科研費基盤研究B (FY2024-2026)

ワークフローエンジンとの連携に基づく臨機応変なジョブスケジューリングの実現

2024年12月26日
ATTA2024@工学院大学

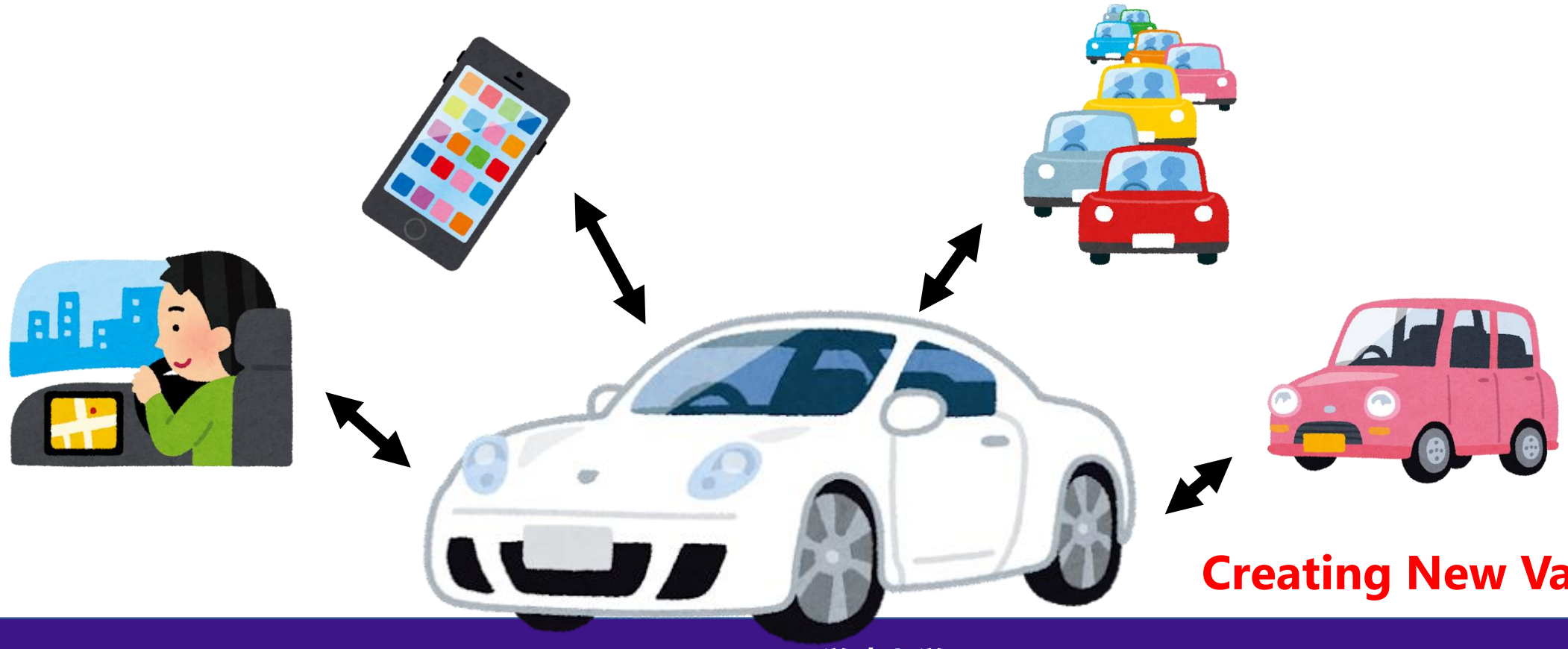
東北大学サイバーサイエンスセンター
滝沢 寛之
<takizawa@tohoku.ac.jp>

概要

- 次世代HPCシステムは外部と接続され、その状況は時々刻々と動的に変化する。本研究の目的は**時々刻々と変化する状況下で多様なワークフローに対して適切な計算資源を臨機応変に割当て、効率よく実行するジョブスケジューリング技術を確立すること**である。現在の多くのジョブスケジューラはワークフローの動的な依存関係を管理していないことから、まずはワークフローエンジンとジョブスケジューラとの密な連携を実現し、その上で自己モニタリングに基づいて性能ボトルネックに配慮したジョブスケジューリングを実現する。その結果、システム全体としての計算資源利用効率を落とすことなく、実行優先度の高いジョブの緊急実行などの動的な要因への迅速な対応も実現する。

背景: コネクテットカー (スマートカー)

- 現代の自動車は様々な**外部資源**や**実世界情報**と直接接続
  - **サイバー空間**と密に接続することで**実空間内**で安全で高度な制御を実現

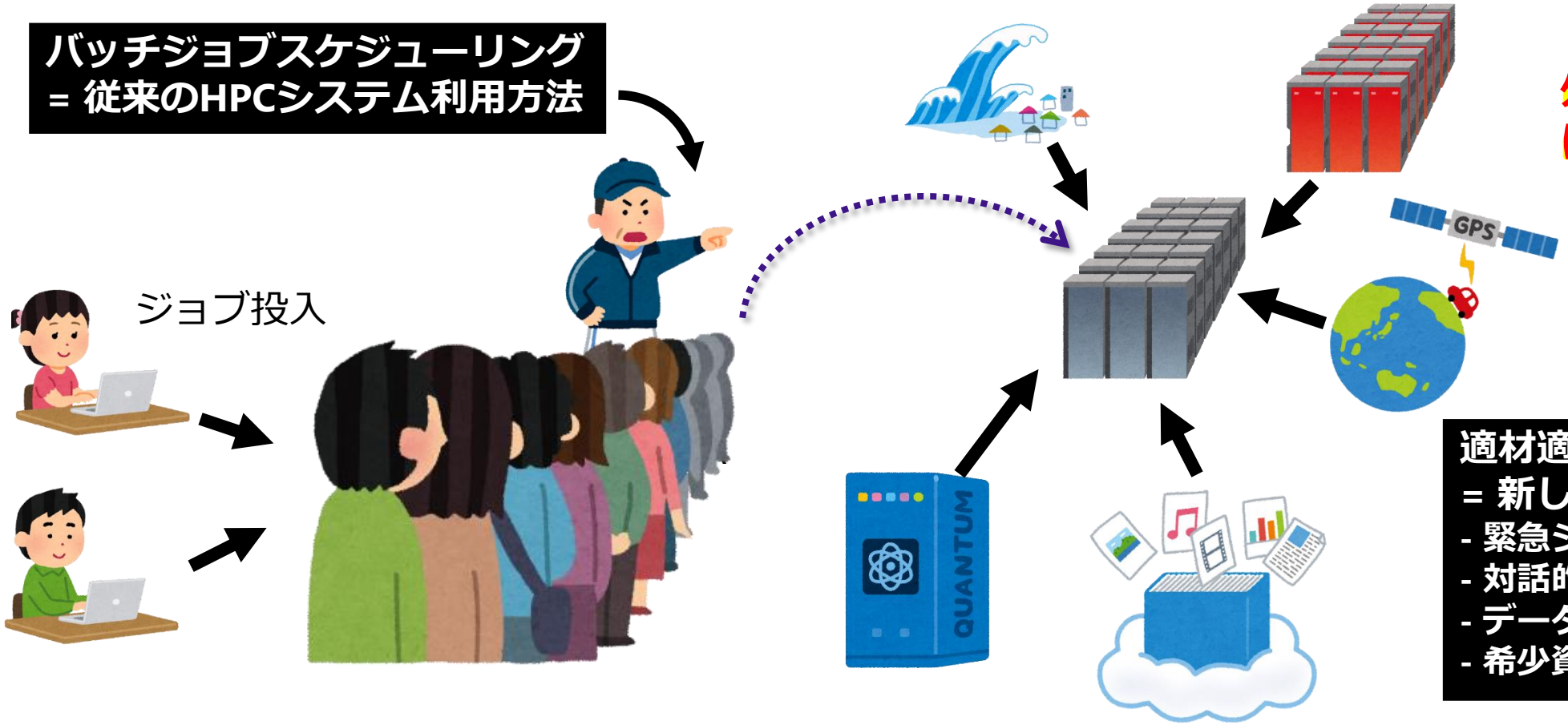


Creating New Values!

