Incorporating Job Migration and Network RAM to Share Cluster Memory Resources [*]

Li Xiao Xiaodong Zhang Stefan A. Kubricht
Department of Computer Science
College of William and Mary
Williamsburg, VA 23187-8795
{lxiao, zhang, kubricht}@cs.wm.edu

Abstract

Job migrations and network RAM are two major approaches for effectively using global memory resources in a workstation cluster, aimed at reducing page faults in each local workstation and improving the overall performance of cluster computing. Using either remote executions or preemptive migrations, a load sharing system is able to migrate a job from a workstation without sufficient memory space to a lightly loaded workstation with large idle memory space for the migrated job. In a network RAM system, if a job cannot find sufficient memory space for its working sets, it will utilize idle memory space from other workstations in the cluster through remote paging. Conducting trace-driven simulations, we have compared the performance and trade-offs of the two approaches and their impacts on job execution time and cluster scalability. Our study indicates that job-migration-based load sharing schemes are able to balance executions of jobs in a cluster well, while network RAM is able to satisfy data-intensive jobs which may not be migratable by sharing all the idle memory resources in a cluster. We also show that a network RAM cluster of workstations is scalable only if the network is sufficiently fast. Finally, we propose an improved load sharing scheme by combining job migrations with network RAM for cluster computing. This scheme uses remote execution to initially allocate a job to the most lightly loaded workstation and, if necessary, network RAM to provide a larger memory space for the job than would be available otherwise. The improved scheme has the merits of both job migrations and network RAM. Our experiments show its effectiveness and scalability for cluster computing.

[*]This work is supported in part by the National Science Foundation under grants CCR-9400719, CCR-9812187, and EIA-9977030, by the Air Force Office of Scientific Research under grant AFOSR-95-1-0215, and by Sun Microsystems under grant EDUE-NAFO-980405.

1. Introduction

1.1. Background and objectives of the study

With the rapid development of CPU chips and the increasing demand of data accesses in applications, the memory resources in a workstation cluster become more and more expensive relative to CPU cycles. Effective usage of global memory resources is an important consideration in the design of load sharing policies for cluster computing. When a workstation does not have sufficient memory space for its assigned jobs, the system will experience a large number of page faults, resulting in long delays for each job. There are two major approaches to more effectively use global memory resources in a workstation cluster, aiming at minimizing page faults in each local workstation and improving overall performance of cluster computing: (1) job-migration-based load sharing schemes and (2) network RAM. A job-migration-based load sharing system attempts to migrate jobs from a workstation without sufficient memory space to a lightly loaded workstation with large idle memory space for the migrated jobs. When a job migration is necessary, the migration can be either a remote execution (where a job is initiated on a remote workstation), or a preemptive migration which suspends the selected job and moves it to a remote workstation where it is restarted. In a network RAM system [5], if a job cannot find sufficient memory space for its working sets, it will utilize idle memory space from other workstations in the cluster through remote paging. Since remote paging is slower than accessing local memory (about 100 to 200 times slower depending on network traffics) but much faster than local disk access (more than 500 times faster), the idle global memory space or the network RAM can be considered as another layer between the local memory and the local disk in the memory hierarchy of a workstation.

Besides sharing the same objective of reducing page faults in each local workstation, the two approaches have another common technical feature in their implementations. Both systems maintain a global load index record for each workstation about how its CPU and/or memory resources are being utilized. This record is either stored in a master workstation or distributed among the workstations, and is updated periodically by the cluster workstations.

There are several major differences between the two approaches in the ways that the global memory resources are shared. Because of these differences, each approach has its own merits and limits. First, in a network RAM cluster system, a workstation is provided with a huge global memory space for its jobs. The global memory space could be even larger than its local disk space. Thus, it is possible to eliminate accesses to local disks due to page faults in a network RAM cluster. In contrast, memory allocations of a job could be limited by the local memory size of a workstation in a migration-based load sharing cluster system where local memory modules are not shared by other workstations. Thus, a network RAM cluster system could be more beneficial to large or non-migratable data-intensive jobs than a migration-based load sharing cluster system. Second, the effectiveness of global paging operations in a network RAM cluster system is heavily dependent on the cluster network speed. In contrast, the network, in general, is less frequently used in a remote-execution-based load sharing cluster system. In other words, a remote-execution-based load sharing system relies less on the network speed than a network RAM system. Finally, a migration-based load sharing system is able to balance the workloads among workstations by sharing both CPU and memory resources, while a network RAM system only considers global memory resources for load sharing. Without job migrations, job executions may not be evenly distributed in a cluster — some workstations can be more heavily loaded than others. Although the lightly loaded workstations in a network RAM cluster system can be used as memory servers for heavily loaded workstations, their CPU resources are not fully utilized by the cluster.

