CODAR Design Document

Release Plan of ECP-CODAR Chimbuko

Version of January 31, 2019

Kerstin Kleese Van Dam, Shinjae Yoo, Wei Xu, Line Pouchard,
Wonyong Jeong, and Hubertus Van Dam
Brookhaven National Laboratory
Kevin Huck
University of Oregon

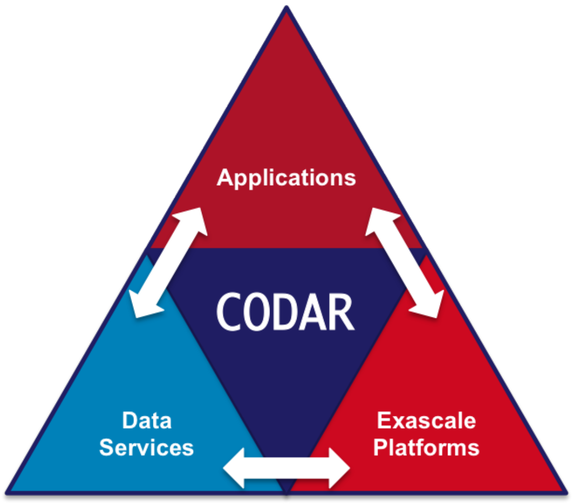

Table of contents

Release Plan of ECP-CODAR Chimbuko

Kerstin Kleese Van Dam, Shinjae Yoo, Wei Xu, Line Pouchard,
Wonyong Jeong, and Hubertus Van Dam
Brookhaven National Laboratory
Kevin Huck
University of Oregon

1 Project Targets

The Chimbuko framework captures, analyzes, and visualizes performance metrics for complex scientific workflows that enables investigations into co-design trade-offs for online data analysis and reduction on extreme-scale machines.

Chimbuko enables co-design studies by allowing scientific applications and workflows to profile their execution patterns on traditional and heterogeneous architectures. The detailed metrics and information (a.k.a. provenance of the execution) are extracted and then reduced and visualized by the Chimbuko data analysis and visualization modules, thus providing insights into the behavior of the codes and workflows at scale. Using provenance, Chimbuko provides these insights for both single applications and the complex orchestrated workflows that are becoming more prevalent to organize complex code execution.

2 Chimbuko v3.0

2.1 Design Target

This year, we will deliver scalable, stable and production-able version of the Chimbuko framework. The main components of the framework **1) Anomaly Detection 2) Visualization 3) Provenance 4) SOSflow** will be integrated through a co-design approach. We will release a prototype version with the following target scalability:

- **Nodes:** 50
- **MPI Ranks:** 500
- **Version: Chimbuko 2.5**
- **Expected Release: June 2019**

Then, we will release stable and more scalable target:

- **Nodes:** 1000
- **MPI Ranks:** 10,000
- **Version: Chimbuko 3.0**
- **Expected Release: December 2019**

2.2 Current Chimbuko v2.0 Status

The current release, Chimbuko version 2.0, provides both online and offline performance visualization and data analysis, compared to offline performance analysis support in Chimbuko v1.0. Figure 1 shows data streaming/workflow execution in Chimbuko v2.0. In the workflow, All the performance trace data is aggregated by SOSFlow and then forwarded into online anomaly detection (AD) and online visualization through ADIOS. When an anomaly is detected, we dump performance traces into a provenance database. The main components of Chimbuko v2.0 are:

- **EventParser**: processes a given aggregated TAU trace (both streaming and batch through Adios). The parser extracts performance metrics and provenance of the execution. It also keeps track of event information such as function, call stack, and function execution times.
- **OutlierDetector**: finds outliers in performance of functions. Due to the scalability, we currently focus on streaming standard deviation as a detector.
- **Visualizer**: enables the real-time visualization for streaming anomaly detection. It provided three major components in the user interface: workflow execution overview in a scatter plot, execution details in a call stack structural visualization, and execution details in a timeline and message communication visualization. On the back end, we supported streaming data processing, reduction and aggregation. As the performance, we tested two different data scale settings for NWChem. The results are summarized in Table 1.