背景：コネクテットスパコン

- HPCシステムは様々な**外部資源**や**実世界情報**と直接接続

バッチジョブスケジューリング
= 従来のHPCシステム利用方法



外部要因が資源管理に動的に影響

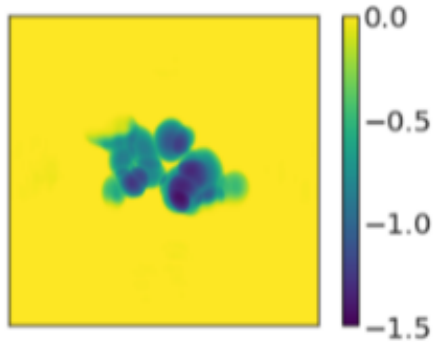
適材適所適時のジョブ実行
= 新しいHPCシステム利用方法
- 緊急ジョブ実行
- 対話的ジョブ実行
- データストリーミング
- 希少資源共有 etc

例) AOBAとNanoTerasuを直結

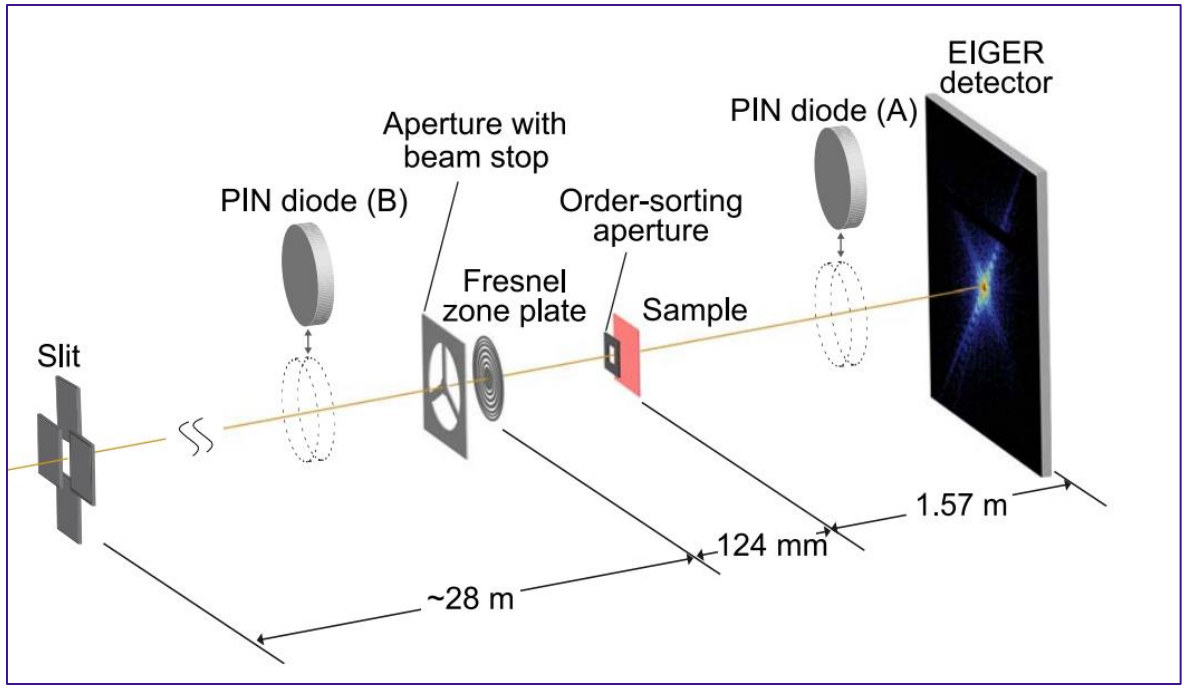
- **巨大なデータの蓄積と解析の実現を目指して**
  - 従来の計算科学分野に加えて、データ駆動科学、材料開発、製薬、農学などの様々な研究分野での顕著な貢献がAOBAに期待されている

例) 実時間データ処理の必要性

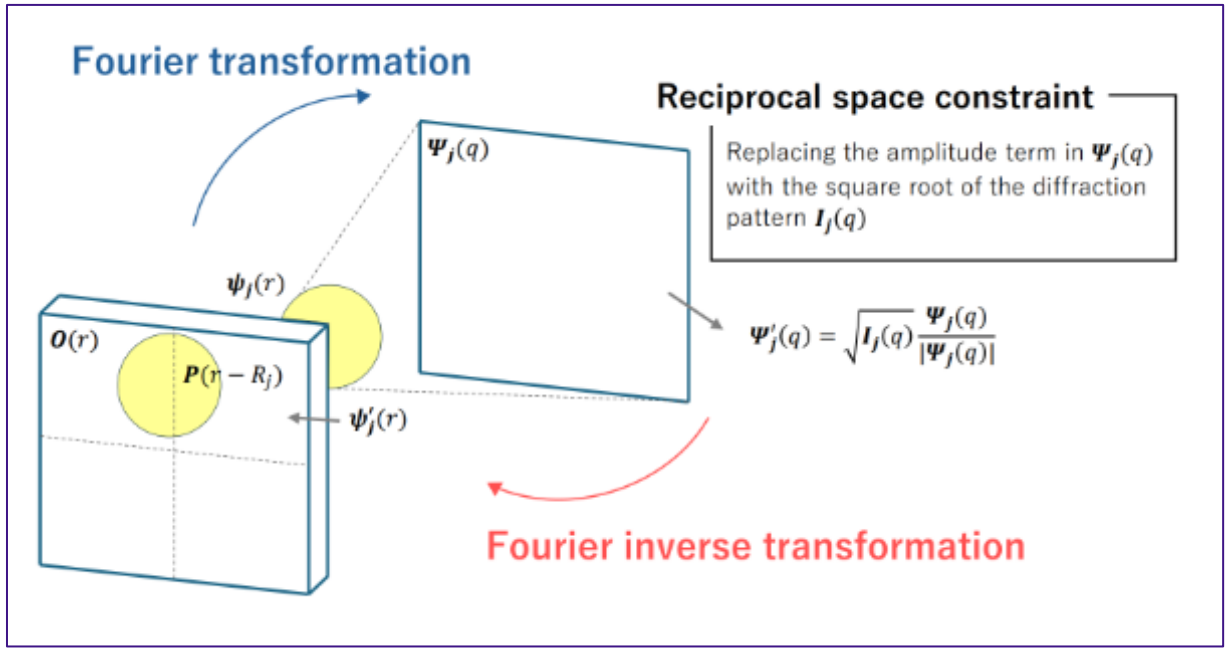
Visualized image of a nano-scale sample



Diffraction intensity patterns are obtained.
(phase information is not)



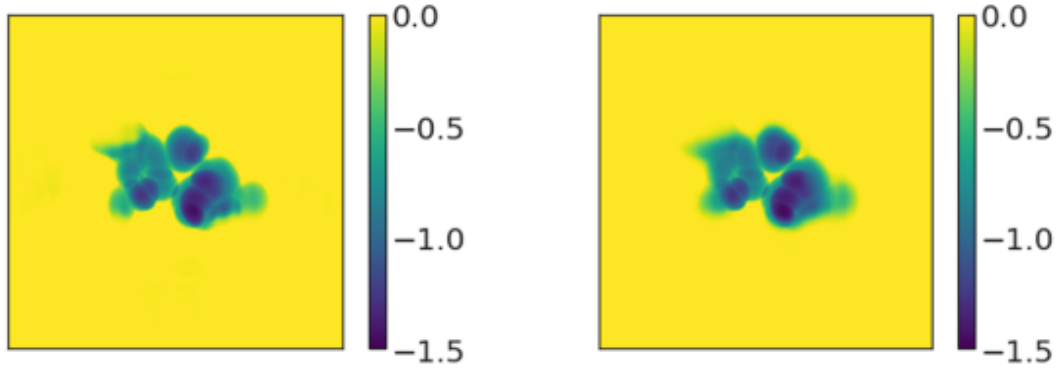
Tender X-Ray PCDI at BL10U in NanoTerasu (Ishiguro et al. 2023)



