

並列分散ワークフローシステム Pwrakeによる大規模データ処理

田中昌宏、建部修見

(筑波大学 計算科学研究センター)

発表内容

- eサイエンス基盤
 - 広域分散ファイルシステム Gfarm
 - 並列分散ワークフロー実行システム Pwrake
- 天文学データ処理
 - ワークフローの記述
 - 性能評価

天文観測データ量

– サーベイ観測

- SDSS (画像) ~ 16 TB
- 2MASS (画像) ~ 8 TB

– すばる望遠鏡

- SprimeCam ~ 1.5 TB/year
- Hyper-SprimeCam ~ 15 TB/year ?

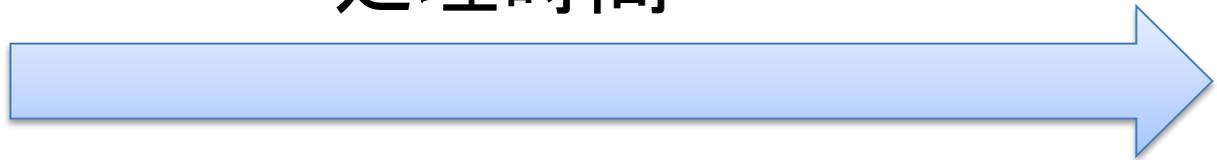
– データ解析には、元データの>10倍の容量必要

– 数100 TB ~ 数 PB のデータ量を扱う時代

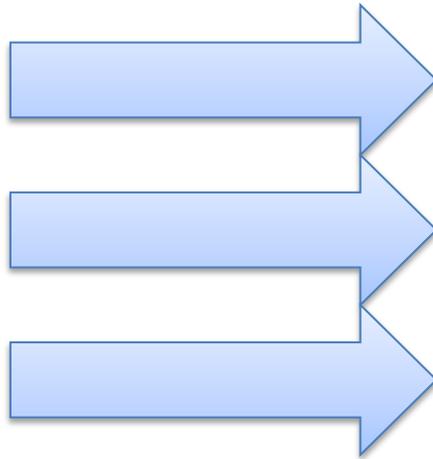
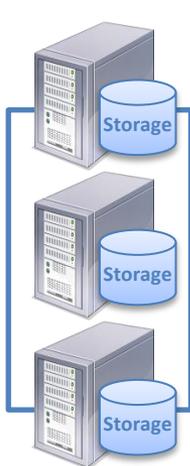
並列処理

