RTI: analisi delle implementazioni

Gabriele D'Angelo (gdangelo@cs.unibo.it)

24 ottobre 2001

1 Introduzione

High Level Architecture (HLA) è un middleware simulativo distribuito; la componente software essenziale per il suo funzionamento è costituita dal Runtime Infrastructure (RTI).

L'obiettivo di questa relazione è svolgere una rapida carrellata sulle diverse implementazioni disponibili indicando le peculirità di ognuna di esse e cercando di evidenziare le varie potenzialità e limitazioni. In particolare verrà analizzata l'implementazione nata dal lavoro della Georgia Tech University sotto la guida del Prof. Richard Fujimoto.

2 Implementazioni

Escludendo le implementazioni del tutto sperimentali ed incomplete il panorama degli RTI si riduce a quattro elementi. La tabella seguente ha lo scopo di riassumere sinteticamente alcuni dati fondamentali.

Produttore	Licenza	Sorgenti	IEEE1516	Completezza
DMSO	non commerciale	no	futura	totale
Pitch	commerciale	no	futura	totale
GeorgiaTech	non commerciale	sì	ignota	51%
Mak	commerciale	no	ignota	parziale

E' importante considerare il problema della reperibilità, sia l'implementazione DMSO che quella GeorgiaTeach sono frutto di investimenti diretti o indiretti del Dipartimento della Difesa Usa (DoD), questo porta ad una serie di restrizioni per quanto riguarda l'uso e la diffusione di informazioni relative al lavoro.

L'implementazione Pitch è invece di origine europea e strettamente commerciale, di conseguenza non sottoposta ad alcun vincolo.

Per quanto riguarda la Mak pur essendo commerciale la politica sembra di libera diffusione ma limitata dalla richiesta di una licenza gratuita ma generabile solo dopo aver seguito una procedura di identificazione[1].

[1] Contattando la Mak risulta immediatamente evidente come questa politica sia destinata

La presenza di sole quattro implementazioni (e non tutte complete) deve far riflettere: HLA è uno stardard decisamente ampio e complesso, l'implementazione necessaria per il suo utilizzo ha costi iniziali piuttosto alti sia in termini di risorse che di conoscenze.

Per il futuro il DMSO si attende la realizzazione di un buon numero di ulteriori implementazioni private ma al momento non ci sono tracce che possano far supporre che questo avverrà realmente.

3 DMSO

L'implementazione DMSO è quella di "riferimento" nel mondo HLA, come detto finanziata dal DoD e per il momento attivamente sviluppata e supportata. In futuro è prevista la sua adesione allo standard IEEE 1516 e il conseguente blocco dello sviluppo con l'ingresso in fase di "bug-fix-only".

Dalle esperienze effettuate risulta perfettamente funzionante, completa, disponibile per una vasta serie di piattaforme e con il vantaggio di essere utilizzabile attraverso svariati linguaggi di programmazione (in particolare C e Java).

La critica maggiore è incentrata sulla "pesantezza" del software decisamente esigente in termini di risorse richieste all'esecuzione e sulla scarsità di informazioni riguardo la sua architettura implementativa.

4 Pitch

Questa versione di RTI è interamente scritta in Java con i conseguenti grossi vantaggi in termini di portabilità ma con l'incognita prestazionale. Anche in questo caso i linguaggi di programmazione utilizzabili sono Java e C ma in quest'ultimo caso solo dopo aver acquistato dei costosi binding.

L'architettura parzialmente centralizzata e l'implementazione Java-based costituiscono i principali dubbi. Per quanto riguarda l'adesione allo standard IEEE 1516 questa è prevista a brevissimo termine.

5 GeorgiaTech RTI

L'implementazione denominata anche Federated Simulations Development Kit (FDK) in realtà focalizza la propria attenzione più sulla creazione di un insieme di moduli che sulla realizzazione di una vera e propria RTI. I diversi moduli (MCAST, DDM-Kit, RM-Kit, TM-Kit ecc.) implementano varie funzionalità necessarie ad ogni implementazione di RTI: in questo modo l'RTI vera e propria assume l'aspetto di un ulteriore modulo facilmente intercambiabile ed espandibile.

ben preso a cambiare, presumibilmente dall'inizio del 2002 il prodotto diventerà completamente commerciale. Allo stesso modo la possibilità di accedere ai sorgenti, precedentemente lasciata aperta ora viene categoricamente esclusa.

Pragmaticamente nella distribuzione oltre le varie librerie sono presenti due RTI, la prima BasicRTI (BRTI) ha il solo scopo di verifica di funzionamento e risulta largamente incompleta; la seconda DetailedRTI (DRTI) è invece decisamente più interessante: implementa una quota pari al 51% dello standard (1.3 R6 interface specification) anche se in maniera non sempre del tutto coerente e funzionante.

Un approccio così spiccatamente modulare e la presenza dei sorgenti hanno come scopo dichiarato la possibilità di espandere l'attuale implementazione al fine di ottenere personalizzazioni e di ridurre i "costi d'accesso" a chi fosse maggiormente interessato alla verifica di algoritmi rispetto che alla vera e propria implementazione di Runtime. Allo stesso modo l'utilizzo del Kit per l'implementazione di simulazioni è scoraggiato da alcune limitazioni come ad esempio il limite di una singola federazione contemporanea attualmente imposto, o come la totale assenza di API per il passaggio di proprietà di attributi.

