LHDコンピュータシステムの統合化の研究

江本雅彦

博士(学術)

総合研究大学院大学
数物科学研究科
核融合科学専攻

平成14年度
(2002)

目 次

序

核融合実験装置では、制御機器や計測機器の数が多く、収集されるデータが膨大になるため、これらを管理するコンピュータシステムは実験を遂行する上で必要不可欠な要素になっている。このため、日本トーラス装置 60（JT-60)、欧州共同トーラス装置（JET）のような大型核融合実験装置では、計画当初からコンピュータシステムが組み込まれて設計、製作が行われており、各種データは収集から保存、提供等の機能が一体化した、しかもセキュリティに配慮した結果クローズドに近いシステムで通常処理されている。

核融合科学研究所の大型ヘリカル実験装置（LHD）[1] は、世界最大にして世界初のヘリカル型超伝導核融合装置であり、非常に大規模で複雑な機器構成となっている。LHD は、超伝導に必要な真空排気装置、冷却装置等に故障が生じた場合、致命的な損傷を受けることから、制御は LHD が安全にしかも長期間高い信頼度で稼動することを最優先に行われる必要があり、超伝導コイルの冷却及び励磁、プラズマ生成等を行うための中央制御システム [3] と表裏一体の LMS(LHD Man-machine interface System)[4] と呼ばれるコンピュータシステムは、当初意図的にクローズド化した独自のシステムとして構築された。また、軽微な故障等により実験が止まらぬよう、中央制御システムには重要な情報だけが送られ、他の情報は、真空排気装置、冷却装置、電源装置等が装置毎にリアルタイムモニタリングシステムを構築し、装置に最適な制御を個々に行う方式が取られた。計測機器も、精力的に R & D を行って空間分解能や時間分解能を従来より 1 桁近く改善する等、高性能化に成功したが、多くの機器が個別データ収集システムと呼ばれる独自のデータ処理システムを持つ必要性が生じることとなった。統一した形式で処理できる計測機器には、プラズマ物理データ計測システムが導入され、LHD のコンピュータシステムは、先に述べた大型核融合実験装置とは異なり、独立した複数のシステムから成る、ハードウエアー優先のシステムとして完成した。

LHD 実験開始後、この LHD 型コンピュータシステムは、当初の計画通り安全にしかも長期間高い信頼度で稼動することに成功し、多大な実験成果を上げることができた。このため、次の段階の要求として当初設定していなかった以下のようなことが、コンピュータシステムに求められることとなった。先ず、核融合科学研究所は全国共同利用研究施設であり、LHD が初期実験で成果を収めてからは

内外の共同研究数が大幅に増加し、遠隔地の共同研究者の利便性を考慮してインターネットを利用した所外共同研究者へのデータ公開と、更に一歩進んで遠隔地からの実験参加の実現が求められた。即ち、外部へのオープン化が強く求められるようになった。また、実験データを解析するには、全計測機器のデータはもちろんのこととして装置が保有する実験条件等の全ての情報を総合的に把握する必要があり、実験を効率的にしかも見落としすることなく進めるには、全情報の把握が処理端末で随時、瞬時に行えること、即ち、独立したコンピュータシステムの情報を統合化することが求められた。これら 2 点は、LHD が当初設定した方針とは、全く反対の方針になるため、LHD のコンピュータシステムは、当初の設定を維持しつつ、それとは反対の要求も満たさなくてはならないと言う難しい問題を抱え込むこととなった。統合化で特筆すべき問題は、プラズマの長時間放電である。これは LHD 計画の主要な目的の一つであるが、通常の 1 分程度以下の放電に比べて 3 時間を超えるような長時間放電を目指しているため、通常の放電とはタイムスケールが全く異なることから、当初計画では別個のコンピュータシステムで処理することが考えられていた。

本論文は、LHD を含む従来の大型核融合装置のコンピュータシステム構築構想を持ってしては満たすことのできない上記要求、即ち、多数のクローズドシステムから成る大型核融合装置用コンピュータシステムの情報の統合化、オープン化の研究を行い、これを成就することができたことから、成果をまとめたものである。

第一部で従来システムの概要を述べるとともに、上記の要求を満たす際の問題点について議論する。第二部および第三部では、統合化の研究を述べるとともに、開発を行ったシステムについて説明を行う。第四部では、遠隔実験参加のシステムについて議論を行う。

尚、本論文で扱ったような多様なシステムが混在するという問題は LHD 計画に限らず、今後国際協力が進む中で、規模の巨大化、計画の長期化、参加組織の増大に伴い自然に起こりうる問題であり、本論文はこのような環境下での具体的な解決策の例を提示するものである。

第I部

核融合実験における
データ処理システム

第1章　核融合科学研究所における データ処理システム

この章では、核融合科学研究所の LHD 実験に使用されていた各種コンピュータシステムの概要と、その問題点を説明する。

1.1　従来システムの概要

核融合科学研究所の大型ヘリカル実験装置によるプラズマ実験が始まった当時、研究所には大別して 4 種類のデータ処理システムが稼動していた。

それらはそれぞれ、

- 中央制御システムと LMS
- LABCOM システム
- リアルタイムモニタリングシステム
- 個別収集系

である。

LHD 実験開始当初のこれらのシステム間の関係を図 1.1 に示す。LMS は実験シーケンスとよばれる、TTL ベースの信号を個々のシステムに対して配信しており、各収集システムでは、このシーケンスに従いデータ収集等の同期を行っている。

各システムが収集した実験データは、個々のプラズマ放電に割り当てられたショット番号によってタグ付けされ保存されているが、このショット番号の管理もまた、LMS の役割である。しかしながら、LMS システムとは直接アクセスを行うことができるシステムは、LABCOM システム等一部に限られているため、それ以外の個別収集サーバのために、ショット番号サーバを設置し、ショット番号を他のシステムから参照できるようにした。

これには MS-Windows の共有フォルダを使用する方法が用いられ、ファイルにショット番号と実験シーケンスを記述することによって行っている。クライアント

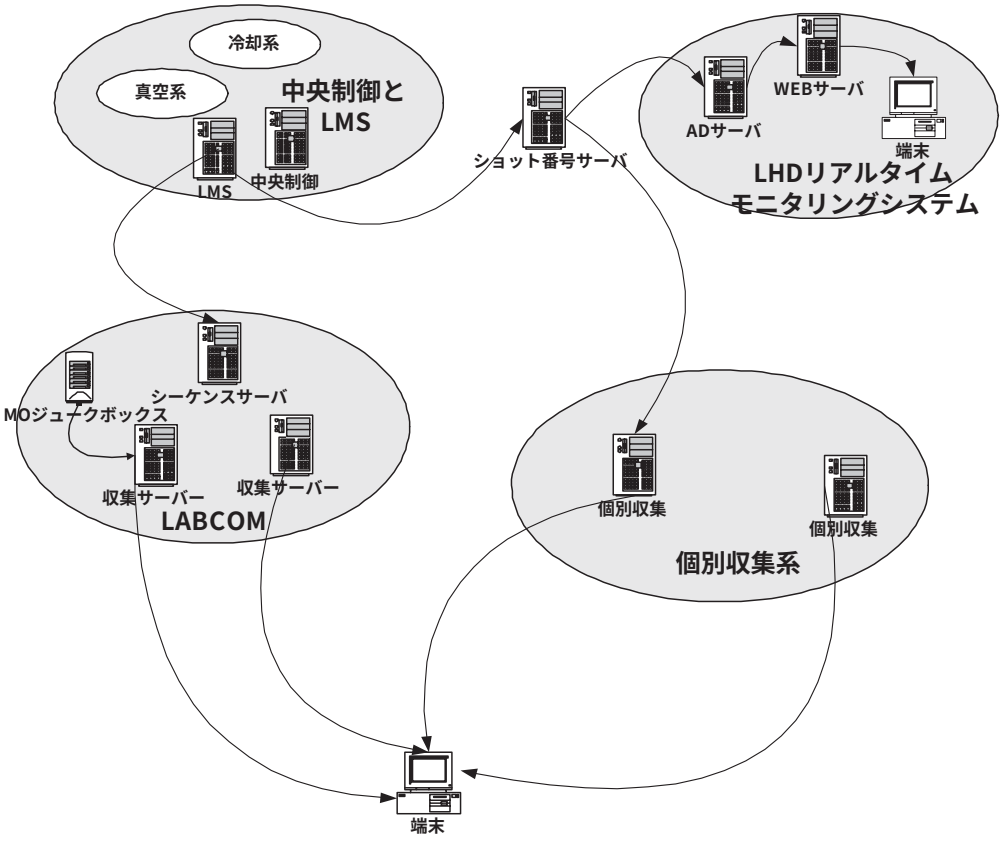

図 1.1: LHD コンピュータシステム概要

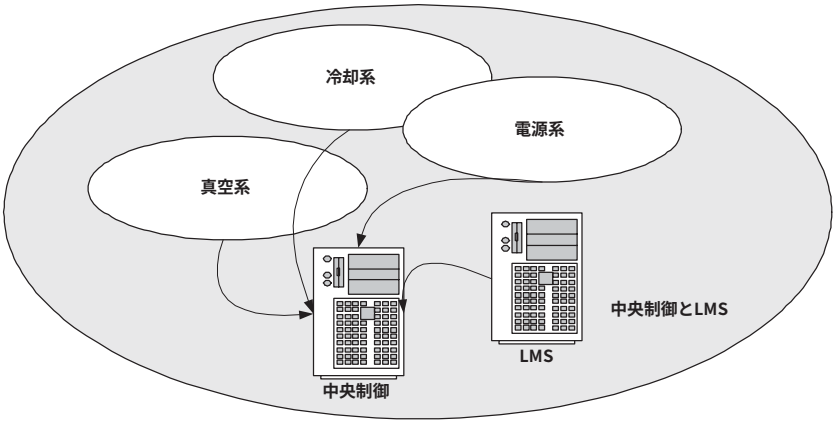

図 1.2: 中央制御システムと LMS

が最新のショット番号を取得するには、定期的にこのファイルを読み取り、ショット番号が更新されたことを認識するという方法が取られていた。

また、LHD リアルタイムモニタリングシステムも他のネットワークからは分離されて設置されており、このシステムにアクセスできる端末は一部に限られている。

収集された実験データは LABCOM システム他、個々の収集システム上に保存され、データを参照する時はこれらのシステムに直接アクセスする必要があった。

以降では、実験開始当初の各システムについて具体的に見て行く。

1.1.1 中央制御システムと LMS

LHD のシステムは本体である LHD とそれを取り囲む真空装置、冷却装置の他、NBI、ECH といった加熱装置や HIBP、FIR 等の計測装置等、30 種類以上のサブシステムから構成される。中央制御システム [3] はこれらサブシステムの制御を行うシステムであり、UNIX ワークステーション、VME マルチプロセッサ、および リアルタイム OS である VxWorks 等を使った複数のコンピュータ群から構成される。中央制御システムはサブシステムとの連携し、PLC(Programmable Logic Controller) 等を用い、光ファイバやリレーで信号で、各実験装置の監視・コントロールを行っている。中央制御システムそのものは、LHD 装置の安定運用のために、独立した設計になっており、自動制御で動作することが可能である。一方で、中央制御システムのサブシステムの中に、LMS(LHDMan-machine interface System)[4] と呼ばれるシステムがあり、GUI によるマン・マシンインタフェイスを提供している。LMS は他のサブシステムとは、TCP/IP を利用したネットワークを介して接続している。ただし、ハードウェアロジックによる制御の方が、LMS を介した

制御よりも優先度が高く設定されている。

また、LMS はさらに以下のような役割がある。

- 実験シーケンスおよび実験状態管理
- 実験装置のセットアップおよび監視

実験シーケンスおよび実験状態は、LMS から TTL の信号として各収集システム等に送信されるもので、この信号に収集タイミング等を同期させることで異なるシステム間でイベントの起こった時刻の同期が可能となる。また、LMS は LHD に取り付けられた 3,000 におよぶセンサからのデータを監視し、機器のセットアップを行っている。これらの役割を行うために LMS は Windows NT で動作する 2 台の主・副サーバシステムで構成され、上記実験条件の記録・設定の為にリレーショナルデーターベース (RDB) である Oracle 7 を使用している。LAN 上のクライアント PC には VisualBasic により作成されたアプリケーションがインストールされ、データベースサーバと SQL*Net [1]により通信を行っている。

中央制御システムと LMS は、装置および実験運用を最優先するように設計されている。このため、他のシステムからは切り離された形で運用されており、LMS システムと直接接続できるシステムは一部に限られている。

1.1.2 LABCOM システム

LABCOM システム [5] はプラズマの計測データ収集を行うネットワーク分散型のバッチ処理システムである。このシステムは 30 余台からなる MS Windows NT ベースの収集サーバにより構成され、一台毎に異なる計測種類のデータ収集を行っている。これらのデータ収集サーバはさらに一台のシーケンスサーバによって管理されている。LMS より送られてきたシーケンシャルシグナルをシーケンスサーバが受信すると、各々の収集サーバに対しデータ収集の準備開始や待機等の命令を送信する。これらのサーバはデータ処理室に設置されているが、データ収集サーバは光 SCSI エクステンダを介しシールドルーム内に設置された CAMAC コントローラと接続されている。CAMAC コントローラ上のタイミングモジュールが LMS システムからの信号を受けると、これをトリガに A/D を開始する。A/D ボード上のメモリが一杯になるとデータをサーバ機に転送する。データは収集サーバ上のオブジェクトデータベース、O2 にオブジェクトとして格納される。O2 のデータベースイメージファイルは収集サーバに接続された 50GB の RAID [2]ディスク上に

[1]Oracle 社が提供するデータベースサーバとクライアント間の接続に用いられるミドルウェア

[2]Redundant Arrays of Independent (Inexpensive) Disks。複数のディスクにアクセスを分散させ、ひとつのディスクに見せる技術

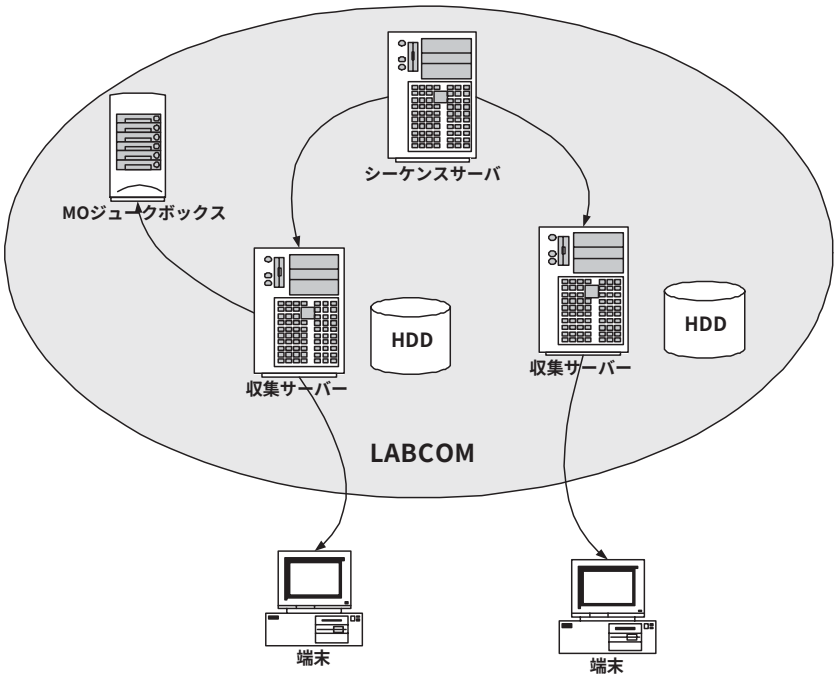

図 1.3: LABCOM

あり、新しいデータはここに格納されているが、ハードディスクの容量に限りがあるため古いデータについてはデータベースイメージファイルをオフライン化し、MO ジュークボックスに移動し保存していた。このため、古いデータを参照したい場合にはデータベースイメージを MO ジュークボックスから手動で HDD 上にコピーし、オンライン化することによって行っていた。

1.1.3 LHD リアルタイムモニタリングシステム

LHD リアルタイムモニタリングシステム [7] は、LHD コイルの安定性のモニタリングのために開発されたシステムで、真空容器に接続された温度センサやひずみセンサ等で測定された信号をリアルタイムで収集・解析を行っている。これらのデータは Web ブラウザを利用してリアルタイムでのグラフ表示が可能である。また、異常があった場合、電子メールや携帯電話に通知を行うことができる。このシステムは Solaris2.6 で動作する Sun Ultra2 をサーバとし、RDB を用いている点、および、クライアントのみならずサーバの大部分が Java により作成されていることが特長である。このシステムについては、7 章で詳しく説明する。

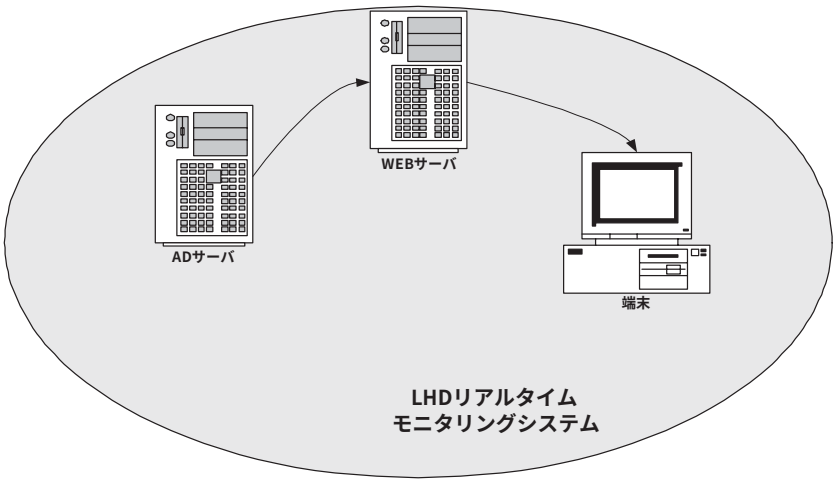

図 1.4: LHD リアルタイムモニタリングシステム

1.1.4　個別収集系

この他に上記 3 つのシステムに含まれないデータ収集システムが多数稼動している。これらは、LABCOM システムが提供している汎用のデジタイザでは対応できず、個別のデジタイザを使用しているためである。各システムは独自に設計運用されているため、個々のシステム毎使用しているプログラムが異なり、OS も MS Windows 以外にも Solaris 等の UNIX が存在する。これらの収集システムをまとめて個別収集系と呼ぶ。

1.1.5　実験の流れ

核融合科学研究所で主に行われているのは、〜10 秒程度の短パルス実験であり、放電実験毎、各システム間では以下のような事が行われている。

1. LMS システムにより、次回の放電の設定情報が中央制御システムにセットされる。
2. 一日の放電実験は、放電開始 2 分半前、放電開始、終了 ··· 等の 10 の状態からなる一つのシーケンスとして繰り返し運転を行う。
3. 中央制御システムは、各状態の区切り毎に TTL レベルの信号 (シーケンスシグナルと呼ばれている) を LABCOM システムやその他個別収集系に送信する。

4. シグナルを受けた各システムは、そのタイミングに応じデータの取り込み準備の開始、データ収集の開始等を実行する。

5. LABCOM システムはデータ収集が終了するとそのデータをオブジェクトデータベースに格納し、次のショットに備える。

6. 一部の個別収集系は収集したデータを元に解析を始める。

1.1.6 新たな要求の発生

一巡の LHD 実験を行った結果、従来のシステムを利用して、初期の目的を遂げることができた。しかし、異なったグループ間で独立に測定されたデータの相互参照によって、より詳細なデータ解析を行う必要性が叫ばれるようになった。また、所外からは核融合研は日本における核融合研究のナショナルセンターであり、国内外の多数の研究者との共同研究がますます盛んに行われるようになってきたため、外部の研究者が遠隔地から容易に実験データにアクセスしたいとの要求が高まってきた。ところが、これらの要求に対処するためには、下記の挙げる点が問題であることが明らかになった。

データ転送速度

LABCOM システムでは、データの収集にオブジェクトデータベース O2 を使用している。このデータベースは、クライアント側の CPU を利用してデータの検索が行われており、この機能によって負荷を分散させることができる。こうした検索を分散させて行う方法は高速なネットワーク上で複雑なデータ構造のデータを扱う場合には効果を発揮するが、必ずしも十分なネットワーク帯域を確保することができない遠隔地からのインターネットを利用したアクセスでは、CPU よりもネットワークがボトルネックとなり十分な速度が得られない。

データ保存場所の分散

LABCOM システムで収集したデータは各収集サーバのローカルディスクに保存されるが、ローカルディスクの容量は 50 GB 弱しかなく、データ量の多い計測では一日あたり、3 〜 4 GB のデータが書き込まれるため、数週間分しか保存することができない。このため、古いデータはオブジェクトデータベースをオフラインにし、データベースのイメージ全体を MO ジュークボックスに移動しており、古いデータが必要となった場合は、イメージをハードディスクから手動で移動しオンライン化するという作業を行っていた。

ファイルの場所の特定

物理データファイルは個々のグループが管理しているため、必要なファイルを見つけ出すことが困難である。また、ファイルの管理が個々のグループに任されているため、データのバックアップ等の作業もグループの負担となる。

多様なデータフォーマットへの対応

データフォーマットがグループ毎に異なるため、データ作成者以外の研究者が利用するためには、そのフォーマット毎にプログラムを作成する必要があった。さらに、トムソン散乱による電子温度測定と FIR による電子密度測定の時間変異を同時に参照するといった、複数の計測を同時に参照したいという事は頻繁に発生するが、この場合は計測種別毎のデータの読出しプログラムを作成する必要がある。

サポートプラットフォーム

外部の研究所からの利用を考えた場合、MS Windows 以外にも UNIX や Macintosh 等の多様な OS のサポートが必要となる。ところが、実験データを MS Windows のファイル共有でファイルを公開するという関係上、このファイルへのアクセスは MS Windows 以外のプラットフォームからアクセスすることが難しく、多くの場合一旦 MS Windows を経由する必要がある。

ショット番号の特定

ショット番号は、放電実験を特定する上で最も基本となる情報である。例えば、実験データはショット番号毎に作成されているため、研究者が自分の必要なデータを特定するためにはそのショット番号を知る必要がある。ショット番号は LMS がデータベースで管理しているが、このシステムは高いセキュリティレベルのネットワークに置かれているため、LABCOM システム等一部のシステムからしかアクセスすることができない。このため、個別収集系等のためにショット番号と現在の実験シーケンスの状態を記述したファイルを MS Windows 共有フォルダに置くことで、ショット番号の参照を可能とした。しかしながら、この方法を用いた場合、サーバによって、ファイルの内容が変更されたことをクライアントが直接知ることができないため、各クライアントは常にポーリングにより内容の変更を確認する必要があり、ネットワークおよび、サーバに負荷がかかる。また、サーバがデータを書き込んでいる最中にクライアントがファイルを読もうとすると、ファイルのロックにより読み込みエラーが発生し、プログラムが異常終了してしまうという事態も発生した。更に MS Windows を利用している PC でなければこのサービスを受けられない問題もある。

実験条件の取得

あるショットがどのような実験条件で行われたかという情報は、上記 LMS システムにより記録されていたが、このデータは前述のように外部からの利用が困難であり、実際には紙ベースの実験ノートに頼る必要があった。

これらの問題を解決するためには、システムの統一化が必要であり、遠隔地から利用する行う場合においても、統一化されたシステムが必要となることから、システムの統一化を優先して研究を進めることにした。

第2章　核融合科学研究所以外の主なシステムとその特徴

本研究所で生じたこのような問題の対処のため、他の研究所では問題が生じていないのか、あるいはそれをどのように対処しているのかの調査を行った。

2.1　MDSplus

現在、核融合プラズマ実験の現場でもっとも広く使われているシステムは、Massachusetts 工科大学の Stillerman らが開発した MDSplus[8] である。MDSplus は開発元である Massachusetts 工科大学やイタリア CNR の他、米国 General Atomic 社の DIII-D、米国 Princeton 大学の NSTX、オーストラリア ANU の H-1 といったシステムで稼動している。MDSplus は元々DEC VMS 上で開発されたが、現在では UNIX や MS Windows への移植が進んでいる。同システムはツリー型をした階層構造でデータを管理するのが特徴であり、一種のデータベースとしての働きを持つ。ツリー構造で管理することにより、ある情報に関連したデータを枝としてまとめて扱うことができる。また、MDSplus が提供するデータ型は文字列、バイト、整数、単精度浮動小数 … という基本的な型にとどまらず、次のような特別なデータ型を有する。

- SIGNAL:データ列
- WITH_UNITS:データの単位名
- WITH_ERRORS:エラーバー
- RANGE:スタート、エンド、増分
- ACTION、TASK、DISPATCH:データ収集、データ解析などのステータス
- EXPRESSION:TCL による「式」

このような柔軟な型を利用することにより、生データや物理データの他、計測に使用した計測機器のセットアップ情報、校正値情報、収集タスクのスケジューリング等を管理することが可能となっている。これにより、MDSplus はデータ収集システムという側面と、データ管理ツールとしての側面をもつ。データ収集を行う場合は、ディスパッチャーと呼ばれるプロセスにより起動されたタスクがツリーの中のセットアップ情報を元に収集を行う。収集された生データはツリーに格納され、自動的に解析処理を行った後、解析データをツリーに格納する。複数のツリーは物理的には一つのファイルに格納されており、複数のショットをまとめて管理することが可能である。このツリー構造のデータにアクセスを行う為のツールとして、TCL (Tree Control Language) というインタプリタ言語を提供されている。また、GUI ツールとして Traverser がある。このツールは X Window System ベースであり、ツリー状に展開されたデータを GUI によりアクセス、編集を行うことを可能としている。また、可視化ツールである IDL によるインタフェイスの他、C、C++、FOTRAN、Java といった通常の言語からのアクセスが可能となっており、ユーザは自身のプログラムの中から、簡単にツリーにアクセスすることができる。またシステムの一部として X Window System ベースの Scope という 2 次元可視化ツールが提供されており、ユーザはこのツールを利用して収集・解析されたデータを簡単に見ることができる。

2.2　各研究所におけるデータ処理システムの現状

2.2.1　Alcator C-Mod (Massachusetts 工科大学プラズマ核融合センタ、米)[9]

Massachusetts 工科大学プラズマ核融合センタでは、OpenVMS ベースの Alapha-Server および VAXServer 等、複数の収集サーバが使用され、CAMAC を用いたデータ収集を行っている。収集されたデータは収集サーバに接続された RAID ディスクに保存され、放電終了後直ちにデータ解析サーバ群（OpenVMS ベースの VAXStation および、AlphaStation）上で動作している MDSplus システムや IDL のプログラムによって、解析・可視化が行われる。

収集・解析されたデータは、RAID セット、ハードディスク、オプティカルジュークボックスの三階層により管理されている。一日の収集が終わると RAID ディスク上のデータはハードディスクおよびジュークボックスに移される。ハードディスクの空き領域が少なくなると、参照頻度の低いデータはオプティカルジュークボックス上に移動される。

実験の結果や、放電中のコメントなどは Electric Logbook に記録され、必要な

データを探すのに役立てている。これにはRDBであるMS SQL Serverを利用しており、IDLから利用できるようにインタフェイスが提供されている。

2.2.2 DIII-D (General Atomic社、米)[11]

General Atomic社では1998年よりMDSplusシステムをデータストレージの管理のために採用した。MDSplus採用以前の状況は核融合科学研究所が経験した状況に酷似している。

MDSplus採用以前は60種類以上の様々な計測機器が存在し、計測機器毎に異なったファイルフォーマットを採用していたため、研究者がそのデータを利用するためにはファイルの構造を知る必要があり、また、データが様々なコンピュータ上に散在していたために保存場所を検索する必要があったため、研究者に負担がかかっていた。MDSplusを採用することで、統一的な取り出し方法でデータを取出すことが可能となり、またデータの保存場所を探す必要もなく、すばやく必要なデータを取り出すことが可能となった。

MDSplusサーバはOSにCompaq Tru64 Unixを用いたAlphaServer上で稼動しており、21種の解析済みデータが保存・提供されている。データ解析は放電サイクル中にMDSplusのディスパッチ機能により自動的に行われる。データの解析には、ロードバランシングされたUNIXクラスタと、Linuxのマルチ CPUクラスタで行っている。UNIXクラスタは一般的な解析用途に用いられ、これはHP 9000 Model T600一台とロードバランシングされた、6台のHPのUNIX Workstationおよび5台のDigital Unix Workstationから成っている。一方、Linuxマルチ CPUクラスタはマルチプロセッサ用に修正されたEFITプログラムにより、平衡データを作成するのに使用される。

さらに、遠隔実験参加支援のため、MDSplusのもつクライアント・サーバインタフェイスを利用し、生データおよび解析済みデータの利用を可能としている。また、可視化ツールとして、ReviewPlusが用いられている。このツールはIDLによって書かれたアプリケーションであり、MDSplusのIDLインタフェイスを利用し、MDSplusシステムにアクセスすることができ、ユーザはMDSplusサーバのアドレスを指定することにより遠隔地のデータを簡単に取り出すことが可能となっている。

2.2.3 JET (JET共同事業体、英)[12]

JETで使用されているデータコントロールシステムはCODAS (COntrol and Data Acuision System)という分散型システムによって運用されているのが特徴

的である。このシステムは、TCP/IP のネットワーク上に分散する 150 台以上の Solaris 上で動作し、70,000 以上の計測シグナルのコントロールや取得を行っている。分散されたシステムは、１)JET 全体のシステムの統括、2) サブシステムのコントロール 3) ローカルユニット、コンポーネントという 3 層構造をなしており、ソフトウェアの再利用が容易になるようになっている。この構造により、システムの大規模な変更にも柔軟に対応することができる。CODAS システムによって収集されたデータは IBM のメインフレームに転送され、解析および永久保管される。ここでは、CODAS から得られた生データやそれを解析したデータ等が統一されたフォーマットでハードディスクおよびテープに保存されている。データは階層型ストレージシステムにより管理され、参照頻度の高いデータはディスク上に、低いデータはテープへと転送される。

2.2.4　JT-60(日本原子力研究所)[13]

JT-60 のデータ収集システムは中央にショット間処理サーバと呼ばれる大型汎用計算機 (富士通 GS8300) を配した構造をなし、このサーバが各サブシステムと連携をとり、データの収集や実験シーケンスを制御している。これらのサブシステムには 1)CAMAC システム、2) 高速 VME データ収集システム (FDS)、3) 大容量データ記憶システム (TMDS) 等の収集システムの他、4) リアルタイム処理システム、5) データベースサーバ等がある。

放電終了後、収集されたデータは直ちにショット間処理サーバに送られ、バッチ処理により解析が行われる。ショット間処理サーバによって処理されたデータはその後、データベースサーバに送られ、ネットワークデータベースアクセスシステム（NDBS）に保存される。

2.2.5　Tore Supra(Cadarache 研究所、仏)[15]

Tore Supra のデータ処理システムはネットワークに分散する複数のシステムからなっており、データ収集は IRMX III 上の Multibus ベースのシステムと、リアルタイム OS である LynxOS 上で動作する VME バスシステムから構成される。また、この他に PC ベースのシステムがある。

データシーケンスや実験条件の設定、データ解析、データ保存等はそれぞれ複数の UNIX サーバ (AlphaServer) が担当しており、各収集システムとは、商用パッケージソフトである、RTworks を用いて通信を行い制御している。実験に関する情報は RDB、CA-Ingres で管理されており、各収集システムは放電開始前にデータベースからデータをダウンロードし、条件設定を行う。収集システムにより収

集されたデータはリアルタイムでデータストリームとして中央ストレージに送られる。一方 PC ベースのシステムは放電終了までデータをローカルに保存しておき、放電終了後データを中央ストレージに送信する。放電終了後、データ処理サーバ上の解析プロセスが起動され、中央ストレージに送られたデータの解析が行われる。

2.2.6 検討

本研究所と比較し、他研究所ではデータの解析プロセスが一元化されている。特に Alcator C-Mod の例では、データの収集・解析・保存・提供という一連の流れが MDSplus という一つのシステムの中で構築されていることが特筆される。

ところが、システム全体を一元化することにより、ファイルの統一や保存・管理が容易になるという利点がある一方で、システムの置き換えが困難となる。コンピュータの進歩や新しい計測技術の導入により、新しいシステムの導入が必要となるが、これらに柔軟に対応するためには、各システム間が「疎」に結合している方が好ましい。

さらに、実際の物理量に変換する作業を完全に自動化することは困難である点が問題点として挙げられる。これは各ショットにおける測定機器の校正値や誤差、無効データの有無など計測担当者でなければ処理できない事項が多いためである。本研究所でも、LABCOM システムで一次解析と呼ばれる、A/D 変換出力を物理量に変換する機能の組み込みがその後行われたが、これは A/D 変換器からの出力値を変換表等により 1 対 1 で物理量に変換するところまでで、さらに高次な物理量への変換作業にはやはり研究者の手による必要がある。

この様な点から、解析を一括に行うのではなく、現状のシステムを活かしたままで、解析済みデータの共有・検索が簡単に行えるシステムの開発を行うことが望ましい。

第II部

統合化 1
～ データ取得法の統一 ～

データの統一化についての方法と手段

第 I 部で紹介したように、当時 (平成 10 年) の分散したコンピュータシステムでは、新たに発生した要求に応えることができず、統合化の必要が迫られた。

筆者はこの問題に対処するため、システムの統合化の研究を行った。

ここでいう統合化とは、

- データ取得法の統一
- 実験に関する情報の共有化

の二つである。

また、統一化を進めるにあたっては以下の点が重要となる。

- 既存のシステムの変更を最小限にする
- オープン技術を利用し、移植性や拡張性を高める。

前者は、LHD 実験を運用を停止せずに統合化を進めてゆく必要があり、既存システムの変更による LHD 実験への影響を避けるため、重要な点である。後者は、今後のシステムの拡張や他の実験設備への移植を行うために必要な事項であり、このためには、商用アプリケーションを利用せず、オープンソースウェアの利用や汎用の技術を用いる必要がある。

二種類の統合化の内、第 II 部では、まずデータ取得法の統一化について取り扱い、情報の共有化は第 III 部で扱う。

具体的なデータ取得方法の統一手法としては、既存システムのデータ取得法に共通のデータ取得方法を設けることにより、既存のシステムに影響を与えずに統一化する方法を考案した (図 II.a)。左側の旧来の方法では、クライアントはシステム毎にデータ取得法を実装しなければならず、新しいシステムが登場すると、そのシステムに対応するためには、クライアントの変更が必要となる。一方、右側のようにクライアント・サーバ間に共通層を設けることで、サーバの違いを吸収させることができる。この場合、クライアントは一種類の取得法のみを実装するだけで複数のシステムに対応可能であり、また、クライアント側の変更なしに、新しいシステムに対応することができる。

より具体的には、この共通層を用いた統合化には方法は段階的な手法を用い、下記のような三段階での統合化を行った。

1. 保存形式によらないデータ取得方法の提供
2. データフォーマットの統一

この統一化の様子を図示したのが、図 II.b である。

最初に行ったのは、中期保存システムによる LABCOM データの保存形式に依存しない統一的なデータアクセス方法の提供である。LABCOM システムでは、収集したデータをオブジェクトデータベース内にオブジェクトとして保存している一方、古いデータはオブジェクトデータベースから削除し、MO ジュークボックスに移動し管理しており、それぞれデータの取得方法が異なっていた。中期保存システムでは、データのインデックス化とデータの保存場所によらない共通のデータ取得方法を提供し、ユーザは実験データが何処に格納されているのかを意識せず、参照することが可能となった。

次に行ったのは、解析済みデータサーバによる LHD 実験の収集・解析データの統一である。このシステムは中期保存システムの考え方を他のデータに適用したもので、中期保存システムで用いたインデックス化とデータ取得方法の他にテキストファイルを用いたデータ形式の統一化を行った。これにより、異なったシステムで収集・解析を行ったデータを統一的に取り扱うことが可能となった。

最後に汎用可視化ツールである NIFScope を開発し、これによってより広い範囲の統一化を行った。NIFScope はデータの読み込み部分を汎用言語を用いてモジュール化したことが特長であり、この機能を用いて既存の LHD 実験データのみならず、新しく登場したデータフォーマットにも、対応するモジュールを用意するだけで統一的にデータの可視化を行うことが可能となった。

