In Proc. of the 5th IEEE International Conference on Cluster Computing (Cluster 2003), Hong Kong, Dec. 1-4, 2003.

Towards Load Balancing Support for I/O-Intensive Parallel Jobs in a Cluster of Workstations

Xiao Qin Hong Jiang Yifeng Zhu David R. Swanson
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115, {xqin, jiang, yzhu, dswanson}@cse.unl.edu

Abstract

While previous CPU- or memory-centric load balancing schemes are capable of achieving the effective usage of global CPU and memory resources in a cluster system, the cluster exhibits significant performance drop under I/O-intensive workload conditions due to the imbalance of I/O load. To tackle this problem, we have developed two simple yet effective I/O-aware load-balancing schemes, which make it possible to balance I/O load by assigning I/O intensive sequential and parallel jobs to nodes with light I/O loads. Moreover, the proposed schemes judiciously take into account both CPU and memory load sharing in the cluster, thereby maintaining a high performance for a wide spectrum of workload. Using a set of real I/O-intensive parallel applications in addition to synthetic parallel jobs, we show that the proposed schemes consistently outperform the existing non-I/O-aware load-balancing schemes for a diverse set of workload conditions. Importantly, the performance improvement becomes much more pronounced when the applications are I/O-intensive.

1. Introduction

In a system consisting of a cluster of workstations, load-balancing schemes can improve system performance by attempting to assign work, at run time, to machines with idle or under-utilized resources. Several distributed load-balancing schemes, based on this architecture, have been presented in the literature, primarily considering CPU [9], memory [1], network [6], a combination of CPU and memory [22], or a combination of CPU and network [3]. Although these policies have been effective in increasing the utilization of resources in distributed systems, they have ignored disk I/O. The impact of disk I/O on overall system performance is becoming increasingly significant due to the rapidly growing I/O demands of data-intensive and/or I/O-intensive applications [24] and the widening gap between CPU and disk I/O speeds. This makes storage devices a likely performance bottleneck. Therefore, we believe that for any dynamic load balancing scheme to be effective in this new application environment, it must be made “I/O-aware” [18].

In the first part of this study, we propose a simple yet effective I/O-aware load-balancing scheme for both sequential and parallel jobs, IOCM-RE, which is able to balance the load of a cluster in such a way that CPU, memory, and I/O resources at each node can be simultaneously well utilized under a wide spectrum of workload. For the second part of the study, we apply a preemptive migration technique into IOCM-RE, and develop a scheme that is referred to as IOCM-PM. The experimental results, generated from extensive simulations driven by both synthetic and real-application traces, indicate that, IOCM-RE and IOCM-PM significantly improve the performance of the existing load balancing schemes that only consider CPU and memory.

The rest of the paper is organized as follows. In the section that follows, related work in the literature is briefly reviewed. Section 3 and Section 4 propose two I/O-aware load-balancing policies for parallel jobs, and evaluate the performance of the proposed schemes. Finally, Section 5 summarizes the main contributions of this paper and comments on future directions for this work.

2. Related work

The issue of distributed load balancing for CPU and memory resources has been extensively studied and reported in the literature in recent years. Harchol-Balter et al. [9] proposed a CPU-based preemptive migration policy that was more effective than non-preemptive migration policies. Zhang et al. [22] focused on load sharing policies that consider both CPU and memory services among the nodes. The experimental results show that their policies not only improve performance of memory-intensive jobs, but also maintain the same load sharing quality of the CPU-based policies for CPU-intensive jobs [22].

A large body of work can be found in the literature that addresses the issue of balancing the load of disk I/O

[12][23]. Lee et al. [12] proposed two file assignment algorithms that balance the load across all disks. The I/O load balancing policies in these studies have been shown to be effective in improving overall system performance by fully utilizing the available hard drives. Zhang et al. proposed three I/O-aware scheduling schemes that are aware of the job's spatial preferences [23]. While the above approaches address the issue of load balancing for explicit I/O load, our technique tackles the problem by considering both explicit I/O invoked by application programs and implicit I/O induced by page faults.

Many researchers have shown that I/O cache and buffer are useful mechanisms to optimize storage systems. Ma et al. have implemented active buffering to alleviate the I/O burden by using local idle memory and overlapping I/O with computation [13]. We have developed a feedback control mechanism to improve the performance of a cluster by adaptively manipulating the I/O buffer size [17]. Forney et al. have investigated storage-aware caching algorithms for heterogeneous clusters [8]. Although we focus solely on balancing disk I/O load in this paper, the approach proposed here is also capable of improving the buffer utilization of each node. The schemes presented in this paper is complementary to the existing caching/buffering techniques in the sense that our approaches are able to offer additional performance improvement when combined with active buffering, storage-aware caching, and a feedback control mechanism.

