Optimization Patterns for the Decentralized Orchestration of Parameter-Sweep Workflows

Selim Kalayci, S. Masoud Sadjadi
School of Computing and Information Sciences
Florida International University
Miami, FL, USA
{skala001, sadjadi}@cs.fiu.edu

Abstract— **A large and diverse group of computational scientific research efforts deal with parameterized studies, in which same or similar computational tools are applied on different sets of data. Such uniform and well-defined analysis efforts can be encapsulated as parameter-sweep workflows. Due to computation and data intensive nature, resources that span across multiple domains may be needed for timely and efficient execution of this type of workflows. In our previous studies, we have designed and developed techniques to orchestrate the execution of large-scale workflows in a decentralized and adaptive manner. Through the usage of generic workflow patterns, centralized orchestration of workflows are transformed into decentralized and adaptive orchestration without modifying the business logic of the workflow. In this study, we propose some additional optimization patterns specific to characteristics and requirements of parameter-sweep workflows. By exploiting the general characteristics of parameter-sweep workflows, we provide ways to reduce control and data overheads associated with the decentralized orchestration. We also discuss some implementation issues that arise from the adoption of these optimization patterns.**

Keywords: *workflow, DAG, orchestration, optimization, parameter study*

I. INTRODUCTION

Parameter studies enable the conduct of research for a certain experiment on a varying set of data and under various possible circumstances. Through the automated design and execution of such studies, large amounts of data/parameters can be processed and analyzed and as a result more accurate research outcomes can be acquired. These types of studies are prevalent in a large and diverse group of research fields, such as bioinformatics, earthquake science, weather predictions, and molecular dynamics.

Automated design of parameter studies generally results in a simple and well-defined structure, which can be easily represented as a Directed-Acyclic-Graph (DAG) based workflow. A DAG-based workflow encapsulates all the computational tasks and data/control dependencies among those tasks. Depending on the size and scale of the parameter study, these workflows may contain large numbers of computational tasks which are generally highly-parallelizable. Thus, if available, deploying parallel tasks of such a workflow on parallelized resources would significantly reduce the overall execution time (makespan) of the workflow.

Deployment and execution of large number of interdependent tasks require utilization of specific tools, namely workflow management systems. Workflow management systems enable and oversee various stages in the lifecycle of a workflow, including the mapping, and orchestration stages. During the mapping stage, a workflow specification that is free of physical resource details (abstract workflow) is translated into one which is associated with actual physical resource information (concrete workflow). Then, a specific component (workflow execution engine) within the workflow management system orchestrates the execution of workflow tasks abiding by and following the data/control dependencies among them.

As mentioned earlier, due to highly-parallelizable nature of parameter-sweep workflows, those parallel tasks can be ideally deployed on parallel resources. Sometimes, these parallel resources may span across multiple computational domains, as in the case of a hybrid (public + private) cloud, academic cloud (e.g. FutureGrid [16]), and national/international cyberinfrastructure (e.g. XSEDE [18], Open Science Grid [17]). In such a scenario, mapping and orchestration of the workflow also spans across multiple domains. In a heterogeneous and dynamic computational platform like this, several factors cause significant overheads during orchestration of the workflow. In our previous study [15], we designed and developed a framework to limit the effect of these overheads through adopting a decentralized orchestration approach rather than a standard centralized orchestration approach.

In this study, we will build up on our existing decentralized orchestration framework. Our optimization efforts are targeted especially for large-scale parameter-sweep workflows, although they may also be applicable to more general workflow instances. By exploiting the specific characteristics of parameter-sweep workflows, we will introduce patterns to optimize the decentralized orchestration process of such workflows.

Optimization patterns we introduce in this paper suggest minor changes in the structure of the workflow and the business logic of certain tasks. As long as they are done properly, we argue that these changes would not affect the validity and integrity of results. To this end, consultation with the scientist and/or a domain expert to verify the

appropriateness of such changes may be optional or required depending on the case scenario.

In the rest of this paper, we first give some background information regarding DAGs, parameter-sweep workflows and their characteristics in Section 2. In Section 3, we provide basic information regarding the decentralized orchestration framework this study is built upon. In Section 4, we explain the optimization patterns and their merits for parameter studies. Section 5 concerns with the implementation issues associated with the usage of optimization patterns. In Section 6, we overview related literature and Section 7 provides a summary of the paper and a brief discussion about further issues.

