Classification of Supernovae Using GridPP

WP2.3: Support of Phase A DEV work

Project Acronym	LUSC-A
Project Title	UK Involvement in the Large Synoptic Survey Telescope
Document Number	LUSC-A-05

Submission date	05/04/2018
Version	1.0
Status	Published
Author(s) inc. institutional affiliation	Darren J. White (University of Edinburgh) Robert Firth (Southampton University) Szymon Prajs (Southampton University)
Reviewer(s)	George Beckett (University of Edinburgh) Bob Mann (University of Edinburgh)

Dissemination level	
Public	*Public after comments by GridPP*

Copyright LSST:UK Consortium 2017

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	05/04/2018	First full draft sent to George Beckett (MGB) for comments	Primary author: DJW Notes: RF
0.2	08/05/2018	Second draft based on MGB comments	Primary author: DJW Notes: RF,MGB
0.3	27/06/2018	Third draft based on comments from Bob Mann (RGM)	Primary author: DJW Notes: RF,MGB,RGM
1.0	30/06/2018	First published version	Primary author: DJW

1 Table of Contents

2 Executive Summary

This work here describes an effort to implement a Core Collapse light curve simulation tool (CoCo) using GridPP [1] resources. We built on the experience from previous case study involving Joe Zuntz [2] and we were able to create a more up-to-date workflow that allows our partners to leverage GridPP resources with relative ease. We gained useful insight into the use of EUPS in the provisioning of software, as well as the job submission and management tool Ganga [3]. Both of these tools are flexible and powerful, and can be utilised by many different areas, both within and outside of GridPP. We also used CernVM [4] and CVMFS [5] resources for our development environment, which was found to be useful.

Overall, this work was a success. Our partners were able to develop, access and successfully run CoCo on GridPP. Further work to scale out the workflow is beyond the scope of this case study, but the experience gained means this can be achieved with relative ease if needed. This work also provided several learning experiences for how GridPP works, which we hope to combine into an overall step-by-step guide for new users within LSST.

We were also able to provide some unique edge-case tests of Ganga, which was able to guide development of the tool for future use.

3 Introduction

3.1 Motivation

Type Ia supernovae are an excellent tool for various cosmological investigations—for example, constraining the content of the Universe by measuring its expansion. The LSST survey will observe many such supernovae, providing an unprecedented opportunity to make real progress on these investigations. However, there are several different types of supernovae and it is important to confirm which are of Type Ia before attempting any research.

The usual and most reliable way to classify a supernova is by looking at its spectrum. However, due to the large volumes of supernovae observed by LSST, for a significant majority of the objects we will not have access to information on their spectra. As a result, it has been proposed to attempt a classification based on sampling the time-evolution of their light curves [6]. The reliability with which one can classify supernovae types, from optical properties, is strongly dependent on the telescope's observing schedule—when and how frequently it images each particular supernova—so, to inform decisions about the schedule, researchers in Southampton and UCL are simulating the reliability of supernovae classification for different survey strategies. This involves seeding a virtual universe with supernovae sources and then using an LSST simulator to observe that virtual universe, and finally attempting to make a blind classification of the supernovae that are detected with each schedule; with the eventual aim to rapidly assess the relative merit of any proposed change to the observation pattern. This is a computationally intensive task though one which can be broken down into many, small jobs, making it ideal for a high-throughput computing (HTC) infrastructure like GridPP.

It is hoped that by employing GridPP's substantial resources, the UK team can provide a rapid, validation service which can be used to quickly turn around assessments of the performance of proposed LSST survey strategies, advising the wider LSST community regarding its likely implications for supernovae science.

3.2 Glossary of Acronyms

LSST	Large Synoptic Survey Telescope
GridPP	Grid for Particle Physics
CernVM	CERN Virtual Machine
CVMFS	CERN Virtual Machine File System
VO	Virtual Organisation
(E)UPS	(Extended) Unix Product Support

4 CoCo on GridPP

CoCo is a light curve simulation code, at the time of writing, developed by Rob Firth and Szymon Prajs at the University of Southampton. Given a series of observing times and Core Collapse supernova templates, it is possible to simulate supernova light curves across a range of optical filters and supernova type. For the LSST project the times of observation are calculated using the expected cadence of observation. Partners at UCL are creating a range of observational cadence patterns using the LSST Operations Simulator (OpSim). Taking the light curves produced by CoCo, it is then possible to estimate how suitable different observing cadences are at allowing astronomers to correctly classify supernovae. The work described here was aimed at the first stage: to create a working version of CoCo on the GridPP system that creates light curves based on OpSim cadences.

