Backend

Anno Accademico 2010-2011

Federico Viscomi

6 luglio 2011

Indice

1 Introduzione **1**

2 Global table manager **1**

3 Local table manager **3**

4 Console gtm and ltm UI **3**

5 Global chat **3**

6 Single table manager and game logic **3**

7 Bot **3**

8 Database Manager **3**

9 Console player UI **3**

10 Changed communication, control flow and mutual exclusion **3**

1 Introduzione

Questo documento descrive quali sono state le varie iterazioni che hanno portato all'implementazione del backend e che alcuni dettagli di quanto e' stato sviluppato in ogni iterazione.

2 Global table manager

In questa iterazione si e' implementato il global table manager o gtm. Il gtm ha diverse responsabilita':

swarm : Gestire un insieme(chiamato in seguito swarm) di local table managers o ltm. In particolare gtm deve:

- Permettere di aggiungere e rimuovere ltm dallo swarm.
- Controllare la consistenza dello swarm. A questo scopo fa polling degli ltm chiamando

```
LocalTableManagerInterface.isAlive()
```

ogni

```
GlobalTableManagerInterface.POLLING_DELAY
```

millisecondi.

- Inoltrare le richieste delle servlet ad un ltm in modo da bilanciare il carico di lavoro tra gli ltm. A questo scopo gli ltm hanno associata una coppia di interi: il primo indica il numero di tavoli correntemente gestiti dall'ltm e il secondo indica il numero massimo di tavoli gestibili dall'ltm. Quando il gtm deve scegliere un ltm sul quale creare un nuovo tavolo, prende dallo swarm un ltm tra quelli che hanno un carico di lavoro piu' basso. Il carico di lavoro o workload di un ltm e' definito come il rapporto tra i tavoli gestiti e il numero massimo di tavoli gestibili.

Ricapitolando lo swarm deve mantere associazioni tra interfacce ltm e coppie di interi, deve fornire operazioni di:

- Aggiunta con chiave una interfaccia ltm.
- Rimozione con chiave una interfaccia ltm.
- Scelta di un ltm con workload minimo.

Si e' deciso di usare un ArrayList di record:

interfaccia ltm, numero di tavoli gestiti, numero di tavoli gestibili

Tale lista viene mantenuta ordinata con workload crescente. Questa struttura dati minimizza il tempo di esecuzione dell'operazione piu' frequente e cioe' la scelta di un ltm. Sia n il numero di ltm allora le complessita' in tempo al caso pessimo delle operazioni dello swarm sono:

aggiunta : O(log(n)) in particolare si tratta di una ricerca binaria.

rimozione O(n) perche' in questo caso non vengono confrontati i workload ma le interfacce.

scelta O(1) perche' in questo caso basta scegliere il primo elemento della lista.

tables : Mantenere alcune informazioni riguardo tutti i tavoli in Cupido. Tali informazioni sono tutte e sole quelle necessarie ad un giocatore che vuole unirsi ad una partita o che vuole vedere una partita e sono contenute nella classe

```
TableInfoForClient
```

Ogni tavolo nel gmt e' identificato da un descrittore di tavolo cioe' una istanza di

```
TableDescriptor
```

Il gtm deve memorizzare associazioni tra descrittori di tavolo e informazioni del tavolo, inoltre deve avere a disposizione le operazioni di: aggiunta, ricerca e rimozione usando come chiave un descrittore di tavolo. Si e' scelto di memorizzare le informazioni dei tavoli in una tabella hash. L'hash e' calcolato sui campi del descrittore del tavolo. In particolare il descrittore del tavolo contiene: una stringa che identifica un ltm, un intero che identifica il tavolo all'interno dell'ltm. In questo modo le operazioni richiedono tempo costante.

3 Local table manager

In questa iterazione si e' implementato il local table manager o ltm. Ltm si occupa di una porzione dei tavoli di Cupido. In particolare ltm ha le fornisce operazioni: creare componenti table chiamate in seguito single table manager o stm, rimuovere stm quando una partita finisce, cercare stm quando un giocatore vuole unirsi ad una partita o guardare una partita.

4 Console gtm and ltm UI

5 Global chat

6 Single table manager and game logic

7 Bot

8 Database Manager

9 Console player UI

10 Changed communication, control flow and mutual exclusion

major changes in control flow, mutual exclusion and communication: using action queue for: local bot, stm and automatic servlet. automatic servlet queue commands received from the UI, and execute them later if they can't be executed rigth now. AutomaticServlet enqueue all incoming messages in an action queue and process them sequentially. ViewersSwarm: all notifications to servlets are now enqueued in an action queue. STM: Use an action queue to avoid deadlocks. completely changed communication between stm, local bot and player ui.