



# ***Genomic Analysis at Scale: Mapping Irregular Computations to Advanced Architectures***

**Kathy Yelick**

**Robert S. Pepper Distinguished Professor of EECS**

**Vice Chancellor for Research  
UC Berkeley**

**Senior Faculty Scientist**

**Lawrence Berkeley National Laboratory**

# 2018 ACM Turing Award for Deep Learning



Yoshua Bengio  
Photo: Facebook



Yann LeCun  
Photo: Google

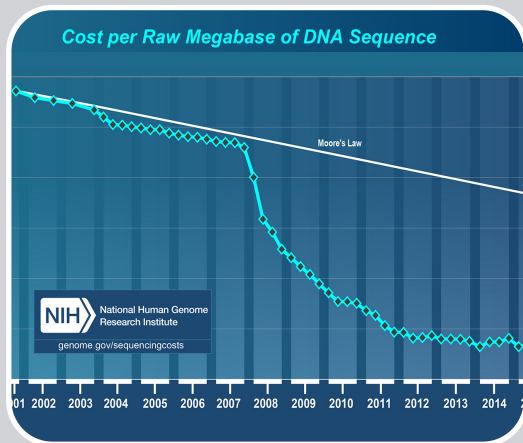


Geoffrey Hinton  
Photo: Botler AI

**Hinton's Turing Lecture:**  
*"So I think a lot of the credit for deep learning really goes to the people who collected the big databases like Fei Fei Li and the people who made the computers go fast like David Patterson and others."*



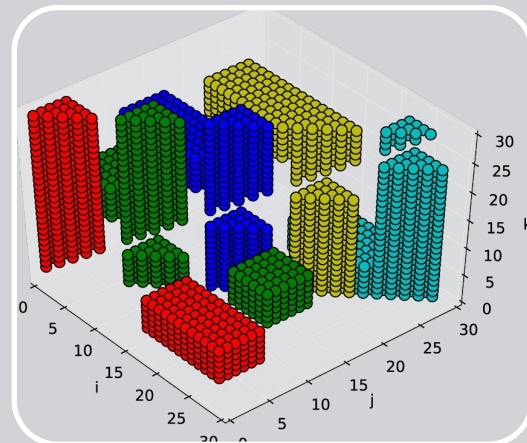
# Other areas where big data + big machines win?



Big Data

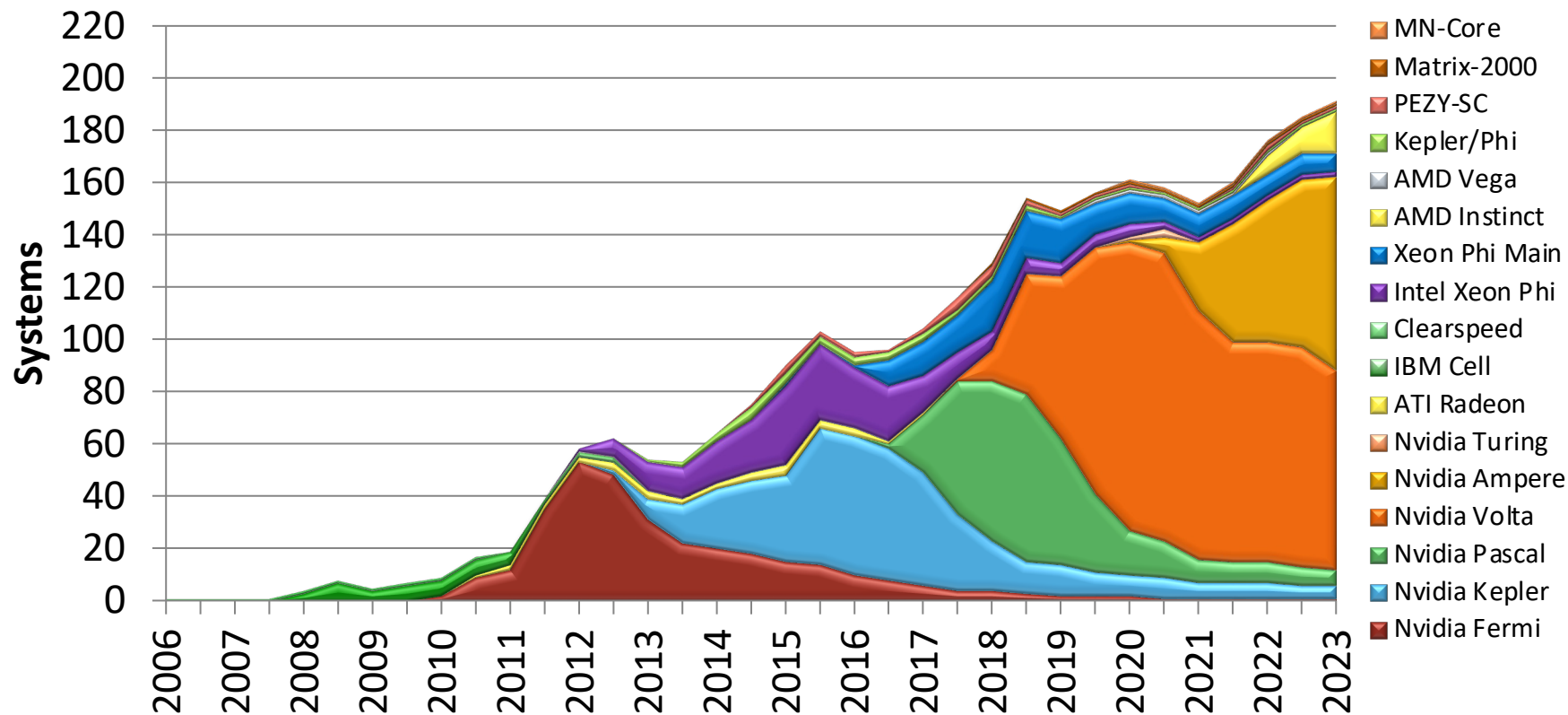


✓ Big Machines



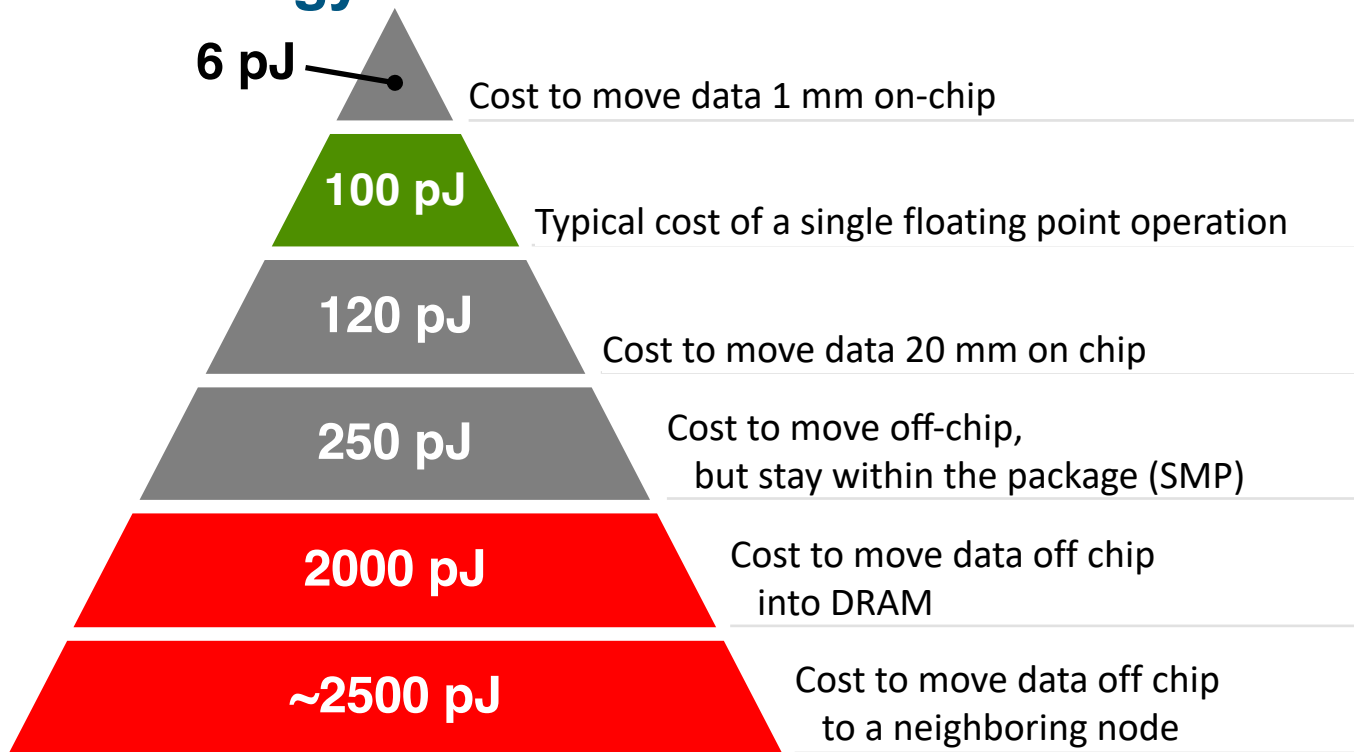
Scalable Algorithms

# Accelerators

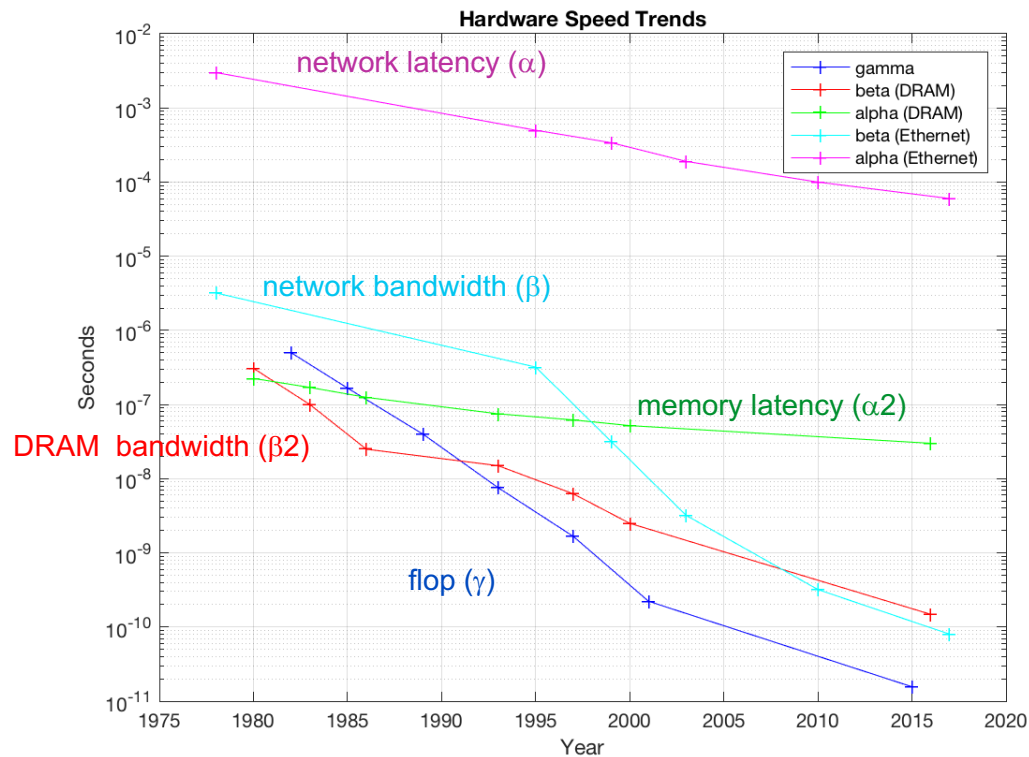


# Data Movement is Expensive

## Hierarchical energy costs.



# Communication Dominates: Dennard was too good



Time =

$$\# \text{ flops} * \gamma +$$

$$\# \text{ message} * \alpha +$$

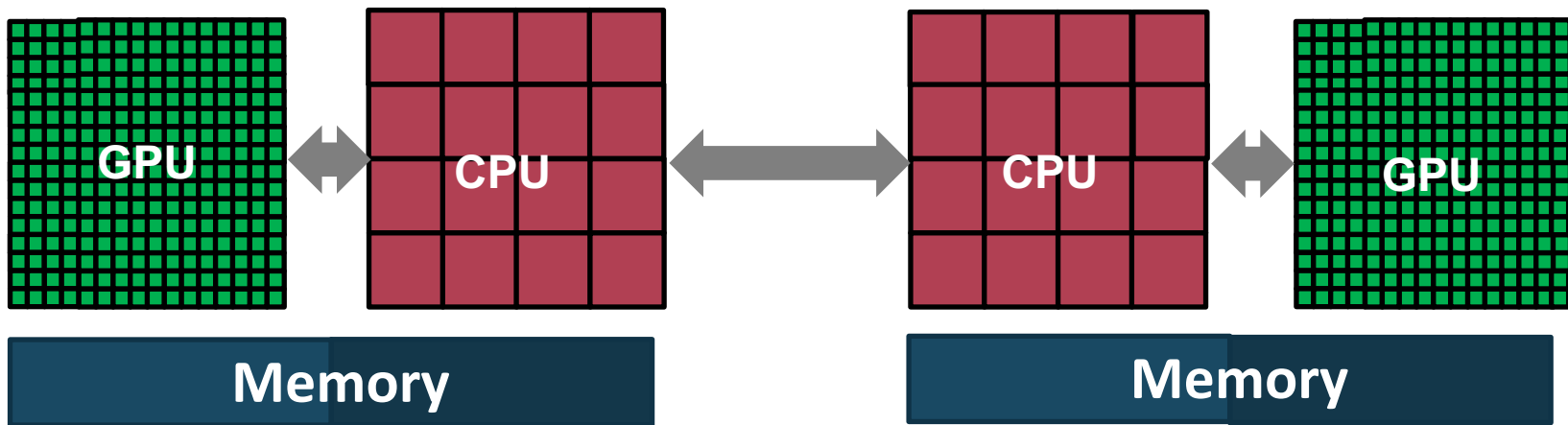
$$\# \text{ bytes comm} * \beta +$$

$$\# \text{ diff memory locs} * \alpha_2 +$$

$$\# \text{ memory words} * \beta_2$$

Data from Hennessy / Patterson, Graph from Demmel

# Put the GPUs in Charge



More  
cores

More data  
parallelism

Narrow  
data types

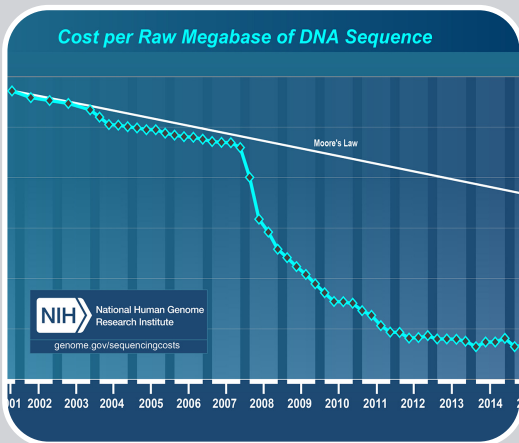
More  
memory  
spaces

CPUs in  
control

CPUs  
communicate



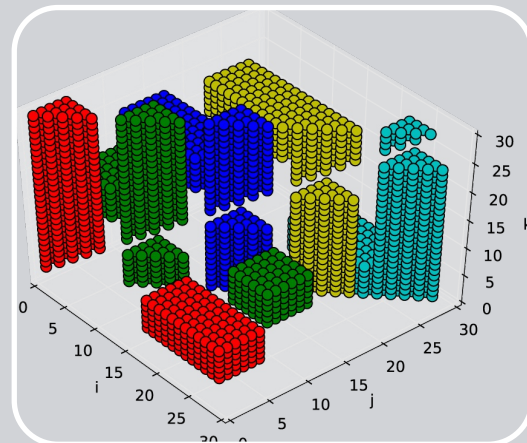
# Other areas where big data + big machines win?