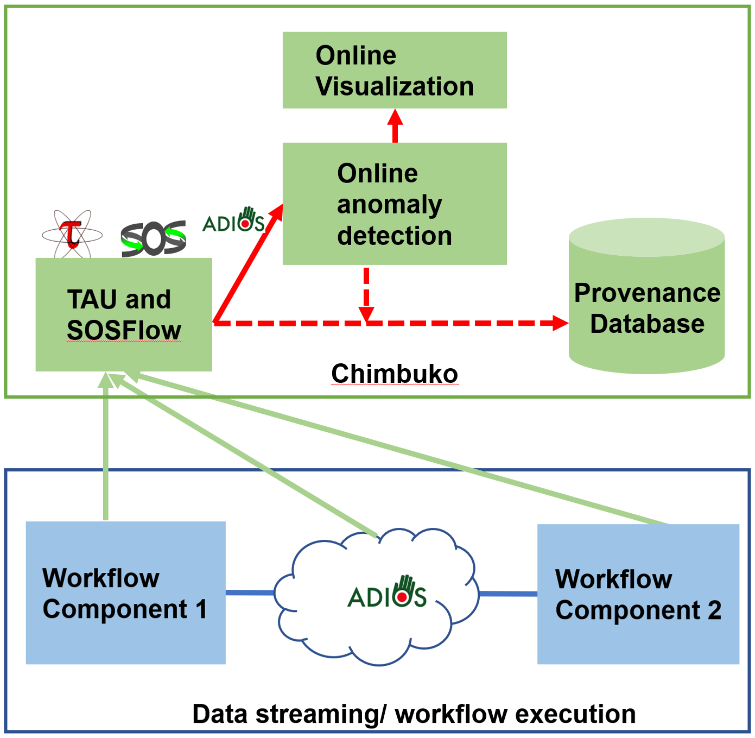

Figure 1: Chimbuko v2.0 (current) Architecture Diagram.

Limitations of Chimbuko v2: Although Chimbuko v2.0 is our first online streaming implementation, the data processing architecture is not scalable. The primary performance bottleneck is coming from SOSFlow aggregation. From Table 1, as we increase MPI Ranks, the number of trace

NWChem Setting	MPI Ranks	Number of trace events	Number of trace events per second	Number of anomalies	Throughput (MBps)
small	2	$393,542$	58.5k	350	5.2
small	4	$1,201,533$	128.1k	445	11.5
small	8	$4,024,651$	224.2k	2235	20.5
large	2	$784,122$	52.9k	818	4.7
large	4	$2,386,634$	101.0k	1121	9.1
large	8	$7,972,872$	172.9k	3683	15.8

Table 1: Summary of experiment datasets and the throughput.

events are increasing proportionally. If we target 10,000 MPI ranks for NWChemEx simmulation with small toy setting, $O(10^9)$ events will be generated. Note that NWChem small configuration (a single ethanol molecule in a solution represented by a few hundred water molecules, totalling less than a thousand atoms) was executed in around only 6 seconds. For what we are targetting NWChemEx simulation case is a molecular dynamics simulation with a transmembrane protein in a lipid cell membrane model with an aqueous environment (about 1 million atoms) and we expect $O(10^{15})$ scale event. Furthermore, there will be other analysis programs or concurrent execution modules. Therefore, we have to design new architecture to avoid this central bottleneck.