処理時間

逐次処理

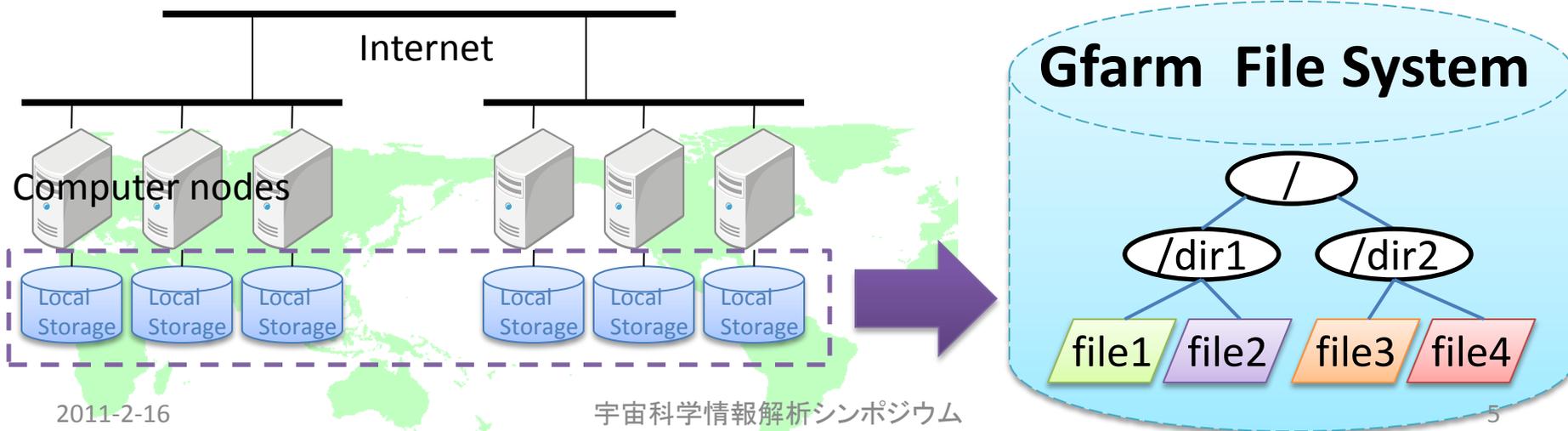


並列処理



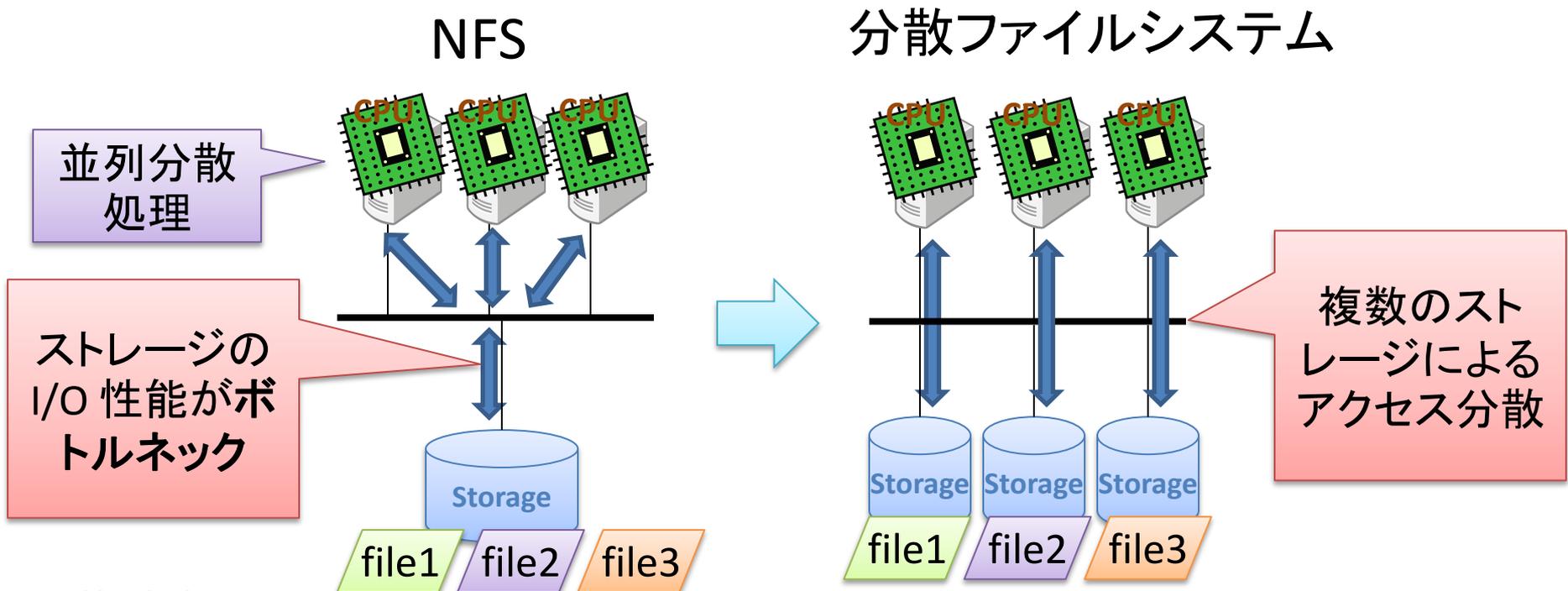
Gfarm 広域分散ファイルシステム

- 各ノードのストレージを統合
- 統一したディレクトリ空間
- 広域でファイルを共有
- <http://datafarm.apgrid.org/>



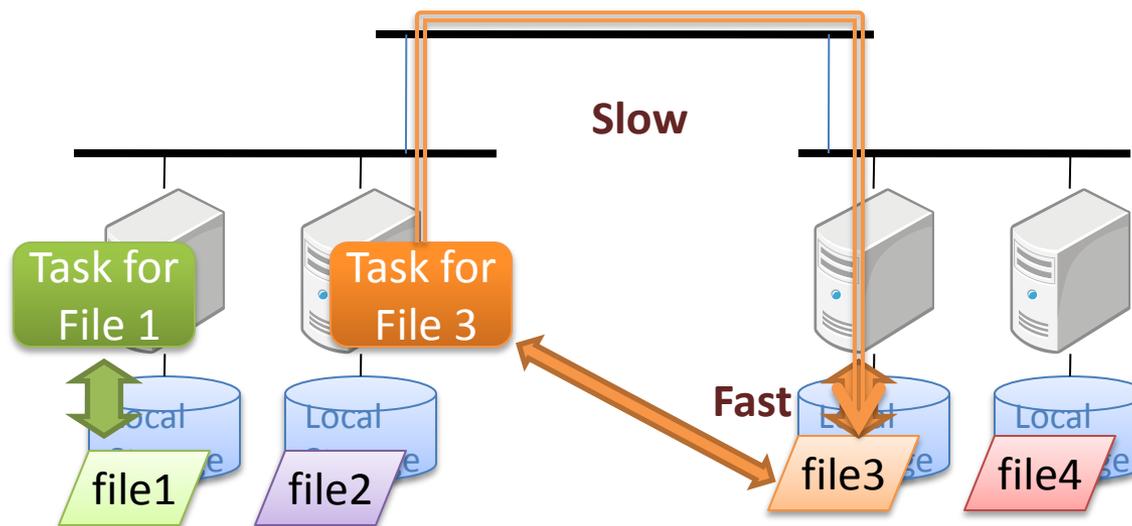
並列I/O性能の向上

- 集中型ファイルシステムでは、ストレージI/Oがボトルネック
- 分散ファイルシステムにより、スケーラブルな並列I/O性能が実現



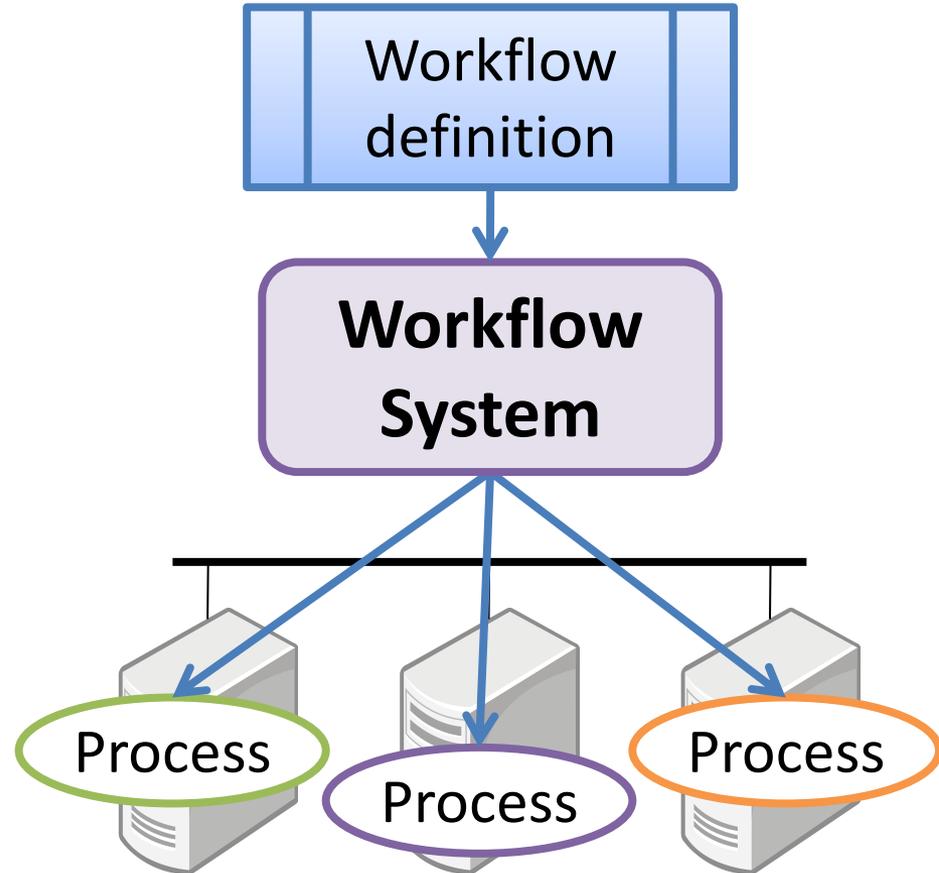
ローカルリティを考慮したタスク実行

- Gfarmでは、計算ノードがとストレージノードを兼ねる
- ファイルが存在するノードで処理すれば、性能が向上
- タスク実行は、ワークフローシステムが行う



ワークフロー処理システム

- ワークフロー
 - タスク
 - 出力ファイル
 - 依存関係
- ワークフローシステム
 - ワークフロー定義に従って、タスクを各ノードで実行



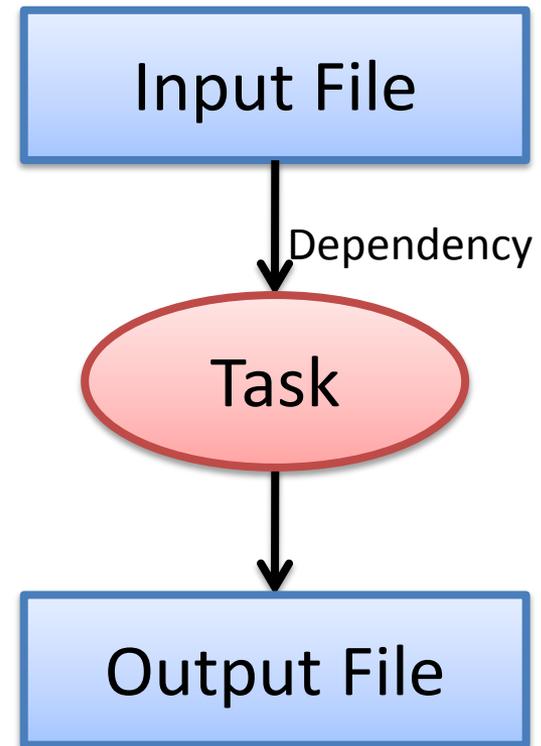
ワークフローの記述言語

- 課題

- システムに詳しくない人でも記述できるか？
- 複雑なワークフローを記述できるか？
- 再利用可能か？

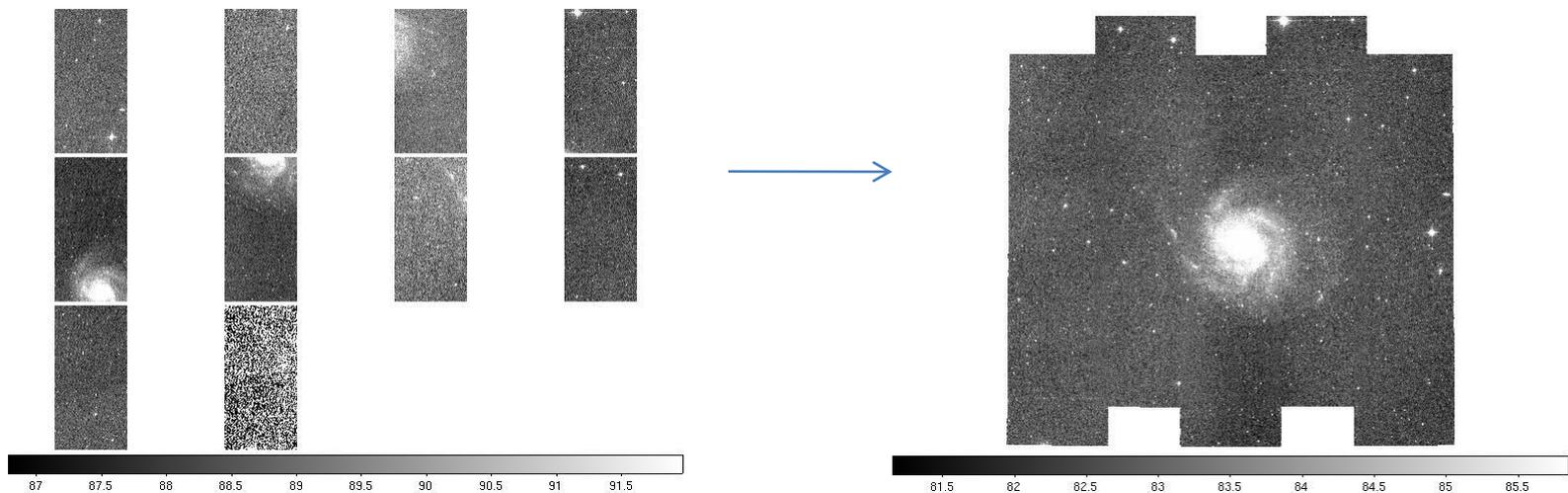
ワークフローのグラフ表現

- Task: 頂点(楕円)
- File: 頂点(四角)
- 依存関係: 辺
- **DAG**
 - Directed Acyclic Graph
 - 有向非循環グラフ

