Theory of Multicore Algorithms

Jeremy Kepner and Nadya Bliss

MIT Lincoln Laboratory

HPEC 2008

This work is sponsored by the Department of Defense 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.
In this slide, we discuss:

- Parallel Design
- Distributed Arrays
- Kuck Diagrams
- Hierarchical Arrays
- Tasks and Conduits
- Summary

The main discussion points include:

- Programming Challenge
- Example Issues
- Theoretical Approach
- Integrated Process
Multicore Programming Challenge

Past Programming Model: Von Neumann

- Great success of Moore’s Law era
  - Simple model: load, op, store
  - Many transistors devoted to delivering this model

- Moore’s Law is ending
  - Need transistors for performance

Future Programming Model: ???

- Processor topology includes:
  - Registers, cache, local memory, remote memory, disk

- Cell has *multiple* programming models

Can we describe how an algorithm is *suppose* to behave on a hierarchical heterogeneous multicore processor?
Example Issues

- A serial algorithm can run a serial processor with relatively little specification
- A hierarchical heterogeneous multicore algorithm requires a lot more information

$X, Y : \mathbb{R}^{N \times N}$

$Y = X + 1$

- Where is the data? How is distributed?
- Initialization policy?
- How does the data flow?
- What are the allowed messages size?
- Which binary to run?
- Overlap computations and communications?
Theoretical Approach

- Provide notation and diagrams that allow hierarchical heterogeneous multicore algorithms to be specified.
Integrated Development Process

\[ X, Y : \mathbb{R}^{N \times N} \rightarrow Y = X + 1 \]

\[ X, Y : \mathbb{R}^{P(N) \times N} \rightarrow Y = X + 1 \]

\[ X, Y : \mathbb{R}^{P(P(N)) \times N} \rightarrow Y = X + 1 \]

1. Develop serial code
2. Parallelize code
3. Deploy code
4. Automatically parallelize code

- Should naturally support standard parallel embedded software development practices
Outline

- Parallel Design
- **Distributed Arrays**
- Kuck Diagrams
- Hierarchical Arrays
- Tasks and Conduits
- Summary

- Serial Program
- Parallel Execution
- Distributed Arrays
- Redistribution
Serial Program

\[ X, Y : \mathbb{R}^{N \times N} \]

\[ Y = X + 1 \]

- Math is the language of algorithms
- Allows mathematical expressions to be written concisely
- Multi-dimensional arrays are *fundamental* to mathematics
Parallel Execution

- Run $N_p$ copies of same program
  - Single Program Multiple Data (SPMD)
- Each copy has a unique $P_{ID}$
- Every array is replicated on each copy of the program

$$Y = X + 1$$

$X, Y : \mathbb{R}^{N \times N}$
Distributed Array Program

- Use $P()$ notation to make a distributed array
- Tells program which dimension to distribute data
- Each program implicitly operates on only its own data (owner computes rule)

$X, Y : \mathbb{R}^{P(N) \times N}$

$Y = X + 1$
Explicitly Local Program

\[ X, Y : \mathbb{R}^{P(N) \times N} \]

\[ Y.loc = X.loc + 1 \]

- Use .loc notation to explicitly retrieve local part of a distributed array
- Operation is the same as serial program, but with different data on each processor (recommended approach)
Parallel Data Maps

• A map is a mapping of array indices to processors
• Can be block, cyclic, block-cyclic, or block w/overlap
• Use $P()$ notation to set which dimension to split among processors

Array

Math

\[ \mathbb{R}^{P(N) \times N} \]

\[ \mathbb{R}^{N \times P(N)} \]

\[ \mathbb{R}^{P(N) \times P(N)} \]
Redistribution of Data

\[ X : \mathbb{R}^{P(N) \times N} \]
\[ Y : \mathbb{R}^{N \times P(N)} \]
\[ Y = X + 1 \]

- Different distributed arrays can have different maps
- Assignment between arrays with the “=” operator causes data to be redistributed
- Underlying library determines all the message to send
Outline

- Parallel Design
- Distributed Arrays
- Kuck Diagrams
- Hierarchical Arrays
- Tasks and Conduits
- Summary

- Serial
- Parallel
- Hierarchical
- Cell
Single Processor Kuck Diagram

\[ A : \mathbb{R}^{N \times N} \]

- Processors denoted by boxes
- Memory denoted by ovals
- Lines connected associated processors and memories
- Subscript denotes level in the memory hierarchy
Parallel Kuck Diagram

\[
A : \mathbb{R}^{N \times P(N)}
\]

- Replicates serial processors
- \textbf{Net} denotes network connecting memories at a level in the hierarchy (incremented by 0.5)
- Distributed array has a local piece on each memory
Hierarchical Kuck Diagram

The Kuck notation provides a clear way of describing a hardware architecture along with the memory and communication hierarchy.

