Using MPI One-sided Communication to Accelerate Bioinformatics Applications

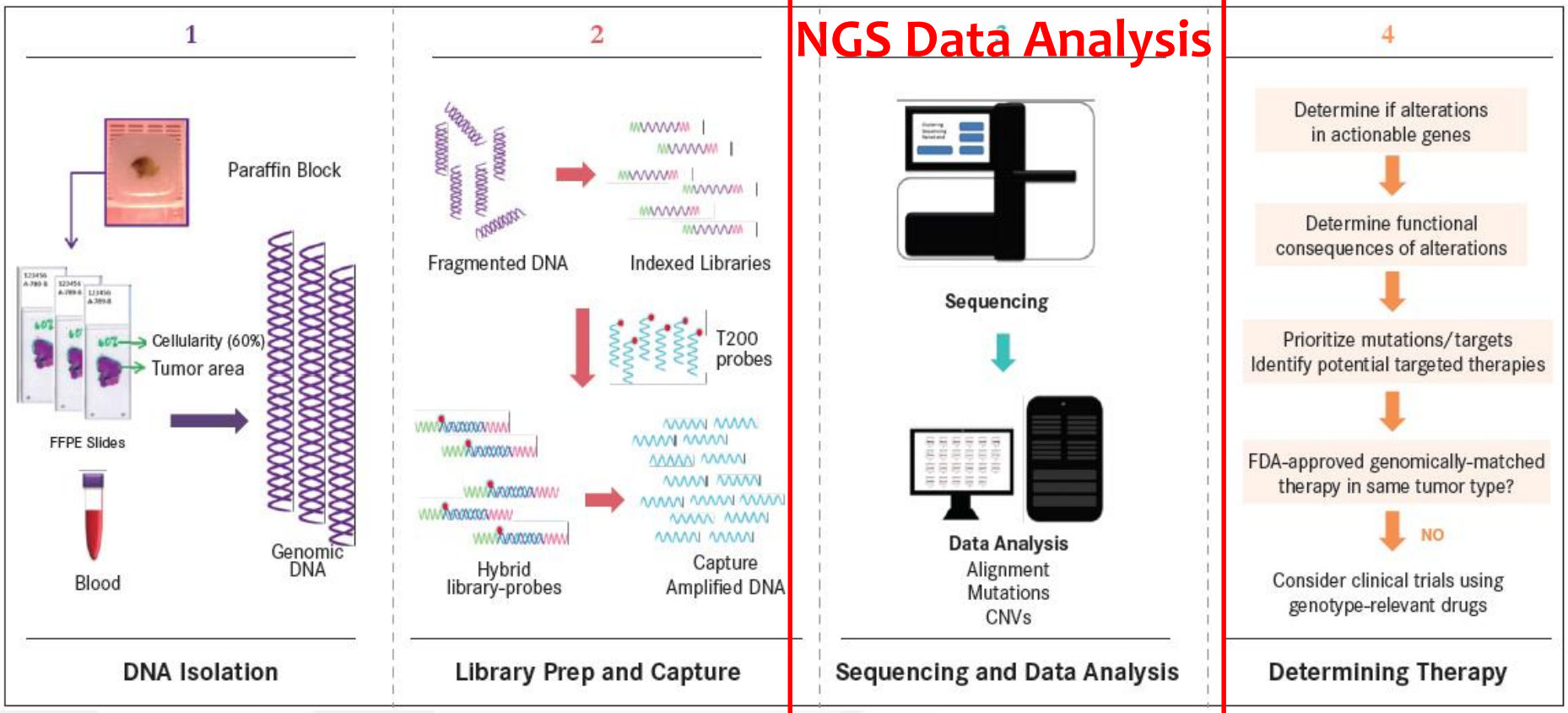
Hao Wang (hwang121@vt.edu)

Department of Computer Science, Virginia Tech

VirginiaTech
Invent the Future

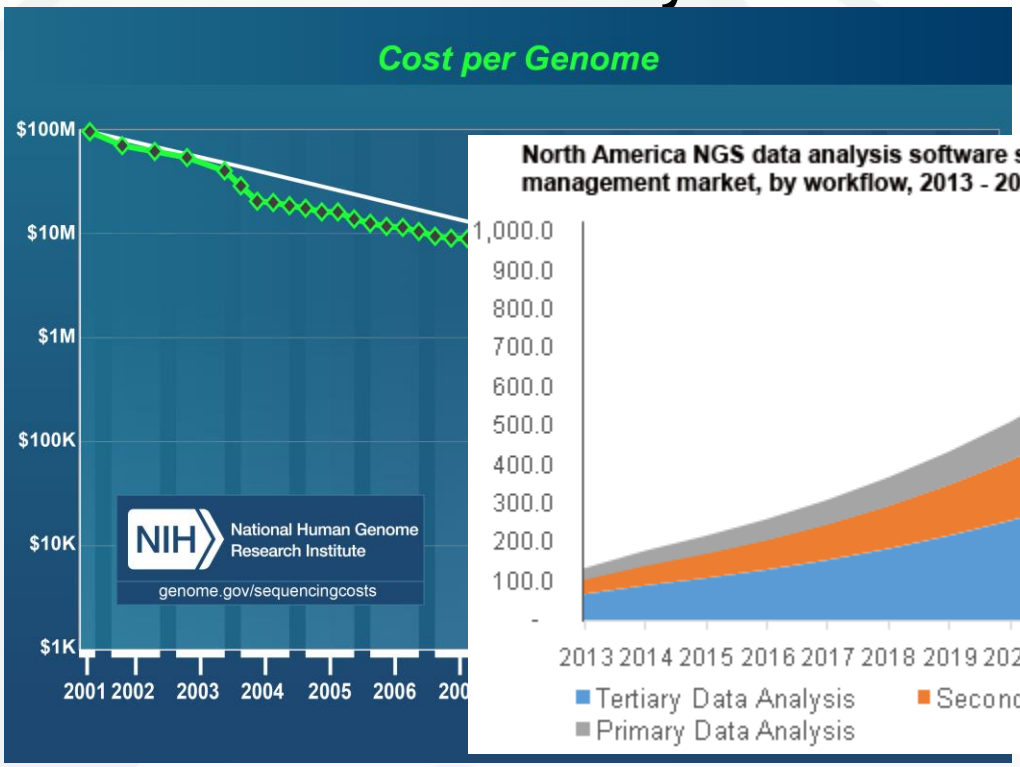
SyNeRG
synergy.cs.vt.edu

Next-Generation Sequencing (NGS) Data Analysis



- DNA is isolation from normal tissue and blood
- DNA is fragmented and the captured DNA is washed and amplified
- DNA is sequenced and analyzed
- DNA is used for clinical trials, e.g., disease detection, personalized medicine, etc.

