Memory Usage in the LANL CM-5 Workload

Dror G. Feitelson

Institute of Computer Science
The Hebrew University, 91904 Jerusalem, Israel
feit@cs.huji.ac.il or http://www.cs.huji.ac.il/~feit

Abstract. It is generally agreed that memory requirements should be taken into account in the scheduling of parallel jobs. However, so far the work on combined processor and memory scheduling has not been based on detailed information and measurements. To rectify this problem, we present an analysis of memory usage by a production workload on a large parallel machine, the 1024-node CM-5 installed at Los Alamos National Lab. Our main observations are

- The distribution of memory requests has strong discrete components, i.e. some sizes are much more popular than others.
- Many jobs use a relatively small fraction of the memory available on each node, so there is some room for time slicing among several memory-resident jobs.
- Larger jobs (using more nodes) tend to use more memory, but it is difficult to characterize the scaling of per-processor memory usage.

1 Introduction

Resource management includes a number of distinct topics, such as scheduling and memory management. However, in the context of parallel processing, scheduling is the single most important issue [9,6]. Memory management is hardly ever exercised, because of its performance implications and effect on synchronization [3,21]. Instead, jobs must be completely memory resident in order to execute.

Nevertheless, memory requirements may place severe constraints on scheduling, and therefore cannot be ignored. For example, in distributed memory machines processor allocation includes allocating part of the system's memory as well — the memory that is packaged with these processors. This memory must be large enough to fulfill the job's requirements. This consideration limits dynamic partitioning schemes and may prevent them from reducing the partition sizes when the load increases, thus undermining the whole idea of dynamic partitioning [16,17].

While there has been some research on the effect of memory requirements on job scheduling, this research has been hampered by the lack of concrete information about actual memory requirements that are experienced in practice. The unique contribution of this paper is to provide such information. We start with a brief overview of the system we analyzed, the LANL CM-5, in the next

section. Section 3 contains the memory usage analysis, including such issues as the distribution of memory usage, the correlation of memory usage with degree of parallelism, the correlation of memory usage with runtime, and the relation between the memory requested and that actually used. Section 4 contains a discussion of the results and their implications, and Section 5 presents the conclusions.

2 The Analyzed System

The analysis presented in this paper is based on a detailed accounting log from the 1056-node Connection Machine CM-5 installed at Los Alamos National Lab. While such machines are no longer manufactured, this one is still in active use, and considered quite powerful — it ranked 21st in the world in the November '96 Top500 list, and came in first among Connection Machines [4].

The CM-5 is a distributed memory machine based on SPARC processors. 1024 of the 1056 nodes are used for parallel computation, with a total of 32 GB of memory (i.e. 32 MB per node). The machine is statically partitioned into partitions with power-of-two numbers of processors from 32 up to 512. Within each partition, jobs may be gang-scheduled, or they may request dedicated use of the partition [20]. While the fact that only 5 sizes are available is restrictive, other work on parallel workload characterization has shown conclusively that users prefer powers of two even if there are no architectural constraints [8,5].

The part of the log we worked on covers most of 1996 (from January 1 to September 23), and contains useful data on 36308 jobs (we ignore jobs that used 0 time etc.). The data includes a lot of information about the submittal process, but we mostly used the data on the number of processors used, the runtime, the requested memory, the memory actually used, and whether or not the nodes were dedicated. Runtimes are expressed in seconds (s), and memory usage in kilobytes (KB). The data was collected by DJM [14], the Distributed Job Manager used on CM-5 machines. Most jobs were indeed run using DJM, but 1492 of them were "foreign", i.e. launched directly by users. The log contains less information about foreign jobs, e.g. they do not have predefined resource requests.

Fig. 1 shows the histograms of job sizes and resource use during this period (there were also three 1024-node jobs, not shown). When counting jobs (left plot), 32-node jobs are the most common, followed by 128-node jobs. While there are less jobs that use 256 or 512 nodes, their numbers are still significant. If we weigh the number of jobs by the time they ran (middle plot), the variance is smaller: 32-node jobs occupied about twice as much time as each of the other sizes, which are all similar. If we also weigh the jobs by the number of nodes they use, and plot the total node-seconds for each size (right plot), then we find that the large jobs use more resources than smaller ones. The dashed lines across the columns denote the boundary between dedicated and non-dedicated use of the nodes: below are dedicated, and above are shared or foreign. Nearly all 256 and 512-node jobs ran in dedicated mode.

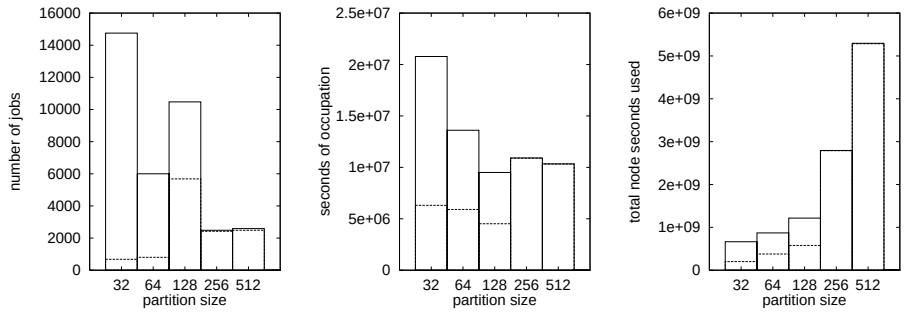

Fig. 1. *Histograms of job sizes in the analyzed log. In the left graph, all jobs have equal weights. In the middle, jobs are weighted by their runtime. At right, jobs are weighted by the product of runtime and parallelism, which is equivalent to counting node-seconds.*

3 Memory Usage Analysis

3.1 Memory Usage Distribution

Our goal is to characterize memory usage on parallel supercomputers. But memory by itself is not the resource in question: it is *the occupation of memory for a period of time*. A job that uses 1MB of memory for one second obviously requires less resources than one that uses the same 1MB for an hour. Therefore, the unit of resource use is not the KB, but the KB·s, or KiloByte Second.