2.3 Chimbuko v3.0 Plan Overview

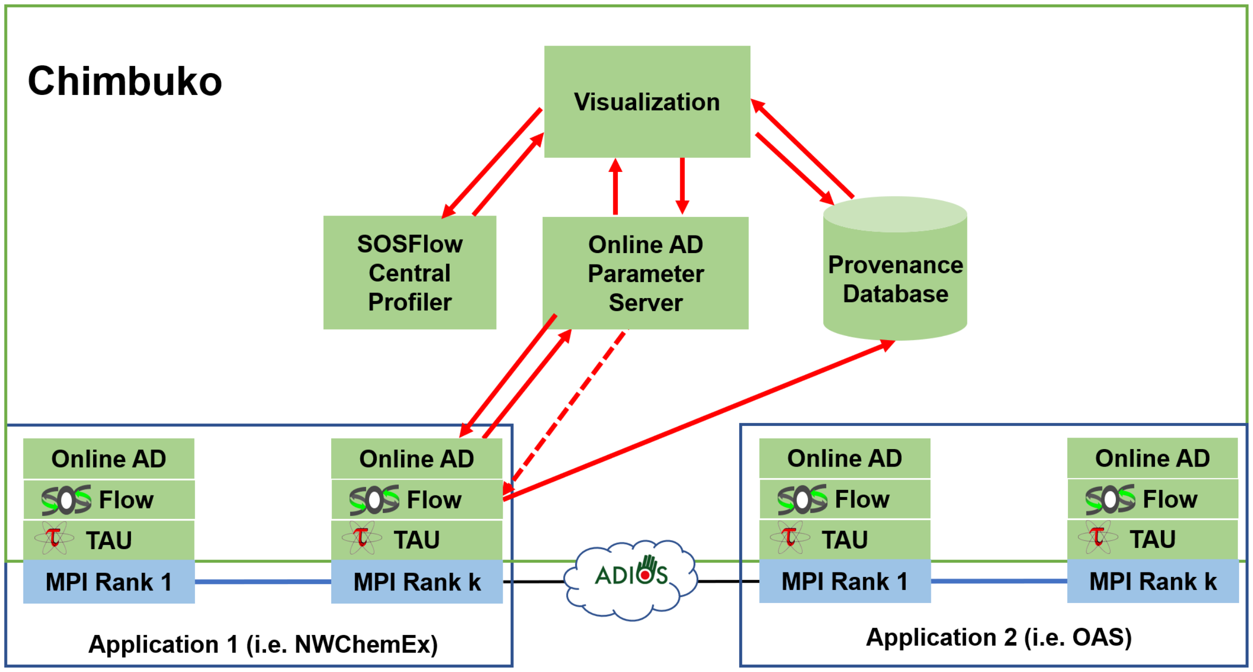

Figure 2: Chimbuko v3.0 Architecture Diagram.

The proposed main idea for efficient and effective performance analysis is **distributed edge processing** of performance trace data, which are critical for the development of upcoming Exascale systems performance study. Figure 2 shows our proposed v3 architecture diagram. Compared to v2 (Figure 1), high velocity trace stream will no long be aggregated. Therefore, SOSFlow is not a central bottleneck anymore. However, we may continue collect all the performance profile statistics from SOSFlow as the amounts of profile data to be collected from SOSFlow will be significantly smaller than trace data. Therefore, in version 3, online anomaly detection is now split into two module online AD parameter server and on-node online anomaly detection.

Online vs Offline: The Chimbuko framework can be run only for some performance/co-design studies. Anomaly detection and visualization parts can be run off-line and on-line. Provenance is extracted and saved in online and offline modes, provenance querying is run off-line.

2.4 Scalable Design for SOSFlow + TAU

The original design for trace analysis involved aggregation of trace data across nodes using SOSFlow. In order to provide a scalable runtime trace analysis solution, the trace analysis needs to be pushed to the compute nodes. Chimbuko 2.5 will only aggregate a filtered (reduced) amount of trace events on a per-node basis, and the analysis will be distributed, rather than centralized. All types of performance data can be analyzed with this approach, whether they are application events, MPI communication events, or GPU kernels. The trace extraction from SOSFlow to ADIOS will be rewritten from Python to C/C++, and upgraded from ADIOS 1.13 to ADIOS2.

While only a subset of the trace events are exported to the anomaly detection, each TAU monitoring instance (within each application process) will maintain a window (i.e. a circular buffer) of trace events representing the short-term memory (or provenance - see Section 2.7) of the application. The trace anomaly detection (see Section 2.5) will be distributed across compute nodes, and when anomalies are detected, the on-node SOSFlow listener daemon will be notified. If the anomaly is local (either one MPI rank or on one compute node), only the local application ranks on that node need be triggered by SOSFlow to dump the event provenance leading up to the anomaly, for offline analysis. If the anomaly is global SOSFlow will communicate to all listeners to dump their trace provenance. The trace data will either be dumped to ADIOS using the same layout as the anomaly detection stream, or to another appropriate file format.