NGS Data Analysis



Market Research Scope

Attribute	Details
Base year used for market estimation	2015
Historic Analysis	Actual data from 2013 to 2015
Forecast	2016 to 2024

- Next Generation Sequencing (NGS) has significantly reduced cost per genome; and data analysis (instead of sequencing) is becoming the bottleneck
- NGS data analysis market is boosting and predicted to exceed 1 Billion in 2024

- NIH, "DNA Sequencing Costs: Data", https://www.genome.gov/27541954/dna-sequencing-costs-data/
- Grand View Research, "NGS Data Analysis Market Analysis 2024", http://www.grandviewresearch.com/industry-analysis/next-generation-sequencing-ngs-data-analysis-market

Irregular NGS Data Analysis Applications

- NGS applications can be characterized by
  - Irregular memory accesses
  - Irregular control flows
  - Irregular communication patterns

- Many such applications exhibit irregularities
  - Basic Local Alignment Search Tool (**BLAST**) for sequence search
    - Heuristic algorithms
  - **BWA, Bowtie1/2,** and **SOAPaligner** for short read mapping
    - Compressed data structures

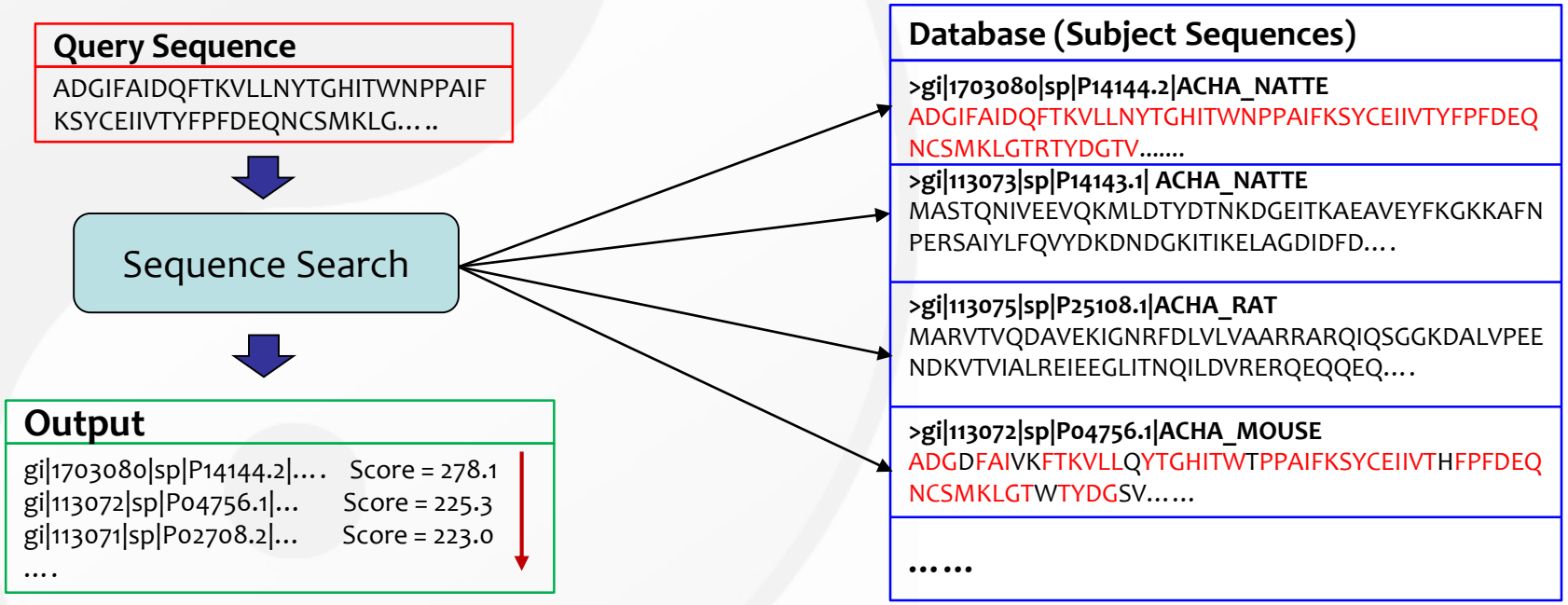
These applications have irregular communication patterns!

Outline

- Background
- **Sequence Search**
- Using one-sided communications for sequence search
- Evaluation (early stage)
- Summary and Future Work

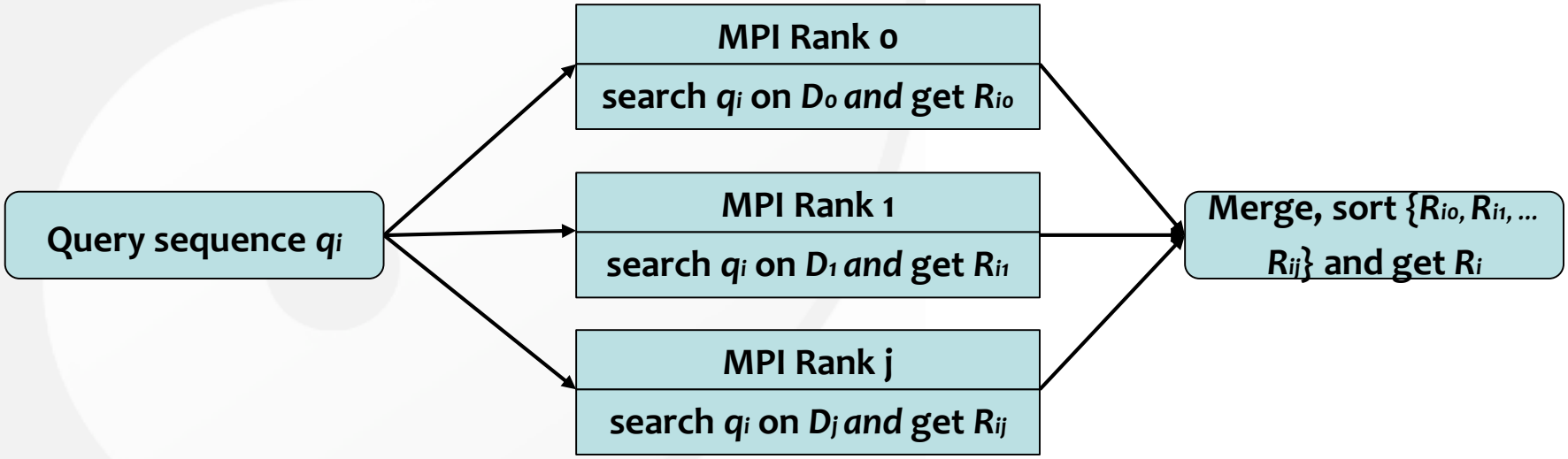
Sequence Search

- Search for similarities between a query sequence and database sequences (i.e., subject sequences)



