

ワークフローシステムPwrake の耐障害性

田中昌宏

筑波大学 計算科学研究センター

内容

HPC155 (SWoPP2016) およびMTAGS16で発表

▶ 背景

- Pwrakeワークフローシステム
- Gfarmファイルシステム

▶ Pwrake耐障害機能の設計と実装

▶ 評価

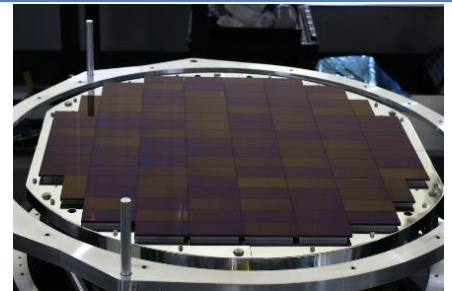
- 自動複製作成の性能への影響
- 障害発生時のワークフロー継続性

▶ まとめ

背景：科学データの並列処理

- ▶ 観測装置の進化による
データ量の増加
 - 例：すばる望遠鏡HSC
- ▶ 計算機クラスタを用いた並列処理
が必要に
- ▶ 並列処理のアプローチ
 - MPIを用いて並列プログラムを記述
 - ・ 実装コストが大きい
 - ワークフローシステム
 - ・ 従来のプログラムを並列分散実行

