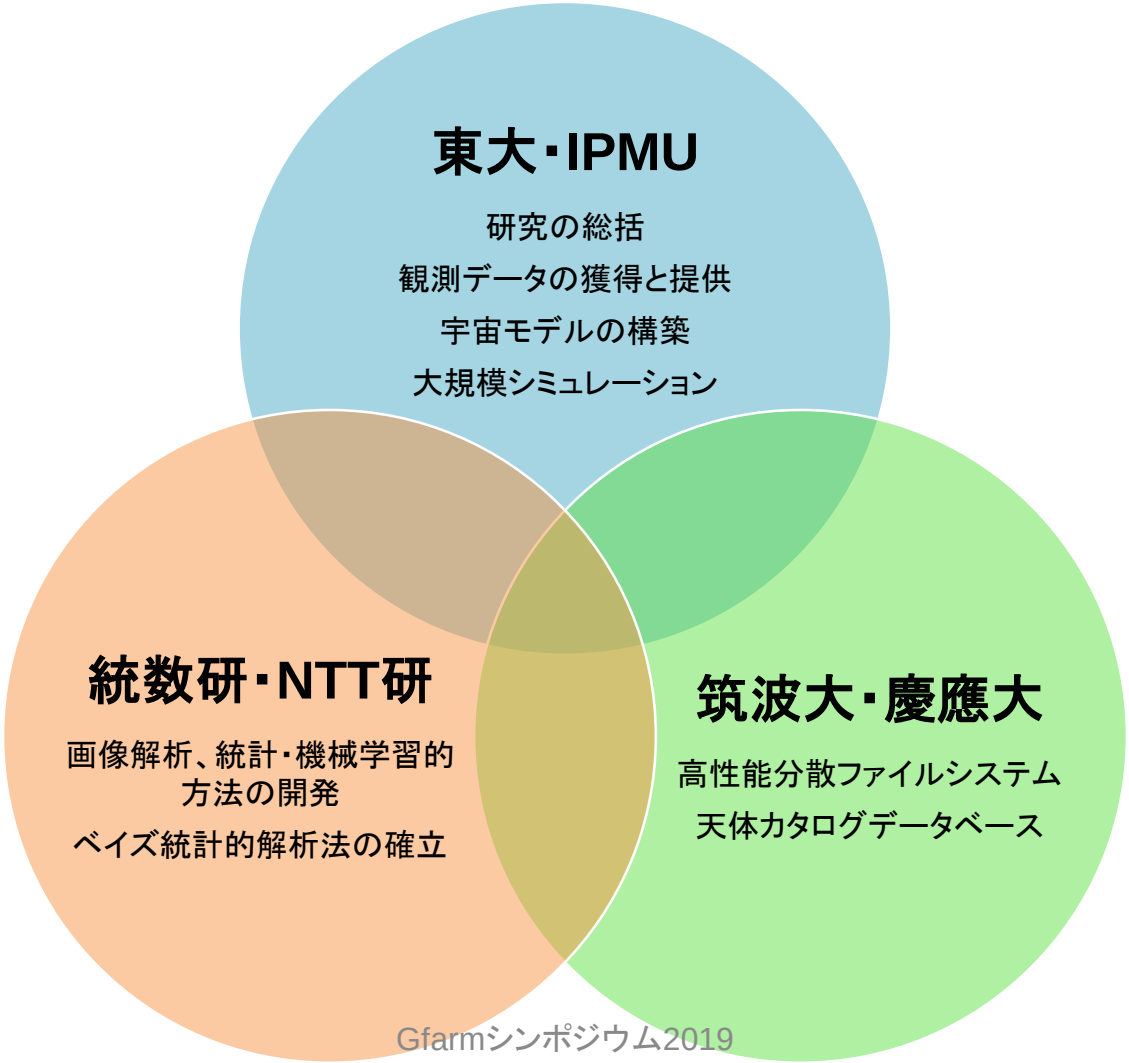
Pwrake/Gfarmによるすばる望遠鏡データの大規模処理

田中 昌宏
慶應義塾大学

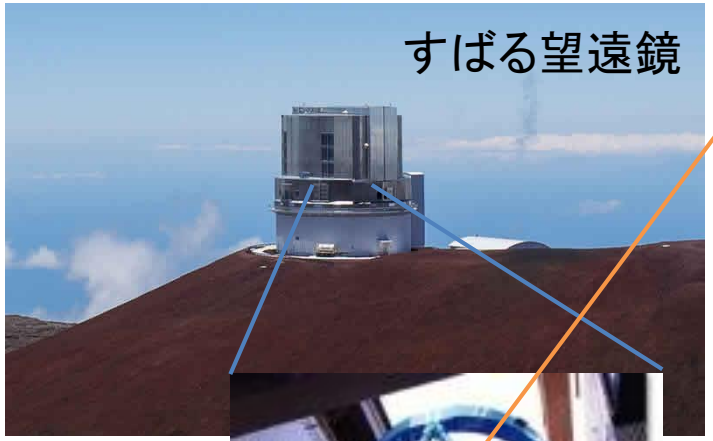
本研究の支援

- 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」
  - https://www.ipmu.jp/ja/node/2014
  - 研究代表者: 吉田 直紀（Kavli IPMU, 東京大）
- 目的
  - 「統計計算宇宙物理学」の開拓
  - Subaru/HSC 5年間の観測による膨大な画像データの取得
  - 最新の機械学習と統計数理、大規模コンピューターシミュレーションを駆使した解析
  - 宇宙のダークマター分布を3次元の解明
  - 大規模化する天文ビックデータと情報統計学を融合させた新領域で次世代アプリケーション技術を開発

CRESTプロジェクト担当研究機関



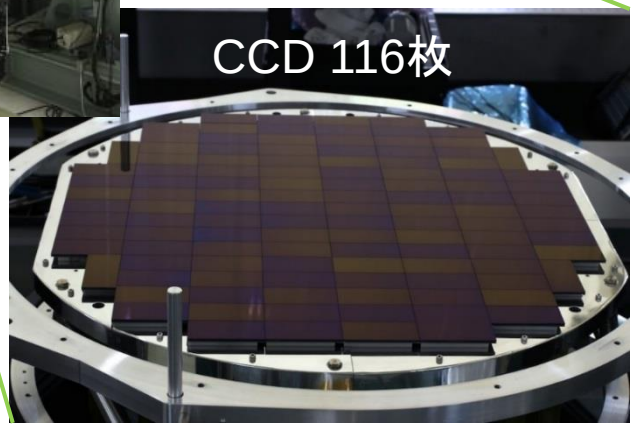
すばる Hyper Suprime-Cam



(画像クレジット:
国立天文台)

HSC

(画像クレジット:
国立天文台・HSC プロジェクト)



HSCの特徴：広い視野

図2: すばる望遠鏡に当初から搭載されている Suprime-Cam (左下、中央) と、今回 HSC (右) が写し出したアンドロメダ銀河 M31 の視野の比較。黄色い枠は過去に Suprime-Cam で撮影された領域を示します。左上には月の典型的な見かけの大きさが示されています。(クレジット：国立天文台)

(https://subarutelescope.org/Topics/2013/07/30/j_index.html より引用)

Hyper Suprime-Cam Subaru Strategic Program (HSC SSP)

- 2014年から5-6年間、300夜の割り当て
- 遠方銀河の観測により、宇宙論や銀河の進化など幅広い研究に活用

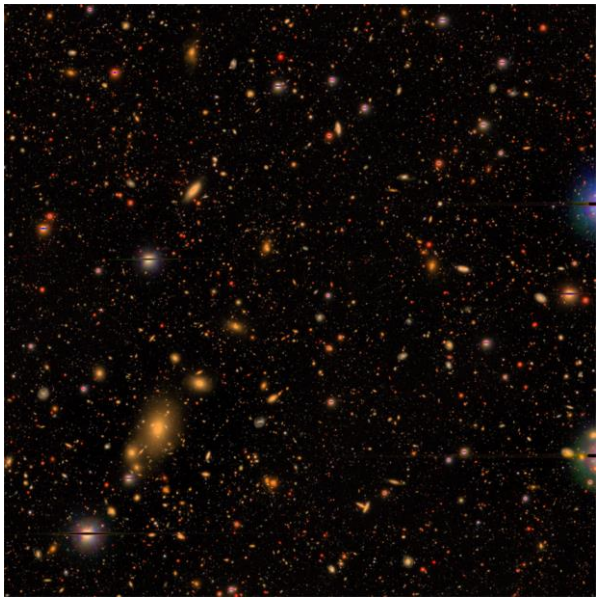


図1: HSC-SSP で観測された COSMOS 領域 (ろくぶんぎ座方向) の g, r, i バンドの三色合成画像。1000 以上もの銀河が含まれており、距離は数十億光年です。画像の中の最も遠い銀河は、宇宙誕生後 10 億年以内に形成されたものです。(クレジット:プリンストン大学/HSC Project)

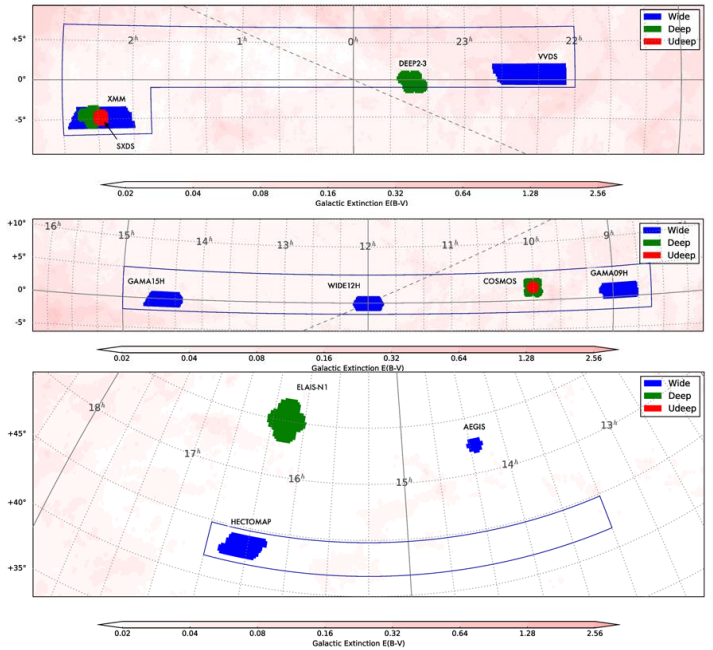


図4: 天球図に観測領域を示したもの。青がワイド、緑がディープ、赤がウルトラディープ領域を示しています。(クレジット:国立天文台/HSC Project)

(https://subarutelescope.org/Topics/2017/02/27/j_index.html より引用)

HSC SSP Transient Survey in COSMOS[1]

- 観測期間：2016年11月～2017年4月
- 観測領域：COSMOS field（HSC5視野分）
- 1,824個の超新星候補を発見（左図の赤い点）
- CRESTプロジェクト成果の機械学習の手法を活用

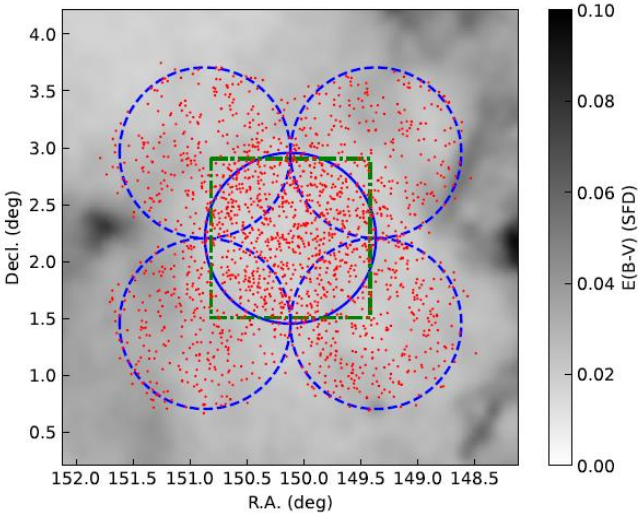


Fig. 2. Pointing layout on the sky (ultra-deep: blue (solid), deep: blue (dashed), original COSMOS (Scoville et al. 2007) coverage: green (dash dot)) overlaid on an SFD (Schlegel et al. 1998) reddening map. Positions of detected supernova (SN) candidates are indicated by red points. Given that we were dithering around fiducial pointings, the actual coverage is wider than that indicated by the dashed blue line. Some SN candidates are detected in those area.

