Algorithms for Task Consolidation Problem in a Cloud Computing Environment

Amandeep kaur
Eternal University, Baru Sahib
H.P, India

Rupinder kaur
Eternal University, Baru Sahib
H.P, India

Prince Jain
Punjab Technical University,
Jalandhar
Punjab, India

ABSTRACT
Cloud computing has recently emerged as a new paradigm for hosting and delivering services over the Internet. Task consolidation problem in cloud computing systems became an important approach to streamline resource usage which improves energy efficiency. The task consolidation is also known as workload consolidation problem which is the process of assigning set of tasks to set of resources without violating time constraints. Three existing energy conscious heuristics such as ECTC (Energy-Conscious Task Consolidation) Task Consolidation Algorithm and MaxUtil (Maximum rate Utilization) Task Consolidation Algorithm and Bi-objective Task Consolidation algorithm offering different energy saving possibilities were analyzed in this study. The cost functions incorporated effectively capture energy saving possibilities and their capability has been verified by evaluation study. The Bi-objective Task Consolidation algorithm combines the two heuristics to construct the corresponding bi-objective search space. The efficiency of proposed algorithm was proved thought evaluation study consisting of different simulations carried out.

General Terms
Cloud Computing, Task Consolidation algorithms

Keywords
MaxUtil, ECTC, Bi-Objective Algorithms

1. INTRODUCTION
Cloud computing has recently emerged as a new paradigm for hosting and delivering services over the Internet. Cloud computing is attractive to business owners as it eliminates the requirement for users to plan ahead for provisioning, and allows enterprises to start from the small and increase resources only when there is a rise in service demand. With the rapid development of processing and storage technologies and the success of the Internet, computing resources have become cheaper, more powerful and more ubiquitously available than ever before. This technological trend has enabled the realization of a new computing model called cloud computing, in which resources such as CPU and storage are provided as general utilities that can be leased and released by users through the Internet in an on-demand fashion [1].

A cloud typically consists of multiple resources possibly distributed and heterogeneous. However, recent advances in virtualization technologies in particular have made it much more compelling compared to the time when it was first introduced. The adoption and deployment of clouds has many attractive benefits, such as scalability and reliability; however, clouds in essence aim to deliver more economical solutions to both parties: consumers and providers.

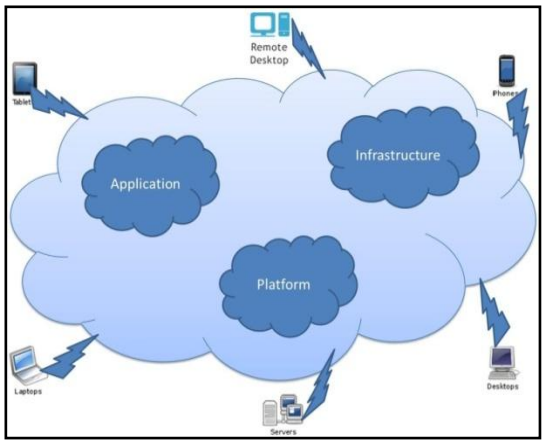

Figure 1: A cloud computing network

By economical it means that consumers only need to pay for what resources they need while providers can capitalize poorly utilized resources. From a provider's perspective, the maximization of the profit is a high priority. In this regard, the minimization of energy consumption plays a crucial role. Moreover, energy consumption can be much reduced by increasing resource utilization. Energy usage in large-scale computer systems like clouds also yields many other serious issues including carbon emissions and system reliability [2, 10]. The first academic use of term Cloud Computing appears to originally suggested that this would be a new computing paradigm where the boundaries of computing will be determined by economic rationale rather than technical limits alone[7].

To better exploit the elastic provisioning of Clouds, it is important that Cloud application Developers, before deploying an application in the Cloud, understand its behaviour when subject to different demand levels. This allows developers to understand application resource requirements and how variation in demand leads to variation in required resources. This information is paramount to allow proper Quality of Service (QoS) to be set and to estimate the budget required to host the application in the Cloud [4, 13].

