Vol.2011-HPC-130 No.33
2011/7/28

MPI-IO/Gfarm におけるデータ配置を考慮したプロセススケジューリングの検討

木 村 浩 希[†1,†2] 建 部 修 見[†1,†2]

本稿では，MPI-IO/Gfarm におけるデータ配置を考慮したプロセススケジューリングについて検討を行うため I/O が計算に与える影響について評価を行う．大規模計算環境では複数のジョブが並列に実行される．このような環境に Gfarm ファイルシステムのようなローカルストレージを束ねる設計のファイルシステムを用いた場合，他のジョブが実行されているノードに対してファイルアクセスを行う可能性がある．また近年メニーコアかが進んでおり I/O 処理等の為にコアを用意することが可能になっていると考えられる．MPI-IO/Gfarm におけるプロセススケジューリングを検討するにあたり，I/O 処理が計算に与える影響，余剰コアの影響についての評価が必要であると考えれ評価を行った．NPB に含まれる CG，FT，BT の 3 つのベンチマークを用いて I/O の与える影響について評価を行った結果，CG ベンチマークでは最大 30%，FT ベンチマークでは最大 20%，BT ベンチマークでは最大 114%性能が低下した．しかし I/O 用にコアを用意することで CG，FT については I/O による影響を最大 2%程度に抑えられた．BT はそれ自体が I/O 処理を行うため，I/O 部分の競合によって性能が大きく低下し，I/O 用のコアの影響も限定的であった．

1. はじめに

近年，科学技術計算分野における解析データの大規模化が進んでいる．今後解析データサイズは exabyte から zettabyte スケールのデータを扱う事が予想される．これまで大規模計算環境において広く使われているファイルシステムとしては Lustre, GPFS, PanFS[1)–3)] などがあげられる．これらのファイルシステムは計算ノードとは別にファイルシステムノードを前提とした設計となっている．スケールアウトする設計とはなっておらず，最大でも数 100GB/s が限界であると考えられる．そのため今後予想される大規模データ解析への対応は難しい．一方で，計算ノードのローカルストレージを束ねる設計のファイルシステムが提案されている．代表的なファイルシステムとして Gfarm ファイルシステム[4)] が挙げられる．Gfarm ファイルシステムは，計算ノードのローカルストレージを束ねることによってスケールアウトする設計となっている．これまでに，Gfarm ファイルシステムを用いた大規模データ解析に関する研究が行われている[5),6)]．また我々はこれまで Gfarm ファイルシステムのための大規模データ処理基板として MPI-IO/Gfarm の開発を行なっている[7)]．

大規模計算環境では単一のジョブがすべての計算資源を占有する場合は稀であり，複数のジョブが並列に実行される場合がほとんどである．これまでのファイルシステムでは，データは集中管理されているため，プロセスがアクセスするファイルの配置について考慮する必要はなかったが，Gfarm ファイルシステムのようなローカルストレージを束ねる設計のファイルシステムを大規模計算環境に導入した場合，ジョブがアクセスするファイルの配置が問題となってくる．ローカリティを考慮したプロセス配置のほか，仮に他のジョブが実行されているノードのストレージ上にアクセスする必要がある場合，計算に対してそのファイルアクセスが与える影響についても考慮したプロセススケジューリングが必要になると考えられる．

本稿では，上記の様に大規模計算環境に Gfarm ファイルシステムを用いた場合のプロセススケジューリングについて検討を行うため，I/O 処理が計算に与える影響について評価を行う．

本稿の構成は以下の様になっている．次章では Gfarm ファイルシステムと MPI-IO/Gfarm の概要，MPI-IO/Gfarm におけるプロセススケジューリングについて述べる．第 3 章で評価について述べ，結果について考察を行う．第 4 章で関連研究について述べ，最後にまとめと今後の課題を述べる．

2. Gfarm ファイルシステム

Gfarm ファイルシステム[4)] は計算ノードのローカルストレージをまとめ，1 つのネームスペースで管理することを可能にするファイルシステムである．ファイルアクセスをローカルストレージに対して分散させることでスケールアウトする設計となっている．Gfarm ファイルシステムの概要を図 1 に示す．Gfarm ファイルシステムは 1 つのメタデータサーバと複数のファイルサーバから構成される．メタデータサーバ (gfmd) では，ディレクトリ構造，ファイル属性，複製管理等を行っている．ファイルサーバ (gfsd) は，gfsd が動作しているノードのローカルストレージに対して実際のファイルアクセスを行う．ノード間での

