GOTHIC: Gravitational Oct-Tree code accelerated by Hierarchical time step Controlling

Yohei Miki, Masayuki Umemura

University of Tsukuba / CREST, JST

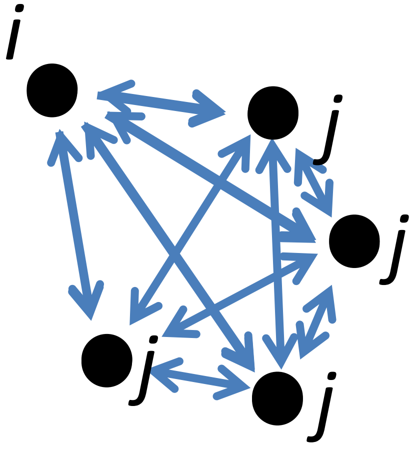
Outline

- Introduction
  - *N*-body simulation in astrophysics
  - Tree method
  - Block time step (hierarchical time step)
- Implementation and Optimization
- Performance Measurements
- Summary

What is *N*-body simulation ?

- Calculating time evolution of gravitational many body system by solving Newton's equation of motion
  - data: $O(N)$
  - force calculation: $O(N^2)$
  - time integration: $O(N)$
- *i*-particle: particle feels gravity
- *j*-particle: particle causes gravity

$$\boldsymbol{a}_i = \sum_{\substack{j=0 \\ j \neq i}}^{N-1} \frac{G m_j \left(\boldsymbol{x}_j - \boldsymbol{x}_i\right)}{\left(\left|\boldsymbol{x}_j - \boldsymbol{x}_i\right|^2 + \epsilon^2\right)^{3/2}}$$

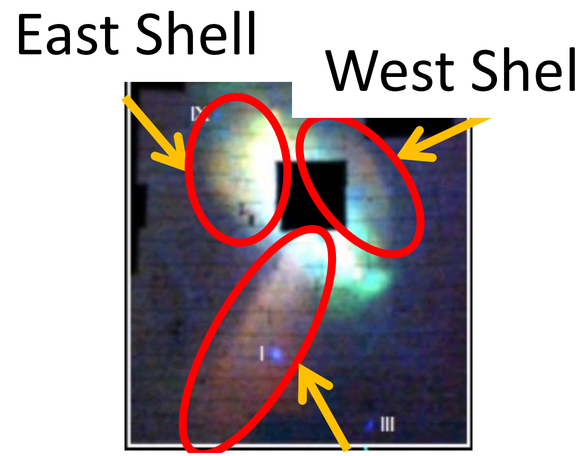


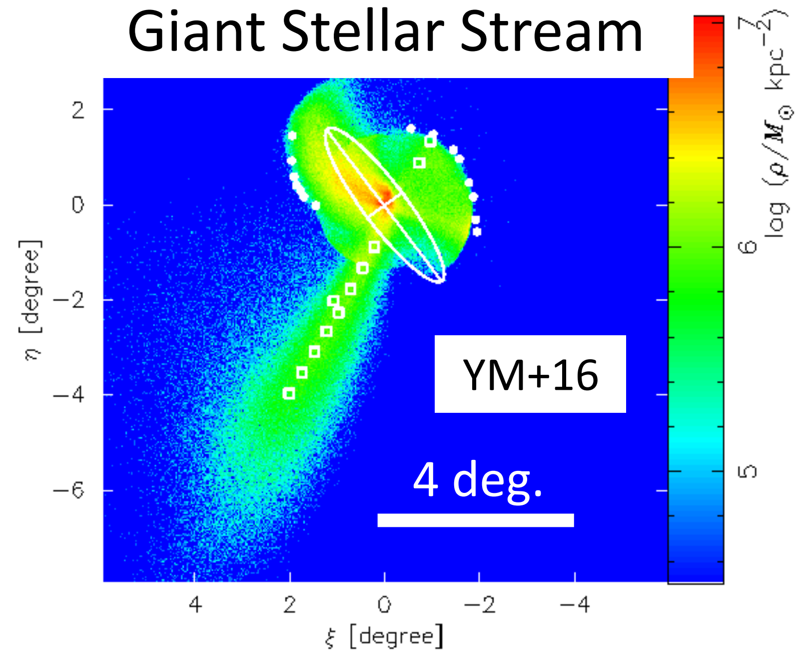
Astrophysical *N*-body simulation (1/2)

- Formation of the large scale structure
- Merger of galaxies
- Dynamical evolution of galaxies
(e.g., spiral arms)

- Many particles as much as possible is required.
  – $N \sim 10^6$ is the upper limit for direct summation.

