グラフ分割による広域分散並列ワークフローの効率的な実行

田中 昌宏，建部 修見 (筑波大学)

発表内容

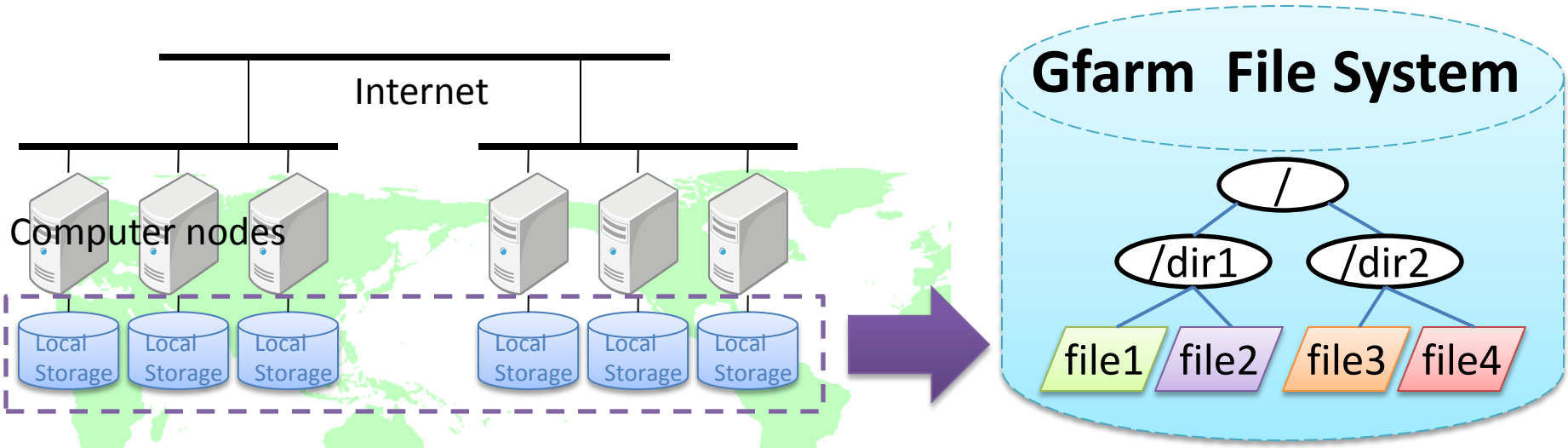
- 背景
  - 複数拠点の計算機資源を用いた広域分散ワークフローを行う際に、拠点間のデータアクセスの削減のためのタスク配置が課題
- 提案手法
  - 各拠点への最適なタスク配置を、ワークフローのグラフから自動的に求めるために、グラフ分割を利用したタスク配置手法を提案
- 性能測定
  - 対象： 天文画像合成ワークフロー
  - プラットフォーム： InTrigger

［背景］eサイエンス

- 高速ネットワークで結ばれた、広域情報処理基盤の活用により可能となるサイエンス
- 適用分野
  - 生命科学、高エネルギー物理学、地球科学、天文学、etc
- 基盤技術
  - 複数拠点の計算機資源の連携技術
  - 今後、eサイエンスで扱うデータのスケールが大きくなるにつれて、広域分散データ解析が必要となることが予想され、その基盤技術が必要

[背景] Gfarmファイルシステム

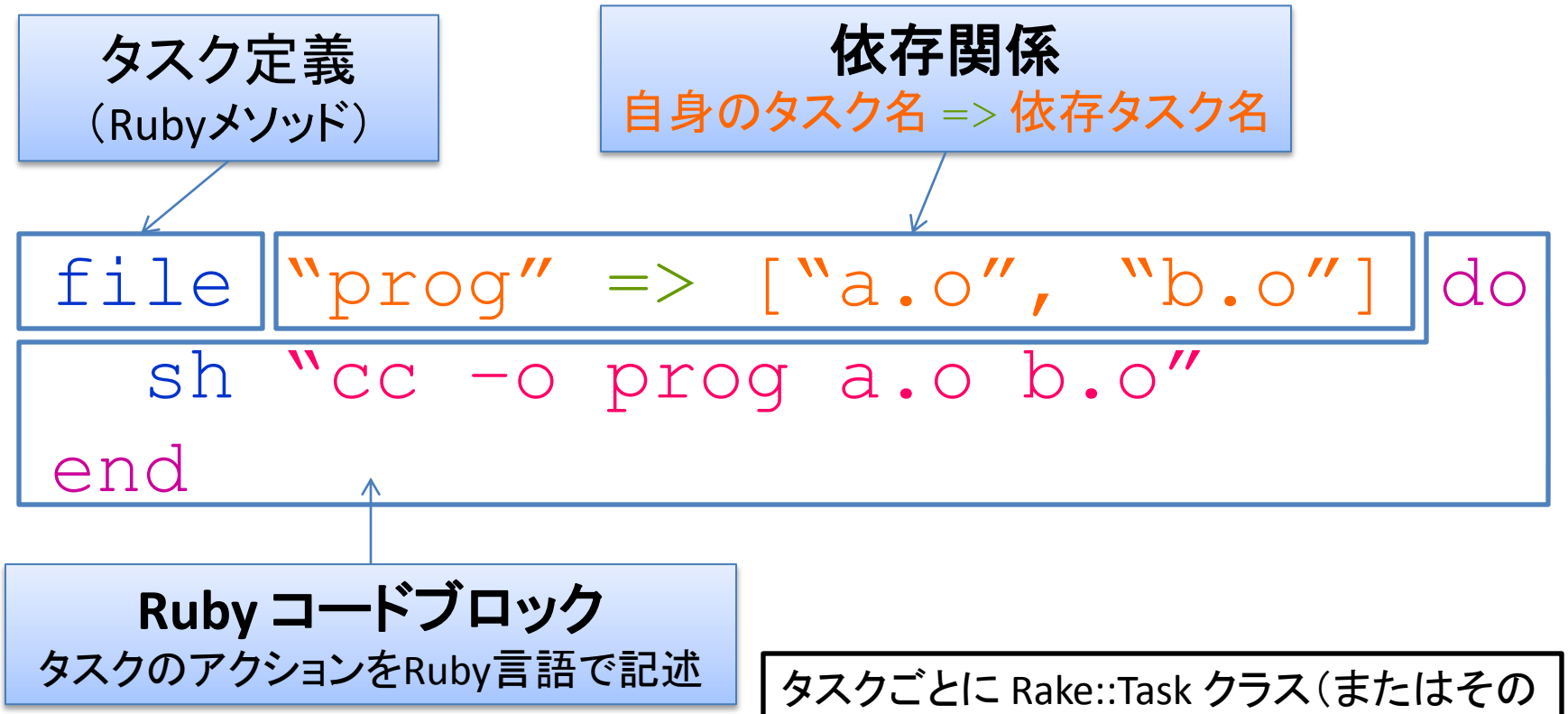
- eサイエンス基盤　広域分散ファイルシステム
- 地理的に分散した、複数拠点のストレージを統合
- 用途:
  - 広域ファイル共有
  - 大規模並列データ処理



