Architecture Conscious Data Mining

Summary of Results and Future Outlook[*]

Srinivasan Parthasarathy
Data Mining Research Laboratory
Department of Computer Science and Engineering
Ohio State University
Email: srini@cse.ohio-state.edu

Abstract

The field of knowledge discovery and data mining is concerned with extracting actionable knowledge from data in as efficient a manner as possible. Over the last decade and a half, much progress has been made in terms of the development of novel and efficient algorithms. There are also many success stories reported in novel application domains, where the discovery of actionable patterns has led to important economic and scientific breakthroughs.

That said, as a field we are still quite young and much remains to be done. In this article we will focus on one aspect of the above theme, namely efficient algorithms. On this front we will take a detailed look at the promise offered by architecture-conscious algorithm designs and demonstrate both the viability and utility of such designs. Broader implications for education and for cyber-enabled discovery for innovation will also be discussed.

1. Introduction

Advances in technology have enabled us to collect vast amounts of data across a myriad of domains for various purposes, ranging from astronomical observations to health screening tests, from computational fluid simulations to network flow data, from genomic data to large scale interaction networks, at an ever-increasing pace. To benefit from these large data stores often housing tera- and even peta-scale datasets, organizations and individuals have increasingly turned to knowledge discovery and data mining (KDD) methods, to extract vital information and knowledge from such data stores.

At an abstract level the KDD process is concerned with *extracting actionable knowledge from data stores efficiently*. Over the last 15+ years much progress has been made. New and efficient KDD algorithms have been developed and deployed. Key scientific, engineering, financial and economic breakthroughs have been enabled by various data mining and data preprocessing techniques. That said we are still a very young field and there is indeed a long laundry list of objectives to be met, ranging from improved theoretical foundations in key KDD sub-fields to a better understanding of how to define and determine pattern interestingness, from new application domains such as interaction graphs and proteomics to biomedical knowledge discovery, from better visualization methods that can aid in data and model understanding to developing even more efficient and scalable algorithms capable of handling peta bytes of scientific and web data. The last of these, efficiency, is the primary focus of this article.

Efficiency is critical to the KDD process since the process is iterative (repetitive) and involves a human-in-the-loop (interactive). In fact interactivity is often the key to facilitating effective data understanding and knowledge discovery since lengthy time delay between responses of two consecutive user requests can disturb the flow of human perception and formation of insight. KDD researchers have tackled the problem of scalability and efficiency in numerous ways: through the development of innovative algorithms which reduce the theoretical or empirical complexity; through the use of compression and sampling techniques to reduce memory and I/O costs; and through the use of parallel and distributed algorithm designs. In this article we submit that an orthogonal design strategy – architecture conscious algorithm designs – in addition to the afore-mentioned ones needs to be investigated and integrated with mainstream KDD algorithms. There are two reasons for this argument: i) it is a very fruitful endeavor often yielding performance improvements of up to 3 orders of magnitude; ii) with impending commodity multicore technology it more or less becomes a necessity.

Over the past several years, architectural innovation in processor design has led to new capabilities in single-chip

[*]The material is based upon work supported by NSF grants CCF-0702587, CNS-0406386, IIS-0347662, RI-CNS-0403342 and an IBM Faculty Partnership award.

commodity processing and high end compute clusters. Examples include hardware prefetching, simultaneous multithreading (SMT), and more recently true chip multiprocessing. At the very high-end, systems area networking technologies like InfiniBand have spurred the development of affordable cluster-based supercomputers capable of storing and managing peta bytes of data. We contend that data mining algorithms often require significant computational, I/O and communication resources, and thus stand to benefit from such innovations if appropriately leveraged. The challenges to do so are daunting.

First, a large number of state-of-the-art data mining algorithms grossly under-utilize modern processors, the building blocks of current generation commodity clusters. This is due to the widening gap between processor and memory performance and the memory and I/O intensive nature of these applications. Second, the emergence of multi-core architectures to the commodity market, bring with them further complications. Key challenges brought to the fore include the need to enhance available fine-grained parallelism and to alleviate memory bandwidth pressure. Third, parallelizing data mining algorithms on a multi-level cluster environment is a challenge given the need to share and communicate large sets of data and to balance the workload in the presence of data skew.

In this article we discuss progress made in the context of these challenges and attempt to demonstrate that "architecture conscious" solutions are both viable and necessary. We will attempt to separate general methodologies and techniques from specific instantiations whenever it makes sense. We will conclude with a discussion on future outlook, both in the context of systems support for cyber discovery – enabling the development of new algorithms on emerging architectures – as well as in terms of educational objectives brought to the fore in this context.

