

Ab Initio Quantum Chemistry on the IBM® pSeries™ 690

A Comparison Between Turbo 1.3 GHz and Turbo 1.1 GHz

Carlos P. Sosa, IBM, pSeries High-Performance Computing, Chemistry and Life Sciences
Development Solutions

Patton Fast, University of Minnesota Supercomputing Institute

February 14, 2002

©2002 International Business Machines Corporation, all rights reserved

Abstract

In this study, we compare the performance of the POWER3 processor and the new IBM® eServer pSeries™ 690. The pSeries 690 can scale up to 32-way POWER4 processor at 1.3 GHz and 1.1 GHz. To perform this comparison we used the *Gaussian98* Revision A.11 series of electronic structure programs. It is an integrated system to model molecular systems under a variety of conditions, carrying out its calculations from basic laws of quantum mechanics. We analyze and compare the performance of methods such as Hartree-Fock and density functional theory (DFT), including first and second derivatives. In addition, we also look at excitation calculations via CIS. This set of benchmarks indicate that the Turbo 1.3 GHz is approximately 15% faster than the Turbo 1.1 GHz. Scalability between these two machines is similar.

Introduction

The combined effort between software and hardware is what makes it possible to carry out calculations that a few years ago were unthinkable. An RHF/6-31G** single point energy calculation on triamino-trinitro-benzene (TATB) using *Gaussian88* on an IBM RS/6000® model 550 used to take 4.5 hours [1]. The same calculation using *Gaussian92* took about 1 hour [1]. Today this calculation takes less than 2 minutes using *Gaussian98* on the latest IBM POWER3 systems.

In addition to faster processors, chemists have long realized that parallel computing requires scalable software [2] and hardware [3]. Scalable hardware can be classified in terms of how memory is organized [4,5]: *distributed*-memory, in which each node has its own local memory (e.g., IBM SP nodes); and *shared*-addressable memory among all the processors (shared memory). The IBM pSeries 690 is a major step that IBM has taken toward providing customers with shared memory machines.

The next generation of the IBM POWER series (pSeries 690) was introduced in October 2001. This new machine scales up to 32-way POWER4 1.3 GHz or 1.1 GHz SMP and up to 256 GB of memory. The IBM pSeries 690 uses multi-chip modules (MCM). The MCM is equivalent to a mainframe's central processing module. This approach optimizes chip-to-chip communications, boosting performance of the overall system. The module-to-module interconnect runs at half of the processor's frequency [6].

Gaussian98 can take advantage of both memory approaches. Previous studies have reported the performance of *Gaussian*® running on different architectures [3,7,8,9]; however, it is beyond the scope of this work to review *Gaussian* implementation on distributed architectures. In the next section we present the design features of the systems tested on this study. We also provide a section that describes how *Gaussian* has been parallelized on a shared memory architecture. In the last four sections we describe the benchmarks and their performance. We conclude this work with a section on resource management and a summary.

Design Features

Three different types of IBM pSeries servers were used in this study: 375 MHz POWER3-II multiprocessor (SMP) and two pSeries 690 servers, one corresponding to the 1.3 GHz POWER4 pSeries 690 Turbo and the other to the 1.1 GHz POWER4 pSeries 690 Turbo. For simplicity we will refer to them as the Turbo 1.3 system and Turbo 1.1 system.

The pSeries 690 server is the latest UNIX[®] server from IBM and provides a new architecture [6,10]. At the core of its architecture is the POWER4 Multi-chip Module (MCM). The building blocks for the systems utilized here are an 8-way MCM Turbo running at 1.3 GHz and 1.1 GHz. The Turbo system (8-way MCM) has two cores per L2 cache, hence, one MCM is 8-way. A full description of the POWER4 architecture is beyond the scope of this work; however, in this section we will provide an overview of the most important features of this architecture. Further details are given in refs. [6] and [10].

Each processor chip on the pSeries 690 consists of two processors, an L2 cache that runs at the same speed of the microprocessor, the microprocessor interface unit, which is the interface for each microprocessor to the rest of the system; the directory and cache controller for the L3 cache; the fabric bus controller and a GX bus controller that enables I/O devices to connect to the central electronic complex (CEC). The L3 cache is a new component not available on the POWER3 architecture. The L3 caches are mounted on a separate module.

The 375 MHz POWER3-II system consisted of 16 processors, 24 GB of memory, and 8 MB L2 cache. The Turbo 1.3 consisted of 32 processors, 8X32 GB memory cards for a total of 256 GB of memory. The Turbo 1.1 had a total of 8X32 GB memory cards for a total of 256 GB of memory, as well.

All the timings in this work correspond to elapsed time. Timings were measured by using the clock and CPU times printed by each link using the *Gaussian #p* option. This information was used as input for a utility program that tabulates timings and speedups. The *Gaussian98* binaries utilized in this study were compiled on a POWER3 system with the xlf 7.1.0.2 FORTRAN compiler [11].

Parallel Gaussian

Gaussian [12] is a connected series of programs and can be used for performing a variety of electronic structure calculations, molecular mechanics, semi-empirical, *ab initio*, and density functional theory. *Gaussian* consists of a collection of programs commonly known as links; each link communicates through disk files and are grouped into overlays [13]. Links are independent executables located in the g98 directory and labeled as *lxxx.exe*, where *xxx* is the unique number of each link. In general overlay zero is responsible for starting the program, this includes reading the input file. After the input file is read, the route card (keywords and options that specify all the *Gaussian* parameters) is translated into a sequence of links. Overlay 99 (*l9999.exe*) terminates the run, in most cases *l9999.exe* finishes with an archive entry (brief summary of the calculation).

As previously pointed out, the *Gaussian* architecture on shared-addressable or distributed memory systems is basically the same [3]. Each link is responsible for continuing the sequence of links by invoking an `exec ()` system call to run the next link. The links that run sequentially are mainly links that are responsible for setting up the calculation and assigning symmetry. Although in

previous publications we have summarized all the links that run in parallel [3], it is important to note that, the use of multiple processors benefit from calculations that make use of the PRISM routines [14].

In a self-consistent field (SCF) scheme, the two-electron integrals are part of the Fock matrix[15]:

$$F_{\mu\nu} = H_{\mu\nu} + \sum_{\lambda} \sum_{\sigma} P_{\lambda\sigma} [(\mu\nu|\lambda\sigma) - 1/2(\mu\lambda|\nu\sigma)] \quad (1)$$

where H represents the core Hamiltonian. m, n, l, s are atomic orbital indices. The quantities $(mn|ls)$ are two-electron repulsion integrals. In *Gaussian*, these quantities are computed once and stored or recomputed as many times as needed, depending on the disk space and memory available and the algorithm chosen.

