Online Meta-data Collection and Monitoring Framework for the STAR Experiment at RHIC

Authors: **Dmitry Arkhipkin, Jérôme Lauret, Wayne Betts, Gene Van Buren**
Brookhaven National Laboratory

The STAR Experiment further exploits scalable message-oriented model principles to achieve a high level of control over online data streams. In this poster we present an AMQP-powered Message Interface and Reliable Architecture framework (MIRA), which allows STAR to orchestrate the activities of Meta-data Collection, Monitoring, Online QA and several Run-Time / Data Acquisition system components in a very efficient manner. The very nature of the reliable message bus suggests parallel usage of multiple independent storage mechanisms for our meta-data. We describe our experience with a robust data-taking setup employing MySQL- and HyperTable-based archivers for meta-data processing. In addition, MIRA has an AJAX-enabled web GUI, which allows real-time visualisation of online process flow and detector subsystem states, and doubles as a sophisticated alarm system when combined with complex event processing engines like Esper, Borealis or Cayuga. The performance data and our planned path forward are based on our experience during the 2011-2012 running of STAR.

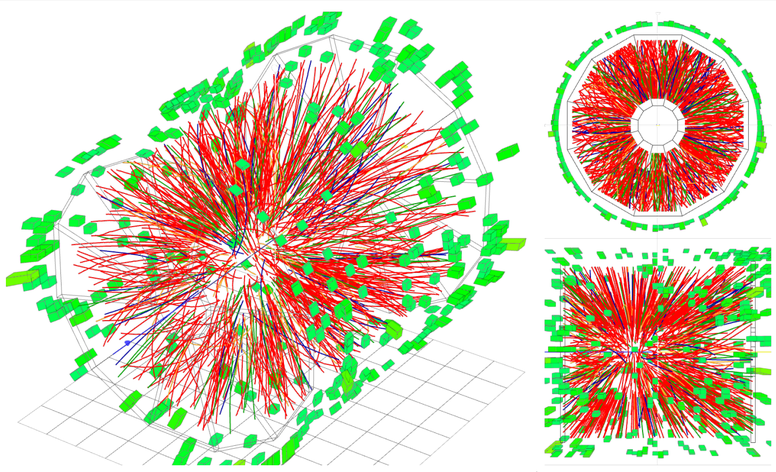


Fig.1 *STAR Detector U+U event, @ $\sqrt{S_{NN}}$=193 GeV*

An acronym for the Solenoidal Tracker At RHIC (Relativistic Heavy Ion Collider), STAR tracks thousands of particles (see Fig.1) produced in Au+Au, U+U, Cu+Cu and polarized p+p collisions. While STAR collects physics data, there are always large volumes of accompanying meta-data to track, store and analyze.

The STAR detector is composed of over two dozen subsystems operating in concert while RHIC provides colliding beams. Hundreds of scientists and engineers are watching the data-taking process and tuning detector performance by checking meta-data streams produced by detector components. To ease the procedure of collection, analysis and review of that meta-data, we created an MQ-based MIRA framework, reported in this poster.

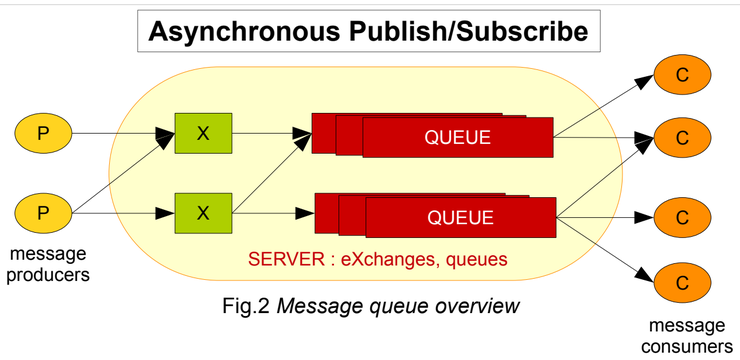


Fig.2 *Message queue overview*

Message Queuing Architecture (MQA): an architecture for distributed systems based on the concept of reliable message queuing. In such architecture, messages are delivered asynchronously between multiple tiers of a system. Typically, it incorporates a message broker component, which mediates communication between message producers and message consumers. MQA is generally characterized by reliability, scalability, and abstraction.

