MIDDLEWARE FOR MULTICLOUD

Cost-Aware Cloud Profiling, Prediction, and Provisioning as a Service

Ryan Chard, Argonne National Laboratory
Kyle Chard, University of Chicago and Argonne National Laboratory
Rich Wolski, University of California, Santa Barbara
Ravi Madduri, University of Chicago and Argonne National Laboratory
Bryan Ng and Kris Bubendorfer, Victoria University of Wellington
Ian Foster, University of Chicago and Argonne National Laboratory

Here we present Scalable Cost-Aware Cloud Infrastructure Management and Provisioning (SCRIMP)—a service-based system that enables application developers and users to reliably outsource the task of provisioning cloud infrastructure. We show that by understanding application requirements, predicting

dynamic market conditions, and automatically provisioning infrastructure according to user-defined policies and real-time conditions that our approaches can reduce costs by an order of magnitude when using commercial clouds while also improving execution performance and efficiency.

The advent of "big data science" has revolutionized almost every scientific discipline, transforming practices that only a decade ago were rudimentary into those that are now reliant on computation- and data-driven methods. Cloud computing has played a significant role in this rapid transformation as it provides *any* researcher with on-demand access to infinite, virtualized infrastructure. As such, a significant number of scientific applications are now hosted on cloud infrastructure.[1] While the most prominent cloud providers remain commercially hosted services (e.g., Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform), there are an increasing number and variety of national cloud testbeds (e.g., Jetstream, Chameleon, and CloudLab) institution clouds (e.g., Argonne National Laboratory's Magellan, National Center for Supercomputing Applications' Nebula, and San Diego Supercomputer Center's OpenStack deployment) and federated clouds (e.g., Aristotle at Cornell, Buffalo, and University of California, Santa Barbara (UCSB)) available to researchers. Given this plethora of cloud platforms, and the inherent differences between them, new methods are needed to efficiently make use of multicloud environments.

Unfortunately, effective and efficient use of cloud infrastructure is technically challenging, and these challenges are exasperated in a multicloud environment. Ultimately, users desire the ability to acquire the "best" resources on which to host their workloads. However, determining the "best" provider or instance type is dependent on many factors, such as data locality, resource cost, execution time, security properties, etc. Often, tradeoffs exist when trying to satisfy these goals, however at present it is difficult, if not impossible for users to explore or balance these tradeoffs. Beyond simply selecting providers and instance types, there are a range of inherent technical challenges associated with acquiring, configuring, and managing the virtual machines (or *instances*) leased from cloud providers. These challenges range from acquiring infrastructure to issues related to data and virtual machine (VM) placement, inter-VM communication, and security. In this article, we focus on the challenges related to provisioning and managing cloud infrastructure independent of data, communication, and security needs. Specifically, we focus on provisioning challenges which include:

1. Users require an understanding of cloud mechanics, such as the available instance types and pricing models, in order to efficiently utilize cloud infrastructure;
2. Applications perform very differently on different cloud providers and even across instance types within a single provider;
3. The cloud is not a panacea for "cheap" and efficient resource usage—naïve provisioning approaches can result in increased cost and poor execution performance;
4. Managing large pools of compute resources is challenging on dedicated infrastructure and doing so on elastic (and often preemptable) multicloud environments further complicates this task.[2]

To address these challenges, we have developed Scalable Cost-Aware Cloud Infrastructure Management and Provisioning (SCRIMP)—a multicloud middleware service that enables the outsourcing of provisioning decisions on behalf of (scientific) applications. SCRIMP optimizes the cloud provisioning process for batch workload-based applications by combining our previous contributions in three key areas: application *profiling*, market *prediction*, and automated cloud *provisioning*, into a single comprehensive service.[3–5] We discuss here, for the first time, how these three areas have been integrated to deliver a flexible cloud provisioning service.

- **Profiling:** Before being able to efficiently provision cloud infrastructure, SCRIMP must first understand how an application is likely to perform on heterogeneous cloud infrastructure. This can be done by profiling application performance on a range of instance types to estimate performance; however, the enormous search space of applications and instance types necessitates the need to automate this process. We have developed an automated profiling service that is able to derive approximate profiles for applications executed on different environments.[3]
- **Prediction:** Cloud providers typically implement a pricing model to facilitate equitable usage of resources. Note: in some cases, non-tangible credit (or allocation) based economies are used in private and institutional clouds. A variety of models are used to match supply and demand including both static and dynamic markets. Some providers also offer dynamic pricing models as an efficient means of establishing a market price (e.g., AWS Spot Market). In these markets, instances can be acquired at a lower cost with reduced service level agreements (SLAs). Thus, when provisioning infrastructure, the task of calculating the cost to execute a particular workload is dependent on the cloud market. To address the complexity of comparing posted price with dynamic markets we have explored methods to predict dynamic market conditions with the aim of computing bids that are based on probabilistic-durability guarantees.[4] This approach allows SCRIMP to compare costs and quantify the risk of instance revocation in the spot market.
- **Provisioning**: Finally, given information obtained from profiles and market prediction SCRIMP aims to provision infrastructure and manage it throughout the course of workload execution. SCRIMP is offered as a user-oriented provisioning service that facilitates outsourcing of provisioning decisions to meet user requirements (e.g., optimizing run time, reducing costs, etc.).[5] This service employs a modular architecture to facilitate provisioning of infrastructure from different cloud providers to satisfy the needs of workloads generated by different applications.

Harnessing the Cloud for Research Computing

To satisfy the growing reliance on data-driven research, scholars are increasingly forgoing on-premise infrastructure and moving to cloud-based solutions. This migration is most often motivated by sporadic computational requirements or a lack of alternative resources. While some researchers use cloud resources as a direct replacement for local resources, we are primarily concerned with those who wish to exploit cloud platforms as an "infinite" elastically scalable execution environment.

