Streaming Data Architecture for ESS benchmarked on virtual AMOR

Dominik Werder [(*)], M. Könnecke, M. Brambilla, Paul Scherrer Institut, Villigen, Switzerland
M. Jones, Tessella, Abingdon, United Kingdom
T. Richter, ESS ERIC, Lund, Sweden, A. Mukai, J. Nilsson, ESS ERIC, Copenhagen, Denmark
M. J. Clarke, F. A. Akeroyd, Science and Technology Facilities Council, Didcot, United Kingdom

Summary:

- Experimental data at ESS will be streamed via a unified messaging layer
- EPICS process variables are forwarded to the messaging layer
- NeXus-compliant HDF files are assembled from message streams

We present:

- The EPICS-to-Kafka Forwarder component
- The HDF File Writer component
- Performance measurements for these components

Data Streaming for ESS

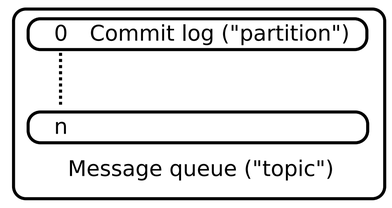
The European Spallation Source (ESS) will use a unified but flexible data aggregation and streaming architecture. The advantages include:

- Decoupling of the individual parts of the system
- Well defined common interfaces
- Component-oriented software design
- Improved scalability due to better separation of concerns

The messaging layer is built on top of:

- A unified message broker (Apache Kafka)
- A common message format (Google Flatbuffers)

Kafka provides "topics", which are named sets of persistent commit logs.



Partitions can live on different machines for scalability and be replicated for redundancy.

Flatbuffers is a:

Efficient serialization format, used for all messages

- Binary, no conversions needed on write
- Statically typed
- Flexible schema language
- Based on offset-pointers, read without parsing
- Verification logic auto-generated
- Supports C, C++, Python, JS, Java, C#, ...
- Designed with maintainability in mind

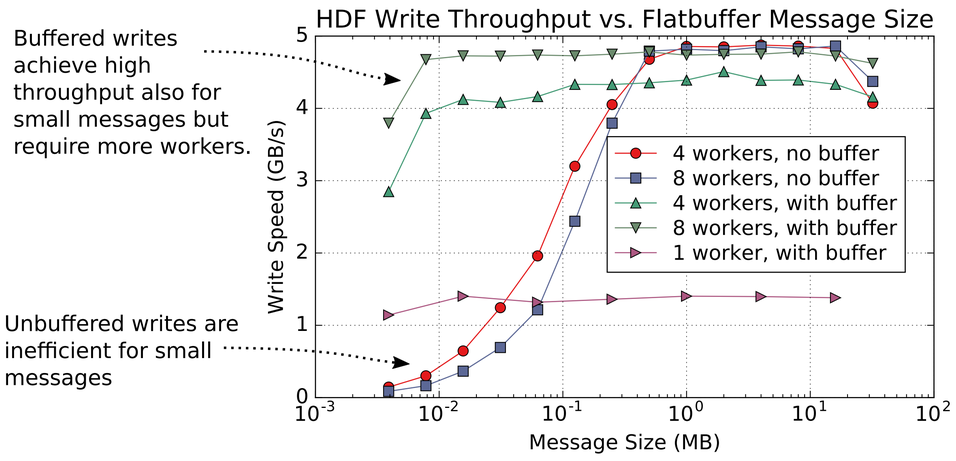
Common schemas at:
https://github.com/ess-dmsc/streaming-data-types

HDF File Writer https://github.com/ess-dmsc/kafka-to-nexus

The HDF File Writer is responsible for reading the data streams from the Kafka messaging layer and assembling them into NeXus-compliant HDF files.
File Writing is initiated and configured using JSON commands which are also distributed via the messaging layer. Typically, the Experiment Control Program initiates file writing on request of the user.
The main process of the file writer is responsible for command handling and can spawn multiple file writer tasks.
Writing of each data stream is handled either:

