IRIX Resource Management Plans & Status

Dan Higgins

Engineering Manager, Resource Management Team, SGI

E-mail: djh@sgi.com

CUG Minneapolis, May 1999

Abstract

This paper will detail what work has been done and the schedule for completing, testing and delivering the remaining work in the areas of IRIX Job Limits, IRIX Comprehensive System Accounting, IRIX Scheduling and Workload Management.

Introduction

There has been much effort focused on improving the Resource Management capabilities in IRIX over the last year. We are adding the concept of a "job" its associated limits, similar to the UNICOS job concept and limits, into IRIX. The functionality our UNICOS Cray System Accounting users are used to is being added to IRIX and will be called IRIX Comprehensive System Accounting (CSA). We have made improvements in IRIX job repeatability in IRIX 6.5.4 and Miser. We introduced the Load Sharing Facility from Platform Computing as our workload management solution. The need for custom scheduling policies is being addressed with our new eXtensible Resource Scheduler Product.

IRIX Job Limits

Overview

UNICOS systems have long allowed system administrators to control machine access on a per-user basis using job limits and the UNICOS User Data Base. IRIX system administrators have requested similar functionality. The IRIX Kernel Job Limits project will provide the ability to define job limits and control user processes in a manner that's similar to what's done on UNICOS.

IRIX currently supports resource limits for individual processes. The "getrlimit" and "setrlimit" library routines are the user interface to process limits. While limits on individual processes are useful, they don't provide all the tools that system administrators need to control the workload on a machine. A login session or batch submission can spawn multiple processes and a mechanism is needed that controls the resource used by all these processes as an aggregate. The UNICOS job concept is such a mechanism. A new job is created for each login session or batch submission and all processes that are created as a result belong to the job. The job becomes the container used to limit resources.

For IRIX, when we talk about the job limits work we are doing, it is best to break it the discussion into three parts; the definition of an IRIX job, Limit Domains, and the specific limits we are going to support.

IRIX Job Concept

An IRIX job is a collection of processes. In particular, it is the collection of all processes that stem from a single point of entry to a machine. As is illustrated in figure 1 below, a point of entry may be an interactive login, a submission from a workload management product like LSF, a cron job, or a remote access via rsh, rcp, ftp, or array services. All processes that are descendants of the original point of entry process are part of the job. A job may contain multiple process groups, sessions, or array sessions but processes in one of these sub-groups are always contained within a single job. Each new process in a job inherits its jobid and limits from its parent and the "job container" is not escapable by non-privileged users. This allows us to implement further IRIX features so that all processes associated with a particular job can be limited, queried, controlled, scheduled and accounted for as a group in a manner similar to UNICOS jobs.

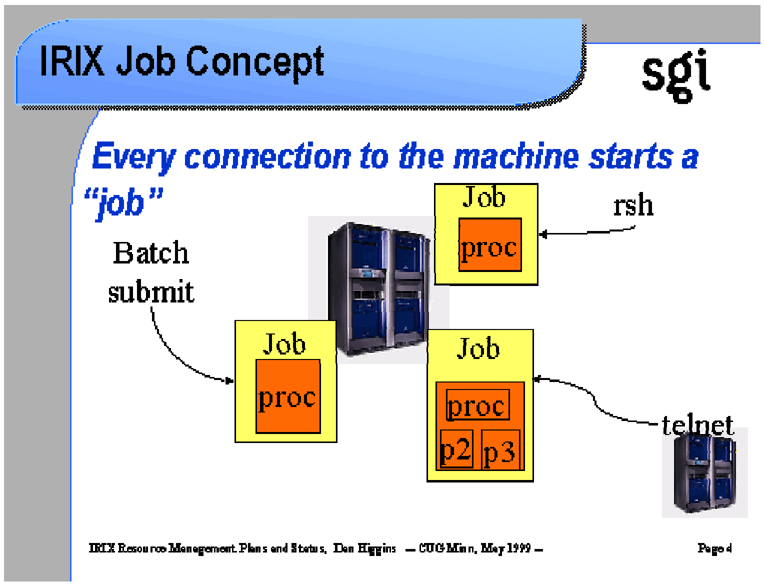
IRIX Job Concept
sgi
Every connection to the machine starts a "job"
Job
proc
rsh
Batch submit
Job
proc
Job
proc
p2
p3
telnet
IRIX Resource Management Plans and Status, Dan Higgins — CUG Minn, May 1999 —
Page 4

Figure 1 - Entry points

Limit Domains

Limit domains are a way of grouping related limits together. The UNICOS UDB provided two fixed limit domains - interactive and batch. With IRIX job limits, we have the same ability to define interactive and batch limits on a per-user basis as we could on UNICOS.

