Resource Allocation and Load Distribution for Mixed Workloads under Cloud Environment

Sutha.K*1,Kaladevi.P*2

1*PG Scholar of Computer Science, K.S.Rangasamy College of Technology, Tiruchengode, India.*

Email: suthakrishnannew@gmail.com

2*Assistant professor (Academic), K S Rangasamy College of Technology, Tiruchengode, India.*

Email: kaladevi@ksrct.ac.in

***Abstract*--Computational resources are provided by the third party cloud providers. The cloud nodes use the shared resources. Cloud computing is Internet-based computing to share resources, software and information. Both transactional and long-running analytic computations are comprised into workloads. Scientific simulations to multi-tier transactional applications are referred as workloads. The workload and resource management models are designed separately with respect to the workload type. Transactional workloads are managed using flow control, load balancing and application placement. Noninteractive workloads need scheduling and resource control. Common virtualization control mechanisms are used to manage mix of transactional and batch workloads. Relative Performance Functions (RPFs) are used to permit trade-offs between different workloads. Application placement controller (APC) provides the decision-making logic for placement of both web and noninteractive workloads. Placement optimizer is tuned to provide dynamic placement for web applications. Placement algorithm with placement control loop is used to place jobs in the server.The autonomic workload management model is enhanced to manage resources for transactional and batch workloads. Placement algorithm with placement control loop is improved to manage low level and high level resources. Data retrieval rate is integrated with the placement process. Workload and web applications are placed with estimated and requested computation loads. Data distribution factor is integrated with the system.**

1. INTRODUCTION

Compute clouds are commonly used by many different users that rely on the existing computing infrastructure to deploy their workloads. As a result, heterogeneous workloads run on the same physical nodes and pose extraordinary challenges on a cloud management middleware. Integrated performance management of mixed workloads is a challenging problem. First, performance goals for different workloads tend to be of different types. For interactive workloads, goals are typically defined in terms of average or percentile response time or throughput over a short time interval, while goals for noninteractive workloads concern the performance of individual jobs. Second, due to the nature of their goals and the short duration of individual requests, interactive workloads lend themselves to management at short control cycles, whereas noninteractive workloads typically require calculation of a schedule for an extended period of time. In addition, different types of workload require different control mechanisms for management. Transactional workloads are managed using flow control, load balancing, and application placement. Noninteractive workloads need scheduling and resource control. Traditionally, these have been addressed separately.

To illustrate the problems inherent in managing these two types of workload together, let us consider a simple example. Consider a system consisting of four identical machines. At some point in time, in the system there is one transactional application, TA, which requires the capacity of two machines to meet its average response time goal, defined as a Service Level Agreement (SLA). The system also includes four identical batch jobs, each requiring one physical machine for a period of time t and having completion time goal of T = 3t. The jobs are placed in a queue and are labeled J1, J2, J3, and J4, according to their order in the queue. The system must decide how many jobs should be running—that is, how many machines should be allocated to the transactional application and to batch jobs, respectively. Let us consider two of the possible configurations. In the first configuration, one machine is allocated to batch workload and three machines are used by TA [1]. Thus, jobs execute in sequence and complete after time t, 2t, 3t, and 4t. As a result, J4 violates its SLA goal, while TA over achieves its SLA target. In the second configuration, two machines are allocated to batch workload, which permits the four jobs to complete at times t, t, 2t and 2t, respectively. Thus, all jobs exceed their SLA goal, while TA also meets its SLA target. Clearly, the second configuration is a better choice.

Our technique relies on common virtualization control mechanisms to manage workloads. In addition, our system uses Relative Performance Functions (RPFs) to permit trade-offs between different workloads. The RPFs define application performance relative to that application's goal. It can therefore be seen that equalizing the achieved relative performance between two applications results in "fairness"— the applications will be equally satisfied in terms of relative distance from their goals. The original contribution of this paper is a scheme for modeling the performance of, and managing, noninteractive long-running workloads.