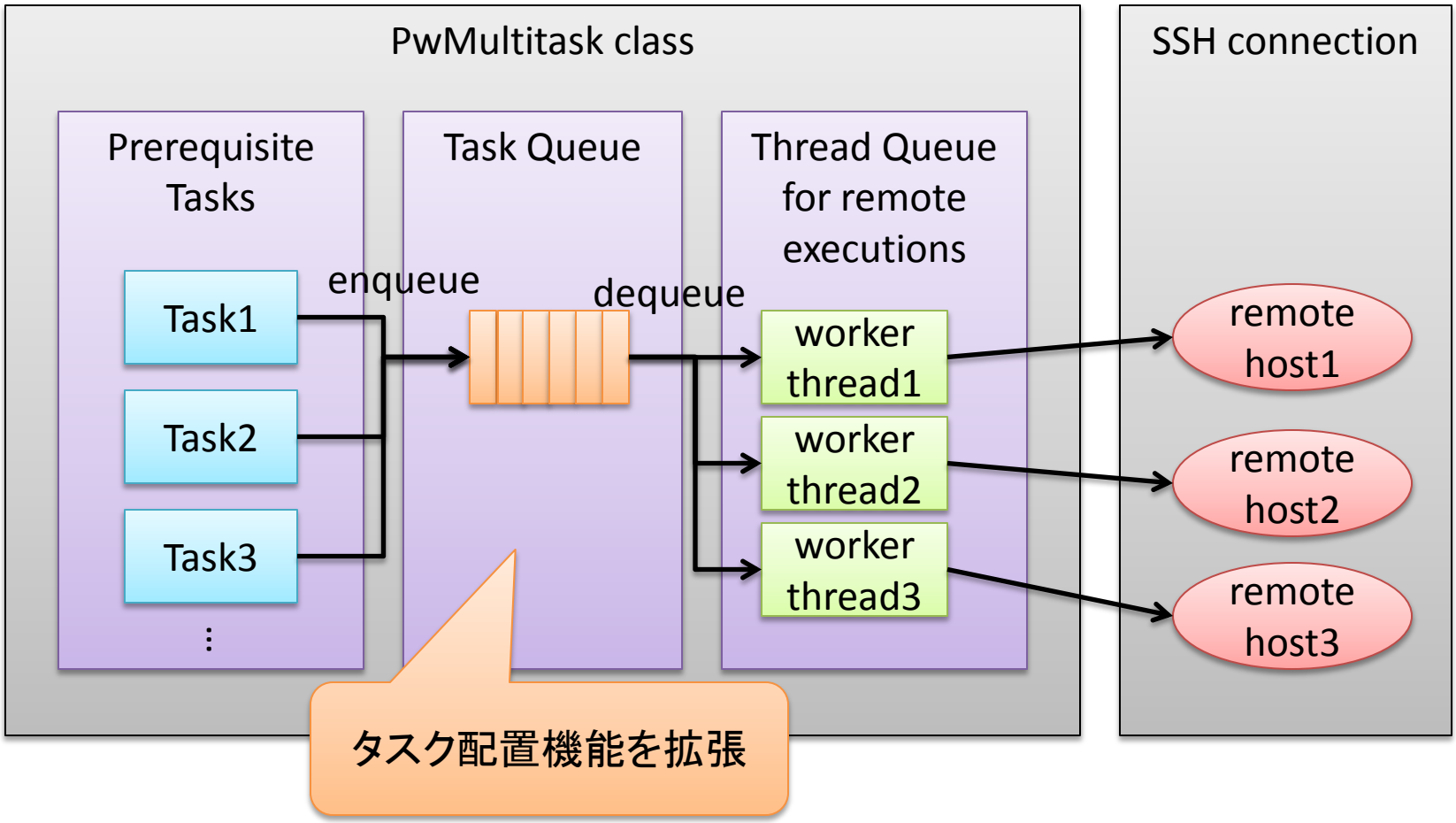
[背景] ワークフロー実行システム

- **Pwrake** (Parallel Workflow extension for Rake)
  - 分散並列ワークフロー実行システム
  - Tanaka & Tatebe 2010 HPDC
  - Ruby 版の make である、**Rake** がベース
  - タスクの並列分散実行機能を拡張

[背景] Rakefile におけるタスク記述

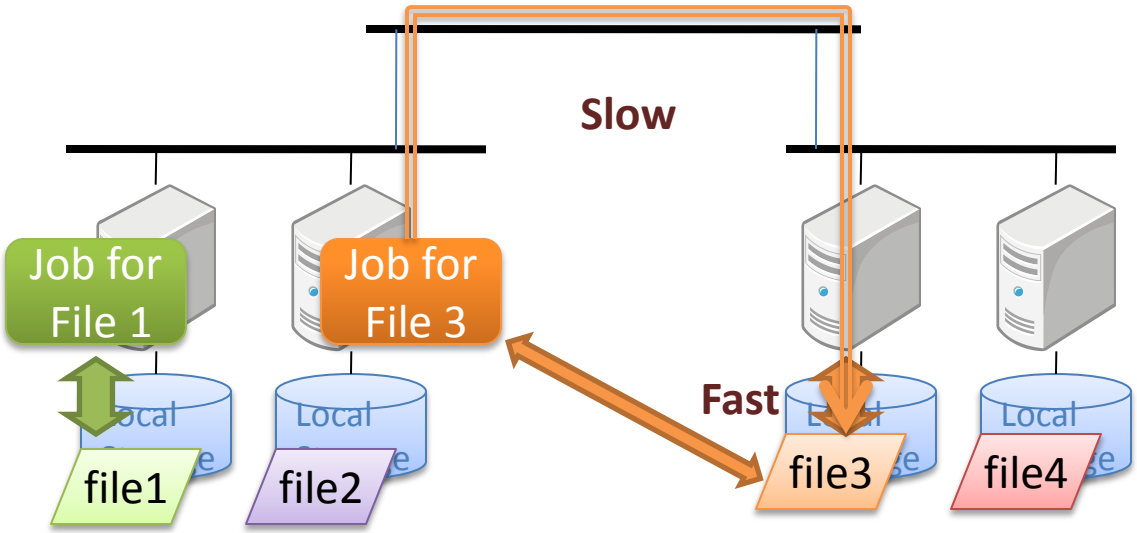


[背景] Pwrake 並列分散実行機能



[背景] ローカリティを考慮したタスク配置

- Gfarmは、計算ノードとストレージノードを兼ねることができる
- ファイルが格納されたノードで、タスク実行を実行することにより、スケーラブルな性能が期待できる
- タスク配置機能は、Pwrakeによって実現

