制御 (1) — J-PARC加速器と分散制御システム —

1　はじめに

J-PARC Main Ring (MR) 加速器は, 3 GeV で入射された陽子ビームを 30 GeV まで加速し, T2K 長基線ニュートリノ振動実験とハドロン実験施設に供給している。その制御システムは, 周長約 1600 m にわたって分散配置された機器を総合的に制御している。制御点数の総数は約 10 万に及ぶ。

このテキストでは, J-PARC MR を題材にして, 加速器制御システムの考え方や設計, 初期設計からの (運転を開始してから 10 年間の経験に基づいた) 変遷を解説する。

2　加速器制御システム

加速器制御システムの役割は, 安定にビームを加速して実験に供給するための手段を提供し, 加速器の最高の性能を引き出すことである。制御システムは複数のサブシステムを統合し, 加速器全体を 1 つのシステムとして制御する。1 つのサブシステムは, 例えば主電磁石電源システムといった機器グループの制御を担当したり, ビーム位置モニタシステムから読み出したビーム軌道を表示することを担当したりする。

制御システムを支える計算機技術の発展がなければ, 加速器の大型化・高度化は実現できなかったであろう。1960 年代に加速器の制御システムに計算機が用いられて以来, 加速器における制御システムの重要度は年々増加している。1985 年からは the International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS) という国際会議が隔年で開催されている [1]。

2.1　加速器制御システムの機能

制御システムの基本的な機能は,

- 加速器のモデルに従って機器に値を設定してビームを操作する。
- ビームや加速器の機器が意図した状態であるかをモニタする。
- 意図した状態から外れている場合は補正する。

ことである。これらの 3 つの要素をそれぞれ筋肉, 目, 頭脳に例えれば, 加速器制御システムはこれらを繋ぐ中枢神経に相当する。

2.2　制御システムに求められるもの

安定した加速器の運転を実現するために, 制御システムには

- 信頼性が高く,
- 可用性があって,
- 保守性が良い

制御機構を提供することが求められる。

その一方で, 制御システムには柔軟性も求められる。加速器の技術的な詳細が決まるまで, 制御システムの仕様は定まらないため, 比較的短期間での実装が求められる。

- 運転が始まったあとに加速器の要求仕様が変わる
- 新しい手法を素早く実装して加速器の運転に適用する
- 新しく導入された機器を制御システムに組み込む

といったことへの素早い対応が必要である。

2.3　制御システムがカバーする範囲

一口に加速器制御と言っても, その分野は次に挙げるような広い範囲を網羅している:

- 機器の制御
  - 加速器を構成する機器の操作と監視
- 加速器のモデリング (シミュレーション)
- 単位の変換
  - 磁場 T, キック量 mrad といった加速器モデリングの単位から, 電流や電圧といった機器の単位の変換

  - ADC/DAC のカウント値と, 電流や圧力といった物理量との変換
- アルゴリズム
  - バンチフィードバック, リアルタイム軌道補正といったフィードバック
  - 操作の自動化
- 機器の構成や設定値のデータベース
- 加速器制御のためのインフラ
  - ネットワーク
  - サーバ計算機, ストレージシステム
- ツール類
  - バージョン管理
  - アーカイブシステム
  - アラームシステム
- 広義の加速器制御に含まれるもの
  - タイミングシステム
  - Personnel Protection System (PPS): 人的保護システム
  - Machine Protection System (MPS): 機器保護システム

その一方で, 加速器制御の範疇は各加速器施設の事情によって様々である。例えば J-PARC MR には, 電子陽電子衝突リングや放射光リングで行っているようなリアルタイム軌道補正システムは存在しない。バンチフィードバックシステムはビームモニタグループの管轄となっていて, 制御グループはフィードバックシステムの制御システムへの接続を提供しているだけである。

ここではタイミングシステムや PPS, MPS などは除外して, 加速器制御の基本的な部分について述べる。

3 標準モデル

現代的な加速器制御システムは, 図 1 に示されるように,

- Human-Machine Interface (HMI) 層
  人間が加速器を操作するための入力装置や, 加速器をモニタするための表示装置

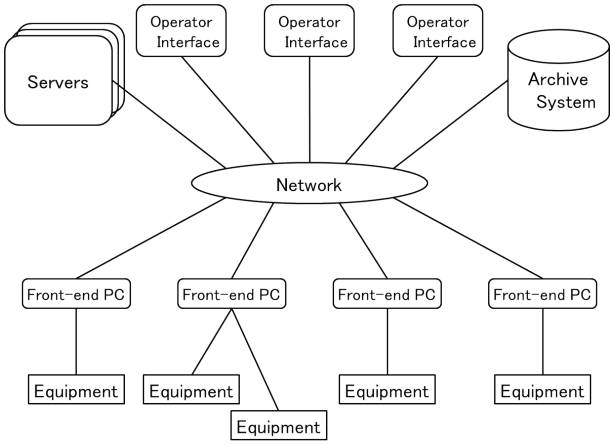


図 1: 加速器制御の標準モデル

- フロントエンド層
  フィールドバスで接続された制御対象機器を制御する計算機群
- ネットワーク層
  HMI 層とフロントエンド層を結ぶネットワーク

の 3 層から構成される。この考え方は加速制御の標準モデル [2] とも呼ばれ, 広く採用されている。このように加速器を分散制御することには,

- 配線をシンプルにできる
  大規模な加速器施設では, 多くの機器が広い敷地に分散して設置されるため, 全ての信号線を制御室まで配線するのは事実上不可能である。ネットワークを用いて信号を多重化することで, 配線を大幅に簡略化することが可能である。
- 耐障害性がある
  機器が故障した際の影響を限定できる。
- 拡張性がある
  機器を増設する場合はフロントエンド層に計算機を追加することで対応できる。
- 柔軟性がある
  加速器は 20 年–30 年, あるいはそれ以上の長い期間にわたって運転される。制御に用いるコンポーネントの寿命は長くても 10 年程度, 短いと 2, 3 年で, 寿命は年々短くなる傾向にある。各層の機器は比較容易に交換可能である。

といったメリットがある。

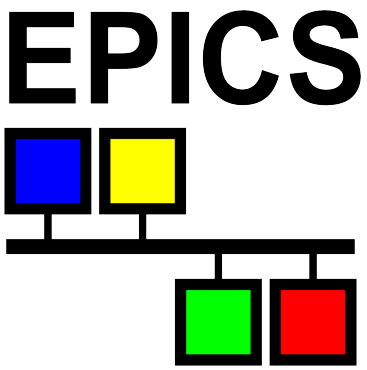


図 2: EPICS のロゴ。横線の上の四角は OPI を, 横線の下の四角は IOC を, これらを接続する黒い線はネットワークを現している。

その反面, ネットワークの信頼性や可用性が制御システムの信頼性や可用性に直接反映されてしまうというデメリットがある。ネットワークを冗長化する, 分割する, 光ファイバを使う, といったネットワーク構成の設計が重要である。

4 制御フレームワーク EPICS

1960 年代に加速器の制御にコンピュータを用いるようになった当初は, 加速器毎に独自のハードウェアとソフトウェアを開発して制御システムを構築していた。1980 年代になると, 複数の加速器制御システムが似たような問題を抱えていてることが明らかになってきた。複数の加速器施設が抱える共通の問題を解決すべく, 制御システムの構築に共通のフレームワークが用いられるようになってきた。こういったフレームワークにはオープンソースなプロジェクトもプロプライエタリなものものあるが,

- EPICS [3]
- MADOCA [4]
- TANGO [5]
- LabVIEW [6]
- Vsystem [7]

などが挙げられる。ここでは, J-PARC 加速器がフレームワークとして採用している EPICS を紹介する。

Experimental Physiscs and Industrial Control System (EPICS) は, 出版-購読型のネットワーク分散型制御システムを構築するためのフレームワークである。アメリカ合衆国のロスアラモス国立研究所とアルゴンヌ国立研究所で 1988 年頃から開発が始まり, 現在はオープンソースで国際共同開発されている。開発当初は VxWorks 上での利用が前提となっていたが,

- Linux や mac OS といった Unix 系 OS
- Microsoft Windows
- VxWorks, RTEMS といったリアルタイム OS

といった, さまざまな OS 上で利用可能となっている。Unix 系 OS の場合は Linux が用いられることが多く, Intel x86_64/x86 以外にも ARM や PowerPC など, 様々な CPU が利用可能である。

