Microsoft® Research
Faculty Summit 2010

A New Approach to Concurrency and Parallelism (Part 1)

Ade Miller (adem@microsoft.com)
Senior Development Manager
Microsoft patterns & practices

Microsoft patterns & practices

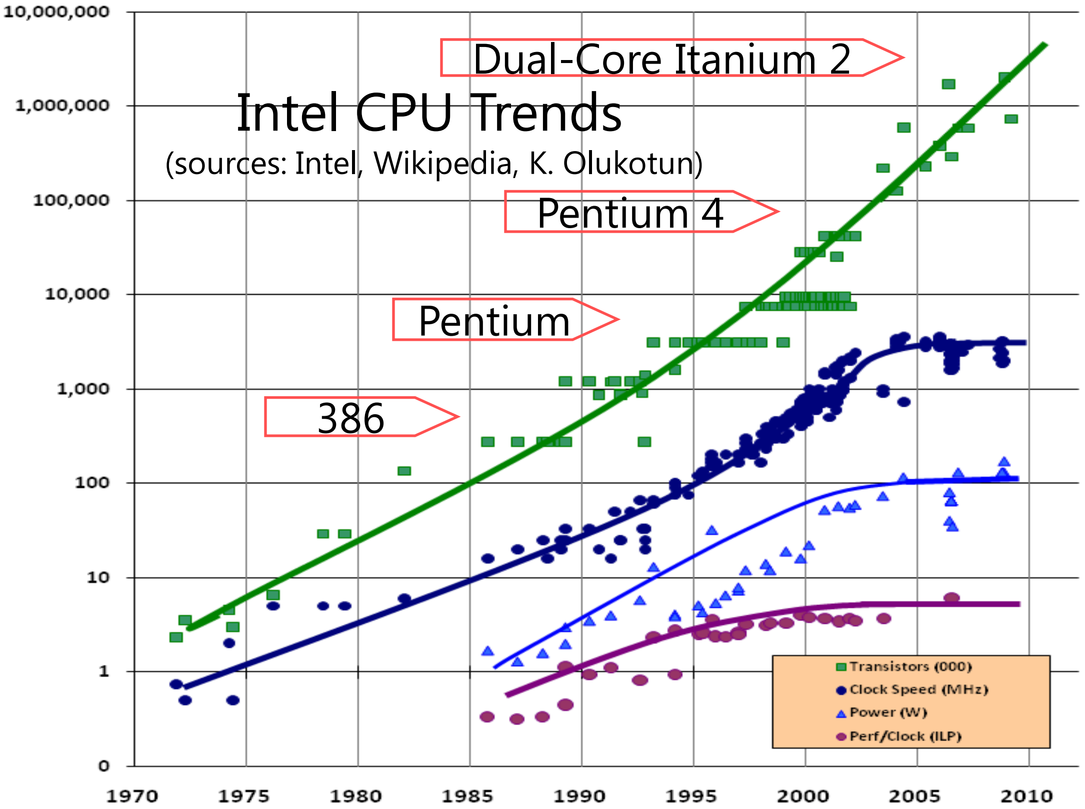
“Save time and reduce risk on your software development projects by incorporating patterns & practices, Microsoft's applied engineering guidance that includes both production quality source code and documentation.”

http://msdn.com/practices/

Introduction

- Why we should care about parallel programming
- Where to start
- Patterns for parallelism
- Conclusions

Why We Should Care?



Then:
Faster clocks

Now:
More cores

End of the Free Lunch

The End of The Free Lunch

- Although driven by hardware changes,
  the parallel revolution is **primarily a software revolution.**
- Parallel hardware is not “more of the same.”
- Software is the gating factor.
- Software requires the most changes to regain the “free lunch.”
- Hardware parallelism is coming,
  more and sooner than most people yet believe.

Where Should I Start?

If you talk to developers you’ll hear…

- “Avoid multithreaded code”
- “Parallel programming is hard”
- “It’s for the experts”
- “Where’s my magic parallelizing compiler?”

How do we help them succeed in this new parallel world?

How Could We Help?

- Looked at:
  - Our Pattern Language (OPL)
    Berkeley, Illinois, Intel, Microsoft, Samsung, U. Victoria, U Florida, Bosch…
  - Patterns for Parallel Programming
    Timothy G. Mattson, Beverly A. Sanders, Berna L. Massingill
  - White Papers from Microsoft & Intel
  - …

- New frameworks & tools in Visual Studio 2010
  - Task Parallel Library (TPL)
  - Parallel Debugger
  - Parallel Profiler

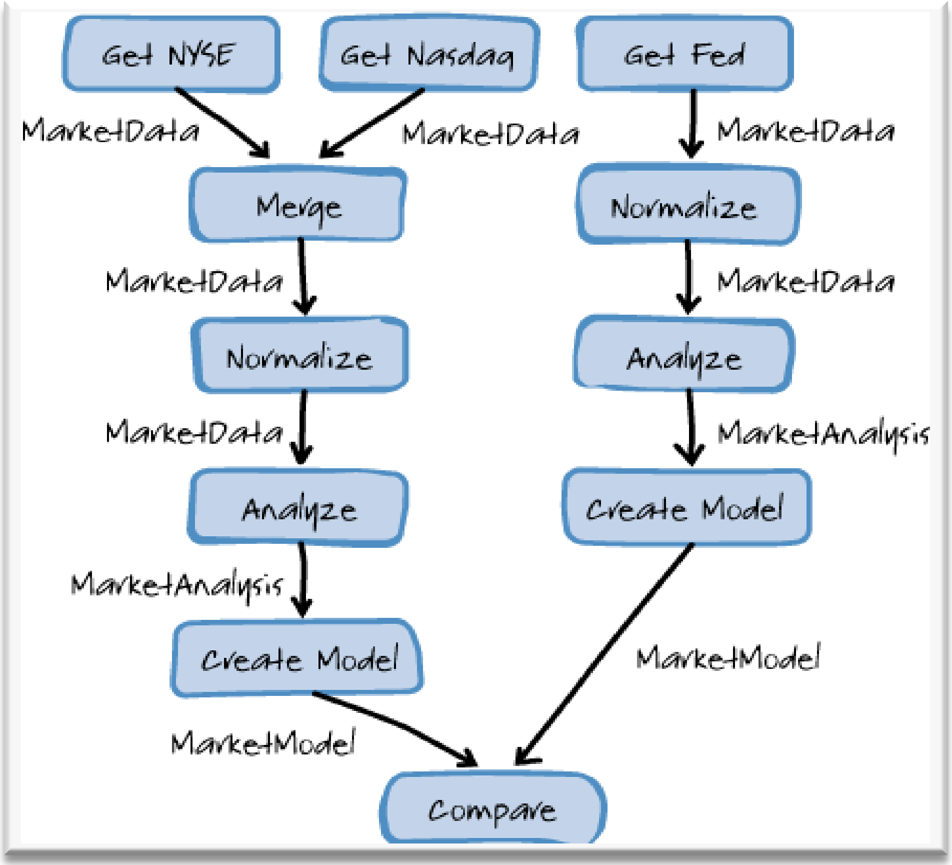
An Example: Adatum-Dashboard

Let’s look at a “real” application…

- A Financial application for portfolio risk analysis
- Look at large chunks of recent and historical data
- Compare models with market conditions

