情報処理学会論文誌　コンピューティングシステム　Vol.5　No.4　61–75 (Aug. 2012)

グリッドコンピューティング環境における ジョブ間データ転送の削減

佐賀 一繁[1]　合田 憲人[2,a)]　三浦 謙一[2]

受付日 2012年1月17日, 採録日 2012年5月3日

概要：現代の科学研究は複雑化，多角化してきており，研究分野や国をまたがる共同研究が多く行われている．このような共同研究の円滑な推進には，参加組織が持つデータ，計算資源，ストレージ資源などを共有できるグリッドコンピューティングが有効である．従来，研究分野や国が異なると使用するグリッドミドルウェアが異なり，資源を連携させることができなかったが，主要なミドルウェアが Open Grid Forum（OGF）が策定したインターオペレーション仕様 HPC Basic Profile（HPCBP）に準拠することにより異種グリッド間の資源連携が可能になってきた．しかし，この異種グリッド間インターオペレーション環境には，各グリッドのアーキテクチャの違いにより，単一グリッド環境とは異なる問題が存在する．その 1 つが，ジョブの実行アカウントの管理方法の違いによるファイル転送方法の違いである．グリッド計算資源のジョブ実行アカウントには，ジョブごとに動的に決定される一時アカウントと一般の計算機システムと同様な静的アカウントの 2 種類があり，どちらを採用するかは各ミドルウェアに依存する．静的アカウント環境におけるジョブ間のファイルの引き渡しは，計算資源間の直接転送が可能であるのに対し，一時アカウント環境では一時ストレージを介した転送が必要であり，同じファイルの転送が 2 回に行われることになる．この違いのため，他グリッドの計算資源へジョブを投入するときには，先方のアカウント管理方法に従ったファイル転送の制御をする必要がある．しかし，依存関係のあるジョブなど先行するジョブの投入時には後のジョブの計算資源が決定されていない場合が多く，あらかじめ相手計算資源に合わせたファイル転送制御を指示することは難しい．また，近年の計算資源の急速な性能向上はデータの巨大化をもたらし，この余分なファイル転送はジョブの長期化やネットワーク負荷の増大などの原因となる．一時ストレージが必要となる理由は，一時アカウントとその作業ディレクトリがジョブ終了とともに無効化されることにある．このため，各ジョブはジョブ内で計算結果を外部資源に転送する必要がある．しかし，このときこの計算結果を必要としている他ジョブの実行状態が不定であるため一時ストレージに転送する制御が必要となる．そこで本論文では，ファイルの引き渡しが必要なジョブのステートを制御して両ジョブの実行状態を同期させファイル転送する方法を提案する．具体的には，ファイル送出元のジョブを終了する前に停止させ，受け取り側のジョブがファイル転送可能状態になるまで待機させる．転送可能な状態になったらファイルを転送し，転送終了した段階で同期を解除する．このような実行同期は，単一グリッドの独自仕様でなら比較的容易に実現できるが，インターオペレーション環境では仕様の共通化が必要となり難しい．そこで，現在 OGF で議論されている次世代インターオペレーション仕様の要件を基に実現する方法を提案する．また，提案手法を実装し，関連するジョブ間で直接ファイル転送できること，ファイル転送が少ない分だけジョブの実行時間が短縮されることが確認できたので報告する．

キーワード：グリッドコンピューティング，インターオペレーション，HPCBP，データステージング，PGI

Reduction of Data Transfer in Grid Computing Environments

Kazushige Saga[1]　Kento Aida[2,a)]　Kenichi Miura[2]

Received: January 17, 2012, Accepted: May 3, 2012

© 2012 Information Processing Society of Japan

Abstract: Because modern scientific researches become complicated and diversified, many joint studies are carried out among different research fields and different countries. The grid computing is effective for smooth promotion of such collaborations because it can share resources such as data, computing resources, and storage resources the participant organizations have. However, because the grid middleware using in different research fields and different countries are different, conventional grid middleware are not able to share resources with other type grids. Therefore many grid projects focus grid interoperation using Open Grid Forum's (OGF) interoperation standard HPC Basic Profile (HPCBP). In the interoperation environment, because the grid middleware have different architectures, there are many issues different from single grid environment. The one is the difference in file transfer method by the difference in management method of the job account. There are two types of job accounts for the computing resources, temporally account, which is assigned for each job dynamically, and static account like conventional computer systems. In the environment which uses static accounts, the file transfer between jobs can be done directly. On the other hand, the transfer through the temporary storage is necessary in the temporary account environment. Therefore in the temporary account environments, the same files are transferred twice. Because of this difference, it is necessary to control the file transfer according to the account management of the other parties. However the computing resources of the later job are not often decided at the time of the submission of the preceding job, and it is difficult to manage file transfer method in conformity to partner resources in advance. In addition, because the rapid performance improvement of recent computing resources brings the make the data gigantic, this extra file transfer causes the prolongation of the job or the increase of the network load. The reason why a temporary storage is necessary for is that a temporary account and the temporary working directory are disabled with the job end. Therefore, it is necessary for each job to transfer the computing results to the outside resources in the job. However, it is necessary to transfer it to a temporary storage because the state of the job needing the results is unsettled. Therefore this paper proposes that a method to synchronize the state of the jobs that the file transfer is needed, and to do file transfer. Specifically, the job which sends the files is stopped before finalizing and is waited until the partner finishes the file transfer. This proposal is based on the requirements of the next generation interoperation specification which is discussing in the OGF. It is confirmed that the prototype could transfer the file between jobs directly and the job run time was shorten.

Keywords: grid computing, interoperation, HPCBP, data staging, PGI

1. はじめに

現代の科学研究は複雑化，多角化してきており，課題を解決するには異なる分野や国をまたがる協力が必要な研究が数多い．たとえば ATLAS [1] や ITER [2] プロジェクトなどでは非常に多くの国が参加している．このような共同研究の円滑な推進には，互いの知識だけでなく，参加機関の実験装置，計算機，ストレージ，データベース，データ，アプリケーションなどの研究資源を共有することが不可欠である．そこで欧米では，このような研究開発環境として過去 10 年以上にわたりグリッドコンピューティング環境（以下，グリッド環境と略）が構築され広く利用されている．その規模は，2 組織間の小規模のものから国や地域を代表する 300 以上のコンピューティングセンタを結び，1 万人以上の利用者が 1 カ月に 1,300 万ジョブを投入する大規模なものまで様々である [3]．このようなグリッド技術による研究資源の共同利用環境はクラウドコンピューティング技術と融合しながら今後も継続される方向にあり，2010 年から 2011 年にかけて更新された欧州の EGI [3]（欧州の大規模グリッド環境 EGEE [4]，DEISA [5]，NorduGrid [6] と各国のナショナルグリッドなどを再編成したもの）や米国の XSEDE [7]（TeraGrid の後継）などの大規模共同利用環境もグリッド技術をベースにしている．

このようにグリッド環境は共同研究環境として有効であるが，これを構成する従来のグリッドミドルウェアは研究コミュニティや国策プロジェクトが独自の仕様で開発したものであり，異なるグリッドミドルウェア間ではその仕様の違いから資源連携することができなかった．そこで Open Grid Forum [8]（OGF）は，異なるグリッド環境間での計算資源連携を実現するジョブインターオペレーション仕様 HPC Basic Profile（HPCBP）仕様 [9] を策定した．グリッド間インターオペレーションの重要性を考える多くのグリッドプロジェクトが HPCBP 準拠のモジュールを開発し，相互ジョブ投入実験などによりそのインターオペレーション動作の確認や問題点の抽出などを行っている．しかし，HPCBP 仕様は相互にジョブを投入するために必要な最小の仕様であり実運用に適用する仕様としては不十分である．そこで，OGF では実運用に耐えうる次世代のインターオペレーション仕様 Production Grid Infrastructure [10]（PGI）の策定を行っている．現在策定作業中であり仕様は未決定であるが，実現すべき機能などの要件は公開されている [11]．なお，EGI や XSEDE では異種グリッド環境との資源連携を重視したアーキテクチャ

[1] 総合研究大学院大学
The Graduate School for Advanced Studies, Miura, Kanagawa 240–0193, Japan
[2] 国立情報学研究所
National Institute of Informatics, Chiyoda, Tokyo 101–8430, Japan
[a] aida@nii.ac.jp