Conducting trace-driven simulations, we have compared the performance and trade-offs of the two approaches and their impact on job execution time and cluster scalability. In this study, we quantitatively address the following three questions: (1) Under what cluster and workload conditions is a migration-based load sharing policy or the network RAM beneficial for performance improvement? (2) What are the performance effects of limited network bandwidths and cluster size to the two system approaches? (3) How do we optimize designs of cluster resource management systems by effectively combining the two system approaches?

1.2. Related work

Regarding network RAM implementations, the Global Memory System (GMS) [4] and the Remote Memory Pager [8] attempt to reduce the page fault overhead by remote paging techniques. DoDo [1] is designed to improve system throughput by harvesting idle memory space in a distributed system. The owner processes have the highest priority for their CPUs and memory allocations in their workstations, which divides the global memory system into different local regions.

Regarding job-migration-based load sharing systems, most load sharing schemes proposed for distributed systems (e.g., [3], [6], [7], [12]) mainly consider effectively sharing CPU cycles. A memory ushering algorithm is used in MOSIX for memory load sharing [2]. Recently, we have developed several load sharing alternatives by considering both CPU and memory resources [11]. The objective of our design is to reduce the number of page faults caused by unbalanced memory allocations of distributed jobs so that overall performance can be significantly improved.

The rest of the paper is organized as follows. We describe the job-migration-based load sharing policies and the network RAM implementations we have used in this study in Section 2. We present the performance evaluation methodology and simulation environments in Section 3. The performance comparisons and analyses are presented in Section 4. We propose an improved load sharing policy supported by network RAM in Section 5. Some implementation issues are discussed in Section 6. Finally, we summarize the work in Section 7.

2. Job-migration-based load sharing vs. network RAM

Network RAM and job-migration-based load sharing related operations on workstation j, for $j = 1, ..., P$, are characterized by the following variables: (1) RAM_j: the total memory space provided for user-level programs in MBytes of the workstation, (2) RP_j: the amount of remote paging in MBytes from the workstation, (3) FM_j: the idle memory space in MBytes of the workstation, and (4) MT_j: the memory threshold (the memory space for the stable working set) in MBytes is the total amount of memory thresholds accumulated from the running jobs on the workstation. If $RAM_j > MT_j$, page faults would rarely occur, otherwise, paging would be frequently conducted during executions of jobs in the workstation.

2.1. Network RAM organizations

A network RAM organization allows each workstation to use not only its own local memory, but also to access idle memory space of other workstations through remote paging in a cluster. The memory allocation decision for a job on workstation j is made by

$$memory\,allocation = \begin{cases} local\ memory & \text{if } MT_j < RAM_j \\ global\ memory & \text{if } MT_j \geq RAM_j, \end{cases}$$

where the global memory allocation is implemented by finding the most lightly loaded workstation one by one for remote paging based on the following search algorithm:

```
Allocate the idle local memory space to the arrival job;
$MD_j = MT_j$;
While ($MD_j \geq RAM_j$) and
    (idle memory space is available elsewhere)
  do
    find workstation $i$ with the largest idle memory space
      among $P-1$ workstations (excluding workstation $j$);
    allocate $RP_i = min\{MD_j - RAM_j, FM_i\}$ MBytes
      from workstation $i$ to the job in workstation $j$;
    $FM_i = FM_i - RP_i$;
    $MD_j = MD_j - RP_i$;
```

where MD_j represents the current local memory demand on workstation j. The while loop continues until the memory demand is met or no idle memory available in the system. If $MD_j \geq RAM_j$ after the global allocations, disk accesses due to page faults will occur in workstation j. In order to minimize the global paging, we give local memory accesses the highest priority. The global paging is only conducted when the remote workstation has additional idle memory space. Therefore, when a new local job arrives, the network RAM paging services for remote jobs will be transferred to other workstations if any memory space occupied by remote pages is needed for this new job.