II. RELATED WORK

The explicit management of heterogeneous workloads was CPU shares are manually allocated to run mixed workloads on a large multiprocessor system. This is a static approach, and does not run workloads within virtual machines (VM). The relative performance functions we use in our system are similar in concept to the utility functions that have been used in real-time work schedulers to represent the fact that the value produced by such a system when a unit of work is completed can be represented in more detail than a simple binary value indicating whether the work met its or missed its goal. Outside of the realm of the real-time systems, focus on a utility-guided scheduling mechanism driven by data management criteria, since this is the main concern for many data-intensive HPC scientific applications. In our work, we focus on CPU-bound heterogeneous environments, but our technique could be extended to observe data management criteria by expanding the semantics of our RPFs. Despite the similarity between an RPF and a utility function, one difference should be pointed out. While utility functions are typically used to model user satisfaction or business value resulting from a particular level of performance, an RPF is merely a measure of relative performance distance from the goal. Hence, we do not study the correctness of RPFs with respect to modeling user satisfaction. If such a satisfaction model exists, it may be used to transform an RPF into a utility function. In [5] and [6], the authors leverage utility-based systems for making placement decisions and provisioning resources: these works do not address the problem of managing heterogeneous workloads, as they are both focused on transactional workloads only.

There is also previous work in the area of managing workloads in virtual machines. The overhead of a dynamic allocation scheme that relies on virtualization, covering both CPU-intensive jobs and transactional workloads, but does not consider mixed environments. Bodı´k et al. [8] uses Machine Learning techniques for making management decisions. The authors stress the max-min-shares approach, focusing on the use of current virtualization control knobs. Work presented focuses on the cost of VM migration, and mitigate it by minimizing migrations over time. In [2], authors propose a joint-VM sizing approach in which multiple VMs are consolidated and provisioned as an aggregate. In [3], authors propose a holistic approach to treat performance, power and cooling of IT infrastructures. Neither of these techniques provides a technology to dynamically adjust allocation based on SLA objectives in the face of resource contention. The authors of [4] present new scheduling algorithms for the cloud, but their effort is focused only on long running jobs and VMmigration is not used. The authors of [7] focus their work on multitiered transactional systems, with special effort on avoiding the damaging effects of workload burstiness. The authors propose a resource manager that is decoupled from the infrastructure provider. In our work, the resource manager is part of the computing infrastructure.

Placement problems in general have also been studied in the literature, frequently using techniques including bin packing, multiple knapsack problems, and multidimensional knapsack problems. The optimization problem that we consider presents a nonlinear optimization objective in contrast. The authors evaluate a similar problem to that addressed in our work and use a simulated annealing optimization algorithm. Their strategy aims to maximize the overall system utility while we focus on first maximizing the performance of the least performing application in the system, which increases fairness and prevents starvation. A fuzzy logic controller is implemented to make dynamic resource management decisions. This approach is not applicationcentric—it focuses on global throughput—and considers only transactional applications. The algorithm proposed allows applications to share physical machines, but does not change the number of instances of an application, does not minimize placement changes, and considers a single bottleneck resource.

III. SYSTEM ARCHITECTURE

The managed system includes a set of heterogeneous server machines, referred to henceforth as nodes. Web applications, which are served by application servers, are replicated across nodes to form application server clusters. Requests to these applications arrive at an entry router which may be either an L4 or L7 gateway that distributes requests to clustered applications according to a load balancing mechanism, and implements a flow control technique. Long-running jobs are submitted to the system via the job scheduler, which, unlike traditional schedulers, does not make job execution and placement decisions. In our system, the job scheduler only manages dependencies among jobs and performs resource matchmaking. Once dependencies are resolved and a set of eligible nodes is determined, jobs are submitted to the application placement controller (APC).

