グリッドを用いた分散環境における
バイオインフォマティクスツール
実行支援に関する研究

2008 年 4 月

木 戸　善 之

グリッドを用いた分散環境における
バイオインフォマティクスツール
実行支援に関する研究

提出先　大阪大学大学院情報科学研究科

提出年月　2008 年 4 月

木 戸　善 之

内容梗概

ライフサイエンス研究は生命機能を解明する分野であり，生物学の範囲にとどまることなく，医学，薬学，化学，農学などの学術分野，さらには商業的応用の観点からも産業にも浸透している．ライフサイエンス研究では共同利用施設を利用した共同研究に期待が寄せられている一方で，他の研究者や研究機関に研究対象を知られたくないという機密性確保の要求もある．そこで本研究ではライフサイエンス研究における研究組織間で協力体制を築く手段の 1 つとしてグリッドコンピューティングを用いた共同利用システムを提案した．本研究では様々な組織の研究者が利用する共同利用施設におけるデータの機密性確保について取り組み，GSI–SFS を利用したシステムによりファイルパス名の隠蔽化を行った．それにより他のユーザにはどのようなファイルを利用しているかを知られないシステムを開発し，このシステムによりユーザは他の組織にどのような研究をしているかを知られることなく，複数の組織間にまたがる計算機クラスタを利用したデータ解析やゲノムの相同性検索が可能となった．

次に増加し続けるデータに対応するため，データ取得方法について研究を行った．この研究の背景としては，爆発的に増加するゲノムデータの問題が挙げられる．バイオインフォマティクスでのゲノム解析手法では，多くの生物種のゲノムデータとの比較解析が求められており，研究者が利用するデータ量は増加する一方である．このようなゲノム解析の研究では，ゲノムデータベースをローカル環境に展開し，独自のゲノム解析環境を構築する必要がある．ゲノムデータベースを公開しているデータベースサイトでは解析のための Web インタフェースや Web サービスも公開しているが，これらは公開元が定義したデータベースと解析サービスのみしか利用できない．つまり特定のデータセット（例えば特定の生物種）のみに限定した解析を行いたい場合，Web インタフェースでは相同性検索の結果に対して手動で絞込みを行う必要があり手間がかかってしまう．よって研究者らは独自の解析環境を用意する必要に迫られる．従来で

は独自の解析環境のためのデータベースの取得方法としてミラーリングが行われてきた．ミラーリングはデータベースをローカル環境に複製を作る技術であり，データベース全体の複製をとるが，ゲノムデータが爆発的に増加している現状で完全な複製を維持するのは限界がある．以上の問題を解決するため，ゲノム解析環境においてデータステージングの適用を提案した．データステージングは，解析などのツール実行単位であるジョブの開始直前と終了直後に必要なデータファイルを計算機に転送する技術である．データステージングをゲノム解析環境に適用することで，必要なデータを必要なときに取得することができ，ローカル環境の計算機のストレージを圧迫することなく解析が可能となった．実験ではデータステージングで構築した解析環境とミラーリングによって作成された解析環境を比較することで，提案手法の有効性を評価した．データステージングによるファイル転送のオーバヘッドが計測されたが，解析環境で利用する計算ノードを増やすことで，ゲノムデータの比較解析を効率的に行うことができることを示した．

関連発表論文

1. 学術論文誌掲載論文

1-1 Yoshiyuki Kido, Shigeto Seno, Susumu Date, Yoichi Takenaka, Hideo Matsuda: A Distributed-Processing System for Accelerating Biological Research Using Data-Staging, *IPSJ Transactions on Bioinformatics*, Vol. 49, No.SIG5(TBIO4), pp.58–64, 2008.

2. 国際会議 (査読付き)

2-1 Yoshiyuki Kido, Susumu Date, Shingo Takeda, Shoji Hatano, Juncai Ma, Shinji Shimojo, Hideo Matsuda: Architecture of a Grid-Enabled Research Platform with Location-Transparency for Bioinformatics, *Genome Informatics*, Vol.15, No.2 , pp.3–12, 2004.

目 次

図 目 次

表 目 次

第1章　序論

1.1　本研究の背景

人間の生物学的な機能を解明することによって医療や産業に貢献する分野，ライフサイエンス研究は社会にとって非常に重要である．ライフサイエンス研究は，分子構造，有機化合物，細胞，臓器，個体という様々な単位やスケールに細分され，物理学，化学，生物学，薬学，農学，医学といった分野にまたがって研究が進められていた．そのため，分野内の機能解明に比べ，分野複合的な生命機能の解明は遅れている．生命機能を解明するためにはまずゲノム配列情報を読み取り，実験によって遺伝子の構造や機能を解析し，遺伝子機能のデータを得ることから始まる．各生物種のゲノムの解読と遺伝子の機能解析が進むにつれデータは増加し，得られたデータを基に，例えば遺伝子由来の疾患に関わる遺伝子の探索，遺伝子機能に影響する化合物のシミュレーション，毒性解明や薬物候補の絞り込みといった，産業，医療に大きく貢献する研究へと発展する．

これらライフサイエンス研究の進歩が意味するところは，データが急激な増加傾向にあるということである．つまり分子構造，有機化合物，遺伝子配列，ゲノムといった生命機能に関わるデータがそれぞれ年々蓄積されており，データ量は年々増加の一途をたどる．図 1.1 は DNA の公共データベースサイトである NCBI GenBank [1] の DNA 配列の量の推移を示しており，左の縦軸の Sequences は遺伝子の登録数を表している．また右の縦軸は塩基対数を表しており，横軸は時間を表している．このグラフから遺伝子の登録数，塩基対数ともに年々増加していることが分かる．

データ量の増加は，計算機上での解析や検索といったバイオインフォマティクスで利用されるツールの利用を複雑化している．バイオインフォマティクスのツールは，配列の相同性を検索するための BLAST [2]，FASTA [3]，SSEARCH [4] など，またタン

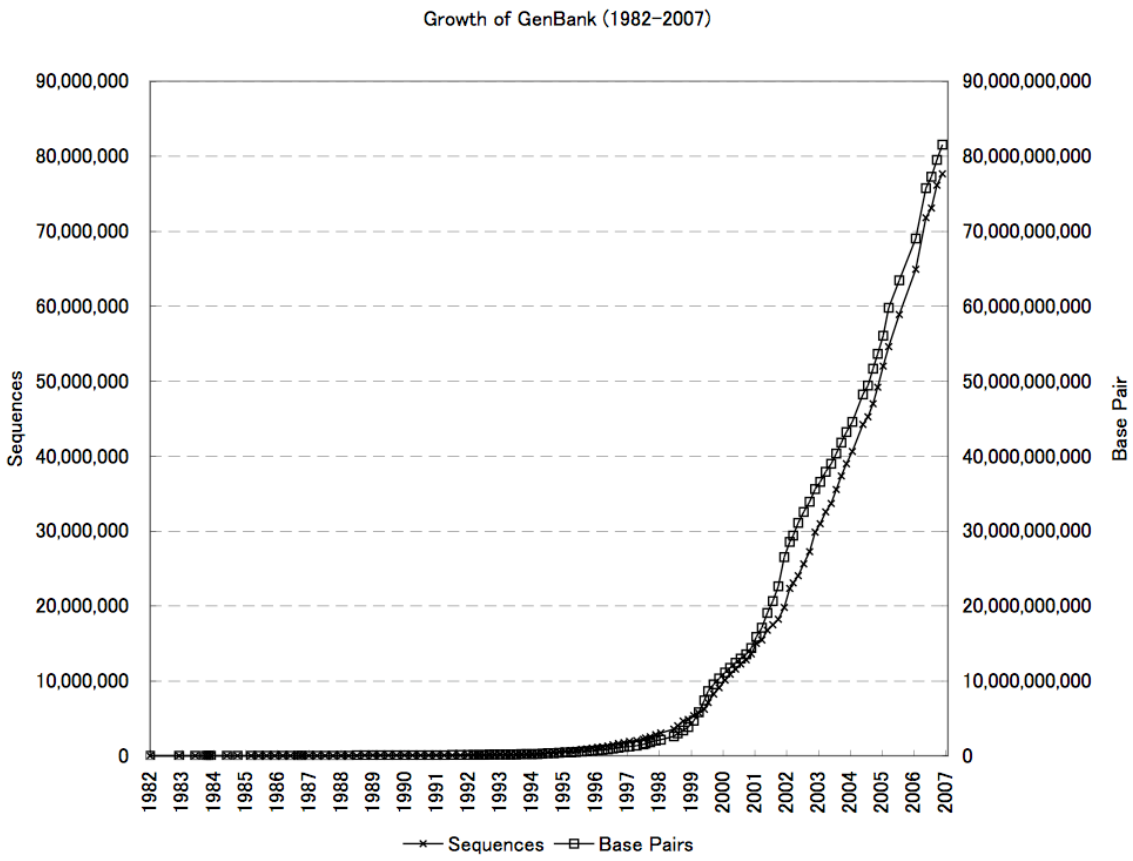

図 1.1: NCBI GenBank の DNA 配列の増加量

パク質と化合物の結合をシミュレーション（ドッキングシミュレーション）するための DOCK [5]，AutoDock [6]，GOLD [7]，myPrest [8] など多岐に渡る．それぞれは用途も異なるが，同一の目的をもつツールも解析の方式や，データ種に応じてアルゴリズムを変更する必要が生じる．また配列の相同性検索などでは相同性をスコアで検出しており，スコア順に結果を表示する．その場合，スコアの閾値をパラメータで渡すことで，相同性の検出を制御し解析を行う．

バイオインフォマティクスでは計算量が大量に必要な高度な配列解析のツールもあり，そのため計算処理やシミュレーションの高速化，高効率化という点で，複数の計算機にて並列に実行させる High Performance Computing (HPC) 分野の利用が進んでいる．HPC では，計算実行プロセスを細分化して，複数の計算機間で並列に実行する並列コンピューティングと，並列に行うプロセス間では通信を行わず閉じた計算を行う分散コンピューティングの研究が進められている．これらの技術の成果はコンピュータクラスタ技術に利用されており，高性能のコンピュータクラスタを低コストで導入することが可能となる．また分散コンピューティングから派生したグリッドコンピューティング技術では，組織間の境界を跨ぎ仮想的な枠組みを形成する仮想組織 (Virtual Organization: VO) というコンセプトを提唱している [9]．高速化という大目標を複数の計算機の統合により実現する一方で，広域に分散している計算機を統合して，組織間で連携し計算資源やデータ資源を共有，互いに有効活用することを提唱している．このようにグリッドコンピューティングは広域にまたがる通信を想定しているため，グリッドコンピューティングの実装である Globus Toolkit [10] は公開鍵基盤による強力なセキュリティコンポーネントを提供している．これらグリッドコンピューティング技術は標準化や実装の充実を目指しオープン・グリッド・フォーラム (OGF) [11] にて活発な議論がなされており，注目されている．

1.2 本研究の目的

本研究では分散した計算機を統合的に利用するグリッド環境下で，バイオインフォマティクスツールの実行支援について取り組み，複雑化するバイオインフォマティクスツールとグリッド環境の利用を簡便にすることを目的とする．

まず，広域に分散した計算機をグリッドコンピューティング技術で統合し，研究者が共同利用するための計算資源へ容易にアクセスできる環境を目指す．具体的には他機関にまたがる研究者らが計算資源，データ資源を共有する共同研究システムを構築する．共同研究において，まず資源の共有化が必要である．そのためグリッドコンピューティングを用い，計算資源の統合，共有化を行う．しかし計算機が共同利用施設などの不特定多数のユーザが利用する場合，他のユーザからも機密性を守る仕組みが必要となる．研究室などで生成された生物学データは，特許や論文につながる新たな知見を含んでいることがあるため，ファイルパス名やデータ名でも機密扱いとしている場合がある．そのためライフサイエンス研究の共同研究において機密性の確保を最重要視されることが多い．また生物学データはデータベースの更新が頻繁に発生すること多く，機密性を確保する既存手法では，ファイル更新時にオーバヘッドが発生することがある．そこで本研究ではファイル更新時のオーバヘッドを抑えた機密性確保の実現を目指す．

次に，増加傾向にあるバイオインフォマティクスでのデータについて着目する．バイオインフォマティクスで扱うデータ量は増加する傾向にある．例えば生物種毎のゲノムデータは年々解析が進められており，増加の一途をたどる．バイオインフォマティクスでのゲノム解析手法では，多くの生物種のゲノムデータとの比較解析が求められており，研究者が利用するデータ量は増加する一方である．このようなゲノム解析の研究では，ゲノムデータベースを研究者らがローカル環境に展開し，独自のゲノム解析環境を構築する必要がある．その一方で，高性能コンピュータクラスタの低コスト化などにより，研究者が利用できる計算機の数は増加している．よって多数の計算機を利用したゲノム解析が可能となりつつある．そこで本研究では，データベースのデータ量が増加している一方で，利用できる計算機が増加している状況を考慮し，より効率的にデータ解析を行える手法の開発を目指す．

1.3 本論文の構成

本論文は 4 章構成である．第 2 章では共同研究施設を利用するために機密性を確保したライフサイエンス研究基盤について述べる．第 3 章では，バイオインフォマティ

クスツールの分散実行環境でのデータステージング手法の提案について述べる．最後に第 4 章で本研究の結論を述べる．

第 2 章では，ライフサイエンス研究を取り巻く共同利用施設の現状と，共同研究の必要性を述べた後，ライフサイエンス研究で求められる機密性について述べる．ライフサイエンス研究では解析するデータが機密扱いの場合が多く，そのため統合する計算機が複数の組織にまたがる場合，通信経路の暗号化による機密性の確保が必要となる．また共同利用施設の計算機では，多くのユーザが利用することが想定され，他のユーザから機密性を確保する必要があり，研究対象名が入ったファイルパス名も他のユーザへ見せたくないという要望がある．そこでライフサイエンス研究において共同利用施設を利用するための機密性を確保，ファイルパス名の隠蔽化について取り組み，機密性確保の既存手法と比較実験を行い，ファイル更新時におけるオーバヘッドの計測を行った．そこで得られた結果を元に考察を行う．

第 3 章では，データ取得方法について提案を行う．そのためまずライフサイエンス研究に関わるデータベースの現状について説明する．またデータベース全体ではなく，多様な個別データの必要性についてタンパク質–タンパク質相互作用のネットワーク解析の研究を例として述べる．タンパク質–タンパク質相互作用の研究ではタンパク質の相互作用の機能解明を試みている．相互作用の実験により得られたネットワークの特性を調べるために，他の生物種と対象になるタンパク質に対し相同性検索をかけて，ネットワークの次数と生物種間での保存度との間の相関を調べている．この研究では BLAST を利用し，研究対象になるタンパク質の配列を問合せとして，完全ゲノムの DNA 配列に対し相同性検索を実行している．つまり解析の実行にはデータベース全体が必要となるのではなく，生物種毎の個別データが必要になり，個別データ毎に解析を実行する必要がある．このようにライフサイエンス研究で求められるデータに対し，現状のデータ取得方法である Web サービスやミラーリングなどの問題点を述べ，データステージング手法を提案する．データ解析を行うときに，データを自動的に取得するデータステージングを利用した分散実行システムを設計，開発について述べ，既存手法である Web サービスやミラーリングとの比較実験を行い，解析時間から考察を行う．解析時間の計測にはタンパク質–タンパク質相互作用の研究で使われている生物種の DNA 配列に対する相同性検索を利用した．

