Journal of Physics: Conference Series **396** (2012) 012002 doi:10.1088/1742-6596/396/1/012002

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

D Arkhipkin, J Lauret, W Betts and G Van Buren

Physics Department, Brookhaven National Laboratory, Upton, NY 11973-5000 USA

E-mail: arkhipkin@bnl.gov, jlauret@bnl.gov, wbetts@bnl.gov, gene@bnl.gov

Abstract. The STAR Experiment further exploits scalable message-oriented model principles to achieve a high level of control over online data streams. In this paper 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 and 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.

1. Introduction

An acronym for the Solenoidal Tracker At RHIC (Relativistic Heavy Ion Collider), STAR [1] tracks thousands of particles 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.

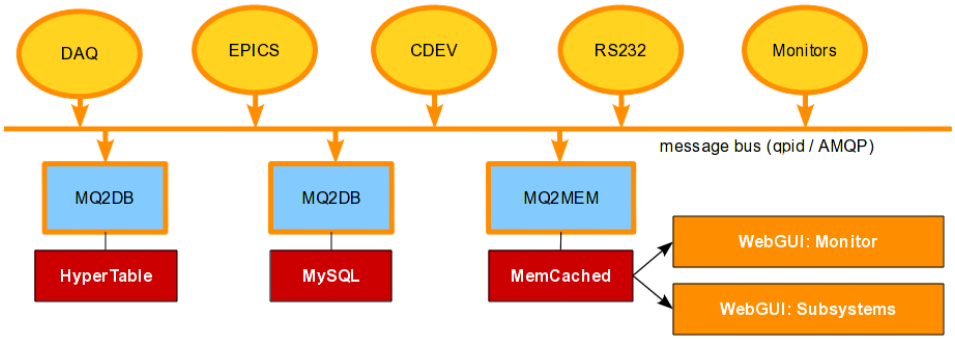

Figure 1. MIRA framework overview. AMQP-based brokers, relying on asynchronous message passing, are designed to provide good scalability for "one publisher, many subscribers" setups. Multiple database storage backends allow flexible load balancing and data accesibility (see section 2 for details), while memcached server exports meta-data for web services (see Fig. 2).

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 a Messaging Interface and Reliable Architecture (MIRA) framework, reported in this paper. It represents a complete version of the MQ-based framework, which we discussed in our previous paper [2].

MIRA is designed around an architecture for distributed systems (see Fig. 1) based on the concept of reliable message queuing (MQ). 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. Next paragraph describes framework implementation details.

2. Overview of the MIRA Framework

The Message Interface and Reliable Architecture (MIRA) framework is composed of the following:

- an AMQP-compliant client-server implementation – the qpid [3] package by Red Hat [4]
- a set of adapter-like daemons to allow conversion from various data sources to MQ format
- a set of adapter daemons to store MQ-produced data streams to database backends (MySQL, HyperTable, Sqlite)
- browser-based tools for meta-data tracking and review (dbPlots, dbPlots Live). The dataflow for such systems is illustrated in Fig 2
- 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).

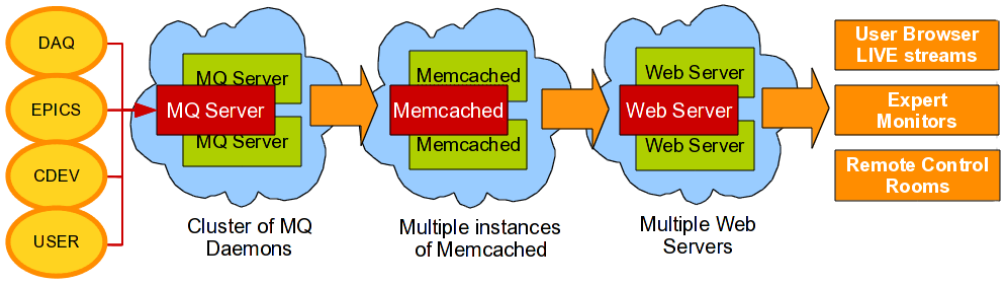

Figure 2. Real-time monitoring data flow.

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 [5]) variables, STAR status and parameters (EPICS [6]), some DAQ channels, and utility parameters like "physics on/off" signals.

As a storage backend, we used widely popular MySQL [7] and scalable, write-tolerant HyperTable [8] databases. Multiple parallel database storage use is also justified by the need to distribute load between writing backends. One can imagine having multiple MySQL servers each archiving its own portion of online meta-data, which provides native sharding, and allows

to overcome 300 IOPS limit for regular HDDs – it is cheaper to use multiple servers with regular drives instead of one powerful server with costly SSD disk array (25-30k IOPS).

As of August 2012, our framework also supports MongoDB storage adapter [9]. MongoDB is an open source document-oriented schema-free NoSQL database system, with native support for (B/J)SON [10] data formats. Schema-free capability makes MongoDB a perfect all-purpose archiver engine for meta-data collection, and native support for JSON reduces the amount of message conversions. MongoDB runs fine on Amazon Elastic Compute Cloud, so MIRA adapter for MongoDB provides a high scalability of the framework within a virtual environment.

