Workflow Performance Profiles: Development and Analysis

Dariusz Król[1,2], Rafael Ferreira da Silva[2], Ewa Deelman[2], Vickie E. Lynch[3]

[1] AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science and Academic Computer Center Cyfronet, Krakow, Poland
[2] USC Information Sciences Institute, Marina Del Rey, CA, USA
[3] Oak Ridge National Laboratory, Oak Ridge, TN, USA

Abstract. This paper presents a method for performance profiles development of scientific workflow. It addresses issues related to: workflows execution in a parameter sweep manner, collecting performance information about each workflow task, and analysis of the collected data with statistical learning methods. The main goal of this work is to increase the understanding about the performance of studied workflows in a systematic and predictable way. The evaluation of the presented approach is based on a real scientific workflow developed by the Spallation Neutron Source - a DOE research facility at the Oak Ridge National Laboratory. The workflow executes an ensemble of molecular dynamics and neutron scattering intensity calculations to optimize a model parameter value.

1 Introduction

Scientific workflows are a popular way of conducting extreme-scale scientific research, which may require composing thousands of computational jobs. Workflows have been widely used in different science domains, including astronomy and gravitational wave, seismology, and others [21]. In a workflow, each job may have different requirements for CPU, I/O, and memory. An accurate specification of these requirements (along with job's runtime) is crucial to optimize the performance and accuracy of resource provisioning and job scheduling algorithms, reduce the overall runtime, decrease resource utilization, etc.

Due to the high complexity of scientific workflows, users often do not have detailed knowledge about workflow jobs requirements. Typically, users overestimate job requirements, since an underestimation may lead to job termination (e.g., due to exceeding the maximum job runtime). Resource requirements specification can also affect job's execution, e.g. setting too few resources may lead to extreme execution time in case of large jobs. When considering running several instances of the same workflow with different input parameter values, it is fundamental that job requirements estimation have good accuracy. A common method to address this issue is to derive predictions from the analysis of past workflow executions. Therefore, we see the need for fine-grained monitoring tools to automatically collect such information, and to build workflow profiles.

Most of the work in workflow profiling target the peak data of job requirements [7, 12]. Although this information allows the estimation of peak requirements (e.g., disk space, memory, etc.), it does not provide any insight into how how resources are consumed by jobs over time within a workflow. This knowledge not only improves the overall understanding of workflow executions, but it increases the efficiency of job scheduling and resource utilization, in particular when planning large-scale workflows. Over the years, several application monitoring systems have been developed, however their practical application to produce performance profiles is often constrained by the inability to compare measurements from multiple executions.

The contributions of this paper include: (1) a holistic process for the development and analysis of workflow performance profiles; (2) description of different phases of the proposed process along with existing software, which can facilitate its practical application; (3) evaluation of the presented approach using a large high performance computing (HPC) system available at the NERSC facility [13]; and (4) the profiling of a real workflow application.

2 Performance Profiles of Scientific Workflows

Scientific workflows are often executed multiple times with different input parameter values to study distinct conditions, e.g. climate modeling with different mesh resolutions. Variations of the input parameters may lead to significant differences in resource requirements for the jobs. Understanding the relationships between the workflow's input and job performance metrics is crucial to accurately estimate these requirements. Also, it is important to understand how a variation in an input parameter may affect job performance. Discovery of these relationships is often referred to as *sensitivity analysis* [19], i.e. the assessment of how output variables are affected based on variations in the input variables. This type of analysis is mostly performed on the final values of responses, and it does not include variability of performance during job runtime. We propose an approach to conduct such analysis, which includes the temporal aspect of the workflow performance behavior, i.e. performance is measured not only at job completion, but also at different time instants during the job execution.

In this context, *a performance profile for a given job and a metric can be described as a time series with values of the metric measured in equidistant points in time during the job execution.* By collecting data from multiple executions of the same workflow configuration, we can compute statistically significant performance profiles for each job in a workflow.

