Capturing provenance as a diagnostic tool for workflow performance evaluation and optimization

Line Pouchard
Computer Science Initiative
Brookhaven National Laboratory
Upton, NY
pouchard@bnl.gov

Abid Malik
Computer Science Initiative
Brookhaven National Laboratory
Upton, NY
amalik@bnl.gov

Huub Van Dam
Computer Science Initiative
Brookhaven National Laboratory
Upton, NY
hvandam@bnl.gov

Cong Xie
Department of Computer Science
Stony Brook University
Stony Brook, NY
xiecng@gmail.com

Wei Xu
Computer Science Initiative
Brookhaven National Laboratory
Upton, NY
xuw@bnl.gov

Kerstin Kleese Van Dam
Computer Science Initiative
Brookhaven National Laboratory
Upton, NY
kleese@bnl.gov

Abstract – **In extreme-scale computing environments such as the DOE Leadership Computing Facilities scientific workflows are routinely used to coordinate software processes for the execution of complex, computational applications that perform in-silico experiments. Monitoring the performance of workflows without also simultaneously tracking provenance is not sufficient to understand variations between runs, configurations, versions of a code, and between changes in an implemented stack, and systems, i.e. the variability of performance metrics data in their historical context. We take a provenance-based approach and demonstrate that provenance is useful as a tool for evaluating and optimizing workflow performance in extreme-scale HPC environments. We present Chimbuko, a framework for the analysis and visualization of the provenance of performance. Chimbuko implements a method for the evaluation of workflow performance from multiple components that enables the exploration of performance metrics data at scale.**

Keywords – provenance, performance, scientific workflows, workflow performance provenance ontology, WFPP, Chimbuko

I. Introduction

In extreme-scale computing environments such as the DOE Leadership Computing Facilities [1] scientific workflows are routinely used to coordinate software processes for the execution of complex, computational applications that perform in-silico experiments. Workflows enable the orchestration of the numerous processes that read, write, and analyze data and calculate quantities of interest for parallel and distributed scientific applications that range from quantum chemistry, molecular dynamics (MD), climate modeling, and many others. Important factors for any application running in such environments include execution time and performance, accuracy of calculations, and the ability to analyze results. Given the limitations of a fixed resource budget (number of cores allocated for a specific period of time) and with simulations running for several days or weeks, it is important to determine if a simulation run is progressing as expected, what variations in performance a run exhibits and where they can be attributed. Monitoring the performance of workflows in HPC provides insights into this progression, how the computational resources are used, and where execution bottlenecks occur. But monitoring performance without also simultaneously tracking provenance is not sufficient to understand variations between runs, configurations, versions of a code, and between changes in an implemented stack, and systems, i.e. the variability of performance metrics data in their historical context. For gaining this type of insights, the provenance of workflow performance is needed. We define the provenance of workflow performance as the provenance that captures and correlates traditional provenance characteristics and performance metrics data. This type of provenance is used for performance analysis in empirical studies on the performance of a software or workflow during a development phase or in different computational environments.

Scientific workflows can play an important role in helping scientists coordinate complex tasks and take better advantage of the computing resources available to them by decoupling the composition of tasks from workflow instance executions and their actual environment [2,3]. Scientific workflows have been extensively studied and numerous workflow management systems are in circulation [4]. However, many research topics remain unexplored, including the real-time monitoring of scientific workflow processes to validate performance and support data re-use and reproducibility [5]. Defining the appropriate level of abstraction for monitoring given the number of components, the complexity of the connections and the rate of execution for each component remains a challenge that we start addressing here with a method and a prototype framework for studying the provenance of workflow performance.

Notice: This manuscript has been authored by employees of Brookhaven Science Associates, LLC under Contract No. DESC0012704 with the U.S. Department of Energy. The publisher by accepting the manuscript for publication acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

Specific aims and scope

Our goal is to show that provenance can be used to improve the performance of workflows in scientific codes and facilitate optimization in code development. We take a provenance-based approach and demonstrate that provenance is useful as a tool for evaluating and optimizing workflow performance in extreme-scale HPC environments. In particular, in forensic analysis of workflow performance metrics, we aim to elucidate the factors present during certain time periods of a run when performance does not achieve as expected. Sluggish performance can be associated with communication between parts of a workflow, data movement, and interdependencies in the sequence of processes related to regions in the software code that directs flow and interaction, and many other factors. It can be associated with parts in the system architecture, for instance the network interconnect between nodes, and with the input and output of data. By capturing metrics related to these factors and correlating them to versions, configuration parameters, input data, system software stacks and other provenance characteristics, we enable comparison between runs.

