CODAR Design Document

CHIMBUKO: CODAR Framework for Analysis and Visualization of Provenance Performance

Version of May 24, 2017

Line Pouchard, Abid Malik, Wei Xu, Shinjae Yoo,
Hubertus Van Dam, Meifeng Lin, Shantenu Jha, Kerstin Kleese Van Dam

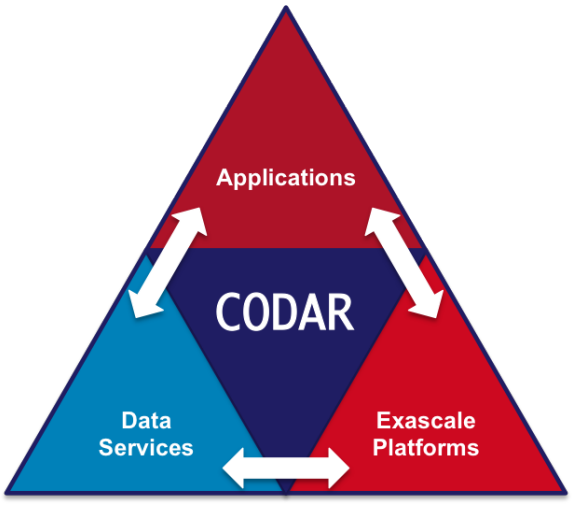

Table of contents

CHIMBUKO: CODAR Framework for Analysis and Visualization of Provenance Performance

Line Pouchard, Abid Malik, Wei Xu, Shinjae Yoo,
Hubertus Van Dam, Meifeng Lin, Shantenu Jha, Kerstin Kleese Van Dam

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. The framework is intended to be used first in offline mode so that a user can determine what metrics are of interest to their case, and then in online mode.

2 Stakeholders

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 Chimbuko technology.

Other systems in the CODAR project, including ADIOS, SWIFT, Savannah, and Cheetah, are also potential stakeholders. We address these systems in the Open Questions section (Section 9) at the end of this document.

3 Use cases

We have the following use cases for the Chimbuko framework.

3.1 NWChemEX

NWChemEX is a scientific code for simulating the dynamics of large scale molecular structures and materials systems on large atomistic complexes. NWChemEX provides a use case for the provenance and visualization modules in Chimbuko and for the data reduction module. NWChemEX uses both the OpenMP and MPI parallel programming models. However, for our use case study, NWChemEx is using MPI only.

3.1.1 Provenance and visualization use case. Provenance and performance analysis is needed to support new code development and testing effort in this phase of NWChemEX. Table 1 shows

the list of metrics selected for NWChemEx performance analysis in both categories.

Performance metrics	Provenance
Execution time per code region	Application time
Call trees	Workflow component
Communication: MPI Calls	Workflow instance
Communication: Interconnect performance	Workflow step
I/O: number of read/writes	Wall time allocated, Version control

Table 1: Performance and provenance needed for NWChemEX.

The performance metrics are useful for testing new development when they are related to provenance. Provenance allows comparison between different runs, different code versions, etc. So in addition to the list of metrics in Table 1, relationships between these metrics are also needed. These relationships are encapsulated in timestamps and identifiers, and described in the data model specified in Section 6.

3.1.2 Data reduction use case in NWChemEX. Data reduction is also needed in NWChemEX. The science requirements demand running dynamics simulations on large atomistic complexes involving millions of atoms for long time scales up to micro seconds at a time resolution of **femtoseconds**. A single resulting trajectory would generate 32 Peta-Byte of data which is too large to be stored for offline analysis. However, key information contained within the data are statistics that can be related to free energies of processes and critical events and points. Examples of key events are transition states and equilibrium structures where reactions happen and where stable molecules are found, respectively. Hence in principle significant data reduction factors should be achievable without losing the scientifically important data. The challenge is to filter these key events.

