

Corso di

Programmazione Concorrente

anno accademico 2016-2017

Valter Crescenzi

`crescenz@dia.uniroma3.it`

<http://crescenzi.inf.uniroma3.it>

Informazioni Generali (I)

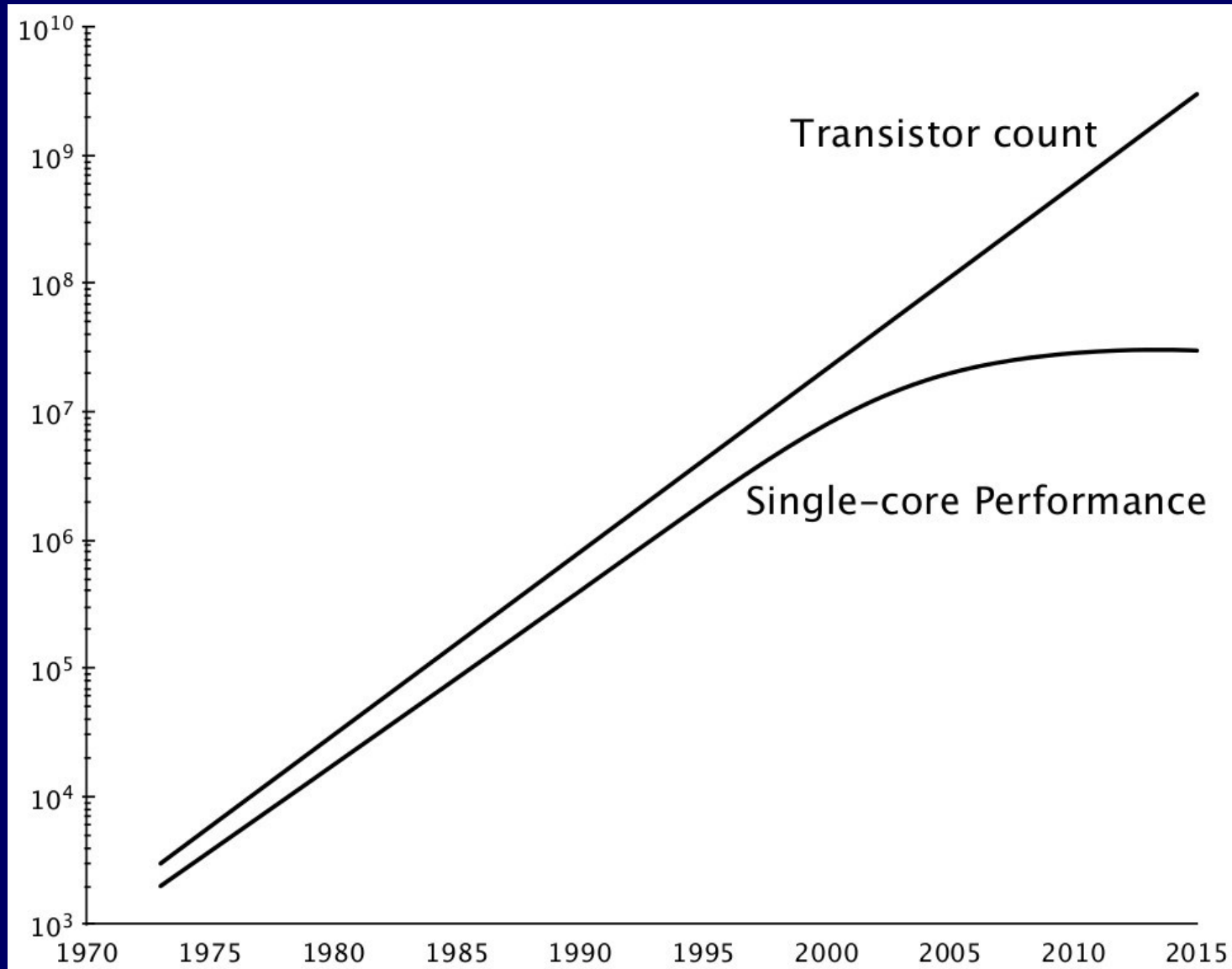
- Docente: Valter Crescenzi
- Email: *crescenz@dia.uniroma3.it*
- Web: *http://crescenzi.inf.uniroma3.it*
- Sito Web del Corso:
http://crescenzi.inf.uniroma3.it/didattica/aa2016-2017/PC/index.html
- Orario di Ricevimento:
 - *giovedì dalle 14:15 alle 15:00 sino al primo appello, oppure*
 - *previo appuntamento da fissare per email*

