
REQUEST BATCH

Request Batch - Esempio

- ▶ Un servizio riceve 10.000 richieste per leggere dati da un database
- ▶ Ogni richiesta avvia una connessione separata e un'operazione su storage e sulla rete
- ▶ Di conseguenza si hanno una alta latenza e un throughput limitato
- ▶ Con Request Batch
 - ▶ Le 10.000 richieste vengono raggruppate in blocchi da 100
 - ▶ Ogni blocco è inviato ed elaborato come un'unica operazione
 - ▶ Riduzione significativa del tempo totale e del numero di round trip

Request Batch

- ▶ Intento: combinare varie richieste per utilizzare in modo ottimale la rete
- ▶ Problema: quando vi sono numerose richieste di piccole dimensioni possono, la latenza di rete e il tempo di elaborazione del sistema lato server (incluse la serializzazione e deserializzazione) possono causare un dispendio (overhead) di risorse e di tempo significativo e scarsa efficienza
- ▶ Come ridurre la latenza e l'overhead causate da un grande numero di piccole richieste?

Soluzione

- ▶ Inviare più richieste in modo aggregato (in un batch o lotto)
- ▶ Il client o un proxy accumula le richieste per un breve intervallo o fino a un numero massimo. La quantità da accumulare può essere determinata in base alla quantità di dati da inviare e ricevere quindi per usare meglio la banda di rete
- ▶ A ogni richiesta è associato un ID per tracciarla e mappare successivamente la risposta
- ▶ Il batch è inviato come un'unica richiesta al server. Il ricevente disaggrega il batch di richieste, processa le singole richieste, e restituisce le risposte in modo aggregato
- ▶ Lato client, al ricevimento del batch di risposte, le risposte saranno smistate ai richiedenti originali

Componenti

- ▶ Request Sender: raccoglie le richieste individuali
- ▶ Batch Assembler: aggrega più richieste in un batch
- ▶ Batch Processor (Receiver): elabora il batch e produce le risposte corrispondenti
- ▶ Response Splitter: distribuisce le risposte ai client originali

Prof. Tramontana - Novembre 2025

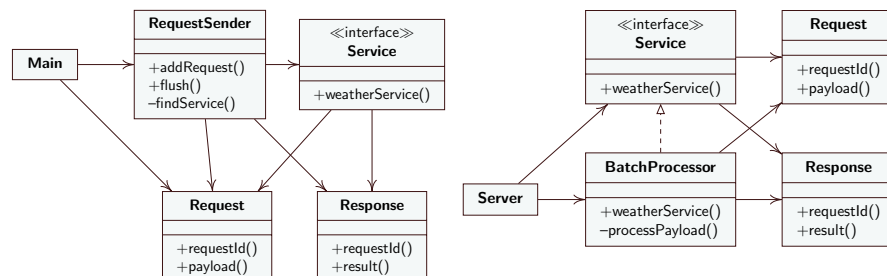
Conseguenze

- ▶ Benefici
 - ▶ Riduce il numero di messaggi e round-trip
 - ▶ Migliora throughput e utilizzo delle risorse
 - ▶ Diminuisce overhead di rete e serializzazione
- ▶ Compromessi
 - ▶ Introduce latenza di accumulo (le richieste aspettano di essere aggregate)
 - ▶ Maggiore complessità nella gestione delle risposte individuali
 - ▶ Rischio di fallimento del batch se una singola richiesta fallisce

Prof. Tramontana - Novembre 2025

Esempio

- ▶ Un servizio fornisce le previsioni del tempo di una città
- ▶ Con il pattern Request Batch le richieste al servizio sono passate come lista di oggetti Request e i risultati sono restituiti come lista di oggetti Response
- ▶ Lato client RequestSender accumula un certo numero di richieste e poi invia le richieste a Service



Prof. Tramontana - Novembre 2025