A Computation- and Communication-Optimal Parallel Direct 3-body Algorithm

Penporn Koanantakool and Katherine Yelick

{penpornk, yelick}@cs.berkeley.edu

Computer Science Division, University of California, Berkeley
Lawrence Berkeley National Laboratory

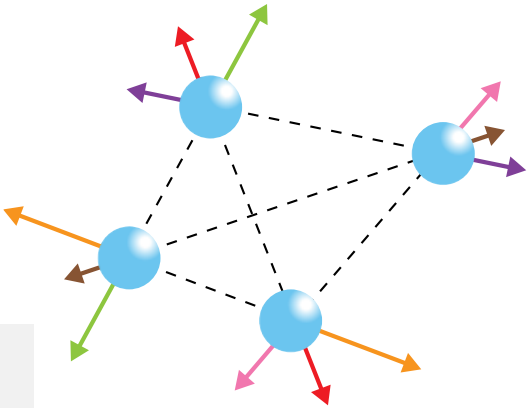
November 18, 2014

Outline

- **Introduction**
- Related Work
- The All-triplets Algorithm
- Performance Results
- Extension for Cutoff
- Conclusions

The N-Body Problem

- For T timesteps, update positions of each of N particles according to **pairwise** forces from all others.



```
for timestep in T:
    for i in N:
        for j in N:
            interact(particles[i], particles[j])
    for i in N:
        move(particles[i])
```

Bottleneck
$O(n^2)$

- Applications:
  - Galaxy simulations, Molecular Dynamics, etc…

Many-Body Interactions

- Full force equation

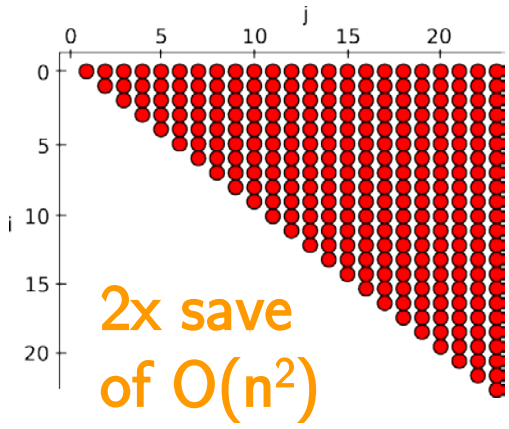
$$f_i = -\nabla(\phi_1(i) + \sum_j \phi_2(i,j) + \sum_{j,k} \phi_3(i,j,k) + \dots)$$

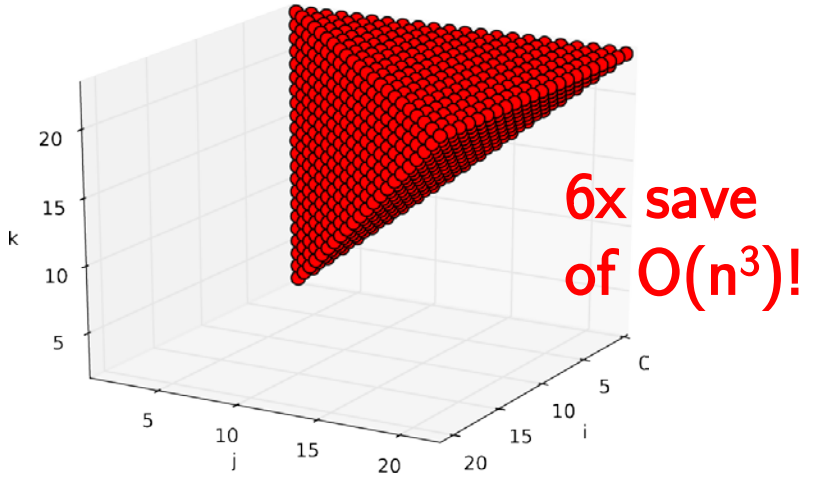
  - ϕ_k : k-body potential energy function
  - 2-body has largest contribution, then 3, 4, …
- Some phenomena are uncaptured by 2-body
  - Water simulation: 80%→98% accuracy*
  - Modeling metals and ceramics
  - Protein folding, etc.

*[O'Suilleabhain 2013]

Challenge: Symmetry & Load Balance

- Force symmetry ($f_{ij} = -f_{ji}$) saves computation
- 2-body force matrix vs 3-body force cube

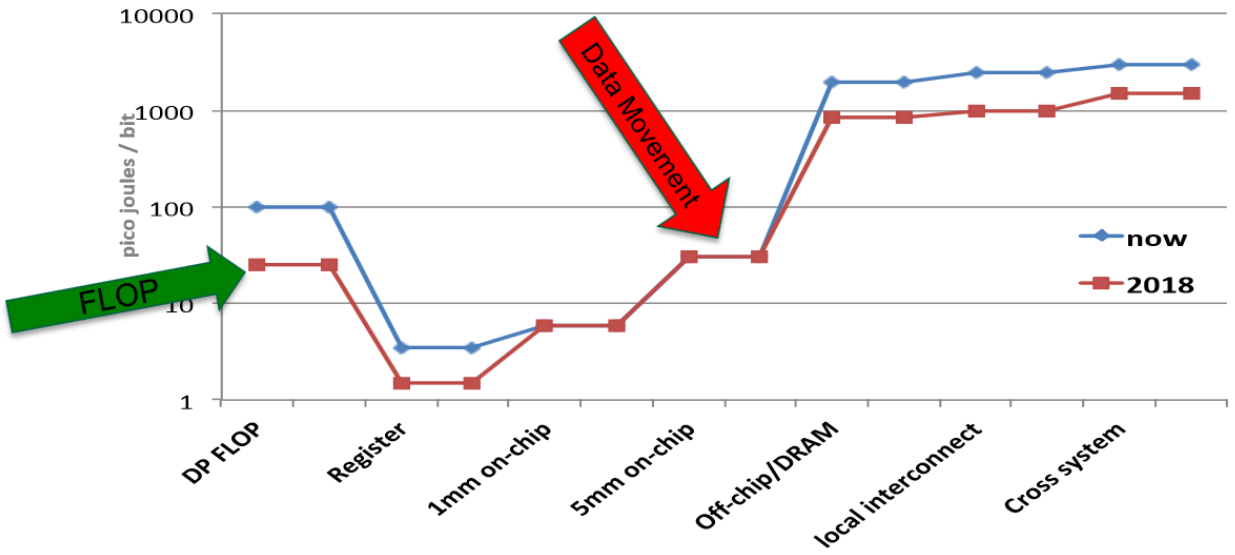




- How to divide work equally?

Challenge: Communication

- Costs = computation + communication
- FLOPS are cheaper than moving data remotely
- In 2018, even local data movement will be more expensive than a FLOP!



Courtesy of: John Shalf and John Urbanic

Communication Lower Bounds

- Per-processor costs along critical path
  - Latency $S = \#\text{messages}$
  - Bandwidth $W = \#\text{words}$
- [Ballard et al. 2014]
  - Z FLOPs per processor
  - M words in memory, F max operations with M words

$$S = \Omega\left(\frac{Z}{F}\right), \qquad W = \Omega\left(\frac{Z \cdot M}{F}\right)$$

- 3-Body
  - $\mathbf{n}$ particles
  - $\mathbf{p}$ processors
  - $\mathbf{c}$ copies ($M = cn/p$)

$$\boxed{S = \Omega\left(\frac{p^2}{c^3}\right), \qquad W = \Omega\left(\frac{np}{c^2}\right)}$$

$$Z = O(n^3/p) \quad F = O(M^3)$$

$$* \; c \leq p^{2/3}$$

Performance Metrics

- "Computation Optimal"*
  - Exploits symmetries.
- Load Balance
  - Processors have equal amount of work.
- Communication Optimal
  - Sends minimum messages and words according to the lower bounds

*In the context of this work.

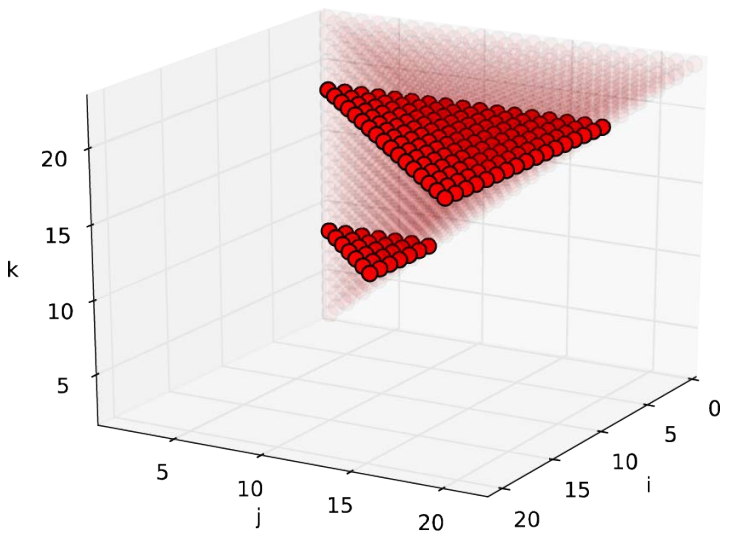
Outline

- Introduction
- **Related Work**
- The All-triplets Algorithm
- Performance Results
- Extension for Cutoff
- Conclusions

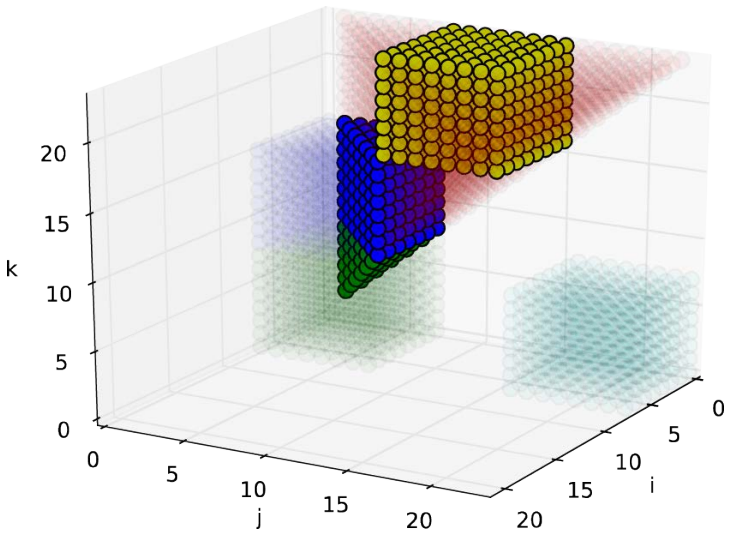
Li et al. 2006

Symmetry ✓
Load Balance ✘
Communication ✘

- Atom/Force decomposition
- Load imbalance
  - 40-50% parallel efficiency on 35 procs



Plane Decomposition (Atom Decomposition)



