YRM: An Advanced Resource Manager

Daniel L. Reese Scott V. Hansen David B. Jackson Quinn O. Snell Mark J. Clement

{dreese, scott}@cs.byu.edu jacksond@mhpcc.edu {snell, clement}@cs.byu.edu

Computer Science Department
Brigham Young University

An effective computer resource manager can give industry important computing resources without an increase in hardware investment. Computer resource managers access idle time that is available on most existing computers systems within an organization without interfering with their original, dedicated purposes. Although parallel computing is common to the UNIX operating system, accessing the large number of machines not included in this particular computer hierarchy would significantly increase computational capabilities. This extra resource would be especially important for processing large and complex problems common to advanced research groups. The BYU Resource Manager (YRM) accesses all resources within a heterogeneous computer system to effectively use their idle time. YRM provides advanced features including data transmission encryption, and a standard communication protocol between all data components. It allows computer resources to be reserved in advance, and has visualization tools to assure a user-friendly environment. And it can effective use non-dedicated resources when they are available and provides meta-computing capabilities. These advanced features of the YRM resolve problems that are not well addressed by existing computer resource managers.

1 Introduction

The need to use existing computer resources more effectively is attractive both from an economic and a computational standpoint. Consequently various groups have developed software to more effectively use existing computer systems. The main emphasis has been to combine a large number of processors to emulate a single, "meta-computer," but without incurring large monetary investments.

The current approach, while successful in solving many serious problems inherent to supercomputing, are still not attractive enough to motivate wide-scale adoption. In particular, current computer resource management systems fail in varying degrees to meet the following essential criteria necessary to facilitate wide-scale use.

1.1 Criteria

Use of Heterogeneous Resources
Computer resource management should solve multiple problems on a variety of computer hardware resources. In particular, the Windows operating system has largely been ignored by the supercomputing community. A computer resource manager system should be portable, easily adapt to various problems, and have the capability to use all available hardware resources. It should also provide the ability to use non-dedicated resources when they are available.

Data Security
Secure data encryption and use authentication are absolutely necessary – no one without authorization should have access to system resources or user data. This is especially true for environments where the hardware resources and the network are shared or otherwise insecure.

Standard Communication Protocols
A standard communication protocol is essential to a computer resource management system to prevent incompatibilities between various computer systems and their software. This protocol must provide the means for describing system resources and how they are being used.

Graphical Interface
To facilitate the use and administration of any computer resource management system, a graphical

interface should be provided. The ability to visualize the complex system will assist users and administrators to use the system and its resources efficiency.

Advance Reservations
The ability to reserve resources in advance guarantees the use of the system for a specific time period. This ability provides users with more flexibility in how they use the system.

Meta-computing Capabilities
Meta-computing allows more computing resources to be shared across large physical distances. A computer resource management system must facilitate this sharing and provide advanced accounting and policy management capabilities.

1.2 Related Work

Current computer resource managers fail to completely address each of the important issues above. For instance, neither Condor [3] nor Loadleveler [4, 5], two advanced computer resource managers, support data encryption or advance resource reservations.

Legion [6], another computer resource manager, does not support Windows – that capability is currently under development. In addition, Legion does not support advance resource reservations.

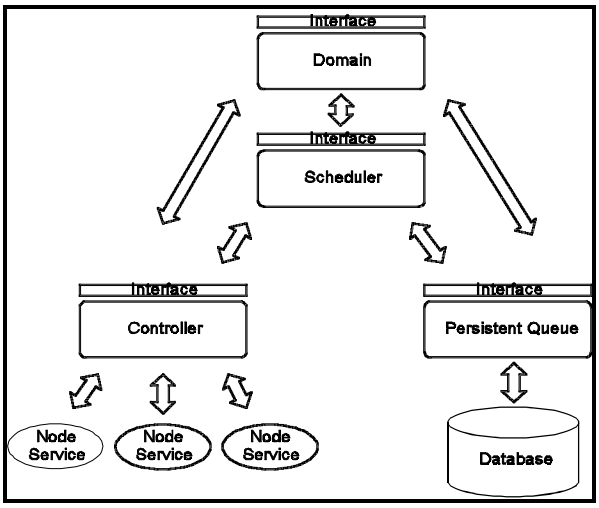
Other resource management systems such as PBS, LSF, Codine and NQE do not support data encryption or advance resource reservations. Of these, only LSF supports the Windows environment.

