Chapter 1: Introduction

Elements of Parallel Computing

Eric Aubanel

Parallel Computing

Solving a computing problem in less time by breaking it down into parts and computing those parts simultaneously.

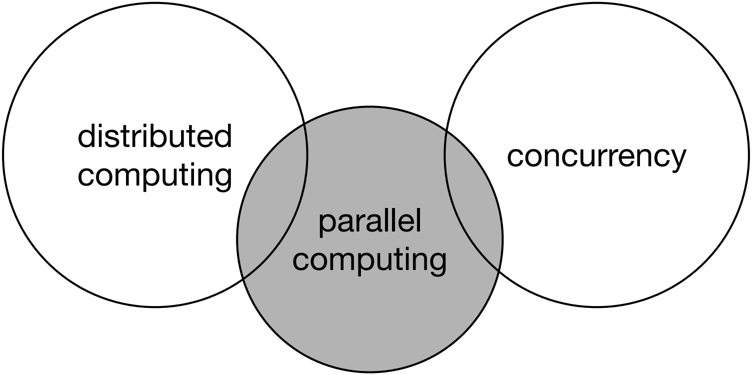
Flynn's Taxonomy

		Instruction streams	
		single	multiple
Data streams	single	SISD	MISD
	multiple	SIMD	MIMD

MIMD

- Shared Memory
- Distributed Memory

Overlapping Discplines



Parallel Computers

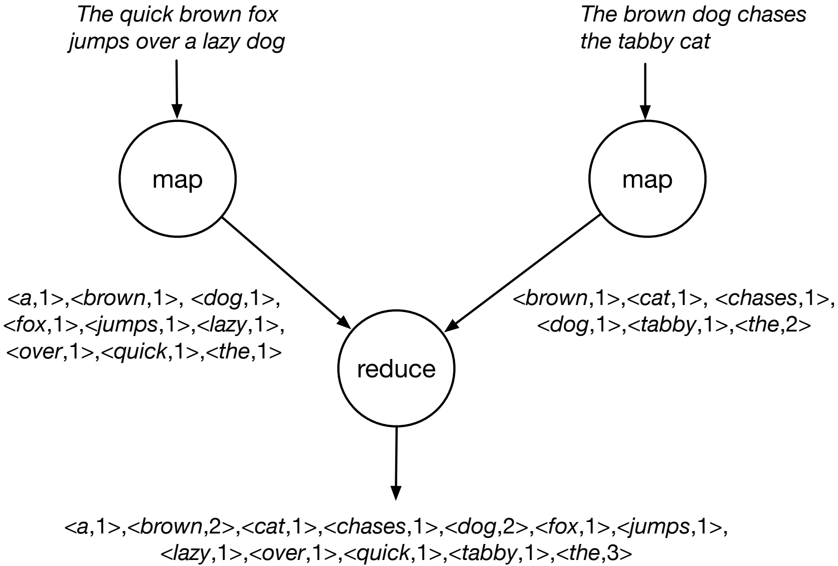
- Multicore processors
  - Olukotun and Hammond, *The Future of Microprocessors*, Queue, September 2005
- Manycore processors
  - lots of cores
  - different architecture than general purpose multicores
  - emphasize throughput
- Multicore/Manycores on a Network
  - Clusters
  - Clouds

Word Count

```
Input: collection of text documents
Output: list of $\langle \text{word}, \text{count} \rangle$ pairs

foreach document in collection do
    foreach word in document do
        if first occurrence of word then
            add $\langle \text{word}, 1 \rangle$ to ordered list
        else
            increment count in $\langle \text{word}, \text{count} \rangle$
        end
    end
end
```

Word Count with MapReduce



The quick brown fox jumps over a lazy dog

The brown dog chases the tabby cat

<*a*,1>,<*brown*,1>, <*dog*,1>,
<*fox*,1>,<*jumps*,1>,<*lazy*,1>,
<*over*,1>,<*quick*,1>,<*the*,1>

<*brown*,1>,<*cat*,1>, <*chases*,1>,
<*dog*,1>,<*tabby*,1>,<*the*,2>

<*a*,1>,<*brown*,2>,<*cat*,1>,<*chases*,1>,<*dog*,2>,<*fox*,1>,<*jumps*,1>,
<*lazy*,1>,<*over*,1>,<*quick*,1>,<*tabby*,1>,<*the*,3>