2.2. CPU-Memory-based load sharing

The job-migration-based load policy we have selected for this comparative study is the CPU-Memory-based load sharing scheme [11], which makes a job migration decision by considering both CPU and memory resources. The basic principle of this scheme is as follows. When a workstation has sufficient memory space for both running and requesting jobs ($MT_j < RAM_j$), the load sharing decision is made by a CPU-based policy where the load index in each workstation is represented by the length of the CPU waiting queue. As long as the CPU waiting queue is not larger than the threshold which is set based on the CPU capability, the requesting jobs will be locally executed in the workstation. Otherwise, the load sharing system finds the remote workstation with the shortest waiting queue to either remotely execute this job or to preemptively migrate an eligible job from the local workstation to the remote workstation. When the workstation does not have sufficient memory space for the jobs ($MT_j \geq RAM_j$), the load sharing scheme attempts to migrate jobs to suitable workstations or even to hold the jobs in a waiting pool if necessary. Again, the migration can be either remote execution or preemptive migration.

During an execution of a memory-intensive job, page faults may occur periodically. Each such period is called a *transition*, where page faults are conducted to bring a working set into memory. The data references will then be memory hits for a while until the working set changes and page faults are conducted, forming the next transition period. The local reference period is called a *phase*. If the phases of a job are disjoint or almost disjoint, the best time to do a preemptive migration is at the end of a phase and before another transition period is started (bring in the next working set into memory). The migrated job would carry no data or a small data set to a remote workstation. However, in practice it may be difficult to predict the data access phase and transition patterns of so many different jobs. If this prediction is impossible, remote executions should be a practically optimal solution for load sharing of memory-intensive jobs [11]. For this reason, remote executions are used in our CPU-Memory-based load sharing policy.

3. Performance Evaluation Methodology

Our performance evaluation is simulation-based, consisting of two major components: a simulated distributed system and workloads. We discuss performance evaluation metrics, the simulation model, and the workloads in this section.

3.1. Performance metrics

We target evaluating and comparing the performance merits and limits for a given workload scheduled by a job-migration-based load sharing policy, supported by network RAM, or without load sharing or network RAM, under various system and workload conditions. The following performance metrics are used in our evaluation:

- *average execution time per job* is defined as $\bar{t} = \frac{\sum_{i=1}^{n} t_i}{n}$, where t_i is the measured execution time of an individual job, and n is the number of jobs in a given workload.
- *execution time breakdowns*: The average execution time is further broken into CPU service time, queuing time, disk access time due to page faults, and networking time for job migrations or remote pagings including network contention time.

3.2. A simulated workstation cluster

We have developed a simulator for a bus-based workstation cluster which has the following functional abilities: (1) to support different job-migration-based load sharing policies, including the CPU-Memory-based policy, (2) to simulate the remote paging system described in Section 2.1 providing network RAM for the cluster, (3) to simulate bus contention, and (4) to simulate system heterogeneity. The simulated cluster is scalable and is configured with 6 to 18 workstations containing 300 MHz CPUs each with local memory of 128 MBytes. The cluster network is an Ethernet bus of 10 Mbps and 100 Mbps. Each disk access time due to a page fault is 10 ms. The size of a memory page is 4 KBytes. The CPU local scheduling uses the round-robin policy.

When a page fault happens during job execution, the job is suspended from the CPU during the paging service. The CPU service is switched to a different job. When page faults happen in the executions of several jobs, they will be served in FIFO order. The overhead of a remote execution is set to 0.1 second for 10Mbps Ethernet and 0.05 for 100Mbps Ethernet based on the numbers of real systems provided by [6].

The bus service and contention are simulated based on typical bus transactions as follows. Each workstation is given a sequence number which also represents its priority rank to access the bus. The priority increases as the sequence number decreases. As multiple requests for bus services arrive in sequence, the requests will be served in FIFO order. If the requests arrive simultaneously, they will be served in an order based on their workstations' bus access priorities.