最後に第 4 章で，本研究の成果をまとめるとともに，本研究の成果の波及効果を含めた今後の課題について述べる．

第2章　ライフサイエンス共同研究基盤システム

2.1　はじめに

ライフサイエンス研究での共同研究の必要性は重要視されている．共同研究の利点は，人材，研究機材，データなど単独では得られない資源を共有することができる点にある．DNA 配列データやタンパク質の立体構造など公共のデータベースに蓄積されているデータもあれば，実験で得られた遺伝子配列など，独自に得られたデータを保持している研究室も多い．それらを研究室単位で閉じた研究をするだけにとどまらず，他の大学，研究所と共同で研究を進めることで新たな考察を生む可能性があり，ライフサイエンス研究の進展が期待できる．またライフサイエンス研究で利用する機器は，遺伝子配列を読み取る DNA シーケンサや，脳磁図測定装置などの医療機器に至るまで，高額かつ特殊なものも多い．主に材料科学などで利用される超高圧電子顕微鏡はナノスケールでの試料の解析ができ，細胞などの立体構造解析に役立つ機器である．しかし一度設置された超高圧電子顕微鏡は動かす事は困難であり，高額ゆえに研究室単位で購入，維持できるものでもない．また高エネルギー物理学の衝突実験で利用される SPring 8 の大規模放射光装置なども直径が約 8km ほどの大規模な機器であり，これら設置された組織，場所に限らず共有して，様々な研究者に利用してもらいたいという要望も少なからず存在する．そのためこうした大規模な機器は共同利用の施設として設置されている場合が多い．

一度設置されると容易に動かす事ができない機器としては，大型計算機もその 1 つである．つまり特殊な計測機器，大型計算機などを共有する場合，広域ネットワークを通じて共有する手段が必要となる．インターネットの普及は研究室単位に留まらず，家庭においても普及しており，様々な場所からインターネットを通じて計測機器や計

算機にアクセスすることが可能となっている．しかし物理的なネットワークで繋がっているだけであり，インターネットを介してデータ通信のやり取りを行うには様々な課題をクリアする必要がある．

本章では共同利用施設を利用する際のライフサイエンス共同研究における問題点，機密性について取り組む．共同利用計算機センターでの機密性を確保するシステムを構築し，共同利用計算機センターの多数ある計算資源の利用を目的とする．2.2 節ではライフサイエンス共同研究の必要性と，共同研究基盤システムを構築するための技術課題について述べる．2.3 節では構築する環境のアーキテクチャについての詳細を述べ，2.4 節にて評価と考察を述べる．最後に 2.5 節にてまとめる．

2.2 ライフサイエンス共同研究での技術課題

2.2.1 ライフサイエンス共同研究での機密性

ライフサイエンス研究でのデータ増加にともなう，データ資源，ストレージの統合の観点から多数の組織間での共同研究に期待されている．しかしながらライフサイエンス研究において共同利用施設の利用に消極的な場合が多い．その理由としては機密性の問題が挙げられる．ライフサイエンス研究での共同研究では，通常よりも高い機密性の確保が求められる．ライフサイエンス研究で得られる DNA 配列などのデータは，生命機能の新たな発見につながり，機能解明によって医療，創薬に発展する可能性を秘めている．医療や産業に発展しうる萌芽であるデータを共同研究の場にさらすことは，競争社会において研究者らの研究を脅かしかねない．特に創薬の分野においては新薬開発の競争が激化しており，研究対象としている疾患名が露見することを恐れる組織もある．そのためライフサイエンス研究者らは共同利用計算機センターの利用に消極的であり，遺伝子配列などのデータを提示することを拒む研究者も少なくない．

こうした理由から研究者らが共同利用計算センターに，十分な機密性の確保ができることを要求として出す場合が多い．通信の傍受や盗聴などからデータの通信を守る手段も必要だが，計算資源を利用する上での機密性の確保も必要となる．特に不特定多数が利用する計算機センターについては，利用するユーザ間でも機密性を要求する．

計算機センターには誰が利用するか判らず，他のユーザから決してデータにアクセスできない保証がない限り，ライフサイエンス研究のデータを利用することは難しい．またライフサイエンス研究は競争社会であり，創薬などの企業では利用しているデータのファイル名やディレクトリの階層すらも露見することを嫌う場合がある．データのファイルパス名は対象としている疾患や化合物を想起させるため，研究対象が露見すると考えるからである．ライフサイエンス研究で使用するゲノムデータの実際のファイルパス名の例を図 2.1 に示す．

```
/drug_design/target/Helicobacter_pylori_26695
/drug_design/target/Helicobacter_pylori_HPAG1
/drug_design/target/Helicobacter_pylori_J99
...
```

図 2.1: ファイルパス名の例

図 2.1 はヘリコバクター・ピロリのゲノムのファイルパス名である．ヘリコバクター・ピロリはプロテオバクテリアの一種であり，胃の病原体としても有名である．すなわちこのファイルパス名からは，胃の疾患，創薬に関する研究を行っている事が，露呈することとなる．こうした研究対象を特定する情報を全て機密として扱う必要がある．そのためライフサイエンス研究での計算機センター利用には，通信経路の暗号化はもとより，計算資源にデータを残さない工夫，計算資源を利用する上で他のユーザからデータを保護，ファイルパス名などの隠蔽化する手段といった，より特殊な機密性の確保を求められる．

さらに 1.1 節で述べたように，ゲノムデータなどライフサイエンス研究で利用するデータベースは日々更新が発生し，データ量が増加している．それはショットガン配列決定法など DNA 配列を決定する技術が向上，高速化し，公共データベースへ登録される DNA 配列の数は年々増えているためである．従って更新頻度の高いファイルに対し機密性を確保する必要があり，本研究では共同利用計算機の利用を想定し，更新頻度の高いファイルに対する機密性の確保，特に他のユーザに対するファイルパス

名の隠蔽化に対し取り組む．以下の節で共同研究基盤システムを構築する上での機密性確保するための技術要素について述べる．

2.2.2 機密性の確保への技術課題

ライフサイエンス研究では多数のファイルを用いることが多い．例えばゲノムのデータベースは，生物のカテゴリごとにファイルが細分化され，また DNA 配列毎にデータファイルとして整理されていることがある．またライフサイエンス研究で利用されるデータはゲノムデータのように更新頻度が高いものが多い．そのため機密性を確保する上での更新時におけるオーバヘッドについても考慮する必要がある．

一方，複数の計算機を利用したゲノム解析が可能となり，多くの DNA 配列を解析する場合，複数の計算機を用い並列処理にて効率化を図ることが多い．複数の計算機を用いた解析ではデータファイルが計算機間にまたがるため，分散した計算機における統合的なデータファイル管理の仕組みが必要となる．複数の計算機間でのファイル管理の仕組みとしては分散ファイルシステムの技術を用いる場合が多い．分散ファイルシステムは多くの実装があり，Network File System (NFS) や，Gfarm [12]，Self–certifying File System (SFS) [13] といったものが挙げられる．NFS は最も多くのオペレーティングシステムが実装している分散ファイルシステムであり，NFS クライアントから NFS サーバへマウント操作を行い，Portable Operating System Interface for UNIX (POSIX) 互換の Application Programming Interface (API) でのファイルに読み書きが可能となる．ユーザは NFS クライアントを通じ NFS サーバのマウントポイントのファイルに対し読み書きが可能となる．しかしながら NFS では通信の暗号化復号は行われず，機密性は確保できると言えない．

Gfarm はメタデータを利用した分散ファイルシステムであり，データの送受信に暗号化復号が可能である．またメタデータを用いることによって，ファイルパス名を隠蔽化することも可能である．メタデータによってファイルパス名を論理名と物理名で管理し，認証されたユーザにしか論理名の物理名への写像ができないようにすることによって，ファイルパス名を他のユーザから隠蔽化することができる．機密性を確保すべき情報は論理名でのみ表現し，物理名は Hash 値などによる複雑な文字列とする

ことで，他のユーザからはファイルパス名を隠蔽することができる．Gfarm クライアントは Gfarm サーバ上にあるファイルへアクセスする場合，まずメタデータに問い合わせ，論理名から物理名への写像を得る．物理名を得た Gfarm クライアントは Gfarm サーバ上のファイルへ読み書きを行う．このようにユーザは物理名を意識することなく論理名で Gfarm サーバ上のファイルへ読み書きすることができる．

Gfarm を用いたファイルパス名を隠蔽化する方法の概要図を図 2.2 に示す．共同利用施設側の計算ノードからデータにアクセスし，計算ノードで解析を行う事を想定しているため図 2.2 では共同利用施設に適用した例を用いている．以下，実際に解析を実行するときの手順を示す．

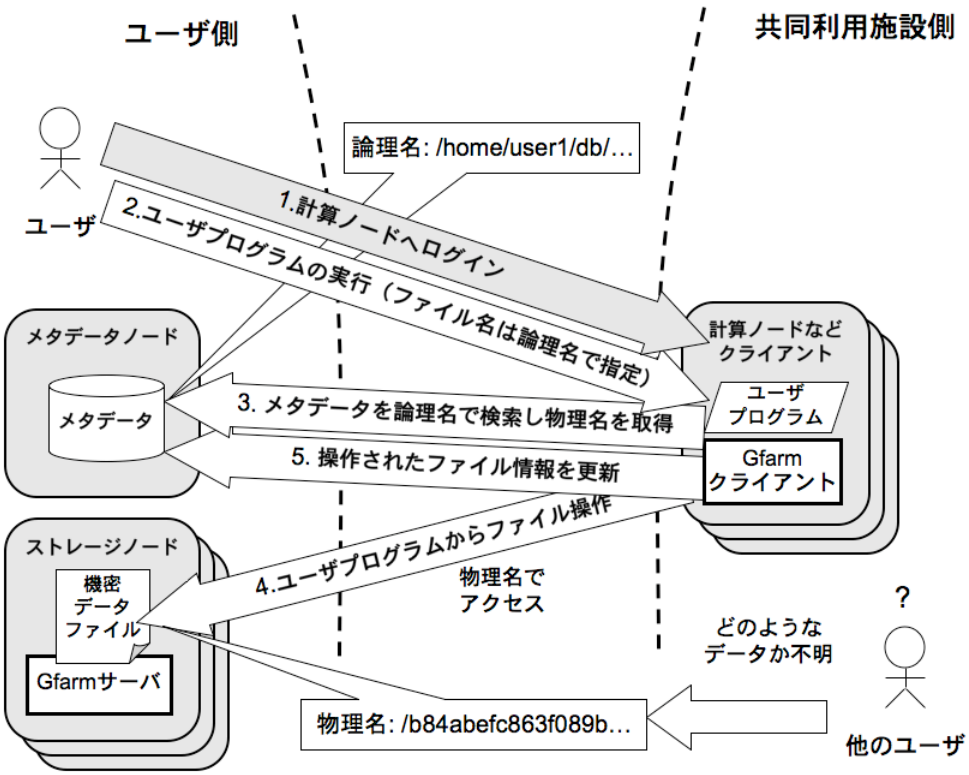

図 2.2: Gfarm を用いたファイルパス名隠蔽化の概要図

1. ユーザは共同利用施設へログインする．

2. ユーザプログラムを実行する．このときデータのファイルパス名は論理名で指定

する．

3. Gfarm クライアントは，指定されたファイルパス名の物理名を得るためにメタデータを検索する．

4. Gfarm クライアントは得られた物理名で Gfarm サーバへファイルアクセスを行う．

5. ユーザプログラムによってファイルが更新された場合，メタデータへもファイル情報の更新を行う．

ユーザはストレージノードに機密データを配置するが，NFS や FTP などのファイル転送技術であると，ファイル名が他のユーザから判ってしまう．Gfarm を利用することによって，メタデータで物理名と論理名を管理することが可能となり，物理名は意味情報を含まないファイルパス名，論理名にはデータの意味情報を含んだファイルパス名を付けることができる．論理名から物理名への写像は Gfarm クライアントが行うので，ユーザは物理名を意識すること無くファイルアクセスが可能となる．また計算ノードを利用する他のユーザからは論理名から物理名への写像ができないため，物理名でしか見ることができなくなり隠蔽化が可能となる．しかしメタデータを利用することによってファイルパス名の隠蔽化は可能になるが，ファイル更新時におけるオーバヘッドが問題となりうる．機密データのファイルが更新されたときに，Gfarm ではメタデータへの更新も発生する．つまり更新ファイルが多い場合，ファイル更新時においてオーバヘッドも増加すること考えられる．

SFS を利用しファイルパス名を隠蔽化することもできる．SFS では物理的にマウントを行うが，マウントポイントをユーザごとに設ける事によって，他のユーザからマウントポイントを隠蔽化する．SFS を用いマウントポイントをユーザごとに設けファイルパス名を隠蔽化する方法を共同利用施設に適用した例を図 2.3 に示す．以下，実際に解析を実行するときの手順を示す．

1. ユーザは共同利用施設へログインする．

2. SFS クライアントに置いた秘密鍵のパスフレーズの入力を行い，SFS サーバに対する認証とマウントを行う．この操作は SFS サーバごとに必要．

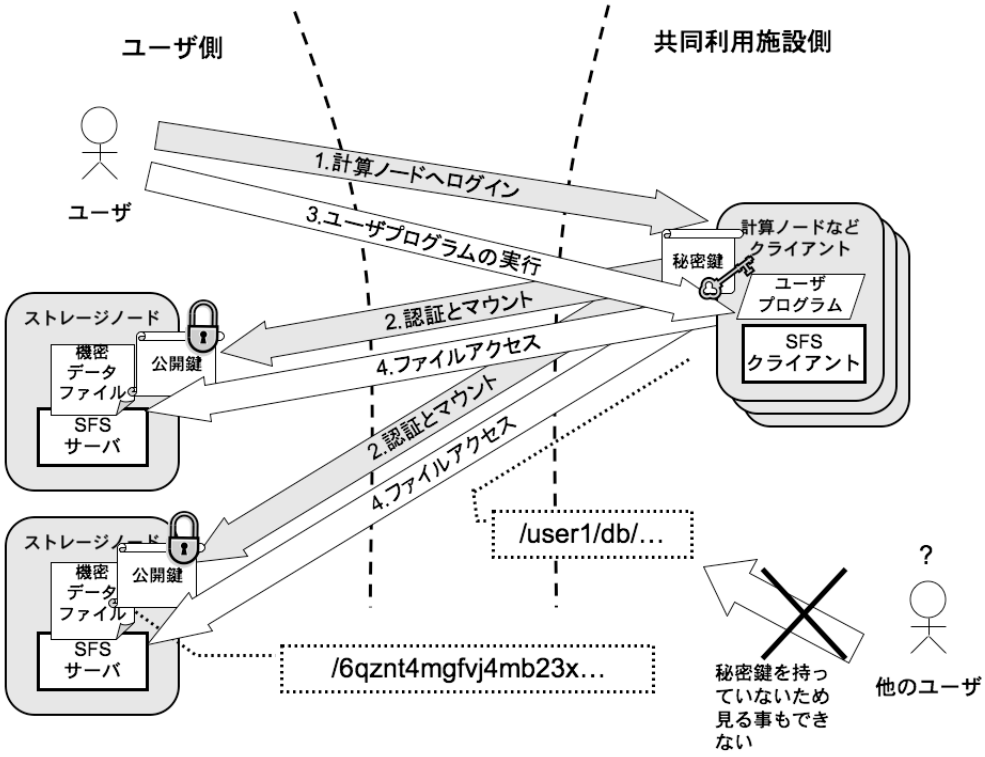

図 2.3: SFS を用いたファイルパス名隠蔽化の概要図

3. ユーザプログラムを実行する．

4. ユーザプログラムは既にマウントしているマウントポイントに対し，ファイルアクセスを行う．

SFS は NFS の通信経路を暗号化し，マウントしているホスト名やマウントポイントであるディレクトリ名は SFS クライアントが管理している．マウントポイントを公開鍵暗号方式で管理しており，ユーザはクライアントに秘密鍵を，サーバ側に公開鍵を置き，マウントするごとに公開鍵認証を行う．つまり秘密鍵を持っていない他のユーザはマウントポイントにアクセスすることが出来ず，マウントポイントすら見る事ができない．それは計算機の管理者であっても同じである．しかし SFS ではマウントするストレージノードごとに個別の認証作業が必要となり，利便性を損なう恐れがある．データ量が増加し続けるライフサイエンス研究ではストレージノードを増加することも想定する必要があり，複数のストレージノードに対し認証作業が発生することは，ユーザへの負担が増加することとなり，利便性が損なわれていると言える．さらに大きな問題としては計算ノード側からストレージノードへファイルアクセスを行う必要があるため，計算ノードに SFS のクライアント，ストレージノードに SFS サーバをそれぞれ設置しなければならず，つまりユーザは計算ノード側にユーザの秘密鍵を置く必要がある．共同利用施設に秘密鍵を置く事は，他のユーザに秘密鍵を奪取される可能性が高くなるため，避けるべきである．