II. BACKGROUND

A. Directed Acyclic Graphs (DAG) and Mapping

Directed Acyclic Graphs (DAG) is an important and commonly used method to capture the automated behavior of workflows. A DAG successfully encapsulates the computational tasks that comprise the workflow as well as the control and data dependencies among them. DAG specification for a certain research study can be either done directly or can be translated from another specification type. For most research studies, the same DAG specification can be utilized many times with only minimal modifications.

The specification of a DAG, whether done by the domain scientist or a computational expert, normally includes only the business logic of the workflow. Details regarding the deployment on computational resources are not specified at this point. Those details are resolved only right before or during the orchestration of the workflow and done mostly by automated software tools (e.g. Workflow management system). This automated process is referred to as the mapping of workflow tasks onto resources and the details of this process are out of the scope of this paper.

B. Workflow Management Systems (WFMS)

Workflow management systems (WFMSs) deal with all aspects of the typical lifecycle of a workflow. Lifecycle of a workflow basically consists of the design, mapping, orchestration, and monitoring stages. They also provide QoS and fault-tolerance based services during the lifecycle of a workflow. There are several well-known WFMS software in literature [2, 4, 8, 11, 14], each with different research foci and agenda. To the best of our knowledge, none of these tools are compatible with each other. Thus, after a researcher decides on a certain WFMS, she has to comply with the proprietary environment of the selected software.

In terms of the workflow structure they support, WFMSs in the current literature are divided into two groups: DAG-based [2, 8, 11] and non-DAG-based [4, 14] WFMSs. At this point, our studies are focused only on DAG-based workflows. Future research may be conducted to extend our studies to non-DAG-based workflows as well.

C. Parameter-Sweep Workflows

Scientific exploration demands repeatable experimentation, and sound and reliable analysis. Computational resources have provided great means to achieve these on a much larger and deeper level in almost all fields of research. Using computational tools, scientists can explore much bigger phenomena at much more detail with a higher-level of precision. Most of the scientific exploration necessitates the same or similar experimentation and analysis tools to be run over a wide-ranging spectrum of parameters/conditions. The multitude of observations helps the scientist to make better decisions concerning her study. Although, such parameter studies are conducted by scientists of almost all fields to a certain extent, here we especially focus on the ones that are compute- and data-intensive in terms of today's existing technological capabilities.

Due to the repetitive nature of large parameter studies, scientists ideally utilize automation tools in this otherwise labor-intensive process. DAG-based workflows can easily and successfully capture the business logic of most parameter studies. After the business logic of the parameter study is captured in a DAG-based workflow specification by the scientist or a computational expert, it can be handed over to a WFMS which will automate the rest of the lifecycle of the workflow.

Fig. 1 illustrates the DAG structure of a typical single-level parameter-sweep workflow. The top-level task acts as to generate and distribute the input data/parameters that would be consumed by the Processing tasks. Each Processing task receives a different set of input data/parameters and independently executes the same software/service using those data/parameters. The results generated by these Processing tasks are then passed on to the Data Aggregation/Post-Processing task. This task, depending on the specifics of the study, performs some sort of data aggregation and/or post-processing activities using the results acquired from Processing tasks.

In a parameter-sweep workflow, typically, the Processing Tasks are the most computation-intensive components of the whole workflow. But, also these tasks are embarrassingly-parallel, meaning they can be executed concurrently on various eligible computational resources. Also, the top-level task (i.e. Parameter Initialization) and the bottom-level task (i.e. Post-processing task) are typically more data-intensive in nature than Processing tasks.

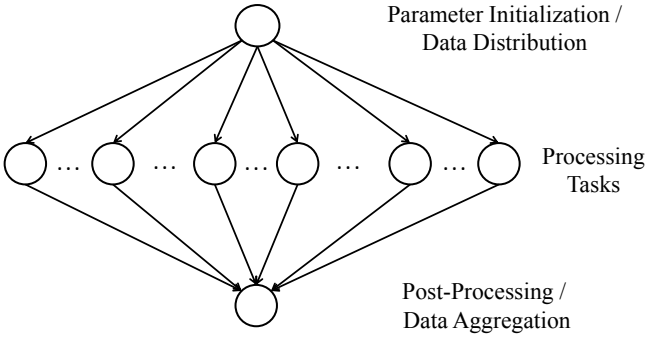

Figure 1. Single-level parameter-sweep workflow structure

Fig. 2 illustrates the DAG structure of a typical two-level parameter-sweep workflow. These types of workflows are used to perform more intense exploration/analysis activities iteratively at each level based on the results acquired at the previous level. A less-intensive version of software/service is run over a higher-number of Processing tasks at the first level. Then, based on the results acquired at the first level, a more intense version of the same or different software/service is run over a smaller-number of Processing tasks at the second level.

