An Inexpensive Platform For Computational Chemistry Applications

Haibo Wang*, Kenneth Rossetti* and Amanda W. Wu+

Abstract. We built up a Linux cluster with inexpensive disk and memory to carry out the large Gaussian 98 and GAMESS jobs, which cannot be finished during the regular system maintenance cycle and these jobs are unable to restart from checkpoint file. Performance benchmark and tuning have been carried out and a model for the production system has been tested and planned.

1. Introduction

The Mississippi Center for Supercomputing Research (MCSR) provides high performance computing resources and technical support services to researchers at Mississippi's eight public universities. Currently we have a vector machine Cray C916, and a parallel machine SGI Origin 2800, they are saturated with around-the-clock research computing jobs. In order to help us distribute some of the load from the supercomputers within the limited budget, we are looking for a low-cost alternative parallel computing environment for our user community. On the other hand, there are many research group in our user community can afford a low cost system. We can share our experience with our user community.

Compared with SMP system, cluster systems are low cost systems both on building a new system and on adding more nodes to increase the system performance. For example, a Linux Beowulf Cluster can be upgraded very easily by adding new nodes on the fly. If the low cost cluster system can offer a reasonable performance on parallel computing, it will greatly reduce the cost on parallel computing.

In this paper, we will share our experiences of installing Gaussian 98 and GAMESS on the Linux Beowulf Cluster. The results of benchmark indicate that we can achieve a very good speedup on the cluster in the certain range of the computational jobs. To compare the price/performance ratio, we also submit the computational jobs to the SGI Origin 2800. The SGI Origin 2800 performances better on the very large jobs that need a lot of memory and disk spaces. The cluster can gain a better linear speedup and efficiency on the small jobs with a lot of iterations.

In section 2, we will describe the system configuration and application software. Section 3 provides detail of our testing applications and design of the experiment. In section 4, we will present the results of benchmark and compare the performance of two configurations. Section 5 presents the overview of the study.

---

* MCSR, University of Mississippi, University, MS 38677, chwang@olemiss.edu, kenr@olemiss.edu
+ Corporate Service, FedEx Corporation, Collierville, TN, 38027, wanhong.wu@fedex.com

2. System Configuration

Beowulf is a term used by NASA to describe the strategy of clustering multiple small, standalone servers on a fast, private network segment, as a virtual multi-processing computing environment[1]. One node acts as the master and it is visible on the external network. The other compute nodes act as slave. The master distributes workload to compute nodes. This approach makes easy on scalability, upgrading can be simply handled by adding more computers to the cluster.

2.1 MCSR Linux Beowulf Cluster System

MCSR installed a 16-nodes Linux Cluster at the beginning of last year. This system is a collection of Intel processor-based workstations and server interconnected by a TCP/IP-based network. This cluster consists of the following:

A front-end (master) node for interactive use, compilation, job submission and applications installation; 16 compute nodes used by parallel jobs; a fast network for inter-node communication and a external network access for the user to connect to the front-end node.

We initially loaded them all with the RedHat Linux 6.1 operating system, but the network driver in RedHat 6.1 created the same MAC address for all 16 nodes. We were eventually able to solve this addressing problem by upgrading to RedHat 6.2. We installed network interface cards (NICs) in the master and compute nodes, and connected them to private 100Mbps Fast Ethernet network segment. We then connected the server to two other networks: the public network, and a private backup network.

RedHat 6.2 includes the GNU compilers such as gcc, g++ and g77. But the performance of code generated by these compilers is not optimized as the commercial packages. So we chose a Portland Group compiler suite to manage the cluster. The version of the Portland Group compiler suite is PGI CDK 3.1 Cluster Development Kit. Later, we upgraded the compiler package to PGI CDK 3.2.

2.2 System Administration

We configured the front-end node to function as a job clearinghouse only, dispatching jobs to nodes, but not running any jobs. The master node would also be the file server for the cluster. The /home directory on the master was NFS mounted to the /home directories on the compute node, it help user to locate files or program easily. There is a copy of user database on each compute node and a rhost file under each user's home directory. So user can access to the node easily. Besides the home directory on the master node, we also configured the disk on one node as /ptmp directory for user to store the scratch files. The Gaussian and GAMESS user would use the scripts provided by the MCSR consultants and submit the jobs to the PBS queue. All the application software is installed to a directory on the front-end node which can be shared with compute nodes using NFS.

