ワークフローシステムPwrakeにおけるI/O性能を考慮したタスクスケジューリング

田中昌宏、建部修見
筑波大学、JST/CREST

発表内容

- 研究背景・目的
- 関連研究
- Pwrakeタスクスケジューリング
  - タスク実行ノード
  - タスク実行順序
- 性能評価
  - copyfile
  - Montage
- まとめ

研究背景:MTC

- Many Task Computing (MTC)
  - Raicu et al. (MTAGS 2008)が提唱
  - 10^3 – 10^6 タスク
- 課題
  - タスク実行効率 (tasks/sec)
  - 複雑なワークフローの記述
  - 大規模データに対する I/O 性能
- MTC向けワークフローシステム:**Pwrake**（プレイク）

研究背景：ワークフローシステムPwrake

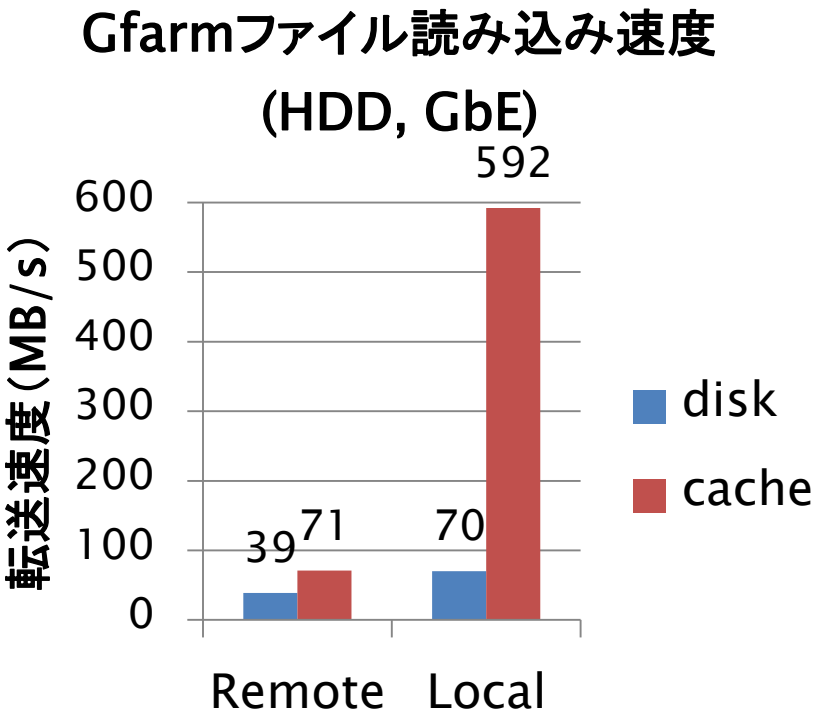
- Parallel Workflow extension for Rake
  - Rakeの並列分散機能拡張
    - Rake ＝ Ruby版 make（ビルドツール）
  - ワークフロー記述言語＝Rake

    ワークフローの高度な記述が可能
  - ノード間のファイル共有
    - 分散ファイルシステムGfarm

    計算ノードのローカルストレージを活用

研究目的：MTCタスクスケジューリング

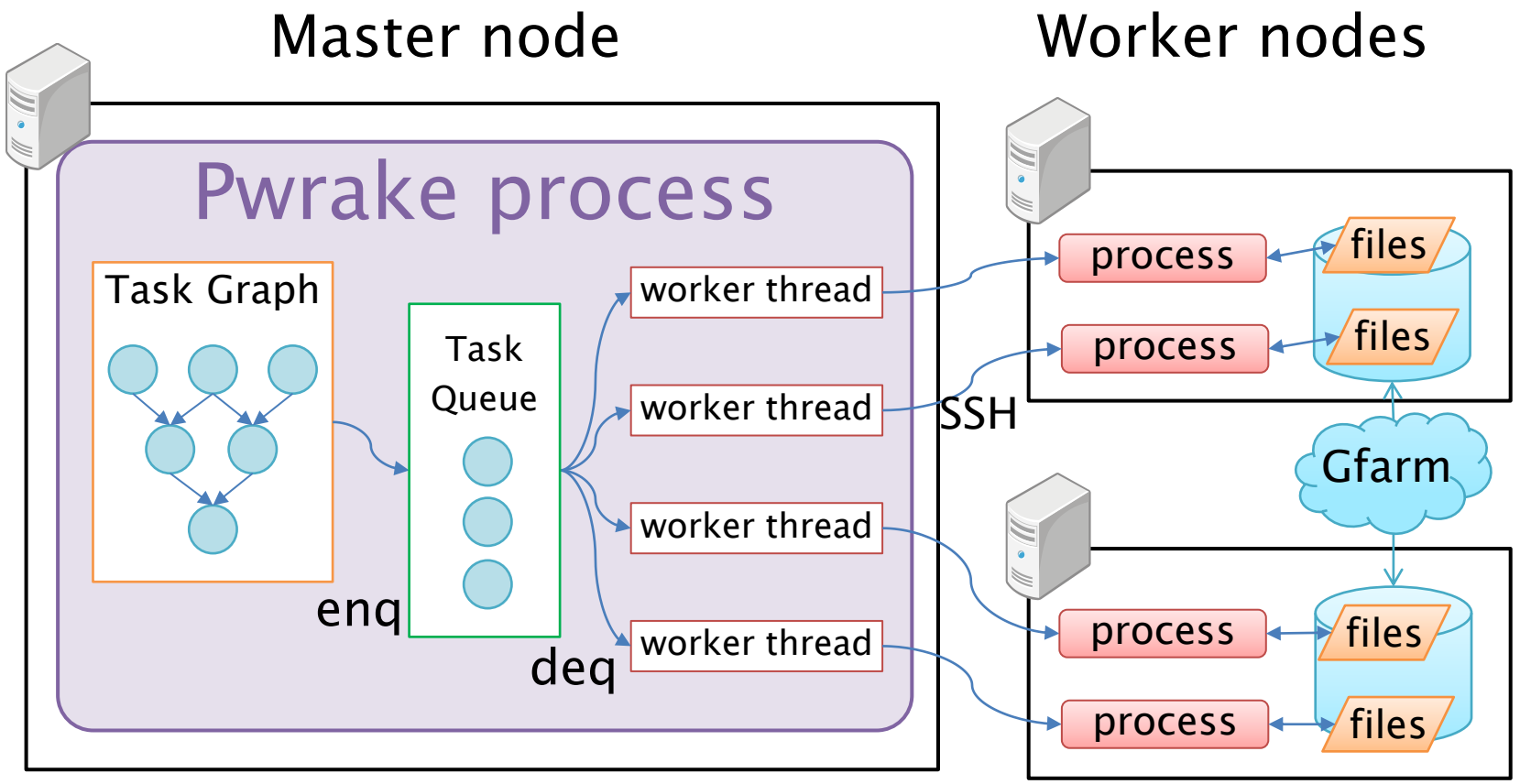
- 大量の均質なタスク
  - 個々のタスクの処理時間の事前把握は困難
  - 動的なスケジューリング
- データインテンシブワークフロー
  - ローカルアクセス
  - バッファ（ページ）キャッシュ
  - I/O性能を引き出すタスクスケジューリング



