Beyond The Web — Homeless Data Server*

Kenji Arisawa

(Aichi University, Nagoya, Japan arisawa@aichi-u.ac.jp)

Abstract

A new type of data server is presented. The server is designed for grid computing. The distinctive feature of the server is that enables to execute programs from clients without allowing any byte to be written to the server. Therefore we need not allocate storage space for clients, which means the time and labor will be reduced greatly, and in addition, we can keep the server perfectly clean.

Keywords: grid computing, data server, Plan 9

1. 近頃巷に流行るもの

近頃ネット社会では「クラウド」とか「ビッグデータ」なる言葉が流行っている。ここでは「クラウド」上の「ビッグデータ」の処理について考察する。ただし「クラウド」とか「ビッグデータ」なる言葉はバズワードと化しているので，定義をはっきりさせる必要がある。

一般に「クラウド」と言えば，インターネット上の信頼できる企業にデータの保存や管理を任せることを意味するが，ここではもっと一般的にインターネット上にあるサーバとして定義する。このように定義するのは，将来普及するかも知れないグリッドコンピューティングを念頭に置いているからであって，その場合に個人あるいは小さな組織もコンピューティグパワーやデータをインターネット上に提供していくことになる。

「ビッグデータ」も盛んに使われる言葉であるが，一般に既存のデータベース管理ソフトでは扱えないほどの巨大なデータを指しており，サイズはテラバイトクラスに昇る。幾つかのネット企業ではユーザの動向を逐一記録し販売促進に活用している。またネット上には膨大な情報が公開されている。そうした情報もマーケッティングに利用可能である。そうして蓄積された情報は膨大になりすぎて，管理の方法や利用のためのツールを見直す必要に迫られているのである。

Webは現在のインターネット社会において最も重要な役割を果たしている情報共有の技術である。テラバイトクラス

* この論文は筆者のWebの記事［1］を基に書かれている。論文としての多少の修正が加えられている。

を想定する「ビッグデータ」に対して，Webのサーバが扱うデータは遥かに小さい。そのサイズはせいぜいメガバイトクラスである。Webではクライアントの求めに応じてデータをクライアントに送信する。データ転送に要する時間は回線の能力とデータサイズに依存する。そのため，ユーザの忍耐力を超えるような大きなデータを扱うわけにはいかないのである。

ではサーバのデータがメガバイトクラスを超え，ギガバイトクラスになると何が問題になるか？　ここではこの問題に焦点を当てる。

データサイズがギガバイトクラスになるとデータをインターネット回線を通して転送するのに適さなくなる。他方，データサイズに比べるとプログラムのサイズは遥かに小さい。データを処理するのに必要なプログラムのサイズはせいぜい数メガバイトである。従って，プログラムをサーバ側に送信して，サーバ側でデータを処理し，処理結果を受け取る方が速い。

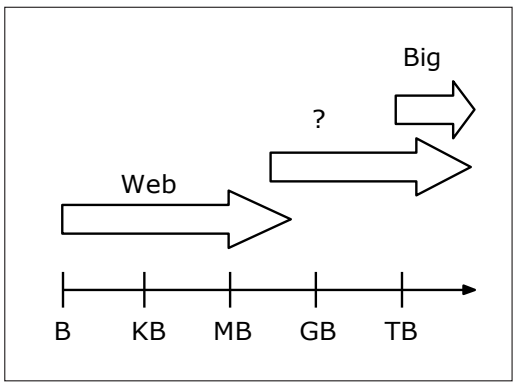

図1　Webデータとビッグデータの間

このようにWebの実用限界との関係で語られるデータサイズを何と言えば良いのだろうか？　筆者の知る限り，この問題に関しては広く知られた用語は存在しない。「ビッグデータ」を既存技術が適用できないほどの巨大なデータであると定義すれば，情報共有技術であるWebの技術が使えないギガバイトクラスのデータも「ビッグデータ」の仲間であると考えることもできる。しかし以下では混乱を避けるために，インターネット回線を通じて転送するのに適さないデータを「大きなデータ」と言うこととする。言うまでもなく，いわゆる「ビッグデータ」は「大きなデータ」である。

「大きなデータ」はインターネット回線を通じて転送するのに適さないのでサーバ上で処理される必要がある。従って，処理に必要なプログラムはクライアントがサーバにアップロードすることになる。クライアントの任意のプログラムを使って，サーバ上でデータ処理をするとなればセキュリティが大問題となる。従って以下では，セキュリティ問題に話を限定し，安全にサーバを運用し，かつクライアント側のニーズとセキュリティを確保するための方策に議論の焦点を当てることにする。

2. リモート実行

2.1 リモート実行の過去，現在，未来

ここではクライアントのプログラムを，遠く離れているサーバ上で実行することをリモート実行と言うことにする。この意味でのリモート実行の歴史はインターネット胎動期（1970年前後）に現れたTelnetとFTPから始まる。サーバの利用者はサーバ上に利用者の個人スペース（ホームディレクトリ）を与えられ，アクセスに必要なパスワードをサーバの管理者から知らされる。このスタイルは現在でも変わらない。

現在では個人所有のコンピュータ，いわゆるパーソナルコンピュータが普及し，大抵のことはパーソナルコンピュータで処理できる。そのためにリモート実行のニーズが少なくなっている。ホームページを運用している場合には保守のためにリモート実行が要求されることがあるが，それ以外の場合には高性能なコンピュータを使った特殊な計算を行いたい場合と，サーバにしかない大きなデータにアクセスしたい場合に限られるだろう。いずれも現在では普通の人々にとっては縁のない世界である。

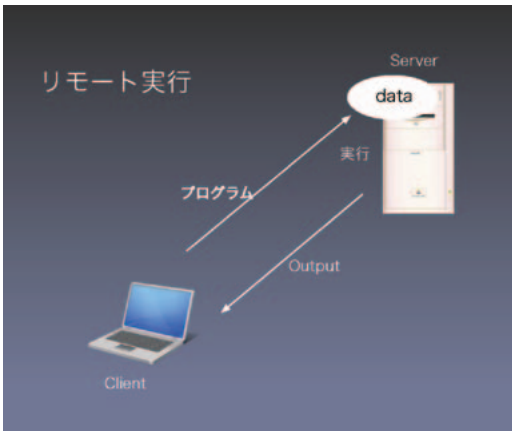

図2　リモート実行

データが大きいときにはリモート実行の方が理にかなっている。データよりもプログラムの方がはるかに小さいのだ

しかし，将来はどうであろうか？　科学の発展にとって収集したデータは可能な限り公開すべきである。生に近いデータが公開されていれば多様な視点からの分析が可能になる。視点が異なれば予想していなかったような発見があるかも知れない。現在は残念ながら特定の視点からの調理済みのデータしか公開されない。公開をWebで行っている限り，そのようになる。また，IT化された社会の中では自動収集された膨大なデータが蓄積される。そのようなデータが適切に公開されていけば，社会にとって有用な情報が得られる可能性がある。分析視点の多様性を重視するならば，分析者のプログラムをサーバ側で実行できる必要がある。

2.2 ホームディレクトリが必要とされる理由

現在，サーバの利用者はサーバ上に利用者の個人スペース（ホームディレクト

リ）を必ず与えられる。なぜ与えられるのか？

マイクロプロセッサが現れる前の時代には，コンピュータと言えば大きく高価で，個人が独占的に使用できるような装置ではなく，共同で利用せざるをえなかった。当時のコンピュータはホストとも呼ばれていた。利用者は端末と呼ばれる装置を使ってホストを利用していた。端末の処理能力は不十分で，利用者が打ち込んだ命令をホストに伝えるだけであった。そのために，ホストには利用者ごとの記憶スペースが割り当てられ，利用者は端末を通じてホスト側にプログラムを作成し，ホスト側でプログラムを実行する他はなかった。