Many frameworks have been developed to simplify the use of elastic cloud computing infrastructure. In effect, these systems implement a science gateway model in which computing infrastructure is abstracted via easy to use interfaces. Two such examples are the Globus Galaxies platform and Cloud Kotta.[6,7] The Globus Galaxies platform supports the construction and execution of scientific workflows. It relies on Globus to manage data and HTCondor to manage execution of tasks orchestrated by a workflow management system.[8] The platform has been used in scientific domains as diverse as biomedicine, cosmology, and climate and energy policy. The most widely adopted deployment, Globus Genomics, currently serves more than 300 researchers at 30 institutions, and consumes more than half a million instance hours per year. Cloud Kotta supports the management and analysis of valuable, proprietary, or sensitive social science datasets. Given the growth of such data, there is a need to provide elastic computing capacity "near" to the securely stored data. Cloud Kotta has been used by dozens of researchers for text analytics, network analysis, and machine learning tasks.

These systems represent an increasingly common "pattern" for leveraging elastic cloud infrastructure to execute workloads on-demand. The general idea is that they provision cloud infrastructure (i.e., instances) on-demand to satisfy the needs of real-time workload and then terminate instances when they are no longer needed. As such, each system has developed its own software to assess waiting workload, utilize cloud APIs to provision resources, compare instance prices, and manage provisioned instances throughout their lifetime. While the specific execution mode may differ (e.g., some systems submit individual applications or scripts for

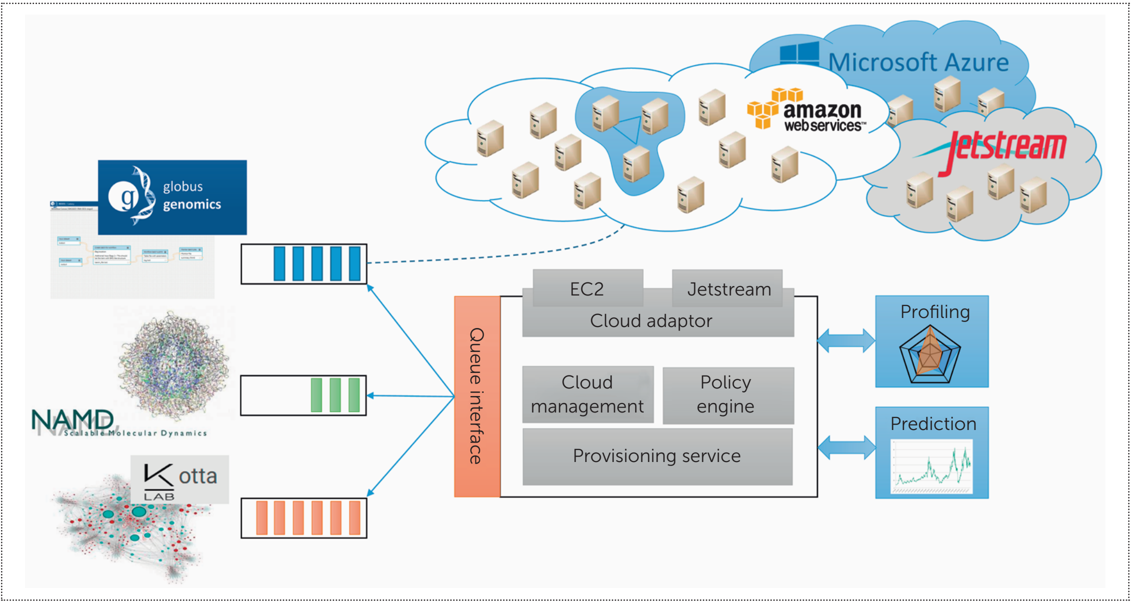


FIGURE 1. General SCRIMP architecture. Several queues, each associated with a different application, are monitored through the queue interface. The cloud adapter interface enables resources to be provisioned across different cloud platforms. SCRIMP leverages application profiles and market predictions to make policy-based provisioning decisions.

execution while others decompose complex workflows into a series of discrete "jobs"), the same provisioning models are used to acquire cloud infrastructure. Both the Globus Galaxies platform and Cloud Kotta include simple provisioning models in which an instance is selected from a list of instances that are predetermined to be capable of satisfying the requirements of any job.

Given the growing popularity of this pattern, several autoscaling and cloud brokering systems have been developed (e.g., Netflix's Fenzo and MIT's StarCluster).[9] These systems focus only on the need to automate the process of launching, configuring, and managing clusters of cloud instances. That is, they are independent from the actual application or management system that submits jobs.

Existing autoscaling systems often rely on the user to define static bid strategies (e.g., 50% on-demand price, or $1 per hour) and select specific instance types to accommodate workloads. They are therefore not capable of adapting to workload requirements or real-time market conditions. Techniques to support users and aid the decision-making process are actively being researched, with recent advancements, for example, showing that Machine Learning techniques can provide adaptive provisioning decisions for a computational linguistics application.[10]

SCRIMP aims to provide these cloud provisioning and management capabilities as a service such that external applications or services, including Globus Galaxies and Cloud Kotta, can outsource their cloud integration and provisioning requirements. SCRIMP removes the need for developers to implement custom software to perform these activities by automating many of the laborious and complicated tasks associated with efficient cloud utilization, e.g., responding to real-time market conditions and selecting optimal instance types for a given workload. Further, SCRIMP provides advanced capabilities for selecting instances that can improve efficiency, reduce the cost of executing workloads, and also remove the need for developers to implement and maintain sophisticated provisioning software.

SCRIMP: The Basic Model

SCRIMP aims to realize a new model for interacting with multicloud infrastructure by enabling developers (and potentially even users) to manage tradeoffs between cost, risk, and execution time. For example, users can select, on a per-job basis, from a two-dimensional surface that represents cost and time. SCRIMP is implemented as a standalone service that can be integrated with external applications and services through its Representational State Transfer APIs.

