Tree Accumulations on GPU

With Applications to Sparse Linear Algebra

Scott Rostrup, Shweta Srivastava,
Kishore Singhal,
Synopsys Inc.

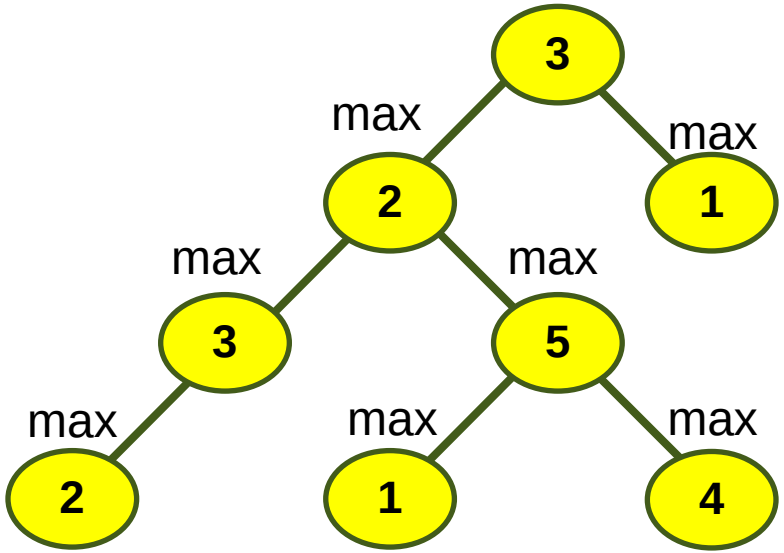
Applications

- Tree Structure Computations
  - i.e. subtree size, subtree weight, depth, …
- Sparse Iterative Solvers
  - Direct Solve of Tree Matrix (Preconditioning)
- Fast Multipole Method
- DNA/Protein Sequence Alignment
- Tree Drawing
- XML Data Aggregation

Overview

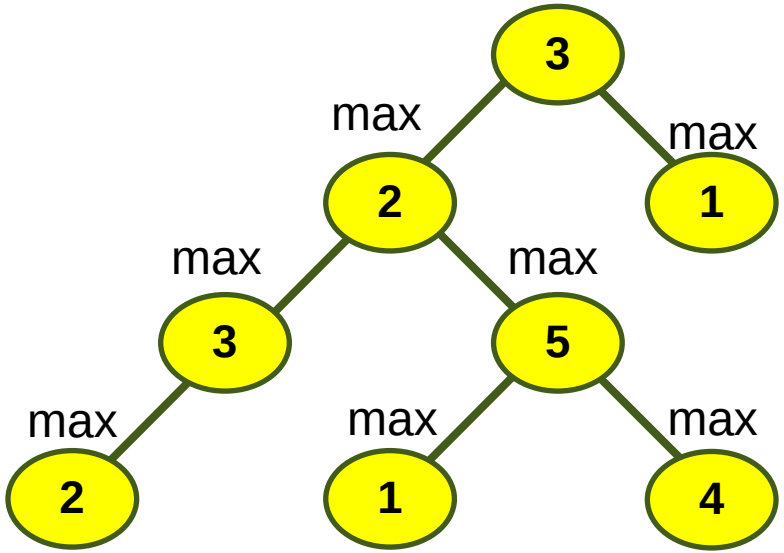
- Up to **10X** Improvement over CPU
- On a 4GB GPU card:
  - Can accumulate trees up to **45M** vertices
  - Can solve Linear systems with up to **25M** unknowns
- Performance is logarithmic with respect to tree depth

Tree Accumulation



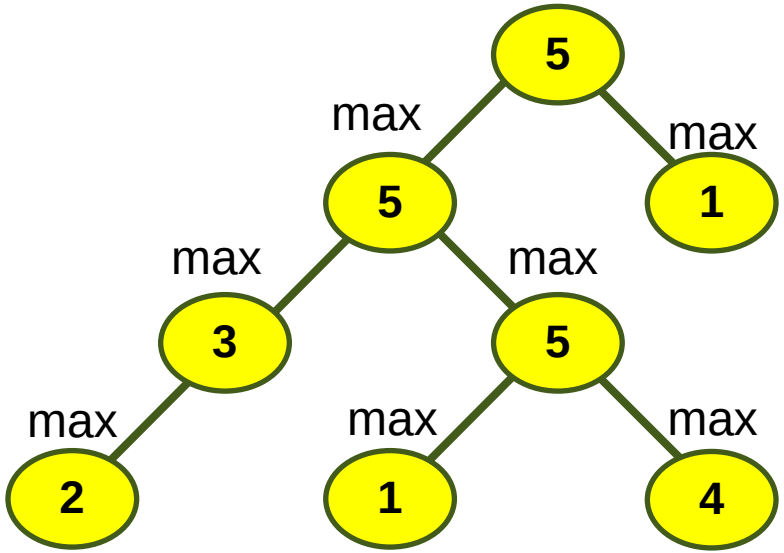
- Data stored at Tree Nodes
- Same operator applied to each edge
  - i.e. +, x, max, min, …
  - Associative and Commutative required for parallel computation
- Compute and Store all intermediate results

Upwards Tree Accumulation



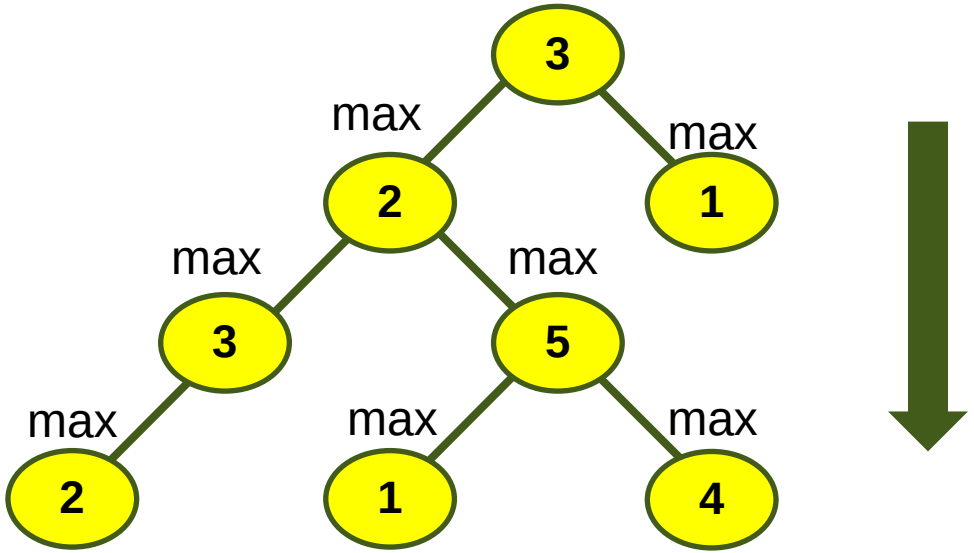
- Leaf -> Root
- Finds Maximum Descendent Value