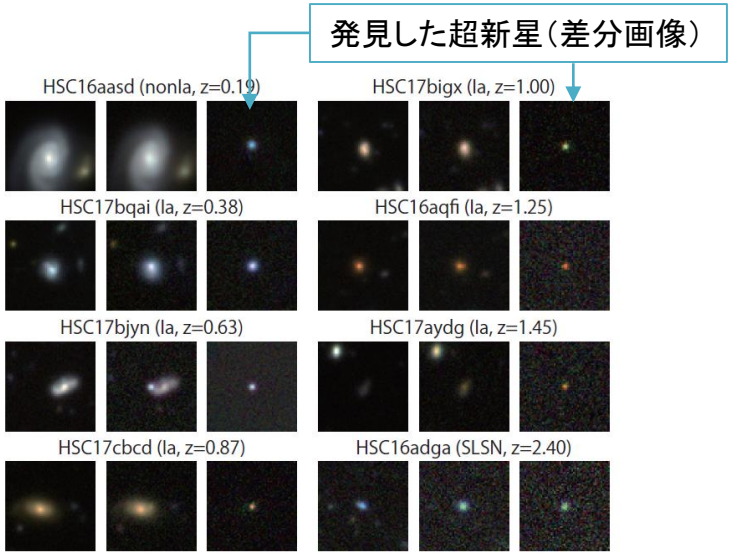


Fig. 14. Images of SN candidates at various redshifts. Redshifts are spec-z, except for HSC17aydg (HSC photo-z) and HSC16adga (COSMOS photo-z). Three panels are shown for an SN: reference (left), new image (middle), and subtracted image (right). Three filter-bands (r2-, i2-, and z-bands) make up this color composite.

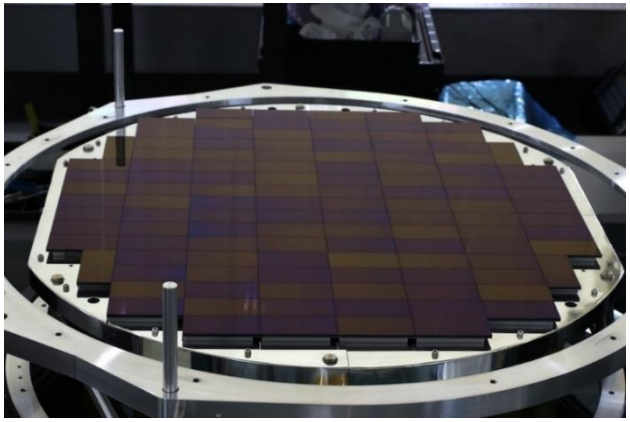
[1] Yasuda, N. et al. (2019). The Hyper Suprime-Cam SSP transient survey in COSMOS: Overview. *Publications of the Astronomical Society of Japan*.

高速データ処理の必要性

- 早くデータを見たい
  - 超新星のフォローアップ観測（暗くなる前に）
  - ショックブレイクアウト（増光中の超新星）なら大発見
- 既存データの再解析
  - パイプラインのバージョンアップ
- HSCデータ解析処理（パイプライン）
  - CCDが出力した生データを解析処理し、サイエンスデータに変換
  - 一晩の生データ量：300 GB（目安）
  - 処理後のデータ量：およそ10倍
  - 伝聞では、一晩のデータの処理に半日くらい
- 本研究の目的：HSCデータ解析処理の高速化

Overview of HSC Pipeline

HSCの装置概要

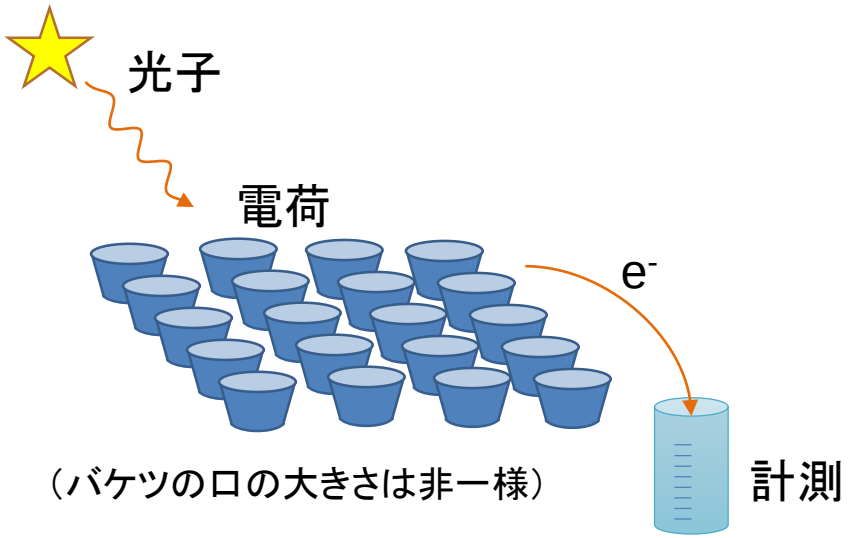


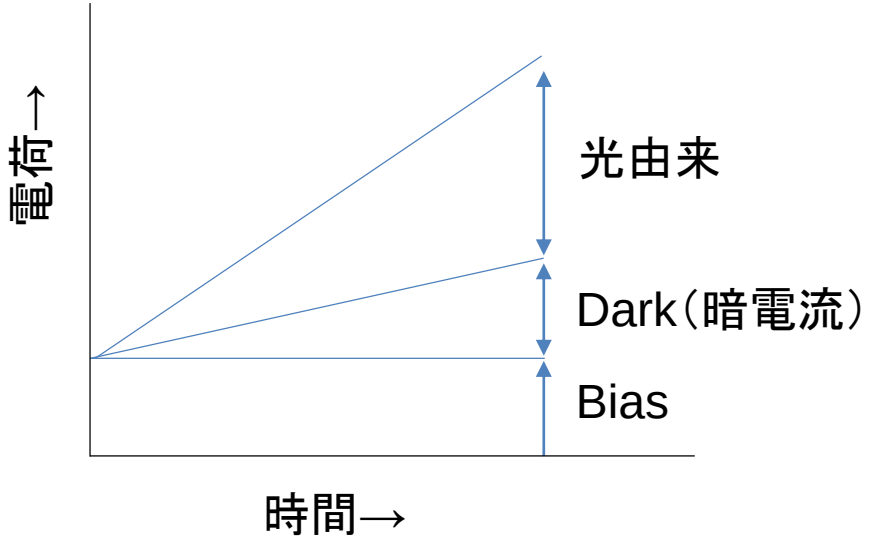
© NAOJ

- HSC (Hyper Suprime-Cam)

- 有効視野角：1.5度角
  - 以前の Suprime-Cam の 3倍
- CCD数：116枚
  - うち、観測用**104**、合焦用8、ガイド用4
- CCD画素数
  - 4272×2272 ・・・ 4Kテレビと同等
- 5つの波長バンドフィルター
  - 可視光から赤外線にかけて g, r, i, z, Y

CCDの1次処理





$$\text{Flat} = \text{一様光照射時の出力値} - (\text{Bias} + \text{Dark})$$

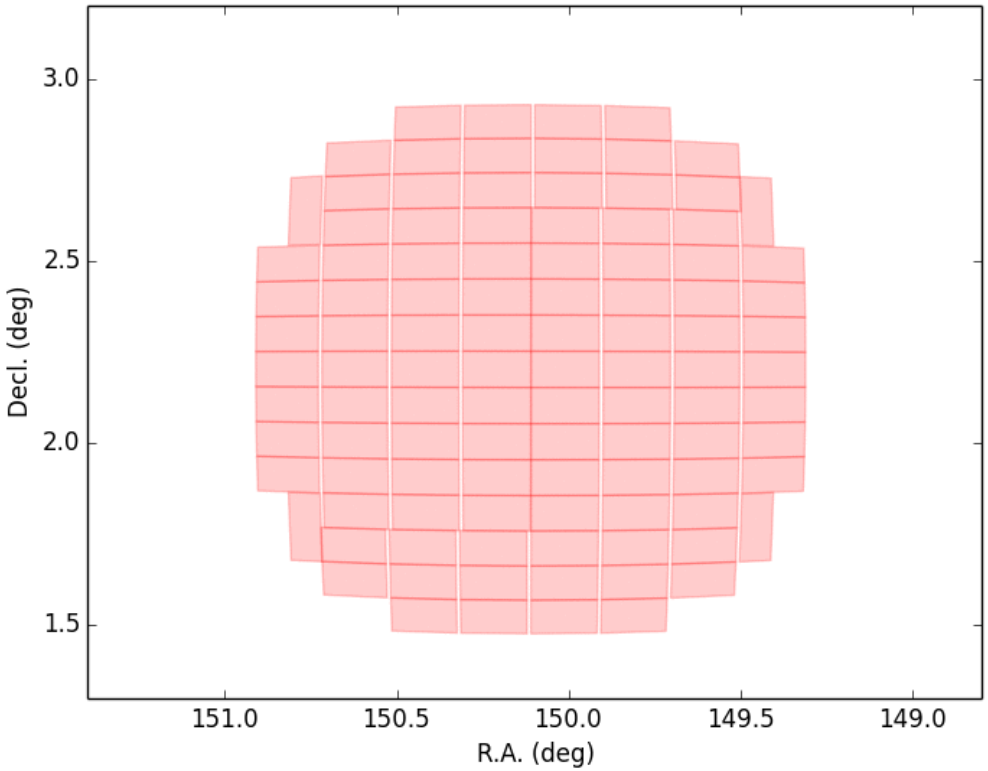
$$\text{CCD1次処理} = \frac{\text{観測時の出力値} - (\text{Bias} + \text{Dark})}{\text{Flat}}$$