Additionally, we hoped to provide experience to non-GridPP users in accessing the system, allowing them to develop further tools to perform the classification stage without the need of project-specific help. This allows us to investigate how well-suited the GridPP system and user environment is for a non-experience user from the LSST community.

4.1 Accessing GridPP

GridPP [1] resource access is controlled via Virtual Organisations (VOs), that typically represent specific science groups. An individual user requests an eScience X.509 certificate from their local authorities that, once registered with the relevant VO, can be used to gain access to the specified VOs resources. Accessing GridPP was done via the recommended CERN Virtual Machine (CernVM), selected as it provided a self-contained and portable option for accessing GridPP. The relatively simple setup procedure is documented in a separate document, aimed at GridPP newcomers within LSST (at the time of writing, in preparation).

The CernVM also provided an easy to manage interface to the CernVM File System (CVMFS), a software distribution mechanism that provides repository access to common tools, including those specified by VOs. Each VO can create and publish a repository and must request that this repository is accessible from each of the compute resources that VO members are able to use. The LSST VO CVMFS repository contains the full LSST software stack, as well as specific package versions required by these tools. As described later, it is also possible to complement these versions with your own, ‘local’ versions, via EUPS.

4.2 Porting to GridPP

The starting point for this work was a previous LSST-GridPP case study by Joe Zuntz [2], which ported the *im3shape* galaxy classification software to work on the GridPP system. In order to achieve this, Joe’s work used a tool called UPS (Unix Product Support) to dynamically load software packages as required. These packages, along with im3shape, were kept on Storage Elements (SEs) as TAR files and transferred to computing nodes when a job was executed. This original case study provided a solid base of understanding, but with some issues.

First, the LSST CVMFS repository had changed to use Extended-UPS (EUPS), a Python variant of UPS. EUPS is a package manager that allows users to dynamically configure their environment to enable the use of software packages (or even specific versions of packages) while handling complex dependencies between those packages. For example, if a user wishes to use package A, which is dependent on packages B and C (each with their own dependencies), the user simply has to “set up” tool A, and the description of that tool within EUPS is used to configure tool dependencies and appends system variables as needed. This change to EUPS resulted in Joe's original UPS packages being incompatible with the GridPP/LSST setup. This was resolved by refactoring Joe Zuntz's UPS-registered packages into EUPS-standard descriptions and layouts, while also updating/ installing prerequisite packages needed for CoCo (notably Cython and a more recent C/C++ compilers).

Once this was complete, we were able to use both the LSST CVMFS EUPS-registered packages, as well as our own local EUPS-registered packages, as EUPS can bring together tools from multiple sources. EUPS can then be used to configure the environments on the compute nodes so that our tools can run. Whilst doing this, we were also able to alter Python scripts to allow pip-installation of Python modules directly into our local EUPS-registered Python package, making it easy to update/include Python modules that are useful for this work further down the line.

The locally created EUPS portion of the workflow was then uploaded to SEs along with the relatively static light curve templates via the Dirac toolkit, which provides command-line tools for uploading and managing files and jobs on GridPP.

The biggest issue however, was that the CoCo software was under constant development during testing, so packaging and uploading a new version to a Storage Element each time a small change was made and needed to be tested was time-consuming. Our solution was to develop job scripts which, along with transferring the files from GridPP SEs, downloaded the most recent version of CoCo from GitHub directly, which was then built and executed on the compute nodes of GridPP. This improved the speed of development and testing.

4.3 Job Execution and Management

For Joe Zuntz's im3shape workflow, a shell script was developed which creates and submits jobs via the Dirac toolkit [7]. However, it was decided that the Python-based Ganga tool provided a more user-friendly approach for new users. Ganga includes the ability to split large jobs into sub-jobs based on input data/files, retrieve detailed information about jobs in progress, retrieve output data automatically and can also submit jobs both locally and via a range of computing services, including GridPP. Ganga creates a database of all jobs submitted via its interface, allowing users to access information for all current and past jobs. Ganga can also be run via a command-line interpreter or via submission scripts. This allows users more flexibility when creating and handling job submission. Within the Ganga CLI, users can access job information directly, and can also resubmit jobs as required (such as if a job fails). Ganga achieves this by treating each job as an object which has common methods, such as 'submit', 'kill', 'resubmit', 'remove', etc.