Figure 1 shows the overall SCRIMP architecture. To support diverse user requirements, SCRIMP implements a modular architecture with extensible internal and external interfaces: queue monitor, cloud adapter, profiles, market prediction, and provisioning policy. Each of these interfaces permits the easy substitution of alternative components (e.g., alternative job queuing systems, cloud platforms, or market prediction methods) to meet user requirements.

To address a wide range of workload modalities, and to easily integrate SCRIMP with existing applications, we focus on queue-based (batch) job submission models. SCRIMP offers a queue adapter interface which can be implemented for different queue systems. Queue adapters require methods to retrieve pertinent information about the queue and particular jobs in the queue (e.g., application, input data, and any performance hints such as those defined in HTCondor class ads). We have developed queue adapters for HTCondor and Apache Spark.

SCRIMP offers a cloud platform interface to support different cloud providers. This interface defines methods to discover instance types, introspect instance details, provision instances of different types, and retrieve market prices (where supported). Initially we have developed support for AWS. We intend to add support for other commercial and academic cloud platforms in the near future.

Finally, one of the aims of SCRIMP is to utilize historical information to improve future provisioning decisions. As such, SCRIMP is designed to record information such as job requirements, application profiles, input data, output data, selected instance type, and periodic instance resource usage information (e.g., CPU, memory). This data is stored in a database accessible to SCRIMP and can subsequently be used to inform provisioning decisions.

Profiling Application Performance on Heterogeneous Infrastructure

Before provisioning resources for an application, SCRIMP must first understand the requirements of that particular application and be able to estimate how it will perform when executed on different cloud instances. This task is complicated by the fact that application requirements may differ across many dimensions including application configuration, execution environment, and input data. In order to make the "best" provisioning decision we require fine grain information about an application. We term this information a "profile"—a concise description of the performance and CPU, memory, network, and disk requirements of an application under different environments and scenarios. Unfortunately, there is no simple method for deriving application profiles and it has been shown that estimates provided by users can be particularly inaccurate.[11]

Characterizing the performance and requirements of applications has long been an area of study, most often in high performance and distributed computing environments.[12] However, in most cases, the target infrastructure is homogeneous. This is, of course, in stark contrast to the dozens of instance types available on many cloud platforms (AWS alone offers over 50 different instance types). Furthermore, cloud providers describe capabilities in different formats and use different units (e.g., vCPUs) making it nearly impossible to compare "apples with apples" based on advertised capacity. While this vast selection offers great flexibility, it further complicates the provisioning process as users need to understand intimately the differences between instance types and providers. Given the enormous search space, it is infeasible to determine application requirements manually. Consider, for example, that the resource requirements and execution performance of a tool is dependent on input datasets, invocation parameters, instance type, and instance settings (e.g., optimized storage or network). Once a range of configurations has been identified for profiling, users must then embark on the painstaking process of trial-and-error analysis of the application over a variety of instance types. This process involves configuring and installing the application and its dependencies, running a workload, measuring performance and resource utilization, and finally recording, analyzing, and presenting the results in a usable format.

To automate the profiling process, we have developed an offline profiling service that deploys and then monitors the execution of arbitrary applications over a collection of instance types.[3] Applications are provided as either a self-contained directory with the application executable and input data, or as a contextualization script that is used to install the application and its dependencies on the target instance(s). Users specify the desired range of invocation configurations and the instance types they wish to explore. The profiling service then provisions instances and manages the execution of the application for each of the specified combinations. The service deploys a lightweight monitoring tool, called Performance Co-Pilot (PCP), to measure and record resource usage over time. The service periodically collects PCP performance metrics from each instance while an application

