Theory of Multicore Algorithms

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

Abstract

The increasing complexity of parallel multicore processors necessitates the use of correspondingly complex parallel algorithms. These algorithms often exploit hierarchical data access patterns and pipeline execution. To facilitate the discussion of these algorithms a mathematical notation for describing these algorithms is introduced. This notation extends existing distributed array notation to handle hierarchical arrays. In addition, pipeline constructs composed of "tasks" and "conduits" are put forth for the describing of general signal flow graphs. This theoretical approach allows complex algorithms to be presented in a succinct mathematical form that is independent of the implementation. The resulting algorithms have a clear a one-to-one correspondence between the data structures and the complex memory hierarchies of modern multicore processors. This correspondence allows algorithms to be developed with a high degree of data locality that is essential for achieving high performance.

Introduction

Effectively using parallel multicore processors requires developing complex algorithms that maximize the locality of data and minimize communication. These algorithms often employ hierarchical descriptions of data and pipeline execution. Describing these algorithms is difficult and can be facilitated with appropriate notation. Such a notation is put forth here. The notation builds upon the non-hierarchical notation presented in the forthcoming SIAM texts "Parallel Programming in pMatlab" [Kepner 2008] and "Graph Algorithms in the Language of Linear Algebra" [Kepner & Gilbert 2008].

Using this notation it is possible to describe a wide range of parallel algorithms and data access patterns. Many of these algorithms can be supported using existing technologies such as PVL, pMatlab, VSIPL++, pMatlabXVM, ROSA II, and PVTOL.

Data Parallelism

Describing parallel algorithms requires augmenting traditional mathematics with some additional notation. In particular, the number of processors used by the computation will be given by N_P. When an algorithm is run in parallel, the same algorithm (or code) is run on every processor. This is referred to as the Single-Program Multiple-Data (SPMD) computation model. To differentiate the N_P programs, each program is assigned a unique processor ID denoted by P_{ID} that ranges from 0 to N_P-1.

In distributed array programming, it is necessary to map the elements of an array onto a set of processors. "P Notation" provides a convenient shorthand for describing this mapping. A matrix that is mapped such that each processor has a block of rows is denoted

$$\mathbf{A} : \mathbb{R}^{P(N)\times M}$$

Likewise, a matrix that is mapped such that each processor has a block of columns is given by:

$$\mathbf{A} : \mathbb{R}^{N\times P(M)}$$

Decomposing along both rows and columns can be written as

$$\mathbf{A} : \mathbb{R}^{P(N)\times P(M)} \quad \text{or} \quad \mathbf{A} : \mathbb{R}^{P(N\times M)}$$

Given two matrices with different mappings $\mathbf{A} : \mathbb{R}^{P(N)\times M}$ and $\mathbf{B} : \mathbb{R}^{N\times P(M)}$, the statement

$$\mathbf{B} = \mathbf{A}$$

will cause the data to be remapped from **A** into the new mapping of **B**.

Access to just the local part of a distributed array is denoted by the ".loc" appendage. For $\mathbf{A} : \mathbb{R}^{P(N)\times P(M)}$ the local part is $\mathbf{A}.\text{loc} : \mathbb{R}^{(N/N_P)\times M}$. This notation is very useful when specifying operations that are entirely local to each processor and require no communication. Using this notation a canonical pulse compression followed by Doppler filtering application can be concisely be written as

$\mathbf{X} : \mathbb{C}^{P(N)\times M}$ $\mathbf{Y} : \mathbb{C}^{N\times P(M)}$ $\mathbf{c} : \mathbb{C}^{M}$

```
for i=1:size(X.loc,1)            // pulse compress
    X.loc(i,:) = IFFT(FFT(X .loc(i,:)) .* c)
Y = X                            // cornerturn
for j=1:size(Y.loc,m)            // doppler filter
    Y.loc(:,j) = FFT(X .loc(:,j))
```

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.

