DRMAA v2 - The Next Generation

Peter Tröger
Hasso-Plattner-Institute, University of Potsdam
peter@troeger.eu

DRMAA-WG Co-Chair

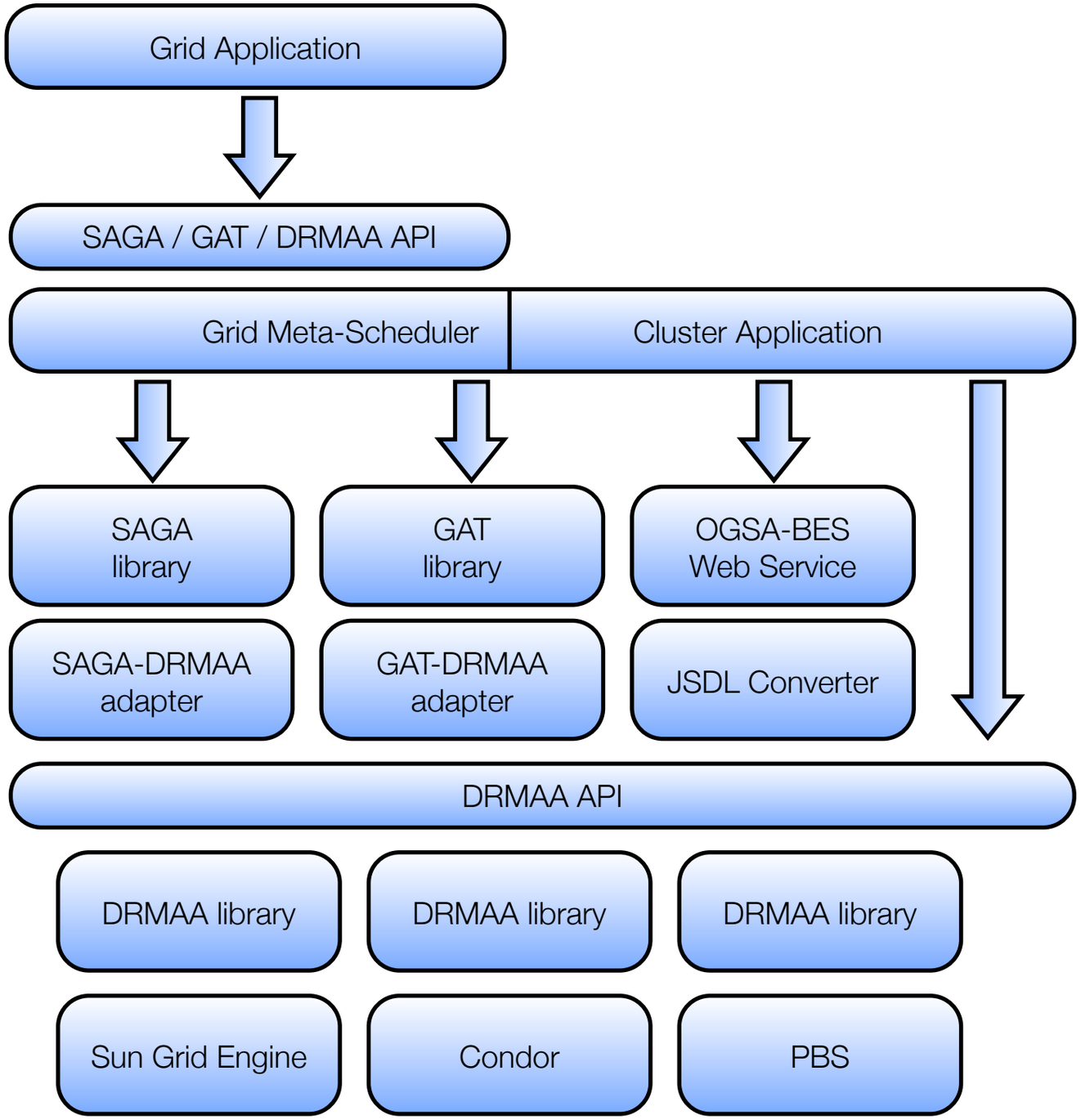
http://www.drmaa.org/

http://wikis.sun.com/display/DRMAAv2/

Past

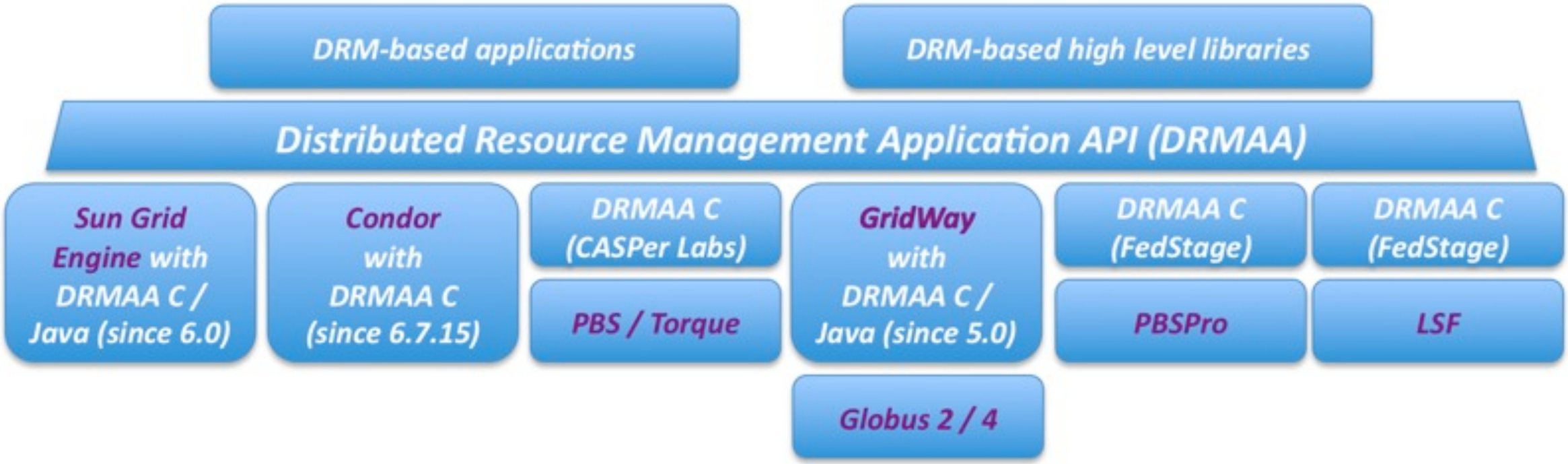
- Open Grid Forum (OGF), former Global Grid Forum (GGF)
- DRMAA group established in 2002
- Goal: Standardized API for distributed resource management systems (DRMS)
  - Low-level portability for different cluster / grid systems
  - Simple design, realization as local library on submission host
  - Leave room for areas of disagrement
- Different DRMAA 1.0 API documents
  - GFD.22 (2004), GFD.133 (2008), language bindings for C, Java, Python
  - Experience reports, tutorials, unofficial language bindings for Perl, Ruby and C#
- Related activities in OGF: SAGA, JSDL, OGSA-BES

The Stack - In Theory



Current Status with DRMAA v1

DRMAA
Distributed Resource Management Application API — www.drmaa.org



- Product-quality implementations, major deployment with some systems
- Some individual projects (support for Perl, Python, Ruby, XGrid, ...)
- Vibrant user community with SGE and GridWay
  - Applications: SGE customers, OpenDSP, Mathematica, SAGA, ...

