The Impact of Operating Systems on the Execution of Parallel Code

1 Executive Summary

Until recently, personal computers have had a single processor. Now, with multi-core processors quickly becoming the new standard, programs need to be written to take advantage of this new technology. The purpose of this project was to investigate the impact of the operating system on the execution of parallel code and determine which operating systems allow parallel code to operate most efficiently.

My hypothesis was that Gentoo Linux would perform the best for two reasons. First, the Linux kernel is lightweight and well written, and second, the Gentoo operating system is completely compiled from source and tailored to the specific process architecture upon which it is installed.

In my experiments I ran five parallel benchmarks on Windows XP, Gentoo Linux, Debian Linux, and FreeBSD on a computer with an Intel quad core processor, and used a Python script to time them. The results were then put into Microsoft Excel and analyzed.

The results showed that the Linux systems were equally efficient at executing parallel code and had the overall lowest execution times, confirming my hypothesis. The results from this project show that Linux is best suited to lead the way in the new era of parallel computing. Other interesting results were that while Windows XP had the slowest overall execution times, it had the best speedup ratios, suggesting that it does a better job at multi-threading. And even though Gentoo is specifically compiled for the processor architecture of the system it is installed on, and Debian is a generic pre-compiled system, they performed nearly identically.

Table of Contents

1 Executive Summary 1
2 Introduction 4
2.1 Background/motivation 4
2.2 Problem investigated 4
2.3 Research methodology 4
2.3.1 Experimental equipment 4
2.3.2 Data collection methodology 5
3 Results and Analysis 6
3.1 Results 6
3.2 Analysis of results 11
4 Conclusions and Future Work 12
5 References 13
Appendix A Raw Numerical Data A-1
A.1 Windows XP A-1
A.1.1 Collected Data A-1
A.1.2 Derived Data A-3
A.2 Gentoo A-4
A.2.1 Collected Data A-4
A.2.2 Derived Data A-6
A.3 Debian A-7
A.3.1 Collected Data A-7
A.3.2 Derived Data A-9
A.4 FreeBSD A-10
A.4.1 Collected Data A-10
A.4.2 Derived Data A-12
Appendix B Source Code B-1
B.1 Matrix-Vector Product Benchmark - C B-1
B.2 Benchmark Data Collector – Python B-3

List of Figures

Figure 1: Equation for Determining Speedup Ratio [1] 5
Figure 2: Table of Acronyms for Benchmarks 6
Figure 3: Logarithms of Average Times on Windows XP 7
Figure 4: Average Speedup Ratios on Windows XP 7
Figure 5: Logarithms of Average Times on Gentoo 8
Figure 6: Average Speedup Ratios on Gentoo 8
Figure 7: Logarithms of Average Times on Debian 9
Figure 8: Average Speedup Ratios on Debian 9
Figure 9: Logarithms of Average Times on FreeBSD 10
Figure 10: Average Speedup Ratios on FreeBSD 10
Figure 11: Standard Deviations of CG Benchmark in Seconds 11
Figure 12: Execution Times on Windows XP A-3
Figure 13: Average Times on Windows XP A-3

Figure 14: Logarithms of Average Times on Windows XP A-3
Figure 15: Standard Deviations of Times on Windows XP A-3
Figure 16: Average Speedup Ratios on Windows XP A-3
Figure 17: Standard Deviations of Speedup Ratios on Windows XP A-4
Figure 18: Execution Times on Gentoo A-6
Figure 19: Average Times on Gentoo A-6
Figure 20: Logarithms of Average Times on Gentoo A-6
Figure 21: Standard Deviations of Times on Gentoo A-6
Figure 22: Average Speedup Ratios on Gentoo A-6
Figure 23: Standard Deviations of Speedup Ratios on Gentoo A-7
Figure 24: Execution Times on Debian A-9
Figure 25: Average Times on Debian A-9
Figure 26: Logarithms of Average Times on Debian A-9
Figure 27: Standard Deviations of Times on Debian A-9
Figure 28: Average Speedup Ratios on Debian A-10
Figure 29: Standard Deviations of Speedup Ratios on Debian A-10
Figure 30: Execution Times for FreeBSD A-11
Figure 31: Average Times on FreeBSD A-12
Figure 32: Logarithms of Average Times on FreeBSD A-12
Figure 33: Standard Deviations of Times on FreeBSD A-12
Figure 34: Average Speedup Ratios on FreeBSD A-12
Figure 35: Standard Deviations of Speedup Ratios on FreeBSD A-12

2 Introduction

2.1 Background/motivation

Until recently, personal computers have had a single processor. Manufacturers have simply been increasing the processors clock speed to make them faster, but recently manufacturers have hit a wall in clock speeds do to cooling problems, and physical constraints.

To solve this problem, and continue the advancement of processors, processors now contain multiple cores, allowing them to run several threads at the same time in parallel. This can lead to a huge increase in speed, but only if programs are written to take advantage of it.

2.2 Problem investigated

Even if code is written to take full advantage of a parallel processor, the program will only be as good as the operating system it is run on. The goal of my project was to investigate the impact the operating system has on the execution of parallel code and to determine which OS allows parallel code to operate most efficiently. Thus, my work will which OS is best suited to lead the way in the new era of parallel computing.

My hypothesis was that Gentoo Linux would perform the best for two reasons. First, the Linux kernel is lightweight and well written, and second, the Gentoo operating system is completely compiled from source and tailored to the specific process architecture upon which it is installed. [1]

2.3 Research methodology

2.3.1 Experimental equipment

My experiments were run on a computer with an Intel Core 2 Quad Q6600 2.4 GHz processor and an ASUS P5N-D motherboard with an Nvidia nForce 750i chipset.

The operating systems tested were Gentoo Linux with kernel 2.6.26, Debian Linux 5.0 (Lenny) with kernel 2.6.26, PC-BSD 7.0 with kernel FreeBSD 7.1-PRERELEASE, and Microsoft Windows XP SP2 with the Cygwin UNIX compatibility layer.

Four of the benchmarks that were utilized were from the NASA Advanced Supercomputing Parallel Benchmarks [2], and the fifth benchmark was written by me using examples from the book *Using OpenMP* [3, p.37]. All of the benchmarks make use of the OpenMP framework for making their code parallel. All benchmarks were compiled with GCC 4.3.2, except for the FreeBSD benchmarks, which were compiled with GCC 4.3.3 due to technical constraints. This difference in compiler should not have affected the results because there were no performance-related changes between GCC 4.3.2 and GCC 4.3.3 [4].

The specific benchmarks used were:

- My Matrix-Vector Product (mvp) benchmark with matrix size 10,000x10,000
- NPB Integer Sort (IS) benchmark with data class C
- NPB Embarrassingly Parallel (EP) benchmark with data class A
- NPB Block-Tridiagonal (BT) CFD simulator with data class A
- NPB Conjugate Gradient (CG) random matrix generator with data class B

Each benchmark was intended to be run with 1, 2, 3, 4, 8, and 16 threads, for a total of 30 experiments. Unfortunately, the IS benchmark can only use a maximum number of 4 threads, and the IS benchmark will not run at all in FreeBSD. Thus, only 28 experiments had meaningful results on every OS besides FreeBSD. Of which 24 experiments produced meaningful results, because the IS benchmark would not run in FreeBSD.

2.3.2 Data collection methodology

Each benchmark was run on each OS (subject to the constraints mentioned in Section 2.3.1), and the execution times were collected by a Python script written by me. Before each set of benchmarks was run, all non-critical processes on the system were killed, except for a process manager to monitor the execution. For Windows this was the "Task Manager" process manager, and for the other systems it was the "top" utility. The Python script ran each benchmark 20 times, using 1, 2, 3, 4, 8, and 16 threads[1]. The script then collected the execution times and put them into a CSV file. The CSV file was then put into Microsoft Excel and used to calculate the average times, logarithmic times, speedup ratios, and standard deviations.

The speedup ratio is the ratio of how much faster each multi-threaded execution was than the initial single threaded execution, and it is used to measure the success of the parallelization. Ideally, the speedup ratio would be equal to the number of processors the program is run on. This can be expressed by the following equation, in which S represents the speedup ratio, T_1 represents the execution time on a single processor, and T_P represents the execution time on P processors.

$$S = T_1/T_P \text{ [3,p.33]}$$

Figure 1: Equation for Determining Speedup Ratio

[1] As noted previously, the IS benchmark could not be run across all the desired experimental parameters.

3 Results and Analysis

Section 3.1 presents the graphical representation of the raw data, and Section 3.2 presents an analysis of the data.

3.1 Results

The numerical raw data collected from the experiments can be found in Appendix A.

In the following graphs, the acronyms are interpreted in this manner:

mvp	Matrix-Vector Product
IS	Integer Sort
EP	Embarrassingly Parallel
BT	Block-Tridiagonal
CG	Conjugate Gradient

Figure 2: Table of Acronyms for Benchmarks

The graphs are grouped by operating system, with the logarithmic times first, followed by the speedup ratios.

In the graphs below, the IS benchmark data is incomplete due to its inability to run with more than four threads.

