ChaNGa: The Charm N-Body GrAvity Solver

Filippo Gioachin[1]
Pritish Jetley[1]
Celso Mendes[1]
Laxmikant Kale[1]
Thomas Quinn[2]

[1] University of Illinois at Urbana-Champaign
[2] University of Washington

Outline

- Motivations
- Algorithm overview
- Scalability
- Load balancer
- Multistepping

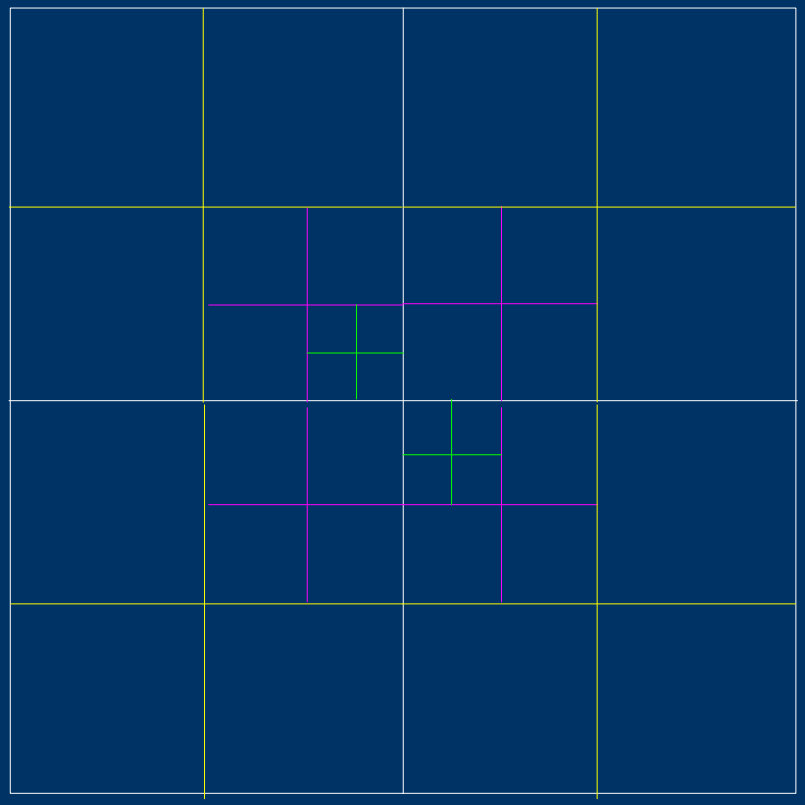
Motivations

- Need for simulations of the evolution of the universe
- Current parallel codes:
  - PKDGRAV
  - Gadget
- Scalability problems:
  - load imbalance
  - expensive domain decomposition
  - limit to 128 processors

ChaNGa: main characteristics

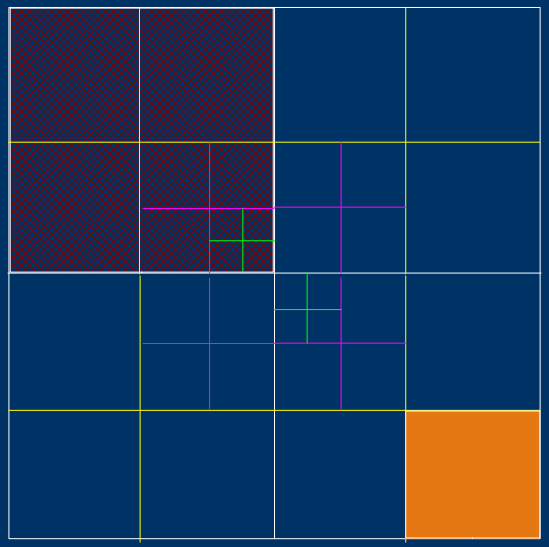
- Simulator of cosmological interaction
  - Newtonian gravity
  - Periodic boundary conditions
  - Multiple timestepping
- Particle based (Lagrangian)
  - high resolution where needed
  - based on tree structures
- Implemented in Charm++
  - work divided among chares called *TreePieces*
  - processor-level optimization using a Charm++ group called *CacheManager*

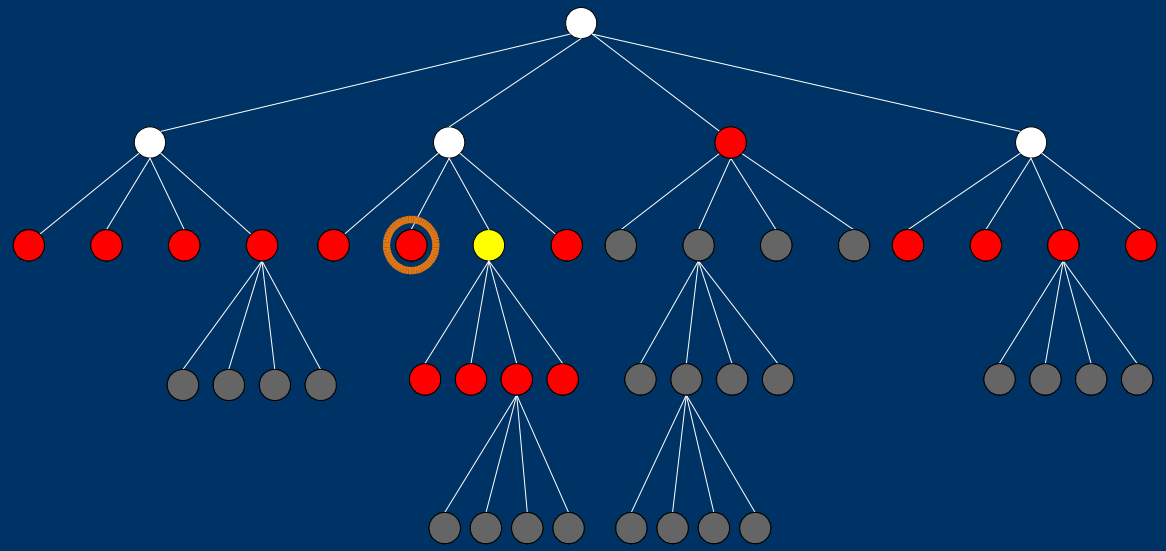
Space decomposition



Basic algorithm ...

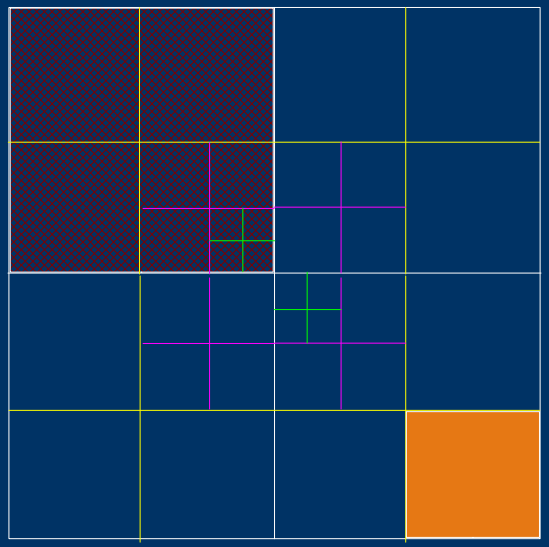
- Newtonian gravity interaction
  - Each particle is influenced by all others: O(n^2) algorithm
- Barnes-Hut approximation: O($n$log$n$)
  - Influence from distant particles combined into center of mass

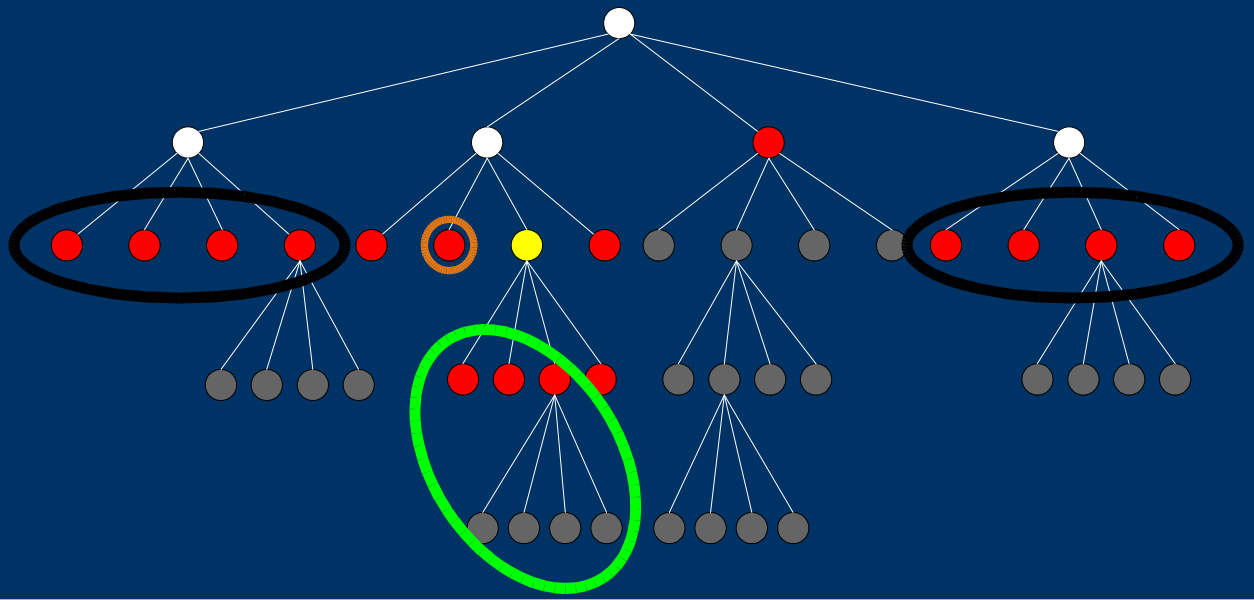




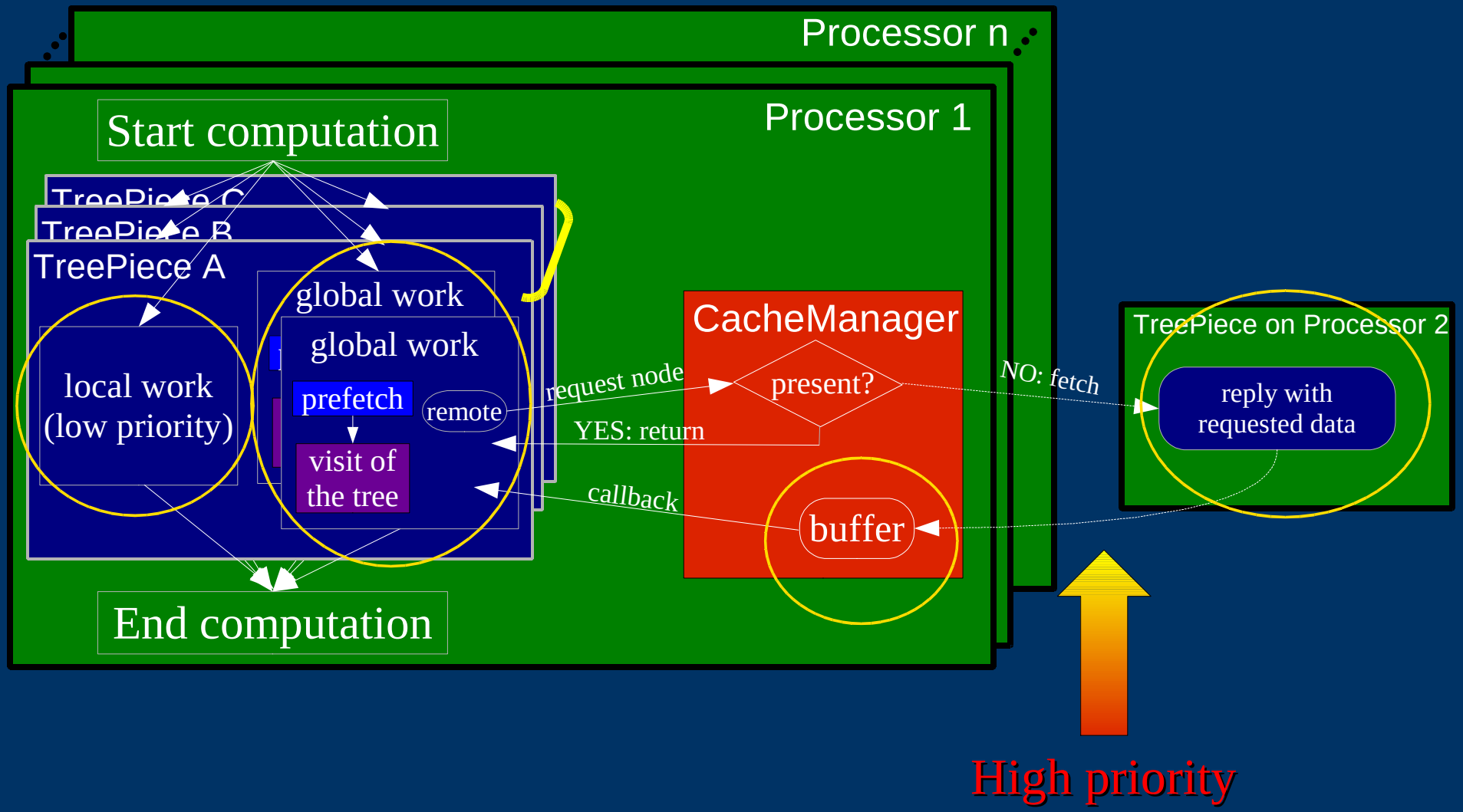
... in parallel

- Remote data
  - need to fetch from other processors
- Data reusage
  - same data needed by more than one particle

