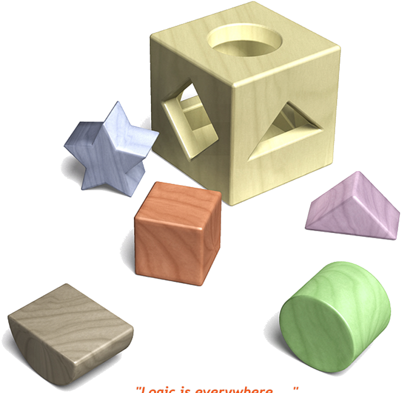
Towards Faster Reasoners by using Transparent Huge Pages

J. K. Fichte, N. Manthey, J. Stecklina, A. Schidler

International Center for Computational Logic
Technische Universität Dresden
Germany

- **Motivation**
- **TL, DR**
- **Preliminaries**
- **THP for SAT**
- **Results**
- **Conclusion**



"Logic is everywhere ..."

Motivation

Motivation

Automated reasoning is widely applied, in many use cases

- **Verification problems**
- **Any-time optimization problems**
- **Time bound nightly analysis/verification**
- **...**

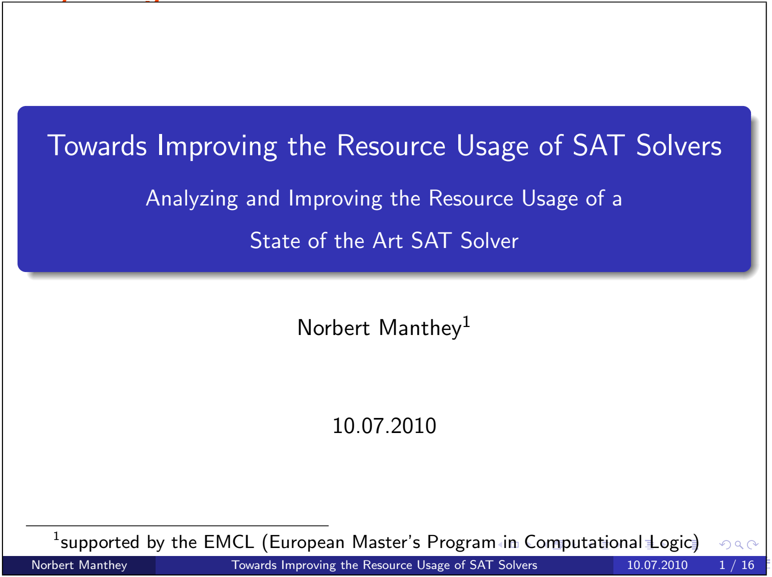
Motivation

Automated reasoning is widely applied, in many use cases

- **Verification problems**
- **Any-time optimization problems**
- **Time bound nightly analysis/verification**
- **. . .**

How to improve all of the above, in an easy to consume way?

History – Pragmatics of SAT 2010



Towards Improving the Resource Usage of SAT Solvers

Analyzing and Improving the Resource Usage of a State of the Art SAT Solver

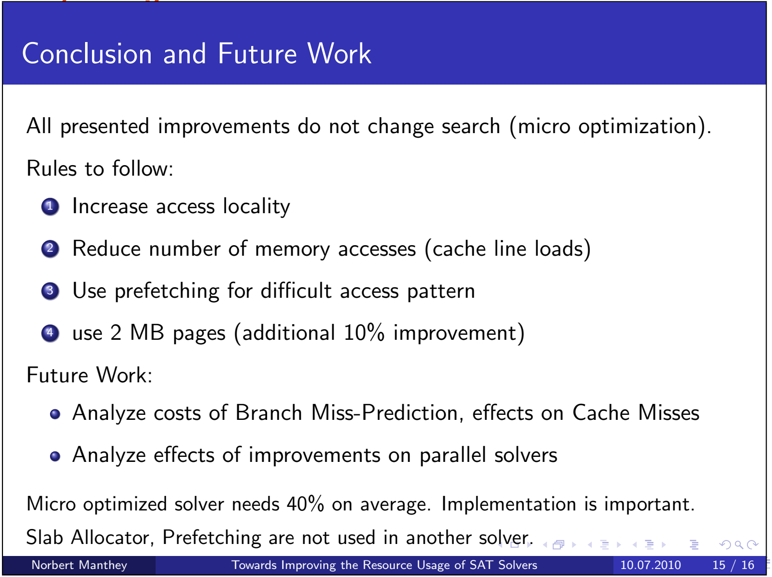
Norbert Manthey[1]

10.07.2010

[1] supported by the EMCL (European Master's Program in Computational Logic)

Norbert Manthey | Towards Improving the Resource Usage of SAT Solvers | 10.07.2010 | 1 / 16

History – Pragmatics of SAT 2010 – Future Work



Conclusion and Future Work

All presented improvements do not change search (micro optimization).

