Project Acronym: MuG

Project title: Multi-Scale Complex Genomics (MuG)

Call: H2020-EINFRA-2015-1

Topic: EINFRA-9-2015

Project Number: 676556

Project Coordinator: Institute for Research in Biomedicine (IRB Barcelona)

Project start date: 1/11/2015

Duration: 36 months

Milestone 19: Programming models release

Lead beneficiary: Barcelona Supercomputing Center (BSC-CNS)

Dissemination level: PUBLIC

Due date: 01/09/2017

Actual submission date: 29/08/2017

Document history

Version	Contributor(s)	Partner	Date	Comments
0.1	Javier Conejero	BSC	18/07/2017	First draft
0.2	Javier Conejero	BSC	27/07/2017	Added improvements descriptions
0.3	Javier Conejero	BSC	17/08/2017	Extended and added improvement descriptions
0.4	Rosa M Badia	BSC	24/08/2017	Internal review
			29/08/2017	Final version

Table of contents

1 INTRODUCTION

COMPSs [1, 2] is the programming model provided by the Virtual Research Environment (VRE) [3] within the MuG project [4]. It is known for notably improving the performance of large scale applications (written in Java, C and Python) by automatically parallelizing their execution. COMPSs is developed by the Workflows and Distributed Computing team at the Barcelona Supercomputing Center (BSC) [5], and the latest release includes useful and interesting features for the MuG applications.

The main challenge for the programming model within the MuG project has been to identify the main requirements of the project applications that were not supported by PyCOMPSs (COMPSs binding for Python applications), and extend PyCOMPSs in order to enable MuG application developers to parallelize their existing Python applications and to implement new applications exploiting the new features. The latest outcome of this work has been the recent PyCOMPSs/COMPSs 2.1 release, published in June 2017.

This document presents the programming model release for the VRE computational infrastructure, focusing on the main features that have been implemented with regard to the MuG applications requirements.

2 PROGRAMMING MODELS RELEASE

The Workflows and Distributed Computing team at BSC announced the new PyCOMPSs/COMPSs programming model release, version 2.1 (codename Bougainvillea), in June 2017.

This version of PyCOMPSs/COMPSs updates the result of the team's work in the last years on the provision of a set of tools that helps developers to program and execute their applications efficiently on distributed computational infrastructures, such as: Clusters, Grids and Clouds.

The new release comes with a wide variety of new features, from which some of them have been implemented focused on the MuG applications requirements and future potential improvements. In particular, this release updates PyCOMPSs/COMPSs with the following main features (emphasizing in bold the ones that benefit the MuG project):

- Runtime:
  - New annotations to simplify tasks that call external binaries.
  - **Integration with other programming models (MPI, OmpSs, etc.).**
  - **Support for Singularity containers in Clusters.**
  - **Extension of the scheduling to support multi-node tasks (MPI apps as tasks).**
  - Support for Grid Engine job scheduler in clusters.
  - Language flag automatically inferred in runcompss script.
  - New schedulers based on tasks' generation order.
  - **Core affinity and over-subscribing thread management in multi-core cluster queue scripts (used with MKL libraries, for example).**
- Python:

  - **@local annotation to support simpler data synchronizations in master.**
  - **Support for args and kwargs parameters as task dependencies.**
  - **Task versioning support in Python (multiple behaviours of the same task).**
  - **New Python persistent workers that reduce overhead of Python tasks.**
  - Support for task-thread affinity.
  - Tracing extended to support for Python user events and HW counters (with known issues).
- Tools:
  - Visualization of not-running tasks in current graph of the COMPSs Monitor

Focusing on the MuG related features, the new release provides a new mechanism for the tasks which invoke binaries. To this end, three new decorators (*@binary*, *@mpi* and *@ompss*) have been created and can be used over the *@task* decorator. These new decorators enable a better integration with programming models such as MPI or OmpSs. Consequently, a PyCOMPSs workflow can be composed of sequential tasks, multi-threaded tasks (with OmpSs) and multi-node tasks (with MPI). The COMPSs schedulers have been extended accordingly to support these new tasks' types.

Also, PyCOMPSs has been improved to support *args* and *kwargs* parameters as task dependencies (this feature enables to call tasks with variable number of parameters as can be done in pure Python code). Two decorators have been provided: for synchronization (*@local*) and for task versioning (*@implements*). The *@local* decorator is intended to provide simpler data synchronization in the code run by the main process when invoking both tasks and non-task functions. This decorator can be used to annotate non-task functions and this indicates to PyCOMPSs that it should perform the necessary data synchronizations over the parameters used by previously invoked tasks (i.e., transferring results of tasks generated in remote nodes to the computing node where the @local task is executed). Additionally, the *@implements* decorator enables developers to define multiple implementations of the same task (which can be annotated with the @constraint annotation). The runtime selects the version that is more appropriate given the available resources.

With regard to PyCOMPSs internals, it has been improved by adding persistent workers. Consequently, the worker overhead is reduced since it is not necessary to start a new Python interpreter for each task, enabling to get better performance with fine grain tasks.

COMPSs has also been extended with a better support for Intel MKL based libraries, considering two-levels of parallelism that can be combined: task-level and thread level.

In addition to all these new features, PyCOMPSs has also been integrated with Jupyter notebook. This integration is intended to enable users to develop and test PyCOMPSs applications from Jupyter notebook. And with regard to containers interoperability, this new release is able to deal with Singularity.

PyCOMPSs previous version 2.0 (codename Amapola), released in November 2016, also included some interesting features that MuG developers can benefit from, such as: an explicit barrier (synchronization API call without data synchronization), performance improvement due to the upgrade to Java 8, and scheduling improvements.

Moreover, some improvements have also been performed, such as: Improved PyCOMPSs serialization and several bug fixes. The serialization improvement has a direct impact on the performance (since the serialization requires to transform the objects to be used among tasks into a byte writable format),

and may become important within the MuG applications since the objects used among tasks may have an important size. Moreover, several bug fixes have been applied, improving the stability during the execution.

This COMPSs release has also been updated within the existing tutorial VM, which enables the MuG application developers to understand and play with the programming model in a ready and controlled environment (using an IDE or Jupyter notebook). Moreover, it allows debugging the applications with tests and/or small datasets.

The COMPSs programming model has been successfully deployed within the MuG Cloud infrastructures: BSC and EBI. Applications developed with the current release can be deployed within the infrastructure in order to launch them from the VRE and benefit from the new improvements.

The COMPSs 2.1 packages, as well as the updated manuals and tutorial VM can be downloaded from the COMPSs webpage (6).

3 REFERENCES

(1) ServiceSs: an interoperable programming framework for the Cloud, Journal of Grid Computing, March 2014, Volume 12, Issue 1, pp 67–91, Lordan, F., E. Tejedor, J. Ejarque, R. Rafanell, J. Álvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, and R. M. Badia, DOI: 10.1007/s10723-013-9272-5

(2) PyCOMPSs: Parallel computational workflows in Python, Enric Tejedor, Yolanda Becerra, Guillem Alomar, Anna Queralt, Rosa M. Badia, Jordi Torres, Toni Cortes, Jesús Labarta, IJHPCA 31(1): 66-82 (2017), DOI: 10.1177/1094342015594678

(3) VRE (beta) – http://multiscalegenomics.bsc.es

(4) MuG project – http://multiscalegenomics.eu

(5) Barcelona Supercomputing Center (BSC) – http://www.bsc.es

(6) COMPSs project – http://www.bsc.es/compss