The CoCo job submission script is written in Python (with Ganga-specific commands). The user provides a list of input files (e.g. the OpSim observing cadence files), the number of required light curves per input file to be simulated, and paths to both the EUPS and template tar files. The user can also select a flag which sets the location of where the job will run: locally or on GridPP via Dirac. The script then handles the process of creating individual sub-jobs based on the number of input files, and also creates an input sandbox for each sub-job which contains the executable to be run and the input files. Each job (and sub-job) is then assigned a unique local ID for managing jobs from either the Ganga command-line interpreter or Ganga scripts.

Beyond the process of submitting a job, our approach follows that of the im3shape case study closely. The main executable of the workflow is a shell script which sets up EUPS, loads the required packages, and calls a python script which performs the bulk of the work. One major difference is that our executable also downloads the CoCo repository from GitHub and builds the package before executing the main work script, which reads in the cadence files and produces the required number of simulations per job.

One other change comes about due to our selection of Ganga as the job submission and monitoring mechanism. Selecting output files during job submission (for example, via a wildcard such as '*.dat') will result in any matching files being transferred back to the local machine of the user automatically, obviating the need for additional scripts to obtain the data locally. It is also possible for users to either use the in-built post-processing steps or to define their own post-processing scripts which are executed automatically by Ganga on the completion of a job (such as to merge the output files of sub jobs into a single output file). There is a small downside in that all output from jobs are stored in the Ganga workspace, which is not always intuitive to work with. For that reason, we have also developed a script that, given a job number and output location, collates all output files from all sub-jobs into a single directory. For this project, each output filename is unique and needs to be handled individually, so this is a suitable solution, but may need some modification to work with other projects if post-processor merger or custom scripts are not suitable.

A sample Ganga script is provided in Appendix A, with an example Bash job script in Appendix B. Both of these can be found at [8].

4.4 Accessing Support

The majority of our support requests were aimed at understand how best to marry the workflow we have with Ganga. For wider GridPP, we used two email lists: helpdesk@ggus.org for site/connection issues and gridpp-dirac-users@imperial.ac.uk for more general queries about Dirac and using GridPP.

4.5 Conclusions

Overall, the project was a success. The workflow developed was successfully ported and executed on GridPP and provides a useful resource for those working on simulating LSST cadence with CoCo.

The time taken to complete the work was longer than one would hope for the majority of workflows ported to GridPP but given some of the unique aspects of the project, this is not unreasonable in our case. We also gained valuable knowledge on the submission and management of jobs via Ganga, which will be disseminated throughout the LSST:UK group in the form of a "How-To" document for new users. This will expand on the generic documentation available from Ganga/GridPP and cover some of the more unusual aspects of the LSST VO which gave us issues with development. We were also able to provide feedback to the Ganga development team, and also provide robust tests of the job submission and management system, especially in ways outside of its original design scope. From discussions with GridPP, in particular, this helped inform future development of the tool to provide more user flexibility and support.

4.6 Acknowledgements

The majority of support came from Rob Currie, a GridPP expert and Ganga developer co-located in the same building as the author. The majority of help from the support email lists came from Alessandra Forti and Daniela Bauer.

5 References

[1] “GridPP,” [Online]. Available: https://www.gridpp.ac.uk/.

[2] M. E. J. Z. G. Beckett, Measuring Density of Dark Energy Using Weak Lensing, on GridPP, LSST:UK Technical Report, LUSC-A-01, September 2017.

[3] “Ganga,” [Online]. Available: https://ganga.readthedocs.io/en/latest/index.html.

[4] “Cern Virtual Machine,” [Online]. Available: https://cernvm.cern.ch/.

[5] “CernVM File System,” [Online]. Available: https://cernvm.cern.ch/portal/filesystem.

[6] M. Dai, S. Kuhlmann, Y. Wang and E. Kovacs, “Photometric classification and redshift estimation of LSST Supernovae,” *Monthly Notices of the Royal Astronomical Society,* vol. 477, no. 3, pp. 4142-4151, 2018.

[7] “DIRAC Interware,” [Online]. Available: http://diracgrid.org/.

[8] “CoCo+Ganga Example Scripts - GitHub,” [Online]. Available: https://github.com/lsst-uk/GangaExample.