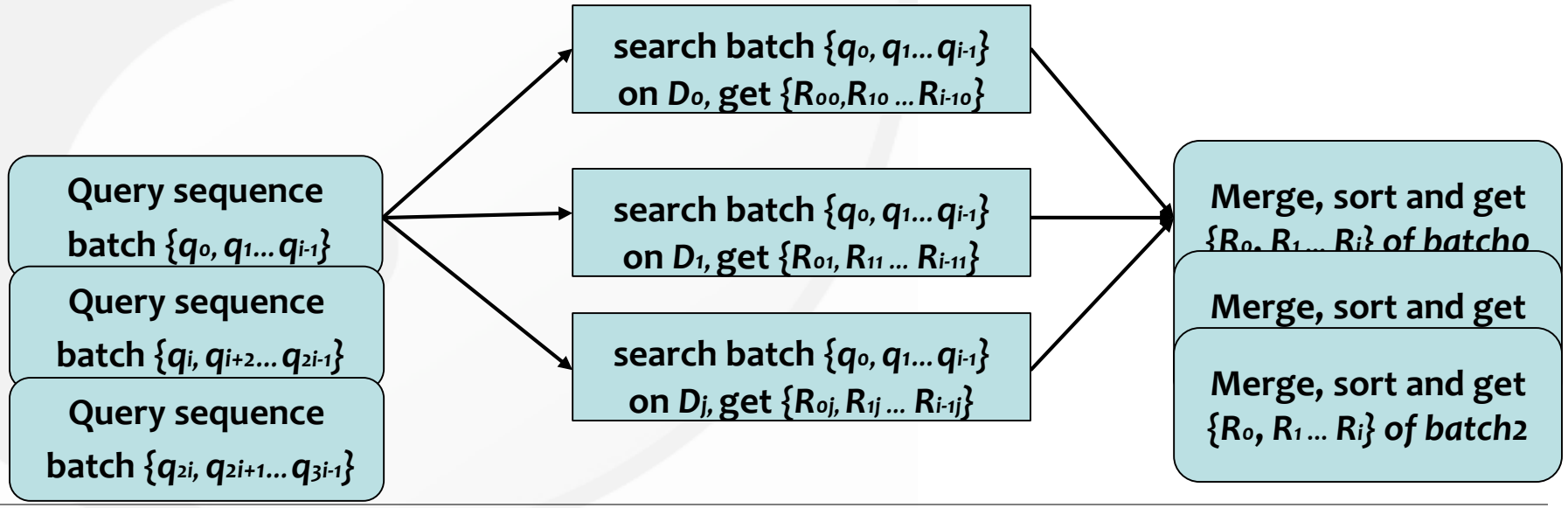
MPI Implementation

- Inter-node parallel implementation using MPI
  - Partition the database ***D*** into ***j*** subsets D_0 D_1 …… D_j
  - For a query sequence q_i, search q_i on each database subset D_j in parallel and get local search result R_{ij}
  - Merge and sort all local search result R_{i0} to R_{ij} and get the final result R_i

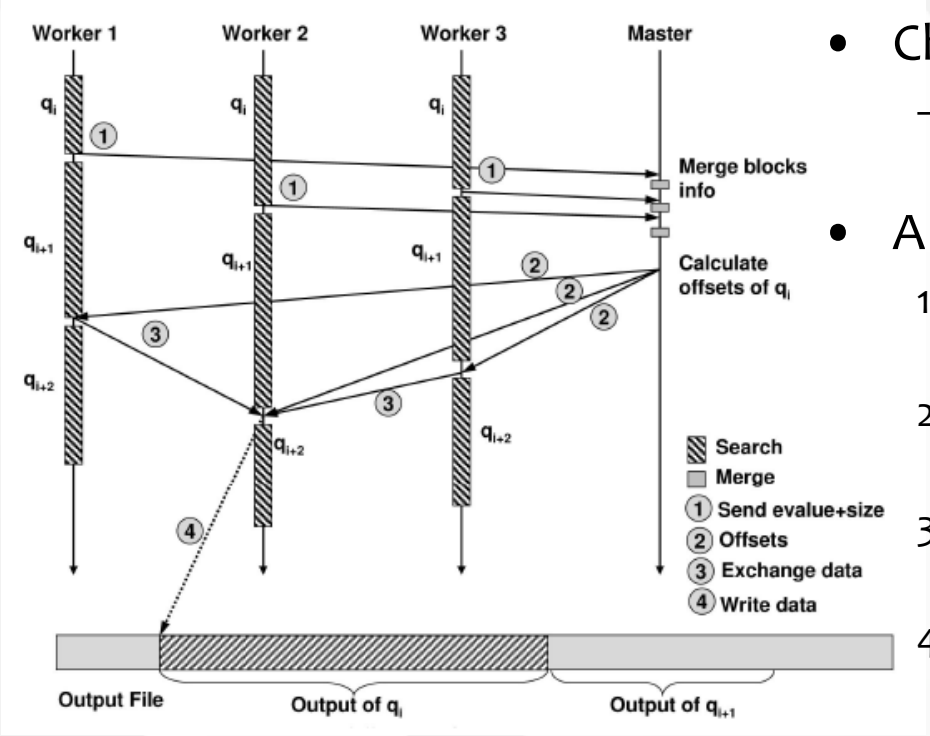


MPI Implementation

- Inter-node parallel implementation using MPI
  - Partition the database ***D*** into ***j*** subsets D_0 D_1 …… D_j
  - For a query sequence q_i, search q_i on each database subset D_j in parallel and get local search result R_{ij}
  - Merge and sort all local search result R_{i0} to R_{ij} and get the final result R_i



mpiBLAST Implementations



- Characteristics
  - Both computation time and data size of compute nodes are highly diverse
- A dedicated MPI process as the master
  1. All workers send meta data, i.e., query id, search score, and data size, to the master
  2. The master merges and sorts meta data, and selects a worker for IO and notifies all workers
  3. All workers send local selected results to the IO worker
  4. The IO worker finally writes data to disk

H. Lin, et al. "Coordinating computation and i/o in massively parallel sequence search." *Parallel and Distributed Systems, IEEE Transactions on* 22.4 (2011): 529-543.

Why Redesign Sequence Search

- Bottlenecks in previous sequence search tools
  - Local search
  - Disk IO

- New tendencies of sequence search
  - Local search is much faster right now, e.g., DIAMOND[1]
  - Sequence search has become a stage of NGS work flow, and search results are resided in memory for reuse[2]

Data communication is becoming a new performance bottleneck!

1. B. Buchfink, Xie C., D. Huson, "Fast and sensitive protein alignment using DIAMOND", Nature Methods 12, 59-60 (2015).
2. Genome Analysis Toolkit, https://software.broadinstitute.org/gatk/

Outline

- Background
- Sequence Search
- **Using one-sided communications for sequence search**
- Evaluation (early stage)
- Summary and Future Work