3.3. Workloads

The workloads we have used are the 8 traces from [6]. The jobs in each trace were distributed among 6 homogeneous workstations. Each job has the following three major parameters: arrival time, arrival workstation, and duration time. We generate the memory demand of each job from a Pareto distribution with the mean size of 1 MBytes [1]. We assume that the memory threshold of a job is 40% of its requested memory space. Each job has the following 4 attributes: arrival time, arrival workstation, requested memory size, and duration time. The number of jobs is doubled and tripled in each trace as the number of workstations is scaled from 6 to 12, and scaled from 12 to 18, respectively.

[1] We have run the simulations for memory demands from Pareto distributions with the mean sizes of 1, 2, 4, and 8 MBytes, and have obtained consistent performance results. Results with a larger mean size are more supportive of our solutions. We only present the results on the mean memory demand of 1 MBytes in this paper.

For the job-migration-based load sharing system, the page faults in each workstation are conducted in our simulation as follows. When the memory threshold of jobs in a workstation is equal to or larger than the available memory space ($MT_j \geq RAM_j$), each job in the workstation will cause page faults at a page fault rate which is proportional to the memory usage of this workstation. In practice, application jobs have page fault rates from 1 to 10 per million instructions. We set the rate in the same range in our experiments.

For the network RAM system, when $MT_j \geq RAM_j$ in a workstation, The remote paging is conducted as described in Section 2.1. The remote paging rate of a job is proportional to the size of the global memory space allocated to this job. If the aggregate global memory space in the cluster is not sufficient for the job, the job in the workstation will cause page faults to access the local disk at a page fault rate.

4. Simulation Results and Analysis

Our performance evaluation targets understanding the effects of network bandwidth changes to both the job-migration-based load sharing scheme and the network RAM supported by remote paging. We have also quantitatively evaluated two performance trade-offs for comparing the two schemes: (1) the trade-off between reducing local disk accesses due to page faults and increasing network contention and delay due to remote paging; (2) the trade-off between reducing local disk accesses by network RAM and balancing job execution among workstations by job migrations.

4.1. Impact of limited network bandwidths

Both job migrations and remote paging rely on the cluster network for data transfers. However, the performance of each scheme is affected differently by changes of the network speed. Contention and scalability are two major factors impacting the performance of the two schemes.

Figure 1 presents the average execution times per job (the left figure) and the network contention portions in the execution times (right figure) of "trace 0" running on clusters of 6, 12 and 18 workstations, where the jobs are executed without load sharing (denoted as "Base"), scheduled by CPU-Memory-based load sharing policy with remote executions (denoted as "LS_RE"), and executed on a network RAM system (denoted as "Net_RAM"). The bus speed varies from 10 Mbps to 100 Mbps. The mean memory demand of the jobs is 1 MBytes.

The page fault rate was set to 5.96 per million instructions for all the experiments on "trace 0". Since the number of jobs proportionally increases as the number of workstations increases in the cluster, the average execution times per job of "trace 0" by "Base" are identical on clusters of 6, 12, and 18 workstations. Using the same page fault rate, we conducted the experiments to compare the execution time performance between “LS_RE” and “Net_RAM” for the same workload of “trace 0”.

We have the following observations based on the experimental results in Figure 1. First, the performance of “LS_RE” is not significantly affected as the cluster is scaled from 6 to 12, and from 12 to 18 workstations. The performance improves only slightly as the bus speed increases from 10 Mbps to 100 Mbps. This is because the data communication via the network by remote executions is a small portion in the total execution time (0% to 0.005%, see the right figure in Figure 1). Second, the performance of “Net_RAM”, supported by remote paging, is highly sensitive to the network speed and the number of workstations in the cluster. For example, the average execution time of “Net_RAM” is 46% lower than that of “LS_RE” on the cluster of 6 workstations where the bus speed is 10 Mbps. As the bus speed increases to 100 Mbps, the average execution time of “Net_RAM” is further reduced 45%. However, as the cluster of 10 Mbps increases to 12 workstations, the average execution time of “Net_RAM” sharply increases (to about 3.6 times higher than that of “LS_RE”). As the cluster speed increases to 100 Mbps, the execution time is significantly reduced, and is 69% lower than that of “LS_RE”. Similar performance data are collected as the number of workstations increases to 18. Our experiments show that the cluster scalability and workload performance when using network RAM are highly dependent on the speed of the cluster because the network latency due to data transfer and contention is a significant portion in the total execution time (0.05% to 46.03%, see the right figure in Figure 1). Finally, in the workload of “trace 0”, some jobs are marked as non-migratable. Therefore, the power and benefits of job migrations may be limited.