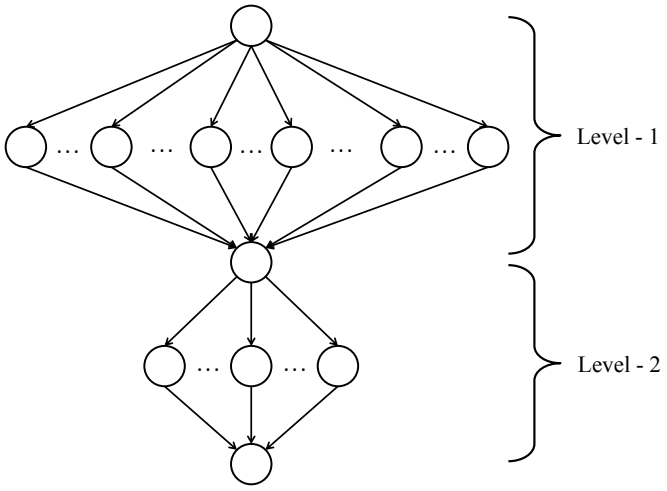

Figure 2. Two-level parameter-sweep workflow structure

III. DECENTRALIZATED ORCHESTRATION

During the mapping of workflow tasks onto physical resources, the main goal is to achieve the minimum makespan possible for the execution of the whole workflow. Accordingly, characteristics of tasks (e.g. estimated runtime) and data artifacts (e.g. estimated size), as well as the availability and characteristics of physical resources play a major role during the mapping stage. The resulting workflow specification is called a concrete workflow. A concrete workflow may span across multiple sites of resources.

In our previous study [15], we designed and developed a framework to adopt a decentralized orchestration approach for a multi-site workflow rather than a centralized orchestration approach. In the decentralized orchestration approach, orchestration of the whole workflow is performed in collaboration with local workflow managers at each site. This way, the control and communication overheads associated with centralized orchestration are minimized. Also, employment of local workflow managers at each site improves the accuracy and efficiency involved with the resource monitoring activities during run-time.

As part of our framework, we basically transform a single workflow specification into several collaborative workflow specifications. Then, each collaborative workflow specification is orchestrated independently by a peer workflow manager. At the end of this process, the orchestration of the whole original workflow is accomplished.

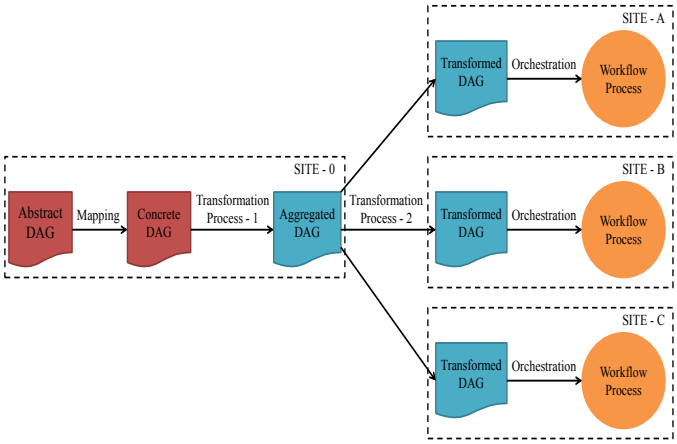

Figure 3. DAG specification stages following our decentralized orchestration framework

Fig. 3 illustrates the stages the DAG specification goes through from the abstract specification to the point it gets to be orchestrated by multiple peer workflow managers according to our decentralized orchestration framework. After the Concrete DAG is created, the DAG specification is aggregated with certain mapping and contact information that allows each peer workflow manager to subsequently generate the Transformed DAG specification. These Transformed DAG specifications are then handed over to local workflow execution engines to be orchestrated. A Workflow Process represents that the lifecycle of the workflow has reached the stage where its tasks are ready to be executed by computational resources.