EPICS は加速器だけでなく, 望遠鏡や核融合実験炉といった大型実験施設の制御に用いられている。EPICS を用いている施設の挙げると,

- J-PARC, KEK Linac, SuperKEKB, 理研 RIBF
- ESS, DLS, FRIB, SNS
- 核融合実験炉: ITER, IFMIF-EVADA
- KECK 望遠鏡
- KAGRA, LIGO

などがある。

EPICS には, 制御システムに必要とされる

- 機器の遠隔操作
- アクセス権限
- I/O の抽象化
- 単位の変換
- フィードバックループ
- シーケンス処理 (処理の自動化)
- GUI ビルダ
- アーカイブシステム

といた部品が揃っている。国際共同開発のプロジェクトであり, さまざまな施設で利用されているので,

- 自分たちで個別に開発・保守する必要がない
- バグ出しがすすんでいる

- 他の施設の支援を受けられる可能性がある
- 国際的に貢献できる可能性がある

といった利点がある。

EPICS では, 加速器のオペレータや機器の担当者が操作する上位制御アプリケーションを OPI, フロントエンドの計算機を I/O Controller (IOC) と呼ぶ。図 2 に示す EPICS のロゴは, OPI と IOC がネットワークで接続されている様子を象徴的に示している。

OPI の開発には GUI ビルダを用いたり, Python (例えば PythonCA [8] という Python module を利用する) や SAD [9] といったスクリプト言語を用いる。下位のソフトウェアを開発する機会は少なくなってきているが, C と C++を用いることが多い。Java や Matlab, LabVIEW から EPICS を利用するためのモジュールも開発されている。

4.1 I/O Controller (IOC)

EPICS では, フロントエンドの計算機を I/O Controller (IOC) と呼ぶ。IOC の約割を一言で表すと, 制御対象機器のフィールドバスから EPICS の通信プロコルである Channel Access (CA) へのプロトコル変換機である。制御対象となる機器が用いるプロトコルは,

- PCI Express, VME, CAMAC, GPIB などのバス
- SPI, I^2C, GPIO といったマイコンの入出力
- RS-232C, RS-485 といったシリアル通信
- USB
- HTTP, telnet, VXI-11 [10] といった標準的な TCP/IP 通信
- 機器独自のプロトコルを用いた TCP/IP 通信

など, 機器の種類や用途によって様々である。IOC は制御対象機器の I/O の詳細を隠蔽し, CA のサーバとなって機器へのアクセスを提供する。CA のクライアントである上位制御系アプリケーションは, 制御対象機器のハードウェアを詳細に知っている必要はない。制御点の名前さえ知っていれば機器への I/O アクセスが可能である。

4.2 EPICS レコードと Database

EPICS では制御点を Process Variable (PV) あるいはレコードと呼ぶ。各 IOC は EPICS Databse と呼ばれるデータベースを持っていて, 制御下にある機器のハードウェアを PV 名に対応付ける。

PV にはデータ型がある。標準で利用可能な基本的なデータ型の例としては,

- longin, longout: 整数値。
- ai, ao: アナログ値 (浮動小数点)。制御対象機器の生データ (ADC のカウントなど) と物理量との相互変換が可能。
- bi, bo: On/Off や OK/NG など, 2 つの値をとるバイナリ値。
- stringin, stringout: 文字列型。

などがある。また, 以下に挙げるようなデータ処理のための型も用意されている:

- calc: ほかの PV の値を使った計算結果を値にもつ。四則演算や指数関数, 三角関数, ビット演算, 論理演算などを利用できる。
- dfanout: 複数の PV に値をコピーする。
- histogram: 他の PV の値を入力にしてヒストグラムを作る。

これら以外にも型があり, ユーザーが独自の型を作ることも可能である。

EPICS Database はデータ型のインスタンスとして PV を生成し, field と呼ばれる属性に I/O に用いる機器の種類や, スロット番号, チャンネル番号といったハードウェアの詳細を記述する。リスト 1 に, EPICS Database の例を挙げる。この例では, 電磁石電源を On/Off する操作と電源の On/Off 状態の読み返しを PLC で制御している。CPU として Yokogawa の linux 対応 CPU モジュール F3RP71/F3RP61 を利用しており, CPU モジュールが IOC となっている。

リスト 1: EPICS Database の例

```
# MR のアドレス 16 番地にある八極電磁石の電源の
# On/Off の制御
# コメントは # で始まる

# bo はバイナリ値の出力レコード
```

```
record(bo, "MRMAG:OCTPS_016:OPE:OUTPUT")
{
    # DTYP で I/O に用いる機器を指定する
    field(DTYP, "F3RP61")
    # 出力先 PLC モジュールの
    # ユニット番号, スロット番号, 接点番号
    field(OUT,  "@U0,S5,Y3")
    # 0 と 1 に対応する文字列を忘れずに
    field(ZNAM, "OFF")
    field(ONAM, "ON")
}

# bi はバイナリ値の入力レコード
record(bi, "MRMAG:OCTPS_016:RB:OUTPUT")
{
    field(DTYP, "F3RP61")
    # データを読み込む PLC モジュールの
    # ユニット番号, スロット番号, 接点番号
    field(INP,  "@U0,S4,X3")
    # 0.5 秒毎にポーリングする
    field(SCAN, ".5 second")
    field(ZNAM, "OFF")
    field(ONAM, "ON")
}
```

4.3 Channel Access

EPICS が用いる通信プロトコルは Channel Access (CA) と呼ばれ, TCP と UDP のポート 5064 と 5065 を用いる。CA は EPICS のランタイムライブラリとして実装されている。上位制御系アプリケーションと IOC を開発したり利用したりするだけなら, CA の詳細は EPICS 利用者の目には触れることはない。

通常は IOC が CA のサーバとなり, クライアントである上位制御系アプリケーションに制御対象機器への I/O アクセスを提供する。IOC が他の IOC が提供する PV にアクセスする場合は, CA のクライアントになる。

制御対象となる PV の名前は, ネットワーク上のフラットな名前空間上で一意でなければならない。クライアントはアクセスしたい PV がネットワーク上のどの IOC によって提供されているかを知っている必要がない。クライアントは UDP のブロードキャストを用いてネットワーク上の PV を検索する。PV を検索するブロードキャストに対して, 当該 PV を提供している IOC がクライアントに返答すると, クライアントは TCP で IOC への接続を確立する。IOC の再起動やネットワーク障害などで接続が切れた場合には, クライアントは自動的に IOC への再接続を試みる。

リスト 2 に EPICS のコマンドラインユーティリティを用いて制御対象機器の状態を読み出す例を, リスト 3 に制御対象機器を操作する例を示す。

リスト 2: コマンドラインからの利用例 (1)。電磁石電源の出力状態を読み出している。

```
% caget MRAMG:OCTPS_016:RB:OUTPUT
MRAMG:OCTPS_016:RB:OUTPUT      OFF
```

リスト 3: コマンドラインからの利用例 (2)。電磁石電源に出力司令を送っている。

```
% caput MRAMG:OCTPS_016:OPE:OUTPUT ON
Old : MRAMG:OCTPS_016:OPE:OUTPUT      OFF
New : MRAMG:OCTPS_016:OPE:OUTPUT      ON
```

4.4 EPICS における GUI アプリケーション

EPICS では, 加速器のオペレータや機器の担当者が操作する上位制御アプリケーションを OPI と呼ぶ。OPI の開発には, GUI ビルダを最大限に活用している。加速器のオペレータや機器の担当者は必ずしもプログラミングに精通しているとは限らない。イベント駆動型のプログラミングの知識がなくても GUI ビルダを使えば, あらかじめ用意された GUI のウィジェット (ラベル, テキストボックス, メーター, チェックボックスといった部品) を並べ, 各部品を PV に割り当てるだけで制御画面を作成することができる。このような GUI ビルダには

- EDM [11]
- MEDM [12]
- Control System Studio (CSS) [13]

などがある。図 3 に, CSS に用意されているウィジェットの例を示す。

GUI アプリケーションを開発する場合, J-PARC MR では X Window System 上で動作する EDM や MEDM が多く使われてきた。PV から取得した現在の値からトレンドグラフを表示するには Strip Tool [14] が使われ, アーカイブから取得した過去のデータのトレンドを表示するには

