Transforming Irregular Algorithms for Heterogeneous Computing - Case Studies in Bioinformatics

Jing Zhang

Advisor: Dr. Wu Feng

Collaborator: Hao Wang

VirginiaTech
Invent the Future

SyNeRG
synergy.cs.vt.edu

Irregular Algorithms

- Characterized by …
  - Operate on **irregular data structures** - trees, graphs, linked lists, priority queues
  - Data are processed in variable-iteration loops
  - Both memory access and control flow are **irregular** and **unpredictable**

- Emerging data intensive applications have increasing irregularity in memory access and control flow over massive data set
  - Bioinformatics applications
    - Human genome: 3.2 billion letters (3.2 Gb)
  - Social network analysis, graph algorithms
    - Twitter (June-Dec 2009) - 17 million users, 476 million tweets (6Gb)

Heterogeneous Computing Systems

- Heterogeneous computing systems can offer massively parallel computation with high power efficiency and throughput
  - **GPU, AMD APU**
  - **Intel Xeon Phi**
  - FPGA, DSP, …
- But, designed for regular programs, relying on data locality and regular computation to tolerate access latencies
  - Thousands of smaller (weak), more efficient (simple) cores
  - Limited cache size, i.e., short cache line lifetimes

Heterogeneous Computing Systems

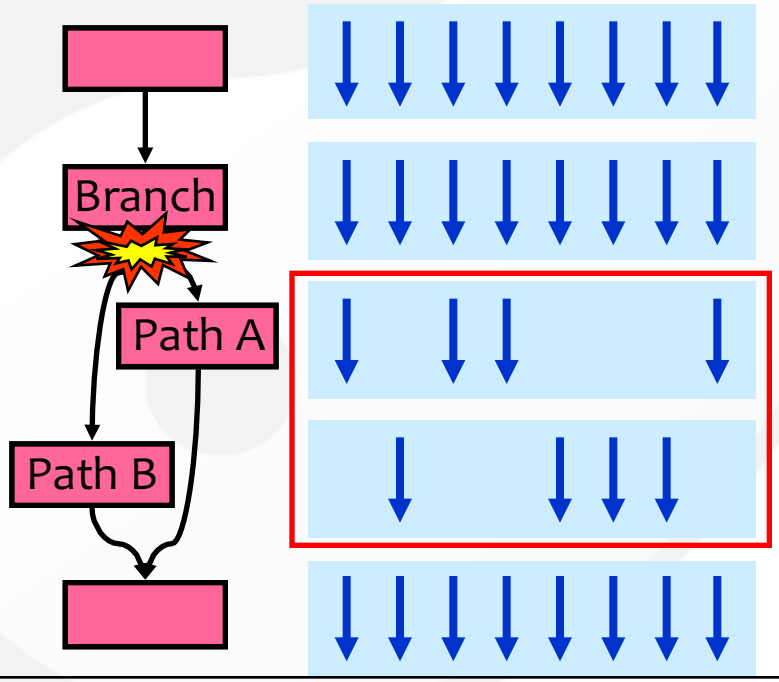
- Heterogeneous computing systems can offer massively parallel computation with high power efficiency and throughput
  - **GPU, AMD APU**
  - **Intel Xeon Phi**
  - FPGA, DSP, …

- But, designed for regular programs, relying on data locality and regular computation to tolerate access latencies
  - Thous… …es, which… …on, etc.
  - Limite…

It is very challenging to map irregular algorithms on heterogeneous computing systems!

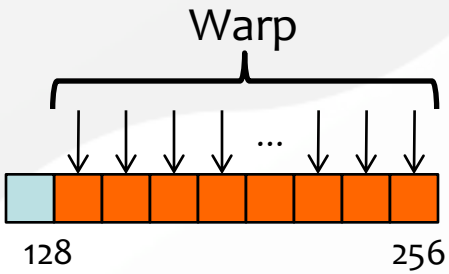
Impacts of Irregularity on GPU Performance

- Irregular control flow
  - Branch divergence
    - Occurs when threads inside warps branches to different execution paths
    - Waste computational resource

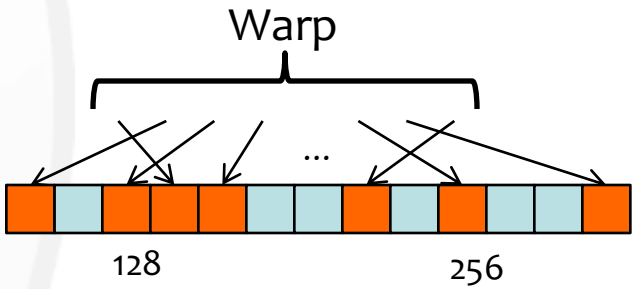


Impacts of Irregularity on GPU Perf (cont'd)

- Irregular memory access
  - Uncoalesced memory access
    - Lower memory bus utilization



One segments
A single 128-byte transaction
Coalesced Memory Access



Cross N segments
N 32-byte transaction
Uncoalesced Memory Access

Irregular Algorithms Optimizations on GPU

- Irregular control flow
  - Kernel fission
  - Sorting work
  - Warp-centric execution

  ……
- Irregular memory access
  - Memory access reordering
  - Exploiting memory hierarchy
  - Data structure adaptation

  ……

Case Study: Mapping BLASTp on GPU*

- BLAST - Basic Local Alignment Search Tool
  - Find most similar sequences from database for a query sequence
  - Use ***heuristic*** algorithms, which have significant irregularities in both memory access and control flow

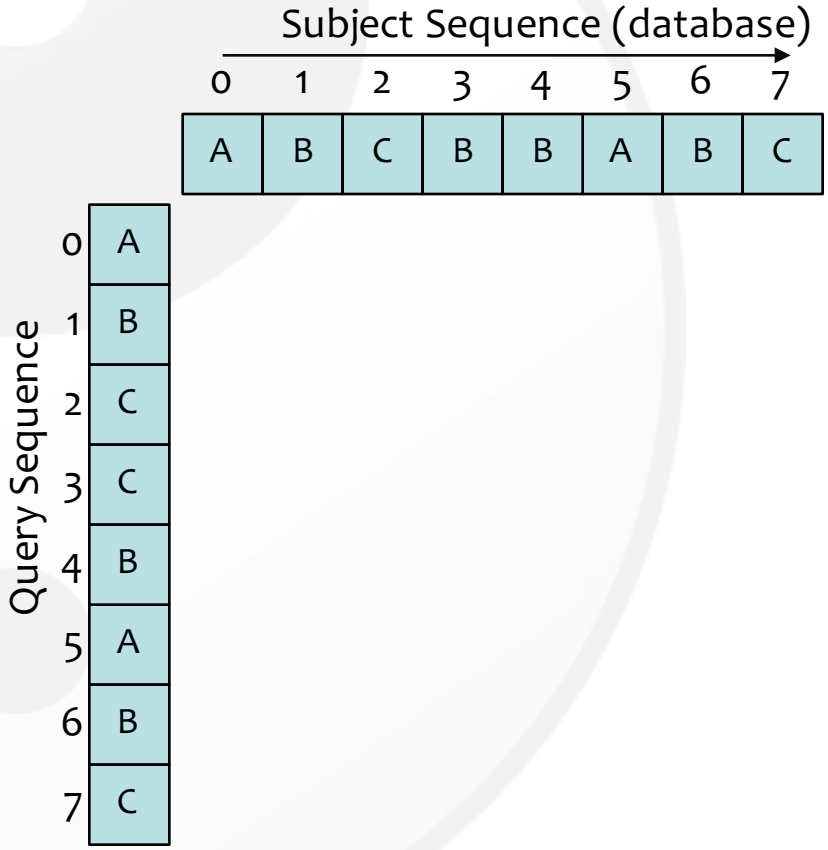
- P is for protein sequence; N is for nucleotide sequence

*: J. Zhang, H. Wang, H. Lin, W. Feng, "*cuBLASTP: Fine-Grained Parallelization of Protein Sequence Search on a GPU*", IPDPS 2014

BLAST Algorithm

Four stages in BLAST

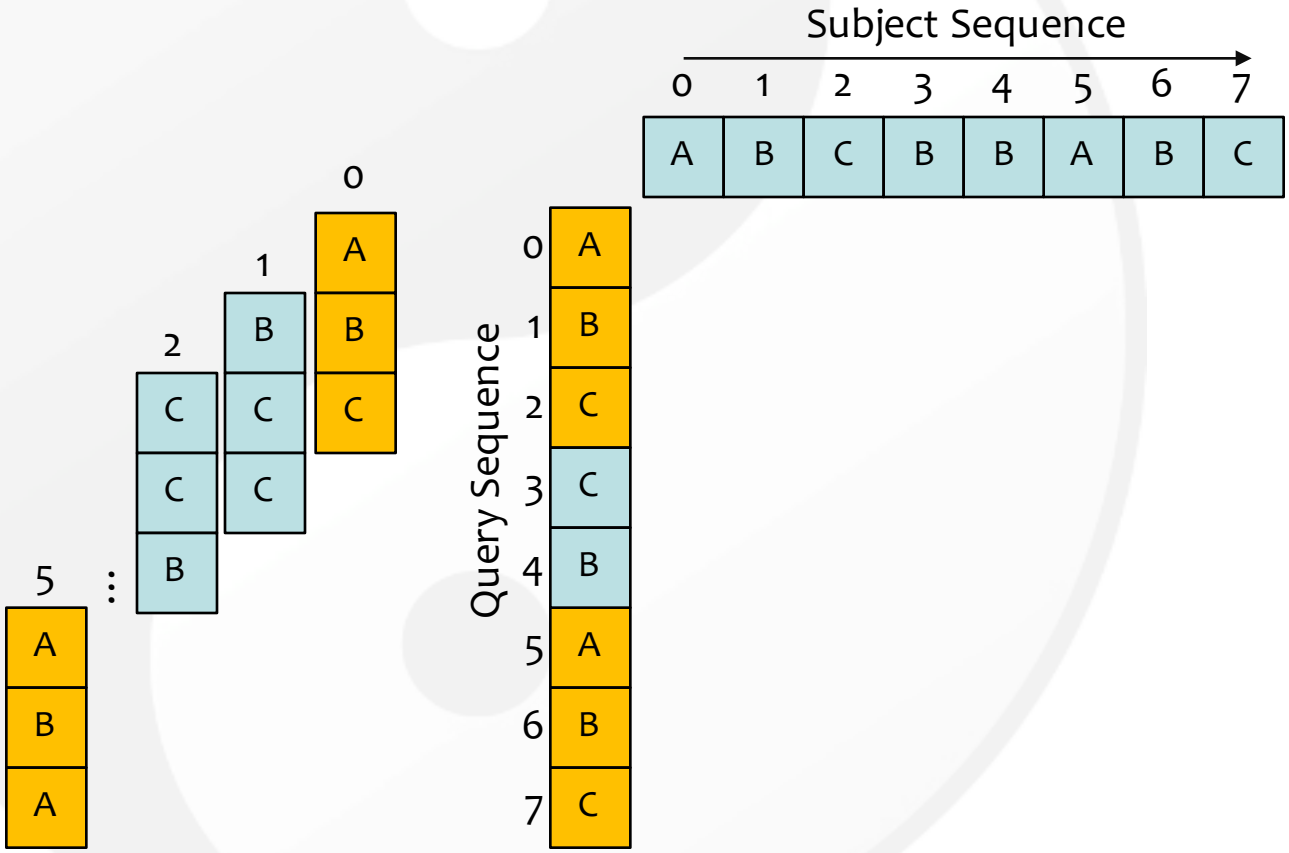
1. Hit detection
2. Ungapped extension
3. Gapped extension
4. Final extension