© 2012 Information Processing Society of Japan

を採用しており，実運用や機能要件のために独自の拡張をしながらも HPCBP 仕様のみでも動作するサービスを実装している．

このように世界のグリッド環境は「グリッドのグリッド」に向かっており，「必要なときに必要な計算パワーを使用できる」というグリッドの理想にさらに近づきつつある．しかし，グリッドインターオペレーション環境には単一グリッド環境とは異なる問題が存在している．その 1 つが，計算資源のジョブ実行アカウントの管理方法の違いによるデータステージング方法の違いである．データステージングとは，ジョブの実行に必要なファイルや計算結果などを外部資源との間で転送することを指す．グリッド環境の計算資源のジョブ実行アカウントには，ジョブごとに動的に決定される一時アカウントと，ローカル計算機システムと同様な静的アカウントの 2 種類があり，どちらを採用するかはグリッドミドルウェアやグリッド環境により異なる．一時アカウントとその作業ディレクトリにはライフタイムがあり，計算資源へのジョブ投入時に生成され，ジョブ終了時に無効化される．このため，ジョブは計算実行に必要なファイルをジョブの一部として入手し，また計算結果をジョブ実行の一部として外部に転送する必要がある．この制約により，依存関係のあるジョブなどジョブ間でファイルを引き渡す必要があるワークフロージョブでは，先行するジョブの結果を一時ストレージに格納し，他方のジョブは一時ストレージから結果を取り出してから計算を実行するといった制御がされている．一方，静的アカウント環境では，アカウントはジョブ終了後も有効であり，また結果を計算資源内（またはそのサイト内）に保持し続けることが可能なため，依存関係のあるジョブでもジョブを実行する計算資源間で直接データステージングすることができる．したがって，これらアカウント管理が異なるグリッド環境のジョブ間でデータステージングを行うには，各グリッドミドルウェアをデータステージング動作の違いを考慮した制御方法に変更する必要がある．このときどちらか一方の方法に統一することも考えられるが，双方とも相反する利点があり，また従来環境との互換性の問題あるため難しい．そこで本論文では，ジョブステートを外部から制御可能とし，関連するジョブのステートを同期させることにより，一時アカウントを使用しながらも一時ストレージを不要とする方法を提案する．また，インターオペレーションに参加するすべてのグリッドミドルウェアで提案手法をサポートする必要があるため，現在検討されている PGI の要件や議論に沿った方法で実現する方法を提案する．具体的には，HPCBP のジョブステートモデルにおいて，1 つのステートとしてまとめられていた複数の動作を独立したステートに分離し，また各ステートでジョブを HOLD 可能とする．さらに HOLD 状態への遷移や，HOLD 状態からの復帰を外部から制御し，依存関係のあるジョブでも矛盾なく一時的に同時に存在することを可能にすることにより，直接ファイル転送を可能とする方法を提案する．

2 章ではグリッドインターオペレーションとデータステージングに関する関連研究を紹介する．3 章では問題を解決するためのジョブステートモデルとジョブ制御方法を提案し，PGI 要件を基にした設計詳細を説明する．また提案手法を実装し，関連するジョブ間で直接ファイル転送できること，ファイル転送が少ない分だけジョブの実行時間が短縮されることが確認できたので報告する．なお，提案手法のドラフトは，2011 年 7 月に開催された OGF-33 の BES-WG で，PGI 要件の仕様化議論のたたき台として提出された．

2. 関連研究

2.1 異種グリッド間インターオペレーション

(1) 標準仕様による異種グリッド間インターオペレーション

異種グリッド間インターオペレーション仕様である HPCBP は機能仕様ではなく，異種グリッド間でジョブを投入するために準拠すべき機能仕様とその準拠範囲を規定するプロファイル仕様である．以下に規定している基本的な機能仕様をあげる．これらの仕様の中には，さらに別の個別仕様を前提とするものもある．

- ジョブ表現

OGF/Job Submission Description Language [12]（JSDL）：アプリケーションパスや環境変数などアプリケーション情報，リソース量など資源情報，入出力データなどのデータステージング情報などから構成される xml 表現のジョブ実行定義言語．

OGF/JSDL HPC Profile Application Extension [13]：HPC アプリケーションをインターオペレーション環境で実行するための JSDL の制約と拡張．

- ジョブ実行サービス

OGF/Basic Execution Service [14]（BES）：ジョブ実行サービス仕様．計算資源，スケジューラインタフェースなどに適用するジョブの生成，実行，キャンセルなどの実行管理仕様．

- 認証クレデンシャル

x.509 証明書 [15]：ITU-T の PKI 規格である X.509 の公開鍵証明書．

ユーザ/パスワードトークン [16]：暗号化されたトランスポートチャネルが確立できたときに使用可能なユーザ名，パスワード．

この HPCBP 仕様に準拠した計算資源とクライアントは，多くのグリッドプロジェクトによってプロトタイプが開発され，その中には実運用で使用されているものもある．

- European Middleware Initiative [17]（EMI）

欧州では，従来からの欧州の有力なグリッドインフラ，

© 2012 Information Processing Society of Japan

DEISA，EGEE，NorduGrid などが，HPC 向けの Partnership for Advanced Computing in Europe [18]（PRACE）と，HTC 向けの EGI に再編された．EMI はこれらの環境を構築するグリッドミドルウェアの開発，保守，管理を行う組織であり，そのグリッドミドルウェアは，従来のグリッドインフラで使用されていた，UNICORE [19]，gLite [20]，ARC [21] などを統合したものである．この統合とは複数のミドルウェアを 1 つのミドルウェアにまとめたのではなく，これらミドルウェアにインターオペレーション機能を実装してミドルウェア間を連携可能としたものである．これにより，各グリッド環境における従来からの使用法，ユーザ開発資産を維持しながら，欧州内の異なるグリッド間での資源連携を可能としている．このインターオペレーション仕様は，ヨーロッパの国際仕様である EMI Execution Servic [22]（EMI-ES）であるが，一部ミドルウェアには，HPCBP 準拠のジョブ実行サービスも実装されており，欧州以外のグリッド環境とのインターオペレーションも可能としている．将来的には後述する PGI 仕様に準拠する予定もある．

- Extreme Science and Engineering Discovery Environment（XSEDE）

XSEDE は米国のナショナルグリッドである TeraGrid の後継のグリッドインフラであり，EGI と同様に複数の異なるミドルウェアから構成されている．大きくは標準仕様に準拠したサービスで構成された XSEDE Enterprise Services と，多くの独自仕様のサービスで構成された Community Provided Services の 2 つに分かれている．XSEDE Enterprise Services はジョブ実行インタフェースとして HPCBP にも準拠している．このため，上記 EMI ともインターオペレーションが可能となっている．なお，Community Provided Services には Globus の GRAM5 [23] や AmazonEC2 [24] なども採用されている [25]．

- REsources liNKage for E-scIence [26]（RENKEI）

国立情報学研究所（NII）では，eScience のための計算資源連携の研究プロジェクト「研究コミュニティ形成のための資源連携技術に関する研究 REsources liNKage for E-scIence」を実施した（2008 年 9 月～2012 年 3 月）．RENKEI プロジェクトでは，同研究所が開発したグリッドミドルウェア NAREGI [27] を HPCBP 仕様に準拠させることにより，他のグリッドとのインターオペレーションを実現することを目標の 1 つとしていた．RENKEI プロジェクトでは開発した HPCBP 準拠の計算資源とクライアントで，EMI（UNICORE）や XSEDE など欧米の環境とのインターオペレーション実験に成功している．

（2） PGI 仕様による異種グリッド間インターオペレーション

PGI 仕様は HPCBP 仕様が持つ多くの問題を解決し，実運用可能な異種グリッドインターオペレーションを実現するための仕様である．まだ，策定中であり詳細仕様は決定していない．HPCBP には，グリッド資源情報交換仕様がなく不適切な資源にジョブを投入する，ジョブステートが単純でジョブ実行待ちとデータステージングを同時にできないなど様々な問題を持っている．多くのユーザが安定して有効に使用可能な実運用レベルのインターオペレーション環境を構築・運用するには，さらに多くの機能仕様やその準拠範囲の規定する必要がある．OGF の PGI-WG ではこれを要件としてまとめ，各機能仕様の担当 WG がこの要件を基に詳細な機能仕様を検討している．要件の一部を説明する．

- グリッド間情報交換仕様

HPCBP では，各グリッドが提供する資源の全体情報，アカウント情報などグリッド環境の運用に必要な情報の交換方法，スキーマが規定されていない．資源情報の一部は個々の計算資源から取得可能ではあるが，資源自体の存在を示すまとまった情報がなく広範囲な運用にあたっては実用的ではない．また，オフラインの情報で代替することも可能であるが，ジョブ投入時の運用状況，混雑度合，予算残高など動的な情報が把握できないため，効率的なジョブ投入ができないなど問題が多い．

- ファイル転送プロトコル仕様

HPCBP では，必須のファイル転送プロトコルが規定されていないため，ジョブは投入できても，グリッド間のファイル転送プロトコルの違いからエラーが発生するなどの問題を引き起こす．最低限のセキュアな必須ファイル転送プロトコルの定義とそのサポートプロトコルを公開する仕組みが必要である．

- ユーザ証明書のデレゲーション仕様

HPCBP では，証明書ベースの認証・認可を行う場合，デレゲーション方法が規定されておらず，パスフレーズを安全に転送する方法が問題となる．クライアント群から計算資源に直接ジョブ投入する場合は不要であるが，一般に中間にサービスが入ることが多く，またジョブをマイグレーションする場合も必要となる．たとえ同一サイトのサービス間でのパスフレーズの転送でもセキュリティポリシ違反になる可能性がある．

- 並列ジョブ実行仕様

HPC ジョブとして不可欠な並列ジョブ実行に関する仕様が弱く，資源情報の表現や MPI 対応などが不十分である．

- ジョブステート制御仕様

グリッド環境で効率的なジョブ実行を実現するには，データストレージが遠隔にあることを考慮し，ジョブの実行待ち時間にデータステージングを行う，ジョブエラー発生時にデータステージングなしに再実行を可能にするなど，柔軟なジョブ実行制御が必要である．これを実現するためには，ジョブステートをジョブ実行サービス外からも制御できる必要がある．

