GridPP
UK Computing for Particle Physics

Task and Resource Descriptions for Computational Grids

Ian Stokes-Rees
University of Oxford
Department of Particle Physics
20 September 2005

Oxford Physics

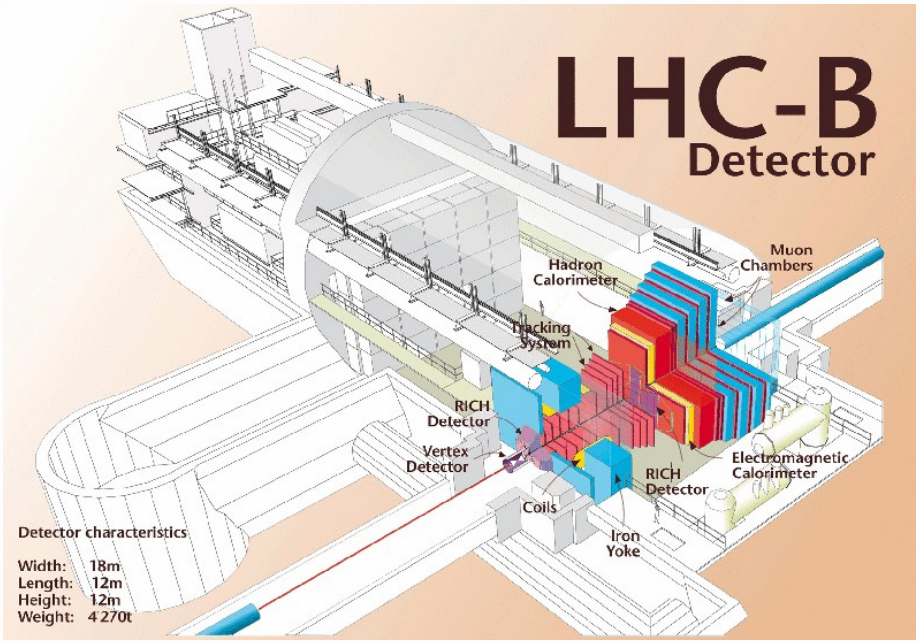
Overview

- Motivation
  - Scalable, Robust, Decoupled Computational Grid
  - Investigate unifying description languages
- Background
  - Operating Systems, Batch Systems, Grids
- Proposed Syntax
  - XML based, multi-faceted, extensible

GridPP
UK Computing for Particle Physics

LHCb Computational Grid Requirements

- Robust mechanism for task management
- Responsive with operations on 1000+ jobs
- Scalable to support large job volumes
- Extensible to allow easy enhancements
- Diverse computational resource configurations
- Maintainable over long operational periods



Grid Task and Resource Descriptions
UK e-Science AHM, Sept 2005

Oxford Physics

LHCb Computational Grid

100,000 queued jobs

10,000 running jobs

100 sites

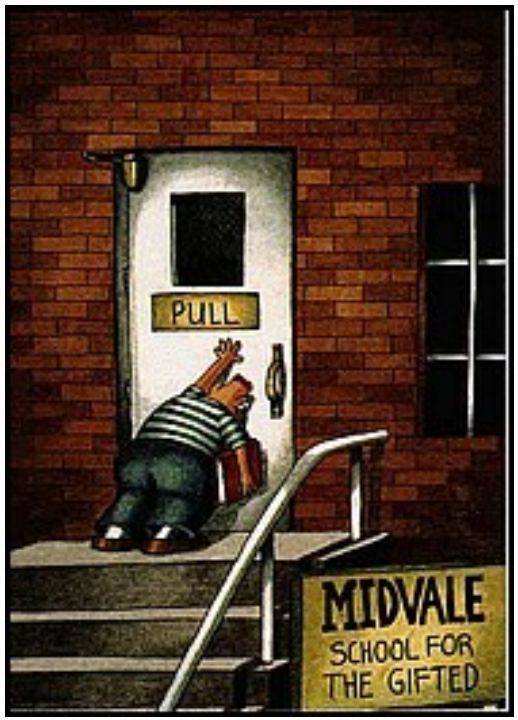
This is what our computational grid looks like