2 Algorithms for Modern Uniprocessors

The widening gap between processor performance and memory subsystem performance in modern processors is a critical bottleneck limiting the performance of data intensive applications [7]. Further hampering performance is the parameter and data-dependent aspects of data mining algorithms that makes predicting access patterns very difficult thus leading to poor data locality. The reliance on dynamic data structures to house important meta information to prune the search space often limits available instruction level parallelism while also hampering data locality leading to poor processor utilization. The challenges to overcome are daunting and below we identify several simple strategies that seem to be quite useful for a range of data analysis and management applications.

The first strategy is to simply improve the spatial locality within such algorithms. The basic idea is to ensure that once a data object is read or written to, objects located in spatial proximity to said object will also be touched soon. Exploiting this strategy requires an understanding of the access patterns in the algorithm, *identifying the dominant access patterns* (to make the common case fast), and then *realizing a memory placement that matches the dominant access patterns*. We have found this to be useful for a host of algorithms including frequent pattern mining, clustering, and outlier detection. In fact as a recent study on frequent pattern mining showed such a strategy can yield up to a 50% improvement in execution times [7].

The second strategy, is to improve the temporal locality within such algorithms.Here, the objective is to schedule all the operations that must be carried out on a data object close together in time. Exploiting this strategy requires the ability to partition the data (or the data structure) and *operate on a partition-by-partition* basis, i.e., process a partition completely before moving on to the next partition. Such a strategy minimizes the number of times one has to cross different levels of the memory hierarchy. Note that partitions may be overlapping but the key is to minimize degree of overlap. Furthermore, partitions may be a natural cut of the data (or the data structure) or alternatively may be constructed on the fly through suitable hash or approximate ordering functions. It turns out one can exploit this strategy for a range of applications including for clustering, classification, outlier detection and frequent pattern mining among others. For example on a state-of-the-art frequent itemset mining algorithm we find that such strategies can yield a 2 fold improvement for in-core datasets and up to a 400 fold improvement for out of core datasets[3].

The third strategy is to *leverage key features of such architectures effectively*. For example improving spatial locality can have the added benefit of effectively leveraging hardware prefetching – a latency tolerating mechanism available in modern architectures. Another recent innovation – simultaneous multithreading – a mechanism that enables one to place multiple thread contexts in hardware with a goal of increasing available instruction level parallelism. A naive realization for data mining, essentially deploying parallel independent threads (tasks), often fails to achieve the desired effect due to cache conflicts and data stalls. However, we find that co-scheduling tasks that operate on identical or related pieces of data, limit the number of data stalls and conflict misses, yielding close to ideal performance for some applications[8].

The fourth strategy, one that requires an in-depth knowledge of the algorithm, is to examine alternative design strategies that *limit the use of pointer-based data structures* with the potential to yield algorithms with smaller memory footprints and better ILP. As an example we recently considered the problem of mining frequent trees within a forest of trees. The key to our approach was to rely on a bijection between trees and sequences (with appropriate structural information embedded in the representation). Subsequently we designed a dynamic programming based algorithm that finds frequent patterns by recasting a specialized case of the subtree isomorphism problem to the much simpler problem of subsequence matching. The approach is memory conscious and much more efficient, compared to the state-of -the-art algorithms which rely on pointer-based data structures. Our algorithms, when compared to the state-of-the-art, exhibited up to 355-fold improvement in overall mining time and a significant reduction in memory footprint size (up to 2 orders of magnitude) [20, 22]. As it turns out this approach has an even more general appeal in that a similar strategy proved to be extremely effective for indexing XML data [21]. Once again we found that the approach outperforms the state-of-the-art XML indexing strategies by up to 3 orders of magnitude while utilizing significantly less memory and disk resources.

3 Mining on Emerging CMP Architectures

Further complicating the already difficult task of utilizing modern architectural platforms efficiently is the recent emergence of true chip multiprocessing (CMP), often referred to as multicore chips. Designs range from the general purpose (AMD, Intel) to the specialized (Sony-Toshiba-IBM Cell, Sun) to the niche markets (GPUs). Although current designs have 4-8 cores, Intel's 2015 Processor road map proposes CMPs with hundreds of cores[1].

Parallelizing existing algorithms is an obvious objective in this context. While much work has been done on this, there are some important challenges. Paramount to leveraging the additional compute capability is an effective task partitioning mechanism to distribute the work among individual processing elements. This can be particularly challenging in the presence of data and control dependencies. Moreover, the data and parameter dependent aspects of data mining workloads makes estimating the lifetime of a task difficult. A final challenge, particular to emerging CMPs, is the fact that bandwidth to main memory is likely to be a precious shared commodity.

