

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 真一郎[†]

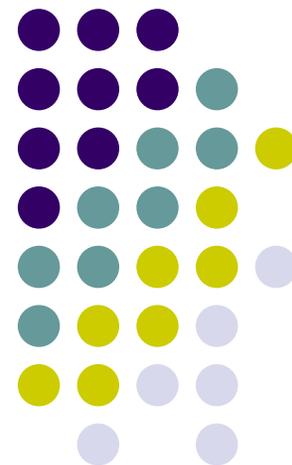
中田 秀基^{††, †}

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

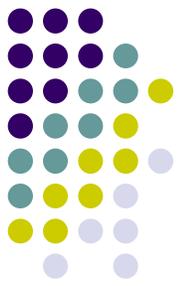




研究背景

- グリッド環境で扱われるデータサイズの大規模化
 - 高エネルギー物理学、天文学、バイオインフォ、etc
 - e.g.) CERNのLHC計画[<http://lhc.web.cern.ch/lhc/>]
 - LHC加速器を用いた陽子衝突実験('07稼働予定)
 - 毎年ペタバイト級のデータを生成
 - 解析には強力な計算能力が必要
 - 世界中に分散したリソースを利用

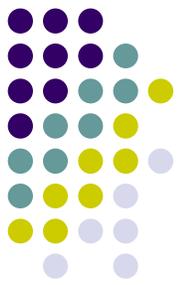
➡ 計算資源とデータの効率的な管理が必要



グリッド環境での大規模データ処理

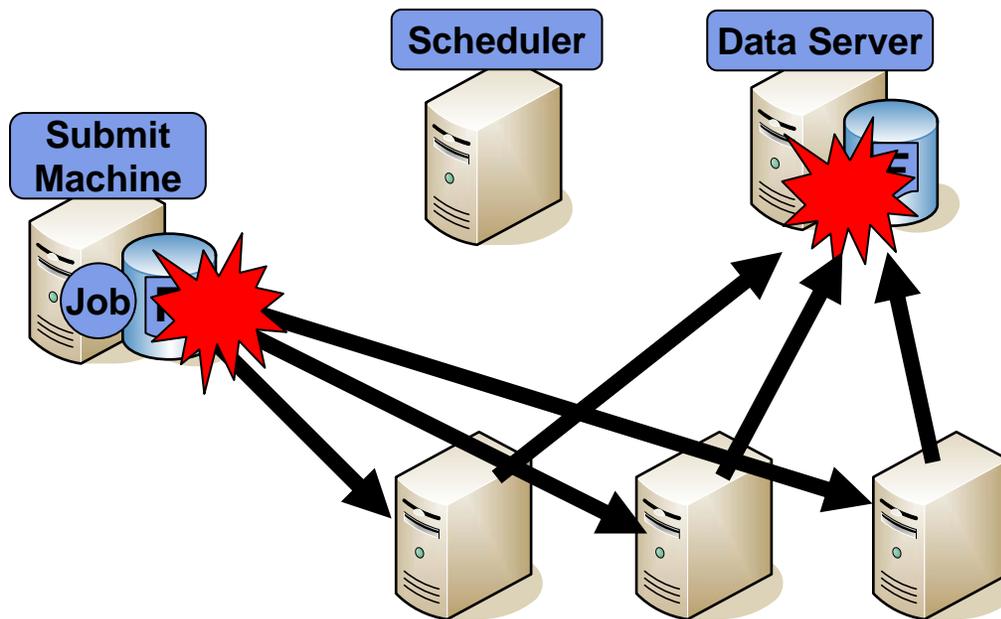
- グリッド環境でのデータインテンシブジョブ実行
 - 観測された大規模データの解析
 - バッチスケジューリングシステムによる実行マシン決定
 - 分散ファイルシステム(NFS, AFS, etc)によるデータ共有
 - 転送ツール(GridFTP, Stork, etc)によるステージング
 - ユーザが転送ノードを指定
 - 同一データセットを利用する均質なタスクの集合
 - 解析時のパラメータなどを変更

