KEK 電子陽電子入射器における Podman を用いた Archiver Appliance の導入

INTRODUCTION OF ARCHIVER APPLIANCE USING PODMAN AT THE KEK ELECTRON/POSITRON INJECTOR LINAC

佐武いつか[#, A)], 佐藤政則[A, B)], 王迪[A, B)], 草野史郎[C)], 工藤拓弥[C)], 櫻井雅哉[D)]
Itsuka Satake[#, A)], Masanori Satoh[A, B)], Di Wang[A, B)], Shiro Kusano[C)], Takuya Kudou[C)], Masaya Sakurai[D)]
[A)] High Energy Accelerator Research Organization (KEK), Accelerator Laboratory
[B)] The Graduate University for Advanced Studies (SOKENDAI), Department of Accelerator Science
[C)] Mitsubishi Electric System & Service Co., Ltd
[D)] Kanto Information Service (KIS)

Abstract

The KEK electron/positron injector linac delivers beams to four storage rings and a positron damping ring. To maintain the stable beam operations, continuous technical research and development are essential. A comprehensive data acquisition system is indispensable for recording the large number of operational parameters from both accelerator components and the surrounding environment. At the KEK injector linac, the Archiver Appliance serves as the core data acquisition software, efficiently collecting and storing over 170,000 process variables. To improve system reproducibility, simplify deployment, and enhance operational efficiency, we containerized the Archiver Appliance using Docker framework. In this presentation, we report on our insights and experiences obtained from deploying the Archiver Appliance in a Docker environment, building on our prior experience with virtual machines.

1. はじめに

加速器運転における大規模なデータの記録・分析ニーズの高まりにともない、Archiver Appliance（以下、 AA）[1]は、重要なデータ収集基盤として多くの施設で利用されている。AA は、Experimental Physics and Industrial Control System (EPICS) [2]と連携して動作するアーカイブシステムで、EPICS が提供する Process Variable (PV) をリアルタイムで取得・記録・検索できる。KEK 電子陽電子入射器[3]（以下、KEK 入射器）においても、仮想マシン 1 台と物理サーバ 1 台で構成された 2 台の AA を運用しており、現在 17 万点以上の制御対象をアーカイブしている[4]。

従来、物理サーバや仮想マシン（VM）環境に導入されることが多かったが、近年ではコンテナ技術を活用してシステムを構築・運用する事例が多数の現場に導入されている。AA は、Java 実行環境や Tomcat、依存ライブラリの組み合わせにより構成されており、導入時にはソフトウェアのバージョンや環境設定を適切に整える必要がある。こうした環境依存性を最小限に抑え、構成の再現性を確保する手段として、コンテナによる統合的な環境構築は効果的である。また、ホスト OS に依存せず移植が容易であり、他環境への展開やアップデートの検証も効率的におこなうことができる。本研究では、AA のコンテナ化に向けた整備を進め、その構成例と運用上の特性を報告する。

2. AA コンテナの概要

2.1 システム概要

KEK 入射器の制御システム環境において、AA をコンテナ化し、整備・動作試験をおこなった。本システムは、仮想マシン上にコンテナ実行環境を整備し、Podman を用いてアプリケーションコンテナとして AA を展開している。Figure 1 に本システムの概要を示す。

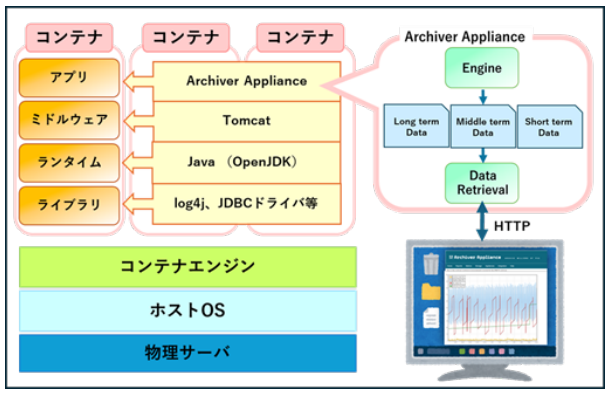

Figure 1: Overview of the Archiver Appliance container system architecture.

