ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-WEAPL05

PARC: A COMPUTATIONAL SYSTEM IN SUPPORT OF LASER MEGAJOULE FACILTY OPERATIONS

J. P. Airiau, S. Vermersch, V. Beau, E. Bordenave, H. Coïc, T.C. Chies, V. Denis, L. Lacampagne, C. Lacombe, L. Le Deroff, X. Julien, P. Fourtillan, S. Mainguy, M. Sozet
CEA - CESTA, F-33116 Le Barp, France

Abstract

The Laser MegaJoule (LMJ) is a 176-beam laser facility, located at the CEA CESTA Laboratory near Bordeaux (France). It is designed to deliver about 1.4 MJ of energy to targets, for high energy density physics experiments, including fusion experiments. The first 8-beams bundle was operated in October 2014 and a new bundle was commissioned in October 2016. The next two bundles are on the way. PARC[1] is the computational system used to automate the laser setup and the generation of shot report with all the results acquired during the shot sequence process (including alignment and synchronization). It has been designed to run sequences in order to perform a setup computation or a full facility shot report in less than 15 minutes for 1 or 176 beams. This contribution describes how this system solves this challenge and enhances the overall process.

INTRODUCTION

LMJ facility is a very complex physic instrument. Many advanced technologies and knowledge are used in this project. All these functions come with dedicated instrument and software. Based on feedback from previous laser facility exploitation, the need of computational system to be able to perform complex and various computation with heterogeneous software has been identified. Such a system will insure performances, modularity and scalability.

MULTIPURPOSE COMPUTATIONAL SYSTEM

PARC is a generic computational platform [1] with four main features in charge of:

- Exchanging data (settings and results) with Control Command System (CCS),
- Executing sequentially heterogeneous code file from various physical field (laser, alignment, synchronization) ,
- Distributing computation in order to reduce execution time (less than 15 min) ,
- Building multi-level reports.

Communication with CCS

PARC is part of the Control Command Application Layer. It uses available API to read and write the following data:

- Settings from the central settings DB (GCI),
- Diagnosis results from the central shot DB (GTIR),
- Logistical data (e.g.: component replacement) from the central maintenance DB (GMAO).

PARC acts as a slave system of the supervisory control. It receives web service calls to perform automatic function (prediction, shot report, etc.).
An internal wrapper has been developed to manage the data transfer between PARC and CCS (and vice versa). It is based on a dictionary file describing the source data (CCS) and the target data (PARC). The wrapper fulfills three features:

- Type conversion : heterogeneous types from CSS are converted in PARC type (string, double), including file conversion (curve, multicurve, image),
- Unit conversion : heterogeneous units from CSS are converted in International System of Units (ISU),
- Data validity or lack: CSS validity attributes are associated to PARC parameter and a default value is defined for each expected but missing data.

These features insure to operate computation in a homogeneous environment (both type and unit). Specific mechanisms have been developed to propagate error code and message. The entry dataset consistency is essential for the computation. It makes the system stronger to unexpected error.

Computation Sequencer

Many elementary software have been developed for the LIL facility (LMJ bundle prototype). These small code modules were characterized by their feature, interface and language.

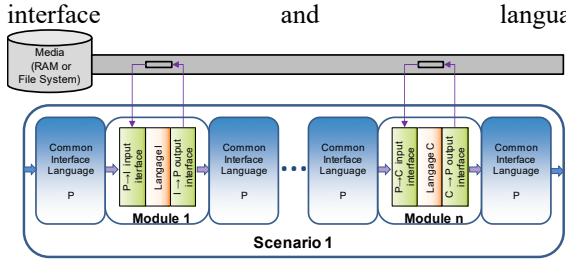

Figure 1: Modules and scenario structure.

In order to create an executable sequence composed of these elementary modules, it is necessary to define a Common Interface Language (CLI). This language is used as a container (Figure 1).

[1] PARC: French acronym for automatic bundle settings prediction.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-WEAPL05

For each language, an interface allows the exchange with the file system or the memory (RAM). The data flow uses a simple formalism for an easy access in the code. The structure of the parameter files (XML) is mapped on the memory structure. Each module is composed of an XML description file (I/O variables, language, feature, link to code file) and a unique code file.

Computation Distribution

The way to distribute the computation depends on the number of simultaneous executions expected. The "granularity level" of the facility requires this distribution (LMJ, bundle, quad, beam).

