Accelerating Approximate Pattern Matching with Processing-In-Memory (PIM) and Single-Instruction Multiple-Data (SIMD) Programming

Damla Senol Cali

Zülal Bingöl, Jeremie S. Kim, Rachata Ausavarungnirun, Saugata Ghose, Can Alkan and Onur Mutlu
Problem:
- Bitap algorithm can perform string matching with fast and simple bitwise operations.
- Due to the von Neumann architecture, memory bus between the CPU and the memory is the bottleneck of the high-throughput parallel bitap computations.

Goals:
1) Overcoming memory bus bottleneck of approximate string matching by performing processing-in-memory to exploit the high internal bandwidth available inside new and emerging memory technologies, and
2) Using SIMD programming with Xeon Phi to take advantage of the high amount of bit parallelism available in the bitap algorithm.

Poster SEQ-15

Accelerating Approximate Pattern Matching with Processing-In-Memory (PIM) and Single-Instruction Multiple-Data (SIMD) Programming

Damla Senol Cali1, Ziulal Bingili2, Jerome R. Kin1,3, Rasibata胡春华4,5, Suna Gokcen1, Can Alkan1 and Onur Mutlu1,3

1Carnegie Mellon University, Pittsburgh, PA, USA • 2Bilkent University, Ankara, Turkey • 3ETH Zürich, Zürich, Switzerland

Bitap Algorithm
- Bit algorithm (or Bitap algorithm, or Brzozowski-Gonnet algorithm) [1] can perform exact string matching with fast and simple bitwise operations. We and Marber extended the algorithm [2] in order to perform approximate string matching.
- Step 1 – Programming: For each character in the alphabet (e.g., ASCII) generates a pattern bitmask that stores information about the presence of the corresponding character in the pattern.
- Step 2 – Searching: Compare characters of the text with the pattern by using the preprocessed bitmasks, a set of bitvectors that hold the status of the partial matches and the bitwise operations.

Problem & Our Goal
- The operations used during bitap can be parallel, but high-throughput parallel bitap computation requires a large amount of memory bandwidth that is currently unavailable to the processor.
- Heat mapping is an application of approximate string matching problem, and thus can benefit from existing techniques used to optimise general-purpose string matching.
- Our Goal: Overcoming memory bottleneck of bitap by performing processing-in-memory, to exploit the high internal bandwidth available inside new and emerging memory technologies.

Accelerating Approximate Pattern Matching with Processing-In-Memory (PIM) and Single-Instruction Multiple-Data (SIMD) Programming

Steps:
1. Generate the pattern bitmask, memory layout and initialization.
2. Perform the comparison within the logic module.
3. Check if any memory read and write operations are required.
4. Write back the results.

Results - PIM
- Bitap PIM: Bitap PIM with state-of-the-art read and write operations available in the PIM architecture.
- Number of memory read operations vs. DRAM cycles
- Number of memory write operations vs. DRAM cycles

Results - SIMD
- Bitap SIMD: Bitap SIMD with state-of-the-art read and write operations available in the SIMD architecture.
- Number of memory read operations vs. DRAM cycles
- Number of memory write operations vs. DRAM cycles

Future Work
- Bitap PIM: Extending the current work to set miss rate, in order to decrease the number of operations performed within a SIMD stage. We will also extend our approach in order to generate cache lines.
- Comparing Bitap PIM with state-of-the-art read and write operations available in the PIM architecture.

Steps:
1. Generate the pattern bitmask, memory layout and initialization.
2. Perform the comparison within the logic module.
3. Check if any memory read and write operations are required.
4. Write back the results.

Results - PIM
- Bitap PIM: Bitap PIM with state-of-the-art read and write operations available in the PIM architecture.
- Number of memory read operations vs. DRAM cycles
- Number of memory write operations vs. DRAM cycles

Results - SIMD
- Bitap SIMD: Bitap SIMD with state-of-the-art read and write operations available in the SIMD architecture.
- Number of memory read operations vs. DRAM cycles
- Number of memory write operations vs. DRAM cycles

Future Work
- Bitap PIM: Extending the current work to set miss rate, in order to decrease the number of operations performed within a SIMD stage. We will also extend our approach in order to generate cache lines.
- Comparing Bitap PIM with state-of-the-art read and write operations available in the PIM architecture.

Steps:
1. Generate the pattern bitmask, memory layout and initialization.
2. Perform the comparison within the logic module.
3. Check if any memory read and write operations are required.
4. Write back the results.