Rules to follow:

1. Increase access locality
2. Reduce number of memory accesses (cache line loads)
3. Use prefetching for difficult access pattern
4. use 2 MB pages (additional 10% improvement)

Future Work:

- Analyze costs of Branch Miss-Prediction, effects on Cache Misses
- Analyze effects of improvements on parallel solvers

Micro optimized solver needs 40% on average. Implementation is important.

Slab Allocator, Prefetching are not used in another solver.

Norbert Manthey | Towards Improving the Resource Usage of SAT Solvers | 10.07.2010 | 15 / 16

TL, DR: Speedup: 10 %

TL, DR: Speedup: 10 % via Transparent Huge Pages

Setup and monitoring:(Ubuntu)

```
cat /sys/kernel/mm/transparent_hugepage/enabled
watch -n 1 "cat /proc/meminfo | grep -i huge"
```

Run with global transparent huge pages enabled:

```
sudo apt install libhugetlbfs-bin
sudo hugeadm --thp-always
$YOUR_TOOL
```

Our Contribution (no root): Recompile your tool statically:

```
git clone https://github.com/conp-solutions/mergesat.git
mergesat/tools/run_in_container.sh \
    mergesat/Dockerfile \
    $STATIC_BUILD
GLIBC_THP_ALWAYS=1 $YOUR_TOOL
```

Preliminaries

SAT Solving

Given: Conjunction of clauses (special cases: Unit, Binary)

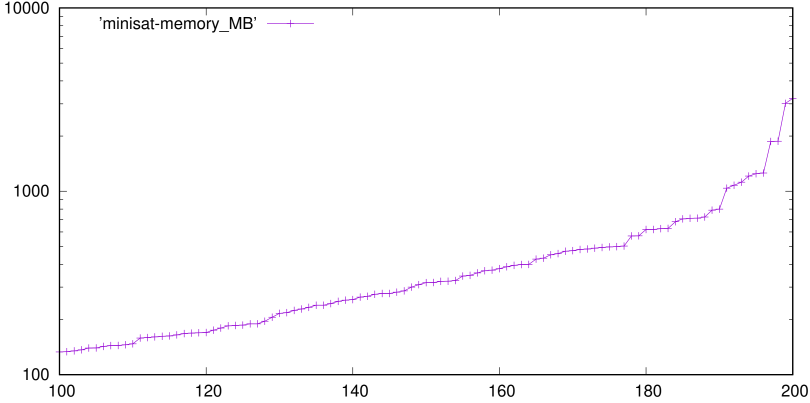
Task: Find satisfying assignment for variables if possible.

Industrial problems: millions of variables and clauses.

Used Techniques:

- **Two-Watched-Literal Unit Propagation (80 - 90%)**
- **Special treatment of binary clauses**
- **Conflict Analysis, Learning and Backjumping, removal (20 - 10%)**
- **Restarts**
- **Heuristics, based on recent search state**
- **Formula simplification**
- **...**

SAT Solver Memory Usage (in MB)



Computer Architecture – Memory Subsystem

- Memory Accesses (Virtual Memory)
  - CPU uses physical address
  - needs to be translated into virtual address
  - page table walk (up to 5 table look ups)
  - TLB cache translation

Computer Architecture – Memory Subsystem

- Memory Accesses (Virtual Memory)
  - CPU uses physical address
  - needs to be translated into virtual address
  - page table walk (up to 5 table look ups)
  - TLB cache translation
- Memory Hardware
  - CPU + Registers
  - L1 data+instruction cache, 64 KiB + 64 KiB
  - L2 cache, 512 KiB
  - LL cache, 4 MiB