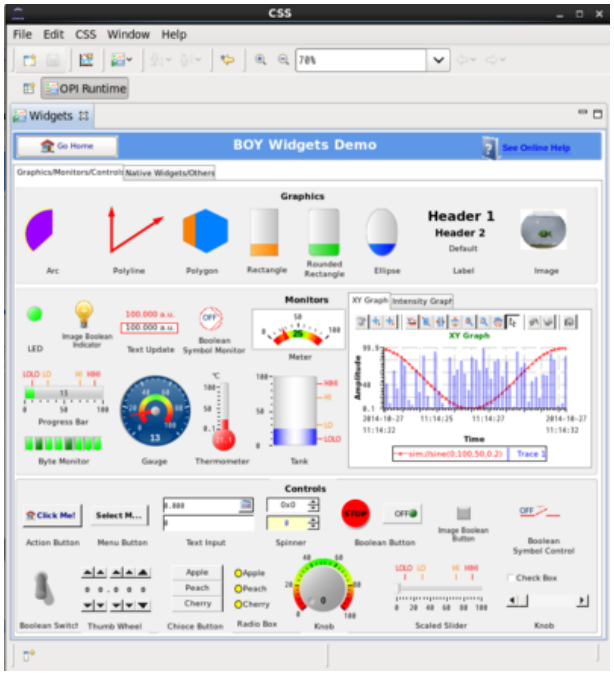


図 3: CSS に用意されているウィジェットの例。

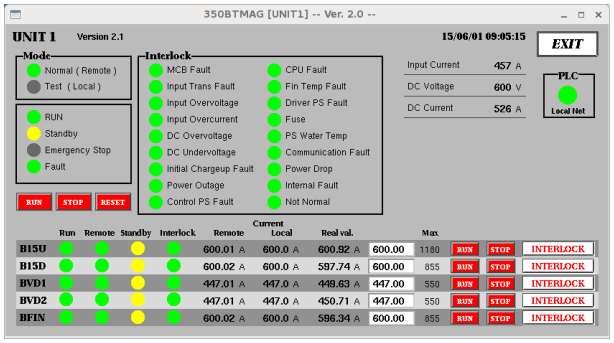


図 4: EDM で作成された, MR 入射路の電磁石電源の制御パネル。

EPICS ArchiveViewer [15] や, 内製の Web ベースのアプリケーション [16, 17] が使われてきた。

表 1 に示すようにこれらのツールは機能が限定されており, 相補的な関係にある。また, これらは互いに独立に開発・保守されているプログラムであり, GUI のルックアンドフィールもそれぞれ異なっている。図 4, 5, 6 に EDM, MEDM, Strip Tool で作成された MR の制御画面の例を示す。

MR で開発されたきた EDM や MEDM を用いた制御アプリケーションの多くは, 各機器のグループが専門的な調整を行うために作成したものである。その一方で, 加速器の運転状況を把握したり, 加速器に問題が発生した際に原因を特定するため

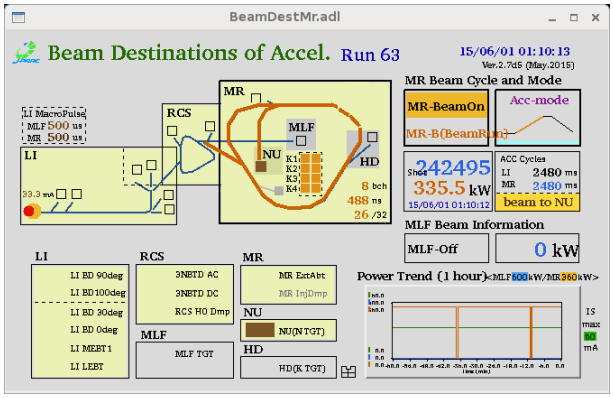


図 5: MEDM で作成された, J-PARC 加速器の統合運転状態表示画面。

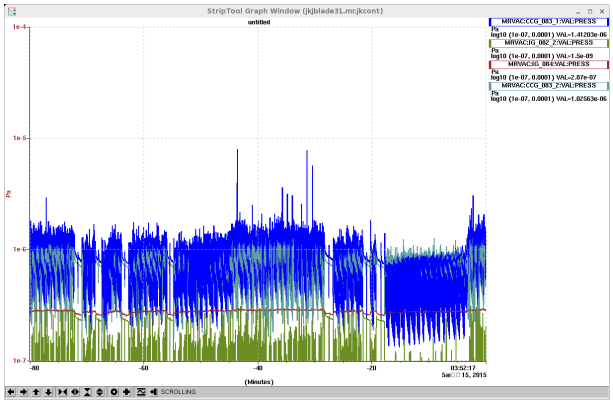


図 6: Strip Tool の画面の例。MR のビームパイプ内の 80 分間の圧力のトレンドグラフ。

にも用いられているが, 誤操作により加速器や機器を意図せずに停止してしまう可能性があり, 慎重な操作が操作が求められる。参考文献 [18] にあるように, 加速器の運転状況を把握し, あるいは問題が発生して加速器が停止した場合に原因を特定できる, 統一的なルックアンドフィールを持った加速器統合アプリケーションが求められている。CSS はそのようなアプリケーションの開発環境の有力な候補である。

CSS は大規模な制御システムの GUI を構築するためのフレームワークで, 近年 EPICS コミュニティでは活発な活動が見られる。その開発は DESY で始まり, 現在は複数の研究機関と大学で共同開発されている。CSS はマルチプラットフォームな統合開発環境である Eclipse をベースにしており, Linux, mac OS, Windows 上で実行することができる。

表 1: GUI アプリケーションの比較

	PV の値の モニターと操作	トレンドグラフの表示	アーカイバから取得したデータの トレンドグラフへの表示
CSS	yes	yes	yes
EDM	yes	no	no
MEDM	yes	yes	no
Strip Tool	no	yes	no
ArchiveViewer	no	no	yes
内製の Web Interface	no	no	yes

CSS には加速器の制御システムに必要とされる,

- GUI ビルダ: BOY
  - Python か JavaScript を用いた動的な画面を作成できる
- トレンドグラフ, アーカイブからのデータ取得: Data Browser
  - アーカイバから取得した過去のデータと PV から読み込んだ現在のデータを連結して表示できる
- アラームシステム: BEAST
- アーカイブシステム
- 電子ログ

といった機能が統合されている。これらの機能を連携して運用できるのが CSS の大きな利点である。その一方, CPU パワーと大容量のメモリを必要とするのが欠点で, 加速器制御グループでは CSS を実行する PC に 64bit CPU と 8 GB 以上のメモリを推奨している。図 7 に CSS で作成された J-PARC MR の制御画面の例を示す。

CSS は EDM や MEDM からの移行も考慮されている。CSS には EDM や MEDM で作成された既存の GUI 画面からの変換を支援するツールが組み込まれている。

4.5 アーカイブシステム

EPICS のアーカイブシステムは, PV の値の変化を時系列データとしてファイルに記録するクライアントソフトウェアである。利用可能なアーカイブシステムには

- Archiver Appliance [19]
- CSS Archiver (RDB Channel Archiver) [20]
- Cassandra PV Archiver [21]
- Channel Archiver [22]

などがある。いずれのアーカイブシステムも, データ収エンジンは設定に応じて (a) PV を監視し値が変化したら, あるいは (b) 設定された時間間隔毎に, PV の値とタイムスタンプを記録する。

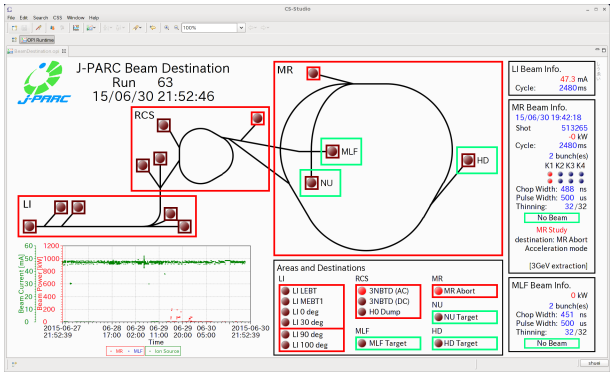
Channel Archiver は多くの施設で長い間使われてきたが,

- 2007 年に開発が終了しており, 最近の C++ コンパイラではビルドできない。
- 32bit に依存した独自のデータフォーマットを採用している。
  - ファイルサイズに 2GB の制限があり, 長い期間のデータを保存できない。
- データの読み出しに XML-RPC を利用する。
  - いちどに大量のデータを読み出せない。
  - XML-RPC で表現できないデータがある。

