Energy Aware Consolidation in Cloud Computing

April 8th, 2015

Adrian J. Mirabel and Rashid Siddiqui

Overview

- Introduction
- Def. of Consolidation
- Challenges in Consolidation
- Experimental measure
- Method Description
- Method Example
- Analysis of Method
- References
- Questions

What is Cloud Computing?

- Focuses on maximizing the effectiveness of the shared resources.
- Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand.
- With cloud computing, multiple users can access a single server to retrieve and update their data
- No need for purchasing licenses for different applications.
- Based on advances in virtualization and distributed computing
- Supports cost-efficient usage of computing resources
- Emphasizes on resource scalability and on demand services.
- Energy Aware Consolidation is consolidating while minimizing energy consumption.

Energy Inefficiency in Data Centers are caused by:

- idle power wasted when servers run at low utilization.
  - ex) 10% CPU utilization can consume more than 50% of peak power (100% CPU utilization)
- Disk, network, or any such resource contention causes performance bottlenecks.
  - causes idle power wastage in other resources.

What is Consolidation?

- Running many dissimilar client applications on the same server cluster.
- In other words running multiple data center applications on a common set of servers.
- This allows for the consolidation of application workloads on a smaller number of servers that may be kept better utilized.

Analysis of Problems of Consolidation

- Effective consolidation is not as trivial as packing the maximum workload in the smallest number of servers.
- Keeping resources at 100% utilization is not energy efficient.
- Goal is to minimize the energy used per unit service.
- Use coefficient of performance to measure efficiency $COP = Q/W$
  - where Q is energy supplied to the system.
  - where W is the work consumed by the system.

- Experiment to verify:
  - Power consumption vs. resource utilization relationship.
  - Performance vs. resource utilization relationship.
- Setup:
  - m = 4, servers. With k clients running many client applications with varying CPU and disk utilizations.
  - Client applications are mock apps, with a uniform resource footprint and execution time (60s).
  - CPU utilization is sampled at a rate of Hz.