In addition on IRIX, limit domains are extendable and we have the ability for administrators or workload management products to define their own custom limits within their own limit domains. The SGI provided commands that allow interactive access (login, telnet, rlogin, rsh, cron, etc) will all access limits from the interactive domain and set these limits when a "job" is initiated.

Supported Limits for Jobs

The new job limits are modeled on IRIX process limits and extend these limits across all the processes in a job with the addition of limits to control number of processes per job and number of tapes per job.

Currently, the existing getrlimit and setrlimit library routines provide access to per-process limits.

Job limit values will be manipulated via the new setjlimit and getjlimit library routines.

There is also a jlimit command for users and administrators to display or alter a job's limits.

The ps command will be modified to display the new job ID. Job Ids are unique in a cluster, similar to array session handles.

IRIX Job Limits Status

The IRIX Kernel Job Limits project has taken pains to gather, document, and analyze all job and limit relates requirements for IRIX. The user Interface and design details are completed and documented and much of the IRIX kernel changes are complete. We will begin beta testing this feature at Boeing in September. Generally availability of the IRIX Job Limits feature will be in IRIX 6.5.7 in Q1CY00 or sooner. We are working with the engineers at Platform Computing to integrate IRIX Kernel Job Limits with their LSF workload management product.

IRIX Comprehensive System Accounting (CSA)

Overview

Currently, IRIX supports two modes of accounting, standard System V accounting, and an extended accounting feature. The current implementation of IRIX standard System V accounting provides the standard set of System V user and administrator accounting commands and kernel counters. The IRIX extended accounting feature additionally provides process and array session accounting but does less resource consumption reporting than the UNICOS Cray System Accounting feature. There are no native IRIX commands to process the raw data collected by IRIX extended accounting (other than using SAT commands to print the accounting records).

IRIX CSA is a superset of the capabilities provided by System V and extended accounting, with extensions for additional resource counters, job accounting, job summary accounting (deferred), parallel processing accounting (deferred), user access to job accounting (ja command), daemon accounting, flexible accounting periods, flexible billing units, and cluster accounting (deferred). In other words, it will provide in IRIX, the functionality that our UNICOS CSA users are accustomed to.

These three modes of accounting are independent of each other such that none, any or all three accounting modes can be enabled at the same time.

There are just too many features planned for IRIX CSA to get them all into the same release and still deliver it to customers in a timely fashion so we are using a Phased Release strategy.

Phase 1 will contain most functionality needed including:

- Job accounting
- User access to job accounting (ja command)
- Daemon accounting
- Flexible accounting periods
- Flexible System Billing Units (SBUs)
- More accurate time accounting
- Machine independent time accounting (i.e. Times externalized in microseconds)
- Additional ID fields: pid, ppid projid, jobid & ash
- More control of volume of information generated by kernel
- All accounting data written to pacct file (no nqacct or tpacct files)
- Support of cluster (pacct file will contain machine information)

Post Phase 1 features include:

- Support for specific hardware capabilities:

  Multi-tasking records

  MPP records for MPI jobs

- Incremental accounting for long running jobs
- Accounting by Array Session Handle (ASH)
- API for reading the accounting records

IRIX CSA Status

The IRIX CSA project has gathered, documented and analyzed all IRIX accounting related requirements. The high level design is complete and the detailed design is nearing completion. Much of the kernel and higher level code is also complete. The IRIX CSA feature will begin beta testing at Boeing in December and will be generally available with IRIX 6.5.8, in Q2CY00 or sooner. We are working with the engineers at Platform Computing to integrate IRIX CSA with their LSF workload management product.