関連研究

- Swift + Falkon + data diffusion
  - Swift (Wilde et al. 2011)
    - 独自のワークフロー言語を備えたシステム
  - Falkon (Raicu et al. 2007)
    - 高速なタスク起動 (1500 tasks/sec)
  - data diffusion (Raicu et al. 2008)
    - ステージングファイルの管理＋タスクスケジューリング
- GXP make (Taura et al. 2013)
  - GNU make に基づくワークフローシステム
  - GNU make が発行したコマンド列を、mkshによりトラップ
    - 実行順序が GNU make に依存
    - 入力ファイル名の取得が困難
      - I/O履歴を元に予測 （堀内,田浦 HPC135, 2012)

Pwrakeの構成

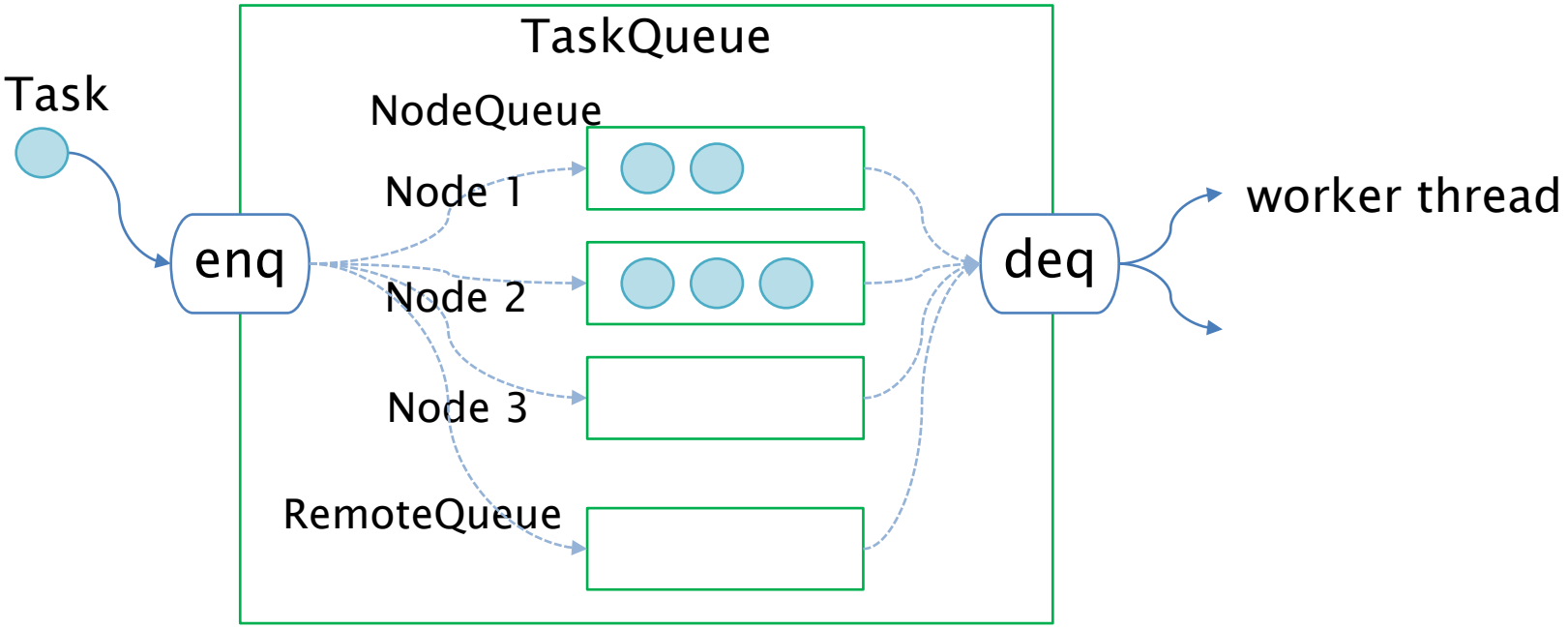

Master node
Worker nodes
Pwrake process
Task Graph
Task Queue
enq
deq
worker thread
worker thread
worker thread
worker thread
SSH
process
process
process
process
files
files
files
files
Gfarm

Pwrake タスクスケジューリング

1. 実行ノード決定方法
2. 実行順序決定方法

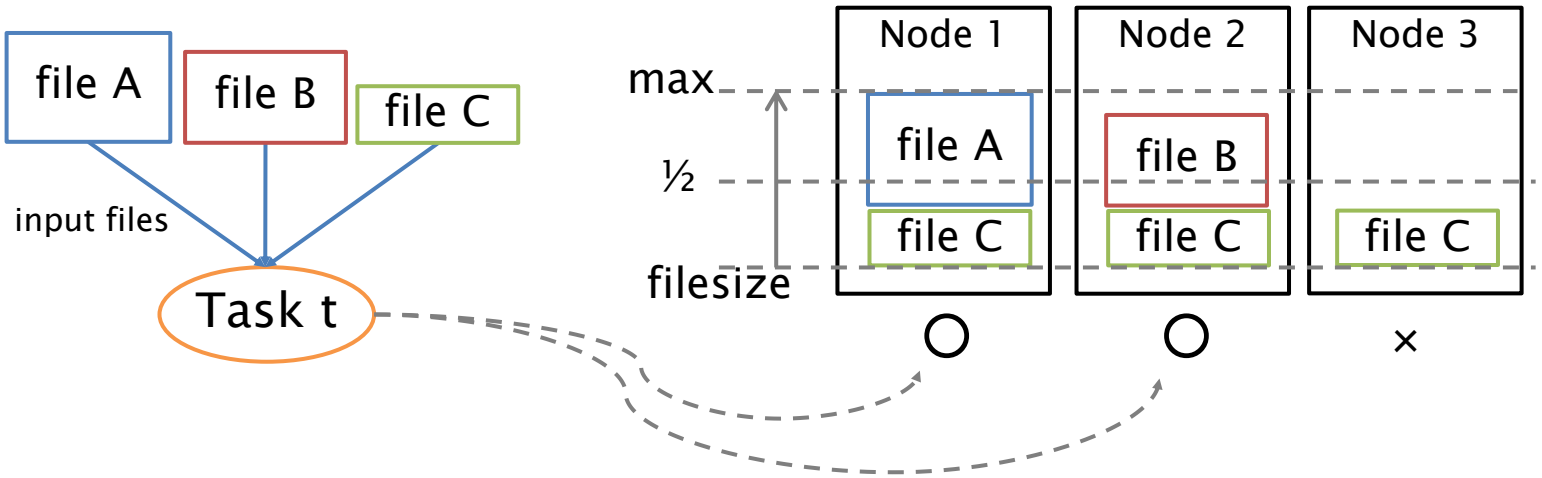
1. 実行ノード決定:
ローカリティを考慮したタスクキュー

- ノード毎にノードキューを持つ構造
- enq: 「候補ノード」のノードキューにタスクを投入
- deq: ワーカー担当ノードのノードキューから取り出し
- スチールによる負荷バランス

