ISCTE IUL
Instituto Universitário de Lisboa

PCD
Programação Concorrente e Distribuída

Parallel and Distributed Programming

2014-2015

Today

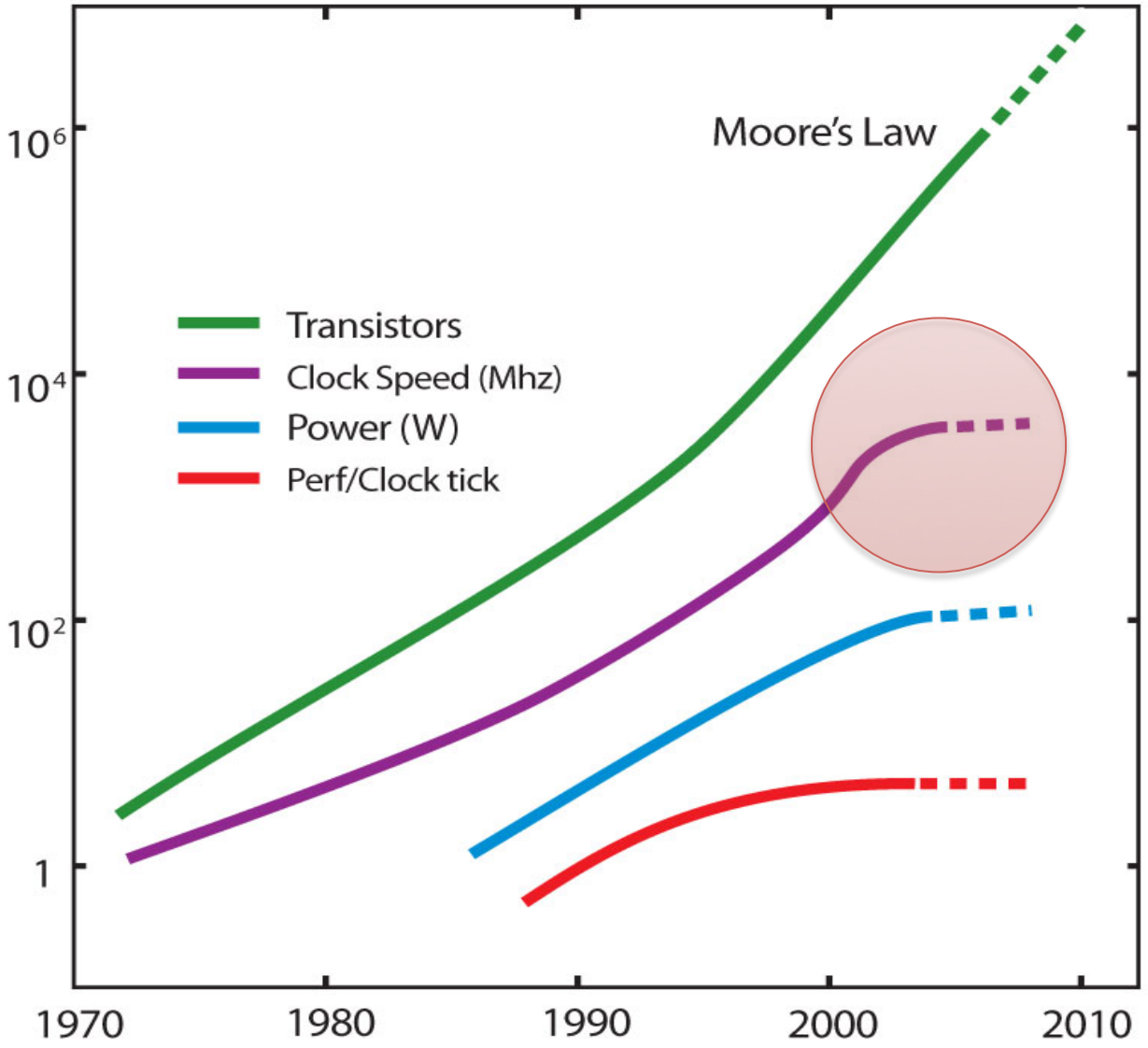
- Parallel programming – what and why?
- Distributed programming – what and why?
- Course overview
- Evaluation
- Practical information
- Revisions / Exercises

What is parallel programming?

