2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

Executing data-intensive workloads in a Cloud

Rizwan Mian, Patrick Martin
Database Systems Lab (DSL), School of Computing,
Queen's University, Kingston, Canada.
mian@cs.queensu.ca

***Abstract*—The promise of "infinite" resources given by the cloud computing paradigm has led to recent interest in exploiting clouds for large-scale data-intensive computing. Given this supposedly infinite resource set, we need a management function that regulates application workload on these resources. This doctoral research focuses on two aspects of workload management, namely scheduling and provisioning. We propose a novel framework for workload execution and resource provisioning, and associated models, algorithms, and protocols.**

Keywords: data-intensive computing; workload management; cloud computing.

I. INTRODUCTION

In the current information technology era, the pace and volume of data being generated is exceeding our ability to manage and analyze it. Cloud computing is, in turn, helping to realize the potential of large-scale data-intensive computing by providing effective scaling of resources. A growing number of companies, for example Amazon [1] and Google [2], rely on their ability to process large amounts of data to drive their core business. The scientific community is also benefiting in application areas such as astronomy [3] and life sciences [4] that have very large datasets to store and process.

Data-intensive computing presents new challenges for systems management in the cloud. One challenge is that data-intensive applications may be built upon conventional frameworks, such as shared-nothing database management systems (DBMSs) [5], or new frameworks, such as MapReduce [6], and so have very different resource requirements. A second challenge is that the parallel nature of large-scale data-intensive applications requires that scheduling and resource allocation be done to avoid data transfer bottlenecks. A third challenge is the need to support effective scaling of resources when large amounts of data are involved.

Workload management is an important component of systems management. In a cloud, the two main mechanisms used for workload management are scheduling requests and provisioning resources. Since the load on a data service in the cloud can fluctuate rapidly among its multiple workloads it is impossible for systems administrators to manually adjust the system configurations in order to maintain the workloads' objectives during their execution. It is therefore necessary to be able to automatically manage the workloads on a data service.

In this paper, we present our research plan for the problem of efficiently managing data-intensive workloads in a cloud. In the next section, we present our study of the area [7, 8] as related work. We formulate the problem of workload execution and appropriate resource provisioning in Section 3. Section 4 presents our framework that addresses this problem, parts of which have already been published [9]. Section 5 outlines our evaluation approach. Intended research is described in Section 6. The concluding sections 7 and 8 highlight the major contributions of the proposed research and its significance to the cloud users.

The authors acknowledge research support from National Science and Engineering Research Council of Canada (NSERC).

II. RELATED WORK

We have examined the state-of-the-art of workload management for data-intensive computing in clouds [7, 8]. In these works, a taxonomy is presented, which is then used to classify and evaluate current workload management mechanisms and systems. We present a summary of the survey below, and focus on the short-comings that our thesis aims to address.

We presently see a gap between data-intensive computing systems and provisioning systems. Most systems surveyed use shared-nothing clusters for large-scale data processing. On the other hand, the provisioning systems surveyed are usually applied to applications that do not require large-scale data processing. HadoopDB [10] uses Amazon's Elastic Cloud (EC2) [1] but does not use elasticity during execution of the MapReduce workflow. Similarly, Amazon has made Hadoop available in its cloud with Elastic MapReduce [11], however the number of VMs has to be selected before the execution starts. We believe that data-intensive computing systems must exploit a cloud's elasticity in order to cope with ever growing data sizes. Our proposed framework uses elasticity during workload execution.

Due to their disjoint nature, we discuss data-intensive computing systems and provisioning systems in separate sub-sections below.

A. Data-intensive computing systems

We see many data-intensive computing systems perform task scheduling and data replication independently, and place tasks close to data. Creating replicas for performance reasons is a good idea since moving large data takes time. However, there is a need to explore different replication strategies that best exploit data locality given a workload.

The data centers used by clouds are likely to scale into hundreds of thousands of nodes [12]. Using low cost unreliable commodity hardware to build a shared-nothing data center has its benefits. However, the probability of a node failure during data processing increases rapidly with scale [6, 10]. Therefore, fault-resiliency must be built into systems to address such issues.

