Blood Flow Simulation at Petascale and Beyond

Richard (Rich) Vuduc

Casey Battaglino · Aparna Chandramowlishwaran · Jee Choi · Kent Czechowski · Chris McClanahan · Logan Moon · David S. Noble, Jr. · Murat (Efe) Guney [Intel] · Aashay Shringarpure [Google]

A. Rahimian · **G. Biros** · J. Vetter [ORNL+GT] · K. Madduri [LBNL]

CS 267 Guest Lecture @ UC Berkeley

April 21, 2011

Team MoBo

Abtin Rahimian

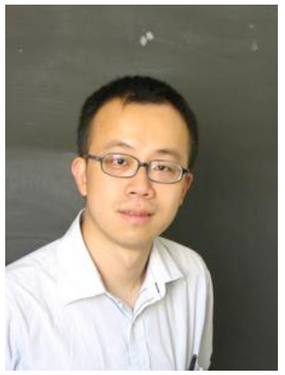
Ilya Lashuk

Shravan Veerapaneni

Denis Zorin

Aashay Shringarpure

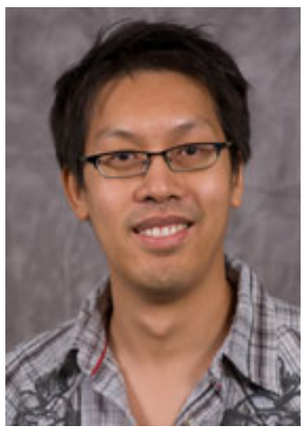
Logan Moon



Lexing Ying

Aparna Chandramowlishwaran

Rahul Sampath



Rich Vuduc

George Biros

Jeffrey Vetter

OAK RIDGE National Laboratory

DARPA

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA ·

NVIDIA

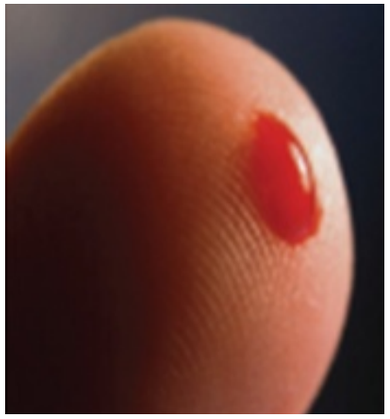
NATIONAL INSTITUTES OF HEALTH

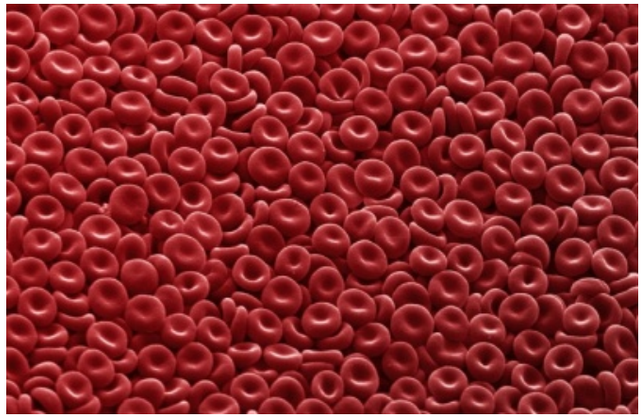
NSF

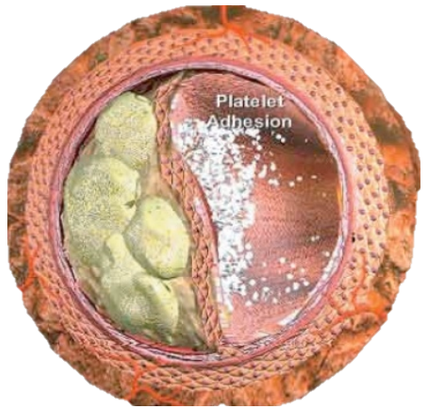
MARBLE OR GNOCCHI?

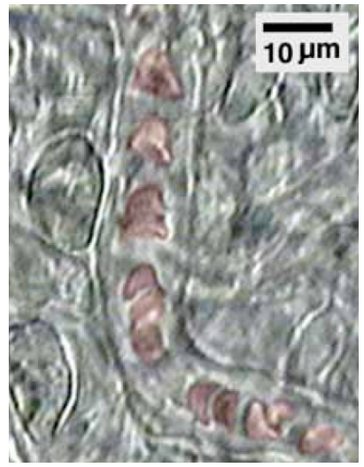
http://www.faqs.org/photo-dict/photofiles/list/2862/3795glass_marbles.jpg

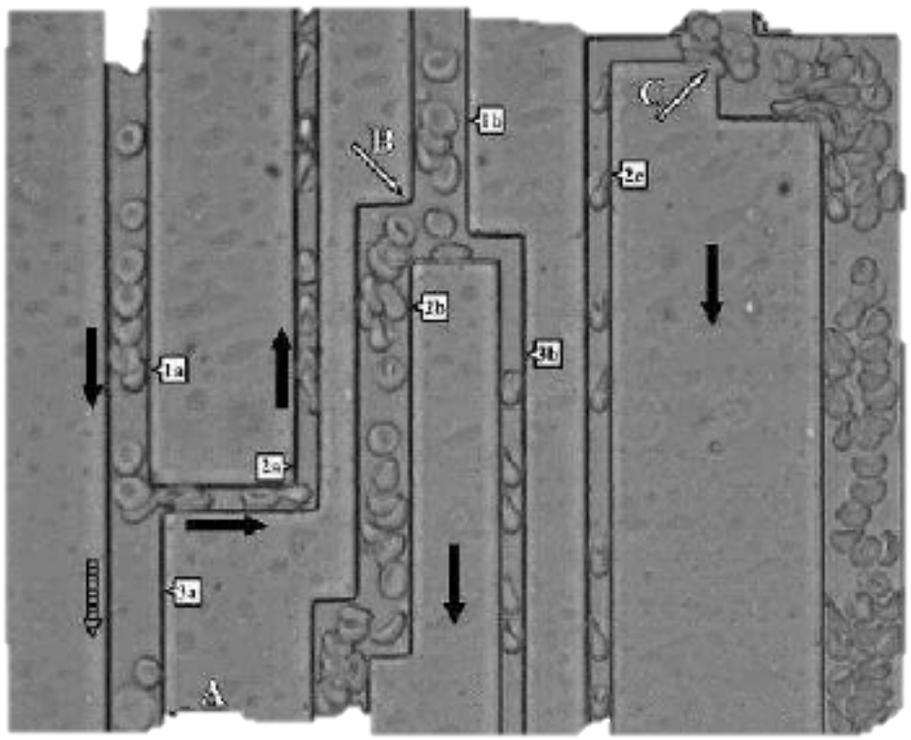
http://www.eatingwithangela.com/EATINGWITHANGELA/assets/Image/EatingWell/WholeWheatGnocchiUncooked.jpg







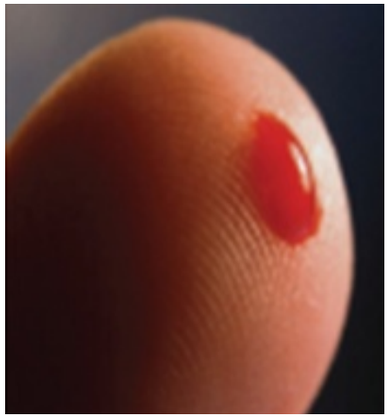


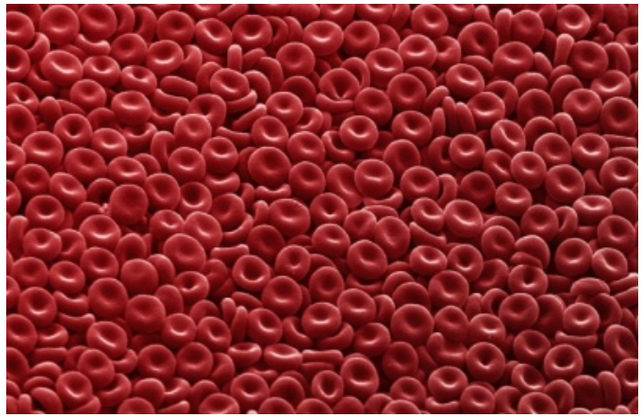


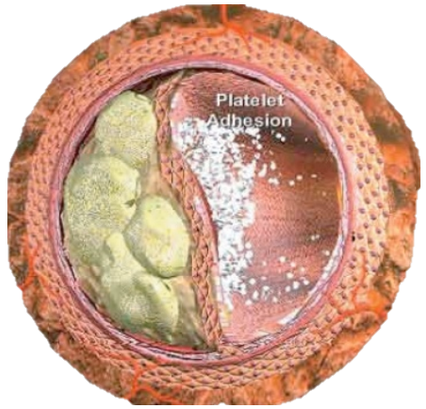
CONTEXT: MOBO

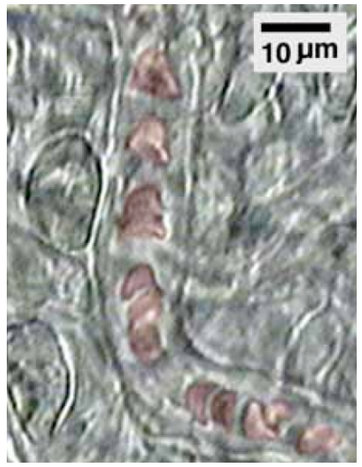
("MOVING BOUNDARIES")

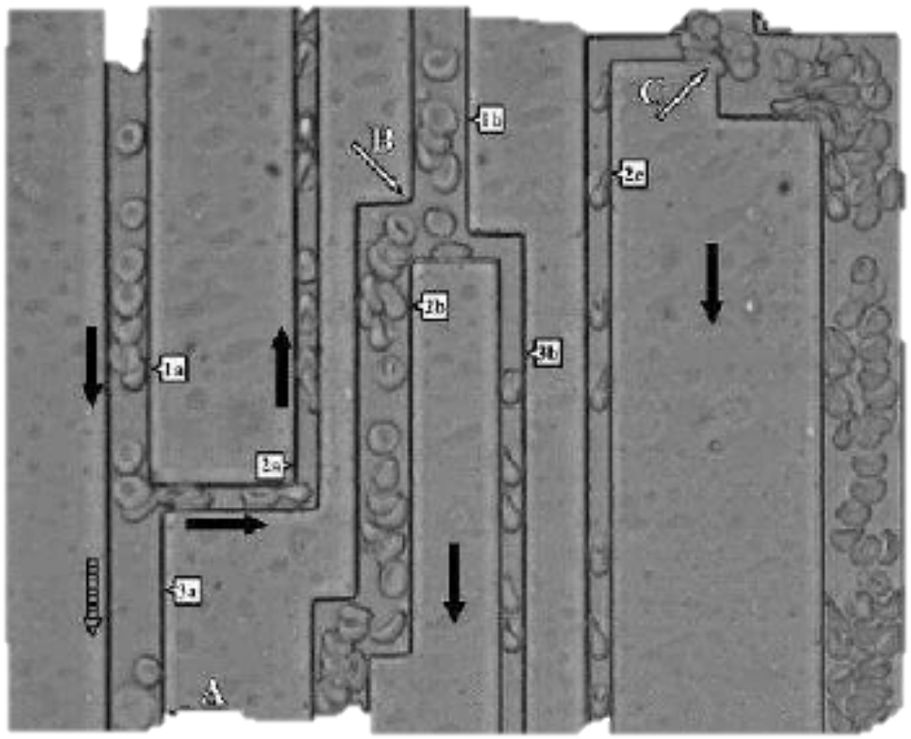
Citation: **A. Rahimian**, I. Lashuk, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, S. Veerapaneni, J. Vetter, R. Vuduc, D. Zorin, and **G. Biros**. "Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures." In *Proc. ACM/IEEE Conf. Supercomputing (SC)*, Nov. 2010.
Winner, Gordon Bell Prize. `http://dx.doi.org/10.1109/SC.2010.42`

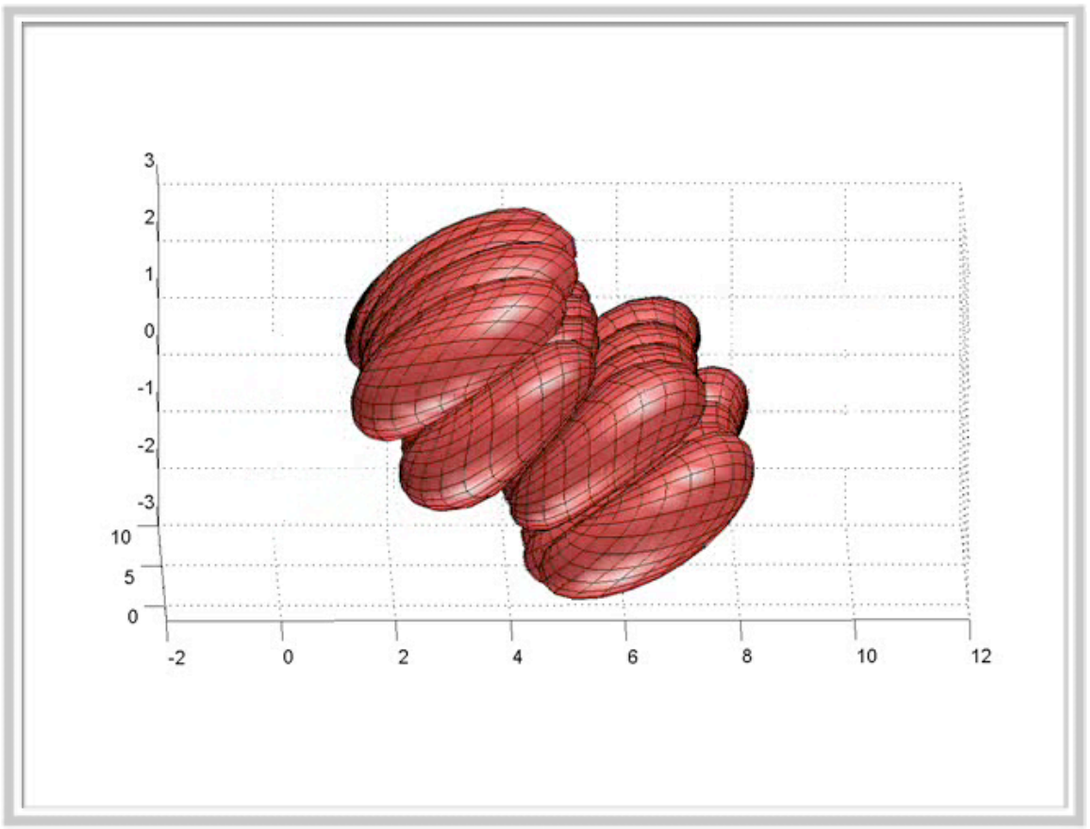










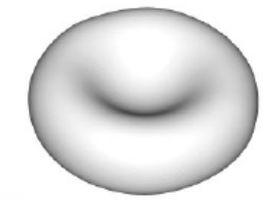


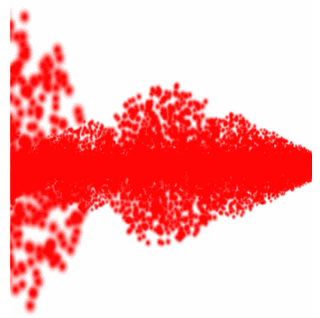
CONTEXT: MOBO

(“MOVING BOUNDARIES”)

Citation: **A. Rahimian**, I. Lashuk, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, S. Veerapaneni, J. Vetter, R. Vuduc, D. Zorin, and **G. Biros**. “Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures.” In *Proc. ACM/IEEE Conf. Supercomputing (SC)*, Nov. 2010.
Winner, Gordon Bell Prize. `http://dx.doi.org/10.1109/SC.2010.42`

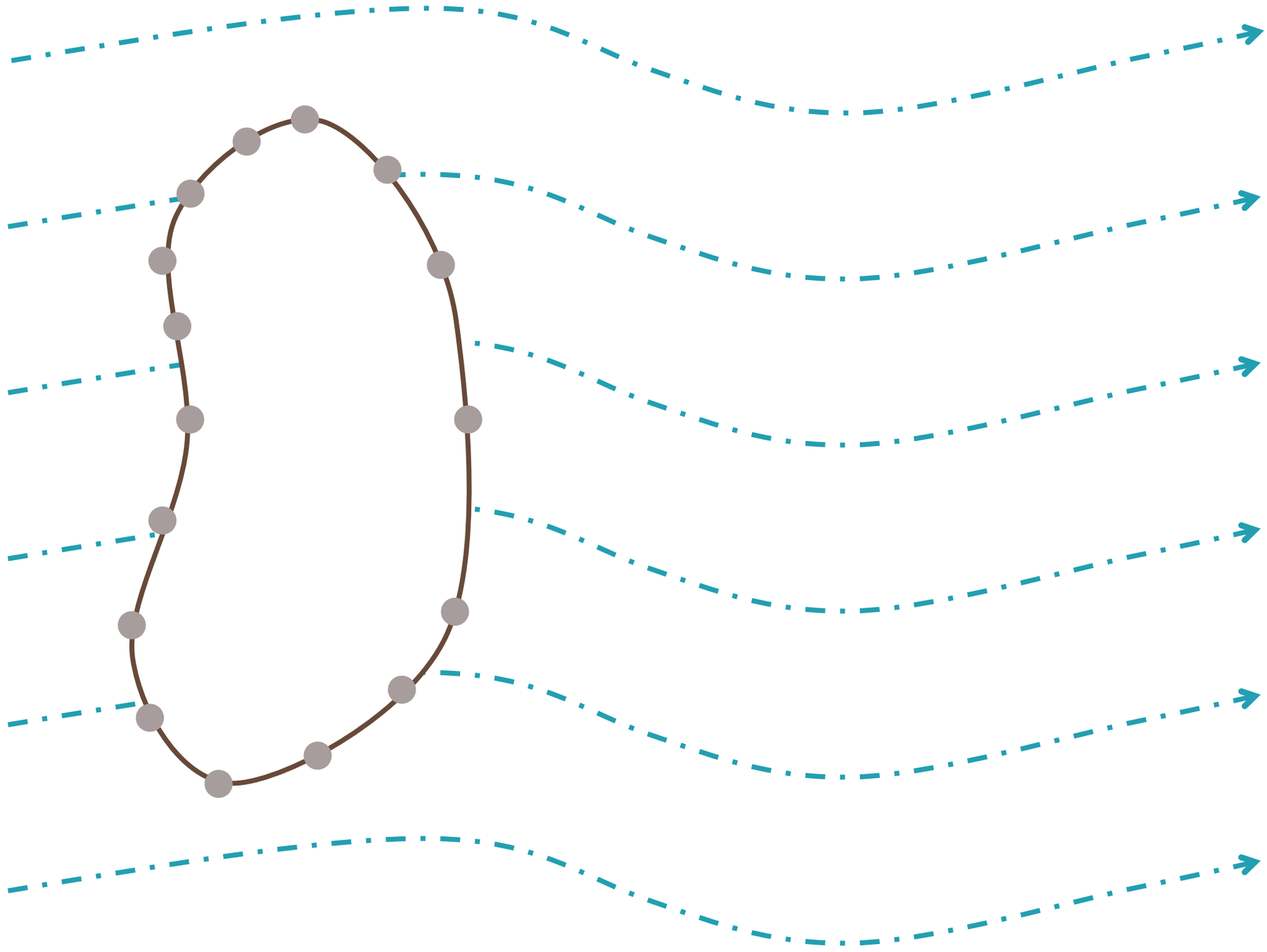
Deformable Red Blood Cells



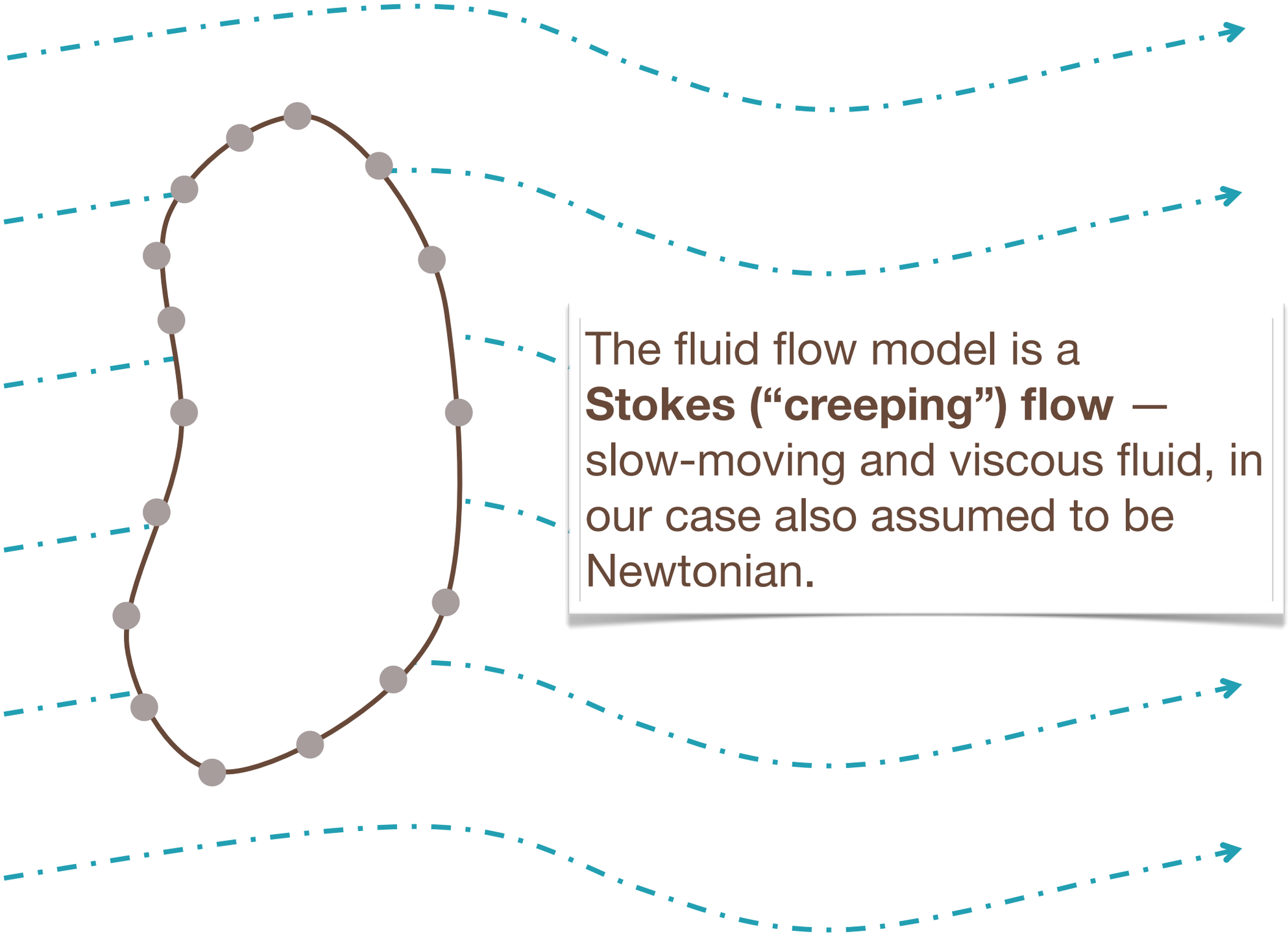


- Prior work with same physical fidelity
  - **1,200 cells**: **Sequential + integral equations**
    Zinchenko et al. (2003)
  - **14,000 cells**: **IBM BG/P + Lattice Boltzmann**
    O(10k) unknowns/cell
    Clausen et al. (2010)
- MoBo: **260 million cells** on **200k cores** (Jaguar @ ORNL)
  - **CPU, GPU + integral equations + implicit AMR**
    O(100) unknowns / cell
  - Key to scaling: Optimal n-body methods based on the
    fast multipole method (FMM) on highly non-uniform domains

- Problem formulation and algorithms

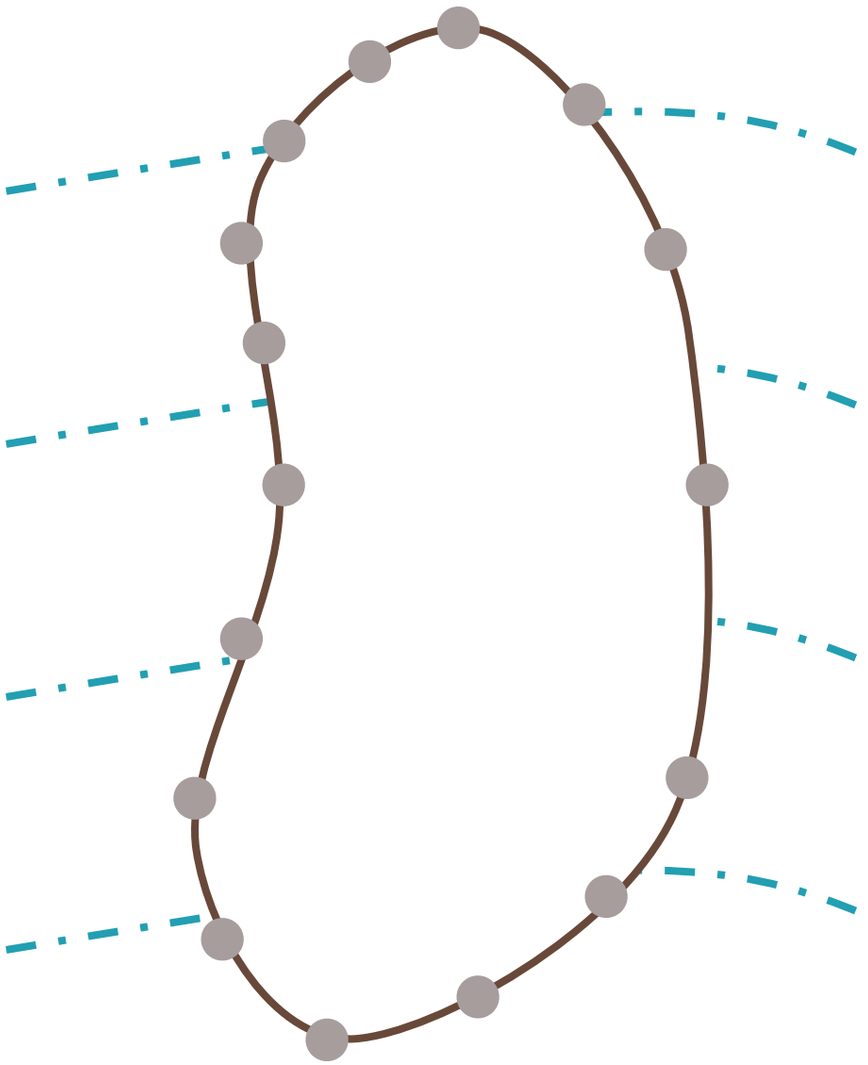


Vesicle flow: Model a red blood cell (RBCs) as fluid-filled deformable and inextensible sac in viscous solution



The fluid flow model is a **Stokes (“creeping”) flow** — slow-moving and viscous fluid, in our case also assumed to be Newtonian.

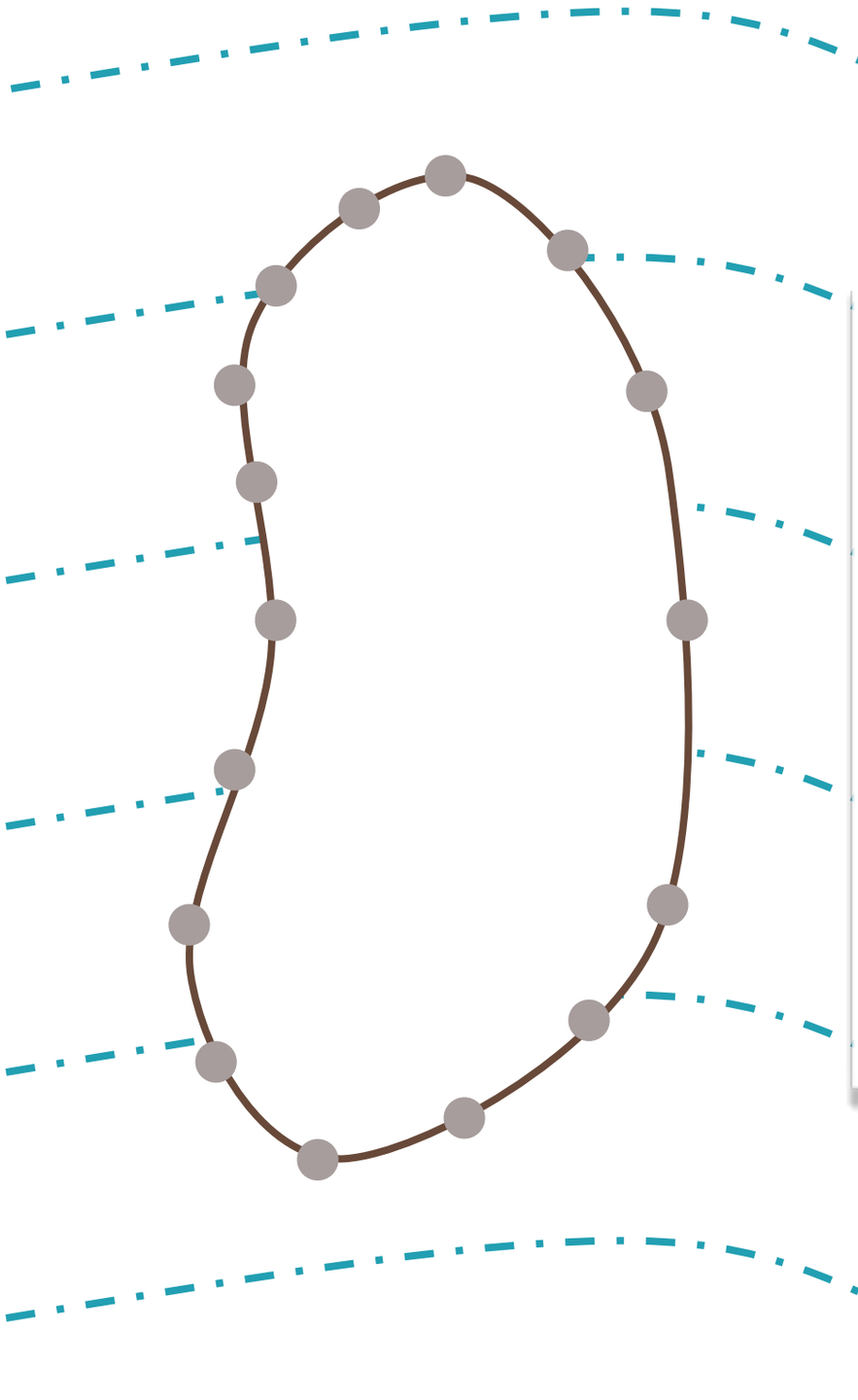
Vesicle flow: Model a red blood cell (RBCs) as fluid-filled deformable and inextensible sac in viscous solution



The numerical method is based on a **boundary integral formulation**, in which the fluid is represented implicitly and we need only discretize the cell boundary.[1]

[1] In contrast to Lattice Boltzman or finite-element methods, in which we discretize the entire domain.

