LHD 定常実験におけるリアルタイムモニタリングシステムの開発)

○大砂真樹 A)、中西秀哉 A)、小嶋護 A)、今津節男 B)、吉田正信 C)

A) 核融合科学研究所 LABCOM グループ

B)(有)プレテック

C) 伊藤忠テクノサイエンス(株)

概要

核融合科学研究所の LHD では､今年度実験サイクルの最終週に 300 秒の定常実験を行う｡その定常実験中、ユーザーがリアルタイムでプラズマデータを監視することができるように、クライアント／サーバ型のリアルタイムモニタリングシステムを開発した。

クライアント及びサーバには通常の PC を利用し、WE7000 や CompactPCI などのリアルタイムに計測データを送出できるデジタイザフロントエンドと接続されたサーバから、リアルタイムにデータを取得し表示するシステムであり、その開発について紹介する。

1　システムの概要

このモニタリングシステムは、予定されている放電時間 300 秒に限らず、通常の 10 秒放電より長いパルスでの実験において、実験データのリアルタイムでの可視化を目的とする。

1.1　システムの要件

- 》研究所内のどの PC 上でも、データを参照することができるように、クライアント／サーバ型のシステムとする。
- 》一つのデジタイザに対して、一つのサーバが接続するため、そこから得られたデータは、リアルタイムにクライアントに送出するだけでなく、そのデータを後々にも参照できるように、サーバはリアルタイムなデータ送出とは別にローカルディスクにデータを保存する必要がある。
- 》データ取得後、可能な限り短い時間で可視化できるようにする。目標は実際のデータ取得時間に対して 1 秒以内とする。また、リアルタイムでのデータ監視が目的であるので、データ表示は 1 秒あたり 10 回以上のリフレッシュを行うことが目標である。

1.2　使用するハードウェア

リアルタイムモニタリングで使用するハードウェアは、クライアント及びサーバ、そしてサーバに接続されたデジタイザフロントエンドである。

クライアント及びサーバには、通常の PC を使用する。サーバは、データの取得・送出・圧縮・保存といった主なジョブがあり、大きな負荷がかかるが、昨今の PC の性能では大きな問題とはならない。

デジタイザフロントエンドには、今年度の実験では、WE7000 と CompactPCI を使用する。昨年度までは CAMAC を使用していたが、CAMAC はデータを AD 変換しつつバッファメモリに貯め、全データ収集後にデータを PC が取り込む、というバッチ処理的な動作をするため、リアルタイム計測はできない。WE7000 及び CompactPCI は、AD 変換しながら PC にデータを送り出すことが可能であるため、今回のようなリアルタイムモニタリングシステムにも使用することができる。

1.3　システム全体図

システムの全体図は図 1 のようになり、これにはリアルタイムモニタリングシステムだけではなく、既存のデータシステムも含まれる。

サーバ PC ではデータ収集サービスがデジタイザからのデータを取得し、メモリを共有したサービス群がそのデータをそれぞれ処理する。クライアントにはデータ取得用の DLL を配布し、それを利用してデータを参照する。現在、通常のデータビューアには PV-Wave や IDL などの可視化ソフトを使用し、リアルタイムビューアは独自に開発したものを使用している。