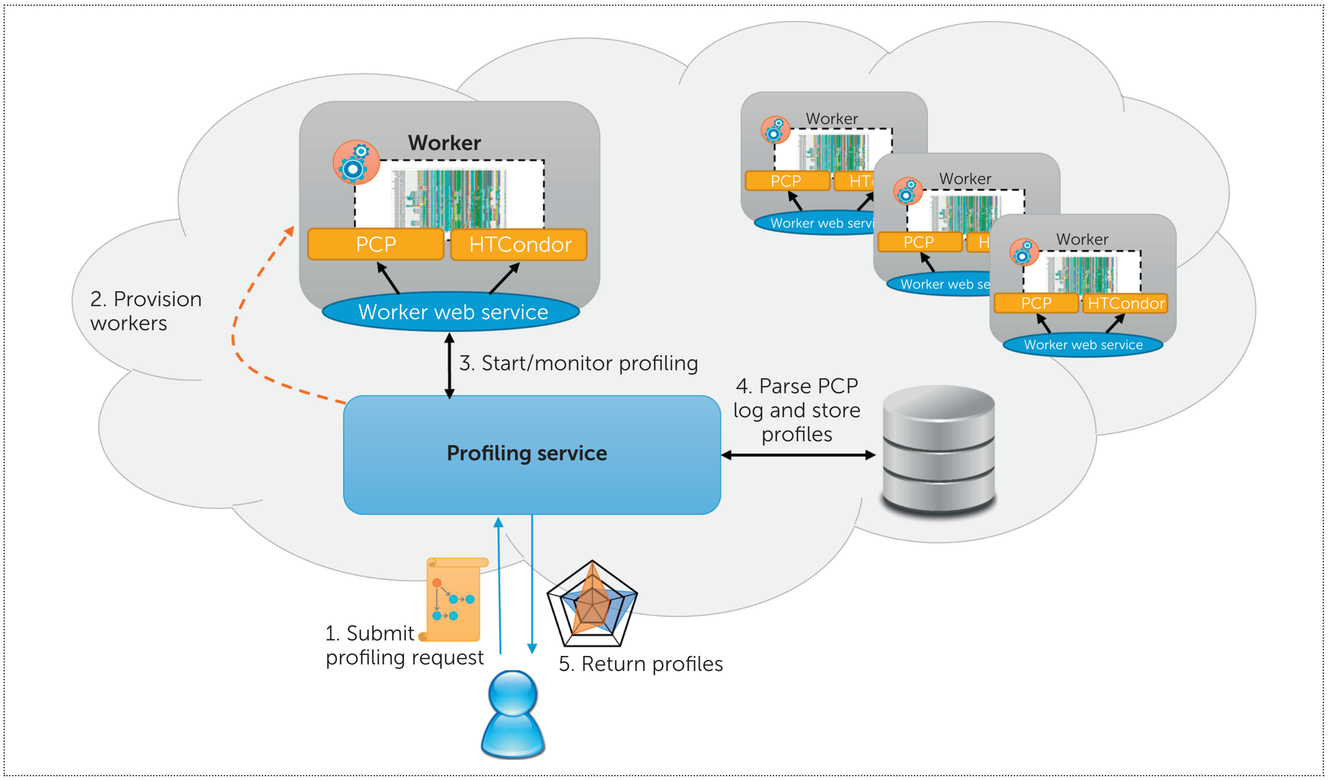
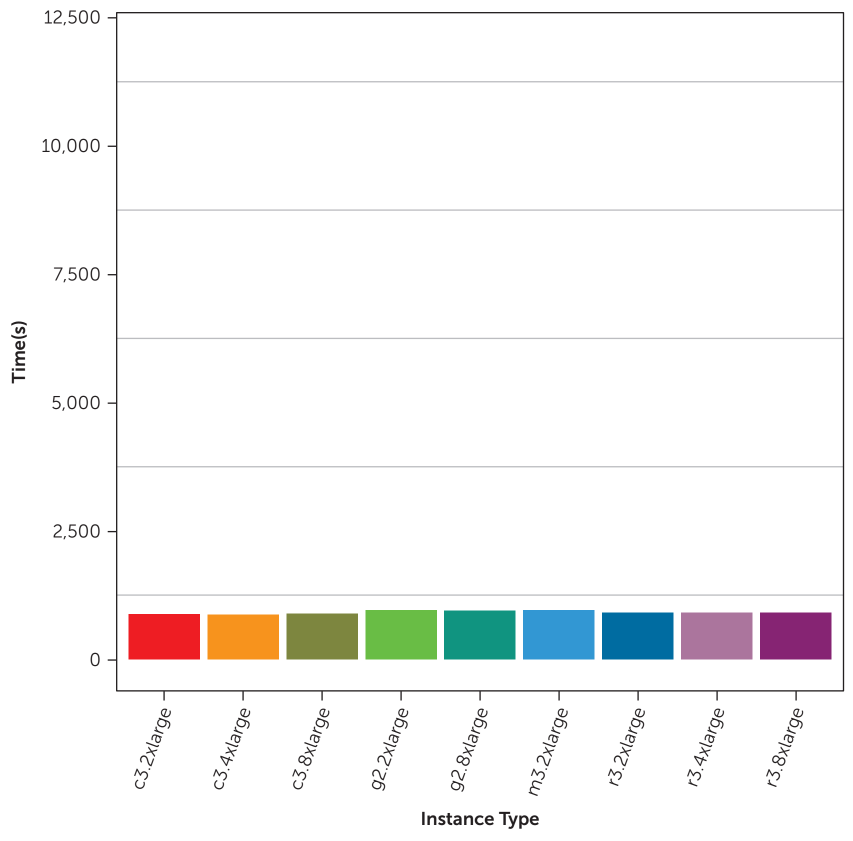


FIGURE 2. SCRIMP profiling service. Users submit the application to be profiled and a range of conditions to be explored. The service dynamically assembles the required testbed, installs the application, its dependencies, and execution and monitoring software. Throughout execution it records resource usage and stores this information in a database. Finally, an aggregate profile is assembled to summarize performance under the requested conditions.

executes. When the application finishes executing the resource usage information is aggregated and combined with high level measurements (e.g., peak resource usage, execution time) to derive a SCRIMP-compatible JavaScript Object Notation profile. This profile summarizes performance characteristics across instance types and under different configurations. Figure 2 outlines this service's architecture.

We have used the SCRIMP profiling service to profile a range of genome analysis applications and demonstrated that the profiles can be used to make provisioning decisions that reduce execution time by up to 15.7% and cost by up to 86.6% in a real-world scenario.[3] This specific group of applications were selected for in-depth profiling as they represent those most frequently executed across a subset of Globus Genomics gateways, thus the benefit of characterizing their execution requirements outweighs the overhead of offline profiling. We note that many applications and gateways exhibit long-tailed distributions of individual application executions.[3] In cases where jobs are executed infrequently, the overhead of such a profiling approach limits its viability. We are exploring approximate and online profiling models to reduce the overhead of profiling in these cases.

Figure 3 shows the execution time from two profiles for FastQC, a quality control application for raw sequence data, and Burrows-Wheeler Alignment Tool (BWA) ALN, an aligner application that aligns short sequences to a reference sequence. This figure highlights the vastly different performance observed for two different applications on different instance types. FastQC performance is almost identical between instance types as the application cannot take advantage of multiple cores. In contrast, BWA ALN shows instance types with related or similar performance characteristics. For example, it takes roughly the same time using the c3.2xlarge, g2.2xlarge, m3.2xlarge, and r3.2xlarge instance types. Execution time is also roughly related within particular instance families. For example, the execution time of c3.4xlarge is roughly half that of c3.2xlarge. It is these patterns that SCRIMP can use when provisioning infrastructure as it allows a more complete comparison of available instance types.



