Analysis and Measurement of the Effect of Kernel Locks in SMP Systems

Akihiro Kaieda and Yasuichi Nakayama

Department of Computer Science,
The University of Electro-Communications

Chofu, Tokyo 182-8585 Japan

Atsuhiro Tanaka, Takashi Horikawa,
Toshiyasu Kurasugi and Issei Kino

C&C Media Research Laboratories,
NEC Corporation
Kawasaki 216-8555 Japan

Abstract

This paper reports the use of case studies to evaluate the performance degradation caused by the kernel-level lock. We define the lock ratio as a ratio of the execution time for critical sections to the total execution time of a parallel program. The kernel-level lock ratio determines how effective programs work on Symmetric MultiProcessor systems. We have measured the lock ratios and the performance of three types of parallel programs on SMP systems with Linux 2.0: matrix multiplication, parallel make, and WWW server programs. Experimental results show that the higher the lock ratio of parallel programs, the worse their performance becomes.

keywords: SMP Systems, Operating Systems, Parallel Programs, Performance Evaluation, Kernel Lock

1 Introduction

SMP (Symmetric MultiProcessor) systems have recently become quite common even in the PC market, and free UNIX-compatible operating systems (e.g., Linux and FreeBSD) supporting the effective use of multiple processors have been developed. Some application programs (e.g., server daemons, thread libraries) which consist of several processes work effectively on SMP systems, but their performance is sometimes degraded by the need to share resources in the UNIX kernel. When one process uses shared resources, other processes are excluded and serialized by the UNIX kernel.

Amdahl's law [1] and Gustafson's law [4] are usually used for estimating the efficiency of application programs. According to these laws the effectiveness of a program on massively parallel systems is determined by execution time of serial and parallel parts of that program. When there are several programs running on a system, however, the laws are not able to estimate the performance of on an individual program because kernel-level locks between the programs are beyond their scope.

This paper shows, through case studies, that the performance of SMP system varies with changes in the kernel-level lock ratio. We define the lock ratio as the ratio of the time for critical sections of a process to the total execution time. Our goal in this study was to quantitatively evaluate the effects of different lock ratios on the performance of SMP systems.

In particular, one of the most widely used operating systems, Linux 2.0, has only one type of kernel-level lock. In this case the entire kernel is locked whenever any process is in a kernel (supervisor) mode. Thus, the lock ratio on the Linux 2.0 is almost equivalent to the ratio of total execution time of system calls to the total execution time. The performance of parallel programs deteriorates when a lot of system calls are invoked.

We have measured the performance of three types of parallel programs — matrix multiplication, parallel make, and WWW server programs — on two types of SMP systems (with 1 to 4 CPUs) with Linux 2.0. These are typical parallel programs which have different kernel-level lock ratios from one another. In order to measure the lock ratios and the performance of these parallel programs accurately, we have employed a hybrid event tracer consisting of a hardware tracer and a software probe inserted into the kernel [5]. Experimental results show that the higher the lock ratio, the worse the performance. In particular, the performance of WWW server programs, which have considerably high lock ratios, are worse on SMP systems than on uniprocessor systems.

The remainder of this paper is organized as follows. In Section 2 we describe some related work, and in Section 3 we define the lock ratio and discuss performance degradation caused by the lock ratio. In Section 4 we show the lock ratios and the performance instrumentations of three applications. And in Section 5 we conclude by summarizing the paper very briefly and mentioning some future work.

2 Related Work

Amdahl's law [1] has been widely used for evaluating the **speedup** effect of parallel numerical computations (e.g., matrix multiplication, LU-decomposition, CG-kernel, FFT). This effect is determined by the amounts of time spent on the serial and parallel sections of a target application program. Amdahl assumed that the size of a target problem was constant, but Gustafson pointed out that a size of a target problem will increase as the number of available CPU increases [4].

Sun and Rover [10] were the first to formally define scalability, and they studied the relation between their scalability and other measures of parallel performance.

The studies mentioned above, however considered only the behavior of application programs. Serialization caused by kernel-level lock was not considered.

