K. Harrison
CERN, 12th June 2003

RELEASE OF GANGA

- Basics and organisation
- What Ganga should do tomorrow
- Ganga design
- What Ganga will do today
- Next steps

GANGA BASICS

- Ganga is an acronym for Gaudi/Athena and Grid Alliance
- Short-term aims:
  - ⇒ Deal with configuring and running Gaudi-based applications
  - ⇒ Deal with submitting and monitoring jobs to/on distributed (Grid) and local batch systems
- Longer-term aim:
  - ⇒ Develop Gaudi services for use in Grid enviroment (enable querying of replica catalogues, enable publication of information for Grid monitoring, etc.)
- Ganga is being developed as an ATLAS/LHCb common project, with support in UK from GridPP
  - ⇒ Good possibilities for contributing to LCG Physicist Interface (PI)

PROJECT ORGANISATION

- Current main contributors to Ganga are:
  - ⇒ Developers: K.Harrison, W.Lavrijsen, A.Soroko, C.L.Tan
  - ⇒ Technical direction: P.Mato, C.E.Tull
  - ⇒ GriPP coordination: N.Brook (handing over soon to G.Patrick), R.W.L.Jones
- Ganga-related information regularly updated on web site: http://ganga.web.cern.ch/ganga
- A mailing list has been active since November 2002: project-ganga@cern.ch
- Usually have telephone meeting at least once every two weeks: details of times are placed on web site, and are circulated to mailing list
- Presentations of ganga status and plans given at various other meetings of ATLAS, LHCb and GridPP

WHAT GANGA SHOULD DO TOMORROW

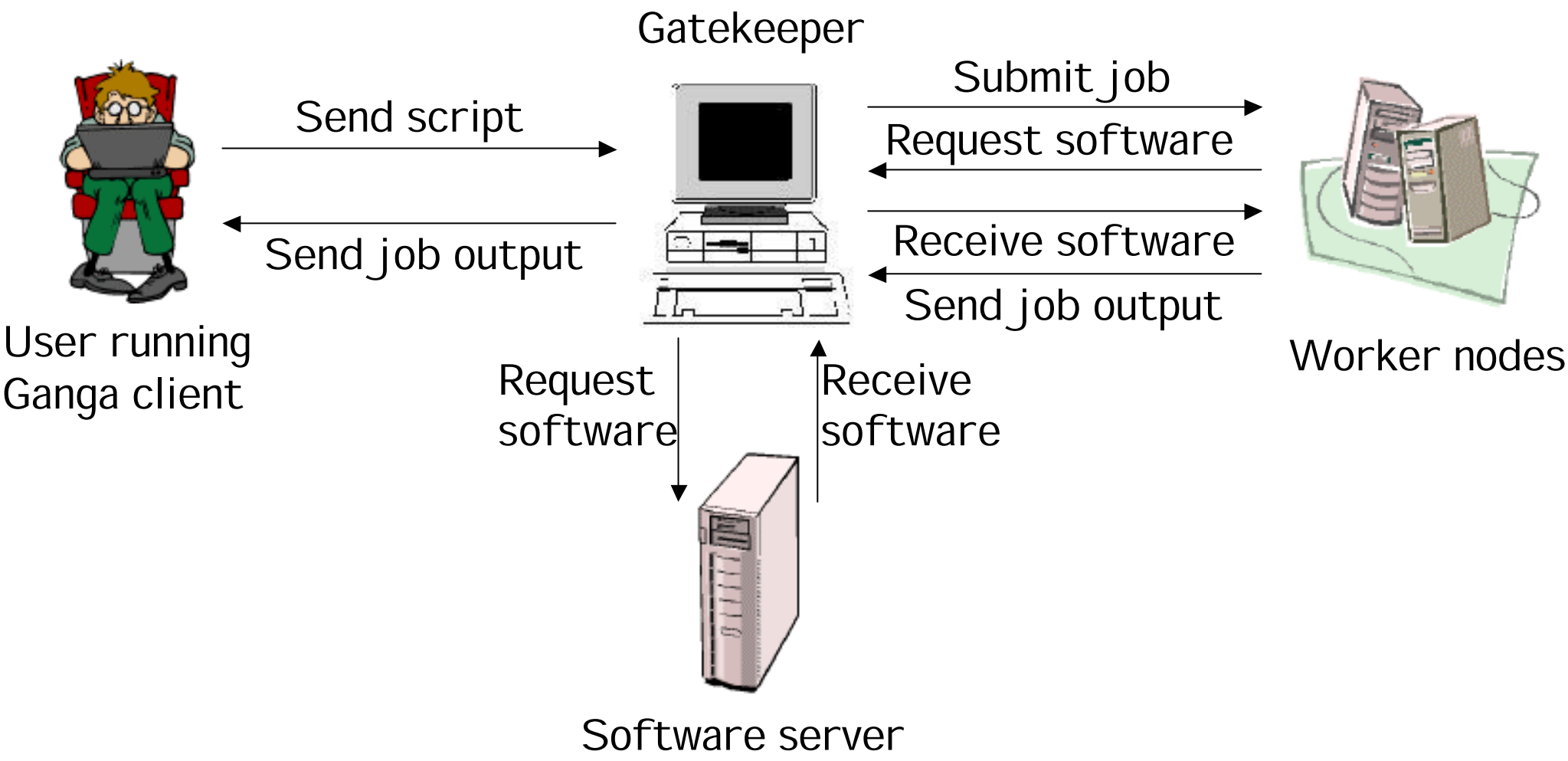
Short-term plans: June-September 2003
Longer-term plans: October 2003 onwards