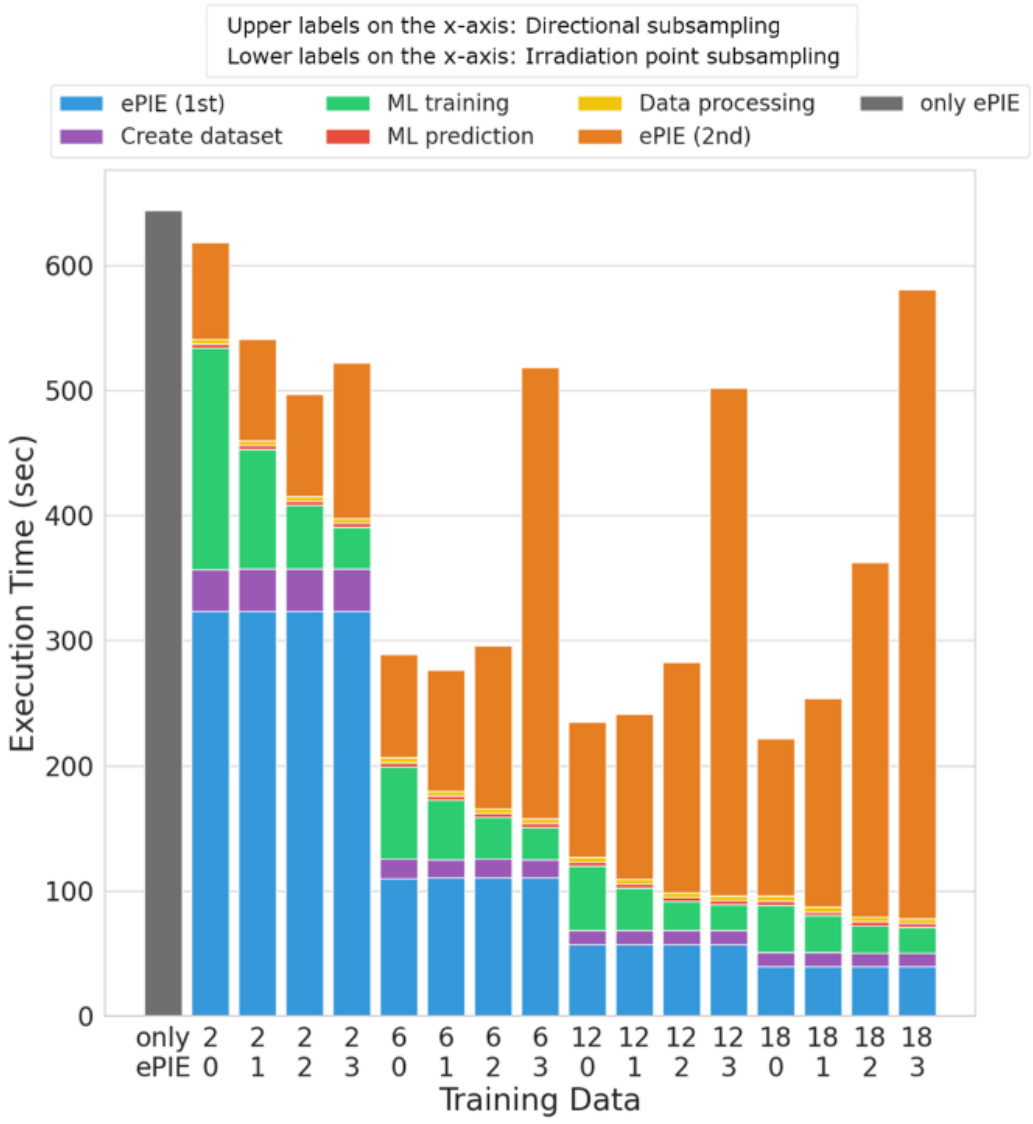
Phase retrieval (ePIE) on **AOBA**

ML-Assisted Diffraction Imaging

- Proposed method is up to 2.9 times faster
  - Training data with a directional subsampling of X and an irradiation point subsampling of Y is denoted as (X, Y)
  - Execution time
    - Conventional: **643.8 sec**
    - Proposed (18, 0): **221.8 sec**



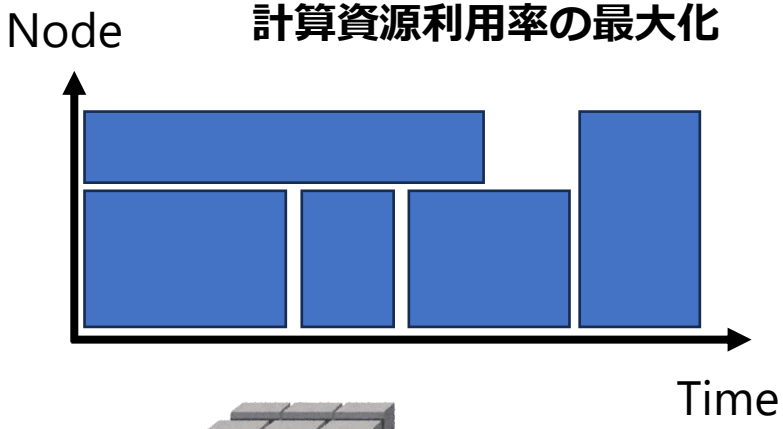
(a) Only ePIE (b) Proposed method (18, 0)

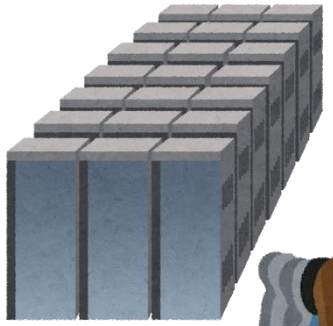


背景: 高度な資源管理の必要性

古典的なスパコン利用

計算資源利用率の最大化

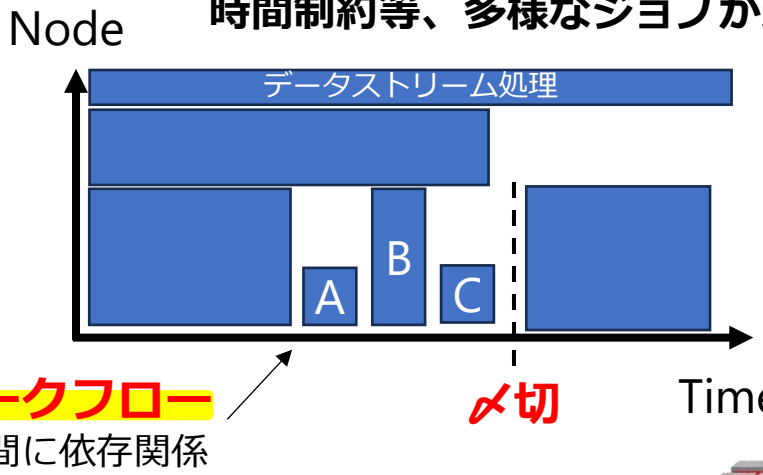




順番待ちをして
公平かつ効率的に利用

現代的なスパコン利用

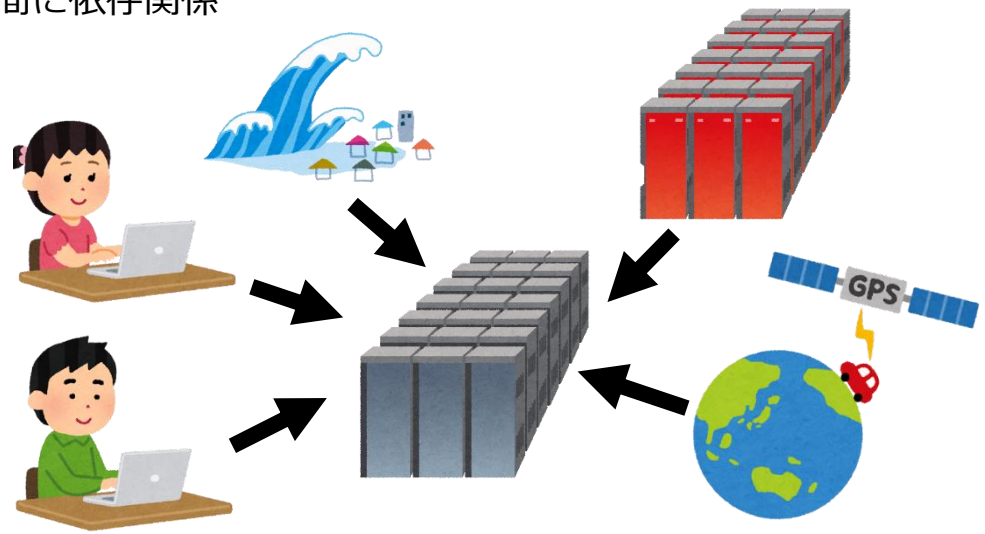
時間制約等、多様なジョブが連携する必要



ワークフロー

ジョブ間に依存関係

〆切

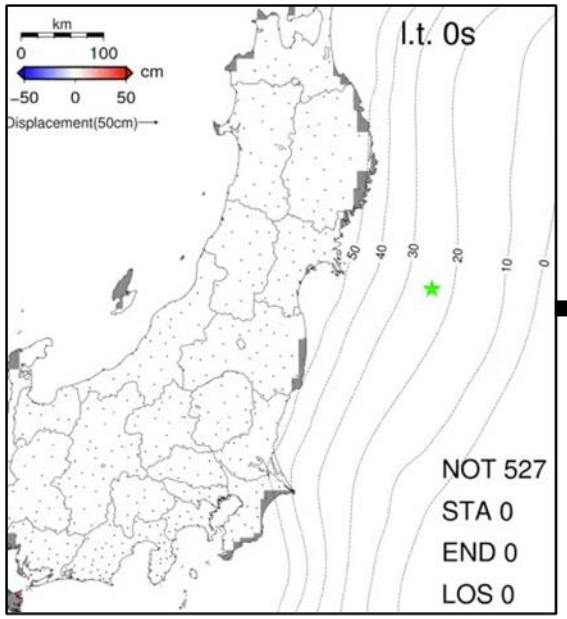


Full-automatic real-time estimation of tsunami damage

Courtesy of Prof. Koshimura

>7min (present)

>3min (target)



Real-time estimation of fault model from GEONET data (REGARD; GSI)

>10min (present)

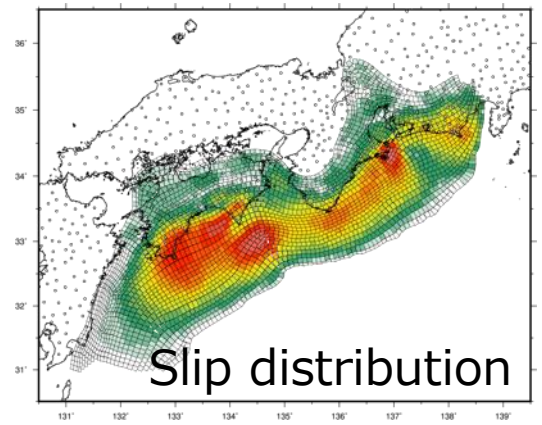
>1min (target)

Real-time simulation on Supercomputer AOBA at Tohoku University with considering current costal facilities and tide height at the occurrence.

