INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING
ICPP/2021/CHICAGO/USA
acm In-Cooperation
sighpc
AUGUST 9–12, 2021

Scaling Generalized N-Body Problems, a Case Study from Genomics

Presented by Marquita Ellis

Marquita Ellis, Aydīn Buluç, and Katherine Yelick
The University of California at Berkeley
Lawrence Berkeley National Laboratory

Preview (Teaser)

Preview (Teaser)

- For a data-intensive irregular application from genomics,

Preview (Teaser)

- For a data-intensive irregular application from genomics,
- two data-independent approaches to the many-to-many communication are considered here,

Preview (Teaser)

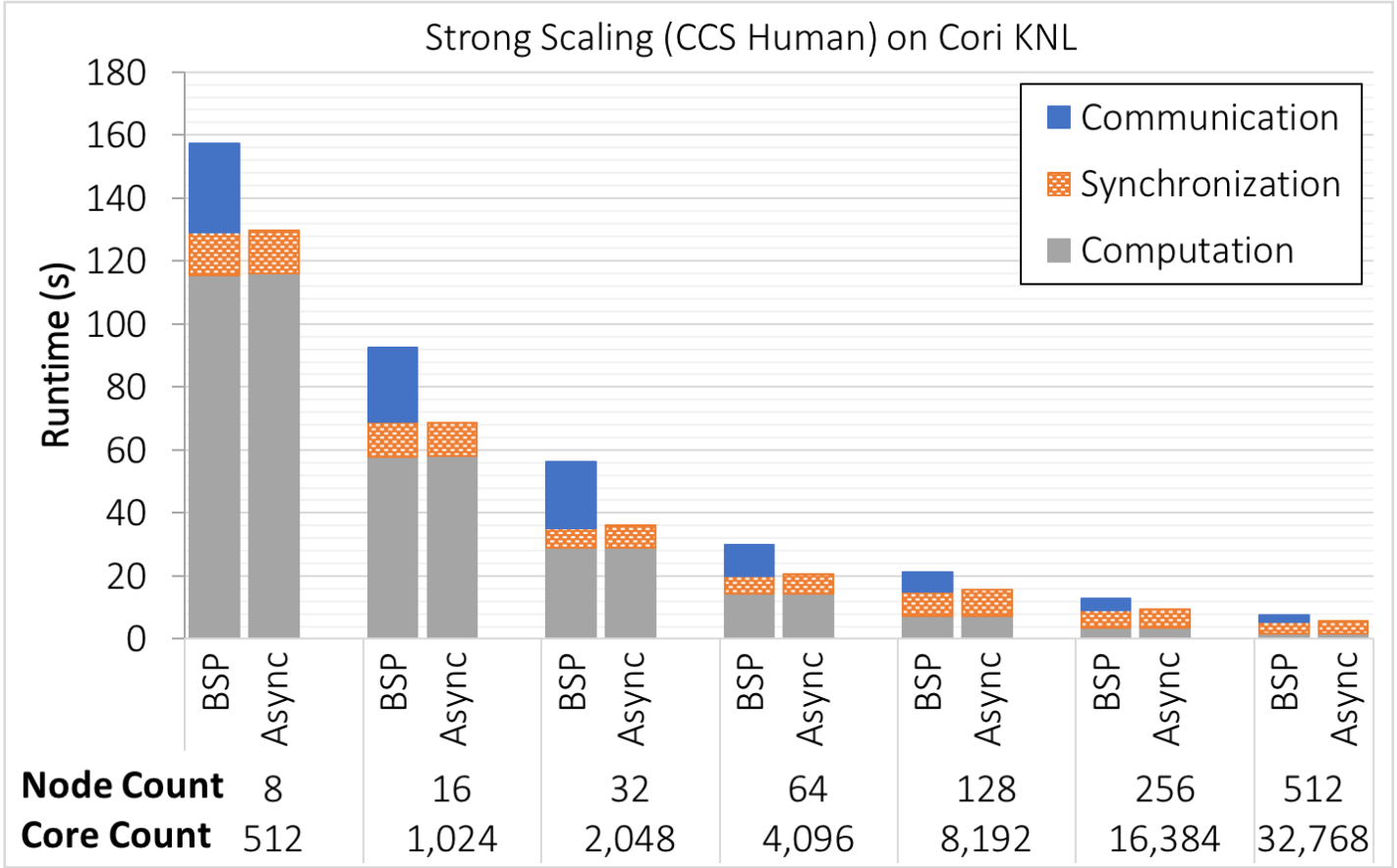
- For a data-intensive irregular application from genomics,
- two data-independent approaches to the many-to-many communication are considered here,
- one maximizing bandwidth utilization and message cost amortization via aggregation (BSP),

Preview (Teaser)

- For a data-intensive irregular application from genomics,
- two data-independent approaches to the many-to-many communication are considered here,
- one maximizing bandwidth utilization and message cost amortization via aggregation (BSP),
- one maximizing injection speed, and communication-hiding (Async)

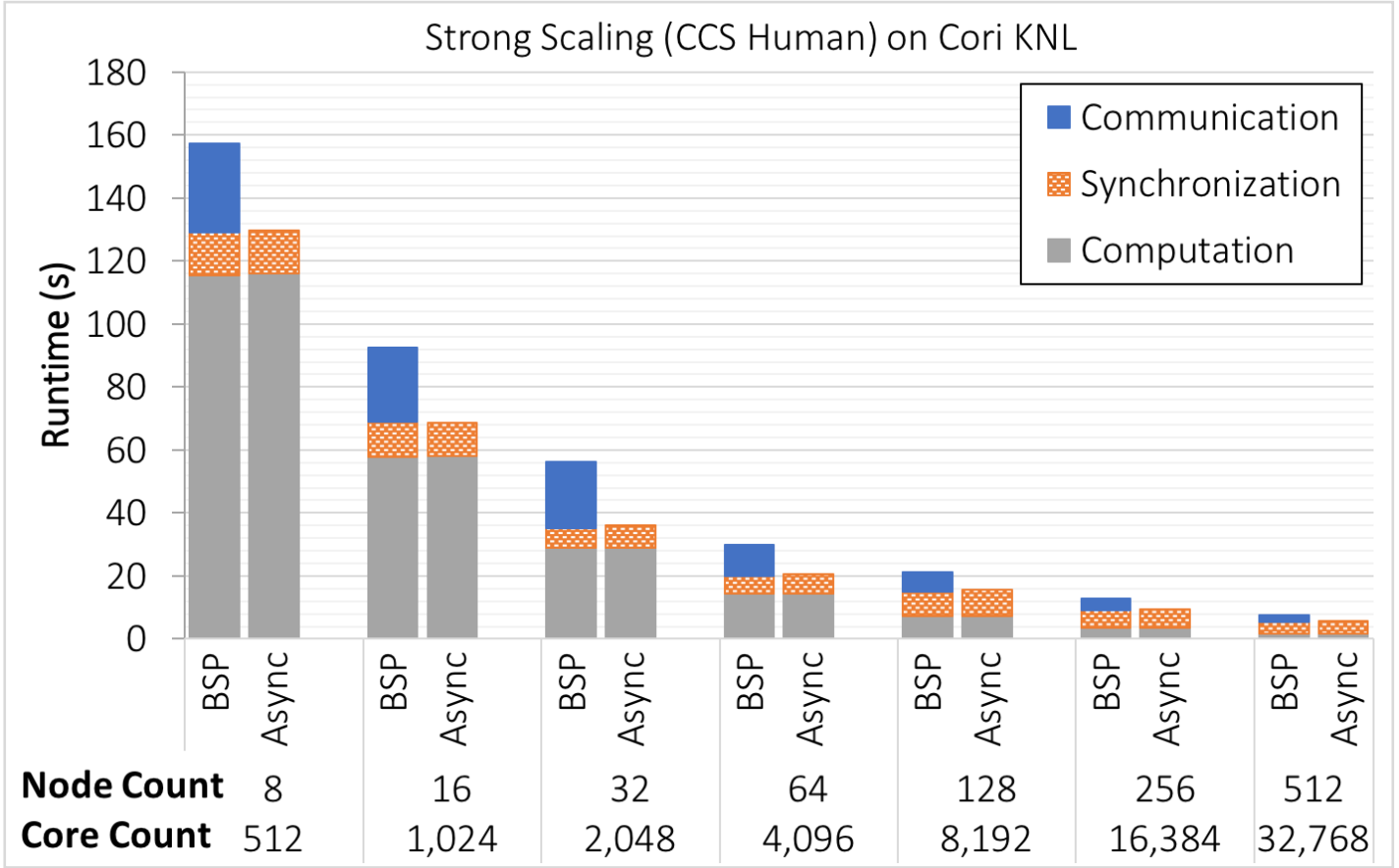
Preview (Teaser)

- For a data-intensive irregular application from genomics,
- two data-independent approaches to the many-to-many communication are considered here,
- one maximizing bandwidth utilization and message cost amortization via aggregation (BSP),
- one maximizing injection speed, and communication-hiding (Async)



Preview (Teaser)

- For a data-intensive irregular application from genomics,
- two data-independent approaches to the many-to-many communication are considered here,
- one maximizing bandwidth utilization and message cost amortization via aggregation (BSP),
- one maximizing injection speed, and communication-hiding (Async)



- At first glance, the communication-hiding approach seems trivially to be the right approach
- However, for this type of problem, there is a non-trivial balancing act between the communication, computation, and memory

Rewind, Why is balancing communication, computation, and memory (and achieving good scaling) for this representative problem non-trivial?

Rewind, Why is balancing communication, computation, and memory (and achieving good scaling) for this representative problem non-trivial?

Application (Case Study) Background

Rewind, Why is balancing communication, computation, and memory (and achieving good scaling) for this representative problem non-trivial?

Application (Case Study) Background

- genome sizes (key factor of the input size) are highly variable and can be large

Rewind, Why is balancing communication, computation, and memory (and achieving good scaling) for this representative problem non-trivial?

Application (Case Study) Background

- genome sizes (key factor of the input size) are highly variable and can be large

Rewind, Why is balancing communication, computation, and memory (and achieving good scaling) for this representative problem non-trivial?

Application (Case Study) Background

- genome sizes (key factor of the input size) are highly variable and can be large
- *sequencers*, which translate nano-meter scale, supercoiling DNA molecules into human- and computer- readable strings, have limitations

Rewind, Why is balancing communication, computation, and memory (and achieving good scaling) for this representative problem non-trivial?

Application Background

- *Sequencers*, translating nano-meter scale, supercoiling molecules into human- and computer- readable strings have limitations
  - cannot read entire genomes at once and so produce many fragments (*reads*)
  - with *errors* — add, delete, substitute base pairs (~characters)

Rewind, Why is balancing communication, computation, and memory (and achieving good scaling) for this representative problem non-trivial?

Application Background

- *Sequencers*, translating nano-meter scale, supercoiling molecules into human- and computer- readable strings have limitations
  - cannot read entire genomes at once and so produce many fragments (*reads*)
  - with *errors* — add, delete, substitute base pairs (~characters)
- Redundancy is used to compensate — also increases the input data size for **our problem** by an order of magnitude

Rewind, Why is balancing communication, computation, and memory (and achieving good scaling) for this representative problem non-trivial?

Application Background

- *Sequencers*, translating nano-meter scale, supercoiling molecules into human- and computer- readable strings have limitations
  - cannot read entire genomes at once and so produce many fragments (*reads*)
  - with *errors* — add, delete, substitute base pairs (~characters)
- Redundancy is used to compensate — also increases the input data size for **our problem** by an order of magnitude
- Redundancy or *repeats* are also inherent in certain genomes, esp. plants, interesting for biofuel, medicine, food, etc.

Rewind, Why are the results non-obvious? Why is achieving good scaling for this problem non-trivial?

Rewind, Why are the results non-obvious? Why is achieving good scaling for this problem non-trivial?

- Sophisticated string similarity measurement (pairwise alignment) is required

Rewind, Why are the results non-obvious? Why is achieving good scaling for this problem non-trivial?

- Sophisticated string similarity measurement (pairwise alignment) is required
- Pairwise alignment is $O(n^2)$ for strings of length n,
  - in practice $n \in (10^3, 10^5)$ and highly variable

Rewind, Why are the results non-obvious? Why is achieving good scaling for this problem non-trivial?

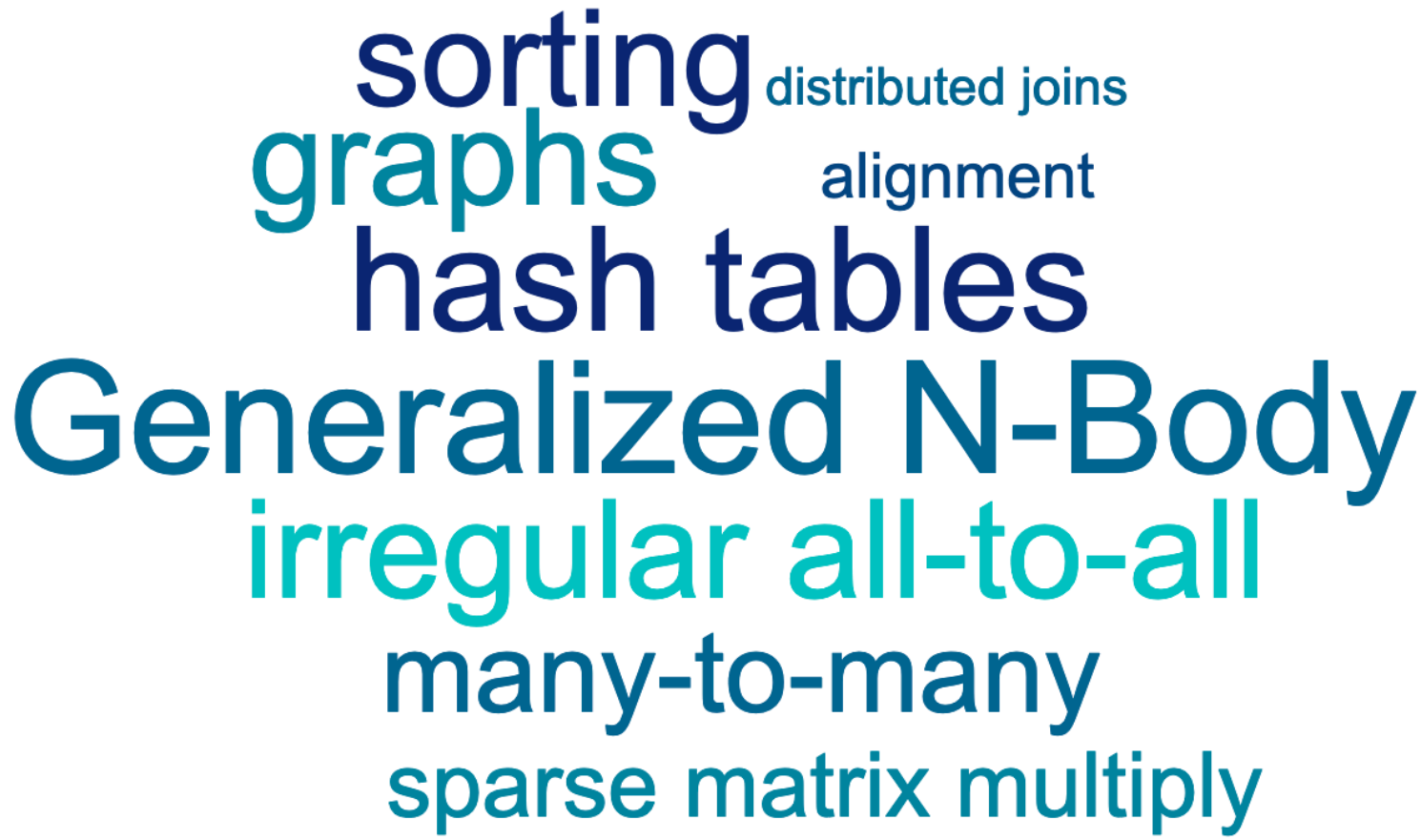
- Sophisticated string similarity measurement (pairwise alignment) is required
- Pairwise alignment is $O(n^2)$ for strings of length n,
  - in practice $n \in (10^3, 10^5)$ and highly variable
- With N *reads*, problem can be solved in $O(N^2 \times n^2)$ — quickly becomes intractable

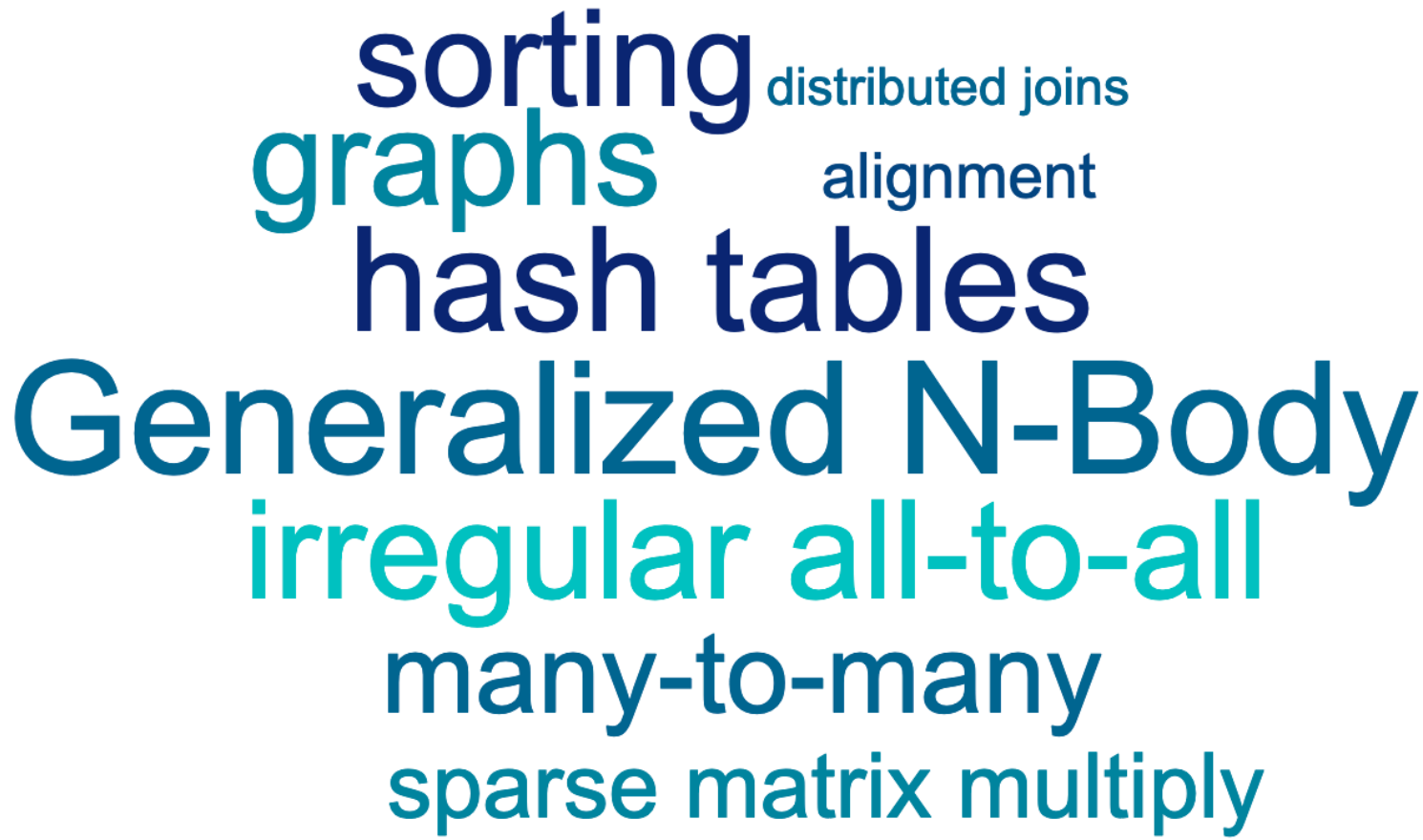
Rewind, Why are the results non-obvious? Why is achieving good scaling for this problem non-trivial?