 - *durante il corso: anche a fine lezione*

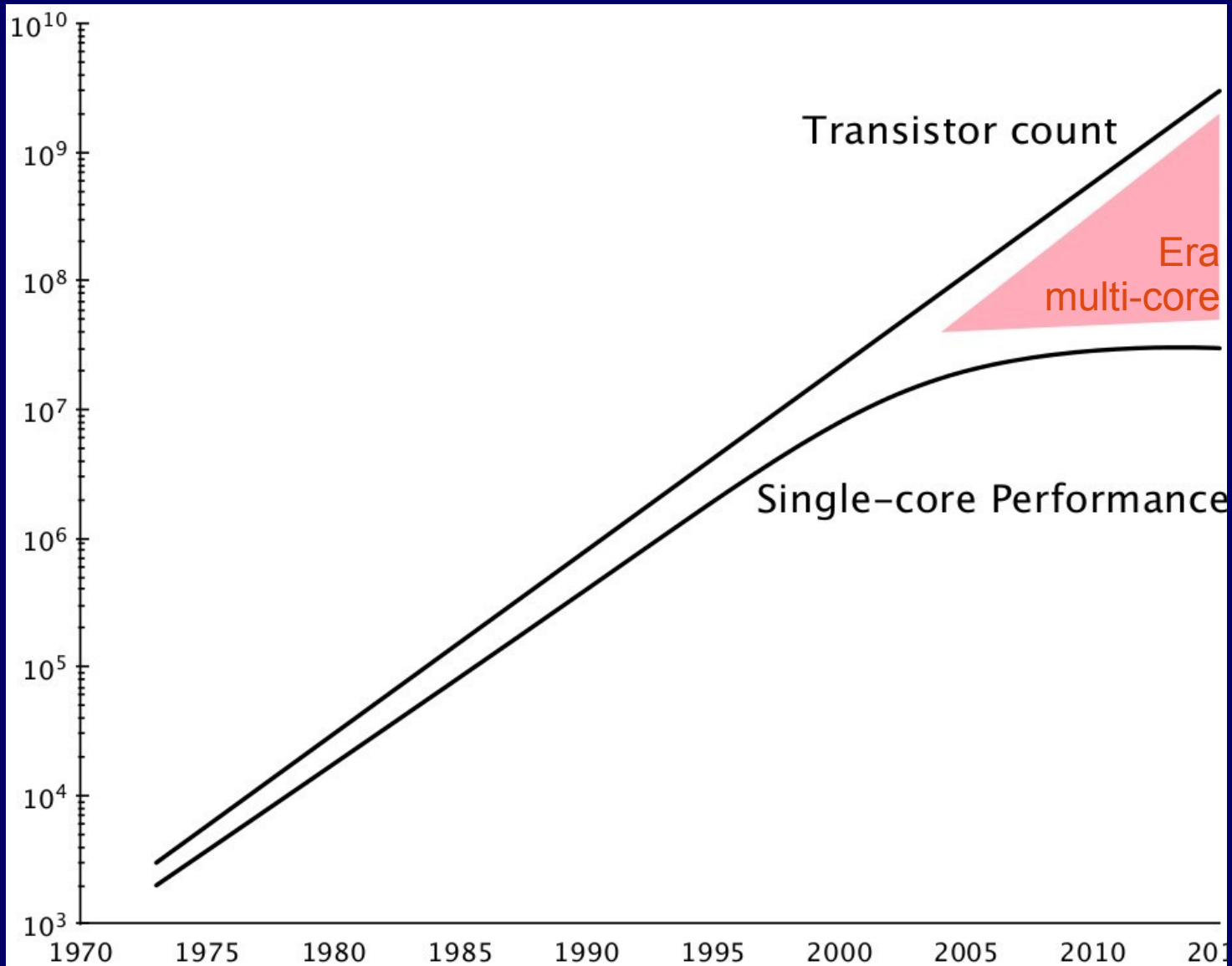
Informazioni Generali (II)