Fully capturing the runtime environment for a scientific workflow or application (i.e. the complete provenance of a run) requires the ability to capture system variables at a granular level not usually available to end users on shared and managed systems. Existing tools for such capture [6,7] are developed to support operations of LCF systems, including Titan, Cori, and others. Although it is theoretically possible to capture a user runtime environment with these tools by querying their databases, access is not always available as these data may be deemed too sensitive for open use. In this paper we focus on the provenance metrics and characteristics that we can access, such as execution times at the call path level and job information.

We present a method for the evaluation of workflow performance that enables the comparison of performance metrics data. We provide a use case with two different kinds of workflows implemented with small modifications in NWChem [8]. This is innovative because 1) provenance is used in conjunction with performance measuring tools; 2) this combination is applied to workflows rather than single applications; 3) we provide web-based visualization that enables both high level and detailed visualization of the performance data; 4) provenance metadata is linked to performance metrics; and 5) we demonstrate that provenance is a critical tool for workflow performance evaluation and optimization in extreme-scale environments. This work is part of the Co-design center for Online Data Analysis and Reduction (CODAR) [9].

II. Methods

We design and implement Chimbuko, a framework for capturing, integrating, and visualizing the provenance of performance that implements our method. Our method relies upon four components each examined in details below:

1) We present a use case for NWChemEx, the next generation of the NWChem computational chemistry and materials code currently starting a new release cycle. For this use case we instrument NWChem routines run in parallel and serial fashions.

2) We rely on the Workflow Performance Provenance ontology (WFPP), an ontology based upon W3C PROV and the Open Provenance Model to classify performance and provenance metrics [10].

3) We use a state-of-the-art performance metrics toolkit to instrument our use case [11]. And we extract and integrate metrics of interest for the scientific use case.

4) We display performance metrics with provenance for workflows in a visualization environment that scales.

The Chimbuko framework captures, integrates, analyzes and visualizes performance metrics for complex scientific workflows and relates these metrics to the context of their execution on extreme-scale machines. The purpose of Chimbuko is to enable empirical studies of performance analysis for a workflow during a development phase or in different computational environments. Chimbuko enables the comparison of different runs at high and low levels of metric granularity. Chimbuko currently provides this capability in offline mode with plans to extend to online (in-situ) modes with data reduction. Chimbuko encapsulates the numerous steps in our method and its design highlighted the needs for wrappers integrating information between the various components of our system.

A. *Use case: NWChemEx*

NWChemEx [12] is being developed as a set of scientific codes for simulating the dynamics of large scale molecular structures and materials systems on large atomistic complexes. The final goal is a capability that supports *ab-initio* as well as MD calculations, although phase I of the project focusses on *ab-initio* capabilities. For the MD module the aim is to calculate trajectories for about a million atoms at a time resolution of a few femtoseconds collecting statistics equivalent to a simulation time of at least a microsecond. All codes are distributed data parallel programs. The programming paradigm is currently envisioned to be MPI/OpenMP exploiting shared memory within the node. NWChemEx provides an interesting use case for provenance as it needs to demonstrate performance improvements with new code versions and similar performance gains on various LCF systems of different architectures. Although NWChemEx is a single application, it is expected to provide energy and force calculations for trajectories of interest in molecular dynamics simulations, parallelized over many cores and nodes. As a single resulting trajectory would generate 32 Terabyte of data even when storing only 1 of every 1000 timesteps this is still very large to be stored for offline analysis. Therefore, we envision an approach where the MD simulation is run, emitting snapshots of the protein structure along the trajectory, and where the data analysis is run concurrently consuming the data as it is produced. Ultimately, such an approach could avoid the need for the data to be stored at all. To demonstrate what is involved in such an approach we ran NWChem molecular dynamics on a small protein where we computed the trajectory parallelizing each time-step using global arrays, and concurrently another NWChem process performed the analysis. For this demonstration a particularly simple analysis option was used, namely the root-mean-square deviation of the coordinates with respect to a point. Our use case is intended to explore the capture and use of provenance that is of interest to NWChemEx while simultaneously highlighting the challenges of such capture for our intended use. These challenges included instrumenting the code, extracting and compiling useful metrics, and visualizing results in usable manner.

The information needed to track the provenance of performance in NWChemEx includes strictly performance-related metrics such as the total execution time per workflow, per node, and per code region, the call tree, communication and interconnect performance, and number and volume of I/O reads and writes. In addition, needed characteristics for provenance include the application name, version, name of who ran the application, SVN branches and revision numbers, time compiled, time started, and wall time and number of processors allocated.

