Loki: Commodity Parallel Processing in Practice

Michael S. Warren

Theoretical Astrophysics (T-6)
Los Alamos National Laboratory

`http://loki-www.lanl.gov`

Motivation

- We need usable, reliable, abundant, affordable computing power
- Supercomputer vendors are going bankrupt, selling out, or giving up. (Cray Research, Cray Computer, Intel SSD, Convex, Thinking Machines)
- The success of supercomputing centers and the Internet means the amount of computing available to each individual research group has actually declined over the past few years.
- Good software is incredibly expensive to develop. We can't afford to pay for useless software hidden in the hardware price. I consider things like HPF and parallel debuggers with a graphical user interface to be useless.
- The rate of change in computing is beyond the ability of big organizations to keep pace with.

Spot Prices – April 7, 1997

Item	Description	Price ($)
ASUS P/I-XP6NP5	motherboard	269
Pentium Pro	200 MHz, 256k L2	547
Pentium Pro	150 MHz, 256k L2	218
SIMM	FPM 8x36x60, 32 Mbyte	190
Disk	Quantum Tempest 3.2GB EIDE	285
Fast Ethernet	Kingston/DEC 21140 PCI	89
Misc.	Case, Floppy, Heat Sink	200
BayStack 350	16 port 10/100 Mbit switch	2564

Table 1: These were obtained from Atipa www.atipa.com, and Sparco www.sparco.com.

A 16 processor 200Mhz-128Mb-2.3Gb system with BayStack switch would be $38k. A 16-way system with 150 Mhz processors, with half the memory and disk would be $25k.

Enabling Technologies

How is it possible that one or two people can design and build a computer in a few weeks that is superior to machines which have tens of millions of dollars in research and development behind them?

- Linux - Hardware independent operating system
- PCI - Hardware independent bus
- Dramatic increase in microprocessor performance (Moore's Law). The latest personal computer microprocessor (Pentium Pro) is about 1/3 the speed of the fastest microprocessor (DEC Alpha or R10000).
- Fast ethernet (cards are now around $80)

Disabling Technologies

- Spending more than 10 minutes per week talking to "customer support." Giant corporations are making you beta test their broken software.

- Anything advertised as "making parallel programming easy." I don't need it. Neither do the people who will waste their time wondering why it doesn't work as advertised. I would rather have attention focused on making the C and Fortran 77 compilers faster, or writing a user-space network driver.

- Anything proprietary. I know it will be broken when I try to use it, so the documentation and source code better be available so I can fix it.

- Shared Memory.

Loki

Benefits

- Commodity components are inexpensive, easy to obtain, easy to maintain, easy to upgrade, and come with good warranties.

- A computer you build yourself to suit your problems can span a much wider range in architecture options.

- A computer that is under your total control can solve your problems in a manner which you choose. If it breaks, you know how to fix it. If it doesn't need to be upgraded with the latest software, you don't have to grit you teeth when "preventive maintenance" happens at just the wrong time.

Empowering the Power User

We are from the Computer Science Department, and we're here to help you...

- It is much easier to make a system work well for a single problem, than to be all things to all people. The greatest advantage I have is a problem that I need to solve. All advances are measured with respect to how they help to solve that problem.
- I believe that people who have a desire to understand the whole of a computer system will always achieve better results than those who wish to remain ignorant of the details.
- The people who write software that actually solve problems in the real world must strive to be self-sufficent. Those who write software dealing with computer-science issues seem to always be solving the wrong problems.

Site	Machine	Procs	Time	Gflops	Mflops/proc
Sandia	ASCI Red	1408	27.82	96.53	68.5
LANL	TMC CM-5	512	140.7	14.06	27.5
Caltech	Intel Paragon	512	144.4	13.70	26.8
NRL	TMC CM-5E	256	171.0	11.57	45.2
Caltech	Intel Delta	512	199.3	10.02	19.6
NAS	IBM SP-2	128	281.9	9.52	74.4
JPL	Cray T3D	256	338.0	7.94	31.0
LANL	CM-5 no vu	256	754.6	2.62	5.1
LANL	SGI Origin 2k	24	394.2	5.02	209
SC ’96	Loki+Hyglac	32	1218	2.19	68.4
LANL	Loki	16	2102	1.28	80.0

Table 2: Treecode performance.