2.2 コンテナ実行環境

本システムのコンテナランタイムには、Red Hat 系ディストリビューションと親和性が高く、yum/dnf リポジトリから標準で導入可能である Podman を採用した。Podman は rootless 実行が可能で、Docker デーモンのような常駐プロセスを必要とせず、セキュリティ上の制約が厳しい制御ネットワークでも有効に運用できるという利点がある。また、Red Hat Enterprise Linux では近年、Docker の公式サポートが終了し、Podman への移行が推奨されている点も選定理由の一つである。

コンテナは、マルチステージビルドと呼ばれる Docker の機能を活用して構築されている。マルチステージビル

itsuka.satake@kek.jp

ドとは、Dockerfile 内で複数の「ステージ」を定義し、それぞれのステージで異なる目的の処理をおこなうことで、最終的なコンテナイメージを軽量かつ効率的に構築できる技術である。これにより、ビルド時に必要なツールや一時ファイルを最終イメージに含めずに済むため、セキュリティや性能の面でも利点がある。

2.3 Dockerfile によるコンテナ構築と設定

今回の Dockerfile では、以下の 2 つのステージが定義されている。

- ステージ 1:ダウンロード・展開ステージ
  軽量な Linux ディストリビューションである Debian Bookworm Slim をベースに、AA の公式 GitHub リポジトリからアーカイブファイル(archappl_v2.0.10.tar.gz)をダウンロード・展開する。展開された WAR ファイルは、次のステージで使用される。
- ステージ 2:実行環境ステージ
  Tomcat 9.0.80 + OpenJDK 21 をベースとした Java アプリケーションサーバ環境を構築する。ステージ 1 で取得した WAR ファイルを Tomcat の webapps ディレクトリに配置し、AA の各種設定ファイルを /etc/archappl/に配置する。WAR ファイルの配置には、Tomcat の自動展開機能を利用する方法を採用しており、webapps ディレクトリにコピーすることで、コンテナ起動時に各 Web アプリケーションが自動的にデプロイされる。また、.war ファイルを展開した状態で配置する方法などがあるが、本構成ではシンプルさと可搬性を重視し、自動展開による方式を用いている。

また、AA の永続化層として使用する MySQL との接続を確立するため、MySQL Connector-J(バージョン 9.3.0)を Tomcat のライブラリディレクトリに追加している。

保存領域は/storage 以下に構成されており、短期(STS)、中期(MTS)、長期(LTS)保存用のディレクトリがそれぞれ分離されている。Tomcat のログディレクトリは /storage/logs にリダイレクトされており、ログファイルの管理をおこなう。さらに、環境変数を用いることで、システムの基本設定(使用するデータベースや保存先のディレクトリ)や記録対象の定義、EPICS との通信に必要なネットワーク設定(EPICS_CA_ADDR_LIST など)が自動的に読み込まれるよう構成している。AA コンテナの構成要素および各種ソフトウェアのバージョンを Table 1 に示す。

Table 1: Configuration of Archiver Appliance Container

Base OS	Debian Bookworm Slim
Application Environment	tomcat:9.0.80-jdk21-openjdk-slim-bullseye
AA	2.0.10
Java	OpenJDK 21
Web Server	Apache Tomcat 9.0.80
Database	MySQL 8.4.5(Connector-J 9.3.0)

2.4 Podman による実行環境

Podman による rootless 実行は、ユーザーのホームディレクトリにコンテナイメージや一時領域を格納する設計となっている。しかしながら、本環境では、ユーザーのホームディレクトリが NFS 上に存在しており、Podman の rootless モードがそのままでは動作しないという制約がある。Podman の rootless モードでは、Linux のユーザー名前空間(user namespace)機能を利用して、rootless ユーザーがコンテナ内で UID/GID を仮想的に変更し、あたかも root 権限を持つかのように動作することが可能となる。この仕組みにより、ホスト側の UID/GID を直接使用することなく、コンテナ内でのファイル操作(例えば、chown による所有権変更)などを安全におこなうことができる。