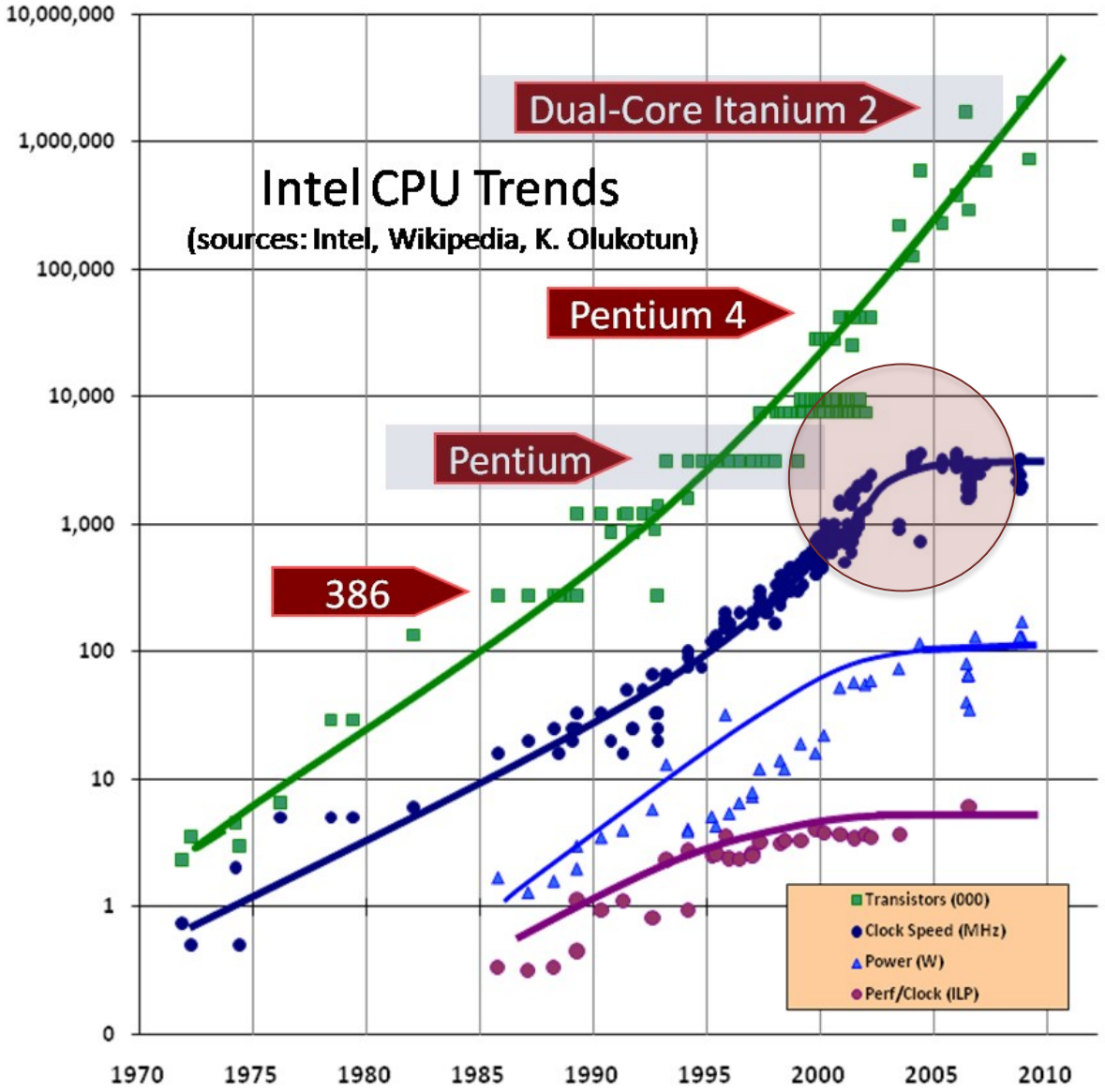
Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel")

http:// en.wikipedia.org/wiki/Parallel_programm

Why Parallel Programming?



