SLAC–PUB–7990
November, 1998

Choosing CPUs in an Open Market: System Performance Testing for the BABAR Online Farm

T.J. Pavel
For the BABAR Computing Group [*]

Abstract

BABAR is a high-rate experiment to study CP violation in asymmetric e^+e^- collisions. The BABAR Online Farm is a pool of workstations responsible for the last layer of event selection, as well as for full reconstruction of selected events and for monitoring functions. A large number of machine architectures were evaluated for use in this Online Farm. We present an overview of the results of this evaluation, which include tests of low-level OS primitives, tests of memory architecture, and tests of application-specific CPU performance. Factors of general interest to others making hardware decisions are highlighted. Performance of current BABAR reconstruction (written in C++) is found to scale fairly well with SPECint95, but with some noticeable deviations. Even for machines with similar SPEC CPU ratings, large variations in memory system performance exist. No single operating system has an overall edge in the performance of its primitives. In particular, freeware operating systems perform no worse overall than the commercial offerings.

Introduction

The BABAR experiment is a high energy physics detector that will operate at the PEP-II asymmetric e^+e^- storage ring at SLAC to study *CP* violation in the *B* meson system. The design of the experiment is described in detail in [1]. An unusual aspect for e^+e^- experiments is the use of a partial-reconstruction software selection (L3 trigger) for reduction of the data set to a manageable rate. Even with the L3 trigger, BABAR will record some 10^9 events per year (resulting in 30 TB/year of raw data alone). As in other HEP experiments, this L3 trigger will be performed on a farm of commercial Unix systems. In addition to the L3 trigger, however, this farm will be required to support the building of events from the front-end processors over a TCP/IP network [2], and some part of the farm will be used to perform a full first-pass reconstruction on all selected events [3].

There are many factors that go into selecting a computing platform for such an application. In the ideal world, one is free to make that choice purely on the grounds of performance and price. The BABAR Online Farm selection was not quite ideal, but we were able to evaluate a large selection of different platforms for consideration. The first goal of such an evaluation is to quantify the principal measures of performance relative to the desired tasks. For the BABAR tasks of event building, L3 triggering, and reconstruction, we identified the main performance components as CPU speed, network speed, and (pseudo-)real-time responsiveness of the operating system (OS). The measurements of network performance are described in another paper in these proceedings [4], while this paper focusses on the CPU and OS measurements.

One can consider the performance of computing systems to be generally factorizable into the performance of the underlying hardware and the performance of the software that runs on top of it. The hardware layer is principally made up of what is considered CPU performance, but it also includes an increasingly important component of memory performance. Since our goal is to compare performance of several candidate machines for the BABAR Online Farm, we can consider the software layer to stop at the operating system, as the remaining software should be identical for different systems.

In order to measure these three aspects of system performance, we drew upon the work of others in measuring performance (see below) and did not need to create any new benchmark programs of our own.

[*] This work was supported in part by Department of Energy contract DE–AC03–76SF00515.

Presented at the International Conference on Computing in High-Energy Physics (CHEP98), Chicago, IL, August 30, to September 4, 1998

