CODAR Design Document

Chimbuko: CODAR Framework for Performance Analysis and Visualization– YEAR 2

Version of January 21, 2018

Kerstin Kleese Van Dam, Shinjae Yoo, Wei Xu,
Line Pouchard, Hubertus Van Dam, Dingwen Tao, Cong Xie, Abid Malik
Brookhaven National Laboratory
Kevin Huck, Sameer Shende
University of Oregon

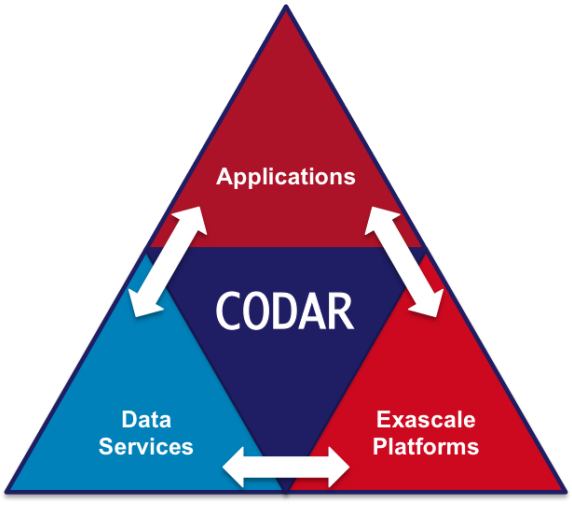

Table of contents

1 Introduction **1**

2 Stakeholders **1**

2.1 ECP Applications 1

2.2 Software Technologies 1

3 Use cases **2**

3.1 NWCHEM 2

3.2 QCD 2

3.3 LAMMPS 3

3.4 Fusion 3

4 Requirements **3**

5 Related work **3**

6 Design **3**

6.1 Introspection (TAU + SOSFlow) 3

6.2 Required Data and the Stream Interface 4

6.3 Data Analysis 5

6.4 Performance Visualization 5

6.5 Prescriptive provenance for online analysis 6

7 Success metrics **6**

8 Work plan **7**

9 Open questions **7**

Chimbuko: CODAR Framework for Performance Analysis and Visualization– YEAR 2

Kerstin Kleese Van Dam, Shinjae Yoo, Wei Xu,
Line Pouchard, Hubertus Van Dam, Dingwen Tao, Cong Xie, Abid Malik
Brookhaven National Laboratory

Kevin Huck, Sameer Shende
University of Oregon

1 Introduction

The Chimbuko framework captures, 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 software or 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 provides this capability in both offline and online (in-situ) modes. Because capturing performance metrics can quickly escalate in volume and provenance can be highly verbose, Chimbuko includes a data reduction module.

Currently, the framework provides performance visualization and data analysis for offline mode. For the second year, the framework will provide performance visualization and data analysis for online mode.

2 Stakeholders

2.1 ECP Applications

Stakeholders include scientific applications that are part of the ECP program. We have consulted many and received positive feedback regarding the usefulness to them of a system such as Chimbuko. ECP applications that exhibit unexplained variability in performance, strong data reduction needs, and different types of workflows are poised to benefit from the Chimbuko technology.

2.2 Software Technologies

Other systems in the CODAR project, including ADIOS, SWIFT, Savannah, and Cheetah, are potential stakeholders. ECP software technology projects such as SZ, ZFP, Cinema, Candle and Spack are all potential contributors of software components that may be integrated into the Chimbuko framework.

3 Use cases

We have following use cases:

3.1 NWCHEM

In the NWChemEx project we will be studying processes involving transmembrane proteins as well as zeolite catalysts. The processes of interest require the calculation of free energies and the dynamics of the molecular structures. In addition, to simulate realistic molecular environments, the molecular structures will have at least 100,000 atoms, different regions may be evaluated at different levels of approximation, and the simulations will work with time steps of about 1 femtosecond. Hence to sample enough of the phase-space a larger number of time steps will be evaluated (think of 1 million time steps). The way the simulations will be run requires the evaluation and forces as well as updating the molecular structures. While these calculations are executed, the statistics needed for the free energies are collected. In addition selected structures along the trajectory will be stored, so that additional properties for those can be calculated. These additional properties will facilitate comparing the calculated trajectories against experimental observations. Hence overall there will be a large number of cores calculating the energies and forces alongside a small number of cores analyzing the results and storing selected time steps for more detailed simulations.

