マルチクライアント環境における実験パラメータ配信システムの構築

○小川英樹、井上知幸、鷹見重幸、小平純一

核融合科学研究所 技術部

概要

核融合科学研究所で行っているプラズマ実験において、実験番号・シーケンス状態・磁場条件は、計測システムの制御や実験データの解析等に利用される重要なパラメータである。これらの情報は実験開始当初からファイル共有等により配信されていたが、クライアント数の増加や機能の追加等によりシステムが複雑化していた。今回 Apache と PostgreSQL を用いてシステムの見直しを行ったので、その詳細について報告する。

1　既存のシステム構成

現在、実験パラメータ配信システムは約 30 台のクライアントに利用されており、クライアント環境として、OS に Windows, UNIX、開発ツールに Visual Basic, Visual C++, PV-WAVE, LabVIEW , Fortran など、さまざまな環境が混在している。ここで、実験パラメータとは「実験番号」と「実験シーケンス状態」を指しており、システム改良前はこれらのデータは Windows の共有ファイルを通して各クライアントに配信されていた(図1)。共有ファイル配信では、サーバ側でファイル書込みの前後 0.5 秒間、LOCK という名前の空ファイルを作成して読み込み処理と重ならないようにしていたが、LOCK ファイルが消えずに残ってしまい、書込みタスクが蓄積するという問題が度々発生していた。