The most recent estimates for U.S. data centres suggest that between 2000 and 2006, their electricity demand more than doubled to approximately 61 billion kilowatt-hours (kWh) or to around 1.6% of 2006 U.S. electricity sales . The rapid rise and growing national significance of this electricity demand has placed increased attention on strategies for improving the energy efficiency of data center operations. The rapid rise and growing national significance of this electricity demand has placed increased attention on strategies for improving the energy efficiency of data center operations. The assessment resulted in a 2007 peer-reviewed report to the U.S. Congress

containing projections of U.S. data center energy demand under different efficiency scenarios. The EPA study also contained policy recommendations for promoting greater data center efficiency [5, 11, 12, 14].

A distributed computing system is a network of computing nodes that interact with each other in order to achieve a common goal. Distributed Cyber Physical Systems (DCPS's) are distributed systems in which computing systems interact with physical environment based on information from the physical and cyber space. Green computing generally refers to the efficient use of resources in computing in conjunction with minimizing environmental impact, and maximizing economic viability. Similar to any other resource problem the goal of green computing is

1. Use fewer hazardous materials
2. Maximize the efficiency of all resource use in computing systems during their lifetime
3. Reuse as many resources as possible and to dispose what cannot be recycled responsibly.

Research and industry continue to derive forward green computing paradigms such as making the use of computer as energy efficient as possible. Such solutions can be applied in different levels of computing systems from low hardware level such as low power electronics to the high software level such as scheduling algorithms. The goal is to incorporate effective green computing parameters in decision making for workload assignment and power management of computing nodes in DCP's. Green computing is important in such systems to increase scalability and sustainability. Green DCP's designers must try to find the optimal balance of energy-latency trade-off, where the system performance is sacrificed to gain energy efficiency. Only in the case where the benefits of green computing such as increased lifetime, increased system reliability, lower cost of ownership, and improved safety outweigh the cost on system performance are such solutions likely to be used [3].

The field of cloud computing uses different management techniques for data center virtualization such as OpenNebula. However, computers composing the cloud infrastructure use a significant and growing portion of energy in the world specifically when dealing with virtualization for high performance computing (HPC). Therefore, energy-aware computing is crucial for large-scale systems that consume considerable amount of energy [6, 15].

2. MODELS

2.1 Energy Model

The energy model is based on the fact that processor utilization has a linear relationship with energy consumption. The proportional relationship means that, for a particular task, the information on the processing time and the processor utilization is sufficient to measure the energy consumption for the task. At any given time, for a resource r_i, the utilization Ui is defined as

$$U_i = \sum_{j=0}^{n-1} u_{i,j} \quad \text{..... (1)}$$

Where n is the number of tasks running at the given time and $u_{i,j}$ is the resource usage of a task t_j. The energy consumption E_i of a resource r_i at any given time is defined as

$$E_i = (p_{max} - p_{min}) \times U_i + p_{min} \quad \text{..... (2)}$$

Where p_{max} is the power consumption at the peak load or 100% utilization and p_{min} is the minimum power consumption in the active mode or as low as 1% utilization. Consequently, at any given time, the total utilization (U_R) as the total energy consumption (E_R) of the system are defined as

$$U_R = \sum_{i=0}^{m-1} U_i \text{ and } E_R = \sum_{i=0}^{m-1} E_i \quad \text{.....(3)}$$

Where m is the number of resources used. The resources in the underlying system are assumed to be incorporated with an effective power-saving mechanism for idle time slots. The mechanism results from the significant difference in energy consumption, between active and idle resources states. Specifically, the energy consumption of an idle resource at any given time is set to 10% of p_{min}. Because the overhead to turn off and back on a resource takes a nonnegligible amount of time, the option for idle resources was not considered in our study or by others.

2.2 Cloud Model

The underlying system consists of a set $R = \{r_{0,\ldots,}r_{m-1}\}$ of m resources that are fully interconnected in the sense that a route exists between any two resources. It is assumed that resources are homogeneous in terms of computing capability and capacity. The aforementioned is achieved through the virtualization technologies. Nowadays, as many core processors and virtualization tools are commonplace. The number of concurrent tasks on a single physical resource is loosely bounded and a cloud computing can span across multiple geographical locations. The cloud computing model I consider is assumed to:

1. Be confined to a particular physical location
2. Have the inter-processor communications performing with the same speed on all links without substantial contentions
3. Allow messages to be transmitted from one resource to another while a task is being executed on the recipient resource.