>10min (present)

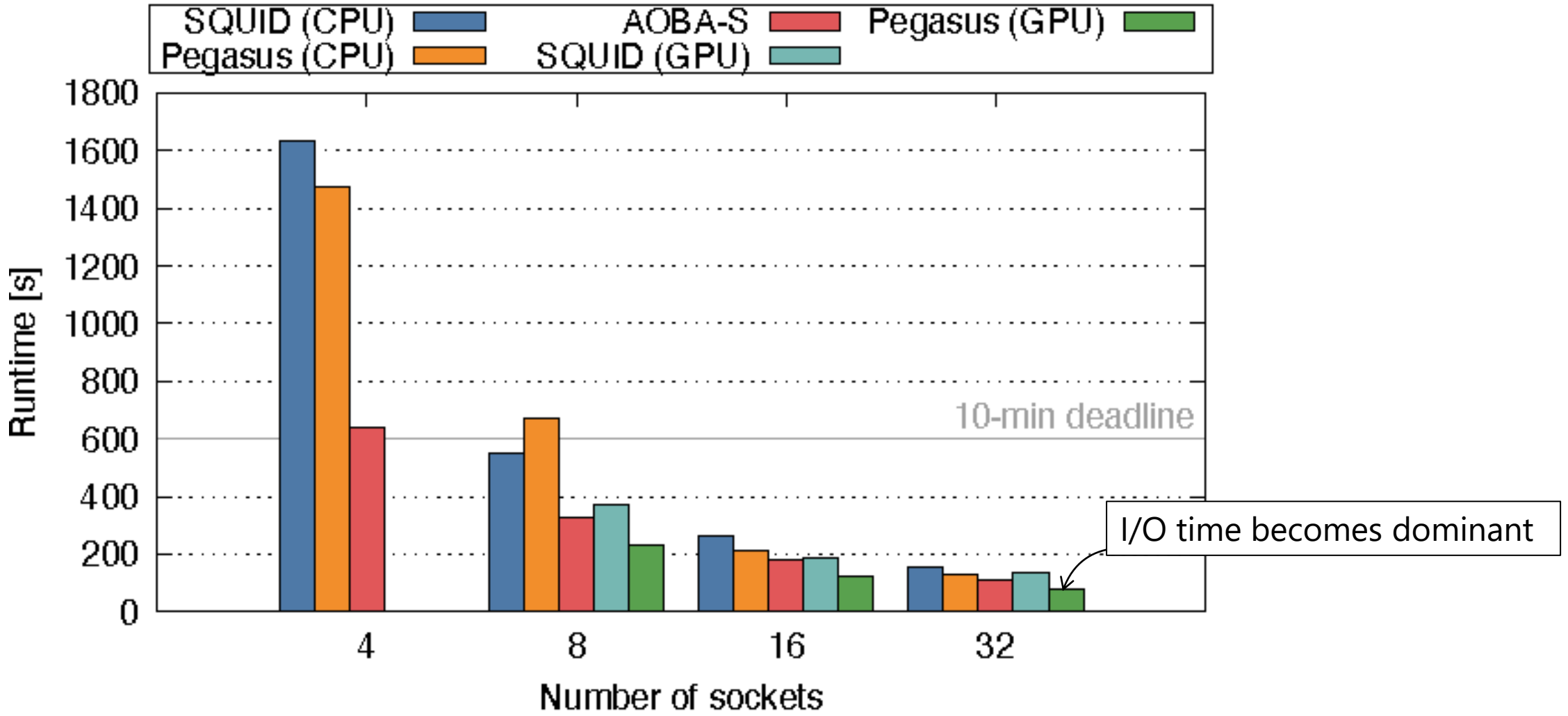
>1min (target)

Quantitative estimation of damages



Successfully operated upon several earthquake in 2023-2024

Porting Tsunami Sim. To GPUs



The ExpressHPC Project (ongoing)

Sensor Data

ExpressHPC

Tsunami Damage Estimation of the entire coastal area of Japan

PSI/J (*) = **P**ortable **S**ubmission **I**nterface for **J**obs

Slurm	PBS Pro	TC-S	NQSV
HW-A	HW-B	HW-C	HW-D

Wide varieties of computing resources and their operation policies

- Performance portability
- Selection of computing resources

* **ExaWorks PSI/J** https://exaworks.org/psij

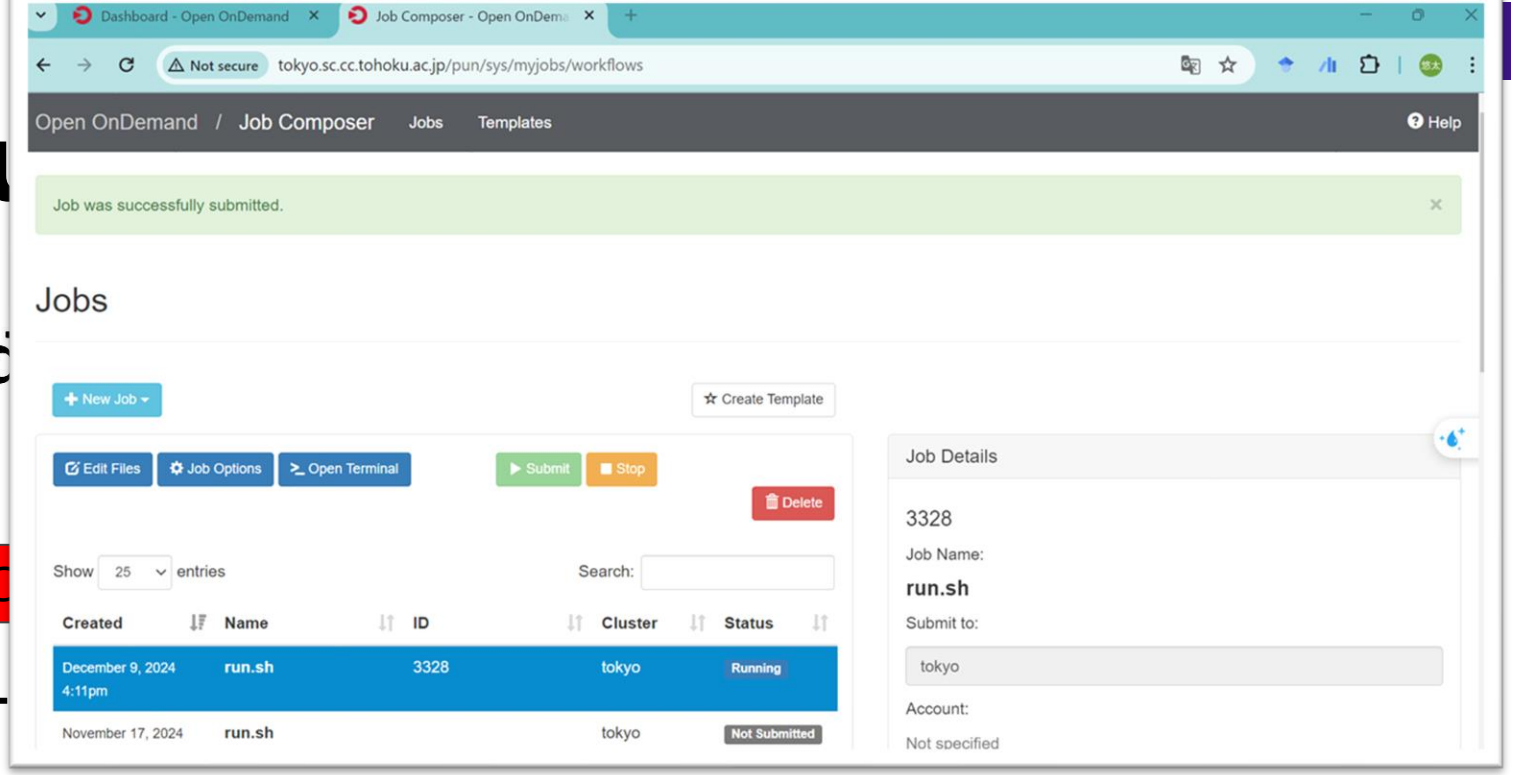
→ **Exploring cost-effective approaches.**

Side Product of Bu

Original

OPEN OnDemand

Slurm | PBS Pro | TC



Ours

OPEN OnDemand

PSI/J (*) = **P**ortable **S**ubmission **I**nterface for **J**obs

Slurm | PBS Pro | TC-S | NQSV

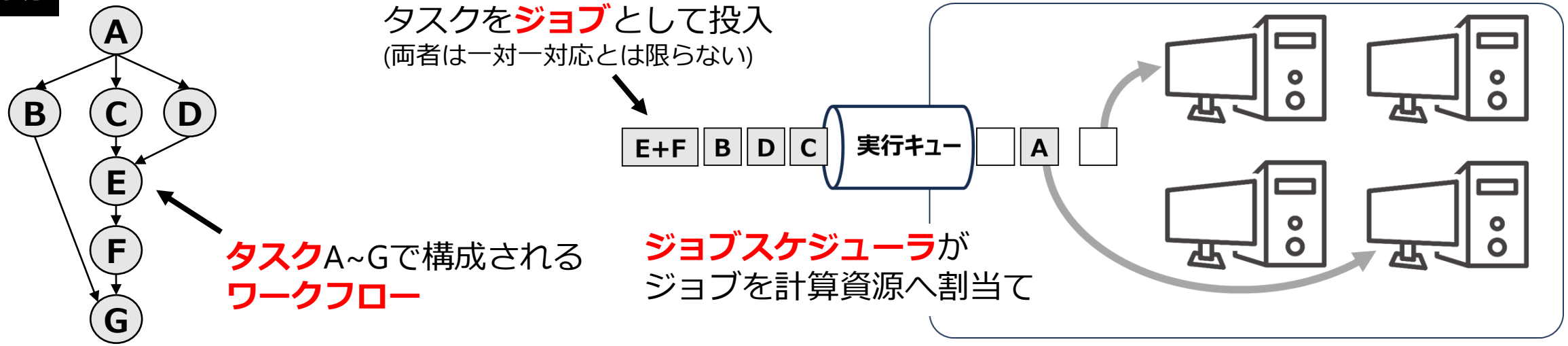
Open OnDemand needs to support only PSI/J. → lower maintenance cost

NEC's job scheduler was not originally supported, but now supported using our portability layer.

