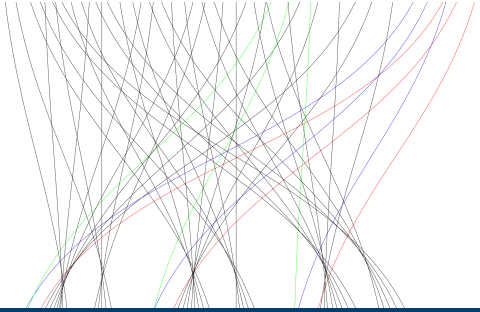


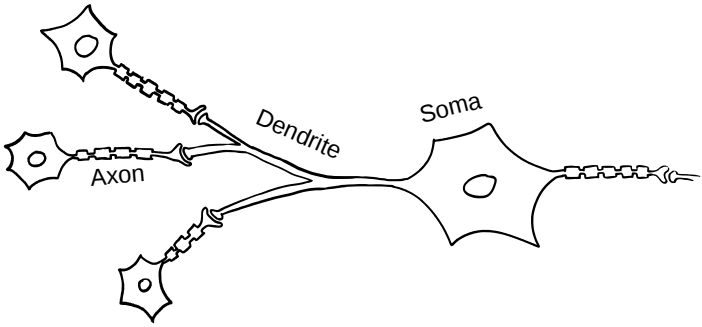
Efficient Solver for Hines Matrices on GPUs

I packed my bag and in it I put lots of dendrites

September 29, 2018 | Felix Huber | University of Stuttgart

Modeling

Neuron



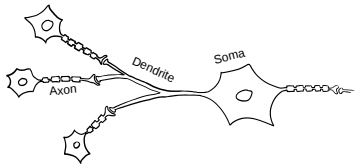
[1]

- Dendrite: 1D electric flux on tree structure
- Soma: Emits spikes if the voltage rises over a threshold
- Axon: The output for the spike signal modeled by a time delay

1D flux with varying properties in a tree structure

Modeling [2]

Discretization



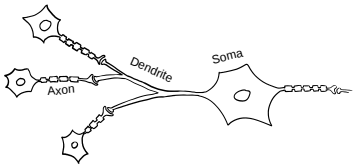
- Each branch: subdivide each branch →tridiagonal matrix
- With branching: almost tridiagonal →Hines matrix

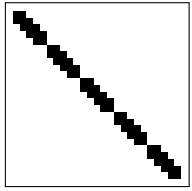
Which Solver?

- Tridiagonal matrix: Thomas algorithm
- Hines matrix: tree solver, starting at the leaves

Modeling [2]

Discretization





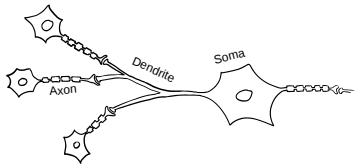
- Each branch: subdivide each branch →tridiagonal matrix
- With branching: almost tridiagonal →Hines matrix

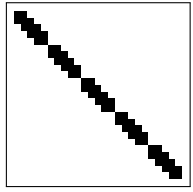
Which Solver?

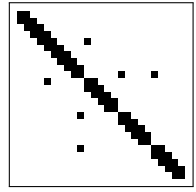
- Tridiagonal matrix: Thomas algorithm
- Hines matrix: tree solver, starting at the leaves

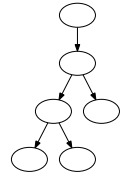
Modeling [2]

Discretization









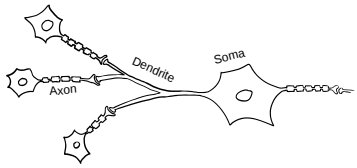
- Each branch: subdivide each branch →tridiagonal matrix
- With branching: almost tridiagonal →Hines matrix

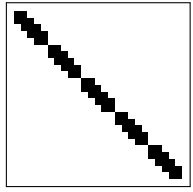
Which Solver?

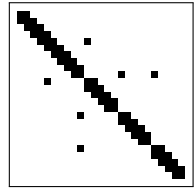
- Tridiagonal matrix: Thomas algorithm
- Hines matrix: tree solver, starting at the leaves

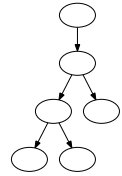
Modeling [2]

Discretization









- Each branch: subdivide each branch → tridiagonal matrix
- With branching: almost tridiagonal → Hines matrix

Which Solver?