Task ID	Task Name	Priority	Releases	Notes
TAU1	Rewrite TAU/SOS data extraction in C/C++ with ADIOS2	High	v2.5	Currently Python, ADIOS1
TAU2	Implement local provenance trace in TAU	High	v2.5	
TAU3	Integrate SOS feedback to trigger provenance trace output	High	v2.5	

Table 2: TAU/SOS Task List with associated priority and release dates.

2.5 Distributed Anomaly Detection

Since we are processing trace data where the trace data is generated, the online anomaly detection algorithm has to be distributed. For this, we split online anomaly detection module into **Online AD Parameter Server**, and **edge processing engine**. The edge processing engine is decomposed into 1) **Edge TAU Stream Parser** and 2) **Edge AD Module**.

- **Edge TAU Stream Parser:** This module ingests each MPI rank's TAU performance trace data and generated features for anomaly detection and visualization. For instance, the execution time for a function call can be collected. For our version 2.5, we will implement the simplest features first and then increase the features later for version 3.0. The communication between Edge SOSFlow and Edge TAU Stream parser will be investigated and consider ADIOS2 as a viable option but depending on the project progress, we may switch to an alternative method.

- **Edge AD Module:** This module collects the local statistics for each MPI rank and detect local anomalies. To ensure anomaly detection quality, this module will communicate with **Online AD Parameter Server** to exchange the global statistics. For instance, this module send local execution time mean for each function to the online anomaly detection parameter server and receive the updated global mean value from parameter server. By doing so, the local detection has the similar detection power with the global anomaly detection. Another important feature of this module is that once it detects anomaly, it notifies the Online AD Parameter Server, so that Provenance data can be collected and Visualization module can decide what to visualize. Our initial target algorithm is based on distributed online streaming stand deviation and if we have extra time, we plan to implement distributed MiLOF algorithm. MiLOF algorithm is a limited memory streaming implementation of LOF (Local Outlier Factor) algorithm.

- **Online AD Parameter Server:** This module essentially communicates with Visualization and in order to collect provenance information to the on-node SOSFlow listener daemon. Since we only collect condensed statistics of performance traces, this module does not need to handle large amounts of data but still needs to aggregate collected statistics and send the updated results to each edge computing node.

Task ID	Task Name	Priority	Releases	Notes
AD1	Edge TAU Stream Parser	High	v2.5	Potential to be Python
AD2	Edge TAU Stream Parser	Medium	v3.0	Improved version, C/C++
AD3	Online AD Parameter Server	High	v2.5	
AD4	Benchmark Study up to 50 nodes	High		June deliverable, ALL
AD5	Edge STD AD Module	High	v2.5	Potential to be Python
AD6	Edge STD AD Module	Medium	v3.0	Improved version, C/C++
AD7	Edge MiLOF AD Module	Low	v3.0	Optional task
AD8	Benchmark Study up to 1000 nodes	High		Dec. deliverable, ALL

Table 3: Anomaly Detection (AD) Task List with associated priority and release dues .

On top of described Modules, we plan to do benchmark study on leadership computing facilities such as Cori. We listed anomaly detection (AD) tasks in Table 3.

2.6 Scalable Visualization

This year, our foci include: 1) adjust the functionality for distributed performance anomaly detection algorithms; 2) increase the scalability of the streaming visualization framework; and 3) make sure its practical usage in super-computing clusters. Therefore, we prioritize the tasks as follows.

Prototype scalable online visualization (VIS1): we will develop a light-weighted proof of concept system to receive and visualize the results from distributed anomaly detection module. The target is to provide an online visualization tool with minimum computation overhead, but present sufficient information for online performance surveillance. The detailed design includes:

1. Develop a visualization server that listens to streaming performance data online.
2. Co-design the data structures of the anomalous function objects and their call stack; receive them in a streaming fashion from anomaly detection module.
3. Present standard layouts such as scatter plot, timeline and tree structure to visualize the detected anomalous behaviors in the workflow.
4. Transfer and store periodically the generated visualization data into the provenance database.

