Sistemi Web Tolleranti ai Guasti

Candidato:

Paolo Romano

Relatore:

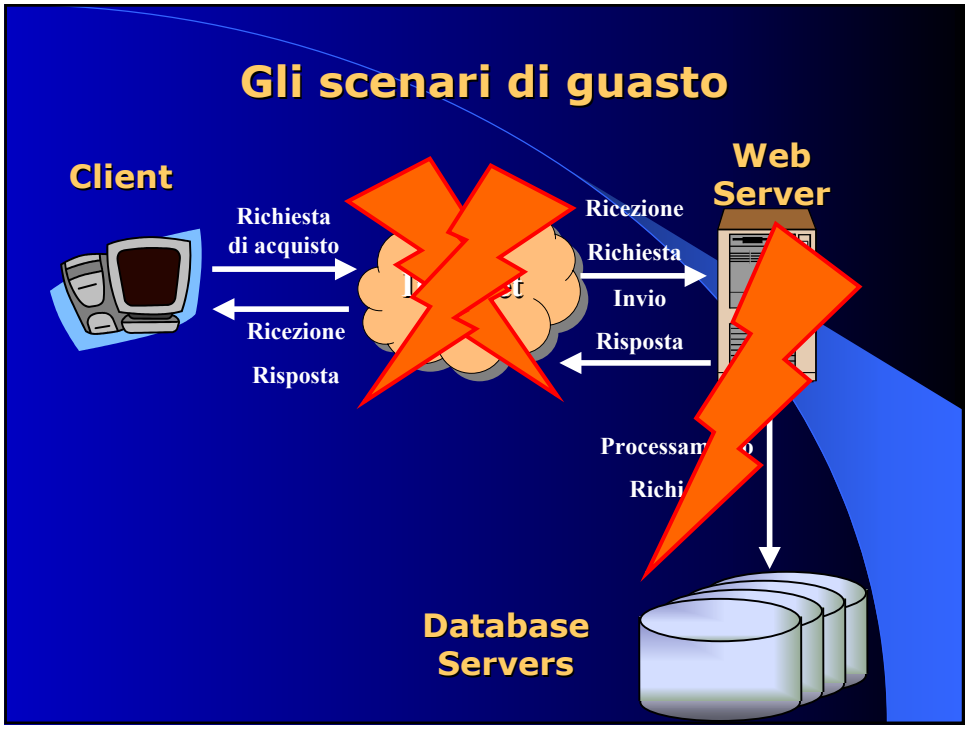
Prof. Salvatore Tucci

Correlatore:

