Co-allocation with Communication Considerations in Multi-cluster Systems

John Ngubiri and Mario van Vliet

Nijmegen Institute for Informatics and Information Science
Radboud University Nijmegen
Toernooiveld 1, 6525 ED, Nijmegen,
The Netherlands
{ngubiri,mario}@cs.ru.nl

Abstract. Processor co-allocation can be of performance benefit. This is because breaking jobs into components reduces overall cluster fragmentation. However, the slower inter-cluster communication links increase job execution times. This leads to performance deterioration which can make co-allocation unviable. We use intra-cluster to inter-cluster communication speed ratio and job communication intensity to model the job execution time penalty due to co-allocation. We then study viability of co-allocation in selected job and system based instances. We also study performance variation with selected job stream parameters. We observe that co-allocation is viable so long as the execution time penalty caused is relatively low. We also observe that the negative performance effect due to co-allocation is felt by the entire job stream rather than only the (few) co-allocated jobs. Finally, we observe that for every value of communication time penalty, there is a job size s^*, where if all jobs whose size is greater than s^* are co-allocated, we get the best performance.

1 Introduction

The load, like the processing power on supercomputers, has been growing fast over the past decade [17]. Supercomputer resources, therefore, remain scarce. This calls for efficient scheduling of competing (job) requests so as to put the resources to optimal utility.

A lot of research has been done in parallel job scheduling [7]. This has mostly been on shared memory computers, distributed memory multi-processors, clusters, multi-cluster systems and the grid. Currently, clusters are the most popular supercomputing platform with over 70% of the top 500 supercomputers [21]. This can be attributed to their cost-effectiveness, scalability and fault tolerance. Multi-cluster systems are set up by combining multiple clusters into a bigger computational infrastructure. Different clusters are connected by wide-area links so that they can collaboratively process large jobs.

Large jobs may be broken into components and each component (simultaneously) processed in a different cluster [4] (co-allocation). Co-allocation is beneficial because it makes use of scattered resources and hence minimizes

E. Luque, T. Margalef, and D. Benítez (Eds.): Euro-Par 2008, LNCS 5168, pp. 243–252, 2008.

fragmentation. However, co-allocated components communicate across (slower) inter-cluster links which increase their run times leading to poorer performance. Co-allocation may become unviable.

The negative effect of co-allocation is the extension of the job run time. We use the ratio of intra to inter cluster speed and job communication intensity compute the execution time penalty. We investigate parameter bounds within which co-allocation is beneficial. We investigate optimal parameter combinations for co-allocation as well as their variation with other job stream parameters. We also investigate the effect of dispersion in job communication intensities.

We observe that (*i*) co-allocation is viable if the execution time penalty caused is low; (*ii*) the threshold of co-allocation viability is a function of intra/inter-cluster speed ratio and jobs' communication intensity; (*iii*) entire job stream results are insufficient to deduce co-allocation viability; (*iv*) for any execution time penalty, there exists a job size $s*$ where if all jobs with size greater than $s*$ are co-allocated, we get best results; and (*v*) due to possible heterogeneous communication pattern, co-allocation may not be as viable as previously implied.

In the rest of the paper, we discuss related work, the research model and scheduling algorithm used in Sections 2, 3 and 4 respectively. The experimental instances used are discussed in Section 5. We investigate the viability of co-allocation in Section 6 and the effect of selected parameters on performance and viability of co-allocation in Section 7. We study how the dispersion of communication intensity among the jobs affects performance in Section 8 and make conclusions and suggestions for future work in Section 9.

2 Related Work

2.1 Communication and Its Effect on Co-allocation

Ignoring communication is one of the common pitfalls in parallel job scheduling [8]. It leads to artificially good but misleading deductions. In multi-cluster systems, communication is more problematic due to relatively slow wide-area speeds. Parallel jobs consist of tasks that communicate as they execute. Communication may be synchronous or asynchronous. In synchronous communication, the tasks communicate after specific time intervals. Job execution is broken into a sequence of processing and communication steps. In asynchronous communication, tasks process independently but different pairs occasionally communicate.

