Progetto Conclusivo CCP (ver 1.0)

"maremare"

M. Danelutto

A.A. 2008–2009

Questo è il progetto conclusivo del corso di Complementi di Calcolo Parallelo, per l'anno accademico 2008–2009. Il progetto riguarda l'implementazione di un semplice prototipo di supporto per ambiente di programmazione paralella strutturata a skeleton. Scopo del progetto è sperimentare semplici soluzioni implementative, utilzzando le tecniche ed i concetti trattati durante lo svolgimento del corso.

1 maremare = MAp, REduce & MApREduce

Si vuole implementare un ambiente che metta a disposizione dell'utente finale tre skeleton: una map, una reduce e una map&reduce in un ambiente di programmazione basato su Java. Si suppone che sia definito un tipo `Bag<V>` che modella o una `Collection<V>` o un array di valori di tipo `V`. Si suppone che sia definita un'interfaccia `Function<A,B>` che modella funzioni da `A` in `B`. La semantica degli skeleton deve essere la seguente:

map la map prende due parametri: una `Bag<A>` e una funzione e applica la funzione a tutti gli elementi della `Bag`. Il risultato della map è ancora una `Bag`. Se la funzione ha tipo `Function<A,B>`[1] la `Bag` risultato sarà una `Bag<B>`

reduce la reduce prende due parametri: una `Bag<A>` e una funzione (assunta associativa e commutativa), e "somma" tutti gli elementi della `Bag` utilizzando la funzione. Il tipo della funzione deve essere `Function<A,A>`, ovviamente, ed il risultato della reduce su una `Bag<A>` sarà un dato di tipo `A`.

map&reduce la map&reduce prende tre parametri: una `Bag<A>`, una funzione di tipo `Function<A,B>` ed una funzione (associativa e commutativa) di tipo `Function<B,B>`. Restituisce un risultato di tipo `B` che costituisce il risultato dell'applicazione di una map della prima funzione seguita da una reduce della seconda.

I tipi dei nostri skeleton possono quindi essere riassunti dalla seguente tabella:

Skeleton	Tipo
map	`<Bag<A>→Function<A,B>→Bag<B>`
reduce	`<Bag<A>→Function<A,A>→A`
map&reduce	`<Bag<A>→Function<A,B>→Function<B,B>→B`

1.1 Architettura target

Il singolo programma a skeleton `maremare` deve essere in grado di girare o in locale (su una singola macchina Linux/Unix/BSD) o in remoto (su una rete di macchine Linux/Unix/BSD raggiugibili senza password mediate ssh ed scp) senza modifiche sintattiche. La configurazione su cui verrà

[1] cioè prende parametri di tipo `A` e restituisce risultati di tipo `B`

fatto girare il programma deve essere determinata dal (dai) file di configurazione letto durante la fase di inizializzazione dell'ambiente.

Si considerano solo macchine con sistema operativo Unix-like (quindi Linux o Mac OS X/BSD) e si deve prevedere l'utilizzo di Java 1.6.

Possono essere utilizzate per la realizzazione del supporto sia la libreria ProActive/GCM (limitatamente alla parte componenti, quindi non con gli active objects) che SCA/Tuscany, a scelta dello studente.

1.2 Scenari d'uso

Vogliamo implementare tre distinte possibilità di utilizzo della libreria:

Esecuzione in locale il programma `maremare` viene eseguito sulla macchina su cui viene lanciato. Utilizza uno o più JVM, inclusa la JVM utilizzata per il lanciatore

Esecuzione distribuita il programma `maremare` viene eseguito su un cluster di macchine, interconnesse direttamente (cioè senza firewall), i cui nomi sono presenti in uno dei file di configurazione. Ci si aspetta che sotto condizioni individuate dallo studente, il programmi scali linearmente con il numero di risorse impiegate nella computazione

Emulazione funzionale il programma `maremare` viene "emulato" nella JVM corrente, ovvero non vengono creati tutti i componenti necessari all'esecuzione locale o remota, ma si calcola semplicemente il *risultato funzionale* del programma stesso.

2 Implementazione

La maggior parte delle scelte implementative sono a carico dello studente, ma i vanno comunque rispettati i seguenti vincoli:

v.1 l'implementazione deve essere basata su componenti. Si può richiedere che i singoli costrutti strutturati siano essi stessi componenti. In tal caso, è opportuno mettere a disposizione dell'utente una factory che permetta di ottenere tali componenti da oggetti POJO.

v.2 il programma a skeleton deve poter girare su una architettura distribuita (cluster di workstation unix-like, con Java 1.6 interconnesse mediante una rete locale priva di firewall).

v.3 il supporto deve essere opportunamente strumentato in modo da poter facilmente ottenere statistiche e misure sui tempi di esecuzione (sia delle funzioni utente che del codice del supporto).

v.4 il supporto deve essere in grado di concludere senza errori l'esecuzione di un programma anche in presenza di uno o più fault (e.g. disconnessione, reboot) nelle macchine utilizzate per l'esecuzione distribuita (ad esclusione della macchina su cui gira il front-end, che si può considerare fault-free).

v.5 L'ingresso/uscita del programma `maremare` deve essere gestito mediante l'utilizzo di un `Iterator<Bag<T>>` che mette a disposizione i task da calcolare uno ad uno, e da un classe `Drain<R>` che mette a disposizione un metodo `void deliver(R res)` che "accetta" un risultato, ovvero lo stampa a video, o lo salva su un file, o ...

3 Modalità di svolgimento e consegna del progetto

Il progetto può essere svolto individualmente o in gruppo (max 3 persone). Si può utilizzare uno qualunque degli ambienti a componenti visti durante il corso (ProActive/GCM o SCA/Tuscany). Il progetto si considera consegnato quando verranno inviati al docente per posta elettronica:

1. tutti i sorgenti, insieme alle istruzioni/script di compilazione necessari per ottenere gli eseguibili, in un file (tar.gz o zip) che contenga anche i file necessari al funzionamento dell'ambiente di programmazione prescelto ed i programmi utilzzati per validare l'implementazione. Il docente deve essere in grado di scompattare il file consegnato, eseguire uno o più script di compilazione e, sempre mediante script o semplici comandi, di mandare in esecuzione i programmi di test

2. una relazione, di non più di 10 pagine, che includa:

   (a) il disegno architetturale di massima della libreria

   (b) un commento sulle principali scelte implementative adottate

   (c) le istruzioni precise per la compilazione della libreria e per l'utilizzo dei programmi di validazione

Dopo la consegna, lo studente (o gli studenti, in caso di consegna di gruppo) verranno convocati per discutere l'implementazione realizzata. La discussione verterà sugli aspetti tecnici dell'implementazione e sulle eventuali tecniche utilizzate. Al termine della discussone verrà verbalizzato l'esame.