Vol.2017-HPC-162 No.9
2017/12/18

すばるHSCパイプラインのPwrake/Gfarmによる高速化手法の提案

田中 昌宏[1,a)]　建部 修見[1]　川島 英之[1]

概要：本研究の目的は，すばる望遠鏡に搭載された主焦点カメラ Hyper Suprime-Cam (HSC) の一次処理パイプラインの高速化である．処理内容は変更せず，数百コアを使用した場合のスケールアウト性能向上による高速化を目指す．HSC パイプラインの 2 種類のタスクに対してスケールアウト性能について評価する実験では，Frame タスクについては GPFS が 400 コア付近でスケーラビリティが限界となる一方，Gfarm ファイルシステムは 576 コアでもスケールアウトすることがわかった．また，36 ノード，全 576 コアを用いた実験では，実運用で用いられている従来手法から，並列処理フレームワークを Pwrake に，ファイルシステムを GPFS から Gfarm に変更することにより，処理全体で 1.71 倍の性能向上を確認できた．さらに，提案手法のタスクのオーバラップによるコア有効利用により，オーバーラップなしより性能が 1.26 倍向上し，実運用の環境からは 2.15 倍の性能向上を確認した．

1. はじめに

すばる望遠鏡に搭載されている主焦点カメラ Hyper Superime-Cam (HSC) [1] は，1.5 度角の範囲を一度に観測できる広い視野が特徴である．広視野をカバーするために，116 枚の CCD（うち 104 枚が観測用）が焦点面に配置されている．この広視野という特徴は，今年の 8 月 17 日に史上初めて観測された中性子星の合体による重力波源の対応天体の追跡観測においても活かされた．重力波およびガンマ線の信号から推定される広い天域を観測し，高確度での対応天体の特定に成功している [2].

広視野を持つ HSC は生成するデータ量も多く，一晩の観測で CCD が出力するデータ量は約 300GB に達する．生の CCD データに対して，装置の特性に由来する補正，明るさと位置のキャリブレーション，画像の重ね合わせ等の一次処理を行うことにより，サイエンスデータが得られる．HSC のチームが一次処理を行うパイプラインシステムを開発しており，これを用いて観測者がこのデータ処理を行う．特に突発天体の発見・追跡観測においては，迅速なデータ処理が，科学的な成果を左右する．そこで，多数の計算ノードを用いた並列処理により，できるだけ高速に処理することが求められている．

本研究の目的は，HSC パイプラインシステム [3] の処理内容を変更することなく，並列処理方法の改善によって高速化することである．そのためには HSC パイプライン処理の性能解析が不可欠である．前回の報告 [4] では，その予備調査を行った．その際，HSC パイプラインをワークフローシステムの Pwrake [5] で実行するための処理フローを構築し，Gfarm ファイルシステム [6] 上で並列分散実行した時のスケーラビリティについて調査した．この時は，実運用のシステムとの比較は行っていない．

本稿における貢献は，次の 2 点である．

(1) HSC パイプライン処理で使用している並列処理フレームワークと分散ファイルシステムの性能解析を行い，Pwrake および Gfarm の使用による性能向上を実証する．
(2) さらなるコア使用率の向上のため，異種類のタスクをオーバーラップして実行する手法を提案する．

（1）に関して，HSC パイプライン処理の実運用では，HSC パイプライン用に独自に開発された並列処理フレームワーク（本稿では MPIpool と呼ぶ）を用いている．5 節で述べる評価実験では，MPIpool と比較して，Pwrake がよりコアを密に使用できることを示す．さらに，IPMU に設置された実運用システムで使用されている GPFS に比べて，Gfarm ファイルシステムの高いスケーラビリティによって，パイプライン処理の実行時間を短縮できることを示す．

（2）に関して，本稿で提案するさらなる高速化手法は，異種類のタスクのオーバーラップによるコア使用率の向上である．この手法を可能にするには，タスクの依存関係を

[1] 筑波大学 計算科学研究センター
Center for Computational Sciences, University of Tsukuba
[a)] tanaka@hpcs.cs.tsukuba.ac.jp

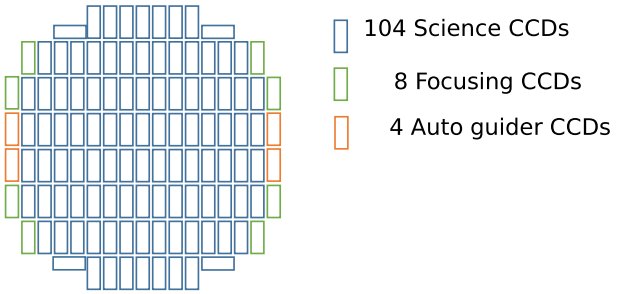

図 1 HSC 焦点面における CCD の並び

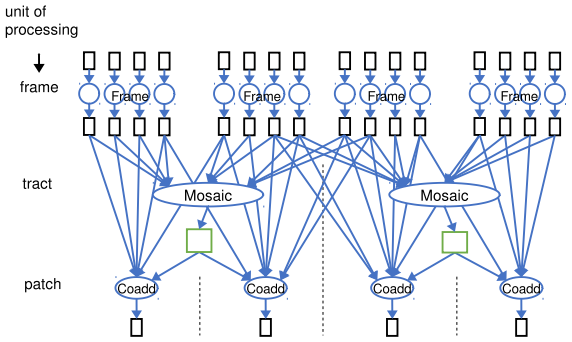

図 2 HSC パイプライン処理のグラフ (Detrend タスクは省略)

