UNIVERSITÀ DEGLI STUDI DI ROMA
"TOR VERGATA"

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria Informatica

Progetto per il corso di Ingegneria del Web

SISTEMA DI PREFETCHING CLIENT-SIDE PER TRAFFICO WEB

Professore:
Prof. VALERIA CARDELLINI

Studente:
STEFANO PERNA

ANNO ACCADEMICO 2007-2008

Prefazione

La seguente relazione è stata prodotta come parte integrante del progetto svolto nell'ambito del corso di Ingegneria del Web.

Il progetto di riferimento è il C: *Sistema di prefetching client-side per traffico Web.*

La traccia integrale del progetto è riportata di seguito:

Lo scopo del progetto è realizzare in linguaggio C usando l'API del socket di Berkeley un sistema di prefetching client-side per trafico Web. Il prefetching (o caching proattivo) si pone l'obiettivo di superare le limitazioni del tradizionale caching di tipo passivo tramite un precaricamento delle risorse Web effettuato in anticipo rispetto alle successive richieste degli utenti. I requisiti salienti dell'applicazione sono elencati di seguito:

- Il supporto del protocollo HTTP/1.1 (in particolare, connessioni persistenti e chunked encoding) nel gestire l'interazione con i server Web;
- il supporto del prefetching di risorse Web sia statiche che dinamiche;
- il supporto di almeno un meccanismo di predizione delle risorse da precaricare;
- il supporto del prefetching di risorse specificate dall'utente, in base alle modalità da questi indicate;
- la gestione dello spazio di memorizzazione su disco disponibile per il prefetching;
- il logging conigurabile dell'attività di prefetching.

Indice

Capitolo 1

Introduzione

Il World Wide Web (o *WWW*) è oggi la principale fonte d'informazione globale presente. Il successo e la diffusione del *WWW* sono cresciuti negli ultimi anni con un tasso molto elevato rendendo l'accesso a tale risorsa un nodo critico nella quotidianità di molte attività oggi centrate su di esso.

Le problematiche ad esso associate sono quindi emerse rapidamente e a volte catastroficamente.

Tralasciando molti dei problemi che affliggono il *WWW*, in questa tesina concentreremo la nostra attenzione su uno dei problemi più tangibili che oggi ci troviamo ad affrontare parlando di *WWW*: il tempo di risposta e la percezione della velocità di navigazione della rete.

La percezione di questi problemi da parte degli utenti sono determinati da diversi fattori. I Web server oggi devono servire una clientela molto più ampia che nel passato, impiegando a volte molto tempo per soddisfare le richieste, specialmente se sono sovraccarichi. Dall'altro lato i Web client possono aggiungere ulteriore ritardo se non sono veloci ad analizzare i dati ricevuti ed a visualizzarli all'utente. La rete può inoltre impiegare molto tempo a trasferire le informazioni da una parte all'altra del mondo.

Possiamo facilmente immaginare quindi la profondità del problema sia in ambienti operativi critici che non, sopratutto combinando tutti i ritardi sopra descritti.

Il ritardo introdotto dai primi due fattori oggi può essere, almeno in teoria, facilmente ridotto acquistando strumentazione che consenta prestazioni

migliori. Ridurre il ritardo introdotto dalla rete è però molto più complicato, poiché è dovuto a due fattori indipendenti: il ritardo di propagazione delle informazioni e la congestione della rete stessa. Mentre la congestione può essere ridotta aggiornando le strutture di rete, il ritardo di propagazione non può essere eliminato essendo la velocità di propagazione costante.

I problemi da affrontare sono quindi essenzialmente due:

- Un tempo di risposta spesso inaccettabile.
- Un traffico crescente a ritmi vertiginosi.

Una delle metodologie oggi più diffuse per ridurre il traffico ed accelerare l'accesso alle informazioni presenti in ambito Web consiste nell'uso di tecniche di caching e di prefetching delle risorse.

