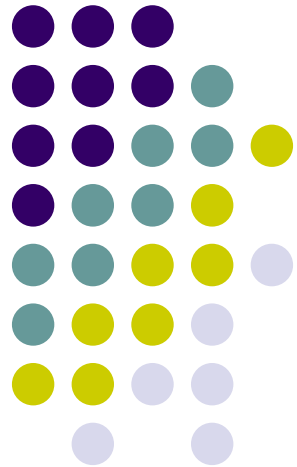
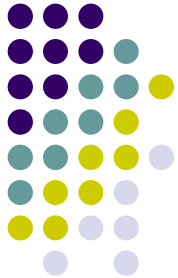
レプリカ管理システムを利用したデータインテンシブアプリケーション向けスケジューリングシステム

町田 悠哉[†]
滝澤　真一朗[†]
中田 秀基[††, †]
松岡 聡[†, †††]

†: 東京工業大学
††: 産業技術総合研究所
†††: 国立情報学研究所

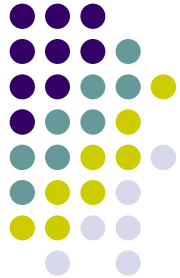


研究背景



- グリッド環境で扱われるデータサイズの大規模化
  - 物理学、天文学、バイオインフォマティクス、etc
  - 同一データセットを利用する均質なタスクの集合
  - バッチジョブとしてサブミット
    - バッチスケジューリングシステムによる実行マシンの決定
    - バッチスケジューリングシステムにおけるデータ利用法
      - 分散ファイルシステムを利用したデータ共有
      - データ転送機構を利用した単純なステージング

グリッド環境での大規模データ処理

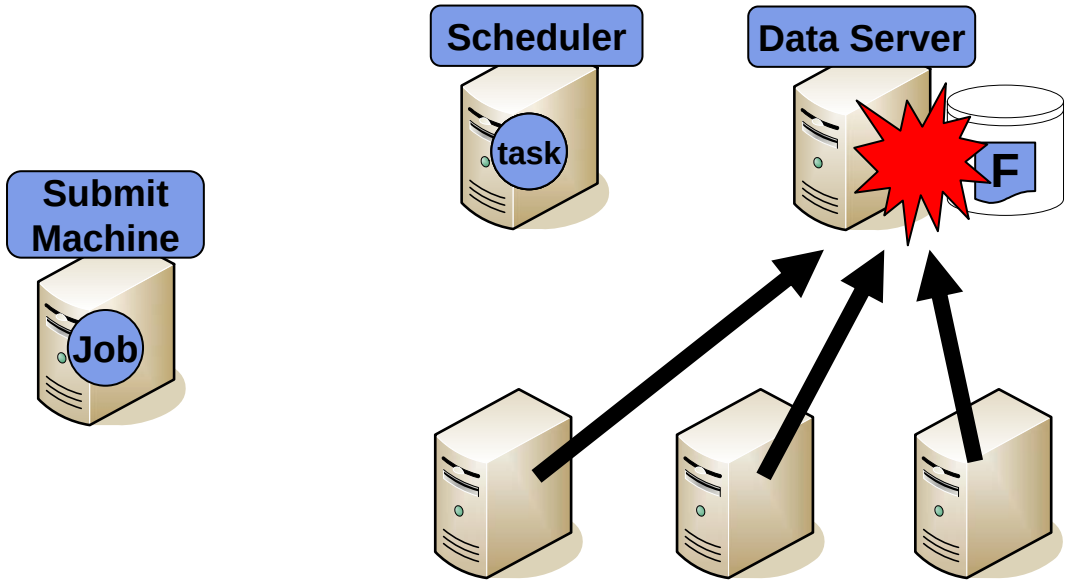


- グリッド環境での大規模データ処理シナリオ
  - データ転送ツールを利用したデータステージング
    - データ転送元・転送先の選択
      - ジョブサブミット時にユーザが決定
      - レプリケーションアルゴリズムにより決定
  - ユーザがサブミットしたデータ解析ジョブをスケジューリングシステムが実行マシンを決定
    - 分散ファイルシステム(NFS, AFS, etc)によるデータ共有
    - 転送ツール(GridFTP, Stork, etc)によるステージング

