REPORT 2

4/23/2003

Name	unity id
Anubhav Dhoot	avdhoot@unity.ncsu.edu
Kunal Shah	kdshah@unity.ncsu.edu

Note that all of the work was done by both of us together.

Table of contents

Installation of ASCI Benchmark SMG200

Compilation of SMG2000 source files

For each of the MPI-only, OpenMP-only and MPI-OpenMP hybrid approaches, the Makefile.include was updated accordingly by changing the 'CFLAGS' and then the source was compiled using a 'make veryclean' command (which removes the .o object files, libraries, and executables) followed by a simple 'make' command in the smg2000 directory.

Installation of DPCL on intel/Linux

We undertook the following steps to install the Benchmark on the Lab machines.

Installation of libelf and libdwarf

We installed the libelf (libelf-0.8.2-2) and libdwarf (libdwarf_shlib-1.1.0-1) libraries required by the dpcl (dpcl-3.3.2-1) rpm.

Installation of dpcl

After that we were able to install the dpcl rpm. (Note: We could not install this on any of the cluster machines due to compiler version problem.)

Changes required to execute dpcl programs

To let the dpcld daemon to be executed the xinetd.conf file was required to be modified. However, we created a file 'xinetd' in xinetd.d directory and set the appropriate directives as required by dpcl.

```
service dpclSD
{
        socket_type             = stream
        protocol                = tcp
        wait                    = no
        user                    = root
        server                  = /opt/dpcl/bin/dpclSD
        server_args             = /opt/dpcl/bin/dpcld /tmp/dpclSD01 /tmp/dpclsd
        disable                 = no
}
```

We added an entry for the port number used by dpcl to the /etc/services file. (This was actually updated automatically when the dpcl rpm was installed).

Finally we changed the .rhosts.file in our own home directory to include the 'localhost' as an allowed host as we got a rhosts_check_error on execution. One of the things that we noticed was that the anaylsis tool can't be run when logged in as root as this may compromise the security.

Execution of the Benchmark

(using OpenMP only, MPI only and MPI-OpenMP hybrid approaches)

Changing of runtime parameters

The problem size for smg2000, which is a 3-D grid solver, is given by the <Px>*<nx> by <Py>*<ny> by <Pz>*<nz>, where Px, Py, Pz forms the processor topology given & the -n option allows one to specify the local problem size per processor, the -P option during runtime as -n <nx> <ny> <nz> –P <Px> <Py> <Pz>,
e.g. " ./smg2000 -n 35 70 35 -c 0.1 1.0 10.0 -P 1 1 2
means a problem size of 35x70x70.

(Note: The -c option which specifies the diffusion coefficients were kept to the above values throughout).

The output wall clock time and cpu time were used to compare

OpenMP-only

The OpenMP only version was executed for a problem size of 35x35x35 by using runtime parameters as
"./smg2000 -n 35 35 35 -c 0.1 1.0 10.0 -P 1 1 1"
with varying number of threads (1 2 and 4 threads) by setting the OMP_NUM_THREADS environment variable. With 8 threads the execution did not finish for a long time and got killed off.

MPI-only

Various executions were done using the following format
"mpirun -machinefile ~/.rhosts -np 8 ./smg2000 -n 35 35 35 -c 0.1 1.0 10.0 -P 2 2 2"