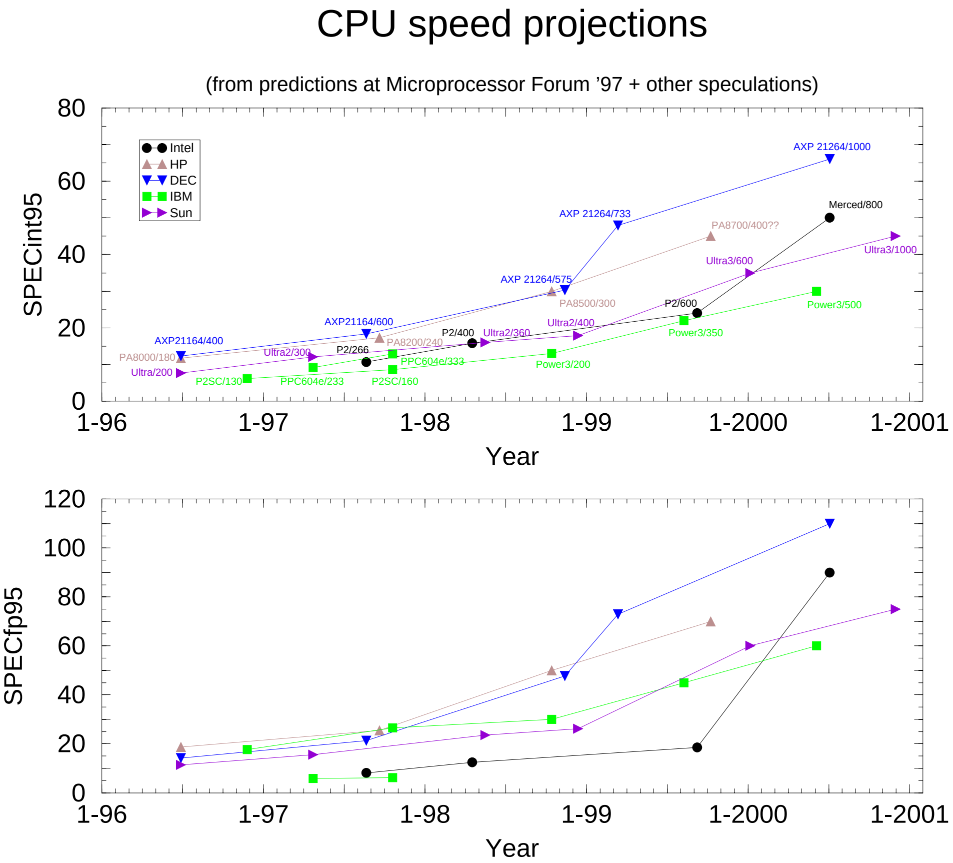

Figure 1: SPEC ratings for various CPU families over the past few and next few years. Future projections are based on [6] as well as vendor press releases. Within the uncertainty of future predictions, it is apparent that all of the vendors considered here will continue to have competitive products tracking Moore's law of exponential growth for the next several years.

These tools should be of interest to anyone faced with making similar evaluations. In the end, we found that performance alone ruled out very few candidate platforms, leaving price and the more subjective criteria as the primary decision-making factors.

CPU Performance

Performance testing of CPUs is one of the oldest pursuits in computer performance measurement. There is a long history of development of different benchmark programs, but the current industry standard metric is the SPEC CPU95 suite [5]. These benchmarks are generally run by the CPU vendors and the results are widely available. The SPEC95 suite attempts to use a cross section of representative "real-world" code, with a particular eye towards programs that have realistic memory and cache footprints. Currently, seven C programs make up the SPECint95 benchmark, and ten Fortran-77 programs make up the SPECfp95 benchmark. The results of each of these sets of tests are scaled to the times taken by a SPARCstation 10/40 and geometrically averaged. Clearly one would like to make use of the wealth of published SPEC results in making CPU decisions (rather than, for example, running test code on hundreds of different machines).

Furthermore, information from vendors about future products is only available as SPEC performance targets (or as MHz). One cannot run application-specific tests on non-existent processors! A collection of such information is shown in Fig. 1. One can see the exponential growth of CPU speeds, and that such growth shows no signs of abating in the near future. Furthermore, one sees from Fig. 1 that no vendor holds a clear advantage. The Alpha is expected to continue to be the fastest individual CPU. In addition, the Intel

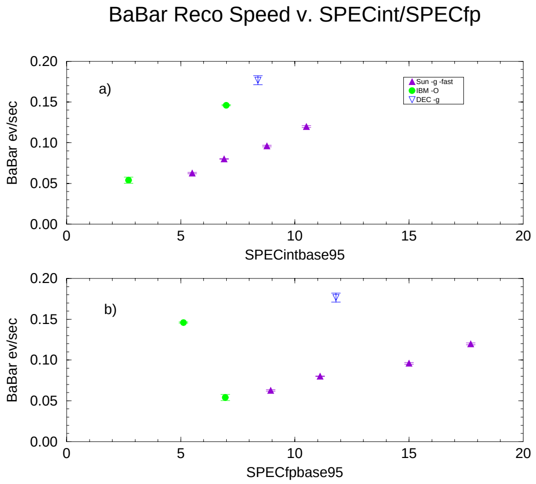

Figure 2: Comparison of BABAR reconstruction performance on several machines with SPECint95(base) and SPECfp95(base). The results show a much more linear scaling with the SPECint95 points (a), indicating that SPECint95 is a better figure of merit for estimating reconstruction performance.