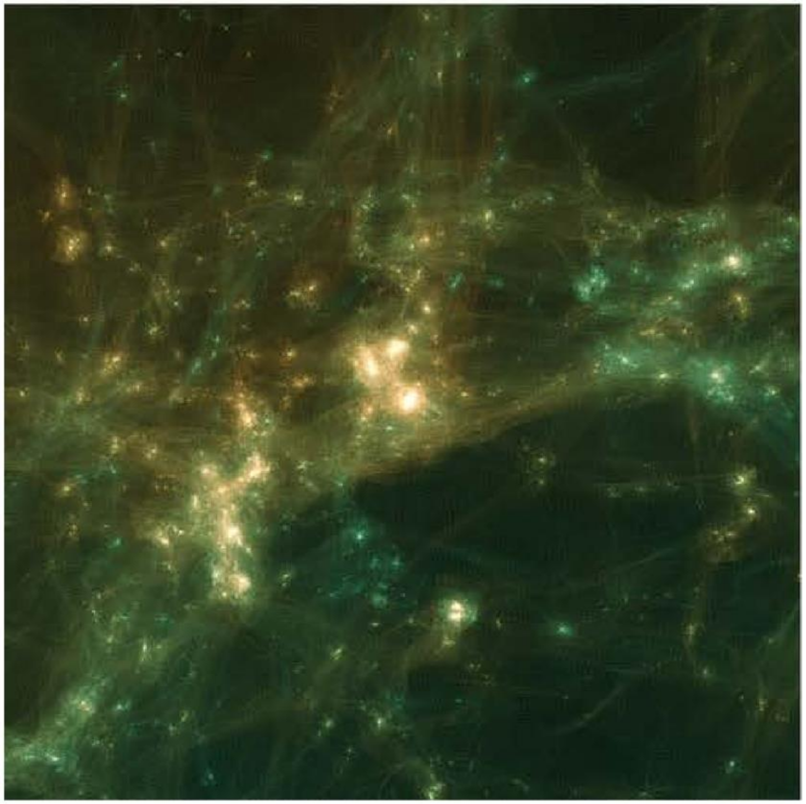
McConnachie et al. (2009)





Astrophysical *N*-body simulation (2/2)

- *N* exceeds 10^{12} (one trillion!) in recent cosmological simulations
  - Ishiyama et al. 2012 (Gordon-Bell winner)



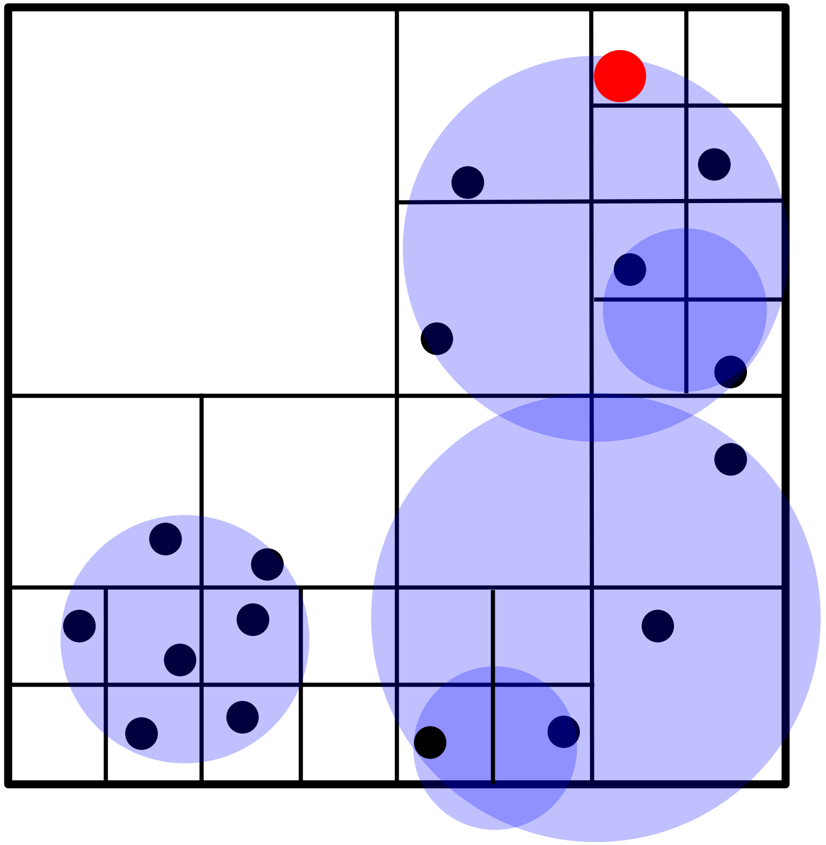
- To deal with such a large *N*, we use some techniques
  - tree-method: reduce N_j (reduce gravity calculation using multipole expansion)
  - block time step: reduce N_i (reduce orbit integration)