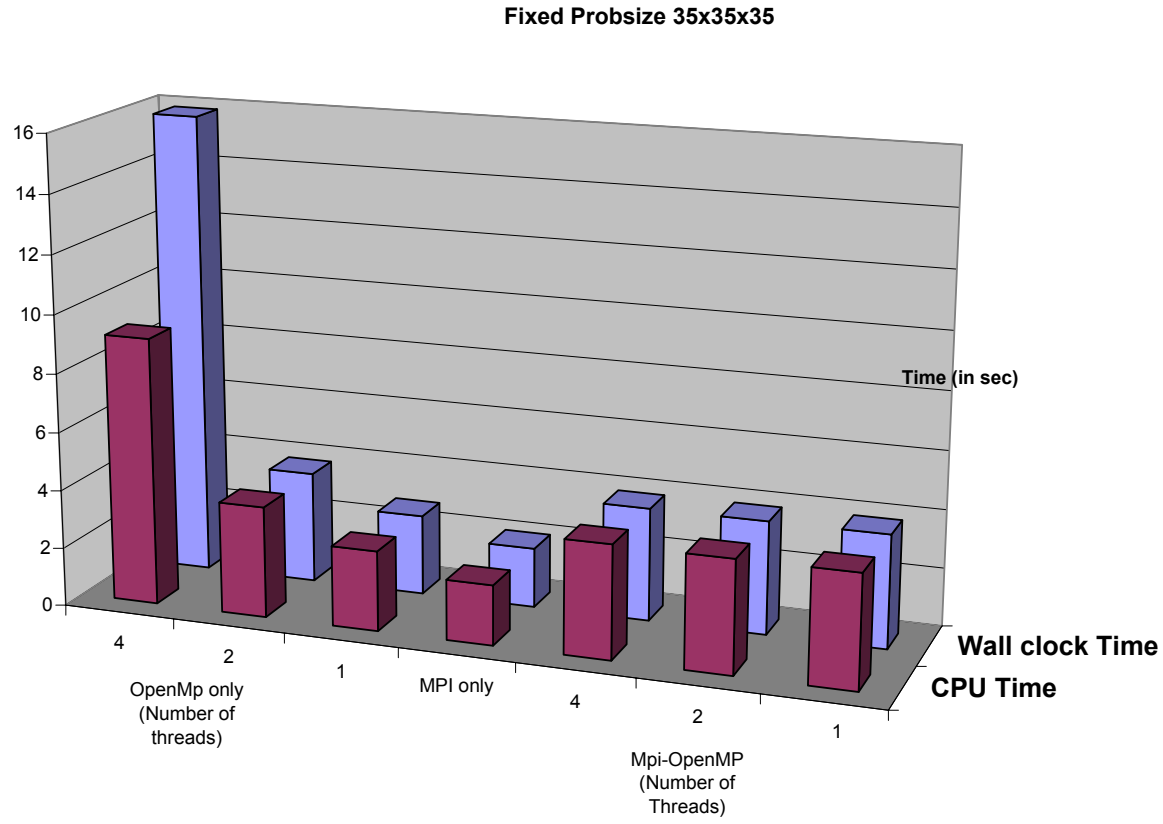
The problem size was kept at 35x35x35 to compare with OpenMP only version.
And then keeping the problem size fixed at 70x70x70 for different number of nodes 1 ,2 and 4 using processor topology -P 1 1 1, -P 1 1 2 and -P 1 2 2 and problem size on each node being 70 or 35 if the number of processor were 1 or 2 respectively for that dimension.

MPI-OpenMP hybrid

The same set of parameters used for MPI only is used for OpenMP-MPI hybrid version as well as for a problem set of 35x35x35 for comparison for all three as well as for different number of threads for comparison with OpenMP only version.

Results of Benchmark

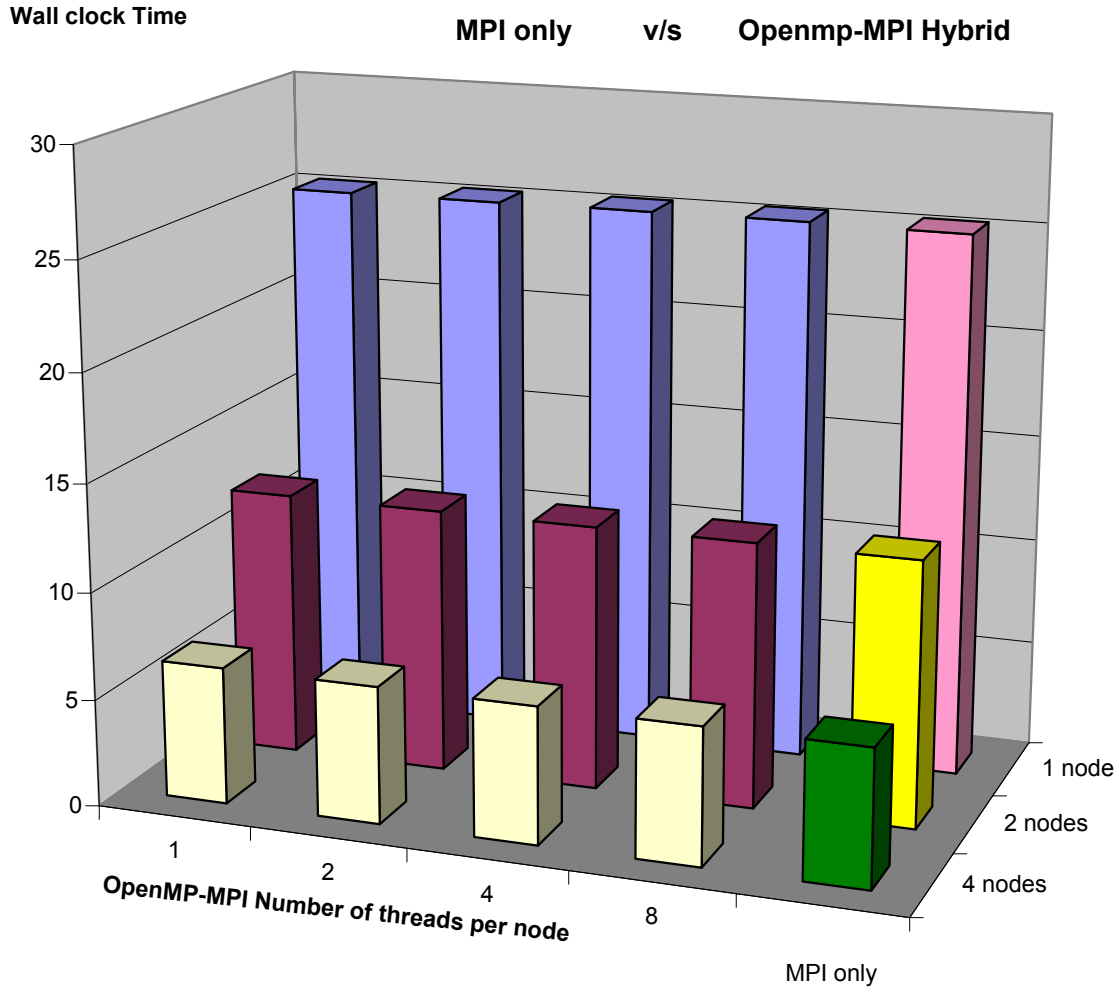
Comparison of OpenMP only, MPI only and OpenMP-MPI hybrid version