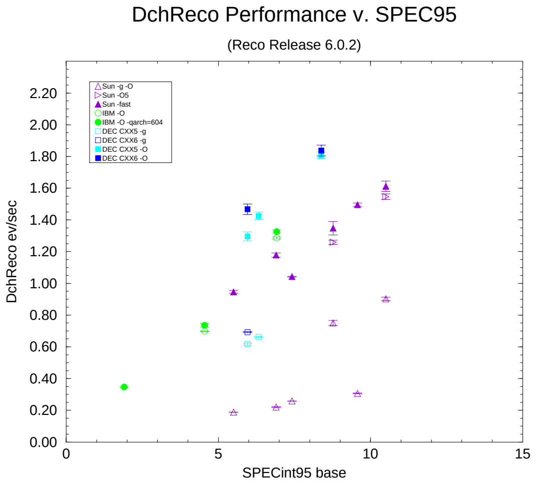

Figure 3: A comparison of application performance (for a small piece of the BABAR reconstruction) versus SPECint95 for three RISC CPU architectures. The effects of different optimization options are also shown (open points are unoptimized compilation, while the solid points are the highest level of optimization which still improved results). For the optimized points, scaling with SPECint95 is quite linear.

Merced is expected to be a very significant force in the market, but it will not have very different performance from other microprocessors which will be available at that time.

However, the SPEC results are only one aggregate number and may not really represent the performance delivered in the BABAR Online Farm. Therefore, our first investigation was to see how well our application-specific performance (as measured by the current BABAR reconstruction program) compares with SPEC95 results. One open question is how floating-point intensive HEP code really is. There clearly are a lot of floating-point calculations involved in event reconstruction, but it seems that typical HEP code is dominated by logic, control-flow, and data structures. Figure 2b shows that BABAR reconstruction does not scale well with SPECfp95. The left-most two points come from IBM machines whose SPECint95/SPECfp95 ratios differ wildly. As a result the machine with the higher SPECfp95 rating performs worse on the BABAR reconstruction than the other (which has the higher SPECint95 rating). On the other hand, in Fig. 2a the points within a given architectural family seem to lie on straight lines going through the origin, showing that scaling with SPECint95 is quite good. In this early study, however, there seems to be a fairly large disparity between different CPU families.

We examined this in detail for one small microcosm of the BABAR reconstruction (the **DchReco** module, which performs first-pass track finding). The results of this are shown in Fig. 3 and demonstrate the sizeable effects of various compilation/optimization flags (as much as ×4–5). With optimization, both UltraSPARC and RS/6000 families scale linearly with SPECint95 and are consistent to about 10%. However, both seem to differ from Alpha systems by 20–30%. In the published SPECint95 results, spreads of 10–20% appear among the individual components. Therefore, this 30% discrepancy may not be more than normal variation of results on different computing problems. Nevertheless, it seems significant in the context of our measurements. One possible explanation for the advantage would be a better C++ optimizer in Digital's compiler. This would not enter into the SPECint95 results, since those tests are exclusively C code. Another possible explanation would be some advantage of the 64-bit Alpha architecture for reconstruction code, but we have no indication of what such an advantage might be.

We also studied several other components of the BABAR reconstruction. Although the relative behavior of various machines on those components did vary somewhat, the overall conclusion of good linear scaling with SPECint95 within CPU architectures remained true. Furthermore, the consistency between SPARC and

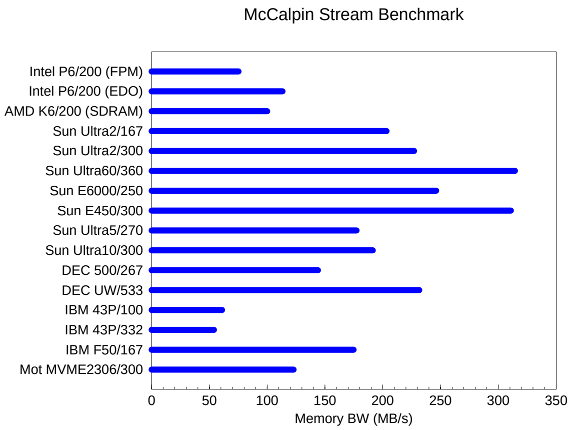

Figure 4: Memory bandwidth as measured by the McCalpin Stream benchmark [7].

