ILCDIRAC, a DIRAC extension for the Linear Collider community

[1]C Grefe, [1]S Poss, [1]A Sailer, [2]A Tsaregorodtsev,
on behalf of the CLIC detector and physics study
[1]CERN, 1211 Geneva 23, Switzerland
[2]CPPM - IN2P3, 168 Av de Luminy, 13009 Marseille, France

E-mail: `christian.grefe@cern.ch`, `spconsulting@gmail.com`, `andre.sailer@cern.ch`, `atsareg@in2p3.fr`

Abstract. IlcDirac is a complete distributed computing solution for the Linear Collider community. It's an extension of the Dirac system and now used by all detector concepts of the LC community. IlcDirac provides a unified interface to the distributed resources for the ILC Virtual Organization and provides common interfaces to all ILC applications via a simplified API. It supports the overlay of beam-induced backgrounds with minimal impact on the Storage Elements by properly scheduling the jobs attempting to access the files. IlcDirac has been successfully used for the CLIC Conceptual Design Report and the ILC SiD Detailed Baseline Design, and is now adopted by the LC community as the official grid production tool. Members of the CALICE collaboration also use IlcDirac within their own Virtual Organization.

1. Introduction

IlcDirac is a grid solution dedicated to the Linear Collider (LC) community. The ILC and CLIC communities share the same Virtual Organization (VO), ILC, as they use the same software and resources. IlcDirac is an extension of the Dirac [1] framework. IlcDirac was originally designed and implemented for the production of the Monte Carlo data required for the physics performance studies of the CLIC Conceptual Design Report (CDR) [2], and was later adopted by the entire LC community. In particular, it was used for the production and analysis of the Monte Carlo data required for the ILC SiD Detailed Baseline Design document [3] and is now also used by the ILC ILD detector concept.

The core of IlcDirac is built on top of the workflow API from Dirac [1]. It is used for application handling during job execution. In addition, IlcDirac depends on several core Dirac services such as the File Catalogue [4] and the Transformation System [1].

Of the many applications used in the LC community, 14 are currently supported natively by IlcDirac. Among them are detector simulation frameworks based on Geant4 (SLIC and Mokka), reconstruction and analysis frameworks (Marlin and org.lcsim, standalone particle flow reconstruction SLICPandora, based on PandoraPFA), Monte Carlo generators (Whizard, Pythia) and ROOT. The LC applications use the LCIO [5] persistency format to pass event data between them. The different programs are JAVA and C++ based and use XML or Geant4 macro-like steering files. For the CLIC CDR the use of all of these programs was necessary, and for the simulation, reconstruction and analysis of millions of events was required.

In Section 2 the handling of the application is explained, both for the installation and the definition within the jobs. The handling of the overlay of the background is explained in Section 3. Finally, Section 4 gives a summary and conclusion.

2. Application Handling

For ease of use, the heterogeneous interface of all the different applications is hidden behind a homogeneous interface providing the typical application parameters: name, version, number of events, steering files, input file name(s), output file name(s), log file, and final location of the output.

The handling of the applications is two-fold: the installation and the configuration of the application during job definition.

2.1. Application Installation

The physics performance of the CLIC detectors was studied for the CLIC Conceptual Design Report. During the validation, the software was often modified. This led to a large number of application versions with rapid releases that were used to produce different samples. In addition, the CLIC computing model does not enforce specific sites to run any application. The management of the application installation at the site became an issue: with limited manpower, having dedicated personnel to handle this task was not possible.

The adopted strategy was to use the DIRAC pilot job paradigm, and install the software once per pilot job, if needed. The size of the software packages vary from a few megabytes to a few hundred megabytes, which is small compared to the data files.

First, the installation directories are chosen: The software can be either installed in the shared area (identified by the environment variables VO_ILC_SW_DIR for the LHC Computing Grid (LCG), or the OSG_APP for the Open Science Grid (OSG)) or the job directory. If the installation fails in all possible areas, the job is set to waiting again to be picked up by another pilot job, and the original pilot job is stopped.