978-0-7695-4691-9/12 $26.00 © 2012 IEEE
DOI 10.1109/CCGrid.2012.18

Most of the systems surveyed use workflow as a unit of execution and employ on-the-fly mapping of tasks to resources. This mapping approach is scalable and adapts to resource heterogeneity and failures [13]. Nevertheless, we believe that a system could benefit from prediction-revision mapping techniques that incorporate some pre-execution planning, workflow optimization, heuristics or history analysis. This additional analysis could help in creating an appropriate number of replicas or determining an appropriate amount of resources required for a computation.

Minimal execution time (makespan) is the prevalent objective function for workload execution in the survey. Clouds, however, are competitive and dynamic market systems in which users and providers have their own objectives. We therefore believe that objective functions related to cost and revenue, or participants' utilities, and scheduling policies based on them are appropriate and require further study.

B. Provisioning Systems

We observe that much of the current work related to provisioning in clouds involves scaling. Some of the scaling techniques involve a user defining rules in terms of condition and action pairs to deal with a situation e.g. overload. With multiple rules, many questions arise such as can multiple rules be defined on the same metrics, can they overlap and contradict each other?

In the ROM system [14], the data storage in a private cloud is decoupled from that in the public cloud so that the latter is not tied to the former through shared or replicated data resources. This seems to be a reasonable approach for large read-mostly data. Maintaining data consistency for read/write operations for large data between sites in a hybrid cloud is still an open problem.

The mechanisms for current provisioning techniques to handle varying workload demand may not scale for large-scale data processing. Nonetheless, one can admire the potential benefits of these techniques and argue that relevant mechanisms need to be developed for large data. Armbrust *et al.* point out that there is a need to create a storage system that could harness the advantage of elastic resources provided by a cloud while meeting existing storage systems expectations in terms of data consistency, data persistence and performance [12].

Systems that jointly employ scheduling and provisioning have been explored in grids. The Falkon [15] scheduler triggers a provisioner component for host increase or decrease. This host variation has also been explored during the execution of a workload hence providing dynamic provisioning. Presently, tasks stage data from a data repository. Since this can become a bottleneck as data scales, scheduling exploiting data locality is suggested as a solution. The MyCluster project [16] similarly allows Condor or SGE clusters to be overlaid on top of TeraGrid resources to provide a user with personal clusters. Various provisioning policies with different tradeoffs are explored including dynamic provisioning. The underlying motivation is to minimize wastage of resources. However, MyCluster is aimed at compute-intensive tasks. Given the similarities between grids and clouds, the joint techniques for scheduling and provisioning in these systems and related work are worth exploring for their relevance in clouds.

III. Problem Statement

We formulate the problem of workload execution and appropriate resource provisioning in a cloud as follows. A data-intensive workload consists of requests that involve a significant amount of data access under some constraints. Given a set of data-intensive applications $\boldsymbol{A}$, we say that the *workload* for $\boldsymbol{A}$ is a set of requests that are issued by the set of clients for $\boldsymbol{A}$. Each request is an instance of a request type $\boldsymbol{Qi}$ from a set $\boldsymbol{Q = \{Q1, Q2, \ldots, Qn\}}$ for $\boldsymbol{A}$. The data used by $\boldsymbol{A}$ is a set of data objects $\boldsymbol{D = \{D1, D2, \ldots, Dm\}}$.

The request types in $\boldsymbol{Q}$ are grouped into request classes. A request class $\boldsymbol{QCi}$ is a subset of $\boldsymbol{Q}$, such that all requests in the class access the same subset of data objects $\boldsymbol{Pi}$ (subset of $\boldsymbol{D}$) and all requests have the same service level objective $\boldsymbol{SLOi}$. We call $\boldsymbol{Pi}$ a *data partition* and we assume that the $\boldsymbol{Pi}$'s can overlap. The SLA for $\boldsymbol{A}$ is composed of the set of all $\boldsymbol{SLOi}$'s for all the request classes in $\boldsymbol{A}$.