3. I/O-aware Load-Balancing Scheme with Remote Execution (IOCM-RE)

3.1 Infrastructure for Load Balancing

We consider the problem of distributed dynamic load balancing among a cluster of nodes (the terms node and workstation are used interchangeably) connected by a high-speed network, where a centralized node (or "head node") could apply a broadcast mechanism (e.g., *gather* and *scatter* like operations) to handle load distribution in a dedicated computing cluster. As the head node becomes increasingly overloaded with the growth of cluster size, the system will inevitably suffer a significant performance drop when the head node becomes a severe bottleneck. To effectively alleviate the potential burden of the head node, load-balancing schemes can be individually applied to each node, thereby distributing some of the head node's workload among other nodes.

For simplicity, we assume that all nodes are homogeneous. This simplifying assumption should not restrict the generality of the proposed model. This is because the proposed scheme can be extended to handle heterogeneous system by considering variations in the performance characteristics of computing power, memory capacity, and disk speed [16].

We assume that, in a realistic cluster structure, every job has a "home" workstation that it prefers for execution [11]. The rationale behind this home model is two-fold: (1) the input data of a job has been stored in the home node and, (2) the job was created on its home node. When a sequential or parallel job is submitted through the client service (which may be managed and housed by the head node) to its home node, the load manager assigns the job to a workstation (for the sequential job) or a group of workstations (for the parallel job) with the least load. The load manager continues to receive reasonably up-to-date global load information from the head node, which monitors resource utilizations of the cluster and periodically broadcasts global load information to other nodes of the cluster. If the load manager detects that the local node is heavily loaded, a migration will be carried out to transfer an eligible process to a node with the lightest load.

3.2 Design of the IOCM-RE Scheme

In this section, we present IOCM-RE, an effective dynamic I/O-aware load-balancing scheme. Each parallel job consists of a number of tasks, and the terms process and task are used interchangeably throughout this paper. The tasks of a parallel job are assumed to synchronize with one another [7]. Specifically, each task of a parallel job serially computes for some period of time, then a barrier is performed so that each task starts exchanging messages with other processes of the parallel job. Each task is described by its requirements for CPU, memory, and I/O, measured, respectively, by the total time spent on CPU, Mbytes, and number of disk accesses per ms. Each workstation serves several tasks in a time-sharing fashion so that the tasks can dynamically share the cluster resources.

For a parallel job, arriving in its home workstation via the client services, the IOCM-RE scheme attempts to balance three different resources simultaneously in the following manner.

(1) When the I/O load of a node is greater than zero, tasks running on the node are likely to experience waiting time on I/O processing. To alleviate the problem of unevenly distributed I/O load, IOCM-RE selects a group of nodes with lighter I/O load. If there are a number of choices, the one with the smallest value of memory load will be chosen to break the tie. The tasks of the parallel job are assigned to the selected remote nodes satisfying a criterion based on remote execution cost, in addition to load distribution. The criterion guarantees that the response time of the expected execution on the selected remote node is less than the local execution. Formally, the criterion is described as: $r(i, j) > r(k, j) + c_j(i, k)$, where

r(i, j) and *r(k, j)* are the expected response times of task *j* on the local node *i* and on the remote node *k*, respectively. $c_j(i, k)$ is the remote execution cost. More precisely, given a task *j* arrived in node *i* and a candidate remote node *k*, the expected remote execution cost, $c_j(i, k)$, can be estimated as follows,

$$c_j(i,k) = e + d_j^{INIT}\left(\frac{1}{b_{net}^{ik}} + \frac{1}{b_{disk}^{i}} + \frac{1}{b_{disk}^{k}}\right). \qquad (1)$$

where *e* is the fixed cost of migrating the job and executing it on another node, b_{net}^{ik} denotes the available bandwidth of the network link between node *k* and *l*, b_{disk}^{i} is the available disk bandwidth in node *i*. In practice, b_{net}^{kl} and b_{disk}^{k} can be measured by a performance monitor [3]. Accordingly, the simulator discussed in Section 3.3 estimates b_{net}^{kl} and b_{disk}^{k} by storing the most recent values of the disk and network bandwidth. d_j^{INIT} represents the amount of data initially stored on disk to be processed by job *j*, and this amount of data is referred to as *initial data*. Thus, the second term on the right hand side of the above equation represents the time spent in transmitting data over the network and on accessing source and destination disks. In our system model, we assume that all the initial data of a job is transmitted if the job encounters a migration. This assumption is conservative, since the performance of the proposed approach can be further improved if the amount of initial data that must be migrated can be accurately predicted by a monitor at run time.

