From Abstractions to MODELS: MOdels for Distributed and Extremely Large-scale Science

Shantenu Jha[1], Daniel S. Katz[2], Matteo Turilli[1], Jon Weissman[3]

[1] RADICAL Laboratory, Electric and Computer Engineering, Rutgers University, New Brunswick, NJ, USA
[2] Computation Institute, University of Chicago & Argonne National Laboratory, Chicago, IL, USA
[3] Computer Science and Engineering Department, University of Minnesota, Minneapolis, MN, USA

Many important advances in science and engineering are due to large-scale distributed computing. Notwithstanding this reliance, we are still learning how to design and deploy large-scale production Distributed Computing Infrastructures (DCI). This is evidenced by missing design principles for DCI, and an absence of generally acceptable and usable distributed computing abstractions. These gaps underlie the following observations:

- Distributed applications are characterized by limited functional sophistication, yet they are not simple to develop. “Heroic” effort is required to add or incorporate new functionality. This suggests a possible lack of reliable abstractions for distributed applications.
- Existing production DCIs such as OSG and XSEDE are designed for specific classes of applications. It is neither easy to port applications from one DCI to another, nor to expand the type of applications supported by each DCI. Furthermore, it is not easy to interoperate across production DCIs. This is due, at least in part, to a lack of infrastructure abstractions that represent DCI well enough to enable developers to build richer applications.
- The lack of an adequate set of abstractions for distributed applications and infrastructure, as well as design principles for DCI results in: (i) applications that are brittle, (ii) an inability to reason about the spatiotemporal execution of distributed workloads, and answer even basic questions about their execution. For example: Which resources should a workload use? What time-to-solution should a workload expect? How do specific execution decisions influence the performance?

The AIMES project was conceived against this backdrop, following on the heels of a comprehensive survey of scientific distributed applications [1]. The survey established, arguably for the first time, the relationship between infrastructure and scientific distributed applications. It examined well known contributors to the complexity associated with infrastructure, such as inconsistent internal and external interfaces, and demonstrated the correlation with application brittleness. It discussed how infrastructure complexity reinforces the challenges inherent in developing distributed applications.

We have identified several additional contributing factors responsible for infrastructural complexity, for example, the absence of extensible and scalable abstractions for resource management. One consequence of which is that different applications manage their execution and distributed resource utilization in diverse if not irreconcilable ways, both in principle and practice. This results in applications being bound to specific platforms, thus acting as a further barrier to interoperability. Determining widely-usable abstractions for resource management is a non-trivial undertaking, made harder by the fact that DCIs are characterized by both spatial and temporal fluctuation.

An important driver of AIMES was to first identify such abstractions, and then to translate them into scalable and uniform distributed resource management approaches and capabilities. A relevant prerequisite was the need to “formalize and normalize” the set of decisions that are needed to execute a distributed application on multiple and diverse resources. A related issue was integrating information from the application and infrastructure as the nature of execution decisions is intimately related to the type of information available.

In the absence of adequate abstractions, how can we design experiments to investigate or support different ways to execute distributed applications, while being extensible and scalable to real science

applications? Furthermore, given the broad range of application properties, resource characteristics, and the type and reliability of resource information, how can we "experiment" with and "integrate" different application requirements, infrastructure types, and information?

AIMES laid the foundations to address the tripartite challenge of dynamic resource management, integrating information, and portable and interoperable distributed applications. Four abstractions were defined and implemented: **skeleton**, **resource bundle**, **pilot**, and **execution strategy**. Skeletons are synthetic applications with a spectrum of characteristics and tunable properties to mimic the behavior of real-life distributed applications [2]. Resource bundles represent an aggregate set of resources of fluctuating capacity that hide the heterogeneity and dynamism inherent in DCI resources. The pilot abstraction is well known in distributed computing but inconsistently used and understood. A sound theoretical basis was provided for the pilot abstraction [3, 4] and its applicability was extended also to high-performance and data-intensive computing [5, 6]. Execution strategies model the set of decisions that need to be made in order to execute a distributed application on diverse resources [7].

