An Infrastructure for (High Performance) Data Analysis at DESY

> Current status
> Plans for further development
> Conclusions

Sergey Yakubov, Stefan Dietrich, Uwe Ennslin, Jürgen Hannappel, Janusz Malka, Cartsen Patzke, Birgit Lewendel, Martin Gasthuber
DESY IT
Hamburg, 25 January 2018

HELMHOLTZ
ASSOCIATION

DESY

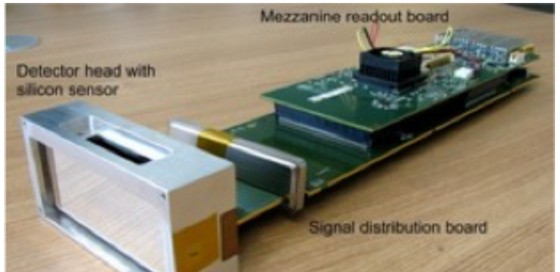
„Starting point“ - Detectors

- High rate detectors
- May run in parallel
- Next generation will come

Detector	OS/Access	File size/rate	Bandwidth
Pilatus 300k	Linux (Black box)	1,2 MB Files @ 200 Hz	240 MB/s
Pilatus 6M	Linux (Black box)	25 MB files @ 25 Hz 7 MB files @ 100 Hz	625 MB/s 700 MB/s
PCO Edge	Windows	8 MB files @ 100Hz	800 MB/s
PerkinElmer	Windows	16 MB + 700 Byte files @ 15 Hz	240 MB/s
Lambda	Linux	60 Gb/s @ 2000 Hz	7.5 GB/s
Eiger	Http (Black Box)	30 Gb/s @ 2000 Hz	3.8 GB/s

