

DRMAA v2 - The Next Generation

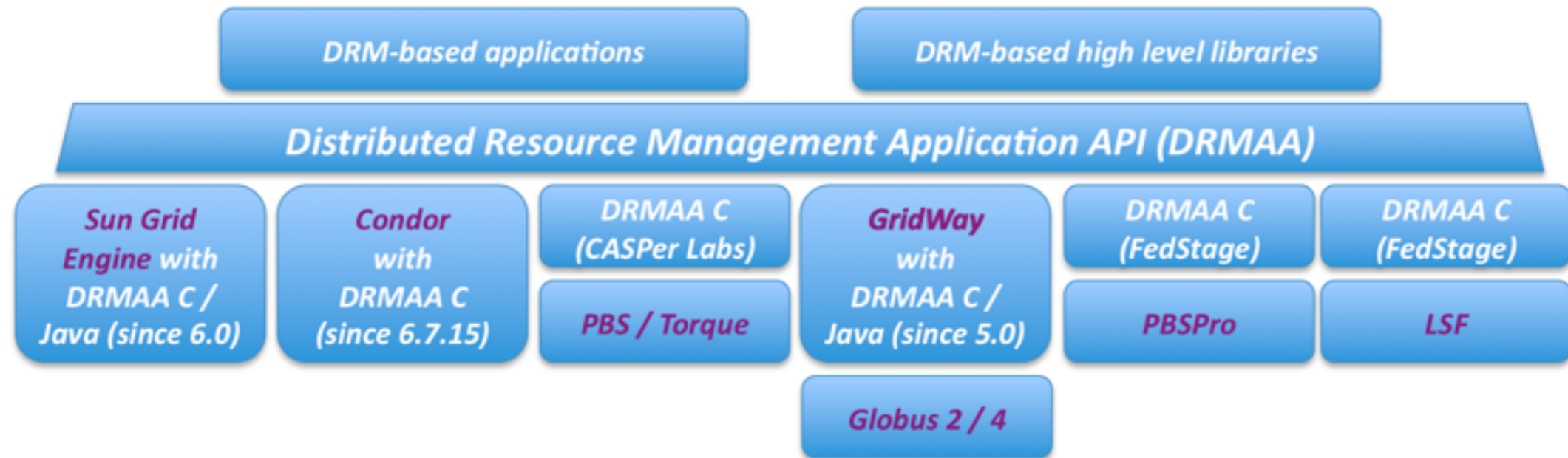
Peter Tröger
Humboldt University of Berlin
peter@troeger.eu

DRMAA-WG Co-Chair

Past

- GGF / OGF specification since 2001
 - Job submission and control in a cluster / grid system
 - Application portability between different DRM systems
 - Simple API design, implementation as local library
 - Leave room for areas of disagreement
- Different DRMAA 1.0 standardization documents
 - June 2004 - DRMAA 1.0 proposed recommendation (GFD.22)
 - 2008 - Shift to IDL based root specification, some clarifications (GFD.130)
 - Official language binding documents for C, Java, Python
 - Experience reports, tutorials, unofficial language bindings for Perl, Ruby and C#

Today



- C-library implementations for all major DRM systems, some also with Java binding
 - Biggest user base with SGE implementation
 - Some famous applications: MOAB, Mathematica integration package, SAGA
- Recent collection of user demands and wishes
 - Public survey, SUN customer feedback, DRMAA implementation experiences
- Design of DRMAA v2 **happens now ! (Deadline: December 2009)**

DRMAA v1 Issues

- Fix C-centric API design
 - Start from IDL version of DRMAA v1
 - Make the API really OO-friendly, but still language-independent
- Add new features
 - Resource monitoring, session handling, job objects, ...
- Remove obsolete / never implemented features
 - Date / time handling, ...
- Modify existing features for better usability / DRMS compatibility
 - Job synchronization, state model, job monitoring, ...