An important component of the NWChemEx project is to demonstrate scalability on the future exa-scale platforms as well as performance portability and of course efficient use of resources. These characteristics will be captured in, so called Key Performance Parameters (KPP). The parameters will be measured on the current code and on the new NWChemEx code and set levels of improvement must be met for the project to be successful. Obviously this requires tracking which codes are being compared, for which information on the SVN branches and revision numbers, time compiled, and time started are important. Critical here is that the a test always tests the whole system, i.e. the hardware, the compilers, the libraries, and the code of your project. As problems can originate in any of these components it is important to capture information not only about the code itself but also information that can be correlated with the state of the computing platform. In addition to measure the impact of the code development work the time taken by different code regions is crucial as changes there will be related to changes in the algorithms or their implementations. Even in cases where the total volume of compute does not change significantly improvements may be obtained by better distributions of the workload hence imbalance in the workload is important to track. Communication, whether by message passing, through files, or yet some other mechanism are always critical to the parallel performance of a code. In all these cases it is essential to record these performance characteristics in such a way that improvements in a code's performance can be attributed to particular coding efforts. This is required to ensure that modifications that successfully improve the performance are selected and kept on the basis of solid empirical evidence for improved performance.

B. *Workflow Performance Provenance Ontology (WFPP)*

WFPP [9] aims to inventory all possible metrics needed for relating provenance to workflow performance. Different metrics are needed for different scientific applications and purposes of capturing performance provenance. Metrics related to communication time and volume, FLOPS per process, wait in barrier, energy usage, memory usage, cache access and misses, page faults, I/O number of reads and writes and volume of data moved are of interest for NWChemEx. On GPUs, divergence of conditional statements will be needed and pose additional challenges. Our experiment focuses on execution times, location and call paths as a starting point. In addition, provenance characteristics included in job submission scripts and user profiles is collected and related to workflow execution.

The use of WFPP to specify provenance is crucial to determine both the required metrics and the level of granularity for these metrics. Taking into account that users may pursue different performance optimization goals WFPP entities specify characteristics allowing them to study trade-offs between different goals. For instance, quality-related measures such as accuracy or adaptability are included along performance metrics such as total execution time of a workflow instance, allowing a user to prioritize their optimization goals. In our use case, WFPP allows a detailed exploration of the needed metrics (execution time and communication) by providing an explicit structure for specifying them in details and extracting them and their meaning from the performance tools, as well as highlighting the gaps in the tools.

C. *Performance tools*

While numerous performance tools for single applications exist, the ability to easily and comprehensively measure the performance of workflow instances and components at appropriate levels of details for the exascale is lacking, and no existing utility monitors the performance of workflows. In this study, we evaluated ScoreP and TAU (Tuning and Analysis Utilities), two state-of-the-art performance tool suites for their ability to extract performance for our use case and their potential development for monitoring the performance of workflows. Tools for distributed and parallel problems such as the TAU (Tuning and Analysis Utilities) emphasize flexibility in the empirical methods chosen for performance instrumentation and portability across platforms and programming models for single applications [10]. TAU's flexibility makes it a good choice for instrumenting NWChemEx while its design for portability makes it a good candidate to demonstrate performance monitoring across platforms. The new developments for TAU to support workflows are described in Findings.

D. *Chimbuko Provenance Framework*

We designed Chimbuko as a multi-view, provenance framework to capture, integrate, and persist workflow performance in the context of their provenance at the exascale. Chimbuko allows different perspectives on the performance of workflow components that can be harnessed by scientific users and by runtime systems developers. The Chimbuko framework is illustrated in Figure 1. Chimbuko includes the performance analysis tools (TAU in our implementation), the ProvEn provenance framework, visualization and analysis modules. Figure 1 shows the basic layout of the Chimbuko Framework. The framework can be coupled with a number of performance analysis tools. The workflow application can be instrumented manually or automatically using a performance tool. For the off-line mode, information is collected during a run of a workflow and analyzed when the execution is over. Performance analysis tools dump information in various format for profiling and tracing. The information management unit of the framework makes sure that the information is stored in the central databases in the same format. The information management system consists of various scripts that help a user transfer the collected data into the JSON format. Section III.B talks about the information management system in more details. The collected information is stored in hybrid databases maintained by the ProvEn [13]. ProvEn consists of a harvester, a triple store, and a time series database [14]. The visualization component of the framework provides the spatial and temporal resource utilization feedback to the user. The feedback helps improve both the computational and resource efficiency of a given workflow application. Section III.C provides details about the visualization component. The data analysis part of the framework is used to analyze computational and communication patterns in a given scientific workflow. These patterns can be used to improve the performance of workflow. Currently, we are using this component for a data reduction problem.

