Autonomic Virtual Machine Placement in the Data Center

Chris Hyser, Bret McKee, Rob Gardner, Brian J. Watson
HP Laboratories
HPL-2007-189
February 26, 2008*

Automation, Manageability, Architecture, Virtualization, data center, virtual machine, placement

We present a high level overview of a virtual machine placement system in which an autonomic controller dynamically manages the mapping of virtual machines onto physical hosts in accordance with policies specified by the user. By closely monitoring virtual machine activity and employing advanced policies for dynamic workload placement, such an autonomic solution can achieve substantial cost savings from better utilization of computing resources and less frequent overload situations.

Autonomic Virtual Machine Placement in the Data Center

Chris Hyser (chris.hyser@hp.com)
Bret McKee (bret.mckee@hp.com)
Rob Gardner (rob.gardner@hp.com)
Brian J. Watson (brian.j.watson@hp.com)
Hewlett Packard Laboratories

December 11, 2007

Abstract

We present a high level overview of a virtual machine placement system in which an autonomic controller dynamically manages the mapping of virtual machines onto physical hosts in accordance with policies specified by the user. By closely monitoring virtual machine activity and employing advanced policies for dynamic workload placement, such an autonomic solution can achieve substantial cost savings from better utilization of computing resources and less frequent overload situations.

1 Introduction

Virtualization allows a great deal of flexibility in the provisioning and placement of servers and their associated workloads in a data center. The problem of provisioning virtual machines is receiving a great deal of attention[1, 2, 3], while placement is often a relatively static manual process. A virtual machine using only virtual resources provides a computational encapsulation that may be moved between physical hosts in its entirety. A feature enabled by this encapsulation is the capability to migrate a running virtual machine from one physical host to another without a significant interruption of service[4]. This technology is currently available in VMware's ESX product[5] and Xen[6], and it is anticipated in Microsoft's Viridian product[7]. This document refers to this technology as *live migration*.

Live migration imposes certain restrictions on the source and destination physical hosts, possibly including requiring compatible virtualization software, comparable CPU types, similar network connectivity, and the use of shared storage. The exact requirements depend on the virtualization software[8, 9]. We define a *migration domain* as a set of physical hosts among which a virtual machine can be migrated if sufficient resources are available.

Live migration opens up the possibility of dynamically changing the mapping of virtual machines to physical hosts in a data center. This could be done to improve quality of service for data center users or to increase profitability of providing service for data center owners.

2 The Virtual Machine Mapping Problem

The cost savings that virtualization can provide[1, 2], are causing IT organizations to switch to virtualized data centers. While this model provides a number of advantages, it becomes necessary to manage the mapping of these virtual machines to the physical hosts in the data center.

Mapping a virtual machine "correctly" onto a physical host requires knowing the capacity of the physical hosts and the resource requirements of the virtual machines as well as an understanding of how to resolve resource conflicts in a way that is consistent with data center policies.

In virtual data centers that do not use autonomic live migration, a suitable physical host is generally chosen at the time of virtual machine creation or activation, and the virtual machine executes there until it is restarted. Operator intervention is required to deal with unanticipated events such as variations in virtual machine load or unplanned downtime of physical hosts.

Using an autonomic controller, it is possible to react in accordance with data center policies to these events, and to introduce policy components that optimize other factors such as power consumption or cooling loads.

Because virtual machine loads often change over time, it is not sufficient to make good initial placement choices. It is necessary to dynamically alter placements if constraints are to be satisfied as conditions change in the data center. Since these changes occur in response to observed conditions, such a dynamic controller is a feedback controller must exhibit the same stability and non-thrashing behaviors as any other feedback controller.

2.1 Resources

As part of evaluating a new mapping, the controller must predict how the current resource loads will change in the new mapping. Having the controller consider many dimensions of resource consumption allows it to more accurately predict the effect that live migrations will have on the resource loads of both the source and destination physical hosts.

Our controller currently considers memory, CPU, network bandwidth and disk bandwidth when evaluating data center layouts.

2.2 Constraints

Service level agreements for a virtual data center can specify not only traditional performance metrics but additional restrictions relating to security or availability. It might not be acceptable for a user's virtual machines ever to run on the same physical host as those of a competitor or a user might require that two of their virtual machines never run on the same physical host so one will remain available even if a physical host fails.

