分散オブジェクトを用いた RIBF 加速器電源制御システムの開発

田辺敏也 A)、益岡俊勝 A)、熊谷桂子 A)、込山美咲 A)、吉田光一 B)、江本隆 A)、加瀬昌之 A)
理化学研究所　A)加速器基盤研究部、B)放射線研究室
〒351-0198 埼玉県和光市広沢 2－1

概要

理化学研究所ＲＩビームファクトリー（RIBF）計画[1]が当初の予定より縮小、延期された為、次期制御系の製作も電磁石電源制御の部分のみに制限されたかたちで実施された。当所では Common Object Request Broker Architecture (CORBA)を通信基盤においた分散オブジェクトを用いたシステムのＲ＆Ｄを 1998 年末より行ってきており[2-4]、今回の実装でもそのシステムの改善版を作成するかたちで進めた。現有システムでは Experimental Physics and Industrial Control System (EPICS)[5]を使用しているが、EPICS、及び CORBA と Java についての特徴の詳細はこの場では割愛し参考文献[6,7]をご覧いただきたい。

1　電源制御システムの要件

加速器制御システムには単純な性能用件以外にも下記のような要件が求められる。
（1）システムの拡張・改造が柔軟に行える。
（2）特定のハードウエア・ソフトウエアに依存しない。
（3）保守が容易である。
最初に製作されたシステムが、後年において修理拡張する際に同じ製品が存在する可能性は年々低下している。同様の機能を異なったメーカーの製品で達成する際にソフトウエア全体を書き換える必要があると非常に高価なことになる。その為に一般の情報技術として発展していて、特定の OS に依存していない分散オブジェクト技術である CORBA を利用することで一部の変更が全体に波及しないフレームワークを構築することができる。GUI に関しては、これまでは Windows しか利用できない VisualBasic や表現力の乏しい X-windows に代わり、言語自身に GUI、ネットワーク機能、マルチスレッド機能を内包し、特定の OS に依存しない Java が世の中の主流になってきており、諸外国で開発されたソフトウエアの利用には最適である。また、開発に際して注意した他の点としては、ドキュメントを常にアップデートできる体制をつくり、拡張や改造の結果を反映させることがあり、JavaDoc や Doxygen を用いた Documentation の構築を開発と同時に行った。

2　システム構成

2.1　全体構成

制御システムの全体構成（予定部分も含む）は図１に示す。GUI はクライアントとして、またアプリケーションサーバー(AS)は機能に応じて、クライアントにもサーバーにもなり得る。残りの CORBA オブジェクトはサーバーとして機能する。データベース（DB）は AS のバックエンドとして動作するため、他のオブジェクトからは直接アクセスはできない。図１内の CORBA ラッパーオブジェクトと CORBA-EPICS サーバーは R&D で製作しているが、現時点では N-DIM と GPIB は既存施設の EPICS でデータを取得している。

2.2　VME 関連機器

当初の RIBF 計画では、第二期計画としてシンクロトロンリング群を用いた施設を建設する予定であった。その為リアルタイム OS の使用を前提に設計をしたことからフィ

