Pwrake/Gfarmによる ビッグデータ解析

田中昌宏

筑波大学 計算科学研究センター、JST/CREST

「ビッグデータ」といえば…

- ペタバイト級のデータ
- 大量データのデータマイニング
  - には直接関わっていません
- 「スケールアウト」が可能な基盤システム
  - Gfarmファイルシステム
  - Pwrakeワークフローシステム
    - について話します

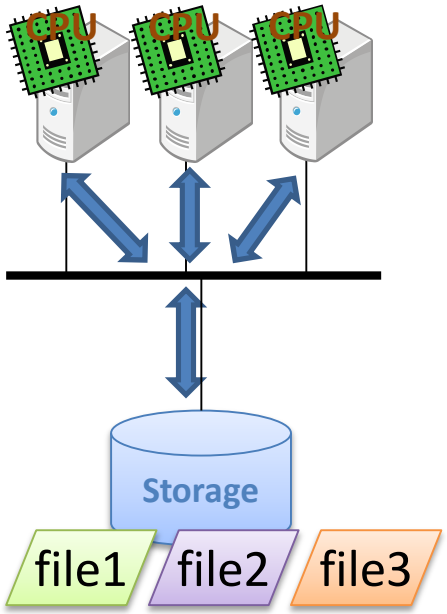
本日の内容

- Pwrakeワークフローシステム
  - Pwrakeシステムの概要
  - タスクスケジューリングに関する研究
- 関連システム
- Gfarm/Pwrakeによる科学データ処理

- Amazon EC2 で Gfarm/Pwrake のデモ

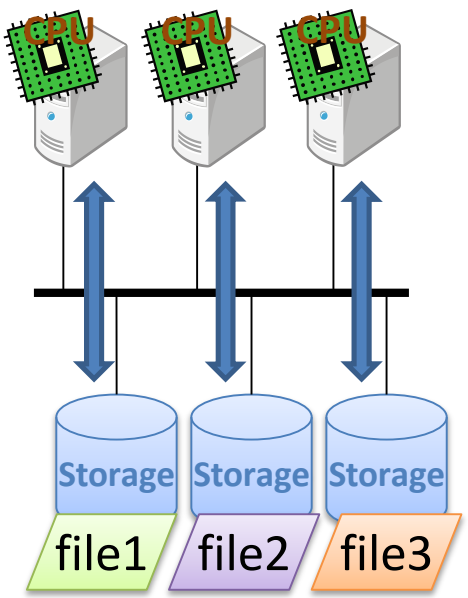
ネットワークファイルシステムの比較

NFS



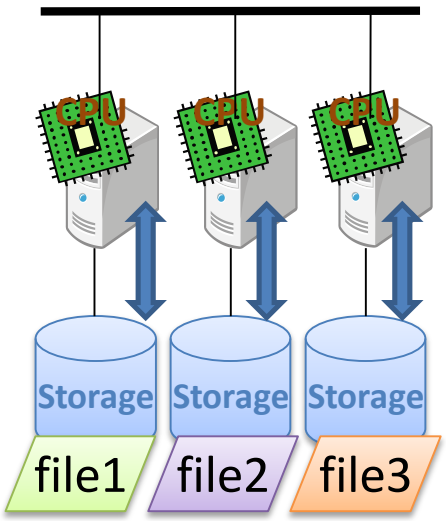
1ストレージにアクセスが集中

分散ファイルシステム



ネットワーク経由のアクセス

Gfarmファイルシステム



ローカルアクセスにより
スケールアウト可能

ワークフローシステムPwrake

- Rake: Rubyによる make に似たツール
  - ファイル入出力の依存関係に従って、コマンドラインを実行
  - Ruby本体に付属
- Pwrake: Parallel Workflow extension for RAKE
  - https://github.com/masa16/Pwrake/
  - Rake を拡張し、複数ノードで並列実行

Rakefileによるワークフロー記述

Rakefile

```
SRCS = FileList["*.c"]
OBJS = SRCS.ext("o")

task :default => “foo"

file “foo" => OBJS do |x|
  sh "cc -o #{x} #{OBJS}"
end

rule ".o" => ".c" do |x|
  sh "cc -o #{x} -c #{x.source}"
end
```

rule
拡張子パターンの他、正規表現、
スクリプトによる変換ルールを記述可能
⇒ 科学ワークフロー記述に適する

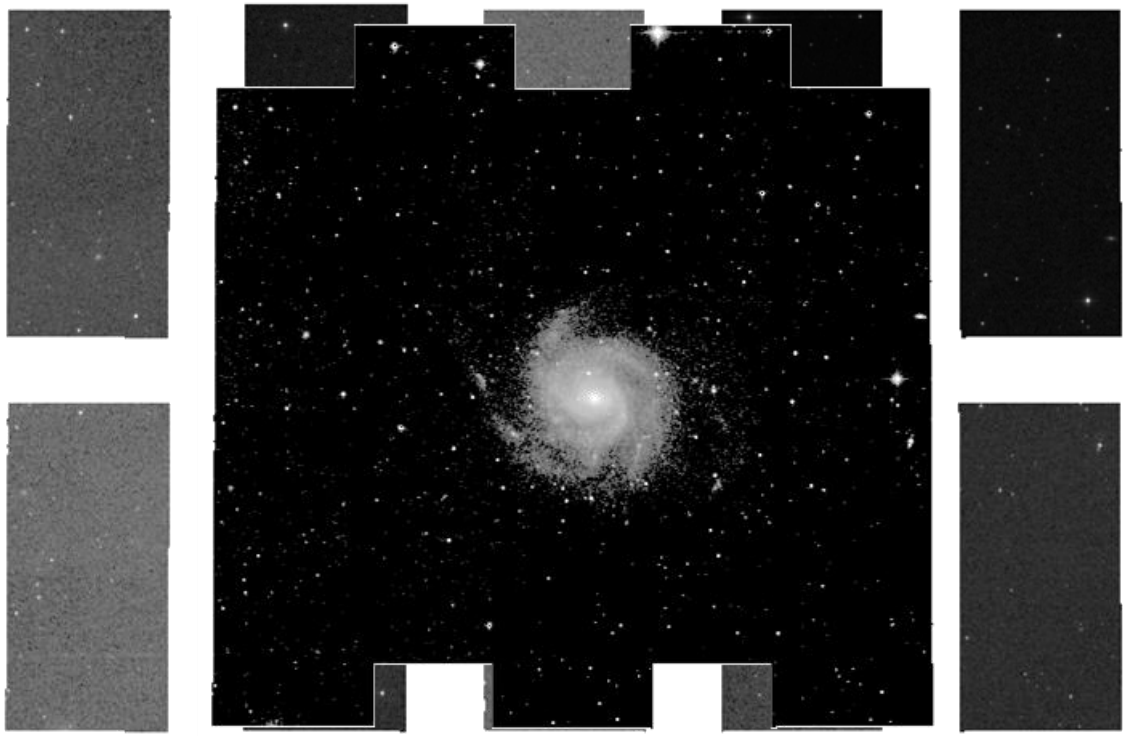
Makefile (GNU make)

```
SRCS := $(wildcard *.c)
OBJS = $(subst .c,.o,$(SRCS))



foo: $(OBJS)
        cc -o $@ $^


%.o : %.c
        cc -o $@ -c $<
```

