User Level Scheduling in the Grid

the outline of the technology potential and the research directions

```
The User Level Scheduling (ULS) techniques have been very successfully
applied in a number of application areas such as bio-informatics,
image processing, telecommunications and physics simulation. The ULS
helped to improve Quality of Service (QoS) in the Grid, what have been
experienced as reduced job turnaround time, more efficient usage of
resources, more predictable, reliable and stable application
execution. To be universally adopted however, the ULS techniques must
be proved to be compatible with the fundamental assumptions of the
Grid computing model such as respect for the resource usage policies,
fair-share toward other users, traceability of user's activities and
so on.  In this talk we will the outline the main benefits and
possible pitfalls of the ULS techniques. We will also try to introduce
initial research ideas for modeling and measuring of the Quality of
Service on the Grid and for analysis of the impact of the ULS on other
users (fair-share). Finally we will present ideas for enhanced support
for certain applications such as the iterative algorithms or
parameter-sweep.

This presentation builds on the "The Quality of Service on the Grid with
user-level scheduling" presented at UvA on September 1st 2006.
```

Start Presentation PhDSlideTemplate SlideShowPlugin

Slide 1: Table of Contents

User Level Scheduling in the Grid

the outline of the technology potential and the research directions

- Introduction
  - the context of my work
  - summary of the activities so far
  - short description what ULS is
- Research ideas
  - definition of QoS metrics
  - analysis of the impact: fair share
- Development ideas
  - enhanced support for iterative and parameter-sweep applications
- Summary

Slide 2: Context of my work

- ARDA group at CERN

- enabling applications and users on the Grid:
  - High Energy Physics: LHC experiments
  - EGEE: biomed, special acitvities
- Grid
  - LCG and EGEE Grid

- the largest Grid infrastructure to date
- over 200 sites
- over 20K worker nodes
- over 5 Pb of storage

Slide 3: Activities

- Succesful activities:
  - *EGEE helps achieve international digital broadcasting agreement*
    - http://www.gridtoday.com/grid/728292.html
  - *Grid searches for avian flu cure*
    - http://news.bbc.co.uk/2/hi/technology/4977150.stm
- We exploit the **User Level Scheduling** in order to achieve better service from the Grid
  - **DIANE** (DIstributed ANalysis Environment) is a prototype of an ULS system
- More info in the UvA seminar 1 Sept 2006
  - http://moscicki.org/Amsterdam-seminar-2006.pdf

Slide 4: Placeholders and late binding

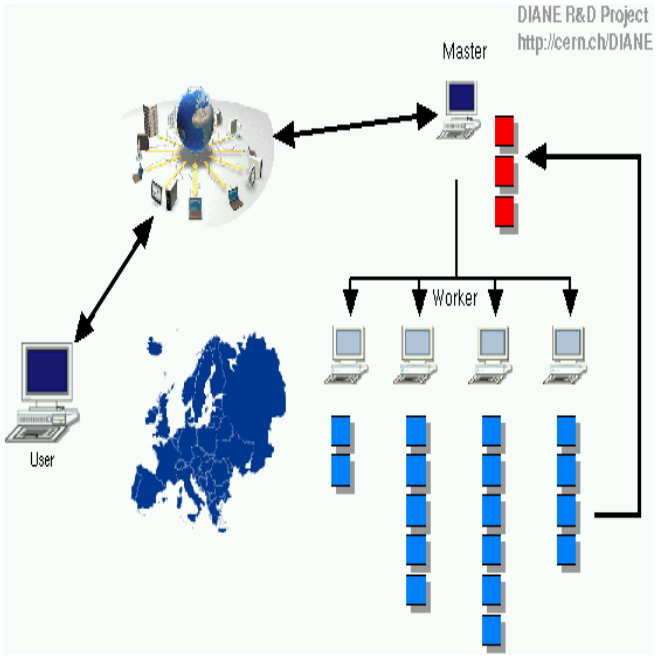
- the technology is also called: placeholder, late binding, pilot jobs
- you do not send specific job to the resource
- you acquire the resource and assign the job dynamically to it
- you free the resource when you are done
- some examples:
  - HEP production systems (centralized task queue, server acts on behalf of the user):
    - Alien (Atlas), DIRAC (LHCb), PANDA (Atlas)
  - Condor glide-ins (build a virtual Condor pool from Globus resources)
  - Boinc (CPU cycle scavanging)

