Antisocial Parallelism: Avoiding, Hiding and Managing Communication

(in Biological Data Analysis)

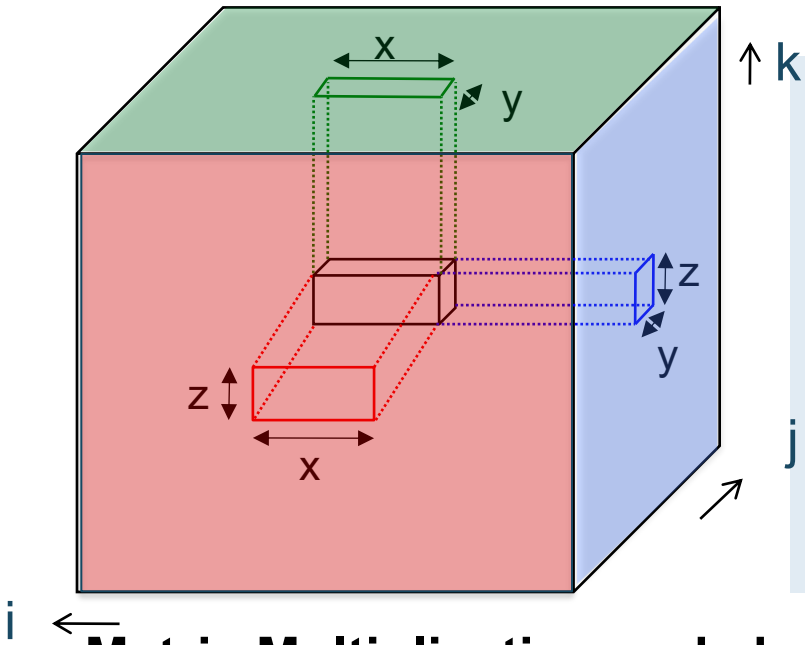
Kathy Yelick
Professor of Electrical Engineering and Computer Sciences
U.C. Berkeley

Associate Laboratory Director
Computing Sciences
Lawrence Berkeley National Laboratory

What's the most expensive operation on a computer?

Communication Avoiding "2.5D" Matrix Multiply

Solomonick & Demmel



- Tiling the iteration space
- 2D algorithm: never chop k dim
- 2.5 or 3D: Assume + is associative; chop k, which is → replication of C matrix
- Optimal for a given memory size (replication factor)

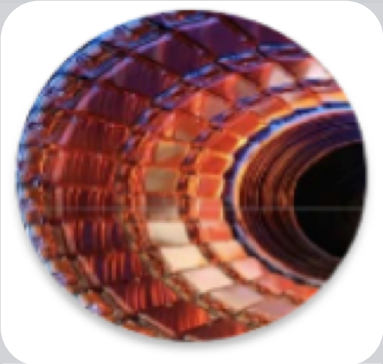
Matrix Multiplication code has a 3D iteration space
Each point in the space is a constant computation (*/+)

```
for i
  for j
    for k
        C[i,j] …  A[i,k] …   B[k,j]  …
```

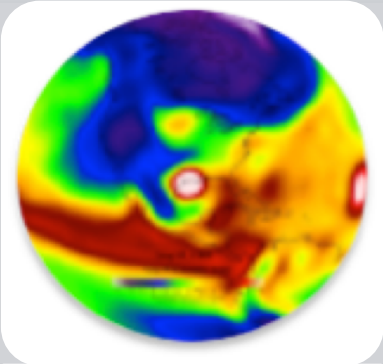
But does communication matter in biology?

Isn’t biology embarrassingly parallel?

Science Problems Fit Across the "Irregularity" Spectrum

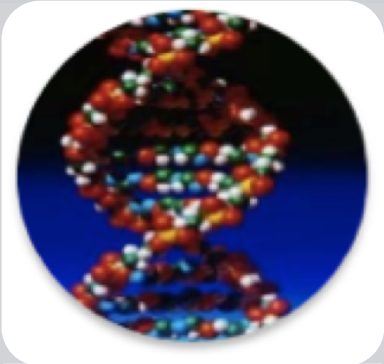


Massive Independent Jobs for Analysis and Simulations



Nearest Neighbor Simulations

All-to-All Simulations



Random access, large data Analysis

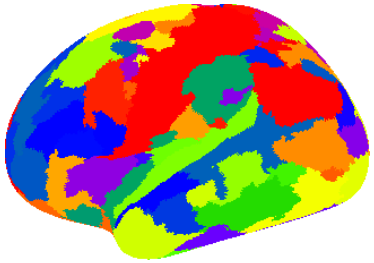
… often they fit in multiple categories

Problems in Computational Biology

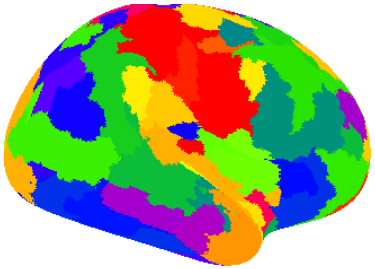
- Graphical Models (Machine Learning)
- Genome Assembly
- Many-to-Many Alignment
- Imaging

And how can we take ideas from numerical computing, linear algebra, and communication avoiding algorithms to this domain?

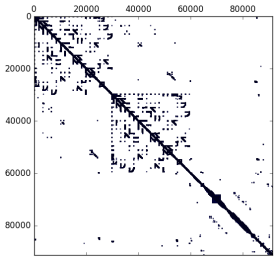
HP-CONCORD on Brain fMRI data



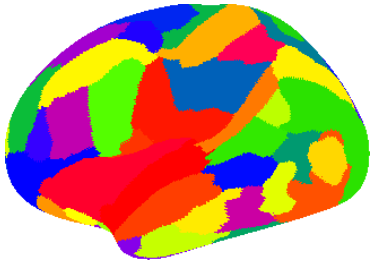
$\lambda_1 = 0.48,\ \lambda_2 = 0.39,\ \epsilon = 3,$
% of best score = 100



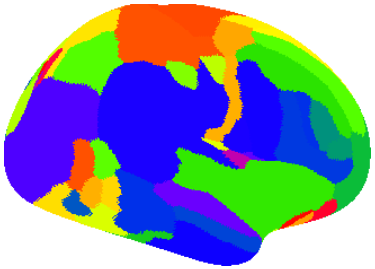
$\lambda_1 = 0.5,\ \lambda_2 = 0.39,\ \epsilon = 3,$
% of best score = 100



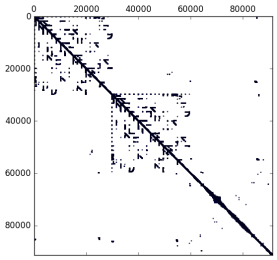
$\lambda_1 = 0.48,\ \lambda_2 = 0.39,\ \epsilon = 3,$
% of best score = 100



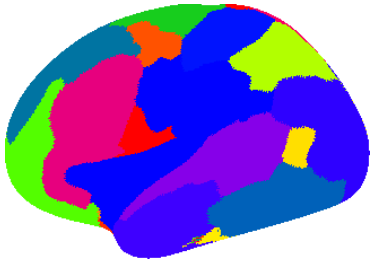
$\lambda_1 = 0.64,\ \lambda_2 = 0.13,\ k = 1,$
% of best score = 75.03



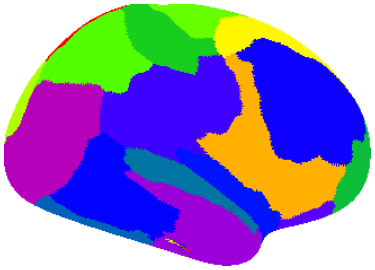
$\lambda_1 = 0.5425,\ \lambda_2 = 0.39,\ k = 0,$
% of best score = 73.45



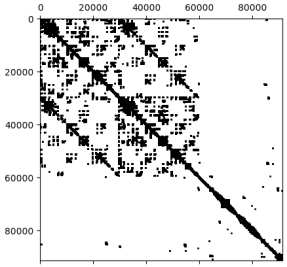
$\lambda_1 = 0.64,\ \lambda_2 = 0.13,\ k = 1,$
% of best score = 75.03