Pwrake の方針:
Rakefile と互換性を保つ

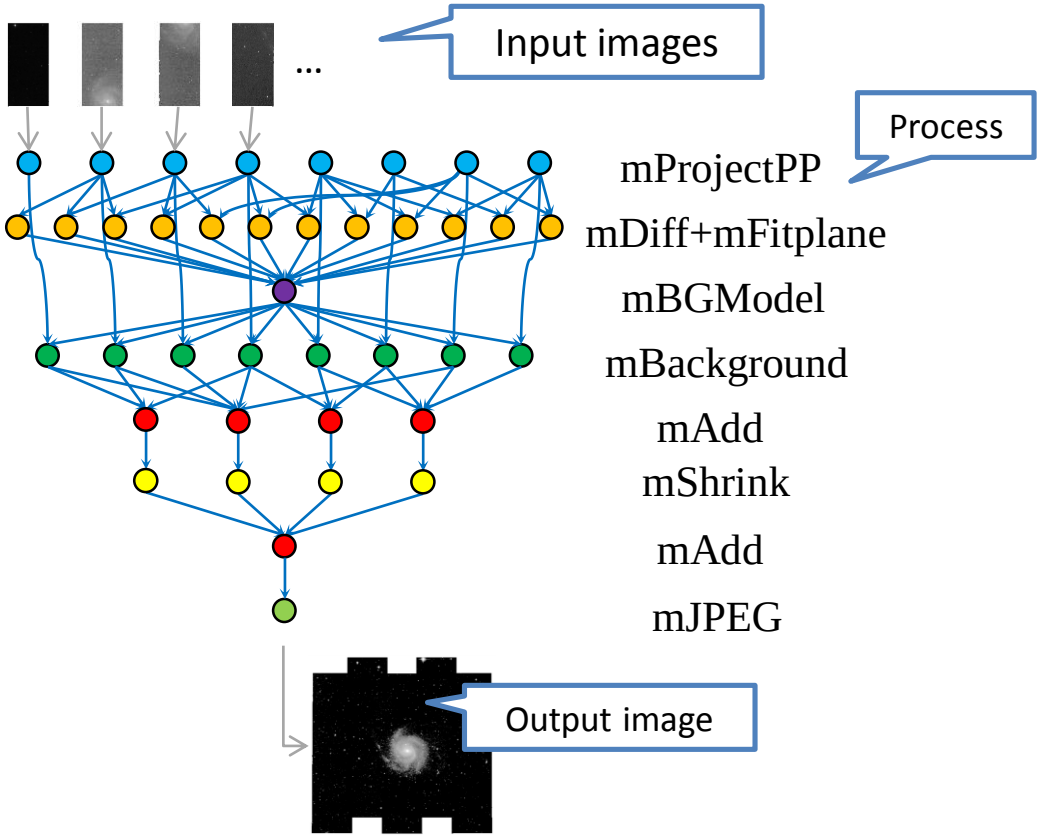
科学ワークフローの例: Montage

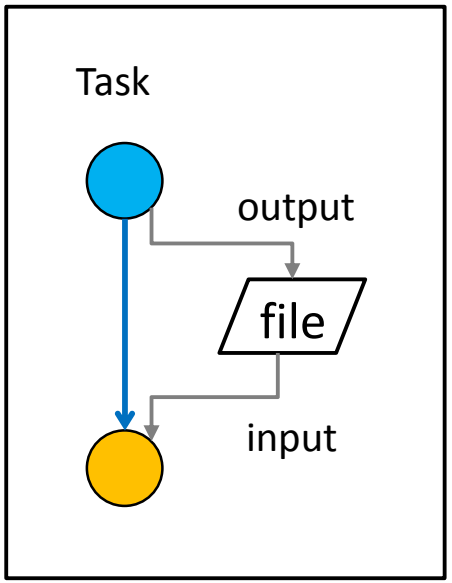
- 少しずつずらして撮像した天文画像を、1つの画像に結合するためのソフトウェア



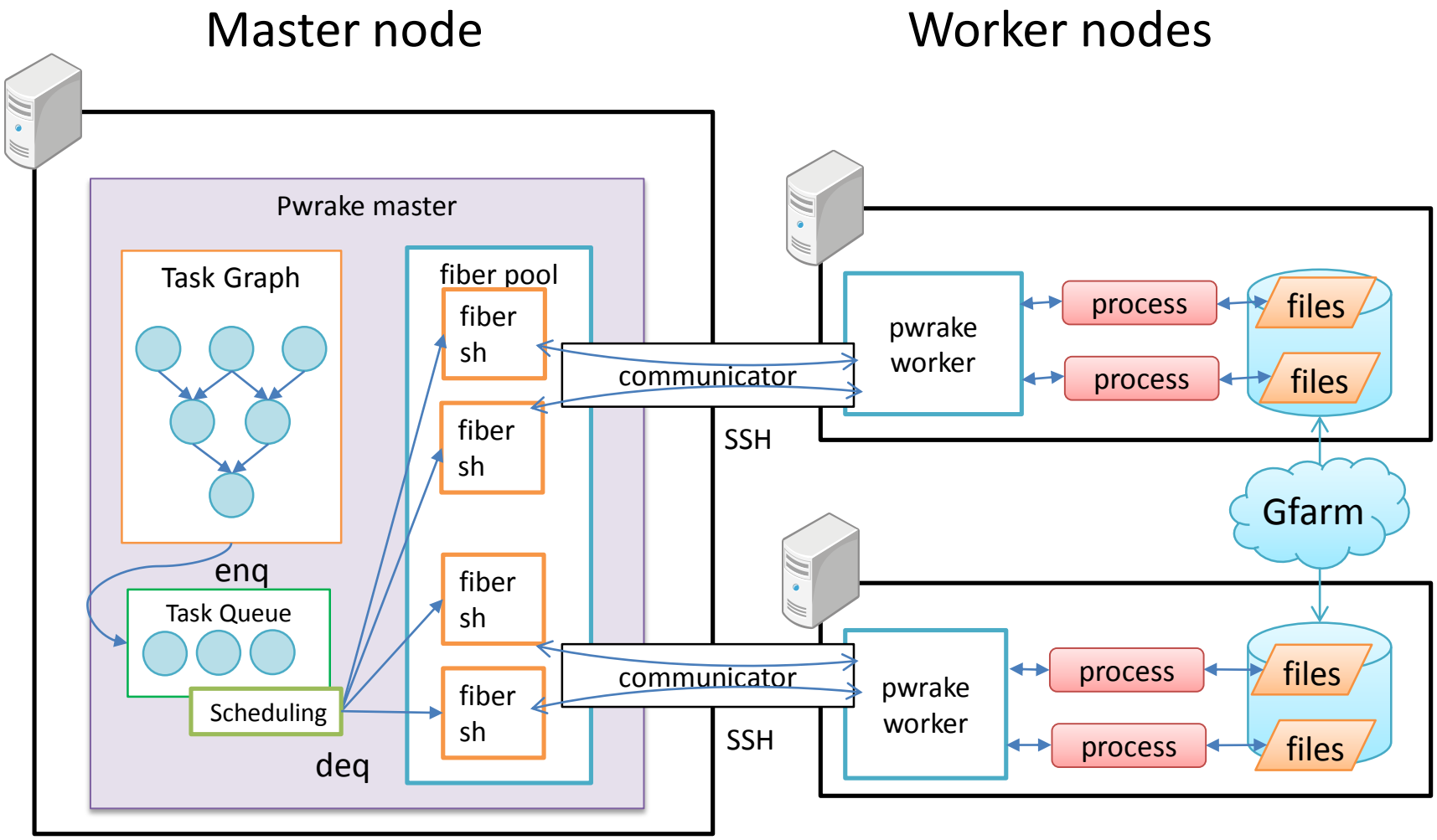
科学ワークフローの例: Montage

Workflow DAG (Directed Acyclic Graph)