E' estremamente interessante notare come tutte le altre implementazioni di RTI condividano il funzionamento esclusivamente su rete TCP/IP based, nel caso del FDK questo *non* risulta assolutamente vero. Importanti ambiti di studio (e funzionamento) sono le architetture SharedMemory e le reti Myrinet: entrambi casi nei quali la latenza costituisce un aspetto fondamentale della ricerca.

Tornando all'ambito TCP/IP un settore di possibile ricerca verterebbe nella sostituzione del protocollo TCP attraverso una sua reimplementazione con minor overhead: il pattern tipico di comunicazione tra federati è costituito da un continuo scambio di informazioni; se questo pattern viene implementato attraverso una continua creazione di nuove connessioni lo scenario diviene decisamente *dispendioso.* Il protocollo TCP in questo caso non è decisamente adatto e porta a problemi soprattutto in termini di latenza. Un'implementazione *reliable* ma basata direttamente su IP o UDP (e quindi decisamente lightweight) potrebbe portare a risultati molto interessanti. Una prima verifica dei sorgenti mostra che il modulo preposto alla comunicazione di rete (FM) utilizza di default il protocollo TCP, sono presenti sorgenti che lavorano su comunicazioni UDP ma non è chiaro come questo sia utilizzabile ed eventualmente implementi quanto descritto sopra.

6 Mak

Fino al momento attuale non è ancora stato possibile valutare questa implementazione. Dalle specifiche tecniche appare decisamente interessante soprattutto in termini prestazionali. Purtroppo non sembrano presenti i binding per la sua programmazione con linguaggi diversi dal C. Al momento di scrivere ci troviamo nel bel mezzo dell'evoluzione di questa implementazione da liberamente scaricabile a prodotto commerciale: la non completa implementazione dello standard rappresenta un problema significativo soprattutto in assenza di informazioni sull'adeguamento a IEEE 1516.

7 Altre implementazioni

Nell'ambiente è noto lo sforzo fatto dalla Mitsubishi (divisione aerospaziale) per l'implementazione di una RTI completa e per la conseguente certificazione, lavoro realizzato in collaborazione con la MITRE. In qualche documento appare anche la volontà di commercializzare questo prodotto ma al momento di scrivere questo testo non sono disponibili informazioni aggiuntive.

8 Conclusioni

Come analizzato sinora è evidente che la situazione RTI non è del tutto rosea e come vi siano molti aspetti da colmare. A scopi di studio ovviamente la GeorgiaTech RTI è di estremo interesse soprattutto grazie alla possibilità di avere accesso ai sorgenti. Rimangono alcuni problemi come la mancanza di aderenza completa allo standard, solo parzialmente coperto e la necessità di programmare sia l'RTI che le simulazioni in linguaggio C.

Attraverso questa implementazione si riducono le barriere all'ingresso nel settore HLA fornendo agli interessati uno strumento già funzionante ma allo stesso tempo è necessaria una fase iniziale di apprendimento per familiarizzare con gli strumenti a disposizione che non sempre sono intuitivi anche per chi ha già avuto modo di lavorare su altre implementazioni dell'RTI.

9 Dettagli implementativi

9.1 Compilazione

Il FDK attualmente in uso è la versione 3.0 e risale al 14 febbraio 2001, una volta ottenuta l'autorizzazione il pacchetto a disposizione risulta formato da sorgenti e da un accettabile quantitativo di documentazione.

La compilazione dei sorgenti per piattaforma Intel/Linux risulta piuttosto facile ad esclusione dei problemi di compatibilità con il compilatore gcc: versioni recenti del compilatore (presumo >= 2.96) rifiutano la compilazione del file

FDK_3_0/FEDSIM/RTIKIT/SESSION/SRC/session.c

per problemi con la funzione *va_arg*. Un "palliativo" utile per portare a compimento la compilazione consiste nell'aggiungere la riga:

#define va_arg(ap,t) ((t *)(ap += sizeof(t)))[-1]

proprio all'inizio del file "session.c"; questa ridefinizione al momento non sembra creare ulteriori problemi.

9.2 Esecuzione

Gli script di esecuzione seppur comodi sono basati sulla presenza del "protocollo" rsh: per motivi di sicurezza questo protocollo è ormai stato rimosso praticamente in tutti gli ambienti di lavoro e questo costituisce uno svantaggio di non poco conto: un'eventuale riconversione degli script sarebbe decisamente utile.

Riferimenti bibliografici

[1] GeorgiaTech PADS Group: http://www.cc.gatech.edu/computing/pads/index.html

[2] FDK: http://www.cc.gatech.edu/~warp/fdk.html

[3] DMSO RTI: http://www.hla.dmso.mil

[4] PITCH.SE: http://www.pitch.se

[5] MAK: http://www.mak.com

[6] Ultima versione aggiornata di questo documento e altra documentazione su HLA: http://www.jpolis.com/gda/hla.html