Verbale Esterno 2016-03-08

Informazioni sul documento	
Versione	1.0.0
Redazione	Marco Boseggia
Verifica	Andrea Giacomo Baldan
Approvazione	Giacomo Vanin
Uso	Interno
Lista di Distribuzione	ScalateKids
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin

Diario delle modifiche

Versione	Autore	Ruolo	Data	Descrizione
1.0.0	Giacomo Vanin	Responsabile	2016-03-08	Validazione documento
0.1.1	Andrea Giacomo Baldan	Verificatore	2016-03-08	Verifica documento
0.1.0	Marco Boseggia	Progettista	2016-03-08	Prima stesura documento
0.0.1	Marco Boseggia	Progettista	2016-03-08	Creazione scheletro del documento

Indice

1 Informazioni sulla riunione

- **Data**: 2016-03-08
- **Luogo**: Torre Archimede aula 1C150
- **Orario d'inizio**: 13:15
- **Durata**: 31'
- **Partecipanti interni**: *ScalateKids*
  - Andrea Giacomo Baldan
  - Marco Boseggia
  - Michael Munaro
  - Francesco Agostini
  - Giacomo Vanin
  - Davide Trevisan
- **Partecipanti esterni**:
  - Riccardo Cardin

2 Domande e risposte

In questo capitolo sono elencate le domande fatte dal gruppo *ScalateKids* in grassetto e le risposte del proponente *Riccardo Cardin* in corsivo.

2.1 Nella progettazione abbiamo incontrato dei problemi nell'identificare un design pattern architetturale adatto al nostro progetto.

Dovete individuare le componenti delle vostre applicazioni perché su ognuna di quelle dovrete andare ad individuare un pattern architetturale che al suo interno avrà più pattern architetturali. L'actor model$_G$ è già di per se un modello di comunicazione, già questo potrebbe essere visto come un qualcosa di simile a un pattern architetturale. Sicuramente un pattern architetturale dovrà essere individuato sulla console perché si tratta di un'interfaccia utente che si presta, quindi, all'MVC.

2.2 Per realizzare il DSL ci consiglia la creazione di una custom REPL Scala?

Vi consiglierei di ricercare qualche template per la creazione di interfacce via shell che magari utilizzino JVM$_G$.

2.3 Deve esserci effettivamente un meccanismo esplicito per creare scalabilità?

Sicuramente dev'essere scalabile e girare su un cluster. Dovrà essere possibile installarlo configurandolo come un nodo su un altro database. Dovrete valutare se sfruttare le configurazioni che offre Akka o costruire un'infrastruttura apposita.

2.4 Che tipo di connessione dovrà supportare il server (TCP, web socket, etc...)?

Lascio a voi la scelta della connessione tra un client e il server. All'interno del server, Akka userà il TCP.

2.5 Nel caso di estensione aggiungendo nodi dovrà essere previsto un meccanismo di load balancing$_G$?

Ci sono degli attori appositi (Manager$_G$) che verificano il carico su ogni Storekeeper$_G$ e suddividono il carico in maniera da evitare che uno Storekeeper$_G$ sia troppo carico. Gli algoritmi di load balancing$_G$

non sono semplici. Voi dovrete costruire un'architettura che sia il più estendibile possibile, vi consiglio di dare un algoritmo semplice di load balancing$_G$ in maniera che in futuro questo possa essere esteso facilmente. Per quanto riguarda il load balancing$_G$ tra più nodi il discorso si complica poiché dovreste fare in modo che un manager$_G$ mandi richieste di ricezione di dati se i propri Storekeeper$_G$ sono liberi.

2.6 Quanti attori main dovranno esserci nel sistema?

Avendo un solo attore main c'è il rischio che diventi un collo di bottiglia per sistemi con parecchi nodi avendo un solo punto di accesso al database e molti client. Vi consiglio di averne un numero che scala insieme alla dimensione del vostro sistema.

2.7 Avrebbe senso avere un main per ogni client?

In questo caso rischiate di avere un numero di main molto alto se il database viene usato da molti client. Un modo per avere un attore per ogni client potrebbe essere l'avere un main unico che crei un attore per ogni utente attivo. Il main unico creerà sempre un collo di bottiglia ma l'effetto sarà smorzato poiché deve fare solo l'operazione di creazione del nuovo attore per il client. Un'altra soluzione potrebbe essere quella di avere un numero di main configurabile e associare ogni client a uno di questi main secondo una politica ad esempio di round-robin$_G$.

2.8 Quante coppie chiave-valore ci consiglia di avere per ogni Storekeeper$_G$?

Il numero di coppie chiave-valore deve essere configurabile. Ogni Storefinder$_G$ deve avere degli indici per tenere traccia degli Storekeeper$_G$.

2.9 Quanto spesso dovrebbero scrivere su disco i Warehousemen$_G$?

Questo dipende da quale caratteristica del CAP theorem$_G$ volete sacrificare. Valutate anche la possibilità di lasciarla configurabile

2.10 Cosa succede nel caso in cui cada un main? Deve essere previsto un meccanismo di leader election$_G$?

Nel caso in cui cada un main il sistema deve crearne uno nuovo. Vi consiglio di cercare di sfruttare qualche escamotage di Akka$_G$ per risolvere questo problema.

2.11 La persistenza su disco in che formato va fatta?

Su questo vi lascio libertà di scelta, basta che sia configurabile estendendone la classe in modo che in futuro sia possibile effettuare una scelta.

2.12 Il DSL$_G$ a che linguaggio già esistente dovrebbe somigliare?

Per quanto riguarda le query$_G$ vi consiglio di seguire una sintassi simile a SQL$_G$.

3 Decisioni prese

In seguito alla riunione sono state prese le seguenti decisioni:

- Per la realizzazione della CLI$_G$ verranno cercati dei template o interfacce da implementare;
- Verrà implementato l'attore Manager$_G$ con un algoritmo semplicistico. Questo verrà sviluppato in modo da consentirne facilmente l'estendibilità nel caso in cui si voglia usare un algoritmo più efficace;
- Il numero di attori main$_G$ per nodo$_G$ sarà configurabile;
- L'attore main$_G$ da associare a un client verrà scelto secondo una politica di routing da definire;
- La persistenza su disco verrà implementata sfruttando il formato BSON (Binary javaScript Object Notation)$_G$;
- La persistenza su disco verrà implementata in modo tale da consentirne l'estendibilità facilmente nel caso in cui si voglia usare un formato di salvataggio differente;
- L'idea di creare un DSL (Domain Specific Language)$_G$ simile a Javascript$_G$ è stata scartata.