To address these challenges we believe adaptable algorithm designs must take center stage. For example, to handle the issue of task granularity we have proposed schemes that allow the size of a task to morph depending on the state of the system. We term this *moldable task partitioning*. where each task makes a decision at pre-set points during its runtime whether it needs to further break up into sub-tasks, which can subsequently be enqueued on a distributed shared task queue or to continue processing. This decision would be based on the current load balance in the system. Adaptive partitioning not only affords a dynamic granularity to accommodate variability in the associativity of the data set, it also can improve cache performance [4]. We find that such a strategy is both necessary and extremely effective when processing highly skewed datasets in the context of graph and tree mining[4, 20] resulting in near linear speedups.

[1]http://www.intel.com/technology/magazine/computing/platform-2015-0305.htm

It is common that algorithms can trade increased memory usage for improved execution times. *We seek to leverage this principle to improve run times for data mining applications by maintaining additional state when it is inexpensive to do so*. Architectures today provide us with performance counters to estimate bandwidth utilization at runtime. We propose to develop solutions that can trade off algorithmic state for reduced bandwidth consumption thereby facilitating execution on CMPs. If there is sufficient memory bandwidth available in the system, then we can choose to increase state for future use. Furthermore, in the presence of contention, we can limit or reduce state as needed. We term this *adaptive state management*.

We evaluated the benefits of using adaptive state management by mining real world data sets for substructure (graphs and trees) mining. The task-level meta data structure is called an embedding list, which maps discovered graphs to their mappings in the data set. The embedding list reduces the search space of a task at the expense of maintaining increased state. We compared the performance of three algorithms, one that always maintains the state (gaston [15]), one that never maintains the state (gSpan [26]) and one that adaptively maintains state (ours). Notably our adaptive algorithm is always the most efficient and when operating on smaller number of cores the algorithm adaptively maintains more state and when operating on large number of cores the algorithm adaptively maintains less state trading memory off for computation. Overall, the adaptive approach makes near optimal use of the available hardware resources[4]. This result, particularly the implicit crossover in performance between gaston (better on lower number of cores) and gspan (better on higher number of cores) underscores an important point in that simpler algorithm designs with smaller memory footprints are like to be the norm for deployment on emerging architectures.

Thus far we have discussed the performance of data mining algorithms on general purpose emerging architectures. A joint venture by Sony, Toshiba and IBM (STI) has produced a nine core architecture called the Cell BDEA. This architecture represents an interesting design point along the spectrum of chipsets with multiple processing elements. The STI Cell has one main processor and eight support processing elements (SPEs) – over 200+ GFLOPS of compute power – and with explicit memory management – 25GB/s off chip bandwidth. While most general purpose CMPs are MIMD with POSIX-style threading models, the Cell's eight cores are SIMD vector processors, and must get specialized task modules passed to them by the main processor. Several workloads seem quite amenable to its architecture. For example, high floating point workloads with streaming access patterns are of particular interest [25]. These workloads could leverage the large floating point throughput, and because their access pattern is known *a priori*, they can use software-managed caches for good bandwidth utilization.

In recent work we have sought to map and evaluate several important data mining algorithms on the Cell, namely clustering, classification and anomaly detection. We investigated these algorithms along the axes of performance, programming complexity and algorithm design [2]. Specifically, we develop data transfer and SIMD optimizations for these applications and evaluate them in detail to determine both the benefits of the Cell processor for data applications, as well as the inherent bottlenecks. As part of our comparative analysis we juxtapose these algorithms with similar ones implemented on modern architectures including the Itanium, AMD Opteron and Pentium architectures. For the workloads we consider, the Cell processor is up to 50 times faster than competing technologies, when the underlying algorithm uses the hardware efficiently. An important outcome of the study, beyond the results on these particular algorithms is that we answer several higher level questions, which are designed to provide a fast and reliable estimate to application designers for how well other workloads will scale on the Cell.

4 Mining on Emerging Clusters

Data mining on a tightly interconnected distributed cluster of shared memory processors is a cost-effective and viable solution for analyzing large datasets. In this regard a number of researchers over the last decade have developed innovative parallel or distributed algorithms for various data mining algorithms [1, 5, 16, 24, 27, 28, 1, 6, 11, 12, 19, 17, 10, 18, 9, 13, 14]. However, we argue that here too architectural resources are not being fully utilized. A multi-level – within a multicore chip, across multiple chips on a shared memory workstation, and across a cluster of workstations – design strategy is needed to fully utilize such a supercomputing resource. Load balancing in the presence of data skew, accommodating data and task parallelism across multiple levels, and effectively leveraging strategies like remote memory paging are key to efficient utilization.

