On the Use of Linear Programming in Optimizing Energy Costs

Fahad Javed[1] and Naveed Arshad[1]

LUMS School of Science and Engineering Lahore, Pakistan
`fahadjaved@lums.edu.pk` `naveedarshad@lums.edu.pk`

Abstract. Efficient energy consumption in large sets of electric devices is a complex problem since it requires a balance between many competing factors. Presently, self-optimization techniques work expeditiously on small and relatively less complex problems. However, these techniques are not shown to be scalable on large and complex problems. In this paper we have used linear programming to optimize the use of energy in a typical environment that consists of large number of devices. Our initial results show that LP is fast, predictable and scalable. Moreover, we have also observed that modeling in LP is quite simple as compared to other self-optimization techniques.

1 Introduction

Self-optimization, the goal of enabling a system to autonomically optimize itself, necessitates a methodology that is able to handle input domain for any given system. This requires an ability to handle a hyper-dimensional variable space involving hundreds of variables with complex relationships.

Some of the recent approaches for self-optimization problems have used traditional methods such as control theory [5] [3] to optimize a given system. However these solutions are limited as it has not been shown that these techniques can scale to hyper-dimensional variable space or handle the complex relationships appropriately.

In contrast our interest centers around the problems of self-optimization for large and complex systems involving hyper-dimensional input and output variable space with complex relationships. We approach this problem using linear programming to find the extremum of the system.

In this approach we have used linear programming to optimize the power consumption of a heterogenous system of machines under variable demand. Our initial results have shown upto 80% savings.

Our contributions in this paper are:

1. A scalable methodology for self-optimizing a system which involves hyper-dimensional variable input and output space as long as the system is linear or can be interpolated to linear domain.
2. A unique time-variate modeling schema for planning with linear programming.
3. Identification of a class of problems where optimization techniques such as linear programming could be used.

2 Related work for Cost Optimization

Efficient energy consumption is a critical issue that has become the focus of academia, industry and general public in recent times. This has roots in ecological as well as cost management issues. Electric consumption for commercial entities such as cyber-cafes, server farms and other facilities where a considerable number of computers are installed face problem of optimization of cost of operations.

A typical cost of operations in a lab has two mostly conflicting components. The first component is the energy cost incurred by running the machines. The second component is the cost of repairs. Both these costs have a weekly inverse relationship. As it is observed that frequent restarts of machines is one of the most common cause for breakages in machines. Labs, including managed by our own IT department, would rather keep the machines "on" rather than pay for costly breakages. The machines are scheduled to go to "hibernate"mode if machine is the machine is not used for a given period of time. However "hibernate" cost is usually around 20% of a typical "on" cost. There is no methodology available that balances the growing cost due to hibernate with the cost of breakages for a more robust systrm.

Optimizing a system consisting of homogenous machines with or without indistinguishable power consumption profiles has been proposed in previous work [4], [3] [5] . But our target system, computer labs usually employ computers with varying configurations, demands and power profiles. Therefore we need a more robust approach that is able to handle the extra requirements.

Our technique for optimization is a balance between [1] and [2], [4]. Almeida and colleagues provides a heuristics based mathematical technique which is scalable but does not guarantee an optimum solution and is extremely complex to grasp[1]. In contrast Nathuji and colleagues used a simplistic greedy algorithm to balance power in a heterogeneous data-center environment[4]. This was possible due to the nature of problem which allowed mapping of input to a single variable domain thereby making greedy algorithm a possibility. Although Femal and Freeh used LP to derive an optimal solution for boosting performance[2], it can be shown that a greedy technique would have been more suitable to find the solution.

We felt that a sizeable number of problems in the autonomic computing domain can be handled by a far less complex system than [1] and with a much better guarantee. At the same time not all problems can be mapped to an input domain which is solvable through greedy algorithms. Our optimization technique targets the class of problems that lies between these two extremes.

3 Approach

Our approach to reduce the operation costs of an environment which consists of large number of machines is based on a hypothesis. This hypothesis states that although a restart cycle has a cost component, and a "hibernate" and "on" also has a cost component. However, there is a an optimum point which balances these two competing cost components. In our approach we take this optimum point to be the reduction of the total cost of operations of the environment.

To model such a system our input domain includes various classes of machines with their demands and costs. Our output domain is the number of machines in "on" "off" or "hibernate" state for each time period. We applied linear programming (LP) to our problem to find the optimum utilization of such an environment based on the reduction of the total cost of operations.

3.1 Modeling in Linear Programming

An LP model requires a series of linear equations constraining the problem domain and an optimization function. Our cost function defines the optimization function and the constraint equations are discussed shortly.

