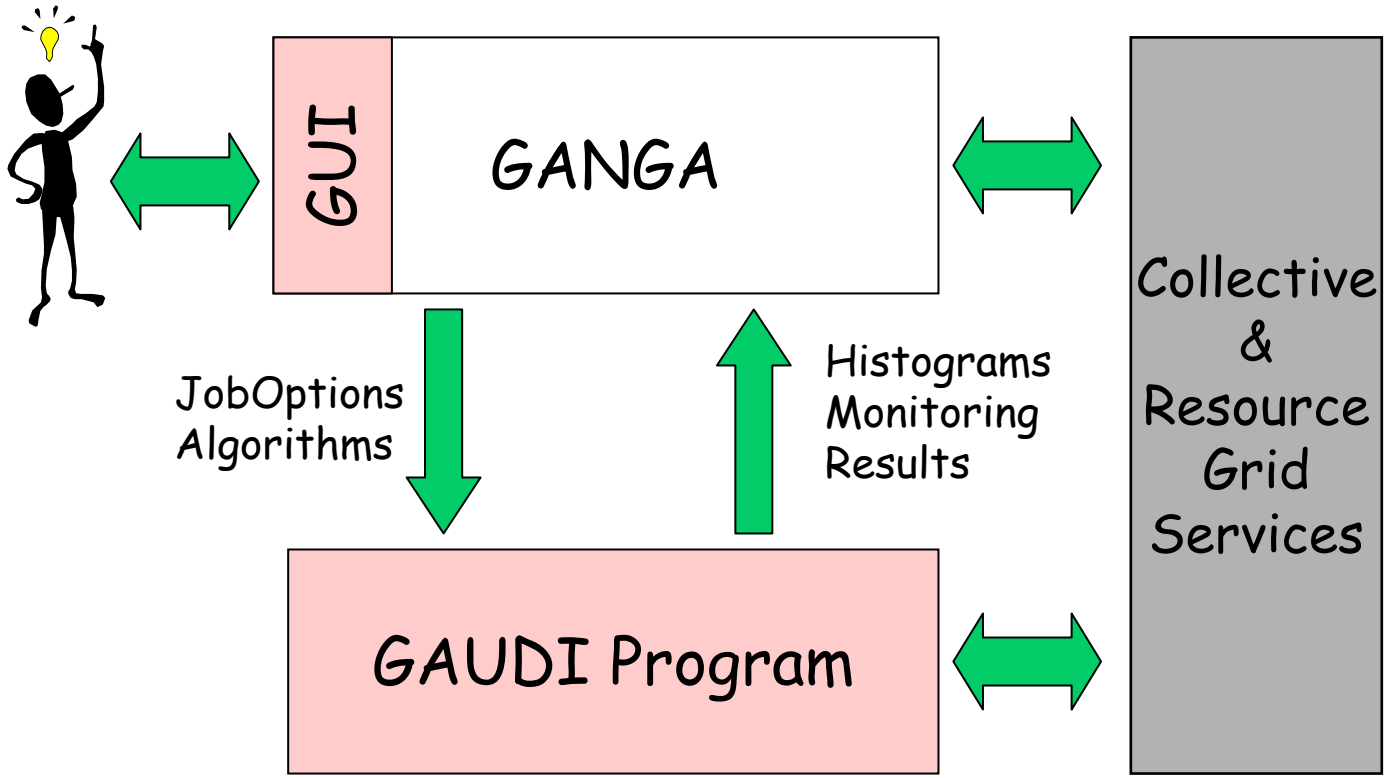
Ganga 4

Andrew Maier
ARDA/LHCb

Contents

- Ganga: Gaudi And Grid Alliance
- Ganga Overview
- Ganga and ARDA
- Ganga 3 – the current release
- Ganga 4 – Introduction
- Ganga Public Interface
- Internal Architecture
  - Client
  - Application Manager
  - Job Manager
  - Remote Registry
- Plans

