Using Message Broker with EPICS: SPX Controls Use Cases

Siniša Veseli
Software Engineer
AES / Software Services Group

EPICS Collaboration Meeting
October 5, 2013

Outline

- About SPX
- Why Use Message Broker?
- SPX Controls Software
- Performance Measurements
- Summary

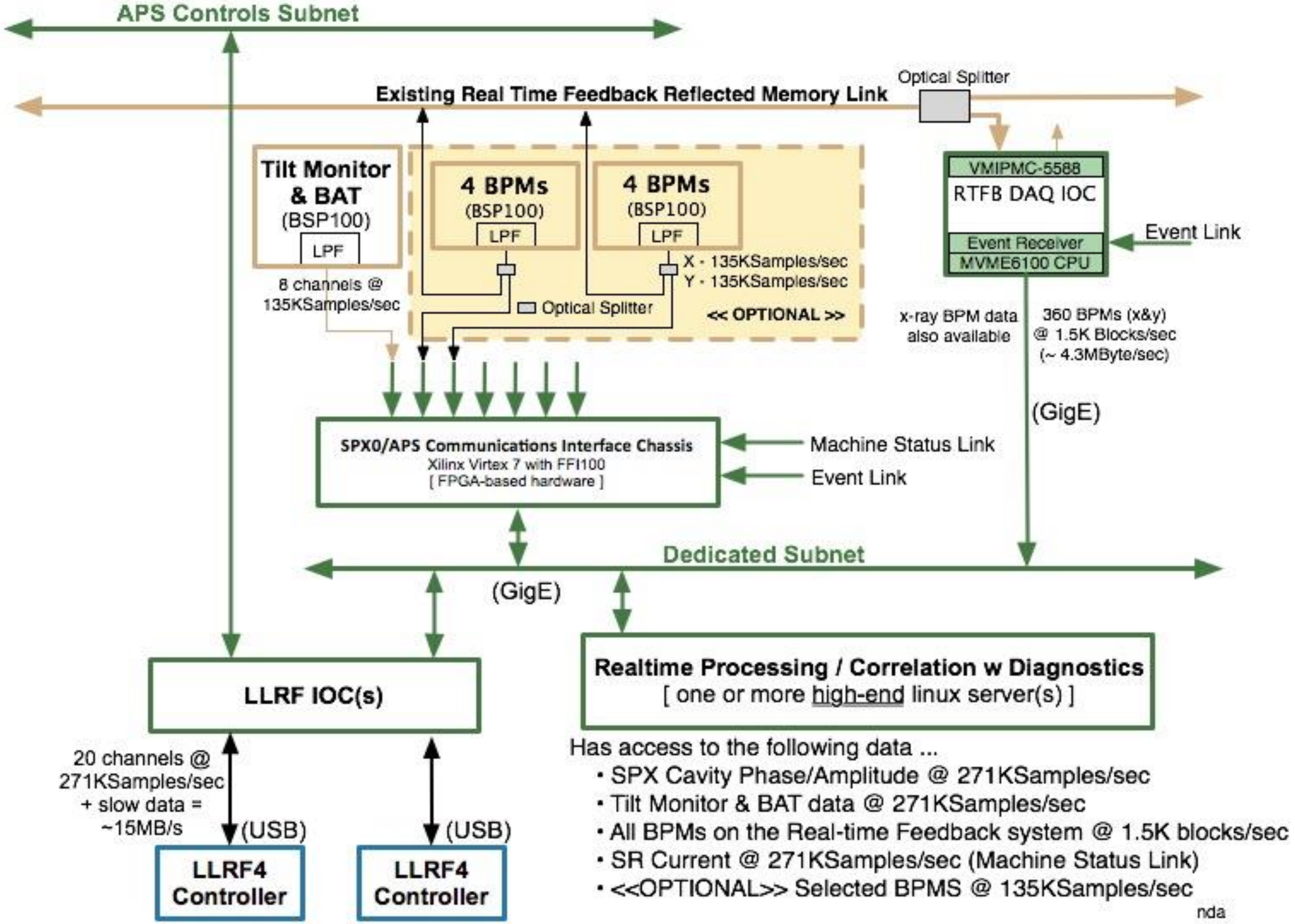
Motivation

- EPICS/Message Broker Integration
- Performance Testing Results Involving EPICSv4