Big Data



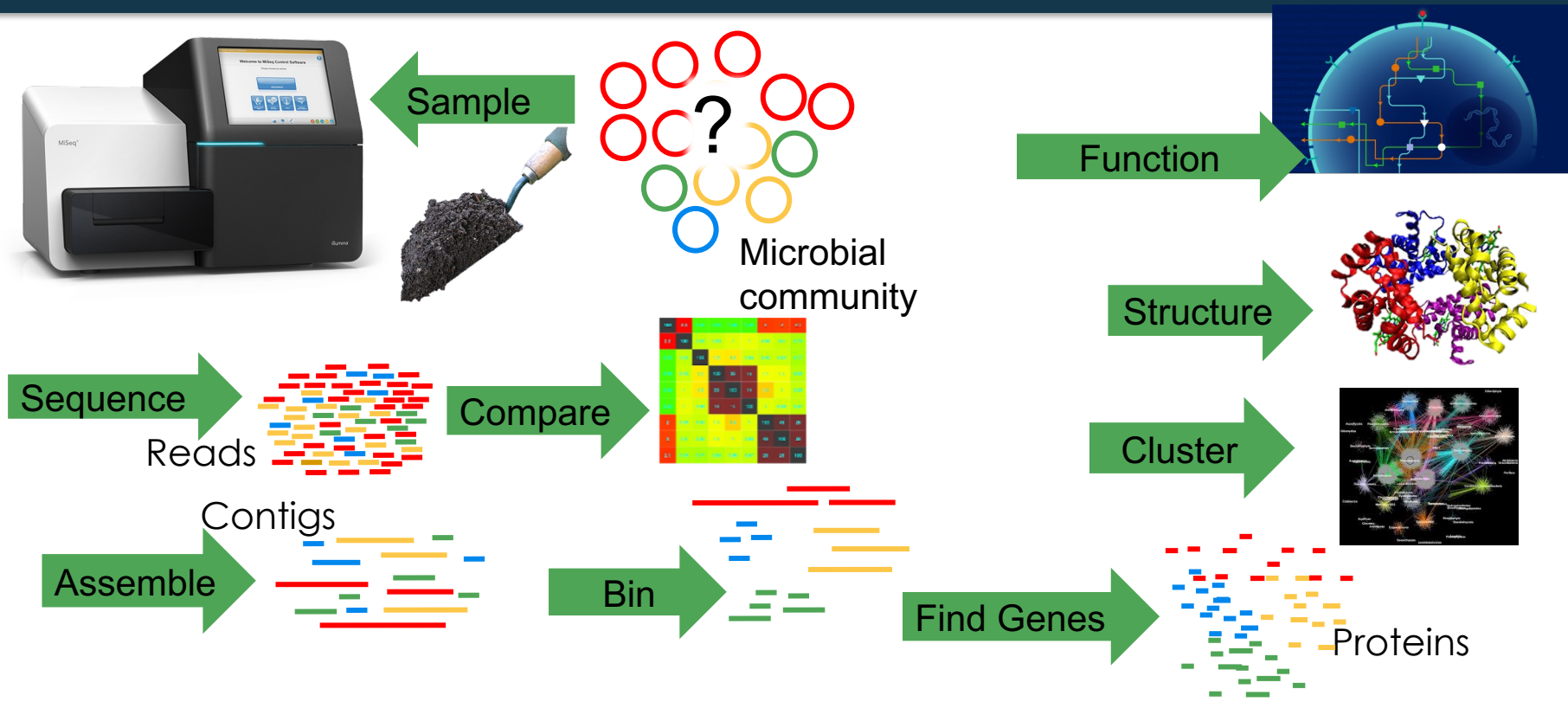
Big  
Machines



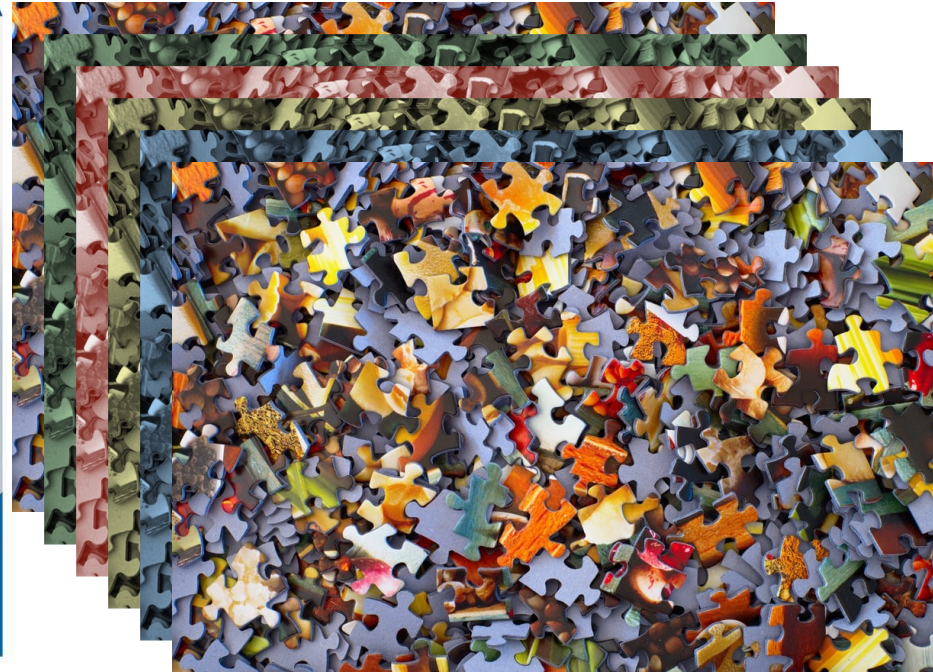
Scalable  
Algorithms



# ExaBiome: Exascale Solutions for the Microbiome



# MetaHipMer: De Novo **Metagenome** Assembler



# What happens to microbes after a wildfire? (1.5 Terabytes - completed)



# How do carbon and metabolism in freshwater lakes change across 17 years? (26TB)



# Tara Oceans



Showing the invisible life of the ocean

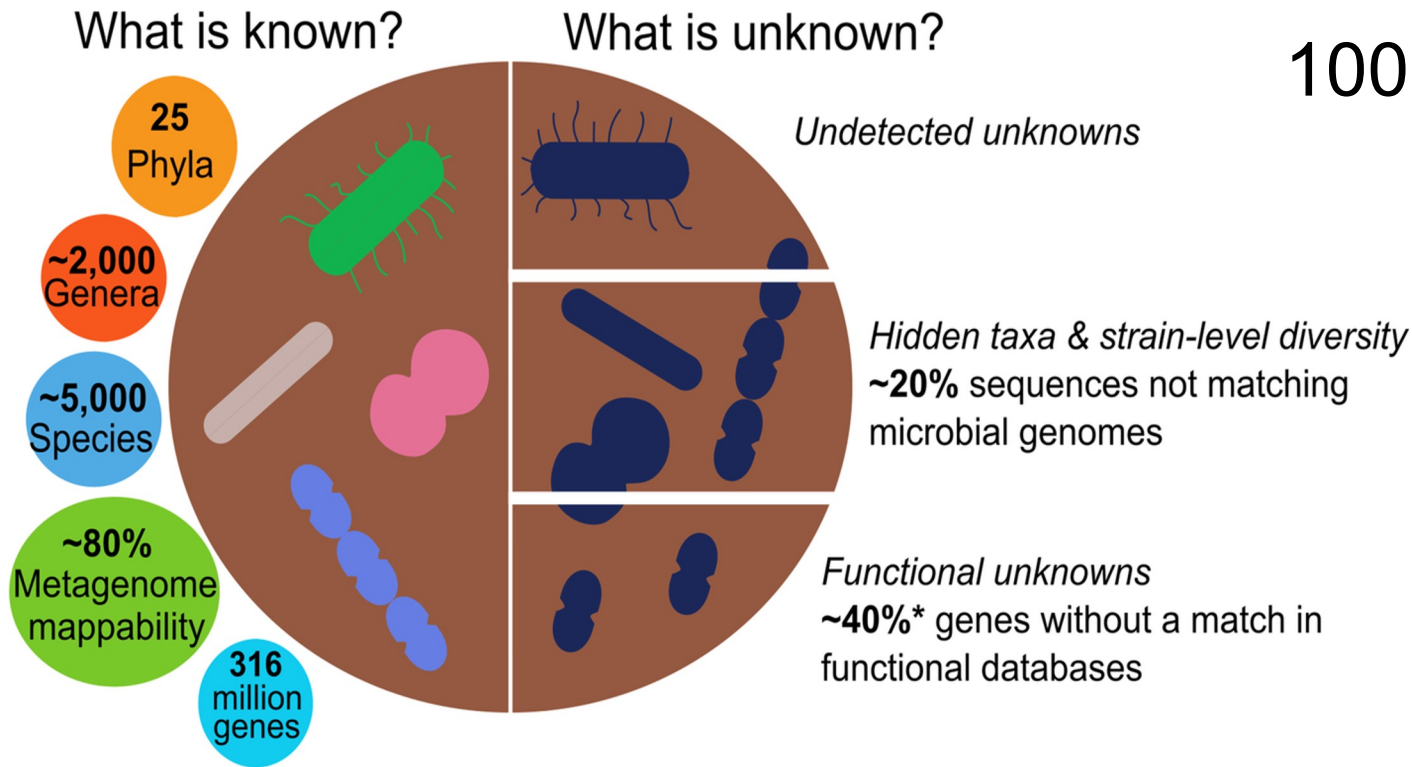
- 2009–2013 expeditions
- 35000 samples from all oceans

100

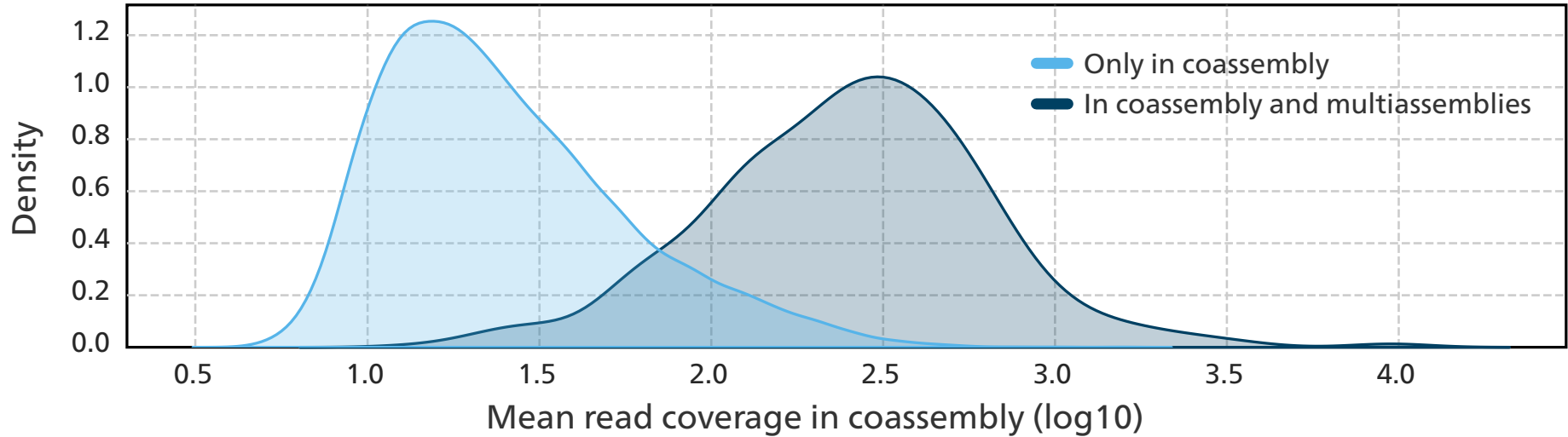
- ~~84~~ TB (71 TB unique) of data!
- Assembled in 94 mins
- Using 36,000 GPUs on Frontier
- Using a Berkeley-designed language