Next, we will describe the data browsing interface and its implementation details.

3. Archiver GUI: historical and live options

dbPlots (see Fig. 3) is the historical data browser for MIRA, providing instant access to thousands of channels recorded in Runs 11 and 12. There are many cases where users need live access to the meta-data as it is being recorded.

Figure 3. Historical data browser interface - dbPlots.

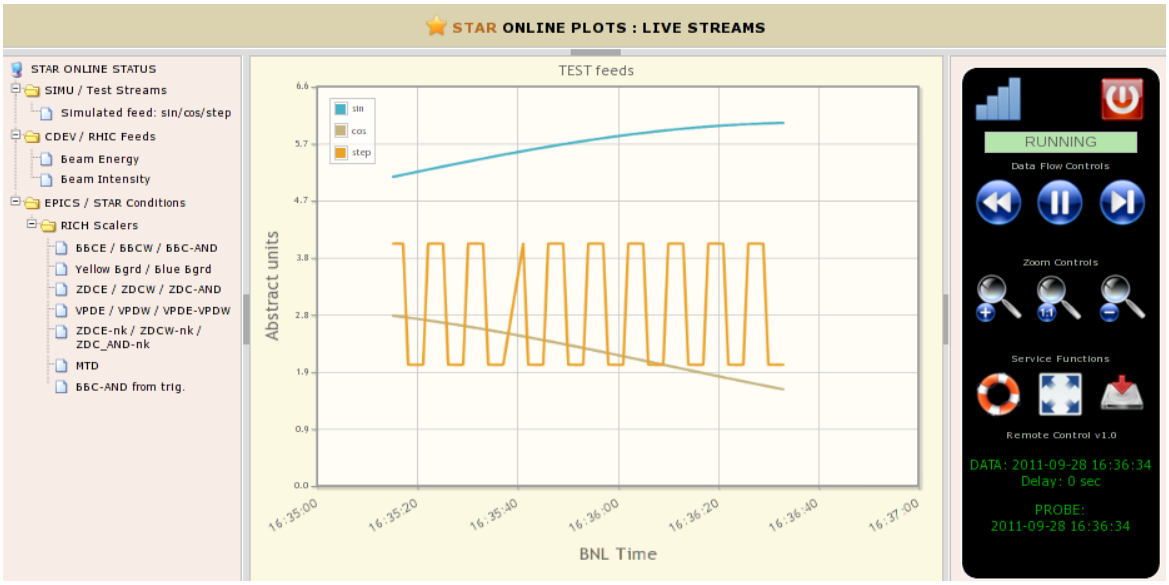

Figure 4. Live data streams user interface - dbPlots Live.

dbPlots Live (see Fig. 4) provides experts with a browser-based, low-latency data visualization display. HTTP long-polling technique is used as primary message delivery mechanism, with expected migration to WebSocket [11] technology (part of HTML5 [12] specification) later on as it gains native support in Opera and Internet Explorer browsers (Firefox and Google Chrome already have full support for WebSocket). Variable status updates are encoded using JSON format, to allow easy parsing on a client side. dbPlots Live provides rich functionality: it allows users to pause plot updates, zoom in on problematic plot areas and export resulting images directly from the web interface in a few clicks.

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, the Forward Gem Tracker group requested a fast and reliable meta-data archive viewer, and MIRA worked perfectly for that task. With the minimal investment of adding a simple meta-data publisher it allowed them to browse historical values as well as display the evolution of their system in real-time.

The service orchestration and control plays an important role in our framework. The next section describes MIRA's service orchestration concept and features implemented in the MIRA Control Center.

4. Service Orchestration and Control

The 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 [13] 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. 5) presents an overview of the system health and status, while remaining tabs serve for detailed investigation of subcomponent problems when needed. The daemon control panel allows administrators to intervene into automated service management.

In the next section we will give an overview of the performance of MIRA, and go over quantative measurements such as messages per second and uptime, illustrating reliability and scalability of the framework.

