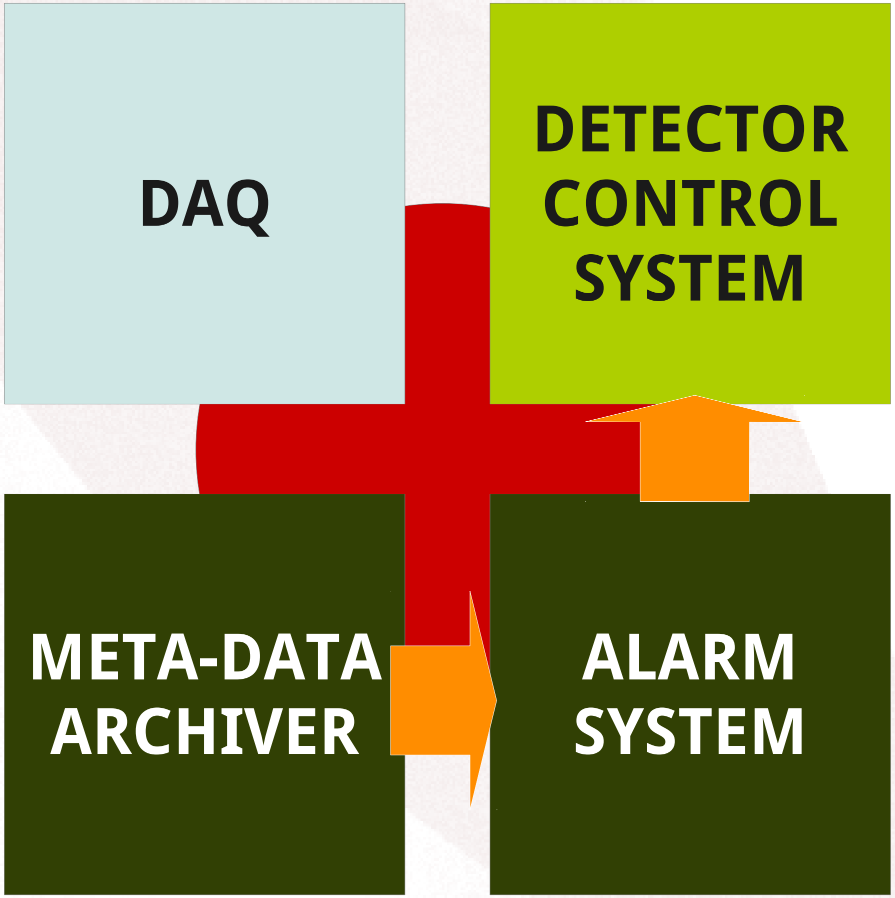
Instant **message rate / activity monitoring**
for all persistent queues configured



Preconfigured CEP rules and activities could be inspected using web interface

MIRA: Evolution

adding features of the Control System



Expanding Framework to provide **Control** features, in addition to Monitoring, Storage, Visualization and Alarms

Detector Control: Requirements

...as gathered from potential developers and users...

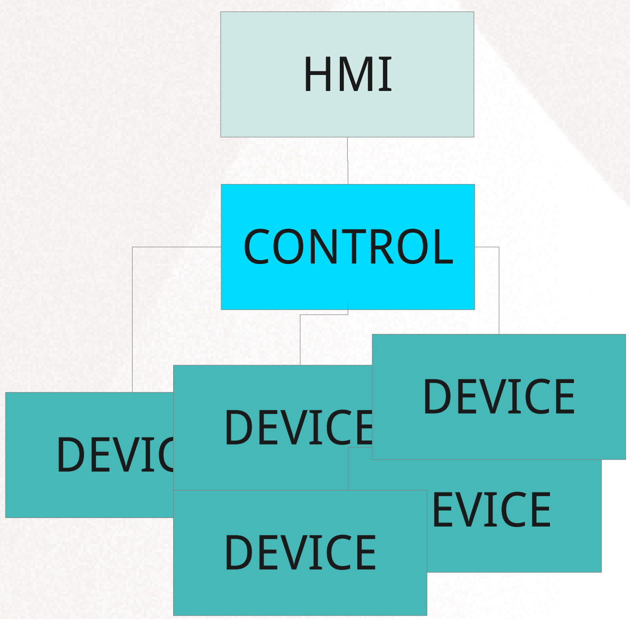
- **Scalable** architecture
- **Inter-operable**, **low-overhead** protocol
- **Payload-agnostic** messaging
- **Quality of Service** regulation
- Improved **Finite State Machine**
- **Real-time Remote Control**
- **Compatibility** with the existing infrastructure

