M3: Scaling Up Machine Learning via Memory Mapping

Dezhi Fang
College of Computing
Georgia Institute of Technology
dezhifang@gatech.edu

Duen Horng Chau
College of Computing
Georgia Institute of Technology
polo@gatech.edu

ABSTRACT
To process data that do not fit in RAM, conventional wisdom would suggest using distributed approaches. However, recent research has demonstrated virtual memory’s strong potential in scaling up graph mining algorithms on a single machine. We propose to use a similar approach for general machine learning. We contribute: (1) our latest finding that memory mapping is also a feasible technique for scaling up general machine learning algorithms like logistic regression and k-means, when data fits in or exceeds RAM (we tested datasets up to 190GB); (2) an approach, called M3, that enables existing machine learning algorithms to work with out-of-core datasets through memory mapping, achieving a speed that is significantly faster than a 4-instance Spark cluster, and comparable to an 8-instance cluster.

CCS Concepts
• Software and its engineering → Virtual memory;
• Computing methodologies → Machine learning;

1. INTRODUCTION
Leveraging virtual memory to extend algorithms for out-of-core data has received increasing attention in data analytics communities. Recent research demonstrated virtual memory’s strong potential to scale up graph algorithms on a single PC [4, 3]. Available on almost all modern platforms, virtual memory based approaches are straightforward to implement and to use, and can handle graphs with as many as 6 billion edges [3]. Some single-thread implementations on a PC can even outperform popular distributed systems like Spark (128 cores) [4]. Memory mapping a dataset into a machine’s virtual memory space allows the dataset to be treated identically as an in-memory dataset. The algorithm developer no longer needs to explicitly determine how to partition the (large) dataset, nor manage which partitions should be loaded into RAM, or unloaded from it. The OS performs similar actions on the developer’s behalf, through paging the dataset in and out of RAM, via highly optimized OS-level operations.

2. SCALING UP USING M3
As existing works focused on graph algorithms such as PageRank and finding connected components, we are investigating whether a similar virtual memory based approach can generalize to machine learning algorithms at large.

Inspired by prior works on graph computation, our M3\(^1\) approach uses memory mapping to amplify a single machine’s capability to process large amounts of data for machine learning algorithms. As memory mapping a dataset allows it to be treated identically as an in-memory dataset, M3 is a transparent scale-up strategy that developers can easily apply, requiring minimal modifications to existing code. For example, Table 1 shows that with only minimal code changes and a trivial helper function, existing algorithm implementation can easily handle much larger, memory-mapped datasets.

Modern 64bit machines have address spaces large enough to fit large datasets into (up to 1024PB). Because the operating system has access to a variety of internal statistics on how the mapped data is being used, the access to such data can be further optimized by the operating system via

\(^1\)M3 stands for Machine Learning via Memory Mapping.
methods including the use of virtual memory as a fundamental, alternative way to scale up machine learning algorithms. M3 adds an interesting perspective to existing solutions primarily based on distributed systems.

We contribute: (1) our latest finding that memory mapping could become a feasible technique for scaling up general machine learning algorithms when the dataset exceeds RAM; (2) M3, an easy-to-apply approach that enables existing machine learning implementations to work with out-of-core datasets; (3) our observations that M3 on a PC can achieve a speed that is significantly faster than a 4-instance Spark cluster, and comparable to an 8-instance cluster.

We will extend our M3 approach to a wide range of machine learning (including online learning) and data mining algorithms. We plan to extensively study the memory access patterns and locality of algorithms (e.g., sequential scans vs random access) to better understand how they affect performance. We plan to develop mathematical models and systematic approaches to profile and predict algorithm performance and energy usage based on extensive evaluations across platforms, datasets, and languages.

5. ACKNOWLEDGMENTS

We thank Dr. Ryan Curtin and Minsuk (Brian) Kahng for their feedback. This work was supported by NSF grants IIS-1217559, TWC-1526254, and by gifts from Google, Symantec, Yahoo! Labs, eBay, Intel, Amazon, and LogicBlox.

6. REFERENCES