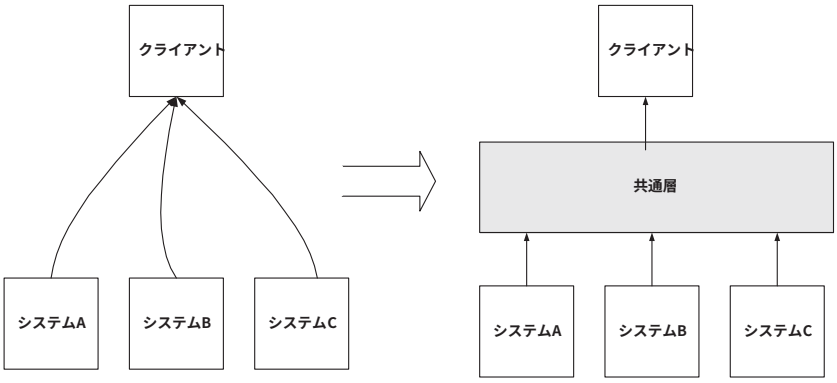

図 II.a: 共通層を用いた統一

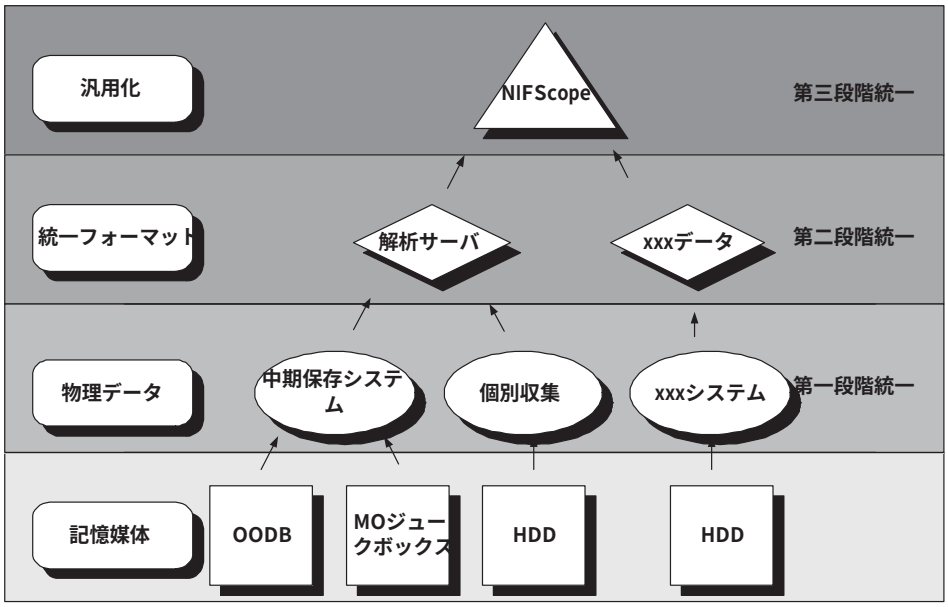

図 II.b: 三段階による統一

第3章　中期保存システム

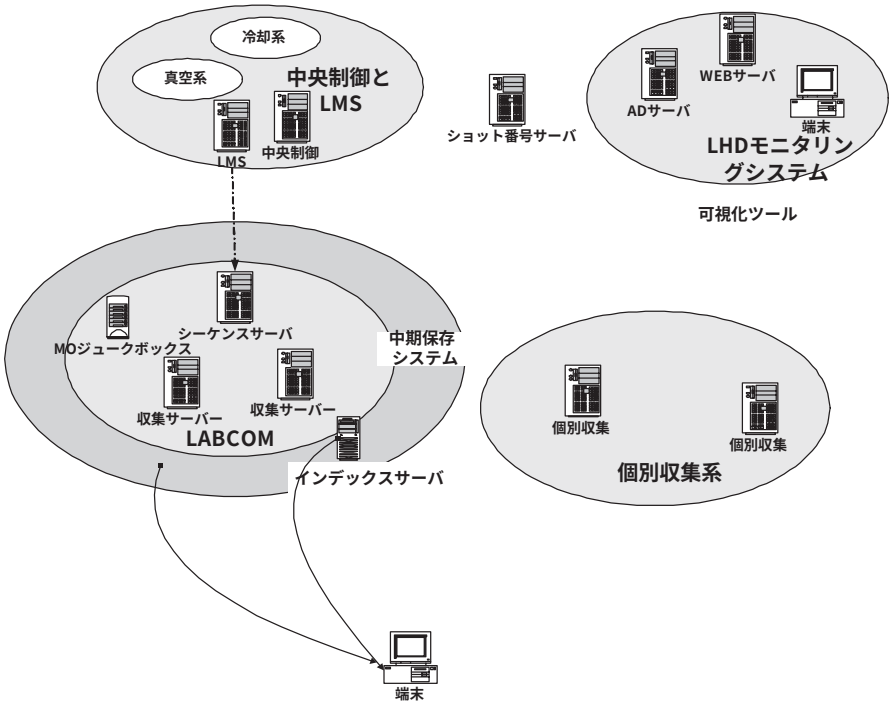

図 3.1: 統合化第一段階

実験開始当時 LABCOM システムは MS Windows NT ベースのオブジェクト指向型データベース O2 [1]を利用し、データの収集に利用していた。O2 は各収集サーバで動いており、収集サーバには 50GB 弱のハードディスクが接続されていたが、一日に収集されるデータは多いもので数 GB になるため、全てのデータをハードディスク上に置くことができず、古いデータはデータベースをオフライン化し、データベースイメージそのものを MO ジュークボックス (1.2TB × 3 台) に保存していた。

データを参照する場合、クライアントはどのサーバにデータが保存されている

[1]LHD 実験開始当時。現在では ObjectStore に置き換わっている

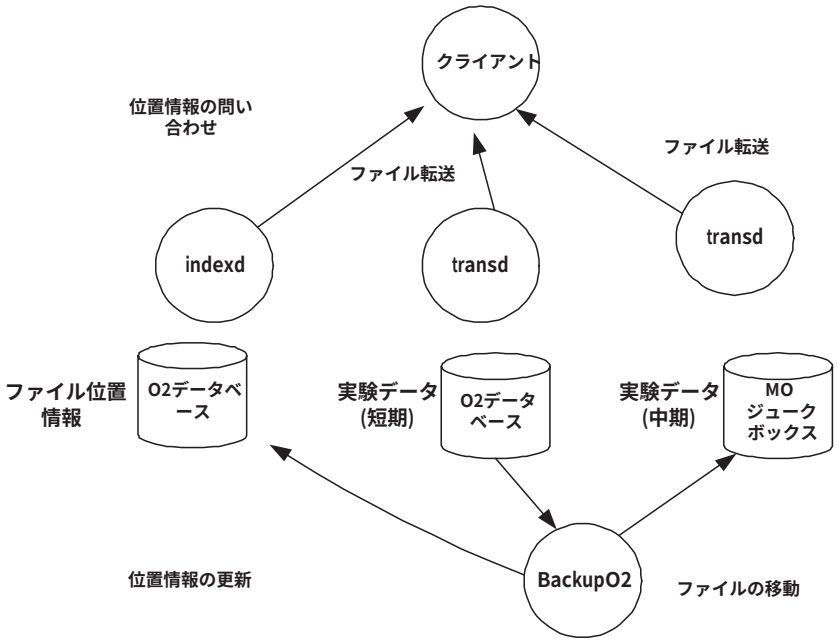

図 3.2: 中期保存システムデータフローダイアグラム

かをあらかじめ知る必要があり、古いデータが MO ジュークボックスに移動していた場合は、MO からデータベースイメージを HDD にコピーし、それをオンライン化することによって参照する必要があった。これらの作業は手動で行われていたため、実際にデータを参照するには数分〜数十分の作業を要していた。

以上のような問題点を解決するため、筆者らは中期保存システムを構築した。

3.1　システム概要

図 3.2 にこのシステムの概略を示す。

本システムでは、インデックスサーバを新たに設置しデータ保存場所の管理を行った (図 3.3)。データの保存場所の情報は O2 を用いたインデックスサーバが管理しており、ここには、どのサーバにどのような形態で保存しているかという情報が記録されている。各収集サーバはデータ収集完了後インデックスサーバに、収集されたデータがオブジェクトデータベース上のオブジェクトとして保存されていることを伝え、インデックスを更新する。

収集されたデータは、実験中は短期的な（数日〜数週間) 保存場所としてオブジェクトデータベース上に保存されているが、夜間バッチでオブジェクトデータベース上のオブジェクトはファイルに変換されて、中期的な (〜数年) 保存場所と

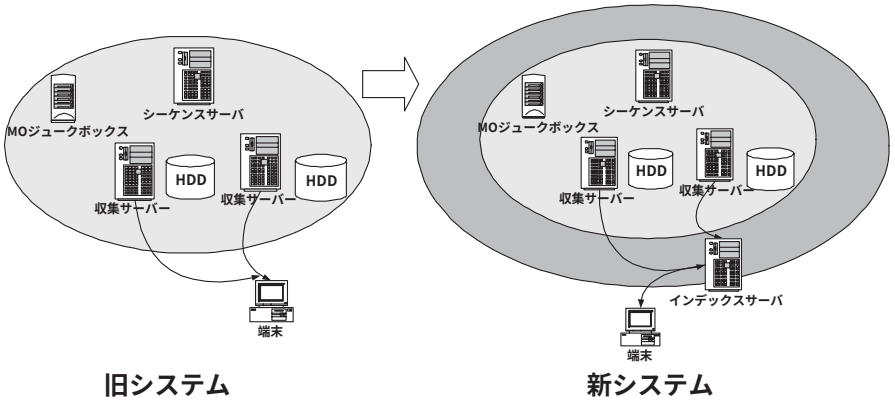

図 3.3: インデックスサーバ

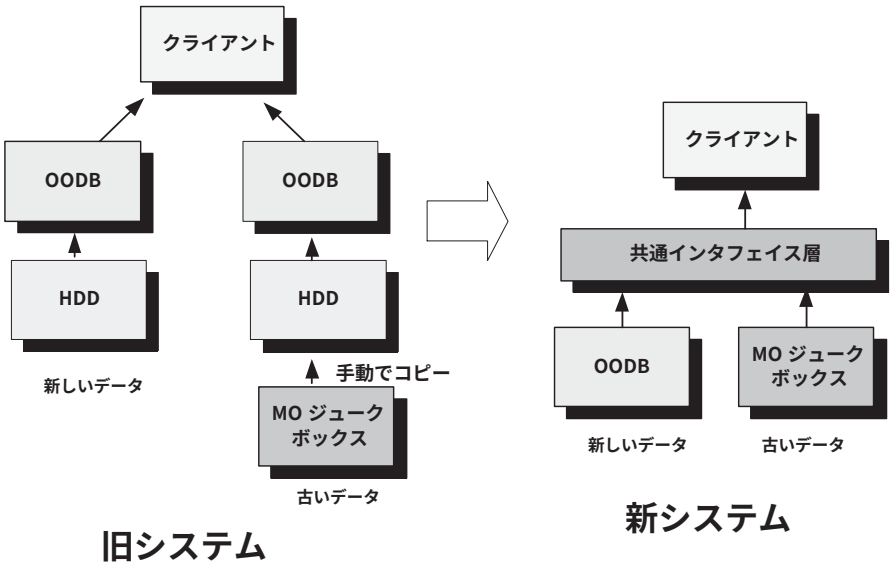

図 3.4: データ取り出し方法の一元化

して MO ジュークボックスに保存される。この時インデックスサーバ上の情報を更新し、収集データがファイルとして MO ジュークボックスに移動したことを知らせる。実験直後はオブジェクトデータベースと MO ジュークボックスの両方に存在しているが、古いデータ (オブジェクト) は定期的にオブジェクトデータベースから削除される。

クライアントプログラムでデータを取得する場合、プログラムはまずインデックスサーバで保存先を検索する。次に保存先サーバの転送プログラムに接続しデータの取得を行う。ここでファイル転送に使用しているプロトコルは、データの保存形態がファイルであるのか、データベースオブジェクトであるのかによらず同一である。

このような仕組みにより、ユーザは保存場所は保存形態を意識することなく統一的な方法によりデータのアクセスを行うことが可能となった。また、サーバ・クライアント間でデータの検索・転送に用いているプロトコルは独自に開発したものであり、特定の OS やアプリケーションに依存しないことから、従来の方法では、不可能であった MS Windows 以外への OS (Linux や Solaris 等) に移植することが可能となった。

また、O2 はネットワーク分散型のオブジェクト指向型データベースで、データの検索機能は主としてクライアント側で実行される。この機能は、CPU の負荷を分散させるという目的には適しているが、サーバ・クライアント間に流れるデータ量は多くなり、インターネットのような必ずしも十分な帯域を確保することができないネットワークで使用した場合には、ネットワーク帯域がボトルネックとなる。

中期保存システムでは O2 を利用したデータの検索・取り出し作業はサーバ内部だけで行われるため、O2 をクライアントから直接使用した場合に比べ、サーバからクライアントへのデータ通信量を抑えることができた。

この結果、10 Mbps の LAN で 200 秒程度かかっていた処理を 30 秒以下に短縮することが可能となった。

第4章　解析済みデータサーバ

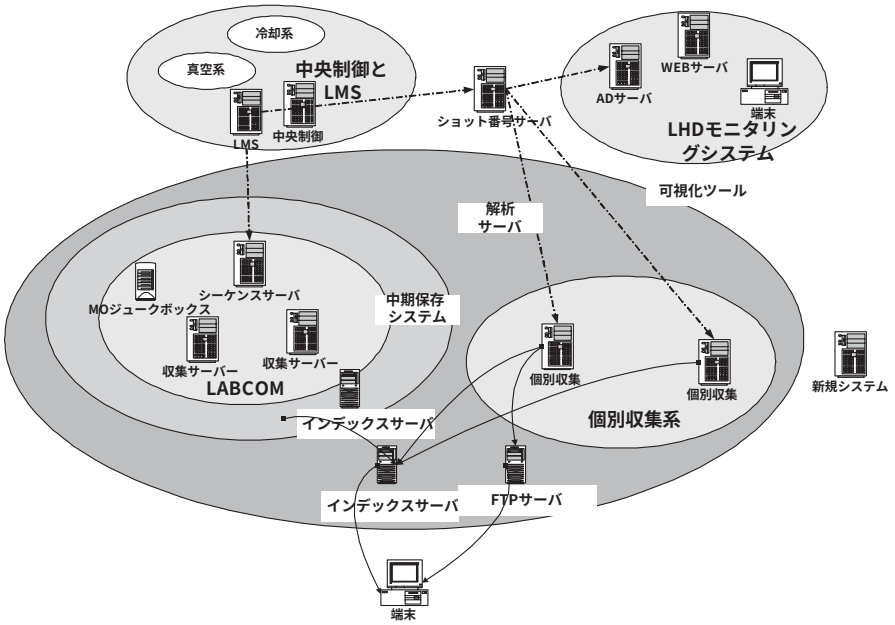

図 4.1: 統合化第二段階

統合化の第二段階として、物理データの統一を行う解析済みデータサーバシステムの開発を行った。

核融合科学研究所では、プラズマ物理現象の研究のため様々な種類の計測器が稼動している。これらの計測器で収集されたデータは、別々の研究グループで解析が行われる。一般により精度の高い解析のためには、種々の計測システムのデータを付き合わせる必要がある。ところが、使用されるコンピュータのプラットフォームや解析したデータの保存法、データフォーマット等がグループ毎に異なるため、他グループが収集したデータを参照することは困難であった。

中期保存システムのもつ「保存場所や保存法に依存しない統一的なデータの取り出し方法の提供」が、LABCOM システムで収集されたデータだけでなく、一般のデータについても有用であると考え、このシステムを元に解析済みデータサー

バシステムの開発を行った。このシステムは、保存場所のインデックス化を行う一台のデータベースサーバとデータを提供する複数台の FTP サーバからなる (図 4.1)。

解析済みデータサーバシステムは、

1. 共通のデータフォーマット (解析済みデータフォーマット）
2. 統一されたデータ取出し手法

を提供するもので、複数のシステムにより収集・解析を行ったデータ (解析済みデータ) を容易に比較できるようにするものである。

4.1　システム概要

システムの概要を図 4.2 に、データフローを図 4.3 に示す。また、サーバの仕様を表 4.1 に示した。各グループにより収集された計測名、ショット番号、保存先等の情報は、データベースに登録され、FTP によりファイルサーバに転送される。反対にデータを取出す場合は、対象となるデータをデータベースに問い合わせ、保存先を取得し、FTP で保存先からクライアント PC またはワークステーションへの転送を行う。

解析済みデータが新たに登録・変更されると、データベースのトリガにより、解析済みデータの変更があったことをマルチキャストパケットで通知する。これにより、このマルチキャストパケットを受信するクライアントは、データの変更があったことが即座に分かり、解析済みデータのグラフの自動更新を行うことで、常に最新データを表示することが可能である (8 章 参照)

また、データベース、FTP サーバにはそれぞれ予備機を用意し、夜間バッチにより内容をコピーしていており、障害時には少なくとも前日の時点まではの状態を速やかに回復することが出来る。

クライアントプログラムとして、プログラム内部から呼び出すことが出来るように、下記のような MS Windows および UNIX 用のコマンドラインのツールを提供している。

- isearch 登録されたデータの検索
- igetfile 登録されたデータの取り出し
- ibackup 解析済みデータをデータベースに登録し、データを FTP サーバに転送する

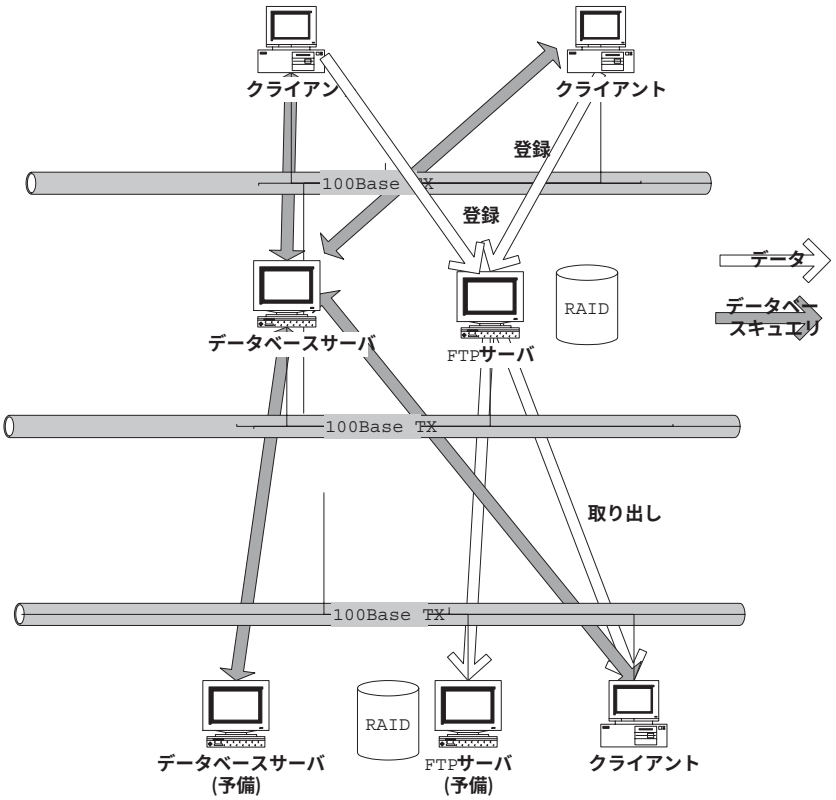

図 4.2: 解析情報サーバシステム概要

	FTP サーバ	DB サーバ
CPU	Pentium III 850MHz × 2	Athlon MP 2000+ × 2
メモリ	1024MB	1024MB
OS	LASER 5 Linux 7.2	LASER5 Linux 7.2
FTP	wu-ftpd 2.6	
RDB		PostgreSQL 7.2.1

表 4.1: サーバの仕様

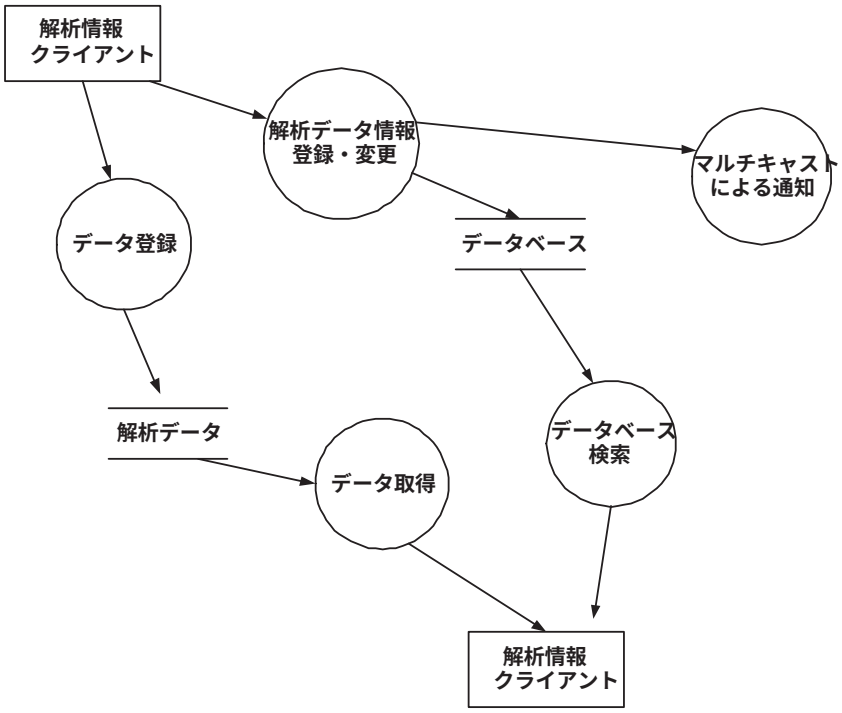

図 4.3: 解析済みデータサーバシステムのデータフローダイアグラム

- ilint 解析済みデータのフォーマットのチェックを行う

また、GUI により操作できるツールを用意した (図 4.4)。このツールは登録されているデータを 1,000 行ごとにまとめたツリー表示を行うことで、必要なデータにすぐにたどり着くように考慮されている。また、可視化ツールである PV-Wave を呼び出すことにより、簡易グラフ表示を行うことが可能となっている他、登録された外部プログラムに選択した解析データを渡すことができる。

データのツリー表示のために集計テーブルを利用しているが、この集計テーブルの作成にはトリガを利用している。データベースに新しいデータが登録、あるいは削除されるとトリガが起動され、各計測の 1,000 ショット毎に登録数の集計を行う。

PostgreSQL のトリガからは C 言語等のプログラムを起動することが出来るため、この機能を利用し、データ変更の通知をクライアントに通知するためにも使用している。変更された内容はマルチキャストによって各クライアントに通知される。マルチキャストは、そのパケットを希望するクライアントとそのクライアントがあるネットワークにしか送信しないので、サーバやネットワークに負荷をかけることなく、非同期にクライアントに情報を伝えることができる。データの登録通知を受け取ったクライアントはこの情報を元に最新のデータを取得することができ、グラフの自動更新に役立てることが可能である。5 章で紹介する NIFScope のバッチ処理では、この機能を利用している。

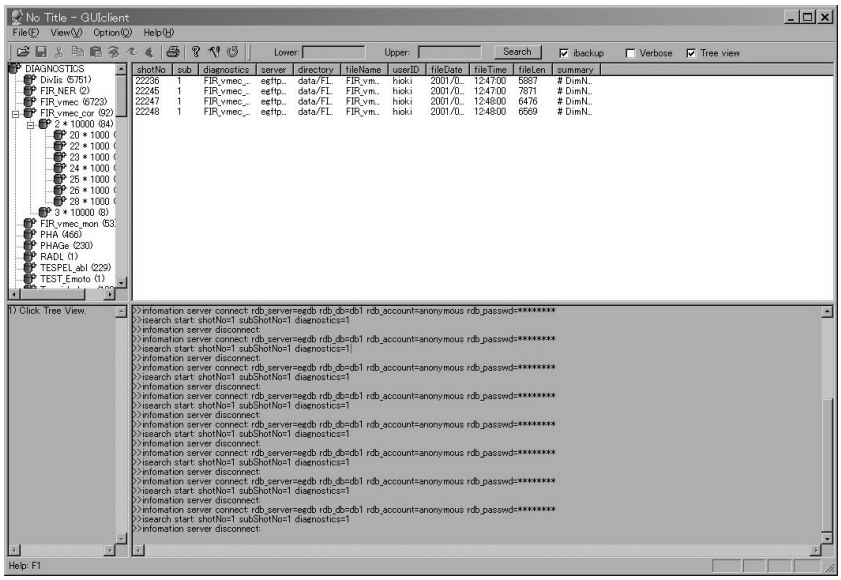

図 4.4: 解析済みデータクライアント (MS Windows)

4.2　共通フォーマット

一つのプログラムで複数の種類の解析済みデータを容易に読み取ることが出来るようにするためには、データの統一が必要である。解析済みデータのフォーマットを決定するに際し、以下のことが考慮された。

- 市販のスプレッドシートやグラフツールでの取り込みが可能であり、FORTRAN、C、IDL 等、種々のプログラム言語から容易に読み込むことが出来ること
- なるべく多くの計測データに対応できる汎用性を備えること。

4.2.1　既存フォーマット

科学、工学分野で使用されているデータフォーマットとしては表 4.2 に挙げるようなものがある。

また、科学技術分野に限らないフォーマットとしては、XML(eXtensible Markup Language) がある。XML は SGML(Standard Generalized Markup Language) を容易に使えるようにしたマークアップ言語[1]で、異なるデータベースやアプリケーション間でのデータ交換に等に使用されている。XML を使った実験データ等の交換も研究されており、これには、NASA 天文データセンタ (ADC) の開発した XDF [21] 等がある。

LHD 実験のデータをこれらの既存フォーマットを利用することが考えられたが、バイナリファイル形式の場合、データの入出力には専用のライブラリを必要とし、またテキストファイルの場合でも、文法が複雑であり専用のライブラリを使用しなければ、データの読み取りや作成の時の文法チェックが困難である。

また、LHD 実験のデータは主として時系列データであり、これを既存のテキストフォーマットで記述すると冗長になりすぎるという理由から、既存フォーマットの使用を断念した。

4.2.2　解析済みデータフォーマット

前節に述べた理由から、既存フォーマットを使用することはできないと判断し、新たにデータフォーマットを定義した (付録 A)。

データフォーマットは、カンマ区切りのテキストファイル形式を用いることにした。これは、市販の表計算ソフトやユーザが開発した FORTRAN や IDL 等の

[1] TEX、HTML 等、文章中に属性情報や組版情報等をコマンドとして埋め込む言語。

フォーマット名	分野	形式
CDF (Common Data Format) [16]	汎用	バイナリ
CXF (Chemical eXchange Format)	化学	テキスト
FITS (Flexible Image Transport System) [17]	天文	テキスト (ヘッダ部) ＋バイナリ
JCAMP-DX (Joint Commitee on Atomic and Molecular Physical Data eXcange Standards) [18]	原子分子物理	テキスト
NetCDF (Network Common Data Format) [19]	汎用	バイナリ
PDS (Planetary Data System)[20]	天文	バイナリ

表 4.2: フォーマット

プログラムからデータの読み取りが簡単に行えるようにするためである。

テキストファイルは UNIX と MS-DOS(または MS Windows) で改行コードが異なる。一般に UNIX では ASCII コード 0x0A であるのに対し、MS-DOS では、0x0D 0x0A となる。ところが、解析に使用されているコンピュータは UNIX と MS Windows があるため、あえて改行コードの特定はしなかった。改行コードの自動判別は研究所で使用されている多くの市販ツールでサポートされており、また、プログラム上はどちらの改行コードも空白と同等に扱われるため問題ないものと判断した。

ショット番号や、データサイズなどの付加情報はファイルの前半部分に集め、データそのものとは分離した。これによって、スプレッドシート等のアプリケーションからデータを読み込む場合は、最初の数行を読み飛ばすように指定することで、簡単にデータを取り込むことが出来るためである。

多くの計測データに対応できるという目的を達成するため、計測データの一般化を試みた。

既存の全ての計測は、任意パラメタ $(x_1、x_2、\ldots x_n)$ における物理量 $f_1 = f_1(x_1、x_2、\ldots x_n)$ 、$f_2 = f_2(x_1、x_2、\ldots x_n)$ ••• を取得するというすることで一般化できる。例えば、トムソン散乱による計測等、温度プロファイルの時系列を測定では、

$$T = T(t, r)$$

を測定することになる。統一フォーマットでは、この計測データを

$$\begin{array}{cccccccc} x_1, & x_2, & \ldots & x_n, & f_1, & f_2, & \ldots & f_m \\ x'_1, & x'_2, & \ldots & x'_n, & f'_1, & f'_2, & \ldots & f'_m \\ x''_1, & x''_2, & \ldots & x''_n, & f''_1, & f''_2, & \ldots & f''_m \\ \ldots, & \ldots, & \ldots, & \ldots, & \ldots, & \ldots, & \ldots, & \ldots \end{array}$$

という形式に並べる。実際の実験データとしては時系列 $(x_1 = t)$ のものが多く、これらの測定時刻 t は等間隔で固定であることが多い。例えば、A/D 変換器による取り込みであれば、

$$t_i = t_0 + i\Delta t$$

となるのが普通である。そのため、計測時刻を知るには、開始時刻 t_0 とサンプリングレート $1/\Delta t$ を記録するだけで事足り、この方が収納効率が良い。しかしながら、より汎用性を重視するという観点から、現在のような形になった。またデータの並びについては特に定めていない。

4.3　解析済みデータサーバの特長

解析済みデータサーバシステムの特長は FTP や RDB といった標準的な技術を採用した点である。従来は中期保存システムをベースにし、上記目的を達成することが考えられていたため、当初 LABCOM システムが採用していた下記のような技術を用いていた。

- データベースとしてオブジェクトデータベースである O2 を採用[2]
- ファイル転送には独自プロトコルを使用
- ファイルのフォーマットとして zlib を利用した独自フォーマットを使用

しかしながら、次節に述べる理由により、本システムにはふさわしくないと考えられ、次のものに置き換えた。

- データベースとして RDB である PostgreSQL を採用
- ファイル転送には FTP を使用
- ファイルフォーマットとして ZIP を使用

これらはいずれも標準的あるいは、より広く使われる技術であり、Linux や Window 等の異なる OS 上、また C++や Java 等の多様な言語での開発が容易である。これにより、MS Windows の GUI を使ったプログラムの他、MS Windows や UNIX 上でのコマンドライン版のアプリケーション、CGI(Common Gateway Interface)[3]、Java アプレット[4] 等、様々な形態のアプリケーションを提供が可能となった。

[2]その後 LABCOM システムはオブジェクトデータベース ObjectStore と RDB の PostgreSQL を組み合わせたものに置き換えられた

[3]Web サーバから外部プログラムを呼び出すための仕組み

[4]Web ブラウザ内部で実行される Java のアプリケーション

4.3.1 リレーショナルデータベース (RDB) の利用

本システムのデータベースはオープンソースである PostgreSQL を採用している。このシステムは当初 MS Windows NT で動作するオブジェクト指向データベース [22] の一つである O2 を使用していた。オブジェクトデータベースの RDB に対するアドバンテージとして下記のようなものが挙げられる。

- データベース上のデータをメモリ上のオブジェクトと同様に取り扱うことができる。
- ユーザ定義の自由な型を使用することができ、アプリケーション上のデータ表現からデータベース上のデータ表現への変換作業が必要ない。
- 構造をもつデータを扱うことができ、複雑なリレーションを表現することが容易である。
- 通常の RDB では扱うことのできないサイズの大きなデータを格納することができる。

C++ではインスタンス変数を生成する時には new オペレータが使用される。インスタンスは実行時のメモリ内だけに存在する一時的なオブジェクトである。オブジェクトデータベースの C++のインタフェイスでは、new オペレータをオーバーロード [5] することにより、同じ new オペレータを用いてハードディスク上に永続可能 (persistent) [6]なオブジェクトの生成を可能としている。永続可能な new オペレータで生成されたオブジェクトは永続型ポインタにより操作でき、これは通常の new オペレータで生成されたメモリ上のオブジェクトを扱うのと同じ方法によってハードディスク上の永続可能オブジェクトを操作することができることを意味している。

RDB の場合、メモリ上のオブジェクトをデータベースに格納するためには、オブジェクトを int、float、char 型等、RDB の持つ基本的な型に展開し、SQL (Structured Query Language) を用いて格納しなければならない。元々RDB は、事務手続きに対する要求から発達してきたものであることから、単純な型しかサポートしておらず、実験で使用されるような巨大なバイナリデータを直接収納できるような型を持つものは少ない。また、これらの巨大バイナリをサポートしているものであっても、これらの型は特別扱いとなり、普通の型のデータに出来るような操作を行うことが出来ないことが多い。

[5]同名のメソッドを再定義すること

[6]オブジェクトデータベースで用いられる用語。通常のオブジェクトがプロセスが生きている間でしか有効でない、一時的 (transient) なオブジェクトであることへの対比として用いられる

また、オブジェクトデータベースの場合、オブジェクト間のリレーションは、永続可能なポインタのリンクとして表現される。これはメモリ上のオブジェクト間のリレーションを通常のポインタのリンクにより表現しているのと同じであり、オブジェクトの検索を行うのにデータベース用の特別な手続きを必要とせず、通常のメモリ上のオブジェクトと同様の操作で永続可能なオブジェクトの検索や変更を行うことが可能である。

一方、RDB の場合、データベース上のデータを操作する場合、SQL という特別な言語を必要とする。このため、プログラムから利用する場合は、同時に二つの言語を利用する必要があるため、開発やメンテナンス上の負担が発生する。

以上に挙げた、オブジェクトデータベースのデータベース上のオブジェクトをメモリ上のオブジェクトと同様に取り扱うことができるという特長は、開発者にとって、プログラマの教育や開発期間の短縮、保守の容易さ等のメリットをもたらす。

一方、RDB と比較し下記のような欠点も挙げられる。

1. RDB に比べ単純な検索では検索速度で劣る事が多い

不定サイズのオブジェクトを扱うというオブジェクトデータベースの構造上、固定長を基本とした RDB に比べて検索速度で劣る。さらに、RDB はオブジェクトデータベースに比べ、歴史も長く、ベンダー間の競争があるため、データの検索速度等のオプティマイズはオブジェクトデータベースよりも発達している。

2. 標準化が遅れており、ベンダー毎に操作が大きく異なる

オブジェクトデータベースの標準化には、ODMG (Object Data Management Group) という団体が行っていたが、強制力はあまりなく、実際にはプロダクト毎にデータベースの操作方法が大きく異なっていた。このため、あるデータベース用に開発したプログラムを別のデータベース用に移植すのにはプログラムの大幅な変更が必要であった。これは、現在販売あるいはオープンソースとして配布されている RDB が、ANSI による SQL92 という規格に準拠し、ほぼその操作が統一されているのと対照的である。

3. マーケットが小さいため、長期的な製品のサポートに不安がある

オブジェクトデータベースは RDB に比べ市場規模が小さいために、製品の販売中止という可能性が高く、実際 LABCOM システムで O2 から ObjectStore に変更された理由の一つは O2 の単体販売中止にある。

4. RDB に比べ、サポートするプラットフォームが少ない

マーケットが小さく、製品数が少ないこともあり、サポートしているOSの数が少ない。また、基本的にはオブジェクト指向型言語によるアクセスを想定しているため、C++やJava等の言語からのインタフェイスしか用意されていないことが多い。RDBであれば、データベースにアクセスを行うためのAPIと言語そのものとは分離されているため、ライブラリを呼び出すことが出来れば、容易に多様な言語から使用することが可能である。

5. データの管理には専用のツールやプログラムを使う必要がある。

RDBの場合はデータに関するほとんどの操作をSQLを使って行うことができる。ところが、オブジェクトデータベースの場合は、データベース上のオブジェクトの内容を変更したり、削除するためにはプログラムを作成したり、専用ツールから行わなければならない。このためメンテナンス性で大きく劣る。

さらに、O2固有の問題点として次のようなものが存在した。

6. データの検索等のデータ操作はクライアント側で行われる

RDBの場合、通常、検索・更新等の操作はサーバ側で完結しているため、クライアントはその結果を受け取るだけである。このため、更新操作であればサーバ・クライアント間で送受信されるデータは極わずかである。

一方、多くの商用オブジェクトデータベースでは、処理の分散化が行われており、この機能で RDB との差別化を行っている。このような分散型オブジェクトデータベースでは、データの検索・更新等に必要なリンク操作は複数のサーバ、あるいはクライアント側のCPUを利用して行われており、O2 も検索等の作業はクライアント CPU を利用して行われる。

分散化は、オブジェクト間のリレーションが複雑で、作業のボトルネックがネットワークよりも CPU にある場合には有効であるが、十分なネットワーク帯域が得られないネットワーク上では、必要とするデータ以外にもリンク操作に必要なデータがクライアント側に送信されるため、ネットワークの方がボトルネックとなる。

特に今回の格納データの保存情報という単純なデータでは、分散化の利点を活かすことができない。

7. 検索エンジンがマルチスレッド化されていない

検索エンジンがマルチスレッド化 [7]されていないため、1つのクライアントプロセスが検索を行っている間は他のプロセスは待ちとなり、多数のクライアント

[7]スレッドはプログラムの実行単位。マルチスレッド化されたプログラムは一つのプロセスの中で同時に複数のスレッドを実行することができるため、複数の作業を同時に行うことができる。

が同時にアクセスする本システムでは高いパフォーマンスが得られない。

オブジェクトデータベースは動画や音声、画像等のマルチメディアデータを扱う場合や、CAD のように非常に多くの部品が複雑に組み合わされているという関係を記述する場合には、プログラム上のデータ構造からデータベースのスキーマへ容易に変換することができるため、開発・設計において威力を発揮する (図 4.5)。また、実験物理への適用例としては、高エネルギー物理実験への利用がある。例えば、CERN における LHC (Large Hadron Collider) 計画では、同実験で生成される 5PB/年もの大量のデータを保存するために、オブジェクトデータベースの Objectivity を利用する計画が行われている。これは加速器実験で一つの衝突イベントを起源として、カスケード的に発生する複数のイベントやキャリブレーションデータなどを関連付け、これらのデータをまとめて取り扱うためにオブジェクトデータベースを利用している。

しかし、本件のような解析済みデータの情報管理という単純なデータ構造で良い場合には、その利点が活かせず、逆にパフォーマンスの点で RDB に比して不利である。今後のユーザ数の増加を踏まえ、RDB の採用に踏み切った。RDB の選択に当たってはコスト削減のため、フリーの PC-UNIX の一つである Linux 上で動作するオープンソースまたはフリーのデータベースを採用することにし、下記のような候補が挙がった。

- Sybase Adaptive Server 11.0.3 (フリー)
- MySQL(オープンソース)
- PostgreSQL(オープンソース)