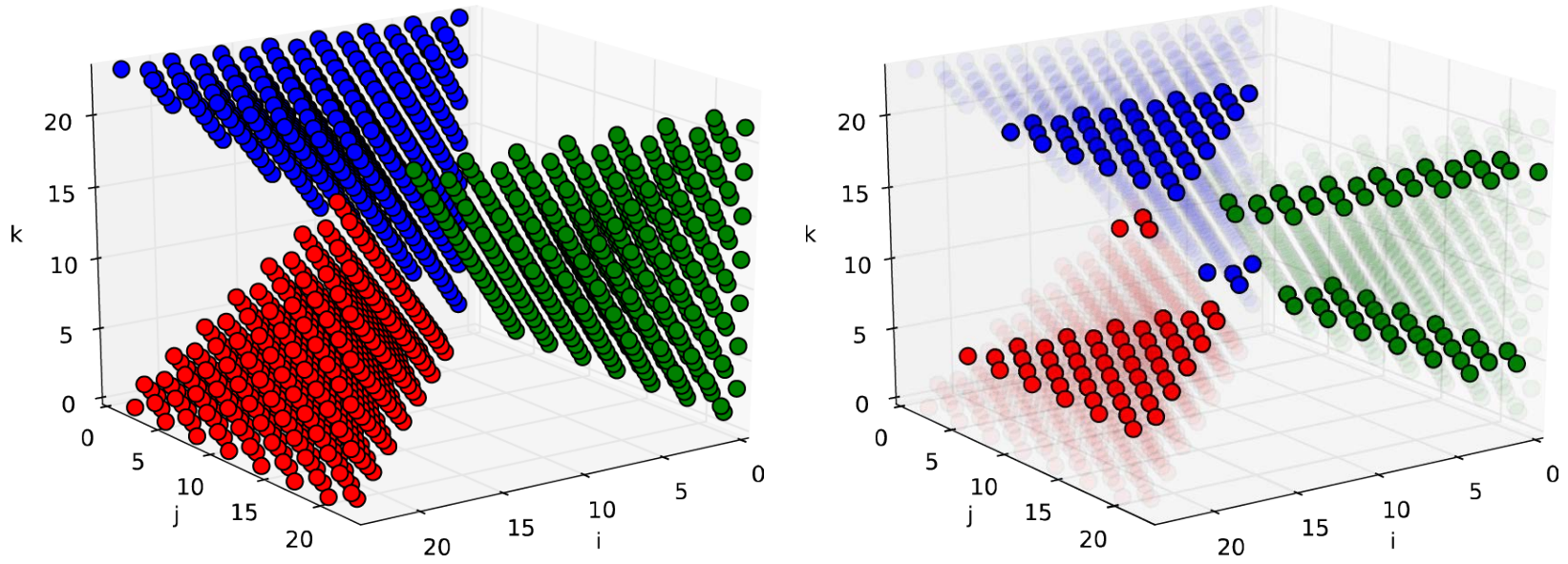
Force Decomposition

[Li et al. 2006][Li et al. 2008]

Slice Symmetric

Symmetry ✓
Load Balance ✓
Communication ?

- Each slice/plane has the same load
  - But also needs all particles

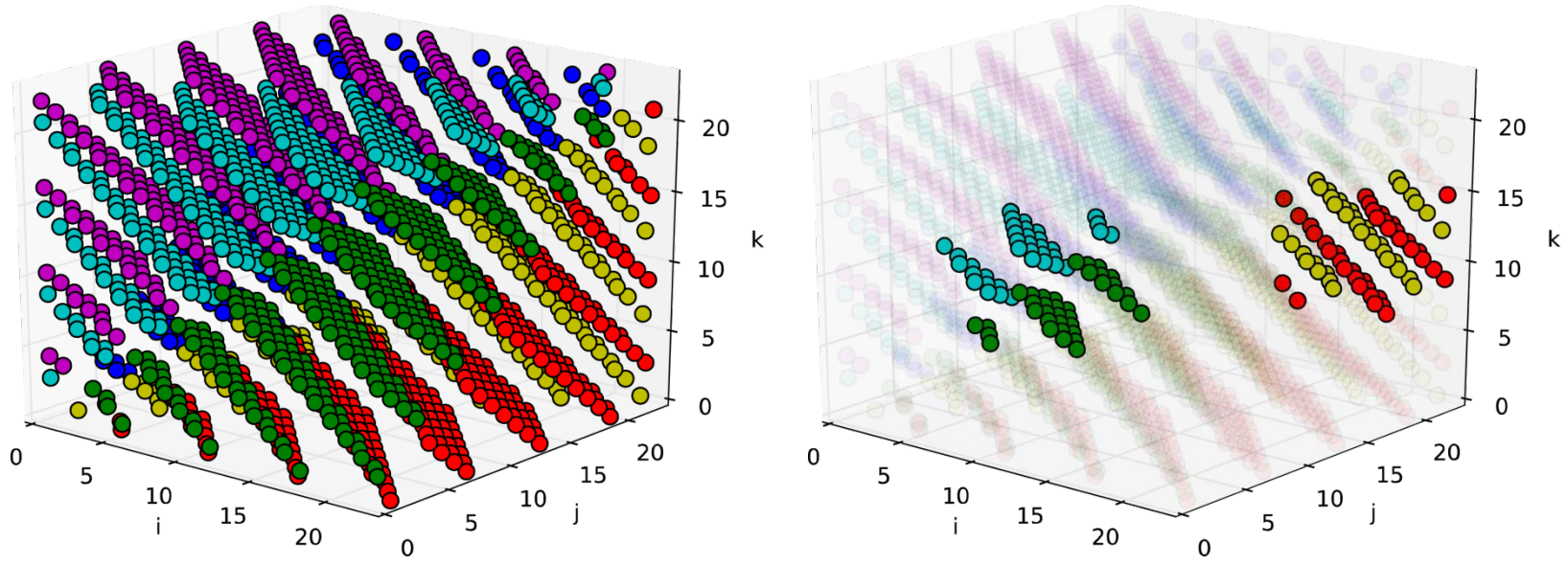


[Sumanth et al. 2007]

Volume Symmetric

Symmetry ✓
Load Balance ✓
Communication ?

- Each subcube/volume has the same load
- Volume is sparse



[Sumanth et al. 2007]

Summary

Algorithms	Symmetry	Load Balance	Communi-cation
Plane/Force Decomposition	✓	✗	✗
Slice Symmetric	✓	✓	?
Volume Symmetric	✓	✓	?

Outline

- Introduction
- Related Work
- **The All-triplets Algorithm**
- Performance Results
- Extension for Cutoff
- Conclusions

2-Nested For Loops

- n/p particles per processor
- Let S_i be the set of particles at processor i
- Interact all triplets of (S_i, S_j, S_k)

```
foreach i          ← Do it in parallel at processor i
    foreach j ≥ i
        foreach k ≥ j
            interact(S_i, S_j, S_k)
```

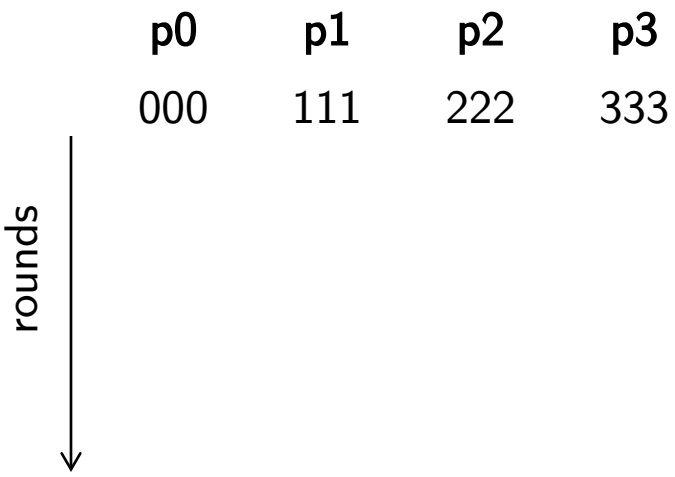
← Code for each processor

2-Nested For Loops

- n/p particles per processor
- Let S_i be the set of particles at processor i
- Interact all triplets of (S_i, S_j, S_k)

```
foreach j in i, ..., p-1, ..., i-1
    foreach k in j, ..., p-1, ..., i-1
        interact(S_i, S_j, S_k)
```

– At most p^2 times

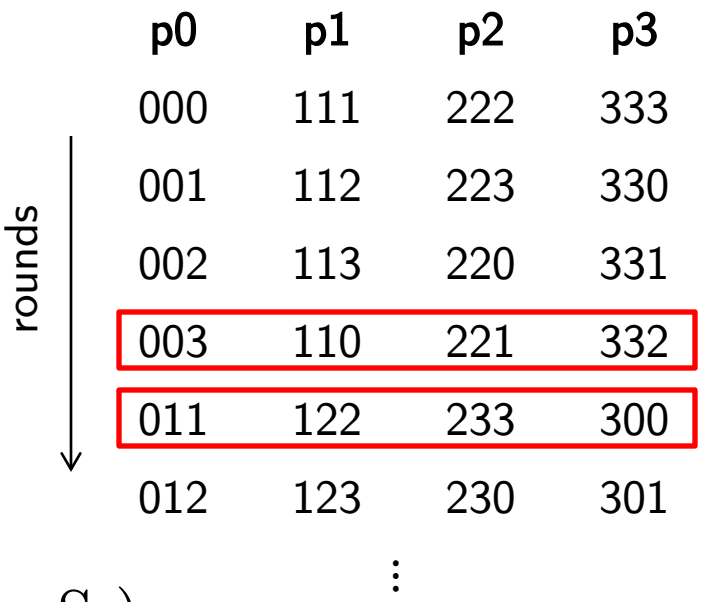


2-Nested For Loops

- n/p particles per processor
- Let S_i be the set of particles at processor i
- Interact all triplets of (S_i, S_j, S_k)

```
foreach j in i, ..., p-1, ..., i-1
    foreach k in j, ..., p-1, ..., i-1
        interact(S_i, S_j, S_k)
```

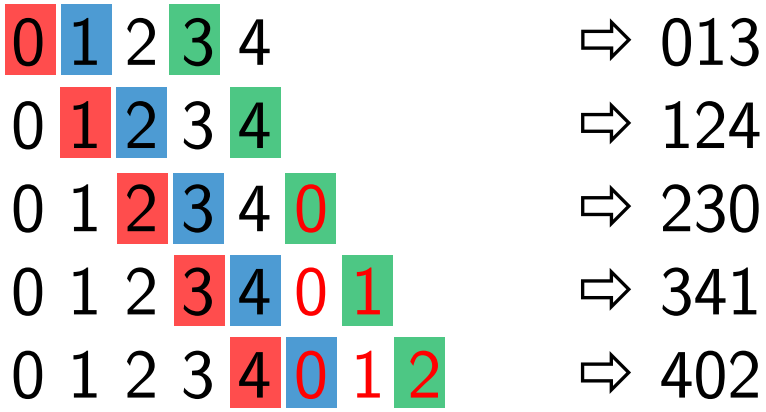
  - At most p^2 times



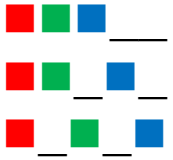
- How to avoid redundancy?
  - Want unique triplets of (S_i, S_j, S_k)

Box patterns

- (S_i, S_{i+x}, S_{i+y}) gives all different (S_i, S_j, S_k) triplets for all i from 0 to p-1
- Boxes illustrate indices i, j, k $(x=1, y=2)$



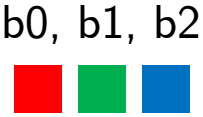
- Generating $\binom{p}{3}$ unique triplets turns into
  - Generating $\binom{p}{3}/p$ unique box patterns:



The All-triplets Algorithm

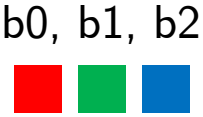
- Each processor has 3 buffers, b_0, b_1, b_2 to store particle subsets S_i, S_j, and S_k
- Creates new triplets by shifting buffer
- Shifts one buffer at a time
  - for $\mathbf{p}$ rounds, right-shift b_2
  - for $\mathbf{p\text{-}3}$ rounds, right-shift b_0
  - for $\mathbf{p\text{-}6}$ rounds, right-shift b_1
  - ...
  - for $\mathbf{p\%3}$ rounds, right-shift b_x

Example: p=8 processors



000 111 222 333 444 555 666 777	8 rounds
007 110 221 332 443 554 665 776	
006 117 220 331 442 553 664 775	
005 116 227 330 441 552 663 774	
004 115 226 337 440 551 662 773	
003 114 225 336 447 550 661 772	
002 113 224 335 446 557 660 771	
001 112 223 334 445 556 667 770	
701 012 123 234 345 456 567 670	5 rounds
601 712 023 134 245 356 467 570	
501 612 723 034 145 256 367 470	
401 512 623 734 045 156 267 370	
301 412 523 634 745 056 167 270	
371 402 513 624 735 046 157 260	2 rounds
361 472 503 614 725 036 147 250	

Example: p=8 processors



■-------	000	111	222	333	444	555	666	777
■------■	007	110	221	332	443	554	665	776
■-----■-	006	117	220	331	442	553	664	775
■----■--	005	116	227	330	441	552	663	774
■---■---	004	115	226	337	440	551	662	773
■--■----	003	11						772
■-■-----	002	11						771
■■------	001	11						770

8 rounds

Generates all unique triplets.