YRM differs from all current systems in that it is implemented in Java. Thus, YRM will run on any computer or computer system with only minor modifications. This attribute minimizes development time and allows easier software bug detection.

YRM is also the only resource management system to support data encryption during network communication. In addition, no other management system takes advantage of non-dedicated computing resources, or allows resources to be reserved in advance.

2 Design and Implementation

YRM is built from modular components that meet each of the basic criteria of an advanced resource manager as listed above. The default behavior of each component can be modified using its simple configuration file. Because of YRM's modular design each component can be completely replaced without affecting the rest of the management system. Designers can thereby use or test various algorithms, or customize a particular component for specific applications. Modularity allows extensive design flexibility and easy product evolution. This capability is further enhanced by designing each component in YRM to handles only one aspect of the resource management process [1, 10]. Figure 1 illustrates the design of YRM.



There are six major components:

- *Domain* – the top level component of YRM. It facilitates interaction between the other components and provides an interface to the computer resources.
- *Job Queue* – provides persistent storage for job information.
- *Scheduler* – decides when and where jobs will run.
- *Controller* – executes the decisions of the scheduler. It manages the resources in the system and monitors running jobs.
- *Node Service* – gives the controller access to the computing resources such as cluster workstations or supercomputer nodes.
- *Visualization Tool* – provides a powerful and user-friendly graphical interface to the system.

2.1 Domain

The domain is the top level component of YRM. It provides vital communication to the components of the YRM system. It also provides a connection between the YRM system and the available computer resources.

The domain receives job requests from users and returns information necessary for those users to confirm their requests. The domain also reports the user's job status. It also acts as a conduit through which administrators may configure the various components in the system. This does not preclude administrators from directly configuring components that are designed to be configured independently. The scheduler (described below) assures this capability.

A central point of interaction simplifies the system and maintains a consistent security architecture. The domain allows users to continue to use a standardized communication protocol regardless of the internal system demands.

2.2 Job Queue

The job queue provides persistent storage about job information that have been submitted to the system. This information is persistent to assure failure recovery. The modular nature of YRM allows job information to be created and stored in any desired location or format. For instance, users may choose a storage location based on their security preferences, on performance, or similar criteria. This flexibility allows YRM to be used in a wide variety of environments and needs. YRM stores job queued information on a disk in an encrypted format. Other possible alternatives to this default would include an LDAP directory service, or a database.

2.3 Scheduler

The scheduler decides when jobs in the system will run and what computer resources they will use. YRM takes advantage of the powerful capabilities of the Maui scheduler [7]. Users can submit jobs for execution when the required computer resources become available or reserve those resources for future use. YRM simplifies the interface to Maui to make the system easier to use, but maintains most of Maui's commonly used features. Maui can be configured through the domain component of YRM or by the tools provided with Maui, if necessary. Maui has been used extensively on UNIX systems, but a Windows port developed at BYU is available.

2.4 Controller

The controller is responsible for carrying out the decisions of the scheduler. The controller starts jobs using the resources allocated by the scheduler. It stops or cancels jobs when instructed. It also monitors each job to insure that the job only uses the resources that have been allocated to it.

The controller also manages and controls access to the resources in the domain. Physical resources are usually cluster workstations or supercomputer nodes. The controller interacts with the nodes in the domain and gives resource information to the scheduler and end users. It also keeps track of non-physical resources in the domain such as software licenses. The controller uses YRM's standard communication protocol to present an aggregate view of the system resources and allow access to them.

