CODE-RADE a user centric system for delivery of science software.

S H T Murray[1] and B Becker[2]

[1] Center for High Performance Computing, CSIR, Rosebank, Cape Town, South Africa
[2] Meraka Cyberinfrastructure, CSIR, Lynwood, Pretoria, , South Africa

E-mail: [1]smurray@csir.co.za

E-mail: [2]bbecker@csir.co.za

Abstract. Scientific computing can be distilled to the execution of sets of applications that are combined into complex workflows. Due to both the complexity and number of scientific packages and computing platforms, delivering these applications to end users has always been a significant challenge through the grid era, and remains so in the cloud era. In this contribution we describe a platform for user-driven, continuous integration and delivery of research applications in a distributed environment - project CODE-RADE. Starting with 6 hypotheses describing the problem at hand, we put forward technical and social solutions to these. Combining widely-used and thoroughly-tested tools, we show how it is possible to manage the dependencies and configurations of a wide range of scientific applications, in an almost fully-automated way. Finally, we will give some insight into how this platform could be extended to address issues of reproducibility and collaboration in scientific research in Africa.

1. Introduction

Since the first computing program exceeded the capabilities of its original execution host, and was migrated to another platform, scientists have squandered countless hours attempting to get programs to run as opposed to actually doing the science they intended to do.

1.1. The paradox of plentiful computing

It is a central theme of scientific computing that there are never enough resources available to any particular researcher. Congruent to this, there has been massive and near-pervasive investment in HPC and distributed computing resources. There has been a coordinated national effort in South Africa to build a well-connected distributed computing platform[1], fully integrated and connected to the national research network[2]. This effort has even been extended on a trans-continental scale, via the signature of a memorandum of understanding between African and European infrastructure providers[3]. With the creation of the Africa-Arabia Regional Operations Centre[1], several sites across Africa are available to researchers to execute their scientific workflows, and support several different virtual organisations, or communities of practice. Despite this effort in interoperability and integration, many of these resources are under-utilised at any given time. This seems to point to a paradox of scientific computing, where there is an over-provision of resources, yet scientists are still impeded from working.

[1] See "http://www.africa-grid.org" for details

1.2. Barriers to Entry

This situation exists due to various barriers to entry. One of the main benefits of a distributed computing platform is also one of the main barriers to entry : the complexity of the environment. Researchers seldom need to take into account the configuration of their application or workflow for different platforms, and it is practically unfeasible to insist on homogeneous hardware and configurations across the sites. This results in significantly increased overhead on the part of the researcher to ensure that their application will properly execute at any of the given resources, which thus leads to oversubscription at certain sites and underutilisation at others[2].

CODE-RADE aims to reduce or remove the barrier to entry for new applications, thus eliminating the paradox of plentiful computing.

2. CODE-RADE Design and Philosophy

The porting and integration of new applications is ordinarily done by site administrators who attempt to compile and install the researcher's applications according to the local procedures and policies. This usually leads to a significant lead-time and depends critically on the ability of the site administrator to interpret the needs of the researcher. There can also be much ambiguity regarding the resolution of dependencies, optimisation and versions. We then come up against a tension between domain expertise and administrative privileges. The *researcher*, not the site administrator is usually the one that best understands how to compile the application. The site administrator has the knowledge of the local setup which would integrate the application appropriately and efficiently. This overhead is multiplied by the number of distinct application stacks which are in use in scientific domains, and exacerbated by the multitude of dependencies and optimisation configurations. For the physical sciences domain, this can be appreciated by considering the wide array of applications registered in the EGI Applications Database[4]. [3]

2.1. CODE-RADE Design Hypotheses

CODE-RADE provides a means to transparently integrate a new application into an arbitrary site configuration, that is both user-driven and satisfies the constraints imposed by site administrators. We reconsidered the way applications are built and deployed in scientific research and computational infrastructure. This was done as far as possible from first principles, by constructing seven hypotheses of bringing applications to distributed infrastructure. In what follows, we discuss the considerations of these design hypotheses and how they have influenced the development and results of the project. **1: Every researcher is an application.** The core aspect of initial communication between a research infrastructure provider and a researcher is the *application*. **2: No software is an island.** Every scientific application has a dependency list. No definition of an application can be complete without an explicit expression of the list of its dependencies, which should extend as far as possible down userspace. **3: All applications need an environment.** As with any organisms, biological or digital, it requires an environment in which to express its functionality. **4: There is more than one environment.** By restricting ourselves to a single environment, we lose main advantages of distributed systems, and erect unncessary barriers to entry for researchers. **5: Solutions Decay.** A "solution" in this case refers to a software expression of the method for creating an executable version of the application. The exploration of these solutions, including the attempts which failed, are important components of the system knowledge. **6: Humans need not apply.** If we assume that the previous hypotheses are true, it stands to reason that most of the work to actuall

[2] Another effect is the reduction in the capacity of the researcher and experts at sites to effectively collaborate with each other, due to geographical and technical divergence.