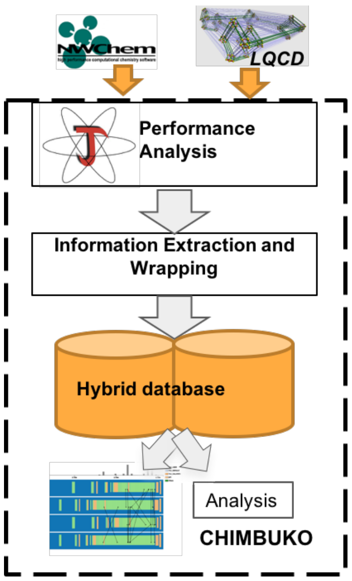


Figure 1: Components of the Chimbuko framework

E. *Visualization*

In order to enable a comprehensive understanding of performance for heavy computational applications with provenance, especially in terms of workflows, performance visualization is essential and facilitates the following aspects [15]: 1) visualizing different types of performance data such as counters, traces, profiles and call paths, 2) satisfying the requirements of performance analysis with different goals such as global comprehension, problem detection and diagnosis, and 3) supporting various performance contexts such as hardware (cluster node or network), software (trace, call graph or source code structure), and others.

The visualization challenge is to accommodate large amount of information into limited display resolution, while still enabling detailed exploration of interesting pieces of data. We solve that issue by designing a visualization tool enabling level-of-detail exploration and user interaction that is an integral part of Chimbuko. The purpose of the tool is not to cover every detailed functionality of existing tools in the TAU analysis suite (e.g. Vampir, ParaProf or Jumpshot). But instead, we enable what is missing – the capability to visualize and analyze the performance of the workflow execution for multiple applications.

III. Findings

In instrumenting NWChem and visualizing workflow performance in its provenance context we encountered challenges related to the usability of the performance tools (choosing the appropriate instrumentation for this scientific application, the configuration of the performance tool to obtain meaningful output, and attributing metrics to their source), and given the volume of traces, the selection of appropriate resolution for the visualization. Our Chimbuko framework that includes new developments for integrating information produced by TAU attempts to answer these challenges.

A. *Usability of Output: Traces and Profiles*

We address the issue of usability for the performance tools by collaborating closely with the TAU team, producing agreed upon design documents and work plans, and repeated iterations of the experimental design. Our experiment demonstrated that capturing provenance is a very useful diagnostic tool for the chosen use case as it highlighted performance variations between different calculations. Capturing application or workflow traces is not sufficient in itself to understand problem locations in hardware or software and relate them to specific runs. Only when provenance is included with performance metrics can traces be compared in their historical and execution context. The need to understand tradeoffs in execution highlights requirements for persisting provenance leads to the development of a data model to translate WFPP entities into queriable and extractable elements.

Obtaining meaningful output from the TAU performance tool with the given NWChem runs proved challenging beyond expectations, in spite of the availability to us of the TAU development team to answer questions. TAU outputs 2 kinds of files, a trace file monitoring execution of code regions on nodes and a profile file summarizing this information and displaying the call tree. New developments in Chimbuko were needed to make use of the performance metrics output by TAU and attribute them to their source function. These metrics were obtained by executing the dynamics and analysis concurrently for an NWChem MD calculation in an attempt to simulate a concurrent workflow situation. The MD calculation loops until a trajectory file appears. At that point the analysis workflow is started. One major challenge encountered for provenance in this scenario is to attribute the metrics output by the tool to its origin. For instance, while a simple subtraction allows to extract execution time for an MPI call on a node, one still needs to know what MPI call we are reading. A trace file was also obtained; the size of this Trace file is 224MB in binary format, 812MB in text format, and represents a classical MD run of 320 timesteps on 4 cores as well as the corresponding analysis of 100 timesteps on 1 core. From start to finish the MD run took 38.0 seconds wall clock time. In addition, the program was compiled to suppress the instrumentation of all subroutines apart from the main MD and analysis routines.

B. *The Chimbuko Information Management System: Relating Metrics to Meaning*

Chimbuko is collecting and summarizing fine level information while preserving the ability to query at the fine granular-level. Figure 3 presents the collection of workflow performance metrics for each component C in the Chimbuko information management system. A typical scientific workflow consists of a number of components/applications which interact with each other for data management and communication. In the current implementation, Chimbuko uses the TAU infrastructure to collect information. Currently, TAU is application specific. Each component is compiled and instrumented using wrappers designed with information extracted from metrics. When the instrumented code is run during the workflow execution, the profile and trace information is collected. TAU provides runtime variables that can be used to collect information in separate directories for each component. The profile information is dumped in a tabular text format, and the trace information is collected using Open Trace Format (OTF). To facilitate the visualization and data analysis components, new wrappers are developed to manipulate the collected data.