■■-----■	701	012	123	234	345	456	567	670
■■----■-	601	712	023	134	245	356	467	570
■■---■--	501	612	723	034	145	256	367	470
■■--■---	401	512	623	734	045	156	267	370
■■-■----	301	412	523	634	745	056	167	270

5 rounds

-■-■---■	371	402	513	624	735	046	157	260
-■-■--■-	361	472	503	614	725	036	147	250

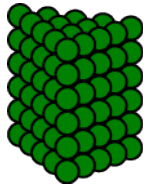
2 rounds

Load Balance?

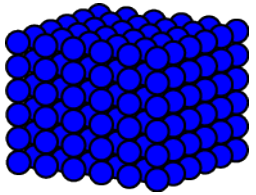
000	111	222	333	444	555	666	777
007	110	221	332	443	554	665	776
006	117	220	331	442	553	664	775
005	116	227	330	441	552	663	774
004	115	226	337	440	551	662	773
003	114	225	336	447	550	661	772
002	113	224	335	446	557	660	771
001	112	223	334	445	556	667	770
701	012	123	234	345	456	567	670
601	712	023	134	245	356	467	570
501	612	723	034	145	256	367	470
401	512	623	734	045	156	267	370
301	412	523	634	745	056	167	270
371	402	513	624	735	046	157	260
361	472	503	614	725	036	147	250

Tetrahedron

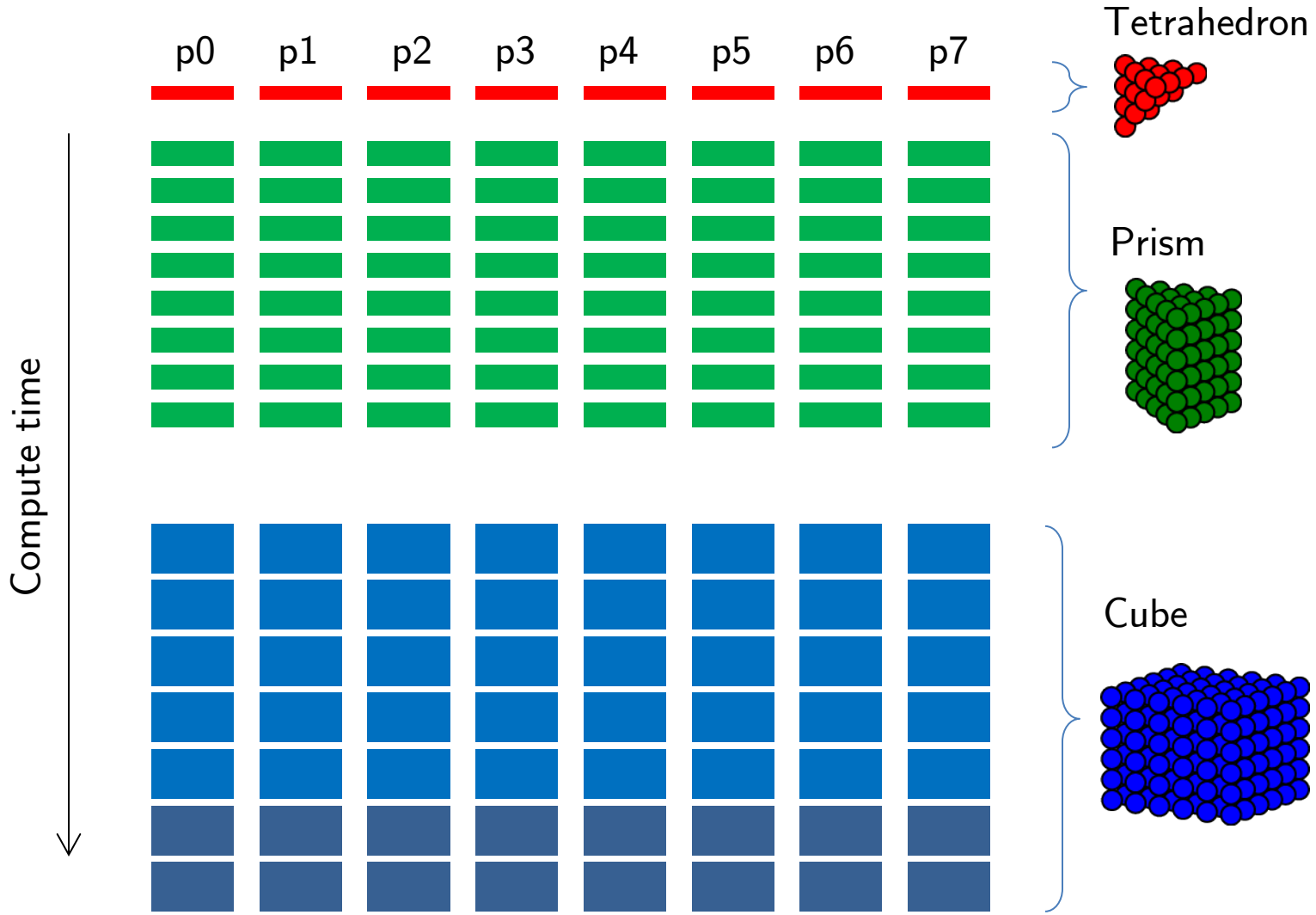
Prism



Cube

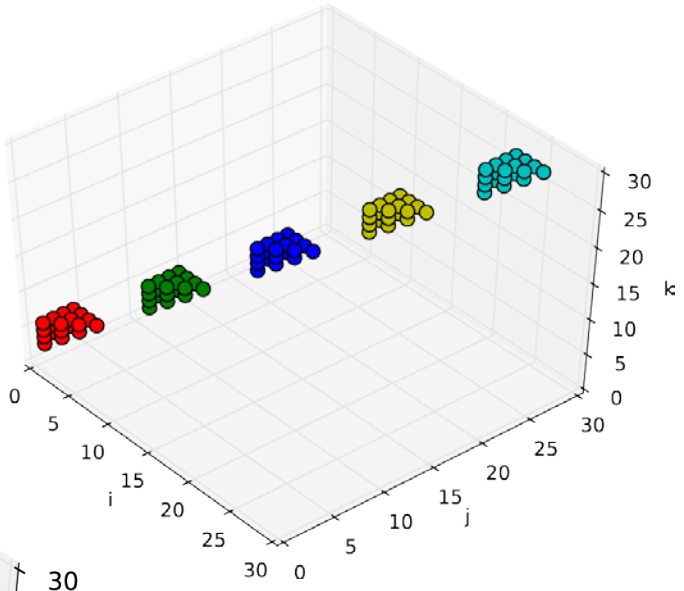


Perfect Load Balance

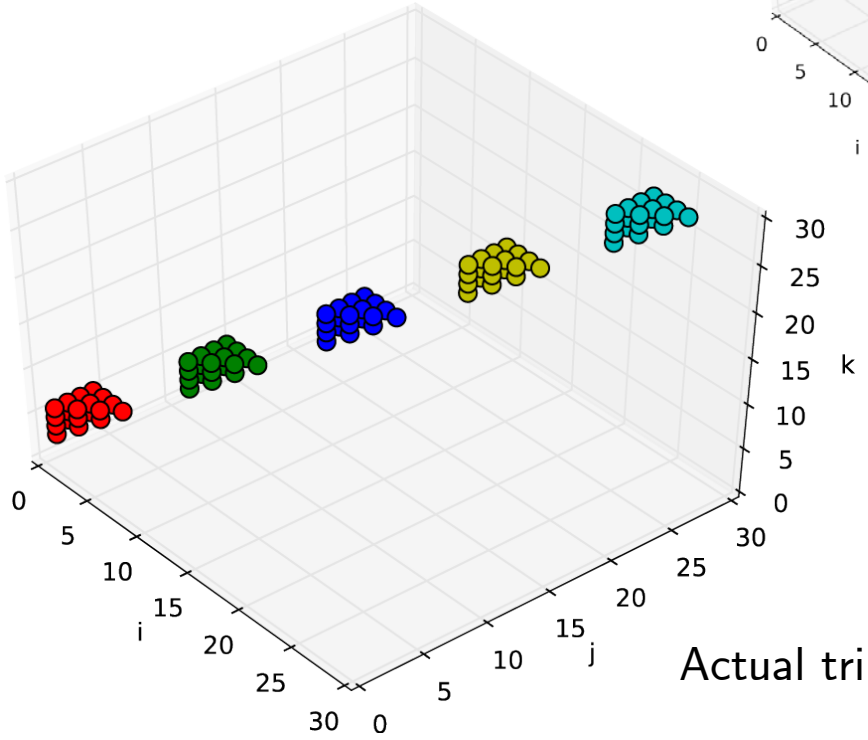


Geometric Meaning

- p=5, n=30
- 6 particles per processor
- 5x5 subcubes



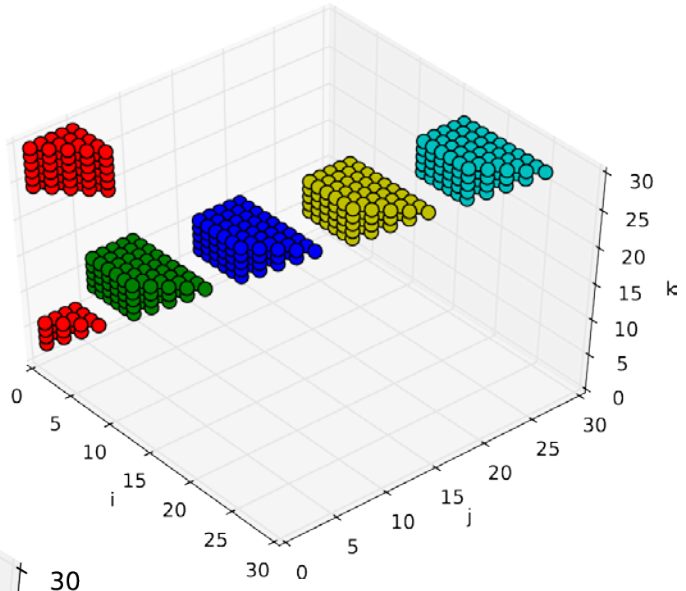
Equivalent triplets in the big tetrahedron