IRIX Scheduling

There are three topics with regards to IRIX scheduling which will be discussed; the Share II fair share scheduler from Aurema, the Miser scheduler, and the eXtensible Resource Scheduler (XRS).

Share II Resource Manager

The Share II Fair Share Scheduler allows an organization to create its own resource allocation policy based on its own assessment of how resource usage should be fairly distributed to individuals or arbitrarily grouped users. Users or groups of users can be guaranteed a certain percentage of the machine. The scheduler makes users compete rather than processes and uses group dynamics to keep overall usage fair. Currently Share II works only on a single system and is available for IRIX 6.5.

Figure 2 shows a quick example of how a machine using Share II could be configured to be shared equally across three separate groups, Physics, Chemistry & Math, and then within each group, different users can be given varying amounts of shares.

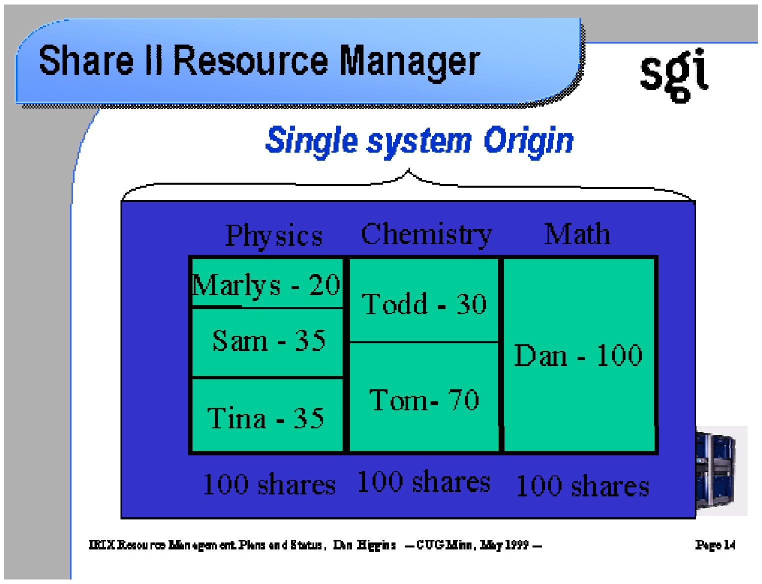

Figure 2 - Share II configuration

Miser

Overview

Miser is a user level deterministic batch scheduler that generates a non-conflicting schedule of jobs with known time and space requirements. Given a set of jobs to schedule, Miser searches through a list of resource allocation schedules to find an allocation that best fits the job using a specific scheduling policy. It was first generally available with IRIX 6.5. Its usage started out pretty light in part because it did not initially meet some of the users expectations and because its early stability was an issue.

Recent Improvements

Much effort has been put into Miser over the last 9 months to increase its reliability and to add some important new features.

- Many kernel panics have been resolved.
- Improving the way in which the kernel scheduling utilizes locks has increased the repeatability of jobs.
- A schedule repack feature was added to increase the over-all machine utilization.
- Miser_cpuset job tracking problem
- Miser_cpuset recovery mechanism
- Additional information in command output

We've also improved the Miser documentation by rewriting the Miser section in the IRIX Admin Configuration & Operations manual and by beefing up the man pages significantly.

These efforts have made Miser much more stable and usable in IRIX 6.5.4. One group that will benefit from these changes is message passing users on time-shared systems whom can now better use Miser to get the synchronized scheduling they often need. For more information about using MPI with Miser, I'd refer you to the paper titled "What's New in the Message Passing Toolkit?" in the CUG proceedings.

Miser Plans

In IRIX 6.5.5, miser_cpusets will be enhanced to be ccNUMA aware, which will allow jobs to ensure that the memory required by the job is allocated on the same node boards as the CPUs used by the job. Also, the job will have the ability to request that memory for its exclusive use. Both of these features will improve the job's repeatability.

