Vol.2017-HPC-158 No.1
2017/3/8

ジョブ実行中の計算ノードにおける DIMM 待機電力削減手法の実装と評価

石原 雅也[1,†1,a)]　三輪 忍[1,†1]　八巻 隼人[1,†1]　本多 弘樹[1,†1]

概要：近年のスーパーコンピュータではハードウェアの待機電力を削減することが求められており，我々は DIMM の待機電力を削減する方法として，メモリホットプラグを利用した省電力化手法を提案してきた．この手法では，ジョブの実行中に使用されていない DIMM の電源を OFF にすることで待機電力を削減する．DIMM の電源を OFF にした場合には，メモリ容量が減少するだけでなくメモリチャネル数の減少によってメモリバンド幅が減少する可能性があるが，これまでの我々の研究ではこの問題を考慮してこなかった．この問題に対し，本論文では，メモリバンド幅を維持する DIMM の電源制御手法を提案する．この手法では，チャネルに複数枚の DIMM が存在し，かつ使用メモリ量が各チャネル 1 枚目の DIMM で賄える場合にのみ，2 枚目以降の DIMM の電源を OFF にする．この提案手法を，Linux Kernel の物理ページ管理システムにメモリホットプラグを組み込むことで，OS から DIMM の電源状態を変更させる実装を行った．非実装の Linux Kernel に対する提案手法を実装した Linux Kernel の性能低下率，並びに DIMM の待機電力削減率を評価した結果，DIMM を ON 状態へ変更する閾値が 4GB の場合に最も効果があり，最大でも約 4 %の性能低下率となり，HPCC ベンチマークプログラムの問題サイズが 30000 以下の場合，DIMM の待機電力の約 50 %を削減した．

1. はじめに

近年のスーパーコンピュータは，処理速度が飛躍的に向上する一方で，膨大な消費電力が問題となっている．そのため，最先端のスーパーコンピュータにとって，電力効率の向上は重要な課題の一つである．電力効率を高めることはスーパーコンピュータ全体のコスト低減に繋がることから，これまで多くの研究者がスーパーコンピュータにおける様々な電力管理技術を開発してきた．その結果，様々なハードウェアの電力効率を改善することに成功してきたが [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]，スーパーコンピュータの運用コストの更なる削減が求められており，そのための電源制御技術が必要とされている．

スーパーコンピュータには長時間使用されないハードウェアデバイスが多く存在し，それらの待機電力を削減することができれば大きな省電力化につながると期待されている [3]．CPU では DVFS やパワーゲーティングが既に実用化されており [11]，これらの技術が CPU の待機電力の削減に大きく貢献している．一方，メモリシステムを構成する DIMM の待機電力削減技術としては，低電力モードなどが実用化されているが，未だ DIMM の消費電力は高く，これらの技術により十分な省電力効果は得られていない．

実際に 8GB の DIMM（DDR3-1600 SDRAM）が待機時に消費する電力を測定したところ，約 0.9W であった．これは，DIMM の低電力モードは，DIMM の DRAM チップの待機電力を削減する一方で，PLL 等の回路の待機電力を削減できないためだと考えられる．したがって，スーパーコンピュータのような膨大な数のノードで構成されるシステムにおいては，膨大な数の DIMM が大量の待機電力を消費する．

そこで本論文では，スーパーコンピュータにおけるジョブ実行時の DIMM の待機電力に着目した．スーパーコンピュータに搭載されている DIMM は，第 3 章で詳述する通り，多くのジョブに対して過剰供給となっている．

DIMM の待機電力の削減に対して有効な方法として，メモリホットプラグを使用した省電力法が挙げられる [12], [13]．メモリホットプラグは Linux カーネル 3.9 以降からサポートされている技術であり，OS が稼働中のコンピュータシステムに接続されている DIMM の交換を可能にする技術である．メモリホットプラグがサポートされたハードウェア，およびファームウェアを用いて，DIMM への電力供給を止めることで，待機電力を削減することが出来る．

[1] 情報処理学会
IPSJ, Chiyoda, Tokyo 101–0062, Japan
[†1] 現在，電気通信大学
Presently with The University of Electro-Communications
[a)] m-ishihara@hpc.is.uec.ac.jp

メモリホットプラグを用いた省電力法の研究はこれまでも行われており，代表的なものとしては ACPI Based Memory Hot-Plug[12] や，On-Demand Memory Hot-Add[13] が挙げられる（2.2.3 で詳述）．しかし，前者はジョブの実行時には DIMM の待機電力を削減できない点，後者は制御用プログラムが動作することで時間的オーバヘッドが増大する点，また DIMM を OFF 状態にすることでメモリバンド幅が減少し性能低下する可能性がある点が課題として挙げられている．

上記の背景を踏まえ，本研究では，ジョブ実行時におけるスーパーコンピュータの DIMM の待機電力の削減，DIMM 単位の電源の ON/OFF の制御，メモリバンド幅を減少させない制御の実現を目的とし，メモリバンド幅を維持する DIMM の電源制御手法を提案する．本手法では，DIMM の電源を OFF にする際，常に各チャネル 1 枚以上の DIMM が接続されている状態とすることで，メモリバンド幅を減少させない．また，Linux Kernel の物理ページ管理システムにメモリホットプラグを組み込み，OS から DIMM の電源状態を変更することで，ジョブ実行時にも DIMM の電力を OFF にでき，かつ，制御用プログラムを必要としない実装を目指した．本提案手法は，Linux Kernel 4.1.34 に実装した．

