C^2LD: un meccanismo per garantire “responsiveness” dei servizi Web.

Vittorio Ghini, Fabio Panzieri, Marco Roccetti
Dipartimento di Scienze dell'Informazione - Università di Bologna
Mura Anteo Zamboni, 7, 40127 Bologna (Italy)
{ghini, panzieri, roccetti}@cs.unibo.it

Sommario

Il successo di un servizio Web dipende principalmente dalla capacità del servizio di rendersi disponibile all'utente e di fornire le informazioni che questo richiede entro un tempo limite accettabile. Un approccio per l'implementazione di un servizio Web con queste caratteristiche è basato sull' introduzione di ridondanza, in particolare sulla replicazione del servizio su più servers geograficamente distribuiti in Internet. Posto che questi servers siano mantenuti mutuamente consistenti, la tempestività del servizio Web nella risposta ad un client può essere assicurata inoltrando dinamicamente la richiesta del client alla replica "più conveniente" (la replica più vicina o meno carica o ancora quella la cui connessione in rete con il client e’ meno congestionata). A partire da questo approccio, abbiamo sviluppato un meccanismo software che opera sul client e che indirizza efficacemente i requisiti di disponibilità e prontezza di risposta del servizio Web. Essenzialmente, tale meccanismo invece di inoltrare una richiesta ad una sola replica, richiede a ciascuna replica alcuni frammenti del documento Web che il client ha chiesto. La dimensione di un frammento richiesto ad una replica è calcolata dinamicamente sulla base del tempo di risposta che la replica può fornire al client stesso. In tal modo, il meccanismo proposto è in grado di adattarsi dinamicamente alle variazioni sia della rete che dello stato dei servers, e di tollerare eventuali guasti delle repliche o dei collegamenti, che dovessero verificarsi run-time. Il meccanismo è stato modellato analiticamente, implementato e validato mediante una intensa sperimentazione condotta utilizzando quattro repliche, collocate due in Italia, una in Inghilterra ed una negli USA. I risultati sperimentali evidenziano la validità della proposta, dal punto di vista sia della tolleranza ai guasti sia dei tempi di risposta. Il lavoro completo qui riassunto è disponibile in [9].

1. Introduzione

La proprietà denominata responsiveness, cioè la disponibilità (availability) del servizio e la capacità di fornire il servizio entro un tempo accettabile (timeliness), è un fattore cruciale da considerare nella progettazione di un servizio Web, per soddisfare l'utente. Dal punto di vista dell'utente, infatti, un servizio che tarda nella risposta può equivalere ad un servizio non disponibile. Un insieme di tecniche [1,2,3,6,8] sono state proposte per fornire servizi Web altamente disponibili. Queste tecniche si basano su i) costruzione di servizi Web implementati su server replicati, distribuiti localmente in un cluster di workstation, e ii) distribuzione delle richieste dei client sui differenti server. Purtroppo, come discusso in [7], queste tecniche i) sono vulnerabili a guasti del router/gateway che interfaccia il cluster dei server con il resto della rete mondiale, e ii) pur aumentando il throughput del servizio, non diminuiscono la latenza delle comunicazioni che intercorrono tra il client ed il cluster dei server. Per sopperire a queste limitazioni, in [4,5,7] è stato proposto un approccio alternativo che distribuisce le repliche dei server in tutta Internet invece che localmente. Il successo di questo approccio dipende dalla capacità di risolvere efficacemente due problemi: i) assegnare dinamicamente la richiesta di un client al server replica più conveniente, e ii) mantenere la consistenza dei dati nelle differenti repliche. Il problema del mantenimento della consistenza tra le repliche esula dagli scopi di questo articolo. Per quanto riguarda la scelta della replica più conveniente, alcune possibili strategie (basate sul DNS o incentrate sul browser) sono state analizzate in [4,5]. In ogni caso dovrà essere minimizzata la necessità di modificare i servizi di rete esistenti (ad es. il DNS). Entrambi gli approcci ai Web replicati (cluster locale di repliche o repliche distribuite in Internet) hanno comunque un punto in comune, che ne costituisce anche un limite. La richiesta di un client viene affidata ad una certa unica replica che fornirà il servizio al client, quindi mentre la replica fornisce il servizio un decadimento delle prestazioni o un guasto di quella replica (o del collegamento tra il client e quella replica) peggiora o addirittura impedisce la fruibilità del servizio da parte del client.