2.1 End-to-end approach to performance profiles generation

The process of generating workflow performance profiles takes as input a workflow and a set of distinct input parameters. For each possible combination of these parameters, a workflow execution is performed. The workflow performance data gathered using monitoring tools is then used to build average profiles, and

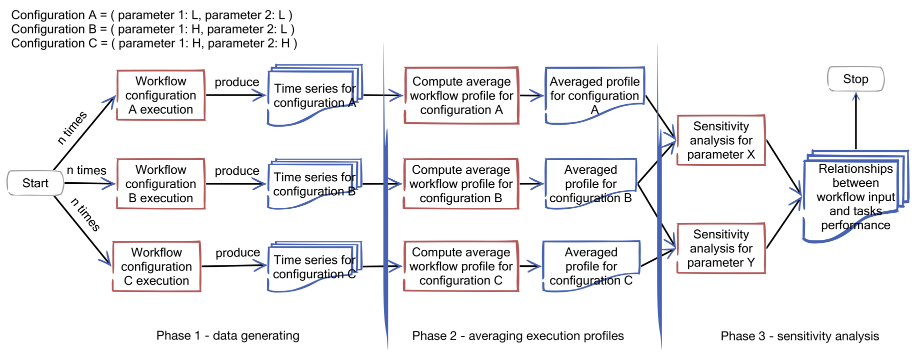

Configuration A = (parameter 1: L, parameter 2: L)
Configuration B = (parameter 1: H, parameter 2: L)
Configuration C = (parameter 1: H, parameter 2: H)
Start
n times
n times
n times
Workflow configuration A execution
Workflow configuration B execution
Workflow configuration C execution
produce
produce
produce
Time series for configuration A
Time series for configuration B
Time series for configuration C
Compute average workflow profile for configuration A
Compute average workflow profile for configuration B
Compute average workflow profile for configuration C
Averaged profile for configuration A
Averaged profile for configuration B
Averaged profile for configuration C
Sensitivity analysis for parameter X
Sensitivity analysis for parameter Y
Relationships between workflow input and tasks performance
Stop
Phase 1 - data generating
Phase 2 - averaging execution profiles
Phase 3 - sensitivity analysis

Fig. 1. Overview of the process for generating workflow performance profiles.

to derive the sensitivity analysis. The ultimate goal of this process is to provide quantitative information about relationships between the input parameters and jobs' performance over time. An overview of the proposed approached is shown in Fig. 1. Below, we describe each of the process' phases in detail:

Phase 1 (Data gathering). In this phase, a workflow execution is performed for each set of different input parameters (referred to as workflow configurations). To increase statistical significance, each configuration runs multiple times. Each workflow execution produces a set of time series for various performance metrics including CPU and memory usage, I/O load, among others. Note that for tightly-coupled parallel programs, time series performance values are collected for each individual process executed within the program.

Phase 2 (Averaging execution profiles). Time series of performance measurements are then used to compute *averaged execution profiles*—a time series describing a performance metric for a given job. Averaged performance profiles are computed based on multiple workflow executions of the same configuration (same set of input parameter values). The outcomes of this phase are execution profiles for each collected performance metric and for each job in a workflow.

Phase 3 (sensitivity analysis). The goal of this phase is to assess the impact on jobs performance by varying input parameters values. In this paper, we address this analysis with well-known statistical methods.

2.2 Generating performance profiles in practice with HPC

The presented approach may pose difficulty in practice, specially when manually implementing it in HPC environments. We identify two detached aspects of our approach, which can be done in an automatic way with existing software: (1) executing and monitoring workflow executions, and (2) conducting data collection.

We use the `Pegasus` [4] workflow management system to run scientific workflows on different computational infrastructures. It includes *in-situ online* monitoring [8] that collects detailed information about jobs performance and the compute resources, including: system and process CPU utilization and memory usage, and process I/O. Each process in a job is monitored separately using information from the `proc` virtual filesystem and with system calls interception. This detailed monitoring information constitutes the basis of the workflow performance profiles.

We use `Scalarm` [9,10] as a platform for parameter studies on heterogeneous computational infrastructures, i.e. to execute the same application (a scientific workflow in our case), with different input parameters. Scalarm supports different steps of this process including: input parameter space specification, application execution, and data collection. It is currently used within the EU FP7 PaaSage project [14] that aims at creating a solution for modelling and optimized deployment of cloud-oriented applications.

By combining Pegasus and Scalarm, we enable the data gathering phase of the process for generating and analyzing workflow performance profiles (Fig. 1). Both tools are generic, and support a vast number of different high performance and high throughput systems, which significantly increases the probability of successful practical application of the proposed approach.