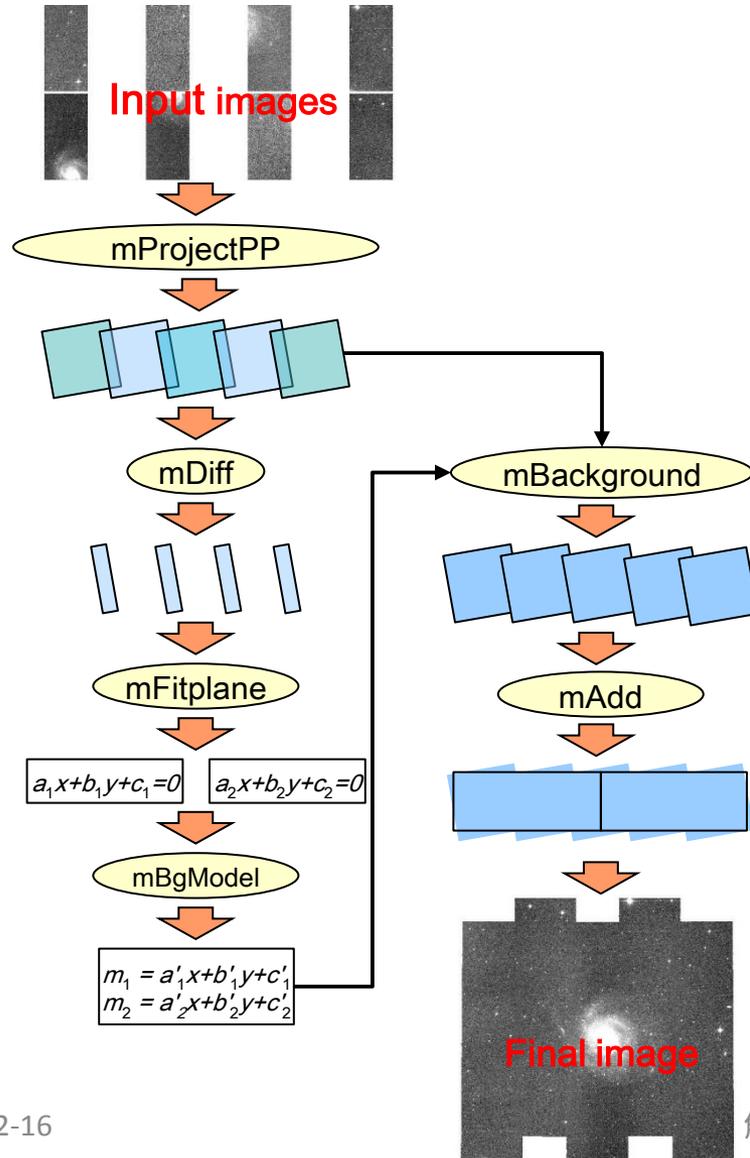


天文データ処理の例

- Montage
 - software for producing a custom mosaic image from multiple shots of images.
 - <http://montage.ipac.caltech.edu/>



Montage Workflow

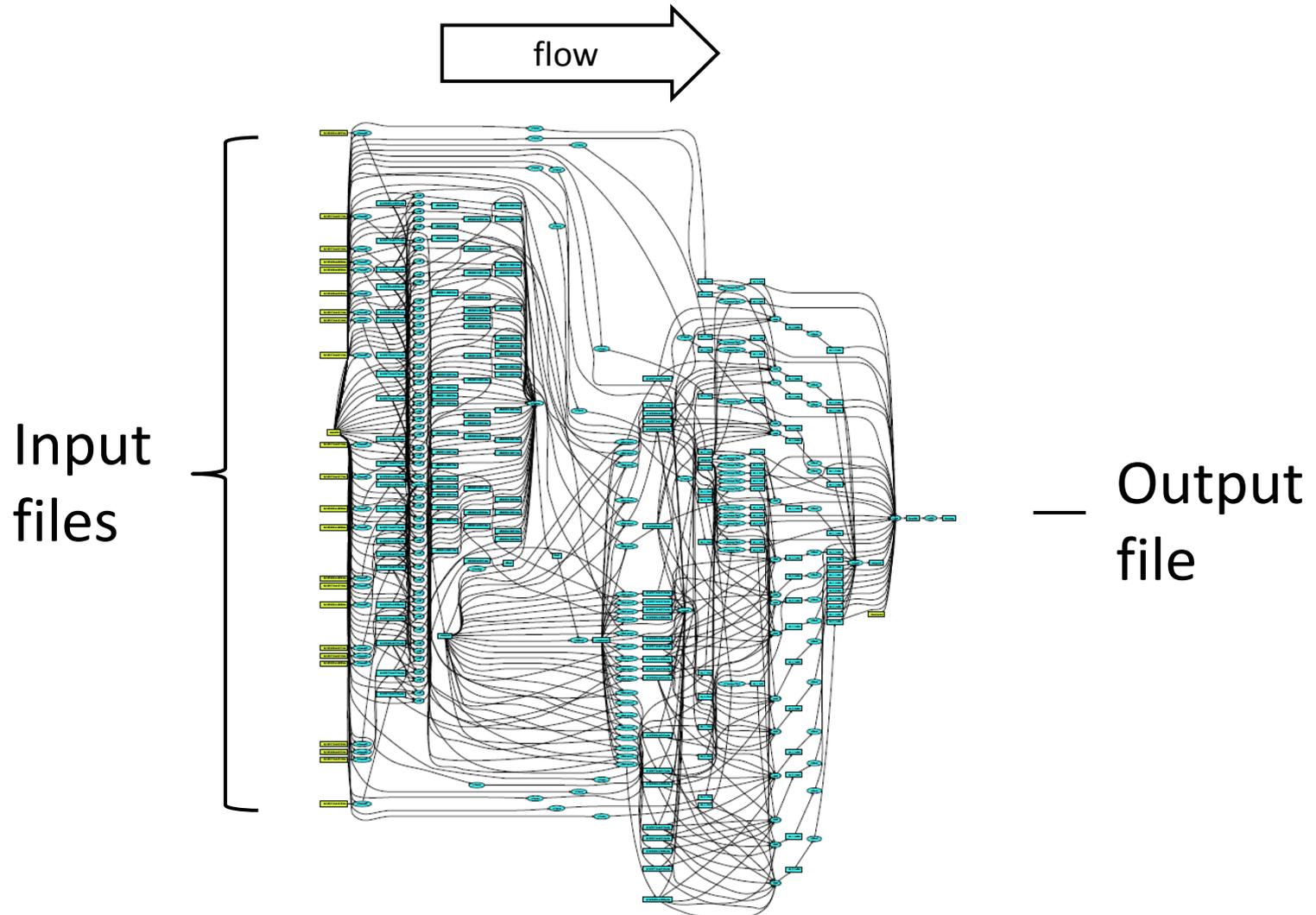


- 処理内容:

- 座標変換
- 明るさ補正
- 足し合わせ

- 1 image : 1 process

Montageワークフローのグラフ



DAGによるワークフロー定義

- 多くのワークフローシステムで採用
- 欠点
 - 手書きは不可能:
 - DAG生成プログラムが必須
 - 再利用が不可能
 - 大規模なワークフローでは、DAGが膨大になる
 - 動的ワークフローが不可能

XMLで記述したDAG:

```
<adag xmlns="http://www.griphyn.org/chimera/DAX"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.griphyn.org/chimera/DAX
      http://www.griphyn.org/chimera/dax-1.8.xsd"
      count="1" index="0" name="test">
  <filename file="2mass-atlas-981204n-j0160056.fits"
    link="input"/>
  ...
  <job id="ID000001" name="mProject" version="3.0" level="11"
    dv-name="mProject1" dv-version="1.0">
    <argument>
      <filename file="2mass-atlas-981204n-j0160056.fits"/>
      <filename file="p2mass-atlas-981204n-j0160056.fits"/>
      <filename file="templateTMP_AAAaaa01.hdr"/>
    </argument>
    <uses file="2mass-atlas-981204n-j0160056.fits" link="input"
      dontRegister="false" dontTransfer="false"/>
    <uses file="p2mass-atlas-981204n-j0160056.fits" link="output"
      dontRegister="true" dontTransfer="true"
      temporaryHint="tmp"/>
    <uses file="p2mass-atlas-981204n-j0160056_area.fits"
      link="output" dontRegister="true" dontTransfer="true"
      temporaryHint="tmp"/>
    <uses file="templateTMP_AAAaaa01.hdr" link="input"
      dontRegister="false" dontTransfer="false"/>
  </job>
  ...
  <child ref="ID003006">
    <parent ref="ID000001"/>
    <parent ref="ID000006"/>
  </child>
  ...
</adag>
```

Makefileによるワークフロー記述

- Makeflow <http://www.nd.edu/~ccl/software/makeflow/>
- GXP make <http://www.logos.ic.i.u-tokyo.ac.jp/gxp/>
- 利点
 - タスク定義言語である
 - 広く使われている
 - ルール定義が可能
 - ファイルのタイムスタンプに基づき、途中から実行を再開
- 問題点
 - 複雑な定義を記述できない
 - 動的なワークフローを記述できない

Rake

- Ruby 版 make。ビルドツール
- タスク記述ファイル : Rakefile
- タスク記述文法 : Ruby
 - 内部DSL (Internal Domain Specific Language)と呼ばれる。
- Rakefile はRubyスクリプトとして実行されるので、Rubyでできることはすべて可能。

Rakefile におけるタスク記述

タスク定義
(Rubyメソッド)

依存関係

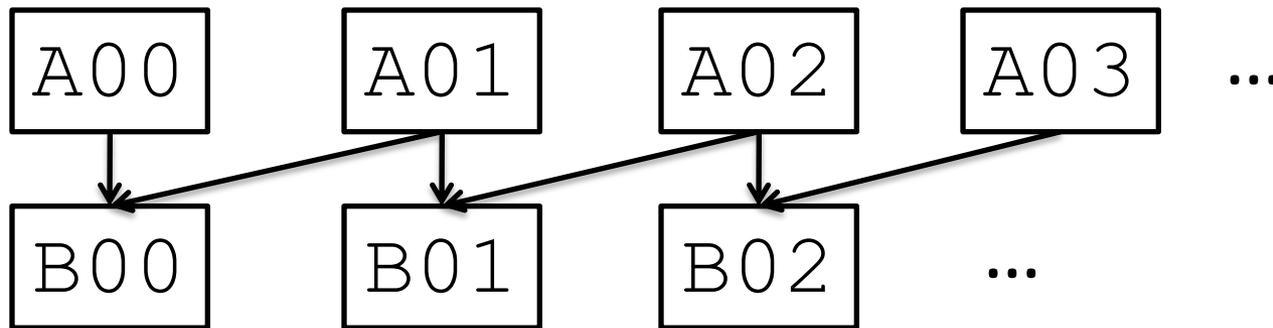
自身のタスク名 => 依存タスク名

```
file "prog" => ["a.o", "b.o"] do
  sh "cc -o prog a.o b.o"
end
```

