POOL2
Un linguaggio parallelo orientato agli oggetti

prima parte

di Giuseppe Cardinale Ciccotti

I linguaggi di programmazione orientati agli oggetti si stanno affermando rapidamente come uno dei più moderni ed efficaci strumenti per l'implementazione di algoritmi complessi.
Contemporaneamente la necessità di garantire prestazioni adeguate induce all'utilizzo di architetture nuove. Il parallelismo offre la teorica possibilità di un significativo incremento delle performance dei sistemi di elaborazione. Per parallelismo intenderemo il concetto astratto ma sufficientemente generale, della possibilità di eseguire più segmenti di codice su dati diversi negli stessi «istanti» di tempo.
Non ci prenderemo cura dell'effettiva realizzazione di questo parallelismo ma analizzeremo i meccanismi formali che permettono di impostare un programma parallelo secondo le strutture definite nel linguaggio POOL2

Il progetto ESPRIT 415

POOL2 nasce nell'ambito del progetto ESPRIT 415 intitolato «Architetture Parallele e Linguaggi per il Processamento Avanzato delle Informazioni — Un approccio VLSI» che ha coinvolto numerose industrie ed i più prestigiosi atenei europei. Il progetto è organizzato in sei sottoprogetti ognuno con il compito di studiare uno specifico campo dell'elaborazione parallela. Sono stati oggetto di analisi la programmazione logica, funzionale, data-flow e orientata agli oggetti. Quest'ultimo ambito è stato affrontato nel sottoprogetto A del quale erano partner Philips e AEG (i progetti ESPRIT prevedono partner commerciali e subcontractor) e il Centrum voor Winskunde en Informatica di Amsterdam, la Technical University di Aachen, la Technical University di Berlino figuravano come subcontractor.

Nei laboratori di ricerca della Philips hanno visto la luce due prodotti complementari: un computer con architettura per l'elaborazione orientata agli oggetti dal nome DOOM «Decentralized Object-Oriented Machine» ed un linguaggio object-oriented, il POOL2.

Il linguaggio POOL2

POOL2 è l'acronimo di «Parallel Object-Oriented Language», come può essere ovvio intuire; meno ovvio può risultare comprendere gli obiettivi che i ricercatori hanno inteso raggiungere con il progetto di questo strumento di programmazione. Nonostante una decade di studi intensivi l'approccio alla programmazione parallela rimane ancora «artigianale» anche se sono ormai disponibili strumenti hardware e software alla portata di tutti, i Transputer e l'Occam tanto per citare un esempio noto ai lettori di MC.

Cionondimeno sono state raggiunte prestazioni elevatissime in campi come il calcolo numerico, mentre nel calcolo simbolico non si sono registrati grandi progressi. La difficoltà maggiore sembra risiedere proprio nella programmazione per l'esecuzione di tali applicazioni sulle macchine parallele. È necessario perciò fornire strumenti che permettano al programmatore di «mappare» nella maniera più semplice ed intuitiva possibile l'algoritmo sull'architettura parallela a disposizione.

È proprio quello che si sono proposti i progettisti del POOL2, arrivando perciò a definire la sintassi e la semantica di un linguaggio eseguibile sulla DOOM ma in generale indipendente dall'architettura ed efficace per una larga classe di macchine.

POOL2 prende le mosse dall'idea di programmazione orientata agli oggetti come implementata nello Smalltalk-80. Il parallelismo è integrato in questo modello realizzando ciascun oggetto con un processo parallelo indipendente in modo tale che gli oggetti siano delle entità attive invece che passive.

La programmazione orientata agli oggetti in POOL2

Senza alcuna pretesa di addentrarci in una descrizione accurata e formale, richiamiamo brevemente i concetti basilari della programmazione orientata agli oggetti per quanto serve ad illustrare l'implementazione di tali principi in POOL2. In tale approccio, un «sistema» è descritto come una collezione di oggetti. Un oggetto è definito a sua volta come una unità integrata di dati e di procedure che hanno effetto su quei dati. I dati in un oggetto sono mantenuti in variabili. I contenuti di una variabile possono essere cambiati eseguendo un'istruzione di assegnazione.

Un concetto fondamentale è che le variabili di un oggetto non sono acces-

sibili ad un altro oggetto, sono cioè strettamente private. L'unica maniera in cui un oggetto può interagire con un altro consiste nel mandargli un messaggio. L'oggetto originatore del messaggio richiede in tal modo che il ricevente esegua una procedura.

In POOL queste procedure, eseguite in risposta ad un messaggio, sono chiamate metodi. L'oggetto ricevente decide se e quando eseguire tale metodo, in qualche caso può anche decidere quale metodo eseguire. Più in generale l'originatore del messaggio fornirà qualche parametro da passare al metodo ed il metodo ritornerà un qualche risultato da passare indietro all'originatore (fig. 1).

