Ganga - a job management and optimisation tool

A. Maier
CERN

Overview

- What is Ganga
- Ganga Architecture
- Use Case: LHCb
- Use Case: Lattice QCD
- New features

Sponsors

- Ganga is an ATLAS/LHCb joint project
- Development work supported by PPARC through GridPP and by EGEE through ARDA
- Contributions from many others, from summer students to senior researchers including the Academia Sinica

What is Ganga?

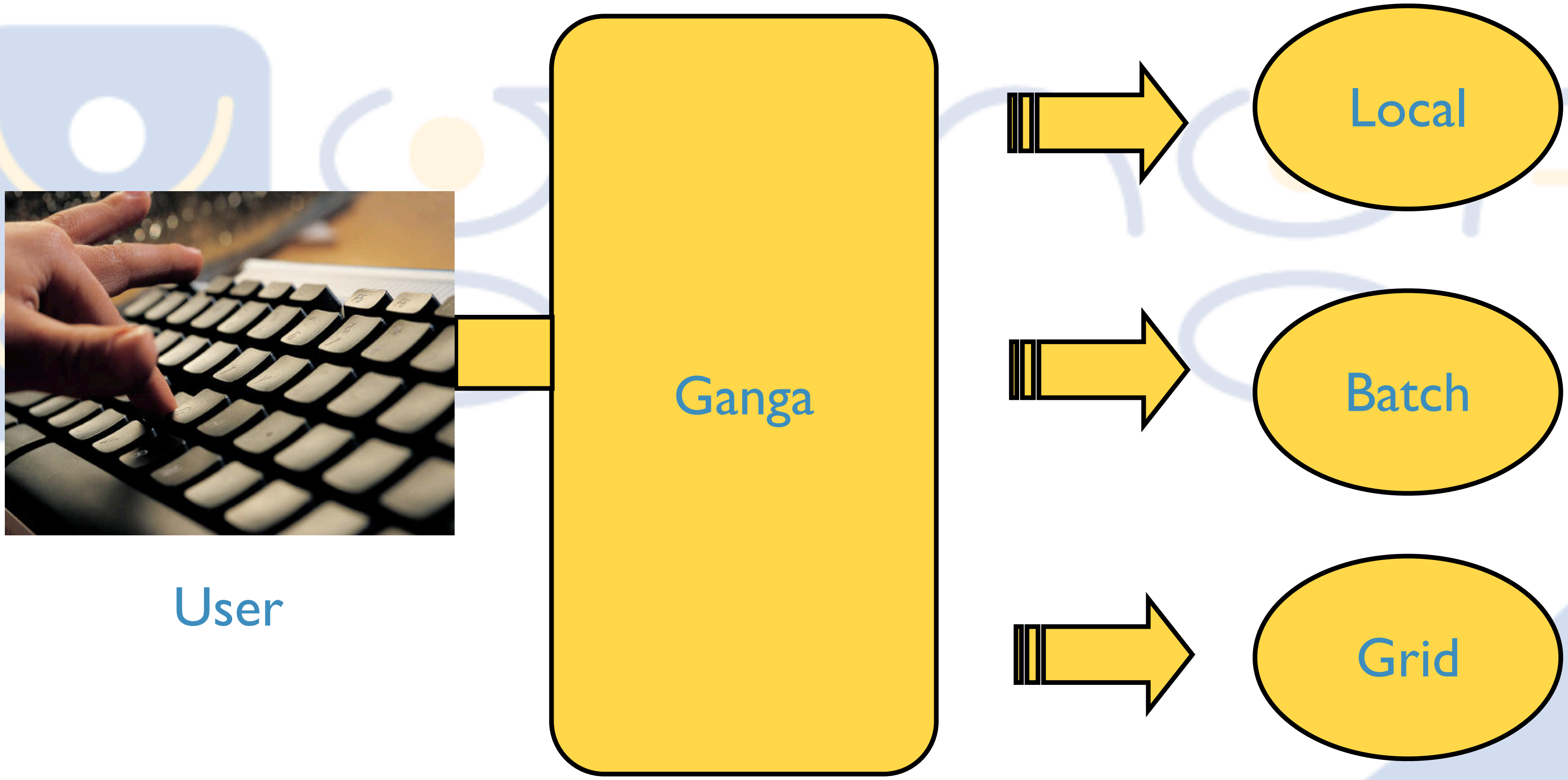
- Started of as a Atlas/LHCb project
- Ganga is an application to enable a user to
  - Configure – Prepare – Submit – Monitor

