A Detailed Timing and Profiling Model of NBODY6++ using MPI and GPU Parallelization

SIYI HUANG

SILK ROAD PROJECT, NAOC

DEC. 12, 2013

NBODY6++ FEATURES

4^{th}-order Hermite integration

Hierarchically blocked time-steps

AC(Ahmad&Cohen) neighbor scheme

Local part:
nearby particles,
smaller time scale;
Global part:
further particles,
longer time scale;

Parallel implementation

GPU acceleration

CODE SUBDIVISION

$$\boldsymbol{TOT = REG + IRR + PRED + COMM + HOST}$$

- $\boldsymbol{REG}$: Regular force computation (on GPU)
- $\boldsymbol{IRR}$: Irregular force computation
- $\boldsymbol{PRED}$: Particle prediction

} Parallelized

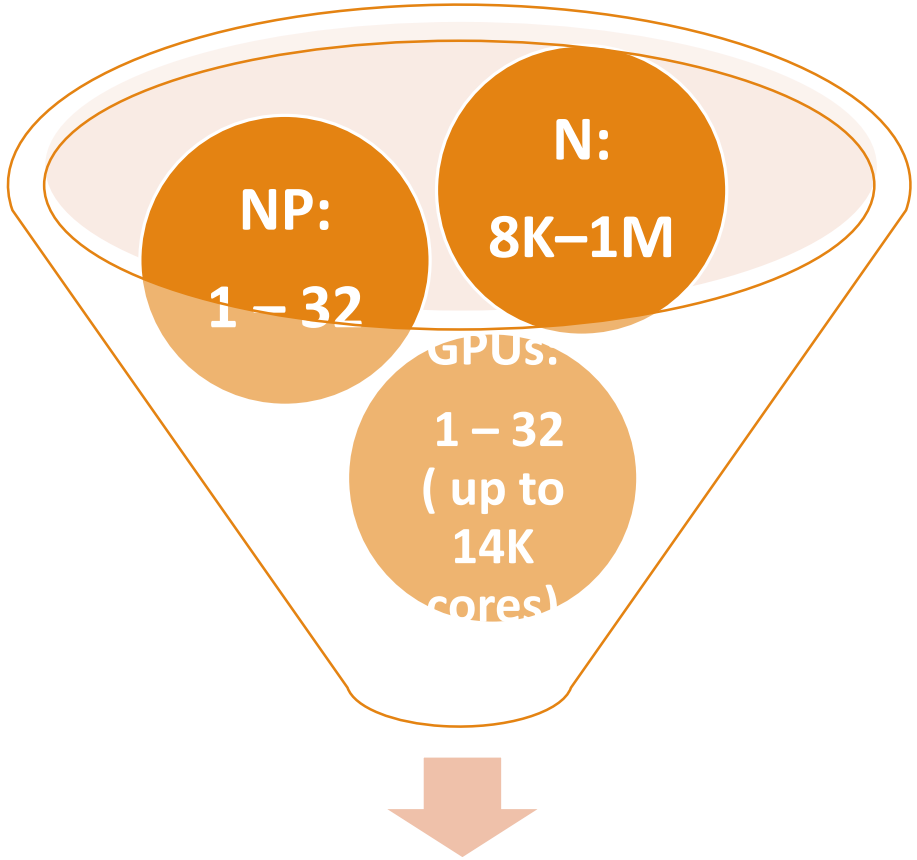
- $\boldsymbol{COMM}$: Communication (data moving, interprocessors communication, synchronization, etc.)
- $\boldsymbol{HOST}$: Serial execution part (branch decision, active particle determination, etc.)