Slide 5: User Level Scheduling

- it's the late binding technology
- the scheduler has the application knowledge
  - may make better error recovery or load balancing decisions
- it runs in the user space
  - resources are accountable for and tracability is not compromized
- it is capable of creating transient/volatile overlays on top of the regular infrastructure
  - "virtual clusters"

Slide 6: DIANE prototype

- DIANE prototype



  - ♦ Master/Worker architecture with cutomizable task dependencies, constraints and fault tolerance
  - ♦ not specific to any particular technology or infrastructure:
    - ◊ Grid, LSF/PBS, explicit IP host list + mixing of the resources
  - ♦ portable (python and CORBA)
  - ♦ self-contained and small distribution with fully automatic installation

Slide 7: Area of applicability

- **communication non-intensive** applications:
  - ♦ GOOD examples
    - ◊ **data analysis**
    - ◊ **physics simulation** (independent tasks)
    - ◊ **image processing** (iterative)
    - ◊ **docking** (bioinformatics parameter sweep)
  - ♦ BAD example:
    - ◊ FME with thousands of cells communicating every few computing cycles
- ability to partition the problem into parametric tasks - units of execution
  - ♦ parameters are passed around to start the application execution unit - so no classic task migration

Slide 8: Review of User Level Scheduling

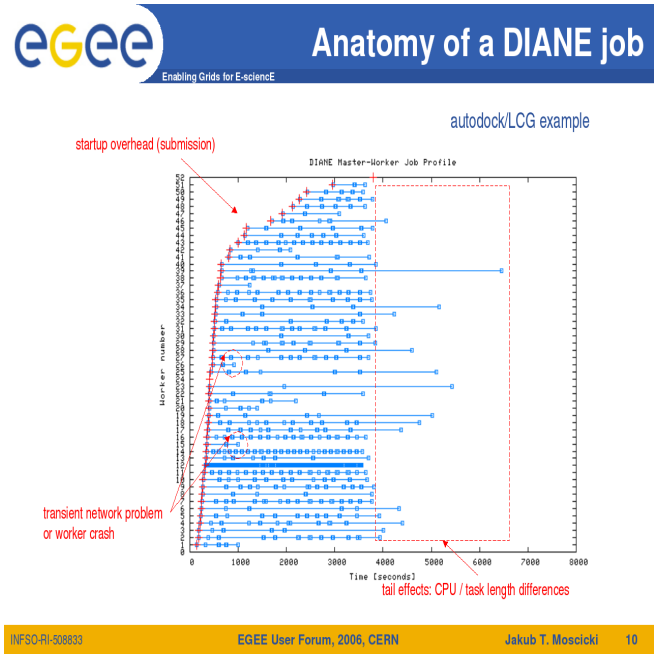
- Improvement of QoS characteristics
  - ♦ extra reliability: fail-safety and application-specific fine tuning
  - ♦ reduction of turnaround time
  - ♦ predictability of job execution (and more stable output inter-arrival rate)
- Potential flaws
  - ♦ effects on fair-share: would other users be penalized by ULS jobs?
  - ♦ potential harmfullness of the redundant batch requests

Slide 9: Research Ideas

- Quantify what we already experimented and sucessfully applied
  - ♦ what it means "a better service" from the Grid?
    - ◊ formalization of how Grid resources are acquired (slot model)
    - ◊ measuring the usefulness of ULS
      - · definition of quantifiable QoS metrics
      - · identification and analysis of potential flaws
- Can we provide useful estimates of the QoS and fulfil them (in statistical sense)?
  - ♦ given reliable methods of predicting Grid queing time exist
- Can we **assure** the QoS using mixed, Grid and dedicated, resources
- We want to do simulation (if required) and practical verification using real applications

Slide 10: Slot model for Grid resources

- slot i is defined by:



  - ♦ tq(i) = queuing time
  - ♦ ta(i) = available computing time (wallclock)
  - ♦ p(i) = weight (power) of the resource
- W is the total work-load of the job