Pilatus 6M

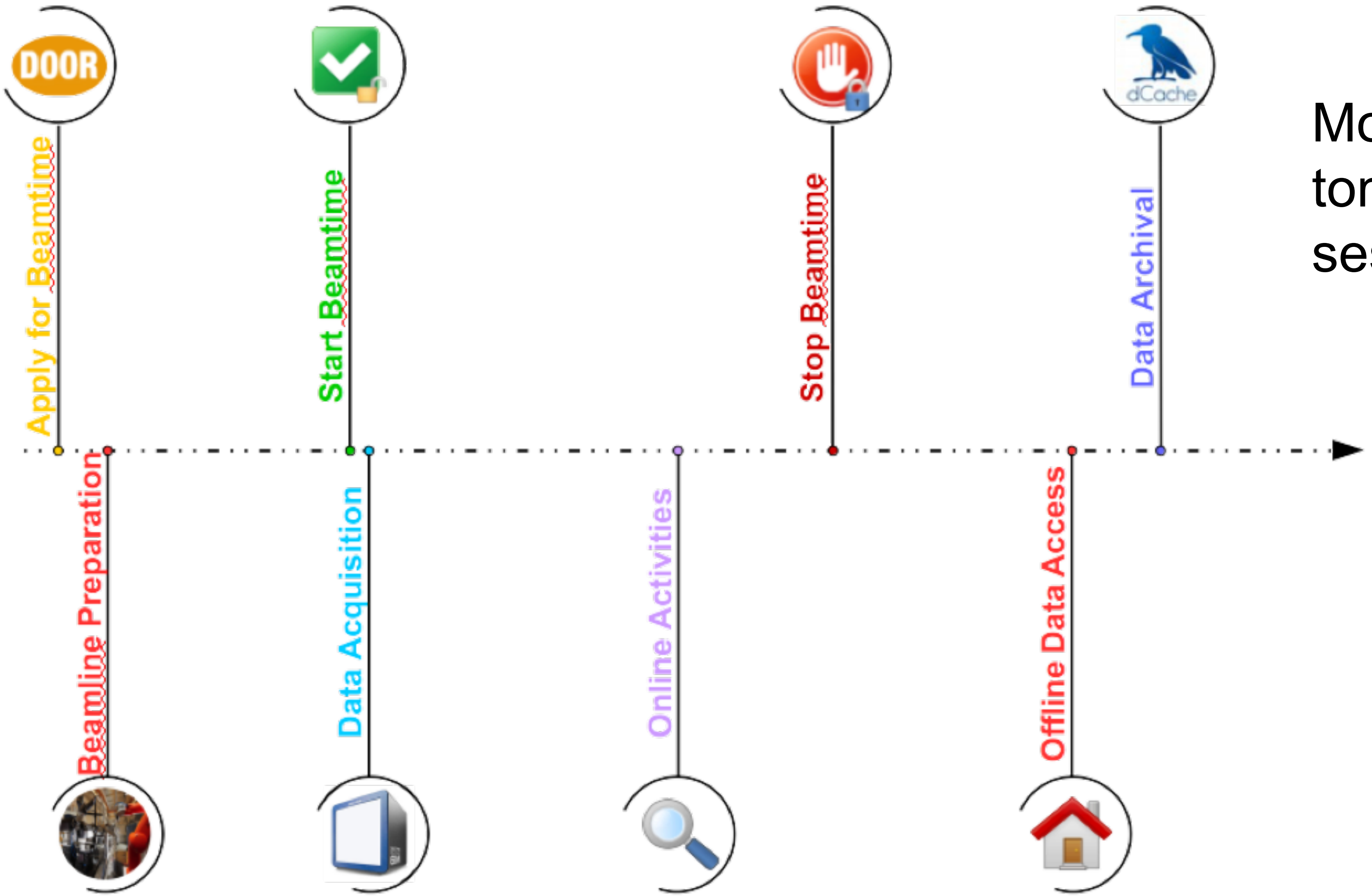
PCO Edge



Lambda

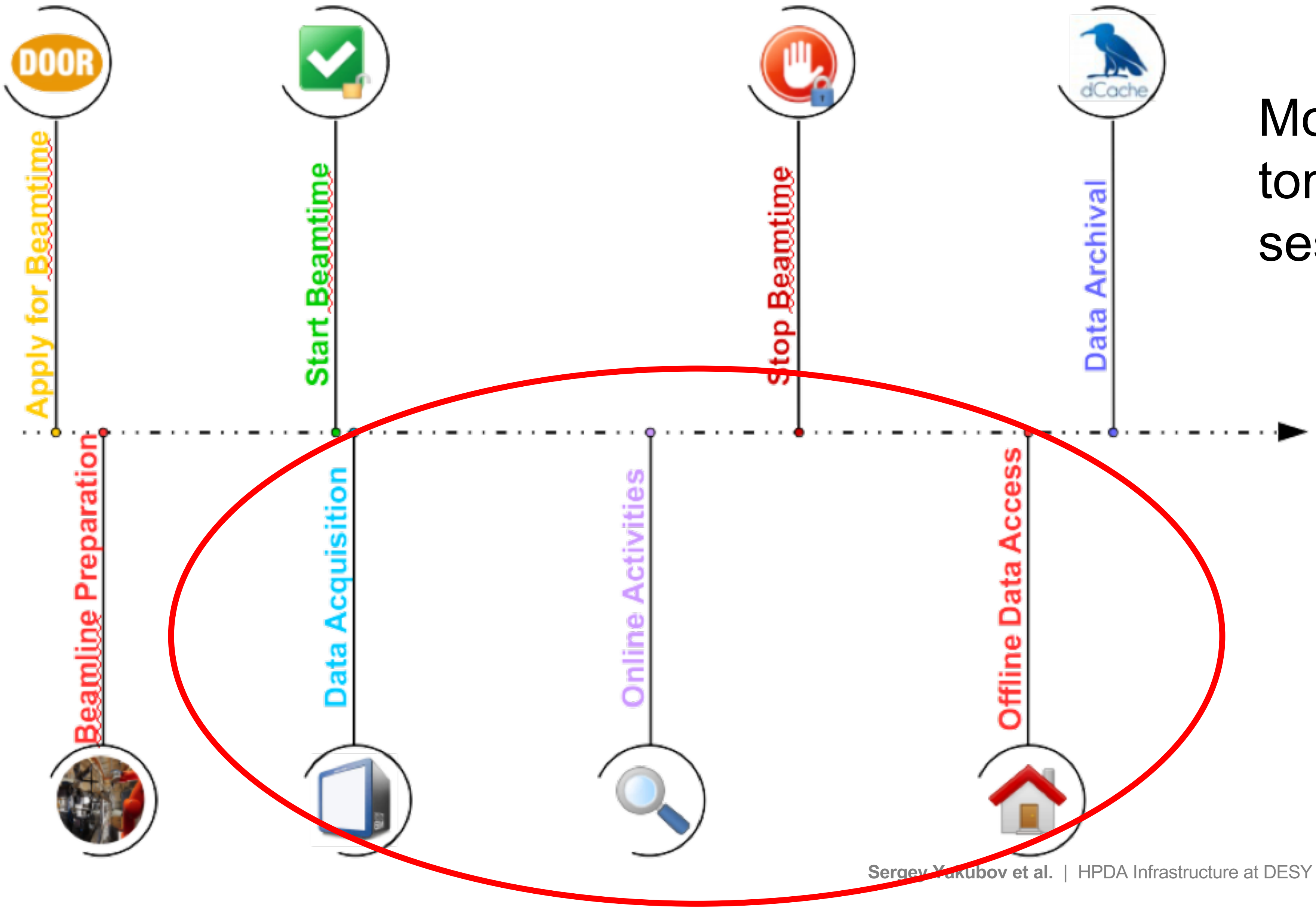
Key assumption – data is stored and analysed in a central compute center