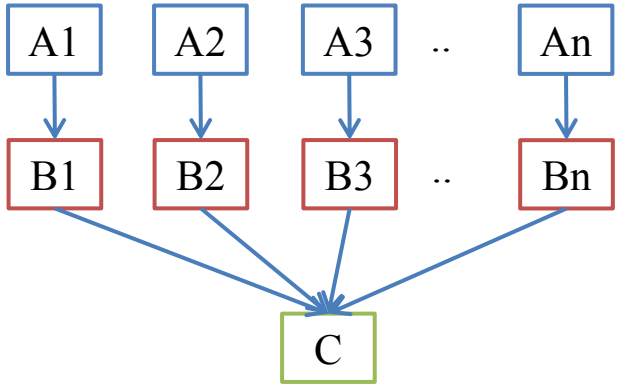


1. 実行ノード決定:
候補ノード決定方法

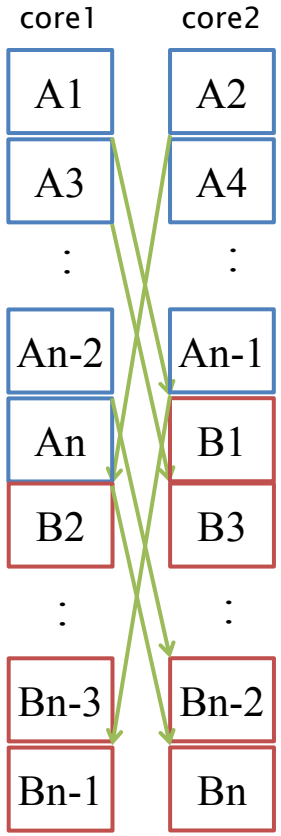
- 1タスクにつき、入力ファイルの格納ノードは複数ありうる
  - 複数の入力ファイルを別ノードに格納
  - ファイルの複製
- ノード毎に入力ファイルサイズを合計
- 最大の半分以上のファイルサイズを持つノードを候補ノードとする



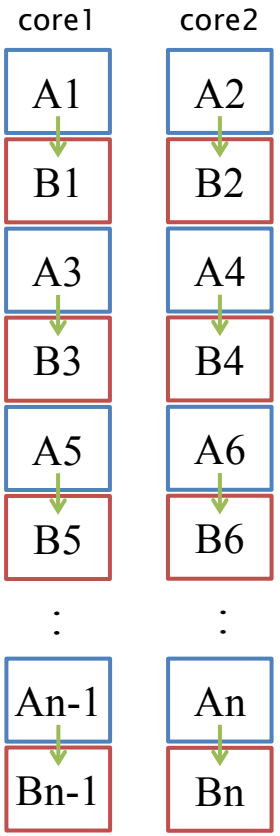
2. 実行順序決定:
FIFO, LIFOとキャッシュ活用



- Directed Acyclic Graph (DAG)
- 2コア1ノードで実行
- A→Bのタスク間隔
  - FIFO:タスク n/2個
  - LIFO:タスク 0個
- LIFOの方が、キャッシュに乗る確率が高い



FIFO

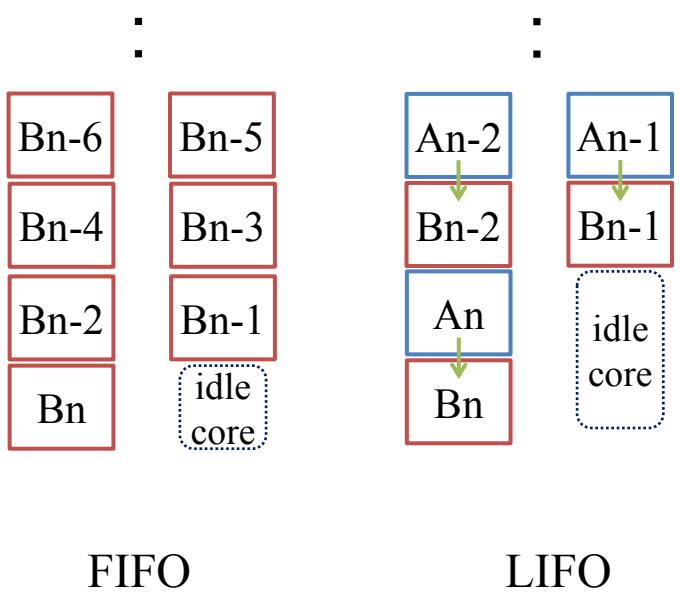


LIFO

2. 実行順序決定:
末尾タスク問題

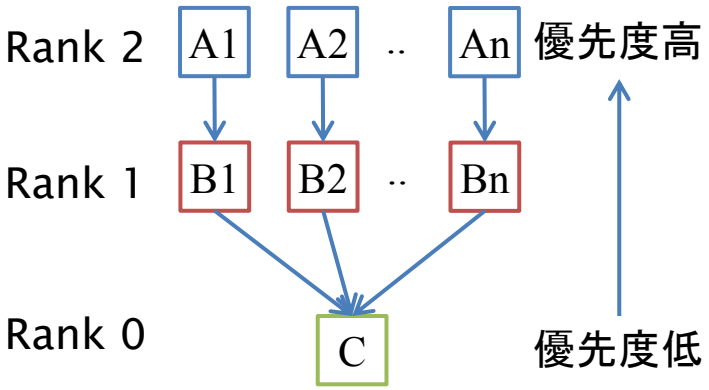
Trailing Task Problem, (Armstrong et al. MTAGS 2010)

- 最後のタスクでアイドルコアが発生
  - FIFO: 最大 B
  - LIFO: 最大 A+B
- FIFOの方が、アイドルコアが少ない



2. 実行順序決定:
HRFによる末尾タスク問題の軽減

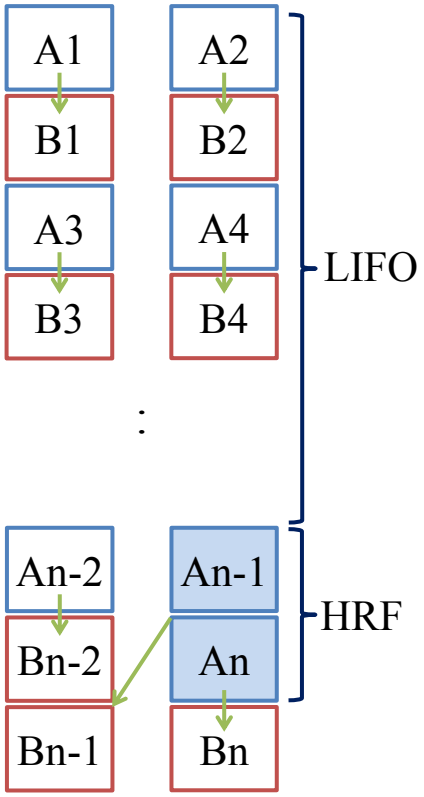
- タスクの「ランク」を定義
  - 最終タスクから数えたDAG中の位置
- Highest Rank First (HRF)
  - 高いランクのタスクを先に実行
  - 末尾タスク問題の軽減に有効
  - 欠点:キャッシュ活用には不利
- 先行研究における例
  - HEFT (Topcuoglu et al. 2002)
    - 計算コスト、通信コストに基づいてランクを決定
    - 高いランクのタスクからCPUに割り振る
- 問題点:
  - LIFOでは、低いランクを先に実行