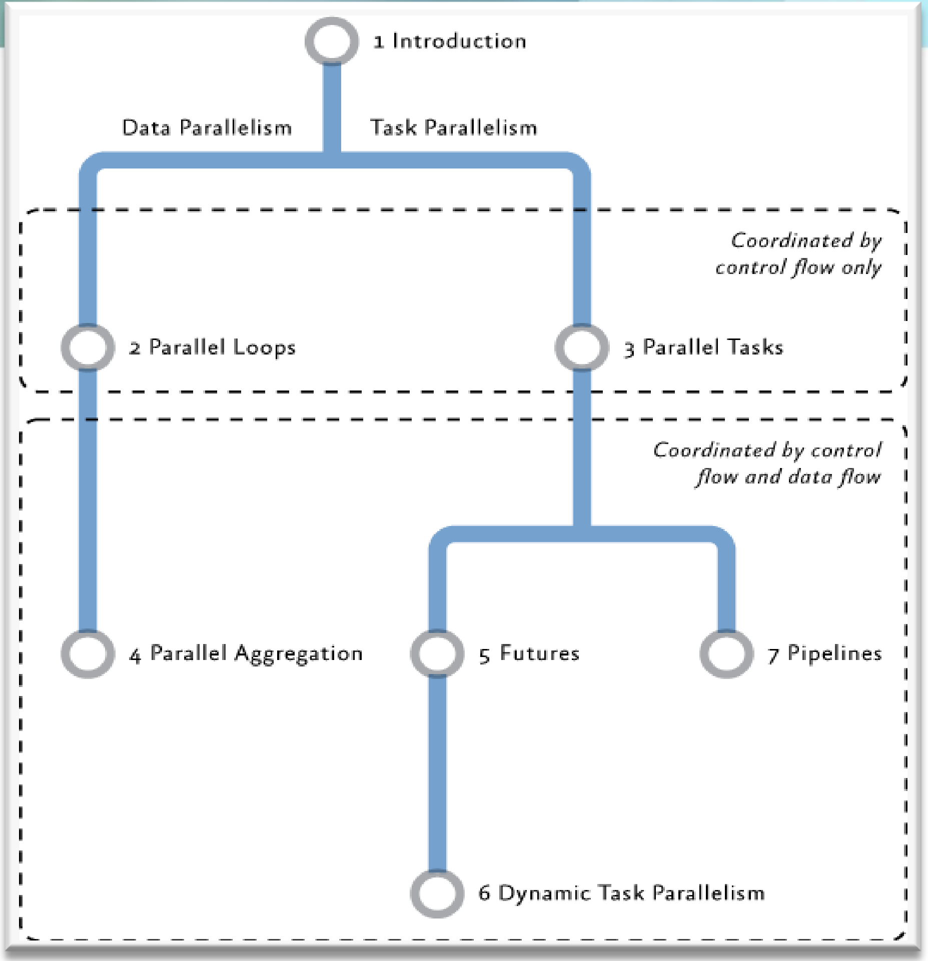
- Source code available: http://parallelpatterns.codeplex.com/

The Adatum Dashboard Scenario



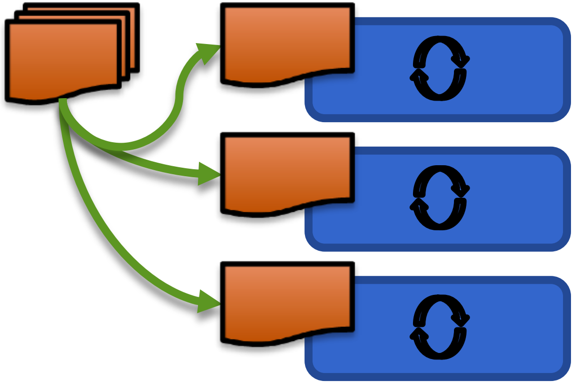
Finding Potential Parallelism

- Tasks vs. Data
- Control Flow
- Control and Data Flow



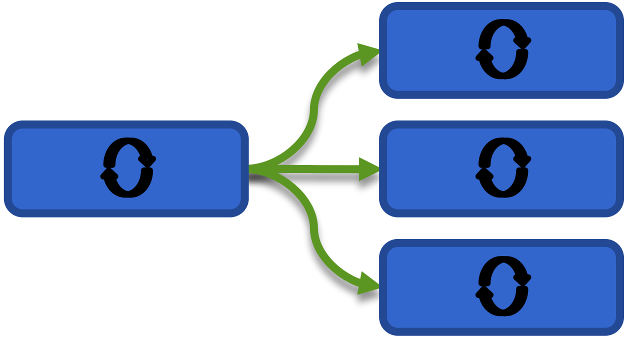
Data Parallelism

- Data "chunk" size?
  - Too big – under utilization
  - Too small – thrashing
- Chunk layout?
  - Cache and cache line size
  - False cache sharing
- Data dependencies?



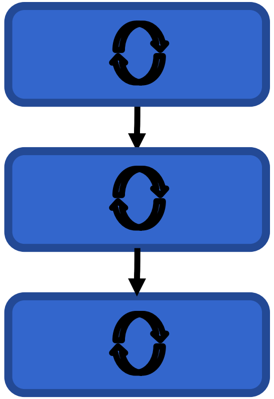
Task Parallelism

- Enough tasks?
  - Too many – thrashing
  - Too few – under utilization
- Work per task?
  - Small workloads
  - Variable workloads
- Dependencies between tasks?
  - Removable
  - Separable
  - Read only or read/write



Control and Data Flow

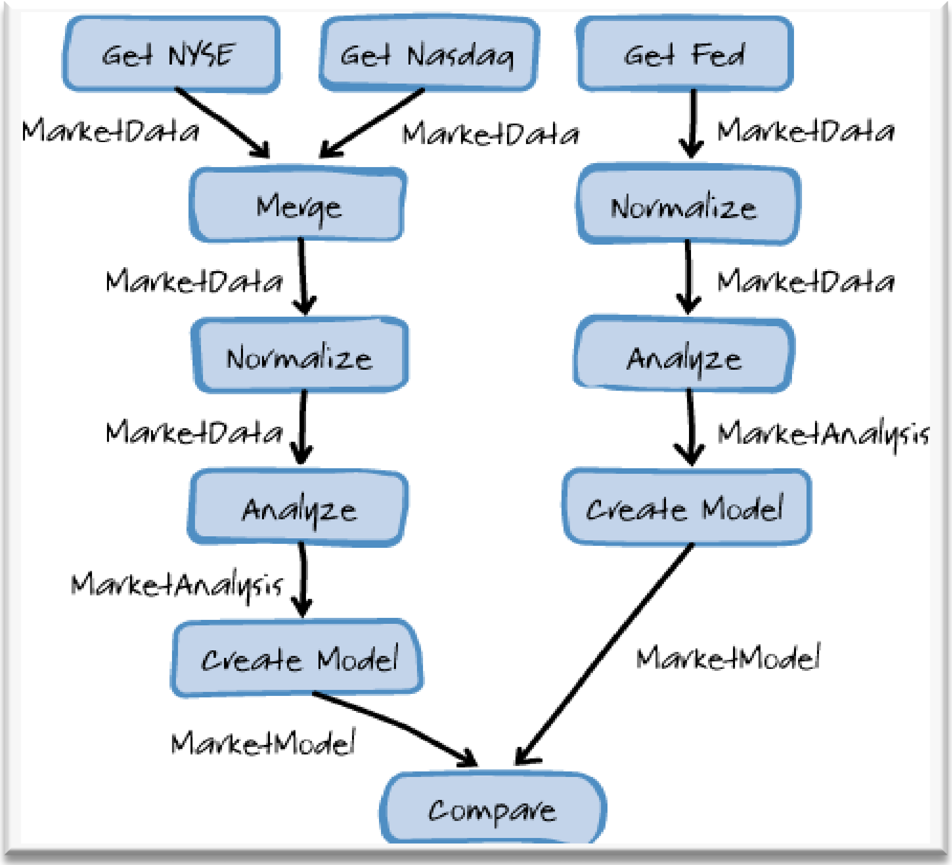
- Task constraints
  - Temporal: A → B
  - Simultaneous: A ↔ B
  - None: A B
- External constraints
  - I/O read or write order
  - Message or list output order
- Linear and irregular orderings
  - Pipeline
  - Futures
  - Dynamic Tasks