自由に構築できるワークフロー記述言語が必要であり，そのような記述言語を備えるワークフローシステムとして，Pwrake を用いる．そして，タスクのオーバーラップを可能にする記述方法について述べる．この提案手法を適用した結果，実行時間が短縮することを示す．

本稿の構成は以下の通りである．2 節で HSC パイプラインの概要について述べ，3 節で提案手法について述べる．4 節で評価方法について述べ，5 節で評価実験について述べる．6 節で関連研究，7 節でまとめと今後の課題について述べる．

2. HSC パイプラインの概要

前回の報告 [4] で，HSC 装置の概要と，パイプライン処理の詳細については述べた．本稿では再び簡単な概要と，前回からの変更点について述べる．

2.1 HSC の概要

HSC の焦点面には，図 1 に示すように，116 個の CCD が並び，104 個の CCD が観測用，8 個は合焦用，4 個はガイド用である．1 つの CCD のピクセル数は，観測に使用されないオーバースキャン領域を含めて，4272 × 2272 である．可視光から赤外線まで，g，r，i，z，および Y の 5 つの波長帯域フィルタを有する．

2.2 並列処理フレームワーク

HSC パイプラインソフトウェア [3] は，国立天文台，Kavli-IPMU，Princeton University のチームによって開発された．HSC パイプラインソフトウェアは主に C/C++と Python で実装されている．

HSC パイプラインには，計算機クラスタを使用した並列処理のための Python クラス hsc.pipe.base.pool が実装されている．これは，Python の multiprocess.Pool に基づいて実装された並列処理クラスであり，MPI を使用した分散処理用に拡張されている．この分散処理フレームワークに特に名前はないが，本稿では便宜的に *MPIpool* と呼ぶ．

2.3 HSC パイプライン処理

図 2 は，HSC パイプライン処理の流れのグラフを模式的に示した図である．丸い頂点はタスクを表し，正方形の頂点はファイルを表す．この図では，主要なタスクのみを表示しており，それ以外は省略している．また，それぞれのタスクは複数のファイルを出力するが，正方形の 1 つの頂点で表す，という省略もしている．

この図の前段階のタスクに，Detrend タスクがある．Detrend タスクは，CCD のデバイス特性を表す補正データを出力するタスクであり，Bias，Dark，Flat タスクなどを含む．Detrend の出力データは，Frame タスクの入力データとして使用する．Detrend の結果は，取得日の前後数週間の観測データに対して適用できるため，毎回実行する必要はないが，5.2 節のパイプライン性能評価では，Detrend タスクの処理時間が含まれている．

タスクによって並列化できる処理単位は異なる．Detrend タスクの単位は CCD，Frame タスクの単位は *frame*，Mosaic タスクの単位は *tract*，Coadd タスクの単位は *patch* である．frame は，1 ショットにおける CCD 1 枚分のデータである．tract および patch は，天球を分割する単位であり，自由に定義することができる．通常 tract は HSC の視野とほぼ同じ大きさであり，patch は tract を 10×10 に分けられた領域である．Coadd タスクの入力データとなるのは，Frame タスクの出力データのうち，Coadd タスクの patch 領域と重なる（複数個の）frame データである．

パイプライン処理を Rakefile（Pwrake におけるワークフロー定義ファイル）に記述する方法については，前回の報告 [4] で述べた通りである．今回は，処理内容にいくつかの変更がある．大きな変更は，tract の設定である．前回は tract を 1 つに設定したため，tract 毎のタスクである Mosaic タスクは 1 タスクのみであった．本稿では，戦略枠サーベイデータ [7] における処理と同様に，天球面を複数の tract に分割する方式を採用する．そのため，今回の処理フローでは，Mosaic タスクも図 2 に示すように複数のタスクとする．その他，hscPipe のバージョンアップに伴い，Coadd タスクの並列実行を行う stack.py コマンドに

おいてデフォルトの振る舞いが変更されたため，それに追随するように Coadd タスクの処理内容を変更した．

3. 提案手法

本稿で提案する高速化手法は，異種類のタスクのオーバーラップによるコア使用率の向上である．ここでオーバーラップの対象となるタスクは，Mosaic タスクと Coadd タスクである．擬似的なタスク定義文法（Rakefile とは異なる）を用いて，Mosaic タスクと Coadd タスクを定義すると，次のようになる．

```
for i in {tracts}
  deftask Mosaic(i)
endfor
deftask MosaicAll depends {Mosaic(i)| i ∈ tracts}
for j in {patches}
  deftask Coadd(j) depends MosaicAll
endfor
```

ここで，`deftask Mosaic(i)` は，i 番目の tract の Mosaic タスクを定義するという意味であり，`deftask Coadd(j) depends MosaicAll` は，MosaicAll の終了後に実行可能となる Coadd タスクを定義するという意味である．このタスク定義では，MosaicAll というタスクにおいて全ての Mosaic タスクの終了を待ち合わせるため，Mosaic と Coadd タスクのオーバーラップが行われないが，GNU make などのワークフロー言語でも記述可能である．一方，オーバーラップさせるためには，Coadd(j) と Mosaic(i) の依存関係に関する情報が必要になる．オーバーラップさせる定義方法は，次のように 2 重の for ループで書く．

```
for i in {tracts}
  deftask Mosaic(i)
  for j in {patches in tract(i)}
    deftask Coadd(j) depends Mosaic(i)
  endfor
endfor
```

