Deciding When and How to Move HPC Jobs to the Cloud

Marco A.S. Netto, Renato L.F. Cunha, and Nicole Sultanum, IBM Research

Now used for high-performance computing applications, the cloud presents a challenge for users who must decide, based on efficiency and cost-effectiveness, when and how to run jobs on cloud-based resources versus when to use on-premise clusters. The authors propose a decision-support system to help make these determinations.

The cloud began as a platform to host Web applications but has since been used for many other types of programs, including those for high-performance computing (HPC). These applications have become an integral part of numerous domains including seismic research for oil and gas exploration, high-resolution solid and fluid mechanics, social-media analytics, and molecular dynamics. While HPC users often have access to on-premise computing clusters, these resources might be insufficient for their application executions, known as jobs, or might force their jobs to wait a long time in a queue.[1] Thus, HPC researchers are exploring the benefits of moving resource-intensive jobs, to the cloud.[2–4]

Renting HPC clusters in the cloud has recently become easier and less expensive. For instance, users can rent a 20-node cluster built with virtualized machines that have 64 Gbytes of RAM and 16 cores each for US$14 per hour. The same cluster using physical machines would rent for US$23 per hour. These prices drop another 10 percent for monthly rentals.

Organizations can rent cloud resources to augment their local computing capacity to meet increasing demand, creating a hybrid operation. However, this creates challenges such as how to decide which jobs should be moved to the cloud and when.

Here, we examine these challenges and describe a tool we're currently developing to help users determine whether their jobs should be run in on-premise or cloud-based clusters.

CHALLENGES FOR HPC CLOUD USERS

Moving HPC jobs to the cloud is a cultural, as well as technical, issue that affects users and IT infrastructure administrators. Users frequently act as if on-premise HPC resources are cost-free and thus don't always utilize them carefully. On the other hand, they appear more aware that the cloud is not free and must be accessed wisely. Users would benefit from a tool that helps them decide whether and how to use on-premise or cloud resources for various tasks.

0018-9162/15/$31.00 © 2015 IEEE

EDITOR **SAN MURUGESAN**
BRITE Professional Services;
san@computer.org

Such a tool would be of value, because although the cloud offers many advantages, it also presents challenges. For example, cloud use entails latency. Tightly coupled parallel applications require processors to communicate among themselves via a high-speed network. Without such a network, many parallel applications don't scale well, causing users to choose to work with on-premise resources.

A significant bottleneck occurs between the user infrastructure (including systems ranging from laptops to clusters) and the cloud. This can ruin the experience for users, who expect quick access to HPC cloud applications' output for purposes such as visualization and analysis.

UberCloud—an online community and marketplace where engineers and scientists discover, try out, and buy computing as a service—reports challenges that companies face when moving HPC workloads to the cloud.[5,6] The US Department of Energy's *Magellan Report on Cloud Computing for Science* contains analyses on running HPC and data-intensive applications in the cloud.[7]

A potential problem for users is estimating the cost of running HPC applications in the cloud. They generally don't know a priori how long their applications will have to run, as many programs might present irregular behaviors that make predicting execution times difficult.[8] Even when users try to estimate this, they frequently can't predict how many application instances will be required because they might need to make multiple computations with different input parameters.

In traditional HPC facilities, such as universities and research centers, users already struggle with estimating how much time they'll need to use on-premise resources. This is more complex in the cloud, in which users are charged based on application running time but the pricing models aren't necessarily linear or certain. For example, on an hourly subscription, they will be charged for a full hour of usage even if their task took only 20 minutes. In some cases, cloud providers will charge less for resources that they have the right to terminate at any time, which are suitable for fault-tolerant applications and services. Thus, estimating costs for the cloud is a daunting, yet crucial, task.

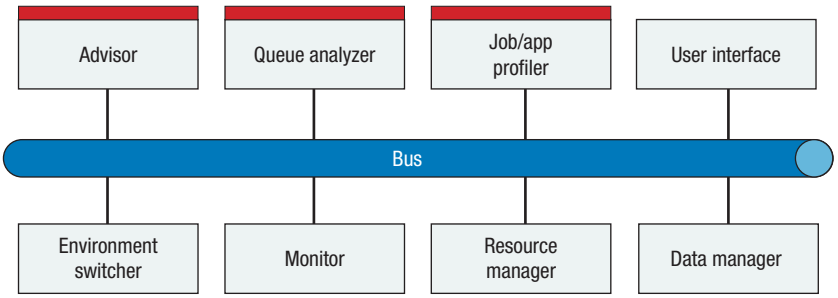

Figure 1. Decision-support system for running high-performance computing (HPC) applications. System components communicate with one another through a common bus. Via an interface, users input their job and business requirements and receive information about the cost of running their task utilizing the HPC cloud versus utilizing on-premise resources, and about which option makes more sense. The components with the red band are the most challenging to design and implement.