DRMAA v1 C Example

Distributed Resource Management Application API — www.drmaa.org

```
#include "drmaa.h"

int main(int argc, char **argv) {
     char error[DRMAA_ERROR_STRING_BUFFER];
     int errnum = 0;
     drmaa_job_template_t *jt = NULL;

     errnum = drmaa_init(NULL, error, DRMAA_ERROR_STRING_BUFFER);
     if (errnum != DRMAA_ERRNO_SUCCESS) return 1;
     errnum = drmaa_allocate_job_template(&jt, error, DRMAA_ERROR_STRING_BUFFER);
     if (errnum != DRMAA_ERRNO_SUCCESS) return 1;
     drmaa_set_attribute(jt, DRMAA_REMOTE_COMMAND, "sleeper.sh",
                         error, DRMAA_ERROR_STRING_BUFFER);
     const char *args[2] = {"5", NULL};
     drmaa_set_vector_attribute(jt, DRMAA_V_ARGV, args, error,
                            DRMAA_ERROR_STRING_BUFFER);
     char jobid[DRMAA_JOBNAME_BUFFER];
     errnum = drmaa_run_job(jobid, DRMAA_JOBNAME_BUFFER, jt, error,
                    DRMAA_ERROR_STRING_BUFFER);
     if (errnum != DRMAA_ERRNO_SUCCESS) return 1;
     errnum = drmaa_delete_job_template(jt, error, DRMAA_ERROR_STRING_BUFFER);
     errnum = drmaa_exit(error, DRMAA_ERROR_STRING_BUFFER);
     return 0;
}
```

DRMAA v1 Java Example (1/2)

DRMAA
Distributed Resource Management Application API — www.drmaa.org

```
try {
      session.init ("");

      System.out.println ("Version: " + session.getDrmaaImplementation ());

      JobTemplate jt = session.createJobTemplate ();
      jt.setRemoteCommand ("<SGE_ROOT>/examples/javaone/sleeper.sh");
      jt.setArgs (new String[]{"30"});
      jt.setWorkingDirectory ("<SGE_ROOT>/<SGE_CELL>/javaone");
      jt.setJobCategory ("sleeper");

      String jobId = session.runJob (jt);
      List jobIds = session.runBulkJobs (jt, 1, 4, 1);

      System.out.println ("Job " + jobId + " is running");

      for (Object id: List jobIds) {
         System.out.println ("Job " + id + " is running");
      }

      session.deleteJobTemplate (jt);
```

DRMAA v1 Java Example (2/2)

```
      JobInfo info = session.wait (jobId, Session.TIMEOUT_WAIT_FOREVER);
      Map usage = info.getResourceUsage ();

      for (Object name : usage.keySet ()) {
         System.out.println (name + "=" + usage.get (name));}

      if (info.hasExited ()) {
         System.out.println ("Job exited: " + info.getExitStatus ());}
      else if (info.hasSignaled ()) {
         System.out.println ("Job signaled: " + info.getTerminatingSignal ());

         if (info.hasCoreDump ()) {
            System.out.println ("A core dump is available.");}}
      else if (info.wasAborted ()) {
         System.out.println ("Job never ran.");}
      else {
         System.out.println ("Exit status is unknown.");}

      session.synchronize (jobIds, Session.TIMEOUT_WAIT_FOREVER, true);
   }
   catch (DrmaaException e) {
      e.printStackTrace ();
      System.exit (1);}
}
```

DRMAA v1 has Issues

DRMAA
Distributed Resource Management Application API — www.drmaa.org

- Desperately missing features
  - Concept of resources, session persistency, advance reservation, parallel jobs, queue support, ...
- Some obsolete / never implemented features
  - Date / time handling, host-to-host file staging, specialized job states, ...
- Awkward design decisions
  - Job synchronization, job monitoring, data reaping, ...
- C-centric API design - First solution with GFD.130
- DRMAA v2 work started 2009, finalization **happens now !**
  - Public survey, Sun customer feedback, implementation experiences, ...
  - Close collaboration with SAGA / GAT people and other implementors

Spec Design Approach

DRMAA
Distributed Resource Management Application API — www.drmaa.org

- All behavioral aspects in the IDL-based root specification
  - API feature set, functional behavior, error conditions, multithreading issues
- Language binding provides syntactical mapping only
- Interfaces are mapped to classes (OO languages) or can be flattened (C language)