(2) If no I/O load is imposed on the node, the IOCM-RE scheme considers the node's memory load, defined as the sum of the memory space allocated to the tasks running on the node. When the memory load exceeds the amount of available memory space, the IOCM-RE policy transfers the tasks of the newly arrived parallel job from the overloaded node to the remote nodes that are lightly loaded with respect to memory.

(3) If both the disk I/O and memory resources of the node are well balanced, IOCM-RE attempts to evenly distribute the CPU load. Specifically, if the node is overloaded in terms of CPU resource, the IOCM-RE policy transfers the tasks of the newly arrived job to the remote node with the lightest CPU load. Therefore, IOCM-RE is capable of resulting in a balanced CPU load distribution for systems under a CPU-intensive workload.

3.3 Simulations

To evaluate the performance of the I/O-aware load-balancing scheme presented above, we have performed a large number of trace-driven simulations. We simulate a time-sharing environment, where a cluster comprises 32 workstations. The workload we used is represented by trace files extrapolated from those reported in [9][22]. Specifically, the traces used in our experiments consist of the arrival time, arrival node, request memory size, the actual running time, and I/O access rate. To simulate a multi-user time-sharing environment where a mixture of sequential and parallel jobs are running, the number of parallel jobs in each trace are chosen, respectively, to be 30% and 60% of the total number of jobs in the trace. The number of tasks in each parallel job is randomly generated according to a uniform distribution between 2 and 32. We simulate a bulk-synchronous style of communication, where processes concurrently compute during a computation phase, and then processes will be synchronized at a barrier so that messages can be exchanged among these processes during the communication phase [7]. In our simulation, the time interval between two consecutive synchronization phases is 100 ms.

Although the durations and memory requirements of the jobs are specified in trace data, the I/O access rate of each job is randomly generated according to a uniform distribution. This simplification deflates any correlations between I/O requirement and other job characteristics, but we are able to control the mean I/O access rate as a parameter and examine its impact on system performance. Data sizes of the I/O requests are randomly generated based on a Gamma distribution with the mean size of 256Kbyte, which reflects typical data characteristics for many data-intensive applications [15][19].

The performance metric used in our experiments is the mean *slowdown* [9][22] of all the jobs in a trace. The slowdown of a job is defined as the ratio between the job's execution time in a resource-shared setting and its execution time running in the same system but without any resource sharing.

3.4 Evaluation of the IOCM-RE Scheme

In this section, we compare IOCM-RE against a simple centralized load balancing approach used in a space-sharing cluster, where the nodes of the cluster are partitioned into disjoint sets and each set can only run one parallel job at a time. Since this approach is commonly used for batch systems, we term this load-balancing scheme as BS (Batch System), or PBS-like [4].

To stress the I/O-intensive workload in this experiment, the page fault rate is fixed at a low value of 0.5 No./ms. This workload reflects a scenario where memory-intensive jobs exhibit high temporal and spatial locality of access. A realistic system is likely to have a mixed workload, where some jobs are I/O-intensive and other jobs are either CPU- or memory-intensive. Therefore, we randomly choose 10% of jobs from the trace to be non-I/O-intensive by setting their I/O access rate to 0.

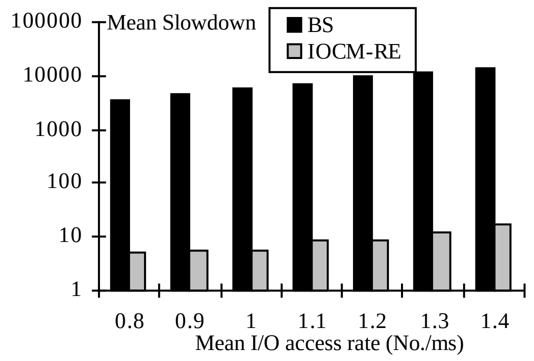

Figure 1. Mean slowdown as a function of the I/O access. Page fault rate is 0.5 No./ms. The traces only contain sequential jobs.

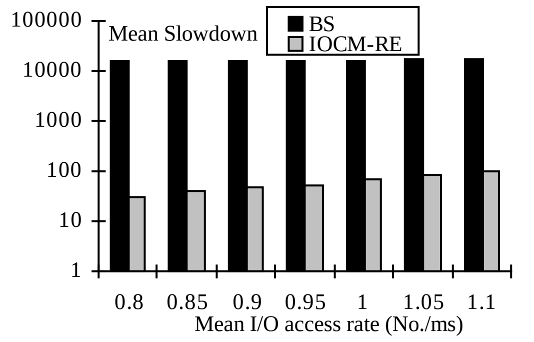

Figure 2.Mean slowdown as a function of the I/O access rate on the traces with 30% parallel jobs. Page fault rate of 0.5No./ms.

