2008 IEEE International Conference on Services Computing

A Two Stage Heuristic Algorithm for Solving the Server Consolidation Problem with Item-Item and Bin-Item Incompatibility Constraints

Rohit Gupta, Sumit Kumar Bose, Srikanth Sundarrajan, Manogna Chebiyam, Anirban Chakrabarti
SETLabs, Infosys Technologies Limited-560100
{rohit_gupta12, sumit_bose, srikanth_sundarrajan, manognar_c, anirban_chakrabarti}@infosys.com

Abstract

The problem of server sprawl is common in data centers of most business organizations. It is most often the case that an application is run on dedicated servers. This leads to situations where organizations end up having numerous servers that remain under-utilized most of the times. The servers, in such scenarios, are allocated more resources (disk, cpu and memory) than are justified by their present workloads. Consolidating multiple under-utilized servers into a fewer number of non-dedicated servers that can host multiple applications is an effective tool for businesses to enhance their returns on investment. The problem can be modeled as a variant of the bin packing problem where items to be packed are the servers being consolidated and bins are the target servers. The sizes of the servers/items being packed are resource utilizations which are obtained from the performance trace data. Here we describe a novel two stage heuristic algorithm for taking care of the "bin-item" incompatibility constraints that are inherent in any server consolidation problem. The model is able to solve extremely large instances of problem in a reasonable amount of time.

1. Introduction

The problem of server sprawl is common in data centers of most business organizations. Server sprawls are characterized by the use of dedicated servers for single applications. This leads to situations where organizations end up having numerous servers that remain under-utilized most of the times. The servers, in such scenarios, are allocated more resources than are justified by their present workloads. Since organizations invest substantial amounts of money in data centers, organizations are undertaking consolidation exercises for reducing the infrastructure costs and maximizing their returns on investment. Server consolidation is a common practice in most data centers and can be categorized into three types – centralization, physical consolidation and application integration. Centralization involves moving multiple geographically dispersed servers into one common location. Physical consolidation involves reducing the number of servers by introducing fewer numbers of more powerful and technologically superior servers. Application integration involves combining multiple applications into one common application. The focus of the current paper is physical consolidation. Advances in system virtualization technologies, Xen and Hyper-visor for example, are responsible for the current interest in server consolidation. Consolidating multiple under-utilized servers into small number of servers is an effective tool for businesses to enhance their operational efficiency. For example, a Unix Server Consolidation Survey conducted by IT industry research and analysis firm Gabriel Consulting Group, Inc. (GCG) revealed that customers are increasingly turning to consolidate their applications onto mid-range and large UNIX servers for realizing significant costs and operational benefits [1]. Findings in [2] indicate that the cost is not the only factor influencing server consolidation projects. Several other factors such as improved performance, ease of management and technology improvement are key drivers behind the server consolidation exercise. According to Gartner Inc., 94% of IT departments are either considering server consolidation or are currently consolidating [3].

In a nutshell, the goal of server consolidation is to minimize the number of destination servers (also called target servers) with the view of reducing cost and real estate space. Till date, server consolidation exercise is primarily a manual process that involves analyzing the historical workload pattern of the servers and finding out the group of (existing) servers that can be moved to a high performing target server. This is often a time consuming process and depends on the subjective assessment of the decision maker. Ajiro and Tanaka [4] has shown that the problem can be modeled as a variant of the bin packing problem called the vector packing problem where items to be packed are the (existing and technically deprecated) servers being consolidated and bins are the (high performing and technically superior destination servers) target servers. The sizes of the servers/items being packed are resource utilizations obtained from analyzing the performance trace data. Zhang et al. [5,6] apply bin-packing algorithms to server consolidation based on performance trace data and user-defined consolidation constraints. The authors extend the deterministic bin-packing heuristics (first-fit decreasing and best-fit decreasing) to high dimensional probabilistic bin

978-0-7695-3283-7/08 $25.00 © 2008 IEEE
DOI 10.1109/SCC.2008.39

capacities. Ajiro and Tanaka [4] model the server consolidation problem as a vector packing problem without the incompatibility constraints and provide an improved first-fit decreasing algorithm for solving the same. None of the current work deals with the complete set of incompatibility constraints that naturally exist in server consolidation exercises. Two commonly encountered incompatibility constraints are the "item-item incompatibility constraints" and the "bin-item incompatibility constraints". "Item-item incompatibility constraints" occur when two (existing) servers cannot be collocated. "Bin-item incompatibility constraints" arise when a given (existing) server cannot be moved to a particular bin. For example, an application currently hosted on a 64 bit machine can-not be migrated to a 32 bit machine. In the bin-packing literature, a number of variants of the classical bin-packing model have been studied. For example, Chu and La [7] and Kang and Park [8] study the variable sized bin packing problems. Gendreau et. al [9], Epstein and Levin [10] and Jansen [11] model the bin packing problem with conflicts and provide approximate solutions. However, the conflicts considered in these research papers are implicitly assumed to be "item-item" incompatibility constraints. To the best of our knowledge, we have not come across any papers that discuss the "bin-item" incompatibility constraints that are inherently present in any server consolidation exercise. In this context, the problem dealt with in the current paper is clearly a generalization of the classical bin/vector packing problem and its variants tackled in the literature thus far. The conflicts and the incompatibility constraints studied in the current paper can be considered to be a super-set of the incompatibility constraints modeled in the bin/vector packing literature till date.

