OPTIMAL ALLOCATION OF JOBS TO THE ONLINE CLOUD SERVERS

PROJECT 1

Dynamic Programming OR 709

By: Muhammad Ali Haider (mhaider2@ncsu.edu)
Submitted to: Dr. Mayorga

Date 4th May 2018

Table of content

Table of content

1. Introduction & Literature Review

Cloud computing is the on-demand delivery of computing power, database storage, applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing. Companies offering these computing services are called cloud providers and typically charge for cloud computing services based on usage, similar to how we are billed for water or electricity at home. The Cloud computing is a growing popular information technology (IT) service paradigm (Met et al, 2011; Sun, 2010). Google, Amazon, Microsoft, IBM have already started their cloud computing infrastructure to run their cloud business in these service model and many small-scale startups are following the trend (Chiang, 2013).

For any cloud provider, an effective resource scheduling policy is expected to conserve operating costs. Besides reducing the hardware cost, it also saves on electricity and other utilities which contributes to a significant portion of the operational expenses in large data centers. Studies have found that servers in many existing data centers are often severely underutilized (Armbrust, 2009; Siegele 2008). Therefore, a policy issue remains as for how to decide on the mapping of jobs allocation so that the resource demands are met while minimizing the total number of servers required. This is challenging when the resource needs of jobs are heterogeneous due to the diverse set of applications they run and vary with time as the workloads grow and shrink.

In this work, a formulation is proposed for the resource allocation problem in shared hosting platforms for static workloads with servers that provide multiple types of resources. High server utilization is paramount for justifying server leasing costs (Brown, 2007). Since running services on dedicated servers achieve performance isolation which leads to low server utilization. There lies a strong incentive for sharing cluster resources among services, establishing the so-called shared hosting platforms. The challenge is then to allocate appropriate resource shares to services that have conflicting resource demands.

Like most resource allocation problems, our problem is related to bin packing. However, it is also related to a variant in which bins and items are multi-dimensional vectors or vector packing. Therefore vector packing would be a more appropriate term to call this problem that will be solved using kansap methodology. The challenges of resource allocation and of application modeling for multi-tier applications have been studied. Versions of standard greedy algorithms (First Fit, Best Fit, Worst Fit, Next Fit, etc.) have been proposed for vector packing (Kao, 1977; Maruyama, 1977). Vector packing heuristics beyond the standard greedy ones are proposed in Leinberger (1999) and in Maruyama (1977). These heuristics do not provide tight performance guarantees but may exhibit good average-case behavior. Genetic algorithms have also been used to solve vector packing problems that arise naturally from resource allocation problems (Rolia, 2003; Gmach, 2009a; Gmach 2009b) by optimizing the number of servers in use. Several systems have been proposed to manage resources in shared hosting platforms (Stillwell, 2010). Cloud computing has been the target of research during the last years (Chiang, 2013).

This work is also addressing a real-time problem of a startup firm situated in California that aims to reduce its number of servers by optimal allocation of jobs to each of the servers. The minimum number of servers used at any time is desirable because it cuts the operating cost and leasing cost associated with each of the servers. Dynamic programming formulation is coded using MATLAB and the results compared against integer programming method using GAMS software.

2. Model description

The cloud computing has become a significant technology trend. The cloud servers are often massive in size and to allocate them with right jobs is a big challenge. The jobs can be running game, video streaming or file download, depending on user's request. As the number of servers requirement increases the rental, energy and other associated cost increase. In order to minimize the cost, it is highly necessary to minimize the total number of servers usage by the optimal allocation of jobs, thus fully realizing the resource capacity of the servers.

It's like a bin packing problem. The items are to be packed into bins in an online manner such that the total size of the items in each bin does not exceed the bin capacity at all times. Where the jobs and cloud servers correspond to the items and bins respectively in the problem. The objective is to minimize the maximum number of bins used in the packing that can be translated into minimizing the total number of servers in use. Each job needs some amount of resources for execution and has to be assigned to run on a cloud server. Each cloud server has a fixed resource capacity that restricts the total amount of resources needed for all the jobs running on the server. Jobs should be assigned to a cloud server that has enough CPU, memory and bandwidth resources to run the requested job. Several jobs can share the same cloud server provided that the server's resources are not saturated. Running each job demands a certain amount of server's resources and the resource requirement can be different.