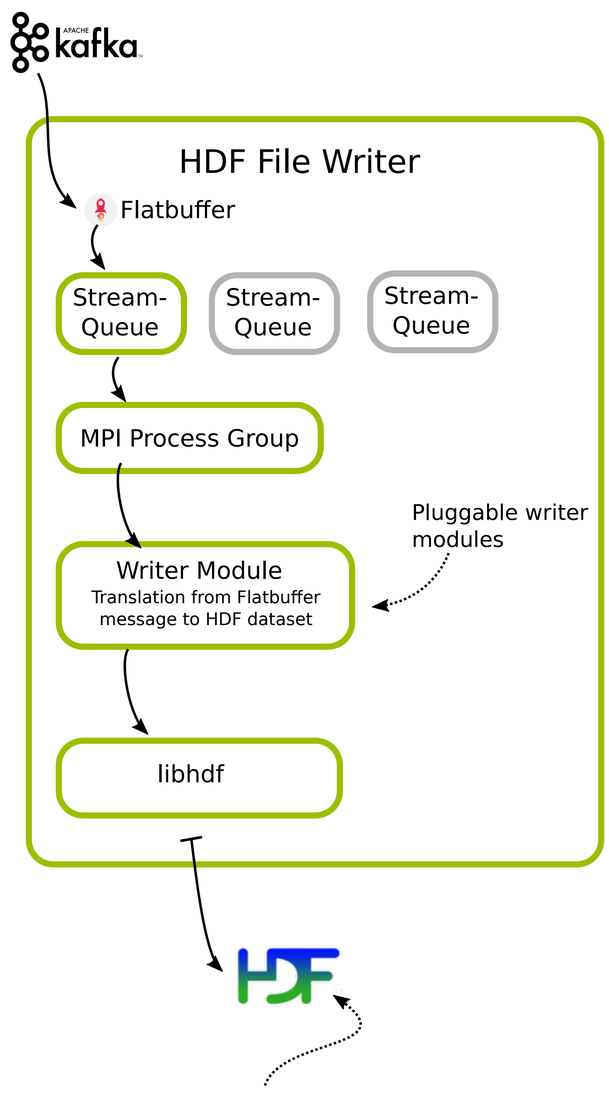
- On a common thread
- In its own thread(s)
- On a set of MPI worker processes

which can be configured as part of the command message.
Results presented here use MPI worker processes and Parallel HDF.

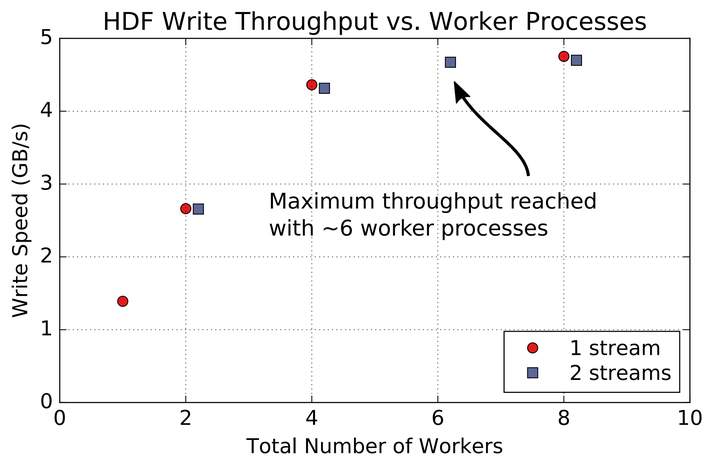


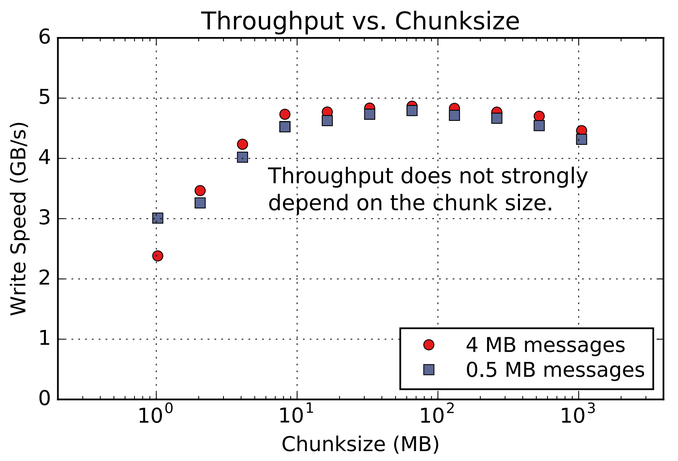
The Message Passing Interface (MPI) is used because the HDF library does not support multiple threads. Instead, its variant Parallel HDF utilizes MPI-I/O for parallel file system access.
The HDF File Writer starts as only one process and spawns the requested number of MPI processes at runtime.