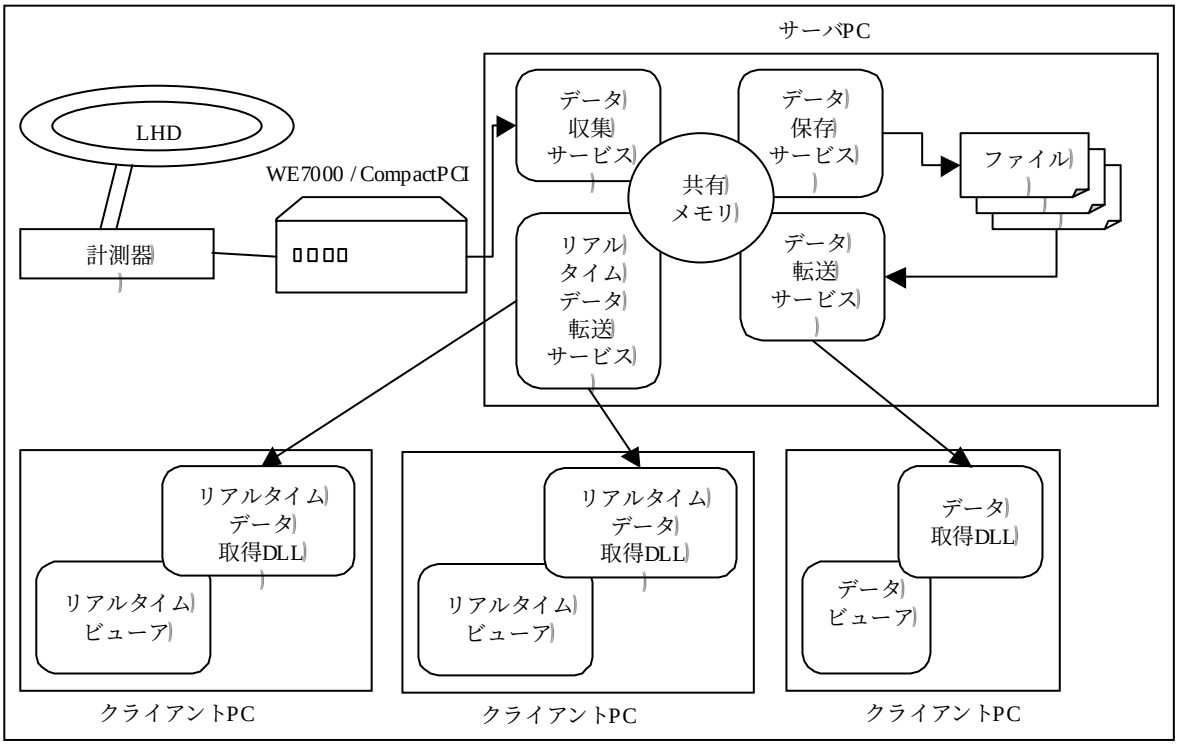

図 1　リアルタイムモニタリングシステム全体図

2　クライアント／サーバ間の通信

開発には Visual C++を用いて、サーバ側のリアルタイムデータ転送サービス、クライアント側の DLL 及びビューアを作成した。クライアント／サーバ間の通信は、まず TCP によってクライアントがサーバに接続し、オープン要求・パラメータ設定要求・データ転送要求などをコマンドとして送り、データ転送要求送信後、クライアントが指定した UDP ポートに対して、サーバがデータを送出する。

これらの処理は、クライアント上では DLL として作成した関数群が行い、リアルタイムビューアはこの関数を呼び出すことで、リアルタイムにデータを取得することができる。DLL として作成した理由は、たとえば将来リアルタイムデータ解析アプリケーションなどが必要になった場合に、開発コストを減らすためである。

クライアント／サーバ間のデータ転送開始までの通信手順の概要は図 2 のようになる。転送終了は同様にクライアント側からデータ転送停止要求とクローズ要求を送り、その後接続を切断する。

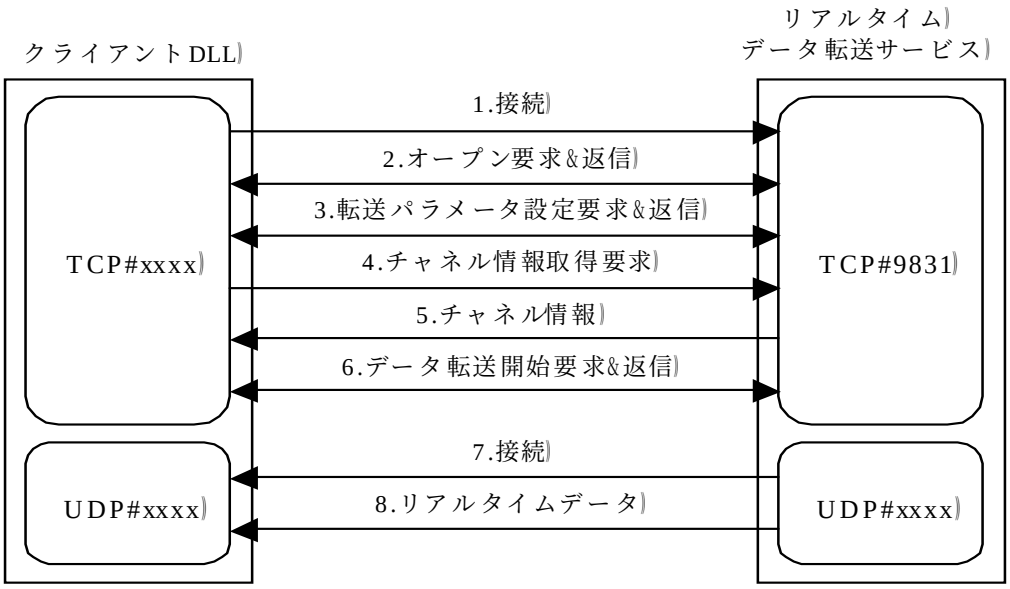

図 2 クライアント／サーバ間の通信手順概要

2.1 制御プロトコル

制御のための通信は TCP を用い、設計上制御経路と呼んでいる。

手順としては、TCP ソケットにより接続後、クライアントが要求コマンドを送り、サーバはそのコマンドに対する返信を行う。コマンド及び送信はテキストにより送られ、決められたフォーマットで送信・返信を行う。通信手順のいずれかでエラーがサーバより返信された場合は、それ以上の処理は行わない。

