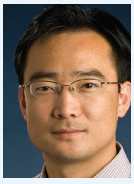
Cross-Layer Cloud Resource Configuration Selection in the Big Data Era

Rajiv Ranjan
Commonwealth Scientific and Industrial Research Organization

Joanna Kołodziej
Cracow University of Technology



Lizhe Wang
China University of Geosciences

Albert Y. Zomaya
University of Sydney

The emergence of cloud computing has facilitated resource sharing beyond organizational boundaries and among various applications. This cloud resource sharing is primarily driven by resource virtualization and utility computing (the pay-as-you-go pricing model). The generic multilayered cloud service model is appealing to many parties—from small businesses looking for a low upfront infrastructure investment, to enterprises wanting to cut the cost of managing infrastructures, to research communities requiring large-scale data processing and computing power. In a cloud environment, computing resources (processors, storage devices, network bandwidth, and so on) and applications are provided as services over the Internet.

Fueled by an insatiable demand for new Internet services and a shift to cloud computing services that are largely hosted in commercial datacenters and in the large data farms operated by companies like Amazon, Apple, Google, Microsoft, and Facebook, discussions increasingly focus on the need to ensure application performance under various uncertainties. Through the infrastructure-as-a-service (IaaS) and platform-as-a-service (PaaS) concepts, datacenters virtualize their hardware and software resources and rent it on demand. In the cloud computing approach, multiple datacenter applications (such as content delivery networks, multitier Web, big data analytics, and large-scale scientific simulations) are hosted on a common set of servers. This allows for consolidation of application workloads on a smaller number of servers that can be better utilized, because different workloads might have different resource utilization footprints as well as temporal variations.

Multiple providers in the current cloud landscape offer IaaS and PaaS resources under heterogeneous configurations. Hence, application owners face a daunting task when trying to select cloud services that can meet their constraints. According to recent estimations, there are hundreds of IaaS providers around the world. Even within a particular provider there are different variations of services. For example, Amazon Web Services

 2325-6095/15/$31.00 © 2015 IEEE

(AWS) has 674 offerings differentiated by price, quality of service (QoS), and service-level agreement (SLA) features and location.[1] Further, every quarter they add about four new services, change business models (price and terms), and sometimes even add new locations. To select the best mix of IaaS and PaaS resource configurations from an abundance of possibilities, application owners must simultaneously consider and optimize application-level performance SLAs while dealing with complex software and hardware configuration dependencies; and managing heterogeneous sets of configurations (price, hardware features, software framework features, location, performance, and so on) remains a hard challenge.[2–4] For instance, it's not enough to just select an optimal cloud storage resource configuration for a content-delivery network application. Allocating corresponding CPU configurations (for example, type, speed, and cores) to content-delivery network media servers and encoding/decoding servers is essential to guaranteeing the ability to serve the content as fast as possible across a variety of devices.

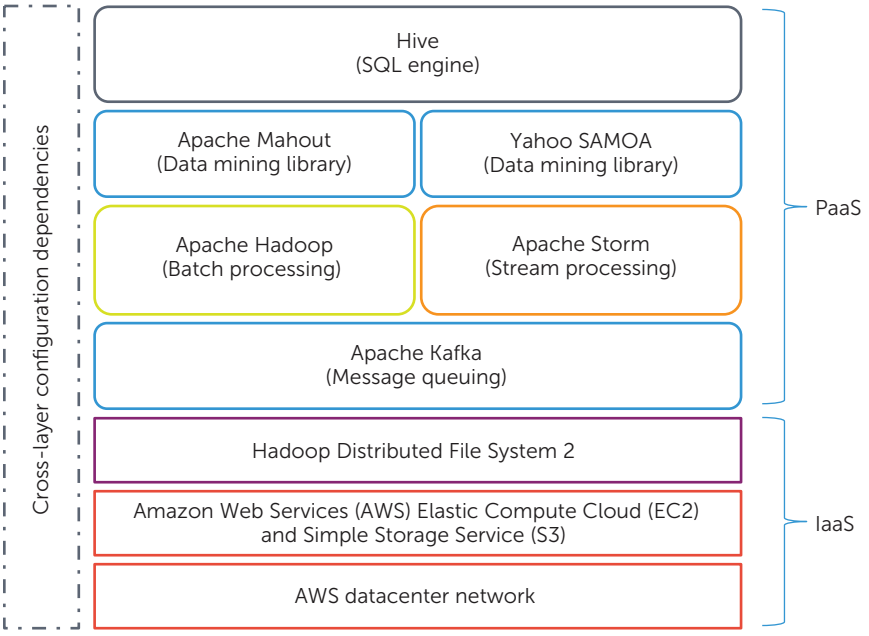


FIGURE 1. A stock recommendation application deployed over an Amazon Web Services datacenter.