ここで，観測領域と重なるパッチ番号のリスト `{patches in tract(i)}` の情報は，外部から読み込む必要がある．それを可能にするワークフロー言語が，Rakefile である．Rakefile は内部 DSL と呼ばれる特徴をもつ．内部 DSL とは，ホスト言語内に定義したドメイン特化言語である．Rakefile では，ホスト言語の Ruby の機能を全て利用できるから，二重ループも，外部ファイルの読み込みも自然に記述できる．Rakefile をワークフロー記述言語とする Pwrake は，後者のようなタスクをオーバーラップするワークフローを実行可能である．

実際のワークフローでは，Coadd タスクの依存関係に必要な情報は，tract と patch 番号だけではなく，visit（撮像ショット）と CCD 番号が必要となる．その情報を取得するために，HSC パイプラインに含まれるライブラリを利用したスクリプトを作成した．

表 1 評価環境

Cluster	Cluster at IPMU
CPU	Intel Xeon E5-2650 2.30GHz
# of cores/node	20 (use up to 16)
# of compute nodes	60 (use up to 36)
Main Memory	64 GiB
(mmfsd resident size	32 GiB)
Network	InfiniBand
OS	CentOS release 6.7
HSC pipeline	hscPipe 4.0.5
Gfarm	ver. 2.7.6
Ruby	ver. 2.4.1
Pwrake	ver. 2.2.5

表 2 データセット

	input		output	
DataSet	全サイズ	ファイル数	全サイズ	ファイル数
Detrend	41 GB	2,245	192 GB	6,439
WIDE	80 GB	4,368	1,183 GB	136,758

4. 評価方法

4.1 評価環境

評価環境を表 1 に示す．評価に用いたマシンは，Kavli-IPMU に配置されたクラスタである．計算ノード数は 60，各ノードに 20 コアの CPU を搭載する．そのうち本稿の実験では，最大 36 ノード，1 ノードあたり 16 コアを使用する．このクラスタでは，ノード間でファイルを共有する分散ファイルシステムとして，IBM Spectrum Scale（GPFS）[8] を使用している．評価では，GPFS と Gfarm File System（Gfarm FS）との比較を行う．Gfarm FS のローカルストレージとして，計算ノードの HDD を使用する．書き込み先をローカルストレージにするため，`gfarm2.conf` に `schedule_idle_load_thresh 100` と指定する．Gfarm メタデータサーバとして計算ノードとは別のマシンを使用し，バックエンド DB の性能の影響を最小限にするため，メタデータをメモリに保存する設定とする．

4.2 データセット

評価に使用するデータは，一晩に観測したデータのうち，i バンドのフィルターで観測した，WIDE という広く浅いフィールドのデータである．データ量としては，一晩の観測のうちの約 4 分の 1 である．各データセットの入出力データのサイズと数を表 2 に示す．

パイプライン処理の準備として，Gfarm FS の場合，これらの入力ファイルを処理ノードのストレージにラウンドロビンで格納しておく．測定前に，hscIngestImages.py コマンドを用いて入力ファイルをリポジトリに登録する．

表 3　ファイル Read/Write の統計

		read		write	
DataSet	task	データ量	回数	データ量	回数
Detrend	Bias	394 GB	5,204	47 GB	614
	Dark	411 GB	5,714	47 GB	614
	Flat	866 GB	14,386	96 GB	1,124
WIDE	Frame	693 GB	32,641	373 GB	43,880
	Mosaic	65 GB	43,844	655 MB	11,655
	Coadd	42 TB	258,151	804 GB	6,794

4.3　統計情報の取得

入出力データのプロファイル情報を取得するため，パイプラインプログラムに対して，ファイル I/O およびタスク実行についての情報をログに記録するコードを追加した．HSC パイプラインのデータファイルの入出力を管理するクラスは *butler* クラスである．butler クラスの読み書きを担当する関数の前後にパッチを挿入し，得た情報をログに出力する．出力する情報は，ファイルパス，ファイルサイズ，開始時刻と終了時刻，経過時間である．タスクを管理するクラスは *pool* クラスである．タスクの開始時刻と終了時刻をログに出力する．次の 5 節における評価は，これらのログ情報に基づいている．

butler によって記録されたファイルの read/write サイズの統計を，表 3 に示す．パイプライン処理では，1 つのファイルを繰り返して読むことがあるため，read サイズはファイルサイズより多くなる．

5.　評価実験

5.1　スケーラビリティの調査

本節では，パイプライン処理で処理時間の大半を占める Frame タスクと Coadd タスクにおけるスケーラビリティの調査について述べる．この測定では MPIpool ではなく Pwrake を使用する．測定条件は，全使用コア数を 48, 96, 144, 192, 288, 432, 576 とし，1 ノードあたりのコア数 (ppn, processers per node) を 4, 8, 12, 16 とする．ノード数（最大 36 ノード）は，コアの総数をノードあたりのコア数で割った数である．(3, 6 ノード)×16 コアの結果については，測定中エラー発生のため未取得である．

5.1.1　Frame task

図 3 (a) に，Frame タスクの経過時間を示す．この図は両対数プロットであることに注意．点線は GPFS を示し，実線は Gfarm を示す．赤い直線は，コア数と実行時間が反比例することを示す線であり，理想的にスケールアウトする場合はこの赤線と同じ傾きになることを表す．図 3 (a) を見ると，コア数が 288 以下の場合，処理時間は GPFS と Gfarm とでほぼ同じであるが，コア数が 432 以上の場合，GPFS はスケールアウトしなくなる一方，Gfarm はスケールアウトが続いている．