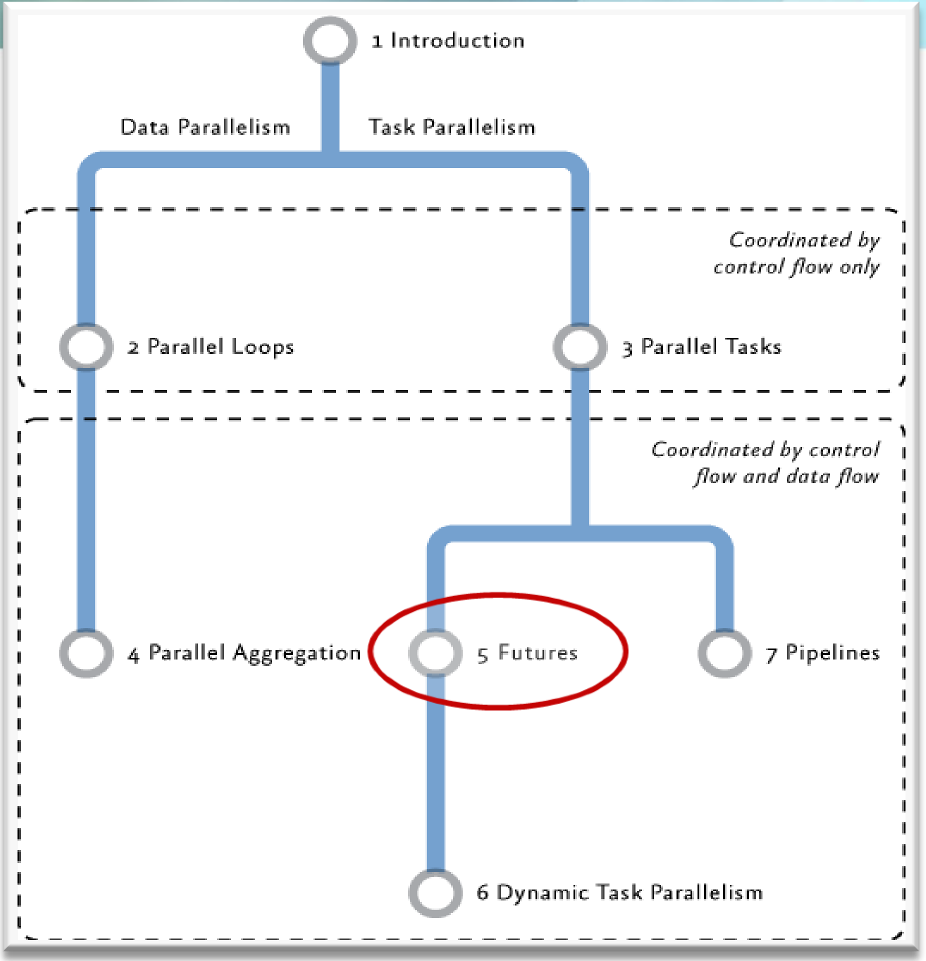
Solution Forces

- Flexibility:
  - Easy to modify for different scenarios
  - Runs on different types of hardware
- Efficiency:
  - Time spent managing the parallelism vs. time gained from utilizing more processors or cores
  - Performance improves as more cores or processors are added – Scaling
- Simplicity:
  - The code can be easily debugged and maintained

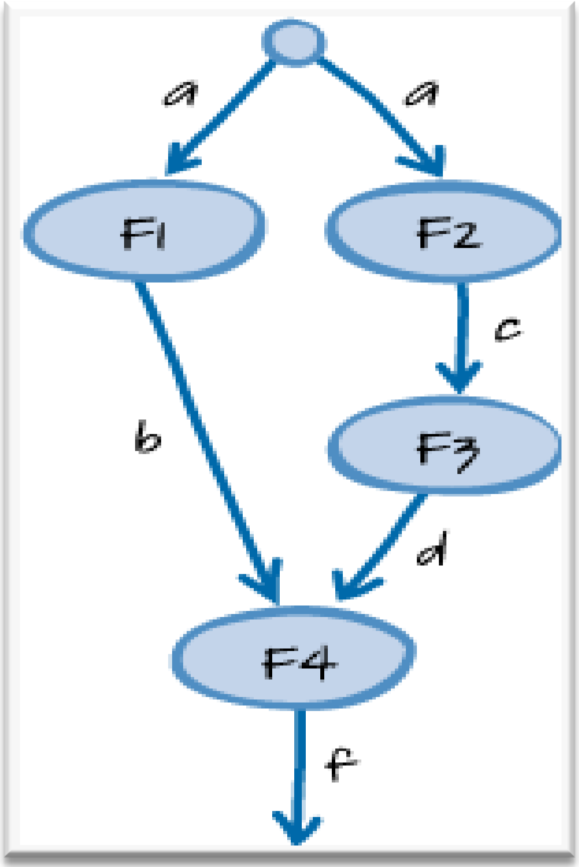
The Adatum Dash Scenario



The Futures Pattern



The Futures Pattern



"Does the ordering of steps in your algorithm depend on data flow constraints?"

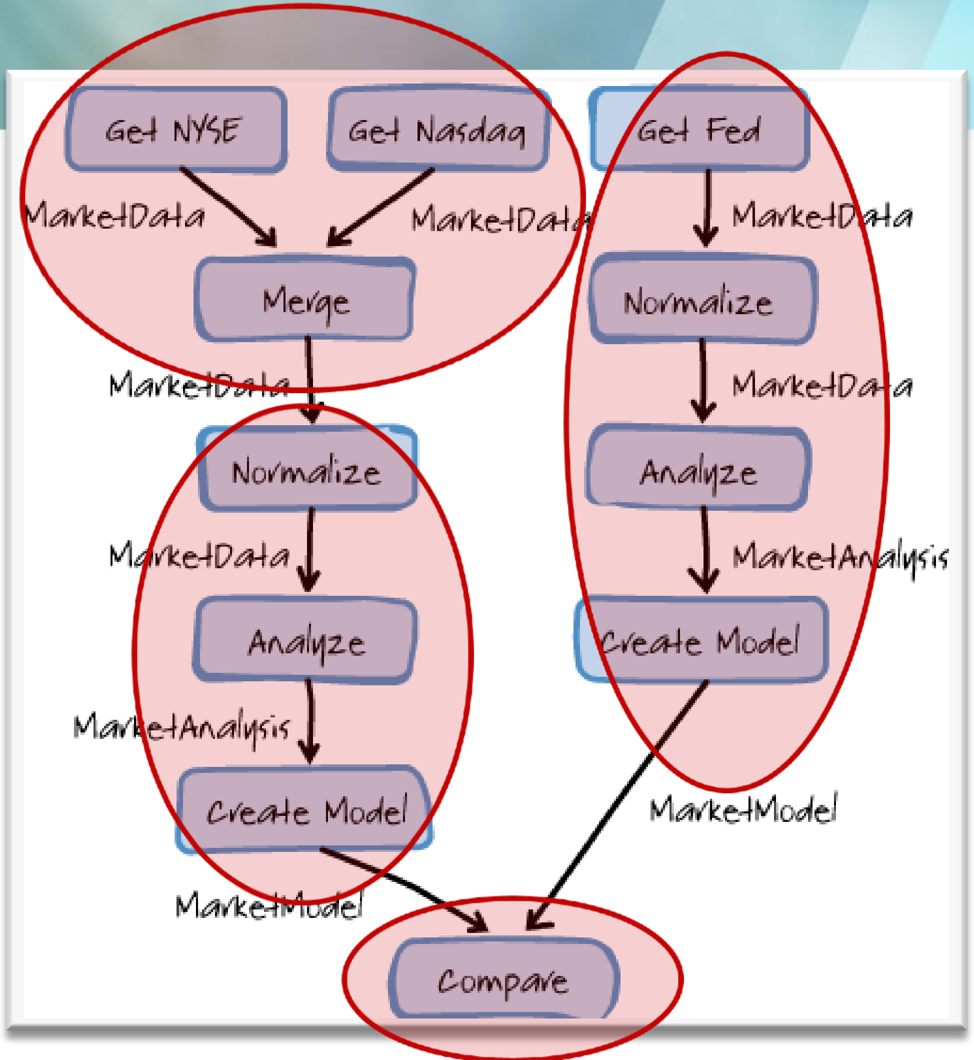
- Directed Acyclic Graph
- Dependencies between tasks
- F4 depends on the result of F1 & F3 etc