3.Gaussian 98 and GAMESS

The performance issues on parallel processing and computational chemistry application will give us useful information of real-world applications. Gaussian 98 is the most popular computational chemistry application used by the MCSR users and also one of the most popular applications on most supercomputer systems. GAMESS is also very popular to the computational chemistry application among the MCSR users. Parallel version GAMESS can get almost perfect speedup on the Beowulf cluster[4].

3.1 Gaussian 98 and Linda

Gaussian 98 is a system of programs for performing a variety of semi-empirical and *ab initio* molecular orbital calculations[2]. It can predict the molecular energies and structures including transition states, calculate the frequencies and other thermo chemical properties. Gaussian 98 can be installed on a broad range of UNIX systems and PC running Windows. It is very easy to install Gaussian 98 on the SGI Origin 2800 even though it is a multiprocessors system. All we need is to run the installation script and set up the Gaussian 98 execution environment. To install Gaussian 98 on a single processor Linux system is also very easy and just like the SGI Origin 2800 system. But it is complicated when it is installed on the Linux Beowulf Cluster and run as a parallel version. First, Gaussian 98 needs to be built as a regular version. Then we will install Linda software and add the Linda binary to the compiling path. We also need to download two additional packages Finally; we can build the parallel version of Gaussian 98. There is two additional files need to be downloaded and put under /usr/local/lib directory: blas-f2c.a and blas-opt.a.

Linda is a coordination language that brings many independent processes together into a single parallel program. Linda provides a simple, complete command set that enables process creation, synchronization and communication. The combination of Linda provides a virtual shared memory (VSM) that is logically shared by all the processes in a parallel program. Gaussian 98 and Linda make electronic structure methods available across the whole range of parallel computing environments.

After the parallel version of Gaussian 98 is built, we can prepare two sets of scripts for user to use it. If the user chooses PBS to control the job submission, he/she can use the PBS submission script prepared by the MCSR consultants. If the user chooses to run the job interactively, he/she can use the Shell script prepared by the consultants to submit the job. Using PBS submission, user allows PBS to decide the working nodes. On the other hand, user can specify the working nodes (define the “nodelist” as the environment variable) when he/she runs the job in the interactive mode. If the user submits the job to multiple nodes, he/she will see the speedup on the wall clock time. In the input file, user need to use the %Nproc command to specify the number of processors on which the job will run. If user forgets to add %Nproc in the input file, the job will not run in parallel even though the idle Linda process on the nodes other than the one from which user starts the job. The process might abort without any error message.

Not every method in Gaussian 98 can be run under multiprocessors and gain speedup. DFT, HF and MP2 methods can be run in parallel and gain truly linear speedup. Other methods such as
MP4, CBS-X, G2, etc cannot gain speedup because of the disk I/O. All of these methods make use of the MO integrals and all integrals need to be available to all processors, which is unsuitable to the distributed system. This makes careful organization of data and maximal use of every integral necessary for an efficient implementation. This is the inverse of the current implementation where allowing for small memory emphasized high reliance on disk storage and access. The current status includes methods like CASSCF and CIS=DIRECT. Parts of the post-HF methods can theoretically be done in parallel but MP2 energy and gradients are really the current limit. In this study, we compared the speedup of different Gaussian 98 methods on the same molecule between two platforms to evaluate the ratio of price/performance.

For a Beowulf Cluster system with a large number of users and a large number of nodes, there are many advantages to use PBS or other queuing software to control job loading, user priority and nodes configuration. We found out four nodes getting best speedup. So we can configure four-nodes queue for most Gaussian 98 users. The MCSR user to submit the job through PBS uses the following script:

```
qsub << EOF
#!/bin/sh
#PBS -l nodes=$1
#PBS -N g98test -o g98test.log
#PBS -j oe
#PBS -l cput=$2
#PBS -l mem=$3
#PBS -m be
cd `pwd`

./g98a $4 >& $5
EOF
```

G98a is the script to specify the environment variables such as LINDA_CLC and LINDA_FLC and also print out the wall time for the job. It looks like this:

```
#!/bin/csh -fx
setenv MP_NEWJOB yes
setenv LINDA_CLC network
setenv LINDA_FLC network
setenv g98root /usr/local
source $g98root/g98/bsd/g98.login
setenv GAUSS_EXEDIR $g98root/g98/linda-exe:$g98root/g98
setenv GAUSS_ARCHDIR $g98root/g98/arch
setenv G98BASIS $g98root/g98/basis
setenv GAUSS_SCRDIR /ptmp/$USER/g98.$$
mkdir $GAUSS_SCRDIR
/usr/bin/time $g98root/g98/g98 $argv
```

For the Beowulf Cluster used by small number of the users and managed by the users, user can edit the nodelist file such as .tsnet.config and submit the job from the starting node.