Current data reduction techniques within the dynamics calculations rely on a simple decimation strategy which might blindly discard possible meaningful structures in the data. Typically, only 1 out of every 100 to every 1000 time steps of the structure is stored and all other data discarded. In addition, scientists run shorter simulations representing shorter simulation timescales. Such crude approaches risk losing relevant results and missing important phenomena.

3.2 Lattice QCD

Quantum Chromodynamics (QCD) is the theory that describes the strong interactions between quarks and gluons that make up elementary particles such as protons and neutrons. The first-principles numerical simulations of QCD are performed through the framework of Lattice QCD (LQCD), in which quarks and gluons are simulated in a discrete four-dimensional or five-dimensional spacetime. A typical LQCD workflow involves the following steps:

1. Generation of gluon field ensembles, which is done through Monte Carlo simulations with Molecular Dynamics updates (Hybrid Monte Carlo). These gluon field configurations are written to disk for the analysis in Steps $2-4$. This step is very expensive and will benefit greatly from good strong scaling as we need to run one single (or just a couple) long Monte Carlo streams.

2. Computation of quark propagators on the gluon field configurations, which involves sparse-matrix inversions using Krylov solvers and is another compute-intensive step. The quark

propagator computation on each gluon field configuration is independent, and can be done in parallel. This step involves reading in the gluon field and sometimes writing out the quark propagators, which are huge files. Sometimes eigenvectors are needed to accelerate the propagator calculations. This would require generating $O(1000)$ very large eigenvectors, saving them to disk, and reading them back to construct a reduced Krylov space.

3. Contraction of the quark propagators to construct particle correlators. This is relatively cheap, and involves I/O, mostly BLAS routines and global reduction at the end.

4. Extraction of physical results from the correlators. This is our final step to obtain LQCD physical results and can be usually done on a PC.

Steps $1-3$ can be bundled into one single binary/application, but more often they are separated into different binaries/applications. The programming models used in current LQCD software stack are MPI+OpenMP on CPUs and MPI+CUDA on GPUs. There are several different software libraries used in LQCD simulations by different groups, see `http://usqcd-software.github.io/`. The main software packages the BNL LQCD group currently uses is Columbia Physics System (CPS) and the newly developed Grid library (`https://github.com/paboyle/Grid`). The exascale LQCD software is still at the design stage, but the programming models will remain the combination of MPI, OpenMP/OpenACC and CUDA. For the purpose of the performance metrics and provenance case study, we will use the existing LQCD software.

3.2.1 Performance metrics for QCD. QCD can benefit from the following performance metrics for software development and optimization, and can also help determine optimal machine partition sizes and runtime parameters:

- Call tree
- FLOPS per process, FLOPS within a function if possible
- Load balance
- Memory usage, grouped by function calls
- L1, L2 cache data and instruction accesses, L2 cache misses
- Communication Time, including intra-node communication for hybrid MPI/OpenMP jobs
- IO Read (size and time)
- IO Write (size and time)
- Interconnect performance metrics

3.2.2 Provenance information for QCD. The following provenance information is useful for QCD software development:

- Computer system performance characteristics
- Interconnect performance characteristics
- Storage system performance characteristics

3.3 MD Workflows

Molecular Dynamics (MD) simulations are an important means to understand physical properties of molecules. MD simulations provide atomistic level resolution with high-temporal resolution, however they can be expensive. An important problem that arises from the ability to run multiple MD simulations is how to gauge their effectiveness as simulations are in-flight: should a user wait till the planned simulations are complete then analyze the output of those simulations? Or is it possible to ”steer” the simulations as they are in progress? There are several scenarios where steering the simulation while in progress is desirable. These include but are not limited to advanced sampling, rare event exploration etc. The driver is not performance per simulation , but the need to steer simulations at scale to be more effective by the in-situ — online reduction and analysis of MD runs. The MD Workflows strongly believe that the Chimbuko framework can help find performance bottlenecks unknown to the team before.

