Virtually Cool Ternary Content Addressable Memory

Suparna Bhattacharya
IBM Linux Technology Center, Indian Institute of Science

K. Gopinath
Indian Institute of Science

Abstract

Fast content addressable data access mechanisms have compelling applications in today's systems. Many of these exploit the powerful wildcard matching capabilities provided by ternary content addressable memories. For example, TCAM based implementations of important algorithms in data mining been developed in recent years; these achieve an an order of magnitude speedup over prevalent techniques. However, large hardware TCAMs are still prohibitively expensive in terms of power consumption and cost per bit. This has been a barrier to extending their exploitation beyond niche and special purpose systems.

We propose an approach to overcome this barrier by extending the traditional virtual memory hierarchy to scale up the user visible capacity of TCAMs while mitigating the power consumption overhead. By exploiting the notion of content locality (as opposed to spatial locality), we devise a novel combination of software and hardware techniques to provide an abstraction of a large virtual ternary content addressable space.

In the long run, such abstractions enable applications to disassociate considerations of spatial locality and contiguity from the way data is referenced. If successful, ideas for making content addressability a first class abstraction in computing systems can open up a radical shift in the way applications are optimized for memory locality, just as storage class memories are soon expected to shift away from the way in which applications are typically optimized for disk access locality.

1 Introduction

Associative lookup structures lie at the heart of many computing problems. Content addressable memories provide fast constant time lookups over a large array of data (content keys) using dedicated parallel match circuitry [10]. A ternary content addressable memory (TCAM) enables compact representations by allowing entries to be stored (and queried) so that any bit position can be a 0, 1 or *, a don't care (wildcard) bit that can match both 0 and 1 [1].

The most widespread exploitation of this technology occurs in high performance routers, for route lookup, access control and packet classification. Examples of other applications include database acceleration [3], frequent items in data streams [4] and several algorithms that use TCAM as an underlying primitive. TCAM based implementations of fundamental techniques in pattern matching, machine learning and data mining, such as regular expression matching [9], nearest neighbor search [11] and subset queries using ternary bloom filters [6], are a few examples that have been developed in recent years. These techniques have diverse real world applications in areas like information retrieval, image search, genomics, proteomics, intrusion detection, and fraud surveillance. What makes the TCAM abstraction such a powerful primitive for many of these applications is the ability to *simultaneously search through a large number of subspaces of a higher dimensional space in one shot*. For example, each subspace can be compactly represented as one (or a few) TCAM entries using the don't care bits to cover ranges that constitute it.

Similarity search and nearest neighbor search are widely used in many algorithms. Locality sensitive hashing is an important technique that maps high dimension feature vectors to lower dimension ones while keeping similar content together. This can be done in a preprocessing step where data points are hashed to a number of buckets. To perform a similarity search, a query is hashed using the same locality sensitive hashing scheme and the similarity search is performed on the data points retrieved from the bucket corresponding to the query hash. However, streaming algorithms that are becoming common still find the "non-parallel" similarity search in the last part slow. A recent technique[11] uses a modified version of locality sensitive hashing to hash data to

ternary values, enabling compact TCAM representations and quick similarity searches for various classification problems.

TCAMs may also be useful in many state space exploration problems (such as those encountered in verification) where many states can be combined into a single TCAM entry using Bloom filters, enabling a fast search for previously visited states or error states.

The usage of content based lookup and similarity matching in systems infrastructure is also growing. For example, de-duplication techniques for cache [5], memory [2], IO [8] and storage data all exploit some form of content lookup or comparison scheme. [6] shows how ternary bloom filters can be used to achieve an order of magnitude throughput improvement over current techniques in high speed multiple string matching (MSM) problems, a key component in data-deduplication, sequence alignment and intrusion detection techniques. Hardware based range caches [12] have been proposed for efficient state tracking to make intensive dynamic analysis of programs viable.