In questo modo gli oggetti comunicano e possono cooperare; è fondamentale notare che le interazioni fra gli oggetti possono avvenire soltanto secondo questo schema predefinito di interfaccia.

Gli oggetti in POOL2 sono entità di natura dinamica; in qualsiasi punto del programma è possibile creare nuovi oggetti cosicché si può avere una proliferazione degli stessi. Il meccanismo di distruzione degli oggetti non è invece esplicitamente previsto ed essi possono essere rimossi con un'operazione di garbage collection, ma chiaramente questa scelta non influenza la corretta esecuzione del programma.

Gli oggetti vengono poi raggruppati in classi, in modo che tutti gli elementi (le istanze) di una certa classe condividano le caratteristiche generali della stessa. In particolare avranno stessi tipi e nomi per le variabili comuni alla classe, anche se ogni oggetto può poi averne di proprie ulteriori, ed eseguiranno lo stesso codice per i metodi. In tal modo una classe può servire da matrice per la creazione di nuove istanze.

Molti dei linguaggi di programmazione orientata agli oggetti implementano in modo diverso il meccanismo di creazione di questi. In generale si può affermare che la creazione di nuovi oggetti non è un compito «naturale» per le istanze della stessa classe, a chi spetterebbe infatti, il compito di creare la prima istanza? È più corretto, almeno da un punto di vista filosofico, che tale prerogativa spetti alla classe. Nello Smalltalk-80 le classi stesse sono considerate come oggetti; in questo modo esse possono essere create e cambiate dinamicamente. È diretto che il meccanismo di creazione degli oggetti sia definito nei metodi propri della classe: un nuovo oggetto sarà così creato mandando un messaggio appropriato alla classe a cui appartiene.

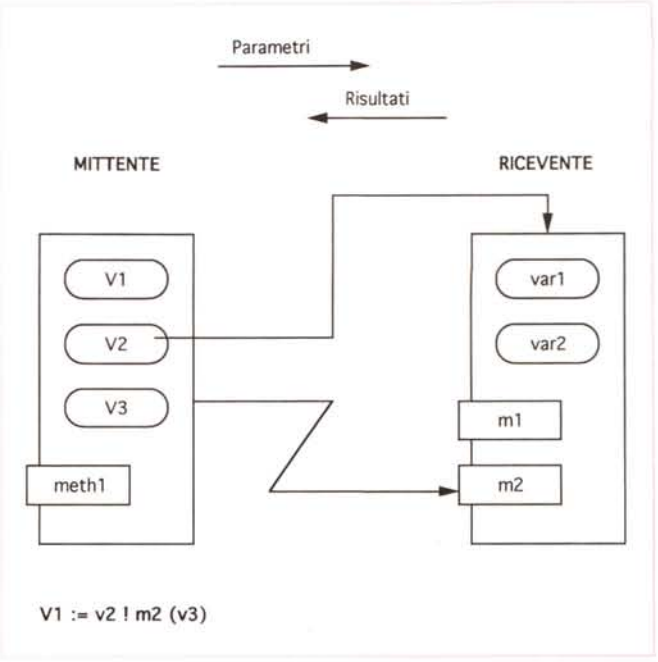

Figura 1 - Schema concettuale semplificato di messaggi tra oggetti. In evidenza la connessione tra gli oggetti ed il riferimento al metodo.

In POOL2 non è invece permesso che una classe possa cambiare o venire creata durante l'esecuzione del programma, quindi le classi e gli oggetti non potranno essere gestiti allo stesso modo almeno dal punto di vista della dinamicità; perciò la creazione di nuovi oggetti avviene attraverso un meccanismo detto routine, un tipo di procedura differente dal metodo, associato a ciascuna classe che può essere eseguita da ogni oggetto che la conosca.

I tipi di dati e la loro manipolazione

Analizziamo ora la natura dei dati che sono memorizzati negli oggetti. In generale, una variabile contiene un riferimento a qualche oggetto; i parametri e il risultato di un metodo sono anch'essi riferimenti ad oggetti. Spieghiamoci meglio: ogni tipo di dato è in sé un oggetto e quindi qualsiasi variabile è un riferimento ad un oggetto. Questo è quanto avviene nei linguaggi cosiddetti «puri» come lo Smalltalk-80 ed il POOL2 appunto; esistono per contro linguaggi come il C++ e l'Eiffel che hanno altri tipi più tradizionali come gli interi e i caratteri che possono essere manipolati dagli oggetti. Per tale motivi questi linguaggi sono detti «ibridi».

Nei linguaggi «puri» invece anche un semplice intero è concettualmente modellato come un oggetto e come tale va trattato. Per esempio, l'addizione 3+4 è ottenuta mandando all'oggetto 3 un messaggio riguardo al metodo add con un riferimento all'oggetto 4 come parametro. Pensate che il compilatore POOL2 ammette una sintassi del tipo 3+4 per tradurla poi nell'espressione 3 ! add (4).