Using MPI One-sided for Sequence Search

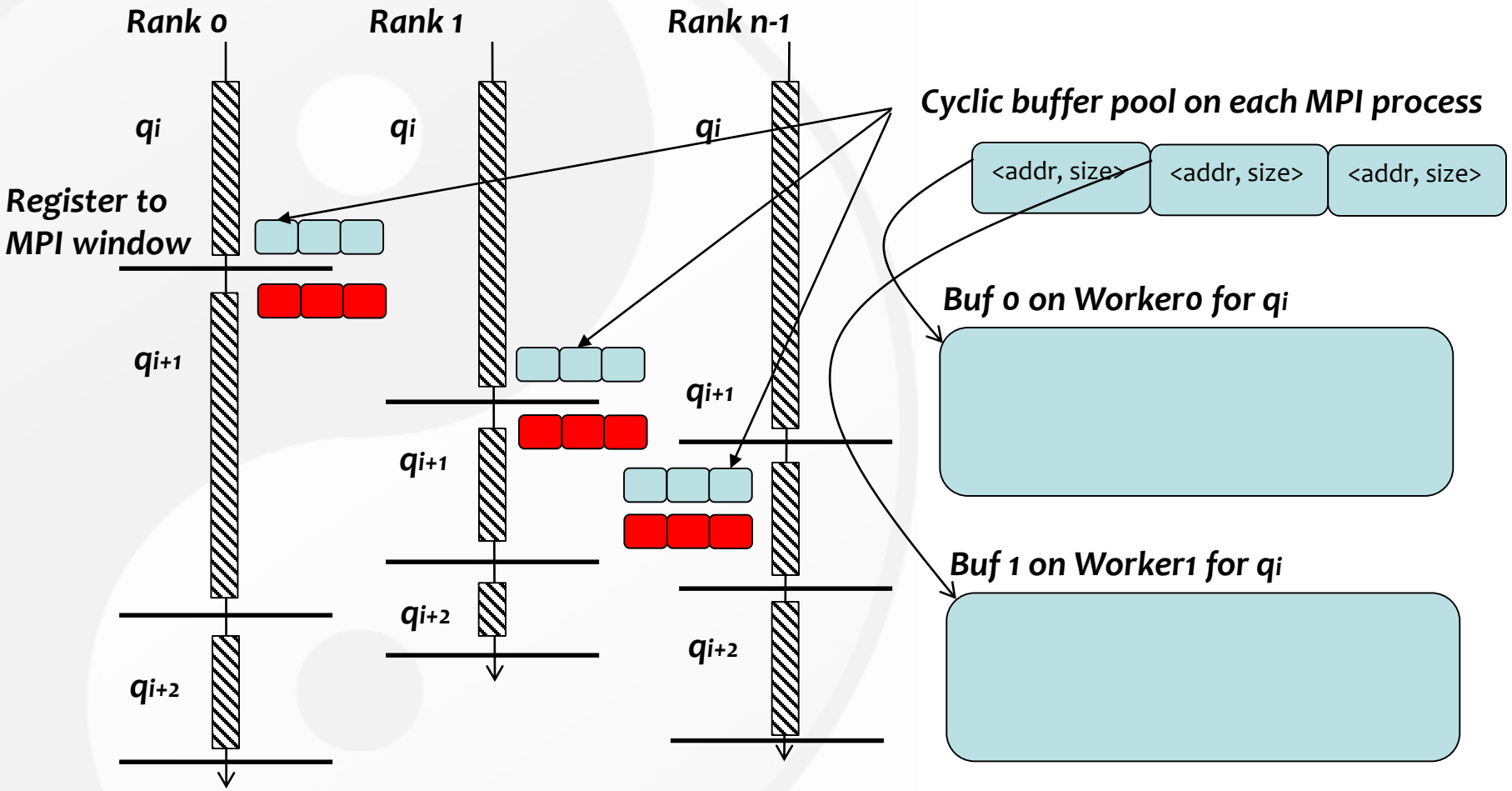
- Benefits of using MPI one-sided communication
  - More economically express irregular communication pattern
  - More efficiently overlap communication and computation
  - Bypass tag matching in two-sided communication

- Basic ideas
  - Use MPI one-sided communications (put and get) to overlap communication and computation
  - Don’t need a dedicated MPI process as the master to coordinate disk IO

Challenges

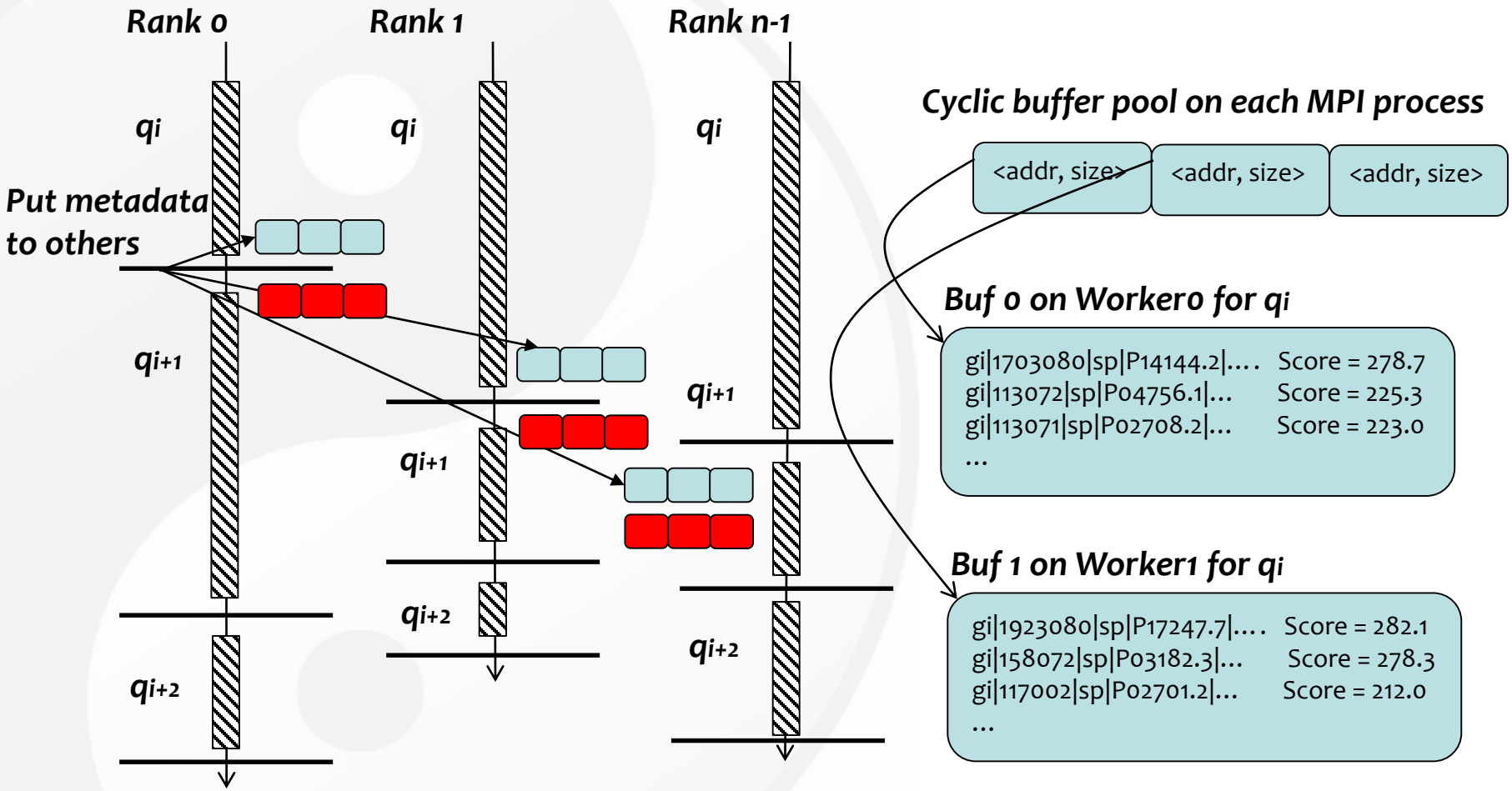
- Challenges
  - Three one-sided synchronization modes: Fence (active target), Post-Start-Complete-Wait (active target), Lock/Unlock (passive target). Which is better?