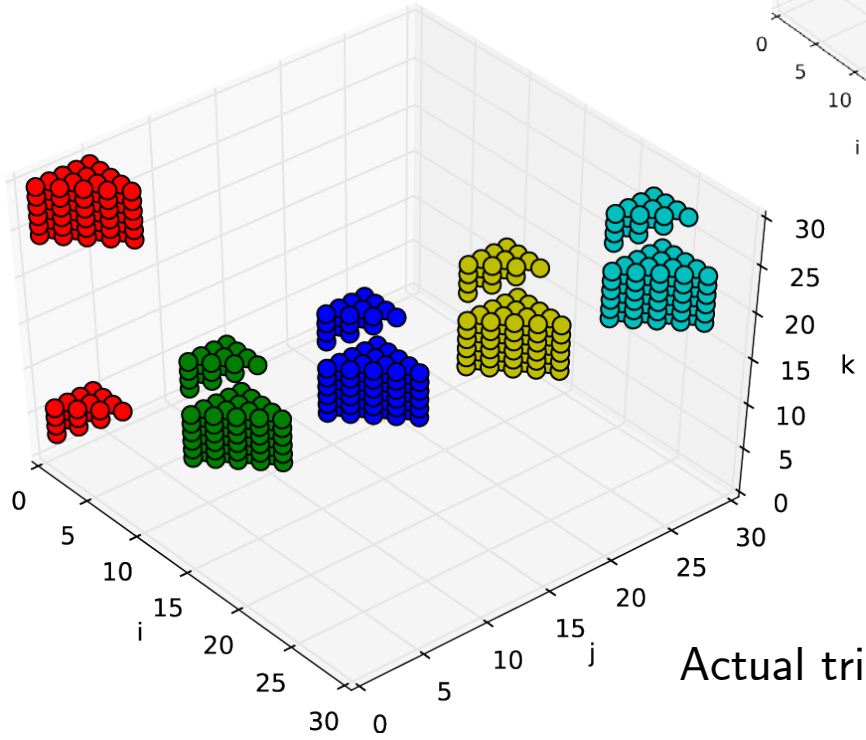
Actual triplets

Geometric Meaning

- p=5, n=30
- 6 particles per processor
- 5x5 subcubes



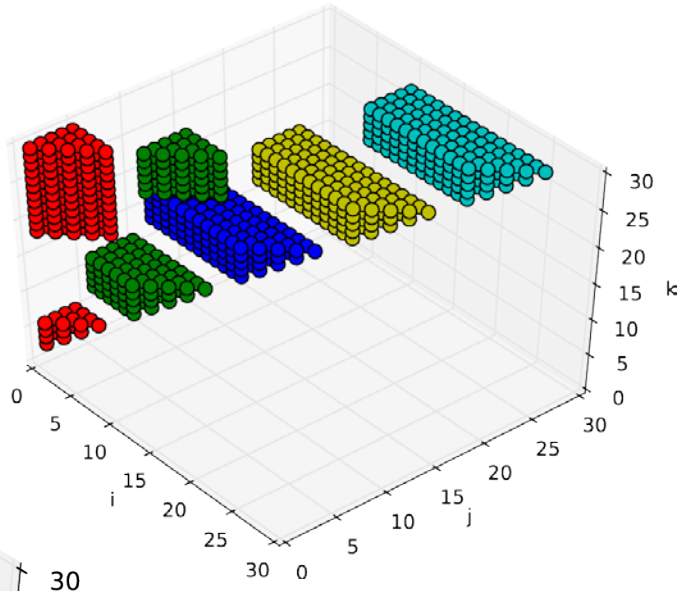
Equivalent triplets in the big tetrahedron



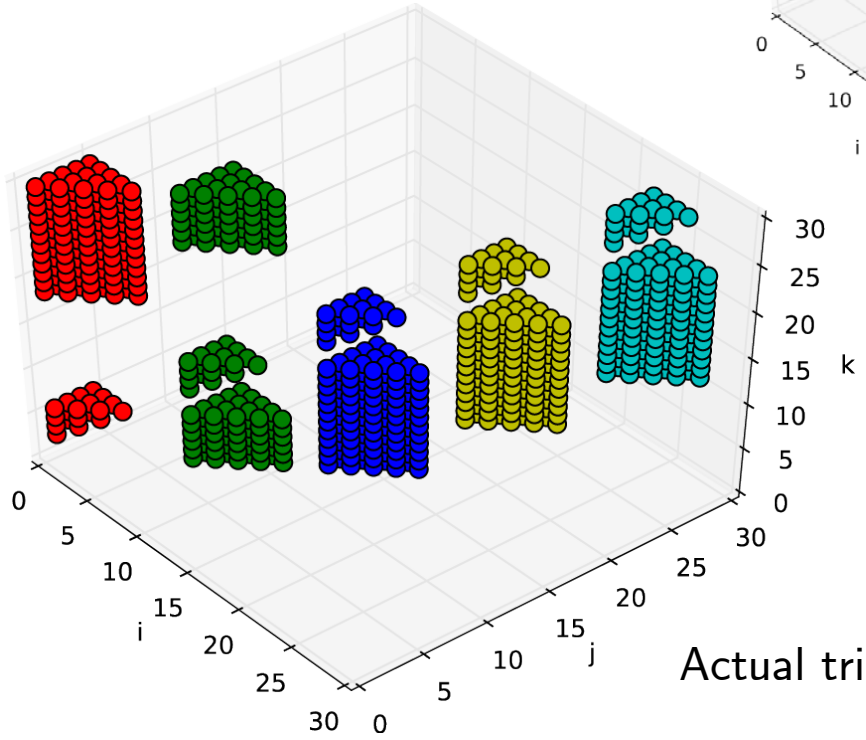
Actual triplets

Geometric Meaning

- p=5, n=30
- 6 particles per processor
- 5x5 subcubes



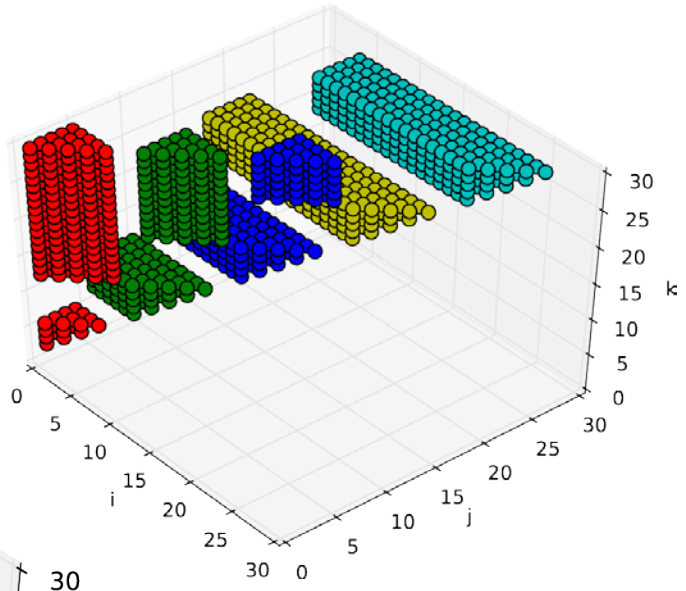
Equivalent triplets in the big tetrahedron



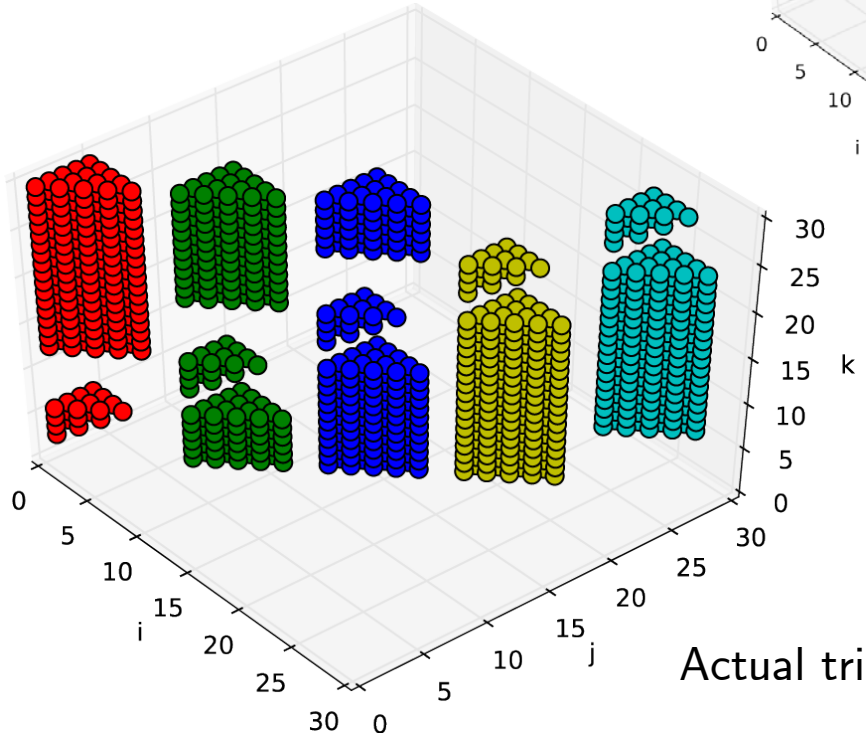
Actual triplets

Geometric Meaning

- p=5, n=30
- 6 particles per processor
- 5x5 subcubes



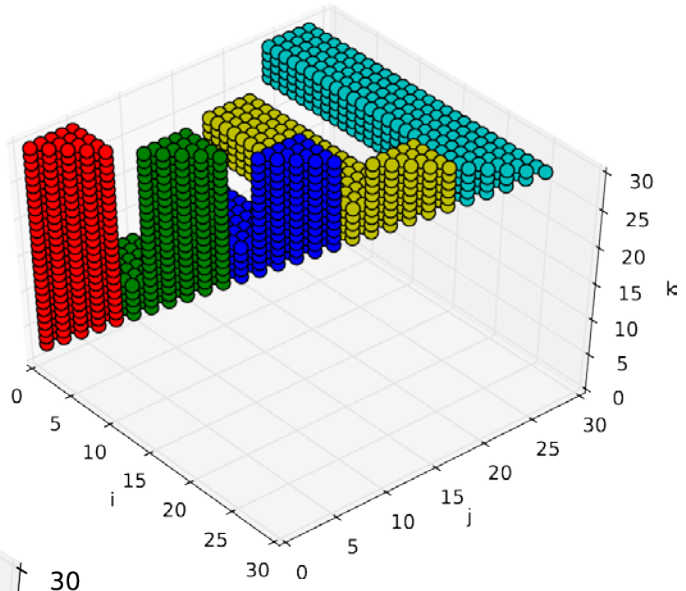
Equivalent triplets in the big tetrahedron