To handle dependencies between applications, they are sorted in dependency relations, the highest in the hierarchy being installed first. Because several jobs can attempt to write in the same directories (e.g., OSG sites which do not use access rights for the shared area), a locking mechanism is introduced. The principle is based on checking the presence of a lock file. If the file exists, the job will wait for one minute before evaluating the presence again. Once the file is removed by the job that originally created it, the next job can proceed. It will then create the lock file for the time needed to install the software. The next step verifies the existence of the software in the area. This check includes the validation with md5 sum [6] of the contents of the software directory.

Some applications are HEAD revisions and must be overwritten systematically so that the latest version is always used. Corrupted installations or HEAD revisions are removed.

The application installation uses pre-compiled tar balls. The tar ball is obtained from a DIRAC Storage Element, and its md5 sum is checked against a value stored in the central configuration. This ensures the integrity of the file. The contents of the tar ball are also checked against a file manifest. Once the installation is completed, the environment variables are defined for each application. In addition, as ILC and CLIC use JAVA applications (e.g., org.lcsim), a check of the available JAVA is performed. Finally, a clean-up of the tar balls is done to free the area of unnecessary remains.

In the future, CVMFS [7] will also be considered. This should considerably improve the software availability. The monitoring of which application is available at which site would be considerably simplified. The installation in the job directory will still be done if the CVMFS cache is outdated or if the software used must be the HEAD revision.

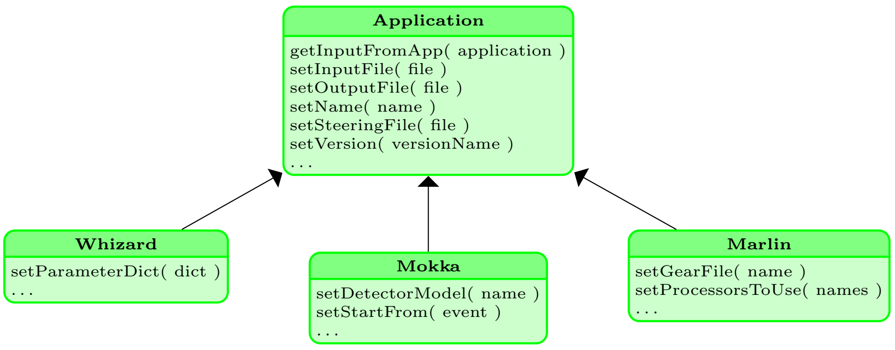

Figure 1. Application class diagrams.

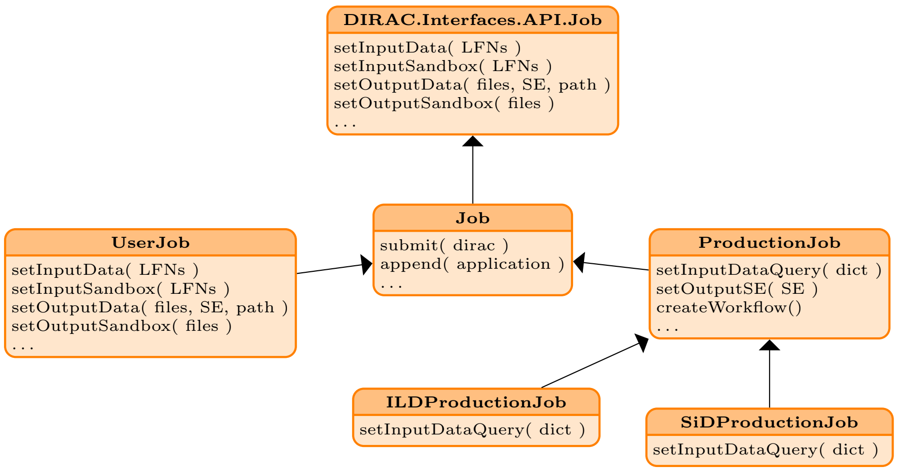

Figure 2. Job class diagrams.