2. Python Language Mapping for DRMAA

A Python module implementation can declare "DRMAA 1.0-compliance" if it realizes the API signature described in the following sections, and the functional behavior as described in [GFD130]. Additional module functionality beside the specified API is allowed, but must be clearly identifiable (e.g. by a function name convention).
The following table provides the basic mapping overview for the DRMAA IDL constructs to the Python programming language:

DRMAA 1.0 IDL specification	*DRMAA 1.0 Python binding*
`module` definition	Python module file named "drmaa.py"
`interface` definition	`class` definition
`enum` definition with enumeration members	`class` definition
`string` type	`str`
`long` type	`int`
`long long` type	`long`
`const` definition	Pre-defined `class` attributes
`boolean` type	`bool`

```
"""This is drmaa.py, implementing the DRMAA Python language binding
   Visit www.drmaa.org for details"""

# Job control action
class JobControlAction:
    SUSPEND='suspend'
    RESUME='resume'
    HOLD='hold'
    RELEASE='release'
    TERMINATE='terminate'

# State of single job
class JobState:
    UNDETERMINED='undetermined'
    QUEUED_ACTIVE='queued_active'
    SYSTEM_ON_HOLD='system_on_hold'
    USER_ON_HOLD='user_on_hold'
    USER_SYSTEM_ON_HOLD='user_system_on_hold'
    RUNNING='running'
    SYSTEM_SUSPENDED='system_suspended'
    USER_SUSPENDED='user_suspended'
    USER_SYSTEM_SUSPENDED='user_system_suspended'
    DONE='done'
    FAILED='failed'

# State at submission time
```

DRMAA v2 Layout

```
module DRMAA{

   interface Session

   interface JobTemplate

...

}
```

```
module DRMAA2{

   interface SessionManager

   interface JobSession

   interface JobTemplate

   interface Job

   interface ReservationSession

   interface ReservationTemplate

   interface Reservation

   interface MonitoringSession

   ...

   }
```

DRMAA v2 Session Manager

- Create multiple connections to one (or more) DRM system(s) at a time
- 3 different session types
  - `JobSession / ReservationSession`
    - Persistent storage for a set of submitted and controlled jobs / reservations (on the same machine, for the same user; string identifiers)
    - Explicit reaping on session level only
  - `MonitoringSession`
    - Read-only semantic, global view on all machines
- Concept should map nicely to high-level API‘s
- Security remains out of scope

DRMAA v2 Session Manager

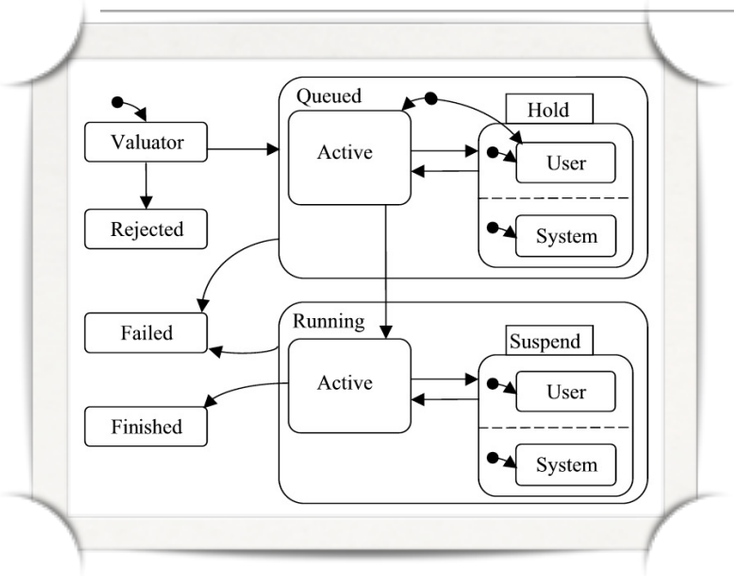
```
interface SessionManager{
  readonly attribute string drmsInfo;
  readonly attribute Version version;
  ReservationSession createReservationSession
     (in string sessionName, in string contactString)
  ReservationSession openReservationSession(in string sessionName)
  void closeReservationSession(in ReservationSession s)
  void destroyReservationSession(in string sessionName)
  StringList getReservationSessions()

  JobSession createJobSession(in string sessionName, in string contactString)
  JobSession openJobSession(in string sessionName)
  void closeJobSession(in JobSession s)
  void destroyJobSession(in string sessionName)
  StringList getJobSessions()

  MonitoringSession createMonitoringSession (in string contactString)
  void closeMonitoringSession(in MonitoringSession s)
};
```

