Energy-Aware Consolidation for Cloud Computing

Article *in* Cluster Computing · November 2008

CITATIONS
649

READS
1,255

3 authors, including:

Aman Kansal
Microsoft
114 PUBLICATIONS **10,476** CITATIONS

SEE PROFILE

Feng Zhao
Chengdu University
193 PUBLICATIONS **12,275** CITATIONS

SEE PROFILE

Energy Aware Consolidation for Cloud Computing

Shekhar Srikantaiah
Pennsylvania State University

Aman Kansal
Microsoft Research

Feng Zhao
Microsoft Research

Abstract

Consolidation of applications in cloud computing environments presents a significant opportunity for energy optimization. As a first step toward enabling energy efficient consolidation, we study the inter-relationships between energy consumption, resource utilization, and performance of consolidated workloads. The study reveals the energy performance trade-offs for consolidation and shows that optimal operating points exist. We model the consolidation problem as a modified bin packing problem and illustrate it with an example. Finally, we outline the challenges in finding effective solutions to the consolidation problem.

1 Introduction

One of the major causes of energy inefficiency in data centers is the idle power wasted when servers run at low utilization. Even at a very low load, such as 10% CPU utilization, the power consumed is over 50% of the peak power [1]. Similarly, if the disk, network, or any such resource is the performance bottleneck, the idle power wastage in other resources goes up. In the cloud computing approach multiple data center applications are hosted on a common set of servers. This allows for consolidation of application workloads on a smaller number of servers that may be kept better utilized, as different workloads may have different resource utilization footprints and may further differ in their temporal variations. Consolidation thus allows amortizing the idle power costs more efficiently.

However, effective consolidation is not as trivial as packing the maximum workload in the smallest number of servers, keeping each resource (CPU, disk, network, etc) on every server at 100% utilization. In fact such an approach may increase the energy used per unit service provided, as we show later.

Performing consolidation to optimize energy usage while providing required performance raises several concerns. Firstly, consolidation methods must carefully decide *which workloads should be combined* on a common physical server. Workload resource usage, performance, and energy usages are not additive. Understanding the nature of their composition is thus critical to decide which workloads can be packed together. Secondly, there exists an *optimal performance and energy* point. This happens because consolidation leads to performance degradation that causes the execution time to increase, eating into the energy savings from reduced idle energy. Further, the optimal point changes with acceptable degradation in performance and application mix. Determining the optimal point and tracking it as workloads change, thus becomes important for energy efficient consolidation.

This paper exposes some of the complexities in performing consolidation for power optimization, and proposes viable research directions to address the challenges involved. We experimentally study how performance, energy usage, and resource utilization changes as multiple workloads with varying resource usages are combined on common servers. We use these experimental insights to infer optimal operating points that minimize energy usage with and without performance constraints. We then discuss consolidation as a modified multi-dimensional bin-packing problem of allocating and migrating workloads to achieve energy optimal operation. To concretely illustrate this research direction, we present a computationally efficient heuristic to perform consolidation in a simplified scenario. There are many issues that affect consolidation, including server and workload behavior, security restrictions requiring co-location of certain application components, and power line redundancy restrictions. The paper focuses only on a manageable but important subspace spanned by CPU and disk resource combinations. The many issues that need to be addressed to achieve a real-world implementation of such consolidation methods are also discussed to help point out fruitful research directions.

2 Understanding Consolidation

Understanding the impact of consolidating applications on the key observable characteristics of execution, including resource utilization, performance, and energy consumption, is important to design an effective consolidation strategy.

To explore this impact, we used the experimental setup shown in Figure 1. A cloud consisting of $m = 4$ physical servers hosting k controlled applications services requests from controlled clients. Each client generates requests at a desired rate. Each request consists of jobs with specified levels of resource usage on each server resource (here, each server is considered as comprising of two resources: processor and disk). Each server is connected to a power meter (WattsUp Pro ES) to track

power consumption. The Wattsup meter samples power at a maximum rate of 1 Hz, and hence any desired utilization state must be sustained for more than 1 second. We designed the request processing to maintain a uniform utilization state for 60 seconds and averaged the readings over this period to get accurate energy data. The resource utilization is tracked using the operating system's built in instrumentation accessed through the Xperf utility to