以下に本論文の構成を述べる．まず，第 2 章では，スーパーコンピュータで一般的に用いられているメモリシステムの構成，本論文の提案手法で用いた技術であるメモリホットプラグについて述べる．次に第 3 章では，SWoPP2016 で我々が発表した論文 [1] から，スーパーコンピュータ CX400 のジョブログの解析とその結果について述べる．そして第 4 章で，本論文で提案する DIMM の電源制御手法についての詳細，および提案手法の実装方法について述べる．続く第 5 章では，提案手法の性能を評価する．最後に，第 6 章でまとめと考察および今後の課題について述べる．

2. 研究背景

2.1 メモリシステムの構成

スーパーコンピュータのノード内のメモリシステムは，DIMM，チャネル，スロットからなる．DIMM はメモリチップを基盤にまとめたモジュールである．1 つの CPU ソケットを有するノードのメモリシステムの簡単な構成図を図 1 に示した [14], [15], [16]．CPU にはバス，チャネルを介して，DIMM スロットが接続されている．

チャネルは DIMM からメモリコントローラ（MC）へデータを転送する際に用いられる転送経路であり，DIMM の各スロットに繋がっている（最大 3 スロット）．使用可能な DIMM が接続されていない場合，そのチャネルは使用されず，メモリバンド幅が減少する．

チャネルを通して転送されたデータは，バスを通り CPU のメモリコントローラに転送される．メモリを正常に機能させるため，コンピュータには DIMM を接続するスロットの順番や配置が接続する DIMM の枚数によって定められている．一般的に DIMM は，一番目のスロットから順番に各チャネル 1 枚ずつ接続される．

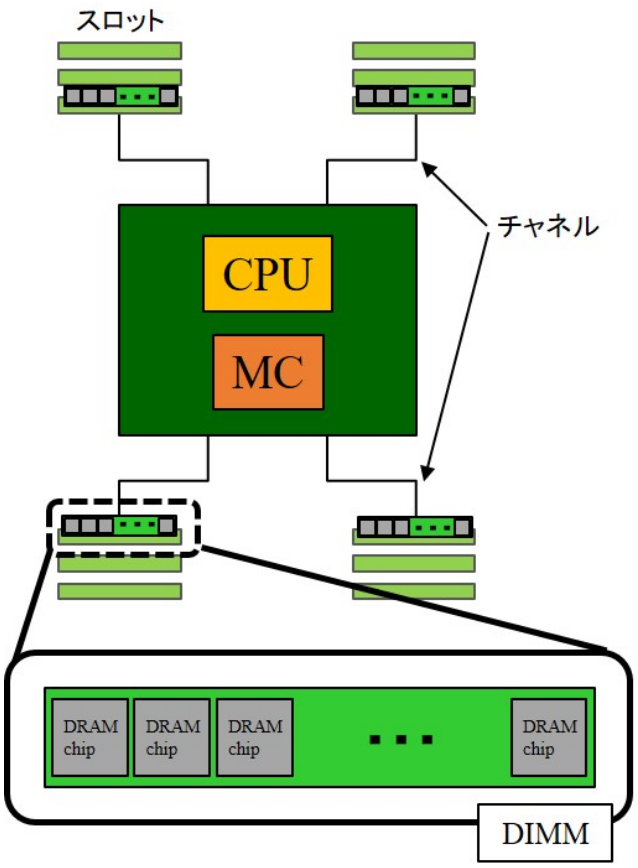

図 1　メモリシステムの構成図

メモリバンド幅は DIMM が接続されているチャネル数に依存する．そのため，各チャネルに出来るだけ均等に DIMM を接続したと仮定すると，DIMM の総数がチャネル数を下回っている場合，DIMM を接続した分だけメモリバンド幅は増加する．一方で，各チャネルに出来るだけ均等に DIMM を接続したと仮定し，一つのチャネルに 1 枚以上の DIMM が接続され，DIMM の総数がチャネル数を上回った場合，複数の DIMM が一つのチャネルを共有することになる．つまり，同一チャネルの 2 スロット目以降に DIMM を追加した場合は，メモリ容量は増加するが，メモリバンド幅はほとんど増加しない．

2.2 メモリホットプラグ

2.2.1 メモリホットプラグの概要

Linux Kernel 3.9 以降からサポートされているメモリホットプラグは，システムの運用中に DIMM を抜き差しすることを可能とする技術であり，一般的には故障した DIMM の交換などシステムの可用性の向上のために用いられる．

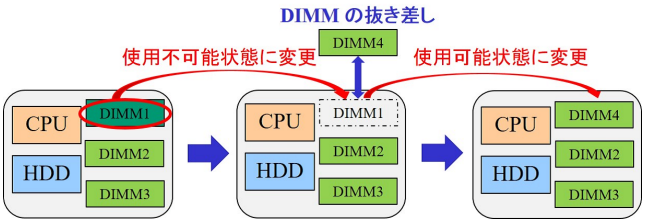

図 2　メモリホットプラグによる DIMM の状態変更の遷移

メモリホットプラグによる DIMM の着脱の過程は，論理フェーズと物理フェーズからなる（図 2）．論理フェーズでは，DIMM の抜き差しを行う前に OS が DIMM の状態（使用可能（以降，ON とする）な状態と使用不可能（以降，OFF とする）な状態がある）を変更し，物理フェーズでは，実際に使用不可能状態とした DIMM の抜き差しを行う．DIMM の状態変更と，変更する DIMM の指定は root 権限を持つユーザーのみが行うことが出来る．OFF を指定した DIMM に既にデータがある場合は，Kernel が他の ON になっている DIMM にデータを移動する [20]．