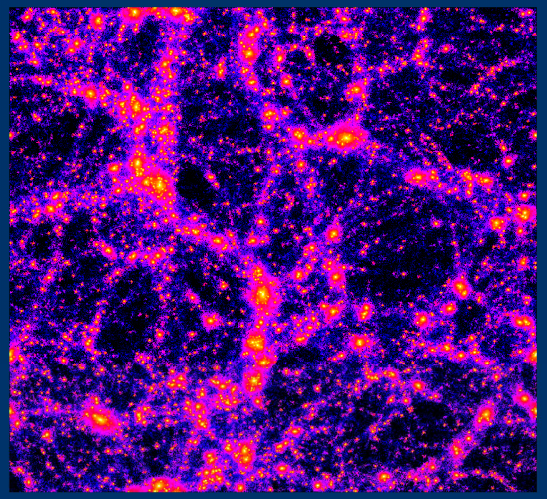




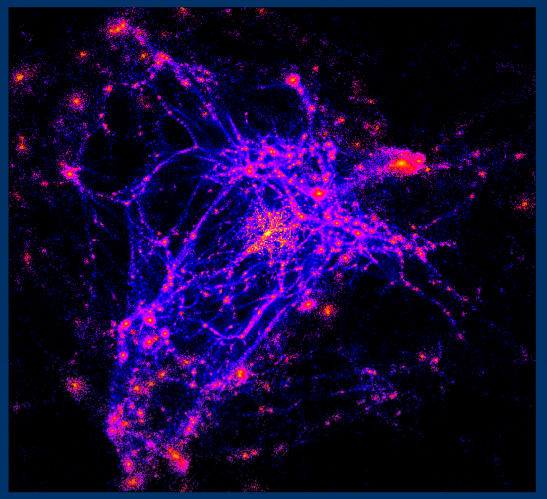
Overall algorithm



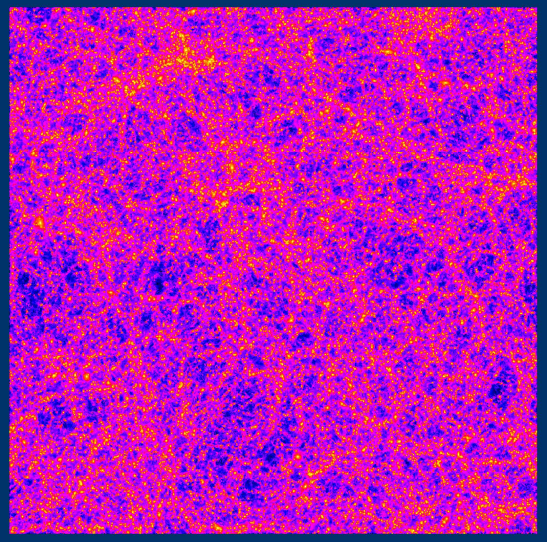
Datasets



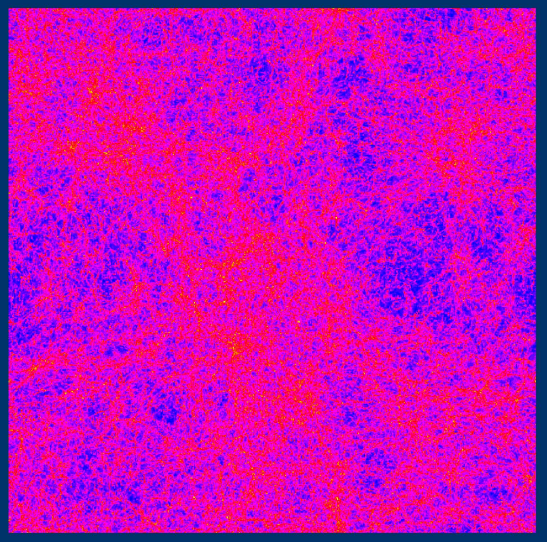
lambs
3 million particles (47 MB)
with subsets



dwarf 5 and 50 million particles (80 MB and 1,778 MB)



hrwh_LCDMs
16 milllion particles (576 MB)



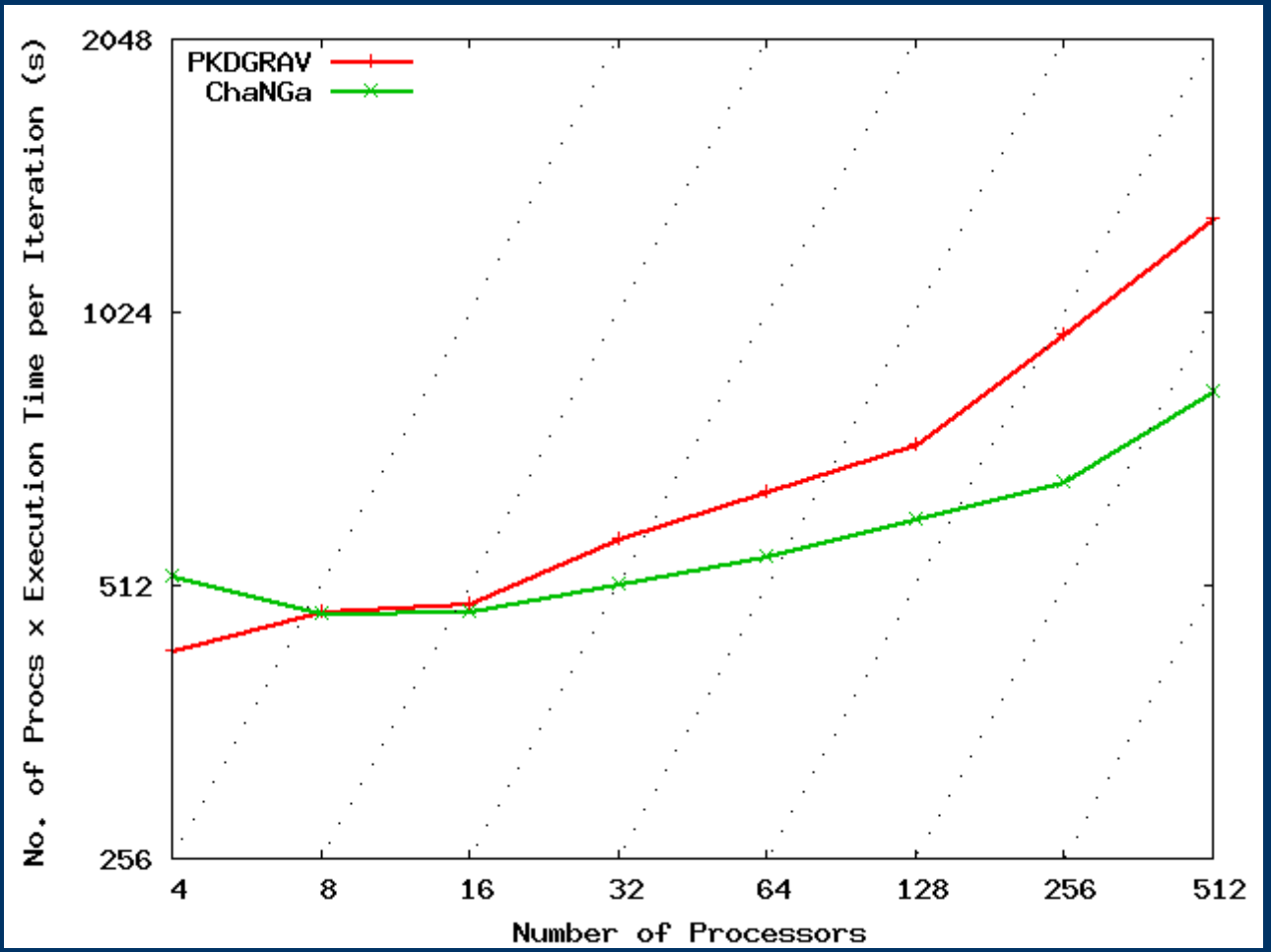
drgas
700 million particles (25.2 GB)

Systems

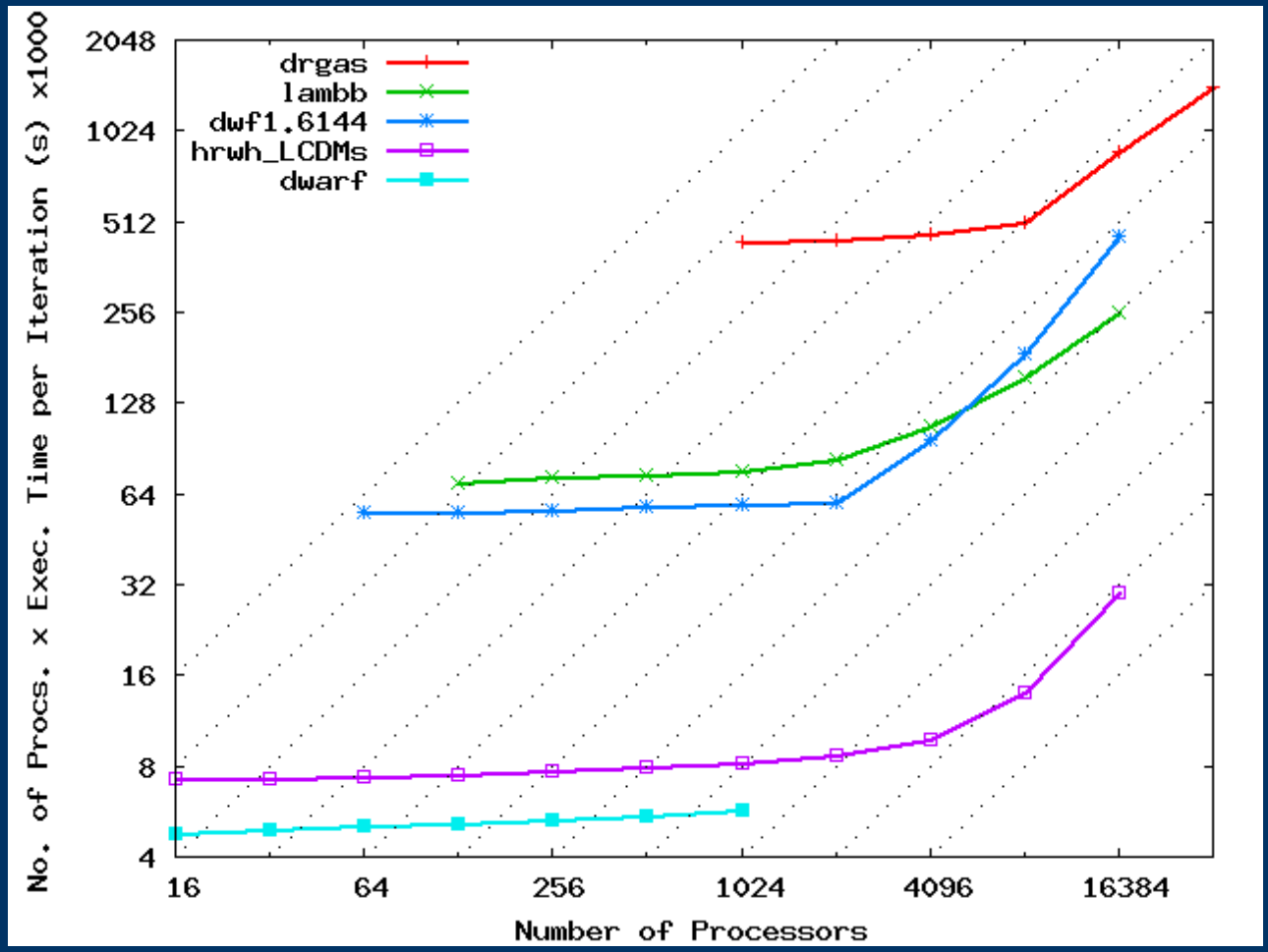
System	Location	Procs	Procs per node	CPU	Memory per node	Network
Tungsten	NCSA	2,560	2	Xeon 3.2 Ghz	3 GB	Myrinet
Cray XT3	Pittsburgh	4,136	2	Opteron 2.6GHz	2 GB	Torus
BlueGene/L	IBM-Watson	40,000	2	Power440 700MHz	512 MB	Torus