GANGA: Gaudi ANd Grid Alliance



Ganga - Overview

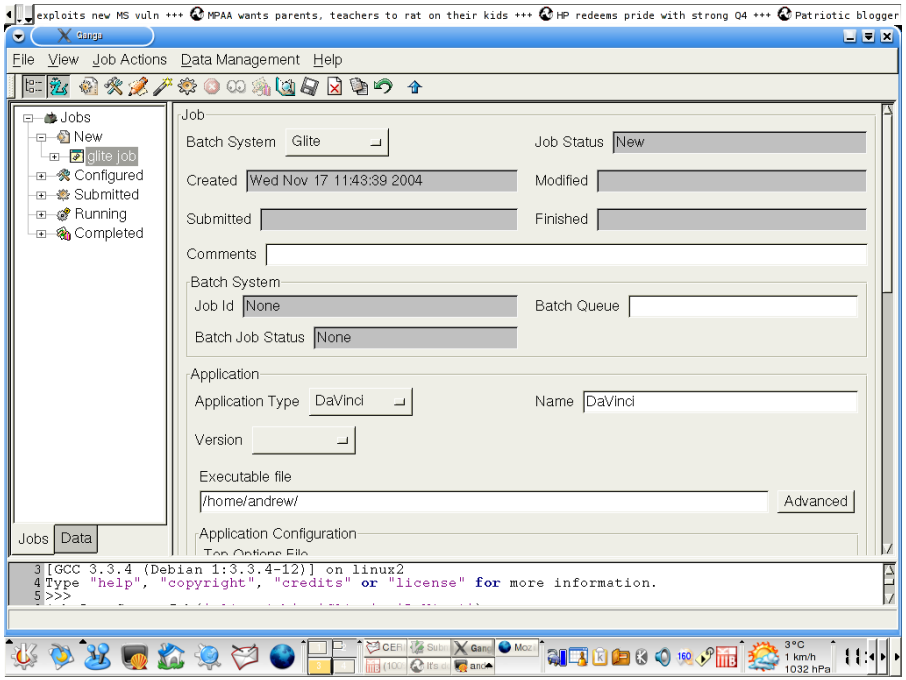
- Ganga (Gaudi / Athena and Grid Alliance) is an interface to the Grid that is being developed jointly by ATLAS and LHCb.
- It will be optimised for the configuration and management of jobs that use the Gaudi / Athena Framework common to the two experiments.
- It shields from the user the complexity and the changes within the middleware
- Ganga provides a uniform high-level interface to the different low-level solutions ranging from
  - Specification of input
  - Choice of submission backend
  - Data to the retrieval and post-processing of the output.

Ganga and ARDA

- Ganga is the ARDA E2E prototype chosen by LHCb
- The ARDA LHCb team are involved as
  - Release manager for Ganga
  - Developers and designers
- So far with involvement of ARDA, 3 minor and 1 major releases have been achieved
- Interfaces to the gLite prototype system have been developed

Ganga 3 - The current release

- The current release of Ganga (version 3) is mainly a GUI application



- New in Ganga 3 is the availability of a command line interface

Ganga 3 - The current release

- What you can do with Ganga 3 Scripting:
  - validation scripts for LHCb software itself
    - ganga -t
    - >>> execfile("examples/advanced_LHCb/loop.py")
  - job submission scripts
    - >>> execfile("examples/davinci.py")

```
j=Job()
f = "/afs/cern.ch/[...]/DAVINCI_v12r7/Phys/DaVinciEff/v3r2/options/DVEffBs2PhiEtac.opts"
j.application = DaVinci(optionsfile=f, version="v12r7")
j.backend=LSF()
j.backend.queue="8nm"
j.submit()
```

- Ganga 3 now has all the functionality needed
- Limitations have been discovered with the current model
  - -> Redesign of Ganga using Ganga3 as a prototype

