2019

Programmazione Concorrente, Parallela e su Cloud

Metriche per il Calcolo Parallelo

Carmine Spagnuolo, Ph.D.

Definizioni

- *Algoritmo Sequenziale:* una sequenza precisa di passi per la soluzione di un problema, computata a partire da un un certo input ed eseguita da un singolo processore.
- Per risolvere il problema con più processori dobbiamo dividerlo in sotto problemi.
- Ogni sotto problema è risolto da un algoritmo separato.
- *Algoritmo Parallelo:* un insieme di algoritmi in esecuzione simultanea, la cui sincronizzazione e gestione dei risultati intermedi è definita dall'algoritmi parallelo.

Definizioni

- I criteri di valutazione di un algoritmo parallelo sono:
  - tempo di esecuzione;
  - e requisiti di memoria (quantità di risorse necessarie per eseguire).
- La valutazione di un algoritmo parallelo, oltre la dimensione dell'input, deve considerare anche il numero di processori.

Time complexity Definizione

- Worst-case parallel running time, anche noto come Time complexity, è il tempo di esecuzione nel caso pessimo (input meno favorevole).
- Consideriamo l'algoritmo R che risolve il problema Z di dimensione n.

$$T(p, n) = sup_{d \in D_n}\{t(p, d)\}$$

dove:

  - $t(p, d)$ è il numero di istruzioni eseguite per l'insieme di dati d fino alla terminazione dell'ultimo processore;
  - p è il numero di processori;
  - D_n è l'insieme dei dati in input d di taglia n;
  - *sup* function è l'ultimo massimo di un insieme di elementi.

Speedup Definizione

- $T^*(1, n)$ è il tempo di esecuzione del miglior algoritmo sequenziale per risolvere il problema Z.
- Lo speedup dell'algoritmo R è

$$S(p, n) = \frac{T^*(1, n)}{T(p, n)}$$

- il massimo speedup è p, $S(p, n) \leqslant p$.
- Il valore di speedup è minore di p dato che è condizionato dall'overhead di esecuzione di un algoritmo parallelo.
- in generale $T^*(1, n) \neq T(1, n)$
- $S(p, n) = \frac{T(1,n)}{T(p,n)}$ è chiamato speedup relativo.

Speedup Considerazioni

- $T^s(1, n)$ è il tempo di esecuzione della parte sequenziale del nostro algoritmo (o inherently sequential).
- $T^r(1, n)$ è il tempo di esecuzione della parte parallela del nostro algoritmo.
- Quindi $T(1, n) = T^s(1, n) + T^r(1, n)$.
- Ciò è necessario a causa del problema della *data dependencies*:

$$x = a + b$$

$$y = c * x$$

- Queste istruzioni devo essere eseguite in sequenziale.

Speedup Considerazioni

- Supponiamo ora che le operazioni $T^r(1,n)$ siano eseguite su p processori:

$$S(p,n) = \frac{T(1,n)}{T(p,n)} = \frac{T^s(1,n) + T^r(1,n)}{T^s(1,n) + T^r(1,n)/p + T^o(p,n)}$$

- $T^o(p,n)$ è l'overhead introdotto dall'algoritmo parallelo.
- L'idea è minimizzare il più possibile T^o al fine di non limitare lo speedup.
- Dobbiamo notare in $T^r(1,n)/p + T^o(p,n)$, il valore T^r decresce al crescere dei processori mentre T^o cresce al crescere dei processori.

Speedup Considerazioni

- Di conseguenza è possibile notare che il valore di speedup, al crescere del numero di processori p, tende ad incrementare fino ad un certo massimo e successivamente tende a decrescere.
- In molte circostanze il tempo T^r cresce più velocemente del tempo T^o al variare di n.
- Di conseguenza la taglia del problema è fondamentale per definire l'efficienza del nostro algoritmo.
- Per piccoli valori di n potrebbe non essere conveniente l'esecuzione parallela.