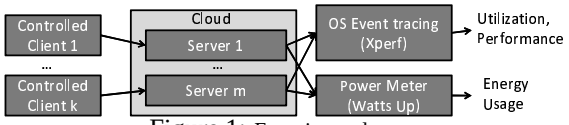


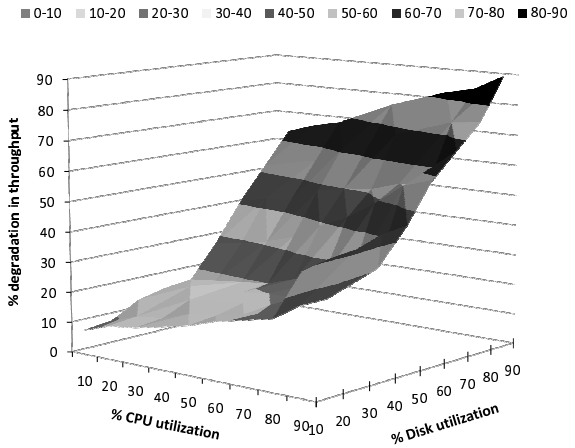
Figure 1: Experimental setup.

Consolidation influences utilization of resources in a non-trivial manner. Clearly, energy usage does not linearly add when workloads are combined, due to a significant percentage of idle energy. But utilization and performance also change in a non-trivial manner. Performance degradation occurs with consolidation because of internal conflicts among consolidated applications, such as cache contentions, conflicts at functional units of the CPU, disk scheduling conflicts, and disk write buffer conflicts.

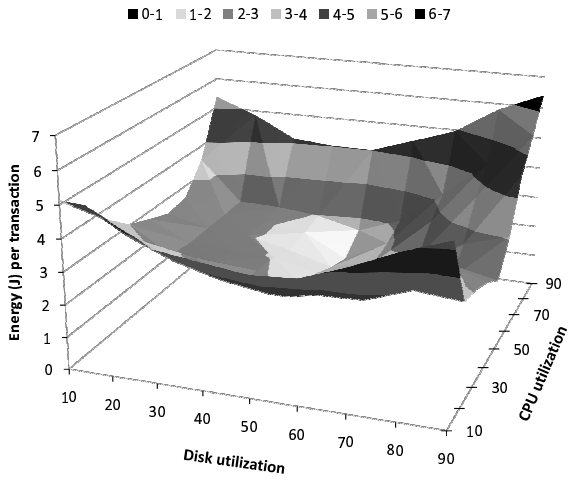
To study the impact of consolidation with multiple resources, we measured performance and energy while varying both CPU and disk utilizations. First, an application with 10% CPU utilization and 10% disk utilization is started. Then, it is combined with workloads of varying CPU and disk utilizations, ranging from 10% to 90% in each resource. The results are plotted in Figure 2(a). The performance degradation observed along the CPU utilization axis is not as significant as that observed along the disk utilization axis implying that increasing disk utilization is a limiting factor for consolidated performance on this server.

Energy consumption *per transaction* of a consolidated workload is influenced by both resource utilization and performance. Typical variation of energy per transaction with utilization (of a single resource) can be expected to result in a "U"-shaped curve. When the resource utilization is low, idle power is not amortized effectively and hence the energy per transaction is high. At high resource utilization on the other hand, energy consumption is high due to performance degradation and longer execution time.

Figure 2(b) plots the energy consumption per transaction of the consolidated workload, for the same scenario. We can make a few important observations from this result. Firstly, energy consumption per transaction is more sensitive to variations in CPU utilization (as seen by the deeper "U" along the CPU axis) than variations in disk utilization as seen by the relatively flat shape of the curve



(a) Performance degradation



(b) Energy consumption

Figure 2: Performance degradation and energy consumption with varying combined CPU and disk utilizations.

along the axis of disk utilization. Secondly, there exists an optimal combination of CPU and disk utilizations where the energy per transaction is minimum. This occurs at 70% CPU utilization and 50% disk utilization for this setup, though the numbers could depend on the specific machines and workloads used. This energy optimal combination may vary if we further impose bounds on the performance degradation tolerable to us, but we can always find an optimal combination of resource utilizations that minimizes energy per transaction taking into consideration both the resources. Note also that this optimal point may be significantly different than that determined by considering each resource in isolation.

3 Consolidation Problem

The goal of energy aware consolidation is to keep servers well utilized such that the idle power costs are efficiently amortized but without taking an energy penalty due to internal contentions.

