ParisSimulator Project

Project participants

- Prof. Gretar Tryggvason , University of Notre Dame
- Dr Sadegh Dabiri, , University of Notre Dame

- Yours Truly, ∂'Alembert

Potential project participants (have agreed but not started)

- Ruben Scardovelli, Univ of Bologna, Italy
- Phil Yecko, Montclair, NJ, USA

May participate

- Pierre Sagaut, , ∂'Alembert

ParisSimulator: Paralllel Robust Interface Simulator

Aims:

1) To have a relatively simple code on which to test ideas about VOF, parallelisation, Front Tracking, Lagrangian Particles. Use relatively simple algorithms and programming techniques.

2) To perform certain types of large scale computations relatively fast compared to Gerris thanks to a simpler mesh structure.

3) To serve as a backup to Gerris in case unsolvable bugs are encountered in certain applications.

Will be released as a free code under GPL.

80% based on FTC3D2011, a code by Gretar Tryggvason and Sadegh Dabiri, 20% on Surfer, by Yours Truly, Ruben Scardovelli, Jie Li and Gianluigi Zanetti.

Numerical Methodology

- Finite Volume/ Finite Difference on a Cartesian Grid with streched x, y, z coordinates. (stretched coordinates will not be available immediately for VOF)

- MAC grid.

- Time schemes : 1) first order Euler 2) second order Runge-Kutta for NS part

- Advection terms by ENO (next slide)

- explicit viscous terms.

- VOF - PLIC – CIAM for interface reconstruction and advection (in the first version we shall just use the relatively simple VOF methods in Surfer)

- Surface tension by elementary Continuous Surface Stress method as in Surfer (in the first version).

Numerical Methodology (continued)

- solids by blocking solid sites. Solid sites can be blocked at each step of time-marching and pressure correction (Poisson solve), so they are always and exactly zero.

- MPI with an adjustable number of ghost layers (two in the present implementation).

- Two boundary types: solid wall or periodic.

Input and output

- Input of parameters by FORTRAN namelist.

- No input format for for geometrical objects so far.

- Output format: vtk and visit (in FTC3D2011, tecplot is also available)

Code style, writing and management

- Code in Fortran90 but some calls to C functions (serious portability issues)

- All of FTC3D2011 (base version without front tracking) is contained in two f90 files !
Choice made to keep the number of files low.

- no configure, only a short Makefile. But f90 compilation involves a different dependency structure.

Makefile

```
#--------- paris-ParisSimulator main Makefile -------------------------

# babbage has several fortran compilers
OMPI_FC=gfortran44

FC = mpif90
CC = mpicc

# remove funny cflags from gerris in my environment
CFLAGS =

# select option for hypre
# default hypre installation without root privileges:
# HYPRE_DIR = $(HOME)/hypre-2.8.0b/src
# Macport installation in /opt
HYPRE_DIR = /opt/hypre
# babbage
# HYPRE_DIR = /share/apps/hypre

#---------------------No changes needed beyond this line ----------------------------------

HYPRE_LIBS =  -L$(HYPRE_DIR)/lib -lHYPRE
```

Makefile - continued

```
OBJ = paris.o utilc.o solids.o modules.o
SRC = $(wildcard *.h *.c *.f90)
INC = $(wildcard *.h)

install: $(OBJ)
	@echo compiler is FC = $(FC), mpi override is OMPI_FC = $(OMPI_FC)
	$(FC) -o paris $(FOPTS) $(OBJ) $(HYPRE_LIBS)
	mv paris ~/bin/paris

all: tags install

clean:
	@rm -fR *.o *.mod paris *.gz stats *~ track out* stats errftc tmp* *.tmp fort.* *.visit TAGS tags core.*
	@cd Speed_Measurement; make clean; cd ..
	@cd Poiseuille_Test; make clean; cd ..

test:  install
	@rm -fR out input
	@ln -s miniinput input
	mpirun -np 8 paris
```

Makefile - end

```
tags:  $(SRC)
# @SZ Create a tags file named TAGS for use by emacs
	@etags $(SRC)
# @SZ Create a tags file named tags for use by vi or textwrangler
# @SZ On MacOS tags and TAGS are identical !
# @SZ 	ctags paris.f90

paris.o:  paris.f90 solids.o modules.o
	$(FC) -c  $<

solids.o:  solids.f90 modules.o
	$(FC) -c  $<

%.o : %.f90
	$(FC) -c $<


.c.o:   $< $(INC)
	$(CC) -c $(CFLAGS)   $<
```

- *paris-stable* has now 1936 lines in four source files.