Data Flow:



HDF structure is defined by the command which initiates file writing. The command is typically issued by the Experiment Control Program.





Benchmarks done on:
Xeon E5-2690v4
14 Cores @ 2.60 GHz
35 MB Cache
256 GB RAM
GPFS via 4x Infiniband FDR

EPICS Forwarder

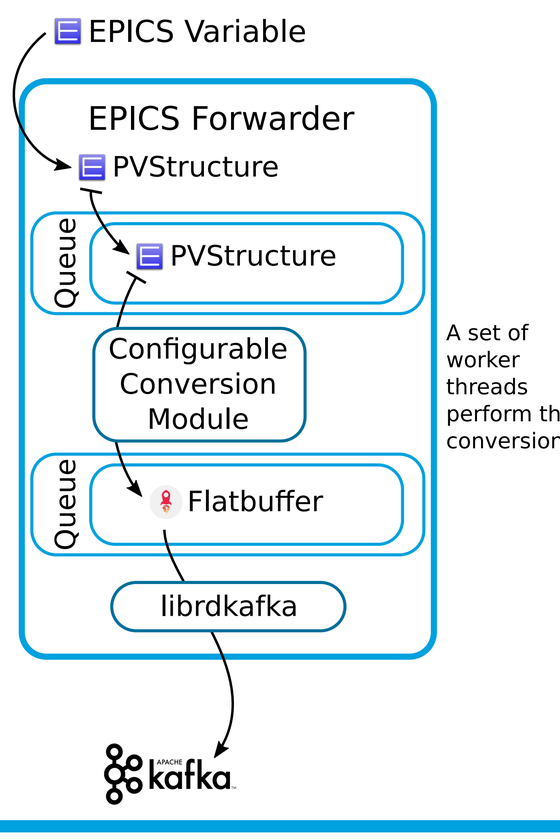
The Experimental Physics and Industrial Control System (EPICS) is used at many scientific facilities around the world, including particle accelerators and telescopes.
At ESS, typical data from EPICS includes the sample environment, choppers and motion control.
To integrate EPICS data sources into the unified streaming architecture, we develop the EPICS Forwarder component.

A set of EPICS variables is monitored by the EPICS Forwarder. Each update is fed through the configured conversion module which is responsible for the conversion to a Flatbuffer message. The resulting messages are then published on the common Kafka messaging layer.

The EPICS Forwarder can be configured and controlled with JSON commands over the Kafka command topics.

https://github.com/ess-dmsc/forward-epics-to-kafka

Data Flow



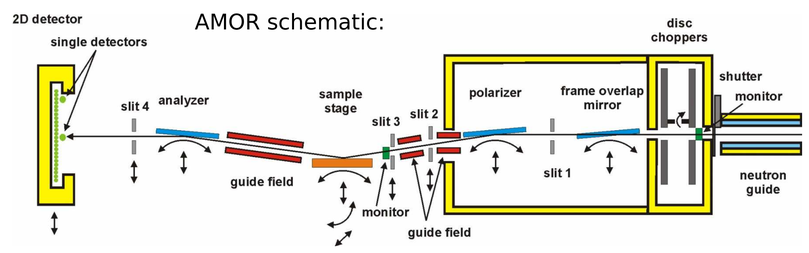
AMORSIM - virtual AMOR

To guide our development before ESS receives operational equipment we use a simulation of the AMOR instrument located at PSI.
The simulation contains

- Neutron event generator (histogram → events)
- Dornier chopper with EPICS facade and TDC events
- Motor controller EL734
- Magnets

Devices are implemented in Python using Twisted. The neutron event generator is written in C++ for higher performance.

This platform will also serve as a test environment during the development of the Experiment Control Program.



[(*)] dominik.werder@psi.ch

We would like to thank Heiner Billich for the possibility to test on one of the compute nodes of the Swiss Light Source at PSI.

ICALEPCS 2017, 7th-13th October, Barcelona, contribution THPHA167.

This work is part of the BrightnESS work package 5.3 and funded by the European Union Framework Programme for Research and Innovation Horizon 2020, under grantagreement 676548.