すばる望遠鏡 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

ワークフローシステムPwrake

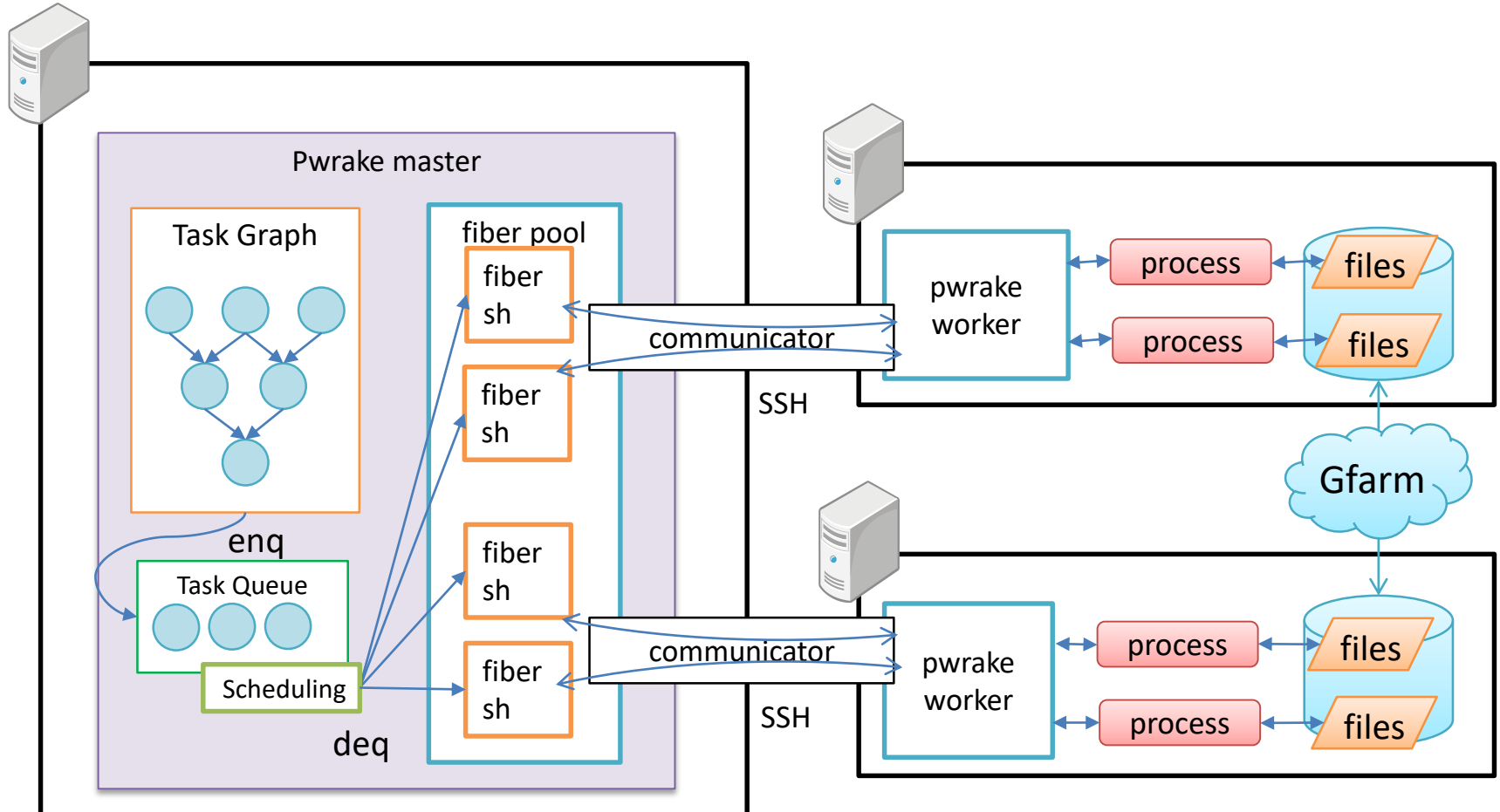
<https://github.com/masa16/Pwrake/>

- ▶ Parallel Workflow extension for RAKE
- ▶ **Rake** (Ruby版Make) がベース
 - ワークフロー記述力が高い
- ▶ **Gfarmファイルシステム**
 - ノード間のファイル共有
 - スケーラブルな並列I/O性能
- ▶ Pwrakeの実装
 - タスクの並列実行・スケジューリング
 - SSHによるリモートワーカー実行・通信