The Message Interface and Reliable Architecture (MIRA) framework is composed of the following components: (a) an AMQP-compliant client-server implementation – the *qpid* package by Red Hat, (b) a set of adapter-like daemons to allow conversion from various data sources to MQ format, (c) a set of adapter daemons to store MQ-produced data streams to database backends (MySQL, HyperTable, Sqlite), (d) Browser-based tools for meta-data tracking and review (dbPlots, dbPlots Live) and (e) a Control Center application, which orchestrates this system by tracking various states for subcomponents and issuing commands to adapter daemons (like stop/start/pause/info). The Control Center also has a jQuery-powered web GUI, which provides a convenient dashboard for system administrators and power users (run experts, database administrators).

During 2011-2012 (RHIC Runs 11 and 12), STAR had the following data sources deployed and controlled by MIRA: collider status (CDEV) variables, STAR status and parameters (EPICS), some DAQ channels, and utility parameters like “physics on/off” signals. As a storage backend, we had widely popular MySQL and scalable, write-tolerant HyperTable databases deployed.

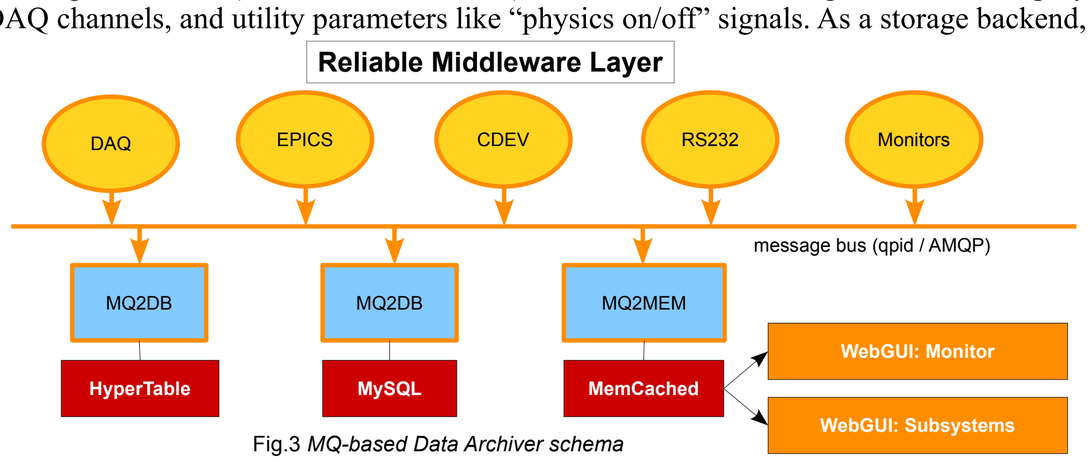


Fig.3 *MQ-based Data Archiver schema*

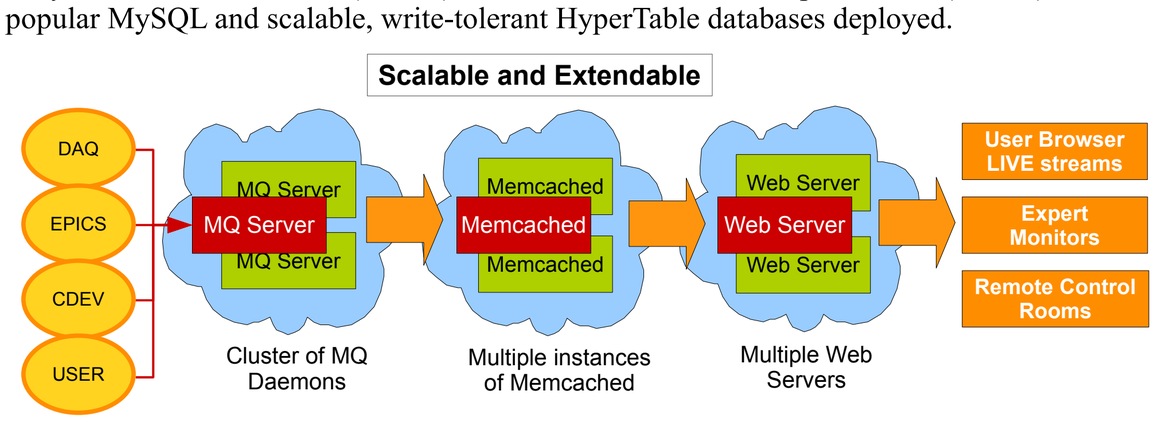
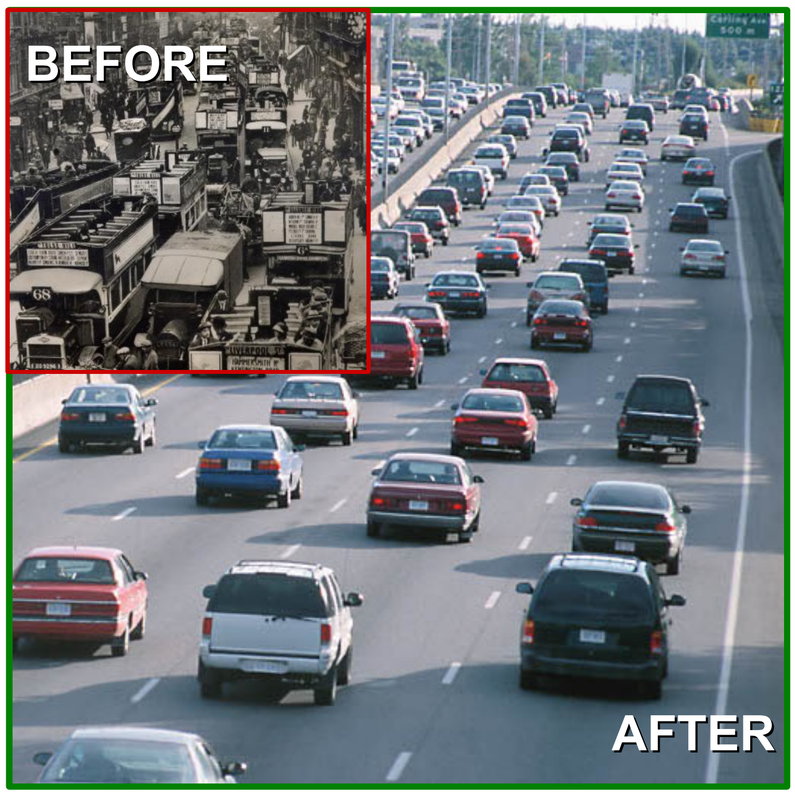


Fig.4 *Real-time monitoring data flow*

Interactive Archive Viewer

Fig.5 *AJAX-ified historical data browser interface (dbPlots)*

