ExaQUte

Exascale Quantification of **U**ncertainties for **Te**chnology and Science Simulation

First public Release of the solver

Document information table

Contract number:	800898
Project acronym:	ExaQUte
Project Coordinator:	CIMNE
Document Responsible Partner:	CIMNE
Deliverable Type:	Software release
Dissemination Level:	Public
Status:	Final version

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No **800898**

Authoring

Prepared by: CIMNE, BSC and EPFL				
Authors	Partner	Modified	Version	Comments
Quentin Ayoul-Guilmard	EPFL	All	1.0.0	XMC
Rosa M. Badia	BSC			PyCOMPSs
Jorge Ejarque	BSC			PyCOMPSs
Sundar Ganesh	EPFL			XMC
Fabio Nobile	EPFL			XMC
Marc Nuñez	CIMNE			Kratos Multiphysics
Cecilia Soriano	CIMNE			Coordination
Carlos Roig	CIMNE			Kratos Multiphysics
Riccardo Rossi	CIMNE			Kratos Multiphysics, XMC
Riccardo Tosi	CIMNE			Kratos Multiphysics, XMC

Change Log

Versions	Modified Page/Sections	Comments
1.0.0	All	Submitted version

Approval

Aproved by:				
	Name	Partner	Date	OK
Task leader	Riccardo Rossi	CIMNE	29/05/2020	OK
WP leader	Riccardo Rossi	CIMNE	29/05/2020	OK
Coordinator	Riccardo Rossi	CIMNE	29/05/2020	OK

Executive summary

This deliverable presents the software release of Kratos Multiphysics, together with the XMC library, Hyperloom and PyCOMPSs API definition [8]. This report is meant to serve as a supplement to the public release of the software. Kratos is "a framework for building parallel, multi-disciplinary simulation software, aiming at modularity, extensibility, and high performance. Kratos is written in C++, and counts with an extensive Python interface". XMC is a python library for hierarchical Monte Carlo algorithms. Hyperloom and PyCOMPSs are environments for enabling parallel and distributed computation.

Table of contents

Nomenclature / Acronym list

Acronym	Meaning
API	Application Program Interface
Kratos	Kratos MultiPhysics
UQ	Uncertainty Quantification

1 Introduction

The Kratos software [5, 6] is designed for dealing with a multitude of different physical problems, spacing from computational fluid dynamics to convection diffusion, from structural applications to fluid structure interaction. For further details we refer to the Kratos documentation.

One of the most recent capabilities of Kratos is Uncertainty Quantification. UQ is the field of science which studies how uncertainties evolve and propagate in a problem characterized by random parameters, and how the output quantities of interest are affected by the randomness. Even though within the ExaQUte project we are interested in applying UQ to wind engineering problems, where wind velocity is the random quantity, one can easily observe that UQ range of application is much wider, and spaces from engineering to physics and finance. UQ methods have been developed within the XMC library [1], and Kratos is the default solver software of the library. Further details about the XMC library can be found in [2].

Moreover, efficiency is an important factor when dealing with large problems. For this reason, both XMC and Kratos are designed for large scale computing in distributed environments. The scheduling in distributed environments is handled by external libraries: Hyperloom [4] and PyCOMPSs [3, 9, 11]. XMC presents a natural integration with these libraries, making the library particularly suitable for running in supercomputers. Due to this inner integration of XMC with Hyperloom and PyCOMPSs, the API definitions of the schedulers have been integrated in the software release.

Therefore, the software release [8] contains Kratos as solver, XMC as UQ library and Hyperloom and PyCOMPSs API definitions.

2 Software structure

Within Kratos, different applications are organized in different folders. One folder is entirely dedicated to UQ and it is called *MultilevelMonteCarloApplication*. The folder is located in:

```
Kratos/applications/MultilevelMonteCarloApplication
```

This folder contains XMC in:

```
Kratos/applications/MultilevelMonteCarloApplication/external_libraries/
   XMC
```

and Hyperlooms and PyCOMPSs API definitions in:

```
Kratos/applications/MultilevelMonteCarloApplication/external_libraries/
   PyCOMPSs/exaqute
```

Examples of usage and tests checking the correct integration between the different libraries are also included.

3 Examples

Different algorithms are developed and can be run with this software release: Monte Carlo, Multilevel Monte Carlo, Adaptive Monte Carlo, Adaptive Multilevel Monte Carlo, Con-

tinuation Multilevel Monte Carlo, Asynchronous Monte Carlo, Asynchronous Multilevel Monte Carlo [10].

