Sistemas Paralelos e Distribuídos

Práticas – Aula 6

Atividade

Métricas

- Speedup
- Eficiência
- Tempo de execução
- Tempo de comunicação
- Escalabilidade
- Exemplos

Speedup

Sumário

- Mede a proporção entre o tempo de execução total de um programa em sua versão sequencial, por sua versão paralela

$$\text{Speedup} = \frac{T_{\text{sequential}}}{T_{\text{parallel}}}$$

- Pode-se utilizar o número máximo (ou ótimo) de slots/threads disponíveis para calcular o speedup otimizado. Caso contrário, utilizar diferentes valores para analisar o impacto do aumento de capacidade versus o ganho.

- Os valores de speedup são usados para cálculo de eficiência.

Speedup

Sumário

- Exemplo

```
#!/bin/bash
BLUE='\033[0;34m'
YELLOW='\033[1;33m'
NC='\033[0m'

par=`./speedup-custom-openmp | awk -F" " {'print $5'}`
seq=`./speedup-custom | awk -F" " {'print $5'}`

echo -e "${YELLOW}The sequential time spent is $seq${NC}"
echo -e "${YELLOW}The parallel time spent is $par${NC}"
echo -e "${BLUE} ##########################################${NC}"
echo -e ${BLUE}"##########################################${NC}"
result=`echo "scale=5;$seq/$par" |bc`
echo -e "${YELLOW}The Speedup Seq/Par Time Raio is: $result ${NC}"
echo -e "${BLUE}##########################################${NC}"
echo -e "${BLUE}##########################################${NC}"
```

Eficiência

Sumário

- Mede a proporção entre o speedup e os processadores utilizados.

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

- A eficiência atinge 100% quando todos os processos/threads estão sendo utilizados de maneira ótima.

Exemplo:

“... Sequential sum: 49999994.123456, Time: 0.250000 seconds
Threads: 1, Time: 0.250000 seconds, Speedup: 1.000000, Efficiency: 100.000000%
Threads: 2, Time: 0.125000 seconds, Speedup: 2.000000, Efficiency: 100.000000%
Threads: 3, Time: 0.083333 seconds, Speedup: 3.000000, Efficiency: 100.000000%
Threads: 4, Time: 0.062500 seconds, Speedup: 4.000000, Efficiency: 100.000000%
Threads: 5, Time: 0.062500 seconds, Speedup: 4.000000, Efficiency: 80.000000%
Threads: 6, Time: 0.062500 seconds, Speedup: 4.000000, Efficiency: 66.666667%
Threads: 7, Time: 0.062500 seconds, Speedup: 4.000000, Efficiency: 57.142857%
Threads: 8, Time: 0.062500 seconds, Speedup: 4.000000, Efficiency: 50.000000%...”

Eficiência

Exemplo

```
#!/bin/bash
GREEN='\033[0;32m'
NC='\033[0m'

seq=`./efficiency-custom | awk -F" " {'print $5'}`

echo -e "${GREEN}##########################################${NC}"
echo -e ${GREEN}"##########################################${NC}"
for i in {2,4,6,8,10,12}
do
 export OMP_NUM_THREADS=$i
 par=`./efficiency-custom-omp | awk -F" " {'print $5'}`
 result_speed=`echo "scale=5;$seq/$par" |bc`
 result_efficiency=`echo "scale=5;$result_speed/$i*100"|bc`
 echo "Using $i threads the efficiency result is $result_efficiency %"
done
echo -e "${GREEN} ##########################################${NC}"
echo -e ${GREEN}"##########################################${NC}"
```

Tempo de execução

Sumário

- Mede quanto um processo demorar a ser executado.

- Fora do código (unix time) ou embutido(clock_gettime).

- Inúmeras ressalvas:
    - precisão das medidas
    - uniformidade de escalas
    - tempo de sistema (I/O, wait, interrupção,...)
    - alta influência da carga do sistema

- Opções
    - time (unix)
    - omp_get_wtime()
    - clock_gettime(CLOCK_MONOTONIC, &start);
    - MPI_Wtime()

Tempo de comunicação

Sumário

- Mede quanto um processo demora em suas trocas de mensagens.

- Comunicação envolve o procedimento de troca de dados (protocolo/API) + comunicações em rede (se houver).

- Pode-se ser difícil em determinados contextos. Por exemplo ao usar MPI Send/Receive.

- Usar funções das APIs ou então o clock_gettime.

- Usar o tempo de comunicação para calcular proporção de execução por tempo de comunicação.

Tempo de comunicação

Exemplo

```
#!/bin/bash

#sequential time
seq=`./shared-priv-no-barrier-time |grep time |awk -F" " {'print $4'}`
#communication time
comm=`./shared-priv-time |grep time |grep barrier | awk -F" " {'print $4'}`
#full execution time
par=`./shared-priv-time |grep time |grep -v barrier | awk -F" " {'print $4'}`

dif=`echo "scale=6;$par-$comm" |bc`
sum=`echo "scale=6;$dif+$seq" |bc`

echo -e "Sequential time is $seq"
echo -e "Full parallel time is $par"
echo -e "Communication time is $comm"

echo -e "Difference between parallel and communication time is $dif"
echo -e "Sum of sequential and communication time is $sum"
```

Escalabilidade

Sumário

- Analisa o comportamento e resposta do programa mediante um aumento gradativo de tamanho do problema e capacidade dos recursos de processamento.

- Weak Scaling: Processadores são incrementados consoante ao aumento de escala do problema.

- Strong Scaling: Processadores são incrementados enquanto o problema permanece na mesma escala.

- Escalabilidade é impactada por:
    - distribuição de carga
    - overhead
    - natureza serial de porções do código

Escalabilidade

Exemplo

```
#include <stdio.h>
#include <omp.h>
#include <stdlib.h>
#include <time.h>

#define SIZE 100000000

int main() {
    struct timespec inicio, fim;
    int max_threads = omp_get_max_threads();
    double* array = (double*)malloc(SIZE * sizeof(double));
    for (int i = 0; i < SIZE; i++) {
        array[i] = (double)rand() / RAND_MAX;
    }

    printf("Strong Scaling Size: %d)\n", SIZE);
    for (int p = 1; p <= max_threads; p++) {
        omp_set_num_threads(p);
        double sum = 0.0;
      clock_gettime(CLOCK_MONOTONIC, &inicio);
      #pragma omp parallel for reduction(+:sum)
      for (int i = 0; i < SIZE; i++) {
          sum += array[i];
      }
      clock_gettime(CLOCK_MONOTONIC, &fim);
      double percorrido = (fim.tv_sec - inicio.tv_sec) + (fim.tv_nsec - inicio.tv_nsec) / 1e9;
      printf("Threads: %d, Time: %f seconds, Sum: %f\n", p, percorrido, sum);
    }

    free(array);
    return 0;
}
```

FIM