Computational Chemistry on Commodity-Type Computers

Marc C. Nicklaus,*,† Robert W. Williams,‡ Bruno Bienfait,† Eric S. Billings,§ and Milan Hodošček⊥

Laboratory of Medicinal Chemistry, Division of Basic Sciences, National Cancer Institute, National Institutes of Health (NIH), Bethesda, Maryland 20892-4255, Department of Biochemistry and Molecular Biology, Uniformed Services University of Health Sciences, 4301 Jones Bridge Road, Bethesda, Maryland 20814-4799, Laboratory of Biophysical Chemistry, National Heart, Lung and Blood Institute, NIH, Bethesda, Maryland 20892, and National Institute of Chemistry, Hajdrihova 19, Ljubljana, Slovenia

Received May 19, 1998

A number of inexpensive computers were benchmarked with the ab initio program Gaussian 94, using both small standard test jobs and larger density functional (DFT) calculations. Several varieties of Pentium (x86) and Alpha CPU based systems were tested. Most of them were running under the open source code operating system Linux. They were compared with several workstations and supercomputers. The most powerful of today's commodity-type processors surpassed current supercomputers in speed. The choice of compilers and compilation options was often found to have a larger influence on job CPU times than details of the hardware. Especially on the x86 type machines, the jobs always ran faster the less memory (RAM) they were given. The fastest machine on a per-CPU basis was an Alpha/Linux system. For the DFT calculation, it was close to twice as fast as a Cray J90 supercomputer.

INTRODUCTION

Computational chemistry calculations at the upper end of the resource requirement range, such as ab initio computations and molecular dynamics simulations, once required supercomputers for any project of more than trivial size. Such machines were—and still are—very expensive, their prices ranging from several $100 000 to many millions of dollars. They are therefore out of the reach of even entire departments, let alone small workgroups or individual researchers. Typically purchased by the organization as a whole (government agency, company, university etc.), the one, or very few, supercomputer(s) available has to be shared among many users.

Beginning about 10−15 years ago, the so-called workstations, mostly based on the Unix operating system, started offering considerable computing power to much smaller entities of researchers, such as small workgroups. These machines, falling in the price range of about $10 000 to approximately $100 000, have become very popular in computational chemistry sections of, say, academic chemistry departments or industrial research and development labs. Parallel to the development in the computer hardware sector as a whole, their performance/price ratio has been continuously increasing, albeit at a lower rate than for the subsequent type of computers.

The past few years have seen the development of more and more powerful commodity-type processors. At more or less constant prices for the top-of-the-line model, both their clock rates and their architecture-based computation power have been increasing rapidly. These developments, coupled with the general price decline for computer components, have made available machines which, for a few thousand dollars, can perform heavy-duty computational chemistry calculations previously necessitating a supercomputer or a top-of-the-line workstation. The most widely used commodity-type processors are the PowerPC chips designed by IBM and Motorola and widely used in the Apple Macintosh computers; the various processors in the x86 family (such as the Pentium, Pentium Pro, Pentium II etc.), designed by Intel and their clones made by companies such as AMD and Cyrix; and the Alpha CPUs, designed by Digital Equipment Corp. (DEC), licensed out for manufacturing to, e.g., Intel and Samsung, and used both by DEC in their medium and upper tier computers and by various third-party system integrators offering generally smaller, single-processor, systems.

Since no computer is very useful without both an operating system and application software, the availability as well as affordability of either type of software is an important aspect in the use of commodity-type computers. A range of commercial operating systems is available for this class of machines. Some, such as Mac OS, DOS, Windows 95, or OS/2, are more or less confined to one type of hardware. Others, such as Windows NT, NextStep, and Solaris, are available for several processor families. The latter is one example of the several commercial versions of the Unix operating system that are available for the x86 type of processors.

Unix is by design multitasking and multiuser capable, adaptable to a wide range of hardware, and generally considered to be very stable. This, and the fact that some variant of Unix is nowadays running on virtually any workstation and on many supercomputers, traditionally the tools of the computational chemist, makes this operating system of particular interest to this user group.

* To whom correspondence should be addressed.
† NCI, NIH, Bethesda.
‡ USUHS, Bethesda.
§ NHLBI, NIH, Bethesda.
⊥ NHLBI, NIH, Bethesda, and NIC, Slovenia.

S0095-2338(98)00092-4
Published on Web 09/01/1998

In recent years, several free varieties of Unix have become available, such as FreeBSD and Linux.[1] The latter one is probably the most well-known and most rapidly developing open source code Unix variant and is available, among others, for both x86 and Alpha CPU based computers. Compilers—many of them free—are also available, and therefore any computational chemistry program for which the source code is available stands a good chance of being portable to such a system. The availability of system software at no, or very little, cost (if one buys one of the repackaged versions of, e.g., Linux) thus further reduces the cost of commodity-type computing to the computational chemist.

These developments have suddenly put supercomputer power into the reach, and onto the desktops, of individual researchers. (As an alternative, such machines can be combined to form a parallel supercomputer of the Beowulf class,[2] at a price point still an order of magnitude below equally powerful "traditional" high-end supercomputers.)

Among the first ones to bring this fact to the attention of the chemical community were Tirado-Rives and Jorgensen.[3] More recently, Yu and Yu[4] benchmarked a number of computers, including commodity-type systems with up to 200 MHz Pentium Pro CPUs, using Gaussian 94 as the test program. We have expanded the range of commodity-type machines tested by these authors to some of the most recent and powerful processors. Several more benchmarks that might be interesting to the computational chemist are available on the World Wide Web (WWW).[5]

Why then is not everyone switching entirely to such computers? First, some applications require the special characteristics of traditional supercomputers, such as very high communication speeds between multiple processors, and very high memory access speeds. However, the commodity-type computers are continuously encroaching upon the range of problems that truly necessitate such resources, and thus the advantage of the high-end systems is rapidly eroding.

Second, if one opts to use one of the free operating systems that provides traditional Unix power at very low cost, the lack of traditional support by a vendor is often seen as a risk factor. While this is to some extent true, support provided by an international users' and developers' community has been found by many to be fast, unbureaucratic, and very honest and seen as an advantage rather than a drawback. Also, the very rapid development of these open operating systems would seem to hold more promise than threat. Still, maturity and stability problems are undeniable on certain platforms, as, e.g., for the Alpha CPU version of Linux, and it pays to investigate the current situation before basing an important project on one of these systems. For those users who do not want to venture out into seemingly less charted waters, the commercial operating systems mentioned before often offer similar performance as well as vendor support.

Third, and perhaps most importantly, the unavailability of many of the widely used molecular modeling and computational chemistry applications on these platforms—here defined as typically a Linux system on a Pentium-class or Alpha CPU based machine—deters many computational chemists from exploring this option. To some extent, this used to be a "vicious cycle" of mutually reinforcing lack of demand and lack of availability. However, this problem appears to be diminishing as installed base and user demand are reaching a critical mass. The availability, for these platforms, of computational chemistry programs, especially if originating from academic sites, is already surprisingly large.[6] It is mostly the proprietary graphical user interfaces which have not been ported (yet).

Several of the widely used computational chemistry programs whose runs typically consume large amounts of CPU cycles are available on these platforms. The performance of one of them—the ab initio program Gaussian 94[7] (G94)—is compared on a wide spectrum of machines in this paper.

Many high-end systems, and especially supercomputers, are multiprocessor machines nowadays. Gaussian 94 is available in a parallel version for some of those machines (for an up-to-date list see http://www.gaussian.com/parallel.htm), but it is unavailable in parallel for any Linux based system such as the mentioned Beowulf class machines. However, these machines can offer quasi-parallel work for all but those projects which consist in one, or a very few, large G94 runs if one simply buys a larger number of them, at a still very moderate cost when compared to a supercomputer. We therefore felt that the least biased comparison would be based on single-CPU timings for all systems including the multiprocessor machines.

In addition to reporting the benchmarking results themselves, we have tried to investigate the factors that influence the performance of a program such as G94. These range from the type of CPU, the amounts of main and/or cache memory, through software versions, to compiler options used. (Gaussian, as other computational chemistry programs, is shipped for most platforms as source code, and therefore the user has the possibility to influence the compilation, e.g., by increasing the optimization level.) Obviously, the combinatorial nature of the entirety of factors potentially influencing program performance precludes any comprehensive analysis. We have instead tried to highlight certain areas that might be relevant for the typical user.

METHODS

Hardware. The spectrum of machines included in the comparison ranged from two Cray Research supercomputers, workstations or servers from Digital Equipment Corp. (DEC), International Business Machines (IBM), and Silicon Graphics, Inc. (SGI), to a variety of commodity-type systems based on both Pentium Pro (P6) and Pentium II (PII) processors from Intel Corp., operated in motherboards from various manufacturers, and two Alpha CPU based systems. All but one system ran under some variant of the Unix operating system. One system running under Windows NT 4.0 is included for comparison purposes.