BLAST Algorithm

Four stages in BLAST

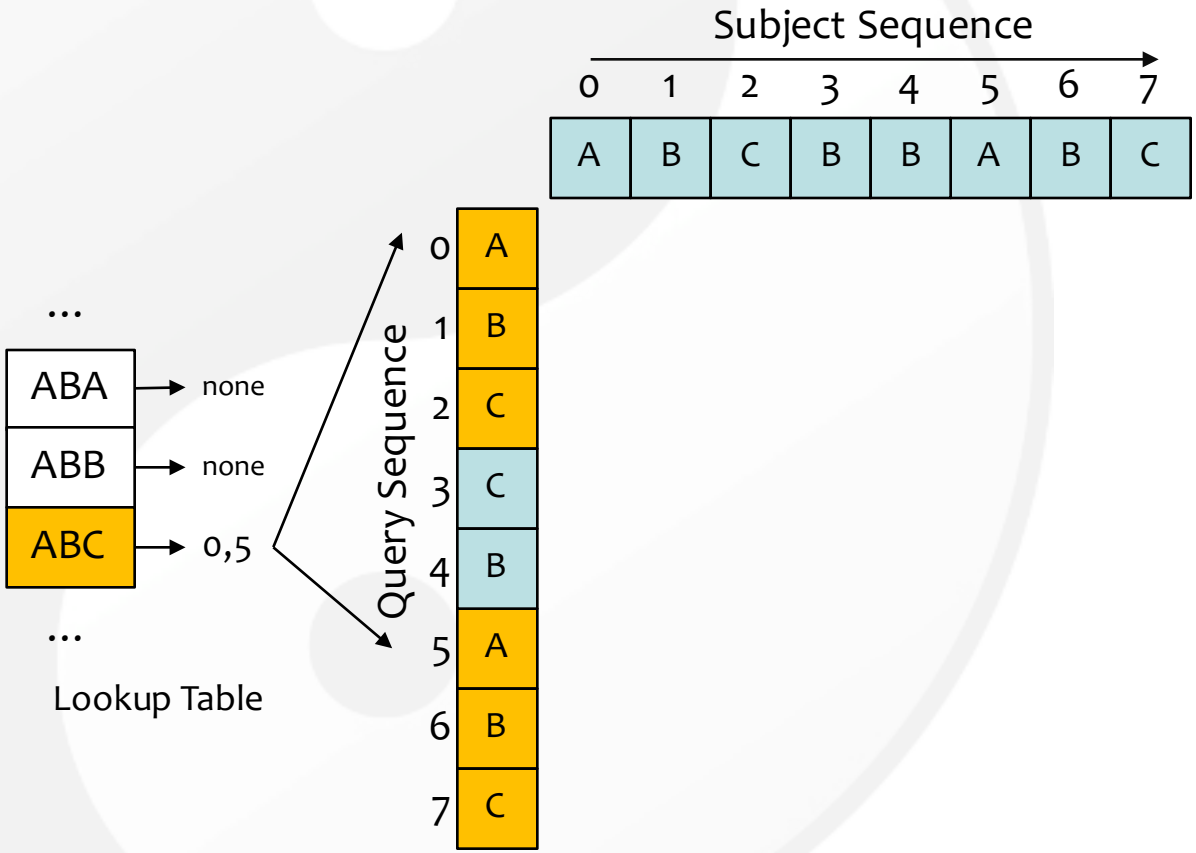
1. Hit detection
2. Ungapped extension
3. Gapped extension
4. Final extension



BLAST Algorithm

Four stages in BLAST

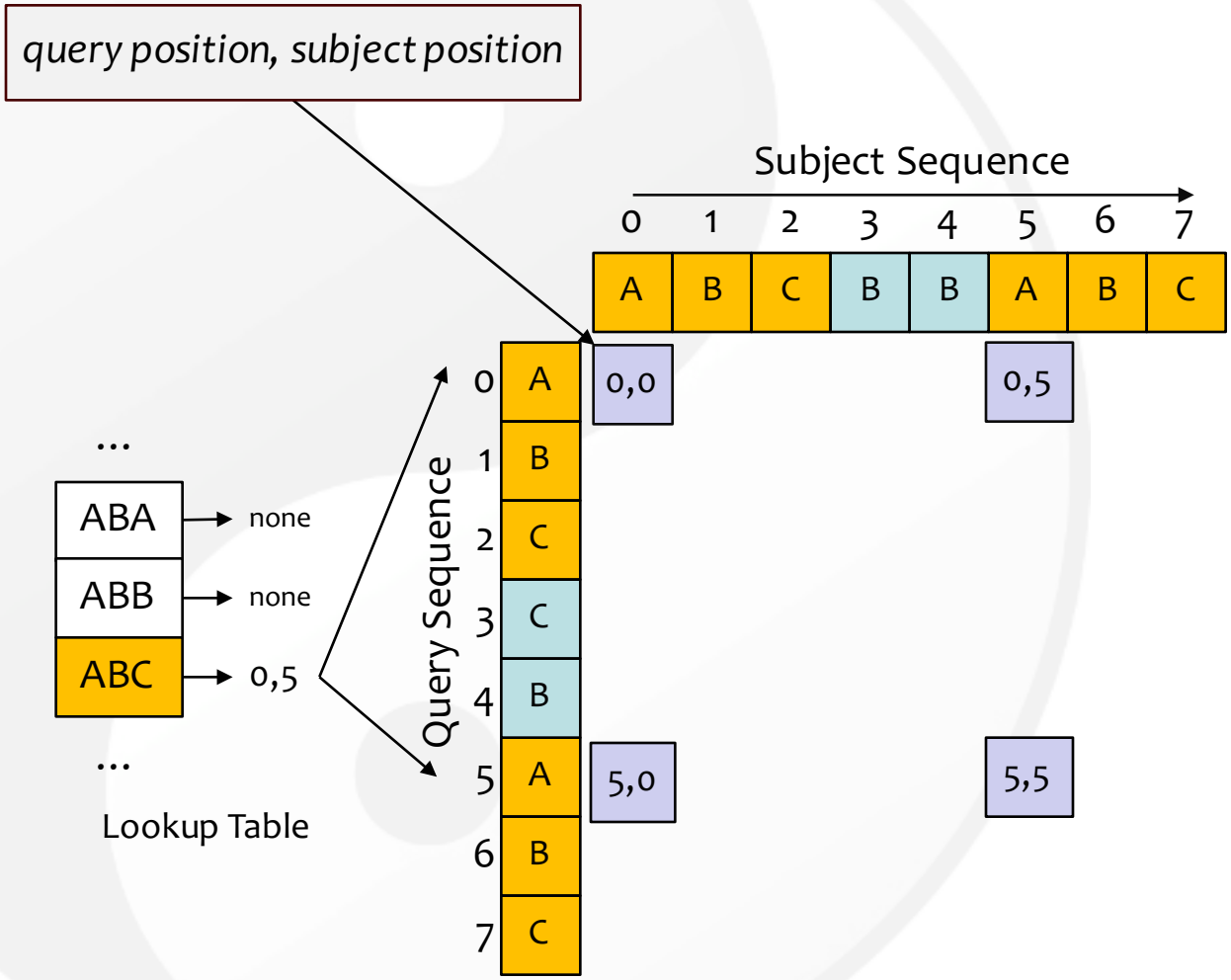
1. Hit detection
2. Ungapped extension
3. Gapped extension
4. Final extension



BLAST Algorithm

Four stages in BLAST

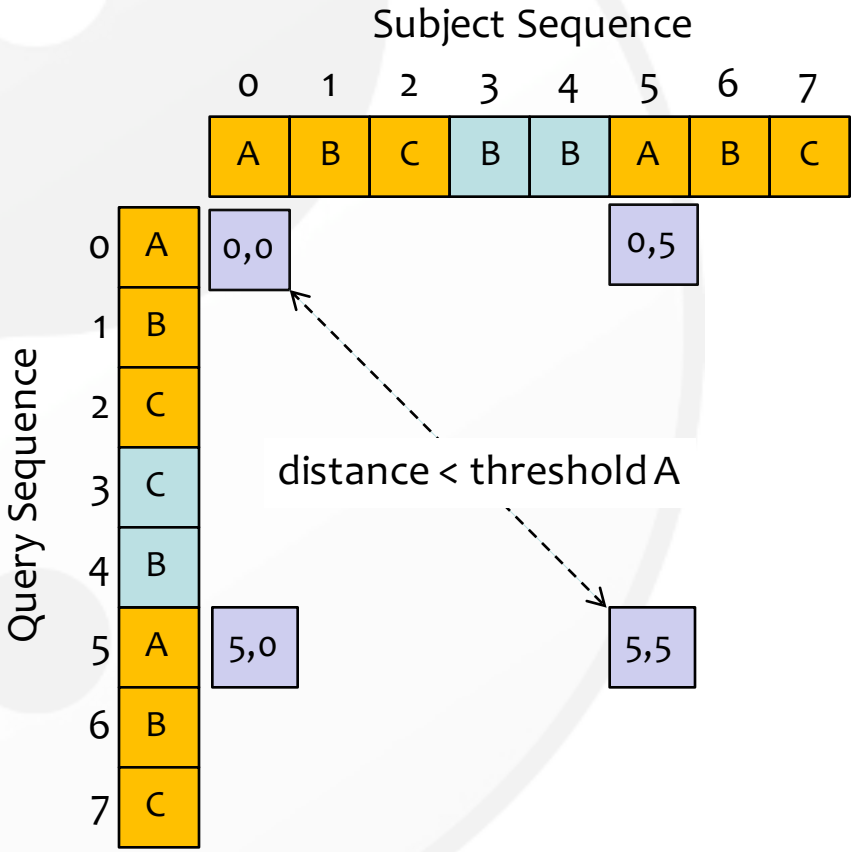
1. Hit detection
2. Ungapped extension
3. Gapped extension
4. Final extension



BLAST Algorithm

Four stages in BLAST

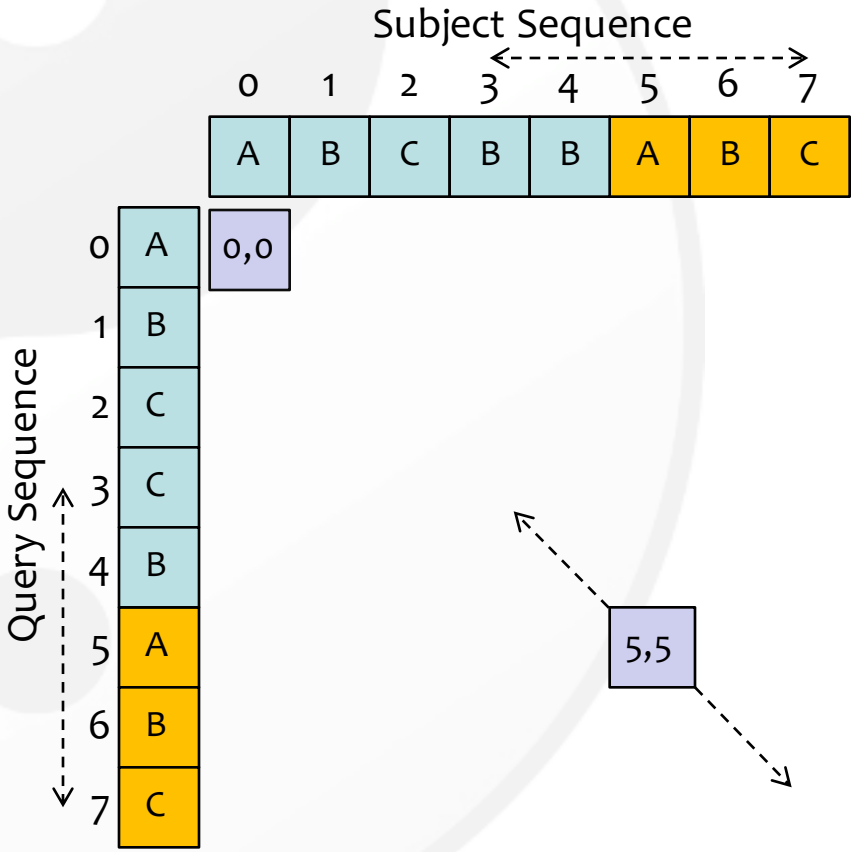
1. Hit detection
2. Ungapped extension
3. Gapped extension
4. Final extension



BLAST Algorithm

Four stages in BLAST

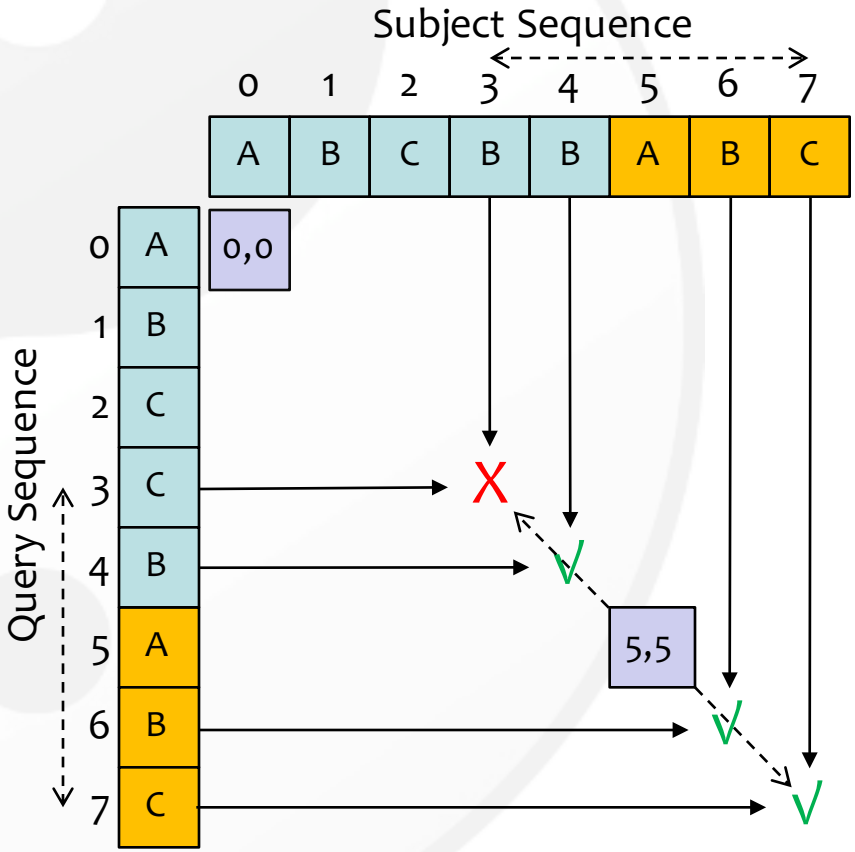
1. Hit detection
2. Ungapped extension
3. Gapped extension
4. Final extension