Sybase Adaptive Server は商用 UNIX 用にパッケージ販売されていたデータベースの無料配布版の製品である。このため、この中では最も高機能であり、実績をもつ。ところが、これは販売促進用のもので、バイナリ配布しかされず、Sybase 社からのサポートは期待出来ない。また、当時 Linux の Kernel は 2.2 へと移行し始めていたが、Adaptive Server は 2.0 しか正式サポートしていなかった。

MySQL [23] は MySQL AB により開発・メンテナンスが行われているオープンソースのデータベースである。MySQL の持つ特徴的な機能として以下のものが挙げられる。

- マルチスレッド対応
- 強力なテキスト検索機能
- 高速な検索機能

一方、当時の MySQL はトランザクションをサポートしないという欠点があった。

データのアップデートが複数のテーブルにまたがるとき、それら複数のテーブルのアップデートを矛盾なく行うために、アップデート操作全体をアトミックな操作すなわち非可分な操作とする。例えば、テーブル A の情報を元にテーブル B を更新するというプロセスがあり、このプロセス実行中に別のプロセスがテーブル B を更新すると矛盾が発生する。この場合、両テーブルの更新操作を一つの操作として行うことにより、他のプロセスが割り込むことが出来なくなる。トランザクションは通常の商用 RDB であれば、必ず備えている機能である。

さらに、SQL92 のコンパチビリティが弱く、サポートしない機能が多かった。

PostgreSQL は Calfornia 大学 Berkeley 校で開発された Postgres を元に開発されたデータベースである。大学における実験的なデータベースとして開発を行われたことから、他のデータベースにはない次のようなユニークな機能を有する。

- クラス、継承等オブジェクト指向をサポート
- 配列、地理型等多様な型を持つ

また、他のオープンソースのデータベースに比べ、

- SQL92 の互換性が高い
- トランザクションのサポート

等の長所が挙げられる。

今後のサポート、トランザクションのサポートの点から PostgreSQL を最終的に採用した。

4.3.2 FTP プロトコルの採用

当初データベースに登録したデータの転送にはソケットを使用した独自プロトコルを採用していた。これはデータやシステムの特性にカスタマイズされたシステムを構築出来ると言うメリットがあるが、反面プログラムが複雑になる。プログラムの複雑さはバグの混入を招き、また複数の人間で開発・メンテナンスを行うことが困難となる。今後のことを考慮し、よく使われるプロトコル (well known) の採用を行った。これには、HTTP、NFS 、SMB [8] 等が候補に挙がったが、不正アクセスから防御するためユーザ認証が必要なこと、UNIX や MS Windows で容易に扱えることなどを考慮し、FTP を採用した。

[8] Server Message Block。MS-Network 等でファイルサービスを実現するためのプロトコル

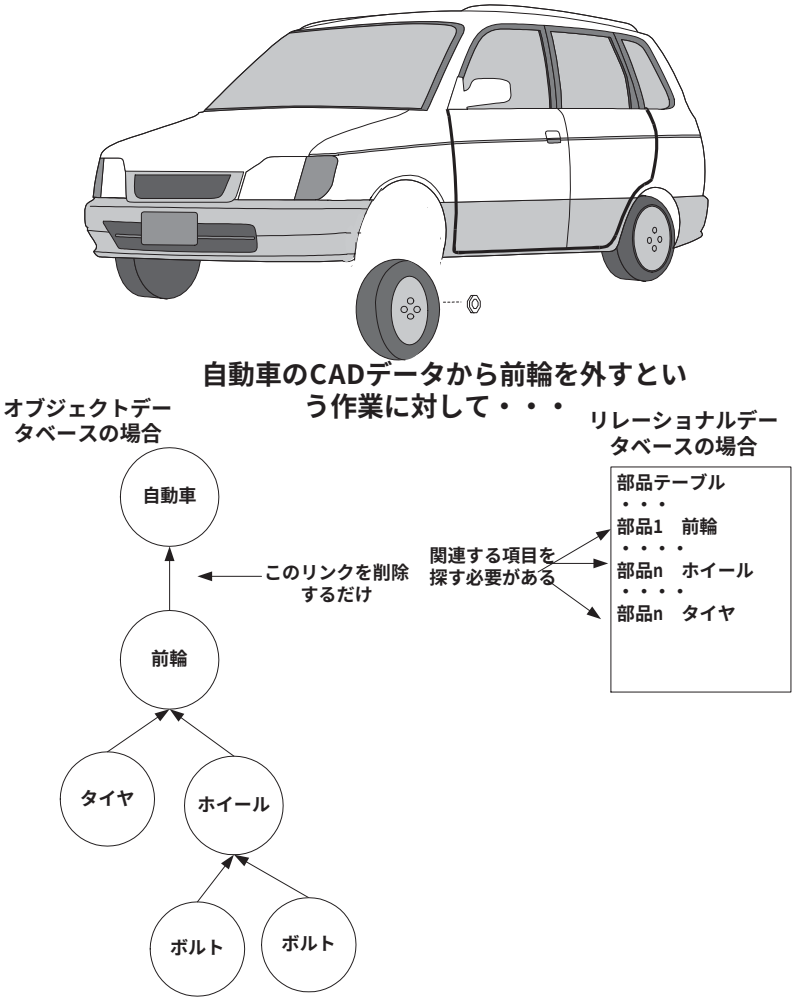

図 4.5: 両データベースを CAD に使用した場合

通常のユーザの便宜を図るため、匿名ユーザでの登録も許可しているが、個々のユーザアカウントを用いれば、そのアカウントでファイルを作成され、他のユーザから改変・削除を不可能にすることができる。一方、外部公開用のサーバでは、匿名ユーザでのログインを禁止することにより、登録利用者以外からの利用を出来ないようにしている。

FTP サーバはネットワーク上に複数の設置することが出来る。例えば、必要とするファイルはグループ毎に異なるため、グループ毎にファイルサーバを用意することで、ファイルのアクセスが一台に集中しすることを避けることが出来る。これらの FTP サーバは UNIX や MS Windows 等で標準で備わっているものが使用できるため、既存の資源を利用することが可能である。また、FTP プロトコルはファイルの転送用としては広く使用されているため、クライアント側も C++や Java のクラスライブラリや Perl のモジュール等を利用し、様々なプラットフォームから利用することが可能である。

4.3.3 ZIP による圧縮

解析済みデータは平成 14 年 10 月現在、84 計測 59 万ファイルが登録されている。ディスク容量の節約のため、これらのファイルは圧縮されていることが望ましい。また、データ圧縮を行うことでネットワークの負荷を低減することが出来る。解析済みデータはテキストファイルであり、“0.0000” といった同じ文字列の繰り返しが多数存在する。そのため、ZIP や lha 等の圧縮ツールが採用している辞書を用いた圧縮方式は非常に効果的である。

圧縮したファイルのフォーマットとして当初採用していた zlib を元にした独自フォーマットからパーソナルコンピュータで普及している ZIP 形式に変更した。ZIP で使用される圧縮アルゴリズムは基本的には zlib と同じものであるため、圧縮率に関しては大差ない。しかしながら、汎用フォーマットである ZIP 形式を採用することで、WinZIP 等のユーティリティを使いユーザが簡単に展開できるようになった他、ライブラリ等が提供されており、Java や Perl 等から圧縮されたファイルを操作することが可能となった。

ZIP による圧縮の結果、圧縮後のファイルサイズは元のサイズに比べ平均 0.12 倍に縮小することができた。(表 4.3)。

一方、同じデータを 8 バイト (倍精度) のバイナリを用いて格納した場合、ファイルサイズはテキストファイルに対して 0.59 倍と小さくなったが、これを ZIP を用いて圧縮した場合は 0.13 倍となった。また、4 バイト (単精度) を用いて格納したファイルを圧縮した場合は 0.11 倍となった。このように、バイナリを圧縮した場合でも、テキストファイルを圧縮した場合とほぼ同程度のサイズとなったが、こ

フォーマット	圧縮前	圧縮後
テキスト	1	0.12
4 バイト浮動小数	0.29	0.11
8 バイト浮動小数	0.59	0.13

表 4.3: サイズの比較 (圧縮前のテキストファイルを 1 とする)

れはテキスト、バイナリの表現に関わらず、情報量は変化しないためだと考えられる。

4.3.4　両システムの比較

表 4.4 が旧システムと現システムでデータの登録等を行った時の結果である。また、テストに使用したシステムの仕様を表 4.7 に示す。データベースサーバとクライアントは共通の 100BaseTX のシェアード HUB に接続され、FTP サーバは別のサブネットに設置した。クライアントと FTP サーバ間を繋ぐネットワークで一番遅い箇所は 10Base-T を用いている。またテストに使用したファイルのサイズは 45KB(圧縮時) である。

この表から現システムの方が検索・削除で 3〜7 倍高速になっていることが分かる。一方、登録ではかえって遅くなっているが、これは検索・削除ではデータベース上の情報だけを更新しているのに対し、データ登録ではデータベースの更新の他にファイルの転送が行われているためである。現システムではファイル転送に FTP プロトコルを使用しているため、認証等の手続きにより遅くなっているものと推測される。このことから、純粋にデータベースの検索速度では旧システムを上回っていることが分かる。クライアント数が複数になった場合には、O2 では資源のロックが起こり他のプロセスは待たされるため、この差は大きくなるものと思われる。

ところが、RDB を使用した場合でもデータ量の増加につれ遅くなることが考えられる。LHD 実験で最終的に保存されるデータは 1,000 万件程度と考えられ、このときのレスポンスを見積もる必要がある。これには次のようにして行った。

1. 当時 (平成 12 年)、31 万件のデータ (ショット番号 $< 26,000$) が登録されていたが、このデータのショット番号に 100,000 を加えたデータを作成し、元の行にコピーする。これにより元データの 2 倍のデータが作成される。

2. 1 で作成したデータのショット番号に 200,000 を加えたデータを作成し、1 のデータに加える。これにより元データの 4 倍のデータが作成される

	旧	現
登録	1.24	1.61
検索	0.74	0.11
取得	0.74	0.72
削除	0.74	0.24

表 4.4: 新・旧システムの比較 (単位:秒)

	旧システム	現システム	クライアント
CPU	Pentium III 800MHz	←	Pentium III 800MHz
メモリ	256 MB	←	192MB
OS	MS Windows NT4.0	LASER5 Linux 6.4	MS Windows NT 4.0
その他	O2 5.0.2 P25	PostgreSQL 6.5.2	

表 4.5: テストに使用した PC の仕様

3. 64 倍になるまで続けて行き、igetfile コマンドによるデータの取出し時間の測定を行った (表 4.6)

この表から取出し時間はデータベースに登録されたデータ数に殆ど影響を受けず、2,000 万件の場合でも、ファイル取出を含めて 0.4 秒程度で終了することが分かる。また、データ量の増加に対して SQL の検索時間は全く影響を受けていないことが分かる。これはデータベースの検索には 2 分木 (binary tree)[9]によるインデックスが使用されており、2 分木検索ではデータ量 N に対し、検索時間は $log_2 N$ に比例するため、データ量の増加にあまり影響を受けない。また、SQL の実行時間は、検索そのものの時間よりむしろ SQL 文のコンパイルや検索のストラテジ決定に要する時間の方が大きいものと推測される。尚、テストに使用した PC は表 4.7 の通りである。

ファイルの取得に要する時間の内、大部分の時間は FTP サーバへのログイン手続きに費やされている。特にこの FTP サーバは認証手続きに NIS [10]を利用しており、ネットワークの混雑に対する影響を大きく受ける可能性がある。この試験では、サーバとクライアントを 100BaseTX の同一 HUB 上に設置し行った。

[9]ソートされたデータ列から目的のデータを大小比較によって探索してゆく方法

[10]Network Information Service。UNIX で多く用いられるネットワークを利用した分散型データ管理システム

	ファイル 取出時間 (ms)				
行数	DB login	SQL 実行	FTP login	ファイル 転送	合計
311,926	30	17	311	85	444
623,852	30	16	306	66	420
1,247,704	30	16	357	91	496
2,495,408	30	18	316	76	440
4,990,816	32	18	316	76	443
9,981,632	31	18	300	66	416
19,9263,264	31	15	307	76	430

表 4.6: データ量と取り出し時間

	データベースサーバ	FTP サーバ	クライアント
CPU	Pentium III 800MHz Single	Pentium III 850MHz Dual	Pentium II 800MHz
メモリ	512MB	1024MB	192MB
OS	LASER5 Linux 6.0	Laser5 Linux 6.0	MS Windows NT 4.0
その他	PostgreSQL 7.1		

表 4.7: テストに使用した PC の仕様

4.4 利用状況

実験中 (2002 年 10 月 13 日〜2002 年 12 月 12 日) における解析済みデータサーバの一日の利用状況を図 4.6 に示す。このデータは FTP デーモン[11]のログによるものである。この図から、一日平均 3,000 件の取出しに利用されていることが判る。また、図 4.7 に同じ期間における一時間毎のアクセス数を示したが、この図から深夜においてもコンスタントにアクセスがなされていることが分かる。このことから、実験終了後バッチ処理により解析済みデータが取出されている事が推測できる。

また、このデータの取り出しの内 40% は 2 機種以上の計測機器データの同時参照であり、最大では 9 計測機器のデータが同時に取得されている。複数データの同時参照、取得は、統合したシステムで初めて可能であること、特に、システム間の情報伝達の同時性、高信頼性が得られた結果、放電終了後速やかにデータを登録できるようになったことによるものである。また、利用されているクライアントには、MS Windows 以外に Linux や Solaris 等が含まれるが、これは OS に依存しないオープン技術を利用した結果によるところが大きい。

尚、アクセス数の多いホストについて調査を行ったところ、毎週バッチ処理によりまとめて解析済みデータをダウンロードし、これを一次データとして二次データの作成を行っていることが判った。運用当初このようなアクセスは想定外であったため、手動でのアクセス件数として〜数十件/分程度を見積もっていた。ところが、このようなバッチ処理では数百件/分以上のデータ取出し要求があり、FTP サーバが処理できず、接続できなくなる事態が生じた。これは、通常クライアントからの接続要求がある度にスーパーサーバである inetd が FTP サーバである ftpd を起動する仕組みになっているが、プロセス起動に伴うオーバヘッドのため、接続要求を処理しきれなくなったためである。そこで、ftpd を inetd ではなく、デーモンとして起動することにより、この問題に対処した。

また、表 4.8 は計測種別毎のアクセス数の割合である。これから、上位 2 つの計測に対する参照が、全体の 40%を超えていることが分かる。解析情報サーバシステムは、サーバを複数持つことができるので、将来アクセス数が増えた場合、これらの計測データ用サーバを他の計測と分離することで、一箇所への集中を避けることが出来る。

[11]daemon。UNIX において何かをサービスするためにバックグラウンドで起動されるプログラム

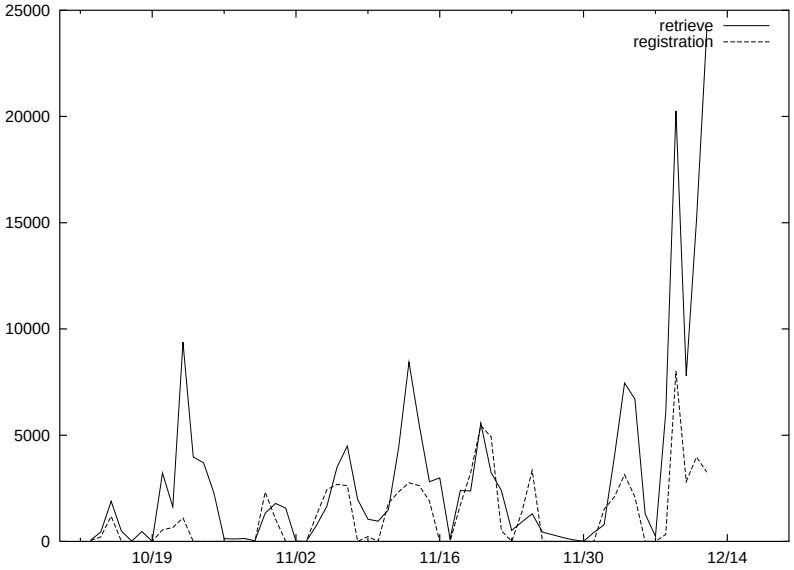

図 4.6: 解析済みデータサーバ利用状況

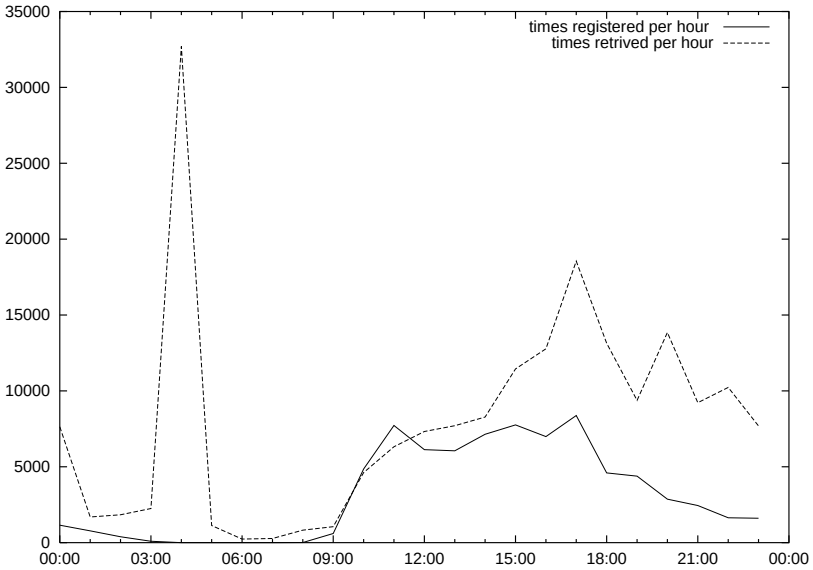

図 4.7: 1 時間毎のデータ取出し件数

計測名	説明	割合 (%)
wp	DIAMAG 計測蓄積エネルギー	20.1
firc	FIR 干渉計密度	20.6
Thomson	トムソン散乱計測電子温度	8.8
nbi2pwr	NBI2 号機入射エネルギー、ポート入射パワー、吸収パワー	7.5
nbi1pwr	NBI1 号機入射エネルギー、ポート入射パワー、吸収パワー	7.0
その他		36.0

表 4.8: 計測名毎のアクセス数の割合

第5章　データ可視化

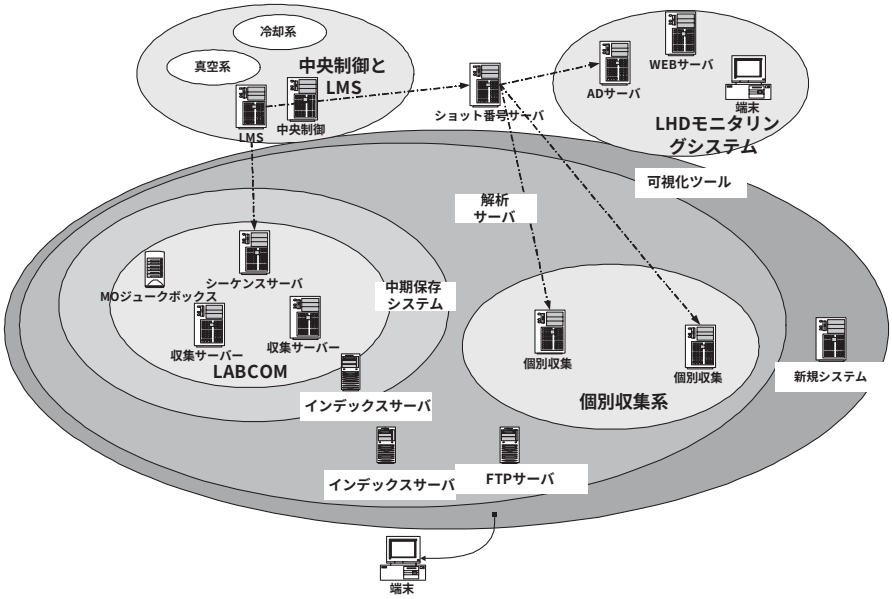

図 5.1: 統合化第三段階

データを直感的にすばやく把握するためには、実験データの可視化ツールが欠かせない。特に、外部の共同研究者が膨大なデータの中から有用なデータを見つけるためには、短い滞在期間の間に使い方を習得でき、自分の研究をサポートすることのできる、ユーザインタフェイスの優れた可視化ツールを必要とする。

このようなツールとしては、米国 Massachusetts 工科大学で開発された MDSPlus に付属する scope や、また米 General Atomic 社の開発した、IDL によるアプリケーションである ReviewPlus [24] 等がある。

これらのツールは、X Window System や特定の OS あるいは IDL のような商用アプリケーションに依存しており、これらの環境を用意することは、外部の共同研究者にとって負担となる。

本研究で開発した、汎用可視化ツール NIFScope は全てオープンソースで作られており、自由な配布が可能であり、MS Windows や Linux、MacOS X といった

複数のオペレーティングシステムで利用可能である。

また、NIFScope はデータ取出法の統一の第三段階として LHD 実験データのみにとどまらず、多様なデータに対応できるよう、汎用性を重視した点が特長である。この NIFScope の持つ汎用性はデータの読み込み部分をデータローダとしてモジュール化していることによる (図 5.2)。

このモジュール化にはオブジェクト指向の考えが取り入れられており、データファイルやデータサーバ上のデータを取得し、配列に格納するという一連の処理をデータローダ内部に隠蔽することで、NIFScope 本体はどのデータローダを使用した場合でも同一の方法によりデータを取得することができる。

例えば、解析データ用に AnaLoader、LABCOM データ用に LABCOM ローダというデータローダを提供しているが、新しいデータフォーマットを取り扱う必要が生じた場合でも、そのデータ用のデータローダを用意することにより、NIFScope 本体を変更することなしに対応することが可能となる。

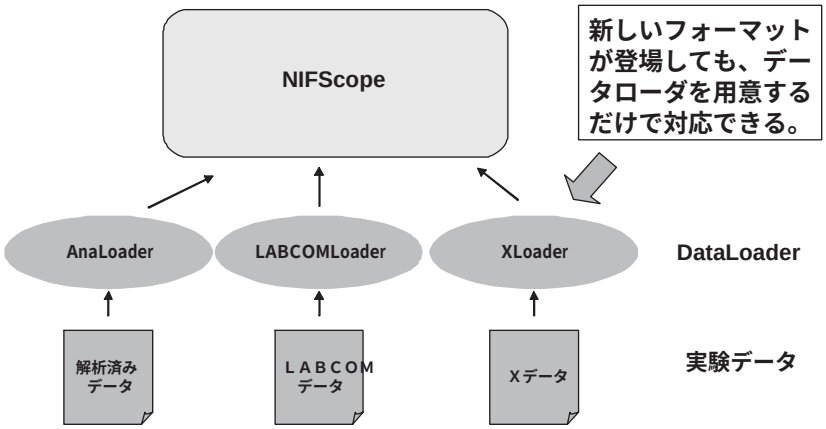

図 5.2: データローダのモジュール化

5.1　汎用可視化ツール NIFScope

NIFScope の開発に先立って共同研究者やプログラミング言語が不慣れな研究者のために、解析済みデータを簡易に表示するためのツール idraw(図 5.3) を開発したが、このツールは PV-Wave のマクロ言語によって作成されているため、他研究所で使用するためは同ソフトウェアの購入が必要となり、経済的な負担が発生するため、他研究所で自由に使用することができない。そこで、必要なアプリケーション一式を自由に配布することが可能なツールが必要になり、この可視化ツー

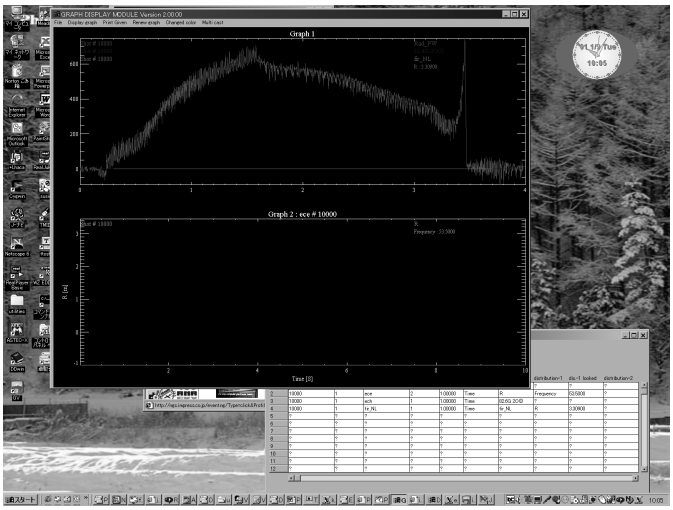

図 5.3: idraw 実行画面

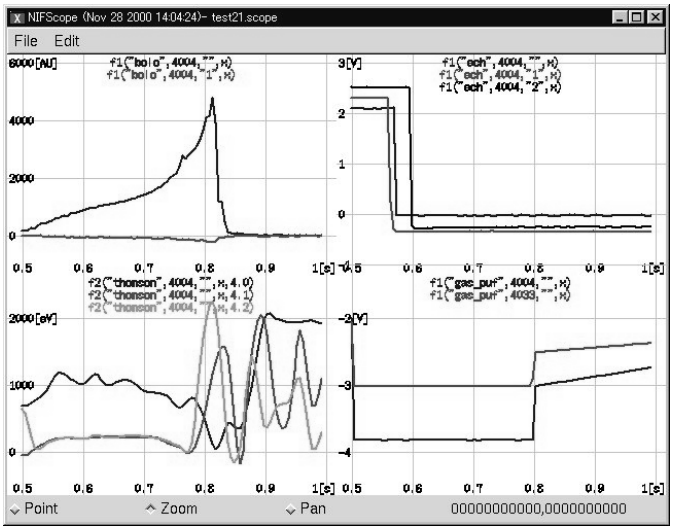

図 5.4: NIFScope 実行画面

ルとして NIFScope(図 5.4) の開発を行った。

NIFScope を設計する上で考慮したのは次の点である。

1. MS Windows/UNIX 両プラットフォームで使用できること
2. 分かりやすい GUI でユーザが自由にグラフを拡大・縮小等の操作が行えること
3. 特定の商用アプリケーションに依存しないこと
4. 特定のデータ形式に依存しないこと
5. ユーザが元データを加工し表示するためのマクロ言語を有すること

可視化ツールは様々な研究所で作られており、例えば MDS-Plus システムで標準的に使われている可視化アプリケーション scope はユーザインタフェイスに優れ、様々な表示設定が可能なソフトであり、また、マクロ言語 TCL により柔軟な運用が可能である。そのため、scope を本研究所のシステムに移植することを考えたが、このツールを本研究所で使用するにあたって次のような点が問題となった。

1. X Window System をベースにし、GUI に Motif が使用されている
2. MDS-Plus システムに強く依存している

1 は Scope で使用されている Motif と完全互換を持つ無料かつ自由に配布可能なツールキットが入手できなかったため (平成 12 年 3 月時点において)、scope が動く環境を無料で配布することは不可能であった。また、scope は X Window System ベースで作られているため MS Windows への移植が困難である。さらに、基本的に scope が MDS-Plus により格納されたデータを表示するシステムであり、このツールだけを切り出して使用することが困難であることがわかった。そのため、可視化ツールを独自に作る方法をとった。

1) MS Windows/UNIX 両プラットフォームで使用できること

UNIX の他、MS Windows をサポートするために採用したのが、GTK+(GIMP Tool Kit) である。このツールキットは GIMP (GNU Image Manipulation Program) のために作られたものであり、GNOME プロジェクト [26] で採用されている。Scope が採用した Motif は X Window System に依存しているが、GTK+は X Window System には依存せず、MS Windows プラットフォームに対応している。また、GUI の設計には優れた RAD (Rapid Application Development) ツールが欠かせないが、

フリーウェアである glade を使用することにより簡単に画面設計を行うことができた。

2) 分かりやすい GUI でユーザが自由にグラフを拡大・縮小等の操作が行えること

scope は優れたユーザインタフェイスを有し、複数データの表示、グラフの一部の拡大等優れたユーザインタフェイスを持っており、新規に開発するツールは scope のユーザインタフェイスを元にすることにした。ただし、scope そのものは、ボタンが押された時の動作等、GUI に関する部分は Motif の UIL という言語によって書かれており、これを GTK+に移植することは困難と思われた。そこで scope を Java に移植した JavaScope [27] を元に開発を行った。JavaScope はこれらの GUI の定義は Java のプログラムで記述されており、Java と C++の文法が似ていることから、比較的簡単に移植できると判断したからである。

3) 特定の商用アプリケーションに依存しないこと

本ツールは全てオープンソースのコンポーネントより作られており、商用アプリケーションは一切使っていない。また、使用しているモジュールはそれぞれ GPL (Generic Public License)[28] あるいは、GPL に準拠したライセンスで配布されているため、NIFScope もこれらのライセンスに沿って、アプリケーションの配布、さらには改造を自由に行うことが可能である。

4) 特定のデータ形式に依存しないこと

Scope や Review Plus 等は個々の研究所がそこで収集されたデータを可視化するために作られたものであるため、他の研究所・システムで使用する場合は移植作業が必要となる。この手間を省くために NIFScope ではデータロード部分をモジュール化した。これにはオブジェクト指向の考え方を採用し、データおよびデータローダをオブジェクトとして取り扱い、データ表現を抽象化することにより、新規フォーマットのデータにも柔軟に対応できるようにした。具体的には、図 5.5 に示すクラス図のように、各種データクラスは抽象クラスである実験データクラスを継承する。実験データクラスは実験データを読み出すのに必要な物理量の数、単位等を取得するメソッドのインタフェイスを提供する。一方、実験データクラスを継承する解析済みデータクラス等は、これらのインタフェイスの実装を行う。これにより、実験データクラスを継承するクラスは、データを読み出すのに必要な最低限のインタフェイスがあることが保証されるため、新しいデータフォーマットが登場しても、これらのインタフェイスを利用することで、既存コードを変更することなく、データを読み出すことができる。また、データローダはダイナミッ

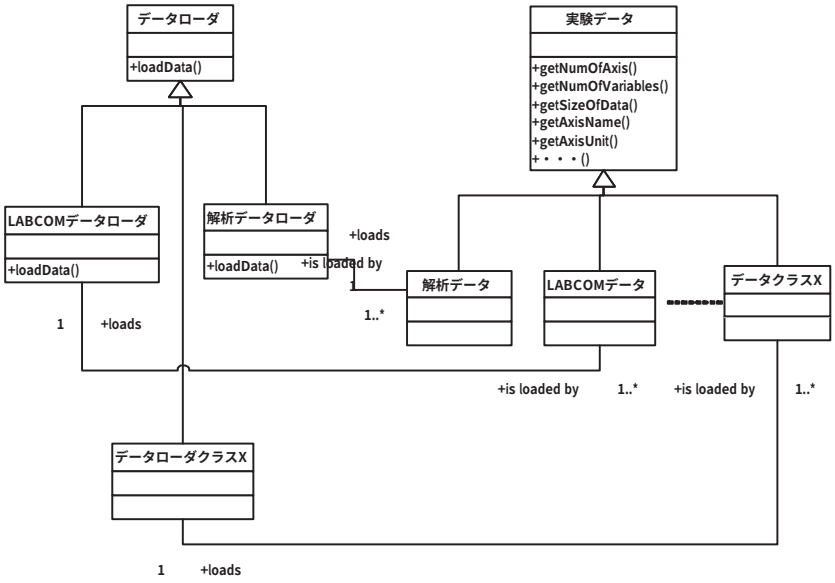

図 5.5: データローダの静的クラス図

クリンクライブラリによるローダブルモジュールとして提供するため、新規データローダを使う場合でもアプリケーション本体に修正を施す必要がない。

また、データローダに与える実験データのソースは「protocol://host/filename」というインターネットで使用される URL(Uniform Resource Locator) 形式で表すことにより、データアクセス方法の違いを隠蔽している。

例) 解析済みデータローダの場合

1) ローカルディスクに保存された、解析済みデータファイル/tmp/shot01234.dat 中の計測量 “t”

file:///tmp/shot01234.dat#t

2) 解析済みデータサーバ上の計測名 diag1、shot 番号 1234、計測量 “x”

remote://diag1/1234#x

5) ユーザが元データを加工し表示するためのマクロ言語を有すること。

ユーザ側からの要望として、「実験データの表示の他に自分で定義した関数を表示させ比較したい。」あるいは、「元データに対し処理を行って表示したい。」というものがあり、これに応えるためにはアプリケーション内部にマクロ言語を内蔵

```
# 自前の sin 関数
# sin(x) = x - x^3/3! + x^5/5! - ...
def mysin(x)
  result = x
  tmp = x
  term = x
  fact = 1
  i = 1
  while ( abs(term) > 1.0e-10 )
     fact = fact * ( i + 1 ) * ( i + 2 )
     tmp = - x * x * tmp
     term = tmp / fact
     i = i + 2
     result = result + term
  end
  return result
end

print mysin(pi()/ 4.0)
```

図 5.6: LL で書かれたプログラムのサンプル

する必要があった。本研究所のユーザの多くは、FORTRAN または、BASIC 等のプログラム言語を使用しており、全く新しい言語の習得を要求することは困難であると思われた。また、どちらも汎用言語として設計されているため、その言語仕様が非常に大きいが、マクロ言語の主目的はユーザが実験データを加工し表示することであるので、これら現代的なコンピュータ言語が持つ、高度な機能は必要としない。そのため、なるべく FORTRAN や BASIC に似た文法を持つ仕様の小さな言語 LL(Little Language) を新たに作ることにした (図 5.6)

5.2 NIFScope Ver.2

NIFScope は与えられた式 $y = f(x)$ の x 座標を変えながら $f(x)$ を評価することによって、グラフの描画を行っている。このため、実験のデータを描画するに

は、離散的に得られた数値列

$$(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$$

を補完する関数 $y_i = F(x_i)(i = 1 \ldots n)$ を定義し、この関数を評価することによりグラフの描画を行う方法をとった。ところが、さらなるユーザの要望として、離散的なデータはそのまま表示する機能に対する要求があった。これを行うためには、x 座標、y 座標を配列として扱う必要がある。一方で、データを表示する際 y 軸を定数倍して表示する等の処理が必要となることがある。データを一つの関数 $f(x)$ として与えられる場合には、関数に定数倍した $a * f(x)$ という式を与えることにより、簡単に描画することができる。しかしながら、配列を与える方法であれば、ユーザはその配列の要素すべてにデータを掛け合わせるというプログラムを書く必要がある。ユーザの利便性を考えれば、配列 y の要素すべてを定数倍するという操作は $a * y$ という表記により行われることが望ましい。

この機能を実現するためには、配列を一つのオブジェクトとして扱うことができるような仕組み、すなわち、オブジェクト指向の考え方を導入する必要があった。また、オブジェクト指向は研究所で必要とされる一般の数学的な計算を行うためにも、有用である。例えば、

$$A + B$$

という単純な演算においても、A が浮動小数、複素数、ベクトル、行列…等それぞれに対して計算方法が異なる。オブジェクト指向型言語であれば、この式は「A というオブジェクトに対し、“+ B” というメッセージを送る」という解釈になり、「＋」という演算子に対しどのような処理を行うかということは、オブジェクト自身の判断によるため、オブジェクトの型ごとに表記を変える必要がない。

他方、現在の LL の文法ではこのような問題に対処するために、例えば、ベクトル型であれば、a[]、複素数であれば a% とあらわすなど、処理の分岐は、文法解釈の段階で行う必要がある。この方法の利点は、文法解釈の段階で行われるため、一度解釈させてしまえば、実行効率がよくなるという点であるが、反面文法が複雑になり、プログラムを行う際に覚えなければならないことが増加する。

今後のメンテナンスのことを考え、LL に新たな文法を追加することをやめ、既存のオブジェクト指向型言語の NIFScope への取り込みを行った別バージョンの NIFScope の開発を行った。これによりユーザには BASIC ライクな文法をもつものと、より高度な言語機能をもつ二種類のツールを提供することができる。

言語の選択

NIFScope が組む込み型言語として採用する上で、以下の条件が必要である。

1. インタプリタであること

2. オープンソースであり、自由な配布が可能であること
3. 少なくとも、UNIX および MS Windows をサポートしていること
4. アプリケーションへの組込み機能があり、アプリケーション側から言語処理系の機能が使えること
5. オブジェクト指向型言語であること
6. オペレータ (演算子) の多重定義が可能であること

1〜4 までの条件は既存の LL を置き換えるのに必要な条件である。この条件を満たすものとして、Python [29]、Ruby [30]、Tcl/Tk [31] 等、多様な選択肢がある。

