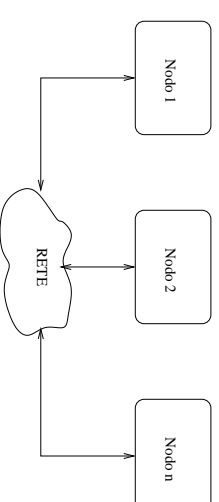


## Introduzione

“Un cluster di computer è un sistema di elaborazione, parallelo o distribuito, che consiste di un insieme di computer, indipendenti ma connessi tra loro, che lavorano insieme come una singola risorsa integrata di computazione.” [BB99].



Roberto Capuano, UNISA

1

# ClusterRMI: Invocazione di Metodi Remoti su Cluster di Computer

Roberto Capuano  
 Università degli studi di Salerno  
 Facoltà di Scienze Matematiche, Fisiche e Naturali  
 odio@libero.it  
 18 ottobre 2001

Roberto Capuano, UNISA

ClusterRMI

18 ottobre 2001

## Presentazione

**Problema:** Nella piattaforma Java è implementato un sistema di invocazione di metodi remoti chiamato RMI. Il sistema risulta semplice da usare ma lento prestazionalmente.

**Obiettivo:** Migliorare l'architettura RMI in modo da:

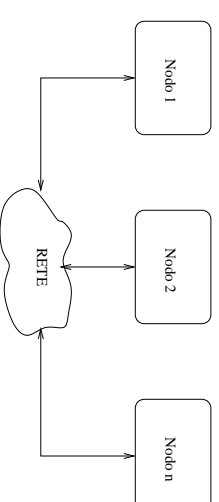
1. Rendere facile la programmazione di sistemi cluster in Java.
2. Produrre applicazioni efficienti in termini prestazionali.

Roberto Capuano, UNISA

2

## Introduzione

“Un cluster di computer è un sistema di elaborazione, parallelo o distribuito, che consiste di un insieme di computer, indipendenti ma connessi tra loro, che lavorano insieme come una singola risorsa integrata di computazione.” [BB99].



Roberto Capuano, UNISA

1

ClusterRMI

18 ottobre 2001

## Soluzione

Realizzare una libreria di programmazione (ClusterRMI) che permetta:

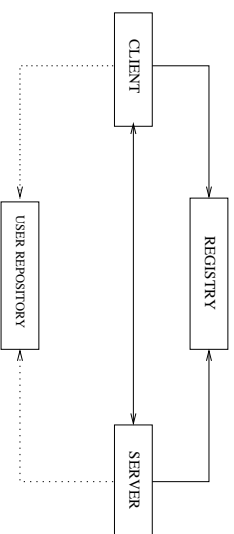
1. Ridurre il ciclo di vita delle applicazioni distribuite, eliminando l'uso di post-compiler.
2. Rendere efficiente a run-time il tempo di computazione inter-nodo, ed intra-nodo in modo da:
  - (a) Ridurre il tempo intra-nodo usando un processo di serializzazione leggero.
  - (b) Ridurre il tempo inter-nodo mediante l'uso di un protocollo di comunicazione efficiente.

Roberto Capuano, UNISA

3

## Architettura del sistema: una visione dall'esterno

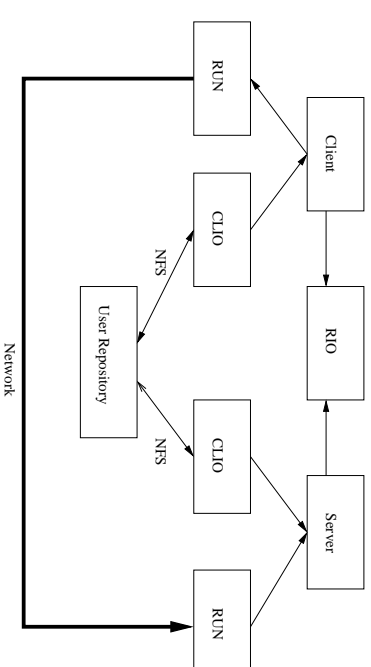
1. Ogni oggetto istanziato in una macchina del cluster implementa un servizio.
2. I servizi sono fruiti dalle applicazioni client.