The problem of loading servers to a desired utilization level for each resource may be naïvely modeled as a multi-dimensional bin packing problem where servers are bins with each resource (CPU, disk, network, etc) being one dimension of the bin. The bin size along each dimension is given by the energy optimal utilization level. Each hosted application with known resource utilizations can be treated as an object with given size in each dimension. Minimizing the number of bins should minimize the idle power wastage. However, that is not true in general, causing the energy aware consolidation problem to differ from traditional vector bin packing:

Performance degradation: Unlike objects packed in bins, the objects here have a characteristic not modeled along the bin dimensions: performance. When two objects are packed together, their performance (such as throughput) degrades. Since performance degradation increases energy per unit work, minimizing the number of bins may not necessarily minimize energy. This aspect must be accounted for in the solution.

Power variation: In a traditional bin packing problem, if the minimum number of bins is n, then filling up $n-1$ bins completely and placing a small object in the n-th bin may be an optimal solution. Here, even given the optimal number of bins, the actual allocation of the objects may affect the power consumed.

Designing an effective consolidation problem thus requires a new solution. The actual implementation must also address several other challenges discussed in section 4. Before discussing those issues, we consider an example heuristic for consolidation, to help illustrate the problem concretely (though not necessarily providing a final or optimal solution).

3.1 Consolidation Algorithm

This algorithm aims to find a minimal energy allocation of workloads to servers. Minimum energy consumption occurs at a certain utilization and performance. However, that performance may not be within the desired performance constraints and hence we design the algorithm to minimize energy subject to a performance constraint. This constraint also helps account for the first difference mentioned above from traditional bin-packing: if we set the performance constraint to that needed for optimal energy consumption, the algorithm will not attempt to minimize the number of bins beyond the number that minimizes power. If the performance constraint is tighter than that required for optimal energy, the number of bins is automatically never reduced below the optimal.

Also, while energy and performance change in a non-trivial manner when workloads are combined, it is worth noting that the resource utilizations themselves are approximately additive, especially at utilizations that are lower than the optimal levels for each resource. Using utilizations as the bin dimensions thus allows our algorithm to operate with a closed form calculation of bin occupancy when combining objects.

The algorithm proceeds as follows: The first step determines the optimal point, from profiling data such as shown in Figure 2(b). Next, as each request arrives, it is allocated to a server, resulting in the desired workload distribution across servers. The crux lies in using a simple, efficient and scalable heuristic for bin packing. The heuristic used here maximizes the sum of the Euclidean distances of the current allocations to the optimal point at each server. This heuristic is based on the intuition that we can use both dimensions of a bin to the fullest (where "full" is defined as the optimal utilization point) after the current allocation is done, if we are left with maximum empty space in each dimension after the allocation. The n-dimensional Euclidean distance (δ_e) provides a scalar metric that depends on the distances to the optimal point along each dimension. If the request cannot be allocated, a new server is turned on and all requests are re-allocated using the same heuristic, in an arbitrary order. This approach is adaptive to changing workloads (temporal variations in requests) as we allocate newer requests to appropriate servers. It also works seamlessly in presence of heterogeneity: in that case the optimal combinations would be different for each type of server.

To illustrate the heuristic, consider the simple example shown in Table 1. The current status of two servers is shown: server A is currently executing at [30, 30], which represents 30% CPU utilization and 30% disk utilization, and Server B is at [40, 10]. A new request with workload requirement [10,10] is to be allocated. Suppose the optimal point for both A and B is [80, 50]. If the application is allocated to A, the sum of the Euclidean distances to the respective optimal points, $\delta_e^A([40,40]-[80,50]) + \delta_e^B([40,10]-[80,50]) = 97.8$, is greater than that achieved by allocating the request to B. Thus the request is allocated to A, leaving A running at [40, 40] and B at [40, 10].

	CPU	Disk	Opt_CPU	Opt_Disk	δ_e	$\sum \delta_e$
A_orig	30	30	80	50	53.8	97.8
A_after	40	40	80	50	41.2	
B_orig	40	10	80	50	56.6	96.2
B_after	50	20	80	50	42.4	

Table 1: Scenario to illustrate proposed heuristic.