（1次処理用データとして他に Fringe と Sky がある）

HSCの撮像観測

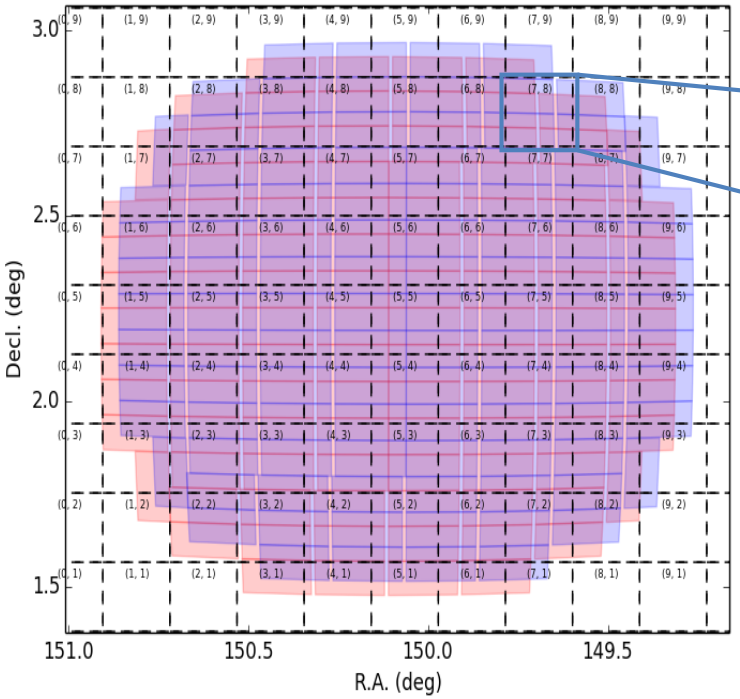
- 同じ領域を複数回の撮像（ショット）
  - CCD間の隙間埋め
  - ノイズ低減
- Frame:
  - 1ショットでCCD1枚が出力するデータ
  - ファイルの最小単位