➡ 従来のデータインテンシブ実行環境には問題点が

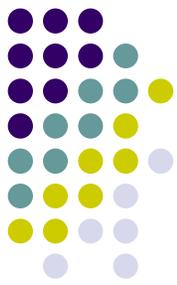


問題点

- 分散ファイルシステムなどを利用したデータ共有
- ステージングによるデータの利用



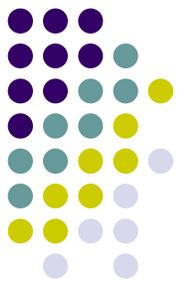
I/Oノードにおいてアクセス集中が発生



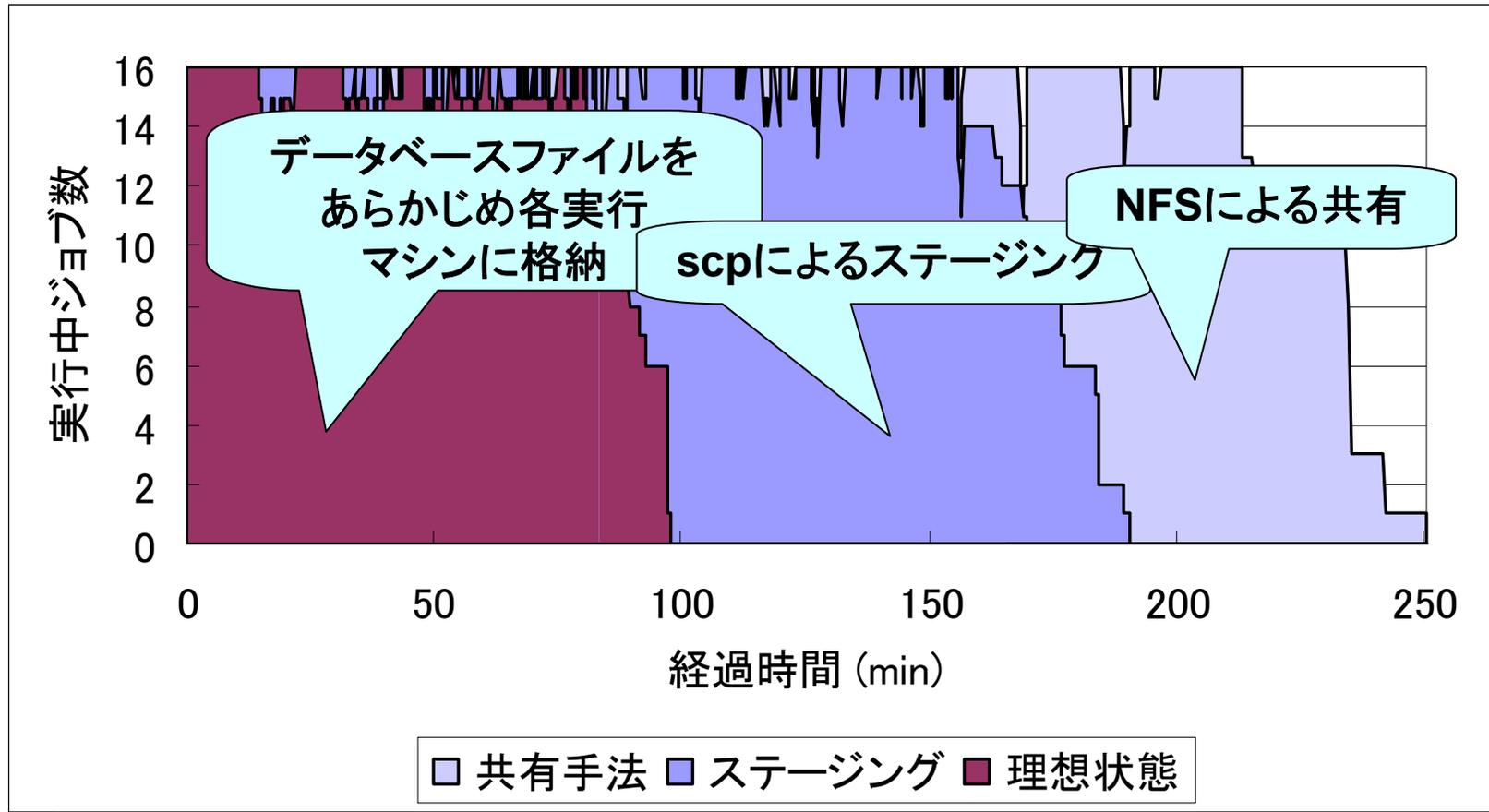
大規模データ処理の問題点

- 計算資源の利用効率を評価
 - 核酸・アミノ酸の相同性検索ツールBLAST[NCBI]
 - クエリに類似した配列をデータベースから検索
 - 5クエリの検索を行うジョブを80個サブミット
 - 約3GBのデータベースを使用
 - 20分弱の計算時間が必要
 - PrestoIIクラスタの16ノードを実行マシンとして使用
 - 1ノードあたり平均5ジョブを実行

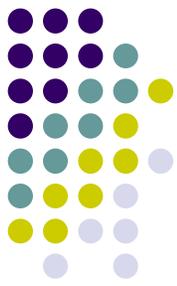
CPU	Opteron 242
Memory	2GBytes
OS	Linux 2.4.27
Network	1000Base-T



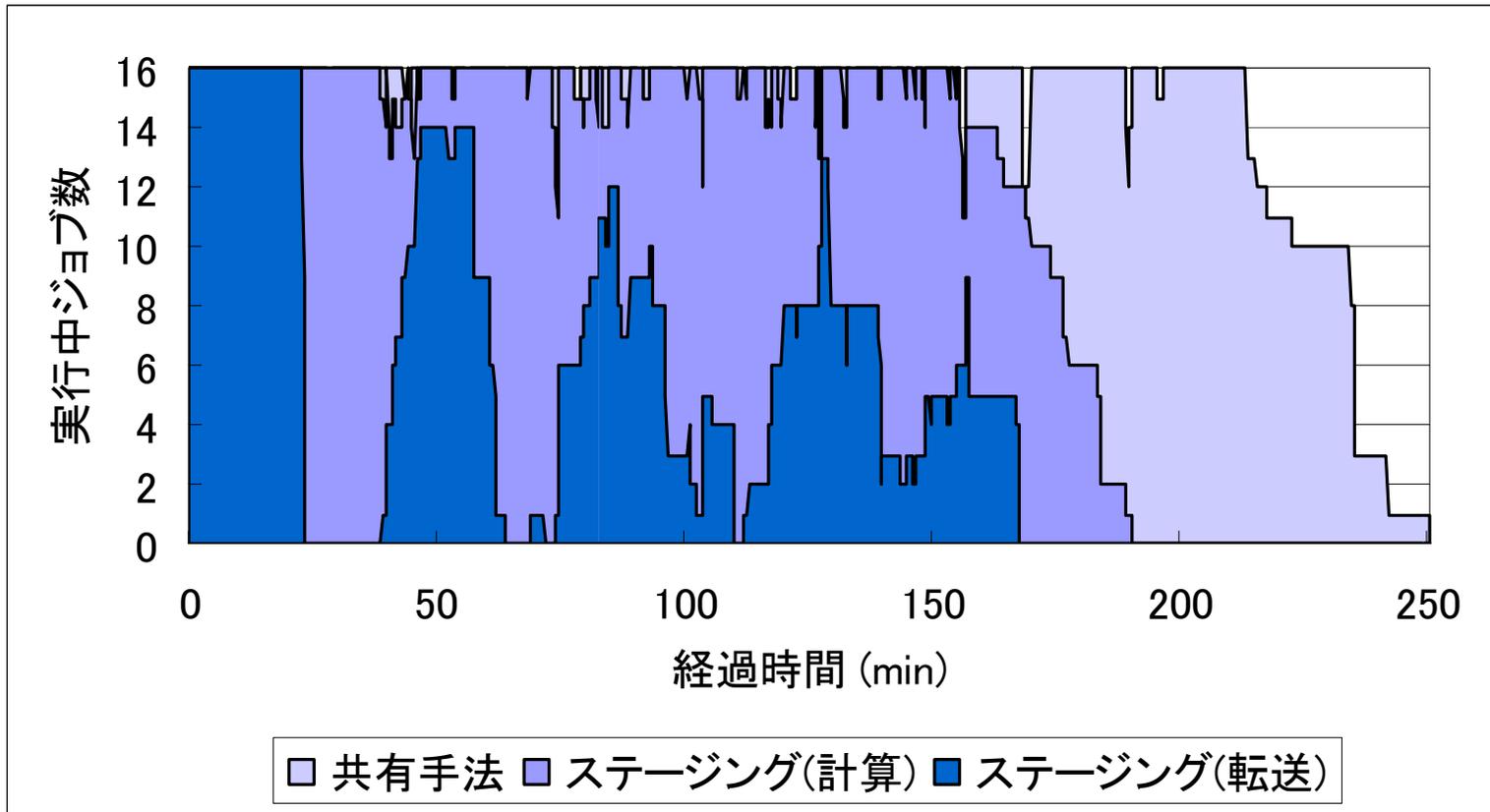
実行中ジョブ数の推移



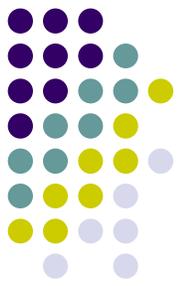
理想状態と比較して大きくパフォーマンスが低下



実行中ジョブ数の推移-ステージング



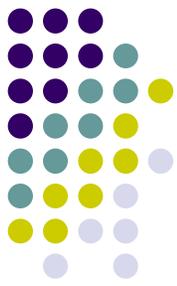
データ転送により遊休時間が発生



問題点 – レプリカによるアクセス分散

- 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
- ↓
- ユーザによるレプリカ管理
 - 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



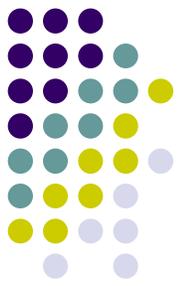
研究目的と成果

- 研究目的

- ユーザ利用負荷を抑え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手法の提供

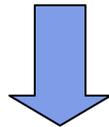
- 研究成果

- 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ジョブ実行手法を提案
- 従来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

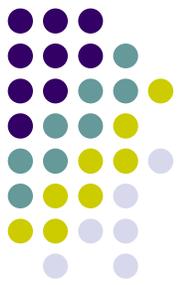


関連研究 - Stork[Kosarら, '04]

- データ転送用スケジューリングシステム
- DAGManを介してCondor[Livnyら, '88]と連動
 - データインテンシブアプリケーション実行環境を提供
 - 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