ASAP³



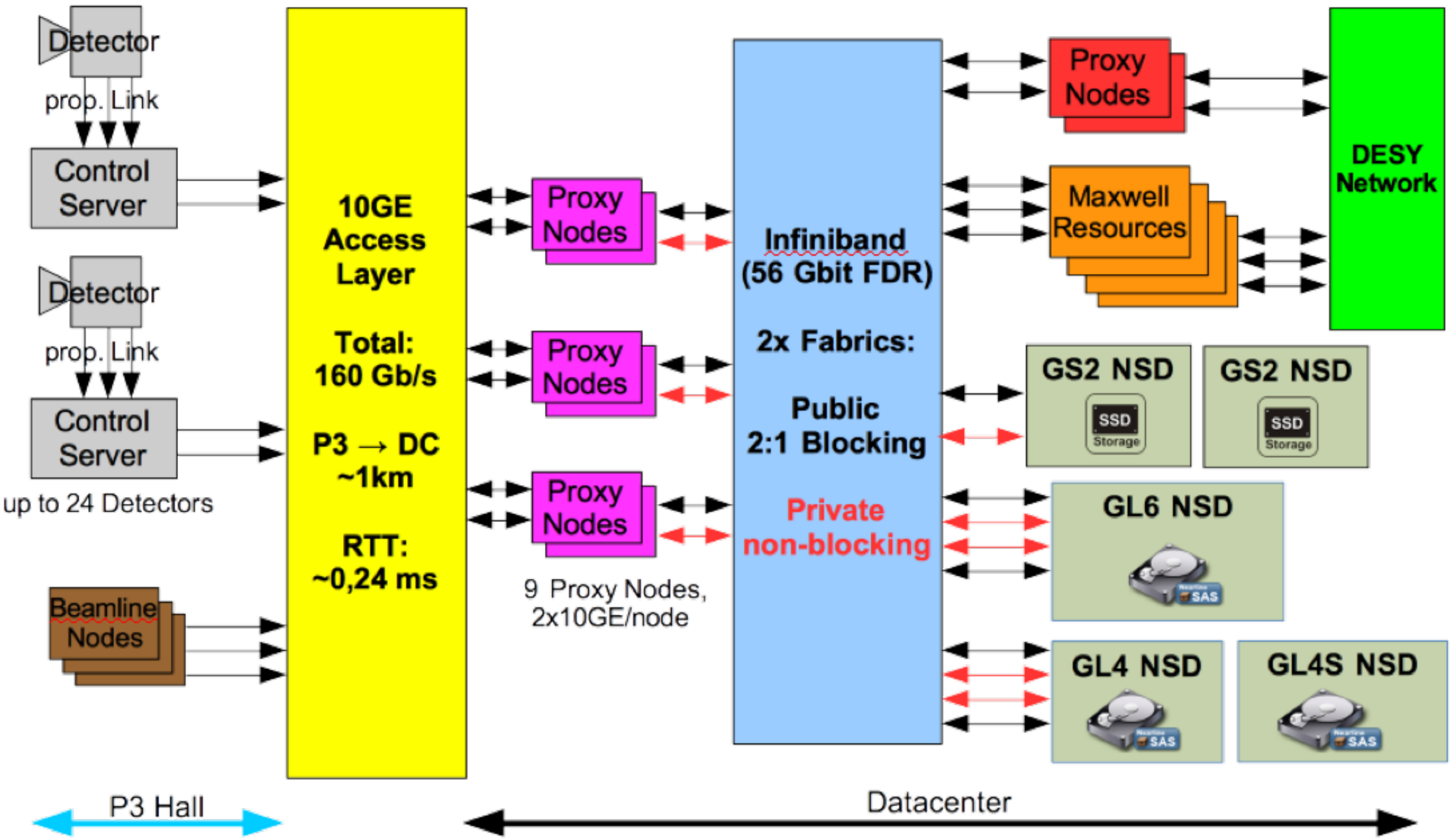
More information – tomorrow's poster session