Ruby コードブロック
タスクのアクションをRuby言語で記述

拡張子から定義できない依存関係

- ファイルの依存関係が拡張子ベースではない



ファイル番号に基づく場合

Comparison of task definition

- Make: 各定義を書き下す

```
B00: A00 A01
      prog A00 A01 > B00
B01: A01 A02
      prog A01 A02 > B01
B02: A02 A03
      prog A02 A03 > B02
```

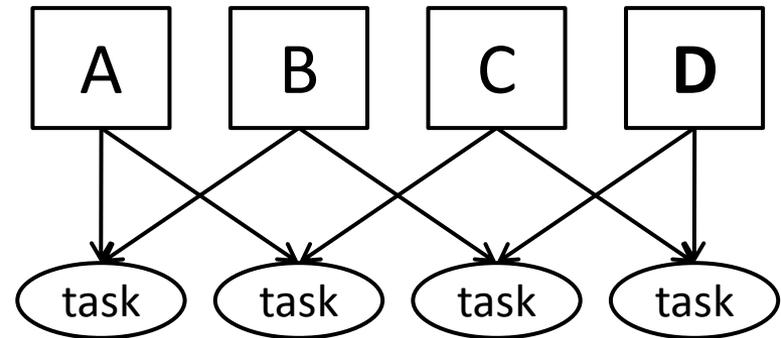
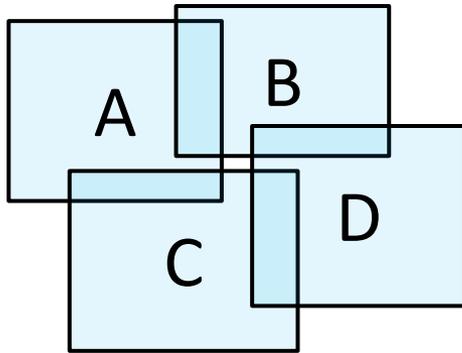
.....

- Rake: ループ内で定義

```
for i in "00".."10"
  file "B#{i}" => ["A#{i}", "A#{i.succ}"] do |t|
    sh "prog #{t.prerequisites.join(' ')} > #{t.name}"
  end
end
```

複雑な依存関係

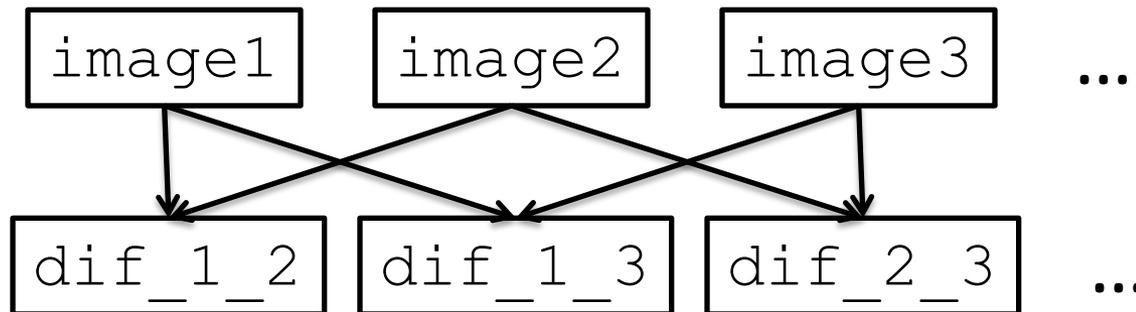
– ジオメトリによって依存関係が決まる



依存関係がファイルに書かれている 場合

- File dependency is given as a list written in a file:

```
$ cat depend_list  
dif_1_2.fits image1.fits image2.fits  
dif_1_3.fits image1.fits image3.fits  
dif_2_3.fits image2.fits image3.fits  
...
```



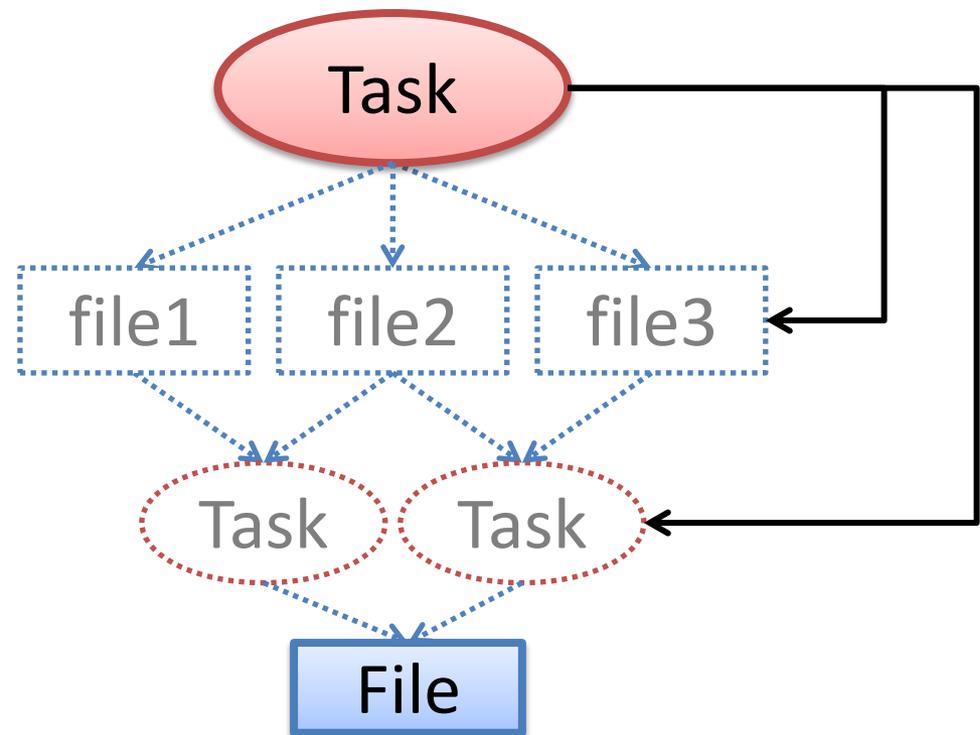
Dependency is given as a file list

- Make:
 - 依存関係を記述したファイルから、Makefile を作成するプログラムが必要
- Rake: ファイルを読み、タスクを定義

```
open("depend_list") { |f|
  f.readlines.each { |line|
    name, file1, file2 = line.split
    file name => [file1,file2] do |t|
      sh "prog #{t.prerequisites.join(' ')} #{t.name}"
    end
  }
}
```

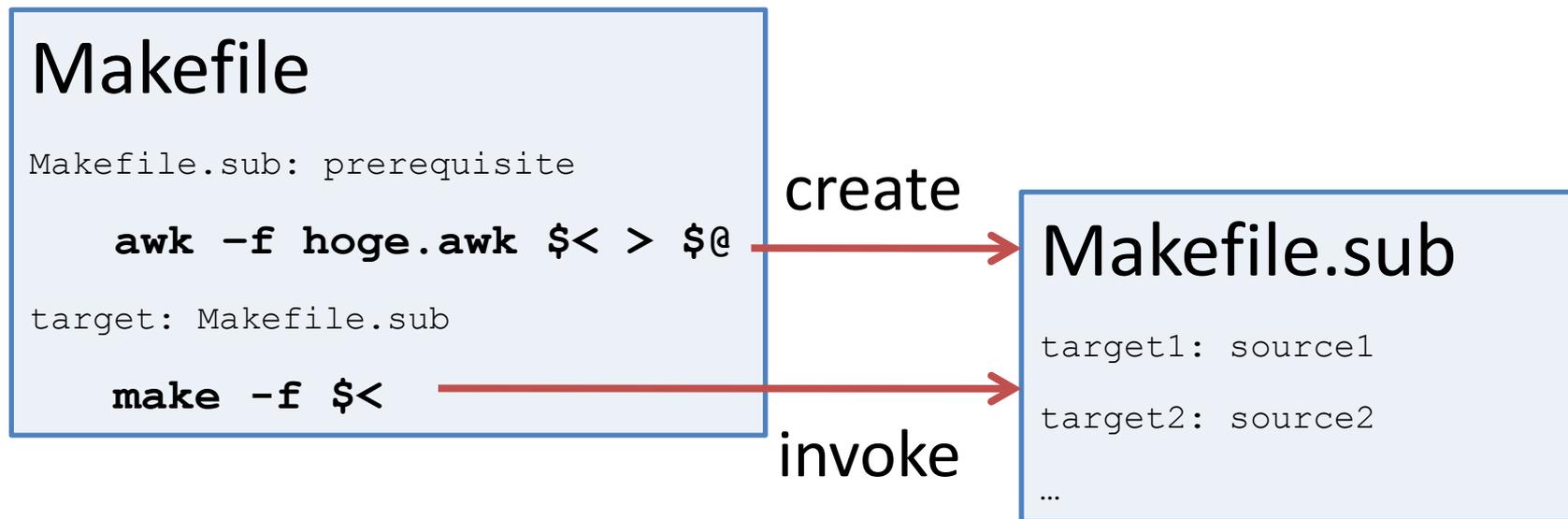
動的なワークフローの生成

- タスクを実行した結果により、それ以降のファイル・タスクが定義できる



Makefileの動的生成

- make実行中にMakefileを生成
- Makefile だけではワークフローを定義できない



Rakeにおける動的タスク生成(失敗)

```
task :A do
  task :B do
    puts "B"
  end
end
task :default => :A
```