Legend:
- P - processor
- Net - inter-processor network
- M - memory
- SM - shared memory
- SMNet - shared memory network

Subscript indicates hierarchy level

x.5 subscript for Net indicates indirect memory access

Cell Example

Kuck diagram for the Sony/Toshiba/IBM processor

\[ \text{P}_{\text{PPE}} = \text{PPE speed (GFLOPS)} \]
\[ \text{M}_{0,\text{PPE}} = \text{size of PPE cache (bytes)} \]
\[ \text{P}_{\text{PPE}} - \text{M}_{0,\text{PPE}} = \text{PPE to cache bandwidth (GB/sec)} \]
\[ \text{P}_{\text{SPE}} = \text{SPE speed (GFLOPS)} \]
\[ \text{M}_{0,\text{SPE}} = \text{size of SPE local store (bytes)} \]
\[ \text{P}_{\text{SPE}} - \text{M}_{0,\text{SPE}} = \text{SPE to LS memory bandwidth (GB/sec)} \]
\[ \text{Net}_{0.5} = \text{SPE to SPE bandwidth (matrix encoding topology, GB/sec)} \]
\[ \text{MNet}_{1} = \text{PPE,SPE to main memory bandwidth (GB/sec)} \]
\[ \text{M}_{1} = \text{size of main memory (bytes)} \]
Outline

- Parallel Design
- Distributed Arrays
- Kuck Diagrams
- Hierarchical Arrays
- Tasks and Conduits
- Summary
Hierarchical Arrays

- Hierarchical arrays allow algorithms to conform to hierarchical multicore processors.
- Each processor in $P$ controls another set of processors $P$.
- Array local to $P$ is sub-divided among local $P$ processors.
Hierarchical Array and Kuck Diagram

- Array is allocated across $SM_1$ of $P$ processors
- Within each $SM_1$ responsibility of processing is divided among local $P$ processors
- $P$ processors will move their portion to their local $M_0$
Explicitly Local Hierarchical Program

\[ X, Y : \mathbb{R}^{P(P(N)) \times N} \]

\[ Y.loc_p.loc_p = X.loc_p.loc_p + 1 \]

- **Extend** `.loc` notation to explicitly retrieve local part of a local distributed array `.loc.loc` (assumes SPMD on \( P \))
- **Subscript** \( p \) and \( \_p \) provide explicit access to (implicit otherwise)
• Memory constraints are common at the lowest level of the hierarchy
• Blocking at this level allows control of the size of data operated on by each $P_i$
Block Hierarchical Program

\[ X, Y : \mathbb{R}^{P_b(4)(N) \times N} \]

for \( i = 0, X.\text{loc}.\text{loc}.n_{\text{blk}} - 1 \)

\[ Y.\text{loc}.\text{loc}.\text{blk}_i = X.\text{loc}.\text{loc}.\text{blk}_i + 1 \]

- \( P_b(4) \) indicates each sub-array should be broken up into blocks of size 4.
- \( n_{\text{blk}} \) provides the number of blocks for looping over each block; allows controlling size of data on lowest level
Outline

- Parallel Design
- Distributed Arrays
- Kuck Diagrams
- Hierarchical Arrays
- Tasks and Conduits
- Summary

- Basic Pipeline
- Replicated Tasks
- Replicated Pipelines
Tasks and Conduits

Task1$^{S_1}()$

\[ A : \mathbb{R}^{N \times P(N)} \]

\[ \text{Topic12} \]

\[ \text{A} \xrightarrow{n} \]

Conduit

Task2$^{S_2}()$

\[ B : \mathbb{R}^{N \times P(N)} \]

\[ \text{Topic12} \]

\[ \Rightarrow_m B \]

- $S_1$ superscript runs task on a set of processors; distributed arrays allocated relative to this scope
- Pub/sub conduits move data between tasks
Replicated Tasks

Task1\textsuperscript{S1}() \quad \text{Task2}\textsuperscript{S2}() 

\begin{align*}
A & : \mathbb{R}^{N \times P(N)} \\
B & : \mathbb{R}^{N \times P(N)}
\end{align*}

\begin{align*}
\text{A} & \quad \overset{n}{\Rightarrow} \\
\text{B} & \quad \overset{m}{\Rightarrow}
\end{align*}

Replica 0 \quad \text{Replica 1}

\begin{itemize}
\item 2 subscript creates tasks replicas; conduit will round-robin
\end{itemize}
Replicated Pipelines

- $\text{Task1}_{S1}^2()$
  - $A: R^{N \times P(N)}$
  - $\Rightarrow$

- $\text{Task2}_{S2}^2()$
  - $B: R^{N \times P(N)}$
  - $\Rightarrow m B$

- 2 identical subscript on tasks creates replicated pipeline
Summary

- Hierarchical heterogeneous multicore processors are difficult to program

- Specifying how an algorithm is supposed to behave on such a processor is critical

- Proposed notation provides mathematical constructs for concisely describing hierarchical heterogeneous multicore algorithms