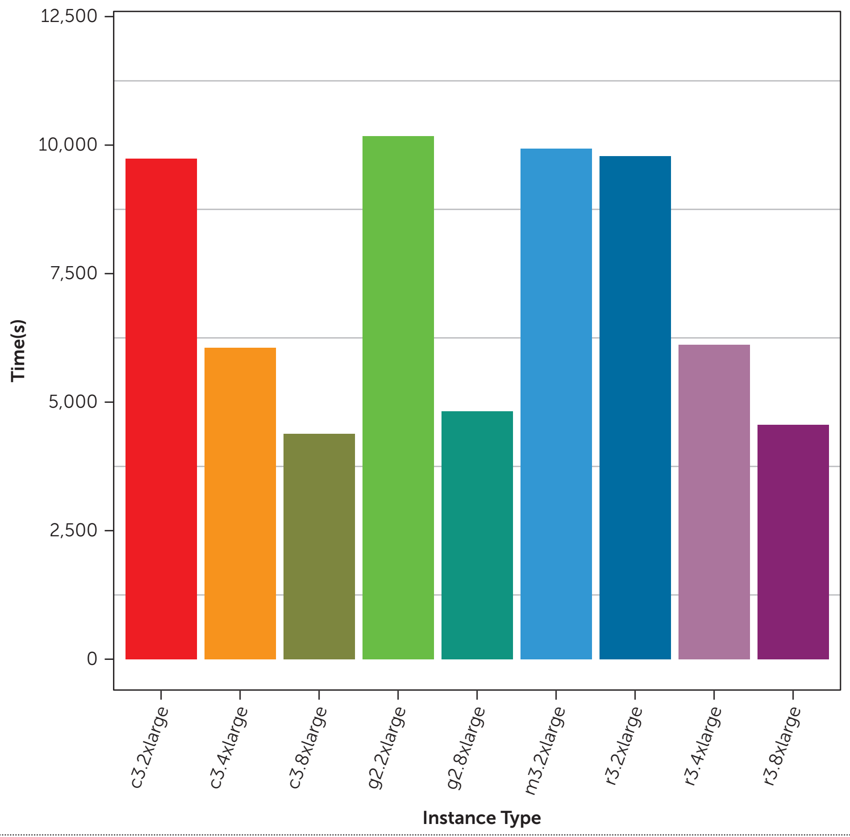


FIGURE 3. Execution time for FastQC and BWA ALN on various Amazon Web Services (AWS) instances.

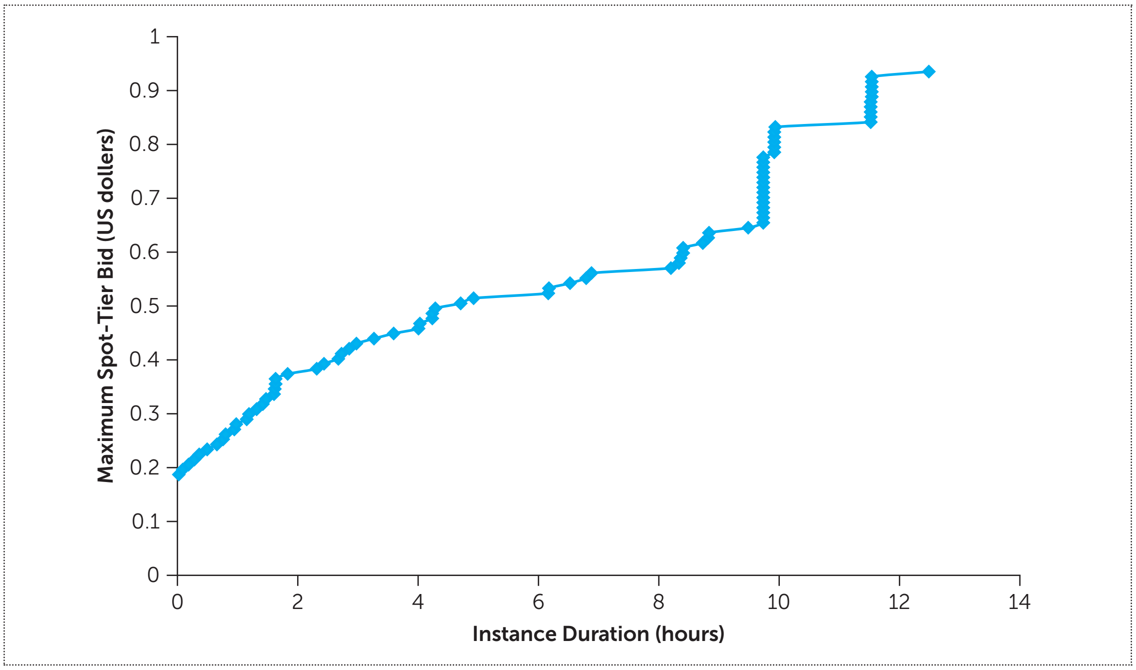


FIGURE 4. Durability Agreements from Time Series (DrAFTS) computed bids vs instance duration for a c3.4xlarge instance type in the us-east-1a availability zone.

Predicting Market Conditions and Establishing Probabilistic Guarantees

An important principle of cloud computing is that users are charged for the resources consumed. Cloud providers often set different prices for different levels of service. For example, they may offer reliable (in terms of an SLA) instances at a fixed, published price; long-term instances at a reduced rate; and volatile (or unreliable) instances (with best effort SLAs) at a further discount. AWS volatile instances are offered through a "spot market"—a dynamic market in which users are charged per hour based on a fluctuating market price. Users place a bid—the maximum price they are willing to pay for an instance. If their bid exceeds the market price they will obtain an instance, if the market price exceeds their bid price at any point in the future then the instance may be revoked.