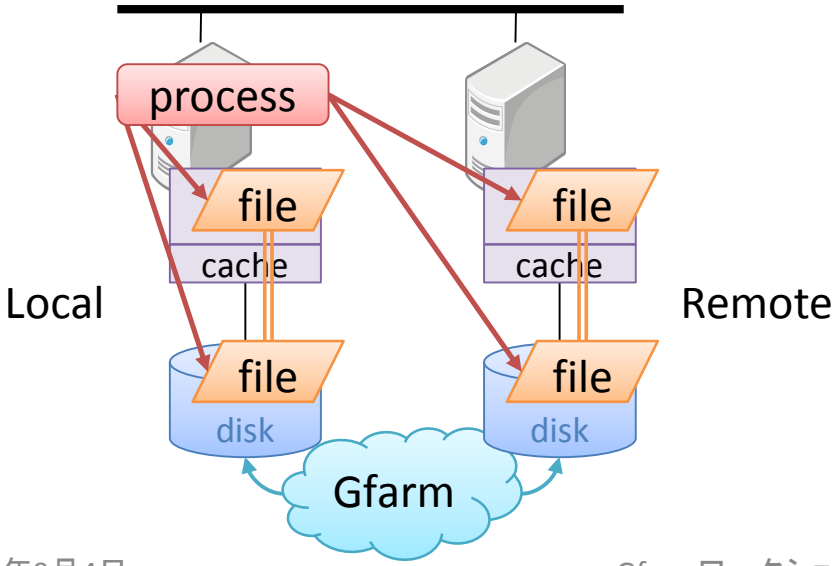


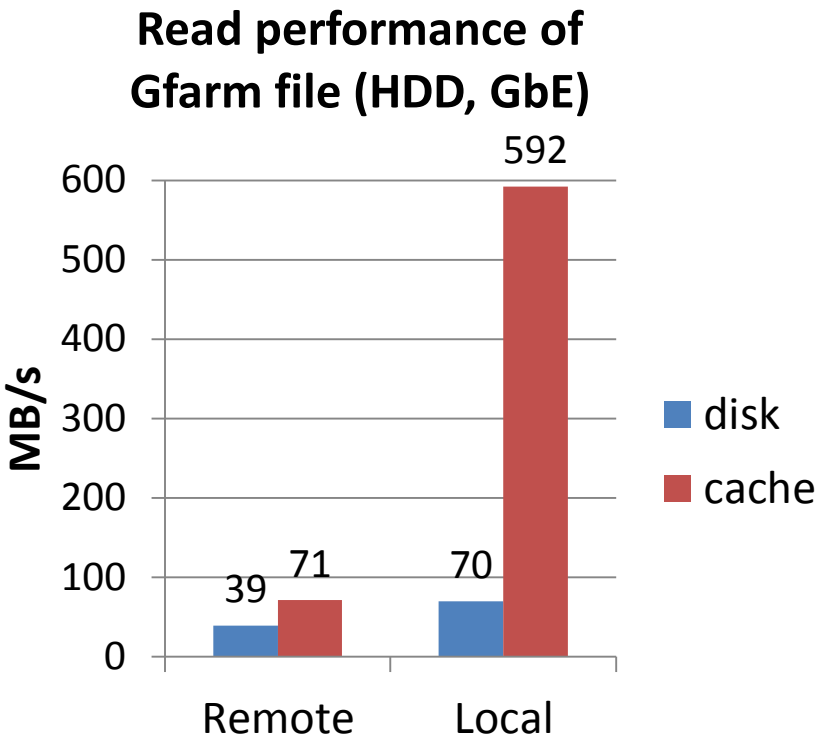
Pwrake の構成


Master node
Worker nodes
Pwrake master
Task Graph
fiber pool
fiber
sh
fiber
sh
fiber
sh
fiber
sh
enq
Task Queue
Scheduling
deq
communicator
communicator
SSH
SSH
pwrake
worker
pwrake
worker
process
process
process
process
files
files
files
files
Gfarm

ファイルアクセスパターンとI/O性能

- ファイルアクセス速度
  - Local > Remote
  - Cached > Disk
    - Disk Cache (Buffer/Page Cache)