IDL-based Language Binding

- All behavioral aspects in the root spec
 - API feature set, functional behavior, error conditions, multithreading issues
- Language binding provides syntactical mapping only (Example: GFD.143)
- 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  
  
    interface JobInfo  
  
    ...  
}
```

```
module DRMAA2{  
  
    interface SessionManager  
  
    interface JobSession  
  
    interface MonitoringSession  
  
    interface JobTemplate  
  
    interface Job  
  
    interface JobInfo  
  
    ...  
}
```

DRMAA v2 Session Management

```
interface SessionManager{  
    readonly attribute string drmsInfo;  
    readonly attribute Version version;  
    JobSession createJobSession(in string sessionName, in string contactString)  
    void closeJobSession(in JobSession s)  
    void destroyJobSession(in string sessionName)  
    string[] getJobSessions()  
    MonitoringSession createMonitoringSession (in string contactString)  
    void closeMonitoringSession(in MonitoringSession s)  
};
```

- Create multiple sessions to one / more DRM systems at the same time
- Distinguishing between job management and machine monitoring
- *JobSession* instances are restartable by their *sessionName*
- Design of *MonitoringSession* interface is still unclear 
 - Intended for ,global view‘ of the DRM system, regardless of submission session

DRMAA v2 Job Session

```
interface DrmaaCallback {  
    void notify(in DrmaaNotification event)  
  
interface JobSession{  
    readonly attribute string contact;  
    void registerEventNotification(in DrmaaCallback callback)  
        raises UnsupportedFeatureExeption, ....  
    JobTemplate createJobTemplate()  
    void deleteJobTemplate(in DRMAA::JobTemplate jobTemplate)  
    Job runJob(in DRMAA::JobTemplate jobTemplate)  
    sequence<Job> runBulkJobs(...)  
    sequence<Job> waitAnyStarted(in sequence<Job> jobs, in long long timeout)  
    sequence<Job> waitAnyTerminated(in sequence<Job> jobs, in long long timeout)  
};
```



- Optional support for event push notification
- *waitAnyStarted()*: Wait for one of the „start states“ to happen
 - *RUNNING, *_SUSPENDED*
- *waitAnyTerminated()*: Wait for *FAILED / DONE* to happen

DRMAA v2 Job

```
interface Job {  
    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()  
};
```

- New *Job* object as root concept (still represented by string in C-binding)
- *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

```
interface JobInfo {  
    readonly attribute Dictionary resourceUsage;  
    readonly attribute boolean hasExited;  
    readonly attribute long exitStatus;  
    ... [old DRMAA1 job information] ...  
    readonly attribute JobState jobState;  
    readonly attribute string jobSubState;  
    readonly attribute string masterMachine;  
    readonly attribute string[] slaveMachines;  
    readonly attribute string submissionMachine;  
    readonly attribute string jobOwner;  
    // amount of time since job was started  
    readonly attribute long wallclockTime;  
    // amount of time remaining until the job will be terminated  
    readonly attribute long wallclockLimit;  
    // amount of CPU seconds consumed  
    readonly attribute long cpuTime;  
    // and so on for submission time, dispatch time, start time, finish time,  
    // memory usage and limits  
    ...  
};
```



Other Decisions

- Some removals (different hold states, partial time stamps) and renamings
- Some things are still rejected - security, job signalling, pending job changing
- Still huge list of open issues
 - Supported job and machine monitoring attributes
 - Maybe DRM monitoring (e.g. list of hosts, queues)
 - Possible new states (e.g. re-scheduled)
 - File transfer capabilities
 - Resource requirement specification in job template
 - More job template macros
 - Workflow support
 - DRMAA JSDL profile
 - ...

Participation

- Please talk with us
 - Subscribe to mailing list (check www.drmaa.org)
 - Bi-weekly phone conference (Tuesday, 19:00 UTC)
 - @Sun: Daniel Templeton, Daniel Gruber
- We need
 - Fresh ideas (still)
 - API design proposals for unsolved issues
 - Check for DRMS implementability (LSF, PBS, or EGEE, anybody ?)
 - Check for language binding issues
 - Your implementation story