

Peer-to-Peer Network Models for a Distributed and Reliable RDF Storage

Andrea Benazzo

Relatori: Angelo Raffaele Meo, Federico Di Gregorio

Aprile 2008

Lo scopo di questa tesi è quello di proporre un database distribuito per triple RDF[3]. Le attuali soluzioni allo stato dell'arte sono basate su architetture centralizzate, che si appoggiano a database relazionali per la memorizzazione dei dati.

Il mio lavoro si è focalizzato sul progetto di una rete peer-to-peer allo scopo di distribuire il carico di lavoro su un insieme di host. In particolare, l'uso estensivo delle joint queries permette di suddividere il carico di lavoro delle interrogazioni su macchine differenti, venendo quindi a mancare il Single-Point-Of-Failure tipico delle soluzioni centralizzate.

Al momento, sono disponibili due soluzioni, non complete, per quanto riguarda le reti distribuite di storage di triple RDF. La prima proposta, Edutella, ha un'efficienza molto bassa in quanto non fornisce un vero servizio di directory. Per cercare un dato specifico, si deve inoltrare la richiesta a tutti i nodi vicini, fino a che non si riceva una risposta affermativa oppure non scatti un timeout. Non si ha quindi certezza di trovare l'informazione richiesta, anche se effettivamente disponibile.

La seconda proposta, RDF-Peers, è basata su Chord[6], ma non gestisce correttamente i picchi di carico che potrebbero presentarsi su uno o più nodi, così come non offre un supporto completo per le interrogazioni da parte dell'utente. In particolare, vengono posti dei vincoli alle richieste effettuate mediante conjunctive-queries, limitando l'espressività di questi frequenti costrutti.

Dopo questa proposta, presentata nel 2004, nessun'altra soluzione ha migliorato lo stato dell'arte.

Inizialmente ho analizzato le architetture Distributed Hash Tables, tra cui Chord, Kademlia e HiPeer[2]. L'ultima proposta, in particolare, fornisce un'ottima efficienza per quanto riguarda il numero medio di hop necessari per connettere due nodi qualunque appartenenti alla rete. L'algoritmo di distribuzione dei dati adottato da HiPeer è basato su assunzioni non paritarie per quanto riguarda le disponibilità dei nodi e il numero di triple da loro fornito singolarmente. In particolare, i nodi più stabili dovrebbero essere posti negli anelli centrali, così da ammortizzare l'evoluzione della rete sui livelli esterni, ma non si possono fare previsioni a priori sui singoli tempi di uptime. Inoltre, non viene garantito ad ogni dato lo stesso numero di repliche, nè ad ogni nodo lo stesso numero di dati di cui essere responsabile.

Questo ultimo punto assume grande importanza in quanto ogni tripletta viene indicizzata in base ai contenuti dei suoi tre campi. Queste informazioni sono composte da URI spesso

riutilizzate da altre relazioni, portando quindi a picchi di carico in corrispondenza dei nodi responsabili degli identificativi più popolari.

Mi sono quindi concentrato sullo studio di un nuovo algoritmo da sostituire all'originale, ottenendo una buona uniformità nella distribuzione dei dati. Ma la poco flessibile struttura gerarchica creata mediante gli anelli concentrici non permette una distribuzione migliore ed è soprattutto troppo legata alle singole inserzioni/rimozioni di nodi, portando ad un elevato uso della banda di rete.

Abbiamo quindi abbandonato l'architettura ad anelli concentrici, e mi sono rivolto ad una struttura basata su un anello singolo. L'idea centrale è simile a quella di base di Chord, ma nel nostro caso abbiamo deciso di modificare il modello per via delle peculiarità nella distribuzione dei dati all'interno di un tipico archivio RDF. Attribuendo segmenti di differente lunghezza ad ogni nodo, abbiamo così cercato di assegnare ad ognuno lo stesso carico di triple. Per arrivare a questo risultato, ogni nodo calcola con i vicini delle statistiche riguardanti il numero locale di triple per nodo, e contemporaneamente, ogni peer fa richiesta di queste statistiche ad altri nodi, così da avere una visione completa su tutto l'anello. Mediante l'analisi di questi dati, si riesce a stimare in modo accettabile la distribuzione del carico e il numero attuale di nodi. Questo ultimo dato è di grande importanza in quanto utilizzato come parametro per adattare il numero di informazioni scambiate in base alla loro effettiva efficacia.