On the other hand, a job using 1MB for one second also requires different resources from one that uses 10KB for 100 seconds, even though the total KB·s in both cases are equal. We therefore characterize memory usage by a weighted distribution, where the x axis denotes the *amount* of memory used (in KB), and the y axis reflects the *cumulative time* that this amount of memory was used. Using such a plot to characterize total memory usage by jobs is equivalent to creating a histogram where jobs are weighted by their runtime, rather than being given equal weights.

While characterizing the total memory usage by jobs is important, it is not enough. For parallel jobs, the memory used *per processor* is also important. Again, there are several ways to combine the requests of different jobs and create a single representation. The most meaningful seems to be to weigh the per-processor usage by *the product of runtime and number of processors*. Thus a job using 1MB across 10 processors for 10 seconds imposes a load of 100KB on each processor for 10 seconds, which is the same as 100 single-processor jobs using 100KB and running for 1 second each.

It should be stressed that choosing the right weights is extremely important, as typically a small fraction of the jobs account for a large fraction of the resource usage. The differences are shown graphically in Fig. 2 for the case of per-processor memory usage. If all jobs are given equal weights, it seems that most jobs only require less than 5MB of memory per processor (top plot). But if the more

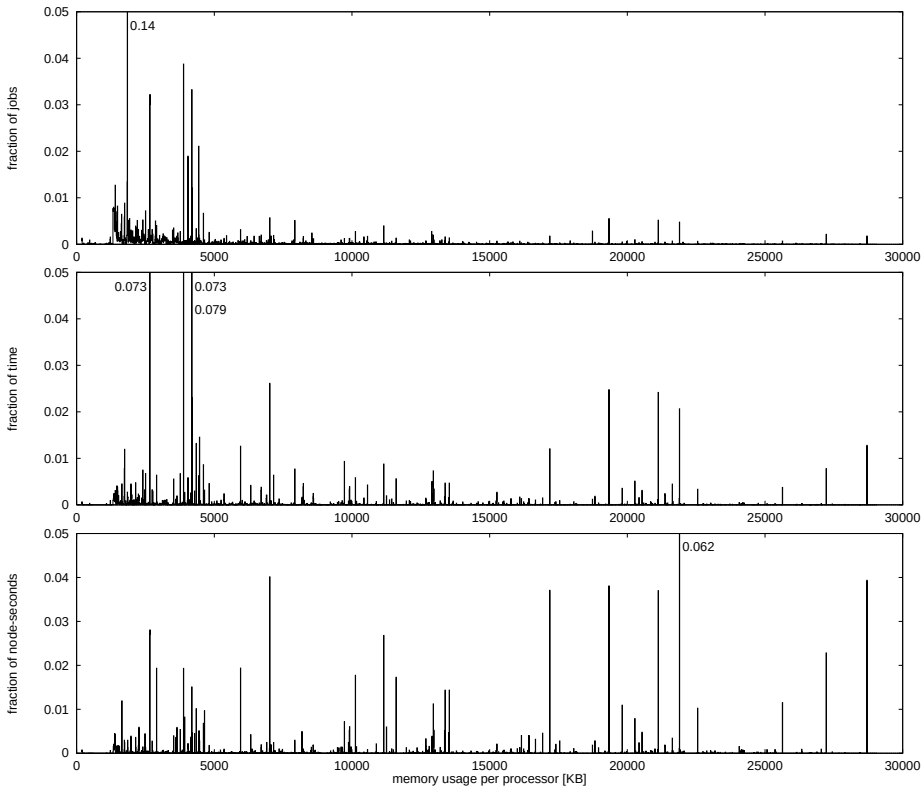

Fig. 2. *The distribution of per-processor memory usage, using a linear scale and buckets of 10 KB. In the top plot, all jobs have equal weight. In the middle, jobs are weighted according to their run time. The bottom plot shows the distribution for individual processors, which is equivalent to weighing the jobs according to the product of runtime and degree of parallelism.*

correct node-second weighting is used (bottom plot), it is evident that actually at any given moment a significant fraction of the processors are using a significant fraction of their memory (in the range of 20–30MB). In particular, the highest peak in the first plot, representing 14% of all the jobs, all but disappears in the other plots, because all these jobs were extremely short lived.

Characterization of discrete components A prominent feature of all these distributions is their discrete nature: they are composed of a number of very high discrete components, and very low "background noise". The question is what leads to this structure.