Chen *et al.* [3] and Lai *et al.* [7] studied the comparison of the performance of personal computer operating systems (e.g., Linux, FreeBSD, NetBSD, Solaris, Windows). They measured system call latency, context switch latency, memory bandwidth, file system performance, network bandwidth, and applications performance. But all the systems they used were for uniprocessor personal computers, so the performance of SMP systems have not yet been carefully examined.

Kurasugi *et al.* developed a performance model for SMP systems by using queueing network models, and they reported the performance predicted for SMP systems showed good agreement with simulation results. Their performance model is one of the two-layer queueing network models [6].

In the work reported in this paper, we measured the lock ratios and the performance of parallel programs running on a SMP system. Our results show that the degradation of the performance of parallel programs increases with the kernel-lock ratio.

3 Performance Degradation Caused by the Kernel-Level Lock

Operating systems for SMP systems manage shared resources in the kernel as exclusive parts. This is because the kernel maintains the consistency of shared resources when processes try to access them simultaneously. These exclusive parts are called critical sections.

When a process tries to hold the shared resource, it behaves as follows.

1. It transits from the user mode to the kernel mode by invoking a system call.
2. It acquires kernel-level locks for using shared resources, and it holds them.
3. It enters the critical section and continues its execution.
4. It releases the locks and transits to the user mode.

When many processes try to enter a critical section simultaneously, only one process is able to enter it; the others are excluded and serialized by the kernel. Thus, the performance of parallel programs composed of several processes

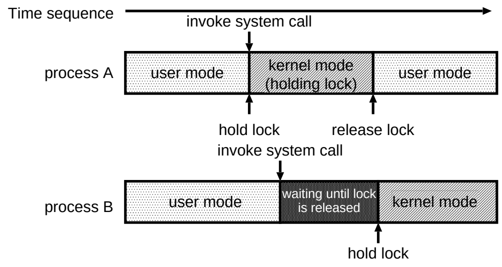

Figure 1: The kernel-level lock on Linux 2.0

sometimes deteriorates if those processes often try to use shared resources at the same time.

For this reason, it is important that this conflict at the critical sections be analyzed. In the UNIX operating systems this conflict is caused by the lock operations in the kernel that manage the critical section. Thus we need to analyze how kernel lock occurs in UNIX.

To quantitatively evaluate the effects of different lock ratios on the performance of application programs running on SMP systems, we defined the lock ratio for a process as a ratio of the execution time in critical sections to the total execution time.

The lock ratios for parallel programs can be measured on uniprocessor systems before execution on SMP systems because the execution time in the critical section on SMP systems is almost equivalent to that on the uniprocessor systems. Then we can measure lock ratios for parallel programs on uniprocessor systems.

In Linux 2.0, for example, the kernel has only one type of kernel-level lock.

When any process is in a kernel mode because it invoked a system call, the entire kernel is locked and the other processes have to wait for the process to release the lock even if they are invoking different system calls (Fig. 1). The lock ratio on the Linux 2.0 kernel is therefore almost equivalent to the ratio of the execution time of system calls to the total execution time. The more system calls invoked, the worse the performance of parallel programs on Linux 2.0.

Library functions may invoke system calls even when the programmer does not explicitly use system calls. In Linuxthreads [8], for example, most of the thread primitives invoke system calls.

The kernel-level lock operations on Linux 2.0 are implemented using spinlock. The waiting ratio for a process is defined as the ratio of waiting time until the lock is released by other process. We predict that the higher the lock ratio of parallel programs have, the higher the waiting ratio they become. Thus their performance becomes worse.

We predict that some parallel programs (e.g., matrix multiplication) will have lock ratios low enough that the programs will run better on SMP systems. These will be programs executed mostly in the user mode. The parallel programs which have the intermediate lock ratios (e.g., parallel make) may invoke a few system calls, and their performance may therefore be worse than that of programs invoking no system calls. The parallel programs which have excessively high lock ratios (e.g., WWW server programs) cannot take advantage of parallel processing on SMP systems. The processes in these programs must frequently invoke system calls resulting in their holding shared resources.

4 Case Studies

We measured the lock ratios and performances of three types of parallel programs, and in this section we show the causal relations between lock ratios and speedup.