⇒ データ転送とジョブスケジューリングが独立

問題点 - 共有手法

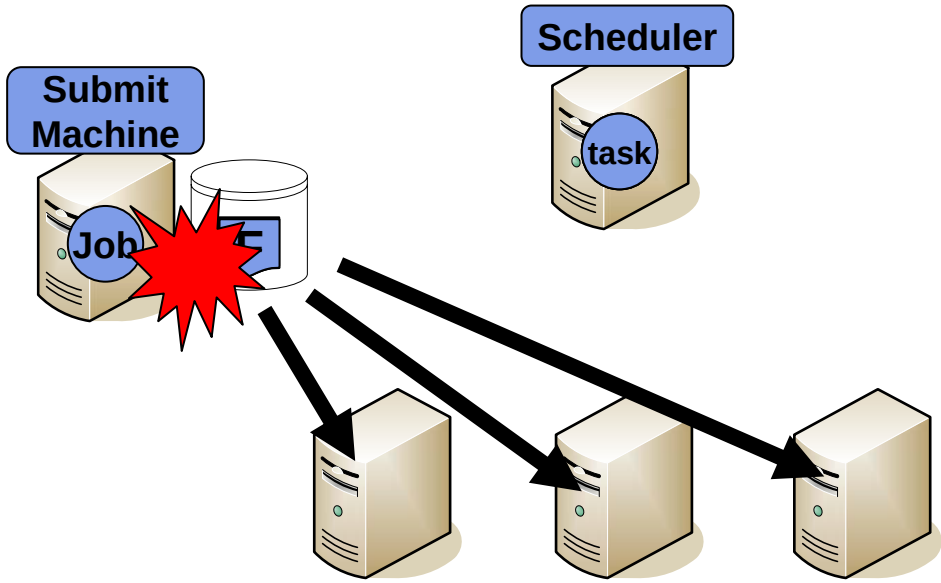
- 分散ファイルシステムなどを利用したデータ共有



I/Oノードにおいてアクセス集中が発生

問題点 - ステージング

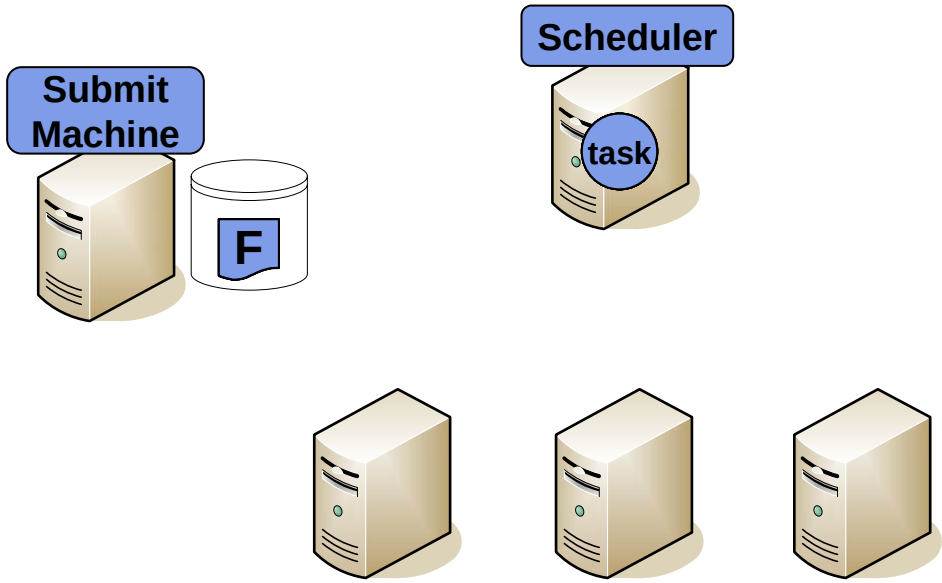
- データ転送機構を利用した単純なステージング



I/Oノードにおいてアクセス集中が発生

問題点 - ステージング

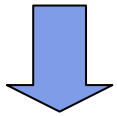
- データ転送機構を利用した単純なステージング



同一データの非効率な転送が発生

問題点 - データの複製

- データレプリケーション
  - データの複製(レプリカ)を作成してアクセスを分散
  - ポリシーに応じたレプリカの作成・削除



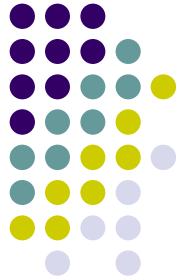
  - 1対1転送を想定しているためアクセス集中発生
  - ジョブスケジューリングとは独立なレプリケーション

データインテンシブアプリケーションを効率的に
実行するためのシステムとして十分ではない

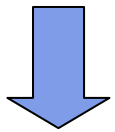
研究目的と成果

- 研究目的
  - グリッド環境でデータインテンシブアプリケーションを効率的に実行するためのスケジューリングシステムの構築
- 研究成果
  - バッチスケジューリングシステムを拡張し、レプリカ管理システムと連動したシステムを構築
  - 従来のシステムよりも効率的にデータインテンシブアプリケーションを実行できることを確認

関連研究 - Stork[Kosarら, ’04]

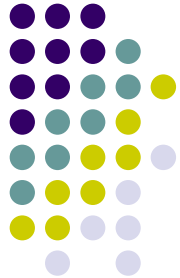


- データ転送用スケジューリングシステム
- Condor[Livnyら, ’88]と連動してデータインテンシブアプリケーションを実行
  - DAGManがジョブの依存関係を解決
    - 計算ジョブはCondorにサブミット
    - データ転送ジョブはStorkにサブミット



転送元・転送先が静的に決定されるためアクセス集中の回避は困難

関連研究 - BAD-FS[Bentら, '04]

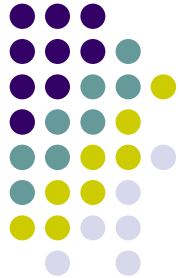


- ストレージのコントロールをスケジューリングシステムにエクスポーズ
  - データインテンシブアプリケーションを効率的に実行
    - WAN上のファイル転送の最小化
  - 各タスクが必要とするデータおよびデータロケーションを記述する必要あり

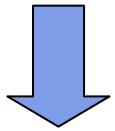
  - ユーザの負荷が大きい
  - 利用されるデータはユーザが静的に決定

```
job      a    a.condor
job      b    b.condor
job      c    c.condor
job      d    d.condor
parent   a    child   b
parent   c    child   d
volume   b1   ftp://home/data  1 GB
volume   p1   scratch  50 MB
volume   p2   scratch  50 MB
mount    b1   a  /mydata
mount    b1   c  /mydata
mount    p1   a  /tmp
mount    p1   b  /tmp
mount    p2   c  /tmp
mount    p2   d  /tmp
extract  p1   x  ftp://home/out.1
extract  p2   x  ftp://home/out.2
```

関連研究 - [Ranganathanら, ’02]



- スケジューリングとレプリケーションの分割
  - 各ファイルへのアクセス数をカウント
  - アクセス数に応じてレプリカ作成・削除



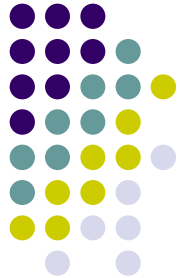
  - 単一データへのアクセス集中発生
  - スケジューリングとの連動なし
    - データのレプリケーション先は実行マシンのロケーション考慮されず

提案手法

- ジョブスケジューリングとレプリカ管理をタイトに結合
  - 最適なデータレプリケーション
    - ジョブ実行マシンへのレプリケーション
  - データロケーションに応じたスケジューリング
    - 同一データの反復転送回避
- 同一データのアクセス集中の回避
  - 同一データ転送リクエストの集約
- 計算資源の遊休時間の削減
  - 計算とデータ転送の同時実行

