



DRMAA: Distributed Resource Management Application API

Hrabri Rajic, Intel

Andreas Haas, Sun Microsystems

Ruben Montero for Ignacio Llorente, DACYA, UCM

GGF 13 Tutorial

Seoul March 13, 2005

Agenda

- Hrabri: DRMAA Introduction
- Hrabri: Implementations
- Andreas: C and Java bindings
- Break: Coffee and cookies ☺
- Andreas: C demo
- Ruben: DRMAA for GridWay
- Ruben: demo

DRMAA-WG history

- BOF at GGF 3 in Frascati, Oct 2001
- WG status at GGF 4, Toronto, February 2002
- Participation from PBS, SGE, Intel, LoadLeveler, Condor, Cadence, Globus GRAM, University of Potsdam, DACYA
- Sideline engagement from EnFuzion, Entropia, LSF, GridIron, UD

03 Jul: Close public comment Jun

04 1Q: 2 Reference implementation prototypes:

C implementations UofW Condor, Sun's SGE

CPAN Perl DRMAA-C module

Sun's SGE Java

Feedback from reference implementations fed back into spec.

04 Jun: DRMAA recommendation document accepted by GFSC

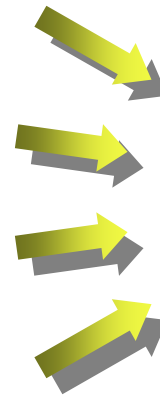
04 3Q: GridWay OGSA/Globus

In a Nutshell

- **DRMAA scope and purpose:**
 - Submit, control & monitor, and query status of jobs.
 - DRMAA library could be implemented on top of DRM systems and also OGSA.
- **Weekly con calls**
 - Toll Free: (866)545-5198 Code: 6898552
 - Regular: (865)521-8904
- E-mail: drmaa-wg@gridforum.org
- **Archive:** http://www-unix.gridforum.org/mail_archive/drmaa-wg/threads.html

Distributed Resource Management (DRM) Systems

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



All are DRM Systems

DRMAA as a Third Type of Parallelism

- Threads on SMP machine
 - OpenMP
 - Win Threads, PThreads
- MPI
 - DVSM
 - MPI + OpenMP
- Grid and compute center type computing
 - Grid web services (OGSI/OGSA)
 - DRMAA (compiled, interpreted languages)

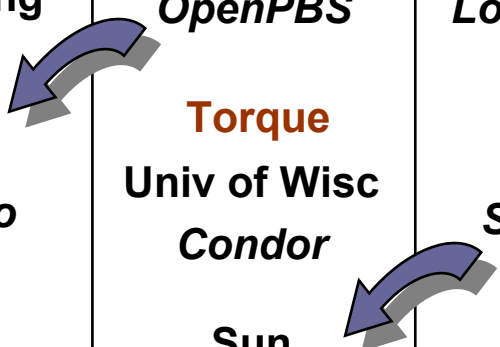
Why DRMAA?

- **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**

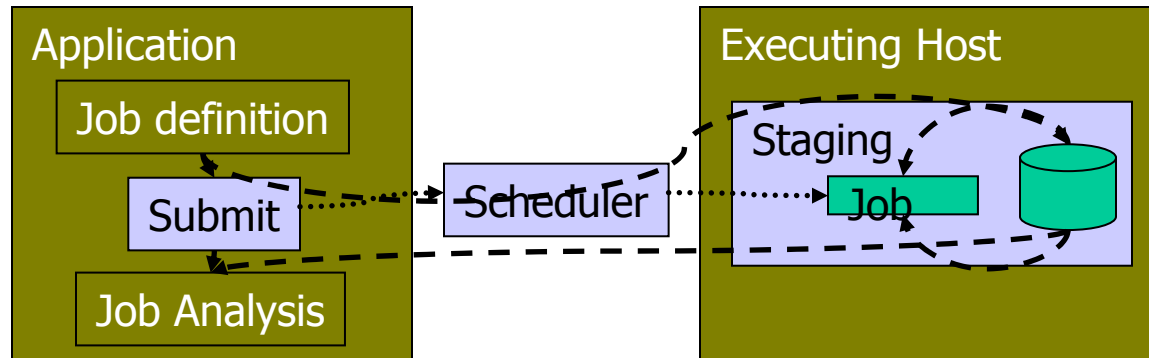
Motivation for DRMAA

There are many DRM solutions available to end users and things keep changing (2005 state of affairs)