Available Architectures

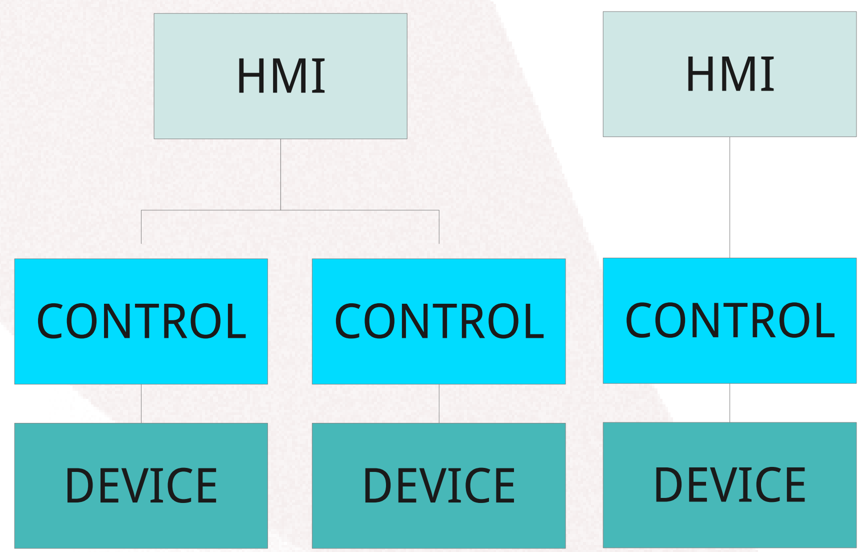
one cautious step forward

Detector Control Systems: there is an architectural choice between industrial **OPC-UA** and common **HEP/NP** approach. Both have strong and weak points:

- OPC-UA: "**complex object with dependencies**" model, centralized,
- HEP/NP: "**single variable without dependencies**" model, decentralized



