Parallel Program Performance Analysis

Chris Kauffman

CS 499: Spring 2016 GMU
Logistics

Today

- Final details of HW2 interviews
- HW2 timings
- HW2 Questions
- Parallel Performance Theory

Special Office Hours

- Mon 2/29 3:30-4:30
- Don’t wait until the last minute to start HW2

Reading: Grama Ch 5

- Performance Analysis
- Performance Metrics

Schedule

- Tue 2/23 PageRank & MPI
- Thu 2/25 Performance Analysis
- Mon 2/29 HW 2 Due 11:59pm
- Tue 3/1 Performance Analysis
- Thu 3/3 Guest Lecture, Mini-Exam 2
- 3/1-3/4 HW 2 Interviews
HW2 Interview Logistics

Will post a means to sign up for GTA interview time shortly.

Synopsis:

► 20-minute interview
► Demonstrate compiling and running a parallel program interactively on medusa
► Demonstrate submitting a parallel job on the batch queue with a certain number of processors
► Outline how the Problem 1: Heat program was parallelized
► Give a brief walk-through of code for Problem 1: Heat
► Explain some MPI calls as they appear in the Heat program
► Outline how the Problem 2: Pagerank program was parallelized
► Explain some MPI calls as they appear in the Pagerank program
► Describe timing results associated with parallel Pagerank runs with different numbers of processors and input sizes
► For groups of 2, interviewer may direct a question at individual group members to assess that both members understand the content
Specific Sample Interview Questions

- "Here you called MPI_XXX(...) in your Pagerank code. What is being accomplished there and why is it necessary?"
- "What kind of decomposition did you use for your parallel Heat code? What kind of communication did it require?"
- "Show me the timing results for running your Pagerank code on 4, 8, and 16 processors for the notredame-8000.txt graph."
- "Show me how you would run your parallel Heat program with 8 processors and width 64 interactively."
- "Submit a job to the batch queue which runs your parallel Heat program with 8 processors and width 64 and puts the output in testout.txt."
- "At the end of your Pagerank program, where are is the entire array of Pageranks stored? Show me where this happens in your code."

Other similar questions possible.
Draw **pictures** or show examples of the following collective communication operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>MPI Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcast</td>
<td>MPI_Bcast</td>
</tr>
<tr>
<td>Scatter</td>
<td>MPI_Scatter</td>
</tr>
<tr>
<td>Gather</td>
<td>MPI_Gather</td>
</tr>
<tr>
<td>All-Gather</td>
<td>MPI_Allgather</td>
</tr>
<tr>
<td>Reduce</td>
<td>MPI_Reduce</td>
</tr>
<tr>
<td>All-Reduce</td>
<td>MPI_Allreduce</td>
</tr>
</tbody>
</table>
PageRank Planning

- Overview Matrix-vector multiplication and parallel decomposition
- Discuss Vector Versions of Collective Comm Ops
- Spend more time planning/coding for PageRank
- Determine specifically which collective comm operations are needed at which parts of the program
Evaluating Parallel Algorithms

- Model problem: adding N numbers
- $T_s$: Serial execution time
- $T_p$: Parallel execution time

Parallel Metrics
- Speedup: $S = \frac{T_s}{T_p}$
- Efficiency: $E = \frac{S}{P}$
- Cost: $C = T_p \times P$

- Amount of work $W = \text{time for best serial algorithm to complete, akin to problem size}$

Amdahl’s Law

Speedup is limited by the portion of the program that can be parallelized and the degree to which that portion can be parallelized

- $W = W_{ser} + W_{par}$
- Supposing $W_{par}$ can be reduced to near 0 through parallelism
- Speedup $S = \frac{W}{W_{ser}}$: upper bound on speedup
- More refined versions of Amdahl’s law exist (see Wikipedia)
Exercise: Expensive Summing

- Adding $N$ numbers on $P = N$ processors can be done in $2 \ast \log_2 N$ steps. How?
- Standard serial algorithm takes $N$ steps to sum $N$ numbers.
- For 8, 32, and 1024 processors, calculate
  - Speedup: $S = T_s / T_p$
  - Efficiency: $E = S / P$
  - Cost: $C = T_p \ast P$
- What happens to efficiency as $N$ increases
- Give an analytic expression for the Efficiency of this algorithm for any $N$
- Is this algorithm worth the cost in terms of processors?
Exercise: Realistic Summing

- Adding $N$ numbers on $P < N$ processors can be done in $N/P + 2 \times \log_2 P$ steps. How?
- Standard serial algorithm takes $N$ steps to sum $N$ numbers.
- Fill in the following table
  - Speedup: $S = T_s / T_p$
  - Efficiency: $E = S / P$
  - Cost: $C = T_p \times P$

<table>
<thead>
<tr>
<th>P</th>
<th>N</th>
<th>Speedup</th>
<th>Efficiency</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>192</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>192</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>512</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- What happens to efficiency as $N$ increases?
- Give an analytic expression for the Efficiency of this algorithm for any $N$ and $P$.
- How fast does $N$ need to increase to maintain efficiency?
Table 5.1. Efficiency as a function of \( N \) and \( P \) for adding \( N \) numbers on \( P \) processing elements.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( p=1 )</th>
<th>( p=4 )</th>
<th>( p=8 )</th>
<th>( p=16 )</th>
<th>( p=32 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>1.0</td>
<td>0.80</td>
<td>0.57</td>
<td>0.33</td>
<td>0.17</td>
</tr>
<tr>
<td>192</td>
<td>1.0</td>
<td>0.92</td>
<td>0.80</td>
<td>0.60</td>
<td>0.38</td>
</tr>
<tr>
<td>320</td>
<td>1.0</td>
<td>0.95</td>
<td>0.87</td>
<td>0.71</td>
<td>0.50</td>
</tr>
<tr>
<td>512</td>
<td>1.0</td>
<td>0.97</td>
<td>0.91</td>
<td>0.80</td>
<td>0.62</td>
</tr>
</tbody>
</table>
Observations about Efficiency

- $E$ decreases as the number of processors increases while the work remains fixed.
- $E$ increases as the number of processors remains fixed while the work increases.
Isoefficiency and Parallel Overhead

- **Isoefficiency**: to go from $P$ processors to $P + K$ processors, amount that work needs to increase to keep efficiency constant; a function of processors and problem size

- Isoefficiency: $W = K \ T_o(W, P)$
  - $K = E/(1-E)$: constant based on target efficiency
  - $T_o(W, P)$: parallel overhead based on algorithm/system

- Smaller isoefficiency is better, indicates more processors can be added

- Parallel overhead: $T_o = P T_p - T_s$

- For adding $N$ numbers on $P$ processors
  
  $$T_o(N, P) = N - P \times \left(\frac{N}{P} + 2 \log_2(P)\right)$$

  $$T_o(N, P) = N - N + 2P \log_2(P)$$

  $$T_o(N, P) = 2P \log_2(P)$$

- For adding $N$ numbers on $P$ procs
  - Increase $P$ to $2 \times P$
  - Increase $N$ by $2 \times (2 \times P) \times \log_2(2 \times P)$
  - Stay at the same efficiency $E$