この理由を調べるために，各測定におけるバンド幅を

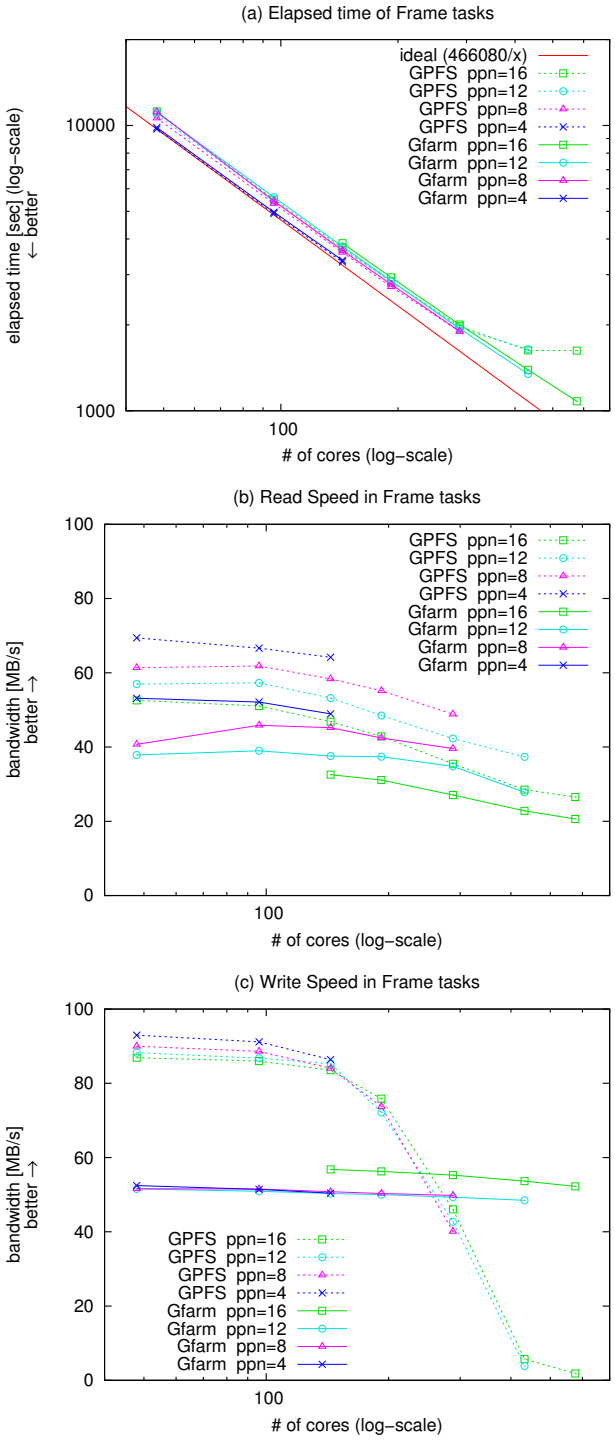

図 3　Frame task の実行時間とバンド幅．(ppn = processsors per node)

Figure 3 (b) と (c) に示す．これらのグラフは，横軸が対数，縦軸が線形でプロットしていることに注意．グラフの計測値は，butler クラスへのパッチにより取得したファイルサイズと経過時間に基づいており，バンド幅はファイルサイズの合計を経過時間の合計で割って得ている．図 3 (b) に，Frame タスクの Read 性能を示す．Read 性能の場合，GPFS のバンド幅は，どのコア数でも Gfarm のバンド幅よりも数割高いが，実行時間には影響していない．

図 3 (c) に，Frame タスクの Write 性能を示す．192 コア以下の場合，GPFS のバンド幅は Gfarm よりも高いが，288 コア以上の場合は GPFS のバンド幅が急に落ちている．この Write 性能の低下が，GPFS でスケールアウトを妨げる原因であると考えられる．GPFS では，ストレージサーバのストレージに書き込むため，コア数が多い場合，並列書き込みで競合が発生しているとみられる．一方，Gfarm はローカルストレージに書き込むため，書き込みの競合は起こりにくく，使用コア数が多い場合でも性能が低下していない．

5.1.2 Coadd task

図 4 (a) に，Coadd タスクの処理時間を示す．GPFS の測定では，コア数が増加しても処理時間は減少していない．つまりスケールしていないことがわかる．この原因は，図 4 (b), (c) の結果を見ると，コア数の増加とともに Read，Write のバンド幅がともに下がっており，GPFS のストレージの I/O 性能がボトルネックとなっているためであると考えられる．

一方，Gfarm の結果については，図 4 (a) から，処理時間はコア数よりもノード数に依存しているように見える．しかしその原因を，図 4 (b), (c) の I/O 性能の測定結果から特定することは難しい．これは，Coadd タスクが I/O バウンドなタスクであり，複数の要因が絡んでいることが考えられる．その要因として挙げられるのは，

- Gfarm のローカルストレージの I/O 性能
- ディスクキャッシュに乗る割合
- ネットワークの競合

の 3 つである．Gfarm のローカルストレージは各ノードに 1 個の HDD であり，書き込みについては，ローカルに書く設定であるため，ppn が増えて競合が発生するとボトルネックになるが，読み込みについては他のノードからのアクセスにも依存する．一方，ディスクキャッシュについては，ノード数が多いほど 1 ノードが持つファイルサイズが小さくなり，キャッシュアクセスの確率が増え，性能が向上する．ネットワークについては，ノード数が増えると，リモートのファイルを読む割合が増え，ネットワークの競合は発生しやすくなるが，書き込みについては，ローカルストレージを選択する設定により，ネットワークの影響は受けない．これらがどう関係しているかについては，測定条件によっても変化すると考えられる．