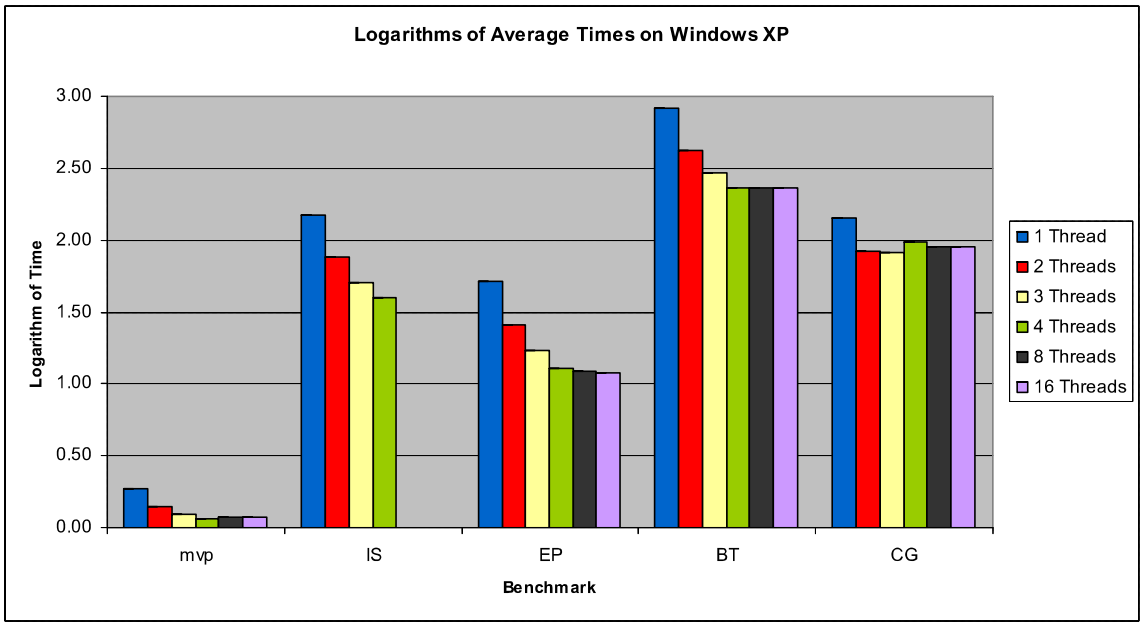

Figure 3: Logarithms of Average Times on Windows XP

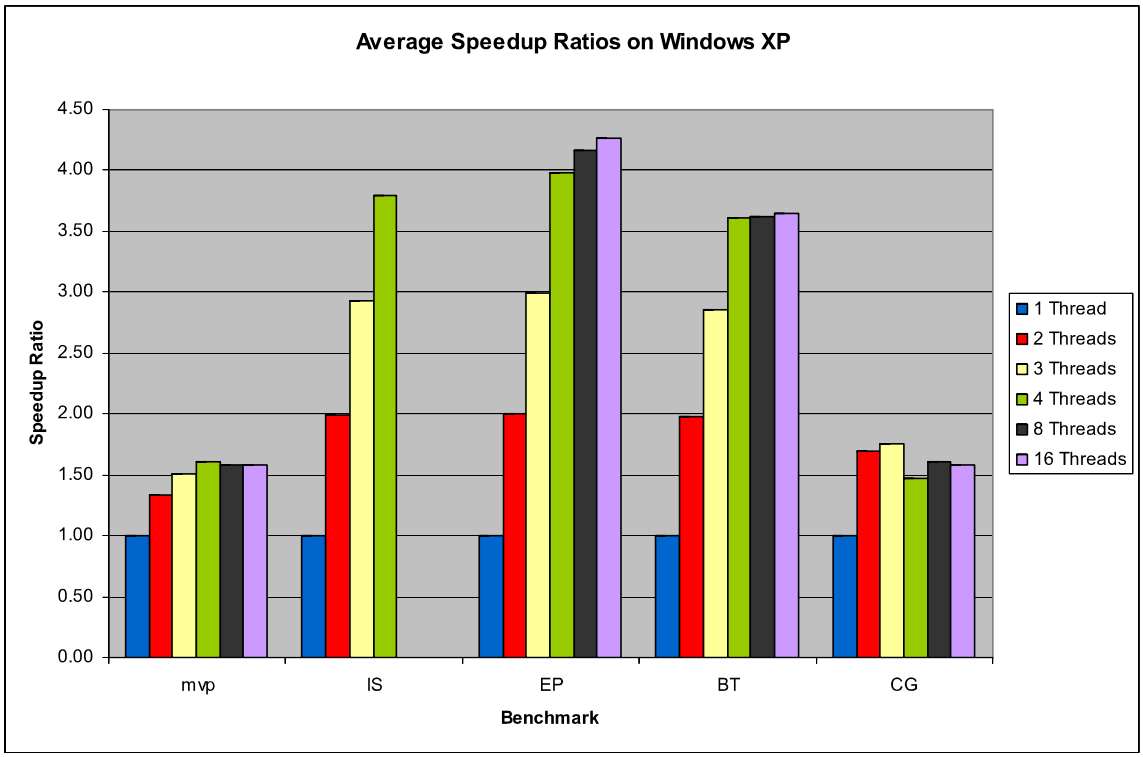

Figure 4: Average Speedup Ratios on Windows XP

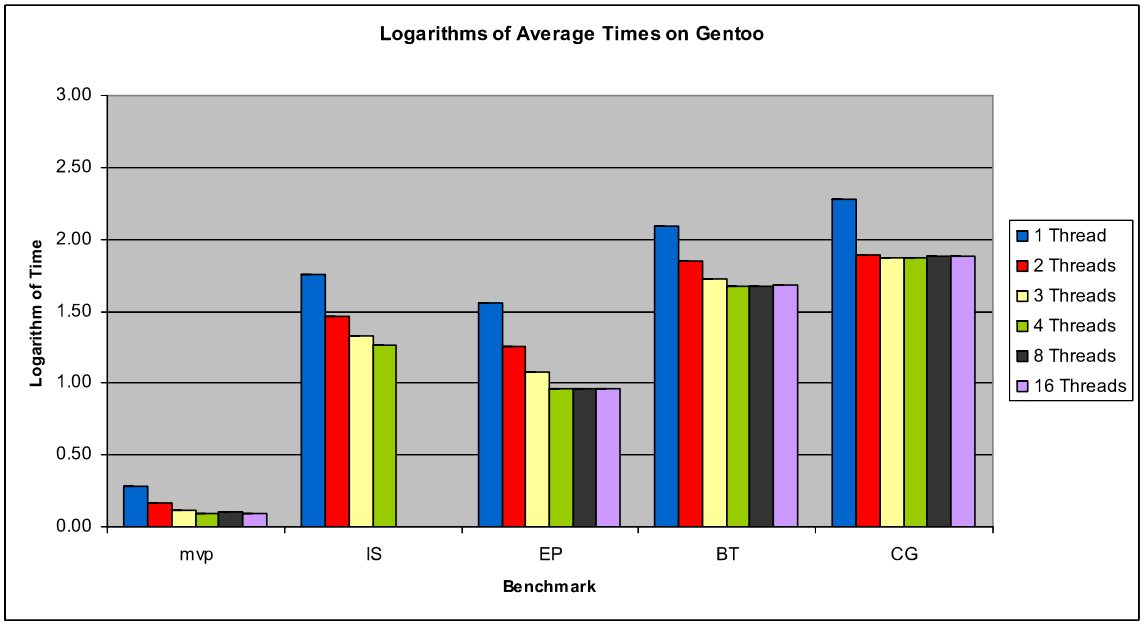

Figure 5: Logarithms of Average Times on Gentoo

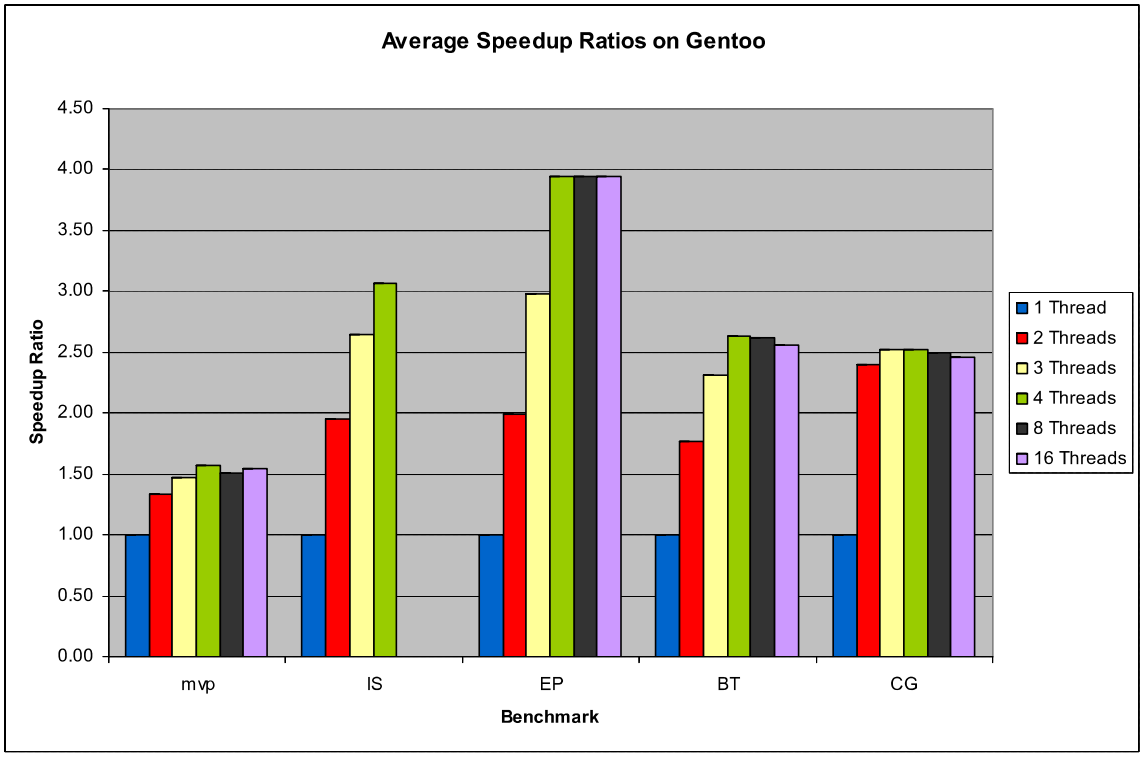