In order to fully take advantage of job migrations, we released the restrictions on the non-migratable jobs so that remote executions can be applied to all the jobs in “trace 0”. Figure 2 presents the average execution time per job (left figure) and the networking portions in the execution times (right figure) of the modified “trace 0” scheduled by “LS_RE” in comparisons with “Base” and “NET_RAM” on the clusters of 6, 12 and 18 workstations. We show that the performance of “LS_RE” is significantly improved. The execution times of “LS_RE” using a 10 Mbps cluster are slightly lower than the execution times of “Net_RAM” using 100 Mbps clusters of 6, 12, and 18 workstations. In this case, the remote-execution-based load sharing policy not only outperforms the network RAM, but is also more cost-effective.

From the scalability point of view, “LS_RE” demands less network bandwidth in order to scale the cluster by connecting more workstations than “Net_RAM” does. For example, “LS_RE” is scalable from 6 to 18 workstations for both 10 and 100 Mbps buses, while “Net_RAM” is only scalable for the 100 Mbps bus.

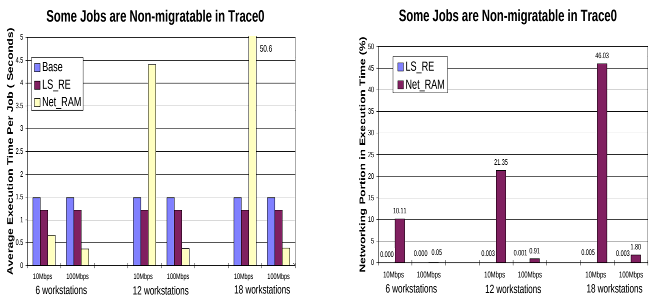

Figure 1. The average execution times per job (the left figure) and the networking portions in the execution times (right figure) of “trace 0” with job migration restrictions running on clusters of 6, 12 and 18 workstations.

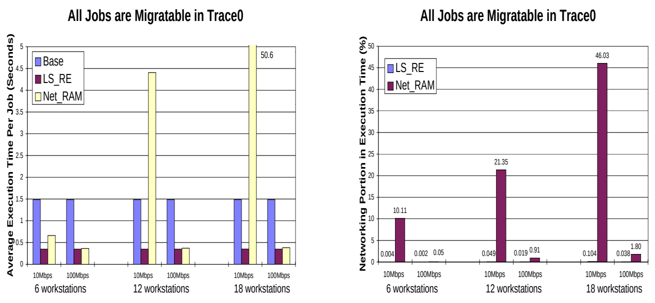

Figure 2. The average execution times per job (the left figure) and the networking portions in the execution times (right figure) of “trace 0” without job migration restrictions running on clusters of 6, 12 and 18 workstations.

4.2. Trade-offs between page fault reductions and load sharing

Page faults in the network RAM are reduced at the cost of additional network contention and delay. Although page fault reductions may be limited by the remote-execution-based load sharing scheme for large data-intensive jobs, the scheme requires less additional network support compared with the network RAM. In order to provide insights into the trade-offs between the two schemes, we present the execution time breakdowns of “trace 0” where all jobs are migratable in Figures 3 and 4. The execution time of a workload consists of “CPU”, “networking”, “page faults”, and “queuing” portions. “CPU” is the execution time by the CPU for the workload. “Networking” is the time spent on the cluster network, which is used for remote pagings by the network RAM, or for remote executions by the load sharing scheme (including network contention time). “Page faults” is the local disk delay time for both schemes. “Queuing” is the average waiting time for a job to be executed on a workstation. When the workload is executed on a 10 Mbps cluster of 6 and 12 workstations, the networking time for remote pagings by the network RAM is one of the major portions in the execution time. For example, the networking times contribute 15.5% and 23.08% to the execution times on the 6 workstation cluster and the 12 workstation cluster (see the left figures in Figures 3 and 4), respectively. In contrast, the networking time for remote executions by the load sharing scheme is insignificant (0.06% and 0.11%). Consequently, the queuing time for each job in the network RAM is significantly increased by networking delay, causing much longer execution times than for the remote-execution-based load sharing scheme.

