Working Group

DRMAA: Distributed Resource Management Application API

WG co-chairs:
John Tollefsrud, j.t@sun.com
Hrabri Rajic, Hrabri.rajic@intel.com

www.mcs.anl.gov/~jms/ggf-sched

DRMAA WG at GGF5

Working Session	**07/22/02**	**13:00-14:30**
Working Session	**07/23/02**	**15:00-16:30**
Location:	**CR Harris 1**	

DRMAA WG Working Session Agenda

- **GGF Prelude**
  - **Please sign the attendance sheet**
  - **IPR Rules of GGF**
- **Brief DRMAA Introduction**
  - **Why DRMAA?**
  - **The DRMAA Charter**
- **DRMAA WG History**
- **DRMAA Document Discussion**
- **Next Steps**

Why DRMAA?

- **Adoption of distributed computing solutions in industry is both widespread and 'early adopter'**
  - **Commercial applications by independent software vendors (ISVs)**
  - **Commercial distributed resource management (DRM) systems**
  - **Scripted command-line integration by end users**
  - **Very little direct interfacing of ISV apps to DRM systems**
- **Adoption is self-limiting to industries where gain exceeds the pain**
- **Fundamental shift in the adoption pattern requires shifting the DRM integration to the ISV**

Distributed Resource Management (DRM) Systems

- **Batch/job management systems**
- **Local Job schedulers**
- **Queuing systems**
- **Workload management systems**

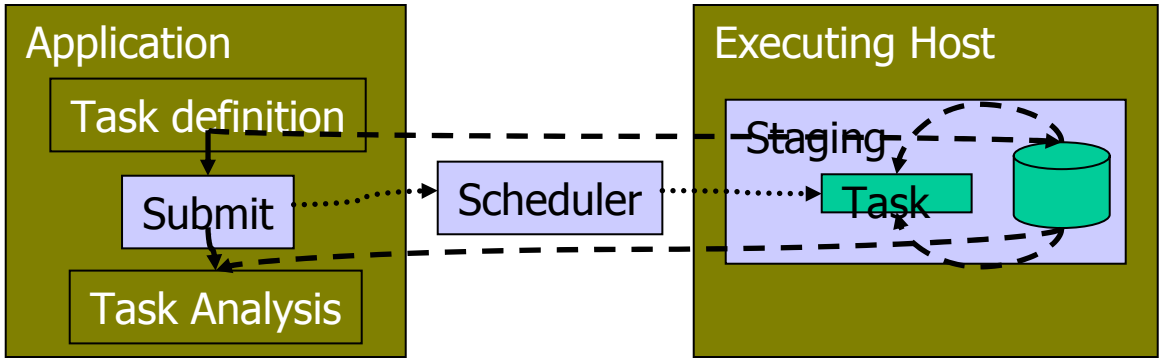
All are DRM Systems

Motivation for DRMAA

There are many DRM solutions available to end users and things keep changing

Independent Suppliers	Open Source / University	OEM Proprietary	Peer-to-Peer
Platform Computing *LSF*	Veridian *OpenPBS*	IBM *LoadLeveler*	TurboLinux *Enfuzion*
Veridian *PBS Pro*	Univ of Wisc *Condor*	Sun *Sun Grid Engine*	Entropia United Devices Parabon
Condor Inc. *Condor*	Sun *Grid Engine*		

Resource Management Systems Differ



- **Core services are fundamentally the same**
  - **especially from the users perspective**
- **DRM programming interfaces differ**
  - **ISVs are disinclined to use**

DRMAA Charter

- **Develop an API specification for the submission and control of jobs to one or more Distributed Resource Management (DRM) systems.**
- **The scope of this specification is all the high level functionality which is necessary for an application to consign a job to a DRM system including common operations on jobs like termination or suspension.**
- **The objective is to facilitate the direct interfacing of applications to today's DRM systems by application's builders, portal builders, and Independent Software Vendors (ISVs).**

Characterizing DRMAA

- **High level attributes**
  - Application centric
  - Ease of use for end users
  - Focused on programming model
- **Benefits**
  - Faster distributed application deployment
  - Opportunity for new applications
  - Increased end user confidence
  - Improvements in Resource Management Systems
  - Distributed application portability

