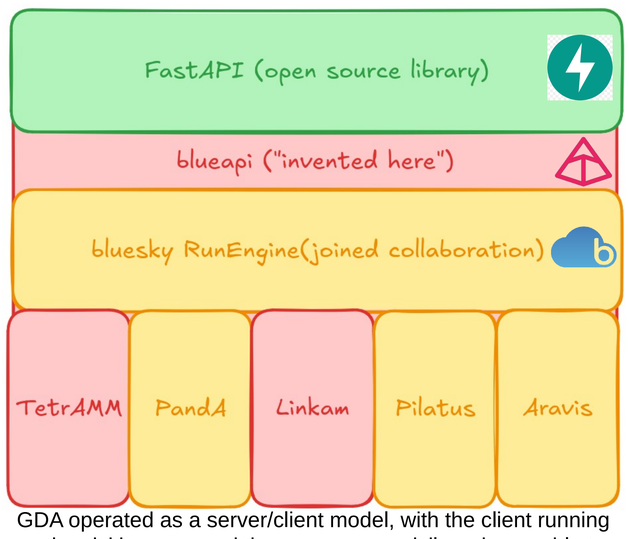
Not-invented-here:
Building an acquisition platform with off-the-shelf components

J. Ware, C.Forrester, A. Emery, R. Syrett, T. Cobb, Diamond Light Source, Harwell Innovation Campus, Didcot, OX11 0DE

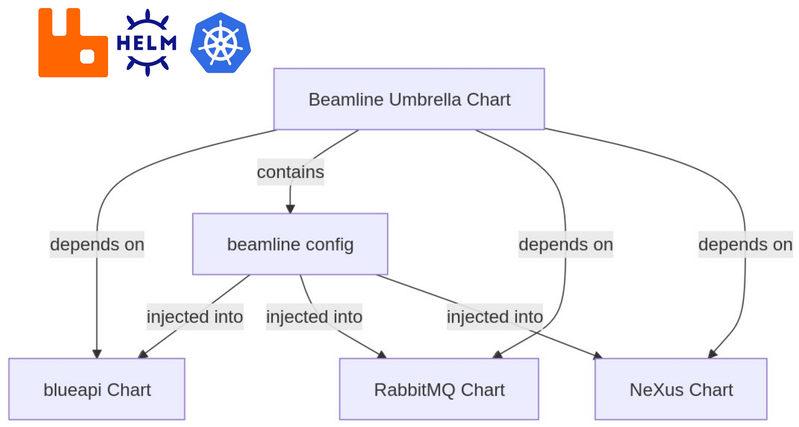
Diamond Light Source has deprecated the GDA (Generic Data Acquisition) platform and is replacing it with Athena, a service-based acquisition platform centred on the bluesky data collection framework from NSLS-II. Historically Diamond has struggled to overcome a mindset that every aspect of the operation of a Synchrotron is equally and inextricably unique. This poster seeks to dispel that myth and describes experiments that have occurred and are ongoing which make use of pre-existing free open-source software, with effort spent on altering Diamond requirements or contributing to the existing: rather than "reinventing the wheel".

Time-Resolved Scanning

In November 2023, an initial experiment was performed with a minimal Athena stack, with the GDA Jython console operating as as REST client for a Blueapi instance installed on a Kubernetes cluster located at the beamline. It sought to replace the use of an end-of-life TFG2 Time Frame Generator and enable time-based scanning with varying duration of points: a feature GDA was previously unable to achieve.



GDA operated as a server/client model, with the client running as the visiting user and the server as a privileged user able to write to all visits on the shared filesystem- however Bluesky is typically directed from the command line as the visiting user. To allow a continuity of service blueapi: which combines a bluesky RunEngine; a REST API; and necessary context mapping between them was produced. Blueapi is a shim or glue adapting device serialisation and ensuring state.



The Athena stack was deployed with Helm as an Umbrella chart with components as dependencies: blueapi, the NeXus service and a RabbitMQ instance. This enabled deploying the stack as a coherent block of functionality, and benefit from Kubernetes service routing, pod auto-restarting, liveness probes and more.

Moving GDA messaging into Kubernetes was an existing desire to address observed reliability problems by making use of monitoring and service restarting, however no up-to-date supported ActiveMQ Helm deployment could be found. As GDA internal messaging was utilising JMS, RabbitMQ was deployed with a JMS plugin and changes to the GDA core libraries to enable switching between ActiveMQ and RabbitMQ were made.