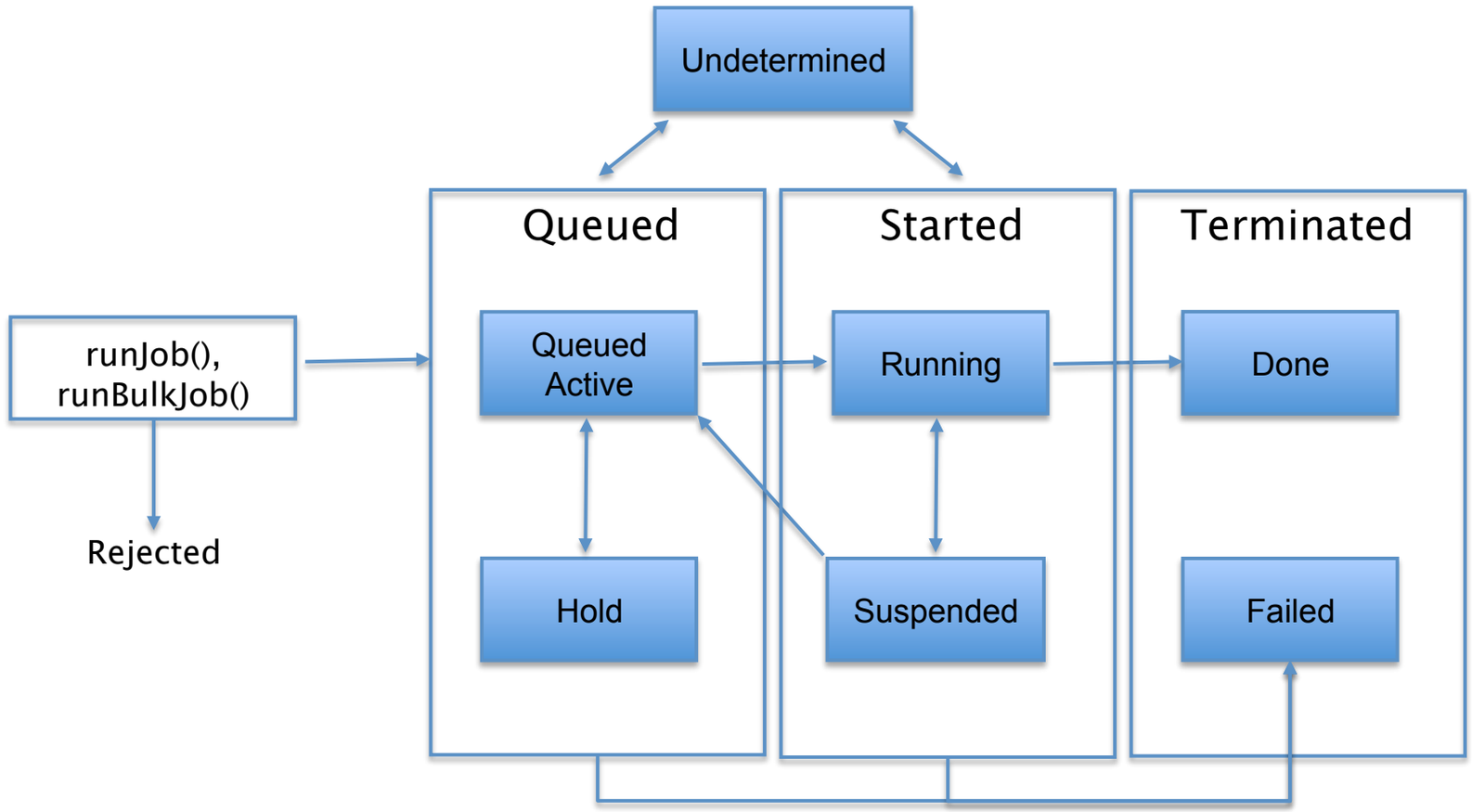
DRMAA v2 Job Session

```
interface JobSession {
 readonly attribute string contact;
 readonly attribute string sessionName;
 Job getJob(string jobId);
 sequence<Job> getJobs(JobInfo filter);
 JobTemplate createJobTemplate()
 Job runJob(in DRMAA::JobTemplate jobTemplate)
 sequence<Job> runBulkJobs( in DRMAA::JobTemplate jobTemplate, ...
 Job waitAnyStarted(in sequence<Job> jobs, in timeout)
 Job waitAnyTerminated(in sequence<Job> jobs, in timeout)
 void registerEventNotification(in DrmaaCallback callback)
```