Acceleration of N-body simulation (1/2)

- Tree method (Barnes & Hut 1986)
  - Reduce the effective number of j-particles by multipole expansion.
  - $O(N_i N_j) \rightarrow O(N_i \log N_j)$
  - Most familiar criterion:

$$\frac{b_{\max}}{d} \leq \theta$$

  - Accelerated on GPU(s) (Nakasato12, Ogiya+13, Bédorf+12, 14 etc.)



Acceleration of *N*-body simulation (2/2)

- Individual time step (Aarseth 1969)
  - Define time step for every particles, calculate gravity and orbit integration for selected particles.
  - Reduce N_i in $O(N_i N_j)$
- Block time step (McMillan 1986)
  - Sometimes called hierarchical time step.
  - Define time step for every group of particles.
  - Suitable for parallelization than individual time step.
  - In most cases, Δt is set to be 2^{-n}.

Our implementation

- Tree code with block time step on GPU
  - Earlier studies: tree code on GPU(s) with shared time step (Nakasato12, Ogiya+13, Bédorf+12, 14)
- Barnes-Hut tree
  - Reduce effective number of *j*-particles using multipole expansion.
- Block time step (⇔ shared time step)
  - Reduce effective number of *i*-particles by reducing orbit integration about particles whose orbit evolution are slow.

Auto-tuning: tree rebuild interval (1/2)

- There is no requirement to rebuild the tree structure every time step.
- If we reuse the old tree structure…
  - Pros: reusing the old tree structure reduce additional cost to rebuild the tree structure.
  - Cons: the mismatch between the tree structure and the actual particle distribution would increase the execution time.
- There is an optimal interval to rebuild the tree structure and finding it is a task suited to auto-tuning.

Auto-tuning: tree rebuild interval (2/2)

- We introduce toy models to fit the execution time and predict the optimal rebuild interval *n*.
- In linear growth model (the simplest model):
  - Execution time is modeled as:

$$t_{\mathrm{tot}} = t_{\mathrm{make}} + n(t_1 - \Delta t) + \frac{n(n+1)}{2}\Delta t$$

  - Optimal rebuild interval is:

$$n^2 = \frac{2}{\Delta t} t_{\mathrm{make}}$$

- The optimal interval depends on the particle distribution or the utilized GPU.

Performance Measurements

- The elapsed time is evaluated as the wall clock time per time step (total number of steps is fixed to 1024)
  - to include the effects of block time step and auto-tuning
  - it also includes the time required to read/write files and allocate/deallocate memory
- Particle distribution: the Andromeda galaxy model
  - Mass distribution (bulge, disk, and halo) is determined by Geehan et al. (2006) and Fardal et al. (2007)
  - Particle distribution is generated by MAGI (YM & Umemura in prep.) as a dynamical equilibrium system
  - Number of particles is 8M (= 8,388,608)

