Parallel Sorting in TFORM

Ana Costa Pereira, Josh Davies

26/06/2026

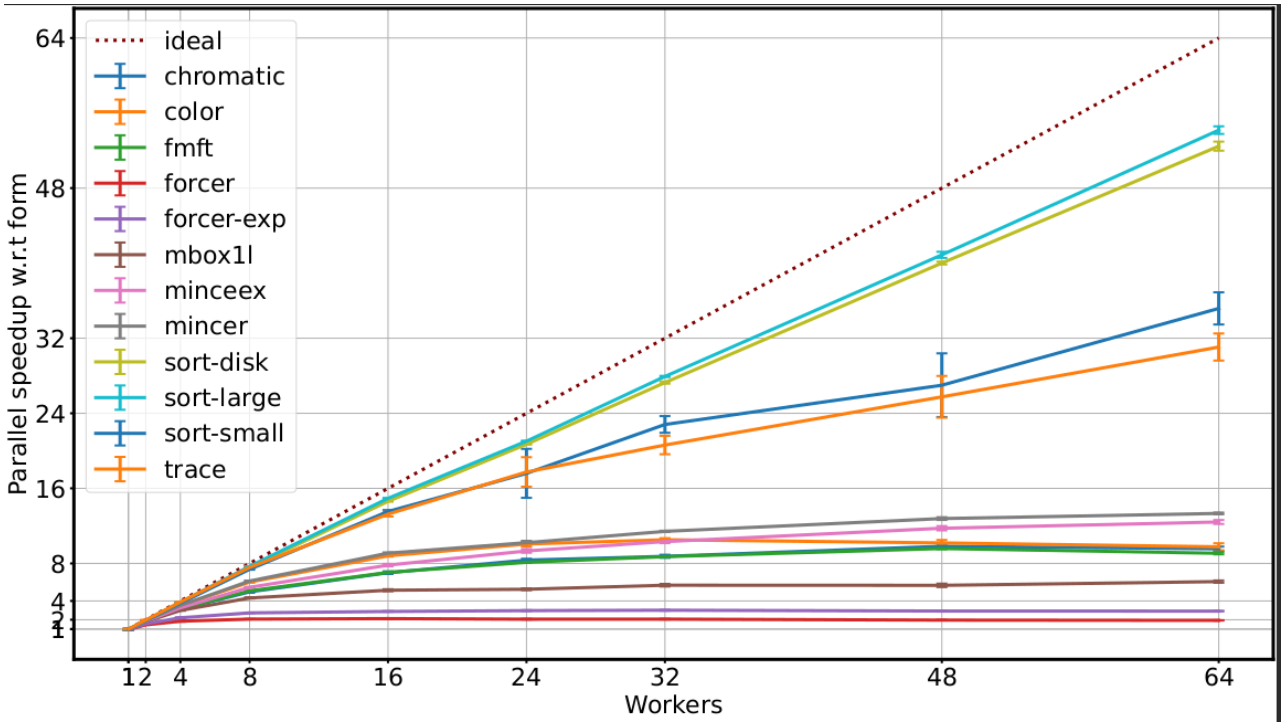
1. SortBot Merging
2. SortBot Bottlenecks and changes
3. Changes in the SortBot Merge
4. SortBot Locking

Outline of talk

Parallel scalings: study of scaling at higher thread number of tform;

- Overview of sorting process in parallel at the SortBot level;
- Benchmark performence at different thread count;
- Changes implemented at the level of SortBlocks and new benchmark results.