2.2.2　メモリホットプラグを用いて DIMM の電源を OFF にした場合のデメリット

本研究では，メモリホットプラグによって使用不可能状態とした DIMM の電源を OFF にすることで DIMM の待機電力を削減する．しかし，メモリホットプラグを用いて DIMM の電源を OFF にした場合，次のデメリットが考えられる．

ひとつは，メモリ容量の減少によるスラッシングの発生である．スラッシングとは，メモリの容量が足りなくなったときにハードディスク上にデータを移動させることを指す．スラッシングが生じると，データを参照する際にハードディスクとメインメモリの間でデータの入れ替えを行うため，大量のディスクアクセスが発生し，アプリケーションの処理速度が低下する．そのため，メモリ量を減らしすぎてしまうと，実行中のアプリケーションが使用できるメモリ量以上を要求した場合，減らす前のメモリ量で実行したときには起きないはずのスラッシングが生じてしまうことになり，アプリケーションの処理性能の低下を引き起こす．

もうひとつは，メモリバンド幅の減少によるデータの転送バンド幅の減少である．同一チャネルに接続された全ての DIMM の電源を OFF にした場合，使用されるチャネルの数が減少し，メモリバンド幅が減少する．ジョブが必要とするメモリバンド幅より少なくなった場合，メモリ容量は十分だが，データの転送速度が遅くなってしまい，アプリケーションの処理速度が低下する可能性が考えられる．

2.2.3　先行研究

DIMM の待機電力削減のためにメモリホットプラグを用いた既存の研究として，ACPI Based Memory Hot-Plug[12] と，On-Demand Memory Hot-Add[13] を紹介する．

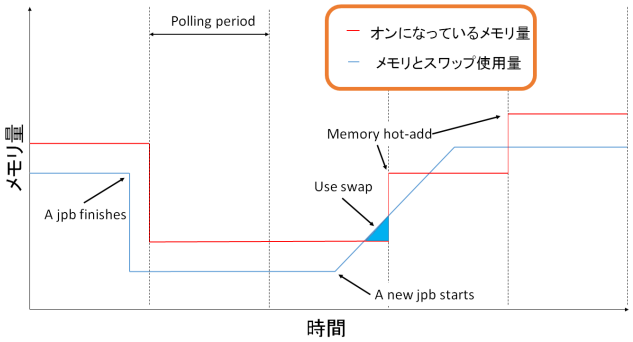

図 3　オンデマンド制御によるメモリホットアド

Chen らが提案している ACPI Memory Hot-Plug[12] では，ノード全体が待機状態の場合，各 CPU とそれに対応した DIMM を自動的に一緒に OFF にすることで，DIMM の待機電力の削減を行っている．そして，ACPI Memory Hot-Plug はファームウェアと OS 間の通信方法を提供することでメモリのホットアドおよびホットリムーブを行っている．また，ACPI Memory Hot-Plug は Linux Kernel 3.11 以降を対象としている．

また，従来手法の On-Demand Memory Hot-Add[13] では，メモリ量を減らしすぎた場合にスラッシングが増加することを考慮して，メモリの使用量に応じて ON 状態のメモリ量を制御している．この方法では，定期的に計算ノードのメモリとスワップの使用状況を監視し，デーモンプログラムによってメモリを ON/OFF 状態に切り替える（図 3）．計算ノードが待機状態になると，ノード上のメモリ使用量が減少する．そこで，デーモンプログラムは監視しているメモリの使用量に応じて ON 状態のメモリ量を減少させる．多くのジョブがノード上で実行を開始され，使用可能なメモリサイズを超えた場合は，スワップ領域を使用する．この状況をデーモンプログラムが検知した場合，スワッピングを回避するために，デーモンプログラムは ON 状態のメモリ量を増加させる．

On-Demand Memory Hot-Add[13] では，上記の手法によってアプリケーション性能がどの程度低下するのかを評価している．計算ノードとスワップの使用状況を監視する期間（ポーリング期間）を 3 パターン（0.1s, 0.01s, 0.001s）用意し，HPL，starDGEMM，PTRANS，RA（MPI RandomAccess），FFT（MPI FFT）の 5 つのベンチマークについて評価を行っている．この実験では測定に HPCC ベンチマークのプログラムを用いている．この実験では，各アプリケーションが実行に使用するメモリ量を全体の約 20 %程度にまで削減でき，これによる実行速度の低下は最大で約 3 %に抑制できた，という結果を示している．このことから，メモリホットプラグは省電力化に十分期待できると述べられている．一方でこの実験は，メモリ量を減らし

表 1　CX400 の性能

Name	Remarks
演算ノード	理論演算性能 345.6GFLOPS 主記憶容量 128GB チャネル数　8 本 DIMM 容量　8GB/枚 メモリバンド幅 102.4GB/s
総ノード数	1,476 ノード
総プロセッサ (コア) 数	2,952 プロセッサ (23,616 コア)
主記憶容量の総和	約 184.5TiB