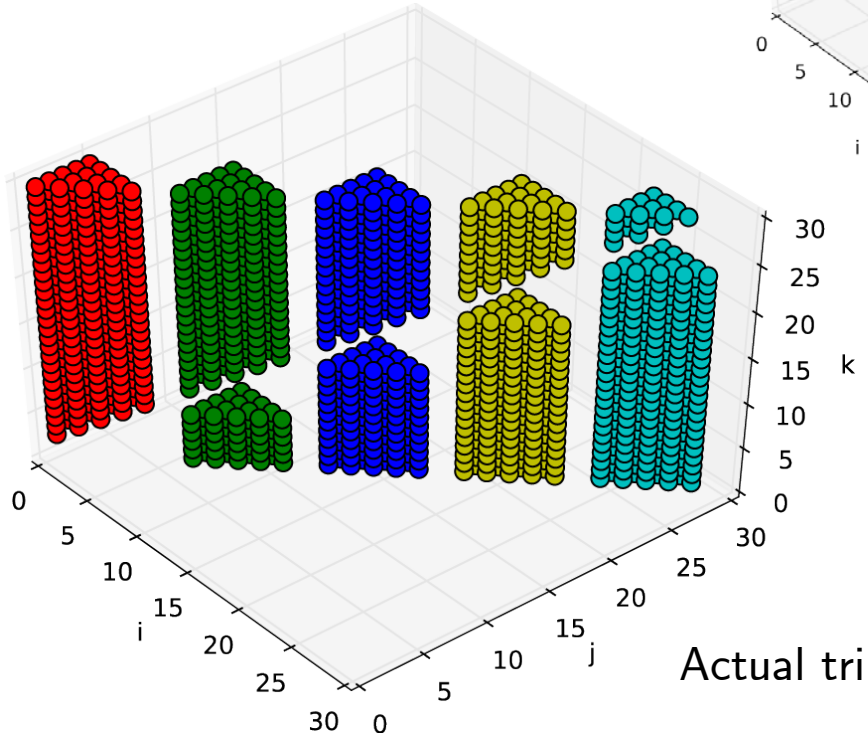
Actual triplets

Geometric Meaning

- p=5, n=30
- 6 particles per processor
- 5x5 subcubes



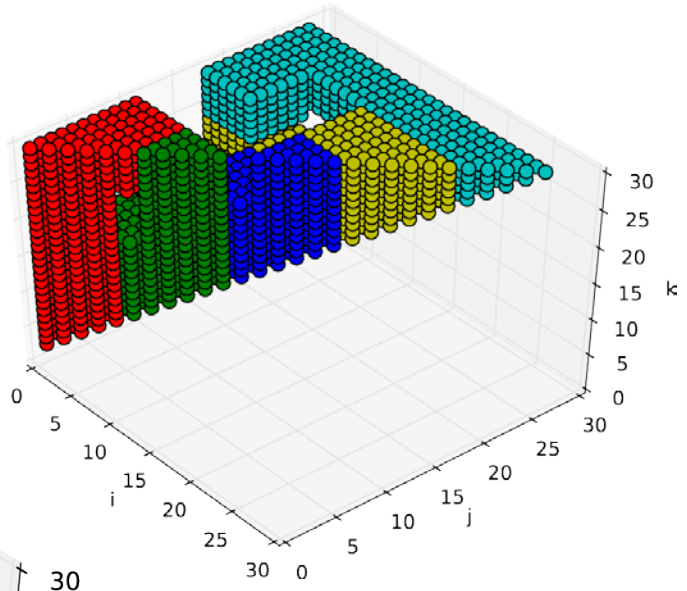
Equivalent triplets in the big tetrahedron



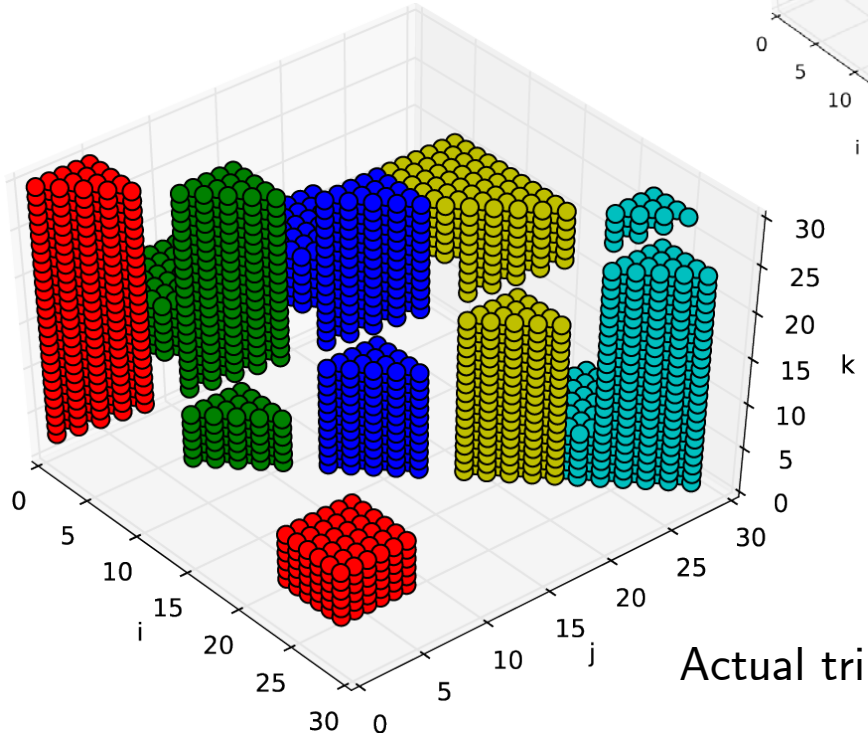
Actual triplets

Geometric Meaning

- p=5, n=30
- 6 particles per processor
- 5x5 subcubes



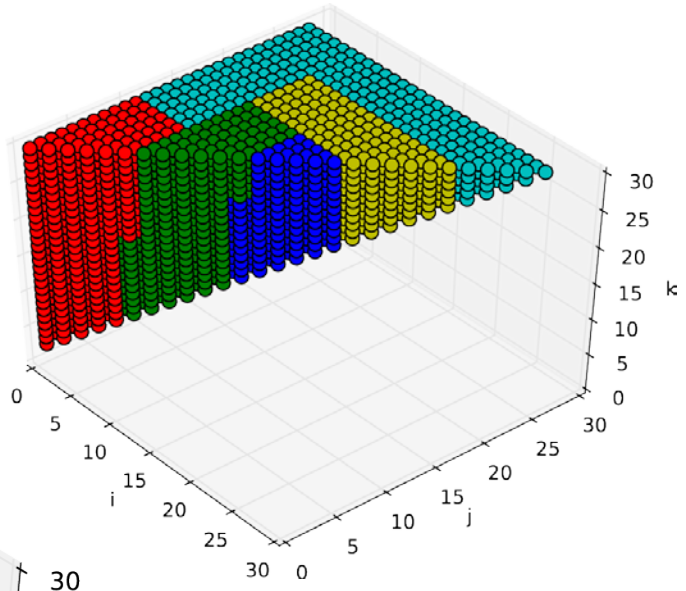
Equivalent triplets in the big tetrahedron



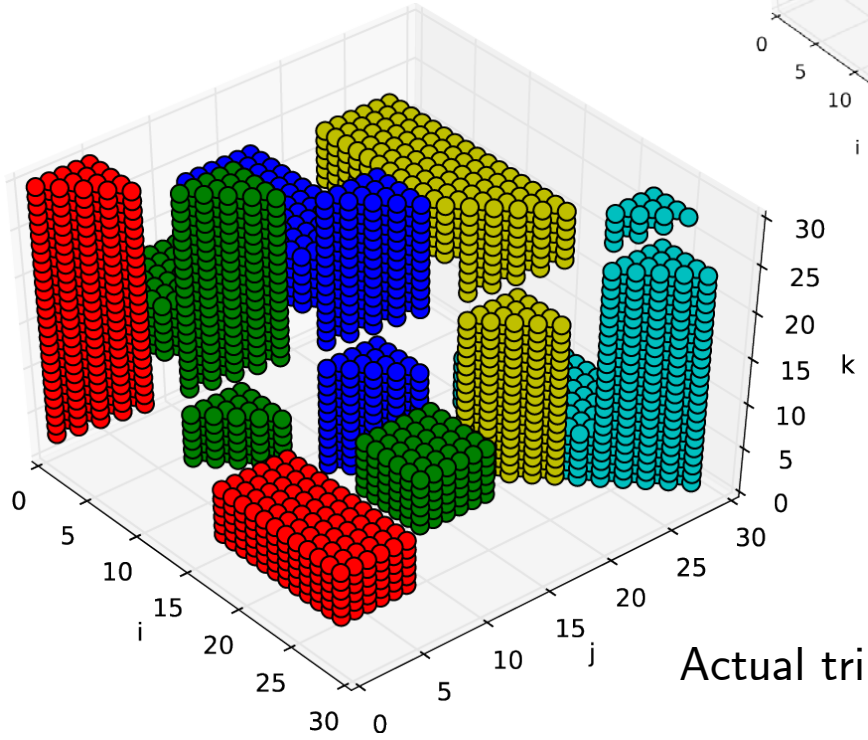
Actual triplets

Geometric Meaning

- p=5, n=30
- 6 particles per processor
- 5x5 subcubes



Equivalent triplets in the big tetrahedron



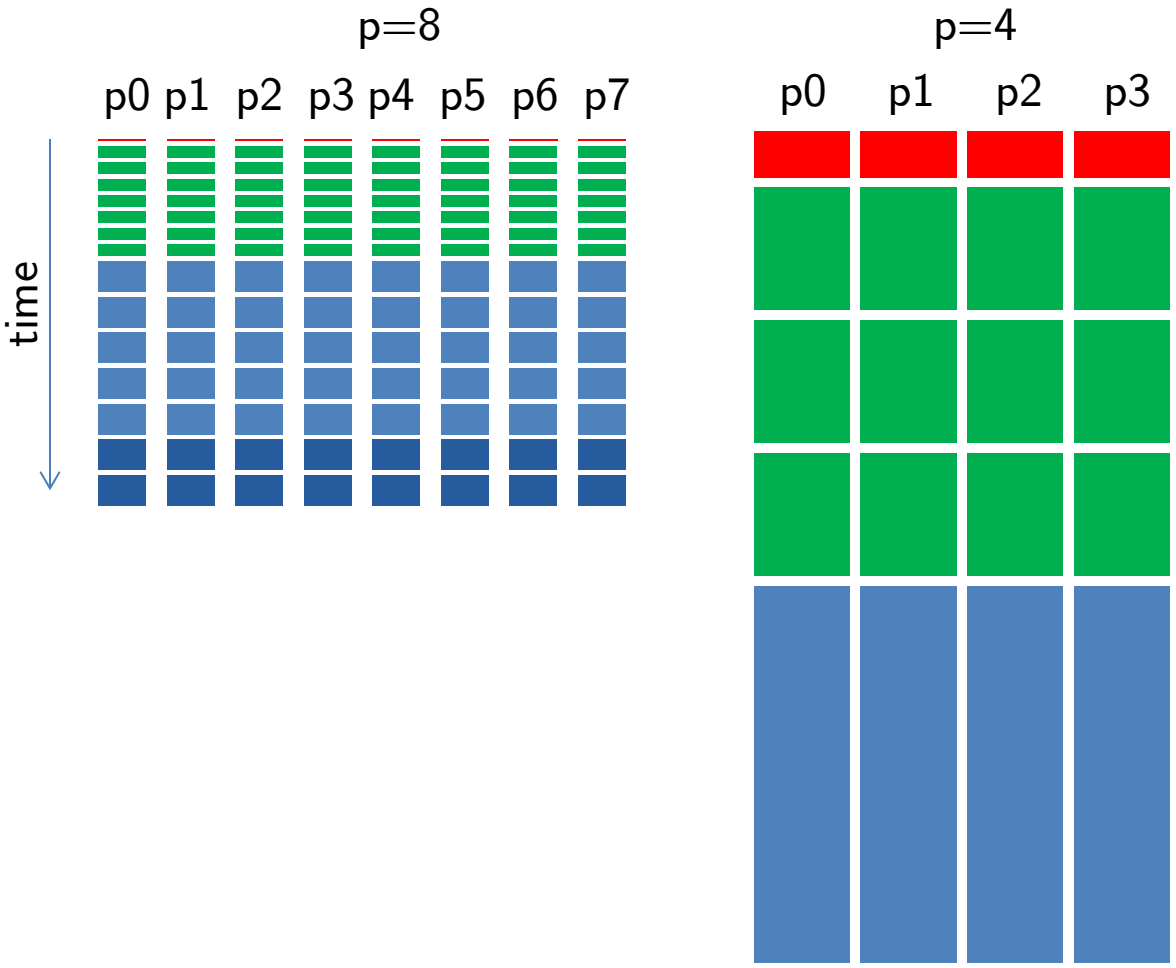
Actual triplets

Summary

Algorithms	Symmetry	Load Balance	Communi-cation
Plane/Force Decomposition	✓	✗	✗
Slice Symmetric	✓	✓	?
Volume Symmetric	✓	✓	?
3-Body	✓	✓	✓ (c=1)