For this problem, we will have to allocate jobs that each has different CPU, memory and bandwidth requirements, such as to minimize the total number of servers required which are subjected to same

CPU, memory, and bandwidth capacity constraints. Therefore it will be a 3D vector packing problem with the jobs having dimensions of CPU, storage and bandwidth units. It is a combinatorial complexity type problem that will be formulated using dynamic programming. Figure 1 shows a 2D vector packing problem for illustration.

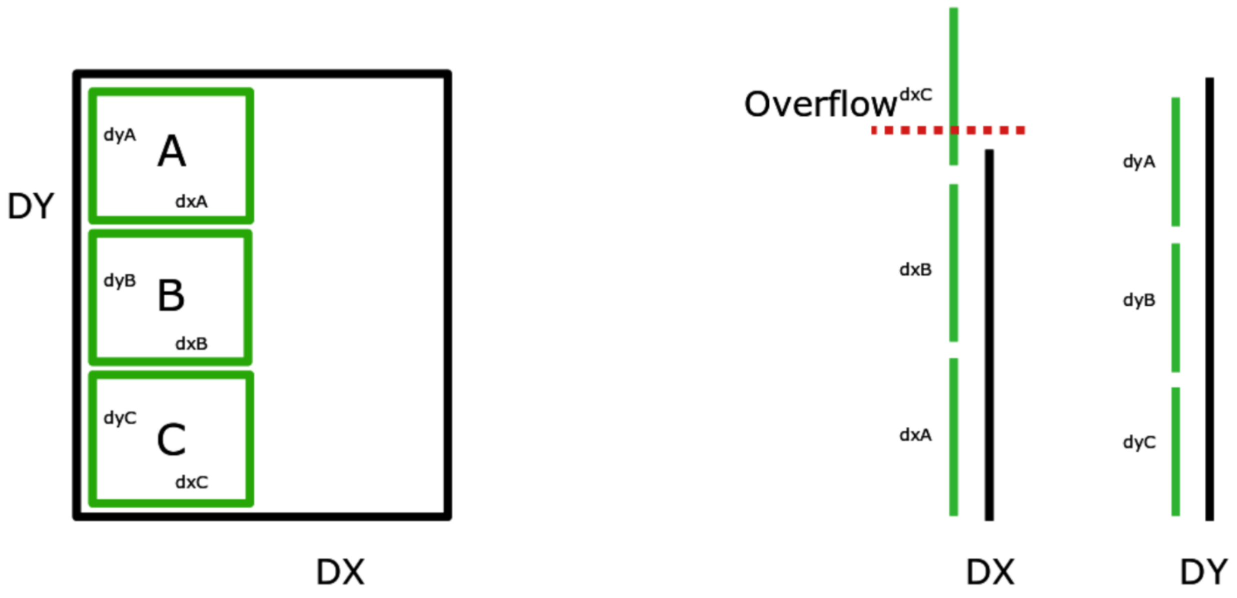


Figure 1 2D Vector packing problem illustration

In reality, the sever offer several resources at a time thus increasing the complexity required for optimization. The assumptions made for our problem are discussed below.

Assumptions

1. The jobs are static, i.e we know there is a certain number of jobs that have to be processed. In a dynamic situation, jobs keep on coming demand over time according to the with certain probabilities.

2. It is assumed that all the jobs consume an equal amount of time.

3. The benefits associated with each dimension is the same and linear.

Model Formulation

3. Model Formulation

1. $\mathbf{w_1, w_2, w_3}$ will be the maximum capacity of a server in each of the dimensions
2. **x, y, z** are the dimensions which are bandwidth, storage and processing respectively
3. $\mathbf{r_{xn}, r_{yn}, r_{zn}}$ will be recourses consumed by job 'n' in each of the dimensions (x,y,z) respectively.

Stages:

n : Number of jobs that have to be satisfied by assigning them to the server

States:

$\mathbf{\{x_n, y_n, z_n\}}$ **:** Where x_n, y_n, z_n are total resources that have been consumed in each of the dimensions respectively from stage n to stage N+1

Actions:

$\mathbf{a_n}$**:** To allocate job 'n' to the server or not

$a_n \in \{0,1\}$

Recursive optimality equation:

$\mathbf{f_n(x_n, y_n, z_n)}$ **=** Total value of $x_n + y_n + z_n$ at stage n and is to be maximized.

