In-Place Computing: Scaling to 1M Similarity Searches per Second

Introduction

This paper presents a patented Associative Processing Unit (APU) that changes the concept of computing from serial data processing—where data is moved back and forth between the processor and memory—to massive parallel data processing, compute, and search in-place directly in the memory array.

Current solutions access data by providing an address to find its location in memory. After the data is located, it is sent across an I/O bus between a processor and memory. This is very inefficient from both a performance and power perspective. A better approach would be to access the data by content and to process it directly in place in the memory array without having to cross the I/O.

GSI's APU removes the bottleneck at the I/O between the processor and memory, resulting in significant performance-over-power ratio improvement compared to conventional methods that use CPU and GPGPU (General Purpose GPU) along with DRAM.

The APU architecture is particularly well suited to search and compute applications. Two applications, recommender systems and data mining, use a particular type of search called similarity search to make predictions, and they will be presented in this paper. GSI's APU provides over a 500x performance-over-power ratio improvement compared to current similarity search solutions.

In-Memory Compute

The APU turns every bit line in a memory into a processor, allowing the memory array to become a massively parallel processor. As seen in **Figure 1a** below, each bit line (that runs vertically in the memory columns from the top to the bottom of the chip) logically computes the NOR of each selected bit (Read) on that bit line. The result of that NOR operation can be written to another location (Write). As seen in **Figure 1b**, this previous output can be used as an input to another NOR operation.

A NOR gate is a universal logic gate that can be used to make any other logic gate. By repeating the process mentioned above and creating a nested series of NOR operations, any logic gate, and thus any arithmetic function, can be generated. This turns each bit line into a processor. GSI's APU contains millions of bit lines and thus millions of parallel processors.

Figure 1a—NOR Operation

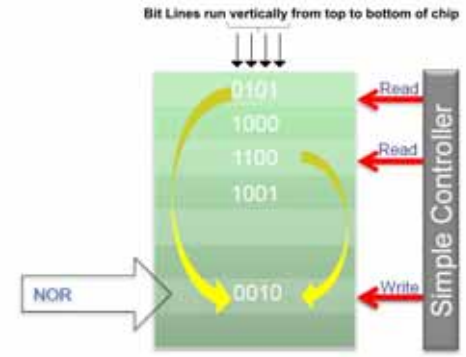
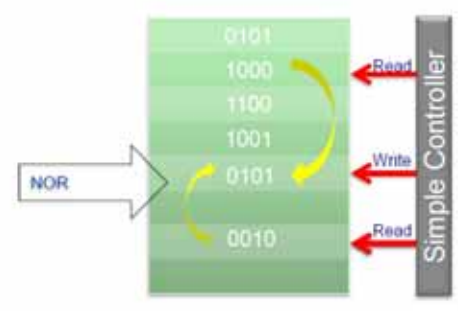