As mentioned before, a great variety of factors can influence program execution speeds. It is therefore not possible to report our results simply as a comparison of different machines (or processor speeds, or compilers used, etc.) We hence refer to the individual sets of program runs as "scenarios" to indicate this fact. The scenarios analyzed are listed in Table 1. They are very roughly sorted by (decreasing) price for the hardware.

Both the CrayYMP and the CrayJ90 are multiprocessor machines and the revision of G94 available on either system was the parallel version of the program. To perform true one-processor runs on the CrayJ90, we forced G94 into

Table 1. Scenarios Tested

scenario	hardware	memory	operating system	compiler	compiler options, runtime options	G94 rev
CrayYMP	Cray Y-MP 8/8128 Model D 8 CPUs	128 MW (1 GB)	Unicos 8.0.4	(precompiled)[a]	N/A	C.3
CrayJ90	Cray J932se 16 CPUs	1024 MW (8 GB)	Unicos 9.3	(precompiled)[a]	N/A	D.4
Origin2000	SGI/Cray Origin 2000 8 CPUs R10000 195 MHz	3 GB	IRIX 6.4	f77 (64 bit)	(std)	D.3
RS6000[b]	IBM RS/6000-590	(unknown)	AIX	(unknown)	N/A	D.3
Alpha2100	DEC AlphaServer 2100 4/275 4 CPUs 21064 (EV4), 275 MHz	256 MB	OSF/1 3.2c	f77 v. 3.8	(std)	C.3
Indigo2	SGI Indigo2 1 CPU R10000 195 MHz	64 MB	IRIX 6.2	f77 (64 bit)	(std)	E.1
Alpha500MHz[c]	Deskstation Ruffian, RPX164-2 motherbrd 1 CPU Alpha 21164 (EV 56), 500 MHz	256 MB SDRAM 2 MB L3 Cache	RedHat Linux 4.2, kernel 2.0.30	compiled on DEC OSF/1 v. 3.2c, using f77 v. 3.8	makefile for alpha-osf1 + "-tune ev5". DEC DXML replaced by K. Goto's assembler DGEMM math subroutine, version 980415	B.3
Alpha533MHz[d]	AlphaPC 164LX motherbrd 1 CPU Alpha 21164A (EV 56), 533 MHz	128 MB SDRAM 2 MB L3 Cache	RedHat Linux 5.0, kernel 2.0.30	compiled on DEC OSF/1 v. 3.2c, using f77 v. 3.8	makefile for alpha-osf1 + "-tune ev5". DEC DXML replaced by K. Goto's assembler DGEMM math subroutine, version 980415	B.3
PII/400/f2c/Opt4[e]	Pentium II 400 MHz 1 CPU memory bus 100 MHz	128 MB SDRAM ECC, 100 MHz	RedHat Linux 5.0, kernel 2.0.33	f2c/gcc[f]	-O3 -m486 -funroll-loops -fomit-frame-pointer -mailgn-double -malign-loops=2 -malign-jumps=2 -malign-functions=2	E.1
PII/400/f2c[e]	Pentium II 400 MHz 1 CPU memory bus 100 MHz	128 MB SDRAM ECC, 100 MHz	RedHat Linux 5.0, kernel 2.0.33	f2c/gcc	(std)	E.1
PII/337/f2c/Opt4[g]	Pentium II 300 MHz, 2 CPUs overclocked to 337.5 MHz, bus to 75 MHz	256 MB SDRAM 512 kB L2 Cache	Debian Linux 2.0, kernel 2.1.87	f2c/gcc	-O3 -m486 -funroll-loops -fomit-frame-pointer -mailgn-double -malign-loops=2 -malign-jumps=2 -malign-functions=2	E.1
PII/337/f2c[g]	Pentium II 300 MHz, 2 CPUs overclocked to 337.5 MHz, bus to 75 MHz	256 MB SDRAM 512 kB L2 Cache	Debian Linux 2.0, kernel 2.1.87	f2c/gcc	(std)	E.1
PII/300/f2c/Opt4[h]	Pentium II 300 MHz 1 CPU SuperMicro P6DLF motherbrd	128 MB SDRAM 512 kB L2 Cache	RedHat Linux 5.0, kernel 2.0.31	f2c/gcc	-O3 -m486 -funroll-loops -fomit-frame-pointer -mailgn-double -malign-loops=2 -malign-jumps=2 -malign-functions=2	E.1
PII/300/f2c[h]	Pentium II 300 MHz 1 CPU SuperMicro P6DLF motherbrd	128 MB SDRAM 512 kB L2 Cache	RedHat Linux 5.0, kernel 2.0.31	f2c/gcc	(std)	E.1
PII/300/NT	Pentium II 300 MHz 1 CPU	64 MB	Windows NT 4.0	(no compilation)	(program shipped as binary executable for this platform)	E.1
P6/AMI/g77/Opt4[i]	Pentium Pro 200 MHz 2 CPUs AMI motherbrd	128 MB EDO RAM 512 kB cache	Debian Linux, kernel 2.0.29	g77 v. 0.5.20	-O3 -m486 -funroll-loops -fomit-frame-pointer -malign-double -malign-loops=2 -malign-jumps=2 -malign-functions=2	D.1
P6/ASUS/g77/Opt4[i]	Pentium Pro 200 MHz 2 CPUs ASUS motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux, kernel 2.0.29	g77 v. 0.5.20	-O3 -m486 -funroll-loops -fomit-frame-pointer -malign-double -malign-loops=2 -malign-jumps=2 -malign-functions=2	D.1
P6/Intel/g77/Opt4[i]	Pentium Pro 200 MHz 2 CPUs Intel motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux, kernel 2.0.29	g77 v. 0.5.20	-O3 -m486 -funroll-loops -fomit-frame-pointer -malign-double -malign-loops=2 -malign-jumps=2 -malign-functions=2	D.1
P6/ASUS2/g77/Opt1	Pentium Pro 200 MHz 1 CPU ASUS motherbrd	64 MB EDO RAM 256 kB cache	Debian Linux, kernel 2.0.29	g77 v. 0.5.20	-O2 -fno-f2c -fno-backslash -malign-double	D.1

Table 1 (Continued)

scenario	hardware	memory	operating system	compiler	compiler options, runtime options	G94 rev
P6/g77/Opt4[i]	Pentium Pro 200 MHz 2 CPUs ASUS motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux 1.2, kernel 2.0.27	g77 v. 0.5.20	-O3 -m486 -funroll-loops -fomit-frame-pointer -malign-double -malign-loops=2 -malign-jumps=2 -malign-functions=2	D.1
P6/g77/Opt2[i]	Pentium Pro 200 MHz 2 CPUs ASUS motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux 1.2, kernel 2.0.27	g77 v. 0.5.20	-O2 -fno-f2c -fno-backslash -malign-double -malign-loops=2 -malign-jumps=2 -malign-functions=2	D.1
P6/g77/Opt1[i]	Pentium Pro 200 MHz 2 CPUs ASUS motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux 1.2, kernel 2.0.27	g77 v. 0.5.20	-O2 -fno-f2c -fno-backslash -malign-double	D.1
P6/f2c/Opt4[i]	Pentium Pro 200 MHz 2 CPUs ASUS motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux 1.2, kernel 2.0.27	f2c/gcc	-O3 -m486 -funroll-loops -fomit-frame-pointer -malign-double -malign-loops=2 -malign-jumps=2 -malign-functions=2	E.1
P6/f2c/Opt3[i]	Pentium Pro 200 MHz 2 CPUs ASUS motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux 1.2, kernel 2.0.27	f2c/gcc	-O2 -m486 -malign-double -malign-loops=2 -malign-jumps=2 -malign-functions=2	E.1
P6/f2c[i]	Pentium Pro 200 MHz 2 CPUs ASUS motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux 1.2, kernel 2.0.27	f2c/gcc	(std)	E.1
P6/AMI/f2c/256k[i]	Pentium Pro 200 MHz 2 CPUs AMI motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux, kernel 2.0.29	f2c/gcc	(std)	E.1
P6/AMI/f2c/512k[i]	Pentium Pro 200 MHz 2 CPUs AMI motherbrd	128 MB EDO RAM 512 kB cache	Debian Linux, kernel 2.0.29	f2c/gcc	(std)	E.1
P6/ASUS/f2c/256k[i]	Pentium Pro 200 MHz 2 CPUs ASUS motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux, kernel 2.0.29	f2c/gcc	(std)	E.1
P6/ASUS/f2c/512k[i]	Pentium Pro 200 MHz 2 CPUs ASUS motherbrd	128 MB EDO RAM 512 kB cache	Debian Linux, kernel 2.0.29	f2c/gcc	(std)	E.1
P6/SuMi/f2c/256k[i]	Pentium Pro 200 MHz 2 CPUs SuperMicro motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux, kernel 2.0.29	f2c/gcc	(std)	E.1
P6/SuMi/f2c/512k[i]	Pentium Pro 200 MHz 2 CPUs SuperMicro motherbrd	128 MB EDO RAM 512 kB cache	Debian Linux, kernel 2.0.29	f2c/gcc	(std)	E.1
P6/Intel/f2c/256k[i]	Pentium Pro 200 MHz 2 CPUs Intel motherbrd	128 MB EDO RAM 256 kB cache	Debian Linux, kernel 2.0.29	f2c/gcc	(std)	E.1
P6/Intel/f2c/512k[i]	Pentium Pro 200 MHz 2 CPUs Intel motherbrd	128 MB EDO RAM 512 kB cache	Debian Linux, kernel 2.0.29	f2c/gcc	(std)	E.1