In density functional theory (DFT), eq. (1) can be rewritten by replacing the last term by the well-known exchange-correlation term F^{XC} [16].

$$F_{\mu\nu} = H_{\mu\nu} + \sum_{\lambda} \sum_{\sigma} P_{\lambda\sigma} (\mu\nu|\lambda\sigma) + F_{\mu\nu}^{XC} \quad (2)$$

In *Gaussian*, the two-electron integrals are parallelized by distributing batches of integrals among all the available processors (normally selected using the `%nproc` keyword in the input file). This procedure is illustrated in Figure 1. The main loop over N_{PROCS} (number of available processors) determines the number of integrals that will be distributed among the processors. Each task looks at $1/N_{\text{PROCS}}$ of the shell quartets, discarding those that do not need to be done due to symmetry, cutoffs, or the fact that they have already been done. All the integrations are carried out in these loops, within these series of loops and still in the main N_{PROCS} loop, the last loop sums up the contributions to the Fock matrix, derivative matrices, or density matrices, depending on the types of integrals computed. The next loop outside the first N_{PROCS} loop, is another N_{PROCS} loop that adds all the contributions to the Fock matrix together in a serial block of code. This scheme can be exploited to compute first and second two-electron integral derivatives, first and second one-electron integral derivatives and electrostatic potential integrals.

```

loop over NPROCS
  loop over total angular momentum
    loop over degrees of bra and ket contraction
      do integrals for 1/NPROCS of shell quartets
    endloop
  endloop
  add integral contributions to partial Fock matrix
endloop
loop over NPROCS (sequential code)
  add 1/NPROCS Fock matrix contributions
endloop

```

Figure 1. Parallel loops to compute two-electron integrals in *Gaussian*.

The parallelization of the Fock matrix in *Gaussian98* on IBM shared-memory architectures has been accomplished by using standard UNIX fork and wait commands [9]. `fork()` creates a new process. Upon successful completion, the fork subroutine returns a value of 0 to the child process and returns the process ID of the child process to the parent process. Otherwise, a value of -1 is returned to the parent process, no child process is created. In *Gaussian98* `fork()` is called within

the N_{PROCS} loop and each new child executes a batch of integrals. All the children that have completed their task wait prior to entering the sequential do loop to add their contributions.

Selected Benchmarks

Similarly as in previous studies [3,9,17], we consider four major characteristics when studying parallel performance on the IBM SP or selecting benchmarks: job type; theoretical method; basis set size; and molecular size. The job type corresponds to a single point energy, a gradient calculation or calculation of second derivatives. Normally single point calculations is used to compute accurate energies at a level of theory that is too expensive to carry out a full geometry optimization for any medium to large size systems. Due to the importance of molecular structure in chemistry, a large majority of calculations are geometry optimizations using Hartree-Fock or Density Functional Theory. This type of calculation is normally followed by a frequency calculation. To a lesser extent, geometries are also optimized at the MP2 level of theory.

There are many options for carrying out calculations using *Gaussian*. In this study we tried to select job types that reflect how users are currently running *Gaussian*. This by no means represents the only options used in the program, but they do represent a large percentage of calculations carried out at computer centers. They also correspond to many of the benchmarks carried out for hardware procurements.

In this study we have chosen the following types of calculations: single-point energy (SP) calculations at different levels of theory, FORCE, this type of calculation corresponds to an SP calculation followed by the calculation of the first derivatives of the energy with respect to the position of the atoms in the molecule. The time required for a geometry optimization is a multiple of the time needed for a FORCE calculation. We also carried out frequency calculations as the third type of calculations. *Rather than doing a full geometry optimization, a FORCE calculation is recommended for benchmarks that involve hardware performance.* It is equivalent to doing one cycle of the optimization and should provide a good approximation for the performance of an optimization calculation.

A large number of approximate theoretical methods have been reported in the literature [15]. These methods range in accuracy and computational cost. Since *Gaussian* provides most of these methods it is important to understand how they perform as a function of system resources. In this study we refer to system resources as the number of processors, memory and disk space needed for optimal performance. The theoretical methods chosen in this study have been extensively discussed in the literature [15] and it is beyond the scope of this work to describe these methods. The approximation used in this work corresponds to Hartree-Fock [15], the three parameter hybrid density functional theory of Becke (B3-LYP) [18], and configuration interaction singles (CIS) energy and gradients [19].

The cases used in this study correspond to the same cases from a previous study [17] augmented with a FORCE calculation using Valinomycin. The system used for Cases I and III is a-pinene. Case I is an SP calculation at the HF level of theory using 6-311G(df,p) basis sets. Case II corresponds to an SP FORCE calculation on a fairly large system (Valinomycin). Case III is an a-pinene frequency calculation at the B3-LYP/6-31G(d) level of theory. Cases II and III exercise several of the links that run in parallel. Case IV is a FORCE calculation of acetyl phenol excited states using CI-single excitations with the 6-31++G basis sets [19]. This set of benchmarks represent small-large systems to test speedup and efficiency for most parallel links.

All the geometries are available from Ref. [20].

Results and Discussion

In this study as in our previous reports we look at performance in terms of speedup [17]. Speedup (S) is defined as the ratio of the serial run time (elapsed time, t_s) over the time that it takes to do the same problem in parallel (elapsed time, t_p).

$$S = \frac{t_s}{t_p} \quad (3)$$

Efficiency (e) is the fraction of time that a processor is doing useful work. This measurement also indicates or provides an indirect indicator for the percentage of parallel code needed for linear or nearly linear scalability.

$$e = \frac{S}{N_{PROCS}} \quad (4)$$

To compare scalability we look at the measured speedup against ideal speedup, rather than using an extrapolated speedup as we have done in the past. We take this approach because we want to compare how POWER4 machines perform, rather than analyzing *Gaussian's* scalability. It is important to note that clock speed of the POWER4 is significantly faster than the POWER3 and this may affect scalability.

Table 1 illustrates the performance between the POWER3 and the POWER4 (Turbo 1.3 and Turbo 1.1) for a Hartree-Fock single-point energy calculation using medium size basis sets (Case I). Parallelization is achieved mainly in *l502.exe* (almost all of the CPU is spent in *l502.exe*). Table 1 looks at the scalability of *l502.exe* (or SCF procedure) and of the entire calculation (Total time or time to solution). The column under the label "Total" includes the overhead introduced by the links that run sequentially. In the case of single processor the improvement in going from the POWER3, 375 MHz to the POWER4, 1.3 GHz (Turbo) is a factor of approximately 3.34 for *l502.exe* and the total time. In this particular case and for single processor performance the improvement in going from the Turbo 1.1 to the Turbo 1.3 is 13% for *l502.exe* and the total time. The speedup for the entire calculation is a factor of 1.15.