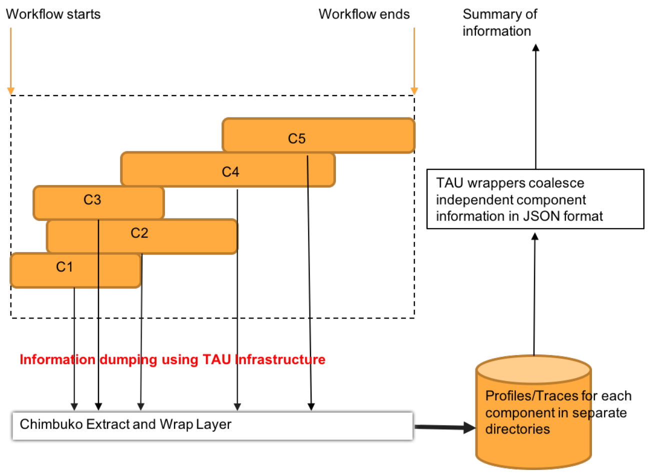

Workflow starts
Workflow ends
Summary of information
C5
C4
C3
C2
C1
TAU wrappers coalesce independent component information in JSON format
Information dumping using TAU infrastructure
Chimbuko Extract and Wrap Layer
Profiles/Traces for each component in separate directories

Figure 2: Workflow performance metrics capture

The wrappers designed can be used:

- To coalesce the information dumped in separate directories from each component. This helps in comparing and visualizing the performance of different components in the same temporal window.
- To convert the profile tabular text format into JSON format.
- To convert the OTF into JSON format.
- To summarize the information at the workflow level.

Chimbuko collects a comprehensive list of performance metrics which can be used for various performance analysis problems, e.g. performance scalability, resilience, better memory management, and runtime optimizations. Figure 3 shows a subset of these features.

```
{
        "Work_flow_performance":{
                        "execution_time": 58.8989
                               },
        "Detail_of_each_Component":[
               {
                        "name" : "heat_transfer.F90",
                        "processors": 12,
                        "aggr_execution_time": "12:22.279",
                        "aggr_communication_time": "11:47.525",
                        "idle_time": "11,132"
                       },
               {
                        "name": "stage_write.c",
                        "processors": 2,
                        "aggr_execution_time": "1:59.244",
                        "aggr_communication_time": "1:40.141",
                        "idle_time":168
                        }
        ],
        "Intra_component_detail":{
                        "volume_of_communication": "3.204E+04 x 6",
                        "number_of_messages": 6
                               }
}
```

Figure 3: Subset of performance metrics one can collect with Chimbuko

C. *Visualization Results*

In this work, we aim to visualize the measurements of instrumentation on our use case -- a parallel workflow designed with NWChemEX. Our acquired data are individual trace and profile files capturing the execution of independent workflow components. It includes several types of data: 1) event table listing the start and end time of all function calls, 2) the messages passing among functions, 3) profiling of certain metrics for time spent in each code region on each computing node/thread, and 4) the call path for each node/thread. In order to connect them together, we devise a data model for each workflow that includes the following information: 1) the overall structural description of the workflow, 2) the metadata about the workflow, 3) the connected trace events of the entire workflow, and 4) the connected profiling of the entire workflow. Thus, in order to explore all the above information, we devise and develop a level-of-detail multiple channel visualization framework with front-end plotting of the data and back-end performing of the necessary analysis and computation.

Visualization for the Chimbuko framework includes four major components: overview, detailed view, statistical view, and profile view that together establish an interactive analysis platform to visually explore and analyze the performance of a workflow. Overview shows the summary of the whole workflow execution as in Figure 4 (top).

