



## Parallel Programming

Josep M. Codina

Intel Barcelona Research Center

Aula Empressa, Facultat d'Informàtica de Barcelona, February 2010

© Intel Corporation, 2010

## Parallel Programming?

I still want to divide the effort of eating pizza to eat all slices as fast as possible!!



But, I wonder...

How we split this hard task ☺

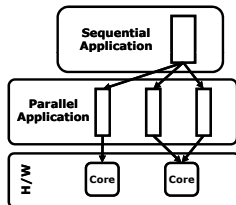
2

Designing Tomorrow's Microprocessors



## Why Parallel Programming?

- Applications are naturally parallel
- Multi-core are out there
- We need to think in parallel!!!!



48- IA cores in the Intel "Single-chip Cloud Computer"

3

Designing Tomorrow's Microprocessors



## Agenda

- Motivation
- Designing Parallel Applications
- Parallel Programming Models
- Automatic Parallelization
- Parallel Programming for Games
- Concluding Remarks

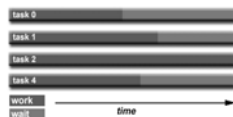
4

Designing Tomorrow's Microprocessors



## Design of Parallel Applications

- Understand the Problem
  - Identify hotspots
    - Profile application
  - Identify bottlenecks to the parallelization
    - Communication
    - Synchronization
  - Consider alternative versions of the application
- Split Application in Parts
  - Maximum parallelism
  - Minimum imbalance
  - Minimum waiting time



5

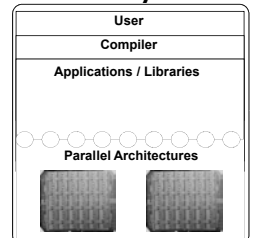
Designing Tomorrow's Microprocessors



## Who Creates Parallel Applications?

- User/ Programmer
  - Parallel languages and libraries
- Compilers
  - Automatic Parallelization
- Parallel Architectures
  - Multi-core in a single chip
  - Many multi-core chips

### SW Ecosystem



6

Designing Tomorrow's Microprocessors



## What Do We Need?

### Parallel Programming Languages

### Provide Tools Today

### Architecture, Thread, Debug & Tune



[www.intel.com/software/products](http://www.intel.com/software/products)

### Research Tomorrow's Techniques



Transactional Memory

Speculative Multi-threading

Data Parallel Languages

University Outreach

Intel @ Press

Intel @ Software College

### Educate Tomorrow's Experts



Intel @ Press

Intel @ Software College

Helped 45 universities add parallel programming courses  
7500 students took them  
2007 Goal: 400+ universities



7

Designing Tomorrow's Microprocessors



## When to Parallelize an Application?

### • Allways!!!!

- Stop thinking in sequential applications
- Multicore era is here and we have to leverage the computing capabilities that we have out there

8

Designing Tomorrow's Microprocessors



## Agenda

- Motivation
- Designing Parallel Applications
- Parallel Programming Models
- Automatic Parallelization
- Parallel Programming for Games
- Concluding Remarks

9

Designing Tomorrow's Microprocessors



## Parallel Programming Models

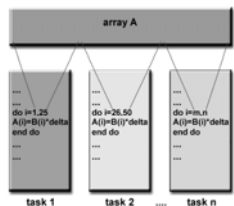
- Data Parallel
- Message Passing
- Shared Memory
- Distributed Shared Memory Model
- NOTE: These models are orthogonal to the actual hardware!!!

10

Designing Tomorrow's Microprocessors



## Data Parallel Model



- Set of cooperating tasks on the same data structure
- Tasks perform the same operation over on different data items

### Advantage

Easy to write and comprehend  
No synchronization

### Disadvantage

No independent code

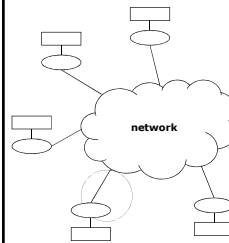
Example: HPF(High Performance Fortran)

11

Designing Tomorrow's Microprocessors



## Message Passing Model



Address space  
Process

- A set of cooperating sequential processes
- Each with own local address space
- Processes interact with explicit transaction (send, receive,...)