3 Experimental Evaluation

In this section, we present an application of our method for generating and analyzing workflow performance profiles. We focus on phases 2 and 3 from the process described in Fig. 1, i.e. calculating averaged performance profiles, and conducting a sensitivity analysis of the workflow performance for different input parameters. The data gathering process (phase 1) is not discussed in this paper, since the data generation process is automatically performed by Scalarm and Pegasus. Due to limited space and a large amount of data collected, we focus our analysis to a subset of the data, which provides the most relevant information.

3.1 Scientific workflow application

To evaluate our approach, we use a material science-related workflow developed at the Spallation Neutron Source (SNS) facility. The workflow executes an ensemble of molecular dynamics and neutron scattering intensity calculations to optimize a model parameter value, e.g. to investigate temperature and hydrogen charge parameters for models of water molecules. The results are compared with experimental data from experiments such as QENS [2].

The SNS workflow takes as input a set of temperature values and 4 additional parameters: type of material, the number of required CPU cores, the number of timesteps in simulation, and the frequency the output data is written. Fig. 2 shows a branch of the workflow to analyze one temperature value. First, each set of parameters is fed into a series of parallel molecular dynamics simulations using NAMD [16]. The first simulation computes an equilibrium (`namd_ID0000002`), which is used by the second (`namd_ID0000003`) to compute the production dynamics. The output from the MD simulations has the global translation and

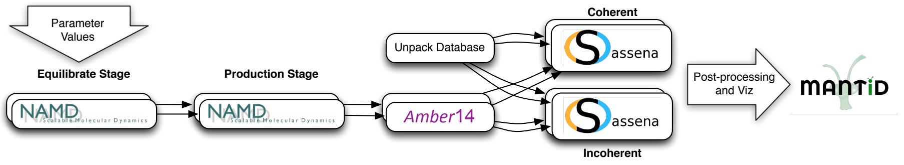


Fig. 2. A diagram of a branch of the SNS workflow.

rotation removed using AMBER's [18] *cpptraj* utility (`ptraj_ID0000004`), which is passed into Sassena [11] to compute coherent (`sassena_ID0000005`) and incoherent (`sassena_ID0000006`) neutron scattering intensities from the trajectories. The final outputs of the workflow are transferred to the user's desktop and loaded into Mantid [1] for analysis and visualization.

3.2 Experiment configuration and execution environment

The data gathering was prepared to run 16 different configurations of the workflow created as combinations of the input parameter values. We define a two-level analysis: *low* (`L`) representing small values, and *high* (`H`) representing large values of each parameter. There are many sampling methods available in the literature (especially in case of sensitivity analysis popular techniques include Morris and Sobol' samplings) however the use of the proposed 2^k method is justified by:

- time-consuming calculations - running a production workflow hundreds of time to calculate input sensitivity would be infeasible,
- interpretation simplicity during the analysis phase,
- exploratory approach where we first use a coarse-grain sampling to develop initial understanding and then we move to a fine-grain sampling in a subspace of the input parameter space to improve initial findings.

Table 1 summarizes the values used for the analyzes. Atoms represent the material used for simulation. Cores represent the number of cores used by NAMD and Sassena jobs, respectively. Each configuration was executed 3 to 5 times to confirm performance homogeneity of task executions and to eliminate any outliers created by using shared resources, e.g. storage systems. To assess the performance impact of each parameter value on the workflow execution, we limit our analysis to pairs of configurations, where only one parameter value is varied. We focus on two specific configurations where the material used is varied: `atoms_L_cores_L_timesteps_H_outfreq_L` and `atoms_H_cores_L_timesteps_H_outfreq_L`. In this case, we can evaluate the impact of different material types on the workflow performance for small number of cores and output data frequency (`L`), and large timesteps (`H`). Sequential jobs (e.g., `ptraj_ID0000004`) have a very short runtime (below 1 minute), thus we do not consider them in our analysis.

Fig. 3 shows an overview of the testbed used in the experiments. Workflows ran on the Hopper Supercomputer at NERSC [13], a Cray XE6 system with a peak performance of 1.28 Petaflops, while Scalarm and Pegasus were deployed

Table 1. Input parameter values for the SNS workflow in the experimental evaluation.