In risposta a questo messaggio l'oggetto 3, sa come addizionarsi all'oggetto parametro 4 e ritorna come risultato un riferimento all'oggetto parametro 7. Chiaramente questa è soltanto una visione concettuale in quanto l'implementazione prenderà vantaggio delle capacità dell'hardware per ottimizzare operazioni come quella descritta.

I contributi della programmazione ad oggetti nello sviluppo del software

Uno dei maggiori vantaggi che si ottengono adottando la programmazione orientata agli oggetti, è quello di focalizzare l'attenzione del programmatore sull'utilizzo dei dati astraendone la loro costituzione. L'efficacia di raggruppare tutte le informazioni pertinenti ad un certo tipo di entità e di incapsulare queste informazioni secondo schemi di interfaccia espliciti con il «mondo» ester-

Bigliografia

P.H.M. America, J.J.M.M. Rutten, ***«A Parallel Object-Oriented Language: Design and Semantic Foundations»*** in ***«Languages for Parallel Architectures»*** a cura di J.W. de Bakker, Wiley, 1989.

C. Giustozzi, S. Polini, ***«OOP Object Oriented Programming - La programmazione degli anni '90»***, Technimedia, 1991.

no, è rilevante non appena si consideri il fatto che uno dei requisiti più importanti della programmazione parallela è quello di mantenere separati i domini dei dati dei singoli processi concorrenti. La soddisfazione di questo vincolo da sola permette di evitare gran parte delle situazioni di dead-lock del sistema.

Dal punto di vista dell'ergonomia della programmazione, il concentrarsi solo sull'utilizzo degli oggetti senza preoccuparsi della loro costituzione rispetto al resto del codice snellisce la fase di impostazione del programma parallelo.

La qualità del software è sicuramente migliorata sotto gli aspetti dell'adattabilità e della riusabilità, nodi tanto più importanti da sciogliere quanto più complesso è il sistema.

Le modifiche apportate al codice, essendo questo contenuto nei metodi di una classe, non hanno impatto su tutto il resto del programma a meno di non modificare l'interfaccia della classe stessa. È intuibile che il beneficio che si ottiene è proporzionale alla complessità del codice; molti lettori avranno avuto modo di provare il triste destino di chi debba accingersi a modificare del codice già scritto! L'altro aspetto rilevante è la riusabilità del codice, poiché una classe testata e sperimentata può essere facilmente riutilizzata in altri programmi. Ancora per il suo corretto uso, è necessario conoscere soltanto l'interfaccia esterna; i dettagli interni sono irrilevanti.

Questa considerazione è tanto più importante quando si pensi che avendo a disposizione un insieme di classi che forniscono delle funzionalità base (che corrispondono grossomodo alle procedure generalmente disponibili in qualsiasi ambiente di programmazione), non è raro che non si debbano affatto modificare i metodi delle classi medesime, ma soltanto arrangiare gli oggetti in modo da ottenere il compito voluto.

Il frazionamento del codice in metodi, consente inoltre di gestire al meglio sistemi complessi che sovente sono affidati ad un team di programmatori: una volta definita l'interfaccia le attività sono ben definite e delineate in modo tale che ogni programmatore possa lavorare in maniera indipendente dagli altri senza nessuna controindicazione sulla correttezza del programma complessivo.

Conclusioni

Terminiamo qui questa prima parte dedicata al POOL2; abbiamo dunque vagliato gli aspetti di questo linguaggio legati alla programmazione ad oggetti, richiamando alcuni concetti generali forse peraltro già noti ad alcuni lettori, da altre pagine di MC. Ci premeva comunque sottolineare come POOL2 sia un linguaggio «puro» e rispetti le condizioni più «strette» dell'object-oriented. Nel prossimo appuntamento consideremo l'altro aspetto pregnante di POOL2, vale a dire il parallelismo. Sarà interessante vedere poi come questo si sposa con la filosofia della programmazione orientata agli oggetti. Infine proporremo un semplice esempio di programmazione per convincere anche qualche S.Tommaso presente fra i lettori (sempreché ci sia) che forse un domani potrebbe trovarsi a programmare in POOL2 su un'architettura DOOM! MC

TOP DIVISION®

...una preoccupazione in meno

Distribuzione prodotti per l'informatica

Mother board, hard disk, streamer, cd-rom, r.a.m., schede hardware, coprocessori, stampanti, modem, monitor, fax, gruppi di continuità, data switch, mouse, cavi, accessori vari, schermi antiriflesso, contenitori, prodotti pulizia, tavoli, supporti magnetici, data cartridge, data display, lavagne luminose, nastri stampa originali, copertine.

3M SONY IBM Rivenditore Prodotti Ausiliari

NEC MANNESMANN TALLY

FUJITSU EPSON Bull

DISTRIBUTORE PER L'ITALIA DEI PRODOTTI «QUOTE MASTER» E «PHONIC»

42024 CASTELNOVO SOTTO (RE) - Tel. (0522) 682428-683963-688076 - Fax (0522) 682585