In comparison of all three versions, the MPI only gave the shortest wall clock time and CPU time. The OpenMP only version gave very poor performance for more number of threads but gave as good performance at one thread. The OpenMP-MPI hybrid version gave almost uniform results throughout.

We saw that changing number of threads has no considerable difference in the MPI-OpenMP hybrid versions as seen in the next graph.

Effect of different number of threads for OpenMP-MPI hybrid version



In the above test the problem size was kept as 70x70x70 by changing the processor topology and the problem size per processor. It was found as expected that distributing the same problem to 4 number of nodes showed the least time while for only 1 node it had the maximum time.

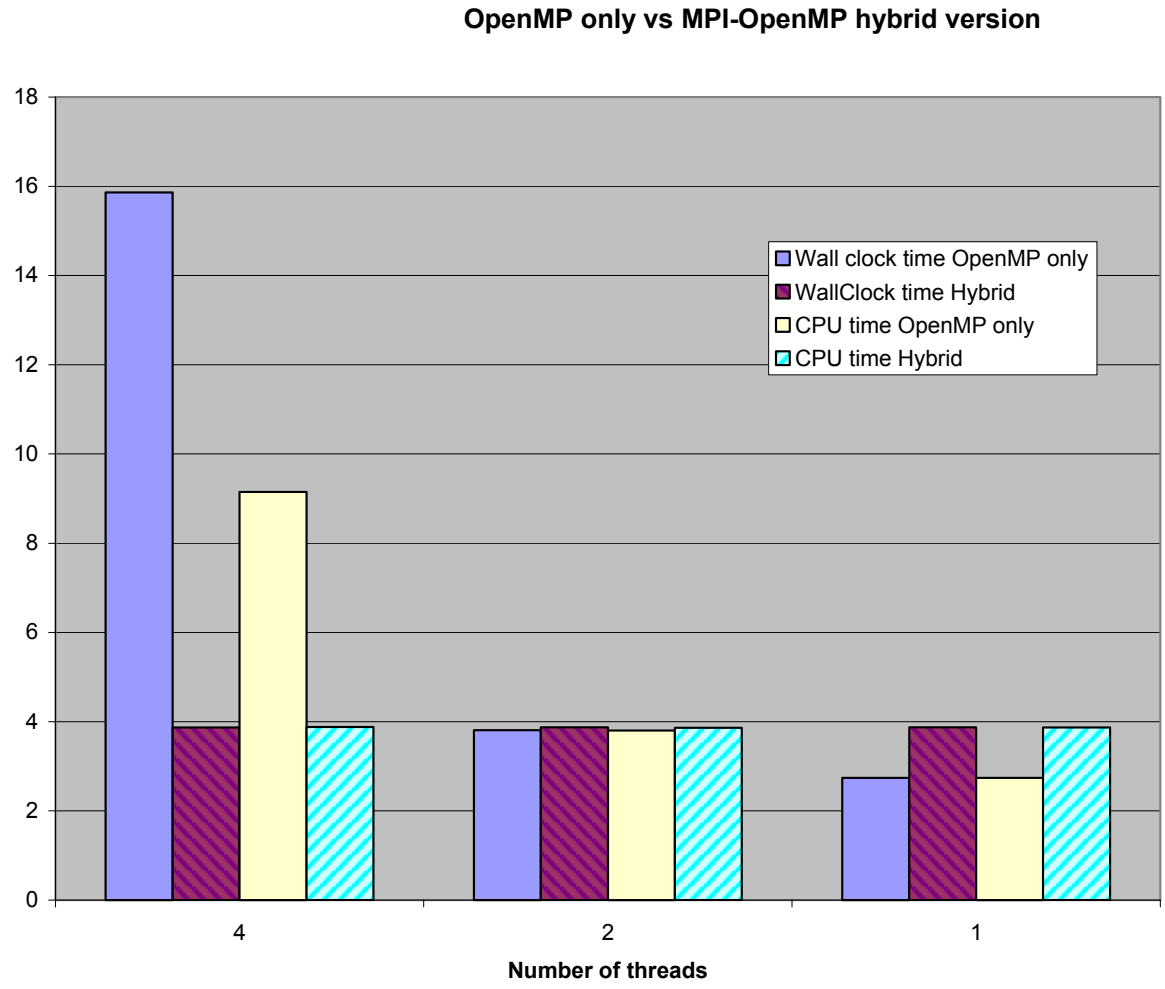
Also note the fact that OpenMP-MPI hybrid and MPI only version gave almost same Wall-clock times. The CPU clock times were also similar and can be seen in the Excel file which contains the data of the tests.