A configuration $\boldsymbol{C}$ for the application set $\boldsymbol{A}$ contains the following:

- A set of VMs $Vi = \{v1, v2, \ldots, vr\}$, where each VM vk is a specific type (e.g. small, medium, large), has a specific cost rate per unit time (e.g. hour), and has attached storage with its associated cost.
- A mapping of the request classes, $\boldsymbol{QCi}$'s, to VMs in $\boldsymbol{V}$ such that every request class and corresponding data partition is assigned to at least one VM. Assignment to more than one VM involves replication of the partition. Overlapping partitions on the same VM share the same copy of the common data objects.

The provisioning problem is to select a configuration $\boldsymbol{C}$ for $\boldsymbol{A}$ such that an objective function is maximized (or minimized) and all the SLOs in the SLA for $\boldsymbol{A}$ are satisfied.

The problem of finding an optimal assignment for a set of tasks is NP-Complete in the general case and in several limited cases, even when communication among tasks is not considered [17]. Therefore, we explore various search methods to find a suitable configuration.

IV. Framework

Our approach of workload execution is simple. A configuration plays a central role in workload execution. A search algorithm hunts for a suitable configuration given an objective such as minimal dollar-cost. Using this configuration, the resources are made available or provisioned, and the workload is mapped on those resources. Finally, dynamic refinements are used to address any changes in the environment, workload or the SLOs.

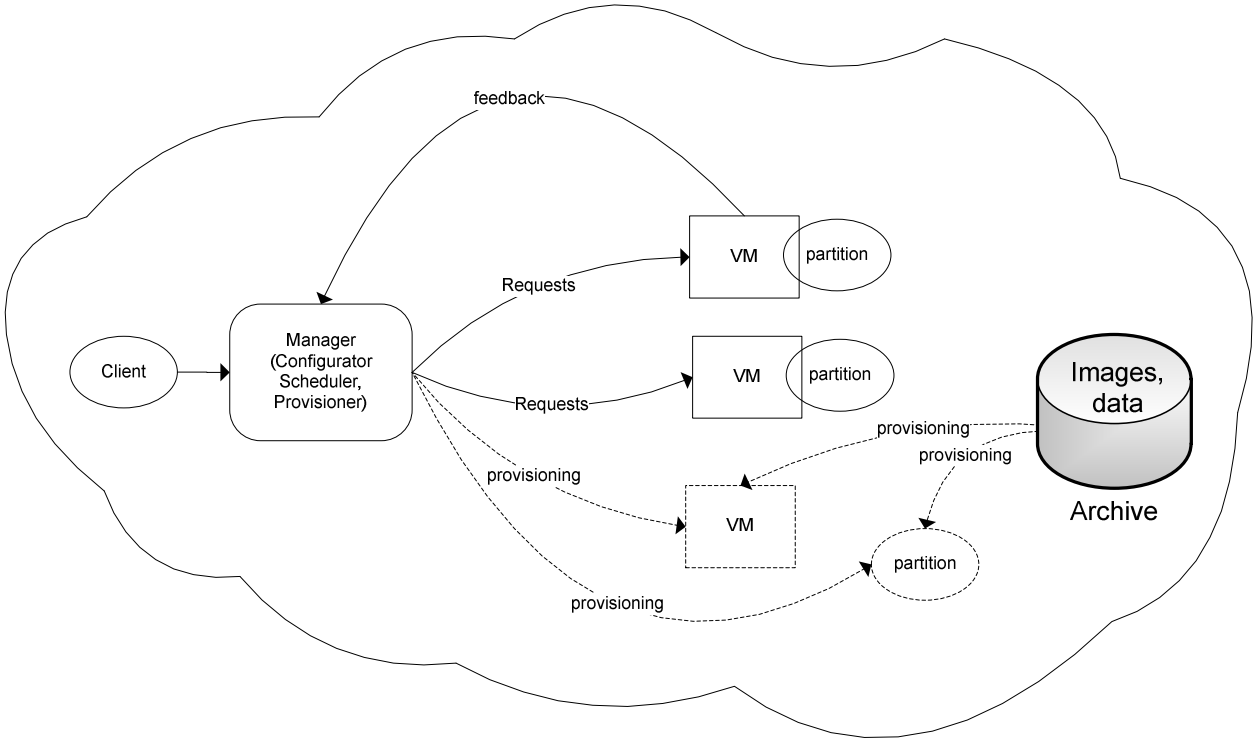

Figure 1. high-level architecture of a workload management framework in a cloud.