2. 実行順序決定:
提案手法 (1) LIFO+HRF

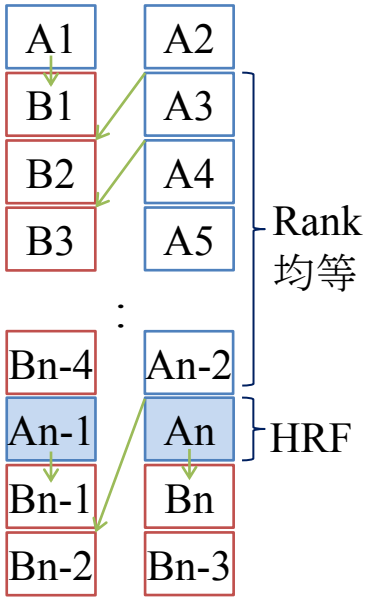
- Nc = コア数
- Nr = キュー中の最高ランクのタスク数

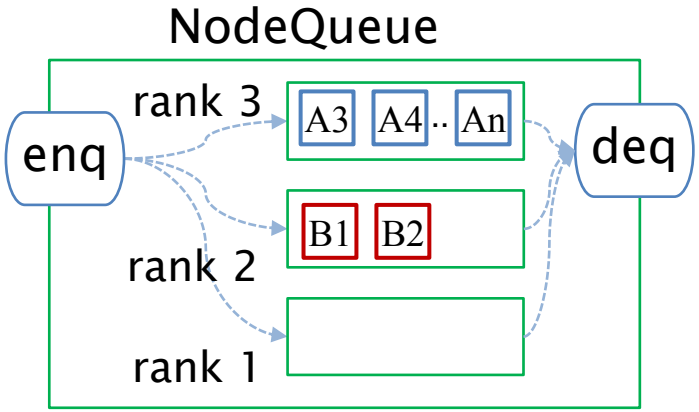
- 手法:
  - Nr > Nc のとき LIFO
  - Nr ≦ Nc のとき HRF



2. 実行順序決定:
提案手法 (2) Rank Overlap+HRF

- 目的:計算とI/Oのオーバーラップ
  - タスクA:　計算インテンシブ
  - タスクB:　I/Oインテンシブ
- 手法:
  - Nr ＞ Nc のとき:
    - デキューするランクを確率的に選択
    - 選択の重み:
      - 平均実行時間の逆数
        ＝1秒あたりの起動回数
    - ランクからLIFOでデキュー
  - Nr ≦ Nc のとき: HRF





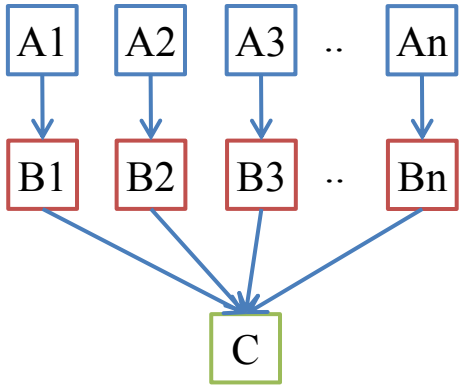
性能評価

評価環境

クラスタ	InTrigger東北大学拠点
CPU	Intel Xeon E5410 2.33GHz
主記憶容量	32GiB
コア数/ノード	8
計算ノード数（最大）	12
ネットワーク	1Gb Ethernet
OS	Debian 5.0.4
Gfarm	ver. 2.5.8.4
Ruby	ver. 2.1.0
Pwrake	ver. 0.9.9

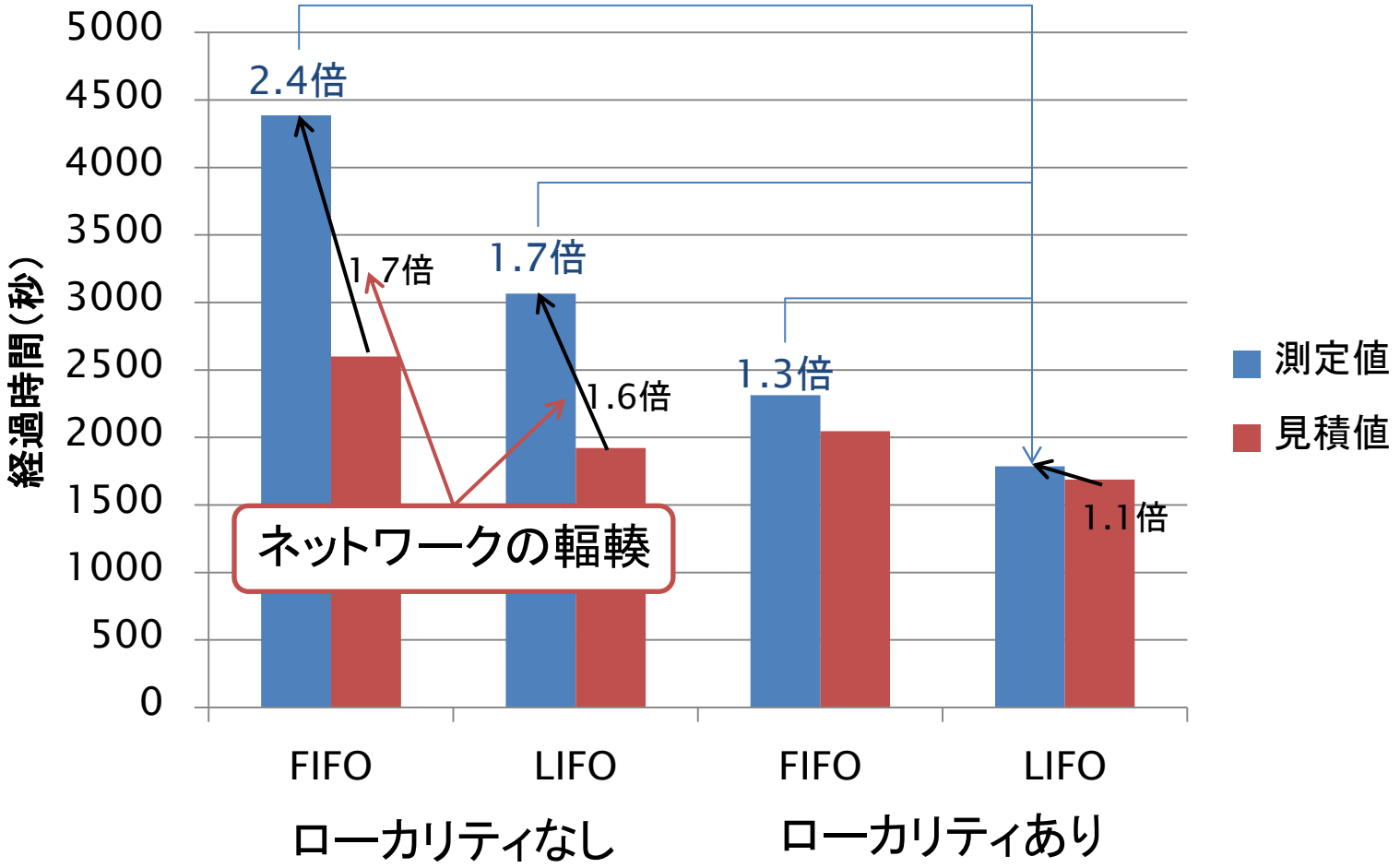
I/Oのみを行うワークフロー

- ワークフロー（右のDAG）
  - タスクA,B：copyfile（自作プログラム）
    - 入力ファイルを主記憶に読んだ後、出力ファイルに書き出す
- 測定条件
  - 10ノード、1コア/ノード
- スケジューリング
  - ローカリティあり、なし
  - FIFO、LIFO
- 単体I/O性能から見積もった実行時間との比較
  - LIFOのタスクB入力のみがcache、他はdisk
  - ローカル・リモートの割合は、測定時の実績