# The Human Microbiome

100 TB of data!



# Co-Assembly vs. Multiassembly: better science

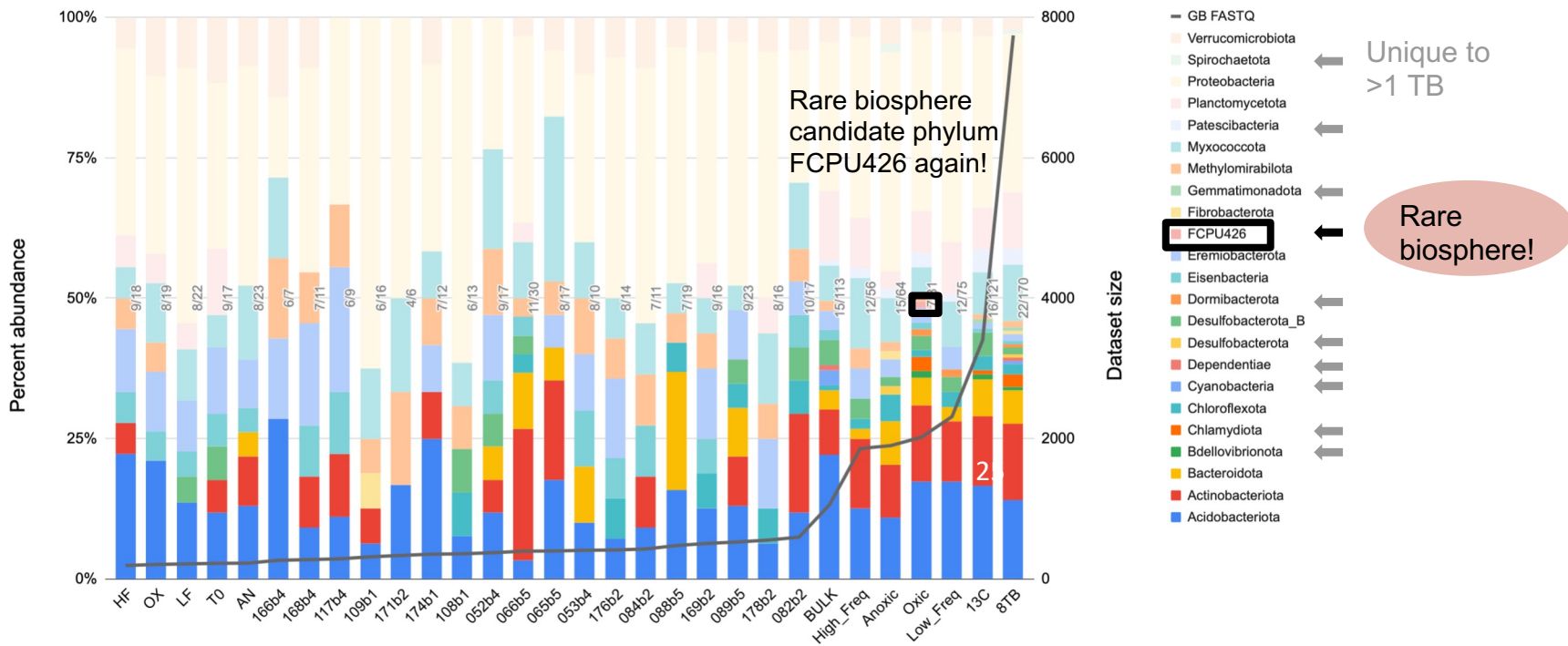


- **Multiassembly (usual approach): Assemble each sample on a shared memory machine**
- **CoAssembly (MetaHipMer): Assemble everything at once on a supercomputer.**

# More taxonomic diversity

## GRE taxonomic abundance (phylum level)

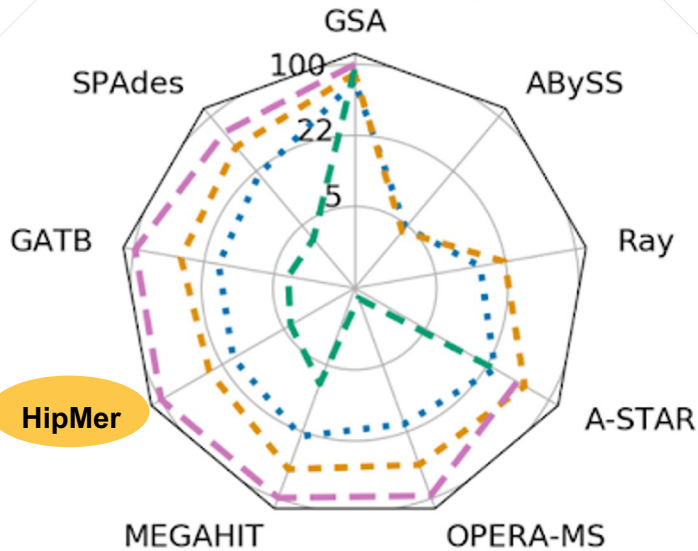
bar labels = n phyla / n MAGs



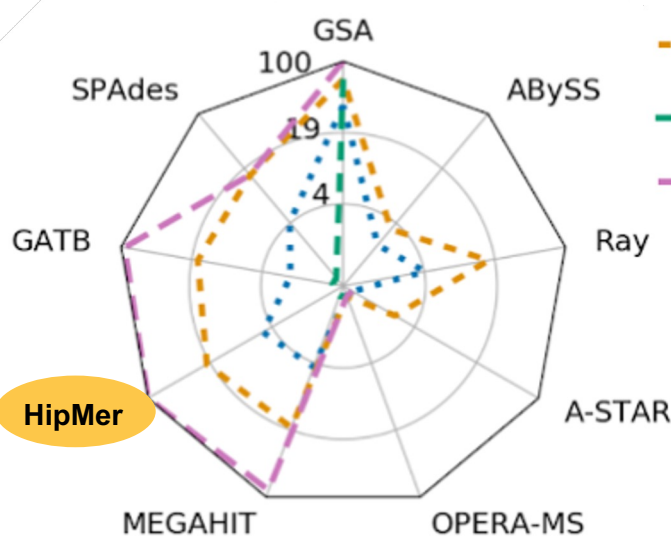


# Ensuring High Quality Assemblies

Genome fraction (%)



Strain recall



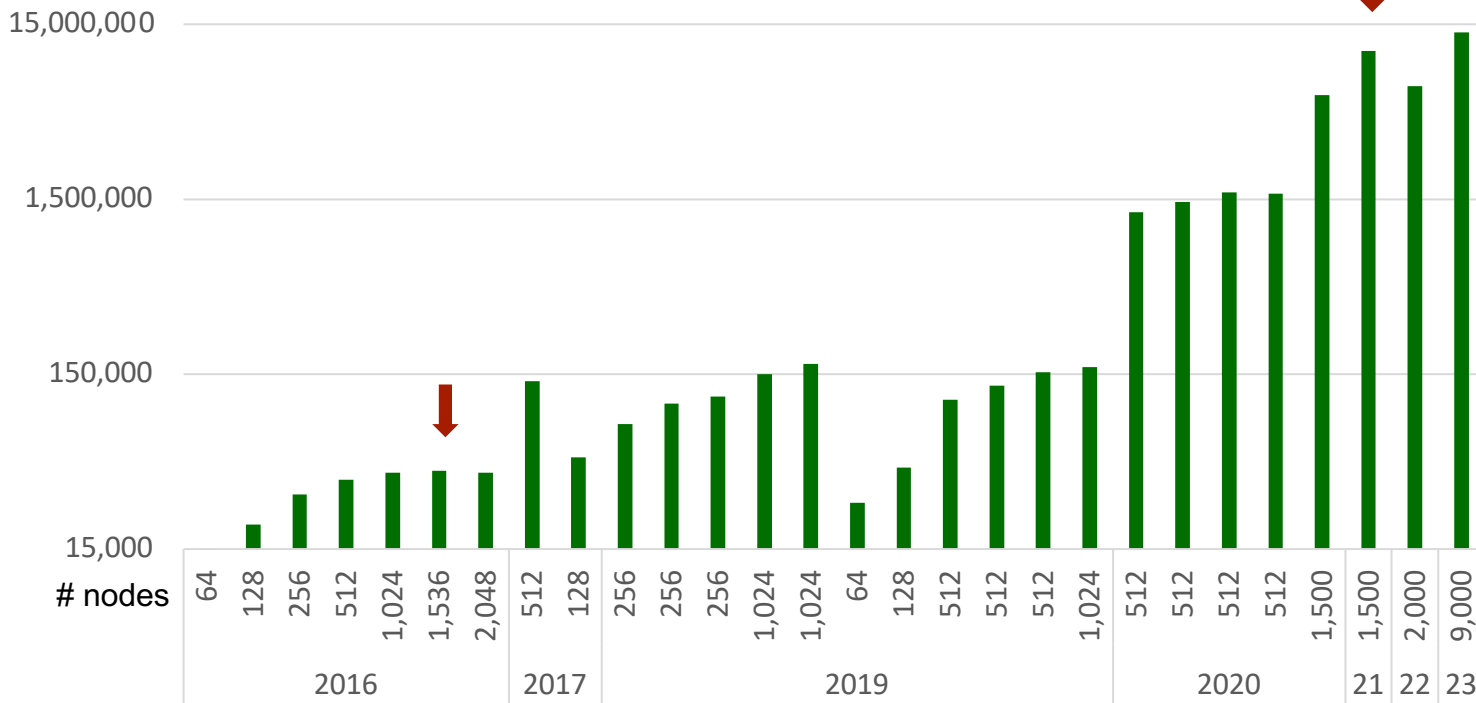
- Common marine
- Unique marine
- Common strain madness
- Unique strain madness

“... we analyze 5,002 results by 76 program versions...

The **best ranking method across metrics and all datasets was [Meta]HipMer....”**

# Assembly Rate on Science Data

Bytes Assembled / Sec



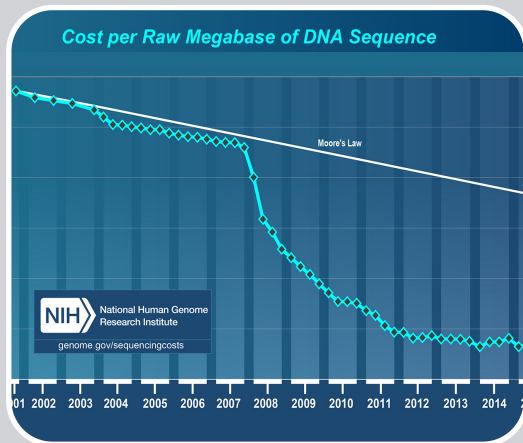
Largest data sets require largest machines

Not just data and machine size

Over 250x on ~equal node counts!