2.3 Application Model

Services offered by cloud providers can be classified into Software as a service (SaaS), Platform as a service (PaaS) and Infrastructure as a service (IaaS). Note that, when instances of these services are running, they can be regarded as computational tasks or simply tasks. While IaaS requests are typically tied with predetermined time frames such as pay-per-hour, requests of SaaS and PaaS are often not strongly tied with a fixed amount of time such as pay-per-use. However, it can be possible to have estimates for service requests for SaaS and PaaS based on historical data and consumer supplied service information. Service requests in our study arrive in a Poisson process and the requested processing time follows exponential distribution. I assume that the processor usage of each service request can be identifiable. It is also assumed that disk and memory use correlates with processor utilization. Hereafter, application, task and service are used interchangeably [2, 8, 9].

3. THE TASK CONSOLIDATION PROBLEM

The task consolidation is also known as workload consolidation problem is the process of assigning a set $T=\{t_{0,\ldots,}\ tn_{-1}\}$ of n tasks to a set $R = \{r_{0,\ldots,}r_{m-1}\}$ of m cloud computing resources, without violating time constraints. The main purpose remains to maximize resource utilization and ultimately to minimize energy consumption. Time constraints

are directly related to the resource usage associated with the tasks. More precisely, in the consolidation problem, the resources allocated to a particular task must sufficiently provide the resource usage of that given task. For example, a task with its resource usage requirement of 60% cannot be assigned to a resource for which the available resource utilization at the time of that task's arrival is 50%.

4. PRESENT WORK

4.1 ECTC (Energy-Conscious Task Consolidation) Task Consolidation Algorithm

The cost function, termed ECTC, computes the actual energy consumption of the current task by subtracting the minimum energy consumption (p_{min}) required to run a task, if other tasks would be running in parallel with that task. That is, the energy consumption of the overlapping time period among the running tasks and the current task (t_j) is explicitly taken into account. The cost function tends to discriminate the task being executed in a standalone mode.

ECTC Algorithm

Step 1: Let r*=Ø

Step 2: Repeat Step 3 to Step 5 for all $r_i \in R$

Step 3:calculate cost function $f_{i,j}$ where
$f_{i,j} = [(p\Delta \times u_j + p_{min}) \times \tau_0] - [(p\Delta \times u_j + p_{min}) \times \tau_1 + (p\Delta \times u_j \times \tau_2)]$

Step 4:if $f_{i,j} > f_{*,j}$
Then go to step 5
Else go to step 2

Step 5: Set $r^* = r_i$
and $f_{*,j} = f_{i,j}$

Step 6: Set $r^* = \mathbf{t_j}$

Step 7: Exit

Flow chart

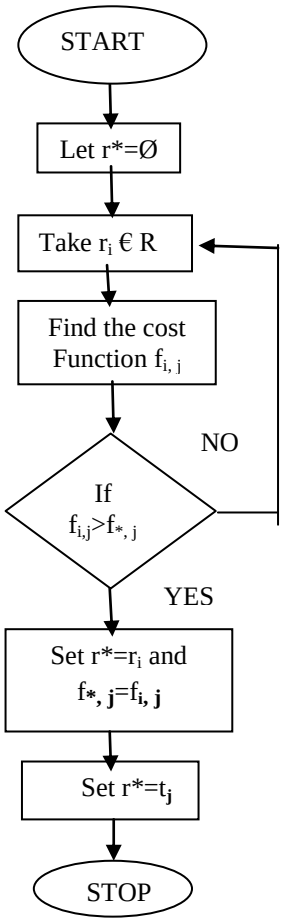

Figure 2: Algorithm and flow chart of ECTC Algorithm

The value $f_{i,j}$ of a task t_j on a resource r_i obtained using the ECTC cost function is defined as

$$f_{i,j} = [(p\Delta \times u_j + p_{min}) \times \tau_0] - [(p\Delta \times u_j + p_{min}) \times \tau_1 + (p\Delta \times u_j \times \tau_2)] \quad \ldots (4)$$

