Incompressible Navier-Stokes with Particles Baseline Performance Measurement

P. Colella
D. F. Martin
N. D. Keen

Applied Numerical Algorithms Group
NERSC Division
Lawrence Berkeley National Laboratory
Berkeley, CA

June 25, 2004

Benchmark Problem Description

The algorithm is described in a separate document entitled "Incompressible Navier-Stokes with Particles Algorithm Design Document" [1].

To evaluate the performance of the incompressible Navier-Stokes with particles code, we use a background flow of a single vortex ring in three dimensions in a $1m^3$ box. For this problem, the vorticity distribution is specified, and the initial velocity is then computed based on the initial vorticity field. The vortex ring is specified by a location of the center of the vortex ring (x_0, y_0, z_0), the radius of the center of the local cross-section of the ring from the center of the vortex ring r, and the strength of the vortex ring Γ.

The cross-sectional vorticity distribution in the vortex ring is given by

$$\omega(\rho) = \frac{\Gamma}{a\sigma^2} e^{(\frac{\rho}{\sigma})^3} \tag{1}$$

where ρ is the local distance from the center of the ring cross-section, $a = 2268.85$, and $\sigma = 2.75$. For this problem, the vortex ring is centered at $(50, 50, 40)$, with a radius of 2, and strength Γ of 1.5×10^5

Particles are then added with stationary initial conditions on the $z = 50cm$ plane, distributed evenly in $25 < x, y < 75$. The particle properties are designed to simulate a particle with twice the density of the ambient fluid (water), a diameter of 1mm, and which follow the laminar drag rule $C_D = \frac{24}{Re}$. [2]

This document presents serial profiling and parallel performance results for a given set of inputs to the incompressible Navier-Stokes code. The inputs file for the $32 \times 32 \times 32$ 900-particle benchmark run is shown in Figure 1.

Target Platform and Compilers

The target platform for this benchmark measurement is a machine named Seaborg located at NERSC. Seaborg is an IBM SP RS/6000 System and it currently consists of 6,080 processors. Each processor is a POWER3 chip with a clock speed of 375 MHz and peak performance of 1.5 Gflops. The Seaborg processors are clustered into 380 symmetric multiprocessor nodes (16 processors per node). Seaborg is a hybrid system in the sense that memory is distributed among nodes, but within a node memory is shared.

The Fortran compiler used for this was the AIX Fortran compiler `xlf` version 8.1.1.5 with the flags set to be `-O3 -qarch=auto -qmaxmem=99999 -qhot -qstrict`. The C++ compiler used was the AIX C++ compiler `xlC` version 6.0.0.7 with flags as `-O3 -qstrict -qarch=auto -qstaticinline`.

Profiling Methodology

The primary metric used in this performance analysis work is wall-clock time of various units and sections of the code. The standard C function `gettimeofday()` is used to

```
#inputs file for 32x32x32 900-particle test case, 4 timesteps
main.max_step = 4    #max number of timesteps to computAe
main.max_time = 5.0  #stop time
main.num_cells = 32 32 32  #base level domain
main.max_level = 0
main.ref_ratio = 4
main.regrid_interval = 100000 10000
main.block_factor = 8
main.max_grid_size = 32
main.checkpoint_interval = -1
main.plot_interval = -1
main.cfl = 0.5
main.particleCFL = 0.1
main.fixed_dt = 0.012
main.is_periodic = 0 0 0

#Note that we're in CGS units here, so 1m = 100 cm
ns.domainLength = 100.0 100.0 100.0
ns.init_shrink = 0.1
ns.project_initial_vel = 1
ns.init_pressures = 1
ns.num_init_passes = 1

ns.specifyInitialGrids = 0
ns.initVelFromVorticity = 1
ns.backgroundVelocity = 0.0
ns.viscosity = 0.0040
ns.num_scalars = 0
ns.scal_diffusion_coeffs = 0.00 0.0

#particle stuff
ns.particle_epsilon = 6.25
ns.particle_drag_coeff = 0.04
ns.particle_body_force = 0 0 0
#if 1, use image particles to help enforce BC's
ns.use_image_particles = 1
# input initial particle positions from a file
ns.read_particles_from_file = 1
ns.particle_file = particles.900p.dat

# this is physical BC info
# 0 = solidWall, 1=inflow, 2=outflow, 3=symmetry, 4=noShear
physBC.lo = 4 4 4
physBC.hi = 4 4 4
physBC.maxInflowVel = 1.0
```