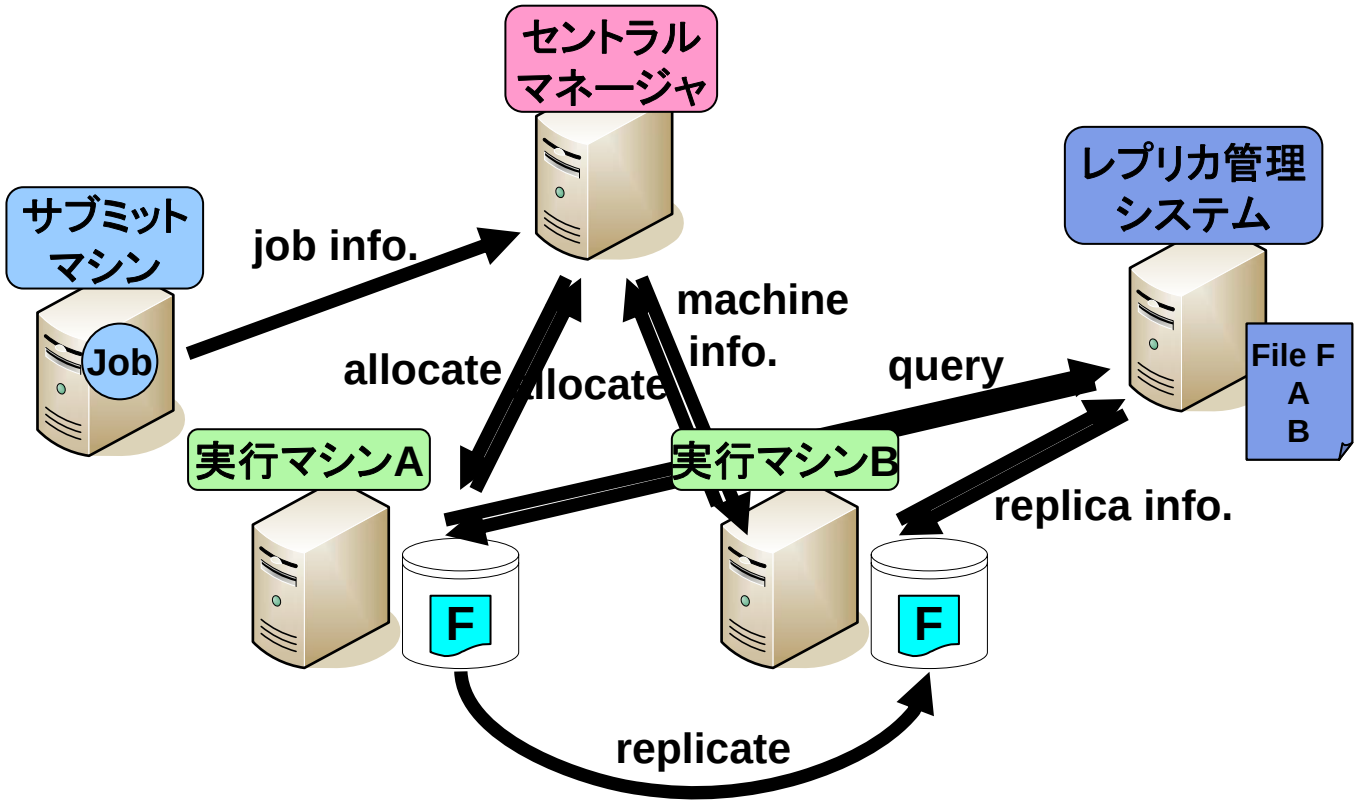
⇒ システム全体の資源利用効率が上昇

提案システムの設計



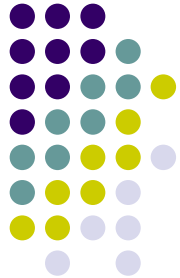
- レプリカ情報を加味したジョブスケジューリング
  - レプリカ保持ノードへ優先的にスケジューリング
    → データの再利用性の向上
  - 転送コストの低いノードへスケジューリング
    → 遊休時間の最小化
- 同一データの転送要求の集約
  - 近隣ノードの中で代表ノードのみがオリジナルファイルを取得
    → WAN上のデータ転送を最小限に抑制
- 計算資源の遊休時間の削減
  - 計算中にデータ転送を実行
  - データ転送中にジョブ実行→サスペンド機構が必要