$t = 99.9,\ k = 4,$
% of best score = 32.24

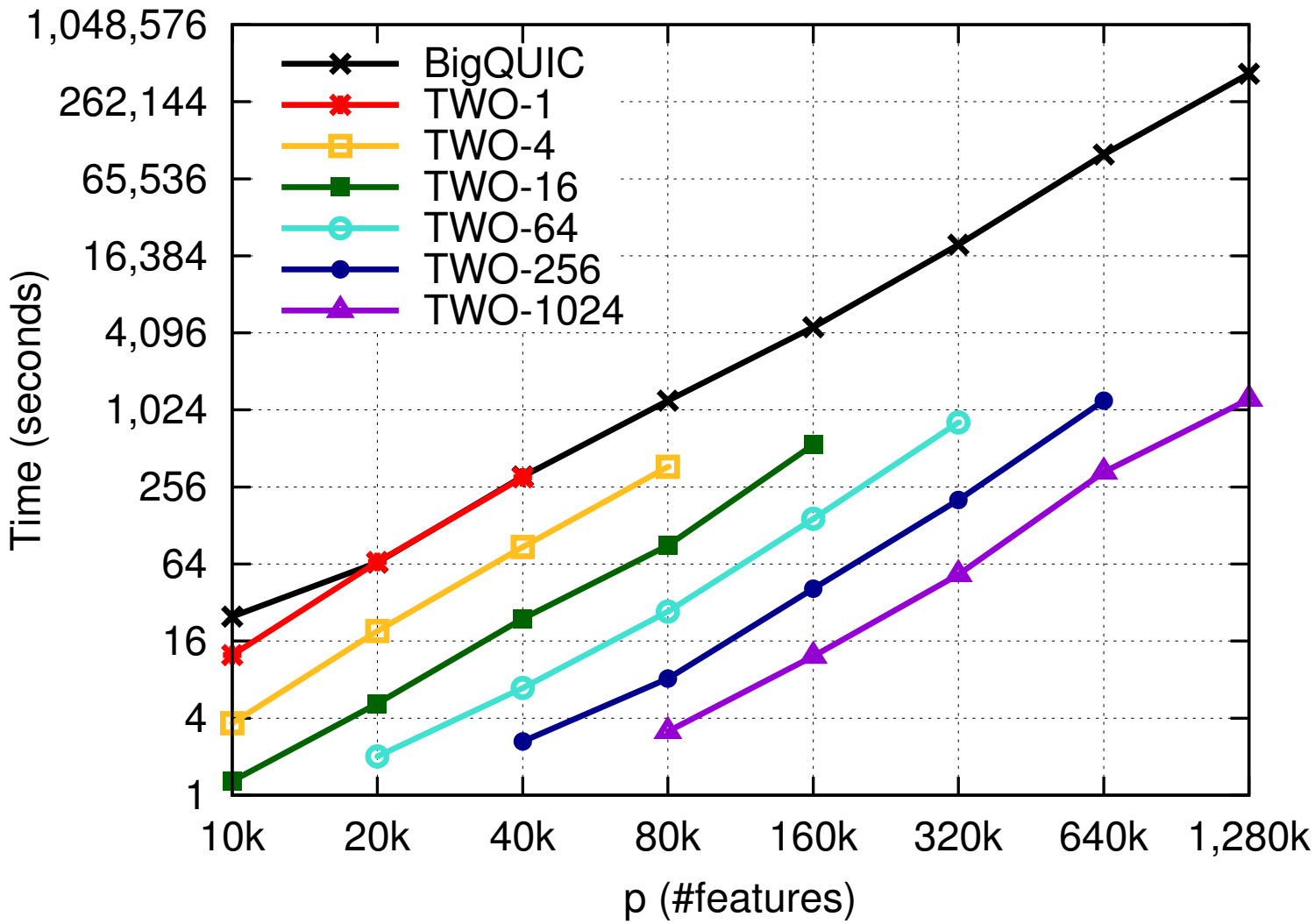


$t = 99.9,\ k = 3,$
% of best score = 32.45

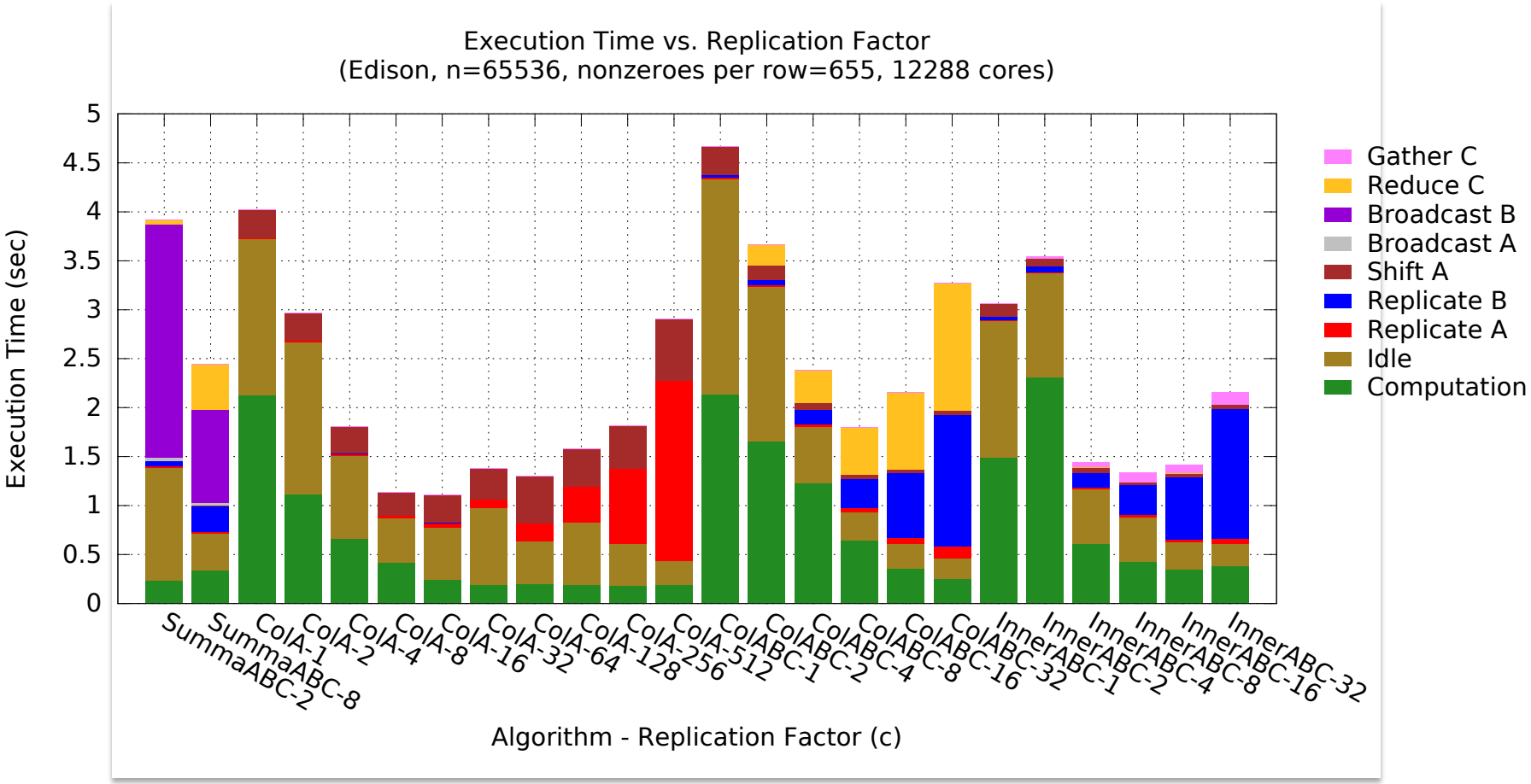


$t = 99.9,\ k = 4$
% of best score = 32.24

Inverse Covariance Matrix Estimation (CONCORD)



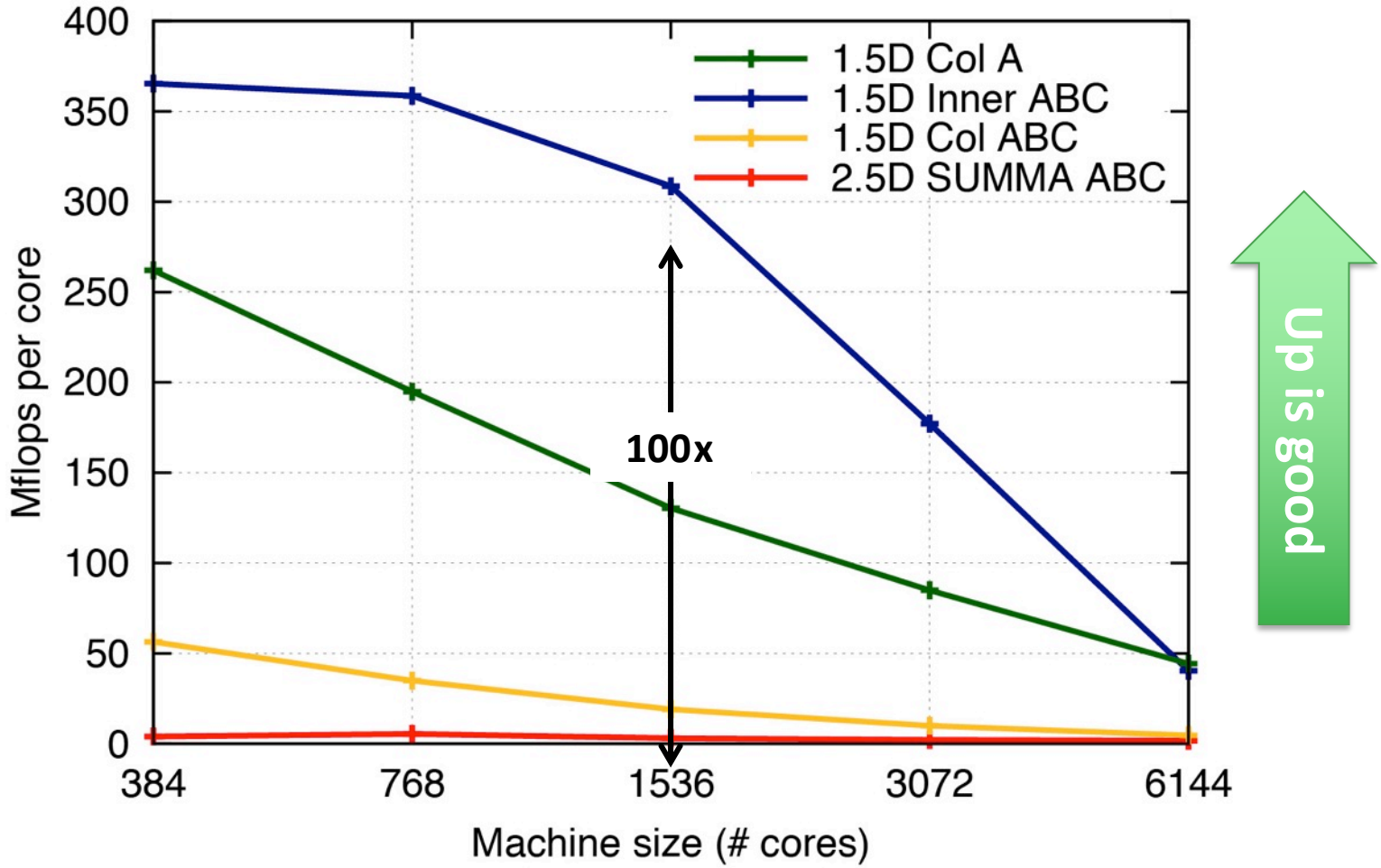
Sparse-Dense Matrix Multiply Too!



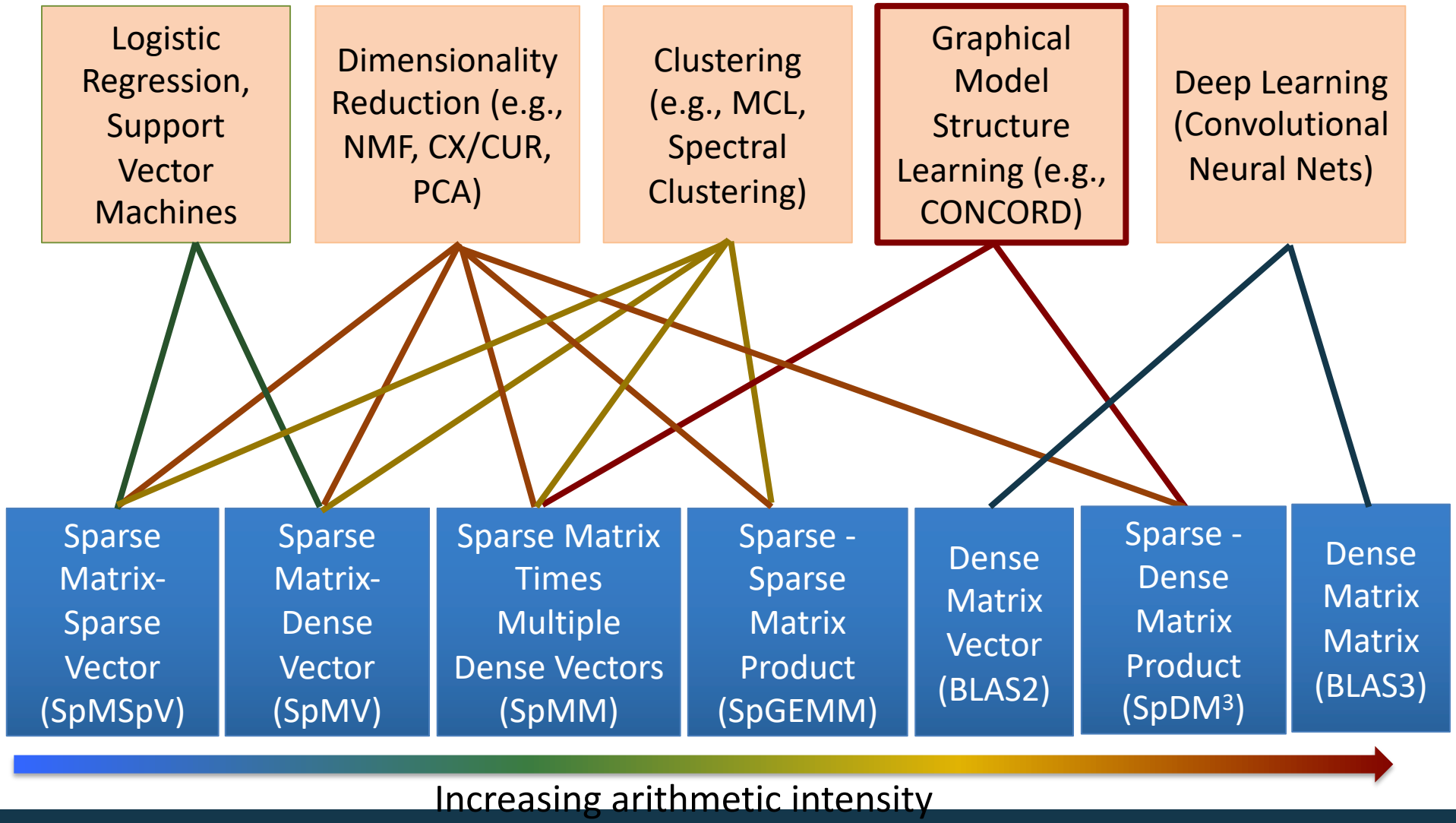
- **Variety of algorithms that divide in or 2 dimensions**

100x Improvement

- $A^{66k \times 172k}$, $B^{172k \times 66k}$, 0.0038% nnz, Cray XC30



Linear Algebra is important to Machine Learning too!



Problems in Computational Biology

- Graphical Models (Machine Learning)
- **Genome Assembly**
- **Many-to-Many Alignment**
- **Imaging**

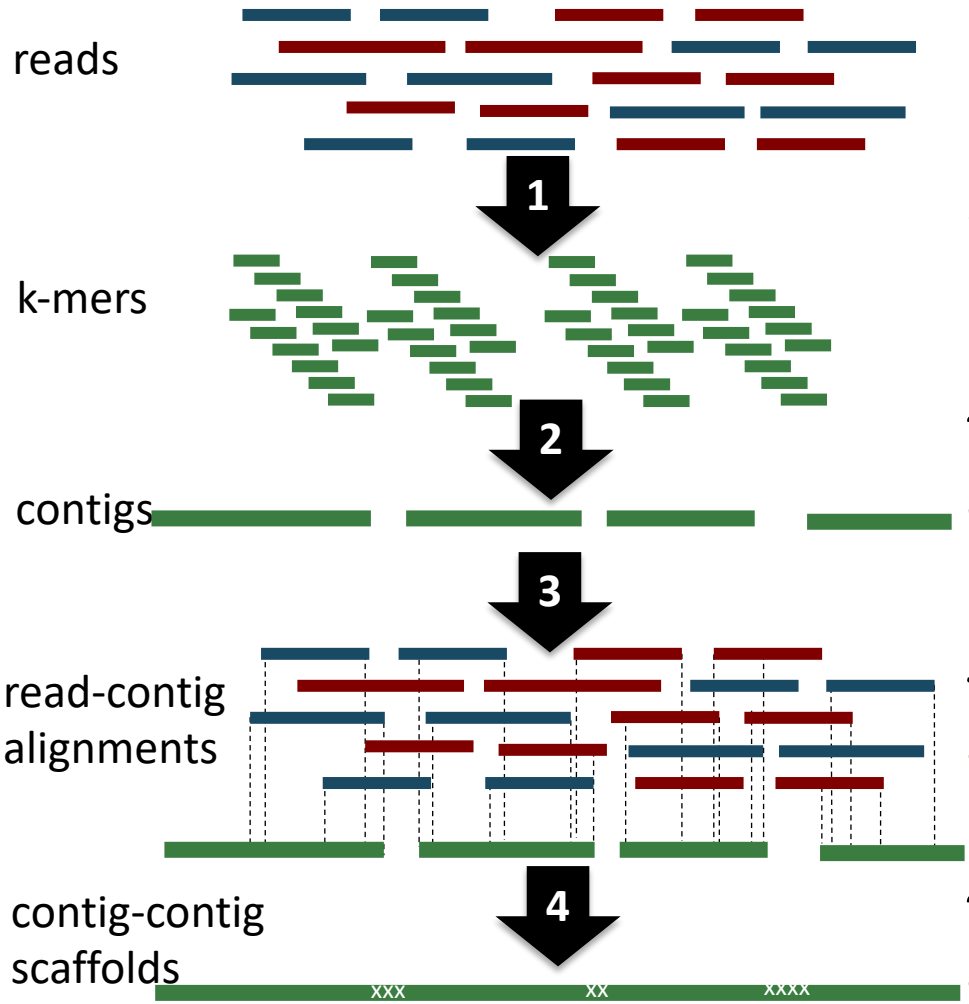
And how can we take ideas from numerical computing, linear algebra, and communication avoiding algorithms to this domain?