Where $p\Delta$ is the difference between p_{max} and p_{min}, u_j is the utilization rate of t_j, and τ_0, τ_1 and τ_2 are the total processing time of t_j. The time period t_j is running stand alone and that t_j is running in parallel with one or more tasks, respectively. For example, consider two tasks t_0 and t_1 that are running in parallel on the same resource r_0 with t_0 arriving time on first resource. While computing the result for $f_{0,1}$

$$\tau_0 = \text{total execution time of } t_1$$
$$\tau_1 = \tau_0 - \tau_2$$
$$\tau_2 = \tau_0 - \tau_1 \quad \ldots\ldots\ldots (5)$$

Where τ_1 is the time period where t_1 will be running stand alone on r_0, and τ_2 the time period where t_1 will be consolidated with t_0 in r_0. The rationale behind the ECTC cost function is that the energy consumption at the lowest resource utilization is far greater than that in idle state, and the additional energy consumption imposed by overlapping tasks contributes to a relatively low increase [2, 8].

4.2 MaxUtil (Maximum rate Utilization) Task Consolidation Algorithm

The MaxUtil cost function is derived with the average utilization during the processing time of the current task, as core component. The cost function aims to increase consolidation density and has a double benefit:

1. Implicit reduction of the energy consumption is directly related
2. Decreased number of active resources

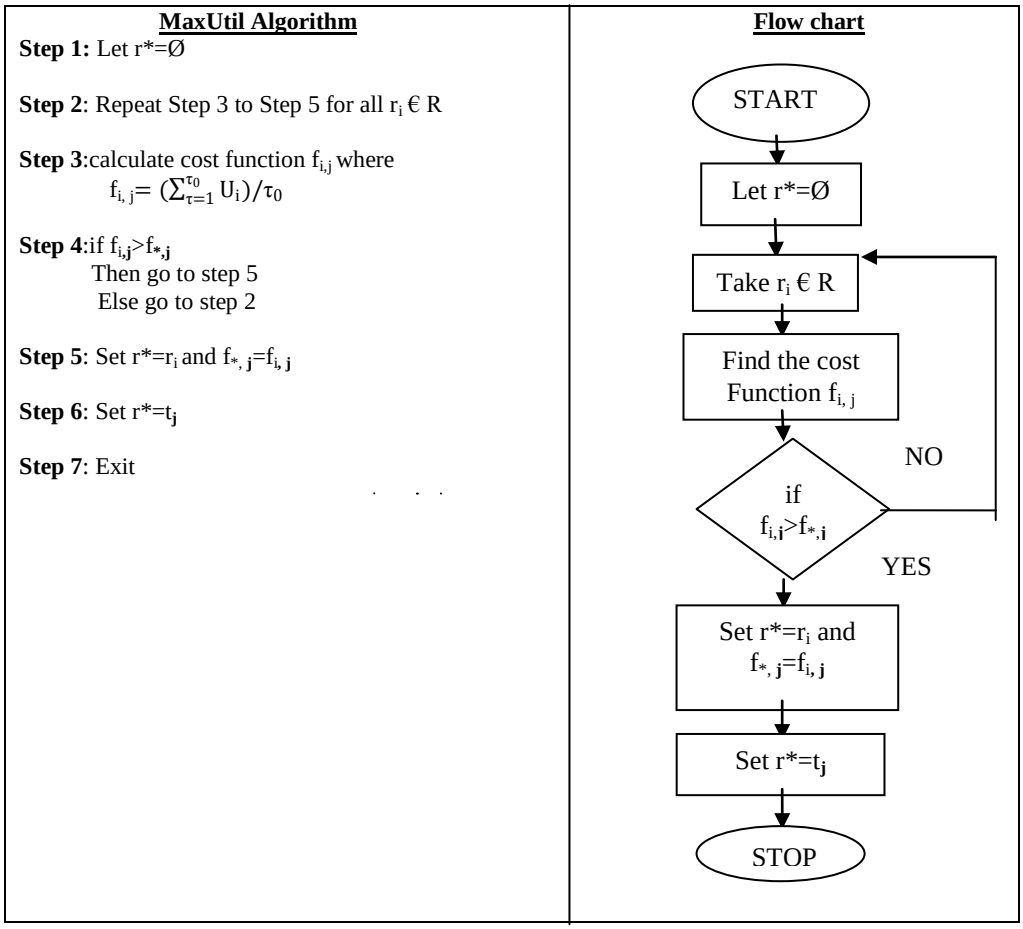

Figure 3: Algorithm and flow chart of MaxUtil Algorithm

In others words, MaxUtil tends to intensify the utilization of a small number of resources. Consequently, the value $f_{i,j}$ of a task t_j on a resource r_i using the MaxUtil cost function is defined as