Upwards Tree Accumulation



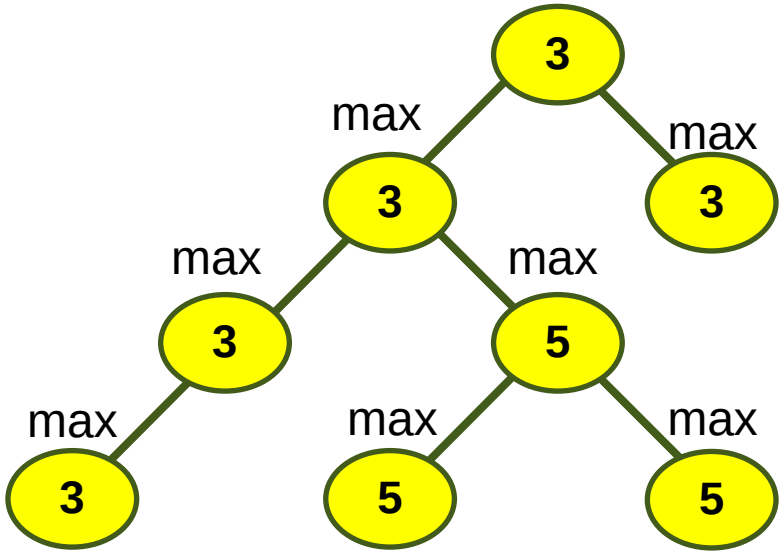
- Leaf -> Root
- Finds Maximum Descendent Value

Downwards Tree Accumulation



- Root -> Leaf
- Finds Maximum Ancestor Value

Downwards Tree Accumulation



- Root -> Leaf
- Finds Maximum Ancestor Value

Application: Symmetric Tree + Star Graph Matrix

- Symmetric Matrix *A*
  - Consider n×n matrix *A* with elements a_{ij}
  - Symmetric: $a_{ij} = a_{ji}$ $x^T Ax > 0 \qquad \forall x, s.t. \|x\| > 0$
  - Positive Definite
  - Graph of A is a tree + star graph
- Admits Zero Fill-In Cholesky Factorization with suitable vertex ordering
- $A = LDL^T$
  - *D* is diagonal, *L* is lower triangular
- Used in Tree-based Preconditioning

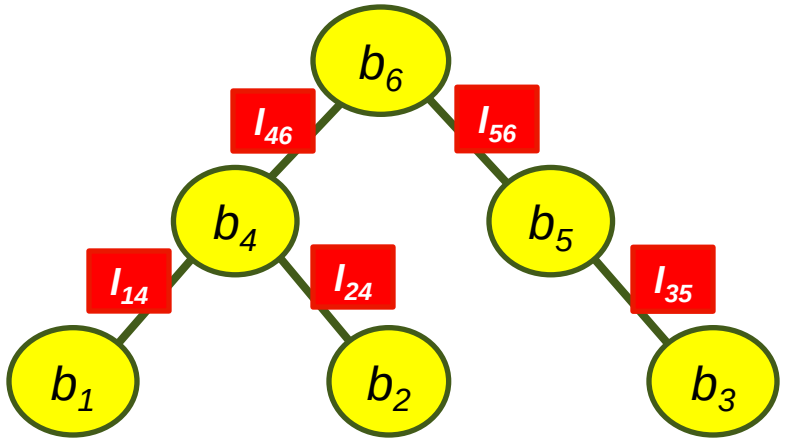
Tree Matrix: Lower Triangular Solve

$$Lx = b$$

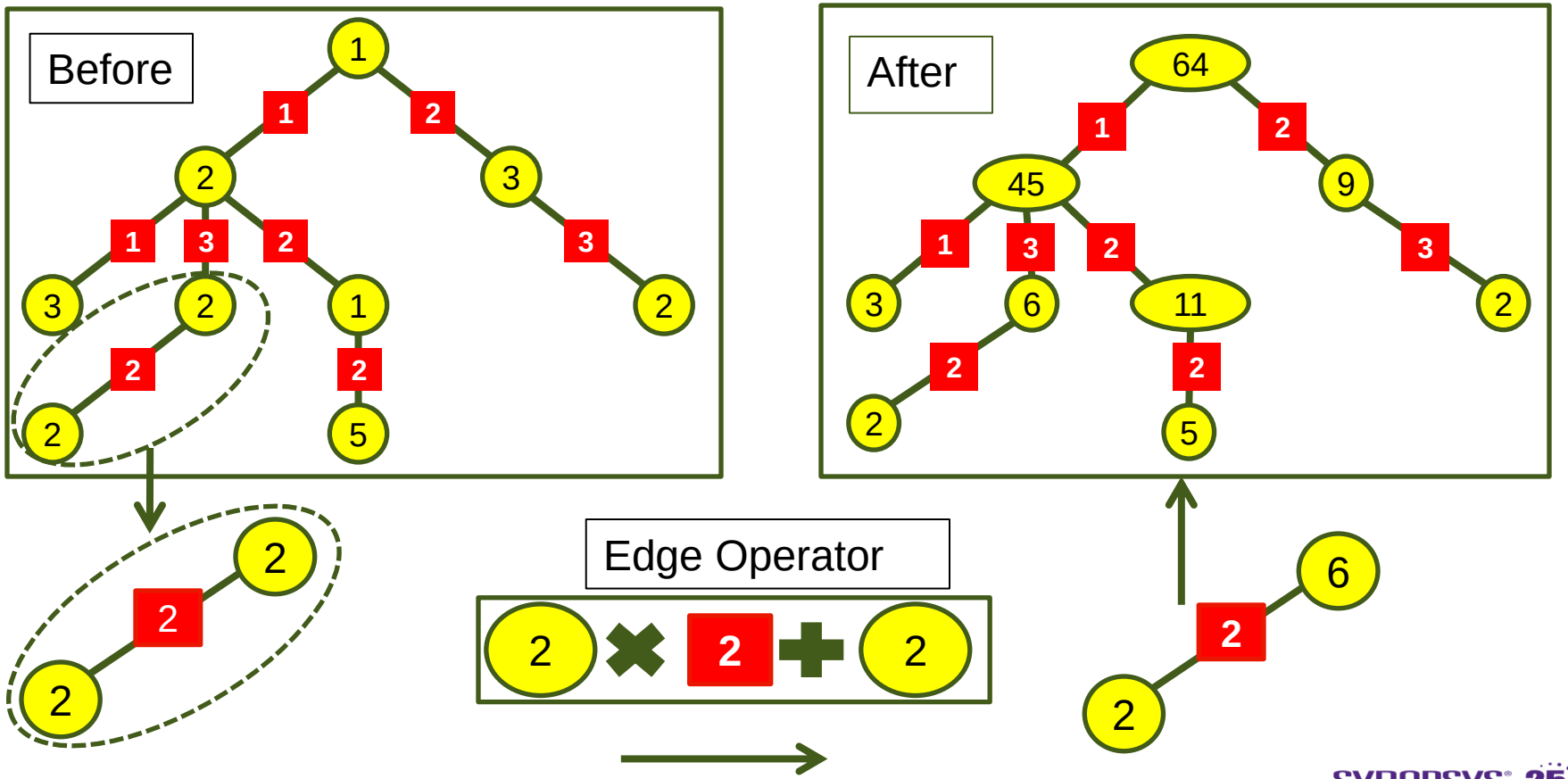
Tree Representation:

- Each row/column becomes a node.
- Each off-diagonal in *L* becomes an edge weight.

$$\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ -l_{14} & -l_{24} & & 1 & & \\ & & -l_{35} & & 1 & \\ & & & -l_{46} & -l_{56} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix}$$