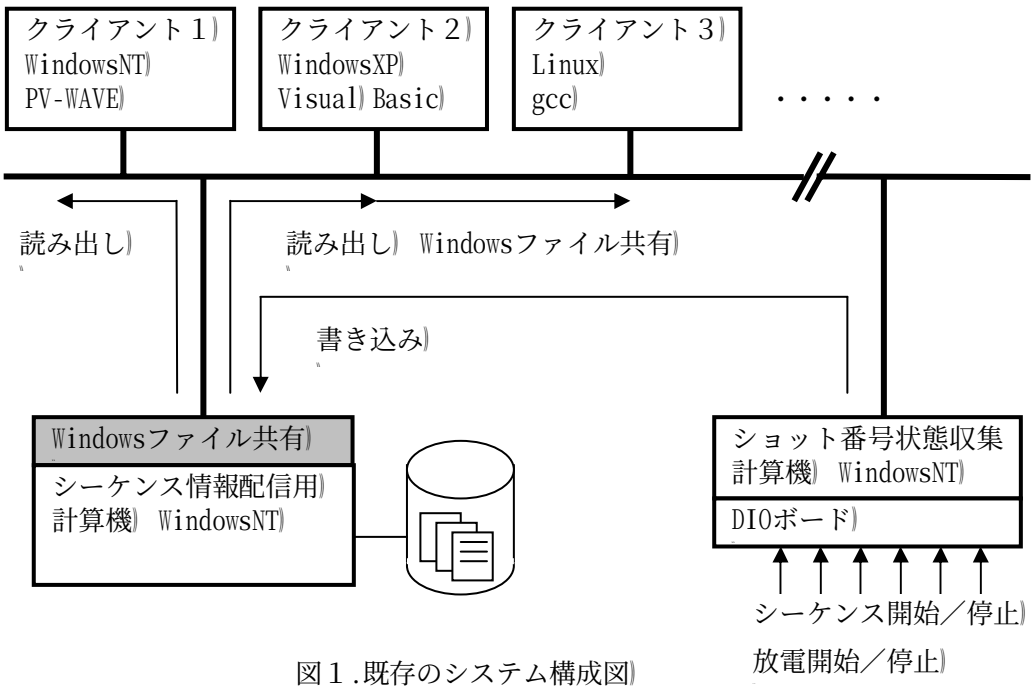

図１.既存のシステム構成図

2　新規システムの検討

我々は上記の問題点を解決するため、新しいデータ配信の仕組みとして、自作ソケットアプリケーション、

IP マルチキャスト、データベース(以下 DB と呼ぶ)の３つを挙げ、実験パラメータ配信システムに要求される表１に示す必要条件から、比較・検討を行った。

表 1.) 新規システム構成案の比較

)	必要条件)	自作ソケット) アプリケーション)	IP マルチキャスト)	データベース)
信頼性)	仕組みそのものの持つ信頼性が高いこと。)	○)	△)	○)
マルチプラットフォーム)	クライアントの OS や開発言語に依存しないこと。)	○)	○)	○)
コスト)	現実的なコストで実現可能であること。)	○)	○)	○)
負荷)	絶対的な負荷が小さく、クライアントが増えても負荷が大きく変化しないこと。)	△)	○)	△)
応答性)	最大 80 クライアントで１秒以内の応答性があること。)	○)	○)	△)
開発容易性)	既存のシステムからの移行が容易なこと。分担して開発可能なこと。)	×)	△)	○)
メンテナンス性)	接続管理やバックアップが容易に行えること。)	×)	×)	○)

まず、ソケット通信アプリケーションを自作する場合、必要な機能だけを組み込むことで高速なシステムを構築可能である。また、TCP/IP は多くの OS・開発環境でサポートされているため、クライアントの環境にも影響されにくい。しかし、自作プログラムの場合、セキュリティーの管理やデバッグを自分たちで行う必要があるため、他の仕組みと比較して開発に時間がかかってしまう。マルチキャスト配信は、クライアントが増えてもサーバの負荷に影響されにくいが、ネットワークのチューニングが必要であることや、通信プロトコルとして UDP を使用するため通信の確実性に欠けるという問題がある。一方、DB を用いる方法は多重アクセスの際の排他処理および通信機能を標準で備えているため、比較的簡単に信頼性の高いシステムを構築可能である。これらの特徴から、新規システムに DB の案を採用し、使用する DB については所内で開発経験のある PostgreSQL を選択した。

3　クライアントとの取合い仕様

クライアントプログラムの変更は、原則としてクライアントとなる各機器担当者が行うため、取り合い仕様の検討に於いては、できるだけ手間と予算をかけずに変更可能な仕組みを考える必要がある。クライアントから DB へのアクセス方法としては、当初 ODBC、libpq システムコールなどを検討していたが、セキュリティー確保のため、データベースに直接アクセスするのではなく、一旦 TCP/IP ソケットで Web サーバ(Apache2.0)に接続し、PHP 経由でデータベースにアクセスする方式を採用した。実際には、クライアントにライブラリファイル(Windows の場合は Visual Basic, Visual C++, Delphi, LabVIEW, PV-WAVE,IDL に対応した DLL ファイル、UNIX の場合は gcc, g77 に対応した C のソースファイル)を配布し、クライアントプログラム内で関数を呼び出すことにより各種パラメータを取得することとした(図 2)。さらに、利用マニュアルおよびサンプルプログラムもあわせて配布することで、クライアント側のプログラム移行作業の円滑化を図った。

既存のシステムは、当面の間新規システムと並行して動作させ、信頼性が確認された時点で順次クライア

ントプログラムの移行を行なう。

なお、今回のシステム移行にあたり、クライアント担当者からの要望を受け、同システムで磁場パラメータ(Bt, Rax, Bq, Gamma)の配信サービスも合わせて開始することにした。

