EUDET-Memo-2009-12

A production system for massive data processing in ILCSoft

S.J. Aplin, J. Engels, F. Gaede*

December 18, 2009

Abstract

This memo presents a new production system for massive data processing on the Grid and other large computing facilities. The system is primarily targeted at being used with programs from the ILCSoft software framework developed partly within the EUDET project. The new system is written in Python and centered around a MySQL database. It provides the functionality that is needed for the submission, monitoring and bookkeeping of many thousands of jobs on the Grid. The same database that is used for the bookkeeping of the production serves also as the basis for a data catalog that can be queried through a webinterface.

*Deutsches Elektronen-Synchrotron, Hamburg, Germany

1 Introduction

This memo presents the design and current implementation of a new production system currently being developed for massive data processing on the Grid and other large computing facilities. It is mainly targeted at being used with the ILCSoft software framework [1], which has been partly developed in the context of the EUDET [2] project. While the production system has been primarily developed for use within ILCSoft, effort has been made at the design stage to ensure that the system is sufficiently generic for use with other simulation, reconstruction and data analysis frameworks. The system allows for massive processing of ILC data involving thousands of jobs distributed over a large number of (Grid) computing facilities. It keeps track of the job progress, the success rates, possible resubmission and eventually provides the basis for a queriable data catalogue. It is based on MySQL [3], Python [4], Django [5] and Ganga [6].

An overview of ILCSoft is given in Section 2. This is followed in Section 3 by a discussion on the use of ILCSoft using Grid computing resources. This covers the approach followed by the test beam experiments and that used for the mass production of Monte-Carlo Data for the preparation of the ILD Letter of Intent [7]. It was during this mass production that a clear need was identified for a more efficient and robust production system for such tasks within ILCSoft. Section 4 describes the design and current implementation of the new production system.

2 Overview of ILCSoft

ILCSoft is a software framework originally developed for studying Monte-Carlo simulation of a full detector design for the ILC. Within the EUDET project, ILCSoft has subsequently been extended and adapted for test beam data processing and is used in particular by all three EUDET JRA's. The software framework provides the core software tools for the development of reconstruction and analysis software, and allows these to be used in conjunction with Mokka [8], a Geant4 based detector simulation application, through the provision of the LCIO [9] Event Data Model API and Gear [10], a detector geometry API. Besides the event data model and geometry description above, ILCSoft contains the application framework Marlin which uses LCIO and can be used for all tasks which involve the processing of LCIO files. It follows a modular design where every computing task is implemented as a Processor that analyses data in an LCIO Event (LCEvent) and produces additional output LCIO Collections that are added to the event.

Several reconstruction and analysis software packages which are based on the Marlin framework are included within ILCSoft. These include both higher level reconstruction algorithms such as the Pandora particle flow algorithm [11] and the LCFI vertexing package [12], as well as complete test beam reconstruction applications such as MarlinTPC [13] and EUTelescope [14]. The software used within the CALICE collaboration [15] is also built upon the software contained within ILCSoft.

Using the same core software framework for the EUDET JRA test beam experiments as that which is used for the large scale detector optimisation studies within the ILD detector concept study, provides synergies in both of these two related areas of study. Such synergy is readily evident not only in the greater number of users contributing to the improved quality of the software overall, but also in the evolution of both the Event Data Model API [17] and the geometry description API [16].

More information on ILCSoft can be found at http://ilcsoft.desy.de.

3 Use of ILCSoft with Grid Computing

Considering that the ILC community has yet to form official collaborations and the present absence of a host lab, the utilisation of distributed computing resources such as those created for the LHC experiments at CERN, using the so called Grid Computing model, presents the community with otherwise unprecedented access to large scale computing resources.

3.1 Grid use by Test Beam Experiments

The test-beam experiments within the EUDET JRA3 work-package are undertaken at various locations around the world and the data needs to be available globally, independent of the experimental site. Here, Grid resources were identified as the ideal tools to meet this goal [15]. At the same time the Grid is used for simulation and re-processing of the data. Following data taking at the test beam the raw data is processed entirely on the Grid. This allows for parallel processing of several runs, thereby reducing the turn-around time for making the data available to users for analysis. This is also advantageous to the test beam operators, as it allows them to supplement their online monitoring with fully processed data during actual data taking runs. Processed data is typically available two hours after the end of a given run.

