

並列 SAT ソルバにおける lemma 共有およびブール制約伝播高速化

大村 圭†

上田 和紀‡

† 早稲田大学大学院 基幹理工学研究科 情報理工学専攻

‡ 早稲田大学 理工学術院

1 はじめに

命題論理式の充足可能性問題 (propositional SATisfiability problem: SAT) は NP 完全という特性から多項式時間で解くことは一般にできないと考えられる。しかしソフトウェア・ハードウェア検証, AI プランニングといった様々な実アプリケーションが多項式時間で SAT に還元可能であり, SAT を高速に解く意義は大きい。

本研究では, 優秀な逐次 SAT ソルバである MiniSat[3] を MPI を用いて並列化し, クラスタ上で実行することにより高速化を実現した。並列化の基本方針として, 各 PE にランダムに適当な探索域を探索させ, その際に学習した, 特定の探索域には解がないことを示す lemma を各 PE で送受信することにより, 探索域の削減を共有するよう実装した。また, lemma の渡し方や各パラメータをチューニングすることにより, 大幅な高速化を実現している。さらに実行時間に対して大きな割合を占めるブール制約伝播 (Boolean Constraint Propagation: BCP) の高速化を目指した。

2 SAT

SAT とはある命題論理式が充足可能 (SAT) になるような変数の割り当てを求めるか, またはどのような割り当てをしても充足不可能 (UNSAT) であることを示す問題であり, CNF で表すことができる。

CNF の例

$$\text{式} = (a + b')(b + c')(a + b + c)$$

式で, $a = F$ と割り当てた場合, 左の clause を真にするためには, $b = F$ と割り当てなければいけない。このような clause を unit clause と呼び, 連鎖的な割り当て作業を BCP と呼ぶ。

次に中央の clause を真にするためには $c = F$ という割り当てをしなければならない。一方, 右の clause を真にするためには $c = T$ と割り当てなければならない。このように真偽どちらに割り当てても式が偽となってしまうことを conflict (衝突) という。conflict を起こした際に, 原因の解析をし lemma と呼ばれる clause とし

て学習することによって同じ conflict を二度と起こさないようにできる。

3 並列 MiniSat の設計と実装

本研究では, 逐次 SAT ソルバ MiniSat をベースとし, MPI を用いて並列化し, 探索域の分割と lemma 共有を実装した。また BCP の並列化を検討し, Pthread を用いて実装した。

3.1 探索域の分割と lemma 共有

SAT ソルバを並列化する際に問題となるのが, 探索域の分割である。SAT の真理値の割り当ては BCP により行われることが多いため, 探索域の大きさが予測できず, 探索域を均等に分割することは難しい。

本実装では, 各 PE が他のプロセスの探索状況に関係なく探索できるようランダムに探索域を探索させている。さらに各 PE が学習した lemma を送受信することにより, 探索域の削減の情報を共有し, 探索域が重複しないようにしている。

探索域の重複を最小限に留めるためには, 各 PE で頻繁に lemma を送受信しなければならないが, 通信コストが膨大になってしまうため, 通信のタイミングと送受信する lemma の量は制限する必要がある。本実装では, 親マスター-複数の子マスター-複数のワーカと階層構造にし, 通信を分散することにより通信コストの削減を図った。また lemma の clause 長を制限することにより共有する lemma 量を制限している。

3.2 BCP 高速化

MiniSat では BCP の実装に clause の二つのリテラルを監視する 2-literal watching を採用している。BCP では割り当て済み変数キューからリテラルを取り出し, 取り出したリテラルを有する clause のリスト (監視リスト) に, unit clause があるか一つずつ調査していく。

並列化の方法として, 各スレッドがキューからリテラルを取りだし, それぞれが監視リストを調査する方法が考えられる。BCP による変数割り当ては多く, キューには常にリテラルが在り, タスク待ちはさほど起きないと考えられる。しかし, 複数の監視リストを並列に調査することにより, 変数割り当てと衝突のタイミングが逐次と大きく異なってしまうので, 衝突が起きた際の学習方法など修正すべき点が多い。

Lemma Sharing and BCP Speedup in the Parallel SAT Solver

†Kei OHMURA ‡Kazunori UEDA

†Department of Computer Science and Engineering, Graduate School of Fundamental Science and Engineering, Waseda University

‡Faculty of Science and Engineering, Waseda University

表 1: 各 PEs における性能向上比 (ratio)

PEs	7	13	19	25	31	37	43	49	55	61	67	73	79
SAT	7.01	10.1	14.4	15.6	20.7	19.8	24.4	22.7	25.5	29.5	27.7	26.4	28.2
UNSAT	2.65	4.98	9.91	10.5	14.6	19.5	21.6	22.2	22.6	25.5	22.6	27.1	24.9