赤道座標系における各Frameの位置

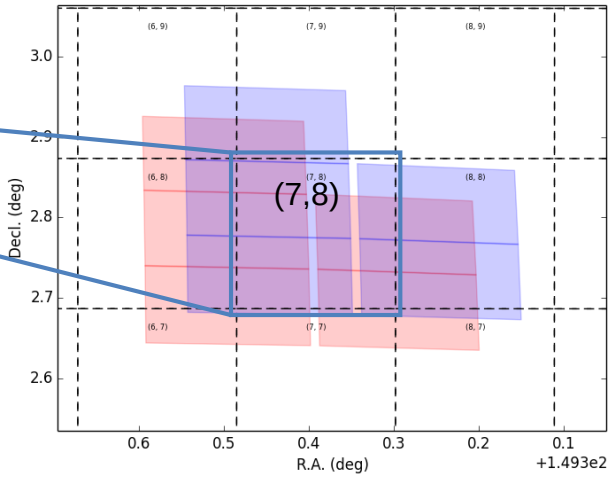


Patch=Coadd処理を行う単位

Patch = 破線で区切られた区画

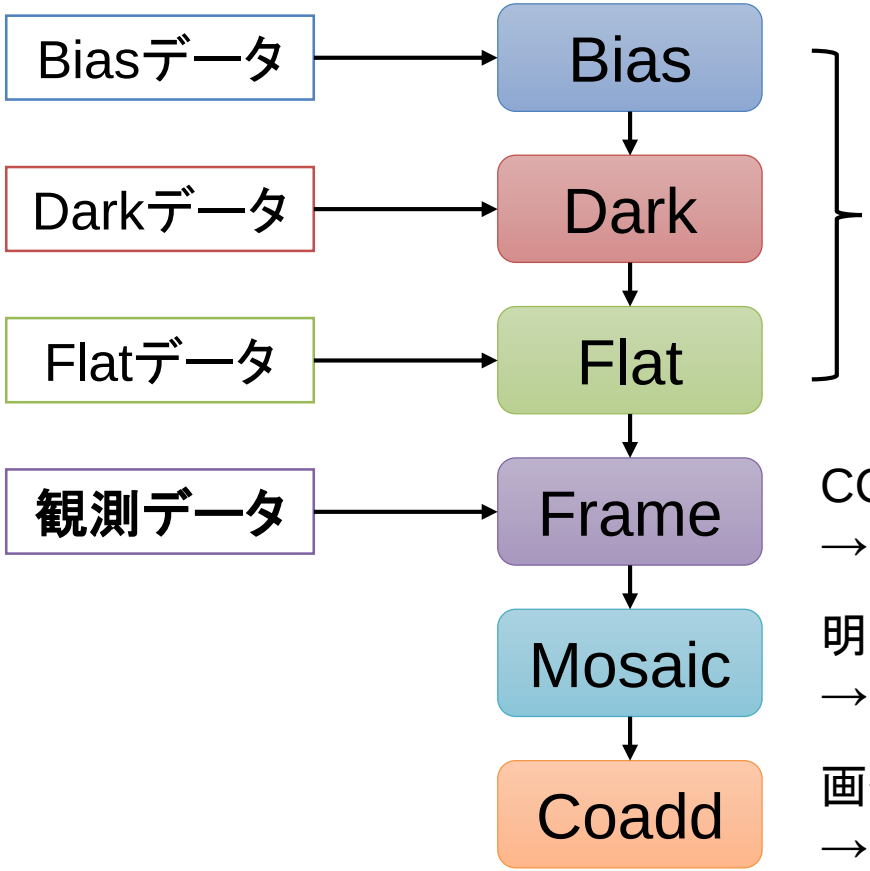


(7,8)のPatchの処理に必要なFrame



- Patchサイズ：4200×4200 pixels
- Patchごとに投影、結合処理を行う

HSCパイプラインの処理内容



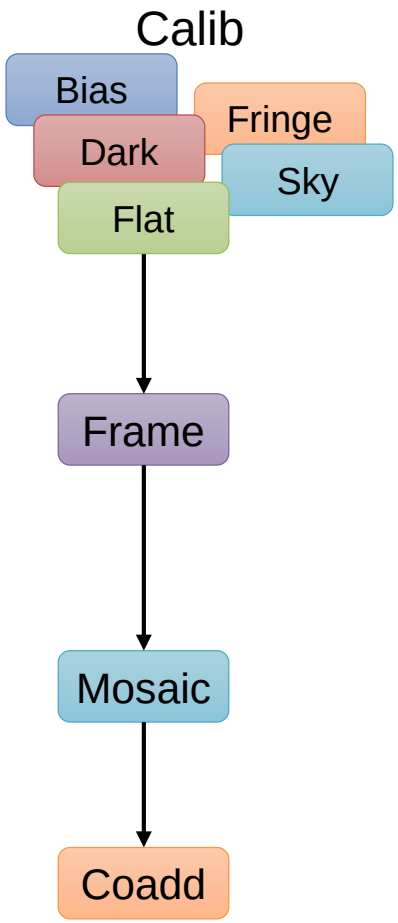
前処理：CCDパラメータ作成
→ CCDごと にタスク並列

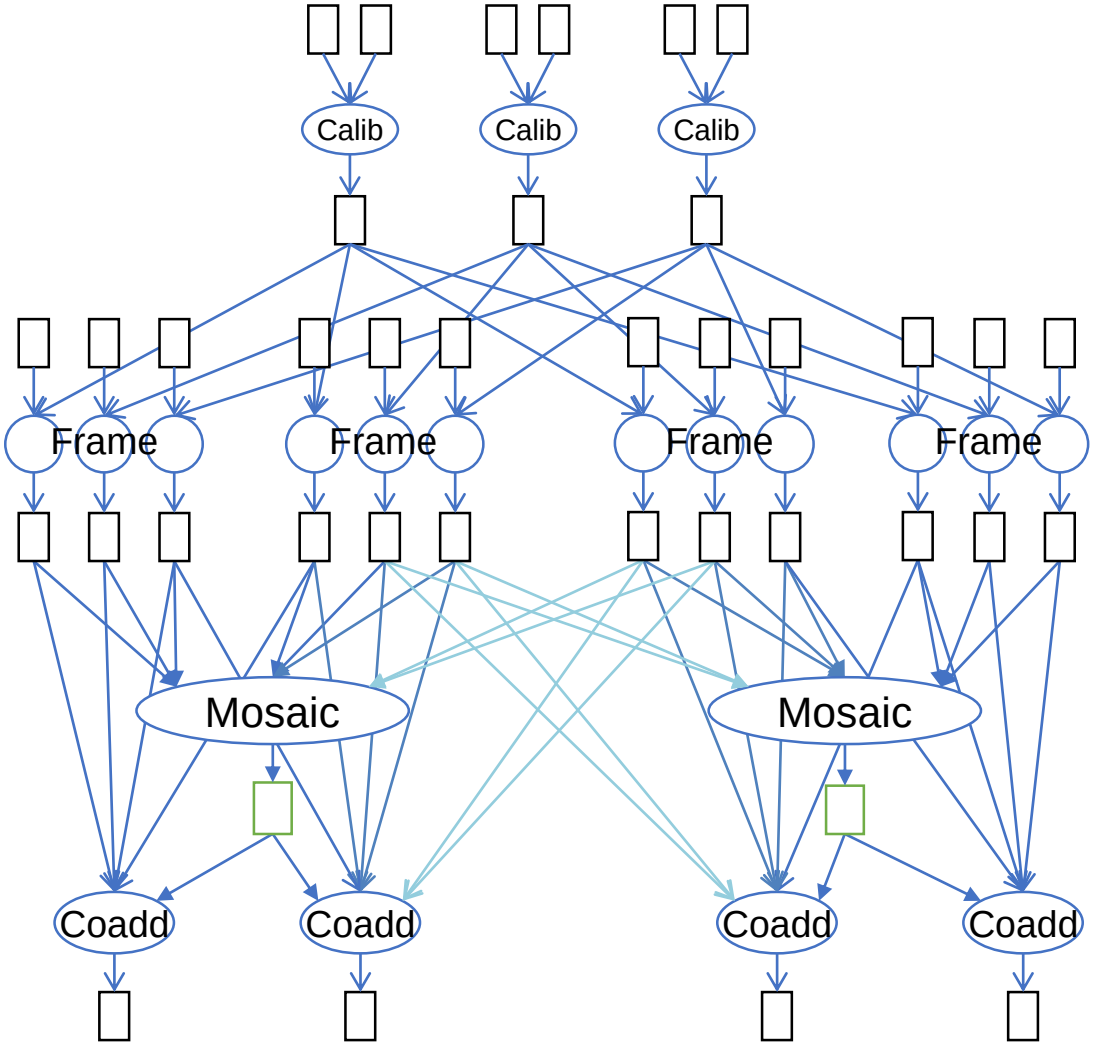
CCDパラメータ補正、天体検出
→ Frameごと にタスク並列

明るさ・位置の較正（行列計算）
→ 1タスク内 でスレッド並列

画像の投影と結合、天体リスト作成
→ Patchごと にタスク並列

HSCパイプラインのワークフロー



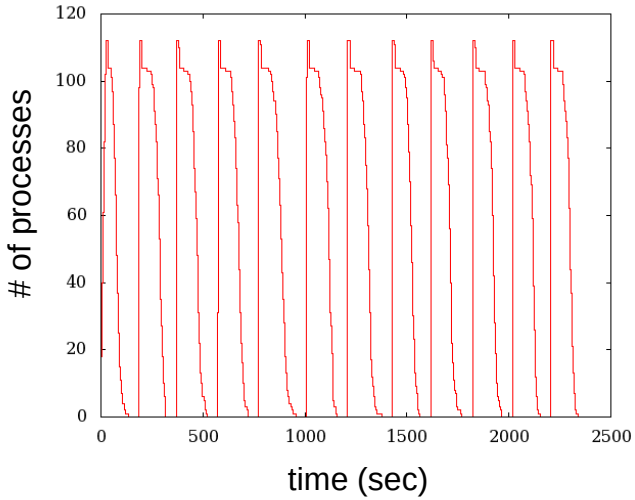


hscPipe: HSCパイプラインソフトウェアパッケージ

- 開発チーム:
  - 国立天文台，Kavli-IPMU，プリンストン大学
- ベースソフトウエア
  - LSSTデータ処理用ソフトウエア
- 実装
  - Python, C++（SWIGでラップ）
- 特徴
  - ファイルを「リポジトリ」に格納
  - リポジトリ以下のパス名は、データIDで決まる
  - コマンドラインにデータIDを指定して実行
  - 独自の並列実行システム

hscPipeの並列処理法

- 独自の並列分散システムを実装
  - Python の multiprocessing.Pool を MPIで実装
  - scatter-gather パターンの実装
  - バッチジョブ投入オプション
- スケールアウトを妨げる欠点:
  - scatterしたタスクの終了待ち
    - コアに空き時間ができる
  - 異種類タスクの並列化未対応



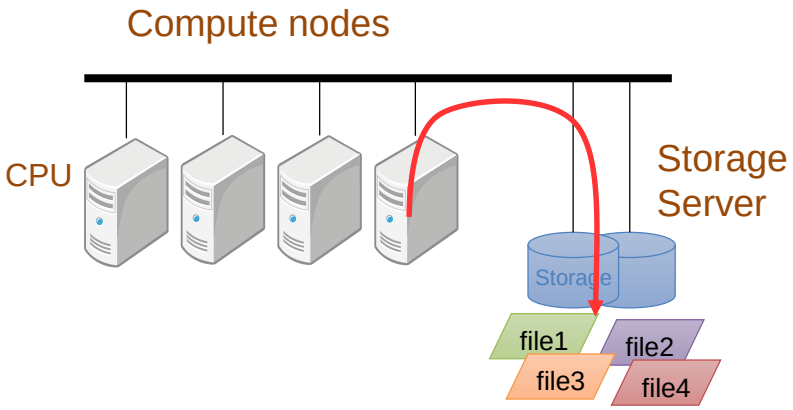
hscPipe 高速化のアプローチ

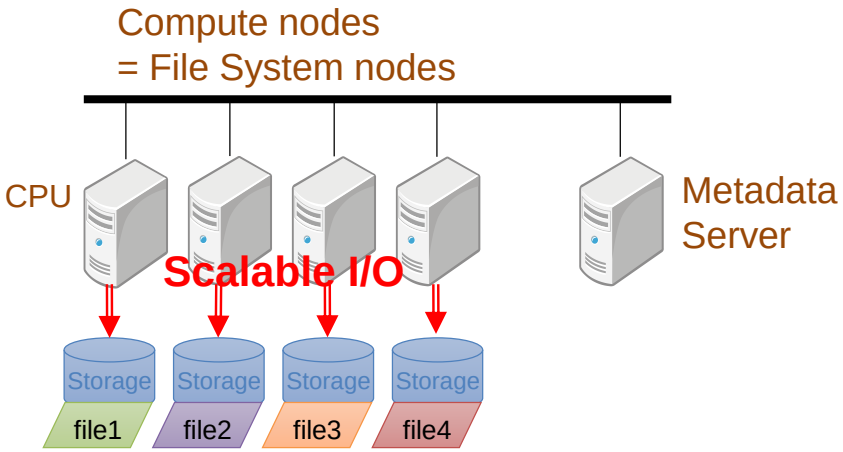
- 行うこと
  - 高並列処理による処理時間の短縮
- 行わないこと
  - 処理内容、アルゴリズムの変更

- 高並列処理のためのシステムソフトウェア
  - 並列ファイルシステム Gfarm
    - I/O性能がスケール
  - ワークフローシステム Pwrake
    - 効率的にコアを使用

Gfarmファイルシステム：スケールするI/O性能

- Storage Server
  - 計算ノード数でスケールしない
- Node-local Storage
  - 計算ノード数でスケール
  - Gfarmファイルシステム





[1] Tatebe, O., Hiraga, K., & Soda, N. (2010). Gfarm Grid File System. *New Generation Computing*, *28*(3), 257–275.

ワークフローシステムPwrake

- Parallel Workflow extension for Rake
  - Rake（Ruby版Make）を拡張
  - UNIX make に似たタスク記述
  - SSHによる複数ノード実行
  - Gfarm対応：プロセス毎に別の gfarm2fs を用意
- タスクスケジューリングに関する研究
  - ファイルのローカリティの向上[2]
  - ディスクキャッシュとコア使用率の向上[3]

[1] Tanaka, M., & Tatebe, O. (2010). Pwrake: A parallel and distributed flexible workflow management tool for wide-area data intensive computing. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10)* (pp. 356–359).

[2] Tanaka, M., & Tatebe, O. (2012). Workflow Scheduling to Minimize Data Movement Using Multi-constraint Graph Partitioning. In *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012)* (pp. 65–72).

[3] Tanaka, M., & Tatebe, O. (2014). Disk Cache-Aware Task Scheduling For Data-Intensive and Many-Task Workflow. In *IEEE Cluster 2014* (pp. 167–175).

ワークフローの実装

Rakeタスク定義方法

入出力ファイルとコマンドを直書きした場合

```
file "../repo/rerun/cosmos/01052/HSC-G/corr/CORR-0011708-009.fits"
  => ["../repo/SSP_UDEEP_COSMOS/2014-11-18/01052/HSC-G/HSC-0011708-009.fits" ,
      "../repo/CALIB/2014-11-18/FLAT/2014-11-15/HSC-G/FLAT-2014-11-15-HSC-G-000.fits"
     ] do
  sh "singleFrameDriver.py ../repo --calib ../repo/CALIB/2014-11-18 --rerun cosmos "+
     "--id visit=11708 ccd=9 filter=HSC-G"
end
```

ルールで記述した場合

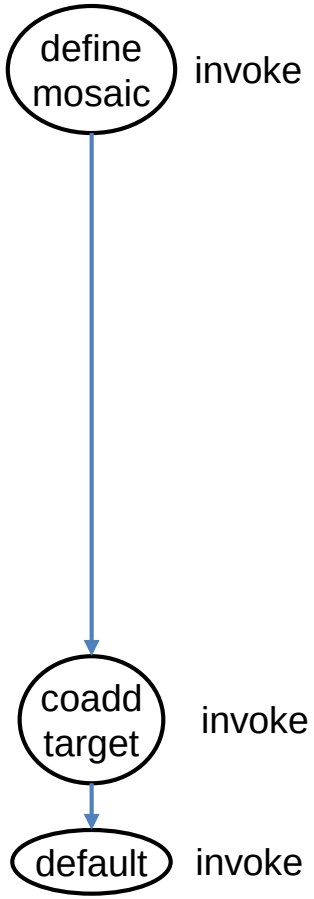
```
rule Frame::RULE => Frame::MAPPER do |t|
  frm = Frame.from_path(t.name)
  sh "singleFrameDriver.py #{frm.rerun.opt} "+
     "--id visit=#{frm.visit} ccd=#{frm.ccd} filter=#{frm.filter}"
end
```

動的タスク定義

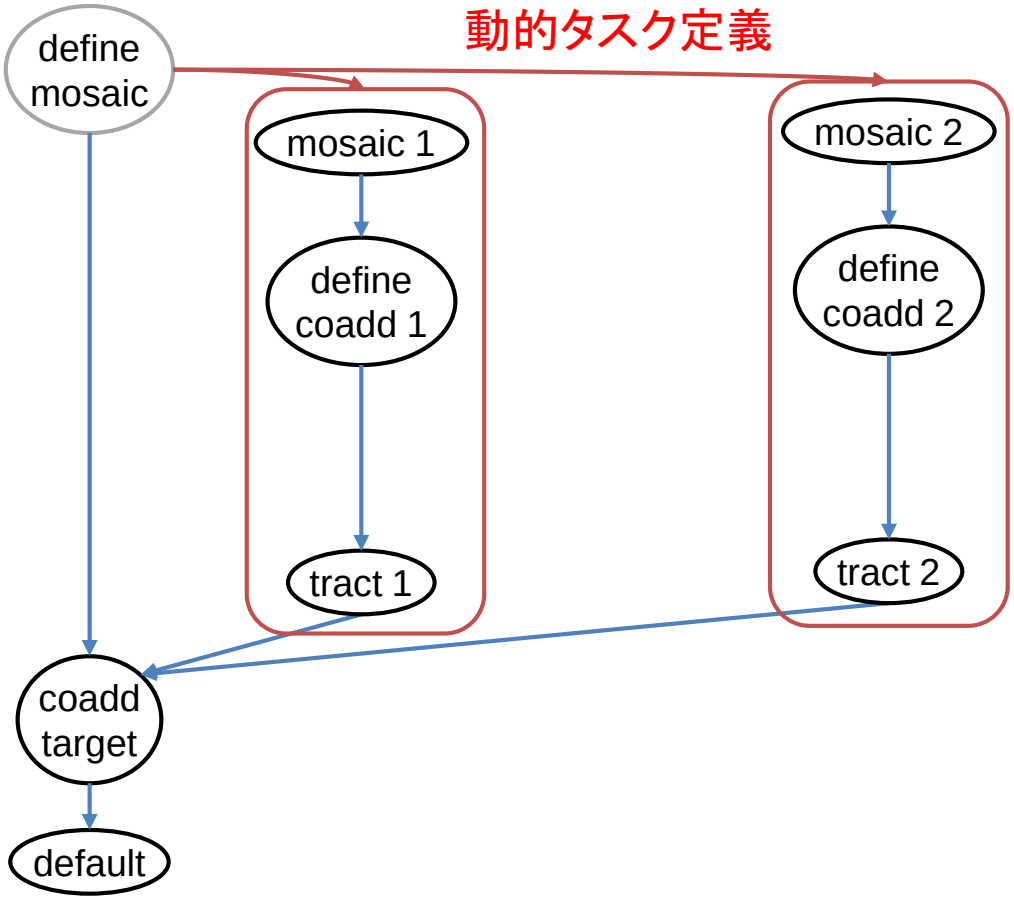
- Mosaic&Coaddタスクを定義するためには、画像の位置情報が必要
- 位置情報取得タスク実行後にタスク定義 → 動的タスク定義
- Rakeでの記述法: タスク定義をネスト

```
task :define_mosaic => TRACT_CSV do
  Mosaic.read_csv(TRACT_CSV).each do |tract|
    coadd_csv = Coadd.csv_files(tract)
    task "define-#{tract.id}" => [tract.target] + coadd_csv do
      reinvoke "tract-#{tract.id}" => Coadd.read_csv(coadd_csv)
    end
    task "tract-#{tract.id}" => "define-#{tract.id}"
    task :coadd => "tract-#{tract.id}"
  end
  reinvoke :coadd
end

task :coadd => :define_mosaic

task :default => :coadd
```

