Parallel Programming

Prof. **Paolo Bientinesi**

pauldj@aices.rwth-aachen.de

WS 16/17

Dense Matrix-Vector Product: GEMV

1D matrix distribution

$$y := Ax, \qquad x, y \in \mathbb{R}^n, \quad A \in \mathbb{R}^{n \times n}$$

Dense Matrix-Vector Product: GEMV

1D matrix distribution

$$y := Ax, \qquad x, y \in \mathbb{R}^n, \quad A \in \mathbb{R}^{n \times n}$$

- A is partitioned along the rows, and distributed over p processes. Process i owns block A_i.

$$A = \left[\begin{array}{c} A_0 \\ \hline A_1 \\ \hline \vdots \\ \hline A_{p-1} \end{array}\right],$$

Dense Matrix-Vector Product: GEMV

1D matrix distribution

$$y := Ax, \qquad x, y \in \mathbb{R}^n, \quad A \in \mathbb{R}^{n \times n}$$

- A is partitioned along the rows, and distributed over p processes. Process i owns block A_i.
- x is partitioned in p blocks; process i owns block x_i.

$$A = \left[\begin{array}{c} A_0 \\ \hline A_1 \\ \hline \vdots \\ \hline A_{p-1} \end{array}\right], \quad x = \left(\begin{array}{c} x_0 \\ \hline x_1 \\ \hline \vdots \\ \hline x_{p-1} \end{array}\right),$$

Dense Matrix-Vector Product: GEMV

1D matrix distribution

$$y := Ax, \qquad x, y \in \mathbb{R}^n, \quad A \in \mathbb{R}^{n \times n}$$

- A is partitioned along the rows, and distributed over p processes. Process i owns block A_i.
- x is partitioned in p blocks; process i owns block x_i.
- **Goal:** Compute y, and distribute it the same way as x. *Rationale*: y overwrites x.

$$A = \left[\begin{array}{c} A_0 \\ \hline A_1 \\ \hline \vdots \\ \hline A_{p-1} \end{array}\right], \quad x = \left(\begin{array}{c} x_0 \\ \hline x_1 \\ \hline \vdots \\ \hline x_{p-1} \end{array}\right), \quad y = \left(\begin{array}{c} y_0 \\ \hline y_1 \\ \hline \vdots \\ \hline y_{p-1} \end{array}\right)$$

Dense Matrix-Vector Product: GEMV

1D matrix distribution

$$y := Ax, \qquad x, y \in \mathbb{R}^n, \quad A \in \mathbb{R}^{n \times n}$$

- A is partitioned along the rows, and distributed over p processes. Process i owns block A_i.
- x is partitioned in p blocks; process i owns block x_i.
- **Goal:** Compute y, and distribute it the same way as x. *Rationale*: y overwrites x.

$$A = \left[\begin{array}{c} A_0 \\ \hline A_1 \\ \hline \vdots \\ \hline A_{p-1} \end{array}\right], \quad x = \left(\begin{array}{c} x_0 \\ \hline x_1 \\ \hline \vdots \\ \hline x_{p-1} \end{array}\right), \quad y = \left(\begin{array}{c} y_0 \\ \hline y_1 \\ \hline \vdots \\ \hline y_{p-1} \end{array}\right)$$

Note: A_i, x_i and y_i indicate a *block* of rows, not a single row.

GEMV: Cost

1D matrix distribution

Algorithm

GEMV: Cost

1D matrix distribution

Algorithm

1. $x =$ `Allgather`(x_i) — x becomes available to every process
2. $y_i = A_i x$ — local computation

Parallel cost (lower bound for $T_p(n)$)

GEMV: Cost

1D matrix distribution

Algorithm

1. $x = \texttt{Allgather}(x_i)$ — x becomes available to every process
2. $y_i = A_i x$ — local computation

Parallel cost (lower bound for $T_p(n)$)

1. $\lceil \log_2(p) \rceil \alpha + \frac{p-1}{p} n\beta \approx \log_2(p)\alpha + n\beta$
2. $2\frac{n^2}{p}\gamma$

Sequential cost

GEMV: Cost

1D matrix distribution

Algorithm

1. $x = \texttt{Allgather}(x_i)$ — x becomes available to every process
2. $y_i = A_i x$ — local computation