Fig. 4 illustrates the DAG Patterns that make up the building blocks of DAG-based workflows. These patterns are also utilized during the generation of Transformed DAG specifications at each site. For more details on these transformation activities, please refer to [15].

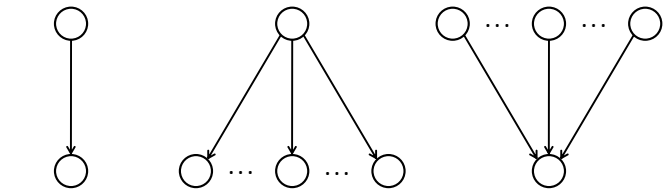

Figure 4. DAG Patterns [15]

A workflow orchestrated according to our decentralized orchestration approach still has the exact same computational tasks as the original workflow specification, but they are scattered across multiple workflow specifications. The transformations in the workflow structure are done only to facilitate collaborative orchestration, which do not affect the business logic at all.

IV. OPTIMIZATION PATTERNS

In this section, we will introduce the optimization patterns that we employ on our existing decentralized orchestration framework. Even though they may be applicable to more general instances of workflows as well, these patterns are

mainly designed to exploit the general characteristics of parameter-sweep workflows.

Optimization patterns that we introduce here suggest minor changes in the DAG structure of the workflow. These changes are proposed due to the specific nature of a parameter-sweep workflow where the dependencies among tasks are much more loosely coupled than a more generic workflow instance. Also, the relationships among different levels of tasks at a parameter-sweep workflow are more uniform and flexible than a more generic workflow instance. By exploiting these characteristics of parameter-sweep workflows, we provide optimizations to the decentralized orchestration of such workflows.

A. Parameter Initialization / Data Distribution

This pattern applies to the Fork/Branch pattern that manifests itself between the top-level Parameter Initialization task and the lower-level Processing tasks. At a multi-site deployment of a large-scale parameter-sweep workflow, all these embarrassingly parallel Processing tasks have control and data dependency on the single Parameter Initialization task. Fig. 5 illustrates a typical scenario for such behavior. Each different color notates deployment of a task on a different site. Fig. 5 also illustrates the data/control logistics of this pattern following a centralized orchestration approach. As seen, all the data and control logistics is handled by the centralized workflow manager even though the tasks span across multiple different sites. This behavior incurs significant control and data overhead to the total workflow execution time.

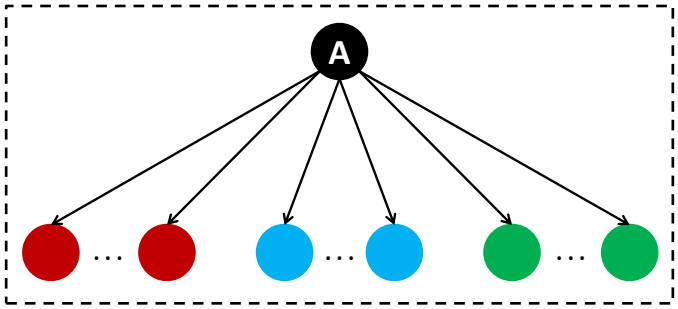

Figure 5. DAG-mapping and centralized orchestration

Fig. 6 illustrates the overall view for the orchestration of the same pattern following our standard decentralized approach. In these Figures, dashed lines notate the boundaries for the responsibility of each site. And each site employs its own local workflow manager. Also, the rectangle boxes that lie between two site boundaries illustrate the Synchronization activities which are inserted to the DAG specification during the second phase of DAG Transformation process (see Fig. 3). Thus, Fig. 6 illustrates the collaborative DAG structures and the interactions among them to ensure synchronization. Comparison between Fig. 5 and Fig. 6 shows a significant reduction in terms of data and control overhead incurred during the orchestration process.