2. L'approccio C^2LD

Partendo da queste considerazioni, abbiamo sviluppato [9] un meccanismo (Client-Centered Load Distribution, C^2LD) per realizzare servizi Web dotati di responsiveness basati sulla replicazione dei server in Internet, che viene interamente implementato sul client. Questo meccanismo opera trasparentemente al browser e al server. Gli scopi principali del nostro meccanismo sono i) realizzare un servizio intrinsecamente fault tolerant, in grado di supplire ai guasti dei server e di portare a buon fine anche le risposte in via di completamento, e ii) minimizzare il cosiddetto User Response Time (URT) cioè il tempo necessario ad acquisire la pagina Web richiesta. In sintesi, il nostro meccanismo invece che assegnare la richiesta di un client per una pagina Web alla replica "più conveniente", come proposto in [4,5,7], intercetta ogni richiesta HTTP effettuata dal browser per una certa pagina, e la suddivide in un certo numero di sotto-richieste ciascuna per un frammento della pagina richiesta. Ad ogni server replicato disponibile viene inviata una di queste sotto-richieste. Non appena tale server completa la sua risposta gli verrà inviata la sotto-richiesta per un successivo frammento della stessa pagina, fino ad esaurimento della pagina. I vari frammenti della pagina provenienti dalle diverse repliche verranno riassemblati nel lato client per ricostruire la pagina richiesta, che verrà quindi consegnata al browser. In tal modo le sotto-richieste del client vengono servite in parallelo dalle diverse repliche, sfruttando percorsi diversi nella rete.

Il meccanismo C^2LD è progettato per adattarsi dinamicamente ai cambiamenti sia della rete (congestione nei percorsi, failure dei collegamenti) che delle repliche (sovraccarico o indisponibilità dei server). A questo scopo il meccanismo effettua un monitoraggio periodico (con un certo intervallo di monitoraggio di S millisecondi) per valutare le prestazioni offerte dalle repliche, principalmente sfruttando i tempi di risposta a ciascuna sotto-richiesta. Al termine della fase di monitoraggio il meccanismo seleziona le repliche a cui deve essere inviata una richiesta per un certo diverso frammento della pagina, e calcola la dimensione del frammento da richiedere. La dimensione del frammento richiesto ad una certa replica è tale che la risposta alla richiesta verrà

presumibilmente completata dalla replica circa nell'istante della prossima operazione di monitoraggio, in modo da poter valutare le performance di ciascuna replica utilizzando dati aggiornati. Il modello analitico sviluppato per il nostro meccanismo consente di determinare a run-time la dimensione dei frammenti di pagina da richiedere per minimizzare lo URT.

3. Implementazione

Il meccanismo C^2LD è stato implementato usando il modello di programmazione a threads fornito dal Java 2 Software Development Kit. Il meccanismo si colloca nel client idealmente sopra all'interfaccia del protocollo HTTP 1.1, in modo da poter intercettare le richieste HTTP inoltrate dal browser ed operare in maniera trasparente a questo. Il meccanismo C^2LD richiede che i server supportino lo standard HTTP 1.1 per i) poter effettuare richieste HTTP di tipo HEAD, e di tipo GET con l'opzione RANGE, e ii) sfruttare il meccanismo delle connessioni permanenti. Il requisito temporale che il nostro meccanismo deve soddisfare è rappresentato dallo User Specified Deadline (USD), un valore che indica quanto tempo un utente è disposto ad attendere per ottenere una certa pagina Web. Tale tempo limite è dato in input al meccanismo, che dovrà assicurarsi che lo URT risulti minore dello USD. La grandezza base usata nel controllo è il Data Rate (DR_i) fornito da una certa replica *i* in risposta ad una certa richiesta HTTP, cioè il rapporto tra la dimensione del frammento richiesto e lo URT sperimentato per la risposta. Sia *S* l'intervallo di tempo che intercorre tra due successive operazioni di monitoraggio delle performance delle repliche.