De novo Genome Assembly

- **De novo genome assembly: Reconstruct an unknown genome from a collection of short reads.**
  - Constructing a jigsaw puzzle without having the picture on the box

- **Metagenome assembly: 100s-1000s of species mixed together**

The HipMer genome assembly pipeline has 4 phases



1) K-mer Analysis

(synchronous) irregular all-to-all

2) Contig Generation

asynchronous remote insert (aggregate and overlap) and get

3) Alignment

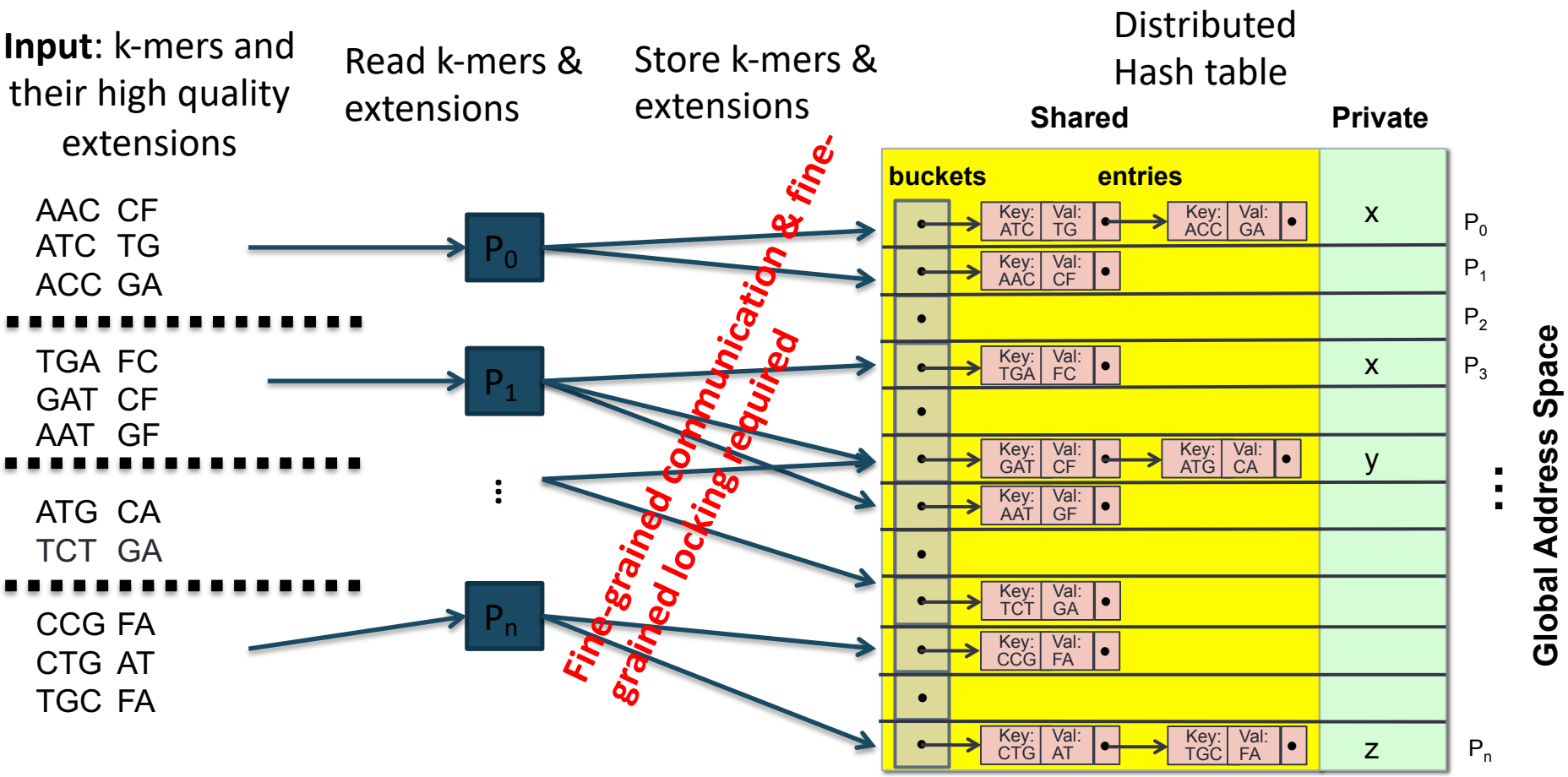
asynchronous remote insert and lookup (software caching)

4) Scaffolding & Gap Closing

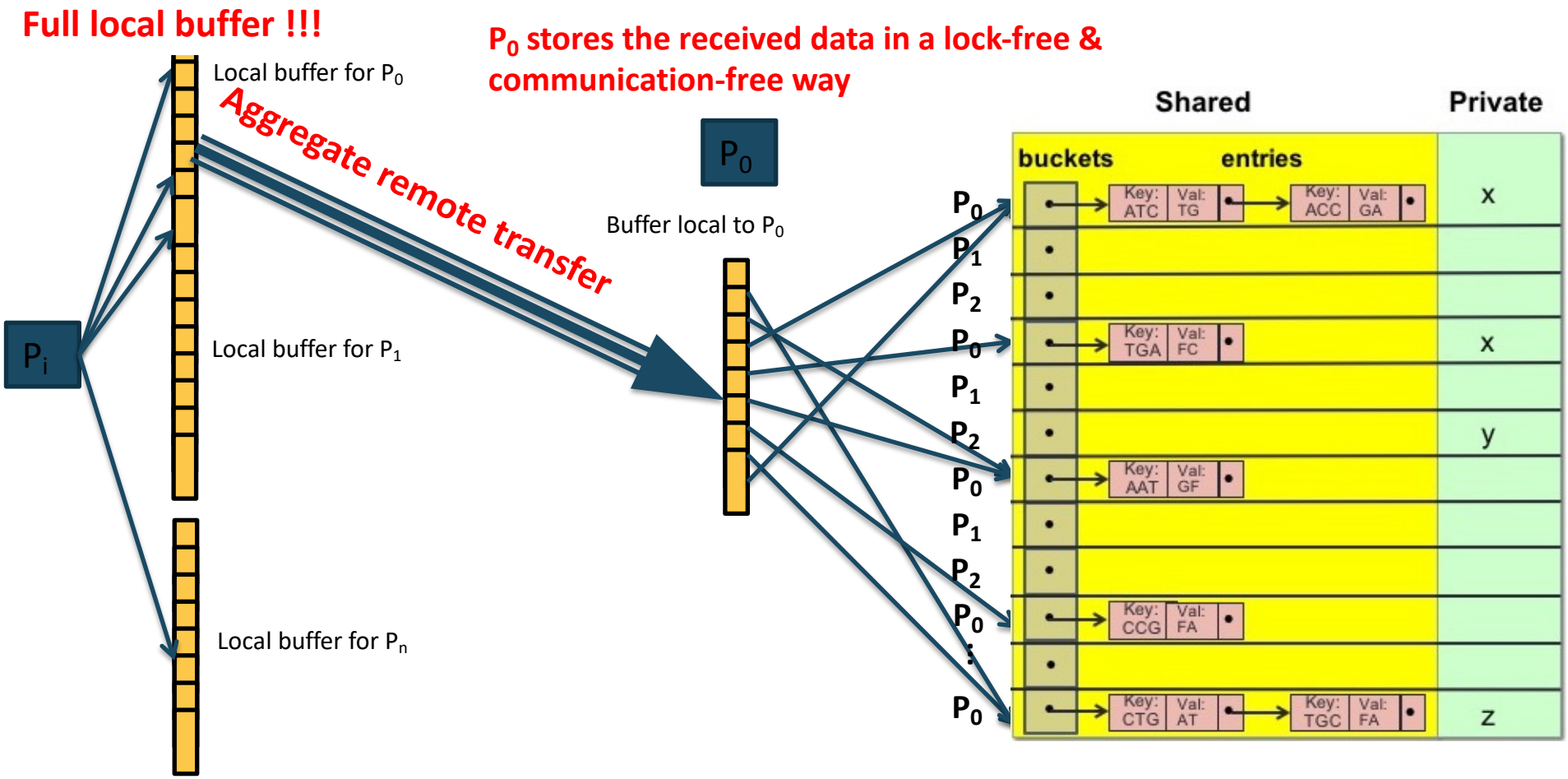
asynchronous remote insert and lookup (software caching)

Alignment also uses hash tables

Graph construction, traversal, and all later stages are written in UPC to take advantage of its global address space