x_n, y_n, z_n = $\{x_{n+1} - r_{xn},\ y_{n+1} - r_{yn},\ z_{n+1} - r_{zn}\}\ if\ a_n = 1$

x_n, y_n, z_n = $\{x_{n+1},\ y_{n+1},\ z_{n+1}\}\ if\ a_n = 0$

f_n(x_n, y_n, z_n) =

$$\max \begin{cases} r_{xn} + r_{yn} + r_{zn} + f_{n+1}(x_{n+1} - r_{xn},\ y_{n+1} - r_{yn},\ z_{n+1} - r_{zn}) & if\ a_n = 1 \\ f_{n+1}(x_{n+1},\ y_{n+1},\ z_{n+1}) & if\ a_n = 0 \end{cases}$$

Solving small scale problem

4. Solving small scale problem

Table 1 below shows the data structure for a small scale problem in which there are four jobs and

Table 1 Small scale problem data

Job number n	**Bandwidth x**	**Storage y**	**Processor z**
1	10	3	4
2	6	1	2
3	4	8	2
4	5	4	3
Max Capacity	12	10	10

Benefits	1	1	1

each having different bandwidth, storage and processor requirements. Server's maximum capacity for each of the dimensions (bandwidth, storage and processor requirements) are also shown in the table. Since each server is subjected towards maximum allowable capacity constraints for each of the bandwidth, storage and processor resources. The goal is to pack jobs such as to use the minimum number of servers. The dynamic programming problem for this is solved below.

$f_5(0,0,0) = 0$

$f_4(5,4,3) = (5 + 4 + 3)(1) + f_5(0,0,0) = 12$ if a =1

$f_4(0,0,0) = (5 + 4 + 3)(0) + f_5(0,0,0) = 0$ if a =0

$f_3(5,4,3) = (4 + 8 + 2)(0) + f_4(5,4,3) = 12$ if a =0

$f_3(4,8,2) = (4 + 8 + 2)(1) + f_4(0,0,0) = 14$ if a =1

$f_3(0,0,0) = (4 + 8 + 2)(0) + f_4(0,0,0) = 0$ if a =0

$f_2(11,5,5) = (6 + 1 + 2)(1) + f_3(5,4,3) = 21$ if a =1

$f_2(5,4,3) = (6 + 1 + 2)(0) + f_3(5,4,3) = 12$ if a =0

$f_2(10,9,5) = (6 + 1 + 2)(1) + f_3(4,8,2) = 23$ if a =1

$f_2(4,8,2) = (6 + 1 + 2)(0) + f_3(4,8,2) = 14$ if a =0

$f_2(6,1,2) = (6 + 1 + 2)(1) + f_3(0,0,0) = 9$ if a =1

$f_2(0,0,0) = (6 + 1 + 2)(0) + f_3(0,0,0) = 0$ if a =0

$f_1(11,5,5) = (10 + 3 + 4)(0) + f_2(11,5,5) = 21$ if a =0

$f_1(5,4,3) = (10 + 3 + 4)(0) + f_2(5,4,3) = 12$ if a =0

$f_1(10,9,5) = (10 + 3 + 4)(0) + f_2(10,9,5) =$ **23 *** if a =0

$f_1(4,8,2) = (10 + 3 + 4)(0) + f_2(4,8,2) = 14$ if a =0

$f_1(6,1,2) = (10 + 3 + 4)(0) + f_2(6,1,2) = 9$ if a =0

$f_1(0,0,0) = (10 + 3 + 4)(0) + f_2(0,0,0) = 0$ if a =0

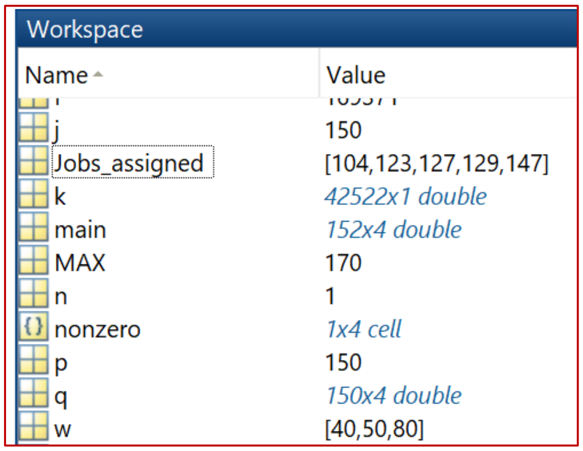
$f_1(10,3,4) = (10 + 3 + 4)(1) + f_2(0,0,0) = 17$ if a =1

Hence the results show that the maximum utilization of the server is 23 units that can be achieved by allocating Jobs 2 and 3 to a server. The same formulation can be repeated after removing jobs 2 and 3 from the set of jobs to optimize the allocation of the second server. This procedure can be continued until all the jobs have been allocated and this way minimum number of the servers can be ensured since every server is being optimally utilized to reduce the total number of servers required.

