Scan

Parallel Algorithm Design WS21/22

N. Kochendörfer, C. Alles, S. Proß

February 17, 2022
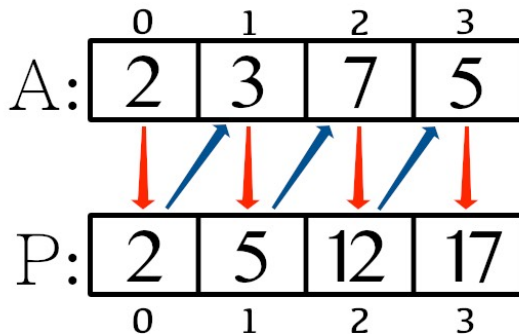
Table of Contents

Scan Theory

- Synonyms: prefix sum, cumulative sum or scan
- inclusive and exclusive version
- further specialization: segmented scan

Inclusive Scan

Prefix Sum
P of A

https://williamrjribeiro.com/?p=132

Inclusive vs. Exclusive scan

https://livebook.manning.com/book/parallel-and-high-performance-computing/chapter-5/v-11/

Segmented Variant

1	2	3	4	5	6	input
1	0	0	1	0	1	flag bits
1	3	6	4	9	6	segmented scan +

https://en.wikipedia.org/wiki/Segmented_scan

Implementation
STL Algorithm

STL provides:

- std::inclusive_scan
- std::exclusive_scan

Essentially equivalent to:

```cpp
float sum = 0;
for(size_t i = 0; i < N; i++)
{
    sum += input[i];
    output[i] = sum;
}
```

$\Rightarrow$ Sequential to a fault!

Implementation
Alternatives

Alternatives to STL:
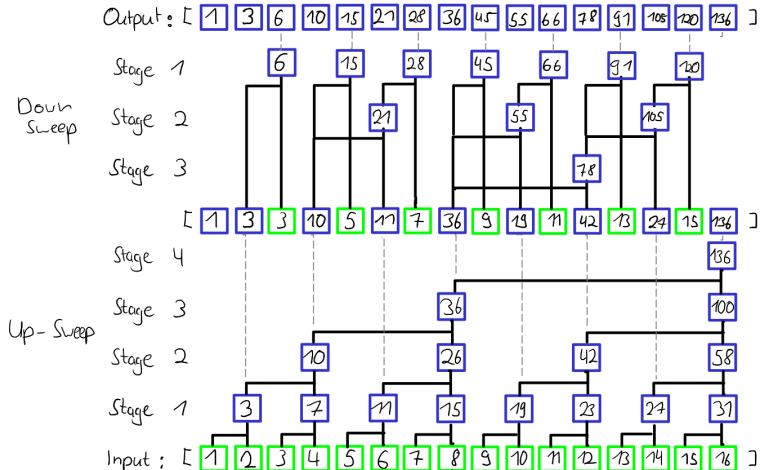
- OpenMP: scan pragma
- TBB: parallel_scan function

Alternative Algorithms:

- Up-Down Sweeping Scan
- Tiled Scan

Up-Down Sweep
Schema Inclusive

Up-Down Sweep

Dependency:

- Only between stages
- $\Rightarrow$ Lots of parallelism

Downsides:

- Workload of 2N
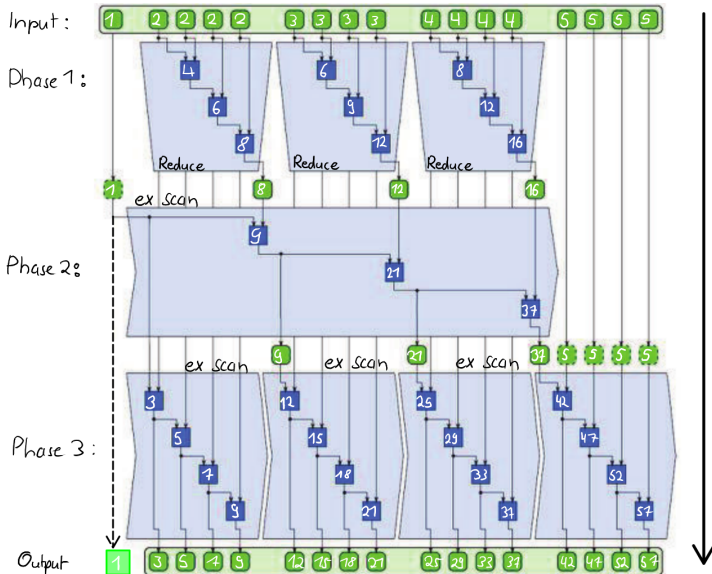- Communication!
- Workload stage dependent!

