GPU Technology Conference (GTC), San Jose, CA, USA, 2014

cuBLASTP: Fine-Grained Parallelization of Protein Sequence Search on a GPU

Jing Zhang[1], Hao Wang[1], Heshan Lin[1], and Wu-chun Feng[1,2]
[1]Department of Computer Science and [2]Department of Electrical and Computer Engineering
Virginia Tech
{zjing14, hwang121, hlin2, wfeng}@vt.edu

synergy.cs.vt.edu

Introduction

- BLAST, short for Basic Local Alignment Search Tool, is a fundamental algorithm in the life sciences that compares biological sequences [1.2].
- With the advent of next-generation sequencing (NGS), the exponential growth of sequence databases is arguably outstripping our ability to analyze the data.
- The previous studies [3.4.5.6] for accelerating BLAST on GPU used coarse-grained parallel approaches, (i.e. one sequence alignment is mapped to only one thread), which are not adapted to GPU architecture and cannot solve the irregular memory access in BLAST.
- Our goal is to propose a faster GPU-BLAST using the fine-grained multithreaded approach.

BLAST Algorithm

- BLAST is a family of algorithms with variants used for different searching alignments, e.g., BLASTp for protein sequence, BLASTn for nucleotide sequence.
- BLAST is a heuristic method that approximates the Smith-Waterman algorithm, searching for similarities between a short query sequence and a large set of database sequences (subject sequences).
- BLAST algorithm locate high scoring short matches (i.e., hits) between the query sequence and the subject sequences, and extend hits to longer alignments. Four stages of BLAST are presented as below (Fig. 1):

1. **Hit detection** identifies high scoring short matches (i.e., hits) with a fixed length between a query sequence and the subject sequences via Deterministic Finite Automaton (DFA) or lookup table.
2. **Ungapped extension** determines whether multiple hits can form the basis of a local alignment without insertions and deletions of residues. Extensions are triggered only if distances of two neighboring hits are within a threshold.
3. **Gapped extension** performs the further extension based on alignments from the previous stage and allows gaps.
4. **Gapped alignment with traceback** re-scores all alignments from the previous stage using a traceback algorithm.

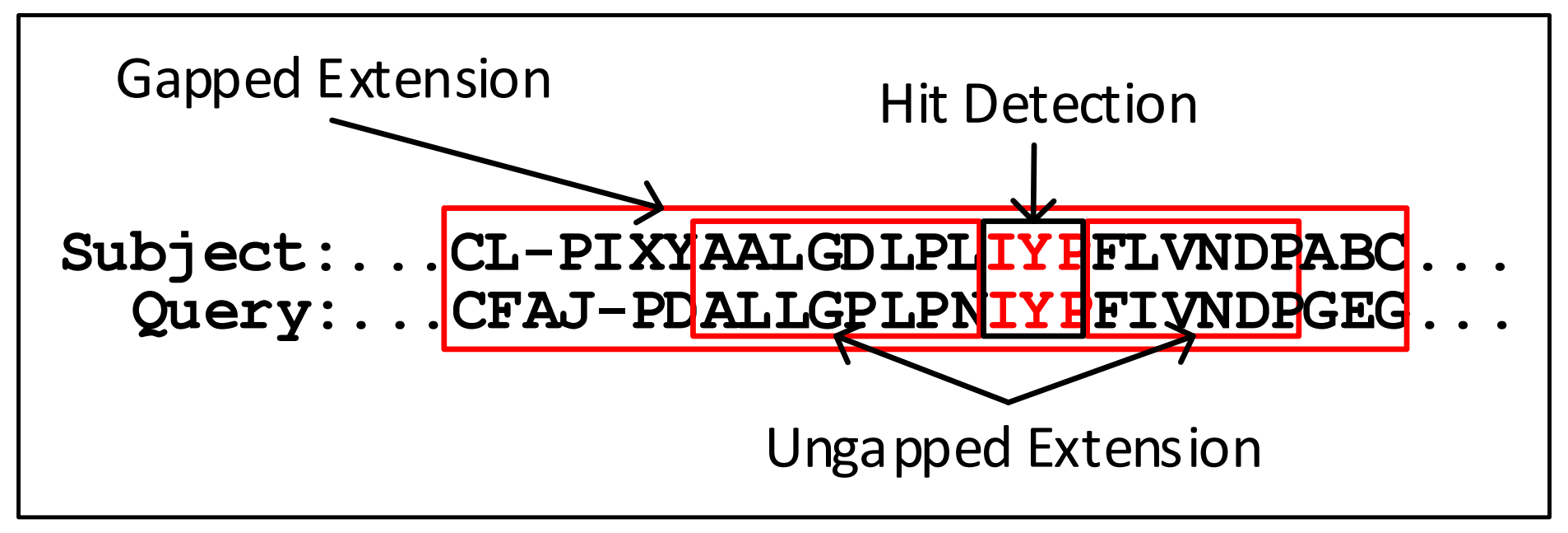
Gapped Extension
Hit Detection
Subject:...CL-PIXYAALGDLPLIYPFLVNDPABC...
Query:...CFAJ-PDALLGPLPNIYPFIVNDPGEG...
Ungapped Extension

