EE382N (20): Computer Architecture
Parallelism and Locality
Fall 2009

Lecture 11 – Patterns for Parallel Programming (III)

Mattan Erez

UT ECE

The University of Texas at Austin

Credits

- Most of the slides courtesy Dr. Rodric Rabbah (IBM)
  - Taken from 6.189 IAP taught at MIT in 2007.

Patterns for Parallelizing Programs

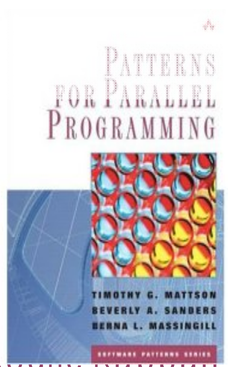
4 Design Spaces

Algorithm Expression

- Finding Concurrency
  - Expose concurrent tasks
- Algorithm Structure
  - Map tasks to processes to exploit parallel architecture

Software Construction

- Supporting Structures
  - Code and data structuring patterns
- Implementation Mechanisms
  - Low level mechanisms used to write parallel programs



Patterns for Parallel Programming. Mattson, Sanders, and Massingill (2005).

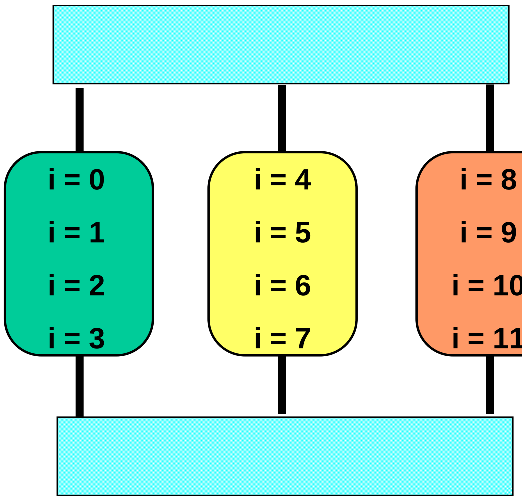
Code Supporting Structures

- Loop parallelism
- Master/Worker
- Fork/Join
- SPMD
- ***Map/Reduce***

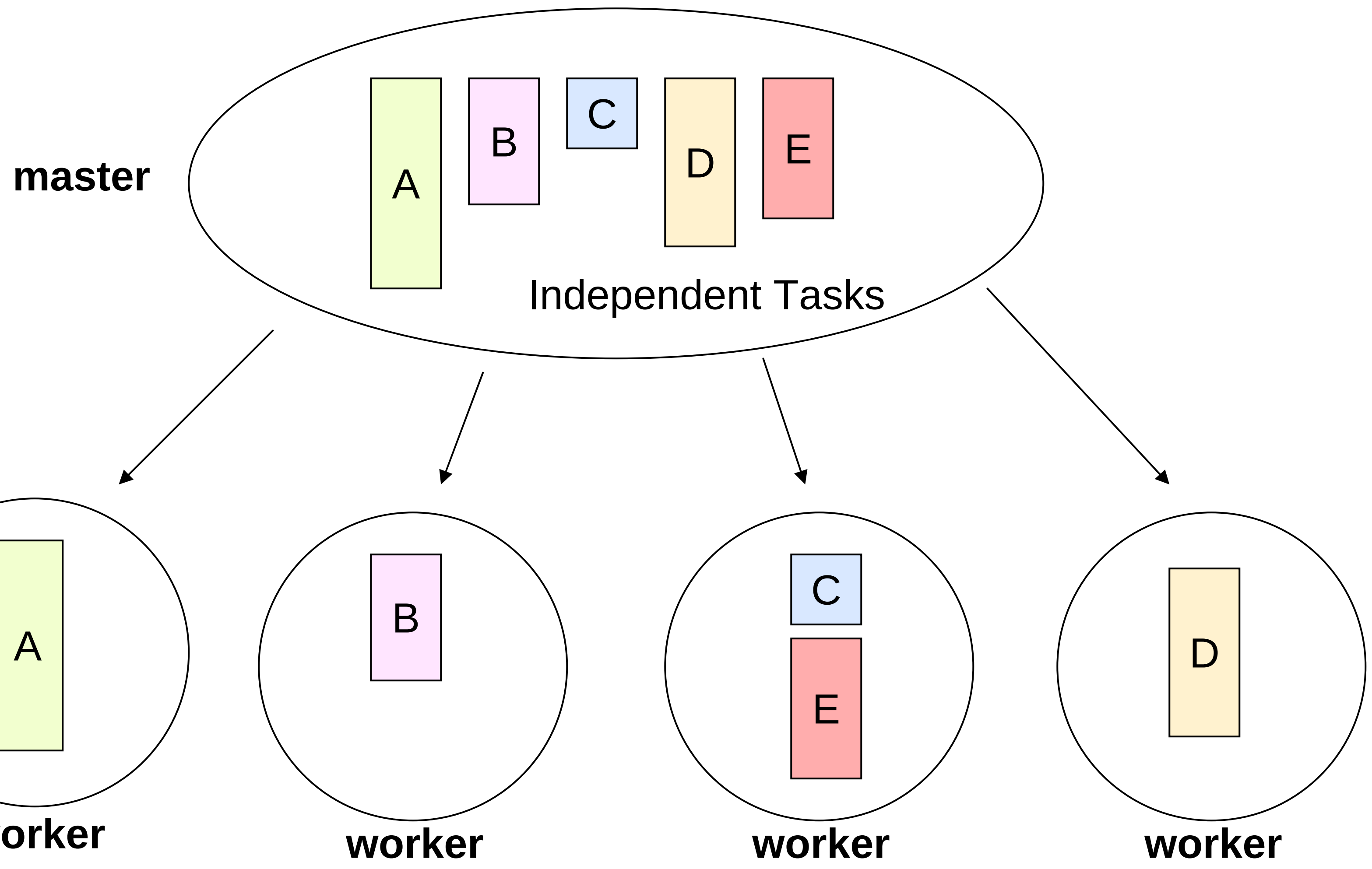
Loop Parallelism Pattern

- Many programs are expressed using iterative constructs
  - Programming models like OpenMP provide directives to automatically assign loop iteration to execution units
  - Especially good when code cannot be massively restructured

```
#pragma omp parallel for
for(i = 0; i < 12; i++)
      C[i] = A[i] + B[i];
```