Within the context of the EUDET JRA1 work-package the EUTelescope software, being part of ILCSoft, has been installed on the Experiment Software Area of all the Computing Elements accepting job submission from ILC VO members. This is used to perform the offline analysis, allowing fast processing of large datasets [18].

3.2 ILD LoI Monte Carlo data Mass Production

For the preparation of the ILD LoI over 50 million events of Monte Carlo data were produced on the GRID, using the full simulation and reconstruction software provided by ILCSoft. This corresponded to over 50TB of data stored on Grid storage elements across over 500,000 files. The production of this Monte Carlo data was performed using a set of scripts written in *bash* which were used in conjunction with a MySQL database. The database served as a queriable data catalogue for the persistent data produced, as facilitated the necessary bookkeeping for the processing of the data.

This system was evolved from a production system, which had been successfully used for a small scale production made over a longer period of time. When scaling up the system to make use of the increased computing resources made available for the ILD LoI (1000's of CPU's) the increased stress on the system revealed some of the shortcomings of the design. The clients placed a prohibitive load on the database server, and the use of *bash* as the scripting language made fail-safe operation difficult to achieve. Additionally, the evolution of the system over time led to a database schema which was un-normalised, redundant and whose relational structure was not optimised. This affected the overall performance of the system, but also meant that repairing errors in the database became time consuming.

4 The Production System

A new production system has been developed for massive data processing in ILCSoft. The system accepts as an input user or community requests for data production. As an output the system delivers the requested data and its respective data catalog entries which can be easily queried through a web interface. The schematic overview of the system is represented in Fig. 1.

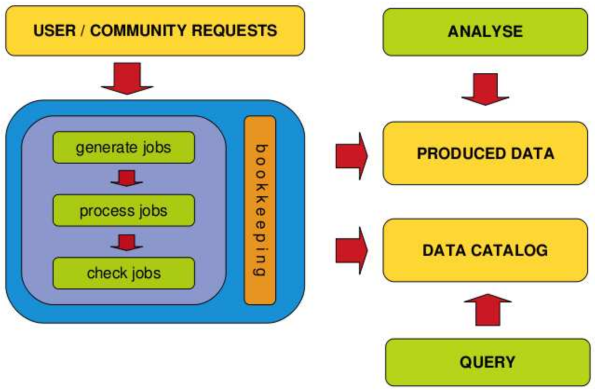

Figure 1: Schematic overview of the production system with its main tasks, generation, processing, and checking of jobs.

4.1 Overview

The production system consists of four main components:

- A central database

- A client-server set of python modules for running the production
- A web interface for managing the production system (internal access)
- A web interface for querying the data catalog (external access)

The central database is implemented in MySQL and it is the main component of the system. It serves different system requirements, such as bookkeeping for the production, storing user requests, storing the data catalog and generating the production jobs from the production requests.

The client-server set of python modules is responsible for running the individual production tasks. The independent production tasks:

- generating jobs from production requests according to available data
- job submission through the Ganga interface and job status monitoring
- final data quality checks and updating of the data catalogue

are assigned to different threads on a production server that uses SimpleXMLRPC [20]. The individual threads communicate to the central database using the MySQLdb [19] python module and can launch their own individual workers in a producer/consumer style by using the python module Queue [21]. The Ganga tool (also developed in Python) is used as an interface for using distributed (heterogeneous) computing resources. It also provides an abstract interface for the production "jobs". Finally the production server is monitored and controlled via the xmlrpc protocol from one or more python clients (threads).

The system incorporates both an internal and external web interface using the python module Django to provide a more user-friendly interaction with the system. The internal web interface is used by the production coordinator to manage the system and to accept new requests for data production. The external web interface provides an interface for the end users to query the data catalog, to create new requests for data production and finally to keep track of their request status (request tracker). Fig.2 shows a schematic design of the complete system.

4.2 Requirements and Design

Special emphasis was placed on the design of the central database schema as it is the core of the whole system and ensuring its data integrity is of the utmost importance.

