Controllo autonomico E SKELETON

controllo delle prop non funzionali

computazioni strutturate

Esempio performance

+ semplicemente

permette di avere
un modello di PERF

rispetto al caso
non strutturato
- che richiede una
conoscenza +
approfondita della
logica dell'applic.

formalizzazione semplice delle
regole che determinano il comportamento
delle parte analisi- plan

Esempio : fault tolerance

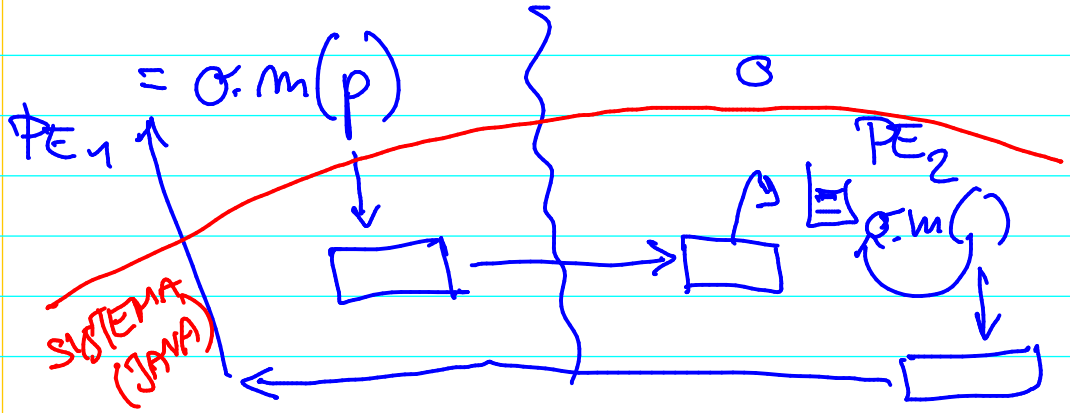
permette uso di tecniche
radicalmente diverse rispetto al
caso non strutturato

muskel

ambiente a skeleton FULL JAVA

ogni architettura che supporta JAVA supporta anche muskel

+) muskel usa RMI per implementare un modello MDF



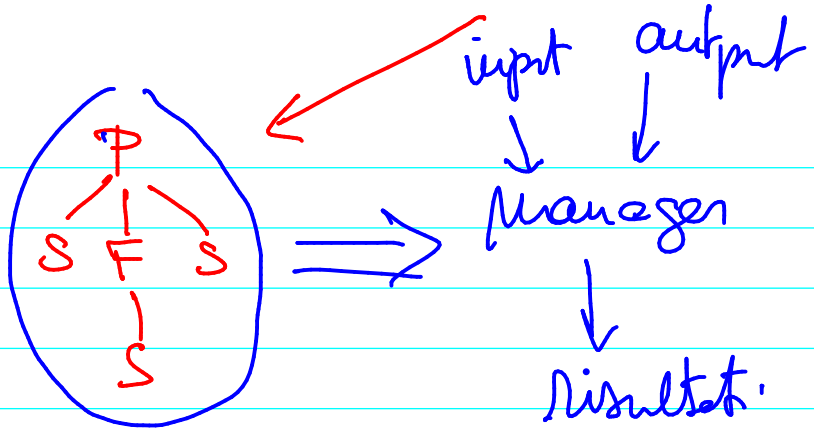
invece di usare una new() per dichiarare l'oggetto
si usa una lookup(...); +)