本研究課題の目的

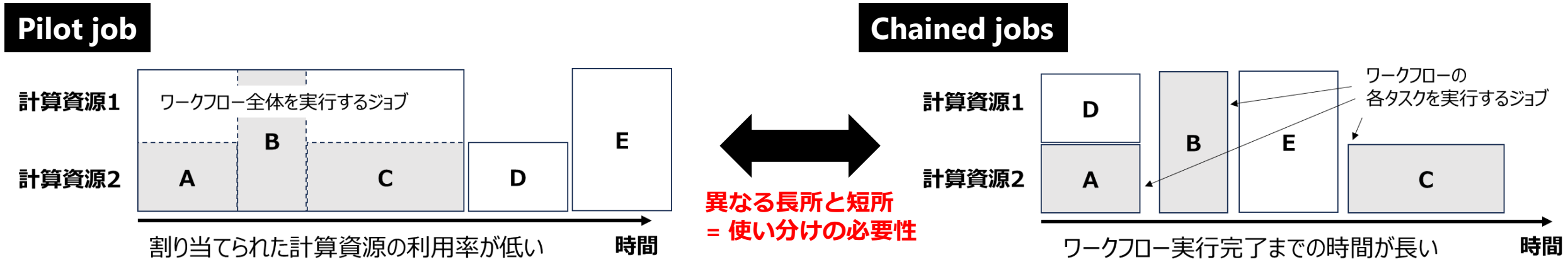
- **依存関係のある複数の処理(タスク)が連携するワークフローを前提として、時々刻々と変化する状況に臨機応変に対応することで、システムの性能を最大限に活用する技術の確立**
  - 多様なタスクの協調実行(=ワークフロー)の効率と計算資源利用率の両立
  - 共有資源競合に起因する性能の低下とばらつきの抑制
  - 外部との連携に伴う動的な状況変化への対応

用語説明



本研究課題の目的

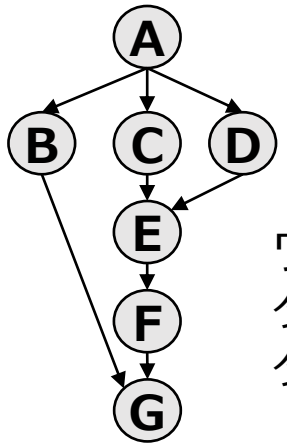
- 依存関係のある複数の処理(タスク)が連携するワークフローを前提として、時々刻々と変化する状況に臨機応変に対応することで、システムの性能を最大限に活用する技術の確立
  - **多様なタスクの協調実行(=ワークフロー)の効率と計算資源利用率の両立**



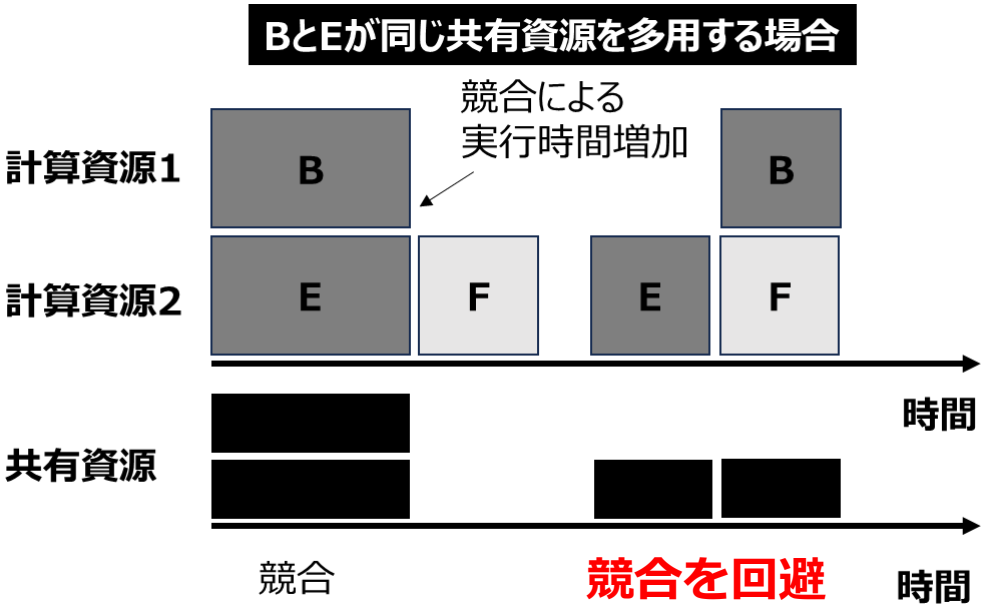
- 共有資源競合に起因する性能の低下とばらつきの抑制
- 外部との連携に伴う動的な状況変化への対応

本研究課題の目的

- 依存関係のある複数の処理(タスク)が連携するワークフローを前提として、時々刻々と変化する状況に臨機応変に対応することで、システムの性能を最大限に活用する技術の確立
  - 多様なタスクの協調実行(=ワークフロー)の効率と計算資源利用率の両立
  - **共有資源競合に起因する性能の低下とばらつきの抑制**



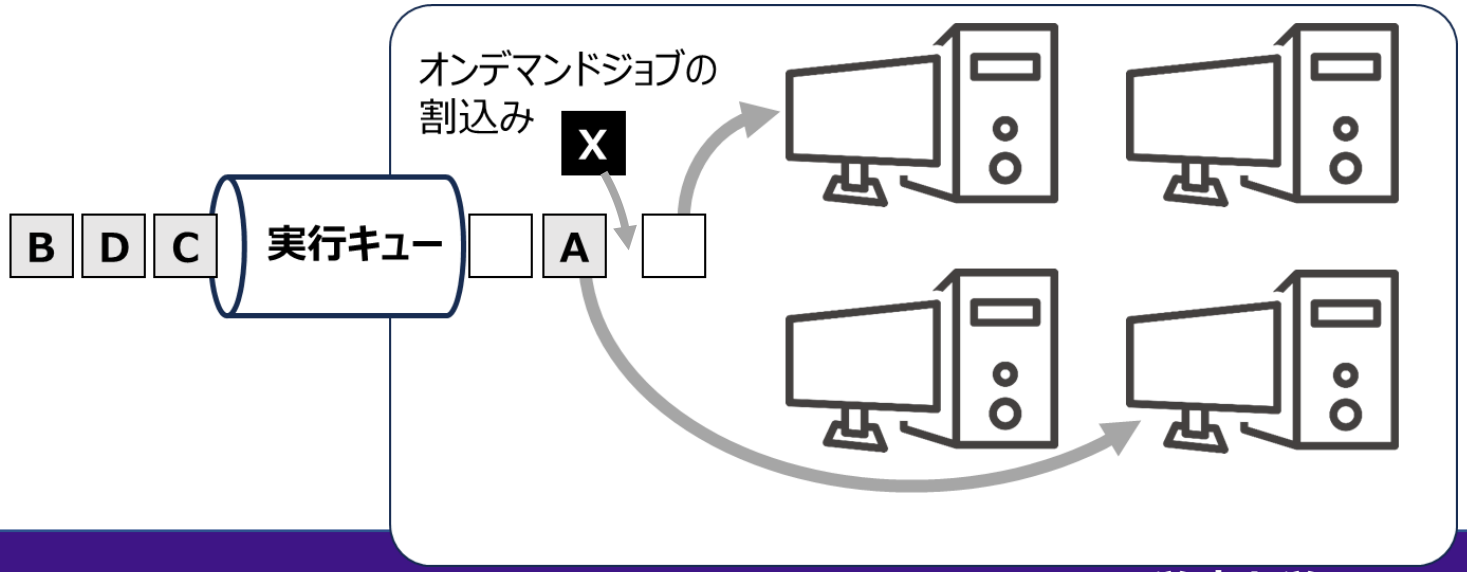
ワークフロー全体の依存関係を俯瞰できれば
タスクEはクリティカルパス上にある一方で
タスクBは実行を遅延可能であることが分かる



  - 外部との連携に伴う動的な状況変化への対応

本研究課題の目的

- 依存関係のある複数の処理(タスク)が連携するワークフローを前提として、時々刻々と変化する状況に臨機応変に対応することで、システムの性能を最大限に活用する技術の確立
  - 多様なタスクの協調実行(=ワークフロー)の効率と計算資源利用率の両立
  - 共有資源競合に起因する性能の低下とばらつきの抑制
  - **外部との連携に伴う動的な状況変化への対応**