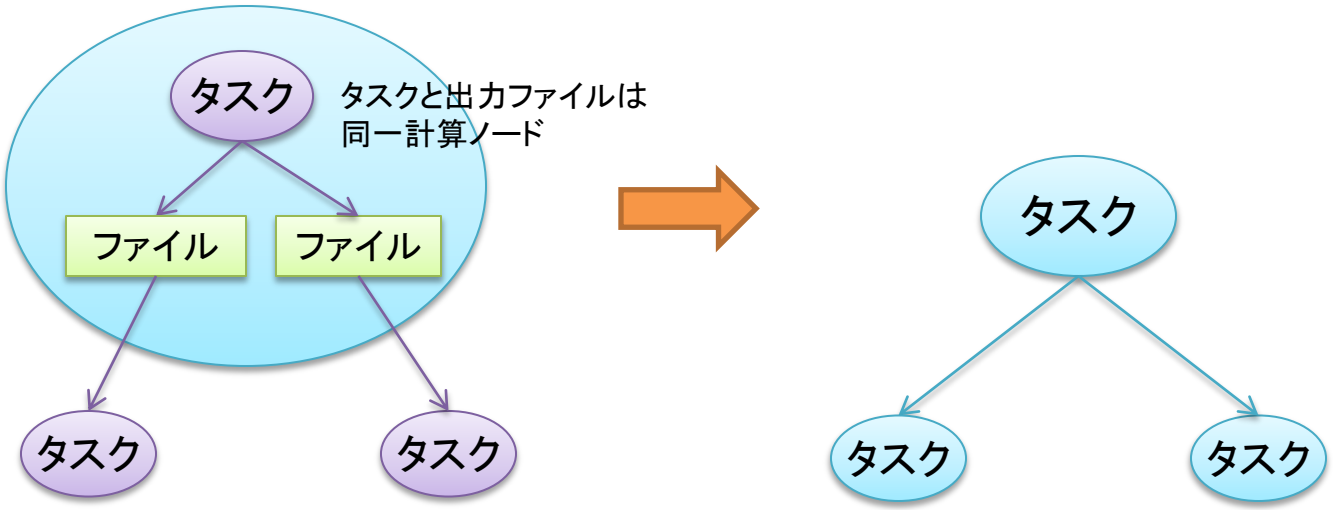


本研究の内容

- 先行研究では、Gfarmにおけるローカリティを生かしたワークフローシステムを実現
  - 入力ファイルの格納ノードを調べ、そこに近いノードでタスクを実行
  - それぞれのタスクと、その入力ファイルのみに着目
  - ワークフローにおけるタスクの依存関係は考慮していない
- 本研究では、さらに効率的なタスク配置を、ワークフローの構造に基づいて、自動的に行う手法について提案

ワークフローのグラフ表現

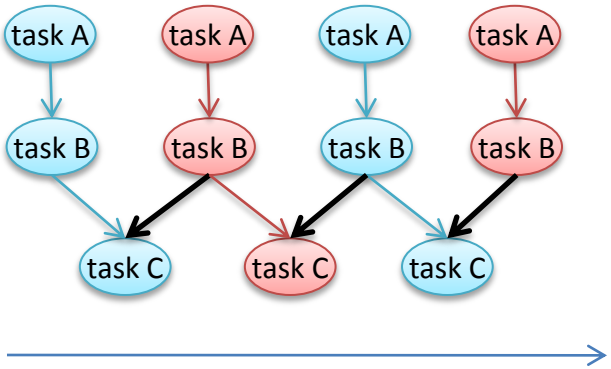
- DAG (Directed Acyclic Graph, 有向非循環グラフ)
  - タスク → 頂点
  - 依存関係 → エッジ
- 出力ファイル： タスクを実行した計算ノードに保存
  - 本発表のグラフ表現では、出力ファイルを省略



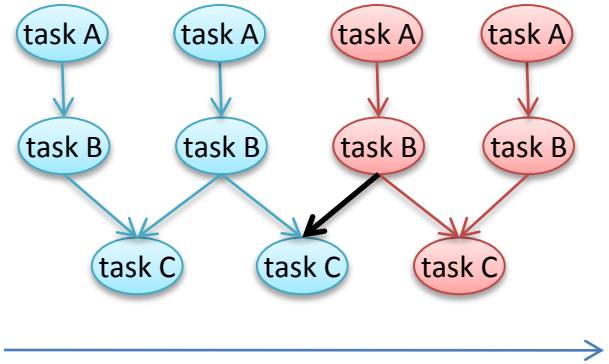
ワークフローの構造が性能に影響するケース

- 拠点へのタスク配置方法によって、拠点間のアクセスの回数が変わる

拠点間アクセス：3回



拠点間アクセス：1回



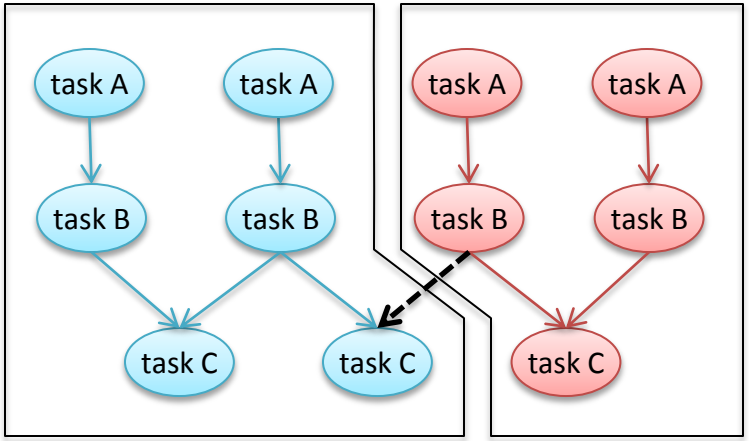
- データ並列の場合のように、
  データに位置関係がある場合に
  典型的

位置情報に基づくワークフロー

- 位置情報によるタスクのグループ化が有効
- 位置でグループ化する手法
  - 関連研究： 天文画像の座標によるグループ化
    - Meyer et al. 2006 SAC'06
- 位置情報に基づくタスク配置の欠点
  - アプリケーションと、ワークフロー処理系の両方について理解が必要
    - 位置情報は、アプリケーションの問題
    - タスク配置は、処理系の問題