マイクロプロセッサが現れて，個人が独占的に使用できるコンピュータが出現した。それらはワークステーションと呼ばれ，その上でプログラムを作成し実行することが可能となった。高い処理能力が必要な場合には共同利用のホストが使えた。ホストにはこれまで通りに利用者の個人スペースが与えられた（図3）。プログラムを編集し保存するためと，FTPによるファイルの受け皿として，ユーザごとのスペース（ディレクトリ＝フォルダ）が必要であると信じられてきた。この状況は現在でも変わらない。

3. ネットワークベースのマウント

3.1 リモートマウント

リモートマウントはサーバのファイルシステムをクライアントのファイルシステムの一部であるかのように見せる技術である（図4）。この技術を使えば，サーバへのファイル転送のためにFTPは不要になる。FTPでやっていたことはOS

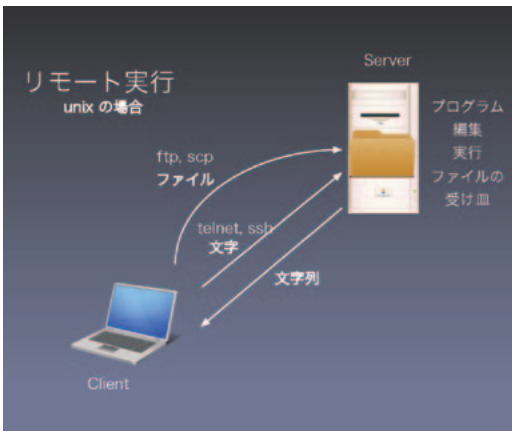

図3　リモート実行とホームディレクトリ

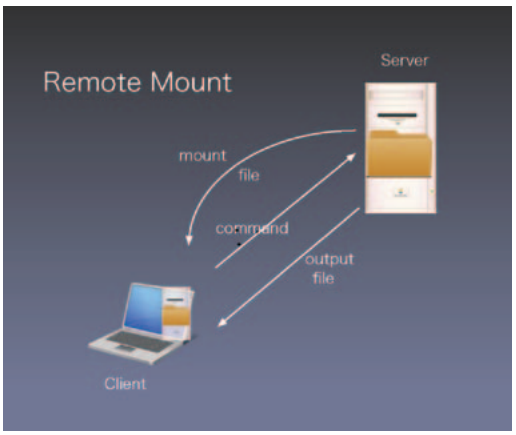

図4　リモートマウント

付属のコピーコマンドでやっていけるのである。

リモートマウントのメリットとしては，普通のユーザにとっては

- クライアント側でサーバのファイルが編集できる
- Drag & Dropでサーバへのファイル転送ができる
- マウスを使ってサーバのファイルをブラウズできる

などが挙げられようが，パワーユーザあるいはシステム管理者にとっては見方が違うであろう。彼らにとってはOS標準の基本的なツールだけでサーバのファイルを扱えるのが大きい。例えばscpコマンドを使わず，クライアントから実行するcpコマンドでサーバとクライアント間のファイル転送が可能である。さらに，クライアントで実行可能な馴染みのツールがサーバ上のファイルに対して一様に適用可能であるので，開発あるいは管理が容易になるなどの利点を挙げるだろう。

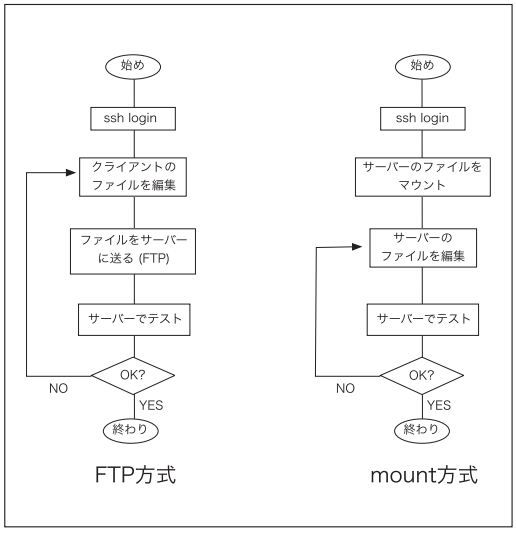

図5　プログラムが完成するまでの流れ

図5はリモートマウントを利用してサーバで実行可能なプログラムが完成するまでの流れをFTPと比較している。マウント方式の方が手間が省けていることに注意する。

3.2　リモートマウントのレイテンシ

ネット上にはマウントはレイテンシ（遅延時間）が大きすぎてLANレベルでしか実用にならないと述べている記事がいくつか存在する。インターネットでのマウントレイテンシは原理的な問題が絡んで改善しにくいと言う［3，4］。その根拠は，光の伝達速度が有限であり（光ファイバーの中での光の伝達速度は，真空中の光速の2/3程度である），マウントのプロセスではRPC（Remote Procedure Call）の技術が使われているために，マウントが完了するまでにサーバとクライアントの間で多数のメッセージのやり取りが発生する。そのために地球規模でのWANでのマウントでは時間が掛かりすぎて実用にならないと言う。こうした議論はいずれもLAN環境を前提にして設計されたNFSやCIFSを話題に採り上げている。また記事が作成された時期も古い。現在では実際にどの程度

のものか？　Plan9の例を紹介する[1]。

まずマウントレイテンシの定義であるが，ネットの議論にはリモートマウントされたファイルの転送速度との混同があるように思えるのではっきりさせておく。ここではマウントの開始要求からマウントが完了するまでの時間を問題にする。具体的にはtimeコマンドを使って

```
time マウントコマンド
```

のように測定する。測定値の中にはDNSの名前解決や認証に要する時間も含まれている。パスワードの手入力の時間を測定値から排除するために，認証は認証エージェントを使った自動認証の仕組みを使う必要がある。

紹介するのは筆者の自宅からBell Labsのサーバをマウントする時のレイテンシである。日本からアメリカまでの距離でのレイテンシの目安になるであろう。測定してみると，レイテンシには結構なばらつきがある。ネットワークの混み具合も関係するが，2回目以降のマウントの場合にはクライアントのキャッシングによってレイテンシが大幅に短縮される。欲しいデータはキャッシングの影響を排除した時間である。そのためにはクライアントを立ち上げた直後に測定することになる。そうして得られた筆者の環境でのレイテンシは殆どの場合1秒台であるが，時には数秒かかることもある。なお筆者の自宅はインターネットと1Gbpsの光回線で繋がっている。実験に使用したクライアントはWiFi（802.11n）を使って家庭内LANと繋がっているのでバンド幅は1/2程度に小さくなるのであるが，結果には大きな影響はないであろう。インターネット回線における実効的なバンド幅はさらに小さいだろうから。

Plan9では生まれた当初（1992年）から，Bell Labsのファイルシステムをローカル側にマウントすることによってソースプログラムの更新を行っている。他のOSの場合は全体をダウンロードして初めて更新の内容が分かるのであるが，マウント方式だと，改訂されたファイルの一覧を基にして必要なファイルのみをコピーすれば済む。ネットワークのバンド幅がまだ大きくない時代においてもマウント方式は実用的に使われていたのである。

マウントに必要なRPCの回数やマウントによるファイルコピーの速度は分散ファイルシステムの設計に強く依存する。ここに述べたのはPlan9によるマウ

[1] Plan9とは1992年にBell Labs（ベル研究所）からリリースされたOSである。開発グループはKen ThompsonやRob PikeなどUnixの生みの親たちであり，ネットワーク時代の前に生まれたUnixをネットワーク時代に正しく適応させることが開発の目標となっていた。またUnixの経験の上に立って，問題点を洗い出し，その解決のための新しい仕組みが提供された。なお，Plan9の正式名称は“Plan 9 from Bell Labs”であり，この省略形は正式には“Plan 9”であるが，ここではさらに簡単に，ネット界で普通に使われている“Plan9”を用いる。Plan9については文献［50］及び文献［51］に詳しい。