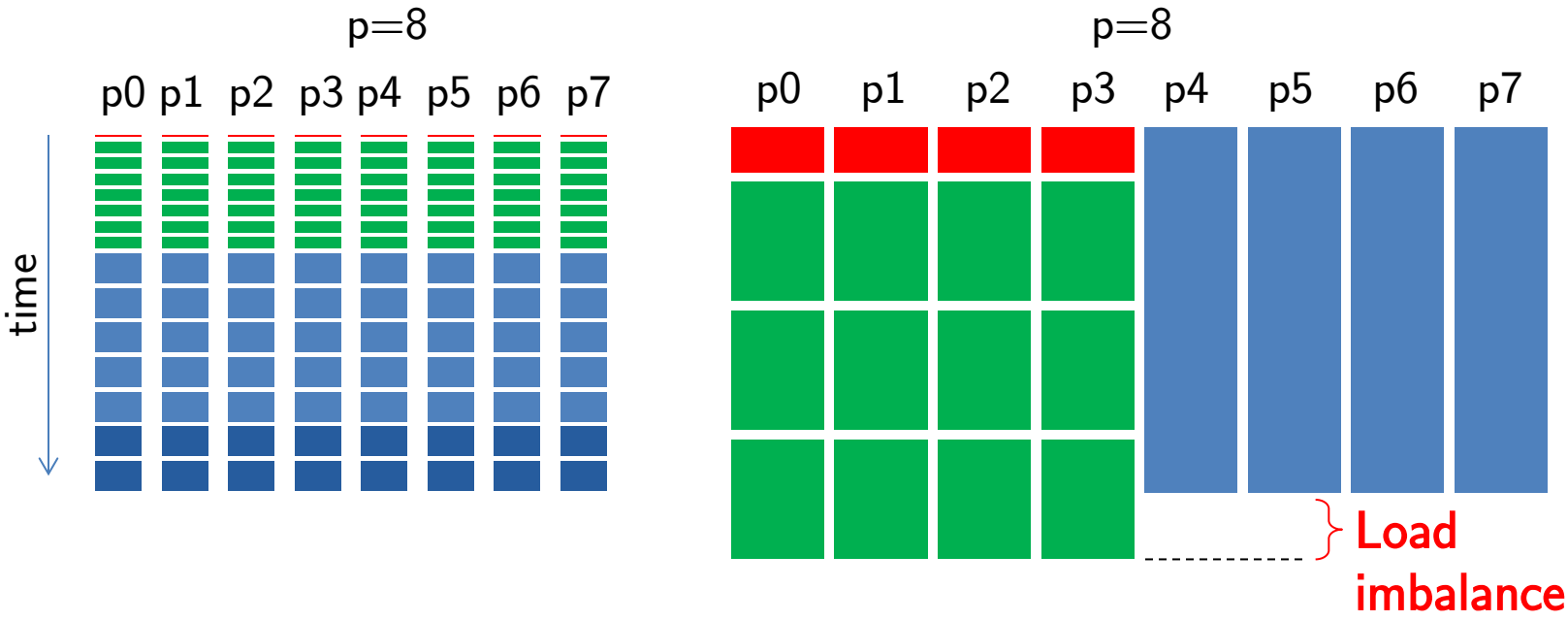
The Communication-Avoiding Algorithm

- Use $p/2$ processors, store 2x more particles



The Communication-Avoiding Algorithm

- Store 2x more particles, use p processors



The Communication-Avoiding Algorithm

- Store $\mathbf{cn/p}$ particles instead of $\mathbf{n/p}$
  - $\#\text{columns} = \mathbf{p/c}$
  - $\#\text{lines} = \mathbf{O(p^2/c^2)}$ instead of $\mathbf{O(p^2)}$
  - $\mathbf{c}$ processors do $\mathbf{c}$ concurrent lines at a time
- $\#\text{messages} = \mathbf{O(p^2/c^3)}$
  - Saves a factor of $\mathbf{c^3}$
- $\#\text{words} = \mathbf{O(np/c^2)}$
  - Saves a factor of $\mathbf{c^2}$
- **Introduces load imbalance**
  - **But it is bounded and can be predicted offline**
  - **We can pick the most appropriate $\mathbf{c}$**

Summary

Algorithms	Symmetry	Load Balance	Communi-cation
Plane/Force Decomposition	✓	✗	✗
Slice Symmetric	✓	✓	?
Volume Symmetric	✓	✓	?
3-Body	✓	✓	✓ (c=1)
CA 3-Body	✓	✗	✓

Outline

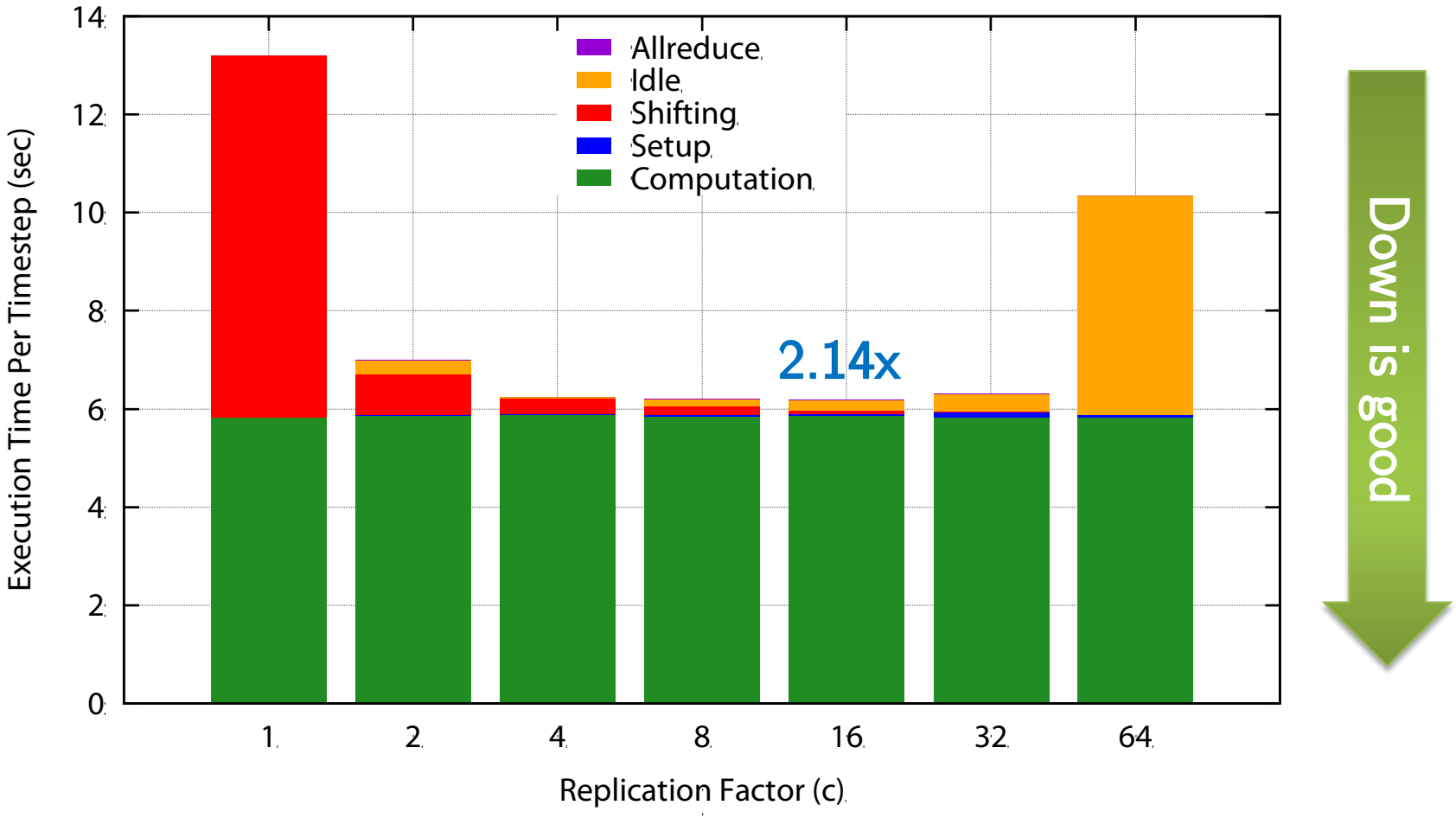
- Introduction
- Related Work
- The All-triplets Algorithm
- **Performance Results**
- Extension for Cutoff
- Conclusions

Test Environment

- Program
  - Axilrod-Teller potential
  - 3D space. 80B particles. Double-precision.
  - C MPI and C MPI+OpenMP
- Edison at NERSC
  - 2.57PFLOPS Cray XC30
  - 12 cores, 2 sockets each
  - Cray Aries, Dragonfly topology
- Mira at ALCF
  - 10PFLOPS Blue Gene/Q
  - 16 cores, 4 hardware threads each
  - 5D torus network, topology-aware task mapping