2.5 Node Service

The node service gives the controller access to the computing resources in the domain. The node service is a system-level service or daemon the controls each node in the domain. It is responsible for executing the controller's commands and reporting state information to the controller. It also monitors a job's use of resources and reports any unauthorized activity to the controller. YRM provides a screen saver that can activate the node service when a computing resource is idle.

2.6 Visualization Tool

The visualization tool provides a powerful and user-friendly graphical interface to YRM. This tool allows users to view information about and make changes to YRM. The visualization tool presents the resource manager's complex data in a compact graphical format. It is a web-based Java applet. Users may submit jobs using this tool. A command line utility is also available for submitting jobs. This utility translates simple Loadleveler and PBS job submission files into YRM's standard communication format and transmits them to the specified domain. The tool can also interact directly with the Maui scheduler. The visualization tool details are discussed further in section 3.4.

3 Evaluation of YRM Features

This section discusses how YRM meets each criteria for an computer resource management system.

3.1 Heterogeneity

YRM is designed to be easily ported. Most of YRM is written in Java. This allows YRM to be run on virtually any machine with minimal modification. The node service is written in Java, but it does use a small native library for operating system calls that are not supported in Java. YRM currently supports Windows and Linux, but can be ported to other operating systems easily. Computers with different operating systems may be used simultaneously.

Figure 2 shows a typical system that uses YRM to provide a cluster based supercomputing environment. The physical resources and jobs in the system are shown. The currently running jobs have been mapped onto the resources that they are using.

Computer resources in YRM may be dedicated or non-dedicated. Dedicated resources are constantly available for use by the resource management architecture. Non-dedicated resources can be configured to be available for specific time periods. During this time, YRM uses the capabilities of the node. The node service screen saver can also be configured to contact the domain when it is active and computing resources are available.

YRM can be used in situations when workstation computers sit idle for long periods of time each day, typical to a corporation or university. YRM can to tap into the numerous Windows based workstations, or into the UNIX based systems.

Figure 2 is a graph of the number available processors over time of one of BYU's computer labs. At any time during the day, there are processors available to run jobs. At night, nearly every processor is available. Taking advantage of the increase in available computing resources increases the amount of computation possible.

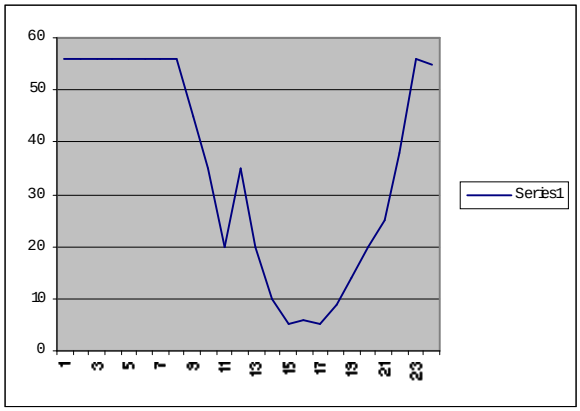
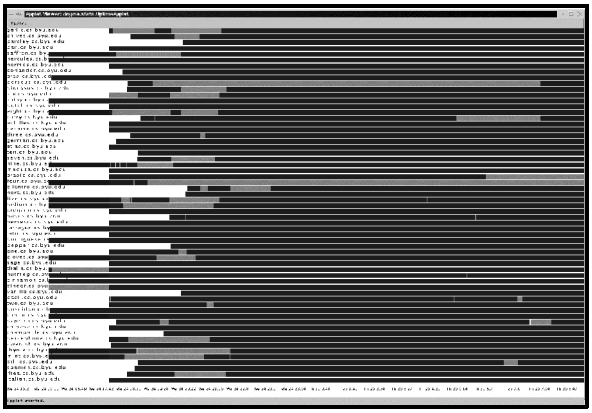


Figure 3 below shows the use of a few computers during a twenty-four hour period. Most are unused for large blocks of time. The difficulty in using these computing resources lies in determining how long a given computer will remain idle and available for use by YRM. This is an area of future research.