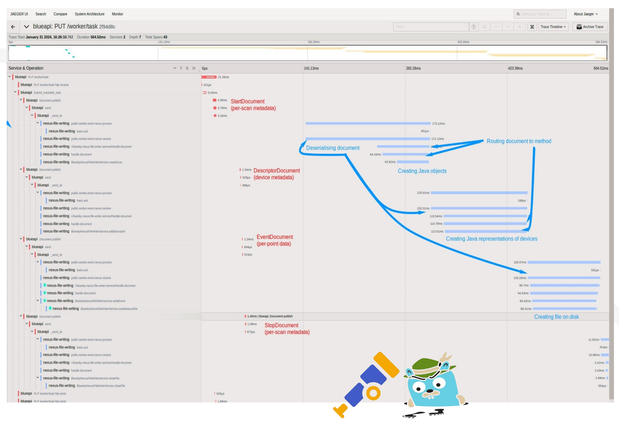
NeXus and Analysis

Diamond's existing data analysis pipelines rely on NeXus SWMR (single write, multiple read) to perform analysis of data as it is written to disk- this has been the only way analysis code was able to access data written by GDA.

To enable continuity at the beamline, and consistency between experiments run with the new stack and the existing stack, GDA's existing Java NeXus writing code was extracted from GDA and a shim layer added to translate the bluesky event-model data schema emitted by blueapi. This code was embedded in a Spring Boot server, which enabled the service to be easily containerised and deployed alongside blueapi without Athena depending on GDA.

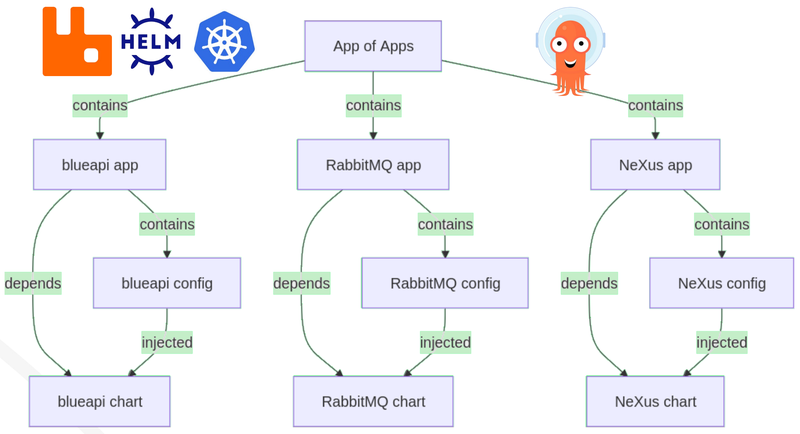
Stopflow Experiment

A 2nd experiment, initially due to occur June 2024 but delayed due to hardware problems to March 2025 will again make use again of time-based PandA hardware scanning, with an increased and improved Athena stack.



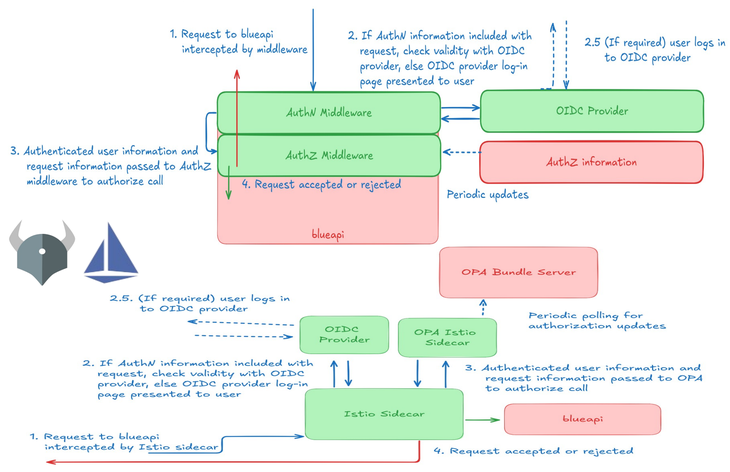
Because GDA was a monolithic application it was simple to track operations and debug locally. Transitioning to a service based architecture loses simplicity of debugging. Tracing calls through the system, in addition to logging within the services, enables introspection of the whole system for debugging and enables collection of metrics. Tracing is being implemented with Open Telemetry & Jaeger.