BLAST Algorithm

Four stages in BLAST

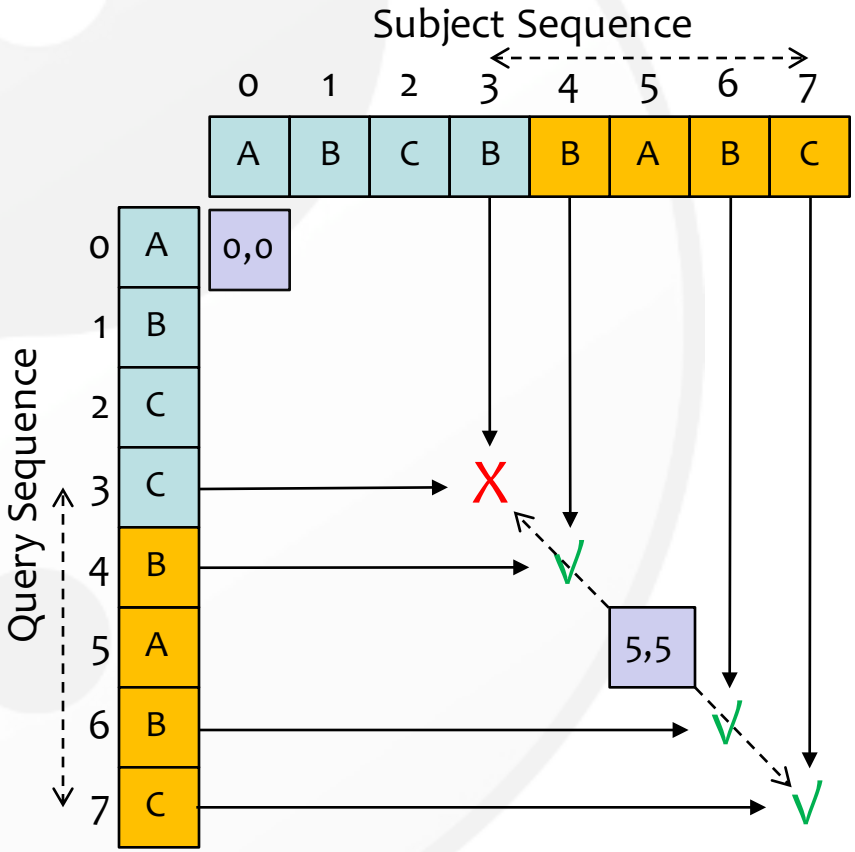
1. Hit detection
2. Ungapped extension
3. Gapped extension
4. Final extension



BLAST Algorithm

Four stages in BLAST

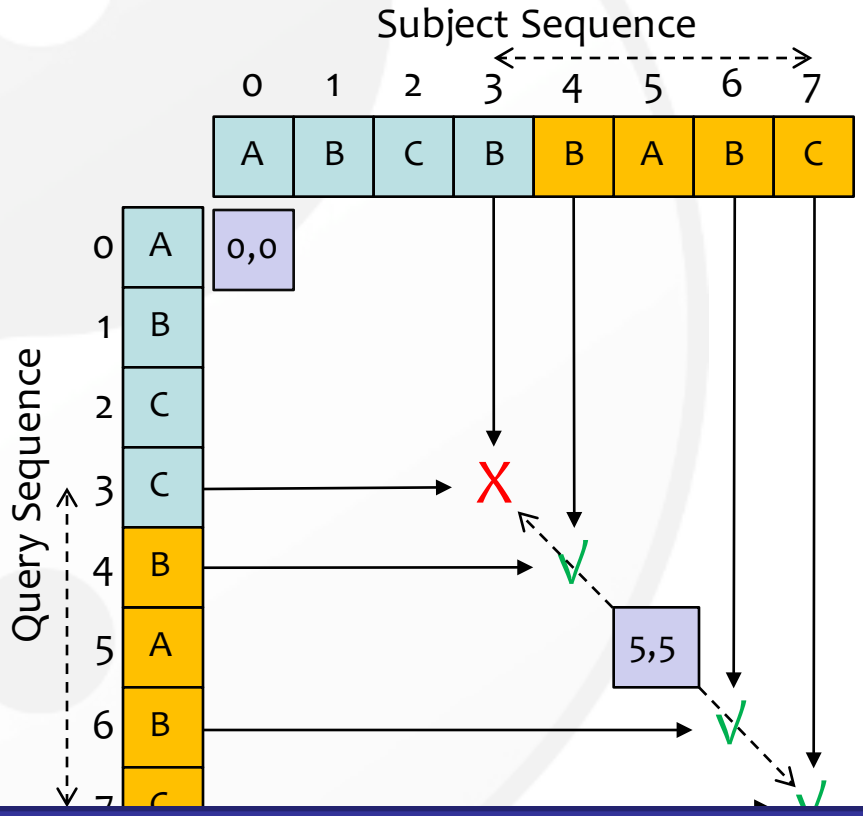
1. Hit detection
2. Ungapped extension
3. Gapped extension
4. Final extension



BLAST Algorithm

Four stages in BLAST

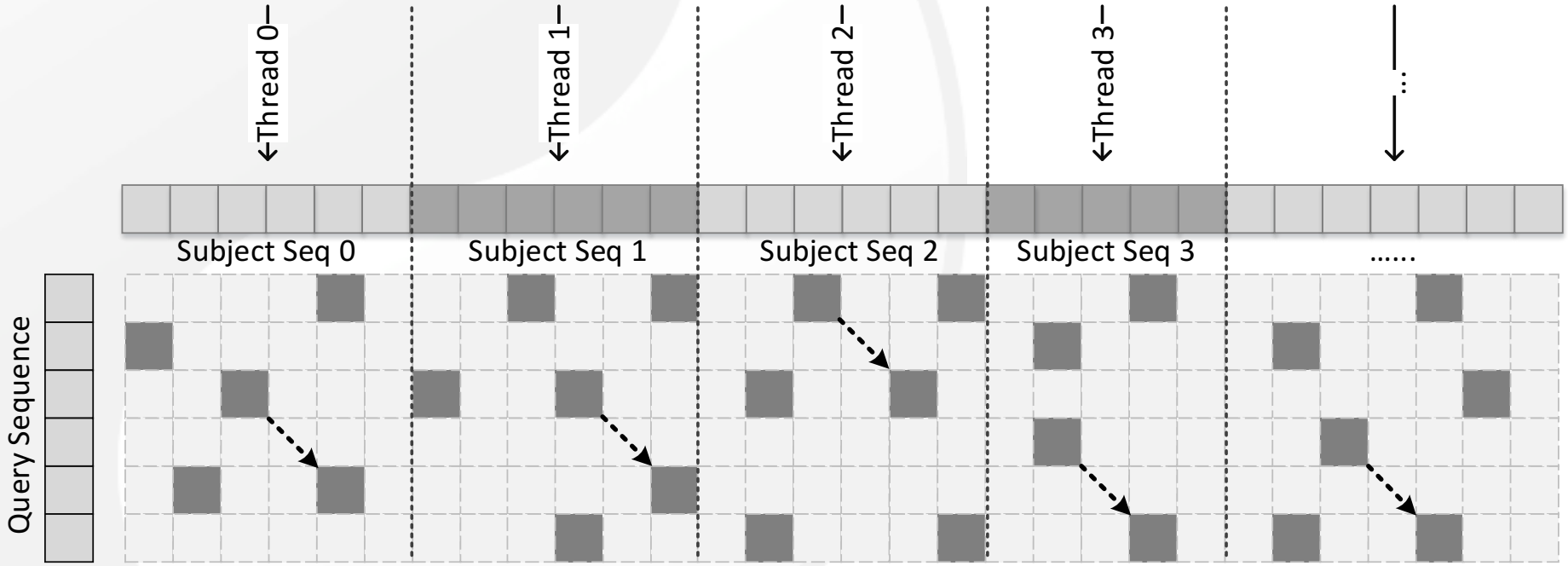
1. Hit detection
2. Ungapped extension
3. Gapped extension
4. Final extension



Knowledge: "seed-and-extend" – the most widely used alignment paradigm

State-of-the-Art GPU BLAST Approaches*

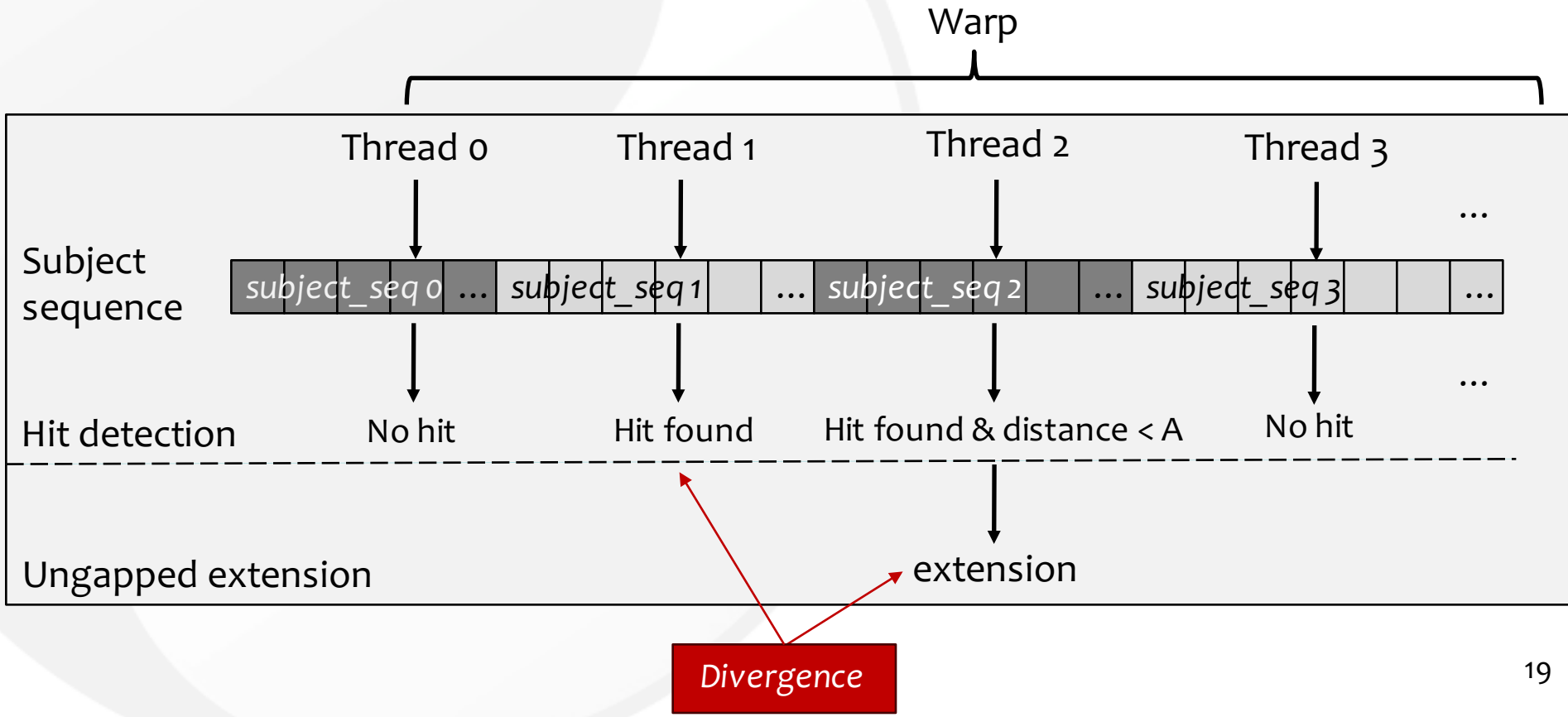
- Use intuitive parallelization - "coarse-grained parallelization", where a thread compares the query sequence to a subject sequence