てもチャネル数が減らない理想的な環境でしか行っておらず，DIMM 単位で ON/OFF をした場合に生じる可能性がある，メモリバンド幅の減少を考慮する必要がある．

3. スーパーコンピュータにおける電力削減効果の見積もり

SWoPP2016 で我々が発表した論文 [1] では，チャネル 2 枚目以降の DIMM を OFF にする機会がどの程度存在するのかを，九州大学情報基盤センターで運用されているスーパーコンピュータ CX400 のジョブログを解析することで明らかにしている．以下，その結果をまとめる．

3.1 CX400 システムの概要

九州大学の高性能演算サーバシステム，スーパーコンピュータ CX400 のシステム構成は表 1 の通りである．なお，今回の解析にあたり，搭載されている DIMM の容量を，8GB（DIMM を各チャネル 2 枚用いている）と仮定した．

今回使用したログは，システム稼働開始（2012 年 10 月）直後の比較的空いている時期（2013 年 1 月〜6 月）のものであり，現在のシステム全体の CPU 稼働率は CX400 で 55〜85 %程度である．また，九州大学では，一般的な共有利用ノード群の他，特定の研究グループにノードを割り当てる占有利用ノード群を設定しており，それらのノードについては，割り当てた研究グループの活動状況によってはアイドル時間が極端に長くなる場合がある．そのため，以上の点に考慮して解析を行った．

3.2 全ノードに対するジョブのメモリ使用量の解析

DIMM の待機電力を削減するためには，ジョブの全実行時間に対して，ジョブによって使用されるメモリ容量がどの程度なのかを知る必要がある．そのためこの解析では，ジョブによって使用されたメモリ容量をチャネル 1 枚目分の容量，チャネル 2 枚目分の容量と大きく二つに分け，各チャネルの 2 枚目以降の DIMM を OFF にする機会がどれだけあるかを検証している．ジョブに割り当てられたメモリ容量と時間の関係を表したグラフが図 4 になる．なお，ジョブが実行されていない間の使用メモリ量を 2GB と仮定した．

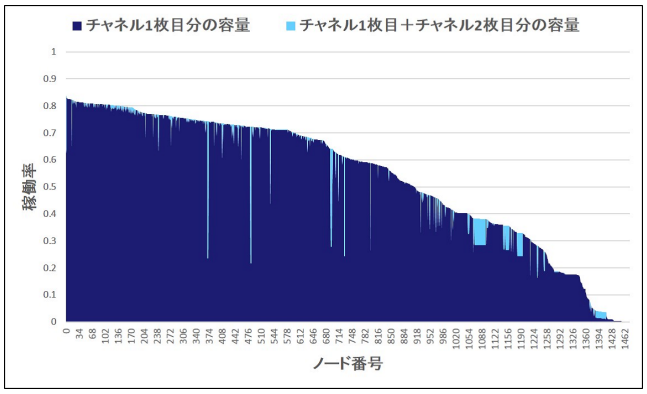

図 4　全ノードにおけるジョブに割り当てられたメモリ容量と時間の関係

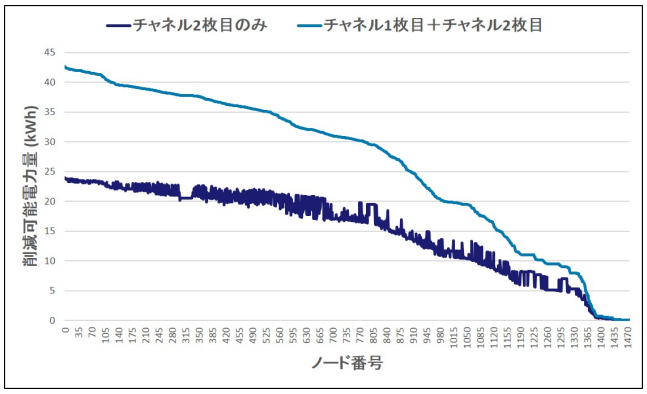

図 5　ノード別の削減可能電力量

図 4 のグラフでは，ジョブが実行されていた時間を稼働率として表記しており，チャネル 1 枚目分の容量で稼働していた割合と，チャネル 1 枚目と 2 枚目分の容量で稼働していた割合を示している．ほとんどの場合，チャネル 1 枚目分の DIMM の容量で稼働していることがわかり，全ノードを平均すると，ジョブの実行時間の約 97 %でチャネル 2 枚目の DIMM を OFF にできることが確認できた．

3.3 省電力効果の試算結果

また，前述の解析結果をもとに，メモリバンド幅を維持して各チャネル 1 枚目分の DIMM は OFF にせず 2 枚目分のみを OFF にした場合と，メモリバンド幅の減少を無視し理想的に各チャネル 1 枚目分と 2 枚目分の DIMM を OFF にした場合について，削減可能な電力量の試算を行った．ここで，DIMM1 枚の待機電力を約 0.9W とし，DIMM を OFF にすることで生じる性能低下分を無視した．

チャネル 2 枚目のみを OFF にした場合でも，性能低下を無視して削減した場合の約 60 %の電力削減が見込めることがわかり，十分に電力の削減が見込めることが確認できた．