ASAP³

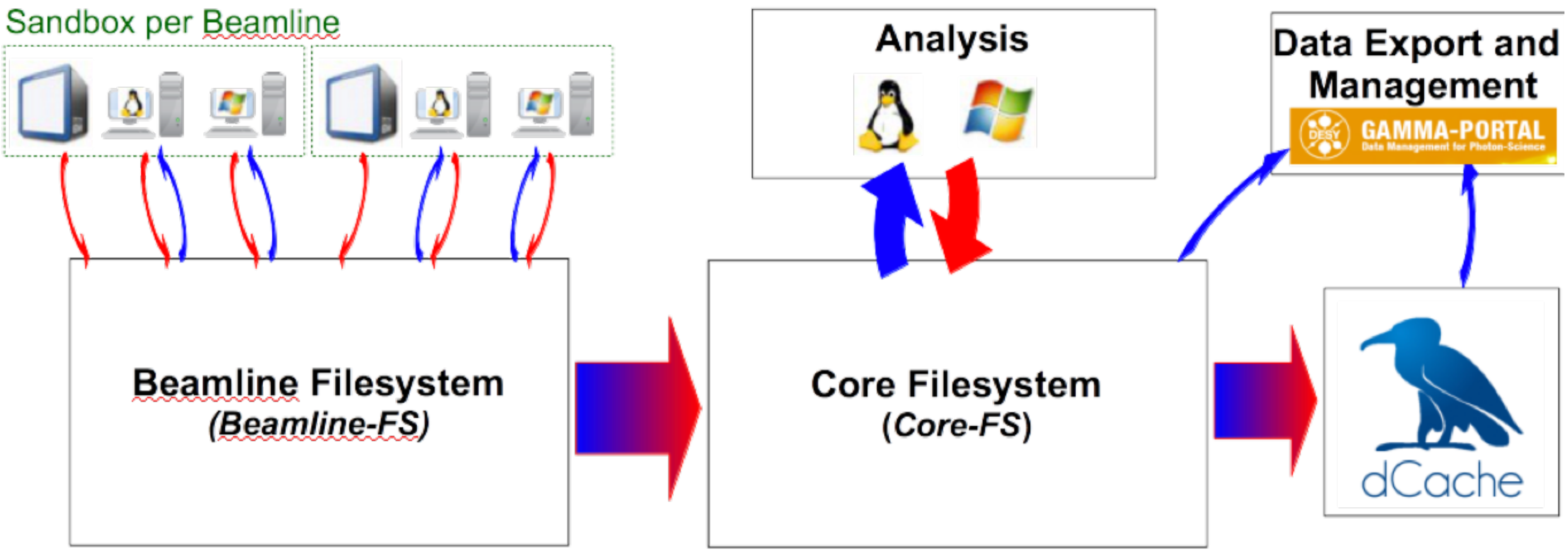


More information – tomorrow's poster session

ASAP³ – Hardware Architecture



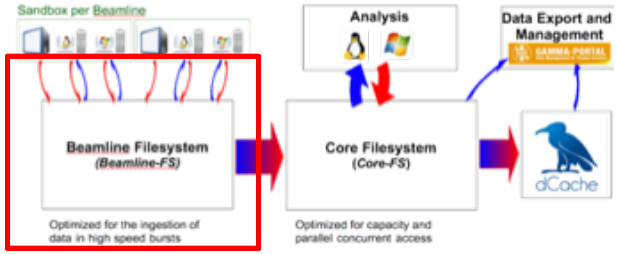
ASAP³ – Data Flow (as of 2015-2016)



Optimized for the ingestion of data in high speed bursts

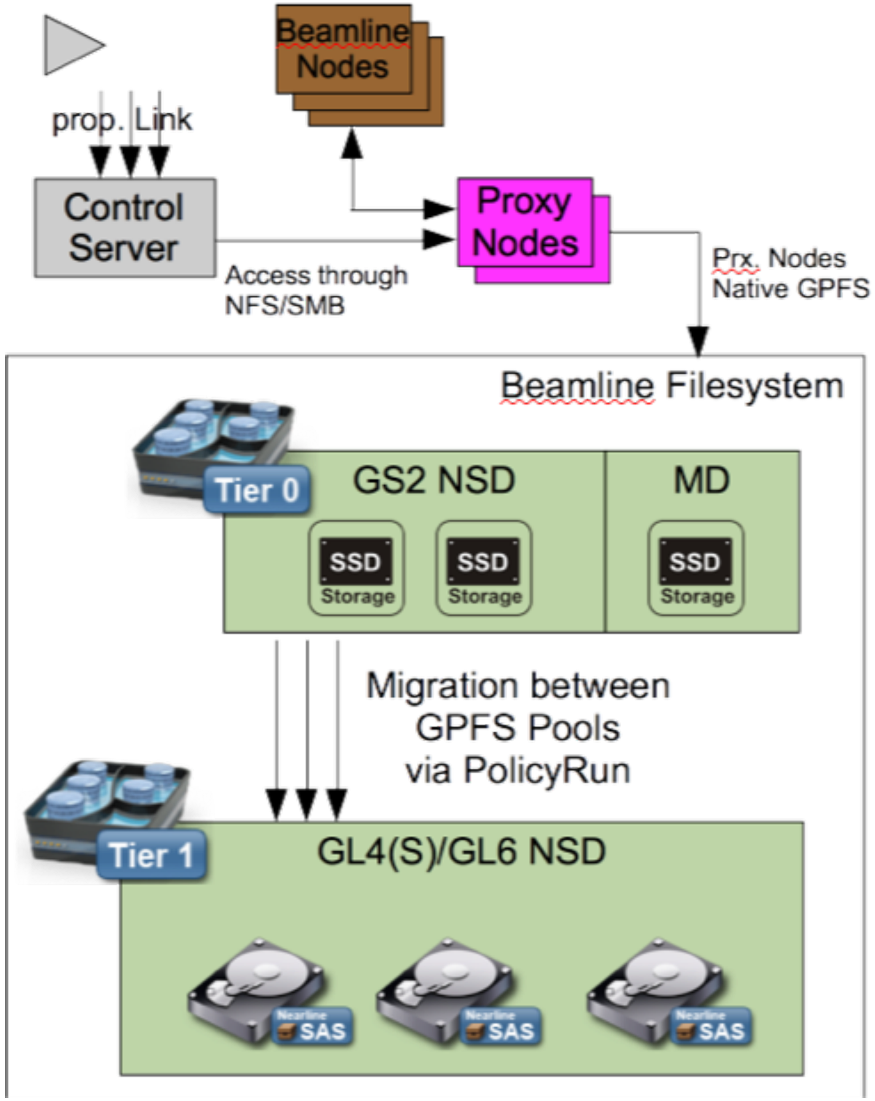
Optimized for capacity and parallel concurrent access