$$\sim \boldsymbol{O(N^x)};\ \ \boldsymbol{O(N_{PE}^x)};$$

PARAMETERS

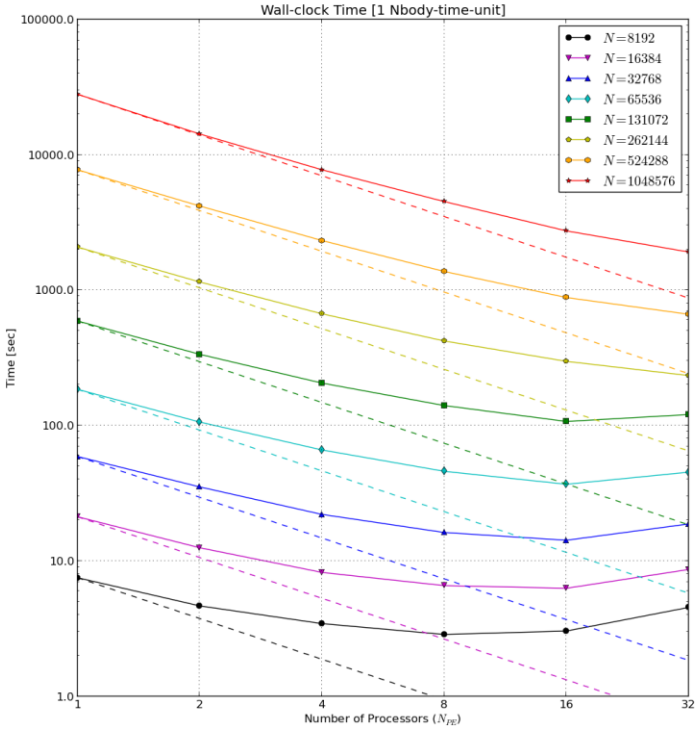
Cluster name	MILKYWAY
CPU	Intel Xeon X5650 6-core processor 2.66 GHz
GPU	NVIDIA Tesla M2070 (Fermi) 1.15 GHz (448 cores) 6GB memory
Memory	96 GB
OS	SUSE SLES 11