Table 1. Hartree-Fock Single-Point Energy Calculations on a-pinene(C₁₀H₁₆)^a

Number of Processors	<i>l502.exe</i> ^b	S ^c	e ^c	Total ^{b,d}	S ^c	e ^c
1						
POWER3	2819	1.00	1.00	2827	1.00	1.00
Turbo 1.3	843	1.00	1.00	847	1.00	1.00
Turbo 1.1	973	1.00	1.00	977	1.00	1.00
2						
POWER3	1385	2.04	1.00	1394	2.03	1.00
Turbo 1.3	439	1.92	0.96	444	1.91	0.96
Turbo 1.1	504	1.93	0.97	509	1.92	0.96
4						
POWER3	702	4.02	1.00	716	3.95	0.99
Turbo 1.3	282	2.99	0.75	288	2.94	0.74

Turbo 1.1	325	2.99	0.75	331	2.95	0.74
8						
POWER3	364	7.74	0.97	385	7.34	0.92
Turbo 1.3	164	5.14	0.64	173	4.90	0.61
Turbo 1.1	145	5.04	0.73	152	5.56	0.70
16						
POWER3	196	14.38	0.90	232	12.19	0.76
Turbo 1.3	97	8.69	0.54	111	7.63	0.48
Turbo 1.1	114	8.54	0.53	129	7.57	0.47
32						
Turbo 1.3	63	13.38	0.42	85	9.96	0.31
Turbo 1.1	73	13.33	0.42	99	9.87	0.31

^a Total of 346 basis functions; basis sets are 6-311G(df,p); C1 symmetry.

^b All timings are in seconds and correspond to elapsed time.

^c Speedup and efficiency.

^d Total time to complete the run.

The scalability observed in this case is what one might expect (see Figures 2 and 3 for scalability of *l502.exe* and the entire run). With 16 processors the POWER3 shows an efficiency of 0.90 (or 90%) for *l502.exe*. Part of the 10% overhead may be attributed to the diagonalization that is done serially.

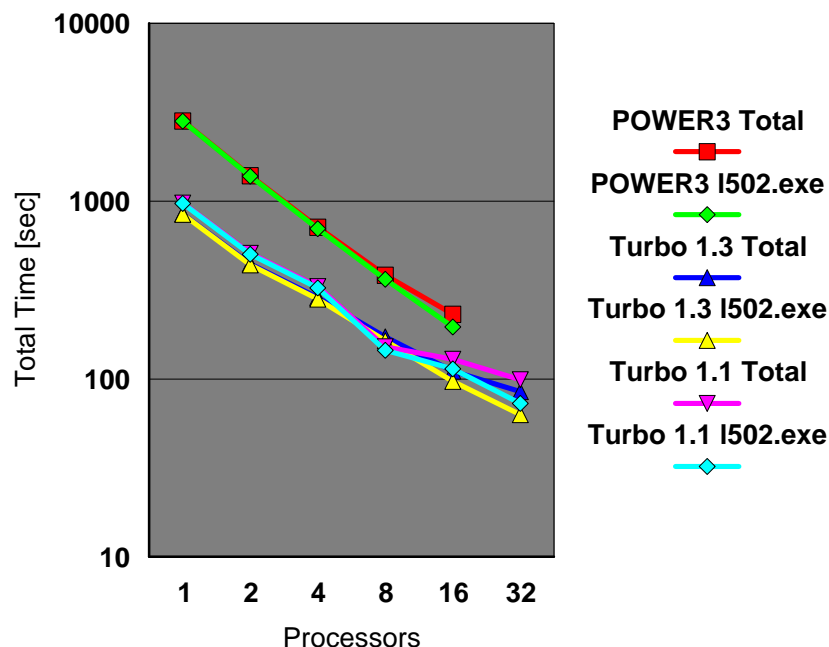


Figure 2. Scalability for *l502.exe* and entire run.

On the other hand, the efficiencies for the Turbo 1.3 and Turbo 1.1 are very similar. Also, it is interesting to note that as the number of processors increases and the parallel regions are completed quicker, the serial overhead begins to dominate the calculation. This is illustrated in Figure 2. The curve between *l502.exe* and the ones that correspond to the entire run are indistinguishable up to 8 processors. This may indicate that the optimal number of processors for

this particular case is between 4 and 8 processors on POWER4. From Table 1, we see that the efficiency of 0.90 for *l502.exe* goes down to 0.76 for the entire calculation. Larger basis sets are required to maintain a very high efficiency with a large number of processors. In other words, if the molecule remains the same (number and type of atoms), to maintain a high efficiency we need to increase the basis sets to increase the number of integrals that are computed by each processor.

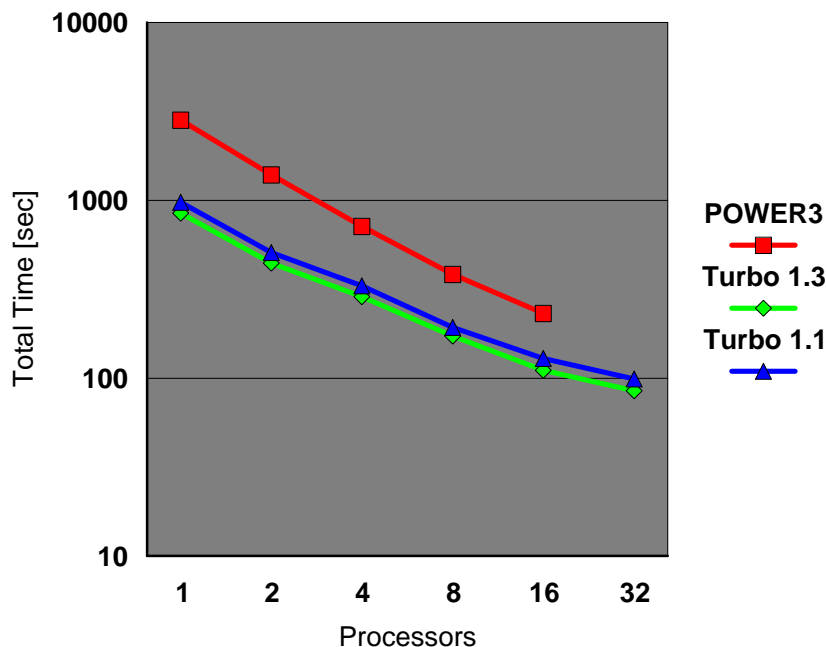


Figure 3. Total time as a function of processors for the Hartree-Fock single-point energy calculation on a-pinene.

Table 2 summarizes a FORCE calculation. In addition to *l502.exe*, this case also illustrates the scalability for *l703.exe* and of course, the scalability for the complete run when using DFT methods. Although these systems (and basis sets) are not the same, an approximate comparison may be obtained between *l502.exe* in Table 1 and Table 2.