The MIRA infrastructure allows STAR to integrate new subsystems and their valuable meta-data in mere hours, compared to weeks (sometimes months!) previously. In RHIC Run 2012 Forward Gem Tracker group requested a fast and reliable meta-data archive viewer, and MIRA worked perfectly for that task.



Low latency streams

dbPlots is the historical data browser for MIRA (see Fig.5), which provides instant access to thousands of channels recorded in Runs 11 and 12. Of course, there are many cases where users need live access to meta-data as it is being recorded. *dbPlots Live* provides experts with a browser-based, low-latency data visualization display. Users may pause plot updates, zoom in on problematic plot areas and export resulting images directly from the web interface in a few clicks.

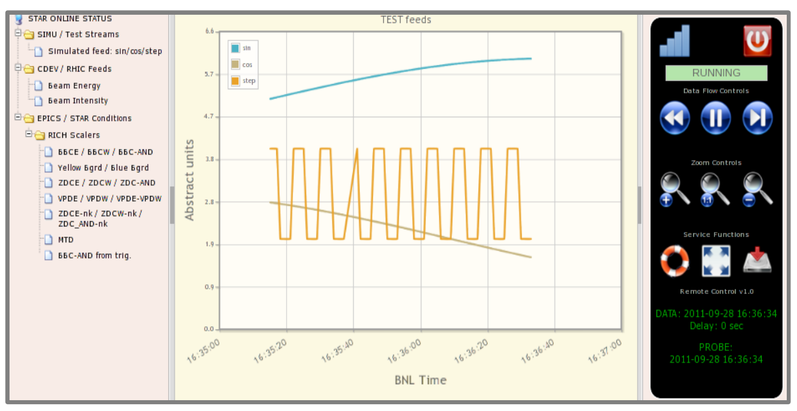


Fig.6 *Live data streams user interface (dbPlots Live)*

Service Orchestration and Control

MIRA Control Center (MCC) plays a key role in service orchestration. Adapter daemons report their states and actions to MCC, and receive guidelines for further actions via a RESTful web interface. MCC is able to raise alarms (visual, email) for administrators, making sure that no critical change in overall system performance goes unnoticed. The MCC web dashboard (Fig.7) presents an overview of the system health and status, while remaining tabs serve for detailed investigation of sub-component problems when needed. The daemon control panel allows administrators to intervene into automated service management.

