Review for the Midterm Exam

1. Exam on Monday 21 October, at noon
   - paper-and-pencil or use-computer version
2. Sample Questions
   - scaled speedup
   - network topologies
   - task graph scheduling
   - compute bound or memory bound

MCS 572 Lecture 23
Introduction to Supercomputing
Jan Verschelde, 18 October 2024

Review for the Midterm Exam

1. Exam on Monday 21 October, at noon
   - paper-and-pencil or use-computer version

2. Sample Questions
   - scaled speedup
   - network topologies
   - task graph scheduling
   - compute bound or memory bound

paper-and-pencil or use-computer version

The exam starts on Monday 21 October, at noon, in two ways:

- as a paper-and-pencil exam, with open book and notes, but without computer experimentation; due by 12:50pm on the same day.

or

- as a use-computer version due Wednesday 23 October, at noon, which requires computer experimentation.

The decision to do either version can be postponed, till Monday 21 October, 12:49pm.

Not submitting the paper-and-pencil version defaults to the use-computer version.

Review for the Midterm Exam

1. Exam on Monday 21 October, at noon
   - paper-and-pencil or use-computer version

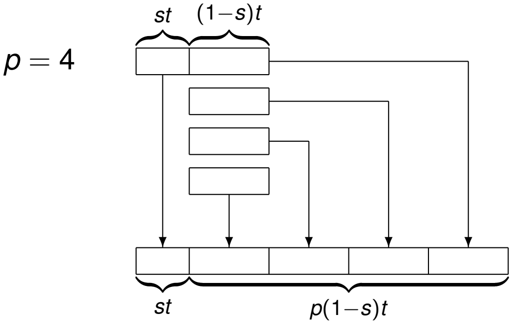
2. Sample Questions
   - scaled speedup
   - network topologies
   - task graph scheduling
   - compute bound or memory bound

scaled speedup

Benchmarking of a program running on a 12-processor machine shows that 5% of the operations are done sequentially,
i.e.: that 5% of the time only one single processor is working while the rest is idle.

Compute the scaled speedup.

solution



Scaled speedup $S_s(p) \leq \dfrac{st + p(1-s)t}{t} = s + p(1-s) = p + (1-p)s.$

Evaluate for $s = 0.05, p = 12$: $S_s(12) = 12 + (1 - 12)0.05 = 11.45.$

Review for the Midterm Exam

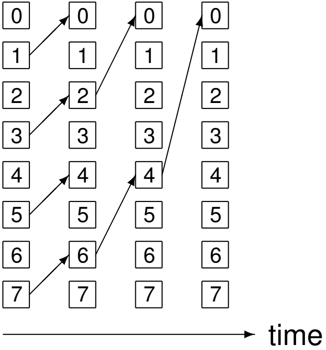
1. Exam on Monday 21 October, at noon
   - paper-and-pencil or use-computer version

2. Sample Questions
   - scaled speedup
   - network topologies
   - task graph scheduling
   - compute bound or memory bound

network topologies

Show that a hypercube network topology has enough connections for a fan-in gathering of results.

solution for $8 = 2^3$ nodes



Three steps:

1. 001 $\rightarrow$ 000; 011 $\rightarrow$ 010; 101 $\rightarrow$ 100; 111 $\rightarrow$ 110
2. 010 $\rightarrow$ 000; 110 $\rightarrow$ 100
3. 100 $\rightarrow$ 000

proof by induction

- The base case: we verified for 1, 2, 4, and 8 nodes.
- Assume we have enough connections for 2^k hypercube.

  Need to show: have enough connections for 2^{k+1} hypercube:

  1. In the first k steps:
     - ★ node 0 gathers from nodes $1, 2, \ldots 2^k - 1$;
     - ★ node 2^k gathers from nodes $2^k + 1, 2^k + 2, \ldots, 2^{k+1} - 1$.
  2. In step $k + 1$: node 2^k can send to node 0,

     because only one bit in 2^k is different from 0.

