



A New Open Resource Management Architecture in the Sun HPC ClusterTools™ Environment

Steve Sistare, Enterprise Server Products

Sun BluePrints™ OnLine—November 2002



<http://www.sun.com/blueprints>

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part No. 817-0861-10
Revision 1.0, 10/31/02
Edition: November 2002

Copyright 2002 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, California 95045 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Sun BluePrints, Sun HPC ClusterTools, Sun Enterprise, Prism, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the US and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government Rights—Commercial use. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the Far and its supplements.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2002 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, Californie 95045 Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. a les droits de propriété intellectuels relatants à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et sans la limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains énumérés à <http://www.sun.com/patents> et un ou les brevets plus supplémentaires ou les applications de brevet en attente dans les Etats-Unis et dans les autres pays.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, Sun BluePrints, Sun HPC ClusterTools, Sun Enterprise, Prism, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

LA DOCUMENTATION EST FOURNIE "EN L'ÉTAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Please
Recycle



Adobe PostScript

A New Open Resource Management Architecture in the Sun HPC ClusterTools Environment

This article presents a new architecture for the integration of the Sun HPC ClusterTools™ parallel computing environment with distributed resource management systems such as the Sun™ Grid Engine system. This new architecture achieves a tight integration with multiple distributed resource management systems in a uniform and extensible framework, which means that any of the popular management systems may be used to launch and monitor Sun™ MPI parallel jobs.

Unlike previously available loose integrations, tight integrations allow a resource manager (RM) to:

- Accurately measure resources used by the parallel processes
- Terminate jobs that exceed resource limits
- Generate accurate accounting information for multiprocess jobs

We have implemented tight integrations with Sun Grid Engine software, PBS from Veridian Systems, and LSF from Platform Computing. We provide examples showing correct resource accounting, ease of use to launch and debug Sun MPI jobs under these systems, and the improvements in behavior that result from the tight integration.

This article consists of the following topics:

- “Introduction to Sun HPC ClusterTools” on page 2
- “Sun CRE Infrastructure” on page 3
- “Integrating a Resource Manager” on page 5
- “New Integration Architecture” on page 8
- “Usage” on page 10

- “Parallel Tool Support” on page 13
 - “Other Features” on page 14
-

Introduction to Sun HPC ClusterTools

The Sun HPC ClusterTools environment enables users to develop, debug, and deploy multiprocess distributed-memory applications. The environment includes:

- Highly optimized version of the industry standard MPI (message passing interface)
- Sun™ Cluster Runtime Environment (Sun CRE) for launching and monitoring parallel jobs
- Prism™ multiprocess debugger
- Sun Scalable Scientific Subroutine Library

Applications developed with this environment are often put into production using a resource management system. High performance computing centers use distributed resource management systems such as Sun Grid Engine software to share resources fairly and accountably, and to guarantee that every job can obtain the resources it needs to run at maximum efficiency. To properly monitor a job’s resource consumption, the management system must be the agent that launches the job. These RM systems do not know how to launch multiprocess jobs such as MPI applications. Instead, they reserve the resources, and then ask the native parallel environment to launch the job. As a result, the RM has no visibility into the new processes and cannot monitor their resource consumption. This means the RM cannot generate accurate accounting records, and cannot terminate jobs that use more than their allowed share of resources. The result can be poor performance for all jobs on the system due to overcommitment of resources. This is commonly referred to as a loose integration between the RM and the parallel environment.

Vendors have solved this problem by communicating the process information between the parallel environment and the RM, thus achieving a tight integration. For example, Sun provided a tight integration between Sun MPI and LSF in the Sun HPC ClusterTools 3.0 release. The disadvantage of this approach is that it tends to yield point solutions that are not easily extensible to additional resource management systems; yet it is critical that the parallel environment support all of the commonly used systems. The RM is an integral component of a production environment, and vendors cannot expect sites to adopt a different system as a prerequisite for running parallel jobs.