- Sophisticated string similarity measurement (pairwise alignment) is required
- Pairwise alignment is $O(n^2)$ for strings of length n,
  - in practice $n \in (10^3, 10^5)$ and highly variable
- With N *reads*, problem can be solved in $O(N^2 \times n^2)$ — quickly becomes intractable
- In practice, runtime analysis and filtering is used to reduce the N^2 to … something smaller…
  - yields a sparse unstructured graph that is also very large, discovered at runtime

Rewind, Why are the results non-obvious? Why is achieving good scaling for this problem non-trivial?

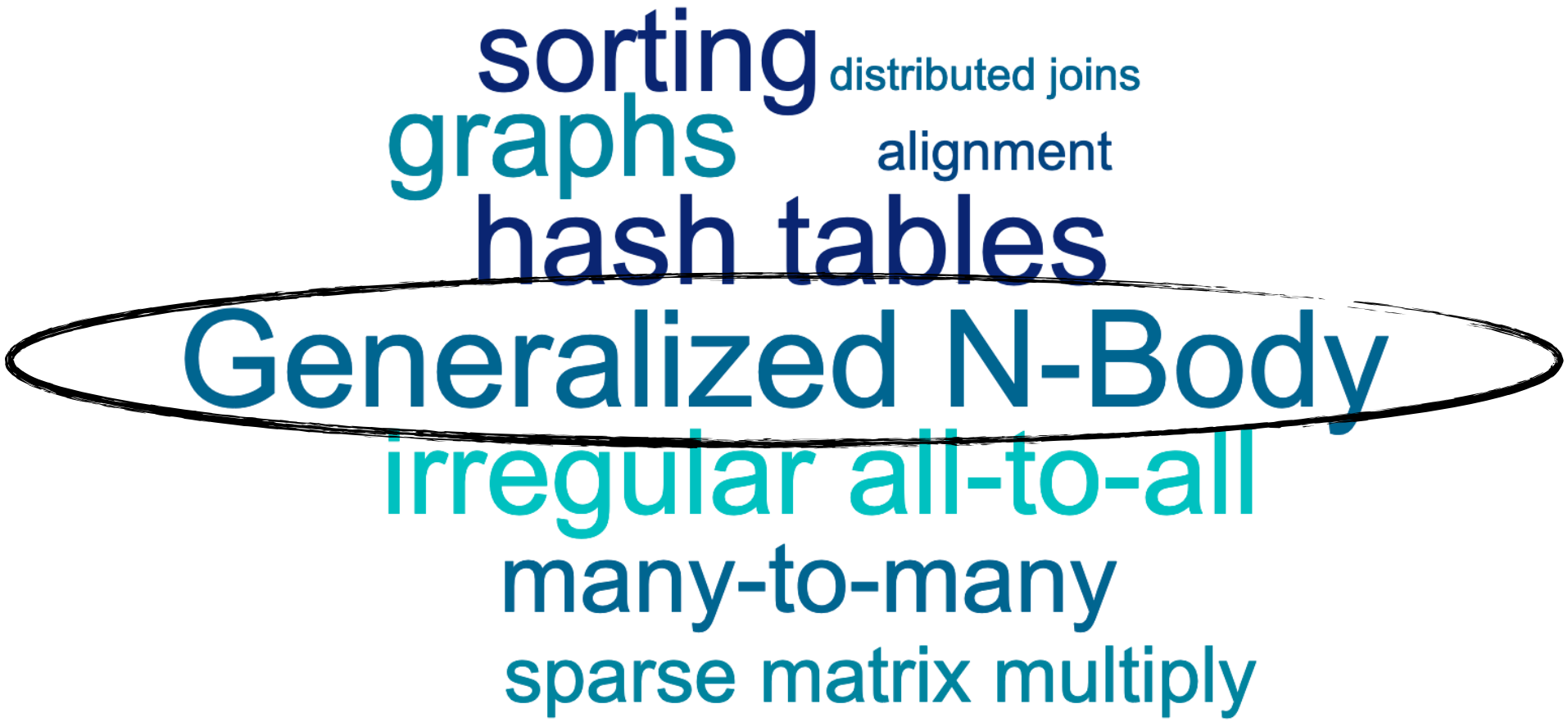
- Sophisticated string similarity measurement (pairwise alignment) is required
- Pairwise alignment is $O(n^2)$ for strings of length n,
  - in practice $n \in (10^3, 10^5)$ and highly variable
- With N *reads*, problem can be solved in $O(N^2 \times n^2)$ — quickly becomes intractable
- In practice, runtime analysis and filtering is used to reduce the N^2 to … something smaller…
  - yields a sparse unstructured graph that is also very large, discovered at runtime
- Heuristic approaches to pairwise alignment can reduce the n^2 to average-case $O(n)$
  - their early termination leads to *more* irregularity in the computation at runtime


sorting
distributed joins
graphs
alignment
hash tables
Generalized N-Body
irregular all-to-all
many-to-many
sparse matrix multiply



See also Yelick et al., "The Parallelism Motifs of Genomic Analysis." In Philosophical Transactions of the Royal Society A 378, no. 2166 (2020): 20190394.

Our case study is a representative Generalized N-Body problem from Genomics



See also Yelick et al., "The Parallelism Motifs of Genomic Analysis."
In Philosophical Transactions of the Royal Society A 378, no. 2166 (2020): 20190394.

A Generalized N-Body Problem (Gray and Moore, 2001)

A Generalized N-Body Problem (Gray and Moore, 2001)

- Classic N-Body: simulate the motion of “bodies” (e.g. stars and planets, atoms, pinballs,…) according to Newton’s Laws

A Generalized N-Body Problem (Gray and Moore, 2001)

- Classic N-Body: simulate the motion of “bodies” (e.g. stars and planets, atoms, pinballs,…) according to Newton’s Laws
- Generalized N-Body: measure some type of similarity between (all or many) pairs or tuples of bodies i.e. measurements are many-to-many or all-to-all and may be non-Euclidean

A Generalized N-Body Problem (Gray and Moore, 2001)

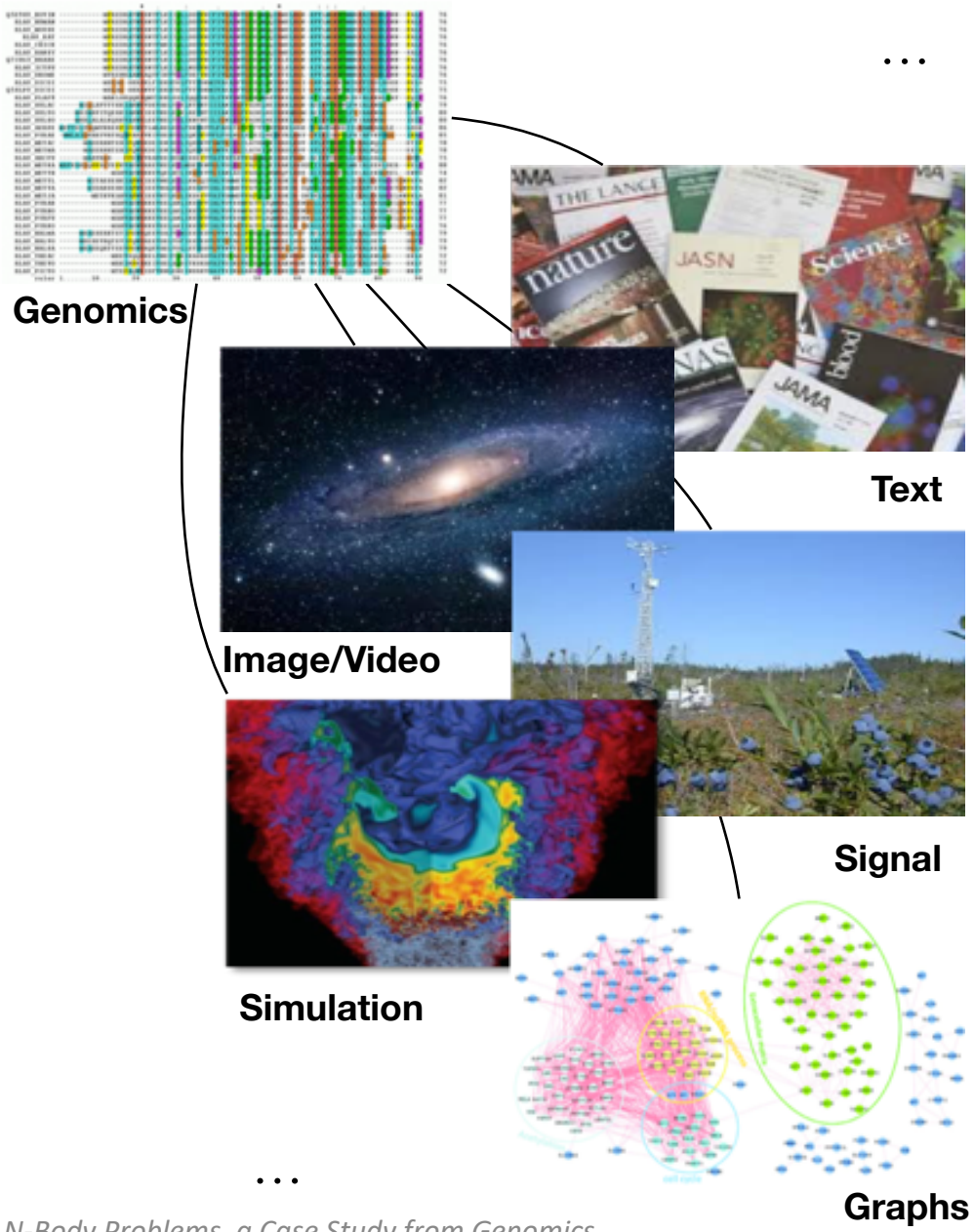
- Classic N-Body: simulate the motion of “bodies” (e.g. stars and planets, atoms, pinballs,…) according to Newton’s Laws
- Generalized N-Body: measure some type of similarity between (all or many) pairs or tuples of bodies i.e. measurements are many-to-many or all-to-all and may be non-Euclidean
- Our Generalized N-Body instance is a many-to-many comparison of long, variable length strings (bodies) via pairwise alignment (similarity metric)

A Generalized N-Body Problem (Gray and Moore, 2001)

- Classic N-Body: simulate the motion of “bodies” (e.g. stars and planets, atoms, pinballs,…) according to Newton’s Laws
- Generalized N-Body: measure some type of similarity between (all or many) pairs or tuples of bodies i.e. measurements are many-to-many or all-to-all and may be non-Euclidean
- Our Generalized N-Body instance is a many-to-many comparison of long, variable length strings (bodies) via pairwise alignment (similarity metric)
- with significant similarities to other Generalized N-Body problems in Bioinformatics
  - pangenomics, similarity across genomes
  - metagenomics, clustering DNA fragments from different species in the same sample
  - proteomics, similarity searches in massive protein data sets
  - informatics, more general text and document analysis

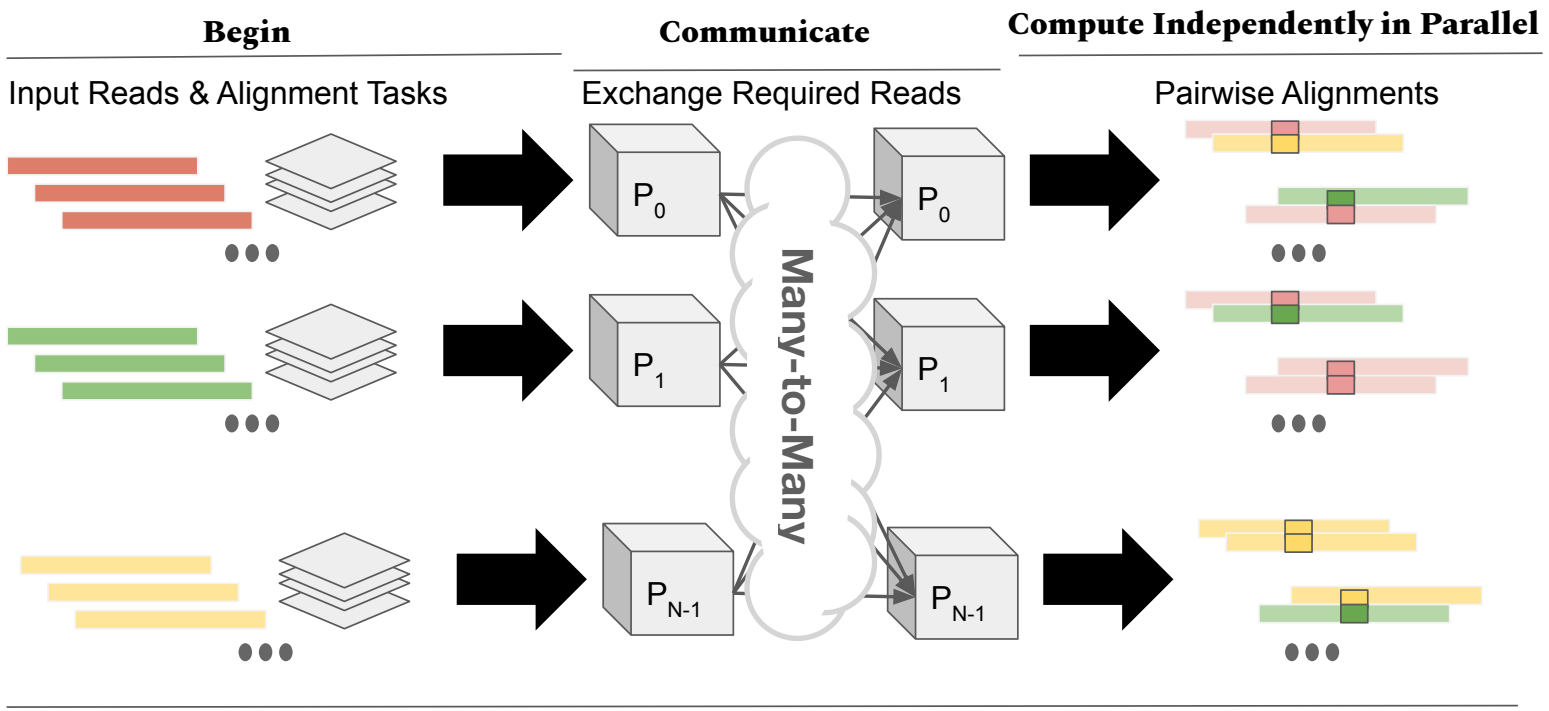
Big Picture Goals

- Develop scalable software for scientific discovery
  - build “wet” science - computer science bridges
  - in collaboration with JGI, UCB, MCB@UCB
  - Exascale Computing Project
- Generalize and generate insights for scaling other irregular big data applications



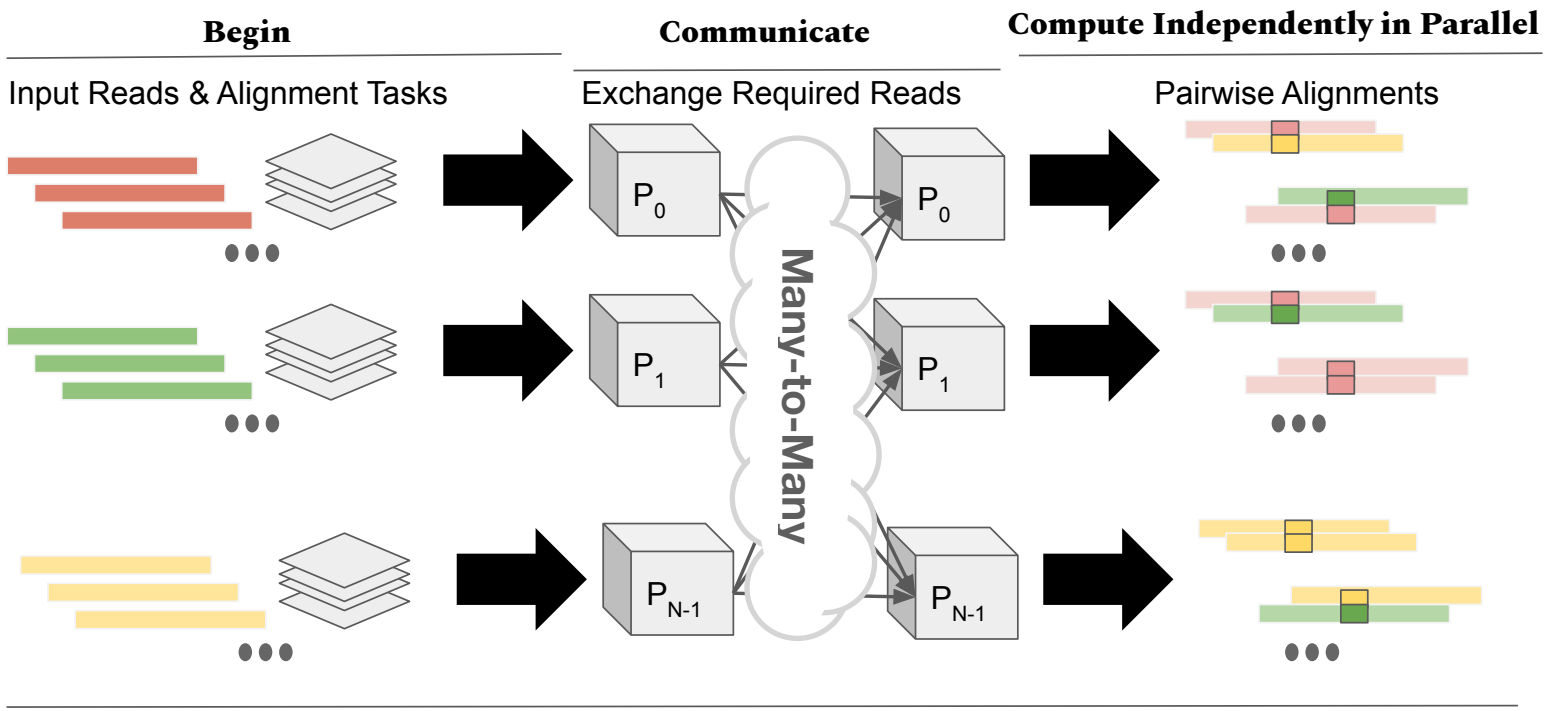
Big Data

Data-Independent Approach (1/2), Bulk-Synchronous Parallel (BSP)