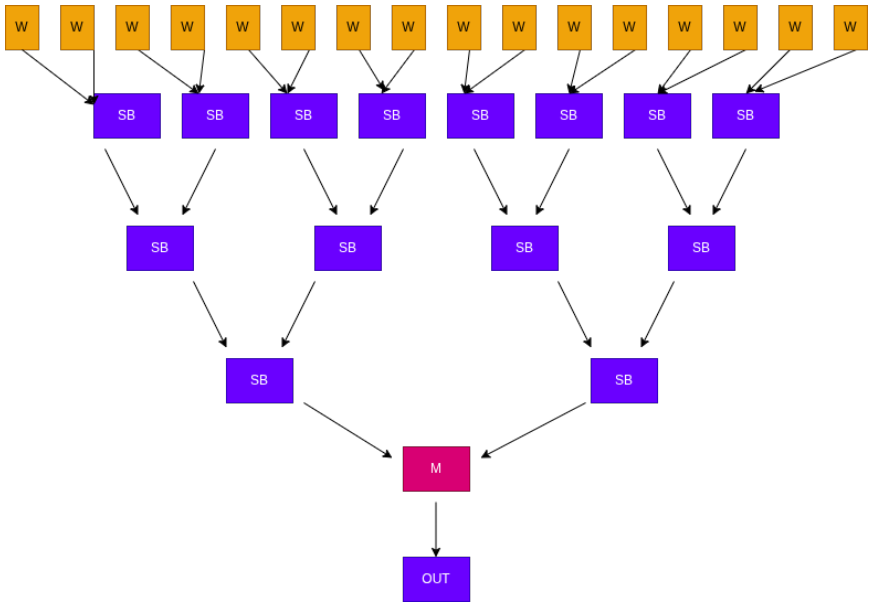
Parallel scaling at higher thread count



- sort-disk, sort-large, sort-small, trace scale up to 64;
- scaling is not good for other tests for > 16 workers.

SortBot Merging

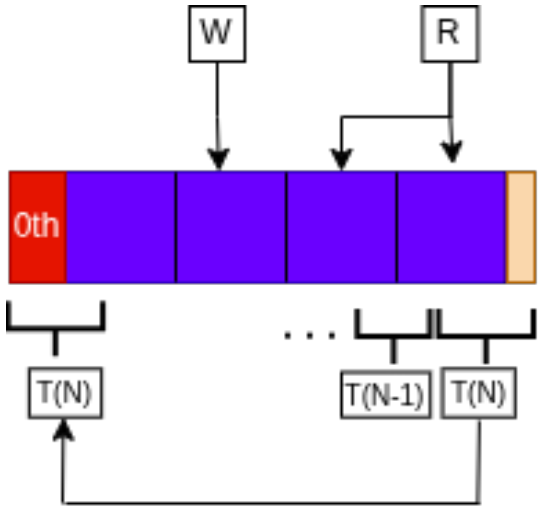
- Master $\longrightarrow$ distributes terms to the workers;
- Worker $\longrightarrow$ independent stream of sorted terms;
- N workers $\longrightarrow$ $N-2$ SortBots are created;
- SortBot routine starts;



- Master performs the final merge and writes output.

SortBot: Overlaping of terms

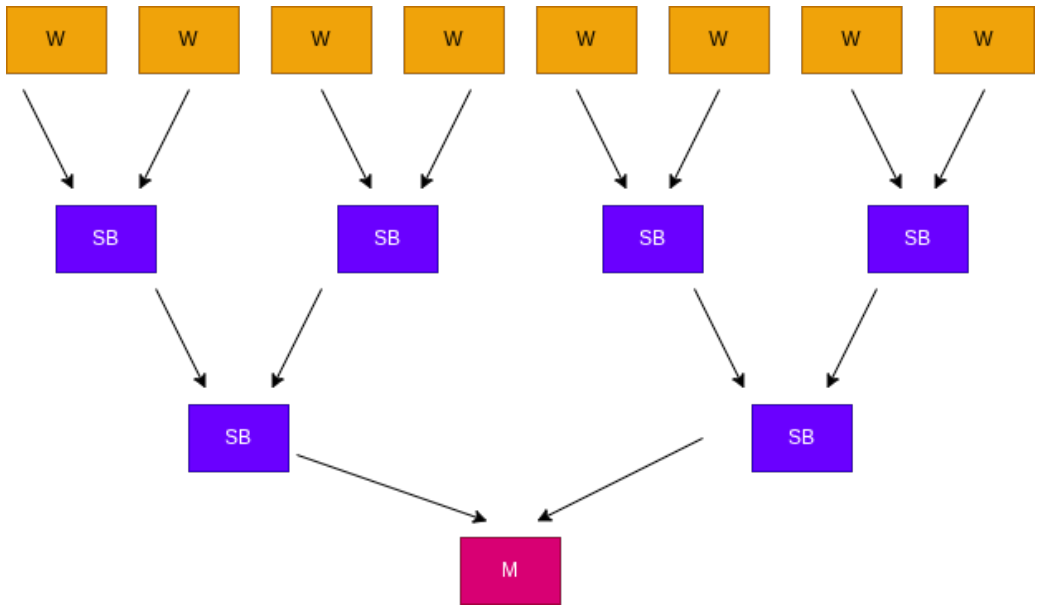
- SortBot $\longrightarrow$ ring buffer with blocks;