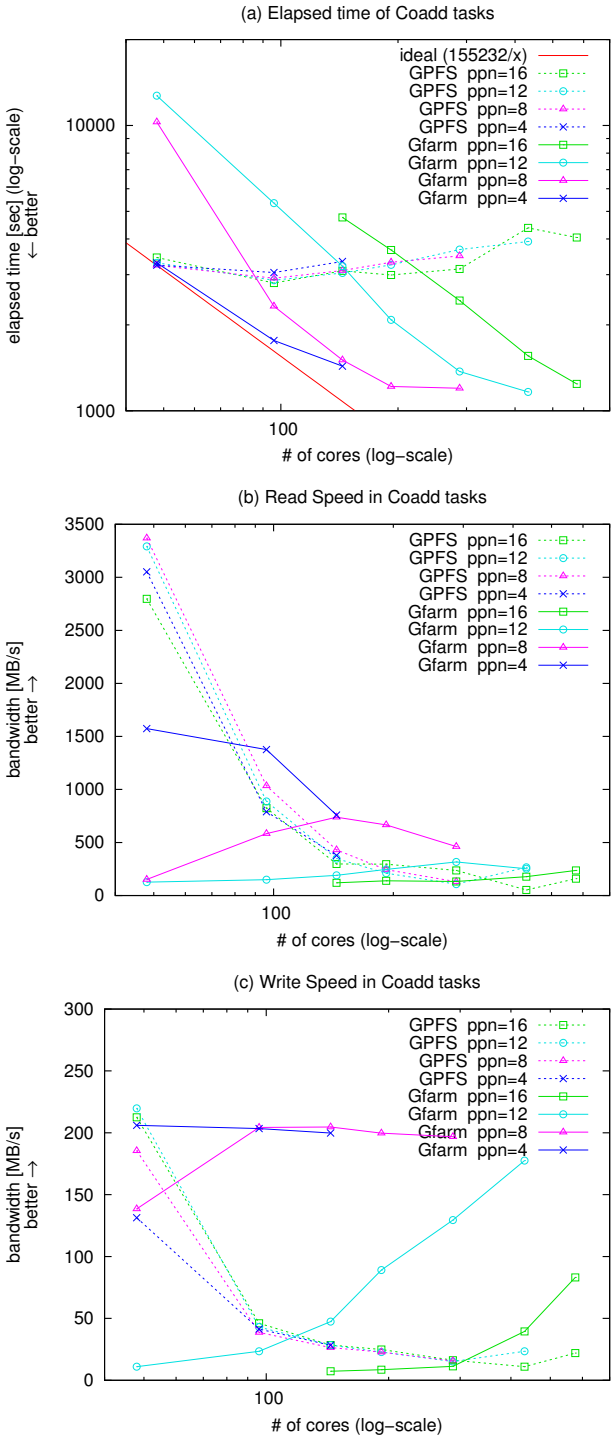

図 4 Coadd task の実行時間とバンド幅 (ppn = processsors per node)

5.2 コア使用率の調査

本節では，従来方法の MPIpool と GPFS に対する，Pwrake と Gfarm の使用による性能向上，および，提案手法のタスクオーバーラップによる性能向上について述べる．データセットは 5.1 節と同じ WIDE を使用し，測定はいずれも 36 ノード，ノードあたり 16 コア，合計 576 コアを使用する．次の 5 つの条件でパイプラインを実行した場合の，コア使用率の時系列プロットを図 5 に示す．

(1) MPIpool + GPFS
(2) Pwrake + GPFS
(3) Pwrake + Gfarm
(4) Pwrake + Gfarm (task overlap)
(5) Pwrake + Gfarm (task overlap, ppn=5 for Coadd)