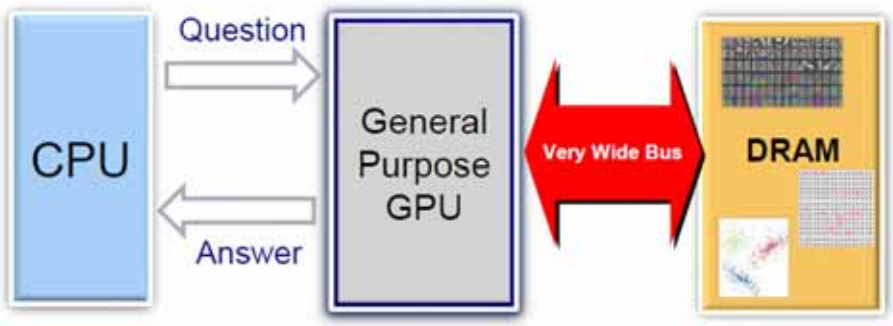
Figure 1b—Nested NOR Operation



Removing the I/O Bottleneck

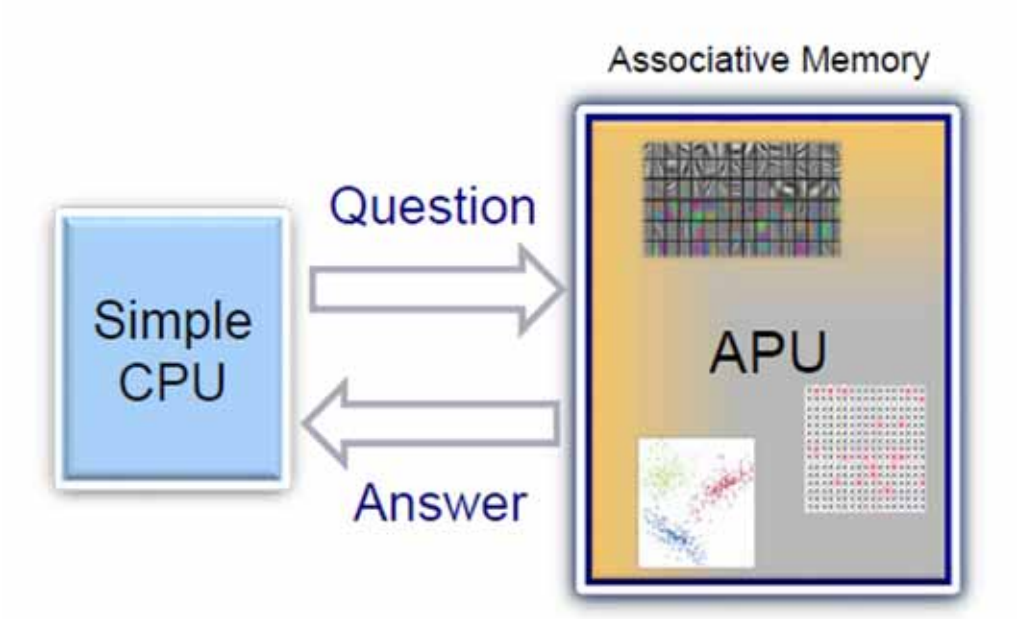
As seen in **Figure 2a**, current solutions rely on serial data processing—where data is moved back and forth between the processor and memory. This creates a bottleneck at the I/O between the processor and memory.

Figure 2a—Current Solution



As seen in **Figure 2b**, the APU performs compute in-place directly in the memory array, thus removing the bottleneck at the I/O between the processor and memory. It does this by leveraging the parallel nature of memory and turning the millions of bit lines in the array into millions of highly-granular, parallel processors.

Figure 2b—APU Solution



Prediction in Recommender Systems and Data Mining

Rcommender Systems

People are being presented with an increasing amount of choices, such as items to buy, who to connect with on social media, news stories to read, and movies to watch. In the process of turning these choices into action, billions of searches are generated daily for large e-commerce and social media companies—translating into 10s of thousands of searches/sec that need to be handled real time. Recommender systems play a key role in filtering these searches by providing the most relevant choices.

Recommender systems predict ratings for each user, such as movie ratings or news article ratings. They are a form of machine learning—they learn user preferences to predict ratings that a user would give to an item and generate recommendations based on those predicted ratings.

Current recommender system solutions have scalability issues. Some large e-commerce and social network companies have millions of customers and billions of items. Providing recommendations in these cases is computationally expensive and scalability is an issue because the performance varies directly with the number of items and users. This problem will only get worse over time as the number of items and users continues to grow.

Data Mining

Another application where prediction is heavily used is data mining. A common data mining task is to predict an unknown dependent variable given the values of independent variables. For example, one could predict the value of a house based on location, square footage, lot size, number of rooms, etc.

Prediction Based on Similarity

Recommender systems and data mining applications use similarity search to help make predictions. An algorithm known as k-Nearest Neighbor (k-NN) is one of the similarity search algorithms commonly used by these applications.

In the k-NN algorithm, items are represented by feature vectors and the “distance” between vectors is used to determine similarity. A popular distance metric is cosine similarity.

Cosine similarity between two vectors measures the cosine of the angle between the vectors to determine similarity. The cosine of 0° is 1, and it is less than 1 for other angles. Vectors are seen as more “similar” as the angle between them approaches zero.

To determine the most similar items to a particular item, the k number of items with the smallest cosine distance are chosen. The item with the smallest cosine distance receives rank 1, the next 2, up to rank k.

Recommender systems use the rating patterns of the k-Nearest Neighbors to predict ratings for a given user. Those predicted ratings are used to make recommendations.