Bucur and Epema [3] studied an all-to-all synchronous communication model with the First Come First Served (FCFS) scheduler. They focused on the intra/inter cluster speed ratio to determine the execution time penalty. They observed that there exists a communication ratio beyond which co-allocation is unviable. Sonmez et al. [13] considered a case where the penalty is proportional to the number of clusters the job is processed. They proposed placement policies that minimize the effect of communication. Components are placed in such a way that the number of clusters processing a job is minimized. Jones et al. [11][10] studied the effect of communication from a bandwidth point of view. They consider a case where a job needs certain amount of bandwidth to process

for the initially allotted time. If within the process of execution the link gets over saturated and the job uses less bandwidth, the job's rate of processing lowers in proportion to the bandwidth shortfall. This leads to a job execution time which is pegged on link states as the job processes.

2.2 Workloads

Using unrealistic workloads leads to unrealistic deductions [8]. Workloads are generated synthetically or from archived logs of existing supercomputers [18, 19]. Synthetically, job characteristics are generated from statistical distributions estimated by workload trace studies. Since they are easy to generate and extrapolate, the stable state can easily be achieved. However, coming up with an accurate distribution is hard; it may therefore be preferable to use archived logs. Workload logs avail job characteristics without necessarily knowing their statistical distribution. However, the job stream may be too short to generate a stable state and extrapolation is hard. The load is also hard to vary. Changing the inter-arrival time for example, as a means of changing load, changes the daily and weekly peaks which is unrepresentative of the real life situation of different traffic. Aspects like communication patterns are not archived. Traces may also contain flurries which highly affect results [20].

2.3 Performance Evaluation

A performance metric used in parallel job scheduling has to put the scheduling scenario into consideration [8]. The deductions made are sometimes more metric than system dependant. The Shortest Job First (SJF) scheduling algorithm, for example, gives an impressive performance when measured by throughput despite obvious starvation of long jobs. Some metrics may have different implications depending on the system studied. Average waiting time (AWT) and average response time (ART) have similar performance implications for dedicated processing but have different implications for time sliced/preemptive cases. Making performance conclusions based on entire job stream metric value may hide internal performance details. Grouping jobs by the characteristics that constrain schedulability provides deeper understanding of performance [5, 14, 15, 16].

3 Research Model

We consider a system of homogeneous clusters $C_1, C_2, \cdots C_n$ that process by pure space slicing. They are connected by wide-area communication links that are slower than intra-cluster links. They are served by one queue and one scheduler.

3.1 Job Stream

We use synthetic online jobs with exponentially distributed inter-arrival and execution times. These distributions have also been used in previous related

work [2][9]. Job execution time is finite but unknown to the scheduler. Job sizes are generated from the distribution $D(q)$ $(0 < q < 1)$ where the probability p_i that a job has size i is $(\frac{q^i}{Q})$ $\frac{3q^i}{Q}$ if i is (not) a power of 2. It is defined over an interval $[n_1,\ n_2]$ $(0 < n_1 < n_2)$. Parameter q varies mean job size while Q is in such a way that p_i sums up to 1. It favors small jobs and those whose size is a power of 2 which is known to be a realistic choice [6].

Jobs with size greater than a threshold *thres* are broken into components and co-allocated. If a job of size s is broken into k components, one component has width $s-(k-1)\lfloor\frac{s}{k}\rfloor$ while the other $k-1$ has $\lfloor\frac{s}{k}\rfloor$ each.

3.2 Communication

We consider a synchronous communication pattern (like in [3]). The execution time (T_E) of the job is made up of the processing (T_P) and communication (T_C) components. Like in [10][11], we represent the total execution time on one cluster as:

$$T_E = T_C + T_P \tag{1}$$

If $T_C = \alpha T_E$, where α $(0 < \alpha < 1)$ represents the job communication intensity, then (1) can be rewritten as

$$T_E = \alpha T_E + (1-\alpha)T_E \tag{2}$$

If the ratio of the time taken by an inter-cluster packet to an intra-cluster packet is $(1+\lambda)$ $(\lambda > 0)$, then co-allocating a job increases T_C by $(1+\lambda)$. The execution time of a co-allocated job T'_E is therefore given by $T'_E = \alpha(1+\lambda)T_E + (1-\alpha)T_E = (1+\alpha\lambda)T_E$. If we define $\psi = \alpha\lambda$, then;

$$T'_E = (1+\psi)T_E \tag{3}$$

This model is similar to the fixed penalty approach employed in [3] though it considers two aspects instead of one.

