Tesi di Laurea

Meccanismi Content-Aware per il Web Dispatching

Candidato:

Mauro Andreolini

Relatore:

Prof. Salvatore Tucci

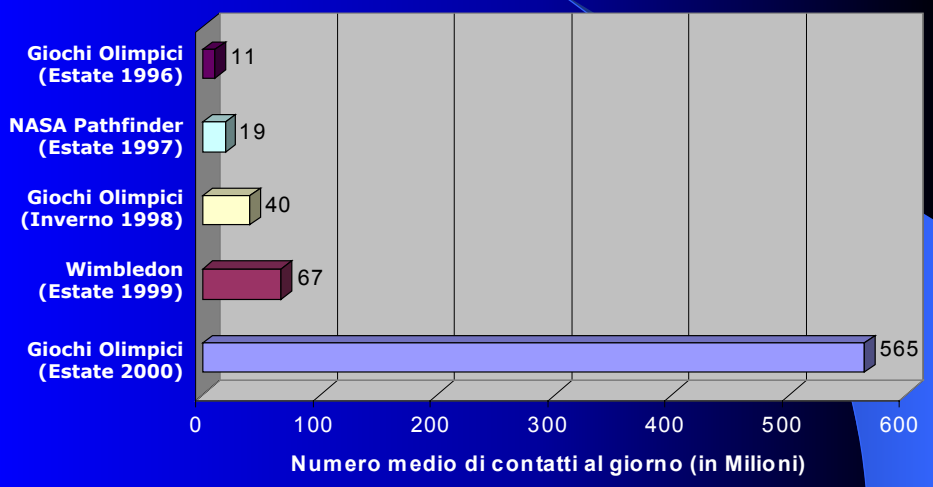
Correlatore:

Prof. Michele Colajanni

Sommario

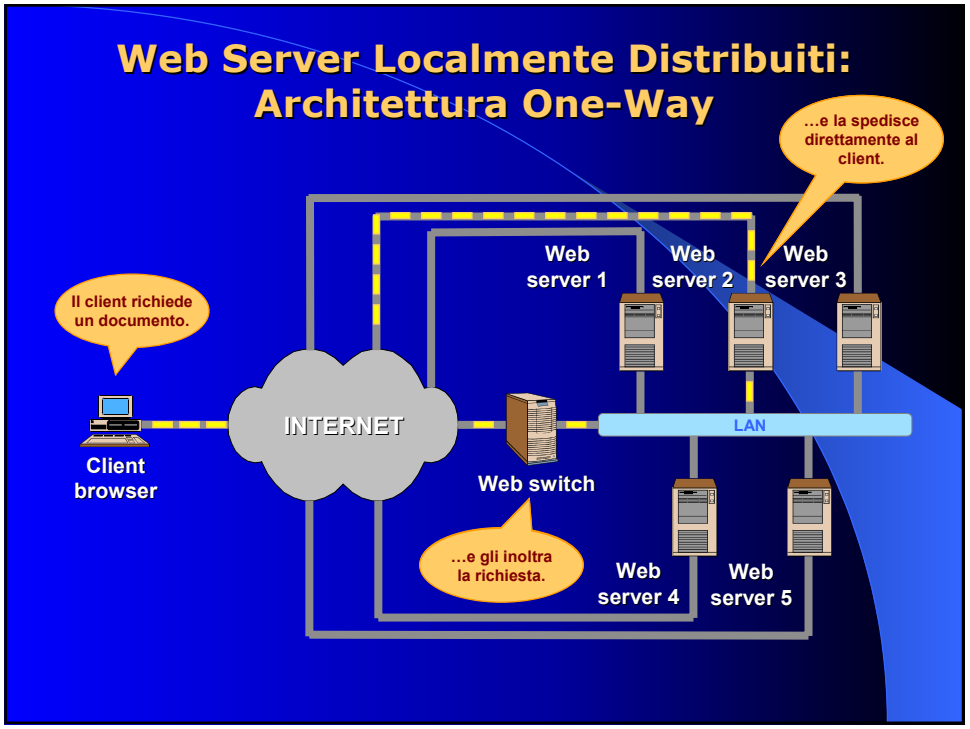
- Siti Web popolari: traffico e soluzioni
- Web Server Localmente Distribuiti
- Algoritmi di selezione
- Algoritmi statici vs. dinamici
- Politiche *CAP* e *LARD*
- Architettura del prototipo
- Risultati sperimentali
- Conclusioni e sviluppi futuri