2.3 Solution Space Size

While it is possible for an operator to manage the virtual machine to physical host mapping, the number of possible mappings quickly becomes too large for human capabilities as the number of physical hosts and virtual machines increases. The number of possible solutions is:

$$(numberofphysicalhosts)^{(numberofvirtualmachines)} \tag{1}$$

which makes examining all possible solutions impractical for an autonomic controller for all but trivial problems.

2.4 Related Problems

Several NP-hard problems initially appear related to the task of placing virtual machines on physical hosts. Two such problems are N-dimensional set packing[10] and N-dimensional bin packing[10], which have been widely studied and for which good approximate solutions can be computed quickly. While the research into these problems can be used as a starting point, there are important differences between the virtual machine mapping problem and the problems these algorithms are typically used to solve.

These algorithms start with a "clean slate" and attempt to pack items into containers - for example optimally loading empty trucks with goods from a warehouse. Since all the goods must be moved from the warehouse to a truck to generate a solution, these algorithms do not need to account for illegal intermediate states or consider the number of moves required to reach their final state.

The virtual machine placement problem starts with an existing mapping and must generate solutions from that mapping. To continue the example, there is no warehouse - all the goods are pre-loaded on trucks. Not only must a final state be generated, but that state must be reachable from the initial state without violating any constraints.

Because live migrations require CPU time and network bandwidth, they place additional resource usage loads on the system, and the changes in VM placement can disrupt load monitoring during the execution and for some settling period afterward. If not considered these effects can result in undesirable behaviors such as system instability and thrashing.

2.5 Iterative Rearrangement Problem

The virtual machine mapping problem starts with a configuration where the virtual machines are already deployed on physical hosts, and any algorithm which solves this problem must generate solutions from this initial layout. Such solutions are lists of serial and parallel live migrations to be executed to move the data center from the current state to the desired state.

As these live migrations are performed, all intermediate states must satisfy at least a subset of the problem constraints, although it is possible for some constraints to be relaxed during this period. For example, if the virtualization software supported oversubscribing physical memory it might be permissible to exceed a host's physical memory size during migrations even if the data center policies generally did not allow it.

It is also sometimes necessary to have an intermediate state that is worse than the initial state if a better final state is to be reached. External conditions might also allow constraints to be relaxed - a fire or an air conditioner failure in the data center might allow a degree of overload not generally permitted.

Virtual machines and physical hosts may be added or removed from the data center and resource loads can vary, so the quality of any virtual machine mapping may degrade over time and will need to be re-evaluated when this happens.

We refer to the problem of evaluating the layout of a data center and generating a list of live migrations to improve the quality of the mapping of virtual machines onto physical hosts in that data center as the *Iterative Rearrangement Problem.*

3 Status, Results and Future Work

Initial work on the controller focused on building a framework with a Load Balancing Policy, in which data center load is spread among all available physical hosts and resource usage is balanced as much as possible across all resource types.

3.1 Prototype

We have constructed a prototype of controller which attempts to solve the Iterative Rearrangement Problem and control the mapping of virtual machines to physical hosts. Our prototype controller takes performance factors, data center policies and service level agreements into account when generating a mapping. The performance resources it monitors and attempts to predict are CPU, memory, network I/O bandwidth and disk/SAN I/O bandwidth usage.

3.1.1 Prototype Architecture

The architecture of our prototype is shown in Figure 1. There is a virtual machine placement service, which communicates with a virtualization management service to get performance data and to execute live migrations. Also reported by the virtualization management service is the collection of migration domains to be managed, and their constituent physical hosts and virtual machines. The placement service can associate a different policy with each migration domain. It is assumed that inter-migration domain migrations are not allowed - if