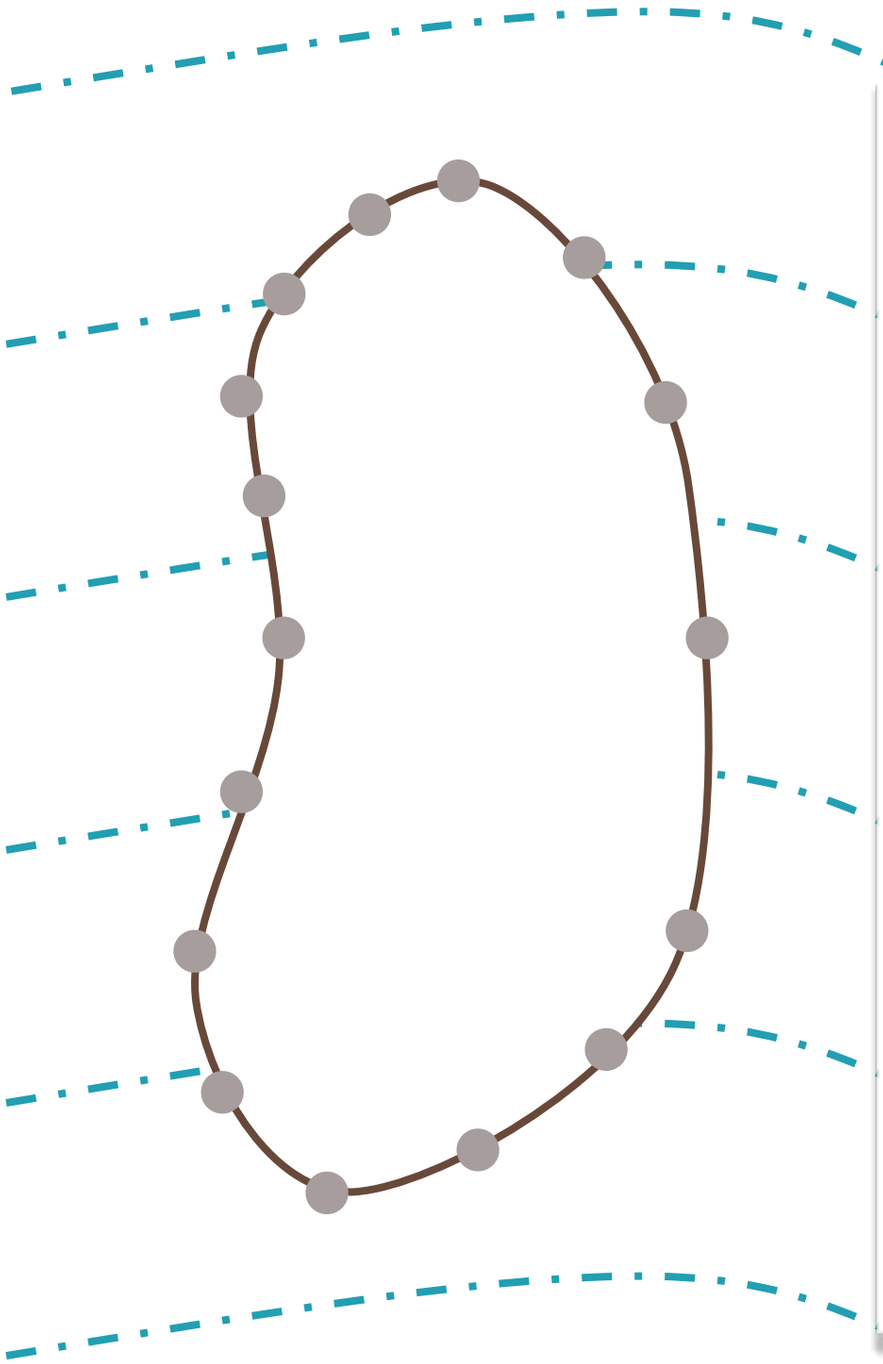
Vesicle flow: Model a red blood cell (RBCs) as fluid-filled deformable and inextensible sac in viscous solution



Challenges in implementing boundary integral methods:

- Stiffness, due to high-order derivatives required to accurately capture RBC deformations
- Efficient evaluation of long-range interactions

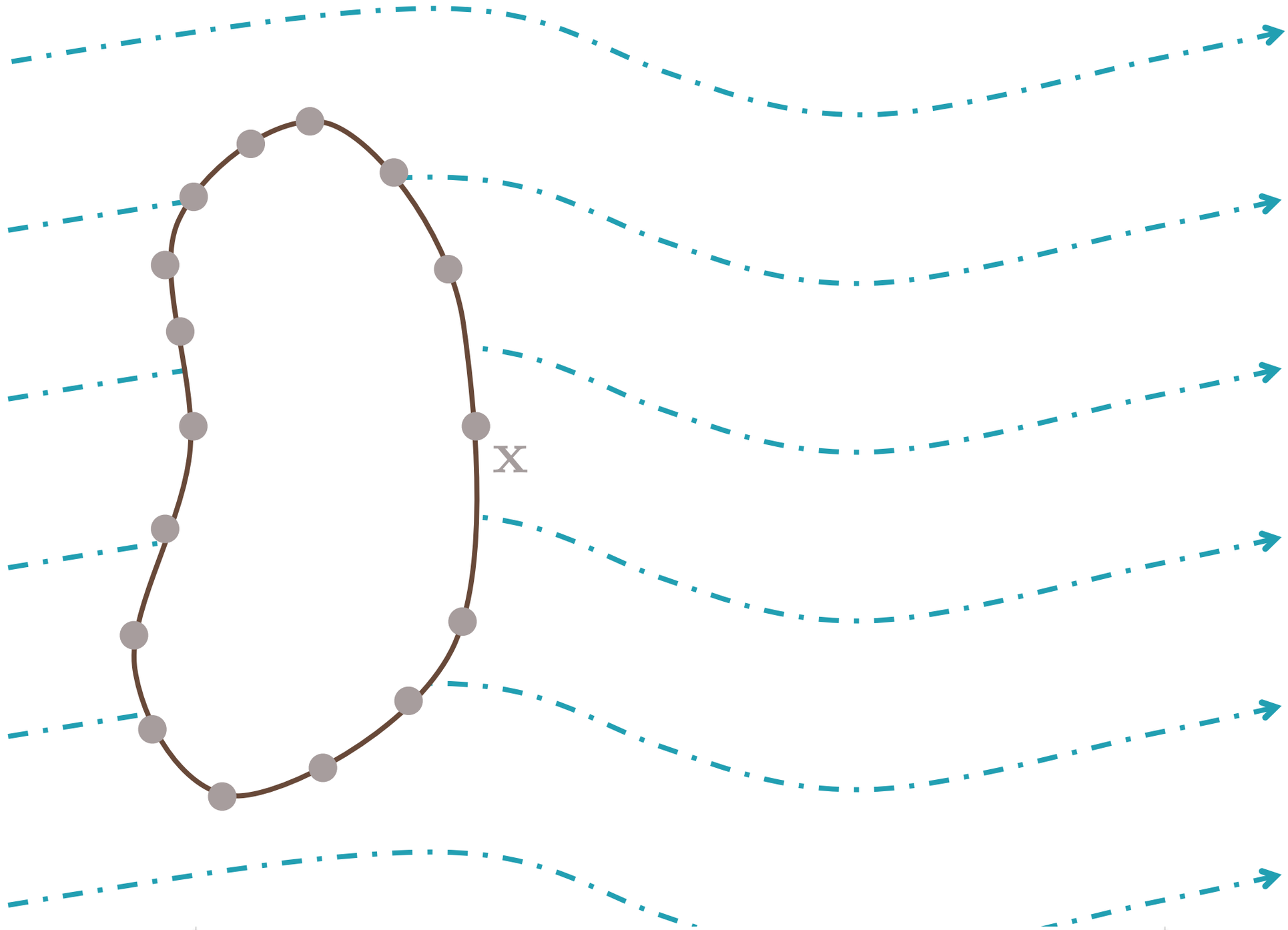
Vesicle flow: Model a red blood cell (RBCs) as fluid-filled deformable and inextensible sac in viscous solution



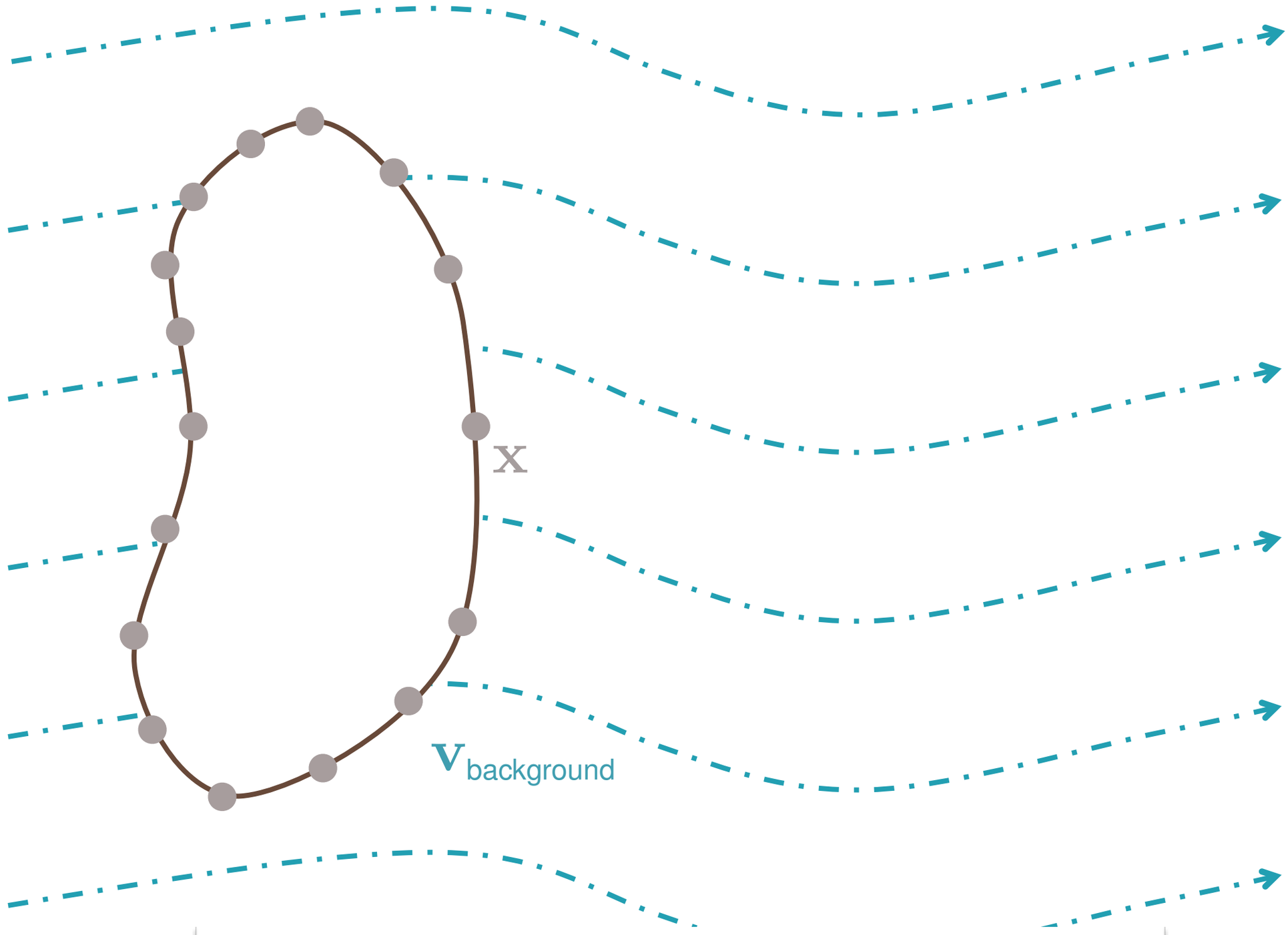
Features of our approach:

- Represent RBC in a **spherical harmonics** basis, which permits accurate high-order derivative computations
- Efficient evaluation of long-range interactions via **FMM**
- Time-stepping with a **multistep, semi-implicit** method that is, at least empirically, unconditionally stable

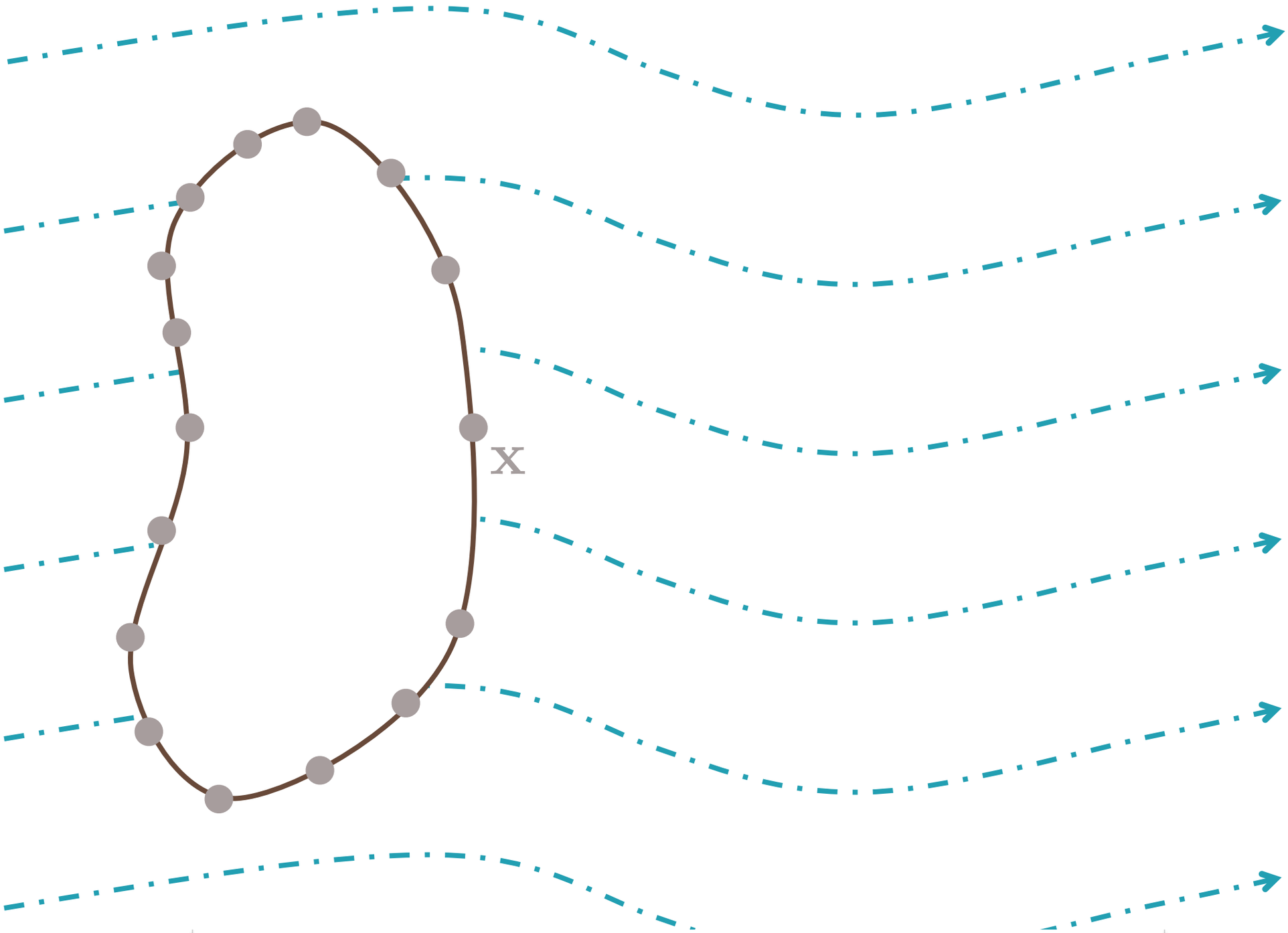
Vesicle flow: Model a red blood cell (RBCs) as fluid-filled deformable and inextensible sac in viscous solution



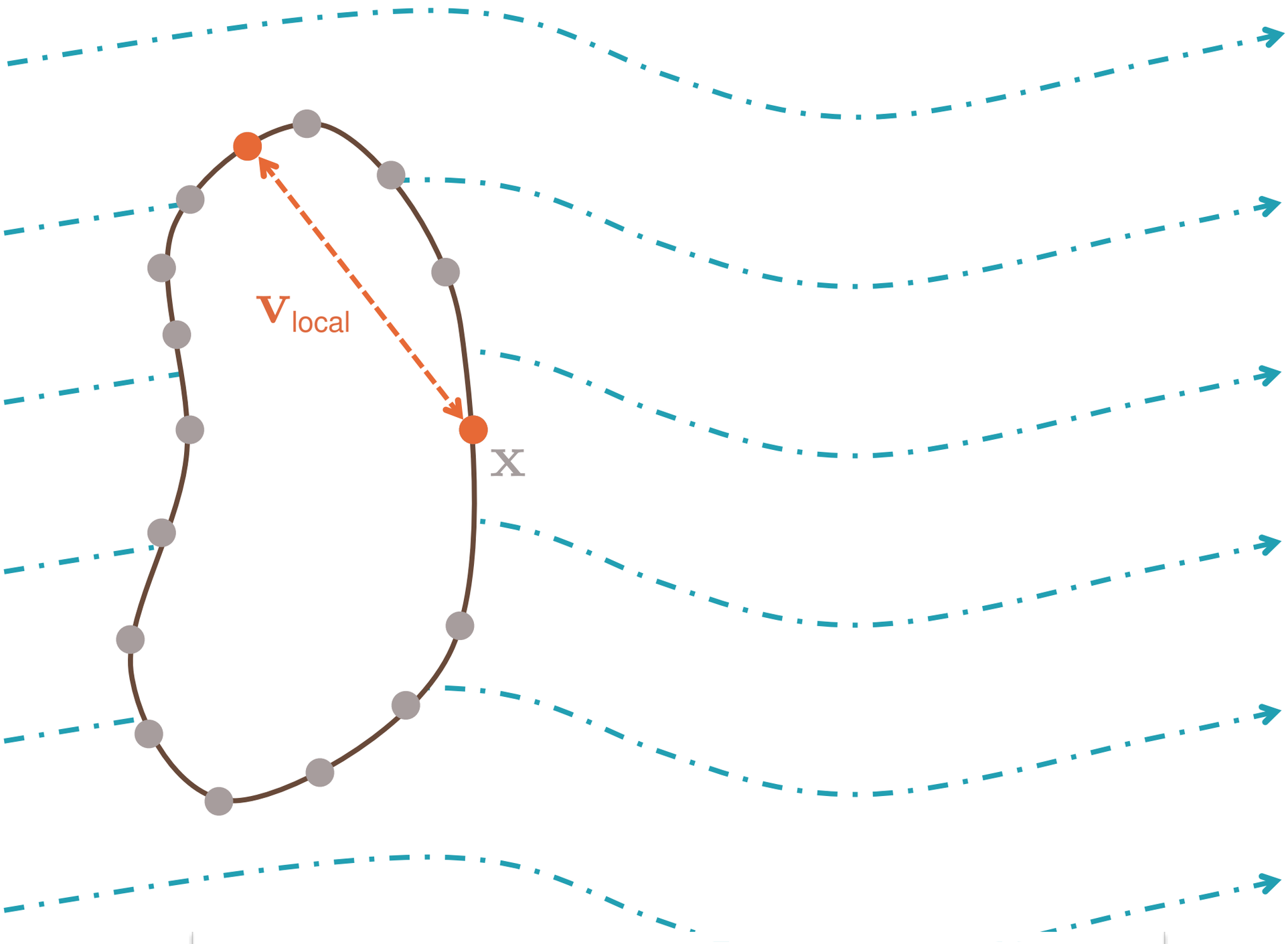
$$\frac{d\mathbf{x}}{dt} = \mathbf{v}_{\text{background}}(\mathbf{x}) + \mathbf{v}_{\text{interaction}}(\mathbf{x})$$



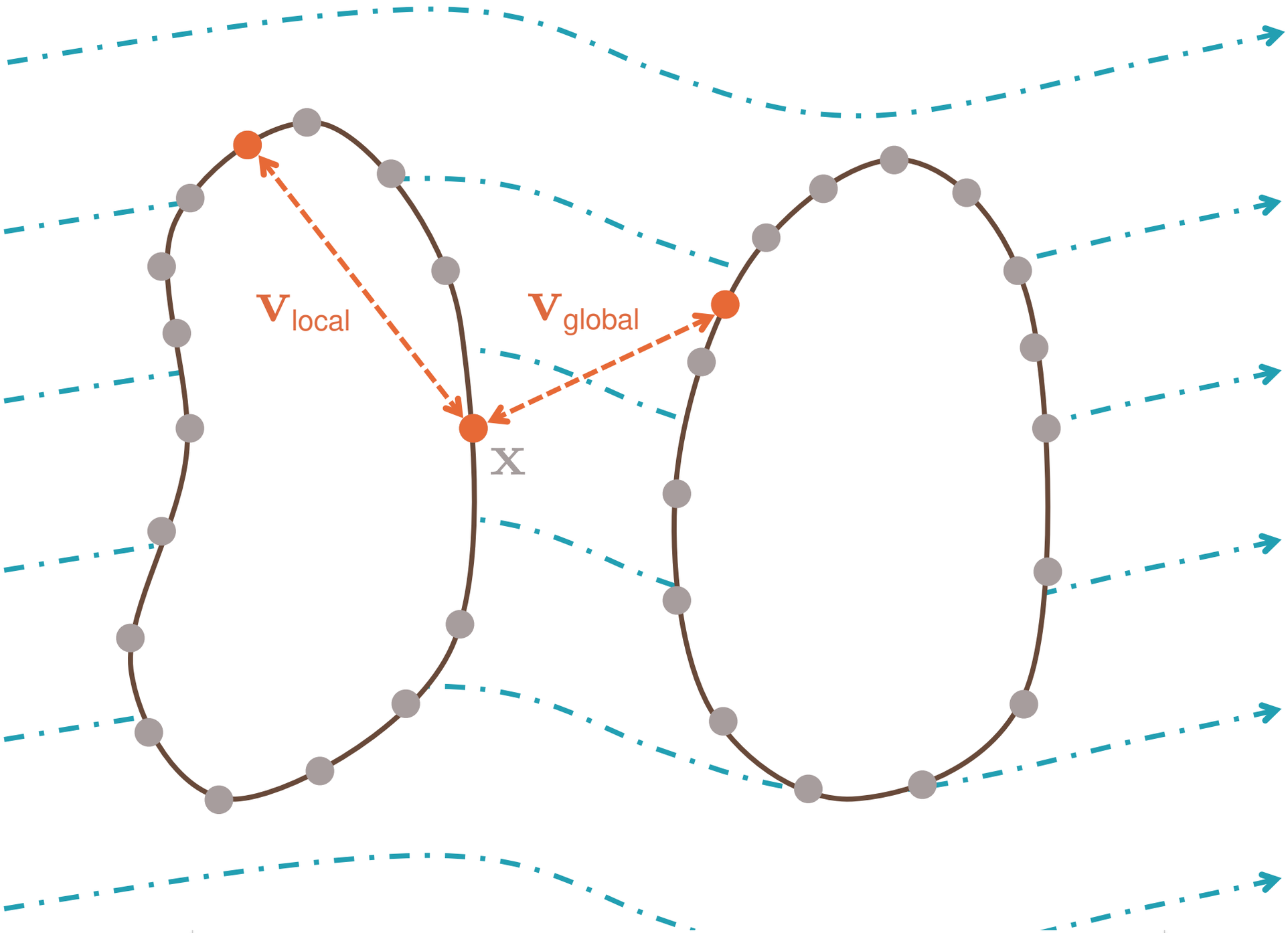
$$\frac{d\mathbf{x}}{dt} = \mathbf{v}_{\text{background}}(\mathbf{x}) + \mathbf{v}_{\text{interaction}}(\mathbf{x})$$



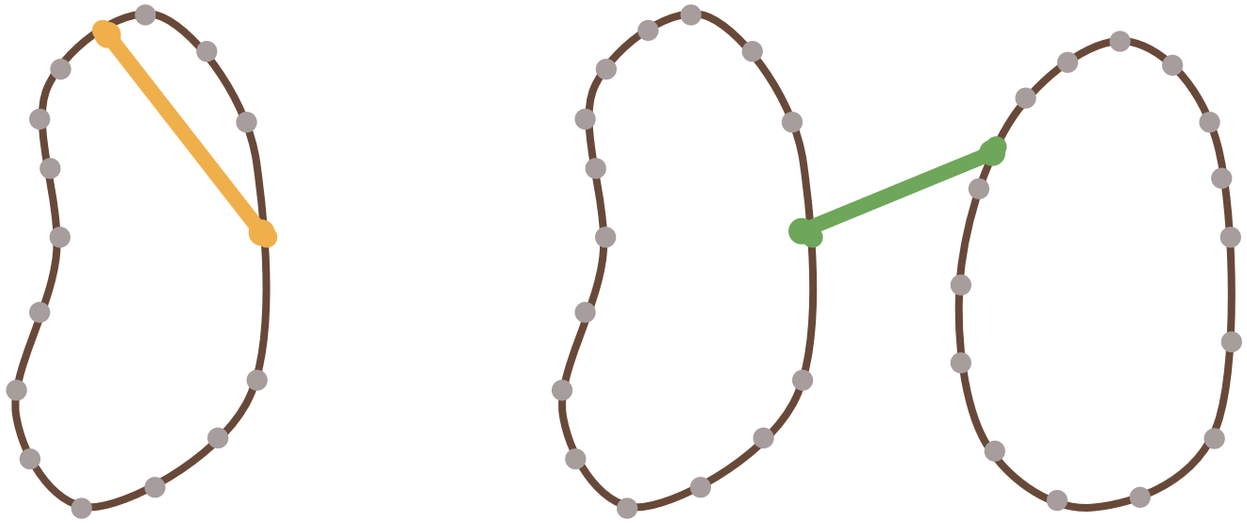
$$\mathbf{v}_{\text{interaction}}(\mathbf{x}) = \mathbf{v}_{\text{local}}(\mathbf{x}) + \mathbf{v}_{\text{global}}(\mathbf{x})$$

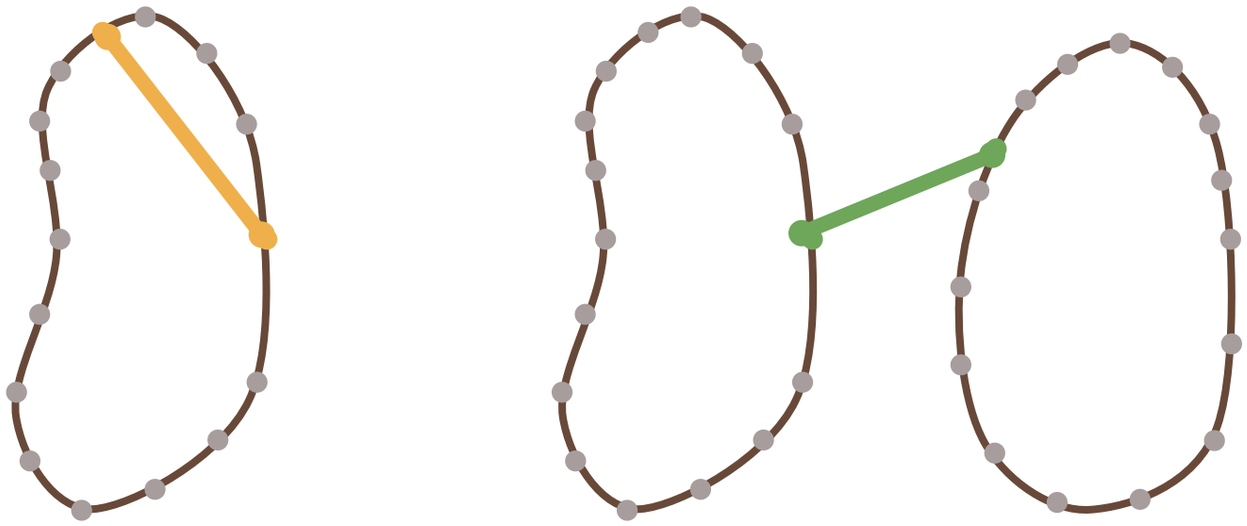


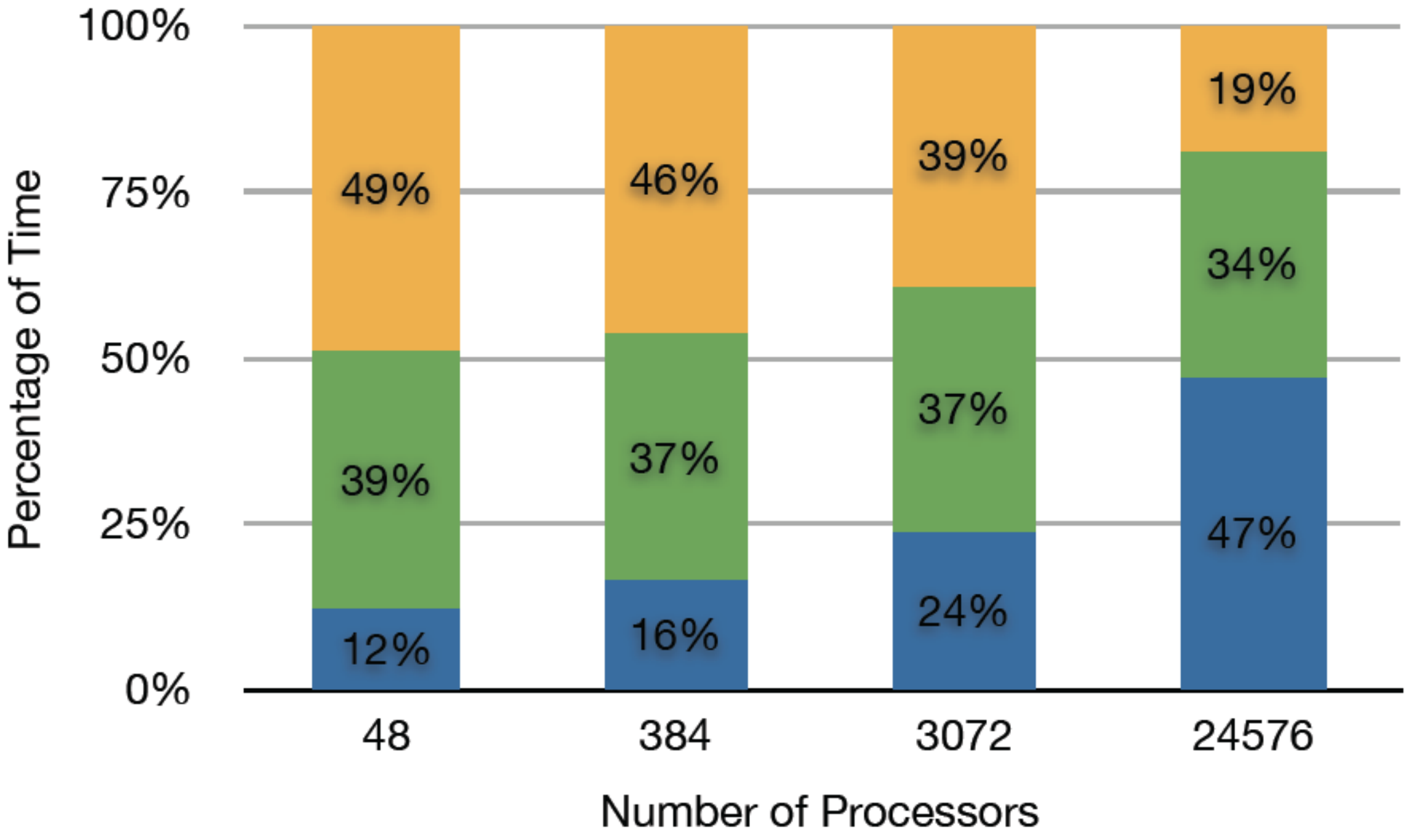
$$\mathbf{v}_{\text{interaction}}(\mathbf{x}) = \mathbf{v}_{\text{local}}(\mathbf{x}) + \mathbf{v}_{\text{global}}(\mathbf{x})$$

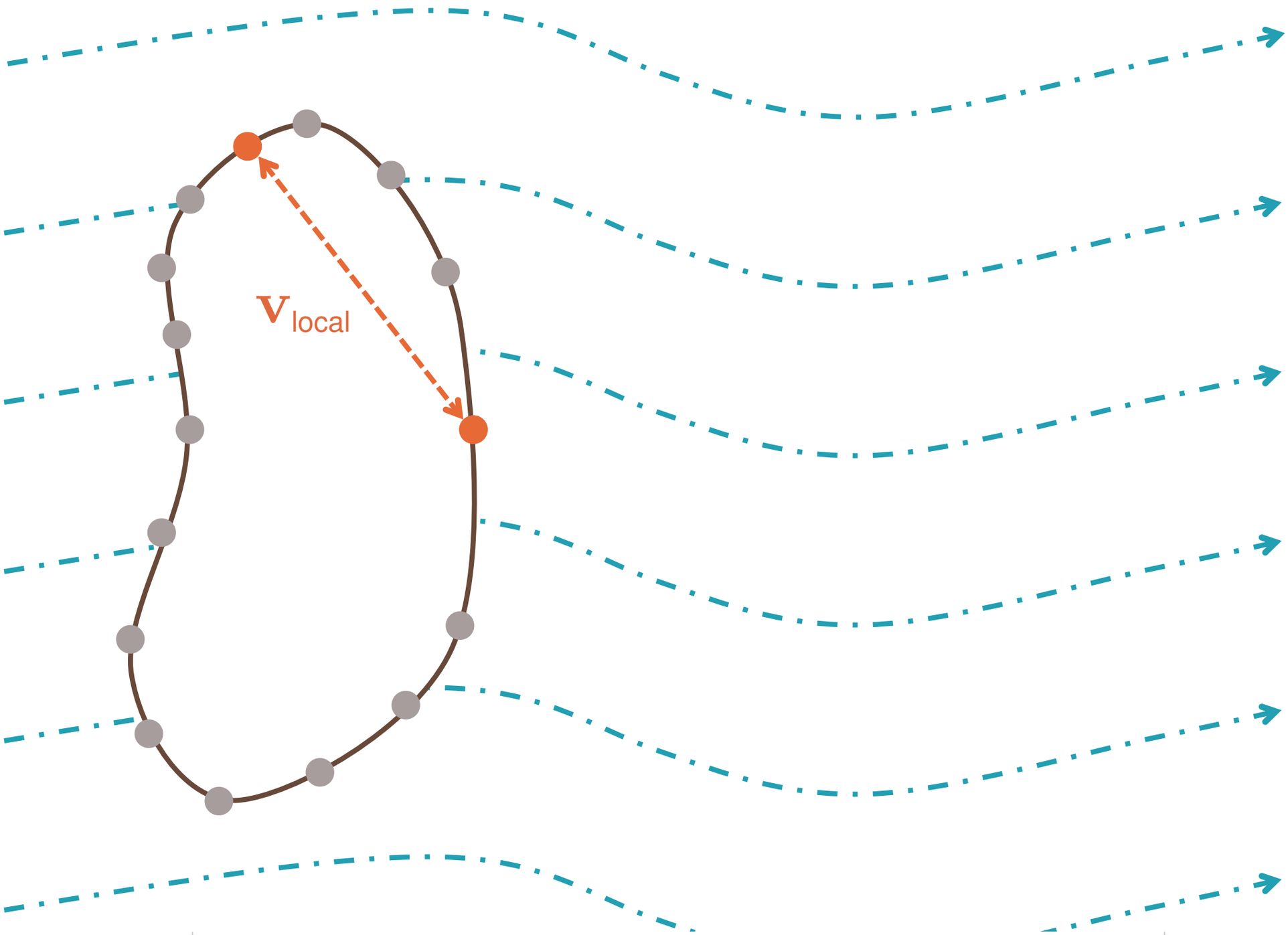


$$\mathbf{v}_{\text{interaction}}(\mathbf{x}) = \mathbf{v}_{\text{local}}(\mathbf{x}) + \mathbf{v}_{\text{global}}(\mathbf{x})$$

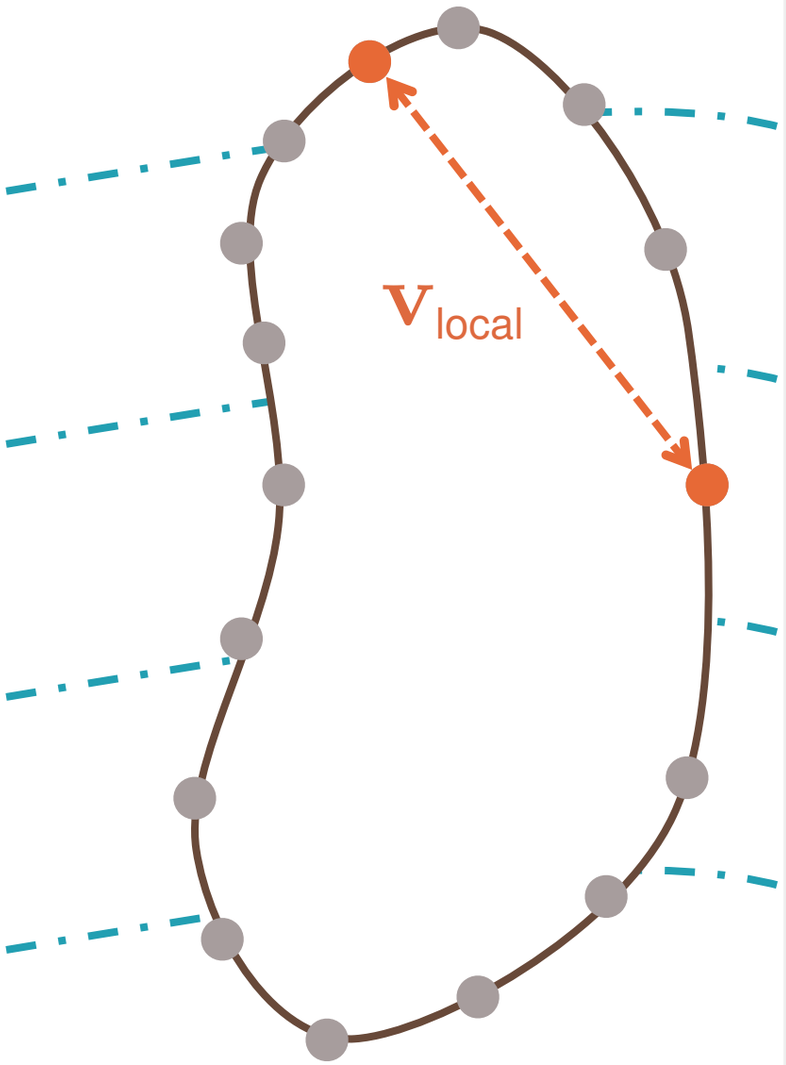





100%
75%
50%
25%
0%
Percentage of Time
49%
39%
12%
46%
37%
16%
39%
37%
24%
19%
34%
47%
48
384
3072
24576
Number of Processors



$$\mathbf{v}_{\text{interaction}}(\mathbf{x}) = \mathbf{v}_{\text{local}}(\mathbf{x}) + \mathbf{v}_{\text{global}}(\mathbf{x})$$