Scaling: comparison

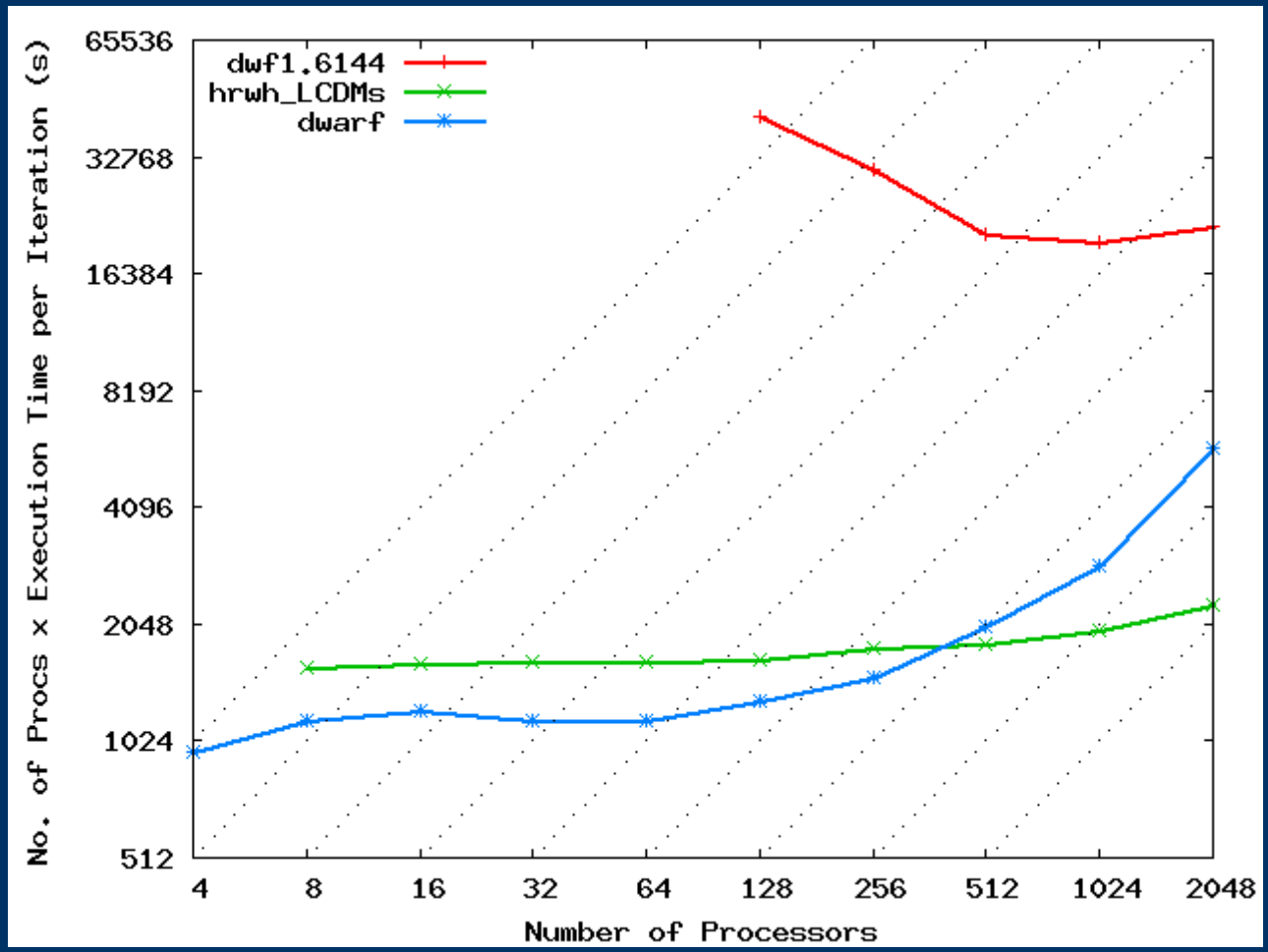
lambs 3M on Tungsten



Scaling: IBM BlueGene/L

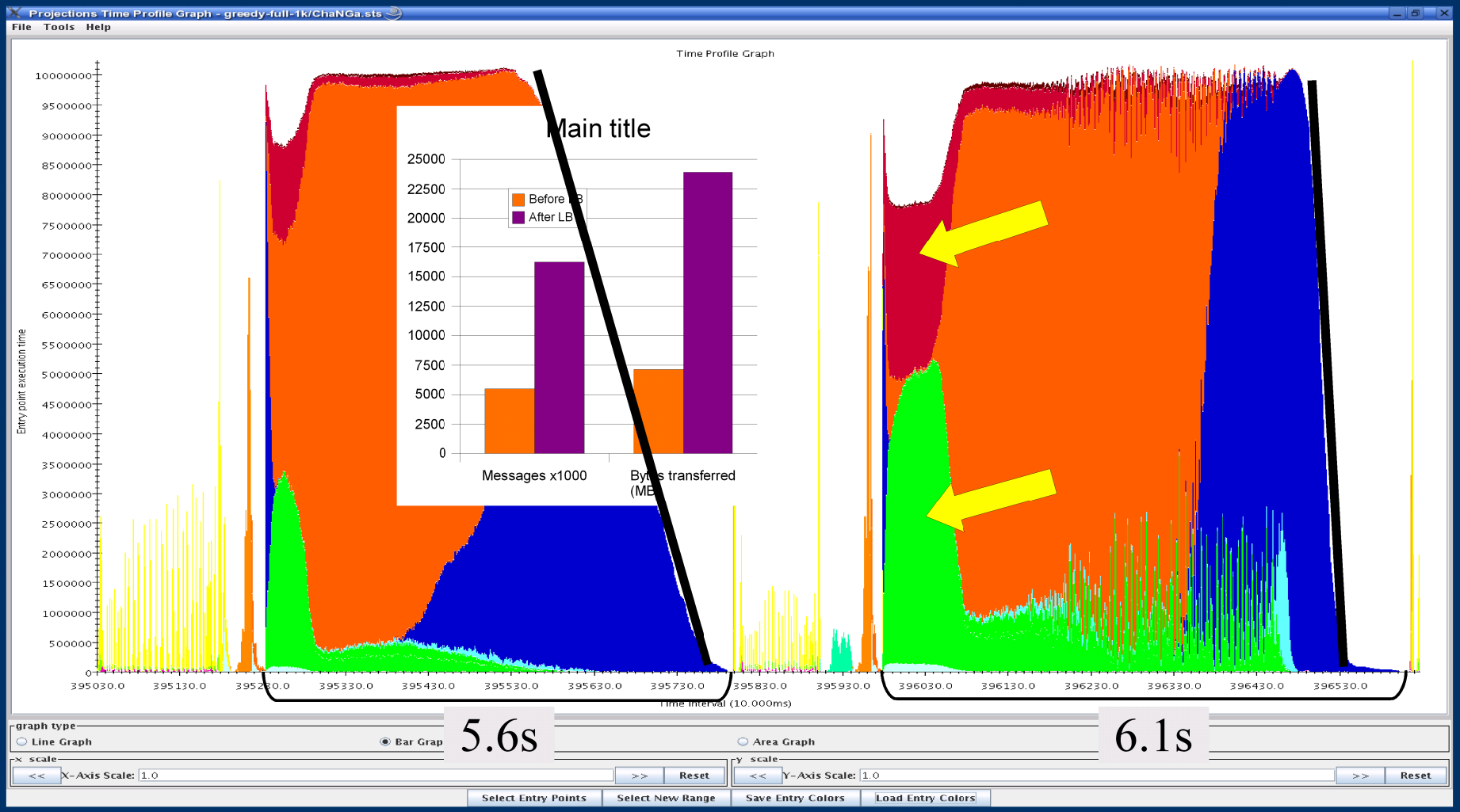


Scaling: Cray XT3



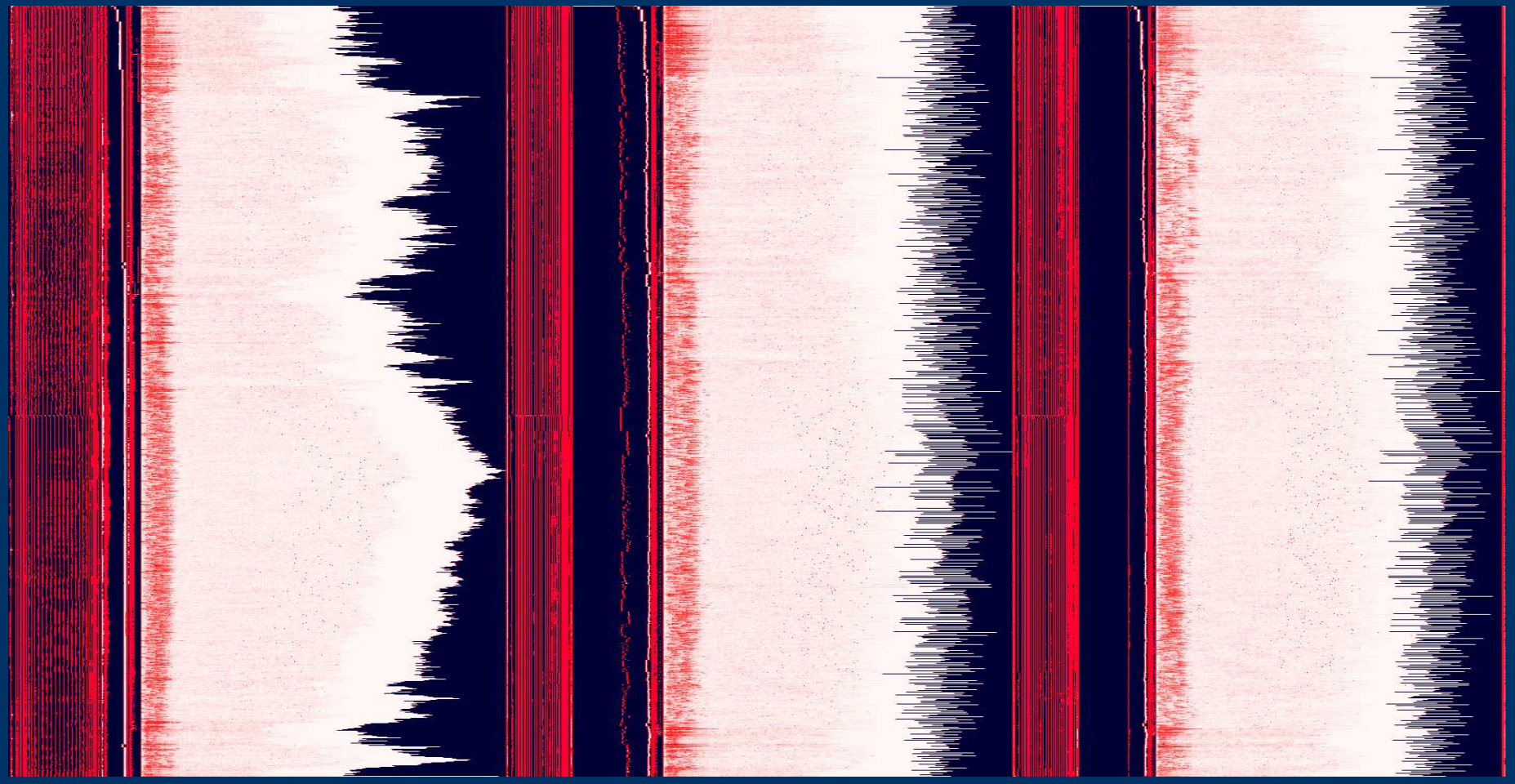
Load balancing with GreedyLB

dwarf 5M on 1,024 BlueGene/L processors



Load balancing with OrbLB

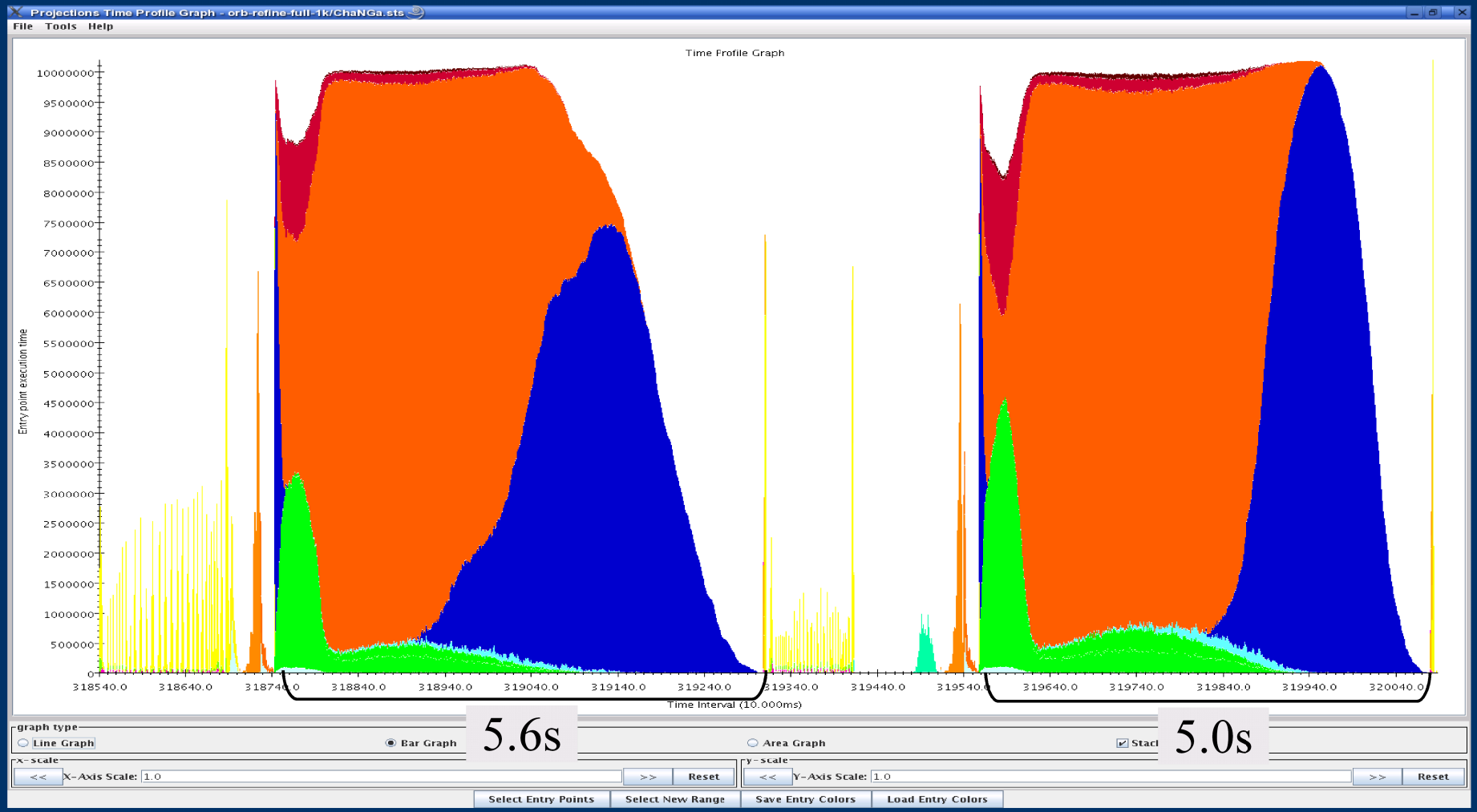
lambs 5M on 1,024 BlueGene/L processors



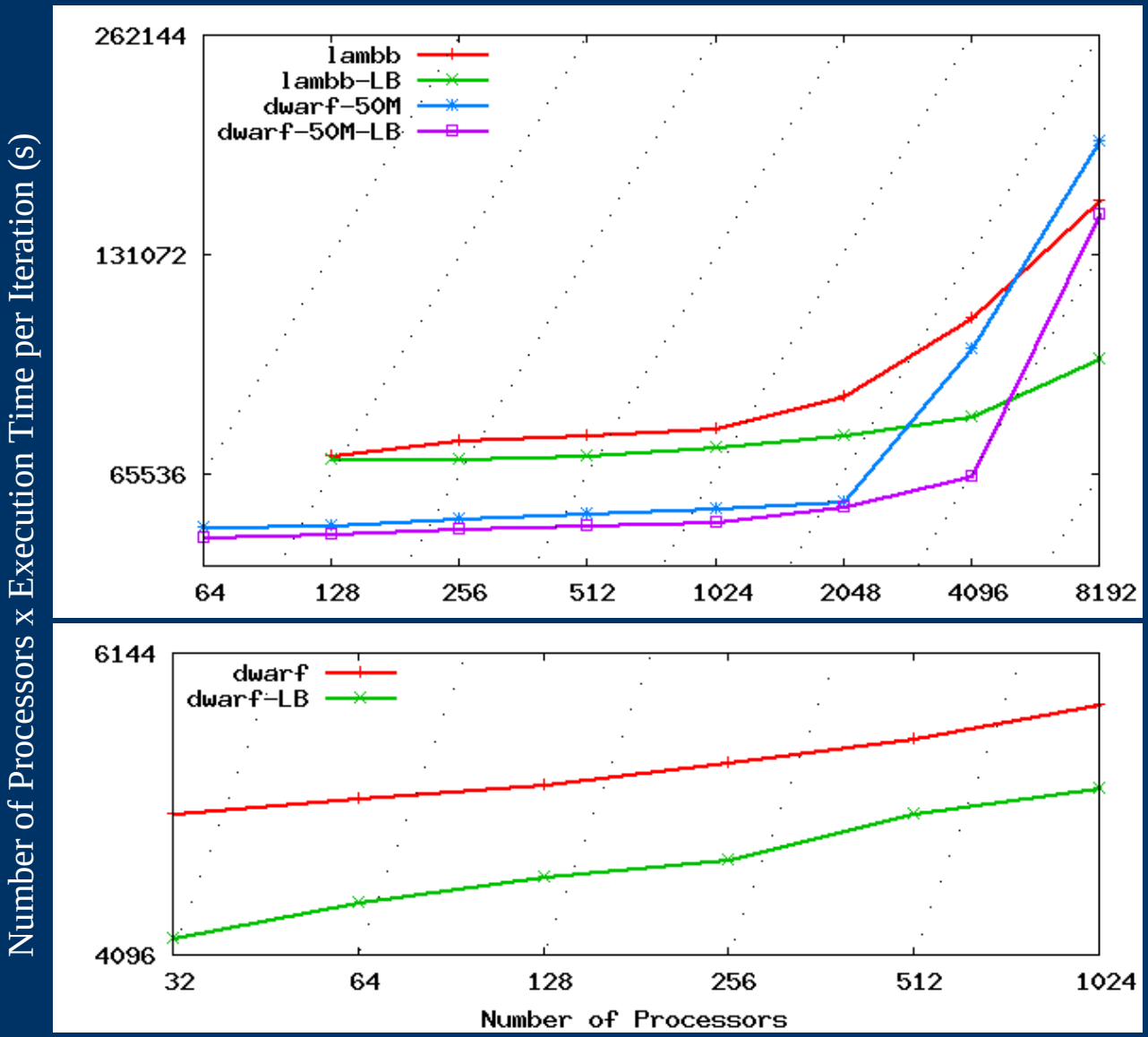
white is good

Load balancing with OrbRefineLB

dwarf 5M on 1,024 BlueGene/L processors



Scaling with load balancing

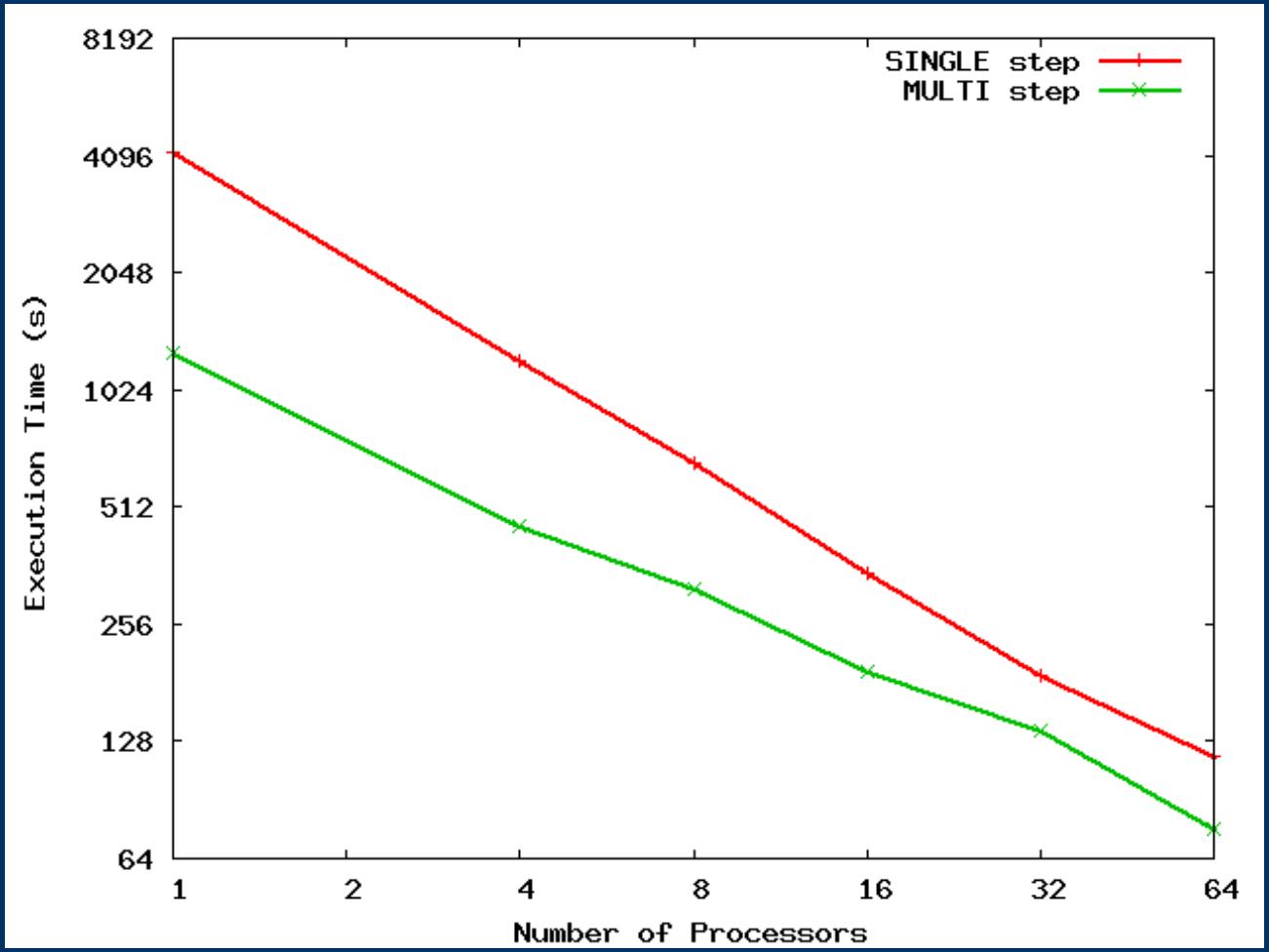


Multistepping

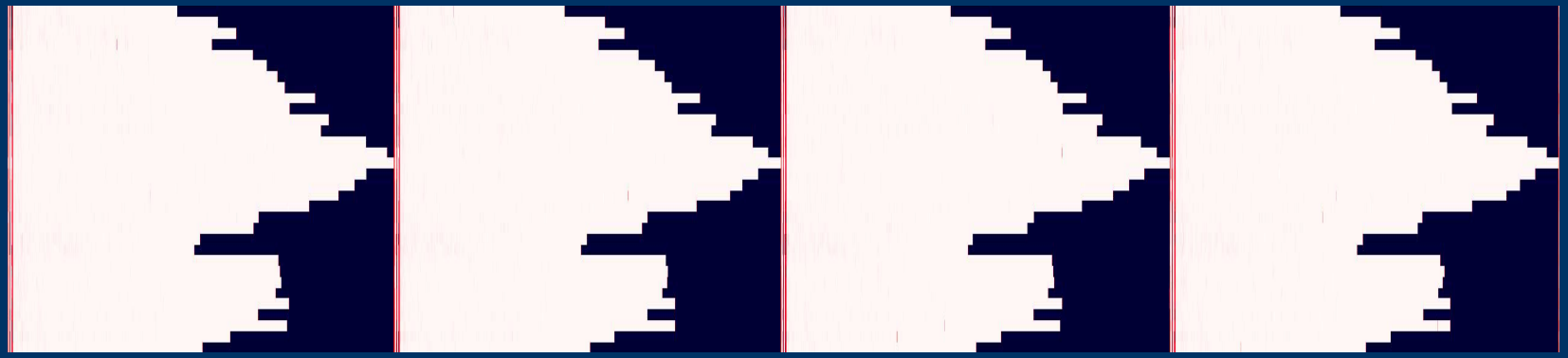
- Particles with higher accelerations require smaller integration timesteps to be accurately predicted.
- Compute particles with highest accelerations every step, and particles with lower accelerations every few steps.
- Steps become different in terms of load.

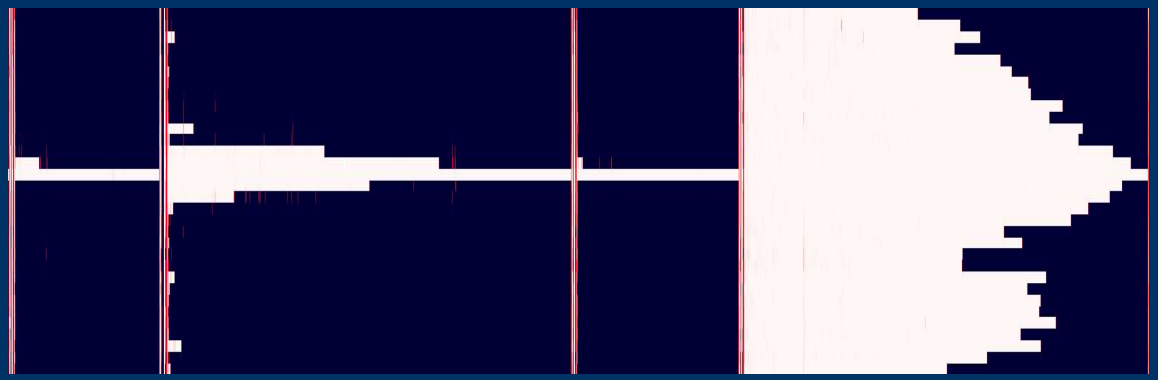
ChaNGa scalability - multistepping

dwarf 5M on Tungsten



ChaNGa scalability - multistepping





Future work

- Adding new physics
  - Smoothed Particle Hydrodynamics
- More load balancer / scalability
  - Reducing overhead of communication
  - Load balancing without increasing communication volume
  - Multiphase for multistepping
  - Other phases of the computation

Questions?

Thank you

Decomposition types

- OCT
  - Contiguous cubic volume of space to each TreePiece
- SFC – Morton and Peano-Hilbert
  - Space Filling Curve imposes total ordering of particles
  - Segment of this line to each TreePiece
- ORB
  - Space divided by Orthogonal Recursive Bisection on the number of particles
  - Contiguous non-cubic volume of space to each TreePiece
  - Due to the shapes of the decomposition, requires more computation to produce correct results

Serial performance

Execution Time on Tungsten (in seconds)

Simulator	Lambs datasets			
	30,000	300,000	1,000,000	3,000,000
PKDGRAV	0.8	12.0	48.5	170.0
ChaNGa	0.8	13.2	53.6	180.6
Time difference	0.00%	9.09%	9.51%	5.87%