Ganga 4 - Introduction

- While the current status presented is the result of an intensive 8 week discussion:
  - A lot of the information shown is work in progress
  - Certain aspects are not yet fully defined and agreed upon
  - Changes are likely to happen
  - Up to date information can be found on the Ganga web page: http://cern.ch/ganga

Ganga 4 - Introduction

- Ganga 4 is the next major release of Ganga
- Ganga 3 has reached its limitations
- Ganga 4 is based on the Ganga scripting interface, the GUI will follow later
- Ganga 4 will overcome some of the shortcomings of the current release
  - By using a modular design
  - Applying functional decomposition of modules
  - Having extended functionality

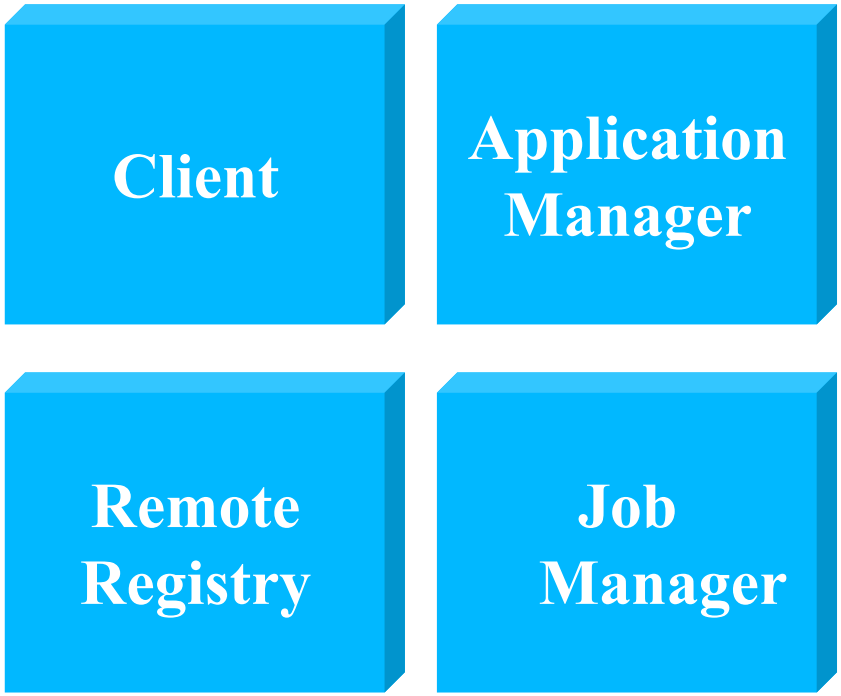
Ganga 4 - Introduction

- Ganga 4 – Software Reengineering
  - Starting point:
    - recode Ganga 4 from scratch where needed
    - add GUI later
  - GPI – Ganga Public Interface
    - 95% based on Ganga3 CLIP
  - Ganga Core:
    - fast and reliable
    - self-contained (NO GUI)
    - clear architecture and module dependency
      - allow remote registry for sharing jobs (near future)
      - allow Client-Server split (far future)
  - Functionality
    - reuse (at least at the prototype level) all existing Ganga 3 handlers