ASAP³ – Beamline filesystem

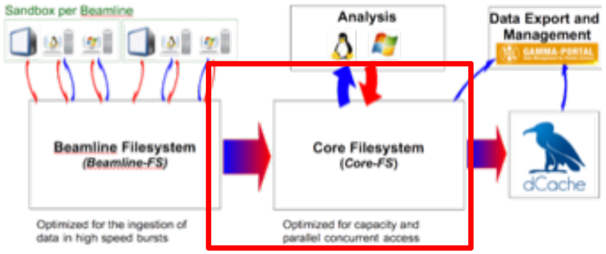


- Only host based authentication, no ACLs
- Access through NFSv3, SMB
- Optimized for performance
  - NFSv3: ~600 MB/s
  - SMB: ~300-600 MB/s
- Tiered Storage
  - Tier 0: SSD burst buffer (<10 TB)
  - Migration after short period of time
  - Tier 1: ~90 TB capacity

Limited/no access for user analysis jobs via native GPFS

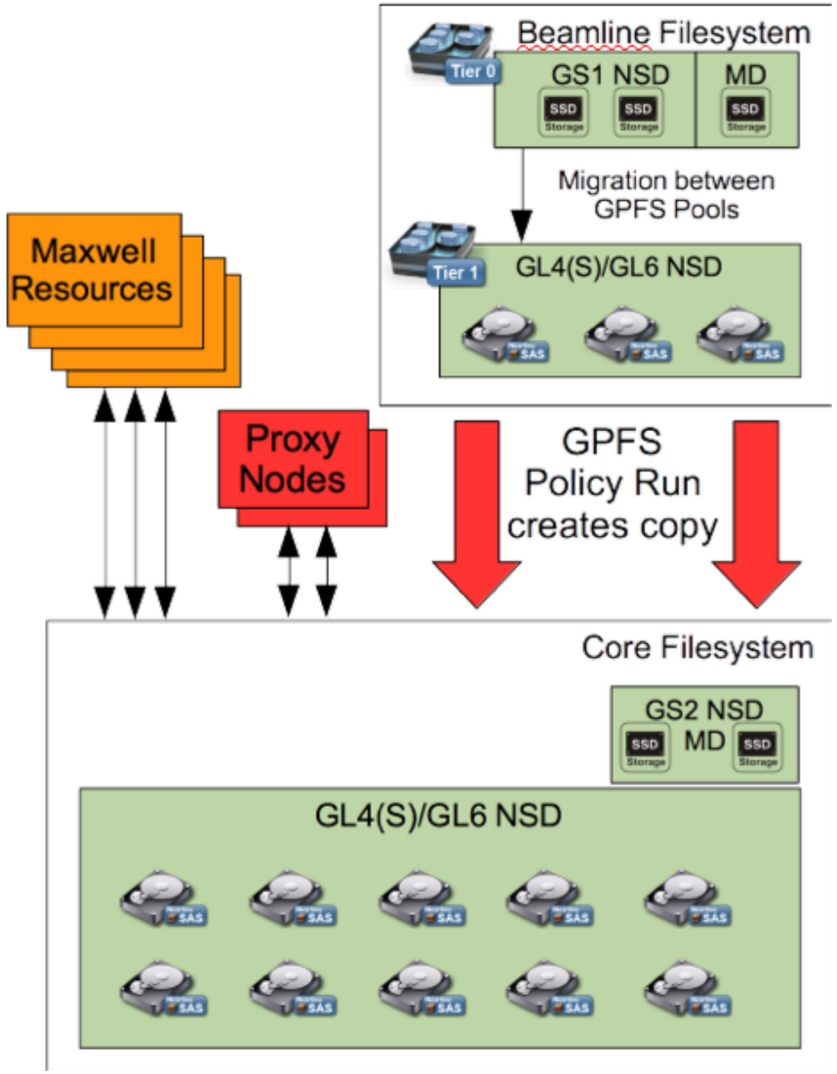


ASAP³ – Core filesystem



- Full user authentication
- Access through NFSv3, SMB or native GPFS
- GPFS Policy Runs copies data
  - Beamline → Core Filesystem
  - Single UID/GID
  - ACL inheritance gets active
- 2 snapshots per day