提案システムの概要



レプリカ管理システムとの連動によりデータの再利用性の向上・アクセス集中の回避に

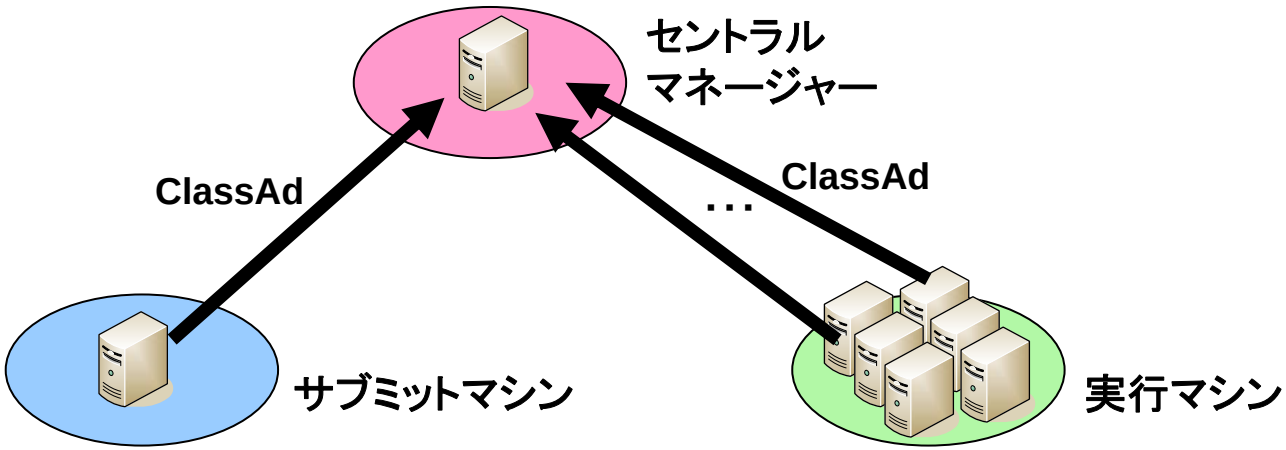
プロトタイプシステムの実装



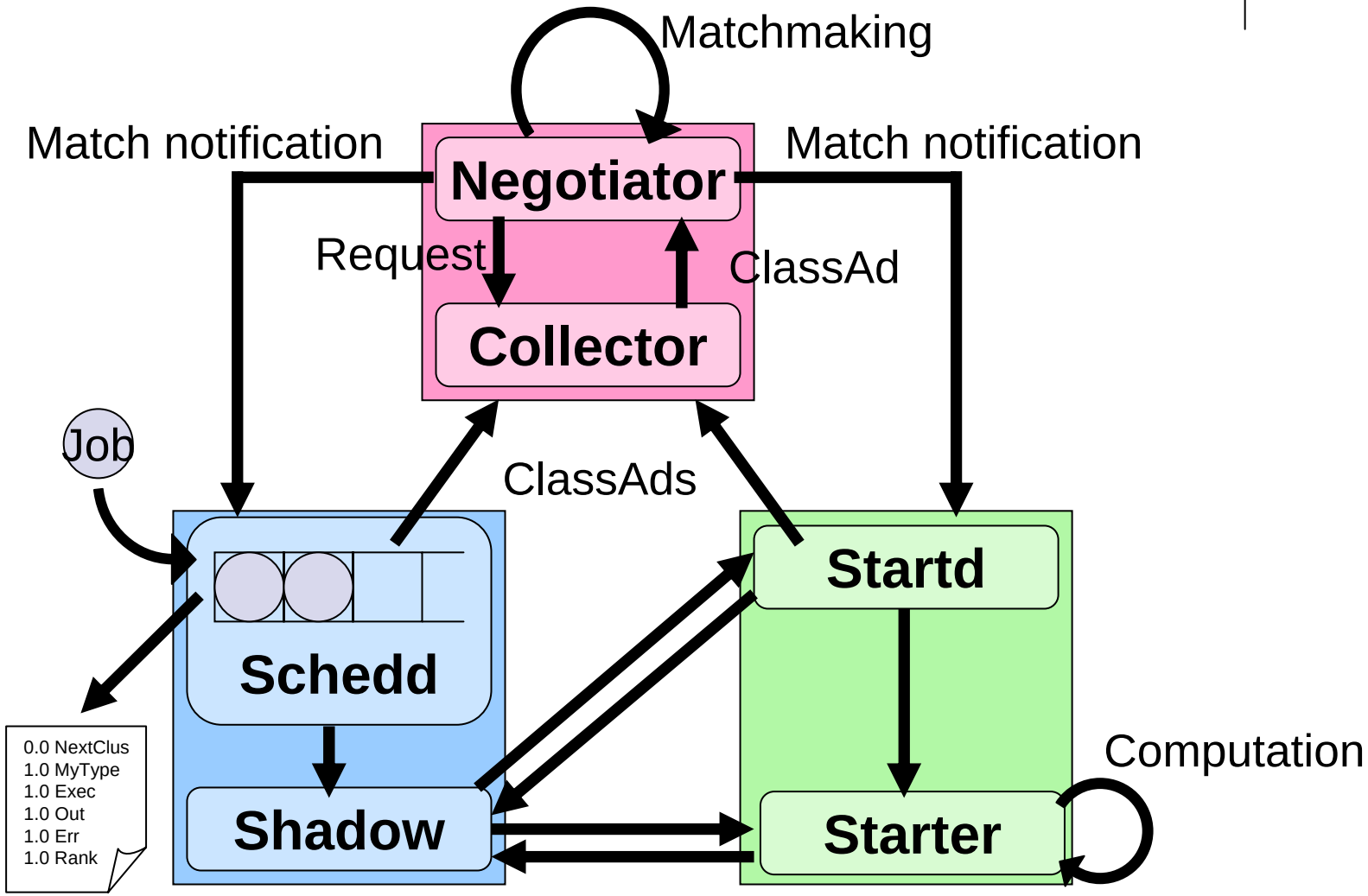
- 以下のコンポーネントを統合
  - バッチスケジューリングシステムJay
    - Condorを規範としたシステム
    - GSI[Fosterら, ’98]を利用したセキュアなシステム
  - マルチレプリケーションフレームワーク MultiReplication[Takizawaら, ’05]
    - レプリカロケーションサービス(RLS)
      - レプリカの位置情報を管理
    - アプリケーションレベルマルチキャスト転送
      - Dolly+[真鍋, ’01]によりO(1)の転送時間

プロトタイプシステムの実装

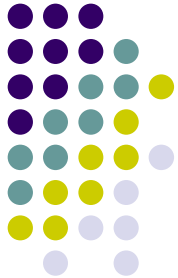
- 容易に拡張可能なバッチスケジューリングシステムを実装し、レプリカ管理との連動のために拡張
  - バッチスケジューリングシステムJay[町田ら, ’04]
    - Condorを規範とした容易に拡張可能なシステム
    - セキュリティ基盤にGSI[Fosterら, ’98]を利用
  - 複製管理システムMultiReplication[Takizawaら, ’05]



Jayシステムの概要


Matchmaking
Match notification
Match notification
Negotiator
Request
ClassAd
Collector
Job
ClassAds
Schedd
Startd
0.0 NextClus
1.0 MyType
1.0 Exec
1.0 Out
1.0 Err
1.0 Rank
Shadow
Starter
Computation

Jayのスケジューリング



- 受信したマシンとジョブのClassAd[Livnyら, ’97]の中からマッチメイキング[Ramanら, ’98]により最適なマシンとジョブの組み合わせを決定

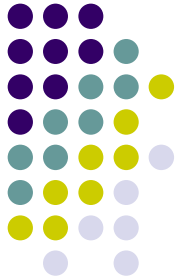
マシンのClassAd

```
MyType = “Machine”
TargetType = “Job”
Memory = 256
Arch = “INTEL”
OpSys = “LINUX”
Requirements =
        (Owner == “smith”)
```

ジョブのClassAd

```
MyType = “Job”
TargetType = “Machine”
Cmd = “sim”
Owner = “smith”
Args = “900”
Out = “sim.out”
Rank = Memory
Requirements =
        (Arch == “INTEL”) &&
        (OpSys == “LINUX”)
```

レプリカ管理システムとの連動



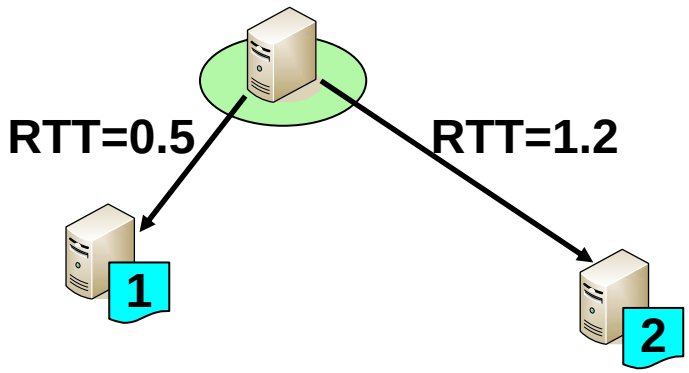
- 実行マシンはセントラルマネージャに送信するマシン情報にレプリカ情報を追加
  - 保持するレプリカファイルの論理名
  - 保持しないファイルのレプリケーションコスト
- マッチメイキング[Ramanら, '98]時にrank値にレプリカのロケーションに応じた値をプラス

```
executable = application
input      = input.$(Process)
output     = output.$(Process)
error      = error.$(Process)
arguments  = $(Replica_Files)
transfer_replica_files = data1, data2
queue 100
```