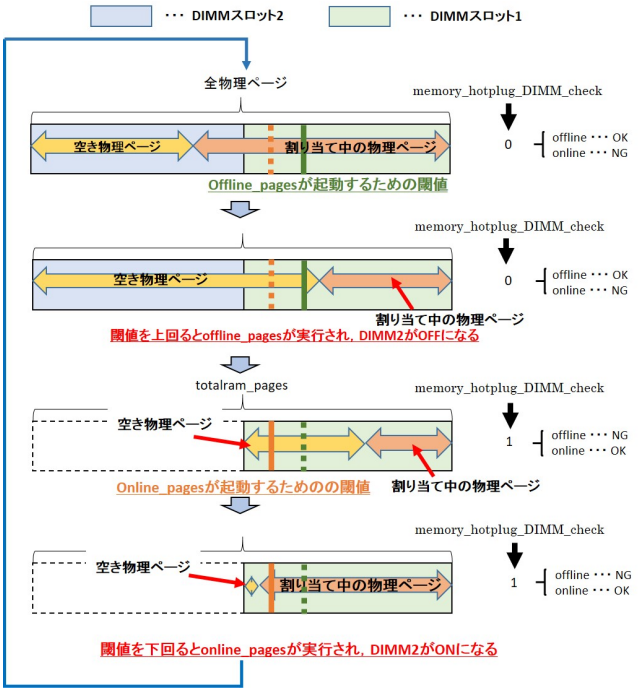

図 6　DIMM の電源制御手法の動作

4. 提案手法

4.1 メモリバンド幅を維持する DIMM の電源制御手法

本論文では，各チャネルに複数枚の DIMM が存在し，かつ使用メモリ量が各チャネル 1 枚目分の容量を下回る場合にのみ，各チャネル 2 枚目以降の DIMM の電源を一斉に OFF にすることによって，メモリバンド幅を維持する DIMM の電源制御手法を提案する．

この提案手法では，各チャネル常に 1 枚以上の DIMM が ON となり，2 枚目以降の DIMM の電源を OFF に変更してもメモリバンド幅はほぼ一定となり，チャネル数（メモリバンド幅）の減少によってアプリケーション性能が低下することを防ぐことが出来る．

また，メモリバンド幅を維持する，DIMM の状態変更回数をなるべく少なくする，メモリ量不足によるジョブの処理速度低下を回避するため，次のような方法をとった．プロセス開始時は各チャネル 1 枚目の DIMM を ON にした状態で動作させる．プロセス開始後にプロセスが必要とするメモリ容量が各チャネル 1 枚目分の DIMM の容量を超えそうになった場合に各チャネル 2 枚目以降の DIMM を順番に ON にする．一度各チャネル 2 枚目以降の DIMM を ON にしたらプロセス実行中は OFF にしない．そして次のプロセス開始時には，また各チャネル 1 枚目の DIMM を ON にした状態で，同様の動作させる．

4.2 Linux Kernel への実装

本論文では，ジョブ実行時に DIMM の電源状態を変更でき，かつ制御用プログラムによる時間的オーバーヘッドを削減するために，Linux Kernel の物理ページ管理システムにメモリホットプラグを組み込むことで，OS から DIMM の電源状態を変更させる実装を行った．

```
1  //グローバル変数　memory_hotplug_DIMM_check :
2  //状態変更可能なDIMMが存在するかを判断する．
3  //閾値　online_threshold :
4  //ユーザーが設定する．2枚目以降のDIMMをon状態に変更するかを判断する．
5  //関数　global_page_state :
6  //呼び出した時点における引数で指定された物理ページ量を返す．
7  //関数　online_pages :
8  //第一引数のページフレーム番号から,第二引数のページ量を
       online状態に変更する．
9  /*2枚目以降のDIMMがONにされていることを確認する*/
10 If{memory_hotplug_DIMM_check == 1){
11 /*空き物理ページが閾値を下回った場合，DIMMをON状態に変更する*/
12         if(global_page_state(NR_FREE_PAGES) <= online_threshold){
13                 online_pages(online_start_pfn, online_nr_pages,
                       MMOP_ONLINE_KEEP);
14                 memory_hotplug_DIMM_check == 2;
15         }
16 }
```

図 7　OFF 状態から ON 状態への変更を行う疑似コード

実装には Linux Kernel 4.1.34 を使用した．本提案手法の動作概略図を図 6 に示す．本論文では，搭載されている DIMM が各チャネル 2 枚の場合を想定した実装を行った．

搭載されている DIMM が各チャネル 2 枚ずつでかつ全て同容量の場合，空き物理メモリ量が総物理メモリ量の半分以上になれば，メモリホットプラグによってどちらかの DIMM にデータをまとめることで，各チャネル 2 枚目の DIMM の電源を OFF 状態に出来る．そのため，削減可能な DIMM が存在するか否かを Linux Kernel 上で判断するためには，空き物理ページの量を確認すればよい．そこで，Linux Kernel の動作の中で，予め空き物理ページが増加すると見込まれる操作の後に DIMM を OFF 状態に変更させ，逆に空き物理ページが減少すると思われる操作の前に DIMM を ON 状態に変更させるような実装を行った．また，メモリ量不足による性能低下を回避するため，本実装における DIMM の状態変更は，空き物理ページを監視し閾値との大小比較によって決定している．DIMM の電源を ON 状態，OFF 状態に変更する実装の詳細は次の通りである．