一方、5、6 はユーザが解析を行う際に必要な機能である。解析を行う為に必要な数式中では言語が持っている型だけでは不十分で、行列やベクトル計算等の型を新たに定義する必要がある。これらの型を自然な形で定義するためにオブジェクト指向という考え方が必要となってくる。また、数学的に自然な書き方をするために、オペレータの多重定義が必要である。行列の計算を例にとると、

$$C = A + B$$

オペレータの多重定義ができない場合、

$$C = A.add(B)$$

あるいは、

$$C = add(A, B)$$

といった不自然な記述をしなければならなくなる。しかしながら、オブジェクト指向型言語が演算子の多重定義を許すだけでは、不十分である。例えば、

$$A * B$$

という演算はオブジェクト指向型言語では、

$$A.*(B)$$

と解釈される。このことは、数学的クラスを定義する際に問題となる。一例として、すでにスカラー型クラスが定義されていたときに、後から行列型を定義することを考える。このとき、スカラー型のオブジェクトを S、行列型オブジェクトを M とすると、両者間の乗算演算 '*' には次の関係が成り立つ必要がある。

$$M * S \equiv S * M$$

ところが、左辺は

$$M.*(S)$$

と解釈されるため、後から定義される行列型の '*' の定義が使用される。一方右辺は

$$S.*(M)$$

となるが、スカラー型である S には後から定義された行列に対する乗算は考慮されていないのでエラーとなる。C++や CLOS 等、一部のオブジェクト指向型言語では、この問題を解決するために、新たな型が作られた場合には、既存の型との演算をすべて再定義することにより、解決を行っている。一方、Ruby 等では、次のような方法をとる。

$$S.*(M)$$

が実行されると、S はオブジェクト M との乗算方法が分からないので、

$$S*(M) \rightarrow (M.coerce(S)).*(M)$$

という具合に、自分自身 (S) を M のメソッド coerce により、M と乗算できる型に換し、その型のオペレータ'*' を呼び出すことにより、演算を行っている。また、Python でも同様に coerce を使う方法が提供されているが、その他に演算子の交換則を使い、

$$S * M \rightarrow M * S$$

という変換を行って演算を行う機能が提供されている。C++の方法では、実行速度の面でメリットがあるが、型が増えると再定義しなければならない演算子が増大するため、必要なコードの量が増大する。そのため、今回のような組込み用の補助的な言語としては、簡潔に書ける Ruby や Python 方が優れていると判断した。

以上のような検討の結果、Ruby、Python が候補として残った。どちらも、他の環境から利用するための API が充実しており、アプリケーションに組み込むことが可能である。また、言語としての特徴としては、下記の点で両者は非常に類似している。

- スクリプト型言語
- 純粋オブジェクト指向型

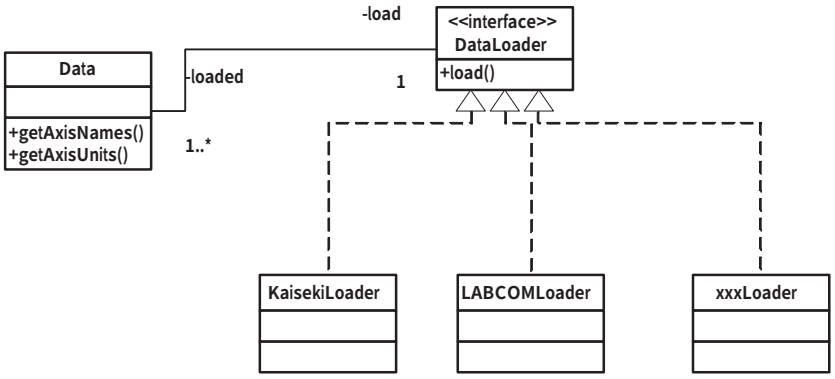

図 5.7: データローダーの静的クラス図 (2)

- 強力なテキスト処理能力
- CGI 用等、インターネット上のプログラミング言語として普及している

しかしながら、両者を比較し、Ruby の方が国内での普及率が高いという点と、Python はインデントがループ等のひとつの制御構文として使用されるため、自由なフォーマットでプログラムを書くことができない。という 2 点により、最終的に Ruby を選択した。

モジュール

使用言語を LL から Ruby に変更するに当たり、データローダを C 言語によるシェアドライブラリから、Ruby の module に変更した。これにより、C 言語の他に Ruby を使って新しいデータローダを定義することが可能になった。尚、この変更により各モジュールの依存関係は図 5.7 のように変更された。すなわち、データローダモジュールはインタフェイスのみを提供し、実際のインプリメンテーションは各実験データに対応したデータローダモジュールによって行われる

NIFScope Ver.2 の実行画面を図 5.8 に示す。前のバージョンでは、データを補完した関数を与えることにより描画していたため、サンプリングによっては、ピークやデータのギャップを見落す可能性があった。このバージョンではデータの X 座標、Y 座標の配列を与え、座標間を直線で描画するので、本来存在しない点を描画することはない。

グラフ上の曲線は一つのオブジェクトであり、これらは x,y という x 座標および y 座標の配列を表す属性を持つ。これらの属性は Ruby の Array クラスを継承した NArray というオブジェクトであり、NArray クラスでは ‘+’,‘ - ’ などのオペレータをオーバーロードしている。これにより、a という曲線の y 座標に全要素に 1000.0

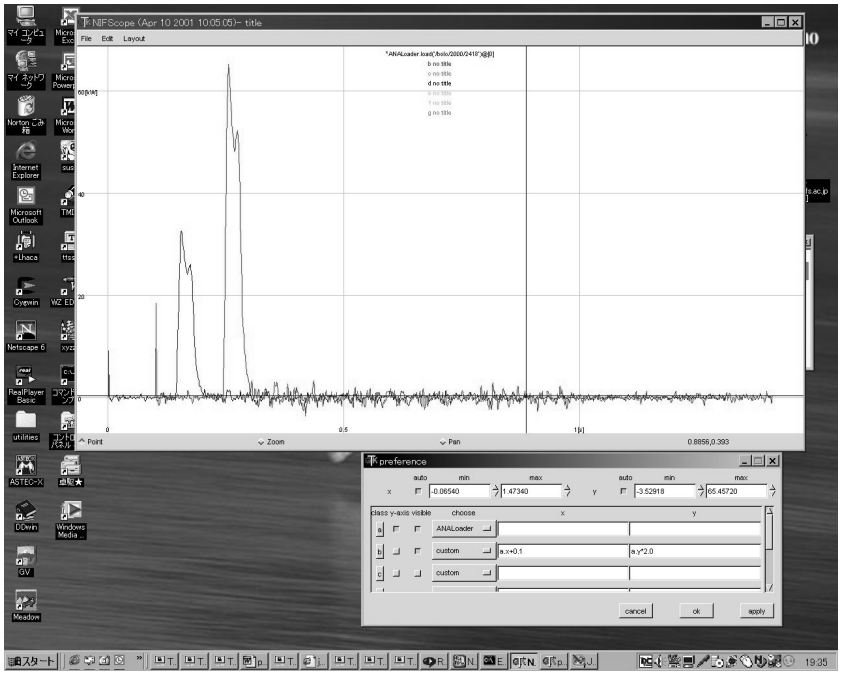

図 5.8: NIFScope Ver.2 実行画面

を加えたものを b という曲線の y 座標にするという作業が、

$$b.y = a.x + 1000.0$$

という簡潔な式によって記述することが出来き、普通のプログラミング言語に必要なループを使ったプログラムを記述する必要がない。これは前述の Ruby の持つオペレータの多重定義の簡便さによるものである。

ユーザからは、このように GUI を使って直感的にデータの可視化を行いたいという要求がある一方で、毎ショット毎のデータをバッチ的に自動的に表示したいという要望があった。これらの相反する要求を満たすためにも、マクロ言語は有効である。この要求に答えるために解析済みデータツール (図 5.9) を作成した。このツールはレイアウトやコマンドなどを記述したファイルを読み込みショット番号をパラメタとして、あらかじめ表示法等が定義されたグラフの描画を行うものであるが、プログラム本体は NIFScope をそのまま利用している。外部から Ruby のプログラムにより NIFScope 内部の変数の値を書き換えることによって、この機能を実現した。このツールは、ショット番号の更新および新規データの登録の IP マルチキャストパケット (8 章参照) を受信しており、最新のデータの自動的な表示・印刷が可能である。

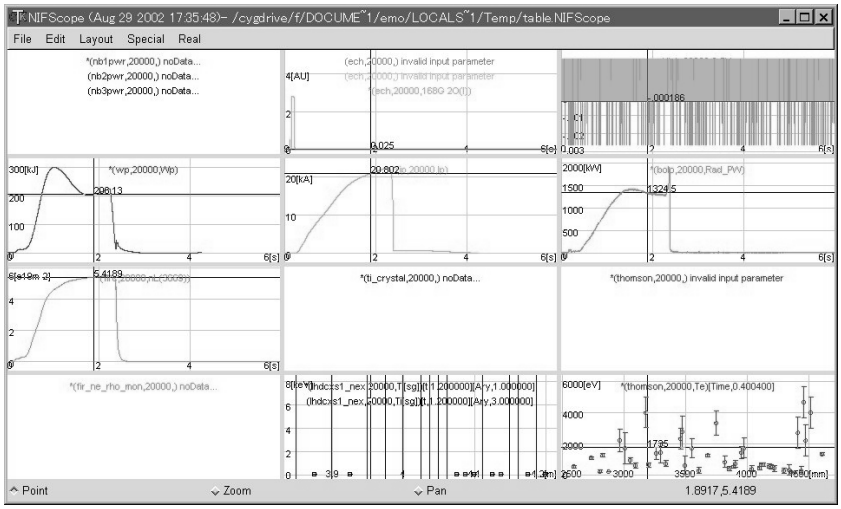

図 5.9: 解析済みデータツール

第III部

統合化 2
～ 情報の共有化 ～

情報の共有化についての方法と手段

第 III 部では、実験に関する情報の共有化について取り扱う。

LHD 実験では、システムが分離されており、一部にセキュリティの高いシステムが混在するため、システム間での情報伝達が円滑に行われていなかった。

データ取得法の統一では共通インタフェイスを用いる方法によってシステムの統合化を行ってたが、ここではミラーサーバやプロキシーサーバ等の仲介者を用いる方法で、システム間の連携を高めた (図 III.a)。

これは、既存のシステム間の連携が強く、新たに中間層を用いてインタフェイスを変更することが困難であると判断したためである。

具体的には以下のシステムの開発を行った。

- 実験情報のデータベース化
- リアルタイムの実験情報通知

前者はミラーサーバにより、LMS が保有しているデータを他のシステムから利用可能にするものであり、後者はショット番号等の通知や実験データの監視をプロキシーサーバおよびマルチキャストを利用することにより、多数のシステムからリアルタイムでデータを参照可能とするものである。

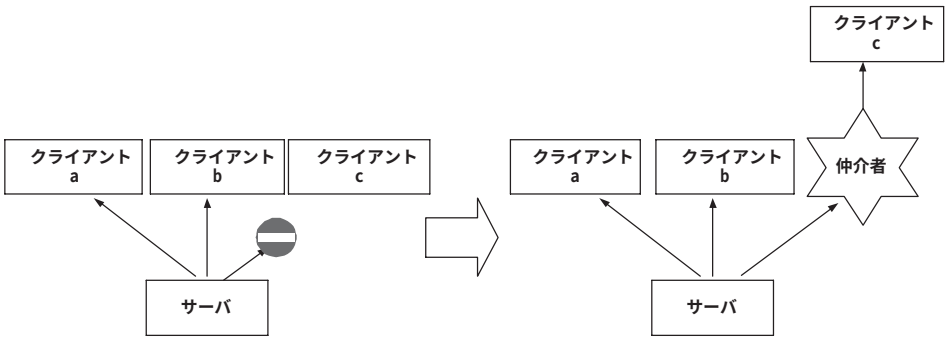

図 III.a: 仲介者を用いた統一法

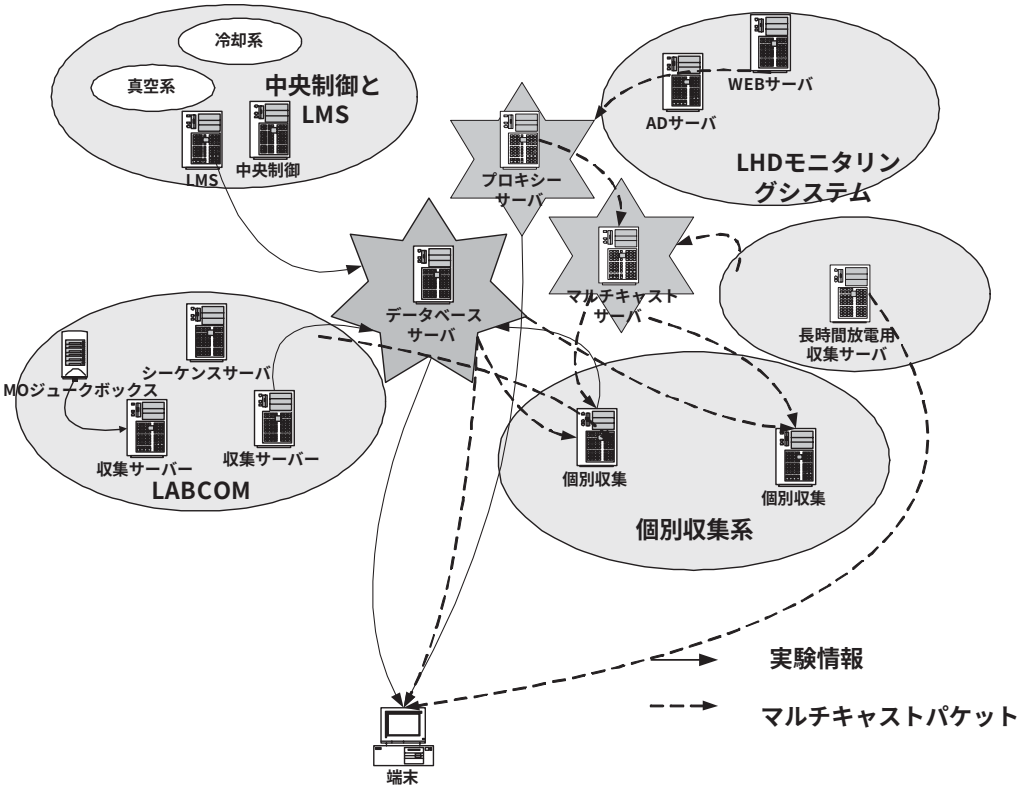

図 III.b: 情報の共有化

第6章　データ検索

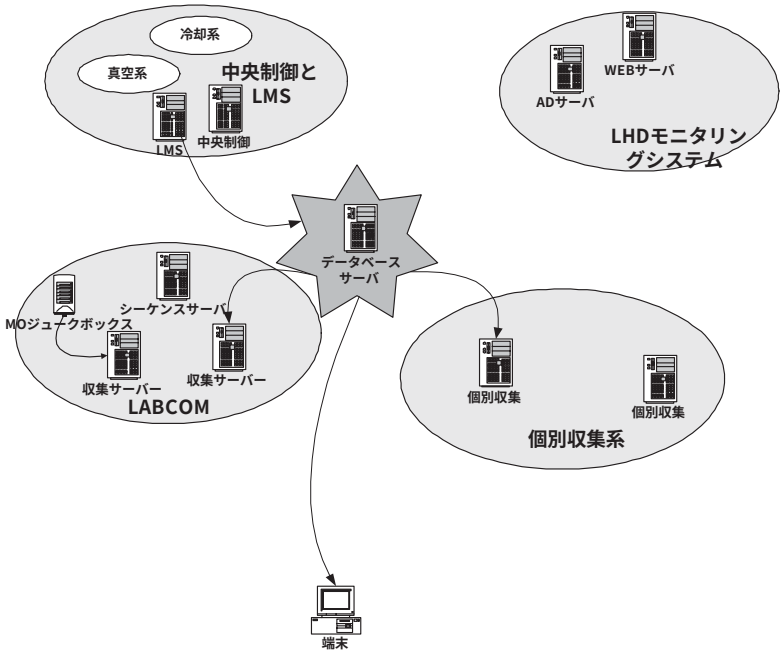

図 6.1: 実験ログシステム

解析済みデータサーバシステムにより、多種の計測データを同一インタフェイスにより取得できるようになり、平成 14 年度まで 59 万件のデータが登録されている。これにより、ショット番号と計測名を指定すれば、必要なデータが取出すことが可能となった。ところが、実際には研究者が必要とするデータはそのうち一部であり、自分が必要とするデータがどのショット番号なのかを知る必要がある。以前はショット番号を知るためには、紙ベースで記録されていた実験ノートを使用していた。しかしながら、この方法では必要なデータを検索するのに手間がかかり、また、外部研究者の場合には、実験がどのような条件で行われたのか判らないことが多い。この問題を解決するためには、NBI の入射の有無、磁気軸、ガス種等の実験条件から実験を特定する必要がある。

これらの情報は、LMS が管理している情報を参照すれば得られる。ところが、LMS はセキュリティの関係から、他のネットワークから分離されているため、直接利用することができない。このため、LMS のミラーサーバを設置することで他のコンピュータからも利用できるようにした。

6.1　システム概要

実験ログの元となるデータは、LMS システムによって収集された各プラズマ放電実験における中央制御システムに接続された周辺サブシステムのステータス情報である。これらのステータス情報は PLC を通じ収集され、MS Windows NT 上の RDB(Oracle 7) に保存される。また、実験コーディネータ (責任者) によるメモなどもこのデータベースに書き込まれ、その実験がどのような条件で行われたかという情報を与え、これらの情報を元に NBI の入射の有無等の実験条件を調べることができる。ところが、LMS システムは実験機器のコントロールを行う為、他のシステムに比べセキュリティの高いネットワーク上にあり、他のネットワークから直接アクセスを行うことができない。そこで、一般のユーザが利用できるように、LMS システムのネットワークの外に実験ログデータベースサーバを設置し、ここで LMS のデータのコピー (付録 C) を検索できるようにした。使用したデータベースはすでに解析情報サーバで実績のある PostgreSQL を採用した。また、LMS・実験ログデータベース間の同期は Java で書かれたデーモンを用いており、下記の手順で行っている。

1. ユーザが必要とするデータは LMS 上の複数のテーブルから、単一テーブル（MainParameter）に値をコピーする。

2. LMS データベース上のテーブル MainParameter にデータの行の鮮度を表すカラム UpdateID と状態を表すカラム Status を追加する。新しく更新された行は大きな UpdateID を持ち、Status は ('I'nserted、'U'pdated、'D'eleted) の三状態がある。

3. LMS データベース上の MainParameter に新規に行を挿入する際にはシーケンシャル番号を+1 し、これを UpdateID に割り当て、Status を'I' にする。

4. 一方、行に変更が行われた場合には、シーケンシャル番号を+1 し、新しい UpdateID とし、Status を'U' とする。また、行が不要になった場合でも、行の削除は行わず、新しい UpdateID を割り当て、Status を'D' とする。従って、最も新しく変更が行われた行が最大の UpdateID を持つことになる。

5. 実験ログデータベースサーバ上のデーモンは LMS データベース上の MainParameter の内容を実験ログデータベース上の MainParameter に反映させるために、30 秒毎に両者の UpdateID を比較する。もし、LMS 側の UpdateID の方が大きければ、実験ログデータベースよりも新しい行があることが分かる。

6. LMS データベース上から、実験ログデータベースの UpdateID の最大値よりも大きな UpdateID を持つ行を取得し、その行の Status が'I' であれば、実験ログデータベースに新たに行を挿入。'U' であれば更新、'D' であれば対象行を消去する。対象となる行の特定には、ショット番号を用いる。

第 4 サイクルでは上記方法によって運用を行っていたが、多様なデータが単一テーブルに書き込まれるため、ユーザが必要とするデータはその極一部にもかかわらず、データの取り出しや書き込みに対するレスポンスが悪化した。これは即時性が要求される実験コーディネータの各ショットに対するコメント入力に重大な影響を与える。このため、実験コーディネータコメントを別テーブル (CoordinatorComment) とし、このテーブルも参照できるようにした。この変更に対応して、LMS から実験ログデータベースへの反映は以下のように変更をおこなった。

1. 上記の手順により、LMS の MainParameter を実験ログデータベースに反映

2. LMS の CoordinatorComment を読み取り、その内容で実験ログデータベースの MainParameter の内容を更新

既存の LMS データベースへの影響を最小限にするよう配慮したために以上のような方法をとった。しかしながら、この方法では機器の増設や新しいデータの追加という要求に対応するためには LMS、実験ログデータベース双方のテーブルの変更が必要とされる。これを前述のデータと同様に LMS 経由にすると負荷が LMS に集中しするという問題が発生する。また、30 秒毎にポーリングを行っているため、元データである LMS データベース上の変更が速やかに実験ログデータベースへ反映されず、最大 30 秒の遅延が発生することである。このため、現在の 3 分間隔での放電実験中に実験ログデータベースから前回のデータを参照し、次回の実験器具のパラメタ変更を行うということが難しい。それ故、新規のデータに関しては直接実験ログデータベースに書き込む方法をとった。

6.2 検索用アプリケーション

実験ログデータは RDB である PostgreSQL により提供されているため、ODBC (Open DataBase Connectivity) を用いることにより、スプレッドシートやデータ

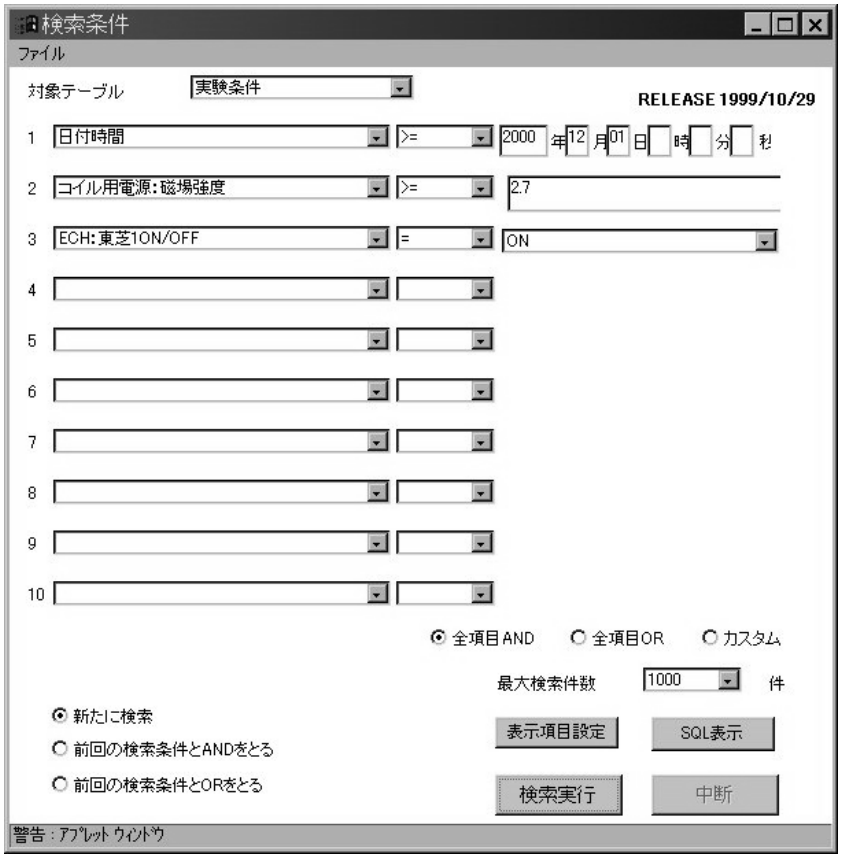

図 6.2: 検索条件設定ウィンドウ

ベースアプリケーションといった汎用のオフィスアプリケーションを使用し、データを取出すことができる。ところが、これらのツールを用意することはユーザの負担となり、また、高度な検索条件で検索を行うためには SQL の知識が必要となる。そこで、ユーザの便宜を図るためフリーで配布できる Java ベースの検索アプリケーション、ShotSummary を開発した (図 6.2、6.3)。ShotSummary は検索条件をメニュー等により直感的に操作し、論理和、論理積の組み合わせを含んだ複雑な検索条件を画面上で SQL の知識なしに設定することができる。

例えば、SQL を知らないユーザにとっては「日付時間」というフィールドに対しどのような比較が可能なのか、比較対照となる値にはどのようなフォーマットが適用できるか判断できない。ShotSummary では、検索条件として日付時間フィールドを選択すると、データ作成日付のデータ型である日付時間型の可能な演算子として、

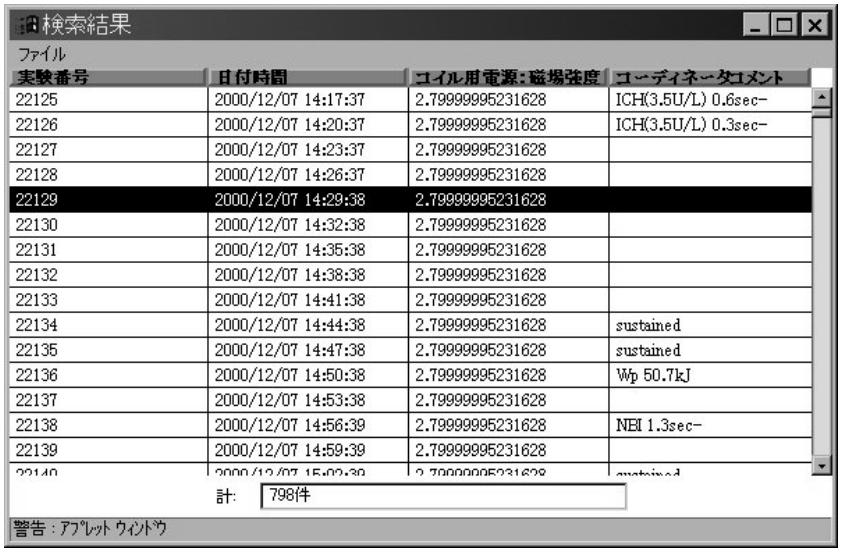

図 6.3: 検索結果

- `is null` [1]
- `is not null`
- `=,!=,<=,>=`

等がリストから選択できるようにしており、また比較対照の入力フィールドでは、年、月、日の入力フィールドを設けていることで、GUI により

```
select * from MainParameter from dDataCreationTime <= '20010208 09:37:00'
```

といったSQL 文を SQL の知識無しに組み立てることができるようにしている。このようにデータベース上の型に応じ自動的に比較演算子、入力フィールドが現れることで、入力間違いが生じないようにしている。複数の検索条件がある場合は、表示されている検索条件フィールドに対して、論理和または論理積をとることによって行っているが、さらに、

条件 1 and (条件 2 or 条件 3)

といった複合条件による検索を行う為に、一度検索した結果に対する新しい条件の論理和・論理積を行えるようにしている。これは、「条件 2 または条件 3」と

[1] is null はカラムが null(空) であるかどうかを判別する演算子

いう検索を行った後、この結果に対して条件 1 による検索をすることにより行っている。さらに、このような検索を直接的に各フィールドに対する論理式を

$$1 + (2 * 3)$$

のような式で与えることによっても、設定することが出来る。

ユーザは ShotSummary を画面上で直接検索することもできるが、こうして組み立てた SQL 文を保存しておき、ShotSummary または、自身が作ったプログラムの中から ODBC 等のデータベース接続機能を使って呼び出すことも可能である。

ShotSummary は Java アプリケーションであるため、OS 非依存であり、JDK (Java Development Kit)1.1 をサポートする環境であれば実行する OS を問わない。また、ShotSummary は Java アプリケーションであると同時に、Java アプレット[2]として提供されており、各種 OS から Web ブラウザや Java 実行環境から検索を行うことが出来る。ShotSummary により得られた検索結果は CSV 形式によりファイルに保存し、他のアプリケーションから利用しやすい形で提供することができるが、通常の Java アプレット版の ShotSummary ではこれができない。これは、通常 Java アプレットは sand box と呼ばれる特殊な環境で実行され、ローカル資源 (ハードディスク等) へのアクセスが制限されているためである。これは、Java アプレットを署名付アプレットとすることで回避している。署名付きアプレットは 10 章で詳しく説明する。

6.3　プラズマパラメタ検索

実験ログは実験中の機器の状態を記録したデータベースであり、これにより NBI や ECH の有無、入射時間等を知ることができるが、計測された最大イオン温度や密度など、実際のプラズマ状態を代表するパラメタなどに対する検索ができれば、より目的にマッチしたデータを見つけだすことが可能となる。このため、プラズマパラメタの情報を提供するため、プラズマサマリーテーブルを作成した。これらの情報を得るためには、実験終了した後、登録された解析済みデータを元にさらに解析を行わなければならない。実験中はプラズマサマリーテーブルを作成することが出来ないため、夜間バッチ処理により解析済みデータから作成を行っている。

これは次のようにして行っている。(図 6.4)

- 解析済みデータの新規登録・更新が行われるとトリガにより、解析済みデータ更新テーブルにどのデータの更新が行われたかを登録する。

[2]アプレットは Web ブラウザ上で動作する Java プログラムであり、一方 Java アプリケーションは単体で実行可能なプログラム。

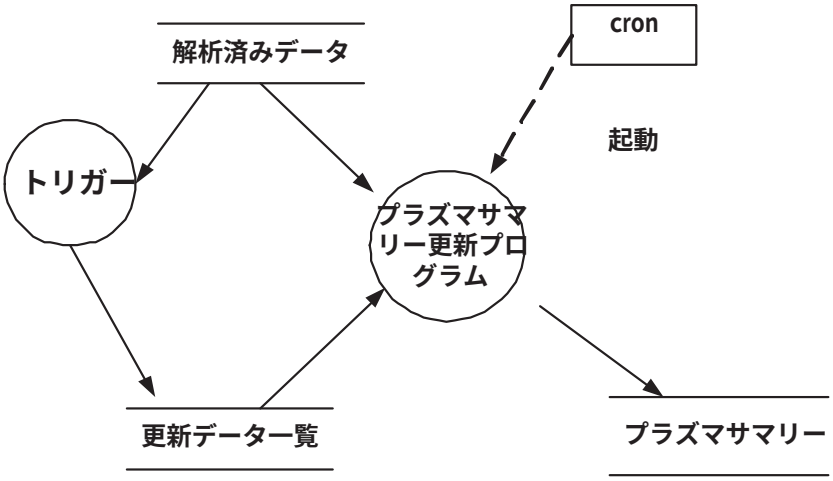

図 6.4: プラズマサマリー作成データフローダイアグラム

- 夜間バッチにより起動されたプラズマサマリー作成プログラムは上記テーブルを参照し、プラズマサマリーテーブル上の該当するデータの更新を行う

データの検索は実験ログテーブルの検索と同様にショット情報検索アプレットを利用し、検索対象テーブルを変更することによって対応することができるので、特に大幅な変更することなく、プログラムを利用するすることができた。その他にODBC を利用することができることから、MS Windows の多様なアプリケーションからの利用を可能としている。

6.4 ExpLog ビュー

プラズマサマリーと実験条件テーブルをショット番号をキーに結合 (join) することにより、機器情報とプラズマの物理パラメタを組み合わせた複雑な条件で検索できる。ここでは ExpLog というビュー[3](付録 C 参照) を提供し、ユーザが SQL 文を書くことなく、普通のテーブルのように検索できるようにしている。

ところが、通常の結合である内部結合では、必要なデータが検索できないことがある。これは同一のショット番号の実験情報とプラズマサマリーの生成されるタイミングが異なるためである。実験情報テーブルはプラズマ放電実験の開始、終了時に新しい行が生成、更新される。一方、プラズマサマリーは夜間バッチまたは手動により作成される。このため、実験情報テーブルには存在するが、プラズマサマリーテーブルには存在しないショット番号が生じる。これを普通に結合を行うと、

[3]仮想的なテーブル

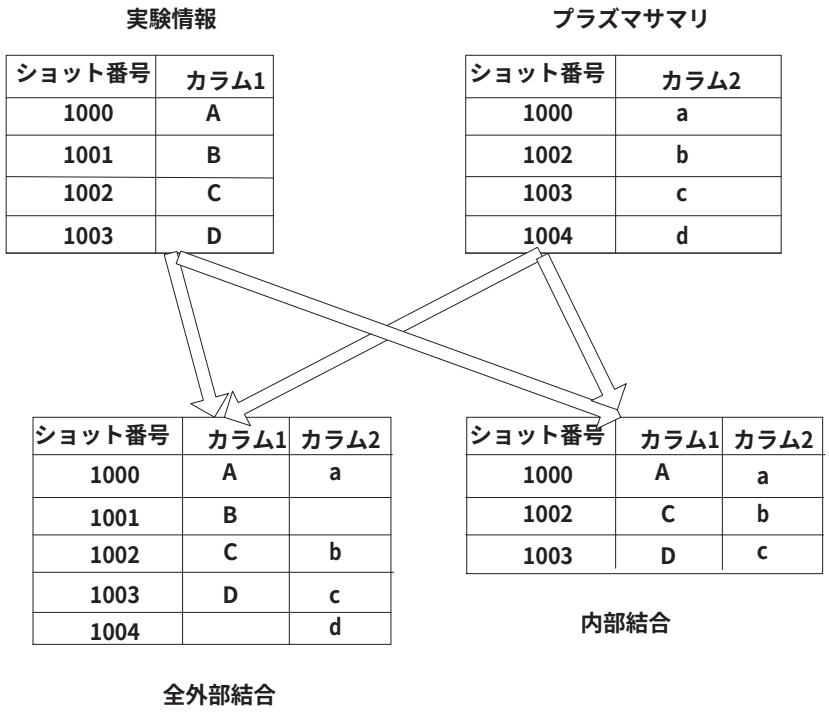

図 6.5: 内部結合と外部結合

一方のショット番号が存在しない場合、結合したテーブルのショット番号が存在しなくなる (図 6.5)。二つのテーブルの内、片方でもショット番号がある場合は、結合したテーブルに対しても対応したショット番号が存在するようにするためには、全外部結合 (FULL OUTER JOIN) の機能が必要である。この機能は PostgreSQL では 7.1 より可能となった (運用当初は PostgreSQL 6.5.3 を使用していた)。

第7章　LHDリアルタイム モニタリングシステム

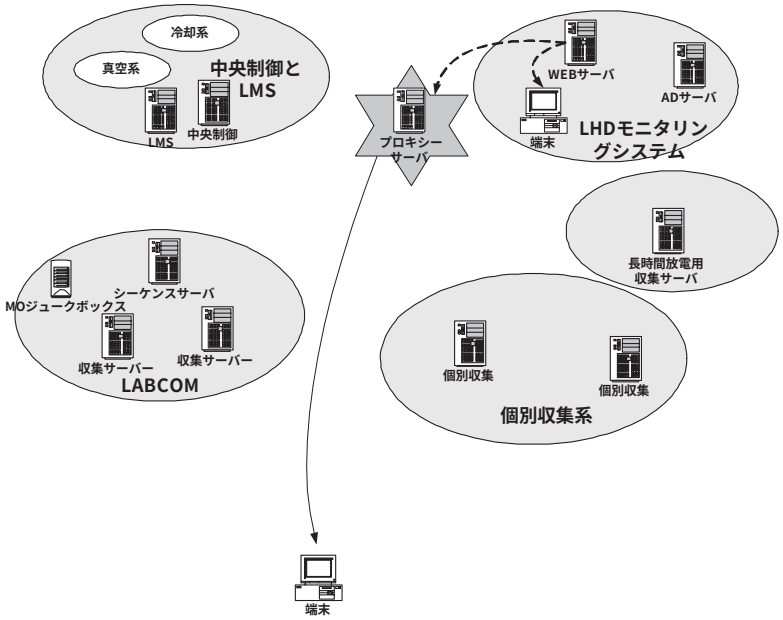

図 7.1: LHD リアルタイムモニタリングシステム

LHD リアルタイムモニタリングシステムは、LHD 実験の安定運用のため、コイルの状態の実時間での監視および過去データの参照を可能とするシステムである。本章では、LHD モニタリングシステムおよび関連するシステムについて述べる。

このシステムも LMS と同様に他のシステムから独立したネットワーク上に設置されているため、直接アクセスできるコンピュータが限れていた。そこで筆者は、プロキシーサーバを設置し、他のネットワークからそのためより多くのユーザが利用できるようにした (図 7.1)

7.1 超伝導コイル監視用リアルタイムシステム

LHD実験に先立って数々の機器の開発がLHD実験の数年前から始められ、その一環として超伝導コイルの冷却・励磁試験[47]が行われた。実験には数ヶ月のおよぶコイルの冷却・昇温期間を必要とし、この期間中リアルタイムでコイルを監視する必要があるため、刈谷らのシステムが用いられた[48]。このシステムの特徴の一つは、他の収集システムに先駆けてUNIXワークステーションベースでシステムを構築したことである。データ収集のため、64チャンネルのA/D変換器が使用された。このA/D変換器はSun Sparcワークステーション用にリアルタイムで高速にデータを取り込む目的で特注されたものである。

このシステムのもう一つの特徴の一つはIPマルチキャスト[49]を用いたことである。Sunワークステーションに取り込まれたデータは、IPマルチキャストを用いてデータを配信する。TCP/IPを用いたデータの送受信法には1) ユニキャスト 2) ブロードキャスト 3)IPマルチキャストがある。ユニキャストはWebやFTP等一般的なTCP/IPの通信で使用されているもので、特定の2台のマシン間を一対一で通信を行う方法である。このため、クライアント数の増加に比例して通信量が増加し、サーバに負荷が集中する。ブロードキャストは特定のネットワーク上に存在する全てのマシンにデータを送信する方法である。それに対しIPマルチキャストは単一サーバから特定のグループに属する全てのクライアントにデータを送信する方法である。このため、ユニキャストに比べサーバにかかる負担が少なく、また、ブロードキャストに比較し、データを必要とするクライアントにのみデータを送ることからネットワークにかかる負担を軽減することができるため、複数のクライアントに効率よくデータを送ることができる。刈谷はさらにUNIXで用いられるフィルタリングという概念をシステムに組み込んだ。フィルタとは標準入力からデータを読み込み、その処理したデータを標準出力に出力するプログラムである(図7.2)。個々のプログラムは単一機能しかもたなくても、あるフィルタの出力を別のフィルタの入力に繋いで行くことによって複雑な処理を行うことができる。このような利点から、動的に変動するコンピュータ環境にも柔軟に対応できるシステムの構築できることが示された。

7.2 LHDリアルタイムモニタリングシステム

山口らは刈谷のシステムをLHD実験用に拡張・発展する形で、このシステムをベースにLHDリアルタイムモニタリングシステムの開発を行った。このシステムの目的は、真空容器の歪や温度をリアルタイムで監視することであり、刈谷のシステムと同様に、容器等に取り付けられたセンサをSun Workstationに取り込ん