- Optional support for event push notification
- New support for filtered list of jobs
- *waitAnyStarted():* Wait for one of the „start states“ to happen
- *waitAnyTerminated():* Wait for one of the „terminated states“ to happen

DRMAA v2 Job States





- Less states, DRMAA v1 semantics mappable by new sub-state support
- Expected SAGA / OGSA-BES mapping is part of the spec
- Waiting for single states would bring nasty timing issues

DRMAA v2 Event Callback

```
interface DrmaaCallback {
  void notify(in DrmaaNotification notification)

enum DrmaaEvent {
       NEW_STATE_UNDETERMINED, NEW_STATE_QUEUED_ACTIVE,
       NEW_STATE_HOLD, NEW_STATE_RUNNING,
       NEW_STATE_SUSPENDED, NEW_STATE_DONE,
       NEW_STATE_FAILED, MIGRATED, ATTRIBUTE_CHANGE
   };
```

- Optional support for event push notification
  - At least supported in SGE, large demand from end users and SAGA
  - Library has the freedom to implement this by state polling
- There might be more interesting events to get from the library

DRMAA v2 Job Template

- Most things remain the same, but:
  - Relative start / end times are gone, switched to RFC822
  - Completely reworked file staging support
    - Reduced version of LSF / SAGA syntax (minimal wildcards, only copy)
      - Only between submission and exception host
  - Standardized job ~~category~~ configuration names (based on GFD.115)
- Additional attributes for resource requirements
  - Input from JSDL, HPC Basic Profile, SAGA, real systems, **...**
  - **Why is this so hard ?**

Example 1: Semantic matching

Distributed Resource Management Application API — www.drmaa.org

A	B	C	D	E	F	G	H
	JSDL		**SGE**				
	JSDL Name		**SGE Queue Properties**	**SGE Description**		**Condor Machine ClassAd**	**Condor submission file**
Resource requirement for job							
	CandidateHosts					MACHINE	
	TotalResourceCount		slots	Number of processes (allowed) to run			machine_count
	FileSystem					-- (no free choice of FS)	
	ExclusiveExecution		(This is accomplished through a special complex as of 6.2u3.)				
	OperatingSystem		(arch built-in complex)			OPSYS	
	CPUArchitecture		(arch built-in complex)			ARCH	
	IndividualCPUSpeed					KFLOPS	
	IndividualCPUTime						
	IndividualCPUCount		(num_proc built-in complex)			CPUS	
	IndividualNetworkBandwidth						
	IndividualPhysicalMemory		(mem_total built-in complex)			MEMORY	
	IndividualVirtualMemory		(virtual_total built-in complex)			VirtualMemory	
	IndividualDiskSpace					DISK	
	TotalCPUTime		s_cpu / h_cpu	Soft / hard limit for CPU time of all processes			
	TotalCPUCount						
	TotalPhysicalMemory						
	TotalVirtualMemory		s_vmem / h_vmem	Soft / hard limit for job virtual memory			
	TotalDiskSpace		s_fsize / h_fsize	Soft / hard limit for bytes on disk			
				Minimum time between			

