Design & Implementation of Monitoring Tool for Condor Scheduler

Raman Sehgal*, P. Shukla, Vineet Kumar and A. Chatterjee
Nuclear Physics Division, Bhabha Atomic Research Centre, Mumbai - 400085, INDIA
** email: sc.ramansehgal@gmail.com*

Introduction

Over the last decade, enormous work has been done in the field of high performance computing from systems composed of small numbers of massive devices to systems with large numbers of commodity components. This architectural shift from the few to the many is causing designers of high performance systems to revisit numerous design issues like scalability, reliability, heterogeneity, manageability, as system evolves over the time.

One of the key challenges faced by high performance distributed systems designers is the monitoring of system state. On a system having large number of compute nodes and Terabytes of data storage capacity, the failure caused by the resources demands placed on them by various applications is quite common.

To deal with node attrition and to maintain the health of the system, monitoring software must be scalable and should quickly monitor and update the system state so that the users can expedite their work by utilizing the resource efficiently. Here we are presenting the design and implementation of simple scalable monitoring tool to address these challenges for the system having condor [1] scheduler installed over them.

Architecture

This visualizer is developed as a wrapper over condor architecture. The application was developed with the view to put it over the web, hence we have used PHP for writing this wrapper script that will make use of existing condor's functionality. The application uses polling mechanism [2,3] to collect the condor pool statistics. So its architecture has two main components (1) A backend tool for collection of statistics (2) The web frontend that displays the collected statistics in graphical format.

It leverages widely used technologies such as XML for data representation, AJAX [4] for communication between frontend and backend, so that the web application can send data to, and retrieve from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page. MySql is used for storing historical data.

Publishing monitored data

All the metrics are published in XML tree format from the point of view of portability and efficiency. The XML tree used here is a four level tree in which level 0 is the root of the tree which is named as Condor. Level 1 represents different condor pools which are monitored using the tool. Level 2 represents internal nodes that shows each node in the condor pool and gives full information about the IP address of that node within that condor pool. Level 3 is the level of leaves of the tree and represents the core metrics of the nodes which are collected using condor utilities.

Because of above mentioned data structure of XML tree this application can monitor multiple condor pools at the same time. Figure 1 shows the XML tree for our prototype condor pool having 3 nodes.

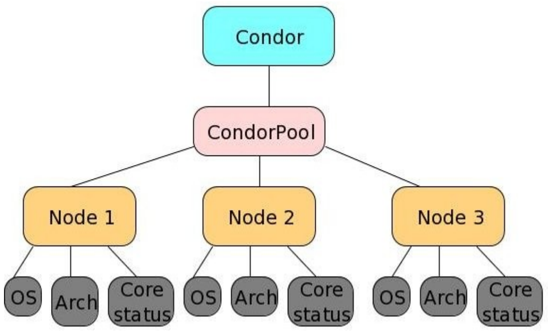

Fig 1: XML tree structure for the condor pool of 3 nodes

This XML tree is transmitted from central manager to client side using polling mechanisms through AJAX.

Web Application Model

As mentioned before that the application is developed using AJAX application model. In this model, the client make asynchronous calls to the server. The browser client interacts with the AJAX engine which then send http request to the web/XML server. The web/XML server fetch the data from backend services and form this data in XML format and return it to the AJAX engine which finally feed this data to user interface. Figure 2 shows this model:

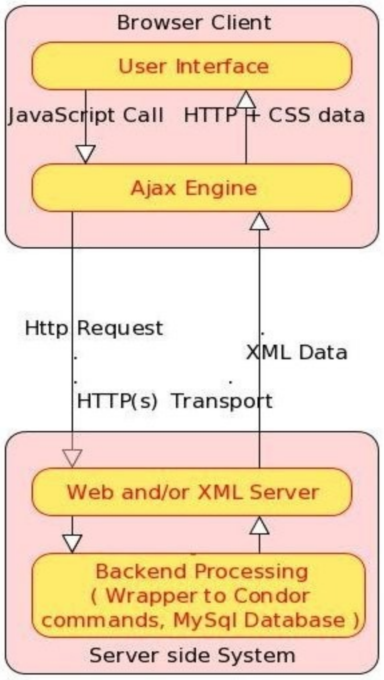

Fig 2: Ajax web application model

Functionalities of Monitoring Tool

Currently this tool supports following functionalities : (a) Status of various nodes in the condor pool (b) Job status information (c) Job history information.

Status of various nodes in the condor pool: This utiility shows the status of whole condor pool at the core level. Metrices shown here includes number of active nodes in the pool, number of cores in each node. state of each core from condor reference point which can be idle, busy, held or suspended. In addition to this information user can also see the total number of jobs, number of running jobs, number of suspended jobs etc.

Job Status information: Using this utility user can see the status of his jobs like running, waiting, suspend, or held.

Job History Information: This can be used to see the history of all the jobs of the user. This utility shows the job submission time, job completion time, job termination status etc.

Whenever the user submits the job he will be assigned a jobid along with the complete URL for job status and job history page which points to above mentioned utilities.

Conclusion

In this paper, we have presented the design, and implementation of a simple scalable monitoring tool for high performance computing systems. The system uses the polling mechanisms at the backend and AJAX calls at the web frontend, results in dynamic and quick update of the system metrices in the user interface.

Using this tool users can easily check the status of various nodes in the condor pool as well the status and history of their jobs. The implementation has been ported to a prototype condor pool of three computers and is working satisfactorily.

References:

[1] http://www.cs.wisc.edu/condor/htc.html
[2] http://en.wikipedia.org/wiki/ Pull_technology
[3] http://en.wikipedia.org/wiki/Polling_ (computer_science)
[4] http://66.14.166.45/whitepapers/programmin g/js-ajax/Ajax%20Tutorial.pdf