Tools for Distributed Analysis in ATLAS

Johannes Elmsheuser

Ludwig-Maximilians-Universität München, Germany

27 Apr 2006/D-Grid HEPCG Workshop

Outline

1. Introduction
2. Job scheduler, GANGA
3. Interactive, PROOF, DIANE
4. Conclusions & Outlook

Outline

1. Introduction

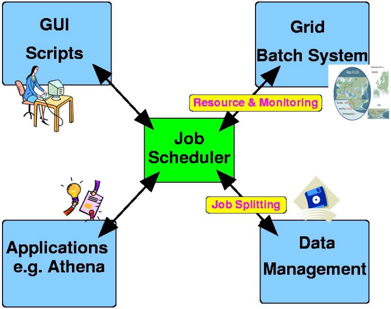
2. Job scheduler, GANGA

3. Interactive, PROOF, DIANE

4. Conclusions & Outlook

Distributed Analysis needs

- Expected LHC data volumes demand a distributed analysis
- Typical user: ∼1000 jobs, several 100 TB per analysis cycle
- Use of Grid resources
- Investigating Job-Scheduler requirements for distributed an interactive analysis



Analysis numbers for ATLAS and DØ

Item	Unit	ATLAS	DØ
Raw Data Size	MB/evt	1.6	0.25
Rec Data Size	KB/evt	~100	~10-50
Events	evt/year	2×10^9	5×10^8
2μ-Skim Size	evt		6×10^7
2μ-Skim Size	TB		1.1
Analysis on TMB	evt/s		10-30
Analysis jobs		1000-10000	100-500

In addition: lots of MC statistics/jobs

User analysis scenarios

- Analysis with fast response time and high level of user interaction
  → e.g. PROOF or DIANE
- Analysis with intermediate response time and interaction
- Analysis with long response times and low level of user interaction
  → Job scheduler (GANGA)

Outline

1. Introduction

2. Job scheduler, GANGA

3. Interactive, PROOF, DIANE

4. Conclusions & Outlook

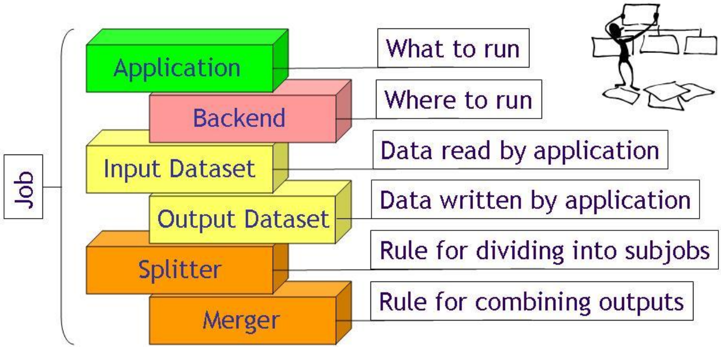
Job scheduler specification

First results from the Gap Analysis:

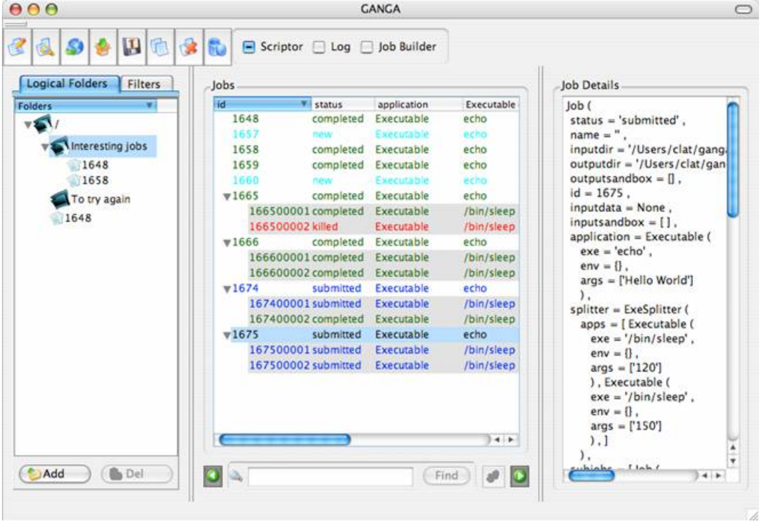
- Interface for job configuration
- Job submission interface for Grid and Batch systems
- Integration of data management
- Resource estimation
- Job monitoring
- Job error checking
- Collecting and merging of the results
- Job archive