  applications to a variety of resources

Possible resources

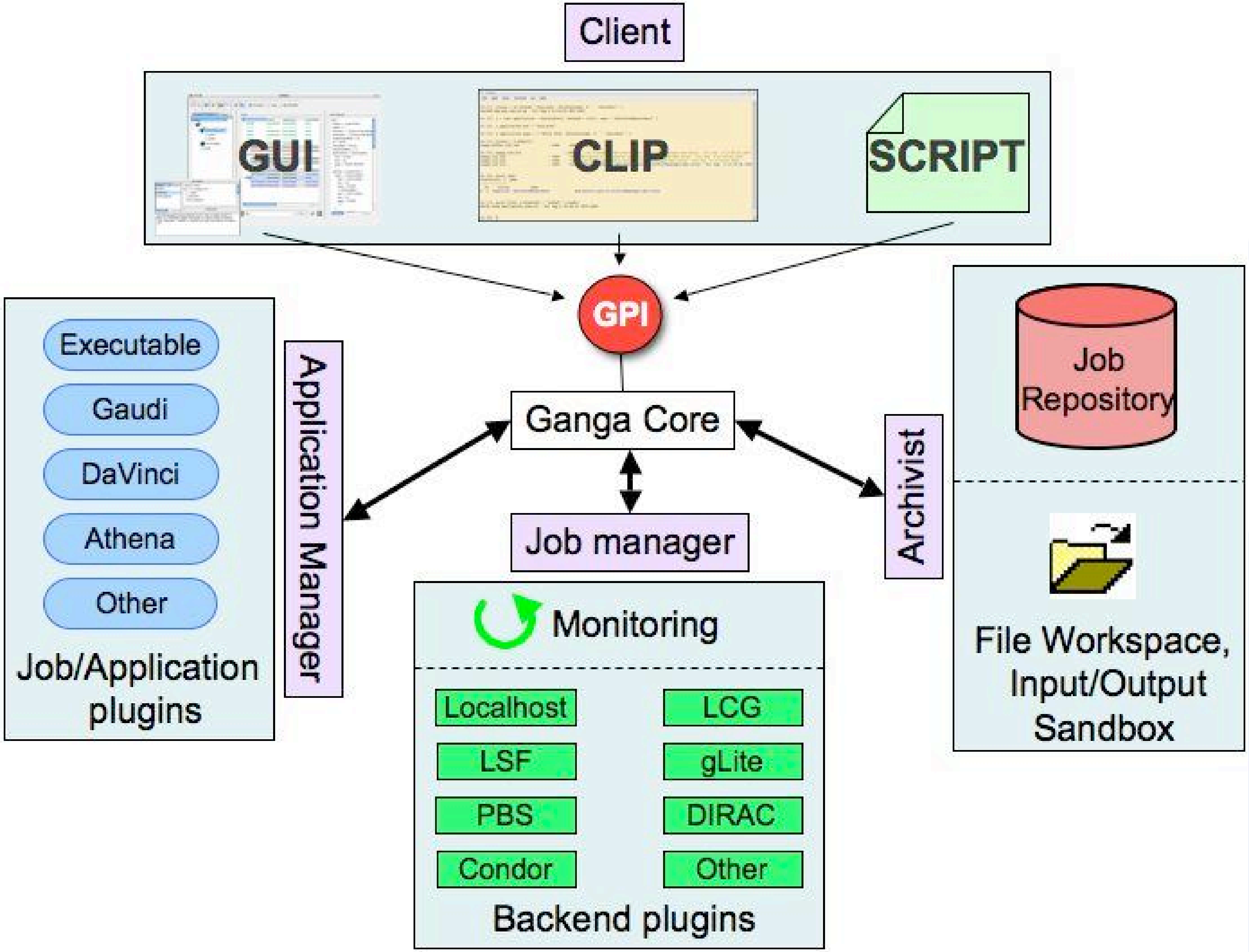
- The local machine (interactive or in background)
- Batch systems (LSF, PBS, SGE, Condor)
- Grid systems (LCG, gLite, NorduGrid)
- Workload management systems (Dirac, Panda)
- Jobs look the same whether the run locally or on the Grid

The Ganga Mantra:



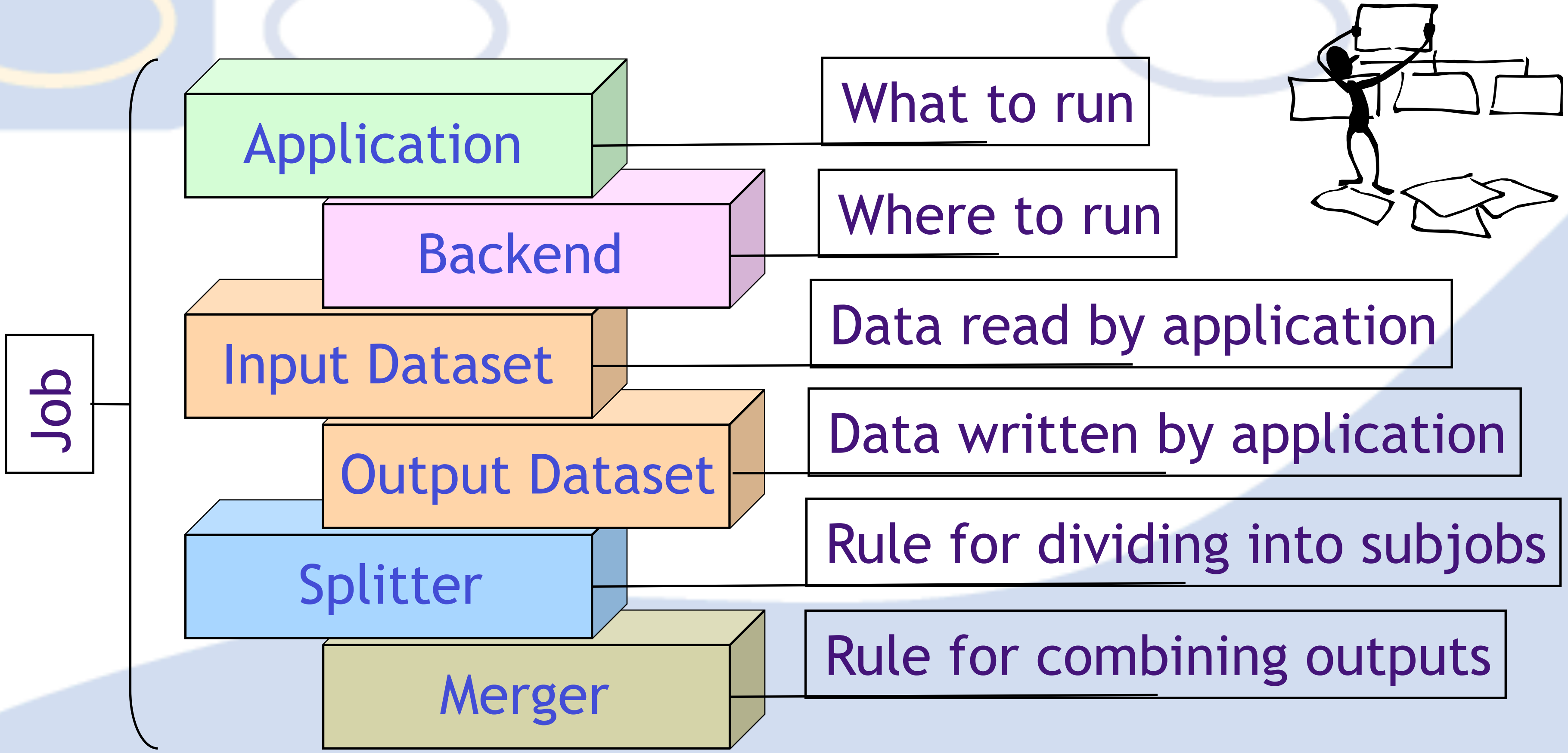
Configure once, run anywhere

Ganga Architecture



Ganga Job Object

- A job in Ganga is constructed from a set of building blocks, not all required for every job