Figure 1. First Three Stages of BLAST Execution

- Stage one and two are most time-consuming phases, taking 75% of execution time. Thus, our studies currently focus on these two stages.

Acknowledgment

This work was supported in part by the NSF BIGDATA program via IIS-1247693.

Fine-grained BLASTp

- Challenges of mapping BLAST to GPUs
  - **Irregular memory access** due to the heuristic nature of BLAST
  - **Different memory access patterns** hit detection and ungapped extension – hit detection execution is in column-major order, but ungapped extension is executed in diagonal-major order (Fig. 2).
  - **No straightforward design** to fully utilize the massively parallel computational capability of GPU

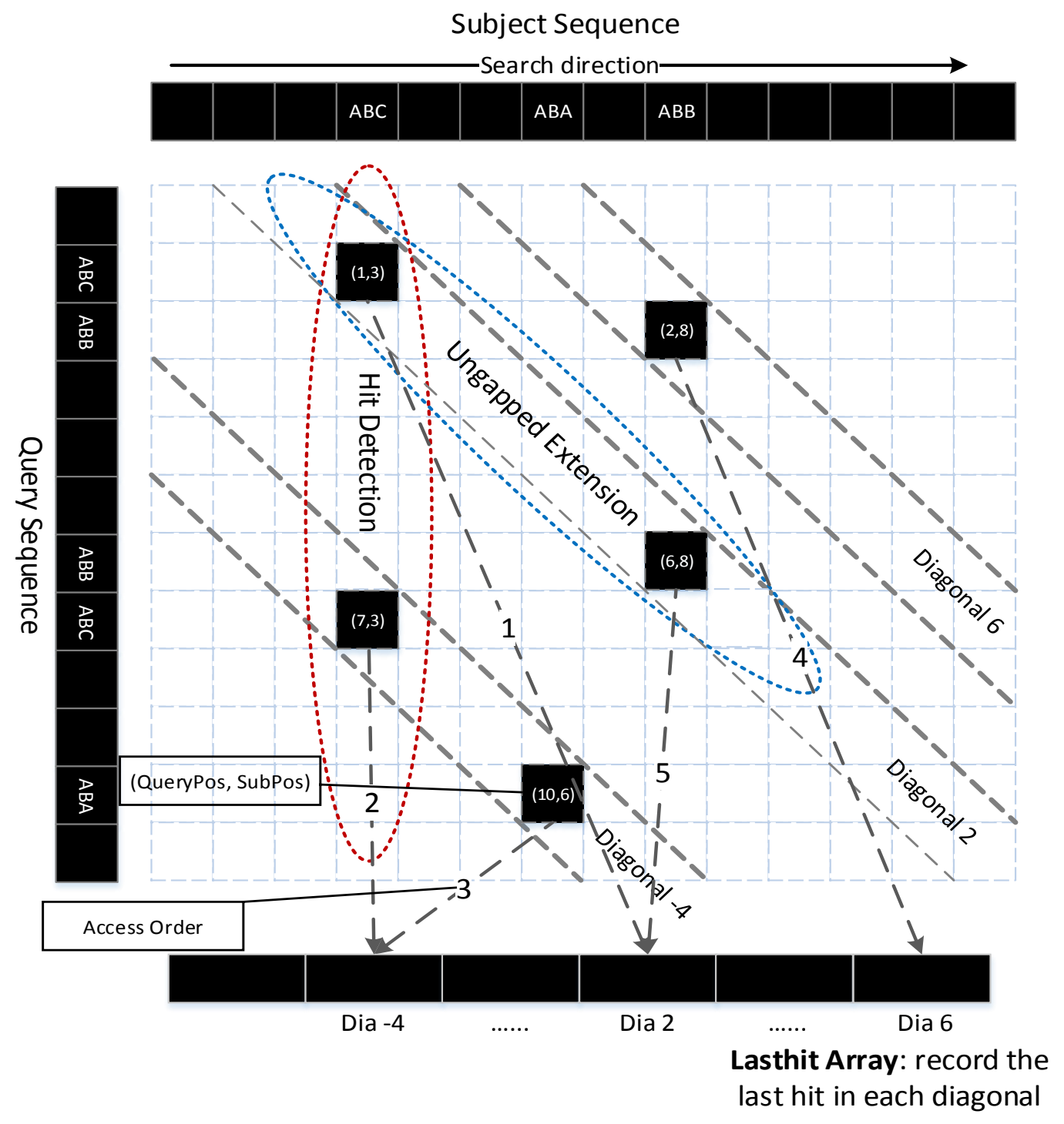

Figure 2. Execution of Hit Detection and Ungapped Extension

- Design of Fine-grained BLASTp (Fig. 3):
  - Fine-grained hit detection with binning
    - Multiple threads are issued in the column-major order for the coalesced access.
    - Binning are introduced to group output by diagonal numbers and sequence numbers.
    - bin number = diagonal number **mod** number of bins.
  - Hit reorganization with sorting and filtering
    - One bin includes hits from different diagonals, leading to hits are out of order. Thus, sorting is introduced to sort hits in each bin with diagonal number.
    - Filtering is also introduce to filter out hits whose distance with neighbors are longer than the threshold.
  - Configurable Fine-grained ungapped extension
    - Diagonal-based extension can avoid redundant hit extension but has more divergence.
    - Hit-based extension has less divergence but more redundant computation.
  - Hierarchical buffering
    - Hierarchical buffering is introduced for core data structure, e.g., DFA.
    - Maintain most reusable part of DFA in the shared memory and remaining part in the constant memory (NVIDIA Fermi GPU) or read-only cache(NVIDIA Kepler GPU).