JOB SCHEDULER EXAMPLE: GANGA

- GANGA is an easy-to-use front-end for job definition and management
- Uniform interface (Python or GUI) for local and Grid jobs
- Developed in the context of ATLAS and LHCb
  built-in support for Athena/Gaudi applications
- Written in Python, modular design



Job scheduler example: GANGA



Jobs

id	status	application	Executable
1648	completed	Executable	echo
1657	new	Executable	echo
1658	completed	Executable	echo
1659	completed	Executable	echo
1660	new	Executable	echo
▼1665	completed	Executable	echo
166500001	completed	Executable	/bin/sleep
166500002	killed	Executable	/bin/sleep
▼1666	completed	Executable	echo
166600001	completed	Executable	/bin/sleep
166600002	completed	Executable	/bin/sleep
▼1674	submitted	Executable	echo
167400001	submitted	Executable	/bin/sleep
167400002	completed	Executable	/bin/sleep
▼1675	submitted	Executable	echo
167500001	submitted	Executable	/bin/sleep
167500002	submitted	Executable	/bin/sleep

Job Details

```
Job (
 status = 'submitted' ,
 name = '' ,
 inputdir = '/Users/clat/gang
 outputdir = '/Users/clat/gan
 outputsandbox = [] ,
 id = 1675 ,
 inputdata = None ,
 inputsandbox = [ ] ,
 application = Executable (
   exe = 'echo' ,
   env = {} ,
   args = ['Hello World']
   ) ,
 splitter = ExeSplitter (
   apps = [ Executable (
     exe = '/bin/sleep' ,
     env = {} ,
     args = ['120']
     ) , Executable (
     exe = '/bin/sleep' ,
     env = {} ,
     args = ['150']
     ) , ]
   ) ,
 subjobs = [ Job (
```

http://cern.ch/ganga/

Our GANGA extensions (I)

We added (so far):

PBS back-end

- Good starting point to learn internals of GANGA
- Existing (non-Grid) back-ends: LSF, Local
- Rewrite existing LSF back-end
- Developed at GridKa
- submit generic or Athena jobs to PBS batch queue

Our GANGA extensions (II)

Athena Job Splitting back-end (still under development)

- Previous design: send 1 job processing complete dataset to a chosen back-end
- Idea of Distributed Analysis: parallelize and distribute processing job(s)
- Adapt newly implemented splitting design of GANGA for Athena jobs
- Splitting based on input-files and number of jobs
- Closely connected to experiment data management system:
  →User only provides: dataset name and number of jobs
  →Dataset: list of files with locations, production & reconstruction version, etc.
  →System takes care of file-list splitting, job configuration, etc.

Our GANGA extensions (III)

extended to Condor-G

- Motivation: LCG resource broker needs ∼10-20s/job submission →Submission time (terminal blocked): 30min for 100 jobs
- Job-Submission with Condor-G successfully used in ATLAS Production System, large fraction of MC Production on LCG is done with Condor-G type-Executor
- Bypass LCG RB with Condor-G and submit directly to a site
- Needs a running local Condor installation with some open ports
- Bulk job submission is much faster
- Future: gLite provides an improved bulk job submission compared to LCG

Still room for improvements in various areas (see next slide)

Job scheduler specification: GANGA (I)

Comparison: Gap Analysis and GANGA features

- Interface for job configuration
  ✔ Command line, scripts, GUI

- Job submission interface for Grid and Batch systems
  ✔ LCG, gLite, LSF, PBS, Condor-(G), Local

- Integration of data management
  ✘ very basic: via LCG resource broker (RB)
  ✘ ATLAS DQ2 currently being implemented
  Q: Benefit from AP 1.2 ?