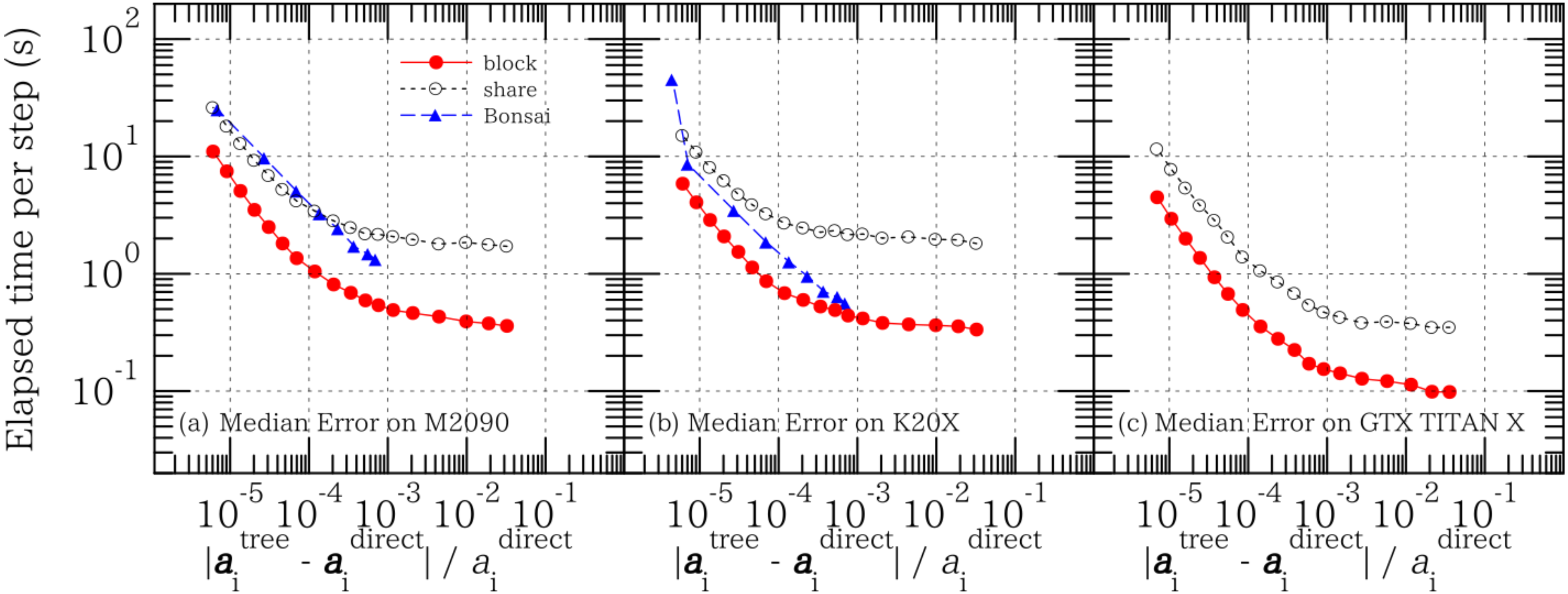
Evaluation Environment

- We have used NVIDIA Tesla M2090, K20X, and GeForce GTX TITAN X with CUDA 7.5.

System	HA-PACS/BC	HA-PACS/TCA	Workstation
Number of nodes	268	64	1
CPU	Intel Xeon E5-2670	Intel Xeon E5-2680 v2	Intel Xeon E5-2640 v3
	8 cores, 2.6 GHz	10 cores, 2.8 GHz	8 cores, 2.6 GHz
	2 sockets per node		2 sockets
RAM	DDR3-1600, 8 channels	DDR3-1866, 8 channels	DDR4-2133, 8 channels
	128 GB per node		64 GB
GPU	NVIDIA Tesla M2090	NVIDIA Tesla K20X	NVIDIA GeForce GTX TITAN X
	512 cores, 1.3 GHz	2688 cores, 732 MHz	3072 cores, 1 GHz
	4 boards per node		2 boards
Video RAM	6 GB (GDDR5, ECC on) per GPU		12 GB (GDDR5) per GPU
C Compiler	icc 15.0.5.223 (gcc 4.4.7 compatibility)		gcc 4.8.5
CUDA Toolkit	7.5.17		