4 Requirements

- Chimbuko will accept performance and provenance data that adheres to the defined input format. Examples of this data format are given in Section 6.
- Chimbuko will define a suitable data format for the visualization and analysis.
- Chimbuko will be used in offline mode, and enable visualization and data analysis of performance and provenance data pertaining to a single workflow execution at a time including time series of metrics.
- The volume of the overall performance data for a single workflow execution could exceed several tera-byte, and Chimbuko must be able to visualize and analyze this volume of data. Chimbuko must enable visualizing and analyzing online performance data streams at the same time, so that several workflow results could be uploaded in parallel.

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 challenges for workflow performance analysis are:

- Limited visibility: Data intensive applications regularly rely on shared resources such as I/O systems, system bandwidth, file systems, archives, local or wide area networks, components

that can strongly influence the performance of the application, however the analysis of those components is usually either not covered by existing tools or the metrics captured are very limited.

- Data size: Capturing all possible performance metrics becomes prohibitive in terms of data volumes with growing system sizes.
- Tools for workflow: There are no common tools available today that investigate workflow performance in the same way that we can examine single application performance. The challenge is that workflows combine several applications, each with their own performance characteristics that can show strong interdependencies in their execution behavior.
- Adaptability: New system and software features that can change behavior at runtime and lead to variable behavior e.g. workflow management systems that can adapt execution pathways at runtime based on system performance, CPU and GPU throttling, Auto-boost and accompanying throttling.
- Performance data streaming: Today's performance analysis tools are created as one time snapshots, rather than as longer term studies that can correlate changes in application performance to specific changes in the software, workflow, system or system software, thus making it difficult in larger development efforts to assess the impact of specific optimization strategies.
- Quantification of information: There are no formal systems available that track qualitative metrics about applications or workflows over time, e.g. level of accuracy achieved, level of data reduction etc.
- Relationships between performance metrics and job information: Today's performance analysis tools do not record all that is needed to correlate changes and persist this information for forensic analysis and comparison of code versions and releases over time.

Chimbuko Framework is aimed at providing solutions for these challenges.

6.1 Basic design layout

Figure 1 and 2 show the Chimbuko framework's layout for off-line and on-line performance analysis respectively. Table 2 explains the information flow among different components. For the off-line mode, information is collected during a run of an application and analyzed after the execution is over. For the on-line mode, information is collected and analyzed during an execution phase. The Chimbuko framework will interact with the TAU infrastructure [3] for performance extraction for visualization and data analysis.

6.2 Example: Integrating Cheetah, Savannah, and Chimbuko

Consider a CODAR analyst seeks to quantify co-design tradeoffs involved in online data analysis and reduction for a particular application. To this end, the analyst runs an ensemble of executions, each involving an application X plus an analysis A and a reduction R, and each with different specifications of what information needs to be saved when (e.g., different data reduction mechanisms and parameters), and what work is to be placed where (e.g., different numbers of nodes allocated to X, A, and R; X, A, and R allocated to the same or different nodes; different mechanisms used to transfer data between components). The analyst uses Cheetah to run the ensemble. Cheetah specifies to the Chimbuko performance framework that for each run on Savannah, a set of

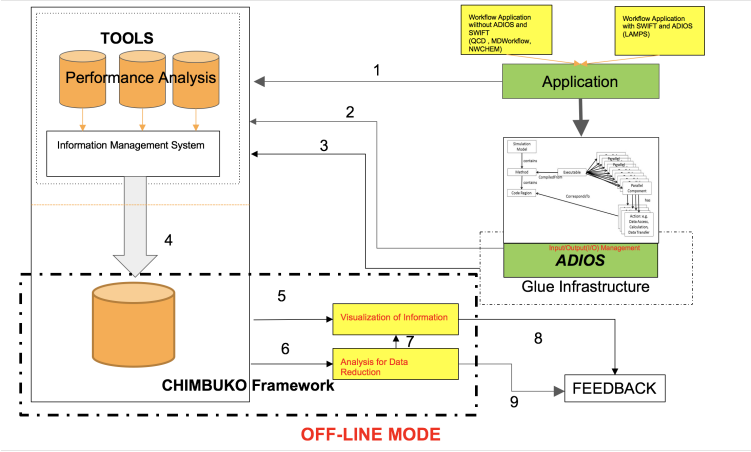

Figure 1: Information flow diagram for off-line performance analysis.

