Java Message Service (JMS) use in the Telescope Automation and Remote Observing System (TAROS)

Adam Czezowski, Anthony Green, Gary Hovey, Mark Jarnyk, Jon Nielsen, Bill Roberts, Kim Sebo, Dione Smith, Annino Vaccarella, Greg Wilson, Peter Young

Abstract. JMS is an enterprise messaging system, part of the Java 2 Platform, Enterprise Edition (J2EE). It provides distributed applications a method for asynchronously sending and receiving critical data and events. It also decouples the message delivery management subsystem from the application itself. TAROS is a distributed system that will allow the Australian National University telescopes at Siding Spring Observatory to be operated automatically or interactively over the Internet. JMS technology is used throughout TAROS, providing a communication path between the TAROS back-end software components as well as between TAROS and its external clients.

1. Introduction

TAROS (Wilson et al. 2005) is a remote observing system currently being implemented at the Mount Stromlo Observatory. We have adopted the approach of using mature technologies and have aimed for platform independence by choosing Java. Reliability of the system has been a major design goal. TAROS will permit observers to use the Siding Spring 2.3m and new SkyMapper telescopes remotely via an integrated GUI regardless of their location. It provides interactive and automated modes of operation. To ensure telescope safety, TAROS will provide fully automated environmental monitoring. It will also control the flow and archiving of data obtained from the telescopes’ instruments.

TAROS is a distributed, multi-threaded and multi-process system. It uses JMS to communicate between its native Java-based subsystems. Non-Java systems are interfaced to TAROS using JNI and RPC.

2. Architecture

TAROS uses a typical client-server configuration. Requests for observations can be submitted using a GUI (interactive mode), a CLI (scripting mode) or an External Scheduler (automated mode) from anywhere on the Internet. JMS is used as the transport layer. Observation requests can be submitted as individual commands or Observation Blocks. On the server side, TAROS comprises a number of independent processes also communicating via JMS. The use of JMS
has provided a common communication platform for both TAROS clients and TAROS server processes. The architecture for TAROS is shown in Figure 1 (a).

The function of each main TAROS component is briefly described below:

**JBossMQ** - JMS Server, component of the JBoss Application Server.

**Displayer** - supplies images taken by the data acquisition system to the observer’s image display tool. It supports image compression.

**Data Distributor** - performs data “trickling” to remote observers and data archiving on various media types.

**Command Handler** - provides decoupling of data acquisition systems and the telescope control systems from TAROS. Currently we use JCICADA, a JNI solution, to interface TAROS to the existing data acquisition software CICADA (Young 1999).

**Master Scheduler** - accepts requests for observations from external schedulers.

**Status, Health and Messages Server (SHAM)** - provides an interface to a central database of system status, health and messages for all TAROS.

**Control** - central process which starts and monitors all TAROS processes.

**Met Client** - provides meteorological data obtained from the Weather Station.

**Monitor** - monitors health of the subsystems by querying the TAROS SHAM database and provides system alerts to users via emails or SMS.

**Gateway** - authenticates TAROS clients and relays TAROS commands from the user to the rest of TAROS. Single entry point to the TAROS Server.

**External Scheduler** - submits observational requests in the form of XML based Observation Blocks to the TAROS Master Scheduler.

**GUI** - Java-based GUI for interactive telescope control.

**CLI** - Java-based command line interface to TAROS.

**Telescope Control System (TCS)** - TCS (Jarnyk & Hovey 2005) is responsible for all telescope hardware, handling telescope commands and maintaining status information such as interlocks, position and motion status.

3. **JMS Server**

The central piece of software that facilitates communication between TAROS subsystems is the JMS Server. We use the open source, Java-based application server, JBoss, and specifically its JMS component JBossMQ. JMS is a standard API for sending and receiving messages either in a point-to-point model, where each message is addressed to a specific queue, or in a publisher-subscriber model in which messages are published on topics available to anonymous subscribers. JMS enables TAROS to asynchronously send messages between its sub-systems and clients on multiple platforms, not all on the same local network. The TAROS messaging system is shown in Figure 1 (b).

4. **JMS Deployment**

**JMS Authentication** - JMS clients need to be authenticated before they can become part of the TAROS communication infrastructure. The authentication is provided by the Database Server Login Module which checks client’s credentials against the TAROS user database (TAC).
**TAROS Client Authentication** - The TAC database is used for storing TAROS client data needed by the Gateway to authenticate TAROS clients. The Gateway and TAROS clients requesting access to the TAROS server use the TAROS Authentication Protocol (TAP), which relies on JMS point-to-point communication. Communication from TAROS clients is encrypted with SSL.

**Destinations** - JMS uses destinations, queues or topics, to establish valid message delivery points. All TAROS clients send their requests to the Gateway queue. The return topic associated with the connected client is used to pass any response messages from the server. Other topics are defined to broadcast specific messages or data.

**Parameter Subscribers** - The SHAM database is a central repository for various TAROS and CICADA parameters. These parameters reflect current telescope, instruments and TAROS status. JMS is used to propagate the changes in those parameters to TAROS clients. TAROS clients register interest in a particular parameter via calls to the addSubscriber method, providing a callback method which will be executed when the parameter of interest changes. Use of JMS combined with parameter subscribers has provided a powerful and flexible solution to give feedback to the remote clients.

**Displayer** - JMS stream message is used as a transport for archived, acquisition and guiding images as well as images read off the CCD. Upon reading the image, the TAROS displayer server compresses the data and publishes them to the Real Time Display topic. Use of JMS allows implementation of image multicasting allowing multiple viewers.
Connection Class - This is the client interface that provides methods to connect to the JMS server. It was introduced to decouple clients’ connection logic from any particular communication technology.

Log4J - Logging system. We use the JMS Appender to disseminate current logging information from all TAROS systems.

Figure 2. JMS Performance

5. JMS Performance

We have measured JMS performance by sending and receiving replays for 1000 messages, on 100 MB/sec network using diverse clients (mostly Sun systems with 1.2 GHz or 0.5 GHz UltraSparc CPU). The JMS server was running on a faster Sun system. In Figure 2 we show elapsed times in milliseconds as averages: i.e. time per message, per client. We vary the number of threads (JMS sessions) per connection. The first two groups of bars are for single client. The first result is for the client running on the same host as JMS server (JVM) and the second is for the client on another host on the network (TCP). All tested clients send and receive 1000 consecutive messages in an asynchronous fashion. It is worth noting that use of multiple threads breaks the order of messages arrival. Use of SSL increases the message transfer time by around 30%.

References