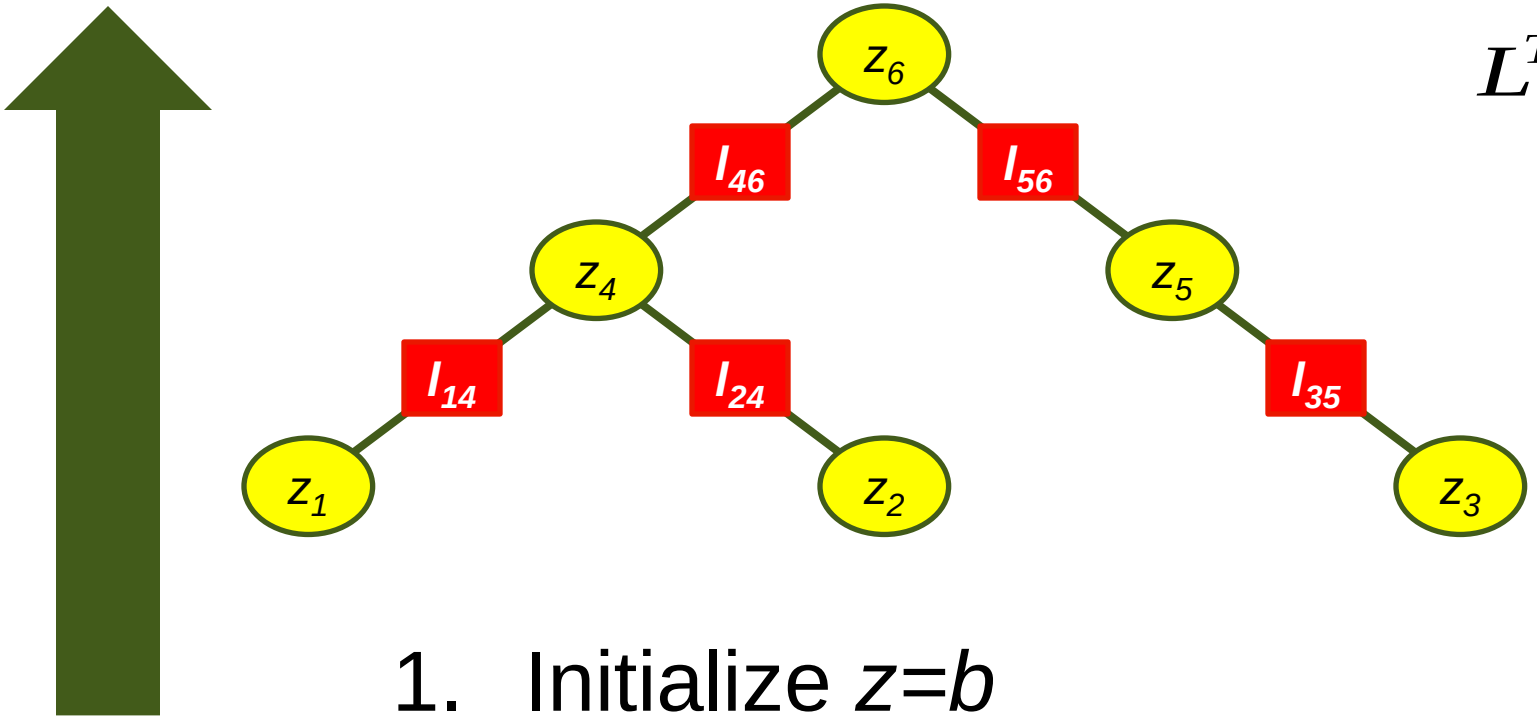


Upwards Weighted Edge Accumulation



Lower Triangular Solve

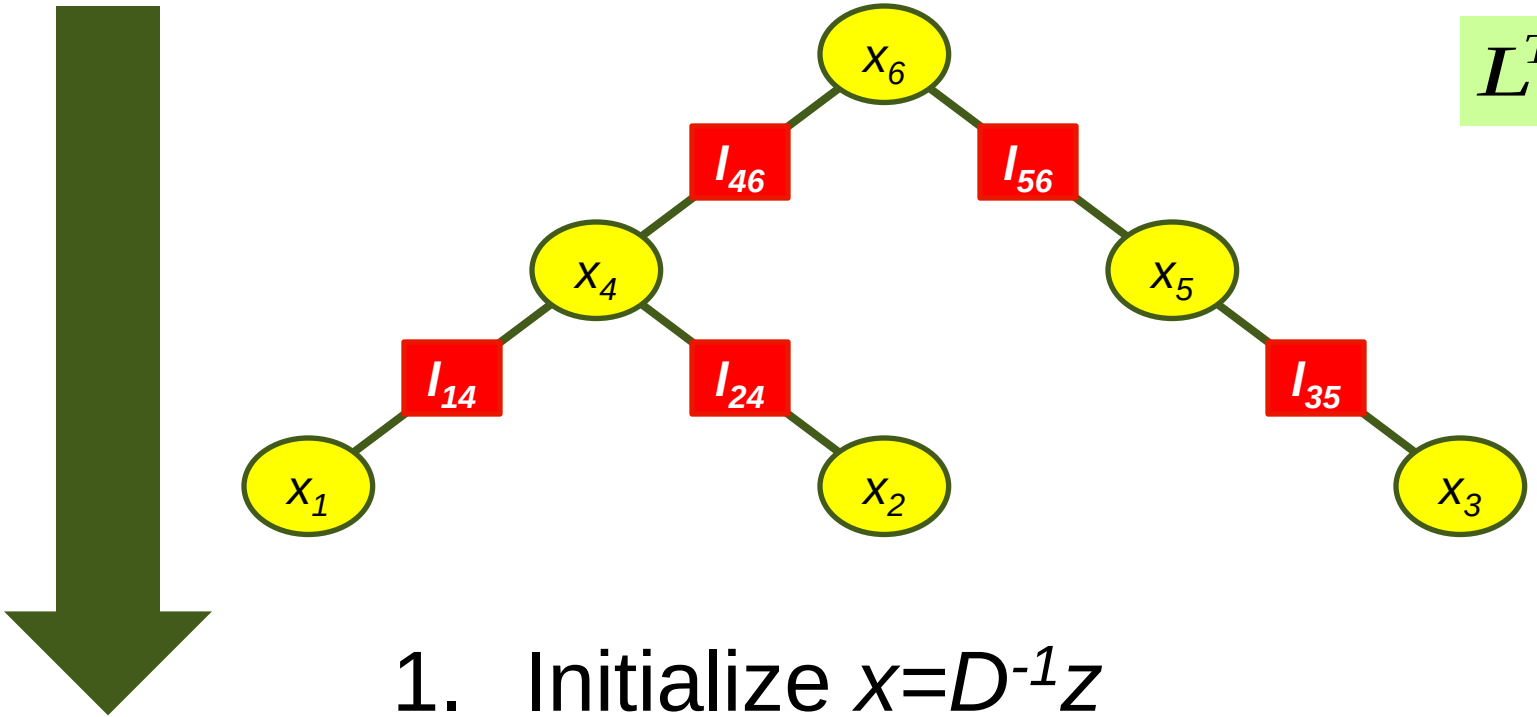
$$Lz = b$$
$$Dy = z$$
$$L^T x = y$$



1. Initialize $z=b$
2. Accumulate Upwards

Upper Triangular Solve

$$Lz = b$$
$$Dy = z$$
$$L^T x = y$$



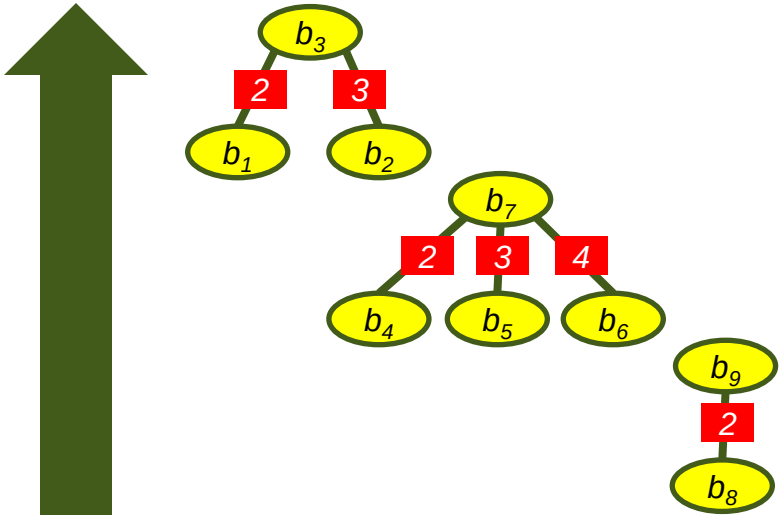
1. Initialize $x=D^{-1}z$
2. Accumulate Downwards

Parallel Tree Accumulation
GPU

- Sequence of Parallel Steps
  - Parallel Step: SpMV
  - Control Flow: Parallel Scan
- Operator Properties
  - Associative, Commutative
  - Distributive for linear algebra: ex. + and x
- Efficient when tree structure used many times

Parallel Step

Child to Parent Reduction



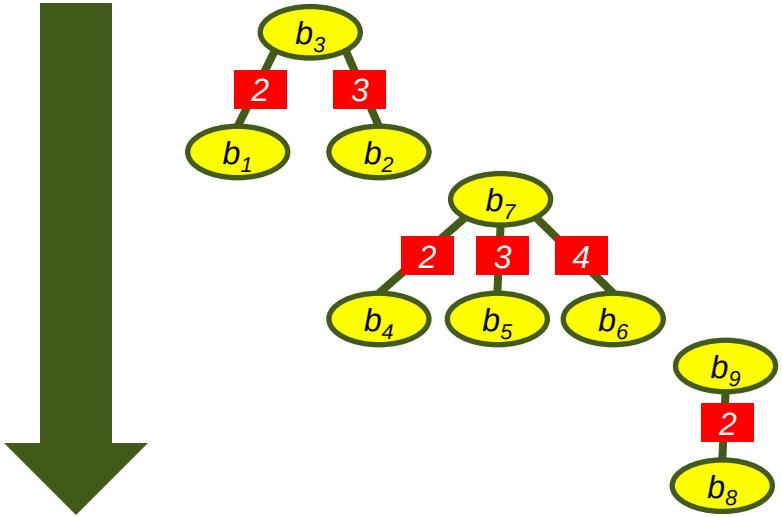
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ 2 & 3 & 1 & & & & & & \\ & & & 1 & & & & & \\ & & & & 1 & & & & \\ & & & & & 1 & & & \\ & & & 2 & 3 & 4 & 1 & & \\ & & & & & & & 1 & \\ & & & & & & & 2 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{bmatrix}$$

