Parallelismo e Haskell

Jacopo Francesco Zemella

Università degli studi di Milano

jacopofrancesco.zemella@studenti.unimi.it

13 dicembre 2016

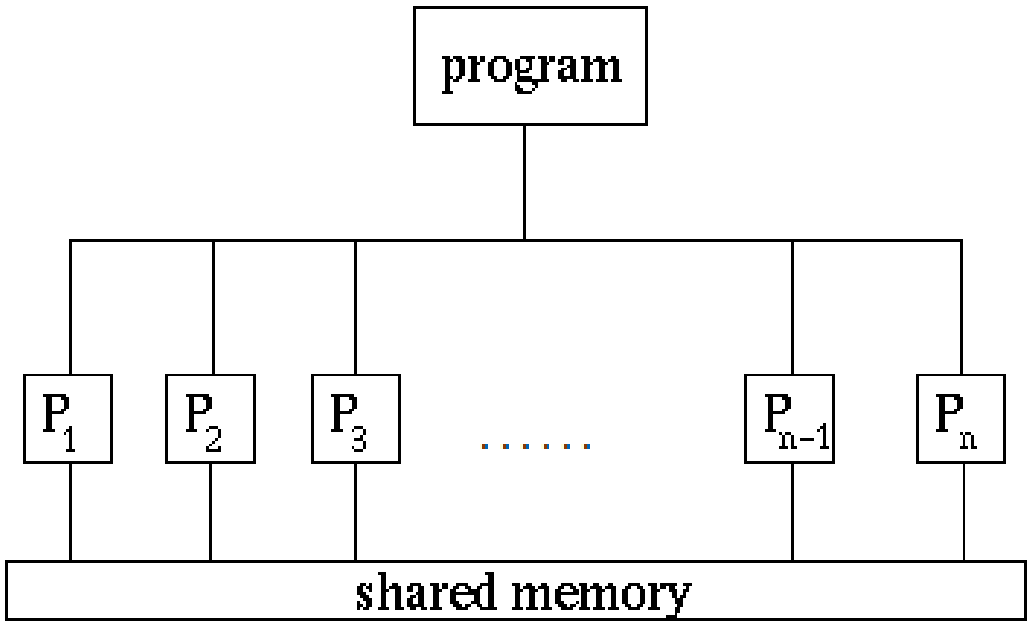
Algoritmi Sequenziali

Algoritmo: Sequenza finita di istruzioni interpretabili da un determinato agente, finalizzate a risolvere un problema

Calcolo Sequenziale: Le istruzioni vengono eseguite in *sequenza*, una dopo l'altra da una singola CPU: ad ogni istante di tempo è in esecuzione una e una sola operazione

Algoritmi Paralleli

Calcolo Parallelo: Le istruzioni vengono eseguite su più processori contemporaneamente



Legge di Amdahl - Limiti della Parallelizzazione

"Il miglioramento che si può ottenere su una certa parte del sistema è limitato dalla frazione di tempo in cui tale attività ha luogo"

Analogamente il guadagno prestazionale ottenuto dal parallelismo è limitato dalla sua componente sequenziale intrinseca.

Haskell: Caratteristiche

- Linguaggio funzionale
- Tipizzazione statica
- Lazy evaluation
- Trasparenza referenziale
- Particolarmente indicato per la parallelizzazione
  - **GpH**: Glasgow parallel Haskell

Le sfide della Programmazione Parallela

Implementare un programma parallelo richiede di specificare:

- **computazione**: creare un algoritmo efficiente e corretto
- **coordinamento**: gestire l'esecuzione per ottenere un "buon" livello di parallelismo

In Parallel Haskell **computazione** e **coordinamento** sono strettamente separate.

Aspetti di coordinazione

Coordinare l'andamento parallelo dell'esecuzione prevede, tra le altre cose:

- partizionamento: quali threads creare e la quantità di lavoro che devono svolgere
- sincronizzazione tra i threads
- load-store management
- comunicazione

Glasgow Parallel Haskell

- GpH è un'estensione **conservativa** di Haskell che permette di realizzare parallelismo **semi-esplicito** basato su threads.
- Solo alcuni aspetti chiave devono essere specificati dal programmatore
- Fornisce una sola primitiva per creare un thread, ma una volta creati essi sono gestiti autonomamente da un sofisticato sistema runtime.

Parallelismo in GpH

GpH mette a disposizione varie metodologie di programmazione parallela:

- Parallelismo semi-esplicito: funzioni *par* e *pseq*
- Monade Eval e Strategie di valutazione
- Dataflow Parallelism: Monade Par

Nella tesi abbiamo utilizzato le prime due, e in questa presentazione presentiamo solo la prima.

par e pseq

Le due funzioni hanno la seguente segnature:

```
par   :: a -> b -> b
pseq  :: a -> b -> b
```

- La funzione *par* segnala il primo argomento come un thread "parallelizzabile" (spark), ovvero che la sua valutazione in parallelo potrebbe essere conveniente
- Tuttavia, poiché la *lazy evaluation* di Haskell non impone un ordine di valutazione **stretto**, abbiamo bisogno anche di *pseq*, che assicura che il primo argomento venga valutato prima del secondo.