Pwrake の構成

Master node

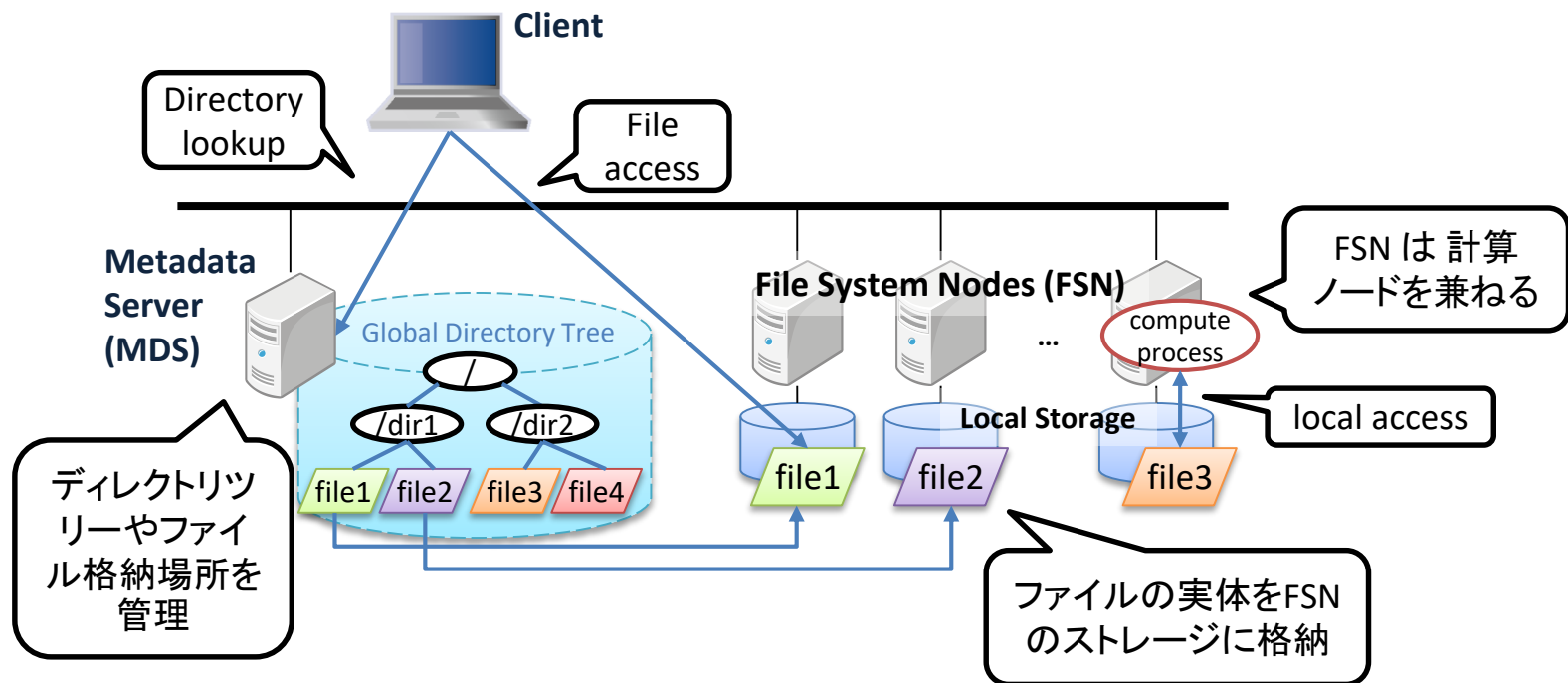
Worker nodes



Gfarmファイルシステム

<http://oss-tsukuba.org/software/gfarm>

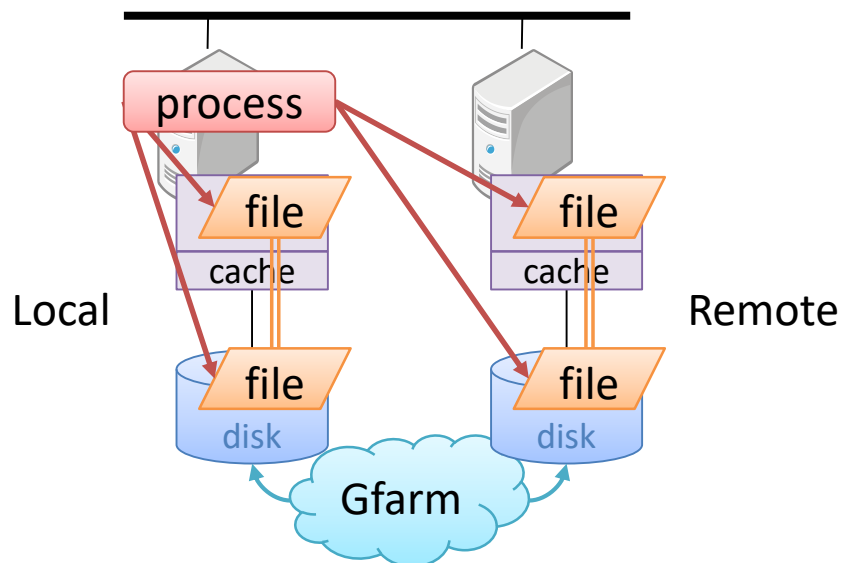
- ▶ 計算ノードのローカルストレージを束ねて構成する分散ファイルシステム



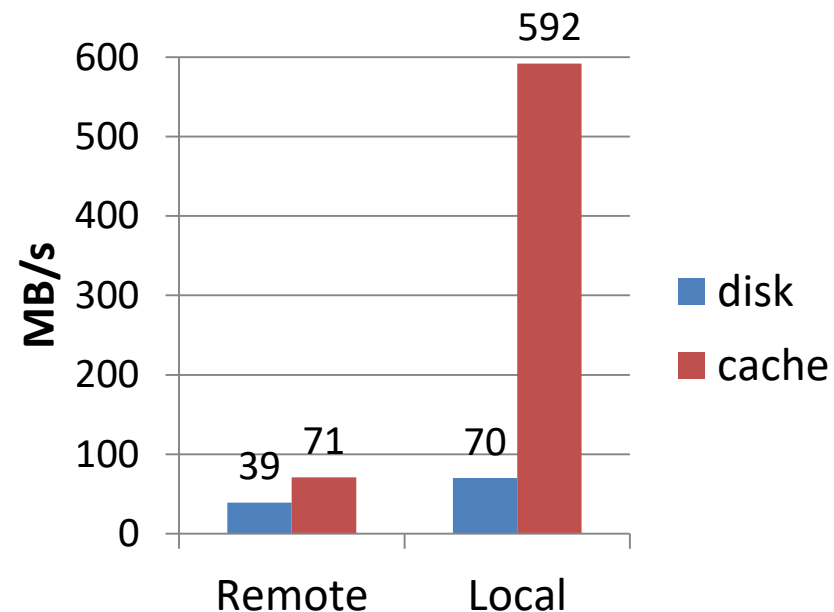
ファイルアクセスパターンとI/O性能

▶ ファイルアクセス速度

- Local > Remote
- Cached > Disk
 - Disk Cache (Buffer/Page Cache)