Parallel Reduce

Segmented Reduction (i.e. SpMV)

Parallel Step

Parent to Child Scatter



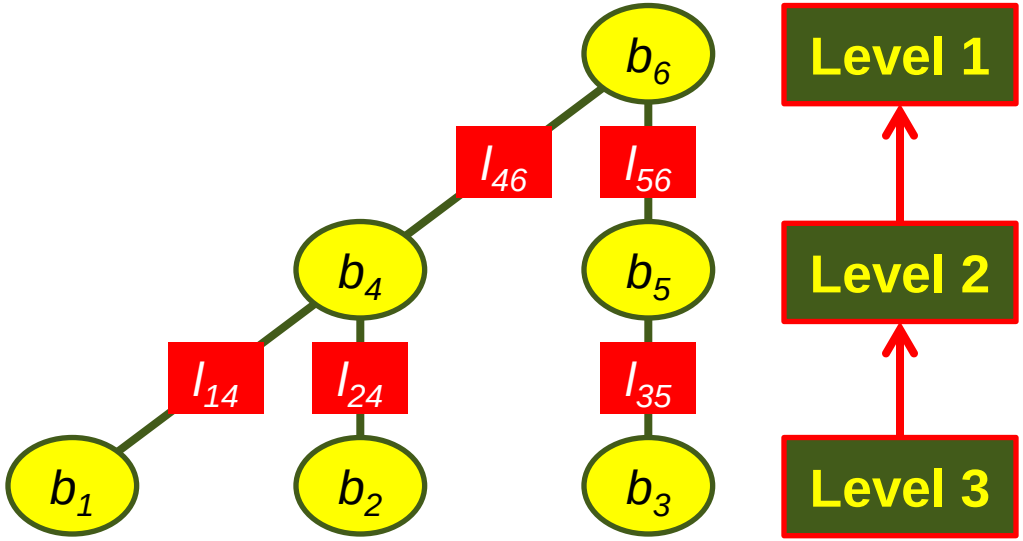
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} 1 & & 2 & & & & & & \\ & 1 & 3 & & & & & & \\ & & 1 & & & & & & \\ & & & 1 & & & 2 & & \\ & & & & 1 & & 3 & & \\ & & & & & 1 & 4 & & \\ & & & & & & 1 & & \\ & & & & & & & 1 & 2 \\ & & & & & & & & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{bmatrix}$$

Parallel Scatter

No Reduction
(Scatter: x and +)

Control Flow

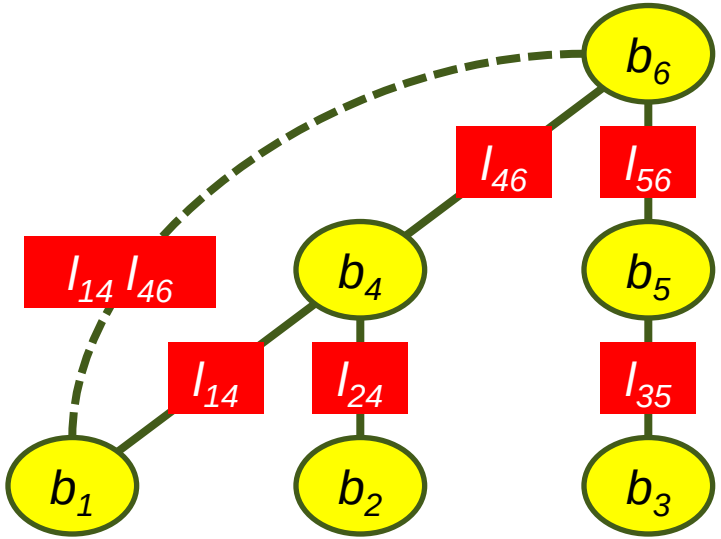
Parallel Scan Algorithm



- Scan between levels
- Blelloch parallel scan algorithm[1]
- Scan algorithm defines the matrices

[1] Guy E. Blelloch. “Prefix Sums and Their Applications”. In John H. Reif (Ed.), Synthesis of Parallel Algorithms, Morgan Kaufmann, 1990.