MPI Windows for Meta Data and Local Results



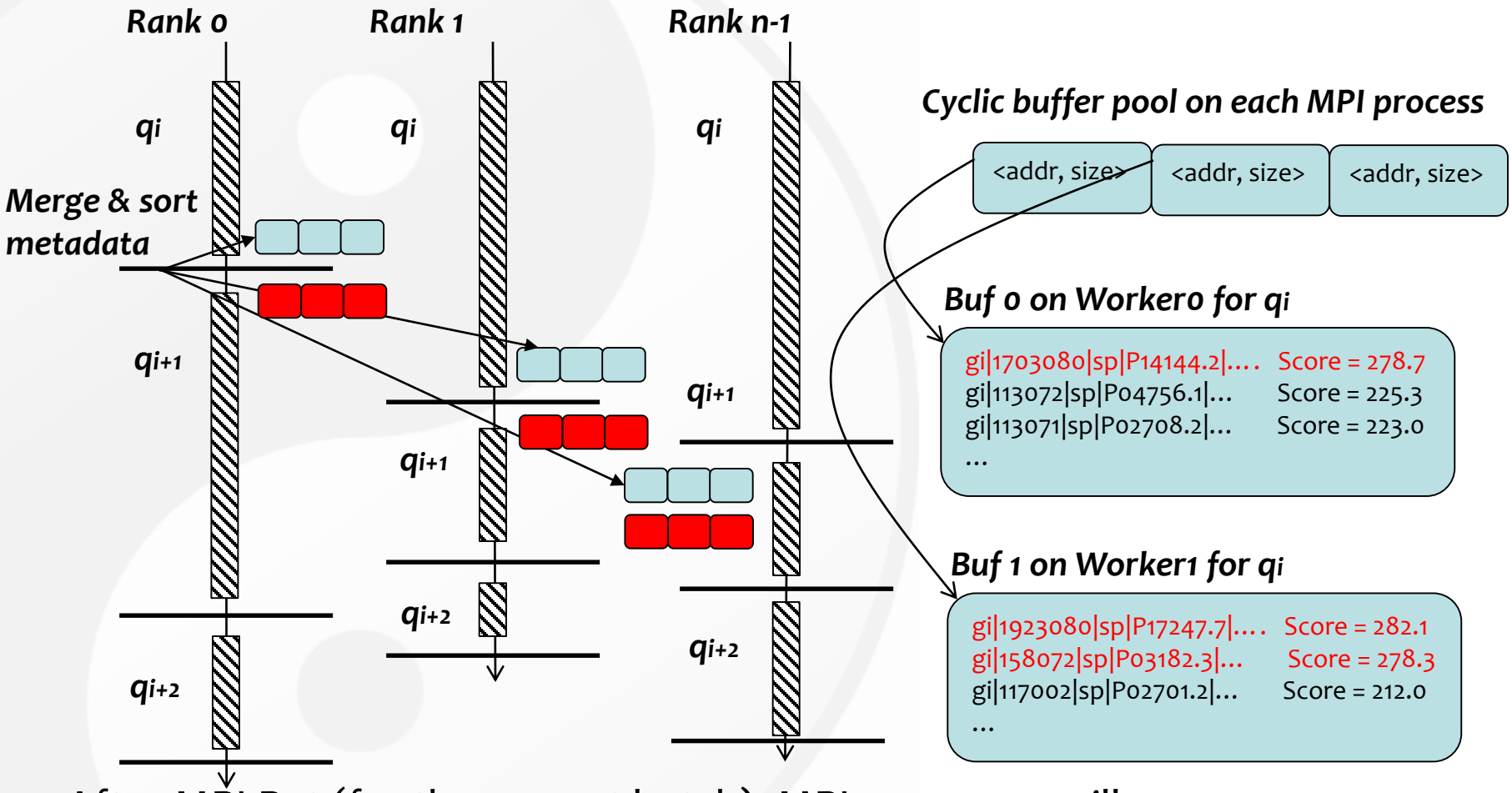
- Each MPI process registers two types of cyclic buffers to MPI window, for meta data and local search results respectively

Use MPI Put to Write Meta Data



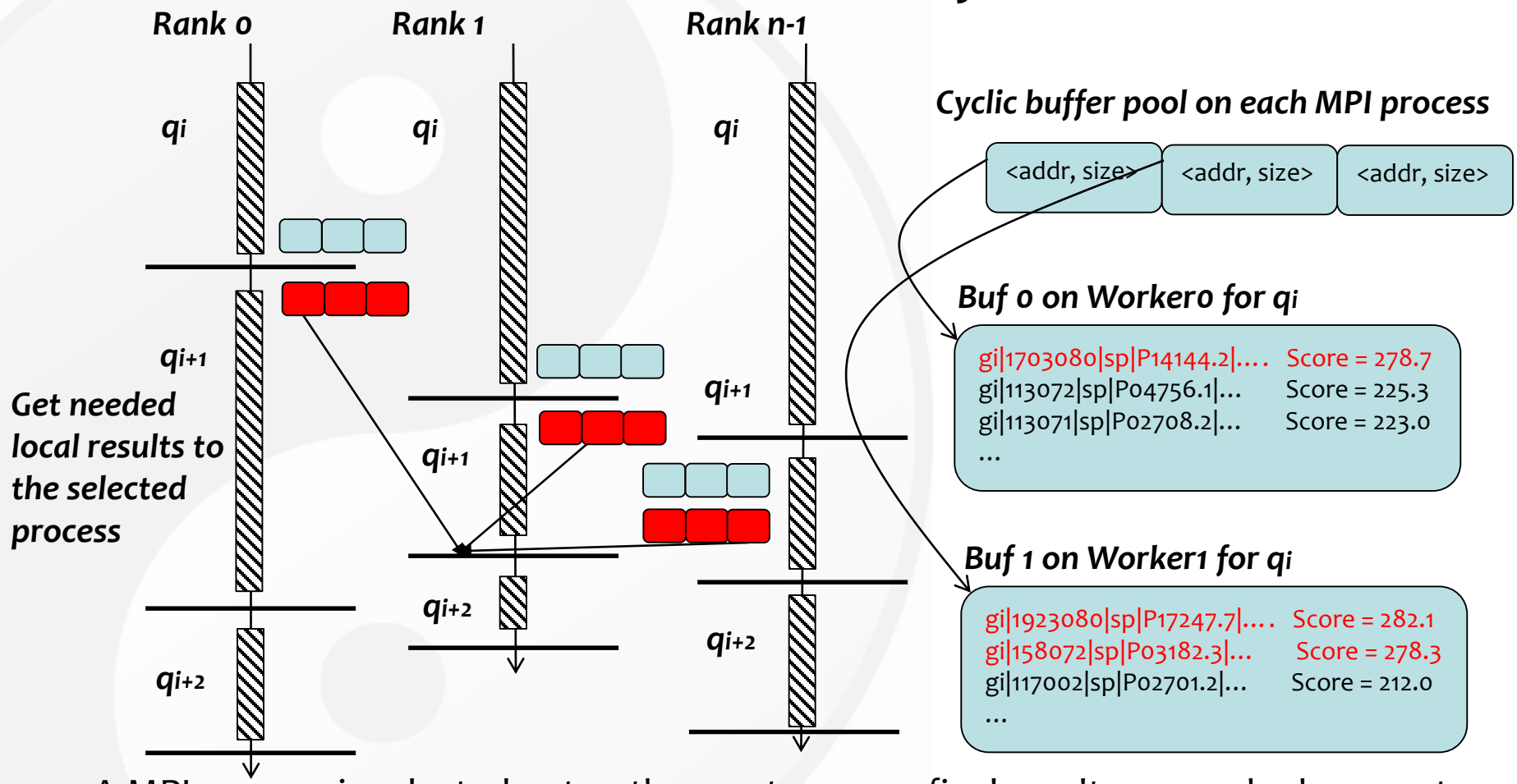
- After local search for a batch of query sequences, a MPI process will write meta data to others with MPI Put