..
+) uso di libreria di skeleton in modo dichiarativo

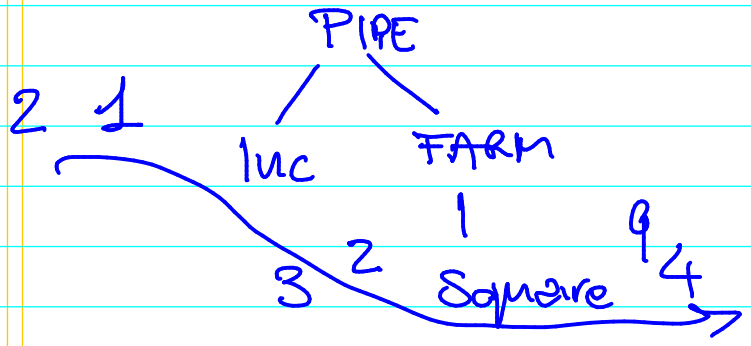
standard

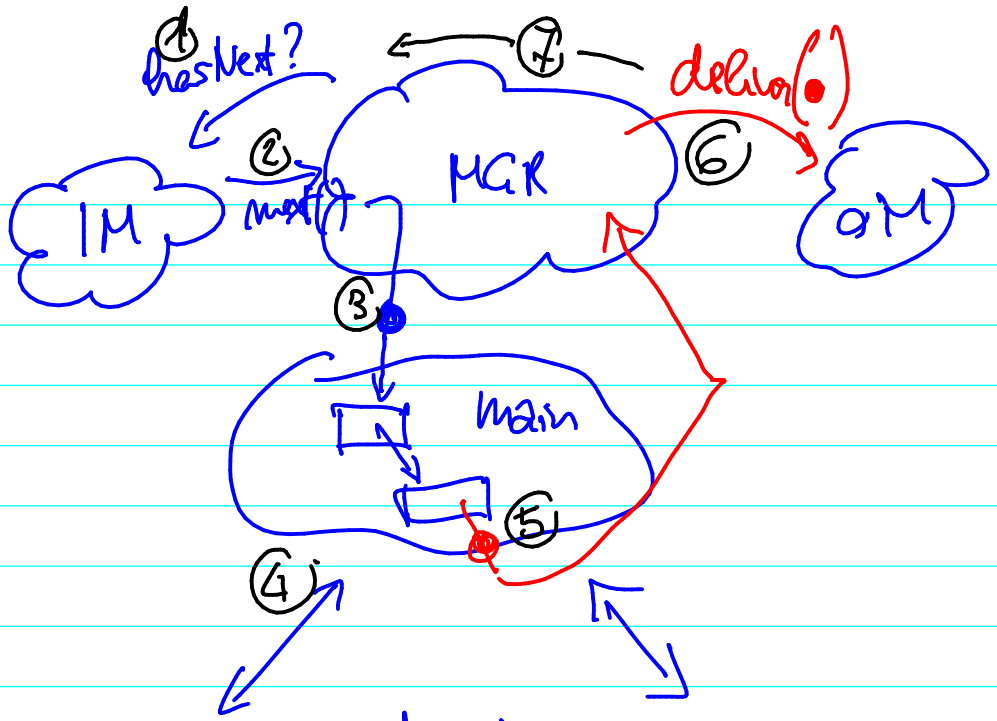
il programma è dichiarato ed eseguito da un manamanager