Tsnet.Appl.maxprocspernode: 8
Tsnet.Node.speedfactor: 4
Tsnet.Appl.nodelist: node1 node2 node3 node4

If the user submit the job from node1, the script can be edited as:

```
#!/bin/csh -fx
setenv MP_NEWJOB yes
setenv LINDA_CLC network
setenv LINDA_FLC network
setenv g98root /usr/local
source $g98root/g98/bsd/g98.login
setenv GAUSS_EXEDIR $g98root/g98/linda-exe:$g98root/g98
setenv GAUSS_ARCHDIR $g98root/g98/arch
setenv G98BASIS $g98root/g98/basis
setenv GAUSS_LFLAGS "-nodelist 'node1 node2'"
setenv GAUSS_SCRDIR /work/g98.$$
  mkdir $GAUSS_SCRDIR
#/usr/bin/rsh node1   (if run the script from master node,remove #)
/usr/bin/time $g98root/g98/g98 $argv
```

The job will run under two nodes in parallel with this script. User can specify the nodes to run the job instead of allowing PBS assign the compute nodes. If the job needs more memory than the memory limit of PBS queue, it is better to use above script instead of using PBS. The current version Open PBS we installed on the SGI Origin 2800 and the Beowulf cluster will stop the job if the job needs more memory than the memory specified by the user in the input file. The future version of PBS or PBS Pro might correct this problem.

3.2 Parallel GAMESS

Parallelization is accomplished using Linux's TCP/IP socket library to support DDI[3]. Since our Beowulf cluster is installed with PGI CDK 3.2, we did some modification such as:

(a) in comp: pgf77 -c -tp p6 -fast $MODULE.f
(b) in ddisoc.c, use only one underscore, as indicated.

There are two scripts to submit the GAMESS jobs. These scripts are available from us. The gms script will need to be tuned for any node resource definitions. The rungms script makes use of the dynamic PBS generated nodefile given in the $PBS_NODEFILE environment variable. User will need to modify the script for the scratch directory layout. If the system has multiple communication adapters, user might need to hack ddikick to remove the check that requires the first nodename to be exactly the same as the response of 'hostname'. The rungms script is also available from the GAMESS source code and need to be modified.

4. Results of Benchmark

The speedup of Gaussian 98 jobs on the SGI Origin 2800 cannot be achieved when the system is so heavily loaded. Actually the speedup value is less than 1.0 with the multiple processors on the small size job. For the medium or large size job, the newly installed machine can gain the speedup of 3.2 over 4 processors and the heavily used system only get the speedup of 2.3 over 4 processors or 2.7 over 8 processors. The results of wall clock time, speedup and efficiency are shown in Table 1. In the heavily used system, if the number of processors are larger than 4, the efficiency will be less than 50%. This result can help us define the layout of PBS queues on SGI Origin 2800 system and help us understand how to improve the usage in the system. Since SGI Origin 2800 is a shared memory system, each processor can access all the memory in the system. The wall clock time on finishing the same job on the SGI Origin 2800 system will be less than the one on the Linux Beowulf Cluster. For jobs that require a lot of memory and disk spaces, running on the SGI Origin 2800 will be faster. For jobs that require less memory but need a lot of iterations (long run job), running on the cluster will be more benefit on speedup (Figure 1).

	HC3NH								
	HF 6-311G(2p,2d)			MP2 6-311G(2p, 2d)			CCSD 6-311G(2p, 2d)		
No. Proc	Wall time (second)	Speedup	efficiency	Wall time (second)	Speedup	efficiency	Wall time (second)	speedup	efficiency
1	5054.73	1	100.00%	16168.25	1	100.00%	37128.99	1	100.00%
2	2780.19	1.818	90.91%	9974.33	1.621	81.05%	25937.61	1.431	71.57%
4	2187.14	2.311	57.78%	8122.91	1.991	49.76%	24010.27	1.546	38.66%
8	1845.63	2.739	34.23%	7555.81	2.139	26.75%	22700.63	1.635593	20.44%