HPC DECISION-SUPPORT SYSTEM

Implementing a hybrid HPC cloud entails several challenges. A major, and largely overlooked one, is awareness of the potential cost of using the cloud, given various job-allocation scenarios. This is necessary for users to determine when running a job in the cloud makes economic sense. To help users, we're developing a decision-support system (DSS) designed to forecast the cost of running HPC applications in the cloud under several possible configurations, allowing for easier comparison among alternatives. The DSS also specifies levels of uncertainty about job-execution time and cost estimates caused by forecasting-model imperfections. Figure 1 shows the DSS components.

Advisor

This main DSS component helps users determine the least expensive way to execute jobs while still generating timely results. To accomplish this, the advisor receives information from other components about the time required to access resources, such as on-premise clusters' queue lengths and the way jobs would run in different computing-system environments and configurations.

The advisor doesn't try to tell users what the best job-allocation solution is because the criteria for that may be too unclear or subjective to express in computational terms. Instead, it looks at the information gathered from other components and provides easy-to-understand suggestions about where best to run jobs such as "if your job takes more than 4 hours, you

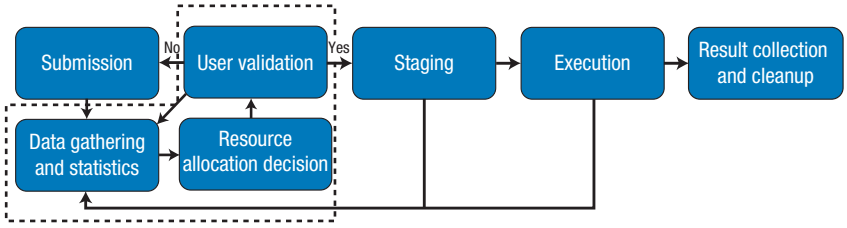

Figure 2. HPC decision-support system's job flow. The dashed line represents where the DSS's advisor, queue analyzer, and job/app profiler operate. If the users don't validate the system's job-allocation suggestions, they resubmit the job. If they do validate them, the job goes to staging, from which data and applications are transferred. Cloud resources are provisioned if the job is running in the cloud. Otherwise, the application is queued in the on-premise resource manager.

should run it in the on-premise cluster." Ultimately, the user is in charge of the process.

Queue analyzer

A major advantage of using cloud systems is resource availability, as utilizing on-premise clusters frequently entails long waits in queues. The queue analyzer predicts the wait times that various jobs would experience. The challenge is determining how various cluster-management policies affect prediction accuracy.

A few existing systems such as QBETS (queue bounds estimation from time series)[9] and the US National Science Foundation Extreme Science and Engineering Discovery Environment's Karnak[10] propose to tackle this challenge. We're developing our own queue analyzer based on these systems' techniques.

Job/app profiler

Benchmarks are necessary to determine the best way to use cloud resources: they help identify how a job would perform in different environments and configurations. However, extensive benchmark execution is neither timely nor economically sustainable. Thus, the profiler combines benchmarks with information obtained from other sources. For example, users could provide information on budgets and resource-access time limits; and historical data of past job executions could yield information on resource-access times, execution times, and the number of processors allocated.[11] This eliminates the need to execute benchmarks for resource-allocation decisions not of interest to users.

User interface

Sometimes neglected by the HPC community, the user interface (UI) is critical for providing good, clear time- and cost-management information. Developers could employ data visualization techniques to create meaningful, functional, and information-rich interfaces.

Environment switcher

Between job submission and execution, the on-premise cluster's queue status could change or users might decide to run tasks in another environment. In such cases, the system must be able to move jobs between environments. If the job hasn't started yet, the switcher interfaces with the resource manager to remove the job from or add it to the cluster queue. If the job is already running, the switcher relies on checkpointing to save the job's execution state in one environment and restore it in another.

Monitor

To provide the status of jobs being executed, the system should monitor itself. For example, if a job reaches a problematic state or the system deviates from its predicted behavior, the monitor could issue a notification.

Resource manager

To execute jobs in the cloud, the system must provision cloud resources with the necessary operating system and libraries. Our tool uses cloud-provider APIs, which contain functions to allocate, release, and configure cloud resources. These functions allow the integration of the cloud resources with an on-premise cluster-management system such as the Platform Load Sharing Facility (LSF), the Portable Batch System (PBS), the Terascale Open-Source Resource and Queue Manager (TORQUE), and the Simple Linux Utility for Resource Management (SLURM).

Data manager

Most nontrivial jobs must read input data and produce output data. Output information must be available for use after the system executes a job. Simply copying all data before job execution and then copying the new output information afterward might not be cost-effective if there is a lot of information, especially if the link between on-premise and cloud resources has low throughput. Instead, the system needs data synchronization to function efficiently.

Another potentially more cost-effective alternative would be placing the information that is likely to be used in an inexpensive cloud-based object storage service with high data-access rates for cloud instances.