Computer Architecture – Memory Subsystem

- Memory Accesses (Virtual Memory)
  - CPU uses physical address
  - needs to be translated into virtual address
  - page table walk (up to 5 table look ups)
  - TLB cache translation
- Memory Hardware
  - CPU + Registers
  - L1 data+instruction cache, 64 KiB + 64 KiB
  - L2 cache, 512 KiB
  - LL cache, 4 MiB
  - TLB, L2 TLB: 4K (or 2M) x 1536 entries = 6 MiB or 3 GiB
  - main memory, +4 GiB

THP for SAT

CDCL Algorithm and TLB Misses

```
UnitPropagate (formula $F$, truth assignment $\tau$, literals $P$, watch lists $L$)

while the list $P$ of literals to propagate is not empty    //closure
    pick $p \in P$, and remove from $P$    //typically DFS
    access watch list $L_p$ of clauses such that $\neg p \in C$    //propagate
    for all clauses $C$ in $L_p$:
        if $C \neq \emptyset$, $x \in C$, and $x$ not falsified in assignment $\tau$
            remove $C$ from $L_p$    //maintain lists
            add $C$ to watch list $L_{\neg x}$ for $\neg x$    //maintain lists
        else if $C = (x)$ unit, extend $P$ and $\tau$ with $x$    //unit rule
        else if $C$ is falsified, trigger conflict analysis($\tau$, $C$)    //conflict
```

CDCL Algorithm and TLB Misses

```
      UnitPropagate (formula $F$, truth assignment $\tau$, literals $P$, watch lists $L$)
B1  while the list $P$ of literals to propagate is not empty        //closure
B2    pick $p \in P$, and remove from $P$        //typically DFS
B3    access watch list $L_p$ of clauses such that $\neg p \in C$        //propagate
B4    for all clauses $C$ in $L_p$:
B5      if $C \neq \emptyset$, $x \in C$, and $x$ not falsified in assignment $\tau$
B6        remove $C$ from $L_p$        //maintain lists
B7        add $C$ to watch list $L_{\neg x}$ for $\neg x$        //maintain lists
B8      else if $C = (x)$ unit, extend $P$ and $\tau$ with $x$        //unit rule
B9      else if $C$ is falsified, trigger conflict analysis($\tau$, $C$)        //conflict
```

- **lines B3 + B4: locations hard to predict, benefits from prefetching, 80% TLB miss**

CDCL Algorithm and TLB Misses

```
    *UnitPropagate* (formula $\boldsymbol{F}$, truth assignment $\tau$, literals $\boldsymbol{P}$, watch lists $\boldsymbol{L}$)
B1  **while** the list $P$ of literals to propagate is not empty      //closure
B2    pick $p \in P$, and remove from $P$      //typically DFS
B3    access watch list $L_p$ of clauses such that $\neg p \in C$      //propagate
B4    **for all** clauses $C$ in $L_p$:
B5      **if** $C \neq \emptyset$, $x \in C$, and $x$ not falsified in assignment $\tau$
B6        remove $C$ from $L_p$      //maintain lists
B7        add $C$ to watch list $L_{\neg x}$ for $\neg x$      //maintain lists
B8      **else if** $C = (x)$ unit, extend $P$ and $\tau$ with $x$      //unit rule
B9      **else if** $C$ is falsified, trigger **conflict analysis($\tau$, $C$)**      //conflict
```

- **lines B3 + B4: locations hard to predict, benefits from prefetching, 80% TLB miss**
- **line B7: location hard to predict, 10% TLB miss**

Use Transparent Huge Pages (THP) Through Recompiling

- Actively using huge pages is difficult
- Always enabled on StarExec (might lead to jitter)
- Kernel supports using huge pages transparently
- Alternatives
  - ~~Modify tool source code~~
  - ~~Modify malloc~~
  - Modify glibc, align to 2M and ask for THP via madvise syscall