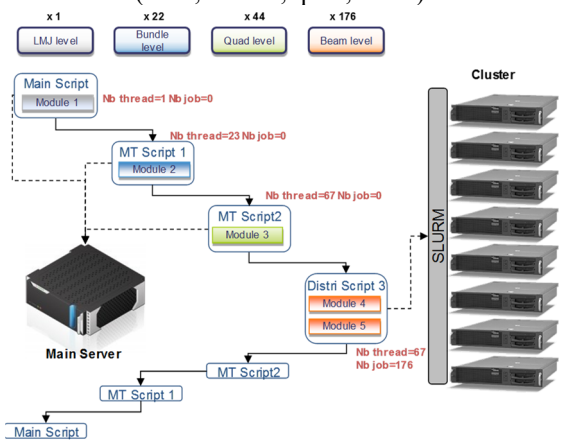

Figure 2: Multithreading and distribution.

The LMJ, bundle and quad levels are multithreaded (67 processes max). Beam level is distributed. This configuration is based on the observation that most of the costly operation are done at beam level. This configuration can be changed. Specific operations at beam level need multiple processors (Figure 2).

The sequence or scenario is reduced to a simple XML file that describes the modules used and their granularity level of execution. We have developed a pseudo-compiler to generate the CLI code (Python) between modules. The modules of same granularity level are grouped in the same script. According to the configuration and the granularity level, the script implements a multithreaded execution (MT script) on the main server or a distributed execution (Distri Script) managed by SLURM [2] on the cluster. By default the pseudo compiler generates a sequential code but it is possible to insert loop instruction (*while* module with end criteria) and conditional instruction (*if* module with boolean criteria). Execution behaviour models based on the decorator pattern have been developed to stop or continue a module execution and propagate the error or warning information (code and message).

Reporting

As PARC can produce a massive amount of results (LMJ, bundle, quad, beam), it is important to be able to summarize data at each level in a simple page. This information has to be shared between multiple actors on the facility. We have decided to use a standard portable format for the graphical interface: HTML.

Because data to be displayed is the same for a granularity level (ex.: same results information for each beam), it is possible to use template to describe the layout of the graphical interface. The concept of module has been extended to the GUI. GUI modules can be used to parametrize a scenario or to display scenario results. A GUI module is composed of a description file and a zip file including:

- Scenario template layout files : one or more HTML file for each granularity with all the data,
- Definition file: an XML file including the link between HTML identifiers and data identifiers from PARC DB,
- Code file: generic python file for parametrization GUI module and reports GUI module.

The execution starts with the loading of each element used in the GUI page and the link to the DB (Figure 3). For each element type, a simple HTML model is loaded from the toolbox. Depending on the context (ex.: beam level / beam n°1) the HTML element is built with the corresponding DB data. This element is combined with the others in the layout and the final page is rendered by Jinja [3].

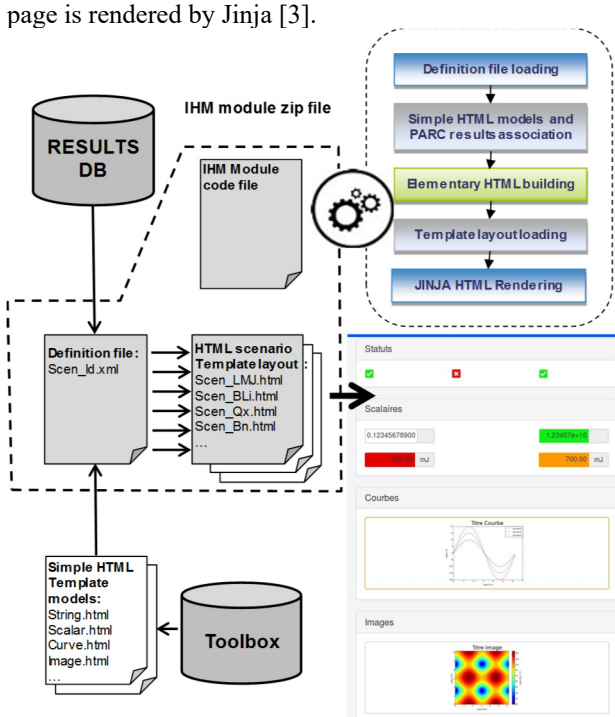

Figure 3: GUI Module.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-WEAPL05

SOFTWARE ARCHITECTURE

A 4-Tier architecture has been used for PARC (Figure 4). The application server (Windows) manages the exchanges between the client software (Windows) and the CCS. The computation server manages the priority (user or CCS request) and the execution of scenario (Linux). The computations are distributed on the cluster (Linux). The distributed storage system GlusterFS gives PARC a shared volume accessible to every user (Samba/GlusterFS). The size of this volume is extensible according to the property of linear scaling of this type of storage. This architecture is hosted on a specific VLAN interconnected with a 10 Gbps Ethernet switch.