End: output alignments exceeding minimum scoring threshold in parallel

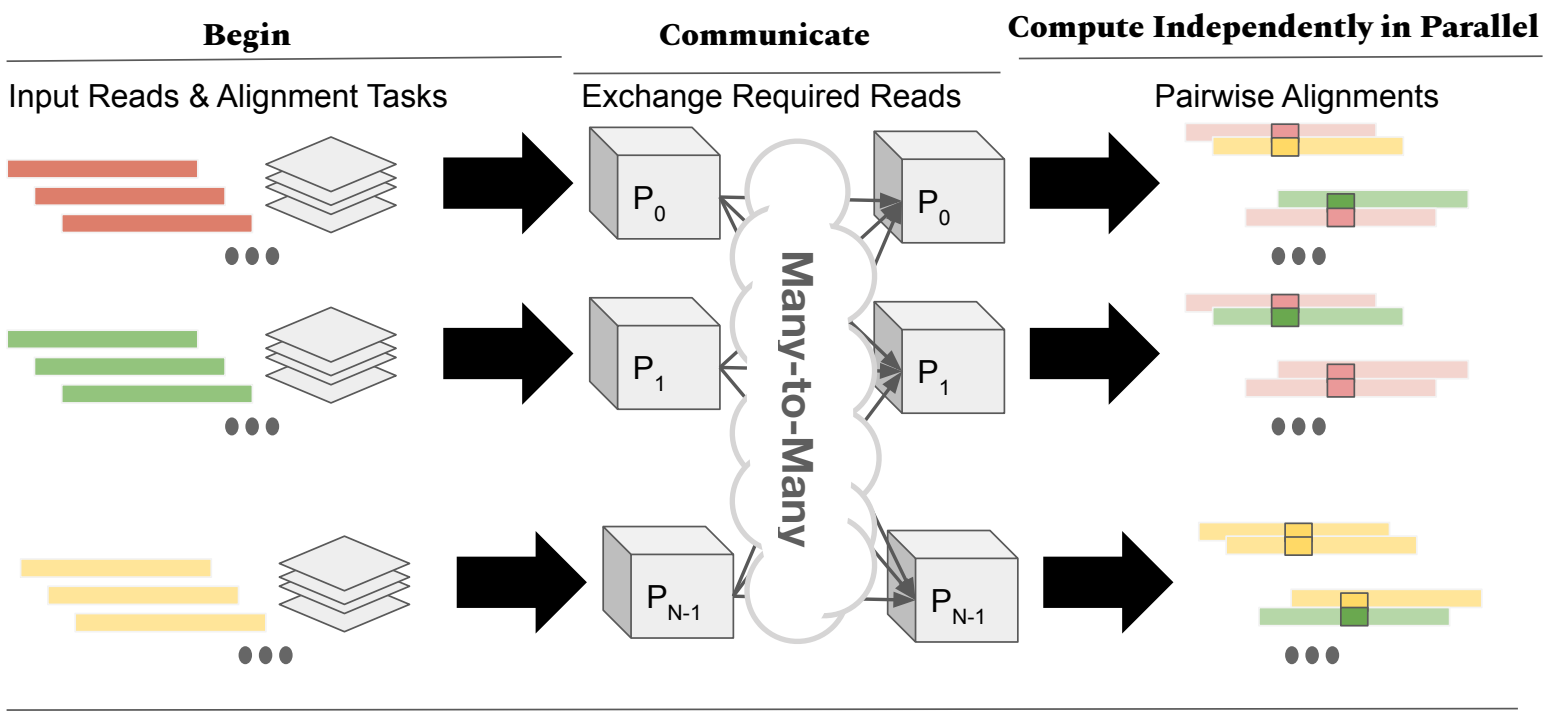
Data-Independent Approach (1/2), Bulk-Synchronous Parallel (BSP)



End: output alignments exceeding minimum scoring threshold in parallel

- Maximizes bandwidth utilization of sparse many-to-many string exchange

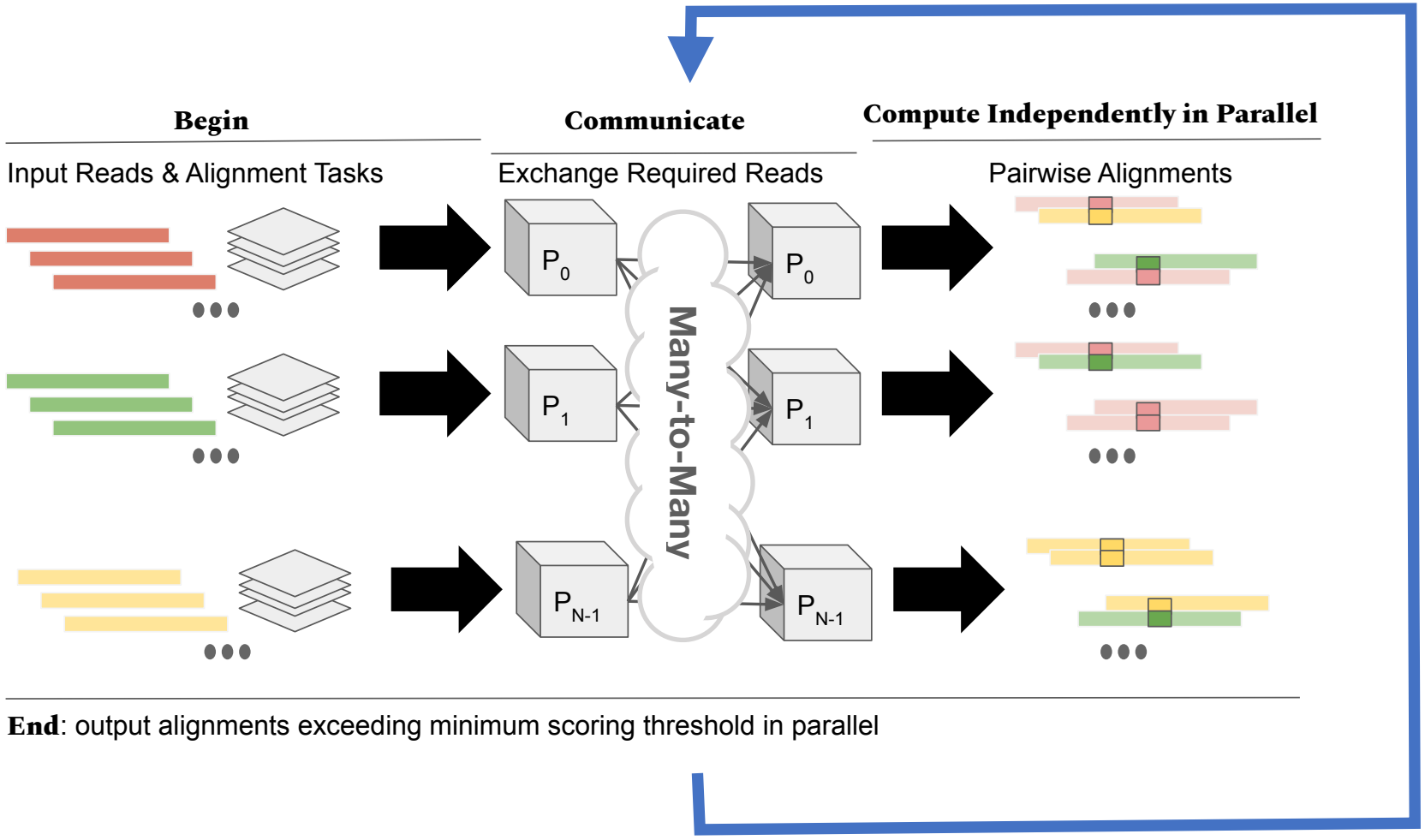
Data-Independent Approach (1/2), Bulk-Synchronous Parallel (BSP)



End: output alignments exceeding minimum scoring threshold in parallel

- Maximizes bandwidth utilization of sparse many-to-many string exchange
- Maximizes independently parallel pairwise alignment computation

Data-Independent Approach (1/2), Bulk-Synchronous Parallel (BSP)



- Maximizes bandwidth utilization of sparse many-to-many string exchange
- Maximizes independently parallel pairwise alignment computation

In practice, communicating in multiple memory-limited exchanges may be necessary

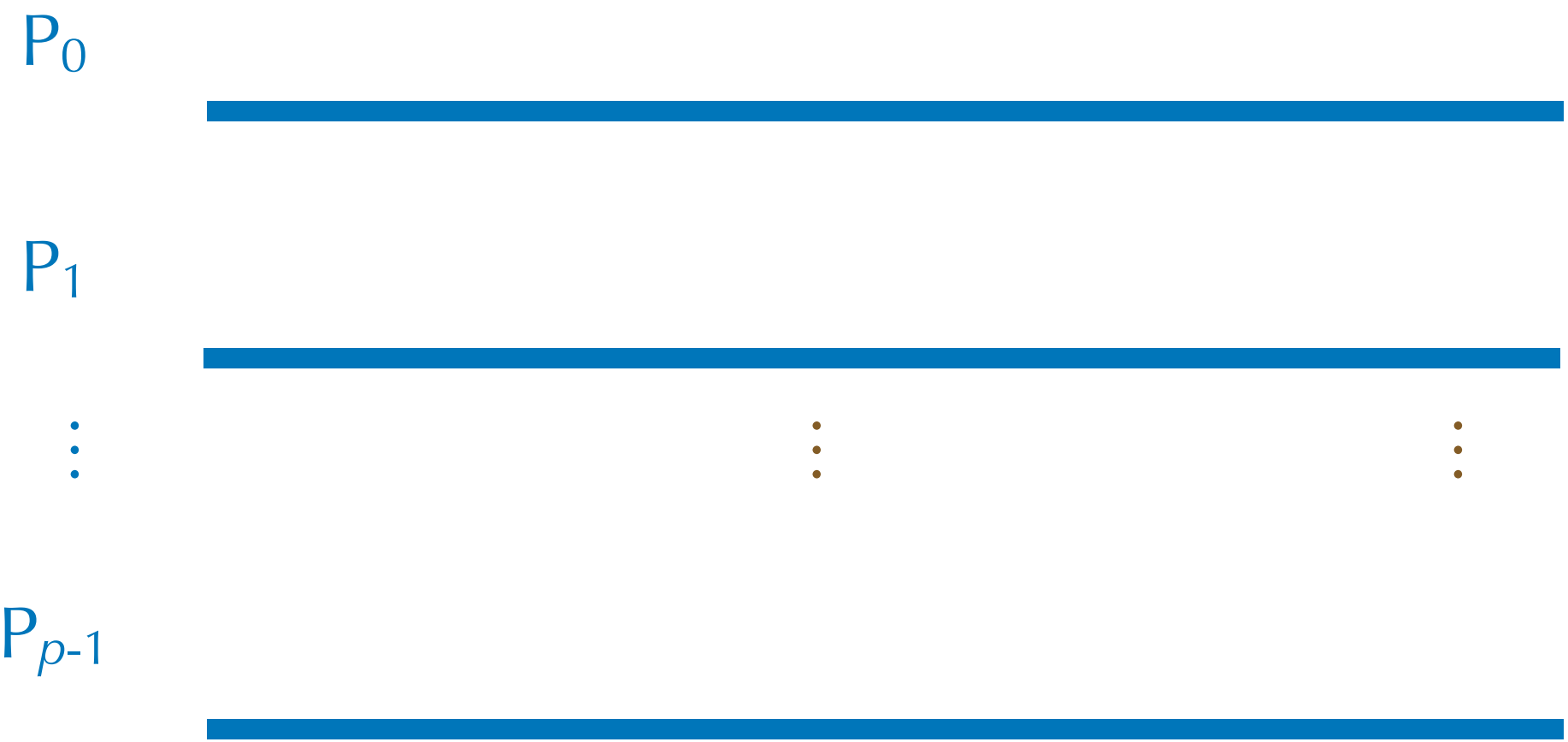
Data-Independent Approach (2/2), Asynchronous (Async)

Data-Independent Approach (2/2), Asynchronous (Async)

- Each remote string is retrieved one at a time, asynchronously (as necessary)

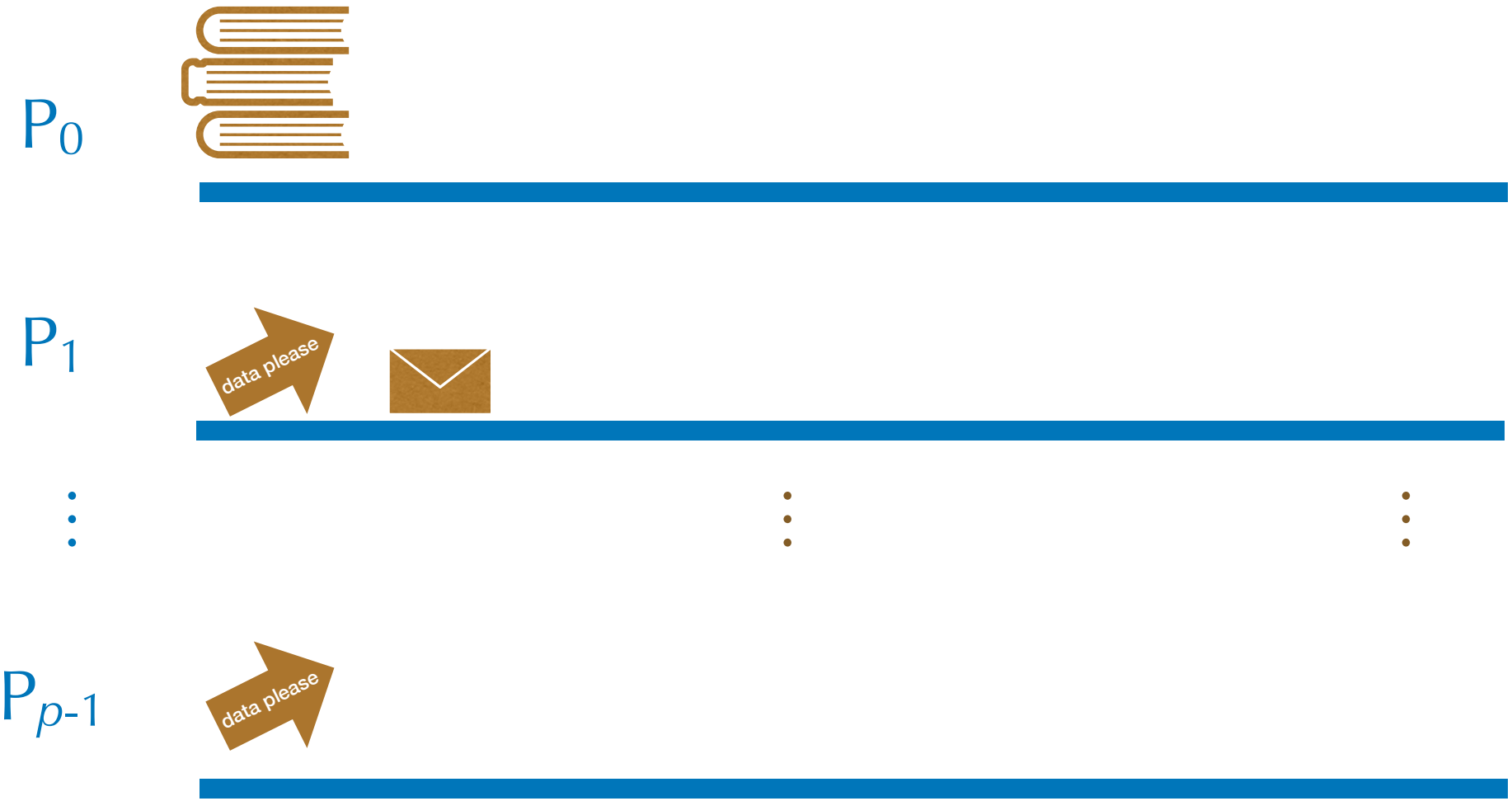
Data-Independent Approach (2/2), Asynchronous (Async)

High-Level Illustration (SPMD)



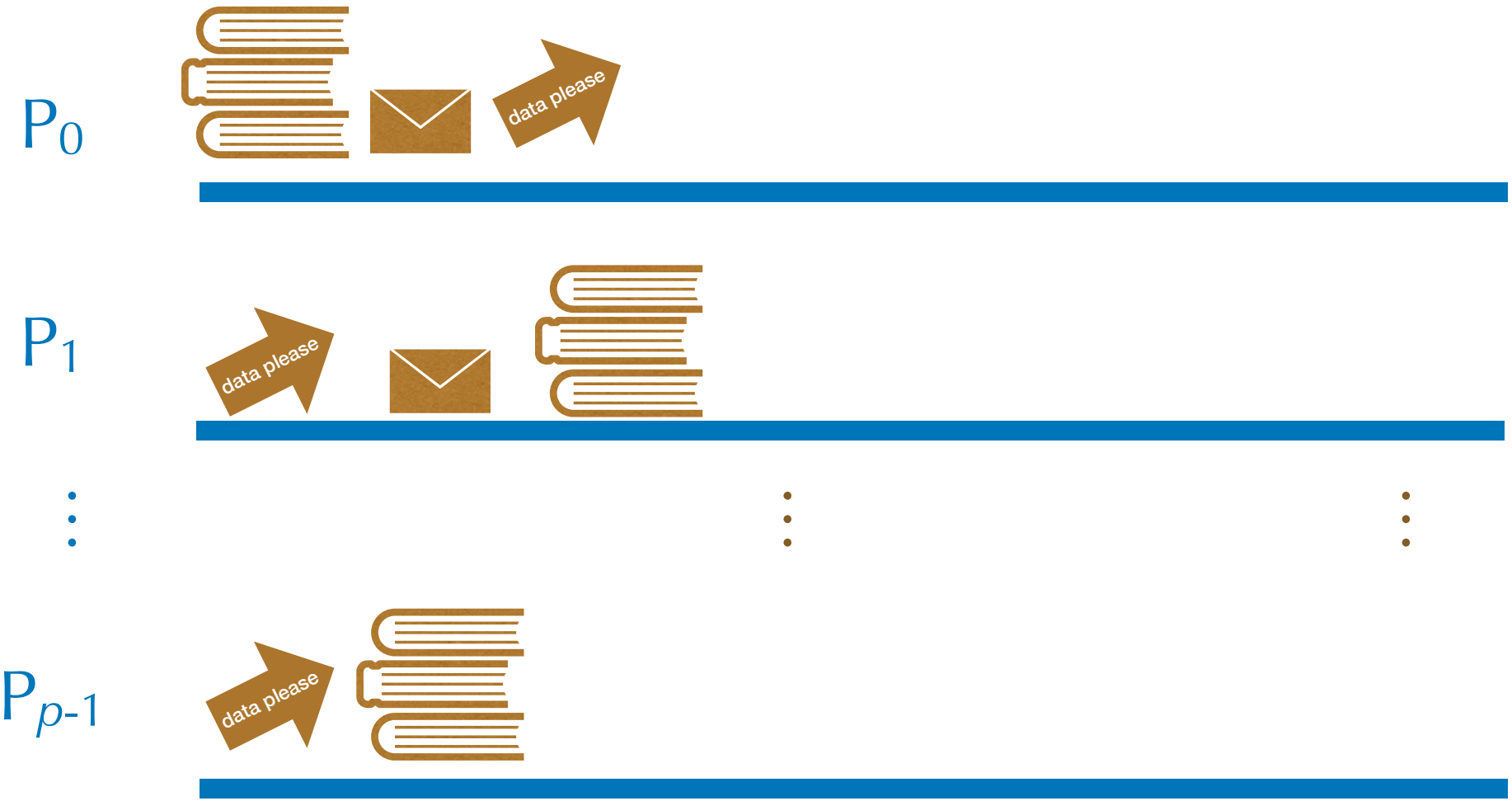
Data-Independent Approach (2/2), Asynchronous (Async)

High-Level Illustration (SPMD)



