Distributed Cube and Conquer with Paracooba

Maximilian Heisinger, Mathias Fleury, and Armin Biere

June 26, 2020

Johannes Kepler University Linz, Austria

Talk / Video

4K version of the talk: `https://maximaximal.com/files/?dir=Publications/SAT2020/Full-Resolution`

Splitting of SAT Problems: Cube and Conquer

φ

Splitting of SAT Problems: Cube and Conquer

$$\varphi$$

$$\varphi \wedge x \qquad\qquad \varphi \wedge \neg x$$

Splitting of SAT Problems: Cube and Conquer

$$\varphi$$

$$\varphi \wedge x \qquad\qquad \varphi \wedge \neg x$$

$$\varphi \wedge x, \neg y \qquad \varphi \wedge \neg x, \neg y \qquad \varphi \wedge x, y \qquad \varphi \wedge x, y$$

So What to Expect From a Distributed SAT Solver?

Solve one problem on one thread

So What to Expect From a Distributed SAT Solver?

Solve one problem on ~~one thread~~ multiple threads using one machine

Multithreading!

So What to Expect From a Distributed SAT Solver?

Solve one problem on ~~one thread~~ multiple threads using ~~one machine~~ multiple machines

Distribution (Static Scheduling)!

So What to Expect From a Distributed SAT Solver?

Solve ~~one problem~~ multiple problems on ~~one thread~~ multiple threads using ~~one machine~~ multiple machines

Multi-User!

So What to Expect From a Distributed SAT Solver?

Solve multiple problems on multiple threads using multiple machines *with dynamic scheduling*

This is Paracooba

Features of Paracooba

- Transmission of formulas to all connected compute nodes
- SAT & UNSAT results via up-propagation through the cube tree
- Inbuilt splitting heuristics & local CaDiCaL solving support

Communication Protocol

- TCP connection-based with automatic reconnects
- Supports temporary switching to UDP if connection fails
- Enables transfer of CNF files without memory constraints
- Minimal bandwidth requirements through tightly packed binary encoding

Auto Discovery

- Using UDP broadcasts in local subnet
- New daemons can be connected at any time, tasks will be automatically assigned
- TCP connections are established upon discovery

Encoding Positions on the Cube Tree

64 bit unsigned integers identify every position on the cube tree

- 58 bits specify the path
- 6 bits the path length / tree depth

To Offload or not to Offload

- Every compute node stores its available tasks are in a tree-depth-sorted queue
- Tasks with smallest depth are offloaded once another compute node has too few tasks in its queue (higher splitting potential)
- Deeper tasks are handled locally

Gathering Results

SAT

First SAT result is sent to master

UNSAT

Only if all branches are UNSAT, the formula is UNSAT

Workflow

Benchmark splits were produced using MARCH and can optionally be produced by internal splitting algorithms, in case MARCH takes too long for specific problems.

Benchmark Results

Threads t	Nodes n	Wall-Clock Time T_t^n	Speedup T_t^n / T_1^1	Network Speedup T_t^n / T_{16}^1
1	1	23 h 27 min 50 s	1.00	0.05
2	1	9 h 19 min 40 s	2.52	0.14
4	1	4 h 57 min 59 s	4.72	0.25
8	1	2 h 33 min 47 s	9.15	0.49
16	1	1 h 15 min 38 s	18.61	1.00
32	2	31 min 51 s	44.20	2.37
64	4	14 min 18 s	98.45	5.29
128	8	7 min 58 s	176.72	9.49
256	16	5 min 10 s	272.48	14.64
512	32	3 min 22 s	418.17	22.47

Time to solve the `cruxmiter` depending on the number of threads.

Re-Splitting Formulas

- Possibility to directly generate and distribute new cubes on-the-fly (if e.g. solving a leaf takes too long)
- This did not lead to improvements in our benchmarks
- Can lead to problems when just using average solving time: Too many splits are produced, exhausting the 58 bit depth limit

Conclusion

Paracooba participates in the SAT2020 cloud track as the first distributed cube-and-conquer SAT solver.

MIT Licensed Source Code at

https://github.com/maximaximal/Paracooba

Thank You

Thank you very much (and stay healthy)!