The four abstractions were implemented into software modules and then aggregated into the AIMES middleware. This middleware successfully integrates information across the application layer (skeletons) and resource layer (bundles), derives a suitable execution strategy for the given skeleton and enacts its execution by means of pilots on one or more resources, depending on the application requirements, and resource availabilities and capabilities.

Experience from AIMES

AIMES has enhanced our understanding of extreme-scale distributed science via the execution of applications and the dynamic federation of resources [8].

- In PY1, we designed, developed, and integrated preliminary implementations of abstractions for distributed applications and resources.
- In PY2, we experimented with a range of execution strategies on dynamic federation of heterogeneous resources via pilot overlays on large-scale production DCI.
- In PY3, we are generalizing the core capabilities to a wider range of resource and information types, applications, and infrastructure classes. Furthermore we are exploring the logical federation of resources with well defined capacity from time-varying resources overlays.

AIMES has advanced the state of distributed computing in at least three important ways:

1. We have designed and implemented powerful and extensible abstractions for distributed applications and resources. Collectively, they provide the ability to execute applications on dynamically varying resources. However, exposing simplicity is not the same as hiding complexity. To that end, we support scalable execution of a range of applications on production DCI by integrating abstractions at multiple levels. Scale, generality, extensibility, reasoning, and repeatability are the basis for the claim that the complexity of DCI at large scale can be managed via abstractions. Collectively, the abstractions provide an integrated approach to resource-management in distributed, high-performance and data-intensive scenarios, which hitherto were considered distinct.
2. Building upon the ability to integrate information, we have proposed the abstraction of execution strategy. Execution strategies support end-to-end reasoning by capturing the decisions involved in the execution of different applications independent of the underlying resource utilization model. Furthermore, the AIMES infrastructure can take an execution strategy and implement it for a distributed application. This permits empirically determining a suitable, if not yet optimal, execution strategy for a given application using a given set of resources, and thus provides the ability to examine the impact of execution decisions on scalable execution as a function of different application and resource characteristics. We have shown that execution strategies can improve the qualitative

and quantitative aspects of distributed execution [7] dissolving the artificial barriers between high-throughput and high-performance.

3. We have implemented an experimental "laboratory" that supports investigation of questions and along dimensions that were not originally conceived. The AIMES infrastructure supports the design of new experiments and analyzing tradeoffs of different decisions; the same software system also supports scalable science. For example, the AIMES infrastructure supports experimental workloads on five and more distributed resources, but the RADICAL-Pilot component also supports thousands of MPI jobs used for biophysical simulations [9].

These advances should be benchmarked against an earlier landscape dominated by distributed applications that needed high-levels of customization and limited scalability on the one hand, with DCIs that were characterized by "gluing it together" and the absence of a systems approaches on the other.

What are the important questions that AIMES motivates us to ask?

Building upon advances arising from skeletons, bundles and pilots, execution strategies allow qualitative reasoning about distributed workload execution that is resource and workload agnostic. There remains, however, a critical need for a quantitative basis for distinction between different possible decisions. For example:

- What are the constraints on the quantitative advantages arising from execution strategies? How are these dependent on workload properties? Or on resource availability? How does this translate into making "effective" scheduling decisions? How can we estimate different metrics or measures of performance?
- How sensitive is the planning and execution of complex workloads on the specifics of distributed infrastructure? How can different heterogeneous infrastructure be unified by a time-dependent capacity model that goes beyond point prediction models?
- How will we design the next DCI to meet the requirements of multiple science projects with predefined quality-of-service? Is it possible to derive models that allow us to "design infrastructure" for specific performance and requirements?

To answer these questions for current and future distributed computing systems requires new research along the following lines:

- From modeling to models: Models to answer the aforementioned questions, if any, are currently restricted to specific application scenarios and infrastructure, and are not general purpose. Quantitative modeling of execution must include models of DCI resources, middleware, and applications. Planning and execution would be more precise with models of DCI than without, but such robust models do not exist yet. How do we build macroscopic models of application execution that are not unmanageably sensitive to microscopic heterogeneities? How would models that support design for specific performance generalize to design principles and architectures for next generation of distributed computing infrastructure? The answer to these questions cannot be derived from a single monolithic model, but require a hybrid set of models at multiple levels, different scales, and granularity. How are such models designed and composed? How can models support quantitative end-to-end reasoning?
- From sensing to actuation: We must transition from resource federation of known and predetermined resources to federation of resources that are discovered dynamically. How does the addition of resources that were not known at the time of initial planning change the course of execution? How do the kind of resources that the middleware "actuates" be influenced by application properties? These questions reiterate the need for quantitative models of applications, resources, and execution.