4.2.1 OFF 状態から ON 状態への変更

OFF 状態から ON 状態への変更処理では，プロセスのメモリ割り当て要求時に alloc __ pages() 関数が呼ばれることで， 空き物理ページ量が減少するため，alloc __ pages() 関数内で物理ページの割り当て作業を開始する直前（alloc __ pages __ current() 関数の実行直前）で，まず OFF 状態の DIMM がないかチェックを行い（図 7，10 行目），OFF 状態の DIMM があった場合は，空き物理ページ量を閾値と比較している（図 7，12 行目）．ここで，空き物理ページ量が閾値以下の場合は， online __ pages() 関数を呼び出

```
1  //グローバル変数　memory_hotplug_DIMM_check：
2  // 状態変更可能なDIMMが存在するかを判断する．
3  //閾値　offline_threshold：
4  // ユーザーが設定する．2枚目以降のDIMMをoff状態に変更するかを判断する．
5  //関数　global_page_state：
6  // 呼び出しだ時点における引数で指定された物理ページ量を返す．
7  //関数　offline_pages：
8  // 第一引数のページフレーム番号から,第二引数のページ量を
       offline状態に変更する．
9  //関数　mutex_lock, mutex_unlock：排他制御を行う．
10 /*排他制御を行う*/
11 mutex_lock(&memory_hotplug_lock);
12 /*2枚目以降のDIMMがONにされていることを確認する*/
13 If{memory_hotplug_DIMM_check == 2){
14 /*空き物理ページが閾値を下回った場合，DIMMをON状態に変更する*/
15         if(global_page_state(NR_FREE_PAGES) >= offline_threshold){
16                 offline_pages(offline_start_pfn, offline_nr_pages);
17                 memory_hotplug_DIMM_check == 1;
18         }
19 }
20 mutex_unlock(&memory_hotplug_lock);
```

図 8　ON 状態から OFF 状態への変更を行う疑似コード

し，各チャネル 2 枚目の DIMM を一斉に ON 状態に変更する（図 7，13 行目）．

4.2.2　ON 状態から OFF 状態への変更

ON 状態から OFF 状態への変更処理では，プロセス終了時に do __ exit() 関数が呼ばれることで，空き物理ページ量は増加するため，do __ exit() 関数内の物理ページの解放作業の完了後（exit __ mm() 関数の実行後）に，まず各チャネル 2 枚目の DIMM が ON 状態かチェックを行い（図 8，13 行目），各チャネル 2 枚目の DIMM に ON 状態のものがあった場合，空き物理ページ量を閾値と比較している（図 8，15 行目）．ここで，空き物理ページ量が閾値以上の場合は offline __ pages() 関数を呼び出し，2 枚目の DIMM を OFF 状態に変更する（図 8，16 行目）．

5.　評価実験

5.1　実験方法

本論文では，提案手法を実装した Linux Kernel と，非実装の Linux Kernel の性能比較，DIMM の平均待機電力の比較することで評価を行った．今回は，DIMM を ON 状態へ変更する閾値を 4GB,8GB とした場合と，後述するメモリホットプラグによる DIMM のモード遷移時の時間的オーバヘッドの値を 1000，10000 とした場合を測定している．

評価に用いたマシンのシステム構成は表 2 の通りで，チャネル数が 4 本，1 枚当たりの DIMM の容量が 4GB のものを使用した．また，ベンチマークプログラムには，HPCC ベンチマークプログラムの中から，メモリ使用量の多いプログラムである PTRANS, starSTREAM, starDGEMM, starFFT の 4 つを実験に使用した．

メモリホットプラグによる DIMM のモード遷移（OFF 状態，ON 状態への変更）には，ある程度の時間がかかることが予想される．しかし，現在の実験環境では，メモリホットプラグに対応したハードウェアが存在せず，実際にはメモリホットプラグを実行しても DIMM の電源を OFF にすることは出来ない．そのため，メモリホットプラグによる DIMM のモード遷移時に発生する時間的オーバヘッドを for 文の空ループによって再現し，実験を行った．

表 2　評価実験に用いたノード構成

Name	Remarks
CPU	Xeon E5-2630L 2.0GHz × 1 6 physical cores
Memory	4GB DDR3-1333 LV-RDIMM x8
OS	Scientific Linux with Linux kernel 4.1.34
チャネル数	4