About SPX

- SPX: Short-Pulse X-ray project
- Originally one of the major goals of the APS Upgrade (APS-U)
  - Addressed the need for intense, tunable, high-repetition rate, picosecond x-ray pulses
  - Ultimate goal: deliver short (2ps) x-ray pulses at 6.5 MHz
- Technically most complex part of the APS-U
  - 2 cryomodules, each with 4 superconducting rf deflecting cavities operating at 2815 MHz
  - Must keep at minimum disturbance of the storage ring during user operation
  - SPX0 Systems: 2 cavity cryomodule, used for testing
- Not compatible with the recent APS-U direction (evaluating incorporation of the Multi-bend Achromat Lattice)

SPX0 Diagnostic Data Acquisition Option



APS Controls Subnet

Existing Real Time Feedback Reflected Memory Link

Optical Splitter

Tilt Monitor & BAT (BSP100) LPF

8 channels @ 135KSamples/sec

4 BPMs (BSP100) LPF

4 BPMs (BSP100) LPF

X - 135KSamples/sec
Y - 135KSamples/sec

Optical Splitter

<< OPTIONAL >>

VMIPMC-5588
RTFB DAQ IOC
Event Receiver
MVME6100 CPU

Event Link

x-ray BPM data also available

360 BPMs (x&y) @ 1.5K Blocks/sec (~ 4.3MByte/sec)

(GigE)

SPX0/APS Communications Interface Chassis
Xilinx Virtex 7 with FFI100
[FPGA-based hardware]

Machine Status Link

Event Link

Dedicated Subnet

(GigE)

LLRF IOC(s)

Realtime Processing / Correlation w Diagnostics
[one or more high-end linux server(s)]

20 channels @ 271KSamples/sec + slow data = ~15MB/s

(USB)

(USB)

LLRF4 Controller

LLRF4 Controller

Has access to the following data ...

- SPX Cavity Phase/Amplitude @ 271KSamples/sec
- Tilt Monitor & BAT data @ 271KSamples/sec
- All BPMs on the Real-time Feedback system @ 1.5K blocks/sec
- SR Current @ 271KSamples/sec (Machine Status Link)
- <<OPTIONAL>> Selected BPMS @ 135KSamples/sec