- Calendario:
 - Dal 3 ottobre 2016 al 27 gennaio 2017
 - Due lezioni a settimana: martedì e giovedì
 - Talvolta dedicate a:
 - prove intermedie
 - esercitazioni
- Orario:
 - Martedì dalle 10:15
 - Giovedì dalle 12:15?
- Aula:
 - N13

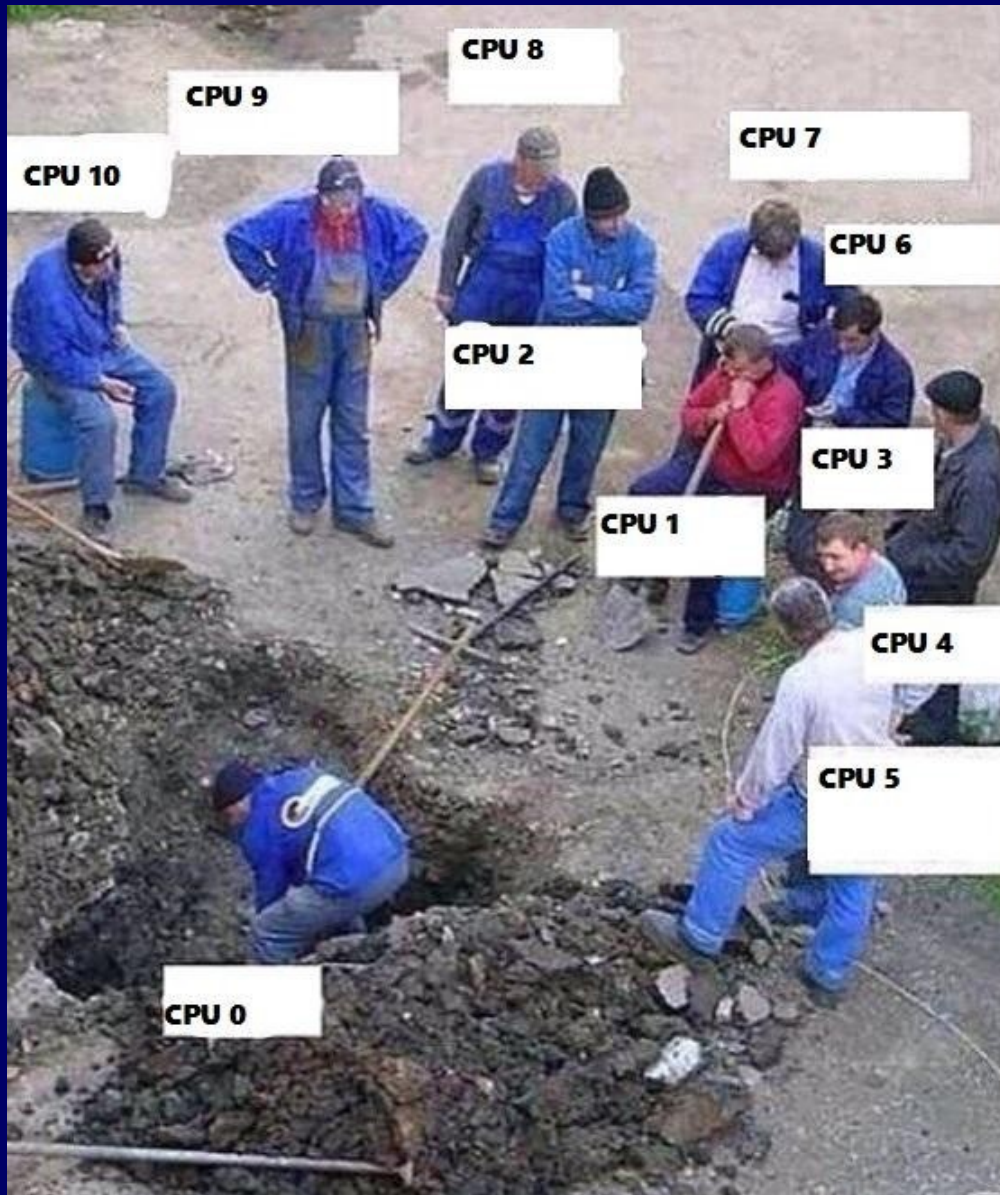
Fine Legge di Moore per Singolo Core



L'Era Multi-Core



“Uno lavora e 10 guardano!”



- Eseguendo il vostro “vecchio” codice seriale sulle nuove architetture
- ✓ *Scommessa*: il problema è così rilevante che in futuro influenzerà la didattica dei primi anni dei CdS in Ingegneria Informatica

Obiettivi Formativi

- Comprendere le problematiche della programmazione di attività concorrenti
- Utilizzare linguaggi di programmazione diffusi per la risoluzione di problemi concorrenti
- Individuare ed approfondire alcune promettenti linee di evoluzione delle tecnologie e delle metodologie che già stanno delineando i linguaggi di programmazione e le tendenze del futuro

Cosa Riusciremo a Fare? (I)

- Comprendere i principi e le tecniche per la programmazione di più esecutori che accedono concorrentemente risorse condivise
- Scrivere programmi concorrenti per sfruttare pienamente le potenzialità delle architetture multi-processore
- Concepire soluzioni pensate sin dal principio per scalare
 - su più processori (scale-up)
 - su più macchine (scale-out)

Cosa Riusciremo a Fare? (II)

- Applicare i concetti appresi nei contesti di più chiara ed immediata utilità applicativa
 - Applicazioni locali multi-thread e multi-processo
 - GUI
 - Calcolo parallelo
 - Applicazioni distribuite
 - multi-utente con accessi concorrenti a risorse condivise