3.2 Security

YRM protects data by encrypting it before they are transmitted over a network. YRM uses the Diffie-Hellman [11] key agreement protocol to construct a session key to assure encryption during transmission over a network. Encrypted communication occurs between the controlling process on a node and the controller. It also occurs between the domain and any external connection. YRM' s node control process, visual tool and command line utility all support this encryption.

The Diffie-Hellman protocol does not provide authentication and is therefore vulnerable to a man-in-the-middle attack. YRM authenticates communication using a shared password. When machines exchange their public keys during the Diffie-Hellman protocol, they are encrypted using a

password-based cipher [13]. Thus, any attacker must know the password in order to complete a successful man-in-the-middle attack. The alternative to a shared password is to use third party certification such as PKI. Because most resource management architectures are fully controlled by one administrative body, shared passwords are not unreasonable. Also, this password can be changed at any time without affecting the system. Diffie-Hellman provides equivalent security to RSA [12] using equal key sizes.

Generating a session key for every communication decreases performance. YRM has the ability to cache each session key for a set amount of time. Reusing cached session keys increases performance, but allows more opportunity for security breaches. An administrator can balance these conflicting goals by configuring the time a session key will remain valid in the cache. By default, a new session key will be generated for every connection.

3.3 YSL: A Standard Communication Protocol

YRM uses BYU's Specification Language [14] (YSL) as its standardized communication format. Using a standard communication protocol increases interoperability between software programs in a resource management architecture. YSL addresses the limitations of previous resource management communication languages.

YSL describes all aspects of a resource management architecture. This includes resources, jobs, reservations, policies, users, groups, configuration information, queries and administrative commands. Customized options inherent to the software allows YSL to be expanded for future applications. The standardization efforts of the Grid Forum [15] will be supported.

YSL can also describe optional job characteristics and preferences, such as communication patterns or computational intensity – a capability that could be used by a scheduler to make smarter decisions about when and where to run a job. This is part of ongoing work under development at BYU.

Globus has developed a standardized specification language called RSL [16]. RSL is limited to describing only the static resources needed by a job or program. YSL is designed to describe all aspects of resource management. In addition, RSL syntax is not easily read by people and does not use an industry standard format such as XML.

YSL is written using XML, an industrial standard for describing information. XML is an ideal format for capturing the state of a distributed system [2]. A great effort has gone into producing generic tools to use of XML. YRM takes advantage of these tools to create, parse and manipulate YSL.

Below are two examples of YSL. The first is a description of a node and the resources it provides.

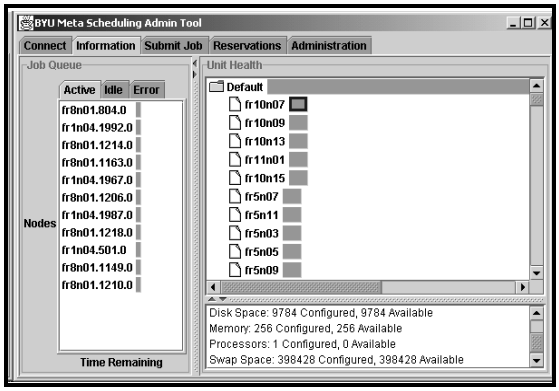
```
<node name = “orion-12” state = “idle”>
    <processor count = “2”/>
    <memory amount = “128”/>
    <swap space = “ 256”/>
    <disk space = “2048”/>
</node>
```