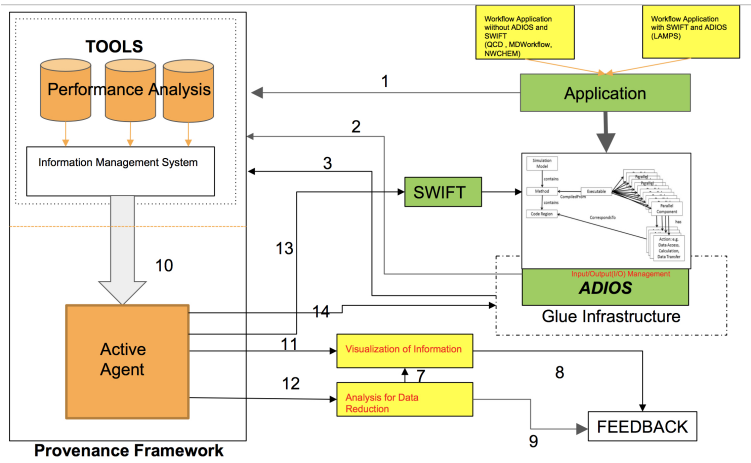

Figure 2: Information flow diagram for on-line performance analysis.

Arrow	Explanation
1	Information collection from an application using a tool
2	Information collection from ADIOS using a tool
3	Information collection from GLUE (Savannah) Framework using a tool
4	Collected information is integrated by ProvEn Framework
5	Information extraction from ProvEn for visualization
6	Information extraction for Data Analysis
7	Information extraction from Data Analysis for visualization
8	Information for feedback
9	Information for feedback
10	On-line information extraction through an active agent
11	On-line information extraction for visualization
12	On-line information extraction for Data Analysis/Reduction
13	On-line information extraction by SWIFT for better performance
14	On-line information extraction by ADIOS for better performance

Table 2: Information Flow in Figure 1 and 2.

performance data (PF) need to-be collected: e.g., communication volumes and times, compute time and idle time in different routines and on different nodes, etc. The information needed to enable understanding of the co-design space, and perhaps to define further co-design experiments (see Appendix at the end for the complete list). The information is collected through the TAU performance tool. Figure 3 explains graphically the whole scenario and how the three frameworks integrate and talk to each other.

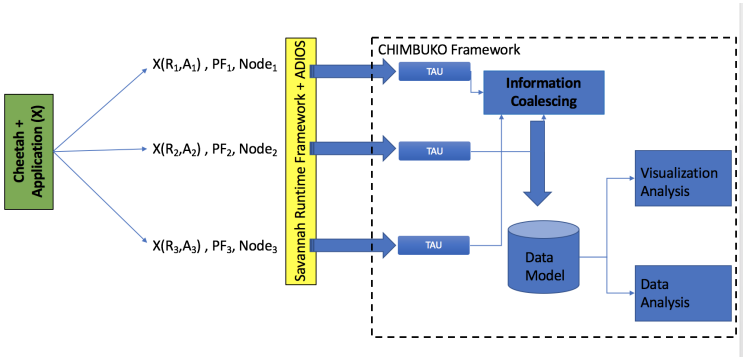

Figure 3: Integration of Cheetah, Savannah, and Chimbuko.