Factor level	Atoms	Cores	Timesteps	Data write freq.
L	3,692	144/72	50,000 = 0.005 ns	1,000 = 0.0001 ns
H	7,496	288/144	500,000 = 0.5 ns	5,000 = 0.0005 ns

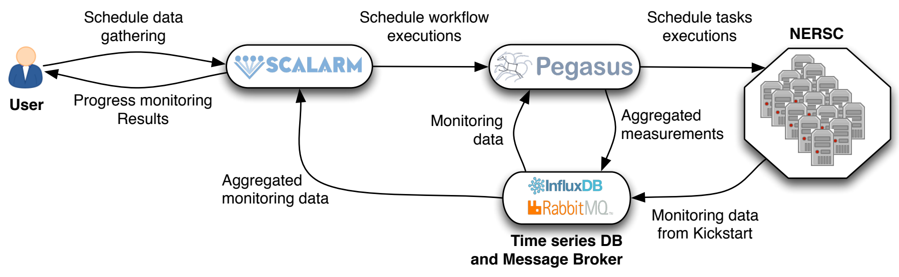

Schedule data gathering
Schedule workflow executions
Schedule tasks executions
NERSC
User
Progress monitoring Results
SCALARM
Pegasus
Monitoring data
Aggregated measurements
Aggregated monitoring data
InfluxDB
RabbitMQ
Time series DB and Message Broker
Monitoring data from Kickstart

Fig. 3. Overview of the testbed used during the experimental evaluation.

on external hosts. A Pegasus workflow was launched for each configuration obtained from the set of parameters. A message broker and a time series database server were also deployed to collect online monitoring data during the workflow execution. Performance metrics were collected every 5 seconds, and include CPU utilization (`stime` and `utime`), I/O (`read` and `write_bytes`, `iowait`, `syscr`, and `syscw`), memory (`vmRSS` and `vmSize`), and number of threads.

3.3 Experimental results and discussion

We focus our analysis on the four MPI jobs for performing molecular simulation (NAMD and Sassena), since they represent most of the total CPU hours of the workflow. Although several analyzes could be derived from this data, we focus in: (1) determining whether a job is CPU- or I/O-bound; (2) studying job behavior as a function of I/O- and CPU-related metrics; and (3) studying the job performance behavior with averaged profiles.

Resource-boundedness. Determining whether an application is CPU- or I/O-bound may aid the resolution of poor performance issues, in particular for large-scale applications. Typically, applications are classified into one of these categories based on the ratio between the time spent in the user (`utime`) and kernel (`stime`) spaces—handling I/O-related interruptions, etc. However, for long running jobs (several hours or days), an application may behave differently along its execution. For instance, an application can be mostly CPU-bound with several instants where I/O operations prevail. Therefore, computing platforms may consider this dynamic behavior during scheduling or performance tuning.

We computed averaged workflow performance profiles for every MPI job. We combined time series from runs of the same workflow input configuration for each monitored performance metric, and then computed the average value of the

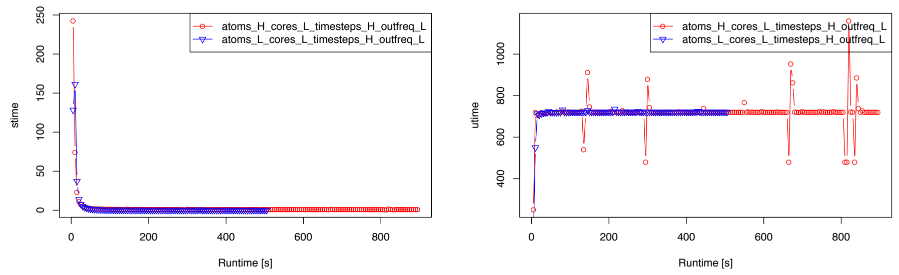


Fig. 4. Averaged performance profiles of `stime` [s] (left) and `utime` [s] (right) for a `namd_ID0000002` job.

metric at each monitoring interval (5 seconds). Averaged performance profiles show how resources are consumed by a job over time.

Overall, jobs present similar profiles for `stime` and `utime`, with larger configurations (`atoms_H`) having longer runtimes. Fig. 4 shows an example of averaged profiles for `namd_ID0000002`. Most of the execution time is spent on the user space. I/O operations are mostly executed in the beginning of the execution, and then few (nearly negligible) write operations are performed along the job execution. In the `utime` analysis (Fig. 4-*right*), occasional spikes disrupt the linear behavior once the heaviest I/O operations have completed. These spikes are due to the short timespan of the monitoring interval. Note that peaks are often followed by troughs of similar magnitude, or vice-versa.

