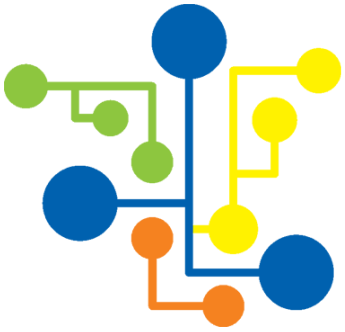

Data Science Academy

www.datascienceacademy.com.br

Programação Paralela em GPU

Métricas de Desempenho - Speedup

Os grandes benefícios que a computação paralela traz são apresentar um maior desempenho se comparado a uma execução sequencial e também poder resolver problemas mais complexos, de difícil solução. Mensurar o quanto pode ser mais rápido a execução de um determinado problema com a introdução de n processadores ou medir o quanto realmente o algoritmo foi eficiente são tarefas muito importantes quando se trabalha com paralelismo. Para isso, existem algumas métricas de desempenho que medem uma série de fatores gerados pelo paralelismo. Aqui veremos o Speedup.

Speedup

Speedup é uma métrica que mede o tempo que se leva para concluir uma tarefa em um único processador em relação ao tempo que se leva para completar a mesma tarefa com p processadores paralelos. A aceleração de um programa paralelo pode ser obtida por:

$S(p) = Tp(1) / Tp(p)$

onde S(p) é o speedup de p processadores paralelos, Tp(1) é o tempo de processamento em um único processador e Tp(p) é o tempo de execução em p processadores.

Alguns exemplos do cálculo do speedup podem ser ilustrados pela tabela abaixo.

	1 Proc.	2 Proc.	4 Proc.	8 Proc.	16 Proc.
Tempo (ms)	10.000	5.200	2.800	1.600	950
S(p)	1	1,92	3,57	6,25	10,52

Observa-se que através da introdução de novas unidades de processamento há uma diminuição do tempo de execução do algoritmo. Porém, como ocorre na grande maioria dos casos, o incremento de processadores faz com que a eficiência do speedup diminua. Esse fato ocorre principalmente devido a fatores como: sincronização, deadlock, balanceamento de carga e troca de informações.

Na prática é difícil obter um speedup linear uma vez que múltiplos processos ou threads, quase que invariavelmente apresentam alguma sobrecarga. Um exemplo disso são os programas que fazem uso de memória compartilhada. Esses programas tendem a ter seções críticas, o que exigirá a utilização de algum mecanismo de exclusão mútua, como por exemplo, o mutex. O uso de mutex faz com que uma região crítica seja executada de forma sequencial. Nesse sentido, o aumento do número de threads em programas com regiões críticas fará com que se aumente o processamento sequencial.

Outro ponto que pode ser avaliado são os programas que utilizam memória distribuída. Nessa situação ocorre a troca de informações entre os processadores através de rede, o que

normalmente é muito mais lento do que o acesso a memória local. O speedup tende a diminuir na medida em que se aumentam o número de processos. Com a adição de mais processos, provavelmente será necessário transmitir mais dados através da rede e o desempenho consequentemente diminuirá.

Não é comum, mas há casos onde a aceleração é superlinear, ou seja, S(p) > p. Nesses casos, o tempo de execução em paralelo através de p processadores é maior que p processadores. A razão para esse comportamento muitas vezes está no custo praticamente inexistente de comunicação/sincronização e na possibilidade de todos os dados do programa estarem na memória cache.