2.3 機密性を確保する共同研究基盤システムの提案

機密性を確保する上で発生するファイル更新時間の課題，利便性の課題を解決する手段として GSI–SFS [14] を適用することを提案する．GSI–SFS は SFS を拡張し，GSI (Grid Security Infrastructure) [15] の認証で利用できるようにした分散ファイルシステムである．GSI とは，公開鍵基盤 (Public Key Infrastructure: PKI) を応用し，電子証明書を用いたシングルサインオンを実現する技術である．シングルサインオンとは一度きりのユーザ認証行為で，複数の計算機が利用できる認証を指す．つまりユーザは複数の計算機を利用する際に，最初に利用する計算機に対しログインし，認証行

為としてパスワードを入力するのみで，他の計算機へは認証行為を必要とせずに利用できるようになる．GSI の最大の特徴は電子証明書を直接認証に使うのではなく，プロキシ証明書を利用する点にある．プロキシ証明書とはオリジナルの電子証明書の有効期間に制限 (通常は 12 時間程度) をかけた電子証明書である．もちろん認証局の署名も継承され，GSI 認証にはプロキシ証明書を利用することとなる．プロキシ証明書の発行には，オリジナルの電子証明書にかけられたパスフレーズの入力が求められる．その後，GSI 認証を行う上でユーザはパスフレーズの入力は求められず，計算機へのログインが可能となる．またプロキシ証明書はユーザがログインした計算機へコピーされる．コピーは PKI によって暗号化された通信で送付される．送付されたプロキシ証明書を利用して別の計算機へログインが可能となり，シングルサインオンを実現している．

GSI–SFS では GSI 認証を利用することによって，ユーザはサーバごとに個別に認証作業をすることなしに GSI によるプロキシ証明書でマウントポイントにアクセスすることが可能となる．GSI–SFS では SFS と同様に，クライアントは NFS サーバと連携し，NFS クライアントからは/sfs をマウントしているように見せている．SFS クライアント内部で，秘密鍵とマウントポイントをユーザ単位で管理しており，そのため他のユーザからは/sfs 以下には自身が利用しているマウントポイントしか表示されないようにしている．またマウントしているホスト名をハッシュ値にて隠蔽化し，他のユーザはどこのホストにアクセスしているかもわからない．

また GSI–SFS ではクライアント内，サーバ内でそれぞれプロセス間通信は行うが，クライアントとサーバの 1 対 1 の通信である．また機密性を確保するために必要な処理としては，暗号化復号とマウントポイントの制御のみであり，計算機間での通信を最小限で抑えている．更に GSI–SFS とは NFS と同様の特性も備えているため，POSIX 互換の API からのファイルの読み書きが可能となる．つまりほとんどのツールはプログラムを変更することなく GSI–SFS 上のファイルを読み書きでき，GSI–SFS サーバの計算機上のファイルもローカル上のファイルと同様に扱う事が可能となる．GSI–SFS では SFS の特性をそのままに，認証部分を GSI で行う事により，シングルサインオンをデータアクセス時も実現している．つまり複数のホストへのデータアクセスに対しても，ユーザはプロキシ証明書を利用したアクセスが可能となり，ユーザの利便性を

損なわない．また秘密鍵を共同利用施設に置くことなくストレージノードへのマウントが可能となり，秘密鍵を他のユーザに奪取される可能性もなくなる．

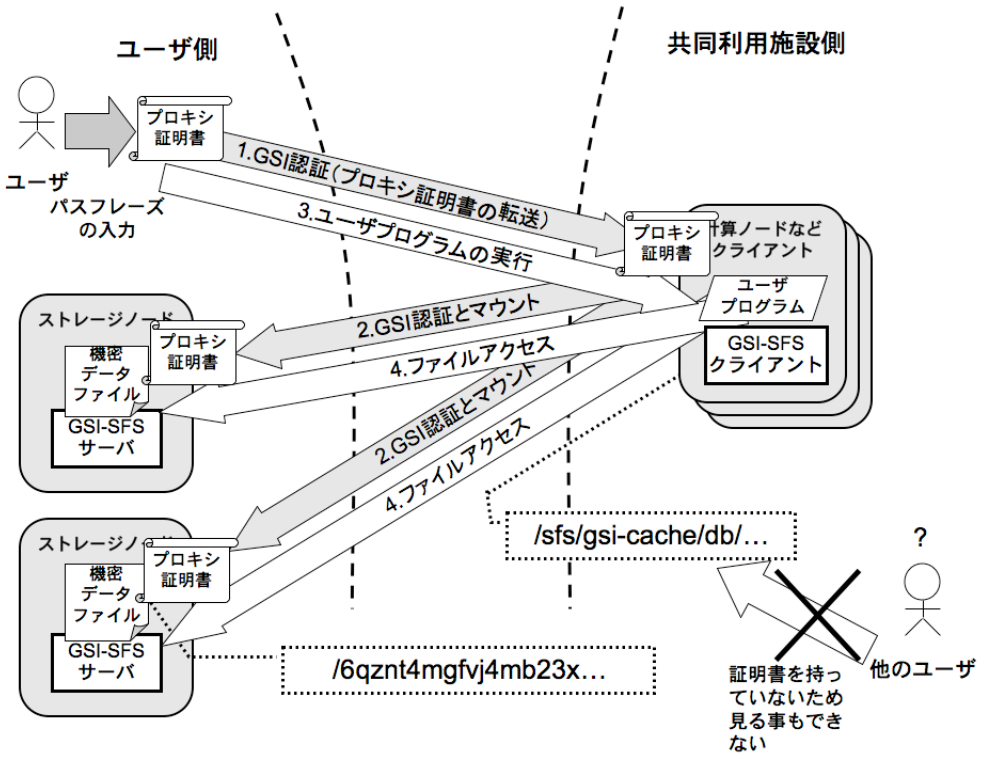

図 2.4: 提案する共同研究基盤システムの概要図

図 2.4 は共同研究基盤システムに GSI–SFS を適用した概要図である．以下，実際に解析を実行するときの手順を示す．

1. 証明書からプロキシ証明書の生成．このときにパスフレーズを入力する．また GSI によりプロキシ証明書は Secure Socket Layer (SSL) [16] によって計算ノードに転送される．

2. GSI–SFS によってマウントを行う．このときも 1. と同様に GSI による認証を行うが，ユーザはパスフレーズを入力することなく認証が完了する．

3. ユーザプログラムの実行要求を行う．

4. ユーザプログラムは既にマウントしているマウントポイントに対し，ファイルアクセスを行う．

共同利用施設の計算機に GSI–SFS クライアントを設定し，ユーザは GSI–SFS を利用して他の計算機にマウントを行い，共同利用施設でユーザプログラムを実行することができる．データファイルは分散したユーザ側のストレージノードに配置し，GSI–SFS 経由で共同利用施設の計算ノードが利用することとなる．ユーザプログラムなどの実行プロセスは GSI–SFS 経由で直接ファイルにアクセスすることができ，共同利用施設の計算機上にはデータファイルを残さない．また GSI–SFS を利用することによって，解析に利用しているデータなどのファイルパス名を他人に知られることなく，解析を行うことができる．また通信経路も暗号化されているため，インターネットを経由した計算機間でのデータ通信が可能となる．

2.4 機密性の評価と考察

機密性を確保するためには様々なオーバヘッドが発生する．ライフサイエンス研究で用いるデータファイルは更新頻度が高いことから，ファイル更新時のオーバヘッドについて着目し，ファイルへのアクセス時間のオーバヘッドを計測した．評価指標としてはメタデータを利用した分散ファイルシステム，具体的には Gfarm との比較実験を行った．GSI–SFS と Gfarm のそれぞれ 1000 ファイル，2000 ファイル，3000 ファイルへのアクセスを行った場合の時間を計測し，図 2.5 に結果をまとめている．それぞれの実験ではサーバにファイルを置き，クライアントから上書き更新を行ったもの (ファイル更新) と，サーバ側のファイルに対し読込みを行うだけのもの (アクセスのみ) に分けて計測した．GSI–SFS ではサーバに配置済みのファイルに対し cp コマンドで上書き更新を行い．サーバ側のファイルへのアクセスのみの実験では同じく cp コマンドにてクライアントの/dev/null に対して行った．Gfarm も同様の実験を行った．ただし Gfarm の場合は cp コマンドと同等の機能を持つ Gfarm 用のコマンドである gfcp を利用した．Gfarm ではファイルアクセスの標準 API である POSIX 互換 API に対応した動的リンクライブラリを用意しており，cp コマンドなどの標準的なコマンドからも操作が可能である．しかし POSIX 互換 API から呼び出すための動的リンクライブラリ

のリンクによるオーバヘッドを避けるために Gfarm 付属のコマンドを利用している．また表 2.1 と表 2.2 は実験に利用した計算機のスペックである．それぞれ 100Mbps のネットワーク環境で行った．計測に利用したファイルは 0 バイトである．実験の目的は機密性確保のオーバヘッドの計測であることから，データ通信によるオーバヘッドや，ディスク I/O など影響を極力避けるために 0 バイトのファイルを利用した．

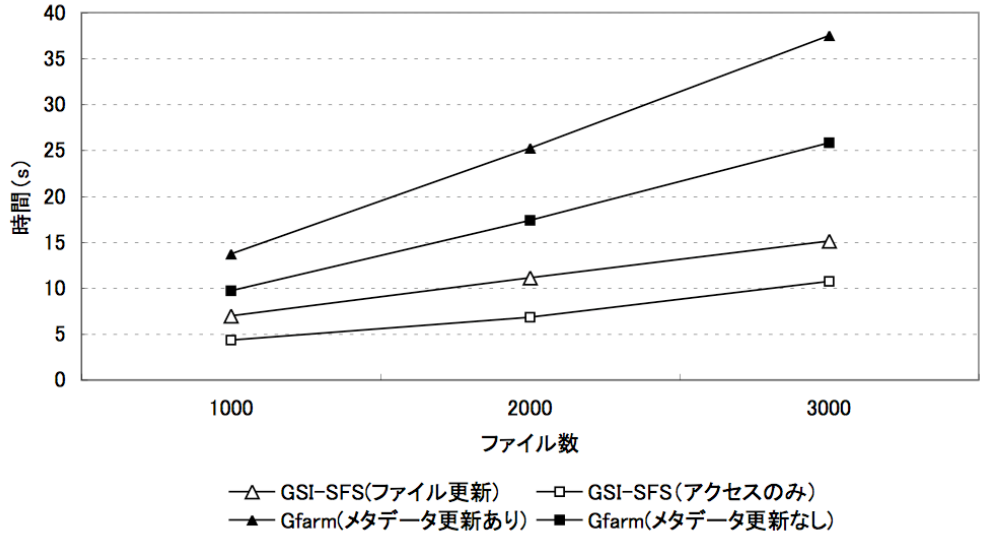

図 2.5: ファイルアクセス時間の比較

表 2.1: GSI–SFS 実験環境計算機スペック一覧

	GSI–SFS サーバ	GSI–SFS クライアント
CPU	Intel Celeron	Intel Core 2 Duo
周波数 (GHz)	1.40	1.40
主記憶容量 (MByte)	512	2048

Gfarm ではファイル数が増加するごとにファイルアクセスにかかる時間が増加しているのが判る．GSI–SFS もまたファイルアクセスにかかる時間が増加するが，Gfarm よりも増加率が低いことがわかる．GSI–SFS の場合は，ファイル更新時とファイルアクセスのみでは，ほぼ同等の時間で終えている．一方，Gfarm ではファイルアクセス

表 2.2: Gfarm 実験環境計算機スペック一覧

	Gfarm サーバ	Gfarm メタデータ	Gfarm クライアント
CPU	Intel Celeron	Intel Celeron	Intel Core 2 Duo
周波数 (GHz)	1.40	1.40	1.40
主記憶容量 (MByte)	512	2048	2048

のみの時間に対し，ファイル更新時の方が明らかに時間がかかっていることが判る．その理由としては，Gfarm ではサーバに配置されたファイルに対し更新が発生すると，更新日時やファイル容量など，ファイルに対するメタ情報をメタデータへと更新を行っていることが挙げられる．そのため Gfarm での更新時とアクセスのみの計測では，メタデータへの更新時間の差が計測時間に現れている．一方，GSI–SFS ではサーバとクライアント間で完結しており，またファイルを管理するためのデータベースが存在しないため，物理的なファイル更新のみのオーバヘッドである．よって GSI–SFS を利用した場合，更新ファイル数が増加した場合のアクセス時間のオーバヘッドが抑えられている結果となった．ファイルを多く利用するライフサイエンス研究，特にゲノムデータを取り扱う上では更新も頻繁に発生しうる．そのためライフサイエンス研究環境で利用するには，Gfarm よりも GSI–SFS を適用した環境でファイルパス名を隠蔽化を実現する方がファイルアクセス時間のオーバヘッド，特にファイル更新時のオーバヘッドを抑えられると言える．

2.4.1 今後の課題

ファイルパス名の隠蔽化については課題が残る．それは計算機上の実行プロセスを表示する ps コマンドなどによるものが挙げられる．ps コマンドでは不特定多数の実行プロセスのコメンド名やオプションなどが表示できる．このため BLAST などのオプションにデータファイル名を渡すツールなどの実行時に他のユーザが ps コマンドを実行した場合，ファイルパス名が露呈する恐れがある．こうした機密性の確保を阻害するコマンドについては，ユーザごとの実行制限などによる計算機上の設定で制限することも可能である．

2.5 おわりに

本研究では共同利用施設における他の利用ユーザからデータの機密性を保持するため GSI–SFS を利用し，機密性の確保，ファイルパス名の隠蔽化の実現を目指した．ライフサイエンス研究で取り扱うデータファイルの特徴としては更新が多いことから，更新時のファイルアクセスのオーバヘッドを重要視し，更新ファイルが増加した場合におけるファイルアクセスのオーバヘッドを計測した．その結果，Gfarm を用いたファイルパス名の隠蔽化手法よりも，提案手法である GSI–SFS を用いたファイルパス隠蔽化が，ファイル更新時のオーバヘッドが抑えられることを確認した．また GSI を用いたシングルサインオンを実現したシステムを提案することで，利便性を損なうこと無くファイルパス名の隠蔽化を実現することを示せた．