レプリカ管理システムとの連動

- 実行マシンはセントラルマネージャに送信するマシン情報にレプリカ情報を追加
  - Startdが定期的にどのくらい低コストでレプリカ作成できるかを表す値(ReplicaValue)をチェック
    - レプリカを作成するのに必要なコストに反比例
    - 現在の実装ではレプリカ作成コストとしてRTT値を使用

```
MyType = “Machine”
TargetType = “Job”
Memory = 256
Arch = “INTEL”
OpSys = “LINUX”
ReplicaInfo = “data1,500,
                    data2,294, …”
```



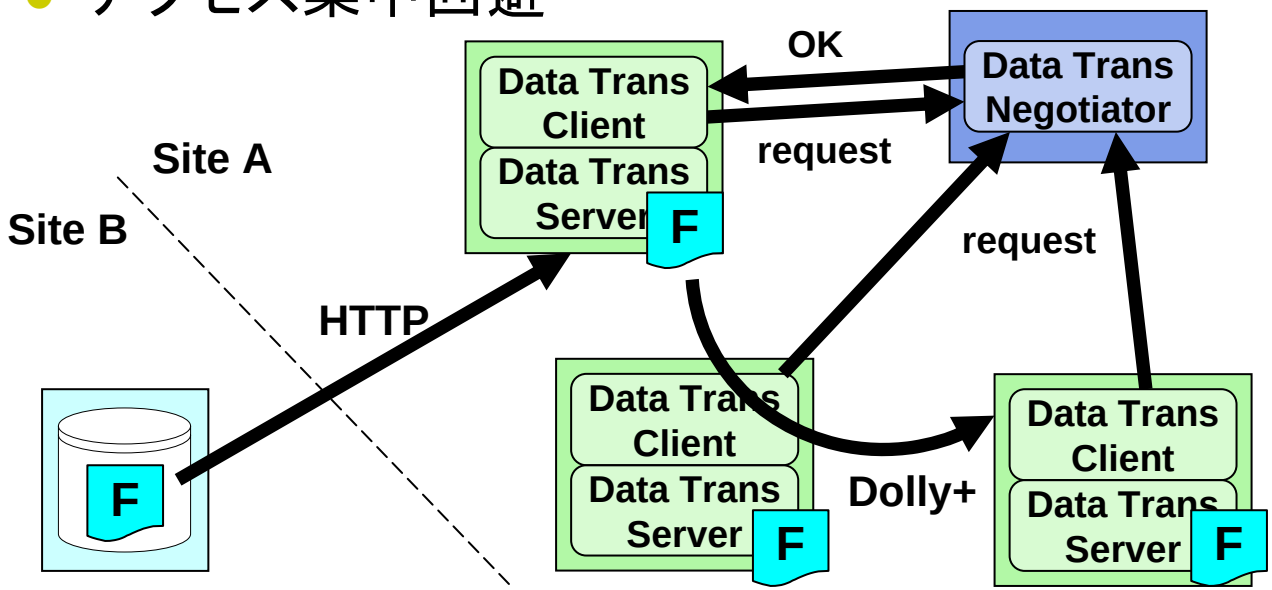
本システムのスケジューリング

- 受信したマシンとジョブのClassAdの中からマッチメイキングにより以下の値が最大となる最適なマシンとジョブの組み合わせを決定
  - マシンのRank値＋ジョブのRank値
    ＋(Σ ReplicaValue(i)) / N
  - ユーザが記述するサブミットファイル

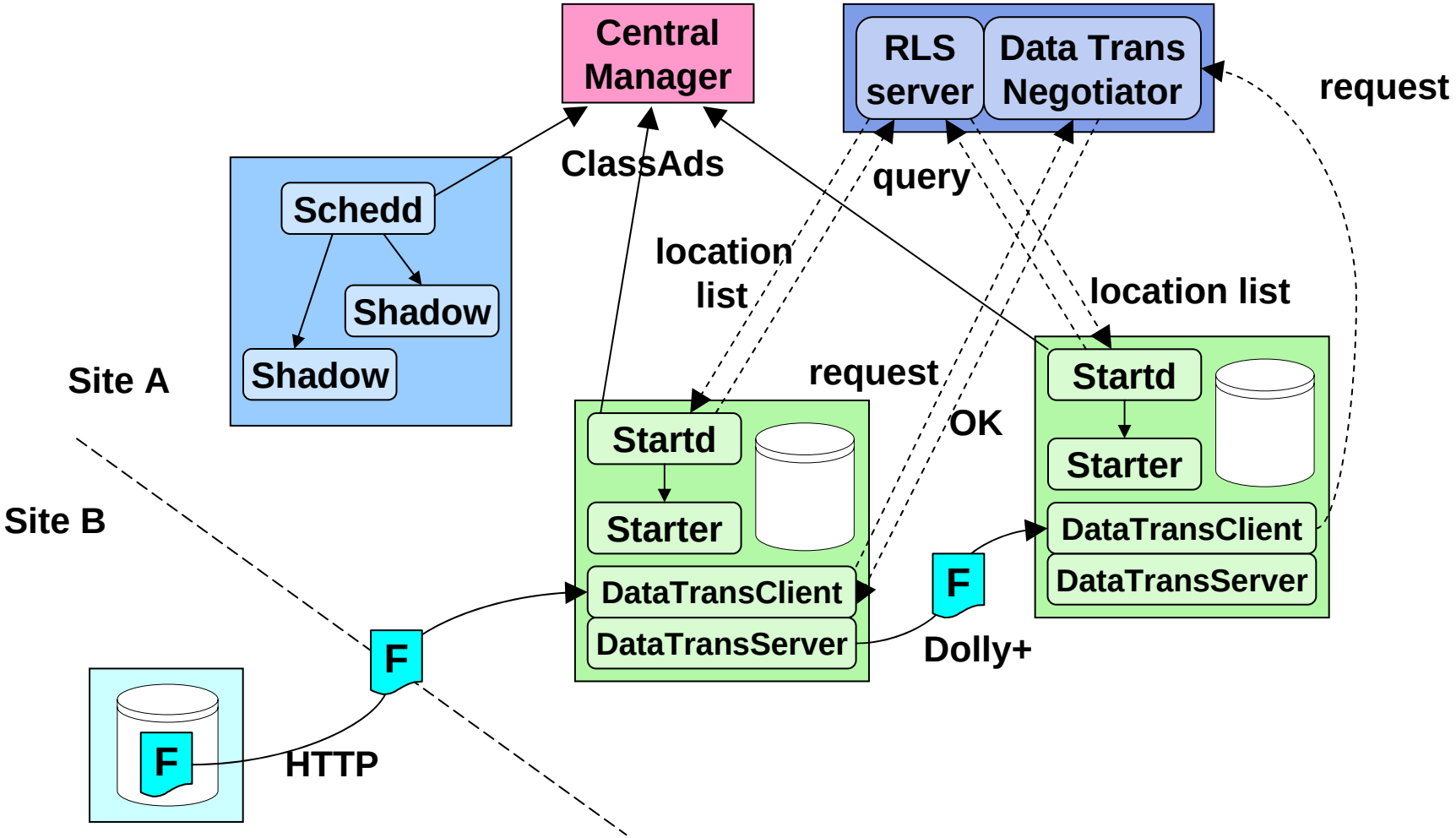
```
executable = application
input      = input.$(Process)
output     = output.$(Process)
error      = error.$(Process)
arguments  = $(Replica_Files)
transfer_replica_files = data1, data2,…, dataN
queue 100
```