†1 筑波大学大学院　システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†2 独立行政法人科学技術振興機構 CREST

ⓒ 2011 Information Processing Society of Japan

アクセスが行われる場合，Gfarm ライブラリを通じてリモートの gfsd に対してリクエストが行われ，その gfsd が自ノードのローカルストレージに対しアクセスを行い，レスポンスを返す．gfsd は基本的に計算ノード上で動作することを前提としている．計算プロセスがアクセスを行う場合，Gfarm では RTT と負荷状況から利用するストレージの選択が行われる．例えば新規ファイルを作成する場合においては容量の問題がなければ優先的にプロセスが動作するノードのローカルストレージ上にファイルの実体が作成される．Gfarm ファイルシステムではローカルストレージを活用し，ファイル複製を用いることによってスケーラブルなアクセス性能を実現している．しかし，Gfarm ファイルシステムではファイルのストライピングを行っていないため，複数のプロセスが 1 つのファイルに対して書き込みを行う場合，単一ストレージ性能に抑えられる問題がある．MPI-IO/Gfarm ではその問題点を解決する手法を MPI-IO の層で実装している．

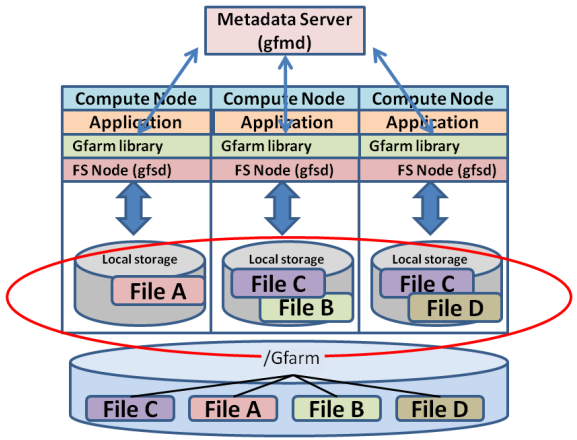


図 1 Gfarm ファイルシステムの概要

2.1 MPI-IO/Gfarm

MPI-IO/Gfarm は Gfarm に最適化された MPI-IO[8] の実装である．特に複数のプロセスが 1 つのファイルに対して書き込みを行う場合の最適化を行っている．MPI-IO/Gfarm では複数のプロセスが 1 つのファイルに対して書き込みを行う場合，プロセスごとに個別のファイルを作成し，プロセスが担当する範囲のデータをその個別のファイルに対して書き込みを行う．プロセスが個別のファイルを作成することで，Gfarm ファイルシステムではプロセスの動作するノードのローカルストレージにファイルの実体が作成される．そのためそれぞれのノードのローカルストレージに対して書き込み処理を分散させることが可能となる．また個別に作成されたファイルに対して，アプリケーションから透過的にアクセスを可能とする仕組みを用意している．MPI-IO によるファイルアクセスは File View と呼ばれるファイルに対するプロセスのアクセス範囲の定義が行われる．そこで MPI-IO/Gfarm では File View の情報をメタデータとして保存し，このメタデータから元ファイルのオフセットと個別ファイルのオフセットとの間の変換を自動的に行う．これまでに IOR，BTIO，HPIO などの並列ファイルアクセスのベンチマークを用いて評価を行った結果，複数のプロセスが 1 つのファイルに対して書き込みを行う場合においてもノード数に対してスケーラブルな書き込み性能を示した[7]．

2.2 MPI-IO/Gfarm におけるプロセススケジューリング

これまで用いられてきたファイルシステムはデータを集中管理しており，データ配置に関して考慮する必要はなかった．しかし，Gfarm ファイルシステムのような計算ノードのローカルストレージを束ねる設計のファイルシステムにおいてはデータローカリティを考慮したプロセス配置を行う必要がある．現状では MPI プロセスの配置をデータ配置を考慮して行う場合，ホストファイルを用いてユーザがプロセスとファイルのマッピングを行う手動で必要がある．大規模な環境で利用するため，プロセスとファイルのマッピングを自動で行う拡張が必要であると考えれる．