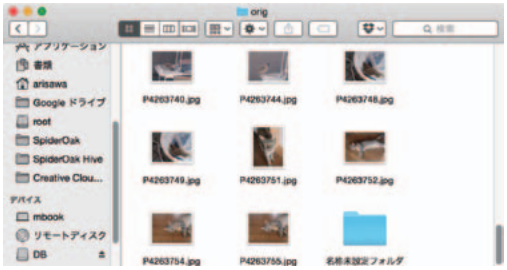

図6　Macのファイルブラウザ

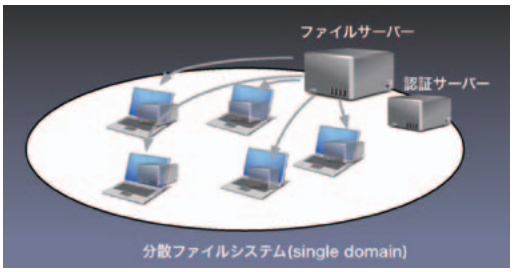

図7　分散ファイルシステム

ントで他のシステムの参考にはならないであろう。例えば文献［5］にはsshfsが非常に遅いという苦情がある。残念ながらPlan9以外でのマウントレイテンシの実測値が手に入らない。

マウントレイテンシに関する誤解の一つに，ファイルブラウザに表示されるまでの時間との混同がある。マウント要求を出してファイルブラウザにファイルの一覧が表示されるまでの時間は，マウントされるのに要する時間と，ファイルブラウザの表示に要する時間との和である。後者はファイルブラウザの設計と深く関わっている。

図6のようにファイルブラウザがファイルの内容を表示している場合には，要する時間は，フォルダーの中のファイル数，ファイルの総サイズ，ネットワークの実効速度が関わっている。そのために，大抵の場合にはLANの中でしか実用にならないであろう。ここではMacの例を挙げたがWindowsでも同様である。

3.3　分散ファイルシステム

クライアントやサーバにネットワークを通じてファイルシステムを提供しているのが分散ファイルシステム（Distributed File System）[2]である。ここにはリモートマウントの仕組みが使われている（図7）。

分散ファイルシステムは大学などのLAN環境では既に古くから整備されている。これによって，学生用のパソコン教室では，どのコンピュータを使っても学生はサーバに保存されている自分のファイルにアクセスできるのである。

表1に示すように，いろいろな分散ファイルシステムが存在し，OS依存性が強い。（強かった）

[2]「分散ファイルシステム」よりも「ネットワークファイルシステム」の方が分かりやすい呼称だが，この名前は既に特定の製品を指す名前として使われている（Sun NFS）[6]。なお，「分散ファイルシステム」の呼称は普通名詞として広く使われてきたにも関わらずMicrosoftが自社の製品名を表す名前として使い出したので混乱している。ここでは「分散ファイルシステム」を製品名ではなく普通名詞として使っている。

表1　よく知られている分散ファイルシステム

server	client	名称	製作者	適用範囲	公表年度
Unix	Unix	NFS（Network File System）ver.2	Sun Microsystem	LAN	1984
Unix	Unix	NFS（Network File System）ver.4	IETF	LAN/WAN	2003
Unix	Win	Samba[a]	Open Source	LAN	1992
Win	Win	DFS（Distribute File System）	Microsoft	LAN/WAN	2008
Win	Win	CIFS（Common Internet File System）[b]	Microsoft	LAN/WAN?	1996
Win	Unix	Windows NFS Client	Microsoft	LAN	1999

[a] UnixにはSamba［10，11］をクライアントとして利用するのもある［11，12］

[b] 一応WAN環境でも使えるようであるが，問題はありそう［9］。なお，CIFS は廃止になりそうである［7，8］

この表で「Win」とはWindowsのことである。また「WAN」とはインターネット環境を指す。LANをVPN（Virtual Private Network）で結んだネットワークは，管理面から見てLANの一種だと考える。この表からは多くのものが省かれている。例えば分散OSであるPlan9は生まれた時から高性能な分散ファイルシステムを備えていた。また最近のUnix系の分散ファイルシステムはFUSEをベースにしているが，それらも表から省かれている。

3.4　FUSE

現在，ファイルシステムのOS依存性を弱めるための新しい技術（FUSE）が注目されている。そしてCeph，Gfarm，GlusterFSなど最近の分散ファイルシステムの設計はFUSEベースになっている［13］。さらに，既存のファイルシステムもFUSEベースで再設計する動きがある［14］。

FUSE（Filesystem in Userspace）とは，ファイルシステムのプログラムコードをカーネルの外に置く技術である。カーネルにはFUSEを実現するための汎用の小さなコードが含まれている必要がある。最近では主要なOSでFUSEがサポートされている。（アイデア自体は1990年前後に発表されたMachやPlan9に由来する）

ファイルシステムがカーネルと固く結び付いていると，ファイルシステムの開発自体が困難であるばかりか，新しいファイルシステムの導入がOS提供者に限定され，ユーザニーズが反映され難くなる。また分散ファイルシステムを構築する場合にはOSを統一しなくてはならなくなる。FUSEによって，このような制約から解放される。

FUSEを応用したファイルシステムは

多数ある。FUSEベースの分散ファイルシステムはグリッドコンピューティングとの関係で注目されており，いくつか開発されている。その中でもGfarm［15, 16］は国際的にも高い評価を受けている分散ファイルシステムである。Gfarmは日本発の技術であり，ホームディレクトリを自動マウントできるように工夫されている［17］。

個人が手軽に使えるFUSEベースのファイルシステムとしてsshfsが注目されている。これまでのUnix系の分散ファイルシステムに比べて

- 個人利用として手軽に使える（インストールと管理が簡単）
- WAN環境でも使える
- 家庭内LANの中でのデータ共有に便利
- 多様なOS間で共通に使える

などの特徴がある。

3.5　Plan9の逆向きマウント

分散ファイルシステムにおける通常のマウントは，サーバ側のファイルシステムをクライアント側のファイルシステムにマウントするのであるが，それに対して，Plan9では逆向きマウントが実現している。つまりクライアントのファイルシステムをサーバ側にマウントする（図8）。

逆向きマウントをサポートしているサーバ側のOSは（現在のところ）Plan9だけである。クライアント側はPlan9の他，Unix，Linux，Mac，Windowsなどでサポートされている。逆向きマウントだけでは意味がないので，実際にはクライアントのリモート実行コマンドと組になっている。リモート実行コマンドを実行すると，同時に自動的にクライアントのファイルシステムがサーバ側にマウントされるのである[3]。リモート実行コマンドとしてはPlan9端末（Plan9クライアント）ではcpu，Plan9以外のクライアントではdrawtermを使う［18，19］。

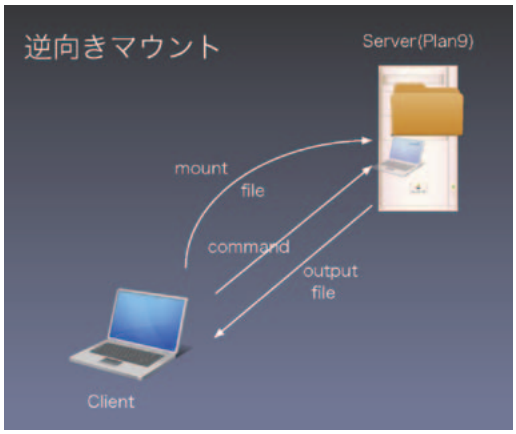

図8　Plan9の逆向きマウント

正方向のマウント（サーバのファイルをクライアントに見せるマウント）の場合には，クライアントのプログラムを

[3] ローカルシステムをリモートシステムにマウントするのに必要な通信チャネルは，コマンドを送る通信チャネルと兼ねている。そのためにファイアウォールの中からでも問題なく接続できる。これが可能なのは，通信チャネルが多重化されているからである。