Job

Application — What to run

Backend — Where to run

Input Dataset — Data read by application

Output Dataset — Data written by application

Splitter — Rule for dividing into subjobs

Merger — Rule for combining outputs

Job definition

- A job can be defined in Ganga starting from an instance of the Job class

- Job properties can be passed as arguments to the constructor

```
j = Job( application = Executable(), backend = LCG())
```

- Job properties and sub-properties can also be set through assignments

```
j.application.exe = “/bin/echo”

j.application.args = [ “Hello World” ]
```

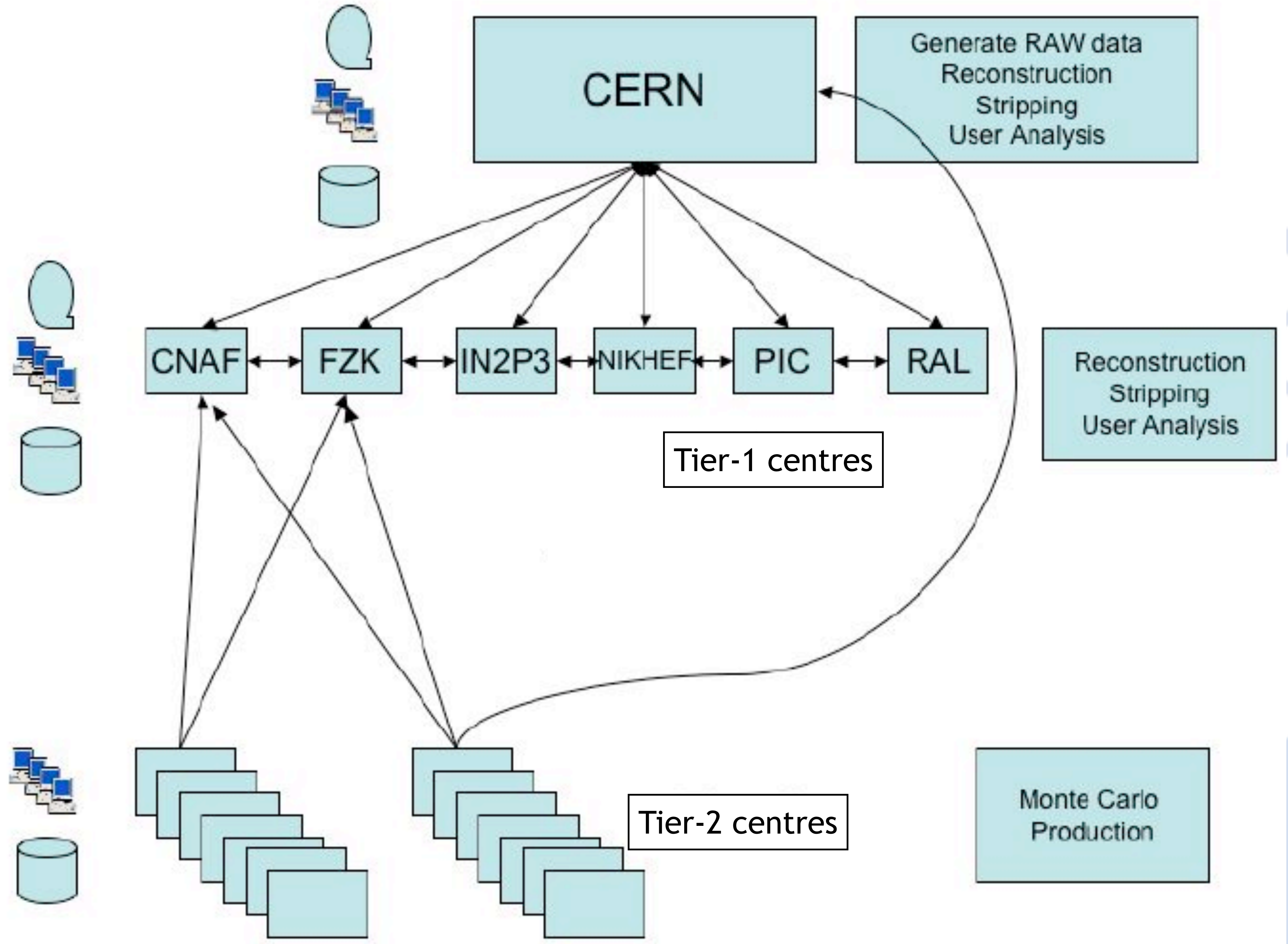
For the user, running a job interactively is no different than running on the Grid:

```
# submit 3 jobs, one local, one on batch, one to the grid

j=Job(backend=Interactive(),application.exe=’/bin/echo’)
j.application.args=[‘Hello world’]
j.submit()

j2=j.copy() # make a copy of the last job
j2.backend=LSF(queue=’8nm’) # submit to LSF
j2.submit()

j3=j.copy(),
j3.backend=LCG() # run on the Grid
j3.submit()
```

Use Case: LHCb

- Customised application plugin eases job creation
- Incremental development of analysis from
  - First test on local machine
  - Intermediate sample analysed on batch
  - Full sample run using Dirac backend