第3章　データステージングを用いたバイオインフォマティクス実行支援システム

2 章では，機密性の確保に重点を置いたライフサイエンス研究共同基盤システムについて取り組んだ．次に増加しつつあるライフサイエンス関連データの取得方法についての報告を行う．

3.1　はじめに

ライフサイエンス研究で利用するデータは年々増加の一途をたどり，研究で利用するデータ量も増加している．バイオインフォマティクスでは様々なデータを扱う解析ツールが増えており，研究者らは多様化するデータと解析ツールを利用して研究を進める必要がある．例えば創薬のプロセスでは，疾患，ゲノム，タンパク質立体構造，化合物といったデータベースからデータを取得し，ドッキングシミュレーション，モデリングを行い，薬物候補の化合物を絞り込むスクリーニングが行われる．このようにライフサイエンス研究では様々な種類のデータベースが利用されており，公共のデータベースサイトもライフサイエンス関連研究のデータ公開に取り組んでいる．ゲノムデータベースでは EMBL [17], NCBI GenBank [1],DDBJ [18] などが，タンパク質立体構造では PDB [19]，化合物では ChEBI [20] や PubChem [21]，疾患関連データベースでは OMIM [22]，発現プロファイルの GEO (Gene Expression Omnibus) [23]，代謝パスウェイのネットワークでは KEGG [24] など様々なデータベースが公共のデータベースサイトより公開されている．またライフサイエンス研究の文献データベースである PubMed [25]，MEDLINE [26] などもテキストデータマイニングにより利用され

ている．データ量の増加はバイオインフォマティクスツールの利用方法の複雑化を意味する．データによって検索方法や解析方法，アルゴリズムの変更を強いられるため，バイオインフォマティクスによるデータの解析が複雑化，多様化している．

本章では，データ量の増加に起因するバイオインフォマティクスの問題点を明らかにするため，まず 3.2 節ではタンパク質–タンパク質相互作用ネットワークの研究を例として背景を述べる．そしてライフサイエンス研究関連データを取得するための技術要素の課題について述べる．3.3 節では，増加しつづけるデータ量や技術要素の課題に対し，解析の実行時にデータを取得するデータステージングを提案し，その詳細を述べる．3.4 節では，データステージングを用いたバイオインフォマティクスツール実行支援システムの構築について詳細を述べ，3.5 節にて提案手法の評価と課題について述べる．最後に 3.6 節にてまとめについて述べる．

3.2 背景と技術課題

複雑化，多様化するバイオ研究に対応するため主要なデータベースサイトでは，複数のデータベースを統合し横断的な検索システムの研究・開発が進められている．NCBI では Entrez [27] というデータベース統合検索ツールを用意しており，Web インタフェースを通じてユーザが検索キーワードを入力する事で，DNA やアミノ酸配列，タンパク質立体構造，文献データベースなどに対し，網羅的に一括全文検索を行うことができる．Entrez は複数種データの横断的，網羅的検索によって，研究者らは一回の検索でキーワードに関連する情報の一覧を取得でき，研究活動に多大な貢献している．しかしながら研究者らが制限無くカスタマイズした解析を行いたい場合 (例えば検索対象の配列データをデータベース全体ではなく特定のものに限定して DNA 配列の類似性検索など)，データベースサイトが提供する機能やデータだけでは不十分である．なぜなら，研究者らは自身が持っているデータやツールと組合せて利用する場合もあり，独自のデータと公共データベースを組合せて行う解析手段，パラメータを細部まで調整した解析手段を公共データベースサイトで提供することは難しいと言える．次の小節では，公共のデータベースサイトでデータ解析を行うことが困難な研究例について述べる．

3.2.1 バイオ研究におけるデータ解析の特徴

バイオ研究者らは今までよりも多くの生物種への有用性を示す必要性が出てきている．特定の生物種の遺伝子機能に対し，同様の遺伝子もしくは類縁の遺伝子が他の生物種にあるかどうかを調査する過程では，調査する遺伝子の配列を，他の生物種配列データと相同性検索をする必要がある．その顕著な例として大腸菌のタンパク質–タンパク質相互作用ネットワークを解析する研究 [28] が挙げられる．この論文では，推定したネットワークのハブと関係する大腸菌タンパク質配列を，多数の生物種完全ゲノムに対し相同性検索を BLAST [2] で行っている．この解析の結果は，ネットワークのハブになっているタンパク質が 148 種の生物種完全ゲノムにて保存されていることが確認され，またネットワークの結合度と一致したゲノムの数との関連性が明らかにあることを証明している．つまりこの研究では，大腸菌ゲノムの配列に対し 148 種以上の生物種完全ゲノムに対して BLAST を行っているが，他の生物種と検索結果を混在させないために生物種毎に BLAST を実行している．

もう一つの例としては，2 種類の生物種ゲノムを互いに双方向から相同性検索を行い，スコア最高値を取ることでオーソロガス遺伝子もしくはパラロガス遺伝子候補を探索する研究が挙げられる [29]．オーソロガス遺伝子が生物種の進化の過程における種分化によって生じた相同性であるのに対し，パラロガス遺伝子は同一種での遺伝子重複によって生じた相同性である．オーソロガス遺伝子やパラロガス遺伝子を探索することは，未知の生物種や遺伝子に対して機能解明をする重要な要素となる．この研究においては対象となる 2 種の生物種ゲノムを双方向から相同性検索を行う必要があるが，1 回の検索で複数の生物種ゲノムを相同性検索した場合，複数の生物種の遺伝子配列がスコアリングされ，閾値に引っかかり結果を取りこぼす事となる．つまり多生物間での重要な共通点を見落とすことを防ぐために，複数の生物種を混在して相同性検索を行うのではなく，生物種 1 つに対して 1 回以上の相同性検索を行う必要がある．つまり解析の実行単位では，生物種データベース全体が必要ではなく，生物種毎の個別データが必要となる．個別データとはデータベース内にあるデータのサブセットであり，例えばゲノムデータベースにおける生物種毎のデータのことを指す．

これら 2 つの例でこのような複数種間での探索を行う場合，BLAST の実行回数が増

加し，データベースサイトの負荷増加につながるため，ゲノムの大規模解析はデータベースサイトによって提供されるサービスでは困難である．また研究者らが検索したいデータに対して細やかなパラメータ設定やフィルタリングなどや，利用したいデータが限定的である場合，データベースサイトが提供しているインタフェース以上の細かな設定が必要である．さらに研究者らは上述した複雑かつ高負荷である研究方法に対応できる解析環境を求めている．よってこれらの要求を満たすために研究者らは独自に解析環境を構築する必要がある．解析環境を構築する場合，一般的に分散コンピューティング技術を用いた分散環境が適用される．BLAST に代表されるバイオインフォマティクスのツールは複数台の計算機を用い，分散並列に実行することで効率的に結果を得る事ができる．しかし分散した解析環境はデータの配置やデータの取得方法について課題が残されている．そこで次に，データベースサイトから供給されるデータの取得方法について検討するため，以下の節では各技術要素について述べる．

3.2.2 バイオインフォマティクスにおける Web サービス

NCBI GenBank，DDBJ や KEGG といったデータベースサイトは，Web サーバを公開しており，誰でも自由にデータの検索に利用できる．また File Transfer Protocol (FTP) にて加工しない状態のデータファイルと，Web サービス [30] によるプログラマブルな API を用意している．Web サービスとは分散コンピューティング分野の技術の 1 つであり，Remote Procedure Call (RPC) をサービス指向に発展させた技術である．API の定義を Web Services Description Language (WSDL)，通信プロトコルを Simple Object Access Protocol (SOAP) とし，クライアントは Web サービスを公開しているホストに対し API を呼び出す事ができる．それにより WSDL で規定している処理の実行をサービスプロバイダ側に要求し，クライアントは処理結果を得ることができる．処理やデータの加工を可能にする技術であるが，データの取得方法という意味でも要件を満たす事ができる．

しかしながら，サービスプロバイダは WSDL にてインタフェースを定義しなければならず，予め定義された処理や，生物種配列を予め集合して定義したデータセットのみを扱うこととなる．つまり扱う生物種が増加した場合，増加する毎に定義する必要があ

り，その都度インタフェースを更新する必要がある．つまりサービスプロバイダ，データベースサイトに運用負荷がかかる．また定義されたデータセットのサブセットのみ，言い換えればデータの一部のみを利用したい場合，利用するデータ毎にインタフェースを定義する必要がある．研究者らはサービスプロバイダが定義するインタフェースに沿ったデータ取得しかできないので，全ての研究者らの要求を包括する Web サービスを提供するのは困難である．つまり多様な個別データ毎にインタフェースを用意することは現実的ではない．したがってカスタマイズ可能な解析を行いたい研究者は，Web サービスに依存しない独自の解析環境を構築する必要がある．

バイオインフォマティクスにおける Web サービスは主に公共データベースサイトなどで解析や検索を行い，その結果をユーザに返すサービスであり，非常に多くのユーザの利用を想定されている．Web 技術をベースにしているため，HTTP におけるリクエストに対し，解析，検索を実行し，結果をレスポンスで返す仕組みが多い．そのため解析，検索は計算量が軽量である必要があり，サーバ側に負荷がかかるようなクエリーを要求できないようなインタフェースのみの公開が多い．また Web サービスの特徴としては，リクエストに対するレスポンスが常に同期処理にて行われる点にある．つまり解析のリクエストの応答としては，解析結果をレスポンスとして返すこととなり解析結果の送信を制御できない．言い換えると，解析の実行と解析結果の送信が常にシーケンシャルに行われることとなり，解析の実行と解析結果の送信を非同期，もしくは並列して行うことができず，効率的でない．また解析結果の送信を独立することができないことによって，送信プロトコルを選択できず，リクエストに利用されたプロトコルを使用することを強いられる．つまり並列化や効率的な転送プロトコルなどを使用できないこととなる．

3.2.3 バイオインフォマティクスにおけるミラーリング

カスタマイズ可能な解析環境を構築する上で，データの取得方法としてミラーリングが挙げられる．通常，ミラーリングはストレージ間でリアルタイムにデータを同期し，データの保存先を 2 つ以上に設ける技術である．Redundant Arrays of Inexpensive Disks (RAID) などのストレージに関連する技術や，rsync [31] などによる WEB サー

バもしくは大規模ファイルの二重化技術などがある．rsync はデータの差分のみの送信によりネットワークトラフィックの軽減も考慮にいれ，大規模ファイルのミラーリングに適したプロトコルである．高稼働率もしくは負荷分散のための技術であるが，ゲノムデータベースやタンパク質データベースをミラーリングによって共有するプロジェクトもある [32]．Bio-mirror プロジェクトでは NCBI GenBank, EMBL, DDBJ などといった主要なデータベースサイトからデータベースをミラーリングし，主要なサイトの負荷集中を分散することを目的としている．利用するプロトコルは FTP や rsync や Hyper Text Transfer Protocol (HTTP) といった主要プロトコルが利用できる．

ミラーリングによって独自の解析環境にデータベースを複製し解析を行うことでユーザはカスタマイズ可能な解析環境を構築することができる．しかしがならミラーリングにおいてはデータの増加による問題点が挙げられる．3.2.1 節で述べたように，タンパク質–タンパク質相互作用の研究においては，ゲノムデータベース全てはなく，特定の生物種の完全ゲノムが必要となる．GenBank などの公共データベースでは，多くのゲノム解析プロジェクトが遺伝子の登録を行っており，機能未知な配列も多く登録されている．それによりデータベースの総量は完全ゲノムの総量を遥かに上回り，全てのゲノムデータベースをコピーするミラーリングでは無駄が多い．図 3.1 と図 3.2 は，ゲノムデータベースの総塩基対数の伸び率と完全ゲノムの総塩基対の伸び率を表した対数グラフである．縦軸が塩基対の対数メモリに対し，横軸は 1 を 1 年とした軸である．図 3.1 では横軸の 0 は 1995 年，図 3.2 の横軸の 0 は 1994 年を表している．図 3.1 は 588 種の原核生物の完全ゲノムの総塩基対数を KEGG [24] から取得し，GenBank のゲノムデータベースの総塩基対数を比較している．このグラフでは指数近似の直線を書き，データ量増加の伸び率を示している．また図 3.2 は 11 種のほ乳類の完全ゲノムと，GenBank のゲノムデータベースを比較している．このグラフではいづれの生物カテゴリの完全ゲノムはゲノムデータベースの伸び率にくらべ，下回っていることがわかる．つまりゲノムデータベース全体をミラーリングする場合，増加するデータを更新し続けなければならないが，伸び率が完全ゲノムのデータ量より上回っており，将来的にゲノムデータベースの差分を取得するよりも完全ゲノムのデータのみを転送した方が転送量が少なくなる可能性が高い．