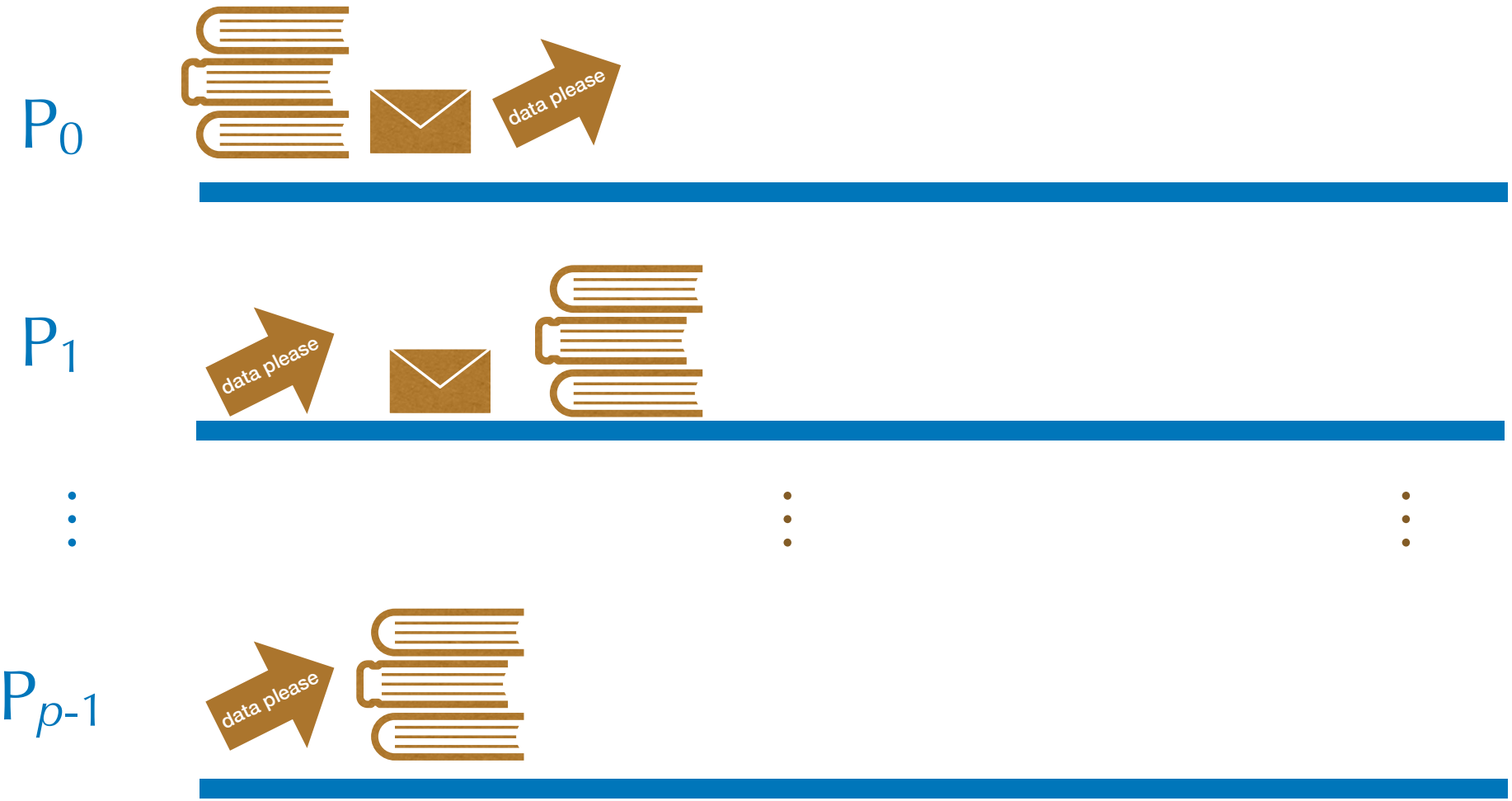
Data-Independent Approach (2/2), Asynchronous (Async)

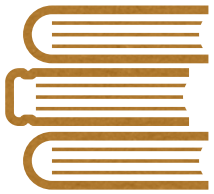
High-Level Illustration (SPMD)



Data-Independent Approach (2/2), Asynchronous (Async)

High-Level Illustration (SPMD)

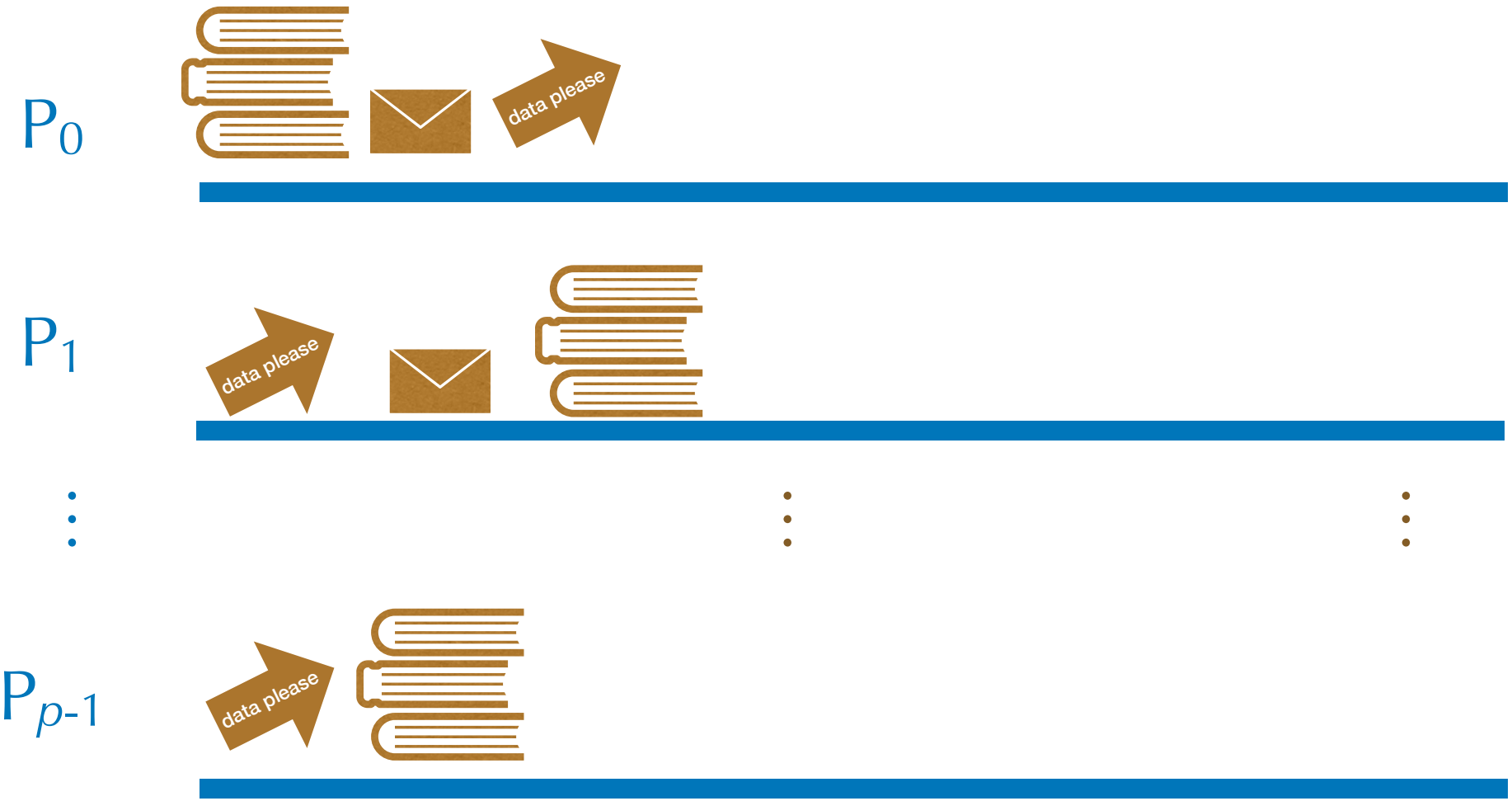


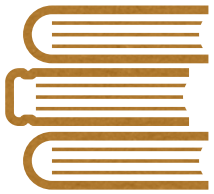


Potentially long-running pairwise alignment tasks

Data-Independent Approach (2/2), Asynchronous (Async)

High-Level Illustration (SPMD)



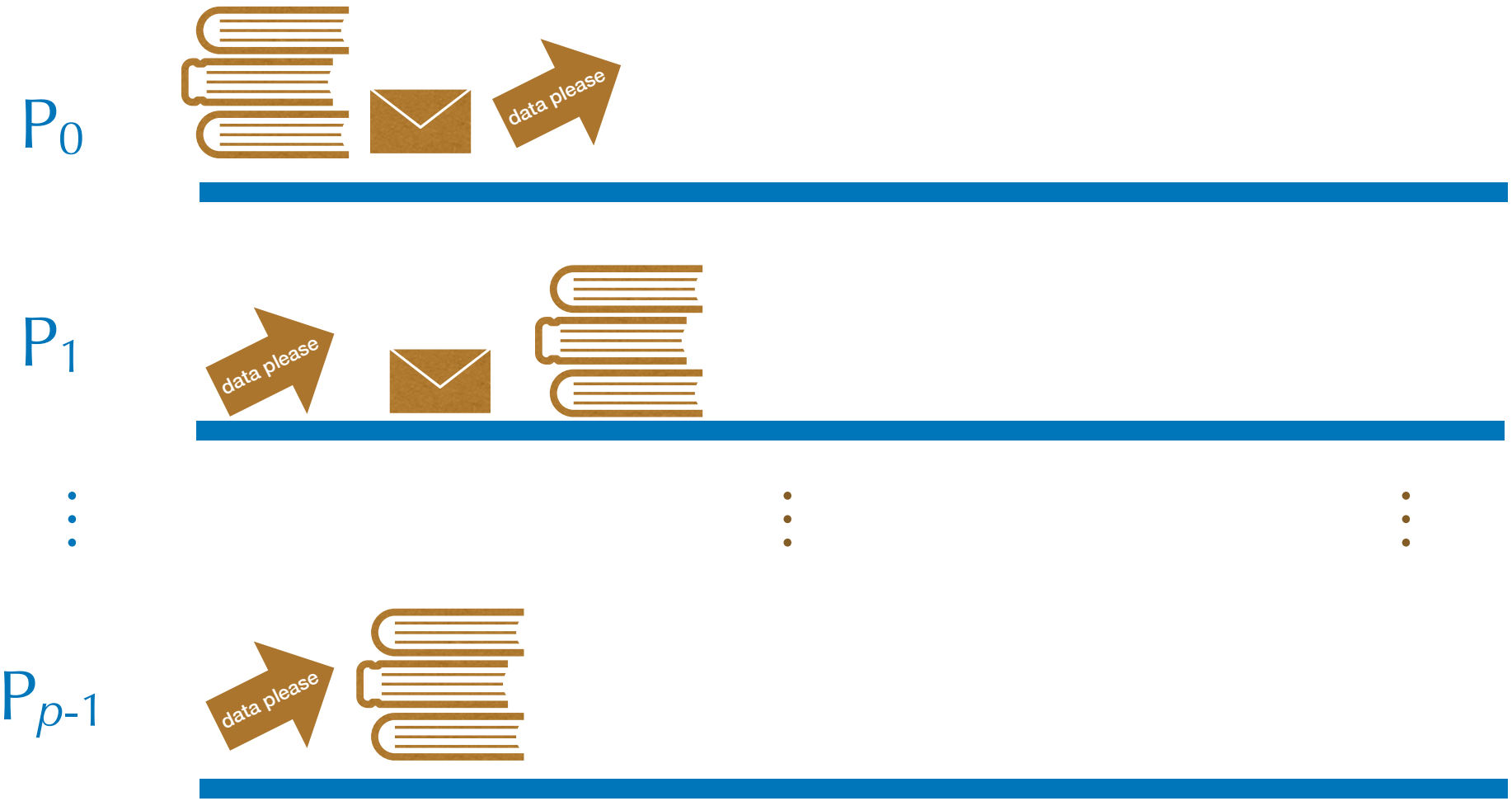


Potentially long-running pairwise alignment tasks

Nonblocking request for remote data

Data-Independent Approach (2/2), Asynchronous (Async)

High-Level Illustration (SPMD)



Potentially long-running pairwise alignment tasks

Nonblocking request for remote data

Asynchronous data lookup-and-send task

Data-Independent Approach (2/2), Asynchronous (Async)

High-Level Illustration (SPMD)

Potentially long-running pairwise alignment tasks

Nonblocking request for remote data

Asynchronous data lookup-and-send task

Data-Independent Approach (2/2), Asynchronous (Async)

High-Level Illustration (SPMD)

Potentially long-running pairwise alignment tasks

Nonblocking request for remote data

Asynchronous data lookup-and-send task

Data-Independent Approach (2/2), Asynchronous (Async)

High-Level Illustration (SPMD)

Potentially long-running pairwise alignment tasks

data please

Nonblocking request for remote data

Asynchronous data lookup-and-send task

Unclear how well the *Async* approach will perform in practice

Potential Advantages

- ~maximizes injection speed
- ~minimizes memory footprint
- communication-computation hiding

…

Potential Disadvantages

- ~maximizes number of messages
- pays round-trip cost for each remote read (datum) as needed
- exacerbated load imbalance

…

How do these approaches perform with real workloads?

Experimental Setup

Intra-node strong scaling

1 - 128 node strong scaling

8 - 512 node strong scaling

Short Name	Species	Reads	Tasks
E. coli 30×	*Escherichia coli*	16,890	2,270,260
E. coli 100×	*Escherichia coli*	91,394	24,869,171
Human CCS	*Homo sapiens*	1,148,839	87,621,409

Cray XC40, “Cori KNL” (KNL partition)
Cray Aries, Dragonfly Topology
Node architecture:

- Single socket
- Intel Xeon Phi Knights Landing processor
- 68 cores @1.4 GHz
- 4-way hardware hyperthreading
- 16 GB MCDRAM HBM
- 96 GB DDR

— more description and details available in the full text —

How do these approaches perform with real workloads?

Experimental Setup

Intra-node strong scaling

1 - 128 node strong scaling

8 - 512 node strong scaling

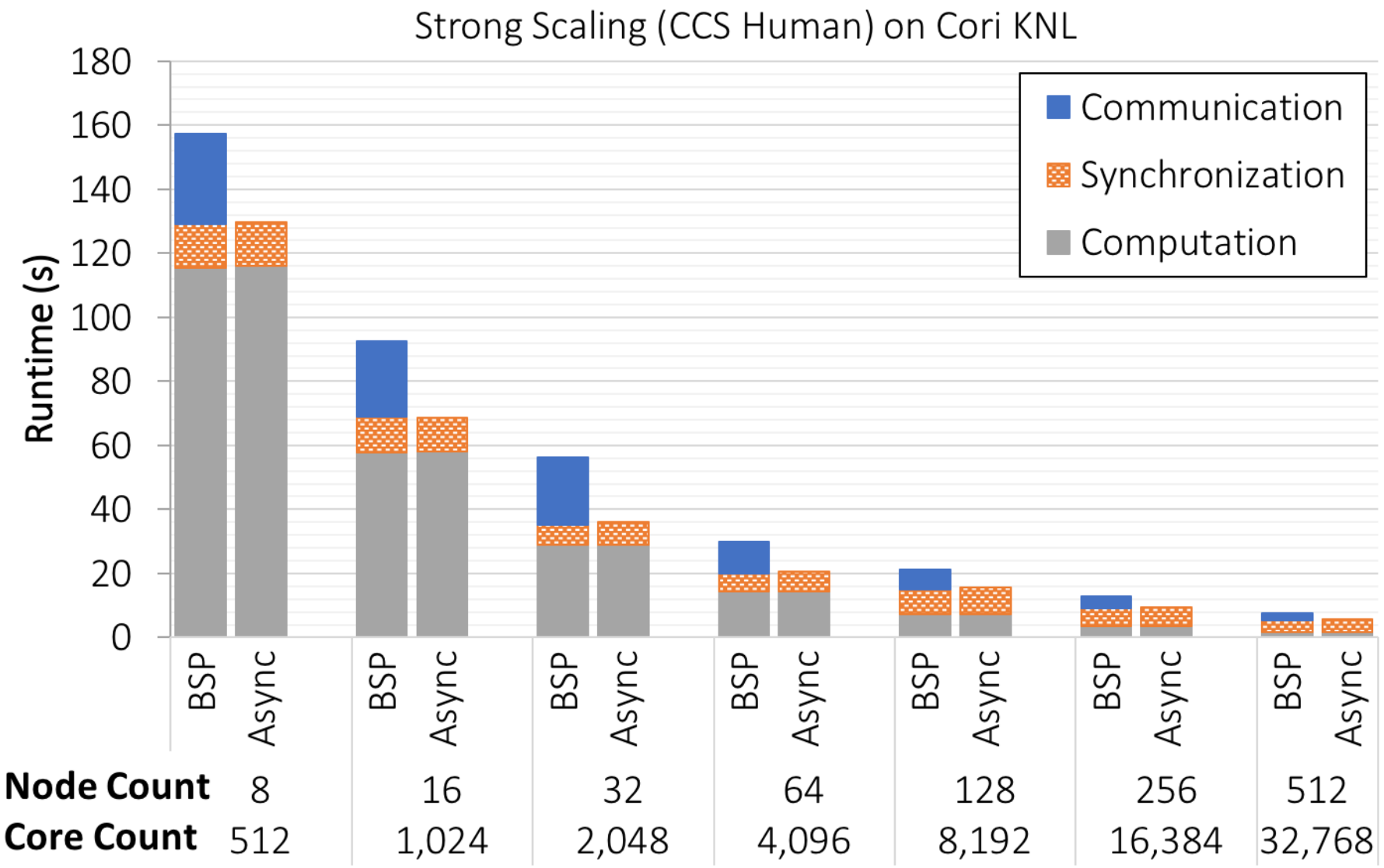
Short Name	Species	Reads	Tasks
E. coli 30×	*Escherichia coli*	16,890	2,270,260
E. coli 100×	*Escherichia coli*	91,394	24,869,171
Human CCS	*Homo sapiens*	1,148,839	87,621,409

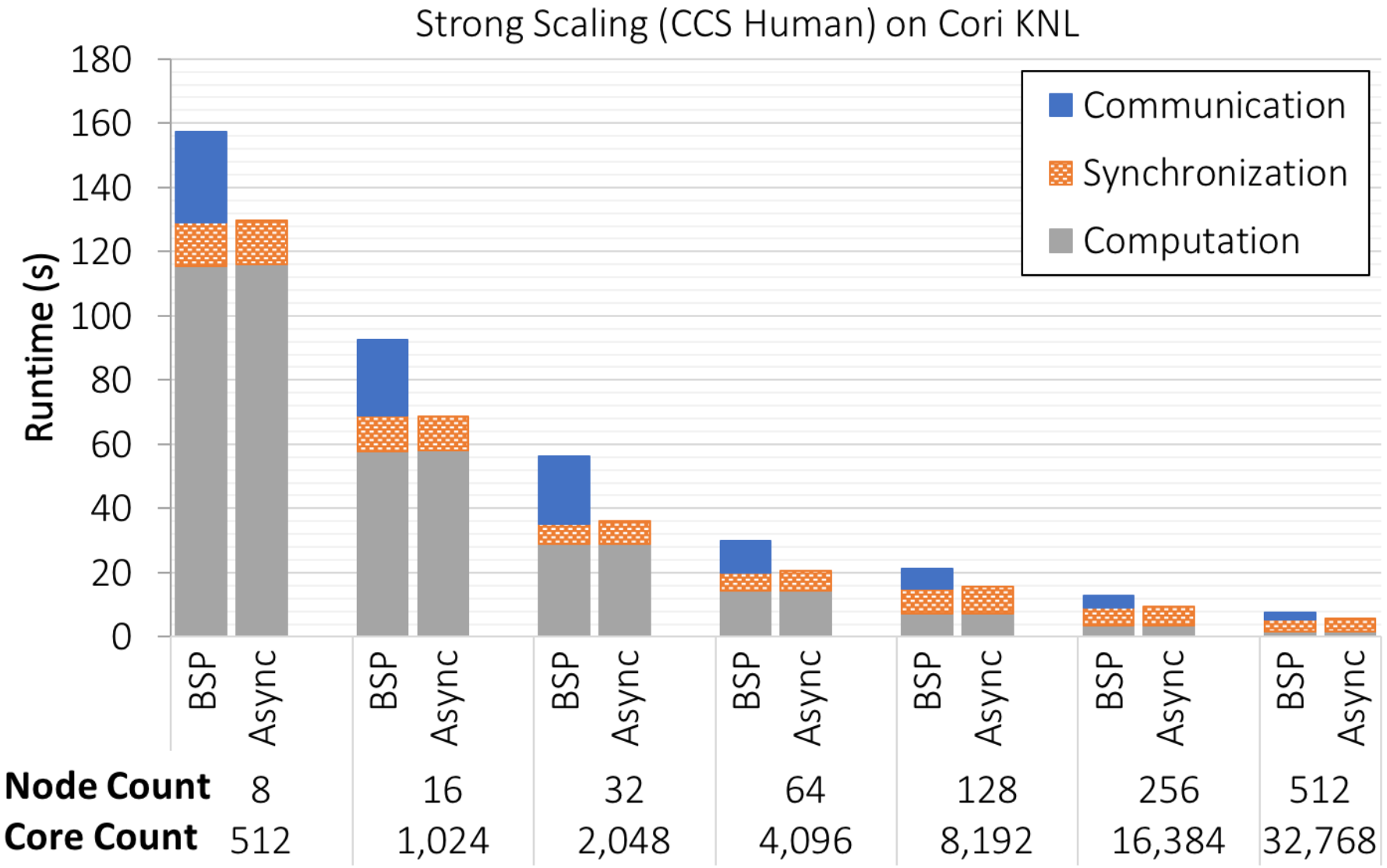
In this talk

Cray XC40, "Cori KNL" (KNL partition)
Cray Aries, Dragonfly Topology
Node architecture:

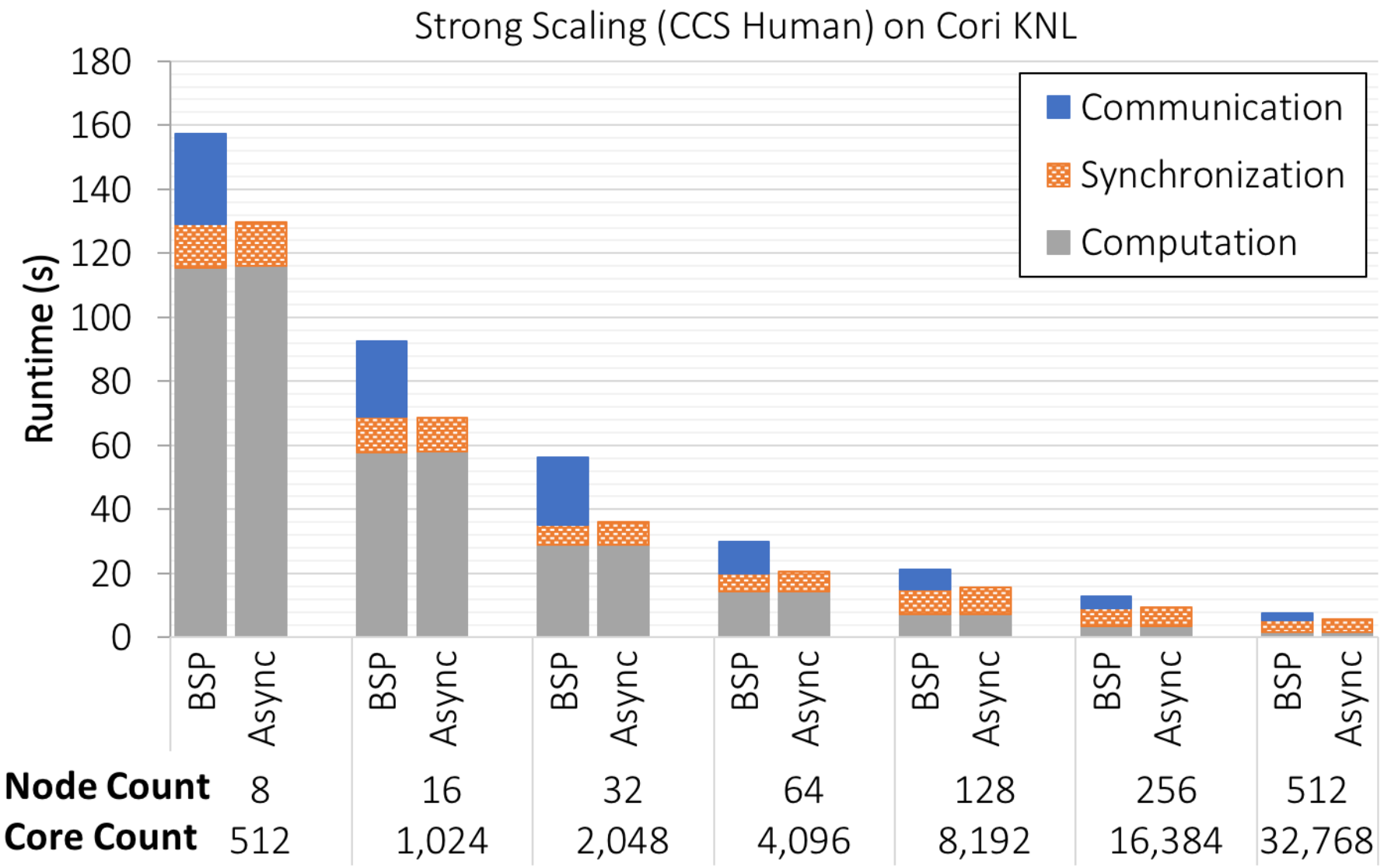
- Single socket
- Intel Xeon Phi Knights Landing processor
- 68 cores @1.4 GHz
- 4-way hardware hyperthreading
- 16 GB MCDRAM HBM
- 96 GB DDR

— more description and details available in the full text —

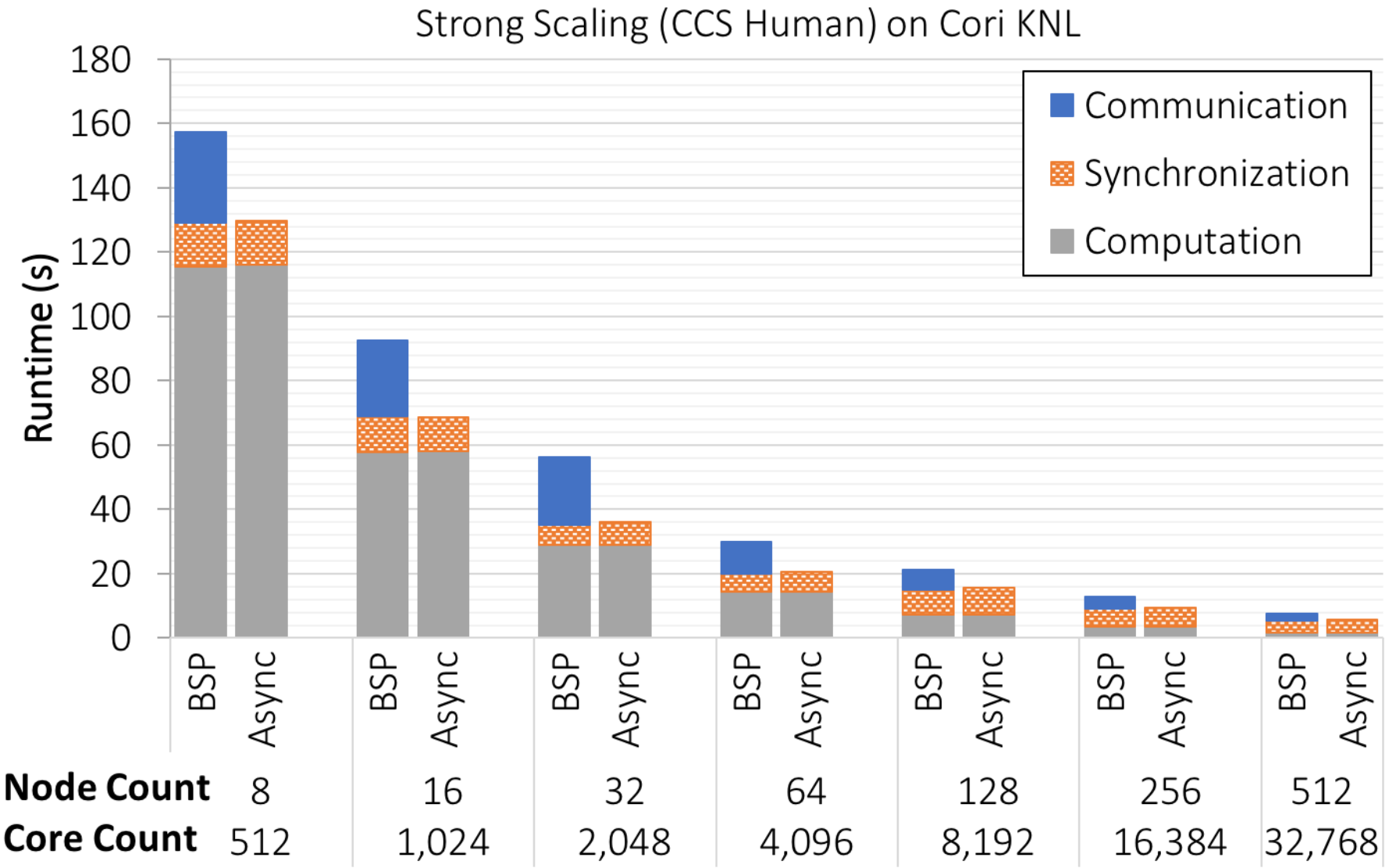

Strong Scaling (CCS Human) on Cori KNL
Runtime (s)
180
160
140
120
100
80
60
40
20
0
Communication
Synchronization
Computation
BSP
Async
Node Count
8
16
32
64
128
256
512
Core Count
512
1,024
2,048
4,096
8,192
16,384
32,768



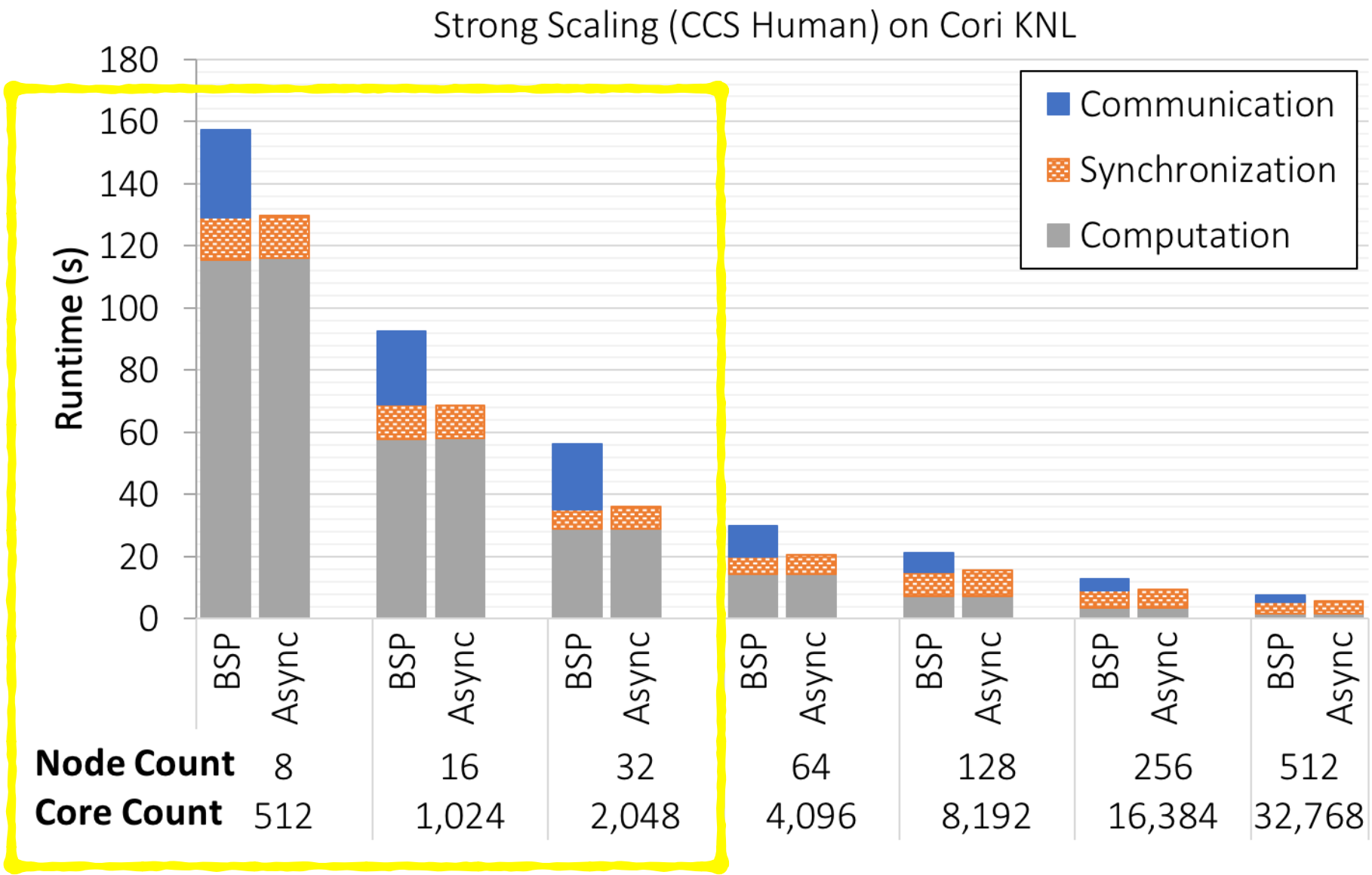
- same computation time (gray, bottom bars) is expected (same workload)



- same computation time (gray, bottom bars) is expected (same workload)
- synchronization time (orange, middle bars) is dominated by load imbalance in individual pairwise alignments (module shared by each)

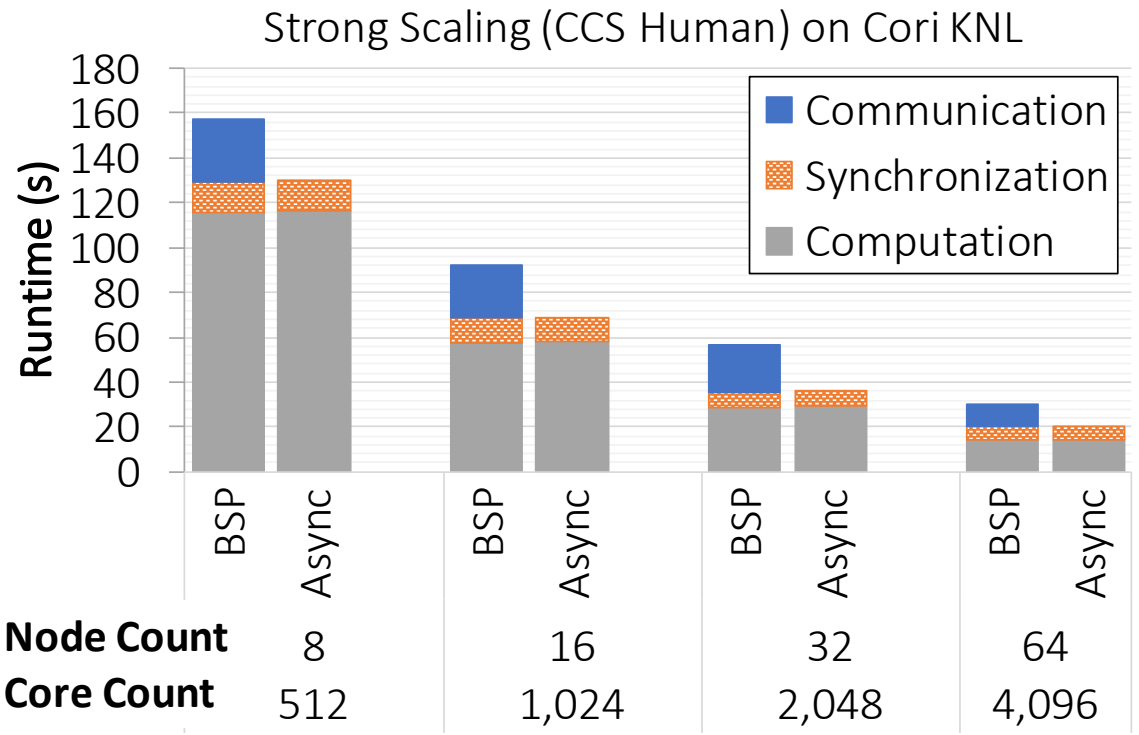


- same computation time (gray, bottom bars) is expected (same workload)
- synchronization time (orange, middle bars) is dominated by load imbalance in individual pairwise alignments (module shared by each)
- focus now: communication time (blue, top bars)

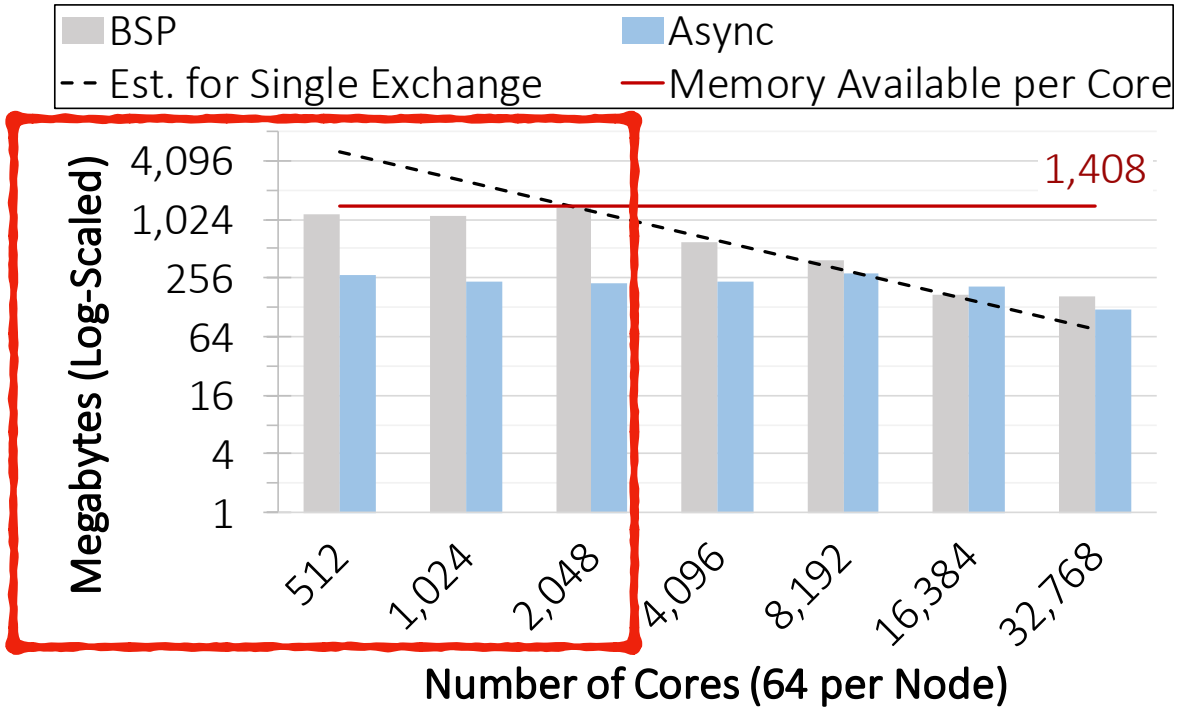


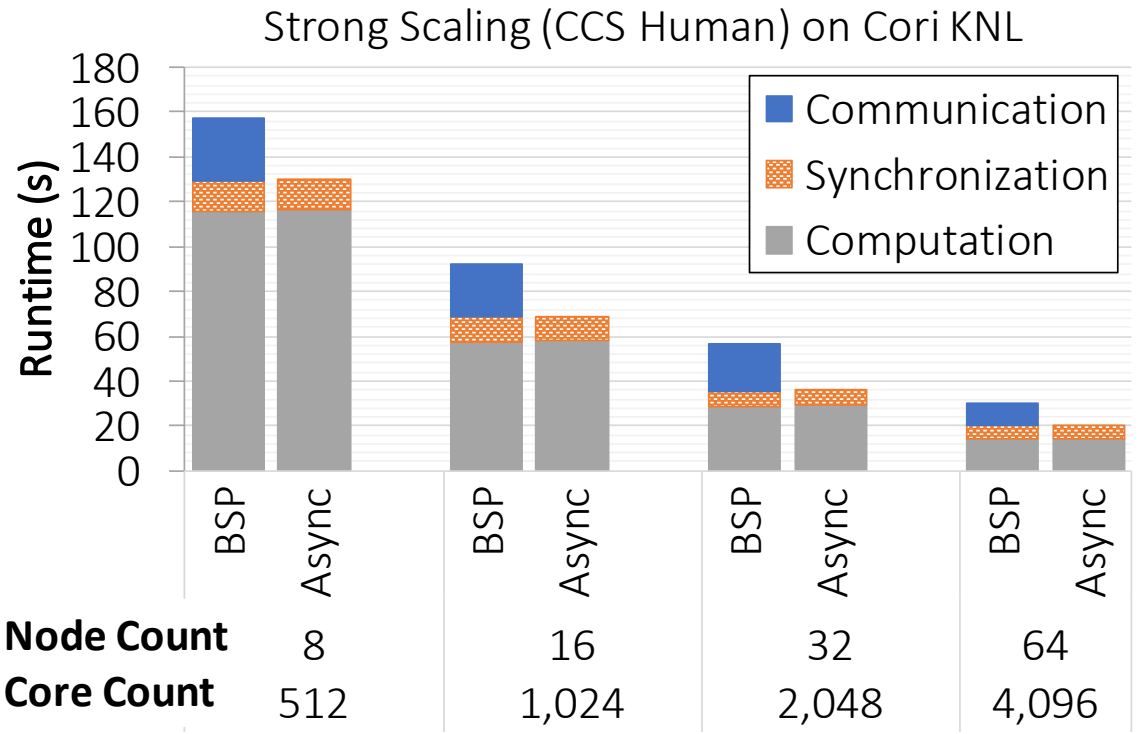
- **8-32** nodes, BSP communication overhead is **18-37%**
- Async is up to 20% more efficient with effective communication hiding