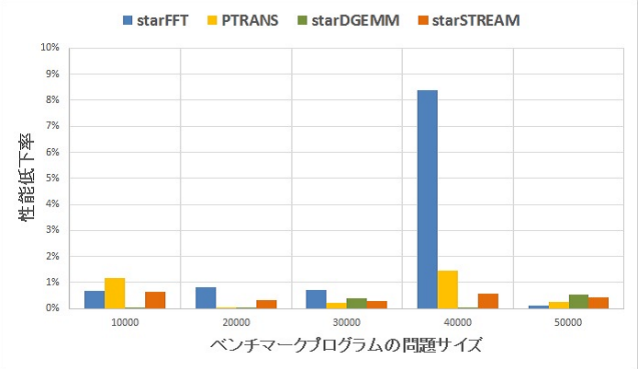

図 9　非実装 Linux Kernel に対する性能低下率（閾値 8GB）

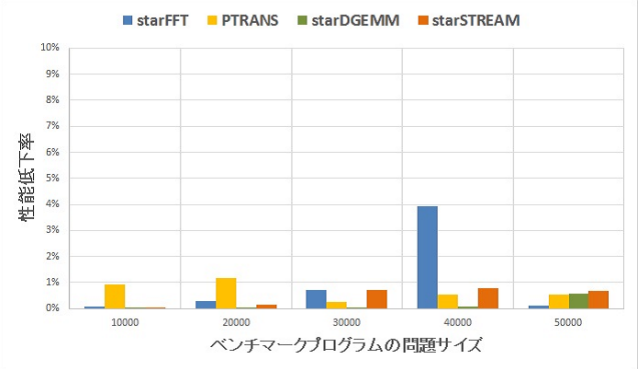

図 10　非実装 Linux Kernel に対する性能低下率（閾値 4GB）

5.2　異なる閾値に対する評価結果

DIMM を ON にする閾値を 4GB，8GB とした場合の，非実装 Linux Kernel との性能比較，DIMM の平均待機電力の比較を行った．ここで，DIMM を OFF にする閾値を 24GB，DIMM のモード遷移による時間的オーバヘッドの for 文を 1000 回としている．

5.2.1　非実装 Linux Kernel との性能比較

非実装の Linux Kernel に対する提案手法を実装した Linux Kernel の性能低下率を測定した結果を図 9，図 10 に

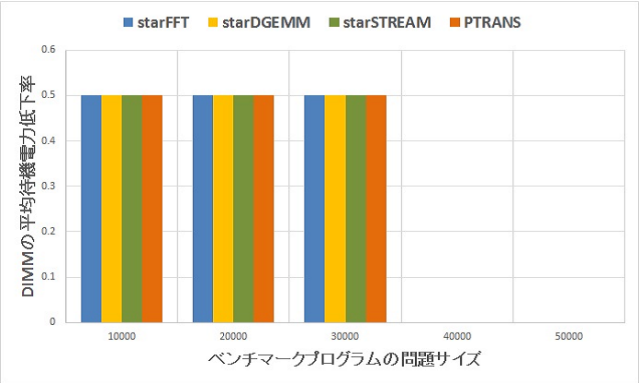

図 11 非実装 Linux Kernel に対する DIMM の平均待機電力削減率（閾値 8GB）

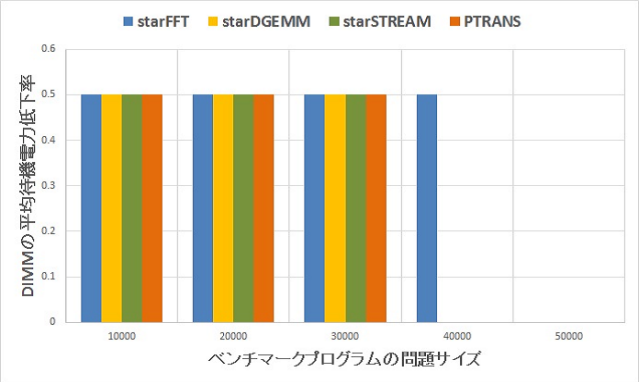

図 12 非実装 Linux Kernel に対する DIMM の平均待機電力削減率（閾値 4GB）

示す．

PTRANS, starSTREAM, starDGEMM のベンチマークプログラムについては性能低下率を約 1 %程度に抑え，特に大きい値を示した starFFT でも，閾値が 4GB の場合，最大でも約 4 %の性能低下率となった．

5.2.2 非実装 Linux Kernel との DIMM の平均待機電力の比較

非実装の Linux Kernel に対する提案手法を実装した Linux Kernel の DIMM の平均待機電力削減率を測定した結果を図 11，図 12 に示す．この評価にあたり，DIMM の待機電力を約 0.9W と仮定した．

HPCC ベンチマークプログラムの問題サイズが 30000 以下の場合，今回測定に使用した PTRANS, starSTREAM, starDGEMM, starFFT 全てのベンチマークプログラムで，約 50 %の DIMM の待機電力の削減率となった．

5.3 DIMM のモード遷移時の時間的オーバヘッドを増加させた場合の評価結果

DIMM のモード遷移時の時間的オーバヘッドを for 文 10000 回とした場合の，for 文 1000 回との性能比較を行った．DIMM のモード遷移時の時間的オーバヘッドを for 文 1000 回とした場合に対する，for 文 10000 回とした場合の性能低下率を測定した結果を図 13 に示す．ここで，DIMM を ON にする閾値を 8GB，OFF にする閾値を 24GB としている．

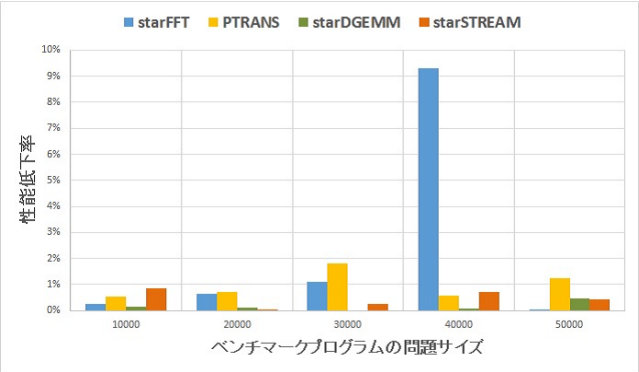

図 13 DIMM のモード遷移時の時間的オーバヘッドを増加させた際の性能低下率

時間的オーバヘッドを増加させた場合，PTRANS, starSTREAM, starDGEMM のベンチマークプログラムについてはほとんど違いは確認できなかった．一方で，starFFT では図 9 で見受けられた性能低下がここでも確認できた．そのため，starFFT のような通信性能測定に用いられるベンチマークプログラムを使用する場合，DIMM のモード遷移時に発生する時間的オーバヘッドが性能低下に影響を与える可能性が考えられる．