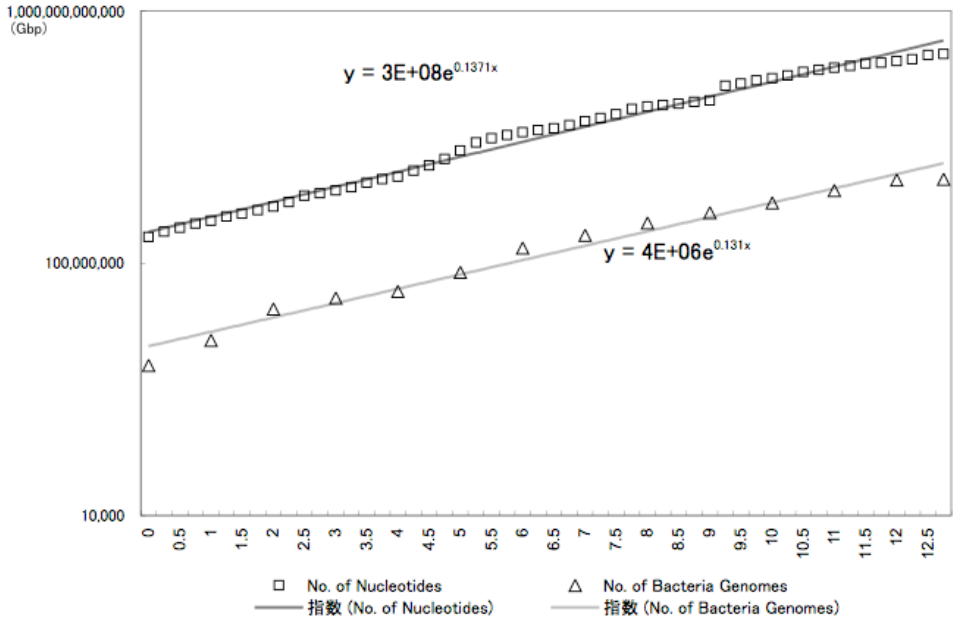

図 3.1: 原核生物の完全ゲノムとゲノムデータベースの増加指数

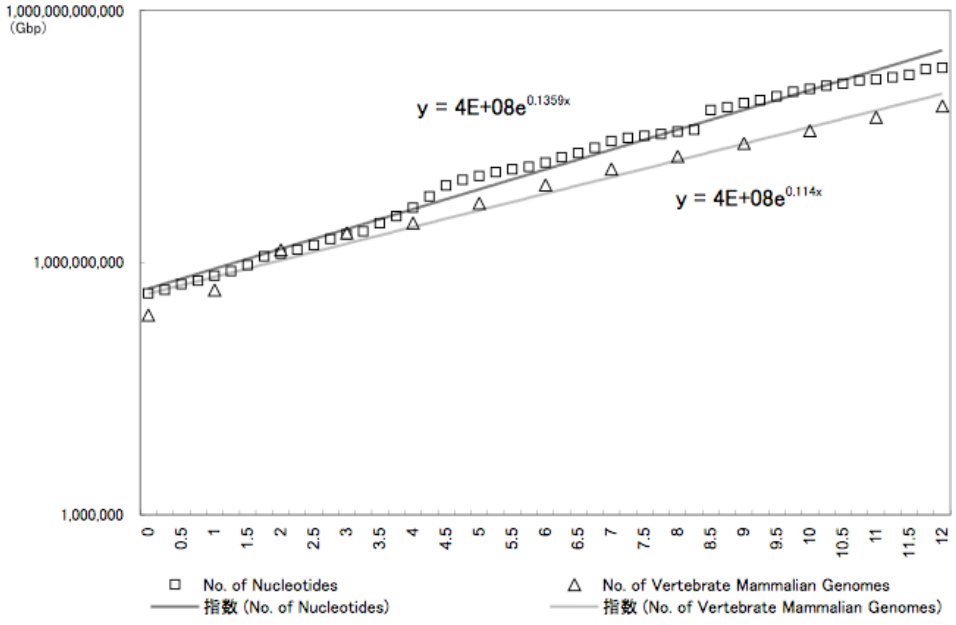

図 3.2: ほ乳類の完全ゲノムとゲノムデータベースの増加指数

3.3 データステージングの提案

3.2.2 節では，Web サービスの問題点について述べ，3.2.3 節では，ミラーリングの問題点について述べた．しかしながら Web サービスによる解析では，データ量が膨大になった場合，データベースサイトに解析を委託するためデータ量の増加の影響を受けないなどの有用な点もある．またミラーリングに関しては複数台の計算ノードを利用した効率的な解析環境が実現できるため，有用である．つまり両手法とも極端であり，一方はデータベースは全く転送しない手法，もう一方はデータベースを全て取得する手法である．そこで本研究では必要なデータのみを転送する手法，データステージングによる解析手法を提案し，ミラーリングにおけるデータ量問題と，Web サービスにおける効率的な処理問題の折衷点を探る．

データステージングとは，ファイルを必要なときに必要とする場所に転送する分散コンピューティングの技術であり，データベースサイトから直接データを取得することから，常に最新のデータを利用するための管理が不必要となる．またユーザが必要な個別のデータを解析時に指定することによって，不必要なデータを取得することなく解析を行う事が可能となる．分散コンピューティングでは，実行環境とデータソースが異なる場合が多い事からデータステージングを利用する研究が進められている [33]．データステージングを利用することによって，ツールの実行単位であるジョブの実行時に入力データを転送，ジョブの終了時に出力データを任意の計算ノードに転送する．ジョブ実行の前に行う転送をステージイン，そしてジョブ実行の後に行う転送をステージアウトと呼ぶ．ステージインは，他の計算ノードからジョブ実行する計算ノードに必要なファイルを転送し，ステージアウトはジョブ実行した計算ノードから他の計算ノードへファイル (例えば，結果ファイルなど) を転送する．またステージイン，ステージアウトしたファイルはジョブ実行した計算ノードからジョブ実行の終了と同時に削除される．データステージングでは，いつ転送するのかが重要な点である．データステージングを利用することによって研究者がジョブを実行する計算ノードを意識せずに済み，予めデータを配置する必要がなくなる．データステージングは，ジョブ実行を管理するスケジューラが行うことが多く，スケジューラがジョブ実行の要求を行うと同時にデータステージングを他の計算ノードに対して行うことができる．その

ためジョブの実行とデータステージングを並列に行うことにより，効率的に解析を行うことが可能となる．必要なデータのみ，個別データのみを転送することでミラーリングよりも転送量を抑えることができると考えられ，また複数の計算ノードが利用可能なことから効率的な解析も実現可能と考えられる．

そこで本研究は分散コンピューティング技術による複数の計算ノードを統合し，バイオインフォマティクスツール実行支援システムをデータステージングを用い構築した．3.4 節で詳細を述べる．

3.4 データステージングを用いた実行支援システムの設計と実装

3.2.1 節で述べたような多数の個別データに対する解析を行うためのバイオインフォマティクスツール実行支援システムを，分散コンピューティング技術であるグリッドコンピューティング [9] を用いてプロトタイプの設計を行った．グリッドコンピューティングは広域分散のためのアーキテクチャであり，グリッドコンピューティングの実装である Globus Toolkit [10] がデファクトスタンダードになりつつある．Globus Toolkit には GRAM [34] といったツールのリモート実行を行うミドルウェアや，GridFTP [35] という高効率転送を可能にした FTP をベースにしたプロトコルの実装も含まれている．また暗号化した通信経路を確保したリモートコンソールや簡易ファイル転送などに利用され広く普及している SSH [36] の実装である OpenSSH [37] のサブセット，GSI–Enabled OpenSSH (GSI–SSH) [38] も含まれている．GSI–SSH は OpenSSH の認証を GSI で行うことができ，OpenSSH と同様にリモートコンソールの提供や，リモートの計算機からのコマンド実行機能を提供している．

バイオインフォマティクスツール実行支援システムのプロトタイプは，動作を軽量にすることを考慮し GSI–SSH をジョブ実行のミドルウェア，データステージングに利用するプロトコルを GridFTP とした．設計したアーキテクチャを図 3.3 に示す．プロトタイプシステムは 3 種類のノードから構成される．スケジューラノードとデータベースノードと計算ノードからなる．スケジューラノードはユーザからの要求を解釈し，ジョブのスケジューリングやデータステージングの機能を持つ．スケジューラノー

ドはユーザからのジョブ要求を解釈し，計算ノードにジョブの割り振りを行う．データベースノードは公共データベースを想定しているがプロトタイプシステムではデータベースファイルを保持する役割である．計算ノードはジョブの実行要求を受け付けるために GSI–SSH を，データステージングの要求を受け付けるために GridFTP を設定する．

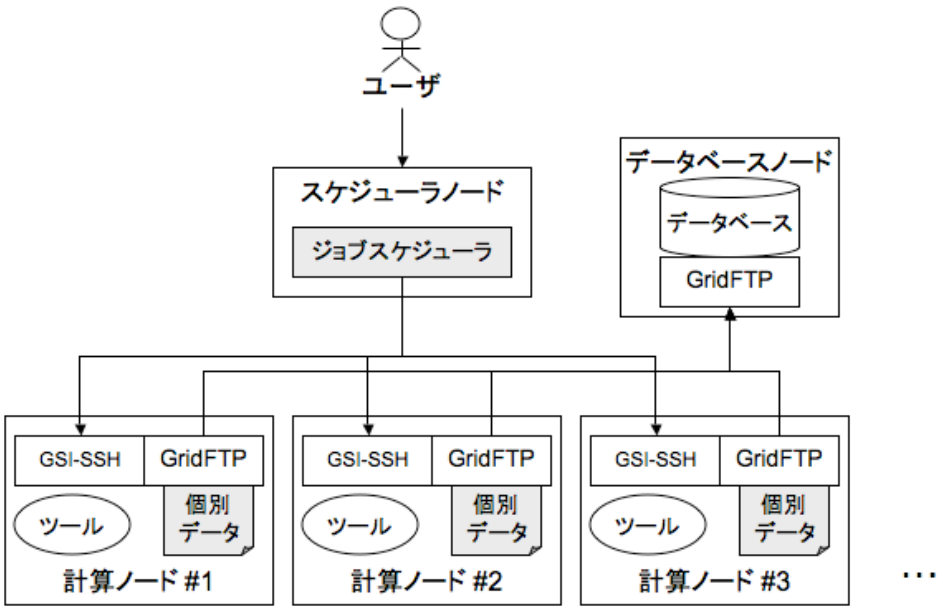

図 3.3: システムの概要

図 3.4 はシステムアーキテクチャの詳細と挙動を示す．図中の網掛けの箇所が実装を行った箇所である．以下，箇条書きにてシステムの動作について説明する．また以下の番号は図 3.4 中の番号に該当する．またジョブとはコマンドラインの実行単位であり，複数のジョブの組合せをジョブセットと呼ぶ．

1. ユーザは解析対象となる複数の個別データ名と解析ツールのコマンドラインを，リクエストジョブセット記述ファイルに記述する．リクエストジョブセット記述ファイル (図 3.5 参照) をジョブスケジューラに送付しジョブセットの実行を要求する．

2. ジョブスケジューラはジョブセットを受け付けたのち，使用する計算ノードを解

析する個別データ毎にジョブを割当てていく．

3. 計算ノードに解析対象となる個別データファイルをステージインする．また計算ノードに個別データファイルが存在する場合，この処理は行わない．

4. ステージインが終わった後，ジョブスケジューラは計算ノードにジョブの実行を要求する．

5. ジョブが終了したのち，ジョブスケジューラは計算ノードから解析結果ファイルをステージアウトし，スケジューラノードに保持する．また複数の計算ノードが利用できる場合，並列して複数の計算ノードに対して要求 (ステージイン，ステージアウト，ジョブ実行) することが可能．

6. すべてのジョブが終了したのち，スケジューラノードにある解析結果ファイル全てをユーザに返却する．

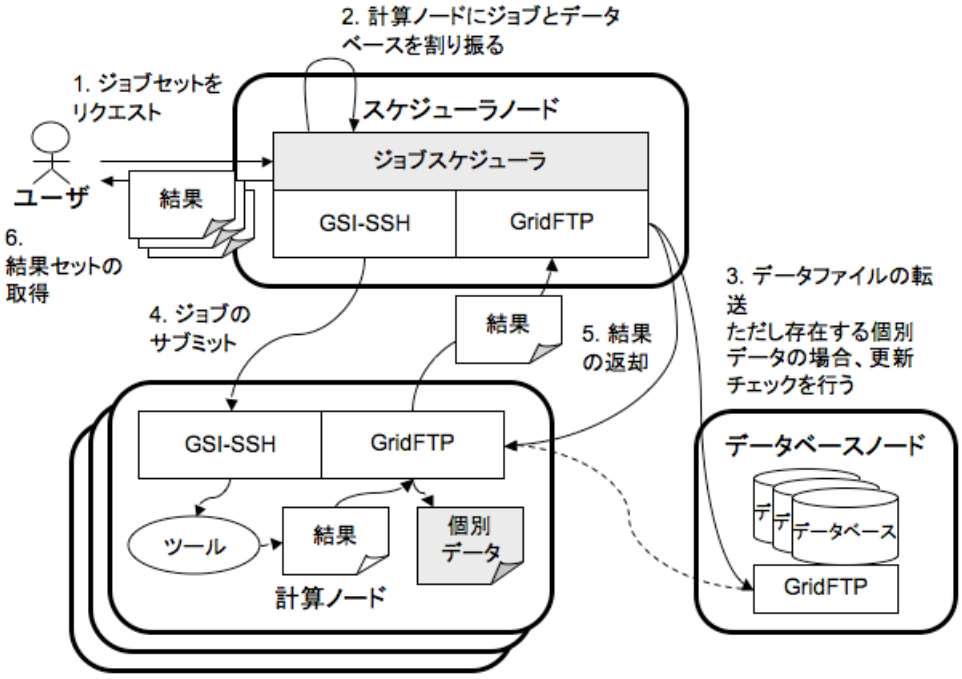

図 3.4: システムアーキテクチャの詳細

次にリクエストジョブセット記述ファイルのサンプルを図 3.5 に示す．図 3.5 は，細

菌ゲノム配列の個別データ (*Helicobacter pylori 26695*) に対して BLAST を実行するために記述したサンプルである．リクエストジョブセットファイルは複数のジョブを記述でき，ジョブスケジューラで解釈されジョブ単位で実行される．そのため実行結果は記述したジョブの数だけ発生する．ユーザはリクエストジョブセットファイルに，ジョブの詳細について記述しなければならない．しかしデータステージングを指定するタグ <database> は，データベースを構成する複数のファイルを個別データの指定のみでデータステージングを行うことを指示する．これによるユーザは興味のある個別データ名を記述するだけでジョブが実行できる．

```
<jobset>
<job>
<executable>/opt/blast-2.2.16/bin/blastall</executable>
 <directory>{$GLOBUS_USER_HOME}</directory>
 <database>Helicobacter_pylori_26695</database>
 <inputfile>/opt/blastdb/test1.aa.txt</inputfile>
 <argument>-p</argument>
 <argument>blastp</argument>
</executable>
</job>
<job>…</job>
…
</jobset>
```

図 3.5: リクエストジョブセット記述ファイルのサンプル

3.4.1 ノード間の処理手順

データステージングを利用した解析におけるノード間の処理手順を図 3.6 に示す．また比較する上で Web サービス，ローカル (ミラーリング) についても同様を図 3.7 に

示す．図 3.6 では，ノード間で発生するデータ送受信の処理手順を時系列で示しており，上から順番に解析の流れとして表現している．ジョブの要求と個別データの転送はジョブスケジューラが要求を行う．計算ノードでは，ジョブ実行の要求があった場合，個別データの転送完了を待ちジョブの実行を行う．またジョブスケジューラは計算ノードがジョブを実行中に並列して次のジョブに必要な個別データの転送を指示する．結果の転送についてはジョブが終了すると同時に計算ノードからジョブスケジューラに転送される．変数はそれぞれ，t_{req_i} を転送やジョブ実行の要求にかかる時間，t_{resp_i} を要求に対する応答に掛かる時間とし，t_{exe_i} をジョブの実行時間，t_{out_i} を結果の転送時間，t_{in_i} を個別データの転送時間とする．

ジョブ 1 の実行時間 : $t_{in_1}, t_{exe_1}, t_{out_1}$

ジョブ 2 の実行時間 : $t_{in_2}, t_{exe_2}, t_{out_2}$

ジョブ 3 の実行時間 : $t_{in_3}, t_{exe_3}, t_{out_3}$

.

.