Note that an optimal algorithm may find better allocations. In the example above, as a result of the allocation chosen, it is feasible to allocate subsequent requests such as [10, 40], [20, 40], [30, 40], [40, 30], and [40, 40], but not requests such as [50, 10], [50, 20], which would have been feasible if the current request of [10,10] was allocated to B. At each decision point the heuristic chooses an allocation that would leave the system capa-

ble of servicing the widest range of future requests. An optimal scheme however, would choose the allocations based on all requests simultaneously rather than allocating the current request without changing existing allocations. In general, while it is possible to develop better heuristic algorithms, the algorithm presented here serves as an illustration to show that energy savings are possible over the base case. We reserve a more detailed study of the pathological cases for this algorithm (like extremely random disk accesses) for future work.

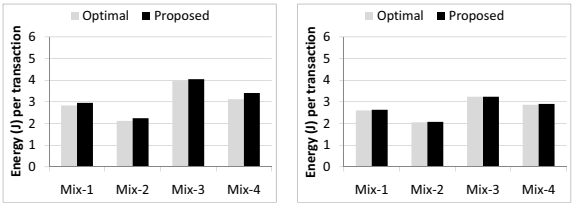
The optimal scheme may have a very high computational overhead, making it infeasible to be operated at the request arrival rate. However, for the purpose of comparison, we implemented the optimal scheme based on an exhaustive search of all possible allocations for a very small cloud consisting of four servers.

	# Apps	Total CPU utilization	Total disk utilization
Mix1	6	84.87	85.86
Mix2	6	93.72	53.87
Mix3	6	78.79	150.58
Mix4	6	91.37	108.92

Table 2: Various mixes of applications considered for our multiprogrammed workload.

3.2 Experimental Evaluations

Table 2 shows the characteristics of various mixes of applications used in the comparison of the example heuristic with the optimal. The application mixes represent a range of consolidated workloads. Mix1 and Mix4 have equal CPU and disk utilizations but vary in the absolute total resource used. Mix2 and Mix3 are skewed toward high CPU utilization and high disk utilization respectively. The energy consumption per transaction of consolidated workloads is experimentally measured using the setup of Figure 1.



(a) $tolerance = 20\%$.

(b) $tolerance = \infty$.

Figure 3: Comparison of our heuristic scheme with the optimal scheme.

Energy per transaction with a maximum performance degradation tolerance of 20% and infinite tolerance are plotted in Figures 3(a) and 3(b) respectively. Clearly, greater energy savings are achieved by both schemes at higher tolerance in performance degradation. The energy used by the proposed heuristic is about 5.4% more than optimal (not considering the energy spent on computing the optimal) on an average at 20% tolerance. It approaches the optimal at infinite tolerance in performance degradation for the test scenario.

4 Discussion

The consolidation problem discussed above helps appreciate some of the key aspects of the performance-utilization-energy relationship. Our ongoing effort in designing a solution has helped uncover several complications, not addressed in the illustrative algorithm above, that lead to a rich set of research issues.

Multi-tiered Applications: Unlike the application described in the previous section, a realistic application may use multiple servers such as a front-end request classification server, a back-end database server to access requested data, an authentication server to verify relevant credentials and so on. Such an application is commonly referred to as a multi-tiered application. Each tier has a different resource footprint and the footprints change in a correlated manner as the workload changes. The consolidation problem involves allocating the multiple tiers of all applications across a subset of available physical servers such that each application may operate at its desired performance level with minimum total energy usage. The allocation may have to be adapted dynamically as workloads evolve over time, either via intelligent allocation of new requests, or via explicit migration of existing state from one physical server to another.

Composability Profiles: We showed how the performance, resource utilization, and energy vary as multiple applications are consolidated on a common server with varying workload serviced. The utilization and performance was tracked using Xperf and power using an external power meter. In a real cloud computing scenario, the overhead of running performance tracking and deploying external power meters may be unacceptable and better alternatives may be needed [4]. Also, measuring the energy-performance relationship itself is non-trivial as the number of applications and the number of possible combinations of their multiple tiers, across multiple server types, may be very large. The use of extra running time for profiling of an application's performance and energy behavior with varying mixes may not be feasible due to cost reasons. Run time variations may even render profiles less relevant. Thus, consolidation research also involves developing efficient methods to understand the composability behavior of applications at run time.

