実験制御用プログラム開発のモダン化とビームライン制御・データ収集・オンライン解析プラットフォーム：BL-774

国立研究開発法人理化学研究所　放射光科学研究センター
本村　幸治
公益財団法人高輝度光科学研究センター　ビームライン技術推進室
安田　伸広
公益財団法人高輝度光科学研究センター　回折・散乱推進室
国立研究開発法人理化学研究所　放射光科学研究センター
今井　康彦
公益財団法人高輝度光科学研究センター　XFEL 利用研究推進室
国立研究開発法人理化学研究所　放射光科学研究センター
中嶋　享、城地　保昌

Abstract

SPring-8/SACLA において開発と導入が進められてきた「ビームライン制御・データ収集・オンライン解析プラットフォーム：BL-774」は、2023 年夏の時点で、SPring-8 の 5 つのビームラインにおいて、光学機器の制御を含めたユーザー共用が行われている。ビームラインにおける BL-774 のシステム構成は、ネットワークの接続を含めてビームライン毎に完結的である。本稿では、特に実験制御用プログラムの作成・開発のあり方に着目し、現代に相応しいソフトウェア開発のコンセプトと、そのコンセプトを実現していく BL-774 の基幹的なソフトウェア・システム「774 Basic System」を紹介する。774 Basic System は、実験制御用プログラムの作成段階ではロジックに関わる部分に注力できるよう、周辺のパーツがビームラインでの実験に適切な形、機能、規模で実装されていて、利用形態に沿って選択可能な複数のマイクロサービスとして提供される。今後、機能の拡張を図るとともに幅広いビームラインでの導入が進められていく見通しである。

1. はじめに

SPring-8 では、そのアップグレード・プロジェクトである SPring-8-II プロジェクト[1, 2]の計画が進められているところであり、光源性能の利点に対して最大限の活用を加速するために、光源だけではなく、光学系、試料準備、2 次元検出器、データ解析、実験制御システムなどのビームライン・コンポーネントに対するアップグレードも進められている。その中で、SPring-8-II のビームラインに求められる機能には、(i)システムの堅牢性と柔軟性、(ii)コンフィグレーション管理、(iii)実験ワークフローのシーケンス管理、(iv)リモート制御、(v)最大 1 メガ・サンプル/秒（MSPS）の高速データ収集に関する項目を挙げることができる。それぞれの目的を達成するための個々のタスクは、項目を越えて互いに密接に関連しているため、迅速性に加えて一貫性を持って統合されていく必要がある。そのため、著者らは「ビームライン制御・データ収集・オンライン解析プラットフォーム：BL-774」[3, 4]の開発と導入を進めてきた。

ここで BL-774 の名称についてであるが、日本語では、語呂合わせで「774」に"nanashi"と音を当てはめている。これは「名無し」に由来し、命名時に、このプラットフォームが持つ機能の範囲を示す概念に対して、特定の一語をあてはめることが困難であったことに関係している。なお、英語での気の利いた読み方は今後、決める予定である。

さて、BL-774 では、これまでに、(i)、(ii)、(iv)の 3 つの機能を実装している。BL-774 の(i)、(ii)の機能を用いることで、新しいハードウェアあるいはソフトウェアの開発者は、実際のビームライン環境を使って開