また大規模計算機環境においての適応を考えた場合に問題となるのは，複数のジョブが並列に実行される点である．問題点の概要について図 2 に示す．ファイル実体の配置によっては他のジョブが実行されているノードのローカルストレージに対してアクセスを行う可能性がある．このような場合，計算に対して裏で行われる I/O 処理がどの程度影響をあたえるのかについて明らかにする必要がある．またアクセス方法についても gfrep コマンドを用いたファイル複製の作成（ステージング）を行う場合や，直接読み込みを行う場合の影響の違いについても調査する必要があると考えられる．さらに近年プロセッサがメニーコア化しているため，I/O 処理等のためのコアを用意する事が可能になっていると考えれる．そのため I/O 処理用にコアを空けることによる影響についても明らかにする必要があると考えられる．

本稿では，MPI-IO/Gfarm におけるプロセススケジューリングを検討するために，I/O が計算に与える影響について評価を行う．

ⓒ 2011 Information Processing Society of Japan

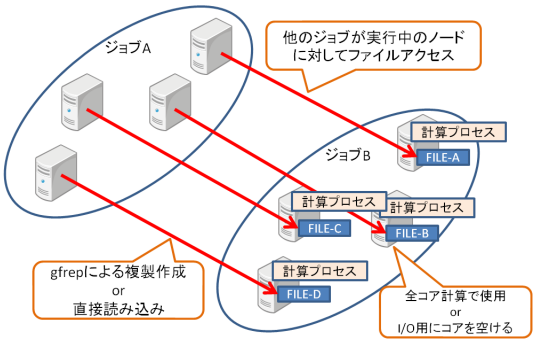


図 2 Gfarm ファイルシステムに対するプロセススケジューリングの問題点

3. 評　　価

3.1 評価環境

Intrigger プラットフォームの Tsukuba サイトを用いて評価を行った．評価環境を表 1 に示す．本環境では 1 ノード当たり 4 コアの CPU が 2 つあり，合計 8 コアである．Gfarm ファイルシステムのメタデータサーバは同サイト内で動作している．Bonnie++[9] を用いたローカルストレージに対する I/O の予備評価結果を表 2 に示す．

Hardware	
CPU	Xeon E5410 2.33GHz * 2
Memory	32GB
Kernel	2.6.26-2-amd64
Compiler	GCC version 4.3.2
NIC	10GigE
Disk	Hitachi HUA72101 1TB
Software	
Gfarm	2.4.2
MPI	mpich2-1.3.2
NPB	3.3

表 1 性能評価環境

3.2 評価方法

本評価では，以下のプロセス配置と I/O 処理の組み合せで評価を行う．プロセス配置については以下の 2 パターンを用いる．

bonnie++	
Seq Output	Ave:52.3 (Max:57.4 Min:47.6) [MByte/s]
Seq Input	Ave:63.5 (Max:69.8 Min:53.3) [MByte/s]

表 2 性能評価環境

- すべてのコア (8 コア) を計算プロセスが使用
- 7 コアを計算プロセスが使用して，1 コアを空ける

I/O 操作については以下の 3 パターンを用いる．

- I/O なし．
- gfrep コマンドを用いて，計算プロセスが動作しているノードのローカルストレージに存在するファイルを他のノードへ複製作成を行う．
- gfs_pio_read によって計算プロセスが動作しているノードに存在するファイルに対して別のノードから直接読み込みを行う．

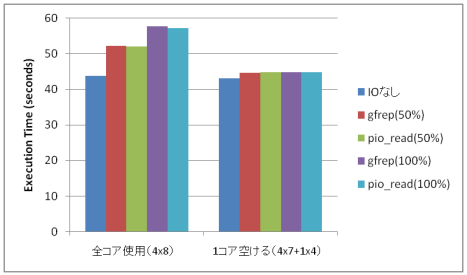
I/O については，実行時間の約 50%の間 I/O 処理が動作している場合 (約 1.5GByte のファイルに対する操作) と，実行時間の 100%の間 I/O 処理が動作している場合 (約 3.2GByte のファイルに対する操作) について評価を行う．読み込みを行うファイルは，キャッシュからクリアした状態で読み込みを行う．gfrep コマンドでのファイル複製では，対象ファイルは複製元のノードからバルク転送され指定されたノードのローカルストレージに対して書き込みを行う．gfrep コマンドでの I/O 性能は，転送とディスクへの書き込み時間を含んだ性能となってるが，今回の評価環境ではメモリの容量がファイルサイズに対して十分大きいため，キャッシュへの書き込みとなっている．gfs_pio_read による直接読み込みでは，4MByte のブロックサイズで読み込みを行なっている．データの送信はリクエスト毎に行われる．