Il caching rappresenta un metodo efficace e semplice per migliorare le prestazioni percepite del *WWW*, ed è impiegata da molti proxy server. L'idea di base e quella di memorizzare in locale le risorse che sono state richieste recentemente. Dato che le successive richieste per questi oggetti possono essere soddisfatte con copie locali delle stesse, il caching migliora il tempo di recupero delle risorse a cui si accede frequentemente e riduce il traffico a generato nella rete. Purtroppo recenti studi hanno dimostrato che il massimo hit rate teorico delle cache, ossia la probabilità di soddisfare la richiesta attraverso la cache, è pari al 50%, cioè un documento ogni due non può essere trovato nella cache.

Una seconda soluzione ai problemi esposti, che può contribuire ad aumentare l'hit rate della cache, è il prefetching. Questa tecnica consiste nel rendere disponibili le risorse prima che queste siano richieste, eliminando in tal modo il ritardo dovuto alla fase di reperimento della risorsa. L'applicazione pratica di questa tecnica comporta però diversi problemi, primi fra tutti l'aumento totale di traffico e la difficoltà nella determinazione delle richieste future, specialmente in ambito Web, attraverso un'appropriata funzione di predizione.

In questo lavoro tenteremo di risolvere le problematiche esposte precedentemente utilizzando un sistema di prefetching locale (client-side) combinato con un meccanismo di caching locale delle risorse.

L'idea proposta è quella di creare un sistema proxy client-side in grado di velocizzare le prestazioni della navigazione Web percepita dall'utente sfruttando tecniche di prefetching per il caricamento delle risorse.

Il software sviluppato si occuperà quindi di gestire tutte le richieste generate da un client verso la rete migliorando le prestazioni di navigazione Web in maniera automatica e non supervisionata.

Capitolo 2

Tecniche di Web Caching e Prefetching

Con la crescente diffusione del *WWW* la scalabilità e le prestazioni della rete sono divenute un problema critico e delicato.

Nel tempo è stata dimostrata l'efficacia dei sistemi di Web caching nel ridurre il traffico della rete ed il carico sui server. Gli studi condotti su queste tecnologie hanno portato a raffinare le attuali tecniche di caching introducendo sistemi di prefetching in grado di migliorare ulteriormente le prestazione dei sistemi attualmente esistenti sfruttando le reali esigenze di accesso alla rete.

2.1 Le motivazioni

Le richieste di accesso al *WWW* crescono esponenzialmente e rappresentano oggi il principale carico di lavoro per la rete Internet.

Il protocollo HTTP è oggi predominante nella rete ed in particolare viene utilizzato per la richiesta di risorse statiche quali file di testo (html, css, ...), immagini etc.

La soluzioni più naturale adottata per far fronte a queste richieste riducendo il carico della rete è stata l'uso di tecniche di Web caching.

Tali tecniche consentono di memorizzare in sistemi locali o distribuiti le risorse richieste riducendo il carico dei lavoro dei server ed il traffico generato nella rete.

Oggi la maggior parte dei browser internet (quali Internet Explorer, Firefox, Opera, Safari) utilizzano tali tecniche per salvare localmente le risorse da caricare in modo da non doverle richiedere nuovamente ai server in caso di nuovi accessi alla stessa risorsa.

Sono inoltre sempre più diffusi server proxy e sistemi distribuiti da poter utilizzare per l'accesso alla rete. Tali server utilizzano molto spesso tali tecniche per ridurre il traffico a monte del server stesso.

I vantaggi del web caching sono quindi evidenti:

- Il traffico della rete viene ridotto ottimizzando la banda disponibile, in quanto se la risorsa è stata salvata localmente non dovrà essere richiesta nuovamente.

- La latenza del Web viene ridotta a sua volta, in quanto l'uso di risorse locali o più vicine consente risposte più veloci.

- L'affidabilità della rete è incrementata, grazie ad un numero minore di richieste che saranno inoltre disponibili anche in caso di malfunzionamenti del server, in quanto salvate localmente.

- Il carico dei server sarà ridotto grazie ad un numero inferiore di client da servire, aumentando il throughput per le risorse dinamiche che richiedono necessariamente un accesso al server.

Le tecniche di Prefetching consento di migliorare ulteriormente le prestazioni dei classici sistemi di caching precaricando le risorse che l'utente richiederà nel futuro, in cache.