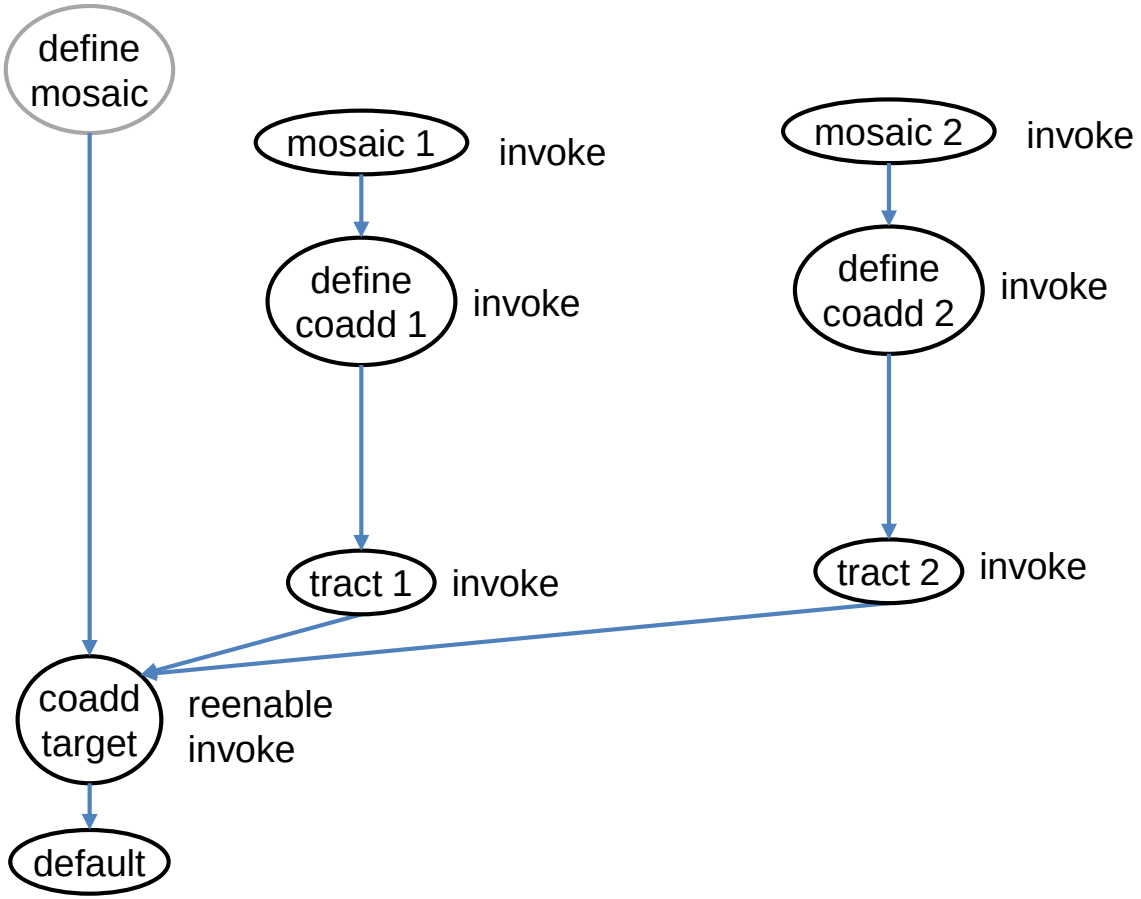
動的タスク定義



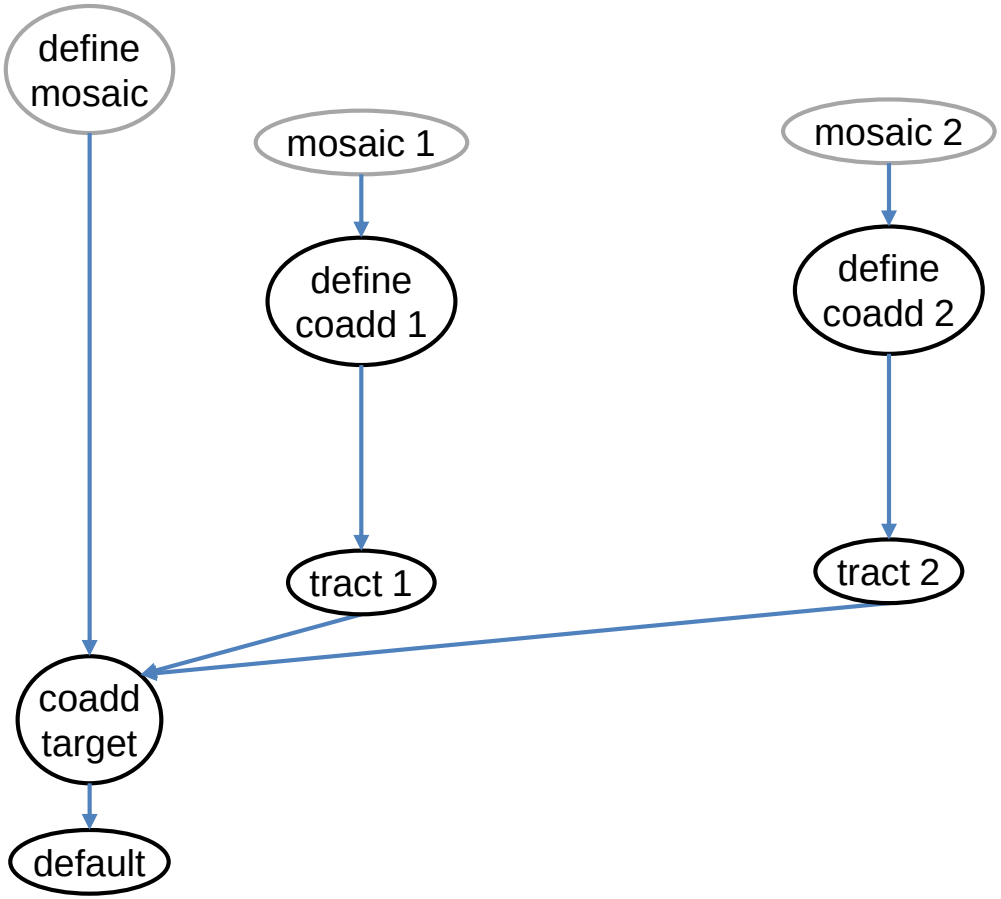
動的タスク定義


動的タスク定義
define mosaic
mosaic 1
define coadd 1
tract 1
mosaic 2
define coadd 2
tract 2
coadd target
default

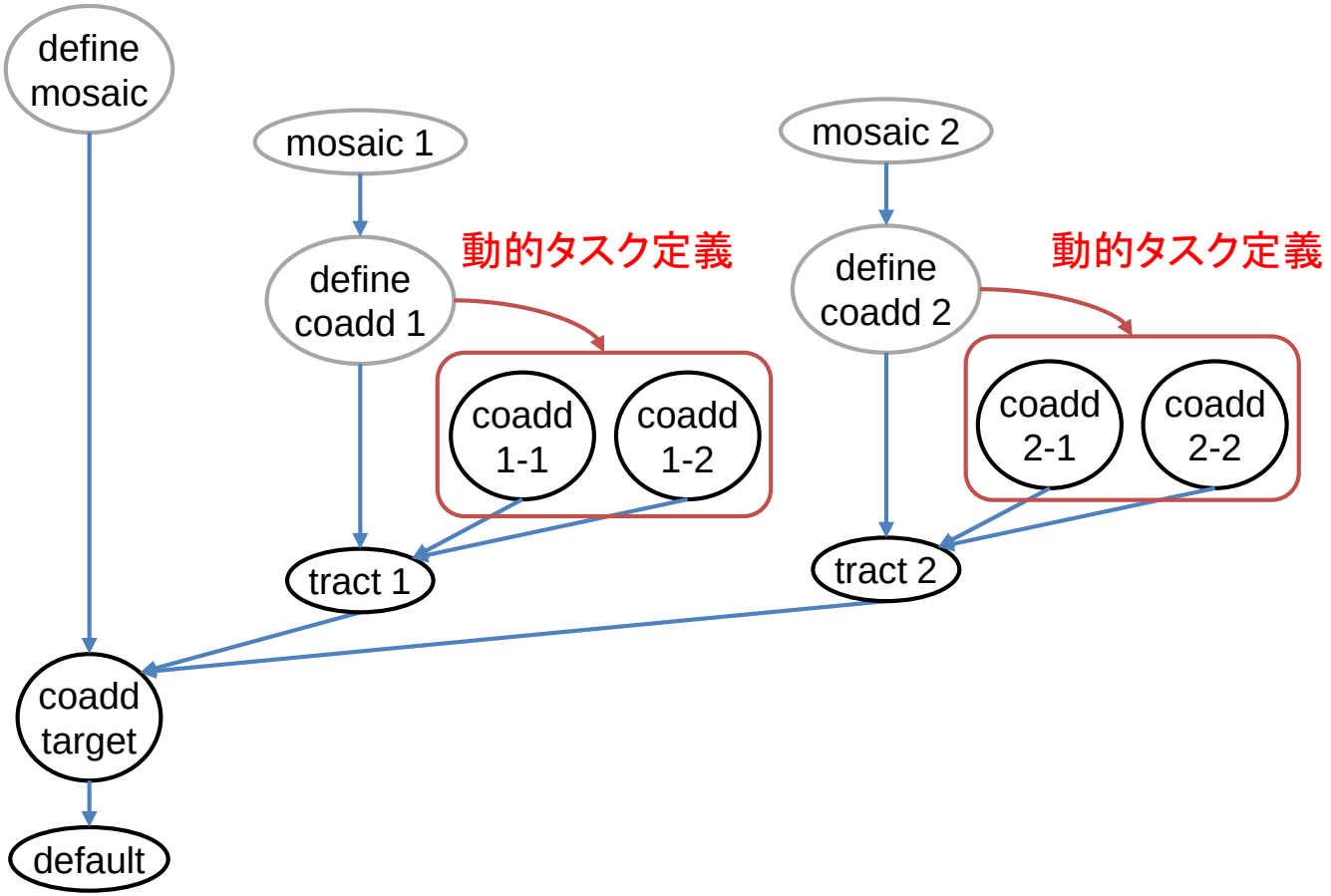
動的タスク定義


define
mosaic
mosaic 1
invoke
define
coadd 1
invoke
tract 1
invoke
mosaic 2
invoke
define
coadd 2
invoke
tract 2
invoke
coadd
target
reenable
invoke
default