3.3 ベンチマークソフト

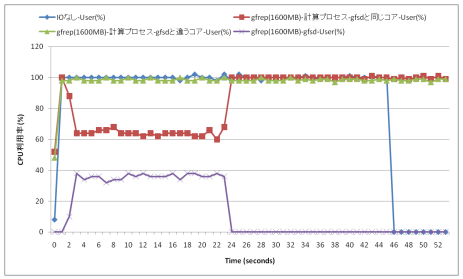
本評価では NAS Parallel benchmarks[10] の内，CG，FT，BT の 3 つのベンチマークを用いて評価を行う．CG ベンチマークは共役勾配法を用いた大規模疎行列の最小固有値を求めるもので，細かい一対一通信を頻繁に行うベンチマークとなっている．FT ベンチマークは FFT を用いて 3 次元微分方程式を解くもので，AlltoAll を用いた集合通信を行い，計算全体に対して通信が大きな割合を占める．BT ベンチマークは ADI 法を用いた CFD アプリケーションで他のベンチマークと違いに I/O 処理を行っている．一対一通信を行っているが通信部分は計算とオーバラップされている．

CG，FT ベンチマークについては問題サイズ CLASS=C を選択し，32 プロセスで測定

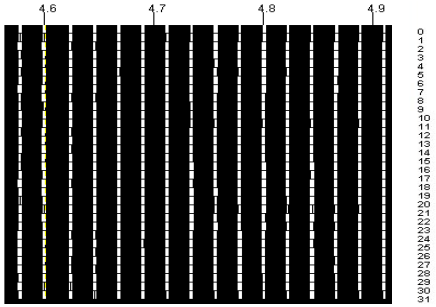
ⓒ 2011 Information Processing Society of Japan



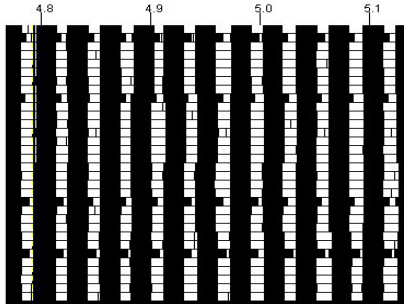
(a) 実行時間



(b) CPU 使用率の変化



(c) tlog を用いた実行プロファイル：I/O なし．黒が計算部分，白が通信部分．



(d) tlog を用いた実行プロファイル：I/O あり．黒が計算部分，白が通信部分．

図 3 評価結果：CG ベンチマーク

を行う．BT ベンチマークでは問題サイズ CLASS=B を選択し，36 プロセスで測定を行う．BT ベンチマークの CLASS=B では 1.6GByte のファイルを 1 つ作成する．MPI-IO/Gfarm を用いているため実際にはプロセス毎に個別のファイルが作成されてそれぞれのノードに対して分散して書き込みを行っている．メモリサイズが書き込みサイズに対して十分大きく，BT ベンチマークでは flush を行っていないためディスクキャッシュへの書き込みとなる．

3.4 評価結果

3.4.1 CG ベンチマーク

CG ベンチマークの評価結果を図 3 に示す．全コアを用いた場合は，4 ノードを用いてそれぞれ 8 プロセス (4x8) を起動して評価を行なっている．コアを空ける場合は，5 ノードを用いて 1 ノード最低 1 コアを空ける (4x7+1x4) ようにして測定を行った．

全体の実行時間の結果を図 3(a) に示す．全てのコアを計算プロセスが使用した場合，I/O なしで 43 秒，I/O が 50%の場合 52 秒 (約 20%増)，I/O が 100%の場合 57 秒 (約 30%増) となった．gfrep と gfs_pio_read とでは 0.2-0.5 秒程度の性能差となった．gfrep ではバルク転送を行っているため，gfs_pio_read に比べ性能が低くなっていると考えれるがほぼ同程度の性能となった．それに対してノード当たり 1 コアを空けて計算を行った場合，I/O なしで 43 秒，I/O が 50%の場合 44 秒 (約 2%増)，I/O が 100%の場合 44 秒 (約 2%増) となった．コアを空けた場合，I/O の時間にかかわらず性能低下を 2%程度に抑えられた．