Table 1. Speedup of Different Methods on SGI Origin 2800

Jobs	HC3NH			test268			test178		
No. Proc	Wall time (second)	Speedup	efficiency	Wall time (second)	speedup	efficiency	Wall time (second)	speedup	efficiency
1	23339.63	1	100.00%	8742.43	1	100.00%	397.47	1	100.00%
2	12393.66	1.883	94.16%	4456.07	1.961	98.10%	254.03	1.564	78.06%
4	6865.81	3.399	84.98%	2436.44	3.588	89.70%	185.16	2.146	54.17%
8	4198.99	5.558	69.48%	1438.33	6.078	75.98%	148.34	2.679	33.34%

Table 2. The Results of Different Sizes of Gaussian98 Jobs on Cluster

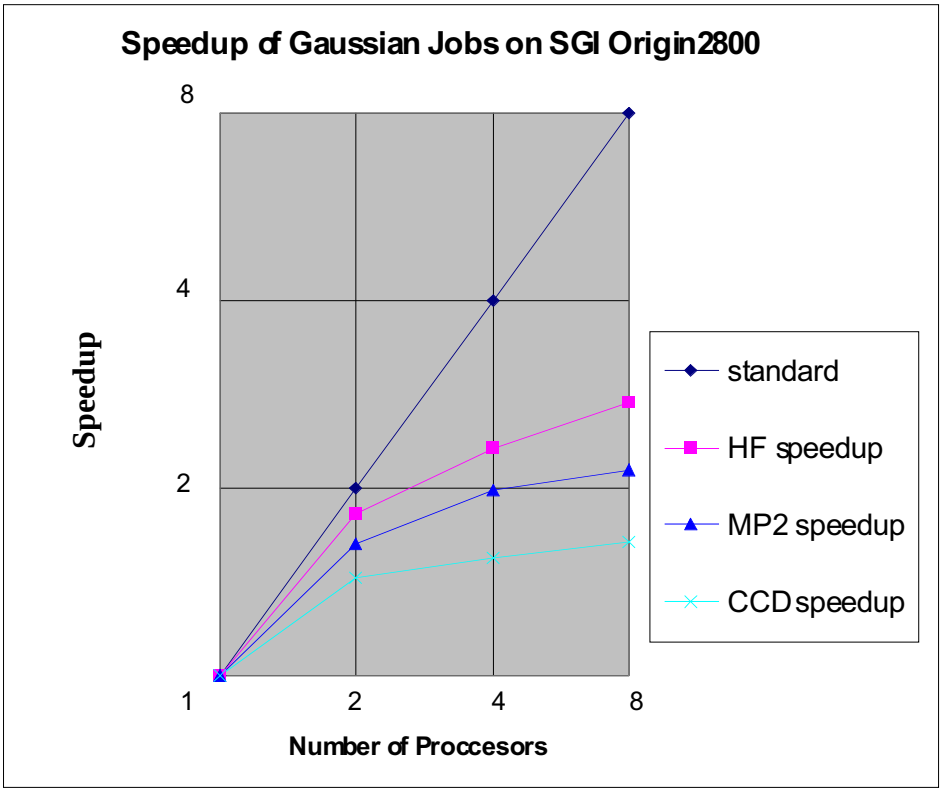

Figure 1. Gaussian98 Speedup of Different Methods on SGI Origin 2800

The speedup results of Gaussian 98 jobs on the cluster are very attractive. We have tested two sets of configurations. First, we tested the speedup on different size of Gaussian 98 jobs with the same method. It shows that the speedup on medium size jobs is better than that on large or small size of jobs (see Figure 2). The results of wall clock time, speedup and efficiency data are shown in Table 2.

In Figure 3, we compared the speedup on different methods with the same molecule on the cluster. The results show that HF method get better speedup than other methods. The speedup value decreases as the level of method increases. The results of wall clock time, speedup and efficiency are shown in Table 3. When the number of nodes is less than 8, the efficiency is larger than 50%. To use the resource efficiently, running parallel Gaussian98 jobs on 4 nodes is the best configuration. For a job with HF method, it can be run up to 8 nodes. For a small size job, it is better run on the queue with less than 4 nodes. For the large size job, it can be run on the queue with 4 nodes. For the medium size job, it can be run on the queue with 8 nodes.

