PDP 12-05

Channels

- Fa parte della libreria `java.nio`
- Lo possiamo aprire e chiudere
- Estende l'interfaccia `Closeable` (nesso per I/O)
  - `NetworkChannel`
  - `AsynchronousChannel`

```
ExecutorService pool = Executors.newFixedThreadPool(4);

AsynchronousChannelGroup group=
AsynchronousChannelGroup.withThreadPool(pool);

AsynchronousServerSocketChannel serverSocket =
AsynchronousServerSocketChannel.open() .bind(new
InetSocketAddress("127.0.0.1", GAME_PORT), 16); pool.submit(() →
{ serverSocket.accept( new GameAttachment(1, new Game(), serverSocket,
group), new AcceptPlayer0()); });

public void completed(Integer result, GameAttachment attachment) {
GameResult status = attachment.game.status(); AsynchronousSocketChannel
socket = attachment.players[status.next]; attachment.readBuf.clear();
socket.read(attachment.readBuf, attachment, new WriteStatus()); }
```

Fallacies and Frameworks

Fallacia: (s,f) l'essere fallace, ovvero che può trarre in inganno, insidioso.

- The network is reliable
  - Fallimento in qualsiasi tipo di nodo
- Latency is zero
  - Limite fisico di trasmissione dei dati
- Bandwidth is infinite
  - Banda disponibile dipendente dai dati da trasferire

- The network is secure
    - Se il contenuto del messaggio ha una qualche importanza o trasporta qualche informazione minimamente confidenziale, il protocollo deve essere reso sicuro/autenticato/autorizzato
- Topology doesn't change
    - Ubiquità della topologia
- There is one administrator
    - Diffusione e frammentazione dela rete
- Transport cost is zero
    - Costi energetici alti
- The network is homogeneous

Altre possibili fallacie:

- Versioning is simple
- Compensating updates always work
- Observability (software) is optional

Web Framework

- Vert.X = proposta per microservizi con API Rest
    - Introduction to Vert.x | Baeldung
    - modulare/scalabile/event-driven
- Most Popular Java Web Frameworks in 2023 | Rollbar

Perché usare un framework

L'approccio del framework Vert.X è tipico dei framework asincroni: ci viene fornito un ambiente all'interno del quale possiamo specificare gli handler che verranno richiamati in seguito a eventi definiti dal framework.

Le conseguenze sono:

- con un po' di disciplina, possiamo mantenere il codice che interagisce con il framework all'interno dell'handler
- gli handler compiono effetti collaterali, non sono quindi funzioni pure
- la struttura dichiarativa costringe ad esternalizzare l'organizzazione del codice
- è in carico a noi la gestione coerente dello stato condiviso

- la struttura dichiarativa costringe ad esternalizzare l'organizzazione del codice
- è in carico a noi la gestione coerente dello stato condiviso

Perché usarlo

Lo stesso framework web, cercherà di rendere trasparente, ed eventualmente configurabile:

- la gestione dei dettagli del protocollo
- la sicurezza nel trattamento della comunicazione
- la suddivisione delle risorse fra le varie parti del sistema

Perché non usarlo

Lo stesso framework web, potrebbe rendere oscuro, oppure inaspettatamente:

- fallire l'esecuzione di una richiesta a causa di un baco nella sua implementazione
- aprire una falla di sicurezza a causa di un default errato
- permettere il sovraccarico del sistema a causa di una mancata limitazione delle risorse

Stato distribuito

Abbiamo giustificato la necessità di realizzare un sistema distribuito con la ricerca di:

- affidabilità
- suddivisione del carico
- distribuzione dei risultati

Gestione dati distribuiti safe secondo raccomandazioni: [RFC 1034: Domain names - concepts and facilities (rfc-editor.org)]

Problemi di distribuzione:

- Ricerca del consenso globale (Byzantine General Problem)
- Utilizzo di algoritmi di consenso
    - Paxos
        - Leader/Votanti/Ascoltatori/Client/Proponenti
        - Raggiunta quorum su voti
        - Ridondanza con ascoltatori

        - Leader che controlla
    - Raft
        - https://thesecretlivesofdata.com/raft/

CAP Theorem

- C | Consistenza - Ogni lettura riceve o il valore più recente o un errore|
- A | Disponibilità - Ogni richiesta riceve una risposta valida (ma non necessariamente l'ultimo valore)
- P | Tolleranza alla separazione - Se fallisce un nodo, ci sono gli altri

N on puoi graantire tutte le proprietà tutte insieme (>=2 insieme)

Sia almeno consistente ultimo dato letto

Estensione - PACELC

- In caso di (P) Partizione, si deve scegliere fra (A) disponibilità e (C) consistenza
- (E) altrimenti, fra (L) latenza e (C) consistenza

Ci sono in letteratura due soluzioni a questo problema:

- Operational Transformation
    - ogni nodo produce delle modifiche al documento in corso di elaborazione
    - le modifiche sono propagate agli altri nodi
    - ogni nodo trasforma le modifiche ricevute in modo da applicarle al suo stato del documento
    - https://medium.com/coinmonks/operational-transformations-as-an-algorithm-for-automatic-conflict-resolution-3bf8920ea447
- *Conflict-Free Replicated Data-Types*
    - They are a family of replicated data types with a common set of properties that enable operations to always converge to a final state consistent among all replicas
    - To ensure that conflicts never happen (there is no concept of conflict resolution) and cause problems in your applications, operations on CRDT data types have to respect a specific set of algebraic properties
    - Introduction to Conflict-Free Replicated Data Types | Baeldung

CALM sta per "Consistency As Logical Monotonicity"

- I programmi rimangano logicamente corretti
- Trovato il deadlock nel sistema distribuito, rimane in piedi
- Shifting the definition of consistency to one of deterministic program outcomes rather than ordered histories of events