Journal of Physics: Conference Series **513** (2014) 032051 doi:10.1088/1742-6596/513/3/032051

Virtualised data production infrastructure for NA61/SHINE based on CernVM

Dag Toppe Larsen for the NA61/SHINE collaboration
University of Silesia, Katowice, Poland
E-mail: Dag.Larsen@cern.ch

Abstract. Traditionally, the NA61/SHINE data production is performed by manually submitting jobs to the CERN batch system. An effort is now under way to migrate the data production to an automatic system, on top of a virtualised platform based on CernVM. This will make it easier to both initiate new data productions, and to utilise computing resources available outside CERN. In addition, there is a data preservation perspective. CernVM is a Linux distribution created by CERN specifically for the needs of virtual machines. Data production software and calibration data are distributed globally via the HTTP-based CernVM file system. The NA61/SHINE data production software has been adapted to run under CernVM through CernVM file system. Databases are used to keep track of the data. This will allow the system to present lists of both raw and produced data. If a new data production is needed, the privileged user may choose the data, software versions, and calibrations to be used. Finished jobs will be scanned for errors, and automatically resubmitted for processing if needed. A web-based, graphical user interface for the data production will be available. Finally, the relevant databases will be updated to reflect the freshly produced data.

1. Introduction

NA61/SHINE [1] is an experiment at the CERN/SPS, and it is a continuation of the NA49 experiment. There are three main physics goals: (1) nuclear physics — energy scan for onset of deconfinement and the critical point; (2) neutrino physics — measure reference hadron production; and (3) cosmic ray physics — measure properties of atmospheric interactions. This wide spectre of physics goals necessitate the study of a wide range of reactions. In the context of NA61/SHINE, a reaction is a given combination of beam type and momentum, target type, and year of data taking.

The raw data collected must be reconstructed over several iterations to gradually improve the quality of the reconstructed data. Currently, there are two software frameworks available to NA61/SHINE to reconstruct the data. The first is referred to as the "legacy" framework, owing to its inheritance from NA49. This software is mostly written in proprietary PGI-Fortran, and is based on a set of standalone clients that are executed in sequence to process the data. The second has been named Shine, and is based on C++/ROOT and is a single binary with loadable libraries. The legacy framework is in the process of being phased out in favour of Shine. However, for now the data production infrastructure must support both.

Traditionally, the data production has to a large extent been a manual process. Before a production is started, text-based configuration files are edited by hand to reflect the data to be processed, version of calibration data ("global key"), and which software versions to be

used. After this, the data is submitted for processing as jobs on the CERN batch system using scripts. Next, the produced data is checked using another set of scripts, and eventual failed jobs are resubmitted for another round of processing. Finally, the bookkeeping database is updated via a web interface. Although this approach has worked well for a number of years, it is labour intensive, and introduces the risk of human error. It was therefore desirable to develop a more automatic approach. Further, there are a number of collaborating institutions within NA61/SHINE that can make processing and storage resources available to the collaboration. Thus, it was also highly desirable that a new data production infrastructure could support distributed computing to take advantage of these resources as well.

Recently, data preservation has received increasing attention. The data from a number of past experiments can no longer be accessed or utilised. Many of the NA61/SHINE reactions are "unique" in the sense that they have not been measured at other experiments. There may be future uses for the data that today can not be imagined. It would therefore be highly advantageous if the data preservation aspect is also in the design of a new data production infrastructure.

Together, these requirements were pointing towards a data production infrastructure based on virtualisation. Virtualisation is receiving increasingly more attention as a means for distributed computing. Traditionally, high-energy physics experiments have relied on grid middle-wares to distribute data production jobs to batch systems at different computing sites. However, the wide variety of grid middlewares and versions, operating system distributions and versions, hardware types and configurations, *e*tc. create a heterogeneous computing environment where subtle differences of the overall processing environment can cause a job to succeed at one site and fail on another. Virtualisation tries to sidestep these challenges by distributing virtual machines, rather than jobs, to different sites. Although the underlying hardware might be different, the virtual machines running on top of them will appear identical, thus creating a homogeneous computing environment. The virtual machines running at physically different sites may all participate in the same virtual batch cluster, thus allowing for distributed computing without a middle-ware. Most experiments have already invested heavily in working grid-based distributed computing, which would make a switch to virtualisation-based distributed computing difficult. However, in the case of NA61/SHINE, there was no existing distributed computing infrastructure.