LHCb computing model

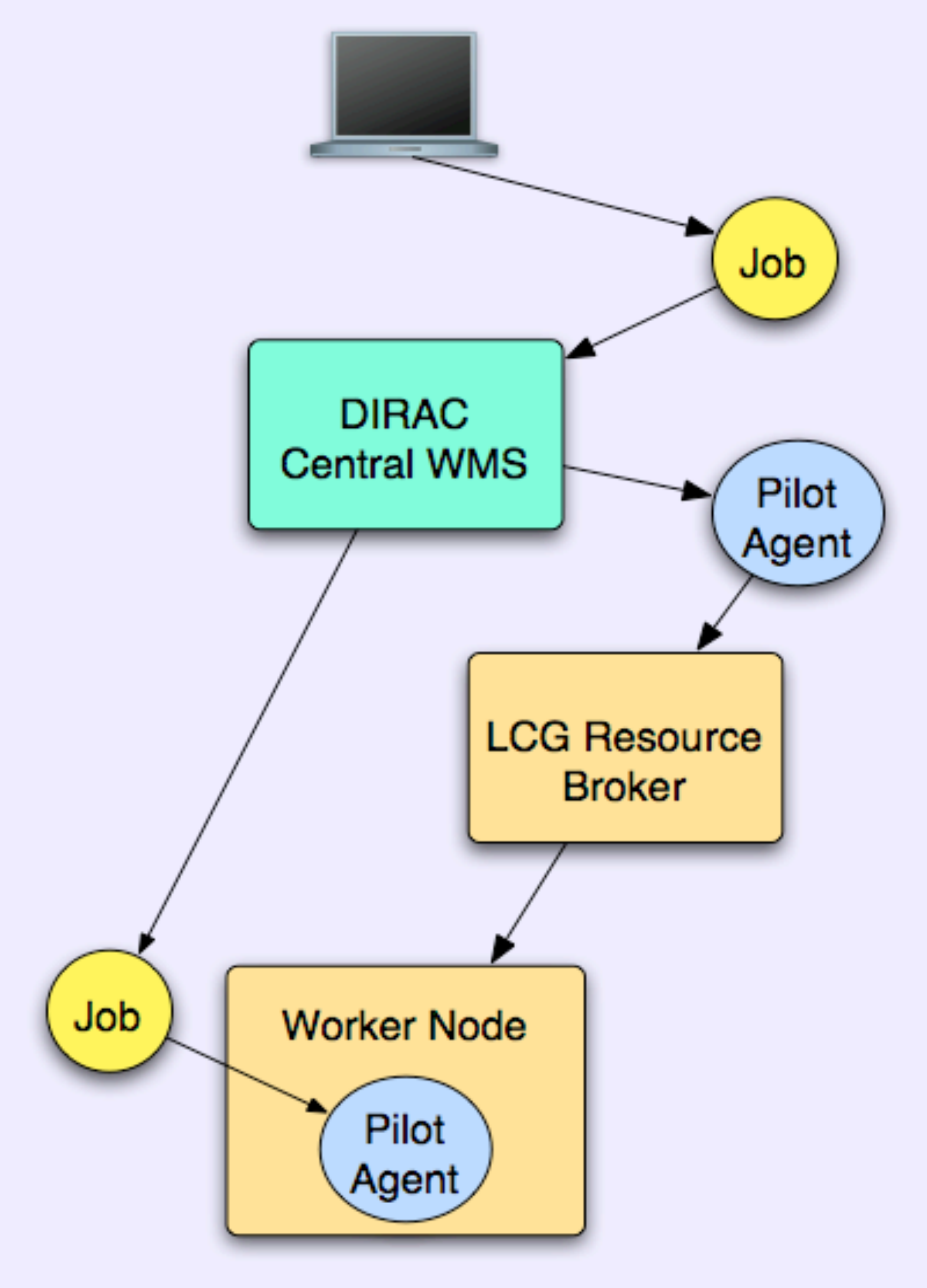


Baseline solution: analysis at Tier-1 centres

Analysis at Tier-2 centres not in baseline solution, but not ruled out

Grid Access for Analysis

- Analysis jobs: No direct submission to LCG
- Instead:
  - Submission to the DIRAC WMS
- Advantages:
  - Provide transparent access to the LFC file catalogue for reading and writing data
  - Allow LHCb to set priorities and or restrictions for analysis jobs
- More see Stuart Paterson's talk



LHCb Analysis Job

- Gaudi based applications:

```
In [3]: dv = DaVinci(version='v12r12')
In [4]: print dv
DaVinci {
 version = 'v12r12' ,
 extraopts = None ,
 package = 'Phys' ,
 cmt_user_path = '/afs/cern.ch/user/u/uegede/cmtuser' ,
 masterpackage = None ,
 optsfile = File {
    name = ''
    }
 }
```

Specify extra option file properties appended to the options file

Specify the package you are working on

Specify the options file to be used

LHCb Analysis Job

LHCbDatasets

```
LHCbDataset (
 cache_date = 'Wed Aug 29 23:49:04 2007' ,
 files = [ LHCbDataFile (
    name = 'LFN:/lhcb/production/DC06/phys-v2-lumi2/00001889/
DST/0000/00001889_00000003_5.dst' ,
    replicas = ['IN2P3-disk', 'CERN-disk']
    ) , ]
 )
```

Easy splitting of jobs

```
j.splitter=SplitByFiles(filePerJob=3)
```

Real Analysis Example

- Total of ~4M events
- One job split into 460 sub-jobs.
- Submitted lunchtime, almost all completed by the end of the day
- Failure rate <4%. Some of these failures fake failures. OK for the user

- Repeatable experience:
  - “260 sub-jobs and had same experience – ran very quickly, painless – low failure rate”
  - “A few days later, 100 sub-jobs – just one failure (and that was my fault…)”
  - “Sub-jobs a godsend”

Summary LHCb

- Ganga has plugins for LHCb applications
- LHCb analysis on the Grid is performed via the DIRAC backend
- Ganga configures, prepares LHCb applictions
- Ganga discovers in and outputs automatically
- Allows simple and flexible splitting of large jobs

Use Case: Lattice QCD

- An application to determine conditions for phase transition of Quark-Gluon plasma
- Uses a 21 space-time lattices as inputs
- Output file of each iteration becomes input file for the next iteration
- Result improves simply by increasing number of iterations.