manager



Esempio Eclipse





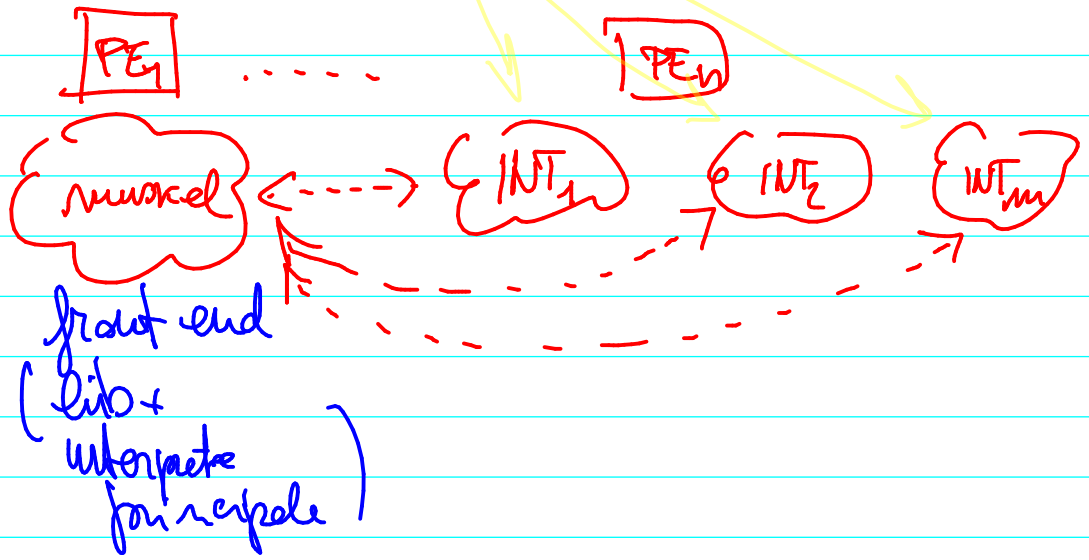
perDegree

specificato dell'utente

+) ha un controllore "autonomico"

si occupa della fault tolerance "BEST EFFORT"

-) servono "piccole" azioni per inizializzare il framework
sono quelle che permettono di far girare gli interpreti remoti



1) interp remoti stabili : RTS su PE lanciato una volta per tutte

usa serializzazione delle Compute
per specializzare gli interp di volta in volta

2) launcher (muskel)

ad ogni lancio fa partire interp remoti

ASSUNZIONI (per 1) e per 2))

=> accesso alle PE remote mediante SSH/SCP

(=> avere login su tutti i PE)

1) lancio un RemoteInterpreter su tutte le macchine

setCode
stats
compute(MDFi)

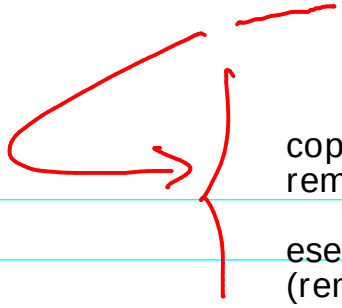
crea un oggetto interprete
cerca un registry RMI
pubblica oggetto interprete sul registry
fa partire un thread PresenceThread

risponde alle invocazioni dell'interprete principale quando si cercano le risorse per implementare il parDegree richiesto dall'utente con il contratto al manager

1) apre un multicast socket
2) join su un gruppo
3) attende messaggi "discovery"
4) risponde con le proprie coordinate (utilizzate per la lookup dell'interp.)

2) Launcher Muskel

usa una lista di host noti (file (XML))
per ognuno degli host tenta un ssh
se ha successo lancia l'interprete remoto



copia di muskel.jar nel classpath remoto

esecuzione del RTS (remoteIntepreter)

1 bis (o 2 bis) utilizzare un demone RMID

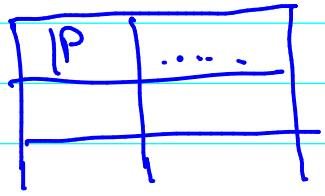
non c'e' un interprete remoto sempre in esecuzione
ma si dichiara che a seguito di una lookup(nome)
devo attivare un certo oggetto

VEDIAMO COSA ACCADE SULL'INTERPRETE PRINCIPALE

sulla macchina utente (FRONT END del calcolo)

manager.compute() ← che succede?