- Also called "Task Graph"

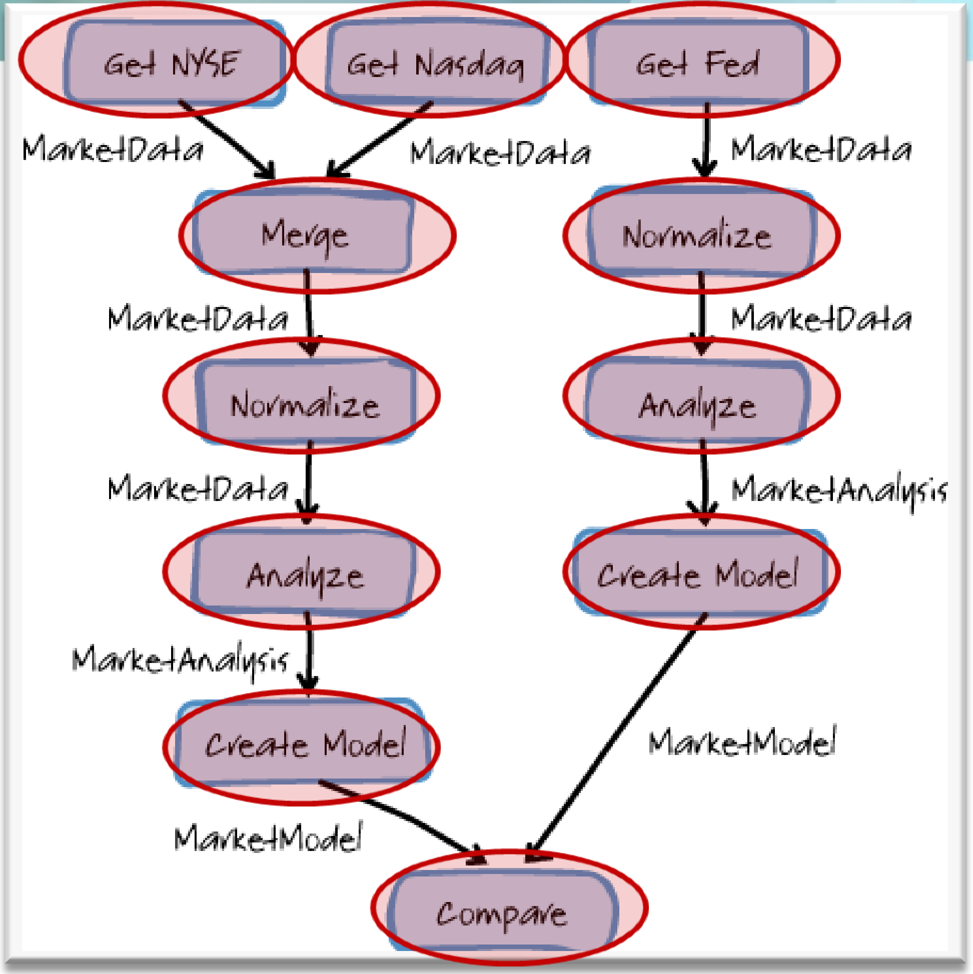
Task Size and Granularity

- Variable size tasks – harder to balance
- Small tasks – more overhead; management and communication
- Large tasks – less potential for utilization
- Hardware specific – more tasks than cores