8-32 nodes, memory limits necessitate multiple BSP exchanges, increasing overall communication overhead

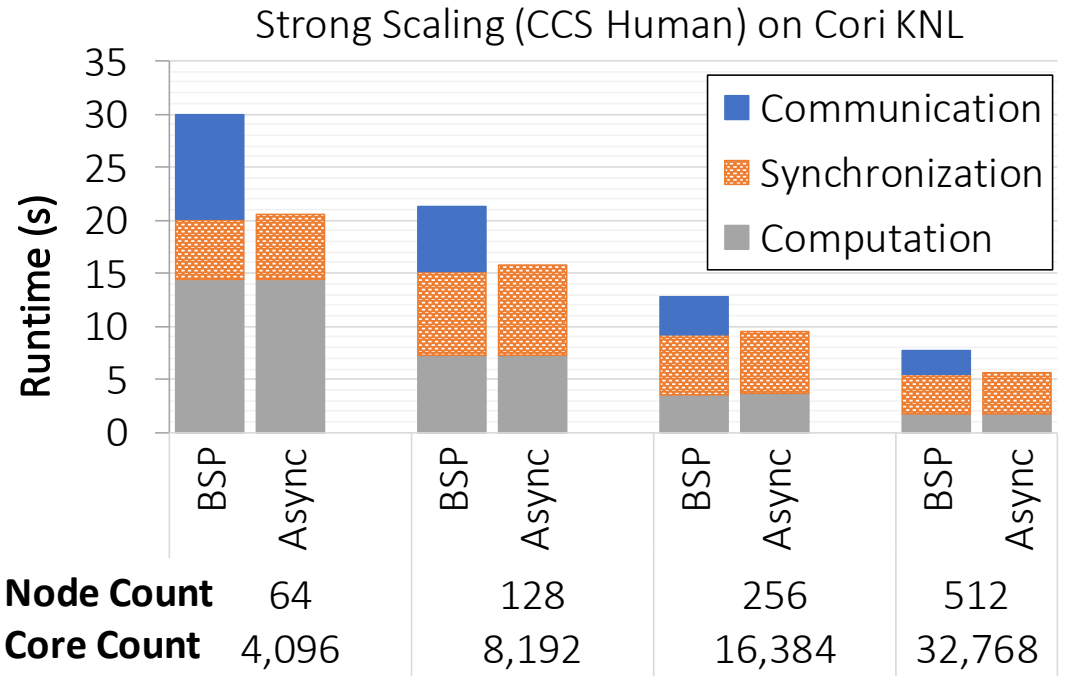


8-32 nodes, memory limits necessitate multiple BSP exchanges, increasing overall communication overhead



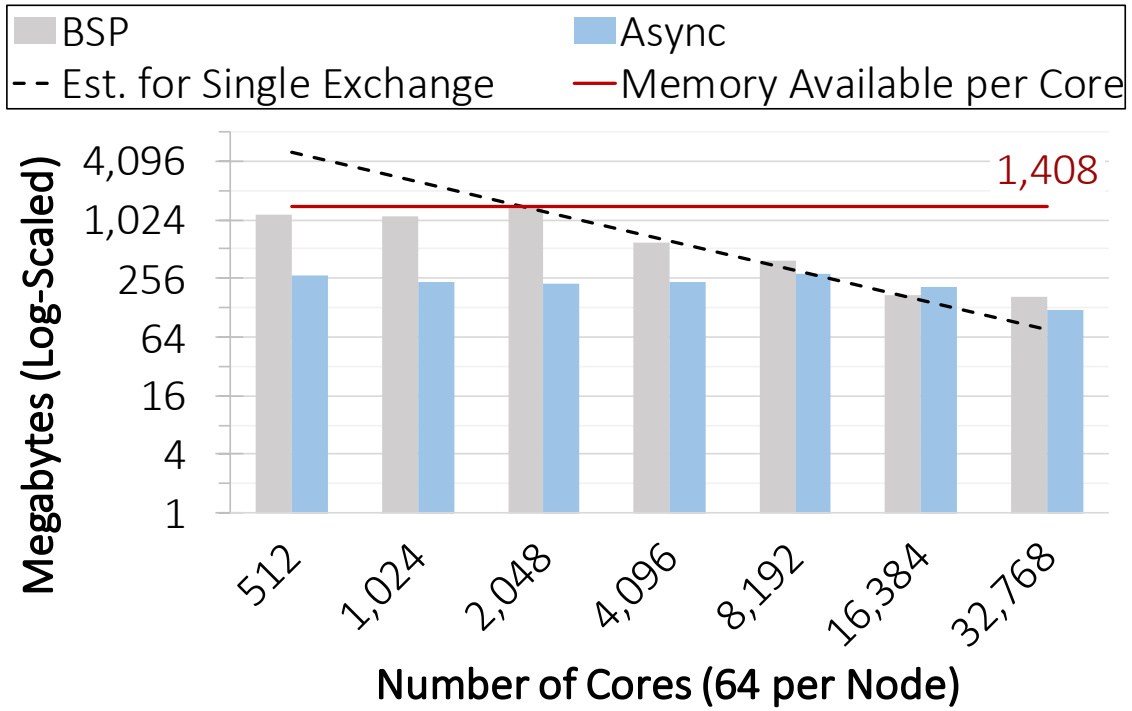


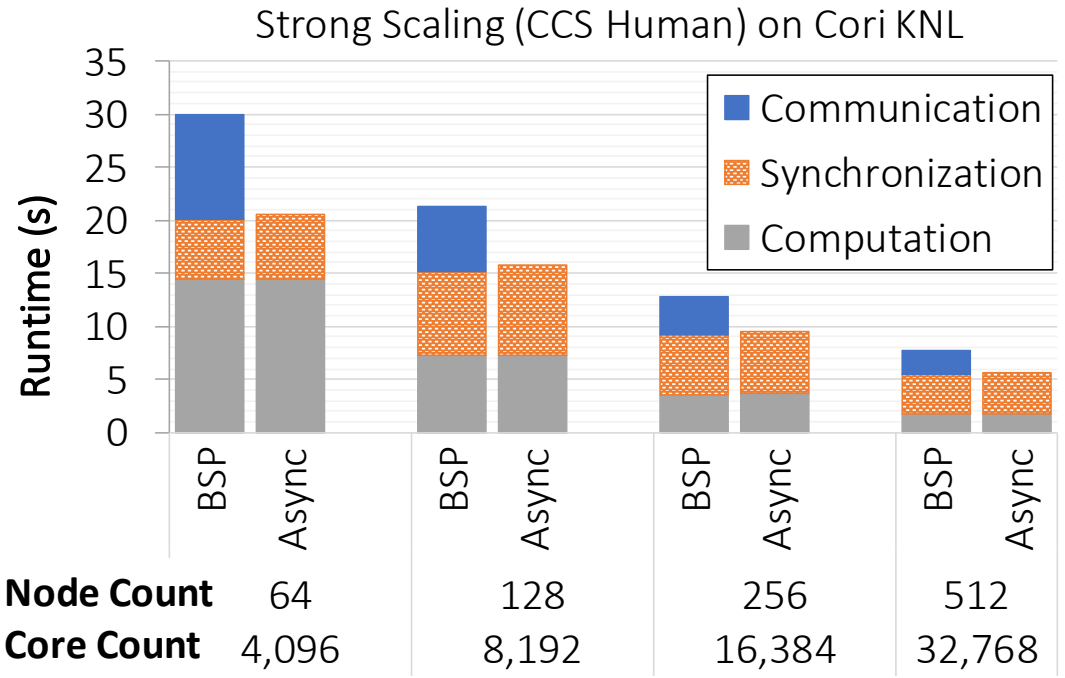
With sufficient memory for a single BSP exchange, 64-512 nodes, efficiency gap decreases to 4-13%



^ note, **y-axis s**cale adjustment of 6x

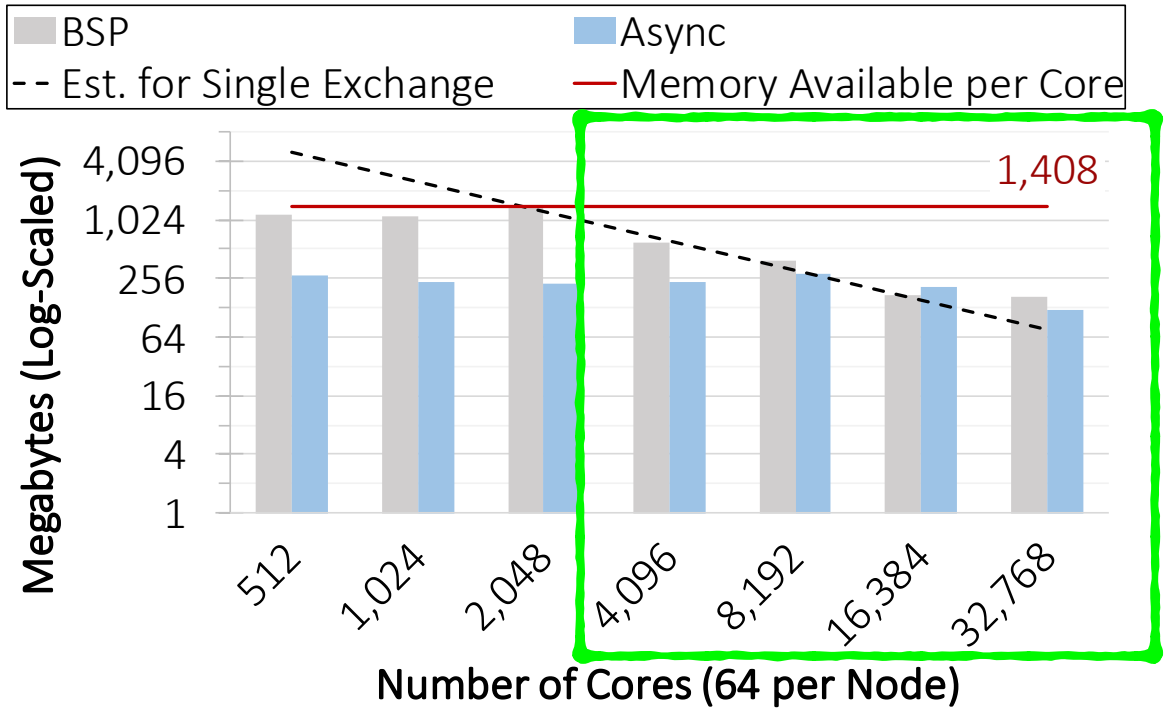
With sufficient memory for a single BSP exchange, 64-512 nodes, efficiency gap decreases to 4-13%

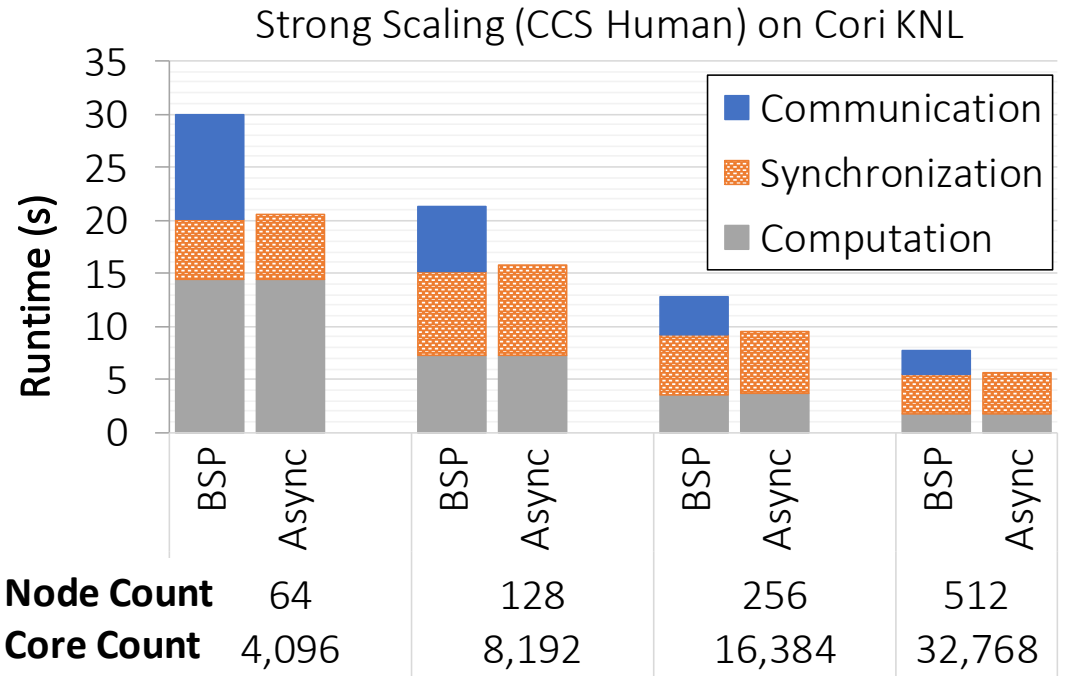




^ note, **y-axis s**cale adjustment of 6x

With sufficient memory for a single BSP exchange, 64-512 nodes, efficiency gap decreases to 4-13%



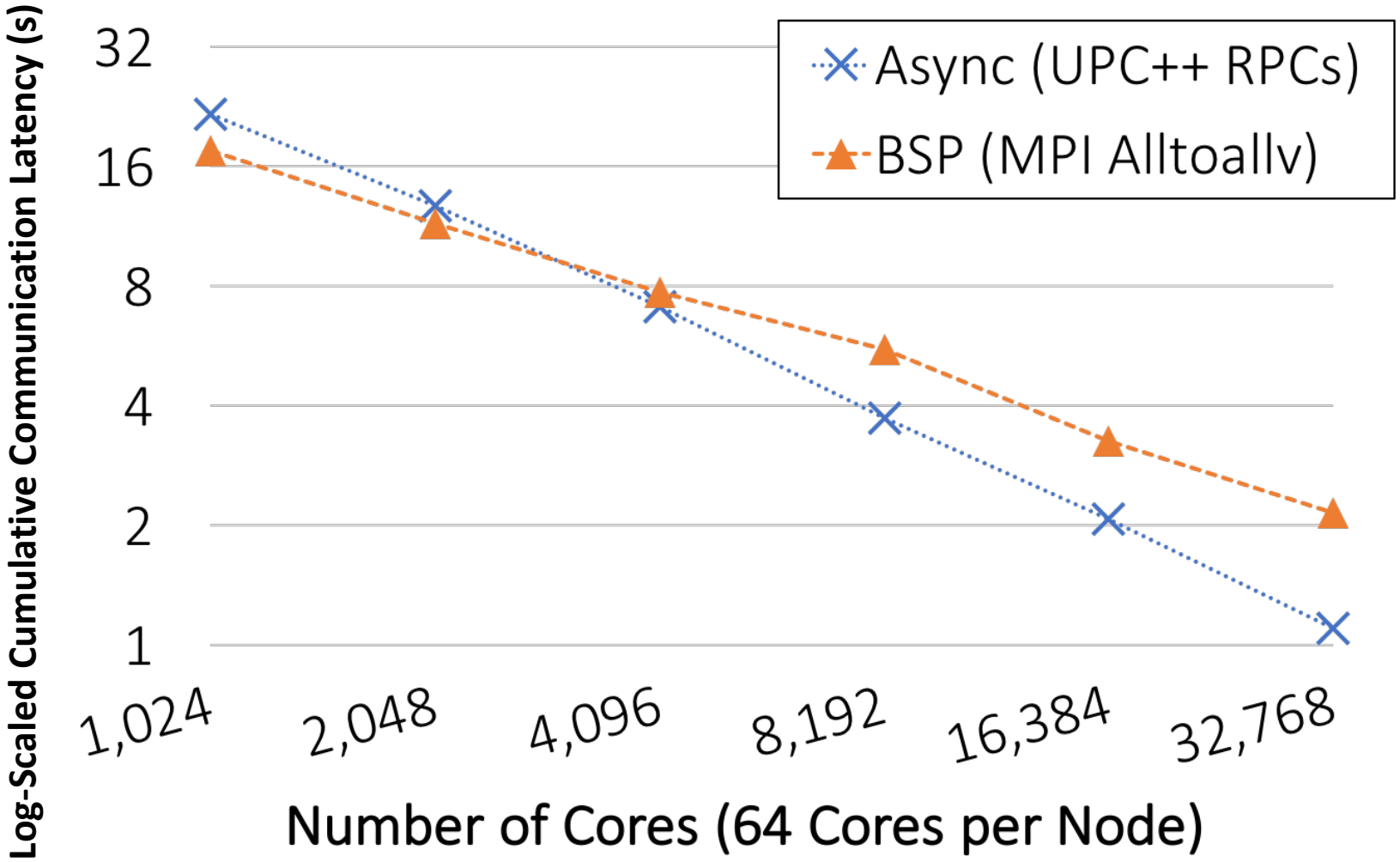


^ note, **y-axis s**cale adjustment of 6x

What are the relative unhidden communication costs?

What are the relative unhidden communication costs?