Parallel cost (lower bound for $T_p(n)$)

1. $\lceil \log_2(p) \rceil \alpha + \frac{p-1}{p} n\beta \approx \log_2(p)\alpha + n\beta$
2. $2\frac{n^2}{p}\gamma$

Sequential cost

- $T_1(n) = 2n^2\gamma$

GEMV: Scalability

1D matrix distribution

Speedup

GEMV: Scalability

1D matrix distribution

Speedup

$$S_p(n) = \frac{T_1(n)}{T_p(n)} = \frac{2n^2\gamma}{\log_2(p)\alpha + n\beta + 2\frac{n^2}{p}\gamma}$$

Efficiency

GEMV: Scalability

1D matrix distribution

Speedup

$$S_p(n) = \frac{T_1(n)}{T_p(n)} = \frac{2n^2\gamma}{\log_2(p)\alpha + n\beta + 2\frac{n^2}{p}\gamma}$$

Efficiency

$$E_p(n) = \frac{S_p(n)}{p} = \frac{1}{1 + \frac{p\log_2(p)}{2n^2}\frac{\alpha}{\gamma} + \frac{p}{2n}\frac{\beta}{\gamma}}$$

Strong scalability

GEMV: Scalability

1D matrix distribution

Speedup

$$S_p(n) = \frac{T_1(n)}{T_p(n)} = \frac{2n^2\gamma}{\log_2(p)\alpha + n\beta + 2\frac{n^2}{p}\gamma}$$

Efficiency

$$E_p(n) = \frac{S_p(n)}{p} = \frac{1}{1 + \frac{p\log_2(p)}{2n^2}\frac{\alpha}{\gamma} + \frac{p}{2n}\frac{\beta}{\gamma}}$$

Strong scalability

$$\lim_{p\to\infty} E_p(n) = 0$$ ✖

Weak scalability

local memory = M $\Rightarrow$ combined memory = Mp
largest problem (n_M) solvable with p processes?
$n_M^2 = Mp$

GEMV: Scalability

1D matrix distribution

Speedup

$$S_p(n) = \frac{T_1(n)}{T_p(n)} = \frac{2n^2\gamma}{\log_2(p)\alpha + n\beta + 2\frac{n^2}{p}\gamma}$$

Efficiency

$$E_p(n) = \frac{S_p(n)}{p} = \frac{1}{1 + \frac{p\log_2(p)}{2n^2}\frac{\alpha}{\gamma} + \frac{p}{2n}\frac{\beta}{\gamma}}$$

Strong scalability

$$\lim_{p\to\infty} E_p(n) = 0$$ ✖

Weak scalability

local memory = M $\Rightarrow$ combined memory = Mp
largest problem (n_M) solvable with p processes?
$n_M^2 = Mp$

$$\lim_{p\to\infty} E_p(n_M) = \frac{1}{1 + \frac{\log_2(p)}{2M}\frac{\alpha}{\gamma} + \frac{\sqrt{p}}{2\sqrt{M}}\frac{\beta}{\gamma}} = 0$$ ✖

GEMV, distribution by columns

- A is partitioned along the columns, and distributed over p processes. Process i owns block A_i. Note that A_i now denotes a block of columns.
- x is partitioned as before; process i owns x_i.
- **Goal:** Compute y, and distribute it the same way as x.

$$A = \left[\begin{array}{c|c|c|c} A_0 & A_1 & \ldots & A_{p-1} \end{array} \right], \quad x = \left(\begin{array}{c} x_0 \\ \hline x_1 \\ \hline \vdots \\ \hline x_{p-1} \end{array} \right), \quad y = \left(\begin{array}{c} y_0 \\ \hline y_1 \\ \hline \vdots \\ \hline y_{p-1} \end{array} \right)$$

Cost & Scalability

Algorithm

Cost & Scalability

Algorithm

1. $y^{(i)} = A_i x_i$ — local computation
2. $y = \texttt{Reduce-Scatter}(y^{(i)})$ — reduction: sum of $y^{(i)}$'s + scatter

Parallel cost

Cost & Scalability

Algorithm

1. $y^{(i)} = A_i x_i$ — local computation
2. $y =$ `Reduce-Scatter`$(y^{(i)})$ — reduction: sum of $y^{(i)}$'s + scatter

