

Evaluation and Enhancement of Linux SMP subsystem Proposal

Bo Hu
Deqing Chen

October 19, 1999

1 Introduction

Which is better, Linux or Windows NT? The Sound and the Fury.

We might not have an answer for this question. But we can come out the answer for a more specific question: which is better, Linux SMP or NT SMP?

There are some answers out there. By doing a web searching, we can get all kind of answers. Most of them are done by individuals. Some of them are quite emotional(Linux fans tend to humiliate Microsoft's products, especially NT server[Sou].) [Mic] gives a detailed summary of comparisons between MS Windows NT Server 4.0 and Linux made by various sources. It reaches the conclusion that NT outperforms Linux. Meanwhile, reports from newsgroups, personal websites keep coming out showing Linux is better than NT in SMP aspect.

Strangely enough, though the dispute between Linux and NT has been undergoing for a while, we can hardly find journal or proceeding papers on this issue. This arouses our interests to make a more thorough investigation into this problem. The purpose of this project is:

1. Make things clear. Though the comparisons in [Mic] are in great details, they are from the application view or from outside the operating systems, which makes the results less convincible. We want to make comparisons from a technique perspective or from inside the operating systems. We want to find out which factors behind the scene, like scheduling, memory management,etc., bring performance differences.

2. Find some hints to improve Linux's SMP performance. Linux supports SMP only after version 2.0, while NT is inborn to support SMP. Furthermore, because of the nature of Linux, the development of SMP subsystem in Linux is not as well organized as that in NT. There must be some problems in Linux SMP. Through comparing with NT, we may find some way to improve performance of Linux SMP subsystem.

In this project, we will run different benchmarks on both of the platforms. We will try to explain the results by scrutinize the source code. We hope an unbiased conclusion will be drawn out.

2 Approach

2.1 Angles

Though Linux has been SMPable for some time, unfortunately we haven't found any detailed description about Linux SMP subsystem. But we can find some materials about SMP implementations of other modern operating systems, such as Windows NT [Sol98] and Digital Unix [jMDaAW94]. We also have Linux source code, and fortunately, NT's source code. By reading these literatures and analyzing source codes, we can find several possible angles that worth looking at, for example,

- Kernel Preemption. On a uniprocessor machine, allowing kernel preemption does not make much sense. But for a multiprocessor machine, kernel preemption could make kernel have more parallelism, which in turn will increase kernel's efficiency and improve CPU utilization. Kernel preemption is allowed on NT and not allowed on Linux.
- Thread Architecture. The "One for One" design policy of Linux kernel essentially results in a heavier thread architecture. NT's thread system adopts the approach much like Mach. We will investigate whether this will influence Linux SMP behavior.
- Memory management. NT has some SMP specific optimization in memory management subsystem. For example, on a multiprocessor machine, NT uses some bits of page table entry differently from uniprocessor ones to avoid TLB flushing and shutdown. Whether Linux also has similar optimizations and how its behavior could be an interesting point.

2.2 Benchmark Application

We will use Uranus as our test bed. Uranus has four PentiumPro 200Hz CPUs and can be dual booted for both NT and Linux. It makes an ideal platform for comparing Linux and NT. We will use some benchmark applications to get performance data for both Windows NT and Linux. To do this, we may need to modify the sources to get our critical statics.

The benchmark applications we are going to use will include:

- Some hand-coded programs. These will be simple programs that stress on some part of system features. For example, we will code some programs that is:
 - computation loaded. The application will mostly run integer and floating arithmetic in its life time.
 - memory loaded. The application will allocate large chunk of memories for its computation.
 - kernel service loaded. The application will constantly require kernel service.
 - I/O bound. The application will do lots of I/O works.
- Standard Parallel Benchmark, FFT, Barnes and LU. These applications could give us a more comprehensive benchmark results. These applications are chosen for several reasons. First we have source code for these applications, both sequential and parallel versions. We need only little port work to get them running on Linux and NT. They are also fairly large programs and can exhibit reasonable load for Uranus.

3 Schedule

Oct. 15 - Oct. 26:

- Get Linux and NT SMP source compiled and running.
- Reading SMP source code.

Oct. 27 - Nov. 7:

- Get benchmark applications running.
- Get performance datas.

Nov. 8 - Dec. 2:

- Preliminary data analyses.
- System enhancement.
- Final data analyses.

Dec. 2 - Dec. 10: Prepare report.

References

- [jMDaAW94] Paula Long jeffrey M. Denham and ames A. Woodward. Dec osf/1 symmetric multiprocessing. *Digital Technical Journal*, 6(3), 1994.
- [Mic] Microsoft. Industry benchmarks show windows nt server 4.0 outperforms linux. <http://www.microsoft.com/ntserver/nts/exec/compares/ntlunix.asp>.
- [Sol98] David A. Solomon. *Inside Windows NT*. Microsoft Press, second edition edition, 1998.
- [Sou] Open Source. Halloween documents. <http://www.opensource.org/halloween/>.