In vector packing with "item-item" and "bin-item" incompatibility constraints, we are given items of different sizes and we have to pack these items into minimum number of bins with different capacities. In the "server" packing problem, items are (existing) servers, bins are (destination) target servers, item sizes are resource utilization calculated from the trace history and bin capacities are utilization thresholds of the new servers. In addition, we are given the different "item-item incompatibility constraints" and the "bin-item incompatibility constraints". We are required to determine the minimum number of target servers that would be required after taking care of the different incompatibility constraints for the problem. This paper formalizes the "item-item" incompatibility constraints as a graph coloring problem and the "bin-item" incompatibility constraints as a pre-colored graph coloring problem. We developed a new heuristic algorithm of determining the number of destination servers in the presence of the incompatibility constraints including bin-item incompatibilities. In addition, our experiments reveal that our algorithm outperforms other heuristics for dense conflict graphs. Section 2 models the server consolidation problem mathematically as a vector packing problem. Section 3 explains the heuristic algorithm in detail. Section 4 describes our experiences with real life data before providing concluding remarks and directions for future research in section 5.

2. Problem Scenario

Given a set of new target servers, I, and a set of old servers, J, ($I < J$) along with the workload history and usage pattern of the resources (disk, cpu, memory) for each of the J servers, the server consolidation problem is to find the best possible way to combine the existing servers into the new target servers such that only a few of the target servers may be used. We use the term bin to represent target servers and the term item to represent the servers being migrated. [12] Proposes mathematical models for static and dynamic server allocation problem. The formulation below, a variant of the ones proposed in [12] uses the notations given in table 1.

Table 1: Notations, Variables and Parameters

Notations	
i	New servers $i \in [1, 2, \ldots, I]$
j	Old servers $j \in [1, 2, \ldots, J]$
Variables	
Y_i	Binary variable, equals 1 if server i is used for consolidation, 0 otherwise
X_{ij}	Binary variable equals 1 if old server j is migrated to new server i, 0 otherwise.
Parameters	
$memory_i$	Maximum available memory of the new server i.
$\widetilde{mem}\, j$	Memory usage of old server j.
cpu_i	Maximum available cpu power of the new server i.
$\widetilde{cpu}_j$	CPU usage of old server j.
$disk_i$	Maximum available disk of the new server i.
$\widetilde{disk}\, j$	Disk usage of old server j.

Binary variable Y_i is 1 if there is at-least one item $j \in J$ that is migrated to a bin i. Binary variable X_{ij} is 1 if item $j \in J$ is migrated to the target server i. $\widetilde{mem}\, j$ is calculated as $(\sum_t memory_{jt} / T) + k\sigma_{memory}$ and represents the memory usage of the server j. Similarly, $\widetilde{cpu}_j$ is calculated as $(\sum_t cpu_{jt} / T) + k\sigma_{cpu}$ and represents the CPU usage of the server j and $\widetilde{disk}\, j$ is calculated as

$(\sum_t disk_{jt}/T) + k\sigma_{disk}$ and represents the disk usage of the server *j*. *k* here is a tunable parameter. The problem is that of minimizing the number of target servers.

Equation (2) constrains the value of Y_i to be 1 when an item *j* is allocated to *i*. Equation (3) assigns an item *j* to only one of the target servers. Equations (4) to (6) model the capacity constraint of the target server. The values of $\widetilde{mem}_j$, $\widetilde{cpu}_j$ and $\widetilde{disk}_j$ are determined from the performance trace data (historical data) available from the data centers.

The problem can therefore be formulated as:

$$Minimize \quad \sum_i Y_i \qquad (1)$$

Subject to :

$$Y_i \geq X_{ij} \qquad \forall i, j \qquad (2)$$

$$\sum_i X_{ij} = 1 \qquad \forall j \qquad (3)$$

$$Y_i * memory_i \geq \sum_j \widetilde{mem}_j X_{ij} \qquad \forall i \qquad (4)$$

$$Y_i * cpu_i \geq \sum_j \widetilde{cpu}_j X_{ij} \qquad \forall i \qquad (5)$$

$$Y_i * disk_i \geq \sum_j \widetilde{disk}_j X_{ij} \qquad \forall i \qquad (6)$$

Additionally, let $J_A \in J$ be a subset of servers such that no two instances of the set J_A can be hosted onto the same instance of the server $i \in I$. In other words, if *j* and *j'* be two servers instances of the group J_A then *j* and *j'* cannot be migrated to the same server *i*.; if *j* is migrated to *i*, then *j*' need to be migrated to *i*'. This condition should be true for all members of the set J_A. We call such constraints as the "incompatibility constraints". This arises from the need to exclude members of the set J_A from being assigned to a server *i*, once a member from the set J_A is assigned to a server *i*. The condition can be modeled mathematically as:

$$\sum_{j \in J_A} X_{ij} \leq 1 \qquad \forall i \qquad (7)$$

Likewise, if we have more such constraints i.e. $J_B \in J$ be a subset of servers such that no two instances of the set J_B can be migrated together to the same instance of the server $i \in I$, then we will have additional constraints as:

$$\sum_{j \in J_B} X_{ij} \leq 1 \qquad \forall i \qquad (8)$$

The number of such constraints will depend upon the number of item-item incompatibilities in the problem. They vary with each instance of the problem. The formulation becomes expensive for very large instances of the problem. Problem instances involving 4000 plus servers is not uncommon in the industry. Solving such large problem instances to optimality using the mathematical solvers will take unusually long time. Under these situations it becomes imperative to explore heuristic schemes that can generate near optimal solution within a reasonable amount of time.