Industrial Model



HEP/NP Model

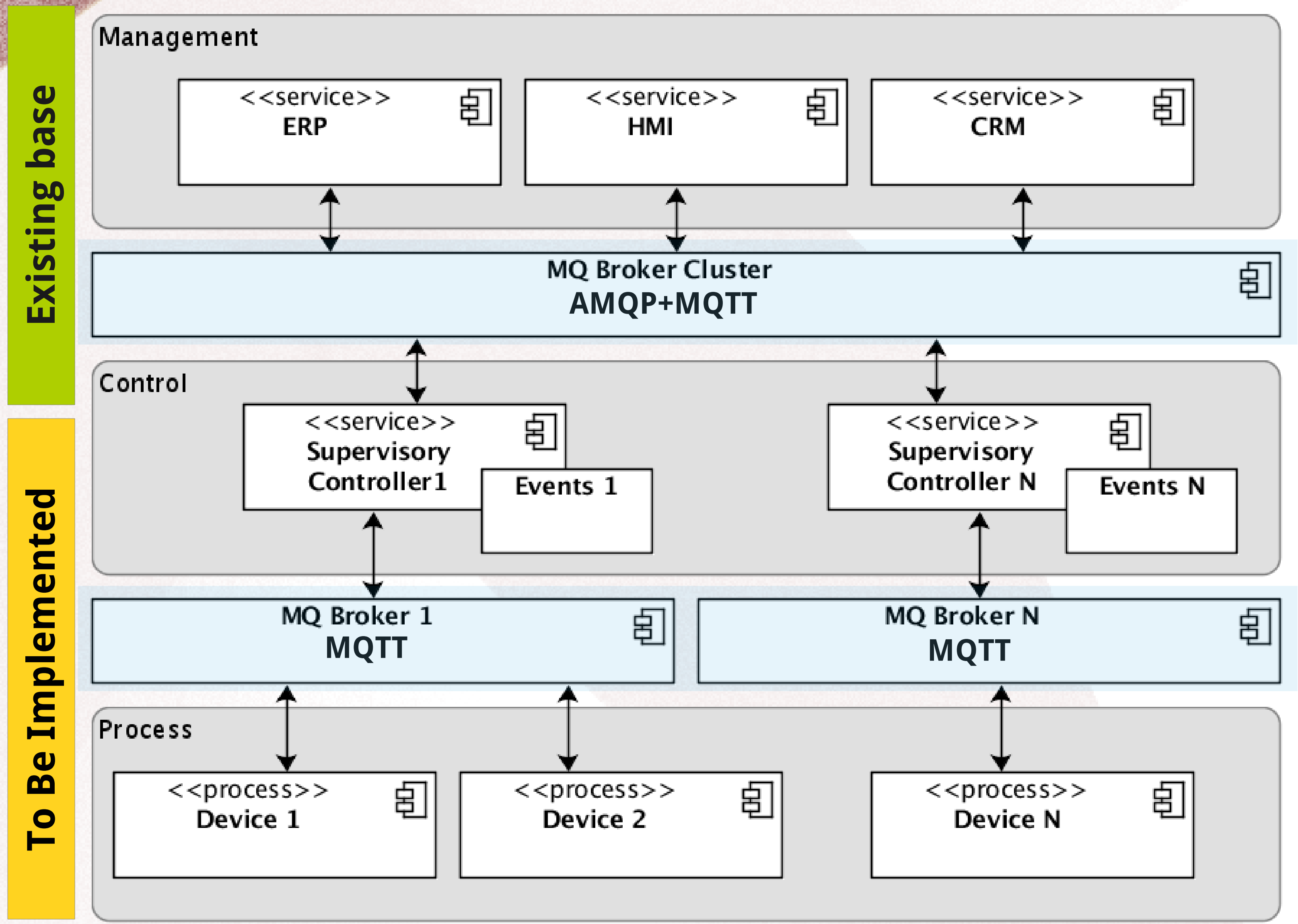
Design Ideas

Our design: "fusion" of both approaches

- **Structures** as first-class citizen
  - better than single variables, and easier to handle than complex objects - no inheritance and no dependencies

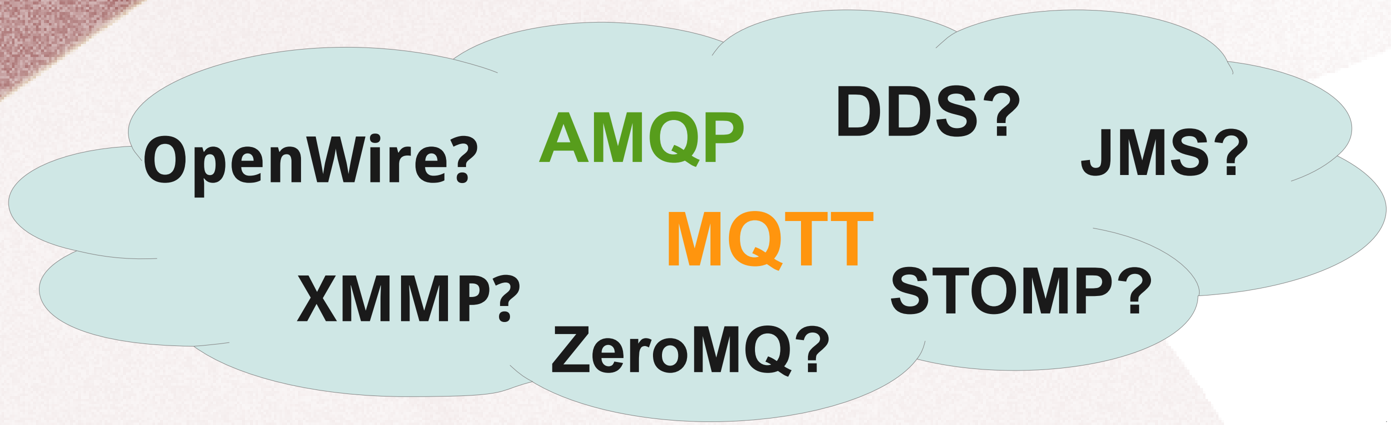
- **Observer Model** implementation:
  - clients decoupled from data producers
  - services attached to MQ may observe any process in the system without interference
  - monitoring, progress tracking, logging..
- **MIRA Framework** addon, to reuse and enhance existing capabilities:
  - automatic archiving and visualization
  - web-based control over services

MIRA: Future Control System architecture overview



Implementation Technologies

with **so many protocols & tools** available,
what do we really want?