Full access for user analysis jobs (delay 4 min)

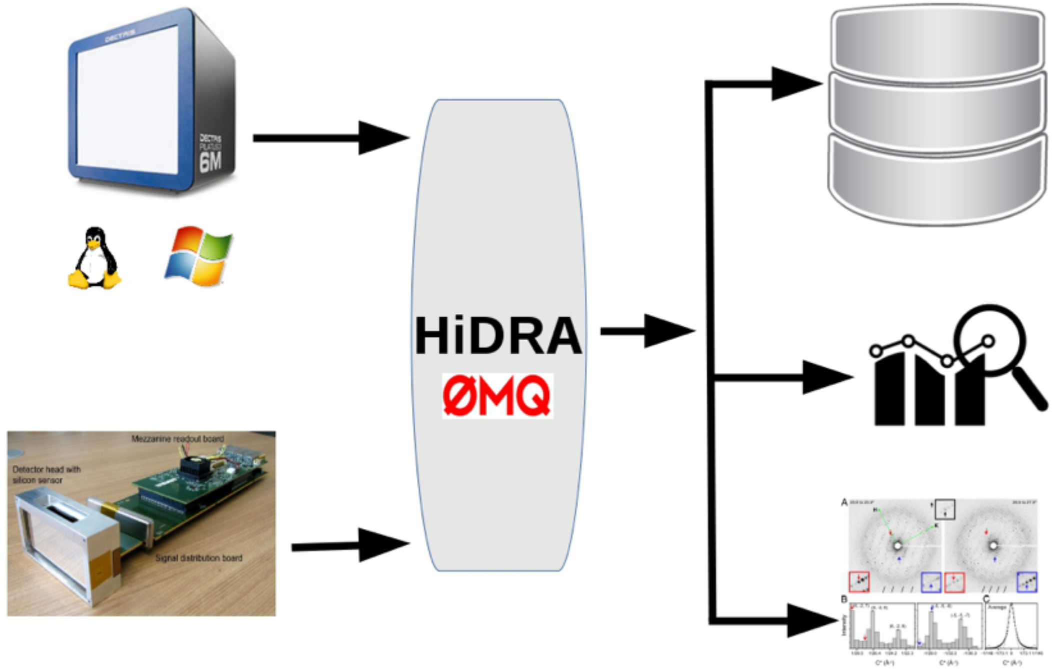


Status as of 2016

		Bandwidth
Detectors	Pilatus 300k	240 MB/s
	Pilatus 6M	700 MB/s
	PCO Edge	800 MB/s
	PerkinElmer	240 MB/s
	Lambda	7.5 GB/s
	Eiger	3.8 GB/s
Experimental Hall to Maxwell (aggregate)		20 GB/s
NFS,SMB		**600 MB/s**

HiDRA*

> Generic tool set for high performance data delivery
> Based on Python and ZeroMQ
> Messaging system
  - Push to subscribers
  - Request-reply
> Various use cases
  - Storing data in filesystem
  - Online analysis/monitoring



* Talk from Manuela Kuhn

Status as of 2017- present

		Bandwidth
Detectors	Pilatus 300k	240 MB/s
	Pilatus 6M	700 MB/s
	PCO Edge	800 MB/s
	PerkinElmer	240 MB/s
	Lambda	7.5 GB/s
	Eiger	3.8 GB/s
Experimental Hall to Maxwell (aggregate)		20 GB/s
NFS,SMB		600 MB/s
HiDRA		**1.2 GB/s – now** 1.2 GB/s x N – in development