3.1.1 Requirements for the Online Performance Analysis. There will be a large number of parameters that define the total amount of work in particular parts of the simulation, and different amounts of work will change the optimal work distribution. An important aspect will be that the performance characteristics need to be recorded in a way that can be compared against prior simulations to establish the figures of merit of the development. This requires capturing some base characteristics that are always the same. For specific performance optimizations it may be necessary to capture the performance of specific parts of the code, depending for example, on the functionality of interest, or on the characteristics of the data distribution, or on the granularity of the tensor blocks and associated task sizes. Dependent on these kinds of characteristics the data collection may be turned on or off. The shear volume of the data expected requires the analysis to be performed online

3.2 QCD

Last year we experimented with using TAU for a single benchmark calculation (single application) (https://github.com/meifeng/Example-LatticeQCD-With-TAU), which is not representative of the typical production lattice QCD simulations. These are orchestrated in workflows and consist of several more complex components, including propagator calculations and contractions. The performances of these calculations are often limited by the data transfer rates, both intra-node (depending on node architecture) and internode (mpi). IO can also be a limiting factor for some of the algorithms in the LQCD calculations. This year, while the LQCD code is undergoing constant development, we will look into using TAU and associated performance profiling and tracing tools to get a more comprehensive understanding of the performance bottlenecks, to guide our development, and to provide feedback to the tool developers. As the code is evolving to adapt to the pre-exascale architectures such as Summit, we will target to have the first comprehensive study of the various performance metrics related to lattice QCD simulations at the beginning of FY19.

3.3 LAMMPS

LAMMPS (Large Scale Atomic/Molecular Massively Parallel Simulator) is widely used Molecular Dynamics simulation engine that studies materials science and adopts MPI for parallel communication. The use case of LAMMPS is a workflow that is composed of three components, the LAMMPS application, the $Voro++$ was (analytics for LAMMPS) and an ADIOS based parallel data writer component (stage_write). The workflow is configurable, so LAMMPS can communicate with $Voro++$ either directly or through stage_wtite for adopting different IO strategies. The flexibility of the LAMMPS use case can help explore performance tradeoffs on different machines.

3.4 Fusion

4 Requirements

R-1) TAU and SOSFlow frameworks from the University of Oregon will provide the BNL team with infrastructure to make performance data accessible online (i.e., performance monitoring) in a form that permits in situ analysis and reduction.

R-2) NWChem Version, Compilation, Run instruction and test case.

R-3) XGC Version, Compilation, Run instruction, and test case.

R-4) QCD Version, Compilation, Run instruction and test case.

R-5) LAMMPS Version, Compilation, Run instruction and test case.

R-6) The Chimbuko framework should be able to run on MIRA, Titan, Theta, and more systems.

R-7) Information requirements from BNL to the Oregon team. See Section 6.2.

5 Related work

Workflows are taking an increasingly important role in orchestrating complex scientific processes in extreme scale and highly heterogeneous environments. However, to date we cannot reliably predict, understand, and optimize workflow performance. Sources of performance variability and in particular the inter-dependencies of workflow design, execution environment and system architecture are not well understood. While there is a rich portfolio of tools for performance analysis [2, 3, 4, 5] modeling and prediction for single applications in homogeneous computing environments, these are not applicable to workflow, due to the number and heterogeneity of the involved workflow and system components and theirs strong interdependencies. In addition, there is currently no tool that tracks the performance of workflow components as it relates to provenance. A literature review to support these claims has been published in [1].

6 Design

The main design components are 1) Introspection, 2) Visualization, 3) Data Analysis and, 4) Provenance. The main goal is to detect abnormalities during runtime.

6.1 Introspection (TAU + SOSFlow)

The University of Oregon (UO) will provide the CODAR project with infrastructure to make performance data accessible online (i.e., performance monitoring) in a form that permits in situ analysis

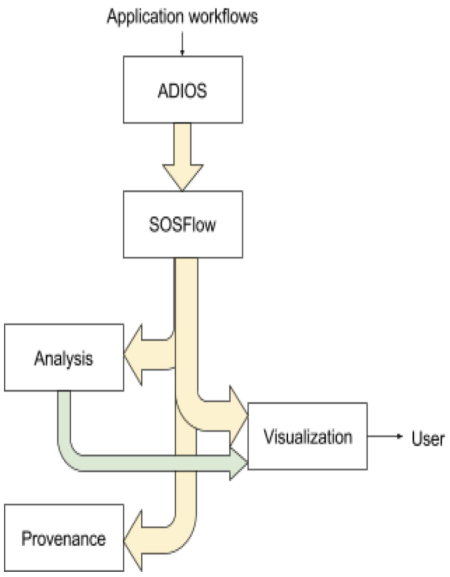

Figure 1: Chimbuko information pipeline.