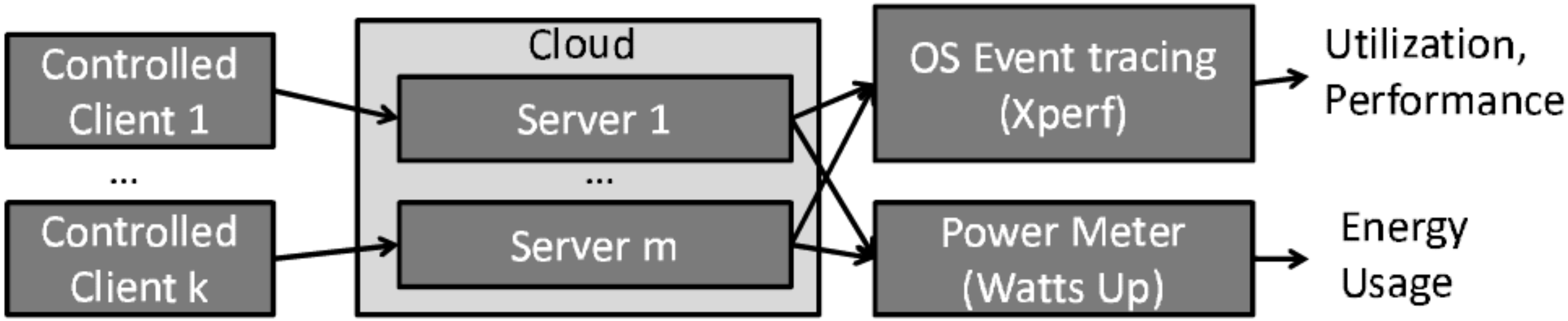
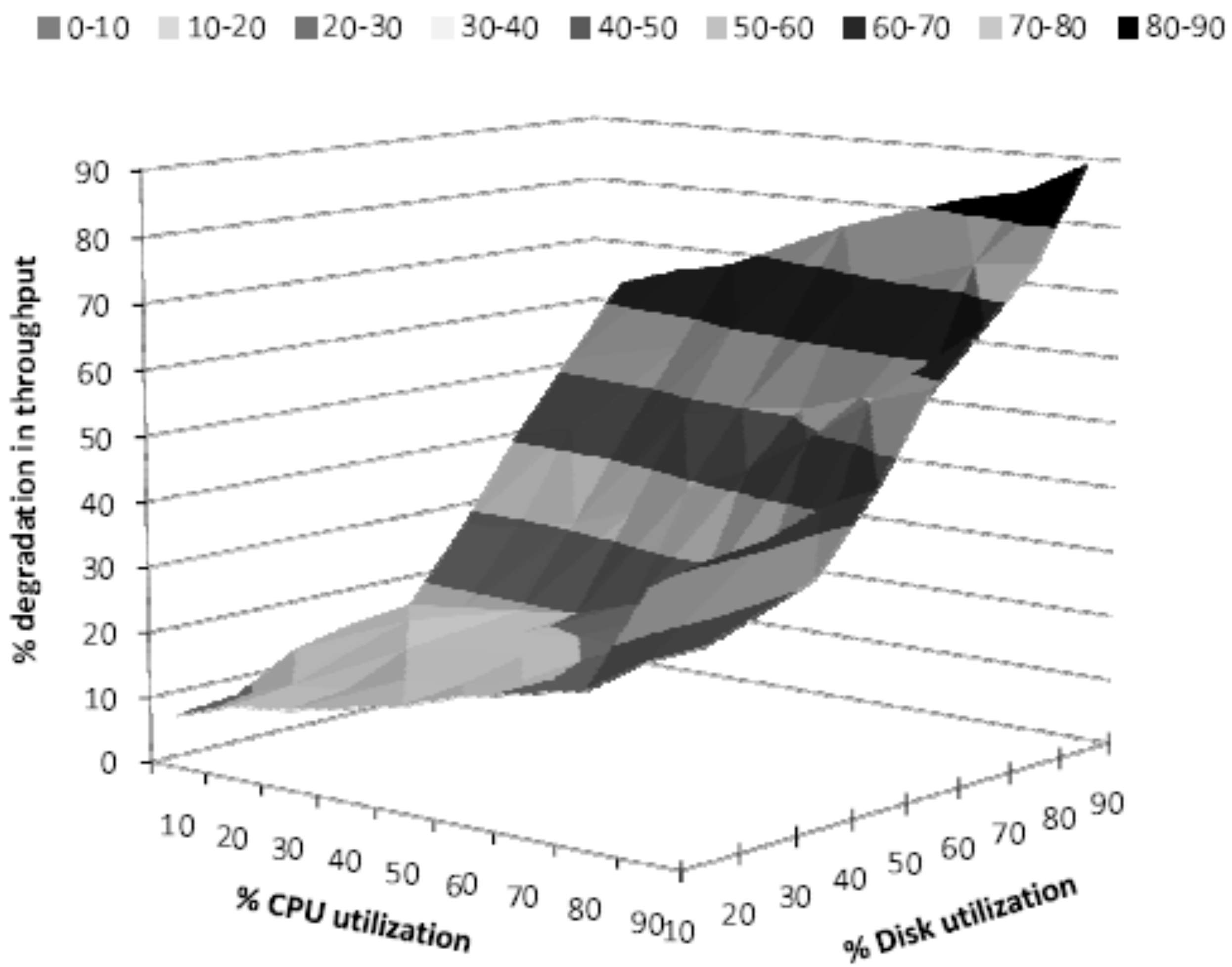


Figure 1: Experimental setup.

Performance vs. Resource Result



The figure shows the performance (throughput) degradation with varying CPU and disk utilizations.

