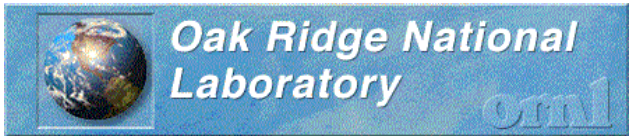

Parallelization of MFIX

Multiphase Fluid Dynamics Research Consortium Annual Meeting

Sep 26 - 28, 1999

Ed D'Azevedo

Oak Ridge National Laboratory

Overview

- Shared memory version using OpenMP directives. 6X speedup on 8 processors.
- Distributed memory version using MPI in development.
- Preliminary results with linear solver module.

Shared memory version

- MFIX converted from f77 to f90.
- Source code in CVS version control.
- Runtime profiles show over 70% time in linear solver. Remaining time evenly distributed among many routines.
- Linear solver options are SOR or CG/GMRES preconditioned with ILU from SLAP library.

SLAP library

- MFIX uses a logically rectangular (I,J,K) grid with regular stencil.
- SLAP uses a general “IA,JA” column-oriented sparse matrix format.
- Difficult to parallelize indirect summation in matrix multiply.
- Forward and backward solves in ILU have little opportunity for parallelism.
- Costly (time and memory) conversion from MFIX to SLAP format.

New linear solver

- SOR in canonical order has data dependencies that cannot be easily parallelized.
- Implement BiCGSTAB and GMRES iterative solvers.
- Preconditioned by line relaxation along longest “J” (vertical) axis. Lead to simple tridiagonal solves.
- Plane sweeps (I+J+K=const) not as effective due to long and slender nature of grid.

New linear solver

- Matrix multiply take advantage of (I,J,K) structure.
- BiCGSTAB uses 3-term recurrence and has low storage overhead but may not have monotone convergence.
- m-step GMRES with restart also available.
- Consider reordering arrays with J-fastest, then I,K for better memory performance and cache reuse.

OpenMP

- Simple “parallel do” directives on most time consuming loops.
- Matrix multiply and line relaxation easily fit within OpenMP framework.
- Other time-consuming loops parallelized based on runtime profile.
- Development and testing performed on SGI multiprocessor.
- Tests performed by Ravi Subramanya at Pittsburgh Supercomputing Center.

3D hydrodynamics

#processors	time	speedup	efficiency	cells/proc
1	13953	1.00	1.00	115200
2	7492	1.86	0.93	57600
4	4264	3.27	0.82	28800
8	2363	5.90	0.74	14400

BiCGSTAB with 20 iterations. Old scalar code timing = 12660s

BiCGSTAB 20 iterations, TSTOP=0.0005

processors	fluidBed2	fluidBed3
1	1188.7	2738.4
2	593.9	1432.6
4	337.4	868.3
8	228.0	559.1
10	210.5	471.5

Distributed memory MFIX

- ORNL deliver domain decomposition, communication and linear solver components.
- FETC + AEOLUS focus on I/O and overall integration. Expect initial parallel run in end of Oct.
- Two-dimensional domain decomposition in I,K axes, keep J on same processor for line relaxation.
- Performance tuning at later stage.

Performance of linear solver

- Tests on x86 linux cluster, each node is dual 266Mhz Pentium II with 256Meg ram and off-the-shelf 100Mbits ethernet.
- pgf90 with mpich on redhat linux 6.0
- Fixed 20 iterations of BiCGSTAB with preconditioning by simple diagonal scaling and line relaxation.

Legend

- *proc*: processor grid determines 1-D or 2-D decomposition.
- *matvec*: time for matrix vector multiply, includes communication time in halo region.
- *dot_prod*: vector dot_product, includes time in global sum and synchronization.
- *precond*: time spend in preconditioner
- *comm*: time for communication exchange in halo region.

30x30x30 diagonal scaling

proc	matvec	dot_prod	precond	comm	overall
1x1	1.4	0.6	0.4	0.0	3.1
1x4	0.6	0.3	0.1	0.3	1.3
2x2	0.6	0.2	0.1	0.2	1.0
1x6	0.7	0.2	0.1	0.3	1.4
2x3	0.7	0.3	0.1	0.3	1.4

60x60x60 diagonal scaling

proc	matvec	dot_prod	precond	comm	overall
1x1	13.7	4.3	2.9	0.0	26.6
1x4	6.5	1.3	0.7	3.3	10.2
2x2	3.7	1.3	0.7	0.9	7.4
1x6	4.5	1.4	0.8	1.8	8.7
2x3	4.5	1.4	0.8	1.2	8.8

30x30x30 line relaxation

proc	matvec	dot_prod	precond	comm	overall
1x1	1.4	0.5	5.2	0.0	8.0
1x4	0.6	0.2	2.7	1.5	3.7
2x2	0.6	0.3	2.5	1.5	3.6
1x6	0.7	0.3	3.0	2.2	4.3
2x3	0.7	0.3	2.6	1.8	3.9

60x60x60 line relaxation

proc	matvec	dot_prod	precond	comm	overall
1x1	13.4	4.4	43.7	0.0	67.2
1x4	5.1	1.2	25.0	15.6	33.1
2x2	3.8	1.2	13.3	4.8	20.0
1x6	4.5	1.2	21.0	12.8	28.7
2x3	5.1	1.3	13.6	7.3	22.0

Observations

- Low Mflops rate (5-10Mflops) in *serial* sparse matrix computation.
- High integer overhead in index calculations.
- High communication overhead.
- 2D seems to be better than 1D decomposition.
- Further tests on NERSC/T3E with faster communication fabric.