Figure 6: Average Speedup Ratios on Gentoo

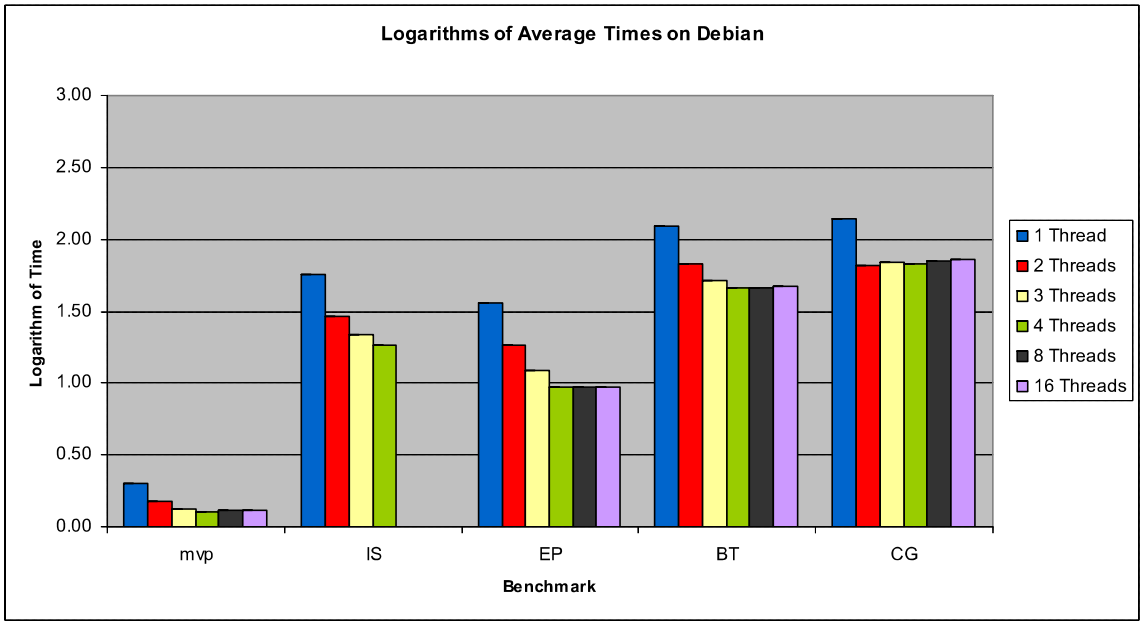

Figure 7: Logarithms of Average Times on Debian

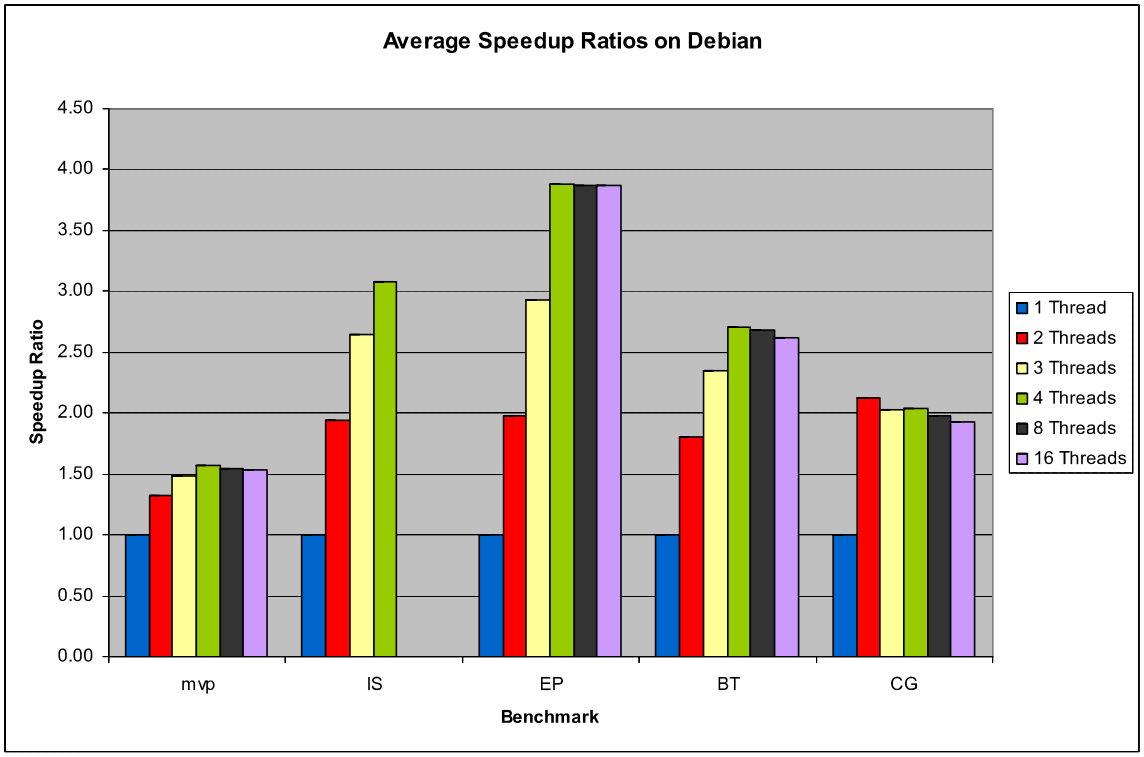

Figure 8: Average Speedup Ratios on Debian

I remind the reader that the IS benchmark would not run under FreeBSD.