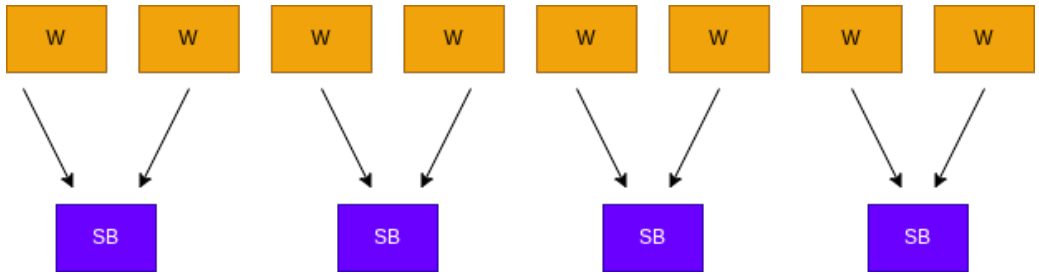
- Terms are allowed to overlap between blocks;
- 0th block - the overlap in the end of term is copied to 0th block so that the terms are contiguous.

SortBlocks Bottlenecks

- **Bottlenecks**: reading of the input with **division of terms** from workers to SortBlocks; inverse tree at the end with the **writing to the file by the** Master;

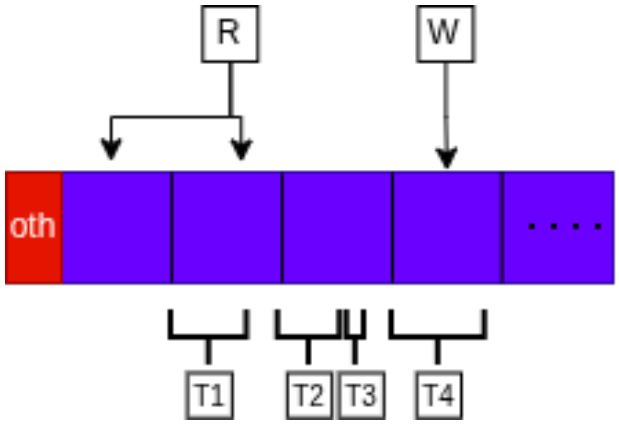


Bottlenecks



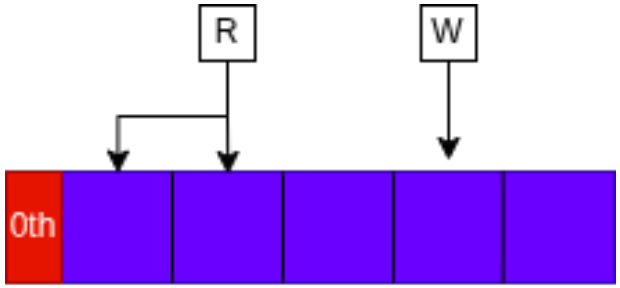
- Default buffer size increased over time $\longrightarrow$ bigger SortBlocks;
- Result from worker fits into **single block** $\longrightarrow$ SortBotTree mechanism does not parallelize
- If a **term** does not fit at the end of a block: Writer fills the SortBlocks all the way up to the end of the block, **overlapping** to the next;
- Once a block is completely full, it reaches the top and moves on to the next block.