I sistemi di Prefetching possono presentare però problematiche non sottovalutabili relativamente al carico sui server e alla banda utilizzata nella rete. Il sistema di prefetching può infatti generare molte richieste per soddisfare il suo bisogno di precaricamento delle risorse, vanificando in parte i vantaggi introdotti dai normali di sistemi di caching. Per affrontare queste problematiche sarà quindi necessario un modello di predizione delle risorse da caricare, accurato e ben ottimizzato.

2.2 Architettura dei sistemi di Web Caching

Il Web caching può essere svolto da differenti componenti della rete. Le più diffuse tipologie sono il Caching locale ed il Proxy Caching.

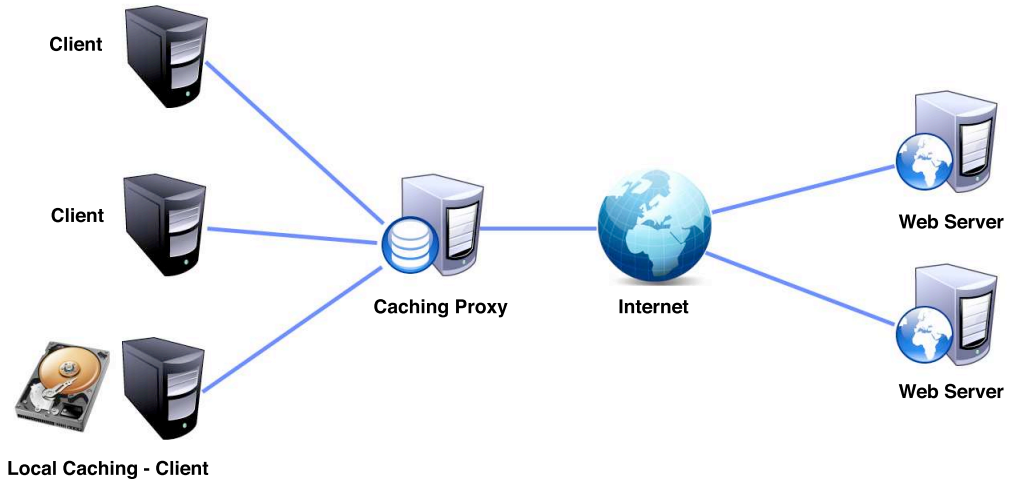

Figura 2.1: Architettura del Web Caching.

In realtà ci sono anche altri tipi di caching, come il Reverse Proxy Caching od il Transparent Caching, ma non sono d'interesse per questo lavoro.

Caching locale

Il caching locale viene svolto dal client, e generalmente sono gli stessi web browser che se ne occupano utilizzando una propria cache. La cache dei browser è comunque generalmente piccola e le tecniche utilizzate sono basate solo su semplici algoritmi di rimpiazzamento delle risorse più vecchie e meno usate.

Questo tipo di caching, anche se non ottimizzato, presenta comunque i risultati (in termini di latenza e velocità di presentazione all'utente) più performanti in quanto risiede localmente al client che genera le richieste.

Di contro la cache non potrà essere condivisa con altri client.

Proxy Caching

Il Proxy Caching viene svolto da dei server proxy situati tra i client e la rete. I server proxy generalmente utilizzano tecniche di caching più raffinate rispetto al caching locale e mantengono prestazioni elevate in quanto spesso molto vicini ai client che servono. Il vantaggio dell'uso di server proxy risiede nel fatto che la cache sarà disponibile per un largo numero di client ottimizzando i tempi di latenza, l'uso della banda ed il carico sui server.

2.3 Il modello generico di caching

Il modello più generico di web caching può essere descritto secondo l'interazione tra client e server.

In questo modello non considereremo le differenti architetture di caching, ma prenderemo come esempio un sistema composto da un Client, un Cache Manager ed il Server remoto.

Il cache manager nelle differenti architetture può essere rappresentato dal Proxy Server, dal Browser residente nel client etc.