Table 1 contains information about all the discrete components that represent more than 1% of the total jobs or more than 1% of the total node-seconds. For each one, users that individually contributed more than 1% are identified. In

KB per			jobs		node sec	
proc	user	of	%	of	%	of
1400		14		1.27		0.42
1640	usr1	6	0.13	0.18	1.18	1.19
1830	usr2	6	1.15	1.33	0.00	0.02
1840	usr2	20	13.25	14.06	0.01	0.07
2650	usr3	10	2.00	3.01	1.96	2.71
2660	usr4	13	1.29	3.22	1.41	2.81
	usr3		1.12		0.88	
2900	usr5	7	0.26	0.40	1.37	1.93
3880	usr6	12	3.64	3.88	1.37	1.93
4040		20		1.89		0.22
4180	usr7	8	1.42	3.33	0.42	1.51
	usr3		1.17		0.46	
4190		4		1.21		0.46
4340		7		0.34		1.01
4430	usr8	17	1.21	2.11	0.01	0.28
5950	usr5	4	0.25	0.31	1.55	1.94
7010	usr3	2	0.57	0.57	4.02	4.02
10120	usr9	4	0.10	0.27	0.70	1.77
	usr5		0.17		1.07	
11150	usr9	4	0.16	0.40	1.08	2.68
	usr5		0.22		1.60	
11600	usr10	2	0.13	0.13	1.73	1.73
12950	usr2	2	0.19	0.20	1.11	1.12
13380	usr3	2	0.15	0.16	1.44	1.44
13530	usr11	2	0.13	0.14	1.43	1.44
17180	usr12	2	0.17	0.18	3.69	3.71
19330	usr5	3	0.41	0.55	2.81	3.81
	usr9		0.14		0.94	
19810	usr12	1	0.05	0.05	1.09	1.09
21120	usr5	3	0.40	0.52	2.85	3.70
21890	usr9	4	0.15	0.48	2.04	6.21
	usr5		0.31		4.08	
22550	usr10	2	0.06	0.06	1.02	1.02
25630	usr10	2	0.06	0.07	1.14	1.15
27220	usr13	2	0.17	0.21	1.76	2.28
28700	usr10	2	0.09	0.18	2.21	3.94
	usr1		0.09		1.73	

Table 1. *Single-user contributions to discrete components that are above 1% of the total. User names are replaced by numbers. Column 3 gives the total number of users contributing to this component. Columns 5 and 7 give the total fraction of jobs and node-seconds in this component, respectively, while columns 4 and 6 give the fraction contributed by the user specified in column 2.*

a few cases (1400, 4040, 4190, and 4340 KB per processor) the component is seen to be a combination of multiple users, who each contributed only less than 1%. But in the other 26 discrete components, most of the resource usage can be attributed to a single user (or sometimes two users). In particular, the huge peak at 1840 KB per processor can be attributed to a single user who ran 13.25% of all the jobs in the log, and in fact did so in just over one week. It is thus risky to assign too much meaning to the discrete components themselves, but it is safe to assume that such a discrete structure is common, because some users are much more active than others.

Rendering with logarithmic scale Much information can be gleaned from the detailed distributions of memory usage such as those shown in Fig. 2. However, when investigating the distribution of an essentially continuous variable, one encounters the problem of choosing the granularity of observation. If the grain is too coarse, interesting details may be smoothed out. If it is too fine, the data will drown in a sea of noisy details.

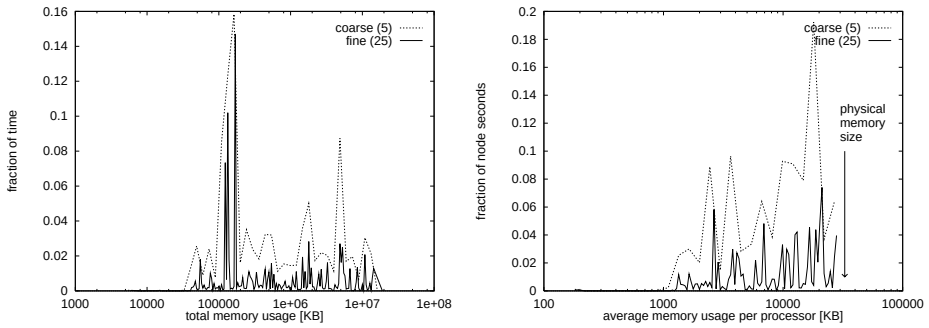

Fig. 3. *Distributions of memory usage using coarse and fine bucket sizes.*

When comparing jobs with different attributes, we shall use logarithmically-sized buckets and count how many jobs (with appropriate weights) fall into each such bucket. In order to reduce the granularity and observe finer details, we multiply the memory usage value by a scaling factor after taking the log. Thus the mapping from memory usage m to bucket b is

$$b = \lfloor f \cdot \log(m) \rfloor$$

The larger the scaling factor f, the more buckets that are used, with each one representing a smaller part of the spectrum. In most of what follows, we use a scaling factor of 5, which we feel is a good compromise. In Fig. 3 we compare the obtained distribution with one that would be obtained by using a scaling factor of 25 (in these figures the values for the different buckets are connected by a line; this is visually more convenient than drawing a bar chart with a bar

for each bucket). This shows that the peaks in the coarse view of the distribution correspond to the larger narrow discrete components in the fine view of the distribution, or to regions where there are multiple peaks that are very close together. Taking this to the extreme, we note that the peaks in the fine distribution typically correspond to discrete peaks in the linear distribution of Fig. 2, where a linear scale and buckets of only 10 KB were used. The whole distribution is a combination of "background noise" with these strong discrete components.