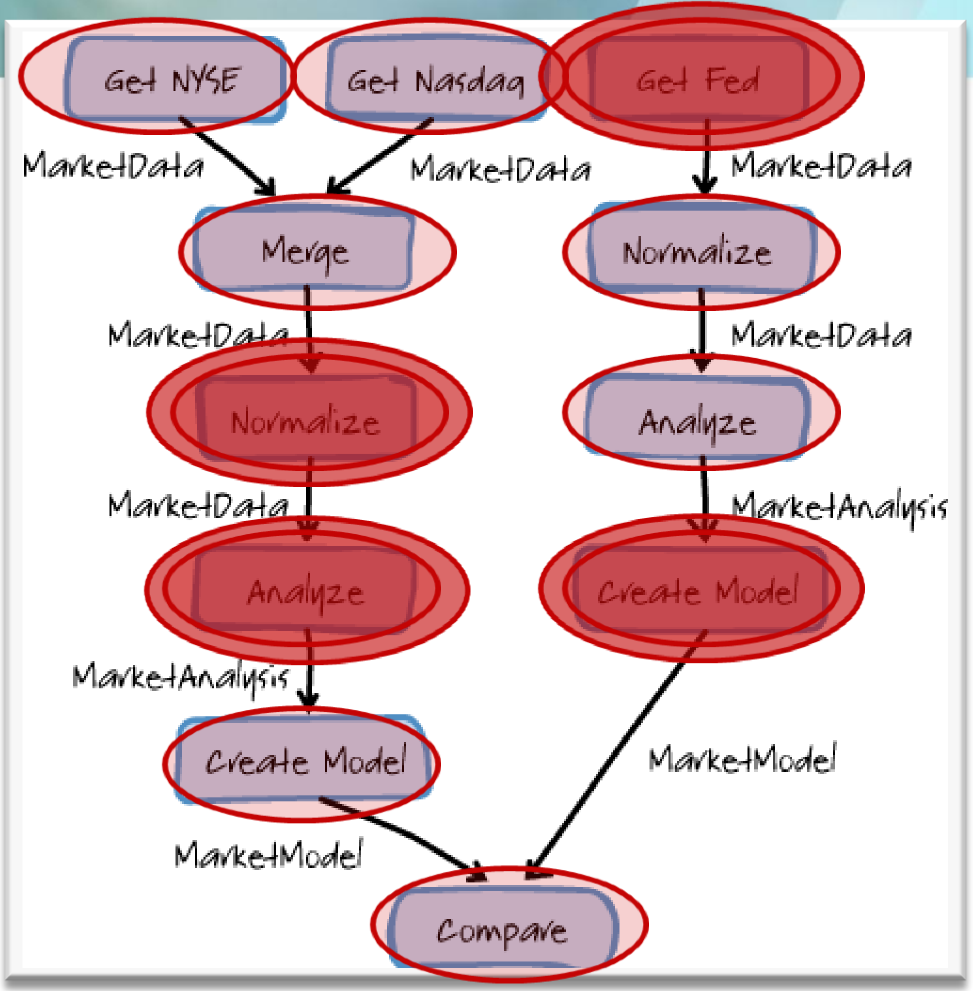
Course Grained Partition



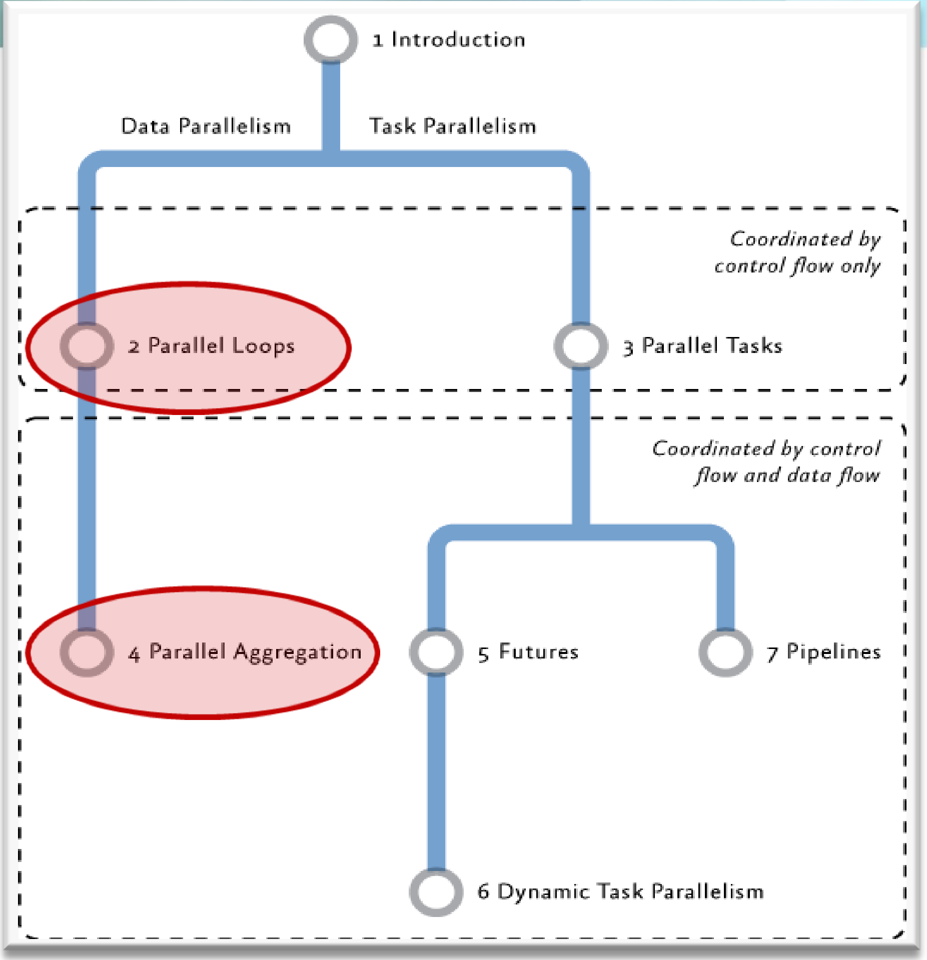
Fine Grained Partition



Finer Grained Partition

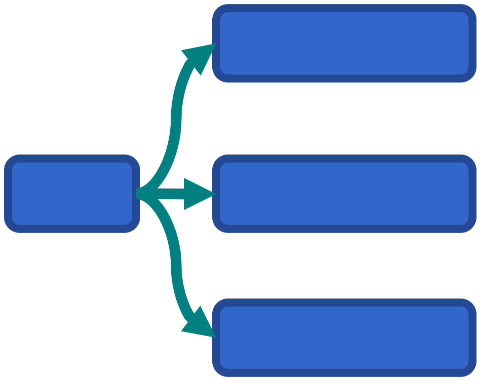


Data Parallelism Patterns



The Parallel Loop Pattern

“Do you have sequential loops where there's no communication among the steps of each iteration?”



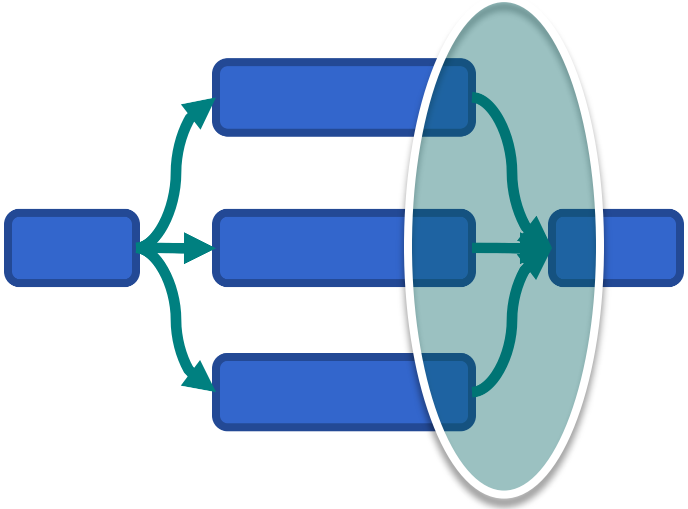
- A very common problem!

The Parallel Aggregation Pattern

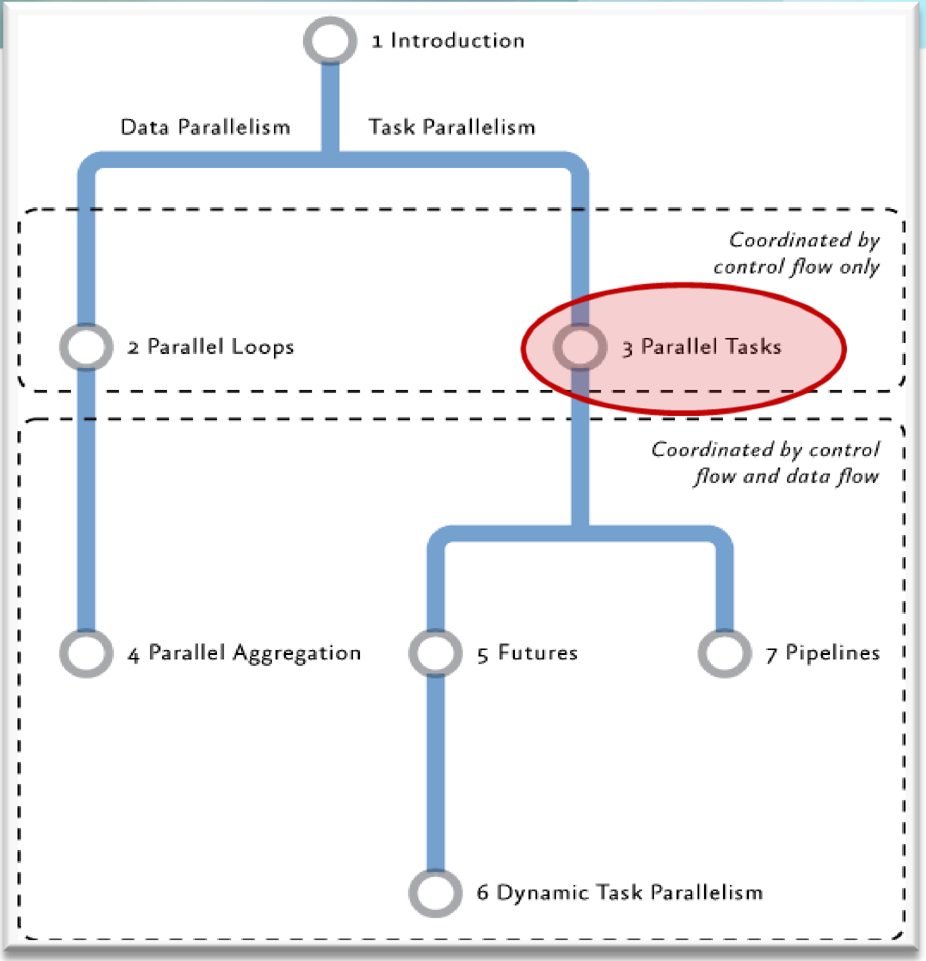
“Do you need to summarize data by applying some kind of combination operator? Do you have loops with steps that are not fully independent?”

- Calculate sub-problem result per task
- Merge results later
- Reduces need for locking

- “Reduction” or “map/reduce”

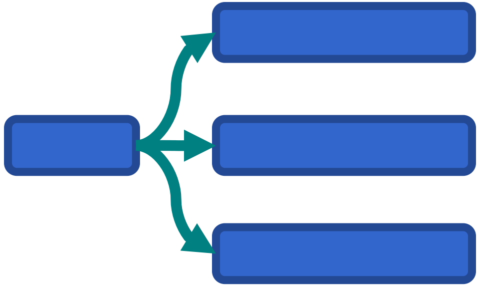


The Parallel Tasks Pattern



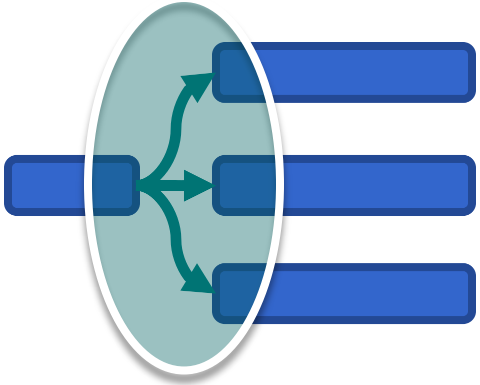
The Parallel Tasks Pattern

“Do you have specific units of works with well-defined control dependencies?”



