Towards Communication-Sensitive Load Balancing*

John Cruz Kihong Park†
Network Systems Lab
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

Abstract

Recent studies have shown that if the overhead associated with process migration is not high when compared to the estimated lifetime of processes, migrating isolated processes to underutilized workstations can improve system performance. However, if tightly-coupled processes from groups of communicating processes or other existing dependencies are separated, system performance can decrease due to increased communication cost. Current load balancers do not make effective use of communication dependencies in their algorithms. We have implemented a communication-sensitive load balancer for DUNES—a user-level distributed operating system—that uses run-time communication pattern between processes when balancing load. We examine communication-sensitive load balancing under different workload conditions and compare its performance against a load balancer that is not communication-aware. Our results show that a communication-sensitive load balancer performs substantially better than a load balancer that is not communication-aware under workloads that have a "good-mix" of CPU-bound and I/O-bound processes.

1. Introduction

In workstation clusters where workstations are connected via high-speed networks, dynamic load balancing algorithms can improve system performance by utilizing idle resources [5, 6]. Further, it has been observed that process lifetime distributions are heavy-tailed [3]. The prolonged presence of long-lived processes on workstations results in uneven utilization of resources. Dynamic load balancers can utilize system resources efficiently by balancing load while focusing on long-lived processes to amortize migration cost [3, 4].

In this paper, we show the effectiveness of a communication-sensitive load balancer under different workloads and compare its performance against a load balancer that is not communication-aware. To evaluate performance, we have implemented communication-sensitive load balancing in DUNES [1], a user-level distributed operating system. We observe that the communication-sensitive load balancer performs substantially better if the workload consists of a "good-mix" of CPU and I/O intensive processes. We also demonstrate the effects of inter-process arrival times and number of idle hosts available on performance of the communication-sensitive load balancer.

2. Communication-Sensitive Load Balancing

Our communication-sensitive load balancer uses I/O load statistics in addition to CPU load statistics. The key performance metric is the communication-computation ratio (CCR) associated with a process. CCR captures the I/O intensity of a process, and is the number of bytes read/written by a process per unit CPU time it receives. If the CCR of a process is high, it implies that the process performs a lot of I/O relative to the CPU time it receives. A process that is tightly-coupled does a lot of I/O with its peers and hence has a high CCR. Meanwhile, an isolated or a loosely-coupled process performs fewer I/O operations which results in lower CCR. To select a process that has high CPU usage and low I/O coupling for migration, we take the weighted average of the normalized

*This is an abstract. This work was supported in part by NSF grant ESS-9806741.

†Additionally supported by NSF grants ANI-9714707, ANI-9875789 (CAREER), EIA-9972883, ANI-0082861 (ITR), and grants from the Purdue Research Foundation, Santa Fe Institute, Sprint, and Xerox.

CPU utilization and CCR:

$$\omega \;=\; \alpha \cdot \text{CPU utilization} \;+\; (1-\alpha) \cdot \text{CCR}$$

Note that if α is 1.0, the procedure uses CPU utilization only. This type of load balancer is called CPU-usage based load balancer. If α is less than 1.0, the load balancer is communication-sensitive. Decreasing α increases the communication-sensitivity of the load balancer. The CPU-usage based load balancer selects the process with the highest CPU utilization for migration. If the CPU utilization of a process is high, it implies that the process blocks less. A process that is I/O intensive is likely to block more when waiting for data to be produced or consumed. This implies that CPU utilization captures the I/O nature a process to some extent. However, if a host is over-utilized, all processes have similar processor utilization, and it is difficult to differentiate between CPU and I/O processes based on utilization alone.

We have implemented communication-sensitive load balancing in DUNES. DUNES (**D**istributed **UN**ix **E**xten**S**ion) [1] is a user-level distributed operating system designed using the library operating system [2] approach. Distributed operating system functionality is injected into a commodity operating system by redefining the system call interface. Thus application binaries need not be recompiled. DUNES enables transparent processor sharing by migrating processes across different workstations. Most importantly, it handles dependencies such as file access, interprocess communication, signals and parent/child relationship. It provides a complete *single system image* to the user. DUNES has a number of performance enhancement features. including active end-point caching and passive end-point caching. These caching mechanisms reduce the cost of communication between processes that are separated due to migration.

3. Workload Generation

Benchmark workload consists of both CPU-bound and I/O-bound processes. A CPU-bound process uses CPU only whereas an I/O-bound process uses CPU and performs I/O. The process composition—number of I/O-bound and CPU-bound processes present in the workload—can be varied to generate different workloads with varying number of CPU-bound and I/O-bound processes. When a process is generated, its type is chosen based on the desired composition specified. If the process type is CPU-bound, a single process is generated that is CPU-intensive. On the other hand, if the process type is I/O-bound, then a pair of communicating processes are generated. The I/O intensity of the I/O-bound process being generated can be varied by modifying the workload specification.