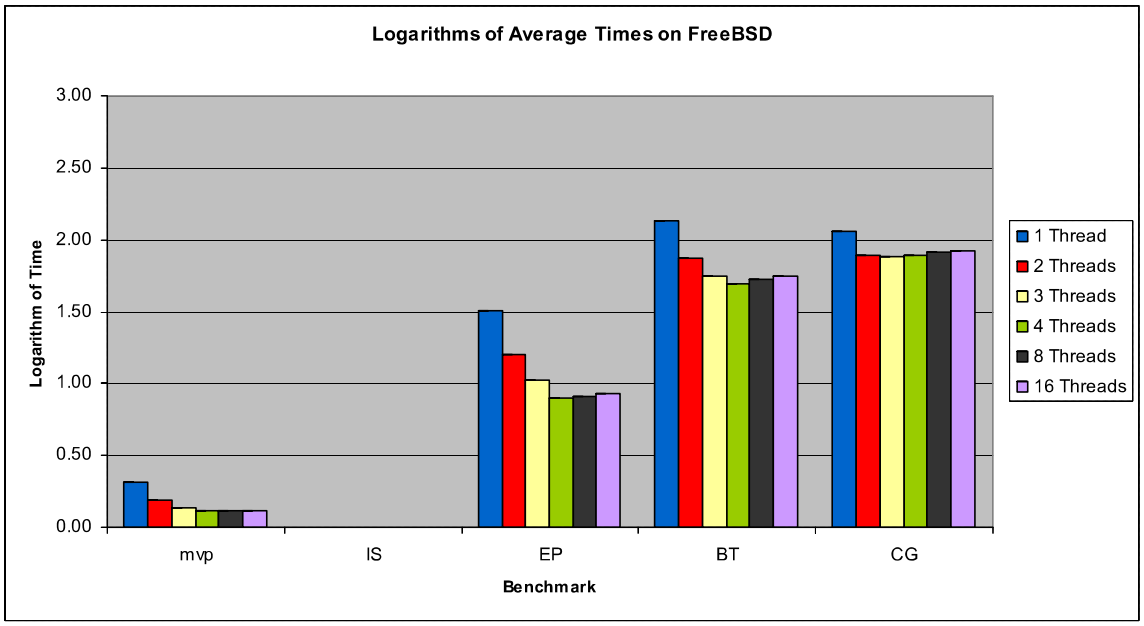

Figure 9: Logarithms of Average Times on FreeBSD

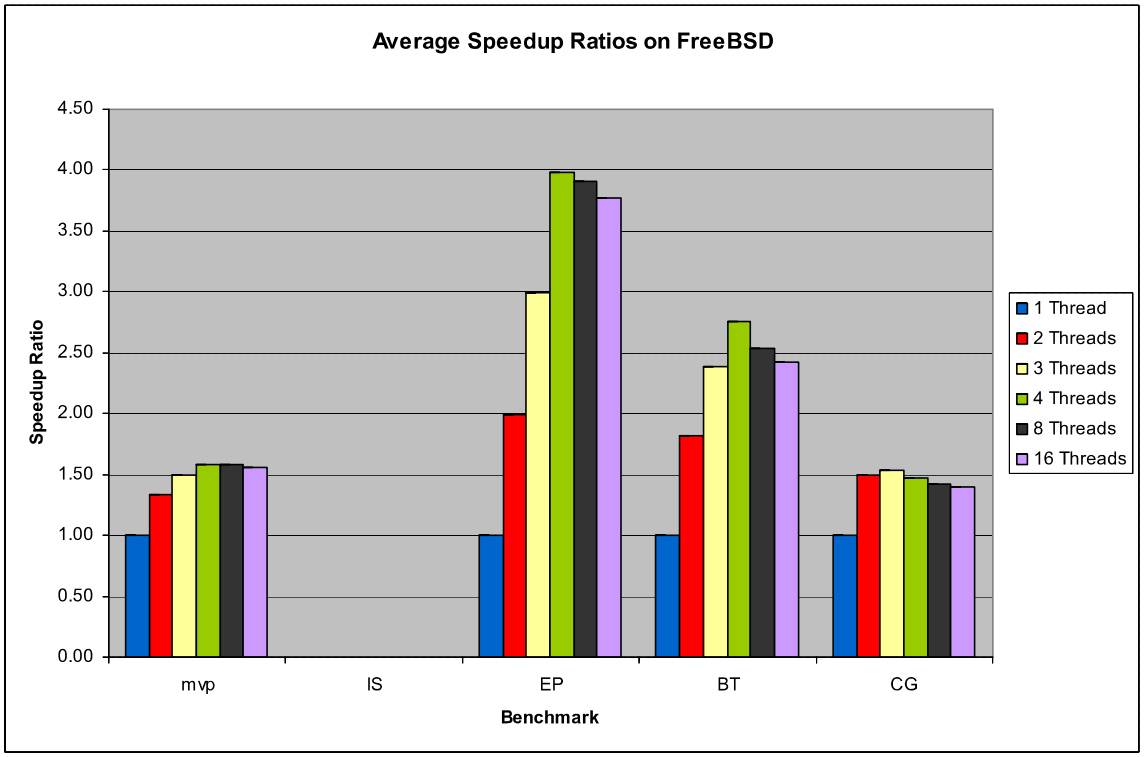

Figure 10: Average Speedup Ratios on FreeBSD

3.2 Analysis of results

The systems with the overall lowest execution times were the Linux systems, as predicted. What was unexpected was that Gentoo and Debian performed almost identically the same. This is surprising because Gentoo is custom tailored and compiled for the specific system it is installed on, which theoretically should make it faster whereas Debian is a pre-compiled generic system.

Another interesting result is that while Windows XP had the slowest overall execution times, it scaled up the best with increasing number of threads. Many of its benchmarks came very close to their ideal speedup ratios, as seen in Figure 4.

In the Linux systems, the performance of a benchmark more or less peaks after exceeding the number of physical cores on the system (4 in this case), which is to be expected. In Windows, the performance continues to increase slowly after exceeding the number of physical cores. This, along with Windows' superior speedup ratios, suggests that Windows does a very good job at multithreading. FreeBSD is quite the opposite, with its benchmarks' performance dropping noticeably after exceeding the number of physical cores, suggesting that there is a larger overhead associated with running multiple threads in FreeBSD.

Across most operating systems, the CG benchmark behaved very sporadically, as seen in the graphs, and in its somewhat high standard deviations, which are reproduced below.

	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
Windows XP	0.16	1.11	0.84	4.28	1.44	1.41
Gentoo	19.69	3.12	2.38	1.10	1.65	1.62
Debian	0.12	0.80	0.55	0.85	0.64	0.60
FreeBSD	0.44	2.15	1.42	0.83	0.68	1.04

Figure 11: Standard Deviations of CG Benchmark in Seconds

These relatively high standard deviations could be due to the benchmark's use of random data.

4 Conclusions and Future Work

Based on the results, both Gentoo and Debian Linux allowed the parallel code to operate the fastest, which confirmed my hypothesis that Gentoo Linux would be the fastest.

Future work on the topic could include:

- Researching additional operating systems.
- Determining which types of parallel code operate most efficiently.
- Studying parallel processing with more cores to test the limits of the operating systems.
- Ascertaining exactly how different operating systems handle parallel code to determine why they behaved the way they did.

5 References

[1] "About Gentoo". Gentoo Foundation. 2/15/09 <http://www.gentoo.org/main/en/about.xml>.

[2] "NAS Parallel Benchmarks". NASA. 2/15/09 <http://www.nas.nasa.gov/Resources/Software/npb.html>.

[3] Chapman, Barbara, Jost Gabriele, and Rudd Van Der Pas. *Using OpenMP*. Cambridge, MA: The MIT Press, 2007.

[4] "GCC 4.3.3 Change Log". Free Software Foundation. 3/7/09 <http://gcc.gnu.org/gcc-4.3/changes.html#4.3.3>.

Appendix A Raw Numerical Data

For each operating system, the original collected and all derived data is included below.

All times are in seconds.

A.1 Windows XP

A.1.1 Collected Data

Benchmark	Rep	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0	1.75	1.41	1.23	1.16	1.14	1.19
mvp	1	1.89	1.41	1.22	1.17	1.19	1.19
mvp	2	1.84	1.41	1.25	1.13	1.14	1.19
mvp	3	1.89	1.41	1.25	1.19	1.13	1.17
mvp	4	1.89	1.42	1.23	1.14	1.20	1.19
mvp	5	1.88	1.42	1.27	1.17	1.16	1.20
mvp	6	1.91	1.38	1.20	1.14	1.19	1.20
mvp	7	1.88	1.41	1.20	1.19	1.20	1.17
mvp	8	1.86	1.38	1.22	1.19	1.19	1.16
mvp	9	1.89	1.38	1.27	1.19	1.20	1.17
mvp	10	1.91	1.39	1.27	1.14	1.20	1.22
mvp	11	1.86	1.42	1.22	1.19	1.16	1.17
mvp	12	1.86	1.42	1.27	1.14	1.22	1.17
mvp	13	1.91	1.38	1.23	1.19	1.20	1.16
mvp	14	1.86	1.44	1.27	1.17	1.22	1.17
mvp	15	1.88	1.42	1.23	1.19	1.17	1.20
mvp	16	1.92	1.42	1.26	1.19	1.17	1.22
mvp	17	1.86	1.44	1.27	1.14	1.23	1.20
mvp	18	1.92	1.39	1.28	1.14	1.19	1.20
mvp	19	1.91	1.44	1.28	1.16	1.16	1.19
IS	0	150.33	75.16	51.27	39.58		
IS	1	150.25	75.31	51.27	39.59		
IS	2	150.23	75.45	51.86	39.64		
IS	3	150.25	75.41	51.25	39.51		
IS	4	150.19	75.42	51.25	39.50		
IS	5	150.22	75.44	51.23	39.58		
IS	6	150.27	75.39	51.25	39.70		
IS	7	150.31	75.47	51.27	39.52		
IS	8	150.31	75.34	51.28	39.59		
IS	9	150.25	75.44	51.23	39.61		
IS	10	150.19	75.36	51.25	39.45		
IS	11	150.27	75.30	51.33	39.61		
IS	12	150.23	75.31	51.28	39.61		
IS	13	150.25	75.31	51.27	39.61		
IS	14	150.25	75.45	51.24	39.63		
IS	15	150.23	75.33	49.75	39.56		
IS	16	150.23	75.38	51.28	39.56		
IS	17	150.23	75.33	51.39	39.66		