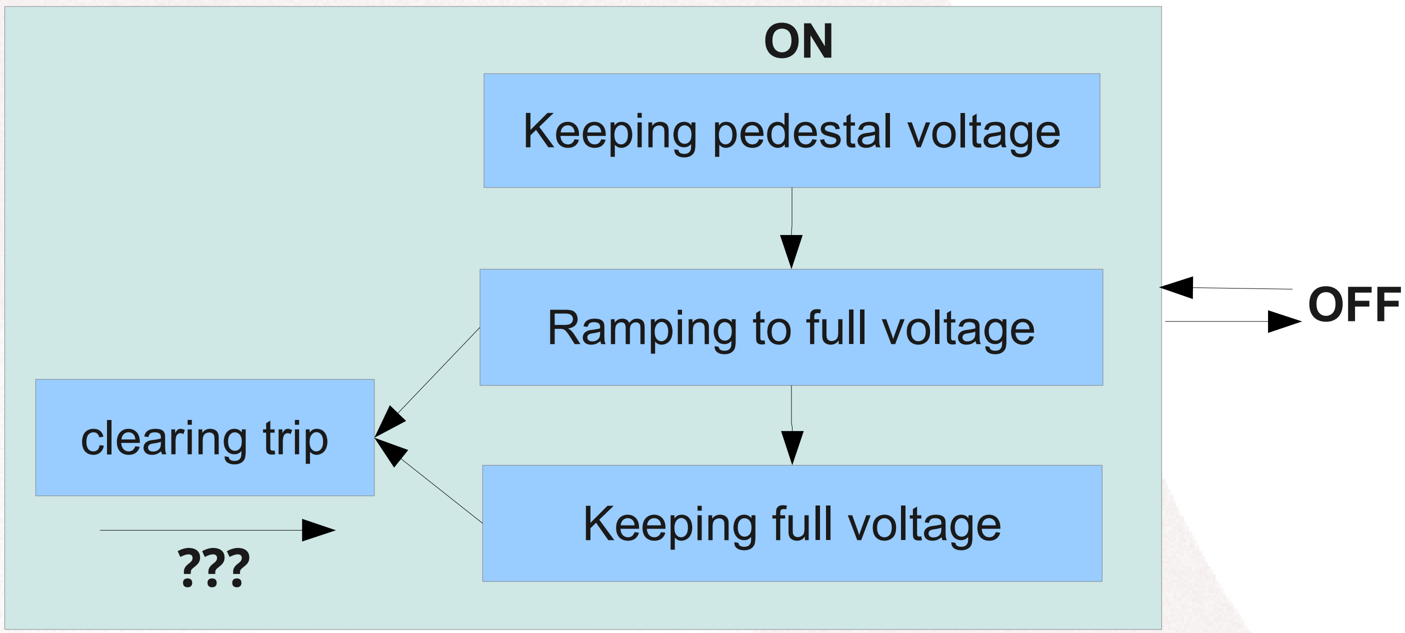


- **Messaging:** AMQP (top) + MQTT (bottom)
- **Engine Core:** H.-S. Finite State Machine
- **Web Interface:** MQTT over WebSockets
- **Local Clients:** same as web interface
- **Multi-protocol broker**: Apache Apollo
    - MQTT: less features, low overhead
    - AMQP: more features, higher overhead

Note: on average, MQTT implementations report <100 µs latency. Some go as low as <50 µs latency, with hundreds of millions of messages per second, and millions of clients connected.

