Karlsruhe Institute of Technology

Institute of Theoretical Informatics
Am Fasanengarten 5
76131 Karlsruhe
https://www.algo2.iti.kit.edu

European Research Council
Established by the European Commission

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 882500).

Scalable SAT Solving on Demand

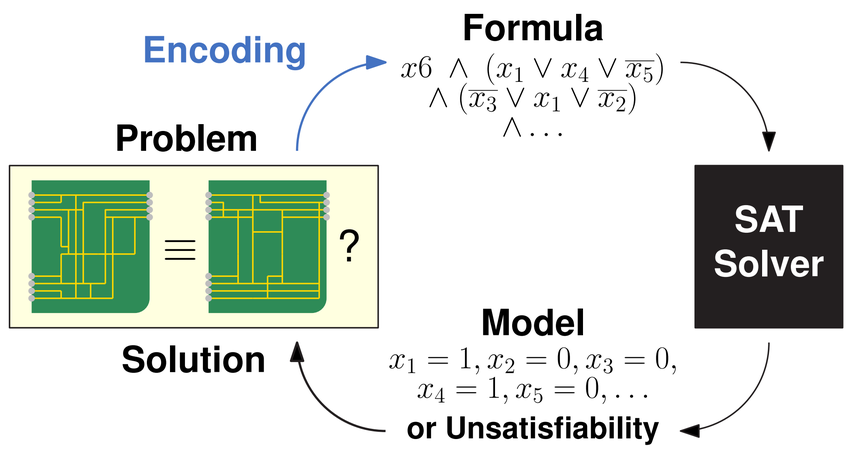
Dominik Schreiber and Peter Sanders • Highlights of Parallel Computing • Nantes, June 2024

0. Background

The Propositional Satisfiability Problem

Given a **propositional formula** F, assign a value (`true` or `false`) to each of its variables in such a way that F evaluates to `true`, or report that such an assignment ("model") does not exist.

— Solving Problems via SAT —



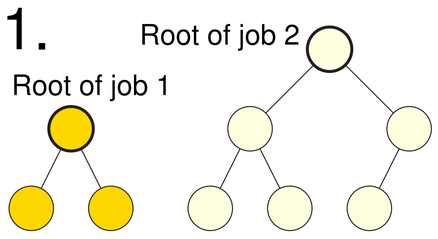
Applications: Software & hardware verification, electronic design, cryptography, theorem proving, scheduling, . . .

I. Decentralized Job Scheduling [1, 2, 3]

- Exploit **large clusters (1000 – 10 000 cores)** for SAT solving
- Substantially **reduce scheduling latencies** for interactive solving
- Combine job parallelism and parallel SAT solving

— Our Approach —

1.



Represent jobs as **binary trees** of processes

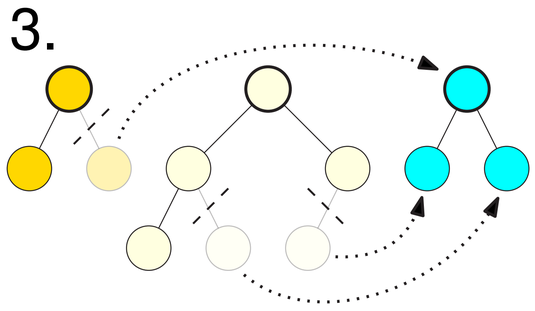
2.

Demand:	3	9	3
Priority:	1	2	8

Calculate fair job volumes

⇒ Volume: **2** **4** **3**

3.

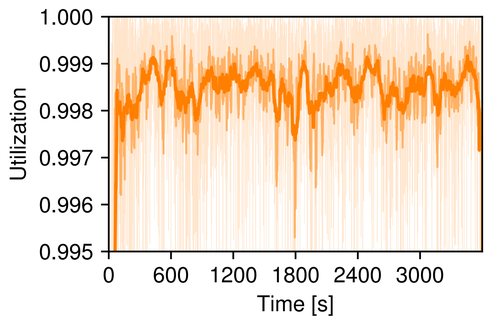


(Re-)assign processes to jobs according to fair volumes, using:

1. **Random walks** of requests
2. Guiding requests along distributed **routing tree** (better)
3. **Asynchronous prefix sums** for request-idle matching (best)