Change: SortBot Block size



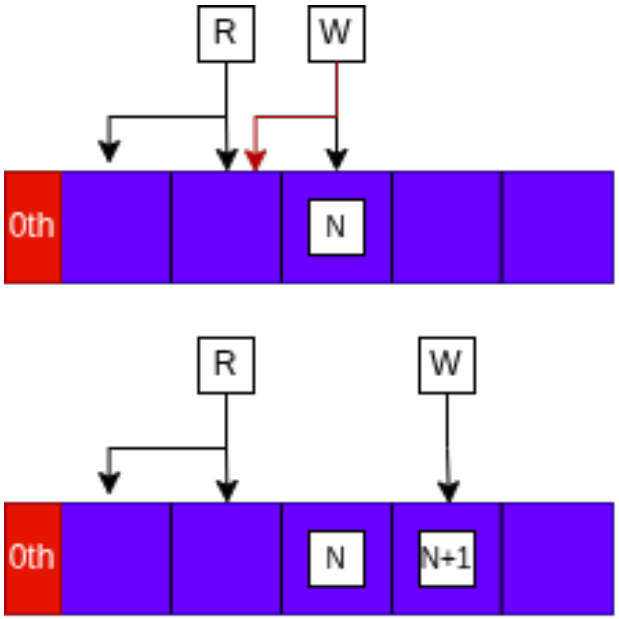
- Require fewer SortBlocks - master buffers not expanded so often;
- Whole terms must fit into the block: if it does not fit, it goes into the next block;

Locking



- Reading thread always locks pair of blocks - the current one and the previous;
- Writing thread locks the block where it is writing;

Change: SortBot Locking



- Writer (W) detects if there is a Reader (R) waiting for it;
- W filling the blocks tries to obtain a lock of the block before the current fill block;
- If W does not obtain the lock, R thread already has the lock - very soon will be done and ready for more terms;
- The W moves to the next block;

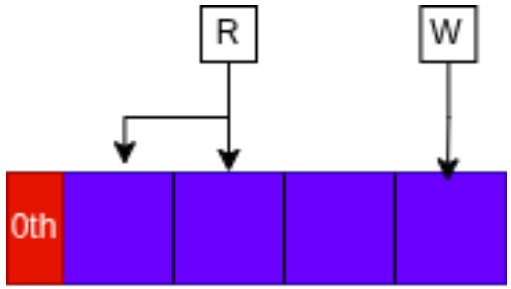
SortBot variables

- The number of blocks in each SortBot has to be at least 4 (due to the locking mechanism) - which corresponds to a minimum of 8 for **NUMBEROFBLOCKSINSORT**;
- There is a minimum number of terms that have to fit the block: **MINNUMBEROFTERMS**;
- Make sure a minimum number of terms is written: never make empty block - **MINWRITENUMBEROFTERMS**;

Final result

PR #831

- **NUMBEROFBLOCKSINSORT** $= 10 \longrightarrow 8$
- **MINNUMBEROFTERMS** $= 10 \longrightarrow 1$
- **MINWRITENUMBEROFTERMS** $= 1$