Cross-Layer Resource Configuration Selection

To simplify understanding of the cross-layer resource configuration selection problem, consider a social-network-driven stock recommendation big data application deployed on an AWS datacenter, as illustrated in Figure 1.

This application needs to process both historical and real-time data, hence its application architecture consists of multiple and heterogeneous big data processing frameworks. Therefore, the application combines streaming free-form text data from the Twitter API with historical tweets (available via Twitter Firehose) stored in Amazon Simple Storage Service (S3) hardware resources. In the example in Figure 1,

- Apache Kafka is deployed as a high-throughput message-queuing framework;
- Apache Storm is deployed as a stream-processing framework that in turn exploits Yahoo Scalable Advanced Massive Online Analysis (SAMOA) as a data mining framework for classifying groups of tweets relevant to a particular stock;
- Apache Hadoop is deployed for processing historical tweets;
- Apache Mahout, which is hosted within the Apache Hadoop runtime environment, implements a Bayesian classifier algorithm for tweet grouping and classification; and
- the output of both batch and stream analytics subsystems is written to the Hadoop Distributed File System (HDFS).

To query the analytics result (for example, the top K most promising stock portfolios), Apache Hive is deployed to support search queries in Standard Query Language (SQL) format.

As Figure 1 shows, there are two application management layers in a big data application platform.[5,6] The first is a big data processing or PaaS framework (Apache Hadoop, Apache Storm, Apache Mahout, and so on) layer that implements software-based data processing primitives (for example, batch processing by Apache Hadoop or stream processing by Apache Storm). In the second IaaS layer, cloud-based hardware or IaaS resources (for example, CPU, storage, and network) provide hardware resource capacity to the higher-level PaaS frameworks.

The hard challenge is determining the optimal approach to automatically select IaaS resource and big data processing framework configurations

such that the anticipated application-level performance SLA constraints (for example, minimize event-detection and decision-making delays, maximize application and data availability, and maximize number of alerts sent per second) are consistently achieved, while maximizing cloud datacenter CPU utilization, CPU throughput, network throughput, storage throughput, and energy efficiency. For example, in the configuration in Figure 1, there's a need to optimally select configurations for Apache Hadoop (number of map and reduce tasks, map and reduce slots per CPU, max RAM per slot, and so on) and AWS CPU resource configurations (I/O capacity, RAM, CPU speed, local storage, cost, and so on) driven by application-level performance SLA constraints (for example, analyze 100 Gbytes of stock purchase tweets in x minutes subject to a maximum budget of y dollars). A similar cross-layer configuration challenge exists for other big data processing frameworks, such as Apache Storm and Apache Hive. In summary, managing such layered SLA dependencies across multiple big data processing frameworks is widely recognized as a challenging problem.[6–9]

Some cloud computing datacenter providers are also offering software-defined networks (SDNs) at the IaaS layer for deploying big data processing frameworks. In general, SDN deployments implement virtual switches and the software-level OpenFlow protocol to realize communication between the controller and forwarding devices. In SDN-enabled datacenters, the configuration selection and placement of big data processing frameworks in virtual CPUs needs to be coordinated with the selection of network routes between physical servers.

Existing big data application orchestration platforms (Apache YARN, Mesos, and Apache Spark) are designed for homogeneous clusters of IaaS resources. These platforms expect application administrators to determine the number and configuration of allocated IaaS resource types and provide appropriate configuration parameters for each IaaS resource type and big data processing framework for running their analytics tasks. Branded price calculators, available from public cloud providers such as AWS (http://calculator.s3.amazonaws.com/index.html) and Azure (http://www.windowsazure.com/en-us/pricing/calculator) and academic projects such as Cloudorado (www.cloudorado.com), allow comparison of IaaS resource leasing costs. However, these calculators can't recommend or compare configurations across big data processing frameworks and hardware resources while ensuring application-level SLAs (such as minimizing event-detection delay).