Figure 1 and 2 plot slowdown as a function of the mean I/O access rate. While Fig. 1 reports the results for seven traces that only contain sequential jobs, Fig. 2 illustrates the mean slowdown of another seven traces where 30% of the jobs in each trace are parallel. Fig. 1 and 2 indicate that both IOCM-RE and BS experience an increase in the mean slowdown when the mean I/O access rate increases. This is because, as CPU load and memory demands are fixed, high I/O load leads to a high utilization of disks, causing longer waiting time on I/O processing.

We observe from Fig. 1 and 2 that, under the I/O-intensive workload, IOCM-RE is significantly better than BS. The main reason is that it is more difficult to utilize dedicated clusters as efficient, multiuser time-sharing platforms to execute I/O-intensive jobs. Fig. 2 shows that the performance of I/O-intensive jobs drops considerably when a number of parallel jobs are waiting in the queue of the centralized node to be executed. This is because the synchronizations among processes of parallel jobs further decrease the utilization of resources in the system.

In what follows, we compare the performance of IOCM-RE with two existing schemes, namely, CPU-based and memory-based policies. For the purpose of comparison, we have also simulated a so-called NLB (Non-Load Balancing) policy that makes no effort to alleviate the problem of imbalanced load in any resource.

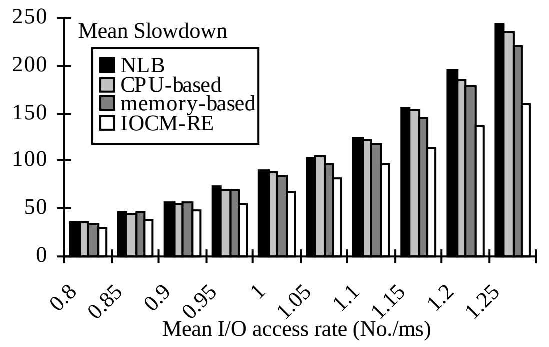

Figure 3. Mean slowdown as a function of the I/O access rate on the traces with 30% parallel jobs. Page fault rate is 0.5 No./ms.

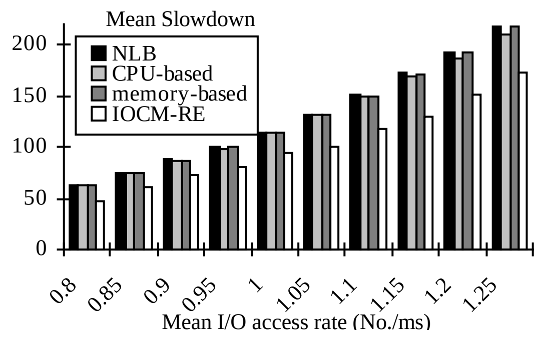

Figure 4. Mean slowdown as a function of the I/O access rate on the traces with 60% parallel jobs. Page fault rate is 0.5 No./ms

Figure 3 and 4 plot slowdown as a function of the mean I/O access rate in the range between 0.8 and 1.25 No./ms with increments of 0.05 No./ms. First, Figure 3 and 4 reveal that the mean slowdowns of the four policies all increase with the I/O load. Second, the results show that the IOCM_RE scheme significantly outperforms the CPU-based and memory-based policies, suggesting that these two policies are not suitable for I/O-intensive workloads. This is because CPU-based and Memory-based policies only balance CPU and memory load, ignoring the imbalanced I/O load under the I/O intensive workload conditions.

After comparing Figure 3 with Figure 4, we realize that, if the mean I/O access rate is fixed, the mean slowdowns of the four policies all increase with the percentage of parallel jobs. This is because a higher percentage of parallel jobs leads to more concurrently running tasks, which in turn increase the overhead in both synchronization and competing with the resources of the cluster. Consequently, waiting time in both CPU and I/O processing becomes longer.

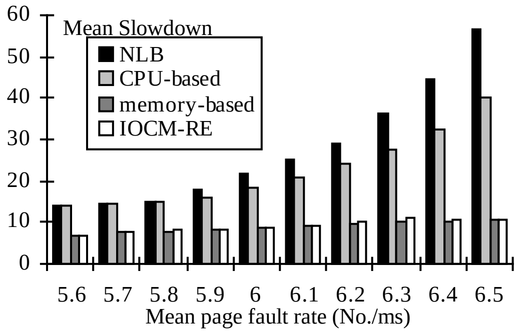

Figure 5. Mean slowdown as a function of the page fault rate on the traces with 30% parallel jobs. Mean I/O access rate is 0.01No./ms.