We have also shown that the networking time portions in the executions of the workload by the network RAM are significantly reduced by increasing the bus speed from 10 Mbps to 100 Mbps. Consequently, the queuing time for each job is also significantly reduced (see the right figures in Figures 3 and 4).

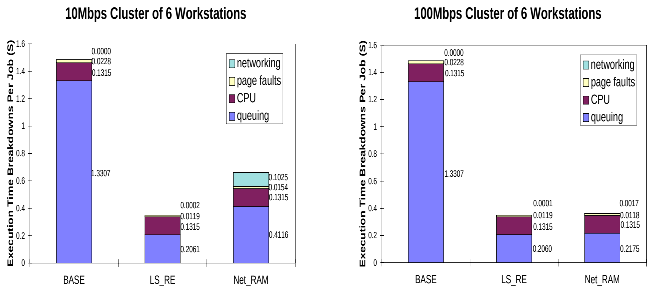

Figure 3. The average execution times per job of “trace 0” without job migration restrictions running on a 10 Mbps cluster (the left figure) and a 100 Mbps cluster (the right figure) of 6 workstations.

Another trade-off of the two schemes is between page fault reductions and load sharing. Without job migrations, job executions may not be evenly distributed among the workstations by the network RAM although page faults can

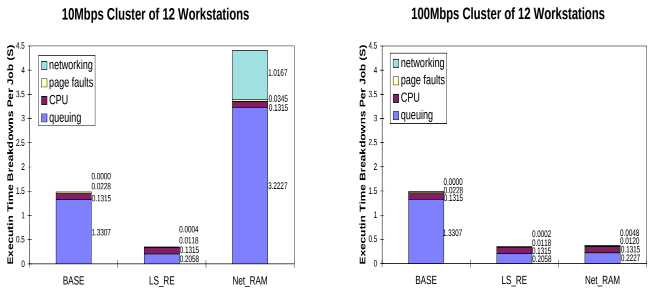

Figure 4. The average execution times per job of “trace 0” without job migration restrictions running on a 10 Mbps cluster (the left figure) and a 100 Mbps cluster (the right figure) of 12 workstations.

be significantly reduced through remote pagings. The unbalanced loads among workstations when using network RAM is another reason for the long queuing times for the workload executed on the 10 Mbps clusters of 6 and 12 workstations.

5. An improved load sharing scheme

Our experiments show advantages and limits of the network RAM and the remote-execution-based load sharing scheme. A natural optimization step for overcoming the limits of each scheme is to combine them. Here is the basic idea of this improved load sharing scheme. When a workstation has sufficient space for both current and requesting jobs, the job execution location will be determined by the CPU-based policy. When a workstation runs out of memory space for both current and requesting jobs, we first attempt to migrate the new arrival job to the most lightly loaded workstation. If the workstation does not have sufficient memory space for the job, the network RAM will be used to satisfy the memory allocation of the job through remote paging. The memory allocation combing both remote executions and network RAM of the scheme is outlined as follows:

```
If ($MT_j \geq RAM_j$)
  find workstation $i$ with the largest idle memory space
     among $P$ workstations;
  If $i \neq j$
    remotely execute the job at workstation $i$;
  If ($MT_i \geq RAM_i$) and ($Q_{net} < NT$)
    allocate global memory by using network RAM;
else
  schedule the job by the CPU-based load sharing policy;
```

All the notations in the above operations except Q_{net} and NT have been defined in the beginning of Section 2. Variable Q_{net} is the number of jobs waiting to access the network, and NT is the network threshold which functions to allow only a limited number of network accesses at a time. The purpose of setting NT is to prevent a large number of bus requests during a small time interval. Such bursty bus requests will cause network contention to sharply increase.

The improved load sharing scheme is denoted as "LS_Net_RAM". Each workload trace is further divided into two types: (1) some jobs are restricted for migrations in a trace and (2) all the jobs in a trace are migratable. Figures 5 and 6 present the average execution times of all the 8 traces of both type 1 (left figure) and type 2 (right figure) executed on the 10 Mbps and 100 Mbps clusters of 6 workstations, respectively.