において，t_{in_2} と t_{exe_1}，t_{in_3} と t_{exe_2} と t_{out_1}，一般には t_{in_i}，$t_{exe_{(i-1)}}$，$t_{out_{(i-2)}}$ の処理を重ねる事ができる．従って t_{req_i} や t_{resp_i} が 0 であると仮定し，ジョブ数を k とした場合，全てのジョブが終了する合計時間は式 (3.1) となる．

$$
\begin{aligned}
T_{stagingtotal} & \\
&= t_{in_1} + max(t_{in_2}, t_{exe_1}) \\
&+ \sum_{i=3}^{k} max(t_{in_i}, t_{exe_{(i-1)}}, t_{out_{(i-2)}}) \\
&+ max(t_{exe_k}, t_{out_{(k-1)}}) + t_{out_k} \qquad (3.1)
\end{aligned}
$$

またノード数 n が 2 以上では，$t_{exe_{(i-1)}}$ と t_{exe_i} も重ねることができるので，さらに複雑になるが，そのときは近似的に t_{exe} の値がノード数に応じて短縮するような形で扱うことができる．

図 3.7 でも同様にノード間で発生するデータ送受信の処理手順を時系列で示している．Web サービスでは Web サービスクライアントと Web サービス間，ミラーリング

では既にデータファイルを計算ノードに配置済みであるのでローカルと記述している．ローカルで解析を行う場合の処理手順は，ジョブスケジューラと計算ノード間について示している．Web サービスでは，ジョブの実行 (リクエスト) から結果の転送 (レスポンス) まで直列に動作することとなる．つまり Web サービスで全てのジョブを終了する合計時間は式 (3.2) となる．

$$T_{webservicetotal} = \sum_{i=1}^{k} (t_{exe_i} + t_{out_i}) \tag{3.2}$$

またローカル (ミラーリング) では，既に利用するデータベースファイルはあるものとして解析を行う．よって全てのジョブを終了する合計時間は式 (3.3) となる．

$$T_{localtotal} = \sum_{i=1}^{k} t_{exe_i} \tag{3.3}$$

3.5 評価実験と課題

3.4 節で述べたプロトタイプシステムを実装し，実際に環境を構築した．図 3.8 に環境を示す．計算ノードには 5 台の計算機を用意し，スケジューラノード，データベースノードと計算機ノードを結ぶネットワークは LAN で構築した．LAN のスループットはいづれも 100Mbps となる．計算ノードのうち，計算ノード 1 から計算ノード 5 まで同等のスペックである．計算ノードの仕様について詳細を表 3.1 に示す．

実際に行った解析は 3.2 節で述べたタンパク質–タンパク質相互作用の研究を例にとり，527 の原核生物種ゲノム配列と大腸菌ゲノム配列の相同性検索を BLAST を用いて行った．原核生物種ゲノム配列は個別データとして BLAST のデータベース形式ファイルとしてデータベースノードに配置し，実行時に計算ノードへ転送した．原核生物種ゲノムのデータファイルの容量の総量は 917MByte である．大腸菌ゲノム配列には 897 配列のマルチ FAST フォーマットデータを利用した．また計算ノードを増加させたときの解析時間短縮の効果を評価基準とするため，1 ノードで行った場合から，5 ノード全てを利用した場合の解析時間を計測した．

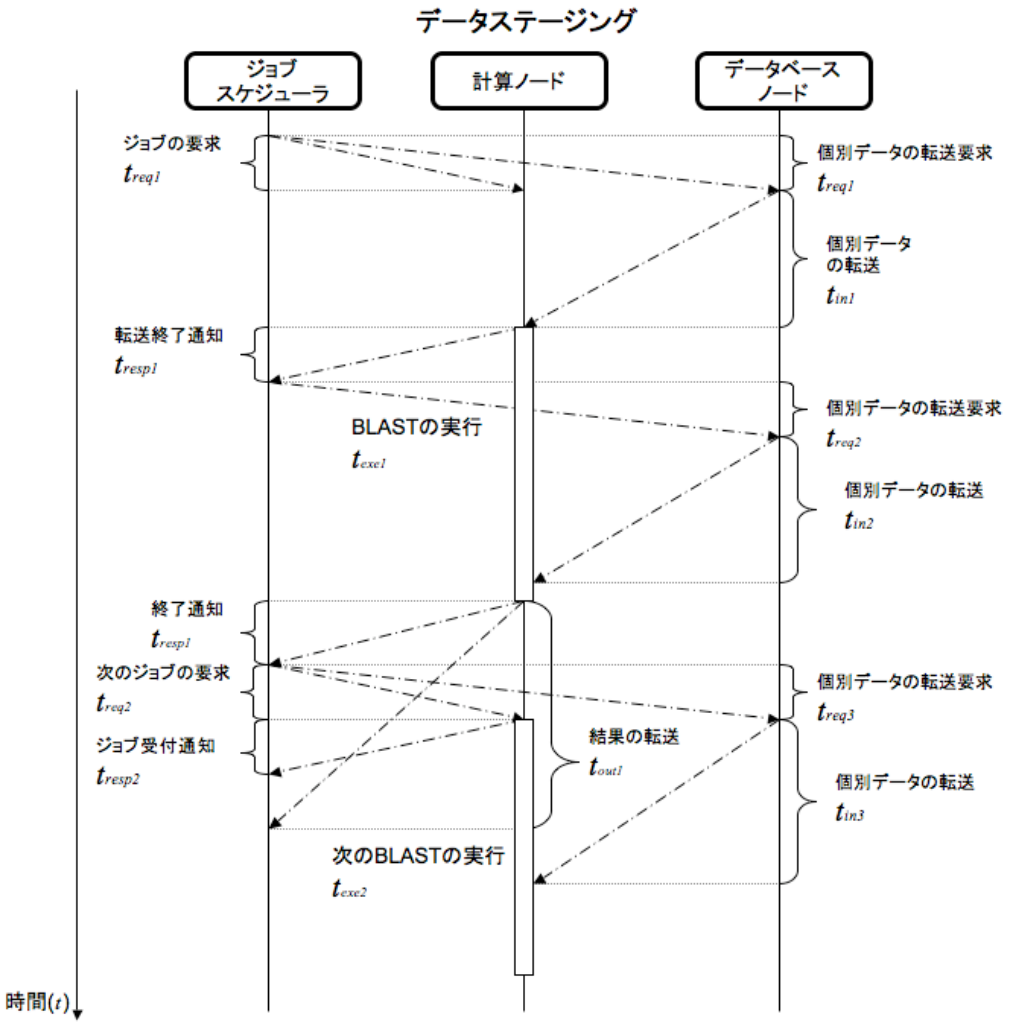

図 3.6: データステージングのノード間の処理手順

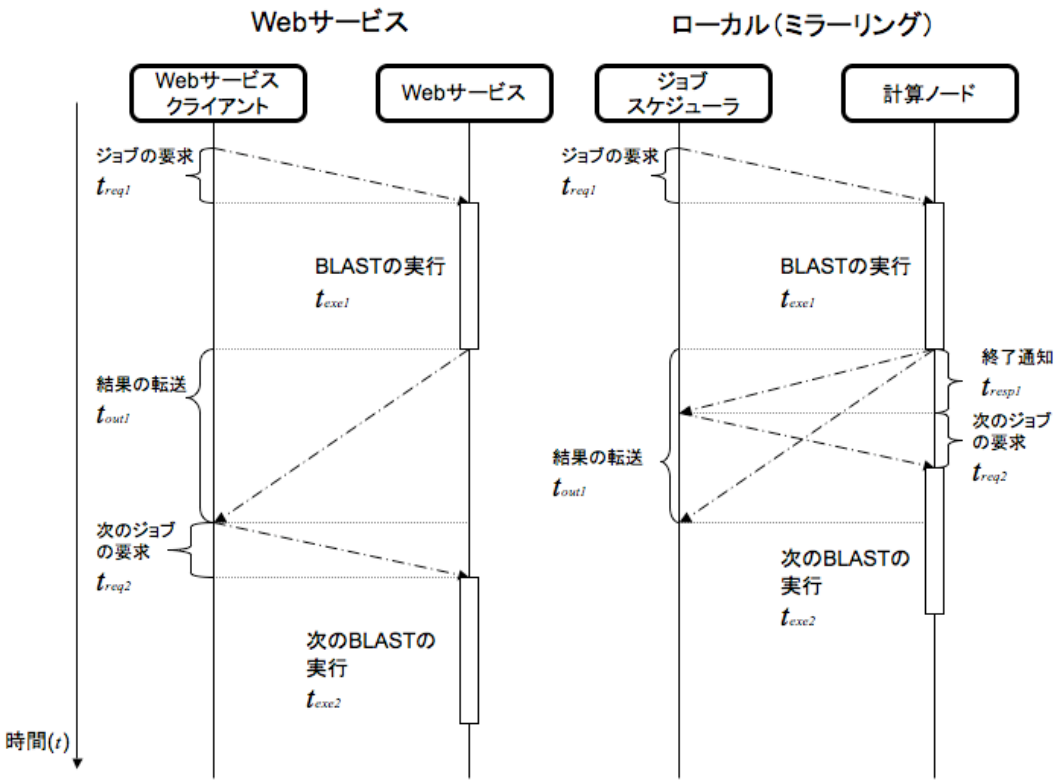

図 3.7: Web サービス，ミラーリングにおけるノード間の処理手順

またデータステージングだけでなくミラーリングと Web サービスについても同様の環境を構築し，同様の解析を行った．その結果について図 3.9 にまとめた．縦軸が解析にかかった時間，横軸が計算ノードの数を示している．また具体的な数字は表 3.2 にまとめた．図 3.9 にミラーリングとは記述していない．それは評価実験を行う際，ミラーリングの転送を行っておらず，各計算ノードに対し既にミラーリングが完了していることを想定している．そのため計算ノードのローカルにデータベースファイルを配置した状態で実験を行った．よって表記はローカルとしている．

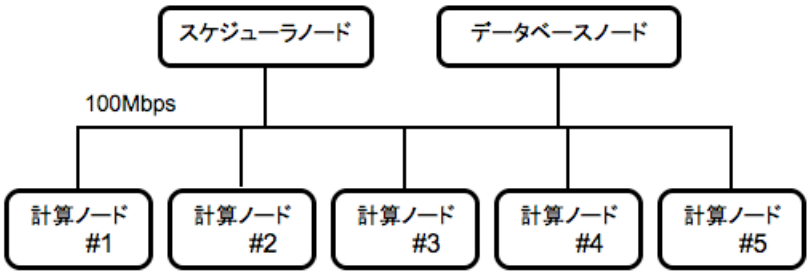

図 3.8: 試験環境の概要

表 3.1: 計算ノードのスペック一覧

	計算ノード#1– #5	クライアント
CPU	Intel Xeon	Intel Pentium 4
周波数 (GHz)	2.00	2.53
主記憶容量 (MByte)	4096	1024

表 3.2: 比較実験の解析時間

計算ノード数	1	2	3	4	5
ステージング	11,751.7s	6,011.2s	3,933.4s	2,921.8s	2,328.9s
ローカル	11,580.2s	5,714.4s	3,830.4s	2,869.4s	2300.0s
Web サービス	11,870.8s	7,346.7s	5,825.3s	4,967.0s	4,084.3s

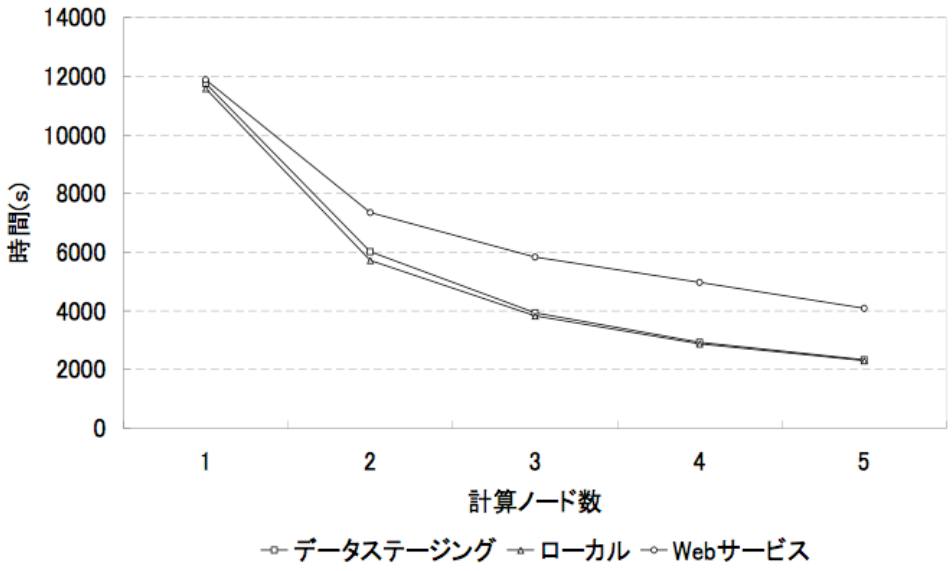

図 3.9: 従来手法との比較実験結果

3.5.1 解析時間に関する考察

Web サービスについては，1 つの Web サービスインタフェースに対し計算ノードを増加させ，計測している．Web サービスは他の手法に比べると計算ノード増加による時間短縮は少ない．これは解析を行う際，Web サービスクライアントが Web サービスを呼び出しジョブの実行を行うが，Web サービスからの結果を受け取るまで，Web サービスクライアントは次のジョブの実行を行うことができない (図 3.7 参照)．つまり Web サービスは実行要求から結果の取得までサーバ側とクライアント側で同期して動くこととなる．一方，データステージングやローカルに配置しての解析では，ジョブスケジューラがジョブ実行，結果転送を制御し，1 つのジョブが終わると同時に，次のジョブの実行要求を行うことができる．それはジョブスケジューラ (クライアント) と計算ノード (サーバ) 間は，相互に通信を行うことができるが，一方の Web サービスでは Web サービス側から Web サービスクライアントへの能動的な通信ができない．そのためデータステージングやローカルではジョブの実行と同時に結果の転送を行うため，効率的となり，図 3.9 での結果では Web サービスのみ差が大きく生じている．

またローカルにデータベースファイルを配置した実験と，データステージングの実

験での比較では，ローカルに配置した解析の方が若干速い．データステージングではもちろん個別データファイルの転送時間も含まれているため，計算ノードを増やした場合も変わらずローカルの方が速い．しかし計算ノード数が増加するごとにデータステージングによるオーバヘッドも短縮していることもわかる．これもジョブ実行とデータベースファイル転送が同時に行われることによる影響だと考えられる．図 3.6 に示すようにデータステージングでは，計算ノードでジョブを実行しているときに，ジョブスケジューラでは次のジョブに必要な個別データを転送している．つまりデータステージングにおけるステージインとジョブ実行を並列して行っている．

図 3.6 と図 3.7 に示す変数を利用し，データステージング，ローカル，Web サービスについて理想的な総処理時間を式 (3.4) に示す．n はノード数であり，また計算ノードへのジョブの割り当てにかかる時間を $Overhead(T)$ としている．

$$T_{dist} = \frac{(T_{total} - Overhead(T_{total}))}{n} + Overhead(T_{total}) \tag{3.4}$$

図 3.12 の外挿の曲線が交わらないことを述べるために，実験で使用した 527 種の原核生物種のそれぞれのデータに対する解析実行時間 (t_{exe})，結果転送時間 (t_{out})，個別データ転送時間 (t_{in}) を図 3.10 と図 3.11 に示す．図 3.11 では，図 3.10 に示した t_{out} と t_{in} のみを抽出したグラフである．それぞれ縦軸は時間であり，横軸は個別データのデータ量を示している．解析時間である t_{exe} は個別データのデータ量に比例している．また結果転送時間である t_{out} は，個別データのデータ量にほぼ無関係に分布している．それは実験で利用したプログラムが BLAST であり，BLAST では相同性が確認された DNA 配列をスコア順に列挙し，付加情報も併せて結果としている．そのため個別データのデータ量とは無関係に解析結果の量が変化するため，この様な結果となった．また個別データの転送時間は，データ量に比例して線形に変化している．t_{exe}，t_{out}，t_{in} のそれぞれの平均と，個別データのデータ量について表 3.3 にまとめた．

式 (3.1) と式 (3.2) にそれぞれ表 3.3 の値を適用し，データステージングも Web サービスも n ノード数の時の総処理時間を式 (3.4) とすると，外挿したグラフ図 3.12 を示すことができる．データステージングと Web サービスはそれぞれ反比例のグラフとなるので，データステージングが 1 ノードから 5 ノードで低い値を示せば，ノードを増