Wait the Finish of Put on Previous Batch



- After MPI Put (for the current batch), MPI processes will
  - Wait for the finish of MPI Put of the previous batch, e.g., MPI_Win_fence()
  - Merge and sort meta data for the pervious batch and select out final results (meta data)

A Process will Gather Data by MPI Get



- A MPI process is selected out as the one to merge final results, e.g., who has most final results, and it will gather data from others by using MPI Get
- Other processes will continue the computation for the next batch

Summary of Our Method

create MPI Windows for metadata and local results

for batch 0 : n-1

Local search: do the sequence search on local partition of database

Write metadata: MPI_Put() metadata of current batch to others

Wait: wait for the finish of MPI_Put() on previous batch

Merge & sort: merge and sort metadata

if (I'm the selected process)

Get local results: MPI_Get() local results from all processes

Generate output: Sort and write final results for the previous batch

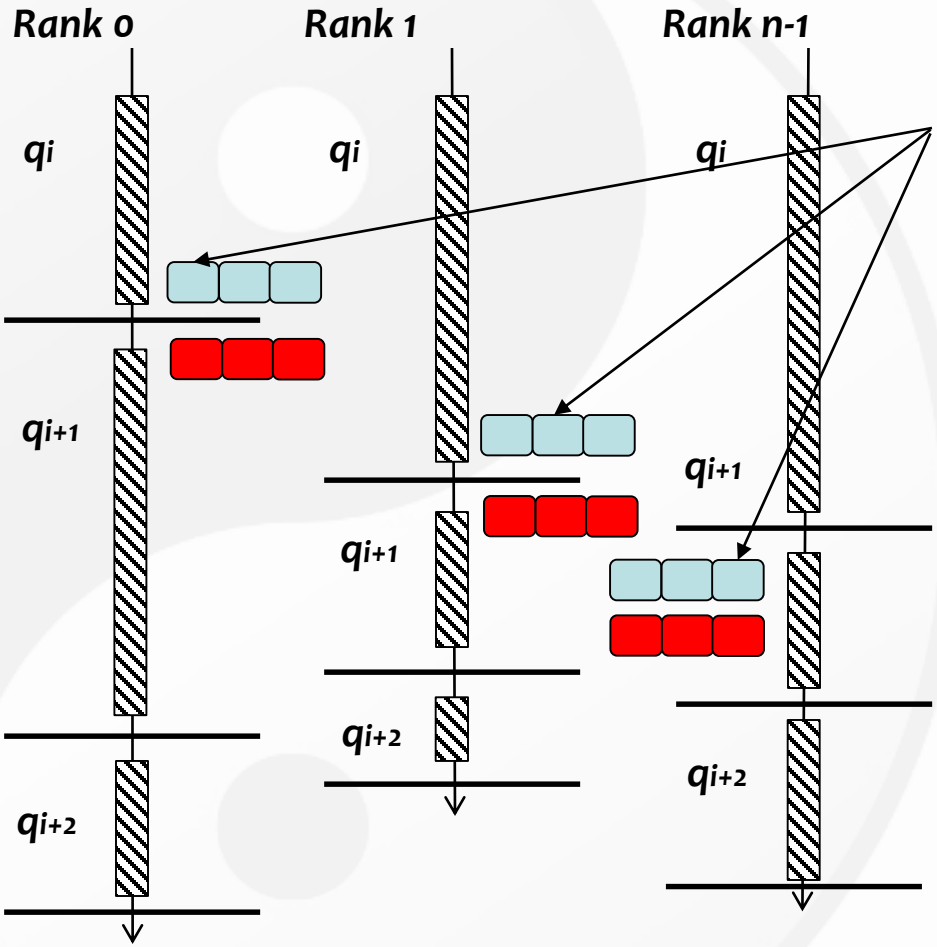
endif

endfor

Implementations and Optimizations

- Double buffering
  - Use double windows for metadata in order to wait for the finish of previous batch, after issuing MPI_Put() for the current batch
- Different types of synchronization methods
  - Fence mechanism: MPI_Win_fence()
  - Lock/unlock mechanism: MPI_Win_flush()
  - PSCW mechanism: MPI_Win_wait()