IS	18	150.20	75.45	51.84	39.53		
IS	19	150.30	75.44	49.70	39.41		
EP	0	51.30	25.66	17.13	12.88	12.25	12.05
EP	1	51.28	25.66	17.13	12.86	12.38	11.98
EP	2	51.27	25.66	17.13	12.86	12.34	12.03
EP	3	51.26	25.66	17.14	12.86	12.27	12.05
EP	4	51.28	25.66	17.14	12.86	12.30	12.02
EP	5	51.28	25.67	17.11	12.86	12.34	12.01
EP	6	51.25	25.64	17.13	12.84	12.31	12.05
EP	7	51.27	25.67	17.13	12.86	12.30	12.02
EP	8	51.27	25.66	17.13	12.86	12.33	12.00
EP	9	51.27	25.64	17.13	12.86	12.33	12.00
EP	10	51.27	25.64	17.11	12.88	12.33	12.06
EP	11	51.28	25.67	17.13	12.86	12.36	12.03
EP	12	51.28	25.64	17.13	12.86	12.33	12.05
EP	13	51.28	25.64	17.13	12.84	12.25	12.00
EP	14	51.25	25.63	17.14	12.86	12.31	12.05
EP	15	51.26	25.66	17.13	12.88	12.38	12.00
EP	16	51.27	25.66	17.11	12.88	12.31	12.03
EP	17	51.30	25.64	17.13	12.86	12.31	12.08
EP	18	51.28	25.66	17.11	12.88	12.33	12.03
EP	19	51.25	25.66	17.11	12.86	12.33	12.02
BT	0	834.30	419.95	293.59	231.38	230.42	227.92
BT	1	834.16	420.28	287.44	231.56	229.28	228.16
BT	2	834.16	420.36	301.25	230.81	229.75	229.59
BT	3	834.14	420.36	288.02	231.08	227.86	228.63
BT	4	834.39	422.31	287.67	231.13	228.86	229.47
BT	5	834.38	420.44	287.09	230.99	229.67	228.81
BT	6	834.30	420.31	287.39	231.30	230.33	228.22
BT	7	834.08	419.98	293.74	231.09	227.17	228.59
BT	8	834.17	420.13	293.78	231.17	228.97	228.30
BT	9	833.91	420.05	295.38	232.01	230.36	229.50
BT	10	833.84	420.23	302.27	230.13	231.25	228.69
BT	11	833.75	420.02	287.28	231.00	230.20	228.36
BT	12	833.69	420.19	302.53	230.92	230.95	228.41
BT	13	833.73	420.14	287.41	224.89	230.81	227.69
BT	14	833.64	420.02	287.33	230.88	231.27	229.50
BT	15	833.76	420.28	293.81	231.13	231.61	228.95
BT	16	833.64	420.38	287.42	230.88	232.03	227.97
BT	17	833.72	420.33	287.98	230.89	230.08	228.44
BT	18	833.77	420.34	302.59	229.78	231.05	228.95
BT	19	833.66	420.52	293.66	231.20	230.61	229.14
CG	0	143.61	82.64	79.00	96.48	91.13	91.05
CG	1	143.39	85.75	83.03	101.97	89.56	90.61
CG	2	143.00	84.03	81.81	102.66	88.09	90.22
CG	3	143.52	83.70	81.55	100.17	87.94	96.56
CG	4	143.83	84.48	81.33	101.48	89.55	90.98
CG	5	143.36	82.97	81.55	97.08	87.92	90.48
CG	6	143.55	85.42	82.08	94.86	88.27	89.70
CG	7	143.33	83.81	81.92	102.69	87.78	90.59

CG	8	143.44	84.66	82.78	93.73	88.03	91.25
CG	9	143.33	83.99	81.75	84.98	91.86	89.91
CG	10	143.61	85.00	81.88	98.63	87.41	90.70
CG	11	143.42	84.19	82.22	102.50	90.45	90.42
CG	12	143.42	86.94	81.33	95.31	88.64	90.34
CG	13	143.48	84.53	81.48	101.23	88.16	90.17
CG	14	143.50	82.39	82.70	96.91	88.55	90.27
CG	15	143.39	85.20	82.70	96.94	90.69	91.28
CG	16	143.31	85.67	82.22	92.63	89.20	90.53
CG	17	143.42	83.94	82.13	98.17	91.45	90.59
CG	18	143.59	84.92	82.30	97.77	91.61	90.63
CG	19	143.53	85.20	81.94	99.59	88.84	91.44

Figure 12: Execution Times on Windows XP

A.1.2 Derived Data

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	1.88	1.41	1.25	1.16	1.18	1.19
IS	150.25	75.37	51.17	39.57		
EP	51.27	25.65	17.12	12.86	12.32	12.03
BT	833.96	420.33	292.38	230.71	230.13	228.66
CG	143.45	84.47	81.88	97.79	89.26	90.89

Figure 13: Average Times on Windows XP

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.27	0.15	0.10	0.07	0.07	0.07
IS	2.18	1.88	1.71	1.60		
EP	1.71	1.41	1.23	1.11	1.09	1.08
BT	2.92	2.62	2.47	2.36	2.36	2.36
CG	2.16	1.93	1.91	1.99	1.95	1.96

Figure 14: Logarithms of Average Times on Windows XP

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.04	0.02	0.02	0.02	0.03	0.02
IS	0.04	0.08	0.53	0.07		
EP	0.01	0.01	0.01	0.01	0.03	0.02
BT	0.27	0.49	5.81	1.44	1.23	0.57
CG	0.16	1.11	0.84	4.28	1.44	1.41

Figure 15: Standard Deviations of Times on Windows XP

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	1.00	1.33	1.51	1.61	1.59	1.58
IS	1.00	1.99	2.94	3.80		
EP	1.00	2.00	2.99	3.99	4.16	4.26
BT	1.00	1.98	2.85	3.61	3.62	3.65
CG	1.00	1.70	1.75	1.47	1.61	1.58

Figure 16: Average Speedup Ratios on Windows XP

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.00	0.04	0.04	0.04	0.04	0.04
IS	0.00	0.00	0.03	0.01		
EP	0.00	0.00	0.00	0.00	0.01	0.01
BT	0.00	0.00	0.06	0.02	0.02	0.01
CG	0.00	0.02	0.02	0.07	0.03	0.02

Figure 17: Standard Deviations of Speedup Ratios on Windows XP

A.2 Gentoo

A.2.1 Collected Data

Benchmark	Rep	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0	1.92	1.50	1.32	1.24	1.33	1.24
mvp	1	1.92	1.45	1.32	1.22	1.21	1.22
mvp	2	1.95	1.45	1.28	1.24	1.23	1.22
mvp	3	1.93	1.45	1.30	1.22	1.30	1.25
mvp	4	1.93	1.47	1.30	1.22	1.27	1.24
mvp	5	1.92	1.46	1.29	1.23	1.22	1.21
mvp	6	1.92	1.46	1.29	1.23	1.31	1.28
mvp	7	1.93	1.44	1.29	1.27	1.29	1.27
mvp	8	1.93	1.44	1.29	1.23	1.31	1.29
mvp	9	1.94	1.44	1.31	1.25	1.22	1.24
mvp	10	1.92	1.49	1.29	1.22	1.28	1.25
mvp	11	1.94	1.44	1.30	1.21	1.34	1.25
mvp	12	1.94	1.46	1.32	1.22	1.22	1.28
mvp	13	1.94	1.44	1.28	1.23	1.33	1.24
mvp	14	1.96	1.45	1.30	1.25	1.30	1.28
mvp	15	1.95	1.45	1.48	1.22	1.33	1.27
mvp	16	1.94	1.45	1.31	1.22	1.31	1.20
mvp	17	1.94	1.44	1.31	1.21	1.34	1.28
mvp	18	1.94	1.47	1.32	1.23	1.28	1.29
mvp	19	1.94	1.44	1.30	1.23	1.33	1.24
IS	0	56.83	29.91	21.51	18.52		
IS	1	56.49	29.00	21.71	18.43		
IS	2	55.95	29.38	21.21	18.48		
IS	3	56.33	28.72	21.59	18.54		
IS	4	56.70	29.21	21.31	18.38		
IS	5	56.03	29.30	21.69	18.48		
IS	6	56.05	29.52	22.29	18.75		
IS	7	56.72	28.70	21.20	18.47		
IS	8	56.08	28.79	21.32	18.42		
IS	9	57.98	29.11	21.53	18.67		
IS	10	57.98	29.32	21.31	18.45		
IS	11	56.17	28.63	21.20	18.58		
IS	12	57.53	29.64	21.45	18.39		
IS	13	56.55	28.89	21.69	18.49		
IS	14	56.77	29.11	21.16	18.38		
IS	15	56.09	28.87	21.16	18.38		
IS	16	56.59	28.87	21.17	18.48		

IS	17	56.82	29.11	21.68	18.40		
IS	18	58.00	28.89	21.62	18.52		
IS	19	57.17	28.79	21.18	18.51		
EP	0	35.91	18.12	12.05	9.10	9.02	9.07
EP	1	35.94	17.92	12.08	9.05	9.03	9.02
EP	2	35.96	18.04	12.05	9.04	9.12	9.09
EP	3	36.09	17.97	11.95	9.15	9.09	9.10
EP	4	35.71	18.05	12.06	9.05	9.06	9.03
EP	5	35.72	18.06	12.05	9.05	9.07	9.09
EP	6	35.89	18.15	12.05	8.98	9.06	9.07
EP	7	36.13	18.06	12.07	9.05	9.05	9.08
EP	8	35.80	18.04	12.06	9.00	9.07	9.12
EP	9	35.76	17.92	11.96	9.05	9.08	9.09
EP	10	36.15	18.08	12.03	9.19	9.09	9.12
EP	11	35.74	18.10	12.05	9.05	9.09	9.10
EP	12	36.08	18.05	12.19	9.06	9.09	9.09
EP	13	35.53	18.05	12.05	9.04	9.07	9.10
EP	14	35.76	18.05	12.06	9.21	9.14	9.11
EP	15	35.50	18.06	12.05	9.16	9.09	9.07
EP	16	35.80	18.05	12.05	9.06	9.11	9.07
EP	17	35.79	17.91	12.05	9.07	9.09	9.11
EP	18	36.09	18.12	12.05	9.23	9.07	9.09
EP	19	35.74	18.10	12.05	9.08	9.11	9.08
BT	0	123.72	70.99	53.21	46.78	47.04	48.32
BT	1	123.42	71.98	53.42	46.79	46.90	48.11
BT	2	123.22	70.32	53.51	46.58	47.39	48.13
BT	3	123.75	69.12	53.24	46.70	47.08	48.39
BT	4	123.51	69.53	53.20	46.79	46.93	48.31
BT	5	123.30	71.09	53.44	47.15	47.55	48.32
BT	6	123.47	71.01	53.40	46.70	47.36	48.19
BT	7	123.90	68.73	53.37	46.78	46.99	47.96
BT	8	123.57	69.93	53.41	46.77	47.00	48.20
BT	9	123.84	70.98	53.86	47.10	47.23	48.21
BT	10	123.42	68.60	53.28	46.74	47.11	48.27
BT	11	123.40	70.74	53.77	46.77	46.93	48.55
BT	12	124.42	70.48	53.66	46.83	47.01	48.32
BT	13	123.63	68.68	53.39	46.82	47.21	48.39
BT	14	123.51	70.79	53.90	46.76	46.93	48.34
BT	15	123.38	68.72	53.35	46.82	47.13	48.20
BT	16	123.55	68.76	53.34	47.24	47.08	48.32
BT	17	123.52	71.11	53.31	47.14	47.12	48.28
BT	18	123.91	68.81	53.73	47.28	47.26	48.25
BT	19	123.42	70.94	53.29	46.93	46.88	48.46
CG	0	193.05	80.87	74.84	74.68	74.95	76.08
CG	1	180.34	76.38	75.47	73.83	77.33	76.02
CG	2	179.46	80.11	72.70	75.20	75.11	81.12
CG	3	178.81	72.55	73.11	74.74	76.33	74.45
CG	4	202.56	80.82	74.84	77.16	75.34	79.38
CG	5	164.84	81.20	72.27	74.56	74.73	75.59
CG	6	170.70	75.48	78.88	74.89	73.93	76.12