Additionally we have the item-bin incompatibility constraints. These impose further restrictions on the items being packed into a bin. The constraints imply that a certain item *j*' cannot be packed into a given bin *i*'. Mathematically,

$$X_{i'j'} = 0 \qquad \exists i', j' \qquad (9)$$

Such item-bin incompatibility constraints are common in server consolidation exercise. The formulation then is:

$$Minimize \quad \sum_i Y_i$$

Subject to :

$$Y_i \geq X_{ij} \qquad \forall i, j$$

$$\sum_i X_{ij} = 1 \qquad \forall j$$

$$Y_i * memory_i \geq \sum_j \widetilde{mem}_j X_{ij} \qquad \forall i \qquad (P1)$$

$$Y_i * cpu_i \geq \sum_j \widetilde{cpu}_j X_{ij} \qquad \forall i$$

$$Y_i * disk_i \geq \sum_j \widetilde{disk}_j X_{ij} \qquad \forall i$$

$$\sum_{j \in J_A} X_{ij} \leq 1 \qquad \forall i$$

$$\sum_{j \in J_B} X_{ij} \leq 1 \qquad \forall i$$

$$X_{i'j'} = 0 \qquad \exists i', j'$$

3. Heuristic Solution

We divide the set of constraints (equations (2) to (9) of the formulation presented above into two mutually exclusive sub-sets. Sub-set (2) to (6) form the constraint set-A, and sub-set (7) to (9) (i.e. equations that are structurally similar to equation (7) and those similar to equation (9)) form the constraint set-B. We call the constraint set-B as the set of "incompatibility constraints". Equations similar in structure to equation (7) are called the "item-item incompatibility constraints" and those similar in structure to equation (9) are called the "bin-item incompatibility constraints". Item-item constraints can be explicitly specified by the user or they can be an outcome of the hot-spot analysis. The set of "item-item incompatibility constraints" for the above formulation is given by the following equations:

$$\sum_{j \in J_A} X_{ij} \leq 1 \qquad \forall i$$

$$\sum_{j \in J_B} X_{ij} \leq 1 \qquad \forall i$$

The set of "bin-item incompatibility constraints" for the above formulation is given by the following equations:

$$X_{i'j'} = 0 \qquad \exists i', j'$$

We solve the server(item) – target server(bin) mapping problem in two stages. In stage 1 we solve a restricted version of the problem (P1). This involves minimizing (1) subject to the constraint set-B. We call this limited version of the problem (P1) as the problem (P2). Succinctly, it can be stated as:

$$Minimize \sum_{i=1}^{i} Y_i$$

$$Subject\,to$$

$$\sum_{j \in J_A} X_{ij} \leq 1 \qquad \forall i$$

$$\sum_{j \in J_B} X_{ij} \leq 1 \qquad \forall i \qquad (P2)$$

$$X_{i'j'} = 0 \qquad \exists i', j'$$

Optimal solution to problem (P2) gives us the heuristic estimate of the minimum number of target servers required for a given problem instance. Furthermore, solution to the problem (P2) organizes the servers into clusters. In other words, the solution identifies groups of servers (clusters) that can be co-located. The heuristic estimate is the minimum number of such clusters (group of servers) that can be formed. The solution obtained at the end of stage-1 would have been optimal for the problem (P1) if the bins were un-capacitated. Stage 2 of the solution building process, refines the partial solution obtained in the previous step by taking into account the actual bin capacities and performing server – target-server mappings. The algorithm can be summarized as:

Step 1: Solve the set of "incompatibility constraints". As explained above, the solution to this step defines the clusters of servers such that servers belonging to the same cluster can be co-located onto a target server. This step is explained in detail in section 3.1 and 3.2.

Step 2: Allocate servers (items) to the different target-servers (bins). This step is explained in detail in section 3.3.

3.1 Solving the "Item-Item Incomp. constraints"

Consider a subset of the problem (*P*2):

$$Minimize \sum_{i=1}^{i} Y_i$$

$$\sum_{i} X_{ij} = 1 \qquad \forall j \in J_A, J_B$$

$$\sum_{j \in J_A} X_{ij} \leq 1 \qquad \forall i$$

$$\sum_{j \in J_B} X_{ij} \leq 1 \qquad \forall i \qquad (P3)$$

A close look at the problem (*P*3) will reveal that the problem definition resembles that of the Graph Coloring Problem. The Graph Coloring Problem involves coloring the vertices of a graph such that no two adjacent vertices share the same color. The optimal solution to the problem (*P*3) helps us to determine the minimum number of target servers that will be required for the consolidation exercise.

Definition: An undirected graph $G=(V,E)$ is composed of a set V of $|J_A \cup J_B|$ nodes or vertices, and a set E of ${}^{|J_A|}C_2 + {}^{|J_B|}C_2 - {}^{|J_A \cap J_B|}C_2$ edges between nodes.

Example 1: Assume that we have two item-item incompatibility constraints:

$$X_{i1} + X_{i2} + X_{i3} + X_{i4} + X_{i5} \leq 1$$

$$X_{i1} + X_{i6} + X_{i7} \leq 1$$

As shown in figure 1, the equations can be represented as a graph of 7 nodes and 13 edges. The target servers are analogous to the colors in this example. Assigning separate target servers to nodes 1, 6 and 7 is equivalent to assigning three different colors to the respective nodes. None of the adjacent nodes should share the same color is equivalent to the stating that none of the adjacent nodes of a graph (i.e. nodes connected together by an edge) should appear on the same target server.