- better algorithms
- less software
- use of GPUs

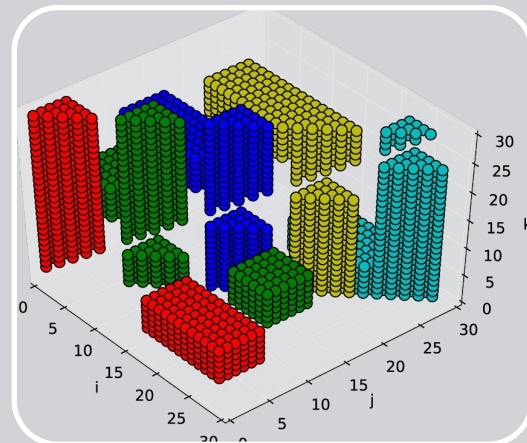
# Genomic Analysis at Scale



Big Data

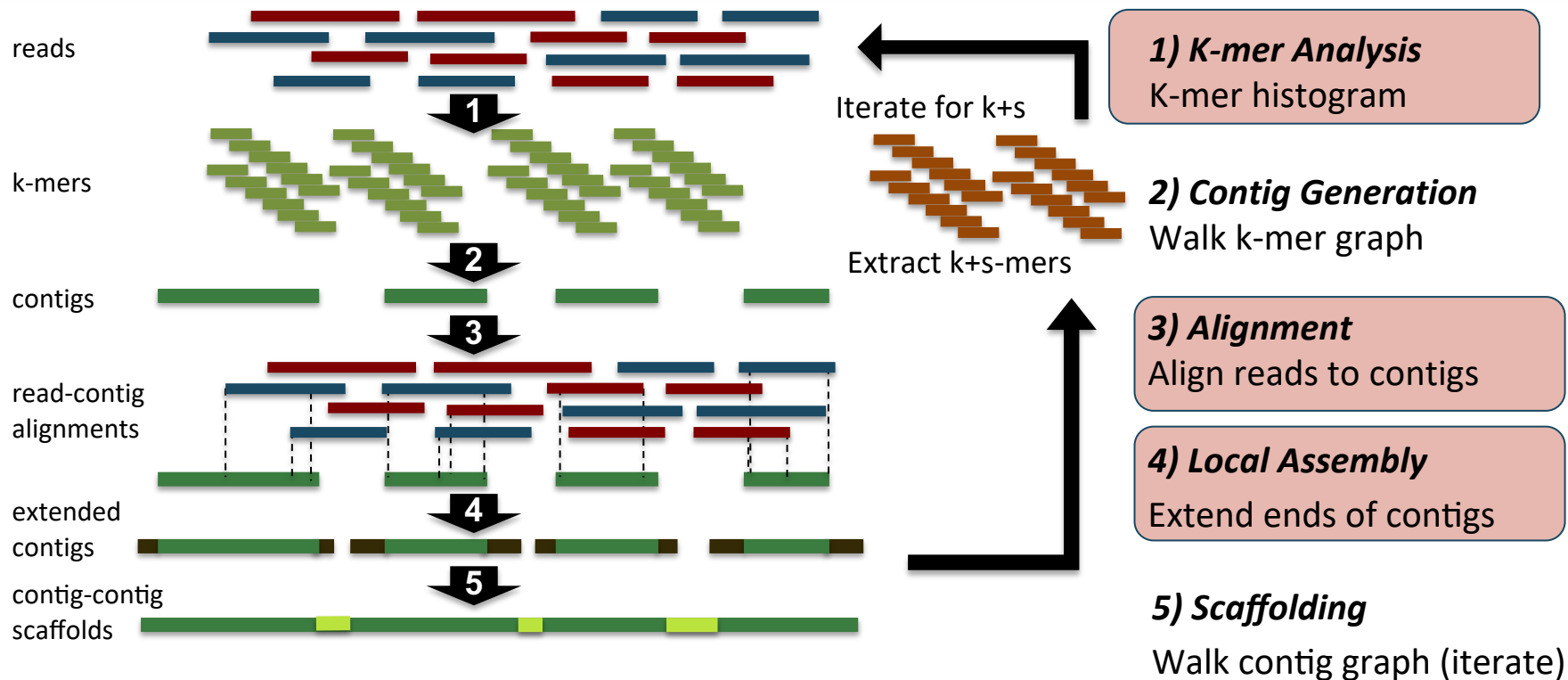


Big  
Machines



✓ Scalable  
Algorithms

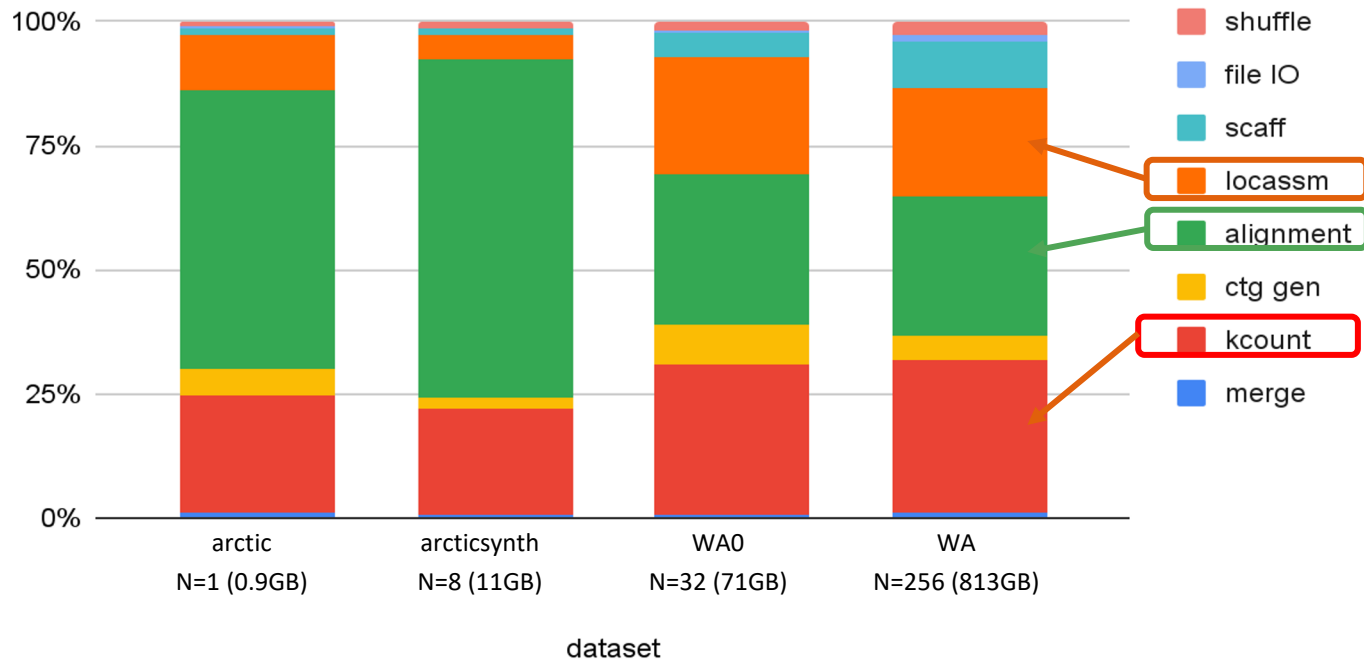
# MetaHipMer Assembly Pipeline (UPC++)



Actual pipeline is more complex, simplified for purpose of presentation

# MetaHipMer Time Breakdown

## Stage Timing, CPU



Weak-ish scaling

\* CPU time for alignment slower than "normal" due to SIMD Power9 issues

# Simulation Vs. Data Motifs

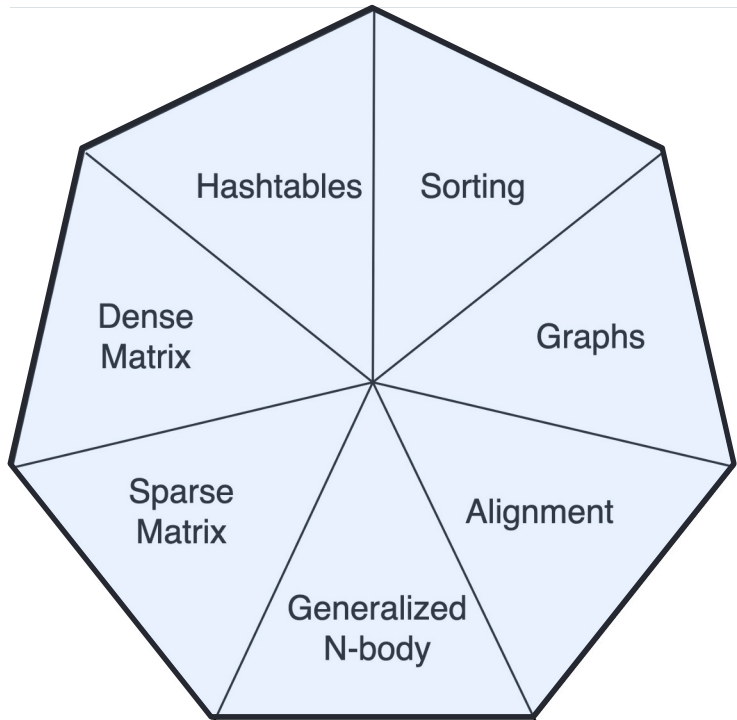
7 Dwarfs of Simulation	7 Giants of Big Data
Particle methods	Generalized N-Body
Unstructured meshes	Graph-theory
Dense Linear Algebra	Linear algebra
Sparse Linear Algebra	
Spectral methods	Hashing
Structured Meshes	Sorting
Monte Carlo methods	Alignment
	Basic Statistics

Phil Colella

NRC Report + our paper

# Motifs of Genomic Data Analysis

## Computational patterns that dominate ExaBiome



### Examples

- Hash all k-mers (k-length strings)
- Count k-mer frequency

aact

ctgt

gtca

- Identify connected components

aact

ctgt

gtca

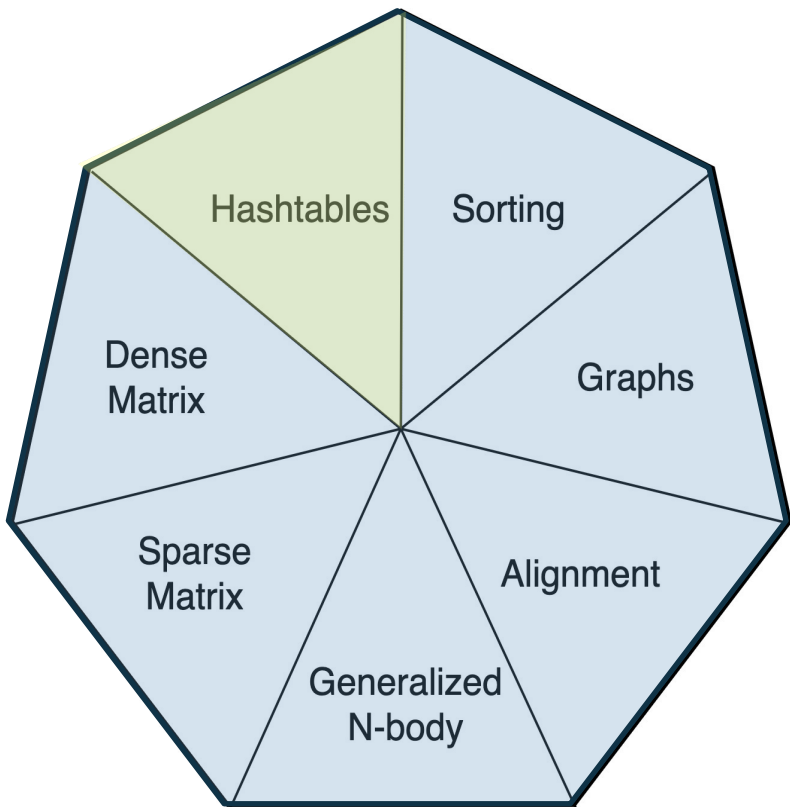
aactca

- Find all pairwise alignments
- Average neighbors on graph

# A Tale of Two Programming Models

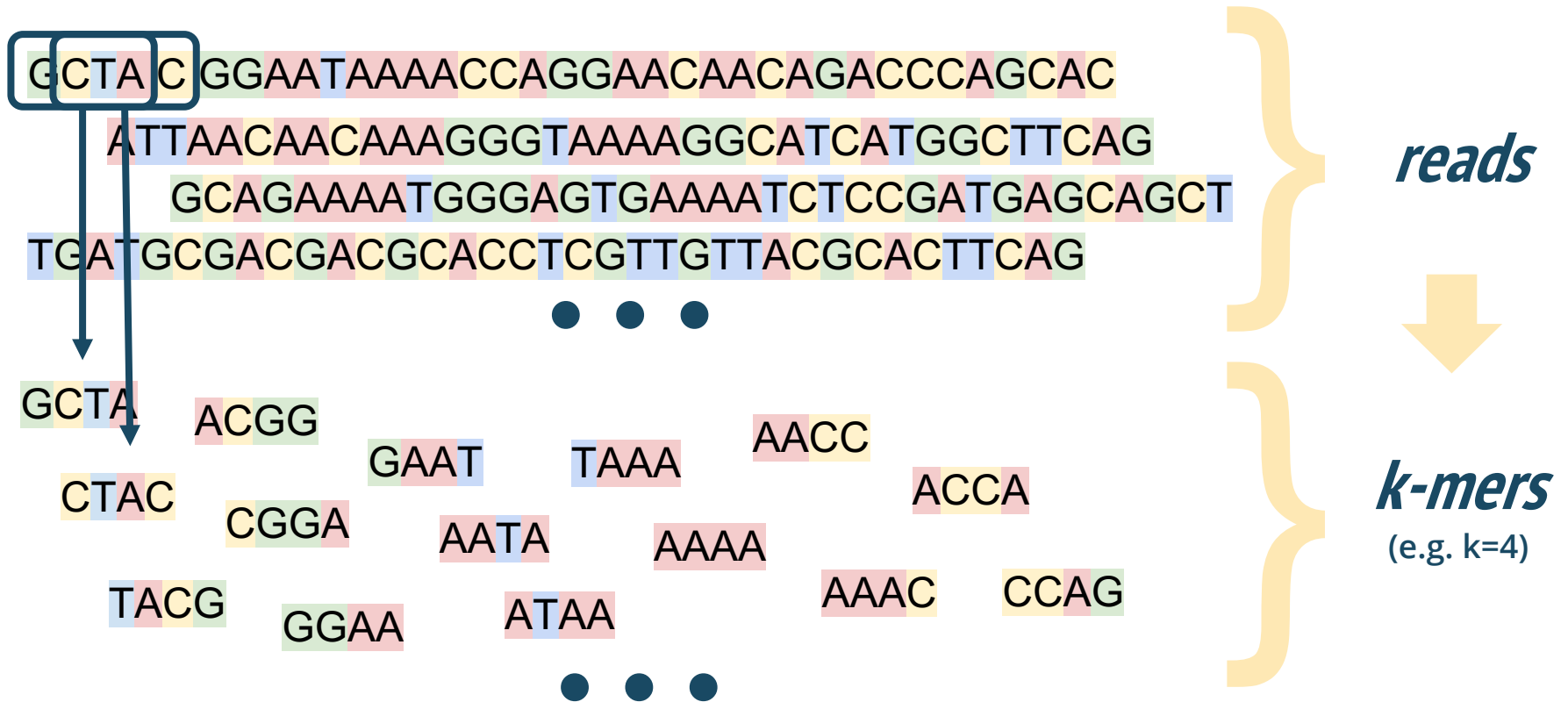
Partitioned Global Address Space (UPC++)	GraphBLAS and MPI
Asynchronous, fine-grained	Bulk Synchronous
Distributed data	Sparse matrix with semiring operations
Logically shared, physically distributed	Local view, distributed
MetaHipMer, KmerProf	PASTIS, HipMCL, diBELLA, ELBA