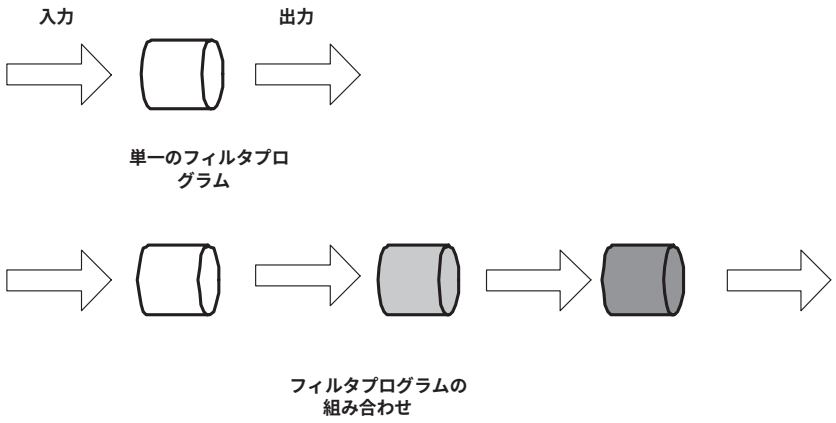

図 7.2: フィルタプログラム

でいる。A/D 変換装置は 1 ボードあたり 256 チャンネルに拡張され、2 ボード 512 チャンネルデータが収集可能である。このシステムはさらに RDB と Web を統合させ、また、ユーザアプリケーションとして、Java アプレット を採用することで、OS に依存せず、直感的に分かりやすいインタフェイスを提供した。RDB を用いることによって、将来チャンネル数が増大しても容易に拡張することができる。また、Web 技術を用いたため、PC をクライアントにすることができ、JDK 1.1 をサポートした Web ブラウザであれば使用するコンピュータに依存せず、離れた場所からデータの取得、表示、A/D 変換器の制御が可能である。

このシステムにおけるデータの流れを以下に示す。

1. LHD 真空容器に取り付けられたセンサから得られた信号は、増幅され Sun Ultra Sparc 上の S-BUS の A/D 変換器によりデジタル信号に変換される。この時のサンプリングレートは最大 100kHz であり、このサンプリングレートは後述するクライアントアプレットにより変更できる。

2. デジタル変換されたデータは、A/D 変換器によるサンプリングをそのまま適用した「高速データ」と、この高速データを移動平均によってサンプリングレートを〜1Hz に落とした「低速データ」の二種類が存在する。主として高速データはコイルの励磁実験やプラズマ放電実験等、変化の速い実験を参照するのに用いられ、低速データはコイルの保全性を保つために、真空容器の温度変化など比較的ゆっくりした変化のデータを参照するのに用いられる。さらにこの二つのデータはセンサの出力電圧そのものの生データと、このデータをもとに物理量に変換した物理データとに分類できる。生データから物理量への変換はデータベース内に保存されたセンサ毎の校正テーブルを利用して行っている。物理データはあらかじめ決められた許容範囲内にあるかどうかチェックされ、もしこの範囲を超えて

いた場合には、担当者に電子メールや電話等で異常があったことを通知する。

3. 低速の生データおよび物理データは常にハードディスクに保存しているが、ファイルサイズの関係から12時間毎に分割している。これらのデータは同時にIPマルチキャストによりネットワーク上に配信している。一方、高速データは512チャンネルすべてを取り込むと物理データで約390MB/sに達するため、ネットワークの帯域を越える。さらに、ハードディスクの容量が限られているため、対象となるデータを絞り、トリガにより収集の開始・終了を制御している。トリガはコンピュータ側から制御するソフトトリガと、特定のシグナルをTTLで受けることにより収集を開始する、ハードトリガが利用可能である。ハードトリガはクエンチシグナル検出時などに用いられる。両トリガともポストトリガが設定可能で、この時は取り込み開始時刻よりも前に収集されたメモリ内のデータを保存することが可能である。

4. マルチキャストで送られた低速データは、Webサーバ上のデーモンで受信する。データの表示等を行うクライアントプログラムはJavaアプレットとして提供されており、Webサーバからダウンロードされたデータ表示アプレットにより過去データおよび、リアルタイムデータを参照することができる。過去データに関しては、CGIにより過去データを適当なデータ間隔でよみとり、テキストフォーマットに変換し取得することができる。一方リアルタイムデータは、Java RMI [1]によってWebサーバ上のデーモンと交信し取得し、リアルタイムでグラフ化を行う。また、高速データの取り込み開始等などA/D変換に関する制御もRMIによる通信でおこなわれる。ここで使用されているJavaプログラムはJDK1.1で作成されており、基本的にはJDK1.1をサポートしたWebブラウザであれば全て実行可能なはずであるが、上に示したようにサーバとの通信にはRMIをサポートしている必要があり、これをサポートしていないMicrosoft Internet Explorer等は使用することができない。このため、平成14年現在使用可能なWebブラウザはNetscape 4.7xおよびHotJavaブラウザに限られる。

5. また、センサの取り付け位置や校正表等のデータはWebサーバ上で動作しているRDB Sybase System 11により管理され、これもJavaアプレットにより参照・修正等が可能となっている。データベースとの通信はJDBCを使用している。

このようにシステム自体には遠隔地から利用を行うための機能が備えられているが、実際にはこのシステムを外部から利用することはできない。これは、当システムが実際の計測機器の制御を行うため、先に挙げた解析情報サーバよりも、よりセキュリティの高いネットワーク上に置かれているためである。

[1]Remote Method Invocation。異なるPC間でメッソドを呼び出す技術

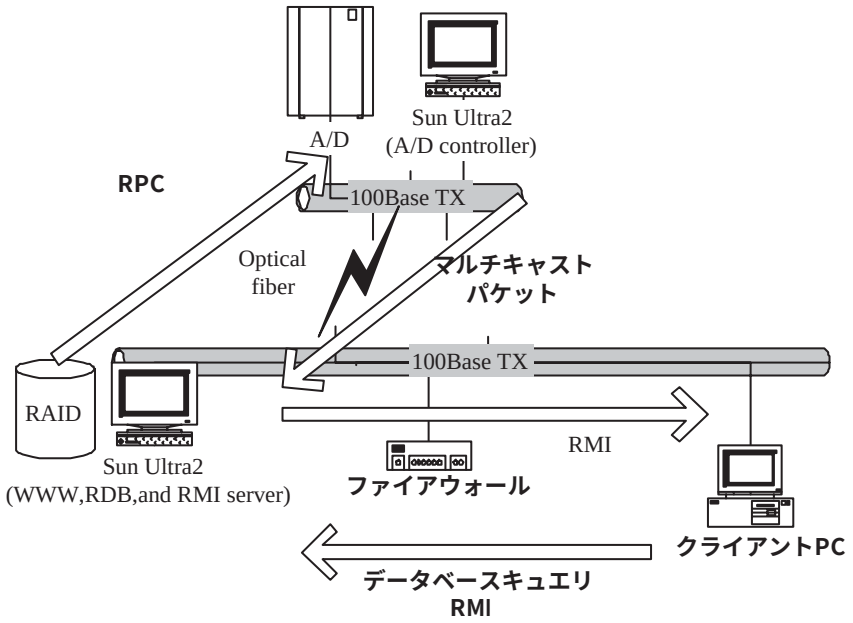

図 7.3: リアルタイムモニタリングシステム概要

7.3 三次元モニタリングツール nifsBrowser

前節で紹介した LHD モニタリングシステムにより、計測データの様子を Web ブラウザを利用し、遠隔地からモニタリングすることができるようになった。この節では、この機能をより進めたものとして、Java3D を使った 3 次元可視化技術を使った三次元モニタリングツール nifsBrowser を紹介する [34]。このツールは三次元情報を扱うことで、複雑なデータをより直感的に理解できることを目指したものである。

7.3.1 リアルタイム監視

LHD モニタリングシステムにより、計測データに異常が認められた場合、携帯電話等でそのことを通知することが可能となった。ところが、LHD のような大型かつ複雑な装置の場合、異常があったことが分かっても、その場所を特定することが難しい。このような複雑な機械の形状を直感的に理解するためには、3 次元モデルを使用した方法が有効である。そこで、Java3D を使った 3 次元リアルタイム監視ツールの開発を行った。このツールは、現在および過去の温度をその値に応じ色分けし、3 次元表示させた LHD 真空容器の上のセンサが取り付けられた位置

に重ねて表示させる (図 7.4)。ユーザはマウスまたはキーボードにより、真空容器のモデルを回転・拡大・縮小させることができ、真空容器の状態の把握が容易にできる。また、異常があったセンサは注意色 (デフォルトで赤) で示され、それが容器のどの部分であるかをすばやく探し出すことができる。Java3D を採用したのは下記の理由による。

1. 特定のプラットフォームに依存しない
2. 既存のリアルタイム監視システムが Java で作られているため親和性が良い
3. プログラムが比較的簡単

nifsBrowser はすべて Java (JDK 1.2) を使用して作られており、基本的に JDK1.2 をサポートする環境であればプラットフォームに依存せず、動作させることが可能である。動作テストは、MS Windows NT/2000,Solaris 2.6,Linux (表 7.1) で行った。ただし、開発時点において Linux 用の JDK1.2 および Java3D1.1 はベータ版であり不具合が目立った。

MS Windows 2000/NT では画面の書き換えが静止状態で 12-15 フレーム／秒、マウスによる図形の回転時で 2 フレーム／秒と良好なレスポンスを示した。一方 Solaris では静止状態で 2 フレーム／秒、Linux では 0.2 フレーム/秒だった。Solaris は使用したワークステーションが旧式のものであり、十分な CPU の能力が得られなかったためだと思われる。一方 Linux をインストールした PC は MS Windows のものと能力的には同等であるが、使用した X サーバがハードウェア 3D アクセラレーション機能をサポートしていないためだと思われる。使用した X サーバは 3.3.6 であるが、4.0 からはハードウェアアクセラレーション機能がサポートされるため、Linux においても実用になるもとのと思われる。平成 14 年 10 月現在では、ローエンドクラスの PC ですら、ここに示した PC よりも性能が上回っており、快適な操作性を有する PC は比較的安価に手に入れることが可能である。今後、CPU やグラフィックスの高機能化、Java3D 等の開発環境の充実により 3 次元を利用した可視化技術は垣根の低いものになってゆくと思われる。‘

7.3.2 データ可視化

一般に LHD 実験で計測されるデータは 1 パラメタ+1 物理量の 2 次元グラフで表現されるものはむしろ例外で、殆どは 3 次元以上の自由度がある。このため、これらの計測データを NIFScope で 3 次元以上の実験データを可視化するために、特定の 2 次元切断面の表示を行うことで可視化を行っている。しかしながら、この方法では直感的な理解が難しい。このような多自由度の計測データ可視化には、3

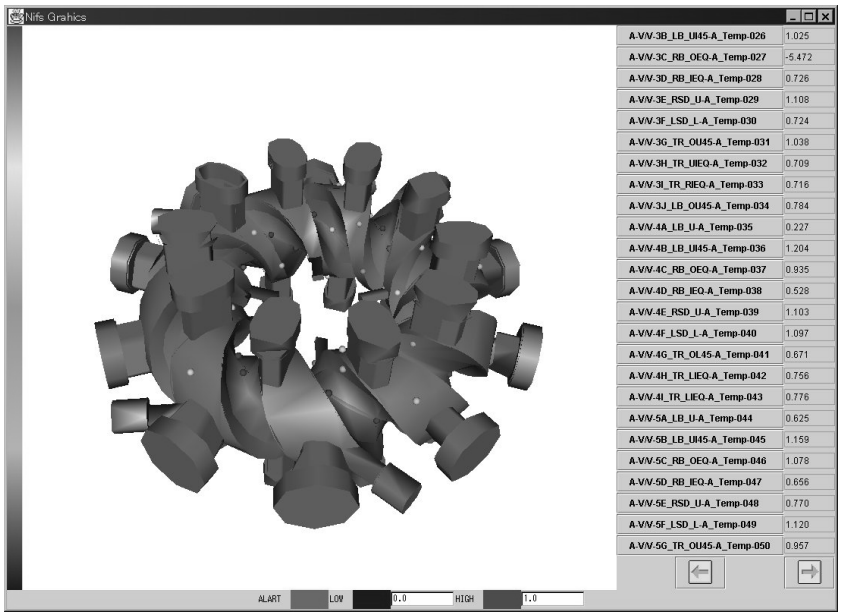

図 7.4: nifsBroser 実行画面

OS	Solaris	MS Windows	Linux
Version	5.5.1	2000	Kernel 2.2.16
			XFree86 3.3.6
CPU	Ultra Sparc 166MHz x 2	Pentium III 800MHz x 1	Pentium III 500MHz x 1
メモリ	256MB	512MB	512MB
ビデオカード	Creator 3D	Canopus Spectra 8400	MatroxG400
解像度	1280x1024	1280x1024	1024x768
色数	24bit	32bit	24bit
JDK	1.3	1.3	1.2.2
Java3D	1.2	1.2	1.1.3

表 7.1: 使用した PC の仕様

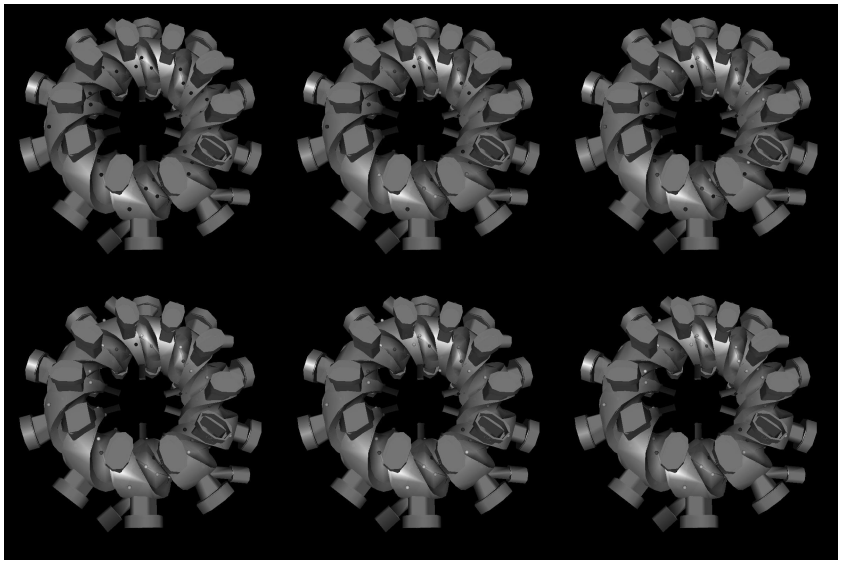

図 7.5: 真空容器の温度変化: 左上:NBI 入射直後、中上:1 秒後、右上:2 秒後、左下:5 秒後、中下:10 秒後、右下:20 秒後

次元での可視化技術が有用である。ここではnifsBrowserを使用し、データの可視化に利用した。

図 7.5 は、入射パワー 2.1MW の NBI 入射 0〜20 秒後の真空容器の温度変化を示した図である。図中で青色は温度上昇が 0 度以下のもの、赤は 2.0 度以上温度上昇があったことを示す。この図から、温度上昇は NBI 入射口付近から始まり、コイルに沿ってヘリカル状に広がっていることが読み取れる。このことから、ビームが直接真空容器の壁面にあたらず、磁場に沿って回っていることがわかる。温度測定に使用したセンサは 100 個であったが、センサの数を増やすことでより精密な測定が可能となる。

7.4 プロキシサーバ

リアルタイムモニタリングシステムは実際の実験装置を扱うため、セキュリティのより堅固なサブネットワークに置かれ、他のネットワークから孤立している。このため、他のネットワークに繋がった PC からは利用できない。一方、外部のネットワークから利用できるようにという要望が高まったために、外部からもアクセスできるように対策を行った。また、システムの開発にあたって、稼動中のシステムへの影響を避けるため、既存のプログラムの変更は最小限にとどめるように

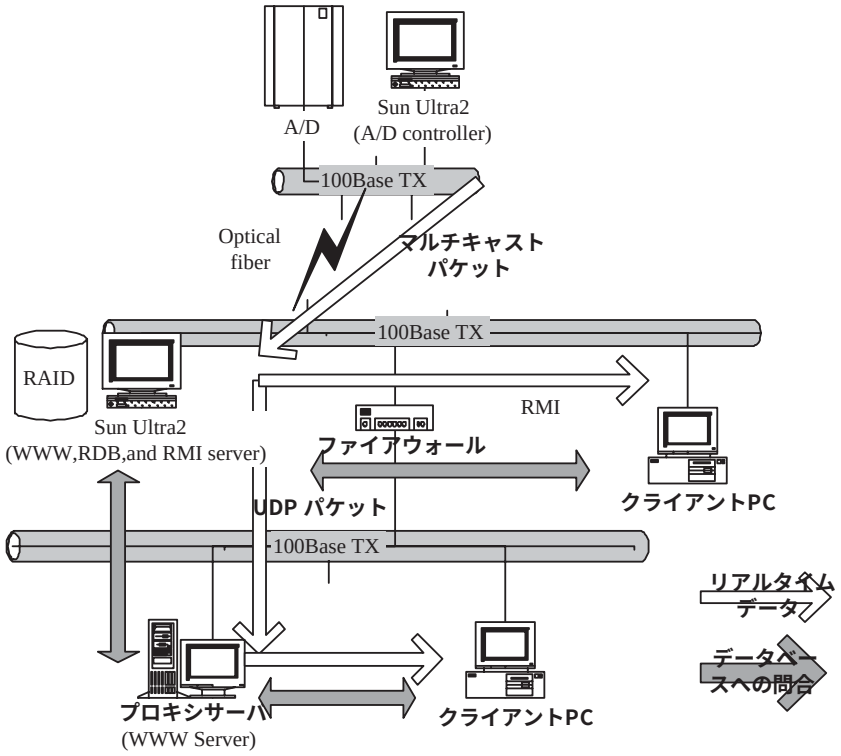

図 7.6: プロキシサーバ

した。

ルータの外側からアクセスを行うため、新たにプロキシサーバを配置した (図 7.6)。セキュリティ確保のため、ファイアウォールの内側へのアクセスはこのプロキシサーバから特定の TCP ポートを通してのみできるように設定した。ファイアウォールの外部のクライアントは、このプロキシサーバに対しアクセスを行う。プロキシサーバがオリジナルのサーバと同じように振舞うよう次節のアプリケーションの開発を行った。オリジナルのサーバは Sun Solaris 5.6 で運用されているが、プロキシサーバは PC Linux を使用した。これは、プログラムの大部分が Java で書かれているため、OS の違いによる変更は最小限で済ませることができたことが大きい [33]。

7.4.1 ダミーデータベースサーバ

クライアント PC はプロキシサーバ上で動作している Sybase OpenServer [35] を実装したダミーデータベースサーバに問い合わせを行う。OpenServer はデータ

ベースサーバインタフェイスを提供する API ライブラリであり、これを実装したアプリケーションはクライアントからは通常のデータベースのように見える。ダミーサーバはクライアントから問い合わせがあった場合、自身での処理は行わず、ぞの内容をそのままオリジナルのサーバ上の Sybase System 11 に転送し、実行結果を返すという動作を行う。現時点において、接続元は研究所内に限られているため、データベースのクエリをそのまま通過させているが、オリジナルのデータベースへのアクセスは別のアカウントを使って接続を行うことができる。この機能を使えば、オリジナルのデータベースへの機能追加を行わずに、外部からアクセスを権限の低いユーザにマッピングを行うことができるため、セキュリティの低い外部からのアクセスを安全に行うことができる。研究所外からの接続を許可する場合にはこのような機能が必要となるであろう。

7.4.2 マルチキャスト中継デーモン

リアルタイムでデータをファイアウォールの外側に送るため、新規にプログラムを作成した。このプログラムはリアルタイムデータであるマルチキャストパケットを一旦 Web サーバ上で受け取り、その内容を UDP のユニキャストに用いてプロキシサーバ上に転送するものである。これは Web サーバ・プロキシサーバ間のルータがマルチキャストに対応していないための処置である。

ユニキャストのデータを受信するプログラムは、Web サーバ上でマルチキャストをデータ受け取りクライアントにデータを送信する既存のプログラムを修正したもので、クライアントからは既存のプログラムと全く同一に見えるため、クライアントプログラムは変更を行うことなく、プロキシーサーバを利用することが可能である。

第8章　IPマルチキャストを用いた情報の共有

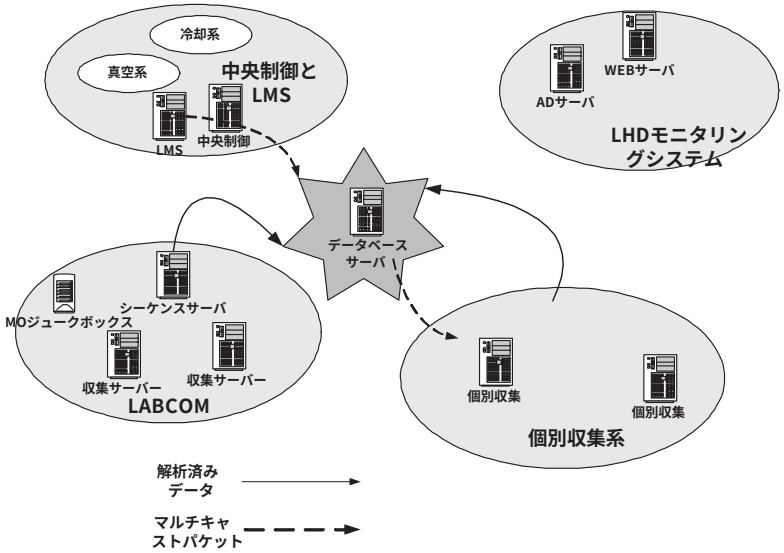

図 8.1: IP マルチキャストによる情報の共有

LHD では多様なシステムが混在するため、個々のシステムでの情報共有を行う方法が必要となる。例えばショット番号は個々のプラズマ放電実験を特定する上で最も基本的な情報で、各収集システムは現在行われている実験のショット番号を把握する必要がある。ところが、ショット番号を管理している LMS は他から独立したネットワークにあるため、直接参照することができず、共有ファイルにデータを書き込み参照するという方法が持ち入れれてきたが、この方法にはショット番号の更新を認識するため、ファイルの更新をポーリングにより監視する必要があり、サーバ負荷等の点で問題が発生していた。

また、プラズマの繰り返し放電実験では、前回収集したデータを元にゲインの調整等を行う必要があり、登録されたデータを即時に参照することが要求される。このためには、データが登録されたことをサーバの側から通知する方法が必要と

なる。

この章では、IP マルチキャストを用いた情報の共有について説明を行う。この方法では、サーバの側からクライアントに通知することができるためににリアルタイム性に優れている。また、IP マルチキャストはユニキャストに比べクライアントを管理する必要がなく、サーバにかかる負荷はクライアント数に依存しない利点があり、計測技術の発達や研究テーマの変更等により、コンピュータの数やその構成が常に変化している核融合科学研究所のような環境では、ソケット通信に比べ適している。また、この方法はサーバの側からクライアントに情報を通知することができ、

8.1　マルチキャストパケットフォーマット

各々のシステムで使用される IP マルチキャストのフォーマットは付録 D に示す。それぞれのパケットには 8 バイトの共通ヘッダがあり、ここには識別子とパケットサイズが記録される。この仕様により、将来新しい用途に IP マルチキャストを利用したとしても、識別子を換えることによりパケットの種類を判別ができるため、複数の IP マルチキャストを使ったシステムが混在することが可能であり、今後の拡張にも対応することができる。このようなフォーマット仕様を用いることで、同一マルチキャストアドレスを使用し、複数のサービスを行うことが可能である。しかしながら、同一アドレスを使用すると、そのサービスを必要とするクライアントがいない場合でも、別のサービスを要求するクライアントがサブネットに存在する時は、ルータがそのサブネットにパケットを転送してしまう。無駄なパケット転送を避けるためにサービス毎に異なるアドレスを使用して運用を行っている。

8.2　ショット番号のマルチキャスト配信

1 章で述べたように、LMS のデータベースに直接アクセスできないシステムからショット番号を参照するために、テキストファイルを用いた方法が利用されていたがこれには次のような問題が発生した。

- MS Windows 以外のプラットフォームへの対応
- 負荷の集中
- ファイルロック

これらの問題を解決するために、ショット番号を IP マルチキャストで通知するシステムの構築を行った。IP マルチキャストは特定の OS に依存する技術ではな

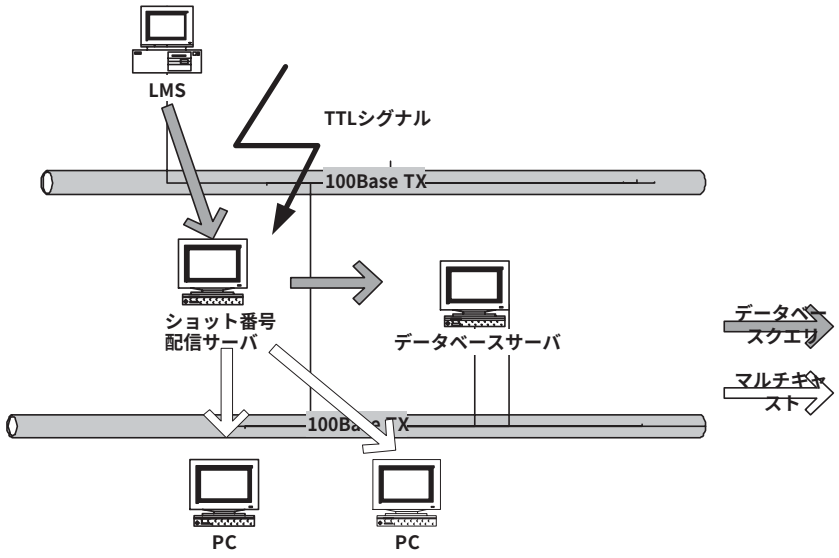

図 8.2: ショット番号マルチキャスト配信

いため、MS Windows 以外の OS でも使用することが可能である。また、IP マルチキャストはサーバからクライアントに一方的にデータが送られるため、複数の PC がファイル等の共通資源に同時にアクセスすることがない。このため、テキストファイルを用いた場合に生じたようなファイルのロックと言う問題は生じない。また、クライアント数が増加しても、サーバにかかる負荷は変わらない。

このシステムでは、中央制御システムが各システムに配信するシーケンスシグナルを光ファイバで中継し、TTL シグナルに変換した後、ショット番号配信サーバのパラレルポートへ入力している。受信するするシグナルは、放電開始 2 分半前および放電終了の二つである。パラレルポートに入力レベルの変化があるとハードウェアの割り込みが発生し、その割り込みを受け取ったデーモンはその時点におけるショット番号を LMS の実験データベースに問い合わせる。デーモンは検索したショット番号とシーケンス状態をマルチキャストで転送する。

この情報を個別システムで受信するためにユーザの多い MS Windows に対しては、DLL(ダイナミックリンクライブラリ) を提供した。このライブラリでは、getMCPacket という関数を提供する (図 8.3)。この関数は指定された時間だけ待機し、その間にショット番号のマルチキャストパケットを受信すると実験ステータス (シーケンス番号)、ショット番号、サブショット番号を返す。この関数の返値は、パケットを受信した場合正、タイムアウトすると 0 となる。

```
//NAME: getMCPacket
// RETURN CODE : 0 : TIMEOUT, NON ZERO : SUCCESS
// PARAMETERS
//(OUT) int* seq : sequential status
//(OUT) int* shot : shot number
//(OUT) int* sub  : sub shot number
//(in) int timeout : timeout
int (WINAPI *getMCPacket) (int *seq,
           int *shot,
           int *sub,
           int timeout);
```

図 8.3: マルチキャスト受信関数

8.3　解析済みデータ登録通知システム

通常 LHD によるプラズマ放電実験は 3 分間周期で繰り返される。放電中に収集されたデータは、次回の放電パラメタや機器のゲイン設定などに反映する必要があり、解析済みデータは放電終了後すぐに参照することが求められる。また、一部のデータは、他の解析済みデータを元データとして、より高次の解析を行って作成されるため、その解析済みデータが登録されたことを知る必要がある。しかし、個々の計測データが登録されるタイミングは計測毎に異なるので、最新の解析済みデータを取得するためには、そのデータが登録されたかどうかをデータベースに問い合わせを繰り返す必要がある。ところが、この方法を用いると、新規にデータを登録するクライアントと、そのデータを参照したいクライアントのアクセスが、放電直後に集中するためネットワークやサーバに負荷がかかり、データの取得や登録に遅延を来たす恐れがあった。

新規データ登録有無の確認に伴うデータベースへアクセス集中を解決するため、データベースが登録された時点で、サーバ側からクライアントに登録されたことを通知するシステムの開発を行った。これにはデータベースのトリガを利用した。トリガはデータベースの変更 (行の挿入・更新・削除等) があると起動されるプログラムであり、このトリガとして、解析済みデータが更新される毎にマルチキャストで変更のあった計測名とショット番号を送信するプログラムの実装を行った。マルチキャストパケットの受信には、ショット番号配信システムと同じ DLL を提供しており、ユーザはこのライブラリを利用することで、更新された計測データの種類を知ることができる。解析済みデータ登録通知のサービスを利用した例と

して、5 章で紹介した NIFScope がある。NIFScope では表示するグラフの対象として、最新ショットが選択できるようになっているが、これはマルチキャストデータを受信した時点で最新データを取得することで、グラフの自動更新を行うことが可能である。現在、一部の代表的なデータは放電直後に中央制御室の大パネルに表示されており、同時に PDF および PS ファイルに変換されて WEB から参照できるようになっている。しかしがら、それ以外のデータは、ここからは利用することはできない。NIFScope の自動更新機能を用いることで、ユーザは自分の PC 上で必要なデータの最新情報を参照することが可能である。

第9章　長時間放電用リアルタイム監視システム

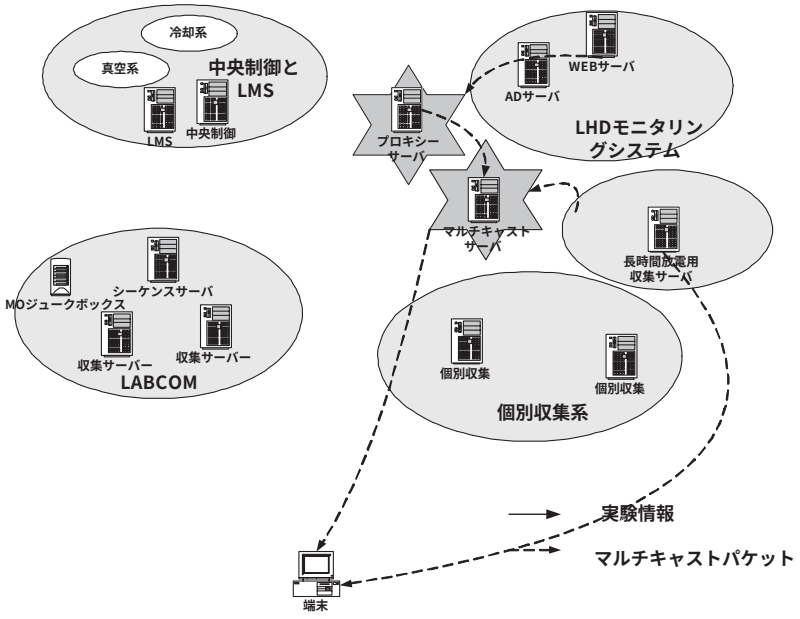

図 9.1: 長時間放電用リアルタイム監視システム

核融合炉の実用化のために長時間のプラズマ放電が必要とされる。ところが、長時間放電では従来の短パルス放で用いられてきたバッチ型のシステムでは対応することができず、リアルタイムでの処理システムが必要となる。この章では、前章で述べたマルチキャストの技術を用いた長時間放電用のリアルタイム監視システムについて述べる。

9.1　長時間放電実験と既存システム

LHD の放電実験はこれまで、短パルス放電の繰り返しを主としたものであった。これは、10 秒程度の短パルス放電を 3 分間周期で繰り返すというものである。このような短パルスの繰り返し放電は、現在の核融合研究では主流のものであるが、一方、実用炉の開発には、高温プラズマの定常運転が必要であり、少なくとも 1 時間以上の放電が必要である。このため、各国研究所では長時間放電のための研究に重点を置き始めている。例えば、トカマク型の実験装置では、九州大学の TRIAM-1M における 2 時間放電 [44] やフランスの Tore Supra における 2 分間放電の例 [45] がある。核融合科学研究所でも第 4 サイクルからは、より長時間 (〜3 時間) の放電を目指し実験が行われた。ところが、LHD 実験で使用されている計測システムの多くは短パルスの繰り返し放電を想定したものであり、長時間放電実験にそのまま適用することはできない。例えば、LABCOM システムは中央制御システムより送られてくる放電サイクルに応じた TTL ベースのシグナルに同期しており、放電 2 分半前にデータ取り込み準備を開始し、放電開始シグナルを受信するとデータの取り込みを開始し、CAMAC 内のバッファが一杯になるとデータベースにデータを転送するという動作を行っている。このため、このシステムをそのまま適用すると、長時間放電では、最初の数秒間でバッファが一杯になり、残りのデータが取得できなくなる。このための対応策として、次の方法が考えられる

1. サンプリングレートを落とし、長時間放電中のデータを全て取得する
2. 特定のトリガにより必要な所のみデータの取得を行う
3. 放電中にシーケンスを繰り返す

1 の方法は最も簡単に実現できる方法であるが、CAMAC に搭載されているメモリが 1M word であるため、10000 秒の放電すべてデータを取得するためには、100Hz 以下にサンプリングレートを落とす必要がある。また、この方法をとった場合は放電終了までデータを参照することができない。2 の方法の欠点は、各計測器間で同期が取れないという問題が生ずること、また、測定対象となるイベントの検出システムが必要となる点である。既存システムへの影響を考慮し、最終的に採用されたのは 3 の方法である。中央シーケンスから送信されるシーケンスシグナルの代わりに擬似シグナルを送ることで既存のデータ収集システムで対応することが可能となった。ただし、この方法の欠点は放電中の一部のデータしか取得できないことがあげられる。全ての方法において避けられない欠点として、バッファにデータがたまるまで、データを参照できないという問題があり、このためリアルタイムでデータを参照することができない。長時間安定したプラズマを維

持するためには、マニュアルによる機器の微調整が必要となる。このためには、計測結果をすぐに参照できる必要があるが、上記の欠点からこの目的で CAMAC を使用することはできない。また、プラズマ計測用のレーザ干渉計等で使用されるレーザは〜3 時間の連続稼動を行う必要があるが、これらのシステムは数十秒程度の連続稼動の実績しかないため、安定運用のため計測中の温度等を監視する必要がある。これらの目的のためには、CAMAC のようなバッチ処理ではなく、リアルタイムでデータ収集を行える計測系の利用が必須である。

この候補としては、WE7000 が検討された [46]。WE7000 は PC ベースの計測システムで、各種 A/D 変換器やデジタイザ等や、それらを制御するソフトウェアがモジュール化されており、これらを組み合わせることで、拡張性や柔軟性が高いのが特長であり、プログラムを作成することなく、これらの機能を組み合わせることにより、データの保存やリアルタイム表示を行うことができる。このため、リアルタイム監視のために、WE7000 が利用されている。しかしながら、想定している利用法から外れた場合には、提供されているコンポーネントだけでは不可能であり、プログラムを組む必要がある。したがって、長時間放電時に多数の研究者が現在計測しているデータを参照するという目的を達するには、新たにプログラムを開発する必要があった。このため、平成 13 年度の実験では、他の研究者のために PC の画面の出力を制御室中央の大画面に投影することで対応したが、この方法では、制御室にいる研究者以外が参照できないため、計測されたデータそのものを他の PC から参照できる方法が求められていた。

9.2 システム概要

システムの概要を図 9.2 に示す。測定対象となるデータは大きく分けて二つあり、LHD 装置の安定性と、プラズマ物理に関する観測データである。前者は、LHD リアルタイムモニタリングシステムを拡張したものである。LHD リアルタイムモニタリングシステムはデータ収集サーバ・Web サーバ間はマルチキャストを使用しているが、実際にこのデータを参照するクライアントは Web サーバ、あるいはミラーサーバと一対一で接続する必要があり、負荷が両サーバに集中するという問題があった。今回の拡張では、新たにマルチキャストサーバを設置し、マルチキャストサーバが Web サーバ上のデータを IP マルチキャストで転送することによって、この問題を解決した。

プラズマ観測に関するデータは CompactPCI および、PCI ベースの A/D 変換器を搭載した Linux ベースの A/D 変換サーバ上でサンプリングしたデータを、直接マルチキャストでネットワーク上に配信するシステムであり、今回新たに開発したものである。CompactPCI はピンソケット式のコネクタを使用し、PCI のカー

ドエッジコネクタよりも信頼性が高い。また、衝撃や振動に強い等、実験のモニタリング等のように、コンピュータにとっては劣悪な環境下で連続稼動させる場合等に適している。今回は、必要なデータの一部が、特に LHD 実験装置に近接し、各種計測シグナルの集中する場所に設置する必要があることから、CompatPCI の採用を行った。この A/D 変換ボードをコントロールするために、CompactPCI のボード PC を用意し、ラック内に格納することで、他の計測機器への電磁的な影響を抑えた。また、故障の原因となりうる稼動部分や発熱を押さえるため、OS および収集アプリケーションはコンパクトフラッシュメモリにインストールし、HDD は使用していない。一方、通常の実験室で受けることのできる信号については、通常の PCI バスベースの PC を設置した。長時間放電時に必要とされる計測データを表 9.1 に示す、また、両 A/D 変換ボードの仕様を表 9.2 に示す。