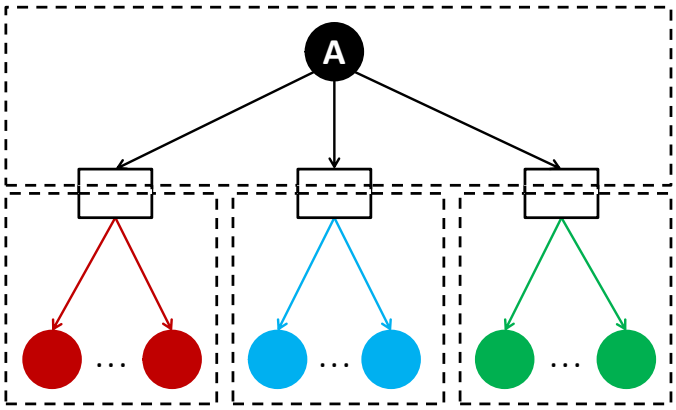

Figure 6. Overall view for the decentralized orchestration

Due to flexible characteristics of the Parameter Initialization task at the top-level of this DAG pattern, we propose an optimization pattern as illustrated in Fig. 7. In Fig. 7, it can be seen that, task A (Parameter Initialization task) has been replicated such that each site employs its own copy. As a result, 3 collaborative peers can orchestrate their own DAG structures completely independent from each other. Thus, the orchestration of the presented DAG structure incurs no overhead to the workflow execution time. Notice that, the replicated task A is referred as A' as it may be necessary to make minor adjustments in the business logic of task A. We will discuss this issue in Section 5.

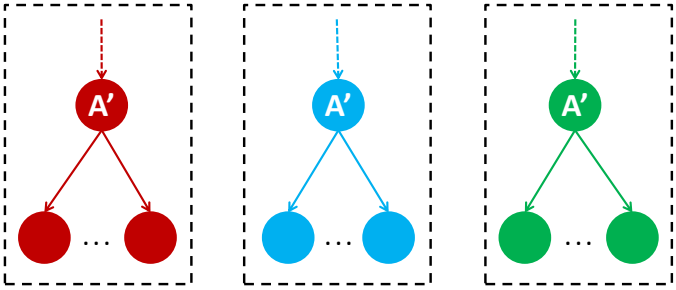

Figure 7. Overall view for the Optimized DAG which is orchestrated in decentralized manner

B. Post-Processing / Data Aggregation

This pattern applies to the Join pattern that manifests itself between the Processing tasks and the lower-level Post-Processing task. At a multi-site deployment of a large-scale parameter-sweep workflow, all these embarrassingly parallel Processing tasks have control and data dependency on the single Post-processing task. Fig. 8 illustrates the data/control logistics of this pattern following a centralized orchestration approach. As seen, all the data and control logistics is handled through the centralized workflow manager even though the tasks span across multiple different sites. This behavior incurs significant control and data overhead to the total workflow execution time.

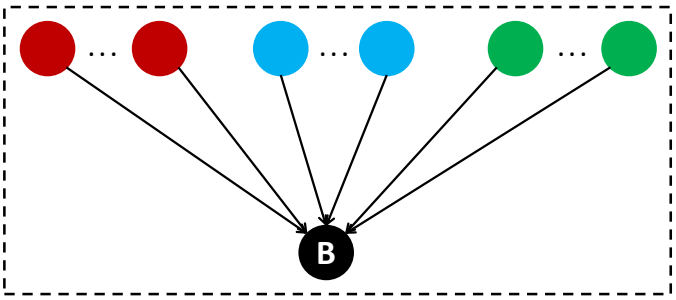

Figure 8. DAG-mapping and centralized orchestration

Fig. 9 illustrates the overall view for the orchestration of the same pattern following our standard decentralized approach. The collaborative DAG structures and the interactions among them to ensure synchronization can be seen. Again, comparing Fig. 8 and Fig. 9 shows a significant reduction in terms of data and control overhead incurred during the orchestration process.