レプリカ管理システム

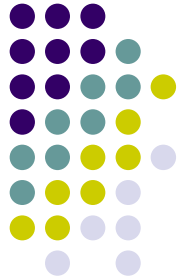
- MultiReplication[Takizawaら, ’05]を利用
  - RLS、転送機構、レプリカセレクター
    - レプリカ選択の指標としてRTTを使用
  - サイト内ではDolly+[Manabe, ’01]による転送
    - ノード数に対してO(1)の転送時間
    - アクセス集中回避



システム全体図


Central Manager
RLS server
Data Trans Negotiator
request
ClassAds
query
Schedd
Shadow
Shadow
location list
location list
Site A
request
OK
Startd
Starter
DataTransClient
DataTransServer
Site B
F
F
Dolly+
F
HTTP

評価実験



- サンプルアプリケーション
  - BLAST(http://www.ncbi.nlm.nih.gov/BLAST)
    - 核酸・タンパク質の相同性検索ツール
    - クエリに類似した核酸・タンパク質の配列をデータベースから検索→クエリの性質を調査

- 実験概要
  - データベースntに対して5つの核酸配列をクエリとしてBLASTを実行するジョブ
  - ジョブを5n(n = 4, 8, 16, 32)個サブミット
  - 実行マシンはnノード

評価手法

- 以下の4手法を比較
  - 共有手法
    - NFSによりデータベースを共有
  - ステージング手法
    - scpによりステージング
  - 提案手法
    - レプリカ管理システムと連携
  - 理想状態
    - すべての実行マシンにデータベースを格納

