Bitap Algorithm

Bitap algorithm (i.e., Shift-Or algorithm, or Baeza-Yates-Gonnet algorithm) [1] can perform exact string matching with fast and simple bitwise operations. Wu and Manber extended the algorithm [2] in order to perform approximate string matching.

- **Step 1 – Preprocessing:** For each character in the alphabet (i.e., A,C,G,T), generate a pattern bitmask that stores information about the presence of the corresponding character in the pattern.
- **Step 2 – Searching:** Compare all 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:**

- The operations used during bitap can be performed in parallel, but high-throughput parallel bitap computation requires a large amount of memory bandwidth that is currently unavailable to the processor.
- Read mapping is an application of approximate string matching problem, and thus can benefit from existing techniques used to optimize 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.
- Using SIMD programming to take advantage of the high amount of parallelism available in the bitap algorithm.

**Problem & Our Goal**

**Processing-in-Memory**

Recent technology that tightly couples memory and logic vertically with very high bandwidth connectors.
- Numerous Through Silicon Vias (TSVs) connecting layers, enable higher bandwidth and lower latency and energy consumption.
- Customizable logic layer enables fast, massively parallel operations on large sets of data, and provides the ability to run these operations near memory to alleviate the memory bottleneck.

**Acceleration of Bitap with PIM**

**Notes:**

- Intel Xeon Phi coprocessor has vector processing unit which utilizes Advanced Vector Extensions (AVX) with an instruction set to perform effective SIMD operations.
- Our current architecture is Knights Corner and it enables usage of 512-bit vectors performing 8 double precision or 16 single precision operations per cycle.
- The recent system runs natively on a single MIC device and the read length must be at most 128 characters.

**Algorithm:**

1. Get 4 pairs of reads and reference segments, prepare bitmasks of each read and assemble them into a vector.
2. Initialize status vectors, start iterating over 4 reference segments simultaneously. While iterating, assign the respective bitvectors of the reads as active and assemble them into a vector. Perform the bitwise operations to get R[0].
3. Integrate the result R(0) with insertion, deletion and substitution status vectors. Deactivate 128b portion of R(0)...R[d] if the respective t ends. Then, perform checking operations on the portion.
4. For d = 1...k,
5. Check the most significant bit of R(0), R(1), ..., R(d). If MSB of R[d] is 0, then there is a match between the text and the pattern with edit distance = d.

**Results - PIM**

- Assuming a row size of 8 kilobytes (65,536 bits) and a cache line size of 64 bytes (512 bits), there are 128 cache lines in a single row. Thus, Memory Latency (ML) = row miss latency + 127*(row bit ops per row, where D = (max # of accelerators) / (actual # of accelerators))

**Results - SIMD**

- We perform the tests with read and reference segment pairs with 1000bp long each. The total number of tested mappings is 3,000,000.

**Future Work**

**Bitap-PIM:**

- Improving the logic module in the logic layer in order to decrease the number of operations performed within a DRAM cycle.
- Providing a backtracking extension in order to generate CIGAR strings.
- Comparing Bitap-PIM with the state-of-the-art read mappers for both short and long reads.

**Bitap-SIMD:**

- Extending the current system to work offload mode for exploiting 4 MIC devices simultaneously.
- Optimizing the expensive adjustment operations (i.e., carry bit operations) to improve the performance of Bitap-SIMD.