4.1 Measured System

In our experiment, we used two types of SMP PC systems. One was composed of two CPUs (180MHz Intel PentiumPro processors), 256KB second-level caches, and 128MB of main memory, the other was composed of four CPUs (500MHz Intel PentiumIII Xeon processors), 512KB second-level caches and 256MB of main memory.

The operating system was Linux 2.0.36. Because in Linux 2.0, the lock ratios are equivalent to the ratios of the execution time of the system calls to total execution time, we could measure lock ratios easily. In addition, we could easily insert the codes for the measurement because Linux 2.0 is free implementations of UNIX.

4.2 A Measurement Method for SMP System Behavior

The lock ratios are measured by analyzing the behavior of operating systems. We observe the behavior of operating systems by the events of processes (e.g., invoking system call, process scheduling). The events of processes are detected by the codes inserted in the kernel sources.

Some measurements for the event of process on SMP systems are required as follows.

1. The time of events on each processor is managed by the same time sequence.

2. The overhead created by the measurement is small.

The measurements by a software alone tool such as system call (e.g., `gettimeofday`) is used in general. These measurements must accomplish detecting, collecting, recording, and sometimes analyzing events. Therefore, those overheads are very large.

Other measurements are by a hardware counter in the processor (e.g., Pentium TSC operation). This resolution is high because of the clock cycle counter and this overhead is small because of a few assembly operations. However, the counter value in each processor is different on SMP systems because boot time of each processor is different. Therefore, this measurement is not suitable on SMP systems.

In our experiments we measured the lock ratios by using a hybrid event tracer consisting of a hardware tracer and a software probe inserted into the kernel [5].

The hybrid event tracer records an event data as follows

1. A software probe detects events.

2. It then output event ID and related information to the interface board. These data are sent to the hardware tracer.

3. The hardware tracer produces an event record according to data it receives from the interface board and the hardware time counter.

4. The hardware tracer writes the data directly to a hard disk, and this trace data becomes a record of the event sequence.

A hybrid event tracer is suitable for use on SMP systems because the time sequence of events in each processor is managed by the timer in the hardware tracer.

Soft Probe overhead is about 250 nanoseconds on 500MHz PentiumIII Xeon PC.

4.3 Experimental Results

In this section we describe and discuss the speedup measured for three types of parallel programs: matrix multiplication, parallel make, and WWW server programs.

The speedup is given by

$$Speedup = \frac{execution\ time\ on\ the\ uniprocessor\ system}{execution\ time\ on\ the\ SMP\ system}$$

We used the uniprocessor kernel and the SMP kernel, on the uniprocessor systems and on the SMP systems respectively. We compared performance on the SMP systems with 1 to 4 CPUs. We referred to the experimental results on the SMP system with 1 CPU (used SMP kernel) as “1 CPU SMP”.

Table 1: Experimental results on the SMP system with 2 CPUs (matrix multiplication program).

number of CPUs	execution time (sec)	speedup	lock ratio (%)	waiting ratio (%)
1 CPU	415	1.00	less than 0.1	–
1 CPU SMP	418	0.99	–	–
2 CPU	213	1.95	–	less than 0.1

4.3.1 Matrix Multiplication

We implemented a program, that multiplies one 1000×1000 matrix by another. The matrix multiplication is divided into several multiplications of partial matrices. These multiplications of partial matrices are executed in parallel and independently by each process on SMP systems. This matrix multiplication program can be executed on main memory.

The matrix for the results was allocated on the shared memory region, which was not excluded because the results of partial matrix multiplications were independent.

The experimental results for this matrix multiplication program are listed in Table 1 and 2, where it can be seen that the lock ratio was less than 0.1%. This is because the processes in this program rarely invoke system calls during multiplication. This program therefore performs very well on SMP systems.