Online visualization module (VIS2): we will work on enhancing the functionalities of the prototype version to facilitate a unique and useful tool. Specifically, our design includes:

1. Scale up the visualization server to handle parallel massive arrival of data from anomaly detection modules, as well as user interaction.
2. Research on enhanced layouts to accommodate complex function executions (such as long executions, deep nested invocations, and large loops); make sure they can scale up accordingly.
3. Test and refine the performance of user interaction with the visualization module; research on new aggregation approaches or technical implementation methods to reduce the potential rendering bottleneck.
4. Launch the server in super-computing clusters such as Cori and Summit, and test its practical usage.
5. Case studies on actual scientific applications or workflows.

Postprocessing visualization module (VIS3): as the lower priority and optional goal, we will work on the features for post processing analysis within the same framework. This design includes:

1. Develop complex visualization modules to present level-of-detail examination of the entire workflows.
2. Support additional data query mechanism from the provenance database.
3. Test and refine its usage in super-computing clusters.

Table 4 gives the list of visualization tasks and priority associated to each task.

Task ID	Task Name	Priority	Releases	Notes
VIS1	Visualization Framework Prototype	High	v2.5	June deliverable
VIS2	Scalable Visualization Module	High	v3.0	Dec. deliverable
VIS3	Postprocessing Visualization Module	Low	v3.0	Optional task

Table 4: Visualization Task List with associated priority and release dues.

2.7 Provenance

We have developed the concept of prescriptive provenance to reflect extraction of selected performance events, their execution stack, and environment. Prescriptive provenance is the provenance of execution for events identified as anomalies by the distributed anomaly detection. The TAU tools collect provenance together with performance for events in streaming mode.

Prescriptive Provenance Selection (PP1): this is a high priority task.

1. Anomaly detection selects which events are the anomalous events. These events are dumped to disk together with the provenance of their execution by the Edge AD Module following notification by the on-node SOSFlow. The entire trace in memory containing the provenance is saved to disk. Provenance metadata includes node, rank, libraries, configurations, MPI rank, execution environment, etc. Provenance metadata will provide a map of the execution model for each workflow run, including data flow patterns and where calculations take place.

2. Provenance for local anomaly detection: In the case of a locally detected event, on-node SOSFlow will output provenance data and metadata to a predetermined format. We are currently considering OTF2, ADIOS bp format, or bson.

3. Global anomaly detection: In the case of a globally detected event, the SOSFlow aggregator will trigger the output of provenance data and metadata for all nodes in the format agreed above.

Maintenance of extracted metadata (PP2): this is a high priority task. Each anomalous event and its provenance is dumped to disk as described above.

1. Design a method for incrementally augmenting the provenance metadata as it is collected and periodically dumped by the application.

2. Implement data structures for persisting this metadata.

Provenance post-processing and querying (PP3): This is a low priority task and an optional goal for this year.

1. We will design methods for storing and querying provenance off line to present to the scientific user or application developer

2. We will design an interface presenting query results to the user.

Table 5 gives provenance task list and priority associated to each task.

Task ID	Task Name	Priority	Releases	Notes
PP1	Prescriptive Provenance Selection	High	v2.5	June deliverable
PP2	Maintenance of Extracted Metadata	High	v3.0	Dec. deliverable
PP3	Provenance Post processing and querying module	Low	v3.0	Optional task

Table 5: Provenance Task List with associated priority and release dues.

2.8 Chimbuko v3 Impact

Chimbuko v3.0 will enable the first scalable trace level workflow performance analysis. Although there has been performance profile performance analysis tools, we believe it is the first distributed scalable trace analysis tool. This version of Chimbuko release may focus on function call execution time and limited call stack information initially but such call execution anomalies is a good proxy for communication bottleneck and IO issues and we also provide offline analysis capability through anomaly detected provenance information.

Tight coupling in Chimbuko is achieved between the scientific application, the anomaly detection algorithm, the extraction tools, the visualization and the provenance collection of each execution. Advantages of this coupling include data reduction, ease of optimization, usability of results, and increased accuracy of sampling based on actual events rather than time interval.