Traditional Situation

- Tasks: Basic job description or shell script
- Job Parameters: set on command line at submit time
- Centralised queue manager and scheduler
- Master/Slave compute hierarchy (head & worker nodes)
- Resource state information (*assumed to be*)
  - complete
  - correct
  - current
- Use of LDAP, R-GMA, MDS, BDII, etc. for aggregating and reporting resource state
  - basis of scheduling decisions

Push Scheduling

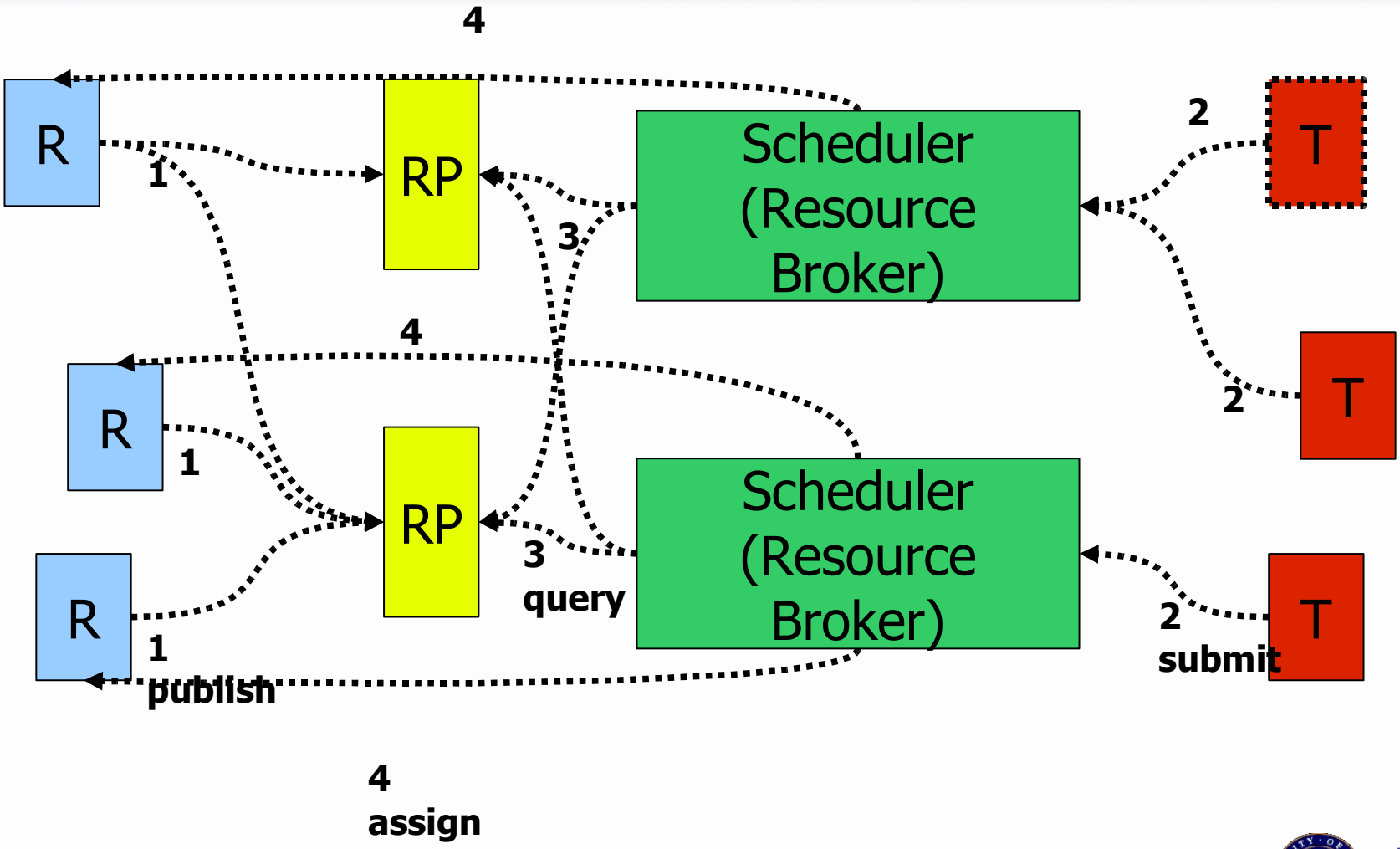
- Central scheduler uses knowledge of "grid resource state" to assign submitted task to specific resource based on a combination of task parameters and resource policy
- Problems:
  - changing resource policy
  - heterogeneity
  - dynamic resource state
  - dynamic grid topology
  - bottleneck NP-hard scheduling
  - single point of failure