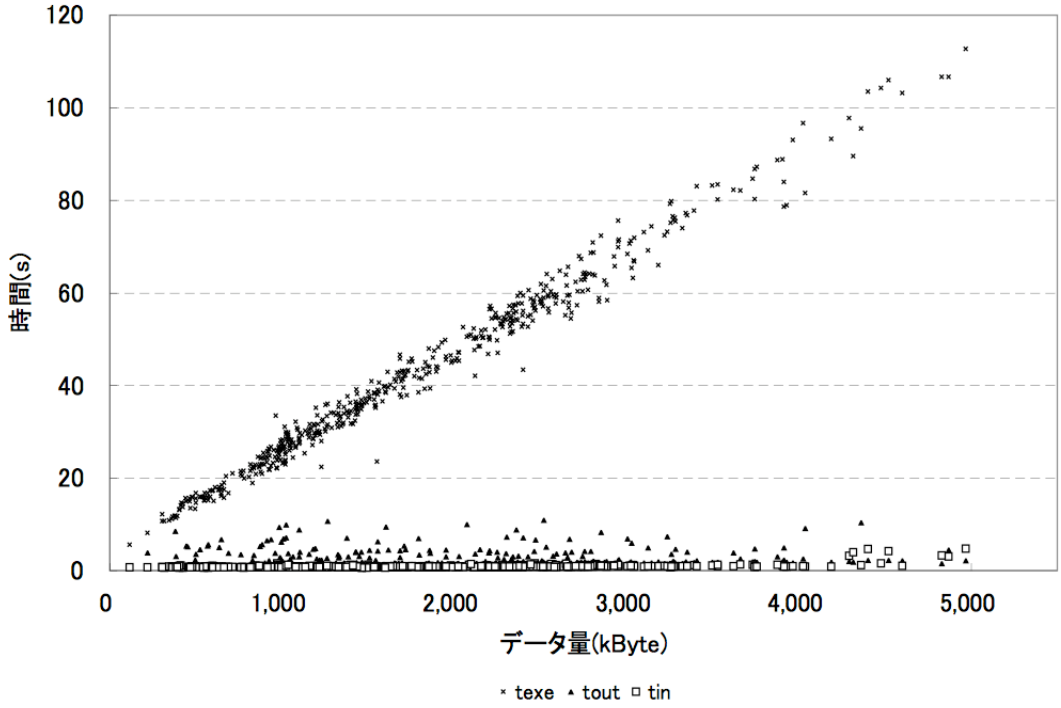

図 3.10: t_{exe}，t_{out}，t_{in} の実測値

表 3.3: t_{exe}，t_{out}，t_{in} の実測値とデータ量の平均

データ量 (Byte)	1,823,296
解析時間 (t_{exe}) の平均 (秒)	42.7
結果転送時間 (t_{out}) の平均 (秒)	2.0
個別データ転送時間 (t_{in}) の平均 (秒)	0.9

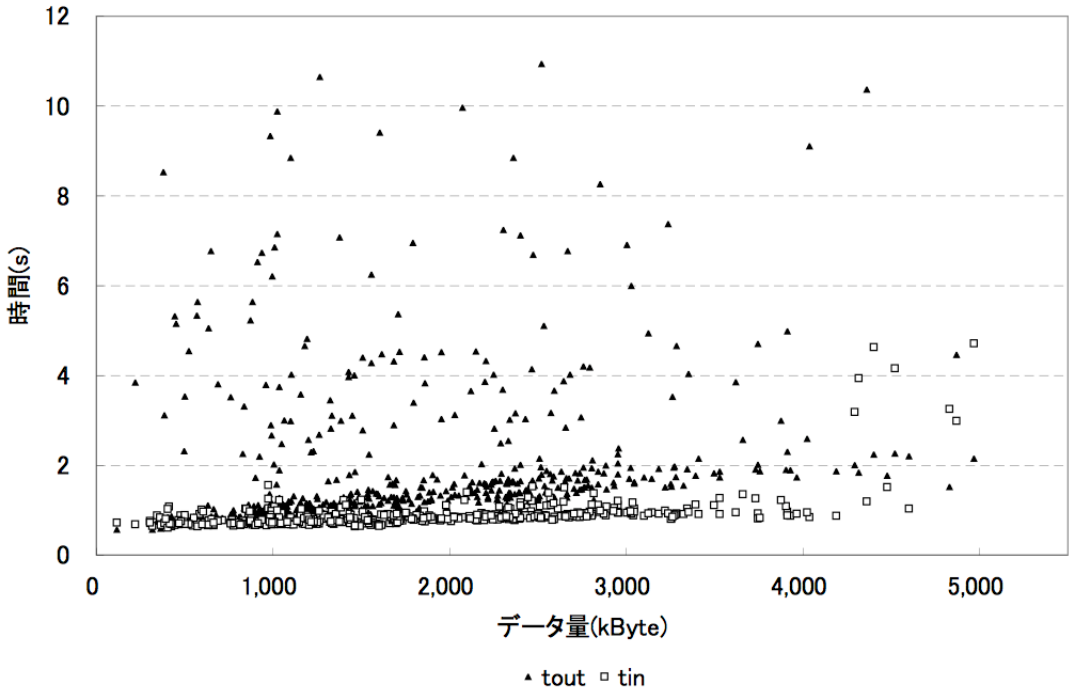

図 3.11: t_{out}, t_{in} の実測値

加させても交わることはない．ただし，$max(t_{exe}, t_{out}, t_{in}) > t_{exe} + t_{out}$ となる場合は，Web サービスとデータステージングが逆転することとなる．つまりデータ転送時間 t_{in} が極端に大きくなる $t_{in} > t_{exe} + t_{out}$ の場合，Web サービスの方が有利となるが，個別データ転送時間 (t_{in}) が少ない場合は，データステージングが有利である．また実験で得られた t_{exe}，t_{out}，t_{in} は常に $t_{in} < t_{exe} + t_{out}$ を示しており，Web サービスと比較する場合，総処理時間においてデータステージングが有用であると言える．

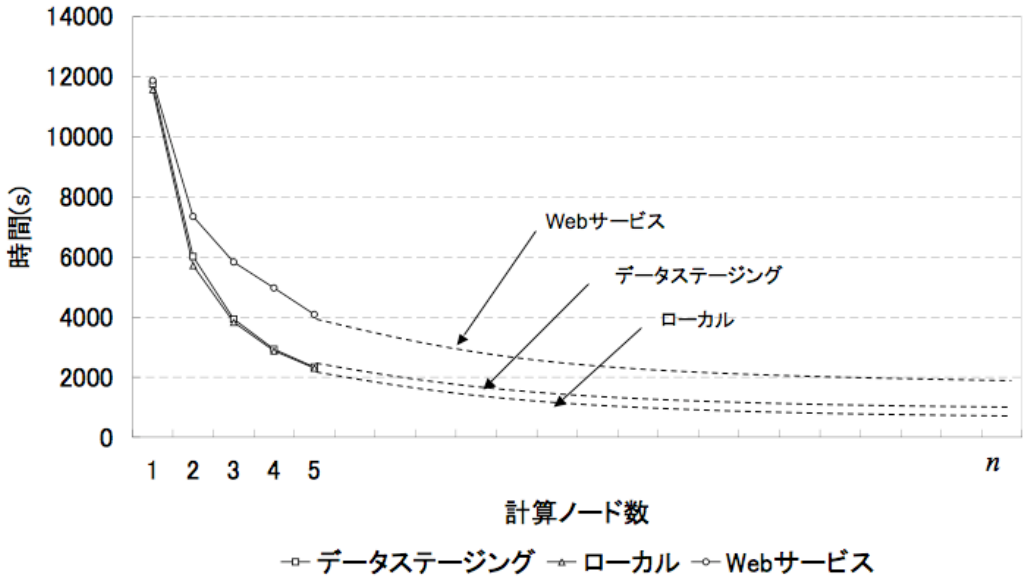

図 3.12: 解析時間の外挿グラフ

3.5.2 データ量に関する考察

3.2.3 節ではゲノムデータベースと完全ゲノムの増加について述べたが，データ量の変化に関して考察を述べる．データステージングでは，個別データ量に対して計算ノードが増加しても，データの転送量の総和に変化はない．なぜなら利用する個別データをジョブスケジューラが計算ノードに割り当てて転送を行うためである．しかし計算ノードすべてにデータベースのミラーリングを保持する場合，データベースの更新データ量がそれぞれの計算ノードに転送され，転送量の総和はノードの増加に比例し

て増加することとなる．図 3.13 はデータの転送量を縦軸に，ノード数を横軸に取ったグラフである．データベースの更新量を $S_{dailyupdate}$ とし，n をノード数とした場合，ミラーリングの転送量の総和は式 (3.5) となり，また $n > n_0$ の範囲でデータステージングの方がデータ転送の総量が小さくて済む．

$$S_{mirroring} = S_{dailyupdate} \cdot n \tag{3.5}$$

ゲノムデータベースの更新量については GenBank の Daily Update [39] を参照した．ゲノムデータベースの更新は毎日行われており，DNA 配列の更新量は 1 日平均約 52.1MByte (2008 年 2 月 13 日から 2008 年 3 月 24 日までの期間) となった．またデータステージングで必要なデータ量として，完全ゲノムのデータ量を KEGG Organisms [40] を参照し，原核生物 588 種について登録された年および完全ゲノムの塩基対数を取得した．1995 年から 2008 年 2 月までに約 2,157.0Mbp が登録されており，1 日平均約 449kbp の塩基数が登録されていることとなる．単純に 1kbp = 1kByte とすると，1 日当り 449kByte の更新となる．これらの数値からミラーリングでは $52.1MByte \cdot n$ の転送量，データステージングでは $2,157.0MByte$ の転送量となり，$n > 41$ の時にデータステージングよりミラーリングの転送量の合計が上回ることとなる．

3.5.3 今後の課題

今後の課題としては，ジョブスケジューラの最適化が挙げられる．今回の実験で使用したジョブスケジューラは計算ノードの行列に対して順に割り当てて行くラウンド・ロビン方式であり，最適化がなされていない．スケジューリング方式としては，ジョブ割当て主体に負荷分散を最適化するマスタワーカスケジューリングと，データ配置主体に最適化するデータアフィニティスケジューリングが挙げられる．マスタワーカスケジューリングでは，ジョブの実行の終わった計算ノードがその都度スケジューラに対し，ジョブの割当てを要求し，それに応じてスケジューラは動的にジョブを割当てていく．つまり割当て要求の送受信のオーバヘッドが発生する．データアフィニティスケジューリングでは，入力データサイズが全ての計算ノードで均等になるようにジョブを割当てる方式であり，計算ノードは割当て要求を行わない．つまりデータアフィ

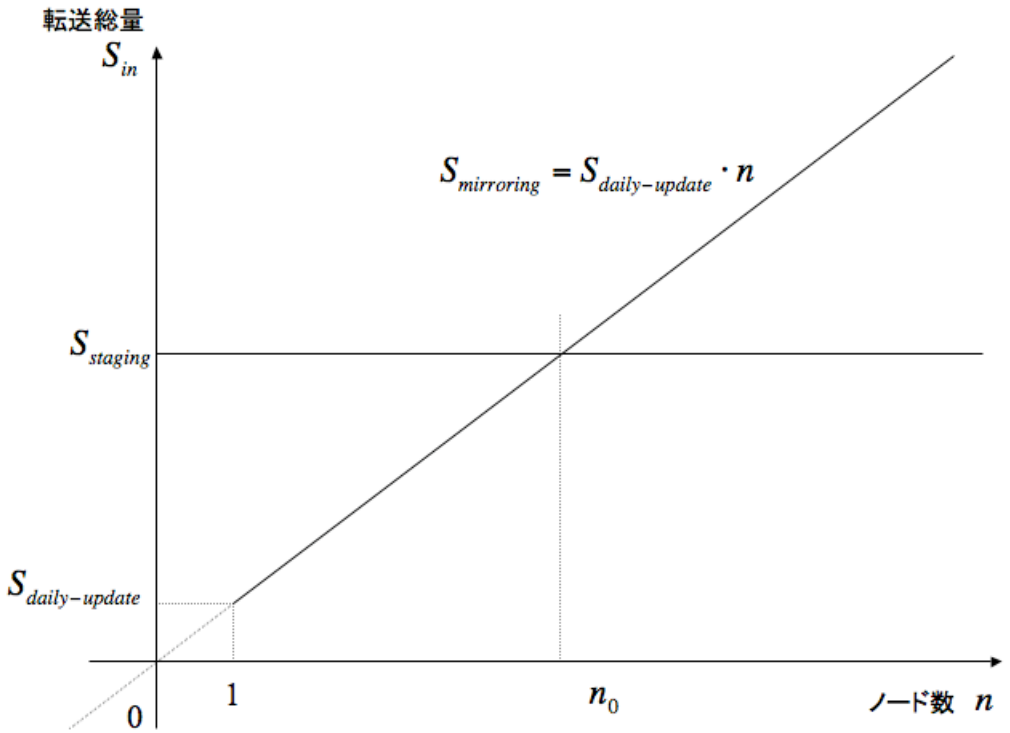

図 3.13: データ転送量の外挿グラフ

ニティスケジューリングでは入力データサイズとジョブの処理時間が比例すると仮定しており，その仮定が成り立つのであれば，ジョブ割当てのオーバヘッドがない分，全体の処理時間を短縮できるものと思われる．しかし前述の仮定が一部で成り立たずに，入力データサイズと処理時間の相関がない場合は，計算ノードの負荷バランスが崩れることで，マスタワーカスケジューリング以上に全体の処理時間がかかる恐れがある．このようにデータ配置主体でのスケジューリングか，ジョブ割当て主体のスケジューリングを選択するかという研究課題が残されている．

3.6 おわりに

本章ではデータの取得方法を検討し，データステージングを用いたバイオインフォマティクスツールの実行支援するシステムを実装，構築した．これにより研究者は制限無くカスタマイズした解析ができる環境，最新の個別データが利用できる環境，そして高効率な分散環境を同時に実現できる．また本研究ではこのシステムの評価として，897 の大腸菌遺伝子配列と 527 の原核生物ゲノム配列との相同性検索を BLAST で行い，計算ノードを増加させた場合の比較実験を行った．計算ノードが増加した場合，データステージングは Web サービスよりも解析時間を短縮することができ，またデータステージングのオーバヘッドは，計算ノードが増加することによって減少することが確認できた．本章で述べたシステムでは課題が残されているが，データステージングを利用した分散環境のツール実行支援がバイオ研究に役立つものと考えられる．

第4章　結論

4.1　本論文のまとめ

本論文では分散した計算機を統合的に利用するグリッド環境下での，共同利用施設における機密性を確保する方法と，増加するバイオ研究データに対応するためのデータステージングを用いた解析環境に関する研究の成果をまとめた．

本研究の１つ目の成果として，ライフサイエンス研究共同研究基盤システムを構築し，共同利用施設の利用時における機密性を確保する方法について提案を行った．ライフサイエンス研究では共同利用施設での大型計算機の利用や，生成した生物種データの共有といった観点から，共同研究に期待されている一方で，創薬や医療情報など機密に関わるデータを扱う事が多いことから，通常とは異なる機密性の確保を要求される．ユーザはデータの種類や具体的な疾患名などをファイルパス名に含め管理しているため，ファイルパス名が他のユーザに知られることで，研究対象としている疾患や化合物，遺伝子などが特定される恐れがある．一方，機密性を確保しなければいけないデータも更新頻度が高いという特徴があるため，機密性を確保する上でファイル更新時のオーバヘッドについても考慮する必要がある．ファイルパス名を隠蔽化する既存の手法としては，ファイルパス名の論理名と物理名を写像をメタデータに保持させることによってファイルパス名を隠蔽化する手法が挙げられる．しかしながらメタデータを利用する方法では，ファイル更新時にメタデータへの更新も行われるためオーバヘッドが増加する恐れがある．また既存手法としてはファイルパス名を隠蔽化する方法としてユーザごとに個別にマウントポイントを与える方法もある．しかし個別にマウントポイントを与える手法においては複数のストレージノードを利用する場合，ストレージノード１台１台に対し認証を行わなければならず利便性が損なう恐れがある．そこで共同利用施設の他のユーザからファイルパス名を隠蔽化するために，GSI–SFS