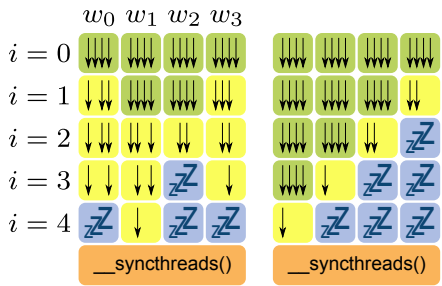
- Tridiagonal matrix: Thomas algorithm
- Hines matrix: tree solver, starting at the leaves

We've got Hines matrices and know how to solve them, let's make it fast!

GPU Recap

Many Threads, grouped in blocks. Good for parallelization:

- enough threads and blocks
- latency hiding, memory access
- low divergence
- independent tasks



kernel

```
int count = counts[threadIdx.x];
for (int i = 0; i < count; ++i) {
    // smart stuff
}
__syncthreads();
```

Goals

Pack our problems in blocks to

- allow larger workloads
- expose enough parallelism to utilize all processing units
- balance workload

Different parallelization approaches:

- One Cell per Thread: not parallel enough
- One Branch per Thread: finer level of granularity, one block only
- One Branch per Thread with Batching: distribute cells in multiple blocks
- One Branch per Thread with Balancing: balancing the work per thread to avoid idle threads

Parallelization

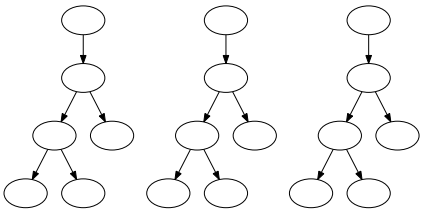
One Branch per Thread

Problem: We need to synchronize the branch solvers.

Why?

- each branch has to update RHS of parent

Backward substitution



Parallelization

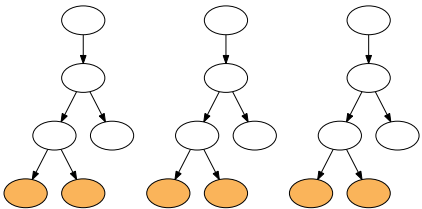
One Branch per Thread

Problem: We need to synchronize the branch solvers.

Why?

- each branch has to update RHS of parent

Backward substitution



Parallelization

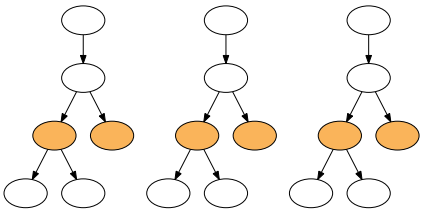
One Branch per Thread

Problem: We need to synchronize the branch solvers.

Why?

- each branch has to update RHS of parent

Backward substitution



Parallelization

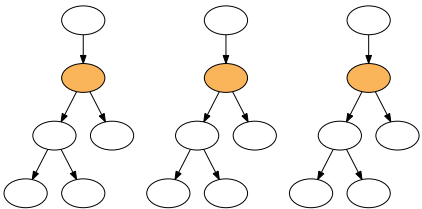
One Branch per Thread

Problem: We need to synchronize the branch solvers.

Why?

- each branch has to update RHS of parent

Backward substitution



Parallelization

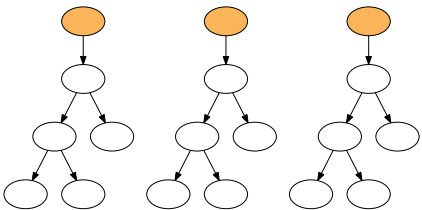
One Branch per Thread

Problem: We need to synchronize the branch solvers.

Why?

- each branch has to update RHS of parent

Backward substitution & forward substitution



Parallelization

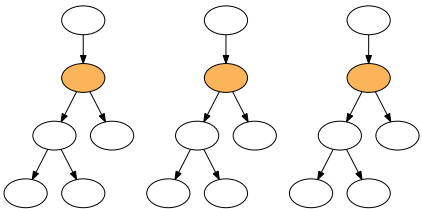
One Branch per Thread

Problem: We need to synchronize the branch solvers.