 - con forti esigenze di scalabilità
 - Utilizzare librerie ad alto livello per la scrittura di applicazioni concorrenti
 - `java.util.concurrent`
 - Akka
 - Java 8 Stream & Future

riuscendo ad apprezzare le motivazioni, ambiti applicativi, e scelte di progetto delle stesse

Cosa Riusciremo a Fare? (III)

- Apprezzare lo stato dell'arte nello studio delle problematiche inerenti la programmazione di esecutori concorrenti, tramite queste attività:
 - Studio di concetti, metodologie e tecnologie fondamentali
 - Studio dello stato dell'arte
 - non-blocking algorithms
 - framework per la decomposizione parallela
 - Vere e proprie scommesse:
 - Approcci alternativi (ad es. modelli basati su attori – Akka)
 - si stanno *già* diffondendo rapidissimamente
 - è difficile predirre il loro impatto nel lungo termine

Programma

- Fondamenti di Programmazione Concorrente
- Processi e Thread in C sotto UNIX
 - PC in C sotto UNIX
- Java Thread
 - Fondamenti
 - Librerie avanzate
- Testing di codice concorrente
- Decomposizione parallela (Framework Fork & Join)
 - Tecniche di work-stealing
- Computazione Asincrone (Future e Stream da Java 8)
- Modelli ad Attori (Akka in Java / Scala)
- Algoritmi *non-blocking*

Fondamenti di Programmazione Concorrente

- F.d.E. sequenziali e paralleli
- Interferenza tra F.d.E.
- Condizioni di Bernstein
- Il Problema della Mutua Esclusione
- Semafori
- Stallo/Starvation/Fairness
- Assunzione di progresso finito
- Regioni Critiche
- Monitor
- Problemi Classici: Produttore / Consumatore, I Cinque Filosofi Mangiatori, il Barbiere Dormiente
- Elementi metodologici: tecniche di prog. concorrente