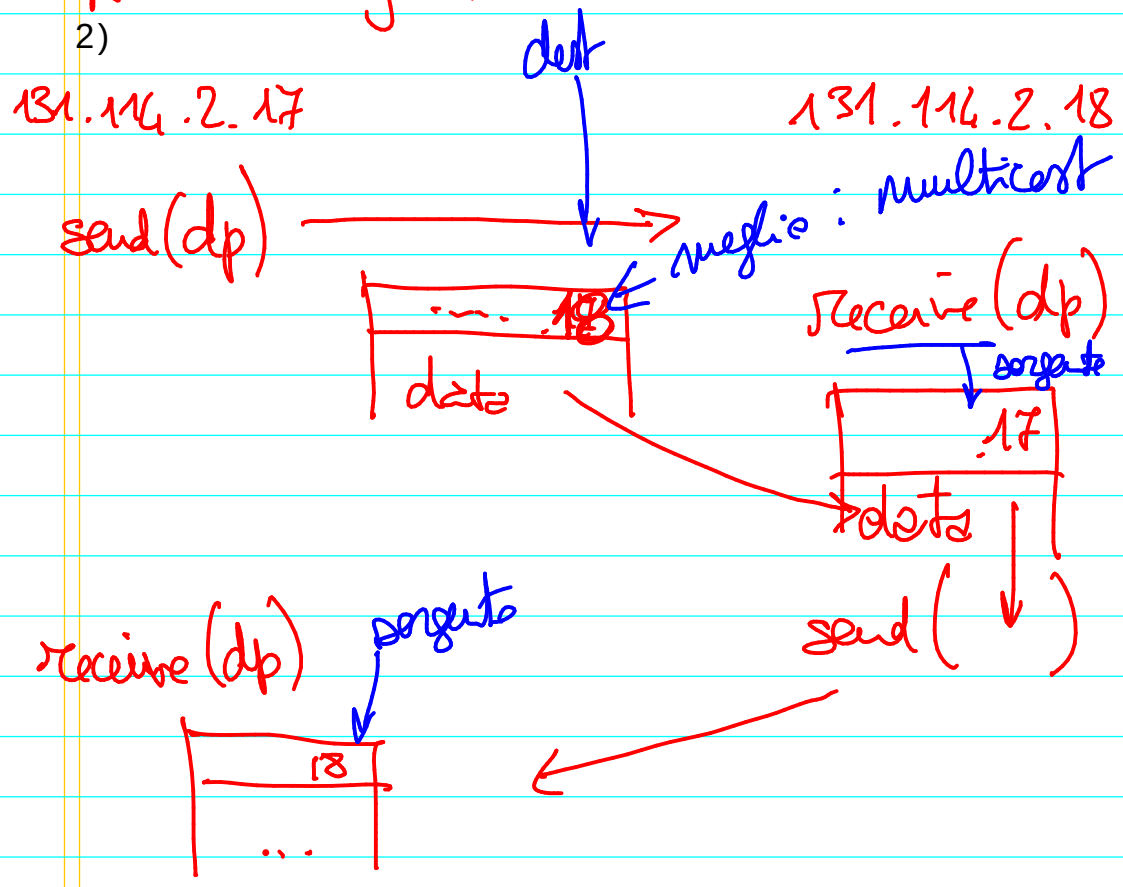
1) si cercano le risorse (parDegree)
mandare un multicast sulla porta muskel
per ogni risposta "ricordarsi l'interprete remoto"



indirizzi per lookup dell'interprete remoto

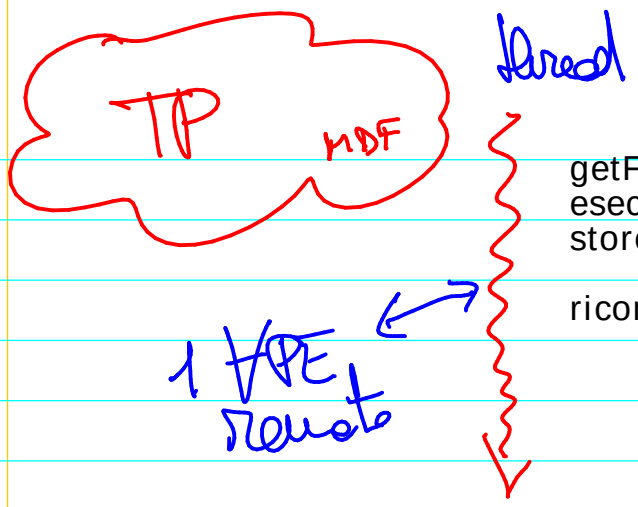
MULTICAST JAVA

2)



2) lancio un thread per ognuno (parDegree) degli interp. remoti che mi servono

- cerca un MDFi fireable
- invoca interpRemote.compute(mdfi)
- memorizza i token risultato



getFireable();
esecuzione remota;
store dei token risultato;

ricomincia il ciclo;

3) vado avanti finchè ci sono istr fireable nel TP oppure ce ne possono essere inserite (ci possono essere attese per task che ancora devono arrivare su stream non chiusi ...)