PowerPC machines with respect to SPECint95 also seemed to hold throughout. More work needs to be done to understand the relative scaling with SPECint95 for Alpha and other architectures not tested here.

Memory Speed

The second major element of hardware performance studied was memory bandwidth. This was found to be very important in network performance [4], and it is also well-known to be significant in large numeric calculations [7]. Unlike the rapid exponential increase in CPU performance, memory speeds have grown much more slowly. Furthermore, it also turns out that memory performance is strongly correlated with many of the OS performance metrics discussed later [8].

We used two different tests of memory bandwidth. The results in Fig. 4 are from the McCalpin Stream benchmark [7], which is becoming a *de facto* standard for memory benchmarks. Many vendors have used it to rate their memory systems. The test consists of iterating through two large arrays copying a double at a time. This makes full use of any cache prefetching, but never reuses any data in the cache. The results are reported as the sum of read and write bandwidths, so one should divide in half to get the copy bandwidth.

The other memory bandwidth test we performed was the **mem_bw** component of the **lmbench** suite [9]. This tests copy performance via the `memcpy()` call, as well as measuring read and write bandwidths individually by looping through arrays. These results are presented in Table 1. The two tests give generally

Table 1: Memory bandwidth as measured by **lmbench** [9].

Machine	OS	**memcpy() (libc)**	`memcpy()` (unrolled)	Mem Read	Mem Write
Sun Ultra60/360 MHz	Solaris 2.6	257 MB/s	115 MB/s	178 MB/s	204 MB/s
Sun Ultra2/300 MHz	Solaris 2.5.1	187 MB/s	99 MB/s	151 MB/s	186 MB/s
Sun Ultra2/200 MHz	Solaris 2.5.1	176 MB/s	82 MB/s	126 MB/s	168 MB/s
DEC Alpha UW/533 MHz	DEC Unix 4.0D	107 MB/s	108 MB/s	200 MB/s	187 MB/s
DEC Alpha 500/266 MHz	DEC Unix 4.0B	74 MB/s	69 MB/s	127 MB/s	104 MB/s
Intel PPro/200 MHz (EDO)	Solaris 2.5.1	61 MB/s	53 MB/s	226 MB/s	90 MB/s
IBM F50/166 MHz	AIX 4.2	58 MB/s	82 MB/s	122 MB/s	126 MB/s
Intel PPro/200 MHz (FPM)	Linux 2.0.27	45 MB/s	37 MB/s	158 MB/s	60 MB/s
AMD K6/200 MHz (SDRAM)	FreeBSD 2.2.5	43 MB/s	43 MB/s	150 MB/s	74 MB/s
IBM 43P/200 MHz	AIX 4.1.5	27 MB/s	28 MB/s	64 MB/s	41 MB/s
IBM 43P/333 MHz	AIX 4.2.1	25 MB/s	26 MB/s	81 MB/s	34 MB/s

comparable results, with some exceptions like the UltraSPARC, where Sun's libc uses the UltraSPARC VIS instructions to sidestep the caches for `memcpy()`. This gives Sun systems a factor of 2 edge, for example, in memory copy over otherwise comparable Alphas.

Of interest also are the Intel systems. Here the memory bandwidth results are fairly good overall, but the write performance is much worse than read on the Pentium Pro. This is due to the overhead of the multi-processing cache-coherency protocol which is built into every Pentium Pro chip and active even on uniprocessor systems [10].