© 2012 Information Processing Society of Japan

2.2　データステージング

1章で述べたように，計算資源のユーザアカウントの管理方法の違いよりジョブ間のデータステージング動作に違いがある．一時アカウントと静的アカウントには相反する特徴があり優劣はつけられない．どちらを採用するかは各グリッドミドルウェアの設計思想による．本節では，両管理方法の特徴と，データステージング動作の違いを説明する．

まず，両管理方法の特徴を説明する．グリッドミドルウェアとして広く採用されている gLite などでは一時アカウントを採用している．計算資源にジョブが投入されたとき，計算資源は認証クレデンシャルによってユーザを識別してプールされているアカウントの1つを割り当て，ジョブを実行するための一時的な作業領域を割り当てる．この一時的なアカウントと作業領域は，そのジョブが存在している間のみ有効である（実際には同一ユーザが同一計算資源に複数のジョブを投入した場合には，同一プールアカウントが割り当てられ，1つのジョブが終了しても他にジョブが実行中であればアカウントは残る．説明が複雑になるので，ここではジョブが1つの場合を考える）．プールアカウントは，実際のユーザアカウントとは基本的に無関係で，ジョブ実行時に一時的な関係を持つだけである．このため，ユーザの増減にともなう計算資源のアカウント管理が不要であり，計算資源数が多い場合には管理コストを大幅に低減できるなどの利点がある．しかし，ジョブ実行時の一時的な関係は，割り当てられた作業領域外に不用意にファイルを残すと，後から同プールアカウントを使用する他のユーザに見られるなどのセキュリティ上の短所がある．一方，RENKEI や NAREGI などでは，従来の計算機システムと同様のユーザごとに固定された静的アカウントを採用し home などの固定した作業領域を持つ．これらは一時アカウント環境と異なり，ジョブの実行状態とは無関係に永続的に保持される．また，home に加えジョブごとの作業領域を持つ場合も多く，これにはジョブが終了しても作業領域が残るものと消去されるものがある．静的アカウントには，ローカル計算機システムと同レベルのセキュリティ管理ができ，またジョブの再実行時や同じ計算資源上のジョブ間ではファイル転送が不要などの利点がある．一方，ユーザが増減するごとに，そのユーザにジョブ投入を許すすべての計算資源でアカウントを管理しなくてはならず，計算資源数が多い場合のコストは大きなものとなる．また，設定ミスのため複数の資源間で動作の違いやセキュリティ上の問題を引き起こす可能性があるなどの短所がある．

これらアカウントの管理方法の違いによる，ジョブ間のデータステージング動作の違いを，依存関係のある2つジョブ（先行：ジョブ1，後行：ジョブ2）を例に説明する．後行のジョブ2の実行には，先行のジョブ1の結果ファイルが必要であり，ジョブのデータステージングにより引き渡されるものとする．**図 1** に静的アカウントの例を示す．依存関係のあるジョブのため，ジョブ2はジョブ1が終了してから起動されるが，Site-A の計算資源にはジョブ1の計算結果が残っているので，ジョブ2は Site-A の計算資源からジョブ1の結果ファイルを直接転送（データステージイン）することができる．一時アカウントの場合の例を**図 2** に示す．一時アカウントと一時作業領域はジョブ終了時に消去されるので，ジョブ1はその実行中に結果ファイルを一時作業領域外に転送（データステージアウト）する必要がある．このとき，ジョブ2の計算資源に転送できればよいが，一般にジョブ1へのジョブ投入時にはジョブ2の実行計算資源が決定されていないため（ブローカリングされていない），ジョブ2の計算資源へは直接転送できない．また，逆にジョブ2からジョブ1の計算資源から実行結果を転送（データステージイン）しようにも，ジョブ2の実行時にはジョブ1は終了しているのでジョブ1の結果は残っていない．このため，これら2つのジョブの間での中間結果の引き渡しには中間に一時ストレージが必要となり，静的アカウントの場合と比べファイル転送が1回多く必要となっている．

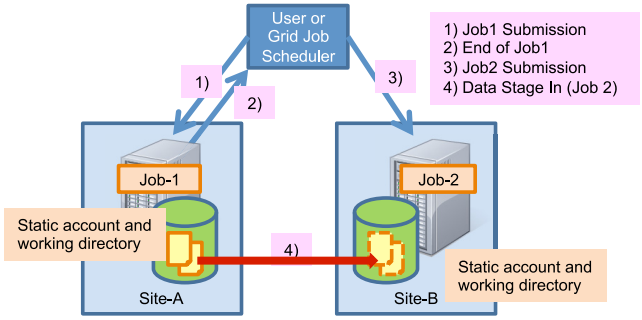

図 1　静的アカウント環境における依存関係のあるジョブのデータステージング

Fig. 1　Data staging of jobs with dependence in static accounts environments.

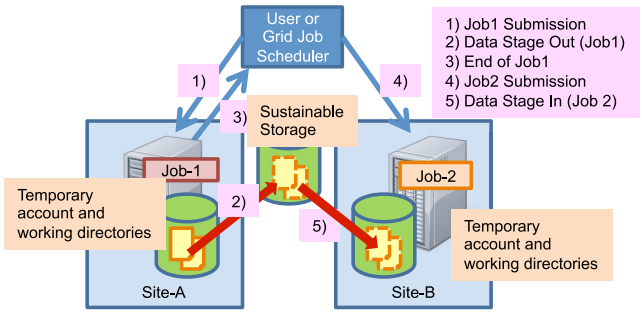

図 2　一時アカウント環境における依存関係のあるジョブのデータステージング

Fig. 2　Data staging of jobs with dependence in temporary accounts environments.

© 2012 Information Processing Society of Japan

3. データステージング問題

2 章で述べたように，一時アカウント方式は計算資源の管理コストの大幅な低減に有効であるため，異種グリッド間インターオペレーションでも必須の機能であると考えられる．近年の計算資源の高性能化による計算量の飛躍的増大は，計算結果として生成されるデータの巨大化をもたらすため，この 1 回多いデータステージング回数を低減させることは，ジョブ実行時間の短縮や資源利用効率の向上などの観点から重要な課題となっている．しかし HPCBP 仕様は機能が低く，この問題を解決することはできない．本章では HPCBP 仕様を拡張してこのデータステージング問題を解決する方法を提案し，PGI 要件を基にした実装設計の詳細を説明する．また，実装したモジュールによる実験で，提案どおり関連するジョブ間で直接ファイル転送できること，ファイル転送が少ない分だけジョブの実行時間が短縮されることを確認したので報告する．

3.1 提案手法

(1) 基本動作と実現方法

2 章で述べたように，一時アカウント環境のジョブ間のデータステージングで一時ストレージが必要な理由は，ファイル送出側のジョブの実行中に転送先のジョブの実行状態や計算資源が不定であることによる．この問題を解決するには，次の 3 つの条件を満たす必要がある．1) たとえ依存関係のあるジョブでも一時的には同時に存在すること（最低条件），2) 従来手法が検出するエラーや警告がすべて検出できること，3) 両ジョブの実行状態の同期制御が可能なこと．HPCBP 仕様では，これらの条件を満たすことができない．そこで，本章では HPCBP のジョブ実行サービス仕様である BES のジョブステートモデルを拡張し，さらにそのジョブステートを外部から管理可能とすることによりジョブ間の同期を実現し，問題を解決する方法を提案する．

単一のグリッドジョブは，計算に必要なデータの転送（データステージイン），計算，計算結果の計算資源外への転送（データステージアウト）の 3 ステップで 1 つのジョブが構成されるワークフロージョブであるといえる．ジョブによってはデータステージインがない場合，静的ユーザアカウント環境では計算ステップしかない場合もあるが，一時アカウント環境では少なくとも計算ステップとデータステージアウトステップは存在する．

一時アカウント環境で，問題となっている依存関係のあるジョブの計算資源間でファイルを直接転送するのに必要な最低条件を考える．一般に依存関係のあるジョブの実行制御は，ジョブ 1 の処理が終了してからジョブ 2 を投入する．一時アカウント環境では，ジョブが終了すると計算資源上のアカウントと作業領域は消去される．このため，直接ファイル転送するには，少なくともジョブ 1 が終了する前にジョブ 2 を起動し，両ジョブの一時アカウントと作業領域を同時に存在させる必要がある．同時に存在していれば，ジョブ 1 の計算ステップが終了した時点で，ジョブ 1 がジョブ 2 の計算資源へ計算結果をステージアウトするか，ジョブ 2 がジョブ 1 の計算資源から計算結果をステージインすることが可能となる．このように，ジョブとしては実行中であるにもかかわらず，途中で計算結果を転送する場合，従来手法と同等のエラーが検出でき，それがワークフロージョブの流れに反映できることが必要である．上記のようにジョブ 1 の計算ステップが終了した時点でデータステージングを行うのであれば，ジョブ 1 の計算結果の正当性は担保されている．このため，計算ステップでエラーが発生した場合は停止せずにジョブの最後まで実行して終了報告としてエラー報告してワークフロージョブの流れに反映することが可能である．またこのとき，ジョブ間の直接データステージングはいずれかのジョブの一部として実行されるので，データステージングで発生するエラーも報告することが可能であり，これもまたワークフローに反映することができる．このような制御を行うにはジョブ間の同期機構が必要である．たとえば，後行のジョブ 2 がジョブ 1 の計算結果をステージインする場合，ジョブ 1 の計算終了とジョブ 2 のステージイン開始，ジョブ 2 のステージイン終了とジョブ 1 のジョブ再開は同期させる必要がある．また，ジョブ 2 の計算ステップの実行は先行のジョブ 1 の計算ステップ終了状態によって実行するか否かが決定されるため，ジョブ 1 の計算ステップの終了状態はジョブ 2 のステージイン時には報告されている必要がある．以上より，ファイルの直接転送の実現には，両ジョブの同時実行，ジョブ間の同期，終了時報告の早期化を実現する機能が最低限必要である．

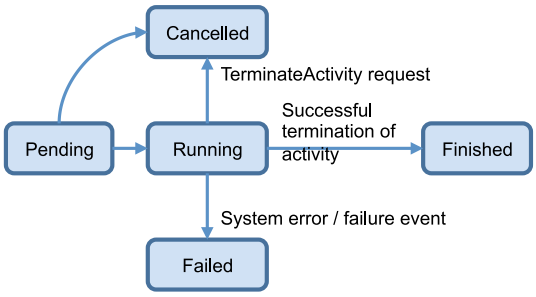

図 3　BES のジョブステートモデル
Fig. 3　Job state model of BES.