このシステムの独自の特徴として、マルチキャストで送信されたデータをフィルタプログラム によって加工し、より高次の物理データを作成することができる点が挙げられる（図 9.3）。マルチキャストパケットを受けたマルチキャストサーバは、登録されたフィルタプログラムを呼び出し、フィルタプログラムの標準入力にパケットのデータを書き込む。フィルタプログラムはこのデータを元に計算を行い、標準出力から結果を出力し、マルチキャストサーバにデータを渡す。これらのプログラムは他のシステムからまったく独立しており、ユーザは C や FORTRAN 等の汎用言語及び PV-Wave や IDL 等の簡易言語を使って自由に作成することができるため、拡張性が優れていると同時に多くのユーザに快適な環境を提供できる。

このように単一機能のプログラムを組み合わせ、より複雑な処理を行うという考え方は UNIX におけるフィルタプログラムとして広く使われているものであり、刈谷らは、この UNIX のフィルタという考えを実験のモニタリングに採用することにより、柔軟なシステムが作られることを示した。筆者らは、この考えを発展させ、分散化を徹底した。刈谷らの実験では、A/D 変換を行っているワークステーション上で最終的にモニタリングに必要なデータまでの変換を行ったが、本システムでは、A/D 変換器によるシグナルデータ取得と物理値の変換部分を分離した。これは、ワンボードコンピュータや組み込み機器等のディスクや CPU パワーの限られたシステムの利用を想定したものである。データ収集部分は、A/D 変換器によって収集されたデータをそのままネットワーク上にマルチキャストにより送信するだけなので、複雑な処理を必要としない。さらに物理値への変換部分に対しては、ユーザが自由に変更できるようモジュール化を行うことで、高度に専門知識を必要とする物理変換部分ロジックを専門家であるユーザに任せることができるようにした。このような改良点により、より柔軟なシステム構築を可能にした。

リアルタイムのデータ表示には、NIFScope を使用した。NIFScope 内蔵の Ruby により単純にマルチキャストで送られてきたデータを表示するだけではなく Ruby

計測名	設置場所
密度 (FIR)	計測室
電子温度	計測室
全放射損失 (Prad)	CAMAC 室 (シールドルーム)
不純物ライン (CIII、OV)	CAMAC 室
中性原子ライン (Ha、HeI)	計測室
ガス熱々 (PFIG)	計測室
ダイバータ粒子束	CAMAC 室
カーボンタイル温度	CAMAC 室
真空容器温度	CAMAC 室
ICRF 電圧	CAMAC 室
Zeff	CAMAC 室

表 9.1: プラズマモニタリングに必要な項目

で記述されたプログラムにより、リアルタイムで計算を行いグラフ表示することができる。また、データをロードする部分を Ruby のモジュールとして提供しているため、新しいデータフォーマットに容易に対応することが可能である。今回は、リアルタイムでのデータ表示のために、新たにリアルタイム表示用のモジュールを作成し、定期的にデータを更新する機能を追加することにより対応を行っている。

	aPCI8815	PCI3135
バス	CompactPCI	PCI
チャンネル数	差動 8ch(シングル 16ch)	差動 8ch(シングル 16ch)
チャンネル切り替え方式	マルチプレクサ	マルチプレクサ
切り替え時間	45 μs (切替+A/D)	2 μs
アクセス方法	I/O	32KB FIFA バッファ
サンプリング周期	12 μs	1 μs
分解能	12bit	12bit

表 9.2: A/D 変換器の仕様

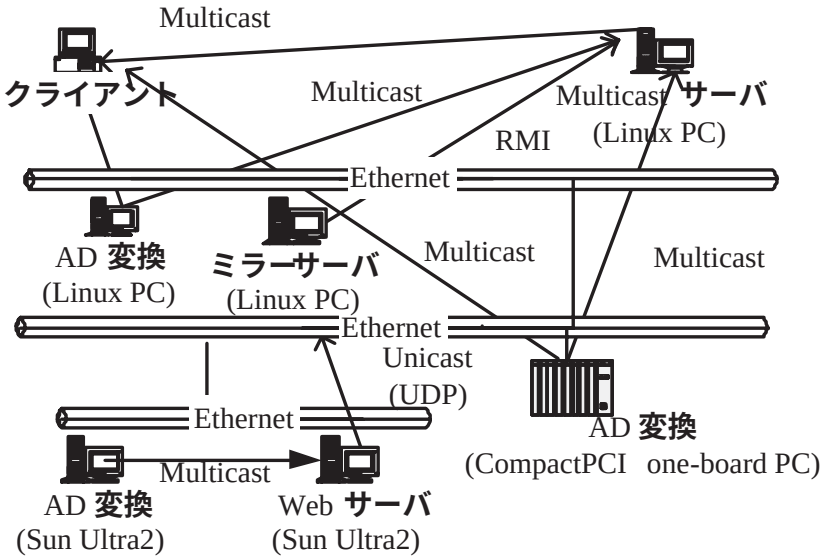

図 9.2: 長時間放電リアルタイム監視システム概要

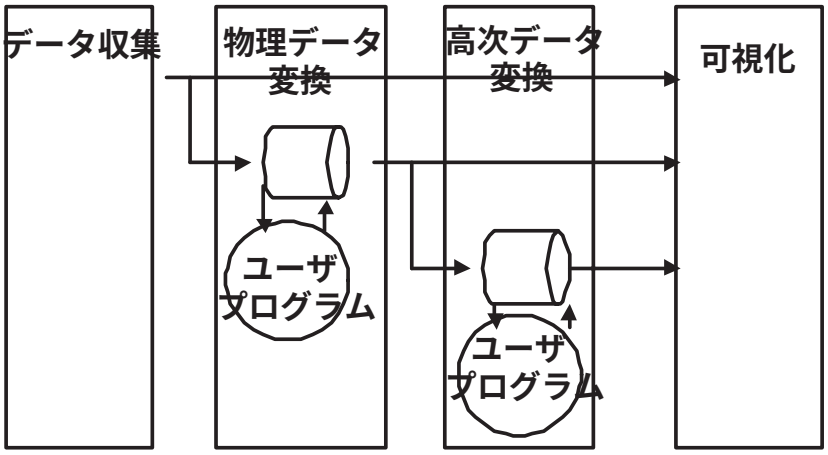

図 9.3: フィルタプログラム

9.3 パケットフォーマット

従来の短パルス放電を対象としたバッチ処理システムでは、各計測器の計測開始は放電開始時刻に一致していたので、放電実験を特定すれば複数の計測間で時刻を揃えることができた。ところが、長時間放電では、本システムのような連続測定を行うシステムと、9.1 節で紹介したように擬似シーケンスによって短パルスモードの運転を繰り返すシステムが混在する。

連続データを切れ目なく収集するシステムでは、他の収集データと比較するために、データに絶対時刻を埋め込む必要がある。本システムでは、表 D.2 に示すように、長時間放電で取得されたデータではフォーマット中に基準となる時刻を埋め込むことで、対応を行っている。マルチキャストで送信されるデータの相対時刻は nsec オーダーで記録できるようになっており、一方、基準時刻に関しては秒の精度である。複数の計測データを比較する場合、重要なのは絶対時刻ではなくむしろ放電開始からの相対時刻なので、基準時刻はこの精度で十分である。

使用している A/D 変換ボードはデータの取り込み開始シグナルとして、外部トリガを受け付けることができるが、長時間放電ではデータの取得は中断なく行われているため、このトリガを使用することができない。このため、LABCOM システムからの擬似シーケンスシグナルを計測データとして取り込み、信号の立ち上がりを監視することにより、ソフトウェア的に基準時刻の更新を行った。

LHD モニタリングシステムでは、1Hz でパケットの送信を行っている。一方、プラズマモニタリングシステムでは、後で物理解析にも使えるよう 256Hz でパケットの送信を行っており、さらにそれを 1 秒間平均したパケットも同時に送信している。手動でグラフを参照しながら、プラズマの制御を行うためには 1Hz で十分だと考えられる。

A/D 変換器の使用しているクロックと PC のクロックを比較すると、プラズマモニタリング用で 1 時間あたり 15 ミリ秒、LHD モニタリングシステム用で、1 時間あたり 30 ミリ秒のずれが生じた。これらのずれは、A/D 変換器の内部クロックを使っていることに起因するためであり、水晶発信子が使われていることを考えれば妥当な数値である。ところが、この誤差は放電時間が長くなるほど大きくなるため、数時間におよぶ放電では無視できない。一方、PC は GPS を利用した stratum=1 の NTP サーバにより同期させており、その分散は 0.35 ミリ秒程度である。また、NTP による誤差は稼働時間に依存しないので、A/D 変換器の内部クロックよりも正確である。このため、PC のクロックによる時刻 (nsec 単位での 1970 年 1 月 1 日からの経過時間) をデータとして記録することにより、A/D 変換器のクロックの誤差を補正できるようにした。

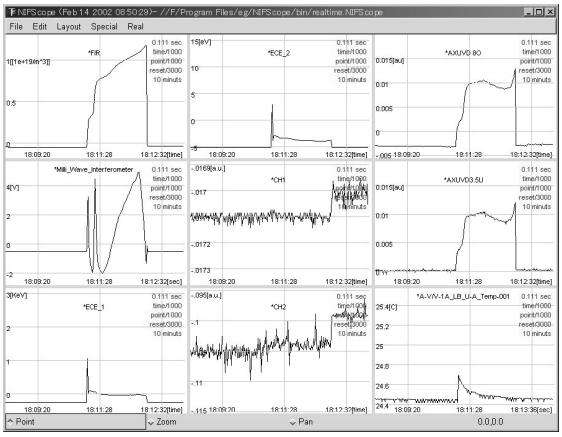

図 9.4: 長時間放電時におけるリアルタイム表示

9.4　新しいシステムの実行結果

図 9.4 が長時間放電実験中におけるリアルタイムでのデータ表示を行ったものである。この実験では、5 分間周期での運転を行ったが、放電時間は最長 100 秒に設定されている。平成 13 年度の実験では、WE7000 を使ったシステムが主に利用されたが、このシステムは、他とは独立したシステムとなっており、WE7000 で取得したデータを他の PC が利用するためには一旦ファイルとして保存し、転送する必要がある。このため、リアルタイム性では長時間放電用リアルタイム監視システムの方が優れている。また、この WE7000 を使ったシステム自体もマルチキャストでデータを送信する機能を追加することで、本システムの一部として組み込むことが可能である。

UDP である IP マルチキャストは一般的に TCP よりも信頼性が低いため、ショット番号の配信やリアルタイムデータの送信の際にパケットの取りこぼしや、パケット順序の入れ替わりが起こる可能性がある。このため、IP マルチキャストで転送されたパケットが正しく送られているかどうかを実際に実験で使用されているネットワークを用いて確かめる必要がある。また、リアルタイムでデータを送るためには、データ送信の遅延時間が問題となる。これには次のようにして測定を行った。A/D 変換サーバからマルチキャストパケットに送信時刻を記録して送信し、これをマルチキャストサーバ上で受信した時刻との差をとる。両サーバのシステムクロックが同期していないため、逆にマルチキャストサーバから A/D 変換サーバへマルチキャストパケットを送信し、この時間差と前者との平均をとる。この結果、送受信に要した時間は、1.45 ± 0.27m 秒であった。また、パケットの送信周期を 1、10、128、1024Hz と変えて 7000 パケットの送信テストを行ったが、パケットの

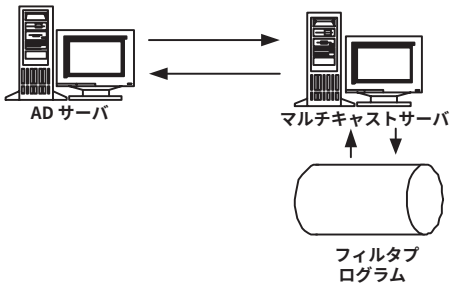

図 9.5: 遅延時間の測定

損失や順序の入れ替わりは観測されなかった。また、長時間放電用リアルタイム監視システムでは、A/D 変換器から直接送信されるパケットの他に、フィルタプログラムを通して送信されるものがあり、これについても測定を行った。A/D 変換サーバからデータを毎秒 1 パケットの速度で配信し、この時データとして送信時刻を μsec 単位で記録し、これを受信したフィルタプログラムが、別の TCP ポートを使用しマルチキャストでデータを送信する。A/D 変換サーバでこのパケットを受信し、パケット内の時刻と取得時の時刻を比較した (図 9.5)。

この結果、A/D 変換サーバ上でパケットの往復に要した時間は 158 ± 3 ミリ秒であった。この結果から、A/D 変換器からのデータを直接参照する場合は、1.45 ミリ秒程度、物理量変換などの処理を加えた場合は、158m 秒+計算時間程度となる。複雑な計算処理を行わないのであれば、A/D 変換開始からデータ参照までに要する時間は高々数百ミリ秒程度で行われると考えられ、長時間放電時における監視目的には十分だと考えられる。ただし、この試験は実験終了時におこなったものであり、ルータに負荷がかかった状況では、異なる結果になることもあり得る。しかしながら、長時間放電用リアルタイム監視システムでは、手動でのパラメタ制御のために計測中のデータを参照することが主目的であり、極小数のパケットの損失は支障がない。さらに、本システムの場合、パケットには時刻が記録されているため、パケットの到達順序が変わっても問題は生じないと考えられる。今回の実験では監視が目的であったため、1Hz という遅いレートでデータを送信したが、計測のためもっと早いレートで送信することも可能である。ただし、現在核融合科学研究所の実験用 LAN で使用されているルーティングスイッチである、アルカテル社の OmniCore 5052 のマルチキャストのパケット転送能力は、IP パケットの転送能力 37Mpps に比べ、20kpps 程度と遅い。このため、スイッチの転送レートがボトルネックになる。実験中に他のシステムに影響を与えないように考慮するならば、送信レートは最大で 1kHz 程度になるであろう。マルチキャストパケットの転送能力が遅い理由は、この機能がソフトウェアロジックで実装されている

ためであるが、現在市販されているスイッチの中にもワイヤーロジックで実装されているものがあり、これらを使用すれば、IP パケットと同程度で送信でき、〜 MHz でのデータ転送が可能になるであろう。

第IV部

遠隔実験参加

第10章　遠隔実験参加

前章までの研究により、LHD コンピュータシステムの統合化を行うことができた。最後のこの章では、統合化されたシステムの遠隔地からの利用について検討を行う。

近年、核融合や加速器等、大型実験装置を使った実験は一国のみでの建設・運用することが困難となり、多国間での協力の必要性が増してきている。研究者の移動という負担を軽減し、円滑な共同研究を進めてゆくためには、遠隔実験参加 (リモート・パーティシペーション) が有用である。本章では LHD 実験の遠隔参加に関し、

- データ参照
- リモートコンピュータアクセス

について研究を行った。

10.1　データ参照

遠隔地から実験参加を行う上で、離れた場所から実験データを参照できるようにする必要がある。ここでは、ミラーサーバを使用し、解析情報サーバシステムを外部からアクセスできるようにしたシステムと、リモートオブジェクトの呼び出し技術を用い、既存の複数のシステムを最小限の変更で外部から呼び出せるようにしたシステムの 2 例を紹介する [32]。

10.1.1　ミラーサーバ

現在本研究所で収集された実験データの提供を行っているコンピュータシステムはいずれも、データアクセスを行うユーザが信頼できるという前提で構築されている。ところが、研究所外の共同研究者が利用できるように、インターネット上で公開した場合、この前提が崩れてしまう。このため、不正なアクセスから保存されたデータを守るためのシステムが必要になる。一つの解として、外部から

のデータを受け付ける代替のサーバを設置することにより、オリジナルのデータを保存しているシステムへの直接アクセスを避ける方法が考えられる。

この節では、解析済みデータサーバの外部公開のために設置したミラーサーバについて説明を行う。元の解析済みデータサーバはサーバの負荷分散を図るため、ファイルサーバとデータベースサーバを分離し、ファイルサーバを複数用意しているが、ミラーサーバではデータベースサーバとファイルサーバを兼用し一台のみで運用する。これは、システムのボトルネックがネットワークであること、セキュリティのためアクセスを一元化するほうが望ましいためである。

セキュリティ

外部からアクセスを許すために、次の変更を行った

1. ユーザ名・パスワードによる認証

研究所内で使われている通常の解析情報サーバシステムでは、匿名でのデータ取得・登録を許しているが、ミラーサーバでは、登録ユーザによるデータ取得しか許可していない。これにより、登録ユーザ以外のアクセスを禁止するとともに、悪意のあるユーザによるデータの改変を防いでいる。

2. データベース操作の制限

同様の理由から、データベースのアクセス制限を行っており、登録ユーザからのデータ参照のみ許可し、非登録ユーザのデータ参照や、登録ユーザによるデータの改変を防ぐ。PostgreSQL 単体の認証方式では、ユーザの成りすましを防ぐことはできないが、研究所外からのアクセスについては Linux の PAM (Pluggable Authentication Modules) を用いたアクセス制限を課すことで、データベースへの書き込み権限を持ったユーザアカウントではデータベースへの接続ができないようにしている。

3. 制限付アカウントの設定

ミラーサーバに対するアカウントは、解析済みデータへのアクセスのみに特化した特別なアカウントに設定している。このアカウントは shell を/bin/true に設定することにより、PC への通常のログインを不可能にしている。このため、通常の passwd コマンドを使ったパスワードの変更は出来なくなるが、SSL [1]による CGI 経由でパスワードの変更を可能としている。

[1]Secure Socket Layer。サーバ・クライアント間の通信路を暗号化することにより安全に通信を行う方法

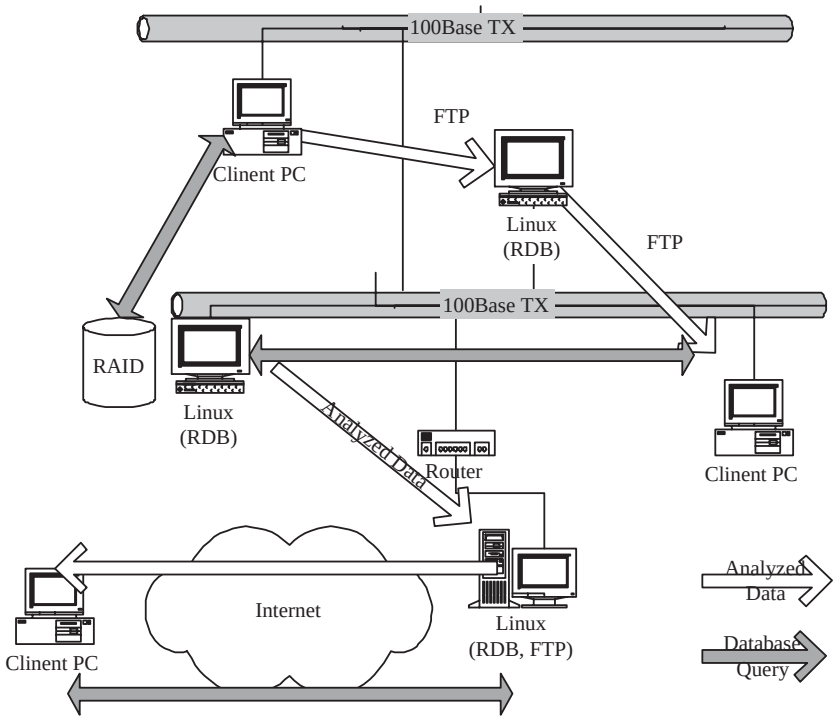
100Base TX
FTP
Clinent PC
Linux
(RDB)
FTP
100Base TX
RAID
Linux
(RDB)
Analyzed Data
Router
Clinent PC
Internet
Linux
(RDB, FTP)
Analyzed
Data
Database
Query
Clinent PC

図 10.1: ミラーサーバ

10.1.2　リモートオブジェクト呼び出しによる実装

遠隔実験参加を行うためには遠隔地からデータを検索・取出す方法が必要となる。このようなアプリケーションを開発する上で障害となるのが、各研究所で使用されているコンピュータシステムが異なる点である。既存のシステムを変更することは、ユーザやアプリケーションの開発者に多大な負担をかける為、できるだけ既存システムを利用する方向でアプリケーションを開発しなければならない。また、バグフィックスや機能追加などによりアプリケーションを再配布する必要があるが、これも不特定多数のコンピュータシステムが共存する環境では困難である。

リモートアクセス技術

米国 Sun Microsystems が開発した Java で開発されたアプリケーションはプラットフォームに依存せず、JRE (Java Runtime Environment) が動く環境であれば動作可能であるため、共同研究のような異種プラットフォームが混在する環境をサポートする目的には最適である。また、Java のアプレットは一般的な Web ブラウザで実行することが可能であり、サーバ接続時にプログラムが自動的にダウンロードされるため、アプリケーションを配布するという手間がかからない。

Java アプレットをクライアント側プログラムとして配布した場合、何らかの方法によりサーバ側アプリケーションと通信を行う必要がある。その方法として、

1. ソケット通信
2. Java RMI
3. CORBA
4. その他

という方法が考えられる。

1. ソケット通信

Java の Socket クラスを使用し、直接サーバプログラムをやり取りする方法である。この方法の利点としては、サーバ側に特別なソフトウェアをインストールする必要がない点、カスタマイズが容易な点である。また他の方法では、最終的にソケットを使うため、直接ソケットを使った方が効率が良いプログラムを書くことができる。一方、欠点としては、コネクションの管理やプロトコルの定義をサーバアプリケーションが自前で行わなければならない点である。これはアプリケーション開発者にとって多大な負担が発生する。

2.JavaRMI

JDK 1.1 以降が備える RMI によりサーバ側の手続きを呼び出す方法である。これは本研究所のリアルタイムモニタリングシステムで使用されている。利点はプログラミングが容易である点であるが、欠点としては Java 間でしか通信が行えないこと、RMI をサポートした JVM(Java Virtual Machine) が必要となる点である。例えば、現在本研究所で使用されている Web ブラウザの内、最も多いのは Internet Explorer である。JDK1.1 の仕様上 RMI はサポートしていなければならない。ところが、Microsoft 社の Internet Explorer 4 以上は JDK1.1 相当の JVM を内蔵しているが、RMI はサポートされていない。

3.CORBA

CORBA (Common Object Request Broker Architecture) [36] は異プラットフォーム、異言語間での通信を行うための技術である。CORBA を使ったリモートアクセスの実装系としては JET で使われているシステムの例がある [37]。CORBA は約 700 社 (平成 13 年 5 月現在) のハードウェアメーカ、ソフトウェアメーカ等が参加する OMG により制定された規格であり、様々なコンピュータ、OS、言語から利用可能である。欠点としては、多種多様なプラットフォームをサポートしなければならないため、仕様が非常に大きくなりすぎたことである。このため、プログラムが困難となり、また、CORBA を動かすために様々なコンポーネントが必要となる。このことは、プログラムのバグを生じやすく、プログラムの保守が大変になる。

4. その他

既存システムとの統合や他プラットフォーム間での通信を考えた場合、低レベルのプロトコルを使った通信よりも、サーバプログラムをオブジェクトとして呼び出すほうが、プログラムの開発やメンテナンスの点で都合がよい。これは、オブジェクト指向によりサーバ側の実装が隠蔽することができるためである。現在アプレットからサーバ側オブジェクトを呼び出す方法として先にあげた JavaRMI や CORBA のほかに、HORB [38] 、SOAP [39] などがある。

HORB は電子技術総合研究所の平野らにより開発された Java ベースの分散オブジェクト環境である。Java によるオブジェクト間通信の実装としては最も古く、JDK1.0 をサポートしている。その他、次のような特徴をもつ

- IDL を必要としない
- Java のみで書かれている

	利点	欠点	Java 以外の言語への対応
Socket クラス	最も効率が良い	プログラムが複雑。プロトコルをインプリメントする必要がある。	可能
Java RMI	プログラムが容易	RMI をサポートした VM が必要	不可
CORBA	有償・無償のプロダクツが豊富。様々な言語から利用できる。機能が豊富	プログラムが複雑になる。様々なコンポーネントを必要とする。	C、C++、Smalltalk 等
HORB	プログラムが容易	開発環境が少ない	可能
SOAP	プロトコルが可読であり、他のアプリケーションから利用しやすい。	SOAP はメッセージフォーマットに関する規格であり、実装は規格化されていない。	C++等

表 10.1: リモートオブジェクト呼び出しの比較

- CORBA IIOP のサポート[2]

- オープンソース

実験的実装

本研究所における既存システムに蓄積されたデータを Web ブラウザから利用するために、実験的システムを構築し、その実行可能性について確かめた。このシステムの特徴は

- 署名つきアプレット
- HORB

を利用したことである。

a) システム概要

図 10.2 にこのシステムの概要を示す。この例では解析済みデータサーバ の問い合わせ、データ取出しを Web ブラウザを通して行うことを可能にしている。Web

[2]CORBA IIOP エクステンションを使用する

サーバよりダウンロードされた署名つきアプレットは、HORB を実装したサーバアプリケーションを呼び出す。サーバアプリケーションは JNI [3]を利用し、解析済みデータサーバのクライアントプログラムを呼び出し、解析済みデータサーバへ問い合わせ、ファイルサーバからのデータ取出しを行う。

b) 署名つきアプレット

Java アプレットを利用する上での問題点は、Java アプレットの実行にはローカルディスクへのアクセスができない、アプレットをダウンロードした所以外のサーバへアクセスすることができない等、様々な制限が課せられることである。これは、プログラム本体をインターネットという信頼できない所からダウンロードするためであるが、これらの制限はリモートアクセスを行う上で障害になる。特にダウンロードしたサーバ以外にアクセスできないという問題は、Web サーバと各種サーバを同一ホスト上で実行することを強いられる。JET の CORBA を利用した例では、GateKeeper というプロダクツを用い、リクエストを各サーバにフォーワードすることにより、リモートの呼び出しを実現しているが、この方法ではサーバへアクセスが集中することになり、負荷の増大は避けられない。

この問題解決のために我々がこのシステムで用いたのは署名つきアプレットを利用することである。アプレットの各種の制限は作成者が不明であるために設けられるものであるが、署名つきアプレットは、アプレットに作成者の「署名」を書き込み、そのアプレットが確かに作成者が作ったものであることを保証したアプレットであるため、安心して実行することが可能である。署名つきアプレットをダウンロードしたユーザはそのアプレットの作成者が信頼に足るものであることを確認し、他のサーバへのアクセス等、一般のアプレットの権限を越えた作業を許可する。

c) HORB

筆者はリモートアクセス技術として、HORB を採用した。これは第一に、HORB の処理系はすべて Java で提供されるため、他に特別なソフトウェアを必要としないためである。この特長により、ユーザは通常のブラウザを用意するだけでアプリケーションが利用可能である。また、CORBA 等に比べプログラムが簡単である点も重要な点である。

このように署名付アプレットおよび、HORB という技術を使って、既存のシステムに影響を与えることなく、遠隔地からのデータ取出しのシステムが構築できることを示した。このシステムは Java を用いることにより、プラットフォームに

[3]Java Native Interface。Java から C++等で記述されたライブラリを呼ぶためのインタフェイス

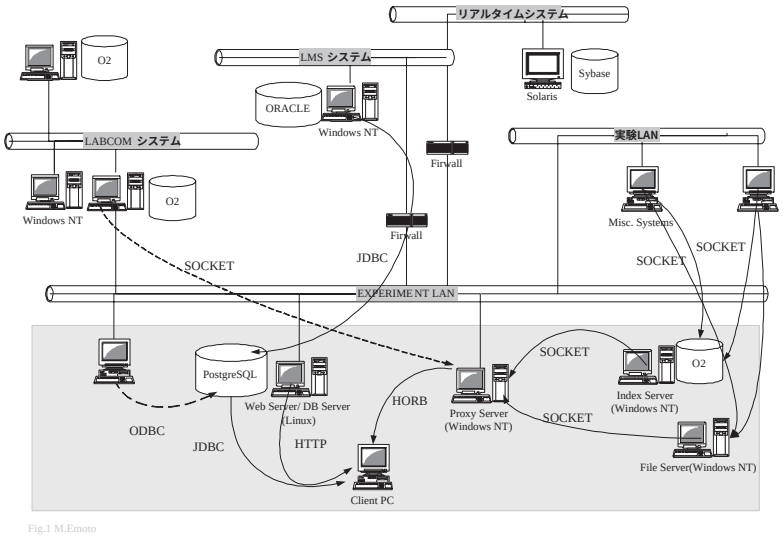

図 10.2: システム概要 (平成 11 年 7 月時点)

依存しないプラグラムを作成することができた。また、Java アプレットであることから、起動の度にサーバからプログラムがダウンロードされるため、クライアント毎のバージョンの不一致に伴う動作の違いや、プログラムの再配布と言ったメンテナンスの労力を軽減することができた。これは、物理的に遠く離れた研究所間でデータのやりとりを行う場合は非常に重要である。

10.1.3　プログラムの実装とインタフェイスの分離

解析済みデータサーバは、各研究グループが共通フォーマットを提供することで、同一インタフェイスにより、複数の解析済みデータを取出し、参照することが可能になった。各国の研究所が同一フォーマットを提供することができれば、データの交換が容易となる。ところが、すでに様々な解析済みデータを提供している各国の研究所で、これらのデータフォーマットをすべて変換するのは、現実的ではない。そこで、解析済みデータのフォーマットにオブジェクト指向を取り入れ、既存データの抽象化を行い、データフォーマットそのものを統一するのではなく、その読出し方法を統一することでデータの交換性を高める。

オブジェクト指向の考え方は移植性を考える場合は非常に有効である。オブジェクト指向とは C++や Java 等のオブジェクト指向型言語を使うことではなく、プログラムの実装とインタフェイスの分離を行うことである。オブジェクト指向を用いたコンポーネントを利用する側は、インタフェイスだけを用いてアクセスを

行い、その実装へはアクセスをしない。このようにすることで、コンポーネントの内部仕様の変更を行った場合でも、そのコンポーネントを利用する側は変更しなくて済むため、メンテナンスが容易になる。

このようなオブジェクト指向の考え方は異なるプラットフォーム間での通信が必要となる遠隔実験参加においても有用である。5 章で可視化ツールの開発において、モジュールという概念を用い、データ構造と実装を分離し、汎用性を有した可視化アプリケーションの開発を行うことができた。このインタフェイスとデータの分離という考え方は、リモートデータアクセスにおいては、いっそう重要になってくる。これは、各研究所で使われる計測システムやデータ保存システムは多様であり、研究所間でデータを交換するためには、システムアーキテクチャに依存しない方法が必要となるからである。

10.1.4 動画データ

動画データは、壁相互作用や不純物移送といったプラズマの振舞いを解析する上で、非常に重要である。ところが、こうした動画データのデータ量は他の計測データに比べ非常に大きくなるため、遠隔地からの利用は困難である。例えば、VGA (640×480 ピクセル) サイズで RGB 各色 8bit の画像を 30 フレーム/秒で送ると、データ転送レートは 26.4 MB / 秒 (= 220 Mbps) となる。この転送レートは現在の標準的な LAN 環境 で用いられる 100Base-TX の 100 Mbps を上回るため、インターネットのようなさらに帯域の狭いネットワークを利用して送信することができない。

従って、一般的にこのような動画データを送信する場合には、MPEG や QuickTime といった非可逆圧縮法を用いて、データを圧縮する方法が用いられる。例えば、庄司らは LHD の監視のため、MPEG-1 および 2 を利用した動画のオンデマンド送信システムの開発を行った [40]。

MPEG 等で使用される非可逆圧縮は、人間の視覚や聴覚に対しあまり重要でない成分を削除することによって、情報量が失われたことを知覚させないようする方法が用いられている。このため、非可逆圧縮されたデータは、LHD の監視のような目的で使用することは可能であるが、このデータを元に物理解析を行うことはできない。このような理由から、遠隔地からリアルタイムで高品質の動画データを解析することは、今まで困難であったが、全国の国立大学および国立の研究機関等を結ぶ高速ネットワーク、SuperSINET [41]、の登場によりこれが可能となった。

平成 13 年度より、共同研究の目的で、核融合科学研究所と東京大学、名古屋大学、京都大学の間が SuperSINET を利用した 1Gbps のネットワークで接続接続され、筆者はこのネットワークを利用した非圧縮の動画データの配信システムの構

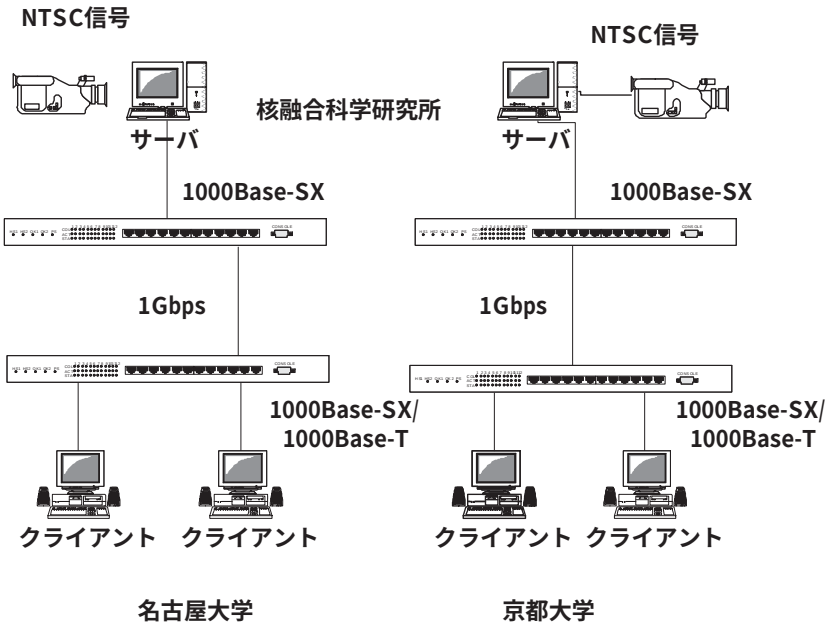

図 10.3: ビデオ配信システム

築を行った [42]。このシステムでは、CCD 等によって取り込まれた画像データを NTSC 信号に変換し、Linux PC サーバに取り付けられたビデオキャプチャカードによって取り込む。取り込まれた画像データは無圧縮でクライアントに送信を行う (図 10.3 参照)。

実際のネットワークを用いたテストでは、サーバ・クライアントの一対一通信で 29 〜 30 フレーム /秒の転送が可能であり、NTSC が毎秒 29.97 フレームであることを考えれば、ほぼフレーム落ちがなくデータが転送できる能力があることが確かめられた。

サーバは同時にクライアントとなることが可能で、上流のサーバから取得したデータを下流のデータに送信するというカスケード状にサーバ群を繋ぐことにより、クライアント数の増加に対応させた (図 10.4 参照)。

このシステムにより、遠隔地からリアルタイムで、物理解析に耐え得る品質の動画データを利用することが可能となった。

10.2　リモートコンピュータアクセス

今までの遠隔利用の例では、コンピュータ資源は遠隔地 (共同研究先) のものを利用し、データの取出しやコントロール等の一部の機能のみホスト側 (実験室側)

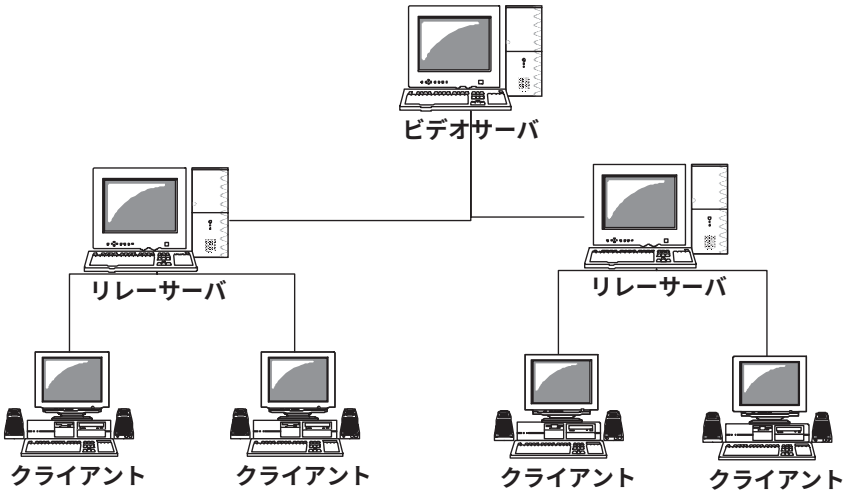

図 10.4: カスケード接続

の資源を利用していた。この節では、大部分のコンピュータ資源をホスト側で行う、所謂シンクライアント (Thin Client) システムを紹介する。この方式の利点は、ある決められたサービスの利用だけでなく、柔軟な運用が可能なことである。一方、欠点としては、サーバ/クライアント間での通信量が増え、良好なレスポンスが保てないこと。また、踏み台として使われるなど、セキュリティの問題が発生することである。特に後者に関しては、悪意のあるユーザによって他のシステムに壊滅的な被害を及ぼすことがあるので、ファイアウォールやハードウェアを利用したワンタイムパスワード方式等によるセキュリティ管理が不可欠である。

10.2.1 X Window System

X Windows System は MIT と DEC の共同研究である Athena プロジェクトから生まれたウィンドウシステムである。このシステムの特徴として、ウィンドウの実際の描画を行ったり、ユーザの入力を受け取る X サーバと、X サーバに対して図形や文字などの描画要求を行う X クライアントからなる、クライアント・サーバ型になっている。Athena プロジェクトは計算機資源の有効利用を目的としたものであり、例えば、X サーバと X クライアント間の通信には X プロトコルという OS に依存しない、TCP/IP のプロコルを使用する。この特徴により、異種コンピュータ間でリモートのプログラムを実行させ、その結果だけの表示することが可能である。ところが、実際には MS Windows や Macintosh 等の OS は基本的にはリモートでの実行を考慮していないため、主として X クライアントを動かすのは UNIX