4) alla fine termino i thread e il programma (main utente)

5) nel frattempo: ricevo task dall'input stream, per ogni task istanzio un grafo MDF col token in ingresso , e lo metto nel task pool

Control Thread :

1) lookup del worker remoto sulla macchina di cui gli è stato passato l'ip/nome

2) worker.setProgram() (quindi gli mando le f e le g delle istruzioni mdf)

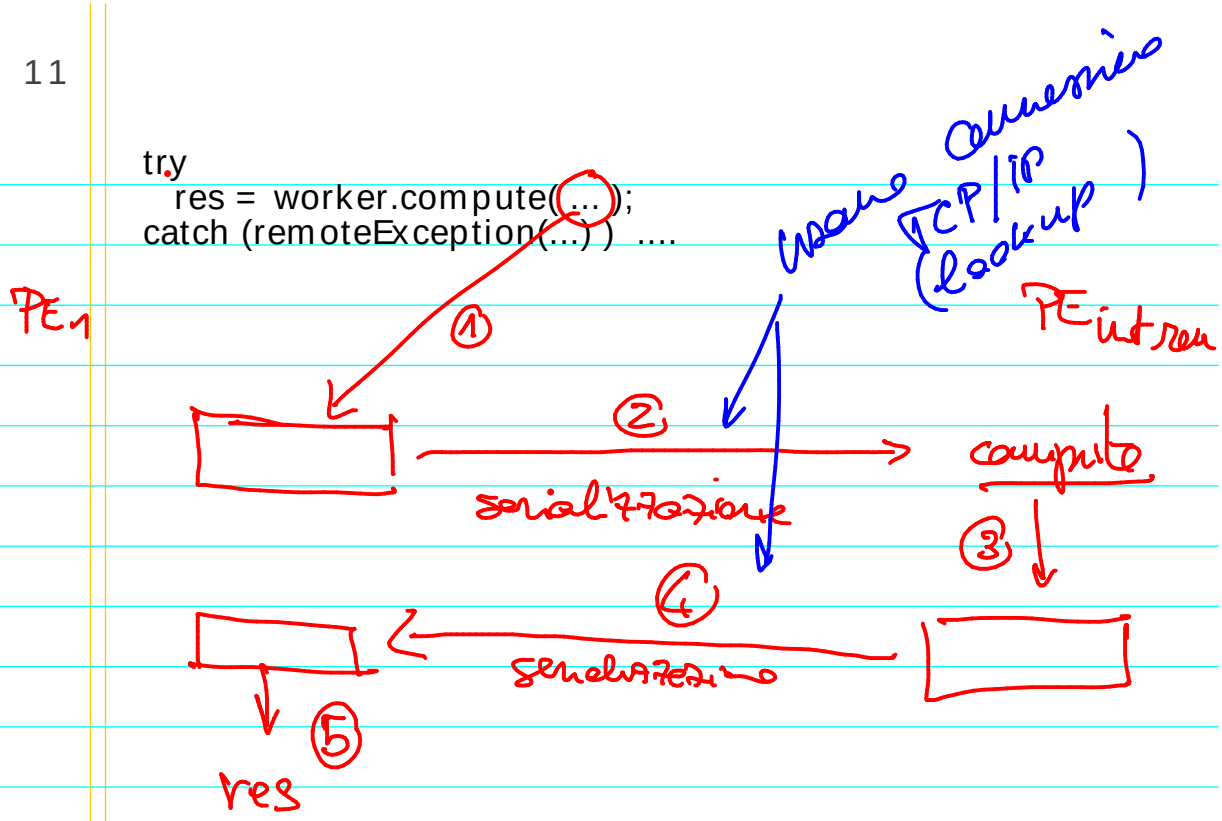
3) entro in ciclo

- predo una MDFI fireable dal task pool
- res = worker.compute(mdfi)
- processo res

  se ha token risultato li metto nel posto giusto del grafo MDF che sto eseguendo

  se sono risultati "finali" invece osm.deliver(...)

```
try
   res = worker.compute(...);
catch (remoteException(...)) ....
```



Cause RemoteException

problemi con la connessione TCP/IP (lookup)
problemi di esecuzione remota

remoteException =>

control thread rimette nel task pool MDFi non calcolata

notifica il fallimento e termina

manager :

quando riceve la notifica
istanzia un nuovo control thread
e lo aggiunge al pool degli esecutori