3A-04

POWER8プロセッサ上でのTop-downアプローチによる幅優先探索の階層的並列処理

Hierarchical Parallelization of Breadth First Search Using Top-down Approach on POWER8 Processor

慶野 一朗† 吉田 明正†
Ichiro Keino Akimasa Yoshida

1 はじめに

Graph500ベンチマーク[1]のTop-downアプローチによる幅優先探索において，複数レベルの並列性を利用する階層的並列処理手法を提案する．Graph500の幅優先探索に関する研究では，従来よりTop-downとBottom-upを併用したHybrid並列性を利用する方法[2][3][4][5]が提案されているが，Top-downアプローチにおいて複数レベルの並列性は利用されていなかった．そこで，本稿ではTop-downアプローチにおいて，従来の探索最前線Frontierレベルの並列性に加えて，隣接頂点Neighborレベルの並列性を利用する階層的並列探索手法を提案する．提案手法はOpenMP実装されており，並列システムIBM POWER S812L（12コア，SMT=8，96スレッド）上で性能評価を行った結果，Graph500ベンチマークにおいて，従来手法より実行時間が26%短縮され，提案手法の有効性が確認された．

2 Top-downアプローチによる幅優先探索の階層的並列処理

本章では，Top-downアプローチによる幅優先探索において，複数レベルの並列性を利用する階層的並列処理手法について述べる．

2.1 Frontier間の並列処理

幅優先探索のアルゴリズムは図1の1〜8行目に示す通りであり，5行目から関数top-down-step()を呼び出す形となる．関数top-down-step()は9〜13行目に示す通りであり，従来のTop-downアプローチでは，9行目の探索最前線Frontier間で並列処理を行う．なお，10行目の隣接頂点Neighbor間の並列性は利用せず，逐次に実行される．

ここで，図2のようなFrontier=4，各FrontierのNeighbor=4のグラフを用いて，4スレッド（INSIDE=1）の割り当てを示す．Frontierのノード2〜5は，それぞれ4スレッドに割り当てられ，並列処理が行われる．なお，Frontier数がスレッド数より大きい場合には，複数のFrontierを各スレッドが実行する．

2.2 Frontier間・Neighbor間の階層的並列処理

本稿で提案する階層的並列処理では，図1のアルゴリズムにおいて，9行目の探索最前線Frontier間で複数スレッドグループを用いて並列処理を行い，10行目の隣接頂点Neighbor間でスレッドグループ内の複数スレッド（INSIDE数）を用いて，階層的な並列処理を実現する．

ここで，図2のようなFrontier=4，各Frontierの

```
     Function breadth-first-search(vertices, source)
01:  frontier ← {source}
02:  next ← {}
03:  parents ← {-1, -1, …, -1}
04:  while frontier ≠ {} do
05:  | top-down-step(vertices, frontier, next, parents)
06:  | frontier ← next
07:  | next ← {}
08:  return parents

     Function top-down-step(vertices, frontier, next, parents)
09:  for v ∈ frontier do        //Frontier間並列処理
10:  | for n ∈ neighbors[v] do //Neighbor間並列処理
11:  | | if parents[n]=-1 then
12:  | | | parents[n] ← v
13:  | | | next ← next ∪ {n}
```

図1 Top-downアプローチによる幅優先探索のアルゴリズム．

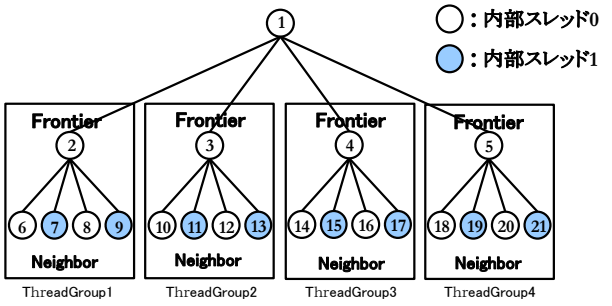

図2 幅優先探索の階層的並列処理におけるスレッド割り当て．

Neighbor=4のグラフを用いて，8スレッド（INSIDE=2）の割り当てを示す．Frontierのノード2〜5は，それぞれ4つのスレッドグループ（INSIDE=2）に割り当てられ，並列処理が行われる．次に，Frontierの隣接頂点Neighborは，スレッドグループ内の2スレッド（INSIDE=2の場合）にサイクリックに割り当てられて並列処理される．

なお，Frontier数がスレッドグループ数より大きい場合には，複数のFrontierを各スレッドグループが実行し，Neighbor数がINSIDE数より大きい場合には，複数のNeighborをスレッドグループ内の各スレッドが実行する．

3 POWER8プロセッサ上での性能評価

本章では，POWER8システム上でGraph500ベンチマーク[1]を用いて，幅優先探索の階層的並列処理の性能評価を行う．

3.1 性能評価の環境

性能評価に用いるPOWER8システムのアーキテクチャを表1に示す．POWER8プロセッサは12コア構成と

†明治大学総合数理学部ネットワークデザイン学科
Department of Network Design, School of Interdisciplinary Mathematical Sciences, Meiji University

表 1　POWER8 システムのアーキテクチャ.