- Measured/validated using a mode that skips the computation
- Async sends many more messages with variable round-trip latency
- Number of messages scale inversely with the number of parallel processors
- In aggregate, Async communication-computation overlap balance out



What is expected across different architectures and applications? (Conclusions and Future Work)

What is expected across different architectures and applications? (Conclusions and Future Work)

- Cori's high-bandwidth low-latency interconnect supports both approaches well

What is expected across different architectures and applications? (Conclusions and Future Work)

- Cori's high-bandwidth low-latency interconnect supports both approaches well
  - with a high-bandwidth, **high-latency** interconnect, these results may very well flip

What is expected across different architectures and applications? (Conclusions and Future Work)

- Cori's high-bandwidth low-latency interconnect supports both approaches well
  - with a high-bandwidth, **high-latency** interconnect, these results may very well flip

What is expected across different architectures and applications? (Conclusions and Future Work)

- Cori's high-bandwidth low-latency interconnect supports both approaches well
  - with a high-bandwidth, **high-latency** interconnect, these results may very well flip

- Keys for effective communication-computation overlap for data-intensive Generalized N-Body problems like these

What is expected across different architectures and applications? (Conclusions and Future Work)

- Cori's high-bandwidth low-latency interconnect supports both approaches well
  - with a high-bandwidth, **high-latency** interconnect, these results may very well flip

- Keys for effective communication-computation overlap for data-intensive Generalized N-Body problems like these
  - **balance** of round-trip or one-sided **message latencies** to **pairwise/tuple-wise computations** on average

What is expected across different architectures and applications? (Conclusions and Future Work)

- Cori's high-bandwidth low-latency interconnect supports both approaches well
  - with a high-bandwidth, **high-latency** interconnect, these results may very well flip

- Keys for effective communication-computation overlap for data-intensive Generalized N-Body problems like these
  - **balance** of round-trip or one-sided **message latencies** to **pairwise/tuple-wise computations** on average
  - one example implication: optimizing the computation (pairwise alignment) in this case is only independent from communication optimization up to a point…

What is expected across different architectures and applications? (Conclusions and Future Work)

What is expected across different architectures and applications? (Conclusions and Future Work)

- Keys for bulk-synchronous approaches for data-intensive Generalized N-Body problems like these

What is expected across different architectures and applications? (Conclusions and Future Work)

- Keys for bulk-synchronous approaches for data-intensive Generalized N-Body problems like these
  - **bisection bandwidth** and **memory** enabling (or **limiting) message aggregation** for the many-to-many communication

What is expected across different architectures and applications? (Conclusions and Future Work)

- Keys for bulk-synchronous approaches for data-intensive Generalized N-Body problems like these
  - **bisection bandwidth** and **memory** enabling (or **limiting) message aggregation** for the many-to-many communication
  - expect that, optimization to the computation will lower the number of parallel processors at which performance crosses-over from being computation-bound to communication-bound for any given workload

What is expected across different architectures and applications? (Conclusions and Future Work)

- Keys for bulk-synchronous approaches for data-intensive Generalized N-Body problems like these
  - **bisection bandwidth** and **memory** enabling (or **limiting) message aggregation** for the many-to-many communication
  - expect that, optimization to the computation will lower the number of parallel processors at which performance crosses-over from being computation-bound to communication-bound for any given workload

What is expected across different architectures and applications? (Conclusions and Future Work)

- Keys for bulk-synchronous approaches for data-intensive Generalized N-Body problems like these
  - **bisection bandwidth** and **memory** enabling (or **limiting) message aggregation** for the many-to-many communication
  - expect that, optimization to the computation will lower the number of parallel processors at which performance crosses-over from being computation-bound to communication-bound for any given workload

- Other opportunities and optimization suggestions are highlighted in the full text

What is expected across different architectures and applications? (Conclusions and Future Work)

- Keys for bulk-synchronous approaches for data-intensive Generalized N-Body problems like these
  - **bisection bandwidth** and **memory** enabling (or **limiting) message aggregation** for the many-to-many communication
  - expect that, optimization to the computation will lower the number of parallel processors at which performance crosses-over from being computation-bound to communication-bound for any given workload

- Other opportunities and optimization suggestions are highlighted in the full text
  - also more depth on both the computational and communication load imbalance…

Final Remarks

- The code is available on SourceForge (DiBELLA)
- for use with general DNA inputs for *read to read overlap and alignment*
  - and for other bioinformatics problems, e.g. proteomics, with reasonable refactoring effort
- for performance-focused studies and benchmarking as demonstrated

Acknowledgements

This research was supported in part by the **Exascale Computing Project** (17-SC-20-SC), a collaborative effort of the **U.S. Department of Energy Office of Science** and the **National Nuclear Security Administration** and by the **National Science Foundation** as part of the SPX program under Award number 1823034. This research used resources of the **National Energy Research Scientific Computing Center (NERSC)**, a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231.

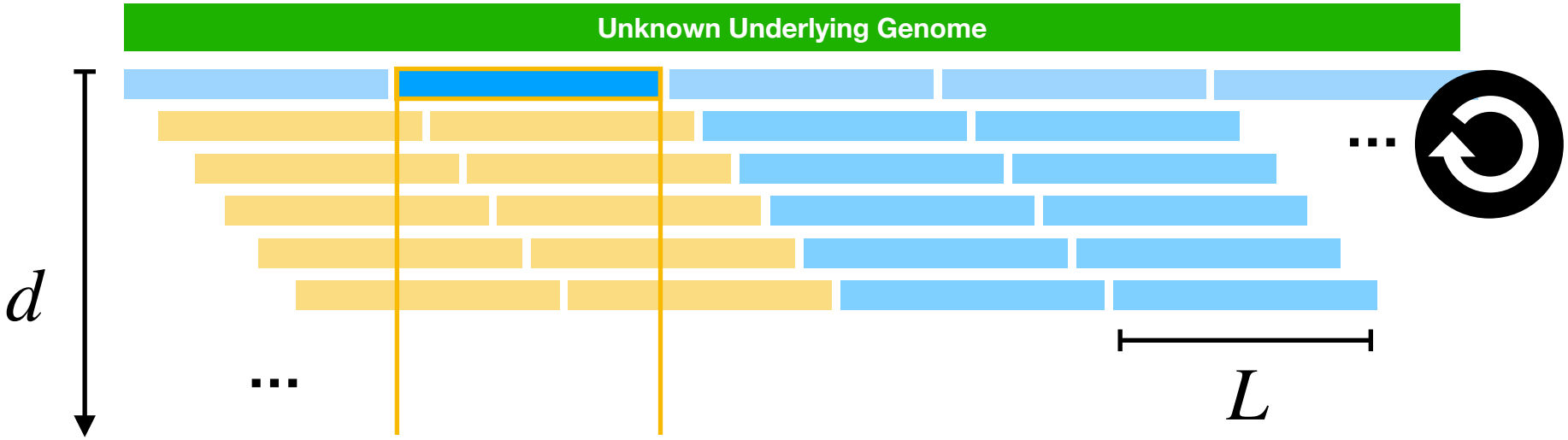
We thank our **5 anonymous reviewers** for their **invaluable feedback**!

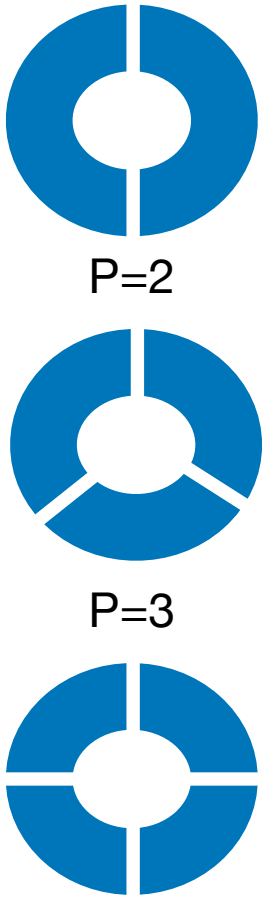
Questions

Appendix

Lemma: (in general) any partitioning of the reads will cut hyperedges due to the underlying sequencing methodology

Perfect world: no errors, uniform read lengths, perfectly uniform coverage, no *repeats*, "circular" genome





- Each **read** overlaps with **2(*d*-1) other reads**
- Partitioning cannot cut fewer than (*P* x *d*) hyperedges, P > 1

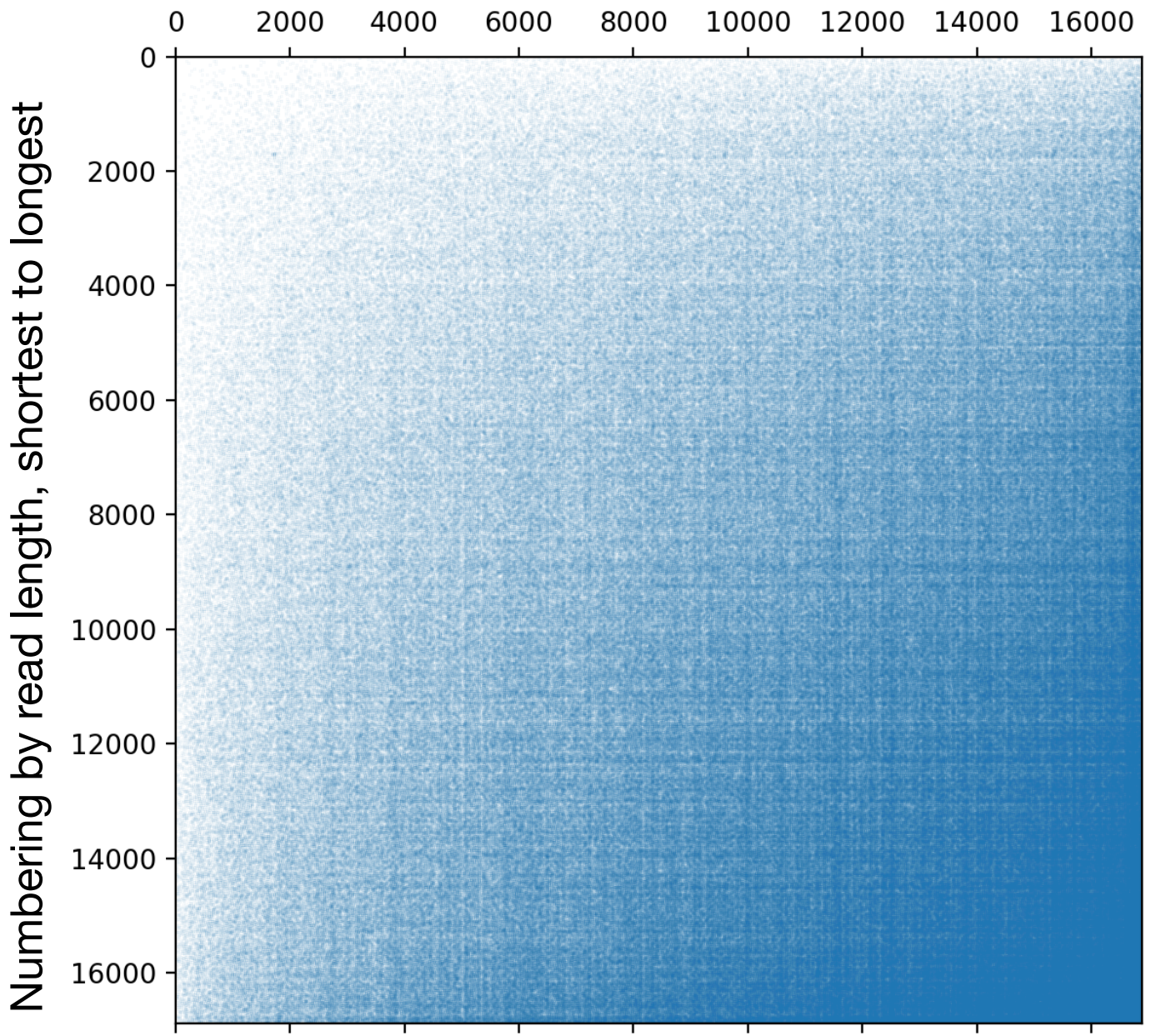
Reality: the hypergraphs are sparse

*** we expect this from k-mer filtering among other reasons**

Reality: the hypergraphs are sparse

*** we expect this from k-mer filtering among other reasons**

Plotted as Read by Read Matrix

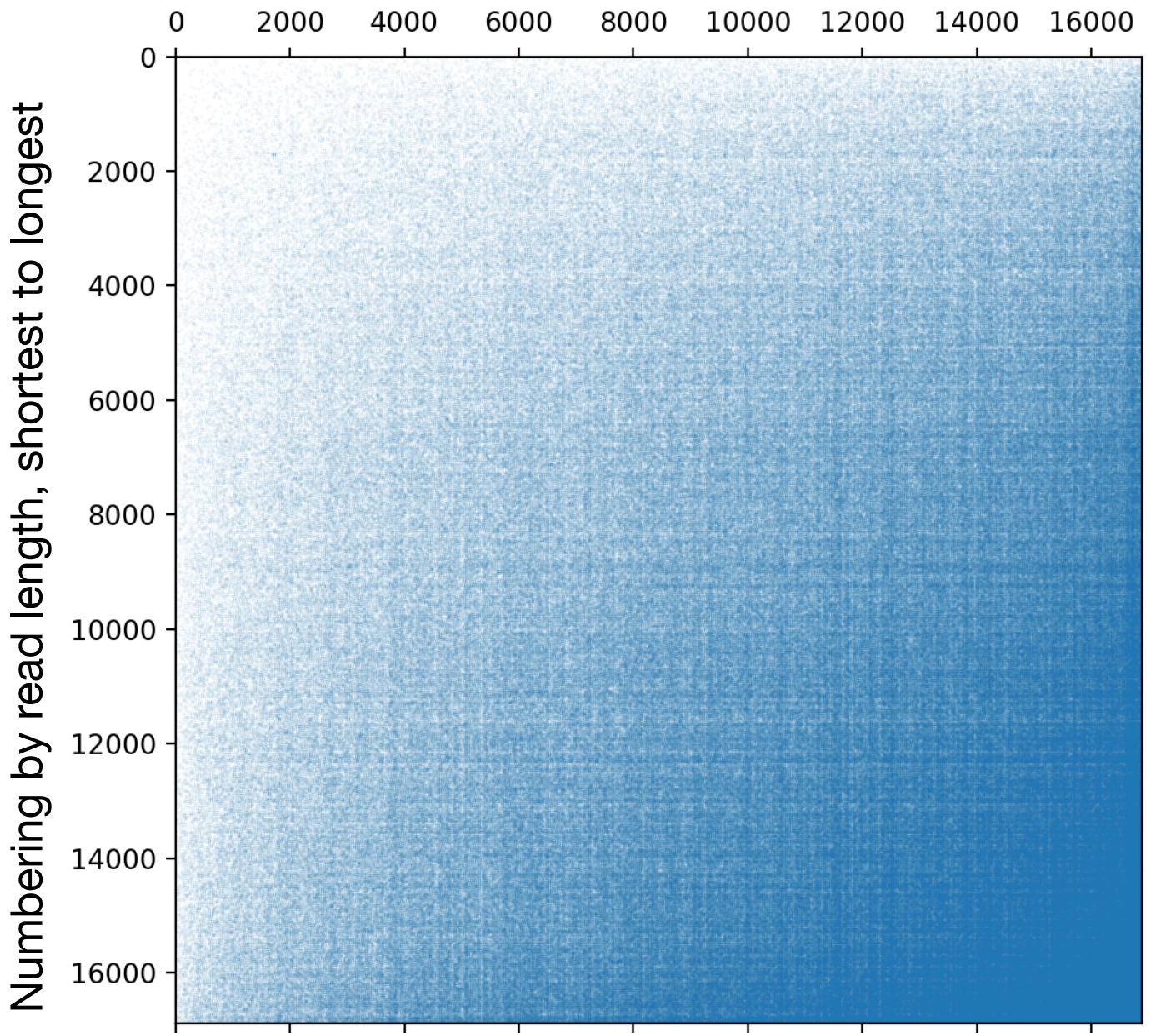


Reality: the hypergraphs are sparse

*** we expect this from k-mer filtering among other reasons**

Looks dense but is very sparse!

Plotted as Read by Read Matrix



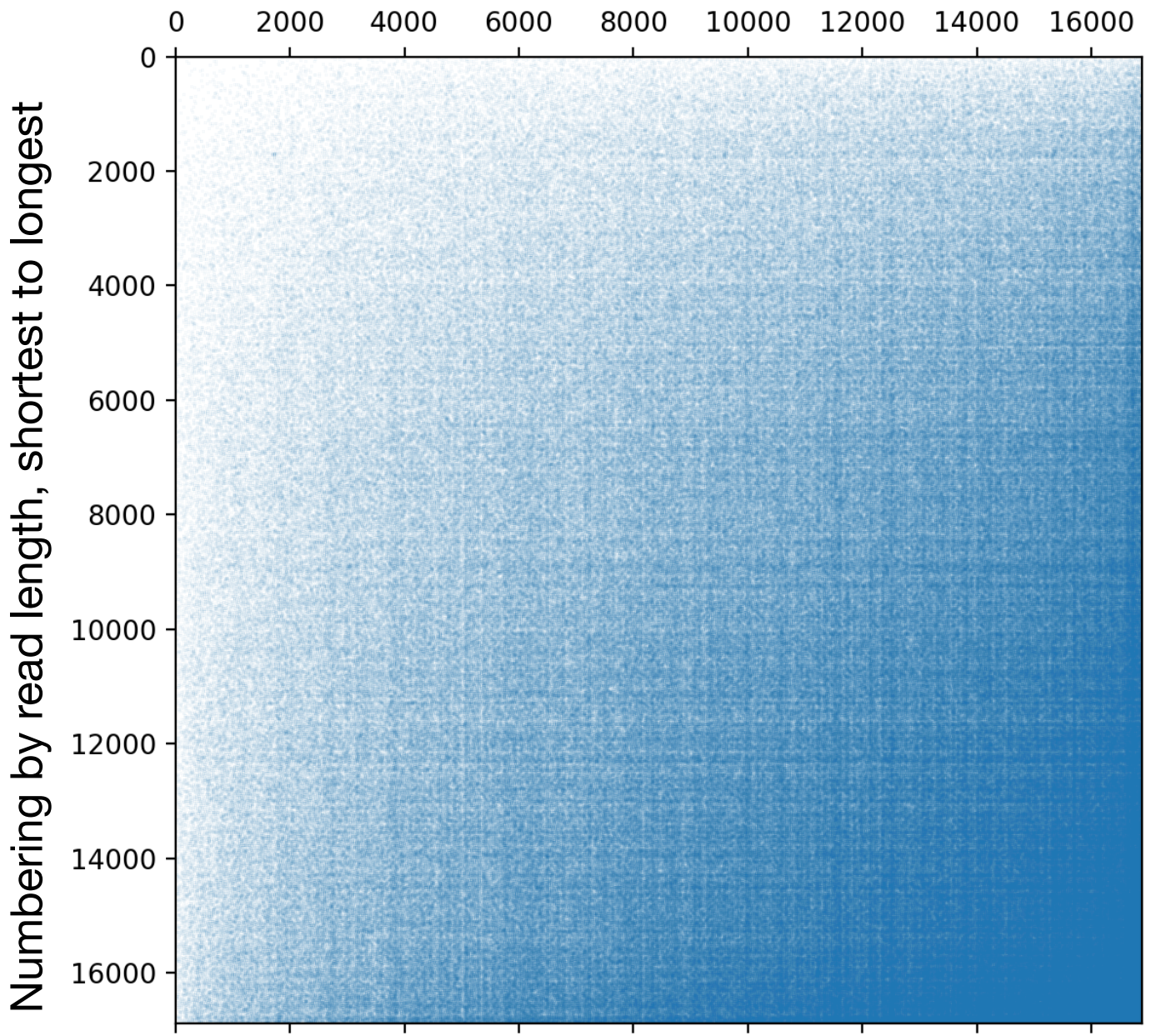
Reality: the hypergraphs are sparse

*** we expect this from k-mer filtering among other reasons**

Looks dense but is very sparse!