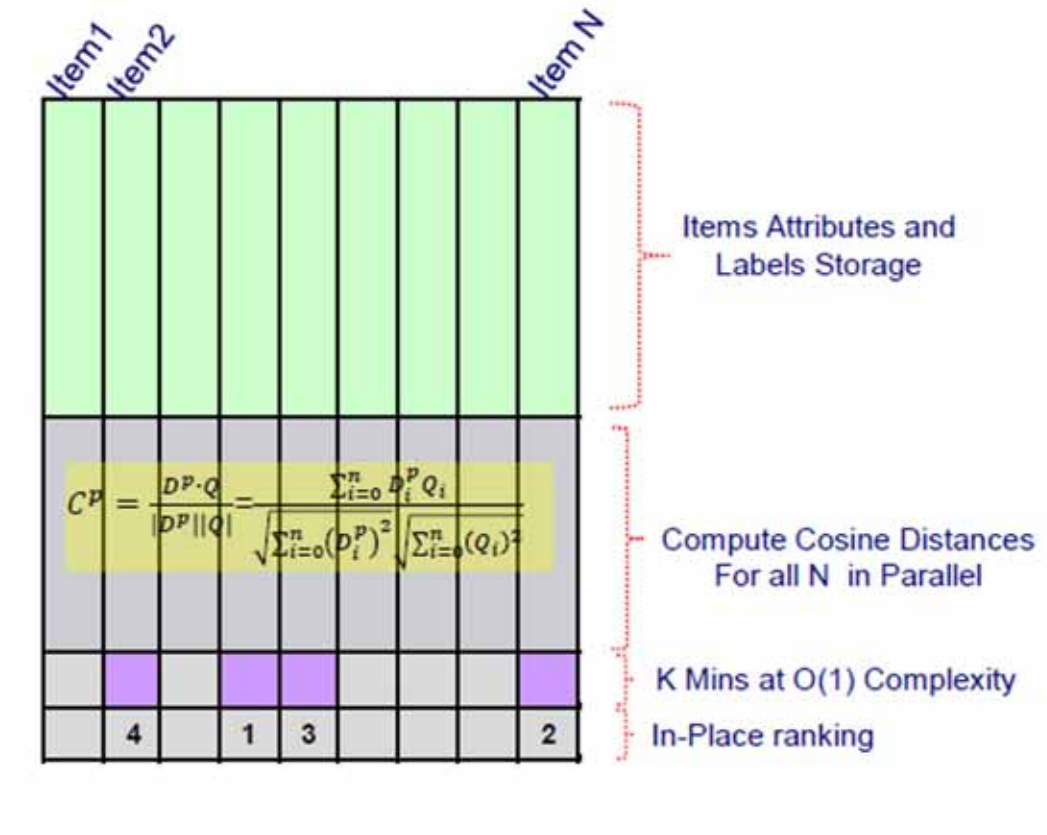
In data mining, one way of predicting a variable for an item is to take the average of that variable for the k-Nearest Neighbors of the item. An alternative is to use a weighted average, with higher weights being assigned to closer neighbors.

One key issue that is common to recommender systems and data mining applications that should be noted is sparse matrix multiplication. A sparse matrix is one where most of the items in the matrix are zero. A detailed discussion of sparse matrix multiplication is beyond the scope of this paper, but it should be noted that GSI’s APU provides roughly 1000x performance advantage over GPGPU-based solutions for sparse matrix multiplication (on the order of TFLOPS vs GFLOPS).

Scaling to 1M Similarity Searches per Second

As mentioned earlier, current similarity search solutions have scalability issues because their performance varies directly with the number of users and items. The APU addresses this scalability issue by computing the cosine similarity for all items directly in place in parallel, as seen in **Figure 3**. As mentioned previously, the APU also finds the nearest neighbors and ranks them. Because each item in the database is independent of the others, the cosine similarity algorithm is well suited to parallel architectures, such as that of the APU.