Data preservation typically does not only require the preservation of the data itself, but also the software used in conjunction with the data. The software preservation is much more challenging than the preservation of the data itself, since it does not only involve preserving "files", but also the whole environment that is required to run the software. Compiling and running the software depend on certain versions of compilers and system libraries, which depend on certain versions of the operating systems, which depend on certain versions of hardware. At some point the physical hardware will no longer be available, and the chain collapses. Another approach may be to port the software to newer compiler and system library versions. This is however a resource demanding undertaking, and there is also a risk of introducing new errors to the code. Virtualisation may provide a different approach. Since virtualisation relies on hypervisors providing "virtual" hardware to the virtual machines, it is also possible to provide legacy virtual hardware that legacy operating systems can run on; *i*.e. preserving "hardware" in "software". If this requirement is taken into consideration while designing the virtualised data production infrastructure, it may come at very little additional cost.

2. CernVM eco-system

In general, any Linux distribution can be used as basis for a virtualised data production infrastructure. However, CernVM [2, 3, 4] is not only a distribution tailored specifically for the needs of a virtual machines, but also has a whole eco-system of technologies related to creating

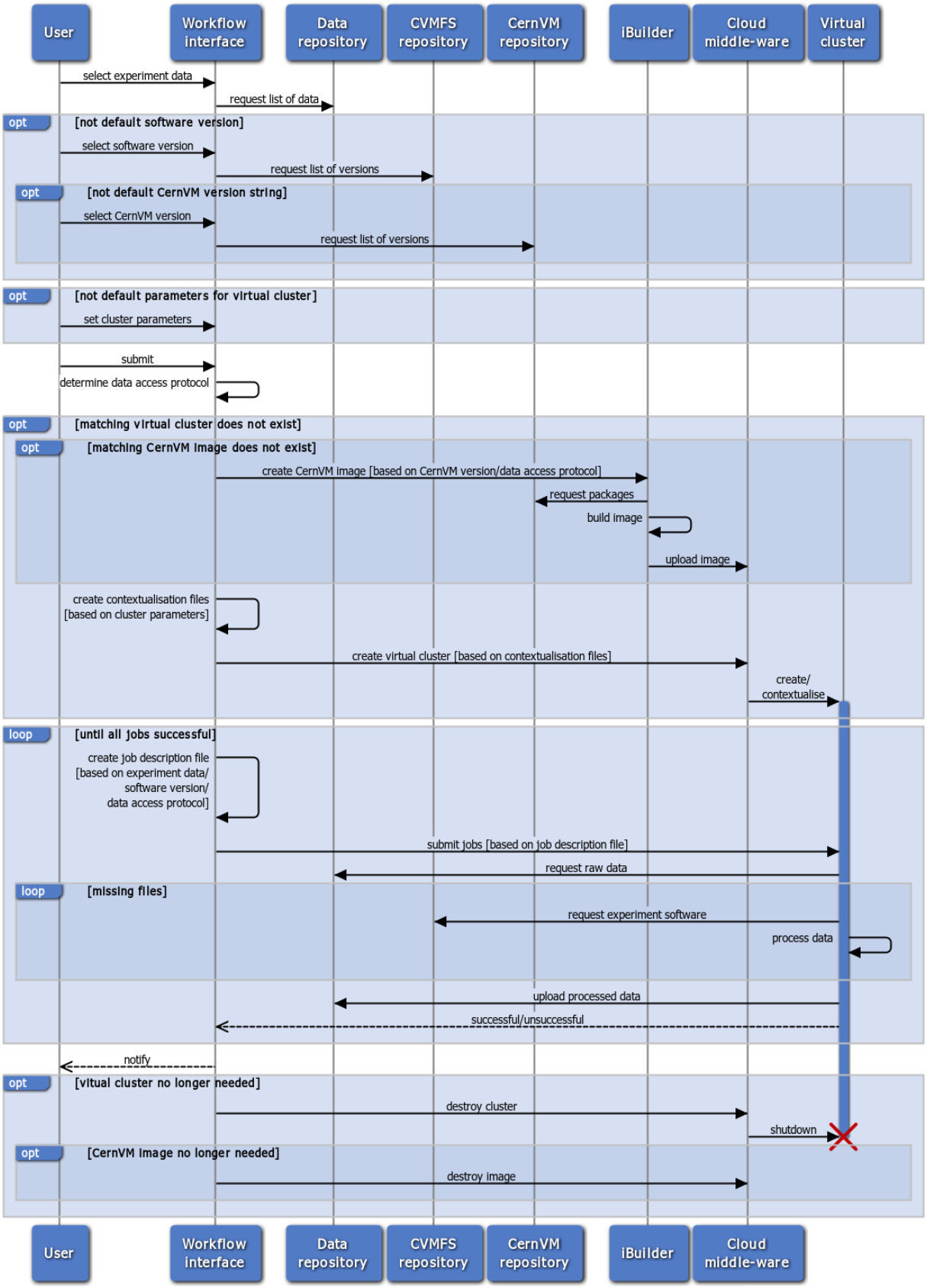

Figure 1. Sequence for data processing using the work-flow interface.

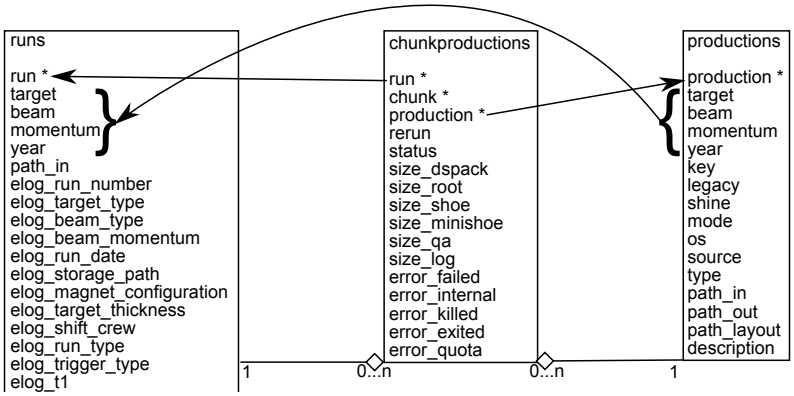

Figure 2. Schema of the production database. Some fields of table "runs" are omitted.

a virtualised processing environment. The combined features of the CernVM eco-system made it a good choice for the basis of the virtualised NA61/SHINE data production infrastructure.

The CernVM operating system itself is based on Scientific Linux 5, and has a very small image size (about 300MB) to reduce the amount of data needed to be copied for the creation of a new virtual machine instance. There are versions for all popular hypervisors, and it comes in different flavours to support different use cases; *e.g.* batch node, developers' desktop, *e*tc. Also, contextualisation is supported.

The CernVM file system [5, 6, 7, 8] is used to globally distribute experiment software and calibration data. It is based on HTTP and a hierarchy of cache-servers. Each file is compressed and a hash key is generated to uniquely identify the file. On first access the file is downloaded, decompressed and semi-permanently cached.

The CernVM on-line is a service for centrally management of virtual machines across multiple physical clouds; *e.g.* some virtual machines can run on CERN OpenStack, some at other collaborating institutes, and some at commercial clouds like Amazon. The service supports multiple protocols transparently to the user. It also handles contextualisation.