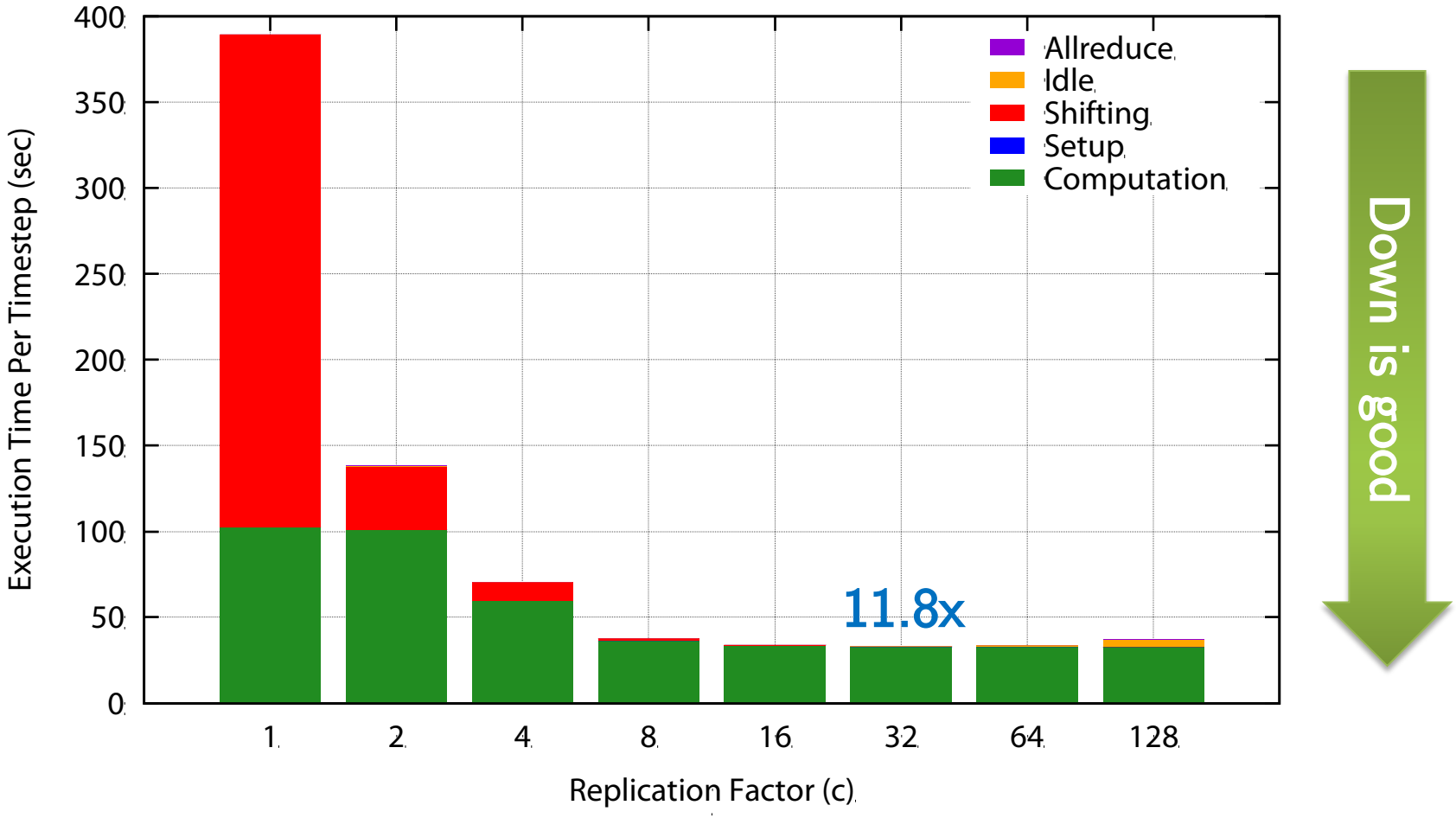
Computation Intensive

- Cray XC30, 1.5k cores, 6k particles



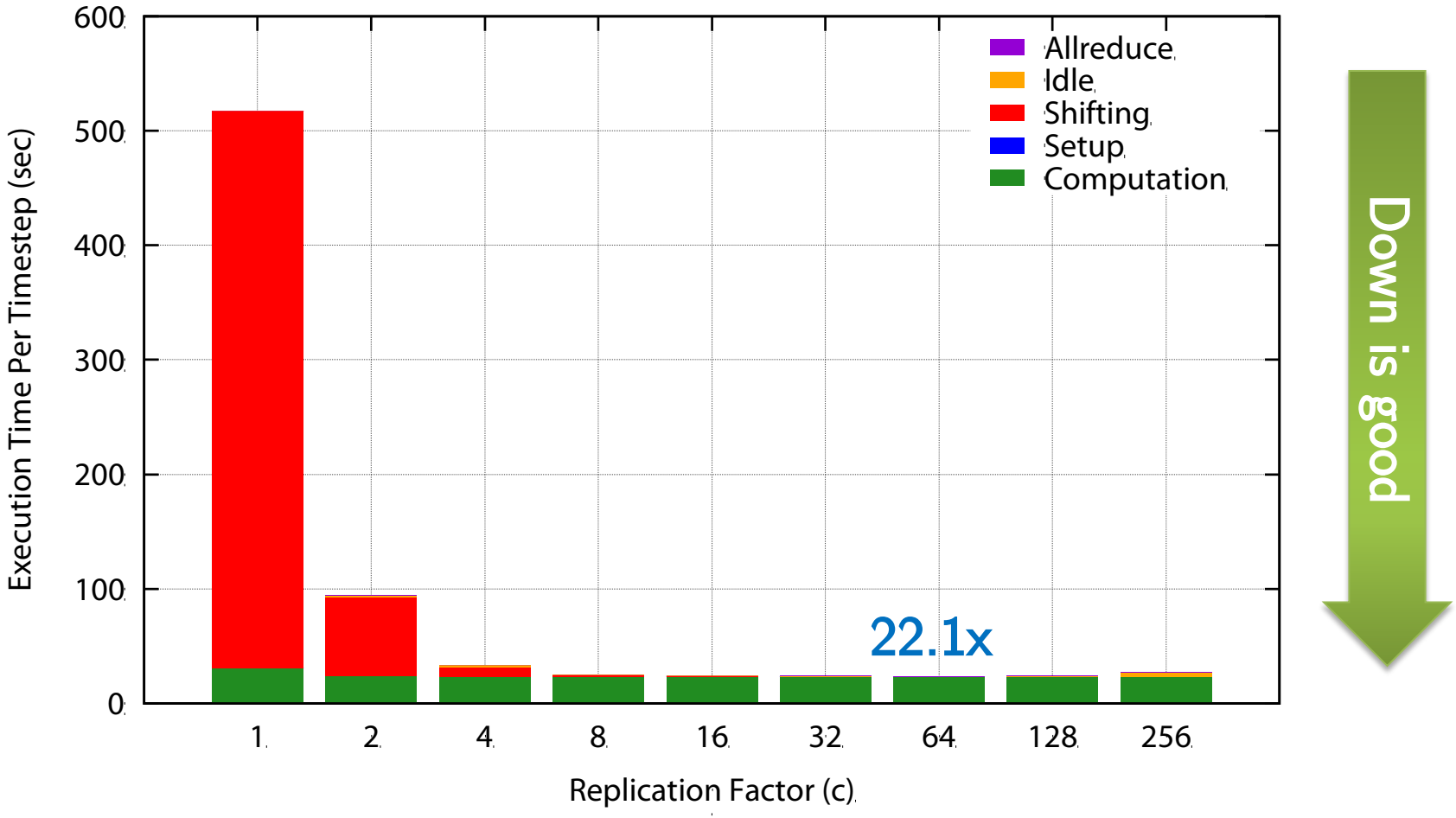
Larger Scale, More Benefit

- BG/Q, 8k cores, 16k particles

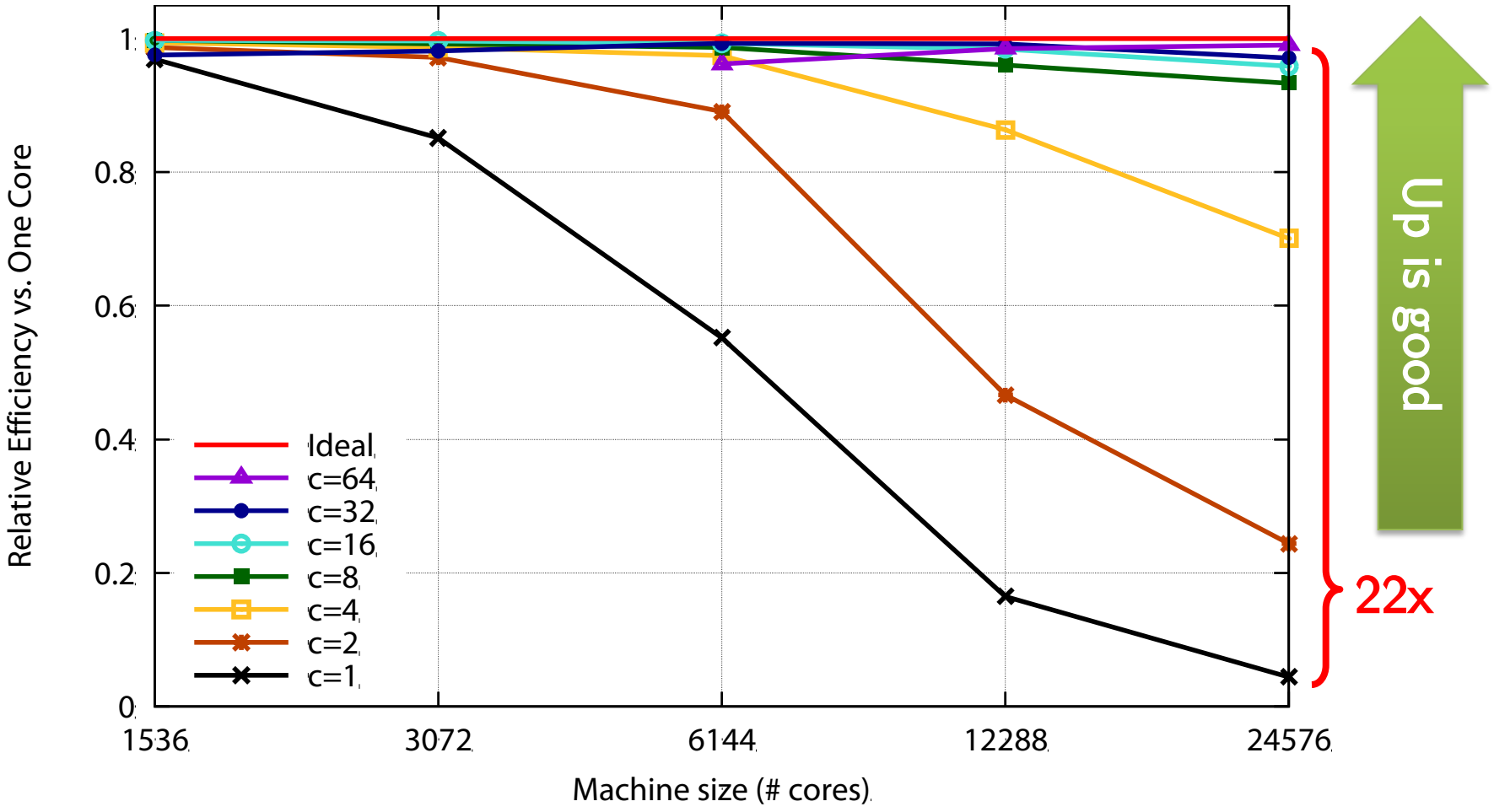


Extreme Communication

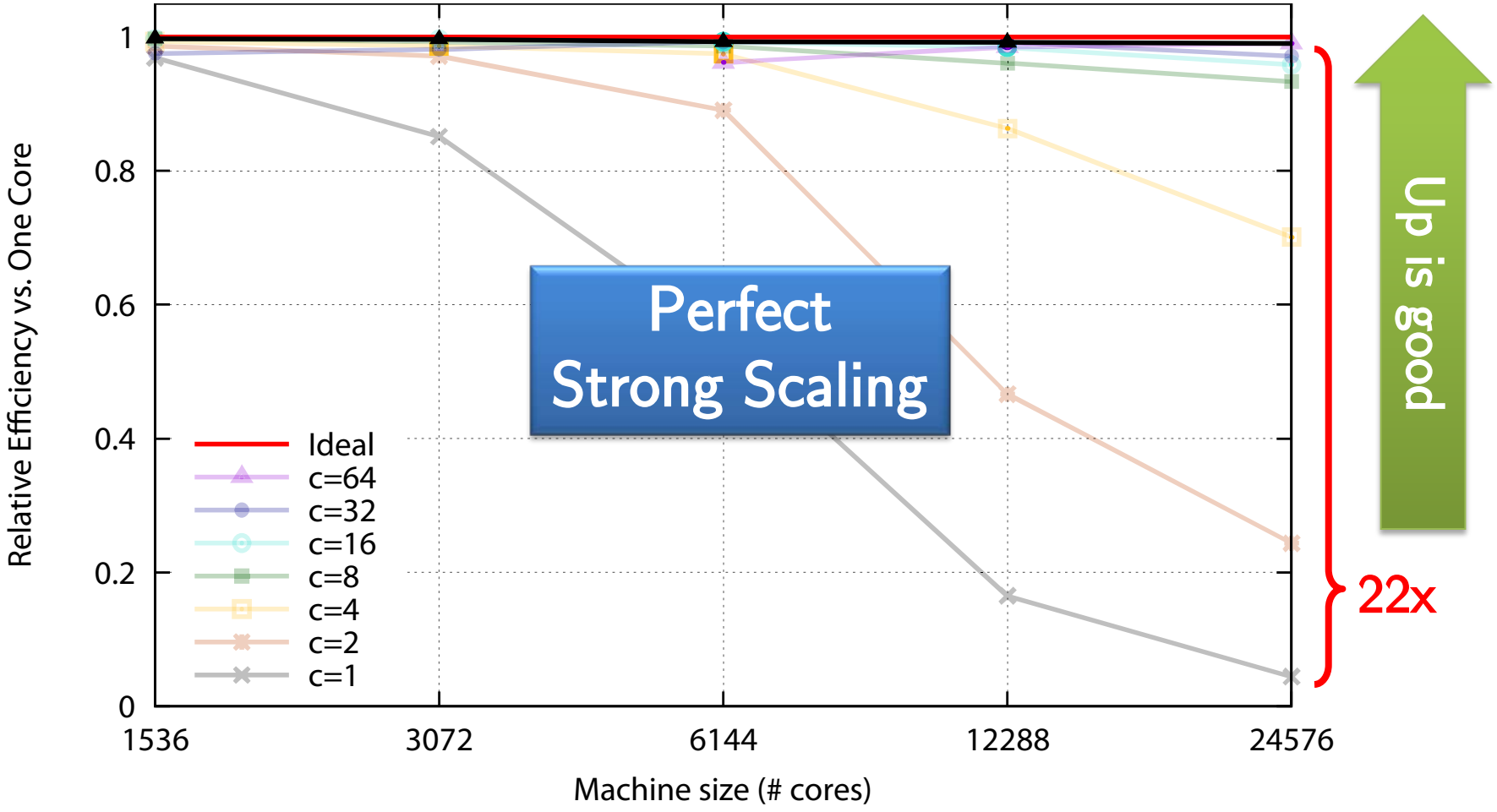
- Cray XC30, 24k cores, 24k particles



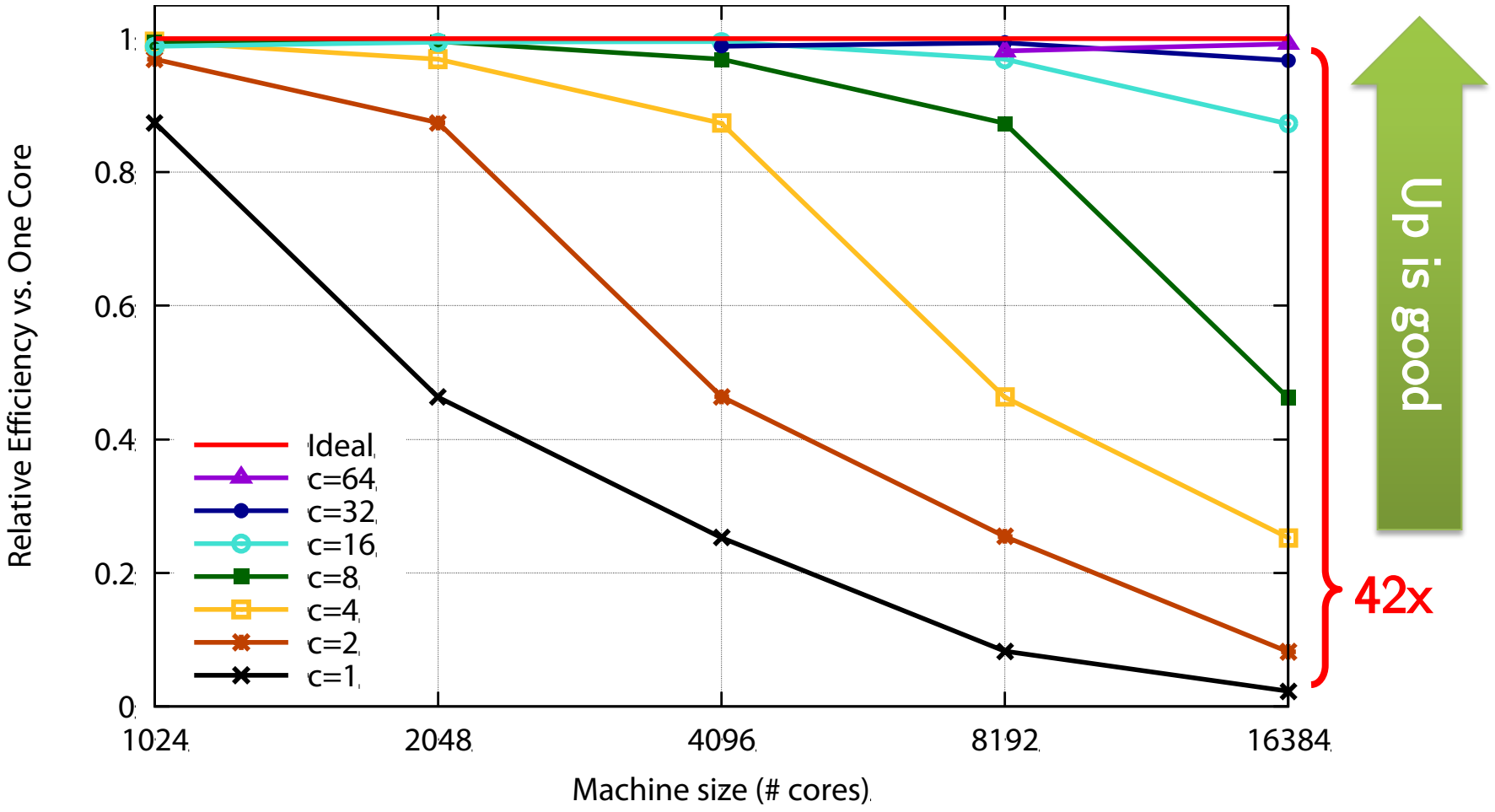
Cray XC30 24k particles, Strong Scaling



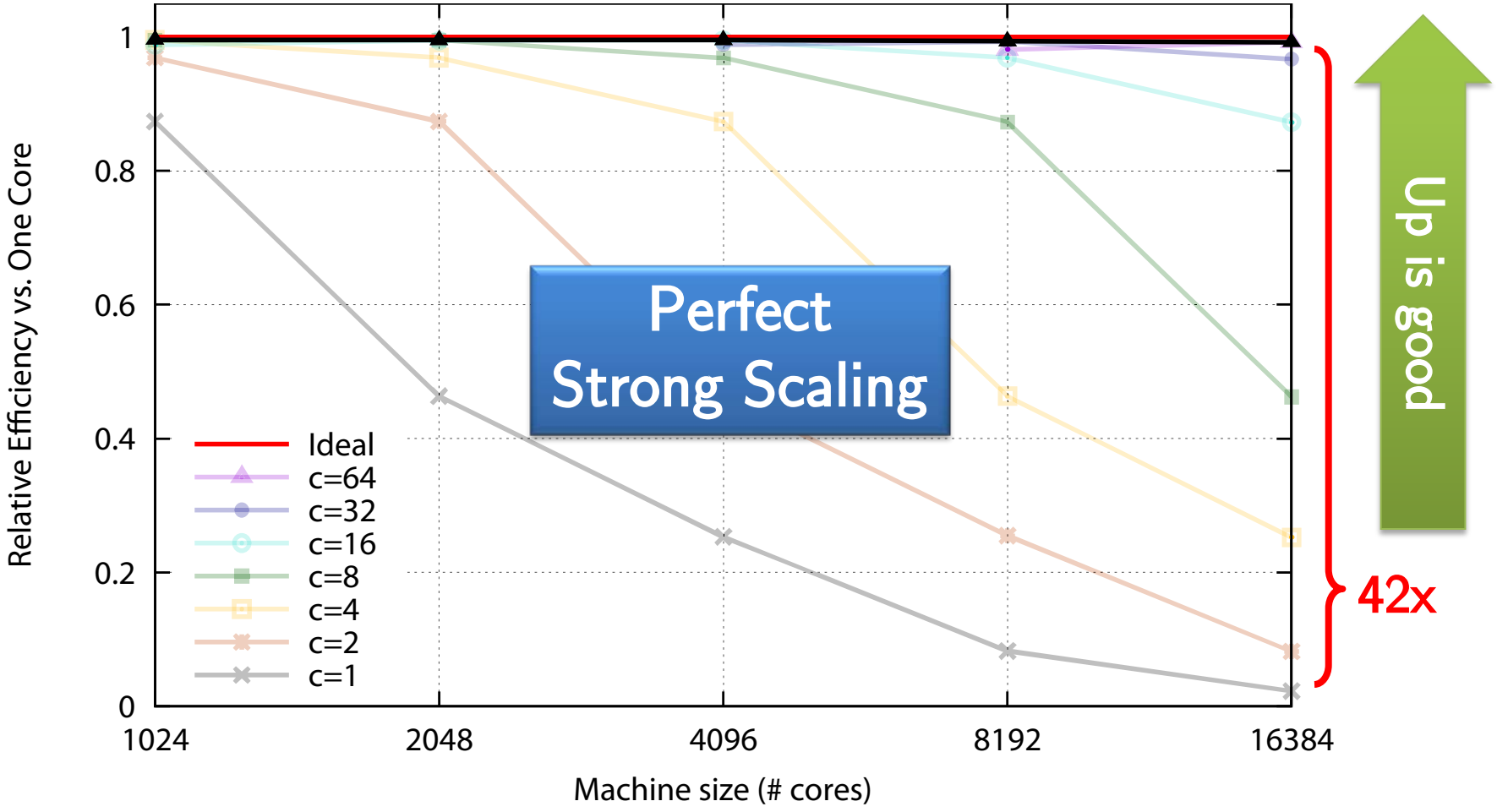
Cray XC30 24k particles, Strong Scaling



BG/Q 16k particles, Strong Scaling



BG/Q 16k particles, Strong Scaling



Extension for Cutoff

- Uniform distribution
- Similar to normal for loops.
- Works with any k-body for $k \geq 2$
- Derived communication lower bounds
- See paper for more details

Conclusions

- Computation- and communication-optimal all-triplets algorithms
- Supports communication-avoiding
  - $\mathbf{c}$ replicas saves $\mathbf{c^3}$ #messages and $\mathbf{c^2}$ #words
- Large scale results experienced up to
  - 99.98% reduction in communication time
  - 41.85x speedup
- Generalized algorithm to support cutoff distance for any k-body ($k \geq 2$)

References

- J. Li, Z. Zhou, and R. J. Sadus. **A cyclic force decomposition algorithm for parallelising three-body interactions in molecular dynamics simulations.** In J. Ni and J. Dongarra, editors, *IMSCCS (1)*, pages 338–343. IEEE Computer Society, 2006.