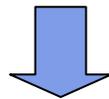


転送元・転送先が静的に決定される
ためアクセス集中の回避は困難



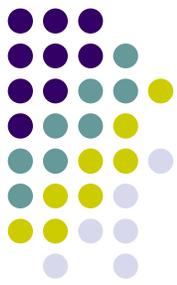
関連研究 - BAD-FS[Bentら, '04]

- 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
 - WAN上のファイル転送の最小化
 - データインテンシブアプリケーションを効率的に実行



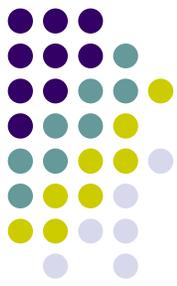
- 利用するデータは静的に決定
- ユーザの負荷が高い
 - タスク間のデータの流れ・サイズなどを記述する必要あり
 - 理想的には利用するデータ名の記述のみ

```
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
```



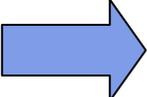
関連研究 – レプリカ管理

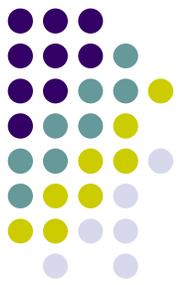
- Globusデータ管理サービス[Allockら, '01]
 - Replica Location Service(RLS)による論理ファイルと物理ファイルのマッピング管理
 - GridFTPやReliable File Transfer(RFT)によるデータ転送サービス
- ↓
- ユーザによるレプリカ管理が必要で利用負荷が高い
 - 1対1転送を想定しているためアクセス集中が不可避
 - ジョブのスケジューリングとは独立な処理



提案手法

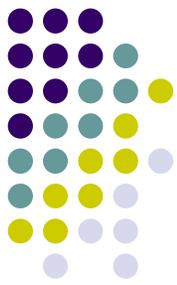
- ジョブスケジューリングとレプリカ管理をタイトに結合
 - データロケーションを意識したジョブスケジューリング
 - データ再利用性の向上
 - データへのアクセス効率の向上
 - 同一データ転送リクエストの集約
 - 複数ノードへのマルチキャスト転送
- 計算資源の遊休時間の有効利用
 - 無視できないアクセスコストの有効利用
 - データ転送と計算の同時実行

 システム全体の資源利用効率が上昇

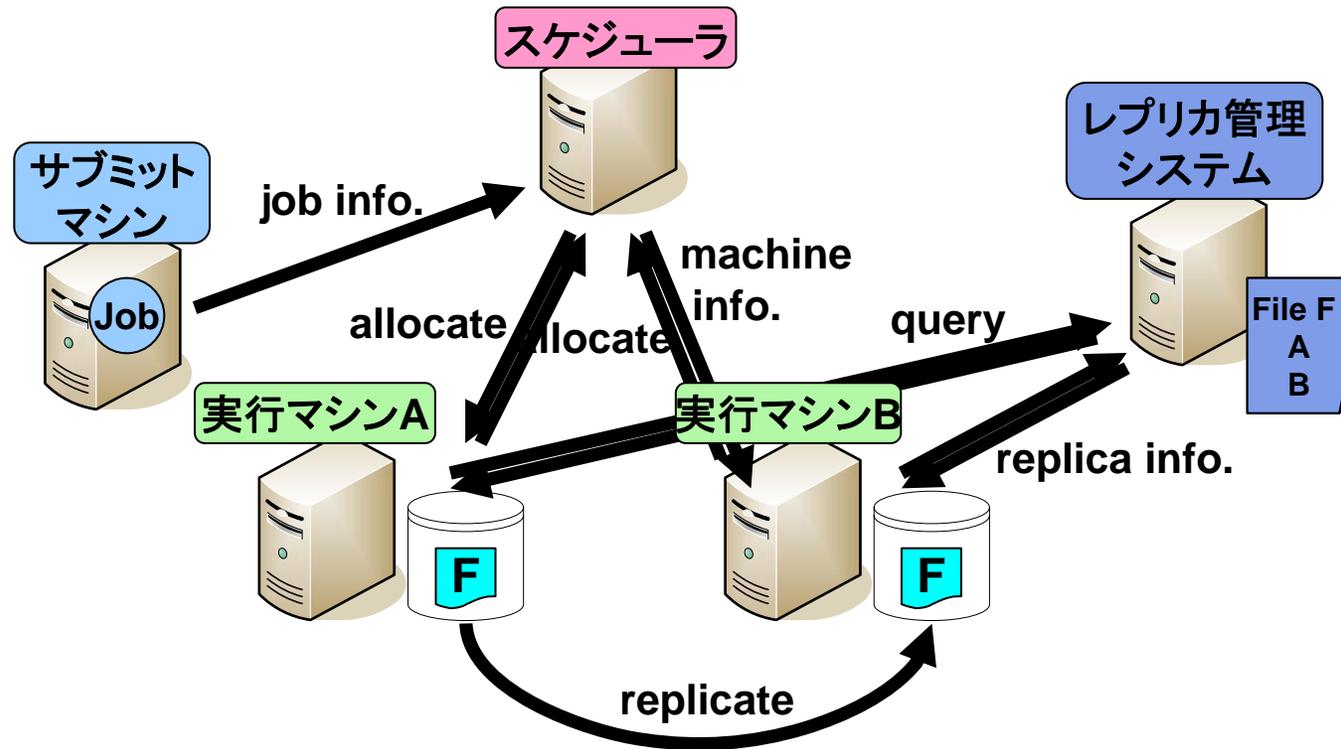


提案システムの設計

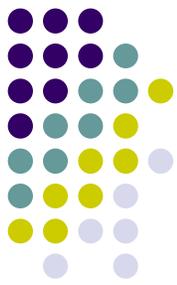
- データの仮想化によるユーザ利便性の向上
 - 仮想的な名前空間と物理ロケーションのマッピング管理
- レプリカ情報を加味したジョブスケジューリング
 - レプリカ保持ノードへ優先的にスケジューリング→再利用性
 - 転送コストの低いノードへスケジューリング→遊休時間縮小
- ローカルディスクへのデータのキャッシング
 - ジョブ実行後にステージングデータを消去せずキャッシング
- 同一データの転送要求の集約
 - WAN上のデータ転送を最小限に抑制
- 計算資源の遊休時間の削減
 - データ転送中に計算ジョブを実行