を用いユーザごとに個別のマウントポイントを与える手法を提案した．GSI–SFS を利用することによって GSI 認証によるシングルサインオンが実現され，複数のストレージノード，複数の計算ノードに対しユーザは 1 度のパスフレーズ入力にて認証を行う事が可能となり，また GSI–SFS によってユーザごとにマウントポイントが与えられ，他のユーザはファイルの存在やファイルパス名を知る事ができない．またメタデータを用いたファイルパス名を隠蔽化する方法として分散ファイルシステムである Gfarm と比較し，更新ファイル数が増加した場合において，更新時のオーバヘッドが抑えられることが実験結果として得られた．

2 つ目の成果として，増加するバイオ関連データに対応するためのデータ取得方法に関する研究について成果をまとめた．バイオ研究で利用するデータは年々増加の一途をたどり，また研究が複雑化，多様化している．タンパク質–タンパク質相互作用ネットワークの研究においては，推定した大腸菌タンパク質–タンパク質相互作用ネットワークが他の生物種との関連性を示すために，148 種以上の生物種完全ゲノムに対し相同性検索を行っている．この相同性検索は検索結果を生物種間で混在させないために生物種ごとに解析を実行している．この様な生物種完全ゲノムごとの解析を公共の Web サービスで行うには，サービス提供側が対象データの限定などを行っているため，ユーザの個別のパラメータ設定が不十分にしか行えない．また解析の実行と解析結果の送信が直列で行われるため，解析結果の送信を制御することができず，非効率である．一方，ミラーリングを用いてデータベースをユーザ独自の環境に複製を置くことで，個別のパラメータ設定可能な解析環境を得ることができる．しかしゲノムデータベースは年々増加の一途をたどり，1 日の更新量も増加し続けている．そこで本研究では，解析を実行するときに必要な個別データを取得するデータステージングを適用した解析環境を提案した．データステージングを用い，解析実行とデータ転送を並列に行う工夫をジョブスケジューラに組み込む事で，効率的にデータ転送と解析の実行を行うことを示した．また計算ノードを増加させることによって解析時間の短縮につながることを示し，解析時間の点で Web サービスよりも全体の処理時間が短縮することを示した．またデータ転送量の総和に関しては，ミラーリングと比較した．各計算ノードにデータベースの更新が発生するミラーリングでは，計算ノード数に比例してデータ転送量の総和が増加する．それに対しデータステージングでは計算ノードが

増加した場合においても，データ転送量の総和は変わらないことを示した．そのためデータ転送量の総和は，ミラーリングよりもデータステージングが少なくなることを示した．

4.2 今後の課題

機密性を確保するライフサイエンス共同研究基盤における課題としては，ファイルパス名の隠蔽化に関して課題が残されている．それはプロセス監視ツールなどによるファイルパス名の露呈が挙げられる．プロセス監視ツール，例えば ps コマンドなどによって他のユーザの実行ツール名と引数が一覧で得られることとなる．このときに解析ツールなどの引数としてファイルパス名を渡している場合，他のユーザに研究データのファイルパス名が露呈してしまい，機密性を保てない．こうしたプロセス監視ツールの実行権限を制御することで，抑止は可能である．

データステージングを適用したシステムにおける課題としては，ジョブスケジューラの最適化が挙げられる．ジョブ割当てを主体とした最適化手法としてはマスタワーカスケジューラが挙げられる．マスタワーカスケジューラでは，ジョブの終了した計算ノードがジョブスケジューラに対しジョブの割当てを要求し，その都度ジョブの割当てを行うため，割当て要求の送受信によるオーバヘッドが生じる．一方，データ配置を主体とした最適化，データアフィニティスケジューラでは，入力データサイズが均等になるように計算ノードにジョブを割当てる．入力データサイズと処理時間が比例する場合，ジョブ割当てによるオーバヘッドがないため，マスタワーカスケジューラよりも全体の処理時間は短縮できる．しかしながら入力データサイズと処理時間が比例しない場合，計算ノードの負荷バランスが崩れ，マスタワーカスケジューラ以上に全体の処理時間がかかる可能性がある．つまり全体の処理時間を短縮するためには解析の性質によってスケジューリング方式を選択する必要がある．今後，OGF などで行われているグリッド環境におけるスケジューラに関する研究の動向を参考にしながら，スケジューラの最適化を取り入れていくことが必要である．

バイオインフォマティクスツールの実行支援という一見地味な研究であるが，増加し多様化するバイオ関連データや，グリッド環境の普及による複数計算機での実行な

ど，ライフサイエンス研究への補佐的な役割は重要性を増していくと考えられる．バイオインフォマティクスツールを中心とした実行支援を充実させることにより，研究者らは研究に集中でき，ライフサイエンス研究の発展に貢献できると考えられる．

謝辞

本研究は，著者が 2005 年から現在まで大阪大学大学院情報科学研究科博士後期課程在学中に行ってきた，生物情報科学に関する研究成果をまとめたものです．

本研究の全過程の遂行ならびに本論文をまとめるにあたり，懇切なるご指導，ご鞭撻を賜わりました大阪大学大学院情報科学研究科バイオ情報工学専攻 松田秀雄 教授には，ここに厚く御礼申し上げます．貴重なお時間を割いていただき丁寧なるご教示を賜りました大阪大学大学院情報科学研究科バイオ情報工学専攻 柏原敏伸 教授，清水浩 教授，前田太郎 教授，四方哲也 教授に厚く御礼申し上げます．

また本論文をまとめるにあたり，研究の進め方に対する様々な御指導，御助言を頂きました大阪大学大学院情報科学研究科バイオ情報工学専攻 竹中要一 准教授には，厚く御礼申し上げます．

大阪大学大学院情報科学研究科バイオ情報工学専攻 伊達進 特任准教授には，本研究の全過程において数多くの御指導，御助言を頂きました．心より御礼申し上げます．

本論文をまとめるにあたり，様々な御支援，御指導を頂きました大阪大学大学院情報科学研究科バイオ情報工学専攻 瀬尾茂人 助教に，深く感謝いたします．

また中国科学院微生物研究所 信息网絡中心 馬俊才 主任，同 秦野彰二 様には，大阪大学と中国科学院をつなぐ共同研究基盤の研究におきまして，多くの議論と適切な提案をして頂きました．深く感謝致します．

本研究の様々な局面におきまして多くの議論と適切な提案，また様々な御支援をして頂きました彩都建設推進協議会 福岡良忠 様，三井情報株式会社 奥村利幸 様に深く感謝いたします．本研究の遂行にあたり，全過程において御激励，御支援を頂いた小林加代子 様に深く感謝いたします．

大阪大学大学院情報科学研究科バイオ情報工学専攻松田研究室の皆様には，本研究をまとめるに際して，様々な御支援，御指導いただきましたことを感謝いたします．

最後に本研究の遂行に際し，著者を御激励，御支援下さいました方々へ心より感謝致します.

参考文献

[1] D.A. Benson, I. Karsch–Mizrachi, D. J. Lipman, J. Ostell and D.L. Wheeler. “GenBank”, *Nucleic Acids Research*, Vol.35, Database Issue, pp.D21–D25, 2007.

[2] S. Altschul, W. Gish, W. Miller, E.W. Myers, and D. Lipman, “A Basic Local Alignment Search Tool”, *Molecular Biology* , Vol.215, pp.403–410, 1990.

[3] W.R. Pearson, “Rapid and Sensitive Sequence Comparison with FASTP and FASTA”, *Meth. Enzymol.*, Vol.183, pp.63–98, 1990.

[4] W.R. Pearson, “Searching Protein Sequence Libraries: Comparison of the Sensitivity and Selectivity of the Smith-Waterman and FASTA Algorithms.”, *Genomics.*, Vol.11, No.3, pp.635–650, 1991.

[5] DOCK, `http://dock.compbio.ucsf.edu/` .

[6] AutoDock, `http://www.scripps.edu/mb/olson/doc/autodock/` .

[7] GOLD, `http://www.ccdc.cam.ac.uk/products/life_sciences/gold/` .

[8] presto X, `http://www.jbic.or.jp/presto_x/index_px.html` .

[9] I. Foster, C. Kesselman, and S. Tuecke. “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, *International Journal of High Performance Computing Applications*, Vol.15, No.3 pp.200–222, 2001.

[10] I. Foster and C. Kesselman, “Globus: A Metacomputing Infrastructure Toolkit”, *International Journal of Supercomputer Applications* , Vol.11, No.2, pp.115–128, 1997, `http://www.globus.org/`

[11] Open Grid Forum, `http://www.ogf.org/` .

[12] 建部修見, 森田洋平, 松岡聡, 関口智嗣, 曽田哲之, “ペタバイトスケールデータインテンシブコンピューティングのための Grid Datafarm アーキテクチャ”, 情報処理学会論文誌 : ハイパフォーマンスコンピューティングシステム, Vol.43, No.SIG 6, pp.184–195, 2002.

[13] D. Mazieres, “Self–Certifying File System”, *Doctoral dissertation, Massachusetts Institute of Technology* , 2000.

[14] 武田伸悟, 伊達進, 下條真司, “GSI–SFS:グリッドのためのシングルサインオン機能を有するセキュアファイルシステム”, 情報処理学会論文誌 : コンピューティングシステム, Vol.45, No.SIG 6, pp.223–233, 2004.

[15] R. Buttler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer and V. Welch, “A National-Scale Authentication Infrastructure’, *Computer*, Vol.12, No.33, pp.60–66, 2000.

[16] A.O. Freier, P. Karlton and P.C. Kocher, “The SSL Protocol Version 3.02”, *IETF draft* , `http://wp.netscape.com/eng/ssl3/draft302.txt` .

[17] G. Cochrane, P. Aldebert, N. Althorpe, M. Andersson, W. Baker, A. Baldwin, K. Bates, S. Bhattacharyya, P. Browne, A. Van den–Broek et al. “EMBL Nucleotide Sequence Database: Developments in 2005”, *Nucleic Acids Research*, Vol.34, Database Issue, pp.D10–D15, 2006.

[18] K. Okubo, H. Sugawara, T. Gojobori, and Y. Tateno. “DDBJ in Preparation for Overview of Research Activities Behind Data Submissions” *Nucleic Acids Research*, Vol.34, Database Issue, pp.D6–D9, 2006.

[19] N. Deshpande, K.J. Addess, W.F. Bluhm, J.C. Merino–Ott, W. Townsend–Merino, Q. Zhang, C. Knezevich, L. Xie, L. Chen, Z. Feng, et al. “The RCSB Protein Data Bank: A Redesigned Query System and Relational Database Based on the mmCIF Schema”, *Nucleic Acids Research*, Vol.33, Database Issue, pp. D233–D237, 2005.

[20] C. Brooksbank, G. Cameron and J. Thornton. "The European Bioinformatics Institute's Data Resources: Towards Systems Biology", *Nucleic Acids Research*, Vol.33, Database Issue, pp.D46–D53, 2005.

[21] PubChem: Information on the Biological Activities of Small Molecules, `http://pubchem.ncbi.nlm.nih.gov/`.

[22] V.A. McKusick. "Mendelian Inheritance in Man", *Catalogs of Human Genes and Genetic Disorders, 12th edn. The Johns Hopkins University Press* , Baltimore, MD, 1998.

[23] R. Edgar, M. Domrachev and A.E. Lash, "Gene Expression Omnibus: NCBI Gene Expression and Hybridization Array Data Repository", *Nucleic Acids Research* , Vol.30, No.1, pp.207–210, 2002.

[24] M. Kanehisa and S. Goto. "KEGG: Kyoto Encyclopedia of Genes and Genomes", *Nucleic Acids Research*, Vol.28, No.1, pp.27–30, 2000.

[25] PubMed Central, `http://www.pubmedcentral.gov/` .

[26] MEDLINE, `http://www.nlm.nih.gov/pubs/factsheets/medline.html` .

[27] G.D. Schuler, J.A. Epstein, H. Ohkawa and J.A. Kans. "Entrez: Molecular Biology Database and Retrieval System", *Methods Enzymol* , Vol.266, pp.141–162, 1996.

[28] G. Butland, J.M. Peregrin-Alvarez, J. Li, W. Yang, X. Yang, V. Canadien, A. Starostine, D. Richards, B. Beattie, N. Krogan, M. Davey, J. Parkinson, J. Greenblatt and A. Emili, "Interaction Network Containing Conserved and Essential Protein Complexes in Escherichia Coli", *Nature* , Vol.433, pp.531–537, 2005.

[29] R.L. Tatusov, A.R. Mushegian, P. Bork, N.P. Brown, W.S. Hayes, M. Borodovsky, K.E. Rudd and E.V. Koonin, "Metabolism and Evolution

of Haemophilus Influenzae Deduced from a Whole-Genome Comparison with Escherichia Coli", *Current Biology* , Vol.6, No.3, pp.279–291, 1996.

[30] M.P. Papazoglou, "Service-Oriented Computing: Concepts, Characteristics and Directions", *Web Information Systems Engineering 2003 Proceedings of the Fourth International Conference on* , pp.3–12, 2003.

[31] A. Tridgell and P. Macherras, "The Rsync Algorithm", *Technical Report TR-CS-96-05* , Canberra 0200 ACT, 1996, `http://samba.anu.edu.au/rsync/`

[32] D. Gilbert, Y. Ugawa, M. Buchhorn, T.T. Wee, A. Mizushima, H. Kim, K. Chon, S. Weon, J. Ma, Y. Ichiyanagi, D.M. Liou, S. Keretho and S. Napis, "Bio-Mirror Project for Public Bio-Data Distribution", *Bioinformatics* , Vol.20, pp.2338–2340, 2004.

[33] 畑中正行, 中野恭成, 井口裕次, 大野利男, 佐賀一繁, 秋岡明香, 中田秀基, 松岡 聡, "OGSA アーキテクチャに基づく NAREGI スーパースケジューラの設計と実装", 情報処理学会論文誌 : ハイパフォーマンスコンピューティングシステム, Vol.2005, No.53, pp.33–38, 2005.

[34] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith and S. Tuecke, "A Resource Management Architecture for Metacomputing Systems", *Proc. of IPPS/SPDP'98 Workshop on Job Scheduling Strategies for Parallel Processing*, pp.62–82, 1998.

[35] W. Allcock, "GridFTP Protocol Specification", *Global Grid Forum 20, GridFTP Workshop Document*, 2003.

[36] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, 2006.

[37] OpenSSH, `http://www.openssh.com/` .

[38] GSI-Enabled OpenSSH, `http://grid.ncsa.uiuc.edu/ssh/` .

[39] GenBank Daily Update, ftp://ftp.hgc.jp/pub/mirror/ncbi/genbank/daily--nc/ .

[40] KEGG Organisms, http://www.genome.jp/kegg/catalog/org_list.html .