発や試験、運用への組み込みを行うためのワークフローを利用することができる。このワークフローは、専用のラピッド・アプリケーション・デベロップメント（RAD）と Web ベースのグラフィカル・ユーザー・インターフェース（GUI）に象徴される 2 フェーズ開発が組み込まれたものである。ここで、このワークフローを、本稿の第 3 章「BL-774 のソフトウェア・システム」の内容に沿ってあらかじめ概観しておくことにする。本稿で紹介する BL-774 のソフトウェア・システム「774 Basic System」[5]は、プログラミング言語である Python[6]の実行環境などと一括して PC 等に構築される。この際に、パルスモーター・コントローラーや 2 結晶分光器（DCM）など、ビームラインで標準的に使用される機器の制御を扱う「ecpy」という名前のソフトウェア・モジュールがあらかじめ組み込まれる。この ecpy は、774 Basic System が持つクラス・ライブラリ「774 API」を構成する一つである。API（Application Programming Interface）については、この章の後半で再び扱うことにする。したがって、774 Basic System の利用者は、Python 環境で、ecpy をインポートすれば 774 API の利用を開始できる。BL-774 の RAD 環境は、JupyterLab[7]などの Web ベースの対話型実行環境を含め、Python の開発に用いられるインターフェースで ecpy をインポートすることにより、774 API を直接的に利用できる環境である。そこで、2 フェーズ開発の最初のステップとして、ハードウェアの開発者は、ビームラインに新しい機器を設置した後に行う初期検査や、それらを使用した初期実験を、BL-774 の RAD 環境を用いて実行できる。ここでの主目的は、開発中の機器における操作手順や取得した値の演算方法のような、結果を得るまでの具体的な処理の流れを規定するロジックを作成したり、それらの中で用いられる変数を決定したりすることである。これらのロジックや変数は、ノートブック形式のファイルなどに保存され、必要に応じて他の資料と共に、ソフトウェアの開発者と共有される。次のステップでは、前のステップで作成されている、プログラミング言語で記述されたロジックを基に、ソフトウェアの開発者が堅牢性も考慮したソフトウェアを作成し、公式の API あるいは Web GUI としてビームラインにリリースしていく。ここでは、特にソフトウェア開発の観点で重要な、エラー処理やカプセル化が追加されたスクリプトが作成される。さらに、これらのスクリプトを利用して Web GUI のアプリケーションが作成される。このワークフローを経ることにより、利用者は、Web GUI を使用して機器の基本操作を実行できるだけでなく、API そのものの利用においても、さまざまな可変パラメーターを Web インターフェースを使用して設定できるようになる。さらに、これらの Web GUI と対話型実行環境等を組み合わせて使用することで、さまざまな研究開発プロジェクトを迅速かつ簡単に実行できるようになる。これは、新しい機器の導入時だけではなく、実験準備やデータ収集においても当てはまることであり、本稿のテーマである実験制御用プログラムの作成・開発においても同様である。また、本稿では詳細に触れないが、BL-774 の(iv)の機能を用いることにより、SPring-8 においてリモート制御を行う際に運用面で必要とされている安全に関する基準を標準的に満たすことが可能になる。

ところで、放射光ビームラインで行われる制御には、互いに関連がありながら異なる側面を持つものとして、ビームライン機器制御と実験機器制御を挙げることができる[8]。前者は、主にフロントエンド（FE）や輸送チャネル（TC）において光を操作することを主目的とする機器に対する制御であり、常設的で共同利用性の高いものと位置づけられる。一方で、後者は、実験ハッチ内の試料ステージや検出器を操作することを主目的とする。これは、前者の間接的な操作を含めて、実験に必要な操作を統合して扱うものであり、常設的な機器のみで構成される場合もあれば、一時的な持ち込み機器を含めて構成される場合もある。774 Basic System は、複数のビームラインにおいて前者および、後者の中の制御目的で標準的に多用される機器の操作を対象とし、後で述べるコンセプトに基づいたソフトウェアの実装を行ってきたものである。さらに、ビームライン制御のためのフレームワークに普遍的に望まれる形態として、再利用性に優れた API 群の整備やエコシステムによる供給様式を挙げることができる[8]。「API（Application Programming Interface）」という用語は既に前で使用しているが、その定義は文脈に応じて広狭がある。ここでは、「他のソフトウェアに特定のサービスを提供する明確に定義されたイ

ンターフェース」[9]としておく。また、「エコシステム（ecosystem）」は、本来は「生態系」を意味する言葉であるが、その、互いに独立している多様な構成要素が相互に依存しあい連携しながら 1 つの大きなシステムを形成する様を表す使い方で、さまざまな分野において利用されている。例えば、「地域経済エコシステム」や「ビジネス・エコシステム」、「イノベーション・エコシステム」といった言葉をお聞きになったことのある読者もおられるかもしれない。IT やソフトウェア・システムの分野でエコシステムという言葉が利用される場合、そこでは、「ソフトウェアの再利用が大きな特徴であり、『全てを自分で作らず、使えるものは使い、車輪の再発明はしない』が基本的な考え方」[8]に基づいて、システムが構築されていくことになる。なお、この「車輪の再発明」もまた、IT やソフトウェア・システムの分野だけでなく、さまざまな分野でしばしば目にする比喩表現であろう。774 Basic System は、これらの API やエコシステムの概念も実現している。

さて、この後に述べるように、BL-774 にはいくつかの利用形態が存在するのであるが、本稿を執筆している 2023 年夏の時点で、主に光学機器の制御に適用し、ユーザー共用を行っているビームラインには、BL20B2[10]、BL09XU[11]、BL13XU[12]、BL46XU[13]、BL07LSU がある。今後、適用範囲を拡大するとともにその他のビームラインでも導入が進められていく見通しである。

本稿では、まず、ビームラインのネットワークから見た BL-774 の位置づけと、BL-774 のソフトウェア・システムの仕組みを、ビームライン実験の担当者や利用者の方々と共有できるよう紹介する※1。その上で、BL-774 のサンプル・コードの実例を用いて、実用的なプログラミングへの展望についても触れることにしたい。

※1 本稿では、774 Basic System について執筆時（2023 年夏）の最新バージョン（Summer 2023 Version）に基づいて記述している。今後のアップデートに伴い仕様の変更や追加等の可能性があるため、774 Basic System を実際に利用する場合には、対応するバージョンに関する情報も併せて参照していただくことを推奨する。