サーバで実行するプロセスは次のようになるであろう。

1. サーバをローカル側にマウントする
2. プログラムをサーバ側にコピーする
3. sshコマンドでリモートログインする
4. プログラムを実行する

これに対してPlan9の逆方向マウントだと

1. cpuコマンドでリモートログインする
2. プログラムを実行する

と手順が簡略化される。この簡略化はグリットコンピューティングでは決定的に重要な意味を持っている。なぜならグリットコンピューティングでは膨大な数のサーバによる並列実行が想定されているからである。正方向マウントしか持たないUnixベースのグリットコンピューティングでは最初の2つ（サーバをローカル側にマウントする，プログラムをサーバ側にコピーする）を1回で済ませるための仕掛けが望まれる。

4　データサーバ

ここではデータの提供とデータ処理のためのCPUパワーを提供するサーバについて考えて見る。以下，これをデータサーバと言う。データサーバにはどのような特性が求められているのだろうか？

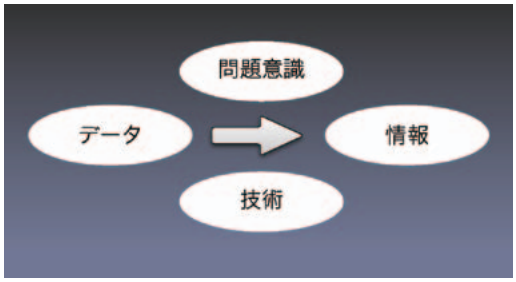

図9　情報の抽出

4.1　データの解析

データを還元処理によって得られたものが情報である[4]。極端には1ビットにまで還元される。還元処理は問題意識（目的）に依存し，大きなデータではさらに技術が求められる（図9）。

記録メディアの価格が低下し続けた結果，最近では膨大なデータを保持するようになった。データ収集時にデータを厳選するよりも，集められるデータは片っ端から集め，後で整理するやり方が可能になっているのである。そこで，我々はデータに対して次の見方を採ることができる：

- データ:そのままではゴミの山
- 情報の抽出には，問題意識と技術が必要

情報の抽出は一意的ではない。多様な問題意識が存在し，そこから多様な情報

[4] データと情報の関係に関しては多様な見解が存在する。概ね，データは計算機寄りのもの，情報は人間寄りのものと理解されているようである。しかしここでは情報量で説明する。

が生まれる。分析する視点が異なれば，意外な結論も得られるであろう。従ってデータを公開し，多様な視点で検討できるようにすることが重要である[5]。

データが小さい場合には既存技術でもやっていける。サーバからユーザがダウンロードすればよい。しかし，大きい場合には…

Webではサーバ運営者の問題意識にそって情報が抽出される。今はそれで我慢しているのである。

4.2 ホームディレクトリは本当に必要か？

大きなデータの処理にはリモート実行を許す必要がある。つまり，データを移動させないで，サーバ上で直接処理する必要がある。その場合には

- データへのアクセス
- ユーザが作成したプログラムの実行許可
- 結果の受け取り
- 会話的実行

が要求される。「会話型実行」を含めたのは，情報抽出の過程で多くの試行錯誤を必要とするからである。

これらを行うために，現在の方式（Unixのリモート実行）ではユーザ登録の際にパスワードとホームディレクトリが与えられる。

しかし，データサーバにとって，利用者は一時的である。必要な結果が得られればアクセスするニーズがなくなるであろう。そうした利用者にホームディレクトリを与えるのは合理的ではない。サーバにそのためのディスクスペースが要求され，しかも必要な大きさは前もってはわからない。サーバ側としては十分な大きさのディスクスペースを提供するしかないであろう。

利用者が彼らのスペースに保存しているファイルは，一時的に必要とされ，使い終わったものなのかも知れないし，後々に必要とされる大切なものかも知れない。また他人には見られたくないものなのかも知れない。サーバの管理者には適切な管理義務が発生する。

ホームディレクトリは本当に必要なのだろうか？　必要ではないなら，こうしたことに悩まされることはないのである。ホームディレクトリが必要とされる理由は，FTPなどによるファイル転送の際にファイルの受け皿が必要と考えられ

[5] データが公開されない原因には社会的なものと技術的なものがある。社会的なものは，過度な競争の結果として研究成果の抱え込みを必要悪と考える風潮，データの1次発掘者が低く見られている風潮（分析しないと研究成果にならない)が考えられる。技術的なものとしては，大きなデータを公表する手段を欠いていたために，これまでは還元した情報しか公表できなかったことが挙げられよう。1次データを公表しないで成果だけを発表しているアカデミーの習慣が研究不正の温床になっている。

るからである[6]。しかし図8について考えてみよう。クライアントのプログラムをサーバ側で実行するにあたって，Plan9の場合には，実はホームディレクトリは大した役割を果たしていないのである。

5. ホームレスデータサーバ

5.1 データサーバの新しい方向性

サーバの管理者にとってグリッドユーザは特殊な存在である。ボランティア的にサービスを提供しているにすぎない相手である。しかも顔が見えない相手である。彼らに対して貴重な記憶装置を特別に準備するのは抵抗があるだろう。ユーザのデータはユーザが所有する記憶装置に保管するのが管理者側とユーザの双方の利益である。そこで図10に示すサーバ側の要求は実現可能か否かを考える。

具体的には次の要求仕様を考えてみる：

- クライアントの認証は行う
- クライアントごとにホームディレクトリを与えない
- クライアントに一切の書き込みを許さない
- クライアントのプログラムはサーバ側で実行可能
- クライアントのプログラム編集はローカルサイドで行うことが可能
- クライアントはサーバ側で実行されたプログラムの実行結果を受け取ることが可能
- クライアントにサーバでの会話的実行を許す

サーバ側の要求は厳しい。

この実現にはPlan9の逆方向マウントを利用すれば可能である。逆向きマウントによってサーバ上で直接クライアントのプログラムを参照できる。ただしPlan9自体はホームディレクトリの存在を想定しているので，多少の手直しが要求される。特にセキュリティ上の理由からカーネルのパッチが要求される。この仕様は，実際に筆者のグリッドサーバ[7]で実現されている［2］。

ホームレスデータサーバの場合には，クライアントとサーバとの関係は図11のようになる。

図11は図8と似ているがホームディレクトリが存在しない。工夫すればホームディレクトリ無しにやっていけるのであ

[6] 他にもOSごとに存在理由があるが，除去可能な理由か否かが問題である。例えば公開鍵を使ったssh認証方式があるが，ホームディレクトリにログイン認証に必要な情報が置かれる。そのような場合にはホームディレクトリは必須になる。

[7] グリッドサーバとは，コミュニティーのメンバーが自由に使えるサーバ群であり，リモート実行を許す。通常は多数のサーバ上での並列実行によって1個のコンピュータでは実現できないような大きな処理能力を得るために使われる。もちろん個々のサーバでユーザ登録しないで，ユーザ登録は一箇所で済ませる仕組みを持つ。

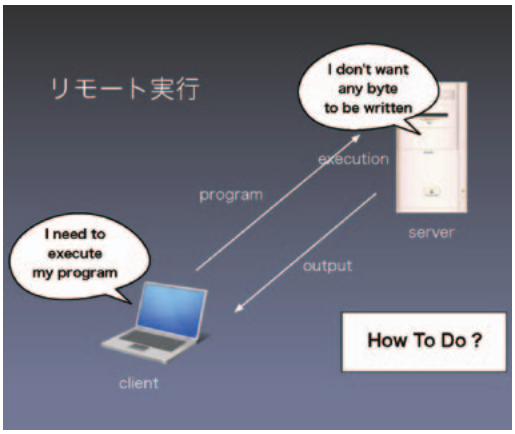

図10　Don't write to me!