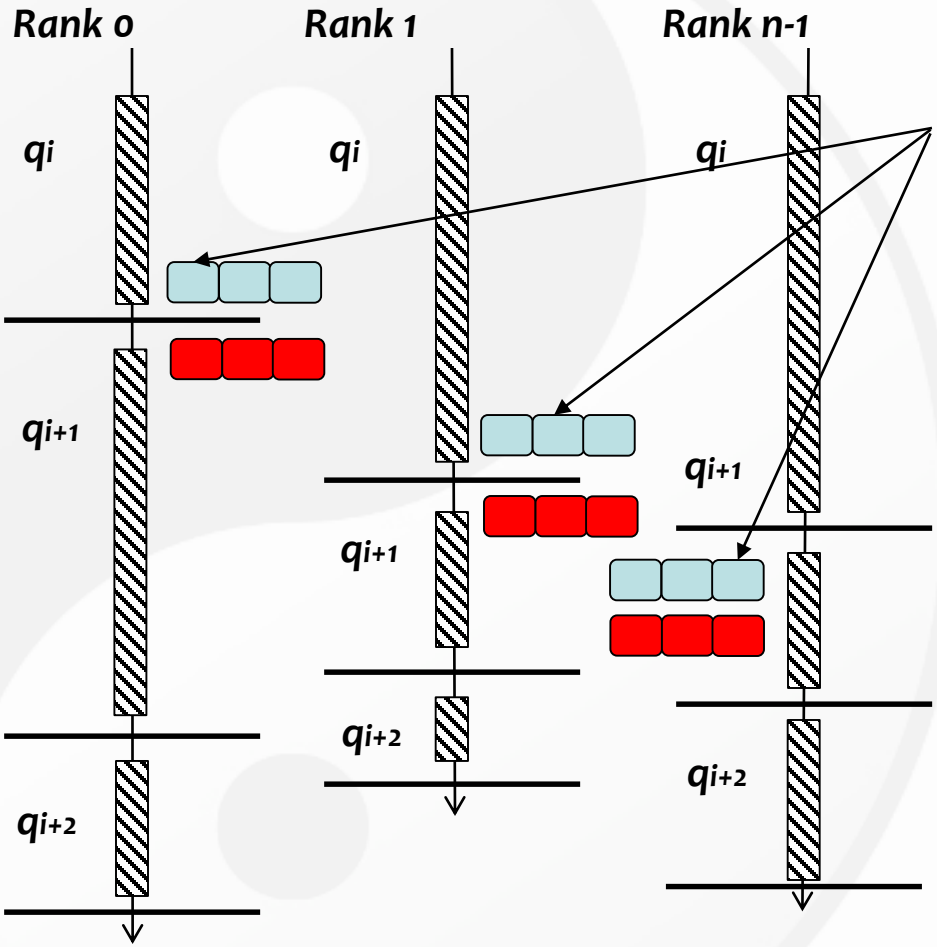
Implementations and Optimizations



Create one MPI window and register buffers to one window

- One window *vs* one windows per rank
  - Create one window per process to avoid the unnecessary wait

Implementations and Optimizations



Create n MPI windows and register the buffer per rank per window

- One window *vs* one windows per rank
  - Create one window per process to avoid the unnecessary wait

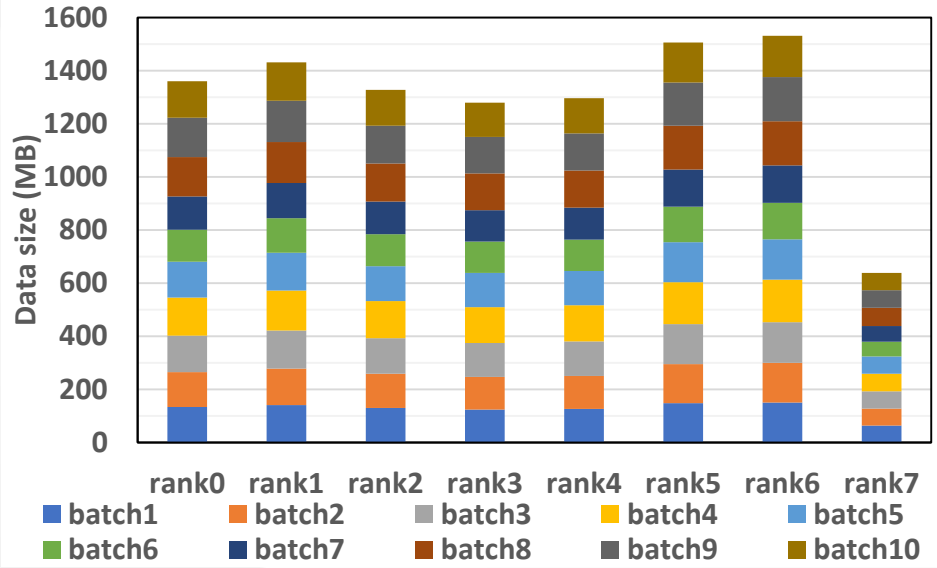
Outline

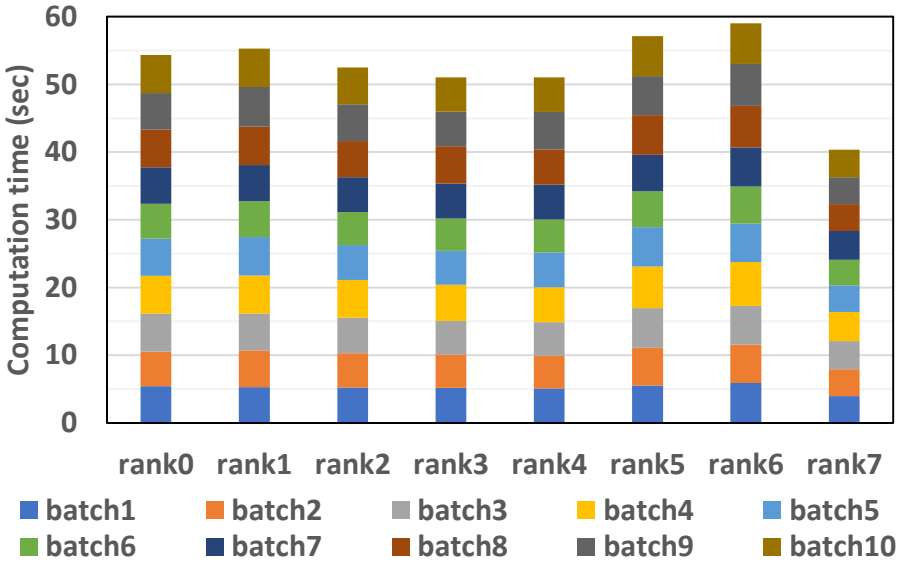
- Background
- Sequence Search
- Using one-sided communications for sequence search
- **Evaluation (early stage)**
- Summary and Future Work

Experimental Setups

- Hardware
  - Up to 16 compute nodes, each of which has 2 Intel Xeon CPU E5-2670 (Sandy Bridge EP, 16 cores in total)
  - 64 GB main memory
  - Mellanox ConnectX-3 MT27500
- Datasets
  - *env_nr* and *nr* databases from NCBI GeneBank
  - Randomly select 10000 sequences from the target database as query sequences
- Data partitions
  - Partition databases evenly on each compute node
- Software
  - DIAMOND (C++ Thread) + MPI
  - MVAPICH2 (version 2.2)