Theorem: Let $G=(V,E)$ be a completely connected graph formed out of the set of "Item-Item incompatibility constraints" in (*P*2), and there exists a unique optimal solution to the problem (*P*3), then the optimal solution to the problem (*P*3) will represent the lower bound to the solution for the problem (*P*1).

Proof: Suppose the optimal solution to the problem (*P*1) is less than the optimal solution to the problem (*P*3). In such a case there will be at least two items *j, j'* belonging to J_A (or J_B say) that will be hosted together on a target server, which will be violating the constraint (7) (or equation (8) say). Hence the number of target servers that is required for (*P*1) can be no less than the optimal solution of (*P*3).

Since, the number of target servers obtained as a result of solving the problem (*P*3) represents the minimal number of target servers that will be required for the problem in (*P*1), we solve the problem (*P*3) first. Optimal solution to (*P*3) implicitly defines the clusters of conflicting items that can be grouped together such that item in one group is not in conflict with any other member of the group. However, the problem is known to be NP-hard [13]. The problem is well studied in the literature and a number of solutions have been proposed in the literature. One such

popular approximation algorithm is the Welsh-Powell method [14].

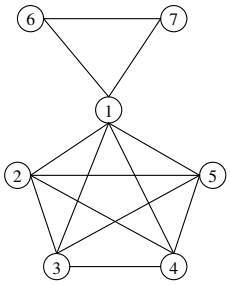

Figure 1: Conflict Graph for the "Item-Item Incompatibility Constraints" for Example 1.

The Welsh-Powell algorithm for solving the graph coloring problem uses a simple heuristic and is as follows:

1. Sort the vertices in decreasing order of degree. To begin with all the vertices are uncolored.
2. Traverse the vertices in the sorted list, and assign a vertex the color 1 if it is uncolored and in case the vertex does not yet have a neighbor having color 1.
3. Repeat this process with other colors until no vertex is uncolored.

The algorithm thus assigns to each vertex a color that is different from the colors of its neighbors. The algorithm is proven to use at most $\Delta(G) + 1$ colors, where $\Delta(G)$ is the maximum degree of the graph. For all practical purposes, the Welsh-Powell algorithm offers a good approximation to the optimal solution to problem (P3). The heuristic explained in this sub-section is similar to that of Gendreau et. al. [9] and Jansen and Oehring [15]

3.2 Solving the "Bin-Item Incomp. Constraints"

Continuing with our example, suppose in addition to the two item-item incompatibility constraints:

$$X_{i1} + X_{i2} + X_{i3} + X_{i4} + X_{i5} \leq 1$$

$$X_{i1} + X_{i6} + X_{i7} \leq 1$$

we also have three bin-item incompatibility constraints:

$$X_{A1} = 0$$

$$X_{A7} = 0$$

$$X_{B7} = 0$$

The three bin-item incompatibility constraints imply that the item 1 cannot be assigned to bin *A* and item 7 cannot be assigned to bin *B* in addition to bin *A*. To take care of such bin-item incompatibility constraints, we suitably modify the conflict graph of figure 1. Particularly, we augment the conflict graph by introducing 'pre-colored' dummy nodes into the graph. Each of the dummy nodes introduced will be colored differently and the number of such pre-colored dummy nodes depends on the number of bins that are in conflict with the items. For the example scenario presented here, we introduce two pre-colored dummy nodes since there are only two bins – *A* and *B* that are not compatible for some of the items. Figure 2 shows these two newly introduced nodes with double circles. These doubly encircled nodes represent the bins. Such nodes we call bin-nodes which are different from the single circled nodes, called item-nodes for the sake of clarity and exposition. A newly introduced 'pre-colored' bin-node is thereafter connected using an undirected edge to those item-nodes that cannot be assigned to this particular bin. As stated earlier, the optimal solution to this problem (Problem (*P*3)) defines a solution the clusters of items that can be grouped together such that none of the incompatibility constraints (item-item and bin-item) are violated. As before the problem is NP-hard and requires a suitable heuristic for solving the problem. We solve the problem by modifying the graph coloring heuristic of Welsh-Powell suitably. Instead of sorting all the vertices in decreasing order of the degree, we maintain two sorted lists. The first one – called the pre-colored list, contains the sorted 'pre-colored' vertices in the decreasing order of the degree. The second – called the uncolored list, contains the sorted uncolored vertices in the decreasing order of the degree. Thereafter, apply the modified Welsh-Powell algorithm as follows –

1. Select the next pre-colored vertex from the pre-colored list (At the start of the algorithm this is the first element in the pre-colored list).
2. Traverse the vertices in the uncolored list, and assign a vertex the color of the pre-colored vertex of step 1 if it is uncolored and in case the vertex does not yet have a neighbor having the same color. Go to Step-1.
3. For the vertices (in the uncolored list) that remain uncolored at the end of the traversal process of the pre-colored list, color them using the usual Welsh-Powell heuristic.

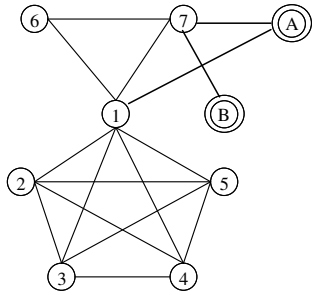

Figure 2: Conflict Graph for the Complete set of "Incompatibility Constraints" of Example 1.