and reduction. In the first year, a performance monitoring prototype will be created for demonstration of online access using three milestones. Subsequent years (as necessary) will see hardening, testing, tuning, and deployment of tools with relevant CODAR applications. Configurability is a key concern in this effort, allowing for targeted and multi-resolution access to data.

1. Extensible Monitoring Plugin Support: TAU will be extended to provide a plugin framework where an event stream will be exported by each MPI rank. An analysis plugin will subscribe to events such as initialization/finalization, metadata values, interval event timers, and counters. The TAU team will modify the current hardwired SOS integration to be a plugin. The SOS client plugin will monitor the data stream at runtime, and provide aggregation/filtering/reduction utilities. The SOS client plugin will also provide a feedback path to TAU, in order to increase/decrease resolution of measurement (sampling rate, callpath depth, hardware counters, etc) if/when that capability is added to TAU.

2. Online Monitoring Access: Profile, event trace and metadata events will be provided to the analysis through SOS[]. In a coupled application / workflow scenario, SOS will regularly/periodically aggregate TAU data over the SOSflow infrastructure. This data will be extracted from SOS through an SOS analysis extension that will output the aggregated event stream as ADIOS streaming output. The SOS extraction client will have the ability to filter data from one or more specific nodes, rather than just an aggregated data stream. The CODAR team will use the ADIOS stream for supporting their trace analysis tools and anomaly detection. These tools may be shipped with TAU as contributed software.

6.2 Required Data and the Stream Interface

Figure 1 gives the Chimbuko pipeline for information. The figure shows how performance data streams through Chimbuko's modules. For data analysis and visualization module, the following features are to be collected from SOSFlow:

1. For each component in a workflow:

(a) start and end timestamp
(b) call stack
(c) memory allocation
(d) IO in network or disk
(e) total Communication time
(f) effective communication time
(g) effective computation time
(h) idle time
(i) number of synchronization points/ barriers
(j) communication size
(k) communication between functions
(l) communication between nodes

2. For each workflow:
   (a) number of components
   (b) amount of communication between a pair of components
   (c) effective communication between a pair of components
   (d) communication size between a pair of components
   (e) aggregated number of communication calls
   (f) aggregated communication execution time

For visualization module, we also require the outlier results from analysis module in terms of the array of call ID, MPI execution ID, or workflow ID.

6.3 Data Analysis

In this year we aim to provide online performance analysis. Last year, we prototyped off-line performance anomaly detection by using Local Outlier Factor (LOF) algorithm. In order to push our data analysis to online level with strong scalability, we plan to design or evaluate online anomaly detection algorithm with the following minimum features to be collected from SOSFlow given in Section 6.2.

6.3.1 Interface design between online data analysis and SOSFlow. The online data analysis could be implemented as two styles. (1) Consumer-Producer where SOSFlow will be the performance profile/trace stream producer, and our online analysis will be the consumer implemented as its client. (2) Another design choice is that we may consider our performance analysis module as a plug-in within SOSFlow and get performance data stream locally.

6.4 Performance Visualization

Last year, we finished off-line performance visualization focusing on “overview first, details on demand” scheme. For this year, analysis before visualization is mandatory to enable real-time data consumption. We thus aim to provide online performance visualization coupled with performance analysis of the chimbuko team.

Therefore, in order to establish the connection between the analysis module, SOSFlow, and the user, the mission of visualization module has two folds: first, a user exploration interface to visualize, monitor and interact with the analysis results and performance data; second, a back end communication to the analysis module and streaming data flow from SOSFlow.

As deliverables, we will complete the following tasks:

- A set of visual analytics modules at the front end to convey analysis results and provide interaction
- Methods on the back end to communicate with data analysis module and SOSFlow. Two styles are considered: (1) passive mode, where SOSFlow or analysis module performs as server, and sending update data to visualization module as its client; (2) active mode, where user queries the detailed information from SOSFlow or analysis module.
- Methods on the back end to maintain a local database collecting only the details of the outliers and other necessary information for users to query workflow details.