Although these markets offer capacity at low cost, they come with inherent risk. Instances may be revoked with little to no warning, potentially wasting computation (if not checkpointed) and/or compromising the user's ability to complete a task before a deadline. Thus, it is important to understand the market in order to derive value from it. When considering a preemptible market where the amount bid for a resource is correlated with its durability, there are two major challenges: computing a bid that is high enough to reduce the likelihood of revocation and computing a bid that minimizes financial risk (i.e., bid as close to market price as possible). While the first goal can be achieved trivially by bidding the maximum price, this strategy is risky as users could be charged the bid price. (In AWS, the market price can significantly exceed the on-demand or posted price.) The second goal can be achieved by bidding the current market price, however there is no guarantee that resources will be provided for any length of time. Ideally, one would like to calculate a bid that is always greater than the market price, but not significantly so.

While there has been considerable research related to bid calculation in dynamic cloud markets few systems have been applied in practice. Furthermore, previous work on cloud market prediction has focused on predicting market conditions at a particular point in time, but not on the equally important goal of satisfying instance durability requirements.[4] To address these more general needs we have developed a forecasting system called Durability Agreements from Time Series (DrAFTS).[4] DrAFTS aims to calculate a (minimum) bid price that guarantees, probabilistically, that an instance will not be revoked before a specified deadline. The approach applies a two-step, nonparametric time-series forecasting technique to the pricing history. DrAFTS first computes a time series of upper bounds on market prices at each moment in a price history. These bounds are probabilistic guarantees that the next value in the market price series will be less than the bounds and, thus, could have served as a maximum bid at each point in time. It then repeatedly increments this bid fractionally and computes a series of time durations until the bid would have been equaled or exceeded by market price. It finally computes a probabilistic lower bound on this series of time durations. The process generates pairs of bids and lower-bounds on durations where each bid guarantees (at a minimum, probabilistically) the duration with which it is paired.

Figure 4 shows a representative result. Note how the computed bid price for a particular instance increases with durability. Such knowledge of required bid prices can enable a more balanced comparison between market models, for example enabling a fair comparison between on-demand instances (with explicit durability guarantees) and AWS spot instances (which have no durability guarantee). We have evaluated DrAFTS across AWS regions, availability zones, and instance types and shown that its predictions are accurate, within specified confidence levels, and the guarantee is rigorous assuming the spot market time series is ergodic.[4] DrAFTS is designed to be applied to the AWS spot market as, at present, it is the only dynamic market model for preemptible resources. We have explored DrAFTS' ability to reduce financial risk and minimize spot instance terminations by having SCRIMP fulfill a representative workload of 1,000 Globus Genomics jobs. When combining DrAFTS with an application's estimated execution duration, derived from application profiles, we found a 27.27% reduction in the number of instances terminated prematurely due to the bid price being exceeded.

Provisioning and Managing Instances in a Multicloud Environment

The final component of our solution is the flexible SCRIMP provisioning service that relies on application profiles and market prediction to elastically provision compute instances to meet the needs of dynamic workloads. While several auto-scaling systems have been developed, including those targeting multicloud environments, these systems typically focus on adding and removing homogeneous instances for a single application and by using simple policies (e.g., add a resource if the queue length exceeds x, bid $0.45 for a c3.8xlarge instance).[13,14] Thus, they are incapable of being leveraged by other applications and services, and they do not enable complex tradeoffs to be evaluated or managed when provisioning instances.

The SCRIMP provisioning lifecycle is comprised of the following steps: assessing awaiting work to determine if resources are required; determining the most suitable instance type to be provisioned for the specific job; selecting an instance based on cost (including calculating costs for dynamic markets); provisioning and configuring the resources for execution (e.g., installing execution software); and monitoring the running job to respond to different situations (e.g., migrating/resubmitting jobs when instances fail or are revoked). SCRIMP provides these capabilities, enabling users to outsource the entire provisioning lifecycle while also being sufficiently flexible to meet the needs of a wide variety of use cases. When a job is designated for execution (e.g., when added to a monitored queue or after some period of waiting), SCRIMP will filter the list of available instance types by those that can support the given application (via profiles). It will then determine approximate resource requirements (via profiles) and compute a set of candidate instance types alongside a cost function (using market prediction models to provide a cost for spot instances) for each instance type. Note: the candidate instance types may in fact be from different cloud providers.

Based upon user policies (e.g., preferences for on-demand instances, or specific instance types), SCRIMP will prioritize instance types and enter the provisioning phase. SCRIMP requests instances using the selected cloud provider's API, it monitors these requests and when satisfied will deploy required software using user-supplied contextualization scripts. Finally, it will record the provisioned instance and monitor the execution of the job until completion. SCRIMP leverages standard cloud-init configuration scripts that ensure instances are appropriately configured regardless of the cloud provider used.

SCRIMP employs several strategies to optimize the provisioning process including overbidding, managing billing cycles, instance sharing, and reverting to on-demand instances. The overbidding strategy is used to request more instances than are needed with the knowledge that instance fulfillment can be unpredictable. This is particularly valuable when using spot instances, as spot instance requests can take several minutes to be satisfied, and in some cases, will not be satisfied at all. It is also useful in situations when particular cloud providers are offline or experiencing service degradation. Note: when an instance is launched excess instance requests (from overbidding) are terminated (or repurposed) by SCRIMP before they are fulfilled. Managing billing cycles aims to understand