Master/Worker Pattern



Master/Worker Pattern

- Particularly relevant for problems using task parallelism pattern where task have no dependencies
  - Embarrassingly parallel problems

- Main challenge in determining when the entire problem is complete

Fork/Join Pattern

- Tasks are created dynamically
  - Tasks can create more tasks

- Manages tasks according to their relationship

- Parent task creates new tasks (fork) then waits until they complete (join) before continuing on with the computation

SPMD Pattern

- Single Program Multiple Data: create a single source-code image that runs on each processor
  - Initialize
  - Obtain a unique identifier
  - Run the same program each processor
    - Identifier and input data differentiate behavior
  - Distribute data
  - Finalize

SPMD Challenges

- Split data correctly
- Correctly combine the results
- Achieve an even distribution of the work
- For programs that need dynamic load balancing, an alternative pattern is more suitable

Map/Reduce Pattern

- Two phases in the program
- Map phase applies a single function to all data
  - Each result is a tuple of value and tag
- Reduce phase combines the results
  - The values of elements with the same tag are combined to a single value per tag -- ***reduction***
  - Semantics of combining function are associative
  - Can be done in parallel
  - Can be pipelined with map
- Google uses this for *all* their parallel programs

Communication and Synchronization Patterns

- Communication
  - Point-to-point
  - Broadcast
  - Reduction
  - Multicast
- Synchronization
  - Locks (mutual exclusion)
  - Monitors (events)
  - Barriers (wait for all)
    - Split-phase barriers (separate signal and wait)
      - Sometimes called “fuzzy barriers”
    - Named barriers allow waiting on subset

Algorithm Structure and Organization (from the Book)

	Task parallelism	Divide and conquer	Geometric decomposition	Recursive data	Pipeline	Event-based coordination
SPMD	****	***	****	**	***	**
Loop Parallelism	****	**	***			
Master/ Worker	****	**	*	*	****	*
Fork/ Join	**	****	**		****	****

- Patterns can be hierarchically composed so that a program uses more than one pattern

Algorithm Structure and Organization (my view)

	Task parallelism	Divide and conquer	Geometric decomposition	Recursive data	Pipeline	Event-based coordination
SPMD						
Loop Parallelism						
Master/ Worker						
Fork/ Join						

- Patterns can be hierarchically composed so that a program uses more than one pattern

Algorithm Structure and Organization (my view)

	Task parallelism	Divide and conquer	Geometric decomposition	Recursive data	Pipeline	Event-based coordination
SPMD	****	**	****	**	****	*
Loop Parallelism	**** when no dependencies	*	****	*	**** SWP to hide comm.	
Master/ Worker	****	***	***	***	**	****
Fork/ Join	****	****	**	****		*

- Patterns can be hierarchically composed so that a program uses more than one pattern

Patterns for Parallelizing Programs

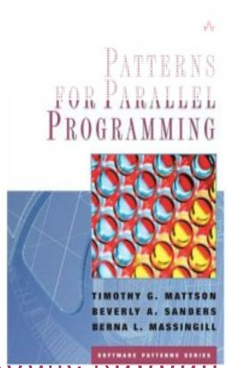
4 Design Spaces

Algorithm Expression

- Finding Concurrency
  - Expose concurrent tasks
- Algorithm Structure
  - Map tasks to processes to exploit parallel architecture

Software Construction

- Supporting Structures
  - Code and data structuring patterns
- Implementation Mechanisms
  - Low level mechanisms used to write parallel programs



Patterns for Parallel Programming. Mattson, Sanders, and Massingill (2005).

ILP, DLP, and TLP in SW and HW

- ILP
  - OOO
  - Dataflow
  - VLIW
- DLP
  - SIMD
  - Vector
- TLP
  - Essentially multiple cores with multiple sequencers

- ILP
  - Within straight-line code
- DLP
  - Parallel loops
  - Tasks operating on disjoint data
    - No dependencies within parallelism phase
- TLP
  - All of DLP +
  - Producer-consumer chains

ILP, DLP, and TLP and Supporting Patterns

	Task parallelism	Divide and conquer	Geometric decomposition	Recursive data	Pipeline	Event-based coordination
ILP						
DLP						
TLP						

ILP, DLP, and TLP and Supporting Patterns

	Task parallelism	Divide and conquer	Geometric decomposition	Recursive data	Pipeline	Event-based coordination
ILP	inline / unroll	inline	unroll	inline	inline / unroll	inline
DLP	natural or local-conditions	after enough divisions	natural	after enough branches	difficult	local-conditions
TLP	natural	natural	natural	natural	natural	natural

ILP, DLP, and TLP and Implementation Patterns

	SPMD	Loop Parallelism	Mater/Worker	Fork/Join
ILP				
DLP				
TLP				

ILP, DLP, and TLP and Implementation Patterns

	SPMD	Loop Parallelism	Master/ Worker	Fork/Join
ILP	pipeline	unroll	inline	inline
DLP	natural or local-conditional	natural	local-conditional	after enough divisions + local-conditional
TLP	natural	natural	natural	natural