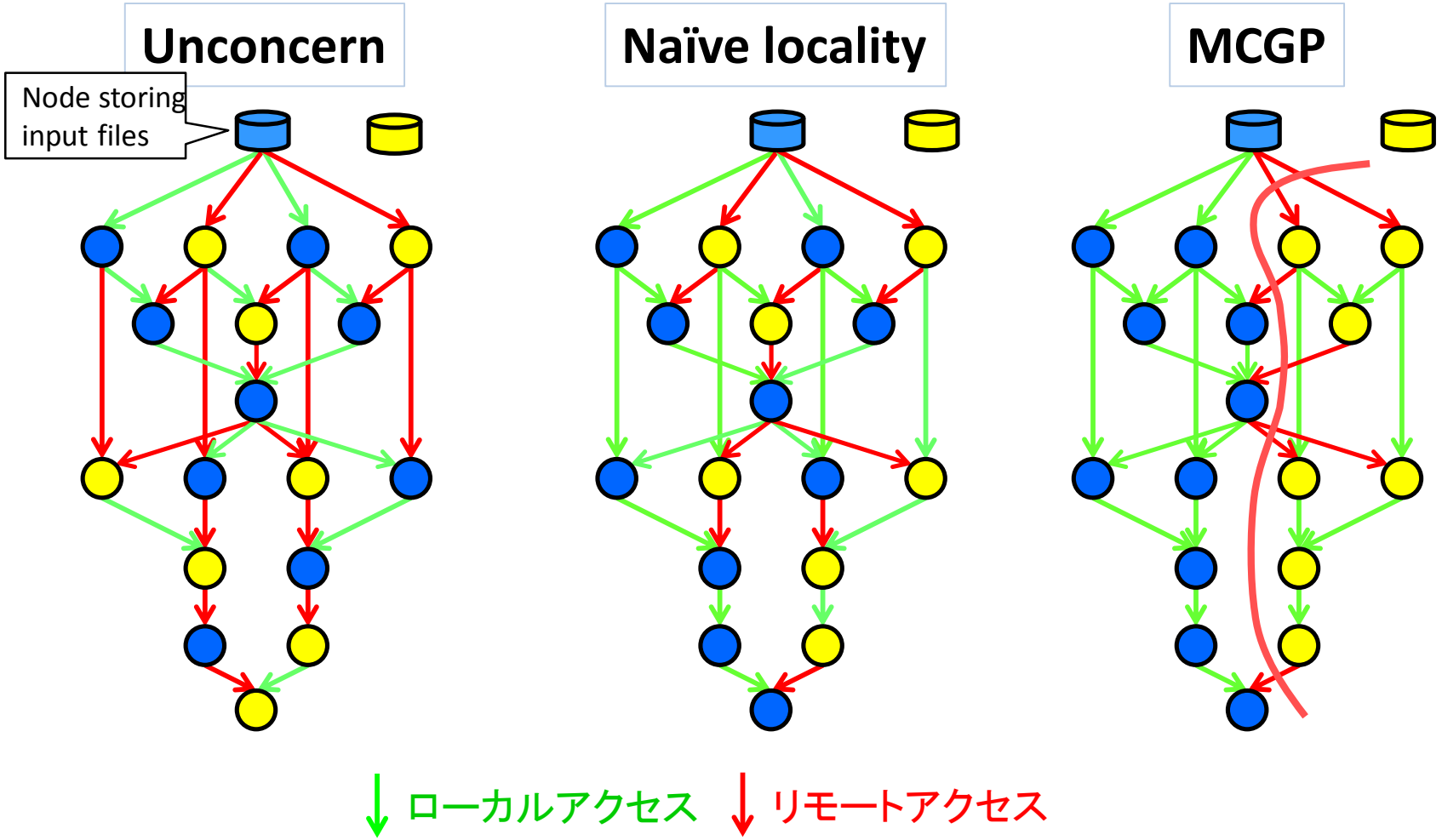




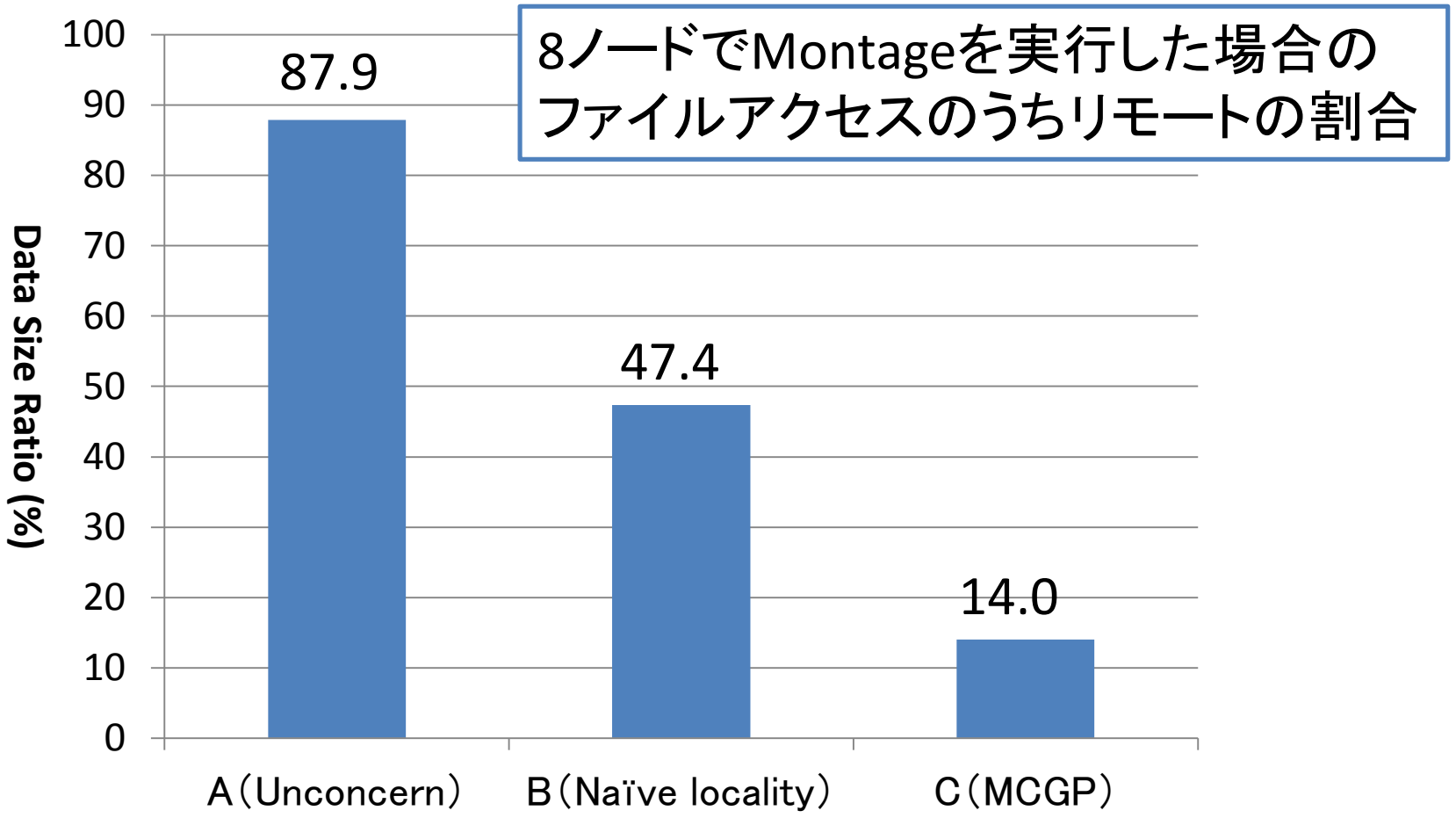
IOを考慮したタスクスケジューリングに関する研究

- ローカルアクセスの向上
  - “Workflow scheduling to minimize data movement using multi-constraint graph partitioning”
  - CCGrid 2012
  - タスク実行ノードを決めるスケジューリング
- キャッシュ効率向上
  - “Disk Cache-Aware Task Scheduling For Data-Intensive and Many-Task Workflow”
  - IEEE Cluster 2014
  - タスク実行順序を決めるスケジューリング