(1) の測定では，MPIpool による並列化に加えて，バッチジョブシステムの Torque による並列化も行っている．と

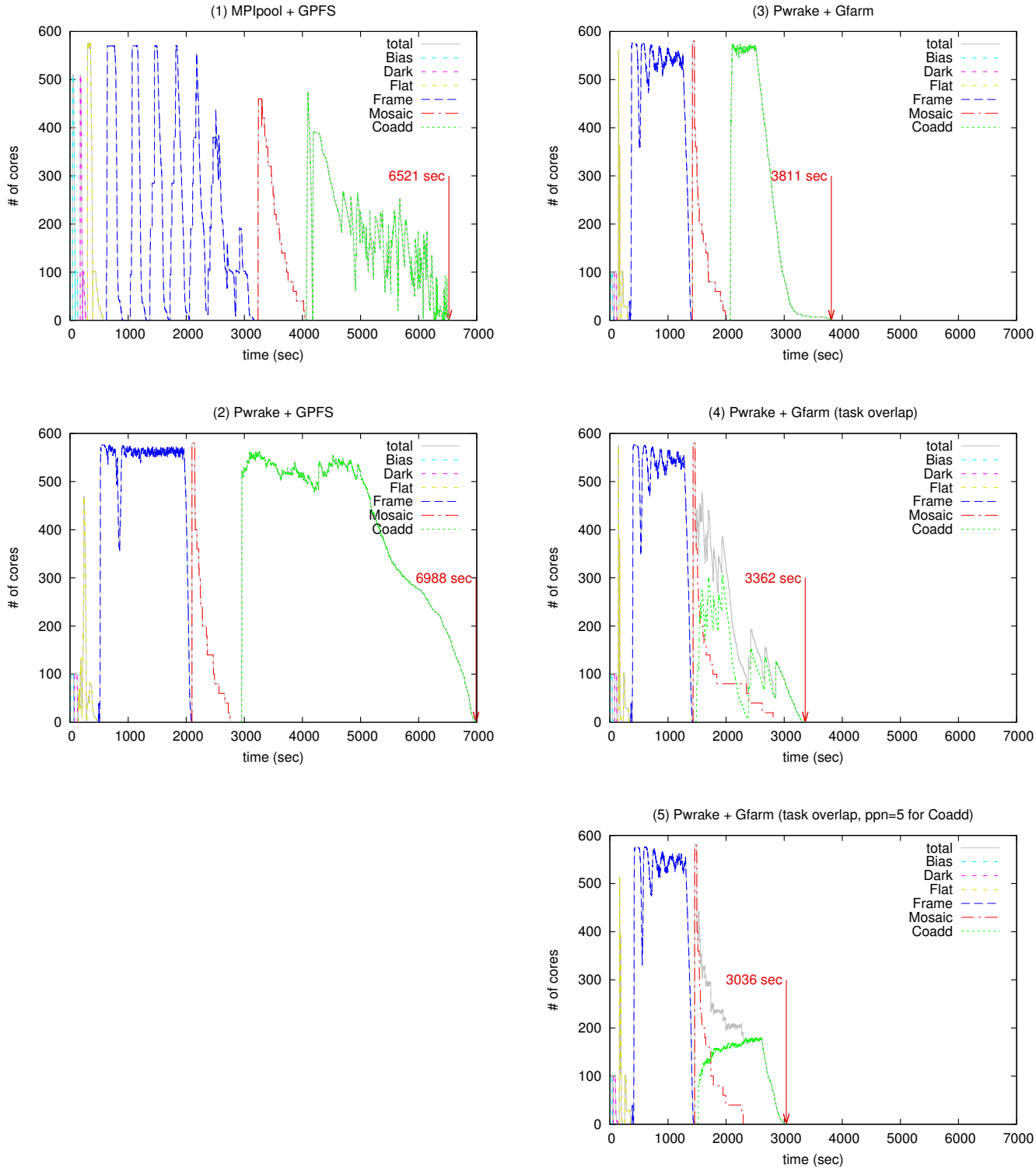

図 5 36 ノード ×16 コアでの測定における使用コア数の推移

いうのは，MPIpool の並列化は 112 までしかサポートしていないためである．このため，Frame タスクと Coadd タスクについては，tract ごとに分割したタスクを MPIpool で 96 並列で実行するジョブとし，それを Torque で 6 並列で実行することにより，576 並列を実現する．Mosaic タスクについては，1 ノード 20 コアで動作するタスクを Torque に投入している．

図 5 (1) のグラフを見ると，576 コアを有効に使用していないことがわかる．これは，MPIpool が，scatter-gather 方式で実装されているためである．この方式では，コア数と同じ数のタスクを開始した後，それらのタスクが全て完了するまで待ち，それから次のタスクを実行する．したがって，タスクの実行時間が揃っていなければ，待ち合わせが長くなり，コアの使用率が低くなる．

図 5 (2) は，(1) から MPIpool を Pwrake に変更した結果である．Pwrake によってコア使用率が上がり，Frame タスクの経過時間が短くなっている．しかし，Coadd タスクについては経過時間が増えており，結果的に (1) のケースよりトータルの実行時間が若干増えている．これは，5.1.2 節で見たように，Coadd タスクが GPFS ではスケールしないためである．

図 5 (3) は，(2) から GPFS を Gfarm に変更した結果である．(1) と比較して，性能は 1.71 倍向上した．これは Gfarm のスケールアウト性能と，Pwrake のコア使用率によるものである．

図 5 (4) は，(3) から提案手法のタスクオーバーラップを取り入れた結果である．(3) より性能は 1.13 倍向上している．しかし，Mosaic タスクの実行時間が増えたため，一部の Coadd タスクの開始が遅れ，結果的に処理終了も遅れるという現象が見られ，提案手法の効果はそれほど大きくはない．Mosaic タスクの詳細なプロファイルを見ると，ファイル読み込みの時間が増えていることがわかった．これは，I/O バウンドの Coadd タスクが別のノードで同時に走っているため，干渉が起きていることが考えられる．

図 5 (5) は，(4) から Coadd タスクのみ ppn を 5 に減らすという条件で測定した結果である．Coadd の影響を減らすことによって，Mosaic タスクの実行時間が (4) よりも減少し，トータルの実行時間も測定条件中で最小となった．(5) の測定では，オーバーラップがないケース (3) と比較して，性能が 1.26 倍向上し，実運用と同じケース (1) と比較すると 2.15 倍の性能向上を達成した．

6. 関連研究

Montage [9] は，天文データ処理の並列化の例として頻繁に挙げられるアプリケーションである．Montage は，一次処理済みの天文画像を加工するための汎用ソフトウェアパッケージである．Montage は，Pegasus [10]，Swift [11]，Pwrake [5] などの多くのワークフローシステムのベンチマークに使用されている．一方，HSC パイプラインは，特定の検出器が出力した生データを一次処理するためのプログラムである．本研究の目的は，一次処理システムの性能を，並列処理によって向上させることである．

天文観測データ処理用のパイプラインシステムは，これまでにも，SDSS 望遠鏡 [12] などでも開発されている．しかし，今後の観測装置はますます大量のデータを生成するようになり，大規模な計算機クラスタを用いた並列処理と分散処理は不可欠である．