これら機能要件の実現方法を検討する．**図 3** に HPCBP のジョブ実行サービスインタフェースである BES 仕様で規定されているジョブステートモデルを示す．このジョブステートモデルでは，データステージングは Running ステートに含まれおり，外部からはデータステージングを実行しているのか，計算を実行しているのか区別できない．BES

© 2012 Information Processing Society of Japan

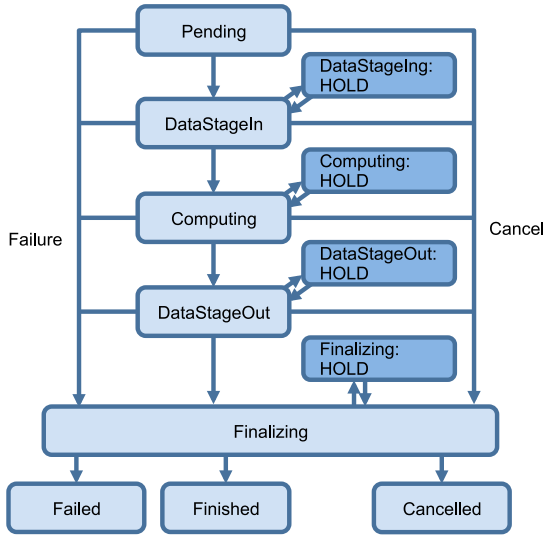

図 4　提案のジョブステートモデル
Fig. 4　Proposed job state model.

の拡張機能を利用して，Running ステートにデータステージイン，実行，データステージアウトのサブステートを設ければ，外部から状態を区別することはできるが，流れは制御できないので，上記のような同期機構を実現することはできない．アプリケーションに同期機構を埋め込む方法も考えられるが，アプリケーションを実行する計算ノードは組織のファイヤウォールの内側にあり外部アクセスが許されない場合が多いので，この方法は使用できない．また，外部にアクセスが許されている計算資源のジョブ実行サービス間で同期機構を持つことも考えられるが，計算資源どうしの制御という従来にはない標準仕様を新たに策定しなくてはならず，また複雑な依存関係のジョブの場合，各計算資源での制御が難しくなることなどの理由から得策ではない．そこで，従来のジョブステートモデルを細かく分割し，分割したステートを外部から制御可能とすることにより，上位のグリッドジョブスケジューラが依存関係に従った同期制御することを提案する．図 4 に提案のジョブステートモデルを示す．具体的には BES の Running ステートを DataStageIn，Computing，DataStageOut の 3 つに分割する．さらに従来 Finished，Cancelled，Failed ステートの動作に含まれていた，ジョブの一時作業領域を消去する動作と各ステートの情報を保持する動作を分離し，一時作業領域を消去するジョブステートを Finalizing，各終了状態を保持するジョブステートを Finished，Cancelled，Failed と再定義したジョブステートモデルとする．DataStageIn，Computing，DataStageOut，Finalizing では，ジョブ投入時の指示によりこれらステートへ遷移したときにその実行前に HOLD させることができるものとする．また，この HOLD 状態は外部からのオペレーションで解除可能とする．さらに，状態遷移したときに上位のスケジューラなどに通知する機能を持つものとする．

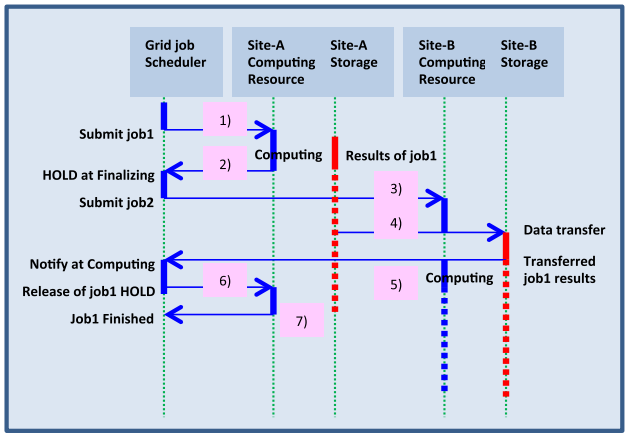

図 5　提案の依存関係のあるジョブのデータステージングの制御
Fig. 5　Management of data staging of jobs with dependence.

なお，本論文の課題とは直接関係ないが，Finished，Cancelled，Failed ステートを一時作業領域消去動作と終了状態保持とに分離することにより，従来取得できなかった Failure や Cancel 発生時までの途中結果も取り出すことができるようになり，デバッグや障害調査にも有効である．

図 5 に依存関係のあるジョブにおける提案方法の動作を示す．ジョブの流れは上位のグリッドスケジューラが制御するものとする．1) スケジューラはジョブ 1 をブローカリングし投入する．このとき，Finalizing で HOLD する指定を行う．なお，計算結果の DataStageOut は指定しない．2) ジョブ 1 は Finalizing に遷移したときにジョブを HOLD し上位グリッドスケジューラに通知する．Finalizing に遷移すれば，Computing のエラーや警告はすべて出つくしているので，終了状態も含めた通知を行う．ジョブ 1 の Finalizing はまだ実行されていないため計算資源上の一時アカウントも作業領域にある計算結果も残っている．3) スケジューラはジョブ 1 の HOLD 通知を受け，その終了状態が正常であれば，ジョブ 2 をブローカリングし投入する．このとき，Computing に遷移したら通知するよう指定する (HOLD は指定しない)．4) ジョブ 2 は，ジョブ 1 の計算資源から計算結果を DataStageIn する．5) ジョブ 2 はステートが Computing に遷移したら，これをスケジューラに通知する．この時点で DataStageIn のエラーや警告はすべて出つくしているので，ジョブ 1 の計算結果はジョブ 2 の計算資源に正常に転送されたことは保障されている．ジョブ 2 はそのまま実行を継続する．6) スケジューラはジョブ 2 からの Computing への遷移通知を受け，DataStageIn が正常に終了していればジョブ 1 の HOLD を解除する．7) ジョブ 1 の計算資源は Finalizing を実行して作業領域などを消去しジョブを終了させる．2) で Computing が正常終了しなかった通知を受けた場合はジョブを継続しないことも可能であるし，6) で DataStageIn が正常終了しなかった場合は，他の計算資源にジョブ 2 を投入するなども可能である．

© 2012 Information Processing Society of Japan

このように，提案手法によれば一時アカウント管理を使用する場合でも，ファイルの引き渡しを必要とするジョブの計算資源間での直接転送が可能となる．また，途中でエラーが発生したときの，ジョブの流れ制御やファイルの保障も可能となっている．

(2)　静的アカウント資源と一時アカウント資源が混在するときの動作

提案手法と従来手法で，依存関係のあるジョブを一時アカウントと静的アカウントの計算資源に投入する場合を比較する．まず，従来手法を使用して，一時アカウント資源で先行ジョブを，静的アカウント資源で後行ジョブを実行する場合を考える．一時アカウント資源の計算結果はジョブ実行中でしか有効でないため，ジョブ実行の一部として計算ステップ終了後に結果を転送するしかない．この転送先はグリッドスケジューラが先行ジョブ投入時に JSDL に書き込む．したがって，ジョブ資源間で直接データステージングを行うには，先行ジョブの投入時に後行ジョブの資源と転送すべき作業領域が決定されている必要がある．しかし，一般に HPC ジョブの実行時間は数日から数カ月と長くなることがあるので，先行ジョブ投入時に後行ジョブの資源を決定すると，後行ジョブ実行開始時には資源が停止している場合や非常に混雑している場合などがあるため得策ではない．このため，図 2 に示す一時アカウント資源間でのデータステージングのように，一時ストレージを使用せざるをえない．逆に，静的アカウント資源で先行ジョブを一時アカウント資源で後行ジョブを実行する場合，先行ジョブの計算結果はジョブ終了後も残っているので，先行ジョブ投入時に後行ジョブ資源を決定する必要はない．後行ジョブはその計算資源のアカウントタイプとは無関係に先行ジョブの計算資源から直接データステージング可能である．この動作は図 1 の静的アカウント資源間のデータ転送と同様にデータ転送回数を低減させることができる．このため，依存関係のあるジョブでは，つねに先行ジョブは静的アカウント資源で実行するようにブローカリング機能を限定することもできるが，資源利用の自由度が低下し利用効率の低下などをもたらすため現実的ではない．したがって，従来手法で依存関係のあるジョブを実行するには，アカウントタイプが異なる資源がブローカリングされても，つねに図 2 のような一時ストレージを使用したデータステージングを行う．