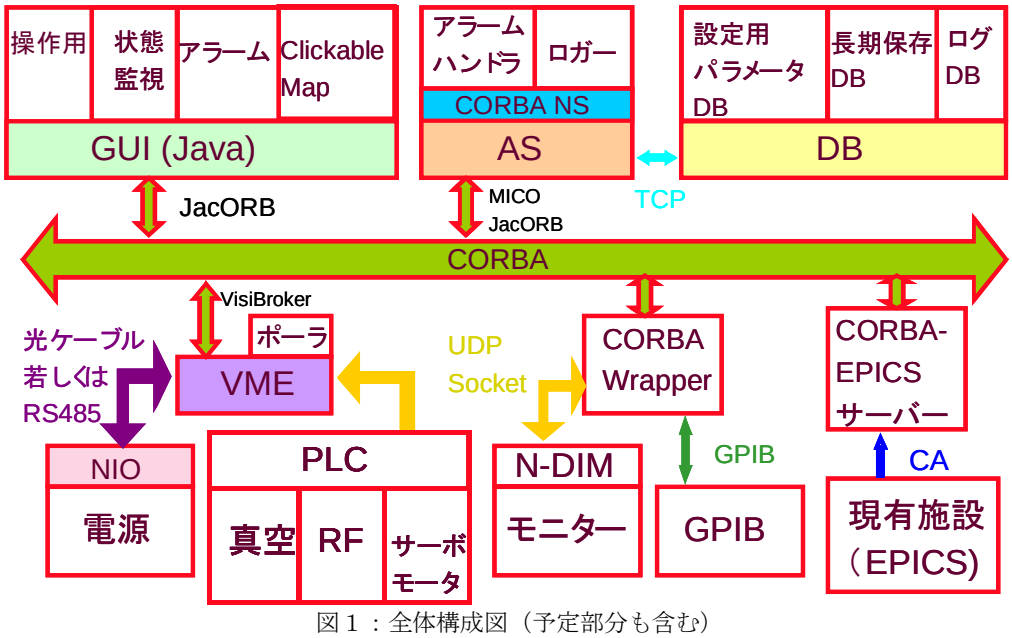

図１：全体構成図（予定部分も含む）

表２：使用した機器とソフトウエア（太字が有料ソフト）

機器類	項目	使用したソフトウエア
クライアント PC　(Java 導入済み)	OS	**Windows2000/Mac OS-X**/Linux
	Programming Language	Java1.4
	CORBA	JacORB
データベース [Dell PowerEdge2650]	OS	**Miracle Linux2.1**
	DB Software	**Oracle 9i (9.0.1)**
コントローラ（VME64x） [Motorola MVME2400]	OS	**vxWorks5.4**
	Programming Language	**C++ / C**
	CORBA	**VisiBroker for Tornado 4.1c**
	XML Parser	libXML2.5.2(modified for vxWorks)
アプリケーションサーバ [Dell PowerEdge4600]	OS	**Red Hat Linux Advance Server 2.1**
	Programming Language	GNU C++ / GNU C / Java1.4/ JDBC1.2
	Web server	Apache、Tomcat
	DNS	BIND
	NFS	Samba、Netatalk
	DHCP	dhcpd
	CORBA	MICO2.3.9/ JacORB1.4.1
	Others	GNU Make

ールドコントローラには VME と VxWorks という組み合わせを選択した。また、電源用の I/F である NDS 社のネットワーク IO システム(NIO)[8]　が VME 専用で、VxWorks のドライバーが既に用意してある点も選択の理由であった。しかし、最初の段階からサイクロトロン群とビームライン群のみの制御に限定して進めた場合にはインテル系の CPU と Linux 等の非リアルタイム OS の組み合わせも可能であったと思われる。ただし、ドライバーソフトの開発の費用を考慮すると全導入管理費用が安価になるかどうかは定かではない。今回使用した個々のハードウエアやソフトウエアの型式等は表２に示している。

VME-CPU ボードは Motorola 社の MVME2400 シリーズで、RAM は 512MB 搭載している。これは 256MB のモデルが生産中止になったためで、実際のメモリー領域は 256MB しか使用していない。VME64x シャシー（１５スロット）は活線挿抜可能な冗長電源を使用しておりメンテナンス性を考慮している。バックプレーンボードは、Ｊ１／Ｊ２／Ｊ０一体型である。また、過電流回路、過電圧保護回路、アラーム機能（出力異常出力）付で、DC 出力は 5V35A, ＋12V4A, －12V1A, 3.3V25A に対応する。19 インチラックは絶縁トランス及びノイズフィルター付で、電気系統は UPS 系と Non-UPS 系の２系統を用意している。

2.3　NIO インタフェースボード

NIO は、一つの親局（NIO-C）が電源筐体上の複数の子局（NIO-S）とシリアル通信し、NIO-S に対して出力の設定、入力状態の収集を行う。NIO-C は VME バス上にある VME 基板で、NIO-C からの通信信号を一旦 BRANCH 基板に入力する。BRANCH 基板は光電変換と信号の分岐を行う。NIO-S には光信号を入力可能であり、NIO-S をスター形式で接続できる。また、NIO-S は入力された光信号を電気信号に変えて外部に出力することが可能であり、複数の NIO-S をバス形式で接続することができる。

