# Project Report

## Objective

This project aims at
a) Evaluating the performance of running SMG2000 benchmark code on the Myrinet Cluster.
b) Installing the Dynamic Probe Class Library (DPCL) tool, integrating it with the ASCI benchmark and evaluating the result, improving the application and re-evaluate the result.

a) About SMG2000 Benchmark

SMG2000 is a parallel semicoarsening multigrid solver for the linear systems arising from finite difference, finite volume, or finite element discretizations of the diffusion equation

$$\nabla \cdot (D \nabla u) + \sigma u = f$$

on logically rectangular grids. The code solves both 2D and 3D problems with discretization stencils of up to 9-point in 2D and up to 27-point in 3D

Multigrid methods are a class of techniques for performing fast, iterative solves of linear systems in linear time and space. These methods improve upon the convergence rate of classic iterative methods by using a hierarchy of grids at different resolutions. Coarser levels of the grid hierarchy are effective for quickly eliminating low frequency components of solution error. As a result, the total computational work required by a multigrid method to achieve a prescribed level of accuracy is proportional to the grid size.

In the SMG2000 benchmark, the solver is based on [1] which focussed on the scalablility for a three dimensional semicoarsening multigrid solver on a distributed memory computer. In the best case the scaled efficiency E(N,P)= T(N,1)/T(PN,P) for N unknowns and P processors, should be equal to 1. This would mean one could double the size of the problem and the number of processors while keeping the solution time constant.The minimum requirements was that the scaled effeciency be bounded away from zero, i.e. a solver is scalable iff there exists En > 0 such that E(N,P ) > En for all P.

SMG2000 is written in ISO-C and is an SPMD code that uses MPI. Parallelism is achieved by data decomposition. The driver provided with SMG2000 achieves this decomposition by simply subdividing the grid into logical P x Q x R (in 3D) chunks of equal size.

The efficiency of the parallel code depends on

- Size of data chunks
- Speed of communications
- Speed of computations – as it's a highly synchronous code.
- Memory access speeds – only about 1-2 computations are performed per memory access

b) DPCL (Dynamic Probe Class Library)

DPCL is a C++ class library whose application programming interface (API) enables a program to dynamically insert instrumentation code patches, or "probes", into an executing program. The program that uses DPCL calls to insert probes is called the "analysis tool", while the program that accepts and runs the probes is called the "target application". The ability of DPCL to dynamically insert probes relieves the need to recompile the code, allows switching from one tool to other without restarting the application and a slew of other advantages that can be found in [4] and [5].

We plan to install this library, and write analysis tools for some test programs. Once this gets working, we plan to write an analysis tool for evaluating the SMG2000 benchmark. This will require us to identify the various probe points (for point probes), as well as the appropriate phases for phase probes, if any. Since DPCL dynamically instruments an application, we will not be actually modifying the benchmark code, but inserting patches into the executable code either at run time or just before execution starts (which means that the target application will be invoked from the analysis tool).

## Project MileStones

- Milestone #1 → 04/15/2003: Evaluation of ASCI benchmark on our cluster
- Milestone #2 → 04/20/2003: Complete installation of DPCL and write analysis tools using DPCL for some test programs
- Milestone #3 → 04/25/2003: Finish writing analysis tool for SMG2000 benchmark
- Milestone #4 → 04/30/2003: Evaluating the application of possible improvements on the performance of the problem using the analysis tool
- Milestone #5 → 05/02/2003: Suggesting possible improvements in the tool and prototyping any one of them (if time permits)

## References

[1] Semicoarsening multigrid on distributed memory machines

[2] Semicoarsening multigrid on distributed memory machines [Peter N. Brown, Robert D. Falgout, Jim E. Jones]

[3] Experiences Tuning SMG98 — a Semicoarsening Multigrid Benchmark based on the hypre Library [Guohua Jin, John Mellor-Crummey]

[4] http://oss.software.ibm.com/developerworks/opensource/dpcl/

[5] http://www.ptools.org/projects/dpcl/

*This report is available at *http://www4.ncsu.edu/~kdshah/report.htm*