L'utente accede ad una pagina Web fornendo l'URL al browser il quale invoca il metodo GET del protocollo HTTP con quell'URL. Il meccanismo C^2LD intercetta questa chiamata, fa partire il timeout con USD, ed interroga il DNS passandogli l'hostname contenuto nell'URL. Il DNS risponde all'interrogazione restituendo tutti gli IP associati all'hostname. Ottenuti gli indirizzi delle repliche, il meccanismo C^2LD inoltra a ciascuna replica una richiesta HTTP di tipo HEAD per quell'URL. Ricevendo la prima risposta alla richiesta HEAD, da parte di una certa replica *i*, il meccanismo C^2LD a) estrae dalla risposta la dimensione DS della pagina richiesta e b) inizializza il timer che farà ripetere il monitoraggio ad intervalli di durata *S*. Subito dopo effettua alcune operazioni (che verranno ripetute ogni volta che viene ricevuta una nuova risposta) ed in particolare c) stima il data rate della replica *i* basandosi sul tempo di risposta, d) calcola il tempo che manca al prossimo monitoraggio, e) calcola la dimensione del frammento che verrà richiesto a questa replica, e f) invia alla replica *i* una richiesta HTTP di tipo GET per quell'URL, aggiungendo l'opzione RANGE per ottenere il frammento richiesto. Le risposte alle HEAD (ed alle successive GET) dalle altre repliche, verranno processate con le operazioni da c) ad f).

Per adattarsi alle fluttuazioni dei tempi di trasmissione, la dimensione del frammento che deve essere richiesto ad una certa replica *i* viene calcolata in base al tempo (URT) con cui è stata servita la richiesta precedente alla stessa replica *i* e in base al tempo che manca per raggiungere il prossimo istante di monitoraggio. Si assume che la nuova richiesta verrà servita con un data rate uguale a quello con cui è stata servita la precedente richiesta. Quindi quando viene completata la risposta ad una richiesta *r-1* di tipo GET (o HEAD) diretta verso una replica *i*, una nuova richiesta *r* di tipo GET può essere inoltrata alla stessa replica *i*, chiedendo un frammento di dimensione $PS_{i,r} = \overline{DR}_{i,r} \cdot S_{i,r}^{*}$ calcolato sulla base del data rate $\overline{DR}_{i,r} = PS_{i,r-1} / URT_{i,r-1}$ della precedente risposta *r-1*, e del tempo $S_{i,r}^{*}$ che manca al prossimo monitoraggio.

È importante notare come il meccanismo C^2LD che protegge il servizio quando una replica risponde lentamente o non risponde affatto, causa problemi al server o congestione in rete. C^2LD fa partire un timeout ogni volta che viene inviata una richiesta ad una replica *i*. Se il timeout termina prima che la risposta sia arrivata si assume che la replica *i* sia *non disponibile*. Il frammento che era stato richiesto alla replica *i* ma che non era arrivato verrà richiesto ad un'altra replica. Tutte le repliche che risultano *non disponibili* vengono periodicamente controllate per verificare se continuano ad essere non disponibili.

4. Scenario Sperimentale

L'efficacia del meccanismo C^2LD è stata validata mediante un grande numero di esperimenti comparativi (60480 download di pagine) svolti nel periodo Novembre-Dicembre 1999. Gli esperimenti hanno coinvolto 4 server. La prima replica è stata posizionata nel CdL in Scienze dell'Informazione di Cesena (Univ. di Bologna), la seconda nell'International Center of Theoretical Physics di Trieste. La terza replica è stata collocata nel Department of Computing Science dell'Univ. di Newcastle upon Tyne (UK), mentre la quarta replica è stata posizionata nel Department of Computer Science dell'Università di California a S. Diego. Il browser è stato posto su una SPARCstation 5 (S.O. SunOS 5.5.1) situata a Bologna nel Dipartimento di Scienze dell'Informazione dell'Università di Bologna. La Figura 1 illustra la collocazione di browser e repliche, ed i percorsi che collegano il browser ed i server.

Figura 1. Percorsi tra client e Web servers.

In questo scenario, un servizio Web replicato può essere implementato mediante una delle 11 possibili combinazioni di due o più tra le 4 repliche esistenti. I nostri esperimenti hanno considerato tutti questi 11 diversi possibili servizi.