nda
9/10/13

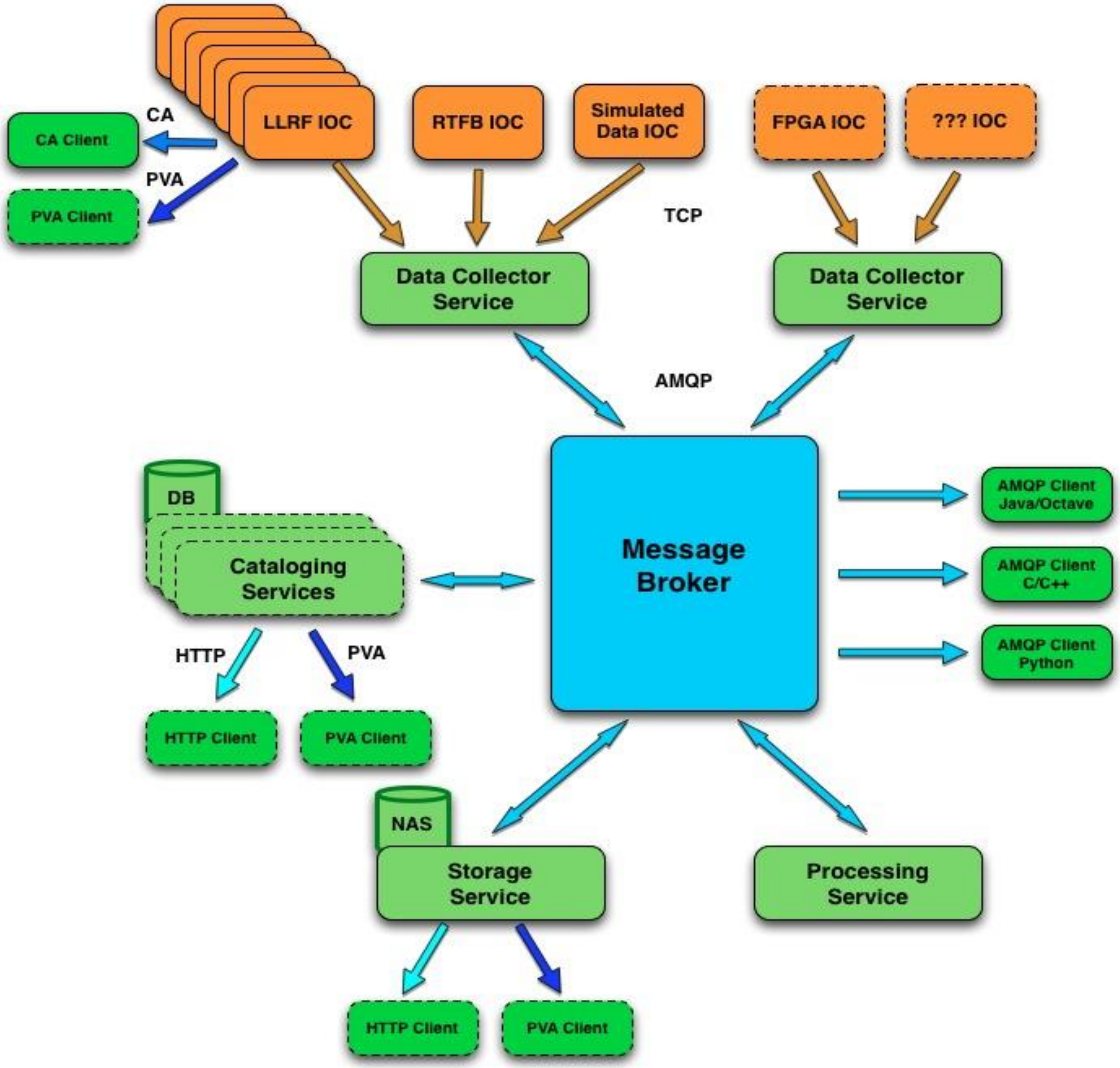
SPX Controls Use Cases

- Keep up with LLRF Controllers (data rates of up to 15 MB/s per Controller)
- Access to complex data structures
- Real-time access to monitoring and diagnostics data to multiple users/tools simultaneously
- Ability to access real-time data using Matlab/Octave
- Data storage services
- Cataloging services
- Fast logging system