$$f_{i,j} = (\sum_{\tau=1}^{\tau_0} U_i)/\tau_0 \quad \ldots\ldots\ldots (6)$$

Which is the utilization of a resource r_i, divided by total execution time (t_0) of task t_j [2, 8].

4.3 Bi-Objective Algorithm

The idea behind the bi-objective model is to combine the two cost functions to only benefit from ECTC and MaxUtil advantages. The algorithm will then provide the more energy efficient resource based on both of the considered aspects. I must note that ECTC computes the energy consumption of a given task on a selected resource, while MaxUtil looks after the more energy-efficient resource in terms of resource utilization. The ECTC cost function is designed to encourage resource sharing; the energy consumption of two tasks running in parallel is slightly superior than the energy consumption of a task ran alone.

To be accurate on the computation of the energy consumption, ECTC uses τ_1 and τ_2. Based on the time periods (τ_x), the cost function gives priority to resources where concurrent tasks can be fully consolidated and tends to discard the resources offering only a partial consolidation. Task t_0 do not fully overlap task t_3 on resource r_0, and then ECTC assigns t_3 on r_1 because t_3 can be fully consolidated with the task t_2. The working example presented pointed out the main drawback of ECTC. Intuitively, the resulting divergence from the behaviour of MaxUtil can be seen as a domino effect that will temporarily affect the system. Being energy efficiency the main concern of the presented heuristics, the eventuality of a domino effect should not be neglected while considering the ECTC cost function for the task consolidation problem. Alternatively, MaxUtil always minimizes the total number of used resources without individually considering the energy consumption of the given task. Because the objective of my study is to minimize the energy consumption as the total number of used resources, my proposal combines the two cost functions to select the resource that will most likely maximize the utilization rate and minimize the energy consumption.

The approach uses the two cost functions described in Equations. The respective results are combined to build a point in a two-dimensional search space where ECTC gives the x coordinate and MaxUtil the y coordinate. Originally, Equation returns a value greater than zero only when applied on a resource allowing task consolidation. Among the collected results, the highest value identify the most energy-

efficient resource (if $\tau_1 \neq \tau_0$), while the null value identifies empty resources (if $\tau_1 = \tau_0$). Figure illustrates the rationale behind the ECTC cost function. To properly construct the point in the search space, the two cost functions have to be slightly modified. Defining the energy consumption e_j of a task (t_j) on a given resource (r_i) as

$$e_j = (p\Delta \times u_j + p_{min}) \quad \ldots\ldots\ldots (7)$$

The value $f_{i,j}$ of t_j on r_i obtained using the ECTC cost function is now defined as

$$f_{i,j} = \begin{cases} (e_j \times \tau_0) & ; \text{ if } \tau_1 = \tau_0 \\ \left((e_j \times \tau_1) + (p\Delta \times u_j \times \tau_2)\right) & ; \text{ otherwise} \end{cases} \quad \ldots (8)$$

The value of $f_{i,j}$ obtained using the MaxUtil cost function as

$$f_{i,j} = \sum_{a_j}^{d_j} U_i \quad \ldots\ldots\ldots (9)$$

Where a_j is the arrival time and d_j the due date of the current task t_j. The design of Algorithm does not identify equivalent solutions. A double dominance check must be introduced and the equivalent solutions added to a subset F ($F \subseteq D$). Because the optimum solution may change by the time the domain space (D) is constructed, F must reset each time a new optimum point is identified. This will ensure that the subset only contains the equivalent solutions to the latest optimum point.