- Resource estimation
  ✔ via LCG RB

Job scheduler specification: GANGA (II)

- Job monitoring

  ✔ monitors job status changes, job lists

  ✘ no continuous job progress monitoring

  Q: Benefit from AP 2.3 ?

- Job error checking

  ✘ very basic, needs more

  Q: Benefit from AP 2.2 ?

- Collecting and merging of the results

  ✘ job output retrieval, no merging

- Job archive

  ✔ job configuration available, nice resubmission

Tests & Experiences

Used GANGA for a small scale ATLAS MC production on the LCG grid:

- Adapt official transformations to GANGA environment
- Several hundred jobs for a few 10000 events
- Performs well in configuring, submitting, monitoring and output retrieval
- Issues with job submission (10-20s per job) and error handling
- 10000 events: 603 jobs only 2 failures
- 2×1000 events: 106 jobs, 50% failures due to external problems (file catalog and ATLAS database)

Further GANGA extensions

Things we'd like to contribute and need improvements:

- Athena job splitting
- ATLAS DQ2 Data Management
- Job bulk submission
- Job error handling
- Automatic Job resubmission

Outline

1. Introduction

2. Job scheduler, GANGA

3. Interactive, PROOF, DIANE

4. Conclusions & Outlook

From Distributed to Interactive Analysis

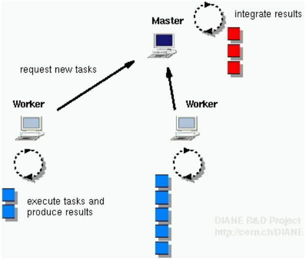
- Use GANGA to skim your data/MC
  → output: ROOT-ntuples
- Produce histograms with ROOT-ntuples

- Use PROOF to parallelize ROOT-ntuple crunching
  → Works fine on LMU Opteron cluster.
- Further alternative for parallel processing and fast response times:
  → DIANE

DIANE (I)

DIANE: Distributed Analysis Environment

Homepage: `http://cern.ch/diane/`



- Distributed framework for parallel scientific applications in master-worker mode
- Start e.g. master-process on desktop and worker processes on a cluster, batch system or grid
- Can be used with many applications: GEANT4, Athena, Image Rendering,...
- Immediate response and job communication via CORBA
- Splitting is done via input files

DIANE (II)

Performance:
1 Master, 2 worker Athena jobs with 100 AOD input files
immediate transfer of histogram and root-tuple

System	Total	Init
local, 1 P4 3 GHz	1600s	15s
local, 2 P4 3 GHz	770s	15/15s
LCG, 1 Site, Xeon 3.0 GHz	1286s	380s/500s
LCG, 2 Sites, Xeon 2.8/3.0 GHz	1384s	500s/550s

Milestones & Schedules

- Gap Analysis
  - Investigate analysis environment and concept for job scheduler and interactive analysis
  - status: advanced
- Job scheduler
  - GANGA good candidate, tests and added functionality
  - status: advanced
- Interactive Analysis
  - Successful tests with PROOF
  - status: started

Outline

1. Introduction
2. Job scheduler, GANGA
3. Interactive, PROOF, DIANE
4. Conclusions & Outlook

Conclusions & Outlook

Conclusions:

- Presented different applications for analysis use cases, with a closer look on ATLAS
- Performed Gap Analysis and Improved GANGA in several areas

Outlook:

- Further improve GANGA functionality to Gap analysis results
- Adapt to the upcoming LCG and ATLAS software upgrades
- Data Management is key for the distributed analysis - experiment specific ATLAS data management redesign in progress
- Investigating interactive analysis