4 Scheduling Algorithm and Placement Policy

We use the Fit Processors First Served (FPFS) [1] scheduling algorithm.

In FPFS, jobs are queued in their arrival order. The scheduler starts from the head and searches deeper into the queue for the first job that fits into the system. In case one is found, it jumps all jobs ahead of it and starts processing. If none is found, the scheduler waits either for a job to finish execution or a job to arrive and the search is done again. To avoid possible starvation of some jobs, the scheduler limits (to $maxJumps$) the number of times a job at the head of the queue can be jumped. After being jumped $maxJumps$ times, no other job is allowed to jump it until enough processors have been freed to have it scheduled. We use FPFS (x) to represent FPFS when $maxJumps = x$.

To map components onto clusters, we use the Worst Fit (WFit) policy. In the WFit policy, the k^{th} widest component is processed in the k^{th} freest cluster. It distributes free processors evenly among the clusters.

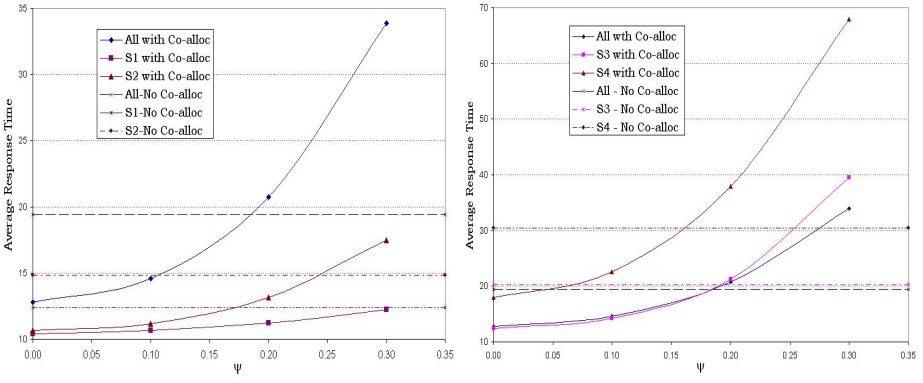

Fig. 1. Comparing co-allocation with no co-allocation

5 Experimental Set Up

We consider a system of 5 homogeneous clusters of 20 nodes each. The jobs are generated from $D(0.85)$ over the interval $[1, 19]$. Jobs have a mean execution and inter-arrival time of 10 and 0.54 respectively. This leads to a load of 0.786 (when $\psi = 0$). ART is used to measure performance. Performance evaluation is done for the entire job stream as well as for job-size based groups. We use four approximately equal size-based groups S_1, S_2, S_3 and S_4. They are bounded by the (size) lower quartile, median and upper quartile. They have a numerical representation of 25.3%, 27.7%, 22.9% and 24.1% and load representation of 6.0%, 27.7%, 23.1% and 57.6% respectively.

6 Viability of Co-allocation

Co-allocation studied without considering communication (like in [12]) is always viable. This is due to the packing advantage of breaking up large jobs but no communication penalty. We now compare performance at different ψ values with a case of no co-allocation. We use $thres = 10$ (effect of $thres$ studied in 7). Where co-allocation is used, jobs where $size > thres$ are broken into 4 components. We present the results in Figure 1.

In Figure 1, we observe that co-allocation is viable for low ψ. We observe that the value of ψ beyond which co-allocation is unviable is different for different job groups. It is 0.185 for the entire job stream, 1.60 for S_4 and over 0.3 for S_1.

Like in [3], we observe that there is a threshold beyond which co-allocation is not viable. However, (i) the threshold is not only dependant on the intra and inter cluster speed ratio. It also depends on the communication intensity of the jobs. Co-allocation is viable for any speed ratio so long as the jobs' communication intensity is low enough. (ii) The (entire) job stream threshold value is practically misleading. This is because large jobs have a lower threshold and they constitute a larger part of the load in the system.

7 The Effect of System and Job Parameters

7.1 The Effect of *Thres* on Performance

We set $\psi = 0.05$ (investigation of ψ in 7.2), vary *thres* from 3 to 19 (Figure 2).