Why Parallel Programming?



http://www.extremetech.com/computing/116561-the-death-of-cpu-scaling-from-one-core-to-many-and-why-were-still-stuck

General-purpose CPUs have up to 16 cores

GPUs have up to 3072 cores (nVIDIA Tesla K10)

Dual-core and quad-core mobile phones and tables.

Parallel and concurrent programming

- More processing power out of CPUs;
- Enables programs to perform several tasks at the same time:
  - spell check as the user writes in Word,
  - send a document to the printer while the user continues to work,
  - ...

What is distributed programming?

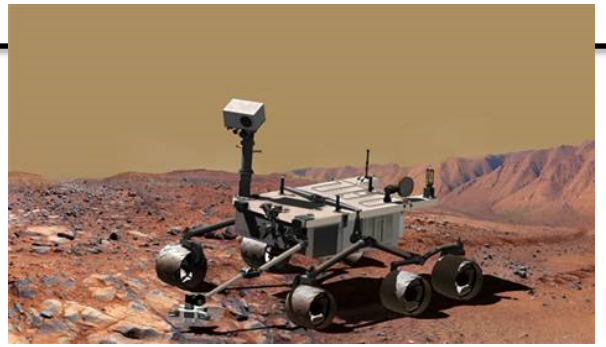
A distributed system consists of multiple autonomous computers that communicate through a computer network.

A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs.

http://en.wikipedia.org/wiki/Distributed_computing

Why Distributed Programming

- Some information is distributed
- Performance, redundancy and fault tolerance
- Cloud computing



Google

facebook

amazon web services™

Windows Azure

Course Overview

- POO Revision
- Swing (Event Driven Programming)
- Threads
- Synchronization
- Coordination
- Project
- Network programming
- Deadlocks, starvation and livelocks
- Serialisation
- Applets
- High-level concurrency objects
- Project support
- Project evaluation

Bibliography

Main:

- Principles of Concurrent and Distributed Programming, M. Ben-Ari, 2006 Addison Wesley

- Foundations of Multithreaded, Parallel, and Distributed Programming, Gregory R. Andrews, 1999 Addison Wesley

- JAVA Threads, Third Edition, Scott Oaks & Henry Wong, 2004 O'Reilly.

Complementary:

- Tutorial for J6SE:
  http://download.oracle.com/javase/tutorial/index.html

- Tutorial for J5EE:
  http://download.oracle.com/javaee/5/tutorial/doc/

Evaluation

- Periodic evaluation
  - 2 Mini Tests (2 points)
  - Frequência (0-10 points, minimum 4 points)
  - Project (0-8 points, minimum 3 points)

    Projects are done in pairs (2 students). Projects are evaluated in an individual oral exam at the end of the semester
- Exam
  - Project (must fulfill minimum requirements to access the final test)
  - Final test (0-20 points, minimum 10 points)

Practical Information

Team:

Sancho Oliveira (coordinador)

- Sancho.Oliveira@iscte.pt
- D605

Luís Nunes

- Luis.Nunes@iscte.pt
- D617

Luís Mota

- Luis.Mota@iscte.pt
- D619

Joaquim Reis

- Joaquim.Reis@iscte.pt
- D609

WebSite:

e-learning.iscte.pt

Pratical Classes:

All students have to attend the class they have enrolled in Fenix.