6.3 Assumptions

These assumptions include dependencies on development by the TAU team to produce the data in the format and with the content that we need. They have been discussed and agreed upon with the TAU team during their visit on April $24 - 25, 2017$. The list below does not necessarily reflect an order of priorities.

1. All required data will be collected by TAU and provided via text files in JSON format, or as reasonable python object for each profile or trace file
2. The required data includes performance metrics and application specific information related to provenance. Chimbuko will be provided with an Instrument script for job submission, and acquisition of workflow information into the data model.
3. Performance evaluation for two kinds of workflows (serial and parallel) and their hybrid in both (1) profile and (2) trace will be provided. The accepted files are (1) a connected profile file in accepted format, and (2) a connected trace file in accepted format.
4. The new profile file or trace file must have sufficient workflow information so users can see (1) if they are from the same workflow, (2) which part of the profile (or trace) is describing which stage of the workflow, (3) data file read - write I/O for communication between stages of a workflow.

Assumptions for the online analysis module to be delivered in December 2017 are addressed in the Open Questions section (Section 9) at the end of this document. We don't anticipate for them to be substantially different than for offline.

6.4 Data model

This data model is inspired from the OPM- WFPP [1]. While OPM-WFPP provides a description of the possible basic variables and provenance information, further details need to be provided.

For instance, the case where these data are distributed among many files, the logic relating these data needs to be defined. Table 3 gives the items in the Chimbuko data model.

Workflows	Locations (HW)	Location(SW)	Metrics	Trace Event ID	Time stamp
Execution instance Component Step Sub-task	Core Node Interconnect	Code region Library Call tree Function	Execution time MPI/OpenMP Messages Message size Time spent Wait time I/O Memory Usage	Enter Leave	Time stamp1 Time stamp2

Table 3: Example of items in the Chimbuko data model.

Application attributes such as job information are part of the data model and described in Table 4.

Application ID	Application Name	Source	Date Compiled	Version Control Number

Table 4: Application attributes.

Table 5 lists the items needed to encapsulate the relationships between files, metrics, and application attributes. These include identifiers and timestamps.

Time stamps	Relationships between execution time and software process (code region, function, call, etc.) in the execution of a codes
HW Locations	Relationships between execution nodes (cores) and software process being executed.
SW Locations	Relationships between software processes and processor ranks as well as code regions executed
Identifiers	Workflow and applications identifiers that capture the logic of the workflows

Table 5: Application and workflow identifiers.

Identifiers that perform the following tasks are part of the data model:

- Identifiers that distinguish one workflow execution from another.
- Identifiers that distinguish one workflow component from another within the same workflow execution (in particular if there are several instances of the same component running in parallel).
- Identifiers that distinguish the location of one measurement from another within the same workflow component (metrics captures within one workflow code execution running across several cores or nodes of a system in the case of parallel workflows).
- It has to be possible to relate these identifiers in time and space - e.g. all metrics identifiers captured for one component within a workflow need to link to that component, and it needs to be possible to express relationships between components to capture when they were run i.e. in parallel or one as a result of one or more previous runs. etc.
- One might also need further identifiers for specific systems - i.e. Titan on which a workflow is executed.

Examples of the data definitions for the model are given in Table 6, 7, and 8.

id	name	location-id	version	event-type	timestamp

Table 6: Workflow instance.

id	name	location-id	application-id	event-type	timestamp

Table 7: Workflow component.

id	name	location-id	measurement	units	datetime

Table 8: Metrics

6.5 Terminologies

6.5.1 Workflow system. A specialized form of data management system for composing and executing a series of computational or data manipulation steps in a scientific application.

6.5.2 Application. An executable program with a specific science purpose and a designated input and output format. For performance analysis, we divide scientific applications in two classes:

1. Workflow applications without SWIFT and ADIOS support. NWCHEM, QCD, and MDWorkflow
2. Workflow applications with SWIFT and ADIOS support. LAMMPS

6.5.3 Granularity of Information. The framework will collect the information both at high and low levels. By high level, we mean workflow level. By low or fine level, we mean a specific phase within a workflow.