2. ビームラインにおける BL-774 のシステム構成

ビームライン制御で対象となる被制御機器の数はビームラインあたり 100 個程度が目安である。通常、これらの機器の詳細はそのビームラインの担当者が把握している。また、特に実験ハッチでは、ユーザータイム中において機器の新規追加や移動、入れ替え等の必要が実験に応じて頻繁に発生する。そこで、BL-774 では、既存の（加速器制御）システムと違って、システムの基本構成をビームライン単位で完結させた。この規模にシステムを集約したことで、適切な設計と実装が図られた。

BL-774 をビームラインへ導入するには、2021 年度以降にビームライン単位で段階的に導入が進められている「ビームラインネットワーク」と呼ばれる新しいネットワークと併せて導入することが標準形である。図 1 は、ビームラインで扱われる機器を操作するためのネットワーク接続とビームライン制御が、ビームラインネットワークと BL-774 の導入前後でどのように変化するのかを、対象となる機器のまとまりごとに示している。なお、ここでは基本的な考え方を示すために、ビームラインのネットワークの一部を切り出し、想定される事例を簡略化して図示した。

まず、図 1(a)は、従来のビームラインのネットワーク接続と制御の経路を示す。蓄積リングやフロントエンド、光学ハッチには、挿入光源（ID）、メイン・ビーム・シャッター（MBS）、ダウン・ストリーム・シャッター（DSS）、DCM、ミラー等の装置があり、実験ホール内の実験ステーション等には、ビームライン制御に用いられる PC が置かれている。ビームラインの PC から、これらの装置を操作する場合には、制御に関するソフトウェアを受け持つ「ビームライン・ワークステーション（BL-WS）」[14, 15]が介在してきた。このためビームラインで標準的に使用されているネットワーク「BL-USER-LAN」に接続された PC から、「制御 LAN」に接続された BL-WS に通信するためには、ネットワーク間の特別な通信許可が必要である。多くの場合で、ビームライン制御に用いる PC には、既に通信許可が与えられている IP アドレスを使用する方法が取られてきた。一方、実験ハッチで使われる試料ステージ、2 次元検出器、Web カメラを制御するには、やはり BL-USER-LAN を用いたネットワーク接続か、独自のローカル LAN を用いたネットワーク構成を用いてきた。

図 1(b)では、BL-774 を、ビームラインネットワー

クと併せて導入した際の、ネットワーク接続を示している。ID や MBS、DSS については、従来通り、BL-WS 内のソフトウェアの機能を利用するものの、ビームラインの PC と BL-WS の通信は「774 サーバー」と呼ばれる 774 Basic System がインストールされた計算機が中継する。一方、主に TC に配置されている光学機器、および、FE に配置されているスリット等の機器については、774 サーバー上のソフトウェアが通信接続と制御を直接的に受け持つ。

ビームライン機器のためのソフトウェア開発の観点から見れば、この構成は、ビームライン制御としてのまとまりをより意識したものである。なぜならば、ビームラインの PC と 774 サーバーは、共に同じ「BL-core Zone」[※2] と呼ばれるネットワーク上に存在し、ビームラインの PC が 774 サーバーと通信するための特別な通信許可設定は不要となるからである。BL-774 の導入時に標準的に設置される 774 サーバーには、BL-WS と通信するための通信許可設定が既定として設定されることになっているため、ビームライン担当者がこの申請を改めて行う必要はない。