Use Transparent Huge Pages (THP) Through Recompiling

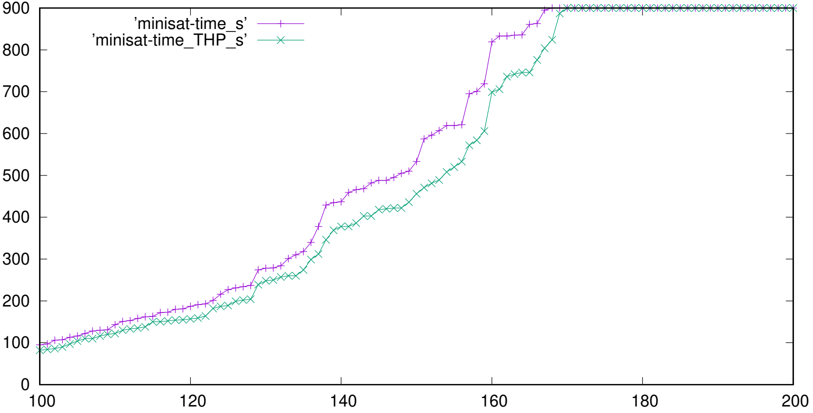
- Actively using huge pages is difficult
- Always enabled on StarExec (might lead to jitter)
- Kernel supports using huge pages transparently
- Alternatives
  - ~~Modify tool source code~~
  - ~~Modify malloc~~
  - Modify glibc, align to 2M and ask for THP via madvise syscall
    - about 500 lines
    - patch to be upstreamed:
      `https://sourceware.org/pipermail/libc-alpha/2020-May/113539.html`

Use Transparent Huge Pages (THP) Through Recompiling

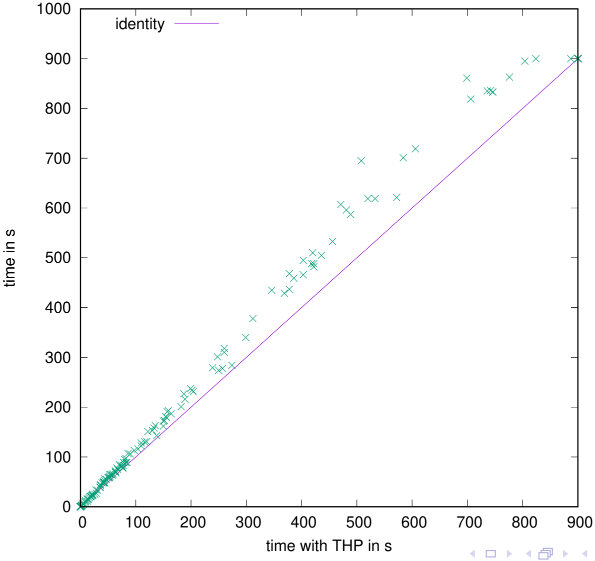
- Actively using huge pages is difficult
- Always enabled on StarExec (might lead to jitter)
- Kernel supports using huge pages transparently
- Alternatives
  - ~~Modify tool source code~~
  - ~~Modify malloc~~
  - Modify glibc, align to 2M and ask for THP via madvise syscall
    - about 500 lines
    - patch to be upstreamed:
      `https://sourceware.org/pipermail/libc-alpha/2020-May/113539.html`
- Try yourself!
  - Compile your solver statically in a docker container
    - `https://github.com/daajoe/thp_docker_build`
  - Enable technique via environment variable
    - GLIBC_THP_ALWAYS=1 minisat cnf/rook-20-0-0.cnf