Finally, we notice that IBM's 43P (low-end) line of workstations has abysmal memory bandwidth. The same 30 MB/s `memcpy()` performance is present on the 333 MHz machines as on the early 100 MHz versions. While the F50 (4-way SMP) machine improves this considerably, it is still not competitive with Sun or DEC machines. Although not measured in our study, high-end IBM machines (with the P2SC chip) have much better memory speeds, and the upcoming Power3 workstations are claimed to perform over 500 MB/s on `memcpy()`. However, these machines were too costly for consideration for the BABAR Online Farm.

Operating System Primitives

One of the most straightforward ways to gain insight into operating system performance is through the use of microbenchmarks, tests that measure the performance of individual OS primitives. Each microbenchmark test is hopefully easy enough to understand, and one can, in principle, break down a macroscopic task into its component primitives and synthesize an understanding of its performance based upon the microbenchmark measurements.

The first systematic benchmarks of OS primitives came from John Ousterhout's study of why OS performance was not scaling well with CPU speed [8]. Some of these benchmarks were primitive, and there were some areas not really covered. This led Larry McVoy to refine Ousterhout's suite and to add additional tests, creating the **lmbench** suite [9]. This suite has been very successful, and has reportedly been used by many OS vendors for measurement and tuning. Development of **lmbench** is still proceeding, while meanwhile there have been some attempts to address some of its shortcomings [10, 11].

We present a selected subset of the **lmbench** measurements here. A sample of "200-300 MHz class" machines were used to provide a roughly comparable hardware assortment. We follow the presentation format of [9] with similar tables showing the results sorted on the column indicated with a bold heading.

System Calls

The overhead of system calls is a basic starting point for OS performance. All other OS functions will require at least the overhead of a kernel trap and associated context switch. The **lmbench** test measures a 1-byte `write()` to `/dev/null` (as opposed to the `getpid()` call, which has been optimized via caching the value in user space on some systems). Most of the systems do simple syscalls fairly efficiently, in 2–5 μs, as shown in Table 2. Unlike the systems in [8], this is usually a fairly negligible overhead. However, AIX seems to have some difficulty with syscalls.

Process Creation

The next most basic OS function is process creation. **Lmbench** tests process creation at three levels: a simple `fork()` followed by `exit()` (null process), a `fork()` which execs a trivial program (the famous hello-world program), and the common case of forking `/bin/sh` (such as in the `system()` call). The results are shown in Table 3. AIX does surprisingly well at processes, given its high syscall overhead. Digital Unix, FreeBSD, and Linux also do very well, although Linux is hurt on the `/bin/sh` test somewhat by its larger `bash` shell. Solaris is the worst of the set on processes. Reference [9] claims that this is partly due to the overhead of dynamically-linked programs, but all the other systems tested also have dynamic linking and don't show the same overheads.

Table 2: System Call Overhead

Machine	OS	MHz	**Null syscall**
DEC Alpha UW/533 MHz	DEC Unix 4.0D	533	2 μs
Sun Ultra60/360 MHz	Solaris 2.6	360	2 μs
DEC Alpha 500/266 MHz	DEC Unix 4.0B	266	3 μs
IBM 43P/333 MHz	AIX 4.2.1	333	3 μs
Intel PPro/200 MHz (256 kB L2)	Linux 2.0.27	200	3 μs
Sun Ultra2/300 MHz	Solaris 2.5.1	300	3 μs
Intel PPro/200 MHz (512kB L2)	Solaris 2.5.1	200	4 μs
Sun Ultra2/200 MHz	Solaris 2.5.1	200	5 μs
AMD K6/200 MHz	FreeBSD 2.2.5	200	5 μs
IBM F50/166 MHz	AIX 4.2	166	8 μs
IBM 43P/200 MHz	AIX 4.1.5	200	19 μs