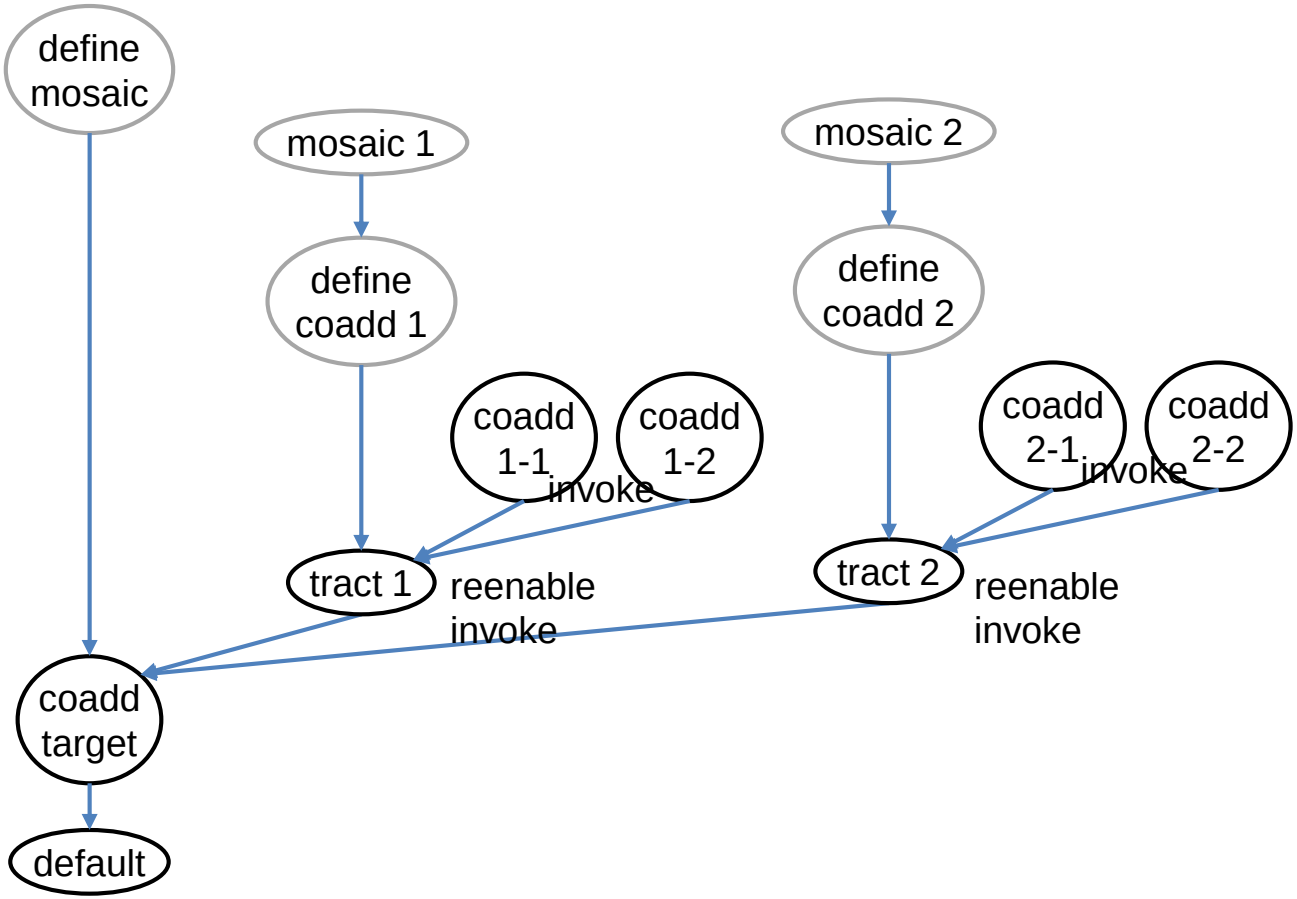
動的タスク定義


define
mosaic
mosaic 1
mosaic 2
define
coadd 1
define
coadd 2
tract 1
tract 2
coadd
target
default

動的タスク定義


define
mosaic
mosaic 1
mosaic 2
動的タスク定義
動的タスク定義
define
coadd 1
define
coadd 2
coadd
1-1
coadd
1-2
coadd
2-1
coadd
2-2
tract 1
tract 2
coadd
target
default

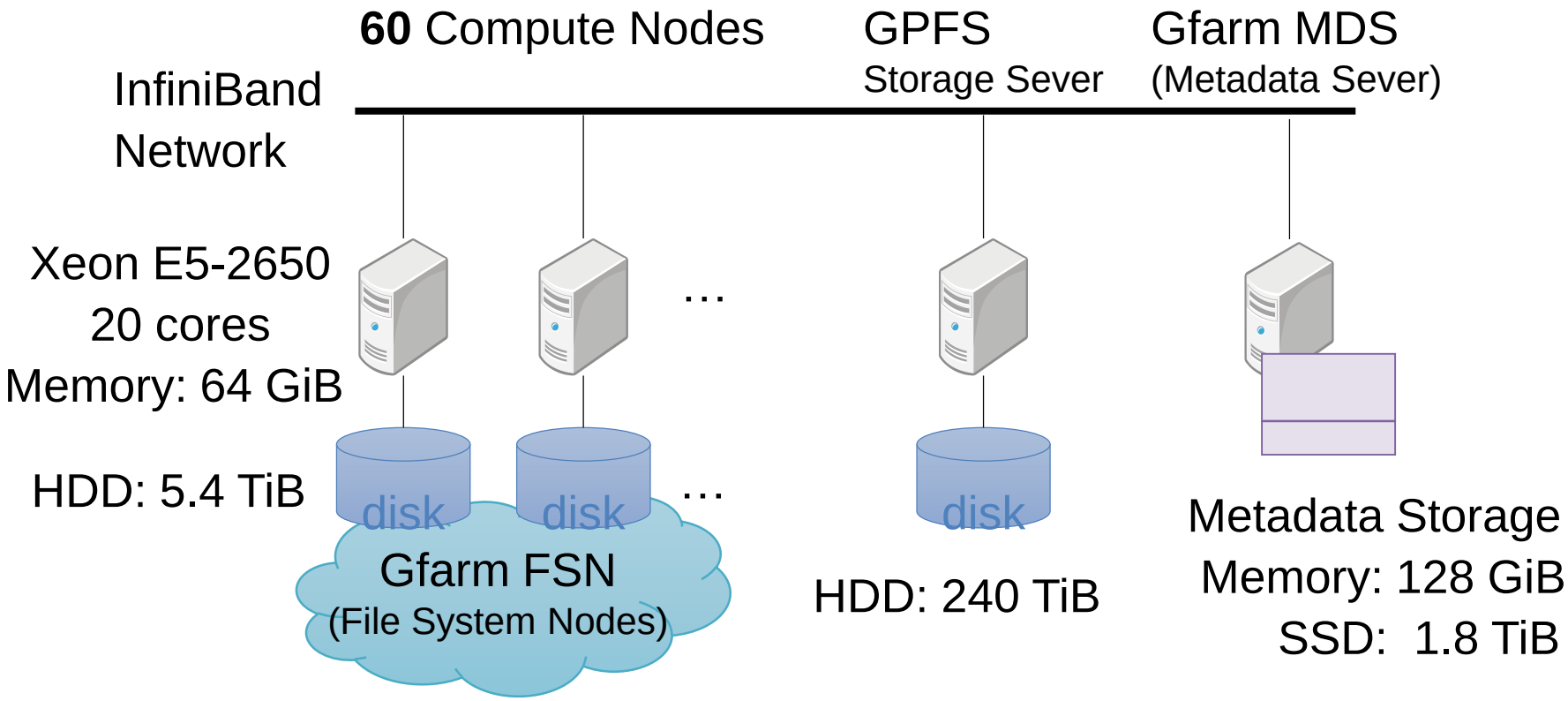
動的タスク定義


define
mosaic
mosaic 1
define
coadd 1
coadd
1-1
coadd
1-2
invoke
tract 1
reenable
invoke
mosaic 2
define
coadd 2
coadd
2-1
coadd
2-2
invoke
tract 2
reenable
invoke
coadd
target
default

その他の面倒な対処

- パス指定方法
  - hscPipe: 絶対パスを想定
  - Pwrake:　相対パスを想定（Gfarmマウントポイントが変わるため）
    - hscPipe内で、絶対パスに強制変換するコード（1か所）を削除
    - シンボリックリンクの絶対パスを相対パスに変換、など
- Calibタスクの並列化方法
  - scatter-gatherを別プロセスに分けられれないため、1日単位の粒度となる。
  - 大規模実行の場合でも Biasタスクは66タスク << 目標1000コア使用
  - 2段階の並列化
    - Pwrakeによる、ノード間並列
    - hscPipeのpoolによる、ノード内並列
      - ファイルパスをGfarmのマウントポイントに振り分けるパッチをhscPipeに適用。

評価環境: IPMUクラスタ



性能評価

- 小規模データ (IEEE Cluster 2018[1]にて発表)
  - 2017/1/23, WIDE field, i-band filter
  - 観測日数: 1晩の1/4
  - 入力: 80 GB, 出力: 1,183 GB
  - hscPipe ver.4
  - 目的: スケーラビリティ、効率的なタスク起動
- 大規模データ
  - 2014-2017, COSMOS field, 5 filters
  - 観測日数: 58日
  - 入力: 5.3 TB、出力: 46 TB
  - hscPipe ver.6
  - 目的: 60ノード30時間（予定）

[1] Tanaka, M., Tatebe, O., & Kawashima, H. (2018). Applying Pwrake Workflow System and Gfarm File System to Telescope Data Processing. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)* (pp. 124–133). Belfast, UK.