Jobs behavior analysis. Understanding jobs behavior is fundamental to the design and optimization of computing systems (e.g., job scheduling, resource provisioning, etc.). Therefore, we assess how different workflow input parameters impact jobs performance. We analyze relative cumulative value (RCV) of a performance metric as a function of normalized job runtime, which describes how many resources where consumed from the beginning of execution till the specified time moment. Since runtime varies among executions (mostly influenced by the machine's performance or external load), the scaled runtime values allow the analysis of (1) overall resource consumption over time, and of (2) infrastructure-related anomalies between different executions of the same workflow configuration. Anomalies can be inferred from abnormal behaviors of the RCV. The analysis and handling of these behaviors is out of the scope of this work.

Fig. 5 shows RCVs for `stime` and `utime` from the NAMD jobs previously analyzed. Notice: for each configuration, a RCV graph was plotted for each run. There is a visible difference between both configurations for `stime` (Fig. 5-*left*). The configuration with the lower number of atoms (`atoms_L`) performs most of the I/O operations within the first 15% of its runtime, while `atoms_H` spreads I/O operations over time. This difference is due to: (1) the lower percentage number of I/O operations performed by `atoms_H` in the beginning of the execution; or (2) a significant increase on the number of I/O operations by `atoms_H` along the execution. In the subsequent analysis, we investigate this difference in detail. The `utime` behavior is similar and linearly correlated to the job runtime in

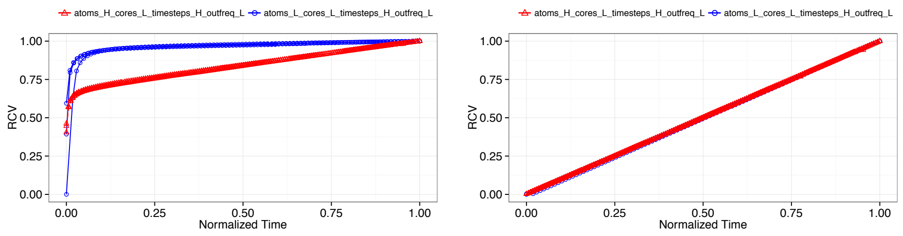


Fig. 5. RCVs of `stime` (left) and `utime` (right) for a `namd_ID0000002` job.

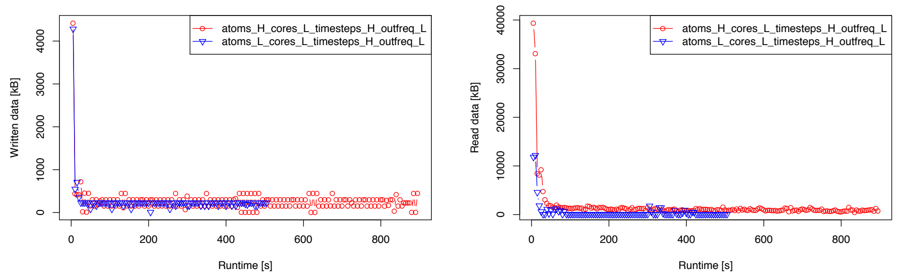


Fig. 6. Performance profile of write (left) and read (right) operations for `namd_ID0000002`.

both cases. This result indicates that computations in both configurations follow nearly the same pattern, and have a consistent number of operations throughout the job execution. This analysis confirms similarity between different runs of the workflow.

Performance profiles. Although RCVs are useful for modeling job behaviors, they cannot characterize and quantify the differences obtained using different input parameters. The difference in `stime` between both configurations can be explained by analyzing performance profiles for I/O-related metrics. Fig. 6 shows the performance profile of `write_bytes` and `read_bytes` for `namd_ID0000002` jobs with different number of atoms. The amount of data written per time interval (Fig. 6-*left*) is nearly identical for both configurations, except that `atoms_H` takes longer. Note that for the larger configuration, the magnitude of peaks and troughs on average is similar to `atoms_L`. In contrast, a significant difference on the amount of bytes read is observed in the first stage of the execution for `atoms_H` (Fig. 6-*right*). Furthermore, there is a significant amount of data that is continuously read during the job execution, while almost no read operations are performed for `atoms_L` (notice the scale difference between Fig. 6-*right* and Fig. 6-*left*). This result indicates that most of the time spent in the kernel space (Fig. 5-*left*) is due to read operations.