Read performance of Gfarm file (HDD, GbE)



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

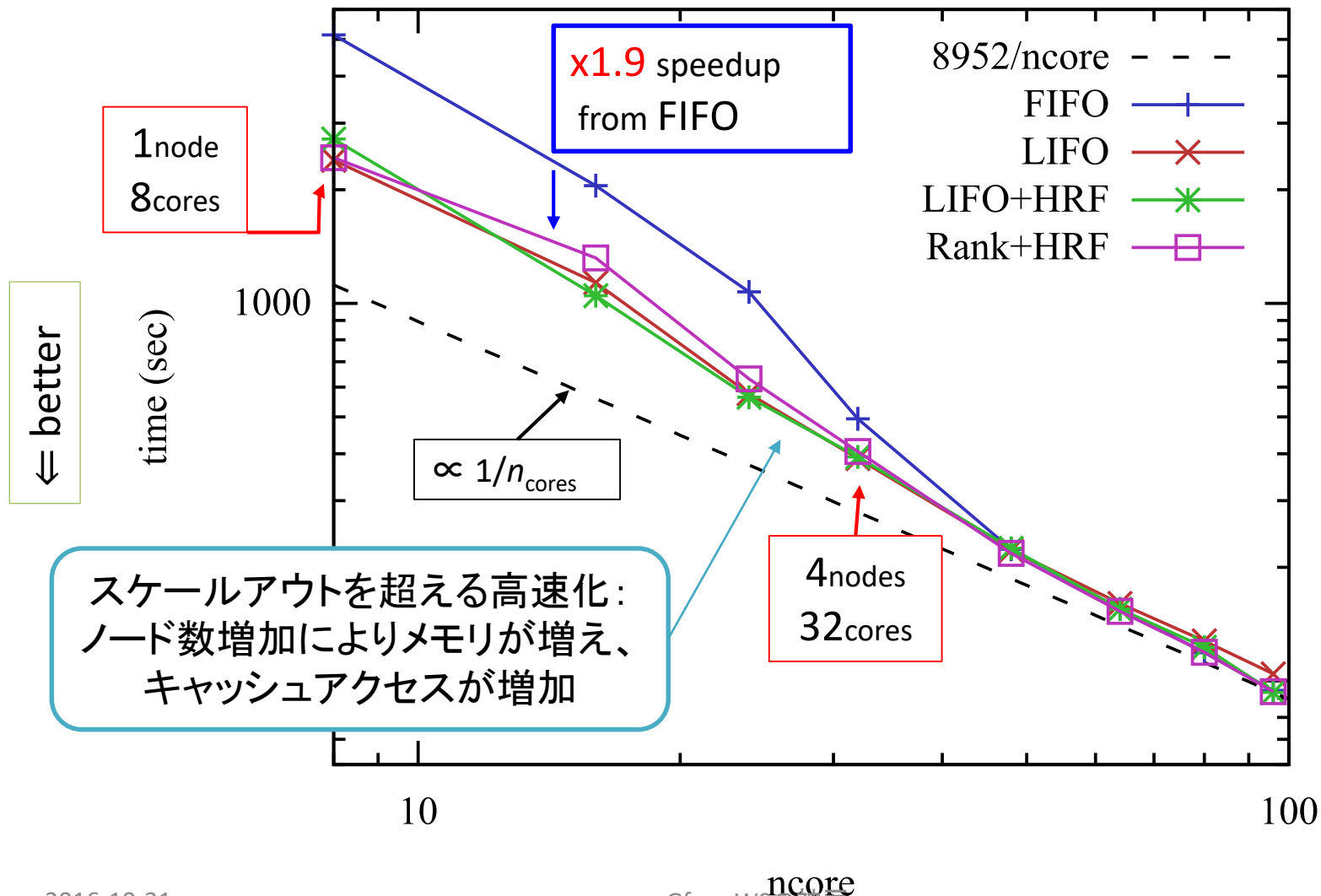
- “Workflow scheduling to minimize data movement using multi-constraint graph partitioning”
- CCGrid 2012
- タスク実行ノードを決めるスケジューリング

▶ 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

キャッシュ効果による性能向上

(1-12 nodes, Logarithmic graph)