6.5.4 Producer. Any component that can produce useful information for performance analysis. For example, ADIOS, SWIFT, scientific applications etc.

6.5.5 Consumer. Any component that can consume the information for performance analysis and performance enhancement. For example, TAU, Score-P, and ProvEn etc.

6.5.6 Active Agent . A component that can manage the flow of information among different components for on-line analysis.

6.5.7 Provenance. A record of the activities, activity occurrences, agents, and timeline involved in producing, influencing or delivering a piece of data. This record is persisted over time to enable inter-comparisons.

7 Metrics

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. The developers should be enabled to significantly improve the performance of the application in large computing environments, particularly on new Exascale machines. The metrics will be the number of applications we can run with Chimbuko and how successful we are at helping improve each application.

8 Work plan

8.1 Visualization part of Chimbuko

1. New visualization methods that are missing in existing visualization tools for workflow-level performance measurement to meet the current requirement.
2. Provide corresponding tools to support representative performance metrics.
3. More advanced visualization and interaction approaches for linking trace data, profile data, and metadata of workflows.
4. Design of online performance analysis and visualization requirement with TAU and ADIOS team (version 1)

The following tasks will be done after June:

1. Further development of online API interface
2. Accommodate the developed offline visualization methods for workflow data into the scenario for online data acquisition.
3. We will devise the visualization to reveal the changes and evaluation for the online data reduction or analysis.

8.2 Data analysis part of Chimbuko

1. We will explore the opportunities of change detection and lossy compression of NWChem generated MD trajectories.
2. Given the problematic and successful application running profiles and trajectories, pilot study report of automated anomaly detection.
3. Unfiltered performance profile and trajectories dataset for data reduction study

The following analysis of Chimbuko will be done after June:

1. Further development of online streaming data API interface.
2. Preparations for causal inference simulation data.
3. Pilot study of performance profile and trajectories reduction.
4. Pilot study of potential causal factors of performance drops and pinpoint potential causes of symptom.
5. Batch version of performance data analysis version 1 (by Dec., 2017) to detect performance anomalies.

8.3 Monthly milestones

Month	Milestone
May, 2017	Architecture design and implementation for performance capture framework and analysis tools for offline performance analysis. Use case studies-NWCHEM.
June, 2017	Create 1^{st} release for offline analysis-including testing, packaging, documentation, and release. Plan for online analysis development, deliver initial work plan. Use case studies– Write up interim reports, continue studies.
July, August, September, 2017	Architecture design and implementation of initial online performance capture and and analysis framework, dataflow and analysis capabilities including visualization for debugging. Respond to user comments and bug reports, specially from the use case studies. Use case studies- NWChem, QCD, MDFlow. Preparations for causal inference simulation data.
October, November, 2017	Test and refine initial online performance analysis framework, working closely with use cases. Use case studies-QCD, MDFlow. Pilot study of performance profile and trajectories reduction. Pilot study of potential causal factors of performance drops and pinpoint potential causes of symptom.
December, 2017	Batch version of performance data analysis version 1 (by Dec., 2017) to detect performance anomalies.

8.4 Work plan for deliverables

Date	Milestone
4/30/2017	Final design document for CHIMBUKO v1.0
6/30/2017	CHIMBUKPv0.5 (Offline mode)
12/31/2017	CHIMBUKO v1.0 (Offline mode + Data analysis)

9 Open questions

Here are some open questions for further discussion:

- How Swift can benefit from the Chimbuko framework?
- How ADIOS can benefit from the Chimbuko framework?
- How Savannah CODAR Runtime infrastructure can benefit from the Chimbuko framework?
- Development of streaming visualization modules.
- Conversion of offline to online modules– discussion with Savannah and TAU teams is needed.
- Requirement for online analysis after June 30^{th}– discussion with Savannah, ADIOS, and TAU teams is needed.
- Discussion on the data format and API to support direct data acquirement from memory– co-ordination with ADIOS and TAU is needed.