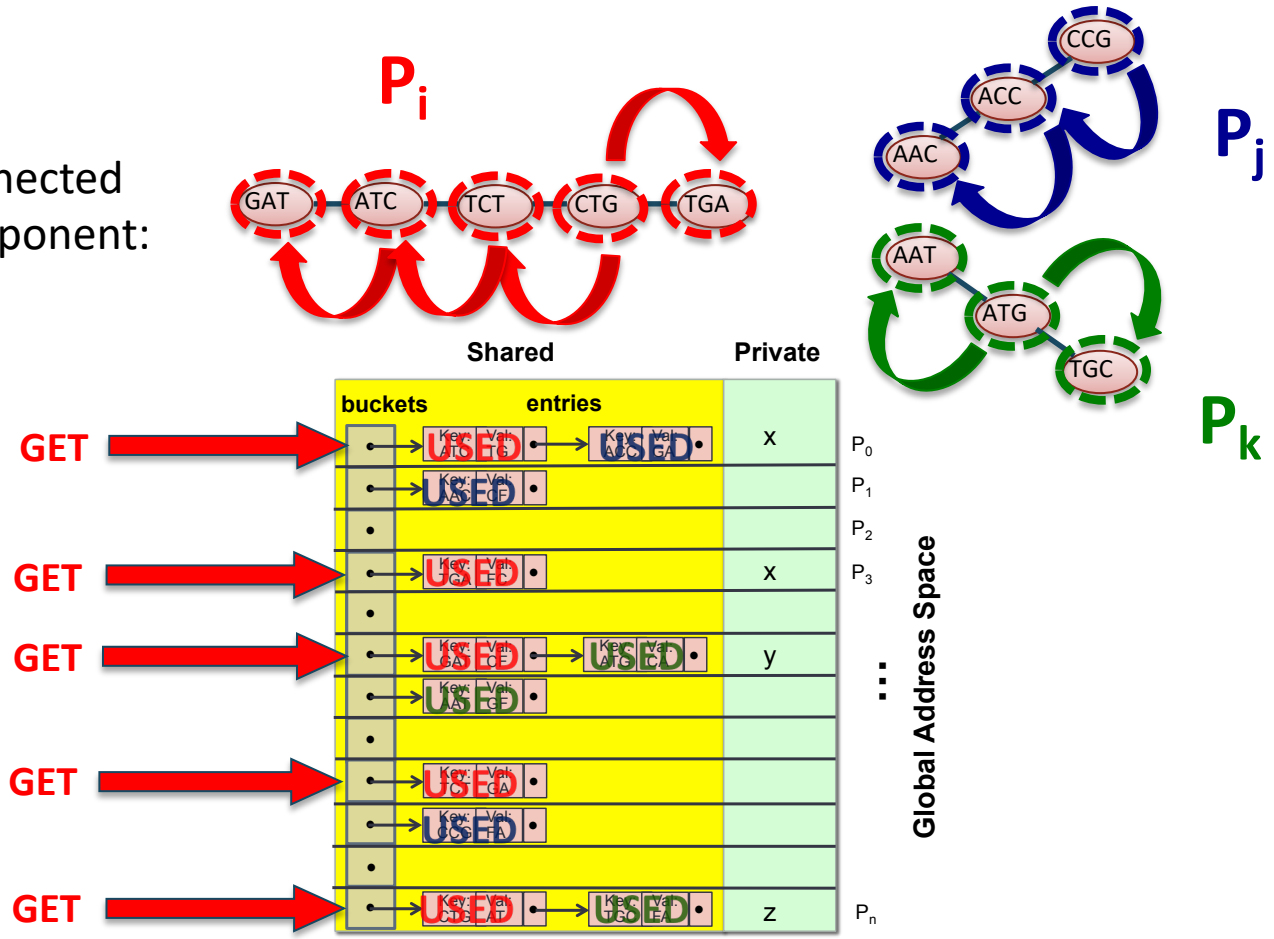


Hash Table Use Case 1: Global Update-Only phase

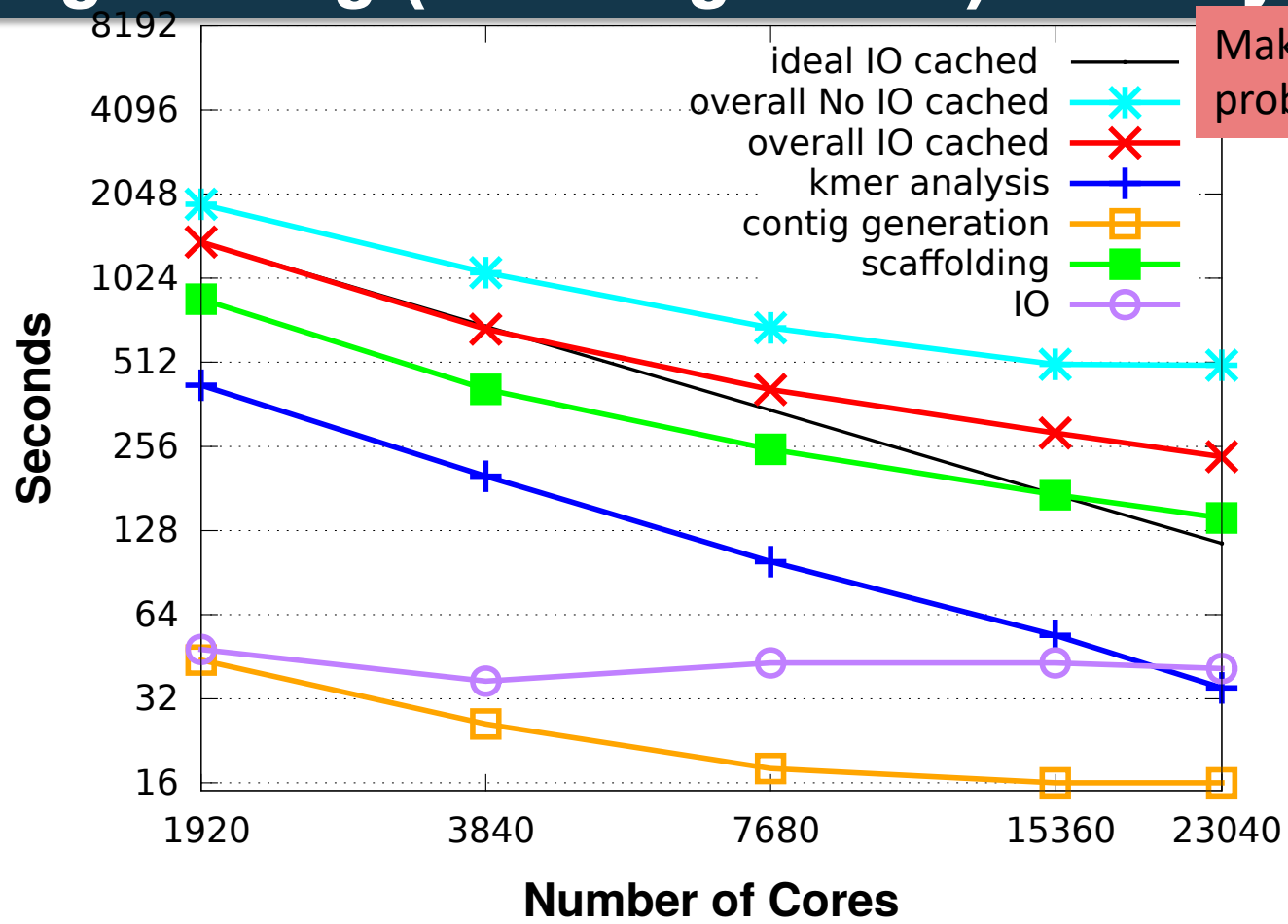


Hash Table Use Case 2: Global Reads & Writes phase

Connected component:



Strong scaling (human genome) on Cray XC30



Makes unsolvable problems solvable!

- Complete assembly of human genome in **4 minutes using 23K cores.**
- **700x speedup over** original Meraculous (took **2,880 minutes** on large shared memory with some Perl code); Some problems (wheat, squid, only run on HipMer version)

HipMer Used in Large Plant Assemblies

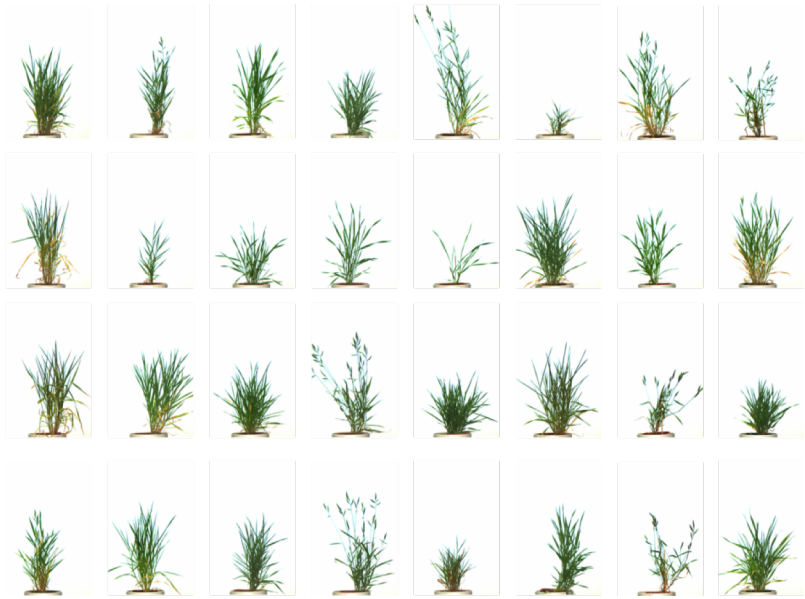
Genome	Red Cedar	Ceratopteris richardii	Sugar Cane
Est. Genome Size	20GB	11GB	10GB
Assembly Size ↑	9.8 GB	6.8 GB	5.0 GB
Scaffold N50 ↑	794.2 KB	132.7 KB	56.1 KB
Contig N50 ↑	73.6 KB	17.1 KB	4.4 KB

Goltsmann, Rokhsar et al

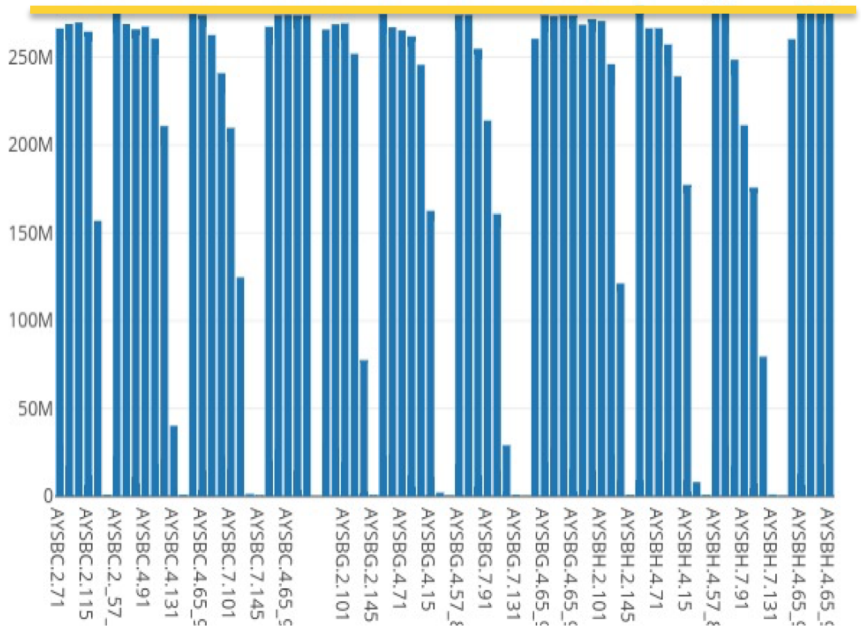
MetaHipMer used for pan genome study

54 de novo assemblies of the grass *Brachypodium distachyon*

- Nearly 2x the number of genes found in any individual genome
- Many “shell genes” species-wide are “core genes” within a subpopulation.



Assembled Genome Size 271 Mb



Different parameters over 10 samples

Sean Gordon (JGI, now Zymergen)

- MetaHipMer speed allows for parameter sweeps and produce higher quality results

MetaHipMer for Understanding an environmental microbiome

Best paper nominee for SC18 by the MetaHipMer team: *Evangelos Georganas, Rob Egan, Steven Hofmeyr, Eugene Goltsman, Bill Arndt, Andrew Tritt, Aydın Buluc, Leonid Oliker, Katherine Yelick*

Hardware and Programming Requirements

distributed hash tables *all the way down*...

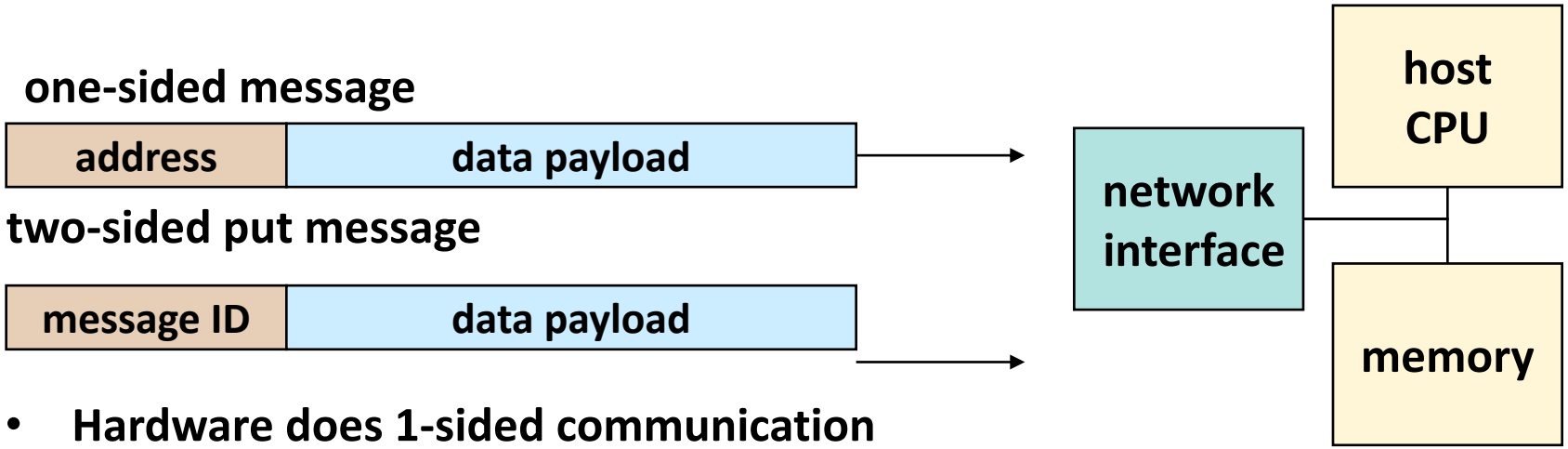
Or at least a global address space

- High injection rate networks
- High bisection bandwidth with modest-sized messages
- Remote (hardware) atomics
- Caching remote values sometimes useful (can be done in software)

Leverages hash table features

- Asynchronous random-access
- Inserts reordered (write-only phase)
- Lookups may involve marking elements (read-only phase)
- Good hash functions for load balance (and locality if genome ~known)

One-Sided Communication is Closer to Hardware



- Hardware does 1-sided communication
- Overhead for send/receive messaging is worse at exascale

Problems in Computational Biology

- Graphical Models (Machine Learning)
- Genome Assembly
- **Many-to-Many Alignment**
- **Imaging**

And how can we take ideas from numerical computing, linear algebra, and communication avoiding algorithms to this domain?

What is (Pairwise) Alignment?

Input

- **pair of sequences**
- **method for scoring a candidate alignment**

Output

- correspondences between substrings that maximize the score

```
ctgatcgtatctga

ctgcgaatccctga
```

```
ctgatcgtatc--tga
|||  ||X|||  |||
ctg--cgaatccctga
```

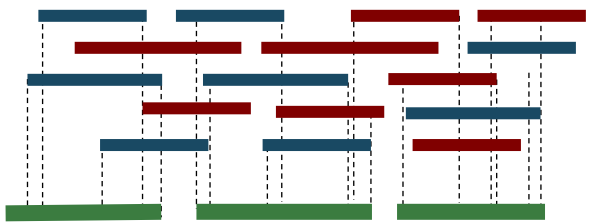
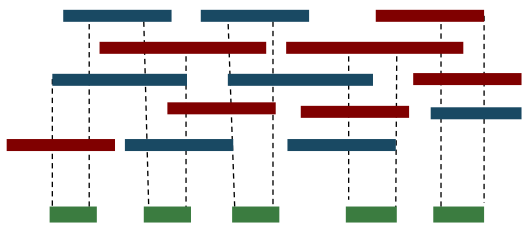
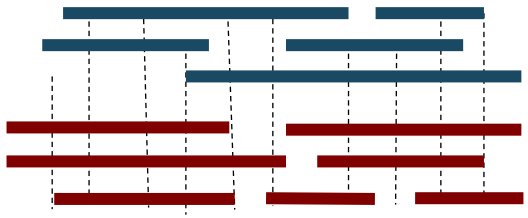
Match = +1
Mismatch = -1
Gap = -2

11 matches:	+11
1 mismatch:	-1
4 gaps:	-8
Score:	+2

Expensive: $O(m^2)$ for strings of length m (although can usually avoid worst case)

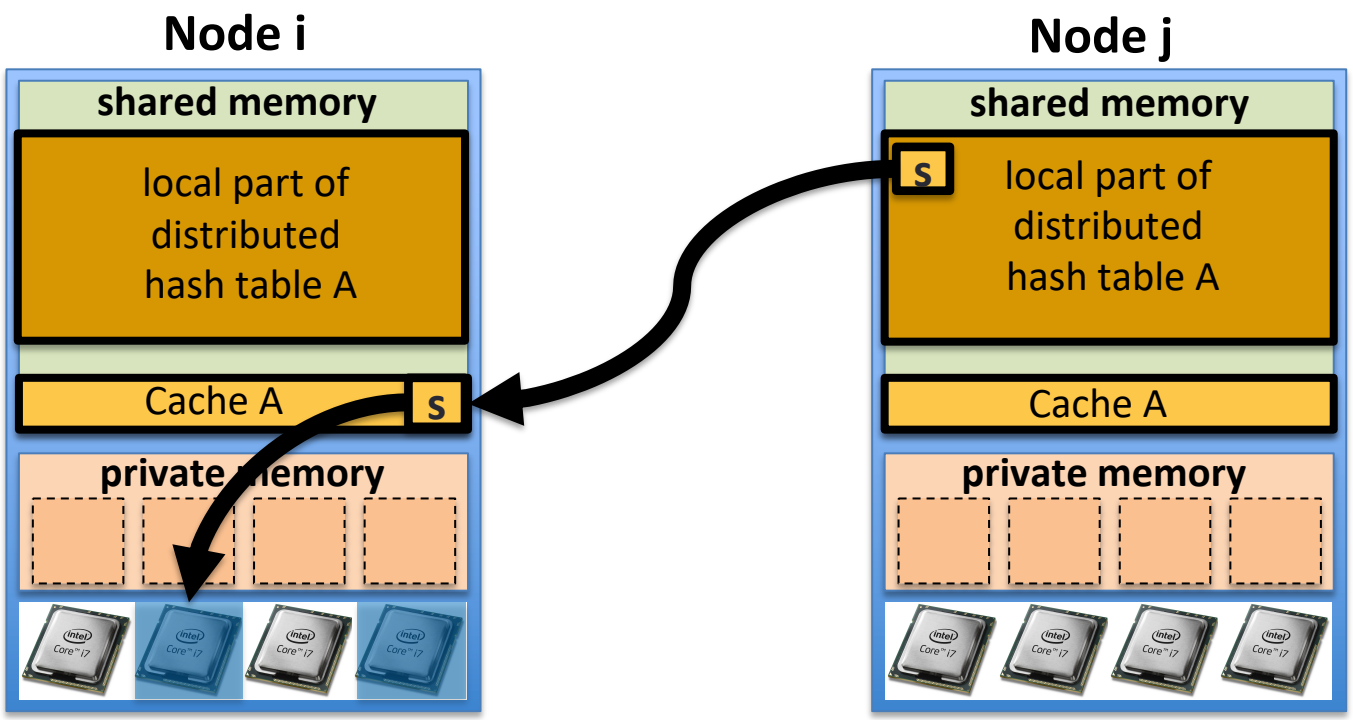
Alignment of Genomes

- **Given sets S and T, find the best alignment of all t ~~in~~ some T to all s in S**
  - Naïve algorithm embarrassingly parallel, O (|S| • |T|)