一方，提案手法では資源のアカウントタイプによらず，図 5 に示すシーケンスで，**図 6** に示す動作を行う．基本動作で説明したように，たとえ先行ジョブが一時アカウント資源であってもジョブを HOLD することによって，計算結果を保持し続けることができる．また，静的アカウント資源の場合 HOLD 状態の有無によらず計算結果は保持されている．したがって，資源のアカウントタイプに依存せずに図 6 の動作を実現でき，データステージング回数を低減させることができる．

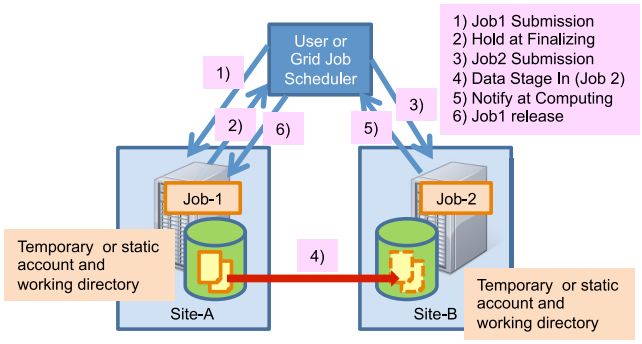

図 6　提案手法による資源アカウントタイプが混在した場合のデータステージング

Fig. 6　Data staging between different account type resources in proposed method.

(3)　HOLD による資源利用効率への影響

ジョブの HOLD 機能をサポートすることにより，資源の利用効率が低下することがある．これを防ぐためには，1) 課題ジョブのデータ転送と第三者ジョブの実行が同時に可能な独立した実行，管理機能，2) データ転送機能においてソースとディスティネーションを起動した資源以外を指定できる第三者転送機能，3) グリッド環境の構成要件として従来手法と同様に一時ストレージとして使用できるグリッドワイドなストレージを用意することが必要である．1) は通常のグリッドミドルウェアで多用される普通の方式であり，2) は GridFTP で実装されている機能であり，3) はグリッド環境の構築ではグリッドワイドな共有ストレージを持つことは通常であり，無理のない提案であると考える．

これら提案手法に従った，先行ジョブ，後行ジョブの計算資源の計算機，ストレージの占有量，利用効率についていて議論する．これらのほかに，各ジョブの管理情報を保持するデータベース資源も HOLD の影響を受けるが，現代の計算機システムが持つ大規模ストレージではジョブの管理情報は無視できる容量のため検討外とする．本提案手法の HOLD は，ローカルスケジューラのジョブホールドと異なり，計算実行途中ではなく計算実行後の HOLD である．このため，上記提案要件 1) により計算実行とデータ転送を独立して管理，実行すれば，課題ジョブの Finalizing の HOLD でデータステージングの終了を待つ状態でも，課題ジョブと関係ない第 3 のジョブが実行可能である．したがって，本提案の HOLD 状態での計算機資源の占有はなく利用効率の低下はない．ただしこれは，ストレージに第 3 のジョブが必要とする空きがある場合である．このため，ストレージの空き状態により区分して議論する．ただし，いずれの場合でも本手法の HOLD 状態は計算ステップ実行終了後であるため計算機の占有はない．ここで，ストレージ資源の空きとは，下位ローカルスケジューラのキュープロパティとしてジョブに許される最大限のスト

© 2012 Information Processing Society of Japan

レージ容量，または第三者ジョブの実行ドキュメントに記述されたストレージ容量要件のうち，少ない方の容量の空きがあることを意味する．また，従来手法との比較議論を容易にするために，従来手法における一時ストレージへのデータ転送速度と提案手法における後行ジョブ資源への直接データ転送速度が同じであると仮定する．

- 先行ジョブ，後行ジョブの計算資源ともストレージに空きがある場合：

上記のように，先行ジョブの計算資源では，計算が終了しだい計算機は解放されストレージにも空きがあるので，ただちに下位ローカルスケジューラ上のキューにある第三者ジョブの実行が開始される．このため，HOLD 状態での先行ジョブの資源の計算機の占有はなく利用効率の低下はない．また，後行ジョブの計算資源でも，後行ジョブの実行に必要なストレージの空きがあるため，たとえ後行ジョブの計算資源で他のジョブが実行中でも，データステージングは先行して実行可能である．このため，他のジョブが終了しだい，計算実行を開始でき計算機の利用効率の低下はない．むしろ，後行ジョブの投入は従来方法より早いため，その分データステージイン，計算を早く開始できるため利用効率が向上する．

- 先行ジョブ計算資源のストレージには余裕がないが，後行ジョブ計算資源のストレージに余裕がある場合：

先行ジョブの計算が終了しても，ストレージに空きがなければ第三者ジョブの実行は開始できない．先行ジョブの計算結果が転送されればストレージは解放され第三者ジョブの実行は開始される．したがって，従来方法と提案方法の利用効率の比較はデータ転送開始までの時間差で議論する．従来方法では計算終了後ただちにデータを一時ストレージにデータステージアウトできるが，提案方法では，Finalizing の HOLD の上位スケジューラへの報告，ジョブ 2 の計算資源のブローカリング，ジョブ 2 のジョブ投入，ジョブ 2 からのデータステージイン開始までの時間が必要となる．この時間は上位グリッドスケジューラの実装に依存するので一概には議論できないが，参考までに著者らのプロトタイプ実装では約 40 秒であった．これは，チューニングにより低減が可能であり，またストレージ資源の空き状態が問題となるような HPC ジョブが生成する大規模データは，数 GB から数 TB となるため，この遅延はデータ転送時間に比べ小さな値と考えられ，利用効率の低下は少ないと考えられる．また，計算結果サイズが小さい場合は，ユーザがあらかじめ認識できるはずであり，JSDL の HOLD エレメント指定を外し従来手法の動作とすることも可能である．

- 先行ジョブ計算資源のストレージには余裕はあるが，後行ジョブの計算資源のストレージに余裕がない場合：

上記のように，提案手法の資源利用効率は先行ジョブのストレージのデータをいかに早く転送するかに依存する．このため，後行ジョブのストレージに空きがない場合，データステージングが開始できず利用効率が大幅に低下することが考えられる．そこで，このような場合，従来手法と同様に先行ジョブのデータを一時ストレージに転送する制御を行う．ただし，従来手法と異なり，先行ジョブがステージアウトするのではなく，後行ジョブが GridFTP などの第三者転送機能を用いて実現する．後行ジョブが自身の資源のストレージの空き状態に従い，直接自身のストレージに転送するか，一時ストレージにいったん転送するかを決定する．一時ストレージに転送した場合，自身のストレージに空きができしだい一時ストレージから転送して計算を実行する．この動作は後行ジョブの計算資源側で判断，実行することが可能なため，ジョブ記述，ジョブ投入インタフェース，上位スケジューラの動作には影響しない．従来手法でも後行ジョブの計算資源に空きがない場合，一時ストレージからのデータステージングは開始できないため，提案手法の資源の利用効率は従来方法と同等と考えられる．

- 両方の計算資源のストレージに余裕がない場合：

上記のように，資源の利用効率は後行ジョブの資源の空き状態に大きく依存するので，上記「先行ジョブストレージ資源に余裕はあるが，後行ジョブストレージ資源に余裕がない場合」と同じである．

以上より，結論として提案方法の資源の利用効率は従来方法より高いか，若干低い程度であり，大きく低下する場合は手動もしくは自動で従来手法と同等動作に切り替える回避できる．一般に現在の計算資源は複数の大規模ジョブを実行するのに十分なストレージを持っており，提案手法を使用すれば利用効率が向上する可能性が高い．このため，提案手法は有効であると考える．

3.2　設計と実装

(1)　基本設計

1 つのグリッド環境でのみ提案手法サポートしても，インターオペレーション環境におけるアカウント管理の違いによるデータステージング制御問題の解決にはならない．インターオペレーション環境のすべてのグリッドミドルウェアがサポートする必要がある．そこで，本提案の要件と PGI 要件を比較した．PGI の機能要件で提案手法が実現できれば，PGI 準拠のミドルウェアでは本提案が動作する可能性があるためである．比較結果として，PGI 要件は提案手法を実現するための要件をおおよそ満たしていることが判明した．不足している要件は PGI-WG に提案していく．本節では，この比較と PGI 要件を基にした設計仕様について説明する．

まず，提案手法を実現するために必要な機能要件を以下にまとめる．

(a) BES の Running ステートは DataStageIn，Computing，DataStageOut の 3 つに分割し，Finished ステー

© 2012 Information Processing Society of Japan

提案	Requirement	Req.ID
(a),(b)	Following Activity states MUST permit 'Hold' : 'Pre-processing', other active states, 'Post-processing'	129
(b)	The service MAY offer the hold and corresponding resume functionality of Activity processing. Change state operation to resume a hold execution and to put the Activity into 'Suspend' (maybe similar to hold).	80
(c)	When the client wants to perform client initiated data-staging-in, the client MUST also specify in the Job Description via 'Hold' points that the Activity holds.	79
(c)	The Job Description document specification MUST permit the Client to request that at specified Activity states, the Execution service sets the Activity on 'Hold'	103
(d)	In order to permit the Client to perform client-directed processing of submitted Activities (for example client-directed data staging), the Execution Service SHOULD manage 'Hold' points for Activities. Fault if not supported.	76
(e)	In order to minimize polling of the Activity Status, the Execution Service MAY implement a mechanism for notification. Asynchronous notification of Activity status (for example: e-mail, SMS, etc.) - human level notification	65
(f)	Data Staging MUST support an agreed set of protocols in PGI, e.g. HTTP(S), scp, gridftp, mailto, RNS/ByteIO	143

図 7　提案の要件と PGI 要件の対応
Fig. 7　Mapping between requirements of proposal and PGI requirements.

