Resource usage accounting for UNICORE

Piotr Bała, Krzysztof Benedyczak, Marcin Lewandowski
Outline

• Aim of the project
• Current solutions
• Architecture
  – Client Tier
  – Server Tier
  – Target System Tier
• Summary and future plans
Purpose of grid accounting

• **Grants** – grid access can be based on resource quotas.

• **Grid monitoring** – allows administrators to track grid status & detect malicious users.

• **Economic aspect** – gives an ability to convert a resource usage unit to a monetary unit.

• Tuning performance of a grid.
Definitions

• Usage Record (UR)
  – XML fragment that describe consumption of a resource.
  – Standard defined by OGF as an XML Schema.
  – Fine grained: every job state can be recorded.
  – Covers all basic resources: CPU time, walltime and also network traffic and disk usage.
  – Most of needed statistical information can be represented, but some properties are missing (VO).
Definitions (2)

- **OGF-RUS**
  - Recommendation of interface to insert, update and retrieve Usage Records.
  - Not a standard - drafts available only.
  - Distributed as WSDL + XMLSchema.
  - Multiple versions, quite different.
  - Seems not to be actively maintained.
Definitions (3)

• OGF AUR
  – Aggregate Usage Representation.
  – An aggregate usage record accumulates job-level usage information for other level usage tracking.
  – Based on the UR specification.
  – Aggregates data from multiple records in a continuous period of time.
Current approach: UNICORE Accounting System

  – OGSA-RUS compliance,
  – Converts data generated at the batch system level on the fly
  – Allows clients to fetch Usage Records.
  – Stores raw data (BSS specific) in Apache Derby.

• Not publicly available(?) UNICORE SVN contains only an unfinished skeleton of a web service.
  – Only one operation implemented.
LLview

- Graphical monitoring of a cluster controlled by LoadLeveler.
- Gives quick and compact summary about the cluster status.
- Lack of User DN, not a grid tool.
EGEE Accounting Portal

- A complex accounting ecosystem that involves different accounting providers (APEL, DGAS, SGAS, and Gratia) publishing to a common central repository.
Motivation

• Accounting for UNICORE deployment in PL-Grid.
  – Developed at Interdisciplinary Center for Mathematical and Computational Modeling.

• Accounting system must be integrated with the PL-Grid gLite accounting system.

• We want to create a universal and generally useful solution.
Resource Usage Accounting for UNICORE

Diagram:
- **Client Tier**
  - Accounting Portal
  - UNICORE Commandline Client
    - RUS Plugin
- **Server Tier**
  - UNICORE Gateway
  - UNICORE WS-RF Hosting Environment
  - RUS Service
  - CRUD for UR
  - RUS Job Processor
  - RUS DAO
    - iBatis
  - DB (H2, MySQL)
- **Target System Tier**
  - RUS-Feeder
  - Accounting data parser (RMS specific)
  - RMS (e.g. Torque, LoadLeveler)
  - job execution

Legend:
- Gray: already implemented
- Light blue: to implement
• RUS Service
  – Web Service which loosely implements Resource Usage Service.
  – Adds some custom methods not present in RUS interface:
    • e.g. `extractRUSUsageRecordsByStatus(...)`
  – Provides methods for:
    • inserting URs,
    • retrieving URs by criteria.
UNICORE Accounting – Server Tier (2)

• RUS Service (cont.)
  – DAO uses iBATIS as ORM.
  – UNICORE native H2 is a default DBMS.
  – Stores UR as raw XML + indexes.
  – Is easy to configure.
  – TLS authN + XACML authZ of Unicore/X.

• RUS Job Processor
  – Gathers data from UNICORE job and sends it to RUS Service.
  – Uses XFire local transport.
UNICORE Accounting – Target System Tier

• BSS Accounting data parser
  – Generates UR from BSS log.
  – Currently support for Torque, SGE in the future.

• RUS Feeder
  – Scans log files for updates.
  – Sequential unidirectional transport with file pointer persistence.
  – Sends URs to the RUS Service.
  – TLS authentication is used.
UNICORE Accounting – Client Tier

• Accounting Portal (current work)
  – A monitoring tool.
  – Status and history of jobs.
  – Users can watch their computing history, resources used and remaining quota in grants they belong to.

• UCC plugin
  – Admin tool for quick and easy retrieving of URs by given search criteria.
    • E.g. statistics for jobs computed in the last month.
Usage record flow

1. submit job
2. queue
3. 5, 7 update UR
4. queue
5. 6, ? write to log
6. logs
7. submit

Client

Job Processor

XNJS

RUS Service

Unicore/X

TSI

Batch Subsystem

BSS Adapter

DB
Flow - merge

• Both actionUUID & (bsid, bssid) uniquely identify a UR.

• When record is found with a given actionUUID or (bsid and bssid), then update can be performed.
  – Old values are overwritten (records are merged).
AUR & EGEE

• We also provide ability to generate Aggregated Usage Records.
• Compliance with OGF AUR.
• Aggregated content will be presented in the portal.
• Aggregated records can be pushed to the central PL-Grid accounting database.
Cooperation

• Sometimes we have to export collected data to an external accounting system.
  – In PL-Grid case we forward data to enhanced BAT system.
  – BAT stands for Batch system Analysis Tool. It was developed in the Baltic Grid project.
  – The protocol is JMS based (OpenWire protocol) with custom XML messages representing usage records.

• Therefore we will provide interface for plugging various export mechanisms.
RUS draft evaluation

• There are authorization problems.
  – XPath querying can be available for admins only.
  – In UNICORE we can effectively (by using XACML engine) only authorize particular Web Service operations.

• EGEE/EGI uses JMS (ActiveMQ) to transport usage records.
  – Not only protocol but also paradigm mismatch.
First results

• Version for UNICORE 6.2.2 was evaluated for a few months on PL-Grid UNICORE testbed.
  – Embedded HSQL DB was used.
• System worked correctly, however
• HSQL DB had scalability problems. After ~100MB of SQL DB performance dropped significantly.
• Currently we are testing H2,postgresql and MySQL.
Summary

• The main goal of this project: a comprehensive accounting tool for UNICORE.

• Project in still under development.
  – Snapshot version is currently tested at ICM, Warsaw at the PL-Grid UNICORE testbed.

• It’s open source – hosted on UNICORE-Life SF project.

• Sources available at:
Thank you