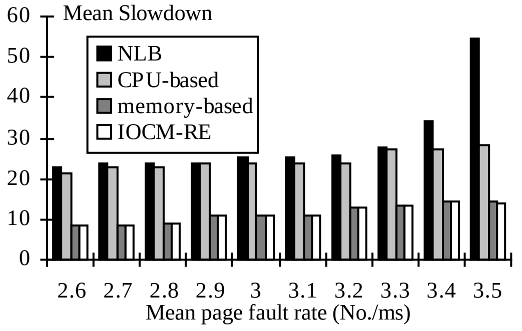

Figure 6. Mean slowdown as a function of the page fault rate on the traces with 60% parallel jobs. Mean I/O access rate is 0.01No./ms.

We now turn our attention to memory-intensive workloads. To simulate a memory intensive workload, the mean I/O access rate is fixed at a low value of 0.01 No./ms, keeping the I/O demands of all jobs at a very low level. In practice, the page fault rates of applications range from 1 to 10 [22]. The results of the mean slowdown as a function of the page fault rate are summarized in Fig. 5 and 6.

As can be seen in Figure 5 and 6, when page fault rate is high and I/O rate is very low, the IOCM-RE and memory-based policies outperform the CPU-based and NLB schemes considerably. These results can be explained by the following reasons. First, the IOCM-RE and memory-based schemes consider the effective usage of global memory, attempting to balance the implicit I/O load, which makes the most significant contribution to the overall system load when page fault rate is high and the explicit I/O load is low. Second, the CPU-based scheme improves the utilization of CPU, ignoring the implicit I/O load resulting from page faults.

4. IO-aware Load Balancing with Preemptive Migration (IOCM-PM)

4.1 Design of the IOCM-PM Scheme

We are now in a position to study IOCM-PM, another I/O-aware load-balancing scheme that improves the performance by considering not only incoming jobs but also currently running jobs.

For a newly arrived job at its home node, the IOCM-PM scheme balances the system load in the following manner. First, IOCM-RE will be invoked to assign the tasks of the newly arrived parallel job to a group of suitable nodes. Second, if the home node is still overloaded, IOCM-PM determines a set of currently running processes that are eligible for migration. The migration of an eligible task is able to potentially reduce the slowdown of the task, and this step substantially improves the performance over the IOCM-RE scheme with non-preemptive migration. The set of eligible migrant tasks is:

$$EM(i,k) = \left\{ j \in M_i \,\middle|\, r(i,j) > r(k,j) + c_j(i,k) \right\}, \qquad (2)$$

where *r(i, j)* and *r(k, j)* are the expected response time of task *j* on the local node *i* and on the remote node *k*, respectively, and $c_j(i, k)$ is the migration cost of task *j*. In other words, each eligible migrant's expected response time on the source node is greater than the sum of its expected response time on the destination node and the expected migration cost. Finally, the eligible processes are preempted and migrated to a remote node with lighter load, and the load of the home node and remote nodes is updated accordingly.

Recall that $c_j(i, k)$ is the cost of task *j* migrated from node *i* to node *k*, and $c_j(i, k)$ is modeled as follows,

$$c_j(i,k) = \begin{cases} e + d_j^{INIT}\left(\dfrac{1}{b_{net}^{ik}} + \dfrac{1}{b_{disk}^{i}} + \dfrac{1}{b_{disk}^{k}}\right) & \text{if remote execution} \quad (3) \\ f + \dfrac{m_j}{b_{net}} + \left(d_j^{INIT} + d_j^{W}\right)\left(\dfrac{1}{b_{net}^{ik}} + \dfrac{1}{b_{disk}^{i}} + \dfrac{1}{b_{disk}^{k}}\right) & \text{otherwise,} \end{cases}$$

where f is the fixed cost for preemptive migration. Like equation (1), the last three terms of both the upper and the bottom line of Equation (3) represent the migration time spent on transmitting data over the network and on accessing source and destination disks, respectively. The second term of the bottom line of Equation (3) is the memory transfer cost. d_j^W and m_j in Equation (3) denote the amount of disk (I/O) data and of main memory data generated at the runtime by the job, respectively. Disk data d_j^W is proportional to the number of write operations that has been issued by the job at the runtime and the average amount of data d_j^{RW} stored by the write operations. d_j^W is inversely proportional to the data re-access rate r_j. Thus, d_j^W is defined by the following equation,

$$d_j^W = \frac{a_j \times \lambda_j \times w_j \times d_j^{RW}}{r_j + 1}, \tag{4}$$

where w_j is the percentage of I/O operations that store data to the local disk, and the number of write operations is a product of a_j, λ_j, and w_j in the numerator. In some I/O-intensive applications, numerous snapshots are spawned by write-only operations. Since the permanent data of snapshots will not be read again by the same job, there is no need to move such write-only data when the job is migrated.