We will likely be integrating Miser queues & miser_cpusets to allow the use of miser_cpusets for jobs submitted and scheduled through Miser. If our prototyping of this goes well this may be available in IRIX 6.5.6.

We are also working with Platform Computing to get Miser & miser_cpusets integrated into LSF 4.0, which will be available the end of this year or at the latest LSF 4.1 available in Q1CY00.

Going forward, we will continue our work to increase Miser's stability and will build on the functionality Miser provides by adding much needed new features into the next generation IRIX resource scheduler which we are calling the "eXtensible Resource Scheduler" or simply XRS.

IRIX eXtensible Resource Scheduler (XRS) Overview

The eXtensible Resource scheduler is an extension of the Miser scheduler concept. XRS is a system that allows clients to query resource availability information, lock resource states, make resource reservation requests, confirm the status of reservations, and eventually claim the reservation.

Where as Miser only considers the logical processor and memory resource requirements of a job while attempting to determine when the job will be able to run, the XRS system provides a flexible resource reservation framework which includes a strict and well defines relationship with the IRIX kernel. In addition, this framework can be extended by customers to meet their unique scheduling requirements.

The focus of the XRS system will be to manage resources, specifically CPU and memory, to help customers achieve better application performance and repeatability of results. This is accomplished by careful scheduling of the system resources so that they are not oversubscribed, or that the resources are oversubscribed in a controlled manner. In addition, the XRS system will incorporate the ability for users to specify placement requirements for their jobs. The scheduling of resources will be done based upon the placement requirements for a job.

Scheduling Domains

Using XRS, the system is partitioned into two scheduling domains - the TimeShare domain and the XRS domain. The TimeShare scheduling domain will utilize scheduling attributes as expected for timeshare UNIX systems. The XRS scheduling domain will utilize a stricter set of scheduling attributes that rely upon reservation of resources by jobs submitted into the scheduling domain through the XRS daemon. By default, all processes will belong to the TimeShare scheduling domain - this is the traditional UNIX timeshare scheduling domain. In order for a job to become a member of the XRS scheduling domain, a resource reservation request must be made with the XRS daemon. When a process claims that reservation, an XRS task will be created. All processes that are members of that task will belong to the XRS scheduling domain. The XRS system will be responsible for evaluating the resource requirements of the request, reserving resources for the request, and blocking the request from creating a task until the reserved resources are available.

Conceptually, there are two paths that allow a task to be launched in the XRS domain. For batch submission users, the user will submit a batch-job to a batch management system such as LSF, PBS, or CODINE. Assuming that the batch management system has been integrated with XRS, it can choose to execute the batch-job in either the XRS domain or the TimeShare domain. If the batch management system chooses to execute in the XRS scheduling domain, it will begin a dialog with the XRS server to communicate resource requirements, obtain a reservation of resources, and eventually claim the reservation and launch the task. Dialog with XRS will take place via a published API.

Interactive users will also be able to execute tasks in the XRS scheduling domain. To execute tasks in the XRS domain, the interactive user will need to use an XRS client command. The XRS client command will perform the dialog with the XRS server to communicate resource requirements, obtain a reservation of resources, and then block, waiting to accept the reservation and launch the task. Essentially, the XRS client will be a simple wrapper for the published API used to communicate with the XRS server. A diagram of the submission of jobs into the various scheduling domains is provided in Figure 3.