Despite all of these developments, hardware TCAMs have not made their way into mainstream computing[1]. This is mainly because the power of TCAMs comes at the price of high cost and energy consumption. A TCAM uses about 20x more dynamic power per bit than an SRAM [1, 6] (the overhead of parallel lookups). As a result practical applications have been mostly limited to niche areas where the tradeoff can be justified for a TCAM size which fits the requirements, e.g. in high speed packet classification (with 50x speedup). Both the delay and energy consumed per access increase with the size (width and number of entries) of a TCAM [1]. This restricts the extent to which the use of TCAMs can be scaled so as to be viable in broader setting.

We think that there may be a way to break this barrier. Most of the power consumed by a TCAM is effectively wasted in mismatches[2]. While this observation has prompted many TCAM power optimizations [10], hardware based techniques tend to have limited flexibility in adapting to actual usage scenarios. Perhaps, this is an area where operating systems can help (with architectural support). There is a well-established precedent for solving such problems - consider the invention of the memory hierarchy and virtual memory management(VMM). Can such mechanisms be extended to scale up the applicability of content addressable primitives?

In this paper we explore this possibility and raise some related questions. What if content addressability were to be made a first class abstraction in computing systems? Is it possible to design this abstraction in a way that subsumes the prevalent location based addressing model of data access? What important technical considerations could determine its feasibility? What opportunities might be enabled by this new infrastructure? How would it impact the way applications are optimized for locality?

[1] even though they have been integrated with NPUs for years

[2] all matchlines are pre-charged before a search; lines that do not match the search word are discharged, leaving only the lines that match in high state

2 Content Addressable VMM (CAVMM)

Let us see how the basic concept of ternary content addressable memories may be extended to a generalized content based memory hierarchy by combining the benefits of TCAMs and VMM principles. This enables (multiple) applications to efficiently exploit the power of the ternary search abstraction at a larger scale than that achievable with hardware TCAM alone.

The proposed hierarchy includes a hardware TCAM based cache and multiple levels of ternary content addressable stores (TCASs). These stores may be implemented in hardware or software with different performance vs efficiency tradeoffs, e.g high performance at levels closer to the processor and high capacity at levels that are further away. Content (search key) words present in these stores are associated with references to data in a traditional (hierarchical) location addressable store (LAS). This data is returned as the result of a content addressed access (search) along with the key.

One of the novel features of this architecture is that traditional notions of pages and blocks are replaced by alternate notions like content subspace pages and content blocks which operate on a content key space (i.e. the domain of the content word) instead of a location based address space. The hardware support required may be implemented using a content addressable memory management unit (CAMMU).

The design must be capable of exploiting the benefits of spatial locality and location based addressing where preferable, while enabling the full power of content addressability at a system level. This is achieved using content mapping schemes that preserve location based addressing where desired (e.g. as a default compatibility mode or where it is more efficient).

Fig 1 illustrates how a content addressable virtual memory hierarchy might be organized. We focus on one possible implementation approach to make this example concrete and highlight a few essential details. Many potential variations or extensions may be explored using similar ideas. Fig 2 depicts a sample view from a snapshot of the virtual content addressable space and its representation in the CAVMM hierarchy. We assume an implementation with two-levels of TCAS (in addition to the content based cache) where the Level 1 store is implemented using hardware TCAM and the Level 2 store (de-

scribed in more detail later) uses a software based implementation with DRAM as the underlying physical store.

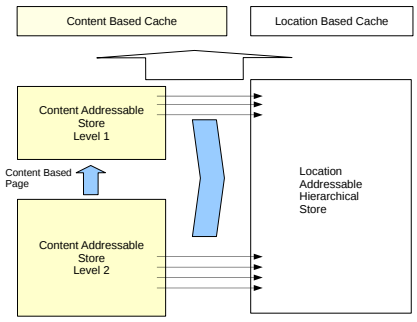

Figure 1: Content Addressable VMM Example

Location Addressable Hierarchical Store: Traditional memory store where data is referenced by its memory address location. Addressing could be physical or virtual, and the hierarchy could span multiple levels of memory and secondary storage.
Location Based Cache: Caches data from the LAS.
Content Addressable Store: Associates ternary content words with data references in a LAS[3]. When presented with a ternary search word, matching content words and the corresponding data referenced are retrieved. Since multiple entries can match, a stream of multiple results may be returned.
Content Based Cache: Transparently caches content word to data associations. The corresponding data is cached in the location based cache. Since multiple matches are possible there could be multiple entries for the same content word. Cache prefetching is content locality based rather than address locality based.