- task A の実行中に task B を生成
- しかし task B は実行されない
 - task B が生成される前に、すでに依存関係が決定

Rakeにおける動的タスク生成(成功)

```
task :A do
  b = task :B do
    puts "B"
  end
  b.invoke
end

task :default => :A
```

- 変数 b に、Taskオブジェクトを格納
- invoke メソッドにより、明示的に実行
- Rake により、動的なワークフローが実現

Rakeまとめ

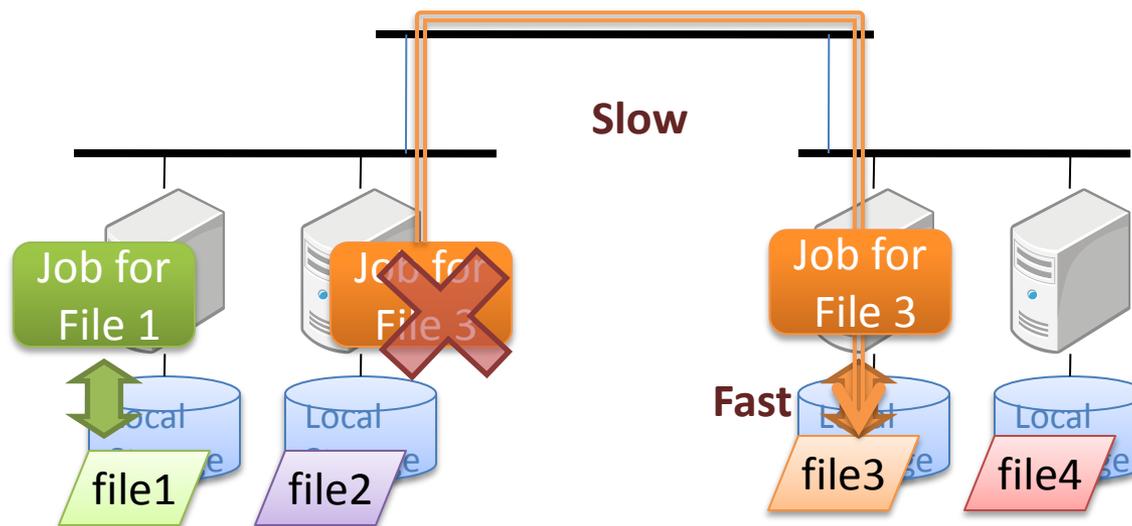
- ワークフローの記述に適した特徴：
 - 複雑な依存関係を記述可能
 - 異なる入力ファイルに対して再利用可能
 - 動的なワークフロー
- Rakeには、並列分散機能はない

並列分散ワークフロー処理システム Pwrake

- 並列分散機能を拡張 : Pwrake
 - Parallel Workflow extension for Rake
 - 「プレイク」と呼んでます
 - <http://github.com/masa16/pwrake>
 - タスク記述は、Rakeと同じ
 - プロセスを並列に実行
 - SSHによる遠隔実行