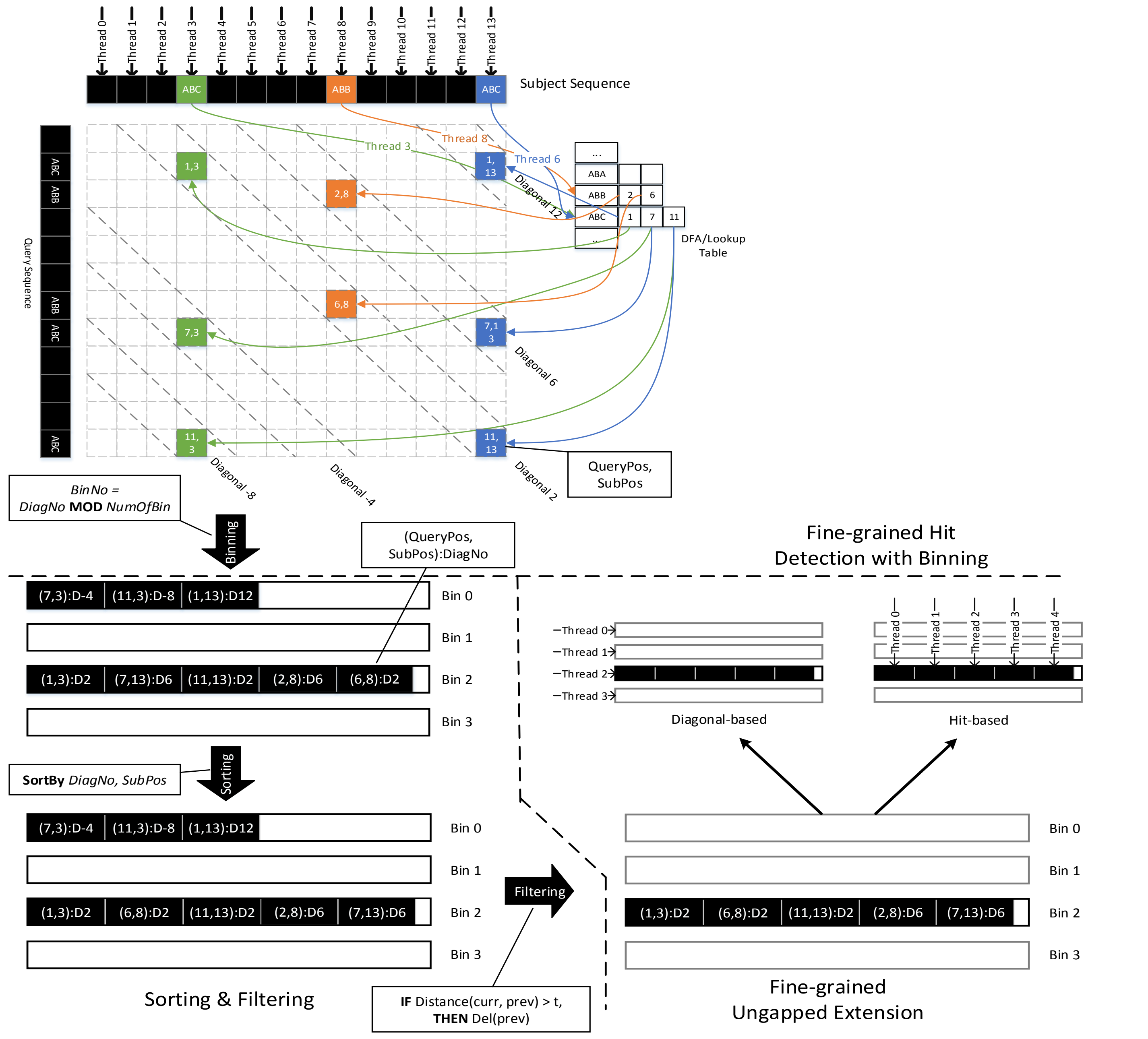

Figure 3. Design of Fine-grained BLASTp for GPU

Results

- For the critical phases, i.e., hit detection and ungapped extension, cuBLASTP has up to 7.8x speedup over a single-thread FSA-BLAST[8], and up to 2.9x speedup over the multithreaded NCBI-BLAST[9] on a quad-core CPU. Moreover, by overlapping the data transfer and the kernel execution, our implementation delivers 4.7x and 3x speedup (since cuBLASTP is based on FSA-BLAST, which is faster than NCBI-BLAST, the overall speedup of cuBLASTP