Table 2: Experimental results on the SMP system with 4 CPUs (matrix multiplication program).

number of CPUs	execution time (sec)	speedup	lock ratio (%)	waiting ratio (%)
1 CPU	100	254	less than 0.1	–
1 CPU SMP	257	0.99	–	–
2 CPU	127	1.99	–	less than 0.1
3 CPU	87	2.93	–	less than 0.1
4 CPU	64	3.96	–	less than 0.1

4.3.2 Parallel Make

We have measured GNU make 3.77. Make is a program designed to simplify the maintenance of other programs. Because "make -j max_jobs" runs on parallel the number of jobs that was appointed by "max_jobs" at any one time on SMP systems, "make -j" is called the parallel make.

In our experiment, we built Linux 2.2.14. We appointed "max_jobs" with 2 and 4, on the SMP systems with 2 CPUs and 4 CPUs respectively.

Table 3 and 4 show these results. *make* forks several commands. The lock ratio for *make* is determined by the lock ratios these commands have. Table 5 shows several typical commands, their execution time ratios and lock ratios. The execution time ratio is defined as a ratio of the execution time of each command to the total execution time.

The greater part of execution time is spent by *cc1*. It takes charge of lexical

Table 3: Experimental results on the SMP system with 2 CPUs (parallel make program).

number of CPUs	execution time (sec)	speedup	lock ratio (%)	waiting ratio (%)
1 CPU	600	1.00	5.31	–
1 CPU SMP	636	0.94	–	– 0
2 CPU	331	1.81	–	1.52

analysis, syntax analysis, intermediate language generator and optimization. The greater part of these processing is executed on memory. Thus, *cc1* invokes few system calls, and its lock ratio is 1.97% on SMP system with 2 CPUs. In contrast with *cc1*, the lock ratio for *cpp* is 14.8% because a lot of execution time is spent reading sources files (by `read` system call) and expanding the macro.

The average of lock ratio for *make* is 5.31% and 4.09% on SMP system with 2 CPUs and 4 CPUs respectively.

4.3.3 WWW server

We have also measured the performance of the Apache 1.3.9 WWW server [2], which has come into widespread use all over the world. The Netcraft Web Server Survey [9] reports that over 50% of all public Internet websites use Apache WWW Servers.

This server creates several child processes for handling requests from clients. It would run better on SMP systems if the requests were handled by these child

Table 4: Experimental results on the SMP system with 4 CPUs (parallel make program).

number of CPUs	execution time (sec)	speedup	lock ratio (%)	waiting ratio (%)
1 CPU	217	1.00	4.09	–
1 CPU SMP	223	0.97	–	–
2 CPU	115	1.90	–	1.27
3 CPU	80	2.73	–	2.41
4 CPU	62	3.48	–	3.46

Table 5: The lock ratio for typical commands of which compose *make* (on SMP systems with 2 CPUs).

commands (number of called)	execution time ratio (%)	lock ratio (%)	waiting ratio (%)
`make`(74)	4.97	11.8	2.24
`ld`(29)	0.39	32.3	4.03
`gcc`(298)	0.96	33.9	4.12
`cc1`(289)	74.8	1.97	1.00
`cpp`(297)	14.1	14.8	3.15
`ar`(13)	0.09	47.4	6.47

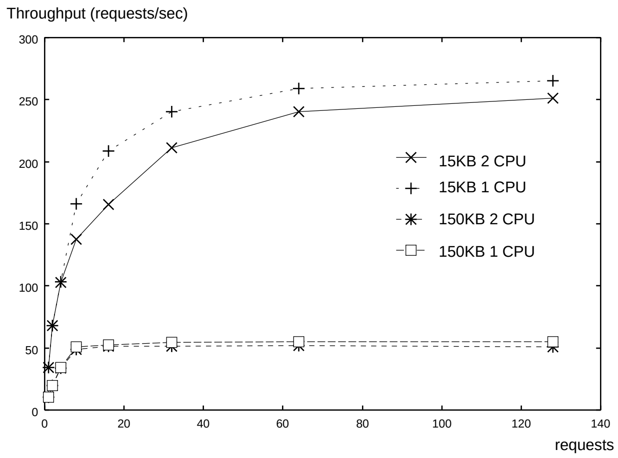

Figure 2: Throughput for the Apache 1.3.9 on the SMP system with 2 CPUs.

processes in parallel.