Why?

- each branch has to update RHS of parent

Forward substitution



Parallelization

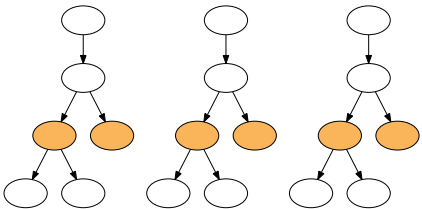
One Branch per Thread

Problem: We need to synchronize the branch solvers.

Why?

- each branch has to update RHS of parent

Forward substitution



Parallelization

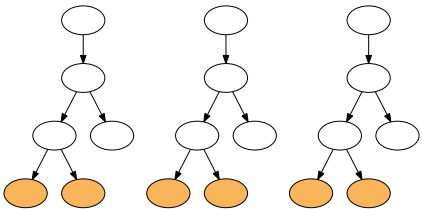
One Branch per Thread

Problem: We need to synchronize the branch solvers.

Why?

- each branch has to update RHS of parent

Forward substitution



Parallelization

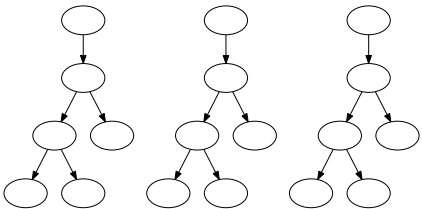
One Branch per Thread

Problem: We need to synchronize the branch solvers.

Why?

- each branch has to update RHS of parent

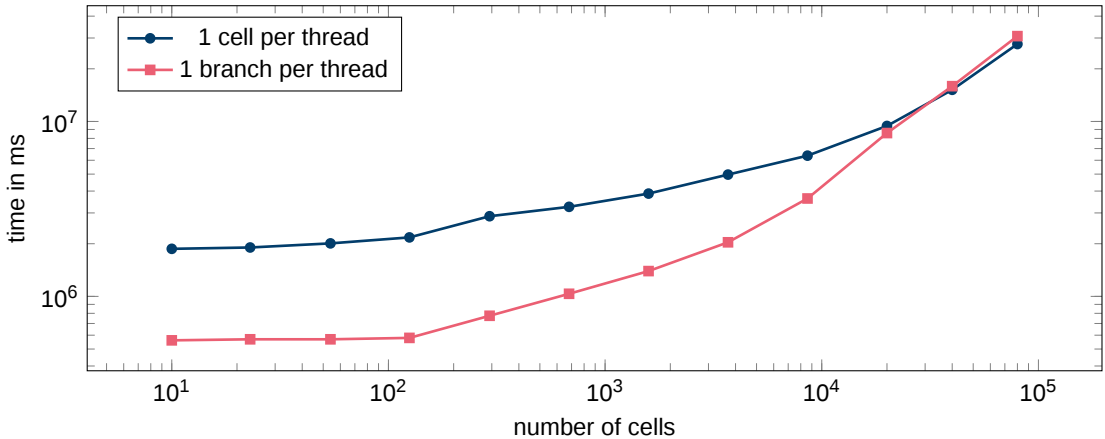
Forward substitution



Distribute cells into Cuda blocks →no global synchronization!

Results

Tesla P100 SMX2 16GB

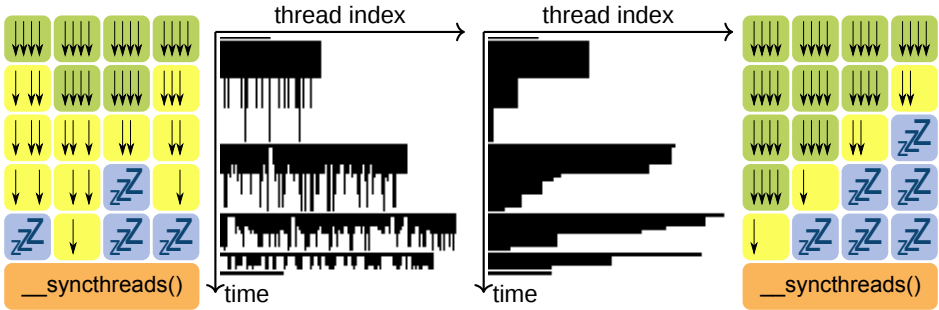


Runtime -60% to +10% – Can we do better?