Sun CRE Infrastructure

Sun CRE is a full-featured, robust parallel environment whose purpose is to support multiprocess jobs in general, and Sun MPI jobs in particular. Sun CRE consists of a set of daemons that run on each node of a cluster. Users and programs interact with these daemons using commands such as `mprun` to start a parallel job, `mpps` to see the status of currently running jobs, and `mpkill` to signal or abort a job. In addition, Sun CRE provides parallel standard I/O: It broadcasts input typed at the controlling TTY to all processes, and it collects output from all processes and redirects it to the standard output of the `mprun` command.

Sun CRE monitors jobs, and cleanly terminates all the processes in a job if any process exits abnormally. This is one of the main advantages to having a parallel environment. In contrast, parallel libraries, such as the public domain MPICH library, that do not rely on a parallel environment are prone to losing processes when a job fails, and it is the user's responsibility to find and terminate them.

Sun CRE also sets up the parallel infrastructure that other Sun HPC ClusterTools components depend on for correct operation. This includes a database that contains both persistent information about the cluster configuration and transient information about currently running jobs. Sun HPC ClusterTools components access this database for a variety of purposes. For example, the Prism parallel debugger uses it to attach to running jobs.

The structure of Sun CRE is shown in FIGURE 1, which illustrates the sequence of events that take place when a new job is started.