Loki

NAS Parallel Benchmarks

	Class	Procs	Loki-GNU	eff.	SGI Origin
bt	A	1	21.0		67.6
sp		1	17.1		51.5
lu		1	28.5		86.9
mg		1	14.8		72.9
is		1	2.6		2.3
bt	B	16	329.7	98	925.5
sp		16	222.9	81	957.0
lu		16	388.1	85	1317.4
mg		16	219.1	92	1039.6
is		16	14.8	35	33.9

Table 3: Single processor performance (Mops) for Class A NPB 2.2 benchmarks. Data from Loki with GNU compilers, and an SGI Origin 2000 are presented.

Price/Performance

	Loki	Origin	SP-2	Alpha
Price, Nov. 1996	$55k	$960k	$3520k	$580k
Number of Procs	16	26	64	8
NPB 2.2 B Time	5049	957	304	
NPB Price/Perf	1.0	3.3	3.8	
SPECint Price/Perf	1.0	7.0	16.0	9.5
SPECfp Price/Perf	1.0	2.6	4.2	5.8
Stream Price/Perf	1.0	5.3	1.8	12.0

Table 4: Loki price/performance vs other NAS Class B capable machines: SGI Origin 2000, IBM SP-2 P2SC, DEC AlphaServer 8400/440. SPEC comparison uses the result from Loki using gcc and g77.

parameter	Loki	4 CPU Origin 200, 2 Gb mem
Price	\$50k	\$140k (includes 30% discount)
Performance	250-1200 Mflops	250-800 Mflops
Memory price	\$5/Mbyte	\$40/Mbyte
Delivery time	1 week	was 3 months, now 1 month
Warranty	1 year to lifetime	90 days
HW Maint.		9% of initial cost per yr
Software		\$600 + \$1200/yr (single seat)

Table 5: Loki vs the SGI Origin 200

Architecture	*SPECfp95*	*Cost ($)*
Intel Pentium 133	3.12	1090
Intel Pentium Pro 200	6.75	1750
DEC Alpha 21164-433	17.0	3325
SGI Origin 200	15.6	10300

Table 6: SPEC floating point benchmark performance and cost (with discounts applied) for minimal systems (CPU, 64 Mb of memory, 2 Gbyte disk and 100 BaseTx network interface.

Benchmarks	Loki-GNU	Alder
099.go	7.72	8.11
124.m88ksim	5.25	7.81
126.gcc	5.87	7.65
129.compress	5.06	6.99
130.li	6.24	8.62
132.ijpeg	6.12	8.43
134.perl	7.66	8.21
147.vortex	5.99	9.14
SPECint95	6.17	8.09

Table 7: SPEC Benchmark CINT95 Summary.

Benchmarks	Loki-GNU	Alder
101.tomcatv	6.67	11.6
102.swim	10.2	16.3
103.su2cor	2.25	3.80
104.hydro2d	2.48	4.08
107.mgrid	3.77	4.11
110.applu	3.51	4.82
125.turb3d	3.93	6.34
141.apsi	4.20	6.74
145.fpppp	9.62	10.5
146.wave5	5.69	7.52
SPECfp95	4.63	6.75

Table 8: SPEC Benchmark CFP95 Summary.

Hardware

- Nothing can beat Fast Ethernet for price/performance at the moment
- DEC 21140 (Tulip) Fast Ethernet cards are excellent performers
- You want as much memory bandwidth as possible
- You can never have too much memory. It's cheap as dirt, anyway.
- SMP is a terrible idea

Qty.	*Description*
1	Intel Pentium Pro 200 Mhz CPU with 256k L2 cache
1	Intel VS440FX board with 82440FX (Natoma) chipset
4	8x36 60ns parity SIMMS (128 Mb per node)
1	Quantum Fireball 3240 Mbyte IDE Hard Drive
1	Cogent EM400 TX PCI Quartet Fast Ethernet Adapter
1	SMC EtherPower 10/100 Fast Ethernet PCI Card
1	S3 Trio-64 1Mb PCI Video Card

Table 9: Loki node architecture.