In generale l'interazione avviene secondo la seguente modalità:

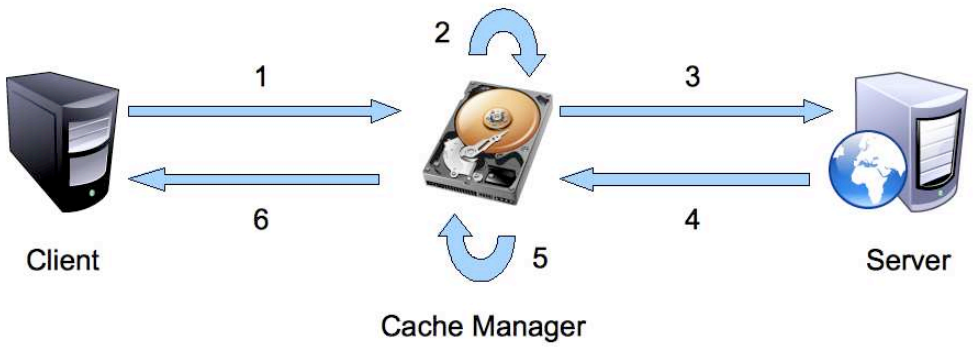

Figura 2.2: Modello generico di Web Caching.

1. Il Client invia una richiesta al Cache Manager per accedere alla risorsa R.

2. Il Cache Manager cerca nella sua memoria se la risorsa R è già presente o va richiesta al Server. Se la risorsa R è presente salta al punto 6, altrimenti continua al punto 3.

3. Il Cache Manager richiede la risorsa R al Server.

4. Il Server fornisce R al Cache Manager.

5. Il Cache Manager usa un algoritmo di caching per decidere se la risorsa R entrerà in cache o no.

6. Il Cache Manager passa la risorsa R al Client.

Il modello descrive in maniera molto generale il sistema di caching usato per il Web. La chiave critica del sistema è l'adozione di un sistema di decisione per l'algoritmo di caching in grado di ottimizzare il sistema.

2.4 Strategie di caching

Le strategie di caching sono centrate sull'algoritmo di decisione per la sostituzione delle risorse nella cache.

Il problema della cache è infatti quello di non avere uno spazio di memorizzazione infinito a propria di posizione. Quando nuove risorse devono essere memorizzate in cache, ma lo spazio a disposizione è terminato occorre adottare una strategia per decidere quali elementi rimarranno e quali andranno sostituiti.

Di seguito sono riportati alcuni degli algoritmi più diffusi:

- *OPT (Optimal)*
  L'algoritmo è stato proposto da Belady. L'algoritmo sostituisce la risorsa che non sarà usata per il più lungo periodo di tempo. Nella realtà tale predizione è impossibile e può essere effettuata solo su stime dell'uso pregresso della risorsa.

- *FIFO - First In, First Out*
  L'algoritmo sostituisce la pagina più vecchia in memoria, senza considerare la reale frequenza d'utilizzo della risorsa.

- *LRU - Least Recently Used*
  L'algoritmo è una buona approssimazione dell'algoritmo OPT. L'assunzione di base è che le risorse che sono state frequentemente usate avranno una probabilità più alta di essere richieste nuovamente nel futuro.

L'algoritmo usata nella nostra applicazione è di tipo *FIFO*.

2.5 Il Prefetching

Come visto i sistemi di Web Caching possono aiutare notevolmente a risolvere le problematiche relative alla navigazione, sia da parte degli utenti che dei server e della rete in generale.

I sistemi di Prefetching sono rivolti ad attuare ulteriori migliorie all'esperienza utente nella navigazione del Web.

Il beneficio derivante da questi sistemi è quello di garantire una minore latenza per il reperimento delle risorse da parte dell'utente. Il Prefetching però ha anche lati negativi rispetto al semplice caching.

I sistemi di Prefetching a differenza dei normali sistemi di Caching si occupano anche del precaricamento su base predittiva delle risorse nella cache. Questo compito ha però un costo in termini di prestazioni. Infatti i sistemi di Prefetching tendono in genere a richiedere maggiori risorse (hardware) sia lato client che lato server. Inoltre il traffico supplementare generato per il funzionamento del sistema genererà un'ulteriore perdita in termini di banda di rete.

Il maggiore problema, oltre all'*hit ratio* utente, per i sistemi di prefetching, è l'uso della banda di rete. Un buon sistema di prefetching dovrà minimizzare tale fattore mediante l'uso di un modello di predizione ottimizzato per lo scopo.

Tra i vari modelli esistenti analizzeremo uno molto comune basato sulla popolarità delle risorse sulle quali applicare il prefetching: il *Prefetching by Popularity*.