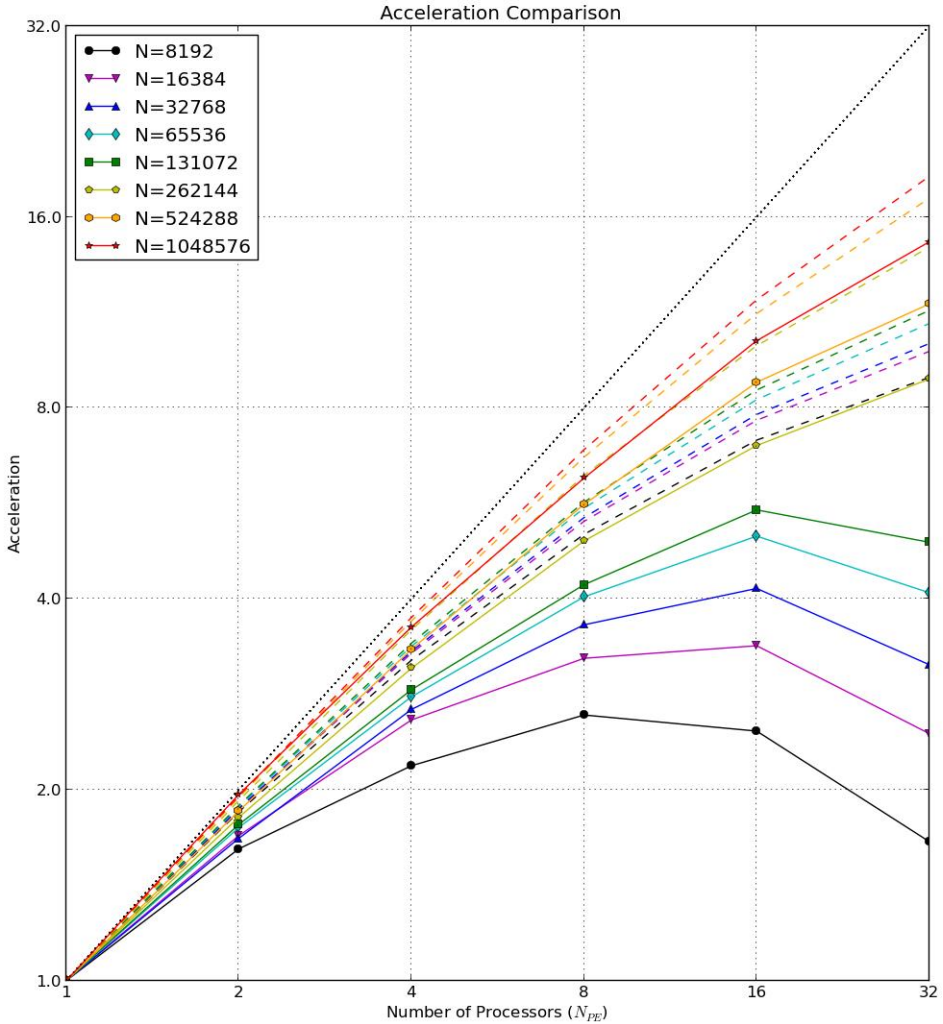
Hardware environment



Initial condition

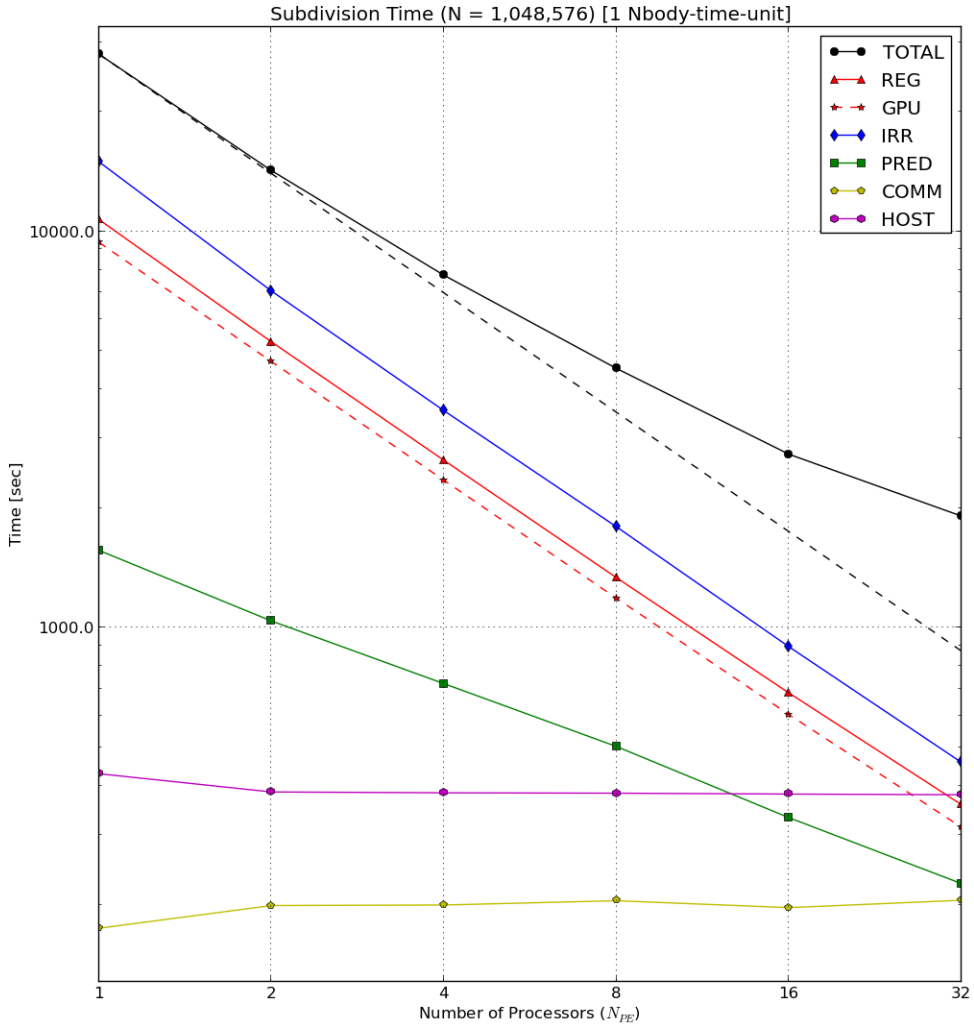
RESULTS OVERVIEW