over NCBI-BLAST is higher than the speedup for critical phases) for the overall program execution.
- Compared to GPU-BLAST, which is the existing fastest GPU implementation of BLAST [7], cuBLASTP has up to 2.8x speedup for critical phases, and up to 1.8x speedup for overall performance.

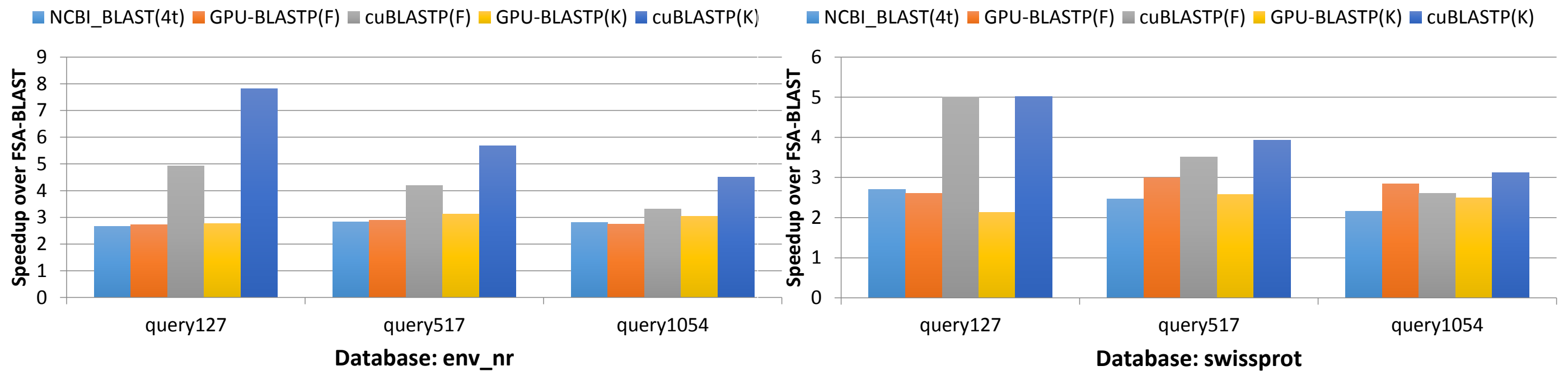

Chart 1. Speedup of cuBLASTP over FSA-BLASTP for critical phases. 4t means the program runs with 4 threads; F means the program runs on NVIDIA Fermi GPU; K means the program runs on NVIDIA Kepler GPU.