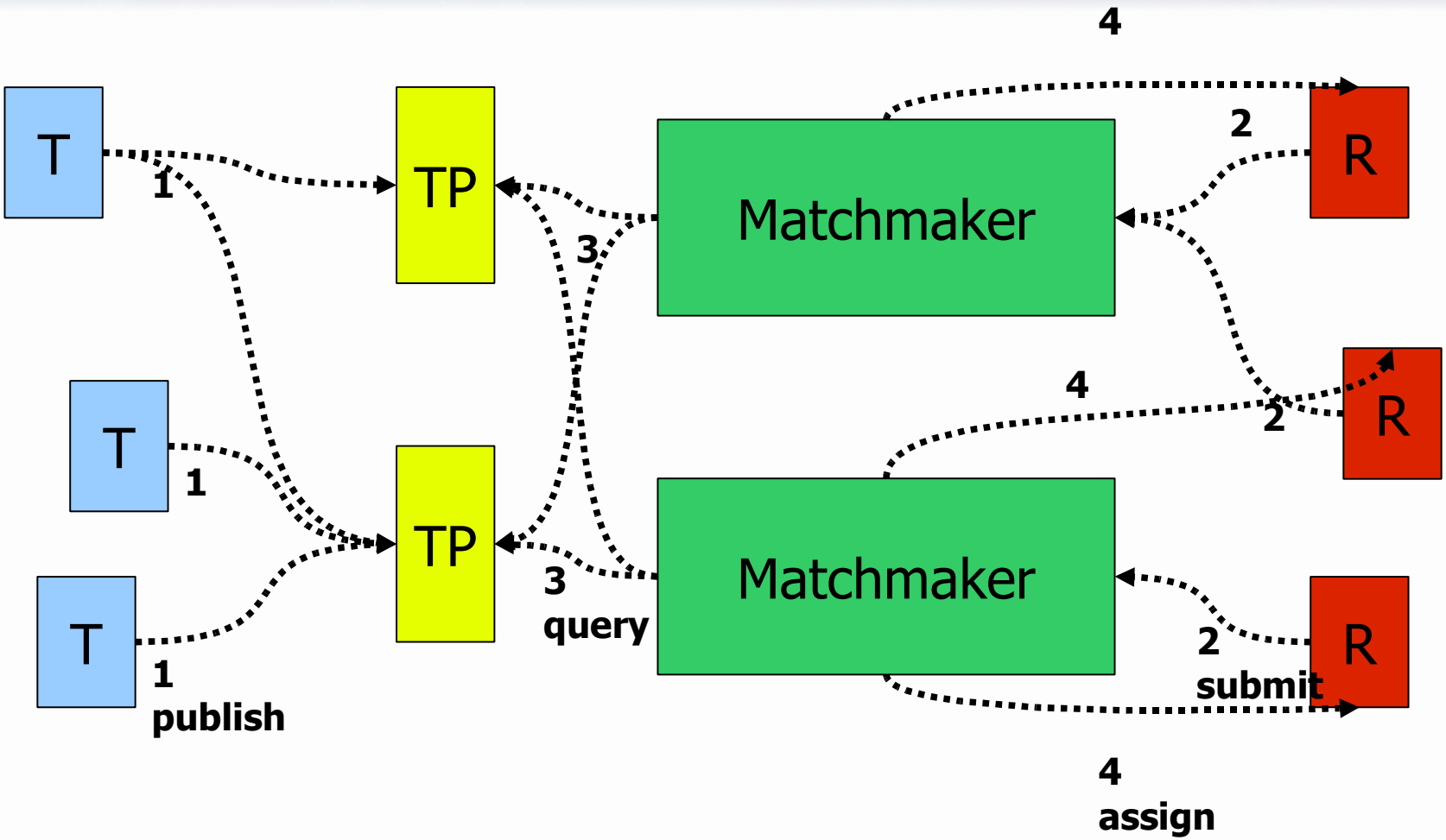
Symmetric Matchmaking

- Remove centralised scheduler
- Allow push and pull scheduling
- Tasks can publish their specific, current requirements to arbitrary locations
  - Then resources can subscribe to these and choose (or not) to execute the tasks **(pull scheduling)**
- Resources can publish their specific, current information to arbitrary locations
  - Then a task manager (or user) can subscribe to these and choose (or not) to submit the task directly to an appropriate resource **(push scheduling)**
- Also allows a third-party scheduler to subscribe to multiple task and resource pools to find matches
- Not a new idea! The Condor Project proposed this in 1998 with ClassAds and Matchmaking (Raman, Livny, Solomon)
- So what **is** new? ...