- J. Li, Z. Zhou, and R. J. Sadus, **Modified force decomposition algorithms for calculating three-body interactions via molecular dynamics**, *Computer Physics Communications*, vol. 175, no. 11-12, pp. 683 – 691, 2006.
- J. V. Sumanth, D. Swanson, and H. Jiang. **A novel force matrix transformation with optimal load-balance for 3-body potential based parallel molecular dynamics using atom-decomposition in a heterogeneous cluster environment.** In *Proceedings of the 14th international conference on High performance computing* (HiPC'07), 552-565, 2007.
- M. Driscoll, E. Georganas, P. Koanantakool, E. Solomonik, K. Yelick, **A Communication-Optimal N-Body Algorithm for Direct Interactions.** In proceedings of the IPDPS 2013
- G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, **Minimizing communication in linear algebra**. SIAM J. Mat. Anal. Appl., vol. 32, no. 3, 2011.

References (cont'd)

- M. Christ, J. Demmel, N. Knight, T. Scanlon, and K. A. Yelick. **Communication lower bounds and optimal algorithms for programs that reference arrays - part 1.** Technical Report UCB/EECS-2013-61, EECS Department, University of California, Berkeley, May 2013.
- S. Plimpton, **Fast parallel algorithms for short-range molecular dynamics.** J. Comput. Phys., vol. 117, no. 1, pp. 1–19, Mar. 1995. [Online]. Available: http://dx.doi.org/10.1006/jcph.1995.1039
- M. Snir, **A note on n-body computations with cutoffs**. Theory of Computing Systems, vol. 37, pp. 295–318, 2004.

Thank you!