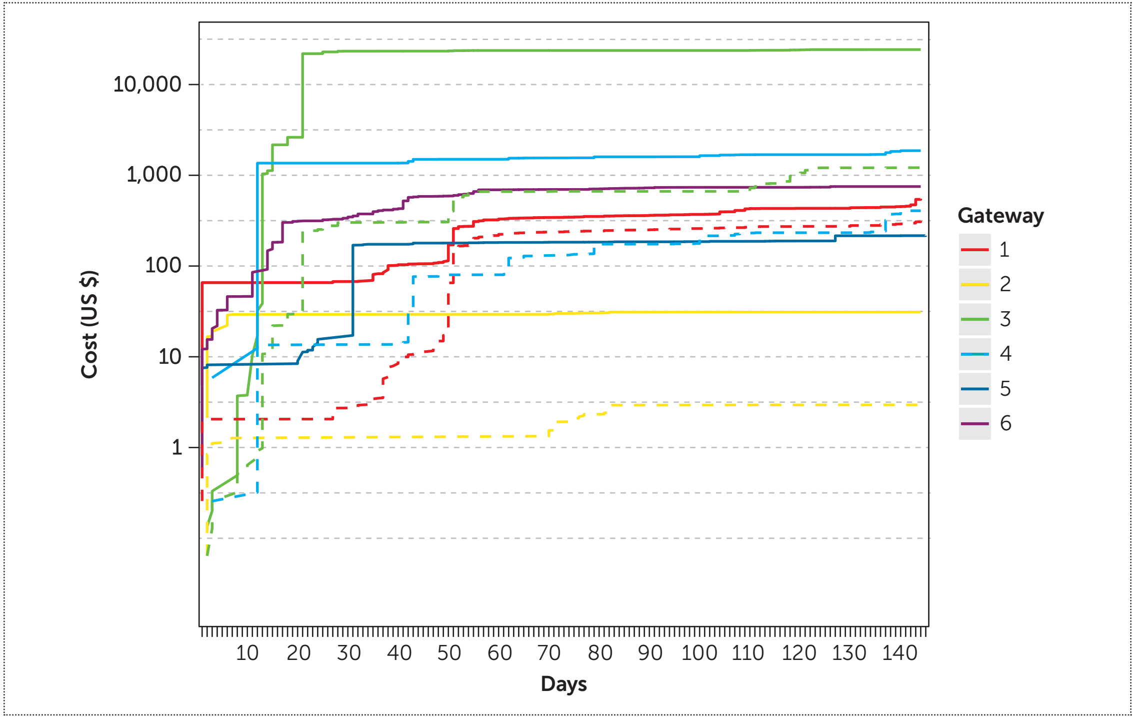


FIGURE 5. The cumulative cost of a simple provisioning policy which uses a single instance type in a single availability zone policy (solid line) vs using multiple instance types and availability zones (dashed line) for six Globus Genomics gateways over a 145-day period on a logarithmic scale.[5]

the "increments" (often hourly) in which cloud instances are billed. If an application completes execution before a billing cycle has elapsed the remaining time can be used without additional cost. SCRIMP uses this knowledge to enable instances to be reused and to terminate unused instances only as they approach their billing cycle. Unused instances are subsequently considered in the provisioning process for new jobs by including the provisioned instance with a cost of 0 for the remaining period (longer durations are modeled by adding predicted costs for that instance type). The instance sharing strategy enables multiple applications to be executed on the same instance. This can save considerable time and cost as there is no need to provision new instances and execution may begin immediately. It is however only possible if SCRIMP is able to determine that two applications can run concurrently on a single instance (e.g., via profiles). Finally, reverting to on-demand (rather than spot) instances aims to prioritize execution time over cost when demand for spot instances is high it will instead use more expensive on-demand instances.

To explore the value of automated provisioning approaches we integrated a prototype version of SCRIMP in the Globus Galaxies platform.[5] While this version did not leverage market prediction models and only used binary profiles that indicated if an application could execute on a particular instance type, the resulting improvements over simple provisioning approaches were pronounced. To analyze these advantages, we recorded production workloads from six Globus Genomics gateways over a 145-day period. Prior to using SCRIMP these gateways were preconfigured with a basic provisioning policy that acquired a single spot instance type (e.g., c3.8xlarge) with a fixed bid price (approximately 80% of the on-demand cost). We then re-executed each workload using SCRIMP where multiple instance types and availability zones were considered. We again used spot instance types and set the bid price to the same fixed value. Figure 5 shows the results when SCRIMP is employed over the baseline approach. In total, SCRIMP was able to reduce costs from over $27,000 to approximately $2,250 for this period, resulting in 92.1% reduction in cost. On average, individual gateways obtained savings of 75.9% (Max 95.0%, Min 43.0%).

Summary

Leveraging multiple cloud providers can reduce costs, improve efficiency, enhance reliability, and eliminate vendor lock in. While these advantages are clearly desirable, the enormous flexibility afforded by many competing, heterogeneous providers also creates new challenges to make use of a multicloud environment. SCRIMP aims to address several of these challenges to reduce the barriers for applications and services that wish to elastically scale workloads over heterogeneous cloud providers. SCRIMP's use of automatically derived application profiles combined with predicted market conditions allow it to efficiently provision infrastructure while also enabling cost, time, and reliability tradeoffs to be managed.

While SCRIMP has been shown to significantly reduce costs, execution time, and instance failures there is yet more that can be done to improve performance. SCRIMP's profiling service can be extended to reduce the overhead of profiling and therefore make the approach more suitable for long-tail job executions. Specifically, we are exploring methods to learn from related profiles and instance types to reduce the search space as well as analyzing features of applications, instance types, and cloud providers to derive powerful predictive classes. From a market prediction perspective, there is an increasing variety of market mechanisms used by providers, and researchers are developing new models to understand and predict these markets. SCRIMP's modular architecture is designed such that it can be extended to incorporate new acquisition and market prediction models as needed. Finally, we are actively working to extend SCRIMP's provisioning components to support a descriptive policy language. This will enable more flexible, user-oriented decision making to meet the needs of more complex use cases.