2.1 Content Paging and Content Blocks

The mapping from a content key to a physical location can be as fine grained as a single memory word, effectively dissociating spatial contiguity from content locality. This breaks the traditional concept of pages as used in virtual memory implementations.

A **Content Subspace Page** is the result of a search matching a lower dimensional subspace of the content key space, i.e. a collection of entries (in a TCAS) whose content key word has a value that falls within the subspace. For example, the content key space may be broken up into uniform subspaces of size 2^k formed by setting the least significant k bits of the search word as don't care when retrieving a content subspace page. The entries belonging to a content subspace page could be distributed across the TCAS with no physical contiguity or ordering implied. They form a logical representation of a page rather than a real memory page. Notice that a content subspace page typically has holes within it (i.e it may be sparse). As a result, the physical size (number of TCAM entries) is usually smaller than a real memory page. On the other hand, since multiple entries may match the same content word, it is even possible for the physical size to be larger than a real memory page. In general, it is not necessary to use only the least significant bits or even contiguous bits when defining a content subspace page, i.e. the subspace could range over any specified dimension(s). Further, it is even possible for a single ternary entry to straddle more than one content subspace page.

A **Content Block** is a group of content words in a TCAS that contain consecutive values in the content key space and reference data at consecutive location units in the location based address space. These entries can be compressed into a single content block entry if the range of content words can be represented as a ternary word. This feature also enables *location based addressing to be trivially supported* with minimal overhead by using a single content word entry (cached in the content cache) that represents a large ternary content block covering the entire location address space.

2.2 Level 2 TCAS

How might a level 2 TCAS be implemented by an OS using an underlying DRAM store? A single ternary content word is represented as a combination of a binary content word and a binary wildcard mask. For each ternion in the original content word that is set to "*" (or don't care), the corresponding bit in the wildcard mask is set to 1 and other bits are set to 0. If the unit of transfer between the level 1 and level 2 store is a content subspace page, it is sufficient to track these content words at the granularity of such a content subspace. Regular memory based data structures e.g. hash tables or integer radix trees may be used to maintain key-value and range-value mappings in DRAM. Instead of creating these structures, however, we devise a simpler scheme that takes advantage of the hardware TCAM at Level 1 (making physical locality or size of content pages irrelevant for paging complexity). This works as follows:

When all entries corresponding to a content subspace page are collected and paged out[4] from Level 1 to Level 2, a single special ternary content word entry is created in the Level 1 TCAM to refer to the location of the content page in the Level 2 store. The same principle may be extended to create a content page container subspace (by paging out content subspace pages that fall within a content page container subspace). Further, as we noted

[3] other interpretations are possible, e.g. inlined data

[4] using a ternary subspace search and one bit in the content space set aside to detect free entries for reclamation