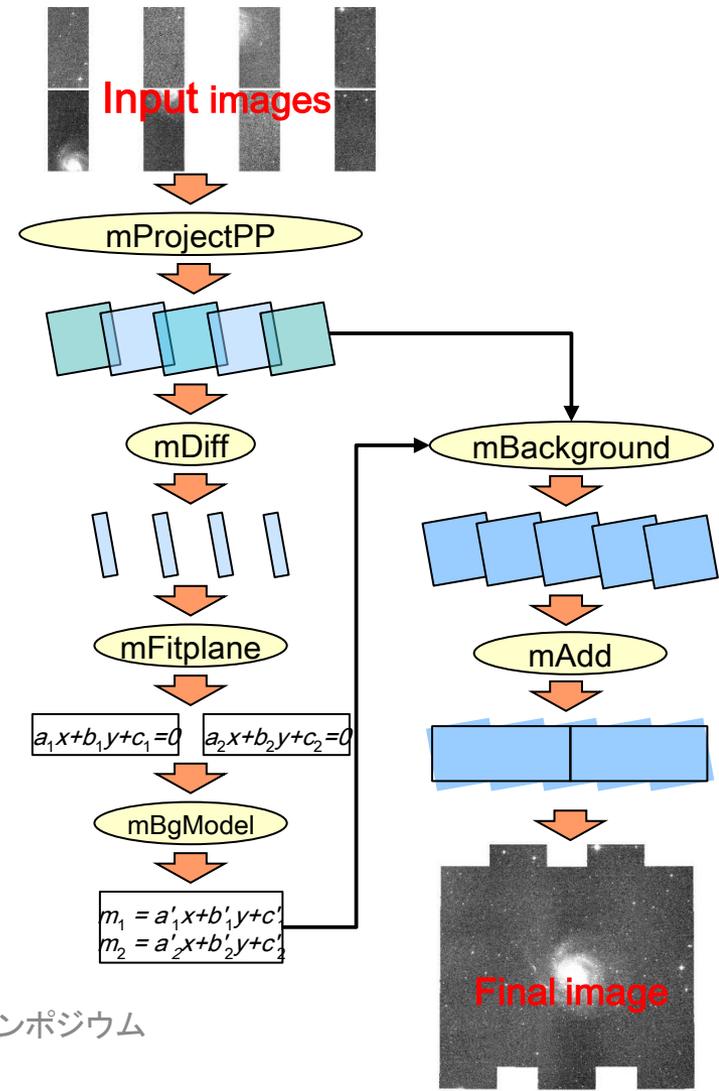
Pwrake のローカルティ機能

- Gfarm ファイルのローカルティに基づき、実行ノードを決定

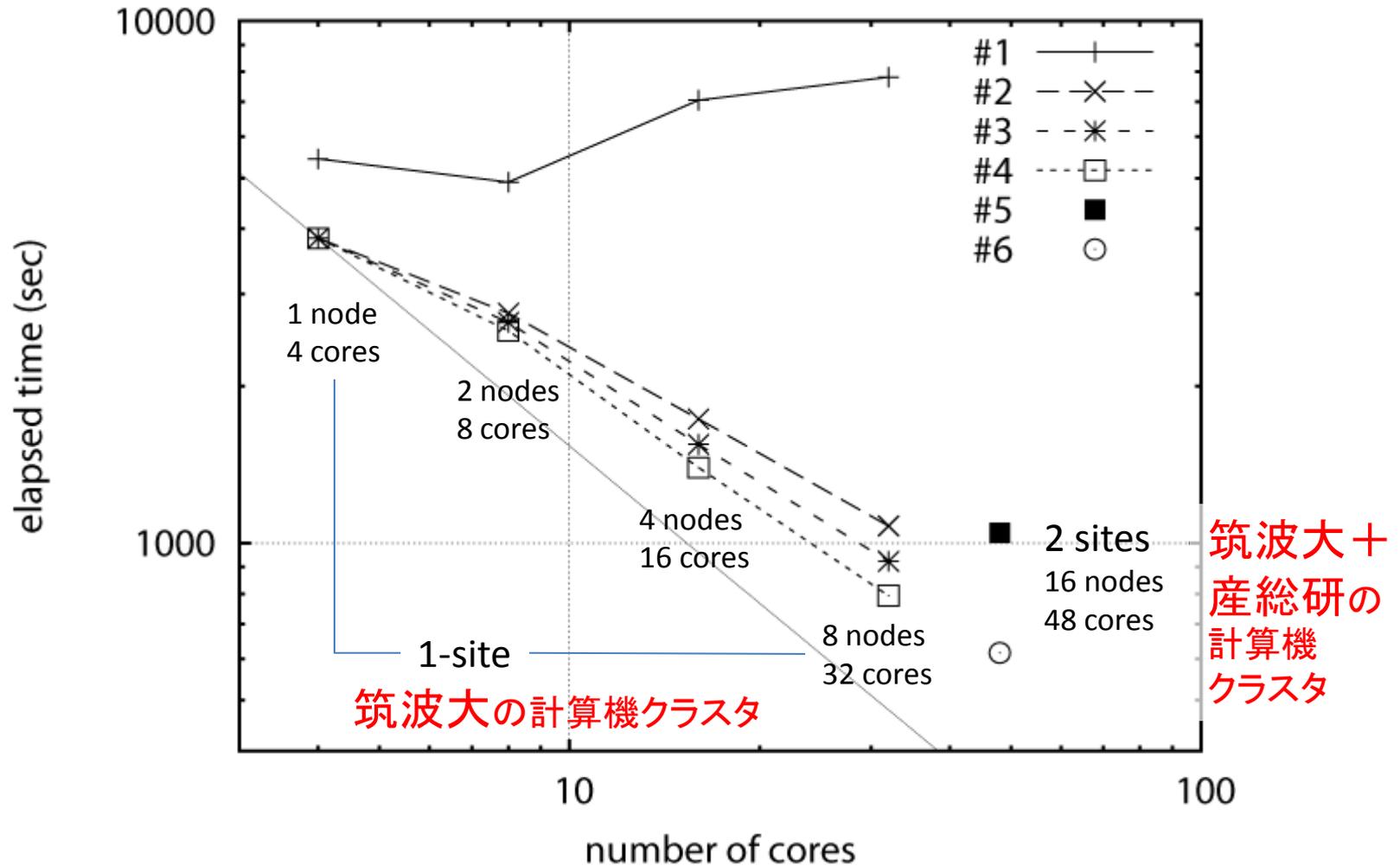


Montage Workflow 性能評価

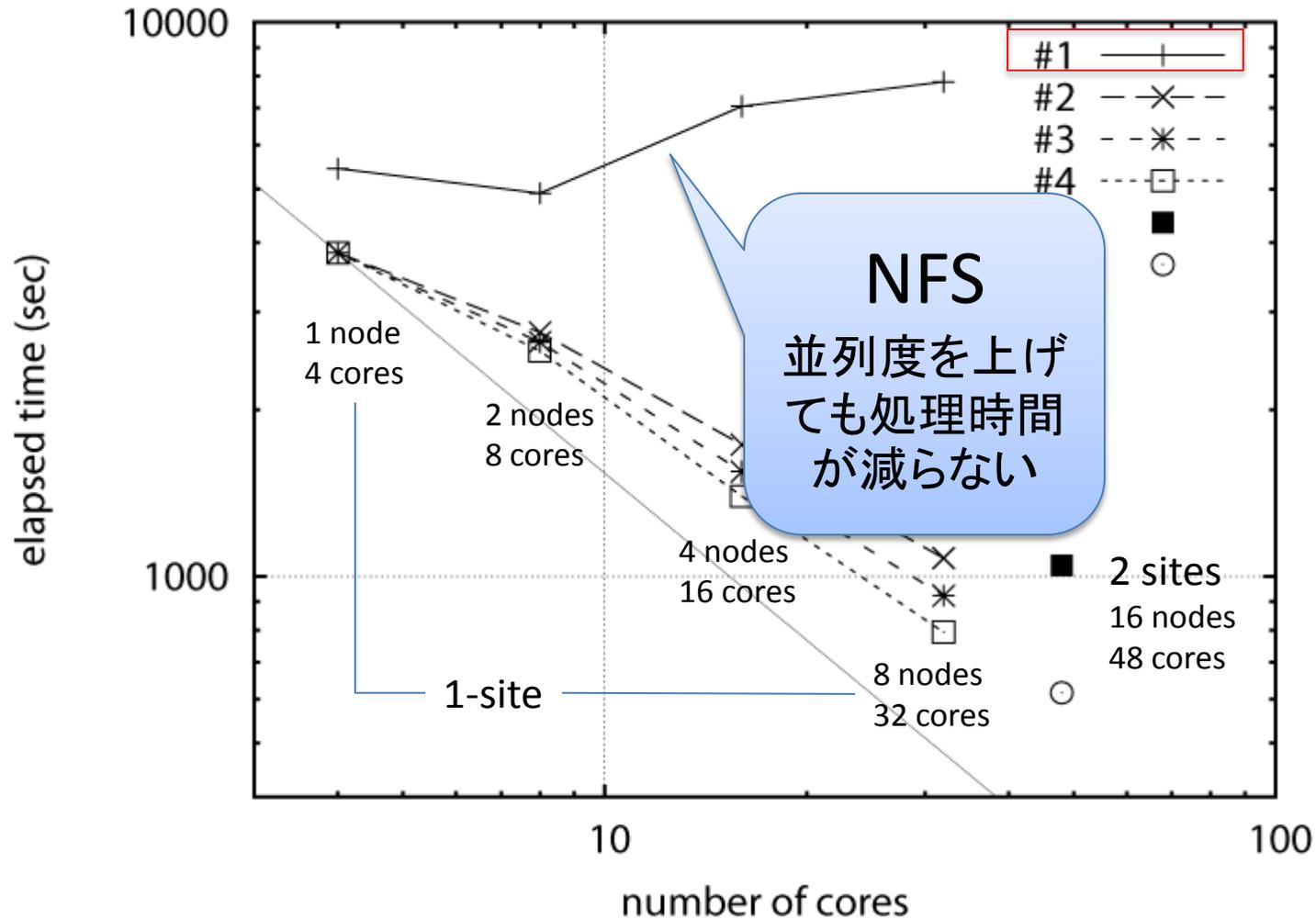
- ワークフローを Rakefileとして記述



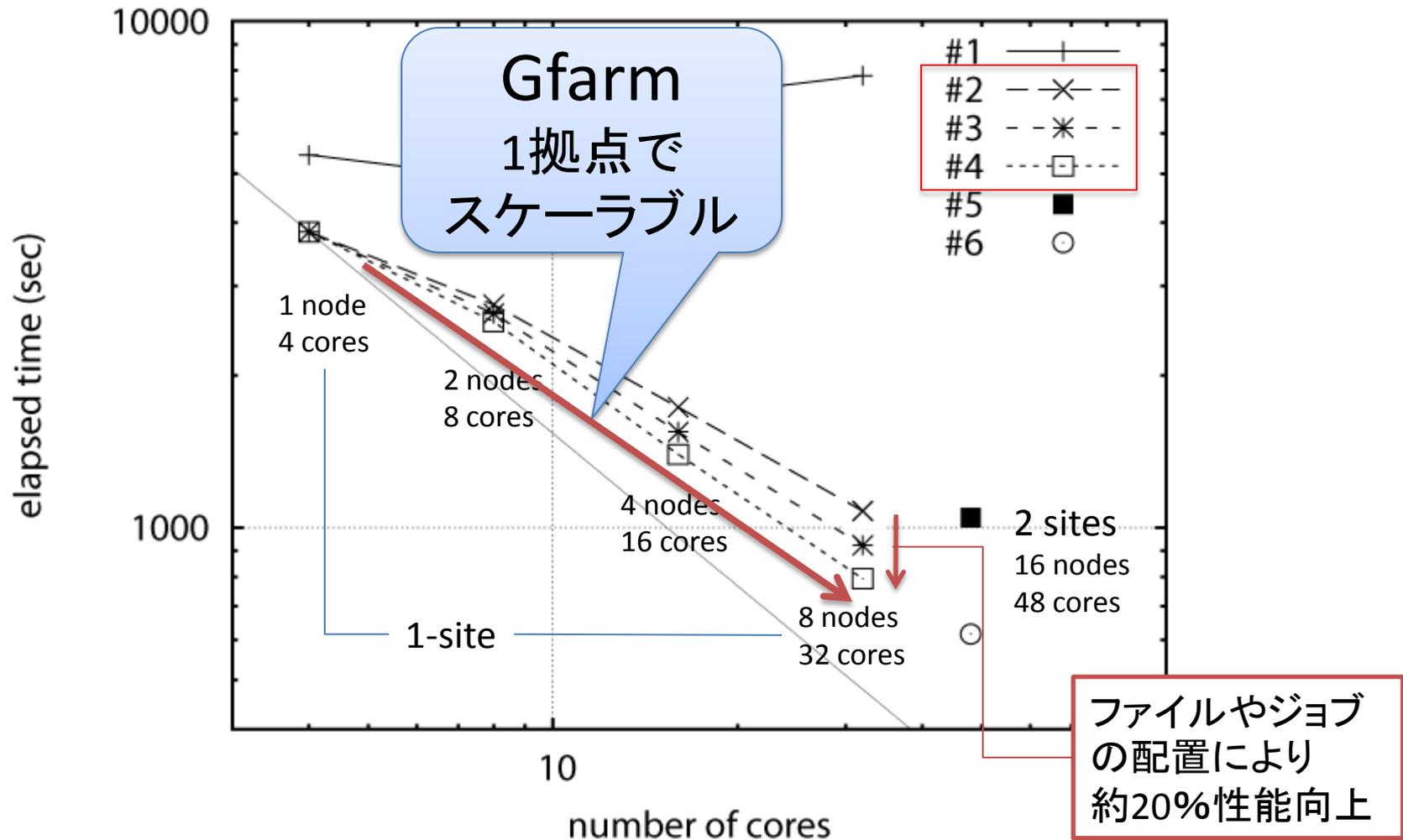
Montage Workflow の実行時間