Fig. 7 shows the I/O profile for `sasenna_ID0000005` jobs. Write operations (Fig. 7-*left*) present a particular behavior of writing significant amounts of data in a regular time interval. Moreover, `atoms_H` writes about twice as much data as `atoms_L`. In contrast to the previous analysis for the NAMD jobs, here the increase in the number of atoms seems to enable an iteration process (e.g., a

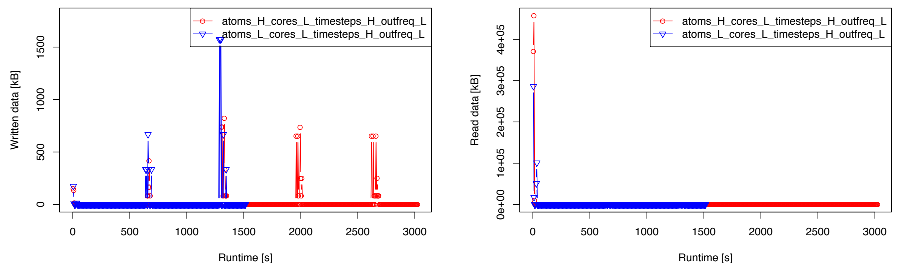
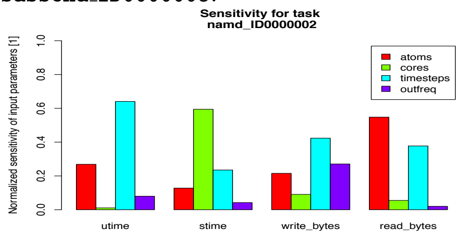
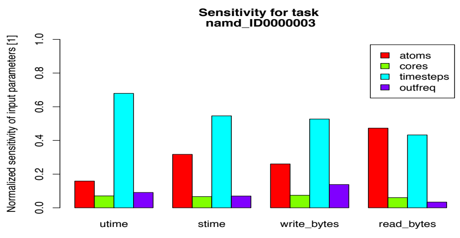


Fig. 7. Performance profile of write (left) and read (right) operations for `sassena_ID0000005`.



(a) `namd_ID0000002`



(b) `namd_ID0000003`

Fig. 8. Cumulative normalized sensitivity of the NAMD jobs in the SNS workflow.

loop condition). Read operations (Fig. 7-*right*) present a similar behavior, but scaled up to the configuration with the higher number of atoms.

Sensitivity analysis. The main effect of an input parameter on a response is calculated with Eq. 1 as a difference between the average response for high and low values of the input.

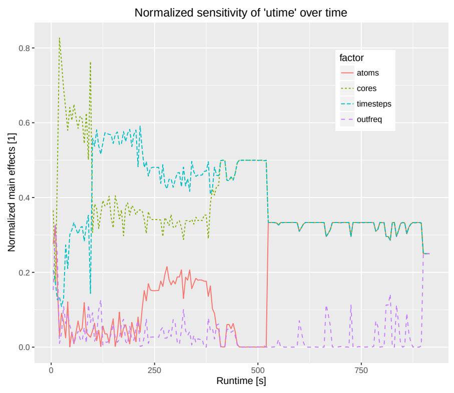
$$E_k = |R_{k,H} - R_{k,L}| \tag{1}$$

where $R_{k,H}$ denotes averaged simulation output (in our case resource consumption) when input parameter k is set to high values H. This difference should be interpreted as a change in the response due to a change in the input. We use main effects to describe how input parameters influence workflow performance. Fig. 8 shows the normalized sensitivity for workflow jobs calculated as:

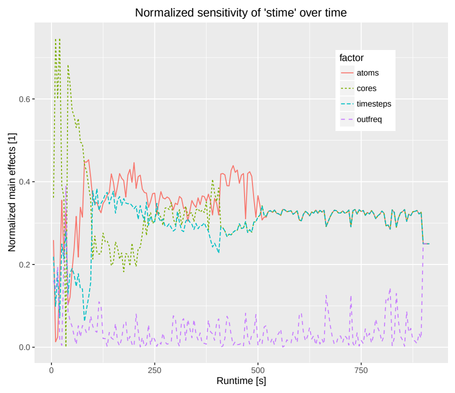
$$S_k = \frac{E_k}{\sum\limits_i E_i} \tag{2}$$

where k denotes an input parameter, E_k is the effect of the input parameter on the model output measure at the end of job's execution. It doesn't have a unit since it is a ratio between the effect of a single parameter to summarised effect of all parameters. NAMD jobs (Fig. 8a and 8b) are mostly influenced by the number of simulation steps and the type of material used in the simulation.