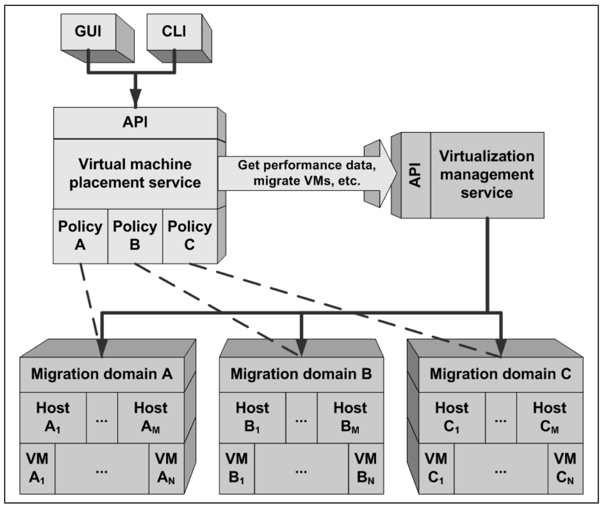

Figure 1: System Architecture.

The prototype uses HP's Virtual Machine Manager (VMM)[11] product as its virtualization management service, and the prototype's GUI is loosely integrated into HP's Systems Insight Manager (SIM)[12] product. Typical experiments consist of running various workloads in virtual machines on a cluster of physical hosts, injecting workload changes into the system, and monitoring the behavior of the placement controller in response to these changes.

3.1.2 Multiple Policies

A policy is used by a solver to evaluate virtual machine to physical host mappings so it can choose mappings which align with the user's goals. Policies consist of rules providing information about how to compare all the variables which the optimizer uses to evaluate mappings. Using the rules provided by the policy, the solver attempts to find an optimal mapping of virtual machines onto the physical hosts in the migration domain while meeting the constraints required for stability and non-thrashing.

The placement service then uses live migration to implement its policies' solutions, either autonomously or with administrator approval. The placement service provides a programmatic interface (API) which allows it to be controlled via a graphical user interface module (GUI), a command line user interface module (CLI) and other programs.

3.2 Results

Results are presented for an experiment performed using our autonomic controller with the Load Balance Policy on a simulator and on four identically configured Hewlett Packard DL360G4 servers, each with 8GB memory, 2x3.60 Ghz

Intel Xeon processors, 2x1Gb Ethernet interconnects and shared Fiber Channel SAN storage connected to a dual channel 2Gb Fiber Channel adapter. Three of these machines were used as the migration domain, while the fourth hosted the autonomic control software.

Running the experiment on our simulator and on the hardware results in the same number of migrations being performed and the same final layout - only the timing varies. When running the experiments on hardware, the controller is forced to wait for migrations to be performed and for the performance numbers to stabilize after migrations are performed. Because the simulator does not require these delays, the x-axis of the resulting graphs is compressed, which makes the graphs smaller and easier to understand. For these reasons, the graphs presented are generated from simulation results.

Three types of load were generated with a load generation tool during this experiment: CPU load, LAN load and SAN load. For each of these resources, there were three virtual machines generating that type of load, for a total of nine virtual machines. The LAN and SAN generators require some CPU cycles to generate their loads, so the also generate some CPU load. For this experiment, the maximum number of migrations per minute was set to one to make the migrations easier to follow. Figure 2 shows the per host CPU loads and the virtual machines generating them, and figure 3 shows a summary of the CPU, LAN and SAN utilizations during the experiment,

In Figure 2, CPU generators are shown as blue boxes, LAN generators as red boxes, and SAN generators as green boxes. This figure shows that at time zero, the initial configuration has all three CPU generators on one physical host (node1), all three SAN generators on a second physical host (node2) and all three LAN generators on a third physical host (node3). Because the virtual machines on each physical host are generating the same type of load, the physical host loads are maximally unbalanced at the beginning of the experimental run.

The Load Balancing Policy attempts to drive all the host's loads toward the migration domain average, as can be seen in figure 3, which shows the amount of the three resources being consumed on each physical host as a percentage difference from the migration domain average.

In the periods from minute zero through minute five, the amount of imbalance exceeds the controller's threshold, so it moves virtual machines to balance the loads across all resource types. Starting with minute six, the load is balanced, as is shown by the zero differences from the migration domain average in figure 3, and by figure 2 showing that there is one CPU (blue), one LAN (red) and one SAN (green) generating virtual machine on each physical host.