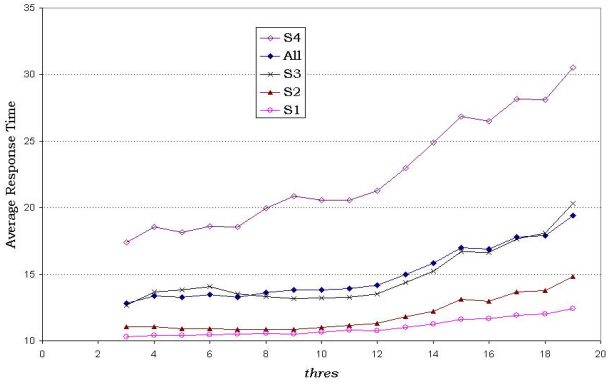

Fig. 2. Performance variation with *thres*

From Figure 2, we observe that increasing *thres* leads to poorer performance. It is therefore better to keep *thres* low. We also observe that all groups take the same trend. The rate of performance deterioration is higher for large jobs. Co-allocation affects both co-allocated and non co-allocated jobs.

7.2 The Effect of ψ on Performance Sensitivity to *Thres*

We now investigate the effect of ψ on the way performance varies with *thres*. Since S_4 jobs perform worst and constitute over half of the load, we consider them more representative. We therefore use S_4 jobs only. We use four values of ψ and summarize the performance trend in Figure 3.

We observe that there exists an optimal value of *thres* for each ψ value. This optimal value increases with ψ. An increase in ψ reduce the *thres* range in which co-allocation is viable.

7.3 The Effect of Load

We now investigate performance trends at different loads. The loads are computed when co-allocation is not employed. Loads are varied by changing the mean inter-arrival time. We consider $\psi = 0.1$ and summarize the performance trends in Figure 4. We observe that an increase in load leads to a poorer performance. Increase in load however does not affect the optimal *thres* value for co-allocation.

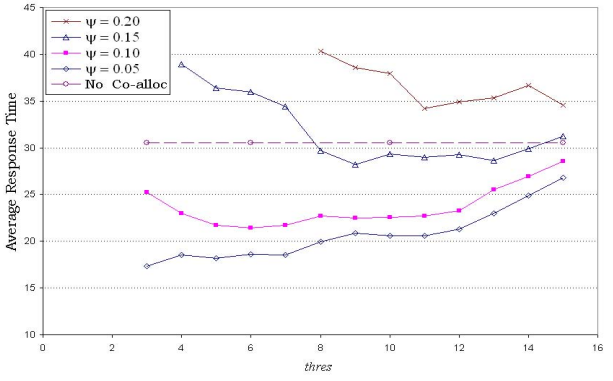

Fig. 3. Performance variations for different ψ values

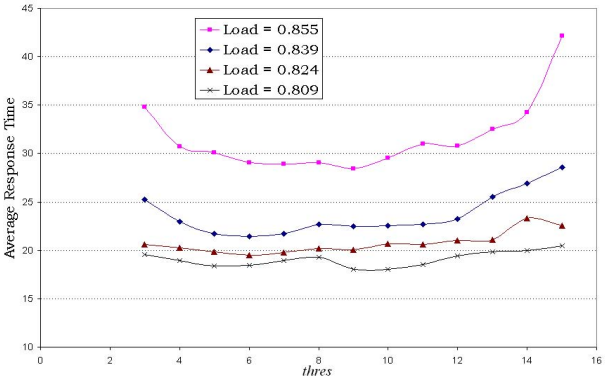

Fig. 4. Performance variation with *thres* for different values of load

7.4 Overall Effect of System/Job Parameters

We now discuss the overall effect of the parameters investigated in 7.1 - 7.3.

First, we observe that at low values of ψ, co-allocation is beneficial. Breaking up of jobs make them easier to pack but the execution time penalty leads to more occupation of processors. If processor hours due to the penalty exceed those saved from reduced fragmentation, then there is no net benefit in co-allocation. Secondly, we observe that there is an optimal *thres* value for any ψ value. It increases with ψ and independent of the load. This is due to the fact that at high ψ, if a lot of jobs are broken into components, a lot of processor hours are lost due to the increased execution time which exceeds those saved from fragmentation. This leads to unnecessary occupation of the processors which leads to jobs over delaying in the queue. This translates into poor performance. This can be solved by breaking fewer jobs (increasing *thres*). If however *thres* is

