Analysis of Distributed Algorithms
Example of parallel matrix times vector

Ned Nedialkov

McMaster University

2 February 2023

Outline

Message passing cost

A message consists of a header and the actual data.

The time for passing a message includes

- startup time, t_s
  - prepare message: header, trailer, error correction information
  - set up communication
- pre-hop time, t_h
  - time for a header to travel between two directly connected nodes (one link); also called node latency
- pre-word time, t_w: time for a word to traverse a link
  If bandwidth is r words/second $t_w = \frac{1}{r}$.

Simplified model. Cost for sending a message of m words is

$$t_{\text{comm}} = t_s + t_w m$$

Parallel matrix-vector product

- Consider parallel matrix-vector multiplication.
- Let $A \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^n$.
- We wish to compute $y = Ax$ in parallel and analyze scalability.
- We consider 1D and 2D distribution schemes.

1D distribution

- Assume that process i stores n_i consecutive rows of A and n_i consecutive elements of x:

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{p-1} \end{bmatrix} = \begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_{p-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{p-1} \end{bmatrix}$$

- Algorithm
  1. Process i gathers x
  2. Process i computes $y_i = A_i x$

Analysis

Step 1 can be done using all-to-all broadcast.

- Assume that all-to-all broadcast of m words takes time

$$t_s \log_2 p + t_w m(p-1) \tag{1}$$

  t_s is start-up time, t_w is pre-word transfer time
  (This is on hypercube. See e.g. V. Kumar, A. Grama, A. Gupta, G. Karypis. Introduction to parallel computing).

- Assume $n_i = n/p$. Then $m = n/p$, and (1) becomes

$$\begin{aligned} t_s \log_2 p + t_w m(p-1) &= t_s \log_2 p + t_w \frac{n}{p}(p-1) \\ &\approx t_s \log_2 p + t_w n \end{aligned} \tag{2}$$

Step 2 is done in $\approx (n/p)n = n^2/p$ operations.

Assuming $t_s \log_2 p + t_w n \ll t_w n$, the parallel time is

$$T_p \approx \frac{n^2}{p} + t_s \log_2 p + t_w n \approx \frac{n^2}{p} + t_w n.$$

Since the serial time is $T_1 \approx n^2$, the speed up is

$$S = \frac{T_1}{T_p} \approx \frac{n^2}{\frac{n^2}{p} + t_w n} = \frac{1}{\frac{1}{p} + t_w \frac{1}{n}}$$

The efficiency is

$$E_{\text{1D}} = \frac{S}{p} \approx \frac{1}{1 + t_w \frac{p}{n}} \tag{3}$$

Scalability

- For fixed n, E_{1D} decreases as p increases. Not strongly scalable.
- If $n \to 2n$, the work $\approx$ quadruples.
- Increase p to $4p$. Then

$$E'_{\text{1D}} \approx \frac{1}{1 + t_w \frac{4p}{2n}} = \frac{1}{1 + t_w \frac{2p}{n}} = \frac{1}{\underbrace{1 + t_w \frac{p}{n}}_{\text{same as in } E} + t_w \frac{p}{n}}$$

- Obviously $E'_{\text{1D}} < E_{\text{1D}}$.

- Let M be the number of words that can be stored per node.
- On p nodes, we can store pM words.
- Let N be the size of the largest problem we can store on p nodes.
- To store A, we store $N^2 = pM$ items; $N = \sqrt{pM}$
  We ignore the storage for the x_i.
- Using $n = N$ in (3), we obtain

$$E''_{\text{1D}} \approx \frac{1}{1 + t_w \frac{\sqrt{p}}{\sqrt{M}}} \tag{4}$$

- This algorithm does not scale well.

2D distribution

- Consider a 2D grid of processes.
- For simplicity, assume $q \times q = p$ process grid.
- Process (i, j) stores submatrix $A_{ij} \in \mathbb{R}^{n_i \times n_i}$.
- Process (j, j) stores subvector $x_j \in \mathbb{R}^{n_i}$.
- This distribution can be visualized as

$$\begin{matrix} A_{0,0}, x_0 & A_{0,1} & \dots & A_{0,q-1} \\ A_{1,0} & A_{1,1}, x_1 & \dots & A_{1,q-1} \\ \\ \vdots & \vdots & \dots & \vdots \\ A_{q-1,0} & A_{q-1,1} & \dots & A_{q-1,q-1}, x_{q-1}. \end{matrix}$$

Algorithm

1. Process (j, j) broadcasts x_j along column j

$$
\begin{array}{cccc}
A_{0,0}, x_0 & A_{0,1}, x_1 & \dots & A_{0,q-1}, x_{q-1} \\
A_{1,0}, x_0 & A_{1,1}, x_1 & \dots & A_{1,q-1}, x_{q-1} \\
\vdots & \vdots & & \\
A_{q-1,0}, x_0 & A_{q-1,1}, x_1 & \dots & A_{q-1,q-1}, x_{q-1}
\end{array}
$$

2. Process (i, j) computes $A_{ij}x_j$.
3. Process (i, i) does sum reduction across row i. Then (i, i) contains y_i:

$$y_i = A_{i,0}x_0 + A_{i,1}x_1 + \cdots + A_{i,q-1}x_{q-1}$$

Analysis

Let $n_i = n/q$

- Assume one-to-all broadcast of m words takes

$$(t_s + t_w m) \log_2 p \approx t_w m \log_2 p \tag{5}$$

  (See e.g. V. Kumar, A. Grama, A. Gupta, G. Karypis. Introduction to parallel computing).

- Here $m = n/q = n/\sqrt{p}$ and we broadcast along $q = \sqrt{p}$ nodes
  Then (5) becomes

$$t_w \frac{n}{q} \log_2 q = \frac{1}{2} t_w \frac{n}{\sqrt{p}} \log_2 p \tag{6}$$

- The number of operation for $A_{ij}x_j$ is

$$\approx \left(\frac{n}{q}\right)^2 = \frac{n^2}{p} \tag{7}$$

- All-to-one reduction is like one-to-all broadcast:

$$\approx \frac{1}{2} t_w \frac{n}{\sqrt{p}} \log_2 p \tag{8}$$

We ignore the time for summations.

- The parallel time is $(6) + (7) + (8)$:

$$T_p \approx \frac{n^2}{p} + t_w \frac{n}{\sqrt{p}} \log_2 p$$

Speedup is

$$S = \frac{T_1}{T_p} \approx \frac{n^2}{\frac{n^2}{p} + t_w \frac{n}{\sqrt{p}} \log_2 p} = \frac{1}{\frac{1}{p} + t_w \frac{\sqrt{p} \log_2 p}{pn}}$$

Efficiency is

$$E_{\mathsf{2D}} \approx \frac{1}{1 + t_w \frac{\sqrt{p} \log_2 p}{n}} \tag{9}$$

Before

$$E_{\mathsf{1D}} \approx \frac{1}{1 + t_w \frac{p}{n}}$$

- As before, assume $n = \sqrt{pM}$
  Then

$$E''_{\text{2D}} \approx \frac{1}{1 + t_w \frac{\sqrt{p}\log_2 p}{\sqrt{pM}}} = \frac{1}{1 + t_w \frac{\log_2 p}{\sqrt{M}}}$$

- $\log_2$ is a very slowly growing function, and can be considered $\approx$ constant here.
- As p increases, the efficiency decreases very slowly and much slower than

$$E''_{\text{1D}} = \frac{1}{1 + t_w \frac{\sqrt{p}}{\sqrt{M}}}.$$

- For practical purposes, this algorithm scales well