Our experiments show that "LS_Net_RAM" performs well for all the 8 traces of both types, while "LS_RE" or "Net_RAM" only performs well on one type of traces. We obtained consistent results on clusters of 12 and 18 workstations.

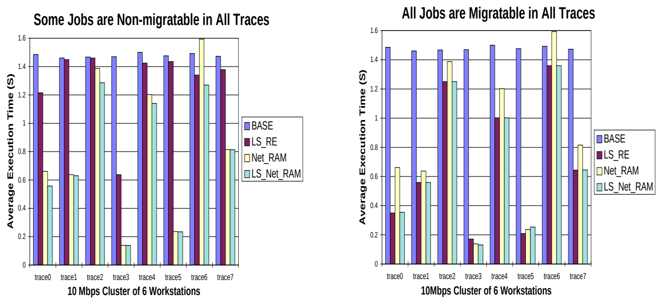

Figure 5. The average execution times per job of all the 8 traces (the left figure for the 8 traces where some jobs are non-migratable, and the right figure for the 8 traces where all the jobs are migratable), running on a 10 Mbps cluster of 6 workstations.

6. Implementation Issues

We are currently in the initial stages of developing this load sharing scheme, combining network RAM with migration techniques, on a cluster of Linux workstations. Our high-level simulation study indicates that this is a worthwhile endeavor, however, the simulation is unable to consider all of the possible performance penalties without additional data. The remote paging is a source of overhead on any workstation providing network RAM. CPU cycles are used in receiving and unpacking the message request (and perhaps the page to be replaced), locating the requested page, and then transmitting the page using the communication protocol. This overhead is unavoidable and difficult to model in detail without experimental results. In building an actual implementation, there are additional factors and questions related to the unknowns that must be addressed:

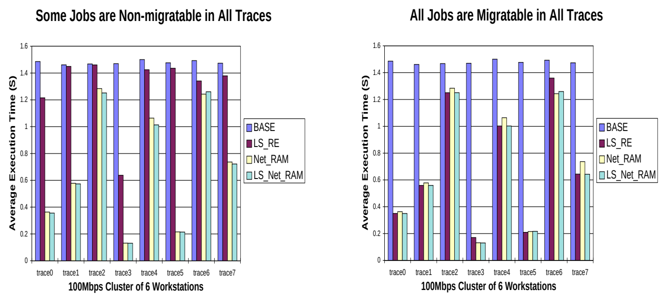

Figure 6. The average execution times per job of all the 8 traces (the left figure for the 8 traces where some jobs are non-migratable, and the right figure for the 8 traces where all the jobs are migratable), running on a 100 Mbps cluster of 6 workstations.

(1) Are there significant penalties for processes executing on a workstation functioning as a server for remote paging when a remote page on the workstation is requested?

(2) Should processes on such a workstation be interrupted to serve remote paging requests?

(3) Should a workstation with active CPU-bound jobs even be allowed to function as a network RAM provider, even if it has idle memory?

(4) What are the implications of the load sharing goal? Is the scheme better suited for high-throughput computing rather than high-performance computing?

Results from previous studies [1][4] imply that there can be a performance impact resulting from the number of CPU cycles required to serve remote pages. It is unclear how significant this impact could be on a cluster using the migration policy. For high-performance computing, the goal is to minimize the execution time of an individual job. When a job must compete for cycles with a remote paging mechanism, the job will not execute as fast as possible. This is contrary to the goals of a high-performance computing policy. To meet the requirements of this type of policy, it may be necessary to use only machines with both an idle CPU as well as idle memory as network RAM providers, or perhaps to inhibit process interruption due to requests to service remote pages. For a load sharing goal emphasizing high performance computing, a prerequisite may be mandatory to limit the availability of memory on a node for network RAM if there are jobs already running on the workstation.

A high-throughput computing policy attempts to efficiently utilize all of the resources in a cluster to generate faster overall execution. It is acceptable for jobs running on a workstation serving remote pages to execute more slowly, if other jobs benefit by achieving faster execution times through the use of network RAM, as long as throughput is maximized.