Same procedure and methodology were applied to the large-scale problem that consisted of a set of 150 jobs and different resource constraints and the results are shown in the next section. Both small and large problems were solved using MATLAB.

4. Results

Figure 2 shows the result where the maximum capacity in each of the three dimension is 40,50 and 80 respectively, as represented by variable 'w'. The maximum utilization of benefit associated with the first server will be 170 which is denoted by the variable 'MAX'. The variable 'Jobs _assigned' represents the job number that has to be assigned to the server to ensure the maximum utilization. So in this case among 150 jobs the job number 104, 123, 127, 129 and 147 have to be allocated to the server to ensure maximum utilization. The optimal number of jobs for the second server can be known by repeating the same procedure but removing the job number 104, 123, 127, 129 and 147 from the set.



Workspace

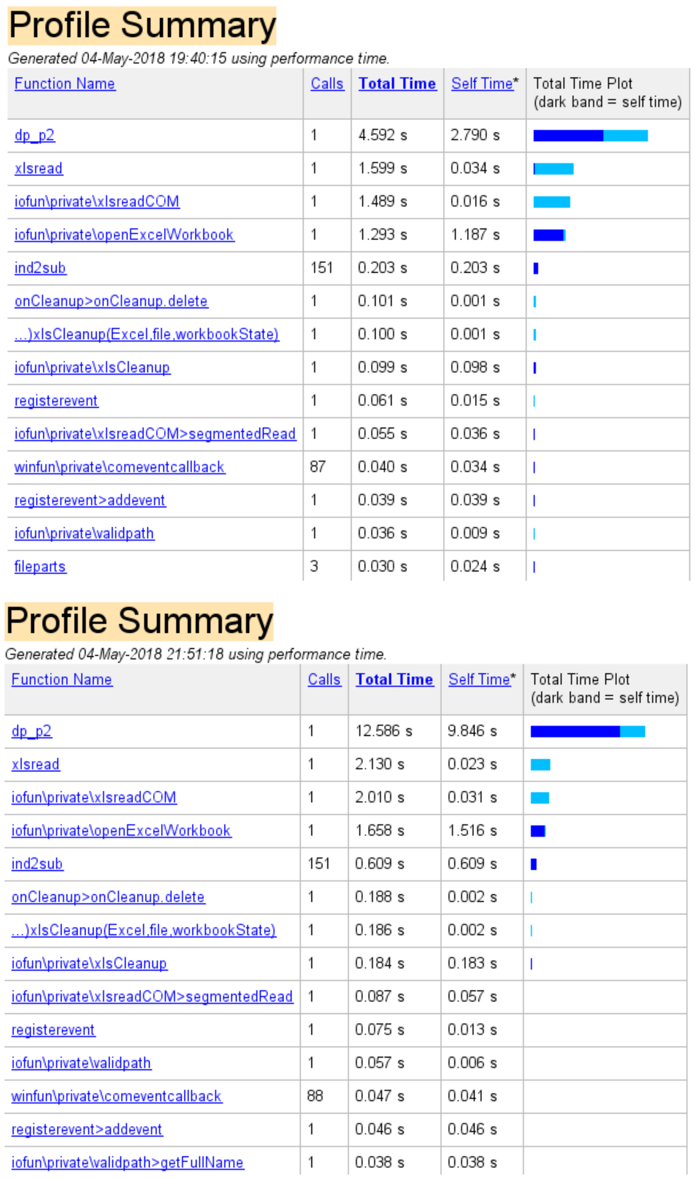
Name	Value
j	150
Jobs_assigned	[104,123,127,129,147]
k	*42522x1 double*
main	*152x4 double*
MAX	170
n	1
nonzero	*1x4 cell*
p	150
q	*150x4 double*
w	[40,50,80]

Figure 2 Large scale problem results for first server

For the small scale problem both the dynamic programming and integer programming gave the same result. Integer programming was also formulated for the large problem and the coding was done on the GAMS software. The results of the integer programming showed that Jobs number 98, 101, 118, 124 has to be allocated to server one to get a utilization factor of 167. The integer programming was much faster than the dynamic programming but the result of the dynamic programming was more accurate. Further comparison of the results will be discussed in the next heading.