Time Table

Horas/Dias	segunda-feira	terça-feira	quarta-feira		quinta-feira		sexta-feira
9:00-9:30							
9:30-10:00	**T** - L5096-1T01		**TP** - L5096-1TP04				
10:00-10:30	**Sala: Auditório B2.04**		**Datas: (...)**				
10:30-11:00	**Turmas: (...)**		**Turmas: (...)**				
11:00-11:30	**T** - L5096-1T02		**TP** - L5096-1TP04				
11:30-12:00	**Sala: Auditório 4**		**Datas: (...)**				
12:00-12:30	**Turmas: (...)**		**Turmas: (...)**				
12:30-13:00							
13:00-13:30	**Sancho Dúvidas**	**TP** - L5096-1TP04	**TP** - L5096-1TP06	**TP** - L5096-1TP05	**TP** - L5096-1TP07	**TP** - L5096-1TP08	
13:30-14:00		**Sala: D1.09**	**Sala: D1.02**	**Sala: D1.06**	**Sala: D1.12**	**Sala: D1.06**	
14:00-14:30		**Turmas: (...)**	**Turmas: (...)**	**Turmas: (...)**	**Turmas: (...)**	**Turmas: (...)**	
14:30-15:00		**TP** - L5096-1TP04	**TP** - L5096-1TP06	**TP** - L5096-1TP05	**TP** - L5096-1TP07	**TP** - L5096-1TP08	
15:00-15:30		**Sala: D1.09**	**Sala: D1.09**	**Sala: D1.06**	**Sala: D1.12**	**Sala: D1.06**	
15:30-16:00		**Turmas: (...)**	**Turmas: (...)**	**Turmas: (...)**	**Turmas: (...)**	**Turmas: (...)**	
16:00-16:30							**Dúvidas Joaquim**
16:30-17:00							
19:00-19:30							
19:30-20:00	**T** - L5096-2T01	**TP** - L5096-2TP02					**TP** - L5096-2TP03
20:00-20:30	**Sala C507**	**Sala 0S01**					**Sala: D1.09**
20:30-21:00	**Turmas: (...)**	**Turma: I-PLB1**					**Turmas: (...)**
21:00-21:30		**TP** - L5096-2TP02					**TP** - L5096-2TP03
21:30-22:00		**Sala 0S01**					**Sala: D1.09**
22:00-22:30		**Turma: I-PLB1**					**Turmas: (...)**
22:30-23:00							

Joaquim
Luís Mota
Luís Nunes
Sancho

Expected dedication

- The course corresponds to 6 ECTS points
- This is equal to approximately 167 hours of work:
  - 50.5 hours in class
  - 57.5 hours of study
  - 55.0 hours of group work
  - ~4.0 hour of evaluation

- ***Remember:*** You learn programming by programming.

Expected Skills

- PCD builds on what you have learned in IP and POO.

- While it is possible to do PCD without having passed POO, it is extremely difficult!

More Practical Information

See the course webpage on

http://e-learning.iscte.pt

Revisões de POO

OBJECT ORIENTED PROGRAMMING REVISION

Objects and Classes

```
import java.util.*;
public class Emprestimos {
   private LinkedList<String> nomes;
   public Emprestimos(LinkedList<String> nomes) {
      this.nomes=nomes;
   }
}
```

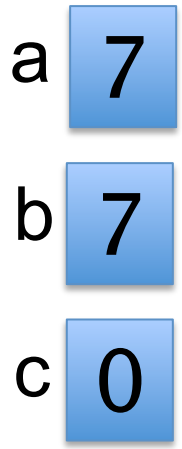
Objects and Classes

```
class Livro {
    private String titulo;
    private Emprestimos emprestimos;
    public Livro(final String titulo,
                 final Emprestimos emprestimos) {
        this.titulo=titulo;
        this.emprestimos=emprestimos;
    }
}
class Biblioteca {
    private LinkedList<Livro> livros;
    public Biblioteca(final LinkedList<Livro> livros) {
        this.livros=livros;
    }
}
```