4.2 Evaluation of the IOCM-PM Scheme

To evaluate the IOCM-PM performance, we compare it against the CPU-based, memory-based, and IOCM-RE schemes under 20 I/O-intensive traces, which use the same configurations given in Section 3.3. The results of 10 traces, in which 30 percent of the jobs in each trace are parallel, are plotted in Fig. 7. Similarly, Fig. 8 illustrates the mean slowdowns of other 10 traces where 60 percent of the jobs in each trace are parallel.

It is observed from Fig. 7 and 8 that IOCM-PM consistently performs the best among all the schemes. For example, IOCM-PM improves the performance over IOCM-RE, memory-based and CPU-based schemes by up to 112.8%, 142.6%, and 146.1%, respectively. These results indicate that load-balancing schemes with preemptive migration outperform those schemes with non-preemptive migration under I/O-intensive workload conditions. Consequently, the slowdowns of the CPU-based, memory-based, and IOCM-RE are more sensitive to I/O access rate than IOCM-PM. This performance improvement of IOCM-PM over the IOCM-RE schemes can be explained by the following reasons. First, one problem encountered in the IOCM-RE policy is that IOCM-RE only considers newly arrived jobs for migration, completely ignoring the running tasks that might take advantages of migration from their overloaded node to a remote node with lighter load. In other words, in the non-preemptive schemes, once a task with high I/O demand misses the opportunity to migrate it will never have a second chance. Second, I/O demand of the tasks in a newly arriving job may not be high enough to offset the migration overhead. Third, IOCM-PM provides better migratory opportunities by considering all the running tasks on a node, in addition to the tasks of a newly arrived job.

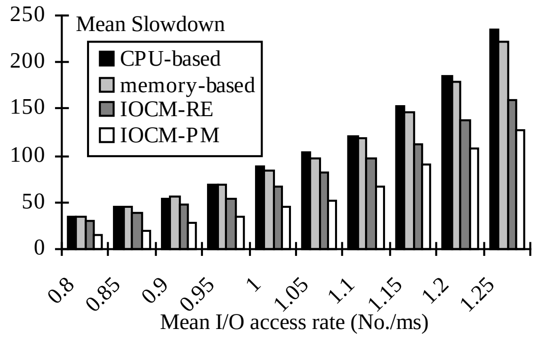

Figure 7. Mean slowdown as a function of the I/O access rate on the traces with 30% parallel jobs. Page fault rate is 0.5 No./ms.

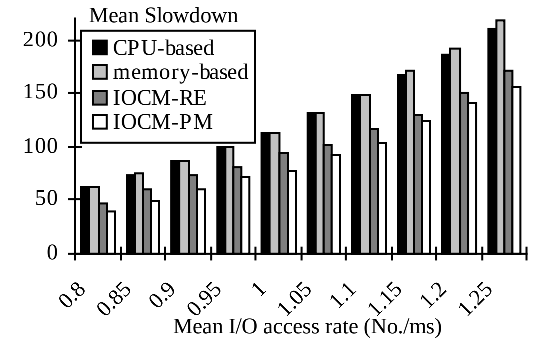

Figure 8. Mean slowdown as a function of the I/O access rate on the traces with 60% parallel jobs. Page fault rate is 0.5 No./ms.

4.3 Real I/O-intensive Parallel Applications

The experimental results reported in the previous sections are obtained from parallel jobs with synthetic I/O requests. To validate the results based on the synthetic I/O workload, we simulate a number of real I/O-intensive parallel applications using five sets of I/O traces collected from the University of Maryland [20]. In particular, we evaluate the mean slowdowns of the following five applications, including both scientific and non-scientific applications with diverse disk I/O demands.

(1) Data mining (Dmine): This application extracts association rules from retail data [14].

(2) Parallel text search (Pgrep): This application is used for partial match and approximate searches, and it is a modified parallel version of the *agrep* program from the University of Arizona [21].

(3) LU decomposition (LU): This application tries to compute the dense LU decomposition of an out-of-core matrix [10].

(4) Titan: This is a parallel scientific database for remote-sensing data [5].

(5) Sparse Cholesky (Cholesky): This application is capable of computing Cholesky decomposition for sparse, symmetric positive-definite matrices [2].

To simulate these I/O-intensive parallel applications, we generate five job traces where the arrival patterns of jobs are extrapolated based on the job traces collected from the University of California at Berkeley [9]. The main purpose of conducting this experiment is to measure the impact of the I/O-aware load balancing schemes on a variety of real applications and, therefore, each job trace consists of one type of I/O-intensive parallel application described above. A 32-node cluster is simulated to run the applications with different I/O demands in each trace.