2.2. Job and Application definitions

The design of the job and application interfaces in ILCDIRAC was motivated by ergonomy and flexibility. The basic principle is that they should not be tightly bonded with each other.

An application must be definable outside a job, and should not be mixed with any job property. Additionally, the applications should all be described by a set of elementary properties that define it. In particular, within ILCDIRAC, an application may require input data and/or a steering file to produce some output. An application must be uniquely defined given a name and possibly a version. It typically requires a steering file or option file, and will produce a log file. Its output could be the input of a subsequent application, so a means to connect applications together must be provided.

All ILCDIRAC applications inherit from the Application class (Figure 1). The design follows the Template Method design pattern. For example, to reduce the failure rate due to errors in the application definitions, the submission algorithm includes checks of the sanity of the definitions: the version used for an application is checked for existence in the central configuration.

The Job classes (Figure 2) are also designed in a generic manner. A base Job class, inheriting from the DIRAC Job class, is implemented and allows for the definition of general job requirements like the CPU time and platform required. Two main subclasses are added. The first one is a `UserJob` class that allows for the setting of `InputSandbox` and `InputData`,

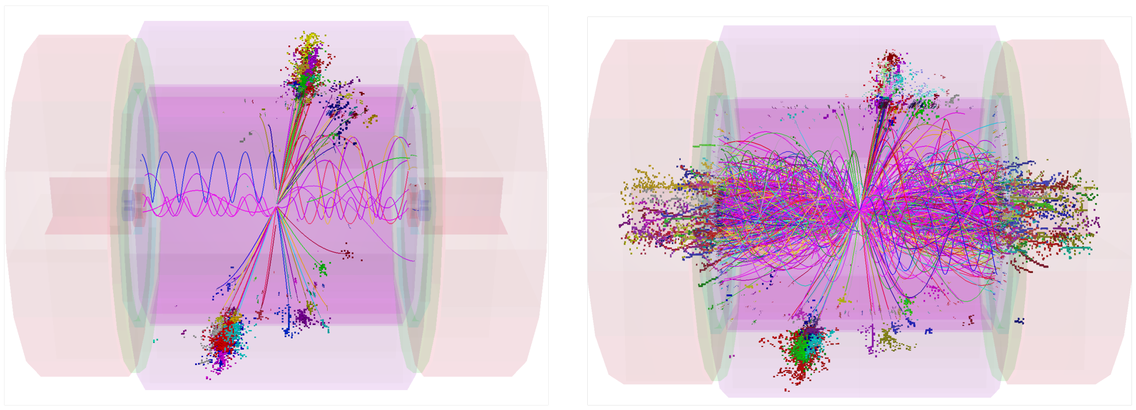
Figure 3. Signal event without (left) and with (right) overlaid $\gamma\gamma \to$ hadron events.

as well as `OutputSandbox` and `OutputData`. This job type, as its name suggests, is intended for user jobs, not production jobs. The latter require the second subclass, `ProductionJob`. It differs in the fact that it does not allow `InputSandbox` nor explicit `OutputSandbox` definitions, and the jobs cannot be submitted directly to the DIRAC Workload Management System. This type of job is intended to be submitted to the DIRAC Transformation System that takes care of preparing and submitting the jobs. For all job types, the application interface is similar, consisting in an 'append' method taking an Application instance as argument. This means that all checks performed for a user job are also available for production jobs. This has the major advantage of re-usability application definition.

3. Overlay of Background Events

A large number of low-energy photons originating from beam-beam effects cause the production of particles in addition to those created by the interaction of interest. The most important background for the physics performance are the $\gamma\gamma \to$ hadron events. For example, at a 3 TeV CLIC machine, 3.2 of these events occur on average [8] during every bunch crossing. Due to the short time between bunch crossings of only 0.5 ns, 60 bunch crossings of backgrounds have to be overlaid with a single physics event. Therefore, a total of about 200 background events are needed for every signal event. An example of such an event is shown in Figure 3, where and event is shown without and with the overlaid $\gamma\gamma \to$ hadron events. With several million events produced for the physics performance studies at CLIC it is obvious that the background events should not be simulated with every physics event. Instead, a limited sample is re-used for an efficient use of the computing infrastructure.