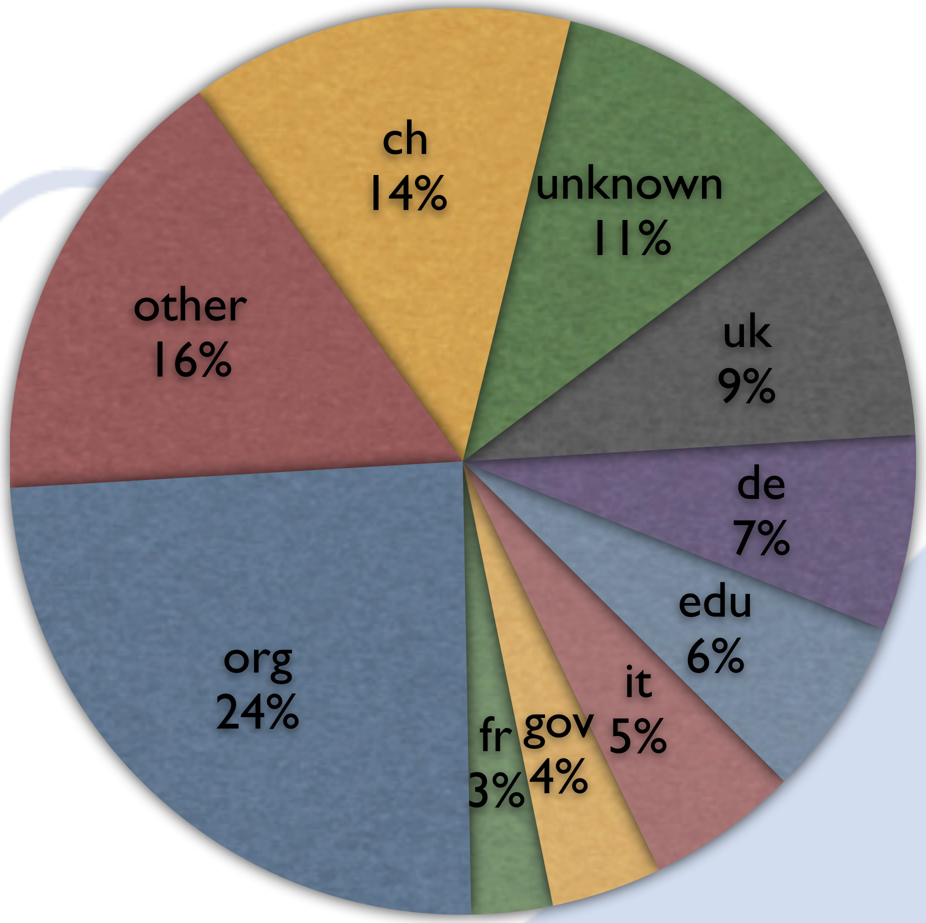
Job Execution

- Standard Ganga used, no customisation
- Each job split into 21 subjobs using standard built-in splitting feature in Ganga
- Job results sent back in 1 hour intervals
- Job runs until queue is exhausted

Lattice QCD: Results

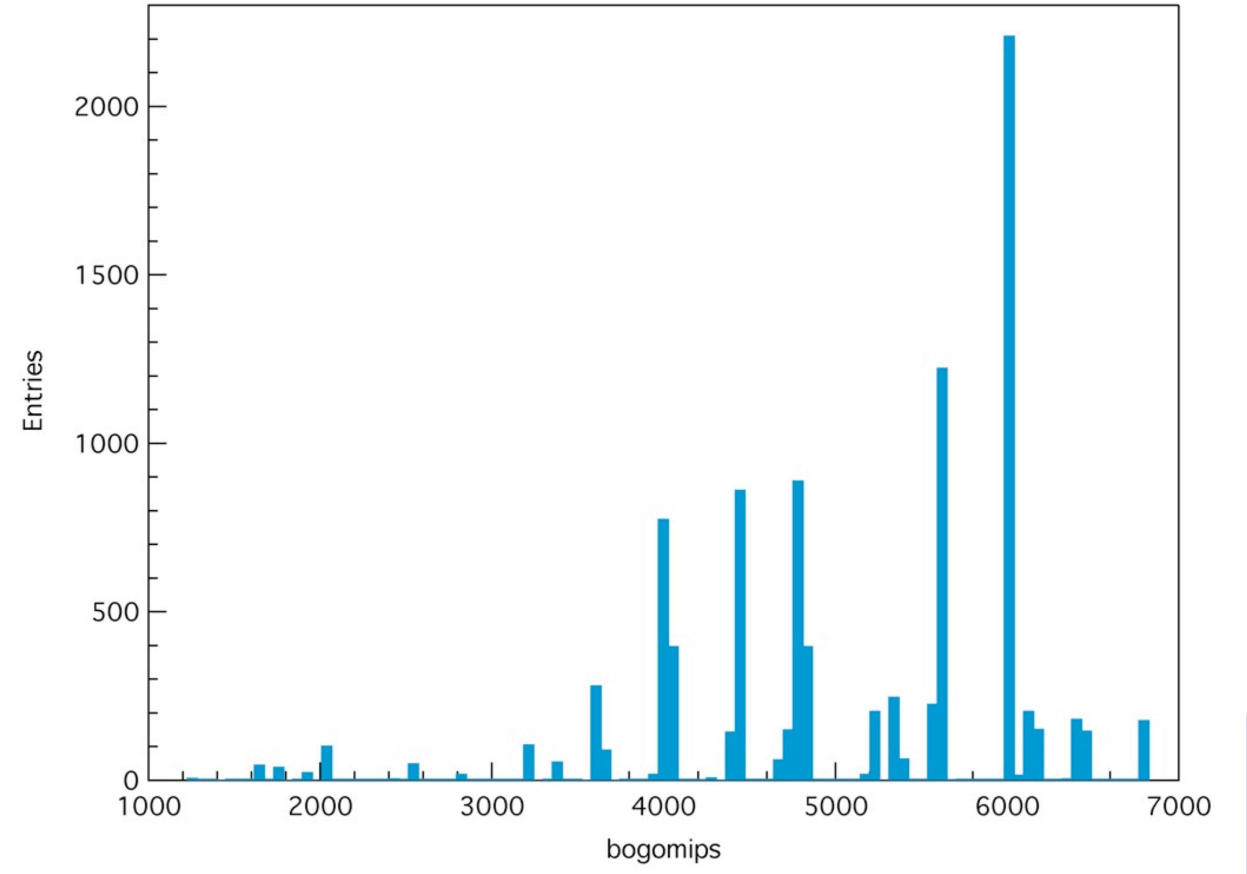
- Jobs sent by 4 people on 4 different VOs and LSF
- Jobs killed after 1 week
- >9500 CPUS used
- ~ 1.5M results sent back
- Jobs ran on > 50 sites