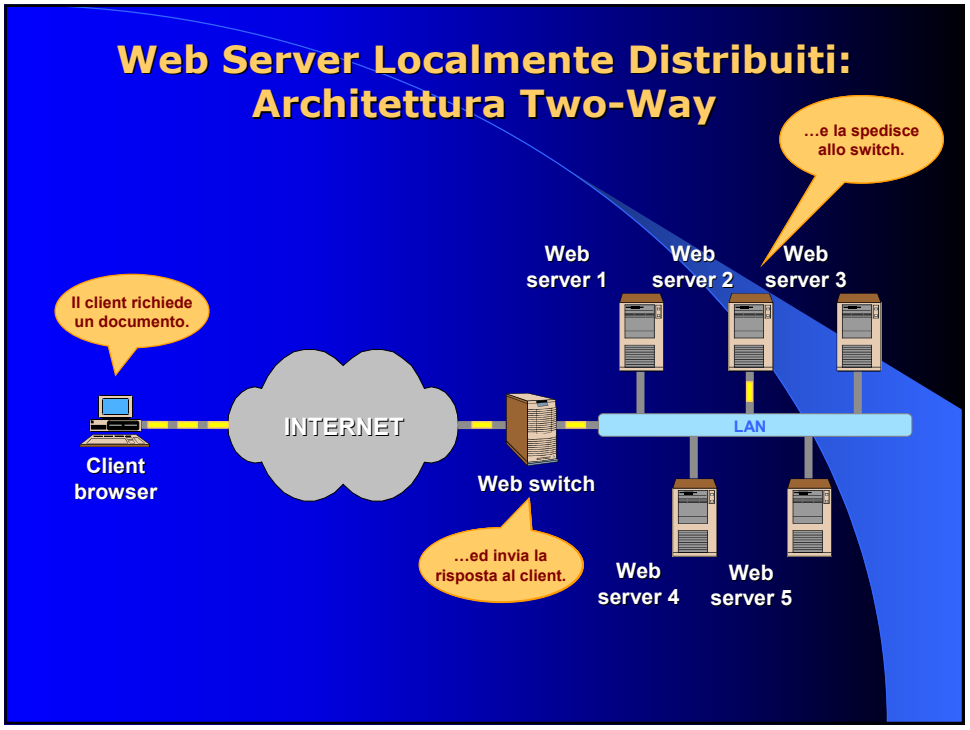
Siti Web popolari: Traffico



Siti Web popolari: Soluzioni

- Incrementare la capacità del Web server
- Mirroring
- **Distribuire le richieste**
  - Architetture locali o geografiche
  - Trasparenza
  - Scalabilità
  - Affidabilità
  - Tolleranza ai guasti


Web Server Localmente Distribuiti:
Architettura One-Way
...e la spedisce direttamente al client.
Web server 1
Web server 2
Web server 3
Il client richiede un documento.
INTERNET
LAN
Client browser
Web switch
...e gli inoltra la richiesta.
Web server 4
Web server 5


Web Server Localmente Distribuiti:
Architettura Two-Way
...e la spedisce allo switch.
Web server 1
Web server 2
Web server 3
Il client richiede un documento.
INTERNET
LAN
Client browser
Web switch
...ed invia la risposta al client.
Web server 4
Web server 5

Web Server Localmente Distribuiti: Algoritmi

- Quarto Livello OSI
  - TCP/IP
  - Content blind
  - Statici e dinamici
- Settimo Livello OSI
  - Application (HTTP)
  - Content aware
  - Statici e dinamici