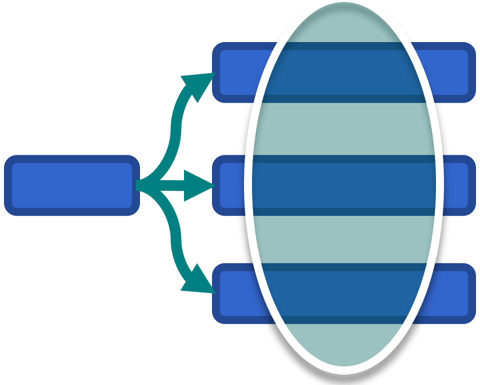
Partitioning

- How do we divide up the workload?
    - Fixed workloads
    - Variable workloads
- Workload size
    - Too large – hard to balance
    - Too small – communication may dominate



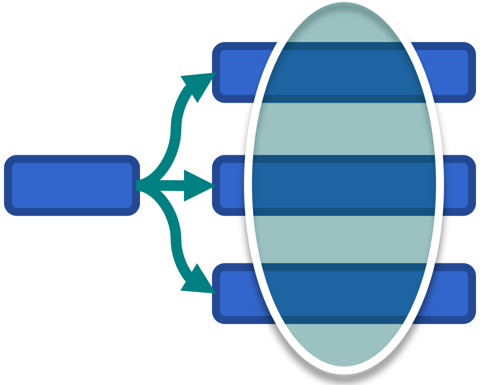
Workload Balancing

- Static allocation:
  - By blocks
  - By index (interleaved)
  - Guided
- Dynamic work allocation
  - known and unknown task sizes
  - Task queues
  - Work stealing
- The TPL does a lot of this work for you



Sharing State and Synchronization

- Don’t share!
- Read only data
- Data isolation
- Synchronization



Conclusions

Success =

- Frameworks and runtimes
  - Task Parallel Library for .NET
  - Parallel Patterns Library & Asynchronous Agents Library for Visual C++
- Tools
  - Visual Studio 2010
- **Guidance!**
  - Patterns
  - Examples

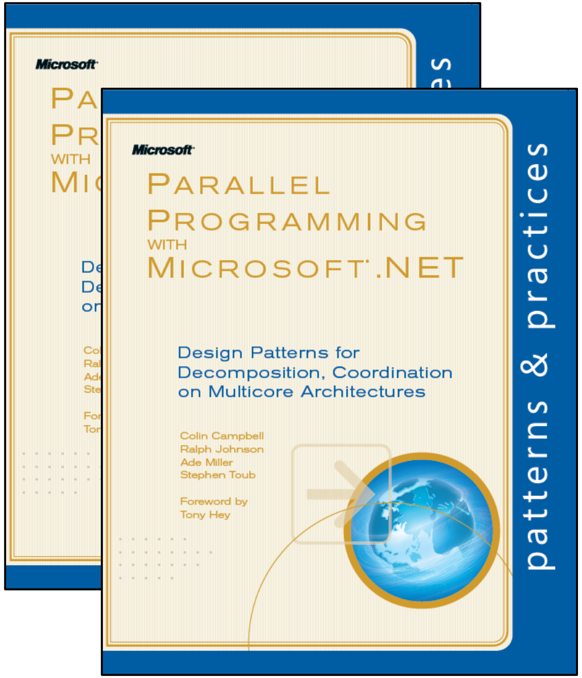
Our Book

Programming with Microsoft .NET:
Design Patterns for Decomposition and Coordination on Multicore Architectures

Colin Campbell, Ralph Johnson, Ade Miller and Stephen Toub
Foreword by Tony Hey

Goal: Help developers make the most of the new parallel features in Visual Studio 2010

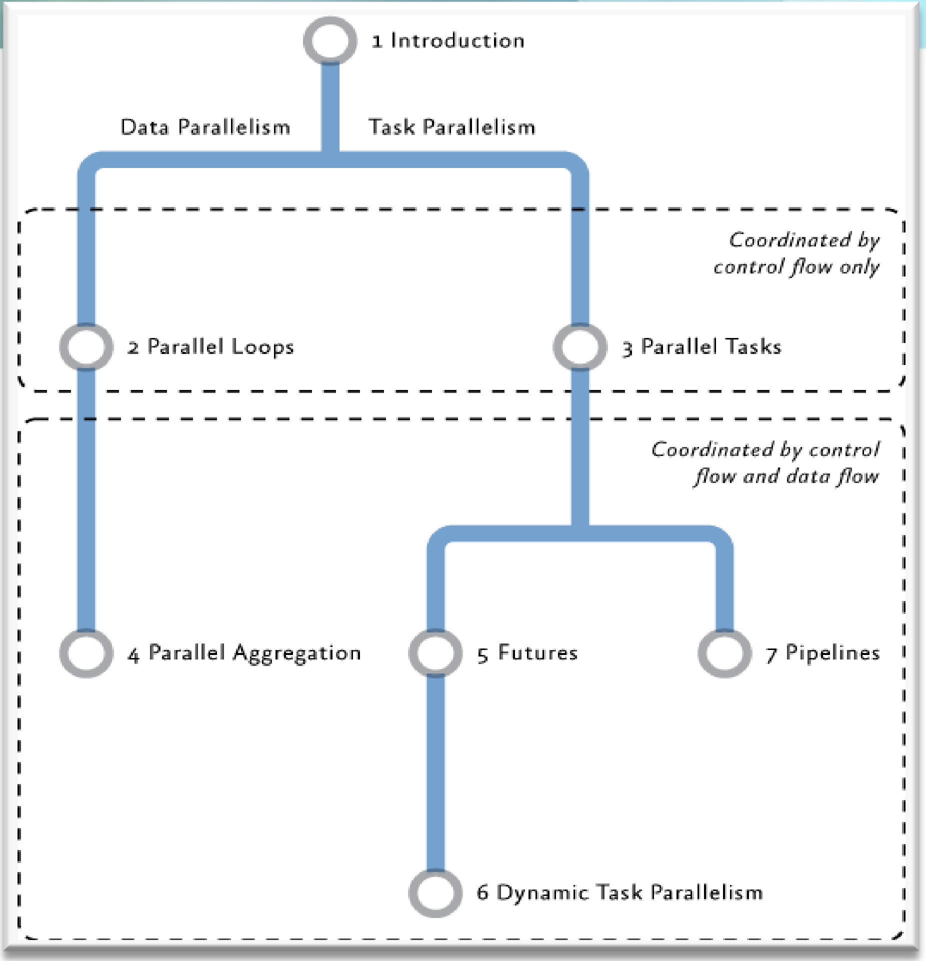
Due for release late summer 2010.
http://parallelpatterns.codeplex.com/



Microsoft
PARALLEL PROGRAMMING WITH MICROSOFT.NET
Design Patterns for Decomposition, Coordination on Multicore Architectures
Colin Campbell
Ralph Johnson
Ade Miller
Stephen Toub
Foreword by Tony Hey
patterns & practices

Our Book

- Introductory material
- Six key patterns
- Adapting OO patterns
- Debugging and profiling
- Technology Overview

