Lecture 17: Analytical Modeling of Parallel Programs: Scalability

CSCE 569 Parallel Computing

Department of Computer Science and Engineering
Yonghong Yan
yanyh@cse.sc.edu
http://cse.sc.edu/~yanyh
Topic Overview

• Introduction
• Performance Metrics for Parallel Systems
  – Execution Time, Overhead, Speedup, Efficiency, Cost
• Amdahl’s Law
• Scalability of Parallel Systems
  – Isoefficiency Metric of Scalability
• Minimum Execution Time and Minimum Cost-Optimal Execution Time
• Asymptotic Analysis of Parallel Programs
• Other Scalability Metrics
  – Scaled speedup, Serial fraction
Speedup and Efficiency

Efficiency of example parallel program

- Ideal speedup
- Speedup (Sp=T1/Tp)
- Efficiency (Sp/p)
Amdahl’s Law Speedup

\[ S(N) \leq \frac{1}{1-F} \]
Scalability of Parallel Systems

• Scalability: The patterns of speedup
  – How the performance of a parallel application changes as the number of processors is increased
  • Scaling: performance improves steadily
  • Not scaling: performance does not improve or becomes worse
Scalability of Parallel Systems

- Two different types of scaling with regards to the problem size
  - Strong Scaling
    - Total problem size stays the same as the number of processors increases
  - Weak Scaling
    - The problem size increases at the same rate as the number of processors, keeping the amount of work per processor the same

- Strong scaling is generally more useful and more difficult to achieve than weak scaling
Strong Scaling

\[ S(p) = \frac{T(1)}{T(p)} \]
\[ E(p) = \frac{S(p)}{p} \]

For ideal parallel speedup we get:

\[ T(p) = \frac{T(1)}{p} \]
\[ S(p) = \frac{T(1)}{T(p)} = p \]
\[ E(p) = \frac{S(p)}{p} = 1 \text{ or } 100\% \]
Weak Scaling of Parallel Systems

Extrapolate performance

• From small problems and small systems $\rightarrow$ larger problems on larger configurations

3 parallel algorithms for computing an n-point FFT on 64 PEs

Inferences from small datasets or small machines can be misleading
Scaling Characteristics of Parallel Programs

• Efficiency:
  \[ E = \frac{S}{p} = \frac{T_S}{pT_P} \]

• Parallel overhead: \( T_o = p \ T_P - T_S \)
  – Overhead increases as \( p \) increase

\[
E = \frac{1}{1 + \frac{T_o}{T_S}}
\]

• Problem size:
  – Given problem size, \( T_S \) remains constant

• Efficiency increases if
  – The problem size increases and
  – Keeping the number of PEs constant.
Example: Adding $n$ Numbers on $p$ PEs

- Addition = 1 time unit; communication = 1 time unit

\[ T_P = \frac{n}{p} + 2 \log p \]
\[ S = \frac{n}{\frac{n}{p} + 2 \log p} \]
\[ E = \frac{1}{1 + \frac{2p \log p}{n}} \]

Speedup tends to saturate and efficiency drops
Scaling and Efficiency

Fixed problem size (W)

- E vs. p
- fixed problem size
- # PEs increasing
- all parallel systems

Fixed number of processors (p)

- E vs. W
- problem size increasing
- # PEs fixed
- scalable parallel systems
Scaling Characteristics of Parallel Programs

• Overhead $T_o = f(T_s, p)$, i.e. problem size and $p$
  – In many cases, $T_o$ grows sublinearly with respect to $T_s$

• Efficiency:
  – Decreases as we increase $p \rightarrow T_o$
  – Increases as we increase problem size ($T_s$)

\[
E = \frac{1}{1 + \frac{T_o}{T_s}}
\]

• Keep efficiency constant
  – Increase problem sizes and
  – proportionally increasing the number of PEs

• *Scalable* parallel systems
Scalability vs Cost-Optimality

• To maintain constant efficiency $\Theta(1)$
  – Cost-optimal $\Leftarrow E = \Theta(1)$

• Any scalable parallel system can be made cost-optimal
  – Requires appropriate choice of
    • Size of the computation
    • Number of PEs
 Isoefficiency Metric of Scalability

Rate at which the problem size \( (T_s) \) must increase per additional PE \( (T_0) \) to keep the efficiency fixed

- The scalability of the system
  - The slower this rate, the better scalability
  - Rate == 0: strong scaling.
    - The same problem (same size) scales when increasing number of PEs

- To formalize this rate, we define
  - The problem size \( W \) = the asymptotic number of operations associated with the best serial algorithm to solve the problem.
  - The serial execution time, \( T_s \)
Isoefficiency Metric of Scalability