2.2 リアルタイムデータ送信プロトコル

データ送信のための通信は UDP を用い、設計上データ経路と呼んでいる。

クライアントは、制御経路からデータ転送開始要求コマンドを送信するまでに、データ受信のための UDP ポートを用意し、データ転送開始要求コマンドを送り、サーバからの接続・データ送信を待つ。

データ送信は、決められたフォーマットのブロック単位で行われる。制御手順中、転送パラメータ設定要求を送信するときにパケットサイズを指定し、パケットサイズがブロック長より小さい場合は、いくつかのパケットに分割されて送信される。

データ送信は、クライアントがデータ転送停止要求コマンドを送信するまで連続的に行われる。UDP を用いるためパケットがいくつか失われる可能性が常にあるが、パケットロスの場合はそのパケットの含まれるデータブロック全体を破棄する。リアルタイムでのデータ通信が最優先目的であるので、データの再送信などの処理は行わない。

3 リアルタイムデータの可視化

データは DLL 内の関数によってリアルタイムに受信するが、そのデータを処理するのは DLL 関数を呼び出した側の仕事となる。将来的には、リアルタイムデータを一時解析するようなアプリケーションも必要になると思われるが、現状の最優先利用方法としては、表題どおりリアルタイムデータを監視するものとなる。

今回、そのための可視化アプリケーションは、Visual C++と OpenGL を用いて作成した。リアルタイムでデータをプロットできるようなユーティリティがあれば、開発コストを減らすために利用したかったが、適当なものを見つけられなかった。OpenGL はグラフィックライブラリであるが、基本的に 3D 表示に向いた設計

となっているため、このモニタリングシステムでの目的である時間軸とデータ軸の 2D プロットにはさほど向いていない。将来的になにか適当なユーティリティがあれば移行する。

3.1　開発環境

開発は Windows2000 上の Visual Studio V6.0 で行い、言語は Visual C++、グラフィックライブラリとして OpenGL を用いた。また、将来 UNIX、Linux、MacOS などに移植する場合のコストを現象させるため、GLUT(OpenGL Utility Tool)を利用した。GLUT は OpenGL ライブラリを利用する場合に、よりわかりやすくソースを書くためのユーティリティ関数群であるが、Windows 以外でも利用可能で、OpenGL の機種依存性をソースレベルで大きく低減させることができる。

リアルタイムモニタリングのための可視化アプリケーションは、データ通信にはデータ転送 DLL の関数を呼び出し、グラフィック表示には OpenGL 及び GLUT の関数を呼び出すため、本体自体はソースも実行ファイルも非常に小さくなっている。

3.2　可視化アプリケーションの現状・問題点

リアルタイムに通信しながらリアルタイムにグラフィック表示を行うため、処理が非常に重く、目標としていたパフォーマンスが出ていない。データ取得後のデータ表示までの時間のずれは、約 1 秒となっておりまずまずであるが、画面表示が 1 秒あたり数回ほどしかリフレッシュしない。今後、チューニングと適当なユーティリティがあればそれを利用することで、1 秒あたり 10 回以上のリフレッシュを目指す。

また、UDP によるデータ送信は、予想よりもパケットロス頻度が高い。ネットワーク上で近いクライアント／サーバ間であればさほど問題にはならないが、ネットワーク距離が大きくなった場合にはパケットサイズの最適化など対応策を考える必要がある。

4　まとめ

- 》LHD 定常実験のため、リアルタイムにデータを表示するモニタリングシステムを作成した
- 》クライアント／サーバ型の設計とし、ネットワークを介して、リアルタイムデータを表示できる
- 》クライアントとサーバには PC を用い、ハードウェアコストを抑えた
- 》制御とデータの二つの経路にわけ、それぞれの性質によって TCP と UDP の両方を利用した
- 》表示に、よりリアルタイム性を持たせるため、チューニングが必要
- 》データ送出には UDP を使用しているため、パケットロスへの対応が必要