> Allocating IaaS-level cloud resources to PaaS-level big data processing frameworks is no longer a traditional time-minimization or resource-maximization problem.

Overview of Multicriteria Optimization and Decision-Making Approaches

The vast configuration diversity among the available cloud resources and big data processing frameworks makes it difficult for application administrators to select configurations or even determine a valid background for their decisions. Consequently, allocating IaaS-level cloud resources to PaaS-level big data processing frameworks is no longer a traditional time-minimization or resource-maximization problem but involves additional simultaneous objectives and configuration dependencies across multiple IaaS resources and big data processing frameworks. These include, but aren't limited to,

- maximizing classification accuracy for Apache Mahout,
- minimizing response time for map and reduce tasks in Apache Hadoop,
- minimizing stream processing latency in Apache Storm,
- maximizing network throughput for HDFS,
- maximizing CPU resource utilization, and
- minimizing energy consumption for the datacenter.

The ever-increasing heterogeneity of hardware being deployed in datacenters comprising multicore processors and co-processors—general-purpose computing on graphics processing units (GPGPU), IntelMIC, and so on—further complicates the decision-making problem. Clearly, selecting configurations at the IaaS and PaaS layers for big data applications is a multiobjective optimization problem that doesn't have a single solution, but rather a set of tradeoff solutions (known as a Pareto front) corresponding to the SLA objective functions of each

big data processing framework (see Figure 1). Given that big data processing frameworks (such as Apache YARN and Apache Storm) share the same cluster of virtualized resources in a datacenter, workload contention among the frameworks further complicates computing the optimal configuration for meeting performance SLAs. For some big data workloads (high-volume batch processing of historical tweets by Hadoop in Figure 1), storage requirements dominate, whereas for others (transactional query processing by Apache Hive in Figure 1), computational requirements dominate, and for still others (real-time Twitter stream processing by Apache Storm in Figure 1), communication requirements dominate. Hence, in such complex application deployment scenarios, configuration selection techniques must have intelligence to help them determine which workloads should be combined on a common physical server (hosting multiple virtualized CPUs) to minimize resource contention due to workload interference. Such contention-related intelligence can be obtained via offline benchmarking as well as real-time SLA monitoring techniques—a very challenging problem.

A decision problem typically involves balancing multiple, and often conflicting, objective functions and constraints. Numerous research publications present different techniques to solve the configuration selection problem. Each technique is characterized by its search computational efficiency and flexibility in handling the diverse set of objectives and criteria. We classify these methods into three categories: constraint optimization, multicriteria evolutionary optimization (MCEO), and multicriteria decision analysis (MCDA).

Constraint Optimization

Constraint-optimization techniques such as linear programming can efficiently solve configuration selection problems where both the objective and the constraints are linear with respect to all decision variables.[10] Problem instances involving nonlinear objectives and constraints can be solved by applying techniques such as integer programming. However, in practice, neither of these techniques can handle the multiple conflicting objective functions (such as minimizing energy consumption while maximizing CPU utilization and HDFS network throughput).

To handle multiple objective functions, researchers have developed goal-programming techniques that transform the multiobjective problem into a single-objective optimization problem by assigning weights to objectives aggregated in an analytical function.[10] These techniques also support a combination of soft and hard (nongoal) constraints that can deviate, allowing for tradeoffs in achieving satisfactory solutions rather than focusing only on optimal solutions. Unfortunately, goal programming also has shortcomings; for example, the weights are problem-dependent and need to be decided a priori via methods such as application benchmarking, and the weights can lead to undesirable solutions if the relationships between the objective functions aren't clearly understood.