Migration and Resume Costs: While the example in the previous section assumed an application serving small stateless requests, that is not the only type of applications. Other applications may maintain a significant persistent state, such as a long lived chat session [1] or a video streaming session. Such applications involve a non-trivial overhead in migration from one machine to another. Migration may involve initiating the re-

quired configuration and virtual machine needed for the migrating application, incurring additional costs. Also, the dynamic nature of allocation implies that the sleep and wake-up costs of the servers need to be considered. When multiple servers are running at low utilization due to workload dwindling off, their applications may be consolidated to fewer servers allowing some physical servers to be set to sleep mode. However, the migration costs and sleep/wake-up costs may sometimes overshoot the benefit from shutdown of servers. Efficient methods to dynamically adapt to the workload variations with realistic migration and server resume costs for different categories of applications is an interesting problem.

Server Heterogeneity and Application Affinities: We initially assumed that any application may be hosted on any machine. However, in practice, certain applications may have affinities for certain physical resources. For instance, it may be more efficient to run a database server directly on the physical server that hosts the disk drives, since the database software may be optimized for the special read-write characteristics of disks and may directly access the raw storage sectors on the disks. Running the database over network connected storage with additional intermittent layers of storage drivers and network file system abstractions may make it inefficient. Aside from configuration, servers in a cloud facility may vary in energy-performance characteristics due to age of hardware. These variations are crucial to be exploited in consolidation as the energy usage of an application would depend on which server is used and in what application mix. The consolidator may not always run an application on the server most efficient for it since a global optimization is performed. Migration and resume costs may restrict the application from migrating to a more efficient server. Ensuring globally optimal operation in a dynamic and heterogeneous setting is an important problem.

Application Feedback: Our performance-energy study controlled the workload to measure energy and performance, assuming constant quality of service is provided at each performance level. However, certain applications may change their behavior in response to resource availability. For instance, a streaming application may be designed to lower the video streaming rate based on system load, the network stack may have built in congestion control, and so on. Consolidation may reduce resource availability by increasing utilization, incurring an acceptable level of performance degradation to save energy. However, if the application changes its behavior in response to resource availability, such feedback may result in complex behavior that makes it hard to guarantee the performance constraints. Designing consolidation methods for such applications or designing applications to be consolidation-friendly is an interesting research challenge.

Additionally, the example methods discussed here only considered the CPU and disk resources. Other resources such as network and memory should also be considered, as these may be the bottleneck resources for certain applications. Operational constraints such as co-location of certain data and components for security, power line redundancies, and cooling load distribution may affect the design as well. Consolidation methods may need to account for several variations such as the multiple types of requests served by an applications - their expected processing times and the exact set of tiers used to service them. Some of the applications may use external services, not hosted by the cloud operator, further making the relationship more complex.

5 Related work

Consolidation of resources and controlling resource utilizations has been researched in recent times. A framework for adaptive control of virtualized resources in utility computing environments is presented in [5]. In the past, cluster resource management has posed similar problems to that of consolidation of virtualized resources. For instance, [7] proposes a two level resource distribution and scheduling scheme, and [8] proposes a cluster level feedback control scheme for performance optimization. Power aware load balancing and migration of activity within a processor, among multicore processors and servers has also been studied in [3, 2, 6]. However, there has been little work on joint power and performance aware schemes for multi-dimensional resource allocation. Specifically, characterizing power-performance characteristics of consolidated workloads with respect to multiple resources has not been adequately addressed.

References

[1] CHEN, G., et al. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *NSDI* (2008).

[2] GOMAA, M., et al. Heat-and-run: leveraging smt and cmp to manage power density through the operating system. *SIGOPS Oper. Syst. Rev. 38*, 5 (2004).

[3] HEO, S., BARR, K., AND ASANOVIĆ, K. Reducing power density through activity migration. In *ISLPED* (2003).

[4] KANSAL, A., AND ZHAO, F. Fine-grained energy profiling for power-aware application design. In *Workshop on Hot Topics in Measurement and Modeling of Computer Systems* (2008).

[5] PADALA, P., et al. Adaptive control of virtualized resources in utility computing environments. In *European Conference on Computer Systems* (2007).

[6] RANGANATHAN, P., et al. Ensemble-level power management for dense blade servers. In *ISCA* (2006).

[7] SHEN, K., et al. Integrated resource management for cluster-based internet services. *SIGOPS Oper. Syst. Rev. 36*, SI (2002), 225–238.

[8] WANG, X., AND CHEN, M. Cluster-level feedback power control for performance optimization. In *HPCA* (2008).