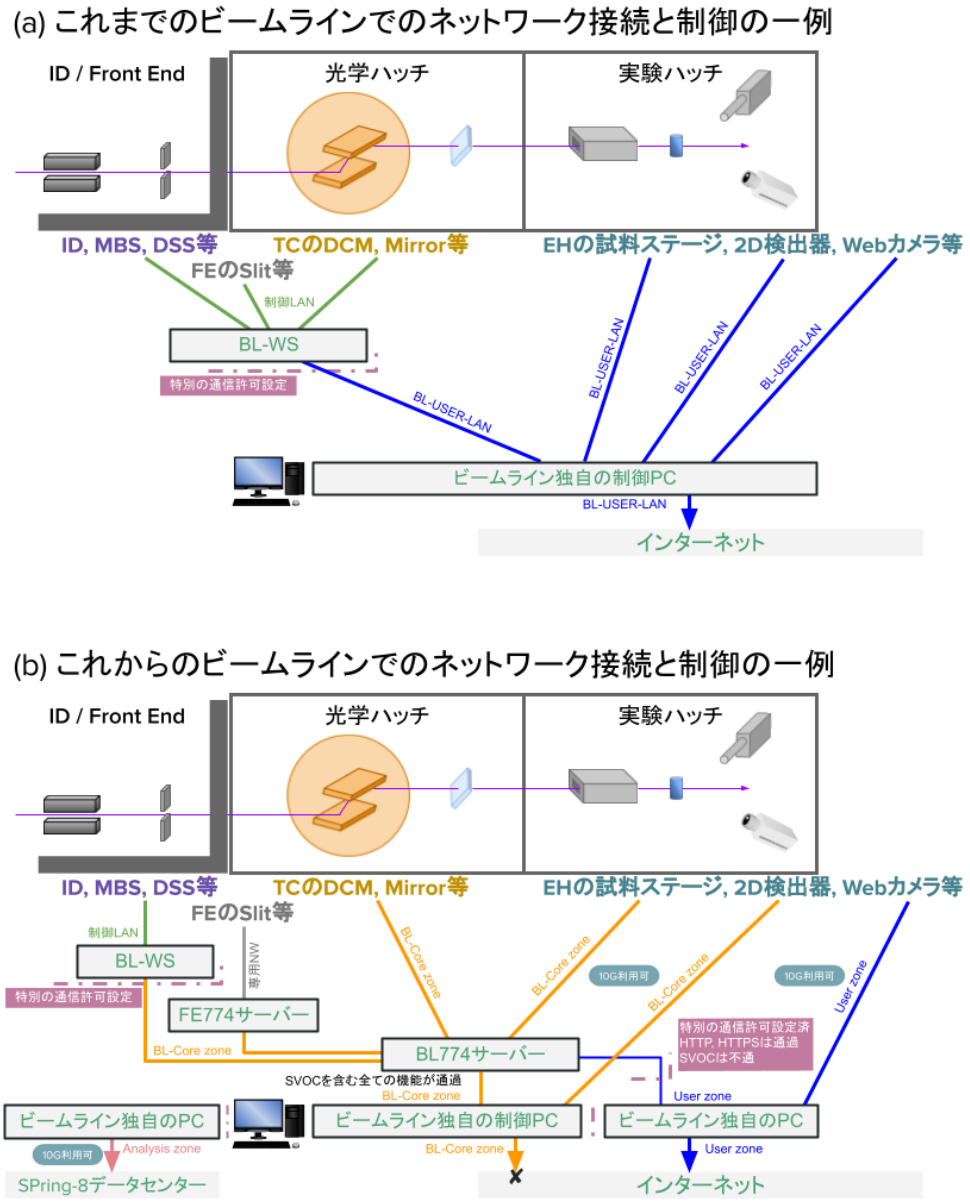

図 1　ビームラインで扱う機器に対するネットワーク接続とビームライン制御の(a)これまでと(b)これから。

また、実験ハッチ内の機器に対するネットワーク接続については、複数の経路が利用できる。例えば、試料ステージの駆動に用いるパルスモーター・コントローラーなど、774 サーバーが標準的にサポートする機器は、BL-core Zone に接続したうえで、774 サーバーのソフトウェアが制御を受け持つことができる。この場合には、光学機器と試料ステージ等が同一の 774 サーバーによって制御される構成になることから、両者の動作を複合的に組み合わせるスキャン測定のような制御のためのソフトウェアも、ビームラインの範疇において、より簡潔に構築できるようになることも特筆すべき点である。

それに対して、専用のフレーム・グラバー・ボードを用いる 2 次元検出器など、774 サーバーが標準的にサポートしていない機器は、774 サーバー以外の PC が直接の制御を受け持つことになる。なお、詳細は省くが、ビームラインネットワークでは、BL-core Zone のほかに、「User Zone」と「Analysis Zone」と呼ばれるネットワークも利用することができるようになる。[※2] このゾーン名称は本稿の執筆時の仮称であるが、2023 年度中に確定する見込みである。

3. BL-774 のソフトウェア・システム

実験計測用プログラムの作成は、その時代のプログラム開発の流行り廃りを色濃く反映するものであり、近年では、プログラムをスクラッチで一から全て作り上げる必要は少なくなっている。その背景には、Docker[16]など、仮想環境において既存のソフトウェアを配置、実行する仕組みの普及が挙げられる。Docker については、この章の「BL-774 とエコシステム」の中で改めて触れることにする。また、数値演算ライブラリでも自らスクラッチで用意するより Web サイトからインストールして利用できることが多い。最近では、コードの生成をサポートする生成 AI[17]が、その急速な進化に伴いホットな話題となっている。このため、例えば実験計測用の GUI アプリケーションの作成でも、様々な GUI のフレームワークが既に利用でき、相対的に比重が大きくなるのは、第 1 章でも述べたように、各ビームラインに特有のソフトウェアを使って実現したい目的に対して、ロジックとなる部分のプログラム化である。