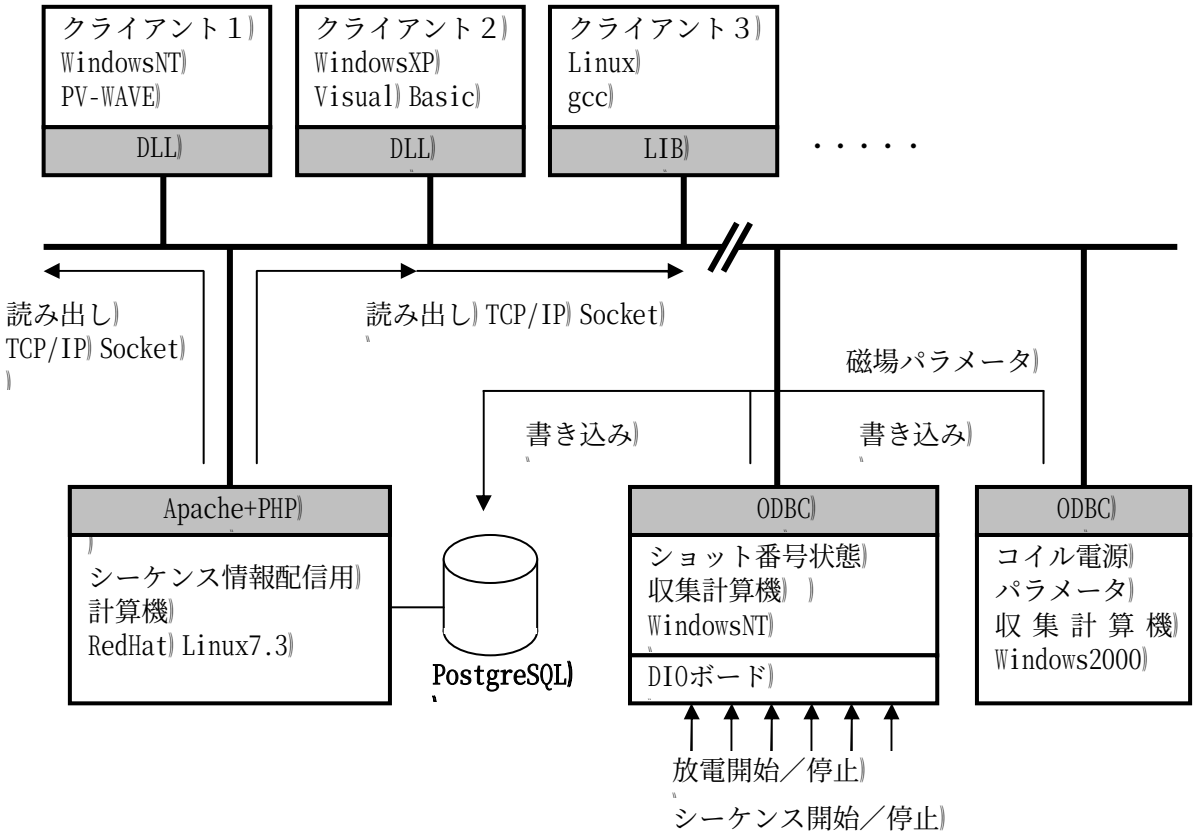

図 2.新規システム構成図

4　パフォーマンステスト

上記の構成で既定のパフォーマンスが得られるかどうかを確認するため、パフォーマンステストを行った。

4.1　テスト条件

以下の計算機環境で、クライアントから一秒周期でサーバに問い合わせを行うプログラムを稼動させ、サーバの CPU 負荷率とロードアベレージを測定した。立上げるプログラムの数は 20、40、60、80、100 の 5 つの場合についてテストを行った。

サーバとクライアントに使用したマシンのスペックは以下の通りである。立上げるプログラムの数が 40 以下の試験ではクライアント 1 のみを使用し、それ以上の試験ではクライアント 2 と 3 で立上げるプログラムの数を同じになるようにした。

サーバ　CPU：Pentium4 1.8GHz、 メモリ：512Mbyte、OS：RedHatLinux7.3

クライアント１　CPU：Pentium4 800MHz、メモリ：512Mbyte、OS：WindowsXP Professional

クライアント２　CPU：Pentium4 800MHz、メモリ：128Mbyte、OS：Windows2000 Professional

クライアント３　CPU：Pentium4 800MHz、メモリ：128Mbyte、OS：Windows2000 Professional

4.2　チューニング

パフォーマンス改善と信頼性向上のため、以下の 2 点についてシステムのチューニングが必要であった。

- RedHatLinux7.3 はデフォルトでは午前 4 時にログ解析処理などの cron 処理を行うため激しいディスクアクセスが発生し、OS の動作が遅くなってしまう。サーバに常時接続する場合には、このことにより通信トラブルなどの問題を引き起こす恐れがある。そのため、/etc/crontab 等を書き直し、必要な cron 処理だけを動作させるようにした。
- PostgreSQL の設定ファイル postgresql.conf において最大同時接続数(max_connections)をデフォルトの 32 から 128 に、共有バッファサイズ(shared_buffers)をデフォルトの 64 から 256 に修正した。

