Adaptive Execution Planning in Biomedical Workflow Management Systems

Marta Jaros*
Brno University of Technology,
Faculty of Information Technology,
Centre of Excellence IT4Innovations
Brno, Czech Republic
martajaros@fit.vutbr.cz

Bradley E. Treeby
University College London, Medical
Physics and Biomedical Engineering,
Biomedical Ultrasound Group
London., United Kingdom.
b.treeby@ucl.ac.uk

Jiri Jaros
Brno University of Technology,
Faculty of Information Technology,
Centre of Excellence IT4Innovations
Brno, Czech Republic
jarosjir@fit.vutbr.cz

KEYWORDS

Workflow management system, Automated execution planning, Adaptive planning.

ACM Reference Format:
Marta Jaros, Bradley E. Treeby, and Jiri Jaros. 2019. Adaptive Execution Planning in Biomedical Workflow Management Systems. In *Proceedings of Supercomputing Conference (SC'19).* ACM, New York, NY, USA, 3 pages.

1 INTRODUCTION

Biomedical workflow management systems enable clinicians to use grid, cloud and high performance computing (HPC) services easily. These systems describe complex problems, such as treatment planning, screening, and diagnosis, using workflows. Workflows can be seen as directed weighted graphs providing a formal way to define and automate multi-step computational procedures [5]. The graph nodes present individual tasks that may differ in their nature, performance and computational demands. They also encapsulate lower level details about the task specific parameters. Since HPC environments are highly dynamic and heterogeneous, efficient manual task execution, tuning to the specific computational machine, monitoring and dealing with various types of failures is very tedious and time consuming. Therefore, achieved cluster throughput may be very limited. The presented framework, called k-Dispatch, is trying to respond to this problem. k-Dispatch mainly focuses on computational problems related to biomedical environment and uses only predefined workflows. Although the workflow structures are predefined, they have a level of adaptivity based on the provided input data. The end users (clinicians) only submit input data and workflow specific parameters. To be compliant with medical security policies, only certified executables for given HW can be used.

Since the task run configuration strongly affects the final tasks mapping on the computational resources, cluster throughput and computational cost, the execution planning is of the highest priority. In order to plan the workflow execution, required code types are hard-coded in the structure of each supported workflow. There may be available several different binaries for each code type accessible only on some HPCs, selected queues and hardware architectures which they have been tuned to. Moreover, their cost factors may vary, which influences the final computational cost. The goals of workflow execution planning are to (1) satisfy the time-constrained result delivery requirements, and (2) minimize the computational cost.

Other similar tools like Pegasus [1], Globus [2] or Kepler [4] offer automated execution of scientific workflows on computational resources in a more general way. These tools focus on well-experienced users who provide workflows with data, binaries and run configurations. Contrary, k-Dispatch focuses on routine execution of predefined workflows by inexperienced users who want to reduce the computational cost and execution time. k-Dispatch achieves this by tuning the amount of computational resources assigned to individual tasks in the workflow. Since the scaling of individual tasks is never perfect, k-Dispatch may find such a workflow configuration even an experienced user would miss.

2 APPLICATION

k-Dispatch is being developed as a module of the k-Plan system. k-Plan performs a model-based treatment planning for ultrasound (US) therapy such as tissue ablation, neurostimulation and targeted drug delivery. The target position and US transducer parameters are defined in the treatment planning module via a medical GUI using patient-specific CT/MR images and therapeutic target. The predicted acoustic and thermal output is calculated using remote HPC resources. The task execution and data transfers are managed by the proposed dispatch server module. The successful treatment plan can eventually be exported to an ultrasound therapy device for patient delivery, see Fig. 1.

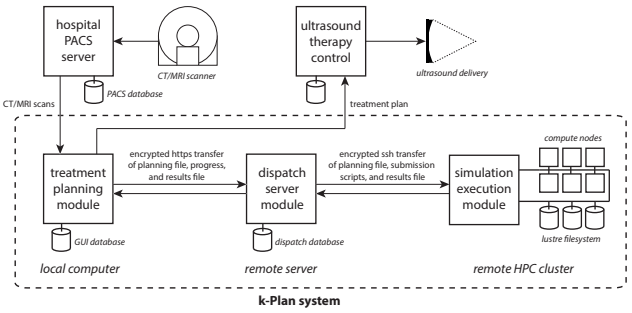

Figure 1: Architecture of the k-Plan system. k-Dispatch directly corresponds with the dispatch server module.

*corresponding author.

SC'19, 2019, Denver, CO, USA
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

3 WORKFLOW EXECUTION PLANNING

Currently, k-Dispatch only allows a static task planning similar to many related tools. For each task in the workflow, a default binary for a specific code type and a default run configuration are selected. This approach enables users to run their workflows without any advanced knowledge of the current HPC service, however, the tasks are likely not to be executed efficiently and may spend a long time in queues waiting for computational resources. The solution is to implement an adaptive execution planning and decision making process that selects the best run configuration for each task based on collected historical performance data and current cluster utilization. The first attempt is to implement a one-pass decision making process where the run configuration is optimized for each task independently. However, a multi-pass optimization process may, in the future, better optimize the run configurations for the whole workflow. Following constraints shall be taken into the consideration:

- Execution planning is time-constrained since the HPC environment is highly dynamic and changes very quickly.
- The workflow completion must not exceed a given time constraint. Thus, the amount of tasks waiting in queues due to unsuitable run configurations shall be limited.