- darcs archive (available at http://www.ida.upmc.fr/~zaleski/darcs/paris-stable)

- no documentation

The ENO scheme of Shu et Osher (similar to the BCG scheme used in Gerris) : the velocities interpolated at the faces of control volumes (centered on i+1/2,j) are :

$$u_{i,j} = \begin{cases} u_{i-1/2,j} + s_i\left(\frac{h}{2}\right), & \text{if } \frac{1}{2}(u_{i,j} + u_{i+1,j}) > 0; \\ u_{i+1/2,j} - s_{i+1}\left(\frac{h}{2}\right), & \text{if } \frac{1}{2}(u_{i,j} + u_{i+1,j}) < 0. \end{cases}$$

where the slope s_i is the smallest, in absolute value of the two expressions:

$$s_i^+ = (u_{i+1/2,j} - u_{i-1/2,j})/h$$
$$s_i^- = (u_{i-1/2,j} - u_{i-3/2,j})/h.$$

The resulting interpolated velocities are factored with simple arithmetic mean interpolations to compute the advection term in conservative form (conservative for constant density only).

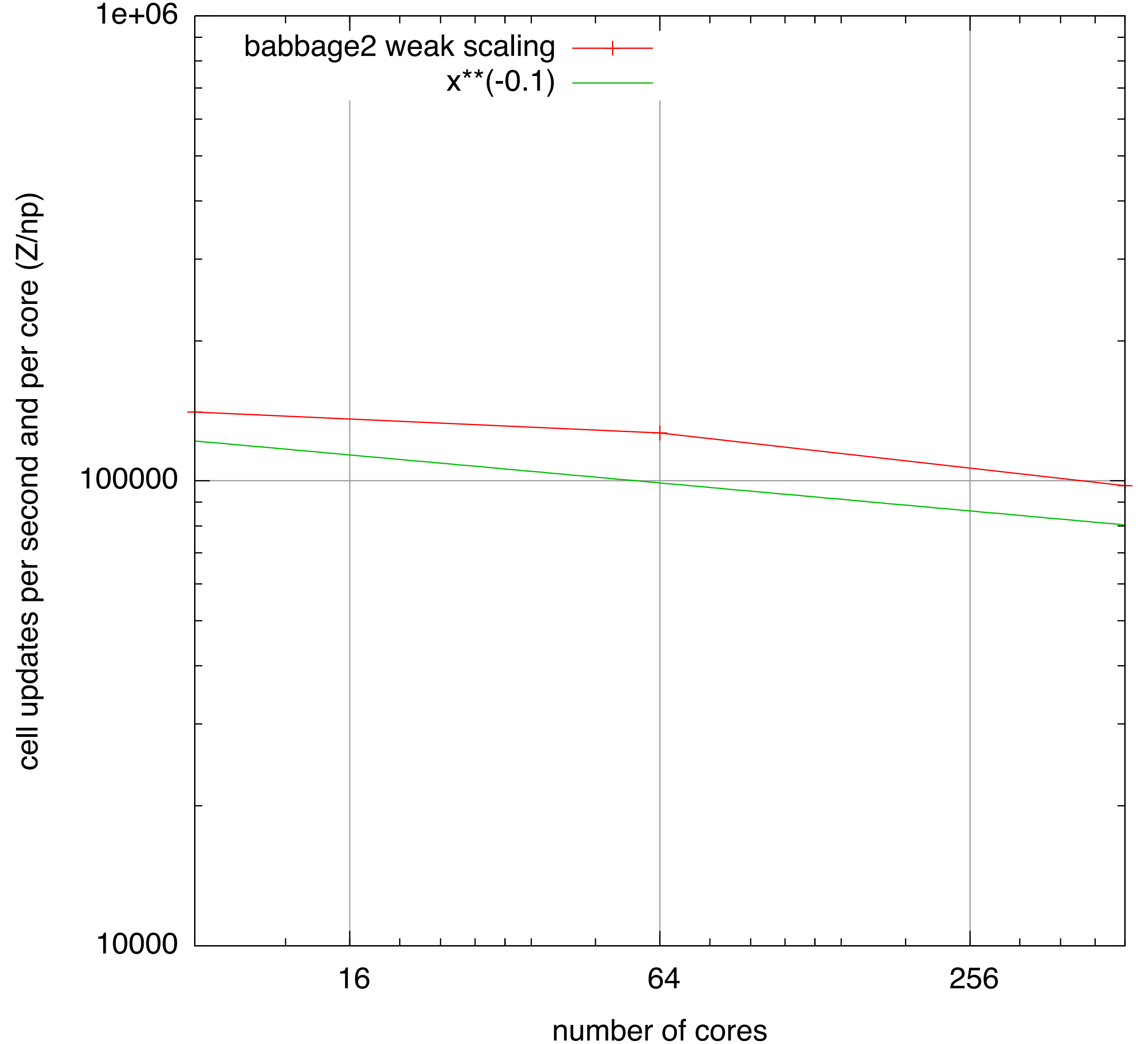
HYPRE Poisson solver: PFMG solver (a geometric solver usable on structured grids, unlike Boomer AMG used in Gerris and on unstructured grids.)