[a] As installed by the Scientific Applications Support Group of the NCI Frederick Biomedical Supercomputing Center. [b] Gaussian, Inc., reference system. [c] Purchased from DeskStation Technology, Inc. (http://www.deskstation.com/). [d] Purchased from DCG Computers, Inc. (http://www.dcginc.com/). [e] Purchased from SW Technology (http://www.swt.com/). [f] All scenarios using f2c/gcc compilation used the same executables, translated from Fortran to C with f2c v. 19960717, and then compiled with gcc v. 2.7.2.2.f.2; the C library version was libc.so.5. [g] Purchased from Softhard Systems (http://www.softhardsystems.com/). [h] Motherboard and CPU purchased from TC Computers (http://www.tccomputers.com). [i] Purchased from PC Importers, Inc. (http://www.pcimporters.com).

single-CPU mode for the (longer) density functional theory (DFT) runs in one of the series. This can be achieved by using the directive "%Nproc=1". Very little difference was found in the job CPU time for these runs when compared with runs not using this directive, which then defaulted to using four processors in parallel operation. (Of course, the waiting time for the user, the so-called "wall time", differed substantially.) Because of this, the input files for the test jobs were not modified to contain the "%Nproc=1" directive. Despite this, the output indicated that both Cray systems dynamically reduced the number of processors to between 1 and 4 (mostly to 1), presumably because the small size of the test jobs renders parallel operation uneconomical.

In all cases, we report, and compare, the "Job cpu time", as written by the program to its output file. This time is the aggregate CPU time of all processors used for a job. It also includes any system overhead that may have cost additional CPU time. For the Cray scenarios, we also report the "User CPU Time" as printed in the batch queue output (for those jobs that had to be run in batch mode for reason of their size), which essentially subtracts the system CPU time, as

well as the "Elapsed Time" the user had to wait for the job to finish.

In the majority of the P6 scenarios reported, the machine was a Dual CPU system running under Linux. However, as mentioned, there is no parallel version available on this platform at the moment, and thus all run times reported are the results of single-CPU runs on these dual-processor machines. Likewise, the Origin2000 and the Alpha2100 were an eight- and a four-processor machine, respectively, but only single-CPU runs were conducted on either system.

Both because no 333 MHz system was available at the scheduled time of the test, and also in order to investigate the principal feasibility of motherboard overclocking for G94, a "337.5 MHz" Pentium II was created by raising the system bus speed from 66 to 75 MHz while leaving the clock divider ratio for the CPU constant. The scenarios involving this machine are denoted "PII/337..." This overclocked system proved to be stable for all jobs run on it. The various "P6/ASUS/..." and the P6/ASUS2/g77/Opt1 scenarios involved different machines, although with motherboards from the same manufacturer. This is indicated by the "2" in "ASUS2" in the latter scenario's name.

For most of the machines (and all of the P6 and PII based systems), the tests were performed with the system being otherwise quiescent. This was not possible for the two Cray supercomputers, which are shared with many users. This apparent, potential, skewing of the results, however, realistically reflects the situation a user will typically encounter, where he or she will have no control over load and other parameters of a heavily shared supercomputer, whereas the commodity-type machine is quite likely to be at the user's sole disposal and under his or her full control.

Included in the comparison is Gaussian, Inc.'s "reference system", an IBM RS/6000-590 computer. The program's distribution includes the output files of the standard Gaussian test jobs (see below) run on this machine. These files, containing the CPU times used, allowed us to include this system in the comparison.

One of the main characteristics, apart from the CPU clock rate, that differentiates the commodity-type computers included in this comparison, the Linux/x86 P6 and PII and the Linux/Alpha systems, is the memory access speed. The 128 bit physical bus width on both the Alpha500MHz's RPX164-2 and the Alpha533MHz's PC164LX motherboards, using the DEC 21174 (Pyxis) chip set, produces a peak bandwidth of 1056 MB/s. The PII systems' peak bandwidth when run with the Synchronous DRAM (SDRAM) memory type enabled by the new chip sets such as Intel 440LX is 528 MB/s, whereas the P6 motherboards equipped with the Intel 440FX (Natoma) chip set, which limits one to EDO RAM, reduces the peak bandwidth by half again, yielding a maximum of 264 MB/s. The very recent systems with the 350 and 400 MHz PII CPUs run on motherboards equipped with the 440BX chip sets, which has a system bus speed of 100 MHz, would then allow peak bandwidths of 792 MB/s.

Software. Gaussian 94 Compilation. For the majority of the systems analyzed, Gaussian 94 was compiled from source code. Both the standard installation/compilation scripts as provided by Gaussian, Inc. were used as well as modified scripts (MAKEFILE) in which higher degrees of optimization during compilation was specified. On the Cray systems, the versions of Gaussian 94 were used as preinstalled by the Scientific Applications Support section of the National Cancer Institute's (NCI) Frederick Biomedical Supercomputing Center.

For most of the scenarios, in particular on the commodity-type machines, Revision E.1 of Gaussian 94 was used. In some of the other scenarios, slightly older revisions of G94 were available on the system, ranging from B.3 to D.3. Since typically new features (and bug fixes) are introduced in new revisions of Gaussian, whereas most of the previous code is left unchanged, we assume that these tests of basic Gaussian capabilities, which, after all, ran on all revisions encountered, were not affected in speed by the variation of revisions.

For those machines/scenarios for which executables are binary compatible, i.e., in particular, the x86 based Linux systems, the program was not usually recompiled in place, but the compiled version was simply transferred from one computer to the other. The Linux compilers f2c and gcc are fairly generic, i.e., written for an entire family of processors, and would thus not have produced machine code any different from one x86/Linux system to the other (even different f2c/gcc versions produced code of virtually identical speed [see Results section]). The central "compilation scenarios", in this sense, for most of the x86/Linux systems included in this study were the P6/f2c scenarios (for the different levels of optimization).

The Linux/Alpha platform, represented here by the Alpha500MHz and Alpha533MHz scenarios, is an exception insofar as Gaussian 94 *could not* be compiled on it natively. The Alpha versions of the Linux compilers and especially the linker (ld) are not as mature and stable in their current development stage as their x86 counterparts, and despite numerous attempts it was not possible to compile G94 directly on the Linux/Alpha system. However, it proved technically possible to compile G94 on the DEC Alpha2100 for execution on Linux/Alpha systems after replacing DEC's proprietary Digital Extended Math Library (DXML) with the DGEMM math subroutine coded in assembler by Kazushige Goto,[8] Kanagawa, Japan. (More details are given in the Discussion section.) Using DEC's loader and several standard libraries, Gaussian 94 was able to successfully complete all jobs on the Alpha500MHz and Alpha533MHz. (It is important that the "a.out" option is switched on in the Linux kernel in order to run executables compiled in the COFF format under Digital Unix.) Of several revisions tested, only Revision B.3 could be successfully compiled for Linux/Alpha because it does not require threads support.

For all G94 runs, the output files were scanned to ensure that the results, in particular the calculated energy values, were identical for all machines and scenarios. In most cases, the energies (in hartrees) differed from each other in none of the typically 11 decimals, confirming the correctness of the run at the 10^{-8} kcal/mol level. A few cases, however, showed deviations on the order several kcal/mol for specific test jobs, which is totally unacceptable for ab initio calculations. This thus indicated a potentially serious problem with the compiled executable. These cases will be detailed below.