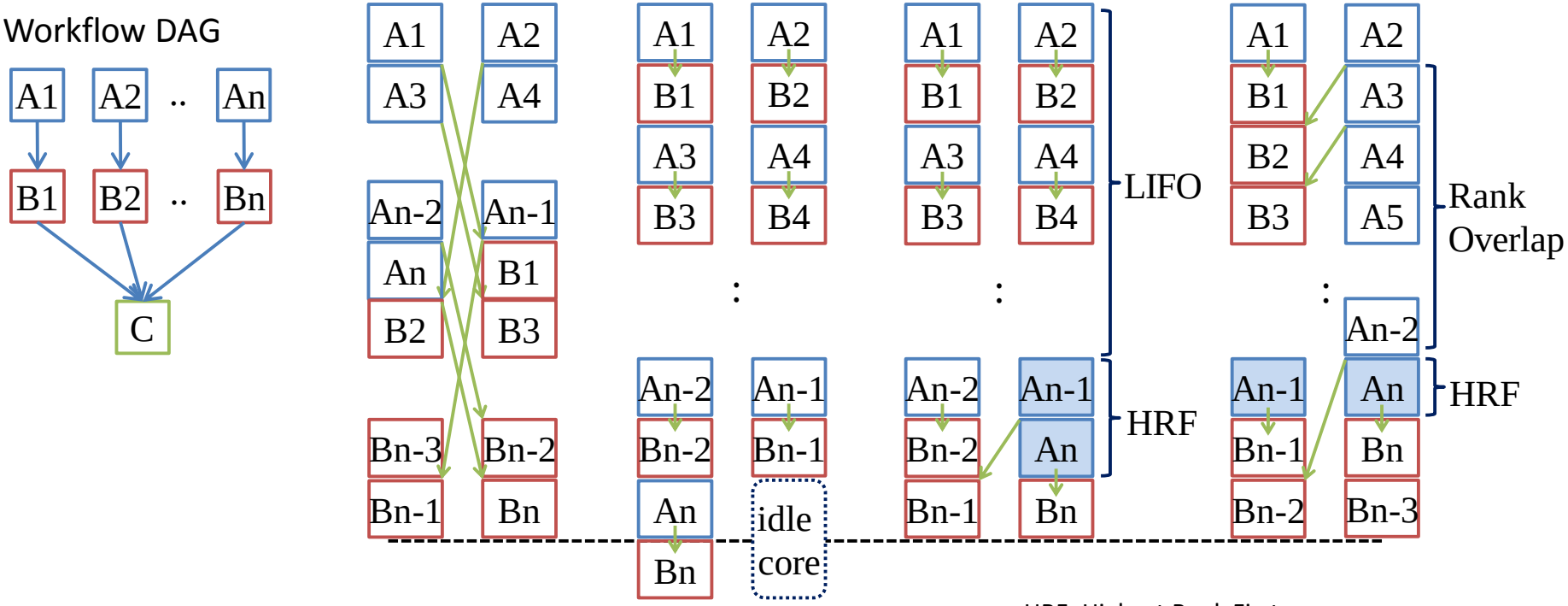
Multi-Constraint Graph Partitioningによるローカルアクセスの向上



Multi-Constraint Graph Partitioningによるローカルアクセスの向上



キャッシュ効率のためのタスクスケジューリング

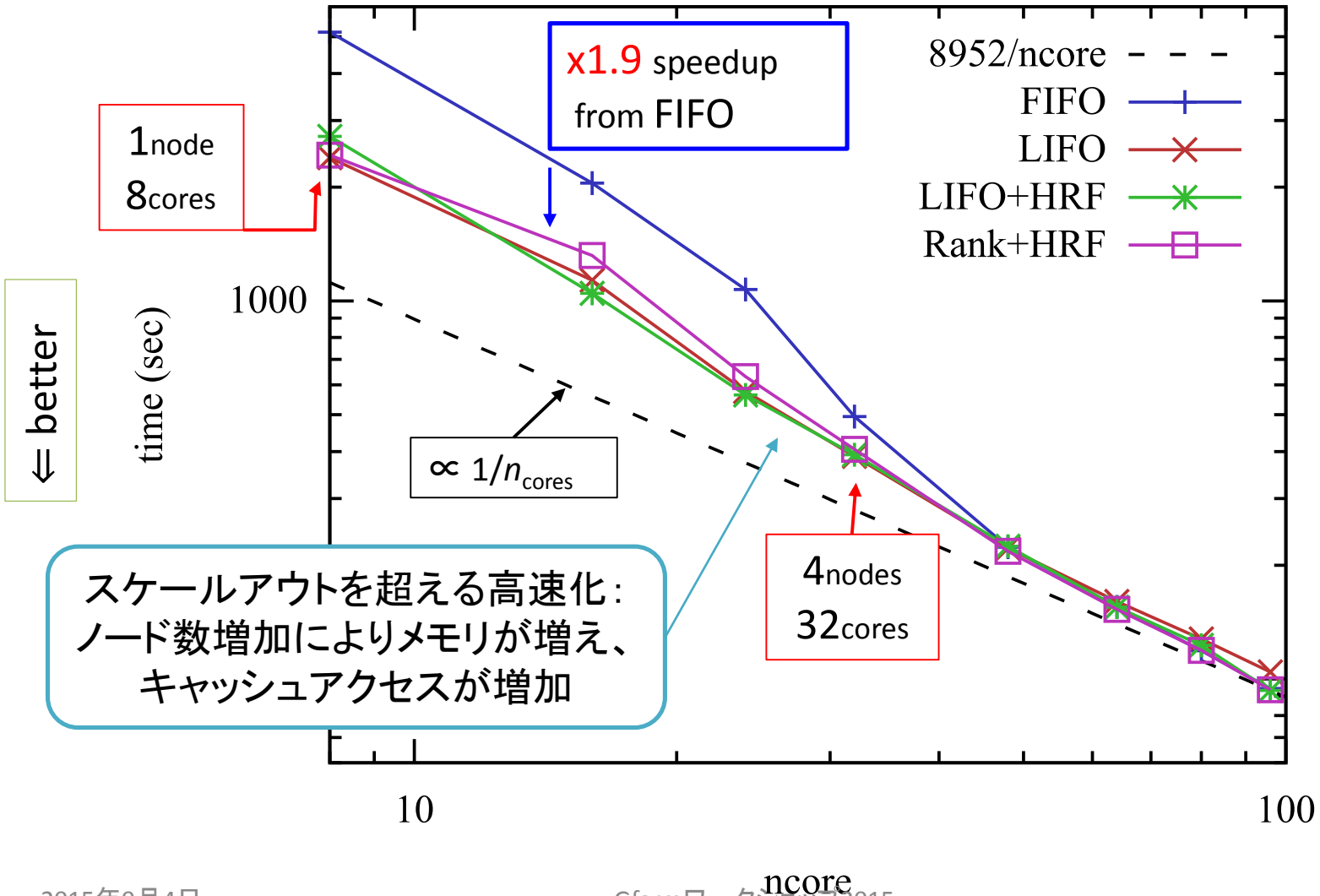


HRF: Highest Rank First

	FIFO (HRF)	LIFO	LIFO+HRF	Rank Equalization+HRF
Disk Cache	×	◎	◎	○
Trailing Task	○	×	○	○
Task Overlap	×	×	×	○