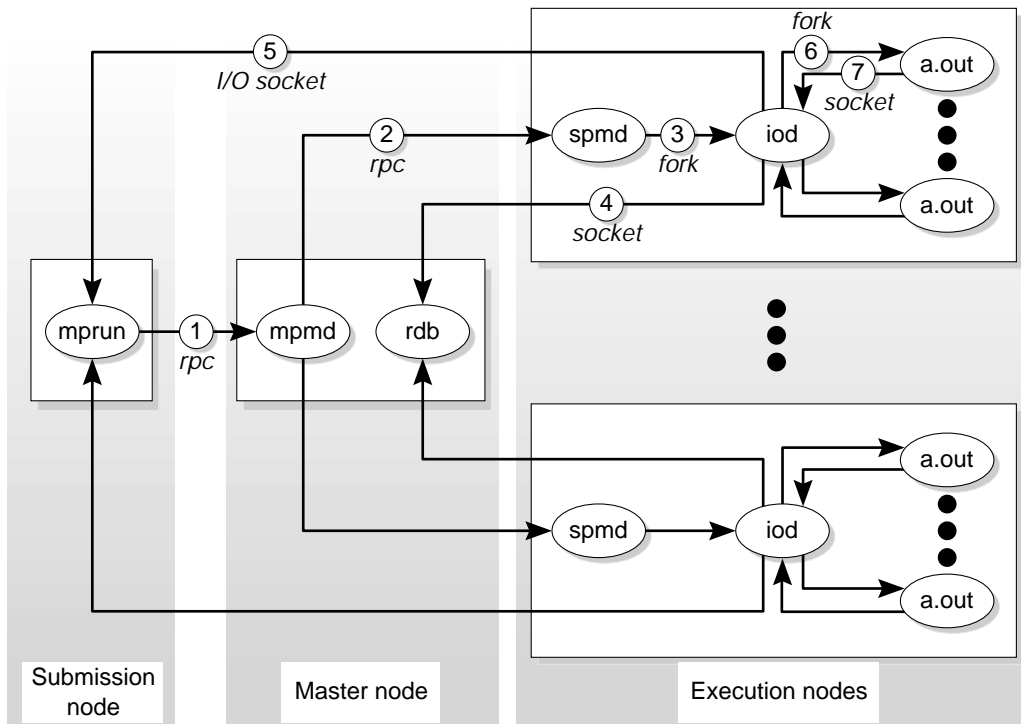


FIGURE 1 Actions From the `mprun` Command

The `mprun` command starts a new parallel job by contacting the persistent master daemon `mpmd` in step 1. The `mpmd` daemon in turn contacts the persistent `spmd` daemon on each host in step 2, and each `spmd` daemon creates an `iod` process in step 3 that is responsible for creating and monitoring the application processes on that host. In steps 4 and 5, `iod` establishes a socket to the `mprun` process to handle the application's standard input and standard output, and also connects to the persistent database daemon called `rdb`. Each `iod` forks several `a.out` processes in step 6. These execute the user's parallel application. Each `a.out` creates a "query socket" back to its parent `iod` in step 7. The query socket is the application's connection to the parallel environment; for example, requests over this socket allow a process to find its peers at application initialization time.

For clarity, FIGURE 1 shows the submission node, master node, and execution nodes as separate host systems. In practice, however, any of the execution nodes might also be submission nodes or the master node.

The Sun CRE architecture is designed for scalability. The multiple `iod` processes operate in parallel, and can start a large number of application processes simultaneously. In the HPC ClusterTools 5 software release, Sun CRE can launch a job containing 2048 processes spread across 256 hosts.

Integrating a Resource Manager

Each RM system defines a command that is used to submit a job script, which is a shell script on UNIX® systems. To submit a Sun MPI job, the user includes an `mprun` command in the job script. The arguments required by the `mprun` command might vary depending on the type of integration and the target resource management system.

Loose Integrations

Loose integrations are currently possible between all of the major RM systems and MPI implementations [1][2]. In a loose integration, the RM reserves the parallel resources, creates a host list that is saved in a file or environment variable, and executes the job script on the first host in the list. The job script invokes the `mprun` command, passing the host file as an argument. A loose integration works adequately when all users voluntarily stay within queue limits, and when programs are bug free and never spin out of control—that is, not in this world! Even in a hypothetical world where everyone is well behaved, the system cannot keep accounting records that reflect actual usage.

Tight Integrations

The key to achieving a tight integration is making the RM aware of the parallel processes. One solution is for the RM to walk the process tree and find all descendants of the job script, which include `mprun` and its children. This works on a single host cluster for versions of MPI in which the `mpirun` command creates the MPI processes directly using the `fork` and `exec` system calls. The public domain MPICH implementation is a good example [3]. PBS implicitly takes this approach, by looking for all processes on a host with the same session ID. However, this approach fails when multiple hosts are involved, or when `mpirun` must contact a daemon in the parallel environment to cause the process launch.

An alternate solution is to have the parallel environment communicate process identifiers back to the RM. However, it is more difficult and less accurate to monitor and control processes that are not children of the monitoring agent, because system calls such as `getrusage` and `nice` cannot be used.

Yet another solution is to eliminate the separate parallel environment and migrate the parallel environment features into the RM system. This was our previous approach for achieving a tight integration between Sun MPI and LSF. This was a large effort, and not one we wish to repeat with other resource management systems. Also, Sun CRE is still required because many sites do not use LSF, and there is an ongoing cost to maintaining the Sun HPC ClusterTools infrastructure in both implementations. It takes twice the effort to add features which require new infrastructure support, and regression testing costs also double. These are real considerations for a systems vendor. In addition to these practical considerations, it is not elegant to require multiple implementations of the parallel environment.

Resource Usage Examples

The differences between a loose and a tight integration can be illustrated with a few simple examples, for which we use PBS. (Examples using Sun Grid Engine software or LSF would show the same effects.) The cluster used for these examples consists of four Sun Enterprise™ 450 servers with four CPUs each. For now, ignore the differences in syntax for each example; they will be explained shortly.