Figure 1: Inputs file for $32 \times 32 \times 32$ 900-particle benchmark problem.

obtain the wall-clock time. This method is robust and has a resolution of approximately one microsecond on the target machine. The execution or run time is the measured wall-clock time to compute 4 timesteps and does not include setup overhead or I/O at the end of the run. We also report memory usage for the code obtained using the `getrusage` system call, which retrieves information about resources used by the current process such as the maximum resident set size.

Serial Performance

Table 1 shows serial performance for the particle code with 0, 1, 9, 100, 900, and 10,000 particles on a $128 \times 128 \times 128$ domain for 4 timesteps. These runs were performed using the parallel code, but with only one active processor. A true serial run would yield similar results, but without the small overhead in time and memory caused by MPI.

Domain Size	number of particles	memory usage (MB)	execution time (s)
128x128x128	0	1184	677
128x128x128	1	1192	1233
128x128x128	9	1192	1236
128x128x128	100	1192	1257
128x128x128	900	1193	1362
128x128x128	10000	1213	2658

Table 1: Serial performance of particle code for $128 \times 128 \times 128$ domain, 4 timesteps.

Parallel Performance

To measure the parallel performance of the code, the benchmark problem was run using 1, 2, 4, 8, 16, and 32 processors. Tables 2 and 3 show execution times and memory usage for the $128 \times 128 \times 128$ case with 0, 900, and 10,000 particles for varying numbers of processors.

It should be noted that in the current implementation of the code, the infinite-domain elliptic solver operates by copying all of the particles over to a single processor and then doing the infinite domain solve on that single processor for the entire domain. So, this part of the algorithm represents a bottleneck in parallel at the moment, both for computational time and for memory usage. We expect that fixing this implementation issue will result in better parallel performance.

Another way to evaluate the performance of the code is to run the same problem with different grid sizes. To this end, we run the 900 particle test case with $32^3, 64^3$, and 128^3

number of processors	0 particles	1 particle	900 particles	10000 particles
1	677	1233	1362	2658
2	338	925	1025	2297
4	177	742	841	2127
8	88	642	756	2040
16	50	611	729	2001

Table 2: Execution time in seconds of particle code for $128 \times 128 \times 128$ domain, 4 timesteps.

number of processors	0 p avg (min-max)	1 p avg (min-max)	900 p avg (min-max)	10000 p avg (min-max)
1	1184	1192	1193	1213
2	632 (632-633)	742 (639-845)	743 (640-846)	755 (654-857)
4	356 (346-366)	445 (349-686)	445 (350-687)	453 (352-696)
8	203 (198-208)	255 (201-591)	255 (200-592)	261 (202-613)
16	127 (121-133)	154 (123-543)	155 (123-544)	158 (124-557)

Table 3: Memory usage in MB for particle code for $128 \times 128 \times 128$ domain, 4 timesteps.

computational domains. The execution times and memory usages for this problem are shown in Tables 4 and 5.

number of processors	32^3	64^3	128^3
1	51	221	1362
4	51	155	1025
8	51	144	841
16	52	145	756

Table 4: Parallel execution time in seconds for particle code for 900 particle test case, 4 timesteps.

number of processors	32^3 avg (min-max)	64^3 avg (min-max)	128^3 avg (min-max)
1	59	186	1193
4	30 (21-59)	83 (81-89)	445 (350-687)
8	26 (21-59)	63 (61-77)	255 (200-592)
16	23 (21-59)	42 (21-77)	155 (123-544)

Table 5: Memory usage in MB of particle code for 900 particle test case, 4 timesteps.

Bibliography

[1] Dan Martin and Phil Colella. Incompressible Navier-Stokes with particles design document. Technical report, Applied Numerical Algorithms Group, Lawrence Berkeley Laboratory, 2003.

[2] Frank M. White. *Fluid Mechanics*. McGraw-Hill, second edition edition, 1986.