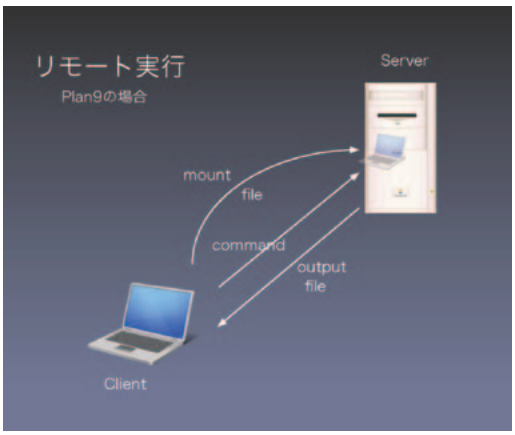

図11　ホームディレクトリ無しの逆向きマウント

る。

筆者のサーバはグリッドコンピューティング用に設計されている。そのために認証はマルチドメイン認証に対応している。しかも，認証の対象となるサーバの所属ドメインがマルチであるばかりではなく，認証チケットを発行するドメインもマルチである。現在はBell Labs のチケットあるいは筆者が独自に発行するチケットで利用可能になっている。

さらにグリッドサーバを踏み台とした不正アクセスを防ぐために，サーバからのネットワークアクセスを防止している[8]。

5.2　実行例

次の実行例はPlan9のユーザを想定している。彼らの多くはBell Labsのアカウントを持っている。そこで，このアカウントの保有者に対してサーバへのアクセスを許可するようにサーバが設定されている。サーバにログインするためには，Plan9の認証エージェントfactotumに対して次の認証キーを登録しておく。

```
key dom=outside.plan9.bell-labs.com
proto=p9sk1 user=XXXXX
!password=YYYYY
```

ここでは，紙面の都合上，3行で書いているが，実際には1行である。また，XXXXXはBell Labsのアカウント名であり，YYYYYはBell Labsでのパスワードである。ドメイン名が指定されていることに注意する。このサーバはマルチドメイン認証に対応しており，サーバ側は他のドメインのユーザも受け付ける。

サーバにログインするにはcpuコマンドを使う：

[8] 完全を期してカーネルへのパッチで実現している。

cpu -h grid.nyx.link

-k 'dom=outside.plan9.bell-labs.com'

ここでも，紙面の都合上，2行で書いているが，実際には1行である。また，grid.nyx.linkは筆者のサーバであり，outside.plan9.bell-labs.comのアカウントを使うことを指定している。一般的に言えばfactotumには複数の認証キーが登録されているので，その内のどれを使うかを指示しているのである。ログインに成功すれば“grid%”のプロンプトが表示される。

まず最初にpsコマンドを実行してみるとよい。図12は筆者がBell Labsのアカウントでログインした場合の結果である。Bell Labsのアカウントでログインした場合にはプロセスのオーナは

XXXXX@outside.plan9.bell-labs.com

となっている。多様なドメインのユーザの利用を許し，プロセスが干渉しない保障を得るためには，このようにプロセスのオーナ名にドメイン名を含めざるを得ないであろう。なお，psの表示が“bell-”で切られているのは，単に表示幅の節約のためである。

次に

ls /usr

を実行してみる。するとホームディレクトリの一覧が表示される。その中には/usr/noneと/usr/arisawaの他に，クライアント側のユーザの一覧が見えるはずである。例えばクライアントのユーザ名がbobであれば，/usr/bobが見える。もちろん/usr/bobの下にあるディレクトリやファイルはクライアントのものである。bobはそれらを使って，自由に自分のプログラムを実行できる。

もしもbobの他にcarolもログインし

```
grid% ps
arisawa            1    0:00   0:00    256K Await   bootrc
arisawa            2    0:00   0:00     0K Wakeme   mouse
...
none             369    0:00   0:00    132K Open    listen
none             370    0:00   0:00    132K Open    listen
arisawa@outside.plan9.bell-   20188  0:00  0:00    124K Await  gcpu
arisawa@outside.plan9.bell-   20195  0:00  0:00    240K Await  rc
arisawa@outside.plan9.bell-   20196  0:00  0:00    124K Pread  gcpu
arisawa@outside.plan9.bell-   20247  0:00  0:00    116K Pread  ramfs
arisawa@outside.plan9.bell-   20252  0:00  0:00     92K Pread   ps
grid%
```

図12　psコマンドの出力

ていたらどうなるか？　Plan9とUnixとの大きな違いの一つにユーザが見る名前空間の根本的な違いがある。Plan9においては異なるユーザは異なる名前空間に属している。その結果，carolは/usr/bobを見ることはないし，逆もまた然りである。

最後にPlan9のテキストエディタacmeを実行してみる。このエディタは（サーバ側で実行しているにもかかわらず）マウスを使え，そしてファイルブラウザを兼ねているのでサーバの様子をざっと見るのに良いであろう。もちろんファイルの編集もできるが，編集はローカル側で行った方がレスポンスが良い。

筆者のサーバではシステム領域や他ユーザの領域への書き込みは禁止されている。書き込みはクライアント側にあるユーザの領域と，ユーザの便宜のために準備されたramfsにのみ許される。

ramfsとはメモリーの中のファイルシステムであり，Plan9ではユーザごとに割り当てられる。ramfsは，ログインで生成され，ログアウトで消滅する。一時ファイル用のディレクトリである/tmpはramfsで実装されている。どのユーザも/tmpである。/tmpもPlan9の私的な名前空間の中にあり，他のユーザと干渉し合うことはない。またクライアントのファイルシステムは/mntにマウントされるが，他のクライアントとは別の名前空間にあるために干渉し合うことはない。

グリッドユーザが見る名前空間は，システムユーザが見る名前空間の一部である。Unixではファイルの保護は許可ビットで与えるが，Plan9ではその他にカプセル化によって隠蔽できる。例えばシステムユーザの個人的なファイルの他，/sys/log，/mailなどがグリッドユーザには隠蔽されている。

5.3　ホームレスデータサーバのレイテンシ

世間一般の認識では世界規模のWANレベルのマウントは遅くて実用になるはずがないと言うことらしい。このような認識は著名な雑誌のレフリーですら持っている[9]。彼らはUnixやWindowsの常識で考えている。しかしPlan9のマウントは速い。既に述べたように，日本からアメリカ（Bell Labs）までのマウントレイテンシは数秒である。マウント後に続くユーザの作業時間を考えた時には，この時間は完全に無視できるだろう。

ではホームレスデータサーバのレイテンシはどうか？　具体的には

```
time cpu -h grid.nyx.link
-k 'dom = outside.plan9.bell-labs.com'
-c pwd
```

[9] 筆者はこの理由によって論文への掲載が拒否された。

をアメリカから実行して貰い[10]，レイテンシを測定する。アメリカから日本の筆者のサーバに接続して，pwdを実行するのに必要な時間の測定である。

認証サーバとしてはBell Labsのものが使われているので，cpuコマンドの実行によってログインするまでには次の3ステップが内部で実行される[11]。

1. クライアントはまず筆者のサーバにアクセスし，チケットを入手するのに必要な情報を受け取る
2. クライアントはその情報をBell Labsの認証サーバに提示し，チケットを受け取る
3. クライアントは，そのチケットを筆者のサーバに提示し，ログインの許可を請う

さらに筆者のサーバではユーザの使い心地を向上させるために，クライアントと交信しながら幾つかの内部処理を行っており，この事がレイテンシを幾分大きくする。

幸いシアトルに住む友人が実験に協力してくれた。報告によれば3回の実験で結果は各々7.58秒，7.22秒，8.17秒である。この値はPlan9による日本からアメリカまでのマウントレイテンシの数倍である。cpuコマンドが完了するまでのサーバとの交信回数は通常のマウントに比べて数倍に昇るので，妥当な数値であろう[12]。この数値が大き過ぎて実用にならないのか否かは行われる内容に依存するが，殆どの場合には問題にはならないであろう。特に，会話的実行環境では完全に無視できる。

5.4 ホームレスデータサーバを支える技術

筆者のホームレスデータサーバはPlan9の技術に基礎を置いている。最も重要で困難な部分はPlan9の標準環境の中で既に実現している。すなわち

- cpuコマンドによるリモートアクセスの技術
- 逆向きマウントの技術
- 認証エージェントに基づく認証技術
- プロセスごとに自由に構築できる名前空間のカプセル化技術