In CODE EXAMPLE 1, a CPU-intensive MPI application is submitted for execution on 16 processors. Here we show results based on a loose integration:

CODE EXAMPLE 1 Inadequate Resource Accounting In a Loose Integration

```
% cat loose.sh
  mprun -np 16 -m $PBS_NODEFILE reactdiff
% qsub -l nodes=4:ppn=4 loose.sh
...
% qstat -f | grep resources_used
resources_used.cput = 00:00:00
resources_used.mem = 5256kb
resources_used.vmem = 7672kb
resources_used.walltime = 00:12:17
```

We submit the job script `loose.sh`, and towards the end of the run we check the status of the job using `qstat`. Despite running for 12 minutes, the CPU time shown is 0! This is because PBS can only see the `mprun` process, which is idle during the entire run.

Now see what happens with our tight integration in CODE EXAMPLE 2. Here, `qstat` shows that the total CPU time is approximately 16 times the wall-clock time, which is the desired result.

CODE EXAMPLE 2 Proper Resource Accounting In a Tight Integration

```
% cat tight.sh
  mprun -np 16 reactdiff
% qsub -l nodes=4:ppn=4 tight.sh
...
% qstat -f | grep resources_used
resources_used.cput = 02:41:38
resources_used.mem = 138520kb
resources_used.vmem = 250304kb
resources_used.walltime = 00:11:54
```

An RM must be able to account for all used resources in order to enforce limits on consumption. This is shown in CODE EXAMPLE 3, in which each process of a parallel program allocates a block of memory.

CODE EXAMPLE 3 Resource Limit Enforcement

```
% qsub -l nodes=3:ppn=3,vmem=30mb -I
qsub: waiting for job 64.hpc-4 to start
qsub: job 64.hpc-4 ready
%
% mprun -np 9 memtest 10
Rank 0 on hpc-6: allocating 10 Mbytes
Rank 1 on hpc-6: allocating 10 Mbytes
Rank 2 on hpc-6: allocating 10 Mbytes
Rank 3 on hpc-5: allocating 10 Mbytes
[...]
=>> PBS: job killed: vmem 47570944 exceeded limit 31457280
Aborted.
```

The program is run in interactive mode using the PBS `-I` flag, under the tight integration, with the restriction that the job may use no more than 30 megabytes of aggregate memory. A short time after this limit is exceeded, PBS kills the job. (The job temporarily exceeds the limit, because PBS only measures resources at regular intervals.) A loose integration would not be able to detect this condition, and the program would continue to run. In CODE EXAMPLE 3, the memory restriction is specified by the user at job submission time, but the system administrator could also define the restriction as a property of the queue with equivalent results.

The shortcomings of the loose integration schemes drove us to develop the following architecture.

New Integration Architecture

The key enabling element in our architecture is that it moves the responsibility for launching processes from the parallel environment to the RM, while maintaining the infrastructure of the environment. This solves the fundamental problem of giving the RM visibility into the parallel processes, and enables the proper monitoring found in a tight integration. Our approach requires that the RM export three basic capabilities:

- Ability to list the set of allocated resources (hosts). This allows the parallel environment to start its own daemon infrastructure on the proper set of hosts.
- Ability to launch a new process under RM control. The parallel environment calls the RM rather than forking a new process.
- Ability for that process to identify its containing job. The launched process must rendezvous with the parallel environment, and needs a means of identifying itself to the environment.

We find these capabilities are ubiquitous amongst distributed resource management systems. TABLE 1 shows the interfaces that export these capabilities for our target systems[4] [5] [6] [7].

TABLE 1 Resource Manager Interfaces

| Resource Manager | Host List | Job ID | Spawn Method |
|------------------|--------------|-----------|--------------|
| LSF | LSB_HOSTS | LSB_JOBID | pam |
| Sun Grid Engine | PE_HOSTFILE | JOB_ID | qrsh-inherit |
| PBS | PBS_NODEFILE | PBS_JOBID | tm_spawn() |

We have defined abstract function call interfaces for each of these capabilities that are called during job startup, as we will describe shortly. The interfaces for a particular RM are collected into a single plug-in library that is dynamically loaded by the parallel environment using the `dlopen` system call. Thus, the environment is insulated from the implementation details of the RM. The plug-in library is small; it requires fewer than 250 lines of source code for each RM that we implement. The plug-in architecture allows customers and other third parties to implement tight integrations with additional RMs without modifying the source code of Sun CRE. The plug-in interface specification is described in the `xrm.3` manual page in the HPC ClusterTools 5 software documentation.