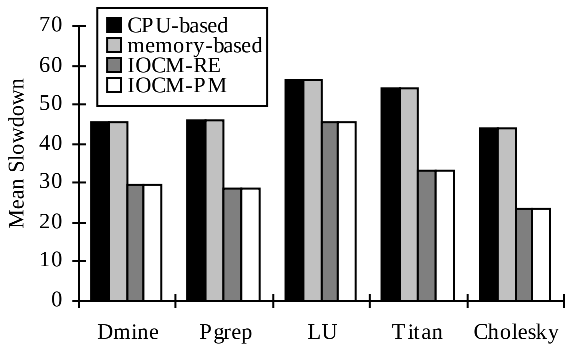

Figure 9. Mean slowdowns of four policies on five applications.

Figure 9 shows the mean slowdowns of the five job traces scheduled by four load-sharing policies. We make three observations. First, the I/O-aware load balancing schemes benefit all I/O intensive applications, and offer a 23.6-88.0% performance improvement in mean slowdown over the non-I/O-aware policies. The performance gain is partially attributed to the low migration cost by virtue of duplicating read-only data. Note that these applications present a very small I/O demand for writes, and the I/O request rates for writes are uniformly low.

Second, IOCM-RE and IOCM-PM yield approximately identical performances. We attribute this result to the fact that, since all jobs running on the cluster in this experiment belong to the same application and have nearly identical CPU and I/O demands, the tasks of a newly arrived parallel job are most likely to become the suitable task for migration due to their low migration cost. In other words, both IOCM-RE and IOCM-PM attempt to migrate the tasks of newly arrived jobs when the local node in the cluster is overloaded and, as a result, IOCM-PM reduces to IOCM-RE when the variance in CPU and I/O demand is minimum.

Third, the trace with LU applications exhibits a larger mean slowdown than the other four traces. Given a fixed job arrival pattern, the mean slowdowns of jobs in a trace depends partially on jobs' total execution time, which in turn is affected by the CPU and I/O execution times of jobs running on a dedicated cluster. Since the total execution time of LU is considerably longer than the other applications, an LU application is expected to spend more time than other applications sharing resources with other jobs running on the cluster. Consequently, there is a strong likelihood that each LU job in the trace will experience a higher slowdown.

Importantly, it is observed from Fig. 9 that the benefits gained from the I/O-aware load balancing schemes are pronounced for Cholesky, Titan, Pgrep, and Dmine, and the performance improvements for these applications are more than 53%. In contrast, the proposed approaches only improve performance in slowdown by 24% for the workload with the LU applications. The reason behind this observation is that LU exhibits a low CPU utilization, thus making most of the running LU jobs in the cluster compete for disk I/O resources.

5. Conclusions

In this paper, we have proposed two **IO**-**C**PU-**M**emory Based load-balancing policies, referred to as *IOCM-RE* (without preemptive migration) and *IOCM-PM* (with preemptive migration), for a cluster of workstations. IOCM-RE employs remote execution facilities to improve system performance, whereas IOCM-PM utilizes a preemptive migration strategy to boost the performance. In addition to CPU and memory utilization, both IOCM-RE and IOCM-PM consider I/O load, leading to a performance improvement over the existing CPU- and Memory-based policies under I/O-intensive workload conditions. Using five real I/O-intensive parallel applications in addition to a set of synthetic parallel jobs with a wide variety of I/O demands, we have demonstrated that applying IOCM-RE and IOCM-PM to clusters of workstations for I/O-intensive workload is not only necessary but also highly effective. Specifically, the proposed schemes offer 24-88% performance improvements in mean slowdown for I/O-intensive applications including LU decomposition, Sparse Cholesky, Titan, Parallel text searching, and Data Mining. When I/O load is low or well balanced, the proposed schemes are able to maintain the same level of performance as the existing non-I/O-aware schemes.

Due to long run times, we have studied the performance of a cluster with 32 workstations. Therefore, future research will deal with a rigorous testing experiment where the performance of a cluster with more than 1000 workstations will be evaluated.

6. Acknowledgements

This work was partially supported by an NSF grant (EPS-0091900), a Nebraska University Foundation grant (26-0511-0019), and a UNL Academic Program Priorities Grant. Work was completed using the Research Computing Facility at University of Nebraska-Lincoln. We are grateful to the anonymous referees for their insightful suggestions and comments.

References

[1] A. Acharya and S. Setia, "Availability and Utility of Idle Memory in Workstation Clusters," *Proceedings of the ACM SIGMETRICS Conference on Measuring and Modeling of Computer Systems*, May 1999.

[2] A. Acharya et al., "Tuning the performance of I/O-intensive parallel applications," *Proceedings of the 4th IOPADS*, Philadelphia, PA, May 1996, pp. 15-27.

[3] J. Basney and M. Livny, "Managing Network Resources in Condor," *Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9)*, Pittsburgh, Pennsylvania, August 2000, pp. 298-299.