1) Simplify users' lives by providing a single interface for working with all Gaudi-based (offline) applications
   - Short term: graphical user interface (GUI) and command-line interface (CLI) for working with analysis jobs (in particular, DaVinci jobs)
     Incorporate features from ATLAS Athena Startup Kit, ASK (W.Lavrijsen)
   - Longer term: allow for running of production-type jobs, integrating with existing production systems of LHCb and ATLAS: DIRAC (A.Tsaregorodtsev et al.) and AtCom (L.Goossens et al.)

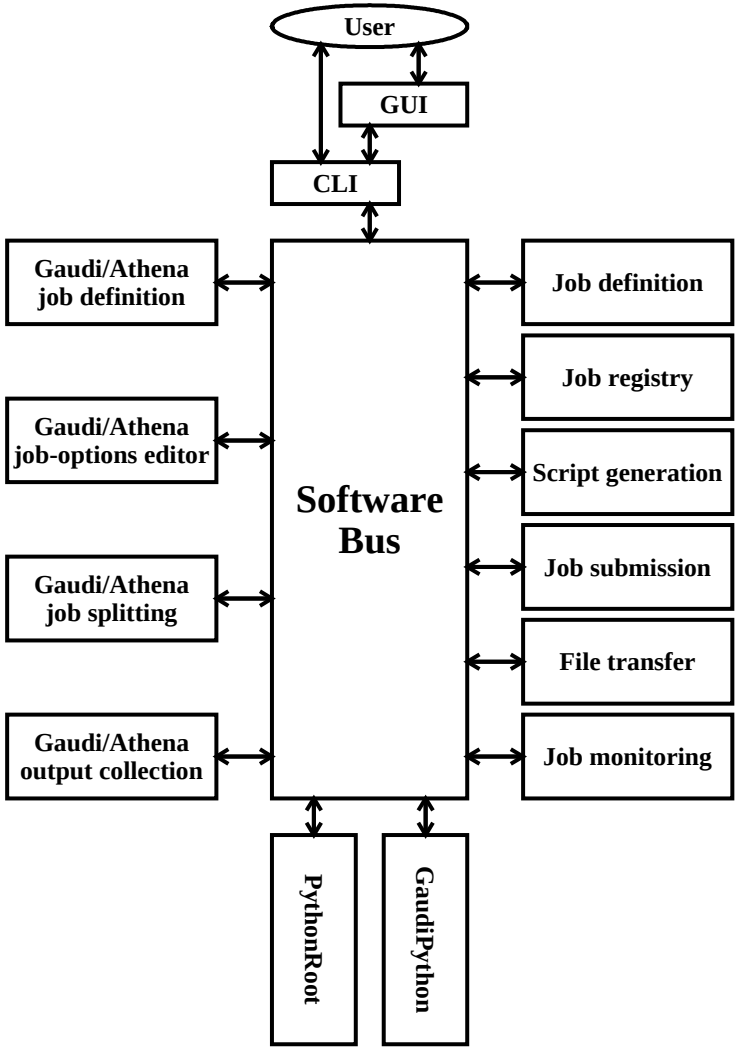
2) Make workflows/data transformations easy to define, store and instantiate, and supply templates for common use cases
   - Short term: Ganga will provide a simple workflow-definition mechanism of its own
   - Adopt more sophisticated workflow definition procedure, for example based on tools developed in context of DIRAC (G.Kuznetsov et al.), or based on Chimera (R.Gardner et al.)
3) Help with application configuration by providing a job-options editor
   - Short term: allow access to, and modification of, all job options, with possibilities for choosing options for a particular algorithm or user favorites
   - Longer term: give guidance on meaningful values (with input from algorithm developers)