5. Metriche

Lo scopo della sperimentazione è la valutazione del meccanismo proposto. Si è voluto in particolare confrontare il comportamento del meccanismo C^2LD con quello del meccanismo HTTP standard di download dei documenti (la richiesta GET). A tal scopo ogni esperimento ha effettuato il retrieval di un certo file, utilizzando prima il meccanismo C^2LD su ciascuno degli 11 possibili servizi Web replicati e poi, per confronto, la modalità GET standard del protocollo HTTP 1.1 separatamente su ciascuna delle repliche che costituivano il servizio replicato. Due metriche sono particolarmente indicate per rappresentare il comportamento di un meccanismo di download: i) la percentuale di richieste di documenti che il meccanismo riesce a rendere disponibili entro il tempo previsto, e ii) il tempo (URT) impiegato per effettuare il download del documento. Ciò significa misurare il fatto che il servizio sia effettuato, e che sia effettuato in tempi accettabili. Utilizzando queste metriche come parametro di confronto è stato analizzato il comportamento del meccanismo C^2LD al variare di alcuni importanti parametri: i) il numero di repliche del servizio Web, ii) il data rate che ogni replica (e la rete) fornisce, iii) il periodo

di monitoraggio *S* adottato dal meccanismo C^2LD e, infine, iv) la dimensione del documento da ottenere. In particolare, ogni replica è stata dotata di un insieme di file aventi dimensioni comprese tra 3 Kbytes e 1 Mbytes, ed il periodo di monitoraggio *S* utilizzato è stato variato tra 50 millisecondi e 10 secondi.

6. Misure

Gli esperimenti sono stati effettuati sugli 11 possibili servizi Web implementabili a partire dai 4 server utilizzati. I valori di seguito riportati in figure e tabelle fanno riferimento quindi al valore medio ottenuto da questi 11 possibili servizi Web. Il primo risultato è illustrato nella Tabella 1, in cui compaiono la percentuale di richieste di pagine Web che si sono concluse in maniera non soddisfacente utilizzando il meccanismo C^2LD sui servizi replicati, e la percentuale ottenuta usando il metodo GET standard HTTP sulle repliche singolarmente. Come si vede il meccanismo C^2LD fornisce una elevata disponibilità del servizio, perché garantisce sempre il retrieval della pagina richiesta, al contrario delle richieste GET classiche.

Kbytes	C^2LD	S. Diego (GET)	Newcastle (GET)	Trieste (GET)	Cesena (GET)
50	0 %	0.3 %	0 %	0 %	1.65 %
100	0 %	0.25 %	0 %	0 %	5 %
200	0 %	0.2 %	0 %	0 %	4 %
500	0 %	0.3 %	0 %	0 %	1 %
1000	0 %	0.45 %	0 %	0 %	2.5 %

Tabella 1: Percentuale di Fault del servizio Web.

La Figura 2 riassume i risultati relativi ai tempi di risposta dei servizi Web, illustrando il miglioramento dell'URT introdotto dal meccanismo C^2LD rispetto al meccanismo standard applicato alla più veloce delle repliche del servizio replicato considerato, in funzione della dimensione della pagina da scaricare. In particolare, le prestazioni del meccanismo C^2LD e della GET standard sono equivalenti quando la dimensione del documento è minore di 50 Kbytes, Invece, per documenti con dimensione maggiore di 50 Kbytes le prestazioni del meccanismo C^2LD sono superiori, e crescono al crescere della dimensione del documento.

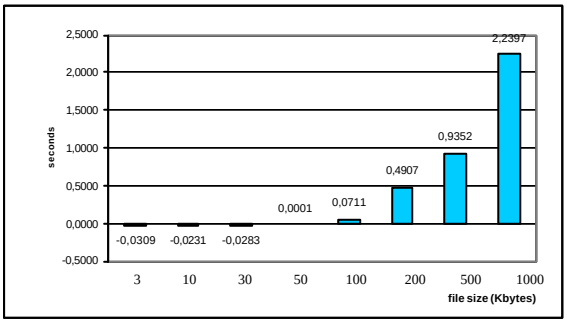

Figura 2: Miglioramento dell'URT con il meccanismo C^2LD.

