pArgoSAT – Parallelization of Boolean Constraint Propagation in DPLL-based SAT solvers

Milan Banković Filip Marić
{milan,filip}@matf.bg.ac.rs

Department of Computer Science
Faculty of Mathematics
University of Belgrade

5th Workshop on Formal and Automated Theorem Proving and Applications. Belgrade 2012.

Outline

Introduction

Our approach to parallelization:

- targets multiprocessor systems with shared memory (including modern multi-core processors and other SMP systems)
- exploits the facts that most of the processing time in modern SAT/SMT solvers is spent in boolean constraint propagation (BCP) and theory propagation (over 80% for DPLL-based SAT solvers)
- event-driven system organized around thread pools processing the events in a parallel fashion
- events correspond to changes of the solver's state (e.g. adding literals to the assertion trail)
- each event is processed by executing appropriate tasks.

Related work

Related work:

- Similar approaches exist, but are not exploited as other parallelization approaches (divide-and-conquer, portfolio)
- Zhao, Moskewicz, Madigan, Malik: Accelerating boolean satisfiability through application specific processing. 2001.
- Norbert Manthey. Parallel SAT Solving - Using More Cores. 2011. (based on clause set dividing)
- SMT not considered.

Basic architecture

The main parts of the solver:

- dispatcher (handles decides, conflict analysis, backjumping, learning, forgeting, restarting)
- a theory solver for each theory (handles conflict detection, theory propagation and explaining)
- a special case: boolean theory, defined by clauses of the formula (BCP being its theory propagation)
- a thread pool for each theory (consisting of one or more threads processing the events)

Basic architecture

Two level of parallelization:

- high-level: theory solvers for different theories can run in parallel, since their tasks are independant
- low-level: specific theory decision procedures can be further parallelized (must be multi-thread aware)

BCP parallelization

So far, we implemented:

- a simple dispatcher
- a multi-thread aware theory solver for boolean theory, based on two-watched-literals scheme
- adding a literal to the assertion trail triggers the processing of the corresponding watch-list
- if more than one thread is in the pool, multiple literals (watch-lists) can be processed at the same time

Experimental results

	Computer 1		Computer 2		Computer 3	
	1	2	1	2	1	2
Instance 1	37s	21s	10s	18s	25s	23s
Instance 2	13m44s	7m21s	3m50s	7m48s	9m4s	8m31s
Instance 3	28s	20s	12s	31s	21s	18s
Instance 4	12s	8s	3s	6s	9s	7s

Further work

Our plans:

- implementation of missing features in the dispatcher
- implementation of other theory solvers (to become an SMT solver)
- further optimization and evaluation

THANK YOU :)