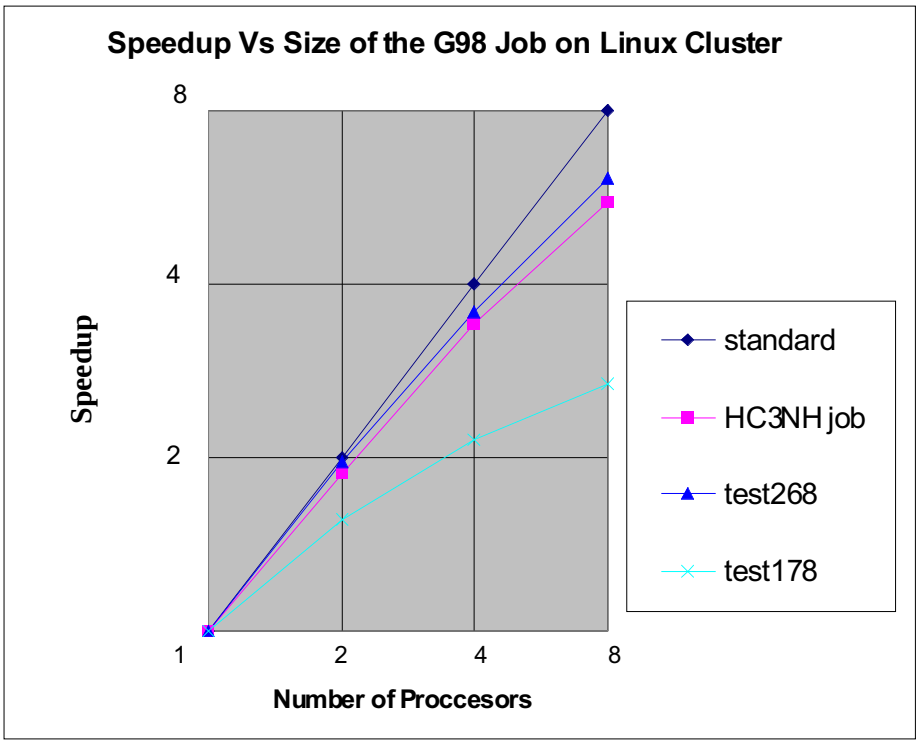

Figure 2. Speedup on Different Size of Gaussian98 Jobs with Cluster

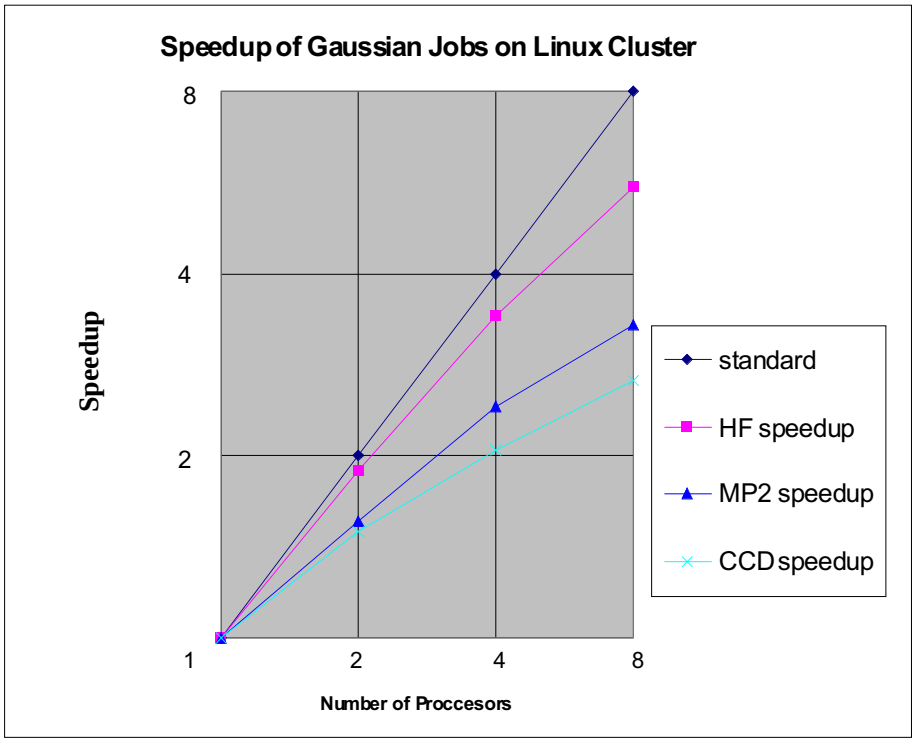

Figure 3. Speedup on Different Methods with Cluster

	HC3NH								
	HF 6-311G(2p,2d)			MP2 6-311G(2p, 2d)			CCSD 6-311G(2p, 2d)		
No. Proc	Wall time (second)	speedup	efficiency	Wall time (second)	Speedup	efficiency	Wall time (second)	speedup	efficiency
1	23339.63	1	100.00%	46730.00	1	100.00%	89109.57	1	100.00%
2	12393.66	1.883	94.16%	29922.99	1.561	78.08%	59656.50	1.493	75.13%
4	6865.81	3.399	84.98%	19368.74	2.412	60.32%	43521.57	2.047	51.32%
8	4198.99	5.558	69.48%	14156.27	3.301	41.26%	33401.25	2.667	33.41%

Table 3. The Results on Different Methods with Same Molecule on Cluster

	HC3NH								
	HF			MP2			CI		
No. Proc	Wall time (second)	speedup	efficiency	Wall time (second)	Speedup	efficiency	Wall time (second)	speedup	efficiency
1	35467.89	1	100.00%	65782.23	1	100.00%	79856.34	1	100.00%
2	18282.42	1.940	97.00%	34440.99	1.902	95.1%	43471.06	1.837	91.85%
4	9427.93	3.762	94.05%	18666.92	3.524	88.1%	22907.73	3.486	87.15%
8	5510.86	6.436	80.45%	10761.04	6.113	76.41%	13667.01	5.843	73.04%

Table 4. The results on GAMESS with Same Molecule on Cluster

From table 3 and table 4, we can see that GAMESS gets better speedup than Gaussian. But the speedup drop dramatically when the number of processors goes up to 8 from 4.

5. Discussion

The team has recently reconfigured the cluster to be more maintainable. The new configuration employs a kick-start program. When a slave node must be added, replaced, or reconfigured, rather than reinstalling and reconfiguring Linux by hand, the kick-start floppy is loaded onto the cluster. The floppy directs the slave node to download the OS and configuration from the master node, which it does in a matter of four minutes. This provides a great time savings, and allows new configurations to be done on the master node only, and semi-automatically propagated to the slave nodes (via the kick-start diskette.) The team is currently investigating options to provide a flexible accounting system for the cluster. Such a system would allow easy and automatic statistical reporting of resource