3.3 Future Work

3.3.1 Different Algorithms

The current prototype controller solver is based upon a modified version of the Simulated Annealing algorithm[13], but other algorithms such as Genetic

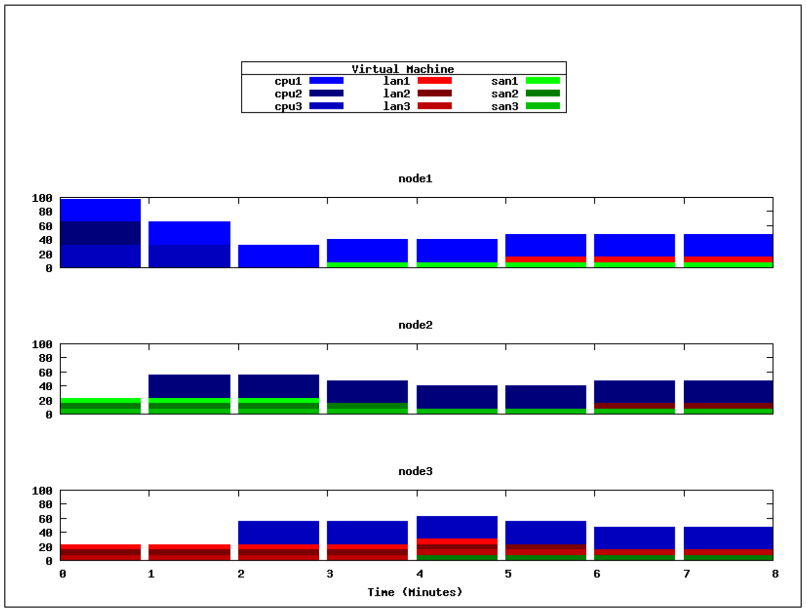

Figure 2: Balance Policy per host CPU loads as stacked bars of virtual machine load.

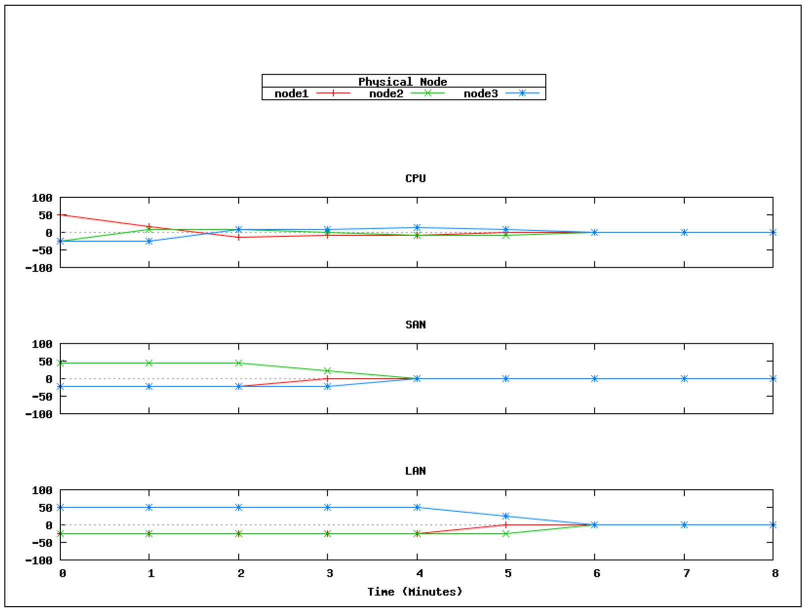

Figure 3: Balance Policy Load Resource Summaries - Host load as a % difference from migration domain average.

Algorithms[14] or Ant Colony Optimization[15] may provide superior performance.

3.3.2 Evaluation Framework

In order to compare algorithms, it is necessary to have some objective framework which can be used to compare the quality of various placement algorithms. Once such a framework exists, it can also be used to judge the results of tuning a single algorithm.

3.3.3 Additional Resources

There are also potential improvements that are algorithm independent. The solver can be enhanced to consider more dimensions of resource consumption. It may be beneficial to consider LAN/SAN reads and writes independently since the underlying media is full duplex. The current solver considers only physical memory, but some virtualization software allow virtual memory for their guests.