Assembly (HipMer)	Compare to reference	Long Reads (diBELLA)
		
Align reads to contigs	Align reads to some kind of reference	Align (long) reads to each other
O(100-200) x O(100-1M)	O(100-200) x O(1B) for human	O(10K) x O(10K)

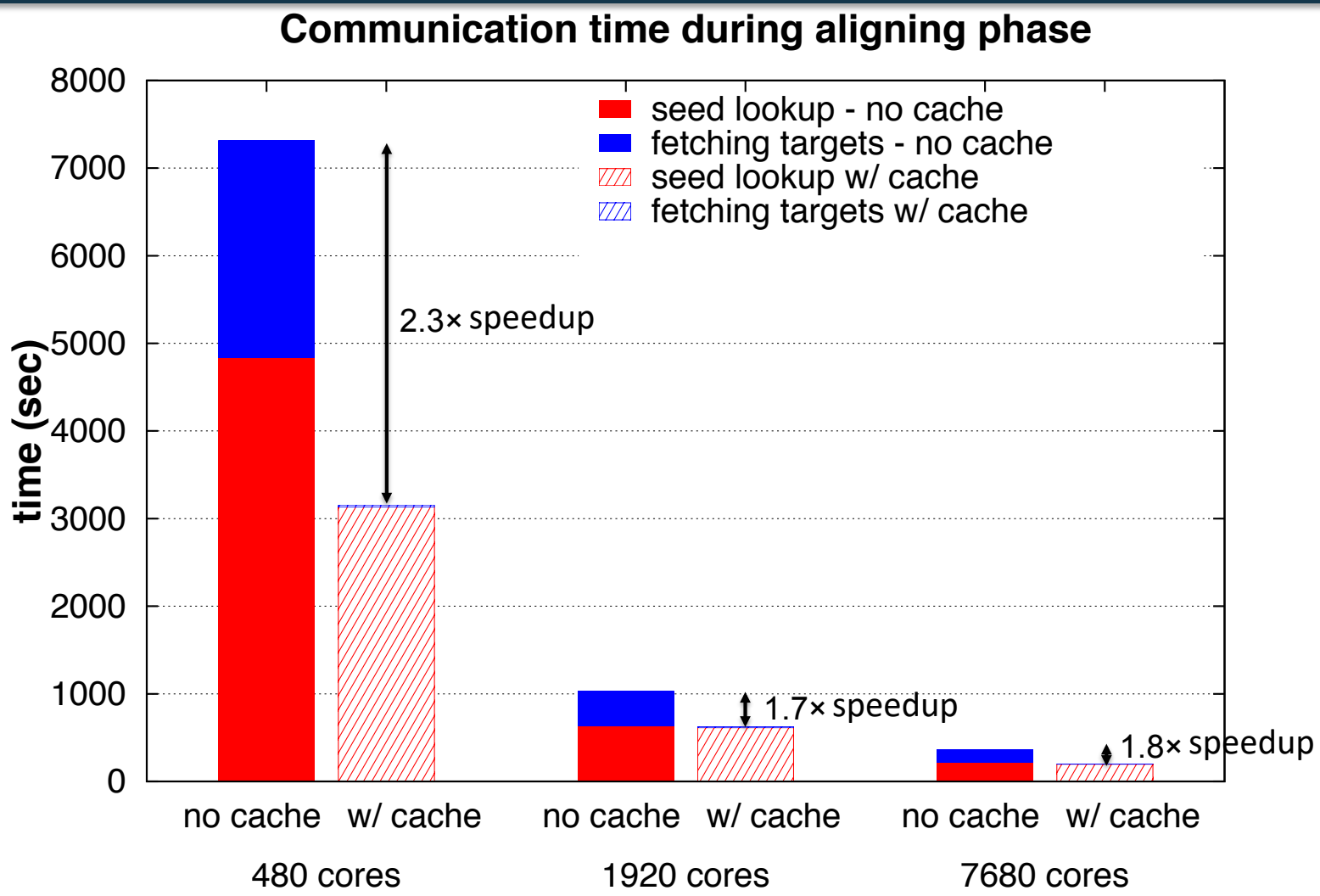
- **Parallelize across alignments, but which ones?**

Use case 3: Global Read-Only phase



CACHE HITS on read s

Communication reduction via software caching



- Software cache performance for human data set on Cray XC30

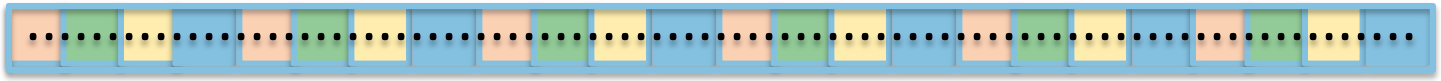
Using .5D ideas on All-to-All Alignment

- **n strings ("reads)**
  - Like molecules, stars in galaxies, etc.
- **Most common: 2-way N-body**

```
for t timesteps
    forall i_1, ..., i_k
        force[i_1] += interact(particle[i_1], ..., particle[i_k])
    forall i
        move(particle[i], force[i])
```

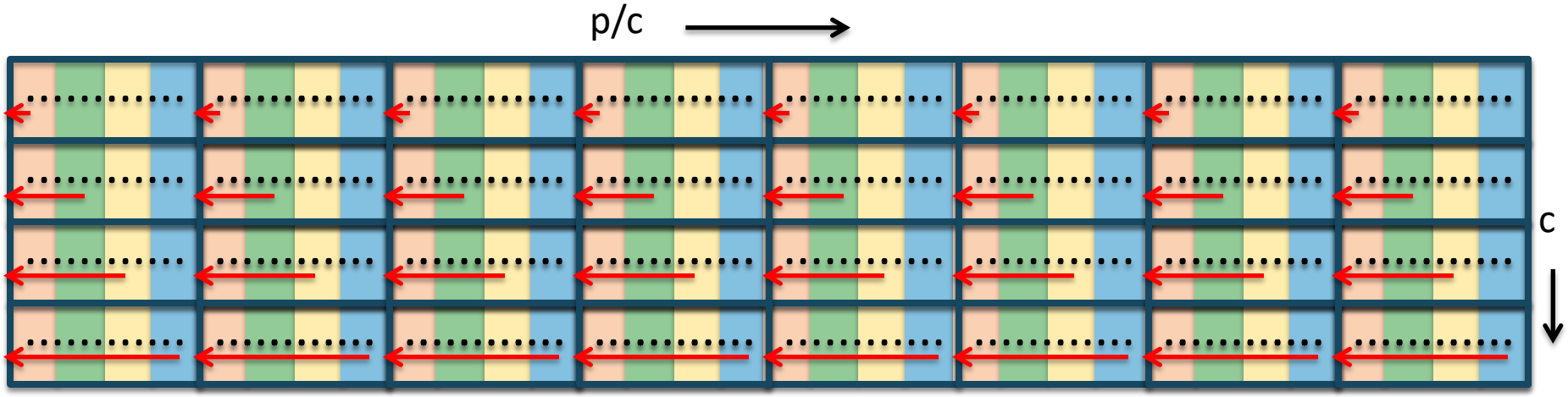
$O(n^k)$.

- **Best algorithm is to divide n things into p groups??**



No!

Communication Avoiding 2-way N-body
(using a "1.5D" decomposition)



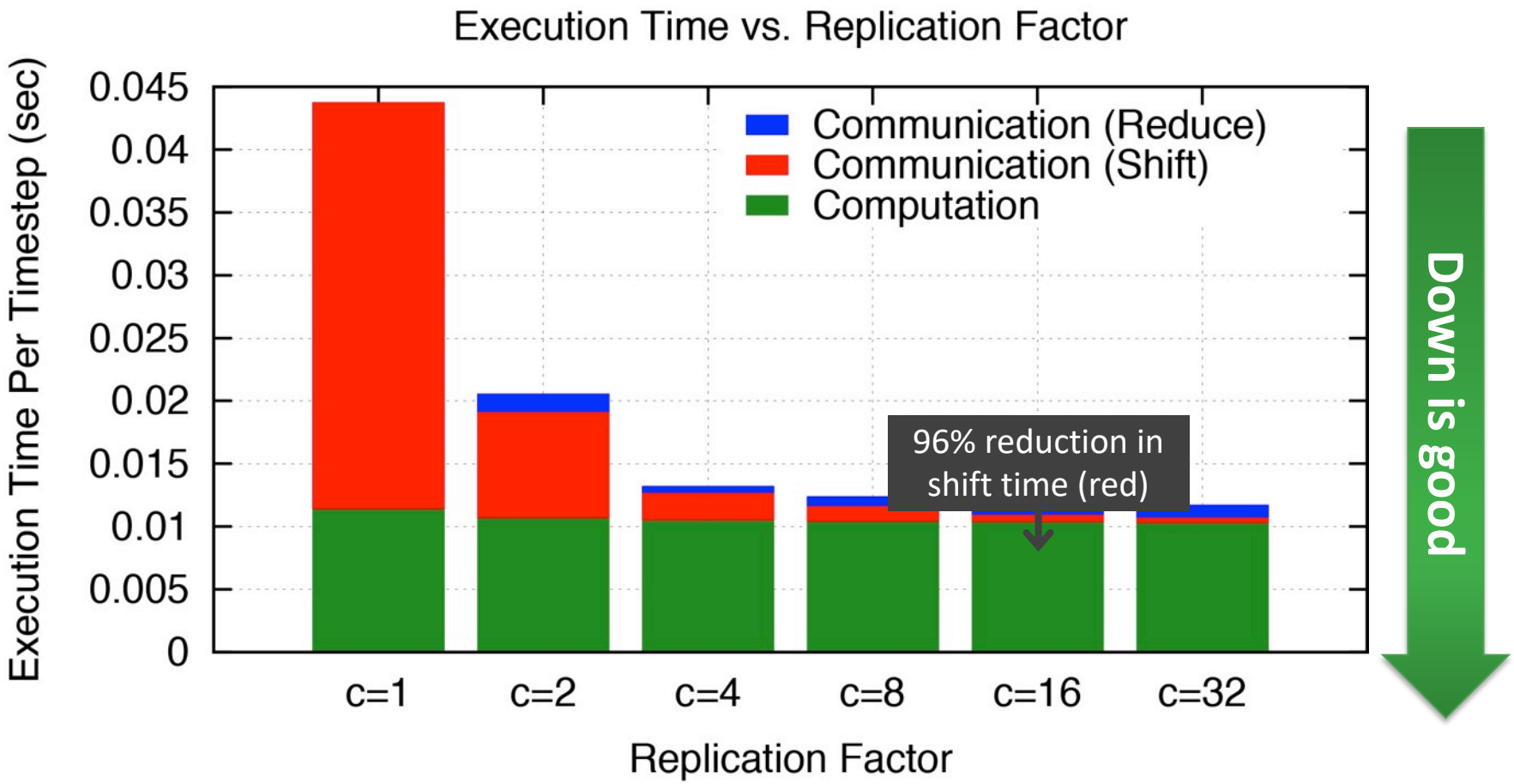
- **Divide p into c groups**
- **Replicate particles across groups**
- **Repeat: shift copy of n/(p*c) particles to the left within a group**
- **Reduce across c to produce final value for each particle**