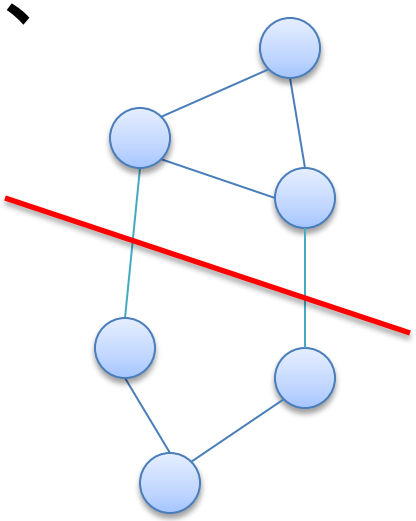
ワークフローのグラフから最適なタスク配置を求める問題

- グループをまたぐエッジの数が最小となるように、頂点をグループ化する問題
- グラフ分割問題 と同等



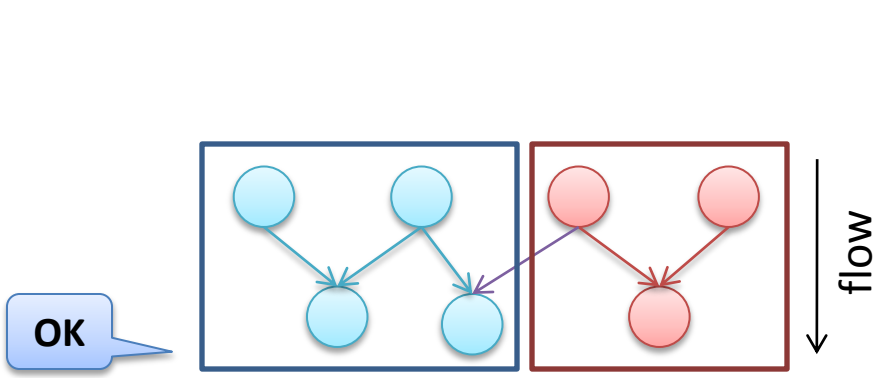
グラフ分割問題

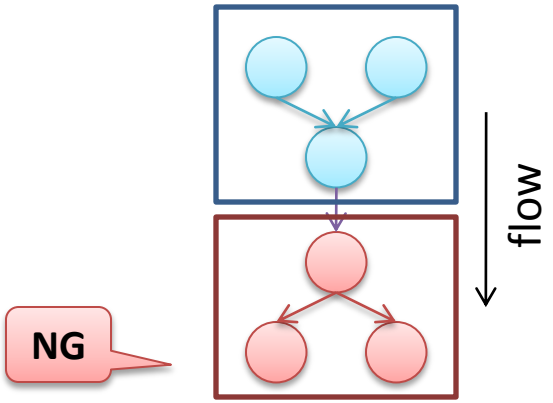
- カットするエッジが最小となるように、グラフの頂点を複数の部分集合に分割する問題
- NP完全
- アルゴリズムの研究が進んでいる
- 高速な実装：METIS
  (http://glaros.dtc.umn.edu/gkhome/views/metis)



グラフ分割問題と タスク配置問題の違い

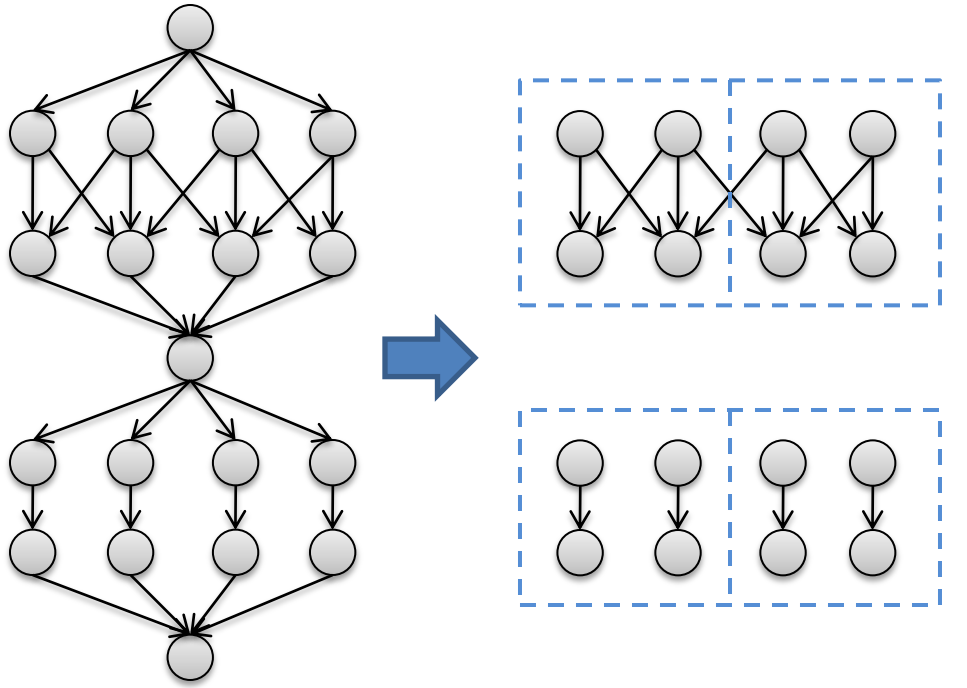
- グラフ分割問題
  - 各グループに属する頂点の数を、グラフ全体でバランスさせる
- タスク配置問題
  - 並列に実行可能なタスクを、各拠点に配分する問題
    - 並列に実行できないタスクを含めてグラフ分割を適用すると、ワークフローの前半と後半で別拠点に配分される





提案手法

1. ワークフローのグラフから、並列化できないタスクを取り除き、並列に実行可能なタスクを残す
2. エッジでつながった並列タスクのグラフについて、グラフ分割を適用