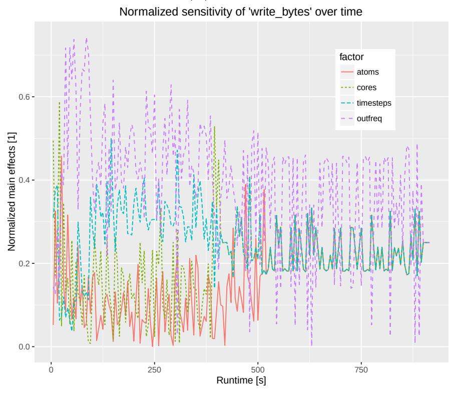
In order to provide enhanced sensitivity analysis of workflow performance profiles, we compute sensitivity over time, i.e. the impact of each input parameter on the application response during task runtime. It is still calculated with Eq.



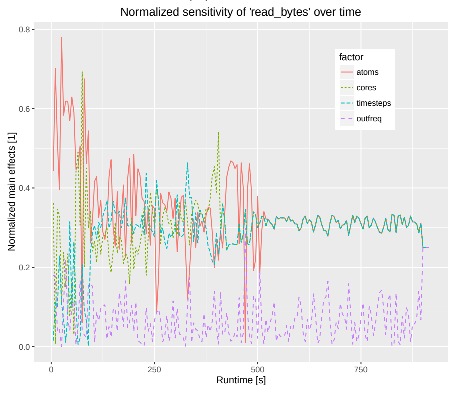
(a) `utime`



(b) `stime`



(c) `write_bytes`



(d) `read_bytes`

Fig. 9. Normalized sensitivity over time for `namd_ID0000002` (calculated with Eq. 2 for each monitoring period).

2 however this time we calculate sensitivity for each monitoring period. Due to limited space, we only include results for the `namd_ID0000002` job (Fig. 9). For `utime` (Fig. 9a), the number of cores has significant impact at the beginning of the execution. However, the number of timesteps becomes more influential along the execution. For runs above 500s, there is not much variation. This behavior is due to the small number of jobs with longer runtime. In Fig. 9b, `stime` is mostly influenced by the material type (`atoms`), however `cores` has important impact at the beginning of the execution. Note that the output writing frequency has no significant impact for time-related metrics. Not surprisingly, `write_bytes` is substantially influenced by `outfreq` (Fig. 9c) and the number of timesteps. The material type (`atoms`) becomes irrelevant after the initial phase, and the number of cores does not drive any influence. In contrast, `read_bytes` (Fig. 9d) is heavily influenced by `atoms`—as it is related to the amount of data read from input files. The influence of `timesteps` and `cores` increases along the execution.

In summary, the main behaviors identified in this analysis include: 1) linear correlations between the amount of computations and job runtime in the `namd` jobs; 2) accumulation of I/O operations in the first stage of the execution in the `namd` jobs; and 3) periodic data dumps in the `sassena` jobs.

4 Related Work

Workflow profiling analysis is often used to drive advancements on workflow optimization studies, including job scheduling and resource provisioning. For instance, in [5, 17] workflow profiles are used to model and predict execution time of workflow activities on distributed resources. In [3,6], heuristics and models are developed from workflow profiles to estimate the number of resources required to execute a workflow. We recently used profiling data from Pegasus workflows to estimate job resource consumption on distributed platforms [20]. Although our techniques yield satisfactory estimates, our studies were limited to the aggregated performance information, i.e. no time series analysis was considered. Several papers have profiled scientific workflow executions on real platforms [7, 12], however none of them have collected time series data from workflow executions at the job level. Workload archives [15] are used for research on distributed systems, e.g. to evaluate methods in simulation or in experimental conditions. Although the data is collected at the infrastructure and application level, the gathered data is also limited to aggregated performance information. To the best of our knowledge, this is the first work that builds and analyzes workflow profiles based on time series data collected from real workflow executions.