*: Design and Implementation of a CUDA-compatible GPU-based Core for Gapped BLAST Algorithm (ICCS '10)
GPU- BLAST: Using Graphics Processors to Accelerate Protein Sequence Alignment CUDA-BLASTP (Binformatics)
CUDA-BLASTP: Accelerating BLASTP on CUDA-enabled Graphics Hardware (TCBB)
Accelerating Protein Sequence Search in a Heterogeneous Computing System. (IPDPS '11)

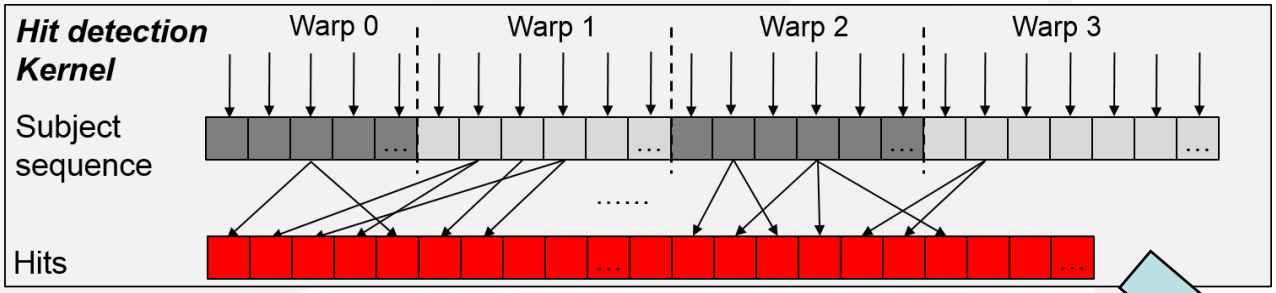
Challenge #1: Control Flow Irregularity

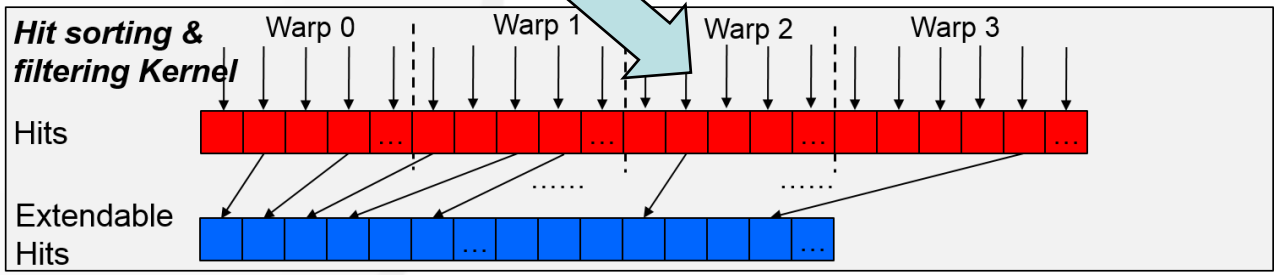
- With coarse-grained parallelism, different threads may execute across different stages

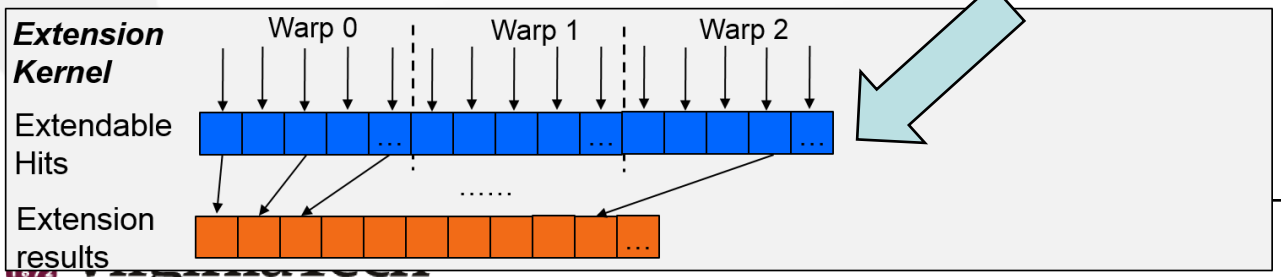


Solution: Kernel Fission

- Split hit detection and ungapped extension stage into separate kernels
- Use intermediate kernels to bridge the two stages

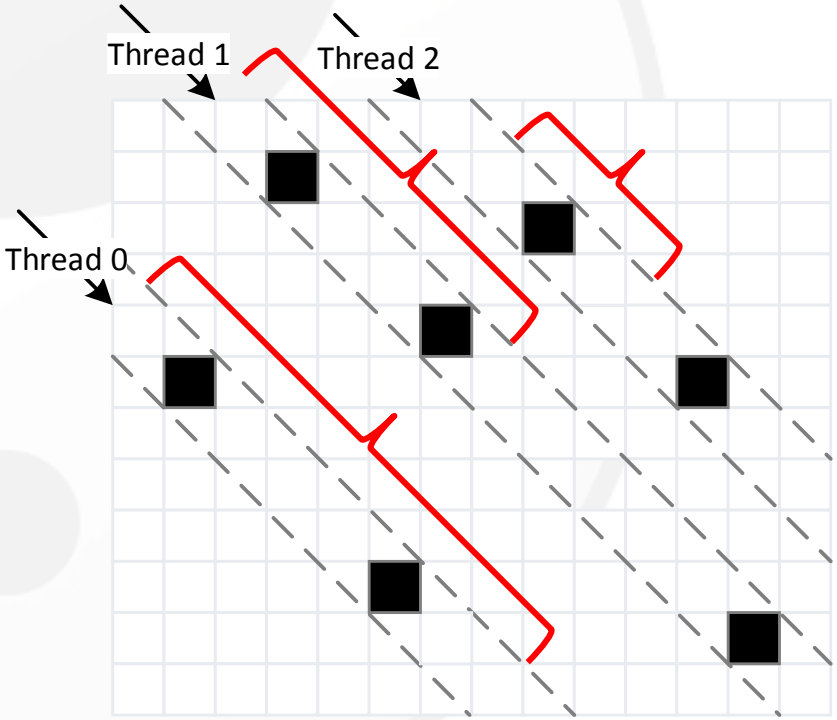






Challenge #2: Control Flow Irregularity (2)

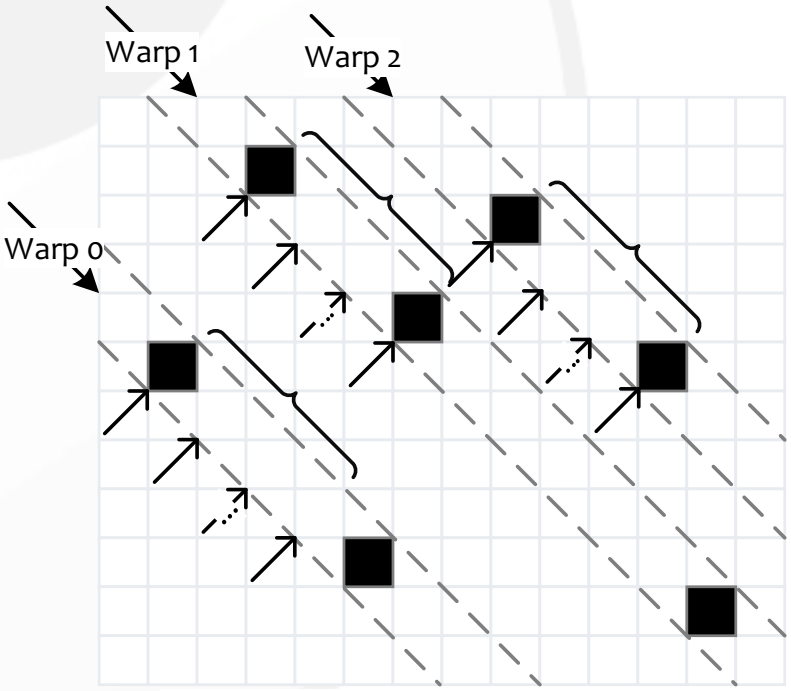
- Coarse-grained extension, where a thread is responsible for a diagonal, can result in divergence, since different diagonals could be extended to different lengths



One thread per diagonal

Solution: Warp-centric Extension

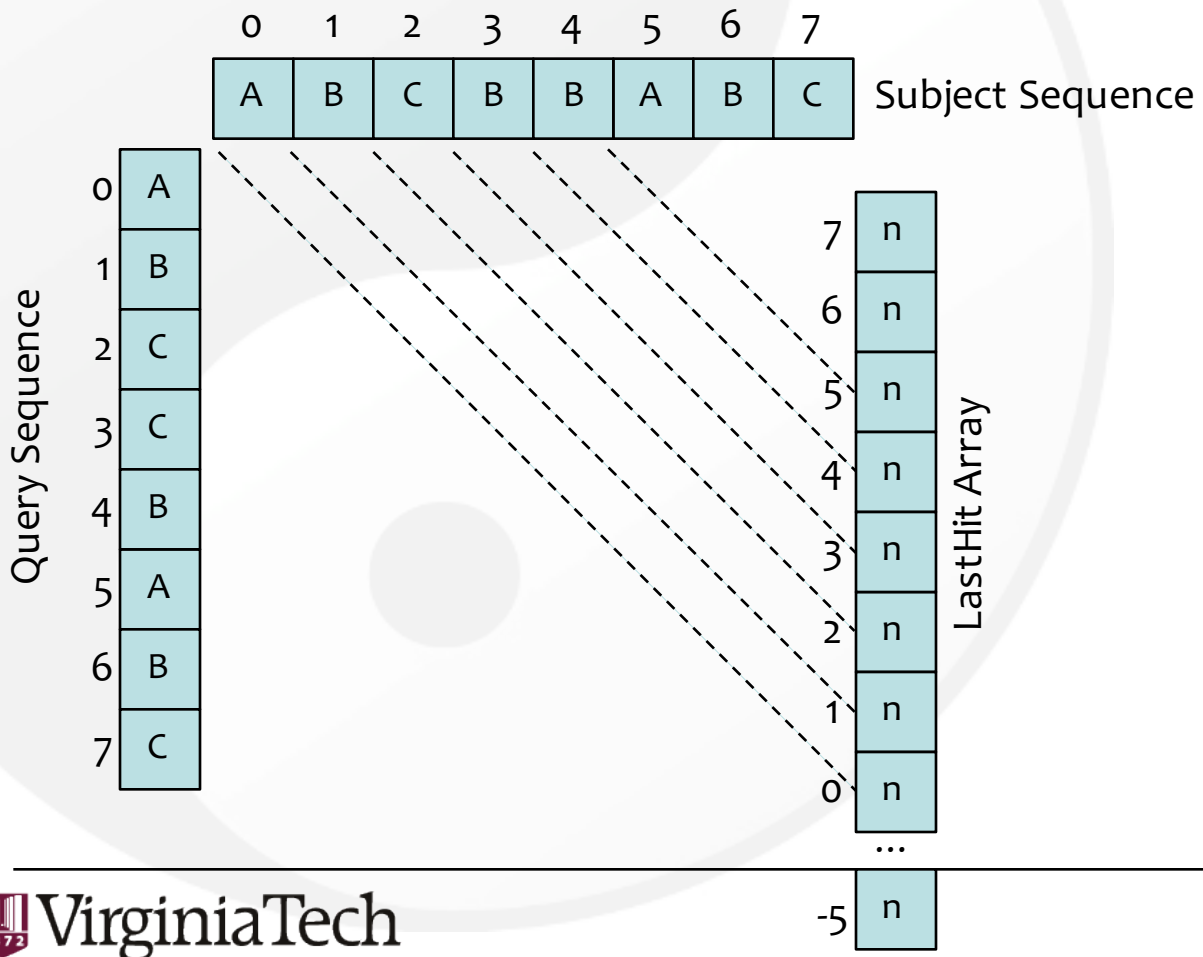
- Map a warp of threads to one diagonal
- Threads in a warp checks different positions concurrently