Slide 11: Estimation of the number of needed slots

- N = minimal number of slots needed:

$$W + overhead = \sum_{i=1}^{N} ta_i * p_i$$

- **overhead** represents:
  - ♦ the ULS overhead (networking, scheduling)
  - ♦ how well we can tile the slot with units of execution ("the tails")
- additional complication: p(i) may change with time
  - ♦ in case of time-sharing on the worker node with other jobs

Slide 12: Estimation of the number of needed slots (2)

- Main options:
  - a largely redundant slot acquisition at the beginning
  - adding a capability to acquire more resources on demand while the jobs are running
- Current practice:
  - we do a rough estimation of the total CPU demand and then request a double or so slots assuming some average processor power (largely ficticious)
- Future extensions:
  - initial estimation could be done from the Grid information system
  - the application-specific benchmarks (so the **measured** p) could be stored externally and reused in subsequent runs

Slide 13: Estimation of the turnaround time

- Currently we do not predict the tq - queing time, however:
  - promising techniques exist (e.g. BMBP Binomial Method Batch Predictor)
    - relying on long traces from batch-queue logs + parallel workload archives
    - using *order-based statistics* (**quantiles**) rather than *moment-based statistics* (**mean, stddev**)
    - arguably in comparison to model-based predictors (which assume certain model of the queue)
      - BMBP has a similar success rate but has more accurate predictions (less "over-prediction")
  - we have a wealth of monitoring data from user activities (ARDA Dashboard e.g. http://lxarda02.cern.ch:8088/atlas)
  - additionally we try to capture the Grid 'background'
    - by sending very short jobs a few times daily to monitor the responsiveness of the system (in 3 different VOs)

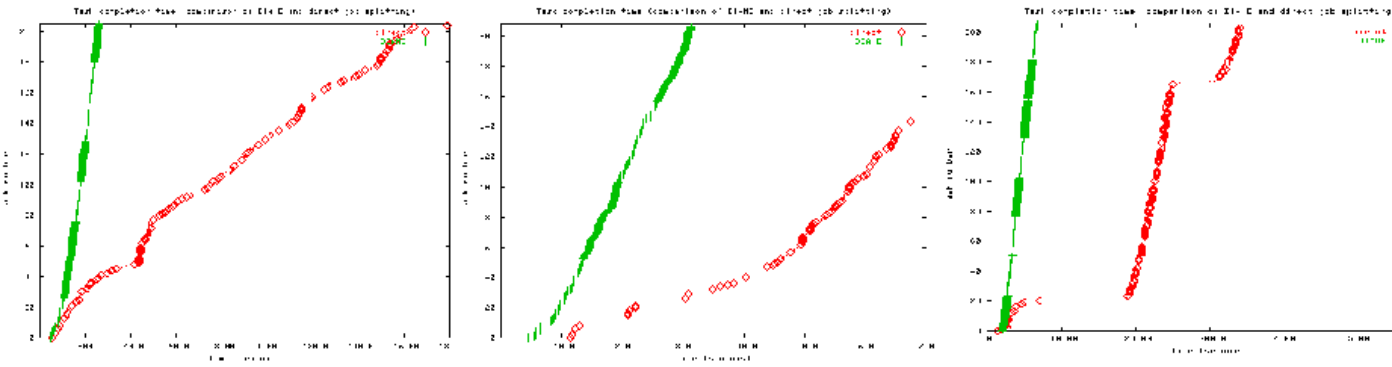
Slide 14: Estimation of the turnaround time (2)

- Assuming that we can get reliable estimates on tq on the Grid (is it possible/been done yet?)
- Let **P(k,t) = probability to complete the k% of the run in the time not greater than t**
  - in real applications the user may also be interested in partial output (e.g. histograms)
  - similar to modelling the queuing time in BMBP
- P depends on how k is defined:
  - if all execution units are equal or their distribution is known
  - if there are synchronization patterns or execution units are independent
- P depends also on other parameters:
  - change in the load of the worker node and a failure rate of the worker nodes
  - the contraints: may any worker node process any task or some are restricted (for example the data access)

Slide 15: Predictability of the job execution

- Define **the coefficient of the variation of the job output arrival times**
  - depends on k i.e. the performance model
    - if all execution units are equal and independent then the expected output arrival time vs time is a straight line