トはクリーンナップ処理の Finalizing ステート終了状態を保持する Finished ステートに分割する
(b) Finished を除くこれらステートで，ステート動作実行前に HOLD 可能であること
(c) ジョブ投入時に HOLD ポイントが指定可能であること
(d) 外部から HOLD 状態を解除できること
(e) ステートの遷移を上位スケジューラなどに通知可能なこと
(f) データステージングには第三者転送機能をサポートすること

PGI 要件や PGI-WG での議論は，OGF の Web ページで確認できる．提案の要件を PGI 要件が満たすかを確認するため，**図 7** に提案の要件と関連する PGI 要件の対応を示す．要件のレベルでは，PGI 要件は提案の要件をほぼ満たしている．Finished ステートについては満たされていないが，これは PGI-WG に提案する予定である．

次に設計仕様の検討を行った．PGI 仕様を構成する BES などの個別機能の仕様は，PGI-WG の要件を基に各担当 WG が検討しているが，まだ決定に至っていない．しかし，PGI-WG では要件を検討するにあたり想定する機能仕様の議論しており，これも公開している．ジョブのステートモデルについても議論している．この議論では，BES の Running ステートを，Pre-Processing，Processing，Post-Processing ステートに分割し，これらステートと Pending ステートで HOLD 可能としている．また，さらに柔軟なデータステージング，エラーリカバリのために，3 階層の複雑なジョブステートモデルの議論をしていた．Pre-Processing，Processing，Post-Processing は，実質的にはそれぞれ提案手法の DataStageIn，Computing，DataStageOut に相当すると考えられる．しかし，PGI-WG 議論のステートでは，提案手法で必要としている作業領域を削除する前で HOLD することができない．また，Pending ステートで HOLD 可能とする意味も不明なため，実装のジョブステートモデルは，図 4 の提案のジョブステートモデルを採用した．

(2)　JSDL の HOLD 指定拡張仕様

提案手法，PGI とも，HOLD すべきジョブステートはジョブ投入時に指定するものとしている．これを実現するために，PGI ではジョブ定義ドキュメントで HOLD 指定することを要件としている（図 7 の ID: 79，103）．実装設計では，この PGI 要件に従い JSDL に HOLD 指示のための拡張を行った．これを**図 8** に示す．JSDL 仕様ではジョブ記述を，`<Application>`，`<Resources>`など情報カテゴリごとにサブセクションを分けている．従来の JSDL のカテゴリの中には HOLD のようなジョブの流れ制御のカテゴリはないため，サブセクション`<JobFlow>`を設け，その中のサブセクション`<HoldPoints>`で HOLD させるステートを記述し，`<NotificationPoints>`で通知するステートを記述する仕様とした．なお，この拡張は JSDL の拡張仕様規約に従い，独立したスキーマとしている．

(3)　BES 拡張オペレーション仕様

PGI では HOLD を解除するには，ジョブ実行サービスのオペレーションでジョブステート遷移を指示することで行うことを要件としている（図 7 の ID: 80）．提案手法でも同様にジョブ実行サービスのオペレーションで HOLD 状態を解除することを想定している．現在の BES 仕様には，ジョブステートを変化させるオペレーションは存在しないため，新たなオペレーション ChangeActivityState を追加定義した．**図 9** にオペレーション詳細を示す．引数は，ジョブ ID と遷移させたいジョブステートであり，現在

© 2012 Information Processing Society of Japan

```
<JobDefinition xmlns="http://schemas.ggf.org/jsdl/2005/11/jsdl">
 <JobDescription>
  <JobIdentification>
   ...
  </JobIdentification>
  <Application>
   ...
  </Application>
  <Resources>
   ...
  </Resources>
  <DataStaging>
   ...
  </DataStaging>
  <jsdl-pgi:JobFlow:xmlns:jsdl-pgi="http://www.naregi.org/jsdl/2010/04/jsdl-pgi">
   <jsdl-pgi:HoldPoints>
    <jsdl-pgi:ActivityState>Pending</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>DataStageIn</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>Computing</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>DataStageOut</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>Finalizing</jsdl-pgi:ActivityState>
   </jsdl-pgi:HoldPoints>
  </jsdl-pgi:HoldPoints>
  <jsdl-pgi:NotificationPoints>
    <jsdl-pgi:ActivityState>Pending</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>DataStageIn</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>Computing</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>DataStageOut</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>Finalizing</jsdl-pgi:ActivityState>
   </jsdl-pgi:NotificationPoints>
  </jsdl-pgi:JobFlow>
 </JobDescription>
</JobDefinition>
```

図 8　HOLD ポイントを指定する JSDL 拡張
Fig. 8　JSDL extensions for HOLD and Notification points.

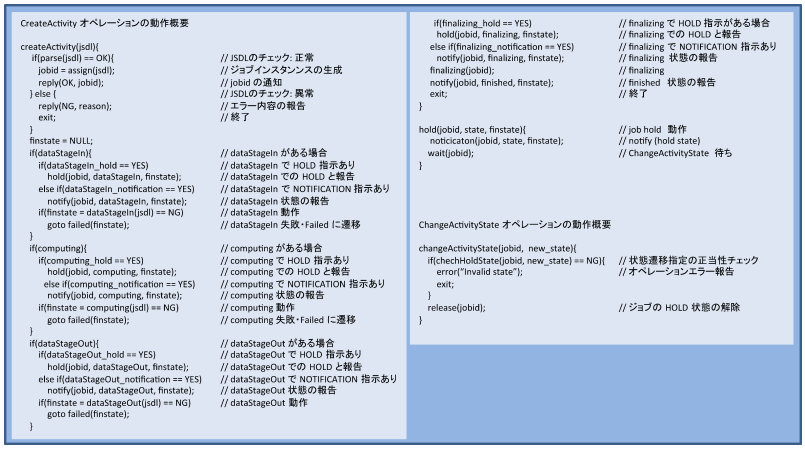

図 10　CreateActivity，ChangeActivityState オペレーションの動作概要
Fig. 10　Overview of CreateActivity and ChangeActivityState operations.

ChangeActivityState

- 変数
  - JobID：　遷移させるジョブのID
  - State：　遷移させるステート
- 動作
  - “JobID”のジョブのステートを”State”に遷移させる
- エラー
  - 認証エラー
  - JobIDのジョブが存在しない
  - 遷移を認められていないステートの指示
  - 未知のステート

図 9　PGI 拡張 BES オペレーション
Fig. 9　PGI extended BES operation.

のステートから指定されたステートへの遷移が許されない場合はエラーとなる．ジョブの HOLD を解除する場合は，HOLD しているステートの直後のステートを指定する．なお，Cancelled ステートへ遷移は既存の BES オペレーションである TerminateActivities オペレーションを使用する．

ChangeActivityStatus オペレーション以外のインタフェース仕様は，従来の BES オペレーションと同じである．しかし，ジョブを投入するためのオペレーションである CreateActivity では HOLD 機能，NOTIFICATION 機能をサポートするために動作が異なる．動作概要を**図 10** に示す．JSDL の HoldPoints 指定，NotificationPoints 指定を検出し，ジョブが当該ステートに遷移したときに，HoldPoints が指定されていればジョブステートの動作を実行する前にジョブを HOLD させて上位グリッドスケジューラに報告する．NotificationPoints で指定されている場合は上位グリッドスケジューラに報告するが，ジョブの動作は継続する．両方が指定された場合は HoldPoints 指定

© 2012 Information Processing Society of Japan

依存関係のあるジョブに対するグリッドスケジューラの動作概要

```
wf_submit(job1.jsdl , job2.jsdl){
  jobid1 = submit (job1.jsdl);                  // job1 の資源ブローカリングと投入
  wait_nofitication(jobid1);                    // job1 の HOLD 待ち
  if(notification .jobid1== OK){                // job1 の計算結果
    jobid2 = submit (job2.jsdl);                // 正常: job2 の資源ブローカリングと投入
  else {
    error(notification.jobid1, NULL);           // 異常: job1 異常終了報告
    exit;                                       //       ワークフロージョブ終了
  }
  wait_notification(jobid2);                    // job2  の NOTIFICATION 待ち
  if(notification.jobid2== OK){                 // job2  の DataStageIn 結果
    release(jobid1);                            // 正常: job1 の HOLD をリリース
    wait_notification(jobid2);                  //       job2  の 終了待ち
  } else {
    release(jobid1);                            // 異常: job1 の HOLD をリリース
    wait_notification(jobid1);                  //       job1  の終了待ち
    error(notification.job1, notification.jobid2);  // job2 異常終了報告
    exit;                                       //       ワークフロージョブ終了
  }
  if(notification.job2 == OK){                  // job2 の終了待ち
    scceess(notification.job1, notificaiton.job2);  // 正常: 正常終了報告
    exit;                                       //       ワークフロージョブ終了
  } else {
    error(notification.job1, notificaiton.job2);    // 異常: 異常終了報告
    exit;                                       //       ワークフロージョブ終了
  }
}
```

図 11　依存関係のあるジョブのグリッドスケジューラの動作概要
Fig. 11　Overview of grid scheduler for jobs with the dependency.