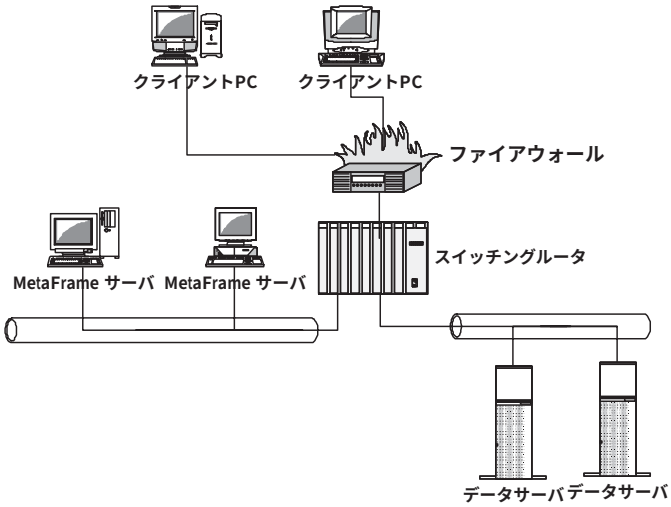

図 10.5: リモートアクセスサーバシステム構成図

や VMS 等のオペレーティングシステム上ということになる。先にあげた可視化ツールの idraw や NIFScope は MS Windows の他、UNIX での動作を保証している。このため、研究所外からこれらのアプリケーションをインストールしたホストにログインし、X 端末機能を利用し、可視化を行った結果だけを自分の PC や Mac 等表示させることが可能である。

10.2.2 MetaFrame

MS WindowsNT/2000 においても、ターミナルサーバと言う機能を用いて、クライアントは表示のみを行い、アプリケーションの実行等はサーバ側のコンピュータ資源を利用するということが可能である。本研究所では、平成 12 年度にこのターミナルサーバ機能を拡張した Citrix 社[4]の MetaFrame というプロダクツを導入した。MetaFrame クライアントは MS Windows の他、Macintosh(図 10.2.2) や UNIX、Java 等の多くのプラットフォームをサポートするため、ユーザが特別な PC を用意する必要がない。

システムの概要を図 10.5 に示す。セキュリティ確保のためにファイアウォールを設置しており、基本的には研究所内からのアクセスだけを通すように設定されている。共同研究等により外部からアクセスを行う場合には、ファイアウォールのルールセットにより、あらかじめ登録された IP アドレスから、ターミナルサーバへ向かうアクセスだけを許している。また、遠隔地からコンピュータを自由に

[4]http://www.citrix.co.jp/

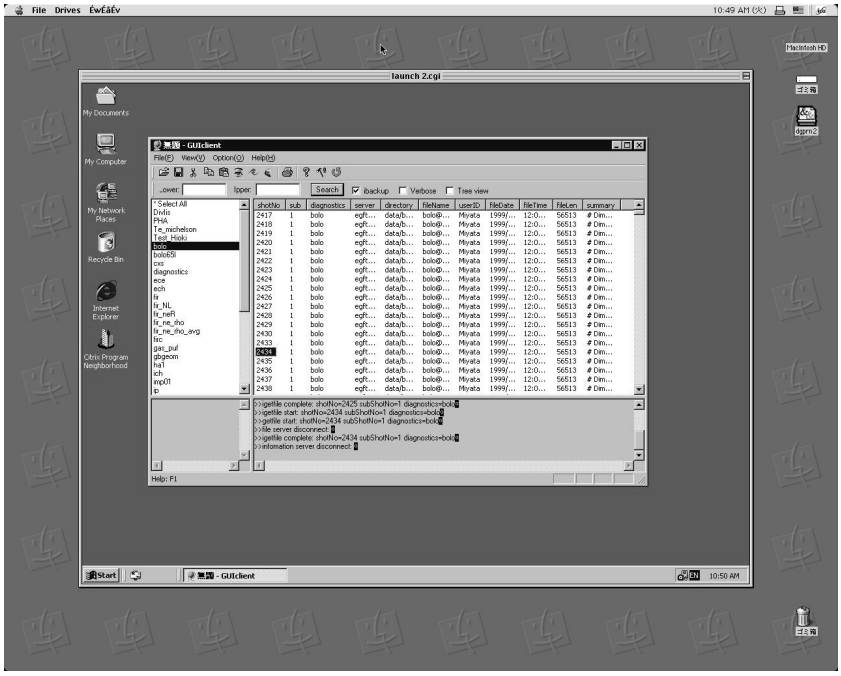

図 10.6: MetaFrame 実行画面 (MacOS9 上)

利用できるとそのマシンを踏み台にされる場合があるので、スイッチングルータのパケットフィルタリング機能により、ターミナルサーバから研究所内に向かう telnet や rlogin 等のパケットの通過を禁止し、データを取得するために必要なパケットのみを通過させるよう設定を行っている。

10.2.3 VPN

核融合科学研究所では外部の共同研究者に、さらに自由度の高い環境を提供するため、VPN (Virtual Private Network) を利用したリモートアクセスを用意している。VPN は、インターネット等の外部のネットワークを介しながら、離れた地点間を同一ネットワークであるかのように見せかける技術である。VPN を利用することによって、核融合科学研究所外のコンピュータを研究所内の LAN に接続されたコンピュータと同等に使用することが可能である。

一方で、悪意のある第三者に VPN を利用されることを避けるため、RSA 社[5]の SecureID を利用したワンタイムパスワードによる認証を導入した。一般に用いられている固定式のパスワードは、安易に推測できるパスワードや辞書を利用した

[5]http://www.rsasecurity.com/

総当り法により解読されてしまうことがあり、また、ネットワークパケットの盗み読みによって、他人に知られる可能性がある。

ワンタイムパスワードは、使い捨てのパスワードを生成するトークンと認証サーバから構成されるもので、トークンは、

- 内部の時計を利用する
- 入力された回数
- サーバーから送られてくる文字列

等を元にランダムな文字列を生成する機械[6]である。ユーザはトークンが生成した文字列を元にパスワードを入力し、認証サーバはこのパスワードと、期待しているパスワードを比較することによって認証を行う。トークンが生成する文字列は、一般に外部からは推測不可能であり、一回毎に変わるため、仮に盗み読まれたとしても安全である。

[6]SecureID のトークンは内部のクロックにより一分毎に新しい文字列を生成する。

第Ⅴ部

まとめ

第11章　まとめ

以上述べてきたように、LHD 実験で運用されている様々なコンピュータシステムの統合化の研究を行ってきた。これらは、以下の通りである。

オブジェクトデータベースと MO ジュークボックスの統合化

以前は、HDD に保存しきれないデータは、オブジェクトデータベースのイメージファイル全体を MO に移動し HDD 上からは削除していた。このため、古いデータを参照するためには、MO ジュークボックスからデータベースイメージを HDD にコピーし、データベースをオンライン化する必要があった。中期保存サーバでは、オブジェクトデータベースおよび MO ジュークボックスのデータ検索・取得方法に共通のインタフェイス層を設けることにより、両者の違いを意識させることのないアクセスを可能にした。CERN の RD45 プロジェクト等、他のオブジェクトデータベースを使った実験データ管理システムでは、階層型ファイルシステムを利用しているが、これらは有料であること、OS やデバイスが限定されること、柔軟性に欠けること等から、本研究で取った共通インタフェイス層を設ける方が優れている。また、ネットワーク分散型のデータベースを利用していたため、十分な帯域を確保できないネットワーク上でのパフォーマンスが問題となったが、オブジェクトデータベースによる検索をサーバ側だけで完結させることにより、10Mbps の LAN 上で 200 秒かかっていた処理を 30 秒まで短縮することができた。

これによって、収集したデータを参照したゲインの調整等を速やかに行うことが可能になり、限られたマシンタイムを有効に使うことが可能となった。

解析済みデータの統合化

解析データの管理を行う為、上記のシステムで用いた方法をベースにして、解析情報サーバシステムの開発を行った。これによって、異なるシステムで収集されたデータの参照が可能となった (図 11.1)。解析済みデータは第 5 サイクル終了までに、84 計測、59 万件のデータが登録され、実験中は一日平均 5,700 件の参照があった。今後もこのシステムに対するアクセスは増加して行くと考えられるが、

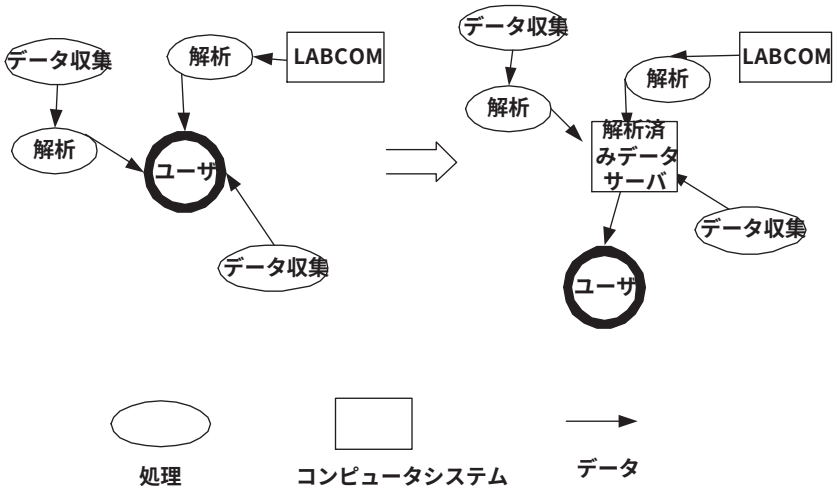

図 11.1: 解析済みデータサーバ

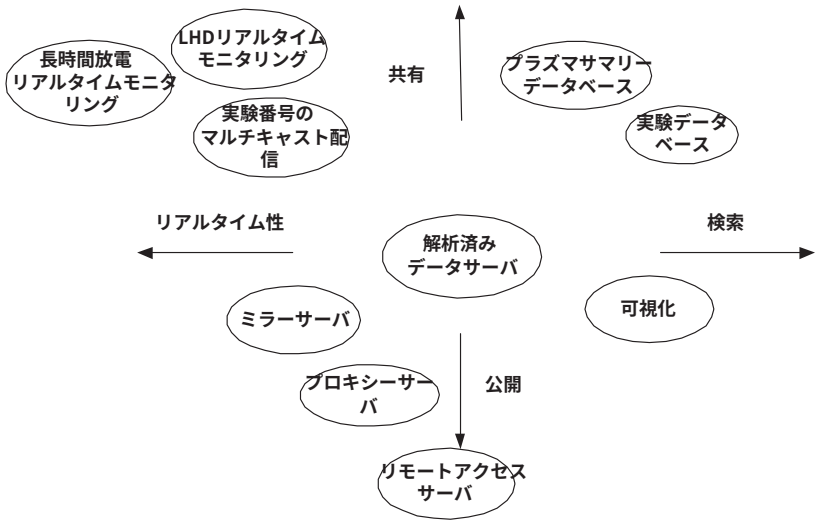

図 11.2: システムの拡張

このシステムはデータを保存する機能とデータの登録情報を管理する機能を分離し、さらにデータ保存部分も複数のサーバにより分散化を行うことが可能となっている。これによって、一台のサーバに対する負荷集中を避けることができるため、将来的な拡張へも柔軟に対応することが可能である。一方、MDSplus 等の従来の実験データ管理システムは、データ保存とデータ登録情報の管理という部分が一体となっており、分散化が困難である。このため、一台のホストに処理が集中するという問題が発生する。

さらにこのシステムでは、テキストファイルによるデータの統一を行い、数十〜数百 GB に達する大きな実験データ保存にテキストファイルを採用したという点が大きな特徴である。このような規模のデータ保存には従来バイナリファイルが用いられてきた。テキストファイルはバイナリ形式に比べファイルサイズが大きくなるという欠点があるが、圧縮技術の発達や、ハードディスクの低価格・大容量化により、これらの欠点は克服可能となった。一方、テキストファイルは、機種非依存であり、可読であることから、近年では HTML や XML 等のテキストベースのデータフォーマットが普及している。解析データのフォーマットは、将来的な拡張性を考慮し、コメント部分を活用することが出来る。例えば、数時間におよぶ長時間放電では、一回の放電あたり多数のデータが生成される。これらのデータを URL 形式のコメントとして記述することにより、可視化ツール等から、関連するデータを取り出すといった拡張も可能である。このような拡張は、コメントを利用しているため、現在使われているプログラムに影響を与えることなく行うことが出来る。

周辺システムの開発

解析済みデータサーバシステムを中心として、機能を拡張する形で周辺のシステムの構築を行った (図 11.2)。

膨大なデータを効率よく取り出すために、LHD 制御情報の管理を行っている LMS データベースとプラズマサマリ等の物理情報を連携させ必要なデータの検索を容易にした。また、可視化ツールの開発を行いデータの閲覧が簡単にできるようになった。これらの機能は、内部のシステムに詳しくない外部の共同研究者にとって有用である。さらに、平成 12 年からは、共同研究のサポートのため、解析情報システムを研究所外部から利用できるようシステムの研究・構築を行った。

これら一連の開発の中では、汎用性を重視した可視化ツールの開発が新しいものであり、これによってシステムの違いによる、データフォーマットの差異を吸収させることができた。

この機能は可視化ツール NIFScope の内部に Ruby を取り込むことにより実現

できた。Ruby はオブジェクト指向型言語であり、この技術を応用し、共通インタフェイスを提供、オーバーライド機能によるデータの違いの隠蔽等、によって解析データは元より他のシステムで取得・保存された多様なデータフォーマットにも簡単に対応することが可能となった。また、共通インタフェイスで定義されていない属性を Ruby の持つ連想配列に格納することで、将来的な拡張性にも対応している。さらに、このオブジェクト指向の機能を利用することによって、NIFScope のユーザ定義の計算式を簡潔に記述することが可能となり、データ解析を行う為に必要な配列計算を、プログラムではなく自然な式として書くことができる。加えて、Ruby は強力なテキスト処理能力を持ち、解析データのようなテキストベースの実験データを読み取るプログラムを開発することは容易である。この機能により、将来新しい形式のファイルが現れても対応することが可能である。

IP マルチキャストを用いたシステムの統合化

IP マルチキャストを用い、ショット番号の配信および新規登録データの通知を行うことによって統合化を行った。

ショット番号は、対象となるプラズマ放電を特定するために、最も基本となる情報であり、LHD の計測や解析に関わるコンピュータすべてが共有すべき情報である。ところが、個別収集におけるショット番号の通知は今まで MS Windows 共有フォルダを用いる方法をとっていたため、MS Windows 以外のプラットフォームで利用することが困難、あるいは OS によっては不可能であった。一方、今回開発した IP マルチキャストを用いる方法であれば、MS Windows 以外で利用することが可能である。平成 13 年度にはデータの解析を行うために Linux ベースの解析用 PC が導入されたが、これらは解析の開始のタイミングや、ショット番号の取得のために IP マルチキャストを利用している。また、解析済みデータシステムは、異なるシステム間でデータを相互参照するために作られたシステムであるが、登録された情報をすぐに参照できるように、データが登録されたことを IP マルチキャストで通知することで、システム間の連携度が高まった。

IP マルチキャストは一般的には動画や音声等のストリーム型のデータを送信するために用いられる技術である。これらの用途に IP マルチキャストが用いられている理由は、サーバ負荷がクライアント数の増加に依存せず、また、データを必要とするネットワークだけに配信することができる為、データの送信が効率よく行えるからである。反面、TCP に比べ信頼性の低いという欠点があるが、動画や音声等のデータであれば多少のパケットの取りこぼしや順序の不整合があっても影響は少ない。例えば刈谷のシステムで IP マルチキャストをリアルタイムモニタリングの為に使用したが、これもストリーム型のデータ配信で、モニタリング

という観点では、ごく少数のパケットの損失であれば、あまり重要ではない。一方、本システムではあえてこの IP マルチキャストをショット番号や新規登録データの通知というイベント型のデータの通知に適用した点が新しいと言える。マルチキャストの送受信のテストでパケットの取りこぼしがおきなかったことから、実験 LAN 内部という限定されたネットワークでは、十分信頼性があると考えられる。また、仮にショット番号の取得に失敗した場合でも、収集時刻とショット番号を付き合わせることにより修正することが可能である。このような目的に IP マルチキャストを使用するのであれば、パケット落ちが生じるかも知れないというデメリットに比べ、複数のクライアントに同時に通知できる点や、クライアント数の増加に耐えうるという点で、メリットの方が大きいと思われる。

さらに、ストリーム型データ配信という点では、長時間放電用のリアルタイムモニタリングシステムに IP マルチキャスト技術を用いた。この点では、刈谷のシステムと同様であるが、本システムではマルチキャストの送受信部分と解析を行う部分を分離し、解析を行う部分をユーザに開放することで、より汎用性に富むシステムを開発することができた。

平成 14 年 10 月 13 日からの 2 ヶ月間に一放電あたり 150MB 以上 (データ圧縮後)、一日あたり 15GB 以上の計測データが収集されたが、これらのデータを個々の利用者が直接利用することは、ネットワーク負荷の上昇による他のシステムの影響を考えると不可能である。一方、これらのデータを元に作成された解析済みデータは一放電あたり 700KB、一日あたり 80MB 程度であり、この程度のデータであれば、十分なネットワーク帯域が確保できないインターネット等からの利用も十分可能である。また、この間の解析済みデータサーバの利用状況は、登録が延べ 76,000 件、参照が 190,000 件となっている。参照のうち、95,000 件は同一ユーザからの 2 機種以上の計測機器データの同時参照であり、最大では 9 計測機器のデータが同時に取得されている。複数データの同時参照、取得は、統合したシステムで初めて可能であること、特に、システム間の情報伝達の同時性、高信頼性が得られた結果、放電終了後速やかにデータを登録できるようになったことによるものである。

今後の大規模データ処理システムの開発に向けた提言

本論文で扱った異なるシステムが混在するという状況は一般的に大規模かつ長期的な計画では発生し得る問題と考えるべきである。また、大型実験装置の開発においては、本来の機能に直結した装置本体やその制御システム、および計測機器等に重点がおかれ、データの収集等は軽視されがちである。このため、その時々

に応じた短期的な視野でデータ処理システムが開発されることになり、将来生じ得るシステムの変更要求に対応することが困難である。

本論文で述べた開発は、既存のシステムに対する影響がなるべく小さくなるように開発をきたが、理想的にはあらかじめ将来起こりうるシステム変更に対し最小限の変更で対応できるように考慮した設計を行うべきである。

例えば、コンピュータ技術は急速に変化しており、他の計測機器に比べ、物理実験で一応の成果がでる前に、機器や OS のサポートが終了してしまう可能性が大きい。そのため、システムを構築するときには常に他の環境で実行する可能性を考慮しなければならない。

この解の一つはコンピュータや OS に依存しない技術の使用である。

CERN で電子文章を参照するために発明された Web およびそのプロトコルである HTTP が今日のように普及したのは、その単純さゆえである。HTTP はセッションの維持や、エラー訂正を持たないなど、他のファイル転送プロトコルに比べて機能的に劣っていたが、単純であるが故、様々なプラットフォームに移植され、拡張されるに至った。解析情報サーバシステムの中で使われている技術は一般的でオープンな技術のみの集まりであり、このシステムのクライアントおよび、サーバを他のシステムに移植することは容易である。当初、解析済みデータサーバシステムでは、独自のプロトコルを開発し利用していたが、運用の結果当初予定していた性能に達しなかったので、標準的な FTP を採用することにした。独自の技術を使うことは、開発やメンテナンスが難しくなる等の問題も発生する。特に、長期のプロジェクトで一つのシステムが何年にも渡り使われる場合は、プログラムのメンテナンスに関わる人間の交代も考えられ、独自技術を使った場合は、その習得に多くの時間を費やすことになる。その意味で今回の開発を通じて得られた上記の経験は、大規模システムの統合化と拡張性、柔軟性を高める上での方法論として、オープンな汎用、標準的技術の利用が如何に重要であるかの知見を得ると同時に、この様なシステムの統合化手法においてはオープン技術をどの様に組み合わせてシステムを最適化するかが最も重要な事項となる。これによって、標準的なプロトコルを用いて、様々なプラットフォームに容易に移植することができる。

移植性のもう一つの解はオブジェクト指向設計である。これは、C++ や Java 等のオブジェクト思考型言語を使用することではなく、そのシステム設計にオブジェクト指向の考え方を導入することである。

オブジェクト指向のシステム設計に対する恩恵は、その抽象化とカプセル化によってプログラムをモジュール化し再利用を可能にできることにある。カプセル化とは、データ構造を隠蔽しデータの操作のみを見せることで、内部のデータ構造の変更を外部に影響を及ぼさないようにすることができる。データ処理システ

ムへの応用では、種々のデータへの直接参照を行わず、共通のインタフェイスを提供することで、システムの再利用を行うことが可能となる。

本論文で扱った一連のシステムの開発に於いて、この移植性、汎用性を重視し、すべてのシステムで常に少なくとも 2 つ以上のシステム (MS Windows または UNIX) で動作させることを念頭において開発を行い、なるべく汎用に使えるものを目指した。例えば、データの検索では、Java というプラットフォームに依存しない技術を用いることで、OS に依存しないアプリケーションを提供し、また、可視化技術では GTK+というツールキットを利用することにより、MS Windows および UNIX で動作させることを可能にした。また、NIFScope では、マクロ言語とデータローダのモジュール化というオブジェクト指向的な考え方を導入することにより、様々なデータに対応することが可能になった。

この研究で用いた

- 標準技術、オープンソースによる、プラットホームに依存しない技術
- オブジェクト指向設計

による移植性の高さや汎用性は世界の核融合研究を行う大学、研究所で使われているシステムにはない特長であり、この特長は LHD 実験のみならず他の物理実験にも適用可能である。今後登場する多国参加による共同プロジェクトでは、多様なシステムが混在することになり、また、プロジェクトが長期化することから、新しい機器の導入や新たな要求の発生等、システムの変更が必要となる。このような場合でも、本研究で用いた手法を適用することで、システム全体の変更を最小限にし、将来的な変更に柔軟に対応することが可能である。

謝辞

本研究を進めるにあたり、研究の機会を与えていただいた核融合科学研究所の須藤滋氏、小森彰夫氏、上村鉄雄氏に感謝いたします。本論文をまとめるにあたり、貴重なご意見ご指導を頂いた国立情報学研究所の日比野靖氏に感謝します。また、システムの開発にあたり、システム開発にご協力いたただき、また、研究のアドバイスをいただいた核融合科学研究所の中西秀哉氏、庄司守氏に感謝します。貴重なご意見等ご指導頂いた、中部大学の山口作太郎氏、職業能力開発大学校の寺町康昌氏、松阪大学の奥村晴彦氏、プログラム作成やデータの集計を手伝っていただいた日本サンマイクロシステムズの田村和一氏、村津良平氏、CTC SP の柴田健二氏に感謝します。また、本研究は LHD グループ諸氏の協力なしでは、行えませんでした、ここに改めてお礼を申し上げます。最後に、長年に渡ってご支援いただいたサンマイクロシステムズの安光正則氏に感謝いたします。

付 録A　共通フォーマット

解析済みデータの例を図 A.1 に示す。解析済みデータはテキストファイルであり、ヘッダとデータからなり、さらにヘッダは必須パラメタとコメントから構成される (図 A)。行末コードは LF(UNIX 形式) または CR+LF(MS Windows 形式) を用いる。ヘッダ部は行頭が'#' で始まる部分で、#で始まる部分を読み飛ばすことにより、データ部を直接読み取るプログラムを書くことは比較的簡単にできる。

ヘッダにおける必須パラメタは 3 カラムめより記述することを必須とするが、可読性を上げるため、=の前後のスペースや、空行等を適宜加えてもよい。

パラメタは、

パラメタ名=値

というフォーマットをしており、必須パラメタとして表 A.1 のようなものがある。

値は文字列、数値、日付のいずれかで、文字列の場合は「'」シングルクォーテーションで囲む。数値は、Fortran における数値表現に準ずる。パラメタが存在しないとき (単位がない等) は「"」を指定する (シングルクォーテーション二つ)

例えば、物理量 V,W の一次元分布 (サンプリング数=n) の時間推移 (サンプリング数=m) を測定し、観測量 $V_{1,1} = V(x_1, t_1)$、$V_{1,2} = V(x_1, t_2)$、. . .、$V_{n,m} = V(x_n, t_m)$、$W_{1,1} = W(x_1, t_1)$、$W_{1,2} = W(x_1, t_2)$、. . .、$W_{n,m} = W(x_n, t_m)$ を得たとすると (図 A.3)、DimNo、DimSize、ValNo は以下のようになる。

```
# DimNo=2
# DimSize=n,m
# ValNo=2
```

コメント部は

```
#[Comments]
```

から始まる行で、ここには必須パラメタ以外のコメントを記述することができる。データ部は

```
#[Data]
```

```
# [Parameters]
# Name = 'SXR'
# ShotNo = 6115
# Date = '1/29 15:14'
#
# DimNo = 2
# DimName = 'R', 'TIME'
# Dimsize = 80, 100
# DimUnit = 'm','s'
#
# ValNo = 4
# Val= 'dt', 'dR', 'SXR', 'dSXR'
# ValUnit = 's', 'm', 'V', 'V'
#
# [comments]
# PHI=3.5
# PHIunit='portNO'
# comments = 'Be thickness = 7.5 micro m'
# [data]

         3.26640, 0.302, 0.01, 0.02 , -0.0000518, 0.0000002
         3.27489, 0.302, 0.01, 0.02 , -0.0000076, 0.0000001
         3.28338, 0.302 ,0.01, 0.021, -0.0000820, 0.0000003
         3.29186, 0.302, 0.01, 0.021,  0.0000149, 0.0000003
        ...........
        ...........
```

図 A.1: 解析済みデータの例

名前	説明
NAME	計測名
ShotNo	ショット番号
SubShotNO	サブショット番号
Date	解析済みデータを作成した日時
DimNo	データの次元
DimName	次元名
DimSize	次元のサイズ
DimUnit	次元の単位
ValNo	物理量の数
ValName	物理量の名前
ValUnit	物理量の単位の名前

表 A.1: 必須パラメタ

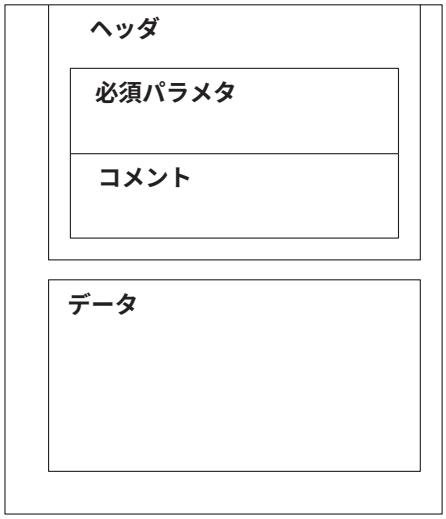

図 A.2: 解析済みデータ書式

文字列	'Thomson'、'Te'
数値	0.123、.123、+1.23e+18、1234
日付	'09/22/2002 13:21'

表 A.2: リテラル一覧

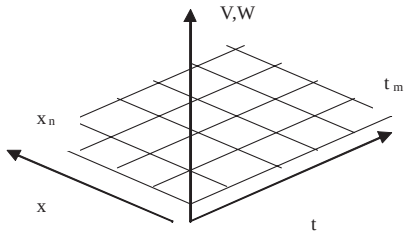

図 A.3: 計測データ

で始まり、この行以降は一行毎にサンプリング点およびその点における値を“ , ”(カンマ) で区切ったデータが続く。先ほどの例では、以下のようになる。

$t_1,\quad x_1,\quad V_{1,1},\quad W_{1,1}$
$t_2,\quad x_1,\quad V_{2,1},\quad W_{2,1}$
$\cdots,\quad \cdots$
$t_n,\quad x_m,\quad V_{n,m},\quad W_{n,m}$

この時、各サンプリング点の並び方 (t と x のどちらが先に増えるか等) については規定しないが、行数はサンプリング数、すなわち、DimSize の積 $(n \times m)$ と一致する必要がある。

尚、フォーマットが文法的に正しいかどうかチェックするためのツールとして、MS Windows および UNIX 用に ilint というコマンドを用意している。

付 録B　解析済みデータ取出しプログラムのサンプル(Java)

Java による解析済みデータ取り出し簡易版プログラムを紹介する。本プログラムは JDK1.2.2 および 1.3.1 で動作を確認しているが、非標準クラスの sun.net.ftp.FtpClient を使用しているため環境によっては動作しない可能性がある。

このプログラムは解析情報サーバのプログラムの簡潔性を示すための例であり、最低限の機能しか提供しない。より完全なプログラムとして使うにはコマンドラインオプションの解析やエラー処理等を加える必要がある。

```
 1 import java.sql.*;
 2 import java.io.*;
 3 import java.util.*;
 4 import java.util.zip.*;
 5 import sun.net.ftp.FtpClient;
 6
 7 // 解析済みデータの取得クラス
 8 public class IGetfile {
 9     // データベースサーバ
10     static final String DBHOST = "server";
11     // データベース名
12     static final String DBNAME = "database";
13     //ユーザ名
14     static final String DBUSER = "scott";
15     //パスワード
16     static final String DBPASS = "tiger";
17
18     //使用法の出力
19     static private void usage(){
20         System.err.println("usage: java IGetfile -s shotno "
21                            + "-d diagnoame [ -m subshotno ] [ -o outputfle]");
```

```
22          System.exit(1);
23      }
24
25      static public void main(String args[]){
26          String shotno = null;
27          String subshotno = "1";
28          String diag = null;
29          String output = null;
30
31          int i = 0;
32          //コマンドラインオプションの解析
33          while ( i < args.length ){
34              if ( "-s".equals(args[i]) ){
35                  shotno = args[++i];
36              }else if ( "-m".equals(args[i])){
37                  subshotno = args[++i];
38              }else if ( "-d".equals(args[i])){
39                  diag = args[++i];
40              }else if ( "-o".equals(args[i])){
41                  output = args[++i];
42              }else{
43                  System.err.println("unknown option "+args[i]);
44                  usage();
45              }
46              i++;
47          }
48
49          if ( shotno == null || diag == null )
50              usage();
51
52          try {
53              Connection conn=null;
54              String username = System.getProperty("user.name");
55
56              DriverManager.registerDriver(new org.postgresql.Driver());
57              String url = "jdbc:postgresql://"
```

```
58              +DBHOST+"/"+DBNAME;
59
60          //データベースへの接続
61          conn =DriverManager.getConnection
62              (url,DBUSER,DBPASS);
63
64          System.out.println(">>information server connect: rdb_server="
65                             + Constants.DBHOST
66                             + "rdb_db=" + Constants.DBNAME
67                             + "rdb_account=" + Constants.DBUSER);
68
69
70          String sql = "select * from _locationfileinfo_ where "
71              + "diagnostics = '"+diag+"'"
72              + "and shotno = "+shotno
73              + "and subshotno = "+subshotno;
74
75          PreparedStatement stmt = conn.prepareStatement(sql);
76
77          //キュエリの発行
78          ResultSet rs = stmt.executeQuery();
79          String server = null;
80          String directory = null;
81          String filename = null;
82
83          //データの格納場所の取得
84          while ( rs.next() ){
85              server = rs.getString("server").trim();
86              directory = rs.getString("directory").trim();
87              filename = rs.getString("filename").trim();
88          }
89          rs.close();
90          conn.close();
91
92          //FTP によるデータの取得
93          FtpClient client = new FtpClient(server);
```

```
 94            client.login("anonymous",username);
 95            client.binary();
 96            client.cd(directory);
 97            InputStream in = client.get(filename);
 98
 99            // ZIP の展開
100            ZipInputStream zin = new ZipInputStream(in);
101            ZipEntry ent = zin.getNextEntry();
102            if ( output == null )
103                output = ent.getName();
104            BufferedInputStream bis = new BufferedInputStream(zin);
105
106            byte buffer[] = new byte[1024];
107            int count;
108
109            BufferedOutputStream  os =
110                new BufferedOutputStream(new FileOutputStream(output));
111
112            while ( (count = bis.read(buffer)) > 0 ){
113                                os.write(buffer,0,count);
114            }
115            os.close();
116        }catch(Exception ex){
117            ex.printStackTrace();
118        }
119    }
120 }
```

使用法

java Igetfile -s ショット番号 -m サブショット番号 -d 計測名 -o 出力ファイル

(例)

```
% Java Igetfile -s 10000 -d thomson
```

付 録C　実験ログデータベース テーブル定義

カラム名	説明	データ型	値範囲
nDataCode		int	
nExecutionFlag	実行済みフラグ	int	”1(済),0(未)”
Status	状態フラグ	char(1)	
UpdateID	行 ID	int	
nShotnumber	ショット番号	int	
dDatacreationTime	日付時間	datetime	
nPSnumber	通電番号	int	
nCoolingnumber	冷却番号	int	
MagneticField	コイル用電源：磁場強度	float	
MagneticAxis	コイル用電源：磁気軸	float	
Quadruple	コイル用電源：四重極成分	float	
GasType	ガスパフ：ガス種	varchar(255)	
GasPuffBulb1Status	ガスパフ：ピエゾ 1 バルブ状態	varchar(255)	”ON,OFF”
GasPuffBulb2Status	ガスパフ：ピエゾ 2 バルブ状態	varchar(255)	”ON,OFF”
GasPuffBulb3Status	ガスパフ：ピエゾ 3 バルブ状態	varchar(255)	”ON,OFF”
GasPuffBulb4Status	ガスパフ：ピエゾ 4 バルブ状態	varchar(255)	”ON,OFF”
GasPuffBulb5Status	ガスパフ：ピエゾ 5 バルブ状態	varchar(255)	”ON,OFF”
GasQuantity	ガスパフ：入射ガス量	float	
GasPuffComment	ガスパフ：コメント	varchar(255)	
LIDACurrentMAX	LID：電流値	float	
LIDStatus	LID：ON/OFF	varchar(255)	”ON,OFF”
LIDComment	LID：コメント	varchar(255)	
Pellet1Time	ペレット：系統 1 入射時間	float	
Pellet1Status	ペレット：系統 1ON/OFF	varchar(255)	”ON,OFF”
Pellet2Time	ペレット：系統 2 入射時間	float	

Pellet2Status	ペレット：系統 2ON/OFF	varchar(255)	”ON,OFF”
Pellet3Time	ペレット：系統 3 入射時間	float	
Pellet3Status	ペレット：系統 3ON/OFF	varchar(255)	”ON,OFF”
Pellet4Time	ペレット：系統 4 入射時間	float	
Pelllet4Status	ペレット：系統 4ON/OFF	varchar(255)	”ON,OFF”
Pellet5Time	ペレット：系統 5 入射時間	float	
Pellet5Status	ペレット：系統 5ON/OFF	varchar(255)	”ON,OFF”
PelletComment	ペレット：コメント	varchar(255)	
ECHToshiba1Time	ECH：東芝 1 入射時間	float	
ECHToshiba1PulseWidth	ECH：東芝 1 入射パルス幅	float	
ECHToshiba1Power	ECH：東芝 1 入射電力	float	
ECHToshiba1Status	ECH：東芝 1ON/OFF	varchar(255)	”ON,OFF,AGI”
ECHToshiba2Time	ECH：東芝 2 入射時間	float	
ECHToshiba2PulseWidth	ECH：東芝 2 入射パルス幅	float	
ECHToshiba2Power	ECH：東芝 2 入射電力	float	
ECHToshiba2Status	ECH：東芝 2ON/OFF	varchar(255)	”ON,OFF,AGI”
ECHToshiba3Time	ECH：東芝 3 入射時間	float	
ECHToshiba3PulseWidth	ECH：東芝 3 入射パルス幅	float	
ECHToshiba3Power	ECH：東芝 3 入射電力	float	
ECHToshiba3Status	ECH：東芝 3ON/OFF	varchar(255)	”ON,OFF,AGI”
ECHGYCOM1Time	ECH：GYCOM-1 入射時間	float	
ECHGYCOM1PulseWidth	ECH：GYCOM-1 入射パルス幅	float	
ECHGYCOM1Power	ECH：GYCOM-1 入射電力	float	
ECHGYCOM1Status	ECH：GYCOM-1ON/OFF	varchar(255)	”ON,OFF,AGI”
ECHGYCOM2Time	ECH：GYCOM-2 入射時間	float	
ECHGYCOM2PulseWidth	ECH：GYCOM-2 入射パルス幅	float	
ECHGYCOM2Power	ECH：GYCOM-2 入射電力	float	
ECHGYCOM2Status	ECH：GYCOM-2ON/OFF	varchar(255)	”ON,OFF,AGI”
ECHCPI1RTime	ECH：CPI-1R 入射時間	float	
ECHCPI1PulseWidth	ECH：CPI-1R 入射パルス幅	float	
ECHCPI1RPower	ECH：CPI-1R 入射電力	float	
ECHCPI1Status	ECH：CPI-1RON/OFF	varchar(255)	”ON,OFF,AGI”
ECHComment	ECH：コメント	varchar(255)	
NBI1Time	NBI：1 号機入射時間	float	
NBI1PulseWidth	NBI：1 号機パルス幅	float	
NBI1Power	NBI：1 号機入射電力	float	
NBI1Stauts	NBI：1 号機 ON/OFF	varchar(255)	”ON,OFF”
NBI2Time	NBI：2 号機入射時間	float	
NBI2PulseWidth	NBI：2 号機パルス幅	float	
NBI2Power	NBI：2 号機入射電力	float	