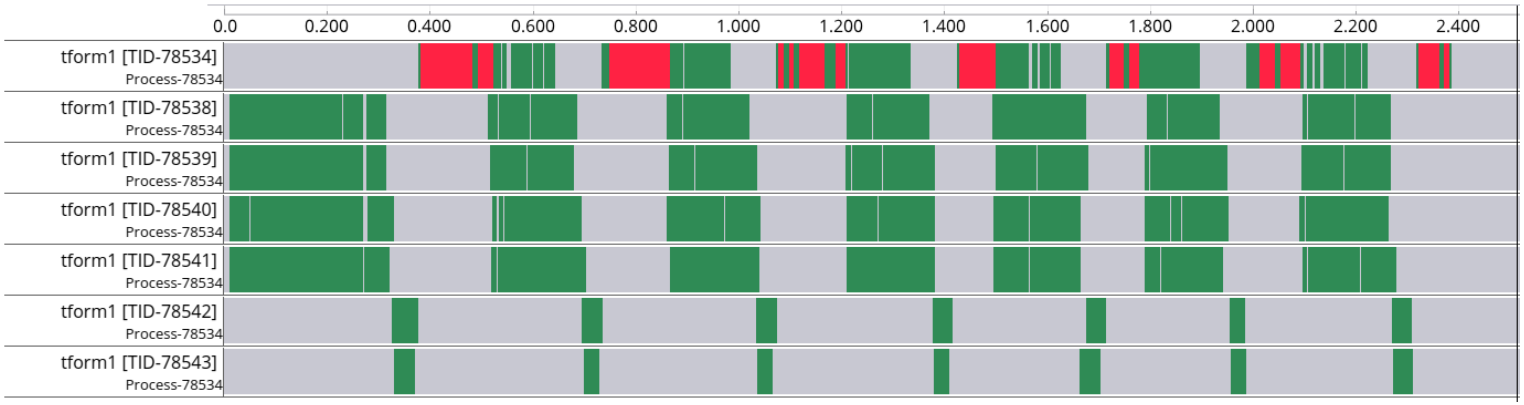
Results after changes

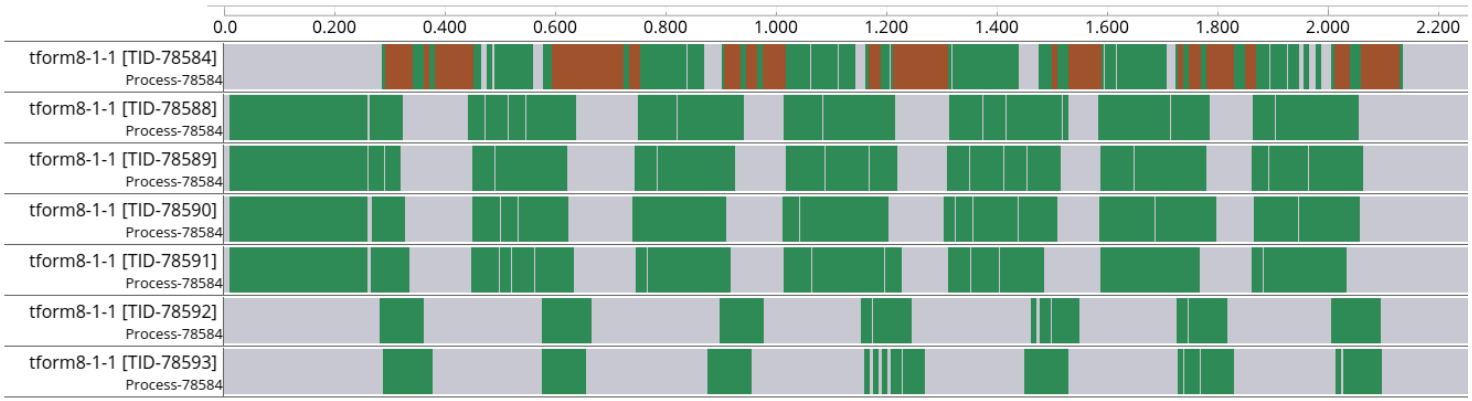
See Josh's talk

- No decrease in performance for any tests;
- Performance improvement for benchmarks where term merging is expensive (PolyRatFun, PolyRatFunExpand)
- Forcer, mbox1l