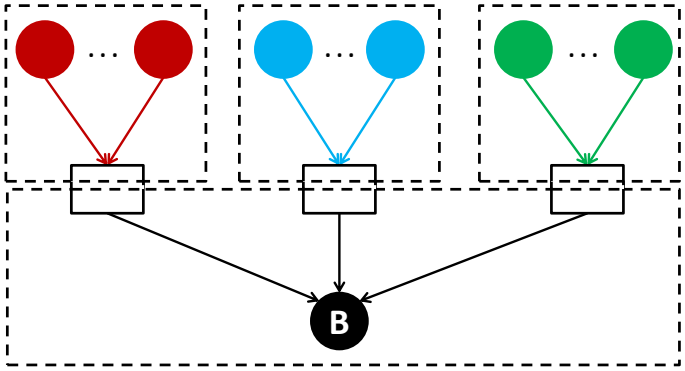

Figure 9. Overall view for the decentralized orchestration

We propose a corresponding optimization pattern for this DAG pattern as illustrated in Fig. 10. In Fig. 10, it can be seen that, task B (Post-Processing task) has been replicated such that each site employs its own copy. As a result, the collaborative peers can perform the Post-processing task at their local site. Notice that, the replicated task B is now referred as B' as it may be necessary to make minor adjustments in the business logic of task B. Also, notice that, in this case we also need a second-level Post-Processing (task B*) to further perform the Post-processing on local results. However, even with the inclusion of task B*, the optimized orchestration can help reduce the overheads incurred due to data transfers between Processing tasks and (original) task B. Due to their nature, most scientific applications generate large sizes of output files. During the non-optimized orchestration (centralized or decentralized) of such a pattern, all of the individual output files generated by Processing tasks are required to be transferred to a remote site, so that a single task B can perform Post-processing activities on all this data. According to our optimized orchestration of this pattern, individual output files are not required to be transferred to a remote site. Each site needs to only transfer the output files generated by the local task B' (that are generally much more modest in size), which are then further Post-processed by task B*.

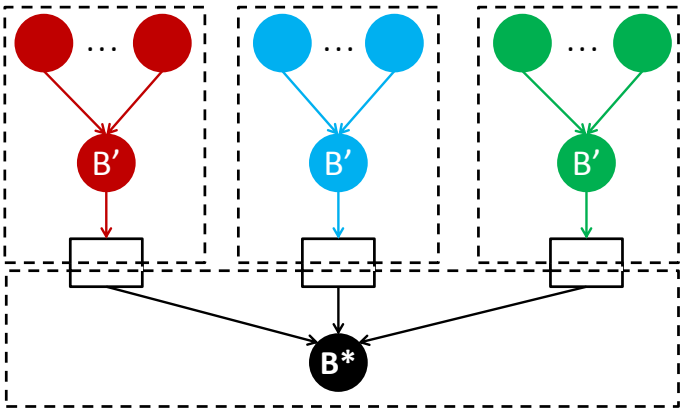

Figure 10. Overall view for the Optimized DAG which is orchestrated in decentralized manner

V. Prototype Implementation

Our prototype implementation is based on Condor DAGMan [5] workflow execution engine which is a widely used workflow execution tool for a wide variety of scientific workflows. The original orchestration approach for Condor DAGMan is centralized and it does not perform the kind of optimization activities that are proposed in this paper.

Condor DAGMan specifies the DAG structure of a workflow including the tasks and dependencies among them in a standard text file. Orchestration of a workflow is basically achieved by submitting/launching subsequent tasks that are eligible to run as their dependencies are met with the completion of previously launched tasks.

A. Decentralization

Condor DAGMan orchestrates a DAG-based workflow according to the DAG specification representing the computational tasks and their control/data dependencies. Due to its centralized orchestration approach, tasks that are mapped on remotes sites are also submitted, controlled and monitored by the same Condor DAGMan instance.

In our decentralized framework, each site employs its own Condor DAGMan instance, and orchestrates its local Transformed DAG specification. Each Condor DAGMan instance submits, controls, and monitors tasks that are deployed on their local site. Synchronization activities among peer Condor DAGMan instances help facilitate the collaborative orchestration of the whole workflow.

Transformation of DAG specification at each site occurs independently. In this transformation process, we make use of standard Condor DAGMan keywords (e.g. DONE) and utilities (e.g. PRE/POST scripts) to provide the necessary changes for the local DAG specification. Here, by marking remote tasks with the DONE keyword, we let the local Condor DAGMan to skip those tasks. Similarly, by creating and inserting PRE/POST scripts at each local site properly, we facilitate the synchronization among peer Condor DAGMan instances. Since we only make use of these standard Condor DAGMan functionalities in our transformation process, no other changes are necessary in the system.

More details regarding the implementation of DAG Transformation Process and the prototype implementation based on Condor DAGMan can be found in [15] and [13].