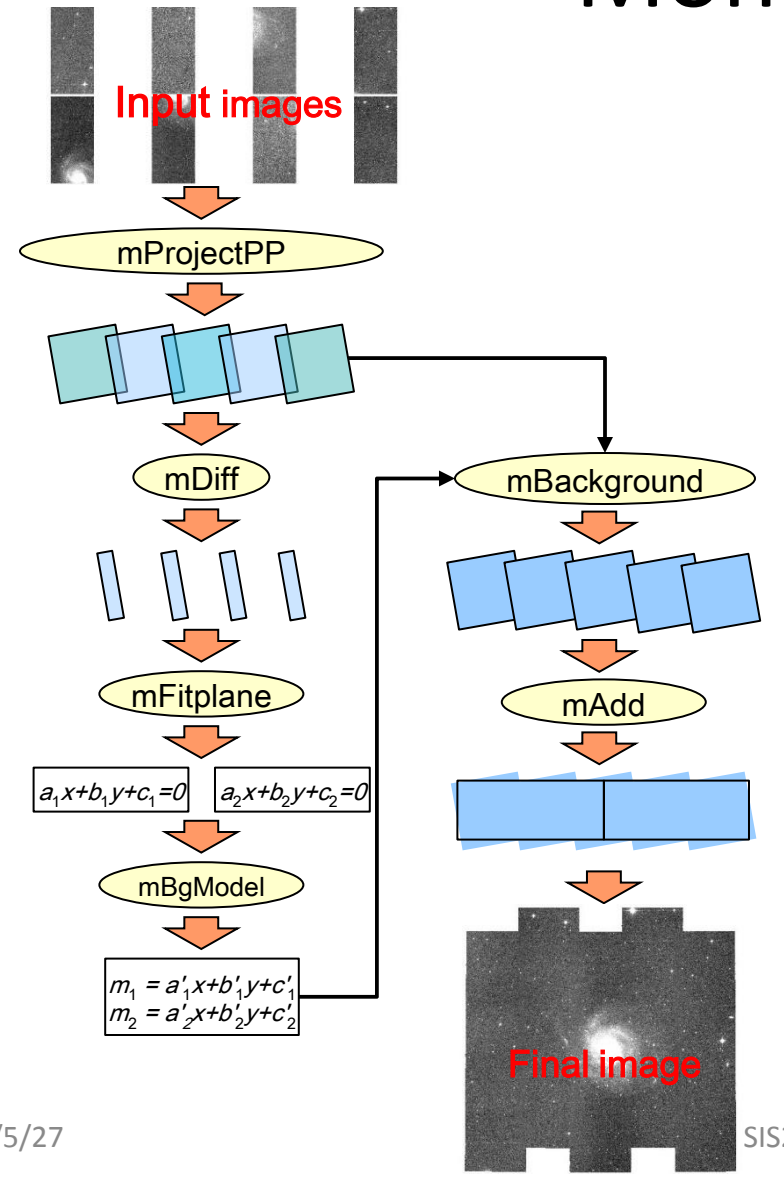
グラフ分割による
タスク配置

エッジカットが不要
→任意の拠点に配分

性能確認のための実装

- ワークフロー実行システム： Pwrake
  - 提案手法による グラフ分割機能を拡張
    - Rake::Taskインスタンスが持つタスク依存関係からグラフを構築
    - グラフ分割に使用するタスクは、タスク記述ファイル(Rakefile)内で指定
- グラフ分割計算： METIS
  - コマンドインタフェース pmetis を使用
  - オーバヘッドが小さい：数100ms
  - 頂点・エッジの重みは等しいとする
    - ファイルサイズやタスクの実行時間によって、重みをつけることが考えられるが、それらの情報があらかじめ得られるとは限らない

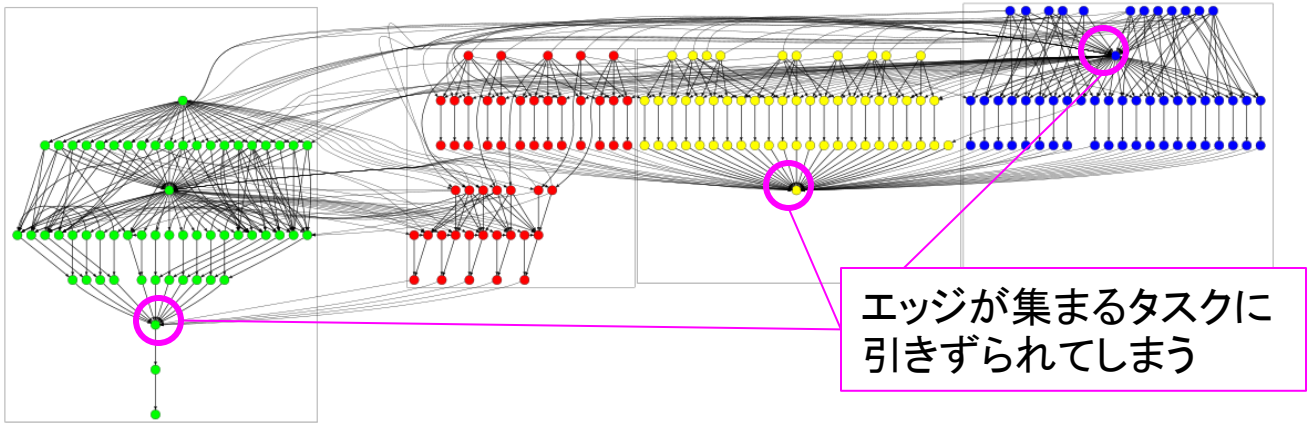
ケーススタディ: 天文画像合成ソフトウェア Montage



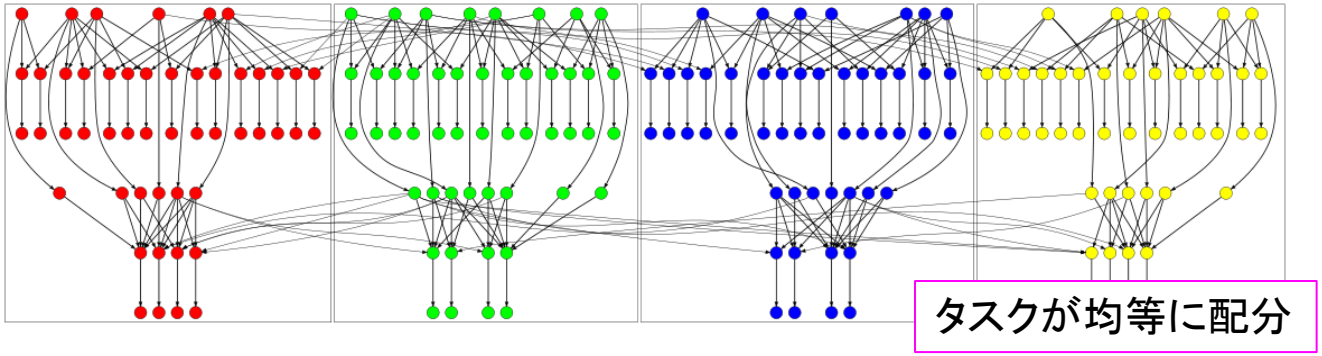
- 画像ファイル1つごとに、プログラムが処理
- 入力画像の数について並列処理が可能
- 処理内容
  - 座標変換
  - 背景の明るさがスムーズにつながるように調整
  - 画像の縮小、結合