	入力ファイル	主記憶
数	n=100	
1個サイズ	3 GiB	< 32 GiB
全サイズ	300 GiB	> 32 GiB

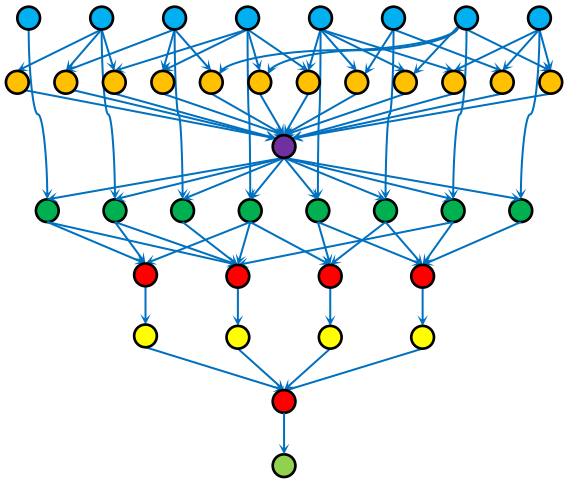
copyfile ワークフロー経過時間



Montageワークフロー

- 天文画像合成処理ソフト
- 使用コア数： 96
  （12ノード、1ノード8並列）

MontageのDAG

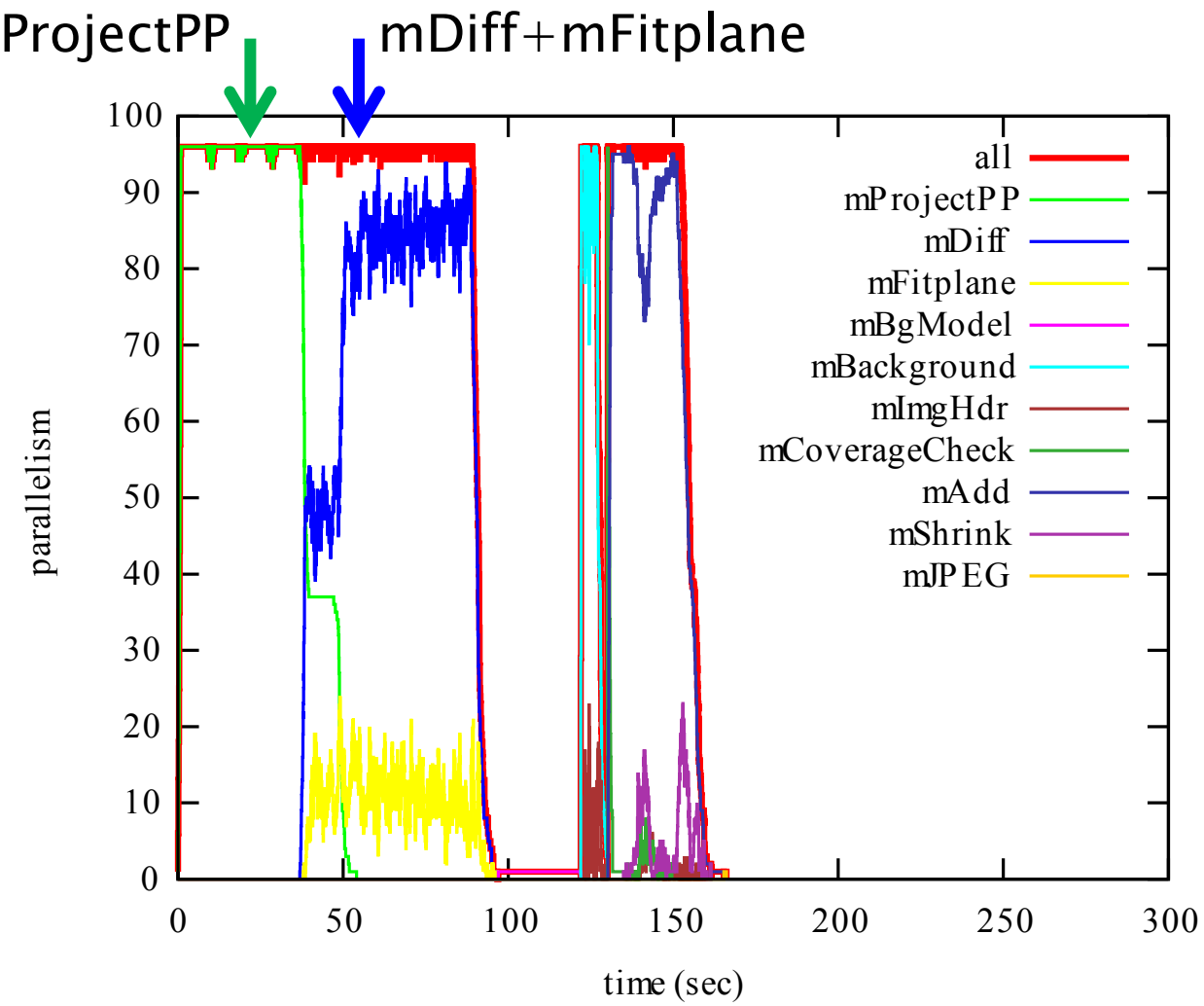


mProjectPP
mDiff+mFitplane
mBGModel
mBackground
mAdd
mShrink
mAdd
mJPEG

入力ファイル	SDSS DR7
入力ファイル数	421
入力ファイルサイズ（1個）	2.52 MB
入力ファイルサイズ（合計）	1061 MB
中間画像ファイル数	4720
中間画像ファイルサイズ（合計）	63.5 GB
タスク数	2707
$\frac{\text{mProjectPPの出力サイズ}}{\text{12ノード}}$	1.6 GB