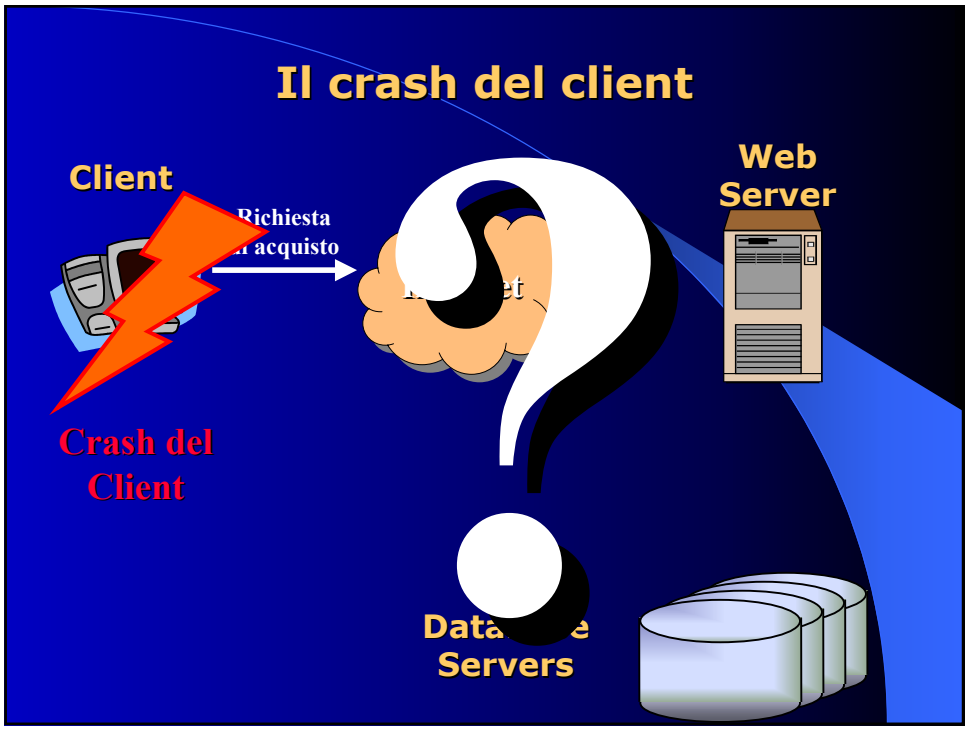
Prof. Bruno Ciciani

Sommario

- Il problema: garantire semantica exactly once alle transazioni Web.
- Sistema end-to-end tollerante i guasti basato su un Web Cluster:
  - Progettazione.
  - Dimostrazione correttezza.
- Verifica formale di HTTPR:
  - Rappresentazione tramite FSM.
  - Validazione con l'ausilio di un tool di Model Checking: analisi esaustiva dei comportamenti definiti dalle specifiche del protocollo.


Gli scenari di guasto
Client
Richiesta
di acquisto
Ricezione
Risposta
Ricezione
Richiesta
Invio
Risposta
Web
Server
Database
Servers


Dal punto di vista del client
Client
Richiesta
di acquisto
Caduta della
connessione
TCP



Exactly once processing

- Il client ignora cosa sia successo alla richiesta e deve decidere se inoltrarla nuovamente o meno.
- Il processamento lato server ha semantica "at-most-once". Gli aggiornamenti sulla base di dati sono atomici (*all or nothing*).
- Se il client decidesse di ripetere la richiesta, si rischierebbe di processare due volte la stessa richiesta: semantica di tipo "at-least-once".
- Si deve poter garantire il processamento exactly-once delle richieste del client.

Caratteristiche delle applicazioni Web

- Lato client: le applicazioni non hanno accesso a disco (Applet Java), essenzialmente per ragioni di sicurezza.
- Lato server: l'approccio più comune per garantire un adeguato livello di prestazioni è quello del Web cluster.
- Data layer: per definire un meccanismo generale, gli application servers eseguono degli aggiornamenti su basi di dati distribuite.
- I tempi di ritardo dovuti ad Internet non sono prevedibili e potenzialmente molto lunghi.

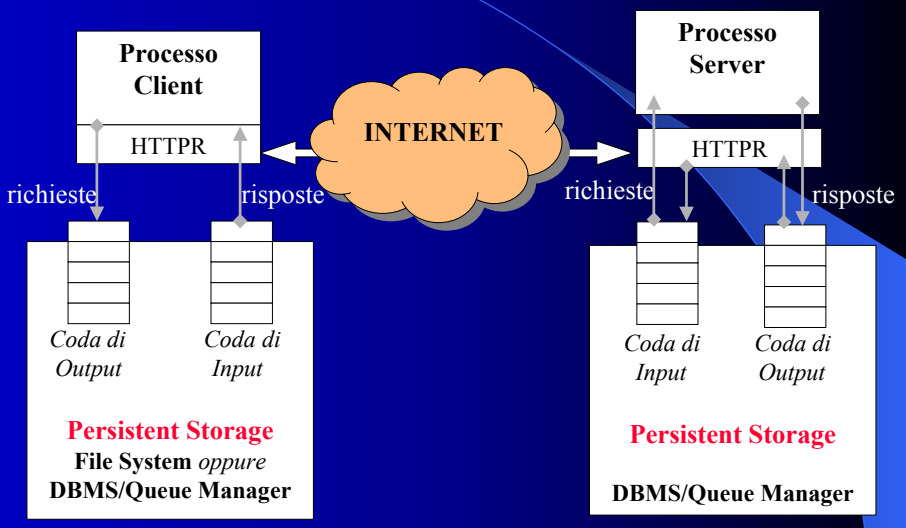
La soluzione proposta

- HTTPR: *Reliable HTTP*. Protocollo di messaging affidabile basato su HTTP.
- Le specifiche sono state recentemente rese pubbliche dalla IBM.
- I messaggi sono incapsulati all'interno di POST HTTP.
- Sfrutta un supporto persistente sia sul client che sul server, garantendo che i messaggi siano consegnati una ed una sola volta anche in caso di guasti della rete o degli hosts.
- Nel caso in cui la consegna dei messaggi non sia possibile, ciò è affidabilmente segnalato all'utente.