CG	7	171.58	71.40	73.81	72.93	76.71	76.13
CG	8	200.57	80.07	72.92	74.53	73.49	75.24
CG	9	166.37	79.35	73.51	74.88	74.23	75.86
CG	10	206.55	77.27	79.72	73.93	75.12	76.76
CG	11	192.88	79.24	72.89	77.14	76.07	78.01
CG	12	175.95	79.21	75.37	75.00	77.27	76.63
CG	13	233.01	82.32	71.60	74.61	73.34	78.06
CG	14	171.28	80.31	76.53	74.24	80.14	75.65
CG	15	208.05	78.82	74.89	72.71	75.38	75.15
CG	16	165.79	73.57	73.26	75.85	75.12	76.65
CG	17	187.37	80.26	79.65	75.18	76.68	79.01
CG	18	198.56	81.97	75.31	74.66	74.39	76.63
CG	19	224.42	79.71	73.08	75.47	73.31	76.45

Figure 18: Execution Times on Gentoo

A.2.2 Derived Data

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	1.93	1.45	1.31	1.23	1.29	1.25
IS	56.74	29.09	21.45	18.48		
EP	35.85	18.05	12.05	9.08	9.08	9.08
BT	123.59	70.07	53.45	46.87	47.11	48.28
CG	188.61	78.55	74.73	74.81	75.45	76.75

Figure 19: Average Times on Gentoo

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.29	0.16	0.12	0.09	0.11	0.10
IS	1.75	1.46	1.33	1.27		
EP	1.55	1.26	1.08	0.96	0.96	0.96
BT	2.09	1.85	1.73	1.67	1.67	1.68
CG	2.28	1.90	1.87	1.87	1.88	1.89

Figure 20: Logarithms of Average Times on Gentoo

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.01	0.02	0.04	0.01	0.04	0.03
IS	0.67	0.34	0.29	0.10		
EP	0.19	0.07	0.05	0.07	0.03	0.03
BT	0.27	1.09	0.21	0.20	0.18	0.13
CG	19.64	3.12	2.38	1.10	1.65	1.62

Figure 21: Standard Deviations of Times on Gentoo

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	1.00	1.33	1.48	1.57	1.50	1.55
IS	1.00	1.95	2.65	3.07		
EP	1.00	1.99	2.98	3.95	3.95	3.95
BT	1.00	1.76	2.31	2.64	2.62	2.56
CG	1.00	2.40	2.52	2.52	2.50	2.46

Figure 22: Average Speedup Ratios on Gentoo

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.00	0.02	0.04	0.02	0.05	0.03
IS	0.00	0.03	0.05	0.04		
EP	0.00	0.01	0.02	0.03	0.03	0.02
BT	0.00	0.03	0.01	0.01	0.01	0.01
CG	0.00	0.22	0.29	0.26	0.29	0.25

Figure 23: Standard Deviations of Speedup Ratios on Gentoo

A.3 Debian

A.3.1 Collected Data

Benchmark	Rep	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0	2.05	1.50	1.35	1.28	1.33	1.27
mvp	1	1.99	1.50	1.35	1.28	1.26	1.31
mvp	2	1.99	1.50	1.34	1.26	1.27	1.37
mvp	3	1.99	1.50	1.34	1.27	1.28	1.26
mvp	4	1.99	1.50	1.35	1.28	1.28	1.29
mvp	5	1.99	1.50	1.35	1.26	1.34	1.31
mvp	6	1.99	1.51	1.34	1.26	1.27	1.27
mvp	7	1.99	1.51	1.34	1.26	1.35	1.28
mvp	8	1.99	1.50	1.35	1.26	1.28	1.32
mvp	9	1.98	1.49	1.34	1.25	1.27	1.27
mvp	10	1.98	1.49	1.34	1.28	1.36	1.29
mvp	11	1.99	1.50	1.35	1.27	1.32	1.30
mvp	12	1.99	1.50	1.35	1.27	1.27	1.31
mvp	13	1.99	1.50	1.34	1.27	1.27	1.34
mvp	14	1.99	1.50	1.35	1.26	1.28	1.29
mvp	15	1.99	1.50	1.34	1.27	1.26	1.26
mvp	16	1.99	1.51	1.35	1.28	1.32	1.30
mvp	17	2.00	1.50	1.34	1.26	1.28	1.26
mvp	18	2.00	1.50	1.35	1.27	1.30	1.29
mvp	19	1.99	1.50	1.35	1.27	1.27	1.32
IS	0	56.50	28.81	21.32	18.63		
IS	1	57.88	29.20	21.81	18.72		
IS	2	57.96	29.19	21.33	18.46		
IS	3	56.32	29.45	21.35	18.61		
IS	4	57.81	29.65	21.39	18.48		
IS	5	56.87	28.93	21.43	18.50		
IS	6	58.27	29.76	21.65	18.75		
IS	7	57.07	29.27	21.34	18.49		
IS	8	56.29	29.65	21.80	18.47		
IS	9	56.80	29.19	21.70	18.51		
IS	10	57.38	29.15	21.60	18.62		
IS	11	57.66	28.89	21.40	18.46		
IS	12	57.20	29.69	21.80	18.68		
IS	13	56.49	29.50	21.72	18.89		
IS	14	56.33	29.66	21.67	18.48		
IS	15	56.45	29.78	21.39	18.49		
IS	16	56.51	29.31	21.49	18.49		

IS	17	56.50	29.04	21.29	18.48		
IS	18	56.73	29.86	21.84	18.53		
IS	19	57.90	29.68	21.34	18.56		
EP	0	36.11	18.28	12.27	9.30	9.27	9.31
EP	1	35.88	18.25	12.37	9.25	9.32	9.35
EP	2	36.08	18.45	12.27	9.28	9.31	9.31
EP	3	35.74	18.44	12.23	9.37	9.33	9.34
EP	4	35.74	18.44	12.15	9.37	9.39	9.31
EP	5	35.68	18.24	12.24	9.26	9.32	9.32
EP	6	36.21	18.31	12.26	9.25	9.30	9.37
EP	7	35.83	18.27	12.27	9.32	9.31	9.37
EP	8	36.10	18.44	12.25	9.28	9.24	9.32
EP	9	36.33	18.13	12.40	9.26	9.25	9.36
EP	10	36.28	18.07	12.36	9.30	9.24	9.33
EP	11	36.47	18.07	12.40	9.26	9.29	9.24
EP	12	35.72	18.44	12.22	9.26	9.33	9.24
EP	13	35.84	18.10	12.32	9.27	9.29	9.31
EP	14	36.35	18.31	12.27	9.21	9.29	9.33
EP	15	36.54	18.14	12.32	9.27	9.30	9.33
EP	16	35.92	18.24	12.27	9.37	9.28	9.37
EP	17	36.38	18.24	12.38	9.25	9.34	9.35
EP	18	35.83	18.17	12.40	9.37	9.33	9.33
EP	19	35.83	18.16	12.40	9.25	9.34	9.36
BT	0	123.25	67.70	52.40	45.76	46.14	47.18
BT	1	123.03	68.90	52.34	45.42	45.91	47.10
BT	2	123.21	67.86	52.27	45.47	45.90	47.15
BT	3	123.12	68.23	52.21	45.82	45.71	47.06
BT	4	123.20	68.94	52.31	45.77	45.96	47.07
BT	5	123.20	68.14	52.34	45.33	46.02	47.21
BT	6	124.10	68.46	52.08	45.29	45.98	47.10
BT	7	123.23	70.10	52.31	45.49	45.83	47.00
BT	8	123.39	67.77	52.27	45.53	45.94	47.04
BT	9	123.31	68.55	52.36	45.27	46.20	47.12
BT	10	123.26	67.86	52.55	45.74	46.31	47.27
BT	11	123.13	67.88	52.32	45.43	46.01	47.10
BT	12	123.55	68.05	52.41	45.70	45.79	47.08
BT	13	123.11	67.75	52.53	45.36	46.02	47.15
BT	14	123.31	67.82	52.70	45.54	45.95	47.05
BT	15	123.10	68.08	52.43	46.15	46.01	47.26
BT	16	123.16	67.73	52.56	45.52	45.88	47.46
BT	17	123.15	67.86	52.82	45.60	45.86	47.06
BT	18	123.10	69.72	52.70	45.67	45.90	47.19
BT	19	123.11	67.59	52.33	45.82	46.03	46.95
CG	0	139.40	65.92	68.62	66.67	69.16	74.04
CG	1	139.24	67.31	68.62	67.78	70.04	72.83
CG	2	139.30	66.61	68.45	68.22	70.32	71.81
CG	3	139.01	65.79	68.11	67.54	69.94	71.77
CG	4	139.21	65.00	67.97	66.75	71.12	71.57
CG	5	139.32	66.58	68.10	68.61	71.18	72.21
CG	6	139.28	65.04	68.39	67.59	70.43	72.01