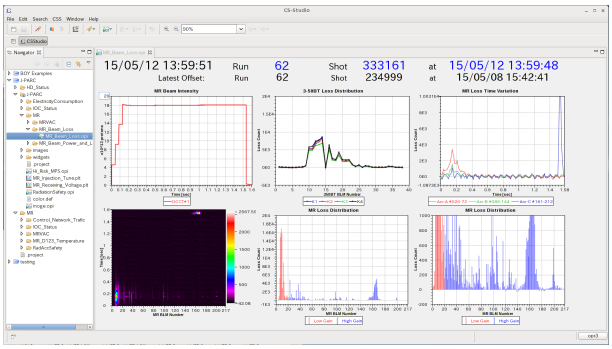
といった制限がある。

CSS Archiver は Channel Archiver の後継を狙ったアーカイブシステムで,

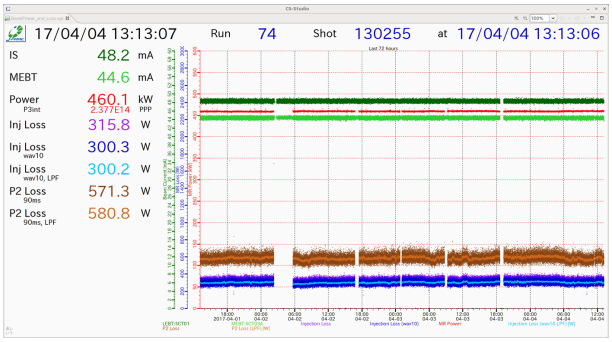
- CSS のコンポーネントの 1 つである。
- バックエンドに RDB (Oracle, MySQL, PostgreSQL から選択可能) を利用している。
  - データ書き込みの信頼性が高い。
  - Channel Archiver に比べるとデータサイズが大きい (J-PARC MR の場合, 約 5 倍)。
  - 高速なデータ読み出しを実現するには, 強力な RDB サーバが必要である。



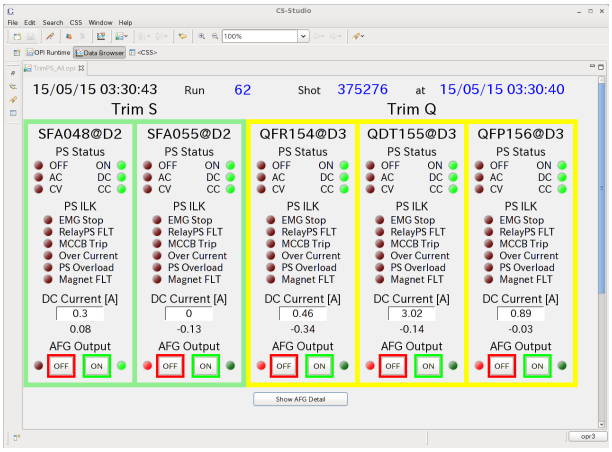
(a) J-PARC 加速器の統合運転状態表示画面。



(b) MR でのビームロス分布。



(c) MR のビームパワーとビームロスの値, 及び 3 日間のトレンド。



(d) トリムコイル電源の制御画面。

図 7: CSS で作成された J-PARC MR の制御画面の例。

といった特徴がある。2016 年頃には開発が放棄されたかに見えた。しかし, 2017 年に時系列データに特化したデータベースである InfluxDB をバックエンドに用いた試みがなされるなど, 完全に放棄されたわけではなさそうである。

Archiver Appliance [19] は 2015 年に登場した, 新しいアーカイブシステムで,

- データの読み出しが速い。
- 管理の負担軽減が図られている。
- クラスタリングによる負荷分散が可能である。
- 短期/中期/長期の 3 層の階層型ストレージをサポートしている。
- データサイズは Channel Archiver と同程度。
- Channel Archiver のデータを Archiver Appliance のデータファイルに変換するユーティリティが存在する [23, 24]。

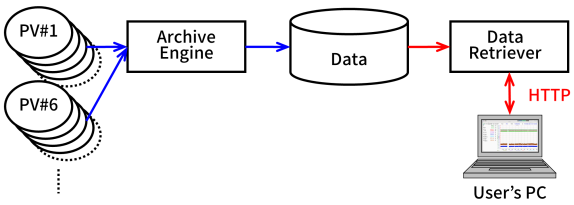


図 8: Archiver Appliance の概要。

といった特徴がある。図 8 に Archiver Appliance の概要を示す。データファイルは Google Protocol Buffers を用いたバイナリファイルである。Archiver Appliance では PV 毎に異なるディレクトリを作成し, データファイルをあらかじめ設定された時間間隔毎に分割することで, データ読み出しの高速化を図っている。Archiver Appliance がサポートする階層型ストレージは, 短期ストレージから中期ストレージへ, 中期ストレージから長

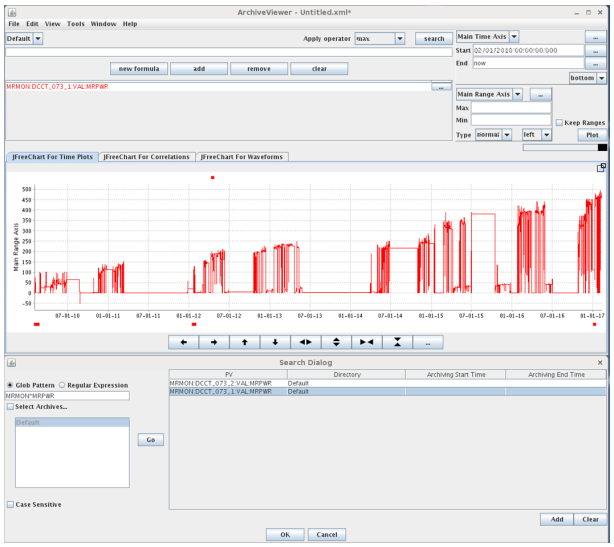


図 9: Archive Viewer の画面の例。2010 年から 2017 年までの MR のビームパワーの履歴を表示している。

期ストレージへと自動的にデータをコピーする。

- 短期ストレージには SSD, 中期ストレージにはローカル HDD, 長期ストレージには NAS を利用する。
- 次の階層のストレージにコピーする際に, データを間引く, 平均値と分散に置き換える, といったデータ処理を行う。

といったことが可能である。

Archiver Appliance からデータを読み出すには, Data Retriever に HTTP でリクエストする。クライアントは取得したいデータの PV 名, 開始時刻, 終了時刻をクエリ文字列として Data Retriever の URL の末尾に追加する。URL を切り替えることで

- JSON
- データファイルと同じバイナリ
- テキスト
- CSV
- MATLAB 書式設定済ファイル

といった様々な形式で時系列データを読み出すことが可能である。また, クエリ文字列に追加することで, 指定した時間間隔の

- 平均値

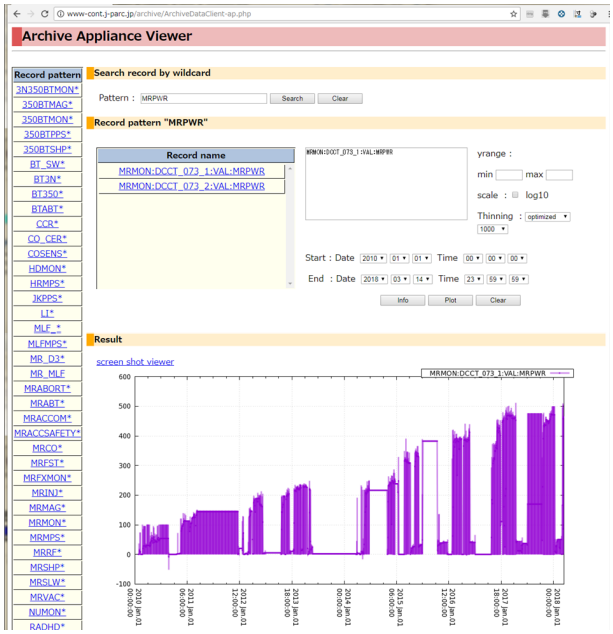


図 10: Archiver Appliance で記録したデータを閲覧する内製の web アプリケーション。

- 標準偏差
- 最大値, 最小値
- 最頻値
- メディアン
- 最初の値, 最後の値

といった値を取得することも可能である。

Archiver Appliance で記録したデータを読み出すクライアントソフトウェアには,

- CSS (図 7)
- Archiver Appliance 対応版 Java Archive Viewer [25] (図 9)
- 内製の web アプリケーション [26] (図 10)