Total Communication: O(log(p/c) + log c) messages,

O(n*(c/p+1/c)) words

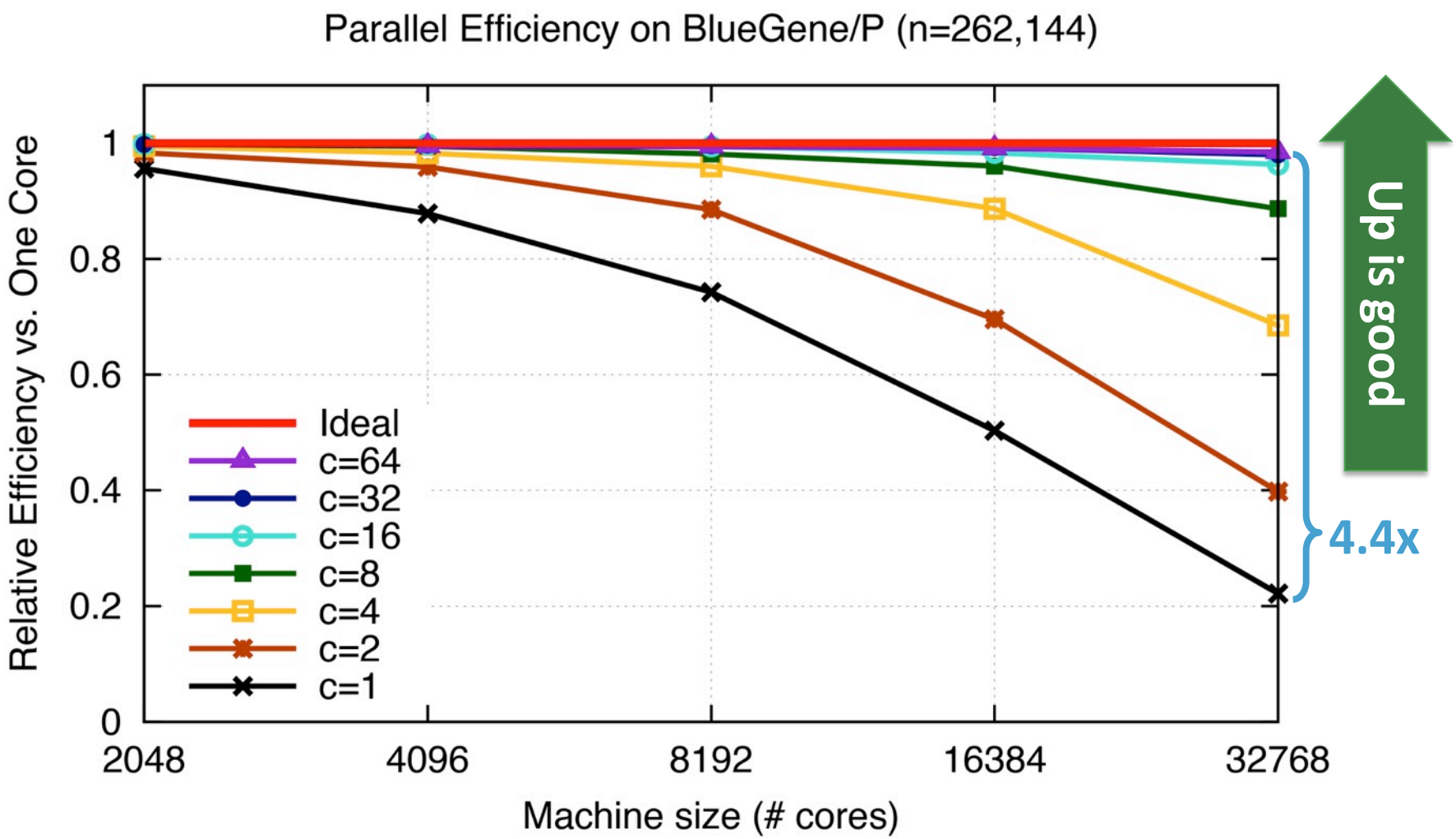
Less Communication..

Cray XE6; n=24K particles, p=6K cores



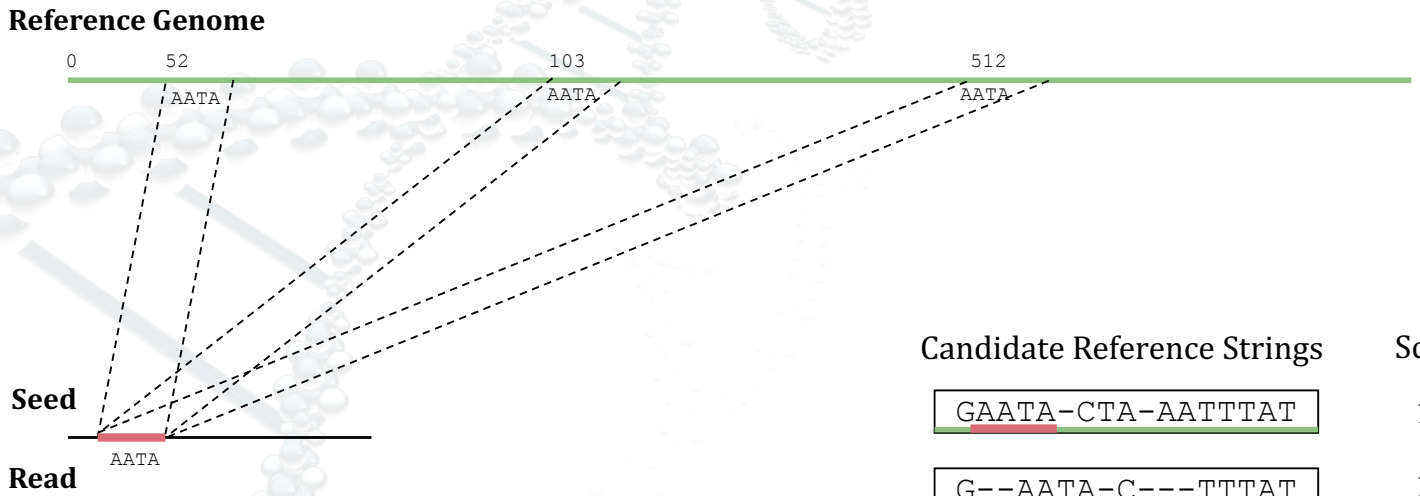
Koanantakool et al

Strong Scaling of 1.5D N-body



Seed-And-Extend to Avoid Full n^2

Seed Extension

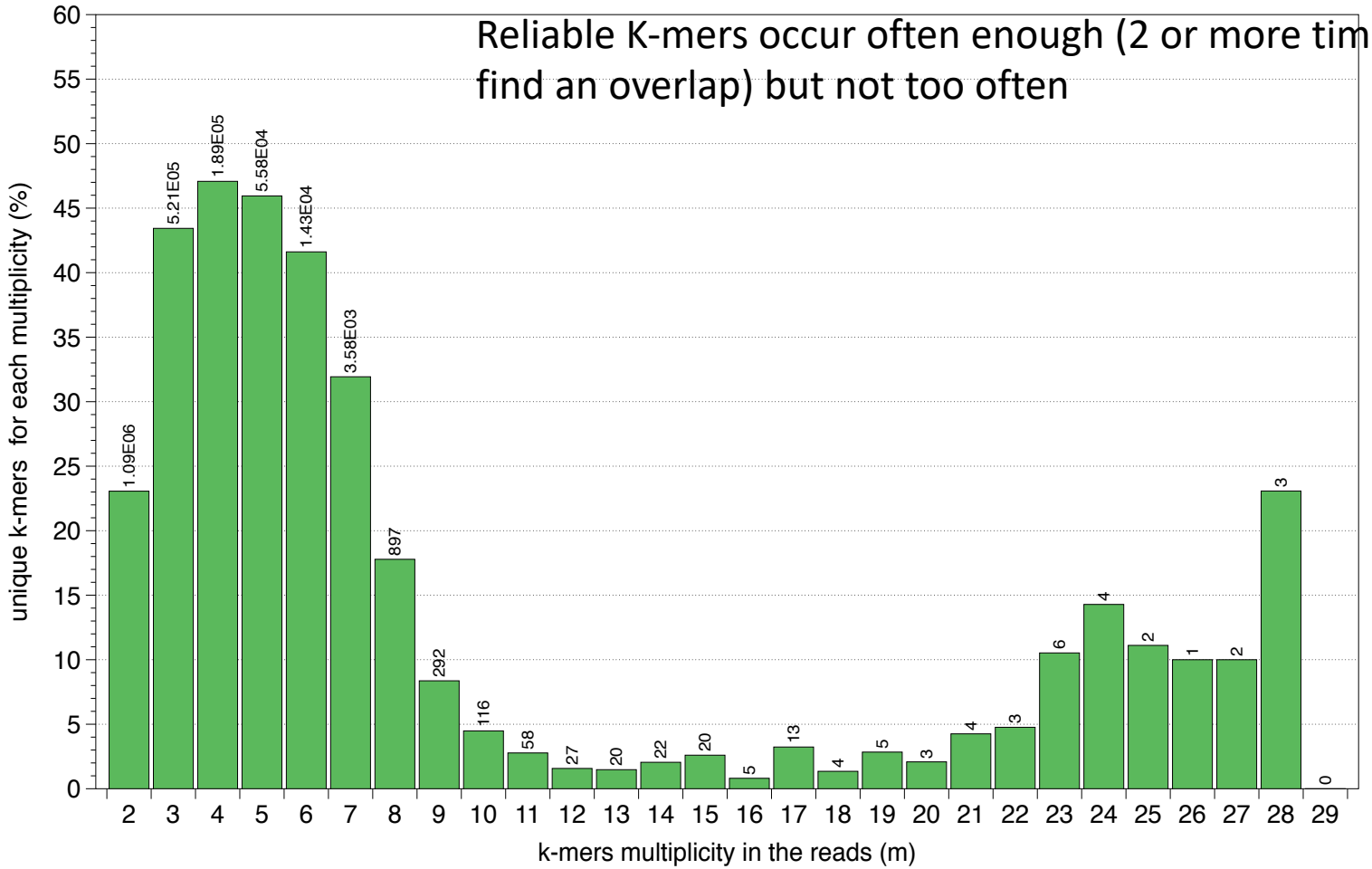


Candidate Reference Strings	Score
GAATA-CTA-AATTTAT	15
G--AATA-C---TTTAT	11
AAATACCTAAAATTTAT	17 ✓