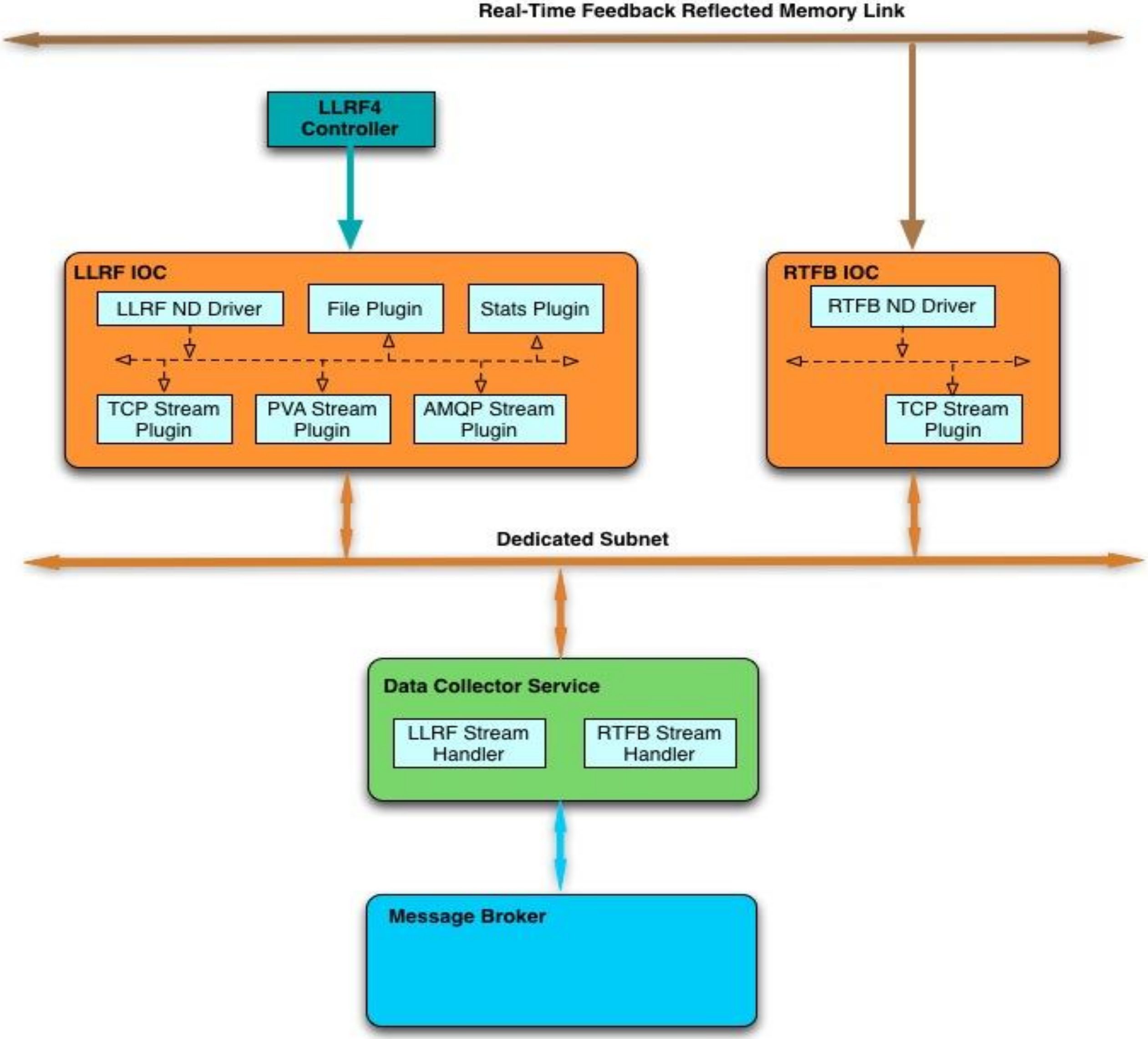
Why Message Broker?

- Advanced Message Queuing Protocol (AMQP) supports wide variety of communications patterns and is frequently used in enterprise applications:
  - Real-time feed or constantly updating data
  - Advanced publish-and-subscribe
- Number of freely available AMQP broker/client implementations
- **Can we leverage some of the available AMQP tools for EPICS applications, not as a replacement for CA/PVA, but alongside those?**