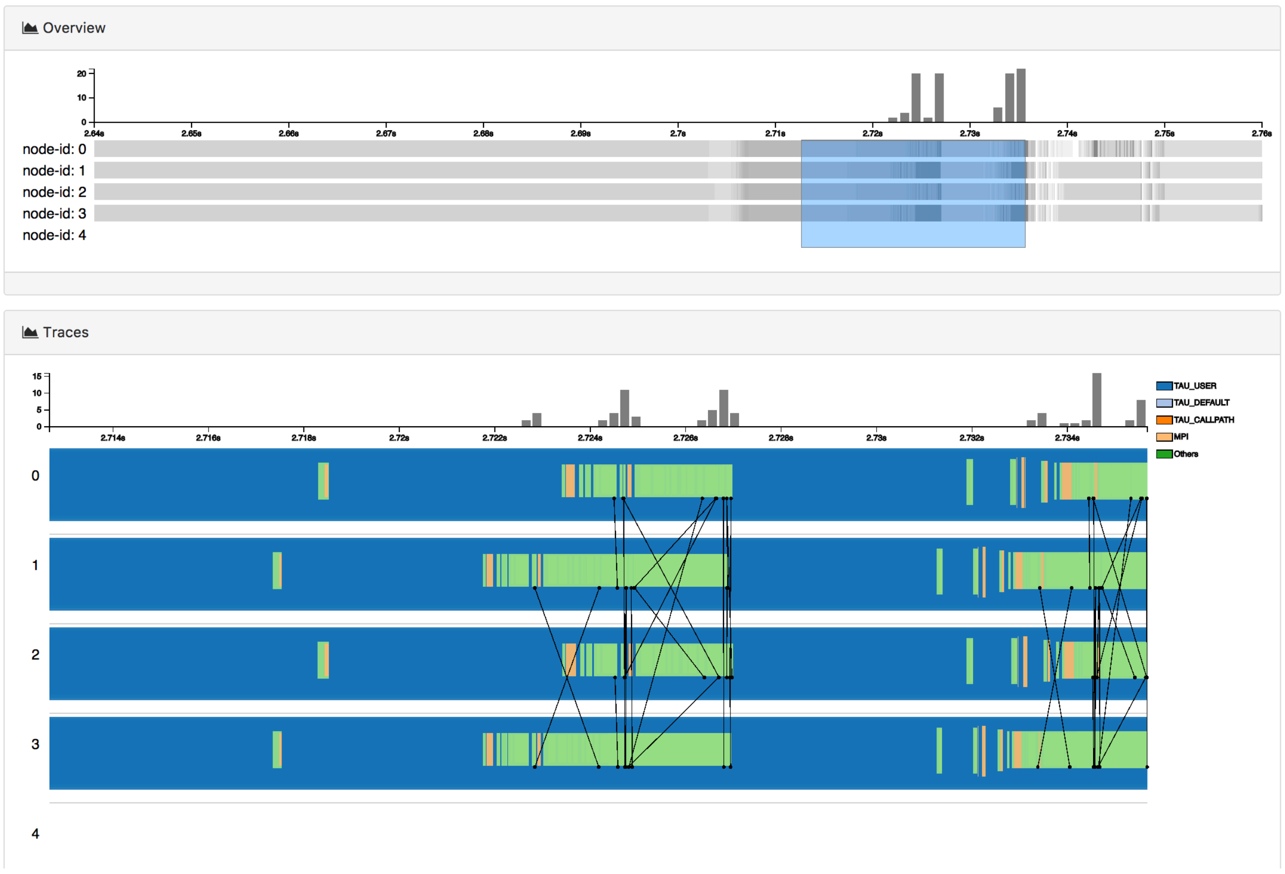
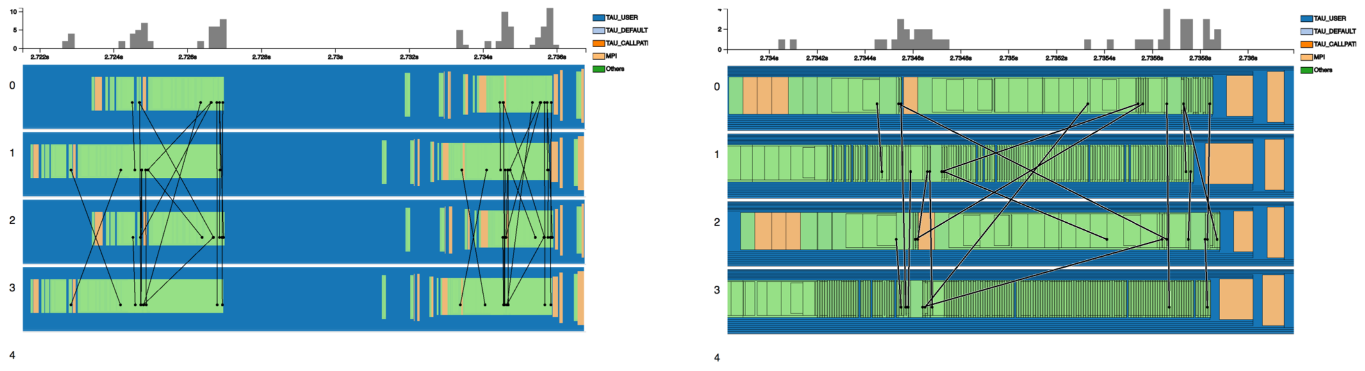


Figure 4: Visualization of trace performance for our use case: the overview panel (top) and the detailed view of the selected region (bottom)



Figures 5a and 5b: Visualization of trace performance for our use case: (a) zoom-in effect, (b) enhance the stroke of rectangles.