が優先される．HOLD した場合は，上位スケジューラから ChangeActivityState，または TerminateActivities オペレーションが投入されるのを待つ．

(4)　グリッドスケジューラの動作

グリッドスケジューラは図 5 に示したワークフローを実現するよう制御する．依存関係のあるジョブに対するグリッドスケジューラの制御概要を図 11 に示す．ジョブ 1 を実行するための資源をブローカリングしジョブ 1 を投入する．ジョブ 1 が Finalizing の HOLD 状態になるまで待ち，HOLD 報告を受けたら，正常に遷移したことを確認する．正常に遷移していたら，ジョブ 2 を実行するための資源をブローカリングしジョブ 2 を投入する．ジョブ 2 が Computing 状態に遷移した Notification を待ち，正常に遷移したことが確認できたらジョブ 1 の HOLD を解除する．ジョブ 2 の終了を待ちワークフロージョブとしての実行を終了する．

3.3　実験

RENKEI プロジェクトで研究開発した HPCBP 計算資源とクライアントに 3.2 節の設計仕様を基に提案手法を実装し，ジョブ投入実験で提案どおり一時アカウントを想定した環境でも計算資源間で直接ファイル転送できることを確認した．また性能測定を実施した．

実験環境を図 12 に示す．NII の東京本部に提案機能を追加した HPCBP クライアントと HPCBP 計算資源を，NII の千葉分館にも提案機能を追加した HPCBP 計算資源を設置した．実験に使用した装置の詳細を図 13 に示す．

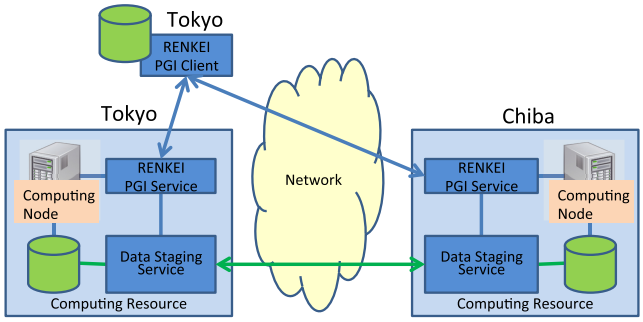

図 12　実験環境
Fig. 12　Experimental environment.

実験 1：動作確認とファイル転送性能測定

計算資源間の直接ファイル転送が実現されていることの確認のために，依存関係のあるジョブで実験を実施した．千葉分館の計算資源のジョブでデータを生成し，これを東京本部の計算資源に転送し，ファイルのサイズを表示するテストアプリケーションである．これを，従来手法と提案手法で実行しジョブ実行時間を比較した．データステージングには FTP プロトコルを使用し，従来手法で必要な一時ストレージには千葉分館に設置した RAID5 構成のファイルサーバを使用した．東京本部，千葉分館の両計算資源とも，ジョブ作業領域は各々のサイトに設置されている RAID5 構成ストレージを持つ NFS サーバ上に割り当てている．

従来手法は，千葉計算資源—(FTP)→一時ストレージ(千葉ファイルサーバ)—(FTP)→東京計算資源の流れの一時ストレージを介したデータステージングとなり，提案手法では千葉計算資源—(FTP)→東京計算資源の直接デー

© 2012 Information Processing Society of Japan

Site	Tokyo Client	Tokyo Computing Resource	Chiba Computing Resource
CPU, No. of CPU	Intel E8400 x 1	AMD 2350 x 2	Intel E5506 x 2
Clock Speed	3 GHz	2 GHz	2.13 GHz
No. of cores/node	2	8	8
No. of nodes	1	1	16
Network I/F	1 Gbps	1 Gbps	1 Gbps
OS, Version	CentOS 5.7	CentOS 5.5	CentOS 5.4
Local Scheduler		PBSProfessional 9.2	PBSProfessional 9.2

図 13　計算資源の構成
Fig. 13　Computing resource configurations.

タステージングとなるため，転送回数が少ない分だけ提案手法のジョブ実行時間は短縮されるはずである．この実験では提案手法のプロトコルによりジョブ実行時間が短縮すること明確にすることが目的のため，実験には実装方法により性能が大きく変わるグリッドスケジューラは使用せず，従来手法，提案手法のプロトコルをプログラミングしたシェルスクリプトによる簡易スケジューラで実験を実施した．実験環境での簡易スケジューラの処理時間は，提案手法の HOLD 処理も含めたワークフローの流れ制御に約 0.1 秒，HPCBP インタフェースによる先行ジョブ，後行ジョブの投入にそれぞれ約 0.3 秒かかり，合計約 0.7 秒の処理時間を要する．また，計算資源の HPCBP サービスの処理時間（ジョブを受信してから実際にジョブの実行が開始されるまでの時間）は，ジョブ情報管理 DB のエントリ生成，作業領域の割当て，ユーザ認証，実行アカウントの割当て，下位ローカルスケジューラのジョブスケジューリングとキューイングなどで構成され，他のジョブが実行されていない状態では約 8 秒必要としている．この時間は本実験環境での値であり，HPCBP サービスが動作する計算機の性能，計算資源の規模，他のジョブの状態，ローカルスケジューラの違いなどに依存する．純粋に計算資源内部の処理時間であり外部との情報交換はないため，ネットワーク性能には影響されない．またこの時間は，先行ジョブ，後行ジョブ各々に必要であるため，HPCBP サービスに使用される時間は合計で約 16 秒となる．したがって，計算，データステージング以外の時間は約 16.7 秒となる．実験結果を**図 14** に示す．これらの結果は 10 回試行した平均である．千葉分館—東京本部間の FTP データ転送性能は，共有ネットワークのため 8 MB/sec～20 MB/sec の幅があり，千葉分館内の計算資源とファイルサーバ間もネットワークを共有しているため，10 MB/sec～30 MB/sec の幅がある．このデータ転送性能に幅があること，一方のデータ転送が同一サイト内であるため，提案手法と従来手法のジョブ実行時間の差がデータ転送 1 回分であることは明示することはできない．しかし，提案手法の JSDL は従来手法より DataStageing エレメントが 1 つ少なく，構造上 DataStageing エレメントの数はデータ転送回数と一致するので，この差分のほとんどはデータ転送回数の低減によるものであると考えられる．本実験により提案手法では従来性能に比べジョブ処理時間が 23%～37%向上したことが確認できた．

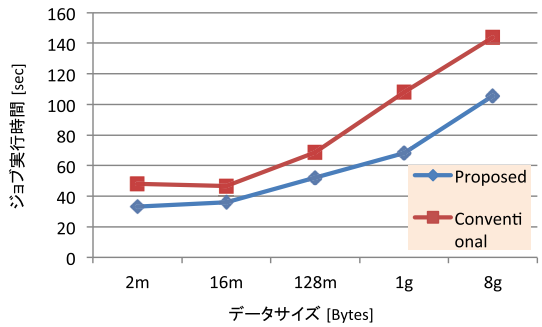

図 14　依存関係のあるジョブの実行時間
Fig. 14　Running time of job with dependence.

実験 2：OpenFOAM

実際のアプリケーションによる実験として，千葉分館の計算資源にインストールされている OpenFOAM でダム崩壊シミュレーションの 3D 拡張版の計算を実施し，東京本部の計算資源上の ParaView で可視化ムービーを作成するジョブを提案手法で実行した．ただし，ParaView はインタラクティブ操作が必要なため，千葉分館の OpenFOAM のジョブ実行開始から東京本部計算資源へのデータステージング終了までを測定した．OpenFOAM は流体シミュレーションアプリケーションであり，多く使用されている．MPI をサポートしており，並列実行による高速化が可能である．本実験での計算規模は 50 万要素であり，計算結果は圧縮して約 350 MB になる．実験結果としてのジョブ実行時間を**図 15** に示す．上記実験環境での計算時間は最短でも 1,447 秒であった（128 並列プロセス）．また，提案手法によるデータステージング時間は平均 30 秒程度であり，ジョブ全体の実行時間のうち，データステージングが占める割合は 2%でしかない．このため，提案手法の従来手法からの性能向上も最大 2%程度であり，OpenFOAM や類似の実アプリケーションでも提案手法が有効であるとはいい難い．これは，実験環境がグリッド環境の大規模計算資源に比べると小規模であり，プロセッサも古く，ネットワークも 1 Gbps のイーサネットでしかないことが原因とも考えられる．そこで，大規模 HPC 環境での有効性を推測する

© 2012 Information Processing Society of Japan

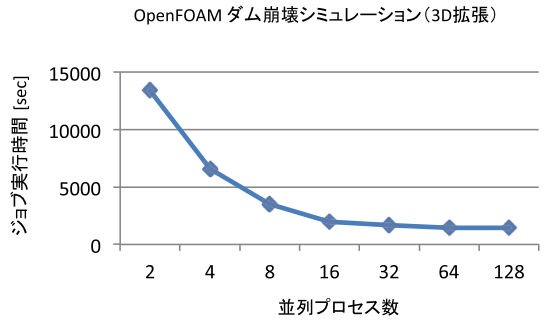

図 15　OpenFOAM のジョブ実行時間
Fig. 15　Running time of an OpenFOAM job.