APC is the most important component of the system. It provides the decision-making logic that affects placement of both web and noninteractive workloads. Its placement optimizer calculates the placement that maximizes the minimum satisfaction across all applications. We introduced a technique that provides such dynamic placement for web applications: APC used in this system is an augmented version of that controller. We modified the algorithm inputs from application CPU demand to a per-application RPF of allocated CPU speed. Permitting resource requirements to be represented by nonlinear RPFs allows us to better deal with heterogeneous workloads which may differ in their sensitivity to a particular resource allocation.

In our work, we leverage the flow controller, which comes up with an RPF for each web application This RPF gives a measure of application satisfaction with a particular allocation of CPU power given its current workload intensity and performance goal. Generating RPFs for the long running jobs is not studied in previous work, and is the main contribution of this work. Each job has an associated performance goal, and when a job completes exactly on schedule, the value of the RPF is zero. Otherwise, the value increases or decreases linearly depending on the distance of completion time from the goal. Currently, we only support completion time goals, but we plan to extend the system to handle other performance objectives.

APC relies on the knowledge of resource consumption by individual requests and jobs. The web workload profiler, obtains profiles for web requests in the form of the average number of CPU cycles consumed by requests of a given flow. The job workload profiler obtains profiles for jobs in the form of the number of CPU cycles required to complete the job, the number of threads used by the job, and the maximum CPU speed at which the job may progress.

IV. INTEGRATED MANAGEMENT OF HETEROGENEOUS WORKLOADS

The goal of the technique introduced in this paper is to make placement decisions that involve applications of different nature, more specifically transactional applications and long-running workloads. Given the different characteristics of each workload, that make their performance hardly comparable, we leverage RPF to produce a normalized representation of their performance. RPFs are leveraged by the placement algorithm to make placement decisions, with
the goal of maximizing the relative performance delivered by all the applications in the system.

The placement algorithm and RPFs for transactional workloads are not a novel contribution of this work. The main contribution of this work is the introduction of a model that allows the creation of RPFs for long running workloads. The placement algorithm is extended to leverage such a model and is therefore able to deal with heterogeneous workloads. In the following sections, we present a formal description of the problem addressed in this work.

The application placement problem is known to be NP-hard and heuristics must be used to solve it. In this paper, we leverage an algorithm proposed and adapted to a nonlinear optimization objective. The basic algorithm, as described above, is surrounded by the Placement control loop, which is designed to have the Application Placement Controller periodically inspect the system to determine if placement changes are now required to better satisfy the changing extant load. The period of this loop is configurable and can be interrupted when the configuration of the system is changed.

The placement change phase is executed several times, each time being referred to as a round. Each round invokes the placement change method, which makes a single new placement suggestion starting from the placement suggestion provided by the previous round's execution. The placement change method first iterates over nodes. For each node, it iterates over all instances placed on this node and attempts to remove them one by one, thus generating a set of configurations whose cardinality is linear in the number of instances placed on the node. For each such configuration it iterates over all applications with some unsatisfied CPU demand, attempting to place new instances on the node as permitted by the constraints. The key to the accuracy and performance of the algorithm is the order in which nodes, instances, and applications are visited in the three nested loops. The order must be driven by the values of um. In the outer loop, nodes are processed according to the highest utility of stopping, which is calculated for a node by calculating the highest RPF among all applications placed on that node after an instance of an application is stopped. In the intermediate loop, instances are processed in decreasing RPF order calculated against the current placement. Finally, in the inner loop, applications are considered in the increasing order of RPF. In addition, numerous carefully tuned shortcuts are used to reduce the computational complexity of the algorithm.

The computational complexity of the technique is O(NMk), where k is the maximum number of instances on a node. The complexity may further increased by the RPF calculation. For transactional workloads, we can evaluate RPFs in O(1), but for long running workloads that bound is nonconstant. In practice, the computation time remains below 10 s for up to 700 applications, reaching 23 s for 1,000 applications. Recall that the average size for enterprise datacenter ranges from tenths of nodes to a few hundreds typically. These values are perfectly acceptable for the purpose of these systems.