Parallel Programming Models

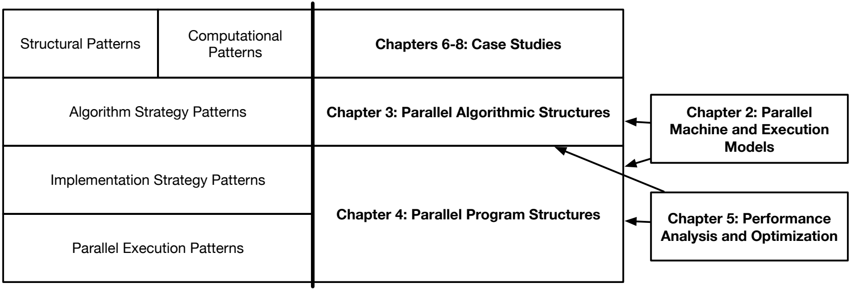
- Implicit
  - MapReduce
- Semi-Implicit

```
parallel for $i \leftarrow 0$ to $n-1$ do
    $c[i] \leftarrow a[i] + b[i]$
end
```

- Explicit

```
scatter($0$, $a$, $n/p$, $aLoc$)
scatter($0$, $b$, $n/p$, $bLoc$)
for $i \leftarrow 0$ to $n/p-1$ do
    $cLoc[i] \leftarrow aLoc[i] + bLoc[i]$
end
gather($0$, $c$, $n/p$, $cLoc$)
```

Berkeley's Our Pattern Language (OPL)



Structural Patterns

Computational Patterns

Chapters 6-8: Case Studies

Algorithm Strategy Patterns

Chapter 3: Parallel Algorithmic Structures

Chapter 2: Parallel Machine and Execution Models

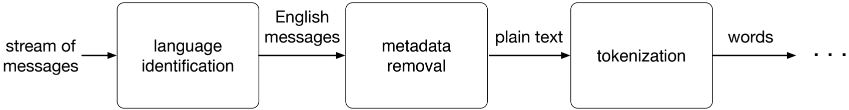
Implementation Strategy Patterns

Chapter 4: Parallel Program Structures

Chapter 5: Performance Analysis and Optimization

Parallel Execution Patterns

Structural Pattern: Pipe and Filter



stream of messages

language identification

English messages

metadata removal

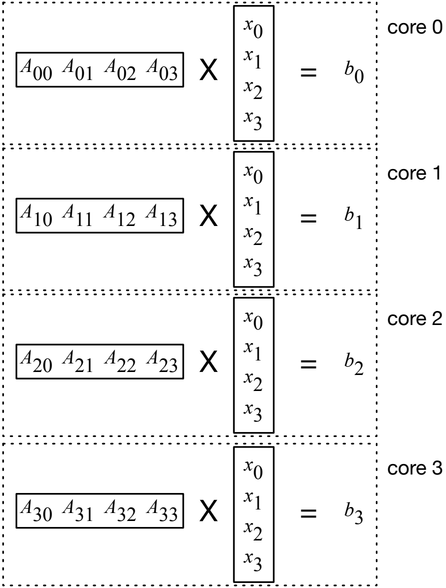
plain text

tokenization

words

. . .

Computational Pattern: Matrix × Vector



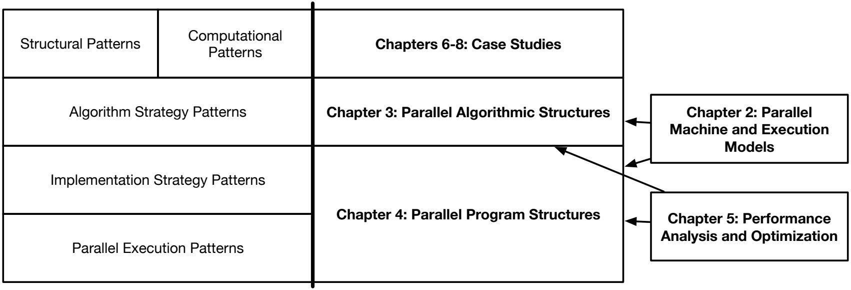
$\boxed{A_{00}\ A_{01}\ A_{02}\ A_{03}}$ X $\boxed{\begin{matrix}x_0\\x_1\\x_2\\x_3\end{matrix}}$ = b_0 core 0

$\boxed{A_{10}\ A_{11}\ A_{12}\ A_{13}}$ X $\boxed{\begin{matrix}x_0\\x_1\\x_2\\x_3\end{matrix}}$ = b_1 core 1

$\boxed{A_{20}\ A_{21}\ A_{22}\ A_{23}}$ X $\boxed{\begin{matrix}x_0\\x_1\\x_2\\x_3\end{matrix}}$ = b_2 core 2

$\boxed{A_{30}\ A_{31}\ A_{32}\ A_{33}}$ X $\boxed{\begin{matrix}x_0\\x_1\\x_2\\x_3\end{matrix}}$ = b_3 core 3

Berkeley's Our Pattern Language (OPL)



Structural Patterns

Computational Patterns

Algorithm Strategy Patterns

Implementation Strategy Patterns

Parallel Execution Patterns

Chapters 6-8: Case Studies

Chapter 3: Parallel Algorithmic Structures

Chapter 4: Parallel Program Structures

Chapter 2: Parallel Machine and Execution Models

Chapter 5: Performance Analysis and Optimization