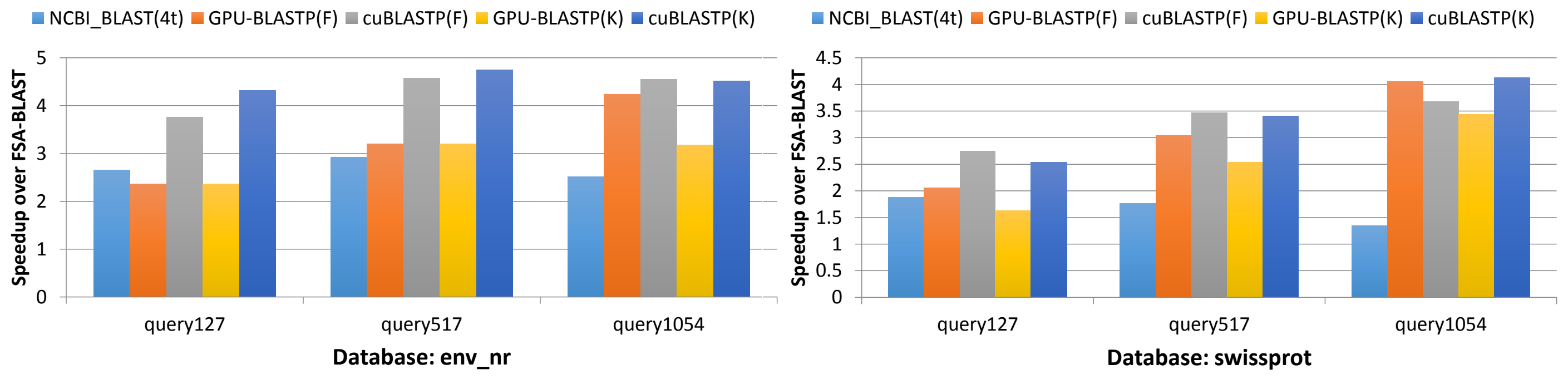

Chart 2. Overall speedup of cuBLASTP over FSA-BLASTP. 4t means the program runs with 4 threads; F means the program runs on NVIDIA Fermi GPU; K means the program runs on NVIDIA Kepler GPU.

Conclusions

- We propose cuBLASTP, an efficient fine-grained BLASTP for GPU using the CUDA programming model.
- In cuBLASTP, we decoupled most time-consuming stages - hit detection and ungapped extension - into separate kernels to apply multiple strategies on different memory access patterns, and added an additional stage: sorting and filtering, to reorganize intermediate results.
- cuBLASTP has up to 7.8x speedup over FSA-BLAST on a single core and 2.9x speedup over NCBI-BLAST on a quad-core CPU for the critical phases, and up to 4.7x speedup and 3x speedup for the overall performance, respectively.
- Compared with GPU-BLAST, the existing fastest BLAST on GPU: GPU-BLAST, cuBLASTP has up to 2.8x speedup for the critical phases and up to 1.8x speedup for the overall performance.

References

1. S. F. Altschul, et al., “Gapped BLAST and PSI-BLAST: a New Generation of Protein Database Search Programs”, *Nucleic Acids Research*, 25, pp. 3389-3402, 1997.
2. Stephen Altschul, et al., “Basic Local Alignment Search Tool”, *Journal of Molecular Biology*, 215:403-410, 1990.
3. W. Liu, et al., “Cuda-blastp: Accelerating BLASTp on CUDA-Enabled Graphics Hardware”. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 8(6):1678–1684, 2011
4. S. Xiao, et al., “Accelerating Protein Sequence Search in a Heterogeneous Computing System”, IPDPS ’11, 2011.
5. P. D. Vouzis, et al., “GPU-BLAST: Using Graphics Processors to Accelerate Protein Sequence Alignment”, Bioinformatics, 27(2):182–188, 2011.
6. C. Ling, et al., “Design And Implementation of a CUDA-compatible GPU-based Core for Gapped BLAST Algorithm”, In P. M. A. Sloot, G. D. van Albada, and J. Dongarra, editors, ICCS, volume 1 of Procedia Computer Science, pages 495–504. Elsevier, 2010.
7. D. Glasco, “An Analysis of BLASTP Implementation on NVIDIA GPUs”, http://biochem218.stanford.edu/Projects%202012/Glasco.pdf
8. FSA-BLAST. Get FSA-BLAST at SourceForge.net. http://sourceforge.net/projects/fsa-blast/
9. NCBI-BLAST. Get NCBI-BLAST at ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/