Performance Analysis in Parallel Programming

Ezio Bartocci, PhD
SUNY at Stony Brook

January 13, 2011
Today’s agenda

- Serial Program Performance
  - An example in the cable
  - I/O statements
  - Taking times

- Parallel Program Performance Analysis
  - The cost of communication
  - Amdahl's Law
  - Work and Overhead
  - Sources of Overhead
  - Scalability
Performance of a Serial Program

Definition of Running Time:

Let be $T(n)$ units the running time of a program.

The actual time depends on:

- the hardware being used
- the programming language and compiler
- details of the input other than its size

Example of Trapezoidal Rule:

$$h = \frac{(b-a)}{n};$$

$$\text{integral} = \frac{(f(a)+f(b))}{2.0};$$

$$x = a;$$

for (i=1; i <= n-1; i++){
  $$x = x + h;$$
  $$\text{integral} = \text{integral} + f(x);$$
}

$$\text{integral} = \text{integral} * h;$$

$$T(n) = c_1 + c_2(n-1)$$

$$T(n) \approx k_1 n + k_2$$

$$k_1 n \gg k_2 \quad \rightarrow \quad T(n) \approx k_1 n$$

$$\int_a^b f(x) \, dx \approx \frac{b-a}{N} \left[ \frac{f(a) + f(b)}{2} + \sum_{k=1}^{N-1} f \left( a + k \frac{b-a}{N} \right) \right]$$
What about the I/O?

In 33 Mhz 486 PC (linux) and gcc compiler
A multiplication takes a microsecond
printf about 300 microseconds

On faster systems, the ratios may be worse,
with the arithmetic time decreasing
I/O times remaining the same

Estimation of the time

\[ T(n) = T_{\text{calc}}(n) + T_{\text{i/o}}(n) \]
Parallel Program Performance Analysis

\[ T_\pi(n, p) \] denotes the runtime of the parallel solution with \( p \) processes.

**Speedup of a parallel program:**

\[ S_\pi(n, p) = \frac{T_\sigma(n)}{T_\pi(n, p)} \]

*Ambiguous definition*

Is \( T_\sigma(n) \) the fastest known serial program?  
Or \( T_\sigma(n) = T_\pi(n, 1) \)  
\[ 0 < S(n, p) \leq p \]
\[ S(n, p) = p \quad \text{linear speedup (very rare)} \]

**Efficiency of a parallel program:**

\[ E_\pi(n, p) = \frac{S(n, p)}{p} = \frac{T_\sigma(n)}{pT_\pi(n, p)} \]

Efficiency is a measure of process utilization in a parallel program, relative to the serial program.  
\[ 0 < E(n, p) \leq 1 \]
\[ E(n, p) = 1 \quad \text{linear speedup (very rare)} \]
\[ E(n, p) < 1/p \quad \text{slowdown} \]
For a reasonable estimatation of the performance of a parallel program, we should count also the cost of communication.

\[ T(n, p) = T_{\text{calc}}(n, p) + T_{\text{i/o}}(n, p) + T_{\text{comm}}(n, p) \]

While the cost of sending a single message containig \( k \) units of data will be:

\[ t_s + k \, t_c \]

\( t_s \) is called message **latency**

\[ \frac{1}{t_c} \] is called **bandwidth**
The Example: The Parallel Trapezoidal Rule

Serial:

\[
\begin{align*}
\text{h} &= \frac{(b-a)}{n}; \\
\text{integral} &= \frac{(f(a)+f(b))}{2.0}; \\
\text{x} &= a; \\
\text{for } (i=1; i <= n-1; i++)\{ \\
    \text{x} &= x + h; \\
    \text{integral} &= \text{integral} + f(x); \\
}\} \\
\text{integral} &= \text{integral} \times h;
\end{align*}
\]

\]

Serial:

\[
\begin{align*}
\text{h} &= \frac{(b-a)}{n}; \\
\text{integral} &= \frac{(f(a)+f(b))}{2.0}; \\
\text{x} &= a; \\
\text{for } (i=1; i <= n-1; i++)\{ \\
    \text{x} &= x + h; \\
    \text{integral} &= \text{integral} + f(x); \\
}\} \\
\text{integral} &= \text{integral} \times h;
\end{align*}
\]
Taking Timings
Amdahl's Law

$0 \leq r \leq 1$ is the fraction of the program that is perfectly parallelizable

$$S(p) = \frac{T_\sigma}{(1-r)T_\sigma + rT_\sigma / p} = \frac{1}{(1-r) + r / p}$$

$$\frac{dS}{dp} = \frac{r}{[(1-r)p + r]^2} \geq 0$$

$$\lim_{p \to \infty} S(p) = \frac{1}{1-r}$$

Example: if $r = 0.5$ the maximum speedup is 2
The amount of work done by a serial program is simply the runtime:

\[ W_\sigma(n) = T_\sigma(n) \]

The amount of work done by a parallel program is the sum of the Amounts of work done by each process:

\[ W_\pi(n, p) = \sum_{q=0}^{p-1} W_q(n, p) \]

Thus, an alternative definition of efficiency is:

\[ E(n, p) = \frac{T_\sigma(n)}{pT_\pi(n, p)} = \frac{W_\sigma(n)}{W_\pi(n, p)} \]
Overhead is the amount of work done by the parallel program that is not done by the serial program:

\[
T_o(n, p) = W_\pi(n, p) - W_o(n) = pT_\pi(n, p) - T_\sigma(n)
\]

**per-process overhead**: difference between the parallel runtime and the ideal parallel runtime that would be obtained with linear speedup:

\[
T'_o(n, p) = T_\pi(n, p) - T_\sigma(n)/p
\]

\[
T_o(n, p) = pT'_o(n, p)
\]

Main sources of Overhead: communication, idle time and extra computation
Scalability