Yamamoto ら [13] は，Gfarm バージョン 1 を用いて Subaru-Cam データの並列処理を提案した．この時点で Gfarm はバージョン 1 であり，異なるプロトタイプの実装であったため，システムコールと gfrun のフックを使用する必要があった．一方，現在の Gfarm バージョン 2 では，gfarm2fs で Gfarm FS をマウントし，ローカルファイルシステムを変更せずに使用するプログラムを実行することができる．本研究では，Gfarm バージョン 2 における高速化を行った．

7. まとめと今後の課題

本研究の目的は，HSC 用に開発された一次データ処理パイプラインの高速化である．提案手法のタスクのオーバラップによるコアの有効利用については，内部 DSL によるワークフロー記述を持つワークフローシステム Pwrake によって可能となる．1 つ目の評価実験では，GPFS と Gfarm FS におけるパイプライン処理のスケールアウト性能について調査した．その結果，評価環境では，GPFS のスケールアウトの限界が Frame タスクで 400 コア付近，Coadd タスクで 48 コア以下であった．一方，Gfarm ファイルシステムは，Frame タスクで 576 コアまでスケールし，Coadd タスクで ppn=8 で 200 コア付近までスケールする，という結果が得られた．2 つ目の評価実験として，全 576 コア（36 ノード）を用いて，コアの使用率と全体の実行時間について調査した．その結果，HSC パイプラインに実装されたタスク並列フレームワーク（本稿では MPIpipe と呼ぶ）のコア使用効率が低いことがわかった．また，MPIpipe を Pwrake に，ファイルシステムを Gfarm に置き換えると，1.71 倍性能が向上することが確認できた．さらに，提案手法のタスクのオーバラップによるコア有効利用と，I/O の干渉による性能低下を防ぐように ppn を調整することにより，性能が 1.26 倍向上し，実運用の環境から 2.15 倍の性能向上を確認できた．

今後の課題は，Coadd タスクのように I/O バウンドなタスクを効率的に実行する手法である．

謝辞 本研究は，JST CREST「広域撮像探査観測のビッグデータ分析による統計計算宇宙物理学」，「EBD：次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」の支援により行った．

参考文献

[1] Miyazaki, S., Komiyama, Y., Nakaya, H., Kamata, Y., Doi, Y., Hamana, T., Karoji, H., Furusawa, H., Kawanomoto, S., Morokuma, T., Ishizuka, Y., Nariai, K., Tanaka, Y., Uraguchi, F., Utsumi, Y., Obuchi, Y., Okura, Y., Oguri, M., Takata, T., Tomono, D., Kurakami, T., Namikawa, K., Usuda, T., Yamanoi, H., Terai, T., Uekiyo, H., Yamada, Y., Koike, M., Aihara, H., Fujimori, Y., Mineo, S., Miyatake, H., Yasuda, N., Nishizawa, J., Saito, T., Tanaka, M., Uchida, T., Katayama, N., Wang, S.-Y., Chen, H.-Y., Lupton, R., Loomis, C., Bickerton, S., Price, P., Gunn, J., Suzuki, H., Miyazaki, Y., Muramatsu, M., Yamamoto, K., Endo, M., Ezaki, Y., Itoh, N., Miwa, Y., Yokota, H., Matsuda, T., Ebinuma, R. and Takeshi, K.: Hyper Suprime-Cam, *SPIE Astronomical Telescopes + Instrumentation* (McLean, I. S., Ramsay, S. K. and Takami, H., eds.), International Society for Optics and Photonics, p. 84460Z (online), DOI: 10.1117/12.926844 (2012).

[2] Tominaga, N., Tanaka, M., Morokuma, T., Utsumi, Y., Yamaguchi, M. S., Yasuda, N., Tanaka, M., Yoshida, M., Fujiyoshi, T., Furusawa, H., Kawabata, K. S., Lee, C.-H., Motohara, K., Ohsawa, R., Ohta, K., Terai, T., Abe, F., Aoki, W., Asakura, Y., Barway, S., Bond, I. A., Fujisawa, K., Honda, S., Ioka, K., Itoh, Y., Kawai, N., Kim, J. H., Koshimoto, N., Matsubayashi, K., Miyazaki, S., Saito, T., Sekiguchi, Y., Sumi, T. and Tristram, P. J.: Subaru Hyper Suprime-Cam Survey for An Optical Counterpart of GW170817, (online), available from ⟨http://arxiv.org/abs/1710.05865⟩ (2017).

[3] Furusawa, H., Tanaka, M., Yasu, Y., Suzuki, S. Y., Itoh, R., Katayama, N., Komiyama, Y., Miyazaki, S., Utsumi, Y., Uchida, T., Aihara, H. and Yasuda, N.: Hyper Suprime-Cam: data analysis and management system, *SPIE Astronomical Telescopes + Instrumentation* (Brissenden, R. J. and Silva, D. R., eds.), International Society for Optics and Photonics, pp. 70161F–70161F–11 (online), DOI: 10.1117/12.788375 (2008).

[4] 田中 昌宏, 建部 修見, 川島 英之, 村田直郁：すばる HSC データ処理の性能調査と Gfarm/Pwrake の適用, 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 (HPC), Vol. HPC-150, No. 36, pp. 1–8（オンライン）, 入手先 ⟨http://id.nii.ac.jp/1001/00144601/⟩ (2015).

[5] Tanaka, M. and Tatebe, O.: Pwrake: A parallel and distributed flexible workflow management tool for wide-area data intensive computing, *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10)*, New York, New York, USA, ACM Press, pp. 356–359 (online), DOI: 10.1145/1851476.1851529 (2010).