提案システムの概要

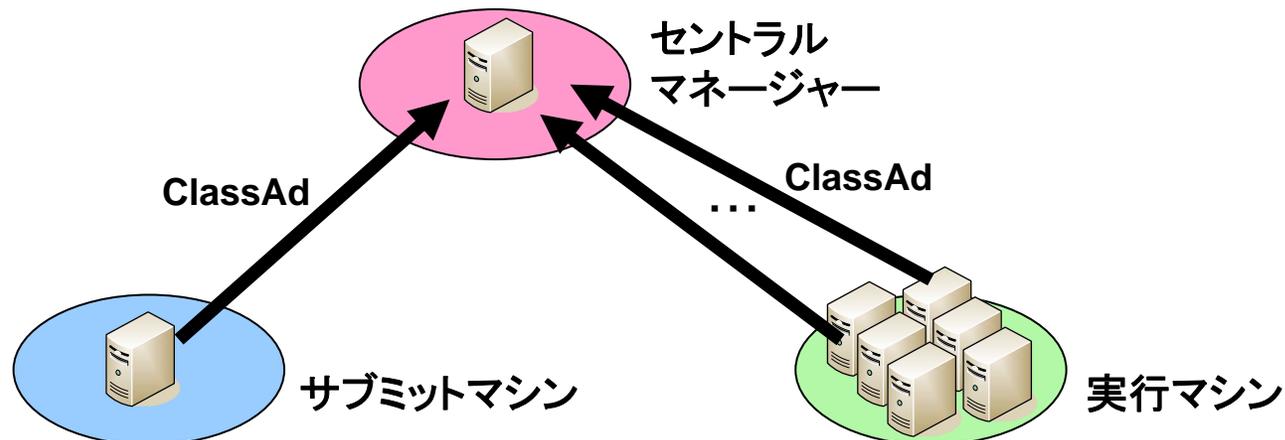


レプリカ管理システムとの連動によりデータの
再利用性の向上・アクセス集中の回避に

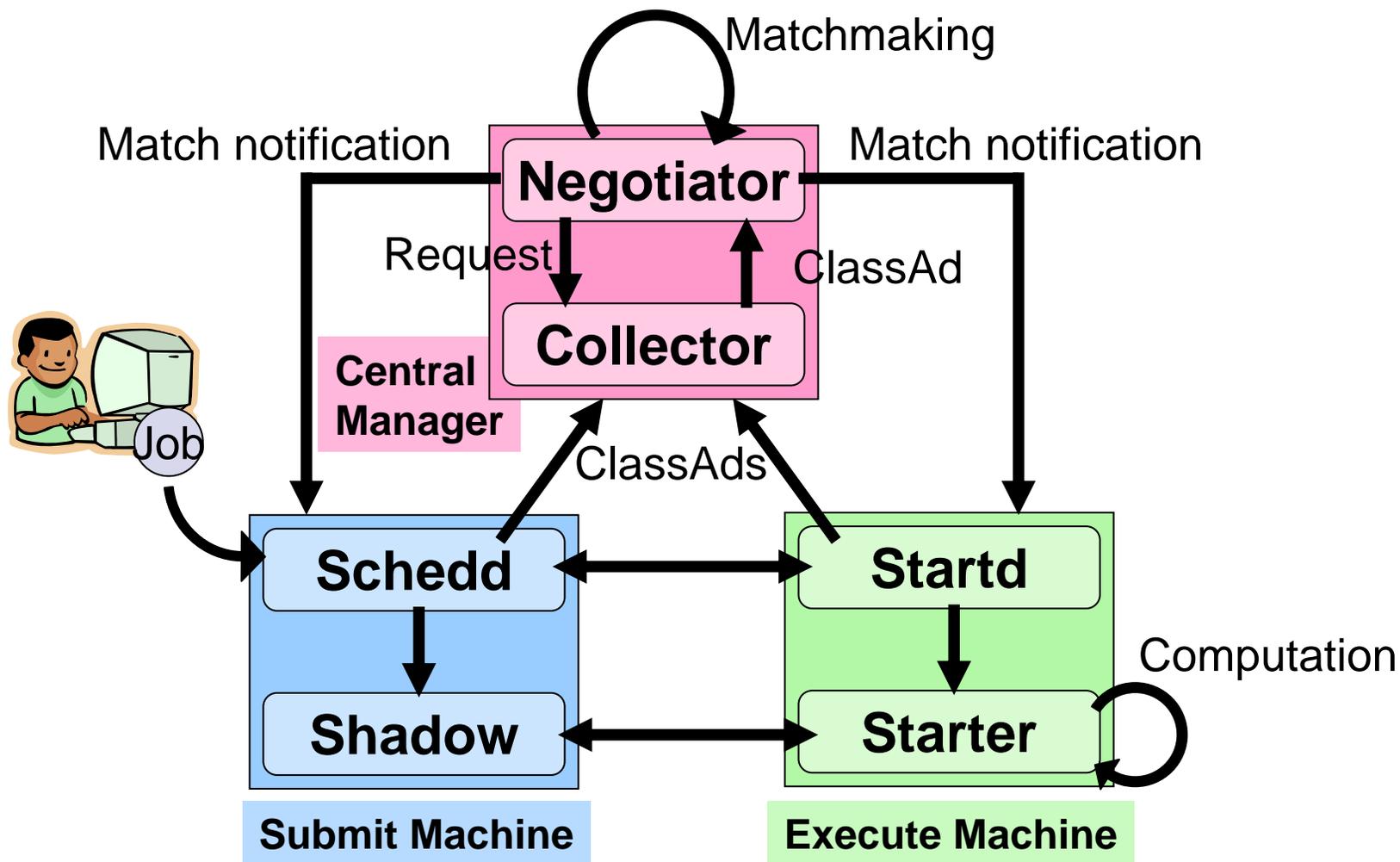


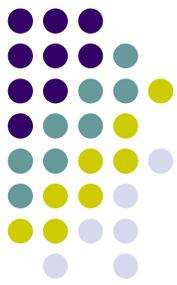
プロトタイプシステムの実装

- 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





スケジューリング機構

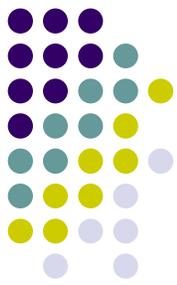
- 受信したマシンとジョブのClassAd[Livnyら, '97]の中からマッチメイキング[Ramanら, '98]により最適なマシンとジョブの組み合わせを決定

マシンのClassAd

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

ジョブのClassAd

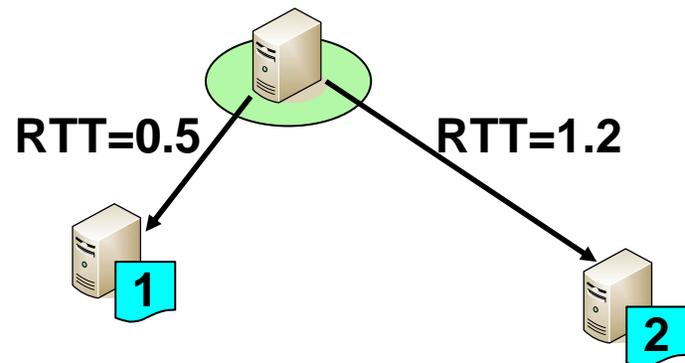
```
MyType = "Job"  
TargetType = "Machine"  
Cmd = "sim"  
Owner = "smith"  
Args = "900"  
Out = "sim.out"  
Rank = Memory  
Requirements =  
    (Arch == "INTEL") &&  
    (OpSys == "LINUX")
```