SUBDIVISIONS

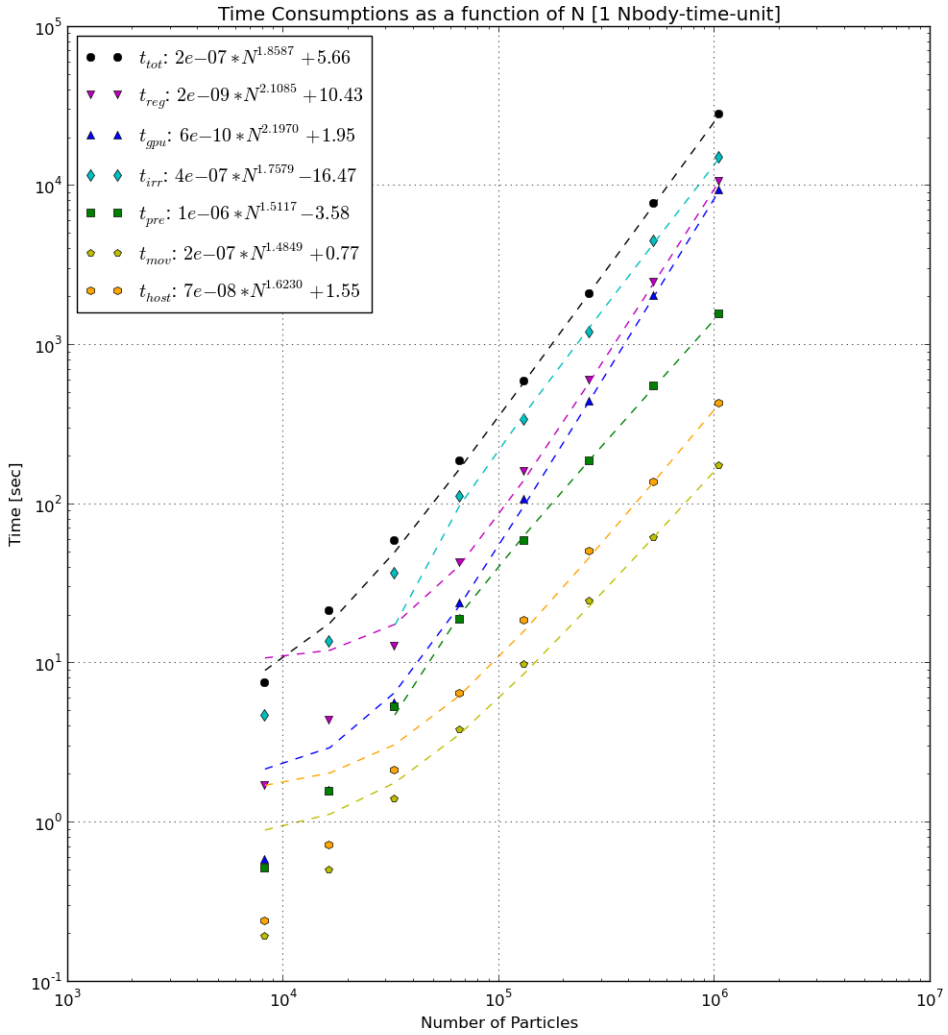
T O T
R E G
I R R
PRED
HOST
COMM



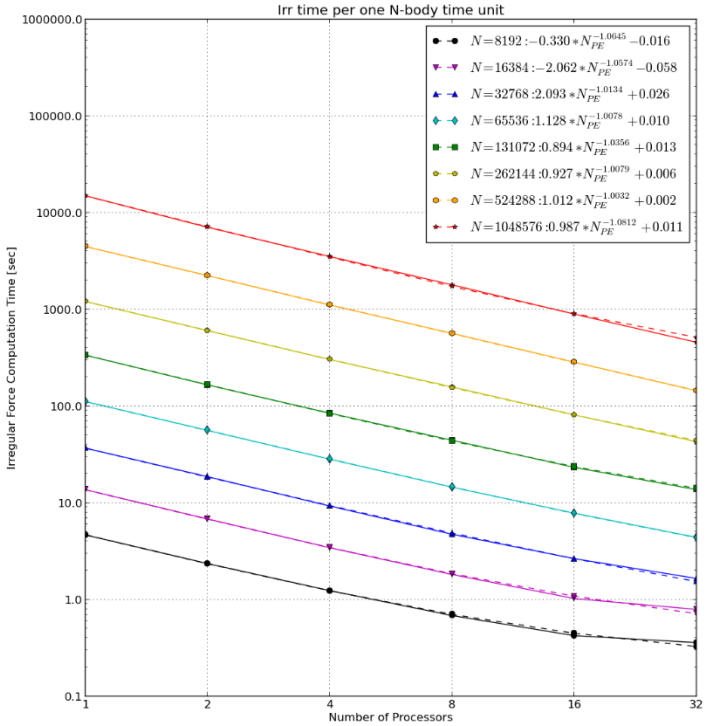