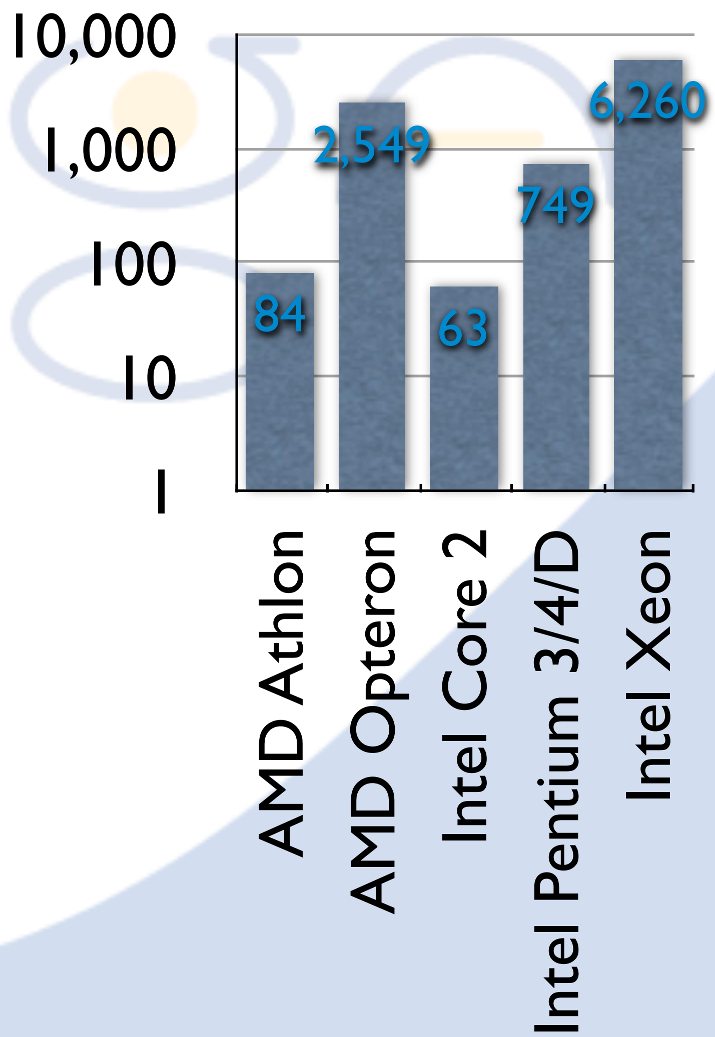
Main top level domains



Lattice QCD: Results

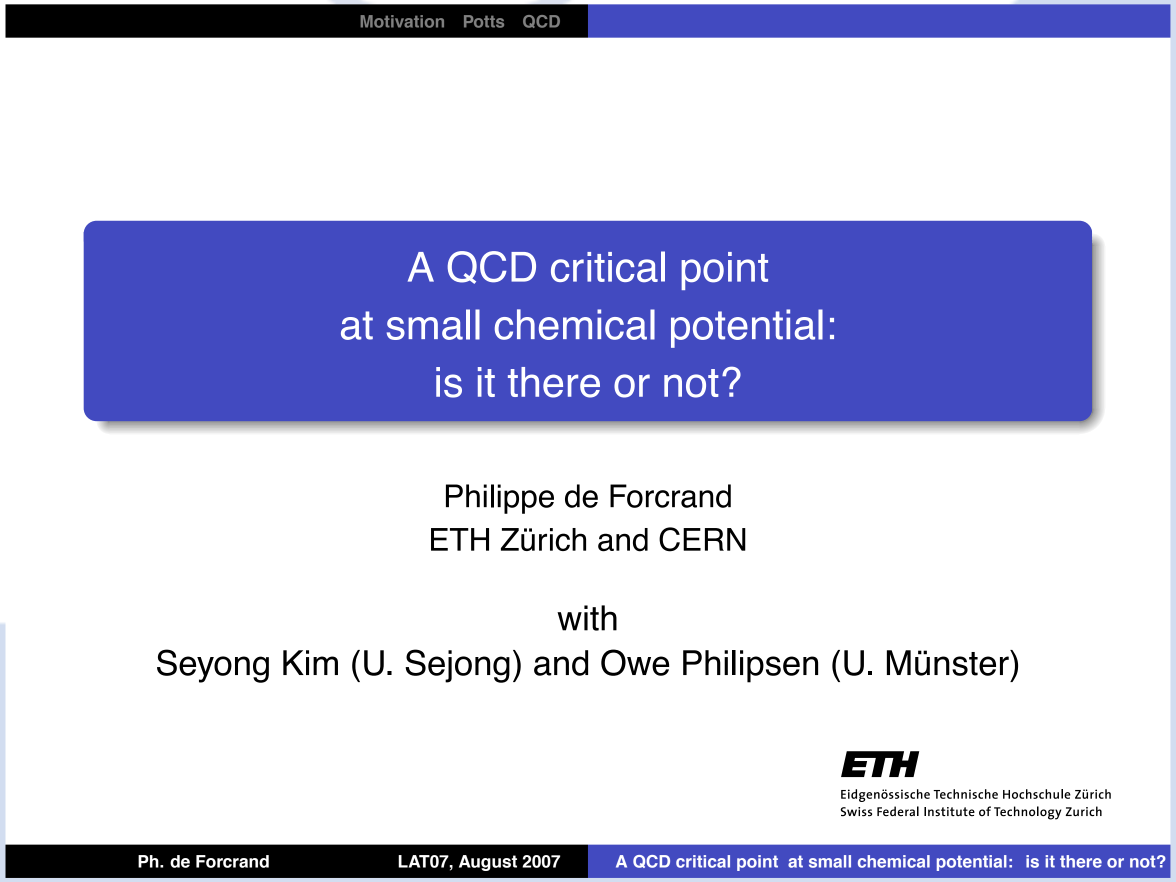
- Majority of jobs ran on Intel Xeons
- And not on Pentium IIIs