キャッシュ効果による性能向上

(1-12 nodes, Logarithmic graph)



関連するワークフローシステム

- GXP make
- Swift
- Hadoop

関連システム1

- **GXP make**
  - GXP: 並列/分散シェル
    - http://www.logos.ic.i.u-tokyo.ac.jp/gxp/
    - 田浦さん（東大）らによる開発
    - Pythonで実装
  - GNU make のタスクを、GXPで実行
    - Makefile でワークフローを定義
    - SHELL=mksh により、コマンドラインを横取り
    - 入出力ファイル名までは取れないので、高度なスケジューリングは難しい

関連システム2

- **Swift**
  - http://swift-lang.org
  - シカゴ大学他による研究開発
  - ワークフロー定義言語
    - 独自の遅延評価型の並列スクリプト言語
  - Swift/T
    - Armstrong et.al., “Compiler Techniques for Massively Scalable Implicit Task Parallelism,” in *SC14*.
    - **1.47 billion** tasks/s on 524,288 cores of Blue Waters
  - 特殊目的の独自言語の学習コスト

関連システム3

- Apache Hadoop
  - http://hadoop.apache.org/
  - 大規模データの分散処理の定番
  - MapReduceフレームワーク
    - 任意のワークフローを実行できない
  - HDFS
    - 既存のツールからファイル読み書きできない

Gfarm/Pwrakeによる科学データ処理

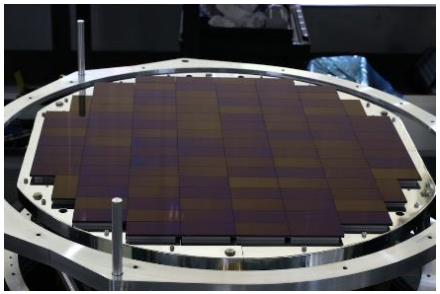
- すばる Hyper Suprime-Cam (HSC)
  - NICTサイエンスクラウド

すばる望遠鏡 Hyper Suprime-Cam(HSC)

(画像クレジット: 国立天文台・HSC プロジェクト)

すばる望遠鏡

HSC 全体像



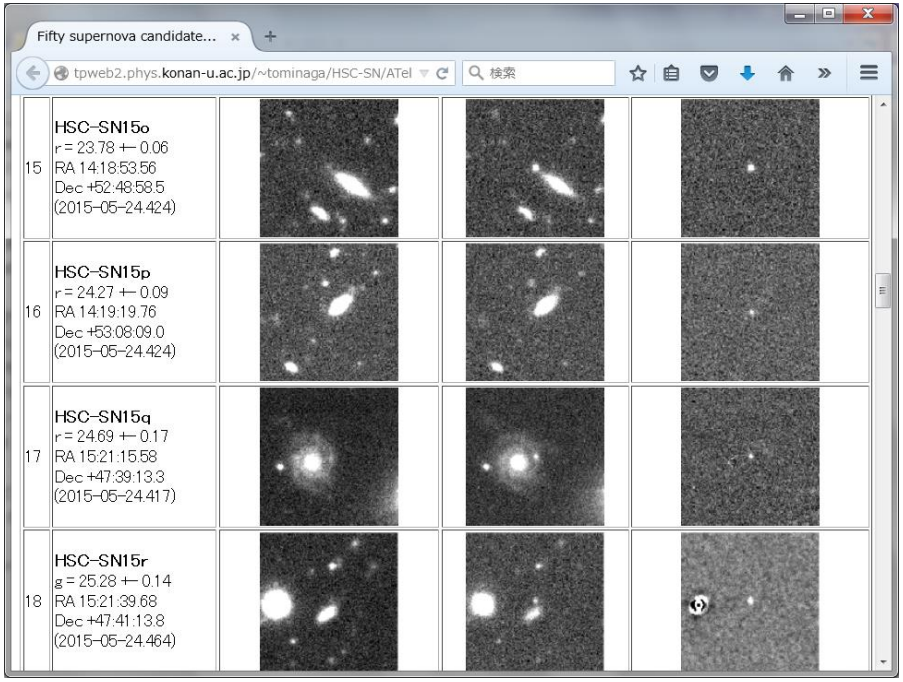
HSC 焦点面CCD

有効視野角: 1.5度角（Suprime-Camの3倍）
CCD数: 116枚
1CCD画素数: 4272×2272
一晩で約 300 GB のデータを生成

HSC観測目的の1つ: 超新星の早期発見

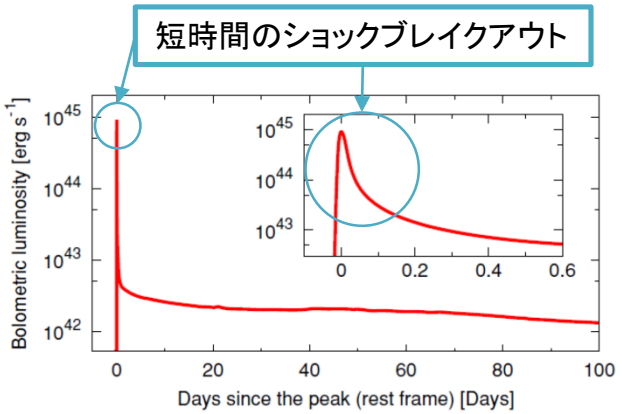
HSCによる超新星の発見の例

1晩のHSC観測で、50個の超新星候補を発見
(Gfarm/Pwrakeは未適用)