Ganga Public Interface

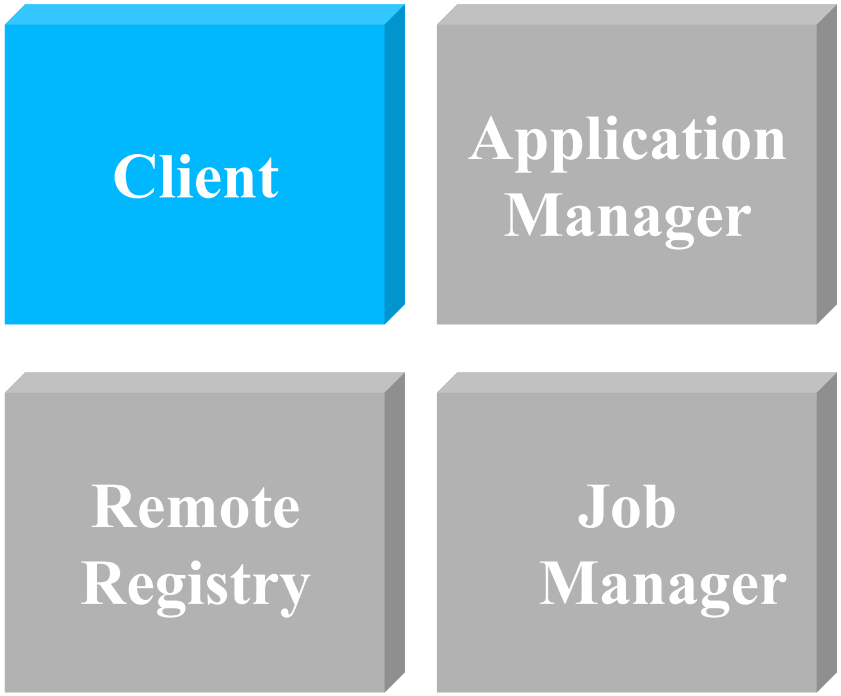
- The Ganga Public Interface (GPI) represents the logical model of Ganga.
- It describes the objects (jobs, applications, backends…) and their properties.
- For the user the GPI is accessible (to a large extent) via the CLIP (Command Line Interface in Python)
- For the developer of application- or backend plug-ins the GPI is the effective Ganga API
- It is extensible in the sense that each application or backend may define their own set of properties visible to the user.

Internal architecture



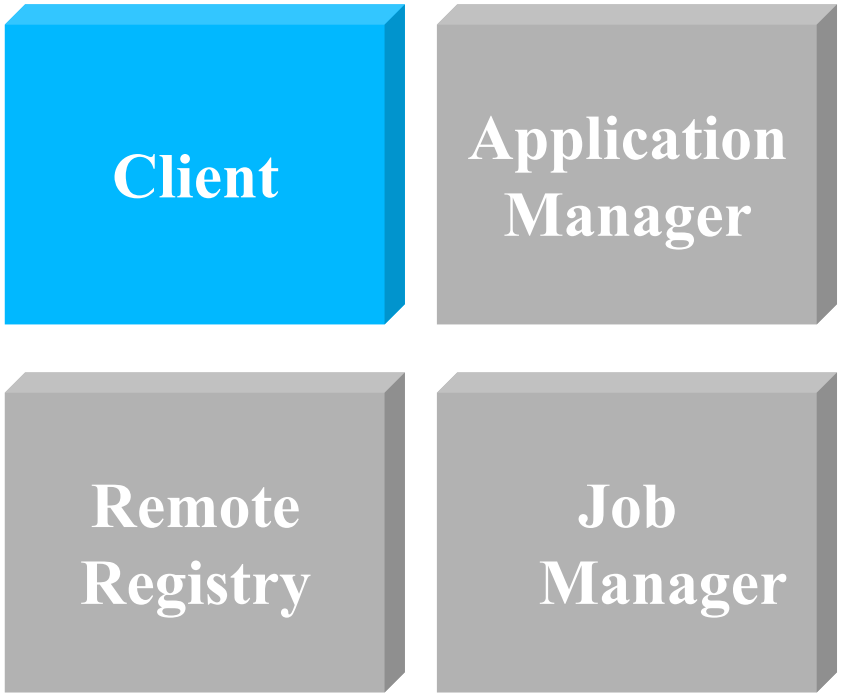
- Ganga 4 is decomposed into 4 functional components
- These components also describe the components in a distributed model.
- Strategy: Design each component so that it could be a separate service.
- But allow to combine two or more components into a single service