Multicriteria Evolutionary Optimization

MCEO techniques are capable of modeling and optimizing several objective functions simultaneously, and can find global optimal solutions.[11] This class of techniques includes various well-known biologically and physically inspired metaheuristics such as simulated annealing, genetic algorithms, particle swarm optimization, ant and bee colony optimization, and Tabu search. These techniques can handle search over an infinite number of feasible alternatives constrained by a finite number of quantitative configuration criteria. Unfortunately, because MCEO techniques are unconstrained, they can lead to high computational complexity. Constraining these techniques' running time requires integrating penalty functions with objective functions during the optimization process. With the evolution of parallel and distributed big data processing frameworks such as Apache Hadoop, it's possible to speed up these techniques via massive parallelization. However, novel research will be required in terms of modeling and implementation of a parallel version of these techniques that incorporates multiple cross-layered SLA objective functions (see Figure 1).

Multicriteria Decision Analysis

MCDA identifies combinations of configurations for frameworks and resources at the PaaS (such as number of map and reduce tasks instances in Apache Hadoop) and IaaS (such as CPU type and speed for hosting instances of map and reduce tasks in Apache Hadoop) layers to achieve application-level performance SLA objectives (such as minimizing data analytics delay). Formally, MCDA consists of "a collection of formal approaches which seek to take explicit account of multiple criteria in helping individuals or groups explore decisions that matter."[12] In general, the cross-layer configuration selection problem is MCDA, which can be briefly defined as a collection of techniques for providing the comparative analysis, ranking, and selection of the alternate combination of configurations[12] that best meets the application-level SLA objectives. Such a combination of configurations can be selected from a finite (known a priori) or very large/infinite (unknown a priori) set of all possible alternatives. A particular

alternative's "usefulness" is expressed by the application-level SLA objective function, the values of which depend on the payoff or decision matrix (static or dynamic) generated for the whole process.

MCDA techniques can be broadly classified as analytic hierarchy process (AHP), multiattribute utility theory (MAUT), and outranking methods.[1,12,13]

AHP is based on a pairwise comparison, with the criteria (cheapest CPU, cheapest storage, fastest map/reduce tasks, and so on) structured in a multilevel hierarchal relationship. The objective is defined at the top level, and the lower levels correspond to super-criteria, subcriteria, and so on. In the AHP tree, the selection process starts at the leaf criterion and progresses toward the top-level goal (final objective function). Each level represents the selection hierarchy corresponding to the weight or influence of different branches originating at that level. The analytic network process (ANP) is an extension of AHP that can be applied to solve decision-making problems that can't be structured hierarchically.

The analytic network process (ANP) is an extension of AHP that can be applied to solve decision-making problems that can't be structured hierarchically.

Unlike AHP, MAUT techniques are based on utility functions that quantify decision makers' preferences. MAUT aims to generate a means of associating a real number with each alternative (solution) to produce a preference order of alternatives consistent with the decision maker's opinion. The attribute values aren't fully determined in the alternative selection process, but can be influenced by some random factors. The consequences of IaaS and PaaS configuration selection should therefore be defined as probability vectors. MAUT techniques combine various preferences in the form of multiattribute utility functions for each criteria, which are combined with attribute weighted functions. The advantage of using MAUT is that the problem is constructed as a single objective function after successful assessment of the utility function. Thus, it becomes easy to ensure the achievement of the best compromise solution based on the higher-level objective function.

Outranking techniques are applied directly to partial preference functions, which are assumed to have been defined for each criterion. These preference functions could correspond to natural attributes on a cardinal scale, or could be constructed as ordinal scales. In this case, the preference functions must satisfy only the ordinal preferential independence condition. The key difference between MAUT and outranking techniques is that MAUT selects the best choice whereas outranking produces a list of alternatives.

Jose Figueira and his colleagues[13] apply AHP to evaluate and compare general features (security, performance, scalability, and so on) of cloud datacenter providers as defined by the Cloud Services Measurement Initiative Consortium.[14] Some authors have used AHP to develop approaches to select and rank SaaS applications such as salesforce automation products.[15] A hybrid decision-making technique proposed elsewhere combines multicriteria decision making (AHP) and evolutionary optimization techniques (genetic algorithms) for selecting the best CPU and webserver images relevant to public clouds (such as AWS).[2] Another approach[16] applies MAUT-based techniques to select SaaS applications driven by trustworthiness.[17] The selection criteria includes quality, cost, and reputation. Using the ELECTRE (elimination et choix traduisant la realité) outranking technique, other authors propose a cloud datacenter selection approach based on general features (not relevant to any application type).[18]

However, to the best of our knowledge, existing approaches based on constraint optimization, MCEO, and MCDA techniques can be used to select configurations of multiple big data processing frameworks and IaaS resources (CPU, storage, and SDN networks) simultaneously while handling cross-layer SLA dependencies.