Appendix

1. Workflow Script Performance Determining Characteristics - Provenance
    - Number of tasks
    - Number of edges
    - Number of joints
    - Number of forks
    - Number of decision points
    - Number of alternative pathways
2. Computer System Performance Characteristics - Provenance
    - Number of Cores
    - Accelerators
    - CPU clock frequency
    - Memory clock frequency
    - Memory hierarchy
    - Interconnect
    - System Layout
3. Wide Area Network Performance Characteristics - Provenance
    - Type
    - Line speed
    - Topology
    - Usage modes
4. Interconnect Performance Characteristics - Provenance
    - Type
    - Line speed
    - Topology
5. Storage System Performance Characteristics- Provenance
    - Type
    - Read Speed
    - Write Speed
    - Storage Volume
    - Cache
    - Access management system
6. Workflow Script Instance Performance Metrics
    - Start Time

- End Time
- Elapsed Time
- Application Queuing Time
- Number Of Joins
- Waiting Time In Joins
- Number Of Forks
- Waiting Time in Forks
- Number Of Calls (to applications)
- Number Of Calls (to functions)
- Number Of Failed Calls (to other programs / functions)
- I/O Read (number and size)
- I/O Write (number and size)
- Peak memory usage
- Memory Usage Per Node and otherTBD
- Paging Volumes Per TBD
- Page Fault Numbers Per TBD
- Virtual Memory Usage
- Total Communication Time
- Number Of Messages
- Volume Of Messages
- Total Data Transfer Size
- Input Transfer Rate per TBD
- Output Transfer Rate per TBD
- Energy Usage
  - CPU
    - per Node
    - perTask

7. Total cycles

- Total instructions
- L1 data cache access
- L1 instruction cache access
- L2 cache data access
- L2 cache instruction access
- L2 cache data misses
- L2 cache instruction misses
- Branch instructions
- Branch mispredictions

- FLOPS

8. Interconnect Performance Metrics
    - Latency
    - Load / Throughput
    - Congestion
9. Storage System Performance Metrics
    - Queue Depth (number of requests, length of requests held in queue)
    - Average I/O size in KB
    - IOPS (read, writes, random, sequential, overall)
    - Throughput in MB/s
    - Percentage write vs. read
    - Capacity (free, used and reserve)
10. Wide Area Network Performance Metrics
    - One Way Latency
    - Packet Loss
    - Packet Duplication
    - Packet jitter
    - Throughput
11. Computer System Performance Metrics (as collectable by e.g. SYSSTAT):
    - Input / Output and transfer rate statistics, global, per device, per partition, per network filesystem and per Linux task / PID
    - CPU statistics (global, per CPU and per Linux task / PID), including support for virtualization architectures
    - Memory, hugepages and swap space utilization statistics
    - Virtual memory, paging and fault statistics
    - Per-task (per-PID) memory and page fault statistics
    - Global CPU and page fault statistics for tasks and all their children
    - Process creation activity
    - Interrupt statistics (global, per CPU and per interrupt
12. Extensive network statistics: network interface activity (number of packets and kB received and transmitted per second, etc.) including failures from network devices; network traffic statistics for IP, TCP, ICMP and UDP protocols based on SNMPv2 standards; support for IPv6-related protocols.
    - NFS server and client activity
    - Socket statistics
    - Run queue and system load statistics

- Kernel internal tables utilization statistics
- System and per Linux task switching activity
- Swapping statistics
- TTY device activity
- Power management statistics (instantaneous and average CPU clock frequency, fans speed, devices temperature, voltage inputs, USB devices plugged into the system)

13. Filesystems utilization (inodes and blocks)

Further metrics might be introduced to capture system utilization and costs

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.*