Parallel SAT Solving
To Share or Not To Share

Armin Biere
Johannes Kepler University
Linz, Austria

Theoretical Foundations of Applied SAT Solving

Schloss Dagstuhl
Leibniz Zentrum für Informatik
Wadern, Germany

Thursday, 23 April, 2015
Parallel Computers

- **Multi-Core CPUs**
  - CPU frequency scaling stalled but Moore’s Law still holds
    - [http://ce.cs.jku.at/events/informationsveranstaltung-computational-engineering-slides/ParallelComputing.pdf](http://ce.cs.jku.at/events/informationsveranstaltung-computational-engineering-slides/ParallelComputing.pdf)
  - number of cores per processor is increasing (8-96)

- **GPU (graphic processing units)**
  - thousands of (dumb) cores
  - focus on data processing (games)
  - 10x-100x more memory throughput than CPU (like 5GB/sec vs 200GB/sec)
  - you need to program 1st level cache explicitly (CUDA)

- **Cluster**
  - also known under the notion of grids
  - cheap, available

- **Cloud**
  - even more cores
  - scalability in terms of allocating less/more resources
  - slight focus on massive data processing (like map reduce algorithms)
Challenge

- we want to solve even harder problem than those we can solve today
- we do have (easy) access to parallel computers
- how to parallelize SAT?
- how to get speed-up (sequential divided by parallel wall clock time)
  - current model in HPC (high performance computing): hero programmer
  - fight between correctness and efficiency (lock vs no-lock)
- different strategies for different parallel computers
Why am I standing here?

- developed parallel solvers Plingeling, Treengeling
  - Plingeling
    - portfolio solver (makes use of the 321 options of Lingeling)
    - technically it only uses call backs from the core Lingeling library
    - sharing of units + equivalences + short clauses
  - Treengeling
    - (concurrent) cube & conquer solver
    - portfolio component sharing of units and refuted cubes
- Cube & Conquer [HeuleKullmannWieringaBiere’11]
  - use look-ahead SAT solver to produce cubes
  - solve those cubes in parallel with CDCL
- won several first places in recent competitions
  - next slide: parallel application track 2014 with time-out 1000 sec (4 cores)
  - http://satcompetition.org/edacc/sc14
plingeling versus lingeling
Levels of Parallelization (in SAT Solving)

- **Service Level** [Cloud, Cluster, Multi-Core]
  - cloud/cluster provider offers compute resources specialized to SAT/SMT/MC…?

- **Application Level** [Cluster, Multi-Core]
  - solve and schedule multiple similar or related problems in parallel
  - in HW model checking quite common (one RTL model + dozens of properties)

- **Portfolio Level** [Multi-Core maybe Cluster]
  - run different solvers (or solver configuration) in parallel
  - share information (clauses, units, equivalences, …)

- **Engine Level** [Multi-Core]
  - use different algorithms which support each other, e.g., pre/inprocessing
  - originally sequential, can (easily?) be parallelized, results shared

- **Search Level** [Cloud, Cluster, Multi-Core]
  - search space splitting, e.g., guiding path, cube & conquer, sharing is hard

- **Implementation Level** [Multi-Core, GPUs]
  - parallel BCP […], use parallel thread for clause minimization [Wieringa…]
  - parallelize CDCL analyze and BCP [unpublished but also does not really work]
Parallel SAT Solving

- dominating approach: portfolio with clause sharing
  - ManySAT, Plingeling, Penelope, …
  - successful in the application track of the competition
  - portfolio already gives substantial speed-up
  - clause sharing of “good” clauses gives another boost

- search space splitting
  - originally used on clusters / grids
    - guiding path principle [ZhangBonacinaHsiang’96]
    - revisited and extended recently [HyvärinenJunttilaNiemelä’10]
  - can be combined with look-ahead
    - Cube & Conquer approach [HeuleKullmannWieringaBiere’11]
    - works well on multi-core as well
    - Treengeling won parallel combinatorial track in SAT Competition 2013/14

- how to merge these two approaches?

- scalability for many cores and larger clusters / grids / cloud
Data Flow Algorithms for SAT

- most paradigms for SAT solving are control-dominated:
  - such as variants of CDCL, WalkSAT, or Look-Ahead based algorithms
  - hard to port to highly parallel computing architectures like:
    - bit-parallel operations on streaming units (SSE, AVX ops with 128 bit - 256 bit)
    - multi-core systems with say 96 or even more cores
    - clusters / grid / clouds with 128 - 100000 cores
    - GPUs with more than 2000 cores
  - control flow dominated algorithms have a hard to time to achieve memory locality

- conjecture is that data-flow orientation allows memory locality
  - challenge is to come up with SAT algorithms organized around data-flow
  - find other ways to change algorithms / machines to become more “local”

- our experiences with bit-parallel SAT and GPU’s are rather negative
  - only focused on preprocessing sofar
  - positive effect for few crafted instances, usually way slower

see Master thesis by Robert Aistleitner
parallelize SAT for solving harder problems
parallelize SAT to economically make use of available HW
we are just at the beginning of making parallel SAT work
talk by Asish at Banff: proof span = computational span = parallelizability
I think we need totally new algorithms (which is quite exiting)
just got 4 years of funding for parallel SAT solving (Post-Doc sought)

see also Dissertation Norbert Manthey, particularly, pages 225ff