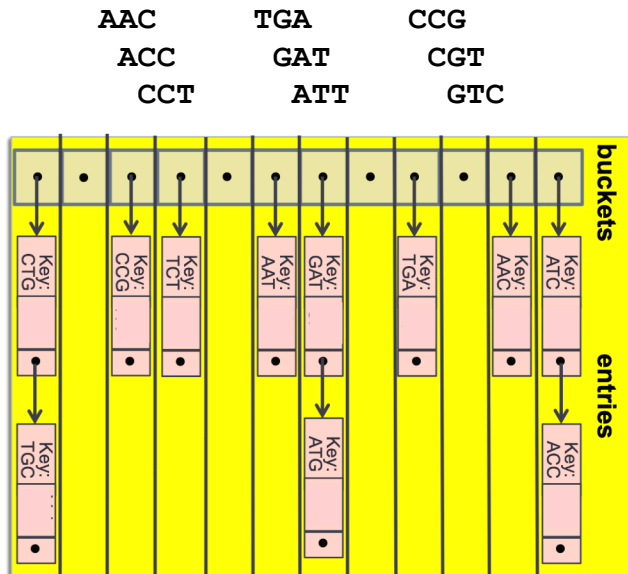
# Hashing

# Counting K-mers to Remove Errors



# Distributed Hash Tables of K-Mers

*Make hash table of k-mers*



*1-sided communication to insert / lookup*

**Keys** are fixed-length strings:

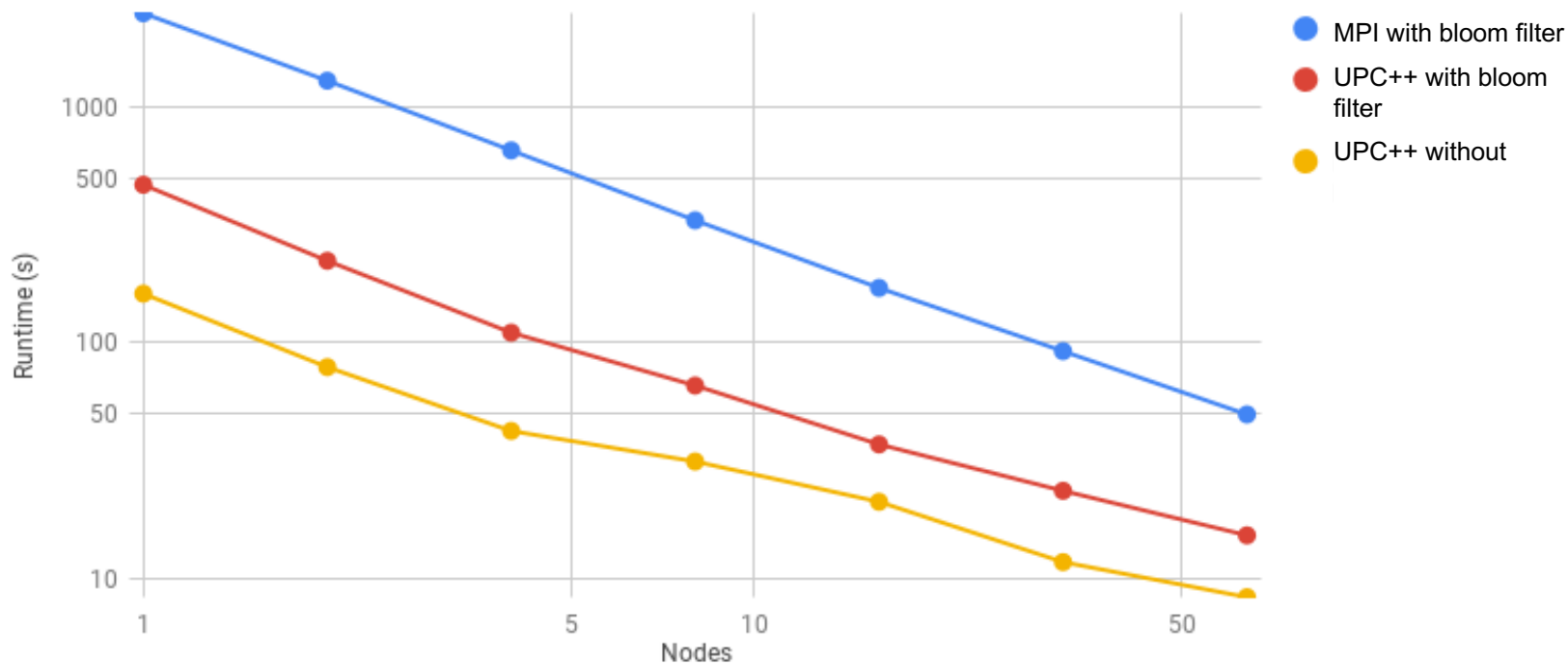
**Values** depend on application:

- A count to remove singletons

Close to k-times memory blowup

- Use Bloom filter to reduce space
- Asynchronous insert with UPC++

# K-mer counting: All the Wires All the Time



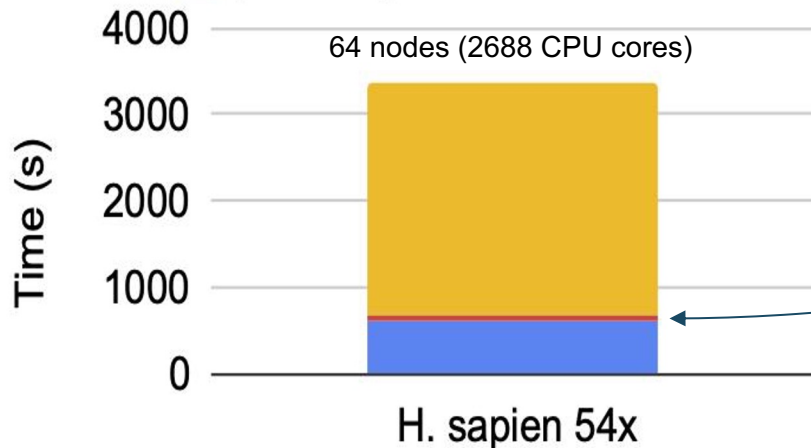
Bulk-synchronous MPI vs Asynchronous 1-sided UPC++ (w/ and w/out Bloom Filter)

Steve Hofmeyr, Rob Egan, Evangelos Gerganas, leads on MetaHipMer software

# K-mer Counting: Finding Data Parallelism

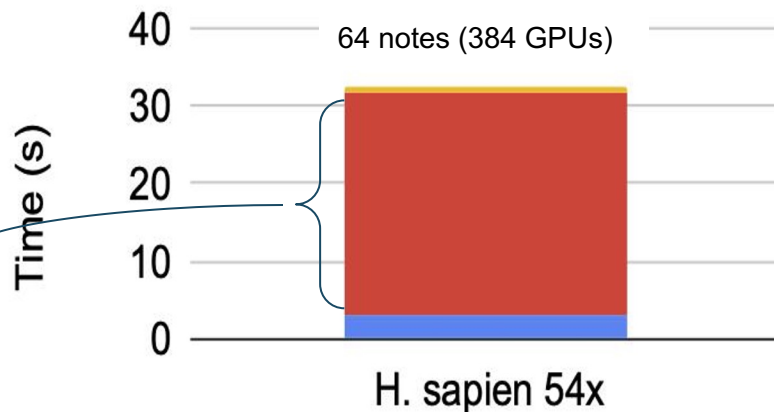
■ kmer counter ■ exchange (incl. MPI call)

■ parse & process kmers



■ kmer counter ■ exchange (incl. MPI call)

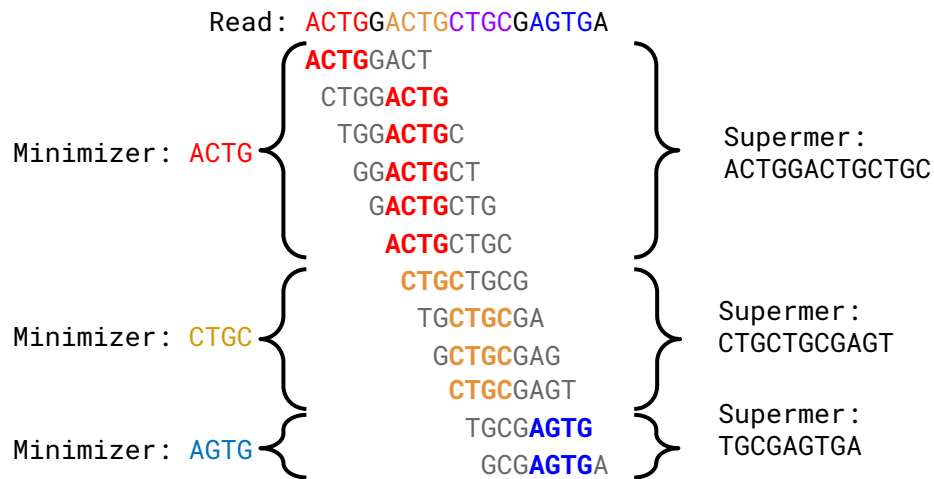
■ parse & process kmers



- K-mer counter on Summit. (Note scales -- red k-mer exchange time is roughly equal.)
- Reduce CPU/GPU communication by parsing as well as processing on GPU

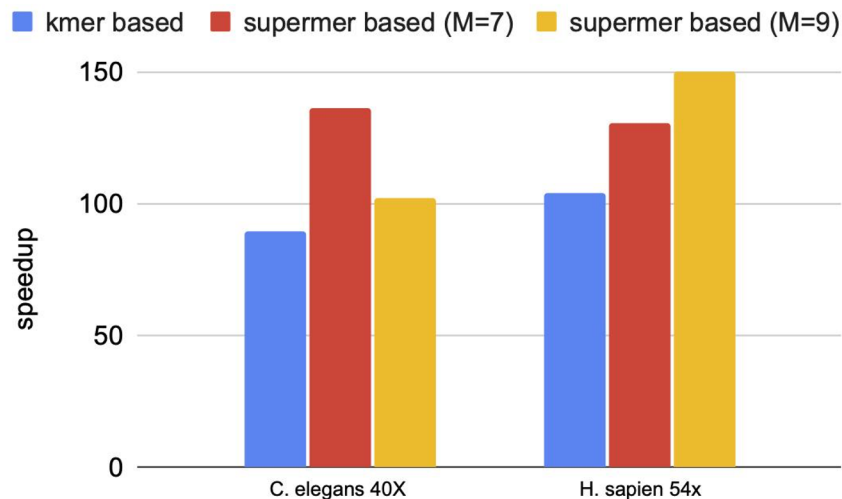
Over 100x speedup!!

# K-mer Counting: Reducing Communication



## Reduce communication with “Supermers”

- Multiple contiguous k-mers ( $k=7$ , minimizer=4)
- map to the same process ID with minimizer-based hashing
- Saves volume (bandwidth) and number of messages (latency)



## Speedup on 64 Summit nodes

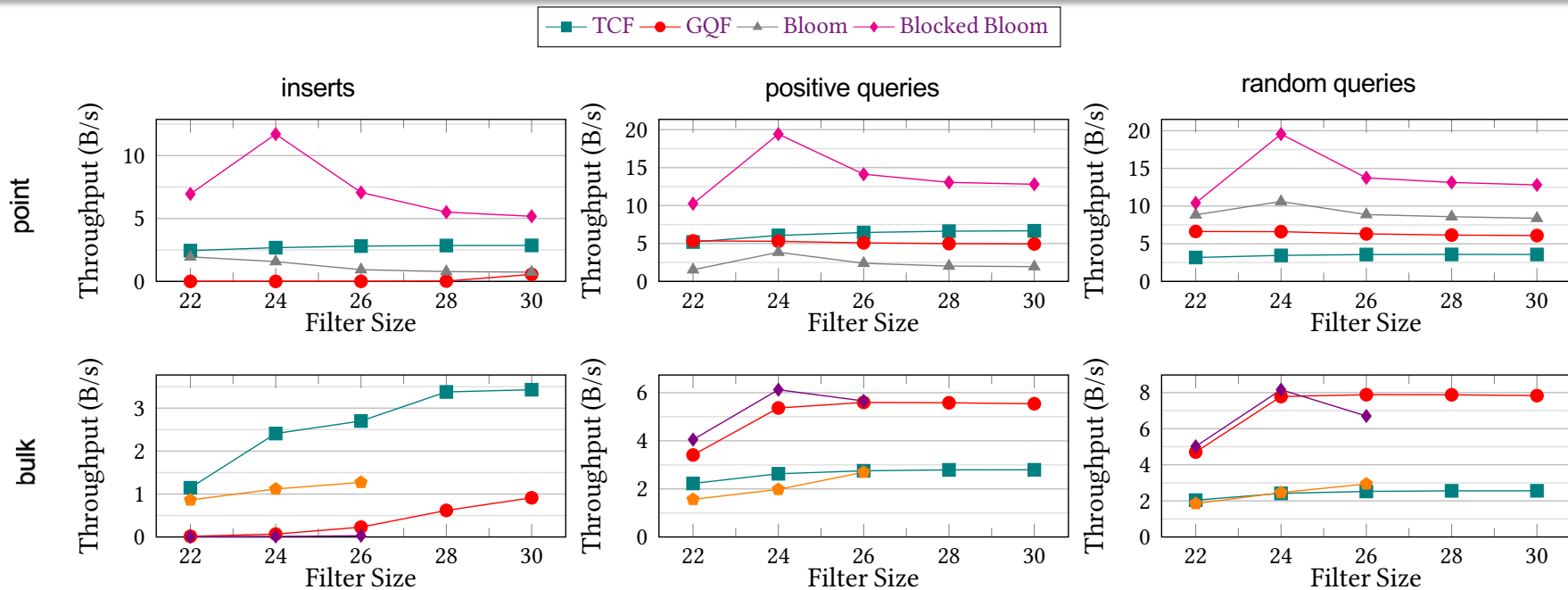
- 6 GPUs / node
- baseline: 42 cores / node

# Universal Filter for GPUs

Filter	Insert		Query		Delete		Count	
	Point	Bulk	Point	Bulk	Point	Bulk	Point	Bulk
GQF	✓	✓	✓	✓	✓	✓	✓	✓
TCF	✓	✓	✓	✓	✓	✓		
BF	✓	✓	✓	✓				
SQF		✓		✓		✓		
RSQF		✓		✓				

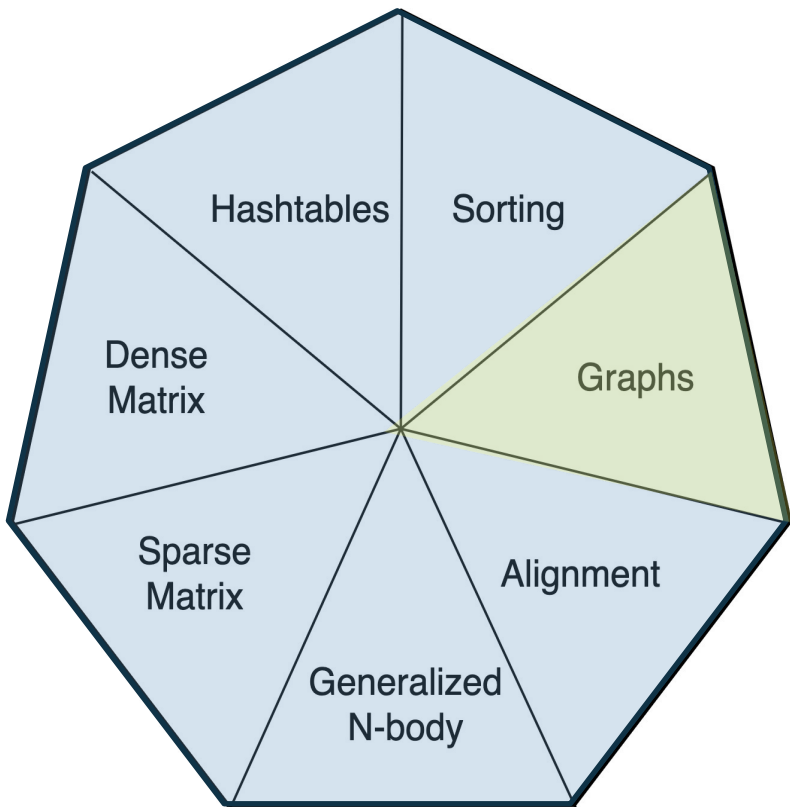
- **Our GPU Quotient Filter (GPU) is fully featured:**
- **Deletions and Counts**
- **Individual element operations (Point) and aggregate (Bulk)**