The database schema is normalized (close to the 3rd normal form) and makes use of stored routines and views to provide a level of abstraction to the underlying schema. This level of abstraction ensures that the underlying schema can be tuned, changed,

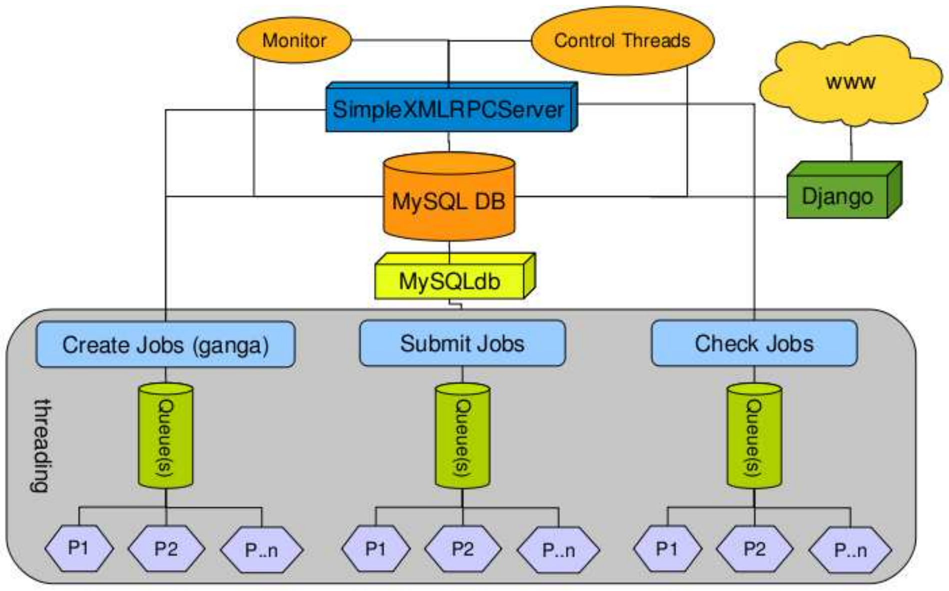

Figure 2: This figure shows the interaction between the different components of the system. The gray box represents the independent production tasks which run in a threading environment. The independent threads connect to the database using the MySQLdb module and are under control of a python XMLRPC server. The server provides an API for controlling and managing the production which can be accessed by any kind of client connection, such as a human interaction or an automated set of monitoring and control threads. Finally, user-friendly web interfaces are provided by Django for the overview of the system and for querying the data catalogue.

updated or even completely replaced without causing a failure in the client-server communication or the return of unpredictable results.

Stored routines are also used for generating the production jobs directly inside the database as this has several advantages over the alternative approach of using a client to generate the jobs:

- No overhead from exchanging the data between the client and the database
- Overall performance is not affected by the data transfers
- Less code needed for validation and error checking
- Clients become more lightweight and can be more easily ported to other operating systems / languages

It should be noted that the use of stored routines, whilst reducing the load on the clients, typically places an increased load on the database server. In addition, they often require for the use of non-standard SQL functionality. Both of these facts were considered during the implementation of the database server.

Care was specifically taken in Normalizing the database, defining it's data structure and the relations between the different entities during the design of the production system. Fig.3 shows a simplified view of the relationship between different file types and their meta data as a result of the normalization process.

A client-server approach was chosen for the implementation of the python modules as this provides flexibility in evolving the user interface. The communication between the clients and the server is done with the python xmlrpclib module. A base class is provided to allow the system to be extended for running other independent production tasks in separate server threads.

5 Conclusion

The new production system has been seen to address a number of the performance bottle necks identified during the mass production of the Monte Carlo data for the ILD Letter of Intent. The utilisation of the functionality offered by a modern database such as triggers, stored procedures, views, etc. is not only able to reduce the load on its client applications, but is also able to play an important role in ensuring the integrity of the whole production system. The development of the system has presently reached the prototyping stage where the core of the system centred around the database has been implemented; the development of further functionality needed for a production release such as a web interface is conceptually solved but remains to be integrated. The prototype system will be thoroughly tested in the near future using actual MC production for ILD.