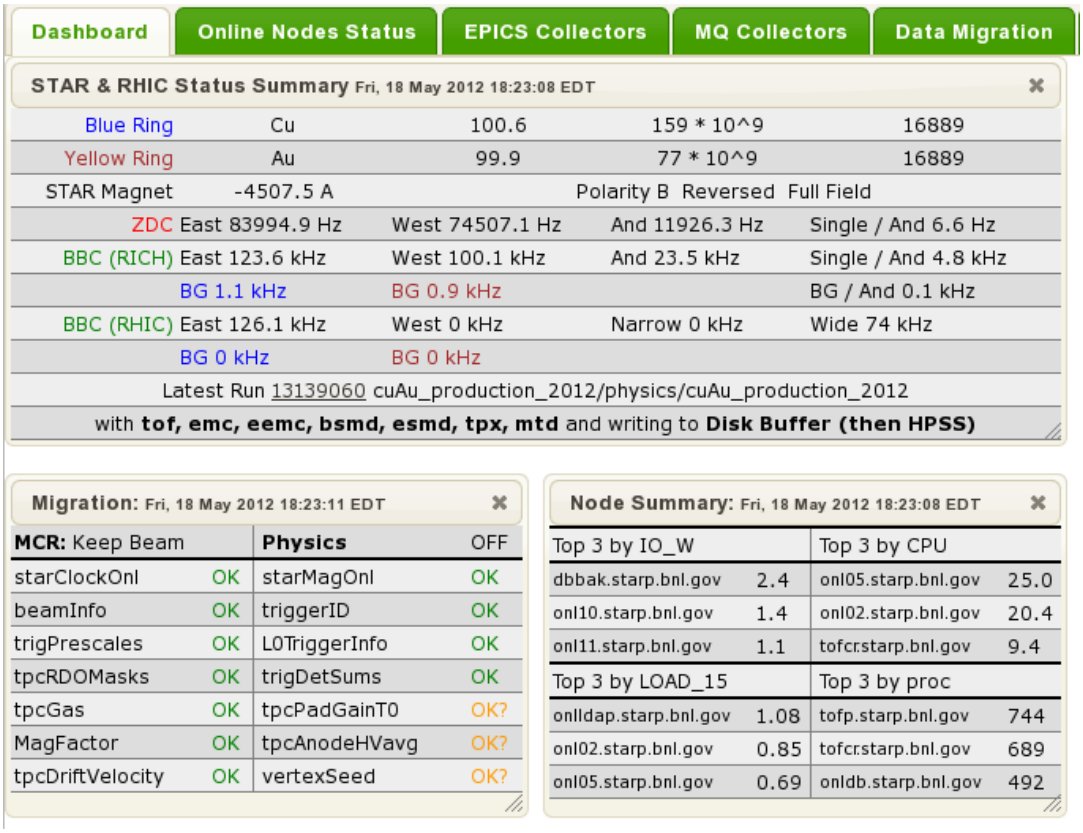

Figure 5. Monitoring and Control System dashboard, operator panel view.

5. Performance Testing and Deployment

MIRA's scalability and 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). The experiment requirements demand an ability to sustain data flow rate of two thousand messages per second, with the additional constrain that there should be no delays or lock-downs at storage backend side (no IO congestion or possible load). Time to display image graph for historical data should be maintained within one second for 1-4 channels, simultaneously requested from archive database.

Final measured performance, testing results and conditions were as follows:

- over 40 data providers were able to connect simultaneously, sending and receiving messages in parallel, reaching up to 2300 messages/second during peak periods (setup-specific, messaging system saturation was not reached)
- 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.

6. Summary and Outlook

In this work we presented an AMQP-powered Message Interface and Reliable Architecture framework (MIRA), which was successfully deployed and used in data taking production during years 2011-2012 of STAR experiment running. MIRA allowed STAR to streamline metadata collection and integration of the new subsystems, while maintaining high level of service coordination and orchestration. This scalable and flexible architecture allows much higher metadata volumes to be collected compared to the system STAR had before 2011.

Possible extensions of the MIRA framework for the year 2013 and later may include:

- generic MQ-based logging and a log browsing web interface to unify and centralize currently independent online log facilities
- real-time event processing using Esper [14] 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 work
- complementary MQTT [15] protocol support, to cover wider range of device sensors. Native support for MQTT will possibly allow us to reduce dependency on EPICS.

Acknowledgments

This work was supported by the Office of Nuclear Physics within the U.S. Department of Energy's Office of Science.

References

[1] M Harrison, T Ludlam and S Ozaki 2003 *Nucl.Instrum.Methods Phys.Res.A* **499** 235–244
[2] D Arkhipkin *et al* 2011 *J. Phys.: Conf. Ser.* **331**
[3] RHEL6 High Performance Network with MRG - MRG Messaging: Throughput & Latency (*Preprint* `http://www.redhat.com/f/pdf/MRG-Messaging-1Gig-10Gig-IB-Xeon-v2.pdf`)
[4] Red Hat [Online] URL `http://www.redhat.com/`
[5] CDEV - Common DEVice [Online] URL `http://www.jlab.org/cdev/`
[6] EPICS - Experimental Physics and Industrial Control System [Online] URL `http://www.aps.anl.gov/epics/`
[7] MySQL: the world's most popular open source database [Online] URL `http://www.mysql.com/`
[8] HyperTable: high performance, open source, massively scalable database modeled after Bigtable, Google's proprietary, massively scalable database [Online] URL `http://hypertable.org/`
[9] MongoDB: scalable, high-performance, open source NoSQL database [Online] URL `http://www.mongodb.org/`
[10] Binary JSON: binary-encoded serialization for JSON-like documents [Online] URL `http://bsonspec.org/`
[11] WebSocket Protocol [Online] URL `http://www.websocket.org/`
[12] Hypertext Markup Language, v5 [Online] URL `http://dev.w3.org/html5/spec/single-page.html`
[13] RESTful web services [Online] URL `http://www.ibm.com/developerworks/webservices/library/ws-restful/`
[14] Esper: an open source event stream processing and event correlation engine [Online] URL `http://www.espertech.com/products/esper.php`
[15] MQ Telemetry Transport [Online] URL `http://mqtt.org/`