Hierarchical, Stack-based FSM

- Why do we need **Yet Another FSM**?
  - Events come from **MQ** – brokerage is a must
  - **complex-state** and **sub-state** support desired
  - **trace back**: need state history to resolve multi-exit state scenarios



Hierarchical = reduces number of state exits, Stack-based = state history

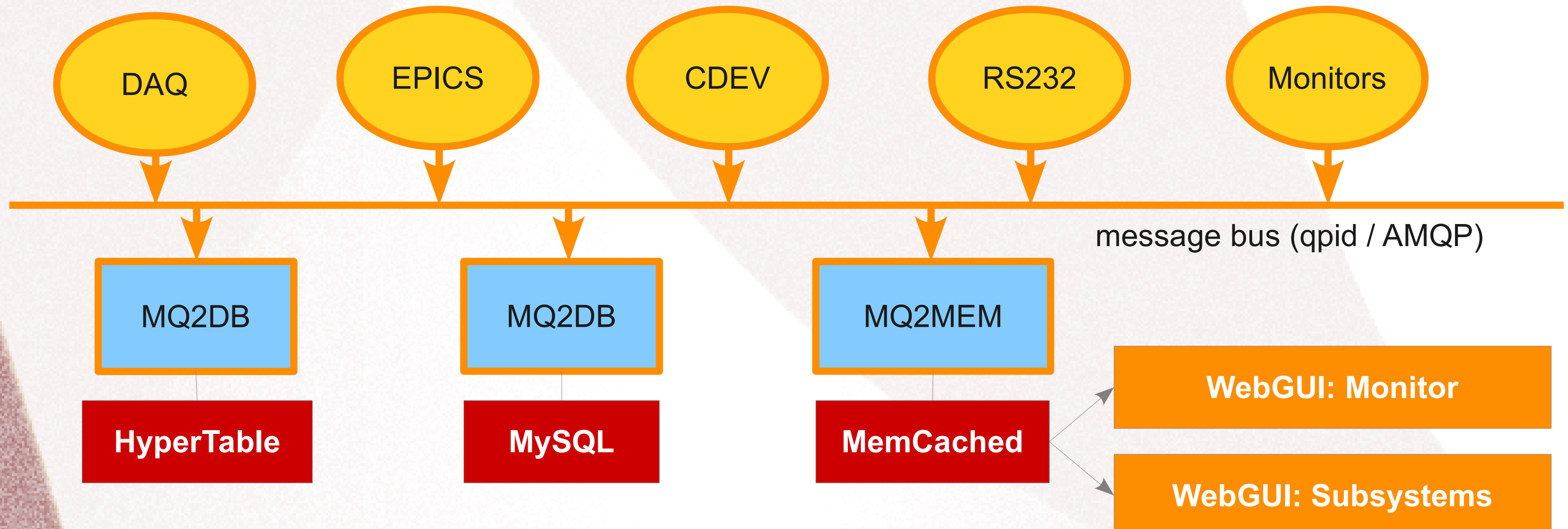
Summary & Outlook

- **STAR Meta-Data Collection Framework** overview was presented
  - Message-Queuing service became an instrumental part of STAR online infrastructure
  - MQ-based: flexible, loosely coupled system
  - Accepted very well by STAR collaborators and detector experts, covers the monitoring needs of all 18 STAR subsystems now
  - Number of channels has increased to ~1700, or x15, number of data structures has increased to ~3000, or ~x25
- **Run 14 Extension: Complex Event Processing**
  - CEP features added and tested in 2014, now we are confident in its capabilities. Deploying for a full production usage in 2015
  - Proven be be helpful: a few alarms implemented in Run 14, saved months of work for the core team and users. More use-cases to be implemented for Run 15 and beyond.
- **Future** framework features planned for 2015 and beyond:
  - Modular, scalable architecture for the Control System, including remote control interface
  - Developers and users are looking forward to implementation

Backup Slides

MIRA Framework References

- 2012. "Online Metadata Collection and Monitoring Framework for the STAR Experiment at RHIC", poster at CHEP'12 http://iopscience.iop.org/1742-6596/396/1/012002
- 2011. "A message-queuing framework for STAR's online monitoring and metadata collection", presentation at CHEP'10 http://iopscience.iop.org/1742-6596/331/2/022003



MQ-based Data Archiver schema

What Is Complex Event Processing?