Energy vs. Resource Result

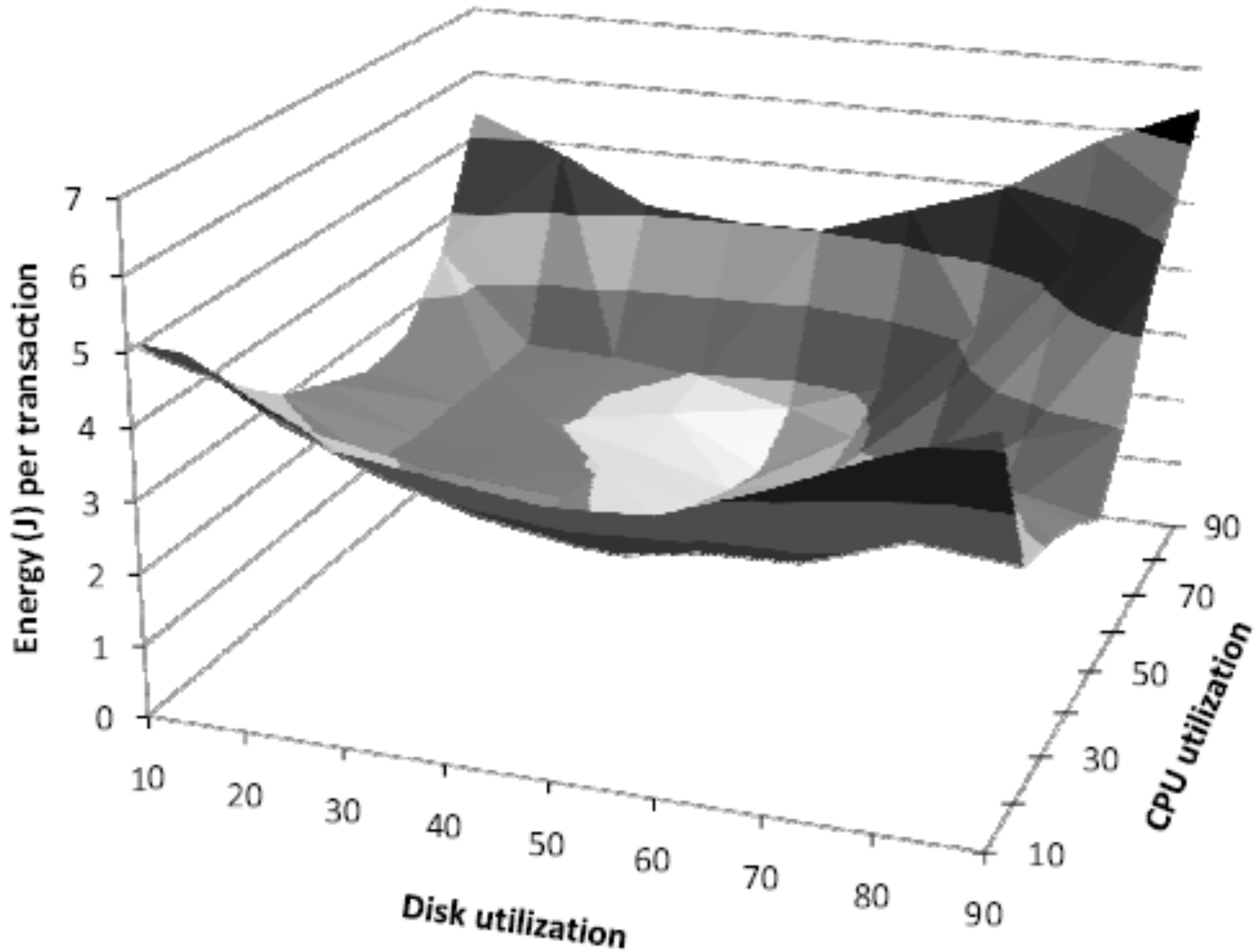


Figure shows the energy consumption for varying combined CPU and disk utilization

- Degradation is more sensitive to disk usage, than CPU usage.
  - implies that increasing disk utilization is the limiting consolidation factor on these server.
- Energy per transaction vs resources relationship is paraboloid
  - in general for any resource it is a shifted quadratic relationship.
- Energy per transaction is more sensitive to CPU utilization.
- Optimal combination of CPU and disk utilization that minimizes energy per transaction occurs at approx. 70% CPU utilization and 50% disk utilization for these servers
- Adding constraints shifts the optimal resource point.

Method Requirements for Optimization

- 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.
- There exists an optimal performance and energy point.
- Consolidation leads to performance degradation that causes the execution time to increase, eating into the energy savings from reduced idle energy.
- 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.
- Performance Degradation: Generally as many client applications are run in the same cluster, they will cause a performance degradation.
- A reduced performance means applications take longer to run and increase their energy per unit work.