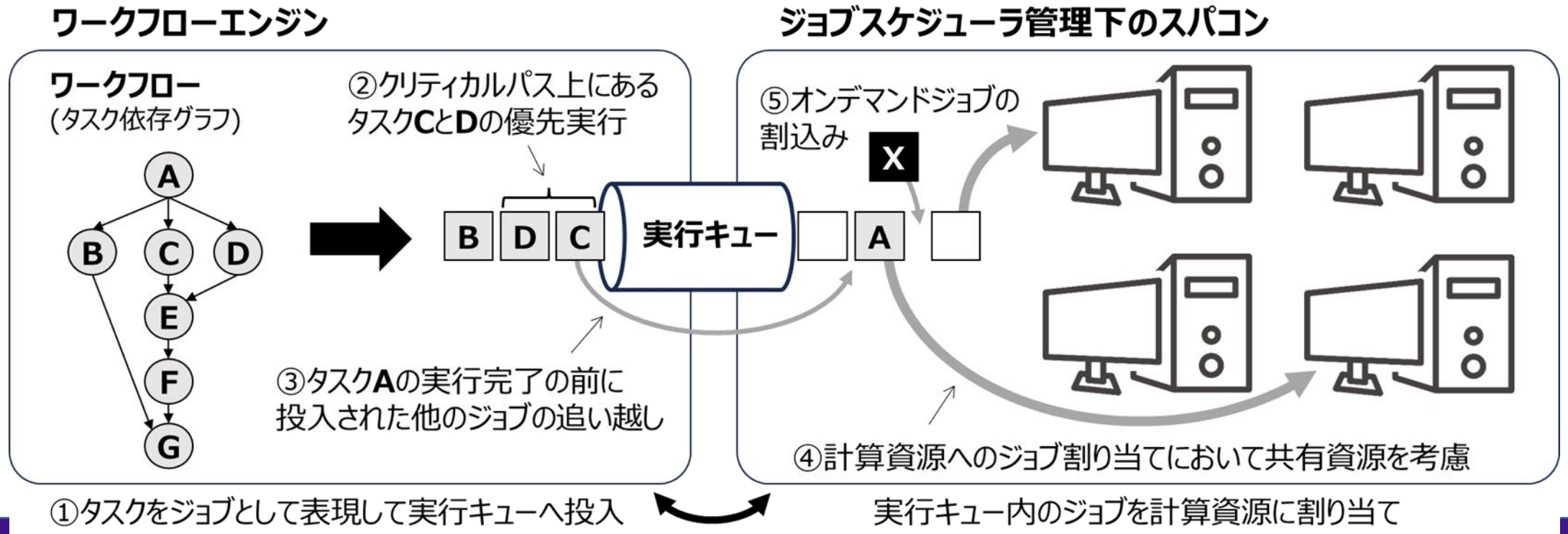


計算資源の動的な再割当て

= タスク間の依存関係および計算資源の割当て状況を見直し

本研究で確立を目指す技術

- 現状：ワークフローエンジンとジョブスケジューラは、それぞれ担当する限られた情報に基づいて動作
- 提案：**ワークフローエンジンとジョブスケジューラの情報共有と連携**



情報共有と連携が必要不可欠

今年度の研究成果(as of Dec 26)

- Cui et al. : Clustering Based Job Runtime Prediction for Backfilling Using Classification, JSSPP24.
- Nakai et al. : A node selection method for on-demand job execution with considering deadline constraints, JSSPP24.
- Ishii et al. : Maximizing Energy Budget Utilization Based on Dynamic Power Cap Control, JSSPP24.
- **柳井ら: タスク間の依存関係を考慮したワークフローのバッチジョブスケジューリング, xSIG24.**
- 谷澤ら: HPCシステム用ウェブポータルにおけるジョブスケジューラの抽象化, HPC研究会(SWoPP).
- Takahashi et al.: Modernizing an Operational Real-time Tsunami Simulator to Support Diverse Hardware Platforms, CLUSTER24.
- Takizawa et al.: ExpressHPC: towards "connected supercomputing" enabling on-demand job execution for disaster resilience, WIUHPC@SC24.
- Koda et al.: Real-Time Phase Retrieval Using On-the-Fly Training of Sample-Specific Surrogate Models, CANDAR24.
- Cui et al.: DRAS-OD: A Reinforcement Learning based Job Scheduler for On-Demand Job Scheduling in High-Performance Computing Systems, CANDAR24.
- Ohmura et al.: A QA-Assisted Job Scheduler for Minimizing the Impact of Urgent Computing on HPC System Operation, PDAA24.
- Shubham et at.: Leveraging Hardware Performance Counters for Predicting Workload Interference in Vector Supercomputers, PDCAT2024.

タスク間の依存関係を考慮したワークフローのバッチジョブスケジューリング

柳井快斗，高橋慧智，下村陽一，滝沢寛之
東北大学

xSIG 2024; 8/7 xSIG-5 (13); 15:15-15:40

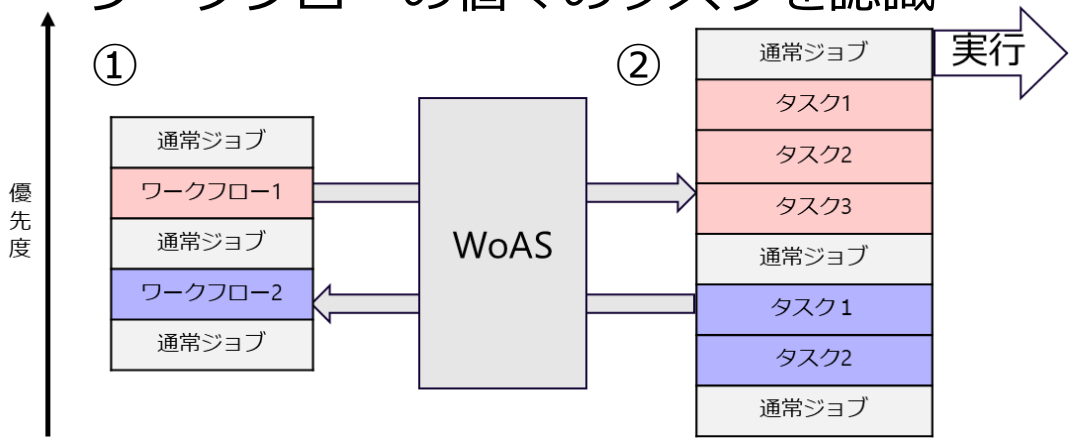
目的とアプローチ

- 目的
  - ワークフローと依存関係を持たないジョブ (通常ジョブ)が混在して実行されるHPCシステムで，計算資源を過剰に割り当てることなく，最初のタスクが投入されてから全てのタスクの実行が完了するまでの時間(ターンアラウンドタイム)を短縮

- アプローチ
  - ジョブ待ち行列内で通常ジョブ，ワークフローの実行優先度を制御

関連研究

- **WoAS (Workflow-Aware Scheduling System)**
  実行優先度を２つの待ち行列で管理
  - 優先順位付け
    - ワークフロー全体を１つのジョブとして認識
  - 計算資源の割り当て
    - ワークフローの個々のタスクを認識



待ち行列内でワークフロー内のタスクに同等の優先度を与え個別に計算資源割り当て

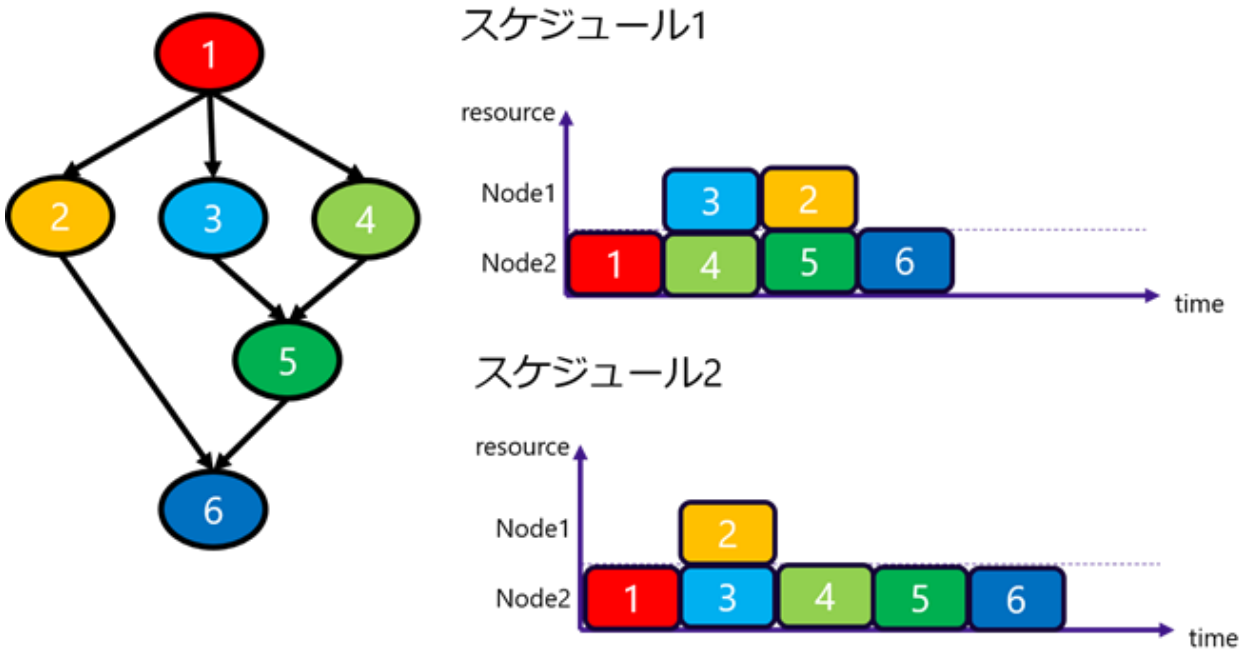
計算資源の無駄が減り，タスク間の待ち時間が短縮→ターンアラウンドタイムが短縮

Gonzalo P Rodrigo et al. Enabling workflow-aware scheduling on HPC systems. In Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing, pp. 3–14, 2017.