5. Discussion

It can be seen from figure 3 that increasing the capacity constraint limits have an exponential affect on the time consumed to run the code. When the capacity constraints were 40, 50 and 80 the time consumed was 4.592 seconds however when the capacity was made 50, 60 and 100. Time increased to 12.586 seconds. That is 25%, 20% and 25% increase in the capacity respectively caused 174.2% increase in time. Different other capacity constraint were established and it was observed that time increased exponentially with the increase in the constraint value.



Profile Summary

Generated 04-May-2018 19:40:15 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
dp_p2	1	4.592 s	2.790 s	
xlsread	1	1.599 s	0.034 s	
iofun\private\xlsreadCOM	1	1.489 s	0.016 s	
iofun\private\openExcelWorkbook	1	1.293 s	1.187 s	
ind2sub	151	0.203 s	0.203 s	
onCleanup>onCleanup.delete	1	0.101 s	0.001 s	
...)xlsCleanup(Excel,file,workbookState)	1	0.100 s	0.001 s	
iofun\private\xlsCleanup	1	0.099 s	0.098 s	
registerevent	1	0.061 s	0.015 s	
iofun\private\xlsreadCOM>segmentedRead	1	0.055 s	0.036 s	
winfun\private\comeventcallback	87	0.040 s	0.034 s	
registerevent>addevent	1	0.039 s	0.039 s	
iofun\private\validpath	1	0.036 s	0.009 s	
fileparts	3	0.030 s	0.024 s	

Profile Summary

Generated 04-May-2018 21:51:18 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
dp_p2	1	12.586 s	9.846 s	
xlsread	1	2.130 s	0.023 s	
iofun\private\xlsreadCOM	1	2.010 s	0.031 s	
iofun\private\openExcelWorkbook	1	1.658 s	1.516 s	
ind2sub	151	0.609 s	0.609 s	
onCleanup>onCleanup.delete	1	0.188 s	0.002 s	
...)xlsCleanup(Excel,file,workbookState)	1	0.186 s	0.002 s	
iofun\private\xlsCleanup	1	0.184 s	0.183 s	
iofun\private\xlsreadCOM>segmentedRead	1	0.087 s	0.057 s	
registerevent	1	0.075 s	0.013 s	
iofun\private\validpath	1	0.057 s	0.006 s	
winfun\private\comeventcallback	88	0.047 s	0.041 s	
registerevent>addevent	1	0.046 s	0.046 s	
iofun\private\validpath>getFullName	1	0.038 s	0.038 s	

Figure 3 Time elapsed be increasing the capacity constraints

Apart from the capacity constraints the effect of the number of jobs on time was also recorded and it was found that time increased slightly. Time elapsed by running the code when total number of jobs were, 4, 150 and 250, varied only slightly. Therefore the total number of jobs had a little impact on the time consumption in comparison with the size of the capacity constraints.

Figure 4 shows a Pareto analysis done for the jobs assigned to the individual server. The maximum value that the server can utilize is simply the addition of all the capacity constraints. Therefore for this case, the maximum possible utilization be each server was 170. The higher the value means less space was left unfilled in the server and more optimally the jobs have been allocated to it. Therefore such Pareto analysis can help to identify the jobs that are not good at server utilization and can be get rid of.