The CernVM iBuilder [9] is the tool used to build the CernVM disk images. Of particular importance is that it uses a database for strong versioning of the software included on the disk image. This makes it possible to exactly recreate the disk images at a later stage. In turn, this allows legacy CernVM versions to be modified and built to include new support for new technology; *e.g.* future hypervisors and file transfer protocols. This has an important data preservation perspective [10].

3. Implementation of the virtualised data production infrastructure

Based on the requirements and choice of core technologies, it is possible to start the implementation of the data production infrastructure. The processing of raw data has several steps: (1) selection of data to be processed (reaction data set); (2) selection of software versions, global key version, *e*tc. (processing environment); (3) crate configuration and job description files; (4) submit the jobs to a batch system; (5) check finished jobs for errors; (6) resubmit failed jobs; (7) declare processing as completed. Each of these steps again has several sub-steps. It is a goal that the overall process should be highly automatic and only require a minimum of user intervention. For example, the system can simplify the selection of steps (1) and (2) by providing a list of valid choices in a graphical user interface. Further, steps (3) and (4) can be done automatically after the user has approved the processing to be started. Steps (5), (6) and (7) can be performed at regular time intervals by a cron job. A more detailed view of this process can be seen in the sequence diagram of figure 1.

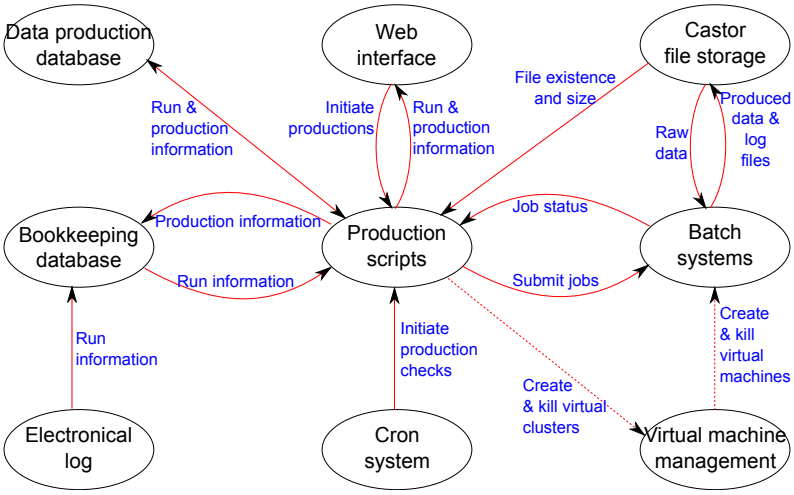

Figure 3. Interaction between different components of the data production framework.

A database is required to keep track of the production of data. Although there is an existing bookkeeping database that contains information for the raw data of the various reactions, as well as the produced data, it appeared difficult to extend it to contain the level of information needed to manage the data production and keep track of the status of each job. It was therefore considered a better choice to develop a database based on SQLite specifically for the data production. However, it allows for information exchange with the bookkeeping database.

The database schema is shown in figure 2. Table "runs" contains all information about the runs. A run is a time-limited O(1h) fraction of a reaction. Each entry correspond to one run, hence the run number is the primary key. Information about the runs are imported from the bookkeeping database and the electronic log. The fields originating from the electronic log are prefixed with "elog_". The fields "target", "beam", "momentum" and "year" are used to define and determine which reaction the given run belongs to. These fields are derived from the electronic log entries. It is expected that this table will contain $O(10^4)$ entries.

Table "productions" contains all information about the productions. A production is a unique combination of reaction and data production environment (software versions, global key version, *e*tc.). The system is responsible for assuring that multiple productions with the same parameters do not exist. This uniqueness will also be enforced at the file system level by encoding the parameters in the path. A unique numeric value is generated and used as a primary key for this table. It is expected that this table will contain $O(10^2)$ entries.