Multilevel algorithms are designed to run with different hierarchy strategies: stochastic adaptive refinement, deterministic adaptive refinement or standard uniform refinement. To perform adaptive refinement, the MMG software [7] is exploited for remeshing. To remesh, a metric is a required input to MMG. Different metrics are developed within Kratos, and the one exploited in the examples is based on the hessian of the numerical solution. We refer to the Kratos MMG tutorial and MMG examples for further details.

Examples are located in:

```
Kratos/applications/MultilevelMonteCarloApplication/external_libraries/
   XMC/examples
```

These examples are configurable through a JSON file, which is located in the *parameters* folder (inside the examples folder). Different configuration files are present and available as a reference to run Monte Carlo, Multilevel Monte Carlo or their asynchronous counterparts. For example, to run a standard Monte Carlo example, one should change to the *examples* folder and execute

```
$ python3 run_mc_Kratos.py parameters/
   parameters_xmc_test_mc_Kratos_poisson_2d.json
```

while to run a Multilevel Monte Carlo example one should execute

```
$ python3 run_mlmc_Kratos.py parameters/
   parameters_xmc_test_mlmc_Kratos_poisson_2d.json
```

By default, such examples run in serial. If one is interested in running exploiting distributed computations, Hyperloom and/or PyCOMPSs must be installed in the machine. For example, to run the standard Monte Carlo example with COMPSs, one should execute:

```
$ runcompss --options=value run_mc_Kratos.py parameters/
   parameters_xmc_test_mc_Kratos_poisson_2d.json
```

However, the user must note that the appropriate imports have to be changed in places marked through XMC from:

```
from exaqute.ExaquteTaskLocal import *
```

to

```
from exaqute.ExaquteTaskHyperLoom import *
```

or

```
from exaqute.ExaquteTaskPyCOMPSs import *
```

For further details about running in distributed environments, we refer to the MultilevelMonteCarloApplication documentation and to the COMPSs and Kratos tutorial.

4 Current results

The work developed and released has been run and tested with several benchmarks, which have been run in different supercomputers. These benchmarks include potential flow, low

and high Reynolds computational fluid dynamics and structural problems. The software has been tested up to 128 working nodes (6144 CPUs), and showed a very good parallel efficiency. The work developed will be submitted to a peer reviewed journal.

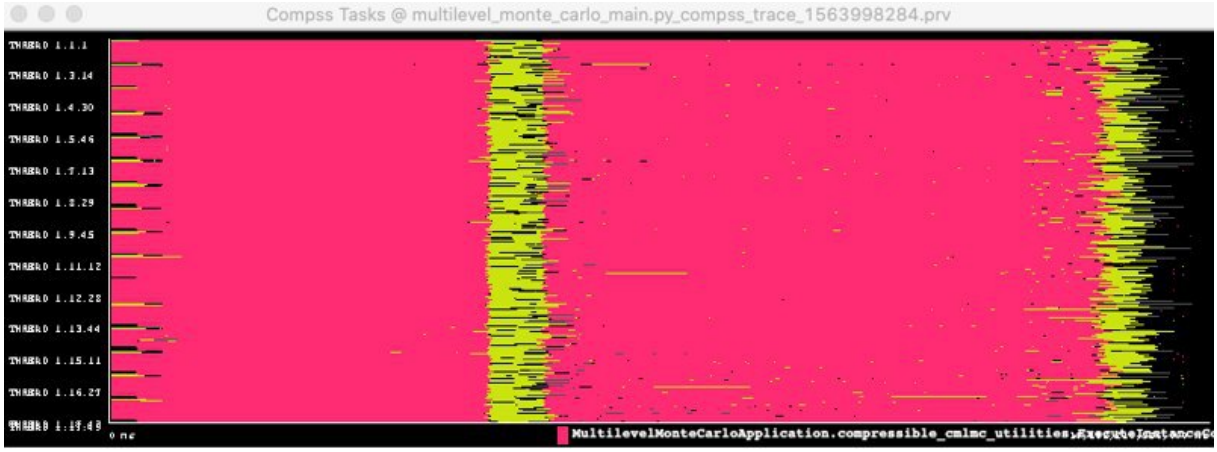

Figure 1: Execution trace of asynchronous Multilevel Monte Carlo algorithm, running with PyCOMPSs.

