NCAM: Near-Data Processing for Nearest Neighbor Search

Carlo C. del Mundo, Vincent T. Lee, Luis Ceze, Mark Oskin
{cdel, vlee2, luisceze, oskin}@cs.washington.edu
University of Washington

ABSTRACT

Emerging classes of computer vision applications demand unprecedented computational resources and operate on large amounts of data. In particular, k-nearest neighbors (kNN), a cornerstone algorithm in these applications, incurs significant data movement. To address this challenge, the underlying architecture and memory subsystems must vertically evolve to address memory bandwidth and compute demands. To enable large-scale computer vision, we propose a new class of associative memories called NCAMs which encapsulate logic with memory to accelerate k-nearest neighbors. We estimate that NCAMs can improve the performance of kNN by orders of magnitude over the best off-the-shelf software libraries (e.g., FLANN) and commodity platforms (e.g., GPUs).

1. INTRODUCTION

Matching of visual features is a fundamental building block for an emerging class of visual applications such as content-based search [1], activity recognition [2], augmented reality [3], and 3D reconstruction [4]. At the core of each matching algorithm is k-nearest neighbors, where pairwise distance calculations between the query and candidate points are computed to find the closest neighbors in a metric space. Figure 1 demonstrates kNN for a single node system.

In (1A), a query point is input to the system. Then, candidate points in the DRAM are marshalled through the interconnect (1B). Each core computes the Euclidean distance between the query and each candidate point, then sorts the results by distance (1C). Finally, the top-k closest points are returned (1D).

Individual distance calculations are cheap, but data movement of the candidate points is not; for high dimensional and high cardinality data, this data transfer from memory dominates execution time. At scale, it would cost 500 GB of communication and 150 GFLOP of computation to service

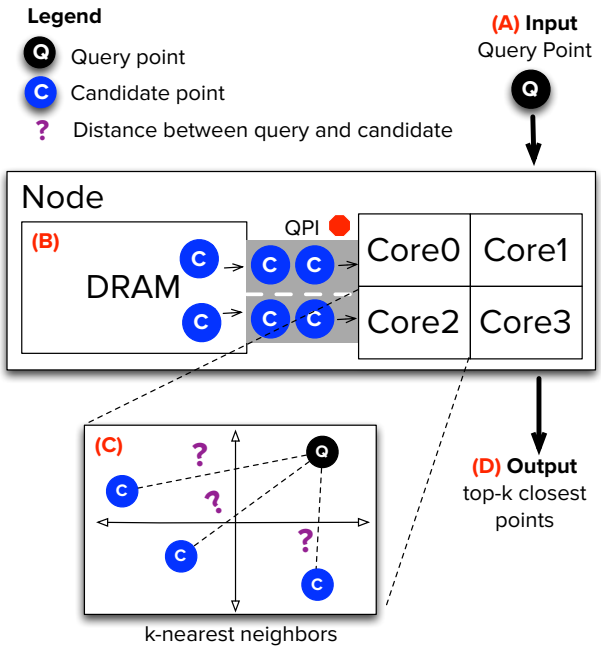

Figure 1: kNN in today's systems. The majority of execution time is in data movement.

one query for one billion candidate points[1].

Solutions such as general purpose GPUs (GPGPUs) and multi-core CPUs are insufficient for kNN. While individual distance calculations are highly parallelizable, these platforms do not provide enough memory bandwidth. PCIe cannot sustain high enough data rates to maximize the compute resource utilization on the GPGPU, and the QuickPath Interconnect (QPI) between the CPU and caches saturates with 4 active cores. In order to solve the problem of memory boundedness, we propose moving computation closer to data by introducing a new class of associative memory called the Nearest Neighbor Content Addressable Memory (NCAM).

2. APPROACH

The NCAM is similar to content-addressable memories in that it addresses data based on the query vector content. However, instead of providing an exact match based on the

[1]These numbers incorporate indexing techniques like kd-trees which already prune the set of candidate points [5].

MEMSYS '15 October 05-08, 2015, Washington DC, DC, USA

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

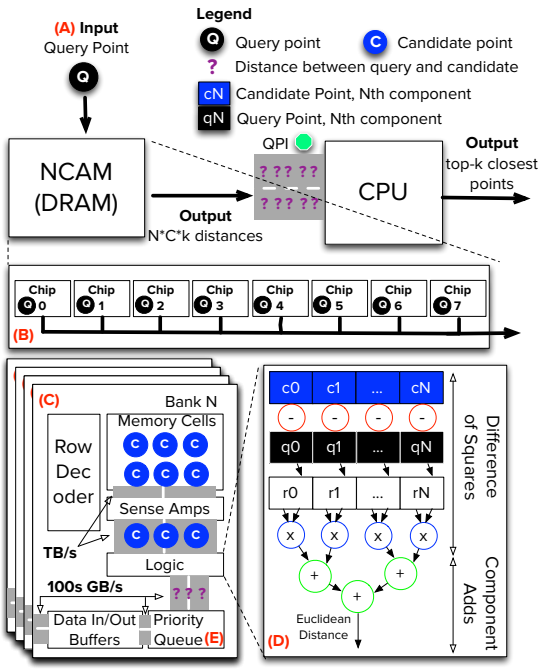

Figure 2: NCAM Architecture. Input: query point. Output: $C \times N \times k$ distance scores. The CPU sorts and returns the top-k results.

