Linear Algebraic Graph Algorithms for Back End Processing

Jeremy Kepner, Nadya Bliss and Eric Robinson {kepner,nt}@ll.mit.edu
MIT Lincoln Laboratory, Lexington, MA 02420

Abstract

A natural byproduct of persistent surveillance and net-centric sensing has been an explosion in post-detection metadata. The processing required to exploit this metadata in real-time is rapidly emerging as a critical bottleneck. This meta-data is often represented as a graph of entities (nodes) and relationships (edges), and these graphs are analyzed using standard graph analysis techniques. High performance graph algorithms are difficult to implement for a two of reasons. First, the complex memory access patterns make these algorithms difficult to write in parallel. Second, these access patterns are very inefficient on COTS processors. This work addresses the parallel programming issue by exploiting the well known graph/sparse-matrix duality which leverages decades of research drawn from parallel linear algebra. Using these linear algebraic graph algorithms the bottlenecks in COTS processors can be identified.

Introduction

Persistent surveillance allows the long term (weeks) observation of wide areas (kilometers) in multiple networked sensor modalities (optical, infrared, RADAR, …). This shift in sensor operation has caused a corresponding shift from using data (images, detects, …) to meta-data (tracks, features, patterns, …) for real-time exploitation. This meta-data is often represented as a graph of entities (nodes) and relationships (edges). Not surprisingly, graph algorithms are the principal tools used for analyzing this meta-data.

Historically, front end processing has always dominated back end (post detection) processing because the front end data rates (GB/sec) are so much larger than post detection rates (megatracks/day). This has changed for three reasons. First, post-detection processing windows are weeks in duration. Second, post-detection algorithms are more complex, often $O(N^2)$. Third, the efficiency of post-detection algorithms is very low on COTS processors: 0.05%. The result is that post-detection processing can now easily consume far more electrical power than front end processing (see Table 1).

Table 1: Front end vs back end requirements.

	Front End	Back End
Input rate	Gigasamples/sec	Megatracks/day
Correlation time	seconds	weeks
Complexity	O(N log(N))	$O(N^2)$
Efficiency	50%	0.05%
Latency	seconds	minutes
Power required	**~ 1 kWatt**	**>> 100 kWatt**

Graph Algorithms and Programming

Post detection processing typically relies on graph algorithms. The essence of many algorithms involves selecting a node, finding the connecting nodes and then proceeding to those nodes. While these algorithms often have ample parallelism (often all nodes can be evaluated independently), it is very difficult to write parallel graph algorithms that exploit data locality, which is essential for good performance on COTS (Commercial Off The Shelf) processors. One approach to addressing this problem is to recast the graph as a sparse adjacency matrix **A**, where **A**(i,j) = 1 if there is an edge between node i and node j. Using this duality, many graph algorithms can be recast as linear algebraic operations (see Figure 1).

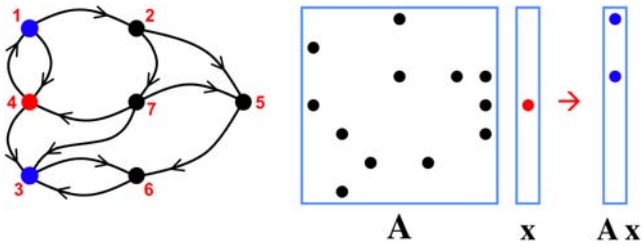

Figure 1: Graph/sparse-matrix duality. Using this notation graph Breadth First Search (BFS) corresponds to matrix vector multiply with the graph adjacency matrix.

Analysis of a number of graph algorithms indicates that nearly all can be efficiently recast as matrix algorithms. Furthermore, these

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

algorithms overwhelmingly rely on a generalization of sparse matrix multiply called semi-ring products. In semi-ring notation a standard matrix multiply (or BFS) can be written as

A +.* **x**

which means that pairs of elements are first multiplied and then these results are summed. For the single-source shortest path (SSSP) algorithm the semi-ring product operation is

A min.+ **x**

which means that pairs of elements are first added and then the minimun of these results is taken.

Recasting graph algorithms as linear algebra on sparse matrices has a number of benefits. First, the resulting algorithms are often significantly shorter than their traditional counterparts (see Figure 2). Second, parallel implementations can leverage the decades of experience with parallel linear algebra. Third, performance issues on COTS parallel processors become readily apparent.

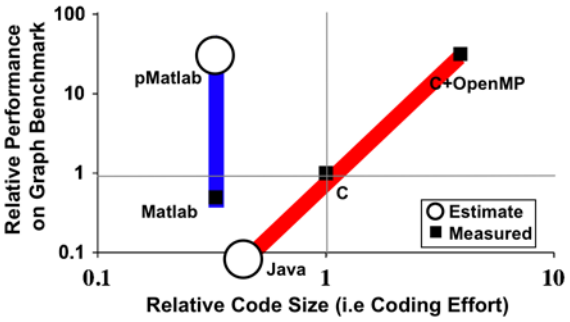

Figure 2: Performance vs effort. Measured and estimated performance and effort for various serial and parallel implementations of the Graph Theory benchmark [Bader & Madduri 2008]

Graph Algorithm Performance

The basic serial performance of graph algorithms can be observed by comparing dense and sparse matrix multiply (see Figure 3) on "power law" graphs. It is readily apparent that this fundamental graph operation is very inefficient on a COTS processor. The primary reason being that a cache architecture is ill-suited to this operation.

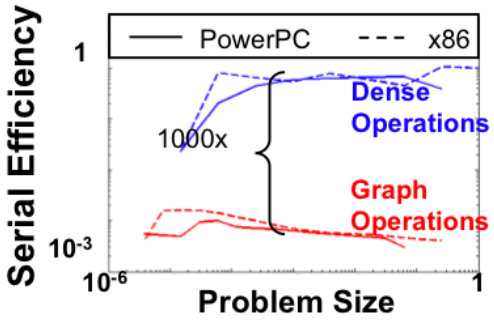

Figure 3: Dense and sparse matrix multiply performance.

The parallel performance of graph algorithms can likewise be understood by looking at the performance of parallel sparse matrix multiply (see Figure 4). The sparse matrix representation highlights the key issue with parallel graph algorithms on COTS parallel processors. Specifically, for every operation performed an equivalent amount of data needs to be moved.

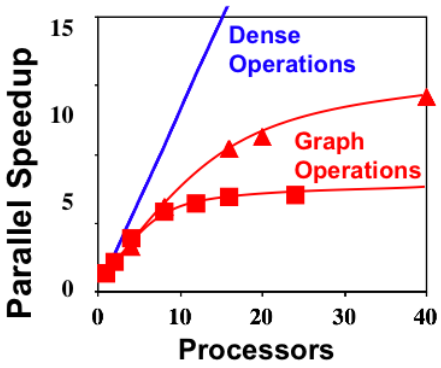

Figure 4: Parallel speedup on graph theory benchmark.

Summary

This work describes the challenges associated with post-detection processing using graph algorithms. A concise mathematical approach to implementing the algorithms using sparse matrix representation is presented. Using these linear algebraic graph algorithms the bottlenecks in COTS processors can be identified.

References

[Bader & Madduri 2008] To appear in *Graph Algorithms in the Language of Linear Algebra* (J. Kepner & J. Gilbert eds.), SIAM, Philadelpha, PA, 2008