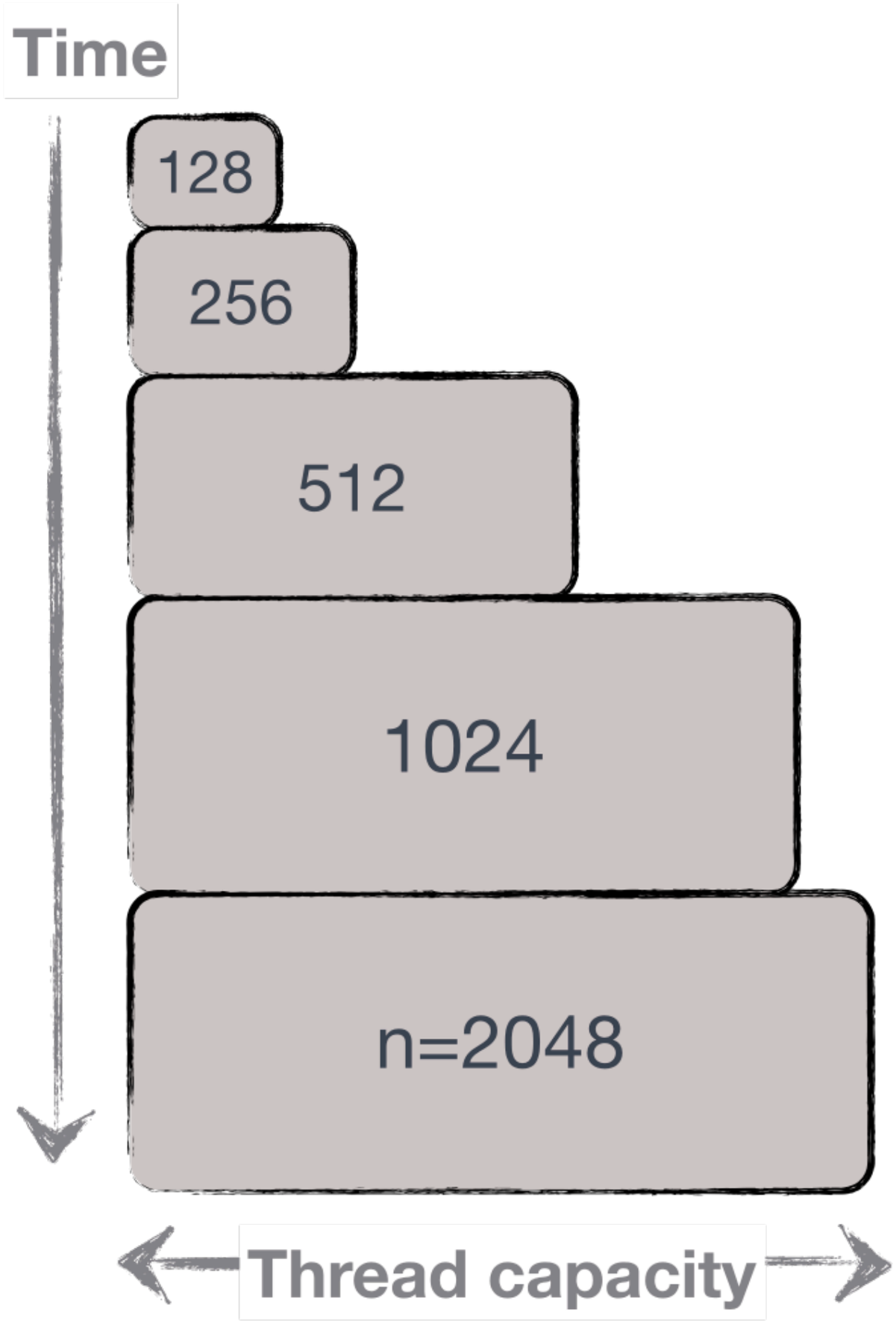
Intra-cell forces computation:

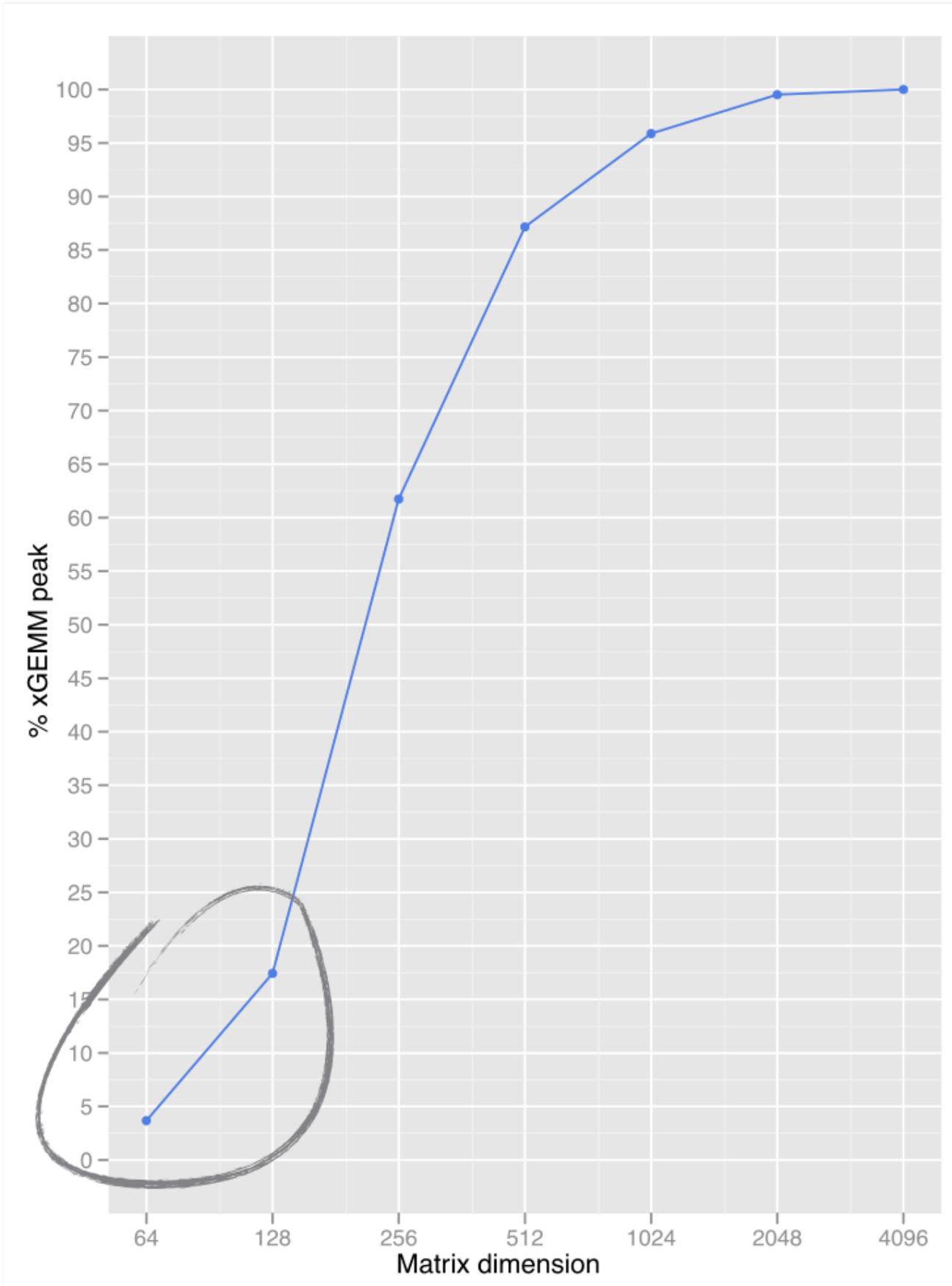
- Spherical harmonics (Legendre & Fourier transforms)
- Singular quadratures
- Derivatives
- Surface remeshing
- Semi-implicit solver (BiCGStab)

Pleasingly parallel over RBCs.

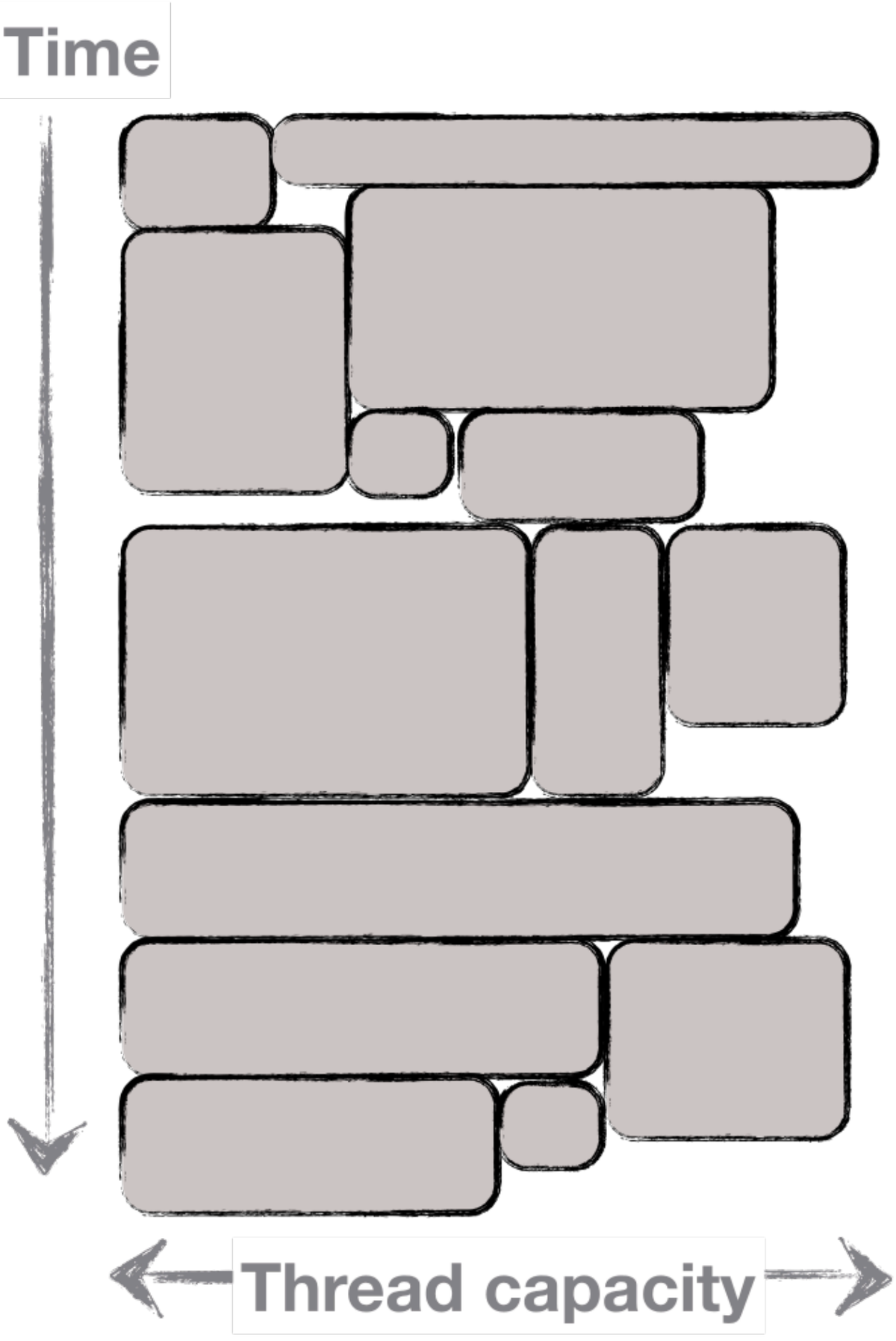
Time dominated by ~ many small matrix multiplies.

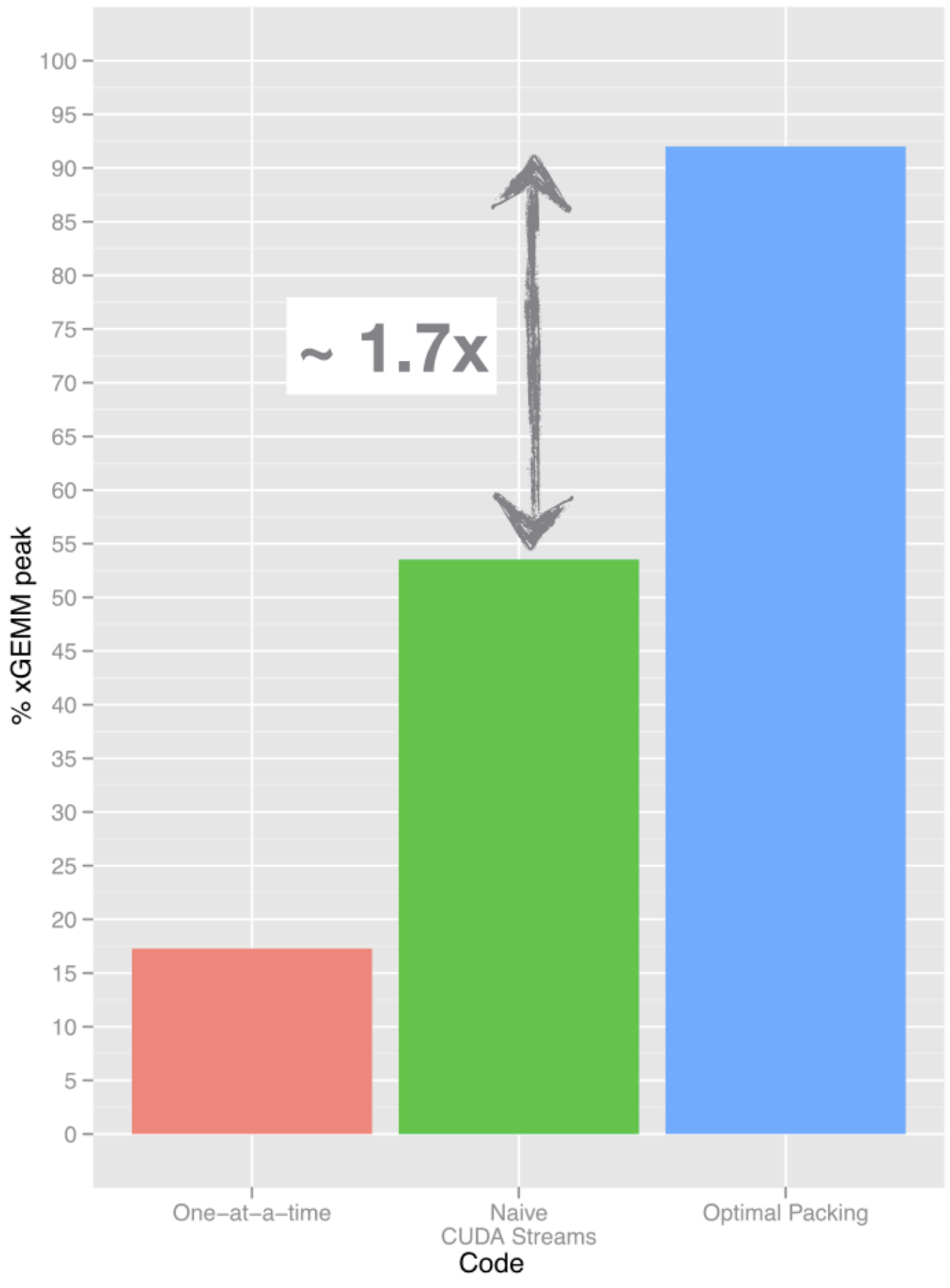
$$\mathbf{v}_{\text{interaction}}(\mathbf{x}) = \mathbf{v}_{\text{local}}(\mathbf{x}) + \mathbf{v}_{\text{global}}(\mathbf{x})$$


Time
128
256
512
1024
n=2048
Thread capacity

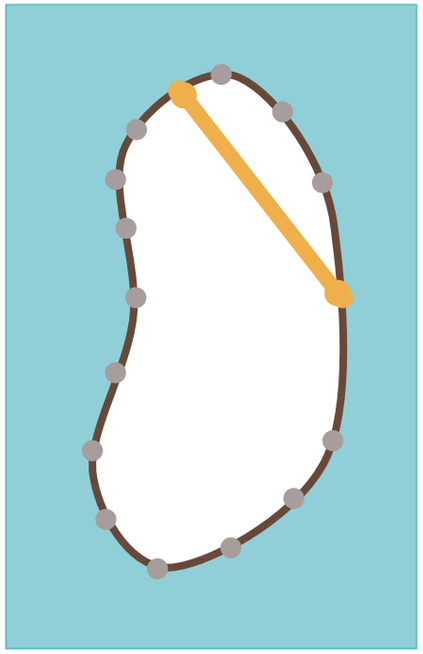

100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
% xGEMM peak
64
128
256
512
1024
2048
4096
Matrix dimension

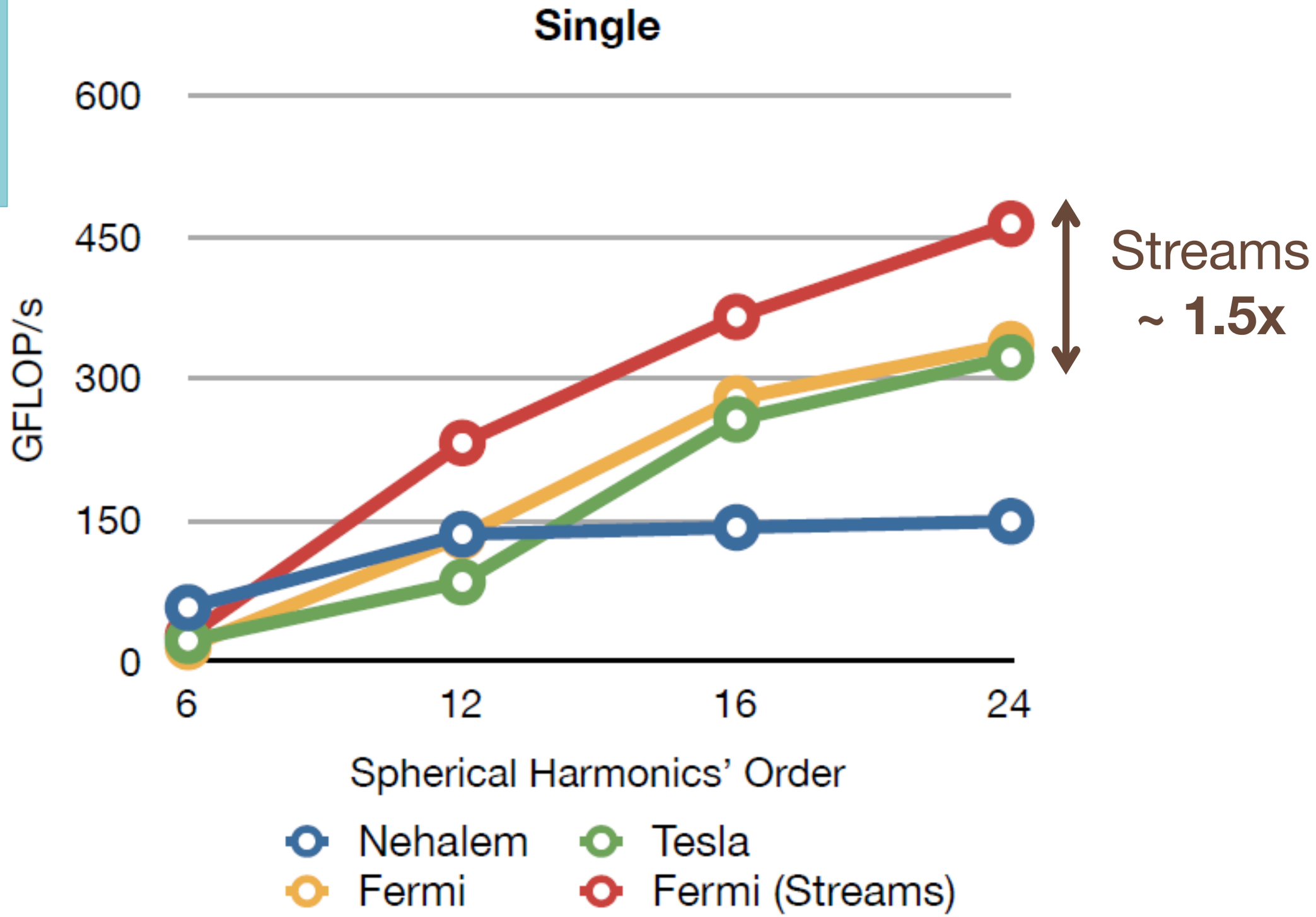
Given a sequence of “small” independent kernel invocations, we use the **CUDA Streams interface** to fill capacity.



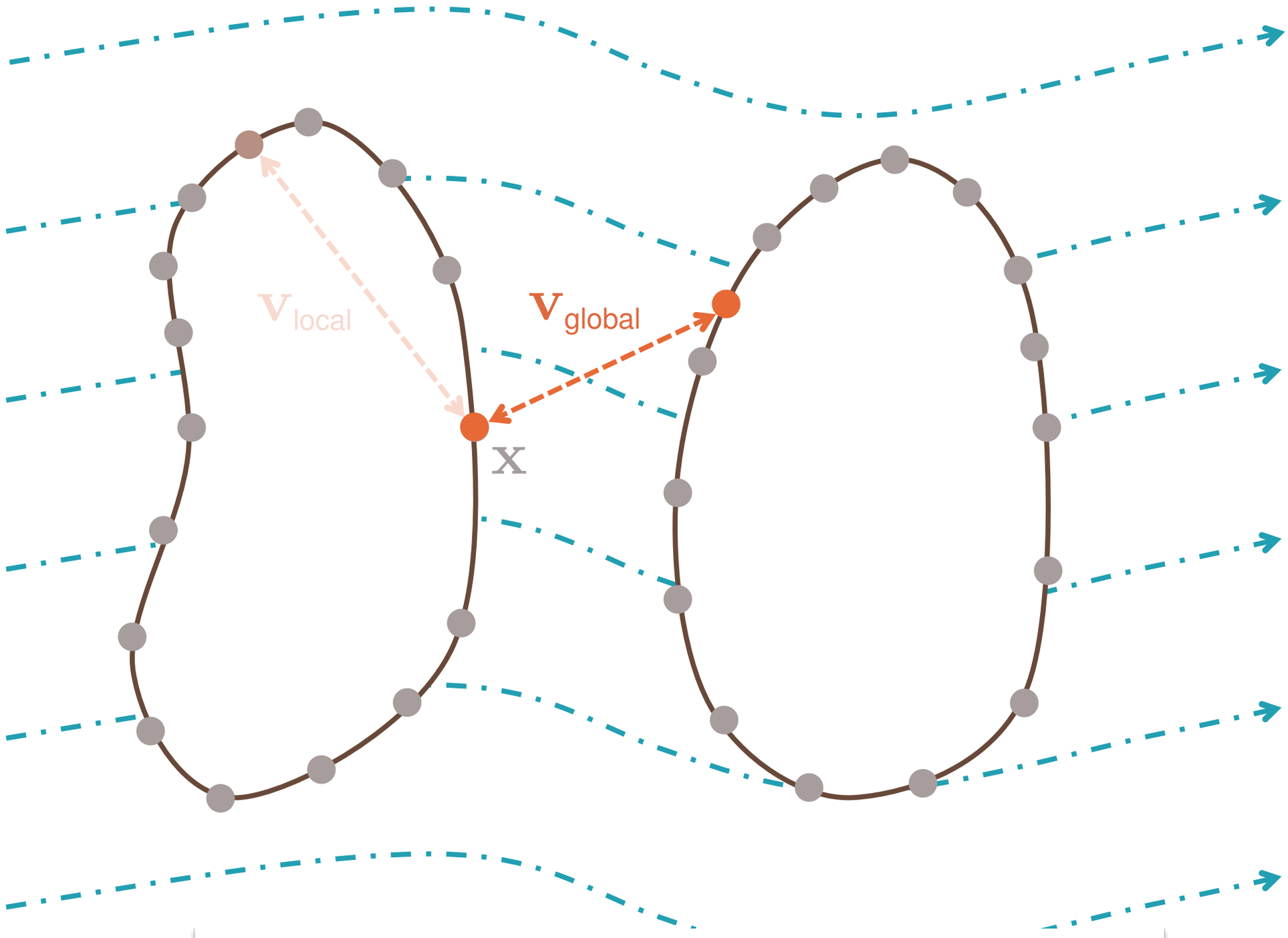


Kent Czechowski <kentcz@gatech.edu>

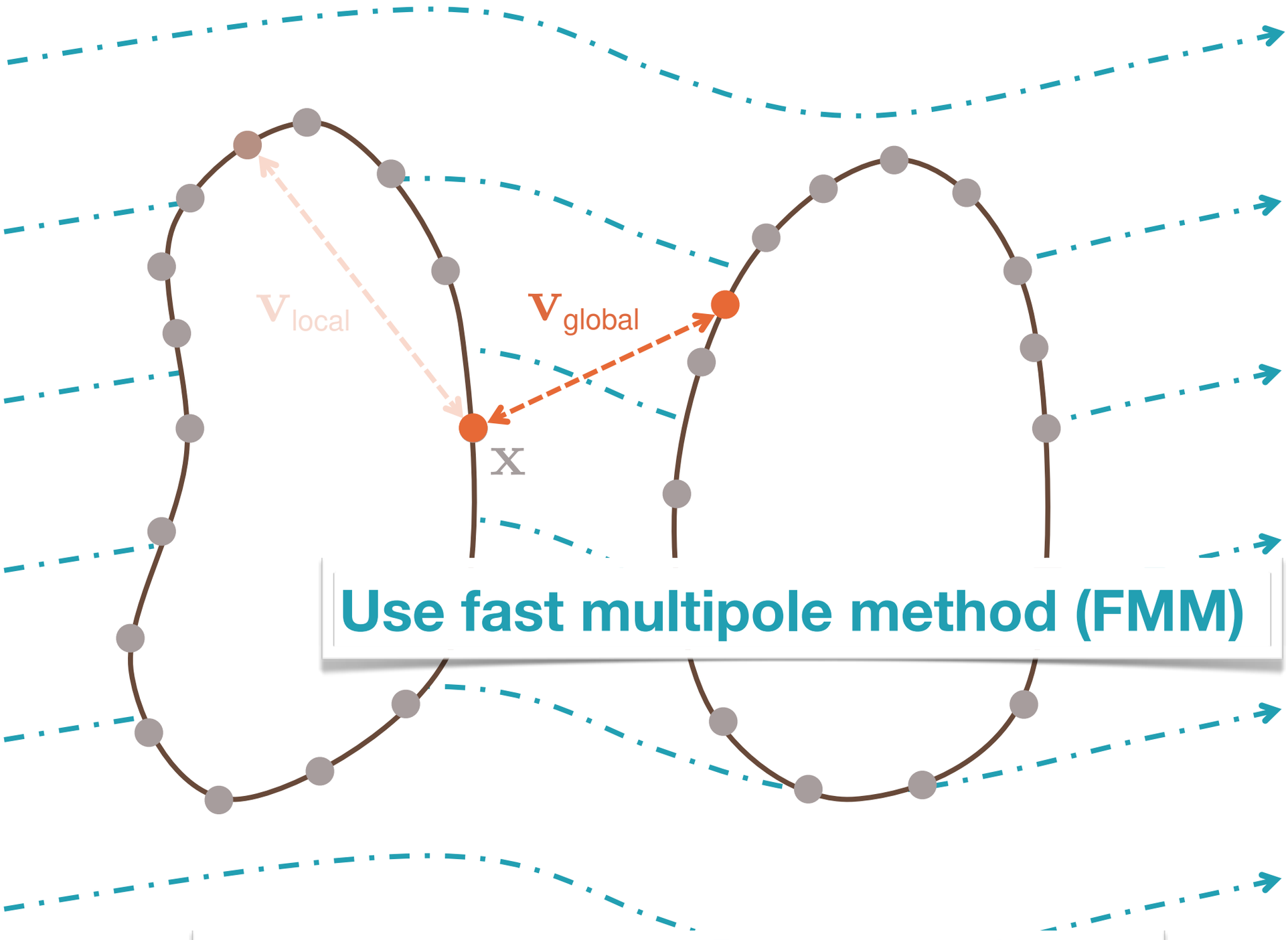




Peng Wang & Steve Parker @ NVIDIA



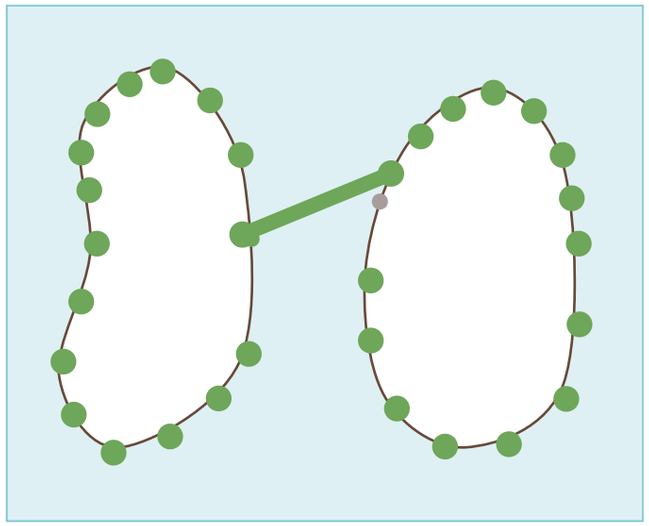
$$\mathbf{v}_{\text{interaction}}(\mathbf{x}) = \mathbf{v}_{\text{local}}(\mathbf{x}) + \mathbf{v}_{\text{global}}(\mathbf{x})$$



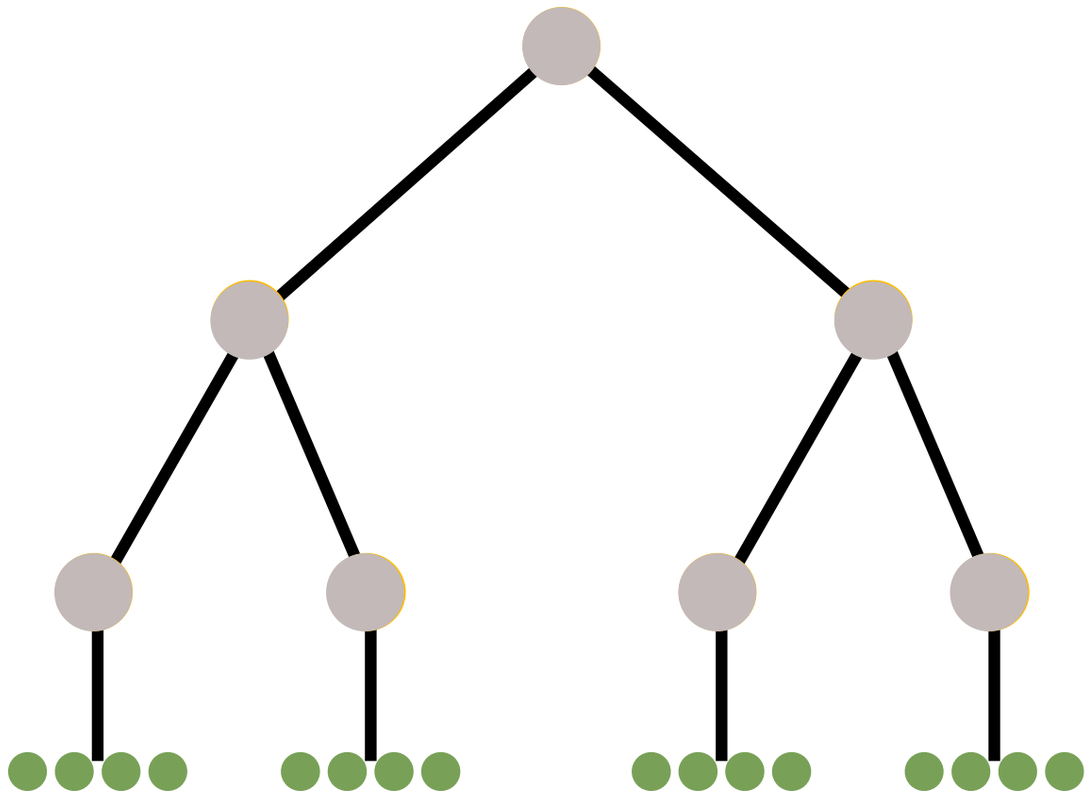
Use fast multipole method (FMM)

$$\mathbf{v}_{\text{interaction}}(\mathbf{x}) = \mathbf{v}_{\text{local}}(\mathbf{x}) + \mathbf{v}_{\text{global}}(\mathbf{x})$$