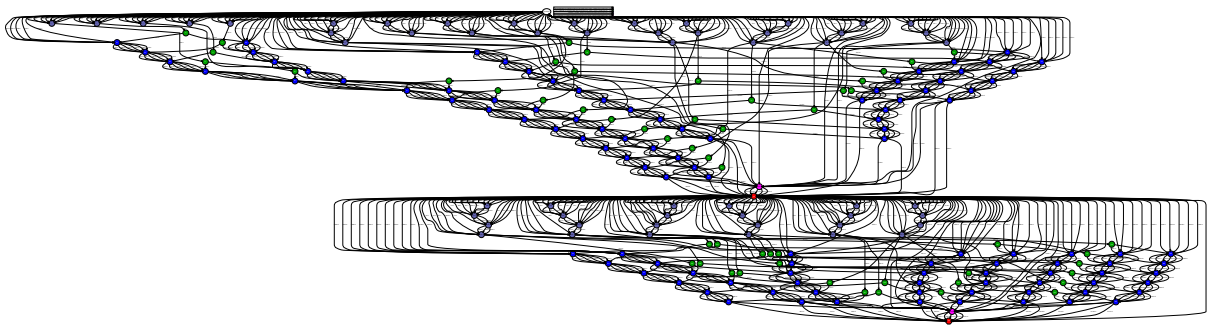

Figure 2: Graph connection of Multilevel Monte Carlo algorithm dependencies, running with PyCOMPSs.

References

[1] R. Amela, Q. Ayoul-Guilmard, R. M. Badia, S. Ganesh, F. Nobile, R. Rossi, and R. Tosi. ExaQUte XMC. May 2019. doi:10.5281/zenodo.3235833.

[2] R. Amela, Q. Ayoul-Guilmard, R. M. Badia, S. Ganesh, F. Nobile, R. Rossi, and R. Tosi. D5.2 Release of ExaQUte MLMC Python engine, May 2019.

[3] R. M. Badia, J. Conejero, C. Diaz, J. Ejarque, D. Lezzi, F. Lordan, C. Ramon-Cortes, and R. Sirvent. COMP superscalar, an interoperable programming framework. *SoftwareX*, 3–4, 12 2015. doi:10.1016/j.softx.2015.10.004.

[4] V. Cima, S. Böhm, J. Martinovič, J. Dvorskỳ, K. Janurová, T. V. Aa, T. J. Ashby, and V. Chupakhin. Hyperloom: A platform for defining and executing scientific pipelines in distributed environments. In *Proceedings of the 9th Workshop and 7th Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms*, pages 1–6. ACM, 2018.

[5] P. Dadvand, R. Rossi, and E. Oñate. An object-oriented environment for developing finite element codes for multi-disciplinary applications. *Archives of computational methods in engineering*, 17(3):253–297, 2010.

[6] P. Dadvand, R. Rossi, M. Gil, X. Martorell, J. Cotela, E. Juanpere, S. R. Idelsohn, and E. Oñate. Migration of a generic multi-physics framework to HPC environments. *Computers and Fluids*, 80(1):301–309, 2013. ISSN 00457930. doi:10.1016/j.compfluid.2012.02.004.

[7] C. Dapogny, C. Dobrzynski, and P. Frey. Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. *Journal of Computational Physics*, 262:358–378, apr 2014. ISSN 10902716. doi:10.1016/j.jcp.2014.01.005.

[8] V. M. Ferrándiz, P. Bucher, R. Rossi, jcotela, J. Maria, R. Zorrilla, M. A. Celigueta, G. Casas, C. Roig, A. C. Velázquez, P. Dadvand, S. Latorre, J. I. González, I. de Pouplana, miguelmaso, M. Núñez, F. Arrufat, dbaumgaertner, B. Chandra, A. Ghantasala, armingeiser, S. Warnakulasuriya, lluís, J. Gárate, MFusseder, Pablo, AFranci, L. Gracia, thomas, K. B. Sautter, and R. Tosi. KratosMultiphysics/Kratos: Release 8.0, May 2020. URL `https://doi.org/10.5281/zenodo.3234644`.

[9] F. Lordan, E. Tejedor, J. Ejarque, R. Rafanell, J. Álvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, and R. M. Badia. ServiceSs: An Interoperable Programming Framework for the Cloud. *Journal of Grid Computing*, 12(1):67–91, 2014. ISSN 15707873. doi:10.1007/s10723-013-9272-5.

[10] M. Pisaroni, F. Nobile, and P. Leyland. A Continuation Multi Level Monte Carlo (C-MLMC) method for uncertainty quantification in compressible inviscid aerodynamics. *Computer Methods in Applied Mechanics and Engineering*, 326:20–50, 2017. ISSN 00457825. doi:10.1016/j.cma.2017.07.030.

[11] E. Tejedor, Y. Becerra, G. Alomar, A. Queralt, R. M. Badia, J. Torres, T. Cortes, and J. Labarta. PyCOMPSs: Parallel computational workflows in Python. *International Journal of High Performance Computing Applications*, 31(1):66–82, 2017. ISSN 17412846. doi:10.1177/1094342015594678.