[3] For physics alone, there is a complexity and scale which is prohibitive to any particular site administrator. Considers the applications maintained in the Homebrew or EasyBuild repositories - about 600 and 1000 respectively - the situation seems untenebale.

compile and integrate applications can be automated. **7: This is not hard.** While it may seem that we are proposing a radical shift from common practice, which may seem discouraging, the fact is that continuous integration[5] and continuous delivery[6] has been common practice in the software development world for several years.

Along with these hypotheses, CODE-RADE attempts to solve problems which impede fluid collaboration and exploitation of computational resources. First of all, the system aims to reduce or remove entire barriers to entry represented by user applications, through the use of continuous integration and delivery, automated builds and version-controlled source code repositories. This allows all involved to know exactly where the application is in its lifecycle, and indeed which build of the application is currently being used.

Automated continuous delivery ensures that once applications are integrated, they are immediately available at sites which wish to host them, further reducing the workload on site administrators.

3. CODE-RADE in detail

3.1. Actors

The CODE-RADE platform identifies four kinds of actors which have distinct roles to play : Researchers, Research Software Engineers (RSE), Infrastructure Operators (Ops), Automated Agents (Bots).

Researchers are the final users of the application, they are not necessarily the authors of the software. Researchers may not know the details necessary to compile and optimise applications for specific environments, they are adept in interpreting the results of these applications and are the end consumers of them. The research software engineer, which in some cases is the researcher themselves, thus provides a means to express the application on the various targets in an executable state. This proves to the resource providers that applicatons can indeed execute properly on target infrastructures is needed before they will agree to actually host and excute the applications on their sites. Indeed, the characteristics of these target infrastructures are defined by the infrastructure operations team, which know them best. The role of the automtaed agents is to execute the code provided by the research software engineers and see whether it passes the tests set by the infrastructure operations team, at every commit pushed to the repository. This provides a direct link between change in the software and expression of that software (source code version linked to executable version), and since the criteria of the tests are predefined, once they pass applications can be integrated without human intervention.

The entire workflow can be driven by researchers themselves, once setup.

3.2. Components

The implementation of CODE-RADE consists essentially of three generic components : Continuous integration tool,Version controlled software repository,Automated software delivery tool. There are several choices which one could use to build a similar platform to CODE-RADE by choosing a particular tool to fulfill each component. In our case, we have selected Jenkins[7], GitHub[8] and CERN Virtual Machine Filesystem[9]. The reasoning and justification for these specific choices are given in some more detail below in section 4. The components have specific roles to play in each phase of the CODE-RADE workflow, which we describe next.

3.3. CODE-RADE workflow

Although the main aim of CODE-RADE is to ensure that applicaitons are available for execution on arbitrary remote sites, there are collateral issues which have importance to a community. We identify several goals in CODE-RADE which have their own flow of actions and we thus define the generic workflow for application **integration, delivery, review** and **publication**.

The most common workflow is 3 steps : Build, Test, Deliver. The first step (build) refers to the synchronisation of the source code with relevant updates. It is initiated by the Research Software Engineer: a commit is made to a repository, detailing a change, which the RSE authors. This triggers an event on Jenkins, which pulls the latest commit and executes the scripts in the repository which will build and test the code. The second step (Test) is conditionally executed on the success of the previous step. This step checks the viability of the built application and executes whatever tests are specified, then installs the application into the integration environment. Finally the third step (Deliver) cleans the built application and installs it into the deploy environment, then ships the executable code to the repository for subsequent replication.

Although there are several tools for building widely-used applications (see Section 4 below), we have chosen a convention within the project to have three scripts which perform the compilation, functional testing and integration into the delivery environment respectively.

Each job, as defined on Jenkins, consists of at least these three tasks. If each passes successfully, a job to update the CVMFS repository is triggered, which puts the repository into transaction, pulls in the newly built applications and module files, then publishes a new version of the repository. This is automatically and transparently available at sites after a few seconds, once the CVMFS client detects the change in the repository hash. In order to make it easier for end users or client applications to check which version of the repository they are using[4], an integer increment is added to a file in the repository which is linked to the job which resulted in the latest version.

In short, a commit to a repository by a researcher or RSE will result in the application being automatically delivered to arbitrary endpoints, as long as quality and functional tests set by Ops and researchers respectively are passed.

4. Discussion

Our particular combination of existing tools (Github, Jenkins and CVMFS) has resulted in a powerful platform with which to exploit any existing computational infrastructure, in an automated user-driven way. Some aspects which together make CODE-RADE a novel and powerful platform are highlighted.

4.1. CODE-RADE is cross-platform

CODE-RADE takes a software description of a scientific tool and builds a digital expression which can be executed on arbitrary architectures. This makes CODE-RADE different from more traditional package managers, since these only build for the architecture on which the user is currently. By simulating the execution environment which may in principle be available at geographically distributed sites.