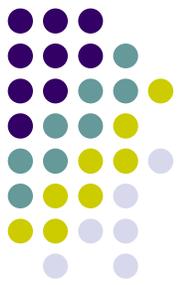


レプリカ管理システムとの連動

- 実行マシンのStartdが定期的に各ファイルのレプリカ作成コストを見積もり
 - セントラルマネージャに送信するマシン情報ClassAdに得られたレプリカコストに応じたレプリカ情報を追加
 - 現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, ..."
```

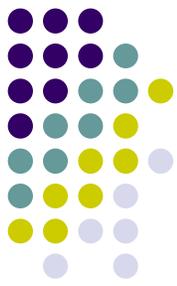




本システムのスケジューリング

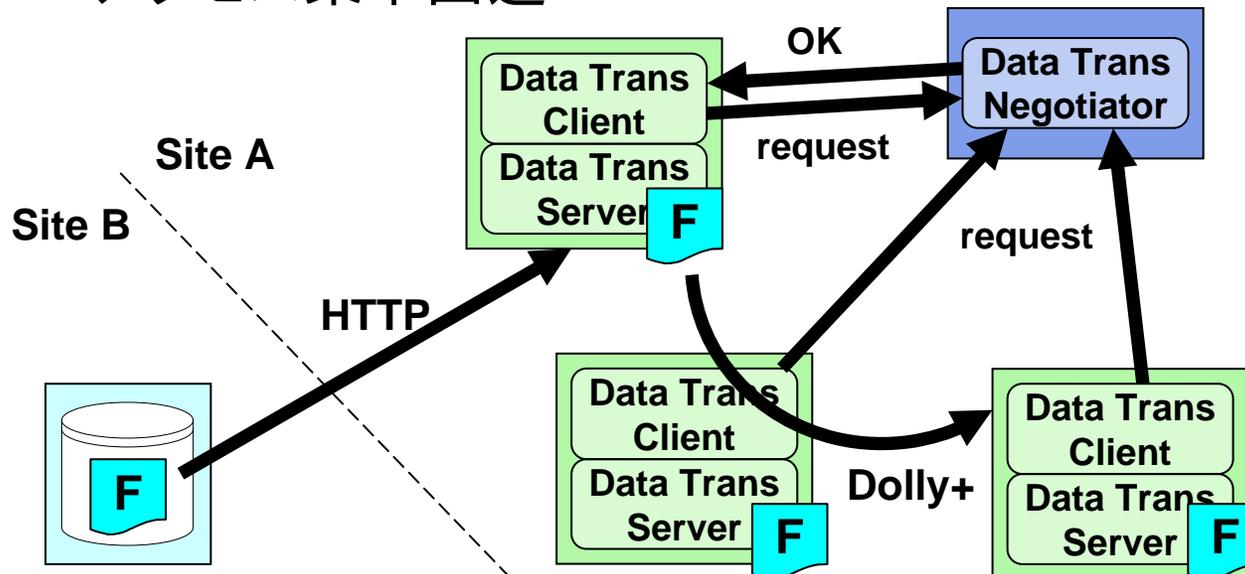
- データロケーションを意識したスケジューリング
 - 実行マシンからセントラルマネージャに送信されたマシン情報に追加されたレプリカ情報を利用
 - マッチメイキング[Ramanら, '98]時にrank値にレプリカのロケーションに応じた値をプラス
- ユーザが記述するサブミットファイル

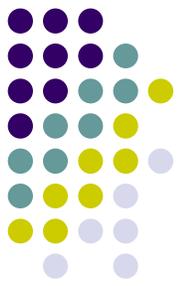
```
executable = application
input      = input.$(Process)
output     = output.$(Process)