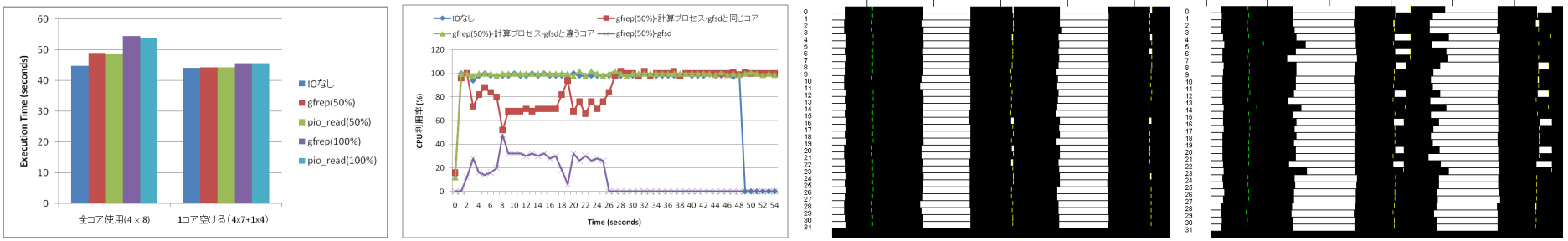
	計算なし	全コア使用	1 コア空き
gfrep	62.4MB/s	57.6MB/s	61.0MB/s
gfs_pio_read	59.8MB/s	58.4MB/s	58.5MB/s

表 3 I/O 評価結果：CG ベンチマーク

図 3(b) は計算プロセスと gfsd の CPU 利用率の 1 秒間平均の推移を示したものである．gfrep を用いた I/O の割合が 50%の場合で，一部のプロセスの CPU 利用率の時間変化を示している．gfsd が約 30%程 CPU を使用しており，gfsd と同一コアで動作している計算プロセスの性能が低下している．図 3(c), 3(d) は tlog[11] による実行プロファイル結果である．図 3(c) は全コアを使用して I/O を行わなかった場合の結果を示している．図 3(d) は全コアを使用し，gfrep による I/O 処理が計算の裏で動作している場合の結果を示している．黒が計算部分を示しており，白が通信部分を示している．cg ベンチマークにおける通信は一対一通信通信が行われている．図 3(d) においてプロセス番号 1，8，20，26 のプロセスが gfsd と同一コアに割り当てられたプロセスである．これらのプロセスの計算にかかる時間が長くなり，通信時に他のプロセスが待ち状態になっている．

表 3 に計算の裏で動作している読み込み性能を示す．それぞれ計算を行っていないノード，全コアを使用して計算を行なっているノード，1 コアを空けて計算を行なっているノードに対して gfrep を用いた複製作成，gfs_pio_read によるリモートノードからの読み込み処

ⓒ 2011 Information Processing Society of Japan



(a) 実行時間

(b) CPU 利用率

(c) tlog を用いた実行プロファイル：I/O なし．黒が計算部分，白が通信部分．

(d) tlog を用いた実行プロファイル：I/O あり．黒が計算部分，白が通信部分．

図 4 評価結果：FT ベンチマーク

理性能である．gfrep の場合，全コア使用の場合約 4%，1 コア空けた場合約 1%，I/O のみを行った場合に比べ性能が低下した．gfs_pio_read の場合，全コア使用した場合と 1 コア空けた場合での性能差は殆ど無く，I/O のみの結果に対して約 2%性能が低下している結果となった．ローカルディスク律速となっているため限定的ではあるが，I/O 用にコアを用意することで性能低下を防ぐことができる．

3.4.2 FT ベンチマーク

FT ベンチマークの評価結果を図 4 に示す．CG ベンチマークと同様に全コアを用いた場合は，4 ノードを用いてそれぞれ 8 プロセス (4x8) を起動して評価を行なっている．コアを空ける場合は，5 ノードを用いて 1 ノード最低 1 コアを空けて (4x7+1x4) 測定を行った．