このように、ソフトウェアの取扱いがより柔軟なものに進化していることを踏まえ、BL-774 は、プログラミング全般にわたる学習コストを抑制した上で、ロジックに関わる部分のプログラム作成に注力できる仕組みを構築してきた。

774 Basic System の実装に使用しているプログラミング言語は基本的に Python であり、現在の典型的な動作環境は Linux (Ubuntu)[18]と Python (>=3.8)である。Python 本体のバージョンは継続的にアップデートされるので、BL-774 のソフトウェア・アップデートも数年ごとにこれを追随している。この際、過去に対応していた機器は継続して利用できるように構築している。

774 Basic System の構成要素については、文献[3, 4]等があり、以下ではごく簡単に述べる。

対象機器と Communication service

対象とする被制御機器は、ビームラインで広く使われる、パルスモーター・コントローラーに代表される、イーサネット・インターフェースを有するものであり、アクセス頻度が 10 Hz 程度の機器である。

774 Basic System において被制御機器との接続・通信を仲立ちするプロセスとして「Communication service」と呼ばれるものがある。これは機器（通信インターフェース）ごとに 1 プロセスを起動させる。複数のプロセスや PC からの通信の制御や制限を行う機能も実装される。

774 API

774 API は、Python で記述された 774 Basic System のクラス・ライブラリである。Ecpy（Experiment Control API）はその構成要素の一つで、ロジックの開発に用いるための API 環境を提供する。Ecpy は、ソフトウェア動作のステップごとに整理された階層構造を持ち、それらには「Device layer」や「Object layer」と呼ばれるものがある。また、ecpy のメソッドは、非同期動作の簡潔な記述として、asyncio[19]の実装形式であるコルーチン関数として記述されている。

パラメーターの扱いと 774 ConfigDB

ConfigDB は、システムにおけるパラメーター設定ファイルの氾濫を避け、パラメーターの一元管理を図

り、登録・閲覧・変更の容易なコンフィグレーション・システムを提供する。これにはビームラインの担当者やユーザーが扱うパラメーターも含まれる。このデータベースには MariaDB[20]を使用し、入出力には専用の Python API (cfpy) が作成されている。データの閲覧・編集には専用の Web フォームを備える。

ユーザー・インターフェースと 774 Service

774 API は、774 Basic System の環境が構築された PC 上の Python 開発インターフェースにおいて ecpy をインポートすることで直接的に利用できる。例えば、JupyterLab や統合開発環境上でのアプリケーション開発が該当する。これに加えて、774 API を実行する利便性を補完するために、Web GUI の形態によるインターフェースを一定の範囲で提供している。機器の基本的な操作や、システムに必要な設定や内部プロセスの立ち上げ等は基本的に Web GUI から行うことができる。これらのアプリケーション群は、必要に応じて選択的にアップデートや水平展開をすることができるように、それぞれが独立性に富んだマイクロサービスの形態で構築されている。その他、ConfigDB に対する Web インターフェース（図 2）や、774 Basic System に対するリモート操作のためのプロトコル等も用意している。

BL-774 とエコシステム

774 Basic System のビームライン等の利用環境への水平展開を考える時、スケーラビリティや、各ビームラインに存在している多様な OS やそれに基づくアプリケーション群との両立に関する課題がある。ここで、「スケーラビリティ（scalability）」も、さまざまな文脈で使用される言葉であろう。IT やソフトウェア・システムにおいて用いられる場合には、「運用上のある次元での増大に対応するソフトウェア・システムの能力」[21]として定義できる。ここでの運用上の次元とは、「システムが同時に処理できるユーザーや外部からのリクエストの数」や「システムが有効に処理と管理ができるデータの量」[21]などが挙げられる。そこで、774 Basic System においても、利用環境におけるソフトウェア・システムのインストールやアップデート等のソフトウェア・デプロイメントに関わる作業を再現性よく行うため、Docker と呼ばれるコンテナ型仮想化のためのオープン・プラットフォームを利用している。「コンテナ（container）」は、本来は物流輸送で用いられる容器を意味する言葉であるが、IT やソフトウェア・システムの分野においては、そのメタファーで、「アプリケーションとその依存関係をカプセル化したもの」[22]を表す用語として使われている。このコンテナに基づく仮想化技術がコンテナ型仮想化と呼ばれるものであり、この他に、いわゆる仮想マシン（Virtual Machine、VM）に基づく仮想化技術が知られている。この 2 つのアプローチは本質的に異なる特徴を持ち、「コンテナの目的はアプリケーションを移植可能で自己完結型にすること」であり、「VM