Push Scheduling



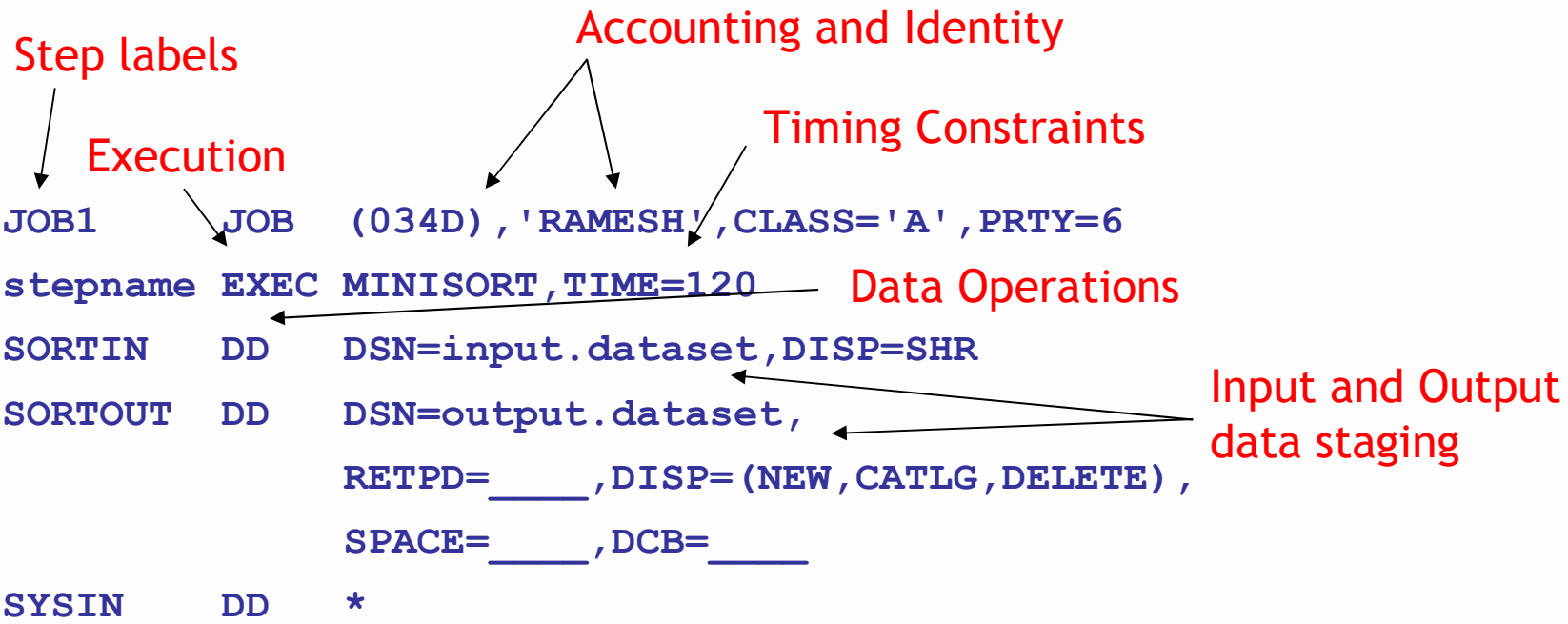
Pull Scheduling



How It's Been Done Before

Job Control Language (JCL)

- circa 1970
- Developed for IBM mainframes and MVS OS



GridPP
UK Computing for Particle Physics

Unix Process Tables

- Identity
  - individual and group
- Current State
  - Running, Ready, Blocked, Zombie
- Data access
  - open file handles
- Timers
  - wall clock, CPU, system, user, custom
- Execution
  - program counter
  - current working directory
  - memory pointers
- Dependencies
  - parent, child, sibling processes
- "Hand of God" control
  - thanks to kernel
  - atomic
  - "instantaneous"

Oxford Physics

PBS/Torque

- Task parameters passed on command line, or via embedded **#PBS** commands in the script header

```
#!/bin/sh
#PBS -N My Test Job
#PBS -o test.log
#PBS -e test.err
#PBS -m ian@example.com
#PBS -q long
#PBS -l nodes=8
#PBS -l walltime=12:00:00
#PBS -l cput=4:00:00
#PBS -l mem=512mb
<script body>
```

- Queue parameters are statically defined via a configuration file
- Possible to manipulate some of these at runtime

```
set queue long queue_type  = Execution
set queue long Priority    = 60
set queue long max_running = 10
set queue long enabled     = True
set queue long started     = True
set queue long resources_max.cput = 12:00:00
set queue long resources_min.cput = 02:00:01
```

Resource Specification Language (RSL)

- Developed by the Globus Project
- Only allows task description
- Provides Or (|) and Multi-select (+) (for co-allocation)

```
& (executable   = myscript.sh)
  (arguments    = 42 "Header String")
  (directory    = /home/nobody)
  (environment  = (PI 3.14) (PATH ~) (ORIG_USER joe))
  (count        = 3)
```

ClassAds

```
Type       = "Machine";
Disk       = 323496;
Memory     = 64;
State      = "Unclaimed";
Name       = "leonardo.cs.wisc.edu";
Friends    = { "tannenba", "wright" };
Rank       = member(other.Owner, ResearchGroup);
Constraint = !member(other.Owner, Untrusted);
Untrusted  = { "rival", "riffraff" };
ResearchGroup = { "raman", "miron" };
```