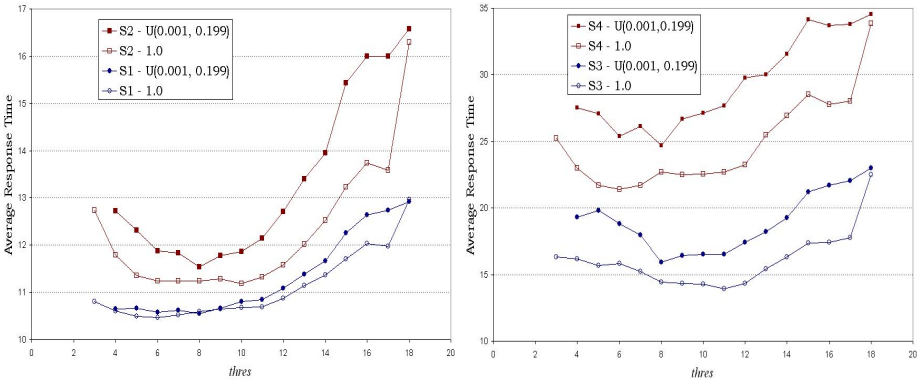

Fig. 5. Group-wise performance for $\psi = 0.1$ and $\psi \sim U[0.001, 0.199]$

too high, the packing problem of unbroken jobs becomes significant. The system suffers fragmentation and hence poor performance.

8 Communication Intensity Distribution

So far, we have considered cases where ψ is fixed. This implies that both α and λ are fixed. As far as we know, there are no documented studies on the extent and distribution of communication intensities in supercomputer workloads. However, we believe that due to the diversity of the sources and applications processed by supercomputers, α (hence ψ) is not fixed. We therefore assume λ to be fixed but α to vary among jobs. We consider a case where $\psi \sim U[0.001, 0.199]$. We compare its performance with a case when $\psi = 0.1$. We also study the relative performance of jobs grouped by ψ.

8.1 The Effect of ψ Distribution

We now compare the performance of S_1, S_2, S_3 and S_4 for the job stream when with $\psi = 0.1$ and $\psi \sim U[0.001, 0.199]$ (Figure 5). We observe that more dispersion in ψ leads to poor performance. Deterioration in performance is felt by both co-allocated and non co-allocated jobs.

8.2 Classification by Communication Intensity

In 8.1, we observed that communication heterogeneity negatively affect performance. We now study group-wise performance of co-allocated jobs grouped by ψ. We create four groups C_1, C_2, C_3 and C_4 consisting jobs whose ψ lies in the ranges $(0.00, 0.05), (0.05, 0.10), (0.10, 0.15)$ and $(0.15, 0.20)$ respectively. We summarize their performance variation with *thres* in Figure 6.

We observe small deviations in performance of the different groups. This implies that the effect of communication is felt in the entire job steam rather than the individual co-allocated jobs.

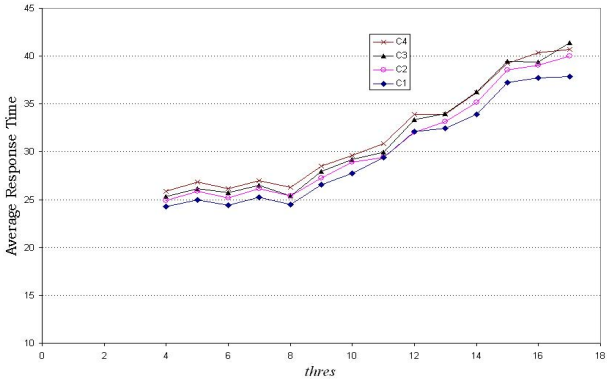

Fig. 6. Performance of communication based groups

9 Conclusion and Future Work

We have used communication intensity and intra to inter cluster speed ratio to represent execution time penalty. The approach is synonymous to the fixed penalty approach but more elaborate. We have also studied the limits of co-allocation viability as a function of ψ, load and *thres*. We have observed that there are parameter limits within which the co-allocation is viable. They depend on both α and λ. We have also observed poorer performance for heterogeneously communicating jobs. This implies that co-allocation is not as viable as depicted in earlier studies that considered fixed penalties.

Our work opens up more avenues for future research. This includes studying asynchronously communicating jobs and communication cognizant scheduling.