といったものが利用可能である。

J-PARC MR では, 加速器の運転が始まる半年前の 2007 年から 2017 年まで Channel Archiver を用いてデータをアーカイブしてきた。アーカイブしている制御点数は約 24000 で, データレートは約 1.3 TBytes/year であった。前述したような Channel Archiver の制限が明らかになってきたため, 2015 年からは Channel Arvhiver と平行して CSS Archiver でのアーカイブを開始し, CSS Archiver への移行の可能性を探った。CSS Archiver のデータサイズが大きいことと読み出

しが遅いことが明らかになったため, 2017 年からは Archiver Appliance を用いてデータをアーカイブしている [27]。2017 年 7 月の加速器の長期シャットダウンと同時に Channel Archiver と CSS Archiver による記録は停止した。2018 年 1 月までに 2007 年から 2017 年までの Channel Archiver の全データを Archiver Appliance のデータに変換し, アーカイブシステムを Arhciver Appliance に一本化した。

5 J-PARC 加速器の場合

加速器制御を設計し構築する際には,

- 国際標準 / De Facto Standard
- 新しい技術 / 枯れた技術
- 柔軟性 / 堅牢性
- オブジェクト指向 / チャンネル指向
- トップダウン設計/ボトムアップ設計
- ボリュームディスカウント/ロット不良のリスク

など, 相反する考え方のなかでバランスを取ることが重要である。例えば, 最先端の技術を採用したとしても, 一時的に流行しただけで数年後には廃れてしまうかもしれない。その一方で, 枯れたて安定した技術を採用した場合は, 加速器の性能を引き出せないかもしれない。

ここでは J-PARC(主に MR) を題材に, 加速器制御システムを構築した際の考え方と, その後の変遷について紹介する。

5.1 ネットワーク

加速器の制御に用いる高速なネットワークは, 世の中の動向に併せて

- 光ファイバ, UTP
- Ethernet
- TCP/IP

の組み合わせ以外はほとんど利用されなくなってきている。

図 11 に J-PARC 加速器制御ネットワークの論理的な構成を, 図 12 に施設間の光ファイバ配線

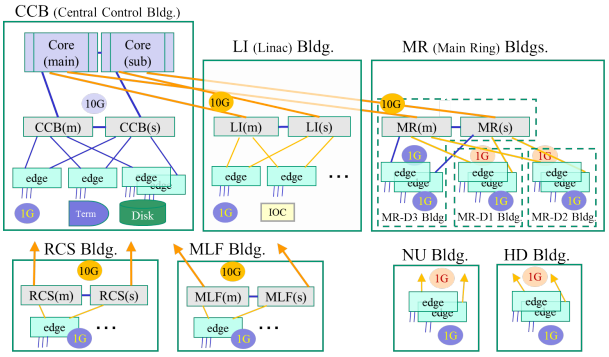


図 11: J-PARC 加速器制御ネットワークの論理的な構成。

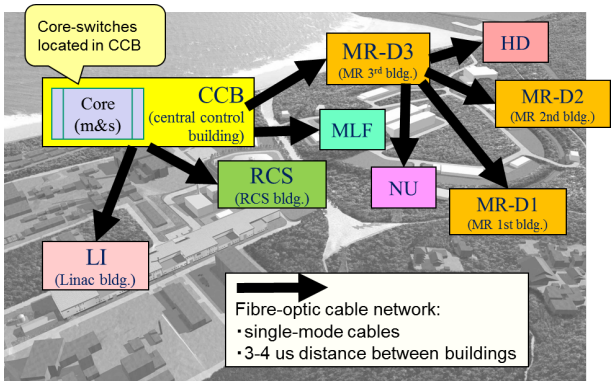


図 12: J-PARC 加速制御ネットワークの光ファイバによる施設間の配線。

の物理的な構成を示す [28]。 ネットワークの中核となるコアスイッチは中央制御棟に配置されている。LI, RCS, MLF の各施設と MR 第 3 電源棟 (MR-D3) に設置されたエッジスイッチまでは 10Gbps の光ファイバで配線されている。コアスイッチ, エッジスイッチ, 施設間の光ファイバはいずれも二重化されており, 障害発生時には主系から従系に切り変わるようになっている。図 13 にコアスイッチとエッジスイッチの外観を示す。表 2 に, J-PARC 加速器制御ネットワークで使用しているエッジスイッチの数をまとめる。

2011 年以降, J-PARC 加速器の運転停止を伴うようなネットワーク障害が年間 10 回程度発生していた。2016 年からは減少傾向にある。

2011 年から 2013 年にかけては, エッジスイッチが原因不明のまま停止する障害が 29 回発生した。2013 年のメーカーによる調査報告書で, 2007 年から 2008 年にかけて温度管理が不適切な状態で

(a) コアスイッチ (Extreme 社の Black Diamond 8900) の外観。

(b) エッジスイッチ (Extreme 社の Summit X460-48t) の外観。

図 13: コアスイッチとエッジスイッチ。

製造されたコンデンサが使用されてていることが判明した。該当するコンデンサを用いているエッジスイッチは 2014 年に交換し, それ以来この障害は発生していない。

2014 年から 2016 年にかけては, 冗長構成の 2 台のエッジスイッチの双方がハングアップないし再起動してしまう障害が 23 回発生した。ファームウェアのバグで, 主系・従系の冗長構成をなしているはずの双方が主系になってしまうのが原因であった。2016 年から順次ファームウェアを更新し, その後は安定して運転している。

スイッチの故障で, 冗長化構成が効果を発揮してネットワーク障害には至らなかったケースは

- GBIC モジュールの故障
- 電源モジュールの故障

などで, 年間数回程度の頻度で発生している。

J-PARC の加速器制御ネットワークは全体では Class-A のプライベートアドレスとし, VLAN を用いて各施設ごとにネットワークを分割している。ネットワークを分割することで, 例えば MR でネットワーク障害が起きても, 上流の加速器である LI と RCS には影響を及ぼさないようにして

表 2: J-PARC 加速器制御ネットワークで使用されているエッジスイッチの数

施設名	エッジスイッチの数
中央制御棟	13
Linac	83
RCS	39
MR	9
MLF	56
HD	2
NU	2
L3BT, 3NBT	48

いる。また, MR では建屋内の配線も可能な限り光ファイバを利用しており, 制御対象機器が発するノイズへの耐性を高めている。

MR では, 原則として制御ネットワークと制御対象機器のネットワークを分離している。IOC の加速器制御ネットワークは第 1 イーサネットポートに接続し, 制御対象機器は第 2 イーサネットポート側に接続している。第 2 ポートに接続する制御対象機器は基本的に 1 台のみで, 多くても 2, 3 台程度に留めているている。ネットワーク対応を謳う機器であっても, 大量のブロードキャストが流れる大規模なネットワークに接続すると動作が不安定になるなど, 過去に障害を生じた機器があった。

各制御対象機器にも Class-A の空間で一意な IP アドレスを割り当てている。それぞれ異なる IOC に制御されている象機器同士が直接通信することはありえないが, IP アドレスの管理の簡便を図っている。

5.2 OS の選択

加速器制御システムでは,

- ネットワークや分散処理との親和性
- プログラミング環境が充実していること
- オープンな環境であること
- 安定性, 信頼性があること

などの理由から, Unix 系の OS が採用されることが多い。

J-PARC MR の加速器制御システムでは, 上位制御系の計算機とフロントエンドの計算機の双方に Scientific Linux (SL) [29] を採用している。SL

表 3: SL のリリース日とサポート終了 (予定) 日

	リリース日	ベースとなった RHEL のリリース日	サポート終了日
SL4	2005-04-20	2005-02-14	2012-02-29
SL5	2007-05-14	2007-03-14	2017-03-31
SL6	2011-03-03	2010-11-10	2020-11-30
SL7	2014-10-13	2014-06-10	2024-06-30

はアメリカのフェルミ国立研究所が開発している Linux ディストリビューションである。RedHat Enterprise Linux (RHEL) をベースとしており, サポート期間が 10 年程度と長いのが特徴である。表 3 に SL のリリース日とサポート終了日の一覧を示す。

MR の制御システムは 2007 年に構築されたが [30], 当時リリースされたばかりの SL5 を避けて SL4 を採用した。2012 年には,

- SL4 のサポートが終了した
- SL4 では動作しないハードウェアが登場し始めた

ため, 5 年程度先を見据えて SL6 に移行した。

SL4 を運用していた期間は, そのライフサイクルの半ばから終盤だったため, ディストリビューションに含まれているライブラリやソフトウェアが古く, 当時最新のコンパイラやスクリプト言語, ライブラリなどが使えない, という状況であった。これらを必要に応じてビルドし, 共有ファイルサーバにインストールしたため, 互換性のない複数のバージョンの同名のライブラリが多数インストールされるようになってしまった。プログラムを開発する際には正しいバージョンのライブラリをリンクするように細心の注意が必要であった。