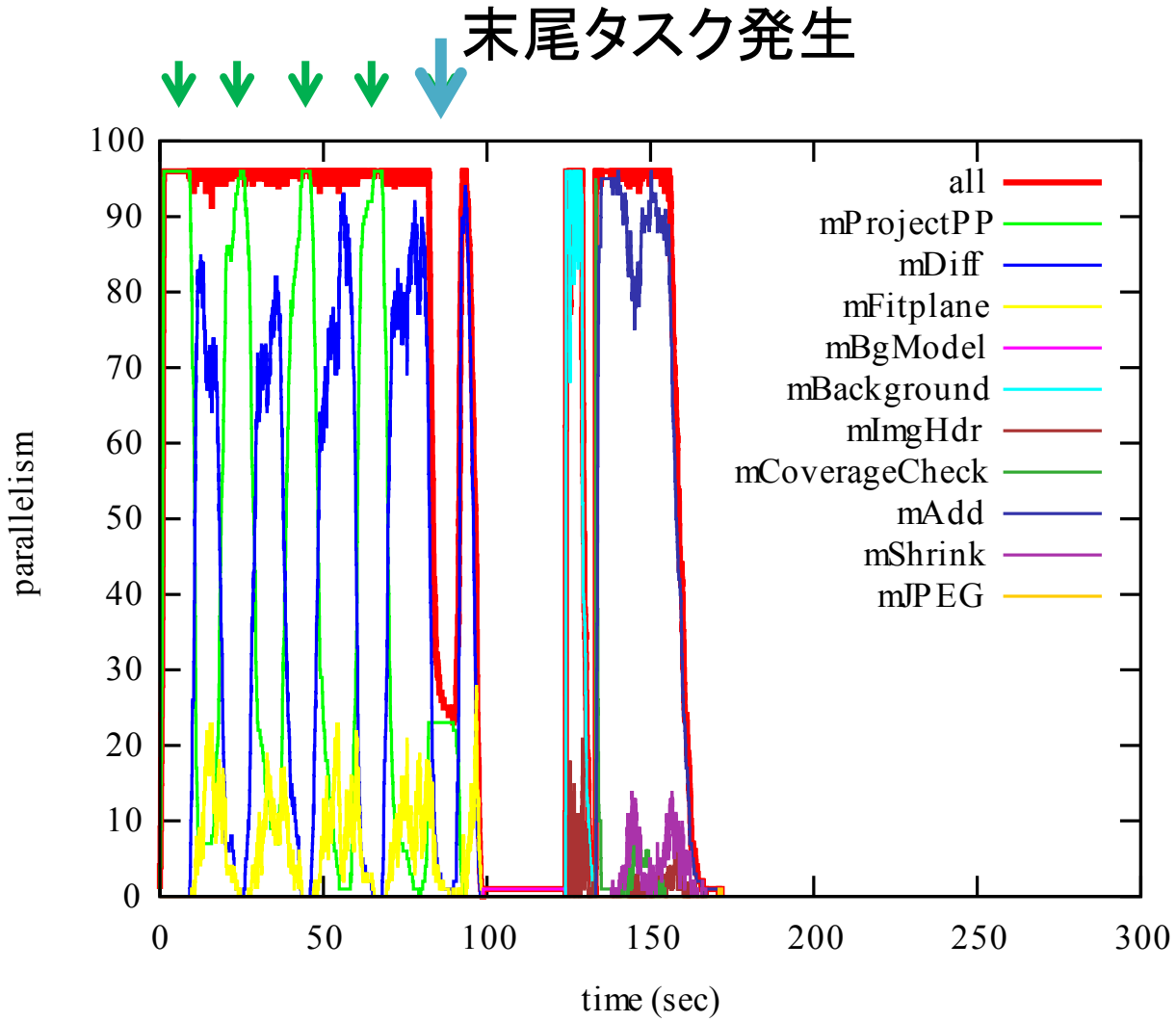
実行プロセス数の推移 (FIFO)

（ローカリティ考慮）



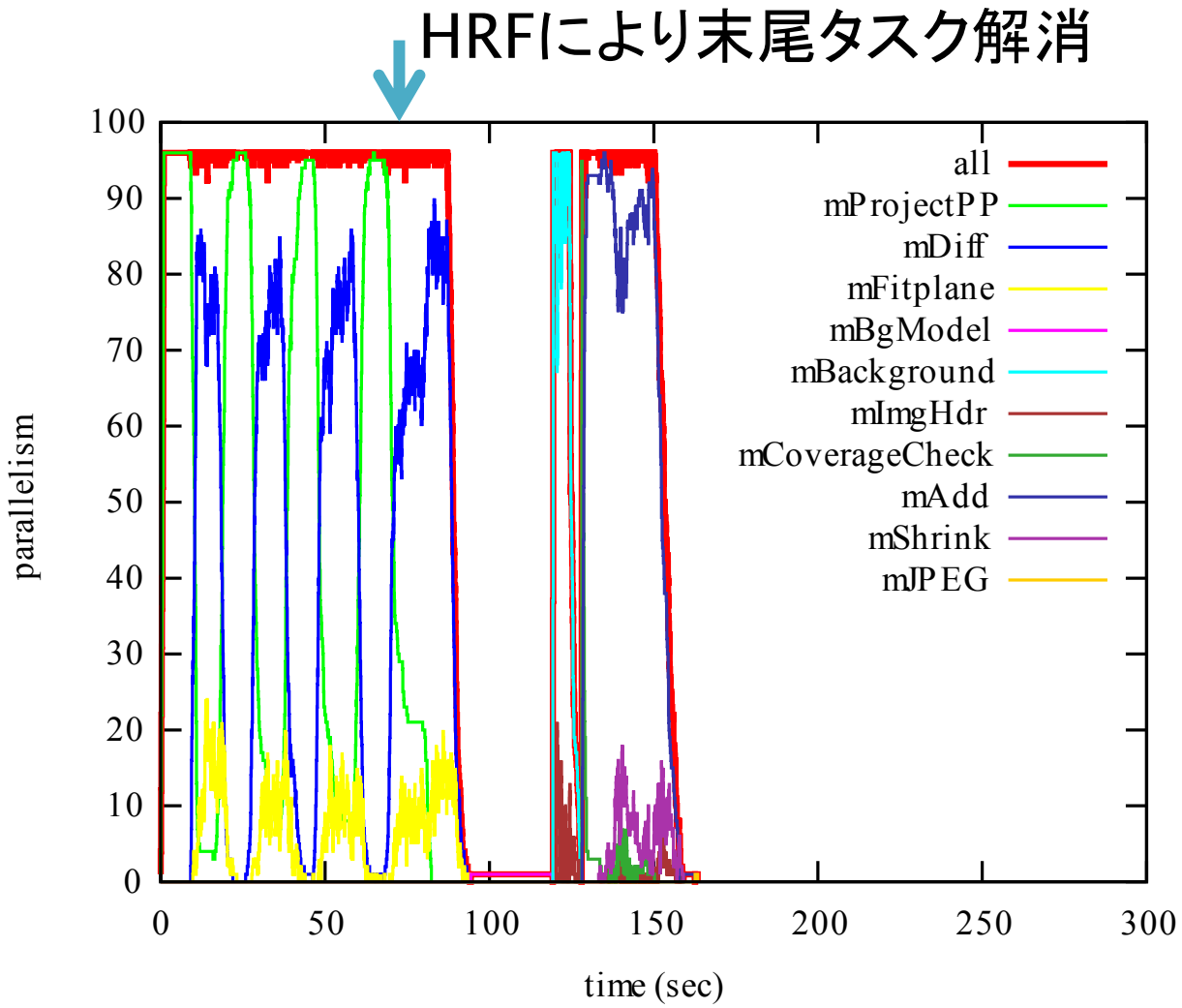
実行プロセス数の推移 (LIFO)

（ローカリティ考慮）



実行プロセス数の推移 (LIFO+HRF)