SPX Controls Software Architecture

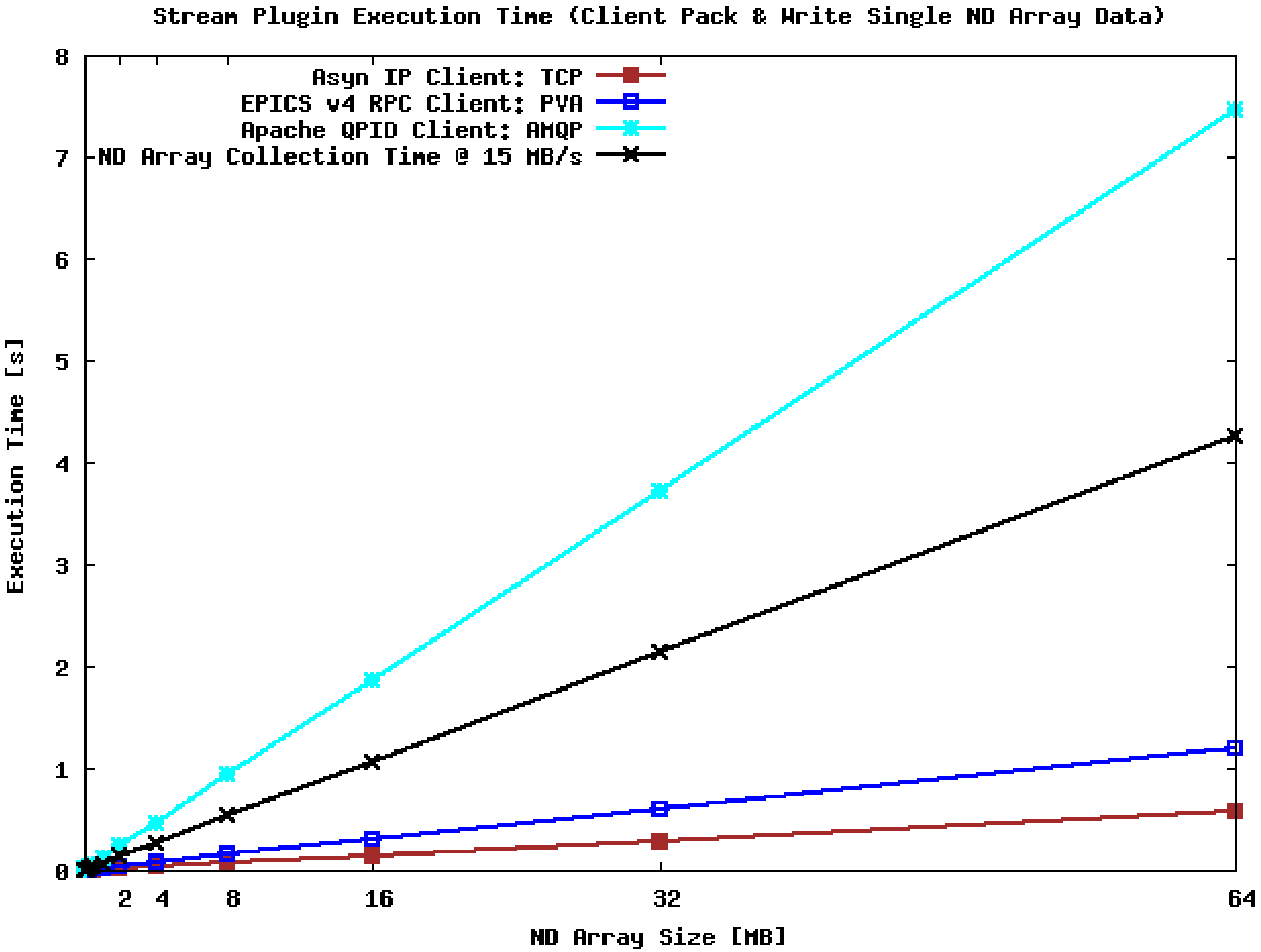


SPXRF Area Detector Framework Usage



Plugin Performance: Testing

- LLRF4 Driver (SPX0) collects data in 32 KB “chapters” (16 I/Q waveforms with 512 integers)
- LLRF “data burst” size is determined by couple of EPICS PVs:
  - Number of chapters to collect in a single ND array
  - Number of ND arrays to collect and stream
- LLRF data bursts are associated with numerous ND Attributes (sent separately from actual ND Array data)
- LLRF IOC has 3 streaming plugins:
  - TCP (uses asyn v4.18 IP port driver, about 3.1K lines of support code)
  - PVA (uses EPICS v4.3.0 RPC client, about 2.1K lines of support code)
  - AMQP (Apache QPID v0.20, about 1.7K lines of support code)
- Client-side performance was measured in terms of time required to pack and send one ND array data to a service running on a remote host over a gigabit network
- Measured times do not include service processing time, but in case of PVA they include empty RPC response (less than 2 ms)
- Client machine: i7-3770@3.4GHz, 8GB RAM, 4 cores/8 threads, 1Gbit NIC


Stream Plugin Execution Time (Client Pack & Write Single ND Array Data)
Asyn IP Client: TCP
EPICS v4 RPC Client: PVA
Apache QPID Client: AMQP
ND Array Collection Time @ 15 MB/s
Execution Time [s]
0
1
2
3
4
5
6
7
8
2
4
8
16
32
64
ND Array Size [MB]

Plugin Performance: Results

- Software can easily keep up with nominal data rates
- One second's worth of LLRF ND Array data is processed in about:
  - TCP Stream Plugin: 0.15 seconds
  - PVA Stream Plugin: 0.30 seconds
  - AMQP Stream Plugin: 1.85 seconds (would require 2 threads to keep up)
- PVA plugin performance is a factor of 6 better than AMQP plugin for streaming arrays (monomorphic data): QPID v0.20 C++ APIs have no support for AMQP arrays and require sending array elements via lists (very inefficient)
- Comparable PVA/AMQP plugin performance for ND attributes (polymorphic data)
- Preparing/sending initial stream message with about 200 LLRF ND Attributes (approximately 16KB of structured data):
  - TCP Stream Plugin: prepare/send message in under 0.5 milliseconds
  - PVA Stream Plugin: 4-5 milliseconds to pack, 4-5 milliseconds to send; initial call to RPC service takes 100-200 milliseconds