FUNCTION OF N

Main Parts:

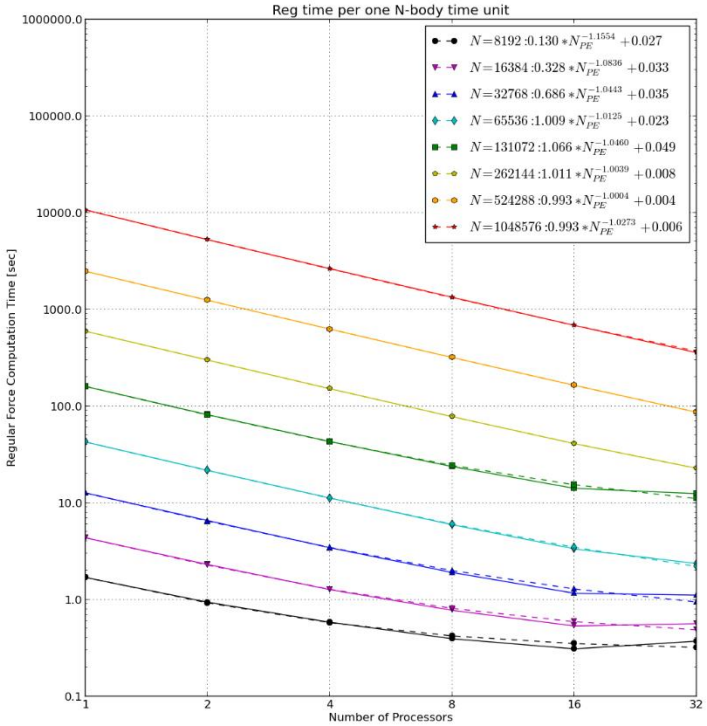
- $t_{irr} \sim O(N^{1.76})$
- $t_{reg} \sim O(N^{2.11})$