- **Complex Event Processing :**
  - time-based, operates on continuous streams of data coming from many sources
  - understands and manages stream relations
  - assumes high event rate
  - detects patterns in data
  - produces output event streams or individual events

- **CEP service includes :**
  - input/output data broker (MQ, REST, WS)
  - event processing engine (persistent queries)
  - stream manager (add/remove sources on a fly)
  - query manager (add/remove queries on a fly)

MiddleWare of choice: WSO2

http://docs.wso2.org/display/CEP210/Complex+Event+Processor+Documentation

- “..WSO2 Complex Event Processor (CEP) is a lightweight and easy-to-use, 100% open source middleware product, available under Apache License v2.0. WSO2 CEP identifies the most meaningful events within the event cloud, analyzes their impacts, and acts on them in real time. It's built to be extremely high performing and massively scalable..”
- The Complex Event Processor consists of the following components: CEP Core, Broker Core, Broker Manager.
- CEP Core contains CEP Buckets which are instances of back-end CEP runtime engines that process events, and Data Converters for converting events from Map, XML, and Tuple types to back end CEP engine's event type. Total processing on received events and triggering of new events happen at the back end CEP runtime engine of each bucket.
- There is a broker between external event publishers/servers and the CEP Core. There are four types of brokers which are Local, WS-Event, JMS/JMS-qpid and Agent. These brokers are responsible for receiving and publishing event on Thrift, SOAP, REST, and JMS transports.

CEP: XML Config Sample

```
<?xml version="1.0" encoding="UTF-8"?>
<cep:bucket name="alarmHandlerBucket"
xmlns:cep="http://wso2.org/carbon/cep">
  <cep:description>Alarm Handler Bucket</cep:description>
  <cep:engineProviderConfiguration engineProvider="SiddhiCEPRuntime">
    <cep:property
name="siddhi.persistence.snapshot.time.interval.minutes">0</cep:property>
    <cep:property
name="siddhi.enable.distributed.processing">false</cep:property>
  </cep:engineProviderConfiguration>
  <cep:input brokerName="qpidJmsBroker"
topic="star.topic/gov.bnl.star.online.storage.conditions_rich.richScalar">
    <cep:mapMapping queryEventType="Tuple" stream="starScalarStream">
      <cep:property inputName="scalar1" name="scalar1" type="java.lang.Double"/>
      <cep:property inputName="scalar2" name="scalar2" type="java.lang.Double"/>
    </cep:mapMapping>
  </cep:input>
<...declare more input streams here...>
```