2.5.1 Il modello di Prefetching by Popularity

L'approccio è stato proposto da Markatos et al. L'idea di base del modello è quella di applicare il prefetching alle dieci risorse più popolari richieste dai

clients.

Il modello adottato in questo lavoro è un evoluzione di questo approccio e prevede l'applicazione del sistema di prefetching non solo alle dieci risorse più popolari, ma esteso anche ad altre risorse.

L'idea alla base del modello è quello di attuare il prefetching a tutte quelle risorse che abbiano superato un certo valore di popolarità e che stiano per scadere.

In particolare l'algoritmo valuterà tutte quelle risorse che avendo superato un determinato valore di popolarità, preimpostato, e che stiano per scadere, mantenendo un valore di utilizzo della banda accettabile ma migliorando l'hit ratio e quindi le prestazioni e la latenza percepite dall'utente.

L'uso di questa tecnica può favorire le prestazioni percepite sebbene porti ad un peggioramento in termini di utilizzo della banda e di richieste ai server web.

Capitolo 3

Disegno del sistema

Il sistema sviluppato consiste in un software con funzionalità di proxy client-side che supporta le funzionalità di prefetching di risorse web statiche e dinamiche.

Lo scopo del sistema sviluppato è quello di creare un'applicazione, facilmente gestibile dall'utente, che possa migliorare le prestazioni del caricamento delle risorse web percepito dall'utente.

I requisiti del software sono stati riassunti nella prefazione. Sono inoltre state aggiunte funzionalità opzionali quali la possibilità di configurare il proxy tramite un'interfaccia web grafica.

L'esecuzione del software può essere gestita dall'utente senza dover avere privilegi di amministratore del PC.

3.1 Caratteristiche del software

Il sistema si presenta come un'applicazione software eseguibile nello spazio utente senza necessità di privilegi di amministrazione.

La configurazione del software può essere effettuata mediante una comoda interfaccia web accessibile dal browser accedendo all'indirizzo `http://configurazione`. I parametri di configurabili dall'utente sono:

- Porta del proxy.
- Directory della cache.

- Dimensione massima della cache.
- Tempo massimo di vita delle risorse in cache.

Il software supporta il protocollo HTTP/1.1 (comprese Persistent Connection e Chunked Encoding, vedi Appendice A) e consente il prefetching delle risorse web, salvando le risorse in una cache locale configurabile dall'utente.

L'attività di prefetching è gestita mediante un meccanismo di predizione delle risorse da precaricare basato sul numero di accessi ed il tempo di vita della risorsa.

Le caratteristiche tecniche del sistema sono riassumibili di seguito:

- Applicazione eseguibile nello spazio utente.
- Funzionalità di **Transparent Proxy**.
- **Prefetching** di risorse web statiche e dinamiche.
- Meccanismo di **predizione** delle risorse da precaricare.
- Gestione della configurazione dell'applicazione mediante *interfaccia web*.
- Supporto del protocollo **HTTP/1.1**.
- Supporto di *Persistent Connection*.
- Supporto di *Chunked Encoding*.

3.2 La struttura del sistema

Il sistema è strutturato in tre librerie principali per una migliore espandibilità:

- Il gestore delle richieste.
- Il gestore del prefetching.
- Il gestore di configurazione.

Il principio di funzionamento del sistema è quello di un normale transparent proxy con l'aggiunta delle funzionalità di caching e prefetching.

3.2.1 La gestione della cache

La cache è gestita dal sistema mediante l'uso di strutture dati nelle quali sono memorizzate tutte le informazioni relative ai file di cache. Ogni file di cache viene gestito in tale struttura tramite un identificatore calcolato tramite una funzione di hash sull'indirizzo web della risorsa stessa.

La funzione di hash consente di identificare univocamente ogni risorsa e permette una migliore gestione delle risorse stesse.

La struttura della lista della cache memorizza inoltre anche altre informazioni relative alla cache quali l'ultima data di ascesso alla risorsa, il numero di accessi alla risorsa, la data in cui è stata recuperata la risorsa e il suo indirizzo web.