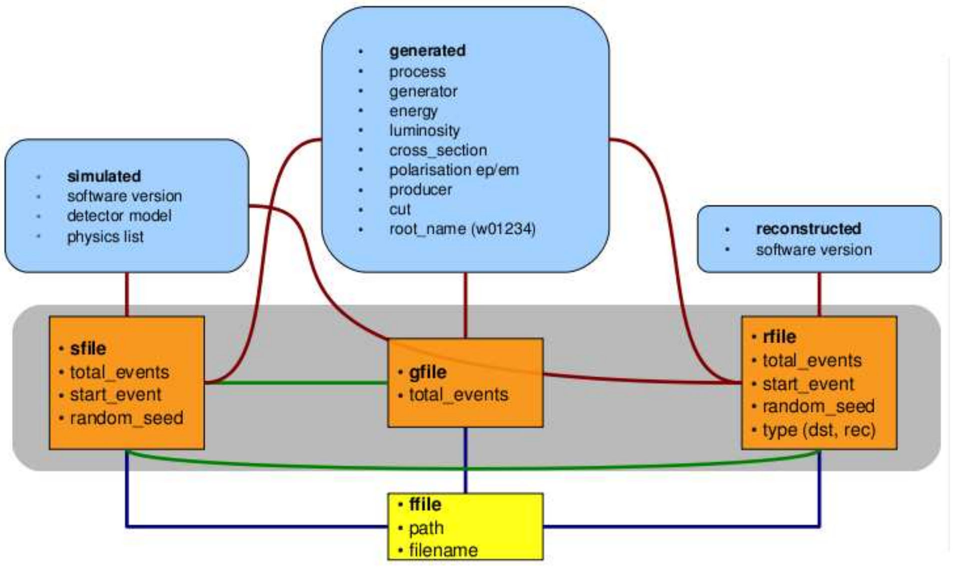

Figure 3: This figure shows the relationship between different file types and their meta data. At the lowest layer (ffile) files are only considered by their path and filename in the storage element and therefore can represent any file stored on the Grid. In the middle layer (gray box) the 3 different file types contain more specific attributes though they all share the underlying ffile properties in a one to one relationship. Finally the upper blue layer contains different sets of meta data which were used for the creation of the files. This meta data is shared by more than one file in a one to many relationship. The relationship between the files themselves is required to keep track of the production process.

Acknowledgement

This work is supported by the Commission of the European Communities under the 6^{th} Framework Programme "Structuring the European Research Area", contract number RII3-026126.

References

[1] F. Gaede, J. Engels, "Marlin et al - A Software Framework for ILC detector R&D", EUDET-Report-2007-11.

[2] URL:www.eudet.org.

[3] URL:http://www.mysql.com.

[4] URL:http://www.python.org.

[5] URL:http://www.djangoproject.com.

[6] Ganga: a tool for computational-task management and easy access to Grid resources, arXiv:0902.2685v2 [cs.DC] 9 Jun 2009

[7] The ILD concept group, "International Large Detector Letter of Intent", http://www.ilcild.org/documents/ild-letter-of-intent

[8] http://polzope.in2p3.fr:8081/MOKKA/

[9] http://lcio.desy.de

[10] http://ilcsoft.desy.de/portal/software_packages/gear/index_eng.html

[11] M. Thomson, "Particle Flow Calorimetry and the PandoraPFA Algorithm", NIMA 611 (2009) 25-40.

[12] S. Hillert for the LCFI Collaboration, "LCFI Vertex Package", arXiv:0811.4759v1 [physics.ins-det] 28 Nov 2008

[13] J. Abernathy, P. Conley, K. Dehmelt et al. "Update on the Status of MarlinTPC", Eudet-Memo-2009-16

[14] A. Bulgheroni, P. Rolof, J. Behr, A. F. Zarnecki, Y. Furletova ,"EUTelescope, the JRA1 tracking and reconstruction software: a status report (Milestone)", EUDET-Memo-2008-48

[15] R. Poeschl, "Calice Data Processing", EUDET-Memo-2007-57-1.

[16] M. Killenberg, S. Turnbull, "A Modular TPC Endplate Description for GEAR", EUDET-Memo-2008-31.

[17] V. Boudry, "Proposal for an LCIO format for the DHCALs", Presented at the Annual Meeting 2009 of the EUDET consortium, http://ilcagenda.linearcollider.org/materialDisplay.py?contribId=88&sessionId=26&materialId=slides&confId=4127

[18] The EUDET JRA1 Group, "JRA1 Delivery: Test Report on the EUDET High Resolution Pixel Telescope", EUDET-Report-2008-04.

[19] URL:http://mysql-python.sourceforge.net.

[20] URL:http://docs.python.org/library/simplexmlrpcserver.html.

[21] URL:http://docs.python.org/library/queue.html.