# GQF Performance on Multi-GPUs



- Saves ½ of peak memory usage in MetaHipMer relative to no filter
- Saves ½ of communication in K-mer counting phase relative to bloom filter

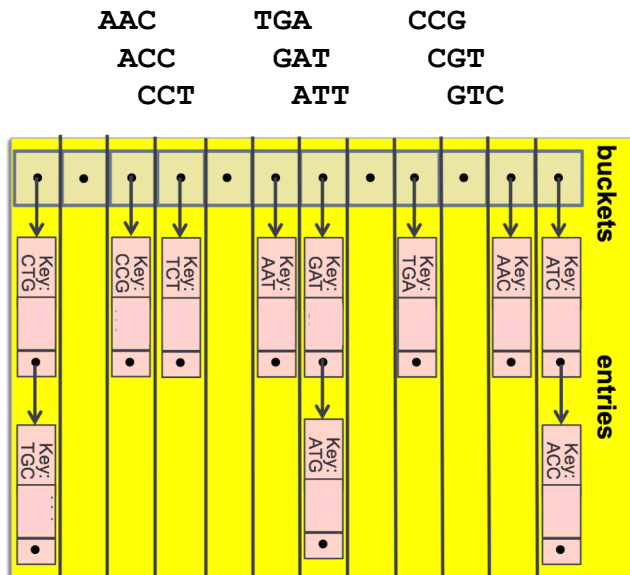




# Graphs

# K-Mer Hash Tables Viewed as a Graph

*Make hash table of k-mers*



*1-sided communication to insert / lookup*

**Keys** are fixed-length strings

**Values**

- Remove branches
- Find connected component “contigs”

Graph walk with poor locality

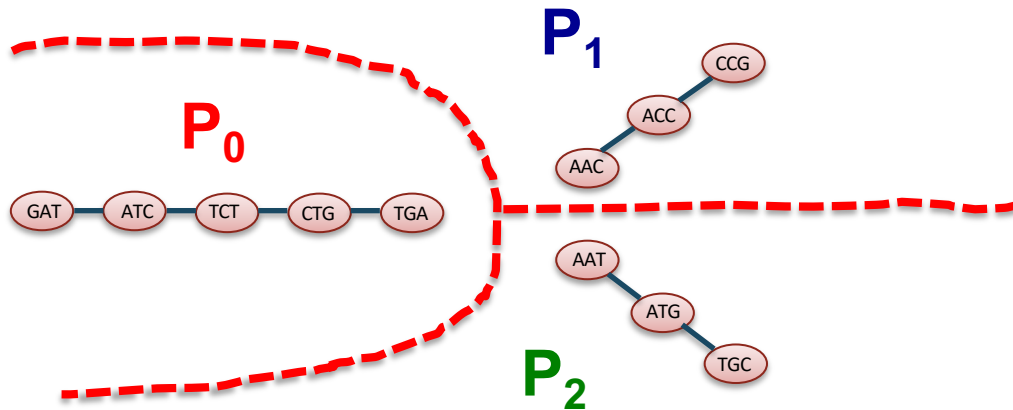
- Asynchronous lookup with UPC++

# Avoiding Communication in Graph Walk

*Next step in this assembler is a DFS on the  $k$ -mer graph (edges are  $k-1$  overlaps)*

**Caching** for temporal locality  
(reuse): if few large items, so  
lookups will repeat

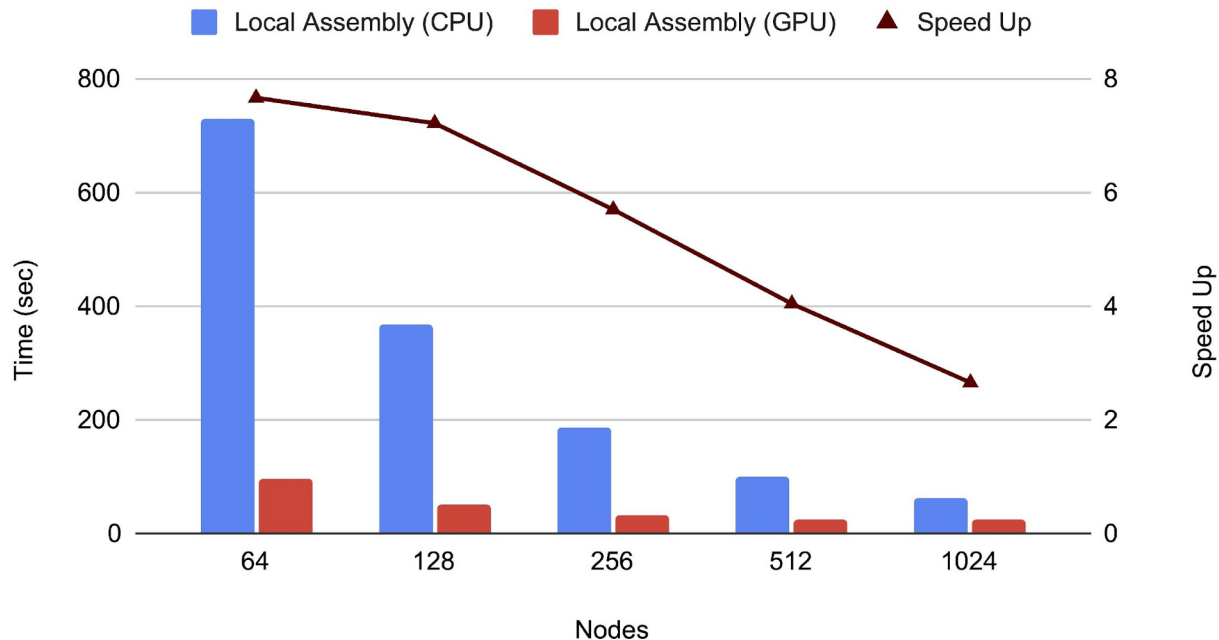
**Layout** for spatial locality: if we  
have an “oracle” that approximate  
final genome



**Traversal is up to 2.8x faster!**  
**Up to 76% reduction of off-  
node communication !**

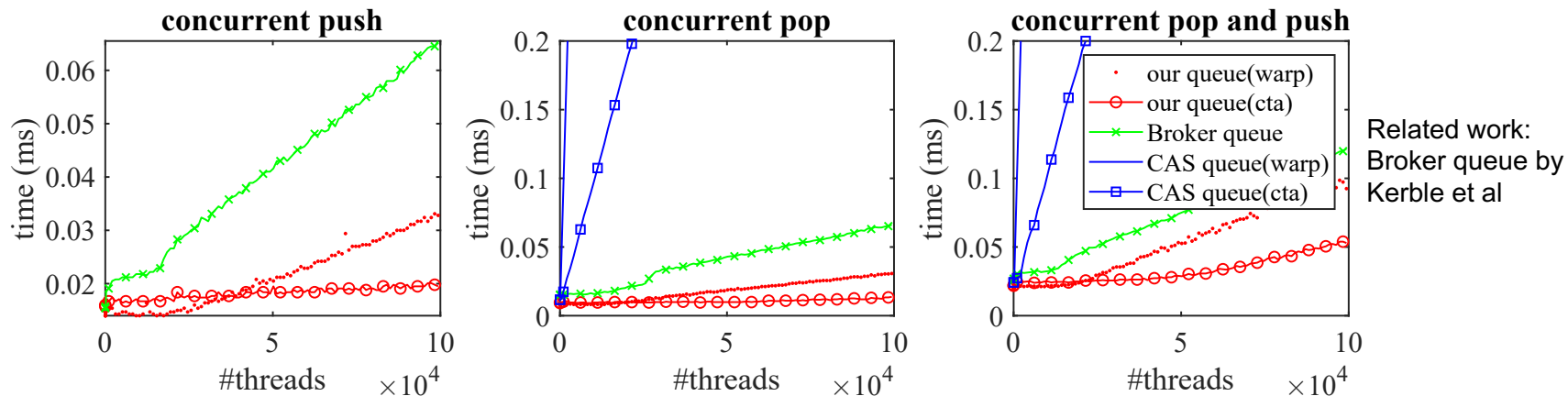
# Local Assembly on Summit

## CPU vs GPU

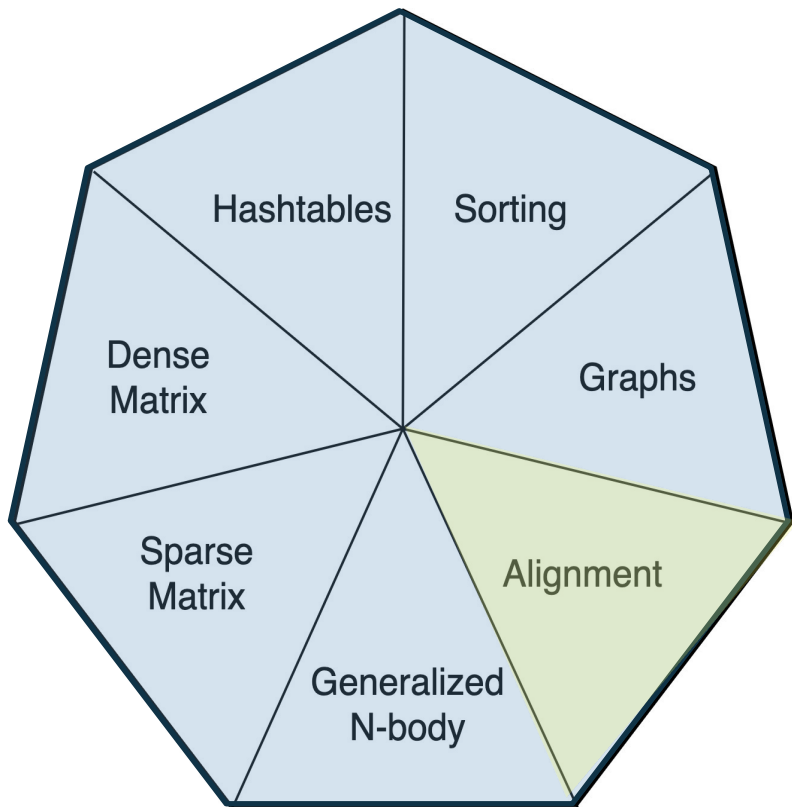


- Speedup of 7x on 64 Summit nodes.
- Lower as expected as machine scales (strong scaling)

# Asynchronous Parallelism on GPUs



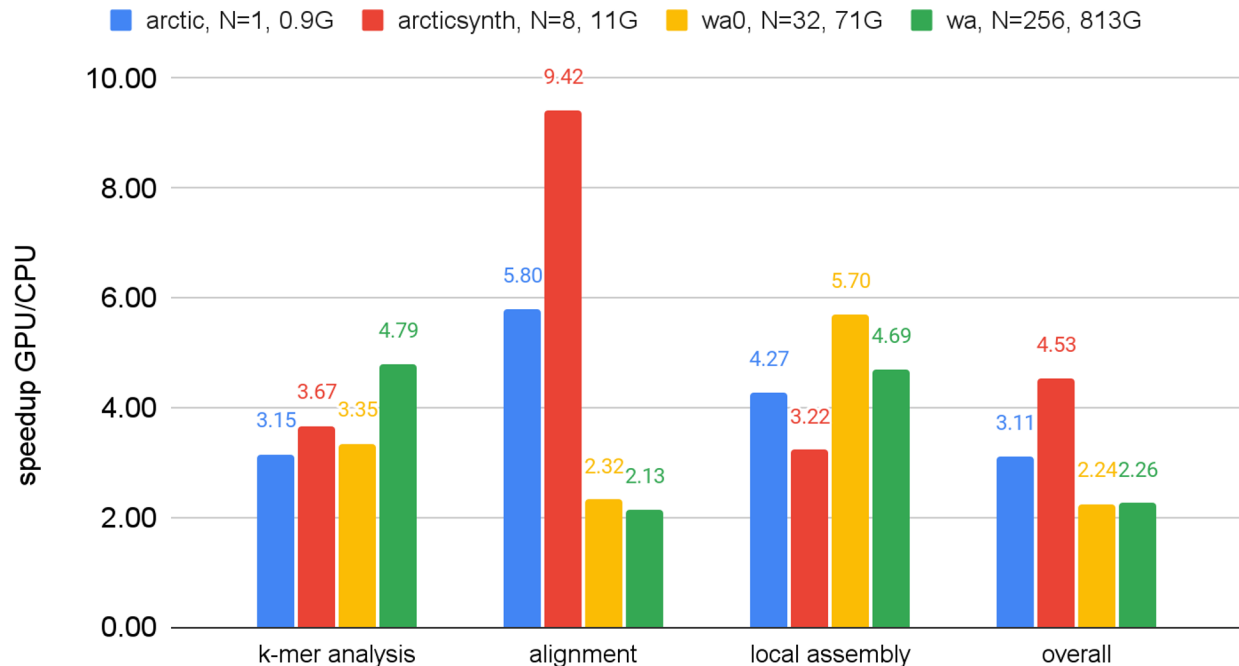
- **Warp-size or Thread-group (CTA)-size workers**
- **Locking vs Compare-And-Swap (CAS) implementations**
- **Avoid level-by-level synchronization**
- **Use persistent threads as an option**



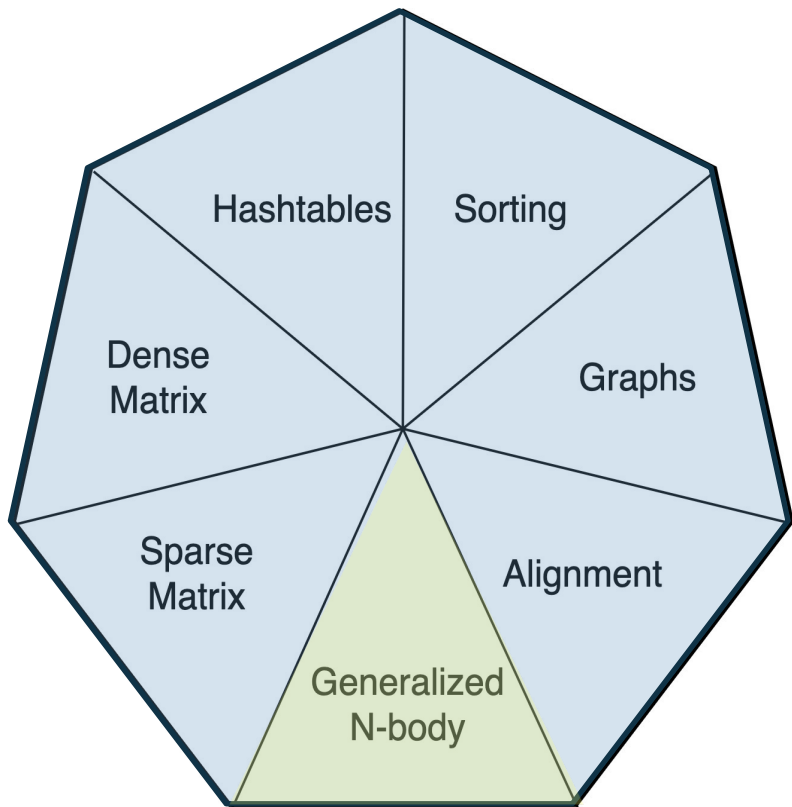
# Alignment

# GPU Optimizations

## Speedups from GPUs



GPU optimizations are complex (hash tables, graphs, etc.)