強スケーリング実験

Detrend Task →

(Bias, Dark, Flat)

02 tasks × 3

(Except broken CCDs)

Frame Task →

4096 tasks

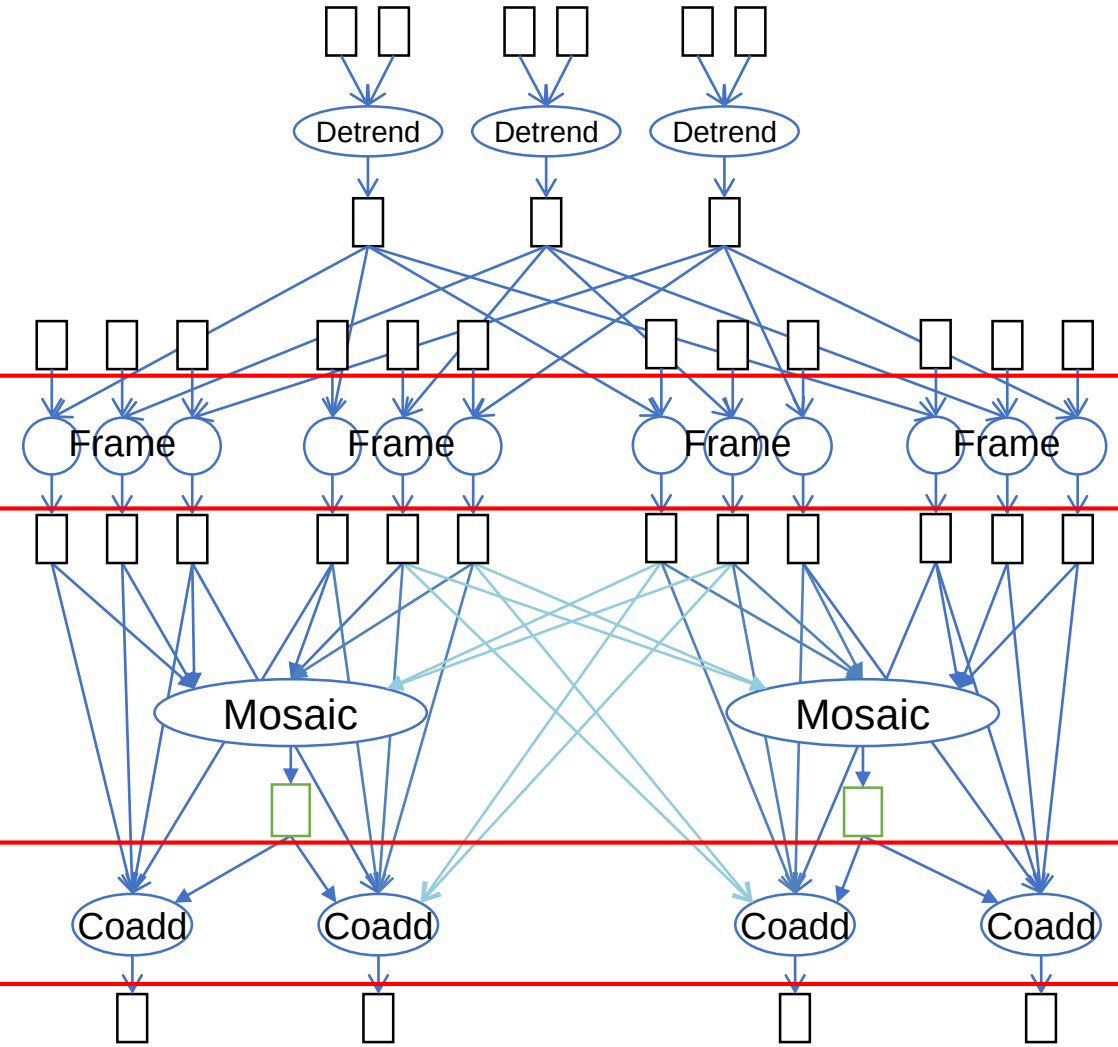
Mosaic Task →

29 tasks

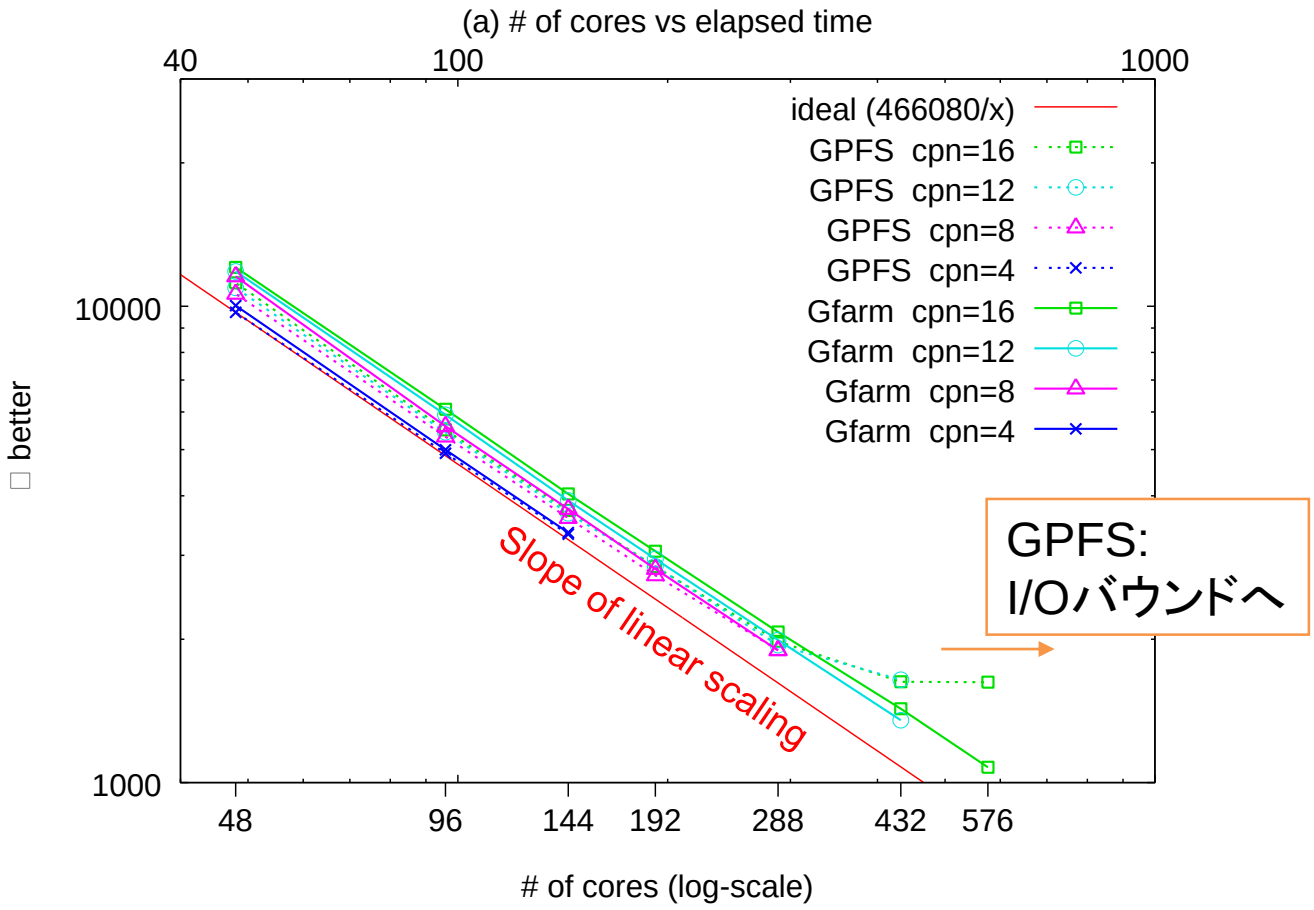
(20 threads for each task)