- Parallel overhead: $T_o(W,p)$

- Parallel execution time:

\[
T_P = \frac{W + T_o(W,p)}{P}
\]

- Speedup:

\[
S = \frac{W}{T_P} = \frac{WP}{W + T_o(W,p)}.
\]

- Efficiency

\[
E = \frac{S}{P} = \frac{W}{W + T_o(W,p)} = \frac{1}{1 + T_o(W,p)/W}.
\]
Isoefficiency Metric of Scalability

• To maintain constant efficiency (between 0 and 1)

\[ E = \frac{1}{1 + T_o(W, p)/W}, \]

\[ \frac{T_o(W, p)}{W} = \frac{1 - E}{E}, \]

\[ W = \frac{E}{1 - E} T_o(W, p). \]

• \( K = E / (1 - E) \) is a constant related to the desired efficiency

\[ W = KT_o(W, p). \]

Ratio \( T_o / W \) should be maintained at a constant value.
Isoefficiency Metric of Scalability

\[ W = KT_0(W, p). \]

\( W = \Phi (p) \) such that efficiency is constant

• \( W = \Phi (p) \) is called the **isoefficiency function**
  – Read as: what is the problem size when we have \( p \) PEs to maintain constant efficiency?
  – \( W_{p+1} - W_p = \Phi (p+1) - \Phi (p) \)
    • To maintain constant efficiency, how much to increase the problem size if adding one more PE?

• **isoefficiency function** determines the ease
  – With which a parallel system maintain a constant efficiency
  – Hence achieve speedups increasing in proportion to \# PEs
Isoefficiency Example 1

Adding $n$ numbers using $p$ PEs

- Parallel overhead: $T_o = 2p \log p$
- $W = KT_0(W,p)$, substitute $T_0$
  - $W = K \cdot 2 \cdot p \cdot \log p$
- $K \cdot 2 \cdot p \cdot \log p$ is the isoefficiency function

- The asymptotic isoefficiency function for this parallel system is $\Theta(p \cdot \log p)$
- To have the same efficiency on $p'$ processors as on $p$
  - Problem size $n$ must increase by $(p' \log p') / (p \log p)$ when increasing PEs from $p$ to $p'$
Examples

• by $(p' \log p') \div (p \log p)$

• If $p = 8$, $p' = 16$
  • $16 \log 16 \div (8 \log 8) = 16 \times 4 \div (8 \times 3) = 8 \div 3 = 2.67$

• $10M$ on 8 cores
• $10 \times 2.67M$ on 16 cores

• $A1 \times x + B1 \times y = C1 \rightarrow A2 \times x + A2 \times (B1 \div A1) \times y = A2 \times (C1 \div A1)$
  • $A2 \times x + B2 \times y = C2$
Cost-Optimality and Isoefficiency

• A parallel system is cost-optimal if and only if
  – Parallel cost == total work
  • Efficiency = 1

• From this, we have:
  – i.e. work dominates overhead

\[ W + T_o(W, p) = \Theta(W) \]
\[ T_o(W, p) = O(W) \]
\[ W = \Omega(T_o(W, p)) \]

• If we have an isoefficiency function \( f(p) \)
  – The relation \( W = \Omega(f(p)) \) must be satisfied to ensure the cost-optimality of a parallel system as it is scaled up
Topic Overview

• Introduction

• Performance Metrics for Parallel Systems
  – Execution Time, Overhead, Speedup, Efficiency, Cost

• Amdahl’s Law

• Scalability of Parallel Systems
  – Isoefficiency Metric of Scalability

• Minimum Execution Time

• Asymptotic Analysis of Parallel Programs

• Other Scalability Metrics
  – Scaled speedup, Serial fraction
Minimum Execution Time

• Often, we are interested in the minimum time to solution
• To determine the minimum exe time $T_P^{min}$ for a given $W$
  – Differentiating the expression for $T_P$ w.r.t. $p$ and equate it to 0
    $$\frac{d}{dp} T_P = 0$$
• If $p_0$ is the value of $p$ as determined by this equation
  – $T_P(p_0)$ is the minimum parallel time
Minimum Execution Time: Example

Adding n numbers

• Parallel execution time:

\[ T_P = \frac{n}{p} + 2 \log p. \]

• Compute the derivative:

\[ \frac{\partial}{\partial p} \left( \frac{n}{p} + 2 \log p \right) = -\frac{n}{p^2} + 2 \left( \frac{1}{p} \right) \]

• Set the derivative = 0, solve for p:

\[ -\frac{n}{p^2} + 2 \left( \frac{1}{p} \right) = 0 \]

\[ \frac{n}{p^2} + 2 \left( \frac{1}{p} \right) = 0 \]

• The corresponding exe time:

\[ T_P^{min} = 2 \log n. \]

Note that at this point, the formulation is not cost-optimal.
Topic Overview

• Introduction
• Performance Metrics for Parallel Systems
  – Execution Time, Overhead, Speedup, Efficiency, Cost
• Amdahl’s Law
• Scalability of Parallel Systems
  – Isoefficiency Metric of Scalability
• Minimum Execution Time
• Asymptotic Analysis of Parallel Programs
• Other Scalability Metrics
  – Scaled speedup, Serial fraction
Asymptotic Analysis of Parallel Programs

Sorting a list of $n$ numbers.