Path Compression



Level 1

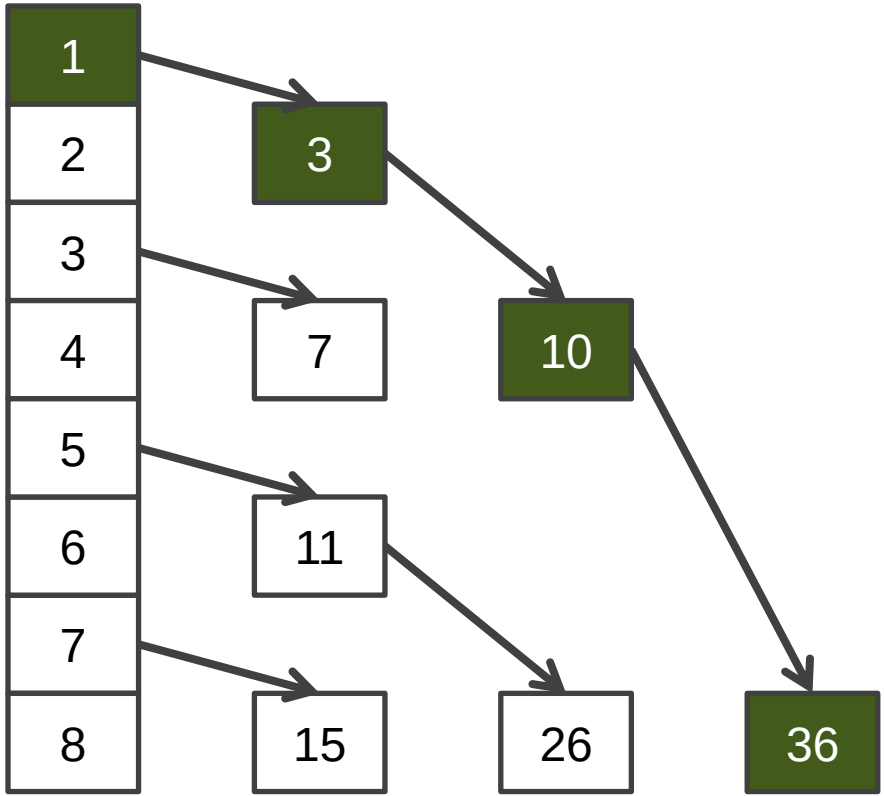
Level 2

Level 3

- Allows jumps between non-adjacent levels
- Precompute combined edge weights
- Works because multiplication and addition are distributive

Reduction Phase

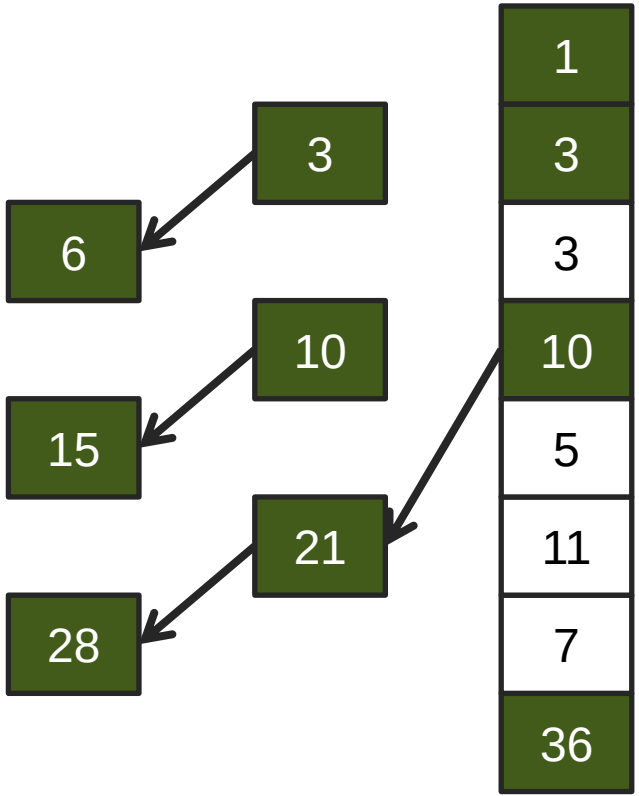
Blelloch Addition Example



- $k = \log_2(\text{length})$
- There are k reduction phases
- At all power of 2 positions, the final value is computed

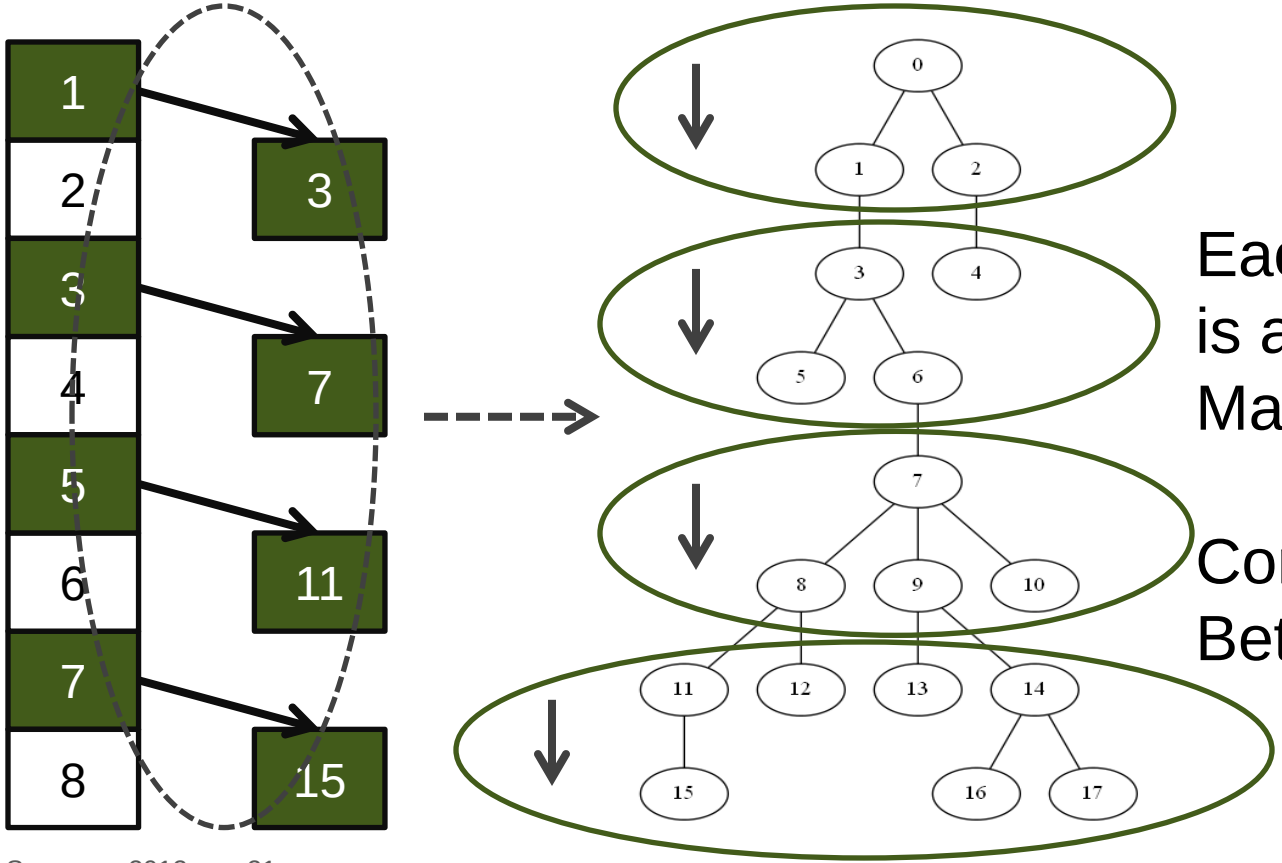
Distribute Phase

Blelloch Addition Example



- There are *k* distribution phases
- Values at multiple of 2^m indices are distributed to the other positions
  - m=1,2,…,k