Montageワークフローの分割例

ワークフローグラフ全体に対してグラフ分割を適用

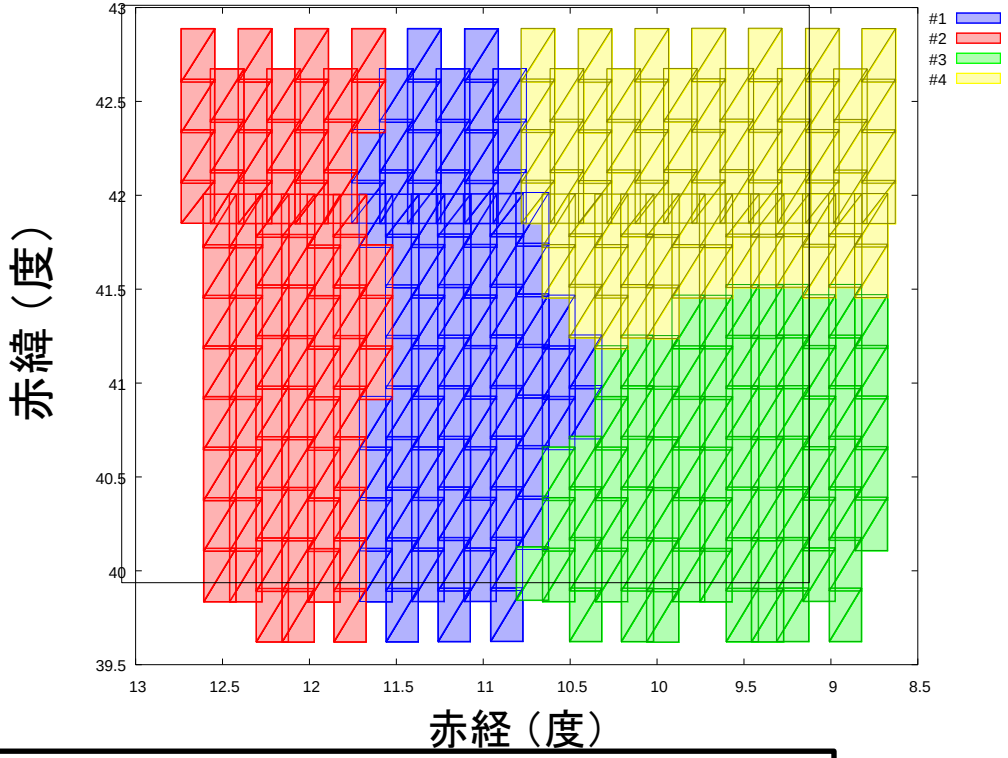


提案手法に基づき、並列タスクのみにグラフ分割を適用



分割結果を座標により確認

- 入力画像の座標をプロット
- 提案手法によるグループ配分の結果により色分け
  - 座標についてもグループ化されていることを確認
- 十字ではなく、縦で区切られた部分がある。
  - 入力画像が縦長のため



- 座標情報がなくても、提案手法により、適切にグループ化することが可能

測定環境

- InTrigger
  - 右表の上から順に拠点を追加し、1拠点から4拠点まで測定
  - 各拠点で同じコア数を使用
- Gfarmメタデータサーバ
  - 筑波ノード

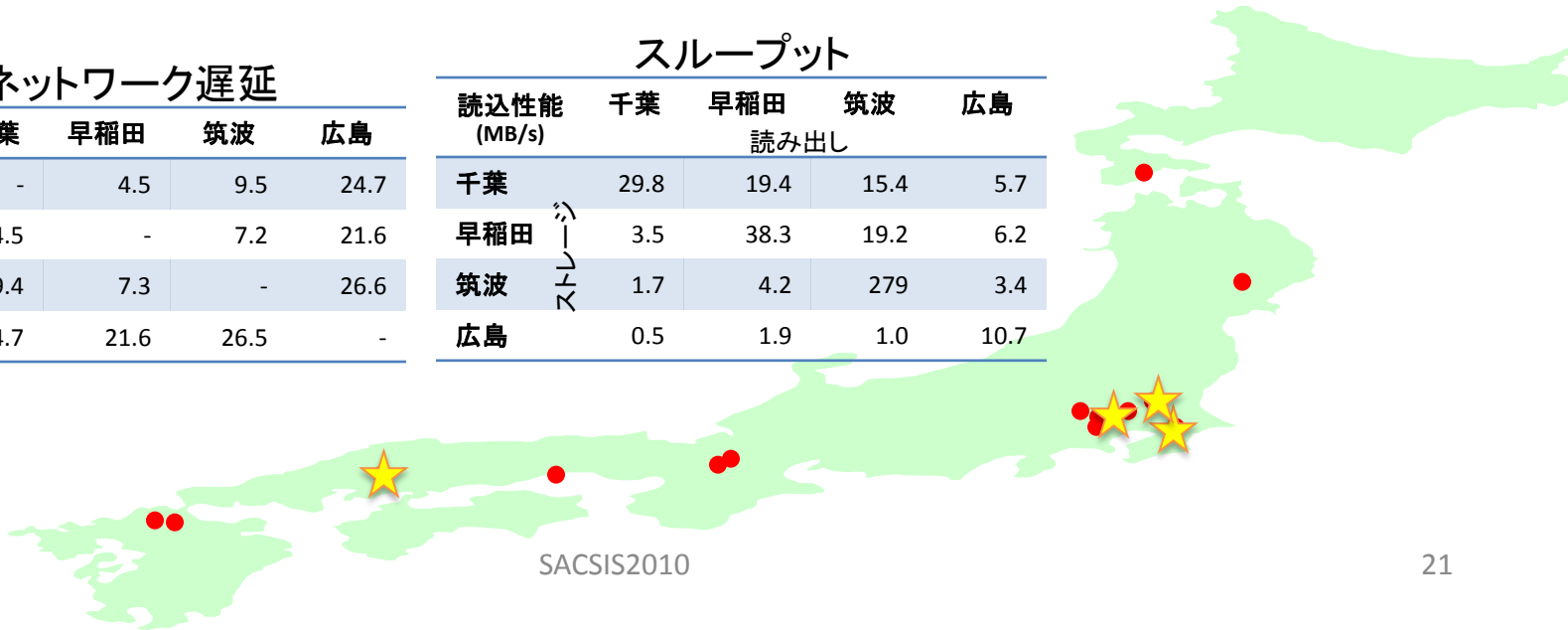
拠点	最大使用ノード数	最大使用コア数	CPU
千葉	16	48	Xeon 2.33GHz
早稲田	12	24	Core2 2.13GHz
筑波	6	24	Xeon 2.33GHz
広島	6	24	Xeon 2.33GHz

ネットワーク遅延

RTT (ms)	千葉	早稲田	筑波	広島
千葉	-	4.5	9.5	24.7
早稲田	4.5	-	7.2	21.6
筑波	9.4	7.3	-	26.6
広島	24.7	21.6	26.5	-