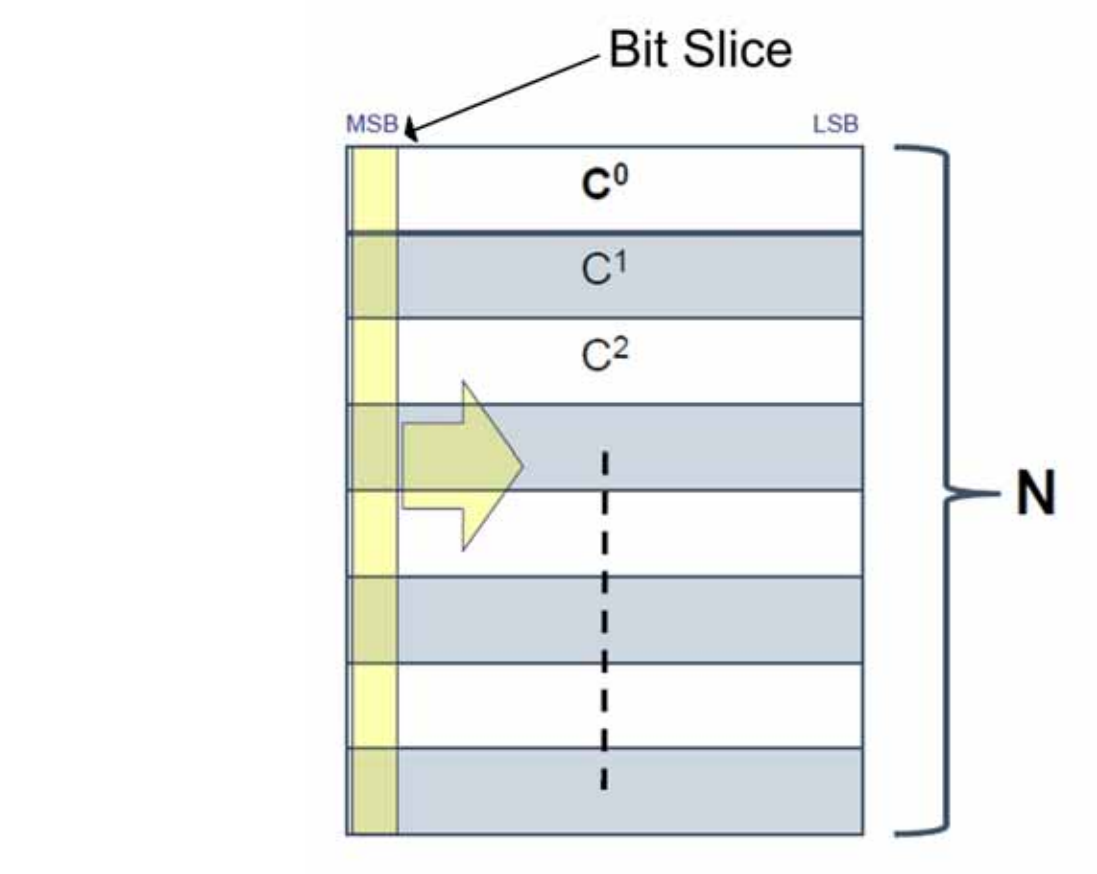
Figure 3—Computing Cosine Distance in Parallel



Once the cosine distances are computed, the k-Nearest Neighbors must be determined.

Figure 4 on the following page shows how the APU determines the nearest neighbors by searching all cosine distances (C^0, C^1, C^2 …) in parallel to determine the minimum values.

Figure 4—Parallel Computation of the k Nearest Neighbors in Constant Time



As seen in **Figure 4**, the algorithm takes bit slices across the entire database. Because the algorithm is looking for minimum distances, it searches for rows that have a "0" value for the current bit. The algorithm iterates from the Most Significant Bit (MSB) to the Least Significant Bit (LSB) of each cosine distance. This is done in constant time, providing O(1) performance, irrespective of the number of cosine distances being searched. The performance is dictated by the number of bits in the cosine distance value; it is independent of the number of values being searched because they are all searched in parallel.

The APU can compute all cosine distances and find the k-Nearest Neighbors in less than 1ms. This represents more than a 500x performance-over-power ratio improvement compared to current similarity search solutions. Additionally, by leveraging clustering algorithms such as k-Means, the APU can scale to 1M similarity searches per second. This is allows large e-commerce and social network companies to scale well beyond the 10s of thousands of searches/sec that they are currently seeing.