CEP Engine

MQ Broker - Input

Event Variables - Input

CEP XML Config, Cont-d

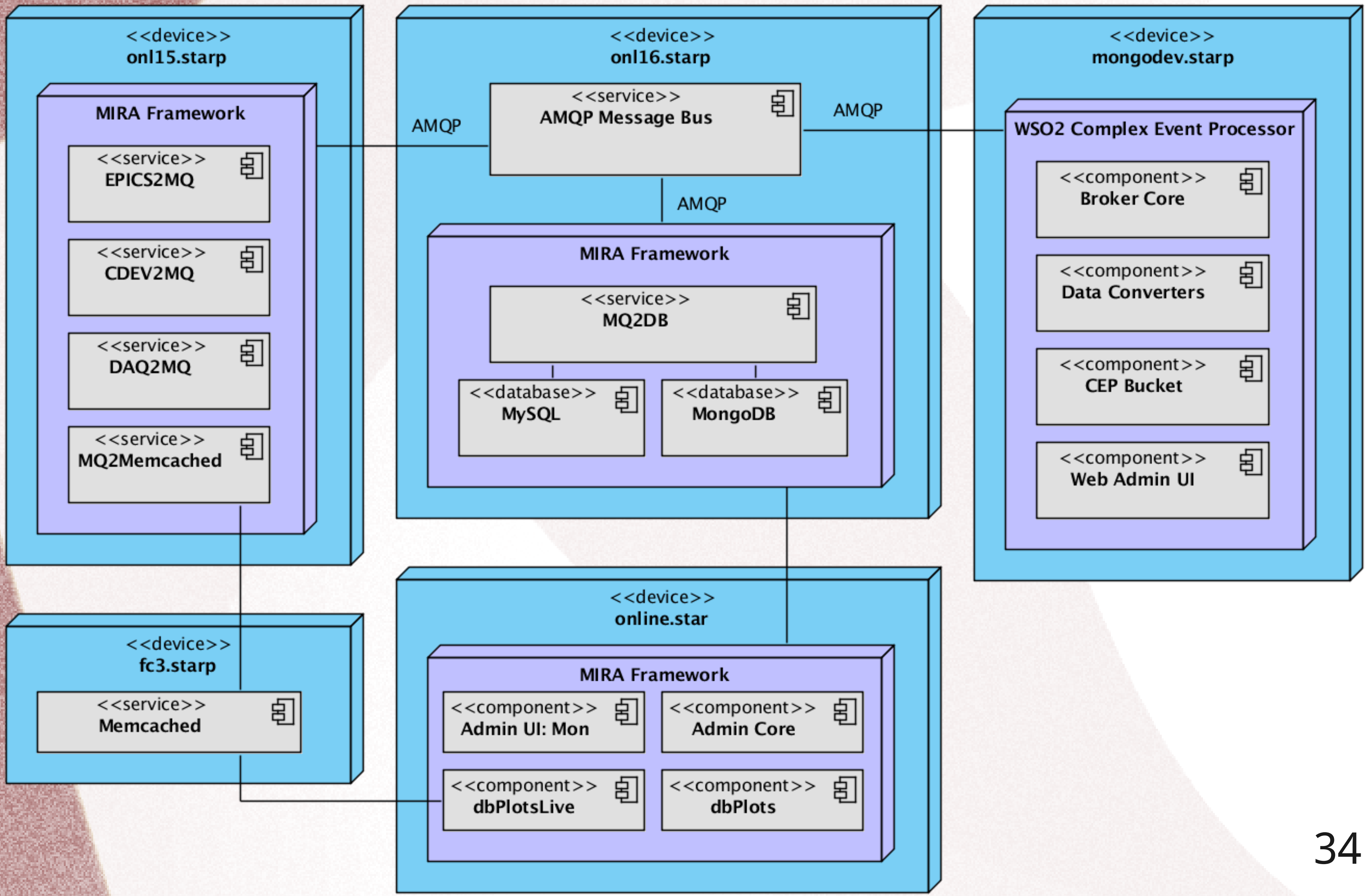
```
<...lines skipped...>
  <cep:query name="outputQuery">
    <cep:expression><![CDATA[
from scalarVoltageStream as bv
join beamStatus[ beam_on > 0 ] as bs
insert current-events into alarmHandlerOutputStream
bv.tpcVoltageInner as innerVoltage, bv.tpcVoltageOuter as outerVoltage, bv.scalar
as scalar, bs.beam_on as alarm_condition
]]></cep:expression>
    <cep:output brokerName="qpidJmsBroker"
topic="star.topic/gov.bnl.star.online.alarms.tpc">
      <cep:mapMapping>
        <cep:property name="outerVoltage" valueOf="outerVoltage"/>
        <cep:property name="innerVoltage" valueOf="innerVoltage"/>
        <cep:property name="scalar" valueOf="scalar"/>
        <cep:property name="beam_on" valueOf="alarm_condition"/>
        <cep:property name="alarm_condition" valueOf="alarm_condition"/>
      </cep:mapMapping>
    </cep:output>
  </cep:query>
</cep:bucket>
```

CEP Query

MQ Broker - Output

Event Variables - Output

CEP Deployment Diagram for Run 14



MQTT Brokers

Open Source: Apache Apollo

http://activemq.apache.org/apollo/

- Multi-protocol broker
- MQTT 3.1 and AMQP 1.0 protocols supported
- Topics, Queues, Durable Subscriptions
- Is shipped with MQTT over WebSockets plugin for web
- Supports secure WebSocket connections

Commercial: IBM MessageSight

http://www-03.ibm.com/software/products/en/messagesight

One appliance can handle:

- 1M Concurrent Connections
- 13M non-persistent msg/sec
- 400K persistent msg/sec
- predictable latency in the microseconds under load

WebSockets

"Reducing kilobytes of data to 2 bytes... and reducing latency from 150ms to 50ms is far more than marginal

In fact, these two factors alone are enough to make WebSockets seriously interesting to Google."

- Ian Hickson (Google, HTML5 Spec Lead)