マシン	IBM POWER S812L
CPU	POWER8 3.026GHz
コア数	12
SMT 数	8
メモリ	128GB
メモリバンド幅	192GB/S
OS(LINUX)	RHEL7 FOR POWER
C 言語処理系	gcc ver.4.8.5

なっているが，SMT(Simultaneous Multithreading)=8 とすることにより，96 スレッドの実行が可能となる．メモリバンド幅は 192GB/S であり比較的大きい．OS は RHEL7 FOR POWER である．

本性能評価では Graph500 の幅優先探索コード [1] をベースに，提案する階層的並列処理を OpenMP により実装した．OpenMP 実装において，`#pragma omp parallel` を用いており，各スレッドは `omp_get_thread_num()` によりスレッド番号を取得し，探索領域に対応したコードを実行する．具体的には，Frontier ノードをスレッドグループ（INSIDE 数のスレッドをもつグループ）にサイクリック割り当てし，その Frontier 内の Neighbor ノードはスレッドグループ内のスレッドに対してサイクリック割り当てを行う．

3.2　POWER8 上での Graph500 の SCALE 毎の性能評価

本節では，POWER8 上での Graph500 の SCALE 毎の性能評価を行う．図 3 の実行結果では，Graph500 の SCALE=20 から 24 まで変化させており，それぞれの SCALE において，従来の Frontier 並列性（INSIDE=1）のみを利用した並列実行，提案する Frontier 並列性・Neighbor 並列性（INSIDE=2）を利用した並列実行を 96 スレッドで行い，逐次実行比を表している．このグラフから SCALE=22 のときに，提案手法（INSIDE=2）は従来手法（INSIDE=1）に比べて 25.9%の速度向上が得られており，最も性能がよいことがわかる．

3.3　POWER8 上での Graph500 のスレッド毎の性能評価

本節では，POWER8 上での Graph500 のスレッド毎の性能評価を行う．図 4 の実行結果では，Graph500 の SCALE=22 において，コア数=12 とし，スレッド数を 12，24，48，96 まで変化させて，逐次実行比を測定した．それぞれのスレッド数において，従来手法の INSIDE=1，提案手法の INSIDE=2，4 を測定した．これらの結果から，いずれのスレッド数においても，INISDE=2 が最も速度向上率が高く，Frontier 並列性と Neighbor 並列性の両方が引き出されていることがわかる．Neighbor 並列は，Frontier 並列に比べて，並列処理の単位となるタスク粒度（granularity）が小さいため，INSIDE=2 の場合が INSIDE=4 の場合より優れた結果となっている．

なお，スレッド数=96 の場合は，従来手法（INSIDE=1）では逐次実行比 10.6 倍の速度向上であるが，提案手法（INSIDE=2）では逐次実行比 14.2 倍の速度向上が得られている．この場合，Frontier 並列に 48 スレッドグループが使用され，Neighbor 並列にスレッドグループ内の 2 スレッドが使用されたことになる．

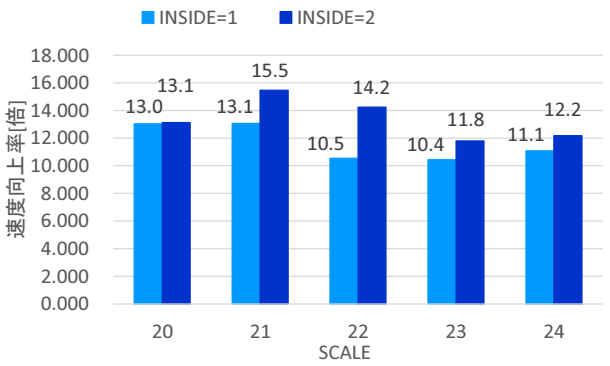

図 3　96 スレッド（INSIDE=1,2）における SCALE 毎の速度向上率.

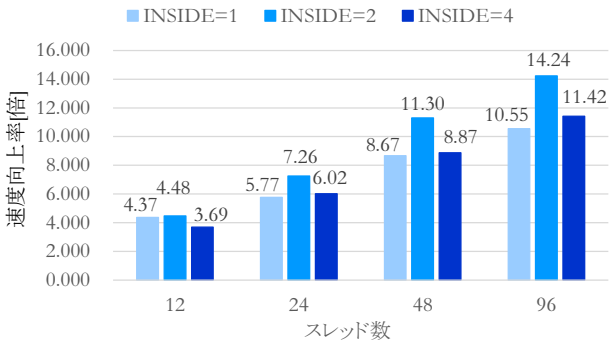

図 4　SCALE=22 におけるスレッド毎の速度向上率.

4　おわりに

本稿では，Graph500 ベンチマークの Top-down アプローチによる幅優先探索において，複数レベルの並列性を利用する階層的並列処理手法を提案した．本手法では，従来の探索最前線 Frontier レベルの並列性に加えて，隣接頂点 Neighbors レベルの並列性を利用しており，Frontier レベルの並列性が十分でない場合にも，隣接頂点 Neighbors レベルの並列性を効果的に利用することが可能となる．提案手法は並列システム IBM POWER S812L 上で OpenMP 実装されており，12 コア（SMT=8，96 スレッド）上で実行した結果，SCALE=22 の場合に逐次実行比で 14.2 倍の速度向上が得られ，提案手法の有効性が確認された．

参考文献

[1] Graph500. Graph500 Benchmark Code for Energy Measurements, http://green.Graph500.org.

[2] S.Beamer, A.Buluc, K.Asanovic, D.A.Patterson. Distributed Memory Breadth-First Search Revisited: Enabling Bottom-Up Search, Technical Report No.UCB/EECS-2013-2, University of California at Berkeley, 2013.

[3] 上野，鈴村，丸山，松岡．大規模分散メモリ環境におけるハイブリッド BFS の最適化，情報処理学会研究報告，Vol.2014-HPC-146，No.21，2014.

[4] 田邊，冨森，高田，城．Graph500 の Hybrid 解法に内在する局所性，情報処理学会研究報告，Vol.2013-HPC-140，No.27，2013.

[5] 安井，藤澤．NUMA を考慮した幅優先探索，情報処理学会研究報告，Vol.2014-AL-147，No.8，2014.