Independent Suppliers	Open Source / University	OEM Proprietary	Peer-to-Peer
Platform Computing <i>LSF</i>	Veridian <i>OpenPBS</i>	IBM <i>LoadLeveler</i>	<i>EnFuzion</i>
Altair <i>PBS Pro</i>	Torque Univ of Wisc <i>Condor</i>	Sun <i>Sun Grid Engine</i>	Entropia
DataSynapse	Sun <i>Grid Engine</i>		United Devices Parabon <i>Grid Iron</i>



Resource Management Systems Differ Across Each Component



	Interface Format	Execution Environment	Platform Mix
LSF	Has API plus Batch Utilities via "LSF Scripts"	User: Local disk exported System: Remote initialized (option)	Unix \leftrightarrow Windows
Grid Engine	GDI API Interface plus Command line interface	System: Remote initialized, with SGE local variables exported	Unix only
PBS	API (script option) Batch Utilities via "PBS Scripts"	System: Remote initialized, with PBS local variables exported	Unix \leftrightarrow Windows
DataSynapse	Proprietary API.	User: Remote initialized	Unix \leftrightarrow Windows

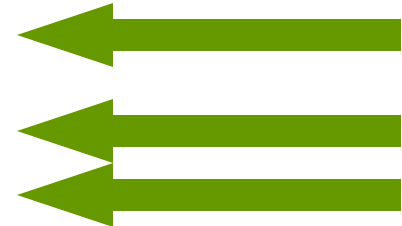
DRMAA Charter

- **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).**

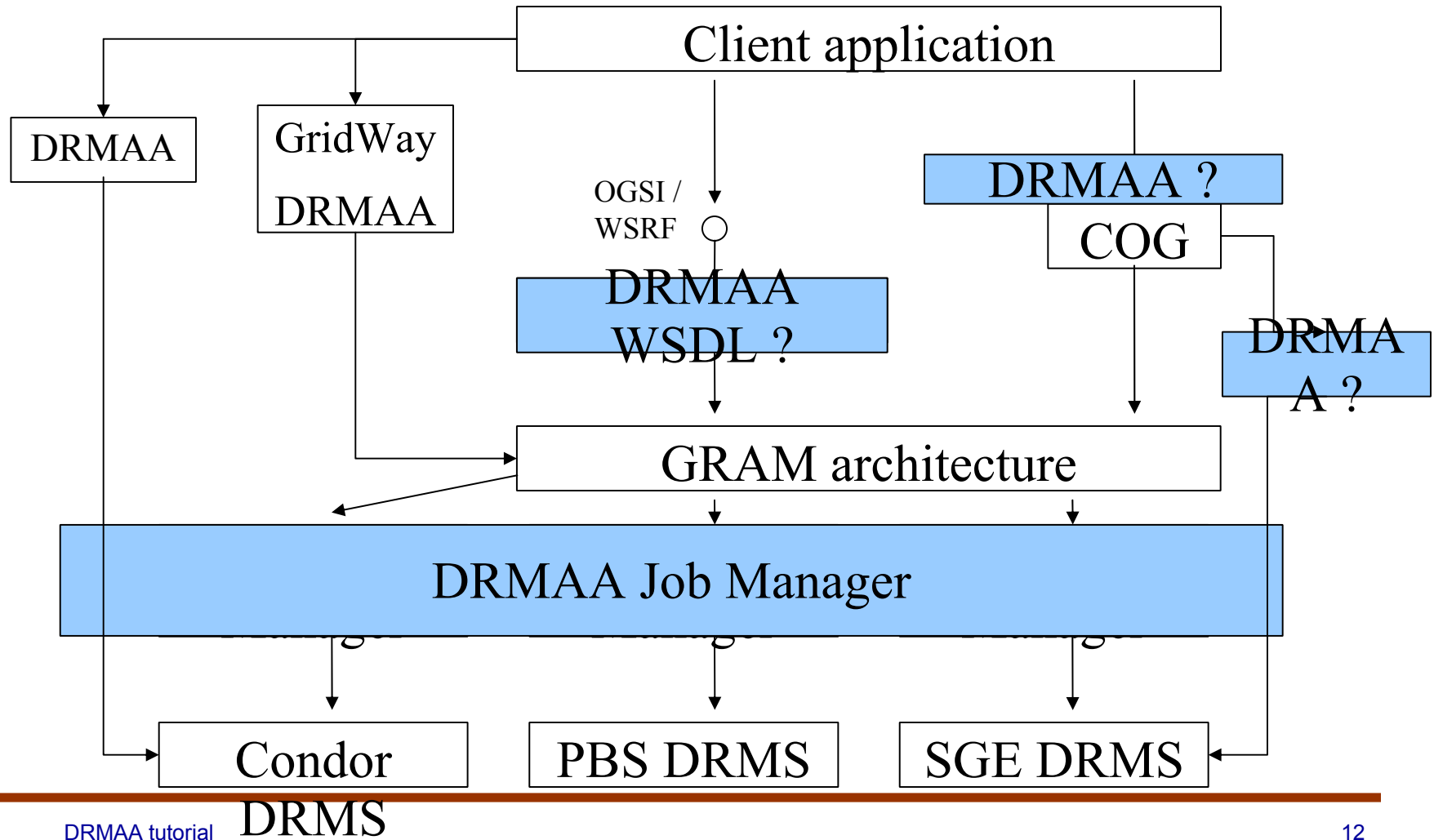
Grid Scope: Run a Job API

(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



DRMAA Placement



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

What have been the Issues?

- **Language bindings**
 - C/C++
 - Perl, Python
 - Fortran, Java
- **General features**
 - DRMAA sessions
 - Asynchronous job monitoring
 - Protocol based
 - Scalability
 - Wide characters
- **Libraries**
 - Serial / thread safe
 - Tracing / diagnosis
- **Advanced features**
 - Debugging support
 - Data streaming
 - Security
 - Categories

Implementation characteristics

- 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 DRMAA
 - Not precluded

API groups

- Init/exit