Un esempio in Parallel Haskell 1/2

Fattoriale *divide et impera*: Caso Sequenziale

```
fact n = fact' 1 n

fact' :: Integer -> Integer -> Integer
fact' m n
  | m == n    = m
  | otherwise = left * right
      where
          mid   = (m + n) `div` 2
          left  = fact' m mid
          right = fact' (mid+1) n
```

Un esempio in Parallel Haskell 2/2

Fattoriale: Caso Parallelo

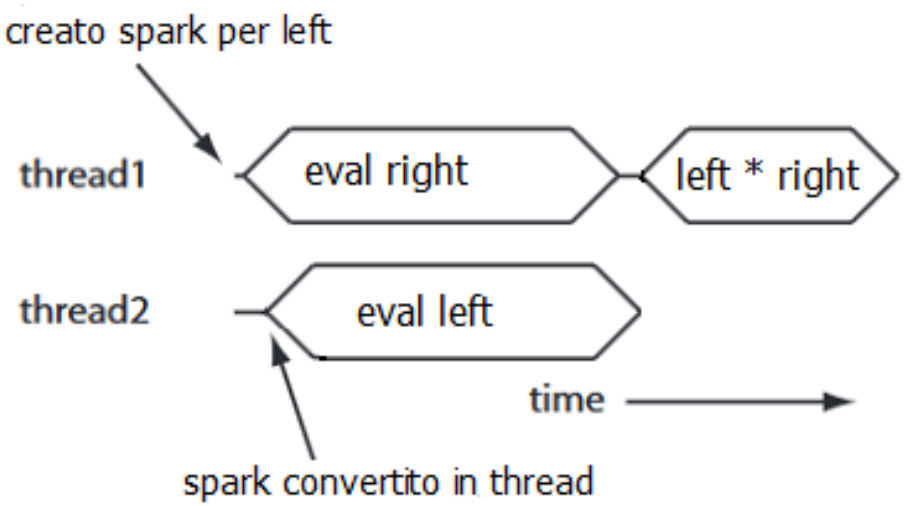
```
pfact n = pfact' 1 n

pfact' :: Integer -> Integer -> Integer
pfact' m n
  | m == n    = m
  | otherwise = left `par` right `pseq` (left * right)
      where
          mid   = (m + n) `div` 2
          left  = pfact' m mid
          right = pfact' (mid+1) n
```

Coordinamento 1/2

Sfruttando le due funzioni otteniamo il parallelismo desiderato.

```
left `par` right `pseq` (left * right)
```

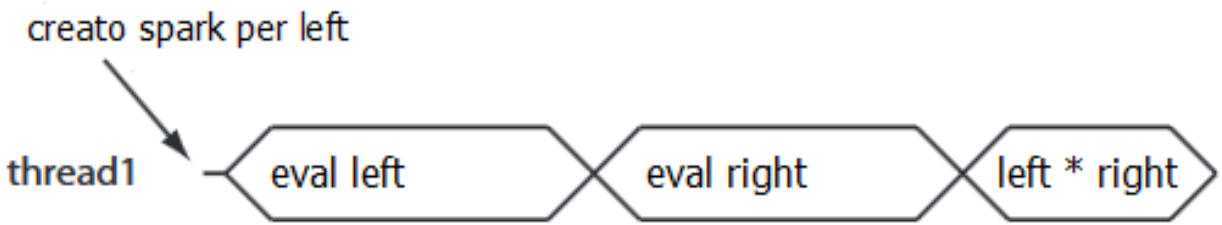


Coordinamento 2/2

Se avessimo scritto la funzione come segue:

```
| otherwise = left `par` (left * right)
```

la somma avrebbe potuto valutare *left* sul core principale, prima che qualsiasi altro avesse la possibilità di calcolarlo.



La *`pseq`* assicura che *left* e *right* siano calcolate in parallelo prima di moltiplicarle.

Problemi Implementati e Risolti nel Tirocinio

- Sorting:
  - quicksort
  - mergesort
- Algebra Lineare:
  - Somma, prodotto, potenza di Matrici
  - Calcolo del determinante
  - Inversione di una matrice
- Grafi:
  - Connettività

Conclusioni e Sviluppi Futuri

Haskell offre un ampio numero di implementazioni per la programmazione parallela mediante una semantica di comodo utilizzo.

Possibili sviluppi:

- Valutazione empirica del tempo di esecuzione degli esempi implementati.
- Confronto delle varie metodologie di programmazione.
- Analisi delle altre tecniche di parallelizzazione non affrontate nel tirocinio.

Grazie per l'attenzione!