Progetto B1: Sistema Chord

Marina Calcaura
matricola 0334044
Corso di Sistemi distribuiti e Cloud Computing
Laurea magistrale in Ingegneria Informatica,
Università di Roma – Tor Vergata
marina.calcaura@students.uniroma2.eu

***Abstract*—Il presente documento mira a delineare il funzionamento e le decisioni di progettazione adottate durante la realizzazione del progetto assegnato nel contesto del corso di Sistemi Distribuiti e Cloud Computing.**

I. INTRODUZIONE

L'evoluzione delle reti distribuite e dei sistemi peer-to-peer ha rivoluzionato la gestione e la condivisione di risorse e dati in ambienti informatici distribuiti. In questo contesto, come un protocollo chiave per la creazione di sistemi di hash table distribuite (DHT) che forniscono una base solida per l'organizzazione, la ricerca e l'accesso a dati in modo altamente scalabile e affidabile emerge Chord. Questo documento rappresenta un'analisi dettagliata dell'implementazione di un sistema basato sul protocollo Chord.
La simulazione del sistema è stata condotta utilizzando Docker.

II. ARCHITETTURA

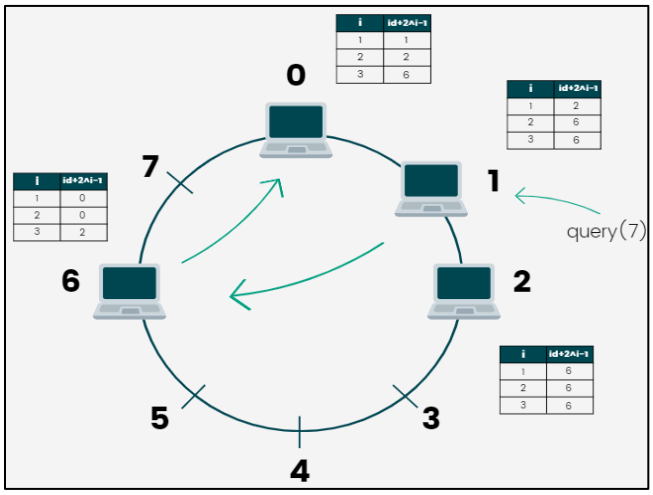
Per comprendere appieno il funzionamento del sistema implementato, è fondamentale esaminare la sua architettura di base.

A. Struttura

Ogni nodo all'interno della rete è posizionato in modo circolare all'interno di un anello virtuale. Questa topologia garantisce che ogni nodo abbia un successore e un predecessore direttamente sulla circonferenza dell'anello, semplificando notevolmente le operazioni di instradamento.

Un aspetto cruciale dell'architettura di Chord è l'assegnazione di identificatori univoci a ciascun nodo. Questi identificatori sono stati calcolati utilizzando la funzione di hash SHA-256. L'utilizzo di questa funzione garantisce una distribuzione uniforme degli identificatori lungo l'anello virtuale, contribuendo così a una distribuzione equa delle responsabilità tra i nodi.

Ogni nodo è responsabile della gestione di un intervallo specifico di identificatori. Questa divisione delle responsabilità permette ai nodi di essere designati come "proprietari" di risorse particolari, che a loro volta sono identificate tramite identificatori. Ad esempio, se esiste un nodo con identificatore "X", quel nodo sarà responsabile di tutti gli identificatori compresi tra il suo predecessore e "X" stesso. Questa suddivisione equa delle responsabilità assicura che il carico di lavoro sia distribuito in modo uniforme all'interno della rete.



B. Finger Tables

Un elemento essenziale per l'efficace instradamento in Chord è l'utilizzo delle "Finger Tables" (tabelle dei finger). Ogni nodo all'interno della rete Chord mantiene una propria Finger Table.

La Finger Table di un nodo contiene una serie di puntatori a nodi lungo l'anello virtuale. Tuttavia, ciò che la rende particolarmente potente è che questi puntatori non si limitano ai nodi immediatamente successivi nell'anello, ma piuttosto puntano a nodi con identificatori significativamente lontani. Questa scelta intelligente consente ai nodi di "salire" rapidamente sull'anello virtuale verso il nodo di destinazione durante le operazioni di routing.