We currently define a target according to a matrix of characteristics, which are encoded as variables in the build and deploy environments : **ARCH**: The CPU architecture of i the execution environment, **OS**: The operating system of the execution environment, **SITE**: A meta-variable encompassing aspects of specialised hardware (GPU, Myrinet, GPFS, etc) and other aspects (software licenses, available middleware etc.)

For each combination of these axes, specific optimisations can be applied, ensuring that the final result (compiled application) is both properly compiled, and will execute efficiently on the target site. By building on as many environments as possible, for as many combinations as possible, CODE-RADE promotes diversity in computing platforms, without exacerbating the paradox of plentiful computing.

[4] This can be determined more accurately using the cvmfs client tools, but this may not be intuitive to users.

4.2. CODE-RADE is atomic
Successfully built git hashes are linked to a cvmfs hash, permitting reproducibility across sites and allowing one to link research outputs to the underlying versions and builds. This is then citable via Zenodo digital object identifiers (doi) from the cvmfs hash, permitting the work done on the infrastructure to be citable and credit given, and critically be reproducible. The researcher has fine-grained control over dependencies, version, and targets that are built and built against.

4.3. CODE-RADE is community-based
There is no restriction on the applications that can be integrated, so long as the software is accesible online in someway. Anyone can contribute applications, resources, code review, testing, or where ever their skill sets lie. The entire community can see the build status and reinitiate failed builds via new pull requests. After a user updates their git repository, a git pull request is done, a build is executed. The results of the build are publicly viewable (`https://ci.sagrid.ac.za`), errors are viewable by the user doing the pull request (and everybody else), via a full build transcript. This can be fixed by merely redoing a pull request with what ever changes are required to fix the errors. This is completely indepdent of staff at sites.

4.4. CODE-RADE is Automated
There is a heavy relilance on automation in CODE-RADE, via automated agents to reduce bias, lead time and is user-driven. As an example, failures on jenkins cause issues on github, and the user is then able to fix these at their discretion. Successful subsequent builds will also close these issues. Notifications are sent to relevant slack channels [5] and bots within slack permit actions to be taken, for instance postponing issues, reassigning, or closing.

5. Summary, Conclusion and Further work
Making the best use of computational infrastructure comes down to running applications. Maintaining and porting them is a hard and often thankless task. We think we have come up with a solution to solve many parts of this and making the work citable. We've built an automated porting system, which will deliver functional, tested, relevant software to your site that is reproducible. The work done is then citable to give credit where credit is due for work that often goes uncredited. It eases the communication blocks in collaborative work, often highly distributed. Everyone is welcome to come on in and help us build it.

Acknowledgements
The authors would like to thank contributors to the project for ideas, discussion and code The original idea co-developed by one of the authors (BB) and Fanie Riekert (University of the Free State). Input and critique was provided by Dane Kennedy and Sakhile Masoka (Centre for High Performance Computing, Cape Town) and Peter van Heusden (South African National Bioinformatics Institute). The authors recognise the technical support of the EGI CVMFS Task Force support, in particular Catalin Condurache (European Grid Initiative and Science and Technology Facilities Council, UK). Several discussions on CODE-RADE design and extension were had with Timothy Carr (University of Cape Town e-Research Centre).

CODE-RADE is supported by the Sci-GaIA project under grant 654237 of the European Commission's Horizon 2020 programme

[5] See `"https://www.slack.com"`

References

[1] B Becker et al. "The South African National Compute Grid" *IST-Africa 2009 Conference Proceedings*, ed P Cunningham, M Cunningham.

[2] "SANReN South African National Research Network" [online] Available at: `http://www.sanren.ac.za/`. [Accessed 25 Feb 2018].

[3] S Andreozzi "Mou between EGI.eu and CSIR Meraka (African-Arabian Region)", Mou between EGI.eu and CSIR Meraka (African-Arabian Region), Apr 2015. [online] Available at: `https://documents.egi.eu/public/ShowDocument?docid=2407`. [Accessed: 25 Feb 2018].

[4] "EGI Applications Database", EGI Applications Database. [online] Available at: `https://appdb.egi.eu`. [Accessed: 25 Feb 2018].

[5] P Duvall *Continuous Integration.* 2007. Pearson Educational Publishers.

[6] J Humble, D Farley *Continuous delivery: reliable software releases through build, test, and deployment automation* 2010 Addison-Wesley Longman.

[7] J Smart *Jenkins: The Definitive Guide* 2011 OReilly.

[8] See https://developer.github.com/v3

[9] J Blomer, P Buncic, and T Fuhrmann "CernVM-FS: Delivering Scientific Software to Globally Distributed Computing Resources," *Proceedings of the First International Workshop on Network-aware Data Management, New York, NY, USA*, 2011, pp. 4956.