Data oriented processing

Bernd Schuller¹, André Giesler¹, Richard Grunzke²

¹ Federated Systems and Data division, JSC Forschungszentrum Jülich GmbH
² ZIH, TU Dresden

18 June 2013, UNICORE Summit 2013, Leipzig
Outline

- Motivation
- High throughput / high volume data processing in UNICORE: current status
- Data oriented processing as new approach
- Current state of development and some first results
- Outlook
Motivation: High-throughput brain scans – the „Data Lifecycle Lab Health“ at Jülich

- Brain section scans (ex vivo) (~2000 slices, 500GB per slice)
- MRT scans (in vivo)
- Goal is to create a 3D brain atlas
- Post-processing: image registration, calibration, segmentation, etc
- Image processing (incl. HPC)
- Raw data often re-processed (new algorithms, new software versions)
- Issues: Workflows, Metadata, data access and movement
Types of data processing available in UNICORE

- Single jobs
- Workflow system
- Space-based approach
Single jobs

- Batch job oriented
  - Data stage-in
  - Execution
  - Data stage-out

- End-user client tasks
  - Setup job definition
  - Select site
  - Upload input data
  - Submit

- Pros
  - Very flexible

- Cons
  - High overhead
Workflow system

- Sequences / Graphs / Control
- Based on single jobs
- End-user client tasks
  - Setup workflow definition
  - Upload input data
  - Submit

- Pros
  - Easy automation of complex processes
  - For-each available
  - Low load on client side

- Cons
  - High overhead on servers
  - Data staging can be a limiting factor (→ UIMA-HPC)
Space-based approach

- Central job queue, UNICORE servers pull jobs and process them

- End-user client tasks
  - Setup job definition
  - Submit to space

See: Richard Grunzke, Bernd Schuller, „Secure high-throughput computing using UNICORE XML Spaces“ UNICORE Summit 2010

- Pros
  - Low load on client, fine tuning on U/X side possible
  - Less overhead on servers

- Cons
  - Not (yet) widely deployed
Data oriented processing
UNICORE SMS
Storage instance(s)

/.UNICORE_Rules
/directory

1. Trigger event

Evaluate rules

2. Create actions

3. Run / submit

Actions

4. Produce new data

Scripts
Batch jobs
Grid Jobs
...

...
Types of triggering events

- Periodic directory scan
  - Files can be written independently of UNICORE
  - Scan interval configurable
  - Directory include/exclude patterns

- (Explicit client invocation)

- (Finished file write(s))
Types of actions

- Local script
  - Executed via XNJS/TSI

- Local batch job
  - Executed via XNJS/TSI
  - UCC-like job description

- Grid jobs, workflows, metadata extraction, …?
An example

- **Goal**: calculate checksums (md5) of PDF files in a certain directory using batch jobs

- **Rule (job is in UCC syntax!)**

  Name: computeMD5Sum, Match: ".*\.pdf",
  Action: {
    Type: BATCH,
    Job: {
      Executable: "/usr/bin/md5sum",
      Arguments: ["${UC_FILE_PATH}"],
      Exports: [
        {From: "stdout",
         To: "file://${UC_BASE_DIR}/checksums/${UC_FILE_NAME}.md5"},
        ]
    }
  }
Example – some results

- Submission to XNJS
- Create uspace, start processing
- Running on localhost using nobatch TSI

→ Performance limited by XNJS job acceptance/processing rate
Some issues still to be solved

- Submission of Grid jobs and workflows
  - Security?
  - Need/want to deploy a Grid client (UCC) on the target system?
  - Submit from UNICORE/X?

- Configuration of the new feature
  - Off by default on storage factory?
  - On by default on other storages?
  - Rule inheritance?

- Scalability – many users, many storages
Outlook
Questions?

- Thanks
  - Jedrzej Rybicky for discussions on this topic