The interaction between Sun CRE and PBS serves to illustrate the specifics of this architecture, and is shown in FIGURE 2. (The interaction with other RM systems is similar).

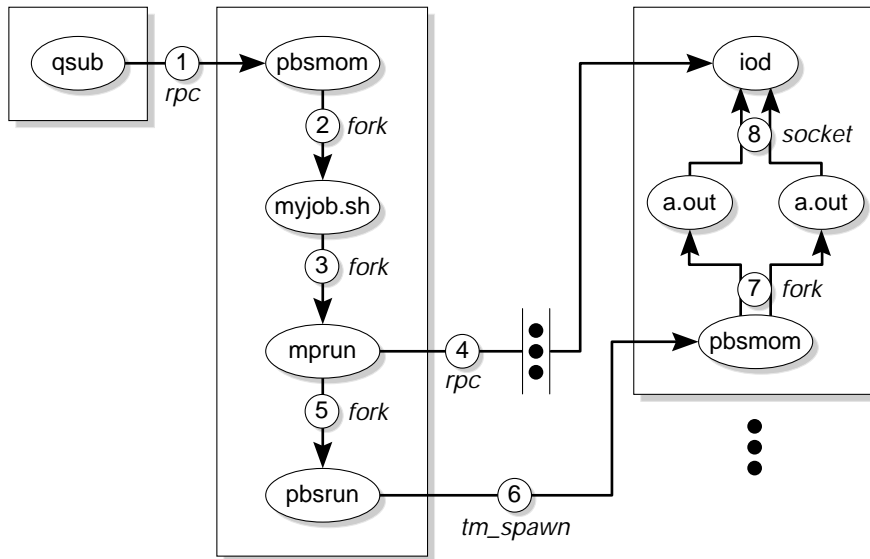


FIGURE 2 Interaction Between Sun CRE and PBS During Job Startup

The user submits a job with the `qsub` command of PBS, which contacts the `pbsmom` daemon in step 1. The `pbsmom` daemon forks the job script `myjob.sh` (step 2), which invokes an `mprun` command (step 3). In step 4, `mprun` reads the list of allocated hosts, and performs the Sun CRE daemon setup on these hosts as was shown in FIGURE 1 (these steps are elided in FIGURE 2 for clarity). Unlike in FIGURE 1, `iod` no longer forks `a.out`. In step 5, `mprun` invokes a launcher program which calls the PBS `tm_spawn` function (step 6), which contacts another `pbsmom` daemon. The `pbsmom` daemon forks the `a.out` process in step 7. Lastly, the process determines its job ID, finds the `iod` that is associated with this job, and establishes a socket connection to that `iod`. This becomes the query socket over which the process obtains services from the parallel environment. Although our PBS integration calls the launcher from the single `mprun` process, we could instead make simultaneous calls to the launcher from multiple `iod` instances, which yields better startup scalability. This is done in our Sun Grid Engine software integration. An RM's ability to handle a distributed launch is a property in our plug-in interface.

Our new architecture has a number of advantages:

- Achieves a tight integration with multiple RMs in a uniform way. Differences between RM systems are abstracted in a plug-in library.
- Allows the RM to have a parent-child relationship with the processes it manages.
- Allows fast parallel startup of jobs.
- No modifications were made to the resource management software.

- Gives the parallel environment and the RM visibility into a job. Commands of both systems can be used to view and signal jobs.
- Complete infrastructure of the parallel environment is available to parallel jobs and tools.