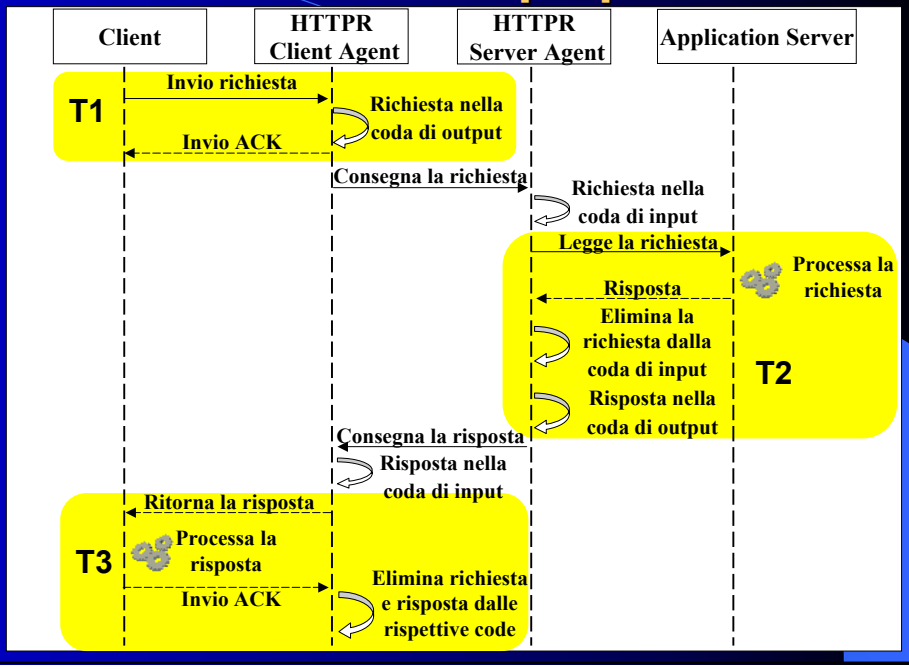
HTTPR: Vantaggi

1. Lato client le applicazioni possono ricorrere alle informazioni di stato memorizzate da HTTPR su supporto persistente (disco, DMBS, Queue Manager), senza la necessità di avere accesso diretto a disco.
2. Le applicazioni non devono curarsi di gestire i guasti dovuti al sottosistema di comunicazione. Decomposizione delle funzionalità =>
   progettazione più semplice =>
   sviluppo più rapido =>
   maggiore correttezza.
3. Poiché i comandi HTTPR sono incapsulati all'interno di POST HTTP, le richieste e le risposte sono scambiate nel formato standard del World Wide Web.