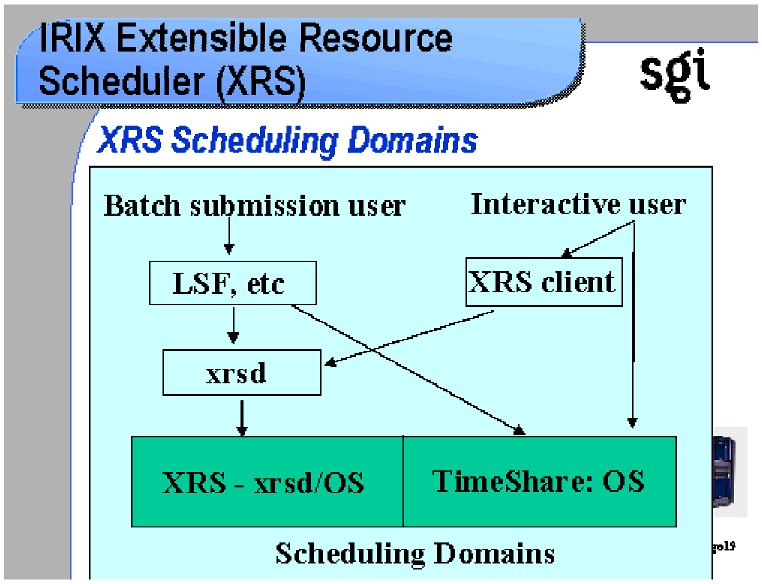

Figure 3 - Submission of jobs into scheduling domains

Scheduling Partitions

The XRS scheduling domain can be organized into various scheduling partitions. A scheduling partition is defined as a collection of resources and the scheduling policy that manages those resources.

The collection of resources that will be managed in the initial implementation will include:

- CPU

  CPU resources will be controlled by indicating the number of CPUs and their attributes. Attributes of a CPU include the speed of the processor, cache size, cache speed, the size of assumed local memory, etc.

- Memory

  Memory resources will be controlled by indicating the amount of memory required. Memory resources will be maintained as discrete items, where an item is comprised of the amount of memory resident on a node board. Memory resources will also have cross reference information that indicates the CPUs that are local to the memory resource.

- Topology

  Topology requirements will be specified using a dplace-compliant placement file that is supplied as part of a resource request. The XRS system will honor the placement specification as per dplace, but will be restricted to the scheduling partition that the request is being scheduled against.

Resources will be related based upon their locations within the hardware graph filesystem (hwgfs). Placement decisions will be made using the hwgfs relationships.

Partition Scheduling Policies

The scheduling policies that will be used to manage the partitions include:

- Predictive

  The Predictive scheduling policy provides predictable completion times. A resource request is scheduled and placed in the first time slot available that satisfies the resource requirements specified by the request.

  The start and end times for the resource reservation are fixed and will not change. If a job finishes prior to the end of the reservation period, the schedule will not be adjusted to fill in the gaps in the schedule. This policy does not provide for the pre-emption of jobs.

- Availability

  The Availability scheduling policy is an extension of the Predictive policy. A resource request is scheduled and placed in the first time slot available that satisfies the resource requirements specified by the request. The start and end times for the reservation are determined, but are subject to change. If a job finishes prior to the end of the reservation period, the schedule will be adjusted to fill the gaps that appear in the schedule. The re-ordering of the schedule will first consider the run-ordering of the requests and consider the efficient packing of the schedule as a secondary concern. This policy can run concurrently with the Preemptive policy.

- Priority

  The Priority scheduling policy is an extension of the Availability policy. All requests for resources that are managed by this policy will be assigned a default priority. Requests with the same priority will be scheduled using a first fit approach. Requests that are submitted with higher priority will be scheduled prior to lower priority requests; this will cause a schedule re-ordering for all requests of lower priority.

Priorities will be managed by an access control list (ACL) for the partition. All users will assume a default priority. Users that have been given privilege via the ACL will be allowed to specify higher priorities for their requests. This policy can run concurrently with the Preemptive policy.

- Shared

The Shared scheduling policy allows oversubscription within a partition. Non-renewable resources, such as memory, must be specified and reserved. However, renewable resources may be shared and can be oversubscribed. This policy can run concurrently with the Preemptive policy.

- Preemptive

The Preemptive scheduling policy is a supplemental scheduling policy that may run concurrently with the Availability, Priority, and Shared scheduling policies. A request cannot preempt another request that is running as the result of a preemption. The request must first be submitted to the partition as per the default scheduling policy. Privileged users can then specify that the request should preempt current request(s) in the resource schedule. The preemption will cause a re-ordering of the schedule so that the preempted requests will be re-scheduled prior to requests that followed it in the schedule prior to the preemption. When specifying active reservations that should be preempted, the privileged user can specify that the job be suspended or check-pointed. If the job is suspended, the memory resources will not be available to the preempting request. If there is not enough memory for the preempting request, the preemption will fail. The suspended or checkpointed jobs will be restarted when the new resource reservation period for the jobs is active.