Lattice QCD: Results

- In 1 week:
- 30 CPU years of simulation results
- Partial (~20 % of total) already used in conferences



Motivation Potts QCD

A QCD critical point
at small chemical potential:
is it there or not?

Philippe de Forcrand
ETH Zürich and CERN

with
Seyong Kim (U. Sejong) and Owe Philipsen (U. Münster)

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

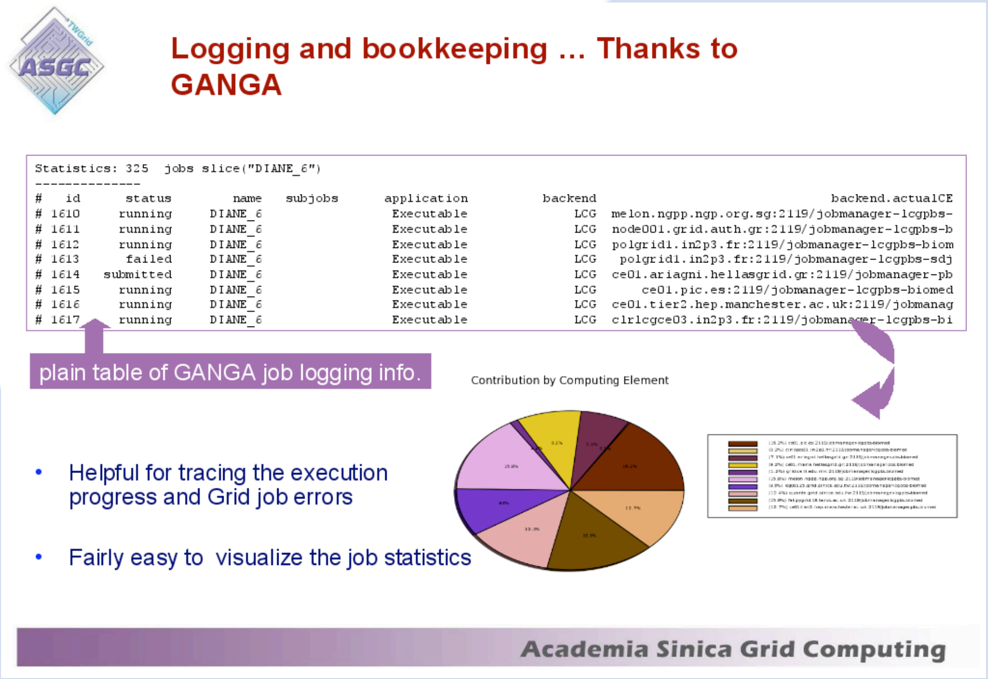
Ph. de Forcrand LAT07, August 2007 A QCD critical point at small chemical potential: is it there or not?

Users

- Ganga has now more than 840 users.
- Probably more than 10% of LCG users
- Mainly used by Atlas - LHCb (the initiators), but also ~20% non HEP use
- More on statistics see J. Elmsheuser talk

Other users of Ganga

- In conjunction with Diane (http://cern.ch/diane):
  - Gridproduction testbed: Tests the functionality and availability of grid sites
  - Geant 4 simulation: new versions are tested against result of earlier version
  - ITU: used to aid the negotiation of new digital TV frequencies
  - Biomed: Search for bird flu cure



Logging and bookkeeping … Thanks to GANGA

```
Statistics: 325  jobs slice("DIANE_6")
--------------
#   id      status      name   subjobs     application       backend                           backend.actualCE
# 1610     running   DIANE_6                Executable           LCG  melon.ngpp.ngp.org.sg:2119/jobmanager-lcgpbs-
# 1611     running   DIANE_6                Executable           LCG  node001.grid.auth.gr:2119/jobmanager-lcgpbs-b
# 1612     running   DIANE_6                Executable           LCG  polgrid1.in2p3.fr:2119/jobmanager-lcgpbs-biom
# 1613      failed   DIANE_6                Executable           LCG   polgrid1.in2p3.fr:2119/jobmanager-lcgpbs-sdj
# 1614   submitted   DIANE_6                Executable           LCG  ce01.ariagni.hellasgrid.gr:2119/jobmanager-pb
# 1615     running   DIANE_6                Executable           LCG      ce01.pic.es:2119/jobmanager-lcgpbs-biomed
# 1616     running   DIANE_6                Executable           LCG  ce01.tier2.hep.manchester.ac.uk:2119/jobmanag
# 1617     running   DIANE_6                Executable           LCG  clrlcgce03.in2p3.fr:2119/jobmanager-lcgpbs-bi
```

plain table of GANGA job logging info.