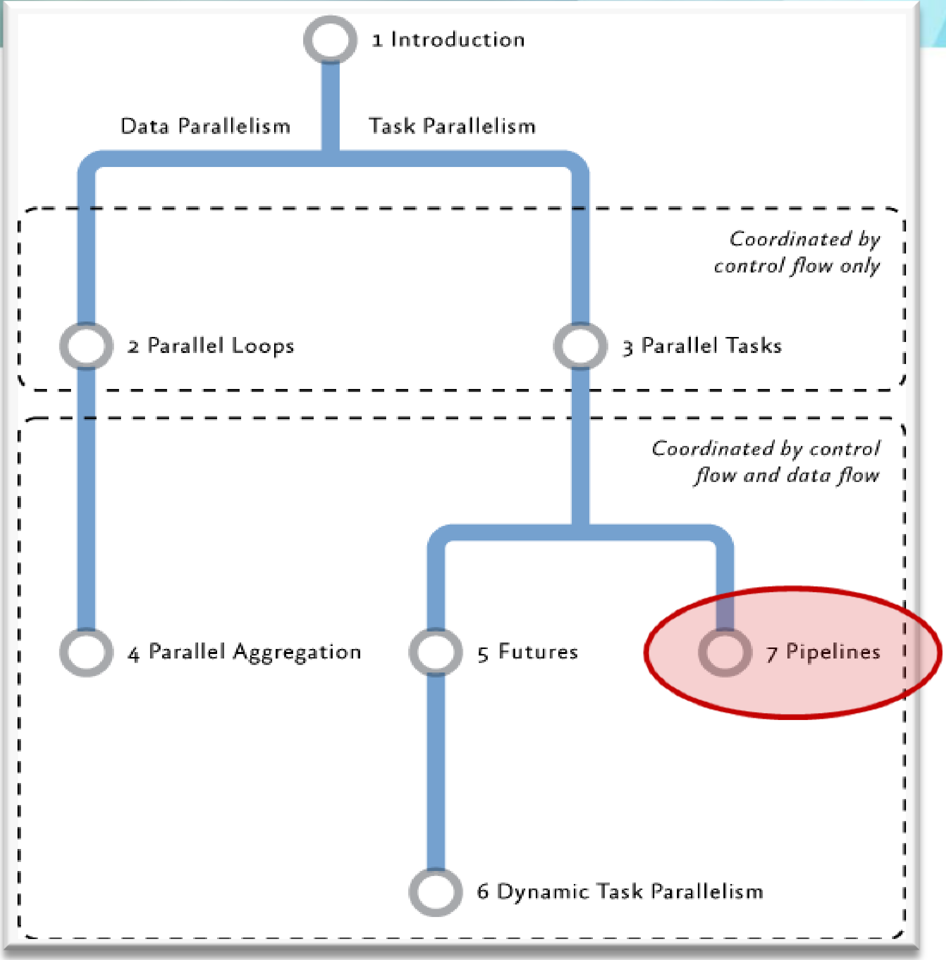


Acknowledgements

- UPCRC Initiative
  - Illinois
  - Intel
  - UC Berkeley
- Microsoft Research
- Numerous others who provided feedback

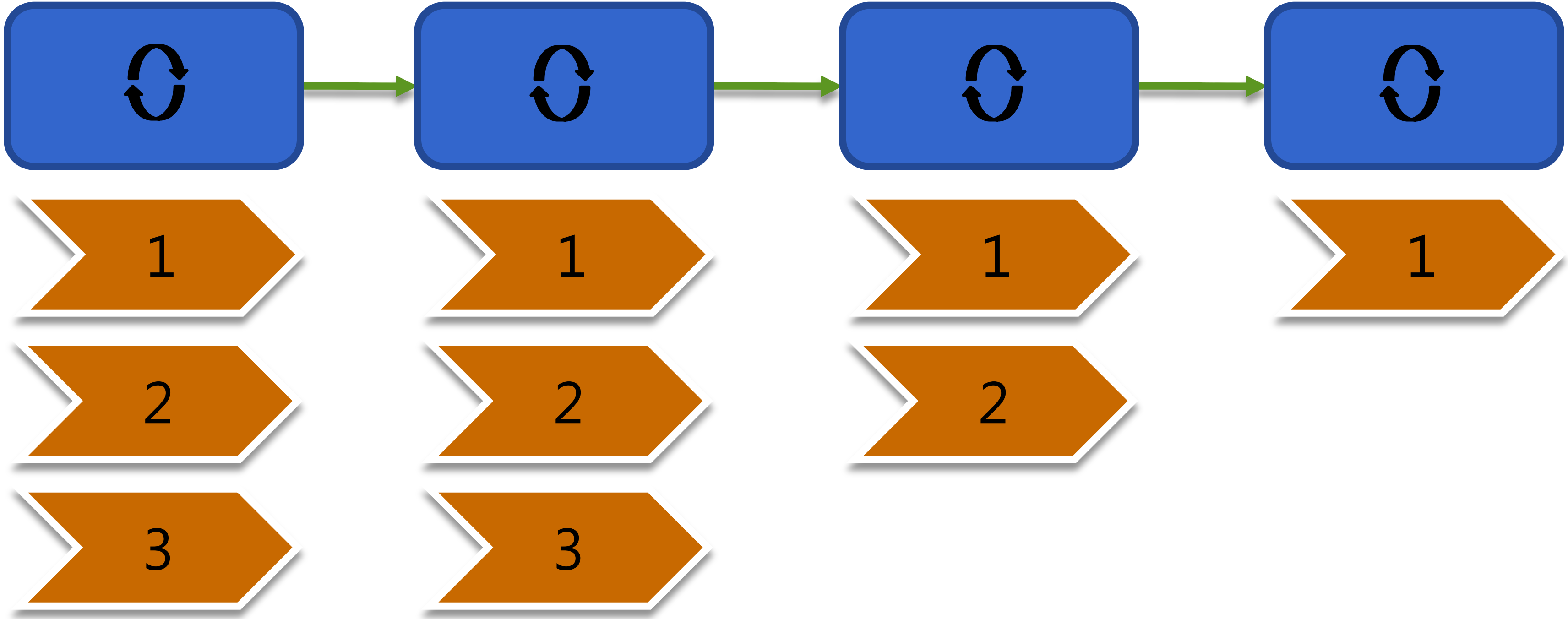
Supporting Material

The Pipeline Pattern



The Pipeline Pattern

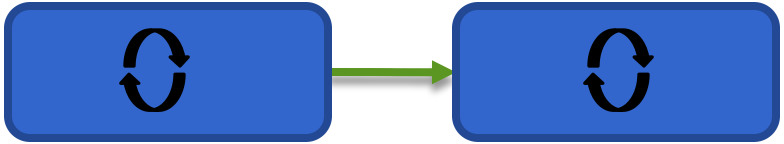
"Does your application perform a sequence of operations repetitively? Does the input data have streaming characteristics?"



The Producer/Consumer Pattern

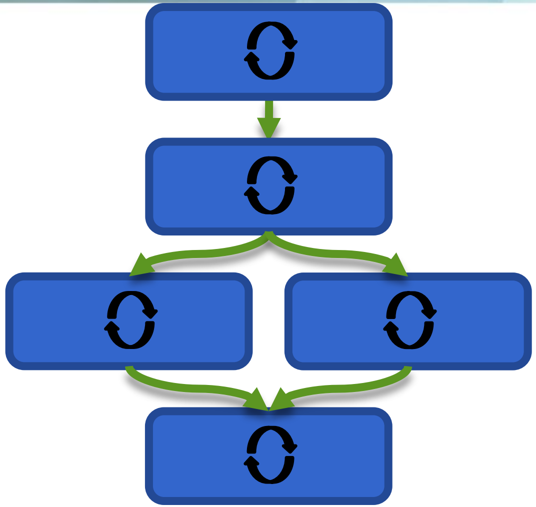
- Organize by Ordering

- Producers… produce!
  - Block when buffer full
- Consumers… consume!
  - Block when buffer empty