Hierarchical Data

Flat distributed arrays are very useful for describing non-hierarchical parallel algorithms. Modern multicore processors often have complex memory hierarchies (e.g. the Cell processor). Let each processor in the set P be responsible for an additional set of processors $\underline{P}$. This hierarchical mapping is denoted

$$\mathbf{A} : \mathbb{R}^{P(\underline{P}(N))\times M}$$

The part of this array that belongs to a particular processor p and sub-processor $\underline{p}$ is denoted by $\mathbf{A}.\mathrm{loc}_p.\mathrm{loc}_{\underline{p}}$. Using this notation a parallel hierarchical pulse compression Doppler filtering application can be concisely be written as

$\mathbf{X} : \mathbb{C}^{P(\underline{P}(N))\times M}$ $\mathbf{Y} : \mathbb{C}^{N\times P(\underline{P}(M))}$ $\mathbf{c} : \mathbb{C}^{M}$

```
for i=1:size(X.loc.loc,1)        // pulse compress
    X.loc.loc(i,:) = IFFT(FFT(X .loc.loc(i,:)) .* c)
Y = X                            // cornerturn
for j=1:size(Y.loc.loc,m)        // doppler filter
    Y.loc.loc(:,j) = FFT(X .loc.loc(:,j))
```

In the above algorithm, each processor implicitly works on just the data it owns locally. Likewise, if data movement is required to move data between layers in the hierarchy, this is also implicit. Such data movements can be also made explicit if this is desired.

Physical Abstraction: Kuck Diagram

Data locality is critical to the performance of parallel algorithms. Algorithms with high locality minimize the amount of communication required. Hierarchical P notation allows this locality to be expressed across the multicore memory hierarchy. This can be physically depicted via a Kuck diagram (see Figure 1). The Kuck diagram notation [Kuck 1996] provides a clear way of describing a hardware architecture along with the corresponding memory and communication hierarchy.

Summary

This work describes a mathematical notation for complex multicore algorithms. The notation includes hierarchical N dimensional block-cyclic array distributions and pipeline constructs for describing signal flow graphs. This theoretical approach allows complex algorithms to be presented in a succinct mathematical form that is independent the implementation.

References

[Kepner 2008] J. Kepner, *Parallel Programming in pMatlab*, SIAM, Philadelphia, PA, 2008

[Kepner & Gilbert 2008] J. Kepner & J. Gilbert, *Graph Algorithms in the Language of Linear Algebra*, SIAM, Philadelphia, PA, 2008

[Kuck 1996] D. Kuck, *High Performance Computing*. Oxford Univ. Press, NY, 1996

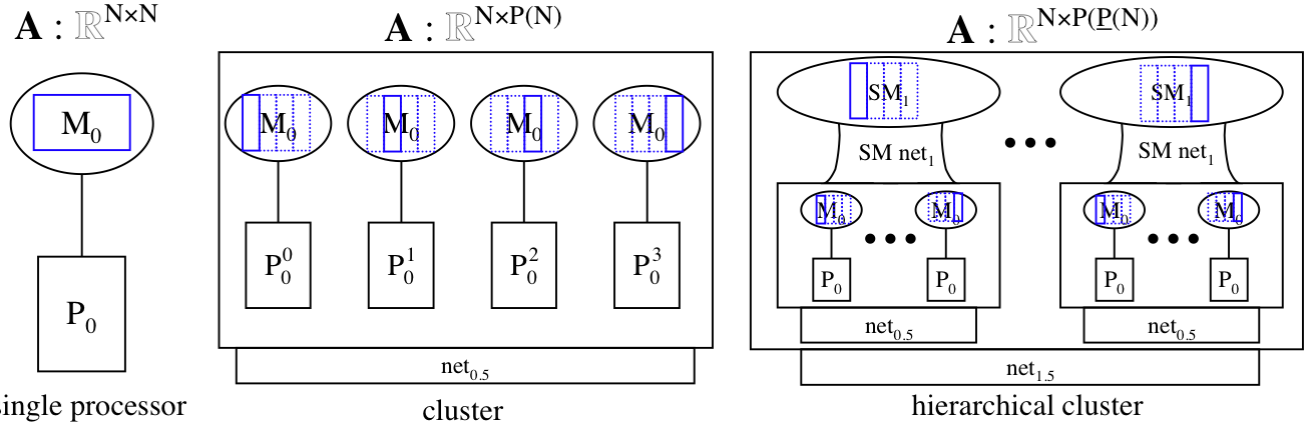

Figure 1: Kuck diagram and corresponding hierarchical arrays. Processing cores are indicated by the letter P. The subscript 0 indicates that the processors are at the 0^{th} level of the hierarchy. There is an implicit superscript that ranges from 0 to the number of processing cores in a hierarchy level. The letter M stands for memory with the S signifying shared memory. M_0 describes the local memory of each processor (such as a cache or a local store), while SM_1 describes shared memory between processors. Similarly, "net" stands for network and "SM net" for shared memory network. Subscripts that end in .5 indicate that the memory access has to occur indirectly, via message passing or a similar approach. For example, a processor P_0 would have to go over $N_{0.5}$ to access another processor's local memory, M_0.