The CLI chosen is Python. Most of the modules are written in Python. I/O interfaces have been developed allowing the execution of IDL and C/C++ modules. The laser oriented scenarii use the CEA laser software Miró [4]. A specific interface has been added to this software in order to execute Miró simulation inside a PARC module.

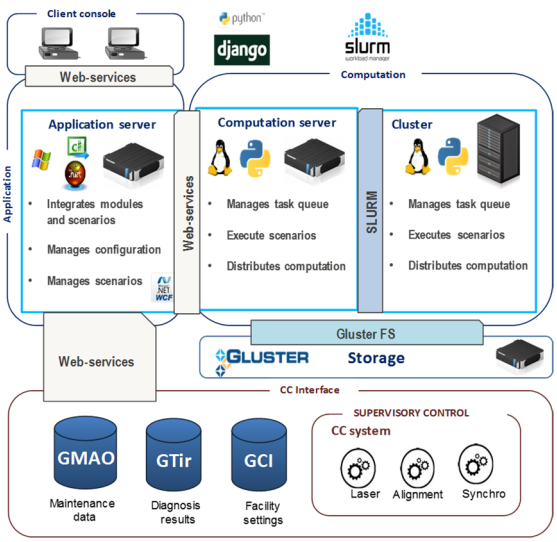

Figure 4: Software architecture.

SUPPORT OF LMJ OPERATIONS

About 15 scenarii are currently used by PARC. The main users of this platform are the laser, alignment and synchronization engineers (about 10 engineers). They can request scenario execution and modify the internal configuration. Reports can be consulted by almost every authorized user on the facility. Some external applications are consumers of PARC results (trends analysis, component aging, etc.).

The main scenarii are:

- Laser Prediction (2 years / 2500 valid predictions): It computes the laser settings (50 parameters / beam) according to the shot request. This scenario can be run several weeks before the experience but an ultimate execution is done 24h before the shot to take into account the configuration changes.
- Automatic shot report (2 years / 3000 reports): It converts diagnosis raw results in physical values and does comparison between predicted, measured and post-simulated signals. It finally generates a shot report (through the GUI and as a zipped file).
- Calibration (1 years / 50 calibration): This scenario is used by the engineers to calibrate the simulation models. It uses fit algorithms on many shot results in order to fine-tune the model parameters (nightly run).

Out of standard shot sequence, PARC is used for machine sequence to configure specific part of the facility:

- Pre-amplifier module calibration: diagnosis sensibility, beam splitter coefficient.
- Amplifier section: diagnosis sensibility, gain, transmission.
- Wave front correction: wave front characterization.
- Frequency Converter Section : diagnosis sensibility, crystal cartography
- Final Optic Assembly: damage characterization (currently done by an external analysis tool: LENA) waiting for incoming inspection Diagnosis.

CONCLUSION

The first step of this project started in 2011 and the conception work in 2013. This project takes his inspiration from the Laser Performance Operating Model (LPOM[5]) developed at Lawrence Livermore National Laboratory for the NIF facility. The development started in October 2013 and the first beta release has been deployed in October 2014. The first stable release has been deployed in March 2015 for the first experience campaign. The service has not been interrupted since this date. A new release is launched each summer during the maintenance period.

The main scenarii are a true substance of 10 years of experience on laser facility and the work of many engineers. Although PARC gives the opportunity to run complex computation in very short amount of time, the real intelligence is in the scenario. Moreover PARC, as a platform, allows the development of new scenario with high performances, scalability and modularity.

REFERENCES

[1] S. Vermersh *et al.*, “The laser Megajoule facility: the computational system PARC”, Proc. ICALEPCS2015, Melbourne, Australia, ISBN 978-3-95450-148938 (2015)

[2] Morris Jette, “Slurm Overview”, Super Computing 2016, SchedMD – Lehi, UTAH, U.S.

[3] A full featured template engine for Python - Armin Ronacher - Austria, http://jinja.pocoo.org/

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-WEAPL05

[4] Olivier Morice, "Miró: Complete modeling and software for pulse amplification and propagation in high-power laser systems", Opt. Eng. 42, 1530 (2003).
[5] M.J. Shaw *et al.*, "Computational Modeling in Support of National Ignition Facility Operations", Proc. ICALEPCS2001, WEAP063, Physics/0111036 (2001).