FMM

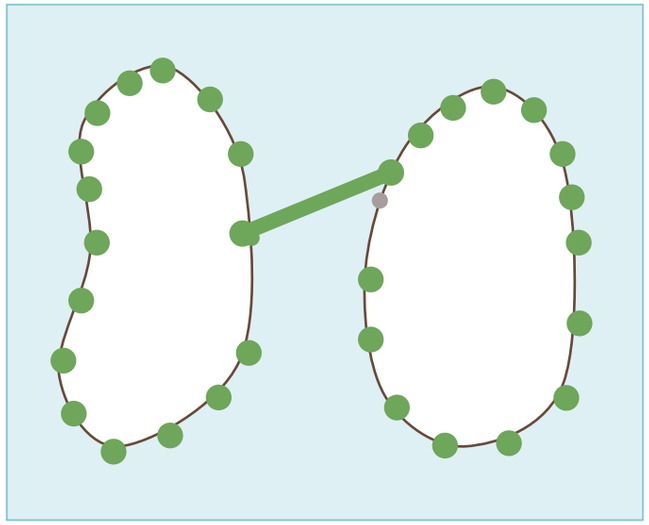


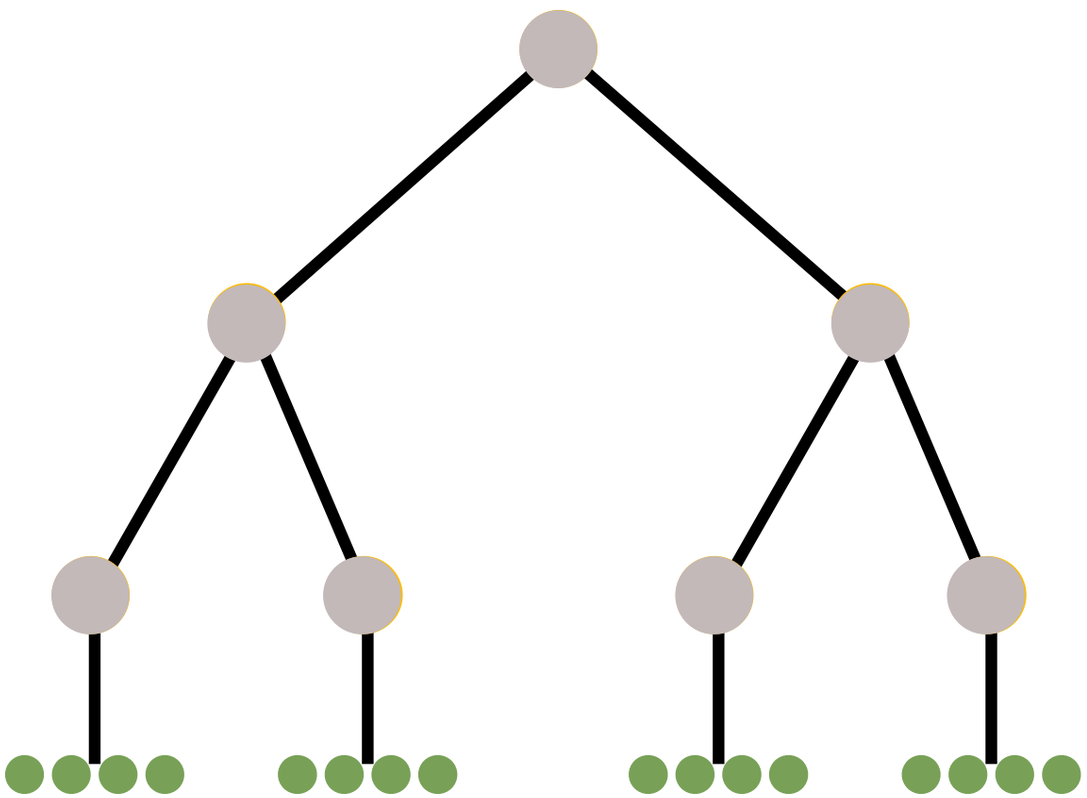
- Need to evaluate all-pairwise interactions among green points: **$O(n^2)$**



- FMM instead computes in **O(n)** time, with an approximation guarantee, using a tree (**n log n** to build)

Distributed Memory Algorithm





- We use a *kernel-independent* variant, but the structure and parallelization are same as classical case [Warren & Salmon (1993)]
- "Control" adaptivity using 2:1 balancing [Sampath et al. (2008)]

Summary: Basic Algorithm

- Given: n RBCs, where the k-th RBC is represented by the set γ_k of its surface points
- Loop over time steps (multistep):
  - **Parallel-for $k \leftarrow 1{:}n$, compute $v_{\text{local}}(x)$ for all x in γ_k**
  - **Compute $v_{\text{global}}(x)$ using the FMM**
  - Evaluate $v_{\text{background}}(x)$ analytically
  - Update positions (semi-implicit)
  - Periodically load re-balance (repartition)

Complexity Estimates

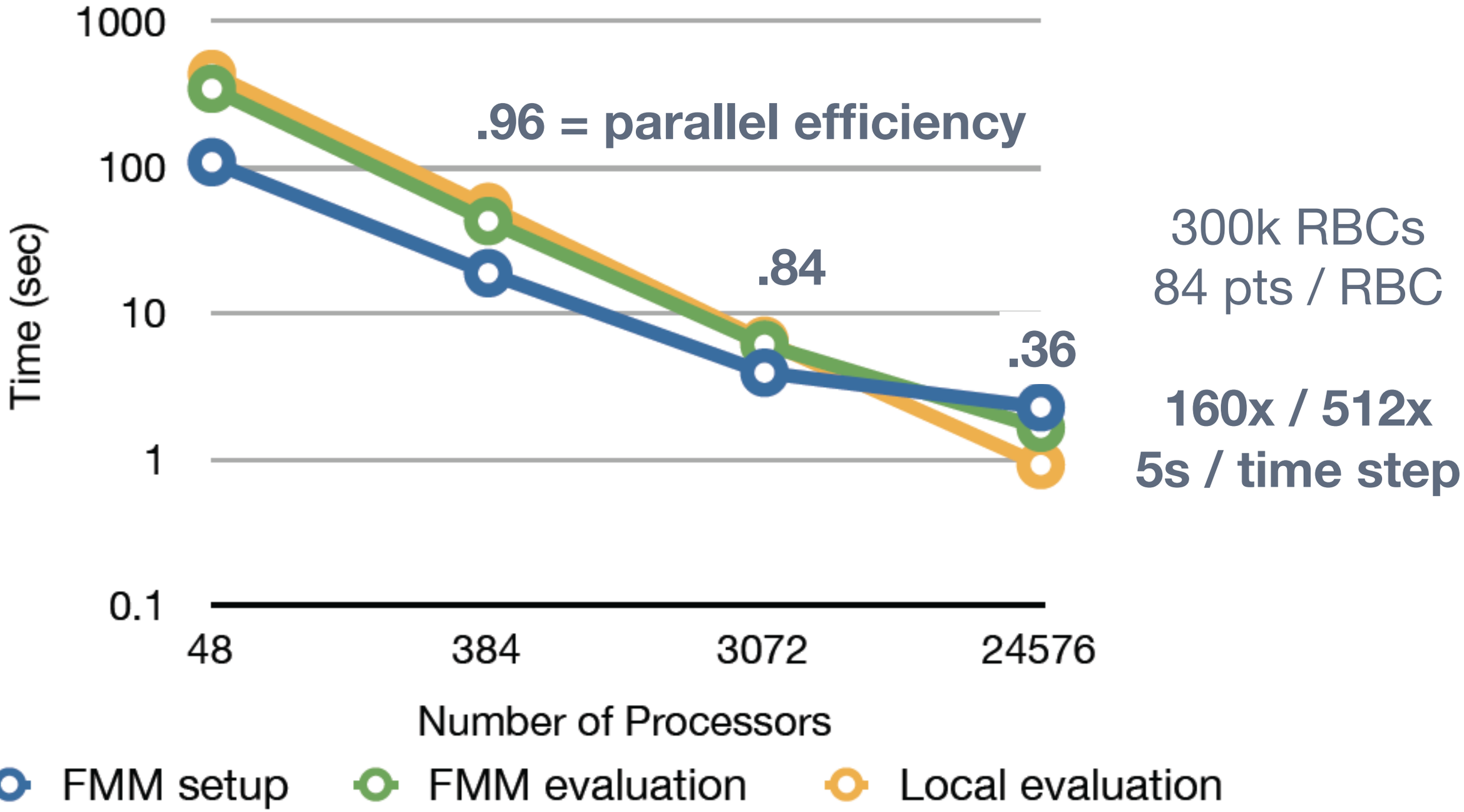
$\mathbf{v}_{\text{local}}$: $$\mathcal{O}\left(\eta^{\frac{3}{2}} \cdot \frac{n}{p}\right)$$

$\eta =$ points / cell, $n =$ total points

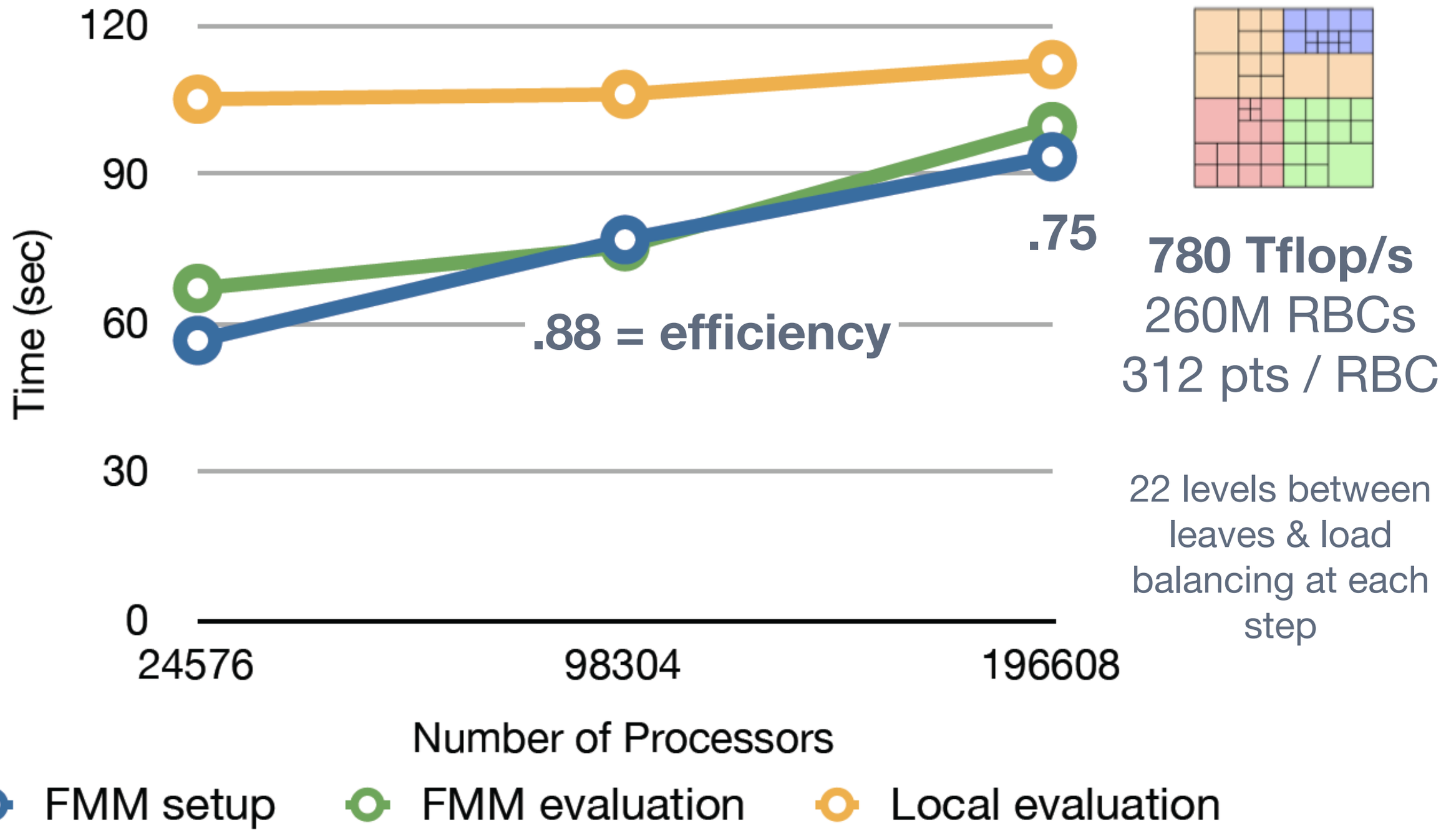
$\mathbf{v}_{\text{global}}$, build tree : $$\mathcal{O}\left(\frac{n}{p}\log\frac{n}{p} + \sqrt{p}\left(\frac{n}{p}\right)^{\frac{2}{3}} + p\log^2 p\right)$$

$\mathbf{v}_{\text{global}}$, evaluation : $$\mathcal{O}\left(\frac{n}{p} + \sqrt{p}\left(\frac{n}{p}\right)^{\frac{2}{3}}\right)$$

STRONG SCALING: JAGUAR



Weak Scaling: Jaguar

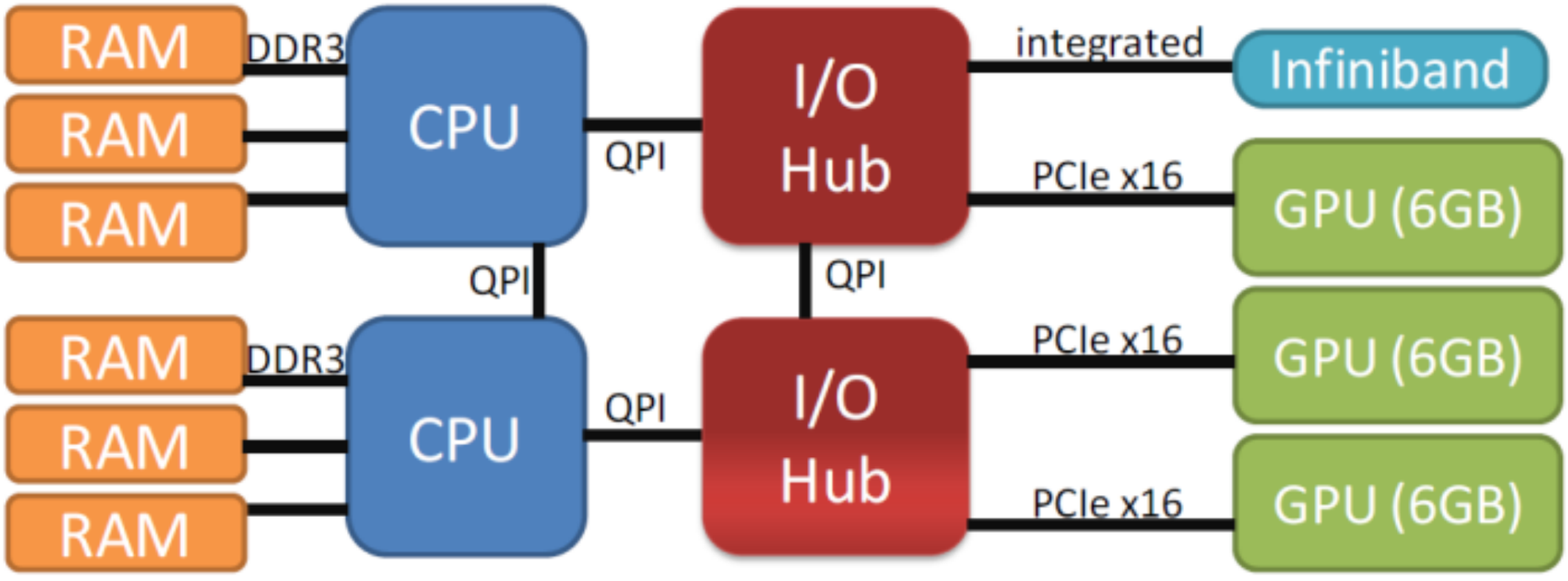


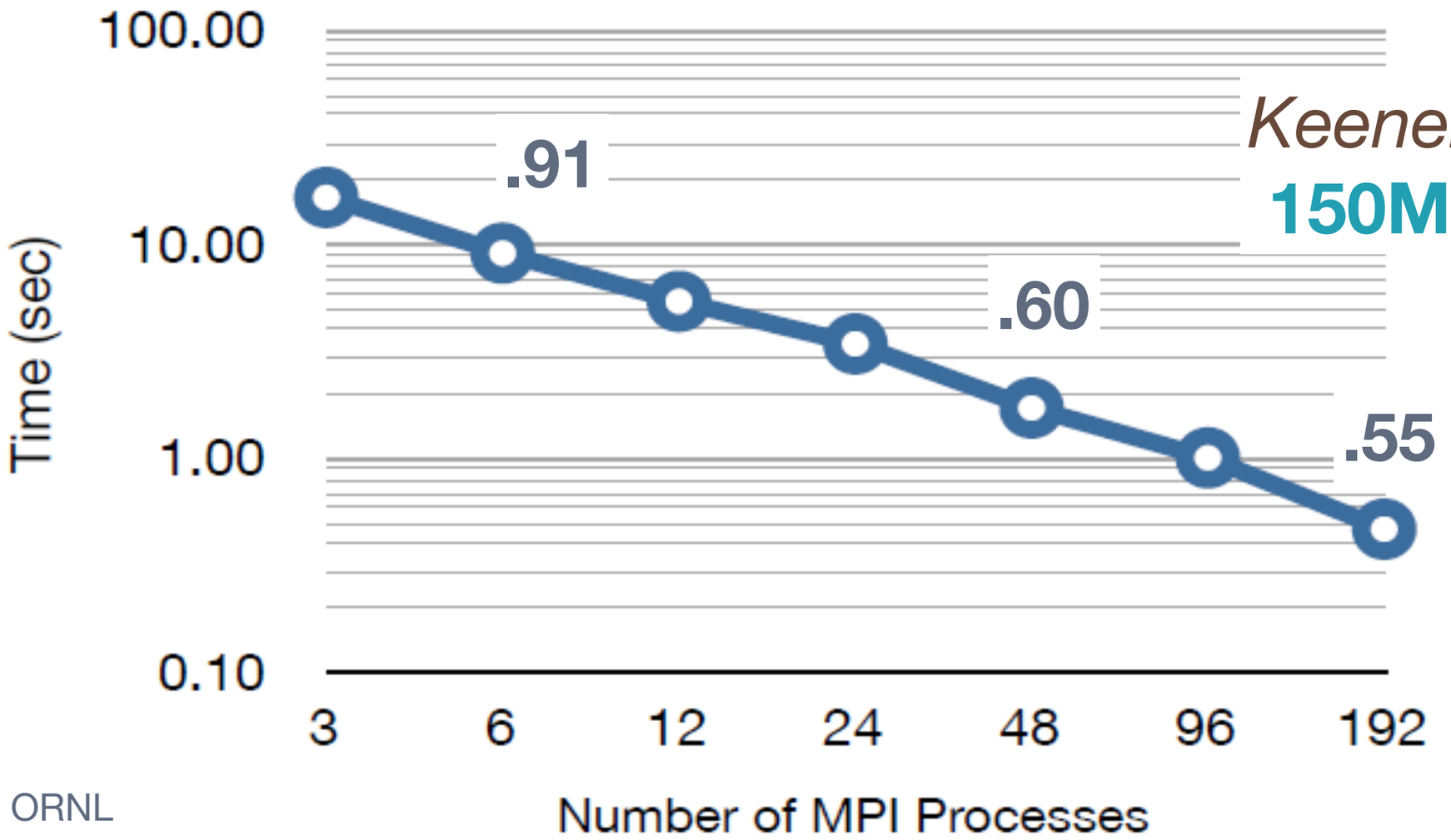
780 Tflop/s

260M RBCs

312 pts / RBC

22 levels between leaves & load balancing at each step





Keeneland (ORNL)

150M pts in 0.4s

Phil Roth @ ORNL

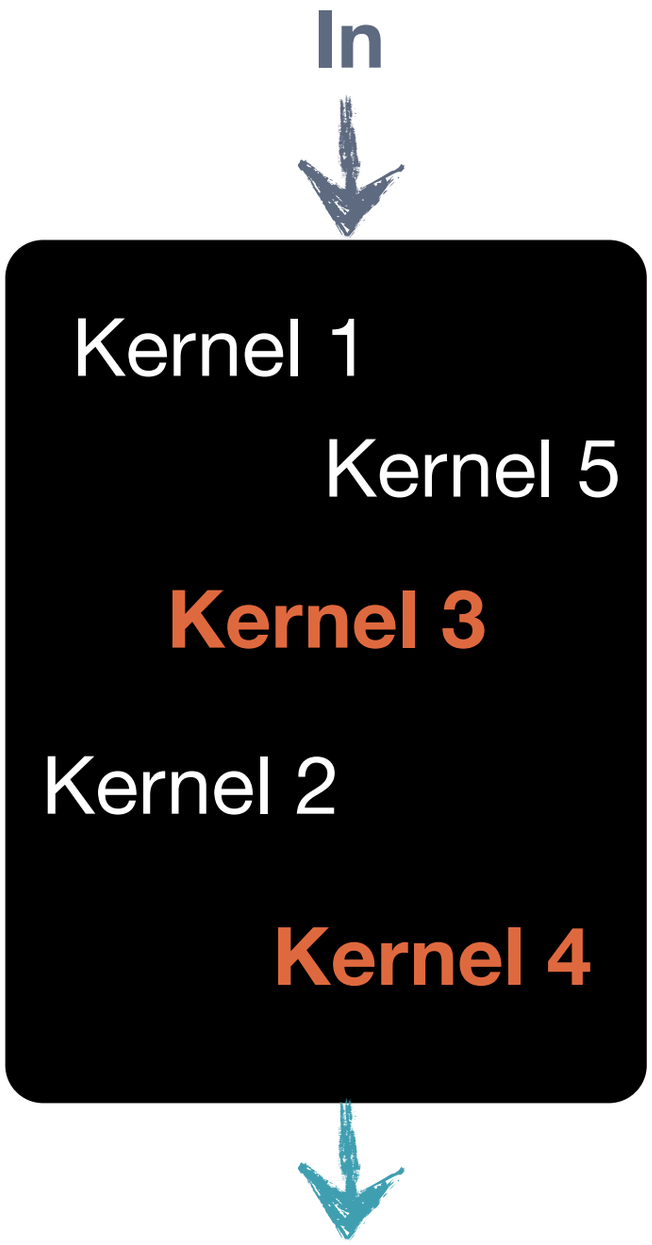
LIMITATIONS

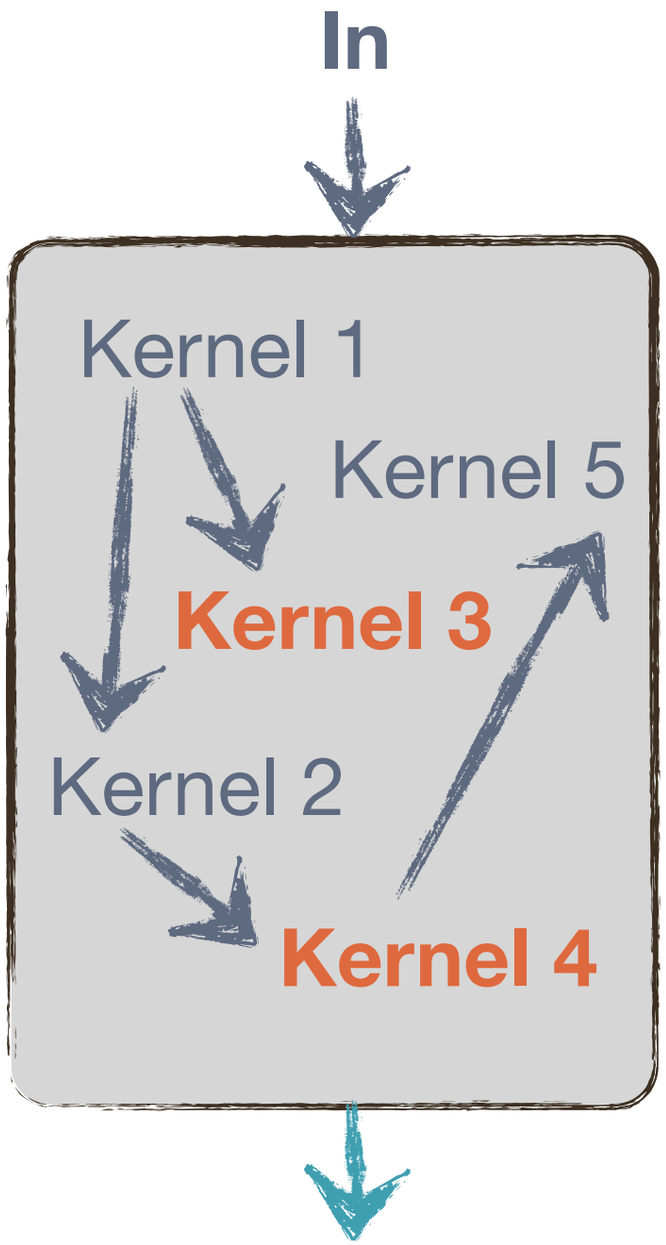
- Physics
  - Free boundaries*
  - Low volume fraction*
  - Newtonian fluid model
  - Low Reynolds numbers
- Algorithms
  - Large memory requirement for v_{local} : ~ η^3
  - No shared memory parallelization in the tree construction*
  - Need scalable data analysis*

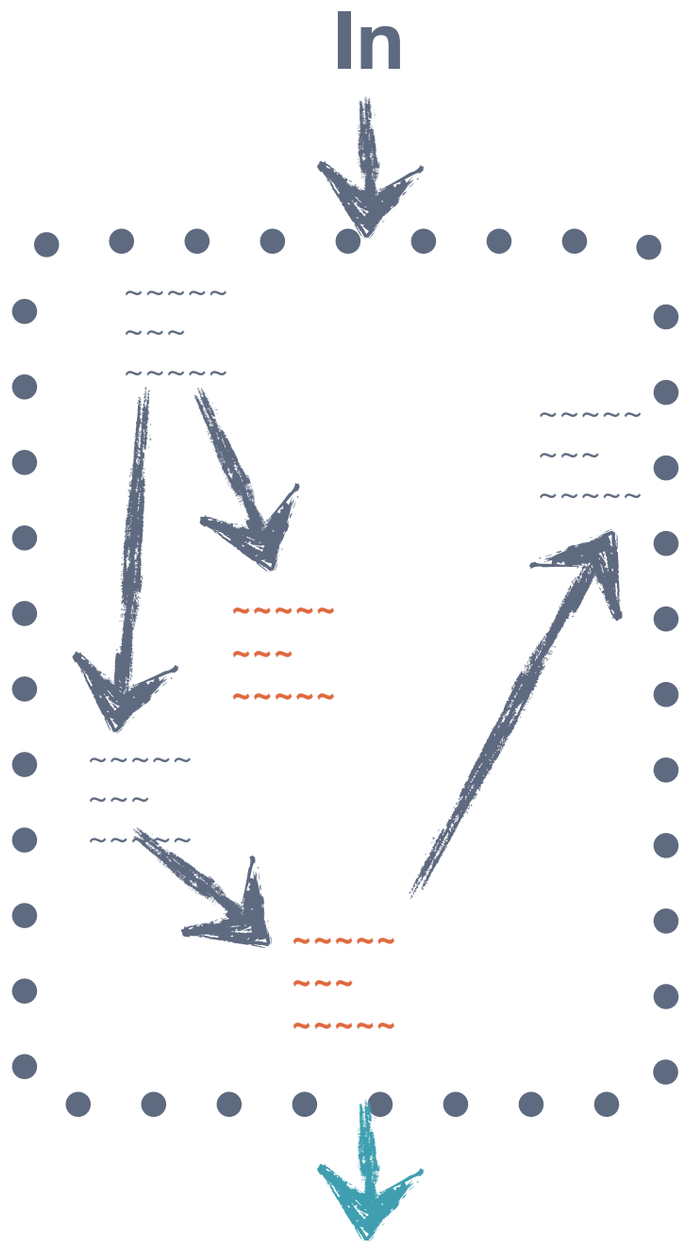
* Mathematical fix but no parallel implementation or fix in progress

- Intra-node tuning of the FMM

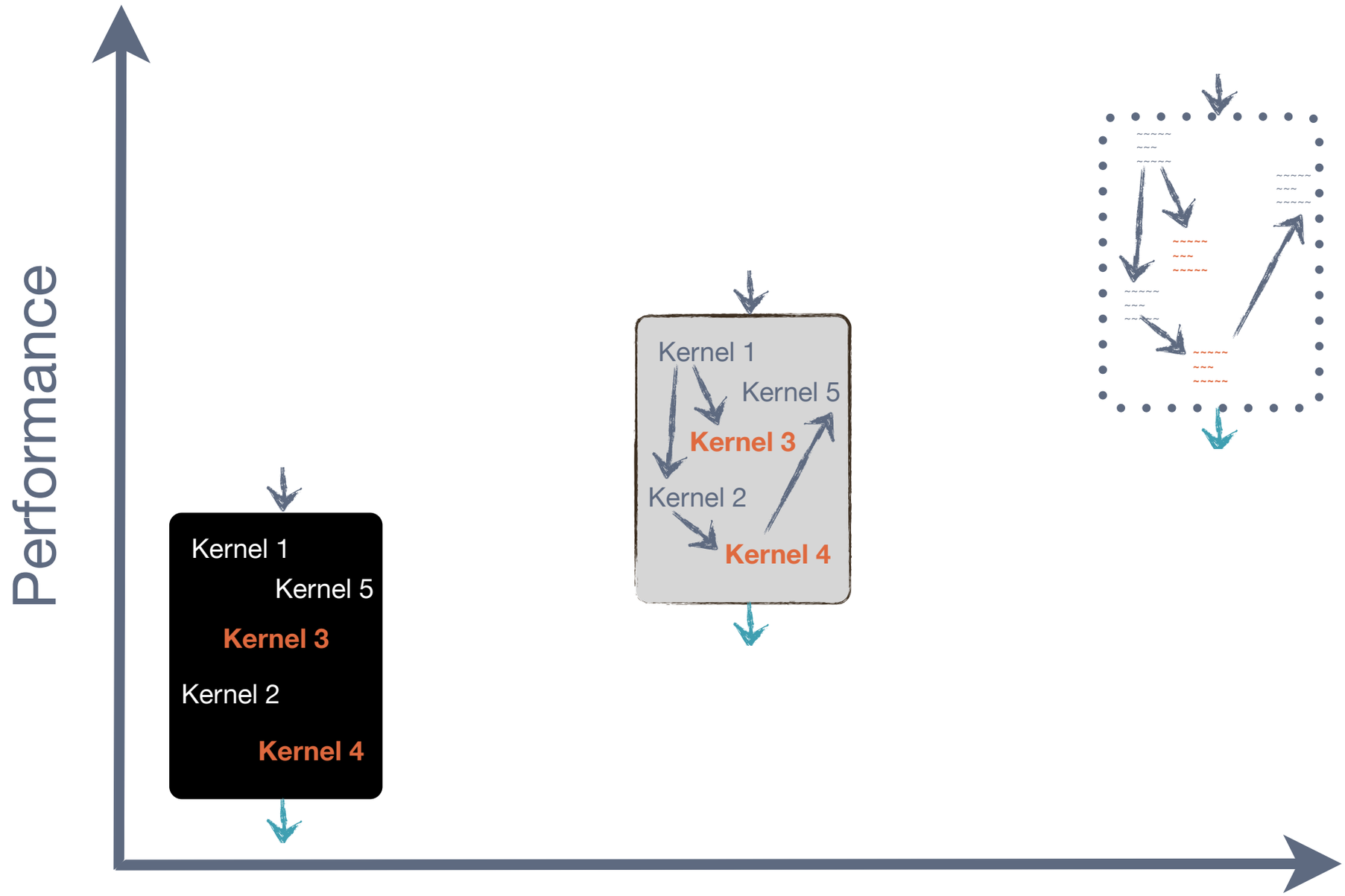
  A. Chandramowlishwaran, K. Madduri, R. Vuduc. “Diagnosis, tuning, and redesign for multicore performance: A case study of the FMM.” (SC’10)
  http://dx.doi.org/10.1109/SC.2010.19





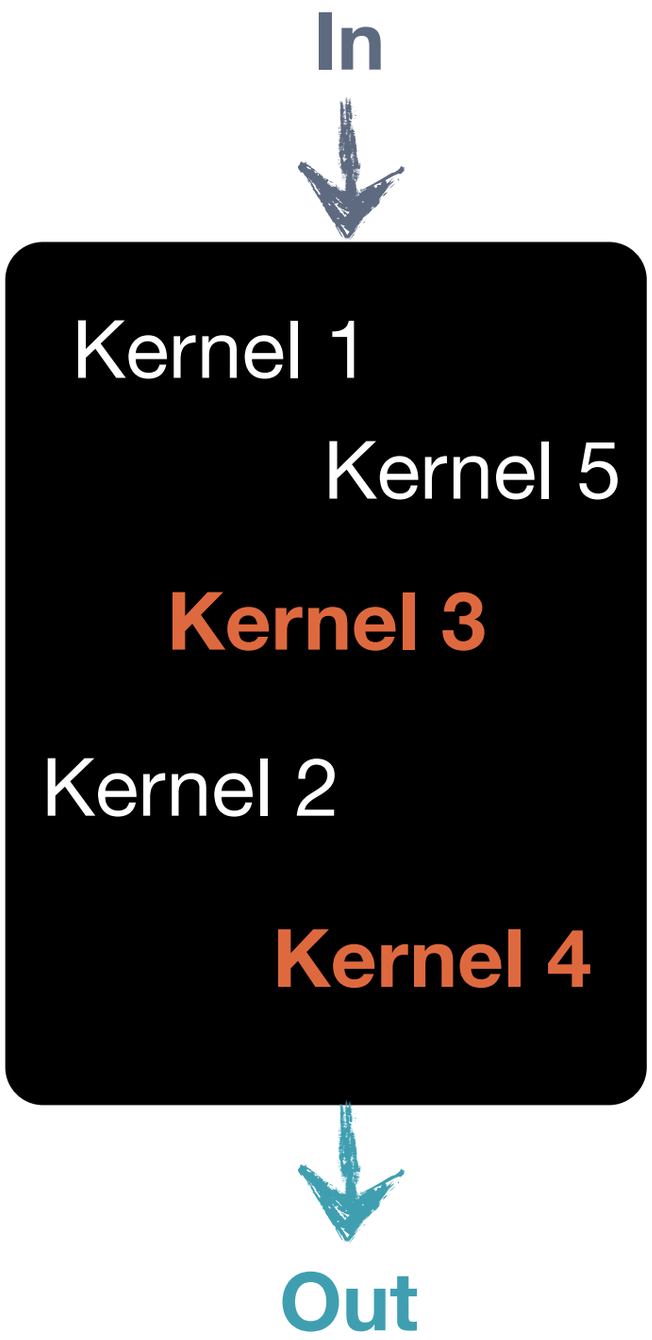


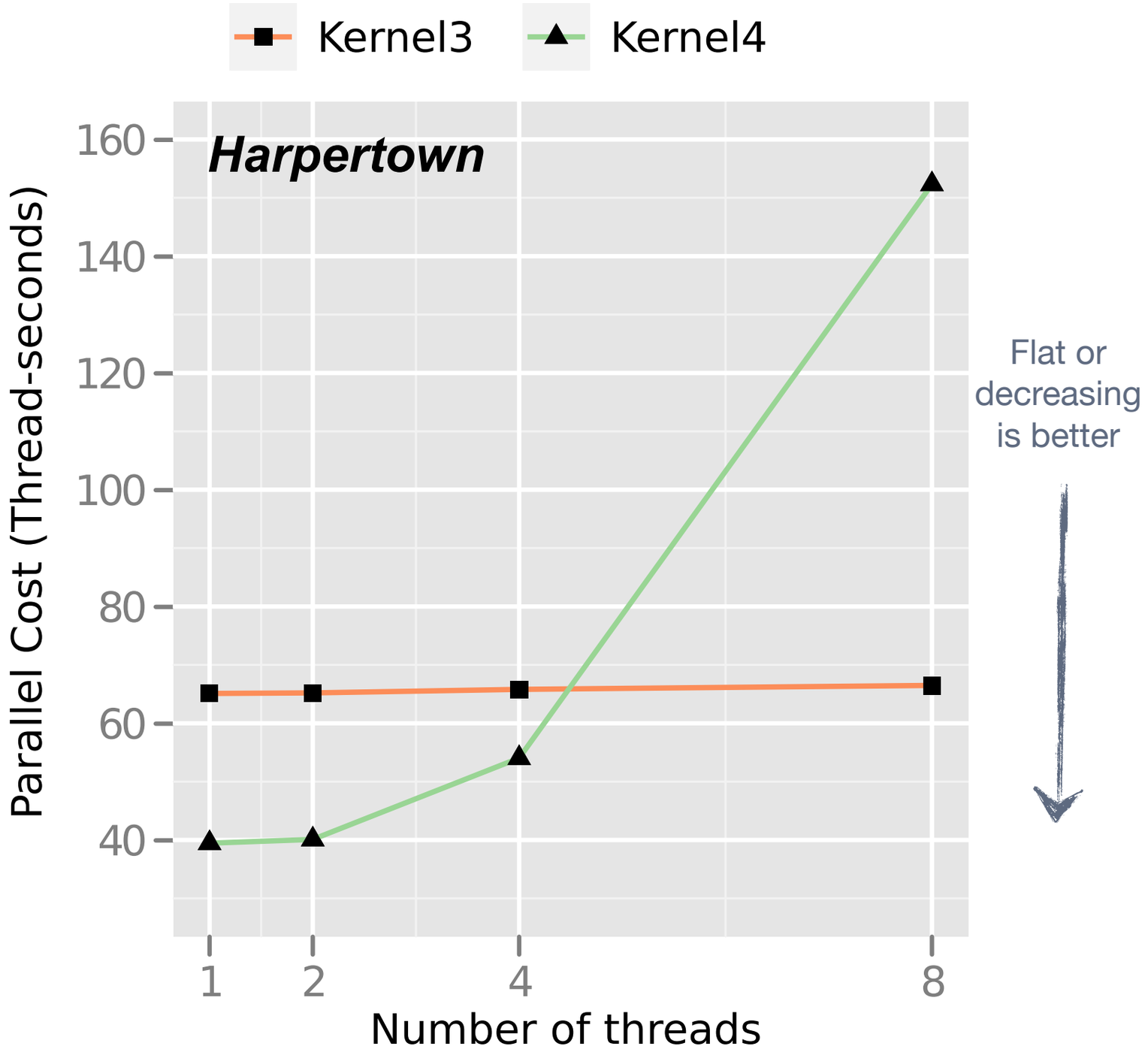
How to find and fix bottlenecks?