WoASの課題

- ワークフロー内のすべてのタスクに同等の実行優先度を割り当て
  - タスク間の実行順序の制御はされていない
  - 優先すべきタスクの実行が遅延

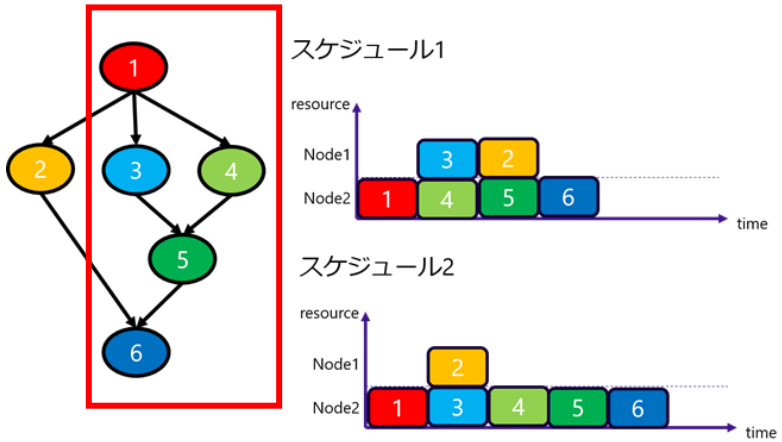
    → **ターンアラウンドタイムの増加**



タスクの実行順序が
ターンアラウンドタイムに影響

タスクの優先順位付け

- クリティカルパス上のタスクを優先的実行により実行時間が短縮
  - **クリティカルパス**：あるタスクから最後のタスクを実行する上で最も時間のかかる一連のタスク



クリティカルパス上のタスクを優先的に実行

クリティカルパス上のタスクを優先的に実行していない

- HPCシステムでは
  - 実行時間はユーザーから与えらる不正確なデータ
  - 使用可能な計算資源量が動的に変化

→正確なクリティカルパスを求めることは難しくどのタスクを優先すべきかが不明確

提案手法

- WoASを基にタスク間の依存関係を考慮して実行優先度を設定し，タスクの実行順序を制御

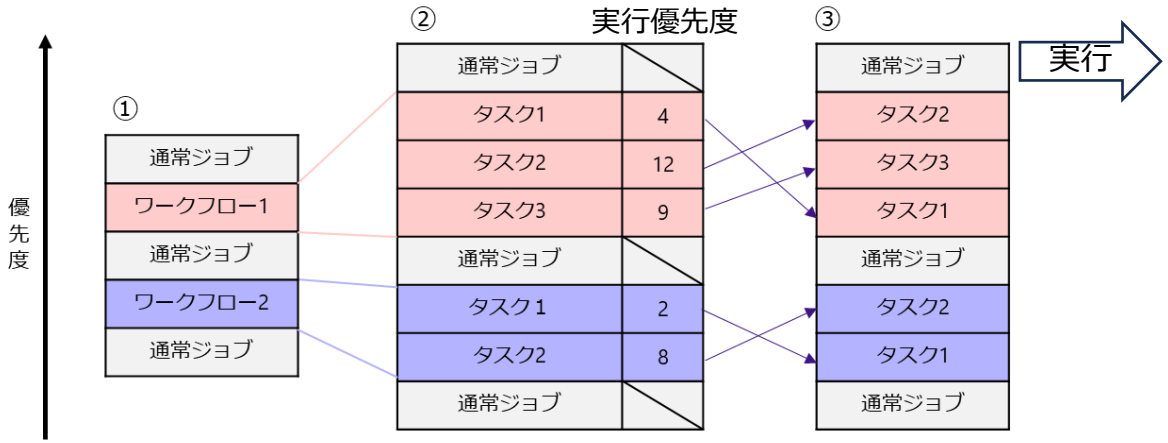
- 待ち行列のジョブ管理手法

① ワークフロー全体を一つのジョブとみて他のジョブとの間に優先順位付け

② 依存関係の満たされたタスクに実行優先度の設定

③ ワークフロー内で実行優先度に基づき並び替え

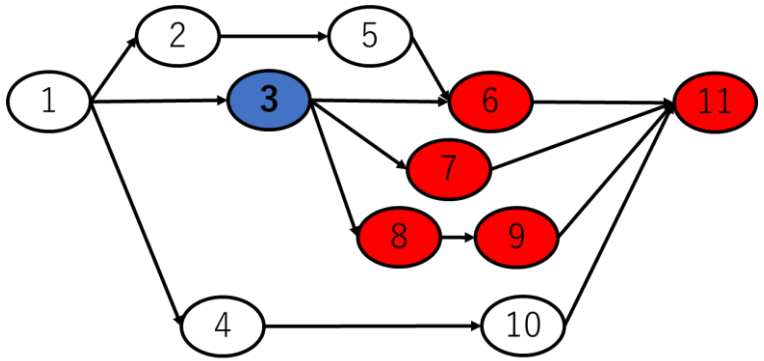
④ タスクごとに計算資源を割り当てて実行



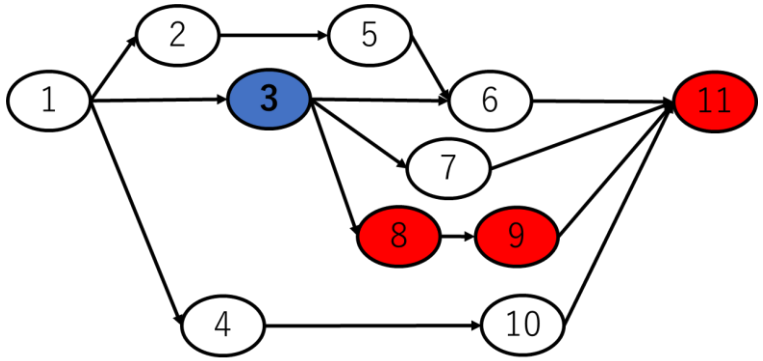
タスク間の実行優先度の設定

- ３つの設定方法
  - **all_path**　後続のタスクが多いタスクを優先
  - **all_time**　後続のタスクの実行時間の和が大きいタスクを優先
  - **path**　後続のクリティカルパス上のタスクが多いタスクを優先
    - クリティカルパスはタスク数に基づいて判断

all_path, all_time



path



タスク間の実行優先度の設定

- ３つの設定方法
  - **all_path**　後続のタスクが多いタスクを優先
  - **all_time**　後続のタスクの実行時間の和が大きいタスクを優先
  - **path**　後続のクリティカルパス上のタスクが多いタスクを優先

- all_timeは依存関係に加えて実行時間を考慮
  - 他の手法よりも高い性能が期待できる
  - HPCシステムにおいて実行時間はユーザーから与えられる情報
    - すべてのタスクの実行時間が正確に判明している状況は限定的

評価：比較手法

- **ワークフローのターンアラウンドタイム**をいくつかの手法で比較
- タスク間の優先順位付け
  - タスク間の依存関係の情報を用いた優先順位付け
    - **all_path**　：後続のタスクが多いタスクを優先
    - **all_time**　：後続のタスクの実行時間の和が大きいタスクを優先
    - **path**　：後続のクリティカルパス上のタスクが多いタスクを優先
  - 基準とする優先順位付け
    - **depth_time**　：依存関係が先に満たされたタスクを優先
  - 他の優先順位付け
    - **max_size**　：ノード時間積(実行時間×ノード数)の大きいタスクを優先
    - **min_size**　：ノード時間積の小さいタスクを優先
    - **random**　：ランダムに優先度を決定
- 他の手法
  - パイロットジョブ
  - チェーンジョブ

7種類の優先順位付けに2種類の手法を加えた計9つの手法間で比較

評価条件

- ジョブスケジューリングシミュレータによって評価

- シミュレーション環境
  - HPCクラスタ：64ノード
  - ジョブスケジューリングアルゴリズム
    - First Come First Serve (FCFS)：先に投入されたジョブを優先
    - EASY Backfilling
      - 待ち行列の先頭のジョブを遅延させない限り，後続のジョブが前方のジョブを追い越して実行