Scope: Run a Job API

(Steps from: Ten Actions when SuperScheduling", GGF SchedWD 8.5, J.M. Schopf, July 2001)

- **Phase 1: Resource Discovery**
  - Step 1 Authorization Filtering
  - Step 2 Application requirement definition
  - Step 3 Minimal requirement filtering
- **Phase 2 System Selection**
  - Step 4 Gathering information (query)
  - Step 5 Select the system(s) to run on
- **Phase 3 Run job**
  - Step 6 (optional) Make an advance reservation
  - **Step 7 Submit job to resources**
  - Step 8 Preparation Tasks
  - **Step 9 Monitor progress (maybe go back to 4)**
  - **Step 10 Find out Job is done**
  - Step 11 Completion tasks

What have been the Issues?

- **Languages**
  - C/C++
  - Perl, Python
  - Fortran, Java
- **General features**
  - Asynchronous job monitoring
  - Protocol based
  - Scalability
- **Libraries**
  - Serial / thread safe
  - Tracing / diagnosis
- **Advanced features**
  - Debugging support
  - Data streaming
  - Security

Timetable

- **API defined by mid 2002**
  - E.g., Jul’02: DRMAA v1.0 GWD submitted for review

DRMAA Activity since GGF4

- **Bi-weekly con calls**
  - Toll Free: (877)288-4427 Code: 691169

    Next call: Tues, July 30, 9AM PDT (email to j.t@sun.com
- **Two proposals merged to a single proposal**

DRMAA Proposal 01 Overview

- **Authors: Andreas Haas, Fritz Ferstl**
- **Presenter: Andreas Haas**

Interface Form

- C-API library interface - no protocol
  - Simplifies utilization by ISV's
- Shared library binding
  - Prerequisite to allow end user to select DRM technology of their choice
- Library supports only one DRM system per implementation
  - Simultaneous support of different DRM systems is beyond the scope of our project

Job Template

- Functions to create/delete job template
  - job_template *newJobTemplate(void)
  - void deleteJobTemplate(job_template *jt)
- Job template can be used multiple times for submitting jobs
- DRMAA WG will agree on a limited ***standard*** set of job attributes
- DRMAA interface to provide a hook for other, e.g. DRM-specific job attributes

Job Template, cont'd

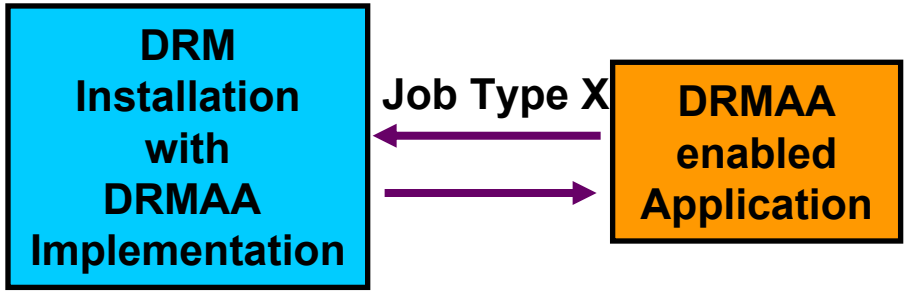
- Access functions allow setting of all job attributes that have a meaning for the DRM system
  - For DRMAA agreed attributes (real time limit, job path, job arguments, ..) via well-defined access functions, e.g.
    - int setRealTimeLimit(job_template *jt, long rtl)
    - int setJobPath(job_template *jt, char *path)
  - For non-agreed DRM specific (project, account, ...) or application specific (database affinity, ...) attributes via generic function
    - int setAttrib(job_template *jt, char *name, char *value)

Job Template, cont'd

- Job template attributes need translation into DRM system job attributes in a way specific to site/appl needs:
  - At site A jobs of type X can only be run on a subset of hosts, while at site B jobs of type X can be run on all hosts
  - At site A jobs of type Y compete with Z jobs and thus are dispatched with a high priority, while at site B jobs of type Y are less important
- Each implementation of API library is responsible for that mapping