Warp-based Extension

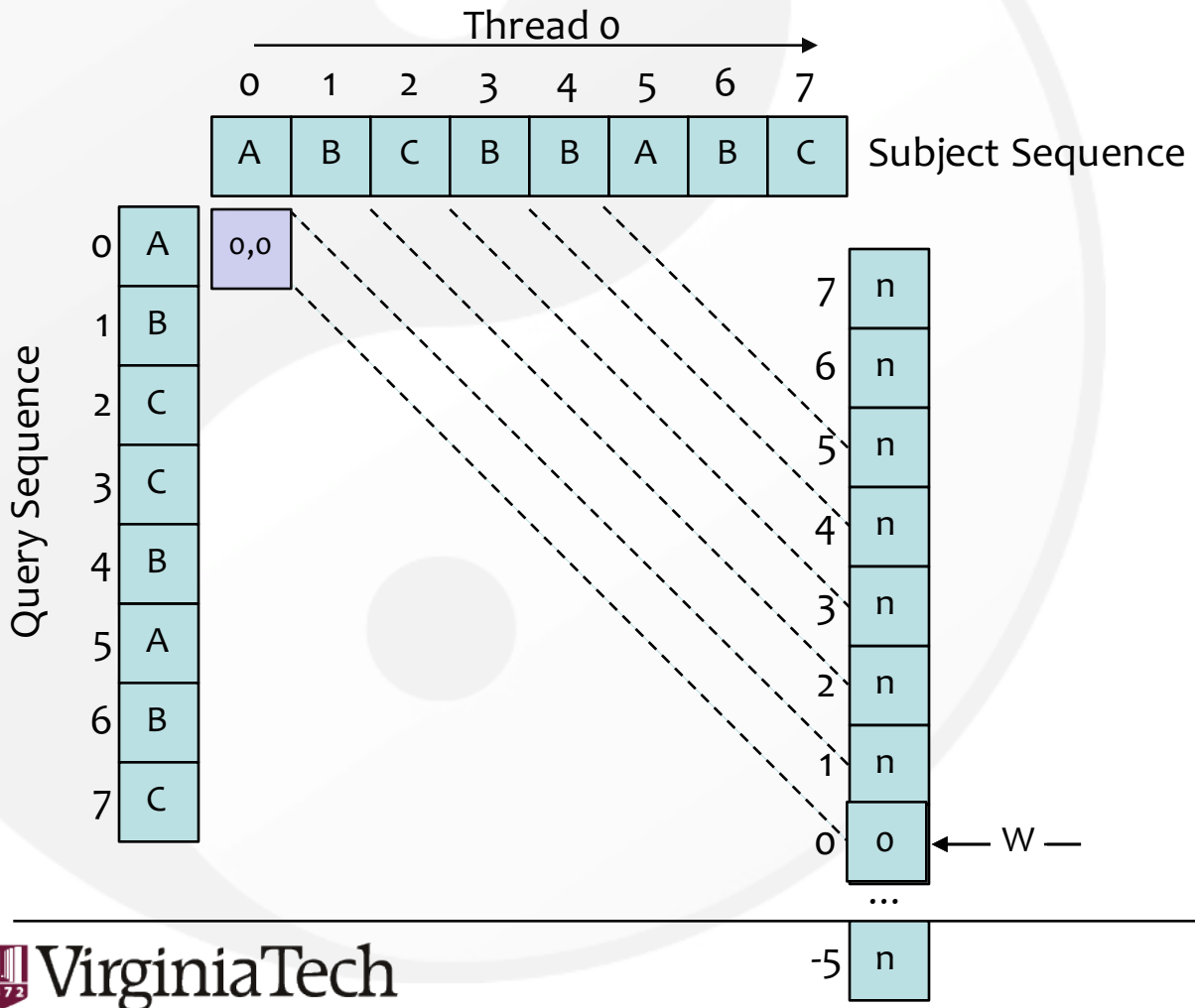
Challenge #3: Memory Access Irregularity

- Use a global array, called lastHit Array, to record the previous hit in each diagonal



Challenge #2: Memory Access Irregularity

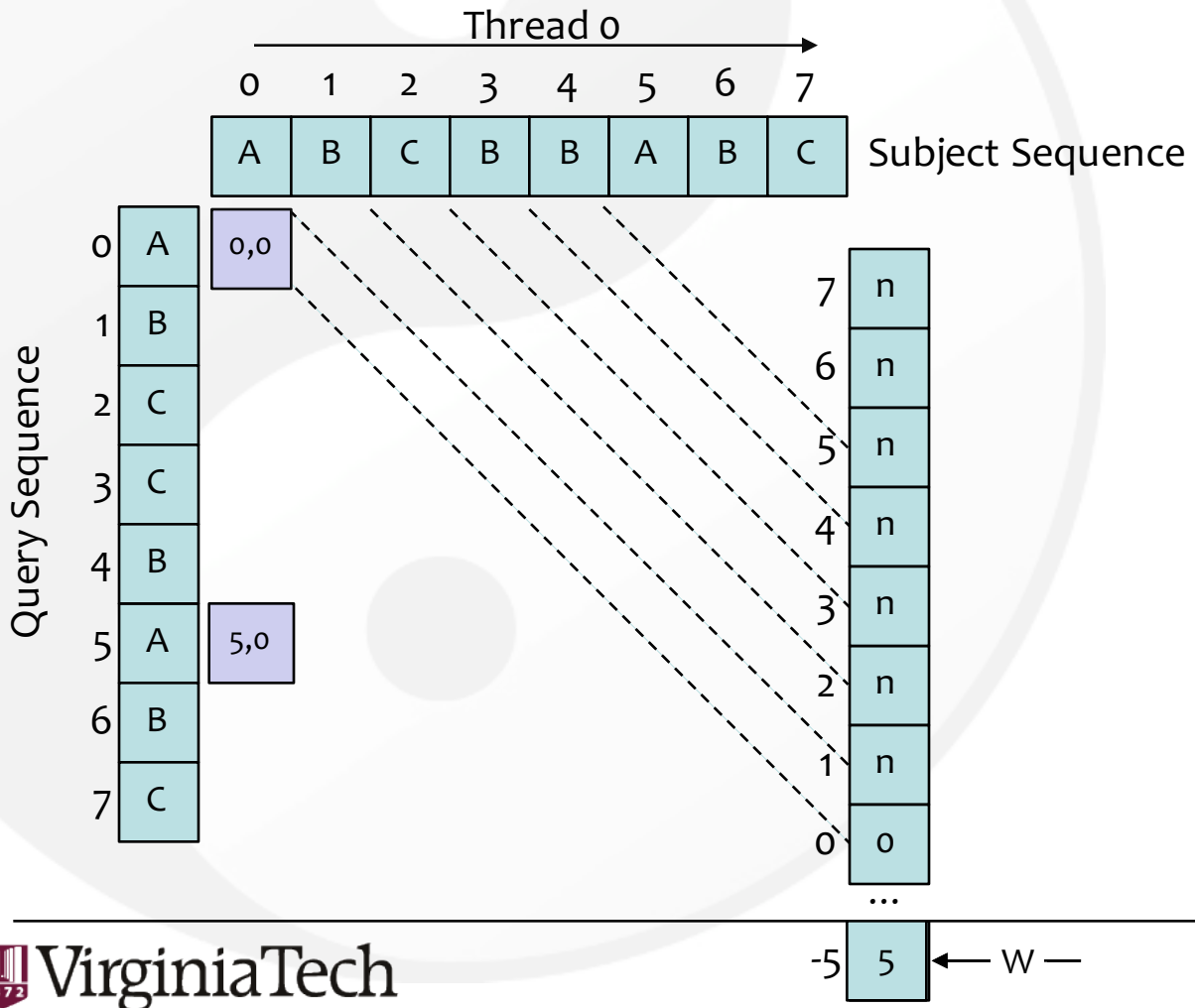
- Read LastHit Array to get last hit position in the diagonal
- Write back current position to lastHit Array



Operation	Position
Read	o
Write	o

Challenge #2: Memory Access Irregularity

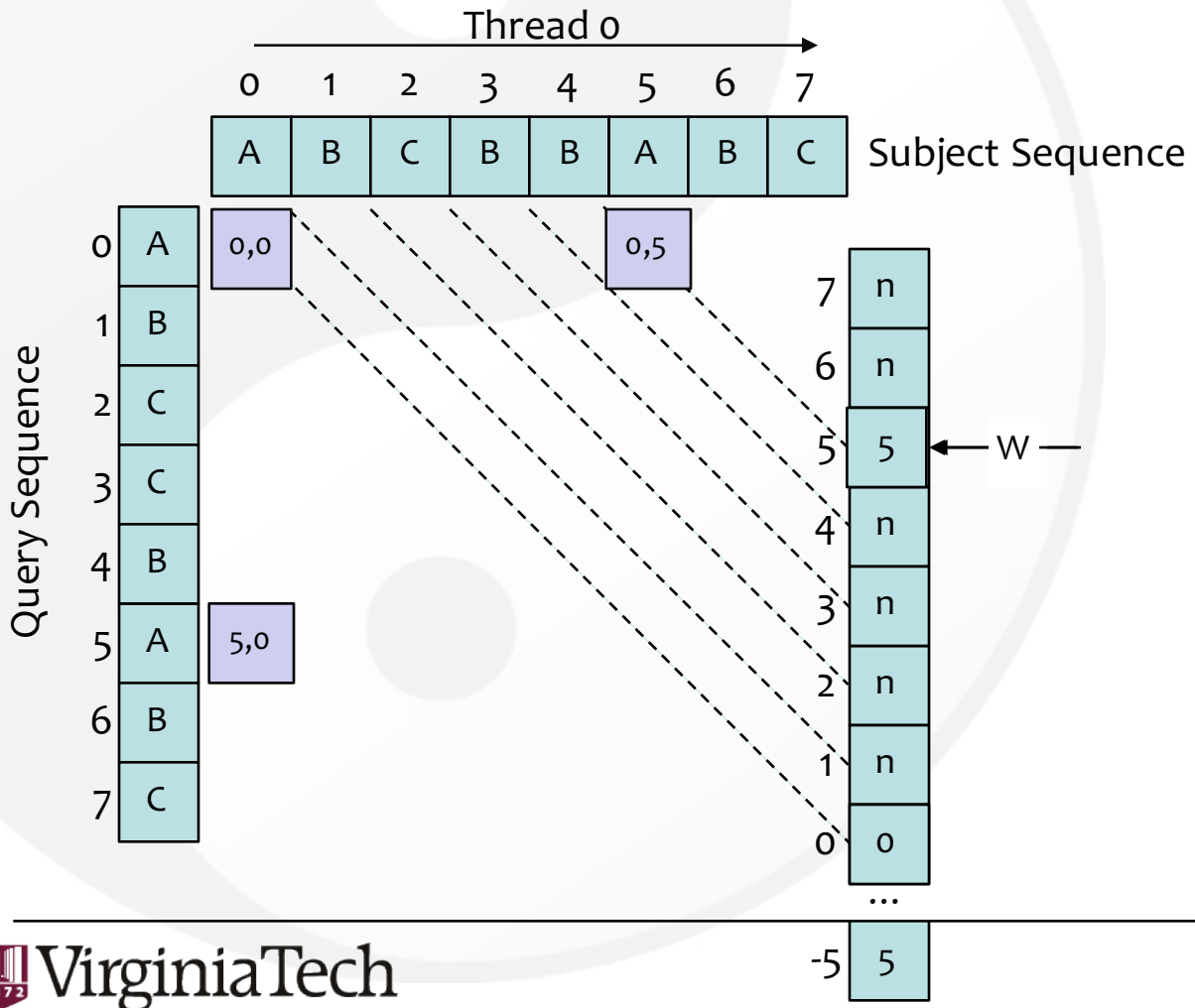
- Read LastHit Array to get last hit position in the diagonal
- Write back current position to lastHit Array