- The fastest serial programs: $\Theta(n \log n)$.
- Four parallel algorithms, A1, A2, A3, and A4

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>$n^2$</td>
<td>$\log n$</td>
<td>$n$</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>$T_P$</td>
<td>1</td>
<td>$n$</td>
<td>$\sqrt{n}$</td>
<td>$\sqrt{n \log n}$</td>
</tr>
<tr>
<td>$S$</td>
<td>$n \log n$</td>
<td>$\log n$</td>
<td>$\sqrt{n \log n}$</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>$E$</td>
<td>$\frac{\log n}{n}$</td>
<td>1</td>
<td>$\frac{\log n}{\sqrt{n}}$</td>
<td>1</td>
</tr>
<tr>
<td>$pT_P$</td>
<td>$n^2$</td>
<td>$n \log n$</td>
<td>$n^{1.5}$</td>
<td>$n \log n$</td>
</tr>
</tbody>
</table>
Asymptotic Analysis of Parallel Programs

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>$n^2$</td>
<td>$\log n$</td>
<td>$n$</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>$T_P$</td>
<td>1</td>
<td>$n$</td>
<td>$\sqrt{n}$</td>
<td>$\sqrt{n} \log n$</td>
</tr>
<tr>
<td>$S$</td>
<td>$n \log n$</td>
<td>$\log n$</td>
<td>$\sqrt{n} \log n$</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>$E$</td>
<td>$\frac{\log n}{n}$</td>
<td>1</td>
<td>$\frac{\log n}{\sqrt{n}}$</td>
<td>1</td>
</tr>
<tr>
<td>$pT_P$</td>
<td>$n^2$</td>
<td>$n \log n$</td>
<td>$n^{1.5}$</td>
<td>$n \log n$</td>
</tr>
</tbody>
</table>

- If metric is speed ($T_P$), algorithm A1 is the best, followed by A3, A4, and A2.
- In terms of efficiency ($E$), A2 and A4 are the best, followed by A3 and A1.
- In terms of cost ($pT_P$), algorithms A2 and A4 are cost optimal, A1 and A3 are not.
- It is important to identify the analysis objectives and to use appropriate metrics!
Topic Overview

• Introduction
• Performance Metrics for Parallel Systems
  – Execution Time, Overhead, Speedup, Efficiency, Cost
• Amdahl’s Law
• Scalability of Parallel Systems
  – Isoefficiency Metric of Scalability
• Minimum Execution Time
• Asymptotic Analysis of Parallel Programs
  🎨 Other Scalability Metrics
  – Scaled speedup, Serial fraction
Scaled Speedup: Example

$n \times n$ matrix multiplication

- The serial execution time: $t_c n^3$.
- The parallel execution time:
  \[ T_P = t_c \frac{n^3}{p} + t_s \log p + 2t_w \frac{n^2}{\sqrt{p}} \]

- Speedup:
  \[ S = \frac{t_c n^3}{t_c \frac{n^3}{p} + t_s \log p + 2t_w \frac{n^2}{\sqrt{p}}} \]
Consider memory-constrained scaled speedup.

- We have memory complexity \( m = \Theta(n^2) = \Theta(p) \), or \( n^2 = c \times p \).

- At this growth rate, scaled speedup \( S' \) is given by:

\[
S' = \frac{t_c (c \times p)^{1.5}}{t_c \frac{(c \times p)^{1.5}}{p} + t_s \log p + 2t_w \frac{c \times p}{\sqrt{p}}} = O(p)
\]

- Note that this is scalable.
Consider time-constrained scaled speedup.

- We have $T_P = O(1) = O(n^3 / p)$, or $n^3 = c \times p$.

- Time-constrained speedup $S''$ is given by:

$$S'' = \frac{t_c c \times p}{t_c \frac{c \times p}{p} + t_s \log p + 2 t_w \frac{(c \times p)^{2/3}}{\sqrt{p}}} = O(p^{5/6})$$

- Memory constrained scaling yields better performance.
• Adapted from slides “Principles of Parallel Algorithm Design” by Ananth Grama