Read AAATACCTAAAATTTAT

9

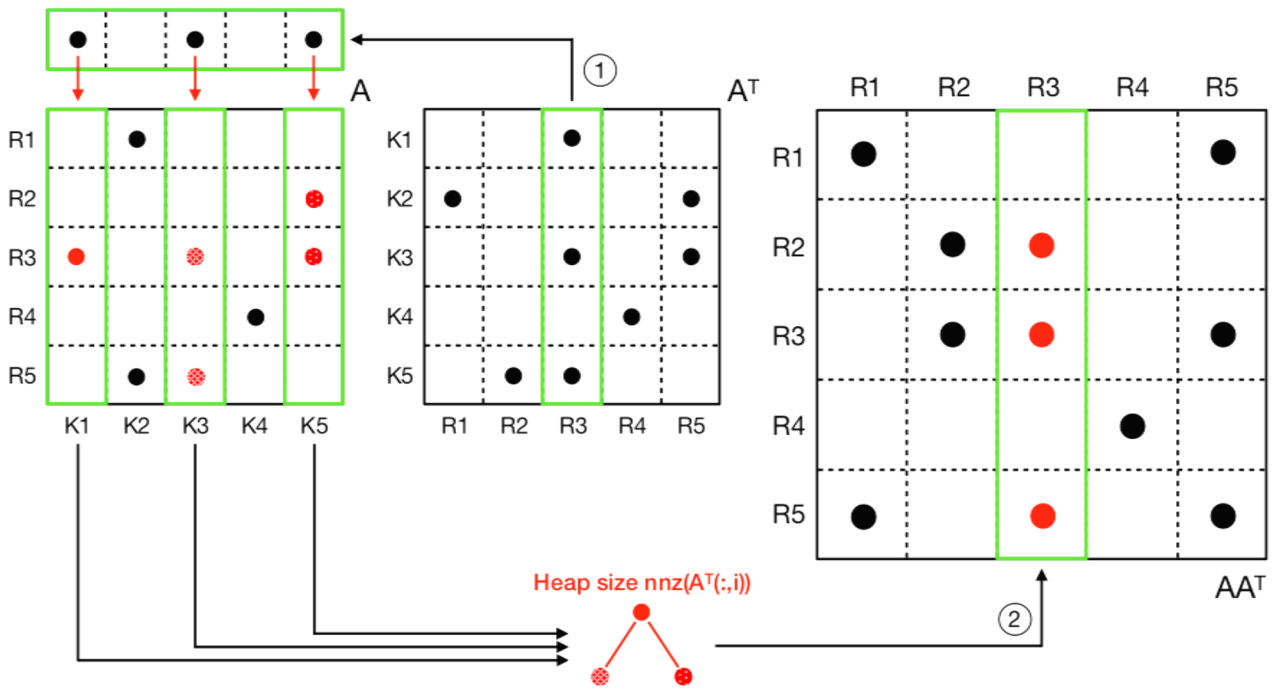
Use only "Reliable" K-mers



Giulia Guidi, Marquita Ellis, Kathy Yelick, Aydin Buluc

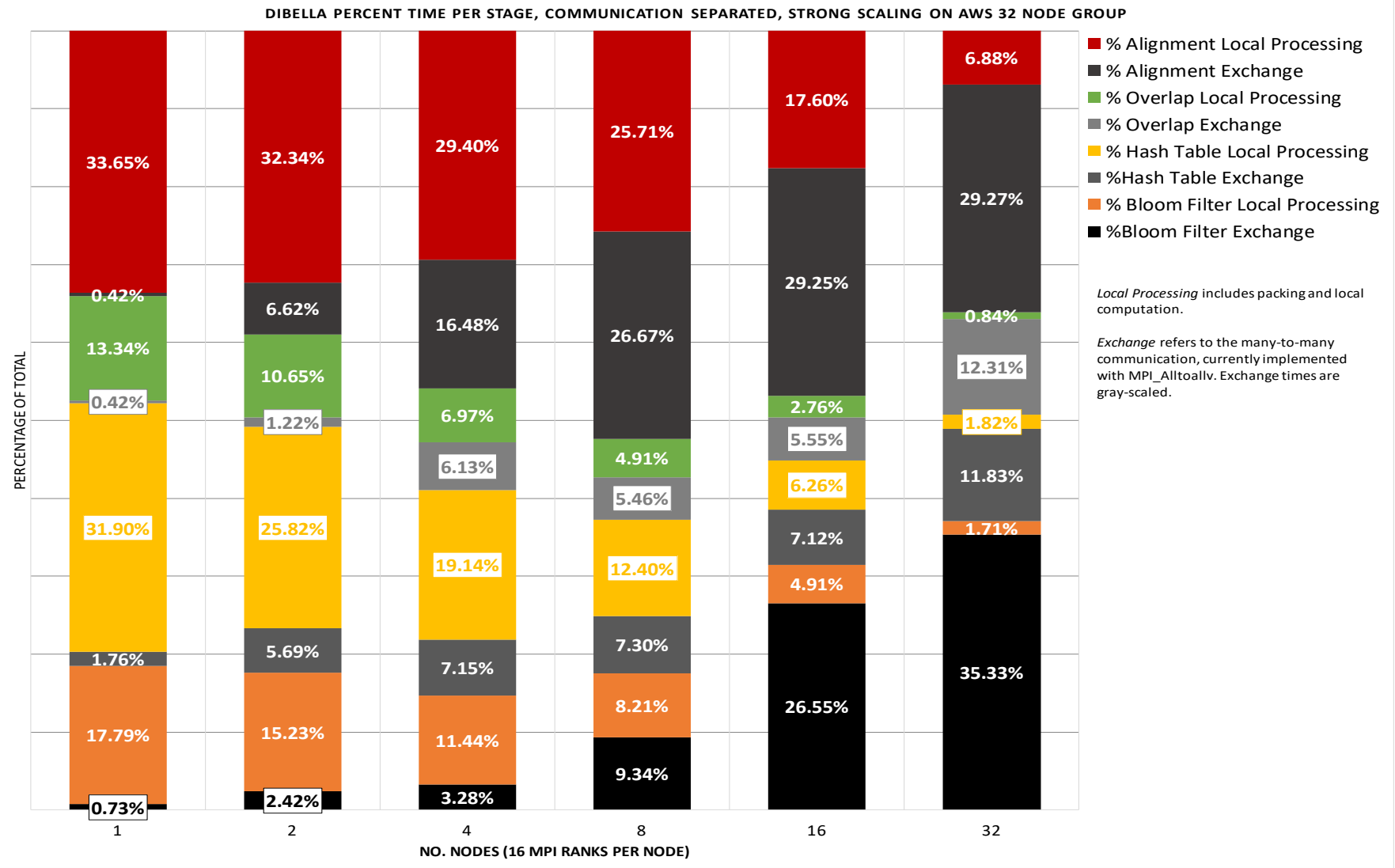
Set alignment is a "Join" / Multiply

We reformulate the problem of overlap detection in terms of a **sparse matrix-matrix multiplication**

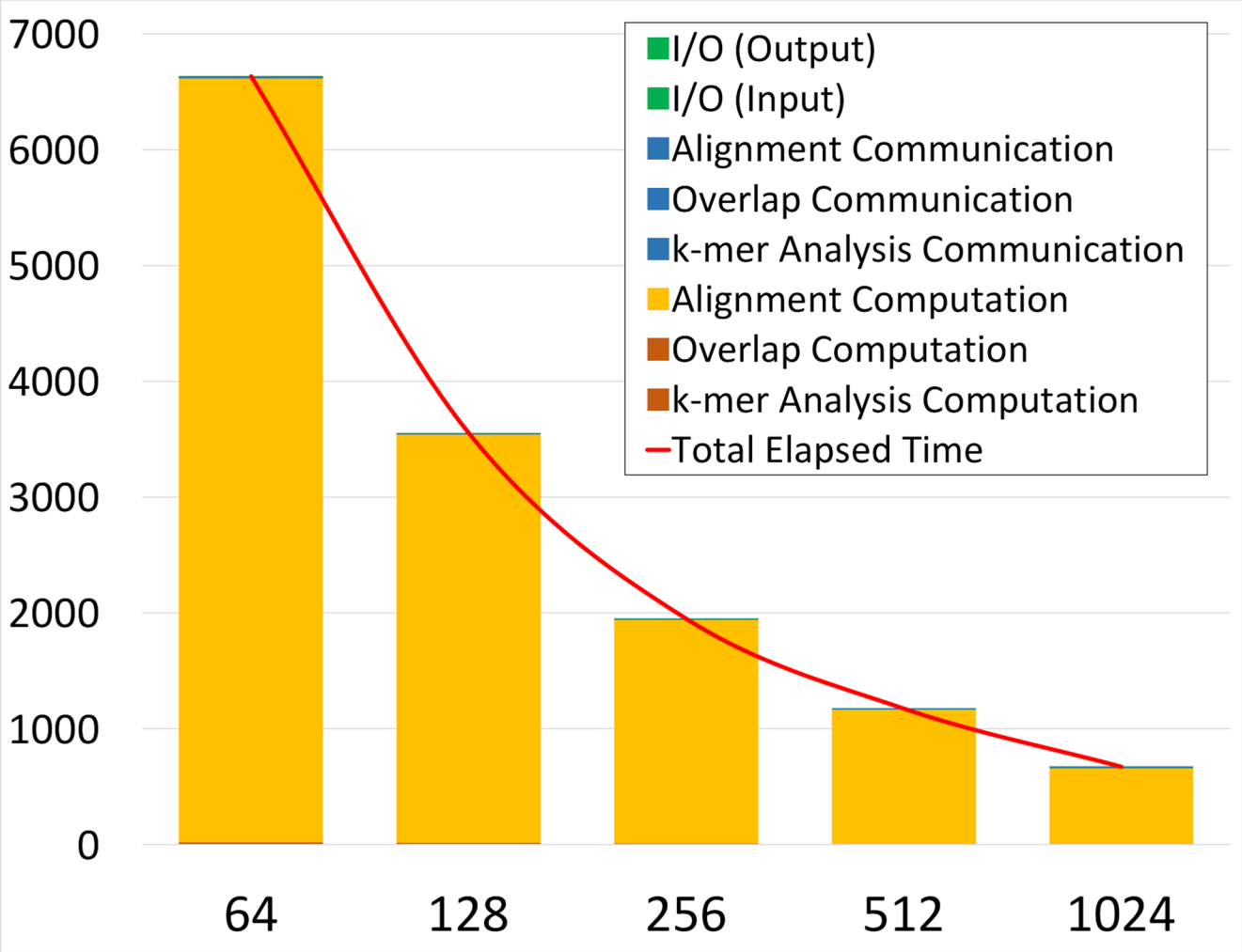


Our overlap detection was **coupled a with state-of-the-art seed-and-extend banded-alignment method** [1] to obtain a pairwise read alignment algorithm

Alignment cost for long (compute intensive) strings



Time breakdown on a Real HPC Machine

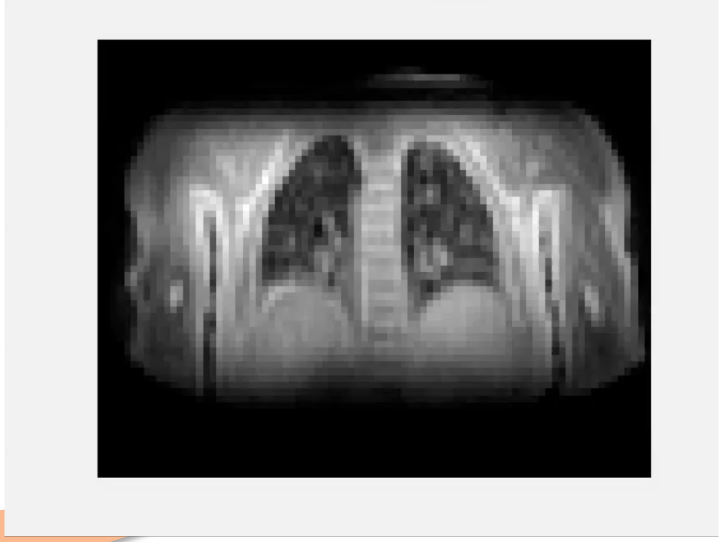


Problems in Computational Biology

- Graphical Models (Machine Learning)
- Genome Assembly
- Many-to-Many Alignment
- **Imaging**

And how can we take ideas from numerical computing, linear algebra, and communication avoiding algorithms to this domain?

Real-Time MRI Challenge



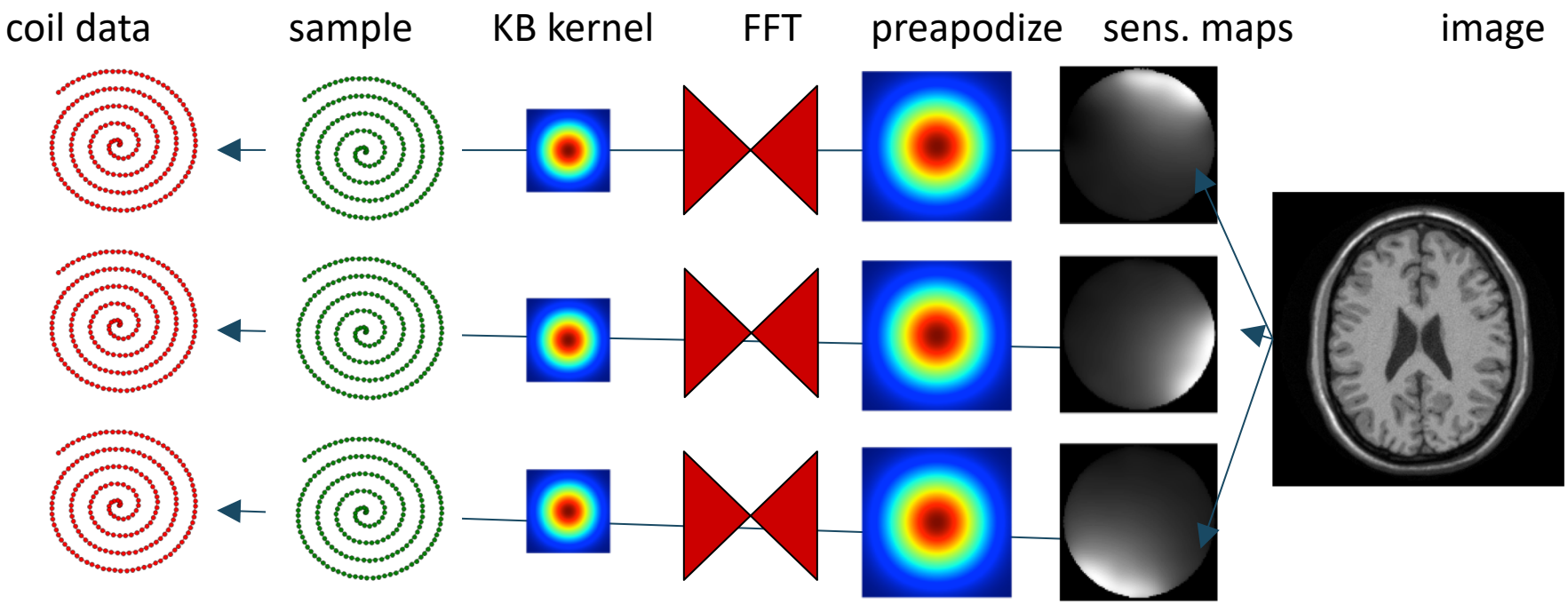
3 min goal

Time (min)	Architecture
6.15	KNL
5.42	Ivy Bridge
4.47	Broadwell
4.31	Kepler
4.12	Haswell
3.71	Broadwell
3.16	Kepler
0.94	Pascal

Michael Driscoll HPC optimization

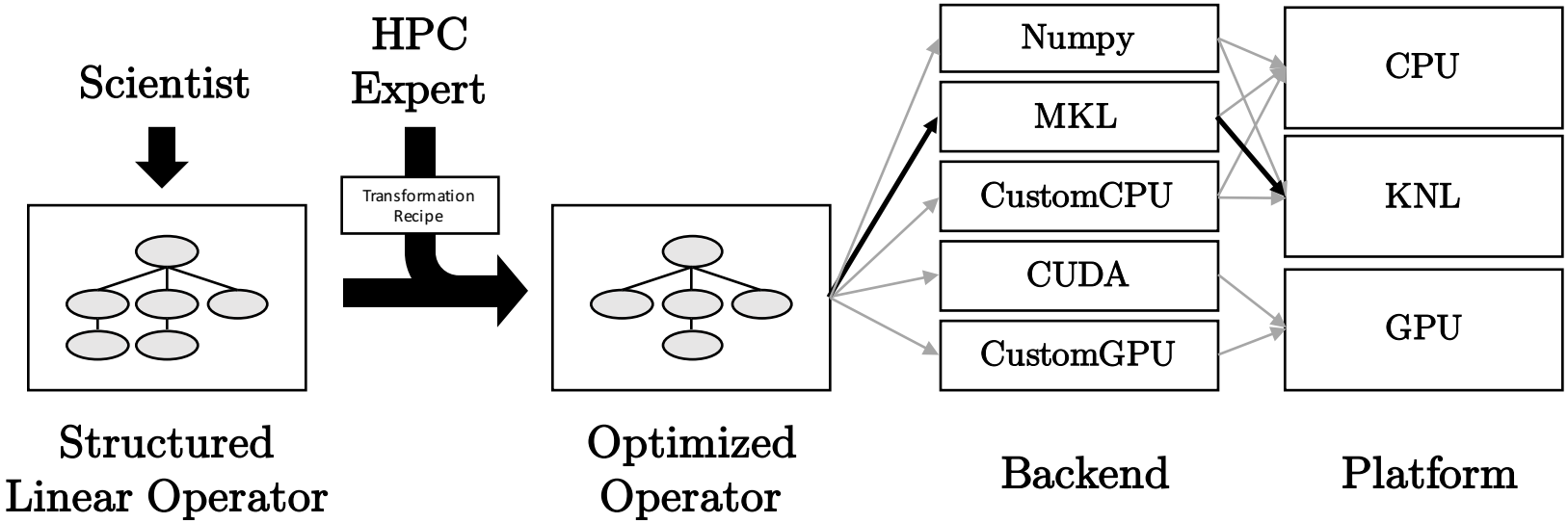
Compressed Sensing Approach by Mike Lustig et al
MRI results Wenwen Jiang