3.3 Allocating servers to the target servers

Let *N* be the solution to the pre-colored graph coloring problem. The nodes having the same color define one cluster and hence denote the items that can be grouped

together into a target server provided capacity constraints of the target server are not violated. Lets label these N clusters as n_1, n_2, …., n_N. In addition to the items that belong to one of these N clusters, we have items which are not constrained by any of the incompatibility factors. These non-conflicting items can be associated with any of the N clusters. However, we decide to group such items into a separate cluster and associate a label n_{N+1} with the cluster. The algorithm is as follows:

1. Sort the bins in the decreasing order of volumes (CPU*memory*disk). To begin with all the bins are empty. Only the bins that are involved in the Bin-Item Incompatibility Constraints have pre assigned label.
2. Sort the items in the decreasing order of volumes (CPU*memory*disk). Each of the items has an associated label.
3. Pop out the next element from the items list.
4. Let bin_counter = 1.
5. Check whether the item can be assigned to the bin identified by the bin_counter. If it can be assigned go to step 6 else go to step 8.
6. If the bin does not have any label assigned to it yet, it is assigned a label that is same as the label of the item being assigned to the bin. Thus, if the label of the item is n_1 (*i.e* the item belongs to the cluster n_1) then the bin is labeled as n_1 as well. This means that henceforth only items belonging to the group n_1 can be assigned to the bin. However, if the item belongs to the cluster $n_{N+1,}$ no label change is required.
7. Remove the element from the list of sorted items and go to step 3.
8. Increment the bin_counter, if the bin_counter is less than or equal to the number of bins repeat step 5 else the item cannot be placed go to step 3. Since we are not placing any restriction on the number of bins, we are assuring that all the items get allocated.

The algorithm is used to allocate different items (servers) to the bins (target servers). This is FFD algorithm which packs items into a bin in descending order of size and adds a new bin for an item which cannot be accommodated in any of the already opened bins. In essence, we partition the items into N clusters of mutually non-conflicting items. The algorithm then solves a simple vector packing problem for each set.

4. Computational Experiments

The algorithm described above is implemented in Java. We conduct our experimental runs on an Intel® Pentium® 4 machine with 2.00 GHz CPU and 1GB RAM. Table 2 compares the performance of the FFD heuristic algorithm with the Hybrid Grouping Genetic Algorithm (HGGA) of Falkenauer [16] for the one-dimensional vector-packing problem (VPP). Further, the variation in the number of bins utilized and the algorithm execution time, when item-item and item-bin constraints are introduced into the problem (VPC), is shown. We experimented with 4 different sets of item numbers (n=120, 250, 500, 1000). For each set we vary the number of item-item constraints and the number of bin-item constraints. As in [16] the item weights are uniformly distributed over the range [20,100] and the bins have a fixed capacity of 150.

Table 2: Comparison of solution for the VPP and VPC (item-item and bin-item) for the one-dimensional case

Run	Items (n)	Optimal Solution For BPP	VPP				VPC			
			HGCA		FFD Heuristic		Our Heuristic			
			Bins	Time (sec)	Bins	Time (sec)	Item-item constraints	Bin–item constraints	Bins used	Time (sec)
1	120	48	48	15.2	49	.015	5	5	73	0.031
								10	73	0.031
								15	72	0.031
2	120	49	49	0.0	50	.015	10	5	71	0.031
								10	70	0.031
								15	71	0.063
3	120	46	46	5.8	47	.015	15	5	73	0.047
								10	72	0.031
								15	72	0.032
4	250	99	99	256.7	101	.031	5	5	137	0.062
								10	137	0.063
								15	132	0.062
5	250	100	100	47.4	102	.031	10	5	147	0.047
								10	147	0.062
								15	147	0.078
6	250	102	102	223.8	104	.032	15	5	148	0.063
								10	148	0.062
								15	148	0.281
7	500	198	198	480.5	202	.078	5	5	283	0.156
								10	289	0.125
								15	286	0.156
8	500	201	201	177.7	205	.078	10	5	291	0.203
								10	294	0.140
								15	294	0.171
9	500	202	202	347.9	206	.078	15	5	300	0.157
								10	300	0.282
								15	301	0.125
10	1000	399	399	2924.7	406	.156	5	5	556	0.281
								10	564	0.234
								15	567	0.172
11	1000	406	406	4040.2	413	.156	10	5	614	0.187
								10	617	0.187
								15	619	0.172
12	1000	411	411	6262.1	418	.157	15	5	620	0.171
								10	619	0.188
								15	618	0.172

Table 3: Comparative evaluation for the case when |J| = 120 and capacity of bins is 150 in all the 3 dimensions