[6] Tatebe, O., Hiraga, K. and Soda, N.: Gfarm Grid File System, *New Generation Computing*, Vol. 28, No. 3, pp. 257–275 (online), DOI: 10.1007/s00354-009-0089-5 (2010).

[7] Aihara, H., Arimoto, N., Armstrong, R., Arnouts, S., Bahcall, N. A., Bickerton, S., Bosch, J., Bundy, K., Capak, P. L., Chan, J. H. H., Chiba, M., Coupon, J., Egami, E., Enoki, M., Finet, F., Fujimori, H., Fujimoto, S., Furusawa, H., Furusawa, J., Goto, T., Goulding, A., Greco, J. P., Greene, J. E., Gunn, J. E., Hamana, T., Harikane, Y., Hashimoto, Y., Hattori, T., Hayashi, M., Hayashi, Y., Hełminiak, K. G., Higuchi, R., Hikage, C., Ho, P. T. P., Hsieh, B. C., Huang, K., Huang, S., Ikeda, H., Imanishi, M., Inoue, A. K., Iwasawa, K., Iwata, I., Jaelani, A. T., Jian, H. Y., Kamata, Y., Karoji, H., Kashikawa, N., Katayama, N., Kawanomoto, S., Kayo, I., Koda, J., Koike, M., Kojima, T., Komiyama, Y., Konno, A., Koshida, S., Koyama, Y., Kusakabe, H., Leauthaud, A., Lee, C. H., Lin, L., Lin, Y. T., Lupton, R. H., Mandelbaum, R., Matsuoka, Y., Medezinski, E., Mineo, S., Miyama, S., Miyatake, H., Miyazaki, S., Momose, R., More, A., More, S., Moritani, Y., Moriya, T. J., Morokuma, T., Mukae, S., Murata, R., Murayama, H., Nagao, T., Nakata, F., Niida, M., Nikura, H., Nishizawa, A. J., Obuchi, Y., Oguri, M., Oishi, Y., Okabe, N., Okura, Y., Ono, Y., Onodera, M., Onoue, M., Osato, K., Ouchi, M., Price, P. A., Pyo, T. S., Sako, M., Okamoto, S., Sawicki, M., Shibuya, T., Shimasaku, K., Shimono, A., Shirasaki, M., Silverman, J. D., Simet, M., Speagle, J., Spergel, D. N., Strauss, M. A., Sugahara, Y., Sugiyama, N., Suto, Y., Suyu, S. H., Suzuki, N., Tait, P. J., Takata, T., Takada, M., Tamura, N., Tanaka, M. M., Tanaka, M., Tanaka, M., Tanaka, Y., Terai, T., Terashima, Y., Toba, Y., Toshikawa, J., Turner, E. L., Uchida, T., Uchiyama, H., Umetsu, K., Uraguchi, F., Urata, Y., Usuda, T., Utsumi, Y., Wang, S. Y., Wang, W. H., Wong, K. C., Yabe, K., Yamada, Y., Yamanoi, H., Yasuda, N., Yeh, S., Yonehara, A. and Yuma, S.: The Hyper Suprime-Cam SSP Survey: Overview and Survey Design, (online), available from ⟨http://arxiv.org/abs/1704.05858⟩ (2017).

[8] Schmuck, F., Schmuck, F. and Haskin, R.: GPFS: A Shared-Disk File System for Large Computing Clusters, *IN PROCEEDINGS OF THE 2002 CONFERENCE ON FILE AND STORAGE TECHNOLOGIES (FAST*, pp. 231—-244 (2002).

[9] Jacob, J. C., Katz, D. S., Berriman, G. B., Good, J. C., Laity, A. C., Deelman, E., Kesselman, C., Singh, G., Su, M.-H., Prince, T. A. and Williams, R.: Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking, *International Journal of Computational Science and Engineering*, Vol. 4, No. 2, pp. 73–87 (online), DOI: 10.1504/IJCSE.2009.026999 (2009).

[10] Singh, G., Su, M.-H., Vahi, K., Deelman, E., Berriman, B., Good, J., Katz, D. S. and Mehta, G.: Workflow task clustering for best effort systems with Pegasus, *Proceedings of the 15th ACM Mardi Gras conference on From lightweight mash-ups to lambda grids: Understanding the spectrum of distributed computing requirements, applications, tools, infrastructures, interoperability, and the incremental adoption of key c*, No. c, New York, New York, USA, ACM Press, p. 1 (online), DOI: 10.1145/1341811.1341822 (2008).

[11] Zhao, Y., Hategan, M., Clifford, B., Foster, I., von Laszewski, G., Nefedova, V., Raicu, I., Stef-Praun, T. and Wilde, M.: Swift: Fast, Reliable, Loosely Coupled Parallel Computation, *2007 IEEE Congress on Services (Services 2007)*, pp. 199–206 (online), DOI: 10.1109/SERVICES.2007.63 (2007).

[12] Lupton, R. H., Ivezic, Z., Kent, S., Gunn, J. E. and Knapp, G. R.: The SDSS Imaging Pipelines, *Astron. Soc. Pac. Conf. Proc.*, Vol. 10, p. 269 (online), available from ⟨http://cds.cern.ch/record/484817⟩ (2001).

[13] Yamamoto, N., Tatebe, O. and Sekiguchi, S.: Parallel and Distributed Astronomical Data Analysis on Grid Datafarm, *Fifth IEEE/ACM International Workshop on Grid Computing*, IEEE, pp. 461–466 (online), DOI: 10.1109/GRID.2004.47 (2004).