CONTROL FLOW

Figure 2 shows our proposed system's control flow. Once the user submits a job, the system gathers user constraints—such as the deadline for job completion and the available budget—while also fetching queued data and estimated execution times from other components. If the job/app profiler doesn't already have

information about the type of job being run, it might execute benchmarks, ask users to provide predictions, or take a conservative approach and provide an initial overestimate of execution times. In addition, the data manager calculates estimated data-transfer times—which influence job-placement decisions when the amount of information moving between on-premise and cloud resources is large—by using historical data from previous transfers stored in the data manager.

With all this information, the advisor calculates and ranks the cost of running the job in different environments,[1] presenting various options to the users, who then make a choice based on their own time- and cost-related priorities. After user validation, the system moves the job to the selected environment.

During execution, the monitor evaluates the job's running time and cost, and if these values rise above a user-defined threshold, the system triggers an alarm. This could lead to additional data gathering and calculation, causing the advisor to suggest alternative configurations in the execution environment.

In the short run, organizations must benchmark their frequently used applications to evaluate the cost benefit of migrating them to the cloud. They must also track the frequency with which applications are executed because this impacts decisions about whether and how to use cloud or on-premise clusters. For example, frequently utilized applications that demand a lot of resources should run in on-premise clusters to reduce cloud-related costs.

For hybrid environments, resource-allocation policies should carefully match jobs and environments. For instance, tightly coupled parallel applications or data-intensive applications should use on-premise resources because slow network connections would cause them to take too long to run in the cloud, which would increase expenses.

Several of these decision-making processes are still done manually or are scripted by experts because, in these cases, estimating execution times and the amount of resources required is complex and difficult. Thus, tools like the one we're developing are necessary to improve and automate more of these decisions.

HPC cloud use is an important topic of investigation. More efficient resource utilization could benefit organizations. And observing users' behavior could shed light on their hard-to-model, subjective criteria for job allocation. Systems could leverage this understanding to make job-configuration suggestions more relevant and helpful, enabling efficient and well-informed decision making. *

REFERENCES

1. A. Marathe et al., "A Comparative Study of High-Performance Computing on the Cloud," *Proc. 22nd Int'l Symp. High-Performance Parallel and Distributed Computing* (HPDC 13), 2013, pp. 239–250.
2. A. Gupta et al., "The Who, What, Why and How of High Performance Computing Applications in the Cloud," *Proc. 5th IEEE Int'l Conf. Cloud Computing Technology and Science* (CloudCom 13), 2013, pp. 306–314.
3. M. AbdelBaky et al., "Enabling High-Performance Computing as a Service," *Computer*, vol. 45, no. 10, 2012, pp. 72–80.
4. C. Vecchiola, S. Pandey, and R. Buyya, "High-Performance Cloud Computing: A View of Scientific Applications," *Proc. 10th Int'l Symp. Pervasive Systems, Algorithms, and Networks* (ISPAN 09), 2009, pp. 4–16.
5. W. Gentzsch and B. Yenier, *The UberCloud HPC Experiment: Compendium of Case Studies*, tech. report, Tabor Communications, 2013.
6. W. Gentzsch and B. Yenier, The *UberCloud Experiment: Tech. Comp. in the Cloud—2nd Compendium of Case Studies*, tech. report, Tabor Communications, 2014.
7. M. Leads et al., *The Magellan Report on Cloud Computing for Science*, tech. report, US Department of Energy-Office of Science-Office of Advanced Scientific Computing Research, 2011.
8. C.B. Lee and A. Snavely, "On the User-Scheduler Dialogue: Studies of User-Provided Runtime Estimates and Utility Functions," *Int'l J. High Performance Computing Applications*, vol. 20, no. 4, 2006, pp. 495–506.
9. D. Nurmi, J. Brevik, and R. Wolski, "QBETS: Queue Bounds Estimation from Time Series," *Proc. 13th Int'l Workshop Job Scheduling Strategies for Parallel Processing* (JSSPP 07), 2007, pp. 76–101.
10. W. Smith, "A Service for Queue Prediction and Job Statistics," *Proc. Gateway Computing Environments Workshop* (GCE 10), 2010; doi: 10.1109/GCE.2010.5676119.
11. D. Tsafrir, Y. Etsion, and D. Feitelson, "Backfilling Using System-Generated Predictions Rather than User Runtime Estimates," *IEEE Trans. Parallel and Distributed Systems*, vol. 18, no. 6, 2007, pp. 789–803.

MARCO A.S. NETTO is a researcher at IBM Research–Brazil, manager of the Industrial Cloud Technologies Group, and an IBM Master Inventor. Contact him at mstelmar@br.ibm.com.

RENATO L.F. CUNHA is a researcher in IBM Research–Brazil's Industrial Cloud Technologies Group. Contact him at renatoc@br.ibm.com.

NICOLE SULTANUM worked at IBM Research–Brazil and is now a University of Toronto PhD candidate. Contact her at nicolebs@cs.toronto.edu.

* This work has been partially supported by FINEP/MCTI under grant no. 03.14.0062.00.