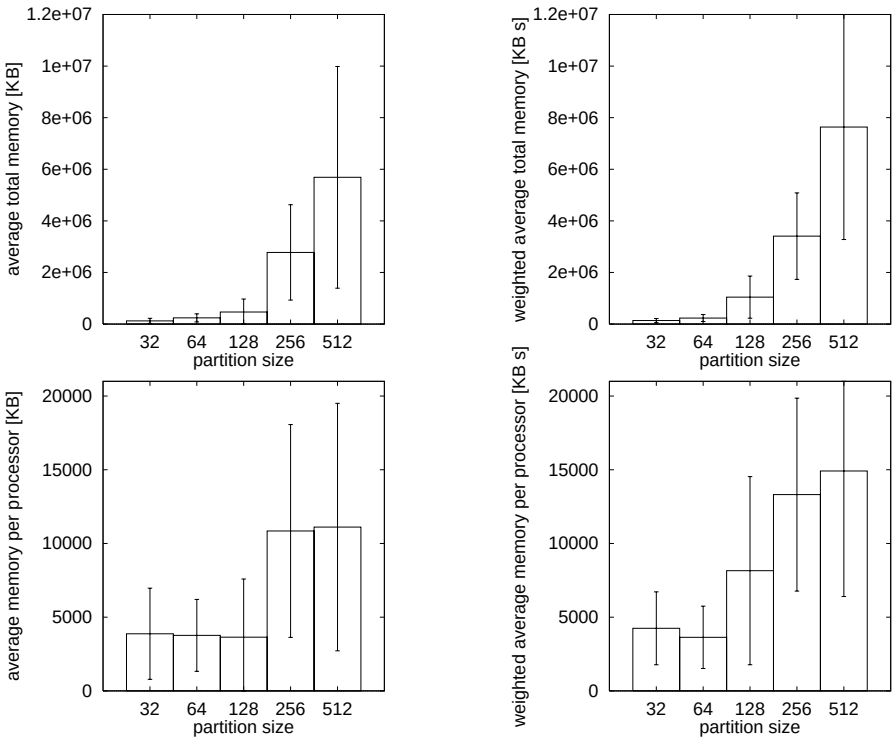

Fig. 4. *Average memory usage by jobs of different sizes. The top graphs are for total memory usage, and the bottom ones for per-processor usage. In each pair, the left graph is a simple average, while the right one is weighted by time. The error bars denote the standard deviation.*

3.2 Correlation of Memory Usage and Parallelism

The average memory usage by jobs of different sizes is shown in Fig. 4. Obviously, larger jobs require more memory, but the distribution is very wide. Interestingly, when the memory usage per processor is plotted, one sees that larger jobs also use more memory per-processor. The effect of larger memory use by larger jobs

is even more pronounced when they are weighted by time. The standard deviation in most cases is somewhat smaller than the average, indicating that the coefficient of variation is less than 1 for each job size.

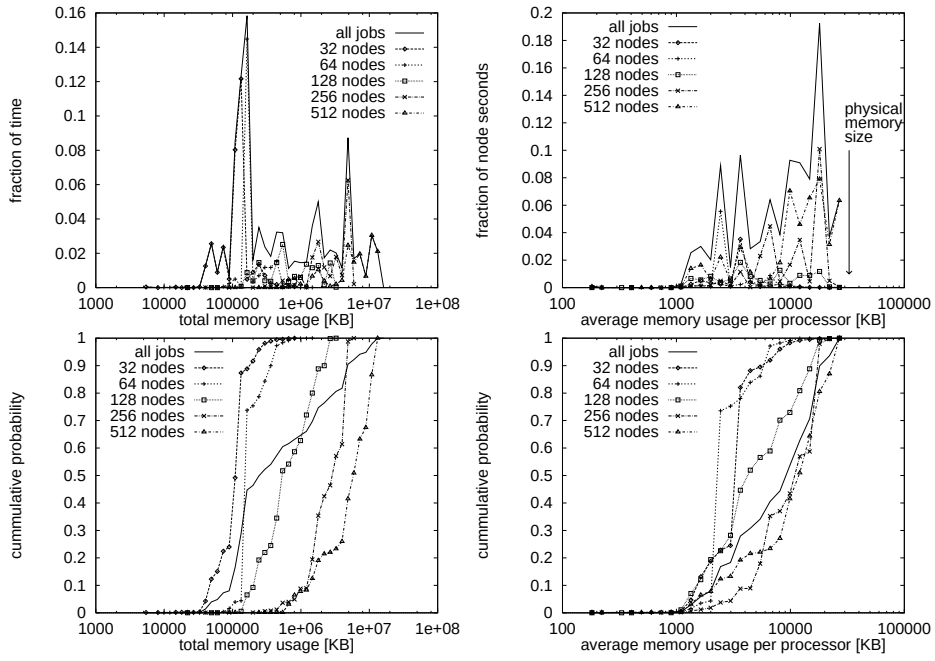

Fig. 5. *Distributions of memory usage by all jobs and jobs of different sizes. Left: total memory usage, weighted by job runtime. Right: average memory usage per processor, weighted by job runtime and number of processors. Top: pointwise distribution. Bottom: cumulative distribution (each normalized independently).*

The full memory usage distributions for jobs using different numbers of processors are shown in Fig. 5, using the logarithmic scale with coarse granularity. Both pointwise and cumulative distributions are shown for clarity, for both total and per-processor memory usage. Recall that peaks in these graphs actually correspond to very narrow discrete components in the distributions, and that weighing by time is used.

The distribution of memory usage for all jobs is rather wide, but the distributions for the different sizes are clearly distinguishable. This is especially clear in the cumulative plot of total memory usage, where the plots are neatly arranged in partition-size order, indicating that when more nodes are used, the weight of the distribution moves to higher memory usage values. The per-processor usage plots show again that this is not only a result of using memory on more nodes.

Interestingly, some of the discrete peaks in the distribution are dominated by a single partition size. This corresponds to the effects that the activity of single

users sometimes have on the whole distribution, as described above.