などである。Plan9の標準的なリモートアクセスではサーバ側にホームディレクトリの存在を前提にしているが，この要件を省いたのが筆者が提唱するホームレスデータサーバである。それでもデータサーバとしてのニーズが満足されるように，またセキュリティ上の問題が発生しないようにサーバの設計を行う必要があ

[10] これも1行のコマンドであるが，紙面の都合上3行で表示されている。

[11] 詳しくは文献［2］のAppendixを見よ。

[12] Plan9にも幾つかの変種が存在する。筆者のは9frontである。これは，このままではレイテンシがいささか大きい。そのために筆者は少しだけ手を加えている。その結果，標準環境に比べてレイテンシは1/3程度になっている。

る。以下に設計の要点を解説する。筆者のサーバではユーザを次のように分類している：

- グリッドユーザ
- システムユーザ
- ホストオーナ
- ユーザnone

グリッドユーザにはホームディレクトリを与えない。筆者はグリッドユーザとしてBellLabsに登録されたユーザを想定している。従って，ここに登録されたユーザは筆者のサーバを使えるように設定している。ところが筆者はBellLabsにユーザ登録されたユーザのリストは持っていないのである。従ってホームディレクトリは与えようがない。筆者のグリッドユーザをBell Labsのユーザに完全に限定してしまえば，ユーザ登録に関する一切の作業は必要がなくなり，またグリッドユーザは利用にあたって筆者と連絡をとる必要もない。

Plan9ではシステムユーザは本来ならネットワークが許されている。しかし，このシステムではグリッドユーザの巻き添えを食ってシステムユーザもネットワークができないようになっている。このサーバは家庭内のLANの中に置かれているために，セキュリティの関係でネットワークは困るのである。ネットワークの禁止は完全を期してカーネルレベルで行っている。Plan9のカーネルは，ユーザを3つに分類している。ホストオーナとnoneとその他である。そのためにグリッドユーザとシステムユーザの区別ができないのである。Plan9のホストオーナはUnixのrootに相当する。Unixと異なりホストオーナに固定した名前はない。マシンを立ち上げたユーザがホストオーナになるのである。

ユーザnoneはUnixのnobodyに相当し，主にネットワークサービスを受け持っている。Unixではnobodyの他にも，八百万の神様（デモン）を持っているが，Plan9ではnone一個で済ませている。

Plan9では名前空間をカプセル化できる。Unixでもある程度はできるが実用の域には達していない。筆者のグリッドサーバは筆者が普段使っているファイルサーバの下で動いている。従ってそこには私的なファイルも存在し，グリッドユーザからは，そうしたファイルの存在自体を隠したいのである。そのためにPlan9の名前空間のカプセル化が利用されている。

以上の説明をまとめると表2のようになる。

ホームレスサーバ自体はPlan9の標準環境に多少の手を加えれば実現できる。次の2つのコマンド：

- 認証エージェントfactotum
- cpuコマンド

にわずかのパッチを当てれば済む。クライアント側は標準環境のままで構わな

表2　筆者のホームレスグリットサーバにおけるユーザの分類

ユーザ	ネットワーク	名前空間	ホームディレクトリ	仕事
グリッドユーザ	不可	限定する	無	
システムユーザ	不可	限定せず	有	
ホストオーナ	可	限定せず	有	システムメンテナンス
none	可	限定せず	有	ネットワークサービス

い。

しかしPlan9は筆者のようなサーバの使い方を想定していないので，そこから発生するセキュリティ上の問題を解決しなくてはならない。例えば，本来のPlan9では，どのユーザもnoneになれるとされている。しかしグリッドユーザもnoneになれるようであれば，表2に示した分類自体が意味をなさないのである。表2の通りに働くためにはカーネルのパッチ当てが必要になる。筆者のサーバの場合以下のようなパッチが当てられている。

- ホストオーナだけがユーザnoneになれる
- グリッドユーザによるサーバ内からのネットワークを防止する[13]
- グリッドユーザに提供されている名前空間を完全にロックする

[13] 既に述べたように，カーネルレベルではグリッドユーザとシステムユーザの区別はできない。従ってホストオーナとnone以外は内部からのネットワークが防止されていると言う意味である。

6　グリッドコンピューティング

6.1　グリッドコンピューティングのパラダイム

筆者がホームレスサーバを考える動機になったのはグリッドコンピューティングである。グリッドコンピューティングが目指すパラダイムは図13で上手に表現されている。

図を注意深く観察すると，小さな魚は実はPCではなく，ワークステーション（PCより少し上位クラスのコンピュータ）である。グリッドコンピューティングを上手にこなすには，Unixワークステーションクラスのコンピュータが必要と考えたのであろう。

この図に示す考えは，実はGoogleやAmazonなど著名なネット企業がシステムを組む際に既に採用しており，クラスターコンピューティングとも呼ばれている。高価なスーパーコンピュータでシステムを組むよりも，安価な市販品を多数組み合わせてシステムを組む方が安く済むからである。さらに大規模なデータ処

図13　PCの大群がスーパーコンピュータを飲み込む（文献［20］）

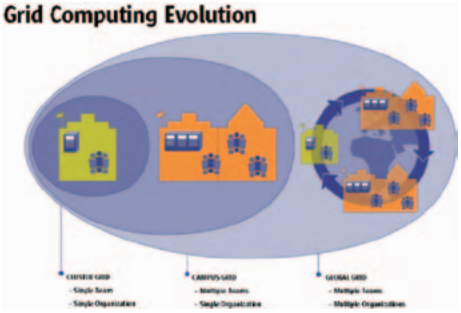

図14　Grid Computing Evolution（文献［27］）

理はこの方が効率的で，また障害に対する耐久性が高い。

グリッドコンピューティングに結びつく考えは1990年代の初頭からすでに提唱され，研究機関で模索された。1990年代末に至るまでのグリッドコンピューティングの研究と将来への展望は，Ian Fosterたちの本に詳しく纏められている［21］。研究機関へのグリッドコンピューティング普及の中心的な役割を担ったのはGlobus［49］で，現在におけるグリッドコンピューティングのソフトウェア基盤を築き上げた。

データセンターにおけるクラスターコンピューティングに対する，研究機関を結ぶグリッドコンビューティングの難しさは，参加するコンピュータの多様性にある。データセンターの場合にはコンピュータは仕様を統一できる。しかし，研究機関を結びつけるグリッドネットワークでは管理主体が異なっているために仕様を統一するのは難しい。特別の努力が必要とされているのである[14]。

6.2　グリッドコンピューティングの分類

次の図14はGlobusのホームページに載っているGentzschの論文［27］から借用している。Gentzschは，データセンターの中で実現されているクラスターコンピューティング（Cluster Grid）と研究機関を結ぶグリッドコンピューティング（Global Grid）の間に，中間的な形態があると言う。分類の視点は，グリッドの運用人員（team）と運用組織（organization）である。

中間的な形態として図では“Campus

[14] 日本では2008年に東大，京大，筑波大を結ぶグリッドコンピューティングの実証実験が開始されたが，その時には基本仕様を統一するために，3大学による共同入札が行われている［23］。2012年には実証実験に関するシンポジュームで認証基盤にかんする問題が取り上げられている［24］。

Grid”となっている。しかし，この著者の図のキャプションでは“Enterprise”となっているのである。両者に共通しているのは，単一組織が運用している点である。この場合にはシステムを組むに当たって認証システムは1つに統一される。それではクラスターとの違いはどこにあるか？　クラスターでは認証はもっとシンプルなはずである。もちろんクラスターの外から中へのアクセスは認証が求められるはずであるが，クラスターの内部で行われているコンピュータ相互のアクセスでは認証は不要であろう。

図の“Global Grid”についても他の呼び方がある。例えば“Collaboration Grid”である［25］。“Global Grid”はインターネットレベルのグリッドであることが強調されているのに対して，“Collaboration Grid”はグリッドを構成する組織が単一ではないことが強調されている。グリッドの難しさの本質は，グリッドの管理にあることを考え，ここでは“Collaboration Grid”を採用する。