Front End

Qty.	*Description*
2	Intel Pentium Pro 200 Mhz CPU with 256k L2 cache
1	ASUS P/I-P65UP5 dual cpu board with Natoma chipset
8	16x36 60ns parity SIMMS (512 Mb)
6	Quantum Atlas 4.3 Gbyte Ultra SCSI Hard Drive
1	Adaptec 2940UW PCI Fast Wide SCSI Controller
1	Cogent EM400 TX PCI Quartet Fast Ethernet Adapter
1	SMC EtherPower 10/100 Fast Ethernet PCI Card
1	Matrox Millenium 4 Mb PCI Video Card
1	21” Nokia 445X Monitor
1	Northgate Keyboard, Logitech Mouse, Floppy Drive
1	Toshiba 8x IDE CD-ROM
1	HP C1533A DAT DDS-2 4 Gbyte tape drive
1	Quantum DLT 2000XT 15 Gbyte tape drive

Table 10: Loki front-end architecture.

Qty.	*Description*
2	3Com Linkswitch 3000 8-port 100Base-TX
1	APC Smart-UPS 2200
500ft	Cat-5 4pr stranded plenum cable
200	RJ-45 Cat-5 heads
1	RJ-45 crimper
2	4-way Keyboard/Monitor switch
8	Keyboard/Monitor Extension Cables
2	GEM 14” SVGA monitors
	Keyboards, Floppy Drives, Mice, Tapes
	Tool Kit, Power Strips
	X-inside, Mathematica

Table 11: Other hardware and software items.

Why SMP is a terrible idea

- We're already short on memory bandwidth and network bandwidth. Multiple CPUs make the problems even worse.
- If you are going to write a message-passing program, why do you want to have to deal with shared memory on top of that?
- Multiple processors sharing memory makes it a lot harder to write a stable and efficient operating system.

Memory and Disk benchmarks

	Bandwidth
Disk	4.2
Stream Copy	79
Stream Scale	79
Stream Add	87
Stream Triad	87
lmbench Mem Read	164
lmbench Mem Write	65

Table 12: Memory and Disk micro benchmarks. Bandwidth is reported in Mbytes/sec.

Reliability

- All Loki nodes were up for a two month period without a single minute of downtime. There have been 0 unexpected node crashes. There have been 0 pieces of hardware replaced since the initial burn-in.

- During burn-in, 1 bad SIMM and 7 bad Quantum EIDE disks were replaced under warranty. We have found that compiling gcc with itself is a very good memory and cache tester.

- During the round-trip from Los Alamos to Pittsburgh for Supercomputing '97, the only hardware problems encountered were 1 bad ethernet cable on the way out, and 3 bad ethernet cables after we came back. I attibute this to my poor wire-crimping skills (although 4 out of 160 cables isn't bad).

Software

The following pieces of software are critical. They must never fail. They are more important than any other components of the system.

1. The operating system.
2. The compiler.

The following pieces of software are very important.

1. The message passing library.
2. The debugger.
3. The profiling and performance monitoring system.

Loki

Message bandwith

- Multiple fast ethernet ports saturate at 20 Mbytes/sec overall. Our hardware (5 ports) should be capbable of 50 Mbytes/sec overall.

- The reason appears to be that memory bandwidth limits message bandwidth through the kernel to about 20 Mbytes/sec due to copies to user space.

- Hacked UNet without user space copy shows bus is capable of 40 Mbytes/sec, at least.

- Doing user space messages while avoiding the receive copy is tricky and will require some excellent software.

Loki

Network micro benchmarks

Version	Bandwidth	Latency	Latency w/switch
SWAMPI	11.7	208	238
MPICH 1.0.13	3.2		503
MPICH 1.1.0	8.8		390
LAM 6.1	7.3		2690
LAM 6.1 -c2c	11.6		322
TCP socket	11.7	158	182
UDP socket	11.7	131	153
U/net	12.3	55	

Table 13: Comparison various message passing protocols.

SWAMPI, MPICH 1.0.13, TCP, UDP and U/Net numbers are with Linux 2.1.29, others with 2.0.29. Bandwidth is reported in Mbytes/sec. Latencies are in microseconds.