Parallel cost

1. $2\frac{n^2}{p}\gamma$
2. $\lceil \log_2(p) \rceil \alpha + \frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma \;\approx\; \log_2(p)\alpha + n(\beta + \gamma)$

Analysis

Cost & Scalability

Algorithm

1. $y^{(i)} = A_i x_i$ — local computation
2. $y = \texttt{Reduce-Scatter}(y^{(i)})$ — reduction: sum of $y^{(i)}$'s + scatter

Parallel cost

1. $2\frac{n^2}{p}\gamma$
2. $\lceil \log_2(p) \rceil \alpha + \frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma \;\approx\; \log_2(p)\alpha + n(\beta + \gamma)$

Analysis

Compared to the previous case (A partitioned by rows), $T_p(n)$ now has one extra term ($n\gamma$); this algorithm is therefore also not scalable ✖

GEMV: 2D matrix distribution

- A is partitioned according to a 2D mesh of processes, with $p = r \times c$. P_{ij}, the (i, j) process in the mesh, owns block A_{ij}.

$$\begin{pmatrix} P_{00} & P_{01} & \cdots & P_{0,(c-1)} \\ P_{10} & P_{11} & \cdots & P_{1,(c-1)} \\ \vdots & \vdots & \ddots & \vdots \\ P_{(r-1),0} & P_{(r-1),1} & \cdots & P_{(r-1),(c-1)} \end{pmatrix} \leftarrow \begin{pmatrix} A_{00} & A_{01} & \cdots & A_{0,(c-1)} \\ A_{10} & A_{11} & \cdots & A_{1,(c-1)} \\ \vdots & \vdots & \ddots & \vdots \\ A_{(r-1),0} & A_{(r-1),1} & \cdots & A_{(r-1),(c-1)} \end{pmatrix}$$

- As before, x and y are partitioned in p blocks; x is mapped to the mesh by columns, y by rows.

Example: $p = 2 \times 3$

$$\left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \end{array}\right), \quad \left(\begin{array}{c|c|c} x_0 & x_2 & x_4 \\ \hline x_1 & x_3 & x_5 \end{array}\right), \quad \left(\begin{array}{c|c|c} y_0 & y_1 & y_2 \\ \hline y_3 & y_4 & y_5 \end{array}\right)$$

Cost

2D matrix distribution

Algorithm

Cost

2D matrix distribution

Algorithm

1. $x_I = \texttt{Allgather}(x_i)$ within columns — x_I is a “block”
2. $y_J = A_{ij} x_I$ — local computation
3. $y_j = \texttt{Reduce-scatter}(y_J)$ within rows

Parallel cost (lower bound for $T_p(n)$)

Cost

2D matrix distribution

Algorithm

1. $x_I = \texttt{Allgather}(x_i)$ within columns — x_I is a “block”
2. $y_J = A_{ij} x_I$ — local computation
3. $y_j = \texttt{Reduce-scatter}(y_J)$ within rows

Parallel cost (lower bound for $T_p(n)$)

1. $\lceil \log_2(r) \rceil \alpha + \frac{r-1}{p} n\beta \approx \log_2(r)\alpha + \frac{n}{c}\beta$
2. $2\frac{n^2}{p}\gamma$
3. $\lceil \log_2(c) \rceil \alpha + \frac{c-1}{p} n\beta + \frac{c-1}{p} n\gamma \approx \log_2(c)\alpha + \frac{n}{r}\beta + \frac{n}{r}\gamma$

Sequential cost

- $T_1(n) = 2n^2\gamma$

Scalability

2D matrix distribution

Parallel cost

$$2\frac{n^2}{p}\gamma + (\log_2(r) + \log_2(c))\,\alpha + \left(\frac{n}{c} + \frac{n}{r}\right)\beta + \frac{n}{r}\gamma$$

Scalability

2D matrix distribution

Parallel cost $$2\frac{n^2}{p}\gamma + (\log_2(r) + \log_2(c))\,\alpha + \left(\frac{n}{c} + \frac{n}{r}\right)\beta + \frac{n}{r}\gamma$$

assuming $r = c = \sqrt{p}$ $$2\frac{n^2}{p}\gamma + \log_2(p)\alpha + \frac{n}{\sqrt{p}}(2\beta + \gamma)$$