  - AMQP Stream Plugin: 3-4 milliseconds to pack, 4-5 milliseconds to send

Message Broker Approach: Lessons Learned

- Our Broker Choice: Apache QPID
  - Open source, supports AMQP v1.0 and several earlier protocol versions
  - Platform Support: Linux, OS X, JVM
  - Extensive set of management tools and easy to use APIs
  - Client Support: C/C++, Java, Python, Perl, PHP…
  - Extensive documentation
  - Excellent support for maps/dictionaries
  - Extremely flexible and configurable
  - Works “out of the box”
  - Active user community, large user base
- QPID-related Issues:
  - Inadequate API support results in subpar performance with arrays
  - No client support for VxWorks
- General issues:
  - Not all brokers support AMQP v1.0, which is not compatible with earlier protocol versions

Summary

- One can successfully integrate message-oriented middleware into EPICS-based systems alongside CA/PVA
- Main advantages of this approach:
  - Flexibility
  - Ability to leverage large number of freely available (open source) tools and frameworks
- AMQP is an open standard protocol that ensures interoperability between different implementations of messaging providers/clients
- Broker choice impacts performance, platform/language/feature support, ease of use, configuration options, etc.

Future Work

- Utilize SPXRF Controls software/techniques to enhance existing diagnostics and DAQ tools at APS
  - Deploy Real-time Feedback IOC and accompanying services to production

Additional Slides

SPX Controls Requirements

- The entire SPX system must be thoroughly integrated with the existing APS storage ring controls, timing, and diagnostics
- Provide remote monitoring and control to all SPX subsystems consistent with APS standards and existing OAG tools
  - Data must be stored in SDDS (Self-Describing Data Sets) format
- Provide the necessary interfaces between the SPX and other APS systems as required by the SPX needs (e.g., RTFB, MPS, Event System, etc.)
- Provide a real-time data processing environment for the SPX control algorithms to ensure they can be executed at the necessary rate
- Provide thorough diagnostic information and tools to assist in quick determination of SPX performance and post-mortem fault analysis (required for maintaining high availability)

Why Message Broker?