SL6 の環境では SL4 での反省を踏まえて, 可能な限り OS に含まれるパッケージや Extra Packages for Enterprise Linux (EPEL) や呼ばれる拡張パッケージを用いてインストールし, ライブラリの依存関係や互換性の問題を解決するようにしている。また, Software Collection とよばれるパッケージを用いることで, SL 標準のものよりも新しいバージョンのコンパイラやスクリプト言語が利用可能である。

SL6 に移行してすでに 6 年が経過した。そろそろ次期 OS を検討する時期がきていると言えよう。

5.3 IOC のハードウェア

J-PARC MR 加速器で用いられている制御対象機器のほとんどは,

- オシロスコープ, スペクトラムアナライザ
- ファンクションジェネレータ, ベクトル信号発生器
- PLC

といった汎用の市販品である。主電磁石やキッカーなどの大型電源は, マイコンや FPGA を用いた制御を行っているが, 制御システムや安全システムとの取り合いには PLC を利用している。また, 一部の特殊用途向けの機器, 例えば高速・長波形の ADC やタイミングモジュール, デジタルディレイなどは専用の VME ボードを製作している。

J-PARC 全体の加速器制御システムを構築した当初は, IOC として信頼性が高く堅牢な VME シングルボード計算機 (Single Board Computer; SBC) のみを使用することが想定されていた。しかし MR では, 運転が始まって加速器が高度化するにつれ, 制御対象となる機器の数と種類も増加し, IOC の種類も多様になっていた。2018 年 7 月現在, MR で用いられている IOC の仕様を表 4 に, MR で用いられてきた IOC の種類と数の履歴を表 5 に示す。

5.3.1 VME SBC

VME-SBC は J-PARC MR がビーム運転を開始する前年の 2007 年に導入が始まった [31]。図 14

表 4: 2018 年 7 月現在, J-PARC MR の運転に利用されている IOC の種類と数

名称	形状	数量	CPU	メモリ	起動メディア
PiNON サバ太郎 Type-P	小型ファンレス	29	Celeron J1900 2.0–2.42 GHz	8 GB	SSD
Virtual IOC	仮想マシン	33	–	0.5–1 GB	HDD
Abaco V7865	VME SBC	1	Core Duo T2500 2 GHz	3 GB	PXE
Abaco V7807RC	VME SBC	13	Pentium M 1.8 GHz	1–1.5 GB	PXE
Sanrits SVA041	VME SBC	24	Celeron M 600 MHz	512 MB	PXE
Abaco VME-7700RC	VME SBC	25	Celeron M 400 MHz	512 MB	PXE
Yokogawa F3RP61-2L	PLC-CPU	63	MPC8347 533 MHz	128 MB	CF
Yokogawa F3RP71-2L	PLC-CPU	(1)	ARM A9 866 MHz ×2	1 GB	SDHC

表 5: MR で利用している IOC の変遷

	VME SBC	PLC-CPU	仮想マシン	サバ太郎
2008 年	∼80	–	–	–
2011 年	∼90	∼30	–	–
2013 年	∼90	∼40	∼30	–
2016 年 6 月	79	45	25	11
2018 年 7 月	63	63	33	29

に VME SBC の例を示す。2018 年 7 月現在, 63 台の VME SBC が IOC として利用されている。形状は VME ボードであるが, いずれも Intel CPU を用いた所謂 PC/AT 互換機である。IOC に要求される CPU パワーに応じて 4 種類を運転に用いている。実際に VME バスを用いている IOC は 25 台ある VME-7700RC のみで, VME バスを通じてタイミングシステムの受信装置 [32] と, タイミングを遅延させるためのディジタルディレイを制御している。残りの VME SBC はフィールドバスとして Ethernet を用いて制御対象の機器と通信しており, VME クレートを電源として利用しているだけである。

MR では省スペースを図るために, 特殊仕様の 9U の VME クレートを採用している。6 スロットの VME バックプレーンが 3 つ並んでいて, 各バックプレーンの電源を独立して投入・遮断することが可能である。当初は IOC の電源を独立させるため, 各クレートに挿す VME SBC は 1 台に制限していた。従って 9U のスペースには最大で 3 台の IOC しか設置することができない。MR の高度化に伴って制御の対象となる機器が増加してくると,

- VME クレートに IOC を増設するスペースがない
- 制御室の 19 インチラックに VME クレートを増設するスペースがない

という問題が生じるようになった。暫定的な解決方法として,

- 1 台の IOC に複数の機能を持たせる。
- 暫定的に 1 つの VME バックプレーンに 2 台の VME SBC を挿す。

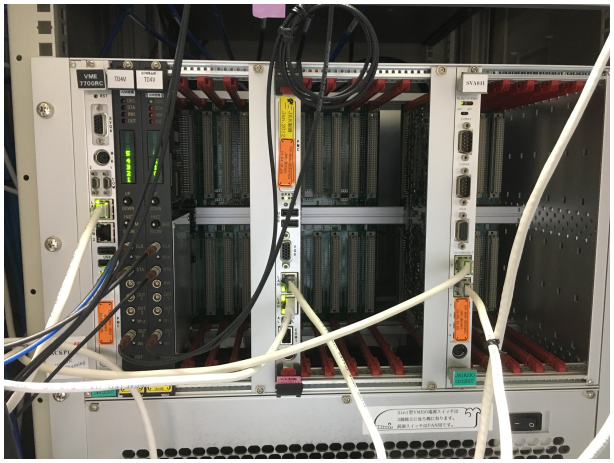


図 14: MR で使用している VME-SBC と, 6 スロット 3 連 VME クレート。左端の VME SBC は VME バスにアクセスしているが, 中央と右端の 2 台はフィールドバスとして Ethernet を利用しており, VME バスにアクセスしてない。

という手段が用いられたが, IOC の保守性, 柔軟性が損なわれるようになった。現在は, CPU パワーを必要とする IOC を手始めに後述する仮想 IOC の導入とサバ太郎への移行が進んでおり, VME クレートの問題は解決しつつある。

2011 年以降, J-PARC の夏季計画停電からの復電時に, 2007 年に導入した VME SBC のメモリが故障して起動できなくなる現象がみられるようになり, 2014 年にメーカーに調査を依頼した。当初は VME クレートや VME SBC 本体の異常も疑われたが, 2015 年に特定のメーカーの特定のロットの DRAM チップに問題があることが判明した。長時間使用した DRAM のセンス回路のトランジスタが劣化し, 電源を投入すると劣化したトランジスタが壊れる, という症状であった。メモリの故障は現在も続いており, 少ない年で 2–3 枚, 多い年で 10 枚程度のメモリモジュールが故障している。故障したメモリモジュールはその都度予備品と交換しており, 2011 年から 2017 年までに 32 枚のモジュールを交換した。2007 年に導入した VME SBC は大量に導入することで低価格になることを図ったが, この経験に基いて, 以後は多数の制御機器を購入する場合は時期を分散するようにし, 特定のロットに集中しないようにしている。

2014 年からは VME SBC 本体の故障もみられるようになった。毎年 3, 4 枚の頻度で故障している。故障したものは予備品と交換してメーカーに修理を依頼し, 修理から戻ってきたものは予備品としてストックしている。

図 15: F3RP71 を用いたシステムの外観。左から順に電源モジュール, F3RP71, 3 台の I/O モジュールが実装されている。

5.3.2 PLC-CPU

2009 年には, PLC の CPU モジュールでありながら OS として Linux を採用している Yokogawa の F3RP61 [33] が導入された。図 15 に後継機種である F3RP71 と I/O モジュールの外観を示す。

F3RP61 上は IOC として動作し, EPICS で PLC の I/O モジュールに直接アクセスすることが可能 [34] であるため, 重宝している。MR では約 60 台の F3RP61 を導入しており, 順次 F3RP71 に移行することが計画されている。また, 導入以来 1 台の F3RP61 も故障していないことは特筆に値する。

5.3.3 仮想 IOC

2010 年には Blade 計算機上の仮想マシンを用いた IOC [35, 36] が導入された。必要に応じて IOC を増設したり, またその機能を分離したりすることが簡便になった。VME SBC のスペース不足と悪化した保守性の問題は解消した。