# Generalized N-Body

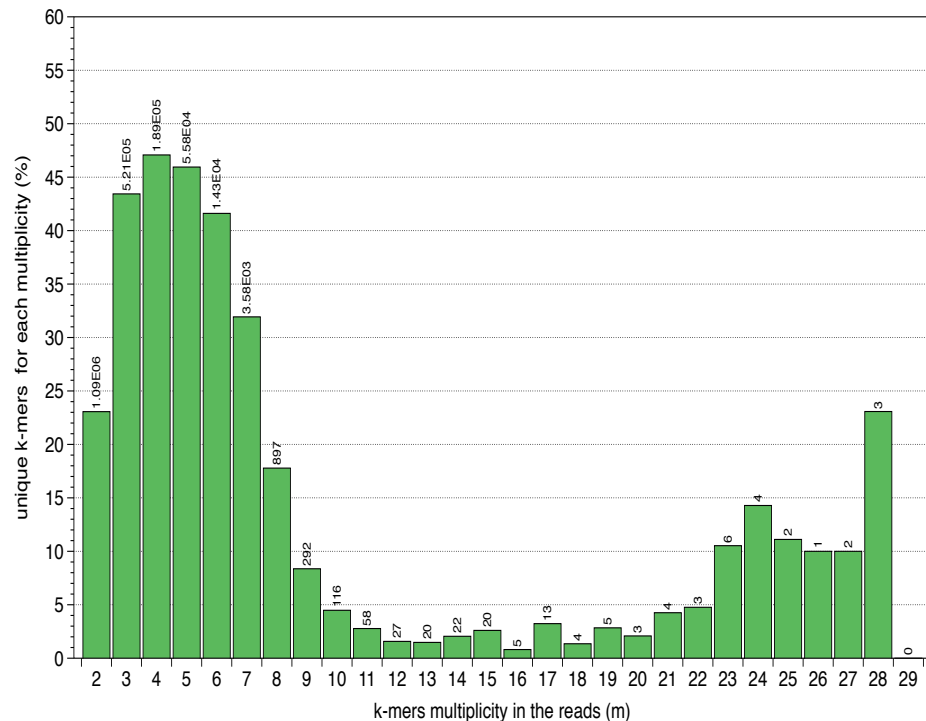


# ELBA: Long Read Assembler; Different Approach

Long reads (PacBio, etc.)

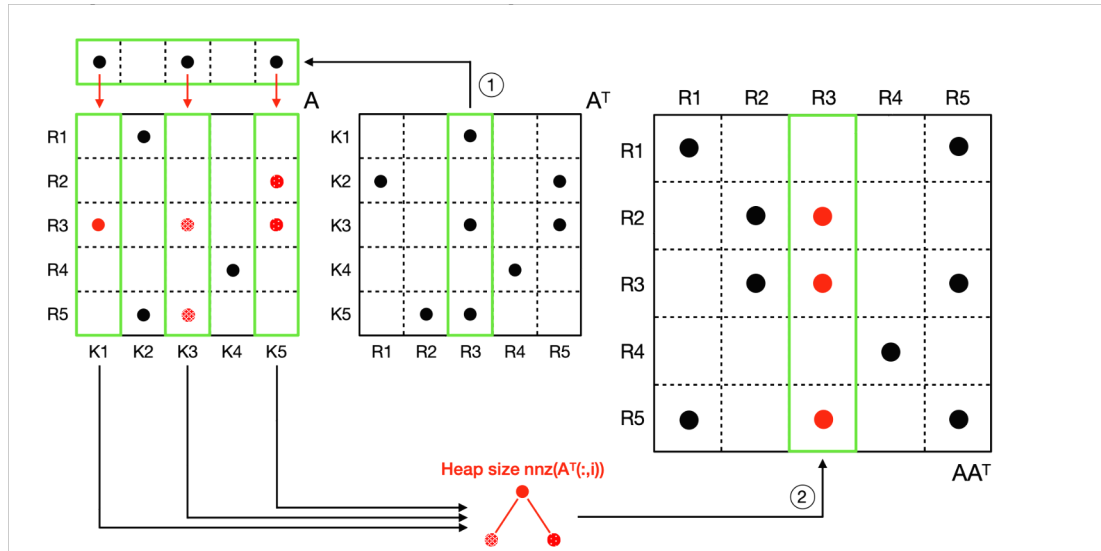
- Longer alignments
- More compute-intensive
- More GPU friendly

Only align pairs of reads that have a common k-mer

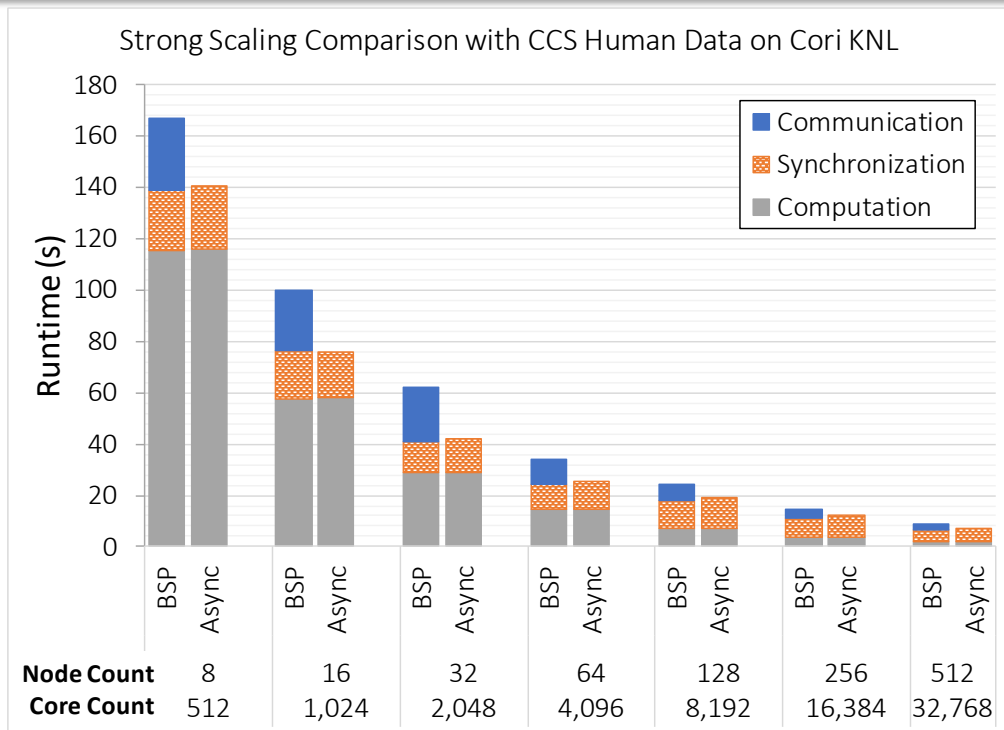


# Set Alignment is a Sparse All-to-All

Run expensive alignment on all pairs with a common k-mer



# Bulk-Synchronous vs 1-sided Asynchronous



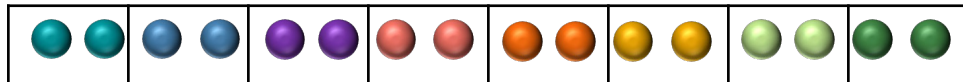
## Asynchronous communication

- Hides latency and uses less memory in general
- Uses “All the wires, all the time”
- But uses linear communication, not log-p complexity collectives

# Avoid Communication, Maximize Parallelism

Compute on all pairs of particles or strings, or...

## Obvious solution

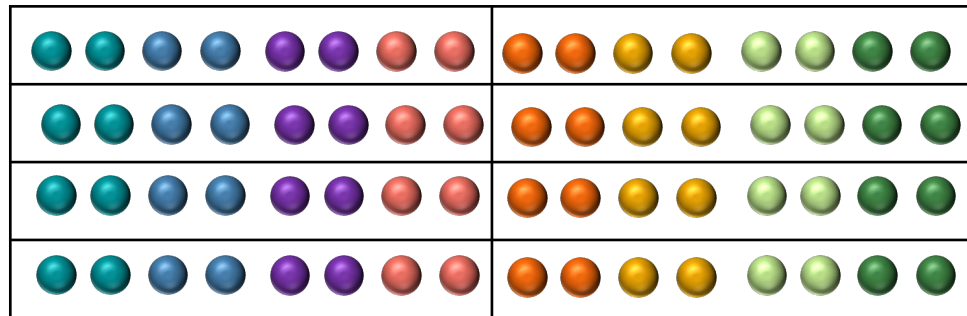


16 particles on 8 processors  
Pass all particles around ( $p$  steps)

Decreases

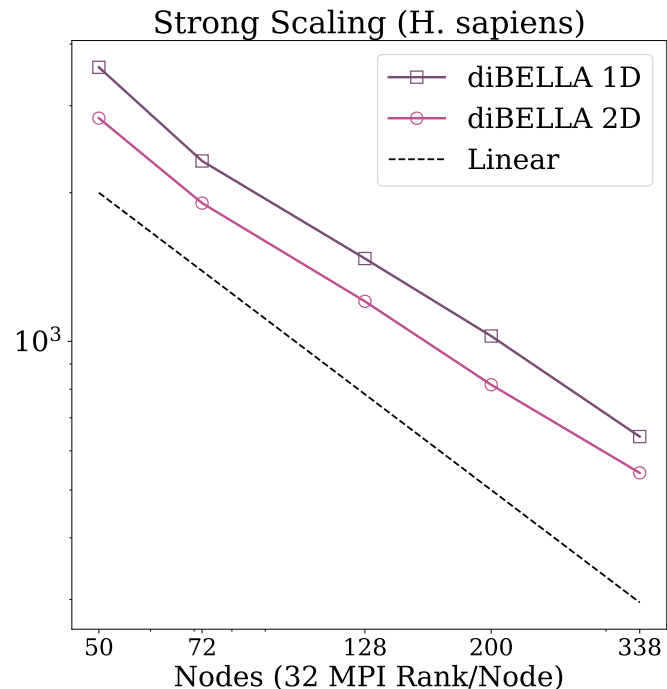
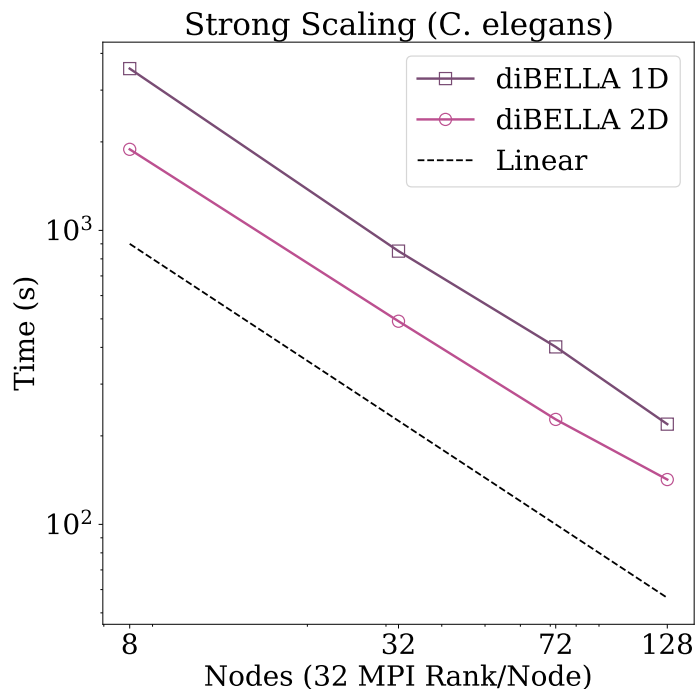
- #messages by factor  $c^2$
- #volume sent by factor  $c$