3.3 Memory Usage in Dedicated and Shared Partitions

The same graphs are plotted again in Fig. 6, except that here the jobs are classified by their use of dedicated nodes rather than by size. About a third of the jobs ran in dedicated mode (12074 out of 36308), while the rest were gang-scheduled. However, it should be noted that nearly all jobs that ran on the large partition sizes did so in dedicated mode, so these jobs account for a very large fraction of the total node-seconds used (about 85.2%).

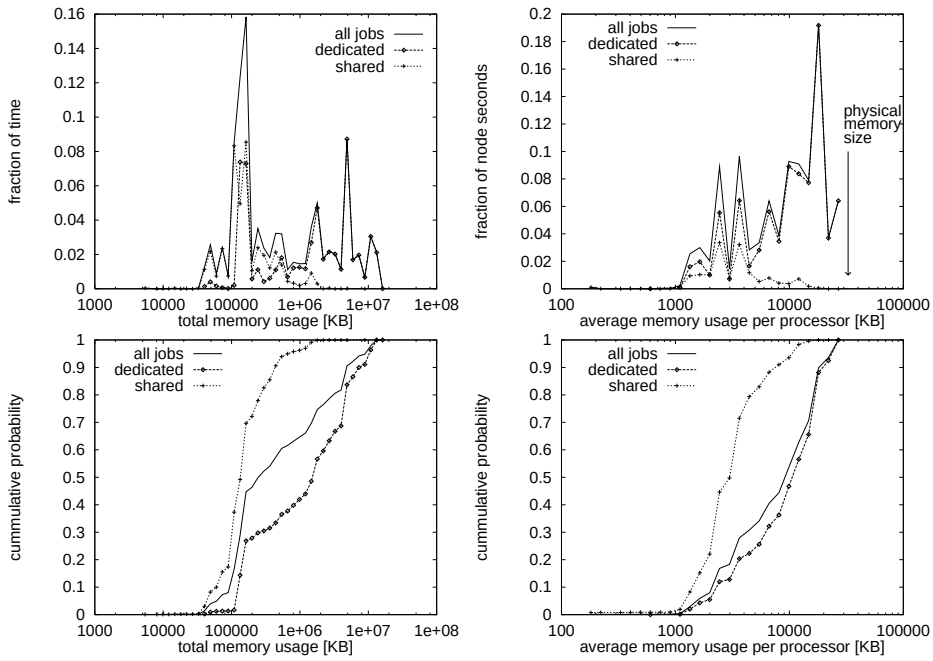

Fig. 6. *Distributions of memory usage by jobs using dedicated or shared nodes. Left: total memory usage, weighted by job runtime. Right: average memory usage per processor, weighted by job runtime and number of processors. Top: pointwise distribution. Bottom: cumulative distribution (each normalized independently).*

On average, dedicated jobs used more memory than jobs that shared their nodes with other jobs, as shown in Table 2. The distributions agree with this observation and show that actually the whole weight of the distributions is higher for dedicated jobs. Indeed, high memory usage values are completely dominated by dedicated jobs. This correlates with the fact that nearly all large jobs (on 256 and 512 processors) were dedicated.

job class	total memory		memory per proc	
	average	sd	average	sd
all	2029762	3299901	12187	8200
dedicated	3295679	3843629	13537	8049
shared	263188	334604	4444	3296

Table 2. *Average memory usage of different job classes. Numbers for total memory are weighted by runtime, and those for memory per processor by node seconds.*

3.4 Memory Usage vs. Requests

Another interesting issue is the accuracy with which users request memory. To get an idea of this accuracy, we plot the distribution of actual memory usage as a percentage of requested memory in Fig. 7, using buckets of 4 percentage points. While there is a peak of over 17% in the range of 4–8% of the request, the second highest peak of nearly 10% is at 96–100%, indicating that at least in some cases users make very accurate predictions (or possibly use up all what they get). However, in general the distribution is rather flat, indicating that using user input as an estimate of memory requirements leads to poor predictions. Moreover, it should also be noted that a significant number of jobs (5992 to be exact, or 16.5%) used more memory than they requested (only partially shown in the graph), with a maximum factor of 32 time more!

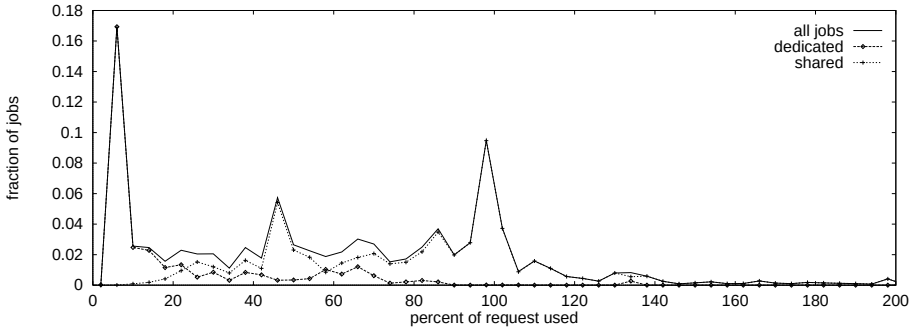

Fig. 7. *Distribution of actual memory usage as a fraction of requested memory.*

Some insight into the nature of user input is obtained by classifying the jobs into those that ran on dedicated nodes vs. those that shared their nodes with other jobs. It turns out that the peak at 4–8% can be attributed completely to dedicated jobs, whereas the peak at 96–100% is due to shared jobs. Furthermore, nearly all jobs that used more memory than requested ran on shared nodes (5887 out of 5992). This means that jobs that ran in dedicated mode typically did so for reasons other than their memory requirements. It also means that when users actually need to provide low memory estimates in order to run (as is the case

on shared nodes) they sometimes make very accurate estimates, and sometimes they lie...