Table 2. B3-LYP FORCE Calculation on Valinomycin (C₅₄H₉₀N₆O₁₈)^a

Number of Processors	<i>l502.exe</i> ^b	S ^c	<i>l703.exe</i> ^b	S ^c	Total ^{b,d}	S ^c
1						
POWER3	36,597	1.00	10,343	1.00	47,084	1.00
Turbo 1.3	12,437	1.00	3,039	1.00	15,541	1.00
Turbo 1.1	14,231	1.00	3,502	1.00	17,806	1.00
2						
POWER3	19,845	1.84	4,947	2.09	24,933	1.89
Turbo 1.3	6,509	1.91	1,651	1.84	8,222	1.89
Turbo 1.1	7,603	1.87	1,805	1.94	9,478	1.88
4						
POWER3	11,160	3.28	2,758	3.75	14,069	3.35
Turbo 1.3	4,421	2.81	1,095	2.78	5,583	2.78
Turbo 1.1	5,122	2.78	1,261	2.78	6,458	2.76

8						
POWER3	6,319	5.79	1,589	6.51	8,073	5.83
Turbo 1.3	2,692	4.62	626	4.85	3,389	4.59
Turbo 1.1	3,137	4.54	739	4.74	3,957	4.50
16						
POWER3	4,044	9.05	995	10.39	5,249	8.97
Turbo 1.3	1,788	6.96	398	7.64	2,273	6.84
Turbo 1.1	2,092	6.80	471	7.44	2,661	6.69
32						
Turbo 1.3	1,370	9.08	294	10.34	1,777	8.75
Turbo 1.1	1,606	8.86	349	10.03	2,084	8.54

^a Total of 882 basis functions; basis sets are 3-21G; C1 symmetry.

^b All timings are in seconds and correspond to elapsed time.

^c Speedup.

^d Total time to complete the run.

Similar to the previous case, we look at the single processor performance. The Turbo 1.3 system versus POWER3 shows speedups of 2.94, 3.40, and 3.03 for *l502.exe*, *l703.exe*, and for the total time, respectively. The Turbo 1.1 in this case is slower than the Turbo 1.3 by 14%, 15%, and 15% for *l502.exe*, *l703.exe*, and the total time, respectively. The results between the Turbo 1.3 and Turbo 1.1 are consistent with the previous case. Figure 3 shows scalability for the entire run.

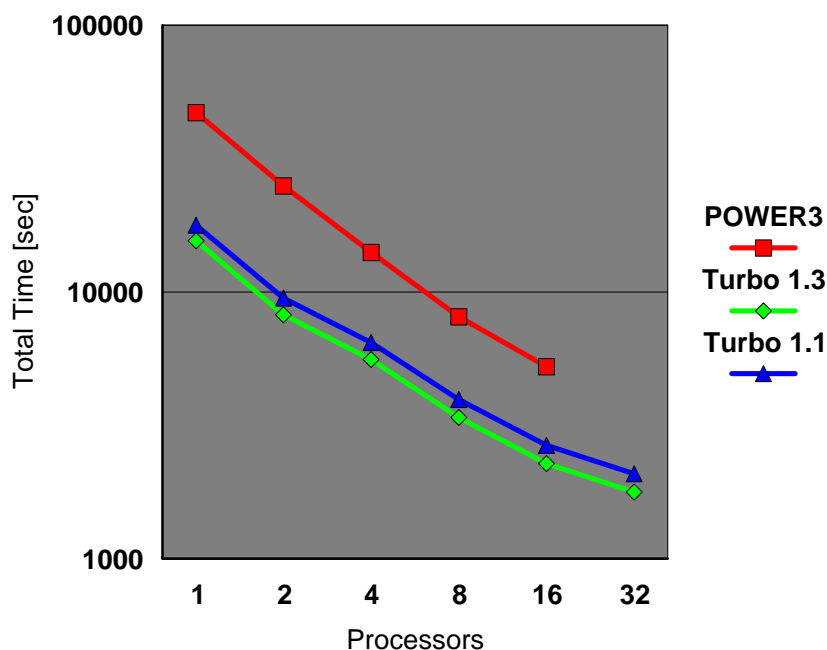


Figure 3. Total time as a function of processors for the B3-LYP FORCE calculation on Valinomycin.

In the case of *l502.exe* the parallel results are similar to α -pinene, although, this is a larger system and has more basis functions but the basis sets are only 3-21G (smaller). This case shows slightly lower scalability than the previous case which may be attributed to less integrals per batch; either due to the small basis sets or cutoff used for the integrals for large systems (loop sizes to compute

integrals are also shorter). The scalability for the two Turbo machines is similar in this case as well.

l703.exe computes the first and second-derivatives of the two-electron integrals. In this case we see that the POWER3 shows superlinear scalability with two processors and very good scalability up to 16 processors. The efficiency on the POWER3 with 16 processors is 65%. On the other hand, the efficiency on the Turbo 1.3 or Turbo 1.1 is about 50%. Again, this might be attributed to faster processor.

Table 3 summarizes timings for a frequency calculation. In Case III, there are several parallel links that are exercised in this example: *l502.exe*, *l1110.exe*, *l1002.exe*, and *l703.exe*. *l502.exe* and *l703.exe* have been discussed previously. *l1002.exe* solves the couple-perturbed Hartree-Fock (CPHF) equations to produce the derivatives of the molecular orbital coefficients. And *l1110.exe* computes the two-electron contribution to the Fock matrix derivatives with respect to nuclear coordinates [13]. Since only the direct scheme of the atomic orbitals (AO) production integrals is parallelized, it is not surprising that *l1002.exe* does not show the same type of scalability as *l703.exe* and *l1110.exe* [3].

Table 3. B3-LYP Frequency Calculation on α -pinene (C₁₀H₁₆)^a

Number of Processors	<i>l502.exe</i> ^b	<i>l1110.exe</i> ^b	<i>l1002.exe</i> ^b	<i>l703.exe</i> ^b	Total ^{b,d}	S ^c
1						
POWER3	1040	2862	3747	3494	11156	1.00
Turbo 1.3	355	1244	1469	1531	4603	1.00
Turbo 1.1	412	1466	1720	1789	5393	1.00
2						
POWER3	510	1426	1956	1746	5654	1.97
Turbo 1.3	182	631	777	807	2403	1.92
Turbo 1.1	212	746	920	924	2809	1.92
4						
POWER3	259	776	1080	959	3095	3.60
Turbo 1.3	108	369	477	454	1416	3.25
Turbo 1.1	125	419	550	533	1636	3.30
8						
POWER3	140	420	656	550	1802	6.19
Turbo 1.3	63	202	313	267	857	5.37
Turbo 1.1	72	232	362	312	992	5.44
16						
POWER3	80	218	452	294	1109	10.06
Turbo 1.3	37	109	227	145	541	8.51
Turbo 1.1	42	127	64	169	628	8.59
32						
Turbo 1.3	25	62	192	86	406	11.34
Turbo 1.1	28	72	222	100	471	11.45

^a Total of 346 basis functions; basis sets are 6-31G(d); C1 symmetry.

^b All timings are in seconds and correspond to elapsed time.

^c Speedup.

^d Total time to complete the run.