Per essere più precisi, la Finger Table di un nodo contiene m entry, dove m rappresenta la lunghezza degli identificatori. Ogni entry nella Finger Table è associata a un valore i, un intero compreso tra 1 e m, e punta a un nodo distante 2^{i-1} identificatori dall'attuale nodo. In altre parole, se $[FT]_p$ rappresenta la Finger Table del nodo p, allora $FT_p[i]=successore(p+2^{(i-1)})mod2^m$.
Questo approccio consente di effettuare ricerche logaritmiche durante l'instradamento all'interno della rete P2P, poiché consente di ridurre notevolmente il numero di passaggi necessari per raggiungere la destinazione desiderata, sfruttando le informazioni contenute nella Finger Table del nodo.
Periodicamente, la Finger Table di un nodo viene aggiornata in modo da riflettere le condizioni attuali della rete,

garantendo che il nodo abbia informazioni aggiornate sui suoi vicini.

C. Server registry

Il “registry” è un componente centrale che agevola l’ingresso dei nuovi nodi nella rete e fornisce un punto di ingresso al client.
Quando un nodo desidera entrare nell'anello, comunica questa richiesta al "registry" attraverso una chiamata di procedura remota (RPC). In risposta, il "registry" fornisce al nuovo nodo tutte le informazioni necessarie riguardo al suo nodo successore e al suo nodo predecessore, consentendo così al nuovo nodo di integrarsi con successo nella rete.
Per quanto riguarda il client, il “registry” utilizza una politica di selezione “Round-Robin” per indicare un nodo specifico nell’anello come punto di accesso principale per l’interazione con il sistema distribuito. Il client ha la possibilità di inserire nuove risorse oppure cercare risorse esistenti per recuperare dati. Inoltre, ha anche la capacità di rimuovere nodi presenti nella rete. Quando un nodo viene rimosso, le risorse precedentemente gestite da questo nodo vengono riallocate in modo appropriato.

III. IMPLEMENTAZIONE DEL SISTEMA

A. Linguaggi di programmazione e tecnologie utilizzate

Di seguito sono elencate le diverse tecnologie, linguaggi di programmazione e framework utilizzati nella realizzazione del sistema:

- **Go**: per implementare la logica di Chord
- **Python:** per generare dinamicamente il file “docker-compose.yml”
- **Bash**: per eseguire azioni post-configurazione
- **Docker:** per creare container che eseguono le diverse componenti del sistema, fornendo un ambiente isolato e riproducibile per l’applicazione
- **Docker Compose**: per definire come i vari componenti dell’applicazione devono essere avviati e comunicare tra loro.
- **Amazon EC2:** per sfruttare l’infrastruttura cloud scalabile e configurabile di AWS, consentendo di eseguire, gestire e scalare l’istanza virtuale sulla quale è stata effettuato il deploy dell’applicazione.
- **GoRPC:** per stabilire una comunicazione affidabile e sincrona tra i nodi del sistema utilizzando il meccanismo delle RPC. Questa ha semplificato la comunicazione tra i nodi, consentendo loro di chiamare funzioni su altri nodi in modo trasparente e integrato, facilitando cosi lo sviluppo di un’applicazione distribuita.

B. Operazioni di base

- **Inserimento di una risorsa (“Put”)**

L'inserimento di una risorsa è stato pianificato seguendo la seguente sequenza di operazioni:
1) Quando il client desidera inserire una risorsa nella rete Chord, contatta il Server Registry, che è responsabile di indirizzare le richieste ai nodi appropriati.
2) Il Server Registry seleziona un nodo responsabile per la risorsa in base a una politica specifica, come il Round-Robin. Questo nodo sarà incaricato di gestire la risorsa.
3) Il client invia quindi la richiesta di inserimento della risorsa al nodo responsabile selezionato.
4) Il nodo responsabile calcola un ID unico per la risorsa utilizzando la stessa funzione di hash utilizzata per identificare i nodi nella rete (SHA-256).
5) Il nodo responsabile verifica se l'ID della chiave della risorsa appartiene al suo intervallo di ID nella rete Chord:
   -se l'ID appartiene all'intervallo del nodo responsabile, il nodo memorizza la risorsa localmente, poiché è responsabile di quel range di chiavi.
   -se l'ID non appartiene all'intervallo del nodo responsabile, il nodo utilizza la finger table per instradare la richiesta al nodo corretto basandosi sull'ID della chiave. Questa instradazione continua finché la richiesta raggiunge il nodo responsabile corretto per l'ID della chiave.

In questo modo, Chord garantisce che ogni risorsa venga inserita nella rete e archiviata presso il nodo responsabile corretto.