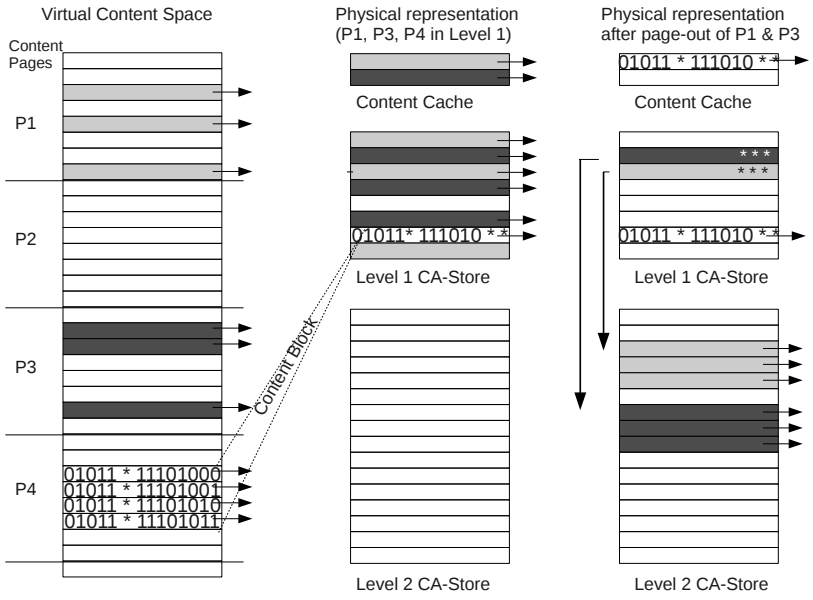

Figure 2: Sample Content Addressable Space

earlier, the notion of a page need not be limited to the range covered by least significant bits in content space - any subset of bits in content words could be defined as a subspace page using wildcards. Different content page subspace masks may be used by different applications (depending on the structure of content locality expected).

3 Content Locality Classification

If "locality of reference breeds the memory hierarchy" [7], then locality of content would determine the potential value of a content addressable memory hierarchy. While a quantitative characterization of content locality in candidate applications requires further research, we can attempt a qualitative assessment to obtain a sense of the implications. We classify application workloads into different categories based on the expected pattern of matches in content addressable space.

1. **Rare Hits**: e.g. intrusion detection. In this case most entries can be moved to level 2 and are brought in when there is malicious traffic or input pattern that is close to a malicious pattern
2. **Frequent Same Item Hits**: e.g. finding frequent items in data streams. In this case, items above the frequency threshold would be in Level 1 or even in the content cache, while others may be moved in and out on-demand based on available capacity
3. **Clustered or Nearby Item Hits**: In this case, content subspace paging will help as it brings in the items mostly likely to be required into level 1 while bulk of the entries can reside at level 2
4. **(Uniformly) Random Item Hits**: In this case performance will depend on the ratio of available level 1 capacity and total number of entries.

In many cases content locality characteristics depend on the input distribution, e.g. similarity search, regular expression matching, packet classification and deduplication. As a starting assumption, we might expect a few frequently hit clusters and potentially many rare hit clusters. In program analysis, dynamic analysis associations exhibit a high range locality [12]. For database join, it depends on the join selectivity and cardinality.

Other potential implications In traditional location based addressing, associations are modeled through spatial relationships (e.g. spatial contiguity, index arithmetic, pointers, hashing). Using content based addressing, these can be expressed directly to the underlying system. This can free the application from spatial constraints and enable a lower level optimizer to move data around at a fine granularity without breaking any dependencies. With search driven execution becoming a common paradigm, data and operational associations are heavily used in general purpose middleware and application software. In a given deployment context, many

conditions change rarely. Thus a small subset of associations are likely to be used most often. Content based caching might be very effective in reducing overheads in these situations.

4 Implementation Challenges

Characterizing content locality and content key working sets of existing workloads is an important first step in determining the design space parameters for feasibility. Early implementations of a CAVMM may be built without requiring any extra architecture support in order to evaluate minimal hardware system mechanisms that are essential. Besides this, there are many design issues that need to be researched, such as policies for allocation and reclamation of TCAS (and LAS), sharing of space across processes, and ternary compaction optimizations that might be applied by the OS (e.g. at the time of pageout) to minimize the number ternary word entries. Furthermore, mechanisms for concurrent access to CAS by independent threads needs to be explored in depth along with a study of how transactional consistency can be achieved, in the concurrent context, when multiple entries/locations are updated on certain "elementary" CAS operations.