- Job template interfaces (equivlent of JSDL)
 - Allocate/delete
 - Setter/getter job template routines
- Job submit
 - Individual jobs
 - One time
 - Multiple times – templates (DRMAA v2)
 - Bulk jobs, implicit parameterization
- Job monitoring and control
- Auxiliary or system routines
 - error message routines
 - informational interfaces

Job Template

- **Functions to create/delete job template**
 - `job_template drmaa_allocate_job_template (void)`
 - `void drmaa_delete_job_template (job_template jt)`
- **Setter/getter job template routines**
 - `int drmaa_set_attribute(job_template jt, string name, string value);`
 - `int drmaa_set_vector_attribute(job_template jt, string name, string array values);`
 - `string drmaa_get_attribute(job_template jt, string name);`
 - `string array drmaa_get_vector_attribute(job_template jt, string name);`

Job Submission

- Jobs submitted to the DRM system are identified via a job identifier
- For flexibility reasons a job identifier should be a string
- Single job identifiers are returned by
 - `int drmaa_run_job(string job_id, job_template jt)`
- Bulk job submissions return multiple job identifiers
 - `int drmaa_run_bulk_job(string array job_ids, job_template jt, int start, int end, int incr)`

Job Monitoring, Control, and Status

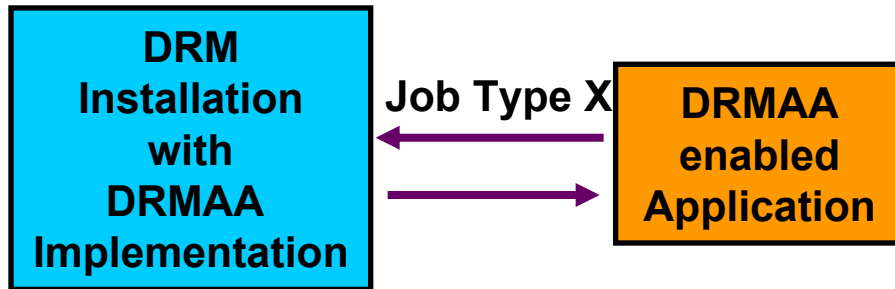
- Monitoring/Control functions
 - `int drmaa_control(string job_id, int action);`
 - `int drmaa_synchronize(string array job_ids, signed long timeout, boolean dispose);`
 - `int drmaa_job_ps(string job_id, int remote_ps);`
- Blocking and non-blocking waiting for one or more jobs to finish (like `wait4(2)`)
 - `string drmaa_wait(string jobid, int status, int timeout, string rusage);`
 - Use Posix functions `drmaa_wifexited`, etc. to get more information about failed jobs.