Usage

Our new implementation looks much like the old from a user's point of view. The user submits jobs using the same RM commands and syntax as previously, and the submitted job script still contains an `mprun` command. The difference is that we have added a `-x rm` option to the `mprun` command, where `rm` is the name of the resource management system being used. This identifies the plug-in module that CRE should use to interact with the RM. If the `-x` specifier is omitted, then `mprun` uses the RM listed in the `default_rm` entry in the file `/opt/SUNWhpc/conf/hpc.conf`. System administrators should set the `default_rm` entry to reflect the RM that is in use at their site.

The `-np n` option is still allowed, but is no longer required, because `mprun` can infer the number of processes to run by communicating with the RM via the plug-in library.

Job submissions with each RM are shown in the following examples. These examples assume that `default_rm` has been defined appropriately. The job consists of a four-process MPI program in which each rank (that is, process) prints its hostname. The only significant difference between each example is the syntax of the job submission command, which should be familiar to current users of each system.

CODE EXAMPLE 4 Job Submission Example on Sun Grid Engine Software

```
% cat myjob.csh
#!/bin/csh -f
mprun -np 4 myprog.x
% qsub -pe cre 4 myjob.csh
your job 33 ("myjob.csh") has been submitted
% cat myjob.csh.o33
Sun Microsystems Inc.   SunOS 5.8           Generic February 2000
Rank 0 is on hpc-6
Rank 1 is on hpc-6
Rank 2 is on hpc-7
Rank 3 is on hpc-7
logout
```

CODE EXAMPLE 5 Job Submission Example on LSF Software

```
% cat myjob.csh
#!/bin/csh -f
mprun -np 4 myprog.x
% bsub -n 4 -q fast -o out myjob.csh
Job <739> is submitted to queue <fast>
% cat out
Job <myjob.csh> was submitted from host <hpc-6> by user <sistare>
...
Rank 0 is on hpc-6
Rank 1 is on hpc-6
Rank 2 is on hpc-7
Rank 3 is on hpc-7
```

CODE EXAMPLE 6 Job Submission Example on PBS Software

```
% cat myjob.csh
#!/bin/csh -f
mprun -np 4 myprog.x
% qsub -l nodes=2:ppn=2 myjob.csh
22.hpc-6
% cat myjob.csh.o22
Sun Microsystems Inc. SunOS 5.8 Generic February 2000
Rank 0 is on hpc-6
Rank 1 is on hpc-6
Rank 2 is on hpc-7
Rank 3 is on hpc-7
logout
```

The user may instead request an interactive job, in which case the job submission command brings up an interactive shell containing the allocated resources. The `mprun` command is typed at the shell prompt in this case, and multiple `mprun` commands can be used within the same interactive session. The following code examples show all three RMs.

CODE EXAMPLE 7 Interactive Job Example on Sun Grid Engine Software

```
% qsh -pe cre 4
waiting for interactive job to be scheduled ...
Your interactive job 24 has been successfully scheduled.
% mprun -np 4 myprog.x
Rank 0 is on hpc-6
Rank 1 is on hpc-6
Rank 2 is on hpc-7
Rank 3 is on hpc-7
% mprun -np 4 hello.x
Rank 0 says hello
```

CODE EXAMPLE 7 Interactive Job Example on Sun Grid Engine Software (*Continued*)

```
Rank 1 says hello
Rank 2 says hello
Rank 3 says hello
```

CODE EXAMPLE 8 Interactive Job Example on PBS Software

```
% qsub -l nodes=2:ppn=2 -I
qsub: waiting for job 20.hpc-6 to start
qsub: job 20.hpc-6 ready
Sun Microsystems Inc. SunOS 5.8 Generic February 2000
% mprun -np 4 myprog.x
Rank 0 is on hpc-6
Rank 1 is on hpc-6
Rank 2 is on hpc-7
Rank 3 is on hpc-7
```

CODE EXAMPLE 9 Interactive Job Example on LSF Software

```
% bsub -n 4 -q short -Is csh
Job <24559> is submitted to queue <short>.
<<Waiting for dispatch ...>>
<<Starting on hpc-7>>
% mprun -np 4 myprog.x
Rank 0 is on hpc-7
Rank 1 is on hpc-7
Rank 2 is on hpc-6
Rank 3 is on hpc-6
```

TABLE 2 summarizes the commands that are used to submit a job under each system.