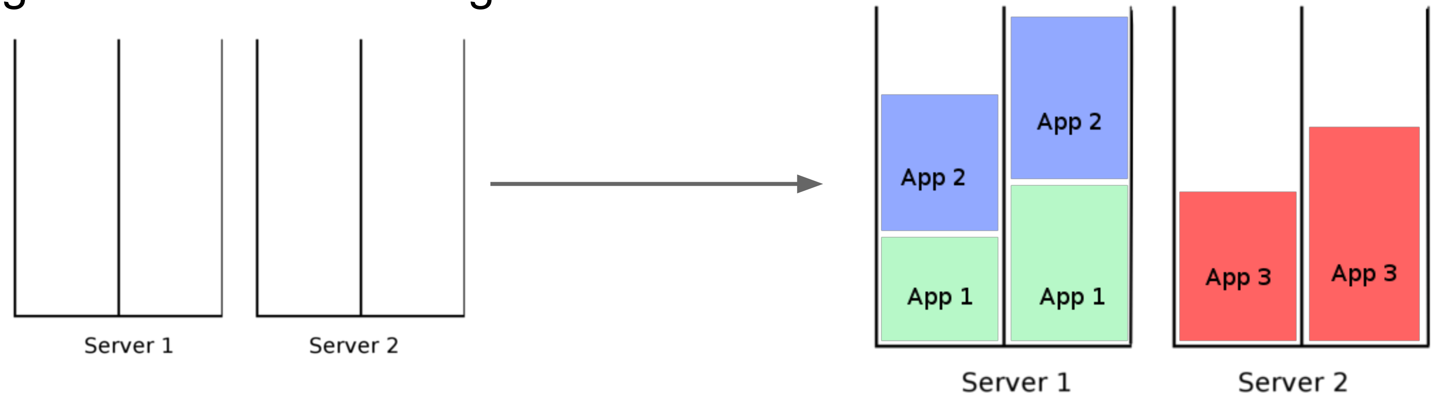
- Generally the method proposed is an algorithm that allocates incoming client applications to specific servers in an optimal manner.
- Prior to using the method, the energy vs. resource relationships needs to be empirically determined for each server type.
  - Used to determine the optimal energy points R(CPU%,HD%,...)
- The method proposed is meant only as a proof of concept and needs additional work before being utilized in a production environment.

General Method Steps

1. Determine optimal resource points from profiling data for each server type used.
2. Allocate incoming client applications according to the Allocation Algorithm.

System Model - Multidimensional Bin Packing

- The method, describes the systems servers as bins, with each resource being one dimension of the bin.
  - The bin size along each dimension is given by the energy optimal utilization points.
- Each client application is modelled as an object that occupies a given size in each dimension.
- After this modelling the goal is to then place all the objects (client apps) into the bins (servers), while using the minimum number of bins.
- In order to find the sequence of object placements, the problems state space is searched using a heuristic search algorithm.



Search Methods - Greedy

- The search algorithm used is a Greedy First-Fit, where the client application is assigned to the best available server from the available pool.
- The authors also specify an Exhaustive Search algorithm, that finds the optimal sequence of client application to server placements.
    - This algorithm is only used to validate the greedy algorithm.

- Let $\delta_e = \sqrt{(x_1^2 + x_2^2 + .. + x_n^2)}$ be the euclidean distance between two resource points.
  - ex) $\delta_e([20,30] - [40,40]) = \sqrt{((-20)^2 + (-10)^2)} = 22.361$
- Each server has a optimal resource point given by s* = [CPU*, HD*]
  - ex) s* = [20,30], which means that s_i has optimal point at 20% CPU and 30% hard disk utilization.
- Each workload has a resource footprint w = [CPU, HD]
  - ex) w = [10,10], so workload w, uses 10% CPU and 10% of hard disk.

Allocation Algorithm

If w is a workload that needs to be allocated:

1. Let score[i] be the sum of distances for allocating the workload to the i^{th} server.
2. For every server available, s_i do the following:
   a. Let $s_i' = w + s_i$;
   b. IF $s_i' > s^*$
      i. THEN we try next server, or wake up a new server.
   c. ELSE
      i. score[i] = $\delta_e(s_i' - s^*) + \sum_{j \neq i} \delta_e(s_j - s^*)$
3. Allocate w to s_i where i is the index of the largest sum in score.

- Consider two active servers, server A running at [30,30] (30% CPU, 30% HD) and sever B running at [40,10].
- Assuming each server has an optimal resource point s* of [80,50].
- We have a workload w = [10,10] that needs to be allocated

First we try adding the workload to server A:

$s_a' = w + s_a$

Then we compute the score for this allocation

$score[a] = \delta_e(s_a' - s^*) + \sum_{j \neq a} \delta_e(s_j - s^*) = \delta_e(s_a' - s^*) + \delta_e(s_b - s^*) = 97.8$

Next we try adding workload to server B:

$s_b' = w + s_b$

$score[b] = \delta_e(s_b' - s^*) + \sum_{j \neq b} \delta_e(s_j - s^*) = \delta_e(s_b' - s^*) + \delta_e(s_a - s^*) = 96.2$

Now we allocated the workload to the server with maximum score, which is server A.