B. Optimization Patterns

As explained in Section 4, we introduce optimization patterns to exploit certain characteristics of large-scale parameter-sweep workflows. However, these patterns necessitate minor changes to the original DAG structure, which in turn may affect the business logic of the workflow application. Thus, these structural changes should be verified to be appropriate by a domain expert. After the DAG structure change is approved by the domain expert, he/she should also be consulted regarding the adjustments intended for Parameter Initialization task and Post-processing task. Once the appropriate task adjustment methods are agreed upon, these methods can be reused multiple times to run similar instances of large-scale parameter-sweep workflows.

As suggested above, the domain expert should contribute to the proper method for replicating the Parameter Initialization task (see Fig. 7). In some cases, the Parameter Initialization task involves generating input data/parameters in a randomized way (e.g. Monte Carlo methods). In some other cases, this task involves splitting a wide-range of input data/parameters uniformly among the Processing tasks. For these and similar cases, the proper task adjustment methods are expected to be much more trivial than more sophisticated cases.

The business logic of the Post-processing/Data Aggregation tasks also usually consists of standard and well-defined behavior, which may be adjusted accordingly through consultation with a domain expert. For example, in some cases this task involves generating statistical results from the data generated by Processing tasks. In some other cases, this task mainly involve choosing the best results (or eliminating worst results) among all the generated results according to certain criteria. For these and similar cases, the proper task adjustment methods are expected to be much more trivial than more sophisticated cases.

After the consultation and proper task adjustment phases are completed successfully, these behaviors are needed to be specified in the Aggregated DAG specification. At this point, we provide this information manually (i.e. tasks to be restructured, task adjustment methods). In the next stage of the transformation process, each local site comes up with its own local transformed DAG specification, similar to the process explained in the previous subsection. Again, since we only make use of standard Condor DAGMan functionalities throughout the transformation process, no other changes are necessary in the system.

VI. Related Work

Extensive amount of research has been conducted on different aspects of scientific workflow management. A large number and variety of workflow management systems [1] have been designed and developed mostly resulting in proprietary environments and custom goals. However, one of the most heavily investigated aspect concerns with the workflow task scheduling algorithms [3] at dynamic and heterogeneous environments. Task scheduling under these circumstances is a very complicated process, and the quality of it significantly affects the quality of mapping and the overall execution time of workflows. However, in this paper, we do not address this aspect and rely on existing techniques for the mapping stage of workflows.

Pegasus workflow management system [2] also makes use of the Condor DAGMan [5] as the underlying workflow execution engine. Before the orchestration of the workflow, Pegasus goes through some optimization activities. One optimization technique, which is named workflow reduction, eliminates those tasks in the workflow for which the output files have been already generated during previous executions. Another optimization technique performs task clustering, so as to increase the granularity of tasks and reduce the scheduling and orchestration overheads. Task clustering is an especially effective technique where a workflow contains large-number of short-running tasks.

ASKALON workflow management system [4] has hierarchical architecture for workflow orchestration. It also provides some optimization techniques such as clustering of tasks similar to Pegasus. It also performs optimization activities to reduce data transfer overheads such as archiving and compressing files.

In [6], authors show that the performance of the workflow execution engine can be a critical factor in determining the workflow completion time. They use Condor DAGMan as the workflow engine, and they analyze the workflow completion time by changing system parameters (scheduling interval, dispatch rate, job submission rate) and also restructuring the workflow (e.g. task clustering).

There are also several pattern-based studies for workflows aimed for business applications [7, 9, 10, 12] that mainly deal with decentralization and fault-tolerance aspects of business processes.

VII. Conclusion

In this paper, we propose optimization patterns to improve the orchestration efficiency of large-scale parameter-sweep workflows. Due to large-scale and highly-parallelizable nature of parameter-sweep workflows, computational tasks may span across multiple sites of resources. This generally causes efficiency problems for the orchestration of such workflows. To improve the efficiency, first we suggest employing our existing decentralized workflow orchestration framework to eliminate various control and data overheads associated with the centralized orchestration. To further improve the efficiency of the orchestration, we propose the utilization of optimization patterns that exploit the general characteristics of parameter-sweep workflows. These patterns cause minor changes to the structure of the original DAG specification and business logic of certain tasks. We discuss the potential drawbacks of making such changes and argue that in most cases they can be easily addressed by incorporating the feedbacks of a domain expert in the process. Even though these patterns were designed and aimed primarily for parameter-sweep workflows, they may be applicable, to certain extent, for more general-purpose large-scale workflows as well.