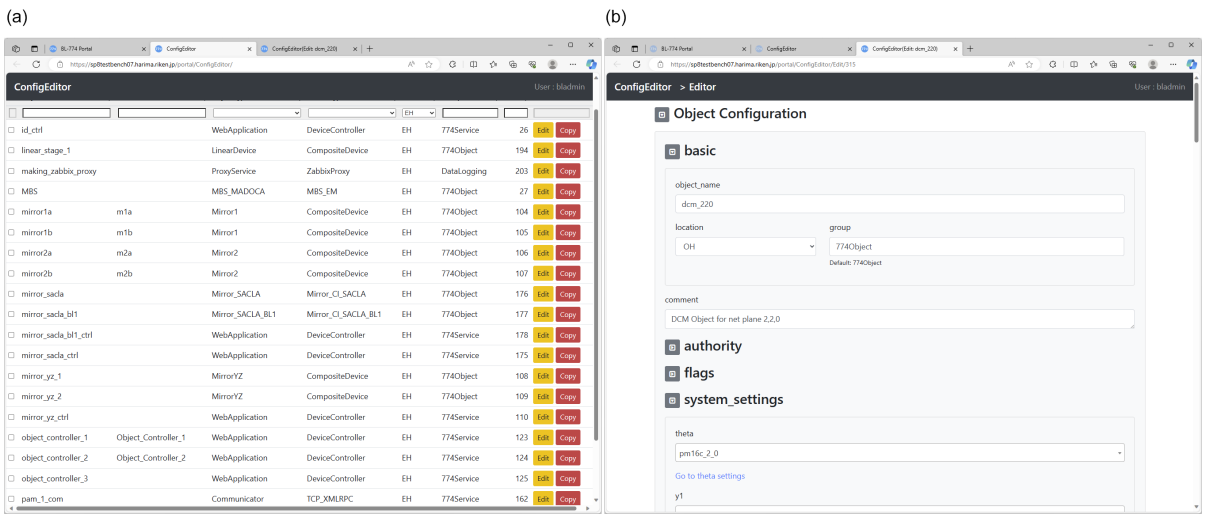

図 2　ConfigDB の Web インターフェース。(a)オブジェクトの一覧画面と、(b)編集画面の例。

の目的は外部環境を完全にエミュレートすること」[22]と捉えることができる。さらに、コンテナは、ホストOSとリソースを共有して使用することから効率が大幅に向上し、起動や停止に要する時間も短縮されている。このことから、ユーザーは、コンフィグレーションやインストールの問題に時間を費やしたり、システムに必要な変更を心配したりすることなく、複雑なアプリケーションをダウンロードして実行できるし、開発者は、ユーザー環境の違いや依存関係の可用性について心配する必要がなくなる、という利点[22]がある。「依存関係の可用性（availability of dependencies）」とは、ここでは、そのアプリケーションが依存関係を持つコンポーネントがどの程度、安定して持続的に動作するかという度合いを示すものと解釈できる。そして、Docker[16]は、同名の企業によって開発、公開されている、コンテナ型仮想化環境を提供するオープン・ソース・ソフトウェアである。Dockerの仕組みでは、Dockerコンテナ・イメージ[16]と呼ばれる、「コード、ランタイム、システム・ツール、システム・ライブラリ、設定など、アプリケーションの実行に必要なすべてのものを含む、軽量でスタンドアロンの実行可能なソフトウェア・パッケージ」があり、これがDockerエンジン[16]と呼ばれるコンポーネントで実行されることにより、Dockerコンテナとして機能することになる。また、Dockerコンテナを生成、実行するために、Docker Compose[23]と呼ばれる、ツール、コマンド体系もDockerから提供されている。774 Basic Systemでは、Dockerコンテナの持つ利点を活かし、774 Basic Systemのサービスや、事前にカスタマイズした環境を供給する手段としてコンテナを利用していて、例えば、「774 Software Development Environment（774 SDE）」や、「774 Portable system（774 Portable）」と呼ばれるコンテナ環境を用意している。

4. BL-774のサンプル・コード

リスト1に、BL-774のecpyを用いた実験計測用プログラムのサンプル・コードを掲載する。ここでは紙面の都合上、デモ的なコードにとどめるが、実践的なサンプル・コードについては順次、BL-774のWebサイトで公開していく予定である。また、リスト1のサンプル・コードを実行した時に得られる出力は、リスト2に示すようなものになる。