- nnz/(# reads ^{2})= ~0.008

Plotted as Read by Read Matrix



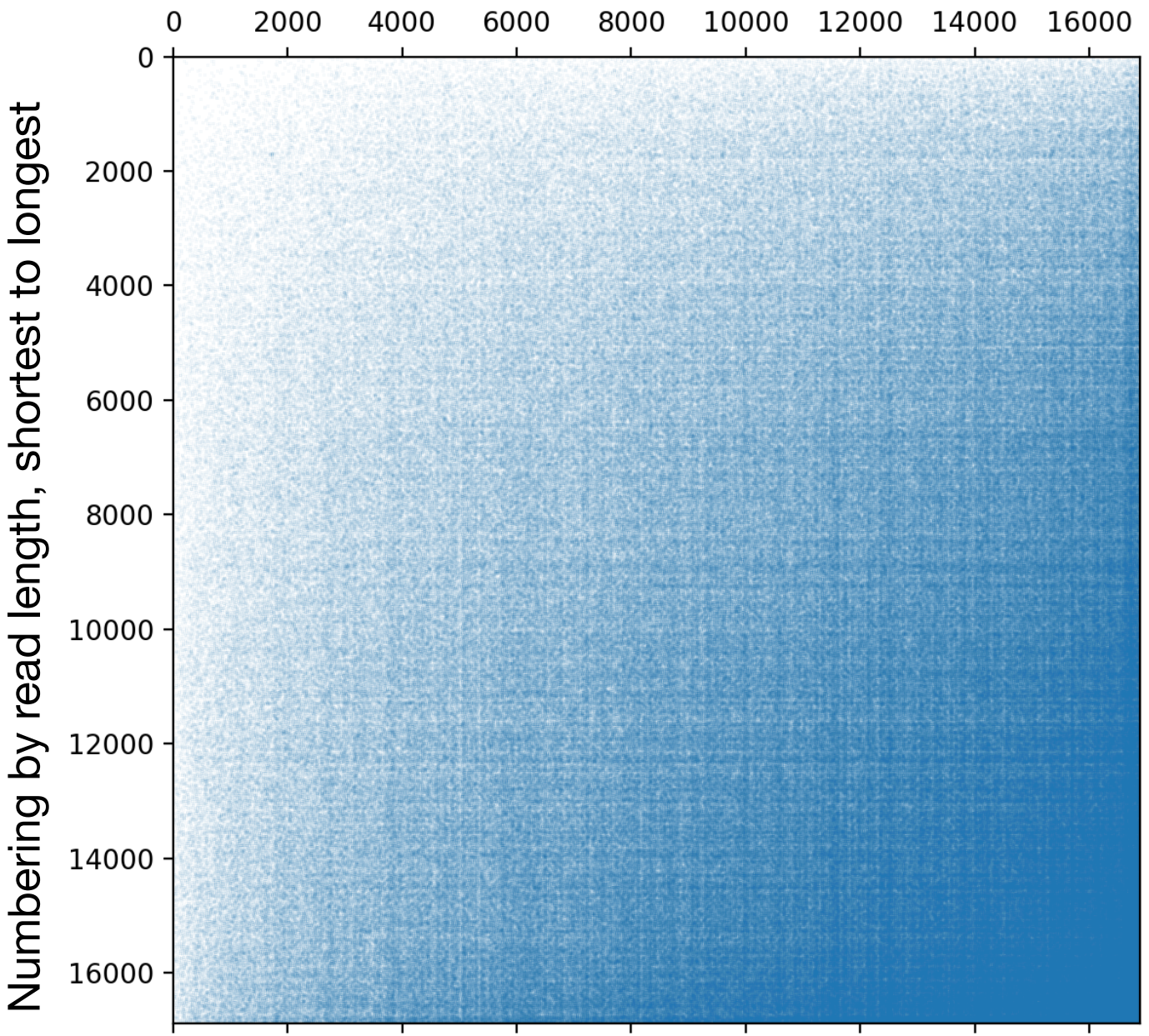
Reality: the hypergraphs are sparse

*** we expect this from k-mer filtering among other reasons**

Looks dense but is very sparse!

- nnz/(# reads ^{2})= ~0.008
- ~16,890 reads

Plotted as Read by Read Matrix



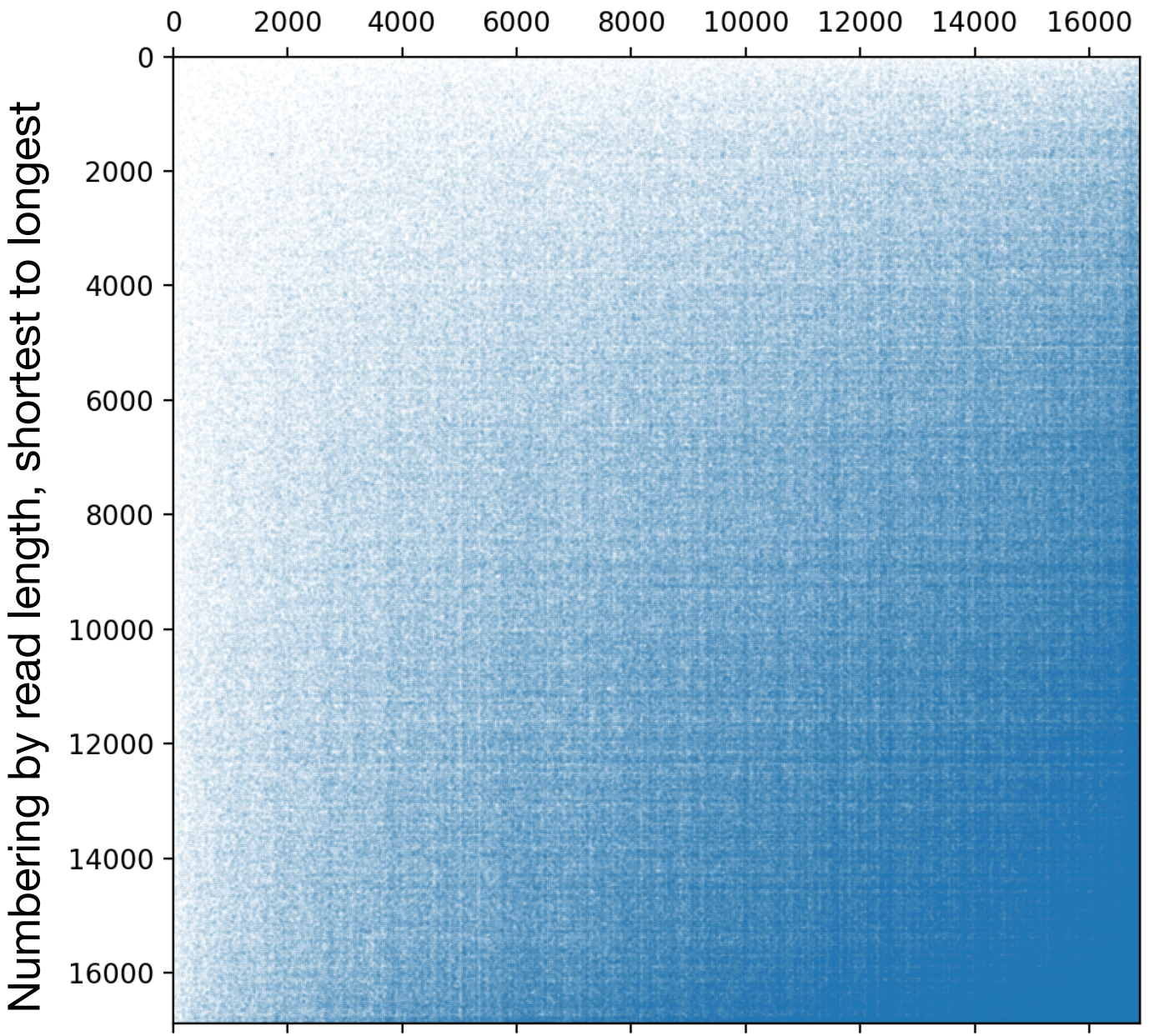
Reality: the hypergraphs are sparse

*** we expect this from k-mer filtering among other reasons**

Looks dense but is very sparse!

- nnz/(# reads ^{2})= ~0.008
- ~16,890 reads
- ~2.2 million overlaps

Plotted as Read by Read Matrix



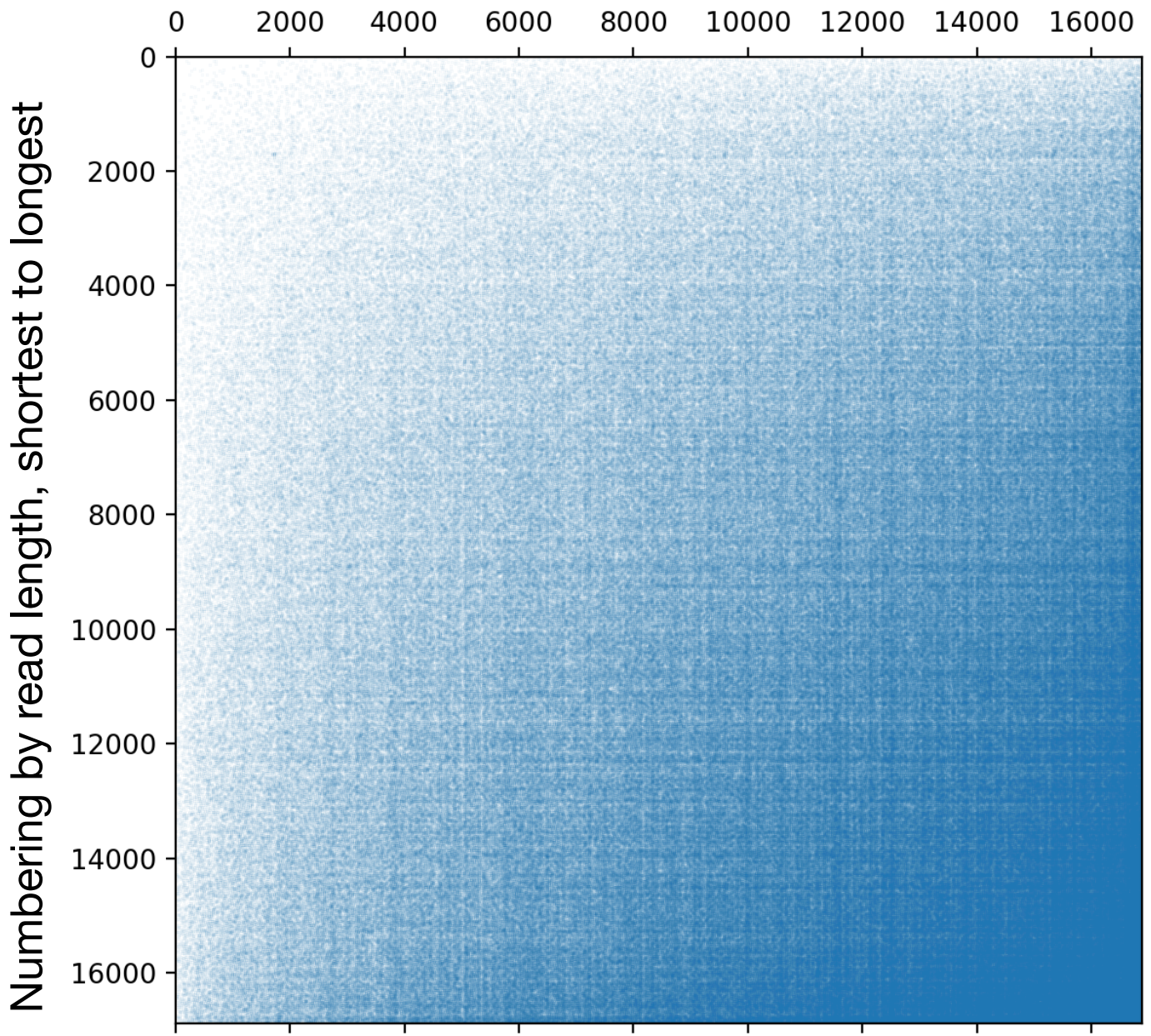
Reality: the hypergraphs are sparse

*** we expect this from k-mer filtering among other reasons**

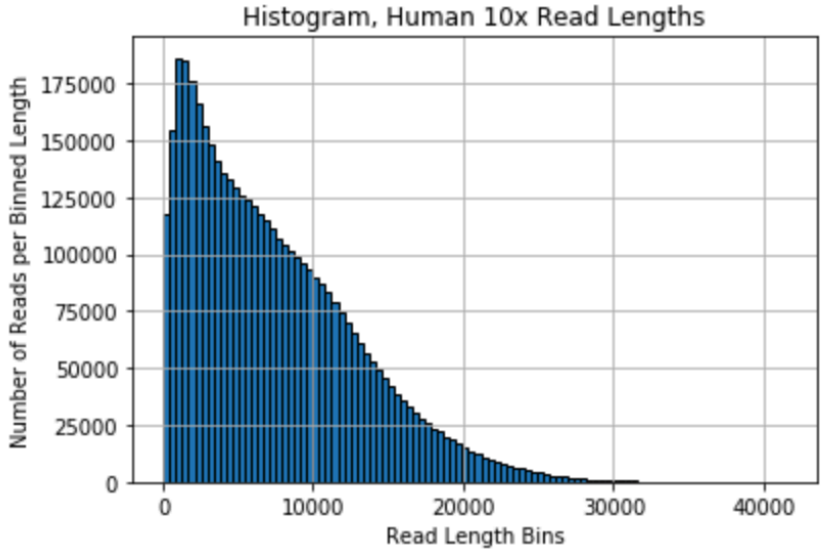
Looks dense but is very sparse!

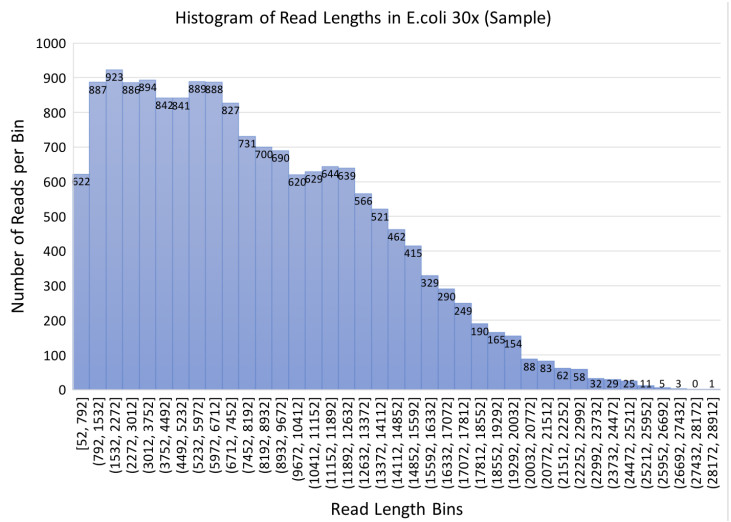
- nnz/(# reads ^{2})= ~0.008
- ~16,890 reads
- ~2.2 million overlaps
- making the non-zeros visible in the plot, makes the matrix look dense

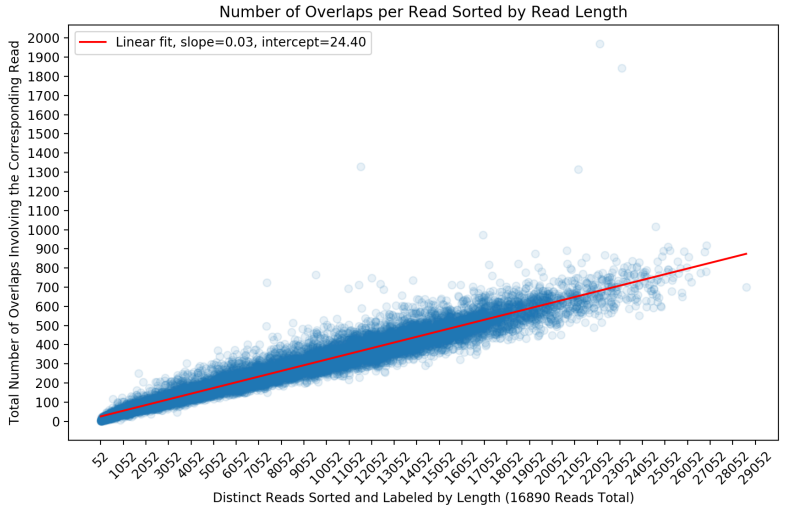
Plotted as Read by Read Matrix



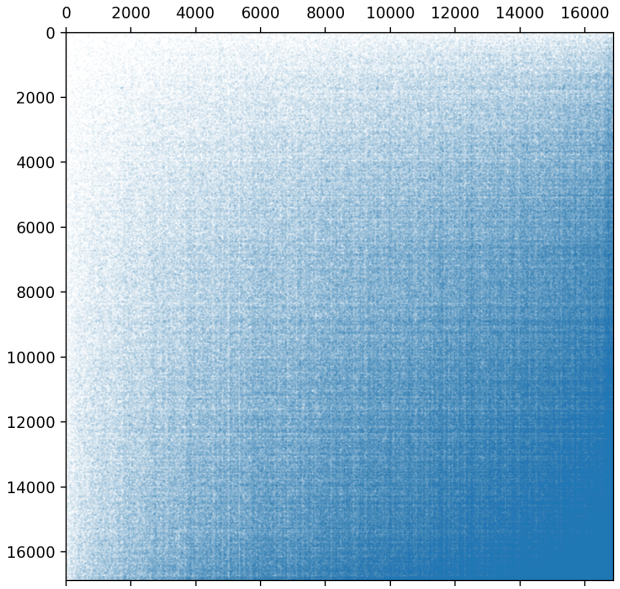
Large sparse unstructured hypergraph

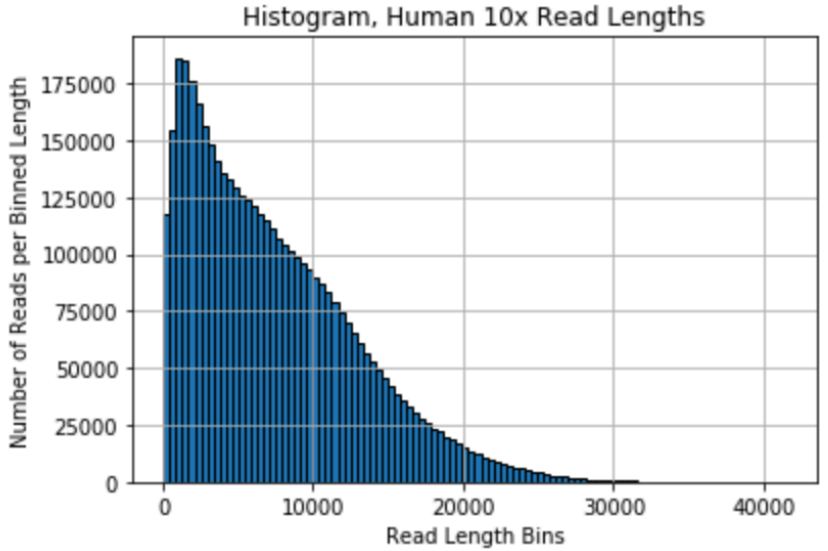


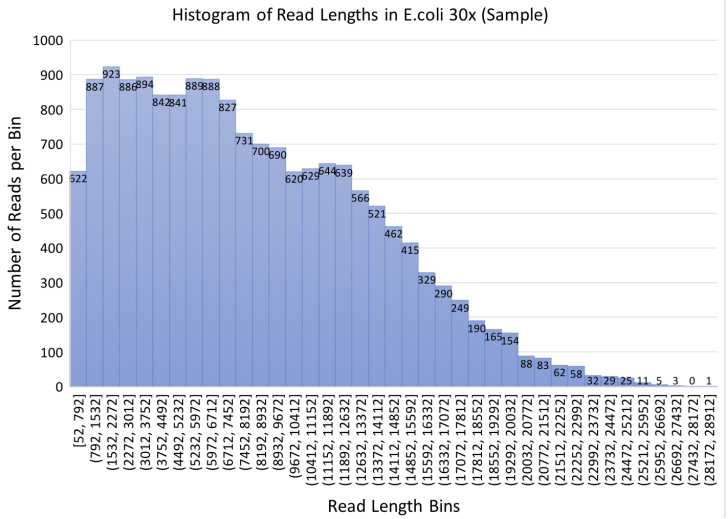


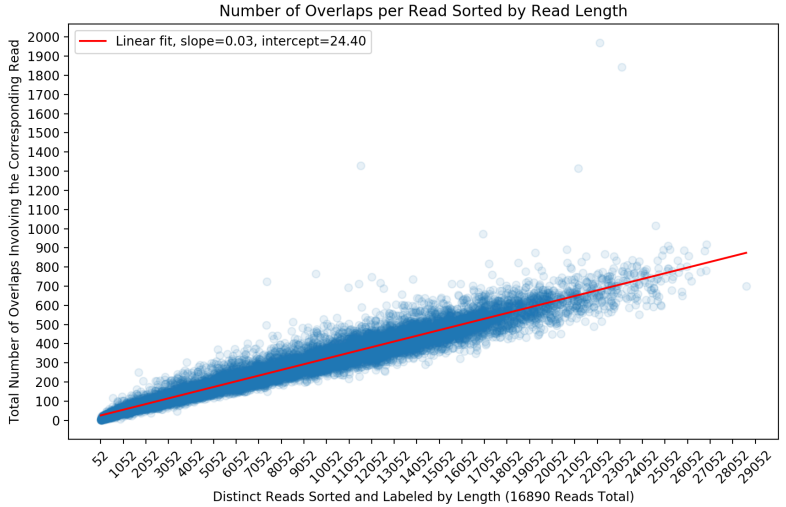


Large sparse unstructured hypergraph









INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

acm In-Cooperation sighpc