MPI versions in application benchmarks

	Class	Procs	Loki-SWAMPI	Loki-LAM	Loki-MPICH
bt	B	16	329.7	306.2	316.0
sp		16	222.9	178.9	190.4
lu		16	388.1	347.2	389.5
mg		16	219.1	193.2	209.5
is		16	14.8	6.0	–

Table 14: Comparison of Salmon-Warren MPI (SWAMPI), LAM 6.1, and MPICH 1.0.13 on the Class B NAS 2.2 benchmarks.

The MPMY Interface

These functions are the only ones which are fundamentally required for message passing programming. Our treecode uses the following interface on top of whatever works best (MPI, PVM, TCP sockets, UDP sockets, shared memory, NX, CMMD, EUI)

- void Init(int *argcp, char *** argvp);
- int Isend(const void *buf, int cnt, int dest, int tag, Comm_request *reqp);
- int Irecv(void *buf, int cnt, int src, int tag, Comm_request *reqp);
- int Test(Comm_request req, int *flag, Status *stat);
- int Nproc(void);
- int Procnum(void);

Debugging

- I am sick and tired of what Cray, Intel and others pass off as parallel debuggers. They are practically useless. I am forced to wonder how they manage to do any parallel code development themselves without a usable debugger.

- gdb "attach" works very well to determine the state of a single process.

- Tracing program flow and correlating events on different processors requires a vast amount of precise information which is not readily displayed in a graphical debugging tool. We use a library called "Msgs", which is basically what you end up with when you force "printf" to be a debugging tool.

Msgs

Msgs provide a primitive message logging facility. Messages can be turned on for particular files or on particular processors.

```
#include "Msgs.h"

Msg_turnon("main.c:0-3");

Msg_do("Data read, nobj=%d, gnobj=%d\n", nobj, gnobj);

Msgf(("%d interactions, pos=%g %g %g\n",
                                    end-pp, ppos0, ppos1, ppos2));
```

A 200 Mhz Pentium Pro is capable of 200 Mflops doing adds or balanced adds and multiples, but only 100 Mflops doing purely multiplies.

Operation	Throughput (clocks)	Latency (clocks)
add	1	3
mul	2	5
div	8	8
sqrt	9	9
sin	84	84

Table 15: Floating Point Performance

Optimization Hints

- Don't waste your time looking for 10% improvements. Make sure that you aren't losing factors of two.
- Double precision floating point quantities should always be 8-byte aligned. This can make more than a factor of two difference. Use -malign-double with gcc and g77 to assure this. You also need an up-to-date Linux libc (5.4.23 or newer) to guarantee the stack pointer is initially 8-byte aligned.
- Unrolling is useless, unless you also eliminate dependencies which cause adder or multiplier pipeline stalls.

Unrolling to Eliminate Stalls

```
for (i = 0; i < n; i += 3) {
    sum += x[i];      /* This doesn't help */
    sum += x[i+1];    /* The adder stalls waiting for */
    sum += x[i+2];    /* the result to arrive in sum */
}
return sum;

for (i = 0; i < n; i += 3) {
    s0 += x[i];       /* This is the right way */
    s1 += x[i+1];
    s2 += x[i+2];
}
return s0+s1+s2;
```

Hardware Performance Monitoring and Profiling

We have written a couple of utilities which we have found to be very useful.

- perfmon
- patches for gcc function profiling

Perfmon results for SPECfp95

Benchmarks	Mflops	Bus Trans
101.tomcatv	33.7	2.50
102.swim	23.4	1.46
103.su2cor	56.3	.012
104.hydro2d	13.9	3.06
107.mgrid	44.3	2.05
110.applu	16.9	2.88
125.turb3d	35.5	2.69
141.apsi	22.5	2.95
145.fpppp	25.1	0.71
146.wave5	17.8	1.22

Table 16: Mflops and BUS_TRAN_MEM for the 10 SPECfp95 benchmarks. The maximum value for BUS_TRAN_MEM is 3.6.

Reduced g77 MSR profile data for mgrid

Function	seconds	Mflops	Bus Trans
MAIN	663.3	44.63	53.0
mg3p	486.9	43.51	52.9
psinv	187.7	49.79	40.2
resid	363.5	48.78	51.9
rprj3	36.7	31.07	77.5
interp	67.8	19.77	75.5
comm3	29.5	4.80	101.2
norm2u3	1.6	25.45	59.7
zero3	4.8	0.39	115.3

Table 17: gcc function profile of SPEC95fp mgrid benchmark. Memory bus transactions are expressed as a percentage fraction of maximum.