Speedup $$S_p(n) = \frac{p}{1 + \frac{p\log_2(p)}{2n^2}\frac{\alpha}{\gamma} + \frac{\sqrt{p}}{2n}\frac{2\beta+\gamma}{\gamma}}$$

Scalability

2D matrix distribution

Parallel cost $$2\frac{n^2}{p}\gamma + (\log_2(r) + \log_2(c))\,\alpha + \left(\frac{n}{c} + \frac{n}{r}\right)\beta + \frac{n}{r}\gamma$$

assuming $r = c = \sqrt{p}$ $$2\frac{n^2}{p}\gamma + \log_2(p)\alpha + \frac{n}{\sqrt{p}}(2\beta + \gamma)$$

Speedup $$S_p(n) = \frac{p}{1 + \frac{p\log_2(p)}{2n^2}\frac{\alpha}{\gamma} + \frac{\sqrt{p}}{2n}\frac{2\beta+\gamma}{\gamma}}$$

Efficiency $$E_p(n) = S_p(n)/p = \ldots$$

Scalability

2D matrix distribution

Parallel cost $$2\frac{n^2}{p}\gamma + (\log_2(r) + \log_2(c))\,\alpha + \left(\frac{n}{c} + \frac{n}{r}\right)\beta + \frac{n}{r}\gamma$$

assuming $r = c = \sqrt{p}$ $$2\frac{n^2}{p}\gamma + \log_2(p)\alpha + \frac{n}{\sqrt{p}}(2\beta + \gamma)$$

Speedup $$S_p(n) = \frac{p}{1 + \frac{p\log_2(p)}{2n^2}\frac{\alpha}{\gamma} + \frac{\sqrt{p}}{2n}\frac{2\beta+\gamma}{\gamma}}$$

Efficiency $$E_p(n) = S_p(n)/p = \ldots$$

Strong scalability $$\lim_{p\to\infty} E_p(n) = 0$$ ✖

Scalability

2D matrix distribution

Parallel cost $$2\frac{n^2}{p}\gamma + (\log_2(r) + \log_2(c))\,\alpha + \left(\frac{n}{c} + \frac{n}{r}\right)\beta + \frac{n}{r}\gamma$$

assuming $r = c = \sqrt{p}$ $$2\frac{n^2}{p}\gamma + \log_2(p)\alpha + \frac{n}{\sqrt{p}}(2\beta + \gamma)$$

Speedup $$S_p(n) = \frac{p}{1 + \frac{p\log_2(p)}{2n^2}\frac{\alpha}{\gamma} + \frac{\sqrt{p}}{2n}\frac{2\beta+\gamma}{\gamma}}$$

Efficiency $$E_p(n) = S_p(n)/p = \ldots$$

Strong scalability $$\lim_{p\to\infty} E_p(n) = 0 \quad ✖$$

Weak scalability

local memory = M $\Rightarrow$ combined memory = Mp
largest problem (n_M) solvable with p processes?
$n_M^2 = Mp$

$$\lim_{p\to\infty} E_p(n_M) = \frac{1}{1 + \frac{\log_2(p)}{2M}\frac{\alpha}{\gamma} + \frac{1}{2\sqrt{M}}\frac{2\beta+\gamma}{\gamma}} = 0 \quad ✖ \quad \ldots ✓$$

Exercise

2D matrix distribution

- A is partitioned according to a 2D mesh of processes, with $p = r \times c$. Process P_{ij} owns block A_{ij}.
- x and y are partitioned in p blocks, and mapped to the mesh by columns.
- Sketch out the algorithm to compute y.
- Determine the cost of the algorithm.
- Study the scalability of the algorithm.

Parallel Matrix Distribution

How to store a (large) matrix using $p = r \times c$ processes?

- 1D, block of rows (or columns) **Bad idea**
- 1D (block) cyclic, either by rows or columns **Bad idea**
- 2D, $r \times c$ quadrants **Not so good**
- 2D (block) cyclic **Good idea!**

References

- Applet:
  `http://acts.nersc.gov/scalapack/hands-on/datadist.html`
- "Introduction to High-Performance Scientific Computing" by Victor Eijkhout (free download).
- "A Comprehensive Approach to Parallel Linear Algebra Libraries" (Technical Report, University of Texas). Chapter 3, pages 19–40.