The goal of the NWChemEx project is to achieve high performance, requiring non-trivial optimization efforts. The effectiveness of this effort needs to be measurable to demonstrate the benefit of investments made. Provenance facilitates this goal by a) identifying performance bottlenecks quicker and b)documenting the effectiveness of performance optimization efforts. In addition, keeping a historical record of this information is further useful so that multiple simulations can be run with different workflow configurations in a co-design study and this information can be mined to discover how anomalous patterns depend on the workflow configuration. Being able to query the execution of a single workflow and analyze the performance anomalies that arise will help identify problems and suggest solutions. Being able to query how the performance changes across different simulation setups will help assessing how much has been achieved and how much of the design space has been explored.

Performance profile analysis can not locate a specific function responsible for the performance bottleneck as it is a summary information. Our proposed trace level performance analysis together with its provenance can enable which function call we need to investigate and which node showed such problems, resulted in thorough and complete performance analysis. In other words, on Exascale computing, there are many parts not correctly working and tracing the cause of performance issues will be nearly impossible but Chimbuko can nail down such problem. Therefore, Exascale application development and tool development can be much easier and LCF (Leadership Computing Facilities) can address their hardware issues much easily as well. Thanks to unique trace analysis capabilities, it is good for both performance analysis and co-design study of hardware and software with aforementioned reasons.

2.9 Major Milestones for Chimbuko v3

Table 6 gives major milestones for Chimbuko v2.5. Table 7 gives major milestones for Chimbuko v3.0.

Month	Milestone
Feb. 2019	TAU1: Re-implement SOS data extraction as C program/plugin TAU2: Design local trace accumulator for event provenance Design scalable visualization framework
March 2019	TAU1: Finish SOS data extraction implementation TAU2: Implement local trace accumulator AD1: Implement Edge TAU Stream processor VIS1: Implement the front end of visualization
April 2019	TAU1-3: Integrated test of extraction, trace accumulator and feedback from SOS AD5: Edge STD AD Module VIS1: Implement the back end of visualization
May 2019	AD3: Online AD Parameter Server Integrated test of full Chimbuko software stack with distributed anomaly detection and triggered trace extraction VIS1: Connect with other Chimbuko modules PP1: Prescriptive Provenance Selection
June 2019	AD4: Benchmark study up to 50 nodes VIS1: Scalable visualization prototype and case study PP2: Mainentance of extracted metadata Delivery of full software stack

Table 6: Milestones for June 2019 Deliverable.

Month	Milestone
July 2019	Begin scalability tests of Chimbuko stack Collect requirements to enhance visualization functionailities
August 2019	AD2: Edge TAU stream parser (improvement) VIS2: implement enhanced features for visualization front end Scalabilty refinements
Sept. 2019	AD6: Edge STD AD module (improvement) VIS2: implement enhanced features for visualization back end Test full software stack with NWChem on HPC system at scale
Oct. 2019	Continue to test full software stack with NWChem on HPC system at scale PP1 and PP2: Improved provenance selection and maintenance of extracted metadata with case study
Nov. 2019	Continue to test full software stack with NWChem on HPC system at scale AD7: (Optional) Edge MiLOF AD Module VIS3: (Optional) Postprocessing features for visualization PP3: (Optional) Provenance post-processing module
Dec. 2019	AD8: Benchmark study up to 1000 nodes VIS2: Case study for visualization module

Table 7: Milestones for Dec. 2019 Deliverable.

3 Dependencies

Some functionalities in Chimbuko depends on third-party libraries, which are listed as fellow:

- Tuning and Analysis Utilities (TAU)
- Scalable Observation System (SOS)
- EVPath
- Adaptable Input / Output System (ADIOS2)
- D3.js
- Flask

4 People and Effort

People	Effort Level (FTE)
Kevin Huck	0.15
Shinjae Yoo	0.25
Wei Xu	0.4
Line Pouchard	0.1
SW Developer (TBD)	1.0
Research Assistant (TBD)	1.0

Table 8: People and effort level.