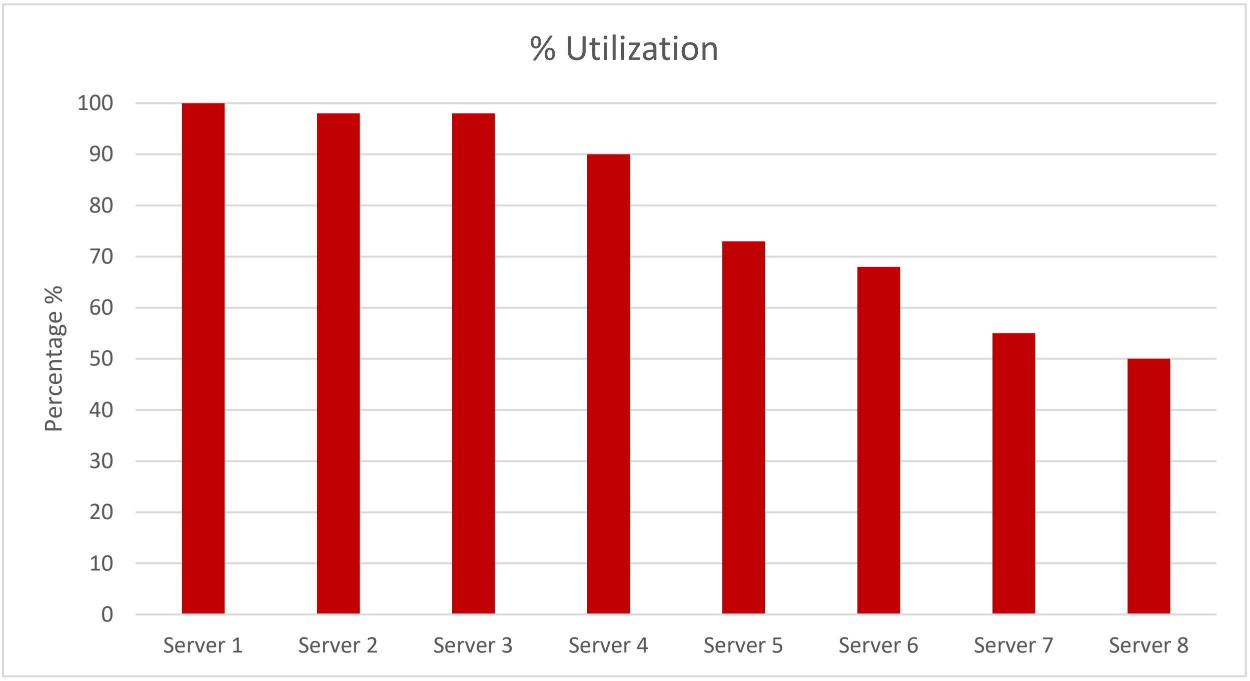


Figure 4 Percentage server utilization and Pareto analysis

It also is observed that at each iteration the total percentage utilization is decreasing because the ‘good jobs’ that can fit in the server better are taken by the first servers. Second servers take the second best set of jobs and it goes on. The last server gets the jobs which are difficult to fit due to the “bad shape” or bad dimensions for optimal allocation. Next section will be discussing the conclusion of the analysis.

6. Conclusion

The optimal allocation of the server was addressed using dynamic programming. It is a huge problem for the companies who use the cloud server platform to host the application. Better utilized servers mean more jobs that have been assigned to it and hence it reduces the overall cost that is associated with the server (Both operating and leasing).

Dynamic programming was formulated for this problem and MATLAB was used for the coding. The results showed what jobs that have to be assigned to each server for optimal allocation to reduce the total number of the servers. From the sensitivity analysis, it was concluded that increase in the size of the capacity constraint increases the time required to run the code exponentially. While the increase in the total number of jobs increased the code running time slightly.

Overall dynamic programming is a powerful tool that can ensure efficient use of servers. Dynamic programming can even address the same problem with lesser assumptions. The aspect dynamic nature of the real-time problem can be incorporated into the problem and each of the jobs can be assigned different weights or significance level.

References

Armbrust, Michael, et al. Above the clouds: A berkeley view of cloud computing. Vol. 4. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

Brown, Richard E., et al. Report to congress on server and data center energy efficiency: Public law 109-431. No. LBNL-363E. Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA (US), 2007.

Chiang, Yi-Ju, Yen-Chieh Ouyang, and Ching-Hsien Hsu. "An Optimal Cost-Efficient Resource Provisioning for Multi-servers Cloud Computing." Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on. IEEE, 2013.

Gmach, Daniel, et al. "Resource pool management: Reactive versus proactive or let’s be friends." Computer Networks 53.17 (2009a): 2905-2922.

Gmach, Daniel. Managing shared resource pools for enterprise applications. Diss. Technische Universität München, 2009b.

Kou, Lawrence T., and George Markowsky. "Multidimensional bin packing algorithms." IBM Journal of Research and development 21.5 (1977): 443-448.

Leinberger, William, George Karypis, and Vipin Kumar. "Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints." Parallel Processing, 1999. Proceedings. 1999 International Conference on. IEEE, 1999.

Maruyama, K., S. K. Chang, and D. T. Tang. "A general packing algorithm for multidimensional resource requirements." International Journal of Computer & Information Sciences 6.2 (1977): 131-149.

Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011).