Test Jobs. Gaussian, Inc. provides about 300 test jobs with the program. As a test for correct compilation and operation of the program, the User's Reference manual recommends running at least a subset of seven of these jobs, which cover a range of Gaussian 94 capabilities. They are test jobs no. 1, 28, 94, 155, 194, 296, and 302. These seven

Table 2. Job CPU Times of Gaussian 94 Test Jobs 1, 28, 94, 155, 194, 296, and 302 Plus Total Time

	test (all times in seconds)							
scenario	1[a]	28[b]	94[c]	155[d]	194[e]	296[f]	302[g]	total
CrayYMP	7.6	21.6	74.3	353.6	4.1	1228.1	49.5	1738.8
("user CPU time":)				(324.9)		(1063.6)		
CrayJ90	13.2	31.5	99.3	620.7	148.8[h]	1935.5	140.5	2989.5
("user CPU time":)				(568.5)	(93.1)	(1735.3)		
Origin2000	4.1	10.6	34.8	131.0	43.2	482.8	49.3	755.8
RS6000	4.0	13.8	55.4	201.5	53.1	749.5	66.7	1144.0
Alpha2100	11.3	26.5	32.7	314.1	108.7	1053.7	111.3	1658.3
Indigo2	5.1	12.4	40.2	162.8	72.5	572.8	57.1	922.9
Alpha500MHz	2.7	6.8	25.3	105.2	42.8	365.8	38.9	587.5
@8MW	3.7	9.3	32.0	115.3	43.3	419.6	51.5	674.7
Alpha533MHz	2.8	7.4	25.5	111.1	52.8	361.8	36.9	598.3
@8MW	4.6	9.9	31.5	123.4	52.7	416.2	49.9	688.2
PII/400/f2c/Opt4	2.7	7.9	27.4	144.2	100.1	486.8	36.6	805.7
PII/400/f2c	2.7	8.5	29.6	193.7	106.1	605.9	42.4	988.9
PII/337/f2c/Opt4	3.1	9.1	31.4	170.1	121.6	571.1	42.6	949.0
PII/337/f2c	3.2	10.2	34.9	233.7	133.6	732.3	50.9	1198.8
PII/300/f2c/Opt4	3.8	11.1	37.9	210.8	151.1	674.9	49.0	1138.6
PII/300/f2c	4.1	11.3	40.9	230.8	156.4	787.2	52.2	1282.9
PII/300/NT	24.0	38.0	45.0	247.0	171.0	700.0	58.0	1283.0
P6/AMI/g77/Opt4	5.5	15.1	55.5	252.2	122.9[i]	806.3	67.3	1324.8
P6/ASUS/g77/Opt4	5.9	15.8	53.4	271.8	127.7[i]	852.8	73.3	1400.7
P6/Intel/g77/Opt4	6.5	19.0	64.0	318.4	190.2[i]	1009.3	83.9	1691.3
P6/ASUS2/g77/Opt1	5.5	15.4	55.5	262.6	165.1	871.8	71.9	1447.8
P6/g77/Opt4	5.7	16.0	53.3	267.8	119.2[i]	858.2	72.3	1392.5
P6/g77/Opt2	5.7	15.7	54.1	272.0	156.8[i]	885.4	72.9	1462.6
P6/g77/Opt1	5.8	15.5	53.6	274.2	156.1[i]	881.8	73.1	1460.1
P6/f2c/Opt4	5.9	15.7	57.0	298.3	178.3	969.9	75.1	1600.2
@8MW	8.6	21.7	72.9	327.4	180.9	1125.6	100.1	1837.2
P6/f2c/Opt3	5.9	16.0	56.5	304.7	180.2	969.6	74.9	1607.8
P6/f2c	7.8	20.3	69.3	468.2	196.8	1392.1	97.6	2252.1
P6/AMI/f2c/256k	6.5	19.0	65.7	462.4	191.5	1356.7	95.7	2191.0
P6/AMI/f2c/512k	6.7	19.0	68.4	456.7	199.8	1344.8	93.9	2189.3
								speed increase: 0.08%
P6/ASUS/f2c/256k	7.1	20.2	71.2	480.7	213.9	1440.6	102.0	2335.7
P6/ASUS/f2c/512k	6.5	19.7	70.4	459.5	189.9	1360.9	97.5	2204.4
								speed increase: 6.0%
P6/SuMi/f2c/256k	6.6	20.9	73.5	490.3	223.6	1449.2	103.1	2367.2
P6/SuMi/f2c/512k	6.4	19.8	68.7	462.1	206.3	1391.3	100.0	2254.6
								speed increase: 5.0%
P6/Intel/f2c/256k	7.8	23.2	80.0	509.9	257.5	1537.7	108.6	2524.7
P6/Intel/f2c/512k					(not completed)			

[a] Route cards and comments. test001: #P TEST STO-3G COMPLEX pop=full scf=conventional; SINGLET DELTA STO-3G//STO-3G DIOXYGEN. [b] test028: #p rhf/3-21g freq=anal; water freq//hf/3-21g freq=anal explicit (compare test 27). [c] test094: #p rmp2/sto-3g opt=calcfc test geom=modela; Water MP2 Opt=CalcFC. [d] test155: #p 6-31+G* freq rcis(full,mo) test; 1-B3u Ethene D2h. [e] test194: #p uqcisd(tq)/gen test guess(mix) symm=noscf iop1=synch; UQCISD(TQ) water at 1.5 Re with [4s2p1d|2s1p] Dunning basis. [f] test296: #p G1 geom=modela test; Water G1 from standard model. [g] test302: #p hf/3-21g opt=qst2 test; reactant SiH2 +H2 → SiH4 ts with constrained transition vector. (The final energy of test302 on CrayYMP, DEC Alpha2100, and the Alpha/Linux systems differred from all other systems by various amounts of −0.00011 kcal/mol or less.) [h] Job did not complete with 4MW of RAM; it was run in 8MW instead. [i] Results (energies) for this job differred from the other platforms/compilers: Last line of a "grep '*E*(CORR)'" gave: *E*(CORR)= −0.76065511656E+02 here, *E*(CORR) = −0.76057707310D+02 on all other platforms; ΔE = 4.9 kcal/mol. E(PMP3) = −0.76065682529E+02 here, *E*(PMP3) = −0.76100375374D+02 on all other platforms; ΔE = 21.77 kcal/mol. QCISD (TQ) = −0.76077023117E+02 here, QCISD (TQ) = −0.76070478817D+02 on all other platforms; ΔE = 4.1 kcal/mol.

test jobs were used for the first set of benchmark values presented in this study. They operate mostly on very small molecules, such as water, and span a wide range of methods, ranging from low to very high (including post-Hartree−Fock) levels of theory. The route cards and titles of each of them, which give an indication of what kind of calculation is performed, are listed in the footnotes of Table 2. These jobs form a convenient benchmark because their run times add up to under 1 h on this class of machines (in contrast to much larger "real-life" jobs in G94 which may take days or weeks of CPU time). They also have been used in other benchmarking work of G94.[9]

On all systems under our control, the test jobs were run without any Default.Route file present (which lets the installer of G94 set default limits for resources such as memory, disk space, number of CPUs, etc.). Absent this file, Gaussian 94 defaults to 4 000 000 words of memory, which was hence the amount used for all test jobs except where noted. G94 on the Cray systems appears to change the amount of memory used depending on the job submitted; the output listed maximal memory sizes used between 2.7 and 4 million words on the CrayJ90 and between 3.2 and 15.6 million words on the CrayYMP.

DFT Job. In the past few years, Density Functional Theory methods have become increasingly popular in quantum chemical calculations. The better ones among them are considered to offer substantially greater accuracy than Hartree−Fock (HF) methods while increasing the computation cost far less than other higher-accuracy methods such as Møller−Plesset Second Order (MP2) perturbation theory

or methods even beyond that. For this reason, and because none of the seven test jobs uses the DFT method, a DFT calculation of a molecule taken from one of our ongoing projects[10] was chosen as a second benchmark job in this study.

This job was a single-point energy calculation of a nucleoside analogue (19 heavy atoms, 15 hydrogens), utilizing the B3LYP method[11−13] with the 6-31G(d) basis set (see Chart 1), yielding a total of 315 basis functions. Because DFT methods perform numerical integration of the integrals vs the analytical integration used by, e.g., HF and MP2, DFT calculations tend to have somewhat less numerical stability than other methods. In our case, this led to the effect that iterative runs of the same input file, reading the wave function each time from the checkpoint file of the previous run, produced slightly different energies (typically differing by less than 10^{-4} kcal/mol) and, more important for the benchmarking, took a different number of cycles to achieve convergence of the wave function. Since each cycle, for a given molecule, takes about a constant amount of CPU time, not controlling this parameter would render the benchmark numbers meaningless. It was therefore made sure that for each job in each of the DFT series that were run (see below), a standard checkpoint file was copied to the actual checkpoint file read by that very run (after an initial, nonbenchmarked run to generate this standard file was performed at the beginning of each series). Utilizing this approach, the DFT calculations reported in the benchmarks achieved convergence of the wave function within two cycles in all cases.

Gaussian 94 allows the user to specify (through the "%Mem=..." command) how much memory (RAM) the program is allowed to use during its run. Initial runs had hinted at an unexpected influence of this parameter on the performance of the program. It was therefore varied in a systematic way for the DFT job on each platform. Memory amounts of 0.25, 0.5, 0.75, and 0.9 Megawords (MW) did not allow this job to run to completion in any case. One MW proved to be the minimum that allowed the job to finish successfully on some of the platforms. On the other platforms, a further increase in memory—to a quite various degree—was necessary to prevent the job from crashing. The memory was then increased in steps to up to 16 MW or an amount that prevented successful completion at the high end of the scale, whichever came first. (One MW is defined here in the usual computer science sense, i.e., the "M" represents 2^{20} and not 1 000 000. A "Word" as a counting unit in Gaussian is always defined as being 8 bytes long; therefore 1MW corresponds to 8MB, and 16MW is equivalent to 128MB of RAM.)