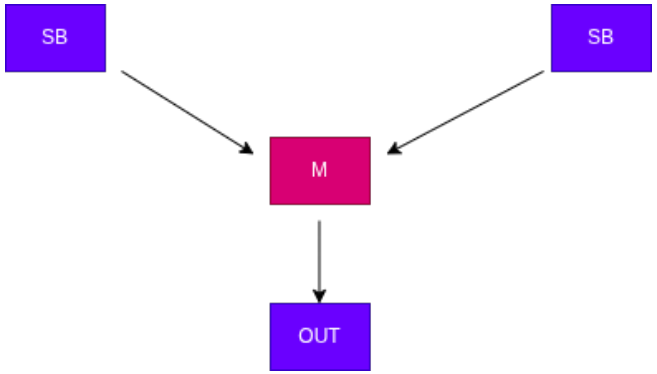
Results after changes: Profiler

- Profiler: AMDuProf with 4 workers;
- Test: generate a large number of terms, mostly sorting them;
- Green: Masters/threads in activity;
- Red: Master using **PutOut**;





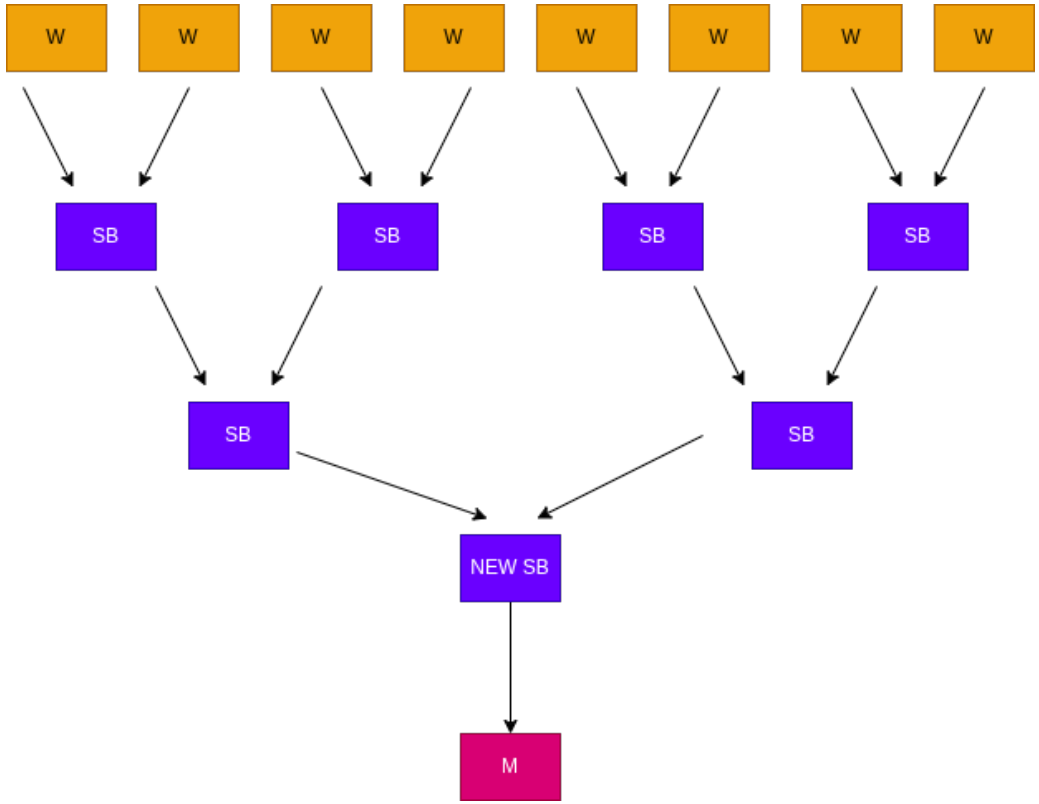
Bottlenecks



- End of the tree: Master performs last sorting as well as writing of terms into the output;
- Performance could be increased by having one last stage of sorting of the last 2 SortBlocks and Master only writing;

Future Work: Addition of new SortBot

- Add a new SortBot at the end of the merging tree to sort the streams that come from last 2 SortBlocks;
- Master only does the writing.



Final remarks

- Parallel performance at higher thread needs to be improved;
- SortBot changes - up to 30% improvement for some tests;
- Current working on improving SortBot parallelization by addition of extra SortBot at the end of the tree.