NFS は、クライアントから送られる UID/GID を数値のまま扱い、サーバ側でも同様に数値として権限チェックをおこなう仕組みとなっている。そのため、rootless Podman が使用する仮想的な UID/GID の情報は NFS サーバ側では認識されず、コンテナ内で行われた所有権変更(chown)などの操作が拒否される。このように、ユーザー名前空間による UID マッピングが通用しないことが、NFS 環境における rootless モードの制約の根本的な原因である。この制約を回避するため、Podman が使用する実体データ(コンテナ・イメージ・一時領域)をローカルディスク上に退避させる構成とした。

2.5 NFS 環境での rootless 実行対応手順

1) サブ UID/GID 範囲の割り当て
2) ローカルディスク上に Podman 用ディレクトリ作成
3) Podman のストレージ設定ファイル作成

Podman のユーザー名前空間機能を有効にするため、対象ユーザーに対して仮想的な UID/GID 範囲を付与する。これは usermod コマンドを用いる。また、設定ファイル(~/.config/containers/storage.conf)を作成し、Podman に対してストレージの使用先をローカルディスクに明示的に指定する。この設定により、Podman は NFS 上のホームディレクトリをバイパスし、指定されたローカルディスク領域を使用して rootless モードでのコンテナ実行が可能となる。

2.6 MySQL コンテナとサービス管理

AA のデータベースとして使用する MySQL については、公式のコンテナイメージを Podman 経由で取得し、Quadlet を用いて systemd による永続的なサービスとして管理している。一方、AA コンテナについては、独自に作成した systemd サービスユニットを通じて起動・停止を制御している。AA コンテナは、各種設定ファイルやデータ保存領域をホスト側と volume 共有する形で構成されており、設定ファイル(tomcat_conf_server.xml)やデータ保存領域、ログ出力先をホストのディレクトリと連携させている。また、環境変数を用いてタイムゾーン(Asia/Tokyo)やアプライアンスの識別名(appliance0 など)を指定しており、複数の AA コンテナを同一ホスト上で運用することが可能となる。

2.7 ネットワーク構成

EPICS との通信には Channel Access（CA）プロトコルを使用しており、PV の検出にはブロードキャスト通信が必要となる。通常のコンテナネットワークモード（例：bridge）では、コンテナがホストのネットワークとは分離されるため、CA ブロードキャストがコンテナに届かないという問題がある。このため、本研究ではコンテナのネットワークモードに--network host を指定し、コンテナがホストと同一のネットワークスタックを使用するように構成した。これにより、EPICS IOC との通信が可能となる。

各コンテナは--network host オプションによりホストネットワークと直接接続され、ポートの分離によって多重起動時の競合を回避している。複数のアプライアンスを同一ホスト上で運用するには、各インスタンスに異なるポート番号を割り当てる必要がある。運用規模が拡大するにつれて、ポートの管理がより重要な課題となる。AA は Tomcat 上で動作するため、ポート設定は server.xml によって制御される。本構成では、アプライアンスごとに異なる tomcat_conf_server.xml を用意し、起動時に-v オプションを用いてマウントすることにより、Dockerfile は共通のまま、アプライアンスごとの設定を柔軟に切り替える構成とした。

2.8 ディレクトリ構成とストレージ構成

AA コンテナ内におけるディレクトリ構成を Fig. 2 に示す。設定ファイルは/etc/archappl/に集約されており、Tomcat 関連のアプリケーションは/usr/local/tomcat/以下に配置される。

アーカイブデータやログなどの記録情報は storage/に格納され、このディレクトリはコンテナ起動時に Network Attached Storage をマウントする設計となっている。

一般的に、コンテナ内のファイルシステムは一時的なものであり、特に設定をおこなわない場合、停止時にデータは失われる。この構成により、コンテナの停止や再起動後もデータが保持され、安定した運用が可能となる。

```
/
├── etc/
│   └── archappl/
│       ├── appliances.xml  archappl.properties  policies.py
│       ├── tomcat_conf_server.xml  tomcat_conf_context.xml  log4j.properties
├── usr/
│   └── local/
│       └── tomcat/
│           ├── webapps/
│           ├── lib/
│           │   ├── mysql-connector.jar  log4j.properties
│           └── conf/
│               ├── server.xml  context.xml
├── storage/
│   ├── sts/
│   ├── mts/
│   ├── lts/
│   └── logs/
```

Figure 2: Internal directory layout of Archiver Appliance container.