3.3.4 Scalability

The current controller architecture is centralized, which causes it to be a single point of failure (workloads would continue to run, but dynamic placement would cease to function) and also raises questions about its scalability. Work is needed to characterize the scalability of the current architecture, and to investigate architectures and algorithms for a distributed solution that would scale to much larger systems and eliminate the single point of failure.

3.3.5 Policies

The current prototype has a load balancing policy which can be used to keep loads on all physical hosts as close to average as can be accomplished. It is possible to create additional policies which are based on higher level business objectives, and these policies could enable the controller to maximize the business impact of the machines it controls. A policy that understood power costs and virtual server service level agreements could choose to violate a service level agreement if the financial penalty for doing so was less than the cost of the power required to meet the agreement.

3.3.6 Additional Constraints

There are a number of additional constraints that could be applied to the evaluation of virtual machine placement, including host evacuation and automatically restarting virtual machines on the best alternate physical hosts. We could also handle more placement constraints, such as sets of virtual machines that must always/never be on the same host, as well as lists of permitted/forbidden hosts for certain virtual machines.

4 Conclusion

Live migration provides a mechanism to very quickly change the mapping of virtual machines onto physical hosts in a way which is transparent to users of the system, and this allows IT organizations to optimize the mapping in ways which are not possible without live migration. The size of the solution space and the desire to allow complex control policies makes manual control of the mapping impractical for all nontrivial problems.

This paper has presented some motivation for why the virtual machine mapping problem is interesting and has shown that it is a computationally difficult problem. An architecture for a system which can be used to generate virtual machine to physical host mappings based upon complex polices was proposed, and a prototype implementation of that architecture was described. Some preliminary results were presented and a number of areas where additional research is needed were described.

References

[1] Hewlett Packard Web Site, *Introducing Integrity Virtual Machines white paper*, http://docs.hp.com/en/9987/Intro_VM_2.1.pdf

[2] VMWare Web Site, *Server Consolidation and Containment* Solutions Brief, http://www.vmware.com/pdf/server_consolidation.pdf

[3] Serdar Cabuk, Chris I. Dalton, HariGovind Ramasamy, Matthias Schunter Trusted Systems Laboratory HP Laboratories Bristol *Towards Automated Provisioning of Secure Virtualized Networks* HP Labs Technical Report HPL-2007-139 September 3, 2007

[4] Christopher Clark, Keir Fraser, Steven Hand, Jakob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt and Andrew Warfield, *Live Migration of Virtual Machines*, in the Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI '05), May 2005, Boston, MA

[5] Michael Nelson, Beng-Hong Lim, and Greg Hutchins, *Fast Transparent Migration for Virtual Machines* Solutions Brief, Proceedings of USENIX 2005 General Track, http://www.vmware.com/pdf/usenix_vmotion.pdf

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebar, I. Pratt, A. Warfield, *Xen and the Art of Virtualization*, Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), October 2003

[7] *Microsoft Windows Server Virtualization News, Support and Training Resources*, http://www.dabcc.com/channel.aspx?id=190

[8] VMWare Web Site, *VMware VirtualCenter 1.3.x Support Documentation, VirtualCenter VMotion Requirements* , http://www.vmware.com/support/vc13/doc/c2vmotionreqs12.html

[9] Hewlett Packard Web site, *VMotion compatible groups*, ftp://ftp.compaq.com/pub/products/servers/vmware/vmmotion-compatibility-matrix.pdf

[10] Steven S. Skiena, *The Algorithm Design Manual*, ISBN 0-387-94860-0

[11] Hewlett Packard Web Site, *HP ProLiant Essentials Virtual Machine Management Pack Overview and Features*, http://h18013.www1.hp.com/products/servers/proliantessentials/valuepack/vms/index.html

[12] Hewlett Packard Web Site, *HP Systems Insight Manager Overview and Features*, http://h18013.www1.hp.com/products/servers/management/hpsim/index.html

[13] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, *Optimization by Simulated Annealing* Science **220** (1983) 671-680

[14] Koza, John R., Martin A. Keane and Matthew J. Streeter, *Evolving Inventions*, Scientific American (February 2003)

[15] Marco Dorigo and Thomas Sttzle, *Ant Colony Optimization*, ISBN 978-0-262-04219-2