REG & IRR PARTS

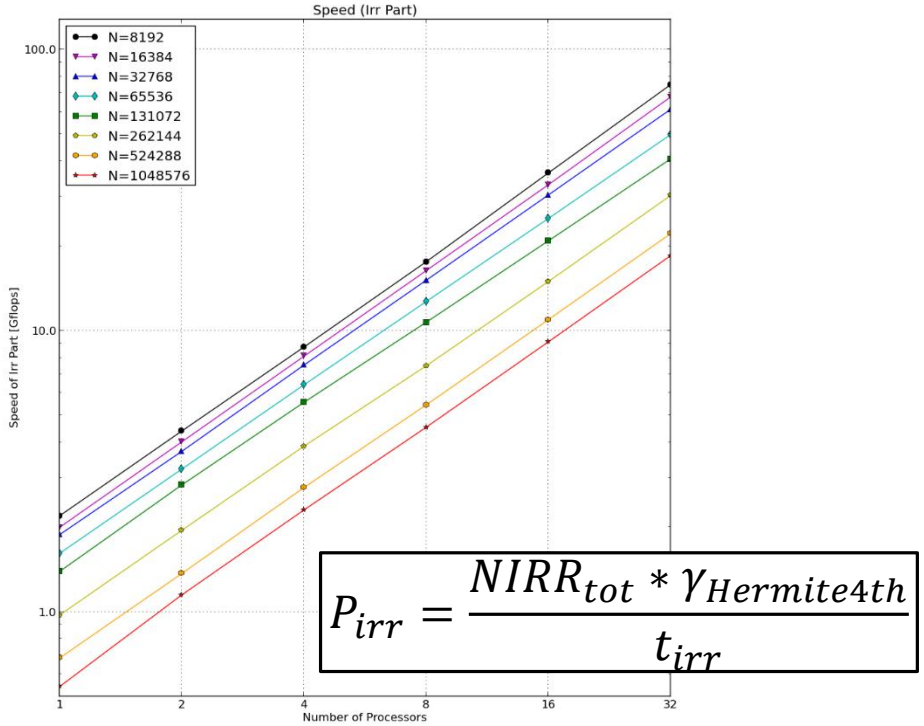


$$t_{irr} \sim O\left(N_{PE}{}^{-1.0}\right)$$

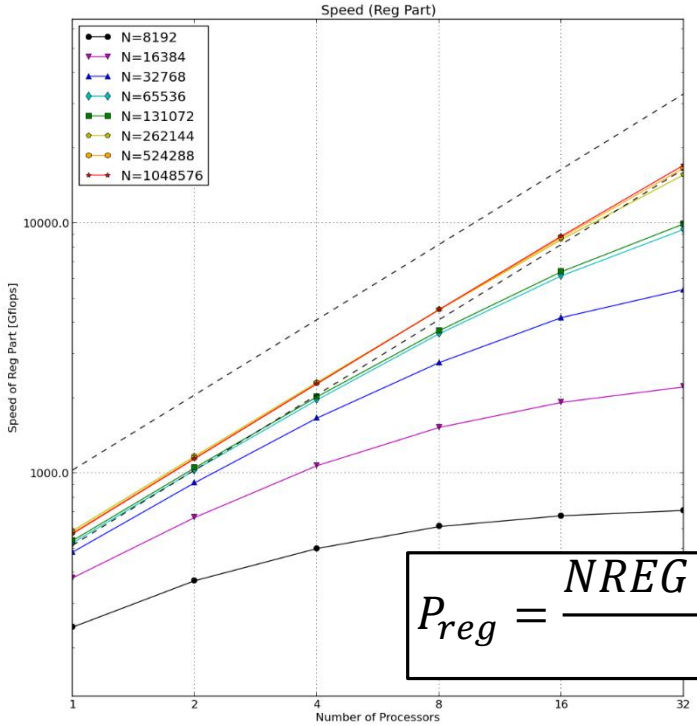


$$t_{reg} \sim O\left(N_{PE}{}^{-1.0}\right)$$

REG & IRR PARTS



$$P_{irr} = \frac{NIRR_{tot} * \gamma_{Hermite4th}}{t_{irr}}$$