Recent Efforts

Although we consider a stock recommendation application here, the challenges are relevant to other big data application types as well. These applications include natural hazard management, credit card fraud detection, remote healthcare, and smart energy grids.

Although interest in deploying big data applications on clouds is growing, the set of concepts needed to understand the decision-making problem across multiple layers is still emerging, rather than being well defined or understood. Re-

cent efforts have attempted to automate the configuration selection of Hadoop frameworks over heterogeneous cluster resources. Gunho Lee and his colleagues proposed dynamically allocating heterogeneous cluster resources to a Hadoop framework based on single-performance SLA constraints on storage size configuration.[19] Karthik Kambatla and his colleagues proposed selecting the optimal Hadoop configuration parameters over a given set of cluster resources by developing and profiling resource consumption statistics.[20] Similarly, others proposed selecting configurations of Hadoop frameworks and heterogeneous Amazon EC2 CPU resources under various what-if scenarios (number of map and reduce tasks, size, and distribution of input data).[21] In the Aroma system, the configuration of CPUs is specified, then Hadoop framework is implemented to meet data processing deadlines while minimizing CPU rental cost.[22]

Progress in optimized configuration selection for Hadoop frameworks is significant and sets a foundation for future research, which must focus on developing holistic decision-making frameworks that automate configuration selection across multiple IaaS resource types and big data processing frameworks to ensure application-level SLAs as required in many emerging application domains.

Developing cross-layer configuration selection techniques is difficult. The space of possible configurations for big data processing framework and IaaS resources grows exponentially with the increasing number of big data processing framework types and cloud datacenter IaaS resource types. Computing optimal solutions is time consuming, and is therefore intractable given current technology. The hard challenge will be to identify the most relevant configurations for each big data processing framework and its dependencies on lower-level IaaS resource configurations. More complexities exist in modeling the objectives and criteria for individual big data processing frameworks and simultaneously computing configuration alternatives at design and run time in response to changes in data volume, velocity, variety, and query types.

References

1. M. Whaiduzzaman et al., "Cloud Service Selection Using Multicriteria Decision Analysis," *Scientific World J.*, vol. 2014, 2014, article no. 459375; doi: 10.1155/2014/459375.
2. M. Menzel et al., "CloudGenius: A Hybrid Decision Support Method for Automating the Migration of Web Application Clusters to Public Clouds," *IEEE Trans. Computers*, vol. 64, no. 5, 2015, pp. 1336–1348.
3. M. Zhang et al., "A Cloud Infrastructure Service Recommendation Technique for Optimizing Real-Time QoS Provisioning Constraints," to be published in *IEEE Systems J.* in 2015.
4. R. Ranjan, "The Cloud Interoperability Challenge," *IEEE Cloud Computing*, vol. 1, no. 2, 2014, pp. 20–24.
5. T. Shah, F. Rabhi, and P. Ray, "Investigating an Ontology-Based Approach for Big Data Analysis of Inter-Dependent Medical and Oral Health Conditions," *J. Cluster Computing*, vol. 18, no. 1, 2015, pp. 351–367.
6. D. Abadi et al., "The Beckman Report on Database Research," *ACM SIGMOD Record*, vol. 43, no. 3, 2014, pp. 61–70.
7. R. Ranjan, "Streaming Big Data Processing in Datacenter Clouds," *IEEE Cloud Computing*, vol. 1, no. 1, 2014, pp. 73–83.
8. M. Kunjir, P. Kalmegh, and S. Babu, "Thoth: Towards Managing a Multi-System Cluster," *Proc. Very Large Databases*, vol. 7, no. 12, 2014, pp. 1689–1692.
9. R. Zhang et al., "Getting Your Big Data Priorities Straight: A Demonstration of Priority-Based QoS Using Social-Network-Driven Stock Recommendation," *Proc. Very Large Databases*, vol. 7, no. 12, 2014, pp. 1665–1668.
10. G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley-Interscience, 1988.
11. M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics*, Int'l Series in Operations Research & Management Science 146, Springer; doi: 10.1007/978-1-4419-1665-5.
12. U. Habiba and S. Asghar, "A Survey on Multi-Criteria Decision Making Approaches," *Proc. Int'l Conf. Emerging Technologies* (ICET 09), 2009, pp. 321–325.
13. J. Figueira, S. Greco, and M. Ehrgott, eds., *Multiple Criteria Decision Analysis: State of the Art Surveys*, Int'l Series in Operations Research & Management Science 78, Springer, 2005; doi:10.1007/b100605.
14. S.K. Garg, S. Versteeg, and R. Buyya, "A Framework for Ranking of Cloud Computing Services," *Future Generation Computer Systems*, vol. 29, no. 4, 2013, pp. 1012–1023.
15. J. Siegel and J. Perdue, "Cloud Services Measures for Global Use: The Service Management Index (SMI)," *Proc. Ann. SRII Global Conf.*, 2012, pp. 411–415.
16. C.W. Chen et al., "Conceptual Framework and Research Method for Personality Traits and Sales Force Automation Usage," *Scientific Research Essays*, vol. 6, no. 17, 2011, pp. 3784–3793.
17. N. Limam and R. Boutaba, "Assessing Software Service Quality and