This example suggests places to look for efficiency gains: deserializing the StartDocument took much longer than similar operations for example. Gathering this data allows us to make justified decisions.



The Helm umbrella chart (see above) made assumptions about inter-requirements of the services, and made atomic changes and maintenance more awkward than intended. We have begun moving towards deploying ArgoCD as an app-of-apps pattern[1], with aims of having beamline state represented by 2 repositories: one containing the list of what services are deployed and enabled[2], and another representing the states of the deployed software[3].

[1]https://argo-cd.readthedocs.io/en/stable/operator-manual/cluster-bootstrapping/#app-of-apps-pattern
[2]example: https://github.com/epics-containers/p45-deployment
[3]example: https://github.com/epics-containers/p45-services



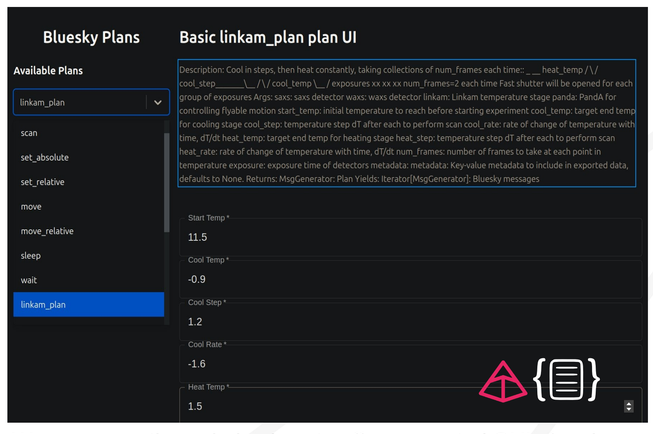
Authentication (knowing which user made a request) and Authorization (deciding whether the user is allowed to make a request) becomes even more important as data collection is no longer limited to on-site.

Athena has been developed with auth concerns from its conception, but delayed by infrastructure requirements. AuthN is anticipated to be in place by the end of the year, with the user logging in to blueapi through an OIDC flow, either as a command line tool or through a web front end, with authorization provided by Open Policy Agent (OPA) or a similar authorization provider afterwards.

A pair of example flows which are being considered are shown, making use of either FastAPI middleware or the Istio Service Mesh.

Towards Athena 1.0

GDA has built up functionality and expectations over more than 20 years. As Athena is rolled out more widely and takes over a larger amount of beamline activity, we hope to continue benefiting from using existing tooling



We are currently exploring how to offer generated GUIs for plans exposed by blueapi and to contribute resources to the bluesky-webclient library. The below example extracts the docstring from a python plan and provides appropriate fields for the type using JSONForms. While beamlines that operate a standard experimental procedure may wish to specialise their UI components, Athena will provide at minimum a viable generic UI and is increasingly aiming to enable rich UI generation by making use of annotations and Pydantic data validation when optionally included.

Data Access and post-processing NeXus

While the existing NeXus service provided a level of continuity with the existing pipelines and expectations of beamline users, the NeXus libraries are currently an isolated island of Java in a wider ecosystem of Python and Rust: the former being easy for users to make contributions of code for and the latter offering appealing benefits for code quality and performance for software engineers.

Diamond will begin storing the output data of the RunEngine, offering them up via a Data API: likely Tiled from the bluesky collaboration, removing the need for live analysis & SWMR mode NeXus files, and allowing for writing the output file as a final step of post processing.

The DataAPI and NeXus writer are opportunities for further collaboration.

Widening adoption

With N-dimensional trajectory fly-scans enabled by PandAs nearing readiness in ophyd-async, Athena is approaching the scanning capabilities of GDA. The array of device classes in ophyd-async and in dodal (the Diamond Ophyd Device Abstraction Layer) and bluesky plans in repositories for beamlines and techniques will continue to grow. As these numbers grow, we must also take responsibility to unify devices and plans where it makes sense to, and to allow developer time to standardise and prevent repetition.

For more information please visit www.diamond.ac.uk or contact Joseph Ware at joseph.ware@diamond.ac.uk

diamond