Acknowledgments

This work was supported in part by the NSF grant ACI-1541215 The Aristotle Cloud Federation and the NIH under contract 1U54EB020406-01 Big Data for Discovery Science Center. The authors acknowledge generous research credits provided by Amazon Web Services that helped support this work.

References

1. D. Lifka et al., "XSEDE Cloud Survey Report," tech. report, National Science Foundation, 2013.
2. S. Yi, A. Andrzejak, and D. Kondo, "Monetary Cost-Aware Checkpointing and Migration on Amazon Cloud Spot Instances," *IEEE Trans. Services Computing*, vol. 5, no. 4, 2012, pp. 512–524.
3. R. Chard et al., "An Automated Tool Profiling Service for the Cloud," *Proc. 16th IEEE/ACM Int'l Symp. Cluster, Cloud and Grid Computing* (CCGrid), May 2016, pp. 223–232.
4. R. Wolski et al., "Probabilistic Guarantees of Execution Duration for Amazon Spot Instances," to be published in *Int'l Conf. High Performance Computing, Networking, Storage and Analysis*, 2017.
5. R. Chard et al., "Cost-Aware Cloud Provisioning," *11th IEEE Int'l Conf. e-Science*, Aug. 2015, pp. 136–144.
6. R. Madduri et al., "The Globus Galaxies Platform: Delivering Science Gateways as a Service," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 16, 2015, pp. 4344–4360.
7. Y. Babuji et al., "Cloud Kotta: Enabling Secure and Scalable Data Analytics in the Cloud," *IEEE Int'l Conf. Big Data*, 2016.
8. K. Chard, S. Tuecke, and I. Foster, "Efficient and Secure Transfer, Synchronization, and Sharing of Big Data," *IEEE Cloud Computing*, vol. 1, no. 3, 2014, pp. 46–55.
9. A. Barker, B. Varghese, and L. Thai, "Cloud Services Brokerage: A Survey and Research Roadmap," *IEEE 8th Int'l Conf. Cloud Computing* (CLOUD), 2015, pp. 1029–1032.
10. F. Samreen et al., "Daleel: Simplifying Cloud Instance Selection Using Machine Learning," *IEEE/IFIP Network Operations and Management Symposium* (NOMS), 2016, pp. 557–563.
11. C. Bailey Lee et al., "Are User Runtime Estimates Inherently Inaccurate?" *10th Int'l Conf. Job Scheduling Strategies for Parallel Processing* (JSSPP'04), 2005, pp. 253–263.
12. M. Calzarossa and G. Serazzi, "Workload Characterization: A Survey," *Proceedings of the IEEE*, vol. 81, no. 8, 1993, pp. 1136–1150.
13. J. Tordsson et al., "Cloud Brokering Mechanisms for Optimized Placement of Virtual Machines Across Multiple Providers," *Future Generation Computer Systems*, vol. 28, no. 2, 2012, pp. 358–367.
14. T. Lorido-Botran, J. Miguel-Alonso, and J.A. Lozano, "A Review of Auto-Scaling Techniques for Elastic Applications in Cloud Environments," *J. Grid Computing*, vol. 12, no. 4, 2014, pp. 559–592.

RYAN CHARD *is a postdoctoral fellow and recipient of the Maria Goeppert Mayer Fellowship at Argonne National Laboratory. His research interests include distributed systems, cloud computing, software-*

defined cyberinfrastructure, and reputation systems. Contact him at rchard@anl.gov.

KYLE CHARD *is a senior researcher and fellow at the Computation Institute, a joint institute of the University of Chicago and Argonne National Laboratory. His research interests include distributed metascheduling, grid and cloud computing, economic resource allocation, social computing, and services computing. Contact him at chard@uchicago.edu.*

RICH WOLSKI *is a professor of computer science at UC Santa Barbra and cofounder of Eucalyptus Systems Inc. Having received his MS and PhD degrees in CS from UC Davis (while a research Scientist at Lawrence Livermore National Laboratory), he has also held positions at UC San Diego, the University of Tennessee, the San Diego Supercomputer Center, and Lawrence Berkeley National Laboratory. Rich has led several national scale research efforts in the area of distributed systems, coleads the UCSB SmartFarm project, and is the progenitor of the Eucalyptus open source cloud project. Rich is an IEEE fellow and his work has been recognized with an NSF CAREER award, a UCSB Distinguished Teaching Award, with selection as a Cloud Computing Pioneer by Information Week. Contact him at rich@cs.ucsb.edu.*

RAVI MADDURI *is a computational scientist in the Mathematics and Computer Science division at Argonne National Laboratory and a senior fellow at the Computation Institute, University of Chicago. His research interests are in large-scale computation and data management. He leads the Globus Genomics project (www.globus.org/genomics), which is widely used for genomics, proteomics, and other biomedical computations on Amazon cloud and other platforms. Contact him at madduri@mcs.anl.gov.*

BRYAN NG *studied network engineering and statistics at University Malaya. He is a lecturer at Victoria University. His research interests include mathematical modeling and performance analysis of networks and protocols. Prior to joining Victoria, he worked in Orange Labs, INRIA, and AC Nielsen consulting. Contact him at bryan.ng@ecs.vuw.ac.nz.*

KRIS BUBENDORFER *is an associate professor in network engineering at Victoria University of Wellington. He received his PhD in computer science, on a mobile agent middleware, from the Victoria University of Wellington in 2002. His current research interests include distributed and cloud computing, social computing, digital provenance and reputation. He teaches courses in networking, operating systems, security and distributed systems. Contact him at kris@ecs.vuw.ac.nz.*

IAN FOSTER *is an Argonne senior scientist and distinguished fellow and the Arthur Holly Compton Distinguished Service Professor of Computer Science. His research deals with distributed, parallel, and data-intensive computing technologies, and innovative applications of those technologies to scientific problems in such domains as climate change and biomedicine. Contact him at foster@anl.gov.*