サンプル数の多い波形データの収集や主電磁石電源の電流パターン生成など, 大容量のメモリや CPU の処理能力を必要とする IOC が増加してお

図 16: 19 インチラックの棚板に設置されているサバ太郎。

り, 既存の VME SBC の能力では足りない場合にも仮想マシンを用いるようになった。仮想マシンを増設するあまりホストマシンの計算機資源を使い切ってしまうこともあったが, Blade 計算機を増強した [37] ことで解決した。MR では現在, 33 台の仮想 IOC を運転に用いている。

5.4 PiNON サバ太郎 Type-P

VME-SBC と仮想 IOC の問題を解決すべく, 次世代の IOC として 2015 年に PiNON 製のサバ太郎 ® Type-P を導入した [38]。サバ太郎はファンレスの小型サーバで, Intel CPU を用いた所謂 PC/AT 互換機である。その外観を図 16 に示す。VME SBC や仮想マシンで運用していた IOC のソフトウェアは移植することなくそのまま実行可能であり, 既存の IOC からサバ太郎への移行は容易である。既存の VME SBC に比べると, CPU の処理能力も搭載可能なメモリ量も倍以上となり, これまで以上に大量のデータ処理が可能になった。

サバ太郎の導入により, VME クレートや NIM ビンの空きスペースなどに IOC を暫定的に置くことが可能になり, IOC を設置するスペース不足と保守性の問題は解決した。19 インチラックに棚板を設置すれば 2U 程度のスペースに 5 台のサバ太郎を並べることが可能である。9U のスペースに原則として 3 台しか設置できない VME SBC に比べると IOC の高密度化を達成できるようになった。

2018 年 8 月現在は 29 台のサバ太郎を使用している。これまでのところ, 最初期に導入した 1 台で故障のために SSD を交換したこと以外には大きなトラブルは経験していない。今後は順次 VME SBC をサバ太郎で置き換えていき, VME バスを必要とするタイミング系のモジュールは 9U の 6 スロット 3 連 VME クレートから 4~5 スロットの横置き VME クレートに入れ替えることで省スペース化を図る計画である。

5.5 上位制御系のハードウェア

MR の上位制御系の計算機は,

- サーバ計算機
- 表示端末用計算機

から構成されている。制御システムが設計された当初は, サーバ計算機には IBM/Lenovo の BladeCenter HS20 および HS21 が, 端末用計算機には の HP Compaq の Thin Client である t5720 および t5730 が採用されていた。GUI アプリケーションとして X Window System 上で動作する EDM や MEDM が多く使われており,

- 端末用計算機は GUI の描画とマウス・キーボードからの入力に特化した X 端末として使用する。
- アプリケーションはサーバ計算機上で実行する。

という運用であった。負荷が高いサーバ計算機があれば, アプリケーションを別のサーバに移動することで負荷の平滑化を図ってきた。

5.5.1 Blade 計算機

MR のサーバ計算機は IBM/Lenovo の BladeCenter を採用している。図 17 に Blade 計算機の外観を示す。

2007 年には 5 枚の HS20 で運用を開始し, MR のビーム運転を開始した 2008 年には, サーバ用・制御アプリケーション実行用・RCS 用の Blade 計

表 6: Blade サーバの仕様

	CPU	コア数	メモリ
HS20	Xeon 2.8 GHz	2	1 GB
HS21	Xeon 5110/5160 1.6 / 2.0 GHz	4	2 GB
HS21a	Xeon E5405 2.0 GHz	8	16 GB
HS22	Xeon E5505 2.0 GHz	4	20 GB
HS22a	Xeon X5690 3.46–3.74 GHz	12 (24 threads)	24 GB
HS23	Xeon E5-2630v2 2.6–3.1 GHz	6 (12 threads)	8 GB
HS23e	Xeon E5-2470 2.3–3.1 GHz	8 (16 threads)	48 GB

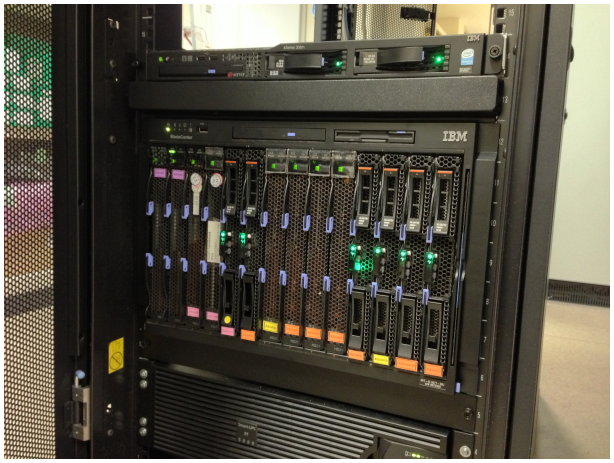


図 17: Blade サーバの外観。

算機を合計 15 枚用いていた。2011 年には Blade 計算機上に仮想マシンを導入し, 仮想マシン上で動作する IOC を MR の運転に投入した [35, 36]。2012 年には Blade 計算機を 24 枚に増強した [37]。これまで 8 枚の Blade 計算機で実行していた主要なサーバ (RDB, ldap, dhcp, Channel Archiver など) を仮想マシンに移行し, Blade 計算機としては 3 台に集約することで計算機資源の有効利用を図った。2014 年からは初期に導入して保守が切れた HS20, HS21, HS21a から順次 HS23e への更新を開始し, 現在に至っている。表 6 にこれまでに導入してきた Blade 計算機の仕様を, 表 7 にこれまでの典型的な Blade 計算機の構成 (予備や特殊用途を除く) を示す。

Blade 計算機は需要と予算に応じて枚数を増やせるのが大きな利点である。残念なことに 2016 年に BladeCenter は販売終了となってしまい, これ以上増強することができなくなってしまった。仮想 IOC の需要は今後も増加することが予想される。その一方で, 後述するように 2014 年に制御端末が Thin Client から NUC に世代交代し, 制御アプリケーションを実行する環境としての需要が減少している。今後はアプリケーション実行用の Blade を計算機を仮想マシンのホストへと転用しつつ, 次期サーバ計算機を検討する必要がある。

5.6 端末用計算機

端末用計算機にはの HP Compaq の Thin Client である t5720 および t5730 が採用されていた。MR の制御システムでは Thin Client を X 端末として利用し, 管理の手間とコストを抑えるべく SL4 をネットワークブートしていた。Thin Client の仕様を表 8 に, 外観を図 18 に示す。

MR では 20 台の Thin Client を導入し加速器の運転に用いていた。Thin Client の運用は概ね順調であったが, 毎月 1 回程度の頻度でいずれかの Thin Client がハングアップしていた。致命的

表 7: Blade 計算機の構成の変遷

	サーバ用	MR 制御 アプリ用	MR 仮想マシン ホスト用	RCS 用
2008	HS20 ×4	HS21 ×9	-	HS21 ×2
2012	HS20 ×2	HS21 ×8 HS22 ×2 HS22a ×1	HS22 ×6	HS21 ×3 HS22 ×2
2014	HS20 ×1 HS21 ×1	HS21 ×6 HS22 ×2 HS22a ×1 HS23e ×2	HS22 ×6 HS23e ×1	HS21 ×3 HS22 ×2
2018	HS22 ×1 HS23e ×1	HS23e ×5	HS22 ×4 HS23e ×4	HS21 ×1 HS22 ×2 HS23 ×2

表 8: 端末用計算機の仕様

名称	CPU	メモリ	GPU	起動メディア
Intel NUC5i5MYHE	Core i5-5300U 2.3–2.9 GHz	8–32 GB	Intel HD Graphics 5500	SSD
Intel NUC DC53427HYE	Core i5-3427U 1.8–2.8 GHz	8 GB	Intel HD Graphics 4000	SSD
HP Compaq t5730 Thin Client	Sempron 2100+ 1 GHz	512 MB	Matrox EpicA TC2 / TC4	PXE
HP Compaq t5720 Thin Client	Geode NX 1500 1 GHz	512 MB	Matrox EpicA TC2 / TC4	PXE

とは言えないまでも加速器の運転に支障を来たす問題であったが, 原因が特定できるほど頻繁にハングアップするわけでもなかった。

Thin Client の GPU はメーカー提供のデバイスドライバが必要であったが, 2009 年でメーカーによる更新が終了しており, SL6 では Thin Client が動作しないことが判明した。2012 年に SL の配布元による SL4 のサポートが終了したことを契機に, SL6 で利用する次期端末用計算機の検討を開始した。2013 年に試験的に Intel NUC を導入し, 2014 年に 5 年程度の先を見据えて次期端末用計算機として Intel NUC を採用した [39]。