リスト1の特徴は、BL-774が推奨しているasyncio[19]形式のコーディングになっていることである。Ecpyは、機器制御の並行処理[24]を扱うコードの実装に、非同期I/Oを実現するPythonライブラリasyncioを利用している。ここで並行処理とは、プロセッサが1つだけであるか複数であるかによらず、計算資源全体として効率的な処理の実現を図る技法である。I/Oは、ファイルの読み書き、データベースへの接続、ネットワーク通信などのシステムの入出力を

```python
import asyncio
import ecpy

# DCM and motor scan with signal counting
d = ecpy.DCM("dcm") # Make DCM instance
p = ecpy.PulseMotor("test_motor1") # Make motor instance
c = ecpy.Counter("test_counter1") # Make counter instance

photon_energies = [10.0, 10.1, 10.25, 10.35, 10.45, 10.5, 10.6]
positions = [-0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6]

async def scan():
    for ph, pos in zip(photon_energies, positions):
        # Many methods have available co-routines with prefix "cr_"
        # Move concurrently
        await asyncio.gather(d.cr_set_photon_energy(ph), p.cr_move(pos, unit="mm"))
        # Count in 0.5 second
        data = c.count("timer", time=0.5)['count'][0]
        print(f"Energy:{ph}keV  Position:{pos}mm  Count:{data}")

asyncio.run(scan())
```

リスト1　BL-774のecpyを用いたサンプル・コード。

```
Energy:10.0 keV   Position:-0.6 mm  Count:10
Energy:10.1 keV   Position:-0.4 mm  Count:50
Energy:10.25 keV  Position:-0.2 mm  Count:150
Energy:10.35 keV  Position: 0.0 mm  Count:300
Energy:10.0 keV   Position: 0.2 mm  Count:450
Energy:10.5 keV   Position: 0.4 mm  Count:500
Energy:10.6 keV   Position: 0.6 mm  Count:500
```

リスト２　リスト１の実行結果の例。

意味する。ビームラインの機器制御の場合では、例えばフライ・スキャンにおいて可動軸の駆動中に他の計測機器へのアクセスを行う際などに、I/O が全体の処理時間を支配する状況が発生しうる。そこで非同期 I/O では、1 つの処理フローの中で入出力の処理を開始した後に、その終了を待ち続けるだけでなく、待ち時間中に別の処理の実行を進める。一般に、非同期 I/O を実現するためのコーディングは煩雑でありスキルが必要とされる。そのためライブラリや構文によって簡潔にコーディングするための方法がさまざまなプログラミング言語において用意されている。リスト 1 も、asyncio を用いたアプリケーションのコーディングで推奨されている async/await 構文を用いて書かれたものである。また、ここで呼び出している ecpy のメソッド自体も、非同期処理として実行できるように実装されている。

なお、リスト 1 で使用している ecpy の DCM クラスには、DCM を制御するための基本的な機能のみが実装されている。ビームラインによっては、さらに複合的な操作、例えば、DCM の $\Delta\theta_1$ をピエゾで制御し、光強度のスキャン測定の結果を用いてさらに微調整を加えることも行われている。この場合でも、774 Basic System のソフトウェア開発環境を用いれば、ecpy の DCM 用のクラスを継承した上で、ピーク・サーチの結果から $\Delta\theta_1$ の微調整を加えるロジック部分の記述に着手しやすい。このように、ecpy が直接サポートしていない機能であっても、ecpy の基本的な機能を利用しながら、そのビームライン固有のロジックを扱うプログラムの作成に注力できるならば、BL-774 のコンセプトの実現に近づいていると言うことができるだろう。

5. まとめ

BL-774 の基幹的なソフトウェア・システム 774 Basic System について紹介した。実験制御用プログラムの作成段階では、ロジックに関わる部分に注力できるよう、周辺のパーツがビームラインでの実験に適切な形、機能、規模で実装されていて、利用形態に沿って選択可能な複数のマイクロサービスとして提供されていることを紹介した。

本稿では記載しきれなかったが、BL-774 では、ビームライン制御以外にも、データ収集などにおいてビームラインに関わる他のテーマにも取り組んでいる。また、BL-774 は、2 次元検出器システムや SPring-8 データセンターなど、現在 SPring-8/SACLA で開発中の他のシステムとシームレスに統合する予定にもなっている。BL-774 を施設内の他のインフラと併せて導入することで、オンライン・データ解析に基づくフィードバック運用やビームライン外からの遠隔運用など、より高度な実験運用に対する期待が高まるであろう。今後の BL-774 の発展にぜひご期待いただきたいとともに、関心を持たれた方は 774 Basic System のコンテナ・イメージをダウンロードし触れてみていただきたい。

なお、本稿はソフトウェア・システムやプログラム開発に関する内容の都合上、いくつかの IT 用語を用いた。編集委員会の意向も踏まえて、本文中で説明を試みた箇所もあるが、十分ではないかもしれない。そこでも触れたように、これらの IT 用語は、既存の表現のメタファーであったり、文脈に応じて幅のある定義がなされたりする場合がある。また、技術の変遷とともに意味づけが変化していくこともある。気になられた用語については、本稿の記述にとらわれることなく、各種文献やインターネット等を通して理解を深められることを希望する。最近、話題の生成 AI に聞いてみることも面白いかもしれない。もし、本稿に関心をもたれた方が、ビームライン制御への興味を深めていただくきっかけになることができれば望外の喜びである。

