Sistemas Paralelos e Distribuídos

Práticas – Aula 5

Atividade

Vamos escrever um programa que calcula a soma quadrática O(n2) da multiplicação de valores inteiros de 1 até n.

e.g.:

```
for (int i = 1; i <= n; i++)
{
        for (int j = 1; j <= n; j++)
        {
         sum += i * j;
        }
}
```

Atividade

Descrição

- Utilizemos OpenMP, MPI e uma forma sequencial para resolver o problema
- Utilizemos Makefile para melhor controle de versões
- Criar um programa com verificações e proteção de erros mínimas
- Explorar a escalabilidade
- Mensurar o tempo
- Automatizar e gerar resultados gráficos

Resolução

Sequencial

```
long long sequential_sum(int n) {
      long long sum = 0;
      for (int i = 1; i <= n; i++) {
          for (int j = 1; j <= n; j++) {
             sum += i * j;
          }
      }
      return sum;
  }
```

Resolução

MPI

```
long long mpi_sum(int n, int rank, int size) {
    long long local_sum = 0;
    long long global_sum = 0;

    int chunk_size = n / size;
    int remainder = n % size;

    int start_row = rank * chunk_size + (rank < remainder ? rank : remainder) + 1;
    int end_row = start_row + chunk_size + (rank < remainder ? 1 : 0) - 1;

    for (int i = start_row; i <= end_row; i++) {
        for (int j = 1; j <= n; j++) {
            local_sum += i * j;
        }
    }

    MPI_Reduce(&local_sum, &global_sum, 1, MPI_LONG_LONG, MPI_SUM, 0, MPI_COMM_WORLD);

    return global_sum;
}
```

Resolução

OpenMP

```
long long openmp_sum(int n) {
    long long sum = 0;
    #pragma omp parallel for reduction(+:sum)
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            sum += i * j;
        }
    }
    return sum;
}
```

Proteção e Qualidade

- Usar diretivas e condicionais de forma que o código possa ser mais concentrado.

```
#ifdef _OPENMP
  #include <omp.h>
  . . .
```

- Usar argumentos de entrada com devida validação

```
      if (argc > 1) {
          if ( argc > 2 )
          {
            printf("MSG: Please, provide a single integer input argument. This will be the size of
the mul  tiplications \n");
           exit(EXIT_FAILURE);
          }
```

- Usar comentários

- Cirar registos e logs para deburação e auditoria

Exploração e testes

- Fazer testes e investigações preliminares acerca do espaço utilizado, carga ao sistema, tempo de execução e dimensão dos dados de entrada.

```
n = atoi(argv[1]);
i ( n > 2000000)
{
  fprintf(stderr, "MSG: Provide an integer less than 2000000.\n");
  exit(EXIT_FAILURE);
}
```

Makefile

- Criar makefile que faz a instalação, remoção e compilação das 3 formas que nosso programa pode assumir (MPI, OpenMP e sequencial)

```
all:

openmp:
	$(CC) -D_OPENMP …. -fopenmp

mpi:
	/usr/bin/mpicc -DMPI_ENABLED …

sequential:
	$(CC) …

clean:
	rm -f ...

install:
	cp …
	cp …
	 cp …

uninstall:
	rm -f ….
```

Mensurar o tempo

- Implementar aferição de tempos externos e internos ao programa

for i in {1..30}; do time <programa>; done

--

clock_gettime(CLOCK_MONOTONIC, &inicio);

Automatização

- Criar script para automatizar a execução. Compilar, interpretar e formatar dados de saída.

```
steps=”100,1000,10000”
for i in $steps
do
    for j in {1..31}
    do
        /usr/local/bin/sum-sequential $i
    done | awk -F"Time" {'print $2'} | awk -F" " '{NR>1}{sum+=$2}END{printf "%.10f", sum/(NR-1)}' && echo ",$i"
done >> /tmp/sequential.csv
```

Gráficos

- Usar a ferramenta que lhe for mais convenient para plotar gráficos. Por norma, usamos arquivos csv como entrada para a ferramenta que vai construir o gráfico

```
. . .
df = pd.read_csv(filename)
if 'seqsec' not in df.columns or 'size' not in df.columns or 'secmpi' not in df.columns or 'secopenmp' not in df.columns:
    raise KeyError("CSV file must have columns named 'secseq', 'secmpi' and 'size'.")

plt.figure(figsize=(10, 6))
plt.plot(df['size'], df['seqsec'], marker='o', linestyle='-', label='Sequential')
plt.plot(df['size'], df['secmpi'], marker='o', linestyle='-', label='MPI')
. . .
```

Explorar outras métricas

- Verificar o uso de memória através do top

- Executar o programa com carga no sistema

- Verificar qual o ponto de inflexão entre performance de tempo de OpenMP e MPI

- Speedup $S(p) = \frac{T_1}{T_p}$

- Eficiência em termos de nro de processos e threads

- https://www.dcc.fc.up.pt/~ricroc/aulas/1516/cp/apontamentos/slides_metrics.pdf

FIM