La Tabella 2 riporta il comportamento di C^2LD al variare del numero delle repliche che costituiscono il servizio Web replicato, riportando la percentuale media di miglioramento dell'URT (rispetto al retrieval effettuato mediante la GET classica) al crescere del numero di repliche usate.

Numero di repliche	miglioramento percentuale dell'URT
2	4%
3	17.2%
4	21.5%

Tabella 2: Miglioramento percentuale medio dell'URT

Tutti gli esperimenti sono stati effettuati modificando anche il periodo di monitoraggio del download *S* tra 50 millisecondi e 10 secondi, allo scopo di individuarne l'influenza sull'URT fornito dal meccanismo. Risulta che il periodo di monitoraggio del download che fornisce il minor URT è pari 0.5 secondi.

7. Conclusioni

Abbiamo proposto un nuovo approccio per la progettazione di servizi Web affidabili e veloci nella risposta, basati su repliche dei server geograficamente distribuiti in Internet. Questo approccio (detto C^2LD) viene implementato sul client e fa si che tutte le repliche concorrano al download della pagina Web richiesta, ciascuna secondo le proprie possibilità, dinamicamente. Abbiamo implementato C^2LD e lo abbiamo sperimentato usando fino a 4 repliche distribuite in Internet. I risultati indicano che C^2LD è estremamente efficace nel fornire un servizio Web affidabile. Infatti, in tutte le prove, C^2LD è sempre riuscito a salvaguardare il servizio dai guasti avvenuti. Inoltre, l'aumento di fault tolerance del servizio non va a discapito delle prestazioni. Al contrario, C^2LD migliora le prestazioni rispetto al meccanismo standard di downloading delle pagine Web (metodo GET di HTTP) in quanto migliora (diminuisce) lo URT mediamente di un fattore che va dal 4% al 21% al variare della dimensione della pagina da scaricare.

Ci proponiamo ora di sviluppare il lavoro in quattro diverse direzioni: i) valutazione del carico di lavoro introdotto da C^2LD sia in rete che nei server, ii) collocazione del meccanismo in un proxy server, iii) confronto con altri meccanismi di replicazione, come quelli forniti da cluster locali di workstation, ed infine iv) studio del mantenimento della consistenza tra le repliche geograficamente distribuite.

Bibliografia

[1] R.B. Bunt, D.L. Eager, G.M. Oster, C.L. Williamson, “Achieving Load Balance and Effective Caching in Clustered Web Servers”, Proc. 4th International Web Caching Workshop, San Diego, CA, March 1999.

[2] Cisco Local Director, CISCO System Inc., White paper,1996.

[3] M. Colajanni, P. S. Yu, D. M. Dias, "Analysis of Task Assignment Policies in Scalable Distributed Web-Server Systems", IEEE Trans. on Parallel and Distributed Systems, Vol. 9, N 6, pp. 585-600, June 1998.

[4] M. Conti, E. Gregori, F. Panzieri, "Load Distribution among Replicated Web Servers: A QoS-based approach", Proc. 2nd ACM Workshop on Internet Server Performance (WISP'99), Atalanta Georgia, USA, May, 1999.

[5] M. Conti, E. Gregori, F. Panzieri, "QoS-based Architectures for Geographically Replicated Web Servers", Cluster Computing (to appear).

[6] P. Damani, P.E. Chung, Y.Huang, C.Kintala, Y.-M. Wang, “ONE-IP: Techniques for Hosting a Service on aCluster of Machines”, Computer Networks and ISDN Systems, 29, 1997, pp. 1019-1027.

[7] D. Ingham, S.K. Shrivastava, F. Panzieri, "Constructing Dependable Web Services", IEEE Internet Computing, Vol. 4, N. 1, January / February 2000, pp. 25 - 33.

[8] A. Iyengar, J. Challenger, D. Dias, P. Dantzig, "High-Performance Web Site Design Techniques", IEEE Internet Computing, Vol. 4, N 2, March / April 2000, pp. 17-26.

[9] V. Ghini, F. Panzieri, M. Roccetti. "Client-centered Load Distribution: a Mechanism for Constructing Responsive Web Services", UBLCS Technical Report n. 2000-07 , Laboratory for Computer Science, University of Bologna, June 2000, ftp://ftp.cs.unibo.it/pub/techreports/2000-07.ps.gz.