By the time the solution space is constructed, the equivalent optimum points will be identified. The selection among the equivalent solutions belonging to F, if any, will rely on d. Because our approach maximizes the considered objectives, the complement of d, denoted as δ, will be considered according the formula

$$\delta : [y_{min} : y_{max}] \rightarrow [y_{min} : y_{max}], \quad \text{and} \quad \delta = y_{max} - d \quad \ldots\ldots\ldots (10)$$

The aforementioned selection process will sequentially compare each f Є F with the actual optimum point. The actual optimum will then be updated based on the δ parameter, or on the sum of the two coordinates ((f_x, f_y) Є p_i), if the pair share the same value for the x(energy consumption) or y (utilization) coordinate [8].

Bi-Objective Algorithm

Step 1: Set r* and optimum = Ø

Step 2: Repeat Step 3 to Step 5 for all r_i Є R

Step 3: Set $x=f_x$ and $y=f_y$ and result=(x,y)

Step 4: if result>optimum
Then go to step 5
Else go to step 2

Step 5: Set optimum=result And r*=r

Step 6: Exit

Flow chart

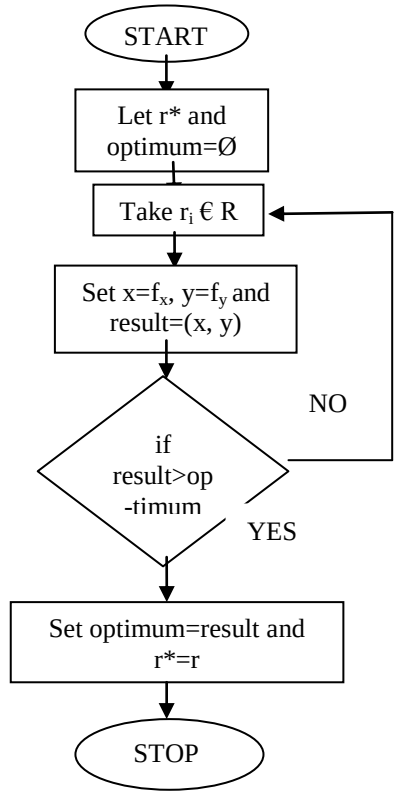

Figure 4: Algorithm and flow chart of Bi-Objective Algorithm

5. EXPERIMENT AND RESULTS

In this section, I describe experimental methods and settings including task characteristics and their generation. Experimental results are then presented based on energy consumption. While resource utilization might be a good performance measure, however, average utilization rates over all resources are not shown since they are already represented by energy consumption. The performance of ECTC, MaxUtil and bi-objective will be thoroughly evaluated with a large number of experiments using a diverse set of tasks. In addition to task characteristics, Energy efficient utilization of resources in cloud computing systems three algorithms (ECTC, MaxUtil and bi-objective) will be used. Variants of these three algorithms were further implemented incorporating task migration

Table 1: Experiment with different task and processing time

Sr. No	Task Number	Task Arrival time	Processing time	Utilization
1	T_0	00	25	40%
2	T_1	04	05	20%
3	T_2	08	10	60%
4	T_3	15	15	10%

The total number of experiments conducted is 500 different numbers of tasks between 1 and 50 at intervals of 10, 10 mean inter-arrival times between 10 and 100 with a random uniform distribution and three kinds of resource usage patterns which are random, low and high. In the first case, resource usage of tasks generated is random and uniformly distributed between 10% and 100%. For tasks with low and high resource usage patterns, usage ranges are generated using a Gaussian random number generator with mean utilization rates of 30% and70%, respectively. Task arrival times are modelled in a Poisson process and task processing times follow exponential distribution. I assume task processing times specified are hard deadlines performance degradation is not acceptable. Task 3 (t_3) arrives at time 15 after tasks 0, 1 and 2, and it is assigned onto resource 1 (r_1) based on energy consumption even though the utilization of resource 0 (r_0) is higher if t_3 is assigned on r_0. MaxUtil assigns