Processi & Thread

- Processi & Thread in C sotto UNIX
 - IPC in C sotto UNIX
- I Thread in Java
 - strumenti offerti dal linguaggio
 - aspetti metodologici
 - `java.util.concurrent`
 - lightweight execution framework
 - decomposizione parallela

Java Thread

- I Java Monitor
- Strumenti offerti dal linguaggio
- Aspetti metodologici
 - immutabilità, sincronizzazione, confinamento...
 - Work-stealing
- Aspetti tecnologici
 - il package `java.util.concurrent` (JSR 166)
 - come libreria che fornisce meccanismi classici per la prog. concorrente
 - Come Lightweight-Execution-Framework
 - supporto ad algoritmi non-blocking
 - studio dell'architettura e di alcune scelte progettuali caratterizzanti
 - Decomposizione Parallela ed il framework Fork/Join (JSR166y)
 - Java 8 Collections: *Stream*

Argomenti Avanzati

- Testing di applicazioni concorrenti
- Decomposizione Parallela e tecniche di Work -Stealing
- Modelli ad Attori
 - Akka
- Algoritmi *non-blocking*

Metodologia di Studio

- Si consiglia di seguire tutte le lezioni e di usufruire dei percorsi basati su homework (>>)
 - Se impossibilitati, almeno le prime 6 lezioni
- “Nucleo” di concetti fondamentali piccolo
 - riuscire ad applicarli in diversi contesti applicativi a vari livelli di astrazione non è affatto banale
- Tesina ed homework saranno l’occasione per metterli in pratica

Homework

- Il corso prevede lo svolgimento di 4 homework
 - pubblicazione e scadenza prestabilite
 - possono essere utilizzati solo nel primo appello
 - sono destinati agli studenti che frequentano il corso
 - ma possono anche essere svolti da chi non segue le lezioni
- Durante l'orale si discutono gli hw, se svolti
 - **HWT**: Serie di quiz a risposta chiusa sulla teoria
 - **HWC1**: In linguaggio C; testing di una soluzione ad un problema classico
 - **HWC2**: In linguaggio C; risoluzione e testing di un problema di programmazione concorrente
 - **HWJ**: In linguaggio Java; risoluzione e testing di un problema di programmazione concorrente

HWC1

Viene corretto e (auto)valutato, ma non considerato ai fini della valutazione finale. Necessario per HW2.

Linguaggi di Programmazione

- C , è il linguaggio utilizzato per
 - IPC su singola macchina e su più macchine
 - esercitarsi sugli elementi di base della programmazione concorrente
- Java , è il linguaggio usato per
 - studiare la programmazione concorrente in un contesto orientato agli oggetti
 - indagare alcuni aspetti metodologici per la progettazione di applicazioni concorrenti
 - approfondire l'utilizzo di librerie di alto livello per la programmazione concorrente
 - approfondire alcune evoluzioni interessanti nella letteratura del settore
- Scala , è il linguaggio consigliato
 - per sviluppare tesine in Akka
 - paradigma misto PF-PO
 - per i più temerari e curiosi !

Modalità d'Esame (1)

- Per il *solo* primo appello:
 - *[Tesina] + Orale + HWT + [Altri Homework]*
 - Per *tutti* gli altri appelli (di recupero):
 - *Tesina + Scritto + Orale*
 - *Tutti* gli homework, tranne HWT, sono facoltativi*
 - HWT
 - [HWC1]
 - [HWC2]
 - [HWJ]
- *chi non svolge HWT deve fare anche lo scritto
- E' quindi possibile crearsi dei percorsi personalizzati:
 - Percorso *Esame Completo*: *Tesina + Scritto + Orale*
 - Percorso *HW*: *HWT + HWC1 + HWC2 + HWJ + [Tesina] + Orale*
 - Percorso *C*: *HWT + HWC1 + HWC2 + [Tesina C] + Orale*
 - Percorso *J*: *HWT + HWJ + [Tesina Akka] + Orale*
 - Percorso *T*: *HWT + Orale*

Modalità d'Esame (2)

- Valutazione basata su 4 elementi
 - Esame orale
 - HWT
 - Esercizi/Homework di Programmazione (Opzionali)
 - Programmazione Concorrente in Java
 - Programmazione Concorrente in C
 - Tesina (Opzionale)
 - di approfondimento utilizzando il framework Akka