NBI2Status	NBI：2 号機 ON/OFF	varchar(255)	”ON,OFF”
NBIComment	NBI：コメント	varchar(255)	
ICRF1Time	ICRF：1 号機入射時間	float	
ICRF1PulseWidth	ICRF：1 号機パルス幅	float	
ICRF1Power	ICRF：1 号機入射電力	float	
ICRF1Status	ICRF：1 号機 ON/OFF	varchar(255)	”ON,OFF”
ICRF2Time	ICRF：2 号機入射時間	float	
ICRF2PulseWidth	ICRF：2 号機パルス幅	float	
ICRF2Power	ICRF：2 号機入射電力	float	
ICRF2Status	ICRF：2 号機 ON/OFF	varchar(255)	”ON,OFF”
ICRF2Comment	ICRF：コメント	varchar(255)	
CoordinatorComment	コーディネータコメント	varchar(255)	
GAMMA	コイル用電源：ガンマ	float	
LIDB1CurrentMAX	LID：B1 電源電流最大値	float	
LIDB12CurrentMAX	LID：B1 電源電流最大値	float	
Pellet6Time	ペレット:系統 6-入射時間	float	
Pellet7Time	ペレット:系統 7-入射時間	float	
Pellet8Time	ペレット:系統 8-入射時間	float	
Pellet9Time	ペレット:系統 9-入射時間	float	
Pellet10Time	エレット:系統 10-入射時間	float	
NBICoordinatorComment	NBI:コーディネータコメント	varchar(255)	
ICRFCoordinatorComment	ICRF:コーディネータコメント	varchar(255)	
PelletCoordinatorComment	ペレット:コーディネータコメント	varchar(255)	
GasPuffCoordinatorComment	ガスパフ:コーディネータガス種	varchar(255)	
ECHCoordinatorComment	ECH:コーディネータコメント	varchar(255)	
Pellet6Status	ペレット:系統 6-ON/OFF	varchar(255)	
Pellet7Status	ペレット:系統 7-ON/OFF	varchar(255)	
Pellet8Status	ペレット:系統 8-ON/OFF	varchar(255)	
Pellet9Status	ペレット:系統 9-ON/OFF	varchar(255)	
Pellet10Status	ペレット:系統 10-ON/OFF	varchar(255)	
ExperimentTheme	実験テーマ	varchar(255)	
wp_max	プラズマ蓄積エネルギー（反磁性計測）の最大値 [KJ]	float	
time_wpmax	プラズマ蓄積エネルギー（反磁性計測）の最大の時間 [s]	float	

nl_max	線平均密度（ミリ波計測）の最大値 $[10^{19}/m^2]$	float	
time_nlmax	線平均密度（ミリ波計測）の最大の時間 [s]	float	
nets_max	密度（トムソン）の最大値 $[10^{19}/m^2]$	float	
time_netsmax	密度（トムソン）の最大の時間 [s]	float	
tets_max	電子温度（トムソン）の最大値 [KeV]	float	
time_tetsmax	電子温度最大（トムソン）の時間 [s]	float	
ticx_max	イオン温度（ＣＸＲＳ）の最大値 [KeV];	float	
time_ticxmax	イオン温度最大（ＣＸＲＳ）の時間 [s]	float	
ip_max	プラズマ電流（電磁計測）の最大値（絶対値が最大）[kA]	float	
time_ipmax	プラズマ電流（電磁計測）最大の時間 [s]	float	
ech1_pow	ポートスルー ECH パワー（ＥＣＨ１）[MW]	float	
ech2_pow	ポートスルー ECH パワー（ＥＣＨ２）[MW]	float	
ech3_pow	ポートスルー ECH パワー（ＥＣＨ３）[MW]	float	
ech4_pow	ポートスルー ECH パワー（ＥＣＨ４）[MW]	float	
ech5_pow	ポートスルー ECH パワー（ＥＣＨ５）[MW]	float	
ech6_pow	ポートスルー ECH パワー（ＥＣＨ６）[MW]	float	
nbi1_pow	ポートスルーＮＢＩパワー（ＮＢＩ１）[MW]	float	
nbi2_pow	ポートスルーＮＢＩパワー（ＮＢＩ２）[MW]	float	
ich1_pow	ポートスルーＩＣＨパワー（ＩＣＨ１）[MW]	float	
ich2_pow	ポートスルーＩＣＨパワー（ＩＣＨ２）[MW]	float	
ich3_pow	ポートスルーＩＣＨパワー（ＩＣＨ３）[MW]	float	

time_dis	放電時間 [s]	float	
ip_wpmax	最大エネルギー時のプラズマ電流 (電磁計測)[kA]	float	
nl_wpmax	最大エネルギー時の線平均密度 (ミリ波計測)[$10^{19}/m^2$]	float	
net0ts_wpmax	最大エネルギー時の密度 (トムソン@真空磁気軸)[$10^{19}/m^2$]	float	
te0ts_wpmax	最大エネルギー時の電子温度 (トムソン@真空磁気軸) [KeV]	float	
ne7ts_wpmax	最大エネルギー時の密度 (トムソン@真空 ρ =0.7)[$10^{19}/m^2$]	float	
te7ts_wpmax	最大エネルギー時の電子温度 (トムソン@真空 ρ =0.7)[KeV]	float	
ne1ts_wpmax	最大エネルギー時の密度 (トムソン@真空 ρ =1)[$10^{19}/m^2$]	float	
te1ts_wpmax	最大エネルギー時の電子温度 (トムソン@真空 ρ =1)[KeV]	float	
nedpro_wpmax	最大エネルギー時の密度 (静電プローブ@ダイバータ)[$10^{19}/m^2$]	float	
tedpro_wpmax	最大エネルギー時の電子温度 (静電プローブ@ダイバータ)[KeV]	float	
ti0cx_wpmax	最大エネルギー時のイオン温度 (CXRS@真空磁気軸)[KeV]	float	
radpow_wpmax	最大エネルギー時の放射パワー (ボロメータ)[kW]	float	
c3_wpmax	最大エネルギー時の CIII 発光強度 [A.U.]	float	
o5_wpmax	最大エネルギー時の CV 発光強度 [A.U.]	float	
echpow_wpmax	最大エネルギー時のポートスルー ECH パワー (積算)[MW]	float	

nbipow_wpmax	最大エネルギー時のポートスルー NBI パワー (積算)[MW]	float	
nbipowst_wpmax	最大エネルギー時のシャインスルー NBI パワー (積算)[MW]	float	
ichpow_wpmax	最大エネルギー時のポートスルー ICH パワー (積算)[MW]	float	

付 録D　マルチキャストパケットフォーマット

マルチキャストパケットは以下に示すフォーマットからなり、パケットの種類を判別する4バイトの識別子と、パケットサイズを記述した4バイトのヘッダを有する。8バイト目以後のはPacketIDに依存する。マルチキャストアドレスは225.1.1.1〜225.1.1.3を使用し、TTL(Time To Live)は4とする。また、バイトデータの格納順はリトルエンディアンで符号あり、浮動小数はIEEE754を用いる。

現在パケットIDは1〜3まで定義されており、それぞれのIDに対するフォーマットは以下のようになっている。

開始-終了 (byte)	サイズ (byte)	型	説明
0-3	4	signed int	パケット ID
4-7	4	signed int	パケットサイズ
8-	不定	パケット ID に依存	

表 D.1: マルチキャストパケットフォーマット

開始-終了 (byte)	サイズ (byte)	型	説明
8-11	4	char[4]	“1.0” (固定)
12-15	4	signed int	データ ID
16-19	4	time_t	基準時:1970 年 1 月 1 日からの経過秒数
20-27	8	signed long long	基準時からの経過時間 (単位:nsec)
28-31	4	signed int	変数の数
32-39	8	double	変数 1 の値
40-47	8	double	変数 2 の値

表 D.2: リアルタイムモニタリングパケットフォーマット ID=2

開始-終了 (byte)	サイズ (byte)	型	説明
8-11	4	signed int	シーケンス状態 bit 0 ... シーケンス実行中 bit 1 ... 放電終了
12-15	4	signed int	ショット番号
16-19	4	signed int	サブショット番号

表 D.3: 実験シーケンスパケット ID=3

開始-終了 (byte)	サイズ (byte)	型	説明
8-43	36	char[]	計測名 (NULL で終わる)
12-15	4	signed int	ショット番号
16-19	4	signed int	サブショット番号

表 D.4: 解析済みデータ登録通知 ID=1

関連図書等

[1] A. Iiyoshi et al., “An Overview of the Large Helical Device Project”, Nucl. Fusion, **39** (1999), pp.1245-1256

[2] http://www.postgresql.org/

[3] K.Yamazaki, et. al., “Design of the central control system for the Large Helical Device (LHD)”,Nucl. Instr. and Math. in Phys. Res. **A 352** (1994) pp.43-46

[4] H.Yamada, et. al., “Design of central control and man-machine interface systems for large helical device”, Fusion Technology 1996, 1997, Elsevier Science, pp. 945-948

[5] H. Nakanishi et. Al., “Distributed processing and network of data acquisition and diagnostics control for large helical device (LHD)”, Fus. Eng. Des., **43** (1999), pp. 293-300

[6] 山口他,“超伝導コイル実験監視システム”, プラズマ核融合, **73** (1997), pp.335-342

[7] S. Yamaguchi, et al.“Control and Plasma Data Acquisition System for LHD Experiment”, Fus. Eng. Des., **48** (200) pp.9-15

[8] J.A. Stillerman and T.W. Fredian, “MDSplus Data Acquistion System”, 11th Topical Conference on High Temperature Plasma, Monterey, 1996

[9] T.Fredian, “C-Mod Data Acquisition System” High Temperature Plasma Diagnostics, Monterrey, May 1996,

[10] S.Ohdachi, Annual Report of National Institute for Fusion Science, APRIL 1998- MARCH 1999

[11] D.P. Schissel et. al., “Recent Enhancements to Analyzed Data Acquisition and Remote Participation at the DIII-D National Fusion Facility”, Fus. Eng. Des., **56** (2001), pp. 1005-1009

[12] J.G. Krom, “The evolution of control and data acquisition at JET”, Fus. Eng. Des., 43 (1996), pp.265-273

[13] 青柳哲雄,“JT-60 のデータ処理”, プラズマ核融合, **72** (1996), pp.1370-1375

[14] T.Matsuda et. al., “Status of JT-60 data processing system”, Fus. Eng. Des., **48** (2000), pp.99-104

[15] B. Guillerminet, “The acquisition system for Tore Supra 1000 s discharges”, Fus. Eng. Des., **48** (2000), pp.155-161

[16] http://nssdc.gsfc.nasa.gov/cdf/html/tech_brief.html

[17] http://www.cv.nrao.edu/fits/documents/standards/fits_standard.ps

[18] http://members.aol.com/rmcdjcamp/index.htm

[19] http://www.unidata.ucar.edu/packages/netcdf/guide.txn_toc.html

[20] http://pds.jpl.nasa.gov/stdref/

[21] http://xml.gsfc.nasa.gov/XDF/

[22] R.G.G.Cattel, オブジェクトデータベース標準:ODMG-93 Release 1.1, 共立出版

[23] http://www.mysql.com/

[24] J. Schachter, et al., “Data Analysis Software Tools for Enhanced Collaboration at the DIII-D National Fusion Facility”, Fus. Eng. Des., **48** (2000), pp.91-98

[25] http://www.gtk.org/

[26] http://www.gnome.org/

[27] G.Manduchi, “The Java Interface of MdsPlus: towards an Unified Approach for Local and Remote Data Access”, Fus. Eng. Des., **48** (2000), pp.163-170

[28] http://www.gnu.org/copyleft/gpl.html

[29] http://www.python.org/

[30] まつもとゆきひろ, オブジェクト指向スクリプト言語 Ｒｕｂｙ, アスキー

[31] http://www.tcl.tk/

[32] M.Emoto et. al., “A trial to combine heterogeneous computer system in NIFS”, Fus. Eng. Des., **48** (2000), pp. 83-89

[33] EMOTO Masahiko et. al.,“A PROXY SERVER FOR A REAL-TIME MONITORING SYSTEM”, PCaPAC2000, Hamburg, Oct. 2000

[34] M.Emoto et. al.,“3D REAL-TIME MONITORING SYSTEM FOR LHD PLASMA HEATING EXPERIMENT”, Fus. Eng. Des., **56** (2001), pp.1017-1021

[35] OpenServer Server-Library/C Reference Manual, Sybase Inc.

[36] http://www.omg.org/

[37] John A. How, et. al., “Remote Participation Technical Infrastructure for the JET Facilities under FEDA”, 21st SOFT, Lisbon, Sep. 2000

[38] S.Hirano, “HORB: Distributed Execution of Java Programs”, WWCA’97, Tuskuba, Japan, March 1997

[39] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[40] M.Shoji, et. al.,“Video on Demand System for Acquiring and Distributing Plasma Image Data in Large Helical Device”, PCaPAC2000, Hamburg, Oct. 2000

[41] http://www.sinet.ad.jp/

[42] M.Emoto, et. al.,“High Quality Video Streaming System for Plasma Diagnostics using Super SINET”,Rev. Sci. Instr., **74** (2003), pp.1766-1769

[43] http://www.javasoft.com/products/servlet/index.html

[44] H.Zushi, et. al.,“Recent progress on TRIAM-1M”,Nucl. Fusion, **40** (2000), pp.1183-1196

[45] Equipe Tore Supra (prepared by F. Saint-Laurent), “Steady state operation and control experiments on Tore Supra”,Nucl. Fusion, **40** (2000), pp.1047-1055

[46] 濁川　和幸 他: 核融合科学研究所 技術研究会報告集 P-38 (2002)

[47] K.Takahata, et. al., “Cooldown Performance of an Inner Vertical Field Coil for the Large Helical Device”,IEEE Transaction on Magnetics, **32** (1996) pp.2252-2255

[48] 刈谷 丈治 他、“拡張性と柔軟性に富む LHD 超伝導コイル監視用システム”、プラズマ核融合, **74** (1998), pp.67-75

[49] Deering, S. ,Proceedings of ACM SIGCOMM8, 55 (August 1988)

[50] 江本 雅彦 他, “IP マルチキャストによる実時間計測システムと LHD 計測用計算機システムの統合化”, プラズマ核融合, **78** (2002), pp.1084-1092

[51] http://www.objectivity.com/

付 録B　解析済みデータ取出しプログラムのサンプル(Java)

Java による解析済みデータ取り出し簡易版プログラムを紹介する。本プログラムは JDK1.2.2 および 1.3.1 で動作を確認しているが、非標準クラスの sun.net.ftp.FtpClient を使用しているため環境によっては動作しない可能性がある。

このプログラムは解析情報サーバのプログラムの簡潔性を示すための例であり、最低限の機能しか提供しない。より完全なプログラムとして使うにはコマンドラインオプションの解析やエラー処理等を加える必要がある。

```
 1 import java.sql.*;
 2 import java.io.*;
 3 import java.util.*;
 4 import java.util.zip.*;
 5 import sun.net.ftp.FtpClient;
 6
 7 // 解析済みデータの取得クラス
 8 public class IGetfile {
 9     // データベースサーバ
10     static final String DBHOST = "server";
11     // データベース名
12     static final String DBNAME = "database";
13     //ユーザ名
14     static final String DBUSER = "scott";
15     //パスワード
16     static final String DBPASS = "tiger";
17
18     //使用法の出力
19     static private void usage(){
20         System.err.println("usage: java IGetfile -s shotno "
21                            + "-d diagnoame [ -m subshotno ] [ -o outputfle]");
```

```
22          System.exit(1);
23      }
24
25      static public void main(String args[]){
26          String shotno = null;
27          String subshotno = "1";
28          String diag = null;
29          String output = null;
30
31          int i = 0;
32          //コマンドラインオプションの解析
33          while ( i < args.length ){
34              if ( "-s".equals(args[i]) ){
35                  shotno = args[++i];
36              }else if ( "-m".equals(args[i])){
37                  subshotno = args[++i];
38              }else if ( "-d".equals(args[i])){
39                  diag = args[++i];
40              }else if ( "-o".equals(args[i])){
41                  output = args[++i];
42              }else{
43                  System.err.println("unknown option "+args[i]);
44                  usage();
45              }
46              i++;
47          }
48
49          if ( shotno == null || diag == null )
50              usage();
51
52          try {
53              Connection conn=null;
54              String username = System.getProperty("user.name");
55
56              DriverManager.registerDriver(new org.postgresql.Driver());
57              String url = "jdbc:postgresql://"
```

```
58              +DBHOST+"/"+DBNAME;
59
60          //データベースへの接続
61          conn =DriverManager.getConnection
62              (url,DBUSER,DBPASS);
63
64          System.out.println(">>information server connect: rdb_server="
65                             + Constants.DBHOST
66                             + "rdb_db=" + Constants.DBNAME
67                             + "rdb_account=" + Constants.DBUSER);
68
69
70          String sql = "select * from _locationfileinfo_ where "
71              + "diagnostics = '"+diag+"'"
72              + "and shotno = "+shotno
73              + "and subshotno = "+subshotno;
74
75          PreparedStatement stmt = conn.prepareStatement(sql);
76
77          //キュエリの発行
78          ResultSet rs = stmt.executeQuery();
79          String server = null;
80          String directory = null;
81          String filename = null;
82
83          //データの格納場所の取得
84          while ( rs.next() ){
85              server = rs.getString("server").trim();
86              directory = rs.getString("directory").trim();
87              filename = rs.getString("filename").trim();
88          }
89          rs.close();
90          conn.close();
91
92          //FTP によるデータの取得
93          FtpClient client = new FtpClient(server);
```

```
94              client.login("anonymous",username);
95              client.binary();
96              client.cd(directory);
97              InputStream in = client.get(filename);
98
99              // ZIP の展開
100             ZipInputStream zin = new ZipInputStream(in);
101             ZipEntry ent = zin.getNextEntry();
102             if ( output == null )
103                 output = ent.getName();
104             BufferedInputStream bis = new BufferedInputStream(zin);
105
106             byte buffer[] = new byte[1024];
107             int count;
108
109             BufferedOutputStream  os =
110                 new BufferedOutputStream(new FileOutputStream(output));
111
112             while ( (count = bis.read(buffer)) > 0 ){
113                                 os.write(buffer,0,count);
114             }
115             os.close();
116         }catch(Exception ex){
117             ex.printStackTrace();
118         }
119     }
120 }
```

使用法

java Igetfile -s ショット番号 -m サブショット番号 -d 計測名 -o 出力ファイル

(例)

```
% Java Igetfile -s 10000 -d thomson
```

付 録C　実験ログデータベース テーブル定義

カラム名	説明	データ型	値範囲
nDataCode		int	
nExecutionFlag	実行済みフラグ	int	"1(済),0(未)"
Status	状態フラグ	char(1)	
UpdateID	行 ID	int	
nShotnumber	ショット番号	int	
dDatacreationTime	日付時間	datetime	
nPSnumber	通電番号	int	
nCoolingnumber	冷却番号	int	
MagneticField	コイル用電源：磁場強度	float	
MagneticAxis	コイル用電源：磁気軸	float	
Quadruple	コイル用電源：四重極成分	float	
GasType	ガスパフ：ガス種	varchar(255)	
GasPuffBulb1Status	ガスパフ：ピエゾ 1 バルブ状態	varchar(255)	"ON,OFF"
GasPuffBulb2Status	ガスパフ：ピエゾ 2 バルブ状態	varchar(255)	"ON,OFF"
GasPuffBulb3Status	ガスパフ：ピエゾ 3 バルブ状態	varchar(255)	"ON,OFF"
GasPuffBulb4Status	ガスパフ：ピエゾ 4 バルブ状態	varchar(255)	"ON,OFF"
GasPuffBulb5Status	ガスパフ：ピエゾ 5 バルブ状態	varchar(255)	"ON,OFF"
GasQuantity	ガスパフ：入射ガス量	float	
GasPuffComment	ガスパフ：コメント	varchar(255)	
LIDACurrentMAX	LID：電流値	float	
LIDStatus	LID：ON/OFF	varchar(255)	"ON,OFF"
LIDComment	LID：コメント	varchar(255)	
Pellet1Time	ペレット：系統 1 入射時間	float	
Pellet1Status	ペレット：系統 1ON/OFF	varchar(255)	"ON,OFF"
Pellet2Time	ペレット：系統 2 入射時間	float	

Pellet2Status	ペレット：系統 2ON/OFF	varchar(255)	"ON,OFF"
Pellet3Time	ペレット：系統 3 入射時間	float	
Pellet3Status	ペレット：系統 3ON/OFF	varchar(255)	"ON,OFF"
Pellet4Time	ペレット：系統 4 入射時間	float	
Pelllet4Status	ペレット：系統 4ON/OFF	varchar(255)	"ON,OFF"
Pellet5Time	ペレット：系統 5 入射時間	float	
Pellet5Status	ペレット：系統 5ON/OFF	varchar(255)	"ON,OFF"
PelletComment	ペレット：コメント	varchar(255)	
ECHToshiba1Time	ECH：東芝 1 入射時間	float	
ECHToshiba1PulseWidth	ECH：東芝 1 入射パルス幅	float	
ECHToshiba1Power	ECH：東芝 1 入射電力	float	
ECHToshiba1Status	ECH：東芝 1ON/OFF	varchar(255)	"ON,OFF,AGI"
ECHToshiba2Time	ECH：東芝 2 入射時間	float	
ECHToshiba2PulseWidth	ECH：東芝 2 入射パルス幅	float	
ECHToshiba2Power	ECH：東芝 2 入射電力	float	
ECHToshiba2Status	ECH：東芝 2ON/OFF	varchar(255)	"ON,OFF,AGI"
ECHToshiba3Time	ECH：東芝 3 入射時間	float	
ECHToshiba3PulseWidth	ECH：東芝 3 入射パルス幅	float	
ECHToshiba3Power	ECH：東芝 3 入射電力	float	
ECHToshiba3Status	ECH：東芝 3ON/OFF	varchar(255)	"ON,OFF,AGI"
ECHGYCOM1Time	ECH：GYCOM-1 入射時間	float	
ECHGYCOM1PulseWidth	ECH：GYCOM-1 入射パルス幅	float	
ECHGYCOM1Power	ECH：GYCOM-1 入射電力	float	
ECHGYCOM1Status	ECH：GYCOM-1ON/OFF	varchar(255)	"ON,OFF,AGI"
ECHGYCOM2Time	ECH：GYCOM-2 入射時間	float	
ECHGYCOM2PulseWidth	ECH：GYCOM-2 入射パルス幅	float	
ECHGYCOM2Power	ECH：GYCOM-2 入射電力	float	
ECHGYCOM2Status	ECH：GYCOM-2ON/OFF	varchar(255)	"ON,OFF,AGI"
ECHCPI1RTime	ECH：CPI-1R 入射時間	float	
ECHCPI1PulseWidth	ECH:CPI-1R 入射パルス幅	float	
ECHCPI1RPower	ECH：CPI-1R 入射電力	float	
ECHCPI1Status	ECH：CPI-1RON/OFF	varchar(255)	"ON,OFF,AGI"
ECHComment	ECH：コメント	varchar(255)	
NBI1Time	NBI：1 号機入射時間	float	
NBI1PulseWidth	NBI：1 号機パルス幅	float	
NBI1Power	NBI：1 号機入射電力	float	
NBI1Stauts	NBI：1 号機 ON/OFF	varchar(255)	"ON,OFF"
NBI2Time	NBI：2 号機入射時間	float	
NBI2PulseWidth	NBI：2 号機パルス幅	float	
NBI2Power	NBI：2 号機入射電力	float	

NBI2Status	NBI：2 号機 ON/OFF	varchar(255)	"ON,OFF"
NBIComment	NBI：コメント	varchar(255)	
ICRF1Time	ICRF：1 号機入射時間	float	
ICRF1PulseWidth	ICRF：1 号機パルス幅	float	
ICRF1Power	ICRF：1 号機入射電力	float	
ICRF1Status	ICRF：1 号機 ON/OFF	varchar(255)	"ON,OFF"
ICRF2Time	ICRF：2 号機入射時間	float	
ICRF2PulseWidth	ICRF：2 号機パルス幅	float	
ICRF2Power	ICRF：2 号機入射電力	float	
ICRF2Status	ICRF：2 号機 ON/OFF	varchar(255)	"ON,OFF"
ICRF2Comment	ICRF：コメント	varchar(255)	
CoordinatorComment	コーディネータコメント	varchar(255)	
GAMMA	コイル用電源：ガンマ	float	
LIDB1CurrentMAX	LID：B1 電源電流最大値	float	
LIDB12CurrentMAX	LID：B1 電源電流最大値	float	
Pellet6Time	ペレット:系統 6-入射時間	float	
Pellet7Time	ペレット:系統 7-入射時間	float	
Pellet8Time	ペレット:系統 8-入射時間	float	
Pellet9Time	ペレット:系統 9-入射時間	float	
Pellet10Time	エレット:系統 10-入射時間	float	
NBICoordinatorComment	NBI:コーディネータコメント	varchar(255)	
ICRFCoordinatorComment	ICRF:コーディネータコメント	varchar(255)	
PelletCoordinatorComment	ペレット:コーディネータコメント	varchar(255)	
GasPuffCoordinatorComment	ガスパフ:コーディネータガス種	varchar(255)	
ECHCoordinatorComment	ECH:コーディネータコメント	varchar(255)	
Pellet6Status	ペレット:系統 6-ON/OFF	varchar(255)	
Pellet7Status	ペレット:系統 7-ON/OFF	varchar(255)	
Pellet8Status	ペレット:系統 8-ON/OFF	varchar(255)	
Pellet9Status	ペレット:系統 9-ON/OFF	varchar(255)	
Pellet10Status	ペレット:系統 10-ON/OFF	varchar(255)	
ExperimentTheme	実験テーマ	varchar(255)	
wp_max	プラズマ蓄積エネルギー（反磁性計測）の最大値 [KJ]	float	
time_wpmax	プラズマ蓄積エネルギー（反磁性計測）の最大の時間 [s]	float	

nl_max	線平均密度（ミリ波計測）の最大値 $[10^{19}/m^2]$	float	
time_nlmax	線平均密度（ミリ波計測）の最大の時間 [s]	float	
nets_max	密度（トムソン）の最大値 $[10^{19}/m^2]$	float	
time_netsmax	密度（トムソン）の最大の時間 [s]	float	
tets_max	電子温度（トムソン）の最大値 [KeV]	float	
time_tetsmax	電子温度最大（トムソン）の時間 [s]	float	
ticx_max	イオン温度（ＣＸＲＳ）の最大値 [KeV];	float	
time_ticxmax	イオン温度最大（ＣＸＲＳ）の時間 [s]	float	
ip_max	プラズマ電流（電磁計測）の最大値（絶対値が最大）[kA]	float	
time_ipmax	プラズマ電流（電磁計測）最大の時間 [s]	float	
ech1_pow	ポートスルー ECH パワー（ＥＣＨ１）[MW]	float	
ech2_pow	ポートスルー ECH パワー（ＥＣＨ２）[MW]	float	
ech3_pow	ポートスルー ECH パワー（ＥＣＨ３）[MW]	float	
ech4_pow	ポートスルー ECH パワー（ＥＣＨ４）[MW]	float	
ech5_pow	ポートスルー ECH パワー（ＥＣＨ５）[MW]	float	
ech6_pow	ポートスルー ECH パワー（ＥＣＨ６）[MW]	float	
nbi1_pow	ポートスルーＮＢＩパワー（ＮＢＩ１）[MW]	float	
nbi2_pow	ポートスルーＮＢＩパワー（ＮＢＩ２）[MW]	float	
ich1_pow	ポートスルーＩＣＨパワー（ＩＣＨ１）[MW]	float	
ich2_pow	ポートスルーＩＣＨパワー（ＩＣＨ２）[MW]	float	
ich3_pow	ポートスルーＩＣＨパワー（ＩＣＨ３）[MW]	float	

time_dis	放電時間 [s]	float	
ip_wpmax	最大エネルギー時のプラズマ電流 (電磁計測)[kA]	float	
nl_wpmax	最大エネルギー時の線平均密度 (ミリ波計測)[$10^{19}/m^2$]	float	
net0ts_wpmax	最大エネルギー時の密度 (トムソン@真空磁気軸)[$10^{19}/m^2$]	float	
te0ts_wpmax	最大エネルギー時の電子温度 (トムソン@真空磁気軸) [KeV]	float	
ne7ts_wpmax	最大エネルギー時の密度 (トムソン@真空 ρ =0.7)[$10^{19}/m^2$]	float	
te7ts_wpmax	最大エネルギー時の電子温度 (トムソン@真空 ρ =0.7)[KeV]	float	
ne1ts_wpmax	最大エネルギー時の密度 (トムソン@真空 ρ =1)[$10^{19}/m^2$]	float	
te1ts_wpmax	最大エネルギー時の電子温度 (トムソン@真空 ρ =1)[KeV]	float	
nedpro_wpmax	最大エネルギー時の密度 (静電プローブ@ダイバータ)[$10^{19}/m^2$]	float	
tedpro_wpmax	最大エネルギー時の電子温度 (静電プローブ@ダイバータ)[KeV]	float	
ti0cx_wpmax	最大エネルギー時のイオン温度 (CXRS@真空磁気軸)[KeV]	float	
radpow_wpmax	最大エネルギー時の放射パワー (ボロメータ)[kW]	float	
c3_wpmax	最大エネルギー時の CIII 発光強度 [A.U.]	float	
o5_wpmax	最大エネルギー時の CV 発光強度 [A.U.]	float	
echpow_wpmax	最大エネルギー時のポートスルー ECH パワー (積算)[MW]	float	

nbipow_wpmax	最大エネルギー時のポートスルー NBI パワー (積算)[MW]	float	
nbipowst_wpmax	最大エネルギー時のシャインスルー NBI パワー (積算)[MW]	float	
ichpow_wpmax	最大エネルギー時のポートスルー ICH パワー (積算)[MW]	float	

付 録D　マルチキャストパケットフォーマット

マルチキャストパケットは以下に示すフォーマットからなり、パケットの種類を判別する4バイトの識別子と、パケットサイズを記述した4バイトのヘッダを有する。8バイト目以後のはPacketIDに依存する。マルチキャストアドレスは225.1.1.1～225.1.1.3を使用し、TTL(Time To Live)は4とする。また、バイトデータの格納順はリトルエンディアンで符号あり、浮動小数はIEEE754を用いる。

現在パケットIDは1～3まで定義されており、それぞれのIDに対するフォーマットは以下のようになっている。

開始-終了 (byte)	サイズ (byte)	型	説明
0-3	4	signed int	パケット ID
4-7	4	signed int	パケットサイズ
8-	不定	パケット ID に依存	

表 D.1: マルチキャストパケットフォーマット

開始-終了 (byte)	サイズ (byte)	型	説明
8-11	4	char[4]	“1.0” (固定)
12-15	4	signed int	データ ID
16-19	4	time_t	基準時:1970 年 1 月 1 日からの経過秒数
20-27	8	signed long long	基準時からの経過時間 (単位:nsec)
28-31	4	signed int	変数の数
32-39	8	double	変数 1 の値
40-47	8	double	変数 2 の値

表 D.2: リアルタイムモニタリングパケットフォーマット ID=2

開始-終了 (byte)	サイズ (byte)	型	説明
8-11	4	signed int	シーケンス状態 bit 0 ... シーケンス実行中 bit 1 ... 放電終了
12-15	4	signed int	ショット番号
16-19	4	signed int	サブショット番号

表 D.3: 実験シーケンスパケット ID=3

開始-終了 (byte)	サイズ (byte)	型	説明
8-43	36	char[]	計測名 (NULL で終わる)
12-15	4	signed int	ショット番号
16-19	4	signed int	サブショット番号

表 D.4: 解析済みデータ登録通知 ID=1

関連図書等

[1] A. Iiyoshi et al., "An Overview of the Large Helical Device Project", Nucl. Fusion, **39** (1999), pp.1245-1256

[2] http://www.postgresql.org/

[3] K.Yamazaki, et. al., "Design of the central control system for the Large Helical Device (LHD)",Nucl. Instr. and Math. in Phys. Res. **A 352** (1994) pp.43-46

[4] H.Yamada, et. al., "Design of central control and man-machine interface systems for large helical device", Fusion Technology 1996, 1997, Elsevier Science, pp. 945-948

[5] H. Nakanishi et. Al., "Distributed processing and network of data acquisition and diagnostics control for large helical device (LHD)", Fus. Eng. Des., **43** (1999), pp. 293-300

[6] 山口他,"超伝導コイル実験監視システム", プラズマ核融合, **73** (1997), pp.335-342

[7] S. Yamaguchi, et al."Control and Plasma Data Acquisition System for LHD Experiment", Fus. Eng. Des., **48** (200) pp.9-15

[8] J.A. Stillerman and T.W. Fredian, "MDSplus Data Acquistion System", 11th Topical Conference on High Temperature Plasma, Monterey, 1996

[9] T.Fredian, "C-Mod Data Acquisition System" High Temperature Plasma Diagnostics, Monterrey, May 1996,

[10] S.Ohdachi, Annual Report of National Institute for Fusion Science, APRIL 1998- MARCH 1999

[11] D.P. Schissel et. al., "Recent Enhancements to Analyzed Data Acquisition and Remote Participation at the DIII-D National Fusion Facility", Fus. Eng. Des., **56** (2001), pp. 1005-1009

[12] J.G. Krom, "The evolution of control and data acquisition at JET", Fus. Eng. Des., 43 (1996), pp.265-273

[13] 青柳哲雄,"JT-60 のデータ処理", プラズマ核融合, **72** (1996), pp.1370-1375

[14] T.Matsuda et. al., "Status of JT-60 data processing system", Fus. Eng. Des., **48** (2000), pp.99-104

[15] B. Guillerminet, "The acquisition system for Tore Supra 1000 s discharges", Fus. Eng. Des., **48** (2000), pp.155-161

[16] http://nssdc.gsfc.nasa.gov/cdf/html/tech_brief.html

[17] http://www.cv.nrao.edu/fits/documents/standards/fits_standard.ps

[18] http://members.aol.com/rmcdjcamp/index.htm

[19] http://www.unidata.ucar.edu/packages/netcdf/guide.txn_toc.html

[20] http://pds.jpl.nasa.gov/stdref/

[21] http://xml.gsfc.nasa.gov/XDF/

[22] R.G.G.Cattel, オブジェクトデータベース標準:ODMG-93 Release 1.1, 共立出版

[23] http://www.mysql.com/

[24] J. Schachter, et al., "Data Analysis Software Tools for Enhanced Collaboration at the DIII-D National Fusion Facility", Fus. Eng. Des., **48** (2000), pp.91-98

[25] http://www.gtk.org/

[26] http://www.gnome.org/

[27] G.Manduchi, "The Java Interface of MdsPlus: towards an Unified Approach for Local and Remote Data Access", Fus. Eng. Des., **48** (2000), pp.163-170

[28] http://www.gnu.org/copyleft/gpl.html

[29] http://www.python.org/

[30] まつもとゆきひろ,オブジェクト指向スクリプト言語 Ｒｕｂｙ,アスキー

[31] http://www.tcl.tk/

[32] M.Emoto et. al., "A trial to combine heterogeneous computer system in NIFS", Fus. Eng. Des., **48** (2000), pp. 83-89

[33] EMOTO Masahiko et. al.,"A PROXY SERVER FOR A REAL-TIME MONITORING SYSTEM", PCaPAC2000, Hamburg, Oct. 2000

[34] M.Emoto et. al.,"3D REAL-TIME MONITORING SYSTEM FOR LHD PLASMA HEATING EXPERIMENT", Fus. Eng. Des., **56** (2001), pp.1017-1021

[35] OpenServer Server-Library/C Reference Manual, Sybase Inc.

[36] http://www.omg.org/

[37] John A. How, et. al., "Remote Participation Technical Infrastructure for the JET Facilities under FEDA", 21st SOFT, Lisbon, Sep. 2000

[38] S.Hirano, "HORB: Distributed Execution of Java Programs", WWCA'97, Tuskuba, Japan, March 1997

[39] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[40] M.Shoji, et. al.,"Video on Demand System for Acquiring and Distributing Plasma Image Data in Large Helical Device", PCaPAC2000, Hamburg, Oct. 2000

[41] http://www.sinet.ad.jp/

[42] M.Emoto, et. al.,"High Quality Video Streaming System for Plasma Diagnostics using Super SINET",Rev. Sci. Instr., **74** (2003), pp.1766-1769

[43] http://www.javasoft.com/products/servlet/index.html

[44] H.Zushi, et. al.,"Recent progress on TRIAM-1M",Nucl. Fusion, **40** (2000), pp.1183-1196

[45] Equipe Tore Supra (prepared by F. Saint-Laurent), "Steady state operation and control experiments on Tore Supra",Nucl. Fusion, **40** (2000), pp.1047-1055

[46] 濁川　和幸 他: 核融合科学研究所 技術研究会報告集 P-38 (2002)

[47] K.Takahata, et. al., "Cooldown Performance of an Inner Vertical Field Coil for the Large Helical Device",IEEE Transaction on Magnetics, **32** (1996) pp.2252-2255

[48] 刈谷 丈治 他、"拡張性と柔軟性に富む LHD 超伝導コイル監視用システム"、プラズマ核融合, **74** (1998), pp.67-75

[49] Deering, S. ,Proceedings of ACM SIGCOMM8, 55 (August 1988)

[50] 江本 雅彦 他, "IP マルチキャストによる実時間計測システムと LHD 計測用計算機システムの統合化", プラズマ核融合, **78** (2002), pp.1084-1092

[51] http://www.objectivity.com/