The architecture of the framework is shown in Fig 1. We provide a high-level description of the framework components and their interactions below. We identify four major parties in the framework: a client, a manager, storage and execution resources, and an archive. A client has some application workload that needs to be executed. The manager supervises the workload execution. The processing resources are booted with *settings* retrieved from the archive. The storage resources get a copy of the data in the archive. Both processing and storage resources are combined to provide an execution platform. The workload is executed on a number of execution platforms.

A client submits a workload to the manager. The manager consists of three components (a) a configurator, (b) a scheduler, and (c) a provisioner. The configurator is the brains of the system. It determines a suitable configuration, consisting of storage and processing resources, to execute the workload against an objective. This configuration is then passed to the provisioning process or the provisioner. The provisioner prepares the execution platforms. It allocates the processing resources (VMs) as required by the configuration.

The provisioner also attaches data partitions to the processing resources. In addition, the provisioner creates replicas of data partitions if needed. Once the provisioner finishes, the scheduler maps requests of the workload to the execution platforms as required by the configuration, and the workload execution begins.

The workload is executed and some feedback is sent back to the manager periodically. The feedback could include heartbeats or execution times. The manager may suggest a new configuration based on the feedback. Revisions to the current configuration may be necessary due to a number of reasons such as excessive SLA violations or a change in the workload. If the deployed configuration is revised, the provisioner and the scheduler respectively adjust the resources and dispatch the workload according to the new configuration .

The provisioning process facilitates the manager in meeting the objective of the workload execution. At the start, it provisions storage and execution resources and, as the time goes on, it acquires more resources or releases held resources as needed.

A. Determining a suitable configuration

We represent the set of all possible configurations for an application A as a directed graph $Configs = (N(A), E(A))$. The set of nodes, $N(A)$, and the set of edges, $E(A)$ are defined respectively as:

$N(A) = \{C \mid C$ *is a valid configuration for* $A\}$ *and*

$E(A) = \{(C_i, C_j) \mid$ *configuration* C_j *is obtained from* C_i *using a permitted modification}*,

An edge (C_i, C_j) in the search space indicates that configuration C_j can be obtained from configuration C_i by applying one of the modifications. Examples of modifications include adding a cheap VM, or load-balancing workload amongst existing VMs.

The configurator employs a search algorithm to explore the search space. The high-level architecture of the configurator is shown in Fig 2. Given a workload and an objective, a search algorithm looks for a suitable configuration. At every iteration, the search algorithm chooses a suitable modification on the current configuration. The modified configuration is evaluated using a *cost model*, and the cost is passed back to the search algorithm. Then, the algorithm decides whether to keep exploring the search space or to flag the evaluated configuration as a suitable one.

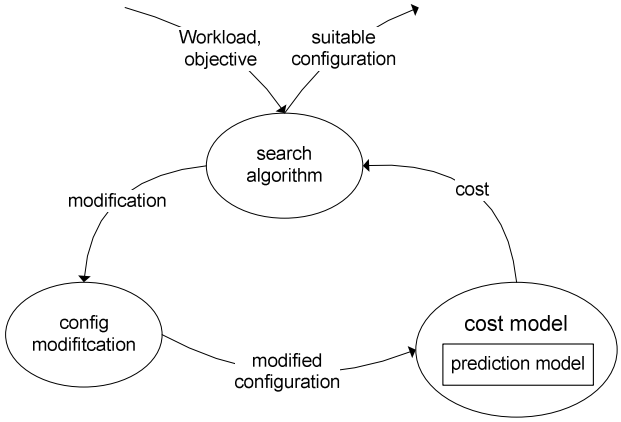

Figure 2. Architecture of the configurator

We have developed and evaluated a cost model that determines the dollar-cost of a configuration [9]. This cost model is a function of resource expense and penalties due to SLA violations. The cost model uses Queuing Network Models (QNMs) to forecast a SLO breach.