Our system consists of various classes of machines. Each machine has states and each state carries a cost per unit time. Our explicit objective is to reduce cost but implicitly we require a solution that meets our demands for each time period and also satisfies the relational constraints such as the demands for a particular class of machines. The variables for our model are given in figure 1.

$ON_{i_{tj}}$	i type of machine on at time j
$OFF_{i_{tj}}$	i type of machine off at time j
$HIBERNATE_{i_{tj}}$	i type of machine hibernating at time j
$SWITCHON_{i_{tj}}$	i type of machine to be switched on at time j

Fig. 1. LP Variables

We will achieve optimization by reducing the two cost components over a period of time. LP selects the least cost combination of machines which will satisfy the requirements (equation 1). The decision that we are interested in is which machines to shut down to minimize the cost. The requirement here are the demands for each usage class for a specific time period (equation 3). We are bounded by the number of machines for each class (equation 2). We put it all together, the "on", "off", "hibernate" and "switch-on" costs, in our optimization function as given in figure 2 equation 1. To complete the model we added the non-negativity constraints for all the variables (not shown in the figure). This is because in our system a negative *on* or off machine does not make sense .

4 Evaluation

We ran several simulations using equations in figure 2 on data collected from one of our labs and compared it with simulated cost of other techniques available. We ran our tests on a week's data. We considered a window of 24 hours and compared our LP results with the existing setup. In our tests we assumed that we will have knowledge of usage for each class priori.

4.1 EnergySaver++

Our application, EnergySaver++, uses LP to provide a plan for the number of machines to shutdown or hibernate in each 1 hour period over a period of 24

$$z = \sum_{i,j} ON_{i_{tj}}.cost_{On_i} + \sum_{i,j} HIBERNATE_{i_{tj}}.cost_{Hibernate_i} + \sum_{i,j} SWITCHON_{i_{tj}}.cost_{Restart_i} \quad (1)$$

$$\forall_{i,j}\{ON_{i_{tj}} + HIBERNATE_{i_{tj}} + OFF_{i_{tj}} = supply_i\} \quad (2)$$

$$\forall_{k,j}\{ \sum_{i:X_i \subset S_i} ON_{i_{tj}} \geq demand_{ktj}\} \quad (3)$$

$$\forall_i \forall_j \{ON_{i_{tj+1}} + HIBERNATE_{i_{tj+1}} - ON_{i_{tj}} - HIBERNATE_{i_{tj}} \leq SWITCHON_{i_{tj}}\} \quad (4)$$

Fig. 2. LP Equations

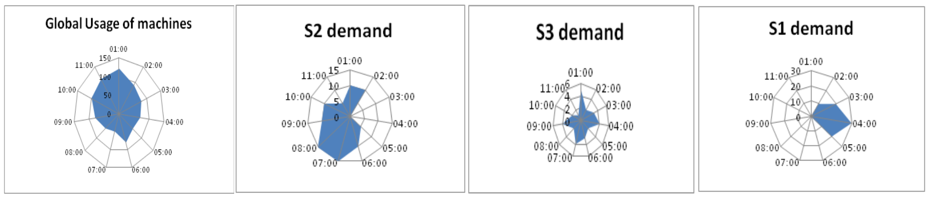

Fig. 3. Demand pattern

hours. We classify the machines in our lab according to their power consumption profile and configuration of each system.

The power consumption profile is derived by calculating the power needed for the CPU, monitor and periphery devices for the three states concerned, namely: *On*, *Hibernate* and the breakage cost due to restart. Breakage cost of restart is calculated by finding the average cost of repair for machines which were diagnosed with power fluctuation related faults divided by number of times an average machine was rebooted in the past one year.

Our second classification criterions, for configuration classes, are the special software or hardware installation on machines for which students make special demands. Due to various issues some facilities are available only at specific number of machines. We classified machines based on these special configurations and measured the usage of the "special" resource for each system configuration class.

Using the above two classification methods, we define our set of general classes X for LP. X is defined as partitions of U created by C and intersections of C and S. Here U is the set of all the machines, $\{C_i \subset C : C_i$ is a consumption class$\}$ and $\{S_i \subset S : S_i$ is a usage class$\}$ Consumption classes are disjoint classes as a computing can belong to only a single profile. Whereas usage classes can overlap one or more consumption classes.

In our initial test-bed consists of 120 machines. There were 6 different power consumption profiles for those machines. We identified 3 configuration classes. For LP model, with overlapping of consumption classes, we had 10 (X) classes.

We collected historical data to predict the workload. Table 1 defines the cost for each class for each of the 6 consumption profile classifications.