The workload generator is a program with a number of parameters which the user can modify to generate different workloads. The pseudo-code of the workload generator is shown in Figure 3.1. The parameters

```
Foreach process generated
   * Set process LIFETIME based on distribution
     desired
   * Set process INTER_PROCESS_ARRIVAL_TIME based
     on distribution desired
   * Set I/O VOLUME
   * Set process type probabilistically based
     process composition desired by the user
   * If CPU-bound process to be generated
        Append to tracefile details of CPU-bound
        process
        Update the number of processes generated
     End-if
   * If I/O-bound process to be generated
       Generate a set of coupled I/O processes
       with the I/O volume specified
       Update the number of processes generated
     End-if
End-foreach
```

Figure 3.1. Structure of workload generator

that can be specified include arrival process distribution, CPU time required for completion, image size, I/O intensity of I/O-bound processes and composition of processes in the workload. Based on the parameters provided, a tracefile is generated which contains the process generation sequence and characteristics of each process. For each generated process, the inter-process arrival time is chosen from the distribution specified by the user. Based on the process composition desired, a biased coin is flipped to check if the process to be generated is a CPU-bound process or a pair of I/O-bound processes. The CPU time required which influences the lifetime of the process and the image size of the process are generated based on the distributions specified by the user. If the process is CPU-bound, its required CPU utilization is set to 1. This can be changed by the user to any desired value less than 1. For an I/O bound process, based on the type of coupling desired by the user, a set of processes that communicate with each other is generated.

4. Performance Results

4.1. Experimental Set-up

The experiments described in the following sections were conducted on a dedicated network of workstations housed in the Network Systems Lab. We employ twenty five x86 machines—10 Pentium II at 399 Mhz, 10 Pentium III at 555 Mhz and 5 Pentium III at 740 Mhz—running Solaris 2.6, and a number of Ultra Sparc workstations running Solaris 2.6 in various benchmark configurations. These machines are connected by FastEthernet switches. The generated processes are long-lived and require 90 CPU seconds to complete. Performance is measured by the actual wall time completion time which is a function of load balancing and the resulting system state. The weight α, in the communication-sensitive load balancer, is set to 0.5 in the experiments described below. This makes the communication-sensitive load balancer give equal weight to both CPU and I/O nature of a process.

4.2. Effect of I/O Intensity

When tightly-coupled processes are separated, the cost of communication between the communicating processes increases, as high-speed IPC becomes low-speed network I/O. This results in increased completion times for these processes. Therefore, in a system where there are tightly-coupled and loosely-coupled processes, the mean completion time of processes increases if tightly-coupled processes are separated. A communication-sensitive load balancer that is aware of the communication pattern between processes tends to keep tightly-coupled processes together. On the other hand, a load balancer which takes decision based on CPU-usage only, may separate processes that are tightly-coupled.

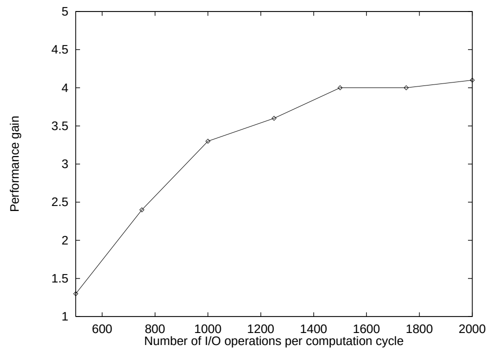

Figure 4.1. Performance gain when using a communication-sensitive load balancer vis-à-vis a CPU-usage based load balancer as I/O intensity is varied.

In Figure 4.1, the performance gain, measured as a ratio of mean completion times obtained by using a communication-sensitive load balancer and a CPU-usage based load balancer, is shown. As the I/O intensity of I/O-bound processes increases, the tightness of coupling between processes also increases which increases the performance gain.

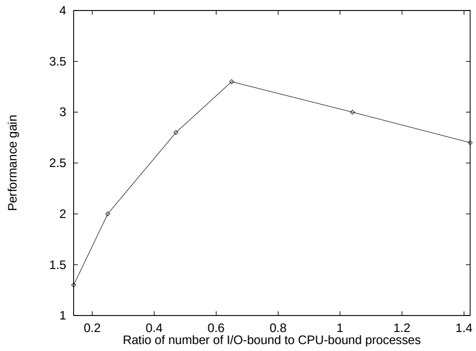

Figure 4.2. Performance gain as a function of process composition in the workload. A "good mix" results in maximum performance gain.