Table 3: Process Creation

Machine	OS	Null process	**Simple process**	`/bin/sh` process
IBM 43P/333 MHz	AIX 4.2.1	2 ms	3 ms	13 ms
DEC Alpha 500/266 MHz	DEC Unix 4.0B	1 ms	4 ms	13 ms
IBM F50/166 MHz	AIX 4.2	2 ms	4 ms	16 ms
IBM 43P/200 MHz	AIX 4.1.5	2 ms	4 ms	17 ms
DEC Alpha UW/533 MHz	DEC Unix 4.0D	1 ms	5 ms	12 ms
AMD K6/200 MHz	FreeBSD 2.2.5	1 ms	5 ms	12 ms
Intel PPro/200 MHz (256 kB L2)	Linux 2.0.27	2 ms	5 ms	32 ms
Sun Ultra60/360 MHz	Solaris 2.6	2 ms	9 ms	18 ms
Sun Ultra2/300 MHz	Solaris 2.5.1	2 ms	13 ms	24 ms
Sun Ultra2/200 MHz	Solaris 2.5.1	3 ms	17 ms	35 ms
Intel PPro/200 MHz (512kB L2)	Solaris 2.5.1	4 ms	20 ms	38 ms

Signal Handling

Another fairly basic component is the installation of and activation of signal handlers. The Unix signal facility is not as rich a facility as threads or the other IPC primitives. Nevertheless, it still has its uses, and it can be particularly important in legacy code.

From the results in Table 4, we can see that Linux and Digital Unix perform best, followed closely by FreeBSD. AIX also does reasonably well. Solaris is poor at signal handling, up to 10 times worse than the best system.

Context Switch

This is one of the primary performance criteria for real-time response in an OS. Unfortunately, it is somewhat difficult to define and correspondingly hard to measure. **Lmbench** uses multiple processes with tokens passed between them via pipes. The overhead of the token passing is calculated in a single process and then subtracted from the multi-process results. **Lmbench** also seeks to measure the cost of bringing a new working set into the CPU caches as part of the context switch values; it does this by spinning through an array of tunable size between each passing of the token. Unfortunately, there is a fair amount of statistical scatter in the **lmbench** measurements, and there are some pitfalls involved in subtracting imprecisely measured quantities [10].

Table 4: Signal Handling

Machine	OS	`sigaction()`	**signal handler**
DEC Alpha UW/533 MHz	DEC Unix 4.0D	1 μs	5 μs
Intel PPro/200 MHz (256 kB L2)	Linux 2.0.27	4 μs	7 μs
DEC Alpha 500/266 MHz	DEC Unix 4.0B	2 μs	10 μs
AMD K6/200 MHz	FreeBSD 2.2.5	3 μs	10 μs
IBM 43P/333 MHz	AIX 4.2.1	1 μs	11 μs
IBM 43P/200 MHz	AIX 4.1.5	2 μs	20 μs
IBM F50/166 MHz	AIX 4.2	4 μs	27 μs
Intel PPro/200 MHz (512kB L2)	Solaris 2.5.1	4 μs	34 μs
Sun Ultra2/300 MHz	Solaris 2.5.1	3 μs	35 μs
Sun Ultra60/360 MHz	Solaris 2.6	2 μs	37 μs
Sun Ultra2/200 MHz	Solaris 2.5.1	5 μs	49 μs

Table 5: Context Switch

Machine	OS	**2-proc CSW**	8-proc CSW
Intel PPro/200 MHz (256 kB L2)	Linux 2.0.27	5 μs	6 μs
IBM 43P/333 MHz	AIX 4.2.1	5 μs	13 μs
DEC Alpha 500/266 MHz	DEC Unix 3.2G	8 μs	12 μs
DEC Alpha 500/266 MHz	DEC Unix 4.0B	9 μs	11 μs
IBM 43P/200 MHz	AIX 4.1.5	10 μs	28 μs
Sun Ultra60/360 MHz	Solaris 2.6	11 μs	12 μs
AMD K6/200 MHz	FreeBSD 2.2.5	11 μs	14 μs
DEC Alpha UW/533 MHz	DEC Unix 4.0D	12 μs	15 μs
IBM F50/166 MHz	AIX 4.2	12 μs	16 μs
Sun Ultra2/200 MHz	Solaris 2.5.1	13 μs	18 μs
Sun Ultra2/300 MHz	Solaris 2.5.1	17 μs	18 μs
Intel PPro/200 MHz (512kB L2)	Solaris 2.5.1	34 μs	42 μs

The results vary with both hardware (e.g. size of the register set) and with OS (i.e. efficiency of task switching), as seen in Table 5. The case of Pentium Pro systems is particularly instructive. Linux on the Pentium Pro is excellent, while Solaris on the same hardware is worse by a factor of 7. The SPARC Solaris systems also tend to be near the bottom of the pack. On the other hand, AIX performance varies tremendously, presumably with the speed of the hardware.