In the future, we would like to further investigate possible optimization techniques for orchestration and lifecycle management of both parameter-sweep workflows and general-purpose workflows. We also would like to explore non-DAG-based workflows and seek ways to extend our decentralized orchestration framework and optimization concepts for this type of workflows as well.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. OISE-0730065.

REFERENCES

[1] Yu, Jia, and Rajkumar Buyya. "A taxonomy of workflow management systems for grid computing." *Journal of Grid Computing* 3, no. 3-4 (2005): 171-200.

[2] Deelman, Ewa, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta et al. "Pegasus: A framework for mapping complex scientific workflows onto distributed systems." *Scientific Programming* 13, no. 3 (2005): 219-237.

[3] Yu, Jia, Rajkumar Buyya, and Kotagiri Ramamohanarao. "Workflow scheduling algorithms for grid computing." In *Metaheuristics for scheduling in distributed computing environments*, pp. 173-214. Springer Berlin Heidelberg, 2008.

[4] Fahringer, Thomas, Radu Prodan, Rubing Duan, Jüurgen Hofer, Farrukh Nadeem, Francesco Nerieri, Stefan Podlipnig et al. "Askalon: A development and grid computing environment for scientific workflows." In *Workflows for e-Science*, pp. 450-471. Springer London, 2007.

[5] Condor team, "The directed acyclic graph manager", www.cs.wisc.edu/condor/dagmang, 2002.

[6] Singh, Gurmeet, Carl Kesselman, and Ewa Deelman. "Optimizing grid-based workflow execution." *Journal of Grid Computing* 3, no. 3-4 (2005): 201-219.

[7] Wohed, Petia, Wil MP van der Aalst, Marlon Dumas, and Arthur HM ter Hofstede. "Pattern based analysis of BPEL4WS". QUT Technical report, FIT-TR-2002-04, Queensland University of Technology, Brisbane, 2002.

[8] Oinn, Tom, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver et al. "Taverna: a tool for the composition and enactment of bioinformatics workflows." *Bioinformatics* 20, no. 17 (2004): 3045-3054.

[9] Pantazoglou, Michael, Ioannis Pogkas, and Aphrodite Tsalgatidou. "Decentralized Enactment of BPEL Processes." (2013): 1-1.

[10] Yu, Weihai. "Decentralized orchestration of BPEL processes with execution consistency." In *Advances in Data and Web Management*, pp. 665-670. Springer Berlin Heidelberg, 2009.

[11] Berman, Francine, Andrew Chien, Keith Cooper, Jack Dongarra, Ian Foster, Dennis Gannon, Lennart Johnsson et al. "The GrADS project: Software support for high-level grid application development." *International Journal of High Performance Computing Applications* 15, no. 4 (2001): 327-344.

[12] Kalayci, Selim, Onyeka Ezenwoye, Balaji Viswanathan, Gargi Dasgupta, S. Masoud Sadjadi, and Liana Fong. "Design and implementation of a fault tolerant job flow manager using job flow patterns and recovery policies." In *Service-Oriented Computing–ICSOC 2008*, pp. 54-69. Springer Berlin Heidelberg, 2008.

[13] Kalayci, Selim, Gargi Dasgupta, Liana Fong, Onyeka Ezenwoye, and Seyed Masoud Sadjadi. "Distributed and Adaptive Execution of Condor DAGMan Workflows." In *SEKE*, pp. 587-590. 2010.

[14] Taylor, Ian, Matthew Shields, Ian Wang, and Andrew Harrison. "The triana workflow environment: Architecture and applications." In *Workflows for e-Science*, pp. 320-339. Springer London, 2007.

[15] S. Kalayci and S. M. Sadjadi, "Pattern-based Decentralization and Run-time Adaptation Framework for Multi-site Workflow Orchestrations", In Proceedings of The 2013 International Conference on Software Engineering and Knowledge Engineering, Boston, USA, July 2013.

[16] "FutureGrid.". Available: http://futuregrid.org/

[17] "Open Science Grid". Available: http://www.opensciencegrid.org.

[18] "XSEDE". Available: http://www.xsede.org/