Attributes

Numbers

Strings

Sets

Functions

Target ClassAd

Job Description Language

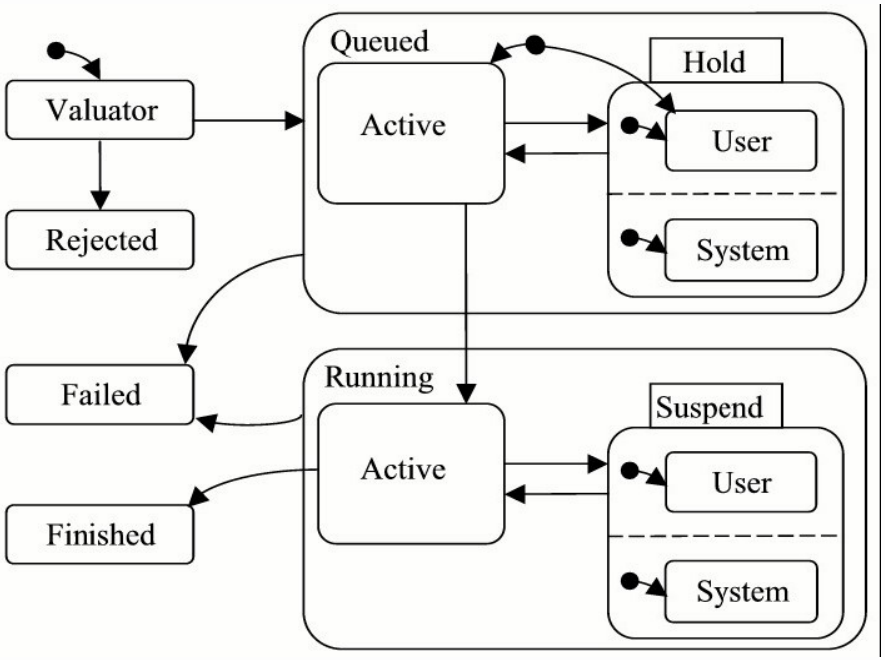
- Base syntax 100% ClassAds
- Adds some well-defined attributes
  - ClassAds have no pre-defined attributes
- Utilises GLUE schema for resource attributes
- Ignores or defers many properties
  - timing
  - ownership
  - logging

GLUE Schema

- Standardized rich description of computing resources
- Focuses on aggregated properties of site
  - assumes homogeneity
- Mix of static and dynamic properties
- Typically stored in DB (MDS, BDII, LDAP, R-GMA)

Distributed Resource Management Application API (DRMAA)

- v1.0 Agreed by GGF June 2004 (GFD.022)
- Provides task submission and execution model
- Specifies task details:
  - Name
  - Timing: start, stop, cpu time,wall time
  - Command to execute and parameters
  - Task environment
  - Standard input, output, and error streams
  - E-mail distribution list
- Plus arbitrary task attributes



DRMAA (2)

- Functions:
  - init, exit
  - **attribute**: set, get
  - **job_template**: allocate, set, delete
  - **control**: run, wait, synchronize, suspend, terminate
  - **monitoring**: ps

- Primarily a function-oriented API for DRMs
- Minimal data support
- Focused on programatic interaction with DRMs
- Allows arbitrary task management process

Job Submission Description Language (JSDL)

- JDSL v1.0 draft 21, 16 June 2005
- Three categories of information
  - Job identification
  - Resource requirements
  - Data requirements
- Only addresses submission requirements
  - does not address task lifecycle
  - does not address inter-task dependencies
  - minimal data handling
    - intentionally deferred to other (future) standards
- XML-based

JDSL (2)

- Inferred future languages:
  - RRL Resource Requirements Language
  - SDL Scheduling Description Language
  - JPL Job Policy Language
  - JLML Job Lifetime Management Language
  - WS-AG Web Services Agreement

GridPP
UK Computing for Particle Physics

Grid Scheduling Architecture

- GGF Research Group
- Not aiming to define syntax, but to define semantics and identify problem space
- Concepts of
  - Meta-schedulers
  - Resource Selection and Prioritisation
  - Candidate Sets
- Numerous Use Cases

Oxford Physics

Putting it all together

Inspirations

- ClassAds
  - Matchmaking
- GLUE
  - Resource descriptions
- JSDL
  - Task descriptions
- DRMAA
  - Process model
  - API
- GSA-RG
  - Use Cases
- REST
  - Encapsulated representation of resource state

Security and Correctness Considerations

- Unauthorized execution
  - Protected data space
  - Not recorded
  - Not charged
- Unauthorized data access
  - ACLs
- Information Leakage
  - Recipe
  - Meta-data
- N-executions
  - Exact
  - At least
  - At most

GridPP
UK Computing for Particle Physics

Requirements

- Identify Resource Type
- Describe Resource
- Define Requirements/Restrictions
- Match Preferences
- Logging and Notification
- Authentication and Authorization
- Execution Workflow
- Current State
- Resource Identification and back-reference

Oxford Physics

KISS (???)