Algoritmi Statici vs. Dinamici

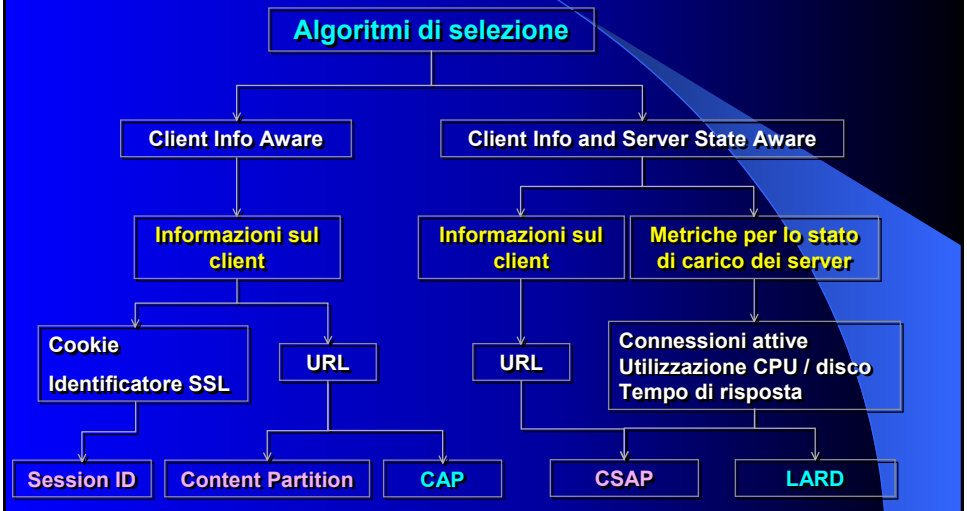
Statici

- Facile implementazione
- Overhead trascurabile (sullo switch)
- Possibili situazioni di sbilanciamento del carico

Dinamici

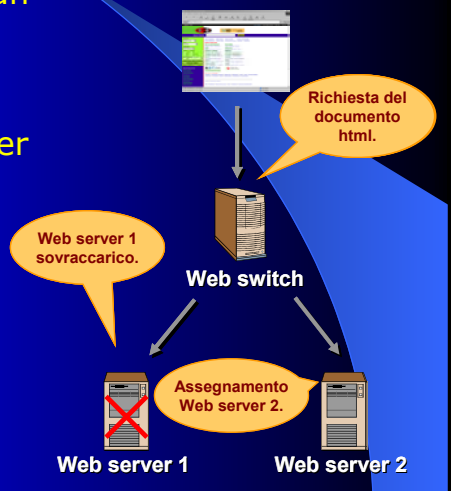
- Implementazione più complessa
- Overhead di comunicazione (tra switch e server) e di computazione (sullo switch)
- Migliore bilanciamento del carico a parità di politica adottata

Algoritmi di Settimo Livello



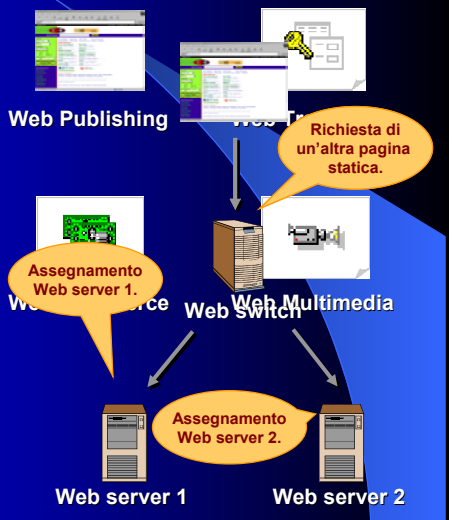
Politiche implementate Locality Aware Request Distribution (LARD)

- Primo assegnamento di un documento ad un server in modalità round robin
- Le richieste successive per un dato documento sono indirizzate allo stesso server
- Se il server è sovraccarico, la richiesta viene inviata temporaneamente al server più scarico



Politiche implementate
Client Aware Policy (CAP)

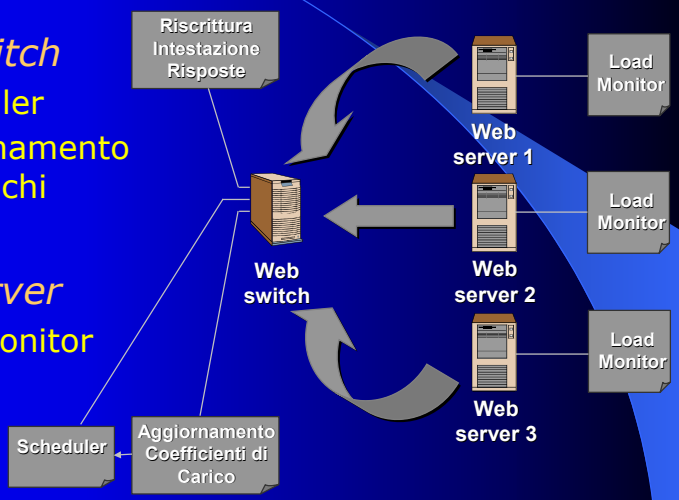
- Politica del tutto innovativa
- Distinzione fra più classi di servizio
- Ciascuna classe di servizio viene schedulata in modalità round robin sullo stesso insieme di server