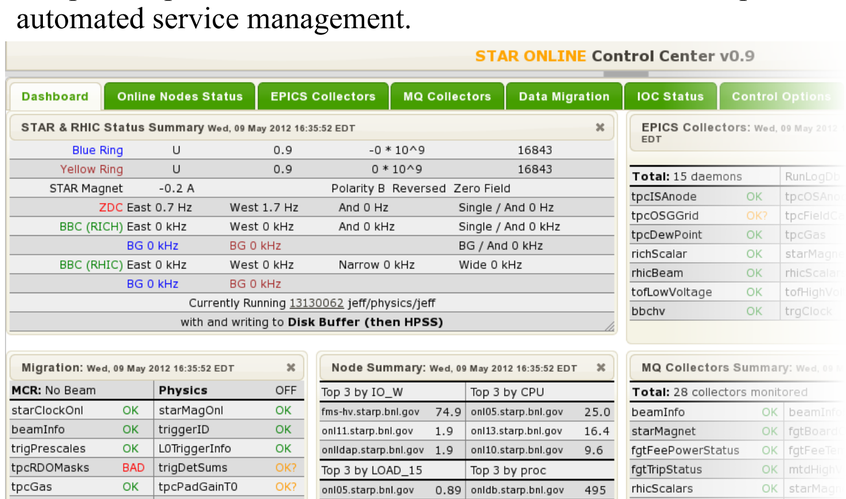


Fig.7 *Monitoring and Control System dashboard*

Future

Path forward:

- generic logging and a log browsing interface to unify and centralize currently independent online log facilities
- real-time event processing using Esper engine (currently under investigation), which will allow multicomponent event triggers and related service notification
- an online stream configuration interface to streamline new detector systems integration and several other features not reviewed in this poster

Performance

MIRA's exceptional scalability and outstanding performance were examined during Run 11 (with basic components deployed and only a few data providers active) and during Run 12 (full-fledged deployment with all run-related meta-data collectors switched to the new API).

Performance highlights:

- ✔ over 40 data providers were able to connect simultaneously, sending and receiving messages in parallel, reaching up to 2300 messages/second during peak periods
- ✔ 173 million messages had passed through the system in total, which corresponds to 2.7 billion “channel name” / “channel value” individual pairs
- ✔ 530 bytes per message, on average
- ✔ 93 gigabytes of message payload served
- ✔ average time to extract and display data for web archiver GUI (dbPlots) was 0.4s/image created, having 2-4 channels displayed on a graph
- ✔ overall stability was excellent - MQ server was stable for the whole test period of ten months of combined running; we saw no failures and no delays in message processing

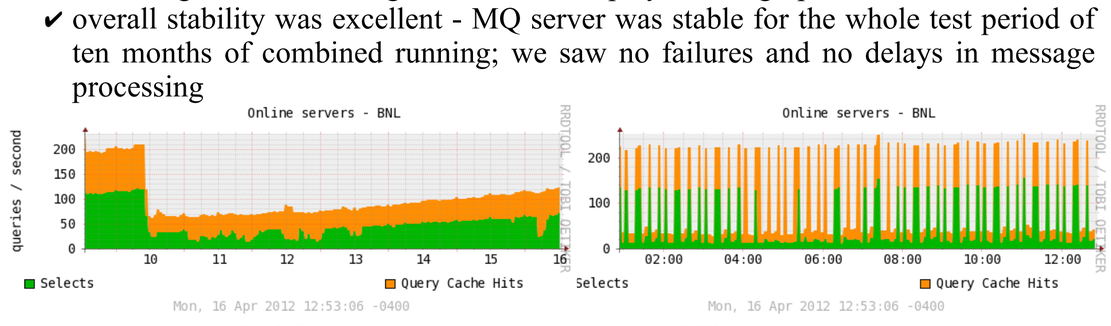


Fig.8 *Data rates plots example (week/day) – STAR online databases*