http://tpweb2.phys.konan-u.ac.jp/~tominaga/HSC-SN/ATel7565/index.html

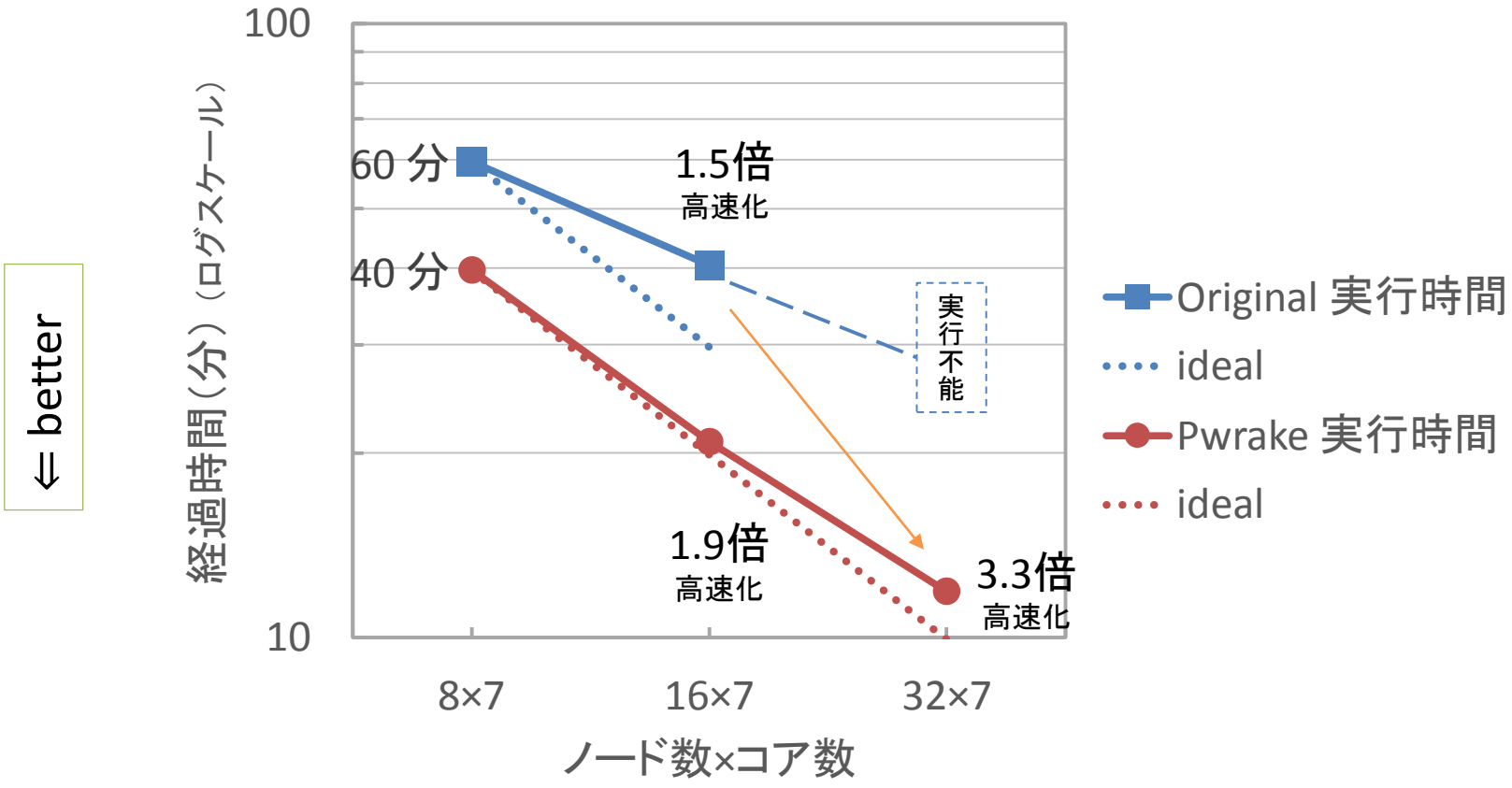
超新星の光度曲線の理論予想



Tominaga, et al. 2009, ApJL, 705, L10

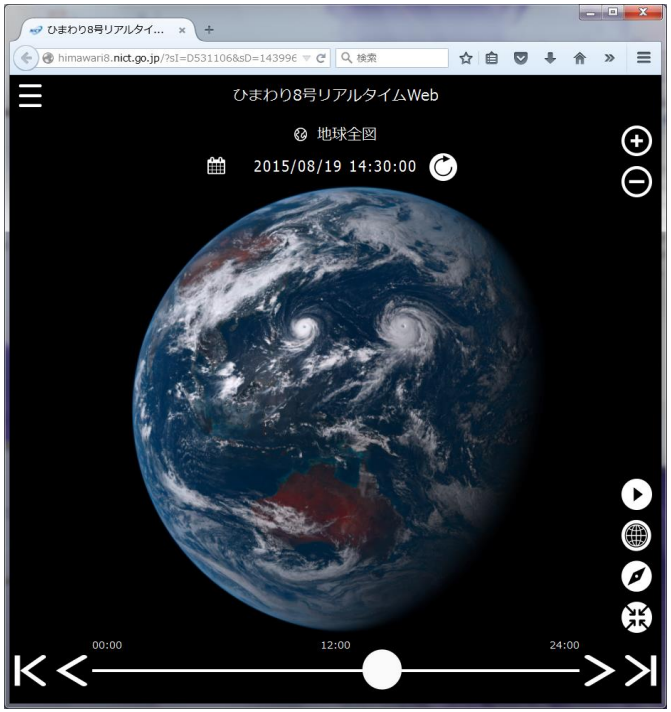
超新星の早期発見
↓
データ処理速度が重要
↓
Gfarm/Pwrake による
処理の高速化

HSCデータのFrames処理実行時間



NICT サイエンスクラウド

http://sc-web.nict.go.jp/



http://himawari8.nict.go.jp/

ひまわり8号リアルタイムWebにも
Gfarm/Pwrakeが活用されているとのこと

デモ

- https://github.com/masa16/amazon-ec2-gfarm-demo
- Amazon EC2 にクラスタ環境を構築
- Gfarmファイルシステムを設定
- PwrakeでMontageワークフローを実行