Tiled Scan

Idea: Process input in independent chunks.

- Each chunk misses previous results
    - $\Rightarrow$ Second pass over data.
- Workload: 2N

Our solution:

- Temporary vector for intermediate sums.
- Only one write to output.
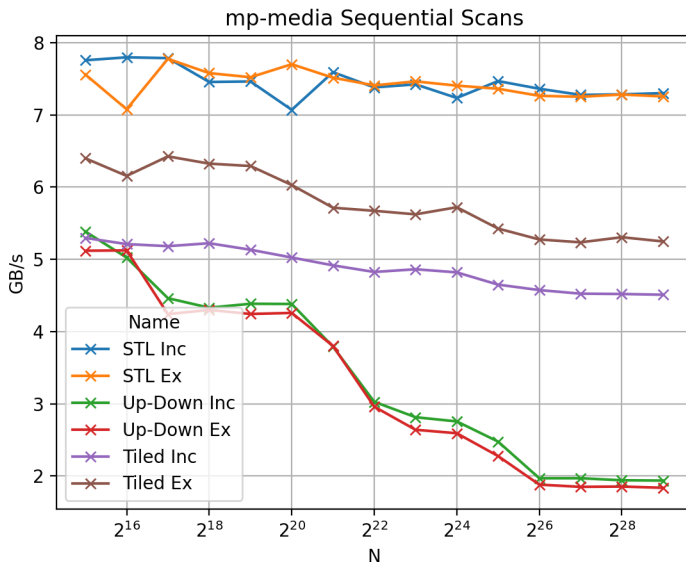
Tiled Scan Schema

Scan

Benchmarks

Parameters:

- In-Place
- Datatype: float
- Values: Linear Distribution. between 0-10.
- Benchmarking Suite: Catch2
- Number of input elements: N

Sequential Scan Results

mp-media Sequential Scans

Segmented Scan
Implementation

- Not present in STL!
- No reference implementations...

Solution: Wrapping the binary operation!

```
[binary_op](PairType left, PairType right){
    PairType new_right = right;
    if (not right.flag)
        new_right.value =
            binary_op(left.value, right.value);
    return new_right;
});
```
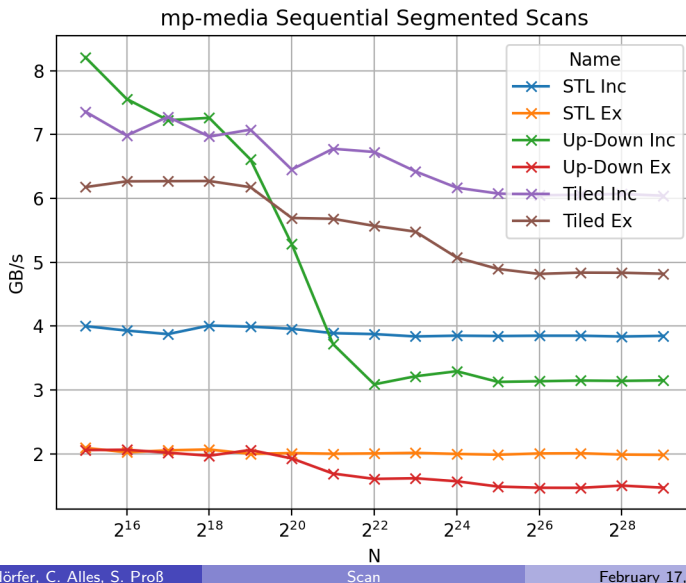
Segmented Scan
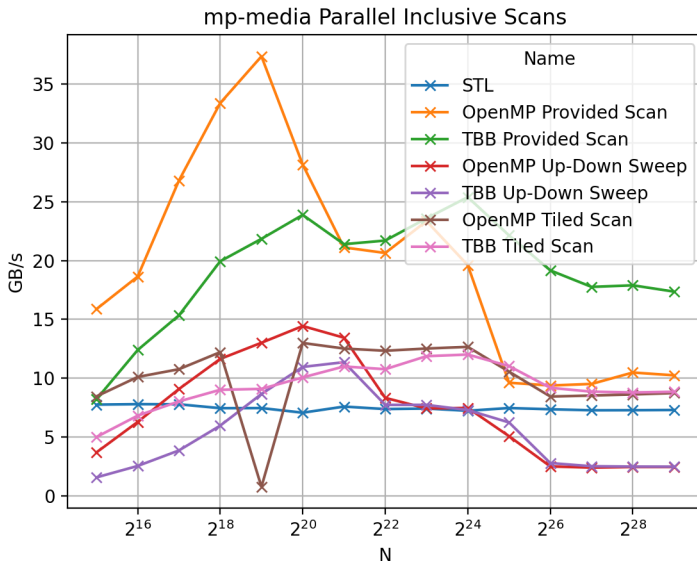Solution