Rolia, Jerry, Artur Andrzejak, and Martin Arlitt. "Automating enterprise application placement in resource utilities." International Workshop on Distributed Systems: Operations and Management. Springer, Berlin, Heidelberg, 2003.

Siegele, Ludwig. Let it rise: A special report on corporate IT. Economist Newspaper, 2008.

Stillwell, Mark, et al. "Resource allocation algorithms for virtualized service hosting platforms." Journal of Parallel and distributed Computing 70.9 (2010): 962-974.

Sun, Yuanhui, et al. "An architecture model of management and monitoring on cloud services resources." Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on. Vol. 3. IEEE, 2010.

Appendix

Appendix 1

This is the data that was generated randomly using normal distribution and was used for the large problem. The top row represents the capacity constraint. This can be changed by changing the value in the top row of the excel file accordingly. Name of the excel file is 'random_data'. In addition to it, the number of jobs and its respectively sizes can be adjusted by manipulating the data in the file accordingly.

Capacity	*50*	*60*	*100*
Job Number	**Bandwidth (Hertz)**	**Storage (Giga bytes)**	**CPU (Giga byte)**
1	9	20	34
2	5	20	35
3	8	21	13
4	3	12	11
5	9	20	27
6	1	17	10
7	4	14	22
8	1	8	22
9	2	21	32
10	10	13	21
11	9	18	19
12	4	21	26
13	12	8	28
14	1	9	22
15	6	17	17
16	5	16	12
17	10	23	24
18	10	22	15
19	3	20	9
20	6	7	29
21	6	8	14
22	8	8	20
23	9	22	27
24	10	24	18

25	4	19	28
26	9	9	19
27	8	20	27
28	2	9	27
29	2	9	20
30	6	19	8
31	12	13	17
32	5	19	19
33	8	21	15
34	3	18	13
35	10	21	31
36	4	11	20
37	7	20	32
38	9	25	18
39	11	23	29
40	12	8	19
41	7	13	30
42	2	14	29
43	2	20	18
44	4	18	14
45	11	21	30
46	4	13	34
47	10	10	17
48	3	8	26
49	12	21	20
50	5	10	31
51	3	14	29
52	4	17	12
53	8	11	32
54	6	19	35
55	5	16	22
56	10	9	32
57	8	21	24
58	7	8	12
59	12	12	13
60	4	11	19
61	10	17	28
62	10	8	31
63	5	14	30
64	7	8	16
65	1	9	22
66	1	21	10
67	7	12	11

68	10	18	11
69	12	25	27
70	2	15	21
71	7	20	13
72	6	21	21
73	1	15	12
74	5	19	9
75	2	9	31
76	10	24	23
77	4	10	34
78	7	12	27
79	2	22	24
80	8	16	30
81	4	21	32
82	8	14	35
83	9	12	8
84	9	7	32
85	6	19	25
86	2	15	35
87	3	15	22
88	11	18	21
89	2	8	30
90	10	13	14
91	7	21	21
92	12	20	33
93	1	9	24
94	6	9	31
95	2	8	28
96	12	7	24
97	1	15	14
98	10	19	26
99	10	20	10
100	11	17	25
101	2	9	26
102	5	19	28
103	4	9	32
104	10	9	35
105	6	8	29
106	11	9	24
107	3	10	33
108	4	10	24
109	2	13	8
110	2	13	11

111	11	11	32
112	7	11	21
113	7	23	31
114	2	20	13
115	11	17	23
116	8	10	25
117	5	11	8
118	7	8	25
119	5	24	18
120	1	20	9
121	3	17	21
122	2	12	13
123	3	10	11
124	3	18	13
125	6	25	12
126	1	10	13
127	11	11	9
128	12	14	25
129	6	8	15
130	6	19	23
131	5	14	27
132	11	25	21
133	5	14	23
134	2	18	20
135	10	9	11
136	5	14	21
137	3	10	31
138	5	21	32
139	2	23	15
140	2	13	13
141	12	20	23
142	12	12	25
143	7	17	19
144	1	22	13
145	3	18	34
146	5	13	10
147	10	12	10
148	1	15	11
149	1	15	12
150	3	13	25

Appendix 2

The MATLAB code is attached in a separate file with the instruction to run it is in the read me file. The variable 'Jobsassigned' will display the set of jobs that have to be assigned to the server. Variable 'w' represents the capacity constraint and the variable 'MAX' represents the addition of all the resource dimensions that have been utilized.