The second is a description of a job and its resource requirements.

```
<job name = “job1” state = “active”>
     <location
         initalDirectory = “c:\users\temp”
         inputFile = “in.txt”
         outputFile = “out.txt”/>
     <executable name = “test.exe”/>
     <arguments content = “-u dreese”/>
     <time limit = “120”/>
     <requirements>
         <node count = “2”/>
     <memory amount = “64”/>
     <swap space = “128”/>
     <disk space = “1024”/>
     <os type = “Windows”/>
</requirements>
</job>
```

3.4 Visualization

The information of the YSL screen is shown in Figure 4. The left section of the information screen shows a visualization of the job queue. The vertical axis represents the amount of resources the job requires and the horizontal axis is a logarithmic representation of that job’s time limit. The right side of the screen is a visual representation of the nodes in the domain. Jobs and nodes are color-coded according to their state.



Only information appropriate for each user is shown. Administrators see information about all jobs, whereas each user can only see information about their own jobs. Any user can cancel or place and remove holds on jobs to which they have access. In addition, administrators can explicitly start a job. Detailed information about a job can be viewed in the lower right portion of the screen by clicking the job.

Node information is displayed to all users. Selecting a job will highlight the nodes that the job is currently using Users can view detailed information about a node in the lower right portion of the screen by selecting that node. Administrators can manually make any resource unavailable for use. This method is more effective to avoid job failures than it would be to merely disconnect the node from the domain.

Presenting the complex job and node information in a compact visual representation simplifies the use of the system. This helps users to more fully understand what is happening in the system, and allows them to make more informed decisions. More detailed information is also available as needed.

3.5 Reservations

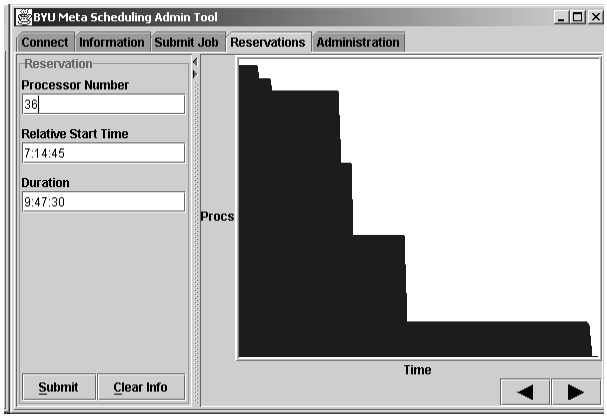
Advance reservations allow users to reserve resources for future use. They guarantee the use of resources for a specific time period. Advance reservations have a definite beginning and ending time, and the reserved resources are only accessible by the creator of the reservation.

Advance reservations are valuable to administrators who wish to dedicate a group of resources for a particular group or job. For example, a professor may want to reserve a one-hour block immediately after class for student use. Users who want to run interactive jobs during convenient times can reserve needed resources in advance during those times.

Instead of being forced to wake up whenever resources needed for a debugging session become available, a developer can reserve those resources during normal work hours.

Advance reservations are necessary for meta-scheduling environments. In order to be able to use resources at different locations together to solve a problem, all resources must be available at the same time. With advance reservations these resources can be scheduled at the same. Users may wish to schedule a telescope at the same time that processing nodes and network bandwidth are available.

Figure 5 shows the reservation screen of the visualization tool in YRM. Users can view the availability of resources over time and choose an appropriate time to reserve the needed resources. Users can drag on the availability graph and the processor number, start time and duration will be updated on the left side of the screen.



3.6 Meta-Scheduling

The YRM is able to participate in a meta-scheduling system.

4 Conclusion

YRM is an advanced computer resource management system designed to meet a large number of industrial needs. It solves problems existing in current systems.

- YRM takes advantage of all resources in a heterogeneous environment including non-dedicated nodes.
- YRM authenticates users and encrypts data.
- YRM uses YSL as its standard protocol for all communication.
- YRM has a powerful, user-friendly graphical interface.

- YRM allow users to make advanced computer resource reservations.
- YRM can participate in a meta-computing environment.