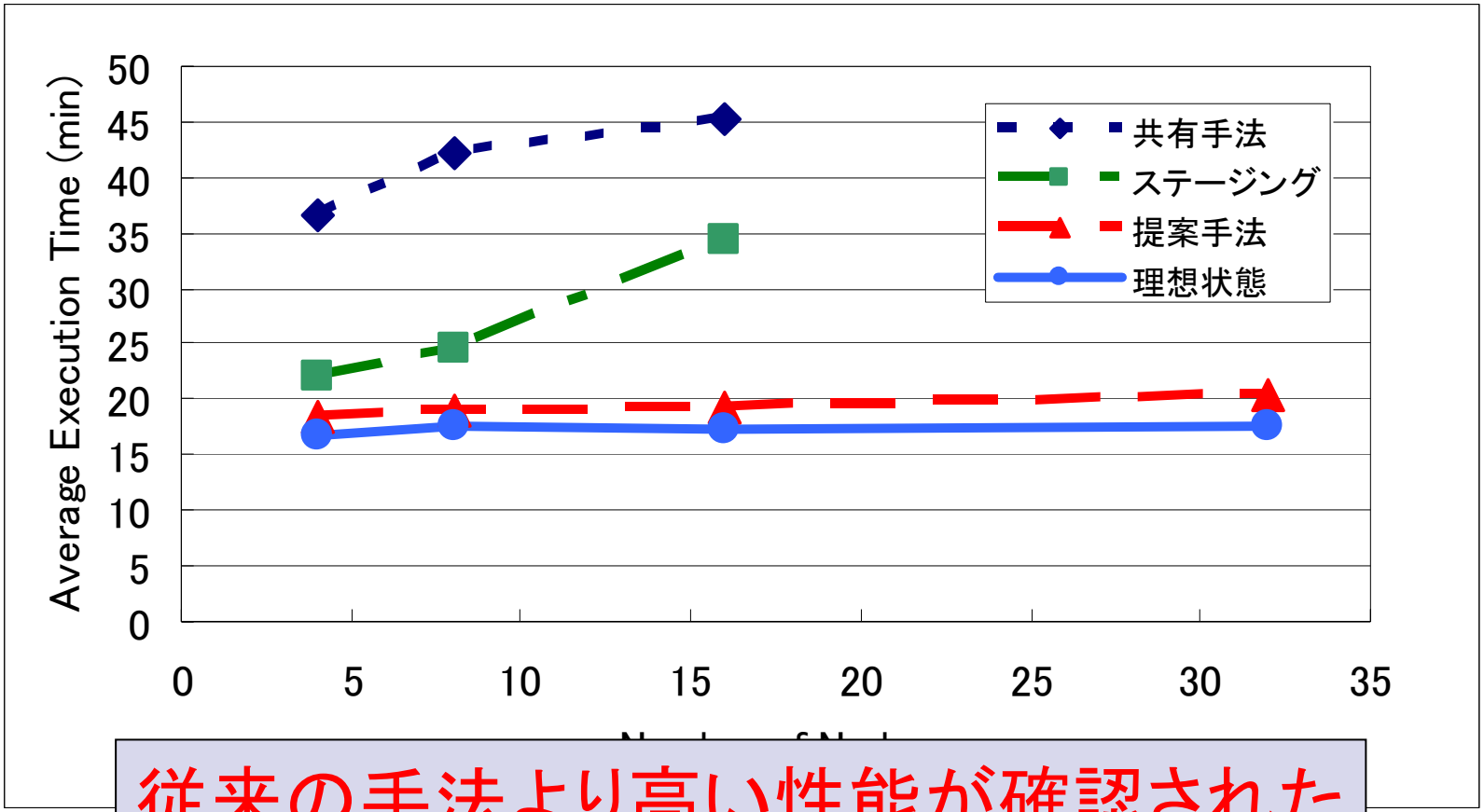
評価環境

- 松岡研究室PrestoIIIクラスタ

CPU	Opteron 242
Memory	2GBytes
OS	Linux 2.4.27
Network	1000Base-T

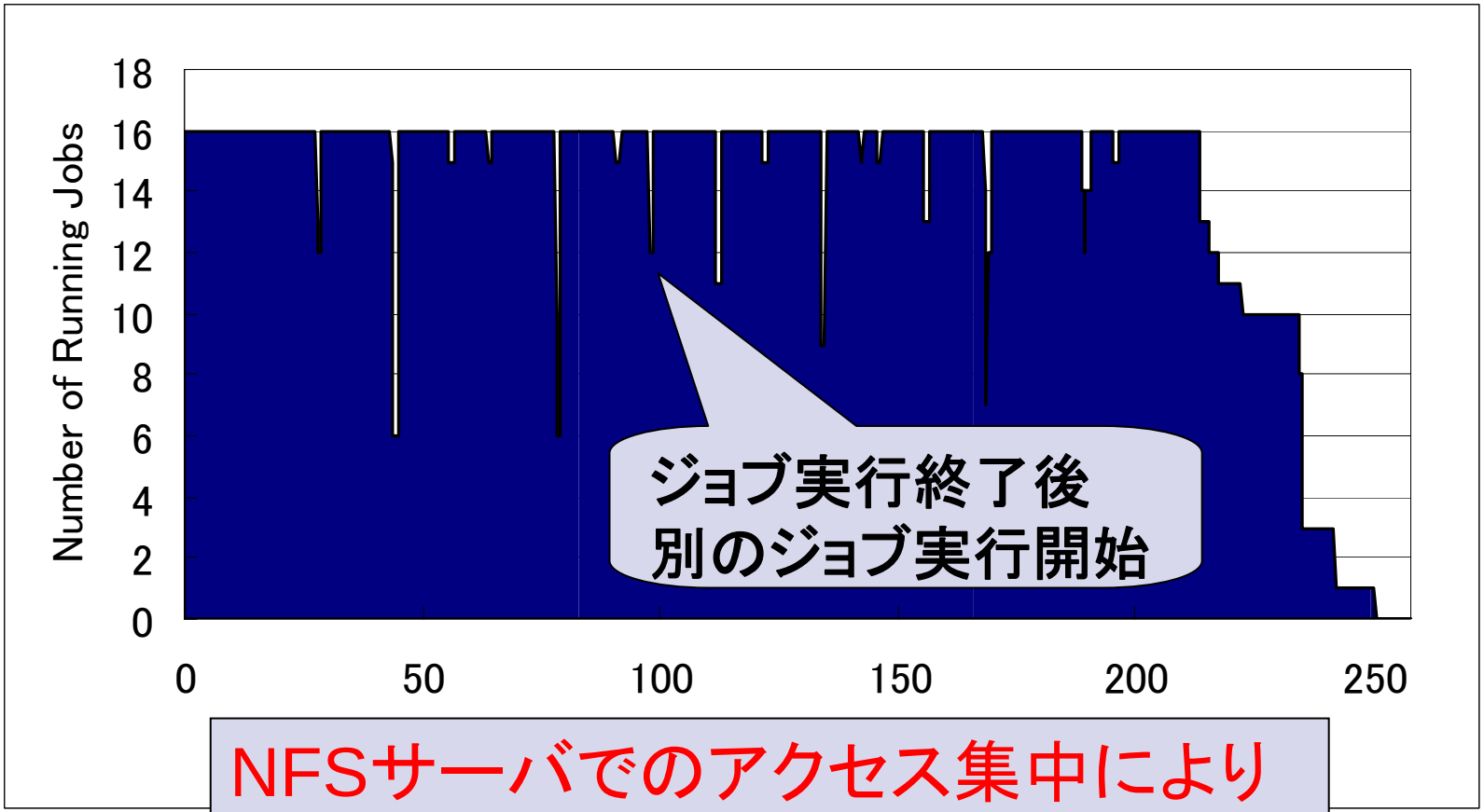
- セントラルマネージャ、サブミットマシン、RLSサーバ、NFSサーバについても上記スペック