6. おわりに

本論文では，ジョブ実行時におけるスーパーコンピュータの DIMM の待機電力の削減を目的とし，メモリバンド幅を維持する DIMM の電源制御方法，Linux Kernel の物理ページ管理システムにメモリホットプラグを組み込み，OS から DIMM の電源状態を変更させる実装方法を提案した．

提案手法を実装した Linux Kernel と，非実装の Linux Kernel の性能比較，DIMM の平均待機電力の比較によって評価を行った．結果，DIMM を ON 状態へ変更する閾値が 4GB の場合に最も効果があり，最大でも約 4 %の性能低下率となること，そして HPCC ベンチマークプログラムの問題サイズが 30000 以下の場合，DIMM の待機電力の約 50 %を削減できることを示した．

今後の課題としては，スーパーコンピュータに対する実装を考えた場合，1 ノード実行時の評価だけでは不十分である．そのため，複数ノードで実行した場合に性能低下することなく稼働出来るか評価が必要である．

謝辞 本研究のために CX400 のジョブログを提供していただいた九州大学情報基盤センターに感謝致します．本研究の一部は JST CREST による．

参考文献

[1] 石原 雅也，三輪 忍，八巻 隼人，本多 弘樹，メモリホットプラグを用いたメインメモリの省電力化に関する初期検討，情報処理学会報告 2016-HPC-155，No.22，pp.1-7，2016.
[2] M. Okuda, “Searching out Trends in the Supercomputer Environment and Technology Development Following the “K computer”,” BioSupercomputing Newsletter Vol.7, 2012.
[3] S. Miwa and H. Nakamura, “Profile-based power shifting in interconnection networks with on/off links,” In Proceedings of the 2015 ACM/IEEE Conference on Supercomputing, 2015, pages .
[4] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, “Exploring hardware overprovisioning in power-constrained, high performance computing,” In Proceedings of the 27th International Conference on Supercomputing, 2013, pages 173182.
[5] Luiz Andr Barroso and Urs Hlzle, “The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines,” Synthesis Lectures on Computer Architecture, 2009.
[6] Luiz Andr Barroso and Urs Hlzle, “The Case for Energy-Proportional Computing,” IEEE Computer, 2007, December, pages 40(12):3337.
[7] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, “Disaggregated Memory for Expansion and Sharing in Blade Servers,” ISCA’09 : International Symposium on Computer Architecture, 2009, pages 267-278.
[8] Lluc Alvarez, Llus Vilanova, Miquel Moreto, Marc Casas, Marc Gonzlez, Xavier Martorell, Nacho Navarro, Eduard Ayguad, Mateo Valero, “Coherence Protocol for Transparent Management of Scratchpad Memories in Shared Memory Manycore Architectures” ISCA’15, 2015, pages 720-732.
[9] Qingyuan Deng, David Meisner, Luiz Ramos, Thomas F. Wenisch, Ricardo Bianchini, “MemScale: active low-power modes for main memory,” ASPLOS‘11: ACM SIGARCH Computer Architecture News, 2011, pages 225-238.
[10] Qingyuan Deng, Luiz Ramos, Ricardo Bianchini, David Meisner, Thomas F. Wenisch, “Active Low-Power Modes for Main Memory with MemScale,” IEEE, 2012, May, pages 60-69.
[11] Qingyuan Deng, David Meisner, Abhishek Bhattacharjee, Thomas F. Wenisch, Ricardo Bianchini, “MemScale: Memory System DVFS with Multiple Memory Controllers,” ISLPED‘12, 2012, July, pages 297-302.
[12] T. Chen, “Introduction to ACPI based memory hotplug,” LinuxCon Japan, 2013.
[13] S. Miwa, and H. Honda, “Memory Hotplugfor EnergySavings of HPC systems,” SC’15, poster, 2015.
[14] David A.Patterson, john L.Hennessy, 成田光彰 訳, ”コンピュータの構成と設計第 3 版 (上)(下),” 日経 BP 社, 2006.
[15] Bruce Jacob, ”The Memory System: You Can’t Avoid It, You Can’t Ignore It, You Can’t Fake It,” Synthesis Lectures on Computer Architecture, 2009, pages 77.
[16] Youngsok Kim, Jaewon Lee, Jae-Eon Jo, and Jangwoo Kim, ”GPUdmm: A high-performance and memory-oblivious GPU architecture using dynamic memory management,” High Performance Computer Architecture, IEEE 20th International Symposium on, 2014, pages 546-557.
[17] Prashant Nair, Chia-Chen Chou, Moinuddin Qureshi, ”A case for Refresh Pausing in DRAM memory systems,” High Performance Computer Architecture, IEEE 19th International Symposium on, 2013, pages 23-27.
[18] Junwhan Ahn, Sungjoo Yoo, Onur Mutluy, Kiyoung Choi, ”PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture,” ISCA‘15, 2015, pages 336-348.
[19] Subhasis Das, Tor M. Aamodt, William J. Dally, ”SLIP: ReducingWire Energy in the Memory Hierarchy,” ISCA‘15, 2015, pages 13-17.
[20] Y. Ishimatsu, ”Memory hotplug” LinuxCon Japan, 2013.