Operation	Position
Read	0
Write	0
Read	-5
Write	-5

Challenge #2: Memory Access Irregularity

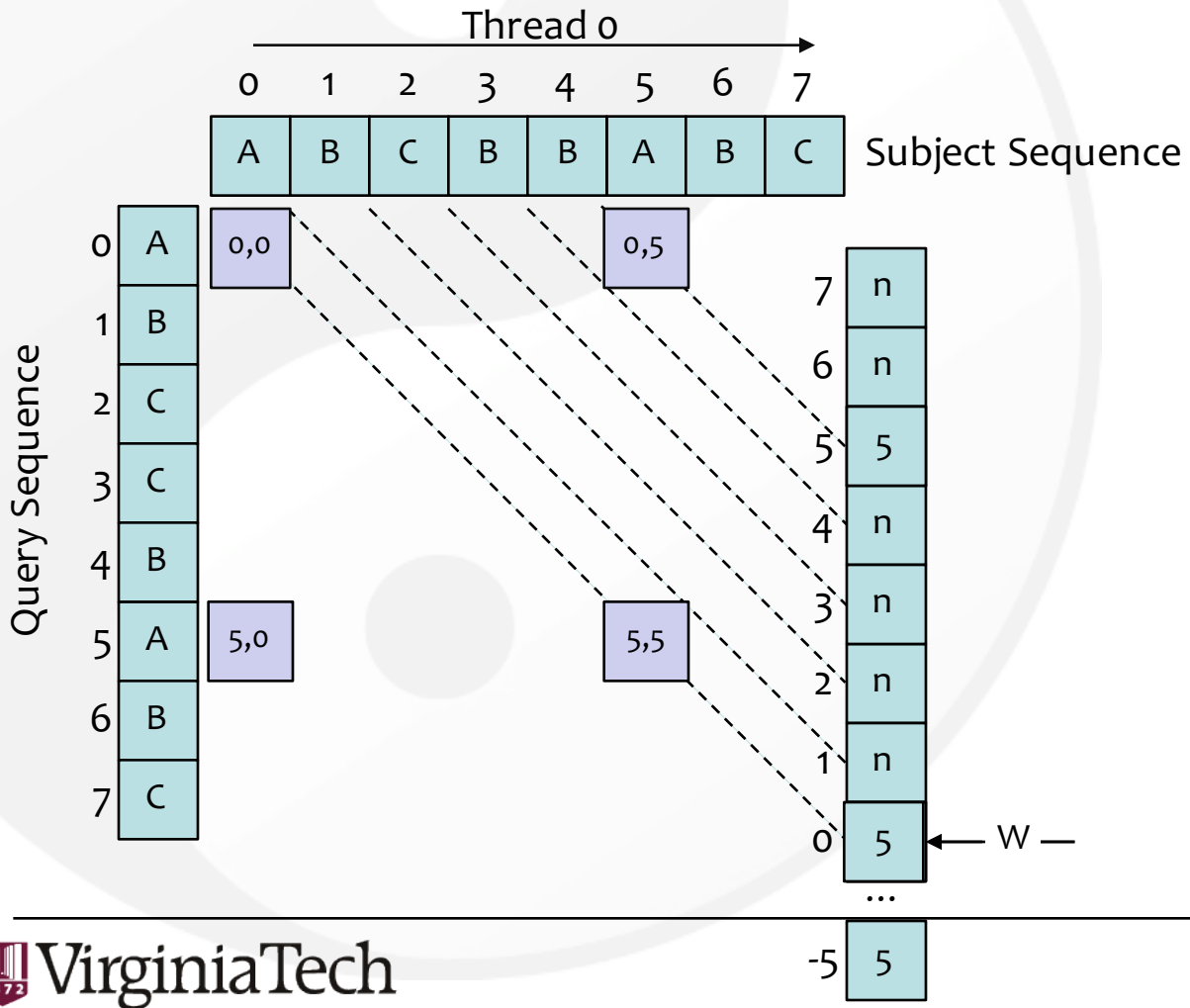
- Read LastHit Array to get last hit position in the diagonal
- Write back current position to lastHit Array



Operation	Position
Read	0
Write	0
Read	-5
Write	-5
Read	5
Write	5

Challenge #2: Memory Access Irregularity

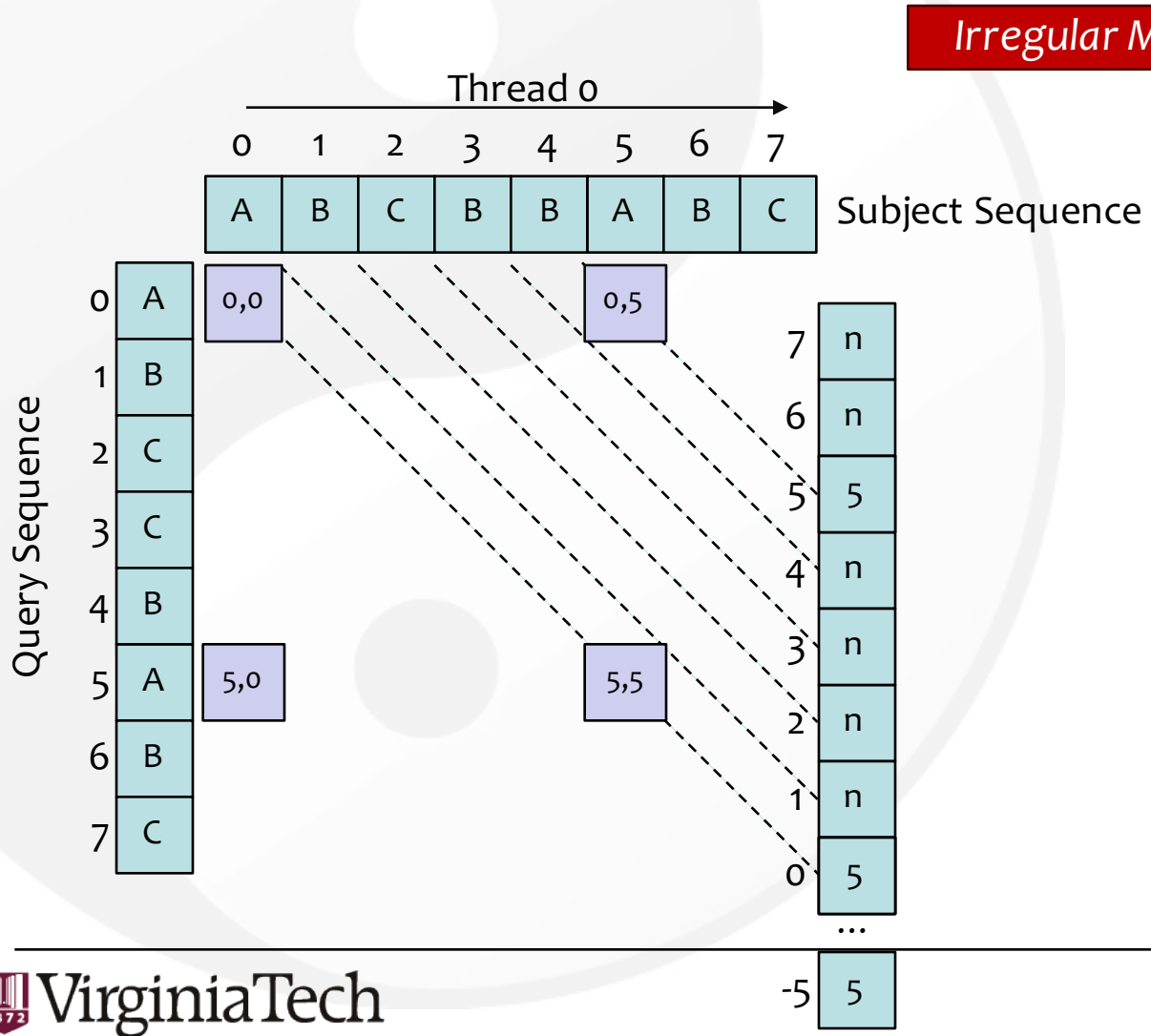
- Read LastHit Array to get last hit position in the diagonal
- Write back current position to lastHit Array



Operation	Position
Read	0
Write	0
Read	-5
Write	-5
Read	5
Write	5
Read	0
Write	0

Challenge #2: Memory Access Irregularity

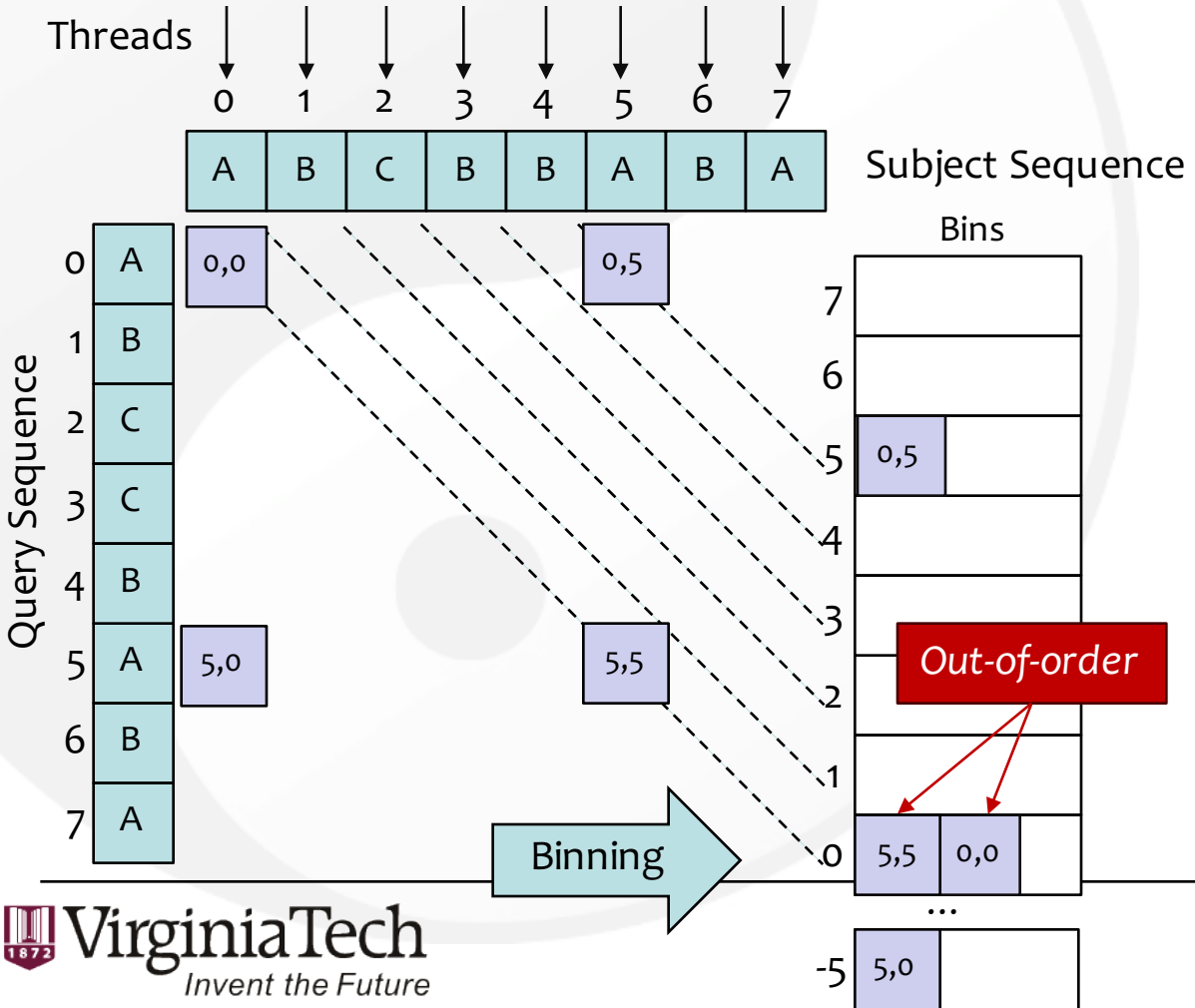
- Irregular Read/Write Ops on LastHit array



