

# Efficienza

L'efficienza è un valore che indica quanto un programma parallelo è efficiente rispetto a quello seriale.

Si calcola con la seguente formula:

$$E = \frac{S}{P}$$

Dove:

- $E$  è l'efficienza
- $S$  è lo speedup
- $P$  è il numero di processori

L'efficienza è un valore compreso tra 0 e 1, dove 1 indica che l'efficienza è massima, cioè che il tempo di esecuzione è proporzionale al numero di processori.

# Speedup

Lo speedup è un valore che indica quanto un programma parallelo è più veloce rispetto a quello seriale.

$$S = \frac{T_s}{T_p}$$

Dove:

- $S$  è lo speedup
- $T_s$  è il tempo di esecuzione del programma seriale.
- $T_p$  è il tempo di esecuzione del programma parallelo.

1.  $T_s(n)$  # Tempo di esecuzione seriale
2.  $T_p(n, p)$  # Tempo di esecuzione parallelo

# Legge di Amhdal

La legge di Amdahl è un concetto che indica quanto un programma parallelo è efficiente rispetto a quello seriale, mantenendo costante l'efficienza all'aumentare del numero di processori.

$$E = \frac{1}{(1 - f) + \frac{f}{p}}$$

Dove:

- $E$  è l'efficienza
- $f$  è la frazione del programma che può essere parallelizzata
- $p$  è il numero di processori.

## Overhead

L'overhead è un valore che indica quanto un programma parallelo è meno efficiente rispetto a quello seriale.

Si calcola con la seguente formula:

$$T_o = \frac{T_p - T_s}{T_s}$$

Dove:

- $T_o$  è l'overhead
- $T_s$  è il tempo di esecuzione del programma seriale.
- $T_p$  è il tempo di esecuzione del programma parallelo.

Per mantenere l'efficienza costante ad  $x$ , la relazione tra  $n$  e  $p$  è:

$$n = x \cdot p \cdot \log p$$

Dove:

- $n$  è il tempo di esecuzione del programma seriale.
- $x$  è il tempo di esecuzione del programma parallelo.
- $p$  è il numero di processori.

## Esempio

Sia  $8 * p * \ln p$  l'overhead generato da un problema parallelo che gira su  $p$  processori e  $2 * n$  il suo tempo seriale  $T_s$ . Affinché l'efficienza venga mantenuta al 80% (cioè 0.8), trovare la relazione tra  $n$  e  $p$ .

Dati:

- $T_o = 8 * p * \ln p$  (overhead)
- $T_s = 2 * n$  (tempo seriale)
- $E = 0.8$  (efficienza)

Problema:

- Scrivere la relazione tra  $n$  e  $p$  affinché l'efficienza venga mantenuta al 80% (cioè 0.8).

Svolgimento:

Partendo dall'equazione dell'efficienza che si basa sul tempo seriale ( $T_s$ ) e sull'overhead ( $T_o$ ), possiamo scrivere la relazione tra  $n$  e  $p$  come segue:

$$0.8 = \frac{2n}{2n + 8p \cdot \ln p}$$

1. Moltiplichiamo entrambi i membri per il denominatore per eliminare la frazione:

$$0.8 \cdot (2n + 8p \cdot \ln p) = 2n$$

2. Espandiamo il membro sinistro:

$$1.6n + 6.4p \cdot \ln p = 2n$$

3. Sottraiamo  $1.6n$  da entrambi i membri:

$$6.4p \cdot \ln p = 0.4n$$

4. Dividiamo entrambi i membri per  $0.4n$ :

$$16p \cdot \ln p = n$$

Quindi, la relazione tra  $n$  e  $p$  affinché l'efficienza venga mantenuta al 80% (cioè 0.8) è:

$$n = 16p \cdot \ln p$$

## Isoefficienza

L'isoefficienza è un concetto che indica quanto un programma parallelo è efficiente rispetto a quello seriale, mantenendo costante l'efficienza all'aumentare del numero di processori.

$$w = T_s$$

$$T_p = \frac{w + T_o(w, p)}{p}$$

$$S = \frac{w}{T_p} = \frac{w}{\frac{w + T_o(w, p)}{p}} = \frac{p \cdot w}{w + T_o(w, p)} = \frac{p}{1 + \frac{T_o(w, p)}{w}}$$

Dove:

- $w$  è il tempo di esecuzione del programma seriale che prende il nome di *work*.
- $T_p$  è il tempo di esecuzione del programma parallelo.
- $T_o(w, p)$  è il tempo di esecuzione ottimizzato.
- $S$  è lo speedup
- $p$  è il numero di processori.

$$E = \frac{S}{p} = \frac{w \cdot p}{\frac{w + T_o(w,p)}{p}} = \frac{w \cdot p}{w + T_o(w,p)} \cdot \frac{1}{p} = \frac{w}{w + T_o(w,p)} = \frac{1}{1 + \frac{T_o(w,p)}{w}}$$

Dove:

- $E$  è l'efficienza
- $S$  è lo speedup
- $T_o(w,p)$  è il tempo di esecuzione ottimizzato.
- $w$  è il tempo di esecuzione del programma seriale che prende il nome di *work*.
- $p$  è il numero di processori.