平均実行時間の比較



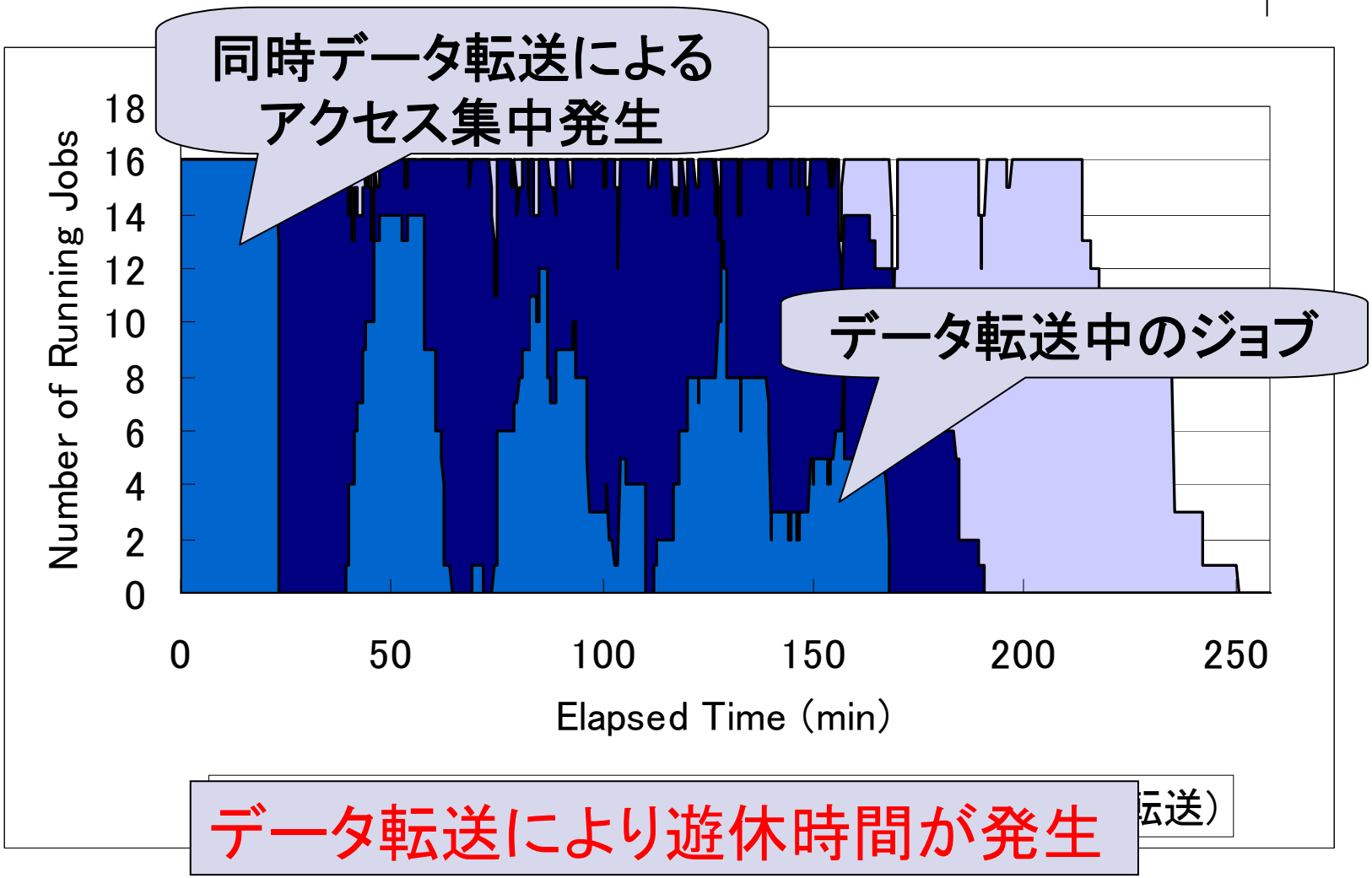
従来の手法より高い性能が確認された

ジョブの稼働状況 - NFS



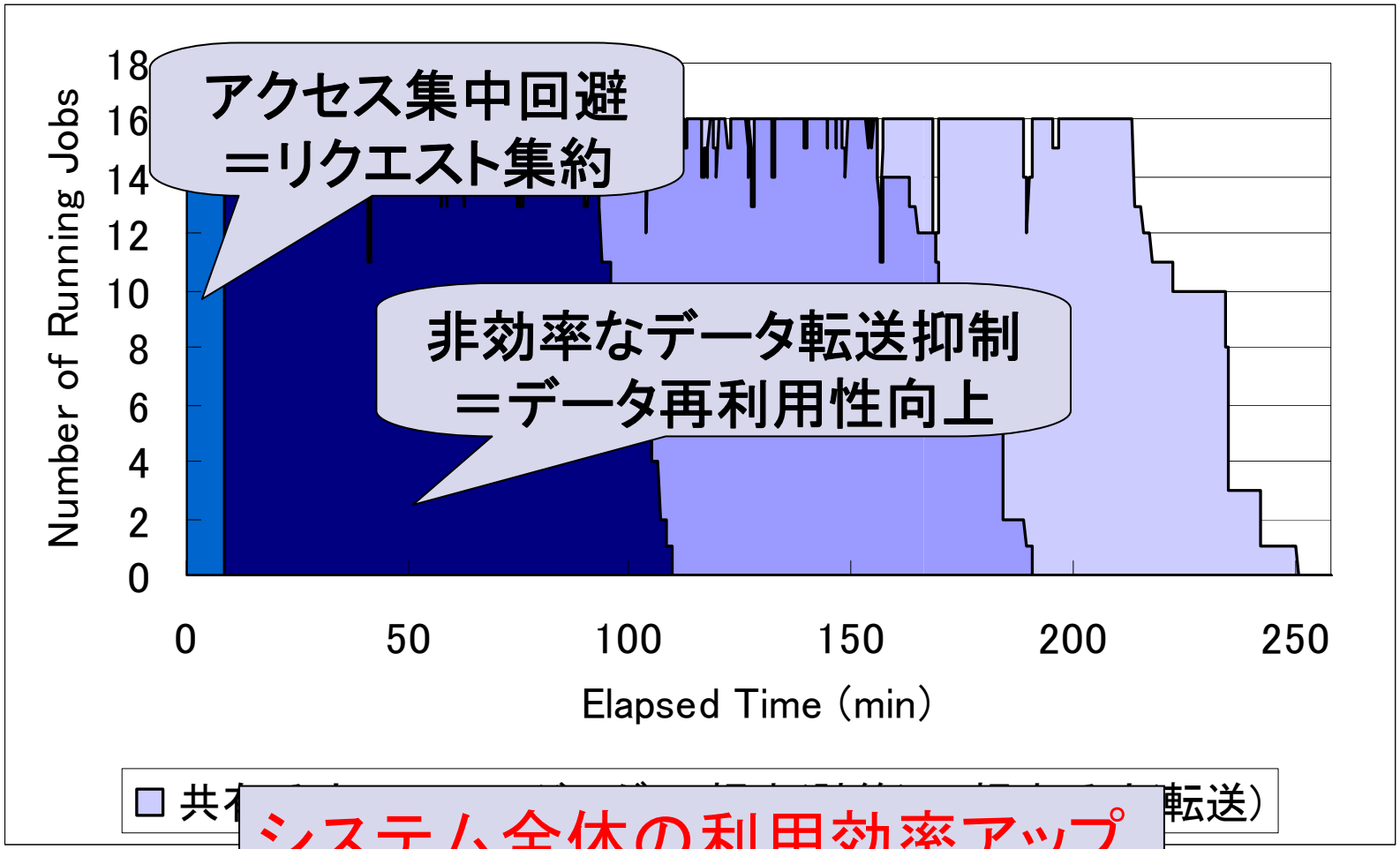
NFSサーバでのアクセス集中により
パフォーマンスの低下が発生

ジョブの稼働状況 - ステージング



データ転送により遊休時間が発生

ジョブの稼働状況 - 提案手法

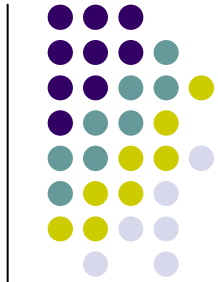


システム全体の利用効率アップ

考察

- 共有手法(NFS)
  - データサーバにアクセスが集中
- ステージング手法
  - ステージング元マシンにアクセス集中
  - データ転送によるアイドル時間発生
- 提案手法
  - 理想状態に近い性能を達成
  - システム全体の利用効率上昇
    - レプリケーションリクエストの集約
    - レプリカファイルの再利用

まとめと今後の課題



- まとめ
  - バッチスケジューリングシステムJayを拡張しレプリカ管理システムと連動したスケジューリングを実現
  - 従来の手法より効率的な環境を構築

- 今後の課題
  - 単一マシン上での計算ジョブとデータ転送の同時実行によるシステム全体の利用効率の向上
  - 複雑なシナリオを用いた大規模環境での評価実験