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 DRM
 - Need to know the remote application installation

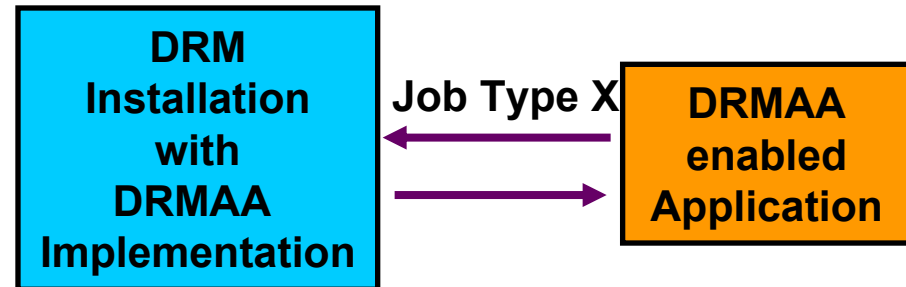
Job Categories

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

DRMAA Documents Status

- DRMAA GFD-P-R or GFD.22 document
 - Since June 2004
- C binding v1.0
 - Ready for submission to GFCS
- C binding experimental v0.98
 - Has improvements that will not be rolled into DRMAA FGD-P-R doc
- Java binding 0.6.1
 - Fairly complete
- .NET binding v0.2
 - Needs a synch with OO v0.3.1 IDL
- IDL recommendation document
 - Targeting OO languages
 - v0.3.1 and feature complete
 - Need to augment with the DRMAA GFD-P-R text
 - Will be submitted as a standalone GFD-P-R doc

Milestones

- GGF 13
 - finish C-Bindings DRMAA document
- GGF 14
 - Java-Bindings DRMAA doc
 - .NET-Bindings DRMAA doc
 - IDL binding DRMAA doc
- GGF 15
 - prepare GFD-E docs
- H1 2006
 - Submit GFD-R DRMAA doc & necessary support materials.

DRMAA Implementations

- C implementations
 - UofW Condor 6.7.1 for Linux and Windows
 - Sun's SGE
- CPAN Perl DRMAA-C module
 - Works only for SGE DRMAA C implementation
- Sun's SGE Java implementation
 - 6.0u3 SGE implements C binding v0.5
- GridWay OGSA/Globus implementation
 - Subset already released, Jan 2005
- Anouncements
 - Rocks Cluster Distribution
<http://www.rocksclusters.org/Rocks/>

Availability Matrix (18.9.2004)

	C library	Java library	Perl wrapper
Sun Grid Engine 6.0u3	Complete ☺	Complete ☺	Works
Condor 6.7.1	Partial implementation	Not available	Not working, but needs only SWIG adoption

→start with *SGE* now, things will become better during this year ☺ ...

Condor 6.7.1

- Missing DRMAA functionality
 - Job categories
 - Some mandatory job template attributes
 - Drmaa_run_bulk_jobs()
 - Drmaa_wtermsig()
 - Drmaa_wcoredump()
 - DRMAA_IDS_SESSION_ALL
 - DRMAA_IDS_SESSION_ANY
- A few bugs (we will see them)
- Condor team is hardly working on these issues, should be fixed with the next Condor releases

Sun Grid Engine 6.0u3

- `$SGE_ROOT/lib/libdrmaa.so`
- `$SGE_ROOT/include/drmaa.h`
- `$SGE_ROOT/examples/drmaa/`
- C and Java HowTo documents
- Integrated DRMAA test suite
- No open bugs for the moment

How to Get Involved

- Let your application use DRMS resources through DRMAA interfaces
- Send bug reports to SGE and Condor authors
- Write a DRMAA library for your favorite DRMS (PBS, LLF, XGrid, fork, ...)
- Design the ultimate DRMAA GUI application
- Please, let us know about your experiences

Our Tasks

- Continuous inspection of available DRMAA libraries (subscribe to drmaa-wg@gridforum.org for latest news)
- More bindings (WSDL, WSRF, Python, ...)
- More implementations (Mono, XGrid, Rocks, Globus Toolkit 4, ...)
- Collection of DRMAA 1.0 issues as tracker items in GridForge

Thank you for your attention!

drmaa-wg@gridforum.org