In the case of the POWER3, while *l502.exe* and *l1110.exe* show superlinear scalability with 2 processors, the speedup for *l1002.exe* is 1.92. On the Turbo 1.3 and Turbo 1.1, although, we do not see superlinear scalability the trend is the same with 2 processors as on the POWER3. Figure 5 illustrates different scalabilities for the various links in this calculation. Parallelization of additional terms in the solution of CPHF equations will help the scalability of *l1002.exe* [17].

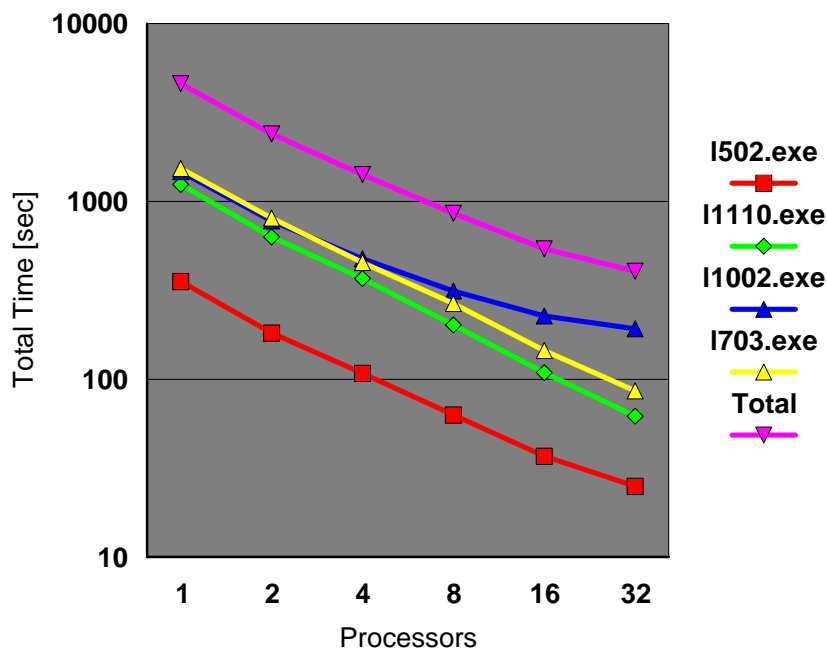


Figure 5. Scalability as a function of processors for the different links in the B3-LYP frequency calculation for α -pinene.

On the other hand *l1110.exe* shows excellent scalability, similar to *l703.exe*. *l1110.exe* shows an almost linear scalability up to 16 processors. S is 13.13, 11.41, and 11.54 (speedups for individual links not shown in Table 3) for the POWER3, Turbo 1.3, and Turbo 1.1, respectively.

Finally, Figure 6 summarizes the performance of the time to solution, that is, the time that it takes to complete the entire calculation. The figure also illustrates (visually) the difference in performance between the POWER3 and the POWER4.

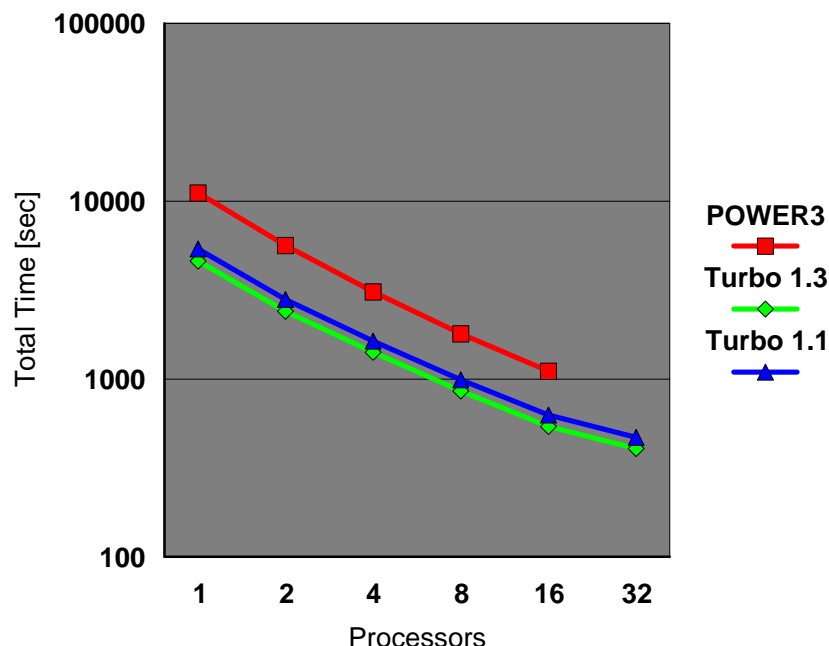


Figure 6. Total time as a function of processors for the B3-LYP frequency calculation on α -pinene.

Table 4 summarizes elapsed timings and total speedups for acetyl-phenol (Case IV). This benchmark consists of a CI-singles energy and FORCE (gradients) calculation. This example illustrates the scalability of *1914.exe*. *1914.exe* computes excited states using CI-singles excitations [19]. This type of calculation runs in parallel since the repulsion two-electron integrals contributing to the CI-singles can be computed using the PRISM algorithm. The PRISM algorithm runs in parallel as shown in the previous sections.

Table 4. CI-Singles Energy and Force Calculation on Acetyl Phenol ($C_8H_8O_2$).

Number of Processors	1502.exe ^b	1914.exe ^b	11002.exe ^b	1703.exe ^b	Total ^{b,d}	S ^c
1						
POWER3	780	1323	694	177	2983	1.00
Turbo 1.3	231	432	205	52	924	1.00
Turbo 1.1	267	497	237	60	1065	1.00
2						
POWER3	347	655	339	87	1441	2.07
Turbo 1.3	110	225	107	27	476	1.94
Turbo 1.1	126	257	124	31	544	1.96
4						
POWER3	201	343	175	44	783	3.81
Turbo 1.3	79	147	68	17	319	2.90
Turbo 1.1	90	167	78	22	364	2.93
8						
POWER3	110	184	90	24	439	6.79
Turbo 1.3	49	92	40	11	203	4.55

Turbo 1.1	56	103	46	12	230	4.63
16						
POWER3	67	106	48	14	290	10.29
Turbo 1.3	30	58	21	6	135	6.84
Turbo 1.1	38	63	26	8	157	6.78
32						
Turbo 1.3	30	39	21	4	130	7.11
Turbo 1.1	38	43	26	5	151	7.05

^a Total of 154 basis functions; basis sets are 6-31++G; C1 symmetry.

^b All timings are in seconds and correspond to elapsed time.

^c Speedup.

^d Total time to complete the run.

This link shows (see Table 4) good scalability consistent with *l502.exe*. Clearly, this case is becoming too small for the POWER4. All the links with 16 processors take less than 60 seconds. *l703.exe* takes only 5 seconds. Furthermore, this is illustrated in Figure 7, the scalability in going from 16 to 32 processors is almost flat. Figure 7 illustrates the total time as a function of processors for the CI-singles energy and gradients.