スループット

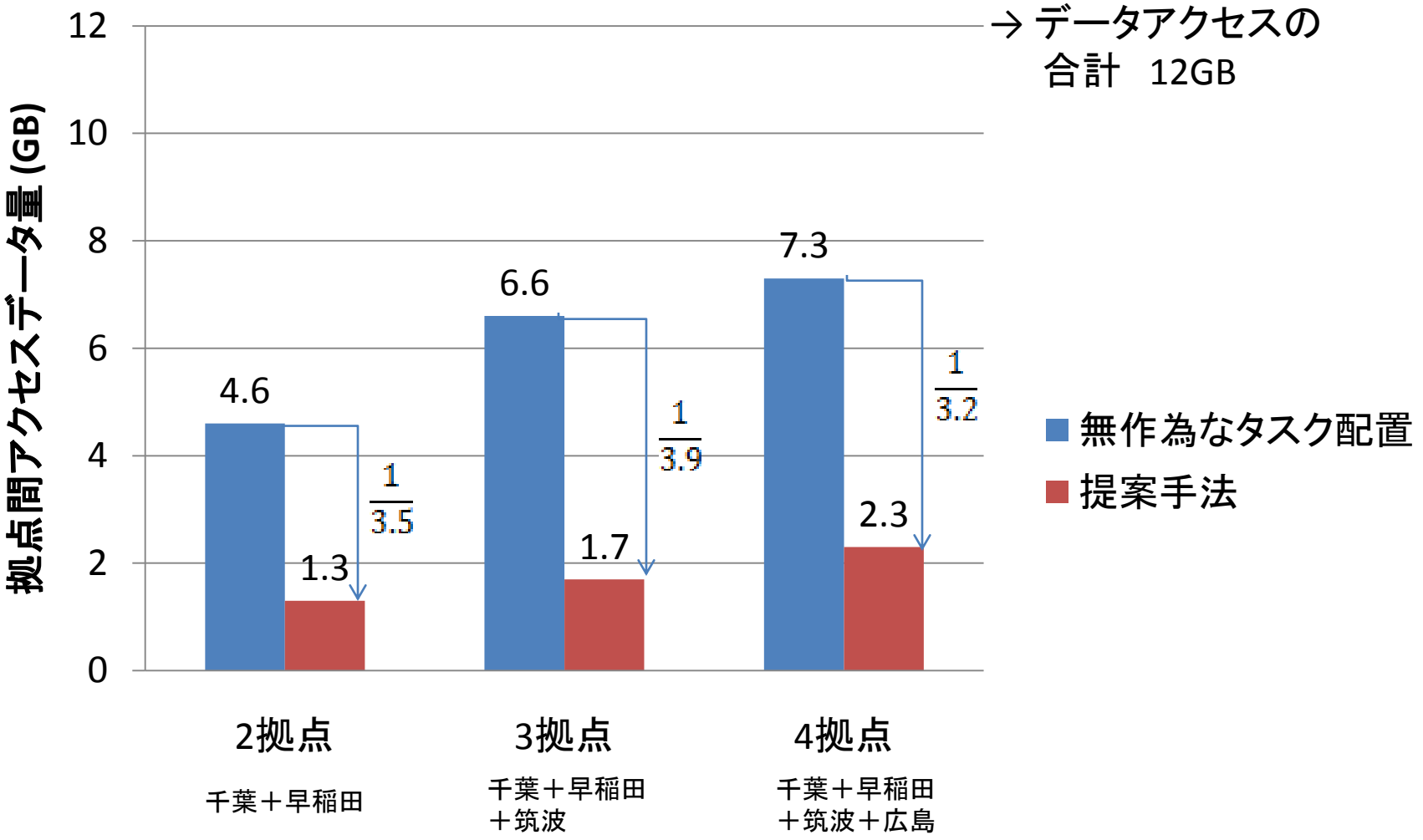
読込性能 (MB/s)		千葉	早稲田	筑波	広島
		読み出し			
千葉	ストレージ	29.8	19.4	15.4	5.7
早稲田		3.5	38.3	19.2	6.2
筑波		1.7	4.2	279	3.4
広島		0.5	1.9	1.0	10.7