3.5 Correlation of Memory Usage and Runtime

Finally, we investigate the possible correlation between memory usage and runtime. The scatter plot on the left of Fig. 8 shows all pairs of runtime and total memory usage. The most striking features of this plot are the well-defined band of memory usage values, the horizontal stripes that indicate preferred memory usage values, and the sharp limits on runtime at the right-end side (probably due to NQS queue limits). But in addition, it is possible to discern a weak correlation: the weight at the left end is lower than at the right end.

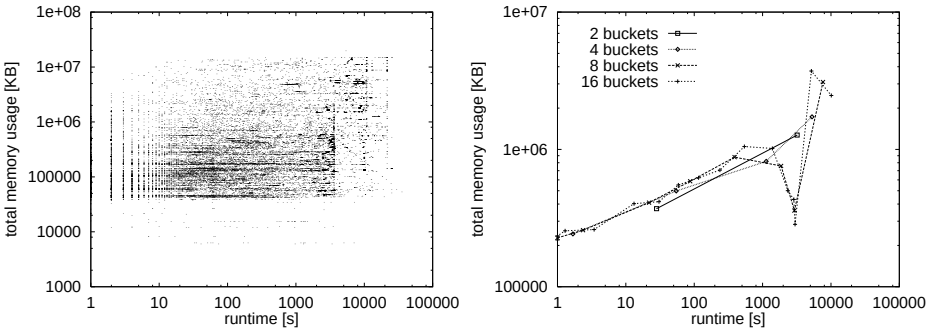

Fig. 8. *Scatter plot and functional relationship of memory usage and runtime.*

The graph on the right shows this correlation more clearly. Here the jobs are partitioned into a number of equal sized buckets according to their runtime (e.g. when using 4 buckets, the first bucket includes a quarter of the jobs, and specifically those with the shortest runtimes; the second bucket includes the next quadrant, and so on). One data point is drawn for each bucket, at the average runtime and average memory usage of the jobs in the bucket. When using 2 or 4 such buckets, the resulting graph is smooth and monotonically increasing. With 8 or 16 buckets, it is seen that some buckets with a high average runtime actually have a low average memory usage. Interestingly, the runtimes of these buckets correspond to the most prominent runtime limit from the scatter plot.

4 Discussion and Implications

The motivation for studying memory usage in parallel workloads is to provide data for the design and evaluation of scheduling algorithms that take memory requirements into account. This takes two forms. One is direct effects on scheduling algorithms and policies, for example the assertion that time slicing may be

used because most jobs use a relatively small fraction of the available memory. The other is incorporation of memory requirements into workload models used to drive simulations or as inputs to analytical evaluations. For example, a model of how memory requirements change with the degree of parallelism facilitates the evaluation of scheduling policies for different machine configurations.

4.1 Time Slicing and Memory Pressure

Previous work about incorporating memory considerations into scheduling algorithms has been quite limited, and included ideas such as the following:

- In systems that use space slicing, place a lower bound on partition size so that enough memory will be available [16,17,13].
- When the partition size is adjustable, do not reduce it too much, because small partitions cause jobs to run longer and thus increase the memory pressure [15].
- In systems that use swapping, make the residence time proportional to the memory footprint size in order to amortize the cost of loading the memory image [1,7].

A recurring theme has been the worry that most applications will use all the available memory, thereby sharply limiting real-world solutions to scheduling problems. In particular, concern has been expressed about the fact that gang scheduling requires multiple jobs to be memory resident at the same time, and thus increases memory pressure. Our results alleviate this concern, as the weight of the distribution of per-processor memory usage is far below the actual memory available on each node, indicating that from a memory point of view there is room for sharing the nodes among a number of jobs (as indeed is done on the CM-5).

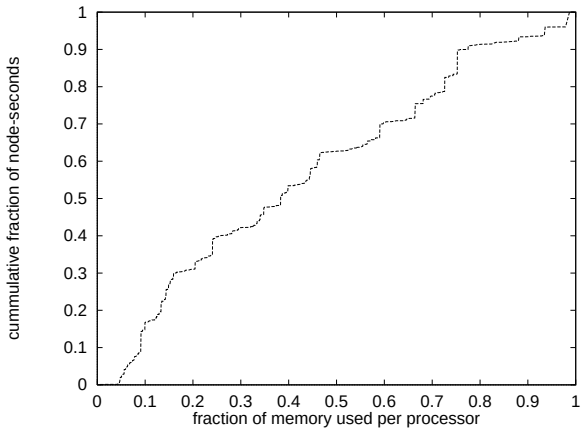

Fig. 9. *Cumulative distribution of per-processor memory usage, weighted by runtime and number of processors.*

To quantify this claim, we plot the cumulative distribution of the per-processor memory usage, weighted by runtime and number of processors (Fig. 9). This is actually the cumulative version of the distribution shown in the bottom plot of Fig. 2. The x axis shows the fraction of memory used on average on each processor, using the conservative estimate that 29.08 MB are available (rather than 32 MB; this was the highest value observed in the trace). The y axis shows cumulative node seconds. The way to read this graph is as follows: for each point (x, y) on the graph, y is the probability that up to x of the memory is being used. But more importantly, it is also the probability that at least $1 - x$ is free.