Further work must be done to fully understand how the overhead inherent in providing network RAM affects running jobs on a workstation administering the service. With this knowledge, we will be able to adapt the load sharing policy to meet the efficiency requirements of the various load sharing goals.

7. Conclusion

We have experimentally examined and compared job migrations and network RAM for sharing global cluster memory resources. Based on our experiments and analysis we have the following observations and conclusions:

- Providing a large memory space through remote paging, network RAM is particularly beneficial for large or data-intensive workloads where some jobs may not be migratable. However, the network RAM performance is heavily dependent on the cluster speed and the availability of the idle memory space in the cluster. Since load balancing is not considered, uneven job distributions may degrade the overall performance of cluster computing using network RAM.

- Dynamically migrating jobs by considering both the CPU and memory resources of the cluster, the load sharing policy using remote executions is particularly beneficial to data-intensive workloads where most jobs are migratable, and where each job fits in a memory space of a single workstation. The requirement of network speed by the remote-execution-based load sharing scheme is not as high as the network RAM. However, if the memory allocation of a job does not fit in any single workstation in the cluster, the additional memory requirement has to be satisfied by local disks, causing longer execution time.

- The improved load sharing scheme overcomes the limits and combines the advantages of the both schemes. We have shown that this scheme is effective for scalable cluster computing.

Memory allocations of jobs are generated by a Pareto distribution in the experiments presented in this paper. We have also run the simulations on the workloads with different memory demand distributions, and observed consistent performance results with that of the workloads by the Pareto memory demand distributions. The other distributions we have used for comparisons are uniform distribution, exponential distribution, and erlang distribution.

Acknowledgements: We wish to thank Yanxia Qu for participating the work of the load sharing project.

References

[1] A. Acharya and S. Setia, "Availability and utility of idle memory in workstation clusters", *Proceedings of ACM SIGMETRICS Conference on Measuring and Modeling of Computer Systems*, May 1999, pp. 35-46.

[2] A. Barak and A. Braverman, "Memory ushering in a scalable computing cluster", *Journal of Microprocessors and Microsystems*, Vol. 22, No. 3-4, August 1998, pp. 175-182.

[3] F. Douglis and J. Ousterhout, "Transparent process migration: design alternatives and the sprite implementation", *Software — Practice and Experience*, Vol. 21, No. 8, 1991, pp. 757-785.

[4] M. J. Feeley, et. al., "Implementing global memory management systems", *Proceedings of the 15th ACM Symposium on Operating System Principles*, December 1995, pp. 201-212.

[5] M. D. Flouris and E. P. Markatos, "Network RAM", Chapter 16, *High Performance Cluster Computing*, Vol. 1, Edited by R. Buyya, Prentice Hall, New Jersey, 1999, pp. 383-508.

[6] M. Harchol-Balter and A. B. Downey, "Exploiting process lifetime distributions for dynamic load balancing", *ACM Transactions on Computer Systems*, Vol. 15, No. 3, 1997, pp. 253-285.

[7] T. Kunz, "The influence of different workload descriptions on a heuristic load balancing scheme", *IEEE Transactions on Software Engineering*, Vol. 17, No. 7, 1991, pp. 725-730.

[8] E. P. Markatos and G. Dramitinos, "Implementation of a reliable remote memory pager", *Proceedings of the 1996 Usenix Technical Conference*, January, 1996, pp. 177-190.

[9] S. Setia, "The interaction between memory allocation and adaptive partitioning in message-passing multicomputers", *Proceedings of the IPPS Workshop on Job Scheduling Strategies for Parallel Processing*, 1995, pp. 146-164.

[10] A. Silberschatz and P. B. Galvin, *Operating Systems Concepts*, 4th Edition, Addison-Wesley, 1994.

[11] X. Zhang, Y. Qu, and L. Xiao, "Improving distributed workload performance by sharing both CPU and memory resources", *Proceedings of 20th International Conference on Distributed Computing Systems*, (ICDCS'2000), Taipei, Taiwan, April 10-13, 2000, pp. 233-241.

[12] S. Zhou, J. Wang, X. Zheng, and P. Delisle, "Utopia: a load-sharing facility for large heterogeneous distributed computing systems", *Software — Practice and Experience*, Vol. 23, No. 2, 1993, pp. 1305-1336.