NUC は Intel が提唱した超小型 PC の規格で, オフィス向けの機種は製品寿命が長いことが期待される。Thin Client よりも小型でありながら, フル HD のディスプレイを 3 枚まで接続可能である。MR の制御システムでは 1 台の NUC につき 2 枚のフル HD のディスプレイで運用し, J-PARC 中央制御室の限られたスペースを有効に活用している。

端末用計算機の運用方法も見直した。Thin Client は X 端末として利用するだけで制御アプリケーションは Blade 計算機上で実行していたが, NUC は Blade 計算機と同等の CPU 処理能力があること, 充分なメモリを搭載できることから, NUC で直接アプリケーションを実行することとした。これにより, MR の制御アプリケーションに CSS を導入する可能になった。従来どおりアプリケーションを Blade 計算機上で実行し NUC に表示することも可能であるが, Blade 計算機と同等の CPU パワーがあること, NUC の方が空きメモリに余裕があることなどから, MR 制御グループとしては NUC 上で実行することを推奨している。

図 18: HP Compaq t5730 (右) と, Intel NUC (左) の外観。中央に置いてあるのはサイズの比較のための単 1 型乾電池。

加速器の長期シャットダウン期間中に全ての端末を一度に置き換えた場合, 評価の際に顕れなかった問題が発生すると加速器の運転再開に支障を来たすことになる。そこで, MR 加速器を運転をしながら毎週 1~2 台程度の頻度で NUC に置き換えていき, 仮に問題が発生したとしても短期間に集中しないような手法をとった。2014 年 1 月から端末計算機の機種更新を開始し, 2014 年 7 月に完了した。MR コミッショニングの際にしばしば端末用計算機が不足していたことを鑑みて, 20 台の Thin Client を 22 台の NUC で置き換えた。

2018 年 8 月現在は 35 台の NUC を運用している。そのうち 2 台は初期不良で交換となり, 3 台は故障した SSD を交換した。これまでのところ, これ以外に大きなトラブルは起こっていない。

6 まとめ

加速器の制御システムは, 加速器内の装置, ビーム物理, ビームの運転など, 加速器の全ての要素と関わりを持っている。制御システムを構築するには, さまざまな選択肢の中で妥協しバランスをとる必要がある。計算機技術の進歩に伴って, 最適な選択肢は刻々と変化する。加速器はその長い寿命に亘って性能向上が追究され続け, 制御システムへの要求仕様も変わっていく。さまざまな要求に柔軟に応答できる制御システムを構築しなければならない。25 年以上前に提唱された加速器標準モデルの考え方は, 現在でも有効である。

参考文献

[1] ICALEPCS, https://www.icalepcs.org/

[2] B. Kuiper, “Issues In Accelerator Controls”, Proceedings of ICALEPCS 91, KEK, Tsukuba, Japan, pp.602 (1991)

[3] EPICS - Experimental Physics and Industrial Control System, http://epics.anl.gov/

[4] MADOCA 制御システム, http://www.spring8.or.jp/ja/about_us/manage_structure/jasri/control_system/madoca/

[5] TANGO Cotrols, http://www.tango-controls.org/

[6] LabVIEW, http://www.ni.com/ja-jp/shop/labview.html

[7] Vista Control Systems, https://www.vista-control.com/

[8] PythonCA; https://pypi.python.org/pypi/PythonCA/

[9] SAD - Strategic Accelerator Design, http://acc-physics.kek.jp/SAD/

[10] LXI and VXI-11, http://www.lxistandard.org/about/vxi-11-and-lxi.aspx

[11] EDM - Extensible Display Manager, http://ics-web.sns.ornl.gov/edm/edmUserManual/

[12] MEDM - Motif Editor and Display Manager, http://www.aps.anl.gov/epics/extensions/medm/index.php

[13] CSS - Control System Studio, http://controlsystemstudio.org/

[14] Strip Tool, http://www.aps.anl.gov/epics/extensions/StripTool/index.php

[15] EPICS ArchiveViewer, http://ics-web.sns.ornl.gov/archive/viewer/ https://slacmshankar.github.io/epicsarchiver_docs/archiveviewer.html

[16] N. Kamikubota, *et al.*, "Data Archive System for J-PARC Main Ring", Proceedings of IPAC'10, paper WEPEB001, p2680-2682 (2010)

[17] T. Iitsuka, *et al.*, "Advanced Applications of Archive Data for J-PARC MR", Proceedings of the 8th Annual Meeting of particle Accelerator Society of Japan, paper MOPS098, p.579 (2011)

[18] N. Kamikubota, *et al.*, "Plan to Develop Overall Status Screens for J-PARC Accelerator Complex", Proceedings of the 8th Annual Meeting of Particle Accelerator Society of Japan, paper MOPS089, p543 (2011)

[19] The EPICS Archiver Appliance, https://slacmshankar.github.io/epicsarchiver_docs/

[20] CS-Studio Guide - Archive System, http://cs-studio.sourceforge.net/docbook/ch11.html https://ics-web.sns.ornl.gov/css/devel.html#archiver

[21] Cassandra PV Archiver, https://oss.aquenos.com/cassandra-pv-archiver/

[22] EPICS Channel Archiver, https://github.com/EPICSTools/ChannelArchiver

[23] Channel Archiver shell tools, https://github.com/epicsdeb/carchivetools

[24] Channel Archiver to Archiver Appliance data file converter, https://github.com/mdavidsaver/ca2aa

[25] Java Archive Viewer, https://github.com/slacmshankar/epicsarchiverap_archiveviewer

[26] K. Doi *et al.*, "Current status and future prospect of J-PARC main ring archive web viewer", Proceedings of of the 14th Annual Meeting of Particle Accelerator Society of Japan, paper WEP103, (2017).

[27] S. Yamada *et al.*, "Deployment of Archiver Appliance at J-PARC Main Ring", Proceedings of of the 14th Annual Meeting of Particle Accelerator Society of Japan, paper WEP100, (2017).

[28] N. Kamikubota *et al.*, "Network System Operation for J-PARC Accelarators", Proceedings of ICALEPCS 2017, Barcelona, Spain, paper THPHA047, pp.1470-1473 (2017)

[29] Scientific Linux, http://www.scientificlinux.org

[30] N. Kamikubota, *et al.*, "J-PARC Control toward Future Reliable Operation", Proceedings of ICALEPCS 2011, Grenoble, France, paper MOPMS026, pp.378-381 (2011).

[31] N. Kamikubota *et al.*, "Operation Experience and Migration of I/O Controllers for J-PARC Main Ring", Proceedings of PCaPAC2016, Campinas, Brazil, paper THPOPRPO09, pp.101-104 (2016).

[32] F. Tamura, *et al.*, "J-PARC Timing System", Proceedings of ICALEPCS 2003, Gyeongji, Korea, paper TU115, pp237 (2003).

[33] J. Odagiri, *et al.*, "Development of Embedded EPICS on F3RP61-2L", Proceedings of the 5th Annual Meeting of Particle Accelerator Society of Japan and the 33rd Linear Accelerator meeting in Japan, paper FO27 (2008).

[34] EPICS Device and Driver Support for Yokogawa's F3RP71/F3RP61 PLC, `http://www-linac.kek.jp/cont/epics/f3rp61/`

[35] N. Kamikubota, *et al.*, "Virtual IO Controllers at J-PARC MR using Xen", Proceedings of ICALEPCS 2011, Grenoble, France, paper WEPMU039, pp.1165-1167 (2011).

[36] N. Kamikubota, "J-PARC MR 制御での仮想マシンの応用", Proceedings of the 10th Annual Meeting of Particle Accelerator Society of Japan, paper SAP092 (2013).

[37] N. Kamikubota, *et al.*, "Improvement of Computer Systems for J-MARC MR Control", Proceedings of the 9th Annual Meeting of Particle Accelerator Society of Japan, paper WEPS117, pp.741, (2012)

[38] S. Yamada, "J-PARC Maoin Ring への小型ファンレスサーバを用いた IOC の導入", Proceedings of the 13th Annual Meeting of Particle Accelerator Society of Japan, paper MOP092, pp.643 (2016).

[39] S. Yamada, *et al.*, "Renovation of Control Computers for J-PARC Main Ring", Proceedings of the 11th Annual Meeting of Particle Accelerator Society of Japan, paper SAP099, pp.782 (2014).