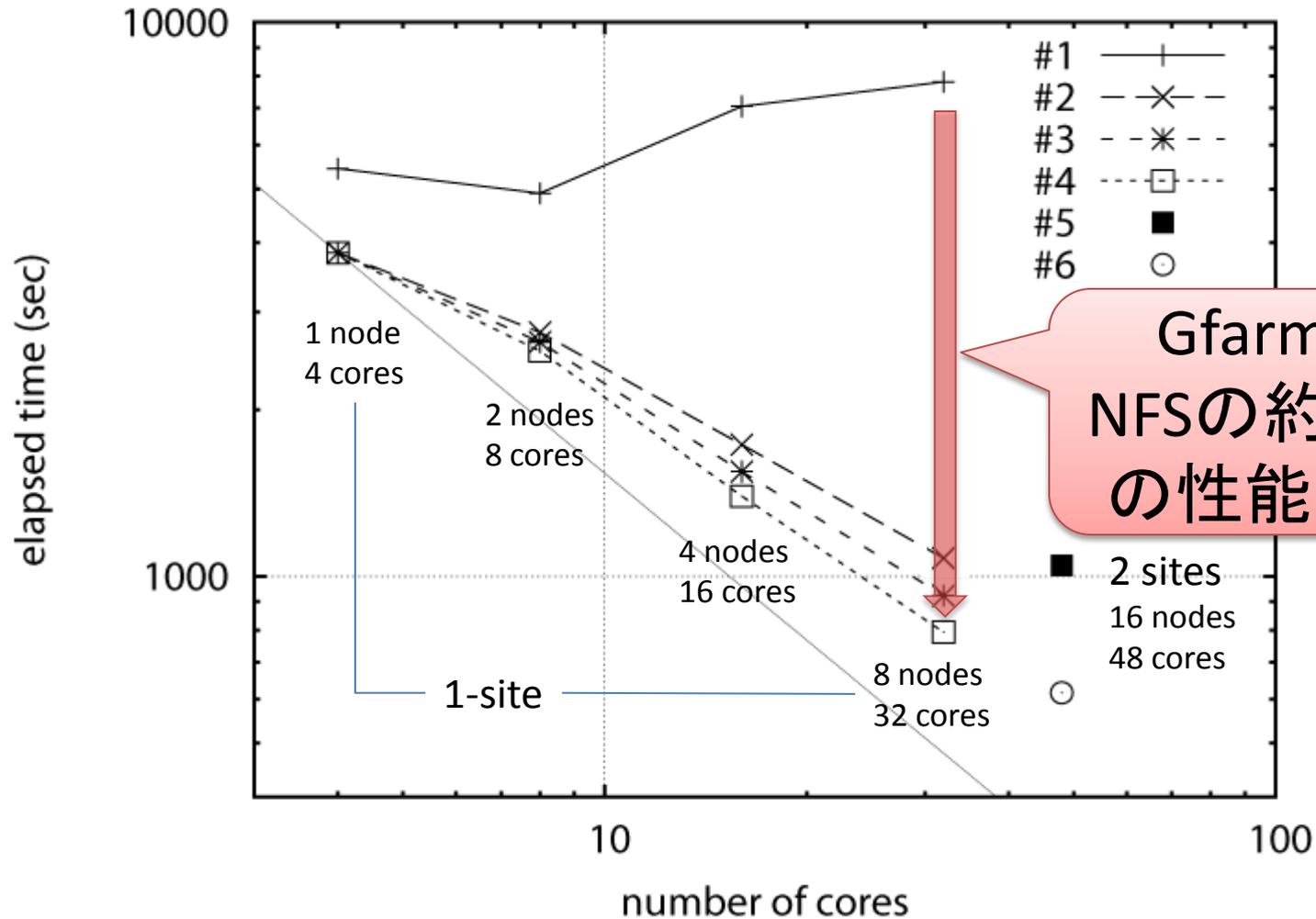
Montage Workflow の実行時間



Montage Workflow の実行時間

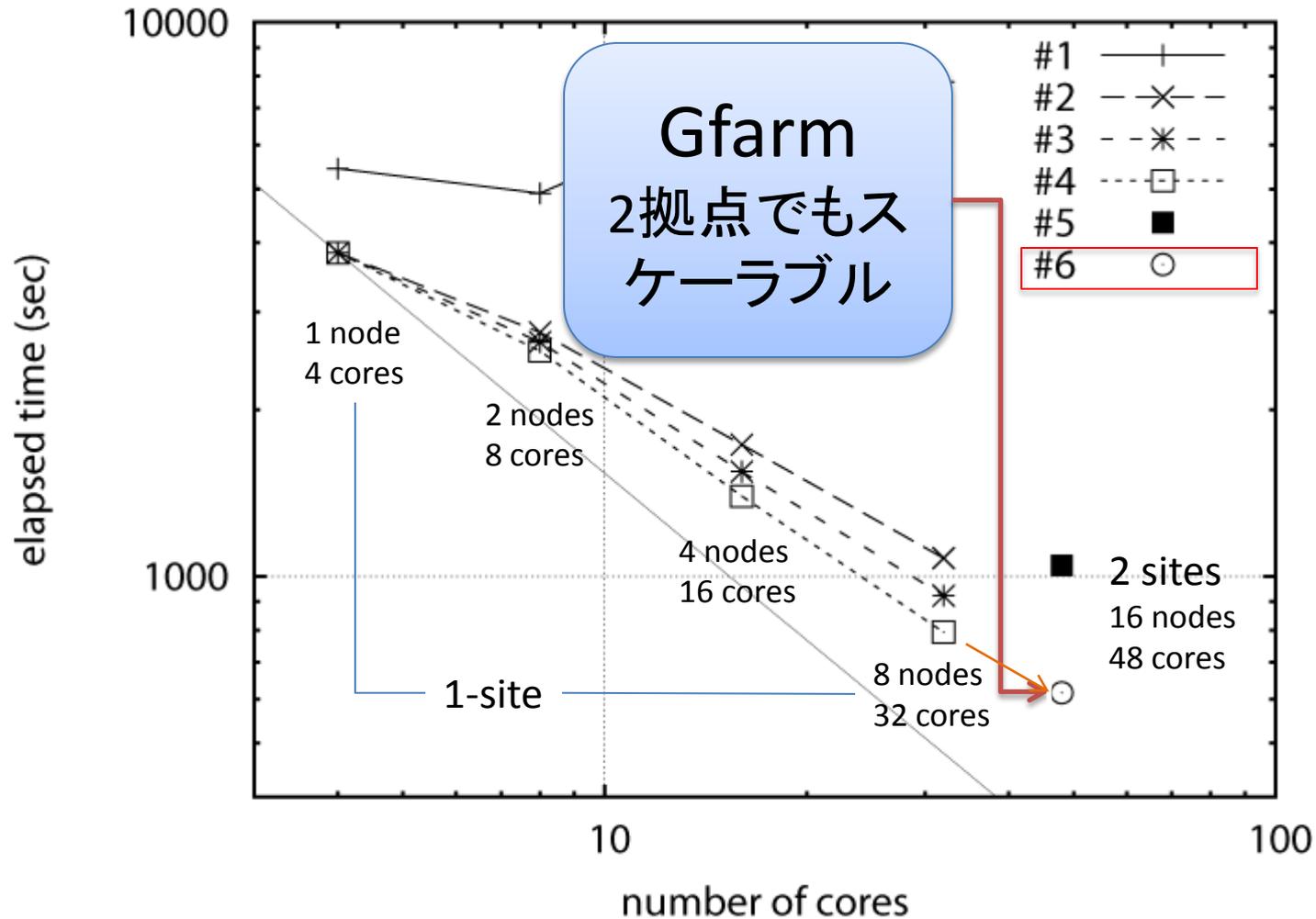


Montage Workflow の実行時間



Gfarmは
NFSの約10倍
の性能向上

Montage Workflow の実行時間

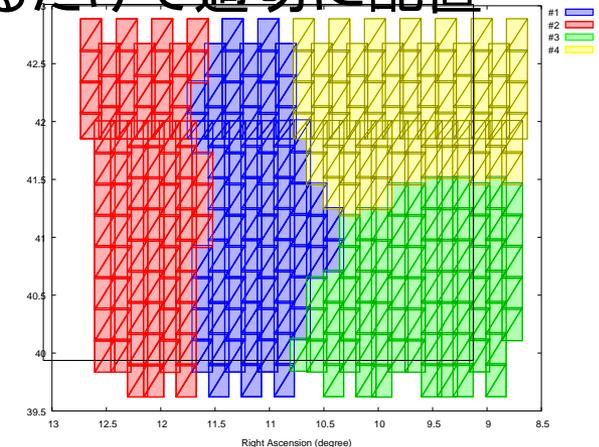
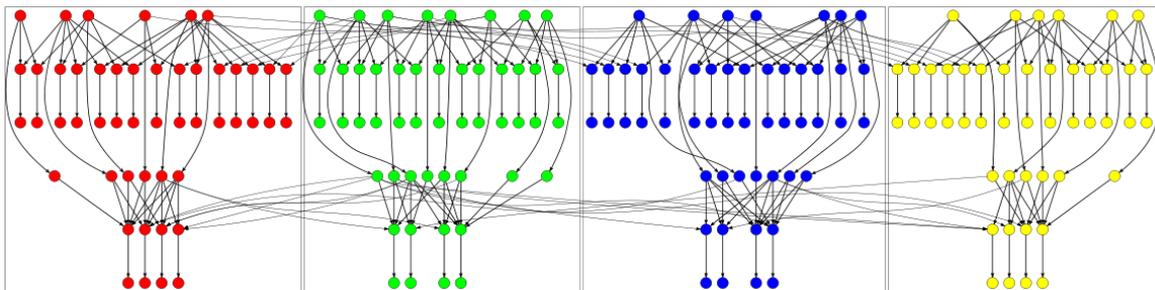