Run	Number of Item-Item constraints	Cardinality of a Item-item constraint (Density)	Number of Bin – Item constraints	Lower Bound (L_1)	Heuristic Solution for the relaxed Problem (H_1)		Heuristic Solution for the Complete Prob.	
				Target Servers Used	Target Servers Used	Time (Sec)	Target Servers Used	Time (sec)
1	5	25%	0	31	35	0.063	42	0.157
			5	31	35	0.031	43	0.032
			10	31	35	0.016	44	0.016
		50%	0	31	41	0.016	51	0.016
			5	31	41	0.015	50	0.047
			10	31	41	0.031	48	0.016
2	10	25%	0	30	37	0.015	47	0.031
			5	30	37	0.015	45	0.015
			10	30	37	0.015	43	0.344
		50%	0	30	45	0.031	49	0.016
			5	30	45	0.032	50	0.031
			10	30	45	0.016	50	0.046
3	15	25%	0	30	38	0.016	49	0.016
			5	30	38	0.016	50	0.015
			10	30	38	0.016	50	0.016
		50%	0	30	53	4.703	53	0.032
			5	30	53	5.703	53	0.016
			10	30	53	0.046	53	0.016
4	20	25%	0	30	36	0.032	48	0.016
			5	30	36	0.015	48	0.016
			10	30	36	0.047	48	0.015
		50%	0	30	54	0.031	52	0.016
			5	30	54	0.032	55	0.016
			10	30	54	0.047	53	0.031
5	25	25%	0	31	37	0.031	47	0.016
			5	31	37	0.032	47	0.016
			10	31	37	0.032	47	0.047
		50%	0	31	66	0.141	59	0.015
			5	31	66	0.046	58	0.016
			10	31	66	0.047	58	0.031

Table 3 to Table 8 compares the performance our heuristic algorithm for the 3-dimensional vector packing problem with item-item and bin-item constraints with (*i*) a lower bound, L_1, for the problem (described below) and (*ii*) the heuristic solution, H_1, proposed in Gendreau et. al. [9] for a relaxation of the problem. We calculate the lower bound L_1 as the

$$\max\left\{\left\lceil \frac{\sum_{j\in J} cpu_j}{\max_{i\in I}(cpu_i)} \right\rceil, \left\lceil \frac{\sum_{j\in J} memory_j}{\max_{i\in I}(memory_i)} \right\rceil, \left\lceil \frac{\sum_{j\in J} disk_j}{\max_{i\in I}(disk_i)} \right\rceil\right\}$$

For calculating H_1, we consider a relaxed version of the problem: one without the bin-item constraints. The algorithm proposed is a modified FFD for the bin packing problem with item-item constraints in Gendreau et. al [9].

For constructing the test cases, we vary the number of items to be migrated and the capacity of the bins. Table-3(Table-4) presents the experimental runs for the case when there are 120 items in the set, J and the capacity of the bins is 150(200). Table 5 and 6 shows the corresponding figures for the case when $|J|$=250. We present one test case each of the scenario when $|J|$=500(Table 7) and $|J|$=1000(Table 8).

Table 4: Comparative evaluation for the case when $|J|$ = 120 and capacity of bins is 200 in all the 3 dimensions

Run	Number of Item-Item constraints	Cardinality of a Item-item constraint (Density)	Number of Bin – Item constraints	Lower Bound (L_2)	Heuristic Solution for the relaxed Problem (H_1)		Heuristic Solution for the Complete Prob.	
				Target Servers Used	Target Servers Used	Time (Sec)	Target Servers Used	Time (sec)
1	5	25%	0	23	31	0.141	41	0.141
			5	23	31	0.046	38	0.016
			10	23	31	0.047	37	0.032
		50%	0	23	45	0.031	46	0.031
			5	23	45	0.032	48	0.016
			10	23	45	0.047	49	0.031
2	10	25%	0	23	28	0.031	34	0.016
			5	23	28	0.047	36	0.016
			10	23	28	0.016	36	0.016
		50%	0	23	50	0.031	52	0.032
			5	23	50	0.047	52	0.015
			10	23	50	0.032	52	0.015
3	15	25%	0	23	32	0.031	36	0.032
			5	23	32	0.016	36	0.016
			10	23	32	0.047	36	0.047
		50%	0	23	60	0.062	52	0.047
			5	23	60	0.047	52	0.015
			10	23	60	0.047	52	0.016
4	20	25%	0	23	38	0.047	34	0.016
			5	23	38	0.015	34	0.078
			10	23	38	0.015	34	0.032
		50%	0	23	54	0.062	52	0.031
			5	23	54	0.062	52	0.016
			10	23	54	0.047	52	0.016
5	25	25%	0	23	31	0.016	38	0.016
			5	23	31	0.078	38	0.016
			10	23	31	0.015	38	0.016
		50%	0	23	59	0.078	54	0.047
			5	23	59	0.047	53	0.015
			10	23	59	0.047	53	0.047

Further, for each of the cases we vary the number of item-item constraints and the number of bin-item constraints. Additionally, we construct the conflict graph at random in the following manner: for any item-item constraint the cardinality of the constraint can vary from 2 to a certain density threshold (d). Each individual member of this item-item constraint set is then generated at random using a uniform distribution. Likewise, we construct the bin-item constrain set at random using a uniform distribution. For experiments, we consider the capacity of a bin to be same in all the three dimensions. The capacity of a bin is larger than any of the items' usage in any of the dimensions.

From the tables, we can observe, that our algorithm does not perform too well when the cardinality of the item-item incompatibility constraint is low (e.g. the instances with d=25%). However, our algorithm outperforms modified FFD when the cardinality is moderate to high(e.g. the instances with d=50%). This is because our algorithm performs exceedingly well when the conflict graph is a completely connected graph. On the contrary, our algorithm does not perform too well when the conflict graph is a disjoint graph. When the number of "item-item incompatibility constraints" and the cardinality for the set of "incompatibility constraints" is high, the likelihood of a completely connected conflict graph is more and this explains the enhanced performance of our algorithm for instances with higher values of d.