5 Conclusions and Future Work

In this paper, we described a generic approach for the development and analysis of workflow performance profiles, which describes application resource consumption over time. Such profiles provide much more information than the aggregated information given at the end of the execution. The presented approach is comprehensive, i.e. it takes into account the processes of generating, preparing, and analysis of data. It is independent of the analyzed workflow and can be used with existing, large-scale HPC infrastructures. The proposed approach was validated with a real-life workflow from material science running on a TOP500 machine. The analysis conducted unveiled useful insights about the workflow regarding the effect of input parameters on task performance.

As part of future we will integrate the proposed solution with Scalarm and Pegasus to minimize workflow runtime by improving job scheduling onto distributed resources based on information extracted from performance profiles, e.g. to identify which tasks can be executed in parallel on the same resource without performance disruption. We will also use the PaaSage framework to deploy and manage both tools and to run scientific workflows on cloud resources in a cost-effective way.

Acknowledgments. This research was supported by DOE under contract #DE-SC0012636, "Panorama–Predictive Modeling and Diagnostic Monitoring of Extreme

Science Workflows". D. Król thanks to the EU FP7-ICT project PaaSage (317715) and Polish grant 3033/7PR/2014/2.

References

1. Arnold, O., et al.: Mantid - data analysis and visualization package for neutron scattering and {SR} experiments. Nuclear Instruments and Methods in Physics Research Section A, 764, 156 – 166 (2014)
2. Borreguero, J.M., Lynch, V.E.: Molecular dynamics force-field refinement against quasi-elastic neutron scattering data. Journal of Chemical Theory and Computation 12(1) (2016)
3. Byun, E., Kee, Y., et al.: Estimating resource needs for time-constrained workflows. In: eScience '08. IEEE 4th Internat. Conf. on eScience (2008)
4. Deelman, E., Vahi, K., et al.: Pegasus, a workflow management system for science automation. Future Generation Computer Systems 46 (2015)
5. Duan, R., Nadeem, F., et al.: A hybrid intelligent method for performance modeling and prediction of workflow activities in grids. In: 9th IEEE/ACM Internat. Symp. on Cluster Computing and the Grid (2009)
6. Huang, R., Casanova, H., et al.: Automatic resource specification generation for resource selection. In: SC '07. 2007 ACM/IEEE Conf. on Supercomputing (2007)
7. Juve, G., Chervenak, A., et al.: Characterizing and profiling scientific workflows. Future Generation Computer Systems 29(3) (2013)
8. Juve, G., Tovar, B., et al.: Practical resource monitoring for robust high throughput computing. In: 2nd Workshop on Monitoring and Analysis for High Performance Computing Systems Plus Applications (2015)
9. Król, D., Kitowski, J.: Self-scalable services in service oriented software for cost-effective data farming. Future Generation Computer Systems 54 (2016)
10. Kvassay, M., et al.: A novel way of using simulations to support urban security operations. Computing and Informatics 34(6), 1201–1233 (2015)
11. Lindner, B., Smith, J.C.: Sassena—x-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers. Computer Physics Commun. 183(7) (2012)
12. Mayer, B., Worley, P., et al.: Climate science performance, data and productivity on titan. In: Cray User Group Conf. (2015)
13. NERSC: Hopper. https://www.nersc.gov/users/computational-systems/hopper
14. FP7 PaaSage project website. http://www.paasage.eu/, accessed 10/05/2016
15. Parallel workloads archive. http://www.cs.huji.ac.il/labs/parallel/workload
16. Phillips, J.C., Braun, R., et al.: Scalable molecular dynamics with namd on the ibm blue gene/l system. IBM J. of Research and Development 26(1.2) (2008)
17. Pietri, I., Juve, G., et al.: A performance model to estimate execution time of scientific workflows on the cloud. In: Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science (2014)
18. Salomon-Ferrer, R., et al.: An overview of the amber biomolecular simulation package. Wiley Interdisciplinary Rev.s: Computational Molecular Science 3(2) (2013)
19. Saltelli, A., Ratto, M., et al.: Global Sensitivity Analysis: The Primer (2008)
20. Ferreira da Silva, R.., Juve, G., et al.: Online task resource consumption prediction for scientific workflows. Parallel Processing Letters 25(3) (2015)
21. Taylor, I.J., et al.: Workflows for e-Science: scientific workflows for grids (2007)