Except for the extreme edges, the graph is above the diagonal, which indicates relatively low resource usage. For example, if we focus on the mid point of the x axis, where up to half the memory is used, we find that this happens 62.7% of the time on average. This means that there is a probability of 0.627 that a running job will leave at least half of the memory free for other jobs. Alternatively, if we focus on the midpoint of the y axis, we find that it corresponds to 38.3% memory usage. Thus half the time we will find that at least 61.7% of the memory is available.

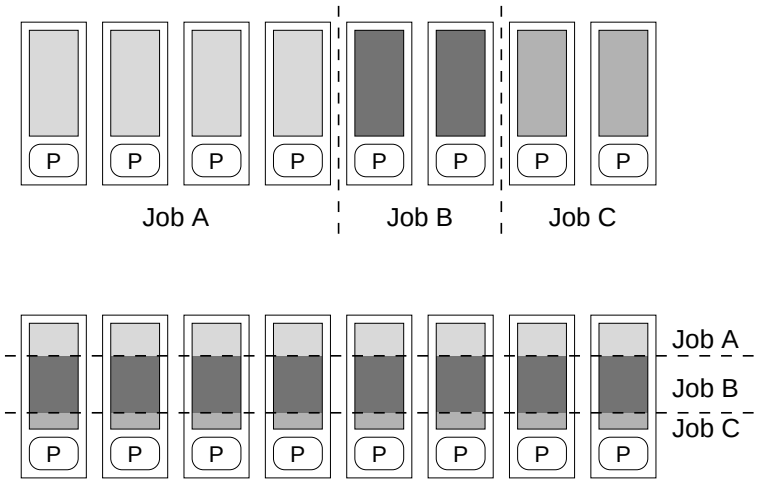

Fig. 10. *With space slicing, processor allocation dictates memory allocation due to the "vertical" partitioning (top). With gang scheduling, memory is partitioned "horizontally", so memory allocation is decoupled from processor allocation (bottom).*

The fact that nodes can be time-sliced without undue increase in memory pressure has far reaching implications. An important observation is that gang scheduling allows "horizontal" partitioning of memory, rather than the more rigid and inflexible "vertical" partitioning that happens when space slicing is used (Fig. 10). This added flexibility is expected to be instrumental in serving

more jobs and reducing fragmentation. As a result, it allows more jobs to fit into the available memory, and delays the need to employ swapping.

The relatively unaggressive memory usage observed also has implications for space slicing policies, and particularly for adaptive and dynamic partitioning. One of the strengths of these policies is that the partition sizes are reduced under heavy load, leading to more efficient use of the resources (because most jobs display diminishing returns when more processors are added, and can use smaller numbers of processors more effectively than large numbers). Again, concern has been expressed that it would not be possible to reduce the partition sizes and exploit this feature, because of memory requirements [16]. Our results indicate that rather small partition sizes may suffice in many cases.

4.2 Modeling Memory Usage

A separate issue is the modeling of memory usage for use in simulations and analysis. Specifically, we would like to be able to model how resource requirements change when applications scale to larger systems. Three models have been proposed in the literature:

- *Fixed work*. This assumes that the work done by a job is fixed, and parallelism is used to solve the same problems faster. Therefore the runtime and per-processor memory usage are assumed to be inversely proportional to the degree of parallelism. This model is the basis for Amdahl's law [2].
- *Fixed time* [11,12,22]. Here it is assumed that parallelism is used to solve increasingly larger problems, under the constraint that the total runtime stays fixed. In this case, the runtime distribution is independent of the degree of parallelism, but the total memory usage is expected to increase with increased parallelism.
- *Memory bound* [19]. This model assumes that the problem size is increased to fill the available memory on the larger machine, so that the per-processor memory usage is maintained. As the amount of productive work typically grows at least linearly with the dataset size, and the overheads associated with parallelism grow with the degree of parallelism, the total execution time will actually increase with added parallelism.

We can get some speculative evidence concerning this question by comparing the resource requirements of jobs that actually ran on different size partitions.

Our preliminary results concerning memory usage, combined with our previous results regarding the correlation between runtime and parallelism [5], indicate that the truth probably lies between the fixed-time model and the memory bound model. In a nutshell, all three resources tend to scale up together: larger jobs use more processors, use more memory, and run longer. However, it seems that all these models are over-simplified to the point where it is hard to correlate them with measured results. In particular, users configure their applications according to their needs rather than according to the way resources happen to be packaged in the machine [18]. Thus users rarely use all the memory available,

on any size partition. It is true, however, that they tend to use more on larger partitions.

Finally, we note that modeling the memory usage distribution itself is not easy, because it does not seem to be similar to commonly used "analytical" distributions. Instead, it has a number of large discrete components (Fig. 2). It is premature to draw too many conclusions about this distribution based on evidence from only one machine.

5 Conclusions

Scheduling is concerned with the allocation of scarce resources to competing jobs. Two of the most important resources are computing cycles and memory locations. The allocation of computing cycles allows for some tradeoff between the degree of parallelism and time — moldable and malleable jobs may use less processors for more time to accumulate the same overall number of cycles [10]. With memory, such a tradeoff is only possible if paging is used. As paging is typically considered to be too expensive due to its overhead and adverse effect on communication and synchronization, parallel jobs typically have to be memory resident throughout their execution. Memory requirements therefore impose a very rigid constraint on the scheduler and may severely limit its options.