Table 5: Comparative evaluation for the case when $|J|$ = 250 and capacity of bins is 150 in all the 3 dimensions

Run	Number of Item-Item constraints	Cardinality of a Item-item constraint (Density)	Number of Bin – Item constraints	Lower Bound (L_2)	Heuristic Solution for the relaxed Problem (H_1)		Heuristic Solution for the Complete Prob.	
				Target Servers Used	Target Servers Used	Time (Sec)	Target Servers Used	Time (sec)
1	5	25%	0	62	73	0.109	92	0.032
			5	62	73	0.046	98	0.047
			10	62	73	0.031	94	0.031
		50%	0	62	79	0.078	101	0.031
			5	62	79	0.062	97	0.047
			10	62	79	0.11	102	0.031
2	10	25%	0	62	73	0.047	94	0.062
			5	62	73	0.047	91	0.078
			10	62	73	0.047	91	0.062
		50%	0	62	97	0.14	107	0.047
			5	62	97	0.11	106	0.031
			10	62	97	0.156	108	0.047
3	15	25%	0	62	74	0.078	95	0.063
			5	62	74	0.047	95	0.031
			10	62	74	0.062	96	0.032
		50%	0	62	95	0.093	102	0.047
			5	62	95	0.094	104	0.062
			10	62	95	0.125	104	0.063
4	20	25%	0	62	75	0.078	100	0.032
			5	62	75	0.062	99	0.047
			10	62	75	0.047	100	0.046
		50%	0	62	100	0.157	101	0.047
			5	62	100	0.125	102	0.063
			10	62	100	0.125	102	0.047
5	25	25%	0	63	78	0.078	100	0.032
			5	63	78	0.079	101	0.047
			10	63	78	0.047	101	0.031
		50%	0	63	120	0.25	105	0.047
			5	63	120	0.265	105	0.032
			10	63	120	0.265	105	0.047

Table 6: Comparative evaluation for the case when $|J|$ = 250 and capacity of bins is 200 in all the 3 dimensions

Run	Number of Item-Item constraints	Cardinality of a Item-item constraint (Density)	Number of Bin – Item constraints	Lower Bound (L_2)	Heuristic Solution for the relaxed Problem (H_1)		Heuristic Solution for the Complete Prob.	
				Target Servers Used	Target Servers Used	Time (Sec)	Target Servers Used	Time (sec)
1	5	25%	0	47	55	0.156	72	0.078
			5	47	55	0.047	73	0.047
			10	47	55	0.062	73	0.031
		50%	0	47	72	0.078	89	0.031
			5	47	72	0.094	87	0.047
			10	47	72	0.062	87	0.062
2	10	25%	0	47	61	0.047	76	0.031
			5	47	61	0.047	77	0.031
			10	47	61	0.062	75	0.031
		50%	0	47	88	0.187	89	0.047
			5	47	88	0.157	89	0.047
			10	47	88	0.172	88	0.032
3	15	25%	0	47	60	0.078	78	0.062
			5	47	60	0.078	77	0.047
			10	47	60	0.062	77	0.031
		50%	0	47	95	0.218	96	0.046
			5	47	95	0.187	96	0.031
			10	47	95	0.203	96	0.047
4	20	25%	0	48	63	0.094	73	0.109
			5	48	63	0.047	73	0.047
			10	48	63	0.062	73	0.047
		50%	0	48	103	0.204	97	0.047
			5	48	103	0.25	97	0.047
			10	48	103	0.234	97	0.062
5	25	25%	0	47	65	0.062	77	0.031
			5	47	65	0.062	79	0.047
			10	47	65	0.062	79	0.031
		50%	0	47	117	0.313	103	0.078
			5	47	117	0.312	103	0.047
			10	47	117	0.328	104	0.079

5. Conclusion

Physical consolidation involves migrating existing servers onto a few large systems for the purpose of reducing the number of servers that an organization requires. In this paper we presented a two stage heuristic algorithm for the problem. The initial experiments suggest that the two stage heuristic algorithm presented in this paper performs

reasonably well. In the current paper we only dealt with the objective of minimizing the number of target servers used.

Table 7: Comparative evaluation for the case when |J| = 500 and capacity of bins is 200 in all the 3 dimensions

Run	Number of Item-Item constraints	Cardinality of a Item-item constraint (Density)	Number of Bin – Item constraints	Lower Bound (L_3)	Heuristic Solution for the relaxed Problem (H_3)		Heuristic Solution for the Complete Prob.	
				Target Servers Used	Target Servers Used	Time (Sec)	Target Servers Used	Time (sec)
1	5	25%	0	95	106	0.172	146	0.078
			5	95	106	0.11	140	0.093
			10	95	106	0.109	140	0.344
		50%	0	95	205	0.453	218	0.109
			5	95	205	0.422	218	0.094
			10	95	205	0.422	220	0.109
2	10	25%	0	94	117	0.156	148	0.109
			5	94	117	0.172	150	0.109
			10	94	117	0.125	150	0.109
		50%	0	94	179	0.75	190	0.172
			5	94	179	0.703	191	0.125
			10	94	179	1	191	0.078
3	15	25%	0	93	116	0.125	148	0.078
			5	93	116	0.109	148	0.094
			10	93	116	0.188	148	0.11
		50%	0	93	207	1.453	208	0.093
			5	93	207	1.484	208	0.109
			10	93	207	1.485	208	0.078
4	20	25%	0	93	118	0.313	147	0.125
			5	93	118	0.359	147	0.078
			10	93	118	0.297	143	0.109
		50%	0	93	179	1.296	178	0.125
			5	93	179	1.187	178	0.484
			10	93	179	1.047	178	0.094
5	25	25%	0	93	128	0.328	143	0.157
			5	93	128	0.359	143	0.11
			10	93	128	0.359	143	0.078
		50%	0	93	213	1.765	194	0.766
			5	93	213	1.781	194	0.75
			10	93	213	1.875	194	0.093