The achieved speedup by increasing the number of nodes was found in terms of Wall clock time to be 3.954715 and in terms of cpu time to be 3.952344 for the hybrid version.

Note that using 8 threads gave very poor performance and hence was not considered. This is obviously due to the architecture not being bale to sustain more than 4 threads

with 8 threads constantly affecting each other leading to no benefit in increasing the number of threads to eight.

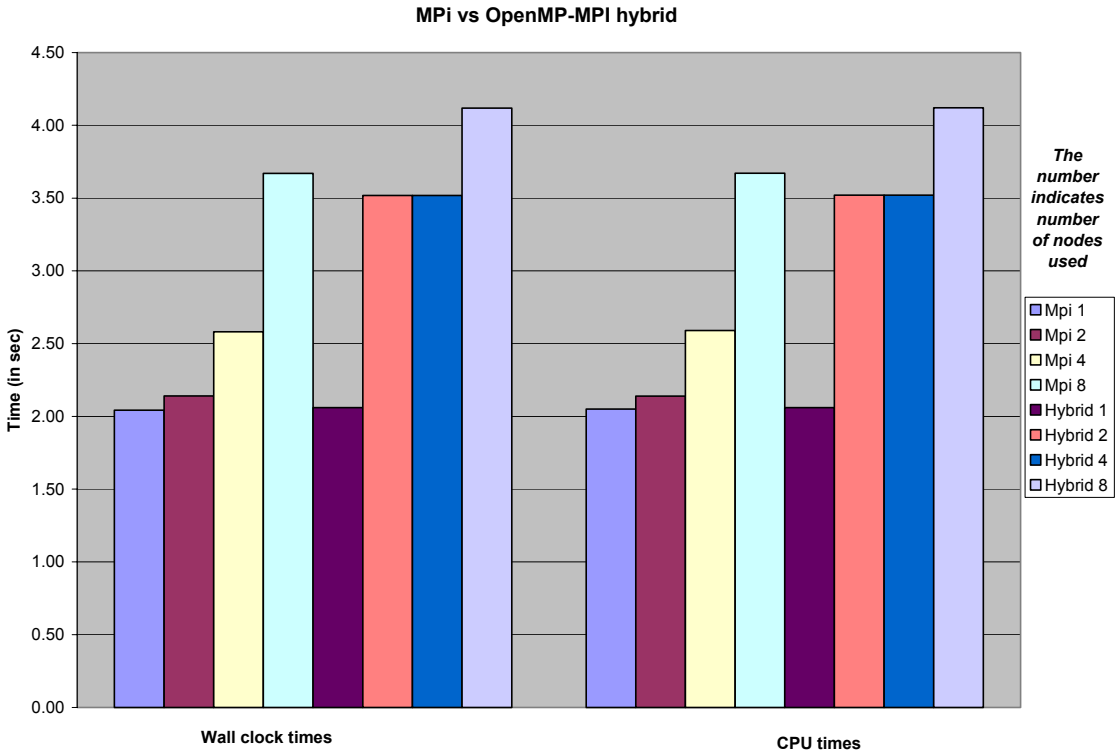
Comparison of OpenMP only versus OpenMP-MPI hybrid version



The test showed that the OpenMP only version gave clearly poor performance at higher number of threads while almost similar performance was reported for lower number of threads. The seemingly slight decrease in performance in the hybrid version at lower number of threads can be attributed to experimental error, as the times reported varied around 20 percent.

Comparison of MPI versus OpenMP-MPI hybrid version

A detailed test of MPI versus OpenMP-MPI hybrid was necessary as they exhibited almost similar behavior.



This shows that the better behavior is for MPI only.

As the OpenMP was found to be poorer that the hybrid versions for more number of threads , it seems that MPI only version is the best.