A client program on another PC (Celeron 450MHz) connected by a 100Mbit Ethernet simulates parallel connections and send many requests simultaneously to the server. The file size requested by clients is either 15KB or 150KB.

The throughput for a WWW server program is given by

$$Throughput(requests/sec) = \frac{requests}{response\ time + clients\ processing\ time}$$

The clients processing time is regarded as 0 because these clients don't use the data sent from the server.

At first, we show experimental results on SMP system with 2 CPUs. Experimental results show that the performance of the server on the SMP systems become worse than that of the uniprocessor system (Fig. 2). Note that SMP kernel includes overhead of SMP processing.

Thus, we have measured the lock ratio for WWW server and its speedup on

Table 6: Experimental results on the SMP system with 2 CPUs (WWW server, 64 requests, file size 15KB).

number of CPUs	execution time (sec)	speedup	lock ratio (%)	waiting ratio (%)
1 CPU	0.25	1.00	49.0	–
1 CPU SMP	0.26	0.96	–	–
2 CPU	0.27	0.93	–	24.3

Table 7: Experimental results on the SMP system with 2 CPUs (WWW server, 64 requests, file size 150KB).

number of CPUs	execution time (sec)	speedup	lock ratio (%)	waiting ratio (%)
1 CPU	1.16	1.00	63.6	–
1 CPU SMP	1.21	0.96	–	–
2 CPU	1.21	0.96	–	35.9

the SMP system for 64 requests (Table 6, 7). We show that WWW server had a higher lock ratio (63.6%) when the client requested 150KB files. This is because that WWW server invokes system calls frequently since it reads (by `read` system call or `mmap` system call) requested files and sends (by `writev` system call) them through the network.

WWW server has too high waiting ratio (63.6%) caused by too high lock ratio (35.9%) with 150KB file. To have high lock ratio is evidently one of the reasons of the performance degradation of WWW server because the idle ratio is low with 150KB file.

In next, we show experimental results on the SMP system with 4 CPUs (Fig. 3, Table 8 and 9). This program did not perform well on the SMP system with 4 CPUs, e.g., the maximum speedup was 1.04. However, each CPU often became idle because CPUs (500MHz PentiumIII Xeon processors) were faster than the network interface. In case of the WWW server on 4 CPUs, the result may not be expressed in our model.

4.4 Discussion

Our experimental results (Table 10) show the following.

1. The parallel programs which have the lower lock ratios, such as the matrix multiplication programs, get note of an effect from parallel processing.

2. The parallel programs which have higher lock ratio than above, such as parallel make, degrades its performance than above.

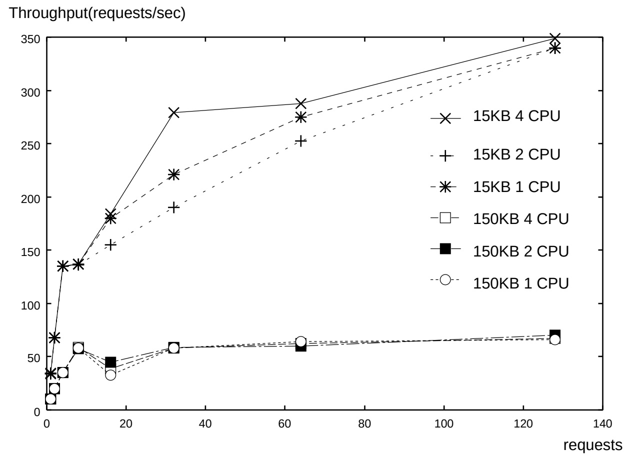

Figure 3: Throughput for the Apache 1.3.9 on the SMP system with 4 CPUs.