V. PERFORMANCE MODEL FOR NONINTERACTIVE WORKLOADS

In this section, we focus on applying our placement technique to manage long-running jobs. We start by observing that a performance management system cannot treat batch jobs as individual management entities, as their completion times are not independent. For example, if jobs that are currently running complete sooner, this permits jobs currently in the queue to complete sooner as well. Thus, performance predictions for long-running jobs must be done in relation to other long-running jobs.

Another challenge is to provide performance predictions with respect to job completion time on a control cycle which may be much lower than job execution time. Typically, such a prediction would require the calculation of an optimal schedule for the jobs. To trade off resources among transactional and long-running workloads, we would have to evaluate a number of such schedules calculated over a number of possible divisions of resources among the two kinds of workloads. The number of combinations would be exponential in the number of nodes in the cluster. We therefore propose an approximate technique, which is presented here.

A. Job Characteristics

We are given a set of jobs. With each job m we associate the following information:

- Resource usage profile. A resource usage profile describes the resource requirements of a job and is given at job submission time—in the real system, this profile comes from the job workload profiler. The profile is estimated based on historical data. Each job m consists of a sequence of Nm stages, $s_1, \ldots, sN_m$, where each stage s_k is described by the following parameters:
  - The amount of CPU cycles consumed in this stage, $\alpha_{k,m}$.
  - The maximum speed with which the stage may runs, $w_{k,m}^{max}$.
  - The minimum speed with which the stage must run, whenever it runs, $w_{k,m}^{min}$. - The memory requirement $\gamma_{k,m}$.
- Performance objectives. The SLA objective for a job is expressed in terms of its desired completion time, τ_m, which is the time by which the job must complete. Clearly, τ_m should be greater than the job's desired start time,T_m^{start}, which itself is greater than or equal to the time when the job was submitted. The difference between the completion time goal and the desired start time, τ_m - T_m^{start}, is called the relative goal, and can be understood as the maximum acceptable job runtime. Notice that job runtime will depend on allocated resources to the Virtual Machine in which the job runs.

We are also given an RPF that maps actual job completion time t_m to a measure of satisfaction from achieving it, $u_m(t_m)$. If job m completes at time t_m, then the relative distance of its completion time from the goal is the job's actual runtime normalized to its relative goal, which is expressed by the RPF of the following form:

$$u_m(t_m) = \frac{T_m - t_m}{T_m - T_m^{start}} \qquad (1)$$

- Runtime state. At runtime, we monitor and estimate the following properties for each job: current status, which may be either running, not started, suspended, or paused; and CPU time consumed thus far, $\propto_m^*$.
- Relative goal factor. For the purpose of easily controlling the tightness of SLA goals in our experiments, we introduce a relative goal factor which is defined as the ratio of the relative goal of the job to its execution time at the maximum speed, $\frac{T_m - T_m^{start}}{t_m^{best}}$.

B. Hypothetical Relative Performance

To calculate job placement, we need to define an RPF which APC can use to evaluate its placement decisions. While the actual relative performance achieved by a job can only be calculated at completion time, the algorithm needs a mechanism to predict the relative performance that each job in the system will achieve given a particular allocation. This is also the case for jobs that are not yet started, for which the expected completion time is still undefined. To help answer questions that APC is asking of the RPF for each application, we introduce the concept of hypothetical relative performance.

VI. LOAD DISTRIBUTION BASED RESOURCE ALLOCATION MODEL

The system is designed to manage the resources and workloads under clouds. The web applications and transactional applications are managed by the system. Data and computational resources are allocated by the system. The system is divided into five major modules. They are resource monitoring, data sources, workload manager, Application Placement Controller (APC) and load distribution. Resource monitoring module is designed to monitor the computational resources. Data sources module is designed to manage the data sources. Workload manager module handles the workload submission process. Application Placement Controller (APC) module is designed to handle resource allocation for the applications. Load distribution module is designed to distribute the workloads to the providers.