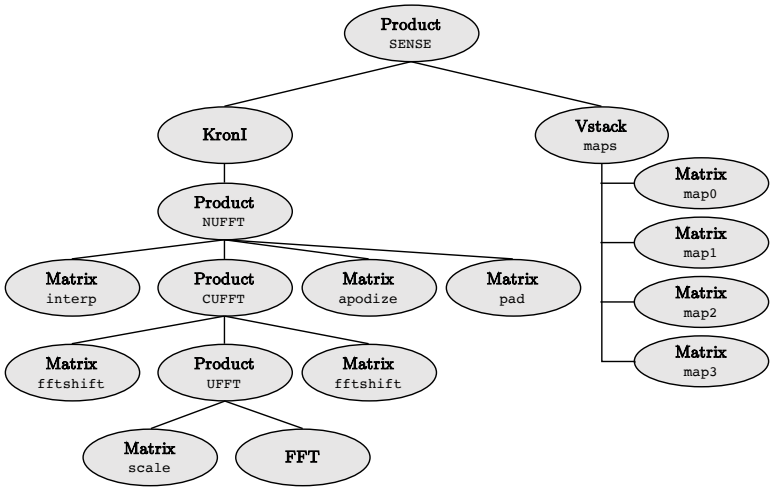
Matrix-free (loop optimization) vs. Matrix-full

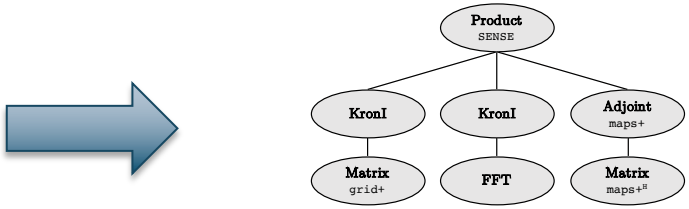


Loops	Structured matrices	Matrices
Operators as loop nests	Operators as matrices with structure that compiler can optimize	Operators as arbitrary sparse matrices

Domain-specific library with runtime optimizations







Tree transformation with matrix pattern knowledge

Image Reconstruction as a

Acquired Signal (known)

Imaging System Matrix (modeled)

Intrinsic Image (unknown)

y = A · x

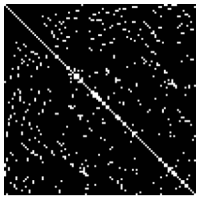
Dominated by linear operator evaluation

Conjugate gradient: $A^H y = A^H A x$

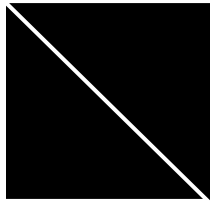
Convex optimization: minimize $| A^H A x - A^H y | + R(x)$

Indigo: A DSL for Image

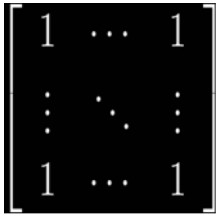
Matrices as building blocks



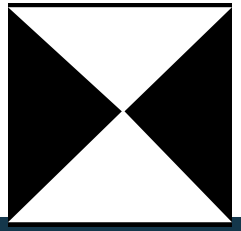
General Matrix



Identity Operator



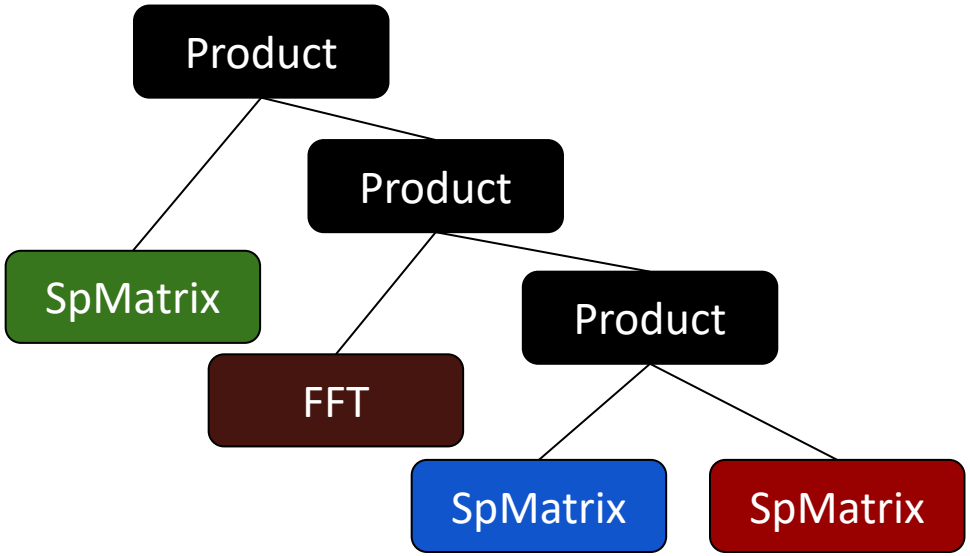
OneMatrix Operator



FFT Operator

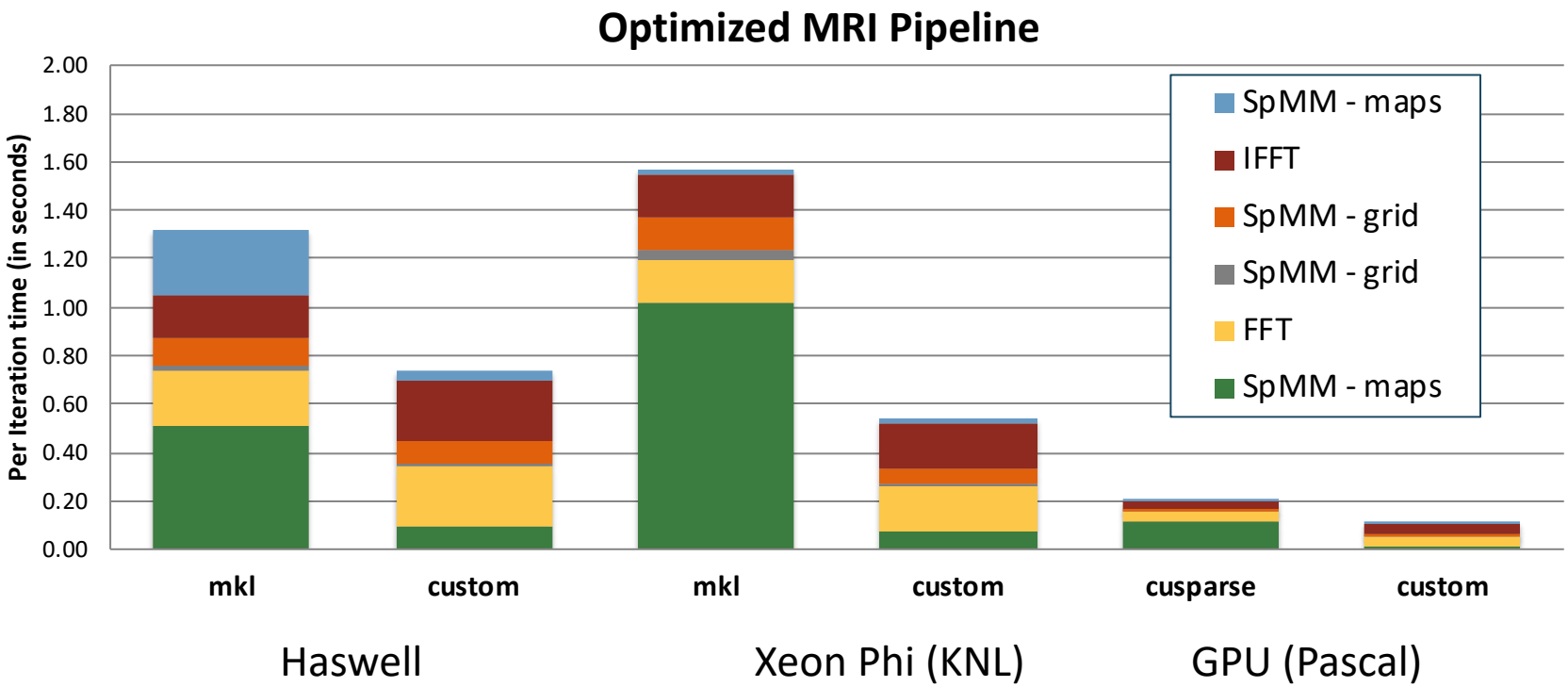
Operators at DGAs of matrix operations

- Arithmetic: Sum, Product, KroneckerProduct, Adjoint, Scale.
- Structural: VerticalStack, HorizontalStack, BlockDiagonal.



- Derived properties, e.g., 1 nonzero per row
- Transformations use the properties

Python-Based Domain-Specific Language (EDSL)



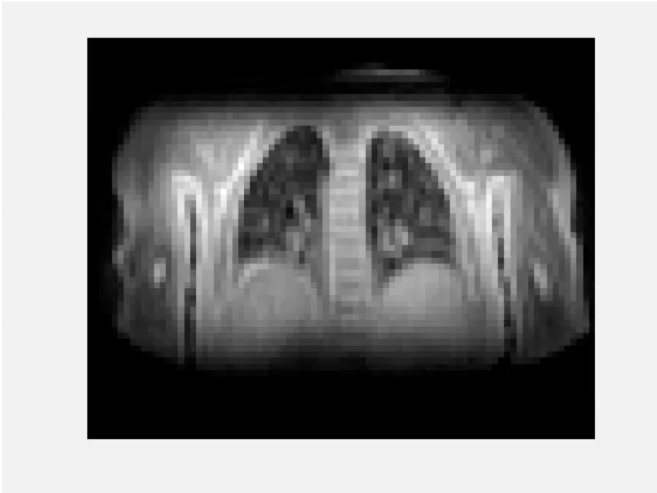
- **Original Numpy code on Haswell: 87 sec/iteration**
- **Runtime optimization reorganize tree of operators (matrices + FFTs) cognizant of matrix structure**
- **Library or custom matrix kernels**

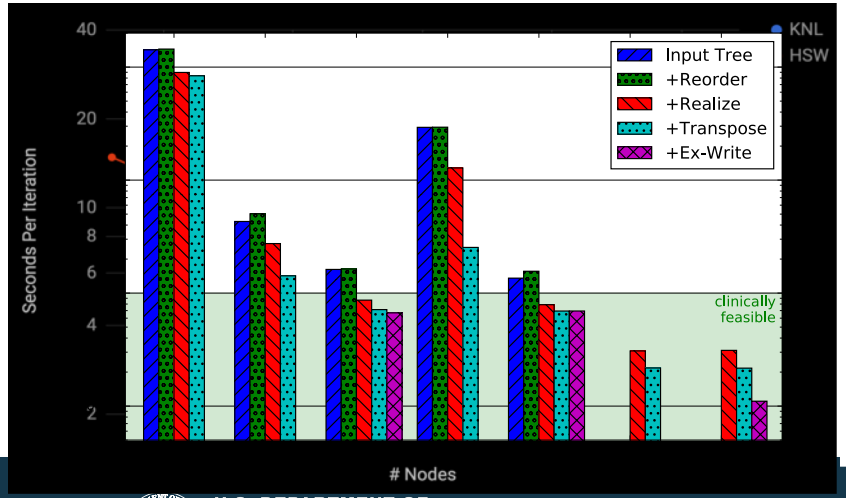
Driscoll et al, IPDPS 2018

Indigo Performance on GPUs, GPUs, Manycore

% peaks for for roofline, in this case memory bandwith peak

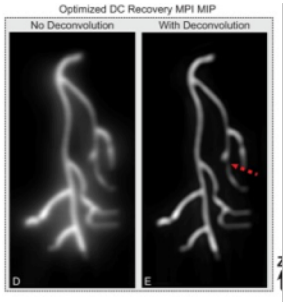
MRI reconstruction (Jiang, Lustig et al)





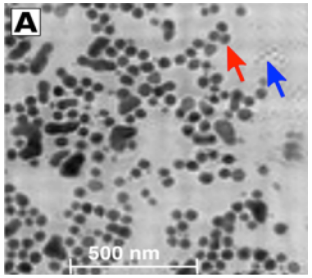
Exa goal (1 sec/iteration)

Magnetic Particle Imaging (Konkle et al 2015)



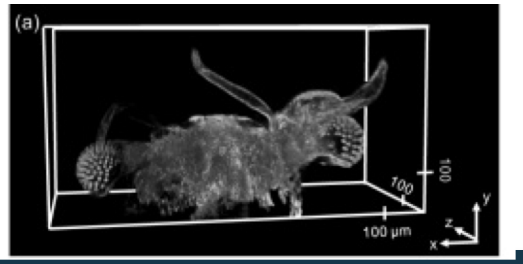
56% CPU peak,
9% KNL,
76% GPU.
258x over Numpy.

Ptychography (Marchesini 2016)



56% CPU peak,
9% KNL,
76% GPU.
258x over Numpy.

Phase-Space Microscopy (Liu et al 2017)



43% Peak CPU,
7% KNL,
46% GPU
186x over Numpy

Driscoll, PhD Thesis

Summary

- Biology has both regular and irregular problems
- Some (alignment / assembly) have no floating point
- Even the regular all-to-all style algorithms look irregular when well-optimized (for computation)
- A bad machine (aka cloud) can make even compute-intensive problems communication-limited
- Linear algebra appears in many forms
- Matrices are sparser than ever (and less regular)
- Communication avoidance: it's not just for linear algebra

Communication Hurts!