In all scheduling policy cases, except Shareable, the user must specify the amount of time that they will require the requested resources. In future work it may be possible to allow large-grain time-slicing of reservation periods, when this is possible the specification of a time restriction may be eased.

IRIX XRS Status

The IRIX XRS project has gathered, documented and analyzed all IRIX scheduling related requirements. Then, a concept paper was created and reviewed. Prototyping, further research and the high level design is in progress. The IRIX XRS feature, if all progresses on schedule, will begin beta testing at Boeing in Q2CY00 and will be generally available with IRIX 6.5.9, in Q23CY00. We are working with the engineers at Platform Computing to integrate IRIX XRS with their LSF workload management product.

Workload Management

Load Sharing Facility (LSF)

SGI is partnering with Platform Computing to deliver the Load Sharing Facility (LSF) as our workload management standard. Some of the highlights of this partnership include:

- LSF 3.2 for IRIX, UNICOS & UNICOS/mk available now
- LSF will support SNx & SVx
- MPT supported with LSF Parallel available now
- NQE features in LSF 4.0 available in Q4CY99:
- File Transfer Agent (FTA)
- Improved output file handling
- UNICOS accounting support
- Job-based limits for major resources
- Integrating IRIX job limits, CSA, Miser, and XRS with LSF

See the paper titled "LSF Workload Management System Status and Plans" in the CUG proceedings for more information regarding LSF.

Network queuing environment (NQE)

The NQE product is feature complete as of the NQE 3.3 release and continues to be supported on SGI/Cray Platforms. This means that there will continue to be occasional (2-4 times per year) updates to NQE 3.3 through 2004. These updates will include fixes for critical customer problems and the occasional hardware related fix. NQE on non SGI platforms is retired.

The IRIX job limits and CSA work will be utilized by NQE in a future NQE 3.3.0.xx update to resolve these long outstanding critical inconsistency issues between the UNICOS and IRIX versions of NQE.

IRIX Resource Management Roadmap

The IRIX Resource Management roadmap is shown in figure 4 and can be summarized as follows:

- IRIX Job Limits in IRIX 6.5.7 (Q1CY00)
- IRIX CSA in IRIX 6.5.8 (Q2CY00)

  Miser much more reliable and performs better in IRIX 6.5.4
- IRIX XRS in IRIX 6.5.9 (Q3CY00)
- LSF is our workload management solution
- NQE 3.3 supported on SGI platforms through 2004

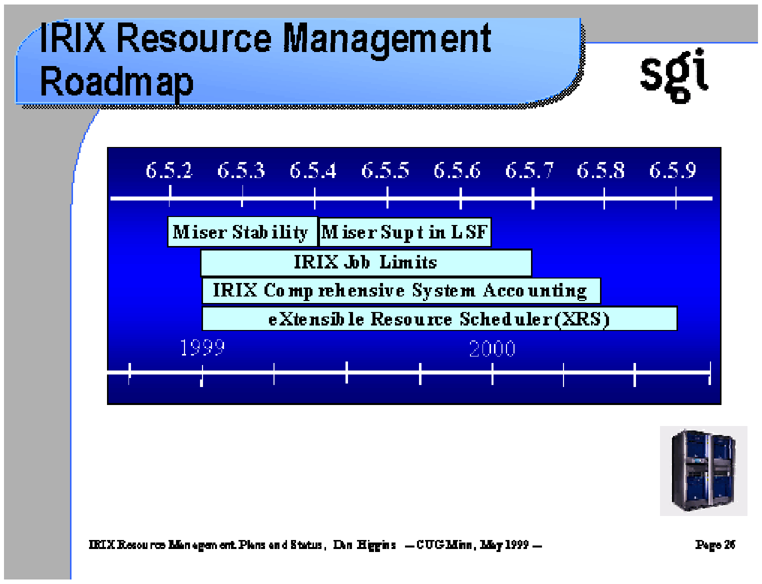

Figure 4 - IRIX Resource Management Roadmap