Explicit Blelloch Matrices



Each Circled Edge is an entry in the Matrix

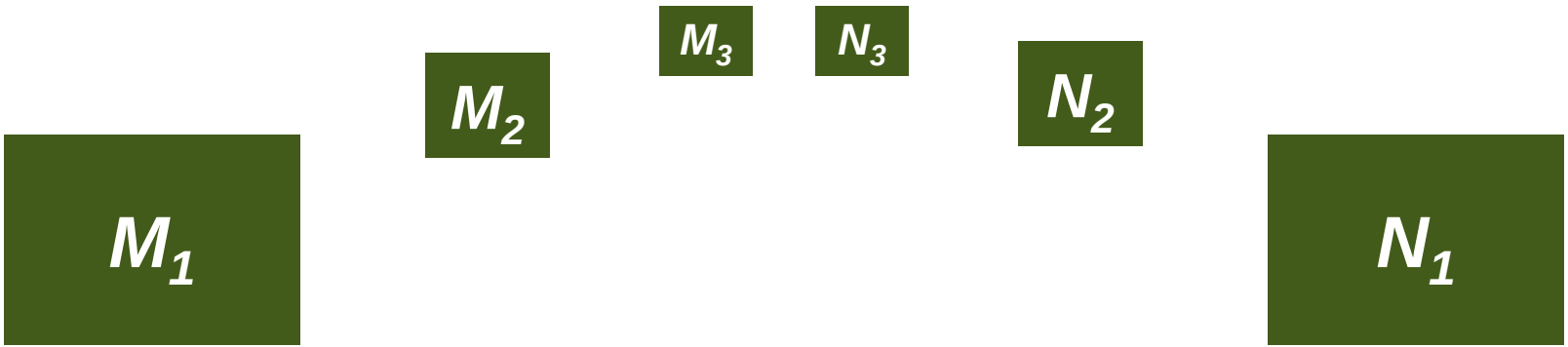
Communication Between Levels

SYNOPSYS 25 Years

Sequence of Matrix Vector Products

Each step is a sparse matrix

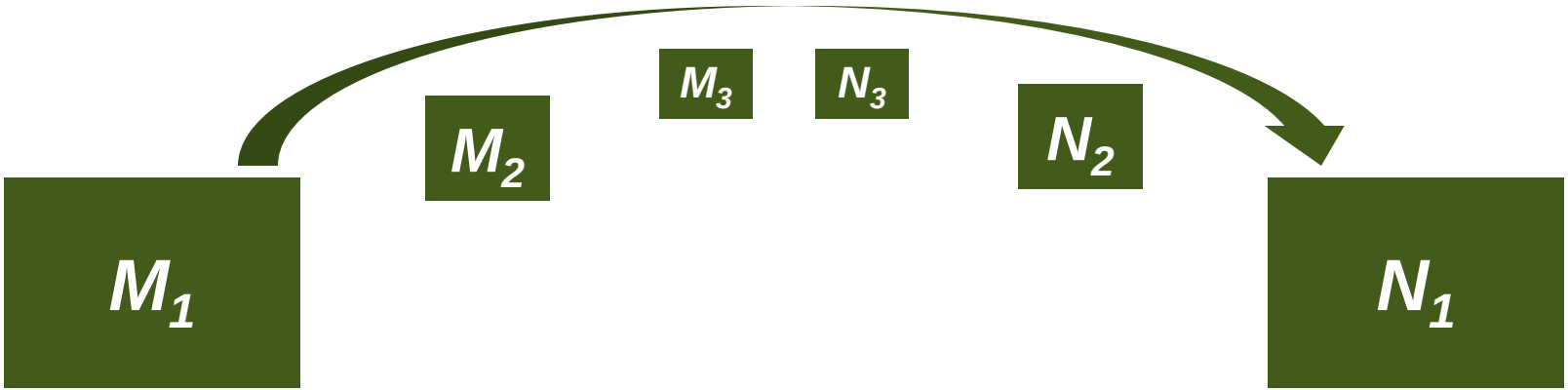
- Reduction Matrices: $\{M_1, M_2, \cdots, M_k\}$
- Distribution Matrices: $\{N_1, N_2, \cdots, N_k\}$



Upwards Accumulation

$$Lz = b$$
$$Dy = z$$
$$L^T x = y$$

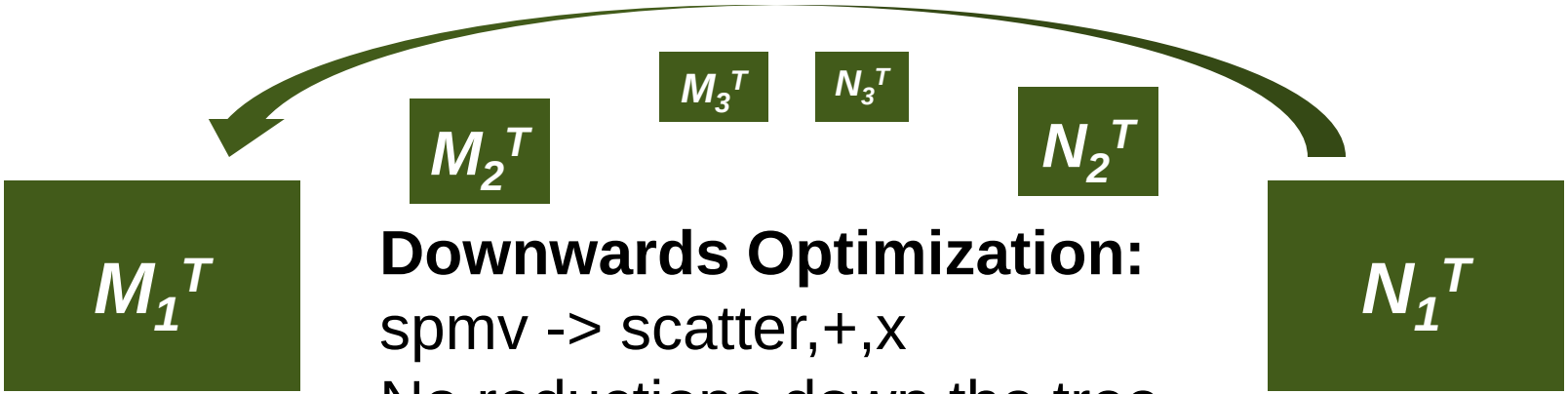
```
z = b
for i in 1,2, … ,k:
    z = SpMV(M_i,z)
for i in k,k-1, … ,1:
    z = SpMV(N_i,z)
```