デモ

flash

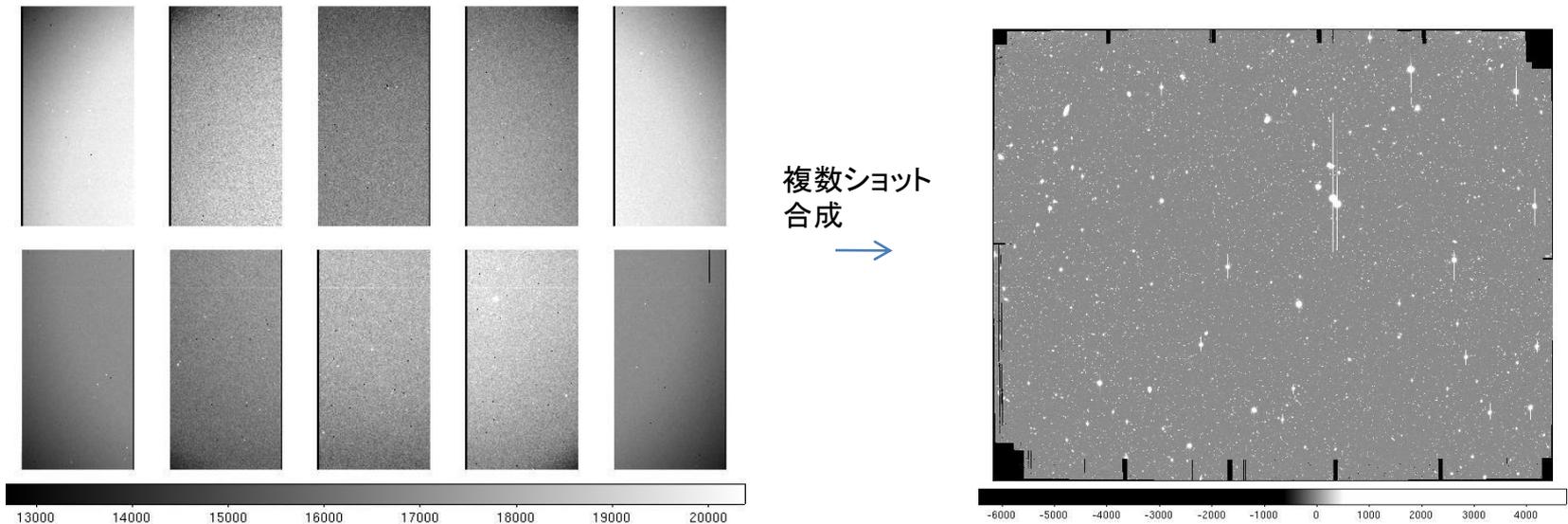
タスク配置による性能向上

- 複数拠点のクラスタを用いたワークフロー実行
- 適切なタスク配置により、拠点間のファイル転送を少なくする
- 「エッジカット最小」となるグラフ分割問題を解くことにより、タスクをグループ化
- 座標によるグループ化と、同等の結果
- 座標情報がなくても、ワークフローを記述するだけで適切に配置



SDFRED (Suprime-cam Deep Field REDuction)

- SprimeCam: すばるの主焦点に 4096x2048 画素の CCD を 5x2 枚並べたカメラ。満月とほぼ同じ大きさの広い視野が特徴。
- SDFRED : SprimeCamが撮影したデータから、位置補正、感度補正などの処理をおこない、1枚の画像に合成するソフトウェア
- <http://subarutelescope.org/Observing/DataReduction/>



SDFRED処理内容

処理の内容	(所要時間A,B)
1. 画像ファイル名の変換および画像の確認	(1秒,1秒)
2. bias引きおよびoverscanの切り取り	(20秒,1分)
3. flat 作り	(6分,20分)
4. 感度補正 (flat fielding)	(30秒,2分)
5. 歪補正および微分大気差補正	(1分,6分)
6. PSF 測定	(*,*)
7. PSF 合わせ	(40分,80分)
8. sky の差し引き	(1分,3分)
9. AG probeの影を自動でマスク	(15秒,2分)
10. 画像を目で見て、悪い部分をマスク	(15秒,1分)
11. 組み合わせ規則作り(matching)	(2分,15分)
12. 組み合わせ(mosaicing)	(3分,12分)

(所要時間)は国立天文台の ana.adc (Intel Xeon 3.0GHz) のローカル作業領域 /wa を使って、a) 練習用データ(5shot)を解析した場合にかかった時間、b) 別のデータ(17shot)を解析した場合にかかった時間

SuprimeCam Data Reduction

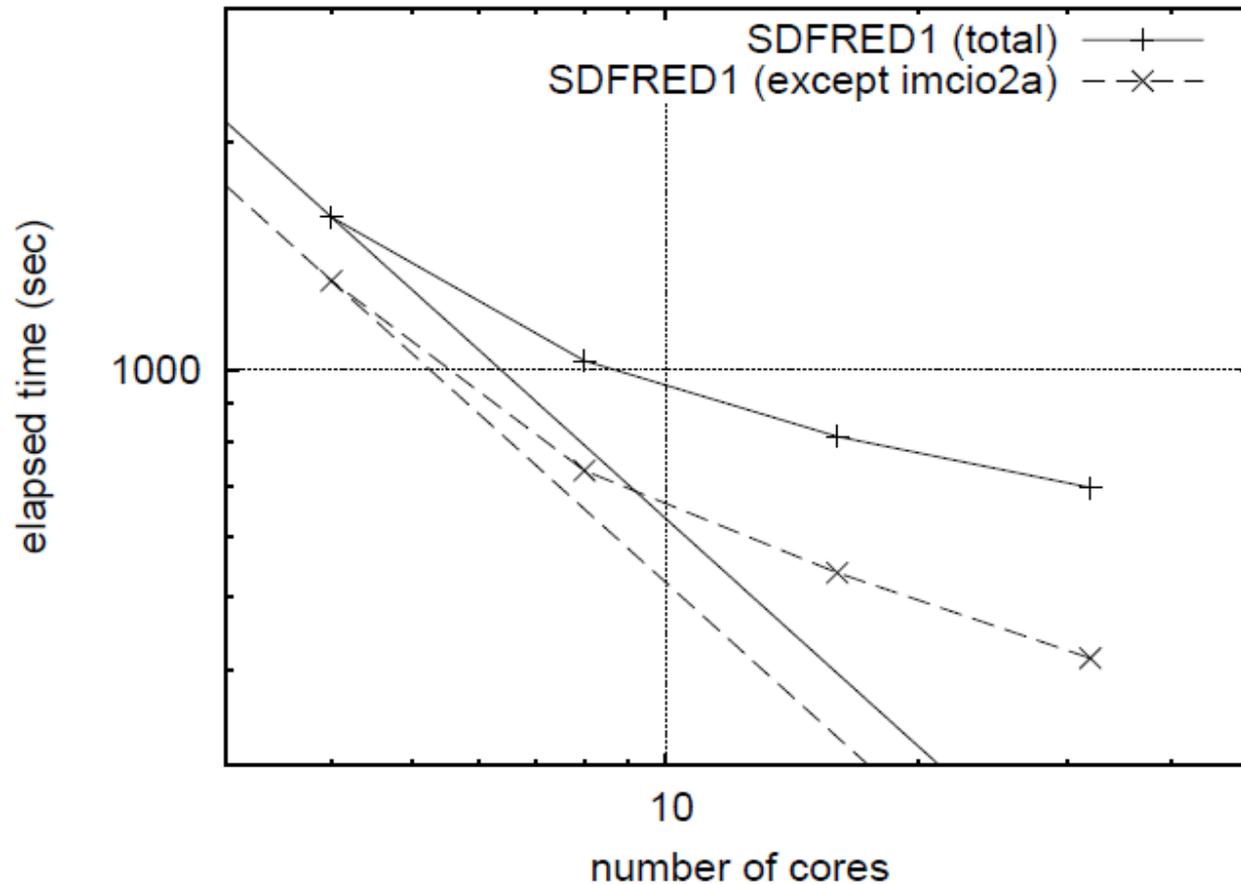
SDFRED1

- 下記スクリプトをRakefileで記述
 - overscansub.csh
 - mask_mkflat_HA.csh
 - blank.csh
 - ffield.csh
 - distcorr.csh
 - fwhmpsf_batch.csh
 - fwhmpsf.csh
 - psfmatch_batch.csh
 - psfmatch.csh
 - skysb.csh
 - mask_AGX.csh
 - blank.csh
 - makemos.csh
- トレーニング画像の自動処理に成功

ワークフローのグラフ



SDFRED1ワークフローの経過時間



Pwrake開発の今後の予定

- 障害対策
 - 障害情報の取得
 - タスクの再試行
 - 故障ノードの判別
- 機能向上
 - 資源に応じた実行制御
 - 接続方法
- 性能向上
 - タスク配置アルゴリズムの向上
 - ファイル複製の自動配置

まとめ

- 大量データの迅速な解析には、並列処理が必要
- ワークフロー記述言語として、Rakeを採用
- 並列分散ワークフロー処理システムPwarkeを開発
 - URL: <http://github.com/masa16/pwrake>
- 分散ファイルシステムGfarmを用いることにより、スケーラブルな性能向上を達成