error      = error.$(Process)
arguments  = $(Replica_Files)
transfer_replica_files = data1, data2
queue 100
```



レプリカ管理システム

- MultiReplication[Takizawaら, '05]を利用
 - レプリカの位置情報管理するReplica Location Service
 - 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

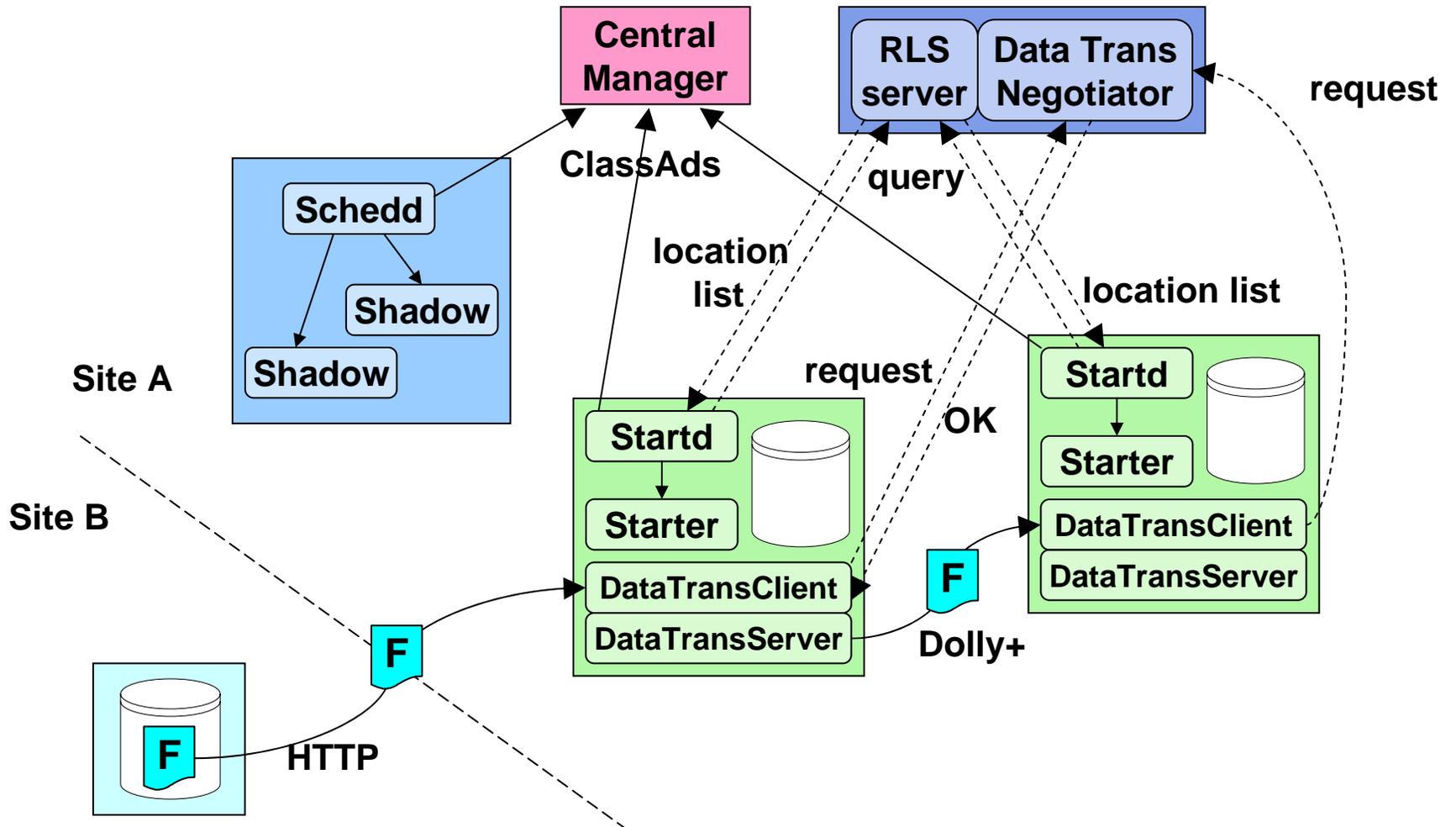


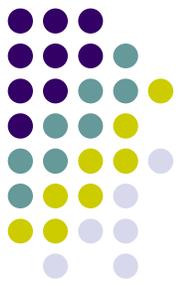


データ転送と計算の同時実行

- データ転送と計算の同時実行
 - ジョブの特性と実行マシンの状態に応じたスケジューリング
 - データ転送中の実行マシンにコンピュータインテンシブジョブ
 - 計算実行中の実行マシンにデータインテンシブジョブのデータ転送
 - ジョブ特性の判定基準
 - ステージングすべきデータサイズにより判定
 - マシンごとに設定された閾値より小さければコンピュータインテンシブジョブと判定

システム全体図





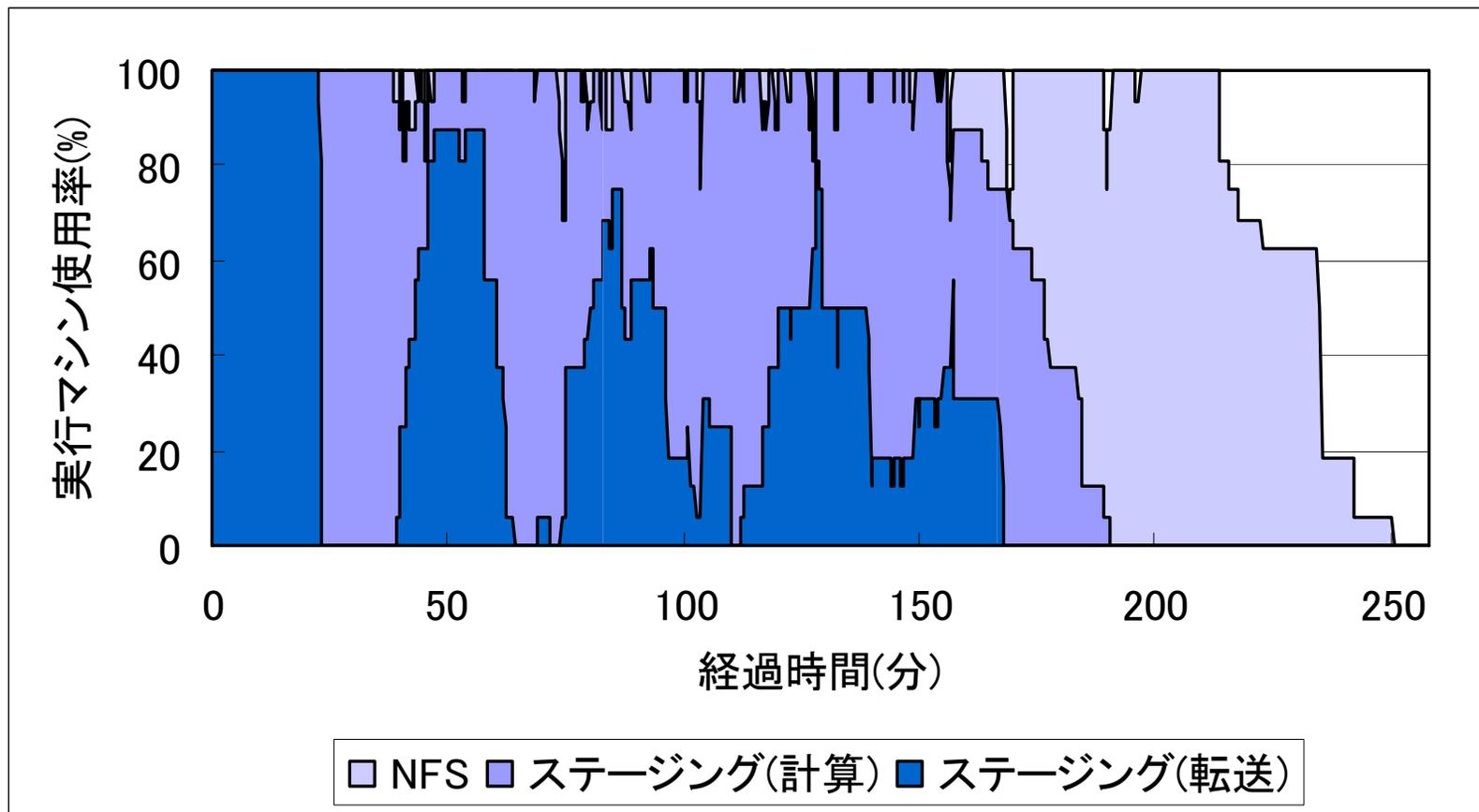