4) Provide a simple, flexible procedure for splitting and closing jobs
   - Short term: introduce splitting/cloning procedure, deal with common use cases, take care of merging of outputs where appropriate/possible
   - Longer term: dependent on user feedback
5) Help users keep track of what they've done
   - Short term: provide catalogue of jobs and their status, and allow access to settings for each
   - Longer term: dependent on user feedback
6) Perform job monitoring tasks on local and distruted batch systems
   - Short term: pull information from jobs, allowing automatic updates of status and user-initiated queries
   - Longer term: move to system where jobs push information to a user-specified location; integrate with NetLogger, to have detailed information on progress of jobs on the Grid

7) Allow for user mobility
   - Short term: provide a single procedure for submitting jobs to different types of batch systems (EDG, LSF, PBS, other), with the batch system accessible from the machine where Ganga is run
   - Longer term: allow user to submit jobs from any machine with Ganga running, to batch queues on any machine (Gatekeeper) where user has an account or is in the Grid mapfile; take care of software installation at remote nodes, for example building on procedure used in DIRAC or using pacman (S.Youssef)
8) Other things, to be determined by user requests, but should consider possibilities for
   - web-portal interface
   - interactive analysis (based on ROOT)
   - data-management services


Gatekeeper
Send script
Send job output
Submit job
Request software
Receive software
Send job output
User running
Ganga client
Worker nodes
Request
software
Receive
software
Software server

DESIGN DETAILS



- User has access to functionality of Ganga components both through GUI, and through CLI, layered one over the other above the software bus
- Components used by Ganga can be divided into three categories:
  - $\Rightarrow$ Ganga components of general applicability (to right in diagram)
  - $\Rightarrow$ Ganga components providing specialised functionality (to left in diagram)
- External components (at bottom in diagram)

GANGA COMPONENTS OF GENERAL APPLICABILITY

- Components potentially have uses outside ATLAS and LHCb
  - ⇒ Could be of interest for LCG PI project and other eScience applications
- Core component provides classes for job definition, where a job is characterised in terms of: name, workflow, required resources, status
  - ⇒ Workflow is represented as a sequence of elements (executables, parameters, input/output files, etc.) for which associated actions are implicitely defined
  - ⇒ Required resources are specified using a generic syntax

- Other components perform operations on, for, or using job objects
  - $\Rightarrow$ Job-registry component allows for storage and recovery of job information, and allows for job objects to be serialised
  - $\Rightarrow$ Scrip-generation component translates a job's workflow into the set of instructions to be executed when the job is run
  - $\Rightarrow$ Job-submission component submits workflow script to target batch system, creating JDL (job-description language) file if necessary and translating resource requests as required
  - $\Rightarrow$ File-transfer component handles transfer between sites of input and output files, adding appropriate commands to workflow script at submission time
  - $\Rightarrow$ Job-monitoring component performs queries of job status