Workload Balancing

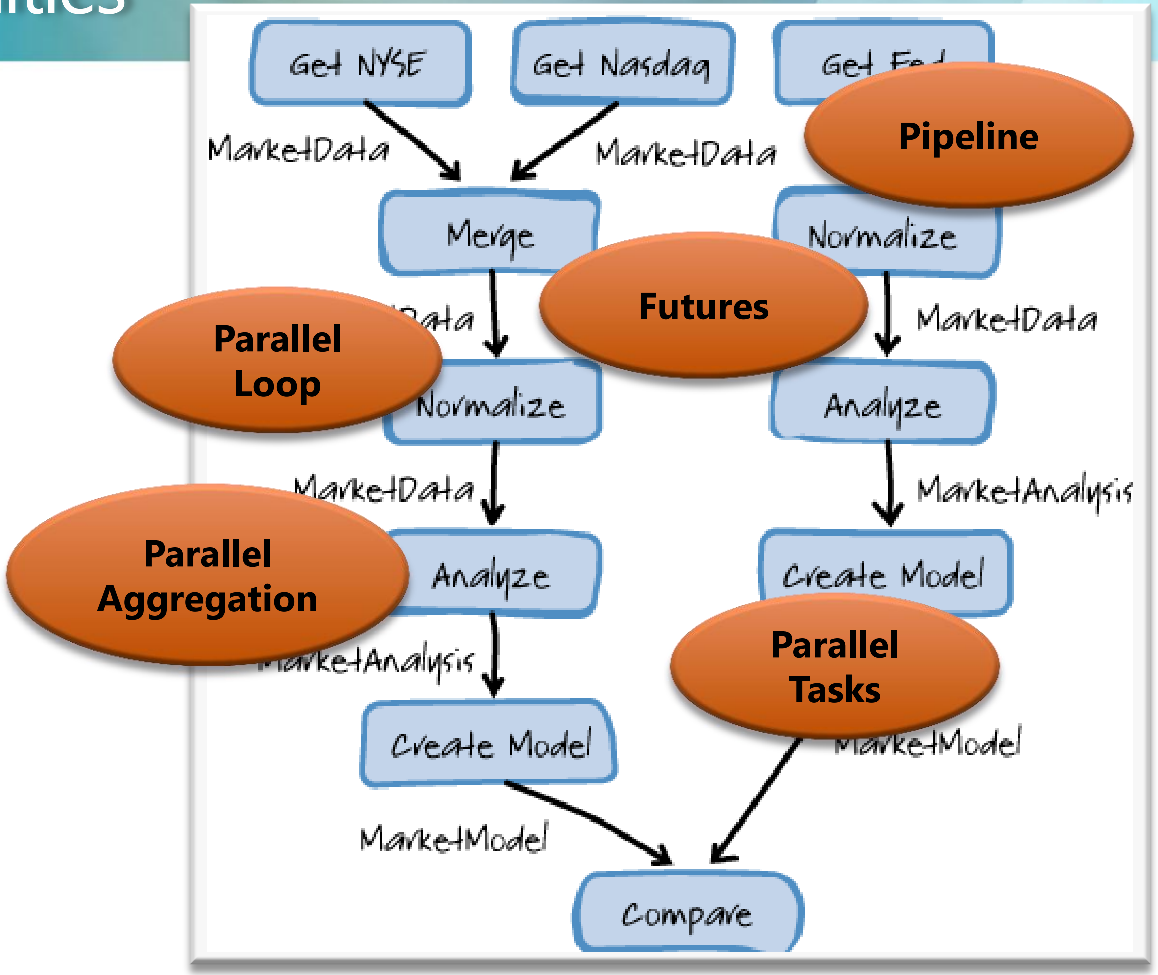
- Pipeline length
  - Long – High throughput
  - Short – Low latency
- Stage workloads
  - Equal – linear pipeline
  - Unequal – nonlinear pipeline



Passing Data Down the Pipe

- Shared queue(s)
  - Large queue items – under utilization
  - Small queue items – locking overhead

Parallelism Opportunities



Pipeline

Futures

Parallel Loop

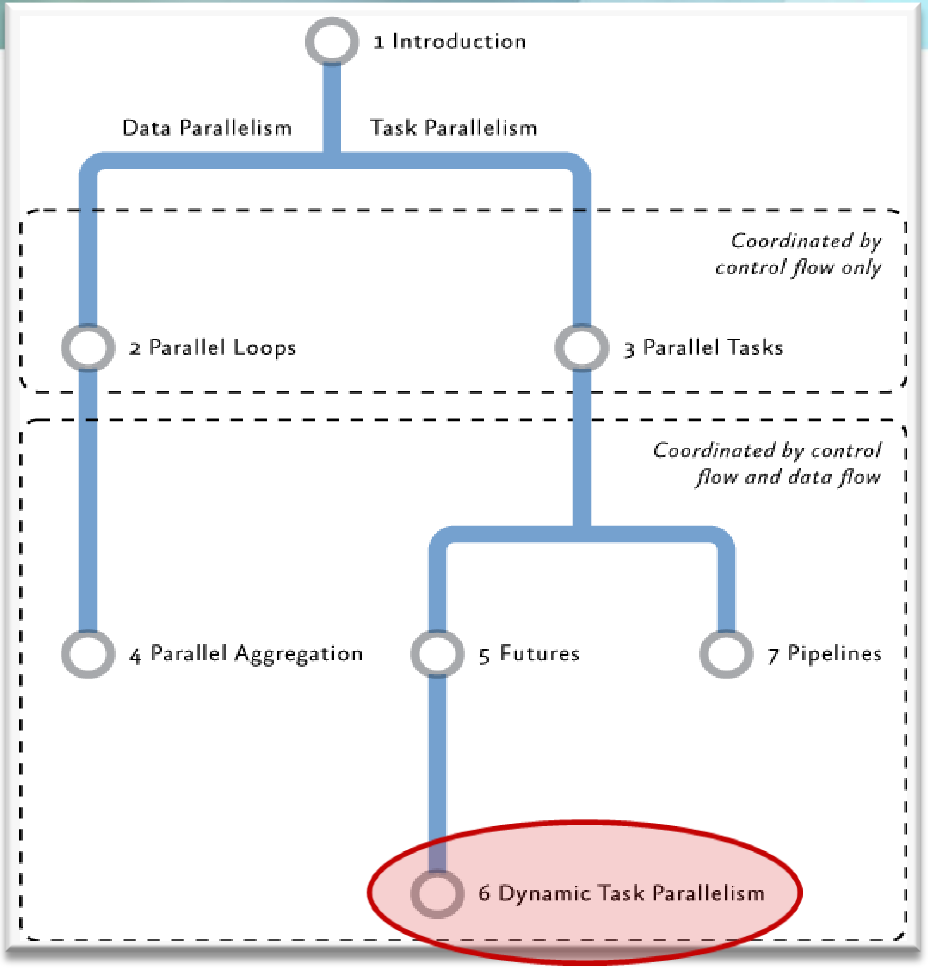
Parallel Aggregation

Parallel Tasks

What About Recursive Problems?

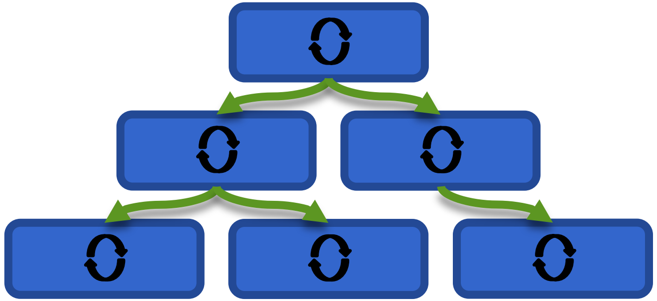
- Many problems can be tackled using recursion:
  - Task based: Divide and Conquer
  - Data based: Recursive Data

Dynamic Task Parallelism



The Dynamic Task Parallelism Pattern

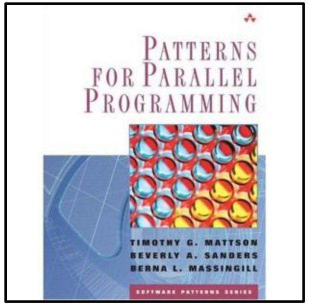
“Does your algorithm divide the problem domain dynamically during the run? Do you operate on recursive data structures such as graphs?”



Workload Balancing

- Deep trees – thrashing
  - Limit the tree depth
- Shallow trees – under utilization
- Unbalanced Trees – under utilization

Other Resources



Books

- Patterns for Parallel Programming – Mattson, Sanders & Massingill
- Design Patterns – Gamma, Helm, Johnson & Vlissides
- Head First Design Patterns – Freeman & Freeman
- Patterns of Enterprise Application Architecture – Fowler

A Brain-Friendly Guide
Head First Design Patterns
O'REILLY
The Addison-Wesley Signature Series
PATTERNS OF ENTERPRISE APPLICATION ARCHITECTURE
MARTIN FOWLER
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Research

- A Pattern Language for Parallel Programming ver2.0
- ParaPLOP - Workshop on Parallel Programming Patterns

- My Blog: http://ademiller.com/tech/
(Decks etc.)

Microsoft® Research

Faculty Summit 2010

Microsoft®