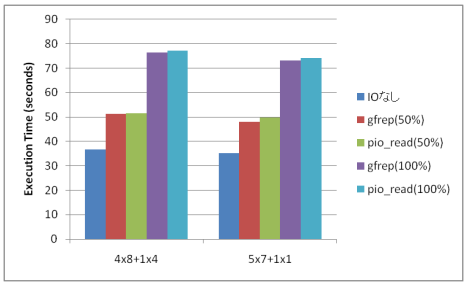
全体の実行時間の結果を図 4(a) に示す．全てのコアを計算プロセスが使用した場合，I/O なしで 45 秒，I/O が 50%の場合 48 秒 (約 7%増)，I/O が 100%の場合 54 秒 (約 20%増) となった．それに対してノード当たり 1 コアを空けて計算を行った場合，I/O なしが 44 秒，I/O が 50%の場合 44 秒 (約 0.6%増)，I/O が 100%の場合 45 秒 (約 2%増) となった．コアを空けた場合，I/O の時間にかかわらず性能低下を 0.6-2%程度に抑えられた．gfrep と gfs_pio_read とでは同程度の性能となった．図 4(b) は，計算プロセスと gfsd の CPU 利用率の 1 秒間平均の変移を示したものである．gfrep を用いた複製作成を行った場合で，I/O の割合は 50%の場合である．ばらつきはあるが gfsd が 30%前後の利用率となっている．CG ベンチマークと同様に，gfsd と同じコアに割り当てられたプロセスの性能が低下し，他のプロセスが通信待ちしてしまうため実行時間全体の足を引っ張っている．図 5(c)，5(d) は tlog を用いた実行プロファイルである．黄色はタイムステップの区切りを示している．黒は計算部分，白は通信部分である．計算部分が遅くなったプロセスによって通信待ちが発生している．

	計算なし	全コア使用	1 コア空き
gfrep	62.4MB/s	59.9MB/s	61.5MB/s
gfs_pio_read	59.8MB/s	58.6MB/s	59.3MB/s

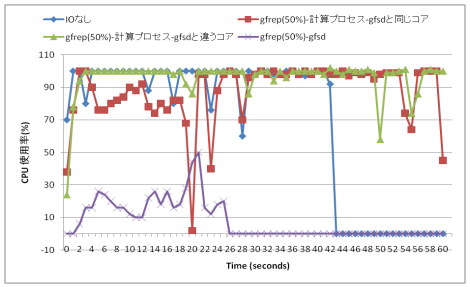
表 4 I/O 評価結果：FT ベンチマーク

表 4 に計算の裏で動作している読み込み性能を示す．それぞれ計算を行っていないノード，全コアを使用して計算を行なっているノード，1 コアを空けて計算を行なっているノードに対して gfrep を用いた複製作成，gfs_pio_read によるリモートノードからの読み込み処理性能である．gfrep の場合，全コアを使用した場合約 4%，1 コア空けた場合約 1%，I/O のみを行った場合に比べ性能が低下した．gfs_pio_read の場合においては全コアを使用した場合約 2%，1 コア空けた場合約 0.8%，I/O のみを行った場合に比べ性能が低下した．CG ベンチマークと同様に，I/O 用にコアを用意することで性能低下を防ぐことができる．

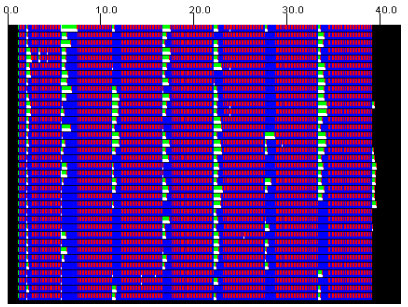
ⓒ 2011 Information Processing Society of Japan



