Implementing Smith-Waterman Scoring Algorithm on the Resistive Associative Processor

Abstract

1. Introduction

Low operational intensity [?] applications require very high memory bandwidth in order to efficiently exploit machine resources. The underutilization worsens when the input resides in a low-bandwidth external storage and the processor employs a massively parallel architecture. In these cases, the time-dominating operations are I/O. In the case of sparse matrix-vector multiplication (SpMV), read operations dominate since the output may be smaller by orders of magnitude than the input.

Several previous works have addressed the input read-time problem arising from insufficient bandwidth. Stevenson et al. [?] addressed the issue by designing sparse matrix storage based on deduplication memory, reducing memory traffic for SpMV. Gregg and Hazelwood [?] showed that memory transfers between the CPU and GPU are the time-dominating operations, and can easily take 2-50 times longer than kernel execution. Several GPUs were compared, where the best performing one simply had the highest CPU-GPU bandwidth. Lee et al. [?] also addressed the issue of executing SpMV on a GPU stating that when the matrix is large and does not fit into on-die storage, a well optimized kernel is usually bandwidth bound. However, external storage has 2-3 orders of magnitude lower bandwidth, enlarging the gap between I/O and processing times.

From our experimentation with large-scale sparse matrices, read time is typically $2000-3000\times$ longer than multiplication on a many-core processor, leaving the CPU idle for more than 99.9% time. One partial solution to the problem is using faster SSD drives, which cost 5-10 times more and typically provide 3-5 times larger bandwidth. But even SSD drives leave a very large gap between read and processing times.

In this paper we study the benefits of methods for significant reduction of read time and energy by compression, in exchange for longer processing time. The methods can be classified into two categories:

(I) Sparse matrix space-efficient representation formats
(II) General purpose lossless compression supporting parallel decompression

The first class include two space-efficient sparse matrix representation formats [?], CSR Delta Units (CSR-DU) and CSR Values Indirect (CSR-VI). These formats exploit certain characteristics of sparse matrices and allow encoding matrix data in a compact manner. The second class consists of Blocked-ZIP (*BZIP*), which divides the input into constant-sized blocks that can be independently compressed and decompressed. A small-block version, sBZIP2, is employed.

Our main contributions are:

- Analysis of SpMV performance on many-core processors when using only space-efficient encoding formats.
- Application of the roofline model [?] to trading read-time off for processing time.
- Introduction of sBZIP2 to SpMV and roofline model analysis.
- Analysis of performance improvement and energy saving due to adding sBZIP2 compression to SpMV.

The paper is organized as follows. The merits of CSR-DU and CSR-VI formats regarding SpMV are discussed in Section **??**. Section **??** shows execution results for SpMV on a many-core processor when very large matrices (from the University of Florida sparse matrix collection [?]) that are stored in external storage, are encoded in these formats. Using the roofline model, we study architectural tradeoffs in Section **??** and estimate maximum possible speedup. In Section **??**, sBZIP2 is applied to large-matrix SpMV and the roofline model is used to assess the benefits. Potential energy savings are discussed in Section **??**. Section **??** concludes the work and suggests future research.