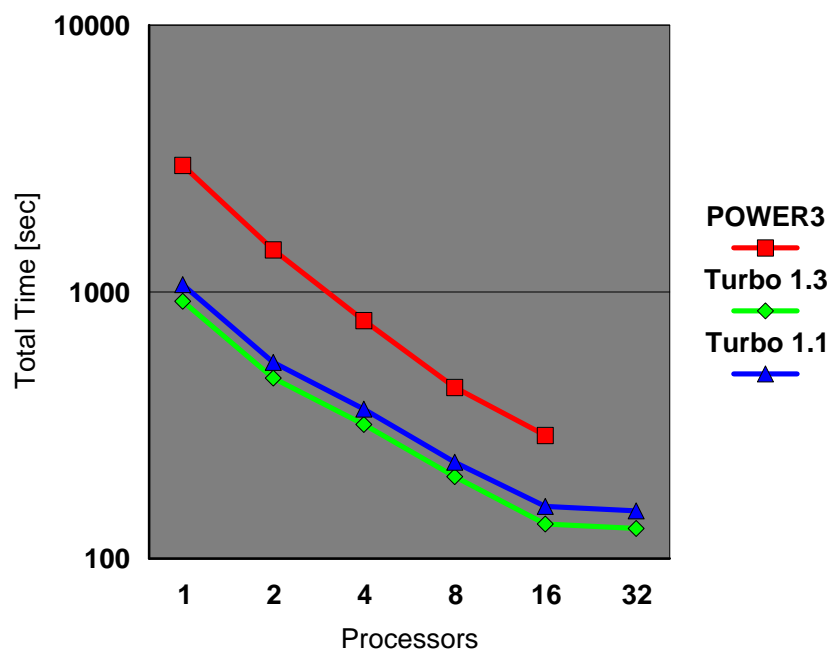


Figure 7. Total time as a function of processors for the CI-singles energy and FORCE calculation on acetyl Phenol.

System Resource Management

Although system resource management is indirectly related to performance, the optimal use of the computer resources will affect the performance of a calculation, either by enhancing or lessening it. Thus, we feel that it is important to include this section in this type of study, especially since *Gaussian* can make optimal use of disk space, memory, and multiple processors. In this study we shall concentrate in looking at memory optimization with respect to the number of processors. Memory allocation when using only 1 processor has been discussed in refs. [7,19] and shall not be covered here.

Gaussian uses a replicated data algorithm for the parallelization of the two-electron integrals [2]. On shared-memory machines this means that allocation of additional memory is required to run parallel calculations. The amount of memory required is directly proportional to the number of processors. In a previous study we have successfully used eq. (5) to optimally allocate memory when using multiple processors [9].

$$\text{Memory (N}_{\text{procs}} \text{ run)} = \text{Memory (1}_{\text{proc}} \text{ run)} + 0.75*(\text{N}_{\text{procs}}-1)*\text{Memory (1}_{\text{proc}} \text{ run)} \quad (5)$$

where memory (N_{procs} run) is the final amount of memory (either in MW or MB) that will be allocated by the %mem keyword in the input file. Memory (1_{proc} run) is the *minimal* amount of memory required to run a particular calculation on a single processor. N_{procs} is the total number of processors that will be used in a particular calculation. For instance, in Case I, we allocated 8MW for a single processor run and 98 MW for the 16 processors [20]. Prior to computing the two-electron integrals and its derivatives (if applicable) in parallel, *Gaussian* checks if there is enough memory to run in parallel. In cases where not enough memory has been allocated, the number of processors requested will be reduced until they fit in the provided amount of memory.

Summary

In this study we have tried to provide information on the performance of several options in the *Gaussian* program. These options are commonly used by researchers at many supercomputer centers but by no means is this an exhaustive set of benchmarks. Also, in this first exploratory work we have not looked at post-Hartree-Fock options in *Gaussian*. The levels of theory tested here correspond to Hartree-Fock, Density Functional Theory and CI-singles excitations. In part this is due to the fact that they tend to show larger scalability with larger number of processors than post-Hartree-Fock methods. As first approximations to many higher-order methods, this set of methods is extensively used and certainly can provide valuable information when comparing the newest POWER architecture.

In general, SCF and DFT calculations with and without first and/or second derivatives have shown to run efficiently on the machines tested in this study. However, when designing benchmarks for scalability it is important to consider the system and basis sets. The first case that we studied here, a-pinene, has been used as a system that scales well with large number of processors [9]. A comparison of the scalability of *1502.exe* for Cases I-III gives an approximate indication (it is approximate because each case uses a different level of theory with slightly different options and different convergence thresholds) that is in order to achieve better scalability, basis set size is more important than either the level of theory or size of the molecule.

In terms of the three computer systems tested in this study, the speedup in going from the POWER3 to the POWER4 is very impressive. For these cases (Cases I-IV) the speedup is between 2.4 and 3.4. Although the parallel scalability between these systems is similar, the POWER3 tends to show parallel speedups for 16 processors that are between 14 - 29 % higher than the POWER4.

Finally, the last case, Case IV provides a direct method of computing excited states with minimal or no disk storage required. Due to its simplicity, this method provides a good approximation for large systems. This and other studies have shown that this is a good method to take advantage of multiple processors. *l914.exe* shows very good parallel speedups.

Acknowledgments

We would like to thank S. Behling for reading this manuscript and making valuable comments. We also thank R. Panda, S. Selzo, and S. Qualters for all their help with the "Regatta" systems. CPS would like to thank Bruce Hurley for encouragement and making this type of study possible.