Gfarm+Pwrakeの耐障害性

Gfarmの耐障害機能

- ▶ メタデータサーバ(MDS)
 - スレーブMDSによる冗長化
 - マスターMDS故障時に交替して運用継続
- ▶ ファイルシステムノード(FSN)
 - ノードの動的な参加・脱退
 - 故障したFSNは離脱して運用継続
 - ファイル自動複製作成
 - FSN故障時に、ファイル消失を防ぐ

Gfarmファイル自動複製作成

- ▶ ファイル書き込み後クローズ時に、自動的に他のノードに複製を作成する機能
- ▶ gfncopy コマンドを用いて設定
- ▶ ワークフロー実行時の設定
 - 出力ファイルのディレクトリに対して、複製数を2以上に設定
- ▶ ワーカーノード=FSN故障時
 - 出力ファイルの消失を防ぐ
 - 複製ファイルへのアクセスを継続

Pwrakeにおける耐障害機能の方針

▶ ワーカーノード障害

- 実行中のワークフローが止まらずに継続
- Gfarmファイル自動複製作成機能を活用

▶ マスターノード障害

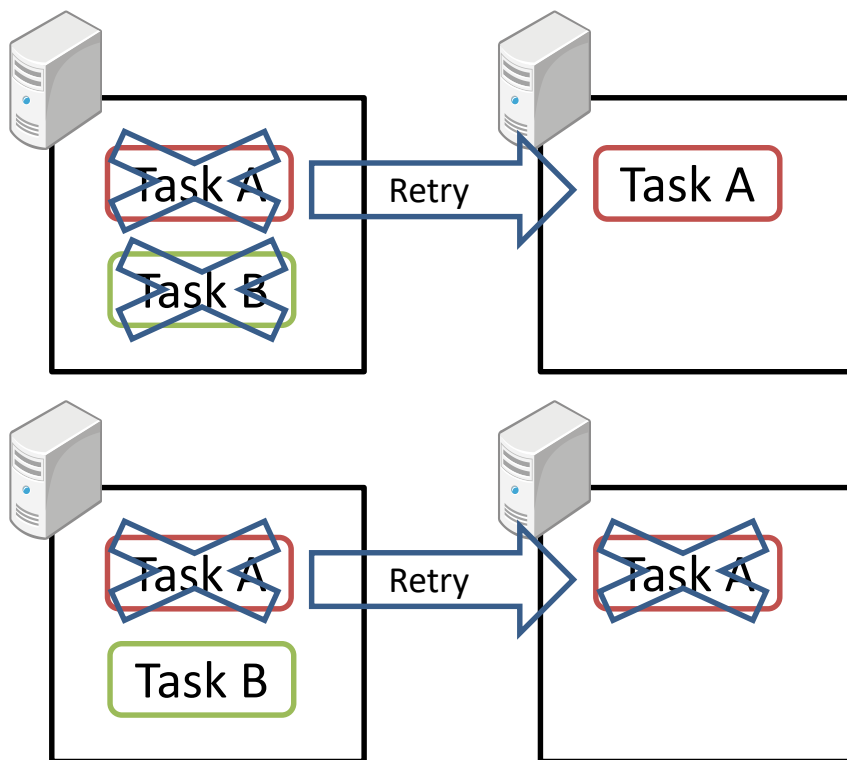
- 自動的な復旧は行わない
- 中断したワークフローの途中からの再開
 - Rakeから引き継ぐ特徴
 - 中間ファイルがチェックポイント

ワーカーノード障害検知

- ▶ Pwrakeの方針：
 - ワーカーノード障害時に脱退して続行
- ▶ 障害検知方法：
 - ワーカーノードとの通信切断
 - ハートビートのタイムアウト
 - 失敗タスクをリトライした結果