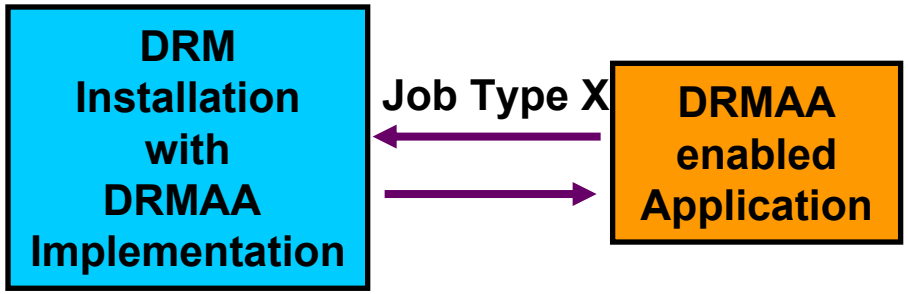
Job Categories

Site A



- Cluster consists of machines where X jobs run and others where they don't run

Site B



- X jobs run at all machines in cluster

Job Submission

- Jobs submitted to the DRM system are identified via a job identifier
- For flexibility reasons a job identifier should be of type char *
- Single job identifiers are returned by
  - int runJob(job_template *jt, char *job_id)
- NEW Bulk job submissions return multiple job identifiers
  - int runArrayJob(job_template *jt, int n, char *jid[])

Job Monitoring, Ctrl & Termination

NEW

- Monitoring functions
  - char *monitorJob(char *jobid);
- Control functions
  - int suspendJob(char *jobid);
  - int resumeJob(char *jobid);
  - int terminateJob(char *jobid);
- Blocking and non-blocking waiting for one or more jobs to finish (like wait4(2))
  - char *waitJob(char *jobid, int *status, int options, char **rusage);

DRMAA Proposal 02 Overview

- **Author and Presenter: Hrabri Rajic**

DRMAA Guidelines

- The API calling sequences should be simple and the API set small.
- The routine names should convey the semantic of the routine.
- The set should be as convenient as possible, even with the risk of being forced to emulate some functionality if missing from a DRMS.
- All jobs manipulation per process is available without explicit job iterating.
- The servers names are hidden, the DRMS is a black box.
- Consistent API structure
  - Err return parameter
- Data structures not exposed

DRMAA Library features

- Shared
- One or more DRMSs targeted
- Versioning
  - Backwards compatible
  - Stub API resolution for forward compatibility

API groups

- Init/exit
- Job submit
  - Individual jobs
    - One time
    - Multiple times -- templates
  - Bulk jobs
- Job monitoring and control
- Auxiliary or system routines
  - trace file specification
  - error message routines

Execution environment

- File system duality
  - Shared
    - Appls. comes to data
  - Distributed
    - Data come to appls.
    - Unique dir per job
      - Copying files?

- Environment passing
  - Use default behavior
  - Export local env.
  - Use remote env.
  - Env specified via API
- Handling of DRMS job execution options
  - Translation/consolidation
  - Allowing native

Native DRMS Options

- The end user interacts with the DRMS via native_resource_options parameter.
  - Simple solution
  - DRMAA implementation ignores the DRMAA DRMS implicitly used and disallowed options
  - Dist. Appls. Developers and DRMS vendors are not involved in the local environment spec.
  - The burden is on the end users to define the execution environment
    - —Need to know DRMS
    - —Need to know the remote application installation

DRMAA WG Agenda -Working Session-

Working Session 02/18/02 15:00-16:30

- **Detailed Review of proposals and proposed enhancements**
  - **Andreas Haas**
  - **Hrabri Rajic**
- **Review of comments received**
- **Topics - Considerations**
  - **API Guidelines**
  - **Resource Options**
  - **Job Categories/Classes/Templates**
  - **Environment Handling**
  - **User/Installer/App Developer experience**
  - **Error Handling / System diagnosis**
  - **Performance Measurement**
  - **Language Binding**

DRMAA WG Agenda -Presentation Session-

- **Why DRMAA?**
- **The DRMAA Charter**
- **DRMAA WG Activity since GGF3**
- **Proposal overviews**
  - Andreas Haas
  - Hrabri Rajic
- **Working session Agenda for GGF4**
- **Q & A, comments**
- **Break**
- **Resume Working Session at 15:00**