We collected the usage of systems by noting the logon/ logoff times for machines. A second script noted if a special resource was used in the last hour. We observed that the usage of machines was highly cyclic. In-fact the usage repeats itself after 7 days. We predicted demand for the 24 hour period for each time period as shown in Fig 3. The global demand is for a general purpose machines where as demands $S1$, $S2$ and $S3$ corresponds to the demand for Matlab, SPSS, and scanner respectively for each time period.

We evaluated our system by comparing the cost of operations incurred by our system with the cost incurred by the three plans provided by OS vendors. Namely: Keep machines always in on state or Hibernate machines after x minutes ($x < 60$) or Switch-off machine if machine has been in hibernate state for 1 hour.

We used data for 7 days of a week and applied LP based planning. We simulated the three plans that are used for optimizing power consumption on the same 7 day dataset. Table 2 shows the result of the cost of operations that these systems incurred given the same usage pattern. The table also shows, in savings column, the percentage savings that our LP based system attains in comparison.

We observed upto 80% savings in comparison to an always on policy for weekend days (day 6 & 7) and upto 43.3% and 53.4% savings for hibernating plan and switching off after one hour of hibernate plans, respectively.

For weekdays(day 1 - 5), we saw a maximum of 62% savings for always on and up to 19% and 32% savings for hibernating plan and switching off after one hour of hibernate plans respectively.

Since our lab was using policy where the machines switched to hibernate, our net savings were 2.43 units or 24%. This means that we are able to slash our costs by 1/4th by using an LP based planner.

Our reason for using LP was scalability. To test our claim that LP is scalable, we evaluated our application against a variety of input parameters. We generated random data to set up scenarios. Our variation in data can come from change in number of machine and change in the classes of machines. These parameters and their results are shown in table 3. It could have been inferred from the LP equations themselves that a change in the number of machines only will not effect the running time. We can see this in our simulations as well. The time for 100, 1000, and 10000 machines is nearly same for the different classes.

State	On						Hibernate						Restart					
Class	C1	C2	C3	C4	C5	C6	C1	C2	C3	C4	C5	C6	C1	C2	C3	C4	C5	C6
Cost	10	5	10	15	20	20	2	1	4	2	3	2	10	16	18	18	10	17

Table 1. Cost function of machine classification

Plan	day1		day2		day3		day4		day5		day6		day7	
	Cost	Savings	Cost	Saving	Cost	Saving	Cost	Saving	Cost	Saving	Cost	Saving	Cost	Saving
Alwayon	3.36	62%	3.36	61%	3.36	61.6%	3.36	62.0%	3.36	62.2%	3.36	80%	3.36	81%
Hibernate	1.56	19%	1.58	18.2%	1.57	18.3%	1.56	18.7%	1.56	18.9%	1.13	42%	1.11	43.3%
1 hour off	1.86	32%	1.89	31%	1.88	31.6%	1.86	31.8%	1.87	32.4%	1.36	52%	1.36	53.4%

Table 2. Daily cost of operations for alternative methods and percentage saving in comparison to LP method

Classes	10			50			100		
Machines	100	1000	10000	100	1000	10000	100	1000	10000
Running Time	0.23	0.12	0.13	0.5	0.53	0.51	1.48	1.38	1.38

Table 3. Scalability results of LP (time in seconds)

5 Future work and conclusion

In this paper we have provided the groundwork for a possible future direction in self-optimization. We have described the issues with using traditional methods for self-optimization.

We have provided an alternative approach for self-optimization by using conventional mathematical techniques. As a sample we have described a hyper-dimensional problem, EnergySaver++, and applied linear programming to derive a solution which is optimal and is derived in polynomial time.

Our future direction is two pronged. First we will build on this work to formulate a better way to handle prediction errors and current load transitions by using some iterative methods on LP. Second, we will try to provide a solution for non-linear time variant system planning.

Our targeted system for these future direction is to manage the power supply at the metropolitan level which will optimize the power consumption while maintaining a basic quality level for the users.

References

1. J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, and M. Trubian. Resource management in the autonomic service-oriented architecture. *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, pages 84–92, 13-16 June 2006.
2. M.E. Femal and V.W. Freeh. Boosting data center performance through non-uniform power allocation. *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on*, pages 250–261, 13-16 June 2005.
3. Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. Server-level power control. *Autonomic Computing, 2007. ICAC '07. Fourth International Conference on*, pages 4–4, 11-15 June 2007.
4. Ripal Nathuji, Canturk Isci, and Eugene Gorbatov. Exploiting platform heterogeneity for power efficient data centers. *Autonomic Computing, 2007. ICAC '07. Fourth International Conference on*, pages 5–5, 11-15 June 2007.
5. Mianyu Wang, N. Kandasamy, A. Guez, and Moshe Kam. Adaptive performance control of computing systems via distributed cooperative control: Application to power management in computing clusters. *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, pages 165–174, 13-16 June 2006.