Client



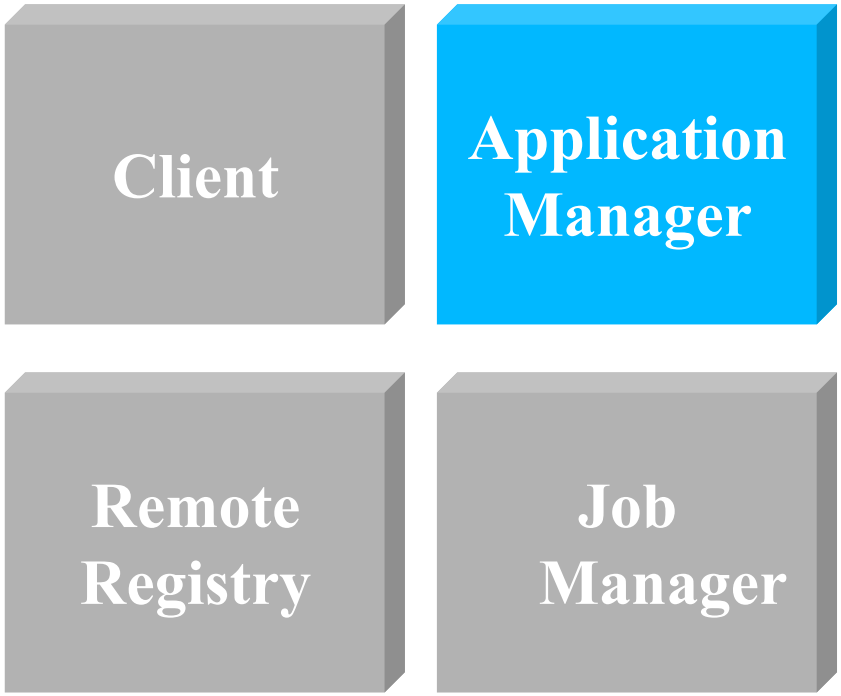
- Runs the Ganga interface (CLIP, GPI, GUI)
- The user interacts exclusively through the client
- With the client, the user creates, modifies, submits and monitors jobs
- Job configuration is kept in a registry which can be local (within the client) or remote.

Client



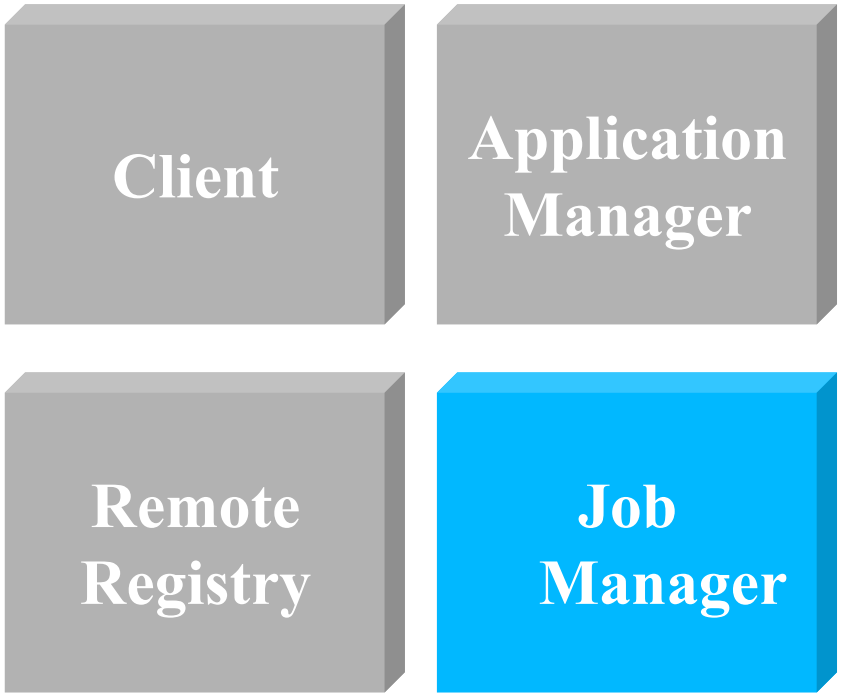
- The client should be a thin client (pure python)
- The client can be a command line client or a GUI
- The client interacts with the application manager to configure applications
- It submits and monitors jobs via the job manager
- It keeps state by storing persistent information in the registry