Reference

1. Aida, K., Kasahara, K., Narita, S.: Job scheduling scheme for pure space sharing among rigid jobs. In: Feitelson, D.G., Rudolph, L. (eds.) IPPS-WS 1998, SPDP-WS 1998, and JSSPP 1998. LNCS, vol. 1459, pp. 98–121. Springer, Heidelberg (1998)
2. Bucur, A.I.D.: Performance analysis of processor co-allocation in multi-cluster systems. PhD Thesis, Technical University Delft, Delft, The Netherlands (2004)
3. Bucur, A.I.D., Epema, D.H.J.: The influence of communication on the performance of co-allocation. In: Feitelson, D.G., Rudolph, L. (eds.) JSSPP 2001. LNCS, vol. 2221, pp. 66–86. Springer, Heidelberg (2001)
4. Czajkowski, K., Foster, I.T., Kasselman, C.: Resource co-allocation in computational grids. In: Proc. HPDC 1999, pp. 37–47 (1999)
5. Feitelson, D.G.: Metric and workload effects on computer systems evaluation. Computers 36(9), 18–25 (2003)
6. Feitelson, D.G., Rudolph, L.: Towards convergence of job schedulers for parallel supercomputers. In: Feitelson, D.G., Rudolph, L. (eds.) IPPS-WS 1996 and JSSPP 1996. LNCS, vol. 1162, pp. 1–26. Springer, Heidelberg (1996)

7. Feitelson, D.G., Rudolph, L., Schwiegelshohn, U.: Parallel job scheduling: A status report. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2004. LNCS, vol. 3277, pp. 1–16. Springer, Heidelberg (2005)
8. Frachtenberg, E., Feitelson, D.G.: Pitfalls in parallel job scheduling evaluation. In: Feitelson, D.G., Frachtenberg, E., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2005. LNCS, vol. 3834, pp. 257–282. Springer, Heidelberg (2005)
9. Jones, W.M.: Improving parallel job scheduling performance in multi-clusters through selective job co-allocation. PhD dissertation, Clemson University, Clemson, South Carolina, USA (2005)
10. Jones, W.M., Ligon III, W.B., Pang, L.W.: Characterization of bandwidth-aware meta-schedulers for co-allocating jobs across multiple clusters. Journal of Supercomputing 34(2), 135–163 (2005)
11. Jones, W.M., Pang, L.W., Stanzione, D., Ligon III, W.B.: Bandwidth-aware co-allocating meta-schedulers for mini-grid architectures. In: Proc. CLUSTER 2004, pp. 45–54 (2004)
12. Ngubiri, J., van Vliet, M.: Group-wise performance evaluation of processor co-allocation in multi-cluster systems. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) JSSPP 2007. LNCS, vol. 4942, pp. 1–13. Springer, Heidelberg (2008)
13. Sonmez, O., Mohamed, H., Epema, D.H.J.: Communication-aware job placement policies for the KOALA grid scheduler. In: Proc. 2^{nd} IEEE Int. Conf. on e-Science and Grid Comp., pp. 79–87 (2006)
14. Srinivasan, S., Kettimuthu, R., Subramani, V., Sadayappan, P.: Characterization of backfilling strategies for parallel job scheduling. In: Proc. ICPPW 2002, pp. 514–520 (2002)
15. Srinivasan, S., Kettimuthu, R., Subramani, V., Sadayappan, P.: Selective reservation strategies for backfill job scheduling. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2002. LNCS, vol. 2537, pp. 55–71. Springer, Heidelberg (2002)
16. Srinivasan, S., Krishnamoorthy, S., Sadayappan, P.: A robust scheduling technology for moldable scheduling of parallel jobs. In: Proc. CLUSTER 2003, pp. 92–99 (2003)
17. Stromaier, E., Dongarra, J.J., Meuer, H.W., Simon, H.D.: Recent trends in the marketplace of high performance computing. Parallel Computing 31(3,4), 261–273 (2005)
18. The Grid Workloads Archive, `http://gwa.ewi.tudelft.nl/`
19. The Parallel Workloads Archive, `http://www.cs.huji.ac.il/labs/parallel/workload/`
20. Tsafrir, D., Feitelson, D.G.: Instability in parallel job scheduling simulation: The role of workload flurries. In: Proc. IPDPS 2006 (2006)
21. Top500 supercomputing sites (Accessed December 20^{th}, 2007), `http://www.top500.org`