There are two parts in the Overview: the trace events, and the message counts. The trace events indicating the start and end time of each function call are shown as timeline. We use intensity to indicate the depth of the call path. A darker color represents a more nested function call. For each node/thread, the trace events are plotted separately. Above the timelines, we also visualized the message counts (sent or received) in a separate histogram view along the timeline. For the interaction, it allows the user to select a time range of interest and see more details in the detailed view panel. In this example, we visualized a parallel workflow running on five nodes/threads.

The detailed view shows the function calls and the messages in the selected time range as in Figure 4 (bottom). Each function call is visualized with a rectangle whose color represents a different call group. The functions are visualized with nested rectangles that indicate their depths in the call path. This fashion is similar to what Jumpshot utilized. In this view, users can still zoom in to explore more details by selecting a smaller time range (Figure 5a). When there are many short function calls in a nested call structure, it can be difficult to observe small events and differentiate each call. Therefore, we designed two features for that issue. First, we use different transparency to enhance the visibility of overlapping functions. Second, when zooming in, we add the stroke of the rectangle to enhance the separation of different functions (Figure 5b).

For the message visualization, we visualized the message passing (send and receive) between functions as straight black lines (Figures 4 and 5). As being organized in a timeline, the line direction is ignored since the message is always passing from left to right. Additionally, when hovering over each rectangle, detailed function name can be seen in text.

The statistical view summarizes the time spent for each function call in the selected region. In this aggregated visualization, it is easier to compare the time difference side by side. We also adjust the text display in order to avoid clutter issue. In the Profiles view, with the selected metric (time or counter), we visualize the percentage spent on each function for each node/thread.

In summary, we devised a visualization platform for workflow performance evaluation. We utilized nested bar charts and stacked graphs to represent events in terms of timeline. We also connect that to profile in the form of stacked bar graphs to reveal the corresponding statistics of the chosen metric. In order to support the scalability, we implemented a few types of visualization with different levels of details -- overview, zoom-in with transparency, and zoom-in with separation enhancement. The message communication is also visualized by line connections and message count histograms.

D. Future Work

In-situ analytics has been shown to be an effective approach to reduce both I/O and storage costs for scientific analytics. Developing an efficient in-situ implementation, however, involves many challenges, including parallelization, data movement or sharing, and resource allocation. We aim to overcome these challenges using our framework.

Many large-scale scientific applications are usually constructed as workflows due to large amounts of interrelated computation and communication. Workflow scheduling has long been a research topic in parallel and distributed computing. However, most previous research focuses on single workflow scheduling. As cloud computing emerges, users can now have easy access to on-demand high performance computing resources, usually called HPC cloud. Since HPC cloud has to serve many users simultaneously, it is common that many workflows submitted from different users are running concurrently. Therefore, how to schedule concurrent workflows efficiently becomes an important issue in HPC cloud environments. We are aiming to extend the framework to optimize the scheduling strategy for concurrent workflows.

The performance portability is a big issue for scientific workflows. A given workflow has portable performance if in addition to running on diverse platforms it exhibits similar accuracy, stability, and reliability across these platforms for a given configuration. Moreover, the time to solution should reflect efficient utilization of available computational resources on each platform. We plan to extend the framework to include an adaptive runtime system that will help ensure performance portability of scientific workflows across different computing frameworks.

For visualization, we will implement a Sankey diagram to show data I/O between workflow components. Then, we will redesign the Chord diagram to support message communications between threads. Finally, we will improve the scalability with data reduction and try more complicated workflow types.

IV. Conclusion

Our method marshals in data from different sources required to identify barriers, compare seemingly alike executions, and enable visualization of workflow performance trace. Integration of provenance with performance analysis tools brings elements of response to questions related to the resources needed to complete a workflow in time and the trade-offs between workflows. As our experiment develops to address increasingly complex workflows, the use of a workflow management system may be needed to manage the workflow graph. In the future, we will also enhance the capability with the help of advanced analysis routines in the back-end.