そこで次ページ表3に筆者による分類をまとめる。ついでに筆者のホームレスグリッドサーバ（9grid）も表に組み込んでいる。

表の認証システムについて補足が必要であろう。Campus/Enterpriseグリッドでは，認証システムはシステム導入の際に既に統一されていると考えてよい。そのために導入は容易である。他方Collaborationグリッドでは，認証システムは統一されていないであろう。既に述べたように，異なる組織で認証システムを統一するとなると大きな努力が必要である。9gridで「統一不要」と書いたのは，認証を与える組織を統一する必要がないという意味である。認証メカニズムは統一する必要がある[15]。

表の「分散ファイルシステム」の欄では，サーバとクライアント間で分散ファイルシステムが利用可能か否かを問題にしている。グリッドコンピューティングに利用可能な多数のサーバが一つの組織に属していれば，それらを分散ファイルシステムで結ぶのは現在では当たり前のことと考えてよい。Collaborationグリッドでは，分散ファイルシステムについて「採用困難」と書いたが，最新の技術であるGfarmを使うと実用のレベルに達しているかも知れない。しかし筆者は評価の手段を持たないし，性能を評価した文献も知らない。

6.3　9grid

9gridと呼ばれるグリッドは，今回の筆者のホームレスデータサーバの以前に，歴史上2度現れ異なるグリッドプロ

[15] Plan9の中では既に統一されている。正確に言えば，Plan9は複数の認証メカニズムをサポートしているが，それらは認証エージェントfactotumを通じて統一されている。

表3　Gridの分類

分類	認証システム	分散ファイルシステム	適用範囲
Cluster	不要	統一可能	LINK[a]
Campus/Enterprise	必要（統一可能）	統一可能	LAN[b]
Collaboration	必要（統一困難）	採用困難→FTP	WAN
（9grid）	必要（統一不要）	不要	WAN

[a] リンクというのはLANよりも狭い範囲で，イーサーネットのブロードキャストが届く範囲である。セグメントとも呼ばれる

[b] 物理的なLANよりも，単一ドメイン構成になっていることが本質的である。この場合，仕様を統一できる

ジェクトに対して使われている。最初に現れたのはBell LabsとUniversity of Calgaryとの共同研究プロジェクトのグリッドである［30］。この成果は文献［28］に纏められている。この内容はまたMirtchovskiの博士論文に詳しく解説されている［29］。これらの論文では，Plan9はグリッドコンピューティングに適したOSであると主張された。

彼らに刺激されて，メーリングリスト9fansに集まるPlan9ユーザがグリッドコンピューティングの実験を始めた。グリッドサーバがボランティア的に提供され，各自が各自のやり方でグリッドサーバを構成した。筆者もサーバを提供し，並列コンピューティングの実験を行い，そのソフトを公開している［31，32］。これが9gridの第2期である。このユーザーズグループによって，標準配布のPlan9に少し手を加えるだけでマルチドメイン認証が可能になることが見つけられた。彼らの実験の成果はPlan9 Wikiに纏められている［30］。Wikiにはグリッドサーバに対する新しいアイデアも述べられている。しかしながら，それらのアイデアは実現されることもなく第2期は終息した。9fansに集まるユーザの関心はもっぱら技術的な問題にあり，その解決のメドが立った段階で関心を無くしたと思われる。

もしかすると何十年か先に9fansによるこの時期の活動は別の視点から歴史家の評価を受けるようになるかも知れない。すなわち，研究所の高性能なコンピュータとネットワークの中で生まれ育まれたグリッドコンピューティングが，Plan9による新しい技術によって初めて研究所の外に踏み出し，普通のコンピュータと普通のネットワーク回線の中で実験されたと。

9gridの第3期になるか否かは不明だが，あれから10年，筆者は9gridを再び考えてみることとした。グリッドサーバの必要用件からホームディレクトリを除

去できるのではないかと考えたからである。ホームディレクトリをグリッドユーザに提供しなくてもよいのなら，グリッドサーバを気楽にユーザに提供できるだろう。これをホームレスグリッドサーバとは言わないのは，グリッドサーバは複数個の存在を想定しているのであるが，ホームレスデータサーバは今の所世界でただ一つしか存在しないからである。

6.4 グリッドコンピューティングの現在と未来

グリッドコンピューティングに関する2000年頃までの状況に関しては文献［34，35］に詳しいので，ここでは省略する。現在，研究機関でのグリッドコンピューティングは，研究の基本インフラとしてヨーロッパとアメリカで定着しているようである。

ヨーロッパでは2002年から2004年のData-Gridプロジェクト［36］，2004年から2010年のEGEE（Enabling Grids for E-sciencE）プロジェクト［37］を経て，2010年からはEGI（European Grid Infrastructure）プロジェクト［38］に引き継がれている。ここには2016年現在，世界中から200以上の研究機関が参加している［39］。アメリカでは早くも1988年から大学でのグリッドプロジェクトが動いており［44］，現在ではOSG（Open Science Grid）が中心になってグリッドコンピューティングを進めている［45］。Wikipediaによると2009年現在42大学がOSGに参加しているという［46］。

EGIとOSGに基礎を置いて，世界最大のグリッドWLCG（Worldwide LHC Computing Grid）が組織されている。この組織を調整しているのはCERNであり，LHC（Large Hadron Collider）から生み出される巨大なデータを世界中の研究者の間で共有することを使命としている［43］。

ある雑誌の記事［47］を次に紹介する。

> 「e-Science」という言葉をお聞きになったことがあるでしょうか。聞いたことのある方は「高度に分散化されたネットワーク環境で実施されるコンピュータを多用した科学などと言われ，主に自然科学の分野で，研究成果や研究過程で生み出される大量のデータを共有し，新たな研究への利活用を行おうとする取組み等といった理解をされているかと思います。一方で近年，大量のデータを共有，活用するというe-Scienceと似た取り組みが，あらゆる分野で盛んに行われつつあります。ビジネスの分野では「ビッグデータ」や「クラウドコンピューティング」と呼ばれるさまざまな技術やサービスが普及し，ログ等の大量の生データを解

析し，ビジネスに活きる知見を引き出す「データサイエンティスト」という専門家が注目を集めつつあります。自然科学分野では「オープンサイエンス」，「ビッグサイエンス」，「eリサーチ」等と呼ばれる取り組みが進みつつあり，一方で人文科学分野ではDigital Humanitiesという分野が隆盛し，研究分野を超えた学際的な研究も盛んになりつつあります。

「e-Scienceとその周辺〜現状とこれから〜」の編集にあたって［47］。（下線は筆者）

グリッドコンピューティングを支える理念はe-Scienceに示されている研究者間でのデータ共有であり，単に高速の計算環境を提供したいと言うことではないのである［34，41，46］。こうした欧米の動きに比べると日本は非常に遅れている[16]。

現在のCollaborationグリッドは研究機関のグリッドである。この分野はグリッドコンピューティングのニーズが高く，多数の高性能なコンピュータを集めやすい。また高速なネットワークが研究機関の間で整備されている。つまりグリッドコンピューティングが発展しやすい分野なのである。しかし，将来，世の中のIT化がさらに進行した時にグリッドコンピューティングが研究機関の外に広がる可能性はどうだろうか？　World Wide WebはCERNから始まり，研究機関に広がり，現在では世界中の人々にとってなくてはならない存在となっている。同様なプロセスを辿るのだろうか？　将来には家庭のあらゆるデバイスがインターネットと接続すると予想されている。そのコンセプトは「モノのインターネット（IoT）」と呼ばれている。そのような時代には研究所のスタイルではない新しいグリッドが求められる可能性が残されている。そこでは高性能なコンピュータや高性能なネットワークを求めることはできないし，またグリッドのために提供できる資源は限られてくる。さらに家庭内にサーバを設置するとなれば完全なセキュリティが求められるだろう。そしてグリッドユーザごとのホームディレクトリの提供はできそうもないだろう。筆者のホームレスデータサーバは，そうした未来を視野に置いた一つの提案である。