Operation	Position
Read	0
Write	0
Read	-5
Write	-5
Read	5
Write	5
Read	0
Write	0

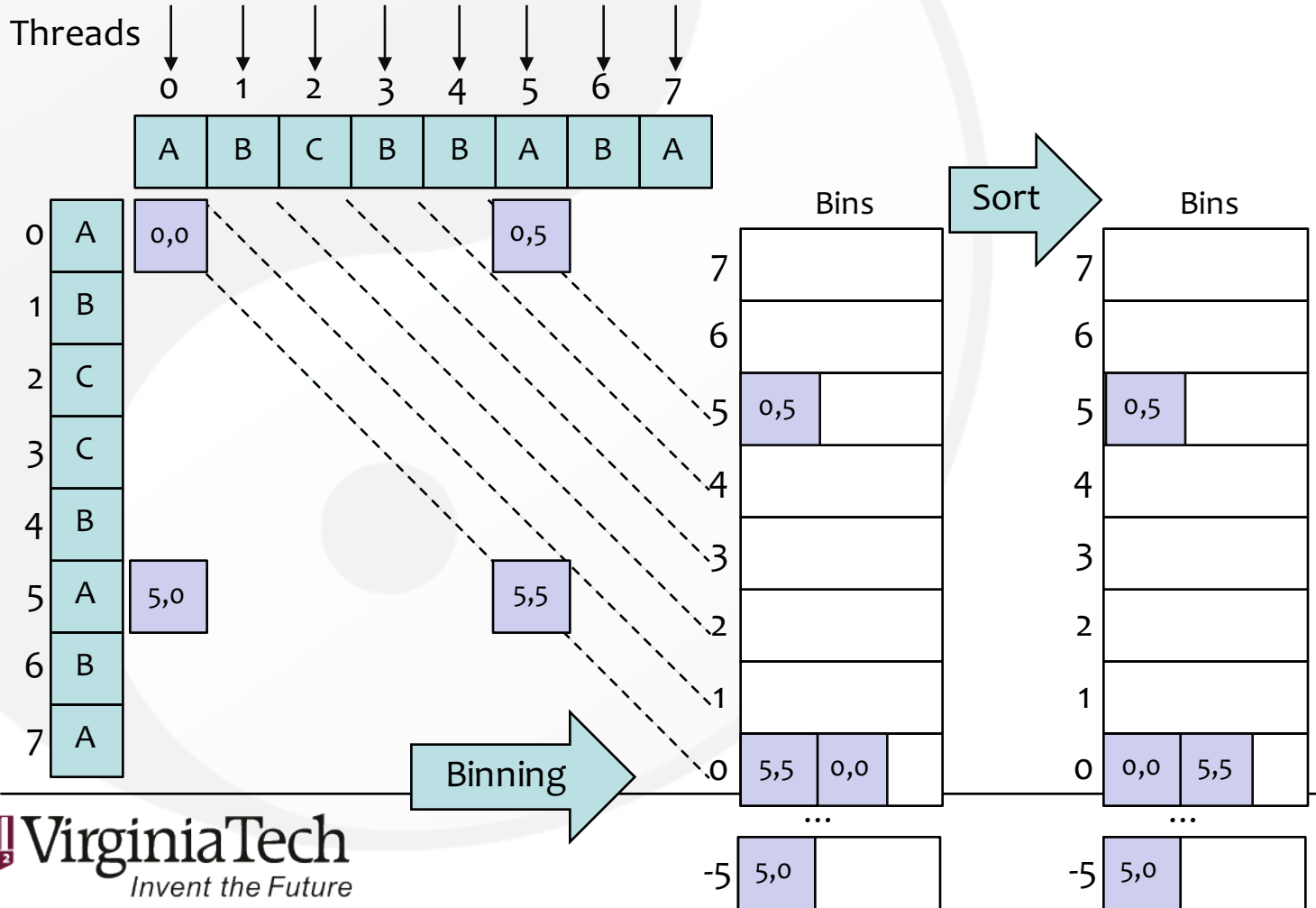
Solution: Memory Access Reordering

- Collect and group hits by diagonal numbers via binning



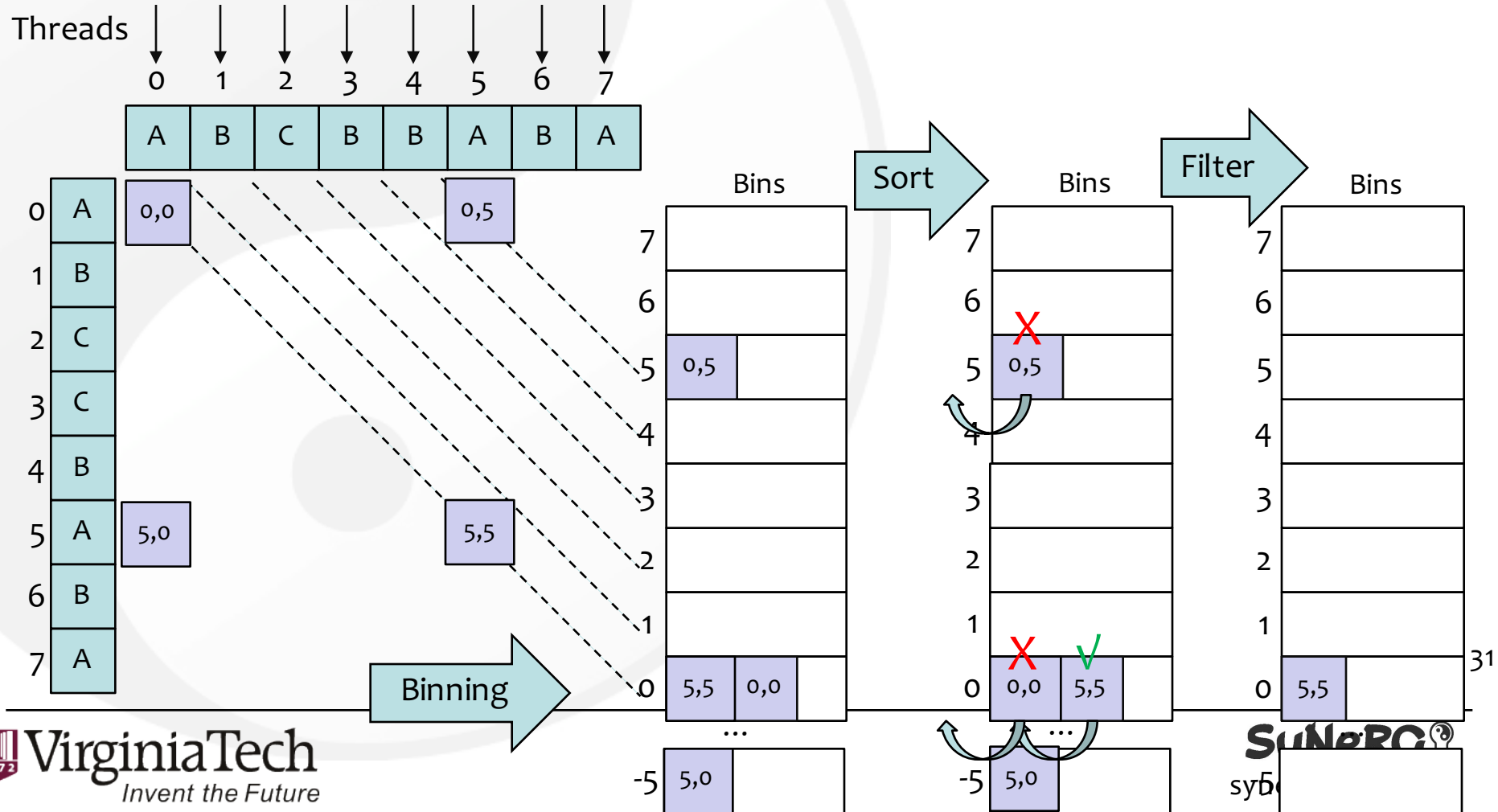
Solution: Memory Access Reordering

- Sort hits by positions in each bin



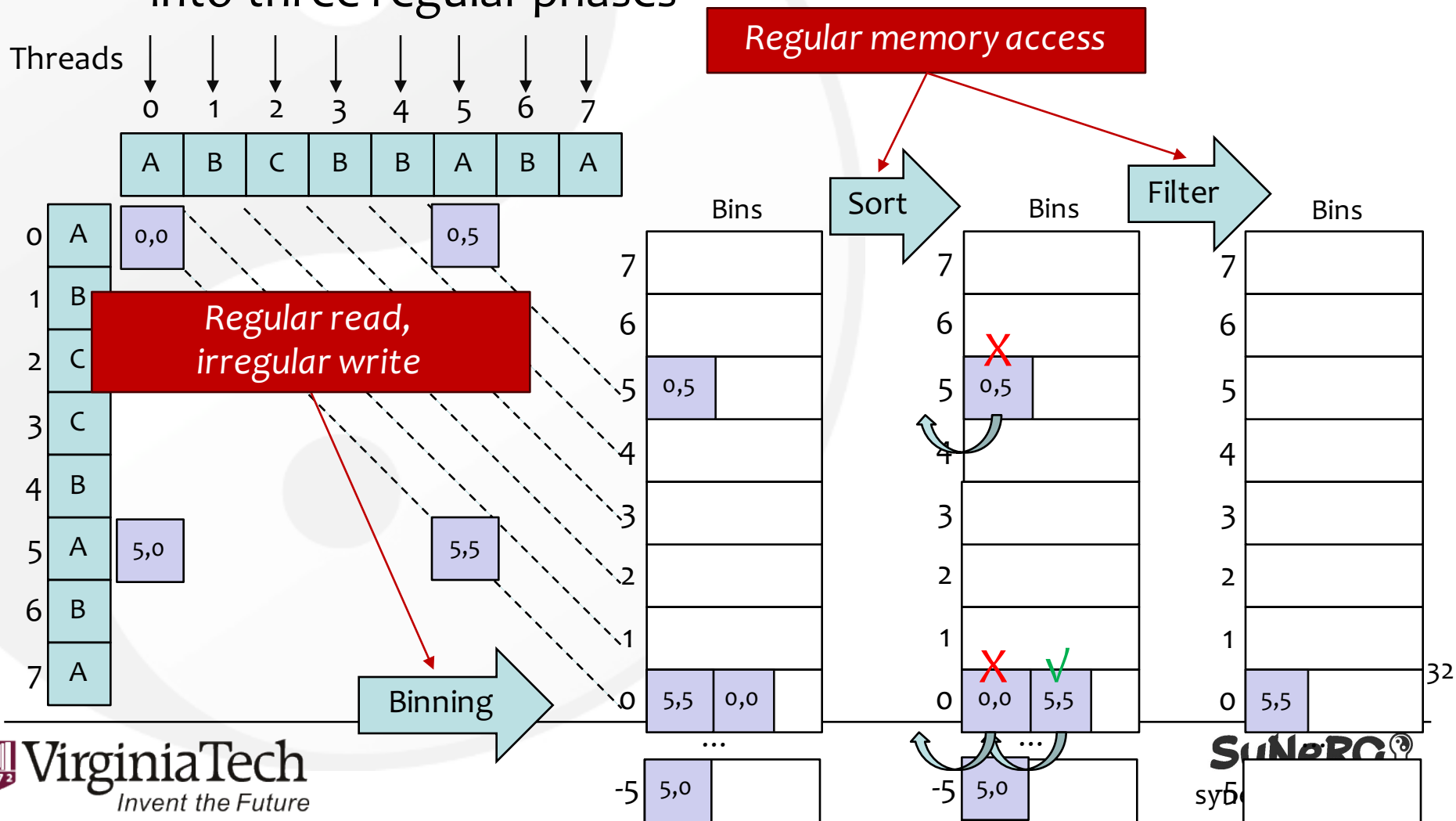
Solution: Memory Access Reordering

- Filter out non-extenable hits by distance of adjacent hits



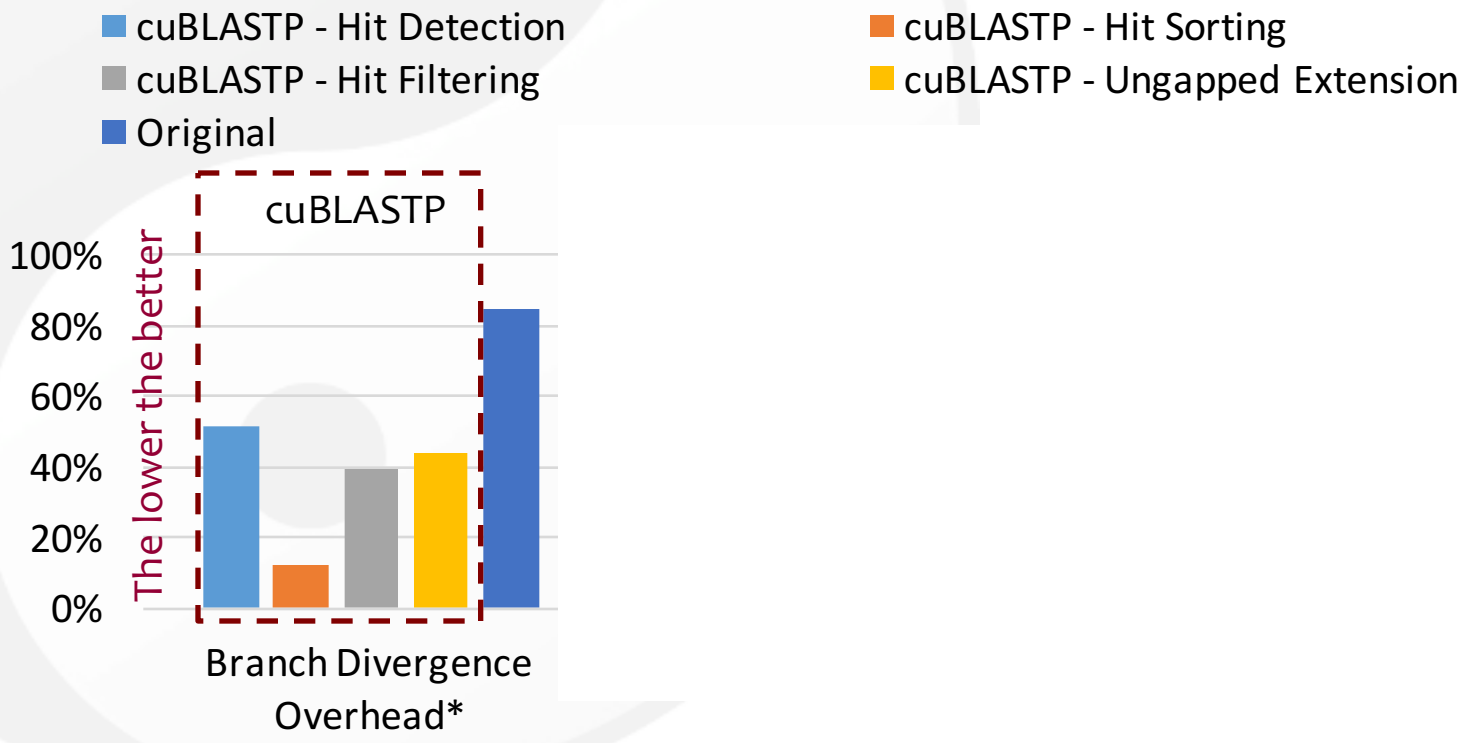
Solution: Memory Access Reordering

- Transform an irregular algorithm (lastHit array method) into three regular phases