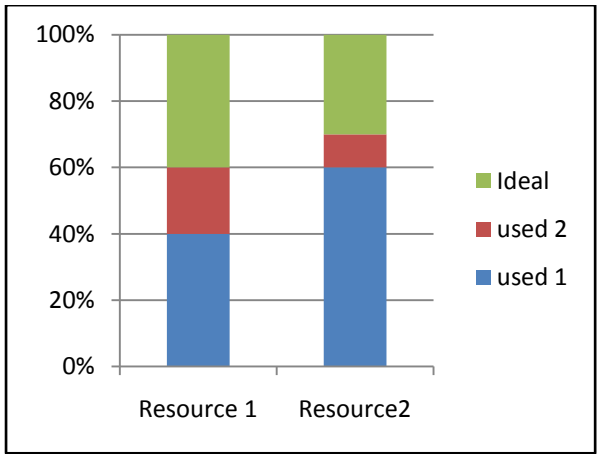

Figure 5: Working Example of ECTC

t_3 onto r_0 and this leads to a better match as compared with ECTC.I used p_{max} and p_{min} of 30 and 20, respectively. These values can be seen as rough estimates in actual resources and can be referenced as 300 watt and 200 watt, respectively. Since existing task consolidation algorithms are not directly comparable to our heuristics, our comparisons have been carried out between ECTC, MaxUtil and Bi-Objective.

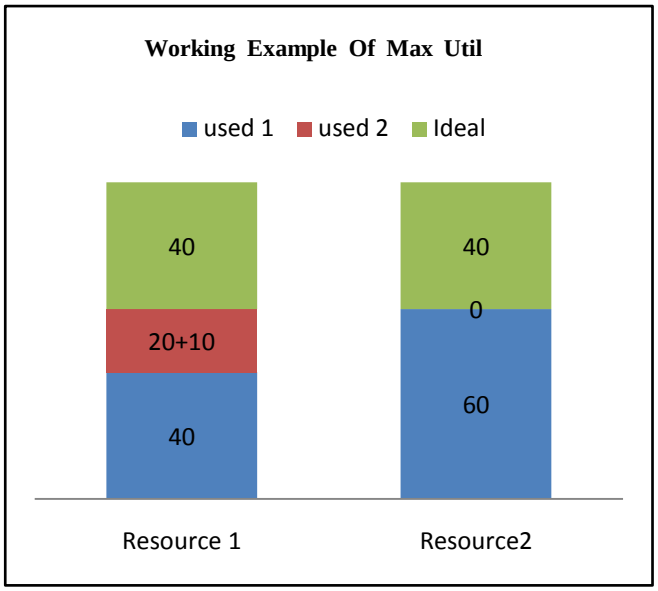

Figure 6: Working example of MaxUtil

Those existing task consolidation techniques introduced exhibit substantial differences in energy and scheduling models. During initial experiments with those three heuristics (ECTC, MaxUtil and Bi-Objective), I observed that in some circumstances the relocation of some running tasks can further reduce energy consumption. This observation motivated us to implement a variant for each of those three incorporating task migration and these variants are named ECTC_m, MaxUtil_m and Bi-Objective _m, respectively.

The entire results obtained from our extensive simulations are summarized in Table and results for different resource usage patterns are presented in Figure. Although the simulations were performed with 50 different numbers of tasks, only results obtained with 11 representative task volumes are presented. Energy savings in Table are relative rates to results obtained from experiments using Bi-Objective algorithm. These results clearly demonstrate the competent energy saving capability of ECTC, MaxUtil and Bi-Objective. Overall, ECTC and MaxUtil outperformed, regardless of the adoption of migration by 18% and 13%, respectively.

While energy savings with high and random resource usage patterns are still appealing, tasks with low resource usage are most suitable for task consolidation. Interestingly, the benefit of using migration was not apparent. This is mainly because migrated tasks tend to be with short remaining processing times and these tasks likely to hinder the consolidation of new arriving tasks, resulting in more energy consumption compared with the case when migration is not considered.

Table 2: Relative energy savings

Usage Pattern	ECTC			MaxUtil			Bi-Objective		
	Low	Random	High	Low	Random	High	Low	Random	High
With Migration Energy Saving	32%	16%	9%	23%	11%	5%	34%	17%	11%
Without Migration Saving Energy	33%	17%	9%	25%	12%	4%	36%	20%	14%
Mean Saving	18%			13%			22%		

