By the bandwidth of the memory in which the data resides, the memory transfer and therefore its performance is determined. Little data locality, i.e., the primitive runs at best at 2 operations. Very in the best case this is similar to the matrix-vector primitive.

$$\frac{u \tau \zeta}{\zeta} + \frac{\tau}{\zeta} = \min H \text{ (temporal locality)}$$

$$\frac{\zeta}{u \zeta} + \frac{\tau}{\zeta u} = \min \Theta \text{ (transactions)}$$

$$\frac{\zeta}{u \zeta} + \frac{\tau}{\zeta u} = \min \vartheta \text{ (data)}$$

$$\tau u = \min \omega \text{ (operations)}$$
Not suitable for register-based vector machines

- \[ \frac{2}{n} \] average vector length

- The column sweep is a series of threads' parallel products

- The row sweep requires an efficient dot product

- The row sweep requires row accesses of the array

- \( \approx \) m.o.p

\[ L \] BLAS I primitives

These two algorithms represent a decomposition into respectively yielding the row-sweep and column-sweep algorithms directly yielding the standard sequential versions vectorize

The inner loops of the standard sequential versions vectorize

Decomposition into Primitives and Fine Grain

Parallelism
Each row or column of \( T \) can be read exactly once. Each row or column of \( T \) can be read exactly once. This is clearly the case for these small systems. can be loaded

\[
\frac{1}{n} \approx \frac{n}{\text{column}} = \frac{\text{row}}{\text{column}} = \frac{\text{row}}{n}
\]

Fact: If \( n \geq \sum \text{row} \), then the number of transfers per operation

Therefore want a small number of transfers per operation

Assume vector registers of length \( R \), i.e., small fast memory in

Local Memory or Register-based Vector Processors
small triangular system solves.

- matrix-vector products

So we want a triangular solver that uses small problems.

Reduction operations at the BLAS2 level also have this optimality
with $L(i)$, $x(i)$, and $f(i)$ being each of order $v$ (we assume that $v$

divides $n$).

\[
\begin{pmatrix}
x(i) \\ x_{i+1} \\ \vdots
\end{pmatrix} = 
\begin{pmatrix}
L(i) \\ x_{i+1} \\ \vdots
\end{pmatrix}
\]

Let $L^{(0)} = L$, $f^{(0)} = f$, and let each of $L^{(i)}$, $x^{(i)}$, and $f^{(i)}$ be of
order $(n - jv) j = 0, \ldots, \frac{n}{v} - 1$ where

Block Column Sweep
solve via COL-Sweep or ROW-Sweep.

\( (\mathbf{I} \mathbf{z} - \mathbf{I} \mathbf{x}) \mathbf{T} \mathbf{f} = \mathbf{I} \mathbf{z} \mathbf{f} \)

\( \mathbf{f} = (\mathbf{I} \mathbf{x}) (\mathbf{I} + \mathbf{I}) \mathbf{T} \)

\( 1 \) \( ? \quad \mathbf{d} = \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \)

\( \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \mathbf{d} \quad \math
\[ u = u, \text{ where } u \text{ is bounded in magnitude by } I. \]

\[ u + \frac{\mathbb{N}}{I} + \frac{\mathbb{Z}}{I} = s \eta \]

It follows that the smaller triangular systems:

algorithm is the column sweep or whatever is used to solve the
just 1 (the algorithm is the column sweep and if the
register-based vector processors.