References

- [1] Gaussian News, **4**, 1(1993), Gaussian, Inc., Pittsburgh, PA.
- [2] R. J. Harrison and R. Shepard, *Annu. Rev. Phys. Chem.* **45**, 623(1994).
- [3] C. P. Sosa, J. Ochterski, J. Carpenter, and M. J. Frisch, *J. Comp. Chem.* **19**, 1053(1998); and references therein.
- [4] M. J. Flynn, *Proc. of the IEEE* **54**, 1901(1966).
- [5] M. J. Flynn, *IEEE Trans. on Computers* **C21**, 948(1972).
- [6] S. Behling, R. Bell, P. Farrell, A. Holthoff, F. O'Connell, and W. Weir *The POWER4 Processor Introduction and Tuning Guide*, IBM Corporation, International Technical Support Organization, Austin TX, 2001; and references therein.
- [7] J. Ochterski, C. P. Sosa, J. Carpenter, Performance of Parallel Gaussian94 on Cray Research Vector Supercomputers, in: B. Winget and K. Winget, Editors, Cray User Group Proceedings, Cray User Group, Shepherdstown, WV, p. 108, 1996.
- [8] D. P. Turner, G. W. Trucks, M. J. Frisch, Ab Initio Quantum Chemistry on a Workstation Cluster, in: T. G. Matson, Editor, *Parallel Computing in Computational Chemistry*, ACS Series 592, American Chemical Society, Washington, DC, p. 62.
- [9] C. P. Sosa, G. Scalmani, R. Gomperts, and M. J. Frisch, *Parallel Computing* **26**, 843(2000).
- [10] H. M. Matis, J. D. McCalpin, M-C, Chiang, F. P. O'Connell, P. Buckland IBM pSeries 690 Configuring for Performance, IBM Corporation, Austin, TX, 2001.
- [11] XL Fortran for AIX®, *User's Guide Version 7 Release 1*, IBM Canada, North York, Ontario, Canada, SC09-2866-00, 2000.
- [12] AE. Frisch and M. J. Frisch, *Gaussian98 User's Reference*, 2nd Edition, Gaussian Inc., Pittsburgh, PA; <http://www.gaussian.com>
- [13] M. J. Frisch, A. B. Nielsen, and AE. Frisch, *Gaussian98 Programmer's Reference*, Gaussian Inc., Pittsburgh, PA; <http://www.gaussian.com>
- [14] P. M. Gill, M. Head-Gordon, and J. A. Pople, *J. Phys. Chem.*, **94**, 5564(1990).
- [15] [W. J. Hehre, L. Radom, P. V. R. Schleyer, and J. A. Pople, *Ab Initio Molecular Orbital Theory*, John Wiley & Sons, New York, NY, 1985.
- [16] B. G. Johnson, P. M. W. Gill, and J. A. Pople, *J. Chem. Phys.*, **98**, 5612(1993).
- [17] C. P. Sosa and S. Andersson, submitted for publication.
- [18] A. D. Becke, *J. Chem. Phys.*, **98**, 5648(1993).

- [19] J. B. Foresman, M. Head-Gordon, J. A. Pople, and M. J. Frisch, *J. Phys. Chem.* **96**, 135(1992).
- [20] To obtain all the input files used in this study send an e-mail message to:
cpsosa@msi.umn.edu
- [21] C. P. Sosa and M. J. Frisch, in preparation.

Special Notices

This document was produced in the United States. IBM may not offer the products, programs, services or features discussed herein in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the products, programs, services, and features available in your area. Any reference to an IBM product, program, service or feature is not intended to state or imply that only IBM's product, program, service or feature may be used. Any functionally equivalent product, program, service or feature that does not infringe on any of IBM's intellectual property rights may be used instead of the IBM product, program, service or feature. IBM makes no representation or warranty regarding third-party products or services.

Information in this document concerning non-IBM products was obtained from the suppliers of these products, published announcement material or other publicly available sources. Sources for non-IBM list prices and performance numbers are taken from publicly available information including D.H. Brown, vendor announcements, vendor WWW Home Pages, SPEC Home Page, GPC (Graphics Processing Council) Home Page and TPC (Transaction Processing Performance Council) Home Page. IBM has not tested these products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of a specific Statement of General Direction.

The information contained in this document has not been submitted to any formal IBM test and is distributed "AS IS". While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. The use of this information or the implementation of any techniques described herein is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. Customers attempting to adapt these techniques to their own environments do so at their own risk.

IBM is not responsible for printing errors in this publication that result in pricing or information inaccuracies.

The information contained in this document represents the current views of IBM on the issues discussed as of the date of publication. IBM cannot guarantee the accuracy of any information presented after the date of publication.

All prices shown are IBM's suggested list prices; dealer prices may vary.

IBM products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Information provided in this document and information contained on IBM's past and present Year 2000 Internet Web site pages regarding products and services offered by IBM and its subsidiaries are "Year 2000 Readiness Disclosures" under the Year 2000 Information and Readiness Disclosure Act of 1998, a U.S. statute enacted on October 19, 1998. IBM's Year 2000 Internet Web site pages have been and will continue to be our primary mechanism for communicating year 2000 information. Please see the "legal" icon on IBM's Year 2000 Web site (<http://www.ibm.com/year2000>) for further information regarding this statute and its applicability to IBM.

Any performance data contained in this document was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be

the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Trademarks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM, e(logo), pSeries, RS/6000

IBM Trademarks information can be found at: <http://www.ibm.com/legal/copytrade.shtml>.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SPEC, SPECjbb, SPECint, SPECfp, SPECweb, and SPECsfs are trademarks of the Standard Performance Evaluation Corporation and information can be found at: <http://www.spec.org>.

TPC, TPC-R, TPC-H, TPC-C, and TPC-W are trademarks of the Transaction Processing Performance Council. Information can be found at: <http://www.tpc.org>.

Other trademarks are the property of their respective owners.

Notes on Benchmarks and Values

The benchmarks and values shown here were derived using particular, well configured, development-level computer systems. Unless otherwise indicated for a system, the values were derived using 32-bit applications and external cache, if external cache is supported on the system. All benchmark values are provided "AS IS" and no warranties or guarantees are expressed or implied by IBM. Actual system performance may vary and is dependent upon many factors including system hardware configuration and software design and configuration. Buyers should consult other sources of information to evaluate the performance of systems they are considering buying and should consider conducting application oriented testing. For additional information about the benchmarks, values and systems tested, contact your local IBM office or IBM authorized reseller or access the following on the Web:

TPC	http://www.tpc.org
GPC	http://www.spec.org/gpc
SPEC	http://www.spec.org
Pro/E	http://www.proe.com
Linpack	http://www.netlib.no/netlib/benchmark/performance.ps
Notesbench Mail	http://www.notesbench.org
VolanoMark	http://www.volano.com
Fluent	http://www.fluent.com
Gaussian	http://www.gaussian.com

Unless otherwise indicated for a system, the performance benchmarks were conducted using AIX V4.2.1 or 4.3, IBM C Set++ for AIX/6000 V4.1.0.1, and AIX XL FORTRAN V5.1.0.0 with optimization where the compilers were used in the benchmark tests. The preprocessors used in the benchmark tests include KAP 3.2 for FORTRAN and KAP/C 1.4.2 from Kuck & Associates and VAST-2 v4.01X8 from Pacific-Sierra Research. The preprocessors were purchased separately from these vendors.

The following SPEC and Linpack benchmarks reflect the performance of the microprocessor, memory architecture, and compiler of the tested system:

- SPECint95 - SPEC component-level benchmark that measures integer performance. Result is the geometric mean of eight tests that comprise the CINT95 benchmark suite. All of these are written in the C language. SPECint_base95 is the result of the same tests as CINT95 with a maximum of four compiler flags that must be used in all eight tests.
- SPECint_rate95 - Geometric average of the eight SPEC rates from the SPEC integer tests (CINT95). SPECint_base_rate95 is the result of the same tests as CINT95 with a maximum of four compiler flags that must be used in all eight tests.