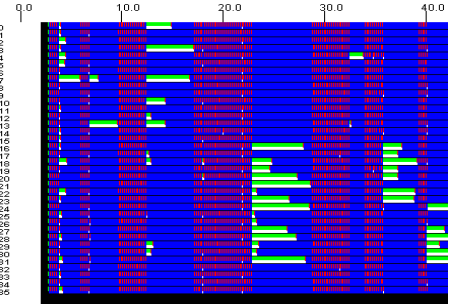
(a) 実行時間



(b) CPU 利用率



(c) tlog を用いた実行プロファイル：I/O なし．青が計算および通信部分，白，緑が IO 部分．



(d) tlog を用いた実行プロファイル：I/O あり．青が計算および通信部分，白，緑が IO 部分．

図 5 評価結果：BT ベンチマーク

3.4.3 BT ベンチマーク

BT ベンチマークの評価結果を図 5 に示す．全コアを用いた場合は，4 ノードを 8 プロセス，1 ノードを 4 プロセス (4x8+1x4) 起動して評価を行なっている．コアを空ける場合は，5 ノードを 7 プロセス，1 ノードを 1 プロセス (5x7+1x1) 起動して測定を行った．

全体の実行時間の結果を図 5(a) に示す．全コアを計算プロセスが使用した場合，I/O 無しで約 36 秒，I/O が 50%の場合約 51 秒 (42%増)，I/O が 100%の場合約 77 秒 (113%増) となった．gfrep と gfs_pio_read とでは 0.2-0.7 秒程度の性能差となりほぼ同程度の性能であった．ノード当たり 1 コアを空けて計算を行った場合，I/O なしが約 35 秒，I/O が 50%の場合約 49 秒 (40%増)，I/O が 100%の場合約 74 秒 (111%増) となった．gfrep と gfs_pio_read とでは，約 1 秒程度の性能差が見られた．BT ベンチマークの場合，I/O のためにコアを空けた場合においても性能はほとんど変化しないことがわかった．図 5(b) に CPU 利用率の推移，図 5(c)， 5(d) に tlog を用いた実行プロファイル結果を示す．tlog を用いたプロファイル結果では，青が計算及び通信を示し，白と緑が書き込み処理，赤はタイムステップの区切りを示している．図 5(b) に示すように，BT ベンチマークにおいても gfsd と同一コアに割り当てられたプロセスの性能低下が起こっていると考えられる．しかし BT ベンチマークにおいては図 5(c)， 5(d) に示すように，I/O 部分の処理時間が長くなっている部分が性能劣化の主な要因となっているため I/O ようにコアを空けたとしても性能は変わらない結果となった．一定間隔毎に BT ベンチマークの I/O が長く掛かっていが，これは pdflush の起動間隔と一致するため，キャッシュの追い出しが発生し I/O 待ちが発生していると考えれる．裏で他の I/O 処理が行われている場合，このキャッシュの追い出しと競合し，性能低下の主な原因となっていると考えられる．

	計算なし	全コア使用	1 コア空き
gfrep	62.4MB/s	54.2MB/s	55.5MB/s
gfs_pio_read	59.8MB/s	56.1MB/s	56.8MB/s
btio output	1029.8MB/s	586.9MB/s	687.6MB/s

表 5 I/O 評価結果：BT ベンチマーク

I/O 性能について表 5 に示す．“btio output” は BT ベンチマーク自体が行っている書き込み性能であり，“btio output” における “計算なし” で示されているのは，裏で I/O が動作していない場合の性能である．gfrep の場合，全コアを使用した場合約 15%，1 コア空けた場合約 12%，I/O のみを行った場合に比べ性能が低下した．gfs_pio_read の場合においては全コアを使用した場合約 7%，1 コア空けた場合約 5%，I/O のみを行った場合に比べ性能が低下した．BT ベンチマークの書き込み処理との競合が発生するため，CG と FT の結果に比べ性能は低下している．BT ベンチマーク自体の書き込み性能では，裏で I/O 処理

ⓒ 2011 Information Processing Society of Japan

が行われない場合に比べて，全コアを使用した場合で約 44%，1 コア空けた場合で 34%低下した．BT ベンチマークの書き込み処理はキャッシュへの書き込みとなり大きく影響をうけている．

4. 関連研究

関連研究としては，Hadoop におけるタスクスケジューリングが挙げられると考えられる．Hadoop では HDFS(Hadoop Distributed Filesystem)[12] と呼ばれる分散ファイルシステムが広く用いられる．HDFS では，ファイルをブロックに分割し配置する点など違いはあるものの，Gfarm ファイルシステムと同様に計算ノードのローカルストレージをまとめる構成をとる．Hadoop では 1 つのジョブを複数のタスクと呼ばれる単位に分割し処理を行う．ジョブ全体の管理する JobTracker がタスクを他のノードに対して振り分けを行う．この際 JobTracker はデータローカリティを考慮したタスク配置を行う．本研究においてはデータローカリティを考慮したプロセス配置を MPI のジョブに対して行うことが目的である．