NIO の特長は以下のような点である。

(1) 遠隔にある I/O を光通信またで制御できると共に同じ筐体上のデバイス同士は安価な RS485 を使用して接続できる。

(2) NIO-S の制御部分のファームウエアは、ダウンロードができるため、用途に応じて処理内容が変更できる。

(3) NIO-S 単独での自立制御が行える。

（３）の点は現有施設で使用している同様のコンセプトを踏襲したいという強い意向があった為に重要な特徴となっている。

2.4　アプリケーションサーバーとデータベース

費用の削減の為に AS と DB は Linux ベースの機器を使用することとした。DB に関しては Oracle 社が間接的にサポートする Miracle Linux を使用し、AS に関してはサーバー用としての正式なサポートがある Red Hat Linux Advance Server 2.1 を採用した。HW としては Raid5 仕様の Dell 社の PC サーバーを使用している。

3　ソフトウエア機能概要

3.1　個別電源制御と一括電源制御

VME 上に作成される個別電源 CORBA サーバーオブジェクトの場合はいかなる CORBA クライアントから直接制御できる。従って個別の Java で作成された GUI をクライアントとする構成になっている。電源オブジェクトを生成する際には AS 経由で DB から設定情報を JDBC 経由で XML 形式に変換した後 AS から VME に CORBA 通信で伝えられる。一括制御の場合はリストの維持を AS で行っており、制御動作も AS を経由して行われる。RT-CORBA の仕様としては priority 制御の機能のみを使用した。

3.2　ロギングとアラーム機能

個別の電源の状態を定期的に取得するポーラーは VME 上で動作する。ポーラーで得たデータは AS に送られロガ

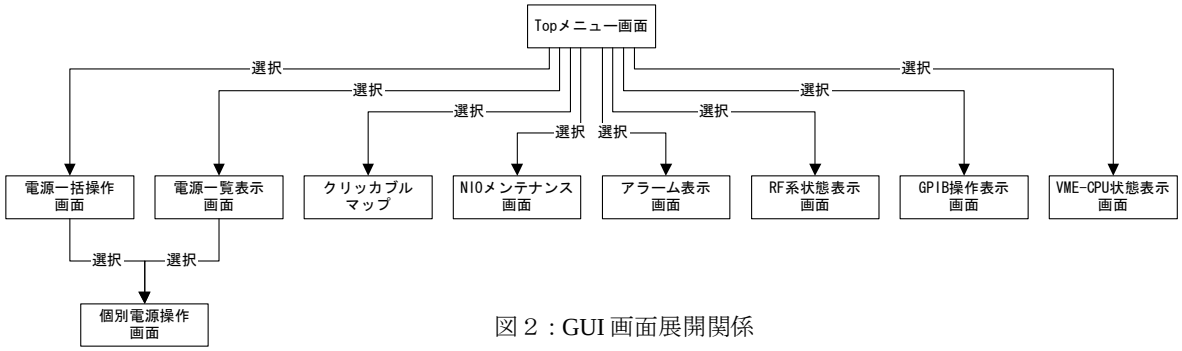

図２：GUI 画面展開関係

ープログラムで内容を共有メモリー上に書き出すと共に、DB のログ DB に格納される。格納の際にはプログラムは C++で記述されているので JDBC ではなく Oracle 社の提供している C ライブラリを用いている。その際に DB のベンダーに依存しないように共通のインターフェースを用いるクラスでラップしてある。

アラームハンドラーは AS で動作しており、ロガーで取得された共有メモリにある情報を元にアラーム条件を判別してアラーム画面に表示する。

4　ユーザーインターフェイス

4.1　電源設定及び監視画面

GUI には R&D で使用したのど同様に Java 言語で作成されている。個別電源画面、RF 状態モニター画面は Cosylab の製作した Abeans を使用することでソフトの再利用を可能にした。図２に画面展開のシークエンスを示す。図３は一括電源設定画面を示しており、データベースに格納されている一括電源設定のファイル一覧を表示し、そのファイル一覧から、選択されたものの詳細データを表示する。また、必要に応じて、画面操作により現在の各電磁石電源の設定値を取得し、データベースへ格納を行う。S 列のチェックボックスを選択することにより個別電源操作画面の表示を行う。