Works for:

- STL Scans
- Most inclusive scans

Challenge: Exclusive Scan

- Exclusive Segmented is complex
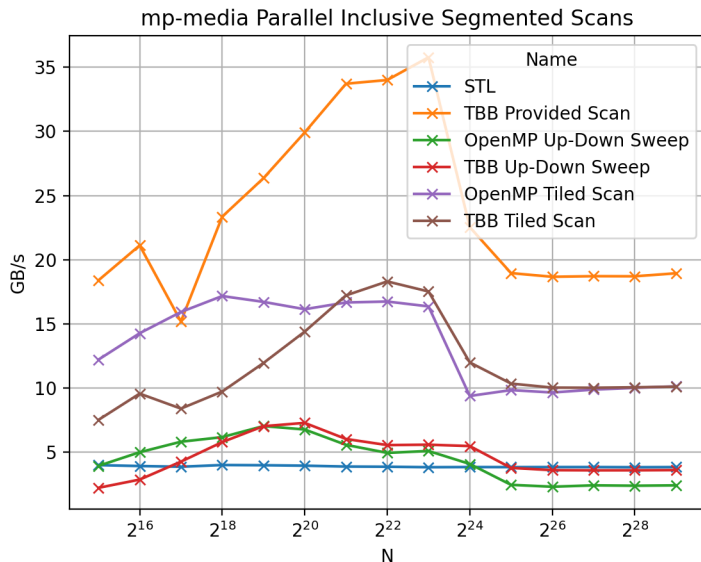- Custom solution for each variant

Sequential Segmented Scan Results

mp-media Sequential Segmented Scans

Parallel Scan Results

mp-media Parallel Inclusive Scans

Parallel Segmented Scan Results

mp-media Parallel Inclusive Segmented Scans

Intermediate Results

Ranking:

1. Library provided implementations
2. Tiled Scan
3. Up-Down Sweeping Scan

Remarks:

- OpenMP $>=$ TBB (if available)
- Up-Down Sweep is slow

Can we do better?

Algorithmic Optimization

Initial Goal: functional correctness.

Algorithmic Optimizations:

- Loop-Fusion:
  - Up-Down Sweep
  - Exclusive Segmented Scan
- Limiting Memory Accesses
- General clean up

Performance gain $\sim$ 1-5 GB/s!