ために，東京工業大学の TSUBAME 2.0 で同じ計算を実行した結果で検討する．ただし，TSUBAME 2.0 へのジョブ投入は HPCBP インタフェースではなく TSUBAME 2.0 のジョブ投入方法に従ったものである．また，計算結果はフロントエンドサーバを介して手動で転送するしかないため，純粋な計算時間のみで検討する．実行したジョブの並列プロセス数は 256 であり GPU は使用していない．このときの計算実行時間は 156〜400 秒の間である（混み具合により異なる）．もし，TSUBAME 2.0 の 256 コア分のシステムが著者らの実験環境にあり，同じ計算性能が得られると仮定すると，ジョブ実行時間における提案手法のデータステージング時間が占める比率は 16%〜7%と推測できる．また，従来手法でのデータステージング時間が占める比率は最大 28%〜13%と推測できる．データステージングが占める比率の 28%から 16%への低下は有意な差であり，提案手法によるデータ転送回数の低減が有効であることが推測できた．

4. おわりに

従来の BES ジョブステートモデルの拡張し，各ジョブステートで HOLD サブステートをサポートし，そのジョブステートを外部から制御可能とすることにより，一時アカウント環境での依存関係のあるジョブにおいても，余分なファイル転送を排除できる手法を提案した．また提案手法の要件は，現在 OGF で検討されている次世代インターオペレーション仕様 PGI の要件でほぼ満たすことができ，PGI 仕様完成時には，標準仕様で機能を実現できる可能性があることを示した．また，提案手法を RENKEI プロジェクトの HPCBP 計算資源上に実装し，ジョブ投入実験によりその有効性を示した．なお，一部 PGI 要件では満たすことができない提案手法の要件については，OGF の PGI-WG ならびに担当 WG に提案していく予定である．

謝辞　本研究は，文部科学省の科学技術試験研究委託事業による委託業務：次世代 IT 基盤構築のための研究開発「e-サイエンス実現のためのシステム統合・連携ソフトウェアの研究開発」の一部として実施した．また，OpenFOAM のダム崩壊シミュレーションの 3D 拡張版ソースコードと TSUBAME 2.0 での実行結果は明星大学田中義一氏にご提供いただいた．

参考文献

[1] ATLAS EXPERIMENT: A WEB Page, CERN (online), available from ⟨http://atlas.ch/⟩ (accessed 2012-01-15).
[2] ITER 国際熱核融合実験炉：A WEB Page, 日本原子力研究開発機構 (online), available from ⟨http://www.naka.jaea.go.jp/ITER/⟩ (accessed 2012-01-15).
[3] Newhouse, S.: EGI Federating Virtualized Resources, OGF (online), available from ⟨http://www.ogf.org/SAUCG/materials/2342/OGF-32-CloudWS-Newhouse.pdf⟩.
European Grid Infrastructure: A WEB Page, EGI (online), available from ⟨http://www.egi.eu/⟩ (accessed 2012-01-15).
[4] The Enabling Grids for E-sciencE: A WEB Page, EGEE (online), available from ⟨http://www.eu-egee.org/⟩ (accessed 2012-01-15).
[5] Distributed European Infrastructure for Supercomputing Applications: A WEB Page, DEISA (online), available from ⟨http://www.deisa.eu/⟩ (accessed 2012-01-15).
[6] NORDUGRID: A WEB Page, NorduGrid (online), available from ⟨http://www.nordugrid.org/⟩ (accessed 2012-01-15).
[7] Extreme Science And Engineering Discovery Environment, available from ⟨https://www.xsede.org/⟩ (accessed 2012-01-15).
[8] Open Grid Forum: A WEB Page, OGF (online), available from ⟨http://www.ogf.org/⟩ (accessed 2012-01-15).
[9] Dillaway, B. et al.: HPC Basic Profile, Version 1.0, OGF Document, GFD-R-P.114, (2007).
[10] Production Grid Infrastructure: A WEB Page, OGF (online), available from ⟨https://forge.ogf.org/sf/projects/pgi-wg⟩ (accessed 2012-01-15).
[11] PGI-WG GridForge: A WEB Page, OGF (online), available from ⟨https://forge.ogf.org/sf/projects/pgi-wg/⟩ (accessed 2012-01-15).
[12] Anjomshoaa, A. et al.: Job Submission Description Language (JSDL) Specification, Version 1.0, OGF Document, GFD-R.136 (2008).
[13] Humphrey, M. et al.: JSDL HPC Profile Application Extension, Version 1.0, OGF Document, GFD-R.111 (2007).
[14] Foster, I. et al.: OGSA Basic Execution Service OGSA Basic Version 1.0, OGF Document, GFD-R-P.108 (2008).
[15] Tuecke, S. et al.: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, (online), available from ⟨http://www.ietf.org/rfc/rfc3820.txt⟩ (accessed 2012-01-15).
[16] Web Services Security Username Token Profile, Working Draft 2: OASIS (2003).
[17] European Middleware Initiative: A WEB Page, EMI (online), available from ⟨http://www.eu-emi.eu/⟩ (accessed 2012-01-15).
[18] Partnership for Advanced Computing in Europe: A WEB Page, PRACE (online), available from ⟨http://www.prace-project.eu/⟩ (accessed 2012-01-15).
[19] Uniform Interface to Computing ResourcesE: A WEB Page, UNICORE (online), available from ⟨http://www.unicore.eu/⟩ (accessed 2012-01-15).
[20] Lightweight Middleware for Grid Computing: A WEB

© 2012 Information Processing Society of Japan

Page, CERN (online), available from ⟨http://glite.web.cern.ch/glite/⟩ (accessed 2012-01-15).
[21] Advanced Resource Connector: A WEB Page, NorduGrid (online), available from ⟨http://www.nordugrid.org/arc⟩ (accessed 2012-01-15).
[22] EMI Execution Service page: A WEB Page, CERN (online), available from ⟨https://twiki.cern.ch/twiki/bin/view/EMI/EmiExecution Service⟩ (accessed 2012-01-15).
[23] Globus Alliance: A WEB Page, Globus Alliance (online), available from ⟨http://www.globus.org/⟩ (accessed 2012-01-15).
[24] Amazon Elastic Compute Cloud (Amazon EC2): A WEB Page, Amazon (online), available from ⟨http://aws.amazon.com/ec2/⟩ (accessed 2012-01-15).
[25] Andrew, G.: XSEDE Architecture Overview & Context, OGF (online), available from ⟨http://www.ogf.org/SAUCG/materials/2342/XSEDE-context-v1.0.pdf⟩.
[26] 研究コミュニティ形成のための資源連携技術に関する研究：A WEB Page, 国立情報学研究所 (online), available from ⟨http://www.e-sciren.org/index.html⟩ (accessed 2012-01-15).
[27] National Research Grid Initiative: A WEB Page, 国立情報学研究所 (online), available from ⟨http://www.naregi.org/project/index.html⟩ (accessed 2012-01-15).

佐賀 一繁　（正会員）

1956 年生．1981 年明治大学大学院工学研究科電気工学専攻博士前期課程修了．同年富士通（株）入社．2004 年から 2012 年にかけて国立情報学研究所特任研究員．2008 年総合研究大学院大学複合科学研究科情報学先攻博士後期課程入学．2011 年から Open Grid Forum - Grid Interoperation Now Community Group 共同議長．現在，富士通（株）にて次世代テクニカルコンピュータの研究開発に従事．電子情報通信学会，日本神経回路学会各会員．

合田 憲人　（正会員）

1996 年早稲田大学大学院理工学研究科博士後期課程単位取得退学．1997 年博士（工学，早稲田大学）取得．1992 年早稲田大学情報科学研究教育センター助手．1997 年東京工業大学大学院情報理工学研究科数理・計算科学専攻助手．1999 年同大学院総合理工学研究科知能システム科学専攻講師．2003 年同研究科物理情報システム専攻助教授．2007 年国立情報学研究所特任教授，現在に至る．科学技術振興機構さきがけ研究員（2001～2005 年）．ハワイ大学 Information and Computer Sciences Department 客員研究員（2007 年）．東京工業大学大学院総合理工学研究科物理情報システム専攻連携教授（2007 年～現在）等を兼任．並列・分散計算技術に関する研究に従事．電子情報通信学会，電気学会，IEEE，ACM 各会員．

三浦 謙一

1968 年東京大学理学部物理学科卒業．1973 年米国イリノイ大学計算機学科博士課程修了（Ph.D. Computer Science）．同年富士通（株）入社．1981 年スーパーコンピュータ推進室長付．1992 年富士通アメリカ社副社長兼スーパーコンピュータ部門事業部長．1998 年富士通コンピュータ事業本部技師長．2002 年（株）富士通研究所フェロー（～現在）．2003 年国立情報学研究所教授．リサーチグリッド研究開発センター長．2011 年国立情報学研究所名誉教授兼特任教授，現在に至る．この間九州大学客員教授（2000～2003 年）．国立天文台客員教授（2005～2007 年, 2008～2010 年）．文部科学省技術参与（2006～2008 年）等を兼任．スーパーコンピュータ・アーキテクチャ，並列・ベクトル計算アルゴリズム，グリッドコンピューティング，計算物理学等に関する研究に従事．日本工学アカデミー会員．

© 2012 Information Processing Society of Japan