4.2　クリッカブルマップ

IRC および SRC の電磁石配置図より、表示されているセクター電磁石部をマウスクリックすることにより、セクター内の電磁石電源のステータスを一覧表示する Java アプレットを製作した。Web サーバーは AS 内で動作する。図４は SRC のアプレット表示画面を表す。

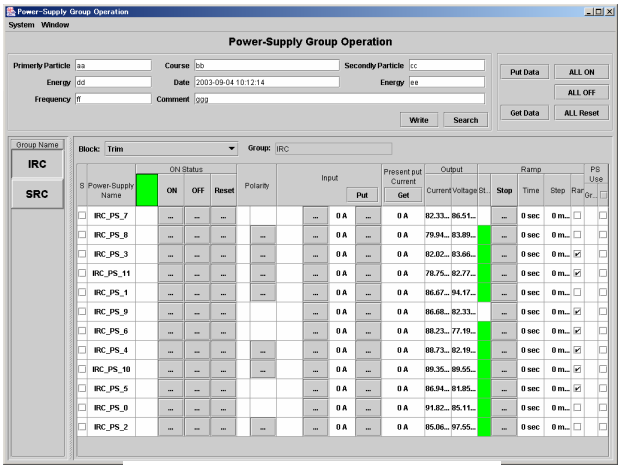

図３：一括電源設定画面

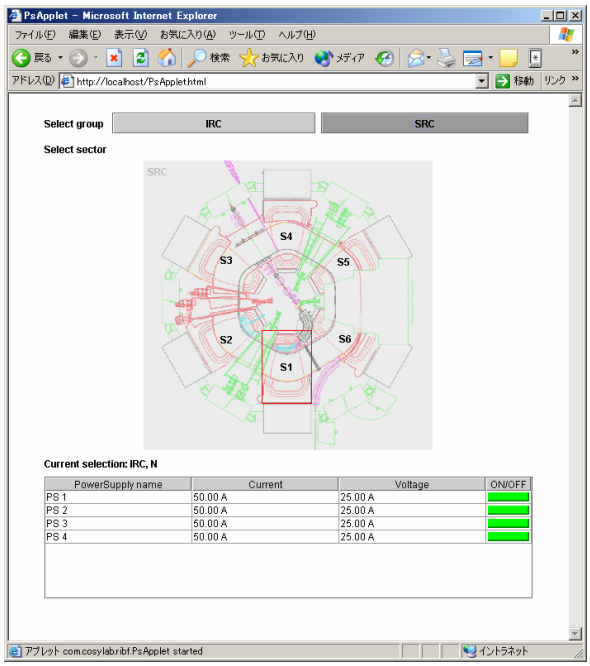

図４：クリッカブルマップ画面一例

参考文献

[1] http://ribfweb1.riken.go.jp/

[2] Tanabe, T., Masuoka, T., and Kase, M." Preliminary Design of RIBF Control System using CORBA and RTOS," The proceedings of EPAC2000, Stockholm, Sweden, (2000) 1871

[3] Tanabe, T., Masuoka, T., Ohnishi, J., Takano, M. and Katayama, T. “Preliminary Design of RIBF Control System Based on RT-CORBA and Java”The proceedings of 8th International Conference on Accelerator and Large Experimental Physics Control System (ICALEPCS '01), San Jose, USA, (2001) 433.

[4] Tanabe, T., Masuoka, T., and Kase, M. “XML-based integration for VMEs, an application server and RDBMS,” The proceedings of EPAC 2002, Paris, France, (2002) 2052

[5] http://www.aps.anl.gov/epics/

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

[7] http://java.sun.com/

[8] http://elc.aska.or.jp/index-e.html