失敗タスクのリトライ

タスクが実行に失敗したとき、別のノードで再実行



同一ノードで連続失敗した場合、ノードに障害があると判定し、ノードを脱退させて継続。

同一タスクが連続失敗した場合、タスクに不具合があると判定し、後続タスクの実行は行わない。

評価実験

- 自動複製作成の性能への影響
- 障害時のワークフロー継続性

評価環境

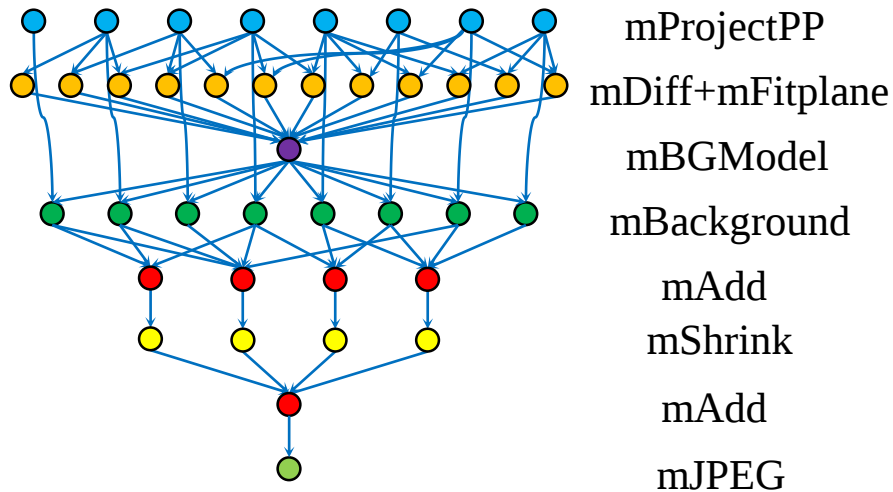
クラスタ	筑波大HPC研究室クラスタ
CPU	Intel Xeon E5620 (2.40GHz)
コア数 × CPU/ノード	4 cores × 2 cpus
主記憶容量	24GB
FSNストレージ	HDD
計算ノード数	8
ネットワーク	1Gb Ethernet
OS	CentOS 6.8
Gfarm	ver. 2.6.11
Ruby	ver. 2.3.0
Pwrake	ver. 2.1.0

Montageワークフロー

▶ 天文画像合成処理ソフト

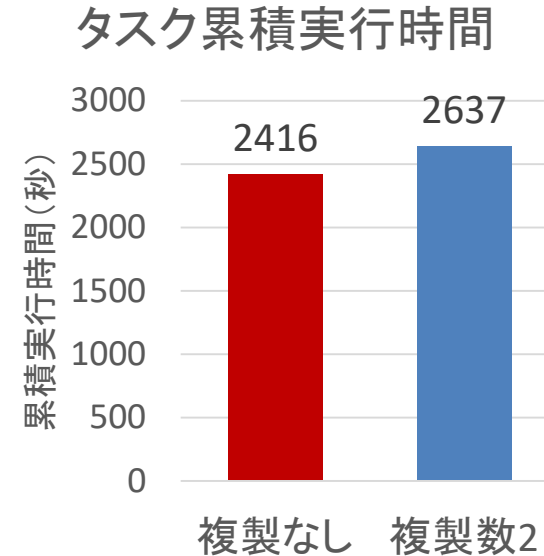
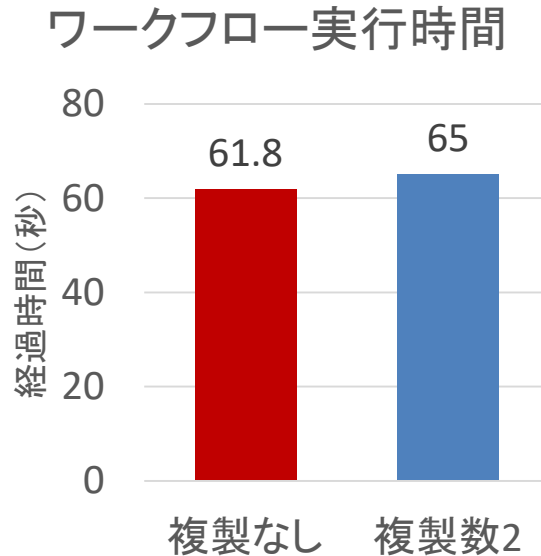
- ▶ 使用コア数: 64
(8ノード、1ノード8コア)

MontageのDAG



入力ファイル	2MASS
入力ファイル数	309
入力ファイルサイズ(合計)	639 MB
出力ファイル数	3,675
出力ファイルサイズ(合計)	6,980 MB
タスク数	2,252

複製作成によるワークフロー実行時間への影響



- ▶ 複製数を2に設定 (gfnccopy -s 2)
 - ワークフロー経過時間が 約 5% 増
 - タスク累積実行時間が 約 9% 増
 - Gbit Ethernetへの負荷が増加したと考えられる。

ワーカーノード障害の実験

▶ 疑似的な障害発生方法

(1) Pwrake のワーカープロセスをkill する.