Table 8: Experimental results on the SMP system with 4 CPUs (WWW server, 64 requests, file size 15KB).

number of CPUs	execution time (sec)	speedup	lock ratio (%)	waiting ratio (%)
1 CPU	0.23	1.00	48.0	–
1 CPU SMP	0.23	1.00	–	–
2 CPU	0.25	0.92	–	7.86
3 CPU	0.25	0.92	–	7.10
4 CPU	0.22	1.04	–	35.2

Table 9: Experimental results on the SMP system with 4 CPUs (WWW server, 64 requests, file size 150KB).

number of CPUs	execution time (sec)	speedup	lock ratio (%)	waiting ratio (%)
1 CPU	1.00	1.00	62.8	–
1 CPU SMP	1.03	0.97	–	–
2 CPU	1.07	0.93	–	7.27
3 CPU	1.06	0.94	–	4.86
4 CPU	1.03	0.97	–	20.7

3. The performance of parallel programs which have high lock ratios (e.g., WWW server programs) actually deteriorates when they are run on SMP systems. These programs therefore get no advantage from parallel processing on SMP systems because the processes constituting those programs are excluded and serialized by the kernel.

   The speedup for a WWW server is less than 1 when its lock ratio is more than 49.0%.

The degradation of the performance of parallel programs increases with the kernel-level lock ratio. It is therefore important to analyze the lock ratios.

Table 10: The causal relation between speedup and the lock ratio.

SMP systems		matrix multiplication	parallel make	WWW server
2 CPUs	lock ratio (%)	less than 0.1	5.91	63.6
	speedup	1.95	1.81	0.96
4 CPUs	lock ratio (%)	less than 0.1	4.08	62.8
	speedup	3.96	3.48	0.97

5 Conclusions

In this paper we have reported our use of case studies to evaluate the performance degradation caused by the kernel-level lock. We have defined the lock ratio as the ratio of the execution time for critical sections of a process to the total execution time. We have explained the causal relation between the performance of parallel programs on the SMP systems and their lock ratio as measured in experiments on Linux 2.0. We have measured lock ratios for and the performance of three types of parallel programs on two types of SMP systems (with 1 to 4 CPUs). As a result, we have shown the parallel program which have lower lock ratios perform better when executed in parallel and that the performance of parallel program which has higher lock ratios is worse on SMP systems than on uniprocessor systems. The analysis of kernel-level lock is therefore very important because this lock is one of the reasons for the performance degradation on SMP systems.

We think future work should include the following.

1. Measurements on the operating systems which have the fine grain of lock operations.
2. Measurements on a system that has extended I/O subsystems (e.g., the RAID controller).
3. Comparison of the results in this paper and the performance prediction model.

References

[1] G. Amdahl, "Validity of the Single-processor Approach to Achieving Large Scale Computing Capabilities", Proc. AFIPS Conference, pp. 483-485, Apr. 1967.

[2] Apache Software Foundation, `http://www.apache.org/`, June 1999.

[3] J.B. Chen, Y. Endo, K. Chen, D. Mazières, A. Dias, M. Seltzer and M.D. Smith, "The Measured Performance of Personal Computer Operating Systems", ACM Trans. on Computer Systems, vol. 14, no. 1, pp. 3-40, Feb. 1996.

[4] J. Gustafson, "Reevaluating Amdahl's Law", *CACM*, vol. 31, no. 5, pp. 532-533, May 1988.

[5] T. Horikawa, "TinyTOPAZ: A Hybrid Event Tracer for Unix Servers", Proc. of the 1999 Symposium on Performance Evaluation of Computer and Telecommunication Systems, pp. 203-210, July 1999.

[6] T. Kurasugi and I. Kino, "Approximation Methods for Two-layer Queueing Networks", Performance Evaluation, vol. 36-37, pp. 55-70, Aug 1999.

[7] K. Lai and M. Baker, "A Performance Comparison of UNIX Operating Systems on the Pentium", Proc. of the USENIX 1996 Technical Conference, pp. 265-278, Jan. 1996.

[8] X. Leroy, "Linuxthreads - POSIX 1003.1c kernel threads for Linux", http://pauillac.inria.fr/~xleroy/linuxthreads, 1996.

[9] Netcraft, "The Netcraft Web Server Survey", `http://www.netcraft.com/survey/`, Aug. 1999.

[10] X.-H. Sun and D.T. Rover, "Scalability of Parallel Algorithm-Machine Combinations", IEEE Trans. on Parallel and Distributed Systems, vol. 5, no. 6, pp. 599-613, June 1994.