The elegance of this architecture is that various search algorithms can be used with various cost models and vice versa. Similarly, different cost models can be used with different prediction models.

In general, the size of a workload is unknown. Therefore, we argue it is appropriate to calculate the cost of workload execution per unit time, say an hour. Consequently, we parameterize our cost function with a unit time.

1) Algorithms

The space of possible configurations is very large and heuristics must be used to prune the search space. Therefore, we have developed some algorithms that explore the search space selectively. These algorithms are: (a) greedy, (b) adaptiveGreedy, and (c) tabu search. We use the greedy heuristics in the evaluation of our approach [9]. We intend to explore other search methods discussed later in the intended research.

2) Approaches for dynamic refinements

Presently, we consider two complementary processes for dynamic refinements: SLA improvements and system load-balancing. The SLA improvement process may acquire additional resources, while system load-balancing may relinquish unneeded resources. Both can operate simultaneously and independent of each other, and may or may not suggest actions to the manager. We also consider where these processes work together to improve SLA conditions of the workload classes while promoting workload balance amongst existing VMs.

V. EVALUATION

We claim to provide a novel, generic and autonomic framework for data-intensive workload execution and appropriate resource provisioning in clouds. The evaluation of the framework will be done at different stages, first by developing individual components, and then combining all the pieces to implement a complete prototype. This involves a considerable amount of work since each of the modules represents a large area of research. We will limit our current work within the scope of optimal workload execution against various objective functions for the Ph.D. dissertation, and continue with the rest of the framework as future research.

In our recent work [9], we evaluate the provisioning aspect of the system. We use a TPC-H [18] like workload, which queries large amounts of data, and hence qualifies as data-intensive. We also consider different cases of SLO penalties on the request classes in order to evaluate the impact of our approach for capturing SLOs in our framework. The experiments are performed on a public cloud i.e. Amazon EC2. The objective function used is minimal dollar-cost. We see that there is at most a 7% difference in the predicted and measured overall costs so our configurator gives a good initial representative model. However, we see that the response times for requests can vary by as much as 70% from the measured response times. We suggest another possible approach to solving this problem later in the intended research. This new approach replaces the current prediction model (QNM), without changing the framework. Our framework is generic since it can execute workload with multiple objectives, when the right cost and prediction models are plugged-in. We also intend to evaluate our work against realistic and/or diverse workloads discussed later in the intended work.

We intend to quantitatively compare our work against other work where possible. This process is simpler when the other work consists of formulae, or the corresponding system has a fairly direct match with our work and it exists online.

VI. INTENDED RESEARCH

We propose to complete the following work to satisfy the requirements for the doctoral thesis.

Prediction Model: Currently, we use an analytical model (QNM) for predictions, and experience highly inaccurate forecasts [9]. This is because the analytical models do not capture the inherent complexity of a cloud environment, such as affects on the buffer pool of concurrently executing requests [19].

Increasingly the research community is using experiment-driven performance modeling for database and multitier systems. We are considering experiment based methods to build a regression and/or a multi-layer perceptron (mlp) predictors. In these models, every request type is considered a dimension. A Latin Hypercube Sampling (LHS)

protocol reduces the number of experiments necessary while providing good coverage of the space of possible request mixes [19]. We can further prune the sample space by only considering the request types that place a measurable load on the system, or considering simple presence/absence of a request in the mix. This is our intermediate milestone. This is because building a prediction model from the generated samples suffers from a fundamental limitation: it requires re-learning the model for new workload or VM types, rendering them ineffective for online performance modeling. Gaussian response time models predict response times of TPC-H workload fairly accurately [20]. In addition, these models are trained experimentally offline as well as online. This way the models are adaptive to changing workload, suitable for our dynamic case. Sheikh et al. also develop a configuration model to deal with changing configuration. This is relevant for various types of VMs.