CG	7	139.18	64.75	68.43	69.39	70.42	72.78
CG	8	139.23	65.09	67.24	68.13	71.21	71.56
CG	9	139.12	64.13	69.00	69.20	70.58	72.26
CG	10	139.33	65.40	68.46	68.32	69.63	71.96
CG	11	139.24	64.02	69.05	68.37	70.73	72.29
CG	12	139.14	65.24	68.61	68.36	70.29	71.71
CG	13	139.37	65.41	67.62	67.21	71.10	71.86
CG	14	139.30	65.00	68.74	68.78	69.40	72.00
CG	15	139.29	65.33	69.80	68.49	69.37	72.33
CG	16	138.98	65.11	69.05	69.18	69.80	71.95
CG	17	139.32	65.42	68.72	69.80	69.65	71.78
CG	18	139.41	64.67	68.50	67.98	69.89	71.33
CG	19	139.18	65.09	68.73	67.38	69.74	71.71

Figure 24: Execution Times on Debian

A.3.2 Derived Data

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	1.99	1.50	1.34	1.27	1.29	1.30
IS	57.05	29.38	21.53	18.56		
EP	36.04	18.26	12.30	9.29	9.30	9.33
BT	123.25	68.25	52.41	45.58	45.97	47.13
CG	139.24	65.35	68.51	68.19	70.20	72.09

Figure 25: Average Times on Debian

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.30	0.18	0.13	0.10	0.11	0.11
IS	1.76	1.47	1.33	1.27		
EP	1.56	1.26	1.09	0.97	0.97	0.97
BT	2.09	1.83	1.72	1.66	1.66	1.67
CG	2.14	1.82	1.84	1.83	1.85	1.86

Figure 26: Logarithms of Average Times on Debian

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.01	0.01	0.00	0.01	0.03	0.03
IS	0.66	0.33	0.20	0.12		
EP	0.28	0.13	0.07	0.05	0.04	0.04
BT	0.23	0.69	0.18	0.22	0.14	0.11
CG	0.12	0.80	0.55	0.85	0.64	0.60

Figure 27: Standard Deviations of Times on Debian

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	1.00	1.33	1.48	1.57	1.54	1.54
IS	1.00	1.94	2.65	3.07		
EP	1.00	1.97	2.93	3.88	3.87	3.86
BT	1.00	1.81	2.35	2.70	2.68	2.62
CG	1.00	2.13	2.03	2.04	1.98	1.93

Figure 28: Average Speedup Ratios on Debian

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.00	0.01	0.01	0.01	0.03	0.04
IS	0.00	0.03	0.04	0.04		
EP	0.00	0.02	0.02	0.04	0.04	0.03
BT	0.00	0.02	0.01	0.02	0.01	0.01
CG	0.00	0.03	0.02	0.03	0.02	0.02

Figure 29: Standard Deviations of Speedup Ratios on Debian

A.4 FreeBSD

A.4.1 Collected Data

Benchmark	Rep	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0	2.08	1.53	1.37	1.30	1.29	1.30
mvp	1	2.05	1.54	1.38	1.29	1.29	1.29
mvp	2	2.05	1.54	1.37	1.29	1.29	1.29
mvp	3	2.05	1.55	1.37	1.28	1.30	1.35
mvp	4	2.05	1.54	1.37	1.29	1.29	1.36
mvp	5	2.05	1.55	1.36	1.30	1.30	1.29
mvp	6	2.06	1.54	1.37	1.29	1.30	1.29
mvp	7	2.04	1.54	1.36	1.30	1.29	1.30
mvp	8	2.06	1.55	1.36	1.29	1.29	1.32
mvp	9	2.05	1.54	1.37	1.28	1.30	1.30
mvp	10	2.05	1.54	1.36	1.30	1.29	1.35
mvp	11	2.05	1.54	1.36	1.28	1.33	1.29
mvp	12	2.06	1.55	1.38	1.28	1.29	1.29
mvp	13	2.05	1.55	1.37	1.29	1.30	1.36
mvp	14	2.05	1.54	1.36	1.30	1.29	1.29
mvp	15	2.06	1.55	1.38	1.29	1.29	1.29
mvp	16	2.05	1.54	1.36	1.30	1.29	1.30
mvp	17	2.05	1.55	1.37	1.29	1.33	1.30
mvp	18	2.05	1.55	1.36	1.29	1.30	1.34
mvp	19	2.05	1.55	1.37	1.30	1.29	1.36
EP	0	31.89	16.04	10.68	8.00	8.39	8.36
EP	1	31.93	15.98	10.66	8.00	8.30	8.46
EP	2	31.93	15.98	10.66	8.00	8.19	8.28
EP	3	31.88	15.98	10.66	8.00	8.02	8.29
EP	4	31.93	15.98	10.66	8.00	8.11	8.18
EP	5	31.85	16.01	10.68	8.01	8.31	8.97
EP	6	31.93	15.98	10.66	8.01	8.21	8.12
EP	7	31.92	15.98	10.66	8.01	8.18	8.33
EP	8	31.85	15.99	10.68	8.01	8.09	8.20
EP	9	31.93	15.98	10.65	8.00	8.10	8.51
EP	10	31.93	15.98	10.66	8.00	8.21	8.84
EP	11	31.92	15.98	10.66	8.00	8.09	8.52
EP	12	31.92	15.98	10.66	8.00	8.21	8.84
EP	13	31.85	16.02	10.68	8.01	8.02	8.53
EP	14	31.93	15.98	10.66	8.00	8.10	8.32
EP	15	31.93	15.98	10.65	8.00	8.03	8.76

EP	16	31.90	15.98	10.66	8.01	8.22	8.09
EP	17	31.85	16.02	10.68	8.01	8.01	8.43
EP	18	31.89	16.01	10.67	8.01	8.21	8.19
EP	19	31.85	16.01	10.68	8.01	8.31	8.78
BT	8	134.40	73.46	56.35	48.62	53.56	56.39
BT	9	134.78	74.20	56.47	48.77	53.36	56.51
BT	10	134.73	74.21	56.36	48.80	53.52	51.48
BT	11	134.70	74.66	56.27	48.84	53.01	56.18
BT	12	134.69	74.05	56.39	48.76	53.16	57.56
BT	13	134.32	73.76	56.28	48.68	53.79	56.59
BT	14	134.56	74.17	56.24	48.79	53.04	51.96
BT	15	134.54	74.31	56.36	48.51	53.42	51.62
BT	16	134.60	74.13	56.49	48.77	53.17	57.43
BT	17	134.47	74.10	56.35	49.10	52.30	56.48
BT	18	134.44	74.16	56.46	48.72	52.59	56.97
BT	19	134.70	74.26	56.23	48.77	53.01	56.49
BT	20	134.63	74.06	56.36	48.80	53.11	58.14
BT	21	134.66	73.85	56.31	49.01	53.11	55.89
BT	22	134.56	74.24	56.37	48.68	52.05	51.07
BT	23	134.59	74.41	56.22	48.90	53.21	56.63
BT	24	134.71	74.00	56.37	48.69	52.33	54.98
BT	25	134.60	74.04	56.29	48.98	52.91	55.58
BT	26	134.37	73.81	56.39	48.59	52.96	56.54
BT	27	134.37	74.27	56.37	48.77	52.33	55.98
CG	0	115.36	71.53	74.18	77.62	81.36	82.30
CG	1	115.48	77.74	74.18	79.11	80.14	85.98
CG	2	115.84	77.05	74.96	79.06	82.83	83.26
CG	3	115.76	80.50	74.51	78.21	81.19	83.73
CG	4	115.96	79.10	75.32	79.09	80.79	81.63
CG	5	115.88	76.06	75.00	79.10	81.97	82.11
CG	6	115.92	77.21	75.55	79.06	80.87	84.63
CG	7	115.44	76.25	75.55	78.42	81.12	83.91
CG	8	115.86	79.16	74.14	79.07	81.51	82.37
CG	9	115.42	78.58	75.26	75.88	80.87	82.34
CG	10	115.41	79.14	76.73	78.70	81.85	82.87
CG	11	115.29	76.59	76.04	79.53	80.72	82.96
CG	12	115.48	75.69	75.00	78.90	80.52	82.30
CG	13	115.80	80.83	74.06	78.14	80.61	83.01
CG	14	116.00	77.17	73.84	77.90	82.12	83.86
CG	15	116.09	74.90	79.88	79.02	81.08	81.87
CG	16	115.98	74.81	76.96	79.02	80.43	83.45
CG	17	117.34	77.05	75.16	77.93	82.00	83.00
CG	18	115.79	75.84	73.65	79.11	81.53	82.40
CG	19	116.05	77.66	76.02	79.25	81.66	82.20