- **Ricerca di una risorsa (“Get”)**

1)Per recuperare una risorsa specifica, il client deve conoscere l'ID unico della chiave associata a quella risorsa. Contatta poi il Server Registry.
2) Il Server Registry seleziona un nodo da cui iniziare la ricerca del nodo responsabile per la chiave, seguendo sempre una politica Round-Robin.
3)Il client invia una richiesta di recupero della risorsa al nodo selezionato dal Server Registry.
4)Il nodo ricevuto verifica se l'ID della chiave cercata appartiene al suo intervallo di ID nell'anello Chord:
   -se l'ID appartiene all'intervallo del nodo, il nodo è il nodo responsabile per la chiave e quindi può restituire direttamente la risorsa al cliente.
   -altrimenti, il nodo utilizza la Finger Table per instradare la richiesta al nodo corretto in base all'ID della chiave.
5)Una volta che il nodo responsabile è stato individuato, esso restituisce la risorsa associata alla chiave al client che ha effettuato la richiesta di recupero.

In questo modo, l'operazione di "Get" in Chord consente al client di recuperare in modo efficiente una risorsa specifica dalla rete distribuita, seguendo una serie di passaggi di instradamento, se necessario. La Finger Table è cruciale per garantire che la richiesta venga correttamente indirizzata al nodo responsabile per la chiave cercata, rendendo il processo di recupero scalabile ed efficiente anche in reti Chord di grandi dimensioni.

C. Gestione della dinamicità del sistema

- **Procedura di join**

La procedura di join di un nodo in Chord è un processo chiave per consentire a un nuovo nodo di entrare in una rete Chord preesistente.
Si articola nelle seguenti fasi:
1) Il nodo che desidera unirsi alla rete contatta il Server Registry, che è responsabile di indirizzare le richieste ai nodi appropriati, per ottenere un punto di ingresso.
2) Una volta ottenuto il suo successore e predecessore, il nuovo nodo comunica al suo successore la sua presenza nella rete. Durante questo scambio di informazioni, il nuovo nodo inizializza la propria Finger Table, che è essenziale per instradare le future richieste nella rete. Inoltre, il nodo informa il suo predecessore della sua presenza.
3) Il nuovo nodo assume la responsabilità di un intervallo specifico di identificatori nella rete Chord. Questo intervallo di identificatori rappresenta l'insieme di chiavi di risorse di cui il nuovo nodo sarà responsabile e recupera eventuali dati relativi a questo intervallo dal suo successore.

- **Procedura di leave**

Il processo di leave di un nodo in Chord è una procedura chiave per mantenere la stabilità e la continuità della rete. Durante la fase di uscita di un nodo, si verificano i seguenti eventi:
1) Tutti i dati precedentemente gestiti dal nodo in partenza diventano responsabilità del suo successore diretto. Questo passaggio evita la perdita di dati e assicura che le risorse siano mantenute all'interno della rete.
2) Il predecessore del successore del nodo in partenza (nodo "p") viene regolato in modo che il suo puntatore indirizzi il nodo che era il predecessore diretto di "p" prima della sua partenza. Questo aggiornamento contribuisce a mantenere la continuità dell'anello Chord.
3) L'aggiornamento del successore del predecessore del nodo in partenza (nodo "p") viene effettuato in modo che punti al successore del nodo "p". Questo passaggio assicura che il predecessore del successore e il successore del predecessore si riferiscano correttamente ai nodi successivi e precedenti nell'anello Chord.
4) Infine, il nodo in partenza viene completamente rimosso dalla rete Chord. Questo comporta anche la rimozione di tutte le informazioni associate al nodo, come le voci nella finger table e le informazioni sul suo indirizzo IP. La rimozione del nodo consente alla rete di continuare a funzionare senza interruzioni.

IV. LIMITAZIONI

Durante lo sviluppo del progetto, sono venute alla luce alcune limitazioni che è importante mettere in evidenza.

Tra queste, la questione delle collisioni è una delle più rilevanti. In particolare, in sistemi di dimensioni ridotte, possono verificarsi collisioni tra nodi o tra risorse.

Come visto in precedenza, il server Registry svolge un ruolo cruciale nella gestione delle comunicazioni tra il client e il sistema. Tuttavia, la centralizzazione di responsabilità potrebbe influire negativamente sulla performance del sistema. Il server Registry deve essere altamente affidabile. Se il server Registry dovesse fallire o subire problemi, l'intero sistema potrebbe essere gravemente compromesso.
Inoltre, l'aggiunta di nuovi nodi o la gestione di un grande numero di nodi nel sistema potrebbe richiedere un carico significativo sul server Registry. È importante avere una strategia di scaling per far fronte a questa crescita.

Altro problema che si potrebbe verificare è il partizionamento della rete. Quando la rete Chord si divide in parti isolate, i nodi in una partizione non possono comunicare direttamente con i nodi nell'altra partizione. Questo può portare a una perdita di connettività tra i nodi, che può rendere inaccessibili alcune risorse.