- we can do the fitting of known performance model to the measured data points



Slide 16: Fault-tolerance and reliability

- Reliability should be measured "on the application's reliability background":
  - ♦ the goal: minimize the infrastructure faults which have no relation to intrinsic application problems
    - ◊ example of instrinsic application problem: a segfault on particular data or parameter
  - ♦ We can enhance reliability of the system as observed by the user by providing fault-tolerance:
    - ◊ for configuration problems on the sites, middleware inefficiency, worker node crash etc.
    - ◊ we can customize the fault-tolerance behaviour
- How to measure the reliability?
  - ♦ Classify the faults, disregard the intrinsic application faults, **the ratio of failures** is the reliability measure

Slide 17: Fair share

- **would other users be penalized by ULS jobs?**
- potential harmfullness of the redundant batch requests
  - ♦ pure redundant requests (submit n, execute 1, cancel n-1) have been studied (-> paper)
    - ◊ jobs which do not use redundant requests are penalized
      - · stretch increses linearly wrt the number of jobs using redundant requests
    - ◊ load on middleware may be a problem

Slide 18: Fair share (2)

- ♦ ULS have certain degree of redundancy (submit n, execute k, cancel n-k)
  - ◊ estimate the level of redundancy
  - ◊ measure the harmfullness in this case
- ♦ Possible solution:
  - ◊ meta-submitter would steadily increase the number of submission according to needs
  - ◊ this is clearly in conflict with minimizing the global turnaround time (QoS)
  - ◊ what should the balance be?

Slide 19: Fair share (3)

- **would fair share policies defined in the CE be compromised?**
  - heterogenity: fair share algorithms are different at different sites
  - concepts of window, depth, decay in the accounting data are applied to VO, groups and users
- fair-share may be modeled and simulated
  - e.g. SimGrid toolkit for sites,clusters,batch schedulers and streams of jobs
- Other pitfalls:
  - Very large activities using ULS could potentially block-off one another from using the Grid for certain time.
- Possible solution:
  - artificial restriction of the slot (either static or dynamic)
  - in conflict with QoS

Slide 20: Other ideas

- Prediction of the availability of the worker nodes based on (semi) Markov Processes
  - used quite successfully with the FGCS (Fine-Grained Cycle Scavanging) system
  - tested with iShare: a peer-to-peer decentralized internet sharing system
  - host availability modeled as states of Markov Process, subject to interactive (and chaotic) user activity
- Could this be used?
  - jobs on the grid have identifiable states

Slide 21: Iterative applications

Iterative applications

- A new application has been recently deployed with DIANE
  - xmipp - X-Window-based Microscopy Image Processing Package
  - image alignment on the Grid
  - one step (iteration) is a full Master/Worker job
  - the stopping criteria defined by the Master
  - the workers are reused in many iterations, great speed-up

Slide 22: White-board for parameter sweep

White-board for parameter sweep

- Autodock application: searching for drug candidates in a huge compound/ligand matrix
- Do a smarter search than just all combinations (or at least do all combinations in a smarter order)
- Some rows or columns of the matrix may be more interesting than others
- A collaboration whiteboard: select interesting areas and assign a master for the job
- If one master terminates the workers may join the other master

Slide 23: Development Ideas

Development Ideas

- sending additional information (benchmarks, constraints, worker group identification)
- dynamic change of connection-oriented / connection-less policy
- support of the constraints layer
- dynamic task decomposition
    - so far we have a rather primitive self-balancing with static decomposition
    - managing the message rate on the master by splitting into larger tasks which may be dynamically decomposed into smaller units
- a directory service: late master binding, switching to multiple masters at runtime
- checkpointing of the jobs to be able to resume an interrupted master

Slide 24: Summary

- we have developed a useful technique which has been succesfully used
- first we want to quantify the improvement in the QoS we observed
- then we want to analyse possible pitfalls (fair-share)
- we also want to make the system more automatic
    - it should compute the number of needed resources to fulfil the QoS requirements
    - dynamic decomposition should relieve the user from making manual splitting decisions
- finally we want to extend our ULS prototype to cover more use-cases

-- JakubMoscicki - 06 Dec 2006

This topic: Main > AmsterdamSeminarDecember2006
Topic revision: r7 - 2007-11-07 - TWikiGuest