TABLE 2 Job Submission Commands

| Resource Manager | batch | interactive |
|------------------|-------|--------------|
| LSF | bsub | bsub -Is csh |
| Sun Grid Engine | qsub | qsh |
| PBS | qsub | qsub -I |

Parallel Tool Support

With our new architecture, it is easier to implement and use layered tools such as parallel debuggers in concert with an RM. The debugger implementation can target Sun CRE services to launch or attach to parallel jobs, and Sun CRE handles the details of interacting with the RMs. Previously, separate implementations were required for each supported RM.

The new architecture supports interactive debugging and tuning sessions using the Prism [8] and TotalView parallel debuggers. These tools are as easy to use with an RM as they are in an unmanaged environment. The only difference is the addition of the job submission command. Examples are as follows:

Prism debugger with Sun CRE:

```
% mprun -np 4 prism myprog.x
(The Prism GUI appears...)
```

Prism debugger with Sun Grid Engine Software:

```
% qsh -pe cre 4
waiting for interactive job to be scheduled ...
Your interactive job 73 has been successfully scheduled.
% mprun -np 4 prism myprog.x
(The Prism GUI appears...)
```

TotalView debugger with Sun CRE:

```
% totalview mprun -a -np 4 myprog.x
(The TotalView GUI appears...)
```

TotalView debugger with Sun Grid Engine Software:

```
% qsh -pe cre 4
waiting for interactive job to be scheduled ...
Your interactive job 122 has been successfully scheduled.
% totalview mprun -a -np 4 myprog.x
(The TotalView GUI appears...)
```

Users can run and re-run parallel programs from within the same debugger session. Because we have a tight integration, resources of the target process are properly measured, as shown in FIGURE 3.

```
% qsub -l nodes=3:ppn=2 -I
qsub: job 74.hpc-4 ready
% prism -C -np 6 julia
Welcome to Prism, Version 6.2.
(prism all)
(prism all) run
Running:
/home/jttest/Src/memtest/julia
Processes 0:5: Process exited with 0
(prism all) run
Running:
/home/jttest/Src/memtest/julia
Processes 0:5: Process exited with 0
(prism all) quit

% qstat -f
Job Id: 74.hpc-4
...
resources_used.cput = 00:00:00

% qstat -f
Job Id: 74.hpc-4
...
resources_used.cput = 00:21:24
```

FIGURE 3 Resource Accounting Under the Prism Debugger

The Prism debugger is itself a parallel job, but it uses little CPU time, as shown by the first `qstat` command in the figure. Once the application runs, however, subsequent `qstat` commands show that PBS accounts for target process CPU time as part of the debugger's parallel job.

Other Features

Our implementation of the tight integration architecture offers a number of additional features that improve the usability of the system, including:

- Job integration
- Support for threads placement
- Verbose output

Job Integration

A nice property of our implementation is that commands from both Sun CRE and the RM can be used to monitor and control a job, so users can choose the command set they prefer. Further, we arrange that the job identifier shown by Sun CRE

matches the identifier chosen by the RM, so that users can see the correspondence between jobs in each system. FIGURE 4 illustrates that both the PBS `qstat` command and the Sun CRE `mpps` command have a view of a running job named `53.hpc-4`.

```

% qstat
Job id          Name          User          Time Use  S  Queue
-----
53.hpc-4       tight.sh      jtest         0 R  dqe

% mpps -epf
JOBNAME        NPROC  UID    STATE  STIME    AOUT    ARGS
pbs.53.hpc-4   16     jtest  RUN    15:21:59 reactdiff ./reactdiff
              RANK   PID    STATE  NODE
              0     27942  RUN    hpc-7
              1     27943  RUN    hpc-7
              2     27944  RUN    hpc-7
              [...]
              15    654   RUN    hpc-4