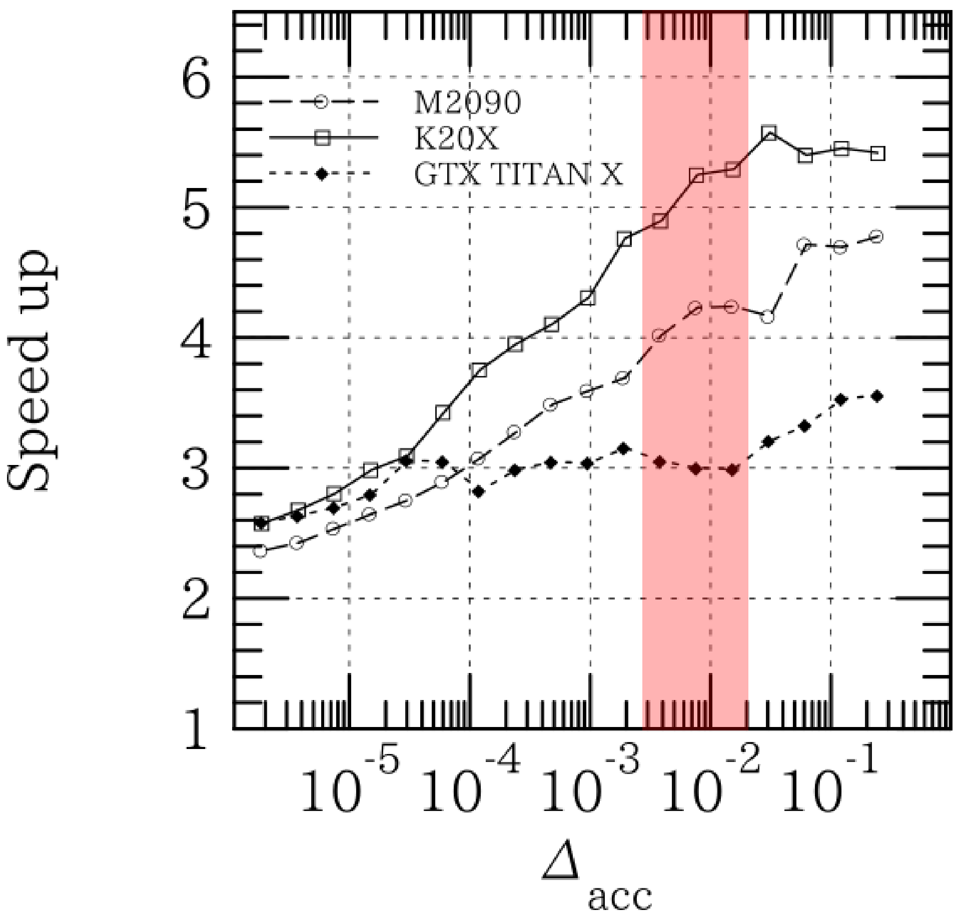
Measured execution time

- On all GPUs, block time step is faster than
  - shared time step
  - public code ``Bonsai'' (Bédorf et al. 2012, 2014)



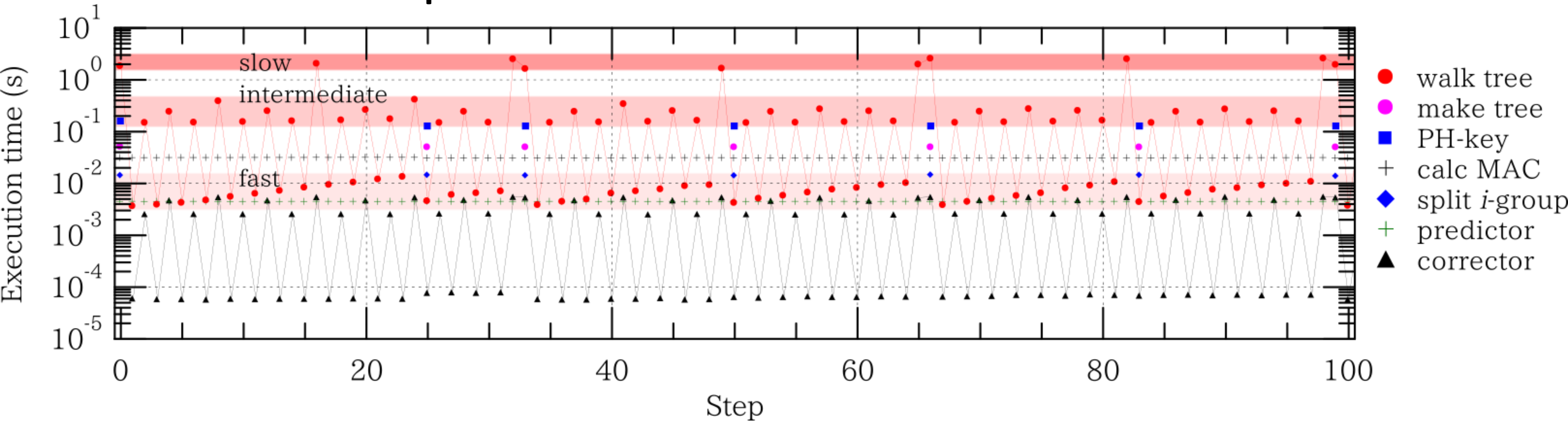
Speed up from shared time step

- Block time step is at least 2X faster.
- In typical force error, it is 3—5 times faster.
- When increasing the accuracy of gravity calculations,
  - the number of calculations in high density regions increases
  - the increase weakens the benefits of the block time step.