5. おわりに

本稿では，MPI-IO/Gfarm におけるプロセススケジューリングの検討を行うために，I/O が計算に与える影響について評価を行った．NAS Parallel Benchmarks に含まれる CG，FT，BT の 3 つのベンチマークを用いて評価を行った結果，CG ベンチマークでは最大 30%，FT ベンチマークでは最大 20%，BT ベンチマークでは最大 114%実行時間が増加した．gfrep による複製作成を裏で行う場合と gfs_pio_read による直接読み込みを裏で行う場合とではほとんど性能差はなかった．I/O のためにコアを空けることで CG，FT については性能劣化を抑えることができたが，BT ベンチマークでは I/O 処理の競合による性能低下が支配的であるため I/O 用コアの効果は限定的であった．

今後の課題としては，MPI プロセスの起動時に指定されるホストファイル等の拡張を行い，プロセスとファイルとのマッピングを可能にする拡張を行う必要があると考えている．その際に今回の評価で得られた結果を考慮した最適化に関して検討を行う必要があると考えている．

謝辞 本研究の一部は，JST CREST「ポストペタスケールデータインテンシブサイエンスのためのシステムソフトウェア」および文科省次世代 IT 基盤構築のための研究開発「研究コミュニティ形成のための資源連携技術に関する研究」(データ共有技術に関する研究) による．

参 考 文 献

1) : Lustre, , available from ⟨http://www.lustre.org⟩
2) : GPFS, , available from ⟨http://www-03.ibm.com/systems/software/gpfs/index.html⟩
3) : PanFS, , available from ⟨http://www.panasas.com/panfs.html⟩
4) Osamu, Tatebe and Kohei, Hiraga and Noriyuki Soda: Gfarm Grid File System, *New Generation Computing, Ohmsha, Ltd. and Springer, Vol.28, No.3, pp.257-275, DOI: 10.1007/s00354-009-0089-5*, Vol.2007, No.122, pp.7–12 (online), available from ⟨http://ci.nii.ac.jp/naid/110006549609/en/⟩ (2010).
5) Naotaka Yamamoto and Osamu Tatebe and Satoshi Sekiguchi: Parallel and Distributed Astronomical Data Analysis on Grid Datafarm, *Proceedings of 5th IEEE/ACM International Workshop on Grid Computing (Grid 2004*, pp.461–466 (2004).
6) Shohei,Nishida and Nobuhiko, Katayama and Ichiro, Adachi and Osamu, Tatebe and Mitsuhisa, Sato and Taisuke, Boku and Akira Ukawa: High Performance Data Analysis for Particle Physics using the Gfarm file system, *Journal of Physics: Conference Series, 119, 062039, doi: 10.1088/1742-6596/119/6/062039* (2008).
7) 木村浩希，建部修見：広域分散ファイルシステム Gfarm の MPI-IO の実装と評価，情報処理学会研究報告. [ハイパフォーマンスコンピューティング] (2010-07-27).
8) Thakur, Rajeev and Gropp, William and Lusk, Ewing: On implementing MPI-IO portably and with high performance, *IOPADS '99: Proceedings of the sixth workshop on I/O in parallel and distributed systems*, New York, NY, USA, ACM, pp. 23–32 (online), DOI:http://doi.acm.org/10.1145/301816.301826 (1999).
9) Coker, R.: Bonnie++ File System Benchmark, , available from ⟨http://www.coker.com.au/bonnie++/⟩
10) Parkson Wong and Rob F. Van der Wijngaart: NAS Parallel Benchmarks I/O Version 2.4, *NAS Technical Report NAS-03-002* (2003).
11) : tlog, , available from ⟨http://www.ccs.tsukuba.ac.jp/workshop/HPCseminar/2008/software/tlog-0.9.tar.gz⟩
12) Apache: HDFS Architecture, , available from ⟨http://hadoop.apache.org⟩

© 2011 Information Processing Society of Japan