```

FIGURE 4 Example of a Job Visible in Both Sun CRE and the Resource Manager

Thread Placement

Developers of parallel applications are increasingly interested in programming models that mix multithreaded parallelism with multiprocess MPI parallelism. However, to date there has been no easy way to run such an application under an RM and guarantee that the MPI processes will be placed on hosts with adequate spare CPUs to handle the spawned threads. To solve this problem, we have extended the syntax of the `mprun -np n` option to allow the form `-np PxT`, where P is the number of MPI processes to launch, and T is the number of threads that will run in each process. When `mprun` is invoked under an RM, it takes the first T resources from the host list, starts a single MPI process there, takes the second T resources, starts a single MPI process there, and so on.

Verbose Output

A system administrator must perform some manual configuration steps to enable the integration between a resource manager and Sun CRE. As an aid in debugging possible configuration problems, we have added a `-v` verbose option to `mprun`, which causes `mprun` to print extra information about its interactions with the RM.

Conclusion

A tight integration between the RM and the parallel environment offers significant benefits over a loose integration. We have implemented a robust and open architecture in Sun CRE that enables tight integrations with multiple RMs, and we have used that architecture to provide tight integrations with the Sun Grid Engine system, PBS, and LSF. The architecture is extensible, so third parties can implement their own integration with their own favored RM. The architecture adds to a growing set of open interfaces in the HPC ClusterTools environment, making it an attractive foundation for building open and integratable HPC solutions.

The software described here is part of the Sun HPC ClusterTools 5 software release, which is available at:

<http://www.sun.com/servers/hpc/software/tryandbuy.html>

About the Author

Steve joined Sun in 1997 as a software architect in the High End Software group, where he is responsible for the architecture of the Sun HPC ClusterTools development environment. He has been working on tools, communication libraries, and run-time environments for parallel computing for over 10 years, including previous experience at Thinking Machines Corporation, where he worked on the data-parallel C* compiler and the Prism parallel debugger. Steve received his Ph.D. in Computer Science from Harvard University in 1991.

Acknowledgements

The author would like to recognize the following individuals for their work which made this article possible:

- Dave Plauger, High End Software group
- Nick O'Donnell, High End Software group
- Fritz Ferstl and Andreas Haas, Sun Grid Engine team

References

- [1] Byun, C. "Loose Integration of Sun HPC ClusterTools with Sun Grid Engine Software," *Sun User's Performance Group*, April 2001.
- [2] Byun, C.; Duncan, C.; Burks, S. "A Comparison of Job Management Systems in Supporting HPC ClusterTools," *Sun User's Performance Group*, October 2000.
- [3] Gropp, W.; Lusk, E.; Doss, N.; Skjellum, A. "A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard," *Parallel Computing*, volume 22, number 6, pp. 789-828, Sep. 1996.
- [4] Sun Grid Engine Software Download: <http://sun-www.central/software/gridware/download.html>
- [5] LSF Parallel User's Guide: <http://www.platform.com/platform/platform.nsf/webpage/LSFDoc?OpenDocument>
- [6] Portable Batch System: <http://www.OpenPBS.org/docs.html>
- [7] PSCHED: An API for Parallel Job/Resource Management:
<http://parallel.nas.nasa.gov/Psched/psched-api-report.ps>
- [8] Sistare, S.; Dorenkamp, E.; Nevin, N.; Loh, E. "MPI Support in the Prism Programming Environment," *Proceedings of SuperComputing '99*, November 1999.
- [9] Sun HPC ClusterTools 5 User's Guide: <http://docs.sun.com>

Ordering Sun Documents

The SunDocsSM program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is:

`http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at:

`http://www.sun.com/blueprints/online.html`