- `kill -KILL [Pwrake worker process ID]`

(2) FSN のデーモンプロセスgfsd をkill する.

- `pkill -KILL gfsd`

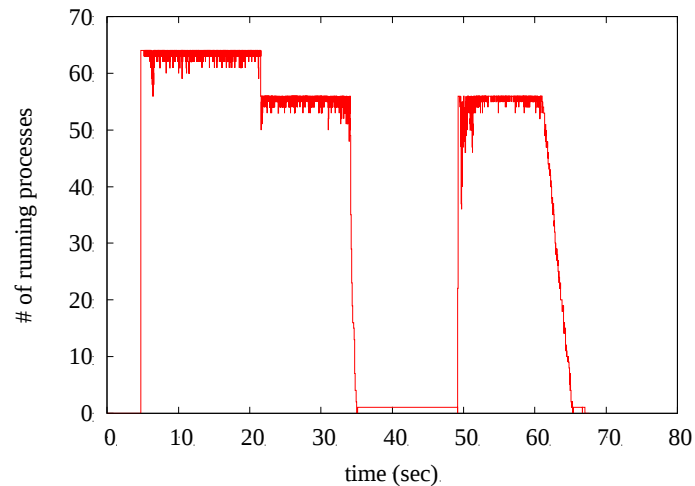
(3) ノード内のユーザ所有プロセスを全てkill する.