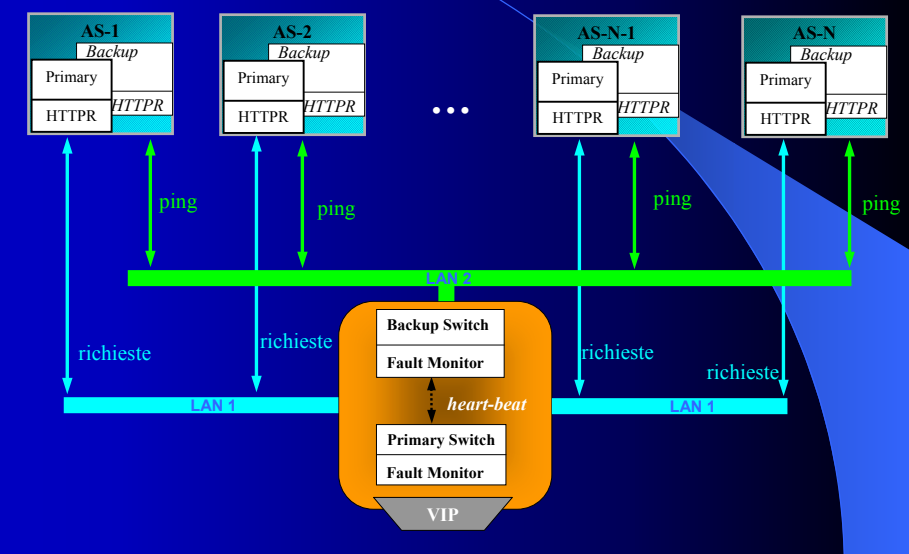
La struttura del sistema



Il meccanismo proposto



L'architettura del sistema: L'application Layer

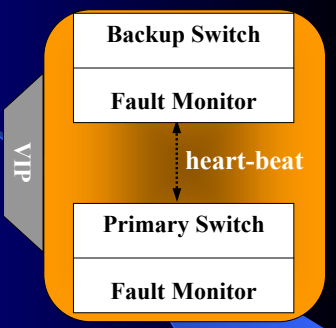


I guasti degli Application Servers

- Il fault monitor determina quale nodo sarà attivato come back-up.
- Il nodo prescelto processa le eventuali richieste presenti sulla coda di input del primary.
- Il meccanismo progettato garantisce l'exactly-once processing anche in caso di false failure detection.

Guasti dello switch

- Hot standby sparing
- IP Takeover(Gratuitous ARP)
- Sincronizzazione periodica della connection table



Vantaggi:

- Tempi di failover molto ridotti ⇒Alta disponibilità
- Implementazione Semplice (Heartbeat)

Svantaggi:

- Il back-up non mette le sue risorse a disposizione del cluster durante il funzionamento normale.

La dimostrazione di correttezza

Safety
Consistenza del client e del database.

Liveness:
…pedite situazioni
…ccanti

Significatività dei risultati

T…

- (T.1) Se il client inoltra un … …osta
- (T.2) Se qualche database… …rima o poi il risultato sarà memor…
- (T.3) Se il client riceve u… …una ed una sola volta.

…ità

- (C.1) Nessun risu… …ma stato …
- …
- … …diverse

Validità

- (V.1) Se il client inoltra una richiesta e riceve una risposta, allora il risultato è determinato dal processamento della richiesta del client da parte di un application server.
- (V.2) Nessun database server memorizza un risultato a meno che tutti i database server non abbiano espresso in maniera uniforme un voto affermativo per quel risultato

HTTPR. La verifica formale

- Rappresentazione tramite macchine a stati finiti.
  - *Client Agent*, determina il flusso di comandi HTTPR.
  - *Client* e *Server Messaging Systems*, responsabili dei meccanismi di trasmissione e di immagazzinamento delle informazioni di stato.

HTTPR. La verifica formale

- SPIN, tool di model checking studiato per analizzare la consistenza logica di sistemi concorrenti.
- Traduzione della rappresentazione FSM in PROMELA (PROtocol Meta Language).
- Definizione delle proprietà di correttezza, tramite Linear Time Logic.
- Analisi esaustiva dei comportamenti possibili per due hosts che comunicano tramite HTTPR, nel rispetto delle specifiche IBM versione 1.1

Le proprietà di correttezza.

Liveness:

At least once

Tutti i messaggi inviati saranno prima o poi ricevuti.

+

Safety:

At most once

I messaggi sono sempre ricevuti nello stesso ordine in cui sono stati inviati e senza duplicati.

EXACTLY ONCE

Le ipotesi di guasto

- L'operazione di validazione del modello comporta la memorizzazione di tutti gli stati già raggiunti. Il consumo di memoria RAM cresce esponenzialmente con la dimensione del sistema da validare.
- Un modello contenente anche i guasti di rete ha saturato 64 Gb di RAM su un nodo SP4 IBM.
- Per ridurre il consumo di RAM sono stati esclusi i guasti della rete ed i crashes degli hosts.

Sono stati considerati :

- i guasti dei persistent storages.
- i guasti degli hosts che implicano solo la chiusura della sessione e non il reset della macchina.

Sviluppi futuri

- Implementazione del sistema end-to-end per lo studio quantitivo dell'impatto sulle prestazioni del meccanismo di fault tolerance introdotto.
- Validazione di HTTPR in presenza di guasti di rete e crashes degli hosts. Possibili soluzioni per contenere il consumo di ram potrebbero essere:
  - Ottimizzare l'attuale modello, per ricavare un modello quanto più possibile astratto e semplificato, ma sufficientemente rappresentativo da garantire la significatività dei risultati ottenuti
  - Partizionare il modello in più sottomodelli da validare indipendentemente.
  - Utilizzare macchine con RAM nell'ordine delle centinaia di Gigabytes.
- Presentazione del lavoro sulla verifica formale di HTTPR presso il 10th International SPIN Workshop on Model Checking of Software, in occasione del ICSE 2003 Portland, Oregon.