$$P_{reg} = \frac{NREG * N * \gamma_{Hermite4th}}{t_{reg}}$$

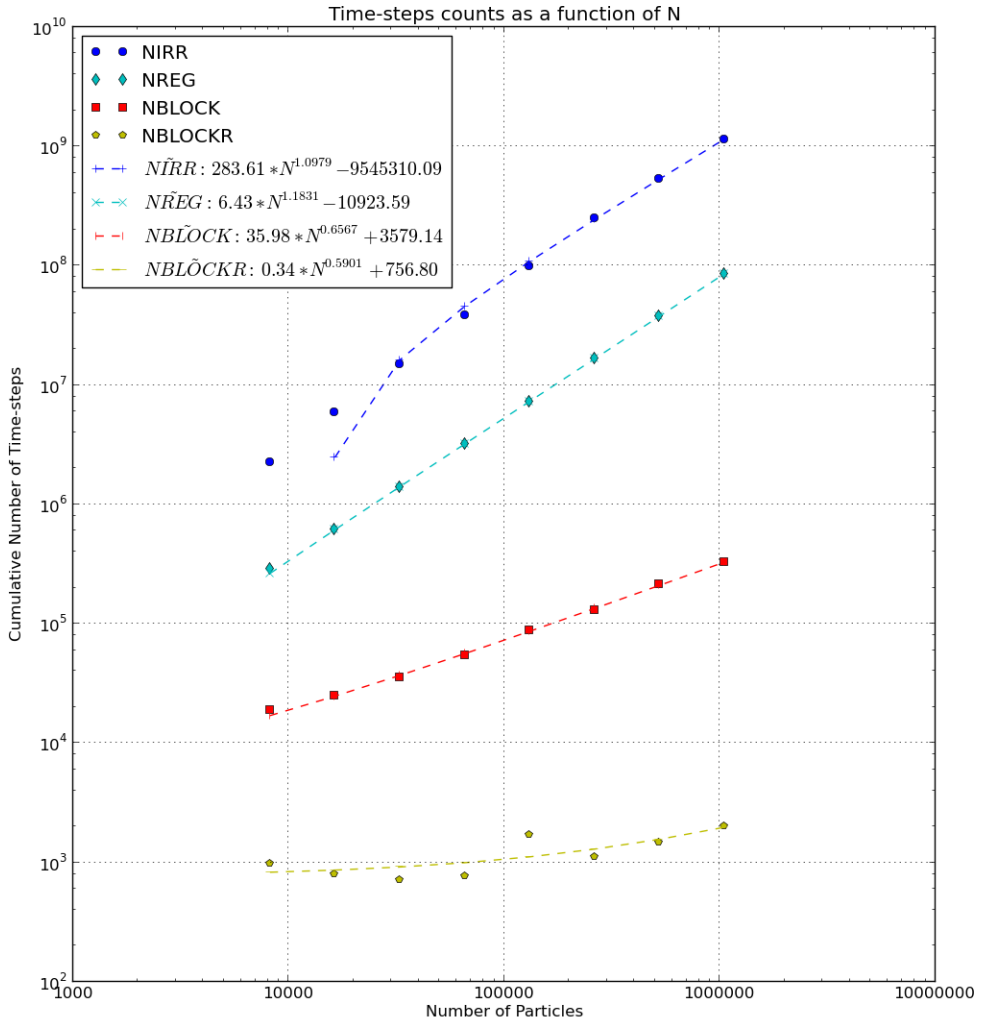
REG & IRR PARTS

$$P_{irr} = \frac{NIRR_{tot} * \gamma_{Hermite4th}}{t_{irr}}$$

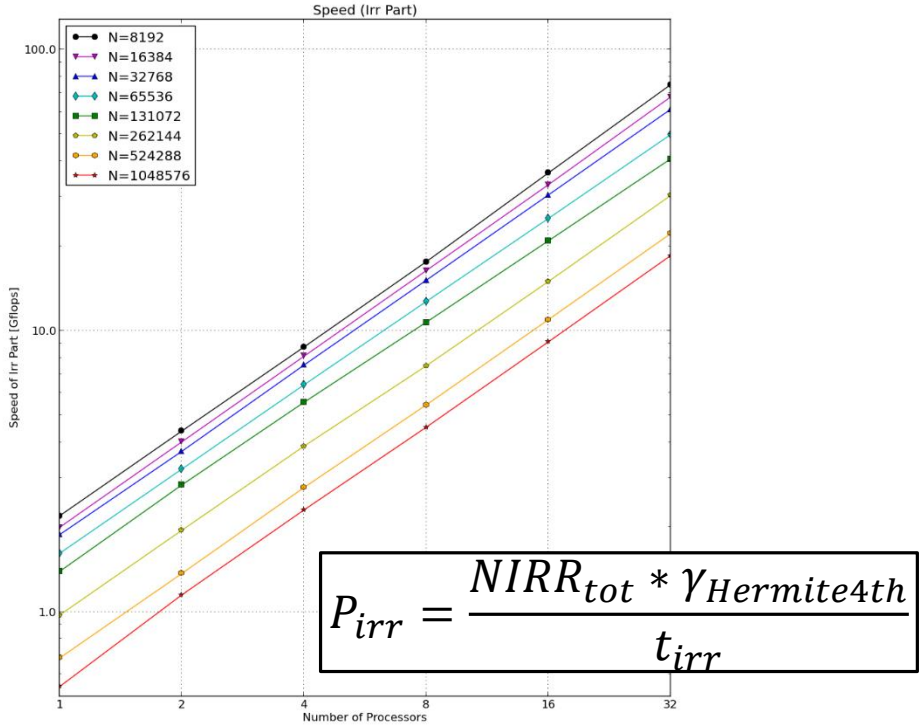
$$P_{reg} = \frac{NREG * N * \gamma_{Hermite4th}}{t_{reg}}$$