Algorithm 1 describes the execution planning algorithm and its presumptions.

Algorithm 1: Adaptive execution planning algorithm

Presumptions:

1 Consider a set of allocations $A^+ \subseteq A$ the user can use.
2 All possible binary executables for $a \in A^+$ are defined as $D \in (B_1, B_2, \ldots, B_n)$, where n is the number of code types within the workflow. $B_i = \{b_1, b_2, \ldots, b_m\}$ is a set of available binaries for a given code type. B_i may be an empty set.
3 p is a price function returning the aggregated computational cost of the workflow. $p : G \times C \times D \rightarrow \mathbb{R}^+$.
4 t is a time function returning the aggregated execution time of the workflow. $t : G \times C \times D \rightarrow \mathbb{R}^+$.
5 Workflow evaluation is defined as $\mathbb{R}^+ \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$ and may be calculated using the formula $f = \alpha \cdot p + (1 - \alpha) \cdot t$ where α is a selectable ratio prioritizing the minimal computational cost or the execution time.
6 Best evaluated workflow is given by $\text{argmin}_{(c \in C, d \in D)} f$.

Algorithm :

1 Create a workflow $G = (V, E)$ from the workflow template and input data. V is a set of tasks and $E \subseteq V \times V$ is a set of task dependencies.
2 Select candidate allocations $C = \{c \in A^+ \mid c.status == active \wedge c.hours\_left > 0.0\}$
3 Generate and evaluate workflows for all combinations of candidate allocations C and binary executables D.

4 IMPLEMENTATION

k-Dispatch is a modular framework developed in Python. As shown in Fig. 2, k-Dispatch consists of the dispatch database, web server and dispatch core. The dispatch database holds data about users, executed workflows, computational resources, allocations, available binaries, etc. It also maintains performance data for various code types important for the adaptive execution planning process.

The web server arranges a secured http based communication with user applications. The dispatch core is the key component implementing the whole logic. It provides data transfers, execution planning, fault tolerance, monitoring and logging. Both, the web server and the dispatch database, may stand alone. Therefore, an outage of the dispatcher core does not limit users in uploading new jobs or downloading results.

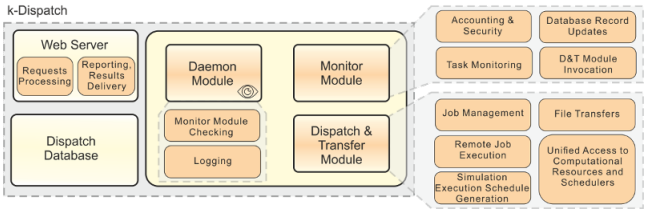

Figure 2: k-Dispatch architecture. k-Dispatch consists of the web server, dispatch database, and dispatch core composed of the daemon, monitor, and dispatch and transfer modules. The main features of the modules are also shown.

5 CONCLUSIONS

k-Dispatch is a workflow manager providing automated execution, planning and monitoring of biomedical workflows. It completely screens out the users from the complexity of nowadays HPC systems. The presented version of k-Dispatch enables users to easily execute predefined workflows on various available HPC facilities by only providing the medical input data. k-Dispatch arranges for the rest, including the selection of the most suitable HPC facility, appropriate binaries, execution configuration considering the actual cluster load, transferring the input data and final results between the user and the HPC facility, and monitoring the execution, task dependencies and error checking.

6 FUTURE WORK

Next steps in the development are to (1) collect performance data for various code types, (2) improve the presented logic to select run configurations, (3) study jobs scheduling simulators, (4) further evaluate the implemented logic and selected run configuration on both, simple and real-world workflows, and finally (5) execute tested workflows in a real HPC environment.

Since collected performance data is sparse and incomplete the nearest suitable record is selected and used for the execution planning. In the future, interpolation techniques and machine learning methods will be used on performance data to better set the execution configuration.

Due to the cost of resources, varying background load and reproducibiltity, the experimental evaluation is difficult to be performed on production clusters. The solution may be evaluation using a cluster simulator [3].

7 ACKNOWLEDGEMENT

This work was supported by the FIT-S-17-3994 Advanced parallel and embedded computer systems project. This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science - LQ1602 and by the IT4Innovations infrastructure which is supported from the Large Infrastructures for Research, Experimental Development and Innovations project IT4Innovations National Supercomputing Center - LM2015070.

REFERENCES

[1] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, Anastasia Laity, Joseph C Jacob, and Daniel S Katz. 2005. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming* 13 (2005), 219–237.

[2] Ian Foster. 2006. Globus Toolkit Version 4: Software for Service-Oriented Systems. *Journal of Computer Science and Technology* 21, 4 (jul 2006), 513–520. https://doi.org/10.1007/s11390-006-0513-y

[3] Dalibor Klusacek, Simon Toth, and Gabriela Podolnikova. 2017. Complex Job Scheduling Simulations with Alea 4. *CEUR Workshop Proceedings* 1828 (2017), 53–59. https://doi.org/10.1145/1235 arXiv:arXiv:1603.07016v1

[4] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. 2006. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience* 18, 10 (aug 2006), 1039–1065. https://doi.org/10.1002/cpe.994

[5] Haiyan Meng and Douglas Thain. 2017. Facilitating the Reproducibility of Scientific Workflows with Execution Environment Specifications. *Procedia Computer Science* 108 (2017), 705 – 714. https://doi.org/10.1016/j.procs.2017.05.116