3. 仮想マシン構成

3.1 仮想マシン構築と管理ツールの活用

AA コンテナは、KEK 入射器の制御サーバ環境上で動作している。仮想化基盤としては、ProxmoxVirtual Environment (VE)を用いている。Proxmox VE は Debian ベースのオープンソース仮想化プラットフォームであり、KVM によるフル仮想化と LXC による軽量コンテナ仮想化を同一環境で運用できる点が特徴である。Web ベースの管理 GUI を用いることで、仮想マシンの作成や設定、スナップショットの取得、自動バックアップなどが容易に実行可能である。さらに、クラスタ構成や高可用性にも対応しており、制御系システムに求められる拡張性と信頼性を備えている。

Figure 3 に、Proxmox の管理 GUI 上で表示される仮想マシンのリソース情報やステータスを示すサマリ画面を示す。

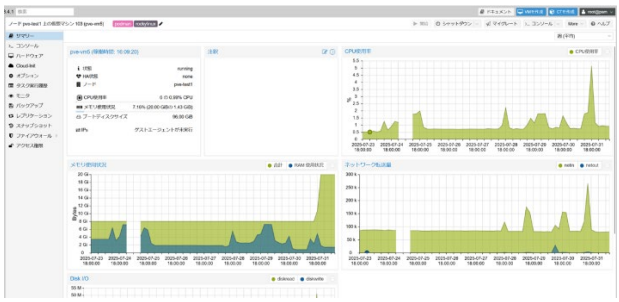

Figure 3: Summary view of virtual machines on Proxmox VE.

仮想マシンへのリモート操作には SSH クライアントとして MobaXterm を用いた。MobaXterm は、Windows 上で動作する多機能なターミナルソフトウェアである。これにより、仮想マシンへのアクセスやログ確認、設定変更作業などを効率的におこなうことができた。

3.2 コンテナ実行基盤の構成

Proxmox VE 上に仮想マシンを構築し、その上で Podman を用いて AA コンテナを実行した。仮想マシンの構成は、Table 2 に示すとおりである。

Table 2: Virtual Machine Configuration for Archiver Appliance Container

Memory	20 GiB
Virtual CPU	6 cores (1 socket)
BIOS	SeaBIOS
Storage	96 GB SSD (local-lvm)
SCSI controller	VirtIO SCSI single
Network	Virtio
OS	Rocky Linux9.5

なお、現在の動作試験は加速器停止期間中に実施しており、実際の運転中の負荷を考慮した評価は今後の課題である。

4. 動作試験と比較

4.1 AA コンテナ動作試験

AA のコンテナ化にともない、基本動作確認、GUI 操作試験、およびアーカイブ試験を実施した。加速器停止期間中に初期評価をおこない、システムとしての立ち上げや基本動作に問題がないことを確認した。基本動作確認では、コンテナの起動・停止処理、ログの生成、初期設定の読み込み動作を中心に検証をおこなった。GUI 試験では、Web ベースの管理画面が正常に表示され、各種操作が想定どおりに動作することを確認した。Figure 4 に、Archiver Appliance の Web ブラウザ上の管理画面を示す。

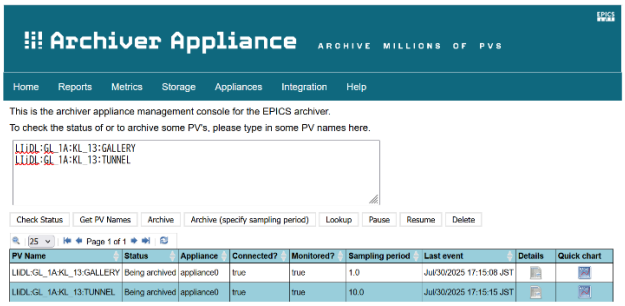

Figure 4: Screenshot of management GUI interface of Archiver Appliance container.