usage by educational institution, once the cluster is made available for general research use. These statistics are important in the center's annual accountability to its funding source, the Mississippi Legislature. Another challenge is how to make good and general use of the largely untapped 20 GB of non-OS disk space available on each slave node. The team is currently looking at ways to combine several of these slave disks into a parallel virtual file system (PVFS) for use a /ptmp directory.

The preliminary results of the study help us understand the behavior of the Linux Beowulf cluster system. We can apply performance tuning based on these results. After we applied several tuning techniques to the system, the Mop/s rate increased 10 times. The speedup value and efficiency doubled.

Most Gaussian 98 job require a lot of memory and disk spaces, we can improve the speedup of the high-level methods or larger jobs on the Linux Beowulf Cluster by increasing the resource on the node. The development of parallel version of Gaussian 98 can also help improve the speedup and performance on both systems.

This study provides very valuable information for the future development on the Beowulf Cluster. It will help us on the future upgrade or considering new application. It also provides reference to the MCSR user community. Based on the feedback from our user community, we are planning to upgrade the cluster to 54 nodes with limited budget.

Currently, we are working on a project to investigate how genetic algorithms work on the cluster and how it help Gaussian job to find global optimum faster.

Reference

1. Donald J. Becker *et al.* BEOWULF: A Parallel Workstation for Scientific Computation, *Proc. Int. Conf. on Parallel Processing*, Aug 1995, Vol 1, 11-14.

2. Frisch, M.J.: Gaussian98 User Reference, January 1998.

3. Gordon Research Group, Iowa State University, GAMESS http://www.msg.ameslab.gov/GAMESS/GAMESS.html.

4. T.L. Windus, M.W. Schmidt, and M.S. Gordon ; Parallel Processing With the Ab Initio Program GAMESS, in *Toward Teraflop Computing and New Grand Challenge Applications*, R.J. Kalia and P. Vashishta, Eds., Nova Science Publishers (New York), 1995.