Workloads: We have used different static SLO workloads in our recent work [9]. We intend to explore dynamic workloads, which change in request types or number, or in SLA. We will also consider realistic and/or OLTP/OLAP mix workloads. Realistic workloads include Yahoo!'s Cloud Serving Benchmarks (YCSB) [21] and analyzing data of E-opinions.com [22]. While TPC-C [23] and TPC-E [24] are standard OLTP benchmarks, we already use a standard OLAP benchmark TPC-H [18]. Finally, we will use a random benchmark, which will be a synthetic dataset aimed to stretch the prediction model and dynamic refinement schemes.

User-centric Objectives: Presently, an objective function is defined over the complete workload, which may consist of multiple applications. An immediate opportunity is to define objective functions on a per application level. We also intend to explore other objective functions for workload execution.

Cost Models: Currently, we have a cost model only for estimating expense of a configuration. This cost model assumes a constant expense for communication. We would extend the current cost model to include more factors such as communication cost, remote data access and data replication.

Space Search: We have developed algorithms that search for the minimal dollar-cost configuration. These algorithms consist of: (a) greedy, (b) adaptiveGreedy, and (c) tabu search. We intend to provide another variant of greedy approach called (d) greedyTabu. Further, we can explore additional exotic search techniques like genetic algorithms or simulated annealing. We also intend to explore mathematical methods like linear programming to determine the optimal configuration.

Dynamic Refinements: Currently, we employ a simple greedy heuristic for dynamic refinements, which stops at the first bad configuration. This approach does not step outside a local minimum. We intend to reuse search algorithms in the configurator to modify the deployed configuration to obtain a more satisfactory configuration. For example, tabu search and regression prediction models can reused for this purpose. This approach is also relevant when the workload or the SLA changes. Further, after suggesting an initial configuration, the configurator need not stop. It can continue to explore the state space, hunting for a better configuration. In case of finding a better configuration, that is deployed at a suitable opportunity.

Partitioning: Our framework currently assumes that the data partitioning has been done prior to workload execution. We could extend our framework to assign partitions to execution resources so that workload being mapped onto resources use local data. We limit the scope of our graduate work, and assume that partitions are provided.

VII. Expected Contributions

We formulate the problem of workload execution and appropriate resource provisioning in the clouds, and the constructs to represent it. In traditional workload execution literature, the resource pool is assumed static. However, we extend the execution constructs to include provisioning resources during execution. An appropriate number of processing and storage resources depend on a suitable configuration. We transform the requirement for a suitable configuration into a search problem, and use standard search methods and heuristics.

We propose a novel, generic and autonomic framework for workload execution in a cloud. The framework executes a workload efficiently against a given objective. The management function in the framework exploits cloud's elasticity and dynamic refinements to uphold the goal.

Our framework allows pluggable cost and prediction models. We develop objective functions and associated cost models. We reuse search algorithms and prediction models where possible.

VIII. Significance of Work

We integrate dollar-cost with workload management using our framework. We expect our work to be useful from a number of angles: (a) estimate the expense of executing a workload in a cloud, (b) offer scale to workload execution by harnessing cloud's elasticity, (c) reduce time to result by exploiting rapid provisioning of cloud's resources, and (d) close the gap between data growth and the processing ability.

Acknowledgment

The authors acknowledge input from Dr. Farhana Zulkernine on the format of a research proposal, and Wendy Powley for reviewing the paper.

References

[1] Amazon, "Elastic Compute Cloud (EC2)," http://aws.amazon.com/ec2/.

[2] "Google App engine," http://code.google.com/intl/de-DE/appengine/.

[3] I. Raicu, I. Foster, A. Szalay and G. Turcu, "AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis," *Proc. TeraGrid Conference*, 2006.

