Solving Triangular Systems in Parallel

Summary of Forward Substitution Algorithms

- Sequential Inner Product Version
- Sequential Vector Sum
- Fine-Grain Parallel Pseudocode

if $i = j$

recv sum reduction $\sigma_i$

$x_i = (b_i - \sigma_i)/l_{ii}$

broadcast $x_i$ to tasks $(k, i)$

$(k = i + 1, \cdots, n)$

else

recv $x_j$

$t = l_{i,j}x_j$

send $t$ for sum reduction across tasks

$(i, k) \ (k = 1, \cdots, (i - 1))$ to task $(i, i)$

end
• Row Partitioning for Vector Sum Algorithm (Fan-out)

The output loop (for all the columns) is sequential. Consequently the number of sequential steps is \( O(n) \). Processors have to wait for information above them before they start updating \( b_j \). Processors broadcast the \( x_j \)'s they are responsible for, as soon as they are calculated.

\[
\text{for } j = 1 \text{ to } n \\
\quad \text{if } j \in \text{myrows then} \\
\qquad x_j = b_j / l_{jj} \\
\qquad \text{broadcast } x_j \text{ to other tasks} \\
\quad \text{else} \\
\qquad \text{recv } x_j \\
\quad \text{end} \\
\text{for } i \in \text{myrows}, \ i > j \\
\quad b_i = b_i - l_{ij} x_j \\
\text{end} \\
\text{end}
\]
• Column Partitioning for Inner Product Algorithm (Fan-in)

The output loop (for all the rows) is sequential. Consequently the number of sequential steps is \( O(n) \). For each row a partial inner product \( (t) \) is calculated and used in a \textit{reduce} across other processors. Processors have to wait for information on their left before calculating \( x_j \)'s.

\[
\text{for } i = 1 \text{ to } n \\
\quad t = 0 \\
\quad \text{for } j \in \text{mycols, } j < i \\
\quad \quad t = t + l_{ij}x_j \\
\quad \text{end} \\
\quad \text{if } i \in \text{mycols} \text{ then} \\
\quad \quad \text{recv sum reduction of } t \\
\quad \quad x_i = (b_i - t)/l_{ii} \\
\quad \text{else} \\
\quad \quad \text{send } t \text{ for sum reduction across tasks} \\
\quad \text{end} \\
\text{end} 
\]
• Wavefront Vector Sum Algorithm

\begin{verbatim}
for j ∈ mycols
    for k = 1 to (# of segments)
        recv segment
        if k = 1 then
            x_j = (b_j - z_j) / l_{jj}
            segment = segment - \{z_j\}
        end
        for z_i ∈ segment
            z_i = z_i + l_{ij} x_j
        end
        if |segment| > 0 then
            send segment to task with column j + 1
        end
    end
end
\end{verbatim}
• Wavefront Scalar Product Algorithm

for $j \in myrows$

for $k = 1$ to ($\#$ of segments $-$ 1)

recv segment

send segment to task owing row $i + 1$

for $x_j \in segment$

$$b_i = b_i - l_{ij}x_j$$

end

end

recv segment

for $x_j \in segment$

$$b_i = b_i - l_{ij}x_j$$

end

$$x_i = b_i/l_{ii}$$

$segment = segment \cup \{x_i\}$

send segment to task owing row $i + 1$

end