Figure 30: Execution Times for FreeBSD

A.4.2 Derived Data

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	2.05	1.54	1.37	1.29	1.30	1.31
EP	31.90	15.99	10.66	8.00	8.17	8.45
BT	134.57	74.11	56.35	48.78	53.00	55.52
CG	115.81	77.14	75.30	78.61	81.26	83.01

Figure 31: Average Times on FreeBSD

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.31	0.19	0.14	0.11	0.11	0.12
EP	1.50	1.20	1.03	0.90	0.91	0.93
BT	2.13	1.87	1.75	1.69	1.72	1.74
CG	2.06	1.89	1.88	1.90	1.91	1.92

Figure 32: Logarithms of Average Times on FreeBSD

Benchmark	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.01	0.01	0.01	0.00	0.01	0.03
EP	0.03	0.02	0.01	0.01	0.11	0.27
BT	0.14	0.26	0.08	0.14	0.46	2.16
CG	0.44	2.15	1.42	0.83	0.68	1.04

Figure 33: Standard Deviations of Times on FreeBSD

Benchmarks	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	1.00	1.33	1.50	1.59	1.58	1.56
EP	1.00	1.99	2.99	3.99	3.91	3.78
BT	1.00	1.82	2.39	2.76	2.54	2.42
CG	1.00	1.50	1.54	1.47	1.43	1.40

Figure 34: Average Speedup Ratios on FreeBSD

Benchmarks	1 Thread	2 Threads	3 Threads	4 Threads	8 Threads	16 Threads
mvp	0.00	0.01	0.01	0.01	0.01	0.03
EP	0.00	0.00	0.01	0.01	0.05	0.12
BT	0.00	0.01	0.00	0.01	0.02	0.10
CG	0.00	0.04	0.03	0.02	0.01	0.02

Figure 35: Standard Deviations of Speedup Ratios on FreeBSD

Appendix B Source Code

B.1 Matrix-Vector Product Benchmark - C

```
// OpenMP Matrix-Vector Product Benchmark
// Michael Craft
// 1/30/09
// Based on code from the book "Using OpenMP".
// Requires C99 standard.

#include <stdio.h>
#include <stdlib.h>

#ifdef _OPENMP
      #include <omp.h>
      #define PARALLEL 1
#else
      #define omp_get_thread_num() 0
      #define omp_get_num_threads() 1
      #define PARALLEL 0
#endif

void mxv( int m, int n, double * restrict a, double * restrict b, dou-
ble * restrict c,int threads) {
      int i, j, tid, blarg=PARALLEL, nthreads, announced=0;

#pragma omp parallel for default(none)
shared(m,n,a,b,c,blarg,announced) private(i,j,nthreads,tid)
num_threads(threads)
      for (i=0; i<m; i++) {
            if(blarg) {
                  tid = omp_get_thread_num();
                  //Uncomment below for debuging.
                  //printf("This is thread %d.\n", tid);

                  if(tid == 0 && !announced) {
                        printf("%d threads were used.\n",
omp_get_num_threads() );
                        announced = 1;
                  }
            }

        a[ i ] = 0.0;
        for (j=0; j<n; j++)
            a[ i ] += b[ i*n + j ] * c[ j ];
      }
// End parallel
}

int main(int argc, char *argv[]) {
      if ((argc!=4) || (atoi(argv[1])<1) || (atoi(argv[2])<1) ||
(atoi(argv[3])<1)) {
            printf("Incorrect arguments.\n");
            return 1;
      }
```

```
	else {
		int urgle=PARALLEL;
		if(urgle)
			printf("Compiled with OpenMP support.\n");
		else
			printf("Compiled without OpenMP support.\n");

		int threads = atoi(argv[1]);
		printf("Attempting to use %d threads.\n", threads);

		int m = atoi(argv[2]);
		int n = atoi(argv[3]);
		printf("Using matrix size of %dx%d.\n", m, n);

		double *a, *b, *c;
		int i, j;

		if ( (a=(double *)malloc(m*sizeof(double))) == NULL )
			perror( "memory allocation for a" );
		if ( (b=(double *)malloc(m*n*sizeof(double))) == NULL )
			perror( "memory allocation for b" );
		if ( (c=(double *)malloc(n*sizeof(double))) == NULL )
			perror( "memory allocation for c" );

		for (j=0; j<n; j++)
			c[ j ] = 2.0;
		for (i=0; i<m; i++)
			for (j=0; j<n; j++)
				b[ i*n + j ] = i;

		(void) mxv( m, n, a, b, c, threads);

		free(a); free(b); free(c);
		return 0;
	}
}
```

B.2 Benchmark Data Collector – Python

```
# Benchmark Data Collecter
# Michael Craft
# 2/7/9

import os
import sys
from time import time
import csv

# Global variables for error detection, and run count.
dang = 0
runs = 20

threads = (1,2,3,4,8,16)

def record(bench, rep, run1, run2, run3, run4, run8, run16):
	data = (bench, rep, run1, run2, run3, run4, run8, run16)
	file = open('cygwin.csv', 'a')
	writer = csv.writer(file)
	writer.writerow(data)
	file.close()

def mvp():
	# Matrix - Vector Product Benchmark
	commandName = "mvp-cygwin.exe"
	times = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

	print "Running Matrix-Vector Product benchmark", runs,"times for
threads", threads,"\n"

	for r in range(runs):
		for t in threads:
			command = "./" + commandName + " " +  str(t) + "
10000 10000"

			start = time()
			exit = os.system(command)
			end = time()
			runTime = end - start

			if exit == 0:
				times[t] = runTime
			else:
				dang = 1

		record("mvp", r, times[1], times[2], times[3], times[4],
times[8], times[16])

def IS():
	# IS NAS Benchmark
	commandName = "NPB3.3-OMP/bin/is.C"
	times = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
	print "Running IS benchmak", runs,"times for threads",
threads,"\n"

	for r in range(runs):
		for t in threads:
			command = commandName + " " +  str(t)

			start = time()
			exit = os.system(command)
			end = time()
			runTime = end - start

			if exit == 0:
				times[t] = runTime
			else:
				dang = 1

		record("IS", r, times[1], times[2], times[3], times[4],
times[8], times[16])

def EP():
	# EP NAS Benchmark
	commandName = "NPB3.3-OMP/bin/ep.A"
	times = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

	print "Running EP benchmak", runs,"times for threads",
threads,"\n"

	for r in range(runs):
		for t in threads:
			command = commandName + " " +  str(t)

			start = time()
			exit = os.system(command)
			end = time()
			runTime = end - start

			if exit == 0:
				times[t] = runTime
			else:
				dang = 1

		record("EP", r, times[1], times[2], times[3], times[4],
times[8], times[16])

def BT():
	# BT NAS Benchmark
	commandName = "NPB3.3-OMP/bin/bt.A"
	times = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

	print "Running BT benchmak", runs,"times for threads",
threads,"\n"

	for r in range(runs):
		for t in threads:
			command = "OMP_NUM_THREADS=\"" + str(t) + "\" " +
commandName
```

```
                start = time()
                exit = os.system(command)
                end = time()
                runTime = end - start

                if exit == 0:
                        times[t] = runTime
                else:
                        dang = 1

        record("BT", r+8, times[1], times[2], times[3], times[4],
times[8], times[16])

def CG():
        # CG NAS Benchmark
        commandName = "NPB3.3-OMP/bin/cg.B"
        times = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

        print "Running CG benchmak", runs,"times for threads",
threads,"\n"

        for r in range(runs):
                for t in threads:
                        command = "OMP_NUM_THREADS=\"" + str(t) + "\" " +
commandName

                        start = time()
                        exit = os.system(command)
                        end = time()
                        runTime = end - start

                        if exit == 0:
                                times[t] = runTime
                        else:
                                dang = 1

                record("CG", r, times[1], times[2], times[3], times[4],
times[8], times[16])

# Main
if __name__ == "__main__":
        print "Benchmark Data Collecter\n"

        #put title row in
        record("Bench", "Rep", "1 Thread", "2 Threads", "3 Threads", "4
Threads", "8 Threads", "16 Threads")

        try:
                bench = sys.argv[1]
        except:
                bench = "all"

        if bench == "mvp" or bench == "all":
                mvp()
        if bench == "IS" or bench == "all":
                IS()
```

```
if bench == "EP" or bench == "all":
      EP()
if bench == "BT" or bench == "all":
      BT()
if bench == "CG" or bench == "all":
      CG()

if dang == 1:
      print "Warnig! Some bad error occured!"
      record("WARNING", "BAD", "ERROR", "OCCURED", 0, 0, 0, 0)
```