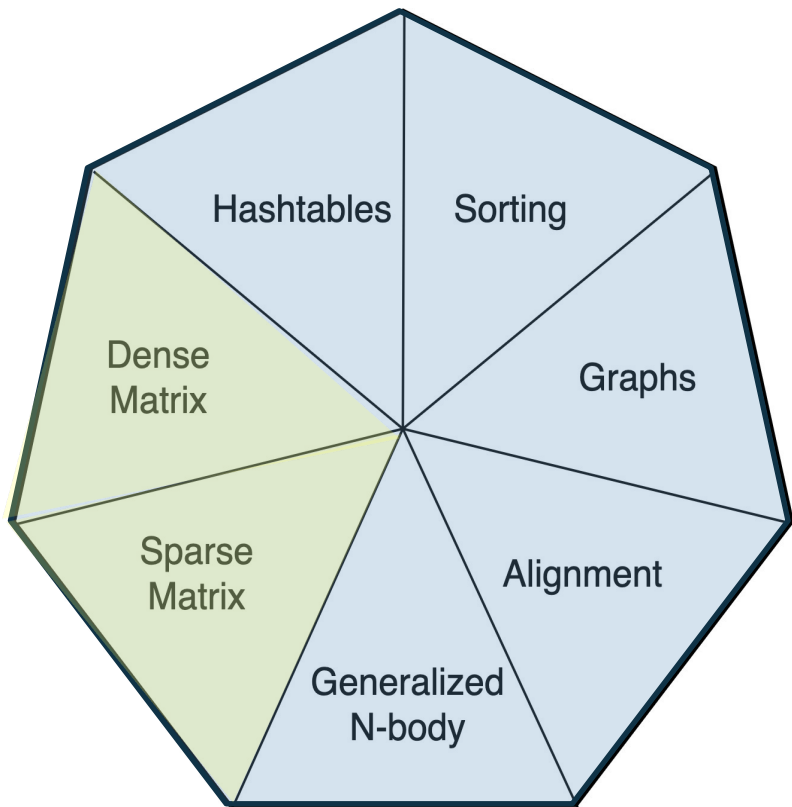
## Better solution



$c = 4$  copies of particles  
8 particles on each

# 1D vs 2D Algorithm on DNA “overlap”





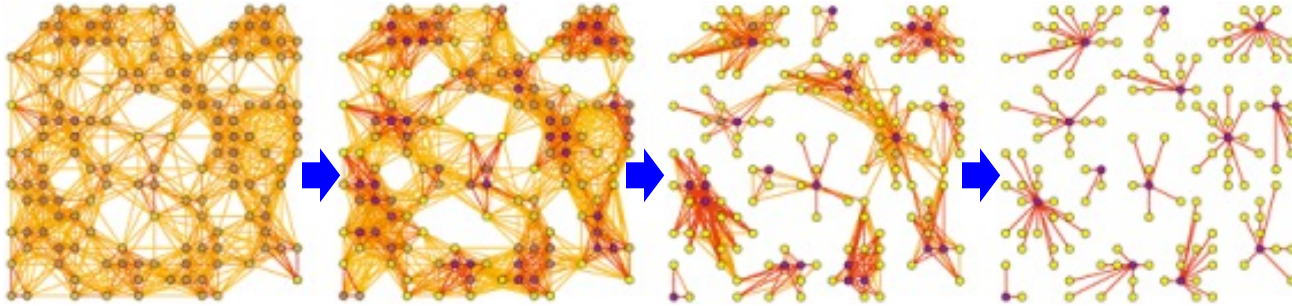
# Sparse and Dense Matrices

(machine learning)

# Protein Clustering with Sparse Matrices

Input: Adjacency matrix  $A$  (sparse)

Image source: <http://micans.org/mcl/>

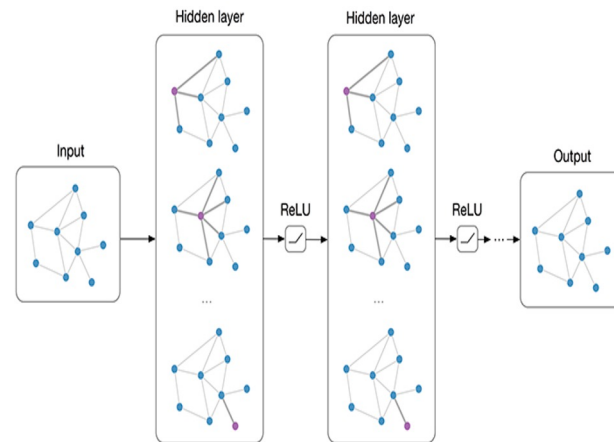
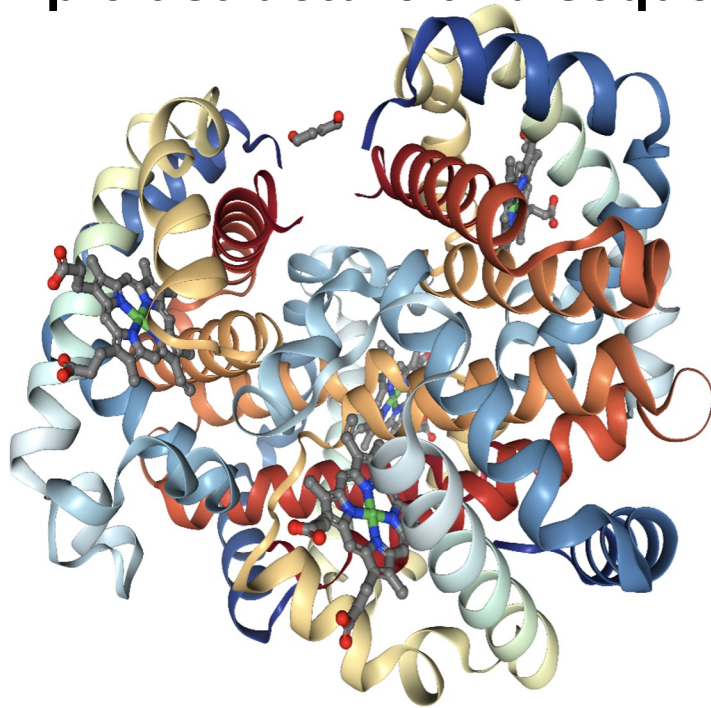


- **Similarity Matrix:** “Many-to-many” protein alignment
- **Expansion:** Square matrix, pruning small entries, dense columns
- **Inflation:** element-wise powers

**PASTIS + HipMCL**

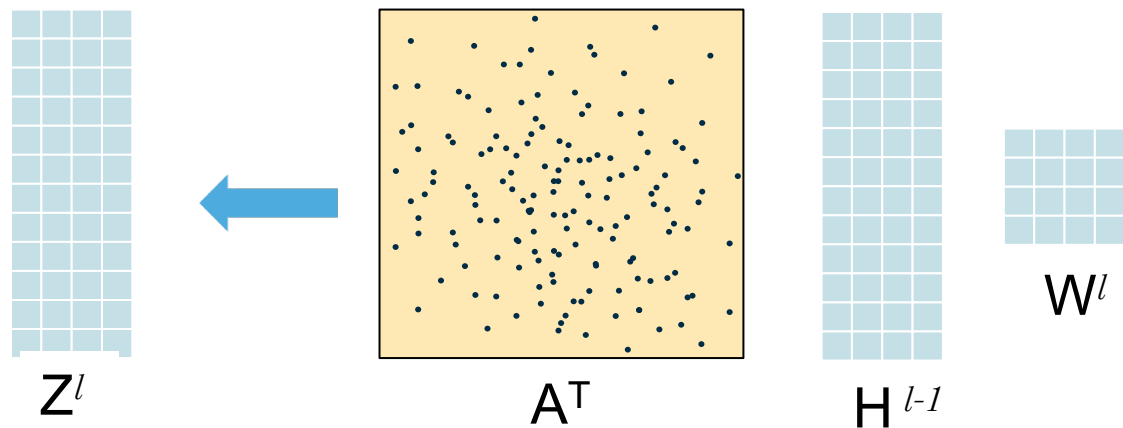
# Learning protein function using GNNs

Exploit structure and sequence to understand the function of protein-coding genes



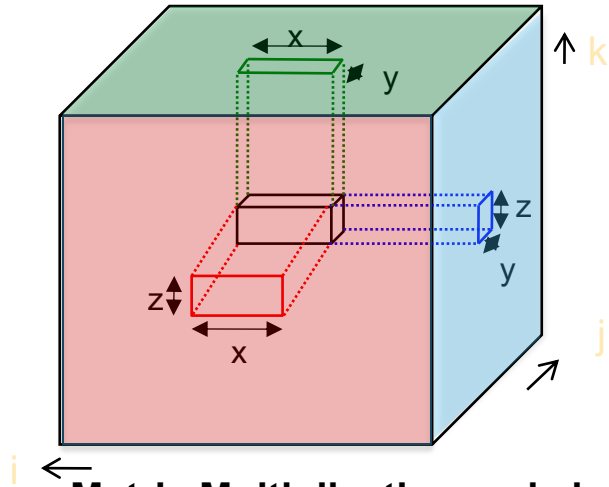


# Bottleneck in GNN Training



- $A^T H^{l-1}$  sparse-dense matmul (SpMM)
- $(A^T H^{l-1}) W^l$  dense-dense matmul (DGEMM)
- **SpMM is the bottleneck, not DGEMM!**

# Communication-Avoiding Matrix Multiply

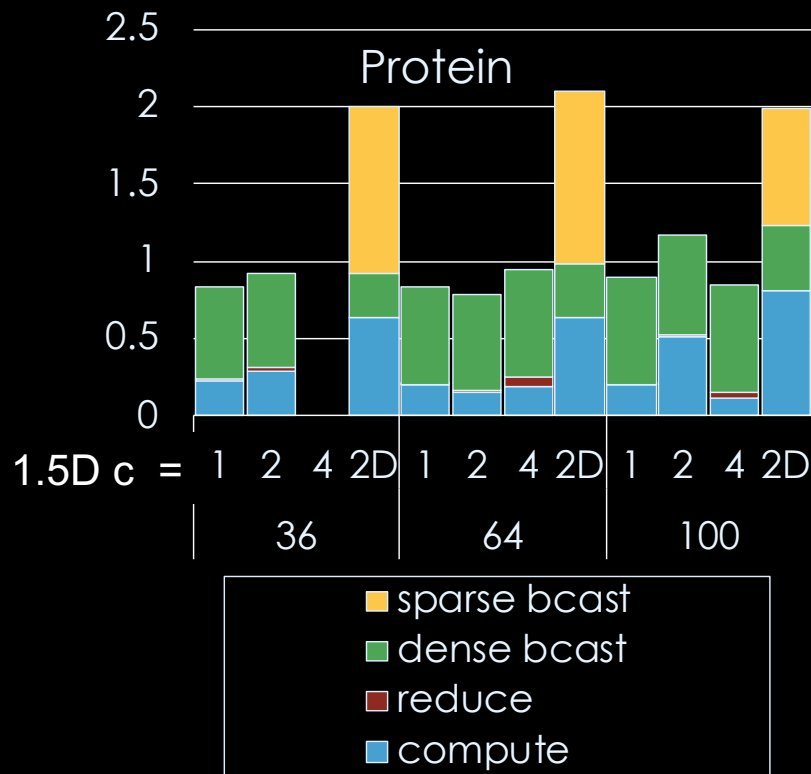
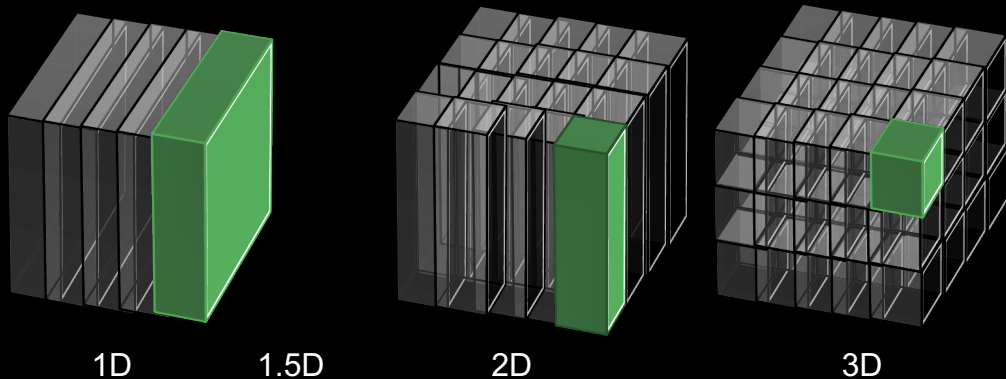


- 2D algorithm: never chop  $k$  dim
- 3D: Assume  $+$  is associative; chop  $k$ , which is  $\rightarrow$  replication of  $C$  matrix

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] ...
```
















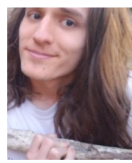


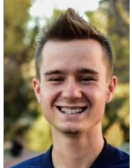


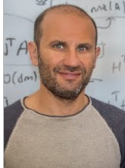



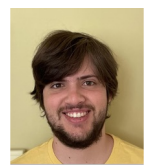












# Avoiding Communication in GNNs



# Take-Aways Messages

- **Applications**
  - More data, more compute → more insights
  - ~7 motifs of data analytics
- **Programming models**
  - Use of PGAS for irregular, fine-grained problems
  - Can still map GPUs
- **Algorithms**
  - Use memory to reduce data (volume)
  - Use all the wires all the time
- **Hardware**
  - Integrate communication on accelerators

# The ExaBiome Team

								
Muaaz Awan	Ariful Azad	Nick Battacharya	Aydin Buluc	Patrick Chain	Brandon Cook	Alex Copeland	Alicia Clum	Rob Egan
								
Marquita Ellis	E. Georganas	E. Goltsman	Giulia Guidi	Steve Hofmeyr	Taufique Hussain	Richard Lettich	Nikos Kyrpides	J. Madson
								
Hunter McCoy	Russell Neches	Israt Nisa	Lenny Olikier	P. Pandey	Robert Riley	Dan Rokhsar	Gabriel Raulet	Oguz Selvitopi
					     			
Migun Shakya	Nick Swenson	Andrew Tritt	Kathy Yelick	Brett Youtsey				



ECCP

The logo features the letters 'E', 'C', 'C', and 'P' in a white, sans-serif font. The first 'C' is partially enclosed by a blue circular arc, and the second 'C' is partially enclosed by a red circular arc. The background is a dark blue space scene with a view of Earth's horizon and a starry field.

EXASCALE  
COMPUTING  
PROJECT