Downwards Accumulation

$$Lz = b$$
$$Dy = z$$
$$L^T x = y$$

```
x = y
for i in 1,2, … ,k:
    x = SpMV(N_i^T,x)
for i in k,k-1, … ,1:
    x = SpMV(M_i^T,x)
```



Downwards Optimization:

spmv -> scatter,+,x

No reductions down the tree

Algorithmic Preprocessing Details

- Factorization is on CPU
- Euler Tour Technique + List Ranking
  - Parent Relationships
  - Vertex Depth
- Pointer Jumping (Downwards Accumulation)
  - Compressed Edge Weights
- All implemented using thrust library primitives
  - Scan, radix sort

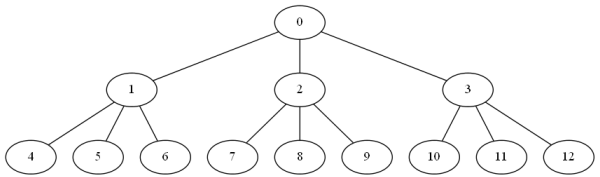
Test Set-Up

- Machines
  - Tesla T10 GPU with 4GB Memory (1/4 S1070)
  - Intel Nehalem @ 2.8 GHz

Test Set-Up

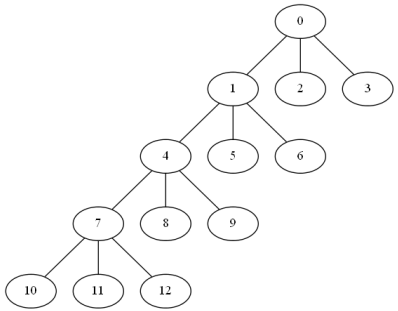
- Random Tree Generator
  - Each Vertex has between 0 and 2*c-1 children
  - Depth Factor scales between **0** and **1**