— Results —

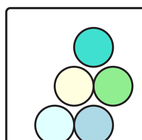
- Experiments at $\leq$ **6144 cores**
- Scheduling and resizing of jobs **within $\approx$ 10 milliseconds**
- **Near-optimal utilization**
  Right: moving avg. over 1/15/60 s
- Less than 10% of workers are rescheduled more than once



II. Distributed SAT Solving [2, 4]

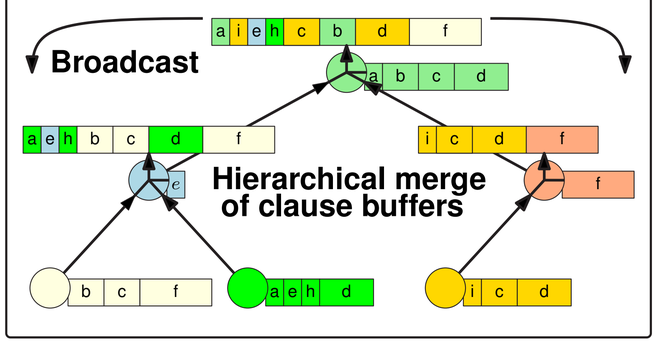
- Improve **scaling behavior** of distributed SAT solving
- Use flexible communication that supports **fluctuating resources**

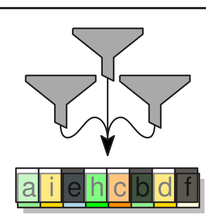
— Our Approach —



Portfolio approach [5]: Run diverse variants of sequential **CDCL solvers**, which produce useful **conflict clauses**

Clause sharing: Exchange **globally best distinct clauses** across solvers



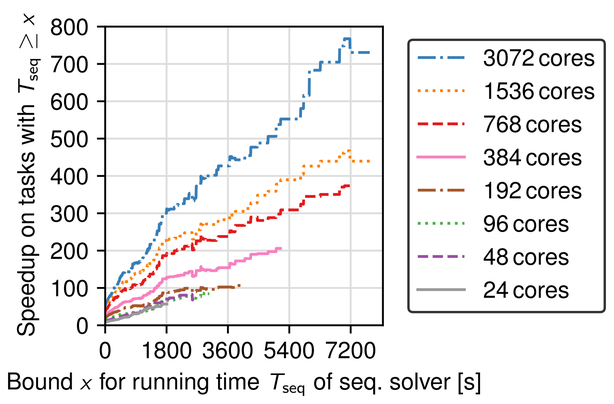


Clause sharing quality control via

– careful **steering** of sharing volume;
– exact **distributed filtering** of repeated / obsolete clauses;
– manipulating clause **quality scores** (LBD)

— Results —

- Speedups observed for **up to 3072 cores**, doubling prior speedups
- Clause sharing is demonstrably **main driver of scalability**
- Can conquer **previously infeasible instances**



III. The MALLOB Platform [3, 6]

- C++ UNIX application built on **MPI** featuring all of our contributions
- Winning system of **International SAT Competition** Cloud Tracks 2020–2023
- LGPL and MIT licensed code available

`github.com/domschrei/mallob`

— References —

[1] Peter Sanders and Dominik Schreiber. Decentralized online scheduling of malleable NP-hard jobs. In *Proc. Euro-Par*, pages 119–135. Springer, 2022.

[2] Dominik Schreiber and Peter Sanders. Scalable SAT solving in the cloud. In *Proc. SAT*, pages 518–534. Springer, 2021.

[3] Dominik Schreiber. *Scalable SAT Solving and its Application*. PhD thesis, Karlsruhe Institute of Technology, 2023.

[4] Dominik Schreiber and Peter Sanders. MALLOBSAT: Scalable SAT solving by clause sharing. *JAIR (to appear)*, 2024.

[5] Tomáš Balyo, Peter Sanders, and Carsten Sinz. Hordesat: A massively parallel portfolio SAT solver. In *Proc. SAT*, pages 156–172. Springer, 2015.

[6] Peter Sanders and Dominik Schreiber. Mallob: Scalable SAT solving on demand with decentralized job scheduling. *JOSS*, 7(76):4591, 2022.