[16] 日本では2008年にようやく3大学のグリッドから始まり，現在では旧七帝大を中心とした共同利用の環境が整っている。もちろん成果に関しては公開されており，他に毎年シンポジュウムが開かれている［48］。

References

［1］Kenji Arisawa: “Beyond The Web Homeless Data Server—”（2016）
http://plan9.aichi-u.ac.jp/9grid2/beyond1.html
http://p9.nyx.link/9grid2/beyond1.html（mirror）

[2] Kenji Arisawa: “A New Grid Server” (2015)
http://plan9.aichi-u.ac.jp/9grid2/9grid.html
http://p9.nyx.link/9grid2/9grid.html (mirror)
[3] EETimes: “File Sharing on the WAN: A Matter of Latency”
http://www.eetimes.com/document.asp?doc_id = 1272058 (2004)
[4] acmqueue: “Bound by the Speed of Light”
http://queue.acm.org/detail.cfm?id = 1900007 (2010)
[5] Super User: “faster way to mount a remote file system than sshfs?”
http://superuser.com/questions/344255/
[6] Wikipedia: “Network File System”
https://en.wikipedia.org/wiki/Network_File_System (参照2016)
[7] デジタルアドバンテージ:
「ファイル共有プロトコル, SMB と CIFS の違いを正しく理解できていますか？(前編)」
http://www.atmarkit.co.jp/ait/articles/1501/19/news092.html (2015)
[8] デジタルアドバンテージ:「ファイル共有プロトコルSMB/CIFS (その1) (1/3)」
http://www.atmarkit.co.jp/ait/articles/0410/29/news103.html (2004)
[9] Rem system:「Windowsを利用していてWAN越しのファイル共有が遅い場合の検討事項」
http://www.rem-system.com/post-304/ (2013)
[10] Wikipedia: “Server Message Block”
https://en.wikipedia.org/wiki/Server_Message_Block (参照2016)
[11] Wikipedia: “Samba (software)”
https://en.wikipedia.org/wiki/Samba_(software) (参照2016)
[12] Wikipedia: “Windows Services for UNIX”
https://en.wikipedia.org/wiki/Windows_Services_for_UNIX (参照2016)
[13] Wikipedia: “List of file systems”
https://en.wikipedia.org/wiki/List_of_file_systems (参照2016)
[14] Ubuntu: “FuseSmb”
https://help.ubuntu.com/community/FuseSmb (参照2016)
[15] 産総研:「世界中のストレージを統合するグリッド基本ソフトウェア「Gfarm」を無償公開」
http://www.aist.go.jp/aist_j/press_release/pr2003/pr20031125/pr20031125.html (2003)
[16] oss-Tsukuba:「つくばOSS技術支援センター：Gfarmファイルシステム」
http://oss-tsukuba.org/software/gfarm (参照2016)
[17] oss-Tsukuba:「Gfarmファイルシステムをautomountする」
http://oss-tsukuba.org/tech/automount (2013)
[18] Russ Cox: “Drawterm”

https://swtch.com/drawterm/
[19] Cinap Lenrek: “DRAWTERM”
http://drawterm.9front.org/
[20] Maya Haridasan: “Cluster/Grid Computing”
http://www.cs.cornell.edu/courses/cs614/2004sp/slides/Clusters4.ppt（2004）
[21] Ian Foster and Carl Kesselman:
“The Grid: Blueprint for a New Computing Infrastructure”
Morgan Kaufmann Publishers Inc. San Francisco, CA, USA ⓒ1999
[22] Globus: “Research data management simplified”
https://www.globus.org/（参照2016）
[23] 朴泰祐：「T2K筑波システムの概要と利用プログラム計画」
http://www2.ccs.tsukuba.ac.jp/workshop/t2k-sympo2008/file/boku.pdf（2008）
[24] 合田憲人，他：「高性能分散計算環境のための認証基盤の設計」Symposium on Advanced Computing System and Infrastructures先進的計算基盤システムシンポジウムSACSIS2012（2012）
[25] Vassiliki Pouli, Yuri Demchenko, Constantinos Marinos, Diego R. Lopez, and Mary Gram- matikou:
“Chapter 9: Composable Service Architecture for Grid”（文献［26］）
[26] Nikolaos P. Preve: “Grid Computing”
“Toward a Global Interconnected Infrastructure”（Springer, 2011）
[27] Wolfgang Gentzsch: “Grid Computing Adoption in Research and Industry”
http://toolkit.globus.org/ftppub/liming/GridCompfeb03.doc（2003）
[28] Andrey Mirtchovski, Rob Simmonds and Ron Minnich: “Plan 9 — an Integrated Approach to Grid Computing”
Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International
[29] Andrey A. Mirtchovski:
“Grid Computing with Plan 9— an Alternative Solution for Grid Computing”
http://mirtchovski.com/p9/thesis.pdf（2005）
[30] Bell Labs: “Plan 9 Wiki —9grid”
http://plan9.bell-labs.com/wiki/plan9/9grid/（参照2015）
[31] Kenji Arisawa: “Plan 9 Grid Computing”
http://plan9.aichi-u.ac.jp/9grid/（2005）
http://p9.nyx.link/9grid/（2005）
[32] Kenji Arisawa:「グリッドツールキット」
http://p9.nyx.link/9grid/gtk.html（2005）
[33] Edited by Fran Berman, Geoffrey C. Fox and Anthony J. G. Hey: “Grid Computing —Making the Global Infrastructure a Reality — ”（John Wiley & Sons, 2003）
[34] Fran Berman, Geoffrey Fox and Tony Hey: “The Grid: past, present, future”（ref.［33］, pp.9-50）
[35] Ian Foster: “The Grid: A new

infrastructure for 21st century science”（ref.［33］, pp.51-65）
［36］CERN:“The DataGrid Project” http://eu-datagrid.web.cern.ch/eu-datagrid/（参照2016）
［37］EGEE:“Welcome” http://eu-egee-org.web.cern.ch/eu-egee-org/（参照2016）
［38］EGI:“HOME” http://www.egi.eu/（参照2016）
［39］EGI:“Virtual organisations” http://www.egi.eu/community/vos/（参照2016）
［40］EGI:“Wiki” https://wiki.egi.eu/wiki/Intranet/（参照2016）
［41］EGI:“Weaving The Internet Of Data” http://www.egi.eu/blog/2016/03/21/weaving_the_internet_of_data.html（参照2016）
［42］WLCG:“Welcome to the Worldwide LHC Computing Grid” http://wlcg.web.cern.ch/（参照2016）
［43］WLCG:“Welcome” http://wlcg-public.web.cern.ch/（参照2016）
［44］HTCondor:“Computing with HTCondor” https://research.cs.wisc.edu/htcondor/（参照2016）
［45］OSG:“Open Science Grid” http://www.opensciencegrid.org/（参照2016）
［46］Wikipedia:“Open Science Grid Consortium” https://en.wikipedia.org/wiki/Open_Science_Grid_Consortium/（参照2016）
［47］情報科学技術協会：2013. 9特集「e-Scienceとその周辺～現状とこれから～」http://www.infosta.or.jp/journals/201309-ja/（2013）
［48］JHPCN:「学際大規模情報基盤共同利用・共同研究拠点」http://jhpcn-kyoten.itc.u-tokyo.ac.jp/ja/（参照2016）
［49］Globus,“Research data management simplified” https://www.globus.org/
［50］Bell Labs:“Plan 9—Programmer's Manual（Volume 1）” http://plan9.bell-labs.com/sys/man/ http://plan9.bell-labs.com/sys/man/vol1.pdf
［51］Bell Labs:“Plan 9—The Documents（Volume 2）” http://plan9.bell-labs.com/sys/doc/
［52］Russ Cox, Eric Grosse, Rob Pike, Dave Presotto and Sean Quinlan:“Security in Plan 9” Proceedings of the 11th USENIX Security Symposium, 2002 https://www.usenix.org/legacy/event/sec02/cox/cox.pdf