RESULTS AND DISCUSSION

Test Jobs. The job CPU times of the seven G94 test jobs (1, 28, 54, 94, 155, 194, 296, and 302) are reported in Table 2, for each of the various scenarios listed in Table 1. All times are given in seconds and are taken as reported by G94 itself at the end of the output file.

Cray Systems. Since on the Cray YMP, larger jobs have to be submitted, as a matter of policy, through a batch queue, the "User CPU time" as reported in the queuing system output file is also given for those two jobs for which direct submission was not possible. These times were on the order of 10% lower than the "Job cpu time" reported by G94 itself. It is difficult to assess how much of this reduction did actually benefit the user. The program output indicated that, during most of the computation, only one of the eight processors had been used in each of the test job runs, and the wall time elapsed was definitely on the order of the reported CPU times. We assume that these runs represent typical conditions a user will encounter on a shared supercomputer. Making this assumption, the results in Table 2 show that many of the commodity-type machines surpass the Cray YMP in speed, at least for the aggregate total time.

The Cray J90 runs shared many of the characteristics of Cray YMP calculations regarding submission mode, number of CPUs, various types of timings reported etc., some of which has already been mentioned in the Methods section. However, the CPU times were much worse on the Cray J90 than on the Cray YMP. In fact, it finished last of all machines benchmarked with the test jobs. This somewhat surprising result may reflect the overhead that is associated with job execution on a supercomputer such as the Cray J90, which makes timings for very short jobs unfavorable in comparison. However, even the longer ones among the test jobs ran more slowly than on most other machines, hence we must assume that these results represent the speed the user can assume for G94 jobs in this size range. (It must be noted that test155 and test296 are multistep jobs.)

Alpha/Linux Systems. The lowest total time (588 s) was scored by the Alpha500MHz scenario, followed very closely by Alpha533MHz (598 s). This apparently paradoxical result is most likely due to the fact that the Alpha533MHz machine was equipped with an IDE hard drive, for which the drivers in Linux/Alpha are known to be rather slow. The Alpha500MHz had a SCSI drive, for which the drivers are much faster. This speed differential, again, affects very short jobs to a relatively larger degree than longer jobs with long stretches of calculation without disk I/O. This assumption is supported by the results of the DFT runs.

To thoroughly validate the Alpha/Linux version of G94, the entire suite of test jobs was run. All 306 jobs, for which input and output files for runs on Gaussian, Inc.'s reference system (RS/6000-590) were included in the program distribution, completed successfully on the Alpha533MHz. It took this system 24.675 h of CPU time to do so vs 33.0 h for the RS/6000.

Other Scenarios. The two SGI machines delivered a strong performance for the test jobs, placing the Origin2000 and the Indigo2 in third (756 s) and fifth (923 s) position, respectively. The Pentium II 400 MHz system, however, was barely outclassed by the SGI/Cray Origin 2000, especially when optimized compilation was used (scenario PII/400/f2c/Opt4), with which it finished in fourth place (806 s). The other PII computers were slower than the 400 MHz system by somewhat more than their ratio in processor clock rates (speed ratios of 1.41:1.20:1.0 vs clock rate ratios of 1.33:1.125:1.0 for the 400, 337.5, and 300 MHz systems, respectively). The increase in bus speed from 66 to 75 to 100 MHz from the slowest to the fastest Pentium II tested may have contributed to these results. The various P6 scenarios ranged from 1324 s to over 2500 s, depending on the motherboard used, the on-chip cache size, and, not in the least, the compiler and compilation options used (see below). Figure 1 allows an easy overview over the perfor-

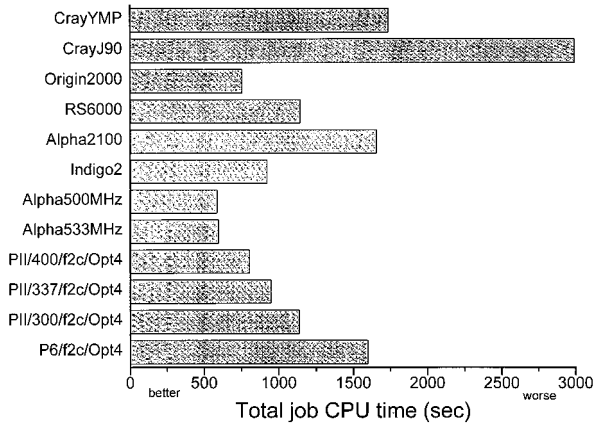

Figure 1. Sum of the job CPU times for all seven test jobs (no. 1, 28, 94, 155, 194, 296, 302).

mance for the test jobs by showing one scenario—generally the best one with confirmed energy results—for each hardware type.

test194. It is interesting to note that test job 194 seems to be an anomaly insofar as it completed much faster on the Cray YMP than on any other machine, and it was generally faster on the workstations/servers (IBM, SGI, DEC) than on the x86-based commodity-type computers. This test job uses the UQCISD(TQ) level of theory. This, along with other very high level, post-MP2, methods, differs somewhat from lower-level methods insofar as it will profit directly from large amounts of fast memory available. These calculations are dominated by very large matrix multiplies over the molecular orbitals, hence the more memory available the more of the matrix is held in RAM and the fewer passes to disk are required.[14] This may have been the reason that job 194 ran much faster on computers having very large amounts of RAM with very high access speed. Higher disk access speed may have additionally contributed to this result. The importance of main memory (RAM) speed as a potential main bottleneck will be further analyzed below.

Motherboards. In order analyze to what degree different motherboards influence the speed of an otherwise more or less identical P6 system, machines with motherboards from four different manufacturers (AMI, ASUS, Intel, SuperMicro) were tested. Inspection of the scenarios P6/AMI/f2c/256k, P6/ASUS/f2c/256k, P6/SuMi/f2c/256k, and P6/Intel/f2c/256k shows that, for the most part, the speeds are quite similar, being within approximately 7% of each other. The Intel motherboard based system was slowest, taking more than 10% longer to complete the jobs than the fastest of these four systems. It also was not able to function with the subsequently tested 512kB-cache variety of the P6 processor, as described below. This led us to speculate that the Intel board might have been flawed. All in all, these results suggested that for practical purposes, make and model of the motherboard (with identical chip sets) should not be of too much concern in most projects, as long as all components are of generally high quality.

Cache Size. The P6 processor comes with three L2 cache memory sizes, 256 kB, 512 kB, and 1 MB. The first two sizes were tested as to their influence on performance, by swapping processors among the four P6 motherboards. (The Intel motherboard could not complete the test with the 512 kB version of the CPU.) The influence of the L2 cache size on the timings for the G94 test jobs was surprisingly small. The speed increase ranged from 6% for the P6/ASUS/f2c/512k scenario to an insignificant 0.08% increase for P6/AMI/f2c/512k. Given the fact that, at the time of the systems' purchase, the 512kB cache P6 was about twice as expensive, the price/performance ratio would not have warranted the purchase of the much more expensive 512kB cache system. We believe that the near irrelevance of L2 cache size for the G94 test jobs, which was also found for the DFT job (see below), is largely due to the peculiarities of the G94 algorithms. This will be discussed in greater detail in the context of the DFT job.

Dual Runs. As mentioned previously, the G94 test jobs whose CPU times are reported in Table 2 were run on a single processor on the Dual P6 systems, with no other job running on the machine. Of course, one can run two Gaussian jobs simultaneously on these systems. We found these jobs to interfere only slightly, but still noticeably, with each other, since we measured a typical slowdown for each of two such jobs run in parallel on the order of 15−20% (data not reported). Still, this is a very viable way of maximally utilizing such a Dual processor system. One comparison performed with test job 302 on PII/337/f2c/Opt4 yielded a slowdown for two simultaneous runs of quite exactly 20% (43.0 s vs 51.6/51.7 s).

Compilers and Compilation Options. The standard compilation scripts (MAKEFILE) that are shipped by Gaussian, Inc. as part of the program distribution take a conservative approach at compile time program optimization. By slight modification of the MAKEFILE, the user can easily increase the optimization level. Doing so, by using the compiler options listed in Table 1, the speed of the program could be increased by 40% for the Pentium Pro (scenario P6/f2c/Opt3 vs P6/f2c). Further increase of the optimization level yields diminishing returns, since going from scenario P6/f2c/Opt3 to P6/f2c/Opt4 (mainly changing compiler optimization flag "-O2" to "-O3") yielded only a very slight additional speedup. Similar speed improvements, albeit somewhat different in size (12.7%−26.3%), were observed for the various Pentium II based systems. The single most important determinant for the speedup appears to be the optimization flag "-malign-double." The entire, important, aspect of program optimization is further discussed in the DFT section below.