$$t_{irr} \sim O\left(N^{1.76} * {N_{PE}}^{-1.0}\right),\ NIRR \sim O(N^{1.10})$$

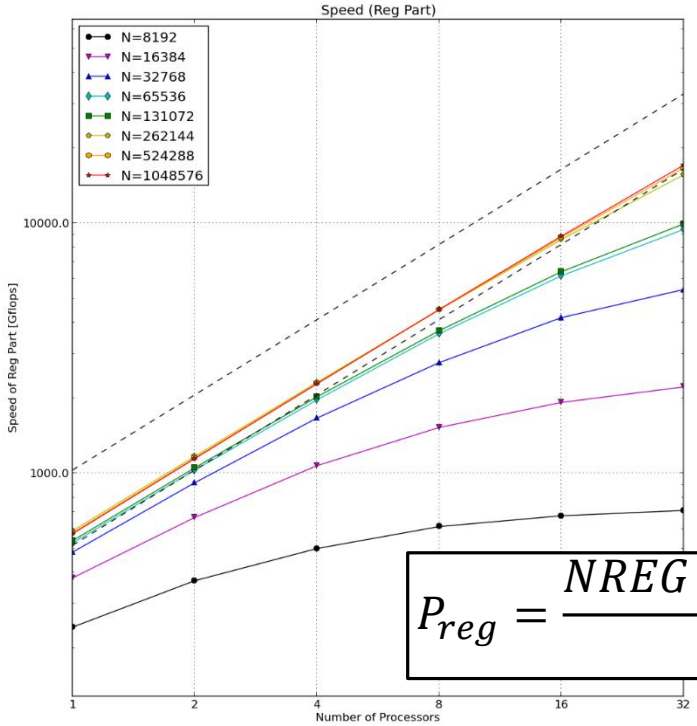
$$t_{reg} \sim O\left(N^{2.11} * {N_{PE}}^{-1.0}\right),\ NREG \sim O(N^{1.18})$$



REG & IRR PARTS



$$P_{irr} = \frac{NIRR_{tot} * \gamma_{Hermite4th}}{t_{irr}}$$



$$P_{reg} = \frac{NREG * N * \gamma_{Hermite4th}}{t_{reg}}$$

OTHER PARTS

	TREG	TIRR	TPRED	TCOMM	THOST
N	$O(N^{2.11})$	$O(N^{1.76})$	$O(N^{1.51})$ $f(N^x, N\log(N)^y)$	TMOV: $O(N^{1.48})$ TSUB: $O(N^{1.21})$ TBAR: $O(N^{1.34})$	$O(N^{1.62})$ $f(N^x, N\log(N)^y)$
N_{PE}	$O({N_{PE}}^{-1.0})$	$O({N_{PE}}^{-1.0})$	$O({N_{PE}}^{-0.3})$	TMOV: $O(1)$ TSUB: $O(\log(N_{PE}))$ TBAR: $O(N_{PE})$	$O(1)$

CONCLUSIONS

The main parts of the code (REG & IRR) have an extremely good scaling with NPE.

As the increasing of NPE or/and N, these computational parts are not always acted as a dominant roles any longer.

Disturbances from the running of other programs will cause a shapely decreasing of performance.