Table "chunkproductions" contains all information about the production (processing) of an individual chunk for a given production. A chunk is a size-limited O(1GB) fraction of a run. It can be uniquely identified by production identifier, run number and chunk number, which are also used as a composite primary key. The additional fields are mainly used for checking of the productions. Fields prefixed by "size_"are used for storing the sizes of the various produced files. Small file size may indicate that the file was not properly created. A "null" value indicates that the file does not exist at all. Fields prefixed by "error_" are used to store the number of errors of given type discovered while searching the log file. A "null" value indicates that the log file has not been scanned for a given type of errors. The field "rerun" indicates the number of times a job has been resubmitted. It is assumed that a chunk that has failed processing three times is inherently broken, and further processing should not be attempted. The field "status" should indicate whether the job is waiting to be processed, is being processed, is being checked, or has finally finished processing either successfully or unsuccessfully. It is expected that this

```
-bash-4.1$ ./produce
Usage:
./produce <command>
<command> one of:
reactions     - list all reactions in database
productions   - list all productions in database
./produce <command> <path_in>
<command> one of:
regreaction   - register all reactions found at path_in in database
./produce <command> <target> <beam> <momentum> <year> [<key> <legacy> <shine> <mode> <source> <path_in> <description>]
<command> one of:
regproduction - register new production in database
produce       - start new production
check         - check production for errors and update database
summary       - production summary
reproduce     - reprocess chunks with with errors for production
okchunks      - list all OK chunks for production
```

Figure 4. Master work-flow manager script for data production.

table will contain O(10^6) entries.

As can be seen in figure 3, there is extensive communication and interaction between the different components. This communication is handled by a set of scripts. In general, the scripts always communicate with both the production database and some other component. Figure 4 shows the options for the master work-flow manager production script. This script is either used directly from the command line, or as the back-end for a web-based work-flow interface. Options "reactions" and "productions" queries the database for the lists of reactions and productions, respectively, and display them. Option "regreaction" registers a new reaction in the database by scanning a xRootd path for raw files and queries the bookkeeping database for electronic log information for the corresponding runs. Option "regproduction" registers a new production in the database based on the the reaction and production parameters the user provides. Option "produce" initiates a new production by creating the necessary configuration and job description files based on the database, and submits the jobs to the batch system. Each batch job will correspond to the production of one chunk. Option "check" checks finished jobs for errors, and flags failed chunks in the database. Option "summary" retrieves production statistics about a given production from the database. Option "reproduce" resubmits chunks flagged as failed to the batch system. Options "check" and "reproduce" may be called at regular time intervals from a cron system. Option "okchunks" will for given reaction display a list of all chunks that are not flagged as failed. By comparing these commands to the interactions indicated in figure 3, it should be possible to see how the options trigger the various interactions.

Figure 3 also indicates a connection to "virtual machine management". Though it is not yet implemented, the data production infrastructure is planed to utilise CernVM on-line to manage the creation and destruction of virtual clusters across multiple physical computing sites, including the OpenStack infrastructure at CERN. This is intended to also have the option to start virtual clusters based on legacy versions of CernVM to run older data production software to allow for virtualisation-based data preservation.

Finally, a work-flow manager web interface is under development. A screen-shot of this can be seen in figure 5. In general, it is a graphical front-end to the work-flow manager script. The exception is the retrieval of information from the production database, since this can be done much more efficiently by querying the production database directly. It follows the use case outline previously closely. The user selects a reaction, and the web interface displays existing productions, and offers to start a new production using validated production parameters from drop-down menus, without the need for the user to manually provide this information. Currently,