Breakdown

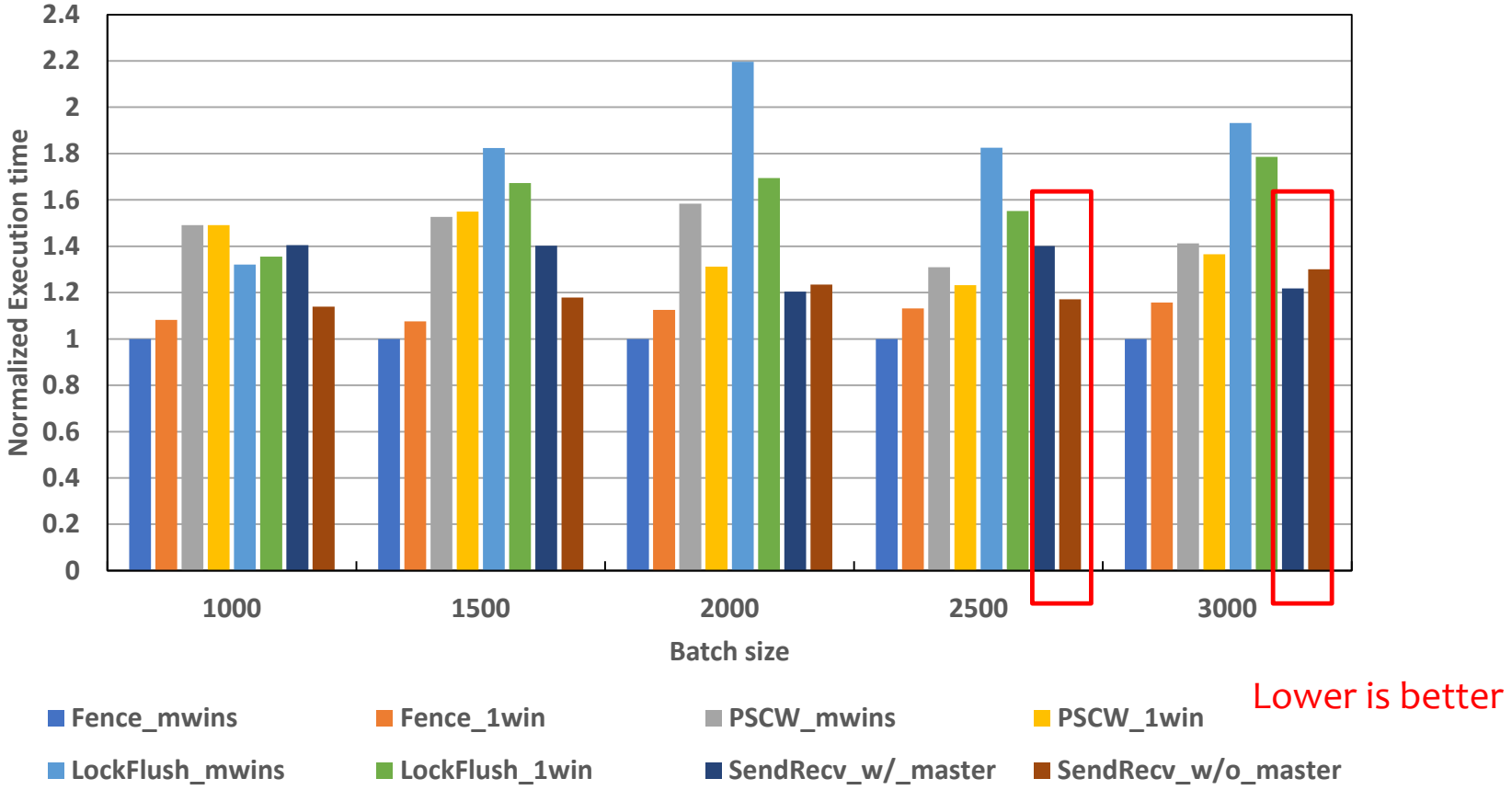




- Different MPI processes contribute different sizes of data to the final results
- Different MPI processes have different computation time in each batch

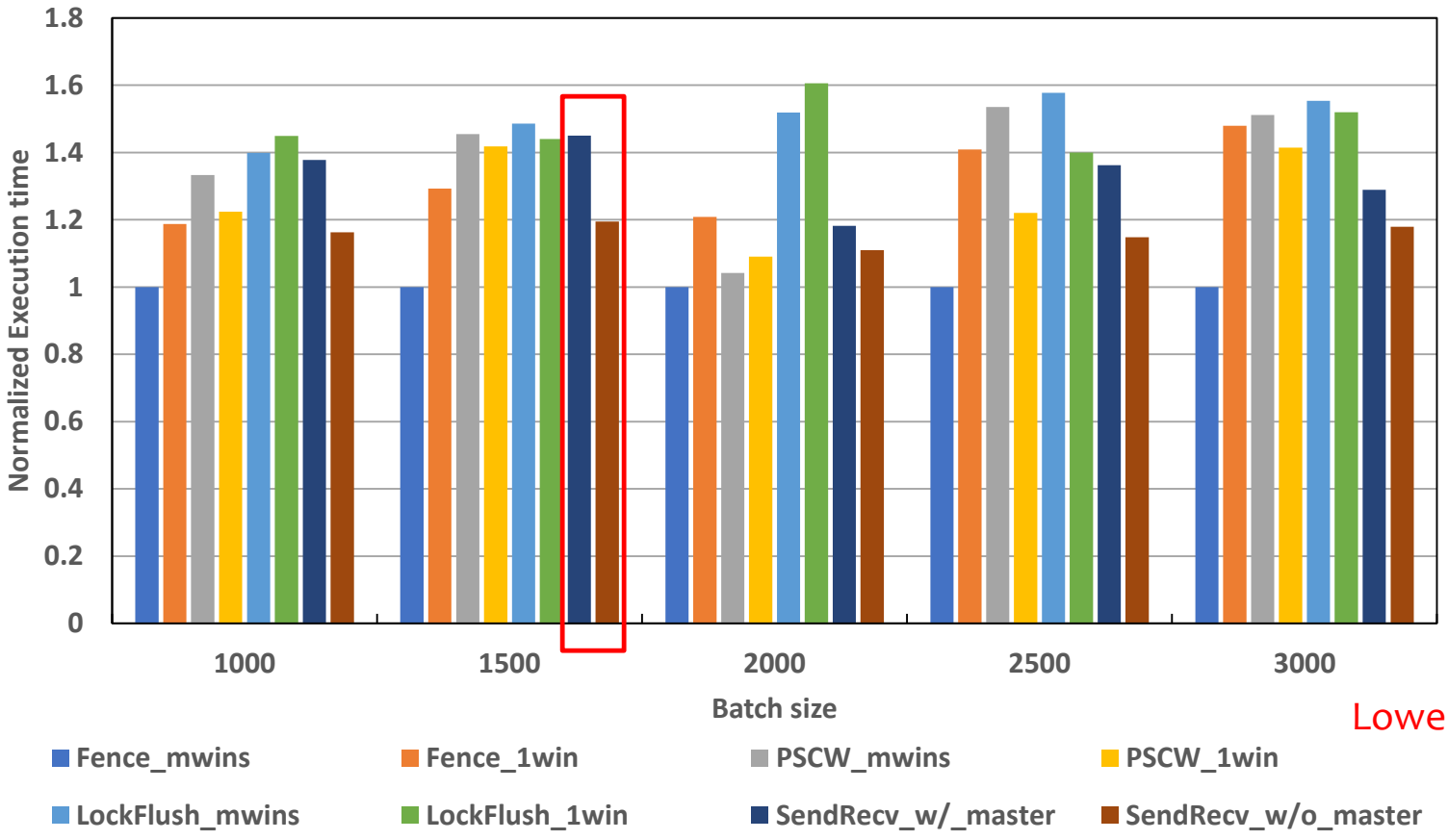
Setups: running on 8 nodes,10000 query sequences in 10 batches

Overall Performance on 8 nodes



- MPI_Win_Fence() with multiple windows is best
  - 1.4x and 1.32 x speedup over 2sided w/ and w/o master, respectively

Overall Performance on 16 nodes



- MPI_Win_Fence() with multiple windows is best
  - 1.42x and 1.19 x speedup over 2sided w/ and w/o master, respectively

Observations

- MPI fence exhibits better performance than MPI flush
  - All-to-all communication pattern in metadata communication

Summary and Future Work

- We use MPI one-sided communication to accelerate sequence search on InfiniBand clusters
- The experimental results show up to 1.42x speedup over two-sided communication
- We are analyzing performance numbers of different one-sided synchronization mechanisms
- We are collecting more application performance numbers, for mpiBLAST, DIAMOND, and pBWA
- We would like to check application performance with MVAPICH2-2.3b