CacheManager importance

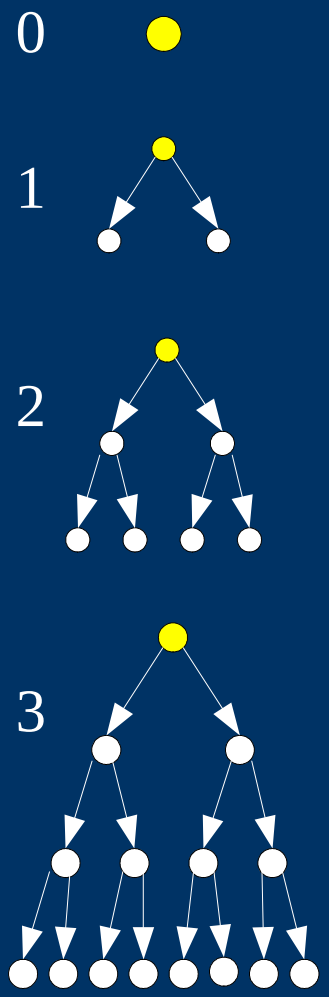
1 million lambs dataset on HPCx

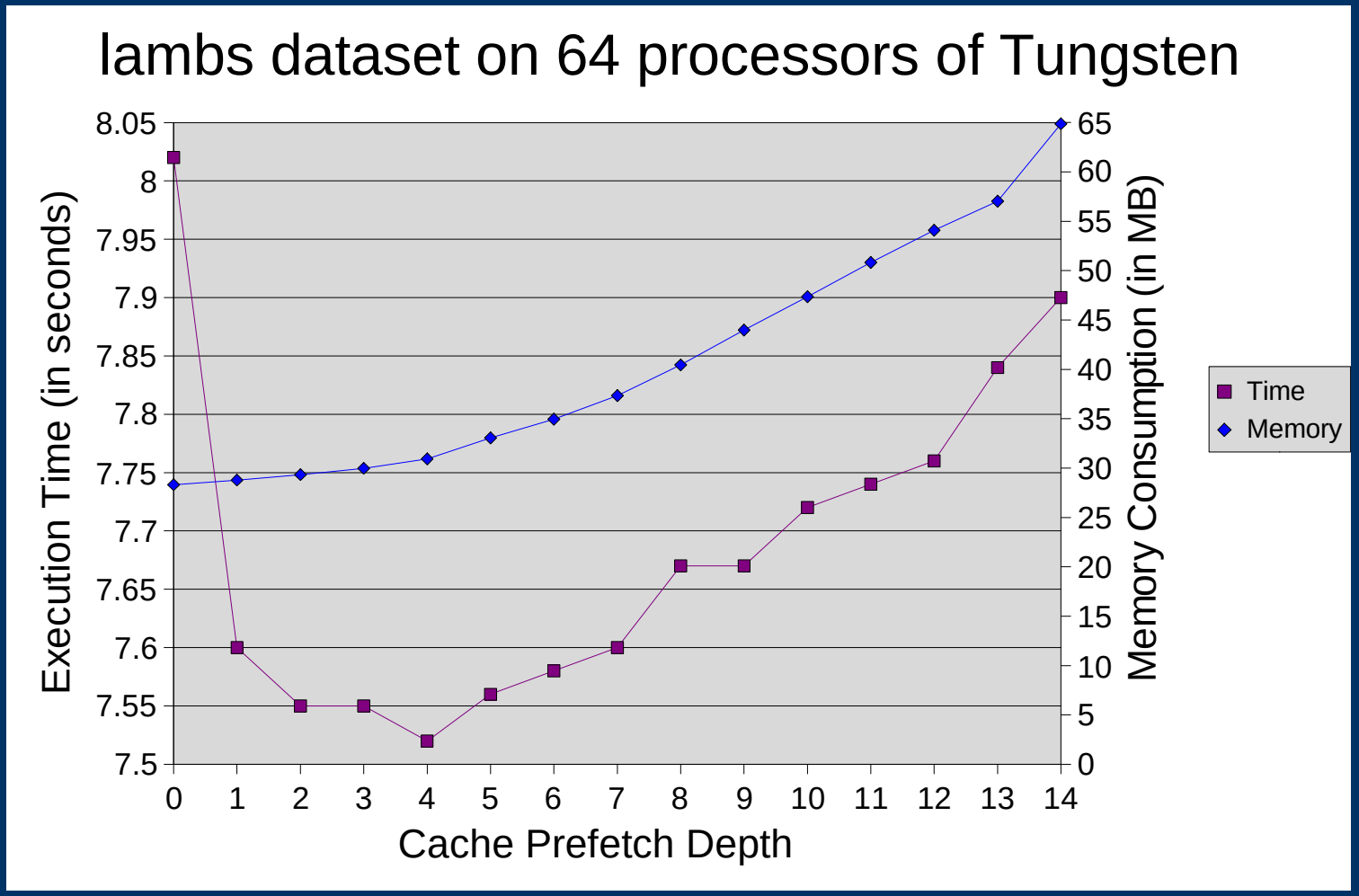
		Number of Processors				
		4	8	16	32	64
Number of messages (in thousand)	No Cache	48,723	59,115	59,116	68,937	78,086
	With Cache	72	115	169	265	397
Time (seconds)	No Cache	730.7	453.9	289.1	67.4	42.1
	With Cache	39.0	20.4	11.3	6.0	3.3
Speedup		18.74	22.25	25.58	11.23	12.76

Prefetching

1) explicit
   - before force computation, data is requested for preload
2) implicit in the cache
   - computation performed with tree walks
   - after visiting a node, its children will likely be visited
   - while fetching remote nodes, the cache prefetches some of its children

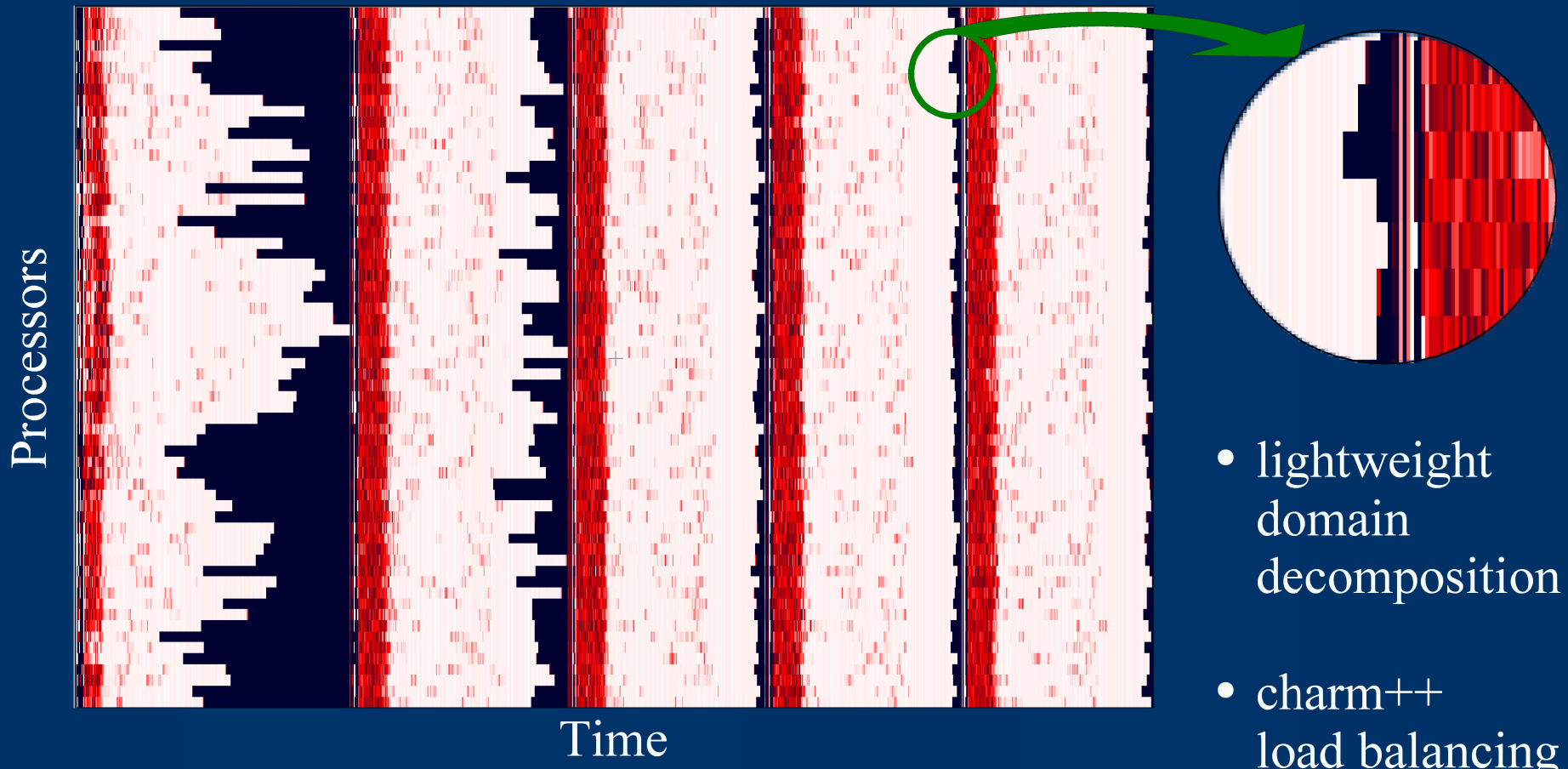
Cache implicit prefetching





Load balancer

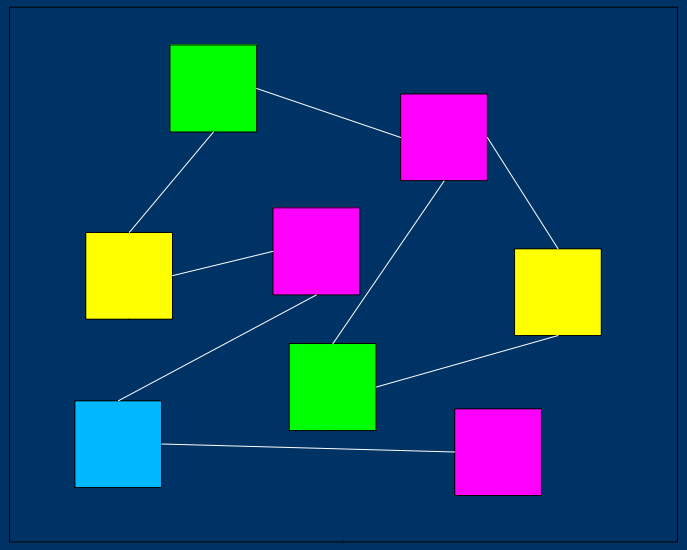
lambs 300K subset on 64 processors of Tungsten



- lightweight domain decomposition
- charm++ load balancing

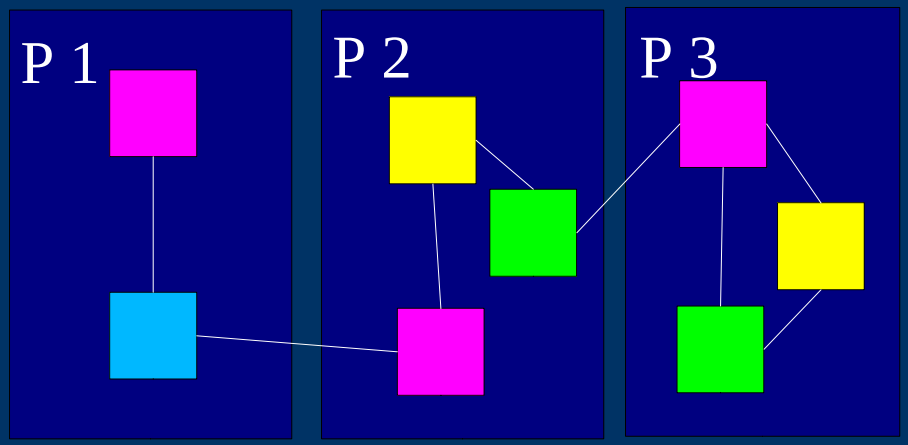
while: high utilization
dark: processor idle

Charm++ Overview



User view

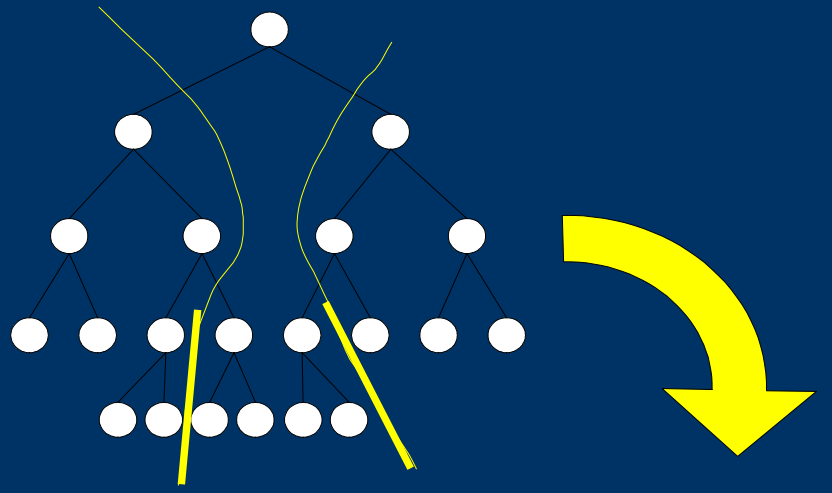
System view



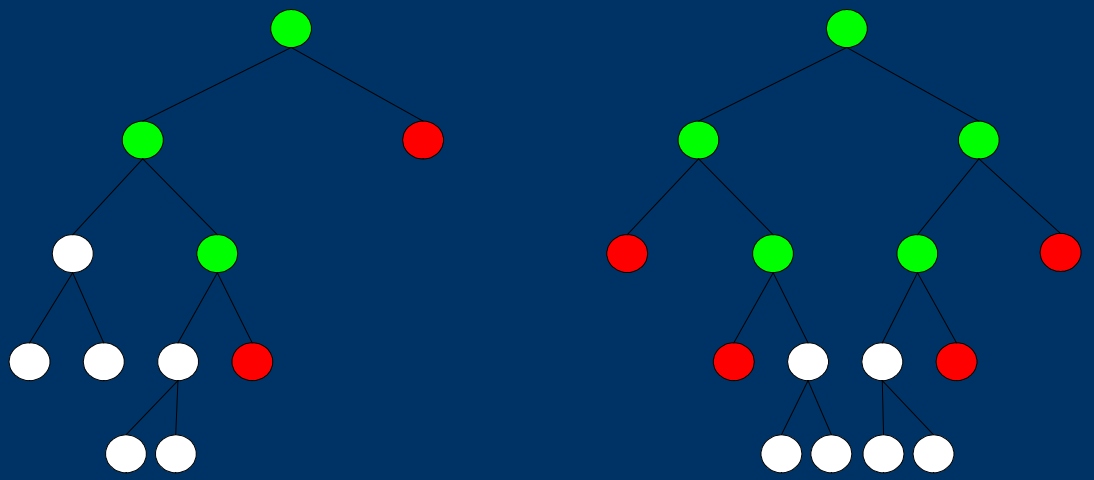
- work decomposed into objects called *chares*
- message driven

- mapping of objects to processors transparent to user
- automatic load balancing
- communication optimization