評価実験

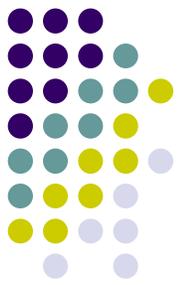
- 計算資源の利用効率を評価
 - 核酸・アミノ酸の相同性検索ツールBLAST[NCBI]
 - クエリに類似した配列をデータベースから検索
 - 5クエリの検索を行うジョブを80個サブミット
 - 約3GBのデータベースを使用
 - 20分弱の計算時間が必要
 - PrestoIIクラスタの16ノードを実行マシンとして使用
 - 1ノードあたり平均5ジョブを実行

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

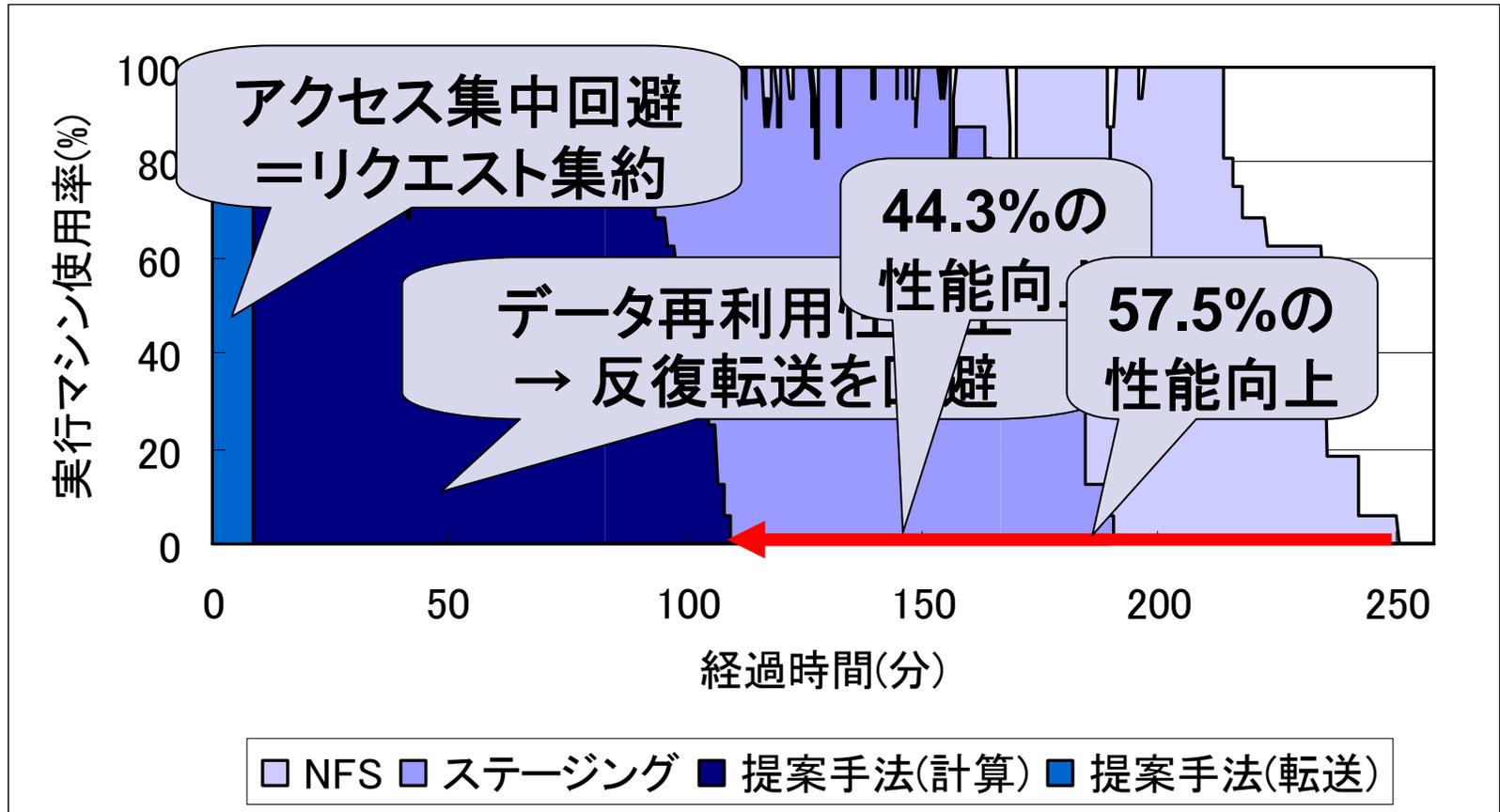


マシン使用率の推移 - 従来手法





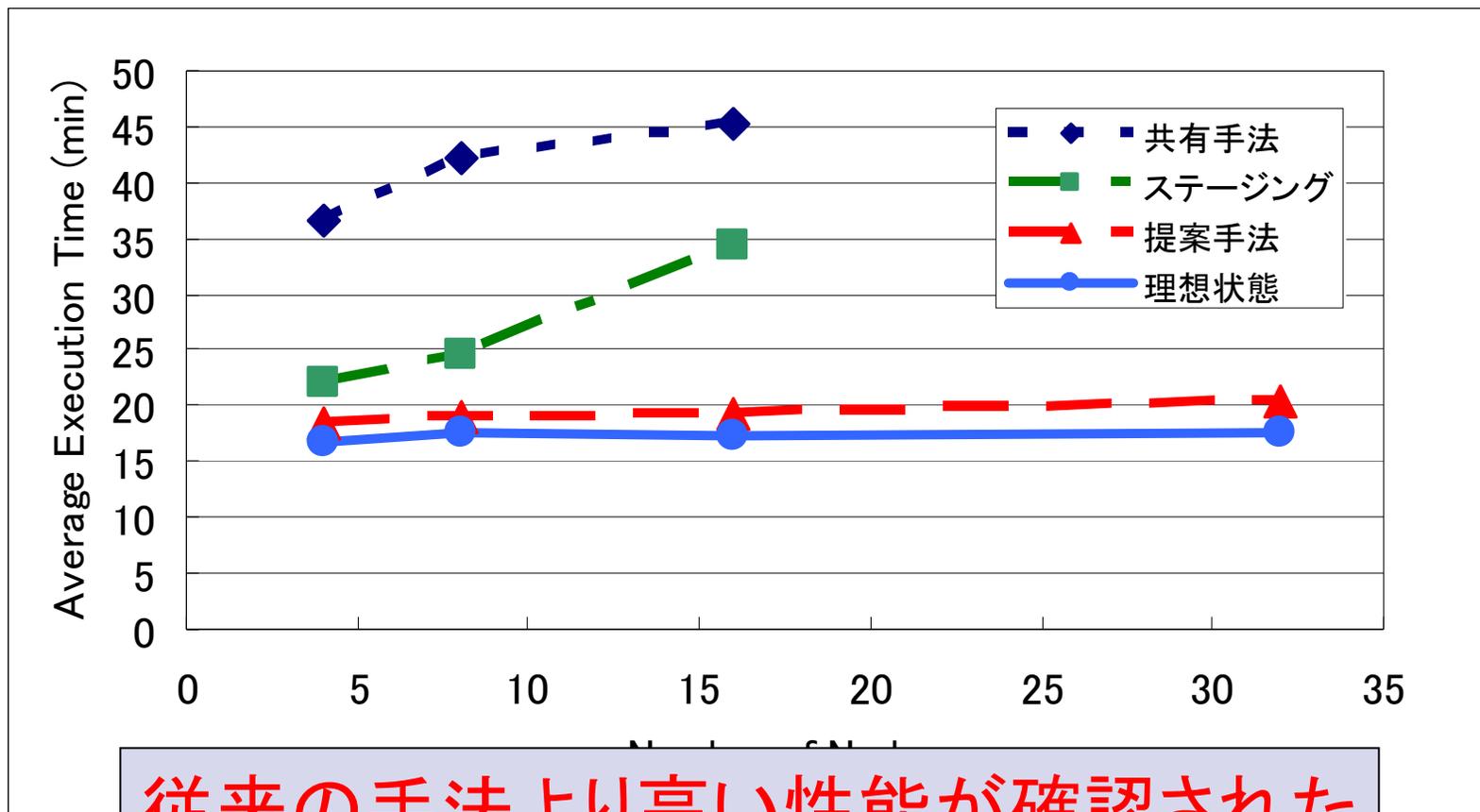
マシン使用率の推移 - 提案手法



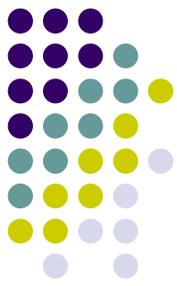
システム全体の利用効率アップ



平均実行時間の比較



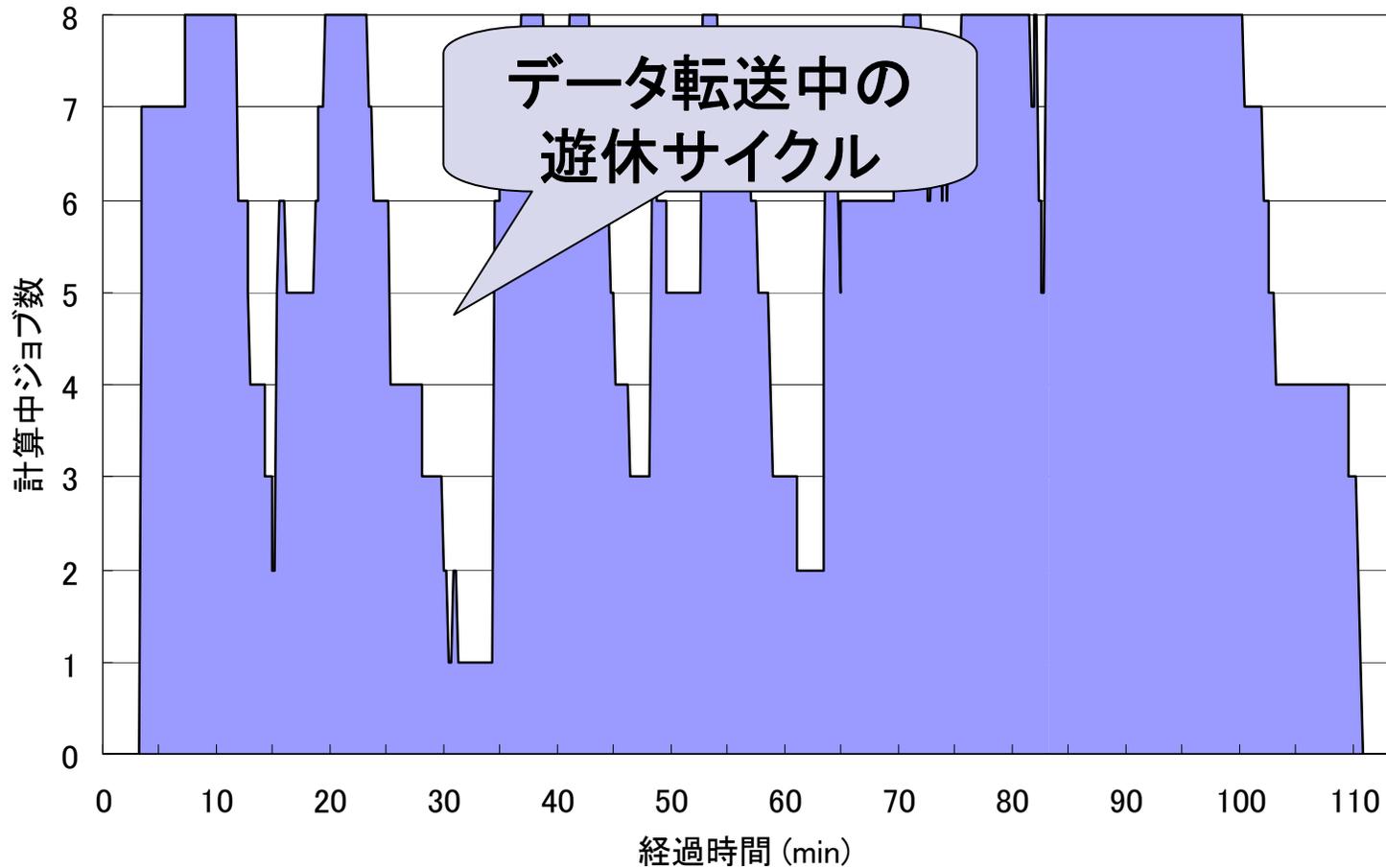
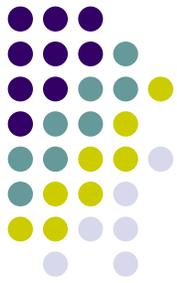
従来の手法より高い性能が確認された