Componenti del sistema

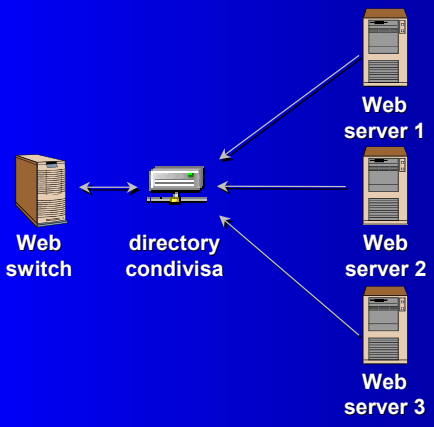
Web switch
- Scheduler
- Aggiornamento dei carichi

Web server
- Load Monitor

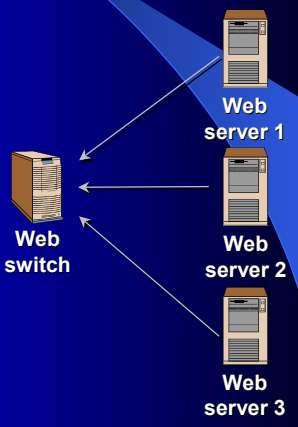


Meccanismi di comunicazione

Network File System



Socket



Meccanismi di comunicazione

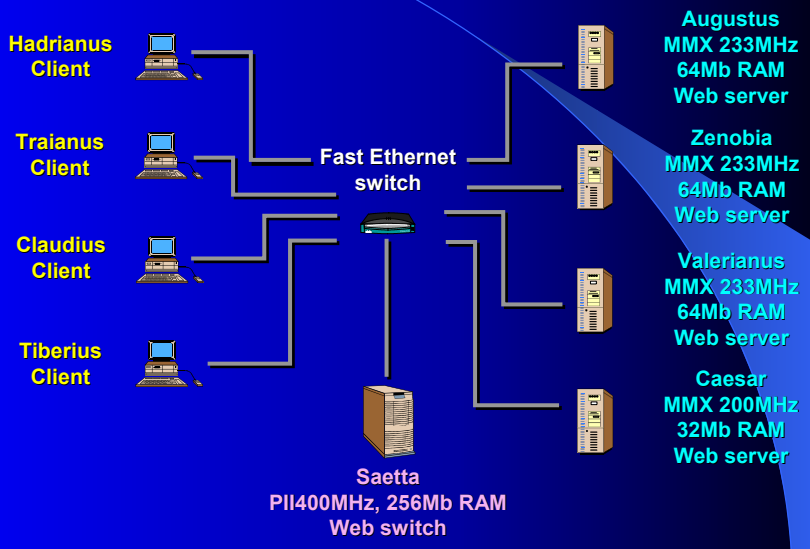
Network File System

- Facile implementazione
- Viene introdotto un overhead di comunicazione non trascurabile

Socket

- Implementazione più complessa
- L'overhead di comunicazione introdotto è trascurabile

Architettura del prototipo



Scenari di test

Webstone (modifiche apportate)

- Introduzione del Think Time
- Modellazione struttura di una pagina Web (Corpo + Oggetti incorporati)

Workload

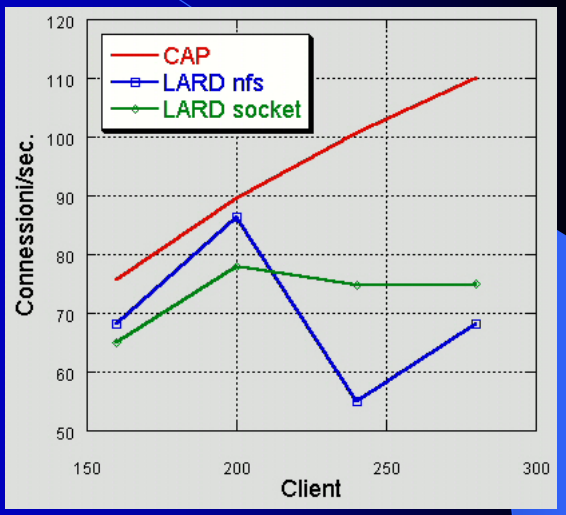
- Light Workload(90% statiche, 10% dinamiche leggere)
- Heavy Workload(90% statiche, 10% dinamiche pesanti)

Metriche misurate

- Connessioni / sec. aperte dal cluster
- Mbit / sec. in uscita dal cluster
- Tempo di risposta (sec.) del cluster
- Utilizzazioni medie CPU server e switch