6. CONCLUSION AND FUTURE SCOPE

Task consolidation, especially in cloud computing systems, became an important approach to streamline resource usage which improves energy efficiency. Three existing energy conscious heuristics for task consolidation offering different energy saving possibilities were analyzed in this study. The cost functions incorporated effectively capture energy saving possibilities and their capability has been verified by our evaluation study. For these heuristics, I identified the corresponding drawback and proposed, as a solution, the Bi-objective Task Consolidation algorithm. This algorithm combines the two heuristics to construct the corresponding bi-objective search space. The efficiency of the proposed algorithm was proved thought the evaluation study consisting of different simulations carried out. at each task assignment I observed three main aspects: total energy consumption, total resource utilization, and time needed to select the optimum solution. To evaluate the performance of the BTC algorithm, the two heuristics were individually implemented and used as key indicator for the energy efficiency and the scalability. Despite the more elaborate selection of the optimum solution, my study reported that the proposed BTC algorithm was the slowest when compared but resulted being the heuristic that provided the best energy efficient solution. During initial experiments with those three heuristics (ECTC, MaxUtil and Bi-Objective), I observed that in some circumstances the relocation of some running tasks can further reduce energy consumption. This observation motivated us to implement a variant for each of those three incorporating task migration and these variants are named ECTC_m, MaxUtil_m and Bi-Objective _m, respectively. Relocation can be considered for each running task at any time resource utilization changes for task completion.

7. REFERENCES

[1] Qi Zhang, Lu Cheng, Raouf Boutaba, 2010, "Cloud computing: state-of-the-art and research challenges", University of Waterloo, Waterloo, Ontario, Canada.

[2] Young Choon Lee, Albert Y. Zomaya, 2010, "Energy efficient utilization of resources in cloud computing systems", Springer Science+Business Media.

[3] Zahra Abbasi, Michael Jones, Ayan Banerjee, Sandeep Gupta, and Georgios Varsamopoulos, 2013, "Evolutionary Green Computing Solutions for Distributed Cyber Physical Systems", Springer-Verlag, Berlin Heidelberg.

[4] Rodrigo N. Calheiros, Marco A. S. Netto, C´esar A. F. De Rose, Rajkumar Buyya1, 2012, "EMUSIM: An Integrated Emulation and Simulation Environment for Modeling, Evaluation, and Validation of Performance of Cloud Computing Applications", Wiley InterScience.

[5] Eric R. Masanet, Richard E. Brown, Arman Shehabi, Jonathan G. Koomey and Bruce Nordman, 2012, "Estimating the Energy Use and Efficiency Potential of U.S. Data Centers", Proceedings of the IEEE.

[6] Yacine.Kessaci, Nouredine.Melab, El-Ghazali.Talb, Nov 6, 2012, "A Multi-start Local Search Scheduler for an Energy-aware Cloud Manager", INRIA Lille Nord Europe, Universite de Lille1.

[7] Kocovic Petar, March 3, 2011, "Challenges in Cloud Computing", AlphaUniversity, Belgrade, Serbia.

[8] Giorgio L. Valentini, Samee U. Khan, and Pascal Bouvry, 2011, "Energy-efficient Resource Utilization in Cloud Computing".

[9] Fan X, Weber X-D, Barroso LA, 2007, "Power provisioning for a warehouse sized", 34th Computer In: Proc annual international symposium on computer architecture, pp 13–23.

[10] Parkhill D, 1996,"The challenge of the computer utility", Addison Wesley Educational.

[11] J. Koomey, 2007, "Estimating Total Power Consumption by Servers in the U.S. and World", Oakland, CA.

[12] J. Koomey, 2007, "Estimating Regional Power Consumption by Servers: A Technical Note", Oakland, CA.

[13] Gustedt J, Jeannot E, Quinson M, September 2009, "Experimental methodologies for large-scale Systems: a survey", Parallel Processing Letters, Page: 399–418.

[14] R. Brown, E. Masanet, B. Nordman, W. Tschudi, A. Shehabi, J. Stanley, J. Koomey, D. Sartor,P. Chan, J. Loper, S. Capana, B. Hedman, R. Duff, E. Haines, D. Sass, and A. Fanara, 2007, "Report to Congress on Server and Data Center Energy Efficiency: Public Law 109-431," Lawrence Berkeley National Laboratory.

[15] Fontan, J, Vazquez, T, Gonzalez, L., Montero,R.S.,Llorente, 2008, "The open source virtual machine manager for cluster computing", San Francisco, CA, USA.