### Advantage

Programmer controls data and work distribution

### Disadvantage

Communication overhead for small transactions  
Hard to program!

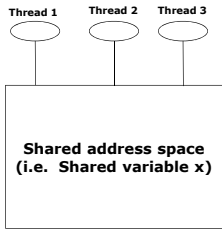
Example: MPI

12

Designing Tomorrow's Microprocessors



## Shared Memory Model



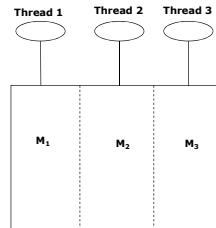
- Different simultaneous execution threads (processes)
  - Read / Write to one shared memory space and invalidate if necessary
- Advantage**
- Read remote memory via an expression
  - Write remote memory through assignment
- Disadvantage**
- Manipulating shared data leads to synchronization requirements
  - Does not allow locality exploitation
- Example : OpenMP

13

Designing Tomorrow's Microprocessors



## Distributed Shared Memory Model



**Partitioned shared address space**  
(with each partition having affinity to corresponding thread)

- Similar to the shared memory paradigm
  - Memory  $M_i$  has *affinity* to Thread  $i$
  - At the same time each thread has global view of memory
- Advantage**
- Helps exploiting locality of references
  - Simple statements as SM
- Disadvantage**
- Synchronization still necessary
- Example: UPC, Titanium, Co-Array, Global Arrays

14

Designing Tomorrow's Microprocessors



## Agenda

- Motivation
- Designing Parallel Applications
- Parallel Programming Languages
- Automatic Parallelization
- Parallel Programming for Games
- Concluding Remarks

15

Designing Tomorrow's Microprocessors



## Traditional Auto Parallelization

- Automatic decomposition of applications into threads
- No need for inserting directives or pragmas
- Compiler identifies suitable parts of the application for parallelization
  - Typically simple loops
  - Considering simple memory disambiguation schemes
- This approach is typically limited to simple applications
  - Dependences limit its applicability to large scale applications

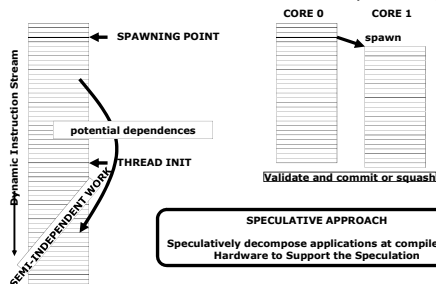
16

Designing Tomorrow's Microprocessors



## Speculative Multithreading

- Automatic decomposition of applications into speculative threads
  - No need to be conservative on the memory disambiguation



17

Designing Tomorrow's Microprocessors



## Agenda

- Motivation
- Designing Parallel Applications
- Parallel Programming Languages
- Automatic Parallelization
- Parallel Programming for Games
- Concluding Remarks

18

Designing Tomorrow's Microprocessors



# Usual Game Structure

```
graph TD; A[Render OnFrameMove] --> B[Physics]; A --> C[AI]; A --> D[Particles]; B --> C; C --> D; D --> A;
```


The diagram illustrates the typical game structure loop. At the top, the text **Render OnFrameMove** is centered, with a curved arrow on the left and a curved arrow on the right, indicating a continuous loop. Below this, three arrows point downwards to the text **→Physics→AI→Particles**, which is also centered. This represents the sequence of operations performed within each frame: rendering, physics simulation, AI processing, and particle updates.