Performance Numbers

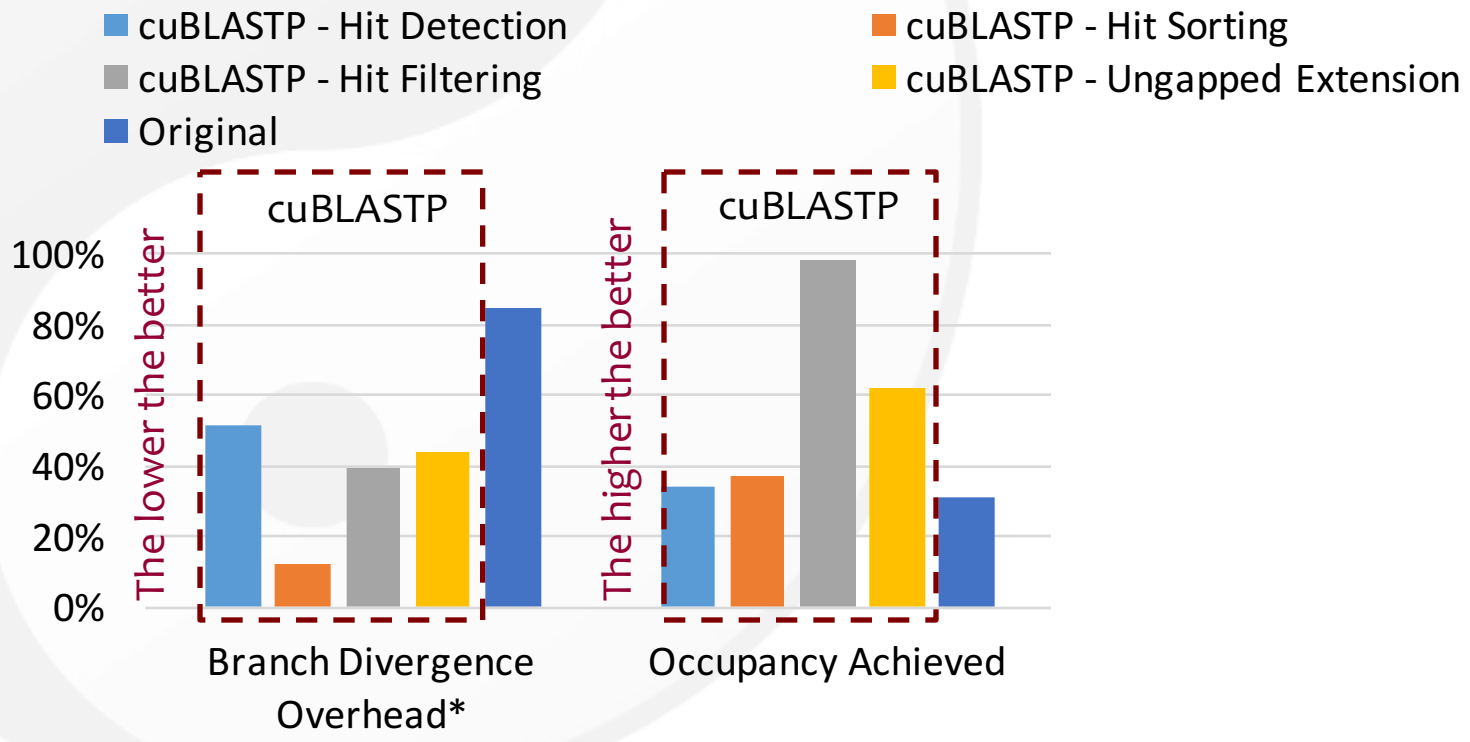
- Compared to original version, cuBLASTP has …
  - Less branch divergence overhead



*: Branch Divergence Overhead: the ratio of divergent execution to total execution

Performance Numbers

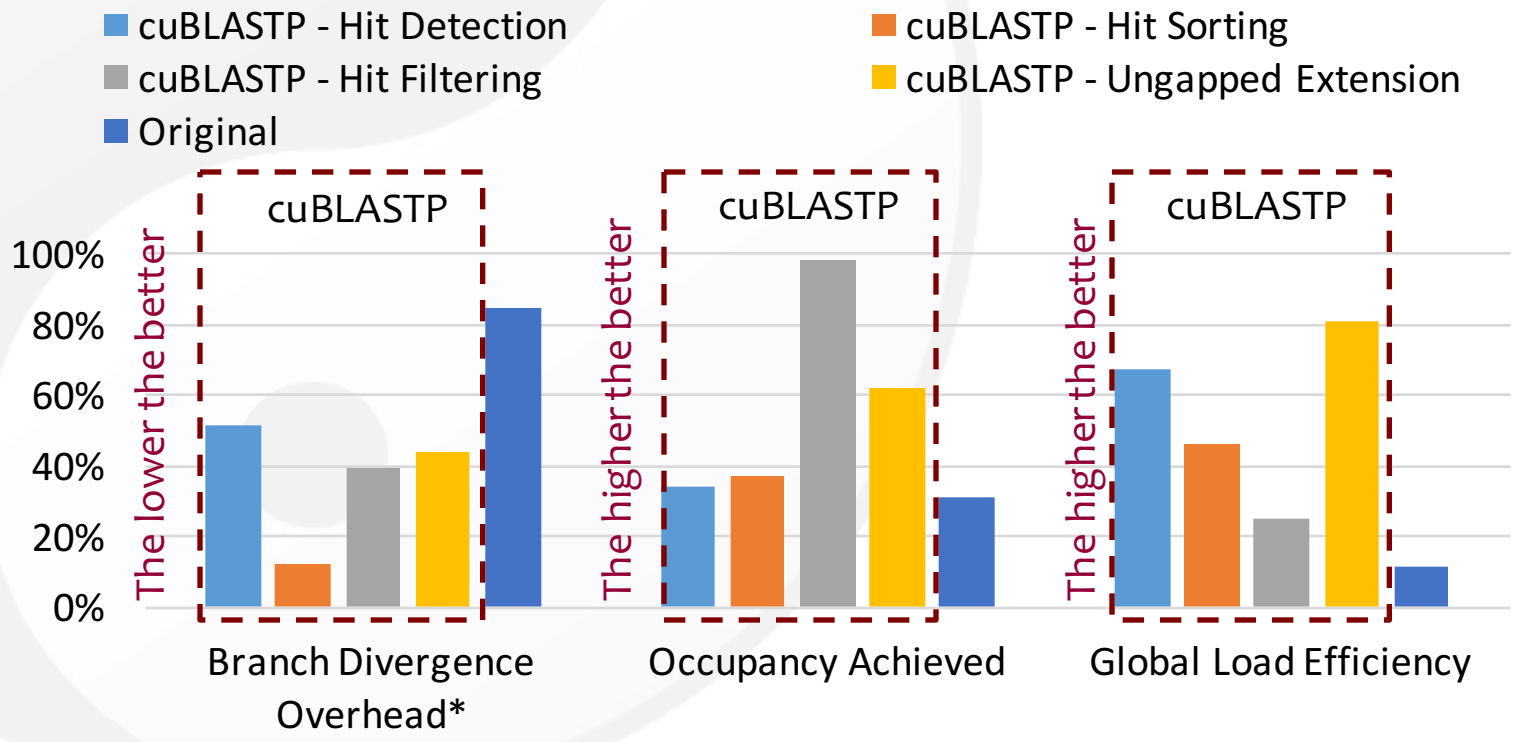
- Compared to original version, cuBLASTP has …
  - Less branch divergence overhead
  - Better occupancy



*: Branch Divergence Overhead: the ratio of divergent execution to total execution

Performance Numbers

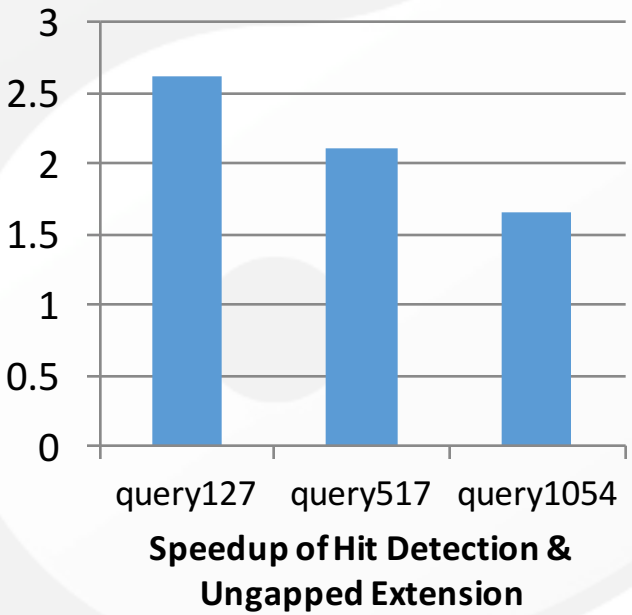
- Compared to original version, cuBLASTP has …
  - Less branch divergence overhead
  - Better occupancy
  - Better load efficiency

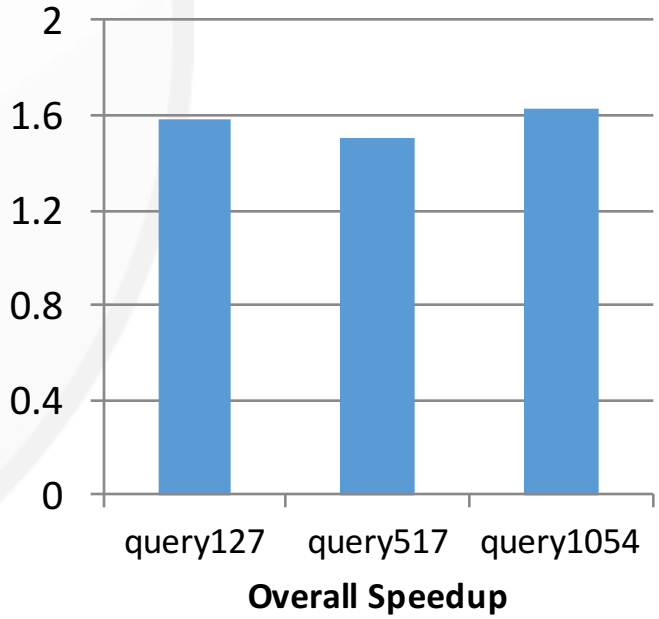


*: Branch Divergence Overhead: the ratio of divergent execution to total execution

Speedup

- With swissprot database, cuBLASTP achieves …
  - Up to 2.6-fold speedup of hit detection and ungapped extension over GPU-BLASTP, which is the fastest GPU-based BLAST
  - Up to 1.6-fold overall speedup over GPU-BLASTP (cuBLASTP has parallelization of the rest of stages)





Conclusion and Future Work

Conclusion

- Mapping irregular algorithms to heterogeneous system is non-trivial, need couples of transformation and adaption on algorithms and data structures
- But these concepts of transformations are propagable
  - Kernel fission, memory access reordering, warp-centric execution, …

Future Work

- Generalize optimizations from cuBLASTP for different algorithms and other heterogeneous systems
- Design a library for irregular algorithms with optimized irregular data structure and operations

Other Research Topics

- Optimizing BWA (Burrows Wheeler Alignment) on multicore CPU and Intel Xeon Phi
- dbblast: database index based BLAST on multicore CPU and Intel Xeon Phi
- Dynamic parallelism on AMD APU (New)
- Optimizing irregular applications for GPU/Xeon Phi clusters (Future)

- Other Research Interests
  - Cloud computing
  - MapReduce with accelerators