Esame Orale

- *Disclaimer:*
 - nessun pregiudizio per chi si presenta all'orale soltanto con HWT

- Orale relativamente strutturato:
 - 4 domande che vertono rispettivamente:
 1. su uno dei quiz sbagliati ad HWT
 2. su HWC2 (oppure su tutto il programma se non è stato consegnato)
 3. su HWJ (oppure su tutto il programma se non è stato consegnato)
 4. su tutto il programma
 - Discussione sulla tesina, se consegnata

Time-Slicing

- il corso è stato pensato per offrire la massima
 - *flessibilità* nella scelta degli hw/tesine
 - *rigidità* sui tempi d'esame.
 - HWT è l'inizio di data certa
 - La verbalizzazione di febbraio è la fine di data certa
 - prima metà per chi *non* svolge alcuna tesina
 - seconda metà per chi svolge la tesina
 - a marzo tutti dobbiamo fare altro...

- NO! a tesine
 - sconfinite
 - interminabili
 - su qualsiasi argomento

- Facciamo variare gli hw, le tesine (anche il voto!) ma *non* il tempo che dedichiamo a questi 6 CFU

Tesine

- In gruppi di max 2 persone
 - *facoltative per il solo primo Appello*
 - *obbligatorie* in tutti gli altri appelli
- Basate sull'utilizzo di Akka
 - un framework (Java/Scala) per la scrittura di codice concorrente scalabile
 - basata sul Modello ad Attori
 - specifiche concordate
- Applicazioni con forti esigenze di scalabilità

Considerazioni Finali: Passato vs Futuro & Presente

- Il corso guarda sia al *Passato* sia al *Futuro*
 - Al passato: per far capire dove siamo arrivati, come, e perché
 - le tecnologie affrontate si usano massicciamente nel mondo del lavoro contemporaneo
 - ✓ ma spesso ben nascoste da vari strati di indirezione
 - ✓ o per piccole porzioni di progetti più ampi
 - Al futuro: si scommette sull'importanza di alcune tecnologie ancora non diffusissime ma in forte espansione
- Troppa poca attenzione al presente?
 - Le tecnologie “passano”, i concetti sottostanti no
 - Scegliere *solo* sulla base della tecnologia presente significa
 - non aver considerato il passato
 - non aver scelto su cosa scommettere per il futuro...

Scegliere una sola risposta

- Sicuramente inserirò il corso nel pds
- Sicuramente NON inserirò il corso nel pds
- Devo ancora decidere, in quanto:
 - Sto valutando il carico di lavoro rispetto ai CFU
 - Orari scomodi per le mie esigenze; precisare:
.....
.....
 - Non sono abbastanza interessato agli argomenti trattati
 - Altro:.....
.....

Testi di Riferimento

- Maurice Herlihy, Nir Shavit
The Art of Multi Processor Programming
Morgan Kaufmann
- Paolo Ancilotti, Maurelio Boari.
Programmazione concorrente e distribuita.
Mc GrawHill

Testi di Riferimento

- Giuseppe Callegarin,
Corso di Informatica Generale vol. III.
Edizioni CEDAM Padova.

Testi di Riferimento

- Peter A. Darnell, Philip E. Margolis.
C Manuale di Programmazione.
Seconda Ed. Mc GrawHill.
- Warren W. Gay,
Linux Socket Programming by Example.
Que.
- W. Richard Stevens
UNIX Network Programming – Interprocess Communications.
Seconda Ed. Prentice Hall.

Testi di Riferimento

- Raoul-Gabriel Urma, Mario Fusco, Alan Mycroft
Java 8 in Action: Lambdas, Streams, and functional-style programming
Manning
- Brian Goetz
Java Concurrency in Practice
Addison Wesley
- Doug Lea
Concurrent Programming in Java
Seconda Ed. Addison-Wesley
- Raymond Roestenburg, Rob Bakker, Rob Williams
Akka 8 in Action
Manning