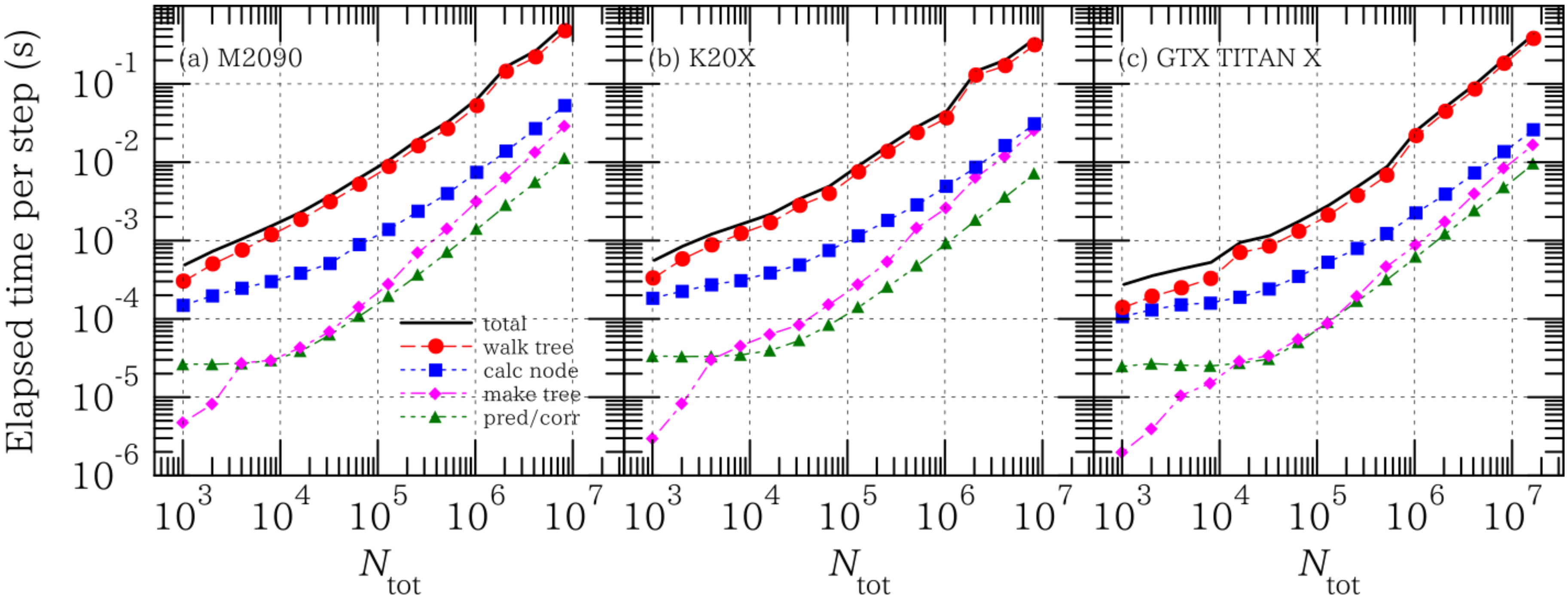
Execution time in each step

- Execution time of each function as a function of the time step.
  - There are 3 distinct ranges of execution times for calculating gravity due to differences of N_i in each time step.



Breakdown, dependence on N_{tot}

- Observed scaling is roughly proportional to *N*.
- Tree traversal always dominates.



Achieved performance

- We have measured execution time and number of interactions independently.
- The averaged performance is 10—30 % of the theoretical peak performance in SP.

GPU	Model[a]	Number of interactions per second			Achieved performance[b] (GFlop/s)		
		average	maximum	minimum	average	maximum	minimum
M2090	NFW	1.06×10^{10}	1.92×10^{10}	3.87×10^{9}	296	536	108
	M31	1.20×10^{10}	1.86×10^{10}	4.90×10^{9}	336	521	137
K20X	NFW	1.45×10^{10}	3.40×10^{10}	3.77×10^{9}	377	885	98
	M31	1.34×10^{10}	3.30×10^{10}	3.81×10^{9}	349	859	99
GTX TITAN X	NFW	6.77×10^{10}	1.59×10^{11}	2.49×10^{10}	1626	3827	598
	M31	7.80×10^{10}	1.50×10^{11}	2.46×10^{10}	1871	3595	590

Summary

- We have developed a gravitational octree code GOTHIC (Gravitational Oct-Tree code accelerated by HIerarchical time step Controlling), which runs entirely on GPU.
  - Block time step
  - Auto-tuning about tree rebuild interval
- Results of performance measurements
  - Block time step enables 3-5X acceleration.
  - Observed scaling is near proportional to N.
  - Performance reaches 30% of GPU performance.