query data, NCAMs provide a "nearest neighbor match" for each query vector. This enables computation to be outsourced away from the CPU and closer to the natural resting place of the kNN search data.

For a typical system, we envision the NCAM to be composed of C chips, each of which is composed of N banks. Given an input query point, the NCAM will output $C \times N \times k$ results as shown in Figure 2, where k is the desired number of nearest neighbors. The results generated by the NCAM are the local top-k nearest neighbor results for each of the $C \times N$ banks in the NCAM. The local $C \times N$ sets of results are hierarchically joined and sorted by the CPU, and the final outputs are the top-k nearest neighbors for the query. The difference between Figures 1 and 2 is the embedding of compute logic inside the DRAM banks in Figure 2D. Instead of transferring all of the candidate points to the CPU, the NCAM only transfers the k closest results per bank which alleviates the memory boundedness of kNN.

Chip-level (Figure 2B). At this stage, the query point is broadcast across all chips exploiting chip-level parallelism. Each chip calculates the local $N \times k$ closest neighbors.

Bank-level (Figure 2C). A chip contains N banks. Within each chip, the query is broadcast across banks and stored in the logic layer. After the row decoder selects which candidate points are output for processing, each point is transferred to the sense amplifiers and fed to the logic layer where a distance score is calculcated. The top k results for each of the N banks are output to the chip-level interconnect to the CPU.

Logic Layer (Figure 2D). The logic layer sits adjacent to the sense amplifiers and is responsible for performing high throughput distance calculations. Microarchitecturally, the units of computation in the logic layer can be either floating point, fixed point, or binary operations. State of the art kNN implementations demand floating point precision feature vector values for distance calculations which is expensive to implement in hardware. At the other extreme, computer vision researchers have shown that it is possible to approximate Euclidean distance with binary encodings and Hamming distance calculations [6]. Using Hamming distance calculations greatly simplifies the hardware since it maps to binary operators like XOR. The results of the distance calculation are then reduced to the local k closest neighbors across all memory cells within a bank.

Priority Queue (Figure 2E). In traditional DRAMs, a column decoder takes row values from the sense amplifier and filters data based on a column input. However, since computation is performed within a bank, a new unit, the priority queue, replaces the function of the column decoder. The priority queue buffers individual results and orders them in ascending order. No memory values exit the bank until the local top-k results are computed. Once the bank completes its logical operation, the local top-k results are flushed to the chip-level interconnect.

3. BEYOND DRAM

Prior work has primarily focused on commodity DRAMs as the target for near-data processing in main memory but presents serious challenges when integrating compute into commodity DRAM memory systems. However, new technologies such as die stacked memory and non-volatile RAM (NVRAM) have created new opportunities and challenges for near-data processing [7]. NVRAM variants such as flash, phase change memory, and STT-MRAM have lower internal bandwidths but can achieve higher memory capacity. Similarly, die stacked memory presents new opportunities with high-bandwidth "vertical" busses which can enable heterogeneous integration. By vertically integrating silicon layers on the same chip, compute layers can be optimized for speed and DRAM layers for capacity. It is still an open architectural question as to which memory technology platform best supports near-data processing memory architectures, but we believe that these advances in memory technology will be instrumental in codesigning and enabling next generation computer vision applications such as large-scale content-based search and beyond.

4. REFERENCES

[1] Y. Deng and B. Manjunath, "Content-based Search of Video using Color, Texture, and Motion," in *ICIP '97*.
[2] D. Tran and A. Sorokin, "Human Activity Recognition with Metric Learning," in *ECCV '08*.
[3] R. Azuma, "A Survey of Augmented Reality," in *'97*.
[4] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, "Building Rome in a Day," in *CACM '11*.
[5] M. Muja and D. G. Lowe, "Scalable Nearest Neighbor Algorithms for High Dimensional Data," in *PAMI '14*.
[6] Y. Gong and S. Lazebnik, "Iterative Quantization: A Procrustean Approach to Learning Binary Codes," in *CVPR '11*.
[7] K. Strauss and D. Burger, "What the Future Holds for Solid-State Memory," in *IEEE Computer 2014*.