Costo Definizione

- Il costo di un algoritmo R è

$$C(p, n) = pT(p, n)$$

- il costo è il numero di operazioni eseguite dai processori.
- Il valore di costo minimo è uguale al tempo $T^*(1, n) = 1 \times T^*(1, n)$.
- ottenere il costo minimo è difficile dato che il tempo $T^*(1, n)$ non considera il costo di sincronizzazione e gestione di p processori, e significa eseguire su p processori solo le istruzioni dell'algoritmo sequenziale.

Efficienza Definizione

- L'efficienza di un algoritmo R è

$$E(p,n) = \frac{S(p,n)}{p} = \frac{T^*(1,n)}{pT(p,n}$$

- il massimo valore di efficienza è 1.
- Nel caso di efficienza massima, i nostri processori non saranno mai in stato idle, e il costo di comunicazione e sincronizzazione sarà pari a 0.

Efficienza Considerazioni

- In alcuni casi i valori $pT^*(p, n)$ e $T^*(1, n)$ sono tali

$$rT^*(1, n) \leqslant pT(p, n) \leqslant sT^*(1, n)$$

- per alcune costanti r e s, $1 \leqslant r \leqslant s$.
- In questi casi abbiamo che $pT(p, n) = \Theta(T^*(1, n))$.
- Tali algoritmi sono definiti cost-optimal: $E(p, n) = \Theta(1)$

Amdahl'sLaw

- Consideriamo un algoritmo sequenziale di running time $T(1, n)$ per taglia fissata n.
- Sia s l'insieme delle operazioni che possono essere eseguite solo in sequenziale.
- E sia r l'insime delle operazioni che possono essere eseguite in parallelo.
- Allora:

$$T^s(1, n) = sT(1, n)$$

$$T^r(1, n) = rT(1, n)$$

- dove $s + r = 1$

Amdahl's Law

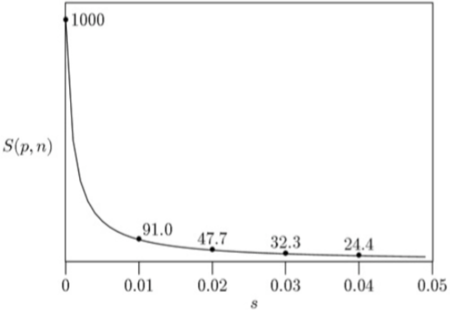
- Possiamo riportare le eguaglianze nella definizione di speedup, semplificando omettendo il valore $T^o(p,n)$:

$$S(p,n) \leqslant \frac{T^s(1,n) + T^r(1,n)}{T^s(1,n) + T^r(1,n)/p + T^o(p,n)}$$

$$\leqslant \frac{sT(1,n) + rT(1,n)}{sT(1,n) + rT(1,n)/p} = \frac{s+r}{s+r/p}$$

$$= \frac{1}{s+r/p} = \frac{1}{s+(1-s)/p}$$

Amdahl's Law



- Amdahl's Law definisce un upper bound allo speedup, considerando la frazione s di codice sequenziale, il numero di processori p e la dimensione dell'input.
- Utilizzando $s = 1\%$ del nostro programma possiamo raggiungere speedup 91 utilizzando 1000 processori.
- In oltre possiamo notare che

$$\lim_{p \to \infty} \frac{1}{s + (1-s)/p} = \frac{1}{s}$$

.

Amdahl's Law Considerazioni

- La legge di Amdahl permette definire l'idea di speedup per un algoritmo parallelo e un certo input n.
- Assume che le operazioni in $T^r(1, n)$ possano essere eseguite in parallelo su p processori.
- $T^o(p, n)$, ossia l'overhead di computazione sia trascurabile rispetto la dimensione di n e del numero di processori p (difficile da calcolare analiticamente dipende dall'implementazione).
- In generale questo significa che il massimo speedup ottenibile è funzione di quante operazioni riusciamo ad eseguire in parallelo e non dal numero di processori coinvolti nel calcolo.

References I

- Czech, Zbigniew J. (2017a). *Introduction to Parallel Computing*. Evaluation of Parallel Algorithms, Section 3.1. Cambridge University Press.
- — (2017b). *Introduction to Parallel Computing*. Amdahl's Law, Section 3.2. Cambridge University Press.