Data Locality

Ensure that the data generated is local to the node:

```cpp
std::vector<float> data(N);
#pragma omp parallel for schedule(static)
for (size_t i = 0; i < data.size(); i++)
{
data[i] = rand();
}
```

- The performance gain by using data local structures is likely to be small due to the warmup of Catch2
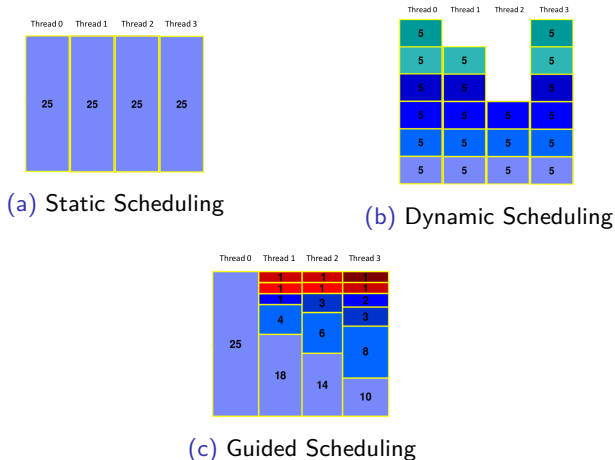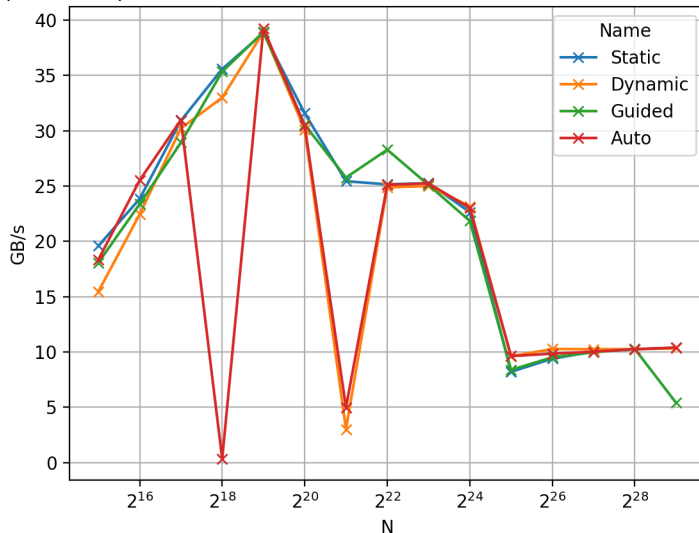
OpenMP Scheduling

(a) Static Scheduling

(b) Dynamic Scheduling

(c) Guided Scheduling

Fig. 1: Different Scheduling Strategies for 100 Iterations and 4 Threads
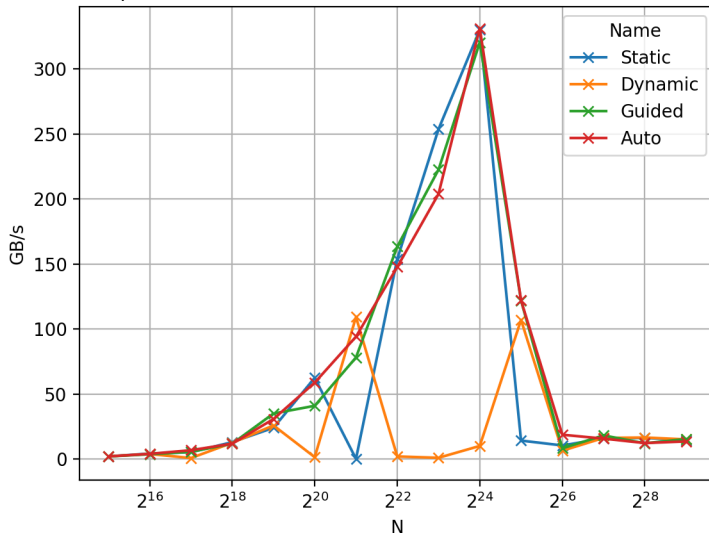
OpenMP Scheduling - Results MP-Media

mp-media OpenMP Provided Inclusive Scan with Different Scheduling (gnu)