For example we have recently presented a parallelization of a fast frequent itemset mining algorithm to enable efficient mining of very large, out-of-core data sets [5]. Our implementation employs such a multi-level "architecture-conscious" design strategy to efficiently utilize memory, processing, storage, and networking resources while minimizing I/O and communication overheads. Our experimental evaluation using a state-of-the-art commodity cluster and large data illustrates linear scale up and significant system utilization (up to an order of magnitude improvement over competing strategies) and the ability to easily process tera-scale datasets on a 48-node cluster.

5 Future Outlook

On the research front, the future outlook for architecture-conscious data mining is clearly very encouraging. Quite simply it offers a new and orthogonal approach to enhancing the efficiency with which algorithms can compute and tackle large datasets. Moreover, with the advent of multi-core processors to the commodity computing market it more or less becomes a necessity since algorithm developers will need to attune themselves to the novel features and limitations of these architectures. There are clearly several directions of work to look at in the future.

While the early signs are encouraging, deployment on other data mining techniques, in novel application settings is still needed and very challenging. In many cases, there are strong dependency structures at a conceptual or algorithmic level limiting available parallelism that need to be overcome (e.g. exact inference in graphical models). Innovative algorithmic restructuring, resorting to approximate solutions to enhance parallelism, pro-active or speculative methods all have a role to play in this context.

A fundamental question to ask here is moving forward what have we learnt from these successful architecture-conscious implementations? What can we take away for future algorithms and realizing architecture-conscious implementations on emerging and next generation platforms? The general principles, outlined earlier, such as improving spatial locality, improving temporal locality, minimizing the use of pointer-based dynamic data structures, lowering memory footprints and trading off memory for computation, minimizing communication offer a good starting point but more needs to be done.

This question and the basic solutions outlined lead us to the related question of whether (cyber-)infrastructure support for realizing architecture conscious knowledge discovery and data mining implementations is feasible? It is our opinion that a *services-oriented architecture* holds significant promise in this context[24, 23]. Such an architecture could include services for data access and partitioning (e.g. hash-sorting, sampling, memory placement, distributed shared data structures) ; services for data and knowledge reuse and caching; services for scheduling and load balancing etc. Various data pre-processing, data-mining, and visualization modules can be implemented on top of these basic services and can in turn serve as front ends for more complex knowledge discovery tasks. The plug-and-play nature of the services model is ideally suited to the interactive and iterative nature of KDD algorithms. Furthermore, each individual service can be attuned to the architecture upon which it is built with the potential to yield architecture conscious solutions for a wide range of of data mining techniques and end applications. Additionally, the "services" oriented approach ensures useful utilization of computational resources.

On the educational front, with the advent of multi-core architectures parallel computing is essentially entering the main-stream commodity market. Designers must be aware of the basic principles underlying parallel algorithm design, familiar with important architectural advances and features, in addition to having an innate understanding of data mining principles. *Co-learning* group projects where groups are formed by matching students with strong systems and architecture background along with students more familiar with data mining and statistical learning algorithms are a useful mechanism in this context. Such efforts are essential and must go hand-in-hand with research advances.

Acknowledgments: Gregory Buehrer, Amol Ghoting, and Shirish Tatikonda have been key partners in our quest for architecture conscious data mining solutions.

References

[1] R. Agrawal and J. Shafer. Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 8(6):962–969, 1996.

[2] G. Buehrer and S. Parthasarathy. The potential of the cell broadband engine for data mining. *Technical Report*, (OSU-CISRC-3/07–TR22), 2007.

[3] G. Buehrer, S. Parthasarathy, and G. A. Out of core frequent pattern mining on a commodity pc. *ACM SIGKDD*, 2006.

[4] G. Buehrer, S. Parthasarathy, and Y. Chen. Adaptive Parallel Graph Mining for CMP Architectures. *Proceedings of the Sixth International Conference on Data Mining (ICDM)*, pages 97–106, 2006.

[5] G. Buehrer, S. Parthasarathy, S. Tatikonda, T. Kurc, and J. Saltz. Toward terabyte pattern mining: an architecture-conscious solution. *Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 2–12, 2007.