However, the larger design issue for debate and discussion, once basic questions of viability have been addressed, is the choice of interface through which the abstraction is exposed to applications. While the idea of a fully transparent virtual memory model where a content key is treated as just another address reference is an appealing one, it also raises some conceptual complications, the answers to which are not yet clear. For example: Is there a need for new instructions to express operations involving ternary content keys? If not how should compilers handle and generate content key variables, particularly when there are multiple entries with the same key? How could atomicity be handled implicitly and at what cost? An exposed interface, on the other hand, might be simpler to design but complex for users.

TCAM Extensions Currently TCAMs are usually configured to return the first match in the event of multiple matches (using a priority encoder). This can be very inefficient in many situations e.g. database operations, content page retrieval. Support for efficient bulk transfer for multiple matches is therefore an important requirement. TCAMs need not be the only hardware content addressability mechanism used in a CAVMM hierarchy. For example, hardware range caches or E-TCAMs which allow non-power of two ranges to be represented efficiently and other pattern matching accelerators may also be worth consideration.

5 Conclusions

The advent of flash and storage class memories is changing virtual memory and storage hierarchy. We bring in another dimension by proposing that content addressability be considered as a first class abstraction in virtual memory design.

While we have provided a flavor of how such ideas may be implemented, and where they might be useful, we believe that we have only scratched the surface of technical challenges and implications of a promising new direction of research. One advantage of our approach is that it enables a natural extension of VMM to support content addressability, while retaining full compatibility with traditional location based addressing. A shift from spatial locality to content locality based optimization can open up possibilities as radical as that opened up by the shift from disk based optimizations to those for storage class memories. Exploration of these opportunities will require a close collaboration between memory system architects, operating system and software researchers.

References

[1] AGRAWAL, B., AND SHERWOOD, T. Ternary CAM Power and Delay Model: Extensions and Uses. *IEEE Trans. on VLSI Systems*.

[2] ARCANGELI, A., EIDUS, I., AND WRIGHT, C. Increasing memory density by using ksm. *OLS* (2009).

[3] BANDI, N., SCHNEIDER, S., AGRAWAL, D., AND ABBADI, A. E. Hardware Acceleration of Database Operations Using Content Addressable Memories. *DaMoN* (2005).

[4] BANDI, N., SCHNEIDER, S., AGRAWAL, D., AND ABBADI, A. E. Fast Data Stream Algorithms using Associative Memories. *SIGMOD* (2007).

[5] BISWAS, S., FRANKLIN, D., SAVAGE, A., DIXON, R., SHERWOOD, T., AND CHONG, F. T. Multi-Execution: Multicore Caching for Data-Similar Executions . *ISCA* (2010).

[6] GOEL, A., AND GUPTA, P. Small Subset Queries and Bloom Filters Using Ternary Associative Memories, with Applications. *SIGMETRICS* (2010).

[7] JACOB, B., NG, S. W., AND WANG, D. T. Memory systems: Cache, dram, disk. *Elsevier Inc.* (2008).

[8] KOLLER, R., AND RANGASWAMI, R. I/o deduplication: Utilizing content similarity to improve i/o performance. *FAST* (2010).

[9] MEINERS, C. R., PATEL, J., NORIGE, E., TORNG, E., AND LIU, A. X. Fast Regular Expression Matching using Small TCAMs for Network Intrusion Detection and Prevention Systems. *USENIX ATC* (2010).

[10] PAGIAMTZIS, K., AND SHEIKHOLESLAMI, A. Content Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey. *IEEE Journal of Solid State Circuits*.

[11] SHINDE, R., GOEL, A., GUPTA, P., AND DUTTA, D. Similarity Search and Locality Sensitive Hashing using Ternary Content Addressable Memories. *SIGMOD* (2010).

[12] TIWARI, M., AGRAWAL, B., MYSORE, S., VALAMEHR, J. K., AND SHERWOOD, T. A Small Cache of Large Ranges: Hardware Methods for Efficiently Searching, Storing, and Updating Big Dataflow Tags. *Micro* (2008).