A NOTIONAL TUNING PROCESS

Achieved performance increases as we increase our knowledge of the details of the application, from “black box” to “full knowledge.”

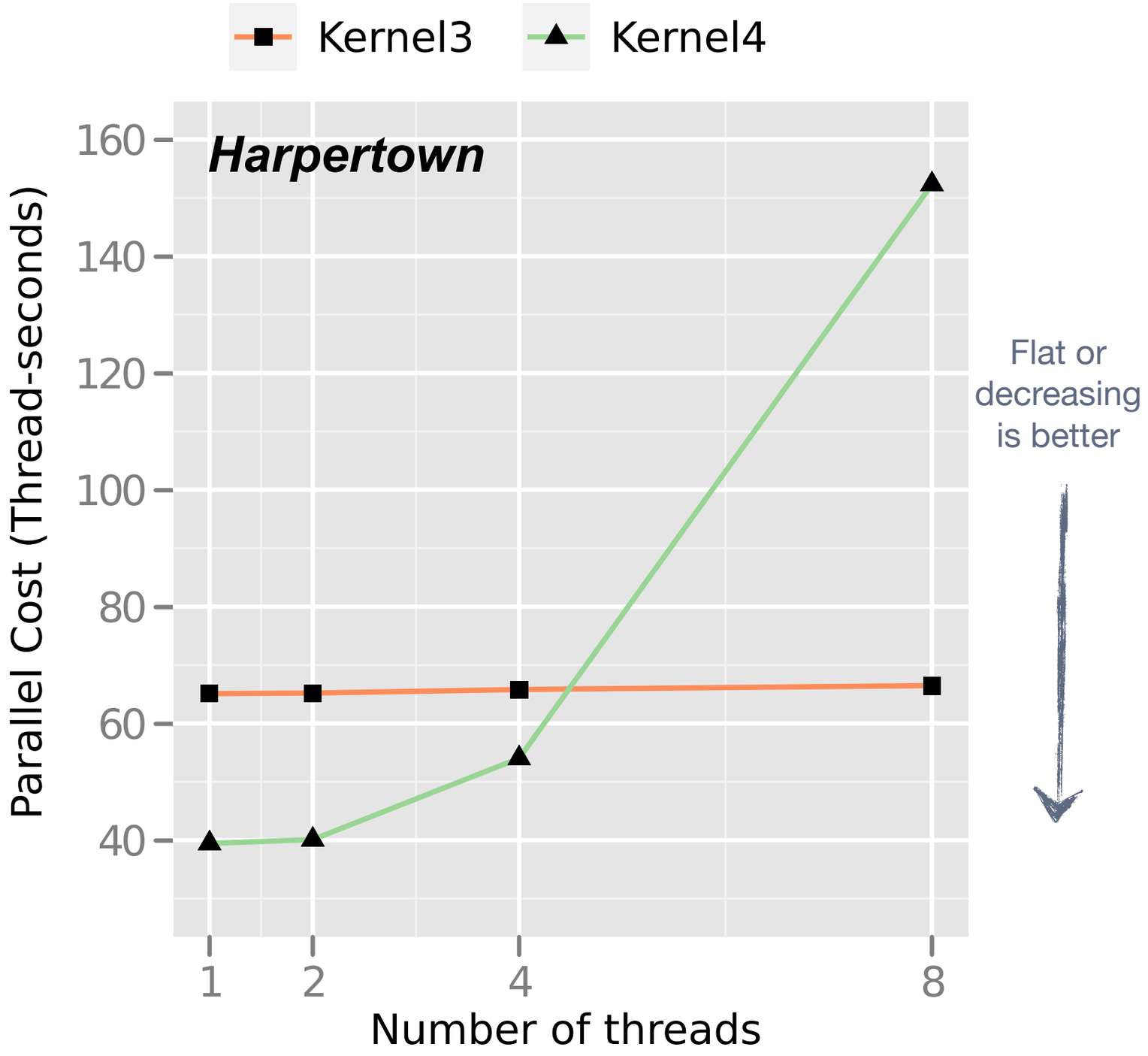




STAGE 1: BLACK BOX

Assume simple profiling and no code changes.

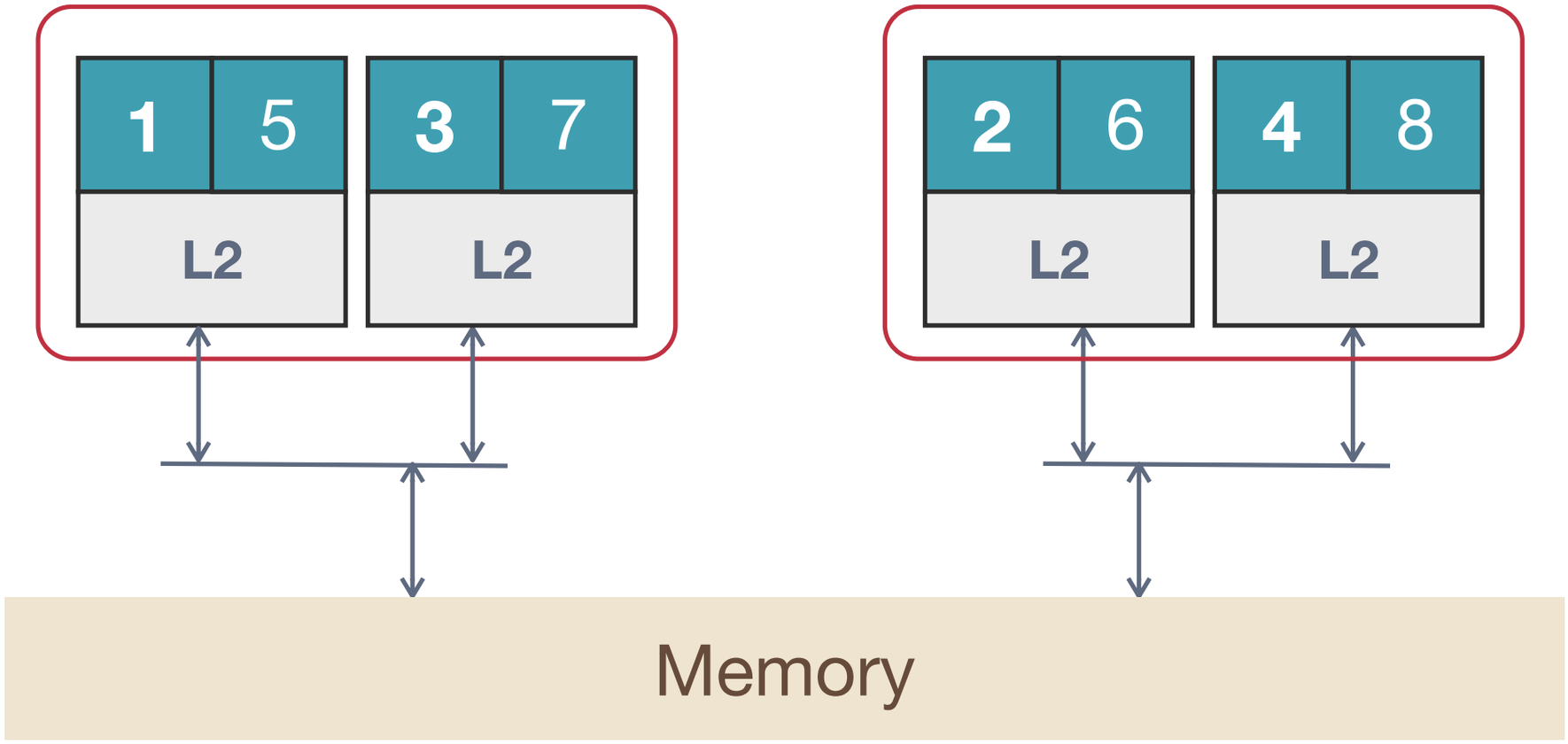
- Observe: K4 scales poorly
- Measure intensity
  - K3 = compute-bound
  - K4 = memory-bound
- Hypotheses:
  - Load imbalance?
  - Memory contention? Cache or bandwidth?



STAGE 1: BLACK BOX

Assume simple profiling and no code changes.

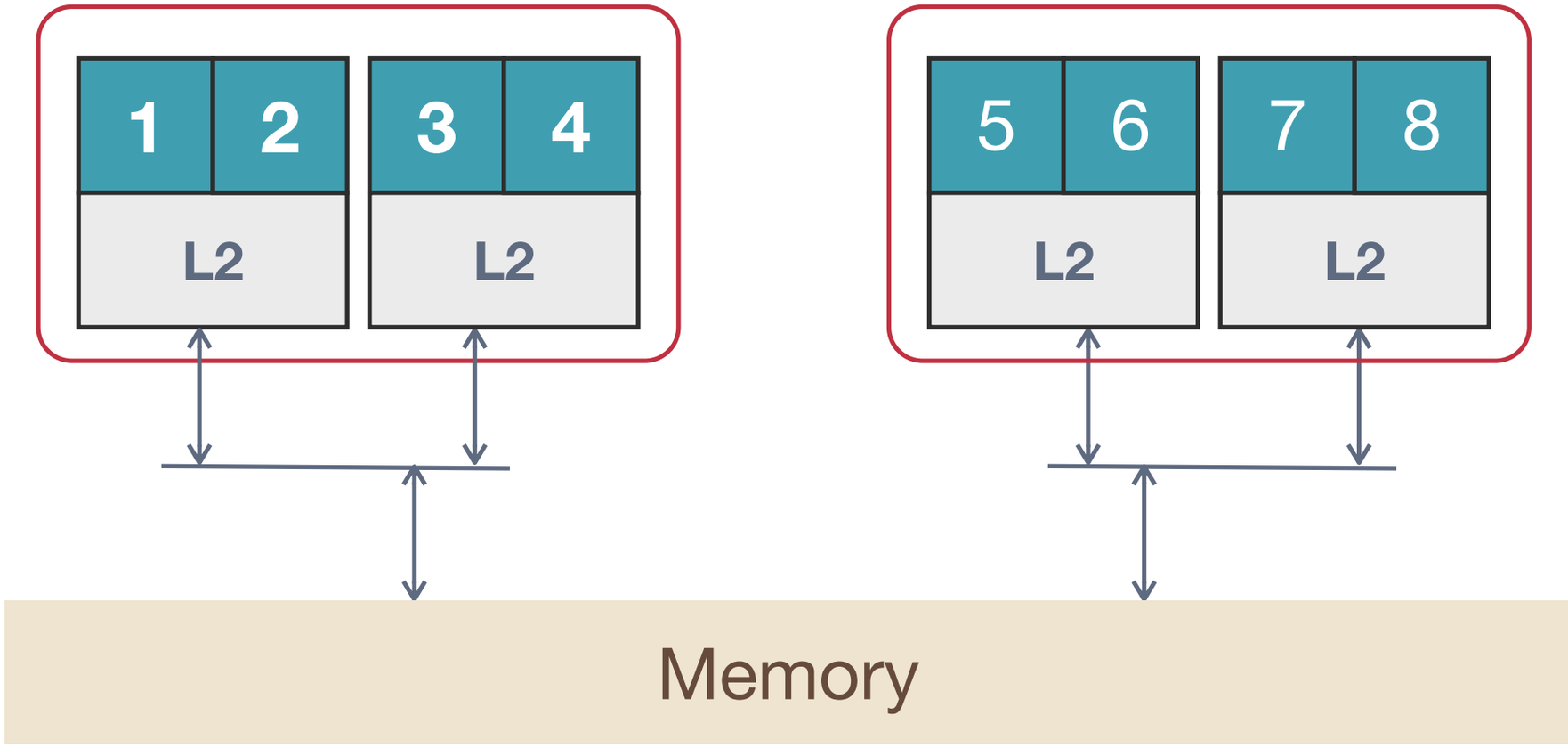
Intel Harpertown



threads = 8

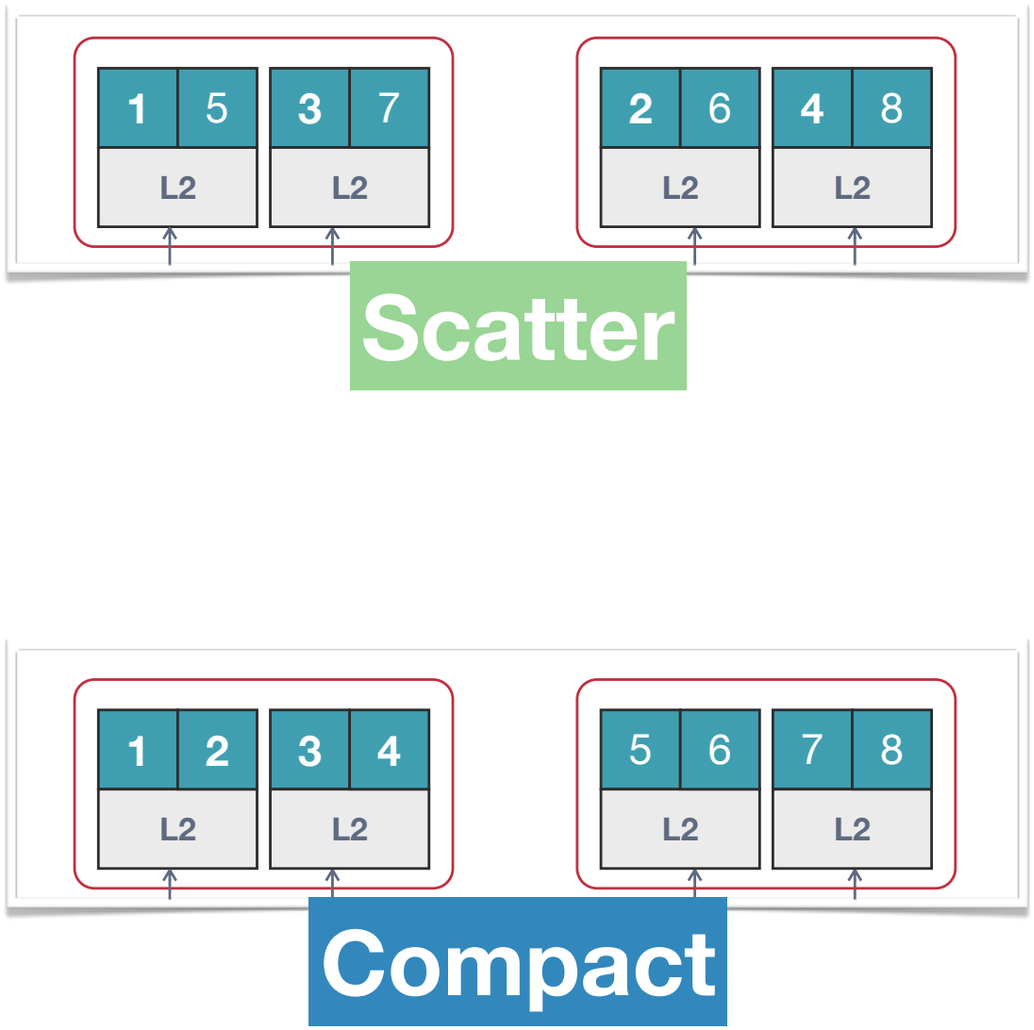
OpenMP *scatter*

Intel Harpertown



threads = 8

OpenMP *compact*

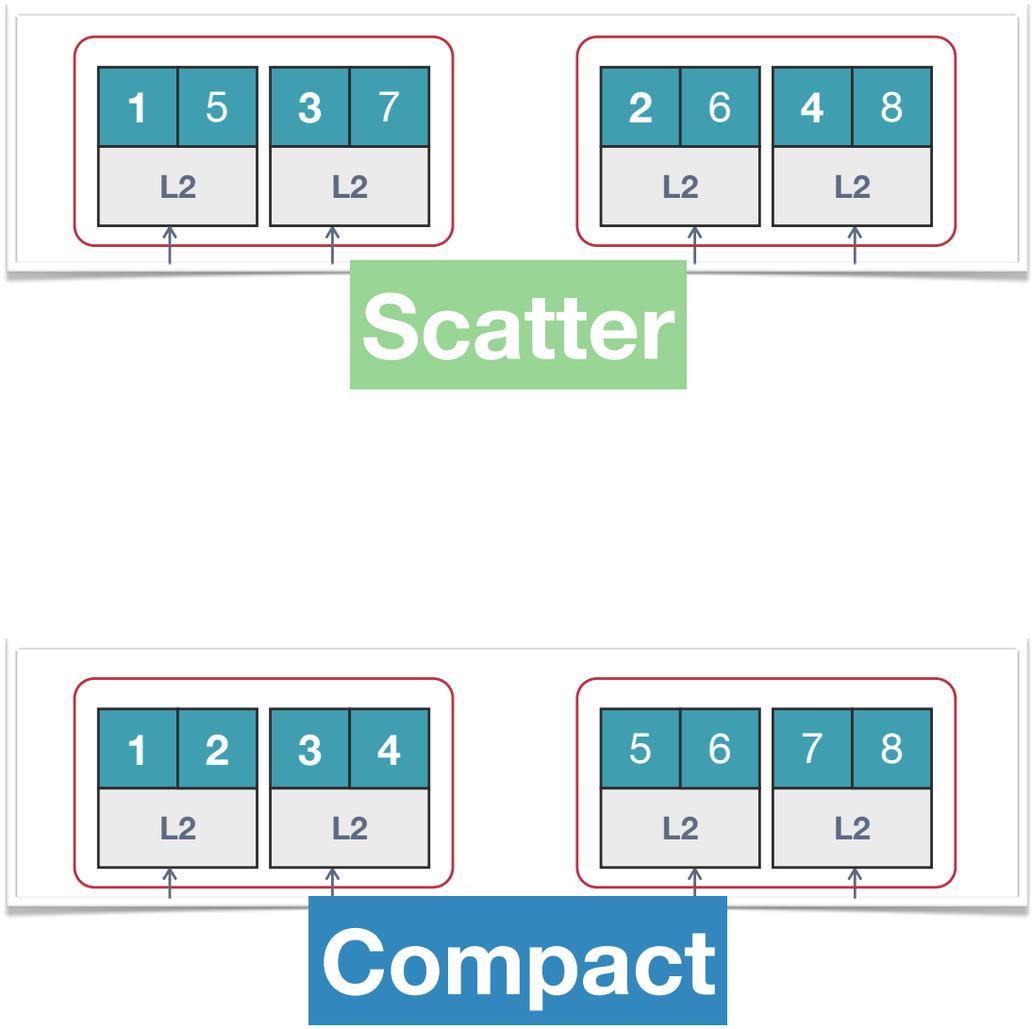


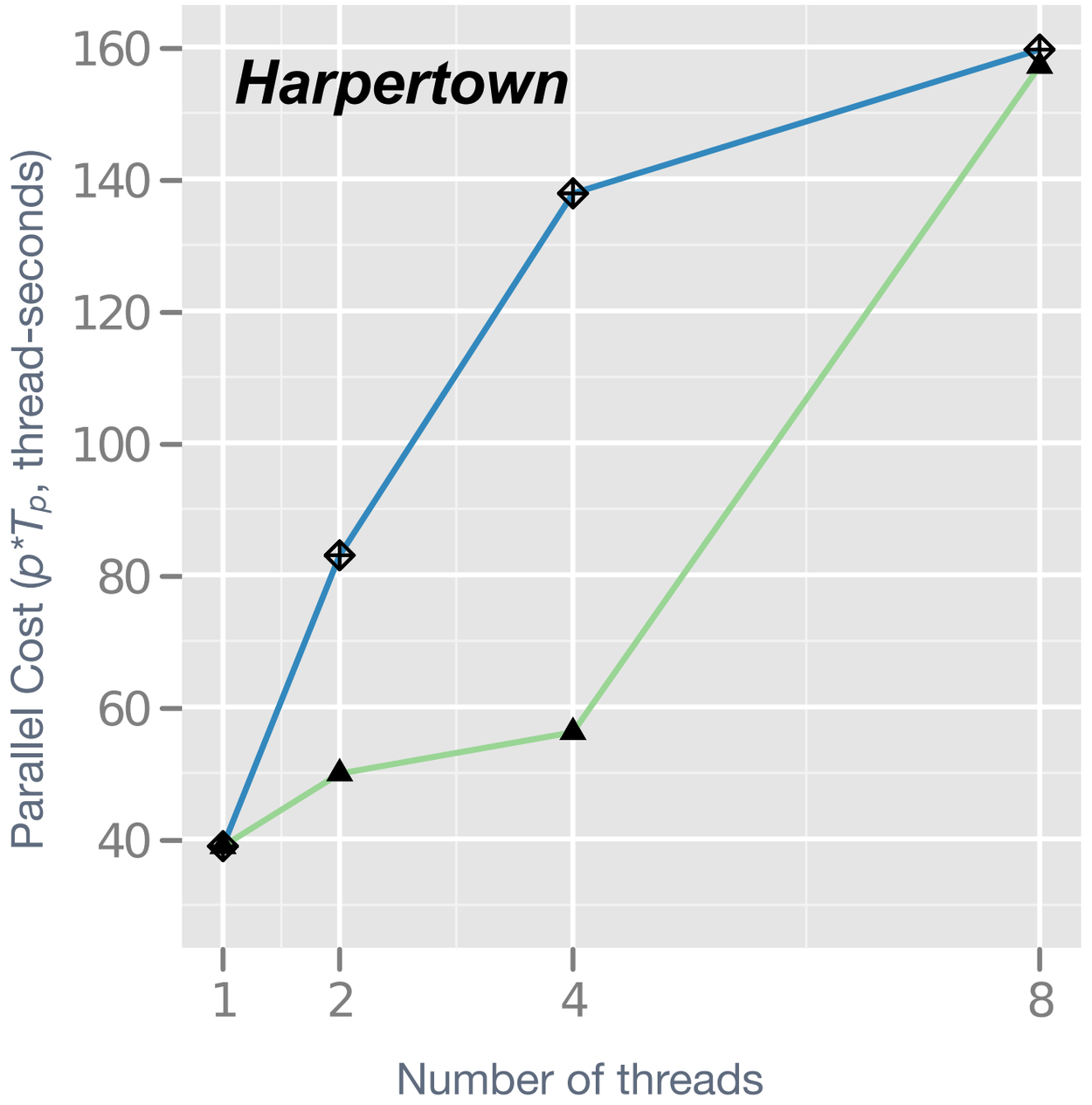
Parallel Cost ($p*T_p$, thread-seconds)

Number of threads

VARYING THREAD BINDING STRATEGY

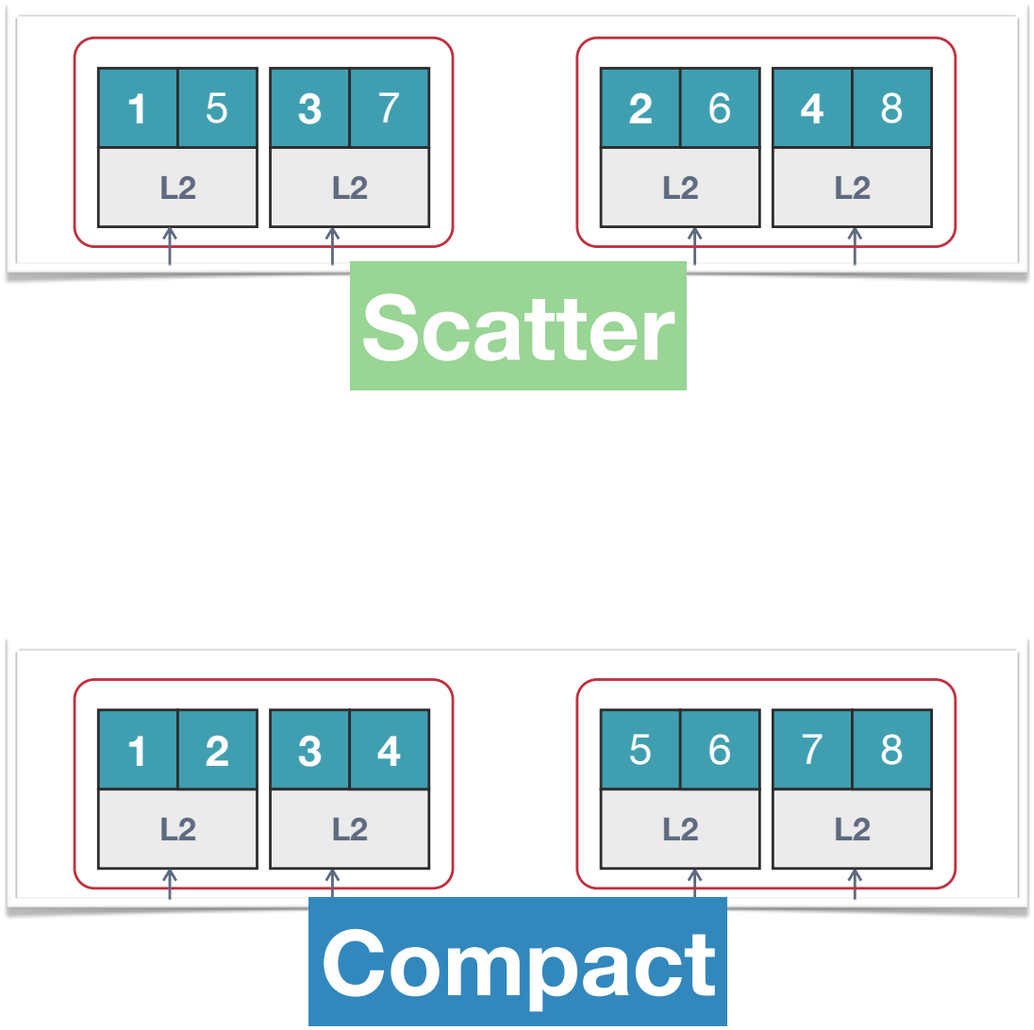
Assume simple profiling and no code changes.

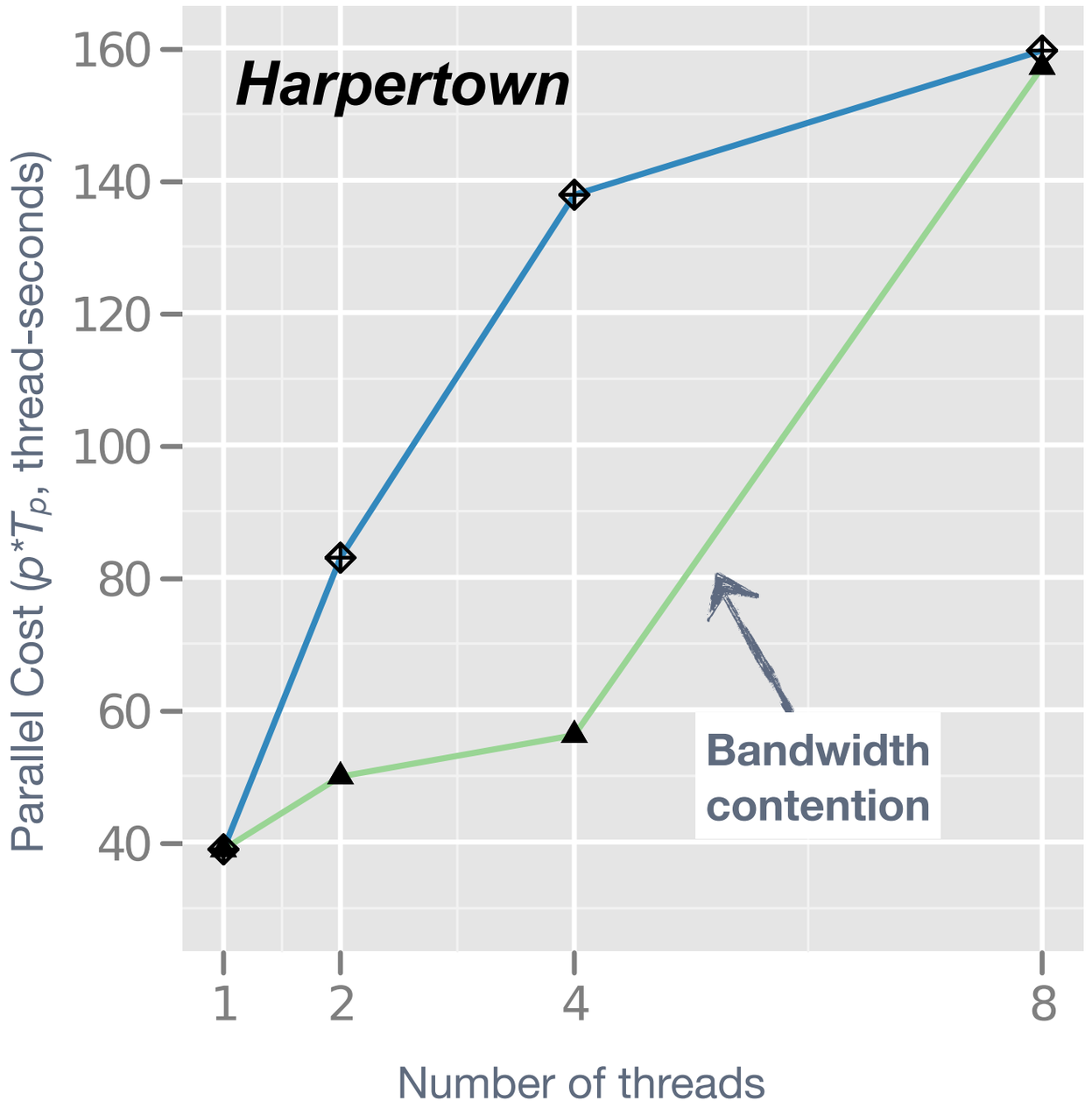




Varying Thread Binding Strategy

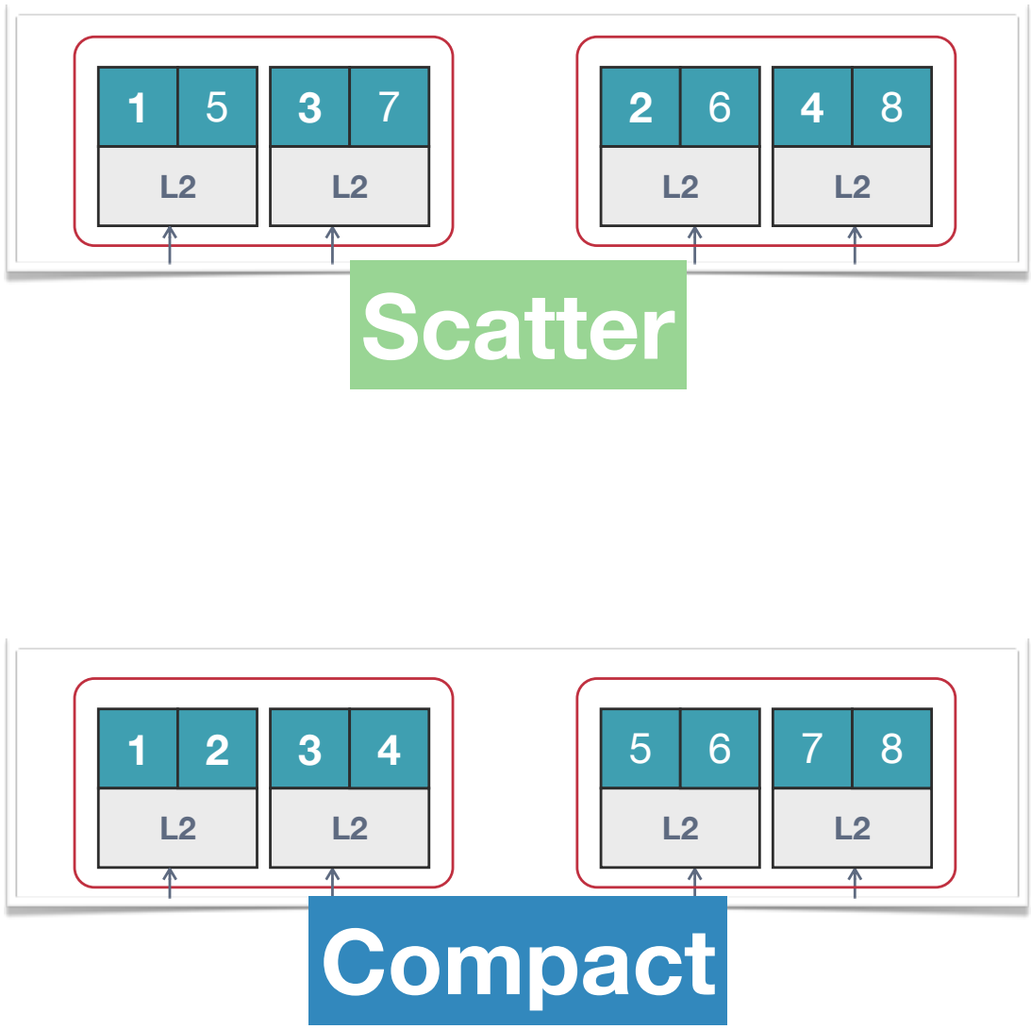
Assume simple profiling and no code changes.