GANGA COMPONENTS PROVIDING SPECIALISED FUNCTIONALITY FOR LHCB AND ATLAS

- Components incorporate knowledge of the Gaudi/Athena framework
- Component for Gaudi/Athena job definition adds classes for workflow elements not dealt with by general-purpose job-definition component, for example applications packaged using CMT; component also provides workflow templates covering common tasks
- Other components provide for job-option editing, job splitting, and output collection

EXTERNAL COMPONENTS

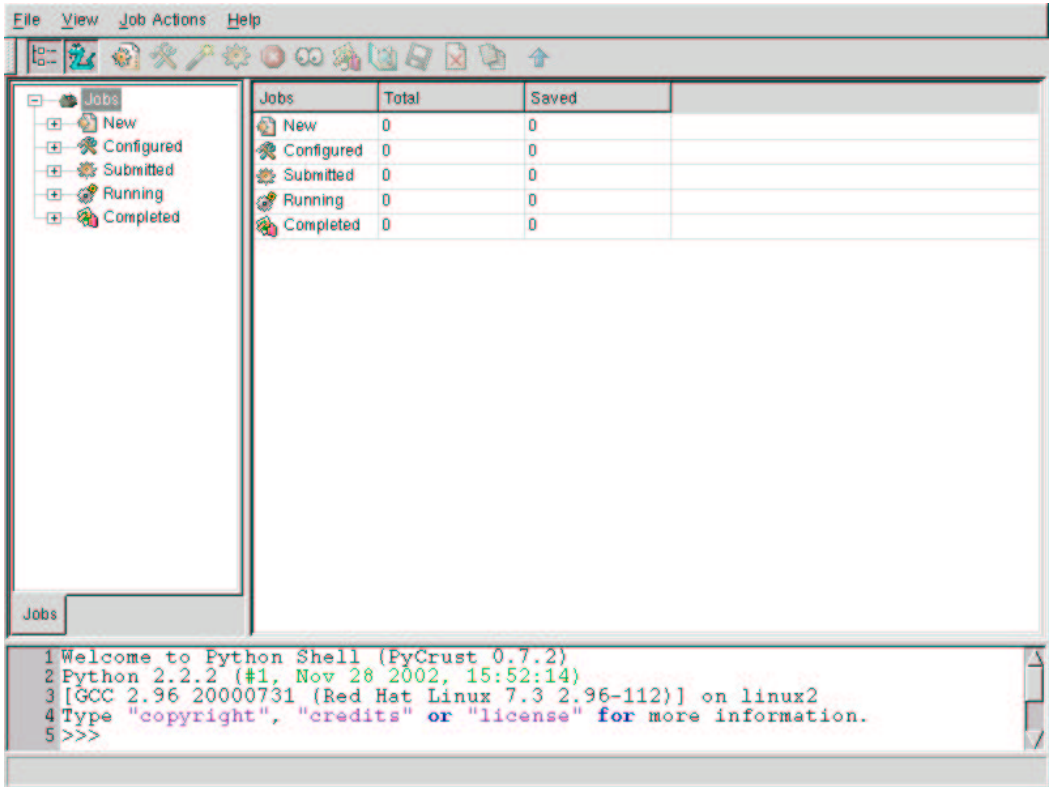
- Additional functionality obtained using components developed outside of Ganga:
  - Modules of python standard library
  - Non-python components for which appropriate interface has been written, for example Gaudi framework itself and ROOT, BoostPython

WHAT GANGA WILL DO TODAY (OR VERY SOON)

- Code for Ganga v1r0 is in Gaudi CVS repository
- Want to carry out a few more checks and do some repackaging, then release during week 16th-20th June
- Ganga v1r0 includes:
  - GUI
  - Command-line access to underlying tools (but not user oriented)
  - Job-options editor (so far set up only for ATLAS fast simulation)
  - Submission of some types of jobs, including DaVinci jobs, to different batch systems (LSF, PBS, EDG)
  - Mechanism for splitting/cloning jobs
  - Job catalogue
  - Monitoring (made more system friendly since March software week)