通常ジョブ

- ４つの通常ジョブセットを使用
  - 以下の表に従いジョブはランダムに生成
  - ポアソン過程に従い確率的に投入

	実行ノード数					実行時間(s)	到着率	特徴
	1	2	4	8	16			
ジョブセット1	60%	25%	5%	5%	5%	30～30000	0.0012	ノード数：小，利用率：高
ジョブセット2							0.001	ノード数：小，利用率：低
ジョブセット3	20%	20%	20%	20%	20%		0.000484	ノード数：大，利用率：高
ジョブセット4							0.0004	ノード数：大，利用率：低

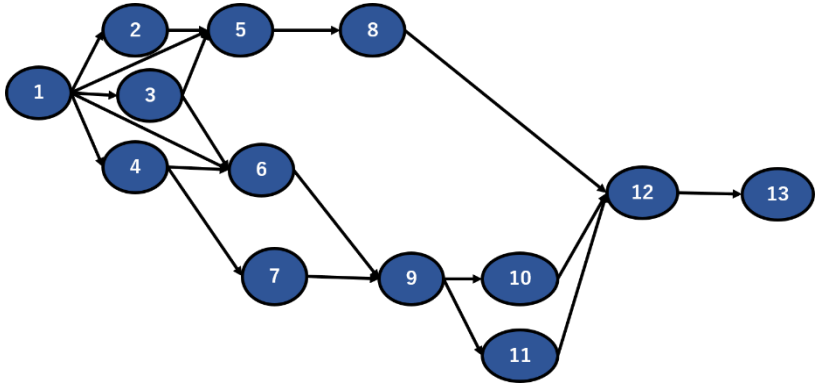
ワークフロー

- タスクを以下の表に従い実行時間，実行ノード数はランダムに生成

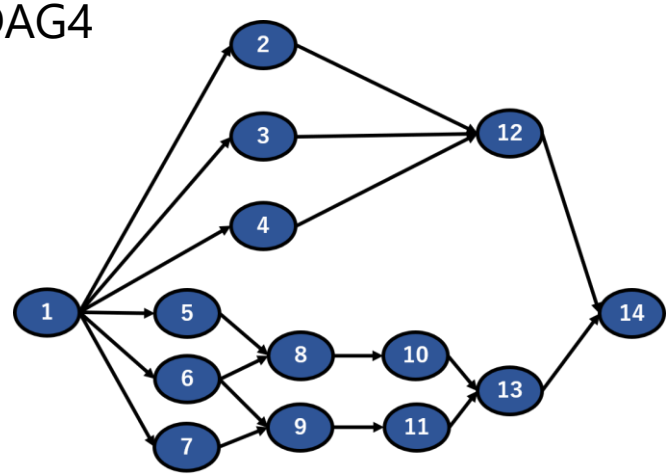
	実行ノード数					実行時間(s)
タスク	1	2	4	8	16	30～3000
	20%	20%	20%	20%	20%	

- 6種類のDAGを使用しDAGごとに50個のワークフローを生成，一定時間間隔で投入

DAG1



DAG4



評価結果

- depth_timeとのターンアラウンドタイムの比較（ジョブセット１）
  - 一対比較(下記の表は結果の一部)
  - better：短くなった回数，equal：変わらなかった回数，worse：長くなった回数

	all_path	all_time	path	depth_time	max_size	min_size	random	pilot	chain
better	**133**	**140**	**135**	0	91	80	64	59	15
equal	113	105	116	300	111	99	169	4	90
worse	**54**	**55**	**49**	0	98	121	67	237	195

all_path, all_time, pathでbetterがworseを大きく上回っている

クリティカルパスを考慮した提案手法が先行研究と比べ
ターンアラウンドタイムを短縮する割合が大きい

通常ジョブによる影響

- all_time とdepth_time間の比較

		ジョブセット1	ジョブセット2	ジョブセット3	ジョブセット4
all_time	better	**140**	**107**	**121**	**95**
	equal	105	171	146	183
	worse	**55**	**22**	**33**	**22**

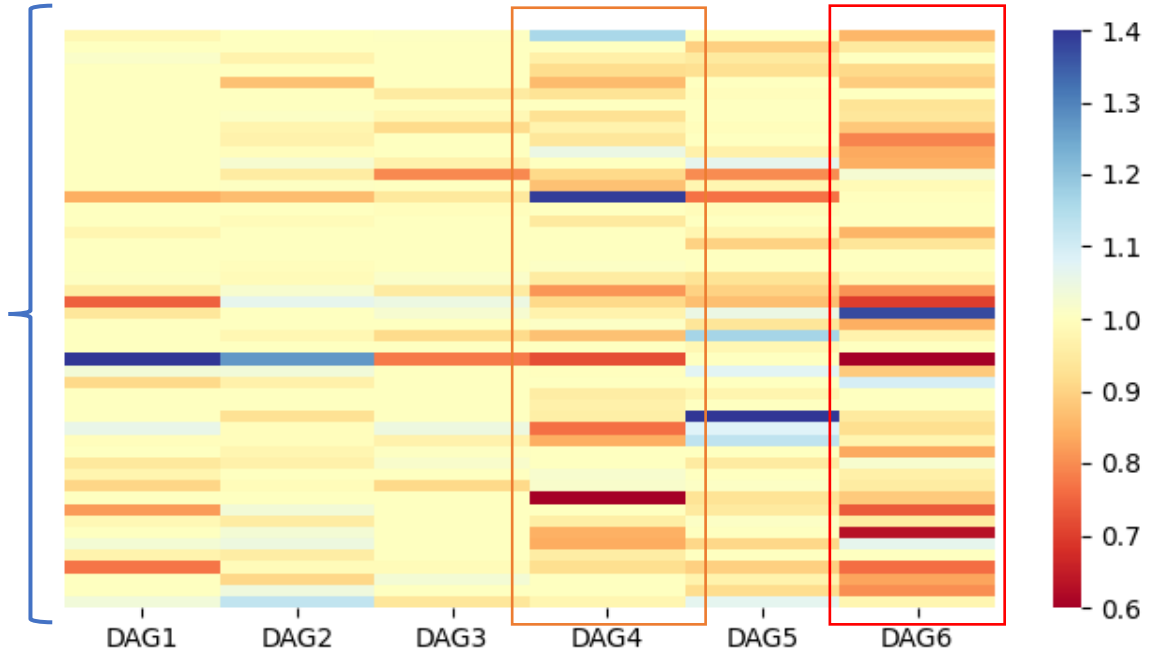
すべてのジョブセットでbetterがworseを上回っている

通常ジョブによらず一貫して高い性能が得られる

タスク間の依存関係による影響

DAGによって提案手法の効果が変化する

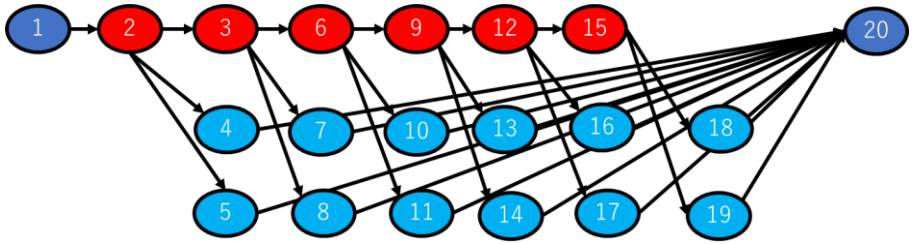
all_timeとdepth_timeの
ターンアラウンドタイムの比
赤いほどall_timeで短縮



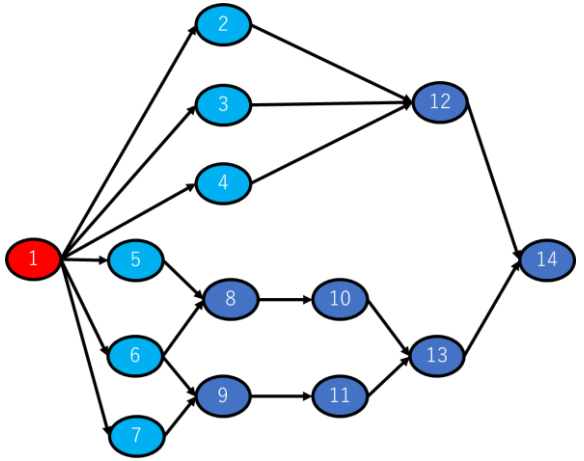
DAG6とDAG4で提案手法の効果がよく表れている

タスク間の依存関係による影響

- DAG6で効果が表れやすい要因
  - **クリティカルパス上のタスク**を実行しないと**多くのタスク**が実行できない
    →クリティカルパスを意識した提案手法の有効性が表れやすい



- DAG4で効果が表れやすい要因
  - **最初のタスク**の結果が**後続の6つのタスクで**必要
    →６つのタスクの実行順序の組み合わせが大きい
    →優先順位付けの効果が表れやすい



まとめ

- **目的**
  - ワークフローと通常ジョブが混在して実行されるHPCシステムで計算資源を過剰に割り当てることなく，ワークフローのターンアラウンドタイムを短縮

- **提案手法**
  - WoASを基にタスク間の依存関係を考慮して実行優先度を設定し．タスクの実行順序を制御

- **結論**
  - 提案手法は先行研究と比較してワークフローのターンアラウンドタイムを高い確率で短縮可能