謝辞

BL-774 の開発と導入にあたり、SPring-8/SACLA のビームライン関係者、理化学研究所のエンジニアリング・チームメンバーには多くのご助言、ご協力をいただきました。厚く御礼申し上げます。

参考文献

[1] H. Tanaka: *Synchrotron Radiat. News* **27** (2014) 23-26.
[2] T. Watanabe and H. Tanaka: *Synchrotron Radiat. News* **36** (2023) 3-6.
[3] K. Nakajima *et al.*: *J. Phys.: Conf. Ser.* **2380** (2022) 012101.
[4] K. Motomura *et al.*: *Synchrotron Radiat. News*, Published online: 13 Dec 2023, doi: 10.1080/08940886.2023.2277638.
[5] 774portable (https://bl774-repo01.usr.common.sp8int.jp/pub/774portable、2023 年 11 月 22 日閲覧、SPring-8 施設内での公開).
[6] Python (https://www.python.org、2023 年 11 月 22 日閲覧).
[7] Jupyter (https://jupyter.org、2023 年 11 月 22 日閲覧).
[8] 大橋治彦、平野馨一編: 改訂版・放射光ビームライン光学技術入門, 日本放射光学会 (2019).
[9] M. Reddy: API Design for C++, Morgan Kaufmann (2011).
[10] 上杉健太郎、星野真人: SPring-8/SACLA 利用者情報 **26** (2021) 448-449.
[11] A. Yasui *et al.*: *J. Synchrotron Rad.* **30** (2023) 1013-1022.
[12] 隅谷 和嗣 他: SPring-8/SACLA 利用者情報 **27** (2022) 274-279.
[13] 安野聡、ソ オッキュン、高木康多、保井晃: SPring-8/SACLA 利用者情報 **28** (2023) 434-438.
[14] 大端通: SPring-8/SACLA 利用者情報 **2** (1997) 15-17.
[15] 古川行人、松本崇博、石井美保: SPring-8/SACLA 利用者情報 **19** (2014) 392-395.
[16] Docker (https://www.docker.com、2023 年 11 月 22 日閲覧).
[17] ChatGPT (https://chat.openai.com、2023 年 11 月 22 日閲覧).
[18] Ubuntu (https://ubuntu.com、2023 年 11 月 22 日閲覧).
[19]asyncio (https://docs.python.org/ja/3/library/asyncio.html、2023 年 11 月 22 日閲覧).
[20] MariaDB (https://mariadb.org、2023 年 11 月 22 日閲覧).
[21] I. Gorton: Foundations of Scalable Systems, O'Reilly Media, Inc. (2022).
[22] A. Mouat: Using Docker, O'Reilly Media, Inc. (2016).
[23] Docker Compose (https://docs.docker.com/compose/、2023 年 12 月 30 日閲覧).
[24] 例えば、B. Barney: Introduction to Parallel Computing Tutorial, Lawrence Livermore National Laboratory (https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial、2023 年 11 月 22 日閲覧).

本村　幸治　MOTOMURA Koji
(国) 理化学研究所　放射光科学研究センター
〒679-5148　兵庫県佐用郡佐用町光都 1-1-1
TEL : 050-3502-6167
e-mail : motomura@spring8.or.jp

安田　伸広　YASUDA Nobuhiro
(公財) 高輝度光科学研究センター　ビームライン技術推進室
〒679-5198　兵庫県佐用郡佐用町光都 1-1-1
TEL : 0791-58-0831
e-mail : nyasuda@spring8.or.jp

今井　康彦　IMAI Yasuhiko
(公財) 高輝度光科学研究センター　回折・散乱推進室
〒679-5198　兵庫県佐用郡佐用町光都 1-1-1
(国) 理化学研究所　放射光科学研究センター
〒679-5148　兵庫県佐用郡佐用町光都 1-1-1
TEL : 0791-58-0833
e-mail : imai@spring8.or.jp

中嶋　享　NAKAJIMA Kyo
(公財) 高輝度光科学研究センター　XFEL 利用研究推進室
〒679-5198　兵庫県佐用郡佐用町光都 1-1-1
(国) 理化学研究所　放射光科学研究センター
〒679-5148　兵庫県佐用郡佐用町光都 1-1-1
TEL : 0791-58-0992
e-mail : kyo.nakajima@spring8.or.jp

城地　保昌　JOTI Yasumasa
(公財) 高輝度光科学研究センター　XFEL 利用研究推進室
〒679-5198　兵庫県佐用郡佐用町光都 1-1-1
(国) 理化学研究所　放射光科学研究センター
〒679-5148　兵庫県佐用郡佐用町光都 1-1-1
TEL : 0791-58-0992
e-mail : joti@spring8.or.jp