- `kill -KILL -1`

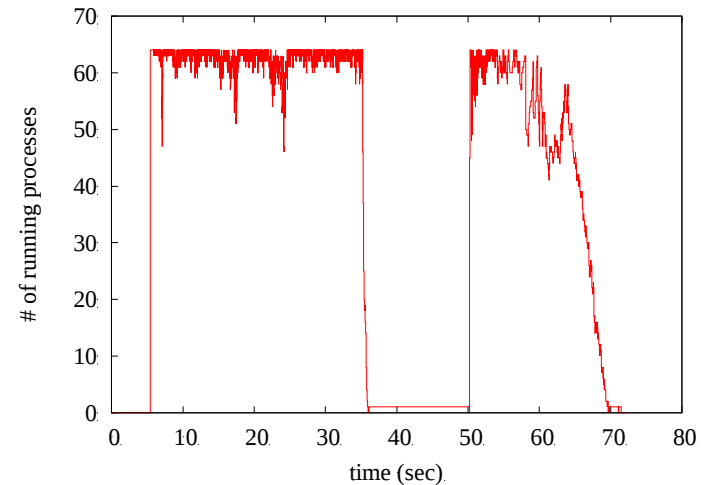
▶ 複製数2でワークフローを実行中にワーカーノードのうち1台に疑似障害発生

障害発生による プロセス数の推移

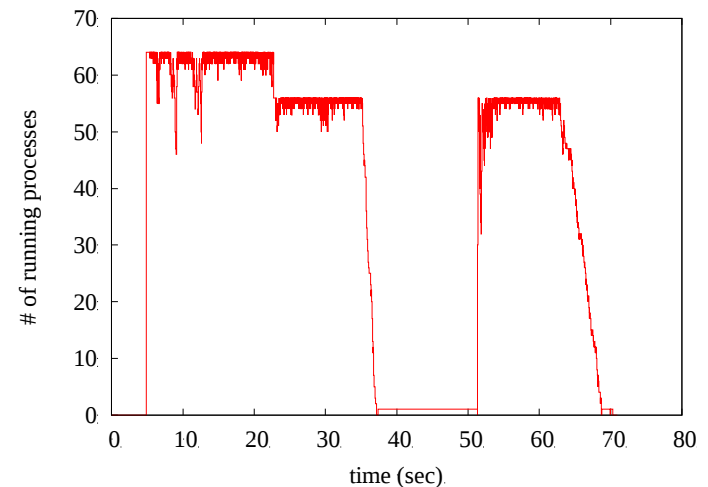
(1) Pwrake ワーカープロセスをkill



(2) gfsd をkill

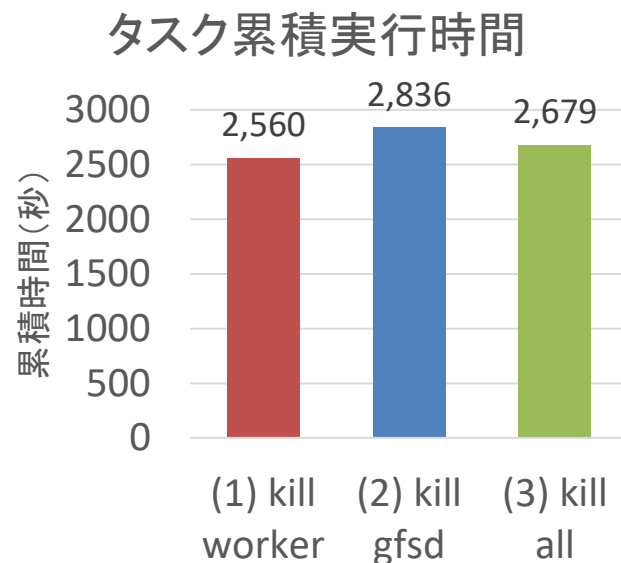
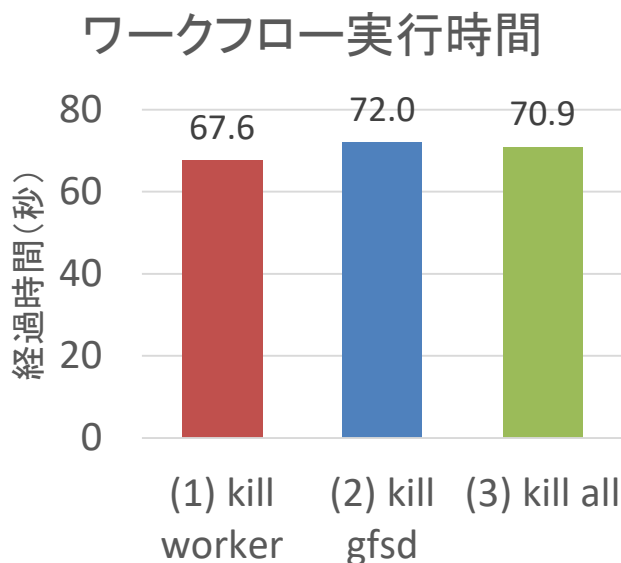


(3) ノード内のユーザ所有プロセスを全てkill



- ▶ 20秒付近で疑似障害発生:
- ▶ (1)(3): プロセス数 64 → 56 に減少して続行
- ▶ (2)(3): 障害ノードのストレージが使用不可に
- ▶ いずれも正常な結果が得られた

障害発生時のワークフロー実行時間



- ▶ (2)のgfsdのみkillのケースでは、使用コア数が減っていないにもかかわらず、(1)(3)の使用コア数が減ったケースより実行時間が長い。
- ▶ gfsd をkillしたノードではファイルアクセスが常にリモートとなり、ファイルI/Oにかかる時間が増えたことが考えられる。

関連研究

- ▶ Pegasus
 - ハードレベル: Condor DAGMan
 - ワークフローレベル:
 - ・ リトライ・チェックポイント
 - チェックポイントのオーバーヘッドが問題
- ▶ Swift
 - ハードレベル: CoG Karajan または Falkon
 - ワークフローレベル:
 - ・ リトライ、チェックポイント、重複実行
- ▶ Pwrake
 - ハードレベル: PBSなどの使用を想定
 - ワークフローレベル:
 - ・ リトライ(今回実装)、チェックポイント(Rake)、ファイル複製(Gfarm)
 - Gfarmファイルシステムのファイル自動複製作成により、ファイル消失を防ぐ

まとめ

- ▶ ワークフローシステムPwrakeの耐障害機能
 - ファイル自動複製 (Gfarm)
 - ワークフローの途中からの再実行 (Rake)
 - タスクリトライ (Pwrake)
 - 障害復帰機能は持たない
- ▶ 評価実験
 - 自動複製作成のオーバーヘッド
 - ・ ワークフローの実行時間の増加が5%程度
 - ワークフロー実行中の疑似障害発生
 - ・ ワークフローが続行し、正常な結果を確認