Application Manager



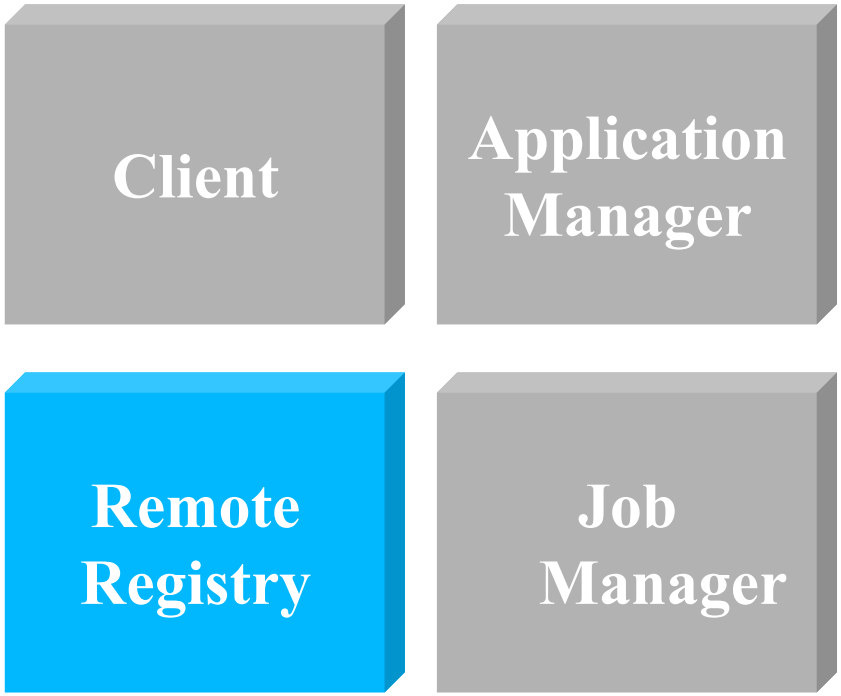
- Prepares and configures the application
- Compiles user code
- Sets-up the necessary environment
- Provides information to the client on available applications, versions, platforms, etc.

Job Manager



- Submits the configured job to the submission backend
- A submission handler submits a job to a backend
  - Creates the starter script and the JDL
  - performs the monitoring
- The application runtime handler
  - prepares the application dependent wrapper script, depending on the backend.
  - E.g., DIRAC knows how to run LHCb applications with a different setup as LSF.

Remote Registry



- Keeps track of jobs
- Is a “passive” data store, typically using a database backend
- Keeps a roaming profile of the user jobs
- Keeps track of the job status
- May cache the output of a job in a local SE.
- This component is the most likely to be run on a separate host.

Ganga 4 - Plans

- Ganga 4 Reengineering
  - Goal: by mid-April: reproduce full Ganga 3 functionality (NO GUI)
  - Planned functionality (-> order of implementation):
    - Applications: Generic Executable -> Gaudi (DaVinci, Brunel, Boole,...)
    - Backends: Local -> LSF -> DIRAC -> Glite
    - Registry: Local -> Remote
  - Design help: quick series of mini-prototypes to explore new areas
    - e.g. remote registry tests (~ create/update10 jobs/s) using ARDA MD
  - General strategy
    - code for now but design for the future
      - start with all-in-one-Client but gradually put more functionality on the Server side
    - limit the number of binary dependencies
    - ability to import Ganga GPI into another python application