Comparative Analysis of Game Parallelization, *Dimitry Eremin*, GDC'08

1

Designing Tomorrow's Microprocessors

# Usual Game Structure



**Render→ Physics→ AI→ Particles**

<http://softwarecommunity.intel.com/articles/eng/1363.htm>

Comparative Analysis of Game Parallelization, Dmitry Eremin, GDC'08

20

Designing Tomorrow's Microprocessors

Intel

Comparative Analysis of Game Parallelization, *Dimitry Eremin, GDC'08*

2

Designing Tomorrow's Microprocessors

# Game Thread Profiling

Sequential execution  
25% of system utilization  
Benchmark: 20.95sec  
*Measured on 4 core test machine*

**OnFrameMove**

**Render** **Physics** **AI** **Particles**

Time (seconds)

CLS CL1 CL2

Main, Rollout.exe

Comparative Analysis of Game Parallelization, Dmitry Eremin, GDC'08

21

Designing Tomorrow's Microprocessors

Sequential execution  
25% of system utilization  
Benchmark: 20.95sec  
*Measured on 4 core test machine*

Comparative Analysis of Game Parallelization, *Dimitry Eremin, GDC'08*

2

Designing Tomorrow's Microprocessors

# Functional Decompostion

**Idle:** no active threads

**Serial:** a single thread

**Under-subscribed:** # threads > 1 && # threads < # cores

**Fully-subscribed:** # threads == # cores

**Oversubscribed:** # threads > # cores

Concurrency level	Time (seconds)
1	0.0035
0.5	0.0025
0.3	0.0015
0.2	0.0020
0.1	0.0030

Concurrency level is the number of threads that are active (not waiting, sleeping, blocked, etc.) at a given time

- **Load imbalance**
- **Benchmark: 10.15sec**

Reader

Physics

AI

Particles

22

Designing Tomorrow's Microprocessors

- Idle: no active threads
- Serial: a single thread
- Under-subscribed: # threads > 1 && # threads < # cores
- Fully-subscribed: # threads == # cores
- Over-subscribed: # threads > # cores

- **Load imbalance**
- Benchmark: 10.15sec

Concurrency level is the number of threads that are *active* (not waiting, sleeping, blocked, etc.) at a given time

2

## Designing Tomorrow's Microprocessors

# Functional and Data Decomposition

- If Split AI in 2 threads
- **Load imbalance**
- Benchmark: 9.44sec

*Measured on 4 core test machine*

Steps (iterations)	Time (seconds)
C1	0.015
C1,2	0.007
C2,3	0.007
C3,4	0.014
C4,5	0.005
C5,6	0.003
C6,7	0.002

23

Designing Tomorrow's Microprocessors

- If Split AI in 2 threads
- **Load imbalance**
- Benchmark: 9.44sec

Measured on 4 core test machine

2

Designing Tomorrow's Microprocessors

# Intel® Threading Building Blocks

- Open Source Library for parallelism in C++
  - Fine-grain decomposition
  - Task scheduler

The diagram illustrates the execution flow of a task scheduler using Intel Threading Building Blocks. It shows a sequence of tasks (update several AI units, update several blocks, update several particles) being scheduled across multiple processors. A callout points to a bar chart showing the benchmark result: Good utilization of 4 cores, Benchmark: 8.66 sec.

Task	Time (seconds)
CL1	0.00
CL2	0.00
CL3	0.00
CL4	0.00
CL5	0.00
Mean, StdDev, min	0.00

Good utilization of 4 cores  
Benchmark: 8.66 sec

- Open Source Library for parallelism in C++
  - Fine-grain decomposition
  - Task scheduler

**Good utilization of 4 cores**  
**Benchmark: 8.66 sec**

2

Designing Tomorrow's Microprocessors

## Agenda

- Motivation
- Designing Parallel Applications
- Parallel Programming Languages
- Automatic Parallelization
- Parallel Programming for Games
- Concluding Remarks

25

Designing Tomorrow's Microprocessors



## Summary - Conclusions

- Parallel Applications are required to leverage Parallel Architectures
- Today we discussed about:
  - Why we need parallel applications
  - Considerations When Designing parallel applications
  - Parallel Programming Models
  - Auto Parallelization and Speculative Multithreading
  - The importance of threading in gaming
- We need to think in parallel to create parallel applications!!!

26

Designing Tomorrow's Microprocessors



## Parallel Programming

Josep M. Codina

Intel Barcelona Research Center

Aula Empresa, Facultat d'Informàtica de Barcelona, February 2010

© Intel Corporation, 2010