Review for the Midterm Exam

1. Exam on Monday 21 October, at noon
   - paper-and-pencil or use-computer version

2. Sample Questions
   - scaled speedup
   - network topologies
   - task graph scheduling
   - compute bound or memory bound

task graph scheduling

Given are two vectors **x** and **y**, both of length n, with $x_i \neq x_j$ for all $i \neq j$. Consider the code below:

```
for i from 2 to n do
    for j from i to n do
        y[j] = (y[i-1] - y[j])/(x[i-1] - x[j])
```

1. Define the task graph for a parallel computation of **y**.
2. Do a critical path analysis on the graph to determine the upper limit of the speedup.

tabulating the computations for $n = 4$

For $n = 4$, the numbers are in the table below:

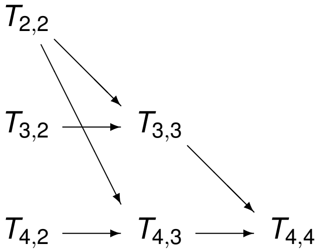
$$
\begin{array}{lll}
y_2 = \dfrac{y_1 - y_2}{x_1 - x_2} & & \\
y_3 = \dfrac{y_1 - y_3}{x_1 - x_3} & y_3 = \dfrac{y_2 - y_3}{x_2 - x_3} & \\
y_4 = \dfrac{y_1 - y_4}{x_1 - x_4} & y_4 = \dfrac{y_2 - y_4}{x_2 - x_4} & y_4 = \dfrac{y_3 - y_4}{x_3 - x_4}
\end{array}
$$

If the computations happen row by row, then there is no parallelism.

Observe that the elements in each column can be computed independently from each other.

the task graph for $n = 4$

Label the computation on row i and column j by $T_{i,j}$.



For $n = 4$, with 3 processors, it takes 3 steps to compute the table. The speedup is 6/3 = 2. Each path leading to $T_{4,4}$ has two edges or three nodes. So, the length of the critical path is 2.

For any n, with $n - 1$ processors, it takes $n - 1$ steps, leading to a speedup of $n(n-1)/2 \times 1/(n-1) = n/2$.

Review for the Midterm Exam

1. Exam on Monday 21 October, at noon
   - paper-and-pencil or use-computer version

2. Sample Questions
   - scaled speedup
   - network topologies
   - task graph scheduling
   - compute bound or memory bound

compute bound or memory bound

A kernel performs 36 floating-point operations and seven 32-bit global memory accesses per thread.

Consider two GPUs A and B, with the following properties:

- A has peak FLOPS of 200 GFLOPS and 100 GB/second as peak memory bandwidth;
- B has peak FLOPS of 300 GFLOPS and 250 GB/second as peak memory bandwidth.

For each GPU, is the kernel compute bound or memory bound?

the CGMA ratio

CGMA = Compute to Global Memory Access

- A kernel performs 36 floating-point operations and seven 32-bit global memory accesses per thread.
- GPU A has peak FLOPS of 200 GFLOPS and 100 GB/second as peak memory bandwidth.

The CGMA ratio of the kernel is $\frac{36}{7 \times 4} = \frac{36}{28} = \frac{9}{7} \frac{\text{operations}}{\text{byte}}$.

Taking the ratio of the peak performance and peak memory bandwidth of GPU A gives $200/100 = 2$ operations per byte.

As $9/7 < 2$, the kernel is memory bound on GPU A.

an alternative answer

- A kernel performs 36 floating-point operations and seven 32-bit global memory accesses per thread.
- GPU A has peak FLOPS of 200 GFLOPS and 100 GB/second as peak memory bandwidth.

Alternatively, it takes GPU A per thread

- $\dfrac{36}{200 \times 2^{30}}$ seconds for the operations and
- $\dfrac{28}{100 \times 2^{30}}$ seconds for the memory transfers.

As $0.18 < 0.28$, more time is spent on transfers than on operations.

On GPU A, the kernel is memory bound.

the kernel on GPU B

- A kernel performs 36 floating-point operations and seven 32-bit global memory accesses per thread.
- B has peak FLOPS of 300 GFLOPS and 250 GB/second as peak memory bandwidth.

The CGMA ratio of the kernel is $9/7$.

For GPU B, the ratio is $300/250 = 6/5$ operations per byte.

As $9/7 > 6/5$, the kernel is compute bound on GPU B.

an alternative answer

- A kernel performs 36 floating-point operations and seven 32-bit global memory accesses per thread.
- *B* has peak FLOPS of 300 GFLOPS and 250 GB/second as peak memory bandwidth.

Alternatively, it takes GPU *B* per thread

- $\dfrac{36}{300 \times 2^{30}}$ seconds for the operations and
- $\dfrac{28}{250 \times 2^{30}}$ seconds for the memory transfers.

As $0.12 > 0.112$, more time is spent on operations than on transfers.

On GPU *B*, the kernel is compute bound.