Table 8: Comparative evaluation for the case when |J| = 1000 and capacity of bins is 200 in the 3 dimensions

Run	Number of Item-Item constraints	Cardinality of a Item-item constraint (Density)	Number of Bin – Item constraints	Lower Bound (L_3)	Heuristic Solution for the relaxed Problem (H_3)		Heuristic Solution for the Complete Prob.	
				Target Servers Used	Target Servers Used	Time (Sec)	Target Servers Used	Time (sec)
1	5	25%	0	188	238	1.265	316	0.312
			5	188	238	0.39	315	0.344
			10	188	238	0.609	315	0.313
		50%	0	188	306	1.328	367	0.219
			5	188	306	1.344	370	0.235
			10	188	306	1.453	369	0.329
2	10	25%	0	186	228	0.656	279	0.235
			5	186	228	0.406	279	0.359
			10	186	228	0.718	280	0.343
		50%	0	186	387	12.781	397	0.219
			5	186	387	7.344	397	0.374
			10	186	387	7.391	397	0.188
3	15	25%	0	185	240	0.344	293	0.203
			5	185	240	0.391	293	0.219
			10	185	240	0.407	295	0.172
		50%	0	185	395	8.968	400	0.187
			5	185	395	8.437	400	0.157
			10	185	395	9.64	400	0.125
4	20	25%	0	187	230	0.687	306	0.109
			5	187	230	0.688	304	0.11
			10	187	230	0.75	302	0.125
		50%	0	187	368	4.906	377	0.125
			5	187	368	5.234	377	0.109
			10	187	368	5.156	377	0.125
5	25	25%	0	187	228	1.078	298	0.125
			5	187	228	1.063	298	0.219
			10	187	228	1	297	0.125
		50%	0	187	407	13.063	385	0.157
			5	187	407	15.14	385	0.203
			10	187	407	16.328	385	0.125

A more realistic scenario would be minimizing the total cost of purchasing the target servers. Currently we are working towards building a model and a solution procedure that can incorporate the cost information as well. Moreover, vendors often offer quantity discounts on bulk orders. Integrating purchase decisions with server consolidation issues gives rise to a completely different genre of problems that will be interesting to look at. We are also looking at probabilistic frameworks which can model the workloads more realistically.

References

[1] UNIX Server Consolidation Survey, Press Release: http://63.247.141.49/~gcg/index.php?option=com_content&task=view&id=24&Itemid=50

[2] Phelps, J. (2004), "CIO Update: Server consolidation can offer a range of benefits", White Paper, Gartner Inc.

[3] Gartner Research, (2002), "Server Consolidation: Benefits & Challenges".

[4] Ajiro, Y., and Tanaka, A., (2007) "A Combinatorial Optimization Algorithm for Server Consolidation", The 21st Annual Conference of the Japanese Society for Artificial Intelligence.

[5] Zhang, A., Safai, F., and Beyer, D., (2005) "Applying Bin-Packing Algorithms to Server Consolidation", Informs annual meeting in San Francisco.

[6] Zhang, A., "A High-Dimensional Bin-Packing Algorithm for Server Consolidation", http://www2.twgrid.org/event/isgc2006/Presentation%20Material/0504-Industry%20Track/Industry-Meichun-05042006.pdf

[7] Chu, C., and La, R., (2001), "Variable-sized bin packing: Tight absolute worst-case performance ratios for four approximation algorithms", SIAM Journal of Computing, 30, 2069–2083.

[8] Kang, J., and Park, S., (2003) "Algorithms for the variable sized bin packing problem", European Journal Operational Research, 147, 365–372.

[9] Gendreau, M. Laporte, G. and Semet, F., (2004), "Heuristics and lower bounds for the bin packing problem with conflicts", Computers and Operations Research , 31, 347 – 358.

[10] Epstein, L. and Levin, A., "On bin packing with conflicts", math.haifa.ac.il/lea/bpc.pdf

[11] Jansen, K. (1999), "An approximation Scheme for bin packing with conflicts", Journal of Combinatorial Optimization, 3, 363--377.

[12] Bichler, M., et. al. (2006), "Capacity Planning for Virtualized Servers", 16th 16 Workshop on Information Technologies and Systems, Milwaukee, USA.

[13] Garey, M. R., and Johnson, D. S., (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H. Freeman, ISBN 0-7167-1045-5.

[14] Welsh, D.J.A., and Powell, M.B., (1967), "The upper bound for the chromatic number of a graph and its application to timetabling problems", The Computer Journal, 11, 41-47.

[15] Jansen., K, and Oehring, S., (1997), "Approximation algorithms for time constrained scheduling", Information and Computation, 132, 85 – 108.

[16] Falkenauer E., (2004), "A Hybrid Grouping Genetic Algorithm for Bin Packing", Journal of Heuristics, 2, 5-30