4.3. Effect of Process Composition

For the communication-sensitive load balancer to perform better than the CPU-usage based load balancer, apart from I/O-bound processes being tightly-coupled, there should be a "good mix" of both CPU-bound and I/O-bound processes present in the system whenever a load balancing decision is to be made. Ideally, if the system has one CPU-bound process and many I/O-bound processes present every time the load balancers profile the load to take decisions, the performance difference between the load balancers will be high. The communication-sensitive load balancer will most likely migrate the CPU-bound process and the CPU-usage based load balancer, unaware of the nature of the processes, is likely to select an I/O-bound process for migration. In this section, we show how the the ratio of number of I/O-bound to CPU-bound processes, which decides the instantaneous composition of processes in the system, affects the performance of the communication-sensitive load balancer.

Figure 4.2 shows how the performance gain in completion time varies when the ratio of number of I/O-bound to CPU-bound processes is varied. As we increase the number of I/O-bound processes, the performance gain starts to increase. This is mainly due to the decisions of communication-sensitive load balancer to migrate fewer I/O-bound processes than the CPU-usage based load balancer. If the number of I/O-bound processes is increased further, the number of processes retained on the home base by the communication-sensitive load balancer also increases.

4.3.1 Effect of Inter-process Arrival Time

The communication-sensitive load balancer retains the I/O-bound processes on the home base while migrating CPU-bound processes. If the mean inter-process arrival time is decreased, the number of processes on the home base is increased which decreases the CPU utilization of the processes due to increased resource contention. The decrease in CPU time received causes an increase in completion time. As the inter-process arrival time is increased, the I/O-bound processes present on the home base get more CPU time due to decreased resource contention which leads to a decrease in completion time. A CPU-usage based load balancer migrates processes unaware of their I/O nature and hence has higher completion times. When the inter-process arrival time is large, the CPU-usage based load balancer tends to migrate only few of the tightly-coupled processes since the home base itself is under-utilized. This results in the completion time decreasing further as fewer processes from a group of communicating processes are separated. Figure 4.3 shows the variation in performance gain as a function of mean inter-process arrival time.

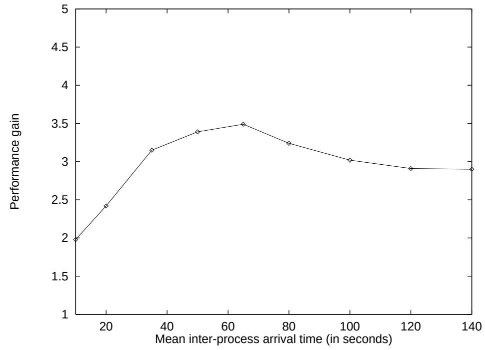

Figure 4.3. Performance gain as a function of load—mean inter-process arrival time.

4.4. Impact of Resource Availability

Increase in the availability of idle resources leads to better performance by the communication-sensitive load balancer by migrating loosely-coupled processes. We observe that the CPU-usage based load balancer's speed-up initially increases. However, it begins to decreases as the number of hosts is increased further. This is mainly due to the migrations of many tightly-coupled processes.

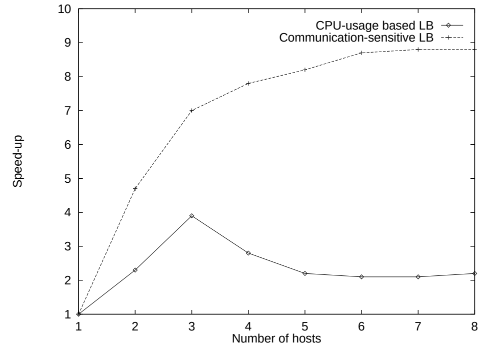

Figure 4.4. Speed-up as the number of participating workstations is increased.

Figure 4.4 shows the speed-up obtained with both load balancers. As the number of participating hosts increases, there is noticeable speed-up with both load balancers. The communicating-sensitive load balancer, aware of the high penalty incurred if tightly-coupled processes are separated, performs better than the CPU-usage based load balancer by migrating mainly CPU-bound processes to idle hosts.

References

[1] J. Cruz and K. Park. Towards performance-driven system support for distributed computing in clustered environments. *Journal of Parallel and Distributed Computing*, 59(2):132–154, 1999.

[2] D. Engler, M. Kaashoek, and J. O. Jr. Exokernel: an operating system architecture for application-level resource management. In *Proc. 15th ACM Symp. on Operating System Principles*, pages 251–266, 1995.

[3] M. Harchol-Balter and A. B. Downey. Exploiting lifetime distributions for dynamic load balancing. In *Proceedings of ACM Sigmetrics '96 Conference on Measurement and Modeling of Computer Systems*, pages 13–24, 1996.

[4] M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. In *Proc. 8'th International Conference on Distributed Computing Systems*, pages 104–111, 1988.

[5] A. Thomasian. A performance study of dynamic load balancing in distributed systems. In *7th International Conference on Distributed Computing Systems*, pages 178–184, 1987.

[6] S. Zhou and D. Ferrari. A measurement study of load balancing performance. In *7th International Conference on Distributed Computing Systems*, pages 490–497, 1987.