Uses two primitives: a small triangular system solver and a
end do.
solve \( \ell f = \ell x^\ell T \) via Col-Sweep or Row-Sweep

\[
(\ell - 1) x^\ell \sigma - \ell f = \ell f
\]

\[d' z = \ell \]
do

solve \( \ell q = \ell x^\ell T \) via Col-Sweep or Row-Sweep

\[
\frac{a}{u} = d
\]

Row-Sweep:

The block algorithm is:

where \( \mathcal{A} \in \mathbb{C} \), if \( \ell \) \( x \in \mathbb{C} \)

\[
\ell L(\ell d, \ldots, \ell f) = f \quad \text{and} \quad \ell L(\ell x, \ldots, \ell x) = (\ell x) \quad \ell L(\ell x, \ldots, \ell x) = x
\]

\[
\ell a \times \mathcal{A} \in \ell \mathbb{C}\text{ and }a_{(\ell - 1) \times \mathcal{A}} \in \ell \mathbb{C}
\]

Partition \( T \) so that each block row is of the form \([0, T, C]\), where

Block Row Sweep
where $u$ is bounded in magnitude by $1/n$.

$$u + \frac{2}{I} + \frac{2}{I} = s_{\text{res}}$$

It follows that tall narrow matrices, matrix-vector product operates on short, wide matrices rather than uses the same two primitives as the block column sweep but the
Block Column Sweep Partitioning

\[ L x f \]

Block Row Sweep Partitioning

\[ x f \]

For both partitionings, \( x i \in R^n \) and \( f i \notin R^n \).
task is half the operations of an M task. refer to these as S tasks and W tasks respectively. Note that an S this can be done indivisibly via a lock/unlock for now.) We will represent matrix-vector tasks \( f' \rightarrow f' x \) if we will assume that the squares represent solve tasks \( i f' = i f' x \) and the circles respect the data dependence graph seen below for \( k = 6 \).

\[ \frac{n}{u} = k \cdots \frac{n}{w} = 1 \]

\[ \ell f = \ell f \]

Consider partitioning into submatrices \( I \in \mathcal{X} \times \mathcal{Y} \) Parallel algorithms
Consider what can happen in parallel for this task graph.

Operations at each level can be done in parallel.

Level 9: the S task for node (5,5)

Level 8: M tasks for nodes (5,4)

Level 7: the S task for node (4,4)

Level 6: M tasks for nodes (5,3) and (4,3)

Level 5: the S task for node (3,3)

Level 4: M tasks for nodes (5,2), (4,2), and (3,2)

Level 3: the S task for node (2,2)

Level 2: M tasks for nodes (5,1), (4,1), (3,1), and (2,1)

Level 1: the S task for node (1,1)
path, is $k$ tasks and $k - 1$ tasks.

The maximum width is $k - 1$ therefore at most $k - 1$ processors are

needed. The height of the task graph, the length of the critical
\[ \frac{uL}{z} \gamma \approx \frac{I}{L} \]

where \( z/\frac{uL}{z} \gamma \approx \frac{s}{L} \) is used and a time on one processor of task respectively we have a total time on \( k - 1 \) processors of \( \frac{uL}{z} \gamma \approx \frac{uL}{L} (1 - \gamma) + \frac{s}{L} \gamma = \frac{1}{L} - \gamma L \)

If \( uL \) and \( sL \) represent the time on one processor for an S and M.
\[
\begin{align*}
\nu \xi & = V \\
\xi / \nu & \approx \nu_{\text{ave}} \\
I - \nu & = \nu_{\text{max}}
\end{align*}
\]

In the limit as \( I \to 1 \) we have

\[
\xi / \gamma \approx (1 - \gamma) / (I - 1) - 1
\]

Efficiency \( \xi / \gamma = 1 - \gamma \frac{I}{\bar{I}} \) Speedup \( \xi / \gamma = 1 - 1 \)
Homework 2 is an example of such an algorithm.

that the transformation-based banded lower triangular solver of
work. This will not be discussed any further in this class, but note
time if we have more processors and we perform some redundant
compared to the sequential algorithm. It is possible to take less
the parallel algorithm does not have any redundant operations.
Note that $\lambda_{max}$ and $\lambda_{ave}$ are computed under the assumption that
correspond to dot products.

row sweep graph is identical with $\mathcal{N}$, replaced by $\mathcal{M}$, and the nodes

clear. The computation graph for the column sweep follows. Each

The source of the critical path for the standard algorithms is