	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	

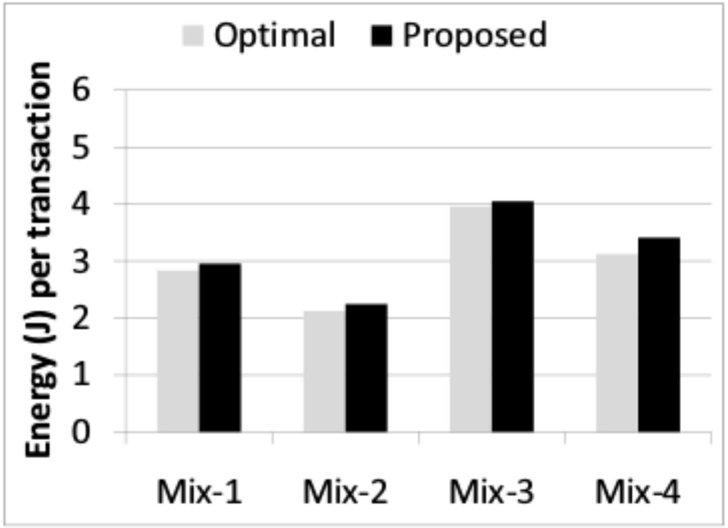
Algorithm Validation Experiment

- In order to validate the proposed method, the authors ran the proposed algorithm against an Exhaustive algorithm that found the optimal sequence of allocations, using 4 different client application mixtures.
- The exhaustive algorithm finds the optimal sequence of object (client app) to bin (server) placements.
- The proposed method uses the allocation algorithm.

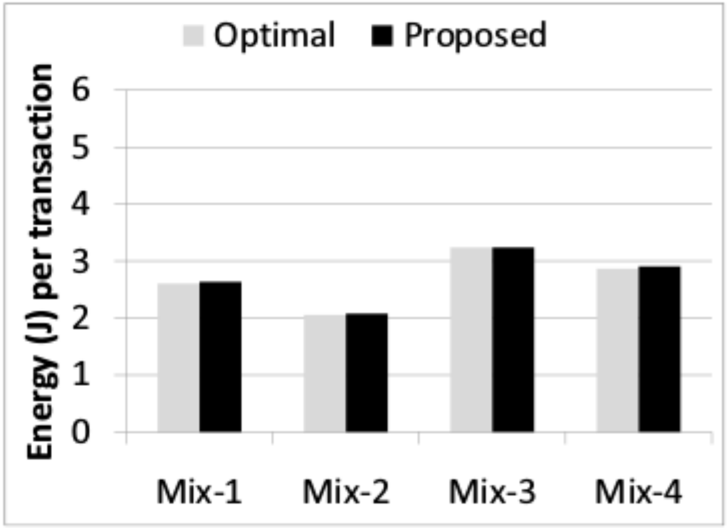
	# Apps	Total CPU utilization	Total disk utilization
Mix1	6	84.87	85.86
Mix2	6	93.72	53.87
Mix3	6	78.79	150.58
Mix4	6	91.37	108.92

Algorithm Validation Results

- The tolerance, is the allowed performance degradation constraint.
- The optimal method is less efficient than the proposed.
    - This odd results is due to inaccuracies in how effective bin packing is at modeling the problem.



(a) $tolerance = 20\%$.



(b) $tolerance = \infty$.

Limitations of the Approach

- This approach makes many idealizations and approximations, such as using mock client applications with constant resource utilizations and execution time.
- Multi-tiered Applications: realistic applications consist of many smaller apps that run on different servers in coordination and have different resource footprints.
- Dynamic Resource Footprint: realistic applications do not have uniform resource footprints.
- Composability Profile: Determining the optimal resource points for server(s), is difficult since it is hard to obtain accurate CPU utilization data from servers running realistic applications.
- Migration Costs: real world applications can run persistently on a set of servers for long periods of time, incurring additional costs when they need to be migrated.
- Server Heterogeneity and Application Affinities: Not all client applications can be hosted on any server, some servers and apps have special requirements.
- Application Feedback: some applications tailor the resource utilizations in accordance with available resources.

References

- Srikantaiah S et al. (2008). Energy aware consolidation for cloud computing. In: Proc of HotPower

Questions?

?