- SPECfp95 - SPEC component-level benchmark that measures floating-point performance. Result is the geometric mean of ten tests, all written in FORTRAN, that are included in the CFP95 benchmark suite. SPECfp_base95 is the result of the same tests as CFP95 with a maximum of four compiler flags that must be used in all ten tests.
- SPECfp_rate95 - Geometric average of the ten SPEC rates from SPEC floating-point tests (CFP95). SPECfp_base_rate95 is the result of the same tests as CFP95 with a maximum of four compiler flags that must be used in all ten tests.
- SPECint2000 - New SPEC component-level benchmark that measures integer performance. Result is the geometric mean of twelve tests that comprise the CINT2000 benchmark suite. All of these are written in C language except for one which is in C++. SPECint_base2000 is the result of the same tests as CINT2000 with a maximum of four compiler options that must be used in all twelve tests.
- SPECint_rate2000 - Geometric average of the twelve SPEC rates from the SPEC integer tests (CINT2000). SPECint_base_rate2000 is the result of the same tests as CINT2000 with a maximum of four compiler options that must be used in all twelve tests.
- SPECfp2000 - New SPEC component-level benchmark that measures floating-point performance. Result is the geometric mean of fourteen tests, all written in FORTRAN and C languages, that are included in the CFP2000 benchmark suite. SPECfp_base2000 is the result of the same tests as CFP2000 with a maximum of four compiler options that must be used in all fourteen tests.
- SPECfp_rate2000 - Geometric average of the fourteen SPEC rates from SPEC floating-point tests (CFP2000). SPEC_base_rate2000 is the result of the same tests as CFP2000 with a maximum of four compiler options that must be used in all fourteen tests.
- SPECweb96 - Maximum number of Hypertext Transfer Protocol (HTTP) operations per second achieved on the SPECweb96 benchmark without significant degradation of response time. The Web server software is ZEUS v.1.1 from Zeus Technology Ltd.
- SPECweb99 - Number of conforming, simultaneous connections the Web server can support using a predefined workload. The SPECweb99 test harness emulates clients sending the HTTP requests in the workload over slow Internet connections to the Web server. The Web server software is Zeus from Zeus Technology Ltd.
- LINPACK DP (Double Precision) - n=100 is the array size. The results are measured in megaflops (MFLOPS).
- LINPACK SP (Single Precision) - n=100 is the array size. The results are measured in MFLOPS.
- LINPACK TPP (Toward Peak Performance) - n=1,000 is the array size. The results are measured in MFLOPS.
- LINPACK HPC (Highly Parallel Computing) - solve largest system of linear equations possible. The results are measured in GFLOPS.

VolanoMark is a 100% Pure Java™ server benchmark characterized by long-lasting network connections and high thread counts. In this context, long-lasting means the connections last several minutes or longer, rather than just a few seconds. The VolanoMark benchmark creates client connections in groups of 20 and measures how long it takes for the clients to take turns broadcasting their messages to the group. At the end of the test, it reports a score as the average number of messages transferred by the server per second.

VolanoMark 2.1.2 local performance test measures throughput in messages per second. The final score is the average of the best two out of three results.

The following SPEC benchmark reflects the performance of the microprocessor, memory subsystem, disk subsystem, network subsystem:

- SPECsfs97_R1 - the SPECsfs97_R1 (or SPEC SFS 3.0) benchmark consists of two separate workloads, one for NFS V2 and one for NFS V3, which report two distinct metrics, SPECsfs97_R1.v2 and SPECsfs97_R1.v3, respectively. The metrics consist of a throughput component and an overall response time measure. The throughput (measured in operations per second) is the primary component used when comparing SFS performance between systems. The overall response time (average response time per operation) is a measure of how quickly the server responds to NFS operation requests over the range of tested throughput loads.

The following Transaction Processing Performance Council (TPC) benchmarks reflect the performance of the microprocessor, memory subsystem, disk subsystem, and some portions of the network:

- tpmC - TPC Benchmark C throughput measured as the average number of transactions processed per minute during a valid TPC-C configuration run of at least twenty minutes.
- \$/tpmC - TPC Benchmark C price/performance ratio reflects the estimated five year total cost of ownership for system hardware, software, and maintenance and is determined by dividing such estimated total cost by the tpmC for the system.
- QppH is the power metric of TPC-H and is based on a geometric mean of the 17 TPC-H queries, the insert test, and the delete test. It measures the ability of the system to give a single user the best possible response time by harnessing all available resources. QppH is scaled based on database size from 30GB to 1TB.

- QthH is the throughput metric of TPC-H and is a classical throughput measurement characterizing the ability of the system to support a multiuser workload in a balanced way. A number of query users is chosen, each of which must execute the full set of 17 queries in a different order. In the background, there is an update stream running a series of insert/delete operations. QthH is scaled based on the database size from 30GB to 1TB.
- $S/QphH$ is the price/performance metric for the TPC-H benchmark where QphD is the geometric mean of QppH and QthH. The price is the five-year cost of ownership for the tested configuration and includes maintenance and software support.

The following graphics benchmarks reflect the performance of the microprocessor, memory subsystem, and graphics adapter:

- SPECxpc results - Xmark93 is the weighted geometric mean of 447 tests executed in the x11perf suite and is an indicator of 2D graphics performance in an X environment. Larger values indicate better performance.
- SPECplb results (graPHIGS) - PLBwire93 and PLBsurf93 are geometric means of literal and optimized Picture Level Benchmark (PLB) tests for 3D wireframe and 3D surface tests, respectively. The benchmark and tests were developed by the Graphics Performance Characterization (GPC) Committee. The results shown used the graPHIGS API. Larger values indicate better performance.
- SPECopc results - CDRS-03, CDRS-04, DX-03, DX-04, DX-05, DRV-04, DRV-05, DRV-06, Light-01, Light-02, Light-02, AWadv-01, AWadv-02, AWadv-03, and ProCDRS-02 are weighted geometric means of individual viewset metrics. The viewsets were developed by ISVs (independent software vendors) with the assistance of OPC (OpenGL Performance Characterization) member companies. Larger values indicate better performance.

The following graphics benchmarks reflect the performance of the microprocessor, memory subsystem, graphics adapter, and disk subsystem:

Bench95 and Bench97 Pro/E results - Bench95 and Bench97 Pro/E benchmarks have been developed by Texas Instruments to measure UNIX and Windows NT[®] workstations in a comparable real-world environment. Results shown are in minutes. Lower numbers indicate better performance.

The NotesBench Mail workload simulates users reading and sending mail. A simulated user will execute a prescribed set of functions 4 times per hour and will generate mail traffic about every 90 minutes. Performance metrics are:

- NotesMark - transactions/minute (TPM).
- NotesBench users - number of client (user) sessions being simulated by the NotesBench workload.
- $S/NotesMark$ - ratio of total system cost divided by the NotesMark (TPM) achieved on the Mail workload.
- $S/User$ - ratio of total system cost divided by the number of client sessions successfully simulated for the Mail NotesBench workload measured.

Total system cost is the price of the server under test to the customer, including hardware, operating system, and Domino Server licenses.