4.3 テスト結果と考察

まず、CPU 負荷率はクライアント台数が 80 台に達するまではほぼ直線的に増加し、その後 70%あたりで飽和する傾向が見られた。ロードアベレージについては 100 台までほぼ直線的に上昇した(図 3)。このことから、クライアントが 80 台を越えたあたりから、ロードアベレージの上昇が原因で、CPU の処理が追いつかなくなってきていると考えられる。

我々のシステムには現在約 30 台のクライアントが接続されているが、少なくとも 80 台までは 1 秒間隔で当初予定していたパフォーマンスをクリアし、問題なく動作することを確認した。また、クライアント 70 台での耐久試験では、1 週間ダウンすることなく稼動することを確認した。

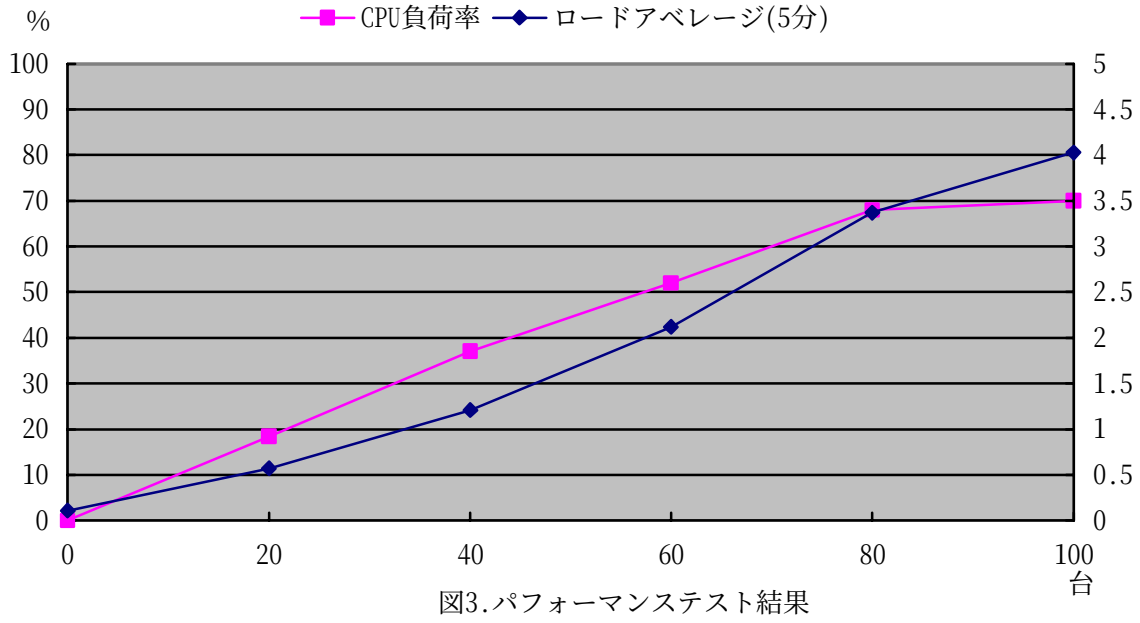

図3.パフォーマンステスト結果

5 まとめと現状

- データ配信に PostgreSQL を用いることで、既存の仕組みで発生していた排他処理の問題が解決し、信頼性が大幅に向上した。
- クライアントとの取合いに Apache、PHP を用いることで、よりセキュアでメンテナンス性に優れたシステムを構築することが出来た。
- クライアント担当者にライブラリ、サンプルプログラム、マニュアルを配布したことにより、既存システムからスムースにシステム移行が出来た。
- サービス開始後もコイル電流値や更新時刻の追加など、クライアントから機能追加の要望があったが、ライブラリファイルをバージョンアップし、クライアントに再配布することで対応した。
- 今年度のプラズマ実験から運用を開始し、クライアント台数約 30 台で現在までトラブルなく稼動している。