Tree decomposition

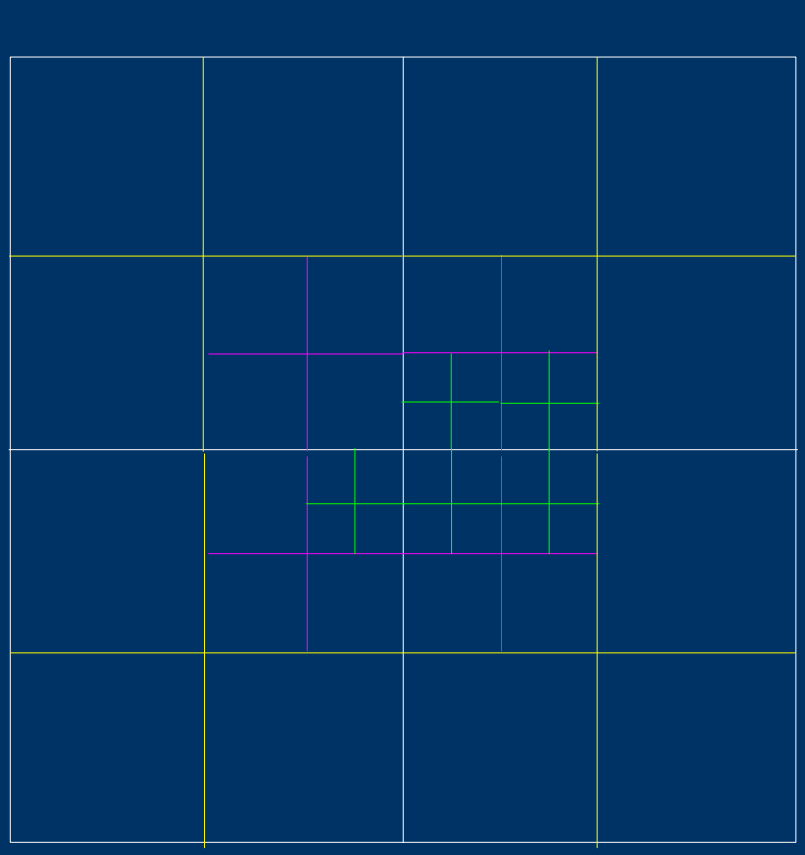


Exclusive
Shared
Remote

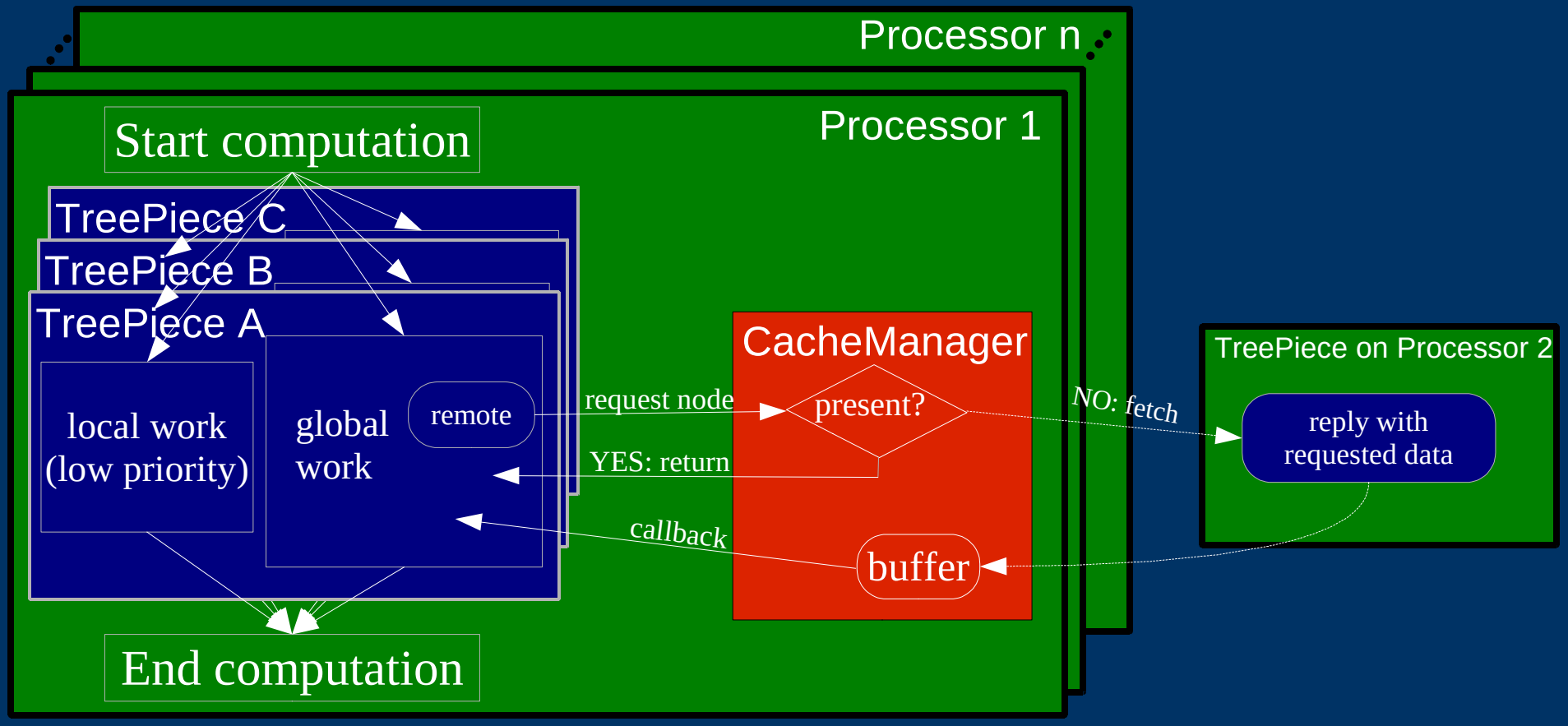


TreePiece 1 TreePiece 2 TreePiece 3

Space decomposition

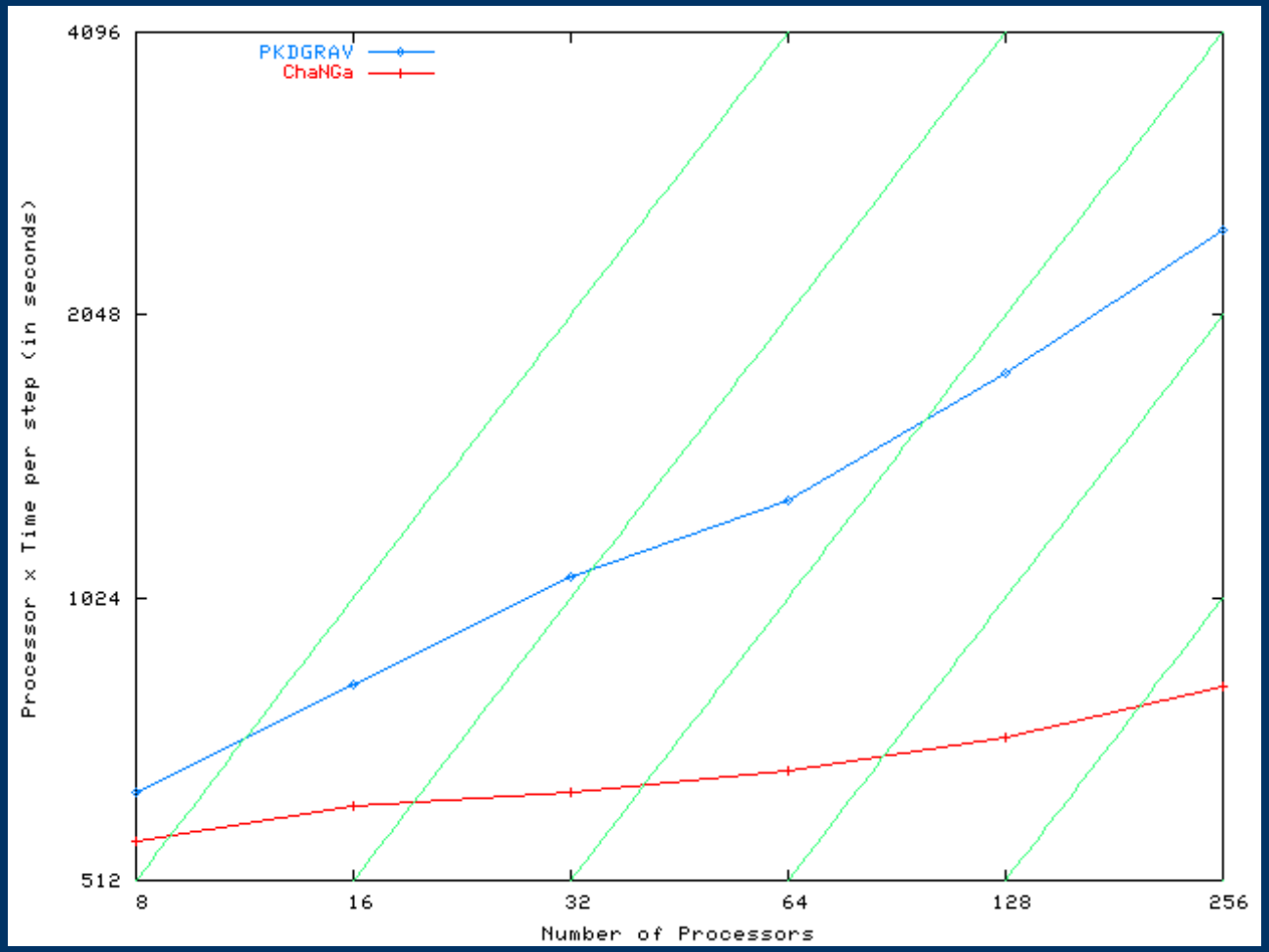


Overall algorithm



Scalability comparison (old result)