別の方法として、あるリテラルの監視リストにある clause を各スレッドで分担する方法が考えられる。この方法だと一つの監視リストに対して、調査する clause の順番が変わるだけなので、修正点は少なく比較の実装が容易だが、キューからリテラルを取り出すごとに同期を取る必要があるため、大きな性能向上は見込みにくい。

今回は、後者の実装を行い、実験的にベンチマークを取ることにした。

4 評価と考察

評価に用いた並列クラスタは、GigE で接続された Intel Core2Duo 2.33GHz を搭載したマシンで構成され、主記憶容量は 1 台あたり 4GB である。

ソルバに解かせる問題として、SAT-Race 2006, SAT-Competition 2007 で使用された問題のうち、逐次版で解くのに 100 秒以上かかる問題を扱い、5 回実行した結果の平均を測定値とした。予備実験より子マスタ 1 台につきワーカー 5 台で実行した際の性能が最も高かったため、以下でも親マスタ 1 台-子マスタ n 台-ワーカー 5n 台で実行をしている。

表 1 に逐次版で 7200 秒以内に解くことができた SAT を 28 題、UNSAT を 41 題、計 69 題を並列実行した際の性能向上比の平均値を示す。表より SAT は PE 数を増やしても性能が低下することがあることがわかる。SAT の問題は探索域のどこに解があるかわからないため、並列実行した際に lemma を共有するタイミングが少し変わるだけで、大幅に実行時間がかかわることがあり、性能にばらつきがでやすいためである。一方、UNSAT の問題は性質上、全ての探索域に解がないことを示さなければならず、PE 数に応じて実行性能も増加している。しかし、SAT、UNSAT 共に 67PEs を超えた辺りから性能があまり向上しなくなっている。現在の実装では、これ以上 PE 数を増やして共有する lemma 量が増えてもさほど効果がないためだと考えられる。

図 1 に 61PEs で実行した際の、問題ごとの性能向上比を示す。SAT、UNSAT 共に元々速く解けていた問題はそれほど性能向上をしていない。しかし、逐次版での実行時間が大きくなるにつれ、台数以上の性能向上を示す問題が見られ、それぞれ最高で、113 倍、272 倍の性能向上を得ることができた。

また BCP を 3.2 節で述べた方法により Pthread によ

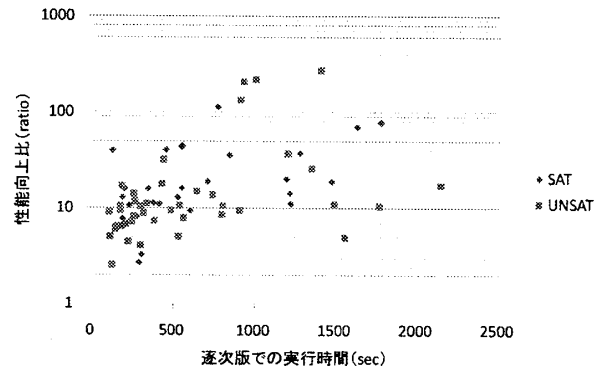


図 1: 61PEs 実行時の性能向上比

り並列化しベンチマークをとったところ、同期によるロック待ちが原因で、大幅に性能が低下した。

5 まとめと今後の課題

SAT ソルバ MiniSat を MPI を用いて並列化し、lemma の送受信をし探索域の削減を共有することにより性能向上を実現した。特に、いくつかの問題においては、SAT、UNSAT 共に PE 数以上の大きな性能向上を得ることができた。しかし、探索域を分割していない今回の方式では、ある程度 PE 数が増え、lemma の効果がそれほど発揮されず、性能向上に限界があることがわかった。今後、さらに大規模な環境で性能向上を得ることができるようになる必要がある。

今回は BCP の並列化を実験的に試した。BCP は逐次実行時に探索時間の 7 割以上を占めており、BCP の高速化は重要な課題であると考えられる。

参考文献

- [1] 大村圭, 渋谷健介, 稲垣良一, 上田和紀: "SAT ソルバ MiniSat の並列化とそのチューニング手法", 並列/分散/協調処理に関する『旭川』サマー・ワークショップ, 情報研報 Vol. 2007, No. 80, pp.31-36, Aug 2007.
- [2] Lintao Zhang, Sharad Malik: "The Quest for Efficient Boolean Satisfiability Solvers", in Proc. CAV'02, LNCS 2404, Springer, pp.17-36, 2002.
- [3] Niklas Eén, Niklas Sörensson: "An Extensible SAT-solver", in SAT 2003, LNCS 2919, Springer, pp.502-518, 2004.