Deterministic Parallel DPLL $(DP)^2LL$

Youssef Hamadi [1,2] Said Jabbour [3] Cédric Piette[3] Lakhdar Saïs[3]

[1] Microsoft Research
7 J J Thomson Avenue
Cambridge, United Kingdom

[2]LIX École Polytechnique, F91128 Palaiseau, France

[3]Université Lille-Nord de France, Artois, F-62307 Lens
CRIL, F-62307 Lens
CNRS UMR 8188, F-62307 Lens

youssefh@microsoft.com {jabbour,piette,sais}@cril.fr

Pragmatics of SAT – POS'11

june, 18 2011

//SAT : some keys

2 categories of approaches

- Divide and conquer
  - Incrementally divides the search space into subspaces
  - Each subspace is allocated to a sequential search worker
  - Needs load balancing strategies
- Portfolio
  - Exploits the complementarity between different sequential DPLL strategies
  - Each worker deals with the CNF given in input
  - Cooperation betweens workers (*nogood* exchanges, heuristical values, etc.)
  - Load balancing strategies not needed

Unfortunately, for both categories...

// Solvers exhibits a non deterministic behavior:

- in term of reported solutions (if SAT)
- in term of refutation proofs (if UNSAT)
- in term of runtimes

Non deterministic behavior of ManySAT

instance	#vars	#models (diff)	n$\bar{H}$	avg time (σ)
12pipe_bug8	117526	10 (1)	0	2,63 (53.32)
ACG-20-10p1	381708	10 (10)	1.42	1452.24 (40.61)
AProVE09-20	33054	10 (10)	33.84	19,5 (9.03)
dated-10-13-s	181082	10 (10)	0.67	6.25 (9.30)
itox_vc1138	150680	10 (10)	26.62	0.65 (22.99)
md5_47_4	65604	10 (10)	34.8	173,9 (31,03)
md5_48_1	66892	10 (10)	34.76	704.74 (74.65)
md5_48_3	66892	10 (10)	34.16	489.02 (68.96)
safe-30-h30-sat	135786	10 (10)	22.32	0.37 (0.79)
total-10-19-s	331631	10 (10)	0,5	5.31 (6.75)
UCG-20-10p1	259258	10 (10)	2,12	768.17 (31.63)
vmpc_28	784	10 (2)	3.67	34,61 (25.92)
vmpc_31	961	8 (1)	0	583.36 (88.65)

About determinism...

Determinism is the scientific principle that states that for everything that happens there are conditions such that, given them, nothing else could happen.

What about //SAT ?

- Non-reproducibility limits the deployment of SAT parallel technology (e.g. formal verification: reported bugs cannot be reproduced)
- Non-reproducibility makes difficult the evaluation of //SAT solvers
- + any procedure using a //SAT solver becomes itselft non déterministic ! (harder to debug, etc.)

Why?

- Phenomena due to the lack of synchronisation between cores, in order to maximize performances
- Determinism and efficiency not compatibles?

Deterministic Parallel DPLL $(DP)^2LL$

Data : a CNF formula $\mathcal{F}$;
Result : *true* if $\mathcal{F}$ is satisfiable; *false* otherwise

begin
  $<$*inParallel*, $0 \leq i < \#core$$>$
    answer[i] = search($core_i$) ;
  for $(i = 0; i < \#core; i{+}{+})$ **do**
    if $(answer[i] != unknown)$ **then**
      return answer[i];
end

Data : a CNF formula $\mathcal{F}$;

Result : $answer[i] = true$ if $\mathcal{F}$ is satisfiable; *false* if unsatisfiable, *unknown* otherwise

```
begin
    nbConflicts=0;
    while (true) do
        if (!propagate()) then
            nbConflicts++;
            if (topLevel) then
                answer[i]= false;
                goto barrier_1;
            learntClause=analyse();
            exporteExtraClause(learntClause);
            backtrack();
            if (nbConflicts % period == 0) then
                barrier_1: <barrier>
                if (∃j|answer[j]! = unknown) then
                    return answer[i];
                updatePeriod()();
                importExtraClauses();
                <barrier>
        else
            if (!decide()) then
                answer[i]= true;
                goto barrier_1;
end
```

Which period to synchronize?

A necessary tradeoff...