OpenMP Scheduling - Results Ziti-Rome

ziti-rome OpenMP Provided Inclusive Scan with Different Scheduling (gnu)
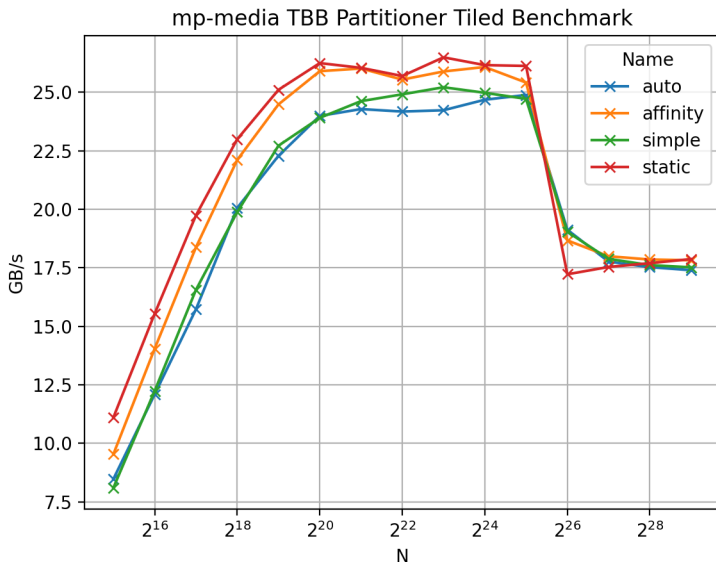
TBB Partitioning

TBB parallel constructs used:
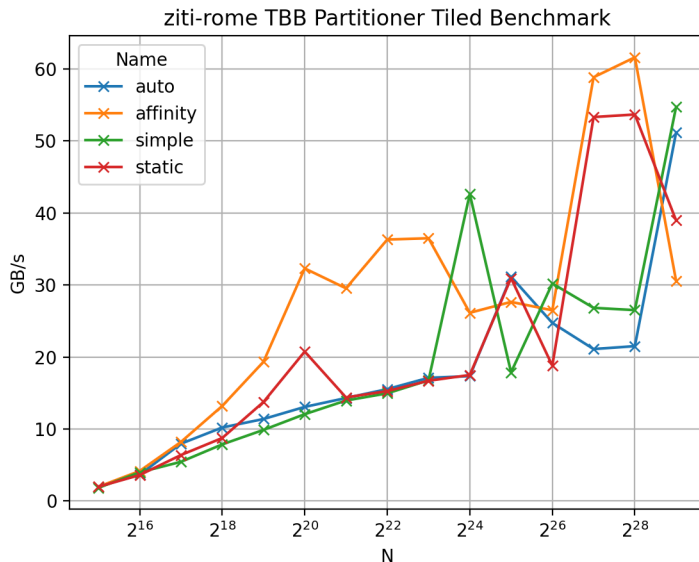
- parallel_scan
- parallel_for

available partitioners:

- auto_partitioner
- affinity_partitioner
- simple_partitioner
- static_partitioner

Performance (inclusive scan)

mp-media TBB Partitioner Tiled Benchmark

Performance (inclusive scan)

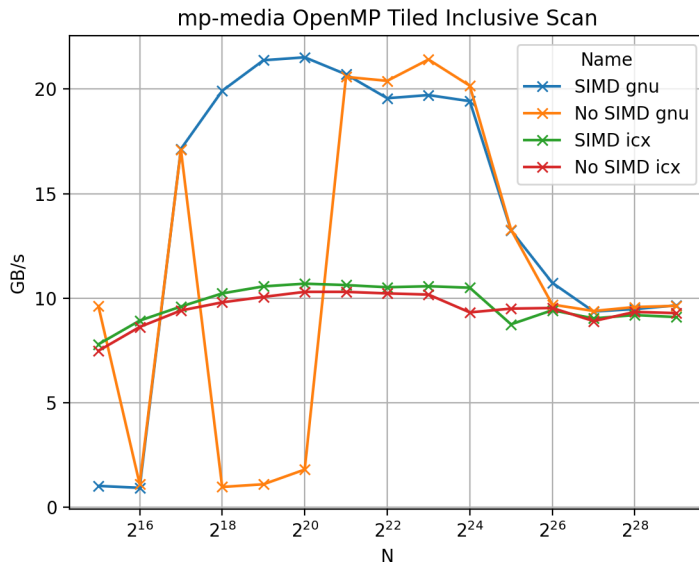ziti-rome TBB Partitioner Tiled Benchmark