Example 2: Common feature sets

DRMAA
Distributed Resource Management Application API — www.drmaa.org

	LSF	Torque	PBS Pro
Wildcard Support	no	not recommended	yes
Other than submission host	no	yes	yes
Appending file	yes	no	no
Directory Staging	no	not by default	yes

- Ensuring true application portability with a unified API is REALLY hard
  - Interoperability (OGSA, JSDL) does not provide portability
  - Profiles with „SHOULD“ and „UnsupportedFeatureFault“ are not helpful
- DRMAA tries to define **mandatory** job template attributes and API functions that are **implementable** in **most** DRM systems
  - This is why DRMAA will never by exhaustive -> use GAT / SAGA !

Agreed Resource Concepts

DRMAA
Distributed Resource Management Application API — www.drmaa.org

- DRM systems contain machines, which have a CPU and physical memory
- Machines can be booked in advance reservation
- DRM systems contain queues, but they are an opaque concept for DRMAA
- Jobs can be submitted ...
  - ... to specified candidate machines, or a queue, or both (?)
  - ... to machines matching an OS type, architecture type, or memory requirement
  - ... as ‚classified‘ job with special treatment by the DRMS
    - configurationName attribute: Central list of recommended strings, realized by configuration on each particular installation
    - Examples: MPICH2 job, OpenMPI job, OpenMP job

DRMAA v2 JobTemplate (March 24th)

```
interface JobTemplate {
  const string HOME_DIRECTORY = "$drmaa_hd_ph$";
  const string WORKING_DIRECTORY = "$drmaa_wd_ph$";
  const string PARAMETRIC_INDEX = "$drmaa_incr_ph$";
  const string BULK_TASK_ID_VARNAME = "$drmaa_taskid_varname$";
  attribute string remoteCommand;
  attribute OrderedStringList args;
  attribute DRMAA::JobSubmissionState jobSubmissionState;
  attribute Dictionary jobEnvironment;
  attribute string workingDirectory;
  attribute string configurationName;
  attribute string accountingId;
  attribute string nativeOptions;
  attribute StringList email;
  attribute boolean blockEmail;
  attribute AbsoluteTime startTime;
  attribute string jobName;
  attribute string inputPath;
  attribute string outputPath;
  attribute string errorPath;
  attribute boolean joinFiles;
```

DRMAA v2 JobTemplate (March 24th)

DRMAA
Distributed Resource Management Application API — www.drmaa.org

```
attribute OrderedStringList stageInFiles;
attribute OrderedStringList stageOutFiles;
attribute AbsoluteTime deadlineTime;
attribute TimeAmount hardWallclockTimeLimit;
attribute TimeAmount softWallClockTimeLimit;
attribute TimeAmount hardRunDurationLimit;
attribute TimeAmount softRunDurationLimit;
attribute string queueName;
attribute long minSlots;
attribute long maxSlots;
attribute long minPhysMemory;
attribute OperatingSystem machineOS;
attribute CpuArchitecture machineArch;
attribute StringList candidateMachines;
attribute string reservationId;
readonly attribute StringList attributeNames;
 …
[language-specific operations  implementation-specific attributes]
```

DRMAA v2 Job

```
interface Job {
     readonly attribute string jobId;
      void suspend()
      void resume()
      void hold()
      void release()
      void terminate()
      JobState getState(out native subState)
      void waitStarted(in long long timeout)
      void waitTerminated(in long long timeout)
      JobInfo getInfo()
};
```