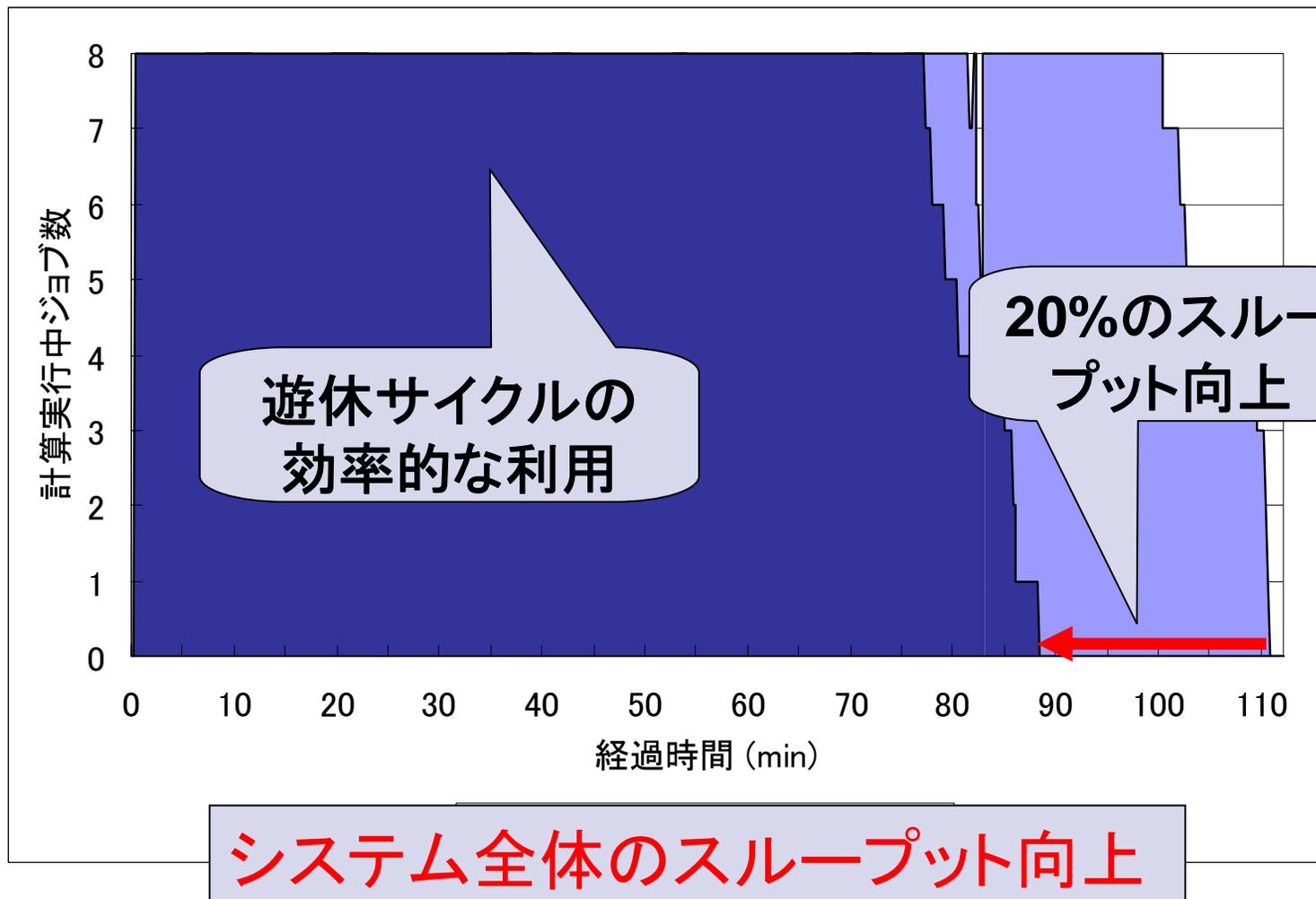
データ転送と計算の同時実行

- データ転送とコンピュータジョブを同時実行
 - ジョブ1
 - BLAST
 - 40ジョブサブミット
 - 約3GBのデータベースファイルを毎回ステー징
 - ジョブ2
 - モンテカルロ法による円周率
 - 8ジョブサブミット
- 評価環境
 - PrestoIIIクラスタ8ノード

計算実行中のジョブの推移 (同時実行なし)



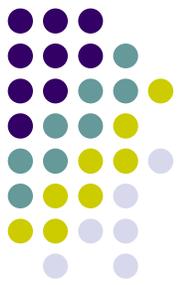
計算実行中のジョブの推移 (同時実行あり)





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
- サンプルアプリケーションの実行により従来手法と比較して効率的なジョブ実行を確認
- 遊休サイクルの効率的な利用によるスループット向上



今後の課題

- より効率的なデータ転送
 - WAN上のデータ転送のさらなる抑制
- スケジューリングアルゴリズムの改良
 - 詳細なジョブの特性、マシン状態の把握
 - スケジューリングコストと最適なマッチングの評価
- 大規模出力データへの対応
 - ワークフロー実行
- チェックポイント機構の導入
 - さらなるスループットの向上
- 複雑なシナリオを用いた大規模環境での評価実験