- Dynamism at extreme scale: Dynamism is going to become more pervasive, diverse, and significant as we move towards greater scale. For example, with better resource discovery and new types of resources, there will be a larger number of resources resulting in greater variation in their temporal properties. Different classes of applications will result in more burstiness in application and data production. In other words, dynamism is set to become a dominant and first class property of both distributed applications and DCIs at larger scale. Inter alia, this will require better algorithms that are more responsive and better able to handle dynamism along the different dimensions. Last but not least, AIMES was designed upon the assumption that information could be dynamically integrated across levels. To what dynamic scales do these assumptions and approaches work? Answers to these questions require conceptual advances in the way we characterize dynamism in large-scale DCIs, as well as a radically different approach to responding to it.

Each of these research tracks and themes represents fundamental challenges as the community attempts to take distributed computing to the next scale. Each track encompasses a plethora of questions in turn. We hope to pursue these important questions as logical continuation of and within the powerful framework of AIMES.

References

[1] Shantenu Jha, Murray Cole, Daniel S. Katz, Manish Parashar, Omer Rana, and Jon Weissman. Distributed computing practice for large-scale science and engineering applications. *Concurrency and Computation: Practice and Experience*, 25(11):1559–1585, 2013. http://dx.doi.org/10.1002/cpe.2897.

[2] Daniel S. Katz, Andre Merzky, Zhao Zhang, and Shantenu Jha. Application skeletons: Construction and use in eScience. *Future Generation Computer Systems*, 2015. http://dx.doi.org/10.1016/j.future.2015.10.001.

[3] Andre Luckow, Mark Santcroos, Andre Merzky, Ole Weidner, Pradeep Mantha, and Shantenu Jha. P*: A model of pilot-abstractions. *IEEE 8th International Conference on e-Science*, pages 1–10, 2012. http://dx.doi.org/10.1109/eScience.2012.6404423.

[4] Matteo Turilli, Mark Santcroos, and Shantenu Jha. A Comprehensive Perspective on Pilot-Jobs, 2015. (under review) http://arxiv.org/abs/1508.04180.

[5] Andre Luckow, Mark Santcroos, Ashley Zebrowski, and Shantenu Jha. Pilot-Data: An Abstraction for Distributed Data. *Journal Parallel and Distributed Computing*, October 2014. http://dx.doi.org/10.1016/j.jpdc.2014.09.009.

[6] Andre Luckow, Ioannis Paraskevakos, and Shantenu Jha. Pilot-Abstraction: A Valid Abstraction for Data-Intensive Application on HPC, Hadoop and Cloud Infrastructures?, 2015. http://arxiv.org/pdf/1501.05041v1.pdf.

[7] Matteo Turilli, Feng (Francis) Liu, Zhao Zhang, Andre Merzky, Michael Wilde, Jon Weissman, Daniel S. Katz, and Shantenu Jha. Integrating Abstractions to Enhance the Execution of Distributed Applications. In *Proceedings of 30th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016. http://arxiv.org/abs/1504.04720.

[8] AIMES Presentations: SC14, NGNS PI Meeting and MAGIC Meeting, http://goo.gl/xJubgS.

[9] Brian K. Radak, Melissa Romanus, Tai-Sung Lee, Haoyuan Chen, Ming Huang, Antons Treikalis, Vivekanandan Balasubramanian, Shantenu Jha, and Darrin M. York. Characterization of the Three-Dimensional Free Energy Manifold for the Uracil Ribonucleoside from Asynchronous Replica Exchange Simulations. *Journal of Chemical Theory and Computation*, 11(2):373–377, 2015. http://dx.doi.org/10.1021/ct500776j.