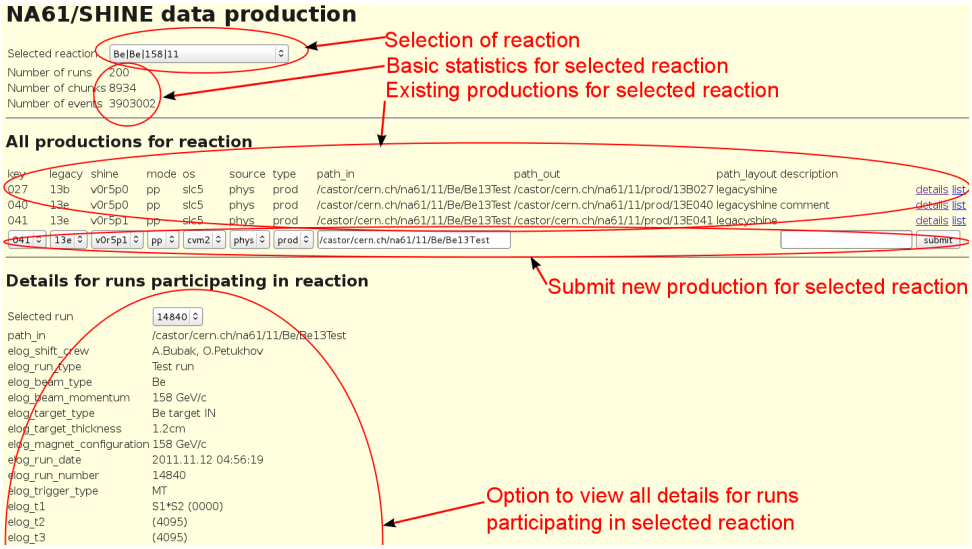

Figure 5. Prototype work-flow interface.

the web interface is "read-only", as the authentication, planed to be based on CERN single–sign-on, is not yet implemented. Also, different interfaces will be tested to see which best confirms to the various use cases. The "reaction" and "production" are believed to be the main entities to centre the interface around.

4. Status

Currently, a production of a full reaction is ongoing to validate the data produced on the CernVM-based virtualised production infrastructure against data produce on the traditional CERN batch system. Once the two systems are found to produce the same results, the virtualised infrastructure will gradually be phased in over the following months. In parallel with this, the web-based work-flow manager will be implemented, and various improvements will be performed. Also, implementing CernVM on-line and upgrading to μCernVM 3 are expected.

Acknowledgements

This work was supported by the National Science Center of Poland (grant UMO-2012/04/M/ST2/00816).

References

[1] NA61/SHINE collaboration *web page* URL `http://cern.ch/na61/`
[2] Buncic P *et al.* 2008 *Proceedings of Advanced Computing and Analysis Techniques in Physics Research (ACAT)* URL `http://pos.sissa.it/cgi-bin/reader/conf.cgi?confid=70`
[3] Buncic P *et al.* 2010 *J. Phys.: Conf. Ser.* **219 042003** (*Preprint* `doi:10.1088/1742-6596/219/4/042003`)
[4] Buncic P *et al.* 2011 *The European Physical Journal Plus* **126** (*Preprint* `doi:10.1140/epjp/i2011-11013-1`)
[5] Blomer J *et al.* 2010 *Proceedings of Computer Communications and Networks (ICCCN)* (*Preprint* `doi:10.1109/ICCCN.2010.5560054`)
[6] Blomer J *et al.* 2011 *J. Phys.: Conf. Ser.* **331 042003** (*Preprint* `doi:10.1088/1742-6596/331/4/042003`)
[7] Blomer J *et al.* 2012 *Proceedings of International workshop on Network-aware data management (NDM)* 49–56 (*Preprint* `doi:10.1145/2110217.2110225`)
[8] Blomer J *et al.* 2012 *J. Phys.: Conf. Ser.* **396 052013** (*Preprint* `doi:10.1088/1742-6596/396/5/052013`)
[9] Charalampidis I *et al.* 2012 *J. Phys.: Conf. Ser.* **396 032022** (*Preprint* `doi:10.1088/1742-6596/396/3/032022`)
[10] Larsen D *et al.* 2012 *J. Phys.: Conf. Ser.* **396 032064** (*Preprint* `doi:10.1088/1742-6596/396/3/032064`)