Acknowledgement

This research was supported by the Exascale Computing Project (ECP), a collaborative effort of two DOE organizations – the Office of Science and the National Nuclear Security Administration. The Project Number for the Co-design center for Online Data Analysis and Reduction (CODAR) that supported this research is 17-SC-20-SC.

References

[1] US Department of Energy Advanced Scientific Computing Research: Leadership Computing Facilities. Available from https://science.energy.gov/ascr/facilities/.

[2] Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., and Zhao, Y.: 'Scientific workflow management and the Kepler system', Concurrency and Computation: Practice and Experience, 2006, 18, (10), pp. 1039-1065

[3] Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., and Good, J.: 'Pegasus: A framework for mapping complex scientific workflows onto distributed systems', Scientific Programming, 2005, 13, (3), pp. 219-237

[4] Garijo, D.: 'AI buzzwords explained: scientific workflows', AI Matters, 2017, 3, (1), pp. 4-8

[5] Deelman, E., Peterka, T., Altintas, I., Carothers, C.D., Kleese van Dam, K., Moreland, K., Parashar, M., Ramakrishnan, L., Taufer, M., and Vetter, J.: 'The future of scientific workflows', The International Journal of High Performance Computing Applications, 2017, pp. 1094342017704893

[6] Agrawal, K., Fahey, M.R., McLay, R., and James, D.: 'User environment tracking and problem detection with XALT', in Editor (Ed.)^(Eds.): 'Book User environment tracking and problem detection with XALT' (IEEE Press, 2014, edn.), pp. 32-40

[7] Huang, R., Xu, W., and McLay, R.: 'A web interface for XALT log data analysis', in Editor (Ed.)^(Eds.): 'Book A web interface for XALT log data analysis' (ACM, 2016, edn.), pp. 31

[8] M. Valiev, E.J. Bylaska, N. Govind, K. Kowalski, T.P. Straatsma, H.J.J. van Dam, D. Wang, J. Nieplocha, E. Apra, T.L. Windus, W.A. de Jong, 'NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations' Computer Physics Communications 181, 1477 (2010) doi:10.1016/j.cpc.2010.04.018

[9] Foster, I., Ainsworth, M., Allen, B., Bessac, J., Cappello, F., Choi, J.Y., Constantinescu, E., Davis, P., Di, S., Di, W., Guo, H., Klasky, S., Dam, K.K.V., Kurc, T., Malik, A., Mehta, K., Mueller, K., Munson, T., Ostouchov, G., Parashar, M., Peterka, T., Pouchard, L., Tao, D., Tugluk, O., Wild, S., Wolf, M., Wozniak, J., Xu, W., and Yoo, S.: 'Computing Just What You Need: Online Data Analysis and Reduction at Extreme Sca'. Proc. EuroPar2017

[10] Kleese van Dam, K., Stephan, E.G., Raju, B., Altintas, I., Elsethagen, T.O., and Krishnamoorthy, S.: 'Enabling Structured Exploration of Workflow Performance Variability in Extreme-Scale Environments'. Proc. 8th Workshop in Many-Task Computing on Clouds, Grids, and Supercomputers (MTAGS) collocated with SC 2015, Austin, TX2015

[11] Shende S, Malony A. Using TAU for performance evaluation of scientific software. In: Allen G, Carver J, Choi SCT, Crick T, Crusoe MR, Gesing S, et al., editors. Workshop on Sustainable Software for Science: Practice and Experiences. No. 1686 in CEUR Workshop Proceedings. Aachen; 2016. Urn:nbn:de:0074-1686-8. Available from: http://ceur-ws.org/Vol-1686/WSSSPE4_paper_12.pdf.

[12] 'Launching a New Era for NWChem' https://www.pnnl.gov/science/highlights/highlight.asp?id=4411 [Accessed May 11, 2017]

[13] Elsethagen, T., Stephan, E., Raju, B., Schram, M., MacDuff, M., Kerbyson, D., van Dam, K.K., Singh, A., and Altintas, I.: Data provenance hybridization supporting extreme-scale scientific workflow applications, in Book: Data provenance hybridization supporting extreme-scale scientific workflow applications (IEEE, 2016), pp. 1-10

[14] Stephan, E., Raju, B., Elsethagen, T., Pouchard, L., and Gamboa, C.: 'A Scientific Data Provenance Harvester for Distributed Applications'. IEEE Proc. New York Scientific Data Summit, New York, NY2017

[15] Isaacs, K.E., Giménez, A., Jusufi, I., Gamblin, T., Bhatele, A., Schulz, M., Hamann, B., and Bremer, P.-T.: 'State of the art of performance visualization', EuroVis 2014, 2014