In order to investigate the effect of memory requirements on scheduling, information about typical memory requirements is needed. We have studied the memory usage patterns of a production scientific workload on the LANL CM-5 parallel supercomputer for this purpose. Our main observations are

- The distribution of memory requests is rather wide, with strong discrete components (i.e. some sizes are much more popular than others). It is not similar to commonly used and mathematically tractable distributions.
- Many jobs use a relatively small fraction of the memory available on each node, e.g. less than half. Thus there is typically room for more than one job to be memory resident at the same time. However, it is advisable to pack the jobs according to their memory requirements, that is, to judiciously choose jobs with small requirements to fill in the space left by a job with large requirements. If this is done, time slicing among several memory-resident jobs is distinctly possible.
- Larger jobs (using more nodes) tend to use more memory than small jobs (using less nodes) in total, and also more memory per processor. However, it is difficult to characterize this scaling precisely, and further investigation (based on data from additional machines) is required.

Acknowledgement

Many thanks to Curt Canada of Los Alamos National Lab for providing the raw data used in this study.

References

1. G. Alverson, S. Kahan, R. Korry, C. McCann, and B. Smith, "*Scheduling on the Tera MTA*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 19–44, Springer-Verlag, 1995. Lecture Notes in Computer Science Vol. 949.
2. G. M. Amdahl, "*Validity of the single processor approach to achieving large scale computer capabilities*". In *AFIPS Spring Joint Comput. Conf.*, vol. 30, pp. 483–485, Apr 1967.
3. D. C. Burger, R. S. Hyder, B. P. Miller, and D. A. Wood, "*Paging tradeoffs in distributed-shared-memory multiprocessors*". *J. Supercomput.* **10(1)**, pp. 87–104, 1996.
4. J. J. Dongarra, H. W. Meuer, and E. Strohmaier, "*Top500 supercomputer sites*". http://www.netlib.org/benchmark/top500.html. (updated every 6 months).
5. D. G. Feitelson, "*Packing schemes for gang scheduling*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 89–110, Springer-Verlag, 1996. Lecture Notes in Computer Science Vol. 1162.
6. D. G. Feitelson, *A Survey of Scheduling in Multiprogrammed Parallel Systems*. Research Report RC 19790 (87657), IBM T. J. Watson Research Center, Oct 1994.
7. D. G. Feitelson and M. A. Jette, "*Improved utilization and responsiveness with gang scheduling*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), Springer Verlag, 1997. Lecture Notes in Computer Science (this volume).
8. D. G. Feitelson and B. Nitzberg, "*Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 337–360, Springer-Verlag, 1995. Lecture Notes in Computer Science Vol. 949.
9. D. G. Feitelson and L. Rudolph, "*Parallel job scheduling: issues and approaches*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 1–18, Springer-Verlag, 1995. Lecture Notes in Computer Science Vol. 949.
10. D. G. Feitelson and L. Rudolph, "*Toward convergence in job schedulers for parallel supercomputers*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 1–26, Springer-Verlag, 1996. Lecture Notes in Computer Science Vol. 1162.
11. J. L. Gustafson, "*Reevaluating Amdahl's law*". *Comm. ACM* **31(5)**, pp. 532–533, May 1988. See also *Comm. ACM* **32(2)**, pp. 262–264, Feb 1989, and *Comm. ACM* **32(8)**, pp. 1014–1016, Aug 1989.
12. J. L. Gustafson, G. R. Montry, and R. E. Benner, "*Development of parallel methods for a 1024-processor hypercube*". *SIAM J. Sci. Statist. Comput.* **9(4)**, pp. 609–638, Jul 1988.
13. C. McCann and J. Zahorjan, "*Scheduling memory constrained jobs on distributed memory parallel computers*". In *SIGMETRICS Conf. Measurement & Modeling of Comput. Syst.*, pp. 208–219, May 1995.
14. Minnesota Supercomputer Center, Inc., *The Distributed Job Manager Administration Guide*. 1993. ftp://ec.msc.edu/pub/LIGHTNING/djm_1.0.0_src.tar.Z.
15. E. W. Parsons and K. C. Sevcik, "*Coordinated allocation of memory and processors in multiprocessors*". In *SIGMETRICS Conf. Measurement & Modeling of Comput. Syst.*, pp. 57–67, May 1996.

16. V. G. J. Peris, M. S. Squillante, and V. K. Naik, "*Analysis of the impact of memory in distributed parallel processing systems*". In *SIGMETRICS Conf. Measurement & Modeling of Comput. Syst.*, pp. 5–18, May 1994.
17. S. K. Setia, "*The interaction between memory allocation and adaptive partitioning in message-passing multicomputers*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 146–165, Springer-Verlag, 1995. Lecture Notes in Computer Science Vol. 949.
18. J. P. Singh, J. L. Hennessy, and A. Gupta, "*Scaling parallel programs for multiprocessors: methodology and examples*". *Computer* **26(7)**, pp. 42–50, Jul 1993.
19. X-H. Sun and L. M. Ni, "*Scalable problems and memory-bounded speedup*". *J. Parallel & Distributed Comput.* **19(1)**, pp. 27–37, Sep 1993.
20. Thinking Machines Corp., *Connection Machine CM-5 Technical Summary.* Nov 1992.
21. K. Y. Wang and D. C. Marinescu, "*Correlation of the paging activity of individual node programs in the SPMD execution model*". In 28th *Hawaii Intl. Conf. System Sciences*, vol. I, pp. 61–71, Jan 1995.
22. P. H. Worley, "*The effect of time constraints on scaled speedup*". *SIAM J. Sci. Statist. Comput.* **11(5)**, pp. 838–858, Sep 1990.