I nuovi nodi andranno quindi ad alleggerire i peer attualmente più carichi, secondo i principi del Consistent Hashing.

Per quanto riguarda la gestione delle tabelle di routing, abbiamo deciso di basarci sull'idea di fondo di Symphony[5], adattandola ai segmenti di differente lunghezza, così come spiegato in Mercury[1]. In questo modo, le tabelle di routing sono popolate in base alle dimensioni attuali degli intervalli in cui è diviso l'anello.

Un'importante contributo dato dalla diretta gestione del load balancing mediante le statistiche è la conoscenza della selettività degli attributi. Ogni volta che viene ricevuta un'interrogazione composta di una serie di richieste da risolvere contemporaneamente, siamo capaci di iniziare la risoluzione a partire dall'attributo più selettivo. Questo porta ad avere meno risultati intermedi da dover spostare tra i nodi responsabili dei vincoli richiesti. Di conseguenza, ogni nodo impiegherà meno tempo ad estrarre la lista dei risultati, confrontarla con la lista precedente e trasmettere questo risultato al prossimo peer. In conclusione, la latenza richiesta per eseguire un'interrogazione viene di molto ridotta, così come la banda di rete utilizzata.

Per quanto riguarda la gestione delle interrogazioni, sono supportate pienamente sia le Atomic Triple Pattern sia le Disjunctive Queries sia le Conjunctive Queries. In particolare, queste ultime sono gestite dall'algoritmo Query Chain[4].

Per quanto riguarda gli sviluppi futuri, siamo molto interessati al testing di un algoritmo di distribuzione ottimizzata del carico. Questo algoritmo sposta le triple da un nodo molto carico al vicino scarico, indipendentemente dalle inserzioni di nuovi nodi. Se questo movimento è controllato, si riesce ad ottenere un ottimo livellamento del carico con un limitato utilizzo di risorse di rete, anche in casi in cui la rete sia prevalentemente statica.

Nella tabella seguente sono mostrati i risultati, ottenuti mediante simulazioni, su distribuzione del carico e uso della banda di rete, rappresentati mediante deviazione standard normalizzata sulla media del numero di triple per nodo. In questo modo, siamo in grado di apprezzare le differenti distribuzioni di triple sui vari nodi e possiamo confrontare la scalabilità del modello nonostante l'evoluzione continua del numero di peer. Nella prima serie di eventi,

denominata simulazione statica, ho inserito 300 nodi, mentre nella seconda parte ho gestito 10 serie di eventi, ognuna delle quali composta da 10 inserzioni e 10 cancellazioni. In entrambi le fasi, i nuovi nodi hanno contribuito con un numero fisso di triple, mentre le rimozioni non hanno tolto dati dalla rete.

Algoritmo	Simulazione Statica		Simulazione Dinamica	
	Distribuzione Triple	Banda di Rete	Distribuzione Triple	Banda di Rete
Base	28% ÷ 30%	160% ÷ 200%	40% ÷ 44%	60% ÷ 240%
Ottimizzato	7% ÷ 8%	250% ÷ 300%	11% ÷ 13%	200% ÷ 300%

Tabella 1: Deviazioni Standard normalizzate sul numero medio di triple per nodo, per la simulazione con 300 nodi e 1 milione di triple. Sono mostrati i risultati conseguiti sia con l'algoritmo base di distribuzione dei dati sia con la versione ottimizzata.

In conclusione, con questa tesi presentiamo un'architettura efficiente e affidabile per lo storage e l'interrogazione di triple RDF, proponendo una soluzione completa e più performante delle proposte precedenti.

Riferimenti bibliografici

- [1] Ashwin R. Bharambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury: supporting scalable multi-attribute range queries. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 353–366, New York, NY, USA, 2004. ACM.
- [2] Wepiwe Giscard and Simeonov Plamen L. Hipeer : A highly reliable p2p system. *IEICE transactions on information and systems*, 89(2):570–580, 2006.
- [3] Graham Klyne and Jeremy J. Carroll. Resource description framework. Recommendation, W3C, February 2004.
<http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>.
- [4] Erietta Liarou, Stratos Idreos, and Manolis Koubarakis. Evaluating conjunctive triple pattern queries over large structured overlay networks. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006*, volume 4273 of *LNCS*, pages 399–413. Springer, 2006.
- [5] G. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world, 2003.
- [6] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.