アーカイブ試験は、Python スクリプトおよび GUI 操作を用いて、PV の新規登録および Pause/Resume 機能の動作確認を実施した。これらの操作において、登録情報の反映および状態遷移が正しく行われることを確認した。

試験中のトラブルとしては、3 台の AA コンテナを同時起動したところ、メモリ不足によりプロセスが Out of Memory（OOM）エラーにて強制終了する事象が発生した。この事象を受けて、仮想マシンのメモリ割当を当初の 8 GB から 20 GB へと増強し、3 台の AA コンテナが正常に動作することを確認した。なお、本試験は加速器停止中という限定的な負荷条件下でおこなわれたため、今後は加速器運転中の実負荷に基づいたリソース構成の最適化をおこなう必要がある。

4.2 従来との比較

AA コンテナの構築と動作試験を通じて、コンテナ環境における構築・運用について評価をおこなった。これを踏まえ、従来の物理サーバや仮想マシンとの特徴の違いについて検討する。Figure 5 に、各アーキテクチャの違いを模式的に示す。

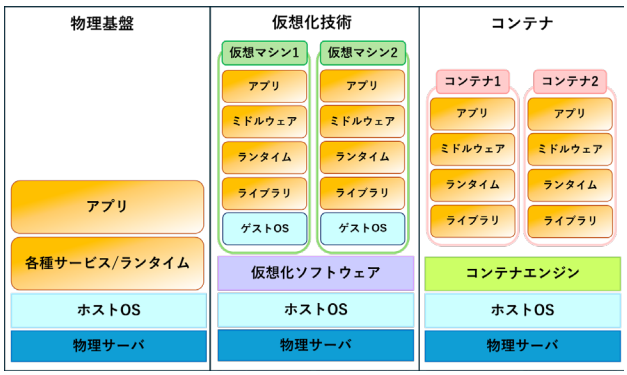

Figure 5: Layered architecture comparison of physical infrastructure, virtual machine and container.

物理サーバ環境では、ソフトウェアの依存関係や構成管理が煩雑になりやすく、再構築や移設のたびに多大な工数を要する。仮想マシンはシステムの独立性を確保できる一方で、OS の起動や更新に時間がかかり、リソース（CPU やメモリ）消費も大きくなりやすい。

これに対して、コンテナはホスト OS のカーネルを共有しながらもアプリケーションを個別の環境として隔離して動作させるため、より軽量で高速な展開が可能であり、複数インスタンスの並列実行にも適している。また、Dockerfile を用いた環境定義により、OS バージョンや依存ライブラリの差異に左右されず一貫性のある構築が可能となり、アップデートや再構築の自動化にも寄与する。これにより、保守性と拡張性の両立が求められる長期運用システムにおいても有用性が高い。

現在は systemctl を用いた単純な構成（数台の AA コンテナと 1 台の MySQL コンテナ）で運用上の不便は見られないが、今後の拡張や管理効率の向上を見据え、Kubernetes などのオーケストレーションツールの導入による運用の自動化・効率化についても検討していく計画である。

5. まとめと課題

KEK 入射器において、AA コンテナの構築と動作試験をおこなった。Docker のマルチステージビルド機能を活用して軽量かつセキュアなコンテナイメージを構築し、MySQL との連携や環境変数による動的設定を導入することで、AA が正常に動作することを確認できた。従来の AA 運用と比較して、コンテナ環境は構築の迅速性、環境の可搬性、運用の柔軟性において明確な利点を示した。

今後の課題としては、加速器運転中の実負荷を踏まえたリソース構成（メモリ割当やコンテナのリソース制限）の最適化を進める必要がある。また、Zabbix を用いたシステム監視体制を整備し、安定した運用を確保する体制を整える。さらに、Kubernetes などのオーケストレーションツールの導入による運用自動化やスケーラビリティ向上についても検討していく。

参考文献

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

[2] EPfOSICS, http://www.aps.anl.gov/epics/

[3] H. Ego *et al.*, “KEK 電子陽電子入射器アップグレードによるビーム入射性能向上”, PASJ2025, Tokyo, Japan, Aug. 2025, THP004, this meeting.

[4] I. Satake *et al.*, “PROBLEMS AND SOME APPROACHES DURING OPERATION OF ARCHIVER APPLIANCE IN KEK ELECTRON POSITRON INJECTOR LINAC”, in Proceedings of the 18th Annual Meeting of Particle Accelerator Society of Japan, Aug. 9-12, 2021, pp. 528-531.