A. Resource Monitoring

The computational resources are provided by a set of cloud nodes. Processor and memory resources are provided by the nodes. Total resources and available resource levels are monitored and updated to the server. Resource levels are updated with workload execution process.

B. Data Sources

Data sources are used to provide databases and data files. Data sources are placed in different machines. Workloads are scheduled with data source requirements. Data source distribution is also managed by the system.

C. Workload Manager

Transactional applications and batch jobs are submitted as workloads. Transactional web workloads are submitted to the web server. Interactive and non-interactive workloads are assigned with data and resources. The workloads are collected from the clients.

D. Application Placement Controller (APC)

Application placement controller handles the service placement process. Placement optimizer is used to verify the performance levels. Services and resource levels are analyzed under the APC. Relative performance functions are used to estimate the performance levels.

E. Load Distribution

The transactional and batch workloads are assigned to the resources. The transactional web applications are assigned with high priority. The batch workloads are also assigned with data source levels. Performance prediction is estimated for load distribution process.

VII. CONCLUSION

Enterprise datacenters consolidate workloads on the same physical hardware to reduce the cost of infrastructure and electrical energy. Heterogeneous workloads have different nature. A heterogeneous set of applications is running a web application and a batch job on the same physical server. The system manages the mixed workloads with long running jobs and transactional applications. The system allocates workload types on the same physical hardware with virtualization control mechanism. The system manages resources with load distribution. Data source allocation is also carried out under the system. The system reduces the computational time. Efficient resource monitoring process is provided in the cloud environment.

REFERENCES

[1] David Carrera, Malgorzata Steinder, Ian Whalley, Jordi Torres, and Eduard Ayguade, “Autonomic Placement of Mixed Batch and Transactional Workloads”, IEEE Transactions On Parallel And Distributed Systems, Vol. 23, No. 2, February 2012.

[2] S. Meng, L. Liu, and V. Soundararajan, “Tide: Achieving Self-Scaling in Virtualized Datacenter Management Middleware,” Proc. 11th Int’l Middleware Conf. Industrial Track, pp. 17-22, 2010.

[3] Y. Chen, D. Gmach, C. Bash, C. Hoover, and S. Singhal, “Integrated Management of Application Performance, Power and Cooling in Data Centers,” Proc. IEEE Network Operations and Management Symp. (NOMS), pp. 615-622, Apr. 2010.

[4] Z. Zhang, L.T.X. Phan, G. Tan, S. Jain, H. Duong, B.T. Loo, and I. Lee, “On the Feasibility of Dynamic Rescheduling on the Intel Distributed Computing Platform,” Proc. 11th Int’l Middleware Conf. Industrial Track, pp. 4-10, 2010.

[5] R. Urgaonkar, U. Kozat and M. Neely, “Dynamic Resource Allocation and Power Management in Virtualized Data Centers,” Proc. IEEE Network Operations and Management Symp. (NOMS), pp. 479-486, Apr. 2010.

[6] J. Hanson, I. Whalley, M. Steinder, and J. Kephart, “Multi-Aspect Hardware Management in Enterprise Server Consolidation,”Proc.IEEE Network Operations and Management Symp. (NOMS), pp. 543-550, Apr. 2010.

[7] A. Caniff, L. Lu, N. Mi, L. Cherkasova, and E. Smirni, “Efficient Resource Allocation and Power Saving in Multi-Tiered Systems,” Proc. 19th Int’l Conf. World Wide Web (WWW ’10), pp. 1069-1070, 2010.

[8] P. Bodı´k, R. Griffith, A. Fox, M.I. Jordan, and D.A. Patterson, “Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters,” Proc. Conf. Hot Topics in Cloud Computing (HotCloud ’09), 2009.