Vectorization

Requirements:

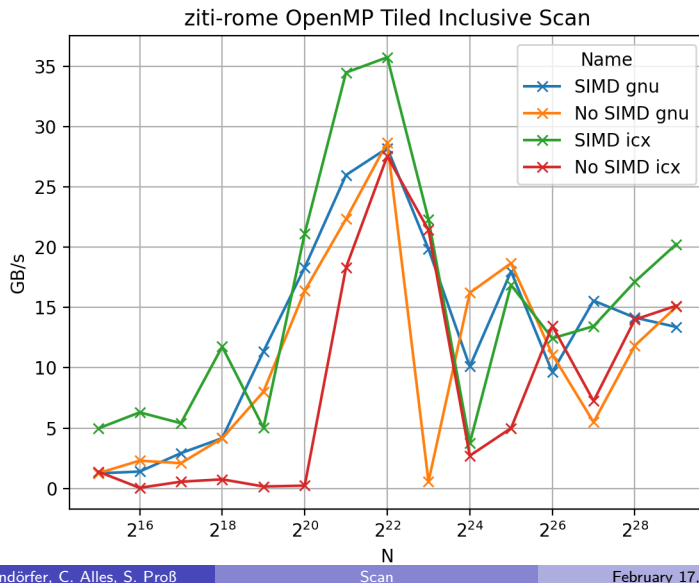- No loop carried dependency
- Loop bounds
- No jumps in code

Realising it:

- #pragma omp simd
- Compiling with -O3
- Using Intel Icx Compiler
  - No OpenMP provided scan!

Vectorization - Results MP-Media

mp-media OpenMP Tiled Inclusive Scan

Vectorization - Results Ziti-Rome

ziti-rome OpenMP Tiled Inclusive Scan

Summary

Library Provided Scans are fastest

Summary

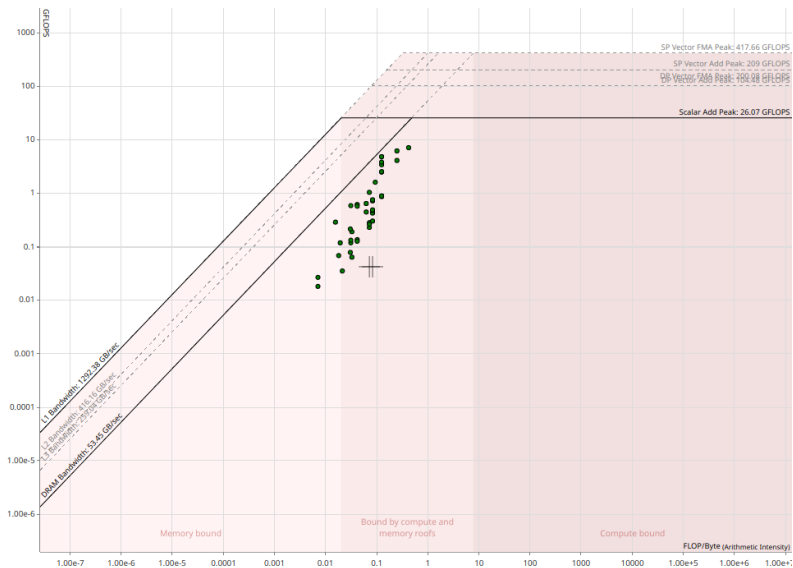Library Provided Scans are fastest

Optimization done:

- Algorithmic
- Data Locality
- Scheduler & Partitioner
- Vectorization

We have

- 4 versions of Scan
- 3 different algorithms
- 2 parallelization libraries + sequential

$\Rightarrow$ 36 Versions to keep track of!

Roofline Mp-Media

Roofline Ziti-Rome