IPC Latency

Lmbench measures a collection of latencies of various IPC channels, all over the loopback network in order to separate out the effects of network link layers. Unlike IPC bandwidths, which are generally dominated by the hardware (*viz.* memory bandwidth), the latencies depend strongly on software organization and efficiency in the kernel. In particular, the time to transit through the kernel for a TCP packet can be very important for real-world applications (*e.g.* Web servers).

These results are presented in Table 6. One can see that Solaris scales well with hardware speed, unlike Digital Unix, which is mostly unchanged as the CPU speed doubles. Solaris also has the fastest RPC implementations of the set, reflecting heavy optimization work expended on this. A curious note is that the RPC implementation worsened going from Digital Unix 3.2 to 4.0. Linux does very well on pipe and UDP latencies, but its TCP code seems to be not so good. FreeBSD, on the other hand, does quite well

Table 6: IPC Latency

Machine	OS	pipe	UDP	RPC/UDP	**TCP**	RPC/TCP
Sun Ultra60/360 MHz	Solaris 2.6	26 μs	86 μs	161 μs	85 μs	193 μs
IBM 43P/333 MHz	AIX 4.2.1	25 μs	97 μs	267 μs	116 μs	312 μs
AMD K6/200 MHz	FreeBSD 2.2.5	39 μs	101 μs	195 μs	120 μs	243 μs
IBM 43P/200 MHz	AIX 4.1.5	63 μs	125 μs	336 μs	127 μs	418 μs
Sun Ultra2/300 MHz	Solaris 2.5.1	51 μs	133 μs	189 μs	130 μs	240 μs
DEC Alpha 500/266 MHz	DEC Unix 3.2G	40 μs	132 μs	255 μs	131 μs	252 μs
DEC Alpha 500/266 MHz	DEC Unix 4.0B	36 μs	131 μs	311 μs	133 μs	317 μs
Sun Ultra2/200 MHz	Solaris 2.5.1	47 μs	184 μs	233 μs	152 μs	307 μs
IBM F50/166 MHz	AIX 4.2	58 μs	146 μs	328 μs	157 μs	370 μs
DEC Alpha UW/533 MHz	DEC Unix 4.0D	37 μs	144 μs	310 μs	159 μs	418 μs
Intel PPro/200 MHz (256 kB L2)	Linux 2.0.27	25 μs	79 μs	229 μs	166 μs	232 μs
Intel PPro/200 MHz (512kB L2)	Solaris 2.5.1	90 μs	242 μs	321 μs	222 μs	362 μs

Table 7: File Operations

Machine	OS	FS type	create/s	**delete/s**
Intel PPro/200 MHz (256 kB L2)	Linux 2.0.27	EXT2	1302	19053
Intel PPro/200 MHz (512kB L2)	Solaris 2.5.1	UFS	74	120
Sun E450/300 MHz	Solaris 2.6	UFS	63	120
Sun Ultra2/300 MHz	Solaris 2.5.1	UFS	63	120
Sun Ultra2/200 MHz	Solaris 2.5.1	UFS	63	120
DEC Alpha UW/533 MHz	DEC Unix 4.0D	UFS	58	117
DEC Alpha 500/266 MHz	DEC Unix 4.0B	UFS	49	110
IBM 43P/333 MHz	AIX 4.2.1	JFS	103	105
IBM F50/166 MHz	AIX 4.2	JFS	101	105
AMD K6/200 MHz	FreeBSD 2.2.5	UFS	38	90
IBM 43P/200 MHz	AIX 4.1.5	JFS	73	77

on TCP and RPC but not as well on pipe. AIX does surprisingly well in latency, even though its memory performance hurts it terribly on the bandwidth measures.