- Items on which work is in progress, but which aren't ready for v1r0, include:
  - Software bus that adds to functionality of python interpreter
  - User-oriented CLI
  - Pure client submission
  - Enhancement of features already present in v1r0 (generalised job-options editor, treatment of more types of job, etc.)
- Ganga v1r1 scheduled for release during week beginning 28th July, and should include some of above

GUI (A.Soroko)



- GUI developed using wxPython
- User presented with main window, job tree and python prompt
- Some simplifications made in response to comments at March software week

UNDERLYING TOOLS AND CLI (K.Harrison)

- Underlying tools currently use the following breakdown:
  - Workflow: a series of WorkSteps, defining the sum of the actions to be performed when the job is run
  - WorkStep: a set of elements providing all information necessary to run one instance of an executable
  - WorkStep/Workflow elements: Command, InputFile, OutputFile, CMTPackage (produces a library), CMTApplication (has an executable associated with it)
    $\Rightarrow$ other elements to follow

- Underlying tools can be accessed from Ganga prompt to submit DaVinci job:

```
from GangaComponent import *
daVinciSetup = GangaCommand(
"source /afs/cern.ch/lhcb/scripts/ProjectEnv.sh DaVinci v8rl")
daVinci = GangaCMTApplication(
"Phys/DaVinci","v8rl","DaVinci.exe","options/DaVinci.opts")
workStep1 = GangaWorkStep([daVinciSetup, daVinci])
workFlow = GangaWorkFlow([workStep1])
lsfJob = GangaLSFJob("daVinciTest",workFlow)
lsfJob.build()
lsfJob.submit()
```

- GUI and CLI simplify use of underlying tools
- CLI under development will reduce above, adding splitting into sub-jobs, as:
  job.create("daVinciTest","DaVinci v8r1")
  job.submit(lsf@cern.ch,20)

JOB-OPTIONS EDITOR (C.L.Tan)

- Job-options editor has been developed with hard-coded defaults appropriate to ATLAS fast simulation, but will be generalised so that defaults for any Gaudi/Athena application can be read from a file or database
- Prevents some errors (mis-spelling of options/values, incorrect syntax)
- Allows definition/manipulation of sequences and lists
- Editor is option-type aware
  - Drop-down menus for discrete choices
  - Arbitrary value entry for simple options
  - Value append for list-type options

- Preferred settings can be saved to file for subsequent reloading
- Future improvements should include:
  - Expansion of include files
  - Support for python job options
  - Display of favorite options first
  - Display of option values by algorithm

SOFTWARE BUS (W.Lavrijsen)

- Prototype software bus, PyBus, has been developed as user-level python module, with no privileges over modules
  - ⇒ Standard python modules corrently handled by PyBus
  - ⇒ PyBus components treated as modules by python interpreter
- Allows component to be loaded by logical, functional or actual name
  - ⇒ First two not necessarily unique: choose on basis of PyBus configuration, priority scheme or user input
- Performs complete unloading and reloading of components, and allows component replacement
- Allows components to register their parameters, permitting component configuration

CLONING AND SPLITTING OF JOBS (W.Lavrijsen, A.Soroko, C.E.Tull)

- Sub-jobs from cloning or splitting a Gaudi/Athena job are near copies of one another, but are distinguished by name and may have a different value for one or more of the job-option parameters
- Experimenting with generic approach, where a "splitting function" returns for all sub-jobs the job-option parameters that differ from those of the initial job
- Splitting functions for common cases should be supplied with Ganga; more specialised splitting functions can be added by the user
- In the typical DaVinci case, the splitting function should examine the list of input files associated with EventSelector.Input in the job options, and assign some group of files to each sub-job
- The Ganga job handler dispatches the sub-jobs, and stores information for each in separate directories

CONCLUSIONS

- A lot of work has been done on Ganga since March software week
- Code for Ganga v1r0 is in Gaudi CVS repository

  $\Rightarrow$ Release during week 16th-20th June, following tests and some repackaging
- Ganga v1r0 is not production quality, but is useful for giving a feel of how things should work
- Ganga team has a well-defined plan of additions and improvements, but would welcome user feedback on what is already implemented, and on priorities for the things that are missing