Trustworthiness at Selection Time," *IEEE Trans. Software Eng.*, vol. 36, no. 4, 2010, pp. 559–574.
18. S. Silas, E.B. Rajsingh, and K. Ezra, "Efficient Service Selection Middleware Using ELECTRE Methodology for Cloud Environments," *Information Technology J.*, vol. 11, no. 7, 2012, pp. 868–875.
19. G. Lee, B. Chun, and H.K. Randy, "Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud," *Proc. 3rd USENIX Conf. Hot Topics in Cloud Computing* (HotCloud 11), 2011, p. 4.
20. K. Kambatla et al., "Towards Optimising Hadoop Provisioning in the Cloud," *Proc. Conf. Hot Topics in Cloud Computing* (HotCloud 09), 2009, article 22.
21. H. Herodotou et al., "A What-if Engine for Cost-Based MapReduce Optimisation," *IEEE Data Eng. Bull.*, vol. 36, no. 1, 2013, pp. 5–14.
22. P. Lama and X. Zhou, "AROMA: Automated Resource Allocation and Configuration of MapReduce Environment in the Cloud," *Proc. 9th Int'l Conf. Autonomic Computing* (ICAC 12), 2012, pp. 63–72.

RAJIV RANJAN *is in the Digital Productivity Flagship at the Commonwealth Scientific and Industrial Research Organization (CSIRO), Australia, where he's also a senior research scientist, Julius Fellow, and project leader. At CSIRO, he leads research projects related to cloud computing, content delivery networks, and big data analytics for Internet of Things (IoT) and multimedia applications. Ranjan has a PhD in computer science and software engineering from the University of Melbourne. Contact him at rajiv.ranjan@csiro.au or http://rajivranjan.net.*

JOANNA KOŁODZIEJ *is an associate professor at Cracow University of Technology, Poland. Her research interests include data-intensive computing, grid and cloud computing, and artificial intelligence. Kołodziej has a habilitation in computer science from the Polish Academy of Sciences. She is a chair of the Polish chapter of the IEEE Computational Intelligence Society. She's a chair of the CHIPSET Cost Acrion project. Contact her at jokolodziej@pk.edu.pl.*

LIZHE WANG *is a ChuTian Chair Professor in the School of Computer Science at the China University of Geosciences (CUG). His research interests include high-performance computing, e-science, and spatial data processing. Wang has a doctor of engineering degree from the University of Karlsruhe, Germany. He's a fellow of the Institution of Engineering and Technology and the British Computer Society. Contact him at lizhewang@icloud.com.*

ALBERT Y. ZOMAYA *is the Chair Professor of High Performance Computing & Networking in the School of Information Technologies at the University of Sydney. His research interests include parallel and distributed computing, data intensive computing, and cloud computing. Zomaya has a PhD from the Department of Automatic Control and Systems Engineering, Sheffield University. He's a fellow of the American Association for the Advancement of Science, IEEE, and the Institution of Engineering and Technology. Contact him at albert.zomaya@sydney.edu.au.*