A further speedup of G94 running on the P6 could be achieved by using a different compiler. The standard Gaussian installation scripts use the compiler combination f2c and gcc, two very widely used and freely available compilation tools in the Unix world. While this is a reliable method, several choices for compilers exist on this, as on most other, platforms. On Linux (and other) systems, another freely available Fortran 77 compiler is "g77".[15] Gaussian 94 could be compiled for Intel x86/Linux using this compiler. However, much more extensive alterations of Gaussian files were necessary, including source code modifications. This procedure, which is obviously not supported by Gaussian, Inc., yielded executable code that was even faster on the various P6 systems than the maximally optimized G94 version compiled with f2c/gcc.

This code appeared to be stable and was able to complete all test jobs. However, the energies for test job 194 differed, very consistently across different machines, by the same

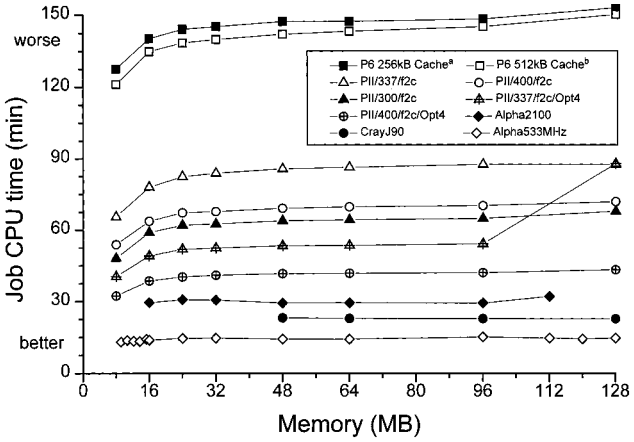

Figure 2. Job CPU time for DFT job as a function of memory amount specified in "%Mem" command. Superscript a: Scenario P6/f2c. Superscript b: Scenario P6/ASUS/f2c/512k.

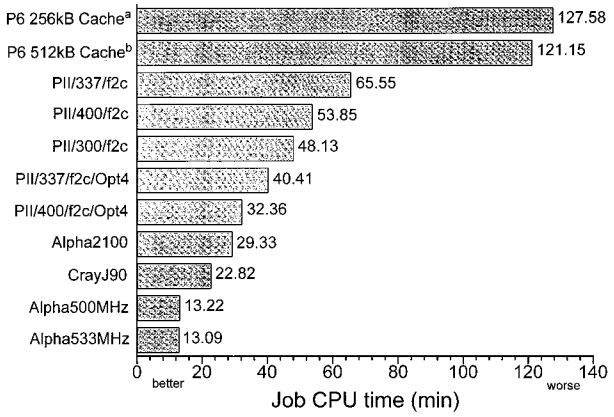

Figure 3. Best DFT job CPU time. Superscript a: Scenario P6/f2c. Superscript b: Scenario P6/ASUS/f2c/512k.

values of 4−21 kcal/mol (for the various stages of this multistep computation) from the reference values that are given by Gaussian, Inc. in the standard output files. These were otherwise reproduced exactly in all other scenarios. This raises the warning flag that speed optimization of the compiled code should not be pushed to the level where program reliability is compromised. Needless to say, this set of executables was not used for production runs despite its higher speed.

DFT Job. Memory Dependency. Gaussian 94 allows the user to specify, through the %Mem command, the maximum memory the program is allowed to use. Traditionally, i.e., on supercomputers such as the Cray YMP, common knowledge as well as Gaussian, Inc.'s recommendation (see, e.g., Gaussian 94 User's Reference,[16] p 161) was that the more memory was made available, the faster the job would run. In the case of memory shortage, G94 resorts to either discarding and then recomputing certain intermediate values (two-electron repulsion integrals), which it otherwise would have stored in RAM, or to swapping those values to disk. Storage on disk is by far the slowest method and is not used by G94 unless explicitly requested by the user. Recomputation, also called the "direct" method by Gaussian, used to be thought of as being slower than calculating the integrals once and storing them in main memory ("in-core" methods). However, because extremely large amounts of memory are needed for larger jobs with "in-core" methods, G94 defaults to "direct" unless otherwise instructed.

When this assumption was tested by varying the amount of memory given to the DFT job, unexpected results were obtained.[17] For all x86 systems, i.e., both for Pentium Pro and Pentium II machines, and irrespective of optimization used, the DFT job ran the faster the less memory it was given. The increase is steep in the beginning, i.e., for the smallest amounts of RAM. The curves start to level out at around 4MW (32MB), but they remain monotonically increasing in all cases for the Pentium class processors. Figure 2 shows this dependency of the DFT job CPU time on the amount of memory allotted for the P6, PII, and several other systems.

The increase in CPU time ranged from 20% for the P6 (with 256kB cache) to 41% for PII/300/f2c when going from the smallest amount of memory that allowed the job to run to completion, 1MW (8MB), to the largest amount tested, 16MW (128MB). (PII/337/f2c/Opt4 showed an anomalously large CPU time increase for 128MB and is therefore not taken into account here. This is discussed further below.)

The CrayJ90, in contrast, exhibited the "traditional" behavior: the DFT job ran faster the more memory it had available, albeit by very little. (The minimum amount of RAM the CrayJ90 needed to complete the job successfully was 6MW [48MB], and the timings were 23.16, 22.95, 22.84, and 22.82 min for 6, 8, 12, and 16 MW, respectively.)

Both the Digital Unix Alpha2100 and the Linux/Alpha 533 MHz machine showed an intermediate behavior, with some decrease of CPU time toward smaller amounts of memory but no strong trend otherwise. No Alpha based system could complete the DFT with 1MW (8MB) of RAM. For the Alpha533MHz, the range below 2MW (16MB) was explored in approximate 0.2 MW steps, and the smallest amount of RAM for which the job completed successfully (approximately 9.2MB) yielded the fastest run. The Alpha2100 could not complete the job with 16MW(128MB) although the machine itself had twice the amount of memory. It achieved the fastest run (29.33 min) at 12MW (96MB). To give an easy overview of the relative performance of the different platforms for the DFT job, the best timing for each scenario is shown in a bar-chart format in Figure 3.

Our tentative explanation for this seemingly counterintuitive result, particularly as seen on the Pentium based systems, is that these systems' processors are so fast in comparison to their memory access speed that recomputation of intermediate values becomes in fact more efficient than storing these values in, and reading them from, memory. This (relatively) slow memory access speed is a well-known bottleneck in today's commodity-type computers. Conversely, much of the high cost of true supercomputers is due to their very fast memory subsystems.

Manufacturers of commodity-type processors try to alleviate this problem through elaborate cache schemes. Programs that take good advantage of the cache memory can experience a significant speed increase. However, since G94 defaults to "direct" methods, and hence tries to avoid memory access to a large degree altogether, one would expect an only modest benefit of larger cache. This was indeed exactly what was found in the benchmarks. There was a clear, but quite small speedup of the DFT job for the 512kB cache variety of the P6 processor when compared with the 256kB cache P6. Both in absolute and relative terms, the

speed difference was larger the smaller the (main) memory specified through the %Mem command was (see Figure 2), clearly showing the cache's increasing relative proportion of the total memory available to the program.

This behavior does not seem to a particular feature of the DFT method. One run of the test job set conducted with 8MW (64MB) of memory (P6/f2c/Opt4@8MW, Table 2) yielded longer execution times for each of the seven jobs (which all use different ab initio methods) when compared to the 4MW (32MB) default runs (P6/f2c/Opt4). The least affected by the memory size change was test job 194, which again points to the special nature of this G94 method as far as resource usage is concerned. A similar observation was made on the two Alpha/Linux systems, where test job runs conducted at 8 million words of memory took about 15% longer on either machine (Table 2), with test job 194 again showing virtually no, or even inverse, effect of the memory size increase.

Compiler and Compilation Options. Similarly to the test jobs, a significant speed difference was found for the DFT job between versions of Gaussian 94 compiled with different compiler options on the Pentium based systems. While different explicit optimization levels (such as "-O2" vs "-O3") were seen to have some influence, the major speedup was brought about by the introduction of alignment flags (see Table 1, column 6). These rather technical compilation options help ensure well-defined alignment of certain program elements along multi-byte boundaries in memory during run time. These problems are specific to x86 processors.

Through experiments not shown here we found that the "-malign-double" flag, which aligns memory addresses on 64-bit boundaries (double-word limits), was the most important one to ensure fastest possible program runs. Lack of such alignment was found to sometimes, under unknown circumstances, cause differences in program speeds of the same code on very similar (or even the same) hardware of up to 100% (data not shown)—or virtually no difference at all. In our study, one of the Pentium II systems was clearly an outlier in this sense. The 300 MHz PII showed a much smaller difference in timings between the standard compilation and the one using "-malign-double" and the other optimization flags than the 337.5 MHz and the 400 MHz machines. For 1MW (8MB), the DFT job took 48.13 min with the standard compilation vs 45.70 min with the optimizing one. For this reason, no benchmark results are reported for a "PII/300/f2c/Opt4" scenario.