Contribution by Computing Element

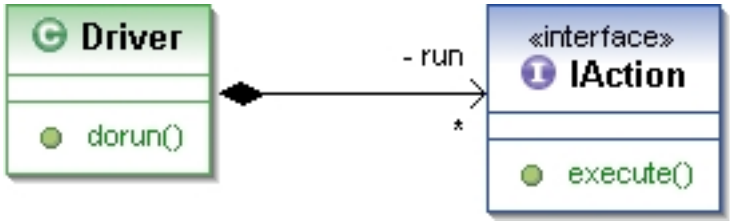
- Helpful for tracing the execution progress and Grid job errors
- Fairly easy to visualize the job statistics

Academia Sinica Grid Computing

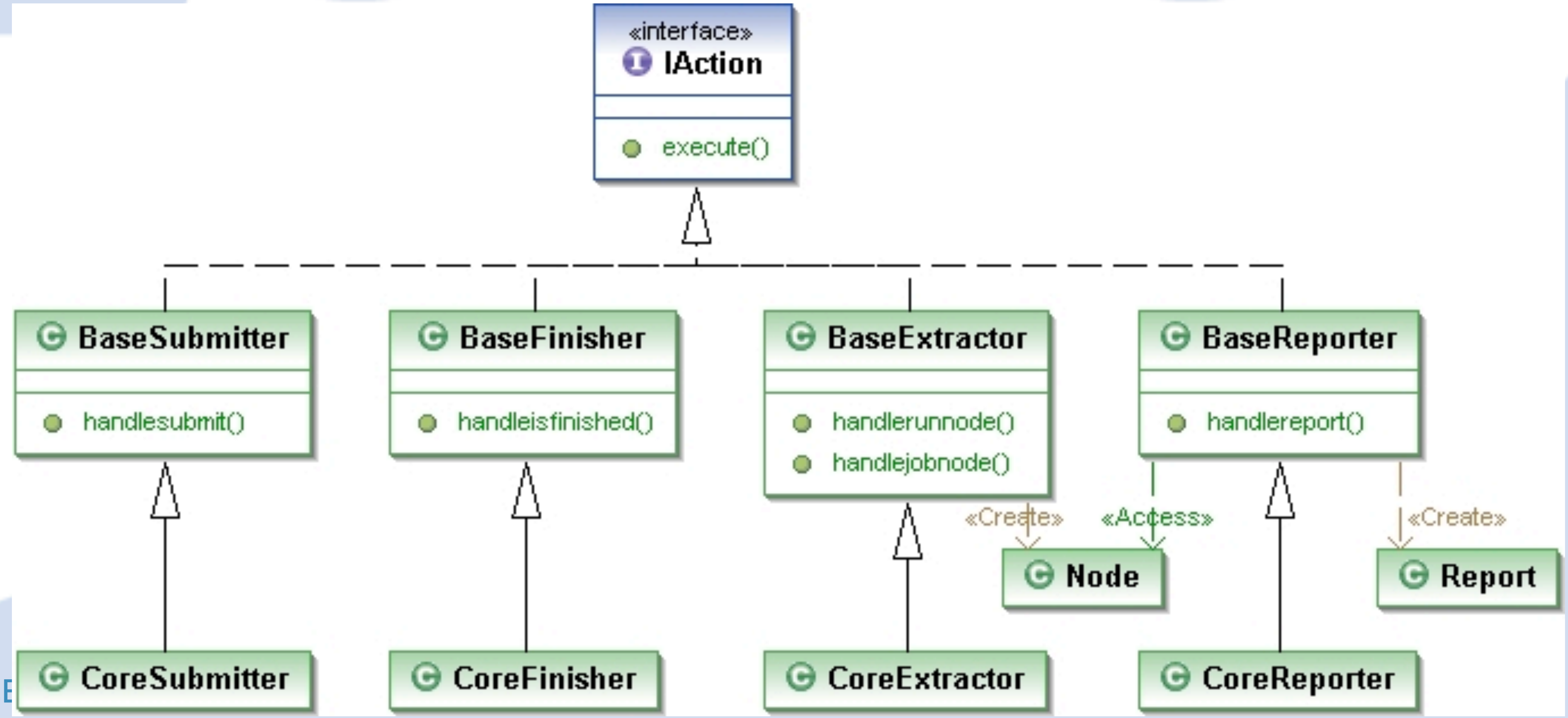
Ganga Robot

- Run a user-defined list of actions within the context of a Ganga session,
- Actions are defined by implementations of an action interface.
- Suited to performing complex tasks involving:
  - Submitting jobs to the grid
  - Extracting data about the jobs and the grid environment
  - Reporting statistics on the extracted data.
- Typical use-case: periodically monitor the end-to-end execution of a set of standard jobs submitted via Ganga.

- The framework consists of a Driver class containing a list of IAction implementations.



- Abstract base action implementations provides a basis for implementing submit / extract / report actions



Windows Port

- Experimental Windows port exists.
- Goal is to allow to submit jobs from Windows to both Windows or Linux
- Streamline code and avoid platform dependent code
- Opens up Ganga to users working from Windows desktop

Windows Port

- Backends for which Windows ports exist can be ported
  - Submission to Local and Condor working
  - LSF possible (but not done yet)
  - Dirac port also possible
- gLite depends on a proper Windows port (Currently only cygwin)

Conclusions

- Ganga is an easy to use system for job submission to a variety of resources - Grid and non-Grid
- Ganga fosters incremental users analysis: From tests on the local machine to full scale runs on the Grid
- Ganga has > 700 users making it one of the popular ways to submit jobs to the Grid
- Ganga is used by both HEP and non-HEP applications
- Ganga can be customised to take advantage of the users application via application plugins, still even without specialised plugins, Ganga is useful from the start