- Advanced Message Queuing Protocol (AMQP) supports wide variety of communications patterns and is frequently used in enterprise applications
- Typical use cases:
  - Real-time feed or constantly updating data
  - Point-to-point messaging
  - Advanced publish-and-subscribe
  - Delivering messages when destination comes online
  - Receiving constant status updates and sending large messages at the same time and over the same network connection
  - Transactional messaging
  - Communication between diverse programming languages/operating systems
  - Remote procedure call patterns
- Number of freely available AMQP broker/client implementations (QPID, ActiveMQ, RabbitMQ, SwiftMQ…)
- **Can we leverage some of the available AMQP tools for EPICS applications, not as a replacement for CA/PVA, but alongside those?**

Advanced Message Queuing Protocol

- Originated in 2003 (JP Morgan & Chase, London UK)
- Open standard, v1.0 became OASIS standard in 10/2012
- Wire-level protocol, mandates behavior of messaging providers and clients to assure interoperability between different implementations
- Few protocol details:
  - Basic unit of data: *frame*
  - Nine frame bodies used to initiate, control and tear down message transfer between two peers
  - Messages on a link flow in one direction only
  - All message transfers must be acknowledged (for reliability guarantees)
  - Multiple links can be combined in a session
  - Application creates (immutable) bare messages that have a body and an optional list of standard (e.g., message id) and application-specific properties
  - Messages may be annotated by intermediaries (via message headers)
  - Application data can be in any form/encoding: one can use AMQP for sending self-describing data