Perfmon Time History: Treecode

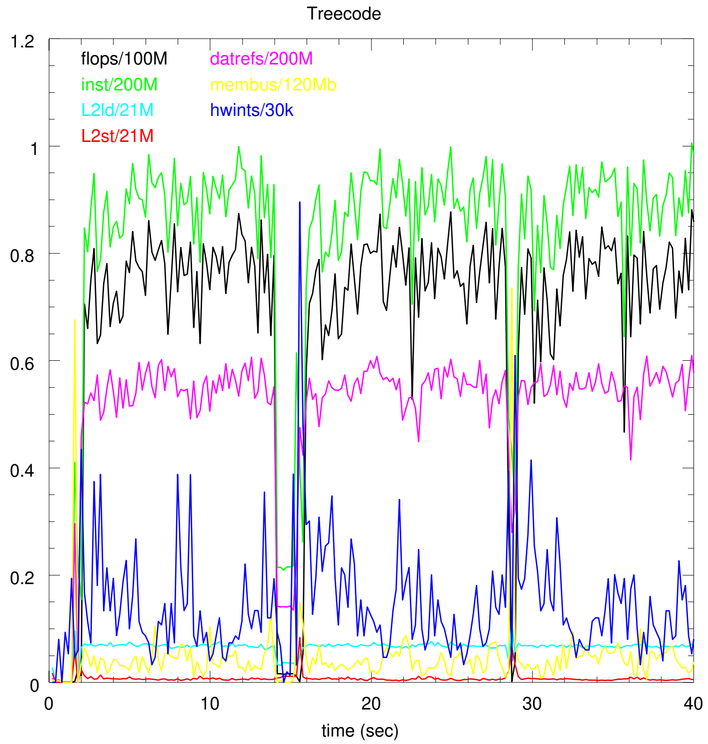

Figure 1: Multiplexed performance monitoring vs time for the treecode.

Linux Performance Monitoring

Linux also maintains a great deal of relevant performance information in the /proc filesystem. (CPU, memory used, swap used, network utilization)

An array of xosview windows is a cheap and easy way to get good idea of the machine state across multiple nodes.

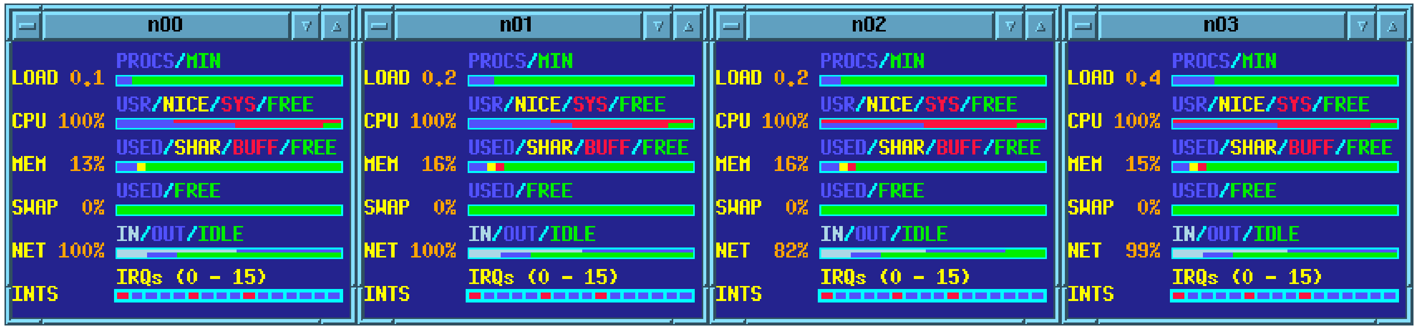

Figure 2: xosview demonstrating the current state of four different processors.

Things to Remember

- Loki is a real scientific computing resource, built by physicists to solve actual physics problems.
- Loki has superior price/performance for the majority of message-passing scientific application codes compared to all current commercial machines.
- In current systems, memory bandwidth is the single most important factor which limits scientific application performance.
- Solving an application problem is the final arbiter of success.