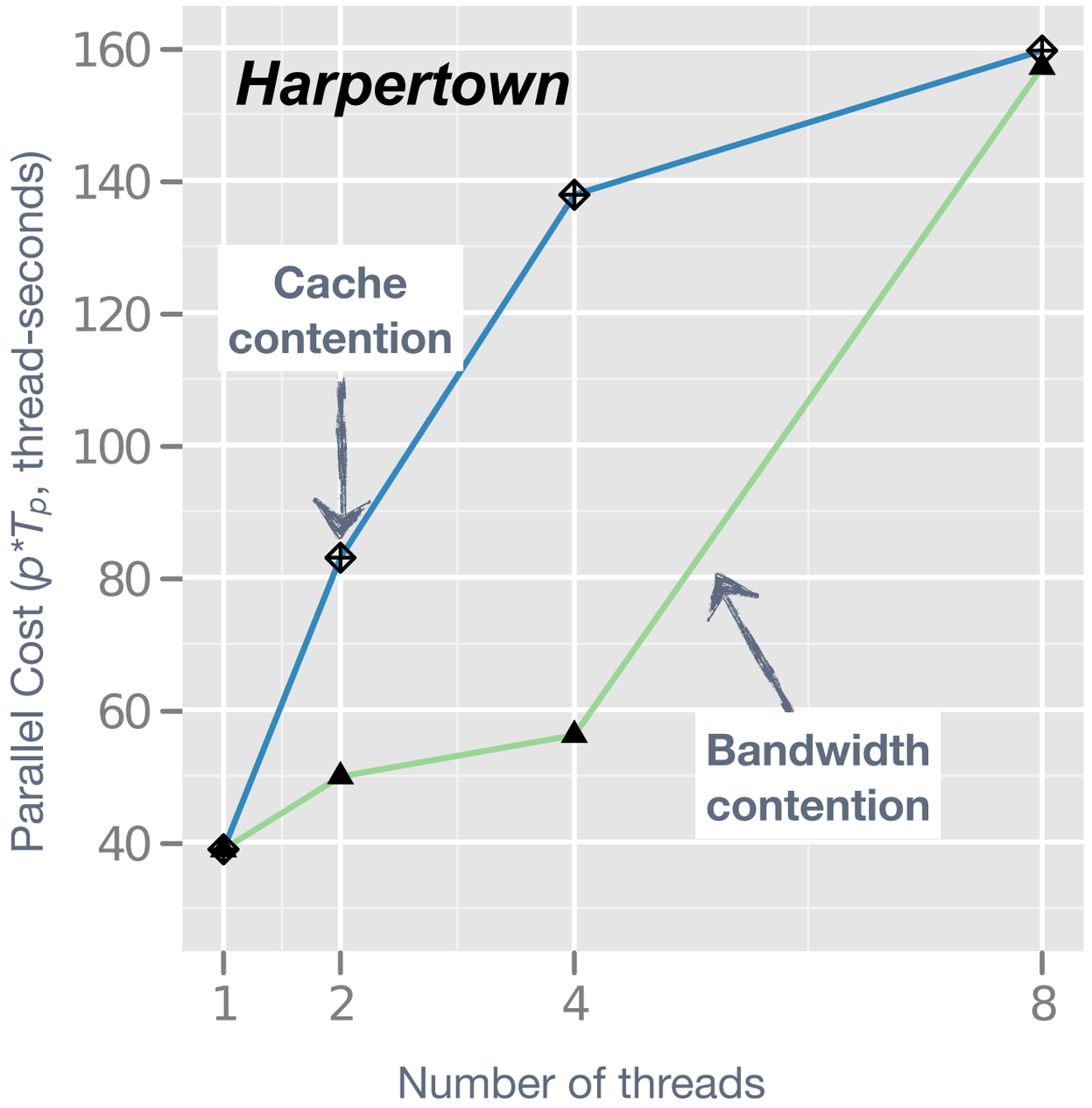




Varying Thread Binding Strategy

Assume simple profiling and no code changes.

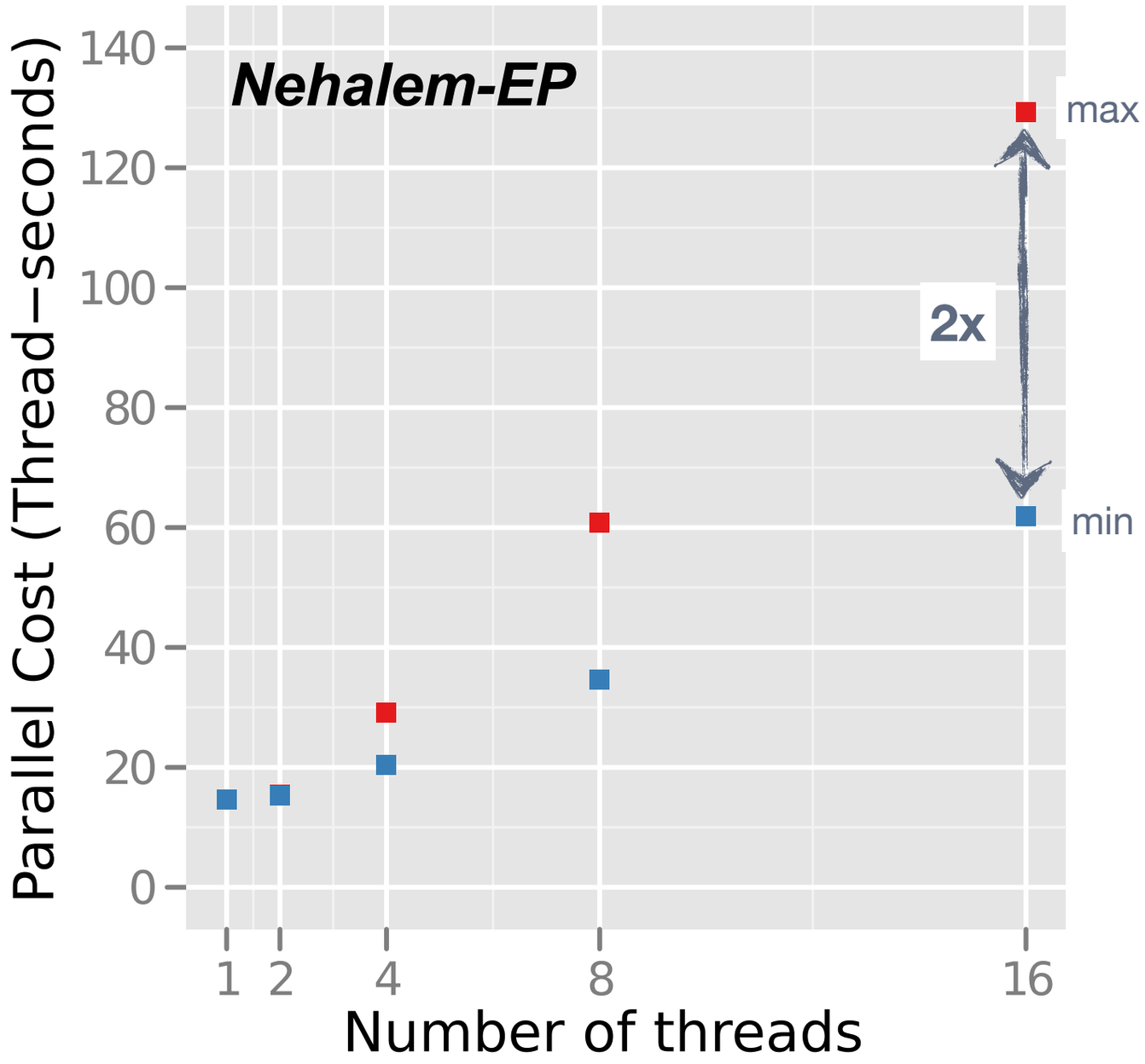




Varying Thread Binding Strategy

Assume simple profiling and no code changes.

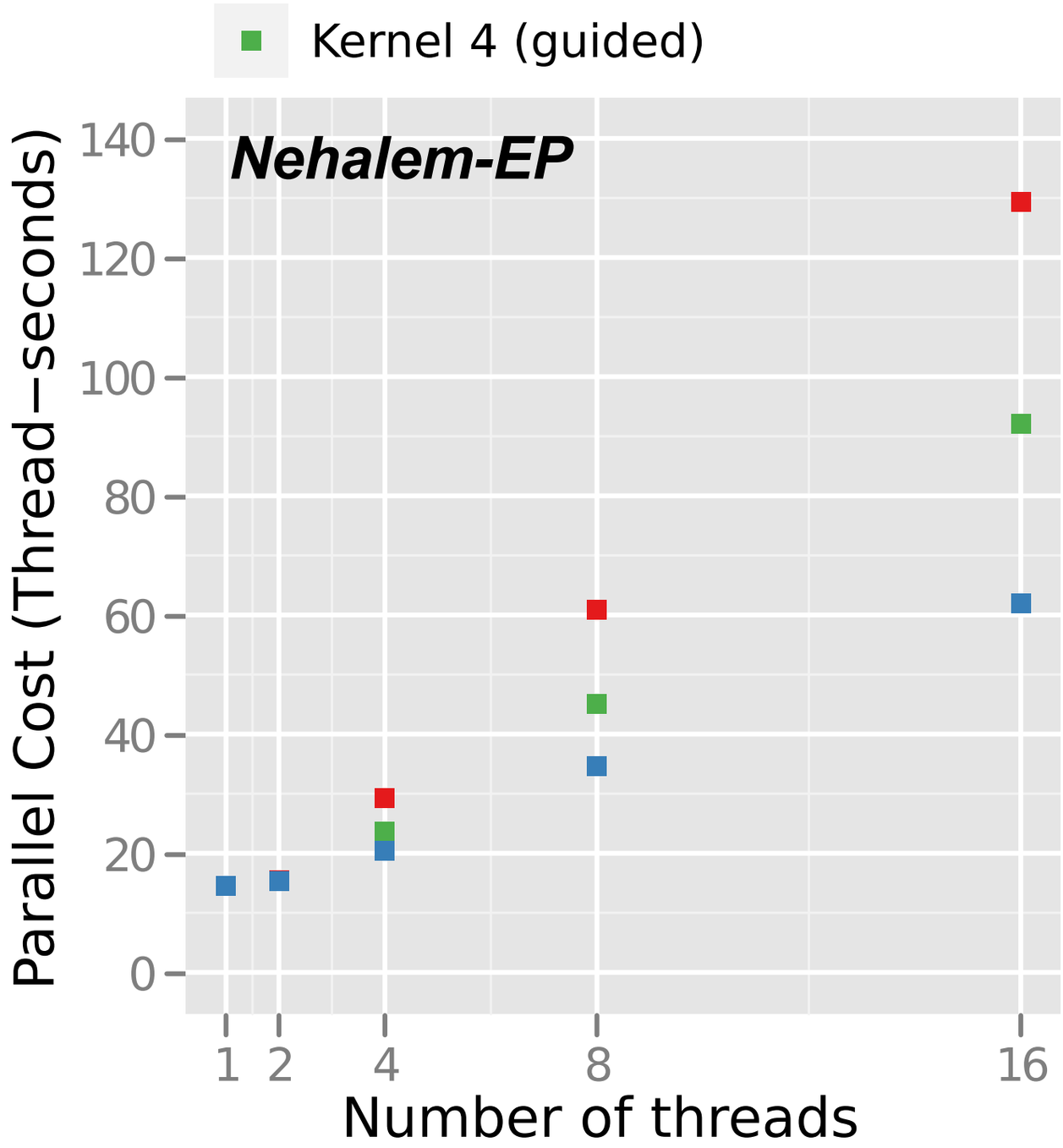
- On a 4 x 4-core NUMA system, observe load imbalance in K4 where previously there had not been one
- Observe identical flop instruction counts, suggesting memory cost imbalance
- Possible quick fixes
  - *Guided* scheduling
  - NUMA-aware allocation



STAGE 1: BLACK BOX

Assume simple profiling and no code changes.

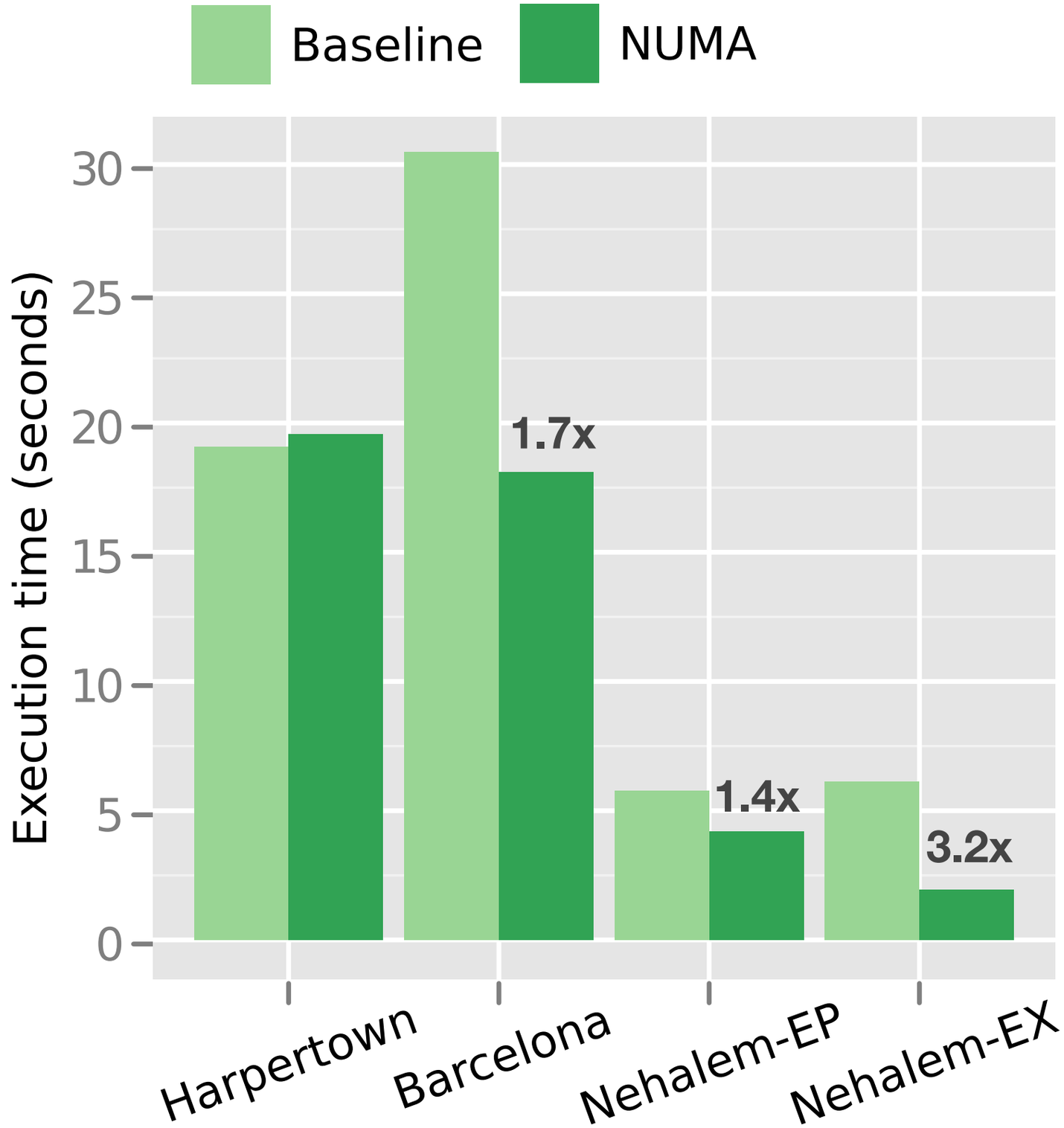
- On a 4 x 4-core NUMA system, observe load imbalance in K4 where previously there had not been one
- Observe identical flop instruction counts, suggesting memory cost imbalance
- Possible quick fixes
  - *Guided* scheduling
  - NUMA-aware allocation



STAGE 1: BLACK BOX

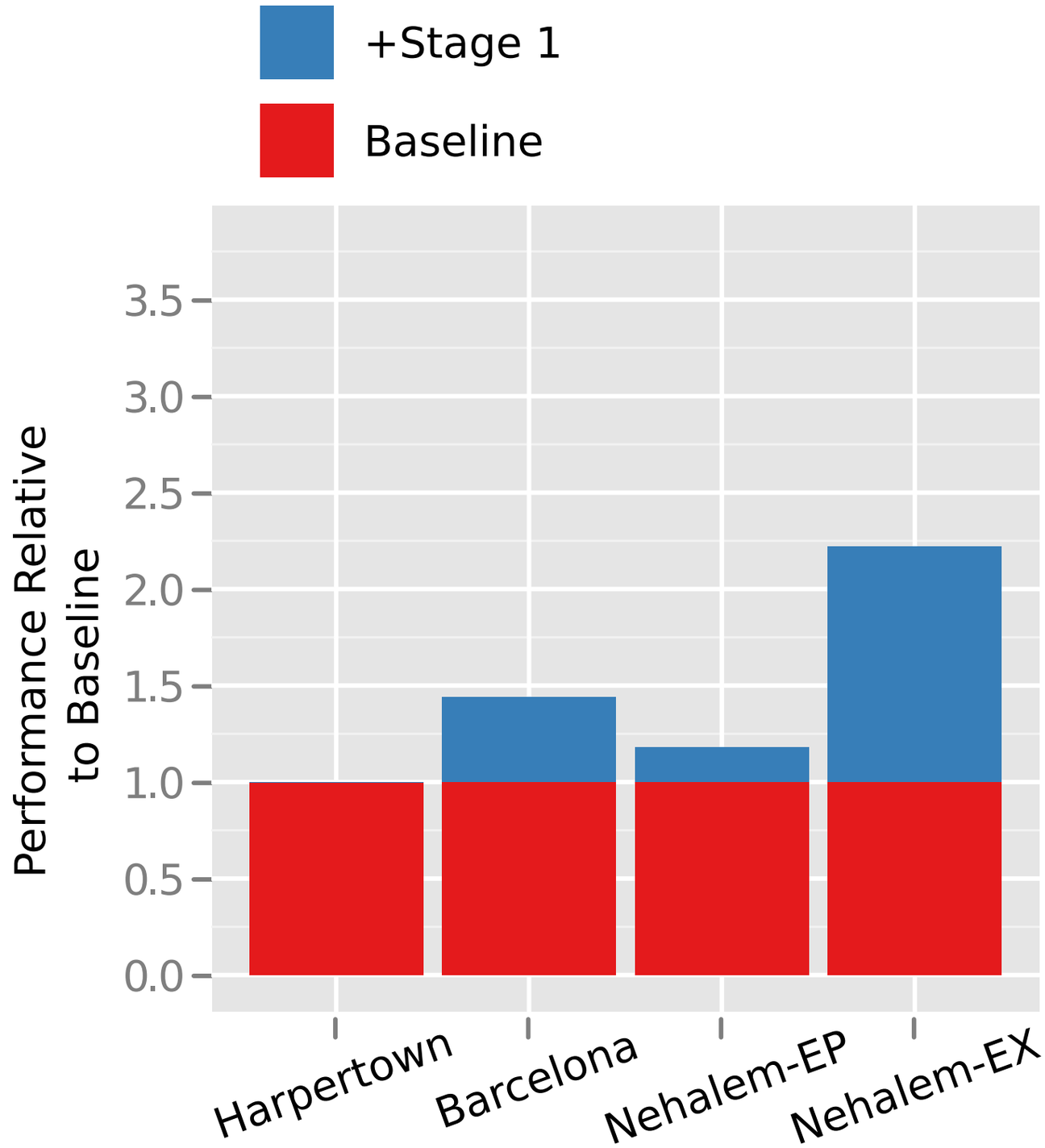
Assume simple profiling and no code changes.

- On a 4 x 4-core NUMA system, observe load imbalance where previously we had not seen one
- Observe identical flop instruction counts, suggesting memory cost imbalance
- Possible quick fixes
  - *Guided* scheduling
  - **NUMA-aware** allocation



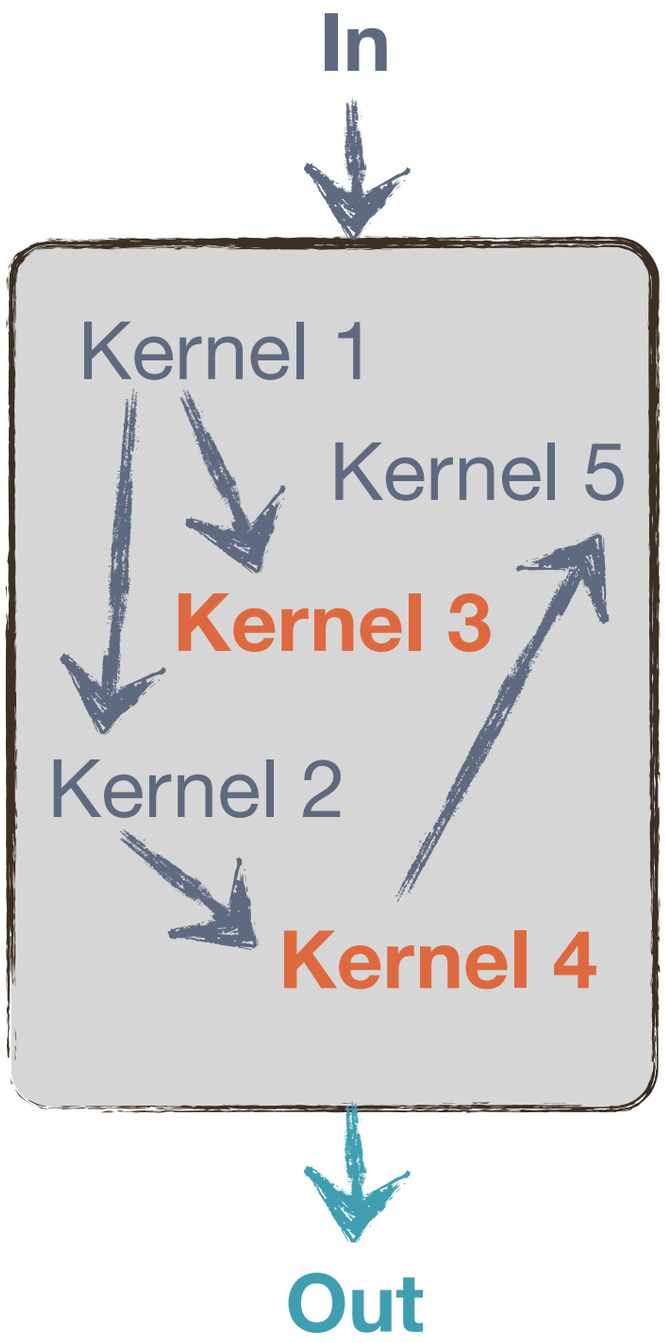
STAGE 1: BLACK BOX

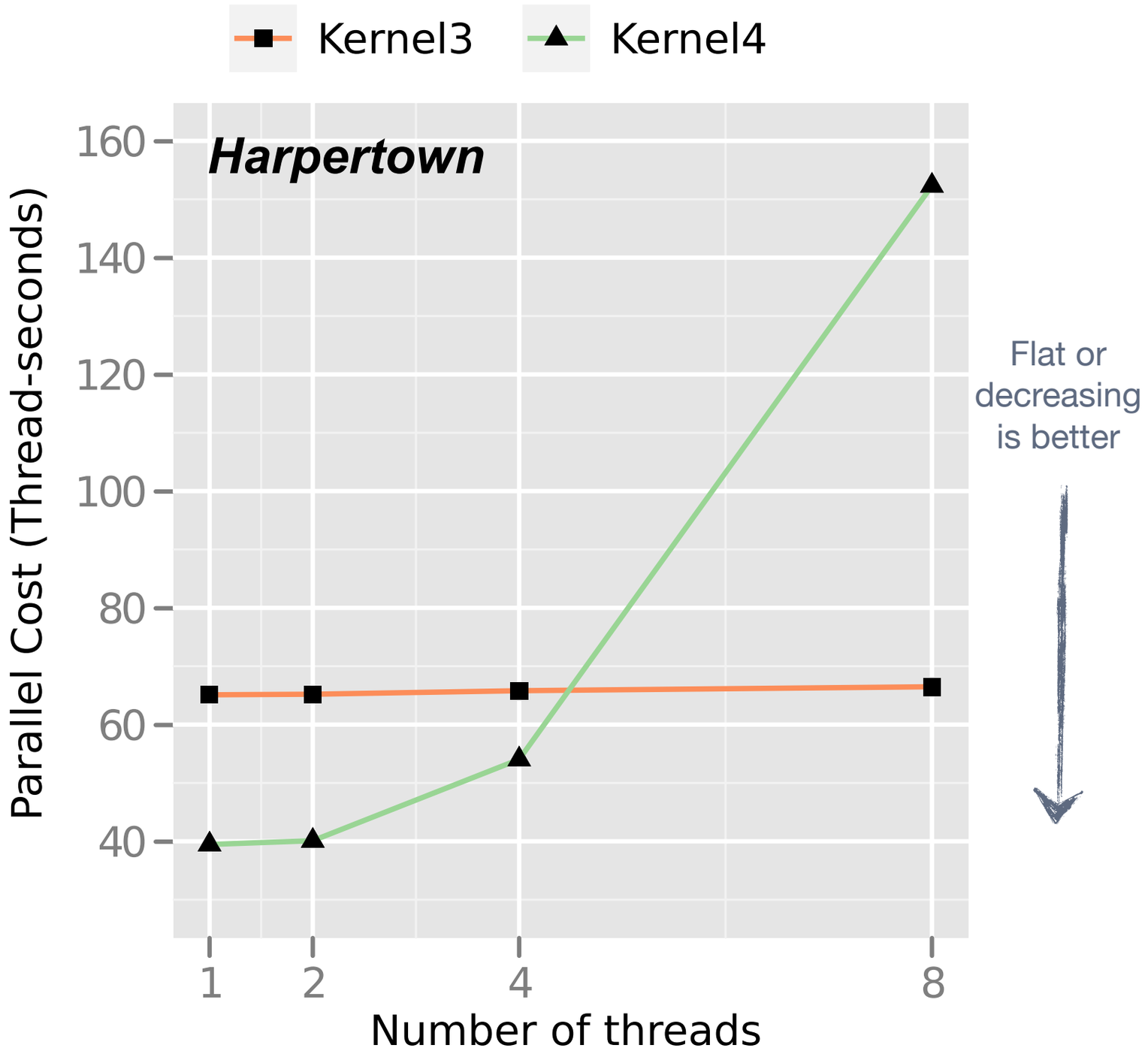
Assume simple profiling and no code changes.



STAGE 1: BLACK BOX

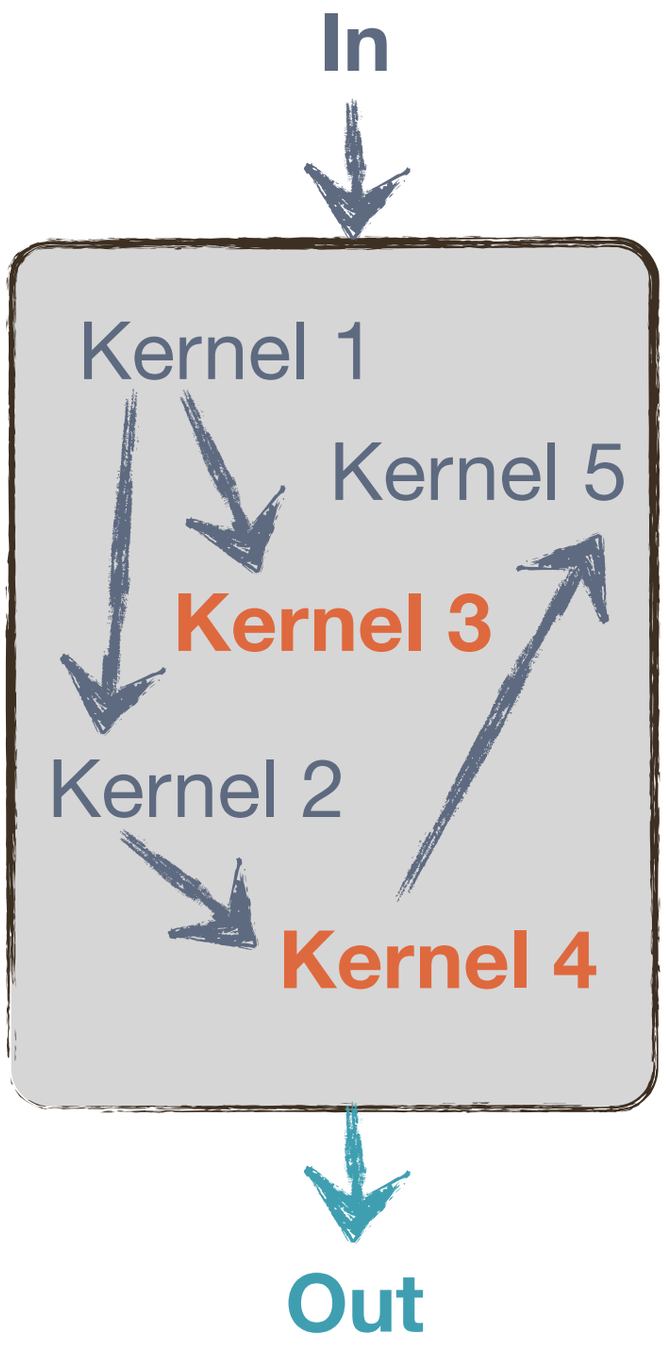
Assume simple profiling and no code changes.

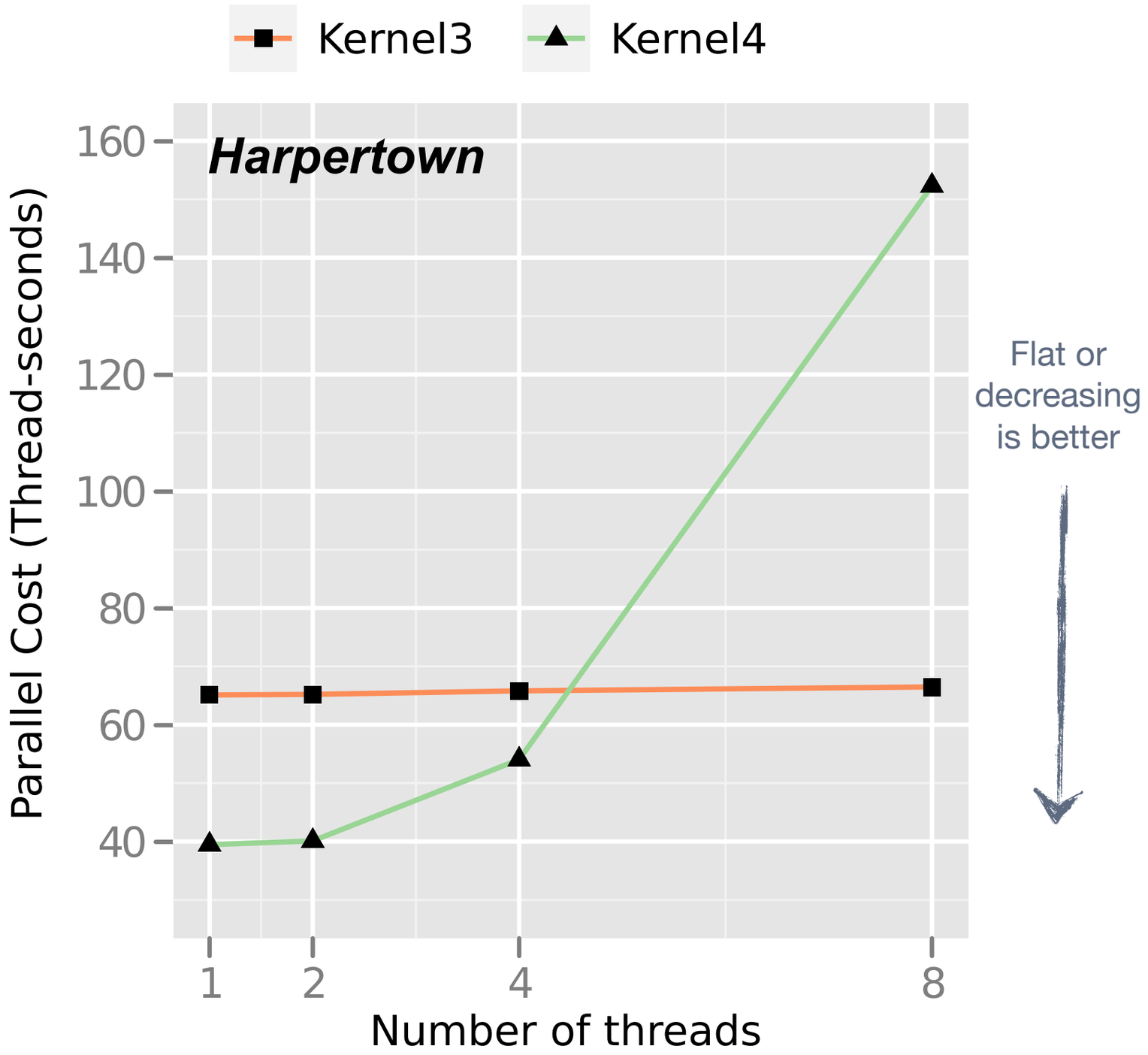




STAGE 2: DEPENDENCE

Assume additional knowledge of the task dependency structure





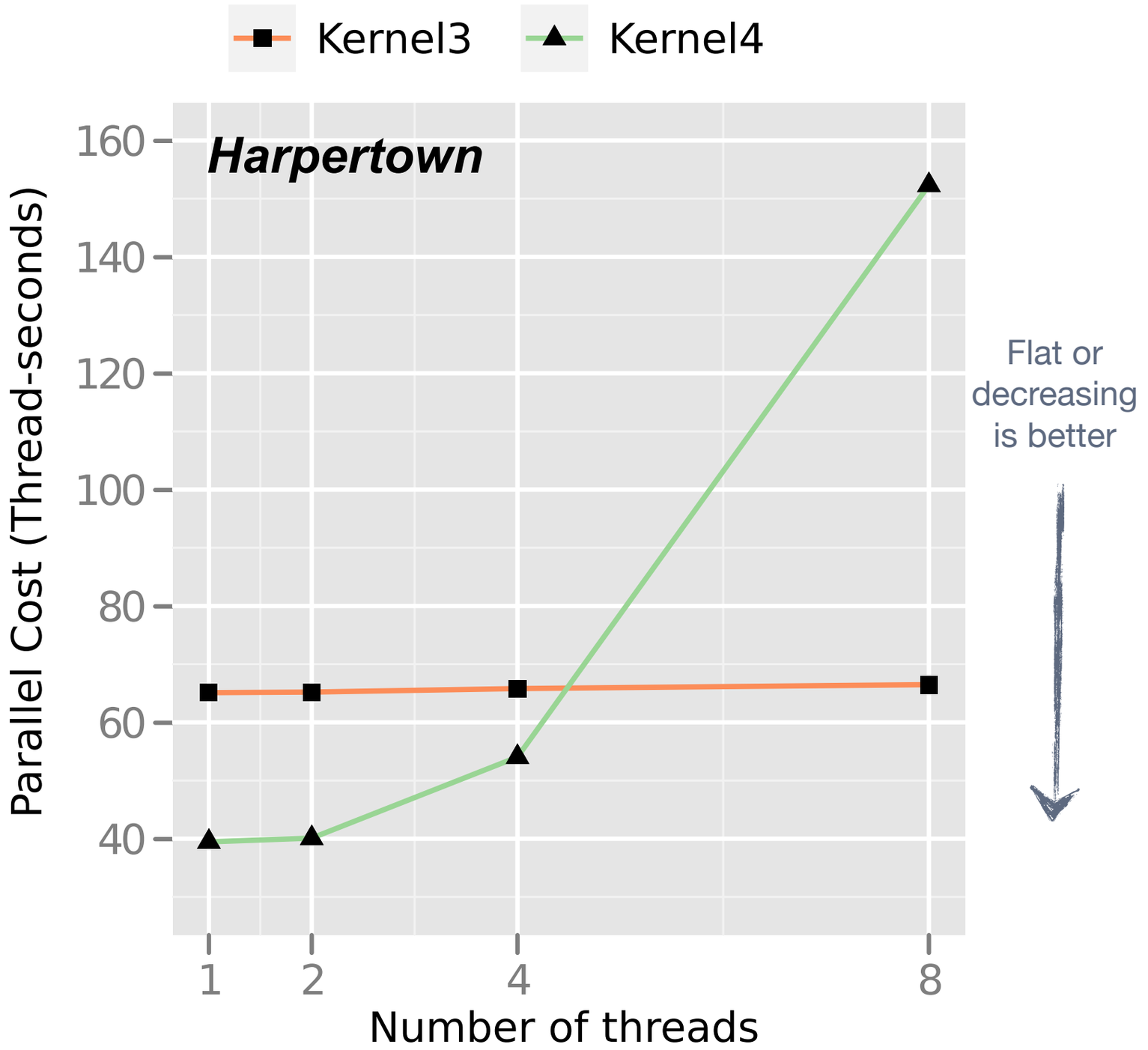
STAGE 2: DEPENDENCE

K3 and K4 are independent. Running them **concurrently → up to 2x** improvement.