L'efficienza rimane costante se rima costante il rapporto tra  $T_o(w,p)$  e  $w$ .

$$w = K \cdot T_o(w,p)$$

$$K = \frac{E}{1 - E}$$

Dove:

- $w$  è il tempo di esecuzione del programma seriale che prende il nome di *work*.
- $T_o(w,p)$  è il tempo di esecuzione ottimizzato.
- $E$  è l'efficienza
- $K$  è una costante

Se conosciamo l'overhead, esempio:

$$T_o = 8 \cdot p \log p$$

allora  $w$  è:

$$w = K \cdot 8 \cdot p \log p \Rightarrow O(p \log p)$$

Se aumenta il numero di processori da  $p_1$  a  $p_2$ , per mantenere l'efficienza costante, il tempo di esecuzione del programma seriale deve aumentare di un fattore logaritmico.

$$\frac{p_2 \log p_2}{p_1 \log p_1}$$

Se il rapporto è maggiore di 1, allora l'efficienza è costante.

## Esempio

Supponiamo di avere un programma seriale che impiega 100 secondi per eseguire un'operazione. Supponiamo inoltre che il tempo di esecuzione ottimizzato sia di 10 secondi e che l'overhead sia di 0.5.

**Calcoliamo l'efficienza:**

$$E = \frac{S}{P} = \frac{100}{10} = 10$$

**Calcoliamo l'overhead:**

$$O = \frac{T_p - T_s}{T_s} = \frac{10 - 100}{100} = -0.9$$

**Calcoliamo l'isoefficienza:**

$$w = T_s = 100$$

$$T_p = \frac{w + T_o(w, p)}{p} = \frac{100 + 10}{10} = 11$$

$$S = \frac{w}{T_p} = \frac{100}{11} = 9.09$$

$$E = \frac{S}{p} = \frac{w}{w + T_o(w, p)} = \frac{100}{100 + 10} = 0.9$$

## Ipercubo

L'iperubo è una topologia di rete che collega i processori in un'architettura parallela. È una rete di interconnessione che collega i processori in modo che ciascuno sia collegato a tutti gli altri. Questa topologia è utilizzata in sistemi paralleli e distribuiti, in quanto consente di minimizzare il tempo di comunicazione tra i processori.

## Numero di nodi

Il numero di nodi di un ipercubo è dato da:

$$p = 2^d$$

Dove:

- $p$  è il numero di nodi
- $d$  è la dimensione dell'iperubo.

## Esempio

Supponiamo di avere un ipercubo con 3 dimensioni. Il numero di nodi è dato da:

$$p = 2^3 = 8$$

## Distanza tra due nodi

La distanza tra due nodi in un ipercubo è definita come la distanza di Hamming tra le loro coordinate, che è il numero di bit in cui queste coordinate differiscono. In un ipercubo di  $d$  dimensioni, ogni nodo può essere rappresentato come un vettore di  $d$  bit. Se hai due nodi,  $A$  e  $B$ , con coordinate di bit  $a_1a_2\dots a_d$  e  $b_1b_2\dots b_d$ , allora la distanza  $D$  tra i nodi  $A$  e  $B$  è il numero di posizioni  $i$  in cui  $a_i$  è diverso da  $b_i$ . Formalmente:

$$D(A, B) = \sum_{i=1}^d |a_i - b_i|$$

Questo significa che conti il numero di bit che devi cambiare per convertire la rappresentazione binaria di  $A$  in quella di  $B$ . In termini più pratici, è il numero di cambiamenti da 0 a 1 o da 1 a 0 necessari per trasformare un vettore di bit nell'altro.

### Esempio

Supponiamo di avere un ipercubo con 3 dimensioni. La distanza tra i nodi  $A$  e  $B$  con coordinate  $101$  e  $111$  è data da:

$$D(A, B) = |1 - 1| + |0 - 1| + |1 - 1| = 1$$

## Calcolo del tempo di comunicazione tra due nodi

Dati i nodi  $x_1$  e  $x_2$ , possiamo calcolare il tempo di comunicazione tra questi due nodi di un ipercubo moltiplicando la distanza tra i nodi  $d$  per il tempo di comunicazione punto a punto  $T_w$ :

$$T = d \cdot T_w$$

Dove:

- $T$  è il tempo di comunicazione tra due nodi
- $d$  è la distanza tra due nodi
- $T_w$  è il tempo di comunicazione punto a punto.

### Esempio

Supponiamo di avere un ipercubo con 8 nodi e un tempo di comunicazione punto a punto di 10 secondi.

**Calcoliamo la distanza massima tra due nodi:**

$$d_M = \log p = \log 8 = 3$$

**Calcoliamo il tempo di comunicazione tra due nodi:**

$$T = d \cdot T_w = 3 \cdot 10 = 30$$

# Rete Omega

La rete Omega è una topologia di rete che collega i processori in un'architettura parallela. È una rete di interconnessione che collega i processori in modo che ciascuno sia collegato a tutti gli altri. Questa topologia è utilizzata in sistemi paralleli e distribuiti, in quanto consente di minimizzare il tempo di comunicazione tra i processori.