- **Frequent synchronisations**: a lot of time is wasted by the cores, waiting for each other
- **Rare synchronisations** : poor dynamics of information exchange (nogood) $\rightarrow$ loss of efficiency

- **Empirically:** some cores reach the barrier "faster" than the others
- **Why?** unit propagation (90 % CPU time) faster w.r.t. some cores
- **Numerous factors:** number of learnt clauses, etc.

Hard to predict unit propagation speed
$\rightarrow$ Use the size of the learnt clauses database (heuristics)

A dynamic synchronization

$period_i^{k+1} = \alpha + (1 - S_i^k) \times \alpha$ where:

- $0 \leq i < \#core$
- α: constant parameter
- $period_i^{k+1}$: sequence of time (in number of conflicts) between synchronization period k and $k+1$ for core i
- $m = max_{\forall i}(|\Delta_i^k|)$, où $0 \leq i < \#core$ size of the largest learnt clauses database
- $S_i^k = \frac{|\Delta_i^k|}{m}$ ratio between the size of learnt clauses database of u_i and m

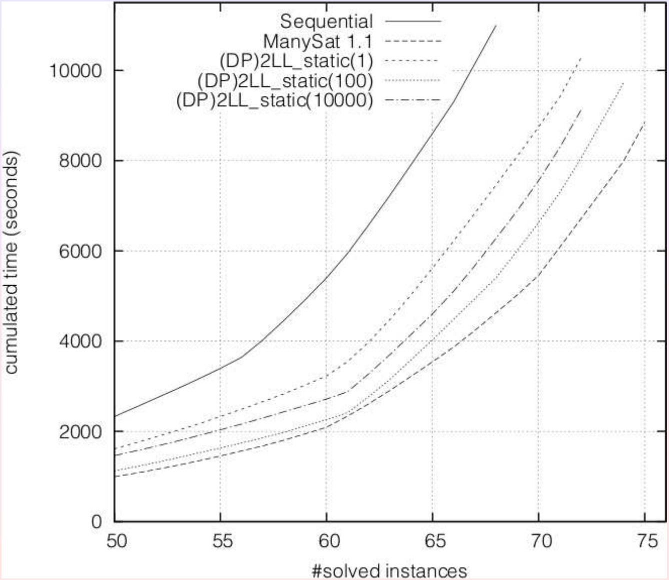
Empirical Environment

- **CPU:** Intel Xeon 3GHz (4 cores)
- **RAM:** 2 GB
- **OS :** Linux CentOS 4.1. (noyau 2.6.9)
- **Cutoff (for each instance) :** 900 seconds
- **Benchmarks:** instances proposed in SAT Race 2010

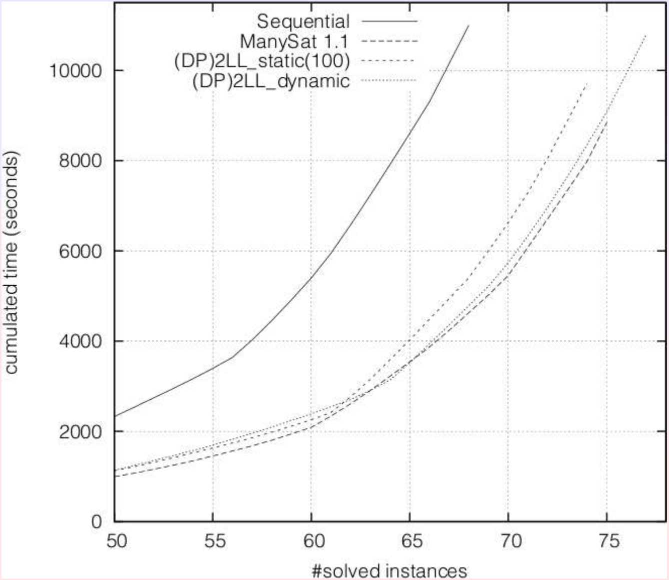
- Deterministic algorithm implemented on top of `ManySAT`[1]

[1] binaries & source code available on the `ManySAT` website

Performance with static synchronization



Performance with dynamic synchronization



Conclusion

In summary...

- Simple but efficient method that enables to make deterministic parallel solvers
- Non negligible waiting time
- Dynamic synchronization approach reduces this waiting time

Future work

Taking better advantage of the synchronization step for new interactions between cores (e.g. resolution between exchanged nogoods, etc.)