Coadd Task →

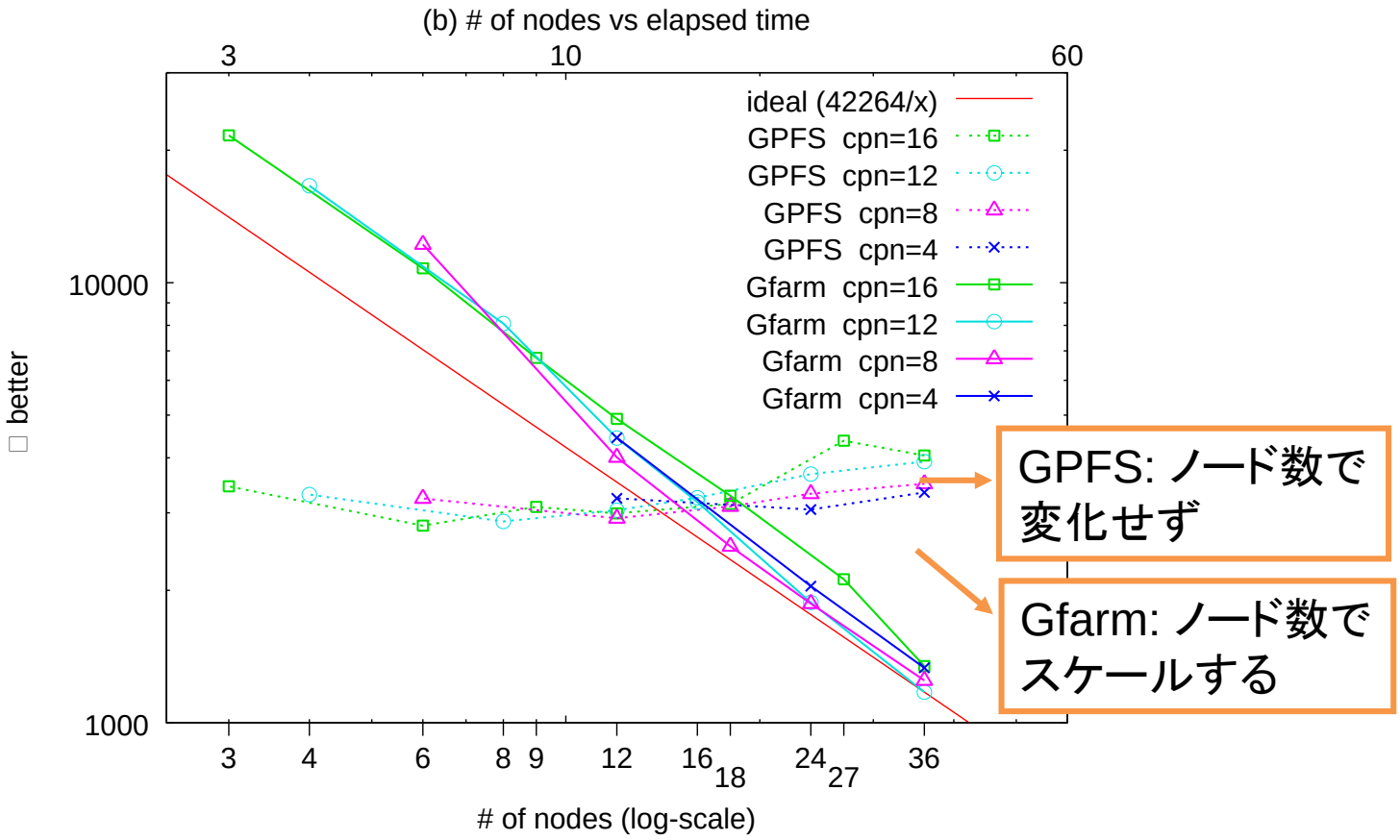
1151 tasks



Frameタスクの強スケーリング測定

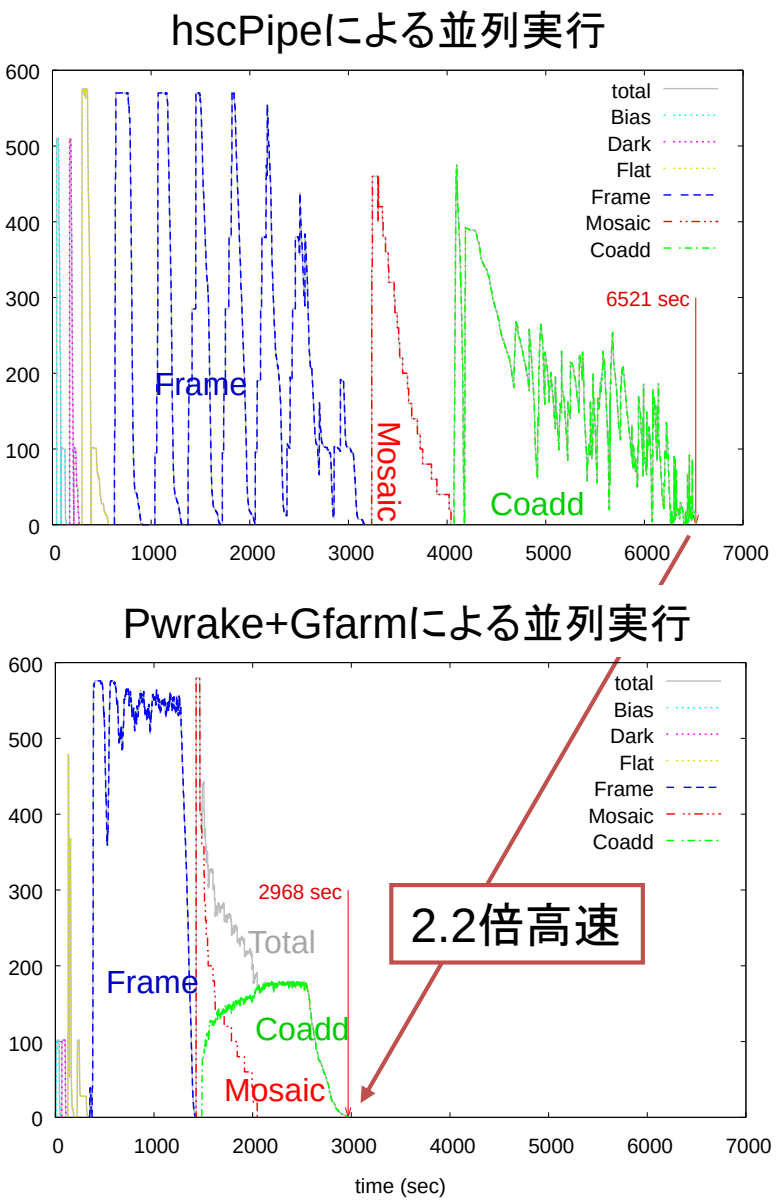


Coaddタスクの強スケーリング測定



Pwrake+Gfarmによる高速化

- 1晩の1/4の観測データ
- 36ノードx16コア=576コア
- 比較対象：
  - hscPipeに実装されている並列処理
- Pwrake+Gfarmにより、I/O性能とコア使用率が向上
- 2.2倍の高速化を達成



大規模データ処理実験

- 目標:
  - IPMUクラスタ（60ノード）全ノード使用
- 対象データ:
  - 2014-2017年観測の SSP COSMOS field
  - 観測日数: 58日
  - ショット数: 1,616
  - フィルターバンド: g, r, i, z, y
  - 入力: 5.3 TB、出力: 46 TB

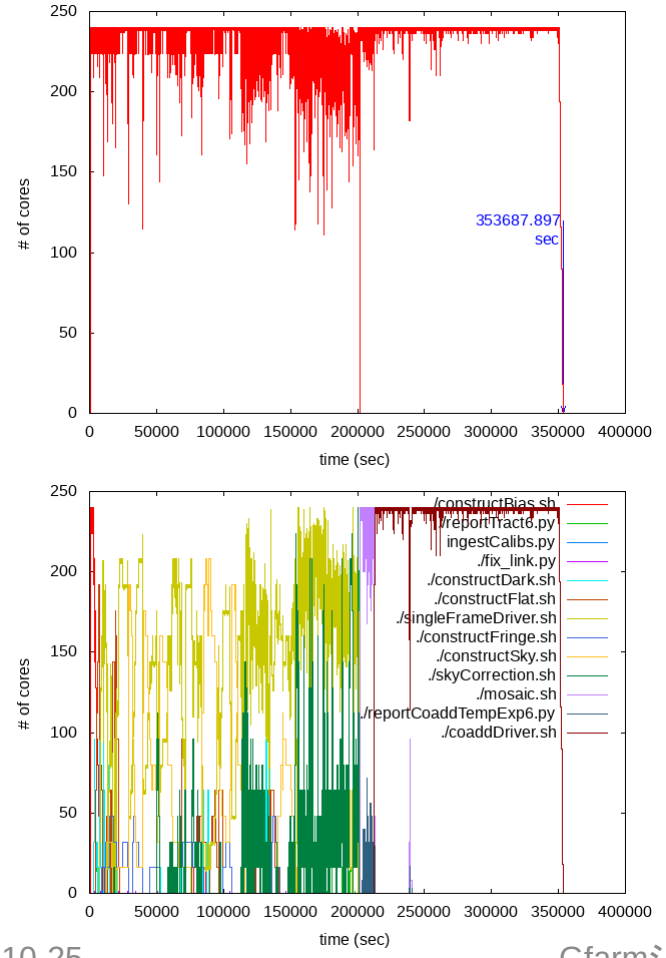
大規模実験結果

ノード数	15	30	60
コア数	240	480	960
経過時間(hours)	98.25	52.24	~28?
性能向上	-	1.88	?
各タスク積算時間	12,652	12,722	?
コア使用率	97.5%	95.4 %	?

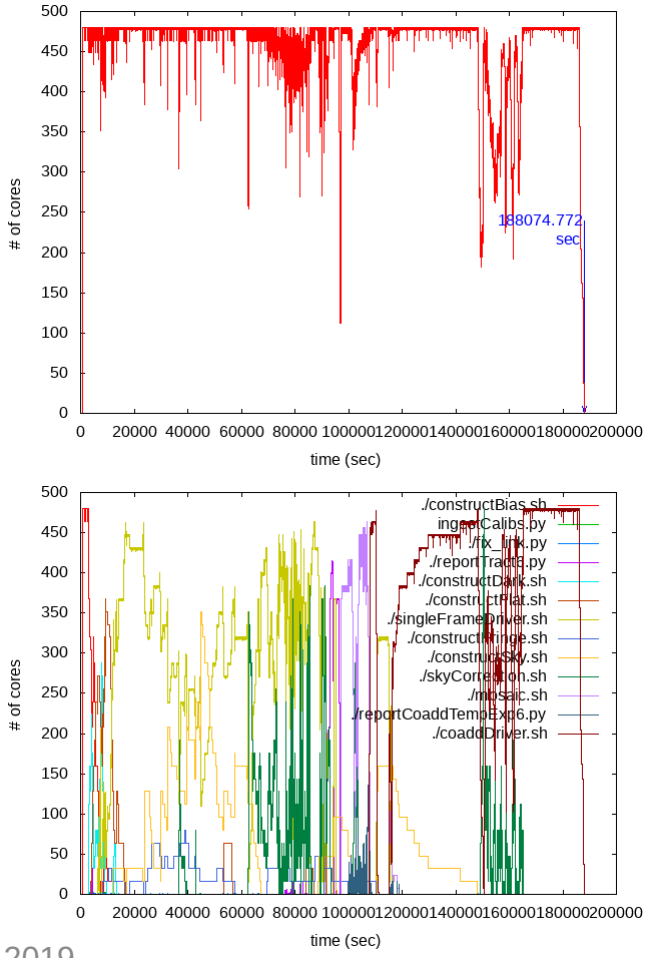
クラスタ使用の予約が取れず未実行

コア使用グラフ

15ノード、240コア



30ノード、480コア



実験におけるGfarmの設定

MDS設定

```
/etc/security/limits.conf:
    * - nofile 262144

/etc/sysctl.conf:
    net.core.somaxconn = 2048
    net.ipv4.tcp_max_syn_backlog = 262144

gfmd.conf:
    metadb_server_listen_backlog 2048
    metadb_server_max_descriptors 262144
    metadb_server_long_term_lock_type mutex
    metadb_server_nfs_root_squash_support enable
    schedule_idle_load_thresh 200
    atime disable
    replica_check disable
```

クライアント設定

```
gfarm2.conf:
    schedule_idle_load_thresh 200
    gfsd_connection_cache 256
    gfmd_authentication_timeout 120
    schedule_rpc_timeout 120
    network_receive_timeout 120
```

大規模実行の負荷

- MDSの負荷は余裕ありそう
  - hscPipe4 → hscPipe6 でファイルアクセスが改善された？
- FSNの負荷
  - libgfarm が abort する問題が発生
    - エラーコードがマイナスとなり、範囲外メモリアクセス
    - I/Oのがビジーになると発生
    - Gfarm ver 2.7.14 で修正済
  - その後も I/Oビジーが原因と思われる現象がsyslogに記録
  - Coaddタスクを 8プロセス/node に減らして実行

Pwrakeの改良

- 複数コアを使用するタスクオプション
- 複数コアが空くまでノードを予約
- 高ランク（最後のタスクから遠い）タスクを優先するアルゴリズムの改善
- invokeの別スレッド化

まとめ

- 目的:
  - HSCパイプラインの高速化
- 手法:
  - Gfarm＋Pwrake による高並列実行
- 評価:
  - Gfarmによる、I/O性能のスケーラビリティ
  - Pwrakeによる、効率的なコア使用
- 大規模実行実験:
  - 58日分の観測データ（入力:5.3 TB、出力:46 TB）の処理を、30ノード480コア使用し、52時間で達成