[4] B. Bode, D. M. Halstead, R. Kendall, and Z. Lei, "The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters," *Proceedings of the 4th Annual Linux Showcase & Conference*, 2000.

[5] C. Chang, B. Moon, A. Acharya, C. Shock, A. Sussman, and J. Saltz. "Titan: a High-Performance Remote-Sensing Database," *Proceedings of the 13th International Conference on Data Engineering*, Apr 1997.

[6] J. Cruz and Kihong Park, "Towards Communication-Sensitive Load Balancing," *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS 2001)*, Apr. 2001.

[7] A.C. Dusseau, R.H.Arpaci, and D.E. Culler, "Effective Distributed Scheduling of Parallel Workload," *Proceedings of the ACM SIGMETRICS Conference on Measuring and Modeling of Computer Systems*, pp.25-36, May 1996.

[8] B. Forney, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Storage-Aware Caching: Revisiting Caching for Heterogeneous Storage Systems," *Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST 2001)*, 2001.

[9] M. Harchol-Balter and A. Downey, "Exploiting Process Lifetime Distributions for Load Balacing," *ACM Transactions on Computer Systems*, vol. 3, no. 31, 1997.

[10] B. Hendrickson and D. Womble, "The torus-wrap mapping for dense matrix calculations on massively parallel computers," *SIAM J. Sci. Comput.*, 15(5), Sept. 1994.

[11] R. Lavi and A. Barak, "The Home Model and Competitive Algorithm for Load Balancing in a Computing Cluster," *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS 2001)*, Apr. 2001.

[12] L. Lee, P. Scheauermann, and R. Vingralek, "File Assignment in Parallel I/O Systems with Minimal Variance of Service time," *IEEE Trans. on Computers*, Vol. 49, No.2, pp.127-140, 2000.

[13] X. Ma, M. Winslett, J. Lee, and S. Yu, "Faster Collective Output through Active Buffering," *Proceedings of the International Symposium on Parallel and Distributed Processing, (IPDPS 2002)*, 2002.

[14] A. Mueller, "Fast sequential and parallel algorithms for association rule mining: A comparison," *Technical Report CS-TR-3515*, University of Maryland, College Park, August 1995.

[15] B. K. Pasquale and G.C. Polyzos, "Dynamic I/O Characterization of I/O Intensive Scientific applications," *Proceedings of the Supercomputing 1994*, pp. 660 - 669.

[16] X. Qin, H. Jiang, Y. Zhu, and D. Swanson, "Dynamic Load Balancing for I/O-Intensive Tasks on Heterogeneous Clusters," *Proceedings of the 10th International Conference on High Performance Computing (HiPC 2003)*, December 17-20, 2003, Hyderabad, India.

[17] X. Qin, H. Jiang, Y. Zhu, and D. Swanson, "Dynamic Load balancing for I/O- and Memory-Intensive workload in Clusters using a Feedback Control Mechanism," *Proceedings of the 9th International Euro-Par Conference on Parallel Processing (Euro-Par 2003)*, Klagenfurt, Austria, August 26- 29, 2003, pp. 224-229.

[18] X. Qin, H. Jiang, Y. Zhu, and D. Swanson, "A Dynamic Load Balancing Scheme for I/O-Intensive Applications in Distributed Systems," *Proceedings of the 32nd International Conference on Parallel Processing Workshops (ICPP Workshop 2003)*, Oct. 6-9, 2003.

[19] J.O. Roads, et al., "A Preliminary Description of the Western U.S. Climatology", *Proceedings of the Ninth Annual Pacific Climate (PAClim) Workshop*, September 8, 1992.

[20] M. Uysal, A. Acharya, and J. Saltz. "Requirements of I/O Systems for Parallel Machines: An Application-driven Study," *Technical Report, CS-TR-3802*, University of Maryland, College Park, May 1997.

[21] S. Wu and U. Manber, "agrep - A fast approximate pattern-matching tool," *USENIX Conference Proceedings*, pp. 153–162, San Francisco, CA, Winter 1992. USENIX.

[22] X. Zhang, Y. Qu, and L. Xiao, "Improving Distributed Workload Performance by Sharing both CPU and Memory Resources," *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, Apr. 2000.

[23] Y. Zhang, A. Yang, A. Sivasubramaniam, and J. Moreira, "Gang Scheduling Extensions for I/O Intensive Workloads," *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, 2003.

[24] Y. Zhu, H. Jiang, X. Qin, and D. Swanson, "A Case Study of Parallel I/O for Biological Sequence Analysis on Linux Clusters", *Proceedings of the 5th IEEE International Conference on Cluster Computing (Cluster 2003)*, Hong Kong, December 1-4, 2003.