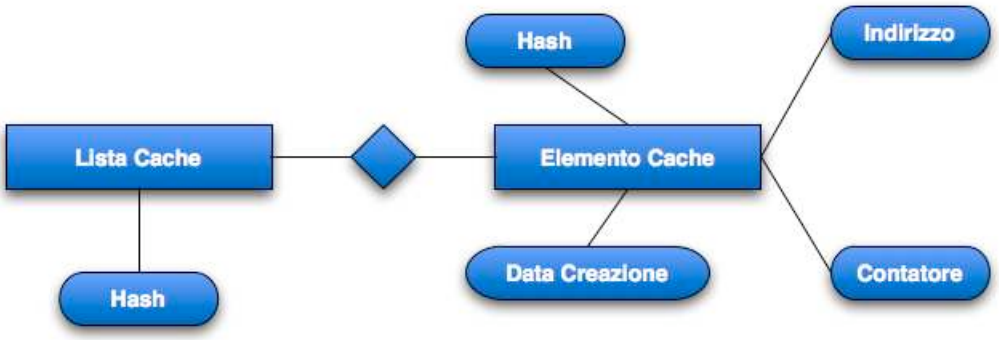

Figura 3.1: Struttura della cache.

Nella figura sono mostrate le relazioni tra la lista e gli elementi appartenenti, con i relativi attributi.

3.3 Lo schema di funzionamento

L'applicazione consiste di un ciclo principale durante il quale vengono effettuate le operazioni di prefetching e di proxy.

Pervenuta una nuova richiesta da parte del client l'applicazione verifica tramite il sistema di caching se la risorsa sia disponibile localmente oppure no.

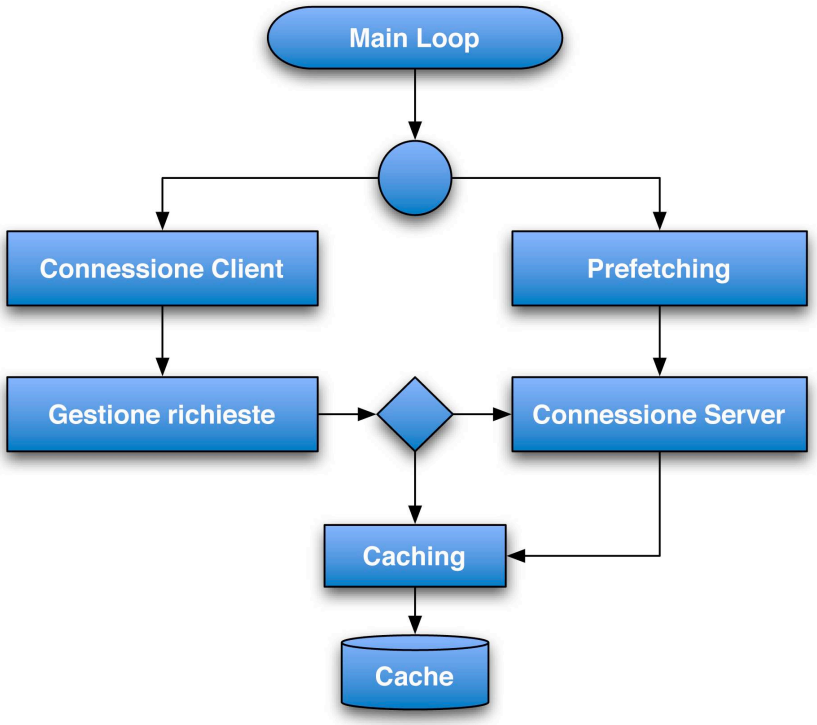

Figura 3.2: Schema semplificato generale.

Se la risorsa è disponibile il sistema la invia di risposta al client. Nel caso la risorsa non fosse disponibile allora il sistema la richiede al server Web, la sottopone nuovamente al sistema di caching e infine la invia al client.

Il sistema di prefetching si occupa solamente di predire quali risorse andranno precaricate nuovamente in cache. Ogni risorsa viene analizzata secondo l'algoritmo di predizione e viene valutato quale risorse resteranno in cache e quali andranno scartate.

Avvio dell'applicazione

All'avvio dell'applicazione vengono caricati in memoria tutti i dati relativi alle strutture dati della cache recuperati dai file di supporto all'applicazione.

Ottenute le liste degli elementi in cache il sistema avvierà l'algoritmo di prefetching verificando tramite l'algoritmo di predizione quali risorse andranno

richieste nuovamente ai web server.