- Heavy cleanup, new *Job* object as root concept (still maps to string in C)
- *drmaa_control(string, JobControlAction)* replaced by dedicated methods
- *waitStarted()* and *waitTerminated()* as on *JobSession* level
- New *subState* concept for implementation-specific state information
- Explicit fetching of job information (instead of implicit *drmaa_wait()* result)

DRMAA v2 Job Info (March 24th)

```
valuetype JobInfo {
 readonly attribute string jobId;
 readonly attribute Dictionary resourceUsage;
 readonly attribute boolean hasExited;
 readonly attribute long exitStatus;
 readonly attribute boolean hasSignaled;
 readonly attribute string terminatingSignal;
 readonly attribute boolean hasCoreDump;
 readonly attribute boolean wasAborted;
 readonly attribute string errorReason;
 readonly attribute JobState jobState;
 readonly attribute string jobSubState;
 readonly attribute OrderedStringList allocatedMachines;
 readonly attribute string submissionMachine;
 readonly attribute string jobOwner;
 readonly attribute TimeAmount wallclockTime;
 readonly attribute TimeAmount wallclockLimit;
 readonly attribute long cpuTime;
 readonly attribute AbsoluteTime submissionTime;
 readonly attribute AbsoluteTime dispatchTime;
 readonly attribute AbsoluteTime finishTime;
```

DRMAA v2 Advance Reservation

DRMAA
Distributed Resource Management Application API — www.drmaa.org

```
interface ReservationSession {
  Reservation requestReservation(in ReservationTemplate rt)
  sequence<Reservation> getReservations();
  ...};

interface  ReservationTemplate {
  attribute string reservationName;
  attribute string configurationName;
  attribute AbsoluteTime startTime;
  attribute AbsoluteTime endTime;
  attribute TimeAmount duration;
  attribute long minSlots;
  attribute long maxSlots;
  attribute StringList candidateMachines;
  attribute long minPhysMemory;
  attribute OperatingSystem machineOS;
  attribute CpuArchitecture machineArch;
  attribute string nativeOptions;};

interface Reservation {
   readonly attribute sequence <string> reservedMachines;
   readonly attribute  reservedStartTime;
   void terminate();
   ...};
```

DRMAA v2 Monitoring (March 24th)

```
interface MonitoringSession {
   //// attributes on DRM system level ////
   readonly attribute StringList drmVersionString;
   ...
   //// attributes on session level ////
   readonly attribute StringList drmMachineNames;
   ...
   //// attributes on machine level ////
   long machineSockets(in string machineName);
   long machineCoresPerSocket(in string machineName);
   long machineLoad(in string machineName, in  coreNumber);
   long machinePhysMemory(in string machineName);
   long machineVirtMemory(in string machineName);
   OperatingSystem machineOS(in string machineName);
   string machineOSVersion(in string machineName);
   CpuArchitecture machineArch(in string machineName)
   ...
};
```

Other Decisions

- Some removals (different hold states, partial time stamps) and renamings
- Many things are still rejected - security, job signalling, pending job changing
- Very clear relationship to SAGA
  - DRMAA is the portability layer, SAGA is the feature layer (e.g. async calls)
- Still impressive list of open issues :(
- Re-use vs. mapping of other OGF specs
  - DRMAA still lives in the non-XML world
  - Provide mapping to JSDL and OGSA-BES enum‘s wherever it makes sense (e.g. `OperatingSystem` and `CpuArchitecture` enumeration)
  - Re-use OGF schemas for monitoring and configuration name attributes

Participation

- Please talk with us
  - Subscribe to mailing list (check www.drmaa.org)
  - Bi-weekly phone conference (Tuesday, 19:00 UTC)
- We need
  - Fresh ideas (still)
  - API design proposals for unsolved issues
  - Check for DRMS implementability (LSF, PBS, EGEE, GAT - anybody ?)
  - Check for language binding issues
  - Your implementation story
- Latest spec: **http://wikis.sun.com/display/DRMAAv2/**