- **Idea 1**: Try simultaneous multithreading (**SMT**) execution

- *Intuition*: K3 is compute bound while K4 is memory bound, so there will be no contention for the same processor functional units

- Only works on Nehalem class systems, which implement SMT



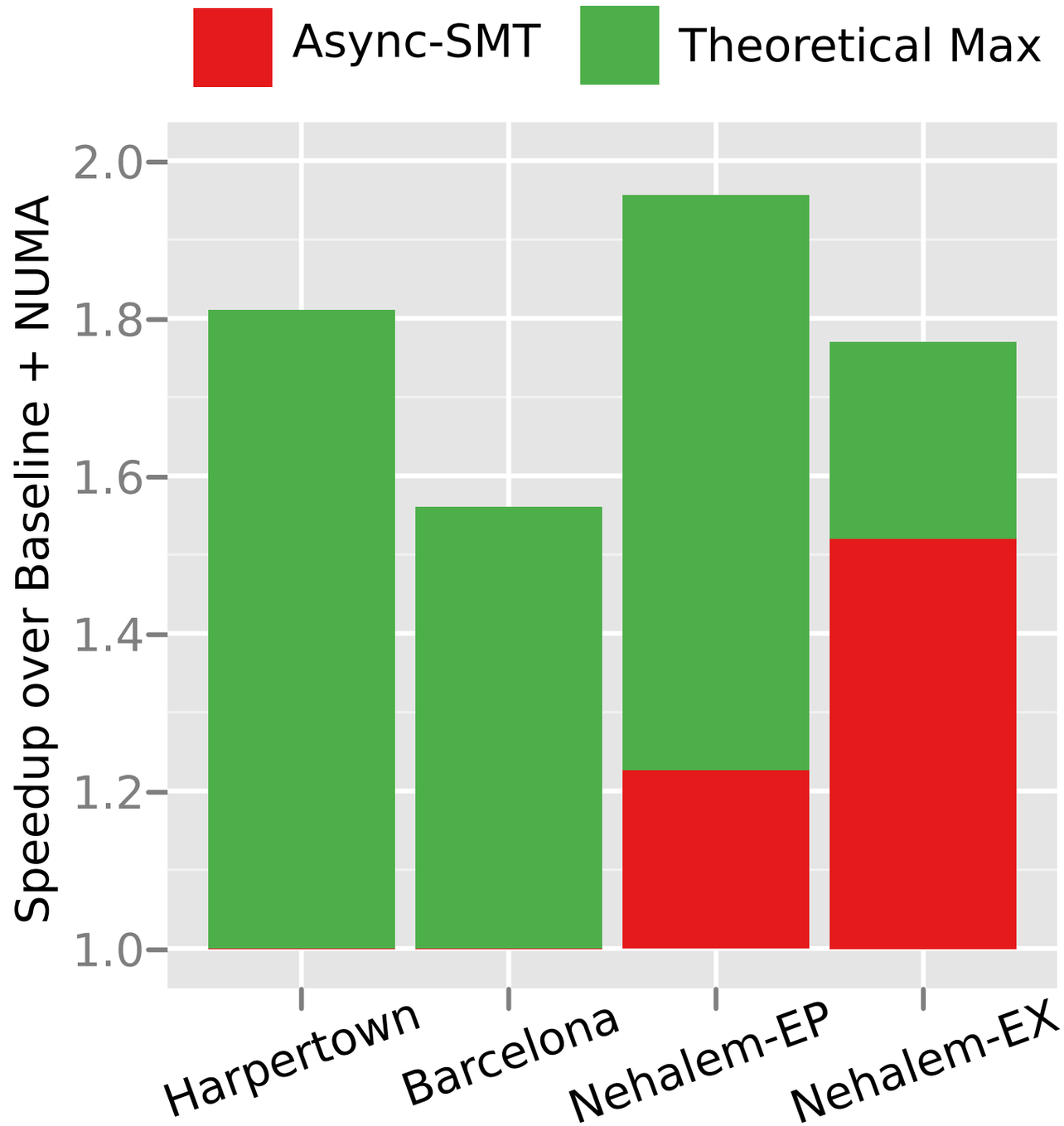
STAGE 2: DEPENDENCE

K3 and K4 are independent. Running them **concurrently → up to 2x** improvement.

- **Idea 1**: Try simultaneous multithreading (**SMT**) execution

- *Intuition*: K3 is compute bound while K4 is memory bound, so there will be no contention for the same processor functional units

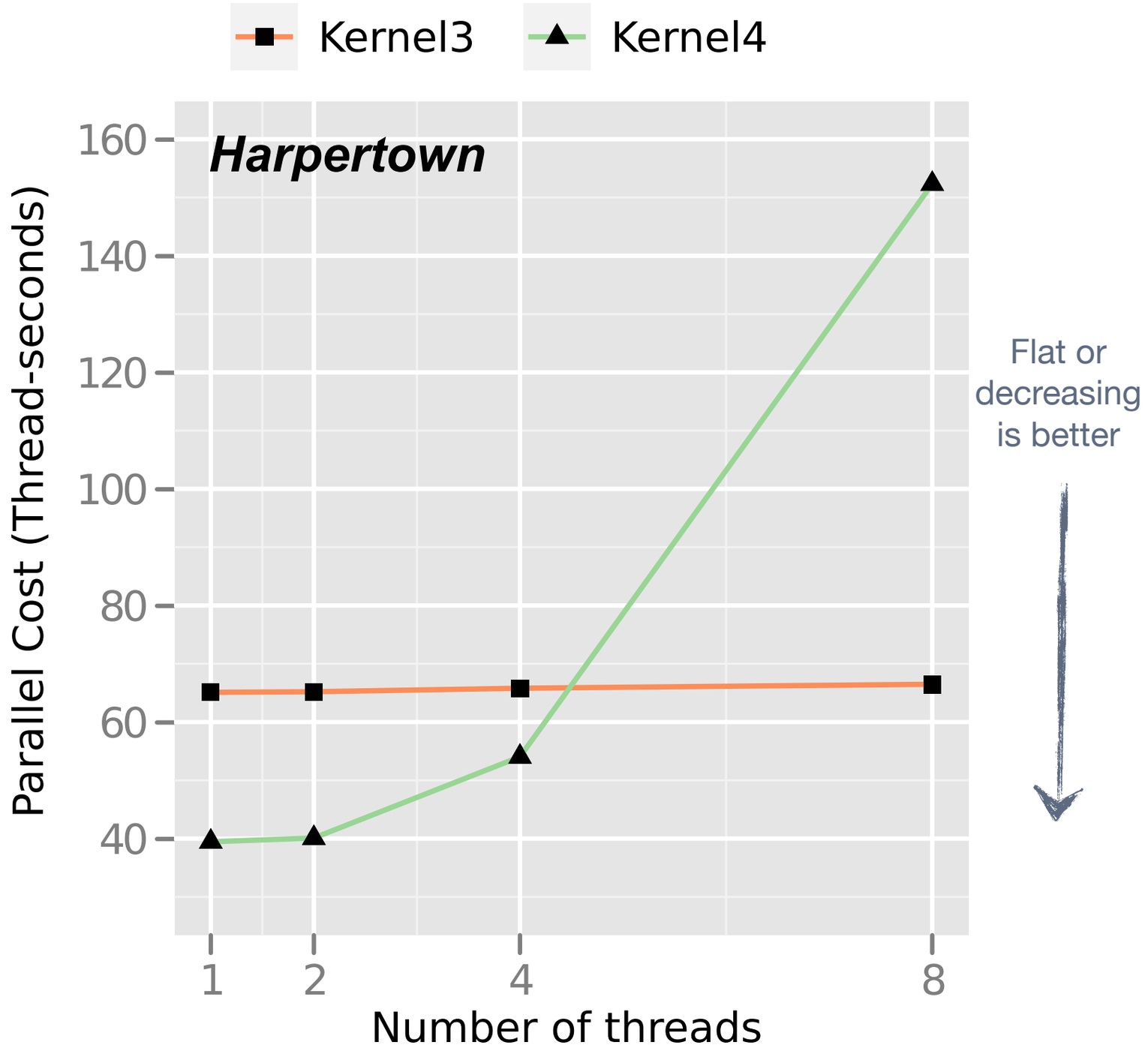
- Only works on Nehalem class systems, which implement SMT



STAGE 2: DEPENDENCE

K3 and K4 are independent. Running them **concurrently → up to 2x** improvement.

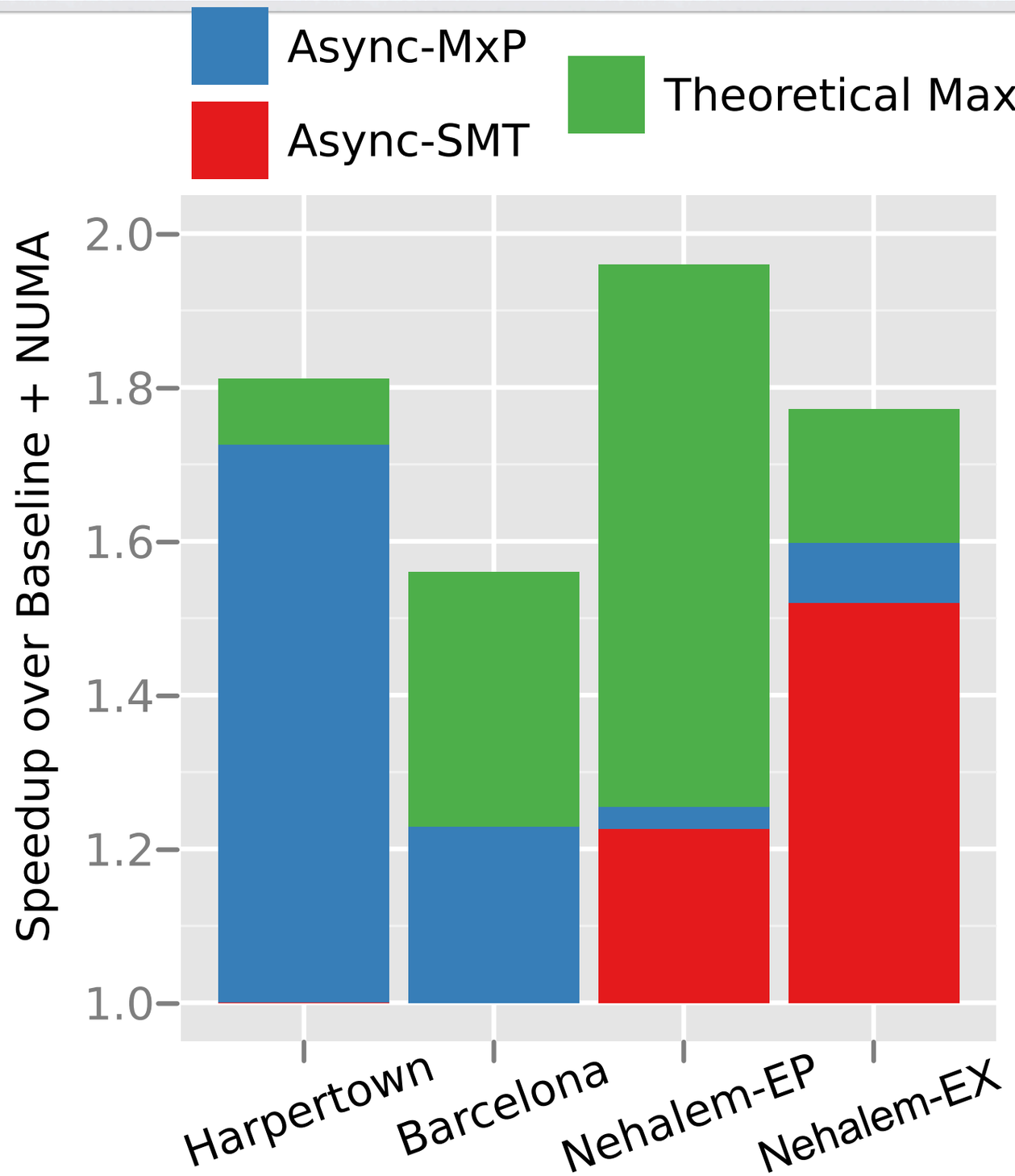
- **Idea 2**: Mixed phase execution
- That is, run K4 up to its scalability limit, and use remaining cores for K3



STAGE 2: DEPENDENCE

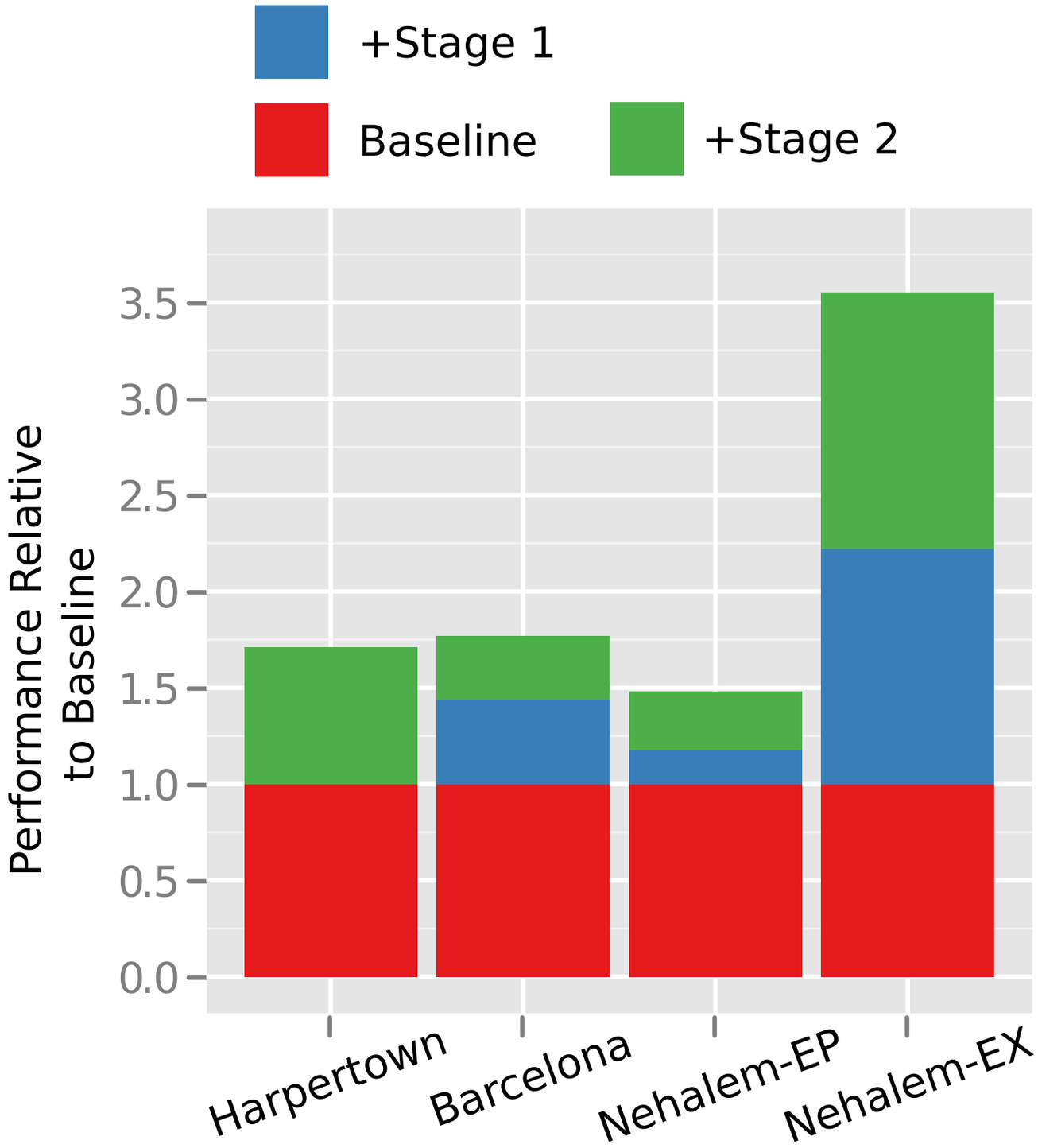
K3 and K4 are independent. Running them **concurrently → up to 2x** improvement.

- **Idea 2**: Mixed phase execution
- That is, run K4 up to its scalability limit, and use remaining cores for K3



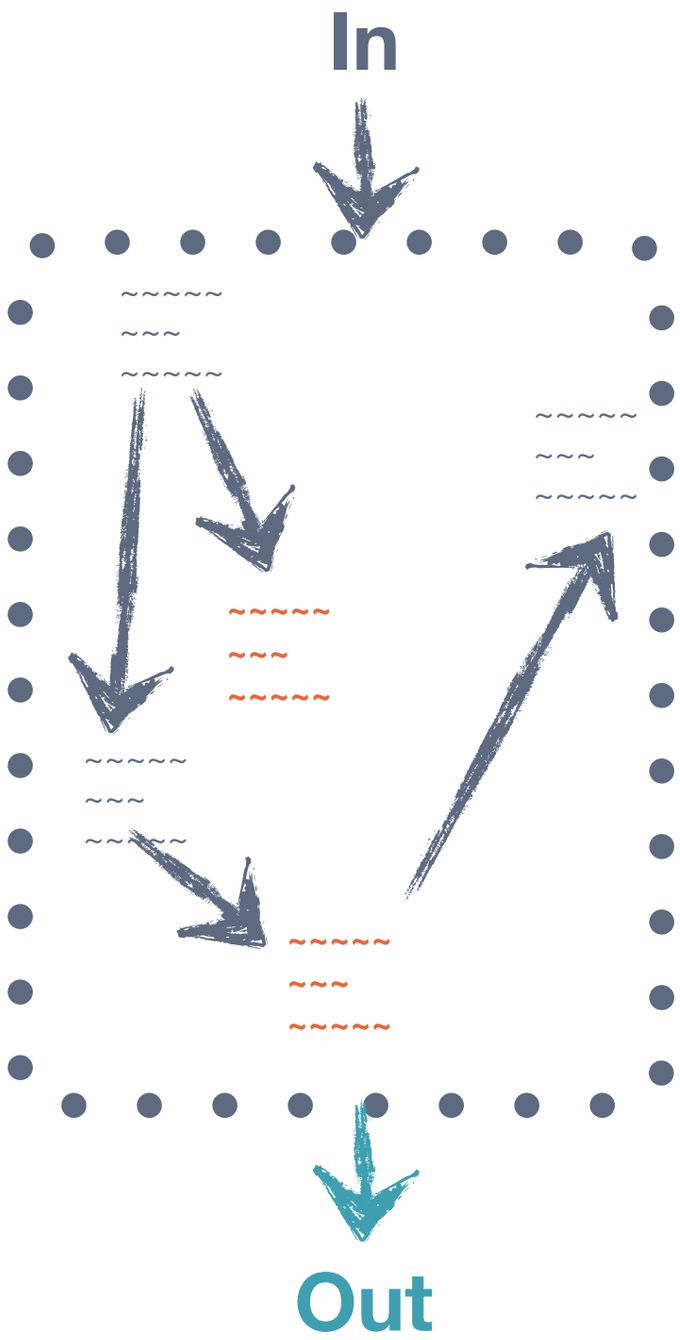
STAGE 2: DEPENDENCE

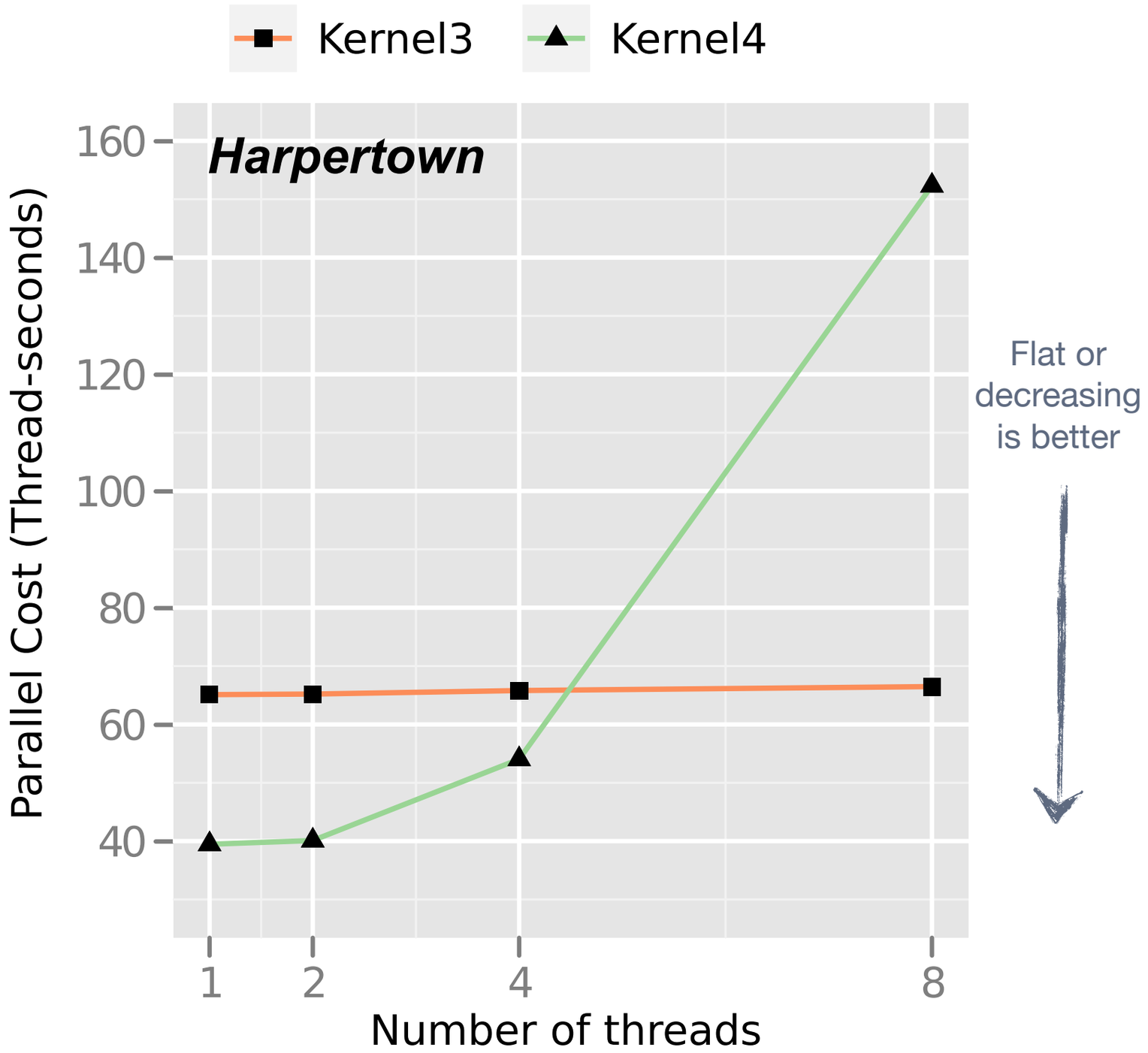
1.2—1.7x improvements possible, regardless of SMT availability



STAGE 2: DEPENDENCE

More extensive code changes but still no “deep” knowledge of code required.





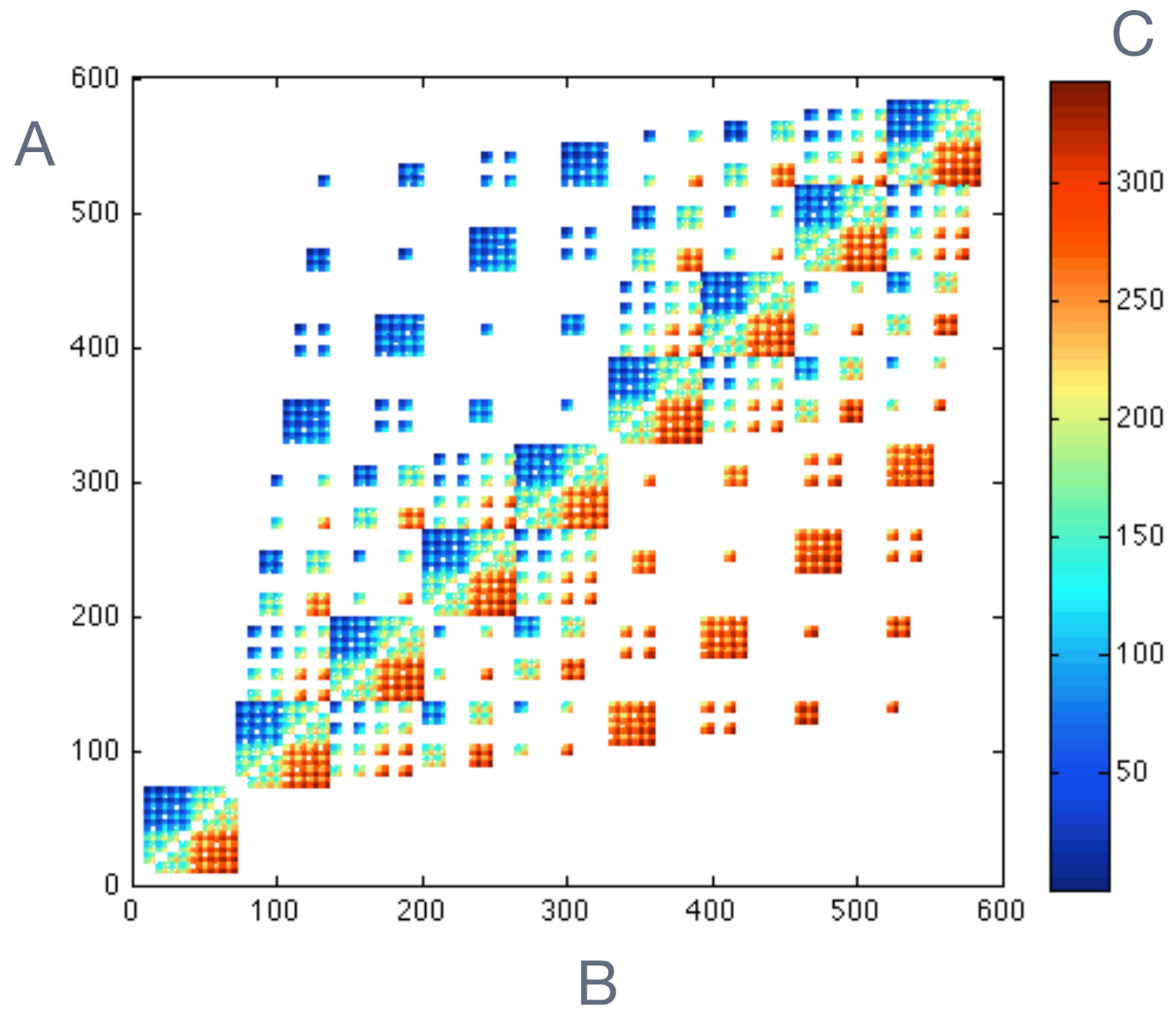
STAGE 3: ORACLE

Assume full knowledge of data access patterns, algorithms, and code

- **Example:** Full knowledge of data access pattern of K4

- Three data arrays, A, B, and C
- Axes (incl. color axis) = array elements
- Each dot = computation on some A(i), B(j), C(k)