Balancing

nvprof: High divergence. Stall reason: Synchronization

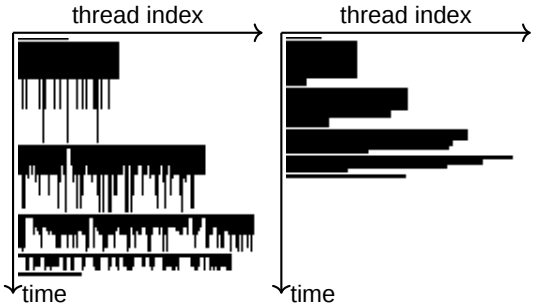
All threads wait on a few long branches



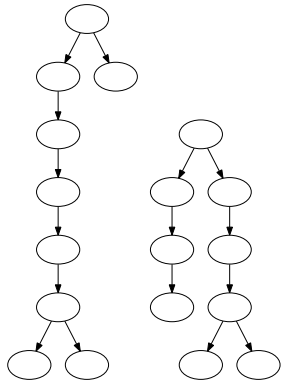
Balancing

Two ways to improve the balancing in each block:

- Cut long branches to reduce divergence.
- Balance tree to reduce depth of tree →less synchronization.



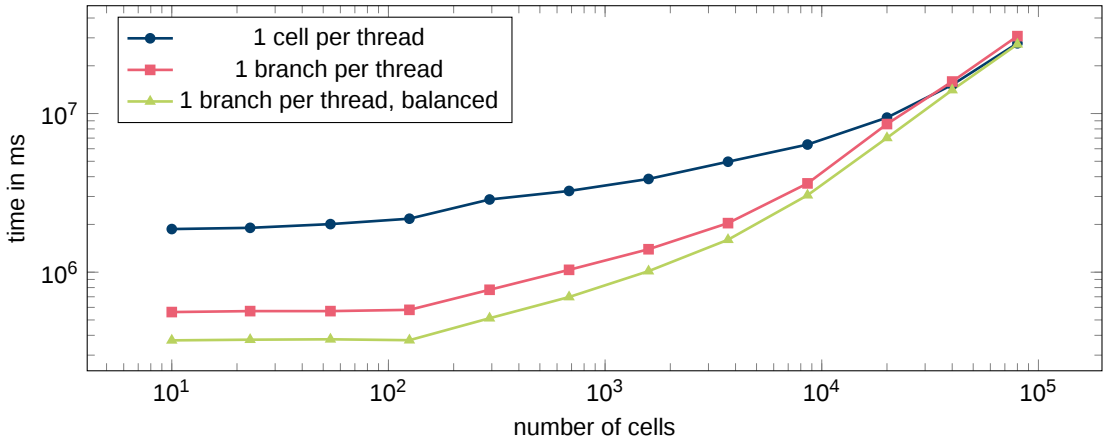
cut long branches



balance tree

Results

Tesla P100 SMX2 16GB



Runtime -80% to -2%

Outlook & Future Directions

- Better heuristics for packing
- Use intermediate packaging to compute heuristics
- Better use of data structures.

Many thanks to

Alexander Peyser (Simulation Lab Neuroscience, FZ Jülich)

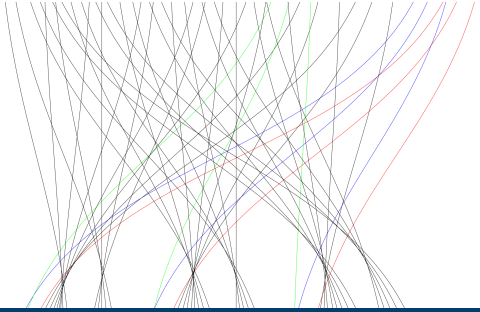
Ben Cumming (CSCS, ETH Zürich)

Anne Küsters (Simulation Lab Neuroscience, FZ Jülich)

References I

[1] Cajal.
Purkinhe cell, 1911.

[2] Collaborators of the Arbor library.
Arbor, high-performance library for computational neurscience simulations.



Efficient Solver for Hines Matrices on GPUs

I packed my bag and in it I put lots of dendrites

September 29, 2018 | Felix Huber | University of Stuttgart