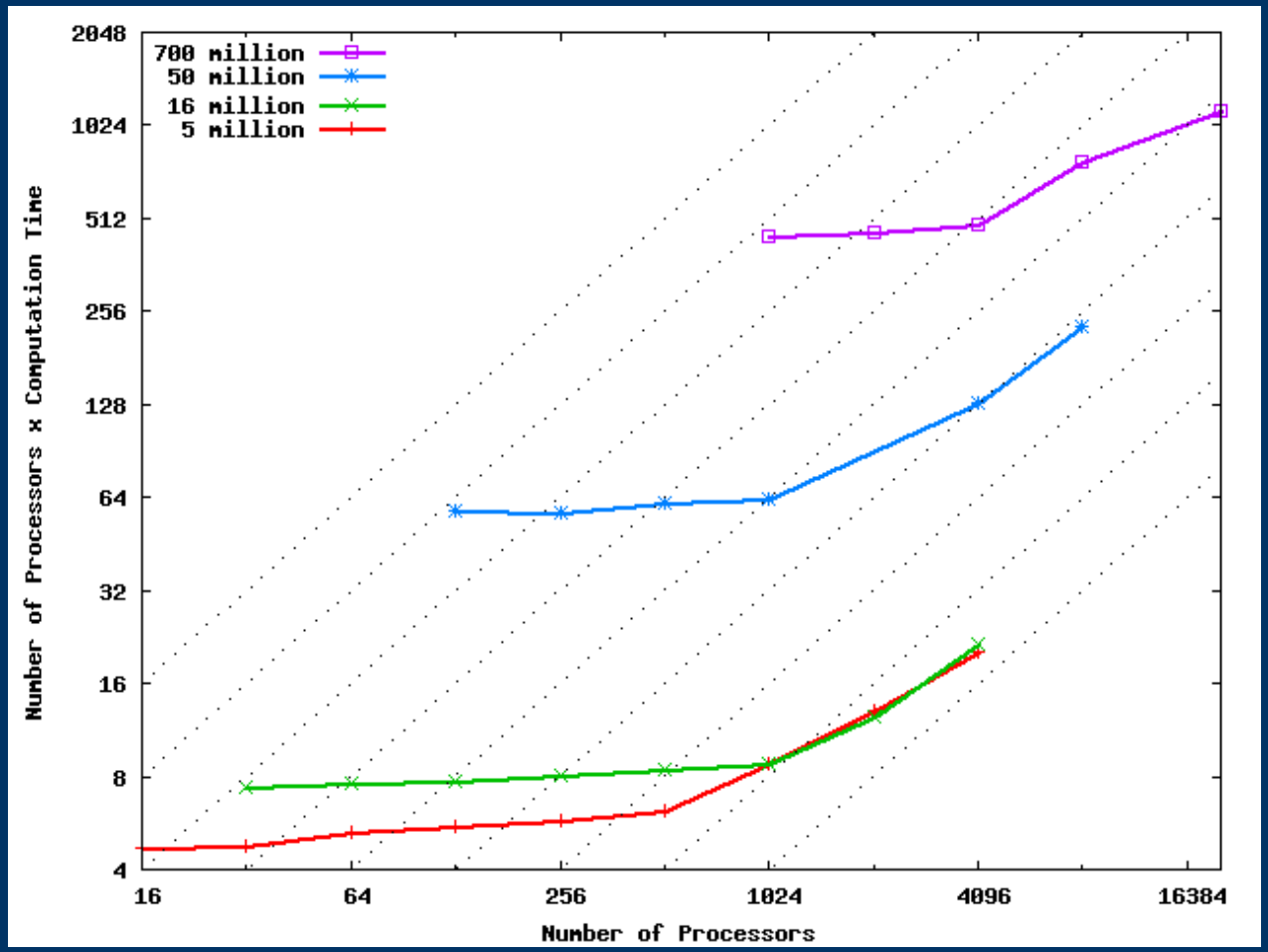
dwarf 5M comparison on Tungsten



flat: perfect scaling diagonal: no scaling

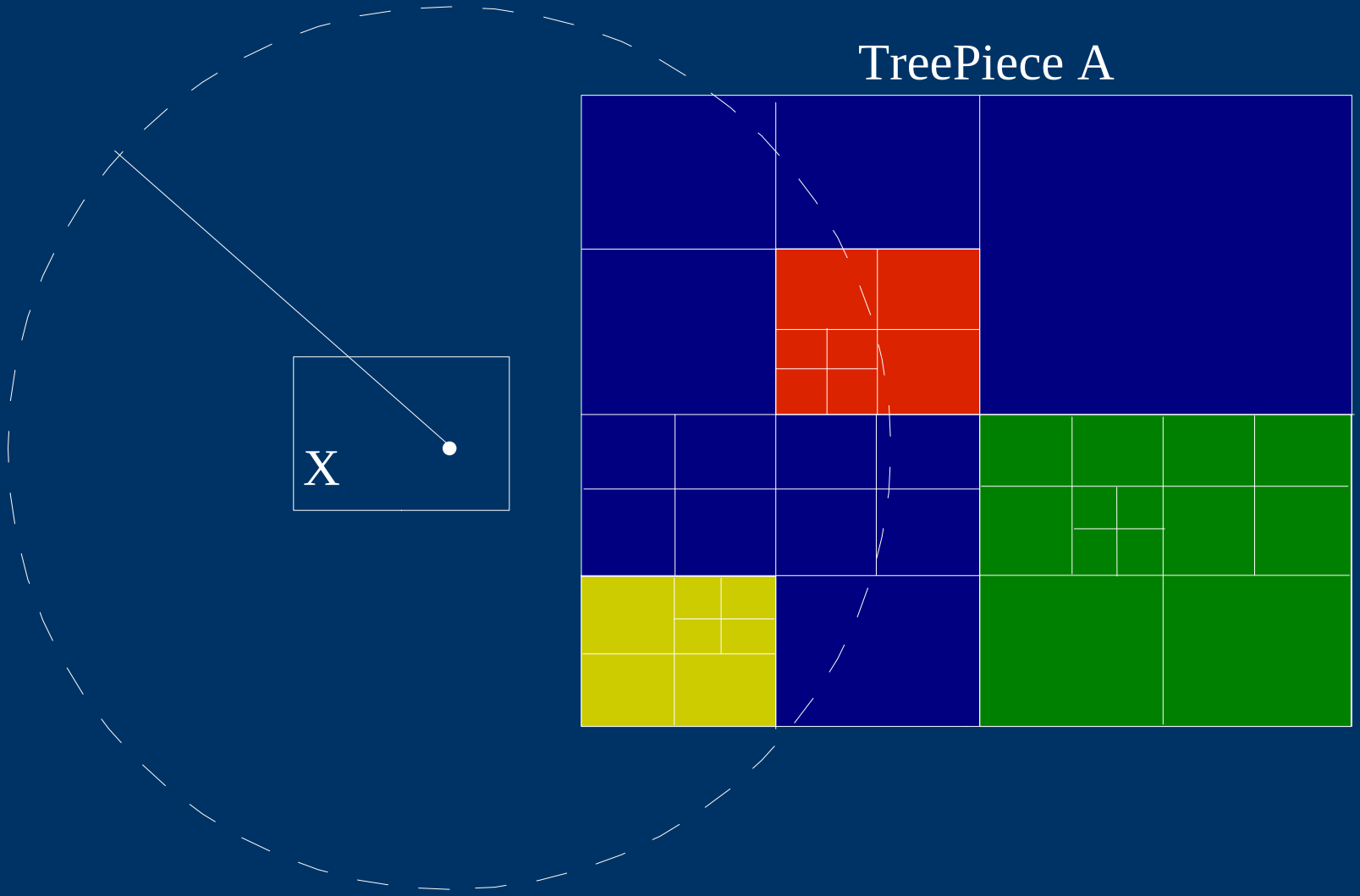
ChaNGa scalability (old results)

results on BlueGene/L

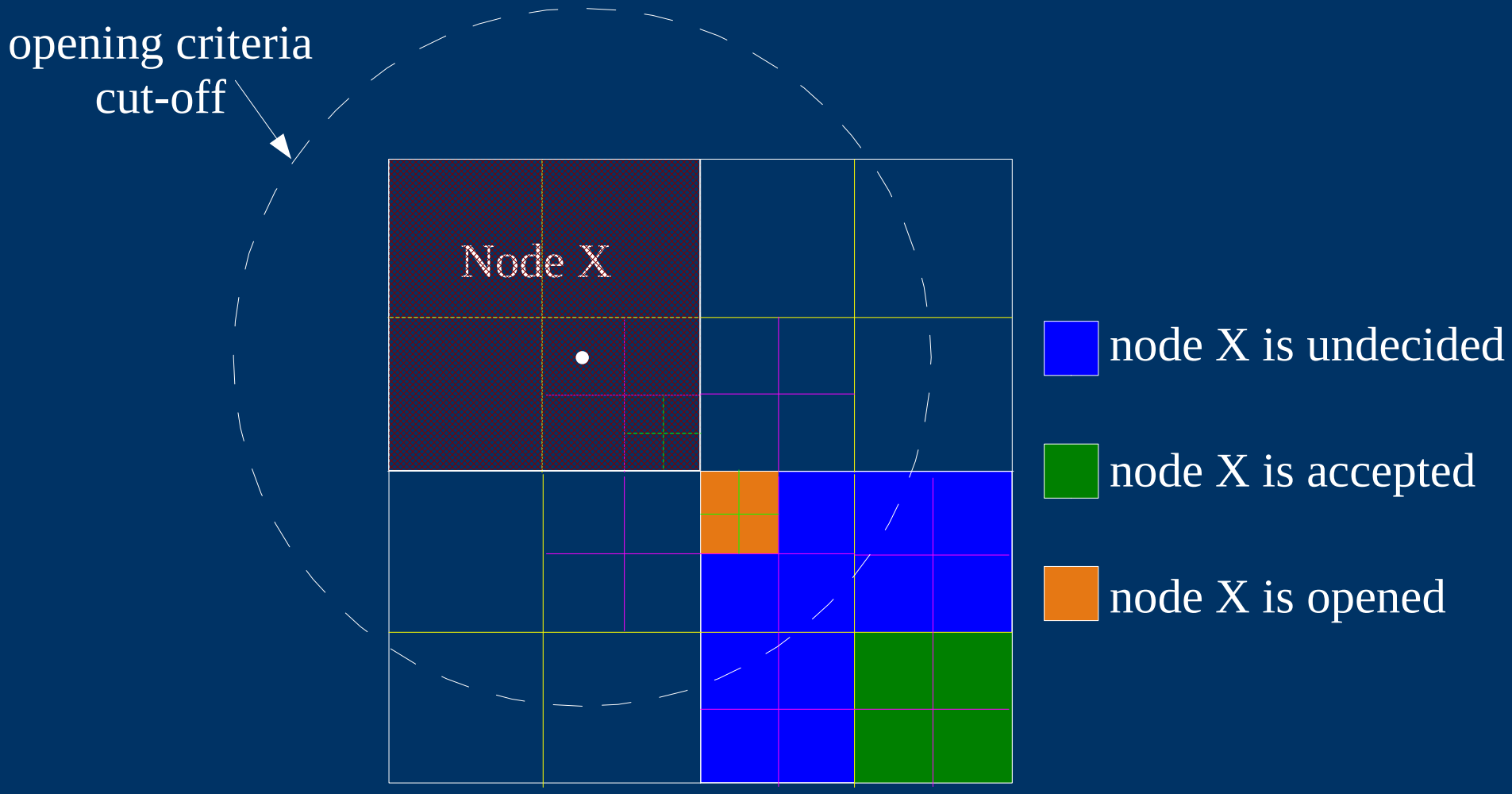


flat: perfect scaling diagonal: no scaling

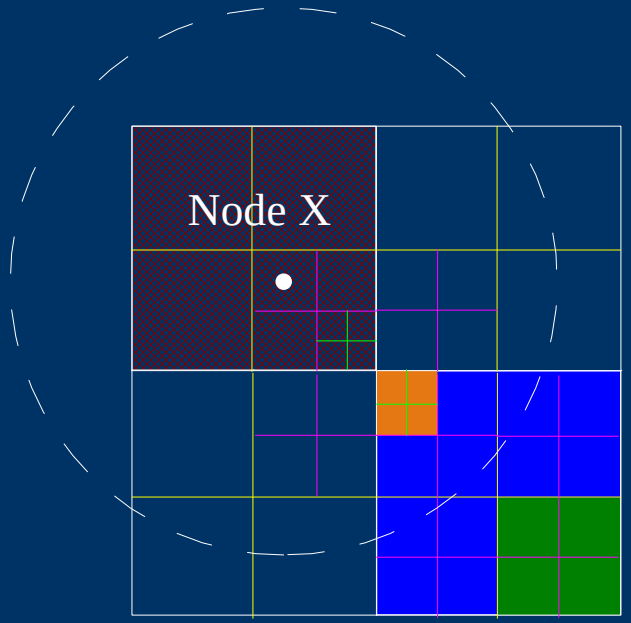
Interaction list

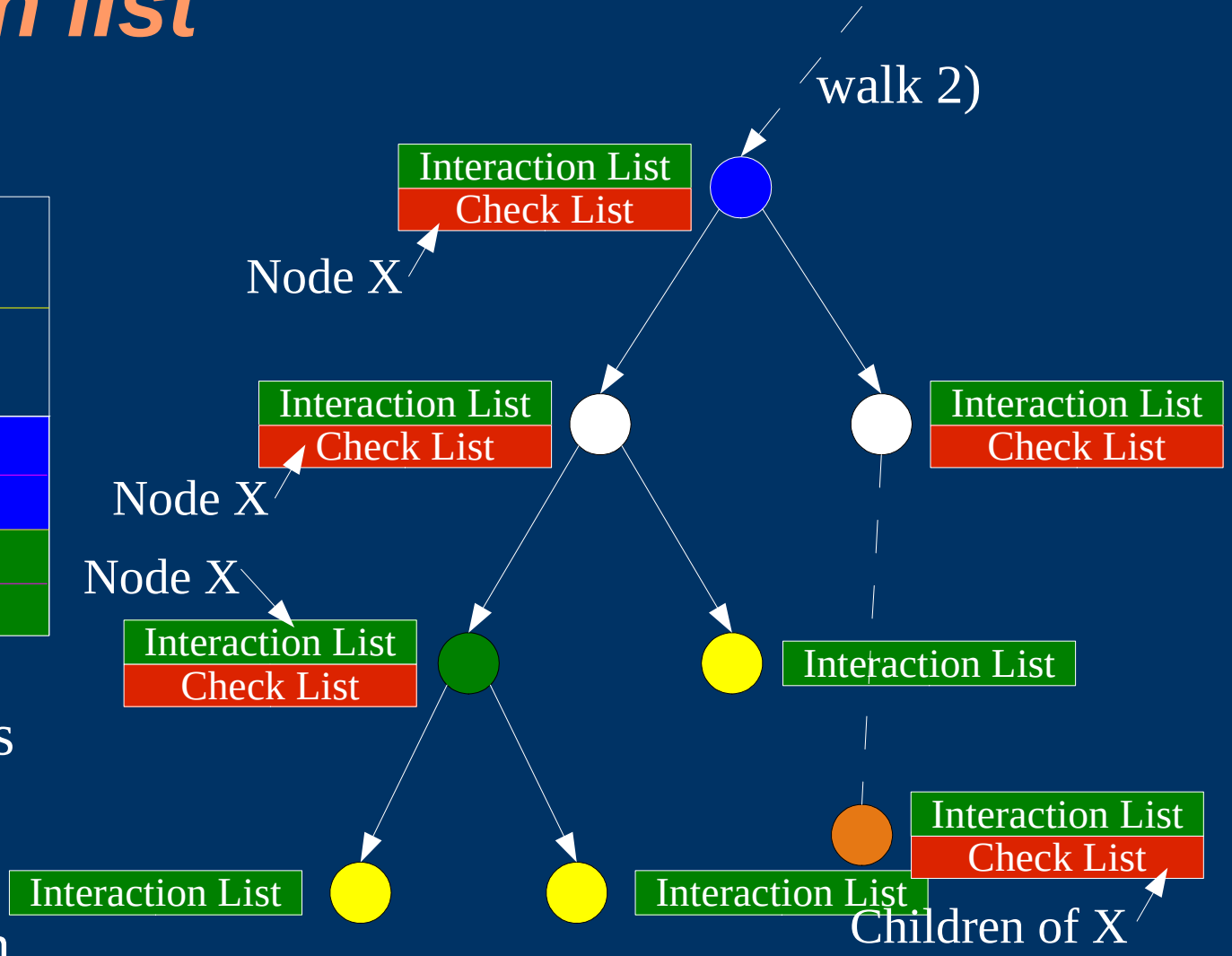


Interaction lists



Interaction list





Double simultaneous walk in two copies of the tree:

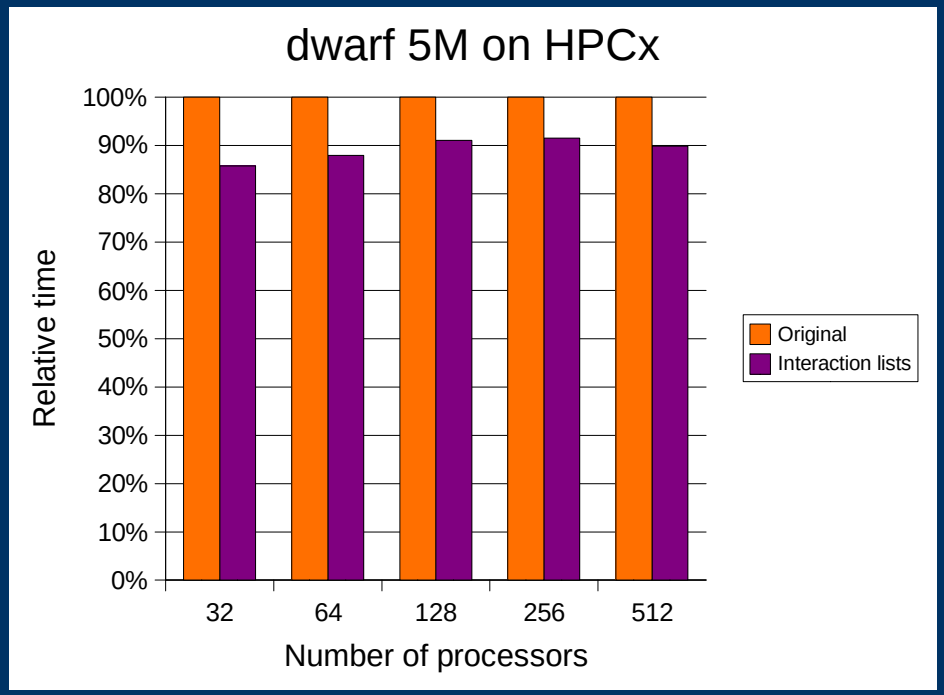
1) force computation

2) exploit this observation

Interaction list: results

Number of checks for opening criteria, in millions

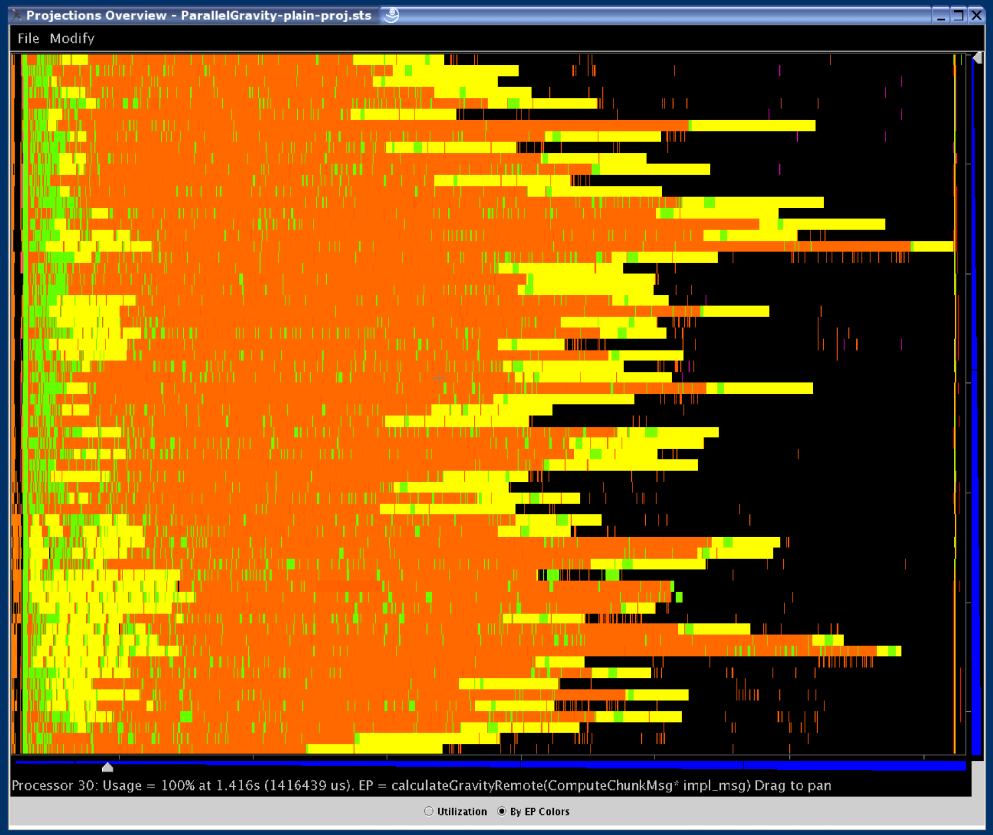
	lambs 1M	dwarf 5M
Original code	120	1,108
Interaction list	66	440

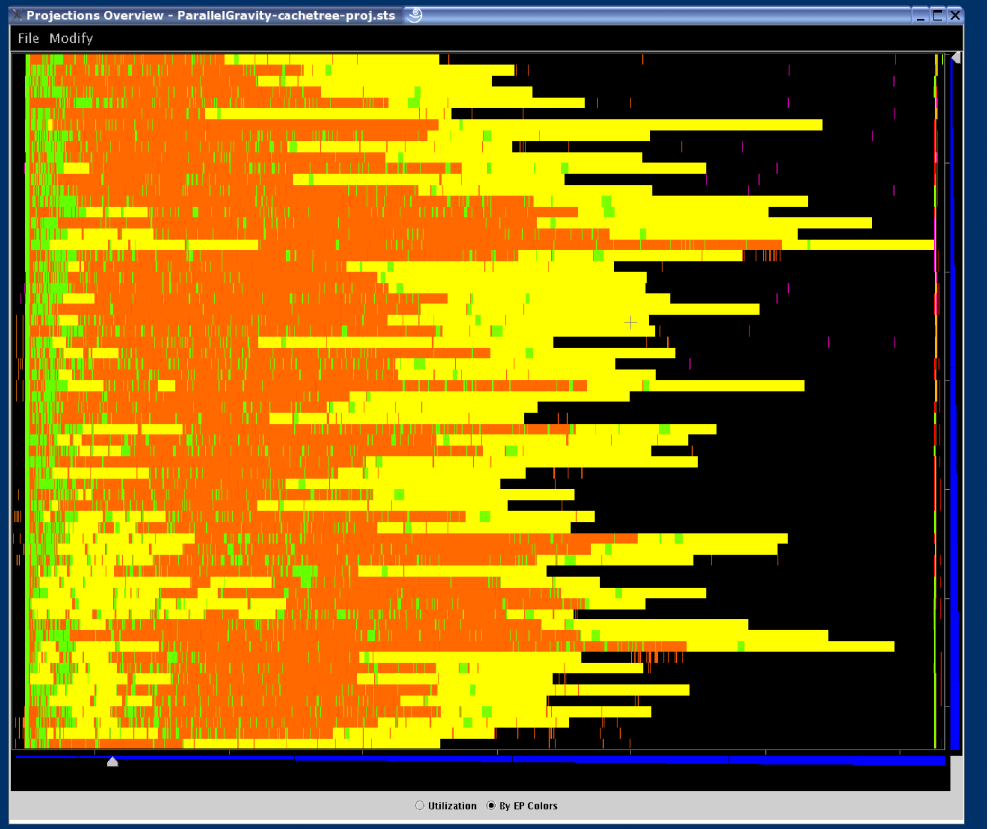


- 10% average performance improvement

Tree-in-cache

lambs 300K subset on 64 processors of Tungsten

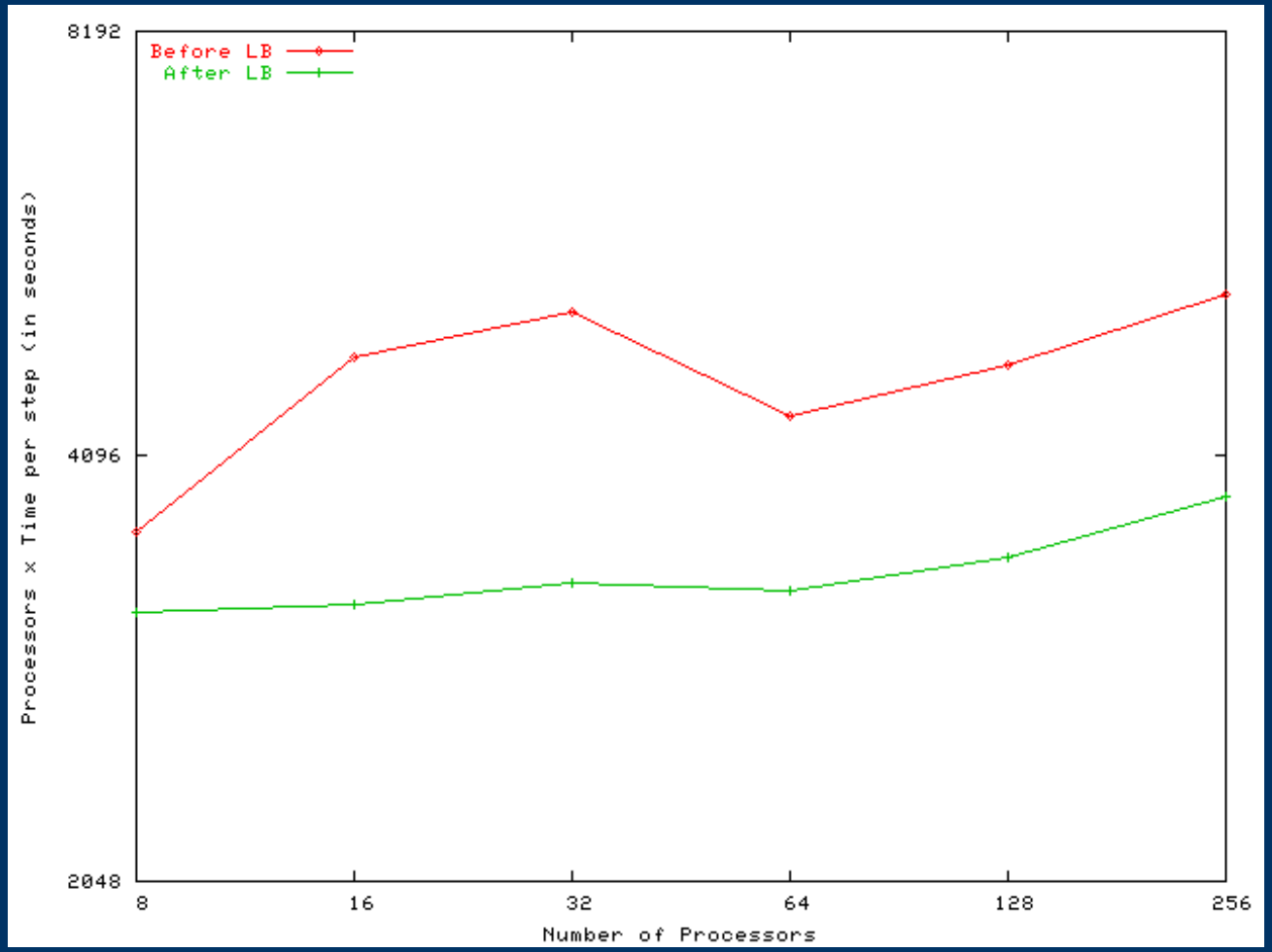




16%	Local computation	42%
84%	Global computation	58%

Load balancer

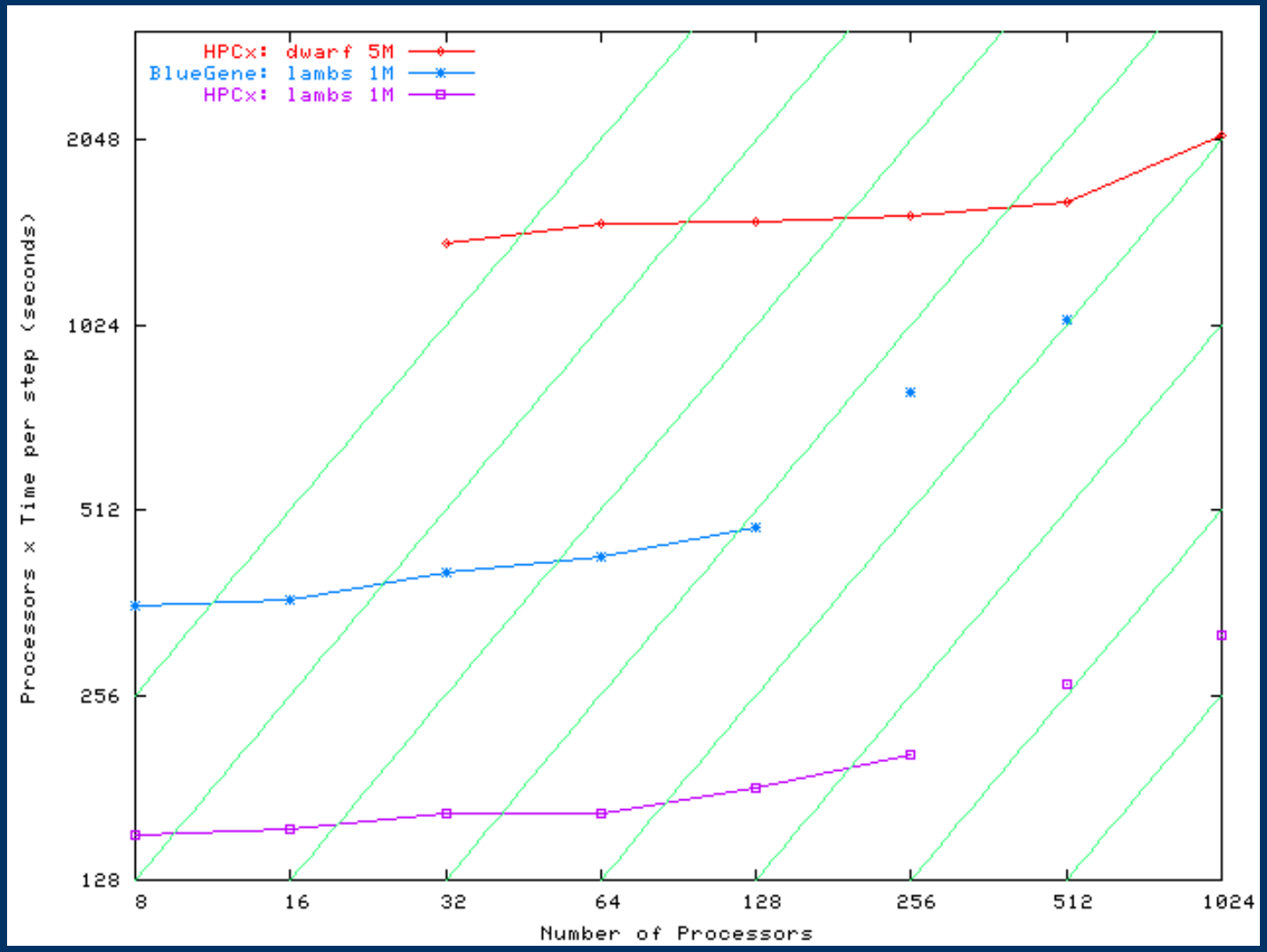
dwarf 5M dataset on BlueGene/L



improvement between 15% and 35%

flat lines good
raising lines bad

ChaNGa scalability



flat: perfect scaling diagonal: no scaling