File Operations

Although more removed from the OS core and dominated by disk speeds, the time for filesystem creations and deletions can also have important effects on application performance. As shown in Table 7, this seems to depend strongly on filesystem type and only weakly on OS vendor. For example, all three of the UFS implementations seem to perform quite similarly. On the other hand, the Linux EXT2 filesystem is fully asynchronous. Unlike other filesystems, it never forces updates to disk for the sake of guaranteeing consistency in the event of crashes. This has a huge effect in file operations. The AIX journalling JFS performs worse that UFS at deletes, but better at creates.

Conclusions

The results presented in this paper looked at system performance in three main areas: CPU speed, memory system performance, and a set of OS primitives. None of these tests showed any conclusive standout machines: it seems there are a lot of good hardware platforms to choose from.

In CPU benchmarking, we found that SPECint95 is a good model for BABAR reconstruction performance, but there are still discrepancies from perfect scaling (∼30%) that are not really understood. Furthermore, we found that mastering compiler optimization is essential on RISC platforms. This can account for as much as ×5 in speed.

Memory bandwidth and latency are getting attention from vendors, but at present there is a wide spread in memory speeds available. The IBM 43P line of workstations stood out as particularly poor in memory bandwidth. As a practical application, this weakness showed up unmistakably in the network performance measurements in [4].

Leadership in the OS primitives was quite evenly distributed among the different platforms, with no single platform performing well on all tests. Interestingly enough, the freeware operating systems (Linux and FreeBSD) performed as well as any of the other systems, indicating the high quality of freeware available today. It is not clear which of the primitives tested are most relevant for the type of workload to be found in the BABAR Online Farm, so it is hard to draw any definitve conclusions from the OS benchmarking.

The Future promises to be exciting, with CPU speeds still growing exponentially every year, and all of the CPU vendors try to keep up with (or surpass) Intel's Merced. This is the promise of open systems. If one can keep application software portable to a range of different hardware, then one will be able to reap the rewards of this competition.

References

[1] BABAR Technical Design Report, SLAC-R-457 (1995).
See also `http://www.slac.stanford.edu/BFROOT/doc/TDR/`.

[2] I. Scott *et al.*, "The BABAR Data Acquisition System," CHEP 98 #19, submitted to these proceedings.

[3] T. Glanzman *et al.*, "The BABAR Prompt Reconstruction System," CHEP 98 #52, submitted to these proceedings.

[4] T.J. Pavel *et al.*, "Network Performance Testing for the BABAR Event Builder," CHEP 98 #31, submitted to these proceedings.

[5] SPEC CPU95 benchmark suite, `http://www.specbench.org/osg/cpu95/`.

[6] L. Gwennap, "Comparing High-Performance Microprocessor Designs," Microprocessor Forum Seminar, San Jose, CA, Oct. 1997.

[7] J.D. McCalpin, "STREAM: Sustainable Memory Bandwidth in Recent and Current High Performance Computers," a continually updated technical report (`http://www.cs.virginia.edu/stream/`).
See also `http://reality.sgi.com/mccalpin/papers/bandwidth/bandwidth.html`.

[8] J.K. Ousterhout, "Why Aren't Operating Systems Getting Faster as Fast as Hardware?" *Proc. 1990 Summer USENIX Conf.*, Anaheim, CA, June 1990, pp. 247–256.

[9] Larry McVoy and Carl Staelin, "Lmbench: Portable Tools for Performance Analysis," *Proc. 1996 Winter USENIX Conf.*, San Diego, CA, January 1996, pp. 279–295.
See also `http://www.bitmover.com/lmbench/lmbench.html`.

[10] A.B. Brown and M.I. Seltzer, "Operating System Benchmarking in the Wake of *Lmbench*: A Case Study of the Performance of NetBSD on the Intel x86 Architecture," *Proc. 1997 Sigmetrics Conf.*, Seattle, WA, June 1997.
See also `http://www.eecs.harvard.edu/~vino/perf/hbench/`.

[11] K. Lai and M. Baker, "A Performance Comparison of UNIX Operating Systems on the Pentium," *Proc. 1996 USENIX Conf.*, San Diego, CA, January 1996.
See also `http://gunpowder.Stanford.EDU/~mgbaker/publications/usenix96.bench.ps`.