[4] F. Desprez and A. Vernois, "Simultaneous Scheduling of Replication and Computation for Data-Intensive Applications on the Grid," *Journal of Grid Computing*, vol. 4, no. 1, 2006, pp. 19-31.

[5] D. Dewitt and J. Gray, "Parallel database systems. The future of high performance database systems," *Communications of the ACM*, vol. 35, no. 6, 1992, pp. 85-98.

[6] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, 2008, pp. 107-113.
[7] R. Mian, P. Martin, A. Brown and M. Zhang, "Managing Data-Intensive Workloads in a Cloud," *Grid and Cloud Database Management*, G. Aloisio and S. Fiore, eds., Springer, 2011.
[8] R. Mian, *Managing Data-Intensive Workloads in a Cloud (Ph.D. Depth Paper)*, 2011-581, P. Martin, School of Computing, Queen's University, http://research.cs.queensu.ca/TechReports/Reports/2011-581.pdf, 2011.
[9] R. Mian, P. Martin and J.L. Vazquez-Poletti, "Provisioning data analytic workloads in a cloud," *Future Generation Computer Systems (FGCS)*, 2011, pp. in press.
[10] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz and S.A. Rasin, "HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads," *Proc. VLDB Endow.*, vol. 2, no. 1, 2009, pp. 922-933.
[11] Amazon, "Elastic MapReduce," http://aws.amazon.com/elasticmapreduce/.
[12] M. Armbrust, et al., "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, 2010, pp. 50-58; DOI 10.1145/1721654.1721672.
[13] D.J. Abadi, "Data Management in the Cloud: Limitations and Opportunities.," *IEEE Data Eng. Bull.*, vol. 32, no. 1, 2009, pp. 3-12.
[14] H. Zhang, G. Jiang, K. Yoshihira, H. Chen and A. Saxena, "Resilient workload manager: Taming bursty workload of scaling internet applications," *Proc. 6th International Conference on Autonomic Computing, ICAC'09*, Association for Computing Machinery, 2009, pp. 19-28.
[15] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster and M. Wilde, "Falkon: a Fast and Light-weight tasK executiON framework," *Book Falkon: a Fast and Light-weight tasK executiON framework*, Series Falkon: a Fast and Light-weight tasK executiON framework, ed., Editor ed.^eds., ACM, 2007, pp.
[16] E. Walker, J.P. Gardner, V. Litvin and E.L. Turner, "Creating personal adaptive clusters for managing scientific jobs in a distributed computing environment," Inst. of Elec. and Elec. Eng. Computer Society, 2006, pp. 95-103.
[17] H. El-Rewini, T.G. Lewis and H.H. Ali, *Task scheduling in parallel and distributed systems*, Prentice-Hall, Inc., 1994, p. 290.
[18] "TPC-H Benchmark (Decision Support)," http://www.tpc.org/tpch/.
[19] S. Tozer, T. Brecht and A. Aboulnaga, "Q-Cop: Avoiding bad query mixes to minimize client timeouts under heavy loads," *Proc. Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, IEEE, 2010, pp. 397-408.
[20] M.B. Sheikh, et al., "A bayesian approach to online performance modeling for database appliances using gaussian models," *Book A bayesian approach to online performance modeling for database appliances using gaussian models*, Series A bayesian approach to online performance modeling for database appliances using gaussian models, ed., Editor ed.^eds., ACM, 2011, pp. 121-130.
[21] B.F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan and R. Sears, "Benchmarking cloud serving systems with YCSB," *Proc. 1st ACM Symposium on Cloud Computing, SoCC '10, June 6, 2010 - June 11, 2010*, Association for Computing Machinery, 2010, pp. 143-154.
[22] P. Massa and P. Avesani, "Controversial users demand local trust metrics: An experimental study on Epinions.com community," *Proc. 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference, AAAI-05/IAAI-05*, American Association for Artificial Intelligence, 2005, pp. 121-126.
[23] "TPC-C Benchmark," http://www.tpc.org/tpcc/.
[24] "TPC-E Benchmark," http://www.tpc.org/tpce/.