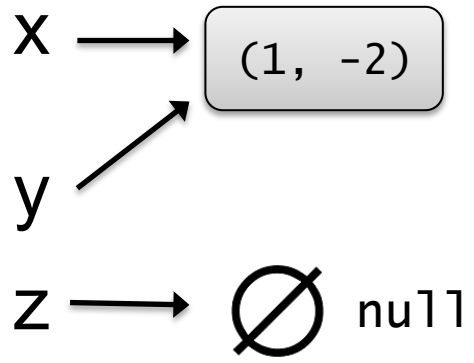
Primitive Types and Reference Types

Primitive Types (int, boolean, etc)	Reference Types (Classes e vectores)

```
int a = 7;

int b = a;

int c;
```




```
Point x = new Point(1, -2);

Point y = x;

Point z;
```



Fonte: Programação Orientada para Objectos ISCTE-IUL

Static Variables, Constants and Methods

- Constants
- Static Variables or Class Variables
- Static Methods or Class Methods
- Dynamic Variables or Instance Variables
- Dynamic Methods or Instance Methods

Thinking in Objects

- Problem Analysis
  - What objects should be defined?
  - What are their responsibilities?
  - How do they collaborate?
  - How to classify them?
- Solution Implementation
  - Classes?
  - Objects?
  - Responsibilities?
  - How they can collaborate?

Inheritance and Polymorphism

```
private LinkedList<Sala> salas = new LinkedList<Sala>();
...
salas.add(new Anfiteatro("A1", 130));
salas.add(new SalaDeComputadores("D101", 25));
salas.add(new Sala("1E2", 30));
salas.add(new Sala("C605", 50));
```

Inheritance and Polymorphism

```
salasComCapacidade =
    salasComCapacidadeConjuntaDeExameParaMaisDe(salas, 42);

System.out.println("Salas com capacidade conjunta de exame"
    + "para, pelo menos, 42 alunos:");

for (Sala sala: salasComCapacidade) {
    System.out.println(sala + " capacidade para exame " +
    sala.númeroDeLugaresParaExame());
}
```

Inheritance and Polymorphism

```
public class Component {
     private int x;
     private int y;
     private int height = 0;
     private int width = 0;

     public Component(int x, int y,
               int height, int width) {
          //...
     }

     public void move(int dx, int dy) {
          x+=dx;
          y+=dy;
     }

     @Override
     public String toString() {
          return "(" + x + "," + y + ")";
     }
     //...
}
```

```
public class Button extends Component {

     private String text;

     public Button(int x, int y, int
height,
                    int width, String
text) {
          super(x, y, height, width);
          this.text = text;
     }

     public void move(Point newPoint) {
          //Overloading
          //TODO: …
     }

     @Override
     public String toString() {
          return "Button " + text + " @ " +
                 super.toString();
     }
     //…
}
```

Abstract Classes and Interfaces

```
public abstract class Component {
     // …
     public abstract void paint();
     // ...
}


public class Button extends Component {
     @Override
     public void paint() {
          // TODO paints the button
     }
}


public interface Listener {
     void actionPreformed();
}
```

```
public class OkListener
               implements Listener {
     @Override
     public void actionPreformed() {
          System.out.println(
               "OK Button pressed...");
     }
}


public class DoneListener
               implements Listener {
     @Override
     public void actionPreformed() {
          // TODO end application
     }

}
```

Abstract Classes and Interfaces

- What is an abstract class?
  - Class rooms
  - Offices
  - Meeting rooms
  - Library
  - Canteen
  - Etc…
- What is an interface?
  - Reservable room

Exception Handling

```
public class ExceptionHandling {
    public  void method1(){
        try {
            method2();
        } catch (MyException e) {
            // TODO: handle exception
        }
    }

    public  void method2() throws MyException{
        if(gotError()){
            throw new MyException();
        }
        // TODO: do what has to be done....
    }
```

```
    private boolean gotError() {
            // TODO check error
        }
}
```

```
public class MyException
    extends Exception {
    // TODO: ....
}
```

Exercise – a gentle start

Make a console application that picks a random number between 0..9.

The user has to guess the number.

Each time the user inputs a number, the program replies whether the user's guess is too high, too low or correct.

After the user as guess the number, the program should terminate.

Optional: Look at JOptionPane.showInputDialog(…)