AMQP vs PVA

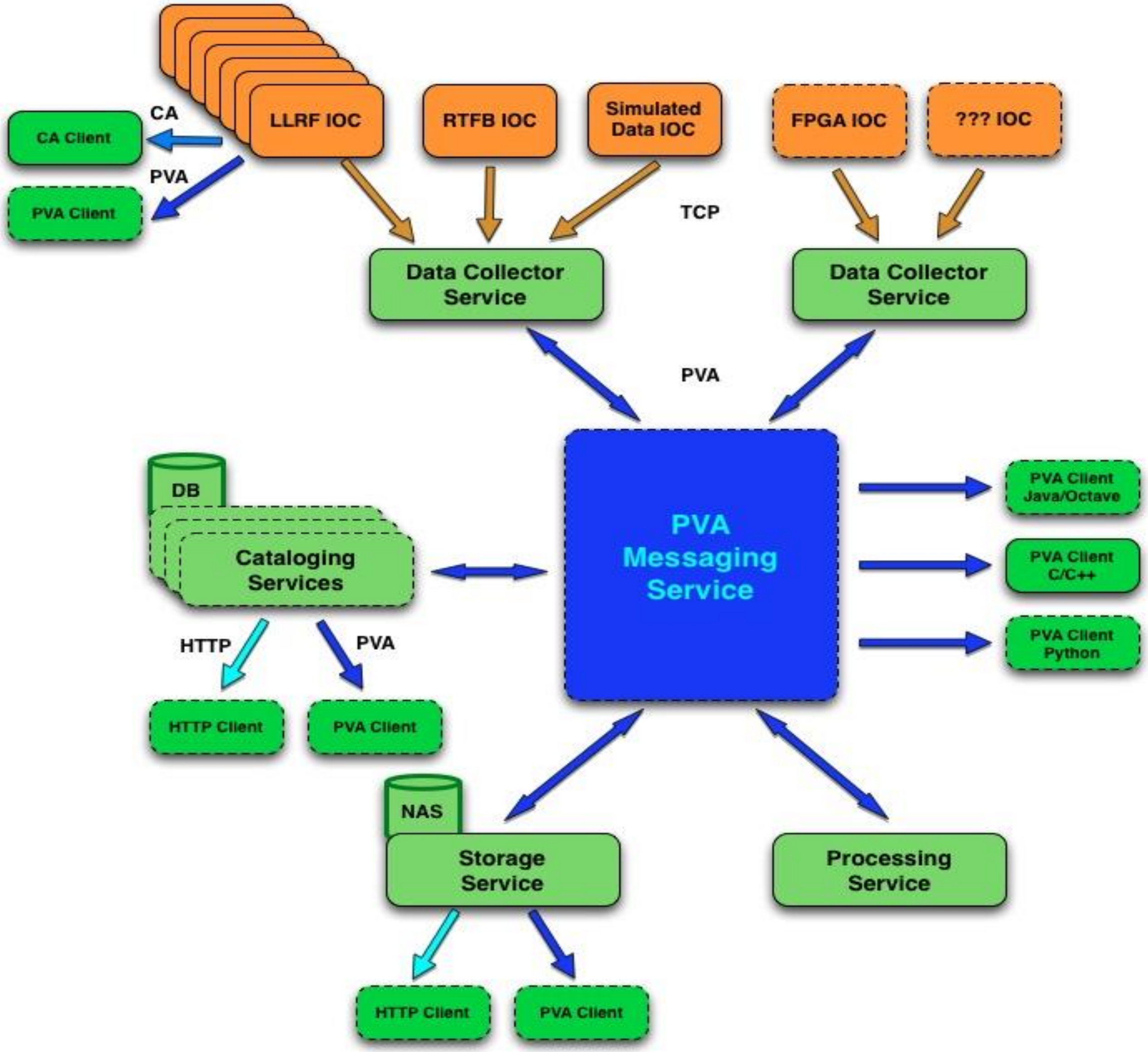
- PV Access: natural evolution of Channel Access, designed with EPICS applications in mind (for signal monitoring, scientific data services)
- Data type support:
  - Both protocols support all basic (primitive) types and strings
  - AMQP also supports Decimal32/64/128, TimeStamp, and Uuid
  - AMQP supports described types (primitive type + descriptor), PVA supports introspection data (describes type of user data item)
  - PVA supports Unions, AMQP does not
  - PVA supports BitSets (finite sequence of bits)
  - Both support composite types (structures)
  - Both support Arrays (sequence values of a single type)
  - AMQP supports (polymorphic) Lists and Maps (polymorphic mapping from distinct keys to values)
- PVA channel: connection to a single named resource that resides on some server (client-server model)
- AMQP type systems involve broker as intermediary: messages on a link flow in one direction only

AMQP vs PVA

- Protocols utilize different channel/link management
- Both protocols have a concept of control vs. application messages
- PVA application message headers are fixed size (8-byte long)
- PVA has predefined messages types (e.g., channel get, channel put, channel put-get, channel monitor, channel array, etc.)
- PVA servers must broadcast beacon messages over UDP (beacons are used for announcing new servers and server restarts); PVA channel search messages are typically sent over UDP, while data transmission uses TCP
- AMQP is built on top of TCP
- AMQP has built in support for transactions and security

- **PVA: optimized for performance, geared towards simplicity and efficiency**
- **AMQP: more flexibility, more complexity**

SPX Controls Software Architecture
(Possible Alternative)



LLRF IOC
RTFB IOC
Simulated Data IOC
FPGA IOC
??? IOC
CA
CA Client
PVA
PVA Client
TCP
Data Collector Service
Data Collector Service
PVA
PVA Messaging Service
PVA Client Java/Octave
PVA Client C/C++
PVA Client Python
DB
Cataloging Services
HTTP
PVA
HTTP Client
PVA Client
NAS
Storage Service
Processing Service
HTTP Client
PVA Client