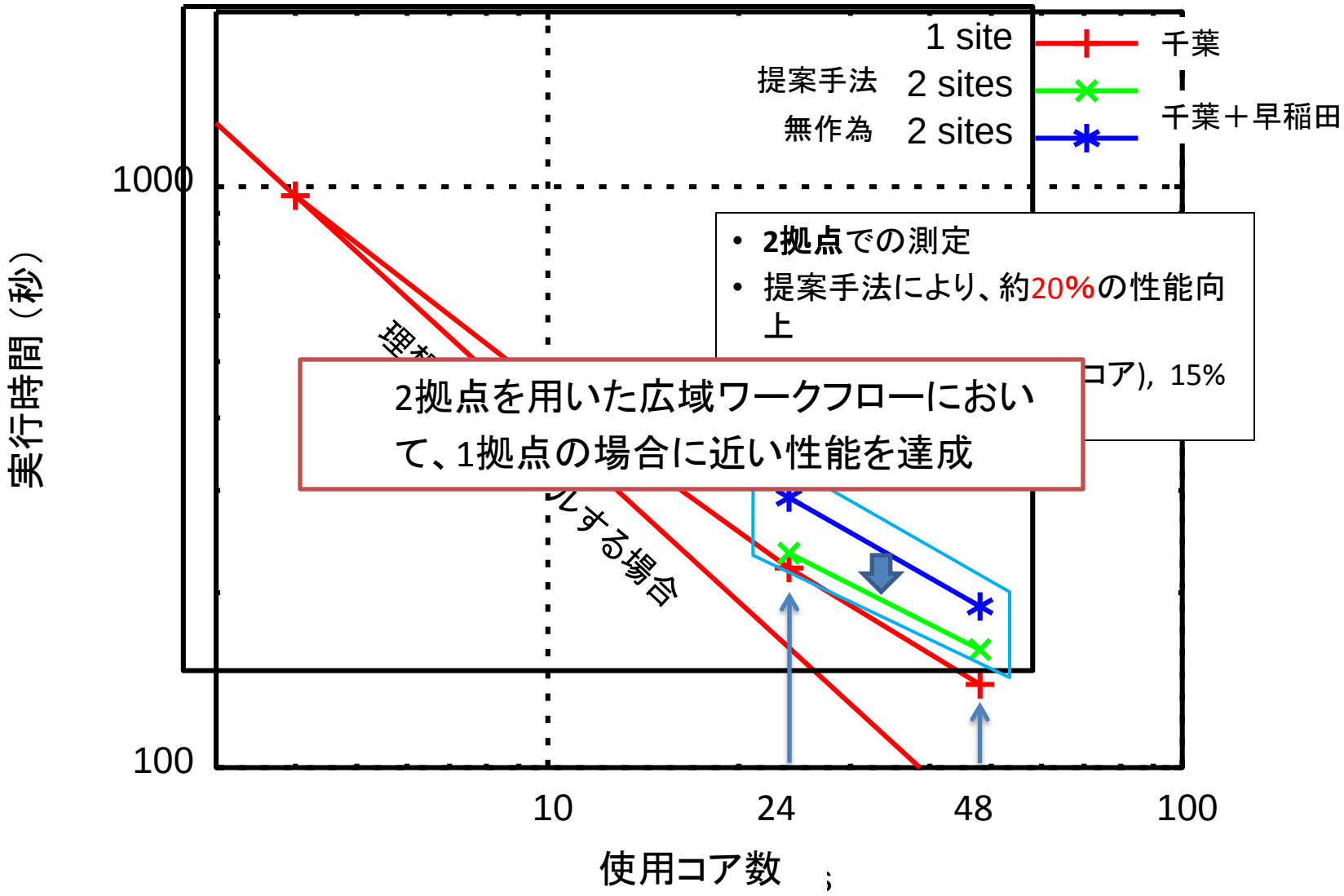
性能測定対象

- ワークフロー:　Montage
- 入力データ:　2MASS画像ファイル
  - 入力ファイル数:　309
  - 入力ファイルサイズ:　639 MB
  - データアクセス（延べ）:　約 12 GB
  - 初期状態:
    - 各拠点にデータセットを複製
    - 拠点内では、ファイルを各計算ノードに分配
- タスク配置方法の比較
  - 無作為な配置
  - 提案手法

拠点間アクセスデータ量

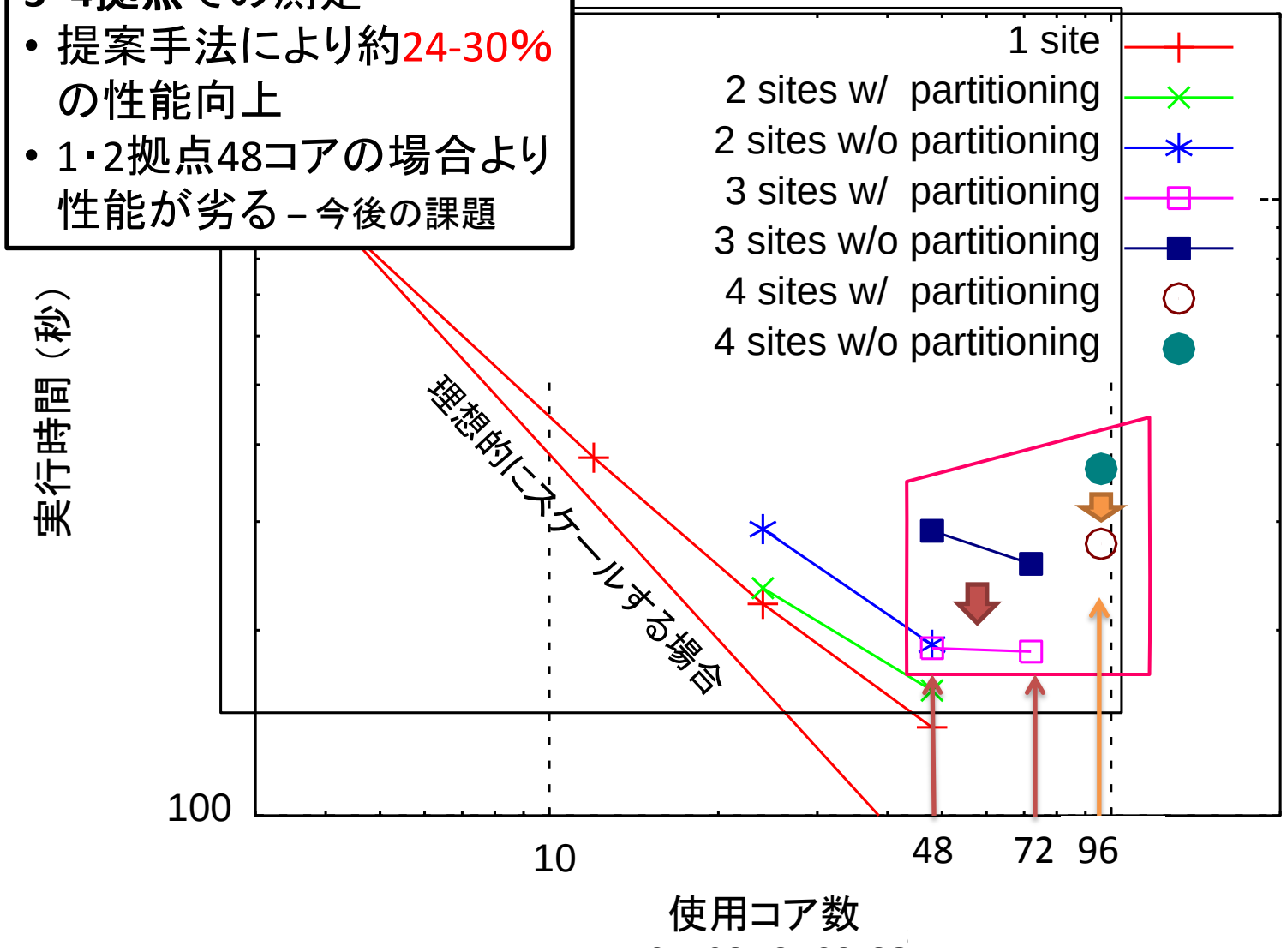


実行時間


1 site
提案手法 2 sites
無作為 2 sites
千葉
千葉＋早稲田
1000
100
10
24
48
100
実行時間（秒）
使用コア数
• 2拠点での測定
• 提案手法により、約20%の性能向上
2拠点を用いた広域ワークフローにおいて、1拠点の場合に近い性能を達成

実行時間

3･4拠点での測定
- 提案手法により約24-30%の性能向上
- 1･2拠点48コアの場合より性能が劣る – 今後の課題



まとめ

- 背景
  - 複数拠点の計算機資源を用いた、分散並列データ処理において、拠点間のファイルアクセスを最小化するため、タスク配置が重要
- 提案手法
  - ワークフロー自身から最適なタスク配置を自動的に決定するために、ワークフローの並列タスク部分に対して、グラフ分割を適用する手法
- 性能確認
  - InTrigger の4拠点を用いて、天文画像合成ワークフローの実行時間を測定
  - タスク配置を考慮しない場合に比べて、拠点間アクセスの削減を確認
  - 実行時間は、20-30% のスピードアップ
  - 2拠点を用いた広域ワークフローにおいて、1拠点の場合に近い性能を確認
  - 3拠点以上でのスケーラビリティ達成が、今後の課題

今後の課題

- 3拠点以上のスケーラビリティ
- 他のワークフローについて性能評価
- タスク配置問題を解くアルゴリズム
  - 各種パラメータの考慮
    - ファイルサイズ、タスクの規模、マシン・ネットワーク
  - 並列実行可能なタスクの考慮
  - 入力ファイルが配置された拠点の考慮
  - 拠点だけでなくノードも考慮したグループ化
- 拠点間のファイル複製手法の利用