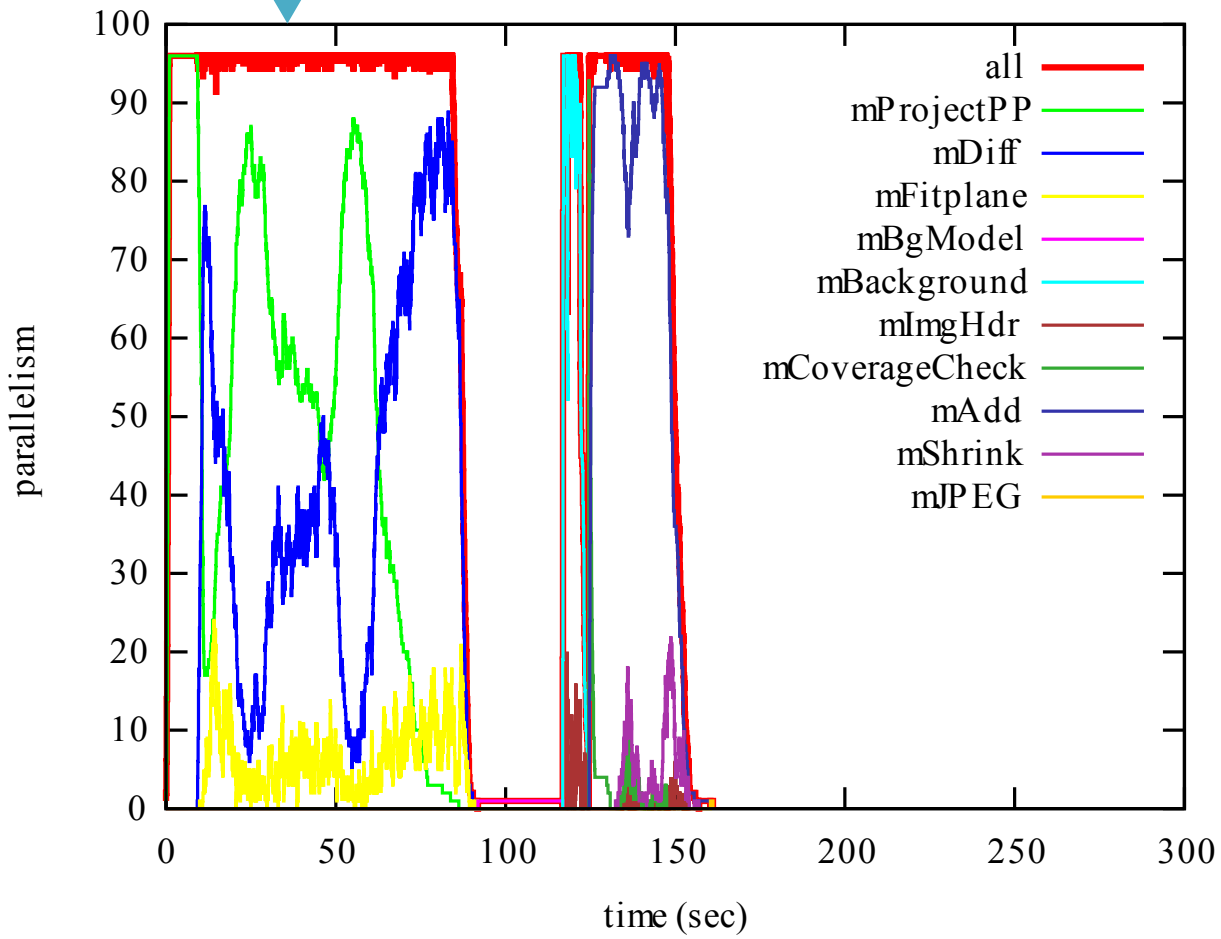
（ローカリティ考慮）



実行プロセス数の推移 (Rank Overlap)

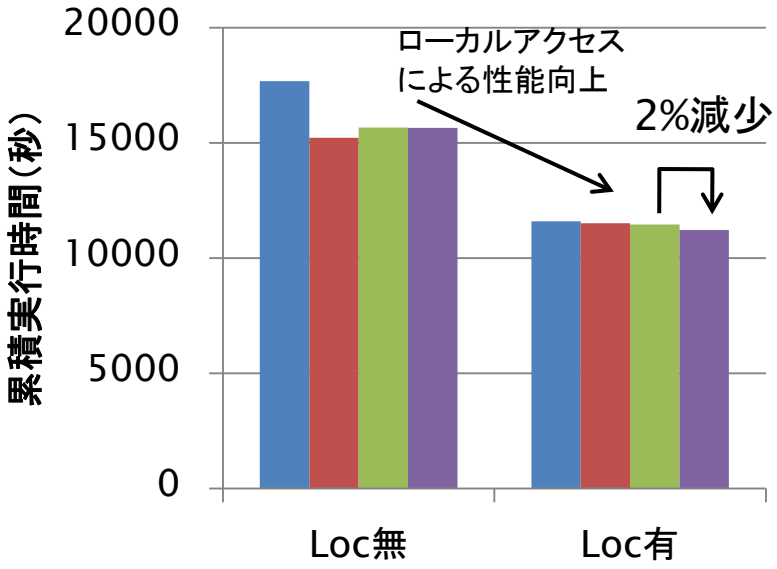
（ローカリティ考慮）

mProjectPPとmDiffのオーバラップ

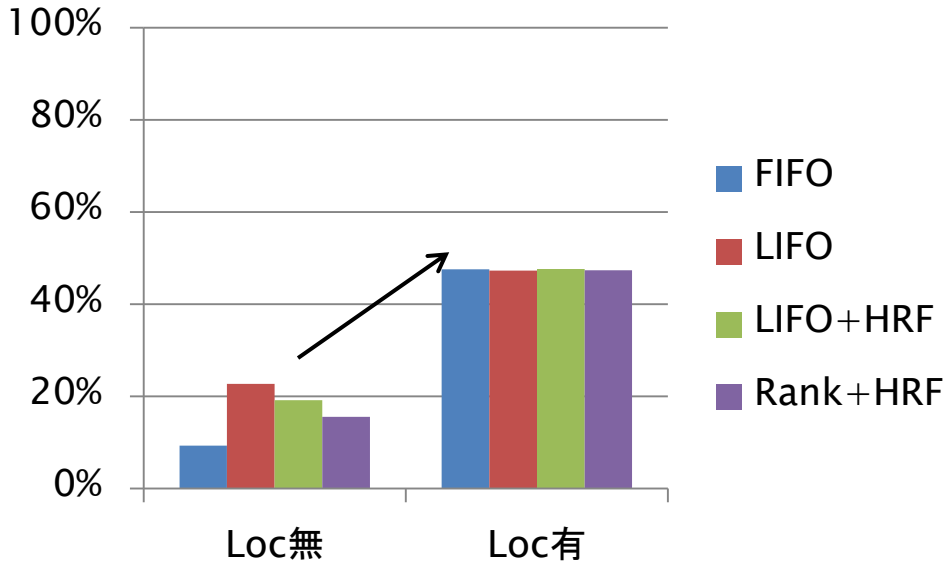


Montage タスクの累積実行時間

- タスクの累積実行時間 ＝ 個々のタスクの実行時間の総和（逐次タスクは除外）
- ローカリティスケジューリングにより、約24–34% 減少
- LIFO+HRFと比較して、Rank Overlapは 約2% 減少