- Underlying principle to Keep It Simple, Stupid
  - But worried that may be slipping away
- Common situations should be easy and obvious to express, and complex situations possible
  - *Principle of Least Surprise*

Simple Examples

```
<task>
    <exec cmd="ls"/>
</task>
```

```
<resource>
    <chars>
        <os ver="3.0.4">RHEL</os>
        <start>Oct 3 9:00:00 2005</start>
        <finish>Oct 7 9:00:00 2005</finish>
    </chars>
    <reqs>
        <vo match="inset">lhcb atlas</vo>
    </reqs>
</resource>
```

Numerous Aspects

ACLs

Preferences

State

Accounting

Data
Staging

Notification

Workflow

Scheduling

Characteristics

Identities

Exception
Handling

Requirements

Key Resource Features

```
<task>
    <chars>
        Task characteristics (internal)
    </chars>
    <reqs>
        Task requirements (external)
    </reqs>
    <prefs>
        Preferences for selecting
        from candidate set
    <prefs>
    ...
</task>
```

- Allows arbitrary attributes
- GLUE, JDL, and JSDL attributes all legal
- Extend syntax to accept
  - units="K|KB|M|MB|..."
  - match="<|>|==|!=|..."
  - scale="<*factor*>"
  - rank="<*float*>"
  - version="*v.m.p*"

```
<chars>
    <ntime units="">normalized exec time</ntime>
</chars>
<reqs>
    <start>earliest start</start>
    <finish>latest finish</finish>
    <sw ver="1.2.3">software name</start>
    <lib ver="1.2.3">library name</start>
</reqs>
```

Notification Conditions

- allevts
- statechange
- stdout
- stderr
- sched
- task
- stagein
- stageout
- ready
- claimed
- install
- exec
- hold
- done
- error
- fail
- success

Execution Environment

- Staged-in data
- Embedded files
- Software installation
- Migrating environment
- Fetching env via URL
- Setting env directly

GridPP
UK Computing for Particle Physics

Authentication and Authorization

- Provides for multiple identity/ authorization tokens to exist concurrently
- Fine-grained ACL
  - Time-bound permissions
- Different operations or services require different identities to be presented
  - Consumer not always able to select correctly if simply a full "identity wallet" is presented

Oxford Physics

Accounting

- Already seeing high contention for grid resources
- Expect billing to be neccessary
- All operations which consume a resource should specify an account to use
- Allow billing for execution, data movement, and data storage

Task Stages

- Support for different task stages, hierarchies, and inter-relationships
  - install or configure software
  - data movement
  - compilation
  - execution
  - concurrent operations
- Meant to allow task pipelining

Issues and Future

- Maintaining simplicity
- The Devil in the Details
  - Wildcards and Variables
  - Binding/resolution
- Security
  - Multiple temporal identity mgmt is complicated
- Task Process Model
  - requires further development and testing
- Software support
  - Ideally combination of: Java, C++, Perl, Python, web-based, and command line tools and libraries
  - Starting with just Python lib and CLI
- Integration with schedulers/WMS

GridPP
UK Computing for Particle Physics

Acknowledgements and Contact Details

This work has been supported by:

Ian Stokes-Rees
i.stokes-rees@physics.ox.ac.uk
http://grid.physics.ox.ac.uk/~stokes/

(NB: Looking for a post-doc or similar position starting January 2006)

Oxford Physics

60 second version

- CERN based LHCb experiment has developed a generic computational grid infrastructure which utilises existing grid (e.g. LCG) and non-grid resources
- Required both task and compute resource descriptions in common format for
  - push and pull scheduling
  - easy and robust tool development
  - extensibility
- ClassAd syntax and tools from Condor were the obvious first choice
  - but lacked a number of key features
- Examined diverse range of task and resource description languages
  - Batch systems (JCL, PBS, LSF, Condor)
  - Existing Grid description languages (RSL, JDL, GLUE)
  - Emerging standards (JSDL, DRMAA, CDDLM)
- Identified requirements of:
  - language
  - usage
  - descriptive properties
  - execution properties
- Resulted in XML-based syntax which builds on ClassAds, JDL, JSDL, and GLUE