The sudden increase in CPU time for PII/337/f2c/Opt4 for 16MW (128MB), to a value even slightly above the one for PII/337/f2c (87.96 vs 87.65 min), is probably also an effect in this category, i.e., a subtle interplay of placement of the binaries in memory coupled with details of the memory hardware architecture of the system, which appears to have destroyed memory alignment.

All in all, it appears that at least the "-malign-double" flag should be present during compilation to ensure consistently fast program execution. This was found to hold for programs in general (results not shown) and is not a peculiarity of Gaussian 94. It is worthwhile to note, in this context, that poor choice of compiler options combined with the wrong strategy for memory allotment can cause more than doubling of the CPU time for the same job on the same hardware: the 1MW (8MB) timing for PII/400/f2c/Opt4 was 32.36 min compared to 71.96 min for the 16MW (128MB) run with PII/400/f2c.

Table 3. CrayJ90 Timings for the DFT Job[a]

timings	memory 6MW (48MW)	8MW (64MB)	12MW (96MB)	16MW (128MB)
	Default			
job CPU time	24.48	24.90	23.08	22.44
user CPU time	24.32	24.77	22.96	22.28
elapsed time	14.02	8.28	7.82	7.78
av concurrent CPUs	1.90	3.42	3.39	3.35
	%Nproc=1			
job CPU time	23.16	22.96	22.84	22.82
user CPU time	22.91	22.69	22.54	22.55
elapsed time	47.25	42.07	40.60	40.75
av concurrent CPUs	1	1	1	1

[a] All times in minutes.

Changing the version of the compilers used for the x86 compilation of G94 (f2c v. 19960717, gcc v. 2.7.2.2.f.2; C library v. libc.so.5) did not affect the speed of the program. Gaussian Revision E.1, compiled with more recent versions of the compilers (f2c v. 19971204, gcc 2.8.1, and libc.so.6, compilation flag -mi486 replaced by -mcpu=pentiumpro -march=pentiumpro), produced, if anything, a minimal slowdown of the code in a number of DFT trial runs, however well within the error margins. This avenue was therefore not further pursued.

CrayJ90. As expected, the CrayJ90 delivered a very strong performance for the DFT job. It was, however, not the fastest platform when measured on a per-processor basis (see below). As mentioned, both a series of runs with the default setting was performed, which used up to four processors in parallel, and a series in which a one-CPU limit was enforced. Table 3 lists the various times that were reported by these runs. As expected, the elapsed time was much reduced when concurrent CPU usage was enabled. However, both the job CPU time and the user CPU time increased as compared to the one-processor runs, presumably because of the overhead of parallel execution.

The elapsed time in the forced single-CPU runs was surprisingly large, on the order of twice the job CPU time. Because the CrayJ90 was not devoid of other runs at the time of the benchmarkings, it is not clear whether this slowdown was due to the load level present at the time of testing or reflects a suboptimal regime for running G94 jobs on this system when enforcing single-processor mode.

Alpha/Linux. While the fastest Pentium II based systems scored impressively for the DFT job, the 500 MHz and the 533 MHz Alpha/Linux systems pulled far ahead of them and, in fact, beat any other platform including the CrayJ90 by a wide margin when compared on a per-processor basis. In contrast to the short test jobs, which may have favored the SCSI hard drive-equipped Alpha500MHz to a somewhat larger degree, the longer DFT job showed a very slight speed advantage of the Alpha533MHz compared to the Alpha500MHz. Because the numbers are so close, no separate curve is shown for the Alpha500MHz in Figure 2.

An additional test helped to confirm these results and to place them in the context of previously published benchmarks. The longest one of the geometry optimizations used

in the benchmark study recently reported by Yu and Yu[4] was repeated on the Alpha533MHz with the exact same input files as used by these authors. This optimization consists of three stages, using, in sequence, the BP86, BLYP, and B3LYP DFT methods with the aug-cc-pVTZ basis set. The Alpha533MHz performed the first two of these steps faster than any other machine tested by Yu and Yu with the exception of a Fujitsu VPP300/2 supercomputer; for the B3LYP optimization, it would rank fourth among 13 machines. (However, some uncertainty remains as to the number of optimization steps performed and the number of cycles needed to achieve wave function convergence [as has been discussed above], since these values are not reported by the authors. Certain trends in the timings indicate that two out of the three machines found to be very fast for the B3LYP optimizations may actually have performed fewer steps before achieving convergence, when compared to all other machines, including the Alpha533MHz. If these two potential outliers are excluded, the Alpha533MHz would again end up in second place after the Fujitsu supercomputer.) The values for the Alpha533MHz, reported in the units hh:mm:ss used by the above authors, are 13:44:35.9, 10:00:12.4, and 11:47:03.2 for the BP86, BLYP, and B3LYP steps, respectively.

Many of the matrix operations of G94, which often constitute a large part of the computation for Gaussian jobs, are handled by a library function of the BLAS3 (Basic Linear Algebra Subprograms, level 3) type. On Digital Unix, this BLAS type library is part of DEC's proprietary Digital Extended Math Library (DXML). The routine that is centrally used by Gaussian is DGEMM (Double-precision GEneral Matrix-Matrix), a BLAS3 routine for matrix multiplication. This had to be replaced by a nonproprietary library for Linux/Alpha.

This nonproprietary version of DGEMM, which was hand-coded in assembler by K. Goto in cooperation with one of us (R.W.),[18] plays an important role in the high speed of this platform. Still, it is not an all-overriding factor, as the following tests showed. Our DGEMM subroutine was optimized for the 21 164 generation of the Alpha chip and its specific memory and cache architecture. It cannot be compared on Alpha/Linux with DXML, because the proprietary nature of DXML does not allow execution on Alpha/Linux. These two libraries can, however, be compared on a Digital Unix system, since K. Goto's GPL-governed[19] DGEMM has no platform restrictions. On the Alpha2100, which has the "wrong" Alpha processor, a previous-generation 21064 chip, G94/DXML performed about 20% faster than G94/DGEMM. On a different Digital Unix system (not listed in Table 1), equipped with a 433 MHz 21164 Alpha CPU, G94/DGEMM was 20% faster than G94/DXML.

It was found that, for optimum performance, DGEMM had to be compiled using the GNU and not the Digital Unix compiler, having been originally developed and optimized with the former compiler. As mentioned before, the actual compilation of Gaussian for Linux/Alpha, however, had then to be performed using the Digital Unix compilers, before the resulting executables had to be transferred back to the Alpha/Linux system.

The somewhat experimental nature of the compilation of G94 for Alpha/Linux, coupled with the necessity to have a Digital Unix system and license available, makes this approach of less than general applicability. However, it shows the potential of the fastest of today's commodity-type processors for applications such as Gaussian 94. Our efforts continue to achieve native compilation of G94 on this platform.

Disk Drives. The types and sizes of the hard disk drives in the individual machines are not reported in Table 1 because we found very little influence of this parameter on the speed of the G94 jobs run in this study. This was especially true for the—more real-life like—DFT job which had longer stretches of pure computation without I/O as compared to the very short test jobs. In fact, a test on PII/300/f2c, with the G94 temporary files residing on the *slowest* storage medium available with sufficient capacity, a removable 1GB Iomega Jaz cartridge, showed a negligible slowdown of the DFT job when compared with the run using the Ultra-DMA IDE fixed hard drive (48.39 min vs 48.13 min, for 1MW [8MB] of RAM). All P6 systems, the PII 300 MHz machine, and the Alpha533MHz computer were equipped with IDE drives; all other systems, as far as it is known to us, had SCSI hard drives.

This near independence of the G94 speed of the disk type used would seem to us to find its natural explanation in the characteristic of the "direct" method of Gaussian. Since this type of algorithms tries to avoid memory access, and even more so disk usage, any speed difference of the magnetic media will simply not affect those parts of the computation that do not use these media at all.

Prices. We list no prices for any of the systems described in this paper, because the very rapid developments in the mass-market computer area would immediately render detailed price information obsolete. However, it can be reported that all of the commodity-type systems cost about or less than U.S. $5000 at the time of their purchase, and most of them were less than U.S. $3000. At the time of this writing, systems with the capabilities (for Gaussian calculations) of any of the computers at or below the Alpha500MHz scenario in Table 1 (without monitor and with an inexpensive hard drive) can be purchased for less than U.S. $3000.

CONCLUSIONS

The analysis we have presented shows that commodity-type computers have not just become viable alternatives for medium-size computational chemistry calculations but have surpassed in power the most powerful workstations and even supercomputers.

If utilizing the most recent and fastest mass-market CPUs, and owing to the unprecedented lows in prices of hardware, the user is now able to purchase for himself or herself a machine costing only few thousand dollars that rivals in performance the last-generation supercomputers, whose cost were maybe 3 orders of magnitude higher.

Free operating systems such as Linux make these platforms even more attractive. The open and somewhat fluid nature of these free operating systems may pose challenges in some instances but generally offers advantages to most users. Commercial repackagers, charging very modest prices, greatly alleviate installation and maintenance difficulties.