Appendix A: Example Ganga Submission Script

```
import argparse
import os

def parse_command_line(description=("CoCo submission script")):
    """
    Parser of command line arguments
    """

    parser = argparse.ArgumentParser(
        description=description,
        formatter_class=argparse.ArgumentDefaultsHelpFormatter)

    parser.add_argument("-e", "--eups", help="EUPS tarball location",
                        default='/lsst/user/d/darren.white/eupscoco.tar.gz')

    parser.add_argument("-s", "--spec", help="Spectra tarball location",
                        default='/lsst/user/d/darren.white/spectra.tar.gz')

    parser.add_argument("-i", "--infiles", nargs='+',
                                                help="List of input files")

    parser.add_argument("-n", "--nlcs", default='1000',
                                    help="Number of required light curves "
                                    "per input file")

    parser.add_argument("-l", "--local", action="store_true",
        help="Flag to run locally. WARNING: Must set local EUPS/Spec "
             "file if set.")

    return parser.parse_args()

def submit(arglist):
    j = Job(application=Executable())

    # Set the application executable (this also contains EUPS setup steps)
    j.application.exe = File('CoCoJobSplit.sh')

    # Set up splitter to generate subjobs for each job
    j.splitter = GenericSplitter()
    # Set arguments and individual input files for each subjob, assign
    # to splitter (makes subjobs across args/infiles pairs) - NOTE:
    # inputfiles here should be path strings, not Local/DiracFile as
    # below.
    appargs = [[os.path.basename(args.eups),
                os.path.basename(args.spec),
                os.path.basename(fname),
                arglist.nlcs] for fname in arglist.infiles]

    infiles = [[fname] for fname in arglist.infiles]

    j.splitter.multi_attrs = { "application.args": appargs,
                               "inputfiles": infiles}

    # Set backend to GridPP if local flag not set.
    # Set input files sent to each worker node (need to be defined as
    # LocalFile(), DiracFile(), etc depending on where job is being run)
    if arglist.local:
        j.inputfiles = [LocalFile(args.eups), LocalFile(args.spec)]
    else:
        j.inputfiles = [DiracFile(lfn=args.eups), DiracFile(lfn=args.spec)]
        j.backend = Dirac()

    # Define output file wildcard from script and where to put it after
    # completion LocalFile brings back to local machines Ganga directory
    j.outputfiles = [LocalFile("*.dat")]

    j.submit()

    return

args = parse_command_line()
submit(args)
```

https://github.com/lsst-uk/GangaExample/blob/master/CoCoGangaSplit.py.

Appendix B: Example Bash Job Script

```bash
#!/usr/bin/env bash
start=`date +%s`

# Echo all commands
set -o xtrace

# Run parameters set via Ganga
EUPS=$1
SPECTRA=$2
INFILE=$3
NLCS=$4

# Retrieving code from GitHub - git not installed on some nodes, so use wget
echo "Retrieving CoCo from GitHub"
wget https://github.com/UoS-SNe/CoCo/archive/gridPP.zip
unzip gridPP.zip
mv CoCo-gridPP CoCo

# Unpack tarballs containing Spectra and EUPS directories
echo "Unpacking EUPS and Spectra tarballs into CoCo directory"
tar -xzf $(basename $EUPS) --directory CoCo 2>&1
tar -xzf $(basename $SPECTRA) --directory CoCo 2>&1

# Copy files to correct locations in CoCo directory
cp $INFILE CoCo
cd CoCo
cp GridPP/CoCosim2.py ./
cp GridPP/sim_functions.py ./

# Use EUPS to set up prerequisites
echo "Setting up EUPS"
source /cvmfs/lsst.opensciencegrid.org/fnal/products/eups/bin/setups.sh
MY_EUPS=$PWD/eups
export EUPS_PATH=$MY_EUPS:/cvmfs/lsst.opensciencegrid.org/fnal/products/
echo $EUPS_PATH
echo "Setup required modules"
echo "gcc v4_9_2"; setup gcc v4_9_2
echo "gsl"; setup gsl
echo "numpy v1_9_1"; setup numpy v1_9_1
echo "setuptools"; setup setuptools
echo "minuit2"; setup minuit2
echo "scipy v0_14_0"; setup scipy v0_14_0
echo "python v2_7_8"; setup python v2_7_8

# Make and build CoCo libraries
echo "Building CoCo"
make
python setup.py build_ext --inplace

# Run simulation script for input file
echo "Running on" `date`
echo "Running CoCo: python CoCoSim2.py "
python CoCosim2.py $(basename $INFILE) $NLCS
echo "DONE"
```

https://github.com/lsst-uk/GangaExample/blob/master/CoCoJobSplit.sh.