YRM uses dedicated resources effectively and will use non-dedicated resources whenever they become available. The ability to use non-dedicated resources decreases job turn around time and the time jobs are in the queue before execution. It also increases throughput.

Because of its modular and customizable nature, YRM is a tool that can generate new understanding in computer resource management research. It is sufficiently flexible and customizable to provide a robust production system. YRM also addresses and resolves problems in current resource managers.

4.1 Future Work

In a shared computer resource environment, it is important to maximize the time each resource is dedicated to supercomputing. However, to provide a stable resource pool the dedicated time should be as continuous as possible. The node service screen saver could be modified to record information about its usage patters and dynamically calculate an optimal time during which that resource would be available for dedicated use. This type of usage tracking would be more flexible and possibly more effective than static configurations.

It is possible that a more appropriate and effective method of session key caching can be developed.. Two potential alternatives are to (1) generate a new session key for each job and (2) use password-based session keys. Passwords are easier to remember and use than numerical keys and might provide a means for users to "safely" communicate with interactive jobs.

Currently, the only parallel language supported is MPICH. To provide a more useful resource management architecture [1], support for other types of parallel languages, such as PVM and HPF, could be added.

YRM has been successful in a small clustered computing environment. More information is needed about how YRM performs when deployed over a more demanding arena. Specifically, YRM needs to be tested in an open-lab and Internet environment.

Using performance surfaces [8] in YRM will allow jobs to be scheduled to use more appropriate resources. This will increase performance for certain types of jobs.

5 References

[1] James Patton Jones. NAS Requirements Checklist for Job Queuing/Scheduling Software. NAS Technical Report NAS-96-003, NAS, NASA Ames Research Center, April 1996.
URL http://www.nas.nasa.gov/Pubs/TechReports/NASreports/NAS-96-003/jms_req.html

[2] Rohit Khare and Adam Rifkin. Capturing the State of Distributed Systems with XML. In *World Wide Web Journal Special Issue on XML*, Volume 2, Number 4, Fall 1997, pg 207-218.
URL http://www.cs.caltech.edu/~adam/papers/xml/xml-for-archiving.html

[3] University of Wisconsin-Madison. Condor.
URL http://www.cs.wisc.edu/condor/

[4] IBM. Loadleveler.
URL http://www.rs6000.ibm.com/software/sp_products/loadlev.html

[5] Maui High Performance Computing Center. SP Parallel Programming Workshop.
URL http://www.mhpcc.edu/training/workshop/loadleveler/

[6] University of Viginia. Legion.
URL http://legion.virginia.edu/

[7] Maui High Performance Computing Center. Maui Scheduler.
URL http://www.mhpcc.edu/maui/

[8] Mark J. Clement, Glenn M. Judd, Joy L. Peterson, Bryan S. Morse, and J. Kelly Flanagan. Performance Surface Prediction for WAN-Based Clusters. In *Proceedings of the 31st Hawaii Interantional Conference on System Sciences*, HICSS-31, January 1998.

[9] Brigham Young University. Y Resource Manager.
URL http://ncl.cs.byu.edu/yrm/

[10] Chapin, S., M. Clement and Q. Snell. A Grid Resource Mangement Architecture. BYU Technical

Report BYU-NCL-99-100, NCL, Brigham Young University, October 1999.
URL http://ncl.cs.byu.edu/publications/grid/

[11] W. Diffie and M.E. Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, IT-22: 644-654, 1976.

[12] RSA Security.
URL http://www.rsa.com/

[13] RSA Laboratories. Password-based Cryptography Standard.
URL http://www.rsasecurity.com/rsalabs/pkcs/pkcs-5/

[14] Brigham Young University. Y Specification Language.
URL http://ncl.cs.byu.edu/yrm/ysl/

[15] Grid Forum.
URL http://www.gridforum.org/

[16] Globus. Resource Specification Language.
URL http://www-fp.globus.org/gram/rsl_spec1.html