Alla fine di questa fase l'applicazione si metterà in ascolto di nuove connessioni da parte del client.

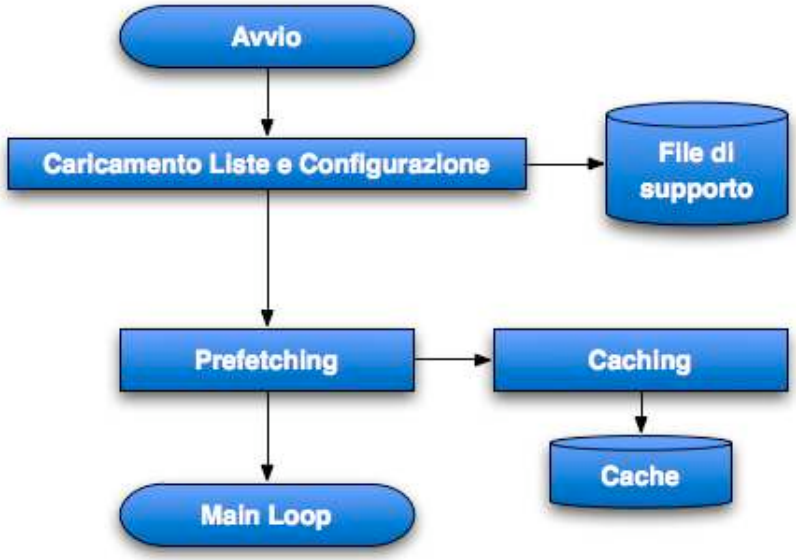

Figura 3.3: Avvio dell'applicazione.

Gestione delle richieste

La gestione delle richieste viene effettuata per ogni nuova connessione ricevuta da parte del client.

Le richieste sono analizzate e sottoposte al calcolo del valore di Hash per la risorsa specificata. Il sistema verificherà se nella lista della cache sia presente l'elemento con l'hash appena calcolato.

Se la verifica andrà a buon fine il sistema recupererà la risorsa salvata nella cache e la invierà al client. Verranno inoltre svolte le funzioni di prefetching per il calcolo del nuovo ranking per la risorsa selezionata, in particolare aggiornando il contatore di accessi alla richiesta e l'ultima data di accesso.

Se la risorsa non è disponibile in cache il sistema la richiede nuovamente al server web. Ottenuta la risorsa specificata, il sistema proporrà la risorsa al sistema di caching il quale valuterà se poter inserire la risorsa in cache.

La risorsa sarà infine inviata al client.