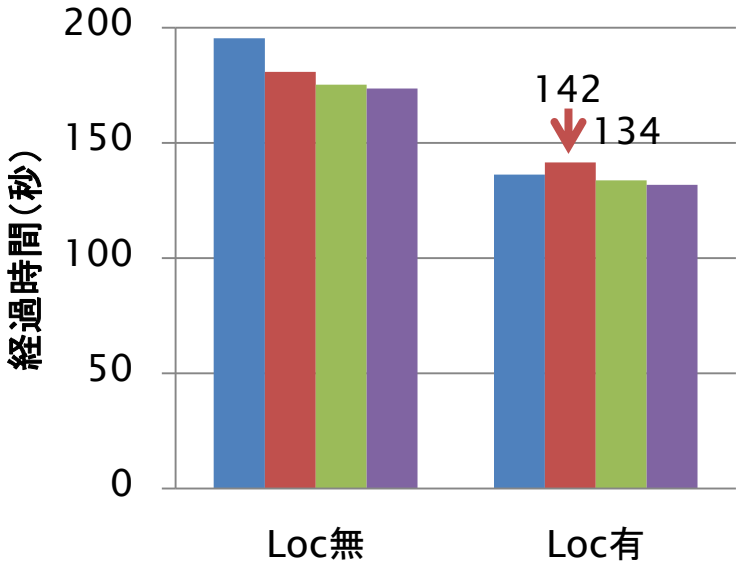
タスクの累積実行時間



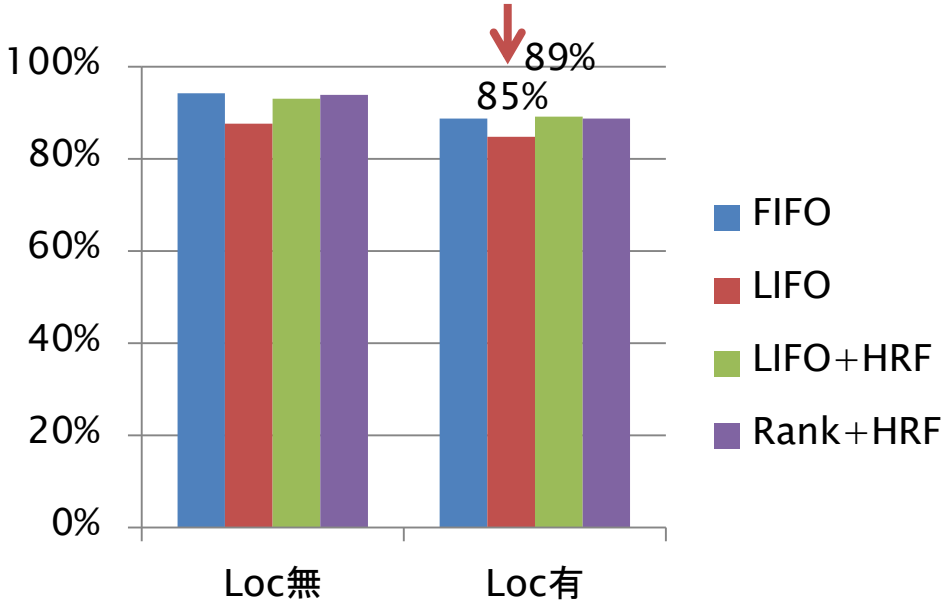
ローカルアクセス率

Montage ワークフロー経過時間

- （逐次タスクは除外）
- コア使用率 ＝ 累積実行時間 ÷ コア数 ÷ 経過時間
- LIFOでのコア使用率低下 → 経過時間が 約5% 増加



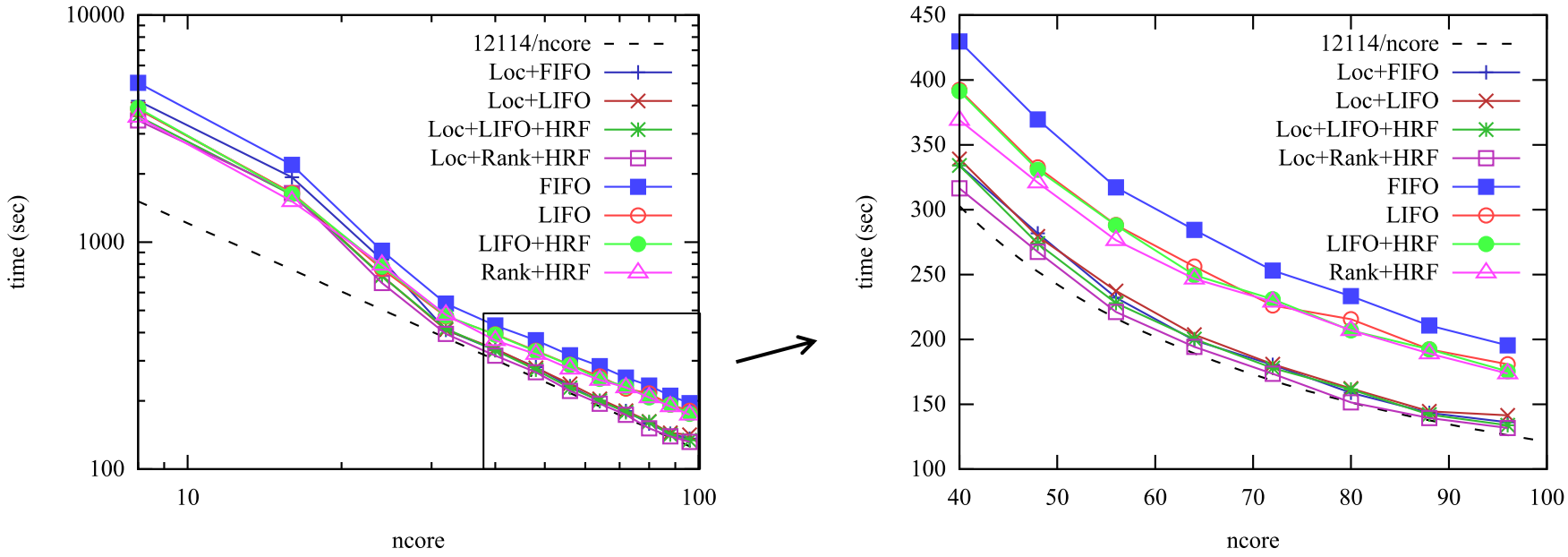
ワークフローの経過時間



コア使用率

強スケーリングの測定

- ノード数にかかわらず、Rank Overlap+HRF の経過時間が最短
- ノード数＝1～3 のとき、1ノードで処理するデータ量が多い
  → キャッシュに乗らず、処理時間が増加 → FIFOの性能が最低



まとめと今後の課題

- 研究目的:
  - MTC向けワークフローシステム Pwrake のための、ローカリティおよびキャッシュを考慮したタスクスケジューリング
- 提案手法:
  - 実行ノード
    - ファイルサイズに基づく候補ノード決定
  - 実行順序
    - LIFO+HRF切り替え: キャッシュを活用しつつ末尾タスクを軽減
    - Rank overlap: 計算とI/Oのオーバーラップ
- 性能評価:
  - copyfileワークフローにおいて、ローカリティ・キャッシュ活用による性能向上
  - Montageワークフローにおいて、HRFおよびRank overlapが有効
- 今後の課題: 他のワークフローへの適用、大規模なシステムでの評価