Results

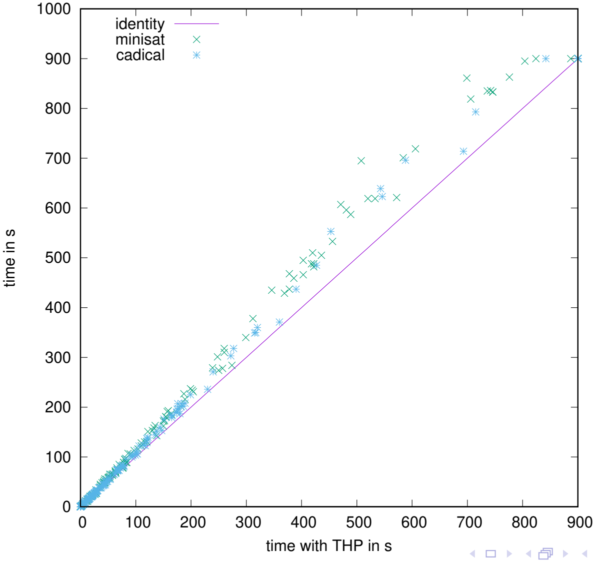
Results – SAT Solver MiniSat Wall Time



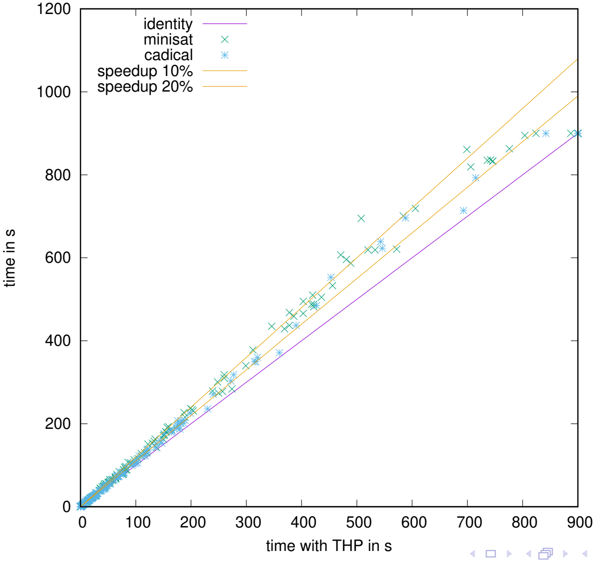
Results – Effect of THP on MiniSat - Scatterplot



Results – Effect of THP on CaDiCaL - Scatterplot



Results – Effect of THP on CaDiCaL - Scatterplot



Results – THP in related domains

Category	Tool	t_n	t_{thp}	s[%]
SAT	MiniSat	8.17	7.03	13.99
SAT	MergeSat	7.94	6.90	13.13
ASP	clasp	3.66	3.29	10.18
MaxSAT	open-wbo	1.19	1.09	8.49
MUS	muser2	4.18	3.97	5.16
HWMC	aigbmc	0.89	0.86	4.11
SWMC	cbmc	0.23	0.22	2.76

- ***t* is PAR1 in hours**
- **Speedup decreases with number of calls to solver, and memory utilization**
- **In public competition:**

HWMCC'19	btormc	45555	43627	4.23

Results – How About Virtual Machines?

- **Results until now for bare metal**

Results – How About Virtual Machines?

- Results until now for bare metal
- VMs: expect higher gain
  - More address translations

Results – How About Virtual Machines?

- Results until now for bare metal
- VMs: expect higher gain
  - More address translations
- Looking for Data?
  - Submitted sequential MERGESAT to parallel track
  - Runs in VM, 2 configurations (with THP)
  - Config of May was buggy, in both submitted configs

Conclusion

Conclusion

- Use THP if you can, it will give you about 10% speedup
- We showed the effect of doing so
- We made huge pages easily accessible

Conclusion

- Use THP if you can, it will give you about 10% speedup
- We showed the effect of doing so
- We made huge pages easily accessible

- What's next?
  - In 2011, Parallelized Unit Propagation

Conclusion

- Use THP if you can, it will give you about 10% speedup
- We showed the effect of doing so
- We made huge pages easily accessible

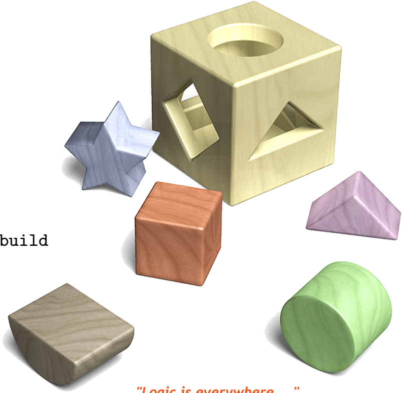
- What's next?
  - In 2011, Parallelized Unit Propagation
  - Get glibc changes applied to upstream code
  - Bring THP into other domains, targeting DFS like algorithms

Towards Faster Reasoners by using Transparent Huge Pages

J. K. Fichte, N. Manthey, J. Stecklina, A. Schidler
International Center for Computational Logic
Technische Universität Dresden
Germany

Thanks

- **Builds with THP**
  - `https://github.com/daajoe/thp_docker_build`
- **Actual GLIBC patches**
  - `https://github.com/conp-solutions/thp`



"Logic is everywhere ..."