Roberto Capuano, UNISA

4

## Architettura del sistema: una visione dell'interno.



Roberto Capuano, UNISA

5

QuiserRMI

18 ottobre 2001

## RUN: Codifica e trasmissione dati

**Invio di una richiesta:** Codifica del servizio richiesto e dei parametri della richiesta (Callgram).

**Serializzazione:** Codifica dei dati per la trasmissione. Si codificano il tipo effettivo dei singoli oggetti con degli identificatori. Ed i riferimenti ad altri oggetti vengono sostituiti con numeri progressivi.

**Trasmissione dei dati:** Il messaggio viene diviso in pacchetti, spedito sulla rete, e ricostruito sull'altro host. Eventuali errori di trasmissione vengono corretti, con la ritrasmissione.

Roberto Capuano, UNISA

6

QuiserRMI

18 ottobre 2001

## RUN: Controllo

1. Generazione dei thread per:
  - (a) Trasmissione dei dati.
  - (b) Esecuzione dei servizi.
2. Controllo dei thread mediante un architettura ad eventi. Gli eventi elementari sono:
  - (a) La ricezione di un pacchetto appartenente ad una sessione di comunicazione.
  - (b) Completamento della ricezione di una richiesta di invocazione remota (Callgram).

Roberto Capuano, UNISA

7

## RIO

RIO è diviso per le sue funzionalità in due parti: una parte locale non condivisa, ed una parte globale condivisa.

**Parte Locale:** Le funzionalità della parte locale operano strettamente con il modulo RUN: forniscono informazioni sui servizi attivi all'interno dell'applicazione server.

**Parte Globale:** Le funzionalità della parte globale sono quelle tipiche di un registry. Sono state implementate come qualunque altro servizio creato dall'utente, solo che è fornito di base.

## CLIO

Sono tre le funzioni raggruppate in CLIO:

1. Caricatore di classi: ClusterClassLoader.
2. Generatore di classi: generazione delle classi stub/skeleton.
3. Modificatore di classi: aggiunta dei serializzatori.

## CLIO

Distinguiamo le classi dell'applicazione client in base al loro ruolo.

**Classe Attiva:** Se ha un ruolo attivo nell'infrastruttura di comunicazione.

- Se definisce uno stub, uno skeleton oppure implementa l'argomento di una chiamata remota.

**Classe Reattiva:** Se ad una sua modifica il sistema reagisce con la generazione di una classe attiva.

- Se implementa una interfaccia di servizi remoti, oppure può essere argomento di una chiamata remota.

**Classe Passiva** Tutte le altre.

## Risultati

L'architettura da noi realizzata:

- È semplice, in quanto l'utente non è costretto ad allungare il ciclo di vita del software con una fase di post-compilazione.
- È innovativa, in quanto introduce il concetto di modifica a run-time di una classe allo scopo di aggiungere i serializzatori.
- È orientata ad un sistema cluster, in quanto:
  1. il registry memorizza l'associazione tra servizi e host per l'intera rete,
  2. il protocollo di comunicazione si basa sulla relativa affidabilità delle comunicazioni locali,
  3. il protocollo di serializzazione si basa sul presupposto che le definizioni delle classi siano tutte disponibili, e nella corretta versione, sia al client che al server.

## Risultati

- È efficiente, in quanto secondo test preliminari ClusterRMI risulta essere più efficiente di RMI nella fase di introspezione, e lo eguaglia nella fase di serializzazione e trasmissione. Benchmark più precisi sono in fase di allestimento.

Roberto Capuano, UNISA

12

## Bibliografia

### Riferimenti bibliografici

[BB99] Baker and Buyya. *Cluster Computing at a Glance: Architectures and Systems*, volume 1. Prentice Hall, NJ, USA, 1 edition, 1999.

Roberto Capuano, UNISA

13