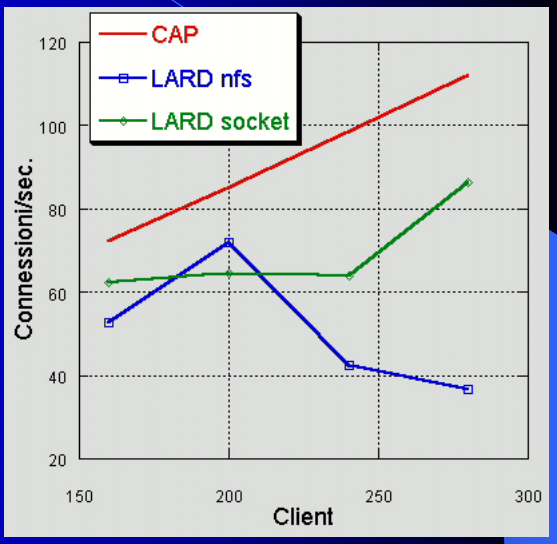
Risultati sperimentali: Modello di carico leggero

- La politica CAP mostra una spiccata scalabilità
- La politica LARD con monitor nfs manda in saturazione il cluster
- La politica LARD con monitor socket è leggermente migliore rispetto alla LARD nfs
- Lo strano andamento della LARD è dovuto ad un cattivo bilanciamento del carico



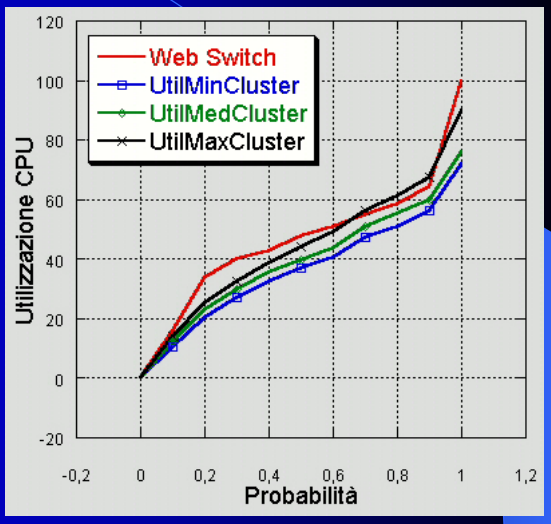
Risultati sperimentali: Modello di carico pesante

- All'aumentare del carico, aumenta il divario fra CAP e LARD
- Lo strano andamento della LARD nfs è dovuto ad un cattivo bilanciamento del carico



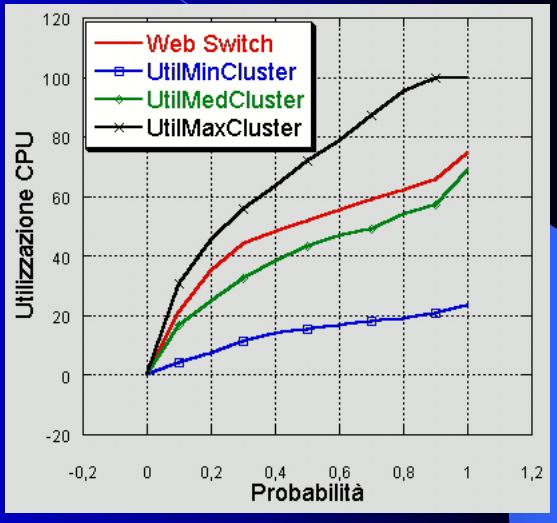
Risultati sperimentali: Bilanciamento del carico

- Studio delle curve di distribuzione delle utilizzazioni
- Esperimento con:
  - dispatcher CAP
  - 200 client
  - carico leggero
- Il bilanciamento del carico è ottimo (curve di utilizzazione molto vicine fra loro)



Risultati sperimentali: Bilanciamento del carico

- Studio delle curve di distribuzione delle utilizzazioni
- Esperimento con:
  - dispatcher LARD nfs
  - 200 client
  - carico leggero
- Il bilanciamento del carico è peggiore rispetto alla politica CAP



Conclusioni

- Realizzazione di un sistema di Web server localmente distribuito
- Implementazione di algoritmi *content aware*
- La politica *Client Aware* si dimostra superiore alla *Locality Aware* nel caso di cluster composto da macchine omogenee di media potenza

Sviluppi futuri

- Implementazione del dispatcher content aware a livello di sistema operativo
- Algoritmi adattativi per l'analisi dei coefficienti di carico
- Utilizzo di metriche diverse per la misura dello stato di carico dei server