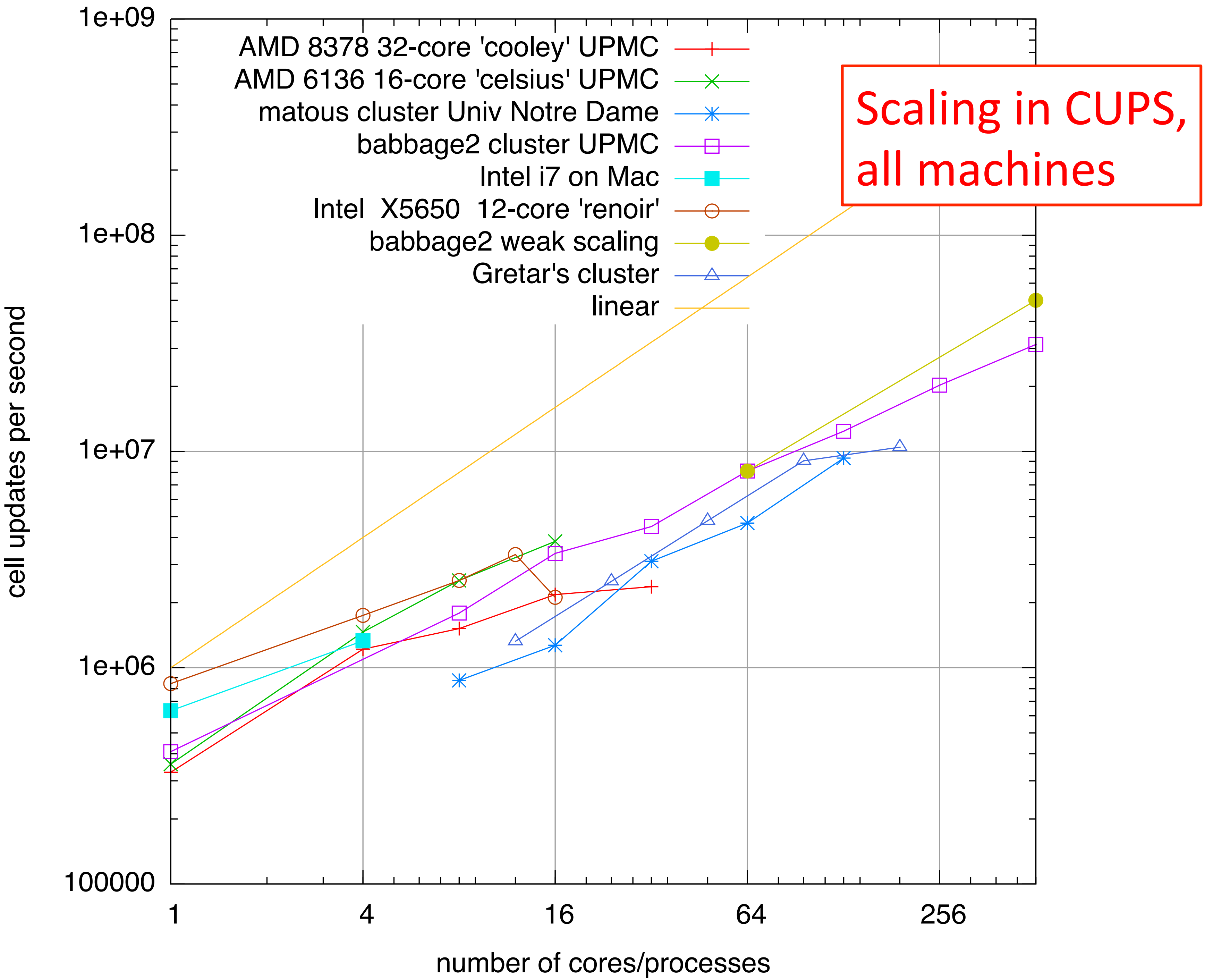
1. Ashby, S., Falgout, R.: A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations. Nuclear Science and Engineering **124** (1996) 145–159

10. Falgout, R., Jones, J.: Multigrid on massively parallel architectures. In Dick, E., Riemslagh, K., and Vierendeels, J., eds: Multigrid Methods VI, Lecture Notes in Computational Science and Engineering, vol. **14** (2000) 101–107, Berlin. Springer

21. Schaffer, S.: A semi-coarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients. SIAM J. Sci. Comput. **20** (1998) 228–242

Scaling tests: weak scaling in CUPS per core, babbage cluster (AMD, d'Alembert)





AMD 8378 32-core 'cooley' UPMC
AMD 6136 16-core 'celsius' UPMC
matous cluster Univ Notre Dame
babbage2 cluster UPMC
Intel i7 on Mac
Intel X5650 12-core 'renoir'
babbage2 weak scaling
Gretar's cluster
linear
Scaling in CUPS,
all machines
1e+09
1e+08
1e+07
1e+06
100000
1
4
16
64
256
cell updates per second
number of cores/processes

Next goal: perform a large scale atomization simulation by end of year