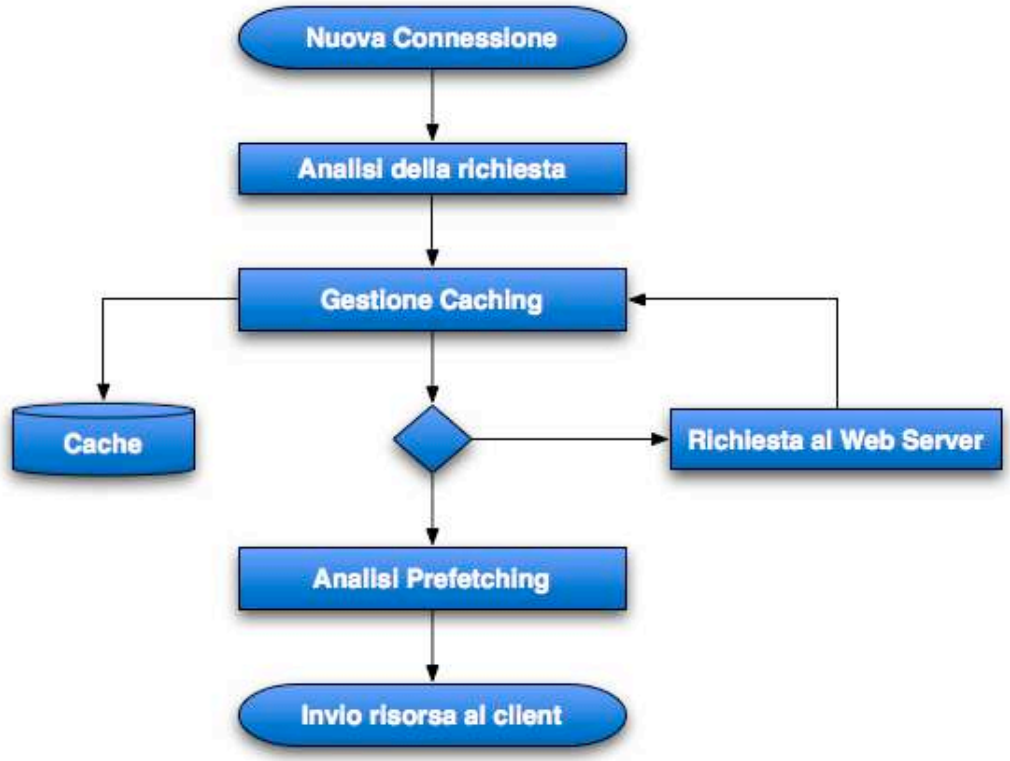

Figura 3.4: Gestione delle richieste.

Gestione del prefetching

La gestione del prefetching si occupa di predire quali risorse web andranno precaricate. Ogni risorsa presente in cache è caratterizzata da attributi che ne identificano la data di creazione e il numero di accessi.

L'algoritmo di predizione verificherà per ogni risorsa se il numero di accessi è maggiore o minore di una determinata soglia prefissata e se il tempo di vita

della risorsa (TTL) sia scaduto. Nel caso entrambe le verifiche abbiano successo, il predittore richiederà nuovamente le risorse ai server web analizzando l'indirizzo salvato tra gli attributi della risorsa.

Se viene analizzata una risorsa per cui il TTL sia scaduto ma non presenta le caratteristiche necessarie per essere richiesta nuovamente dal sistema di prefetching, allora tale risorsa verrà eliminata dalla cache.

Ottenuta le nuove risorse il sistema provvederà infine ad aggiornarle all'interno della cache.

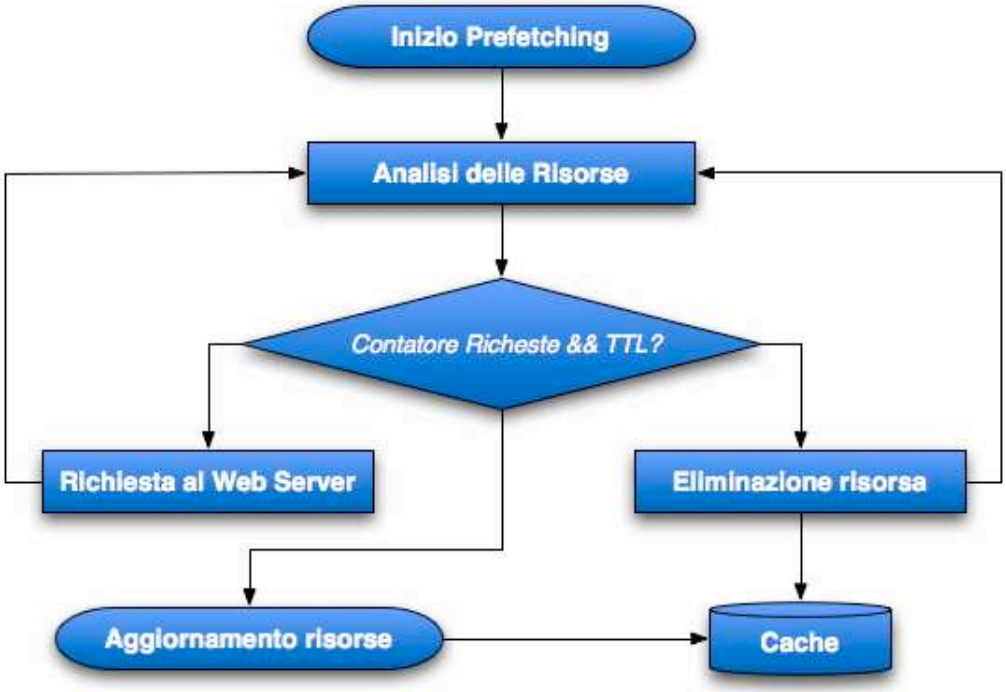

Figura 3.5: Gestione del Prefetching.