6.5 Prescriptive provenance for online analysis

This year we are expecting to extract and store provenance relevant to online performance analysis. Provenance will be extracted, aggregated, and made available while the simulation is in progress with the TAU/SOSFlow plugin. As provenance can be very verbose and persisting provenance for an entire run is impractical, we plan on persisting detailed provenance ONLY for the anomaly events or events of interest detected by online performance analysis. Ultimately, provenance is to be persisted in SOSFlow for a moving window prior to and posterior to an event with the additional capability to increase or decrease the verbosity of provenance (size and granularity of the moving window) selected for storage around the event. In addition, static information describing the system, runtime configuration, and workflow or application metadata similar to what is already extracted for offline will be persisted at the on start and end of the run (static information). This is what we call prescriptive provenance: provenance selection and use prescribed by the results of performance data analysis described above. Persisted prescriptive provenance is an end-product of the analysis after training sets have been built. Specifications of the moving window of provenance prior to an outlier event will need to be studied for tradeoffs. For instance, we will study if this window is defined in terms of execution time or number of timesteps. Constraints of size will apply, as well as tradeoffs in the amount of details needed, and will need to be balanced with usefulness to a scientific code developer. The metadata values that need to be extracted for prescriptive provenance will vary depending on the application and the node architecture where application components run. The following values are of interest, with a priority first given to detailed communication, understood as mpi communication, then I/O. The choice of communication as a priority for data extraction is motivated by its demonstrated impact on performance in 2017 for applications such NWChem and LAMMPS.

7 Success metrics

The purpose of Chimbuko online mode is to give a runtime view of different features important to complex scientific simulation and workflows. This helps an end user to manipulate different runtime configuration parameters during the execution phase to improve performance of a given workflow application particularly on a new Exascale machines. The success metrics will be the number of applications that can assess and visualize the number of performance metrics through Chimbuko's

online mode for performance improvement. Also, a user's ability to explore the performance by interacting with the visualization interface. Details include workflow overview showing the identified outliers, function call details in a call tree structure, in a timeline format, and other additional representations.

8 Work plan

Date	Milestone
6/31/2017	Demo for NWCHEM and Fusion showing the initial capabilities of online analysis.
12/31/2017	Distribution version of Chimbuko with enhanced capabilities for online performance visualization and analysis for scientific simulation and workflow.

9 Open questions

Some research questions that are relevant to this year's work:

- Level of granularity of collected data: This is an important research question for performance analysis tools. Size of data depends on the level of granularity. And this level depends on the needs of each application, for some applications communication contributes significantly to latency, while for others, I/O is the bottleneck.
- Scalability of introspection: How to scale the information collection without putting too much over head? Collecting a complete holistic view of the environment during runtime is important. However, if the overhead is too much , then the net gain at the end would be too small which can kill the idea of online analysis.
- Maintainability of data: How long should we maintain the collected data and when?
- Potential publications: We are aiming to publish research papers in refereed conferences/-workshops based on this year work.

References

[1] *Enabling Structured Exploration of Workflow Performance Variability in Extreme-Scale Environments*: Kleese van Dam, K., Stephan, E.G., Raju, B., Altintas, I., Elsethagen, T.O., and Krishnamoorthy, S., Proc. 8^{th} Workshop in Many-Task Computing on Clouds, Grids, and Supercomputers (MTAGS) collocated with SC 2015.

[2] *Lammel S., Zahn F., Frning H. (2016) SONAR: Automated Communication Characterization for HPC Applications. In: Taufer M., Mohr B., Kunkel J. (eds) High Performance Computing. ISC High Performance 2016.*

[3] *S. Shende, A. D. Malony, J. Cuny, K. Lindlan, P. Beckman and S. Karmesin, Portable Profiling and Tracing for Parallel Scientific Applications using C++, Appears in: Proceedings of SPDT'98: ACM SIGMETRICS Symposium on Parallel and Distributed Tools, pp. 134-145, Aug. 1998.*

[4] *Knpfer A. et al. (2012) Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir. In: Brunst H., Mller M., Nagel W., Resch M. (eds) Tools for High Performance Computing 2011.*

[5] *L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent. HPCToolkit: Performance tools for scientific computing. In SC '08: Proc. of the 2008 ACM/IEEE Conference on Supercomputing, New York, NY, USA, 2008. ACM.*