- Optimization: Some blocking/tiling/ scheduling of this space



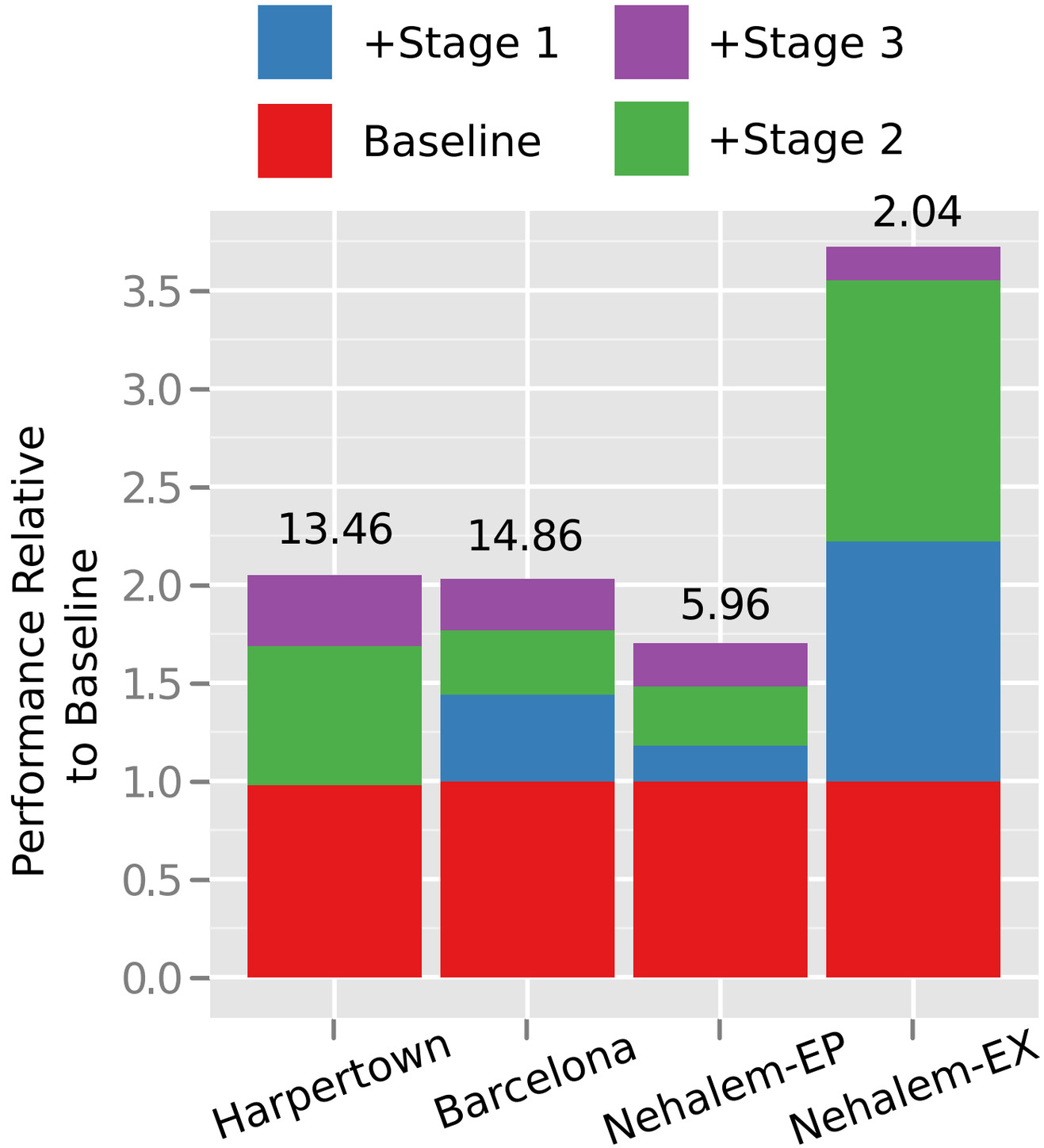
STAGE 3: ORACLE

Assume full knowledge of data access patterns, algorithms, and code

- **Example:** Full knowledge of data access pattern of K4

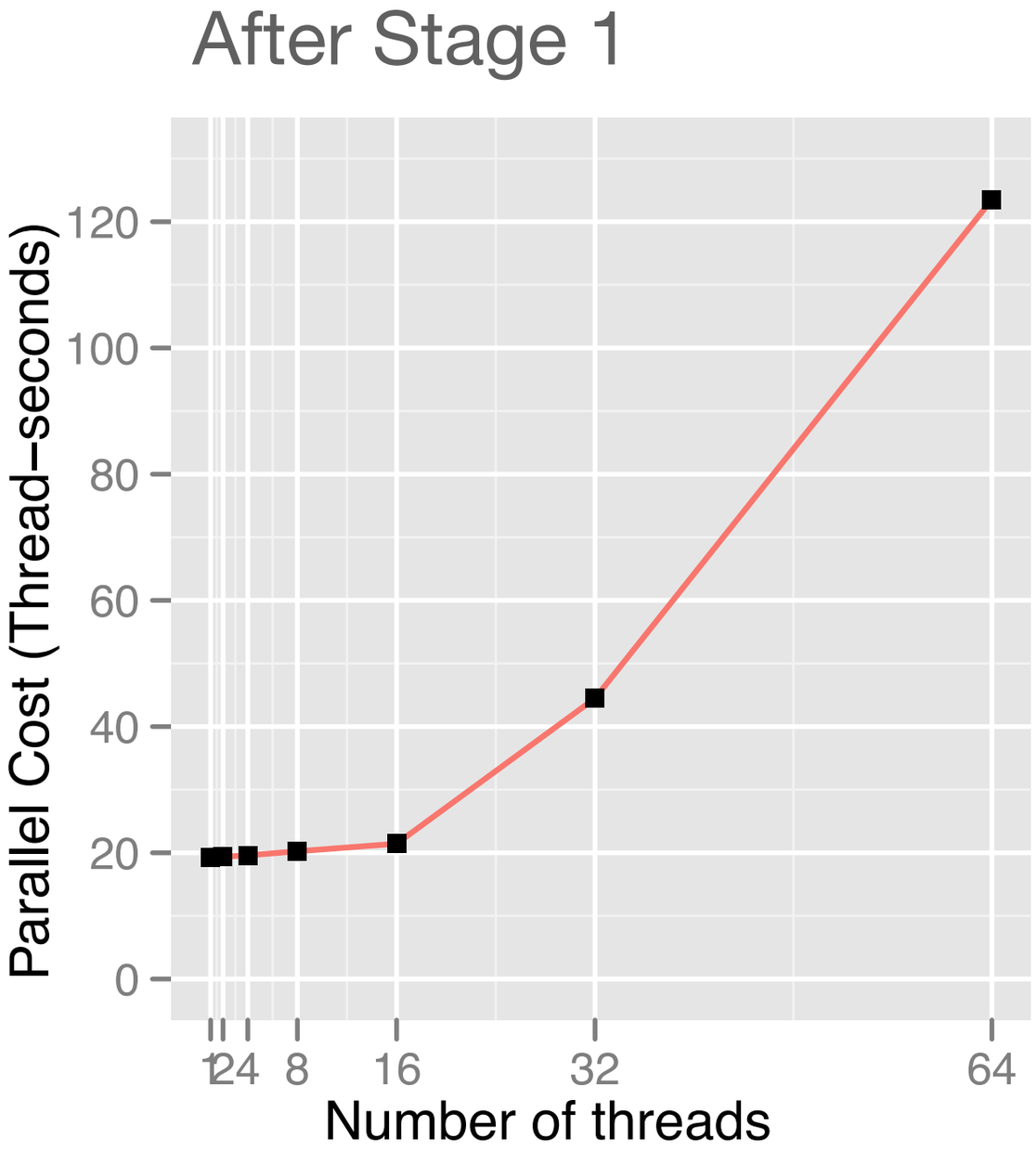
- Three data arrays, A, B, and C
- Axes (incl. color axis) = array elements
- Each dot = computation on some A(i), B(j), C(k)

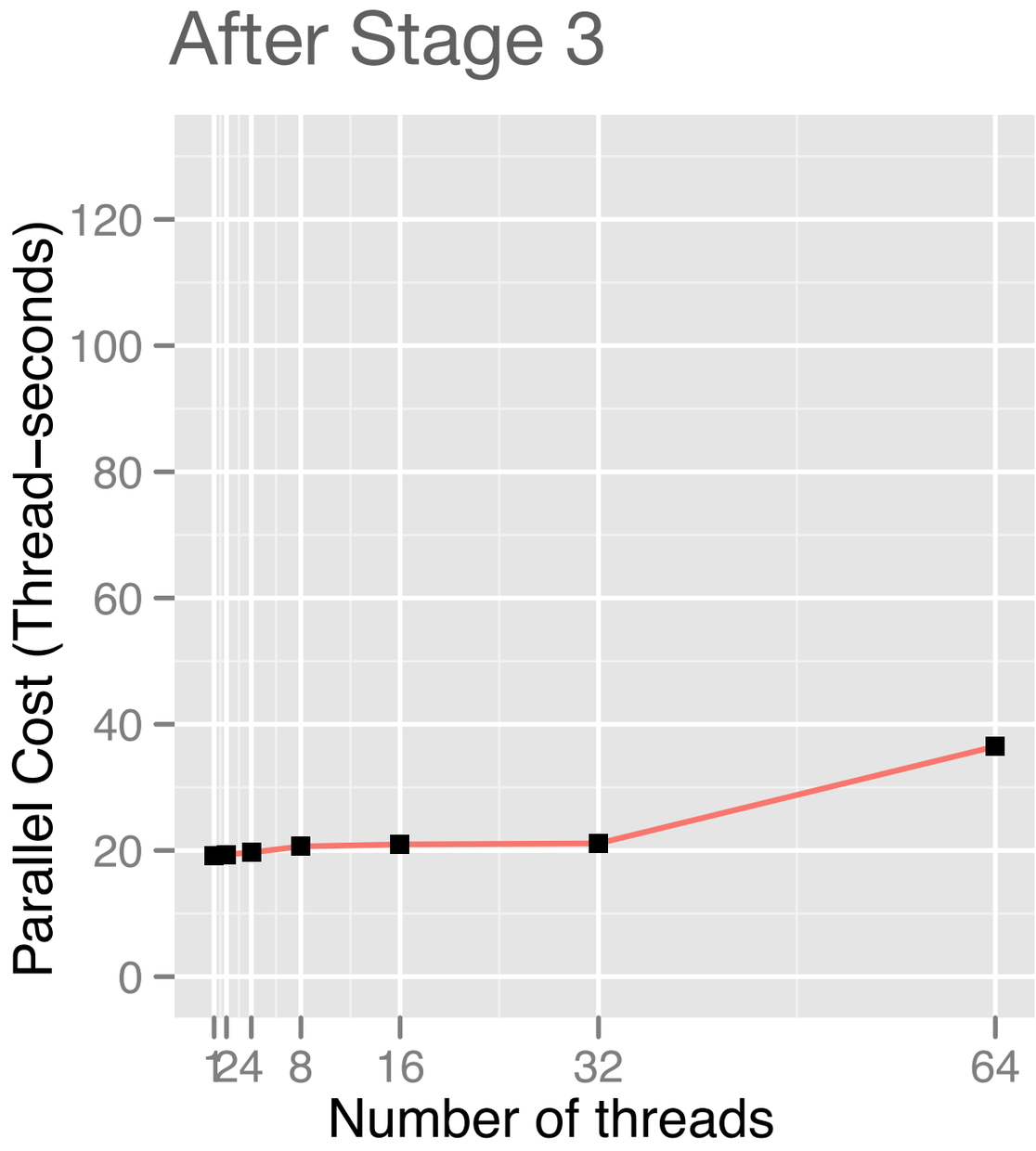
- Optimization: Some blocking/tiling/ scheduling of this space



STAGE 3: ORACLE

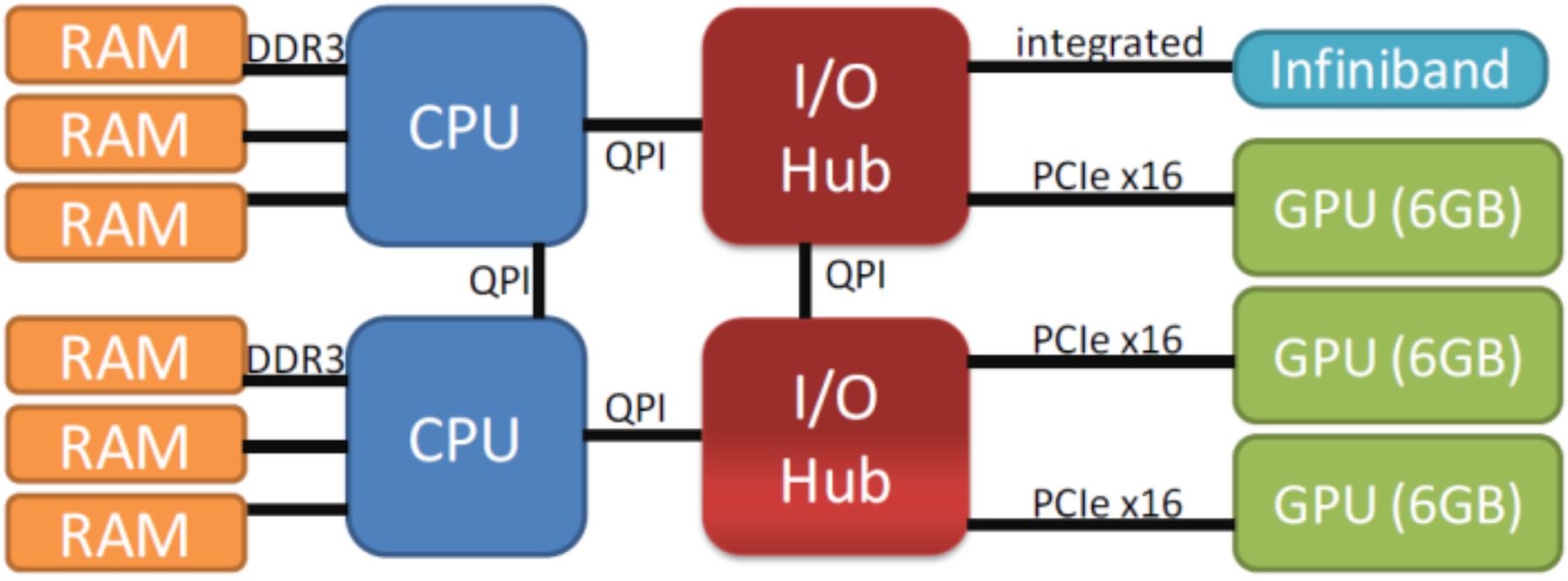
Assume full knowledge of data access patterns, algorithms, and code

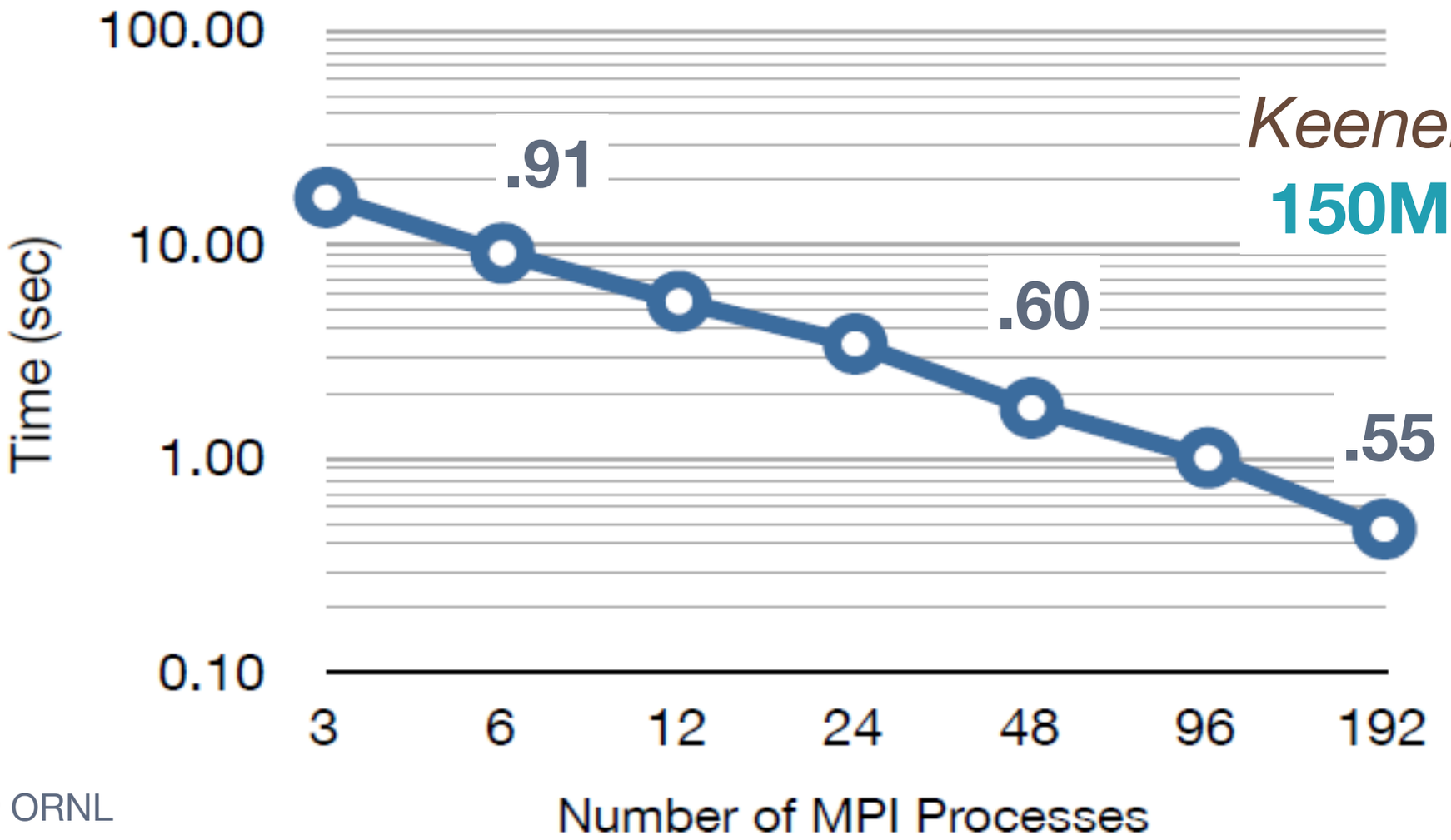




BEFORE & AFTER (NEHALEM-EX)

- Looking to exascale: Two cautionary predictions

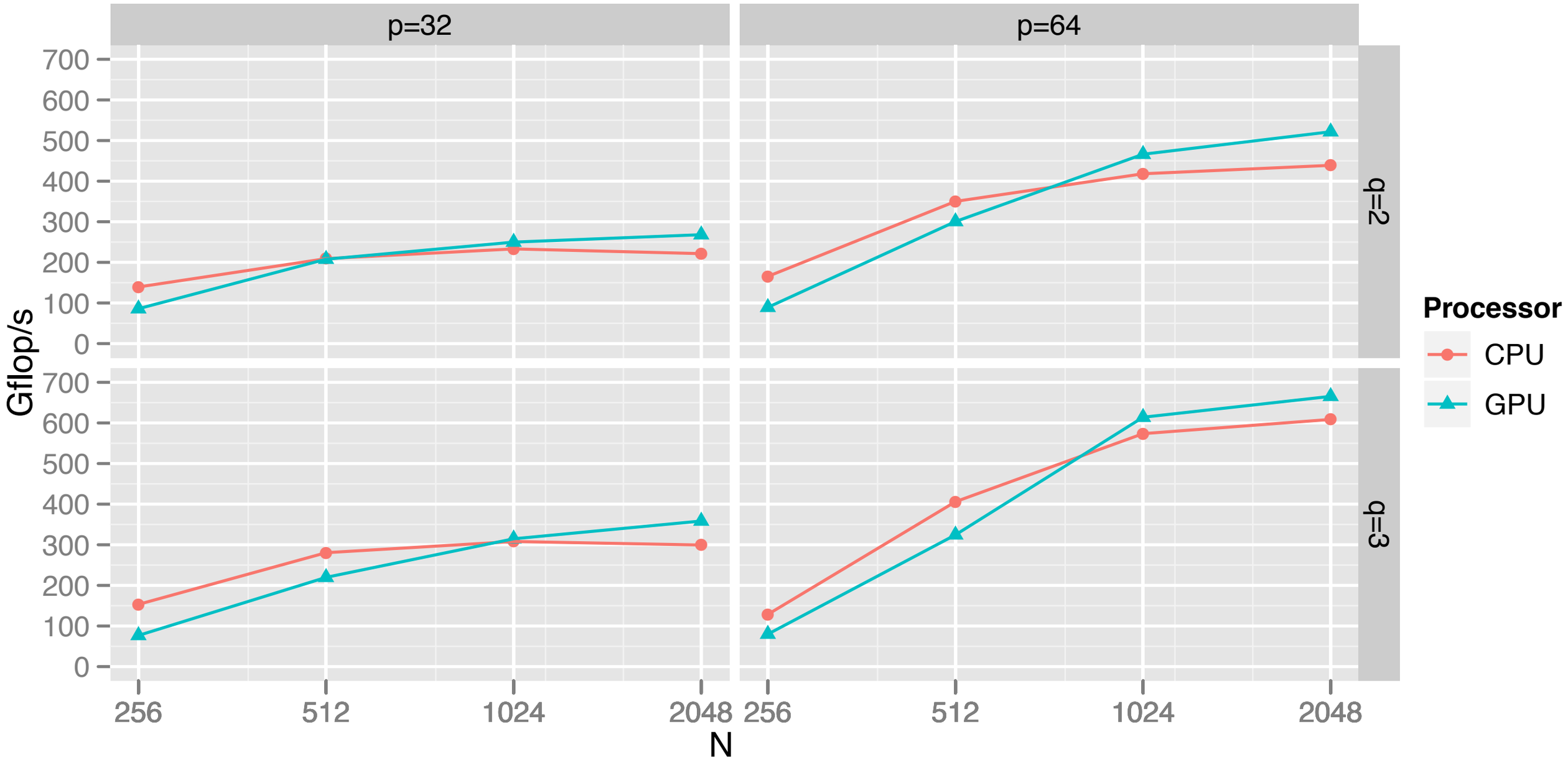




Keeneland (ORNL)

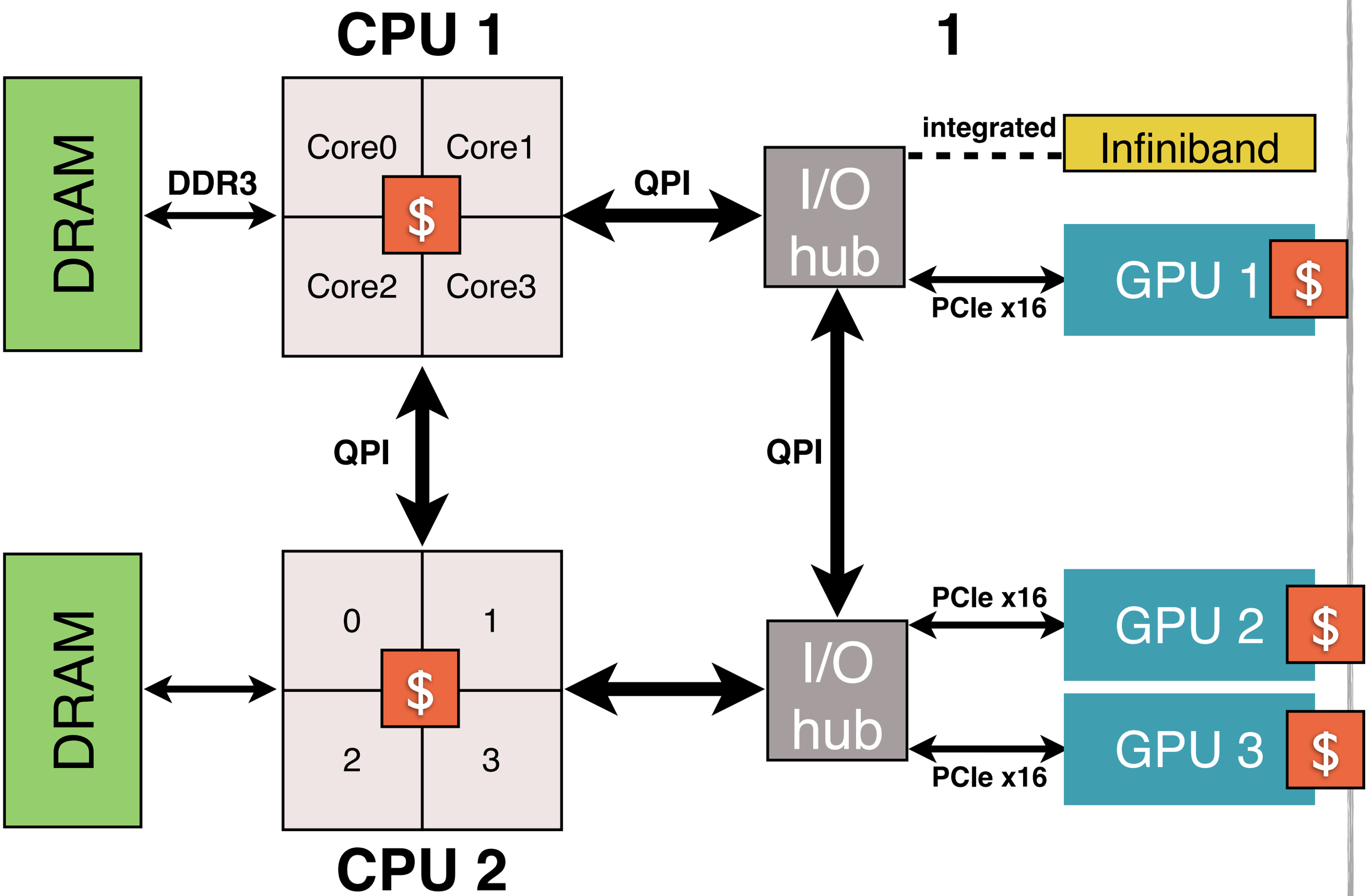
150M pts in 0.4s

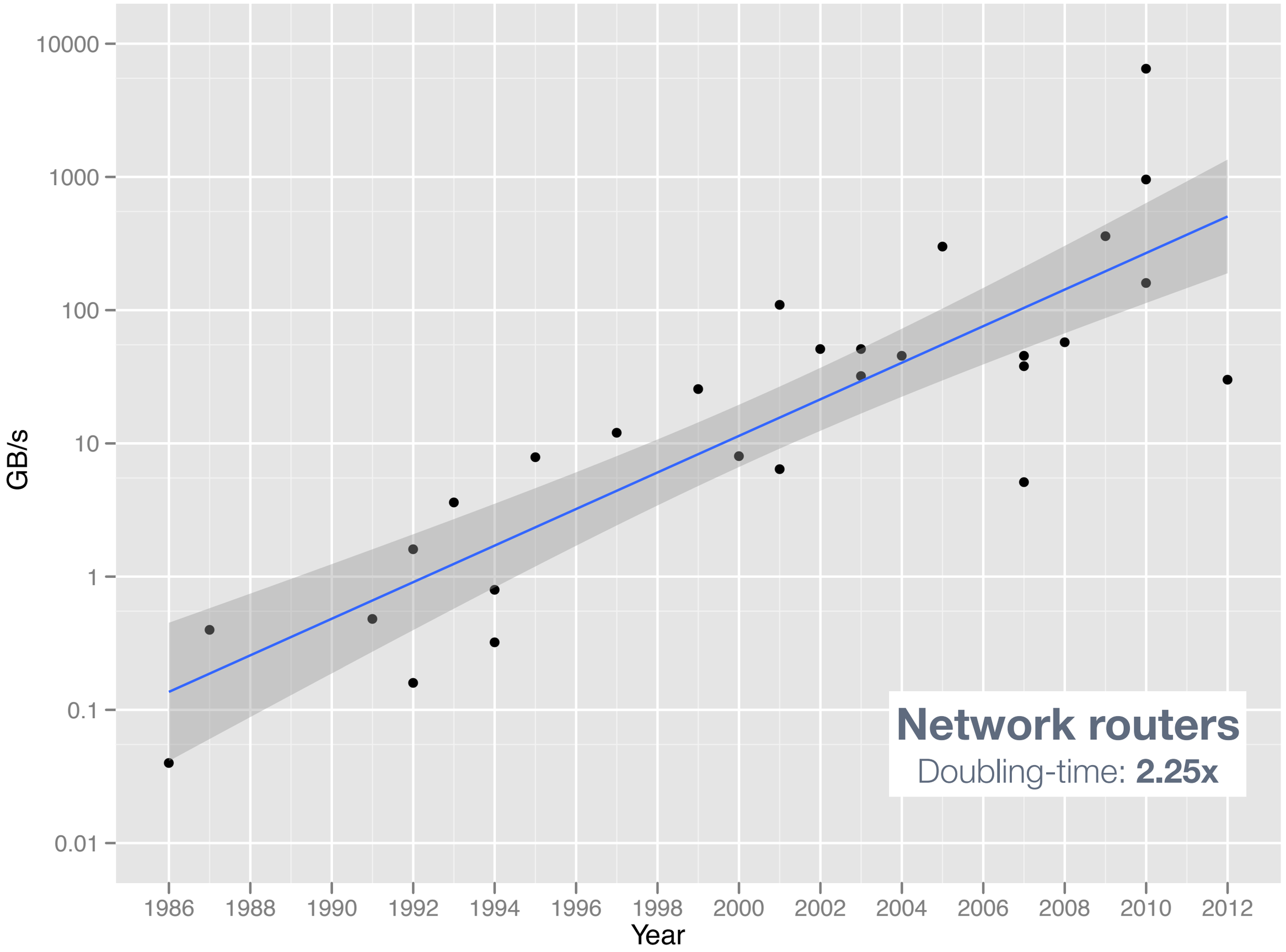
Phil Roth @ ORNL

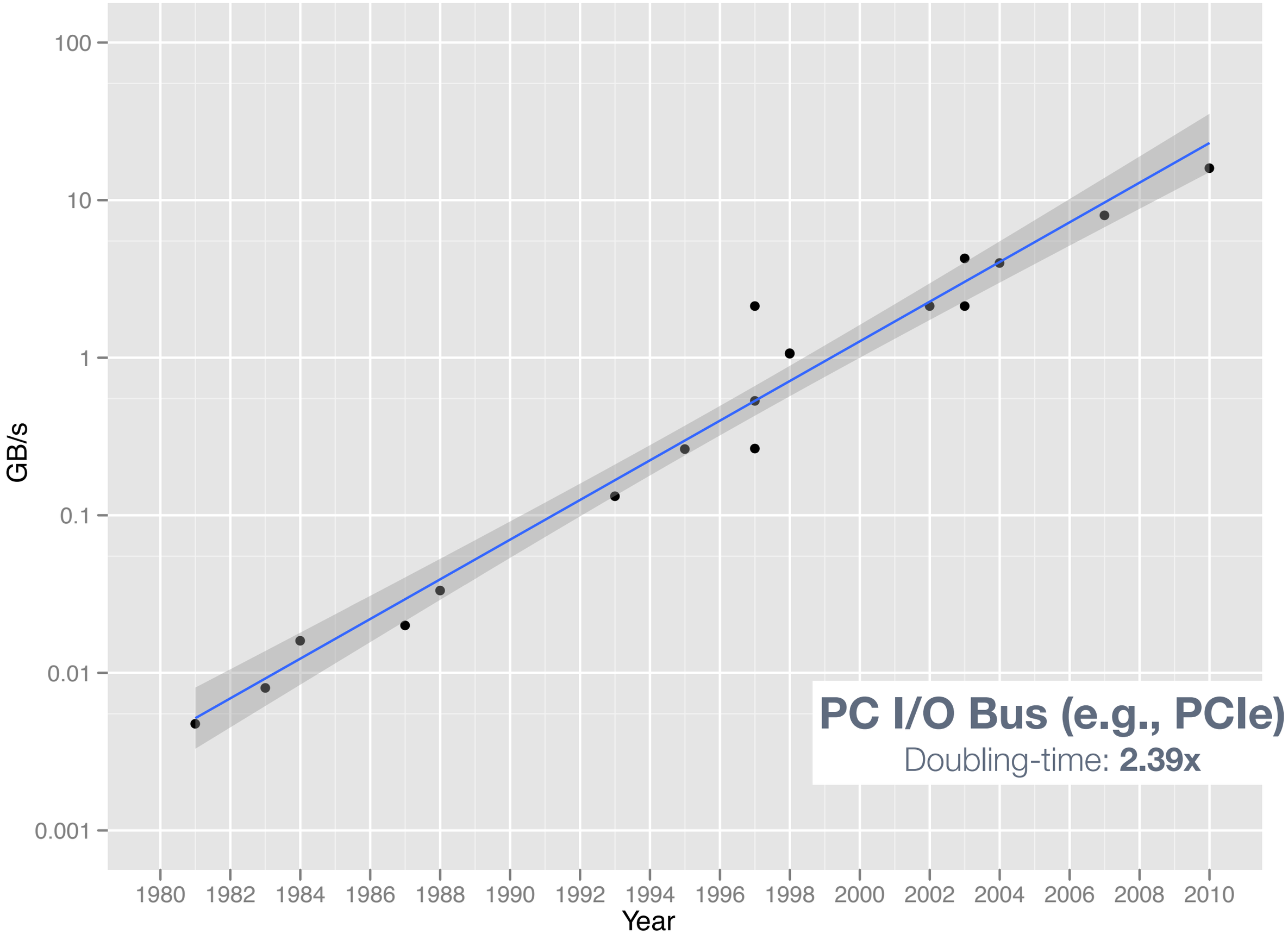


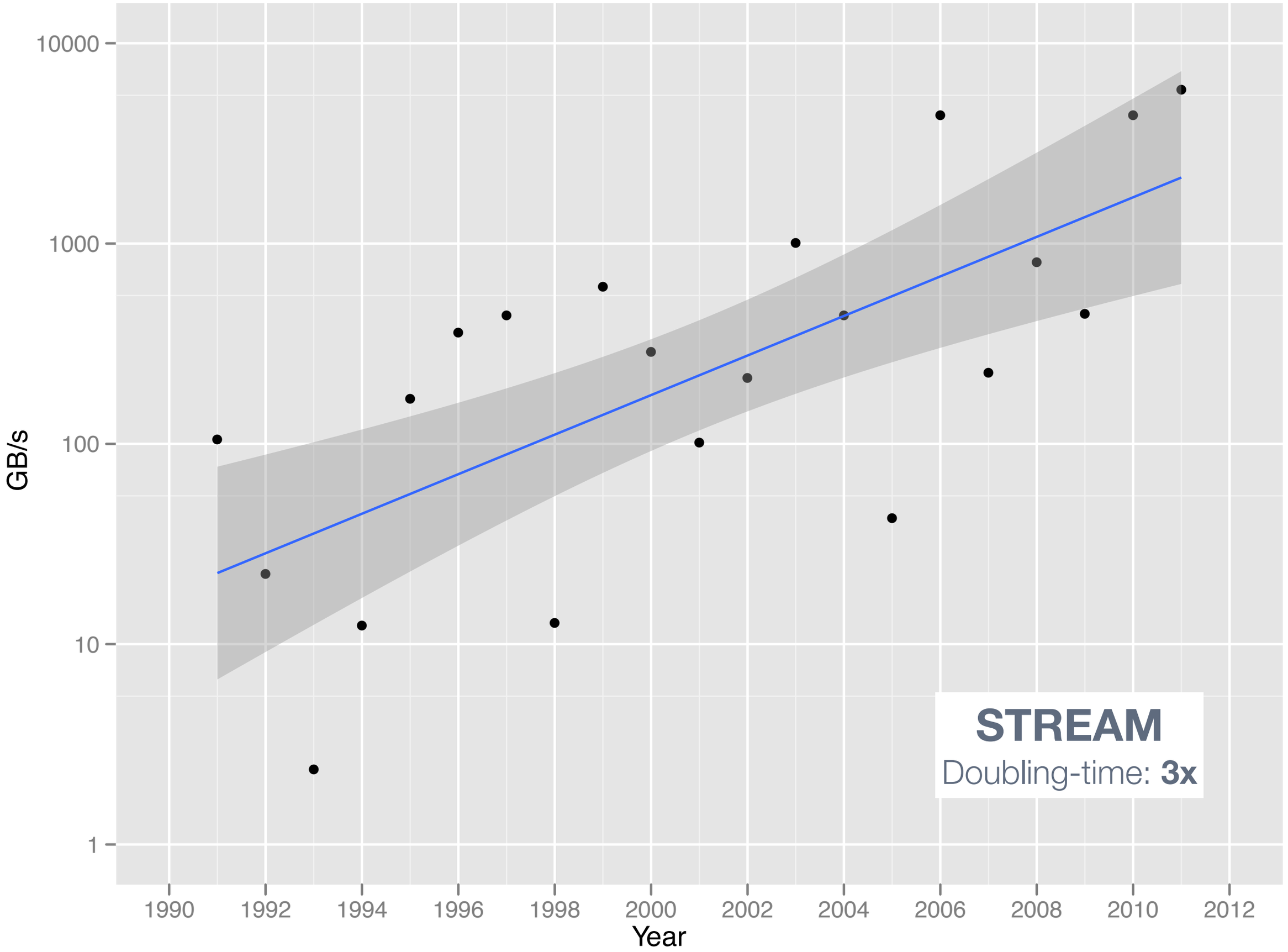
3D FFT, $N * N * N$ — P3DFFT, CUFFT vs. FFTW / MKL

Node


CPU 1
1
DRAM
DDR3
Core0
Core1
$
Core2
Core3
QPI
I/O hub
integrated
Infiniband
PCIe x16
GPU 1
$
QPI
PCIe x16
GPU 2
$
I/O hub
PCIe x16
GPU 3
$
DRAM
0
1
$
2
3
CPU 2


GB/s
10000
1000
100
10
1
0.1
0.01
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
Year
Network routers
Doubling-time: 2.25x


100
10
1
0.1
0.01
0.001
GB/s
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
Year
PC I/O Bus (e.g., PCIe)
Doubling-time: 2.39x


10000
1000
100
10
1
GB/s
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
Year
STREAM
Doubling-time: 3x

3D FFT Swim Lanes: A prediction for 2020