The adopted strategy is to simulate the background and signal events separately and merge their hits before digitization. For this purpose, 67 000 $\gamma\gamma \to$ hadron events were simulated and distributed over 67 files. These background events can be re-used, as they are uncorrelated with the signal events and, in addition, randomly distributed across the 60 bunch crossings for each signal event [9].

As the limited storage resources available for the ILC VO makes the replication of the background samples to all possible sites impossible, it is necessary to first download some of the background files, chosen randomly from the list. This leads to potentially serious issues from the storage elements: the background files being required many times for different jobs, accessing them through the SRM services often leads to a reduction of their performance.

An Overlay service was developed within the ILCDIRAC extension to overcome those issues. This service is in charge of scheduling the file accesses across the jobs. The flow is the following: when a job starts, it contacts the service to check if it can run at a given site. The service holds a database of the number of jobs accessing the data at the different sites. If the number of running jobs at the given site is lower than a configurable threshold, the job starts getting the files. Meanwhile, the database is updated to include this job. In case the threshold is reached, the job is told to wait. In that case, the job queries the system every minute until it can proceed. In case it was not able to proceed after a large number of iterations, it stops running, and is declared as Failed in the Dirac Workload Management System. In case it can proceed, the job reports back to the system once it finished accessing the files. The database is updated, and the job can proceed with the data analysis. To account for the case where the jobs get killed or fail during the execution, an agent was developed to count and correct the number of jobs running at each site that are attempting to get the background files.

This procedure was used successfully for the entire CLIC Conceptual Design Report production periods as well as for the SiD Detailed Baseline Design report.

4. Summary and Conclusions

IlcDirac provides a common interface to the distributed resources for the ILC VO and provides common interfaces to all LC applications via a simplified API. It supports the overlay of beam-induced backgrounds with minimal impact on the Storage Elements by properly scheduling the jobs attempting to access the files. IlcDirac has been successfully used for the CLIC Conceptual Design Report and the ILC SiD Detailed Baseline Design, and is now in use by the entire LC community as the official grid production tool. Members of the CALICE collaboration also use IlcDirac.

[1] Tsaregorodtsev A *et al.* 2010 *Journal of Physics: Conference Series* **219** 062029 URL `http://iopscience.iop.org/1742-6596/219/6/062029`

[2] Linssen L, Miyamoto A, Stanitzki M and Weerts H (eds) 2012 *Physics and Detectors at CLIC: CLIC Conceptual Design Report* (CERN) CERN-2012-003, `http://arxiv.org/abs/1202.5940`

[3] Behnke T, Brau J E, Foster B, Fuster J, Harrison M *et al.* 2013 (*Preprint* `1306.6327`)

[4] Tsaregorodtsev A and Poss S 2012 *Journal of Physics: Conference Series* **396** 032108 URL `http://stacks.iop.org/1742-6596/396/i=3/a=032108`

[5] Aplin S, Engels J, Gaede F, Graf N, Johnson T and McCormick J 2012 *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE* pp 2075–2079 ISSN 1082-3654

[6] Rivest R 1992 *MIT and RSA Data Security, Inc.* URL `http://tools.ietf.org/html/rfc1321`

[7] Blomer J, Buncic P, Charalampidis I, Harutyunyan A, Larsen D, and Meusel R 2012 *Journal of Physics: Conference Series* **396** 052013 URL `http://stacks.iop.org/1742-6596/396/i=5/a=052013`

[8] Barklow T, Dannheim D, Sahin O and Schulte D 2011 Simulation of gamma+gamma to hadrons background at CLIC `https://edms.cern.ch/document/1157767`

[9] Schade P and Lucaci-Timoce A 2011 Description of the signal and background event mixing as implemented in the Marlin processor OverlayTiming CERN `http://edms.cern.ch/document/1144892/`