Our analyses showed that, in many instances, slight variations in hardware, such as different motherboard

Chart 1. DFT Input File

```
%Chk=DFTbench.chk
# B3LYP/6-31G(d) Guess=Read    test

   From Marquez, et al., Nucleos. Nucleot. 16(7-9),  1431-1434 (1997).
   Instructions for benchmarking: Run initial job without "Guess=Read";
   save .chk file; re-run job with "Guess=Read", using saved .chk file.
   Wave function should converge in 2 cycles.
   Energy should be -890.4784 hartrees.

   0   1
C          0.000000000000      0.000000000000      0.000000000000
C          0.000000000000      0.000000000000      1.527217962002
C          1.486190619759      0.000000000000      1.941430191892
C          2.256865299558     -0.636402012517      0.746986584060
C          1.394762618406     -0.357639581514     -0.478852501473
C          0.850929319266      1.037769892272     -0.683543494421
C          2.456222107196     -2.139330704527      0.955640899555
O          1.867932602009      1.356027421793      2.178789369127
O          3.098195132181     -2.666765686916     -0.199442516582
N         -1.170418733105     -0.498816670300     -0.669605836405
C         -1.368454614806     -1.767571784543     -1.186801501454
N         -2.562444786776     -1.964465738489     -1.691616984415
C         -3.206083144955     -0.755389534735     -1.488192502927
C         -2.367589979053      0.170064977105     -0.856181659692
C         -4.504691363168     -0.301390274381     -1.797424414622
N         -4.846804552127      0.958943546223     -1.479338713580
C         -3.931986285953      1.729368447894     -0.865371798530
N         -2.679034093253      1.427399131714     -0.513799649730
N         -5.434689888631     -1.106311264261     -2.379250215475
H         -0.493457911185     -0.910623245052      1.887545901971
H         -0.520522978384      0.860466235939      1.958076821600
H          1.636282526113     -0.586290001930      2.861626155309
H          3.251989402157     -0.179285145833      0.652138004950
H          1.548652011501     -0.981490284312     -1.353371406392
H          1.229255681520      1.855597018109     -0.079562401900
H          0.566948249648      1.299582660894     -1.698397506572
H          1.480292405953     -2.625618011967      1.123418040969
H          3.065140009087     -2.305154524998      1.859523922704
H          2.822885311553      1.369636976660      2.350040750088
H          3.203366295989     -3.622181527121     -0.076098808709
H         -0.570058121343     -2.498131651997     -1.169001924954
H         -4.262445542218      2.738300873596     -0.626087699978
H         -6.263869654783     -0.661839268197     -2.746167134765
H         -5.123336658946     -1.965369447828     -2.808241185857
```

brands, or different cache sizes, make much less of a difference for the program speed than the choice of the compiler and compiler options, in particular optimization levels and flags. However, it was also seen that aggressive optimization, leaving the path of the distributed and supported installation procedures, has to be carefully validated before any scientific project is based on such a "tuned" program.

The somewhat peculiar nature of the Gaussian algorithms, which can be instructed to use a "direct" method of recalculating intermediate values instead of storing them, coupled with the possibility to change the program's usage of memory, appears to convey an unexpected opportunity to circumvent a shortcoming of many of the commodity-type systems, the much smaller memory access speed vs the ever-increasing processor speeds. This kind of algorithm may be a model for programming, or re-programming, other computational chemistry applications for optimal speed on today's commodity-type processors.

The commodity-type machines are quite likely to exert a growing force on the market for molecular modeling and computational chemistry. Their price/performance ratios will make them extremely attractive for many chemists who do not have an unlimited budget, even if availability of commercial applications, such as modeling programs with graphical user interfaces, is still limited for these platforms as of now. This situation will improve as alert vendors will follow this shift dictated by the astounding developments in the hardware area.

ACKNOWLEDGMENT

We gratefully acknowledge the contribution of Kazushige Goto, whose DGEMM subroutine was instrumental in making Gaussian 94 on Alpha/Linux possible. We thank Michael D. Bartberger for valuable input. We thank J.-S. K. Yu and C.-H. Yu for providing us with their input files. We thank Bernard R. Brooks and Robert Pearlstein for support of this study. Milan Hodošček thanks for support in part by an appointment to the Research Participation Program at the CBER administered by the Oak Ridge Institute for Science and Education (shortened: ORISE) through an interagency agreement between the U.S. Department of Energy and the U.S. Food and Drug Administration. We acknowledge the National Cancer Institute for allocation of computing time

and staff support at the Frederick Biomedical Supercomputing Center of the Frederick Cancer Research and Development Center. All trademarks mentioned in this paper are the properties of their respective owners.

REFERENCES AND NOTES

(1) http://www.linux.org.
(2) Becker, D. J.; Sterling, T.; Savarese, D.; Fryxell, B.; Olson, K. Fourth IEEE International Symposium on High Performance Distributed Computing, August 1−4, 1995, Pentagon City, VA. See, also: http://cesdis.gsfc.nasa.gov/beowulf/.
(3) Tirado-Rives, J.; Jorgensen, W. L. Viability of Molecular Modeling with Pentium-Based PCs. *J. Comput. Chem*. **1996**, *17*, 1385−1386.
(4) Yu, J. S. K.; Yu, C. Benchmarks of the PC−Unix computer with electronic structure calculation. *J. Chem. Inf. Comput. Sci*. **1997**, *37*, 1111−1114.
(5) http://mephisto.ca.sandia.gov/benchmarks.html; http://www.emsl.pnl.gov:2080/docs/tms/abinitio/cover.html; http://www.cc.mie-u.ac.jp/~ne70101/gms-bench.html; http://www.chem.joensuu.fi/people/juha_muilu/Misc/benchmarks.html.
(6) http://SAL.kachinatech.com/Z/2/; http://lem.ch.unito.it/cl/; http://www.lmcp.jussieu.fr/sincris-top/logiciel/result.html.
(7) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Gill, P. M. W.; Johnson, B. G.; Robb, M. A.; Cheeseman, J. R.; Keith, T.; Petersson, G. A.; Montgomery, J. A.; Raghavachari, K.; Al-Laham, M. A.; Zakrzewski, V. G.; Ortiz, J. V.; Foresman, J. B.; Cioslowski, J.; Stefanov, B. B.; Nanayakkara, A.; Challacombe, M.; Peng, C. Y.; Ayala, P. Y.; Chen, W.; Wong, M. W.; Andres, J. L.; Replogle, E. S.; Gomperts, R.; Martin, R. L.; Fox, D. J.; Binkley, J. S.; Defrees, D. J.; Baker, J.; Stewart, J. P.; Head-Gordon, M.; Gonzalez, C.; Pople, J. A. Gaussian 94 (various revisions) is provided by Gaussian, Inc., Carnegie Office Park, Bldg. 6, Pittsburgh, PA 15106.
(8) Kazushige Goto, 1-7-6, Kamimizo, Sagamihara City, Kanagawa, Japan, ZipCode 229-1123. The DGEMM library can be downloaded from ftp://ftp.eni.co.jp/.2/LinuxAlphaJP/ftp.statabo.rim.or.jp/BLAS/.
(9) Durant, J. http://mephisto.ca.sandia.gov/benchmarks.html.
(10) Marquez, V. E.; Ezzitouni, A.; Siddiqui, M. A.; Russ, P.; Ikeda, H.; George, C. Conformational analysis of nucleosides constructed on a bicyclo[3.1.0]hexane template. Structure-antiviral activity analysis for the northern and southern hemispheres of the pseudorotational cycle. *Nucleosides Nucleotides*. **1997**, *16*, 1431−1434.
(11) Lee, C.; Yang, W.; Parr, R. G. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B* **1988**, *37*, 3098−3100.
(12) Becke, A. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A* **1988**, *38*, 3098−3100.
(13) Becke, A. Density-functional thermochemistry. III. The role of exact exchange. *J. Chem. Phys*. **1993**, *98*, 5648−5652.
(14) Gaussian 92 Workshop Notes; Gaussian, Inc.: Pittsburgh, PA, 1994.
(15) http://www.gnu.org/.
(16) Frisch, M. J.; Frisch, Æ.; Foresman, J. B. *Gaussian 94 User's Reference*, Version 5.0; Gaussian, Inc.: Pittsburgh, PA, 1995;.
(17) It was ascertained that none of the jobs had run "in-core" by scanning the output files for (the nonoccurrence of) the line "Two-electron integrals will be kept in memory." See Foresman, J. B.; Frisch, Æ. *Exploring Chemistry with Electronic Structure Methods*, 2nd ed.; Gaussian, Inc.: Pittsburgh, PA, 1996; p 31.
(18) A library containing the DGEMM subroutine plus a few other functions can be downloaded from ftp://rw.usuf2.usuhs.mil/pub/libdgemm/. The file libdgemm-980415-2-coff-gnu.a can be used as a direct replacement for the DXML library in the make file command "ESSLIB = −ldxml".
(19) This subroutine has been placed under the GNU Library Public License. For its text, see, e.g.: http://www.netppl.fi/~pp/glibc/libc_32.html.

CI9800920