GAG Meeting

April, 17 2003

Testing the HEPCAL use cases

J.J. Blaising, F. Harris, Andrea Sciabà

Introduction

- The LHC experiments have defined their common use cases for production in HEPCAL
- It is desirable to have a tool to test the available implementations of these use cases
- For example, this tool could simulate a Monte Carlo production
- Eventually, this tool should try to implement as many HEPCAL use cases as possible

D8.3 analysis of 43 HEPCAL use cases for EDG 1.4.3

Classification : ‘not implemented’ / ‘partially implemented’ / ‘mostly implemented’ / ‘implemented’

- General (authorisation, login, browse resources) — 4 use cases
  - Fully 2
  - Almost 1
  - Partially 1
- Data Management (metadata and data operations) — 19 use cases
  - Fully 4
  - Almost 3
  - Partially 3
  - No 9 (includes metadata, virtual data…)

- Job Management (submission, control, monitoring, error, resource estimation, job splitting…) 15 use cases
  - Fully 4
  - Almost 3
  - Partially 3
  - No 5 (includes job catalogues, job splitting)
- VO and software environment Management (resource reservation, user rights, conditions publishing, software publishing…) 5 use cases
  - Almost 1
  - Partially 1
  - No 3 (VO Resource handling,condition publishing)
- Global fraction in implemented use cases: 60%

Use cases not satisfied by EDG 1.4

Use case	EDG 2.0	EDG 2.1
DS Metadata Update	LFN attributes (strings)	general attributes
DS Metadata Access	LFN attributes (strings)	general attributes
Virtual Dataset Declaration	out of scope	out of scope
Virtual Dataset Materialisation	out of scope	out of scope
User-Defined Catalogue Creation	no	unknown
Dataset Access Cost Evaluation	get_accessCost()	get_accessCost()
Data Retrieval from Remote Dataset	no	maybe
Dataset Verification	no	unsure
Browse Experiment Database	out of scope	out of scope
Job Catalogue Update	user defined attributes stored in L&B	user defined attributes stored in L&B
Job Catalogue Query	user defined attributes stored in L&B	user defined attributes stored in L&B
Error Recovery for Failed Production Jobs	checkpointing	checkpointing
Job Splitting	no	DAG support
Analysis I	out of scope	out of scope
Conditions Publishing	no	no
VO Wide Resource Reservation	no	maybe
VO Wide Resource Allocation to Users	no	maybe

(Thanks to E. Laure)

This test suite

- Main purpose: to simulate a Monte Carlo production on an EDG-based Grid enviroment
- Inspired by the test suite written by J. J. Blaising
- Implemented in Perl
  - requires Getopt::Long, Pod::Usage, Net::LDAP, Term::Readkey
- Tested on the EDG application testbed and on LCG-0
- Supports both LDAP RC and RLS
- Available from the EDG CVS repository
  - http://datagrid.in2p3.fr/cgi-bin/cvsweb.cgi/edg-tests/tests/Stress/

Functionalities

- Job submission
  - arbitrary job duration
  - arbitrary output data size
  - match making with input data (for two-stage productions)
- Data management
  - output retrieval with sandbox
  - supported protocols: GridFTP
  - copy and registration of input and output files
  - replication and deletion of files

Covered use cases

Covered

- Grid login
- Dataset registration
- Dataset upload
- Dataset access
- Dataset replication (not tested)
- Dataset deletion
- Dataset browsing
- Job submission
- Job output access or retrieval
- Steer job submission (not tested)
- Production job
- Job monitoring
- Simulation job

Not covered

- Obtain Grid authorization
- Ask for revocation of Grid authorization
- Browse Grid resources
- Dataset transfer to non-Grid storage
- Dataset replica upload to the Grid
- Physical dataset instance deletion
- Catalogue deletion
- Job control
- Job resource estimation
- Job environment modification
- Data transformation
- Software publishing
- Experiment software development for the Grid

30% of the HEPCAL use cases

50% of the HEPCAL use cases at least partially implemented

Not scriptable

"Trivial"

Barely implemented

Commands

- `Test.exe <evts> <output_size>`
  - represents the event generator
  - performs an amount of useless computations proportional to `<evts>` and creates a file with size equal to `<output_size>` bytes
- `prod_submit <cards_file>`
  - submits a bunch of jobs following the directives in the cards file
  - creates a *.db file which will contain all the information about the jobs in the “production”

Other commands

- `prod_status <db_file>`
  - updates the database with the most recent information about the jobs and prints the status and the CE of each job
- `prod_getout <db_file>`
  - retrieves the output of finished jobs
- `prod_summary <db_file>`
  - prints a summary of the production
- `prod_delete <db_file>`
  - deletes from the SE and from the replica catalog the files of the production

Card file parametres

Variable	Definition
OUTPUT_DIR	directory where to store the outputs
DATASET	dataset name
INIT_RUN	first run number
RUNS	number of runs (=jobs)
EVENTS	number of events (∝CPU time)
DATA_NAME	output file name (the run no. is appended)
INPUT_DATA	LFN of the input file (the run no. is appended)
DATA_SIZE	size of the output file
OUT_SANDBOX	YES if the output file must the returned in the sandbox
COPY2SE	YES if the output file must be copied to a SE
OUTPUT_SE	CLOSE for the close SE, otherwise the SE host name
LFN_DIR	“directory” where to store the file (part of the LFN)
REPLICATION	YES if the output file must be also replicated elsewhere
REP_SE	host name of the SE where to replicate the output file
VO	virtual organization

The rc.conf file is embedded in the card file

Output analysis

- This information is collected:
  - success/failure of the dg-job-* commands (submission, status, output retrieval)
  - success/failure of every relevant command in the job script (e.g. the replica manager commands)
  - CE name, WN name, SE name, storage directory, LFN of the output file
  - duration of the job

Present issues

- Fragile: no consistency and syntax checks on the config file
- a job can require only up to one input file
- the replication is not tested
- additional requirements in the JDL not (yet) available
- Poor documentation

EDG 2.0/2.1 expectations

- With EDG 2.0, the fraction of supported use cases should be (76.7 ± 4.7) %
- With EDG 2.1, the fraction of supported use cases should be (85 ± 8) %

Further development

- Make it compatible with EDG2/LCG-1 as soon as possible
- Add new use cases and new tests when possible
- find new manpower in LCG
- help from the loose cannons welcome (but available only until EDG ends)