Uniform Tree

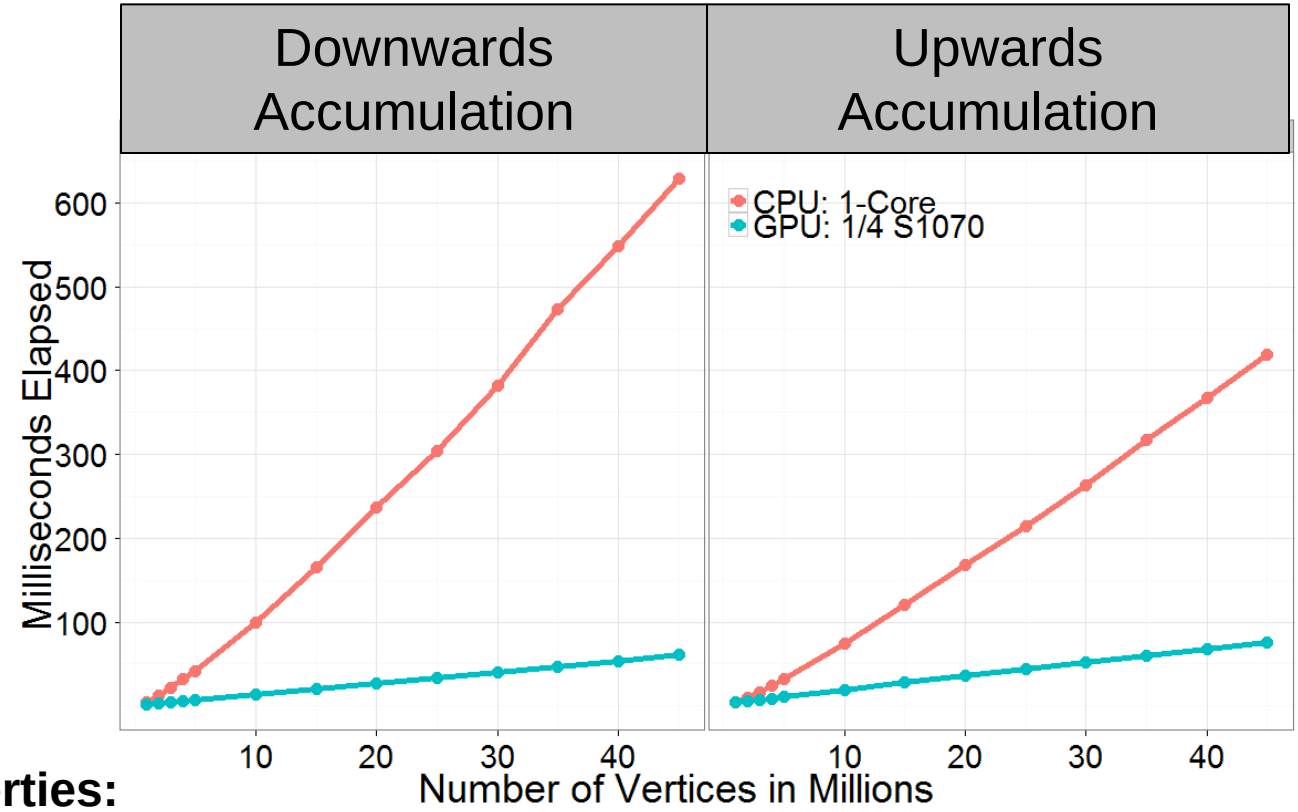


0 **1**

Linear Tree



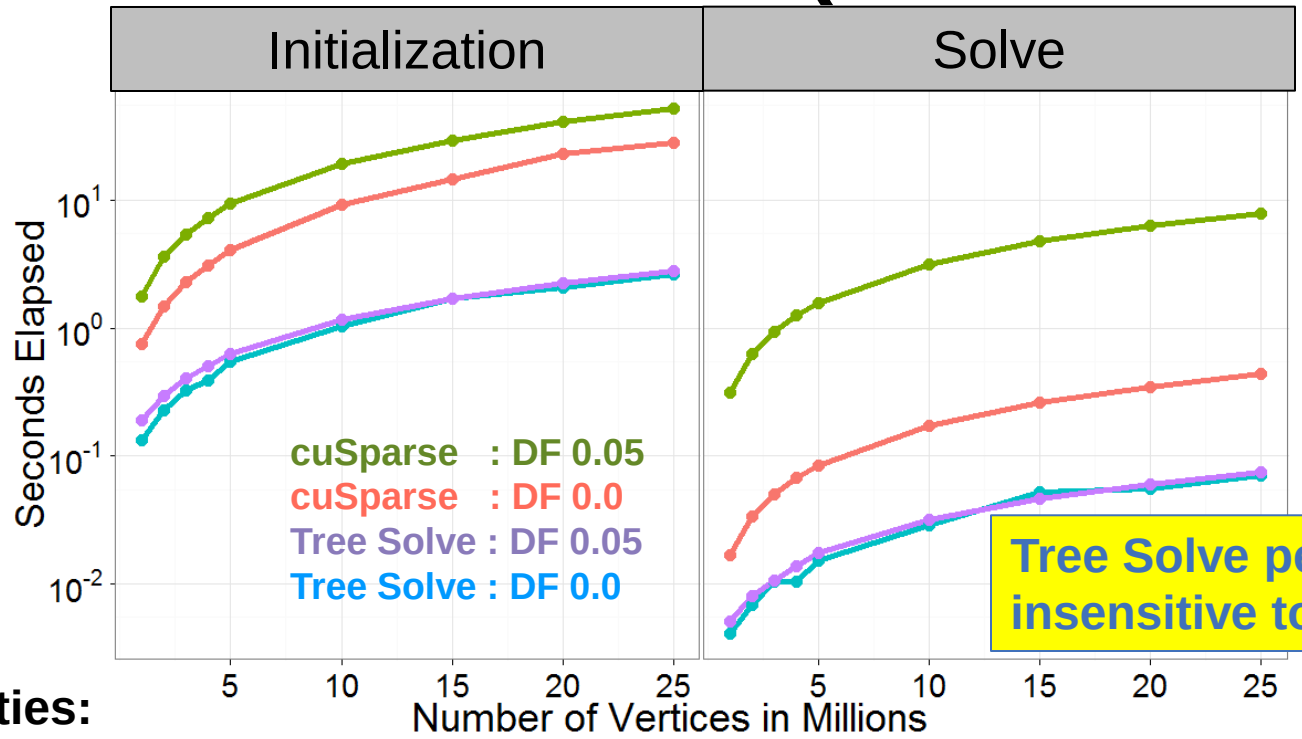
CPU vs. GPU



Tree properties:

- Average children per vertex is 3
- Depth Factor = 0.5 -> depth is approximately (number of vertices)/6

Tree vs. Generic Solve (cuSPARSE)



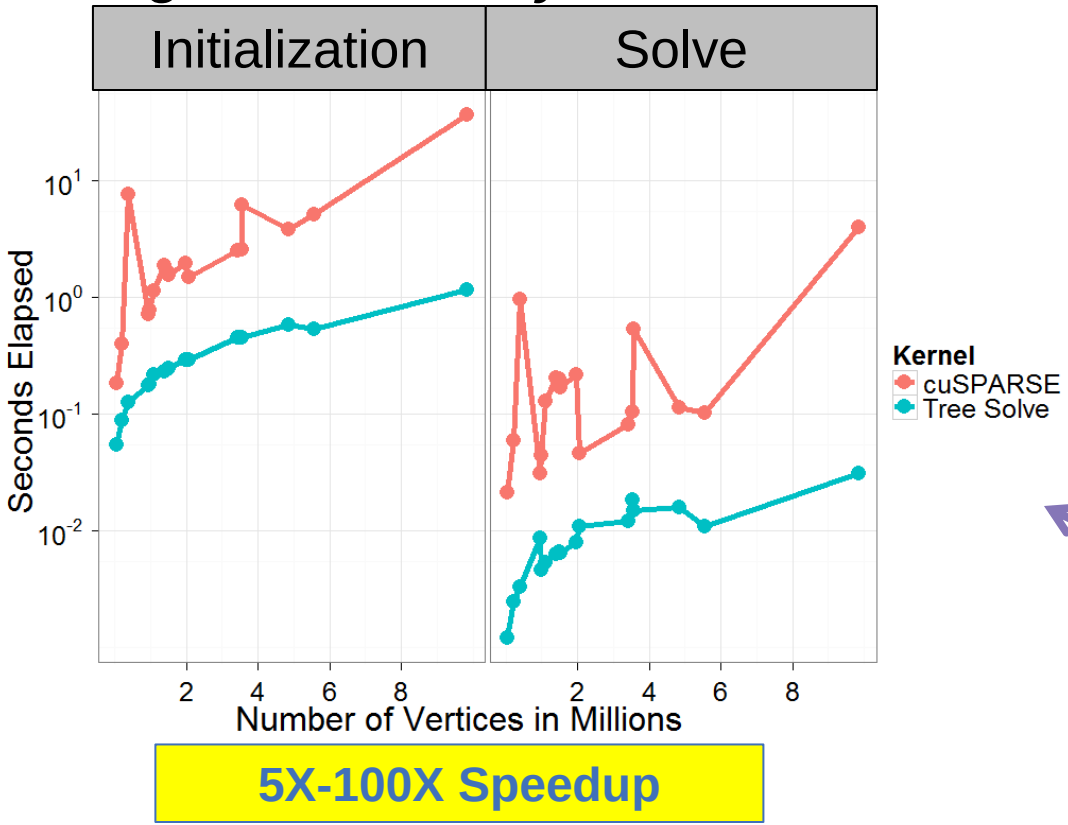
Tree Solve performance insensitive to depth

Tree properties:

- Average of 3 children per vertex
- Depth Factor (DF) = 0.05 -> depth is approximately n/60
- Depth Factor = 0 -> depth < 20; **Best Case for cuSPARSE**

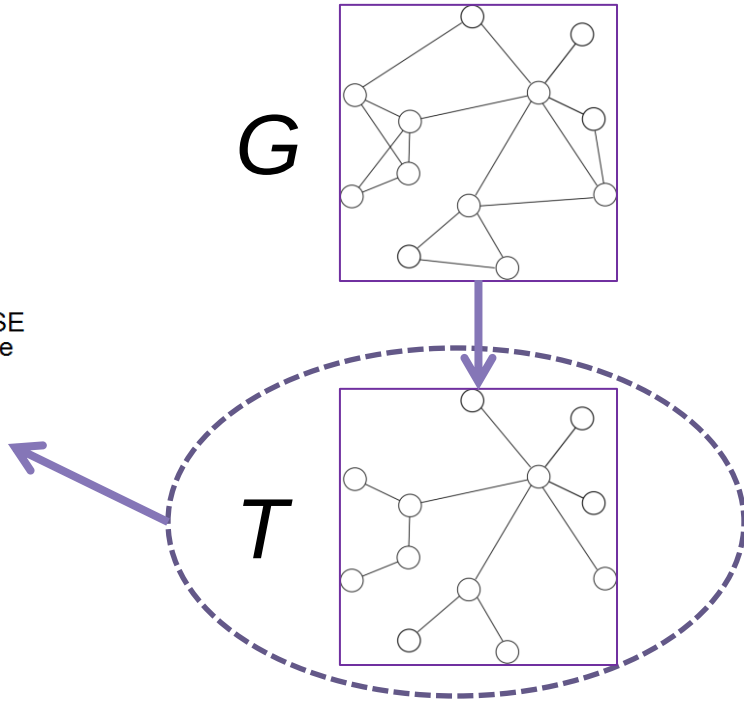
Minimum Spanning Tree Solve

Solving a reduced system



5X-100X Speedup

Graphs from Florida Sparse Matrix Collection



SYNOPSYS 25 Years

Conclusions

- Up to **10X** faster than CPU
- Significantly more efficient than using a generic solver
  - Greater than **5X** faster for all instances tested
  - Runtime unaffected by tree depth
- On a 4GB GPU card:
  - Can accumulate trees up to **45M** vertices
  - Can solve Linear systems with up to **25M** unknowns
- Data-Parallel Primitives are effective for building irregular algorithms

List of Graphs From Florida Sparse Matrix Collection

- roadNet-TX
- roadNet-PA
- roadNet-CA
- RM07R
- wikipedia-20070206
- wb-edu
- thermomech_dK
- soc-LiveJournal1
- af_shell10
- atmosmodl
- audikw_1
- bone010
- bone010_M
- circuit5M
- Freescale1
- kkt_power
- mouse_gene
- nlpkkt120