[6] D. Cheung, J. Han, V. Ng, A. Fu, and Y.Fu. A fast distributed algorithm for mining association rules. In *4th Intl. Conf. Parallel and Distributed Info. Systems*, Dec. 1996.

[7] A. Ghoting, G. Buehrer, S. Parthasarathy, D. Kim, A. Nguyen, Y. Chen, and P. Dubey. Cache-conscious frequent pattern mining on a modern processor. *Proceedings of the 31st international conference on Very large data bases*, pages 577–588, 2005.

[8] A. Ghoting, G. Buehrer, S. Parthasarathy, D. Kim, A. Nguyen, Y. Chen, and P. Dubey. Cache-conscious frequent pattern mining on Modern and Emerging Processors. *VLDB Journal*, pages 77–96, 2007.

[9] S. Goil and A. Choudhary. Efficient parallel classification using dimensional aggregates. In *Proceedings of Workshop on Large-Scala Parallel KDD Systems, with ACM SIGKDD-99*. ACM Press, Aug. 1999.

[10] V. Guralnik, N. Garg, and G. Karypis. Parallel tree projection algorithm for sequence mining. *Lecture Notes in Computer Science*, 2150, 2001.

[11] E.-H. Han, G. Karypis, and V. Kumar. Scalable parallel datamining for association rules. *IEEE Transactions on Data and Knowledge Engineering*, 12(3), May / June 2000.

[12] H. Kargupta and B. Park. *Mining Decision Trees from Data Streams in a Mobile Environment*. IEEE ICDM, CA, 2001.

[13] M. Mehta, R. Agrawal, and J.Rissanen. Sliq: A fast scalable classifier for data mining. In *In Proc. of the Fifth Int'l Conference on Extending Database Technology*, Avignon, France, 1996.

[14] G. J. Narlikar. A parallel , multithreaded decision tree builder. Technical Report CMU-CS-98-184, School of Computer Science, Carnegie Mellon University, 1998.

[15] S. Nijssen and J. Kok. A quickstart in frequent structure mining can make a difference. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 647–652, 2004.

[16] S. Parthasarathy, M. Zaki, M. Ogihara, and W. Li. Parallel Data Mining for Association Rules on Shared-Memory Systems. *Knowledge and Information Systems*, 3(1):1–29, 2001.

[17] A. Schuster and R. Wolff. Communication-efficient distributed mining of association rules. In *Proceedings of the 2001 ACM SIGMOD Conference*, June 2001.

[18] J. Shafer, R. Agrawal, and M. Mehta. SPRINT: a scalable parallel classifier for data mining. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1996.

[19] T. Shintani and M. Kitsuregawa. Hash based parallel algorithms for mining association rules. In *4th Intl. Conf. Parallel and Distributed Info. Systems*, Dec. 1996.

[20] S. Tatikonda, S. Parthasarathy, and M. Goyder. An efficient parallel tree mining algorithm for emerging commodity processors. *Technical Report*, (OSU-CISRC-3/07–TR18), 2007.

[21] S. Tatikonda, S. Parthasarathy, and M. Goyder. Lcs-trim: Dynamic programming meets xml indexing and querying. *Proceedings of 33rd Very Large Data Bases Conference (VLDB)*, page to appear, 2007.

[22] S. Tatikonda, S. Parthasarathy, and T. Kurc. TRIPS and TIDES: new algorithms for tree mining. *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 455–464, 2006.

[23] H. Wang, A. Ghoting, G. Buehrer, S. Tatikonda, S. Parthasarathy, T. Kurc, and J. Saltz. A services oriented architecture for next generation data analysis centers. *Next Generation Software Workshop held with IPDPS*, 2005.

[24] H. Wang, S. Parthasarathy, A. Ghoting, S. Tatikonda, G. Buehrer, T. Kurc, and J. Saltz. Design of a next generation sampling service for large scale data analysis applications. In *Proceedings of the 19th annual international conference on Supercomputing (ICS)*, pages 91–100. ACM Press, 2005.

[25] S. Williams, J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. Yelick. The potential of the cell processor for scientific computing. *Proceedings of the 3rd conference on Computing frontiers*, pages 9–20, 2006.

[26] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. *Proceedings of International Conference on Data Mining (ICDM)*, pages 721–724, 2002.

[27] M. Zaki, W. Li, and S. Parthasarathy. Customized dynamic load balancing for a network of workstations. *Journal of Parallel and Distributed Computing (JPDD)*, 43(2):156–162, 1997.

[28] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel Algorithms for Discovery of Association Rules. *Data Mining and Knowledge Discovery*, 1(4):343–373, 1997.