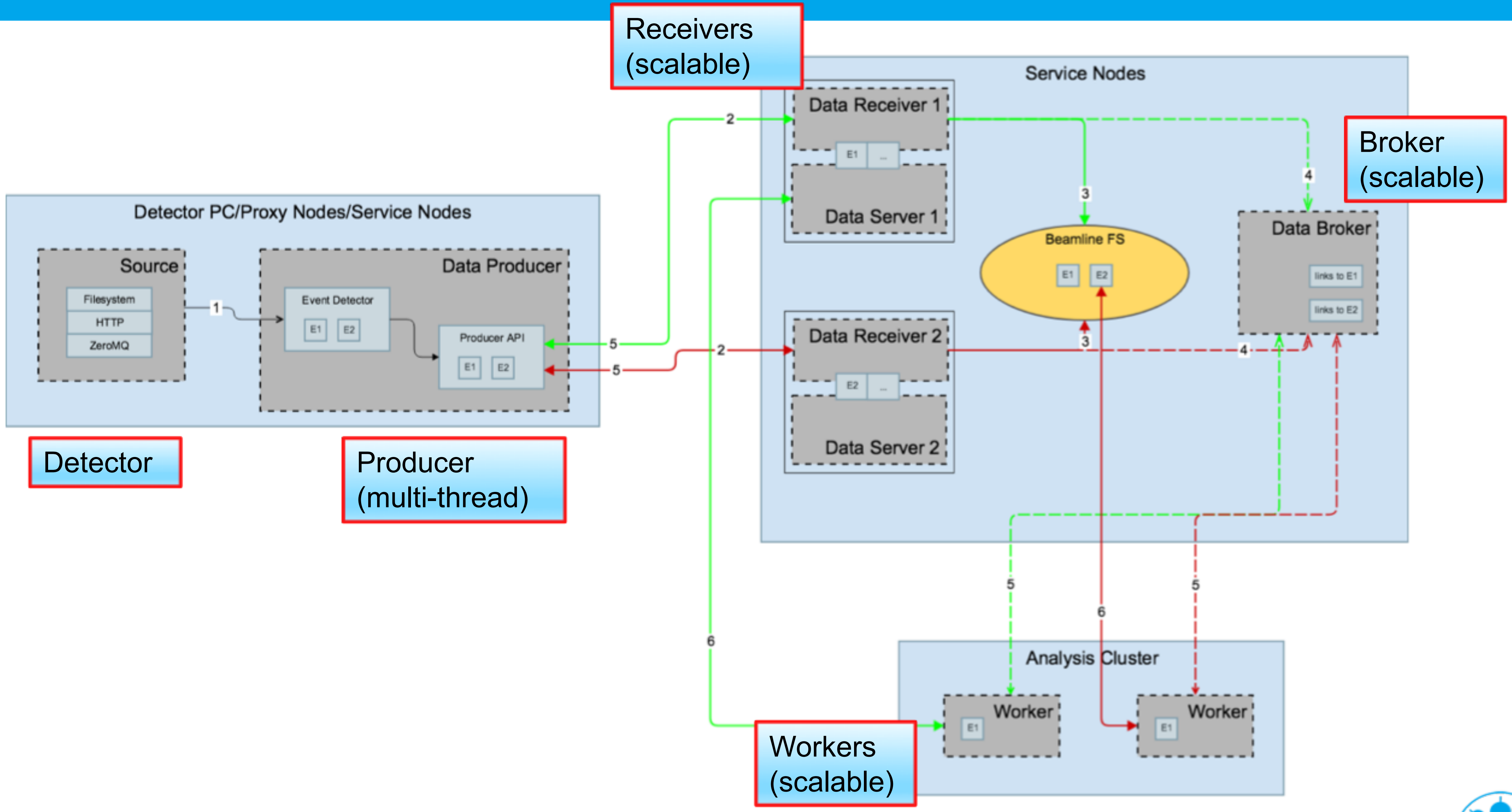
Challenges Ahead (PETRA 3/4 and Flash)

- Scalability
  - Producer – network scaling: Nx10GE, Nx40GE (multithreading - Python?)
  - Workers – able to start large number (100-1000) of workers
- Higher transfer rates
  - Multiple links
  - RDMA over Infiniband or RDMA over Ethernet (ZeroMQ?)
- Decouple data taking and data processing
  - Variable injection/data processing rate
  - Support for offline analysis

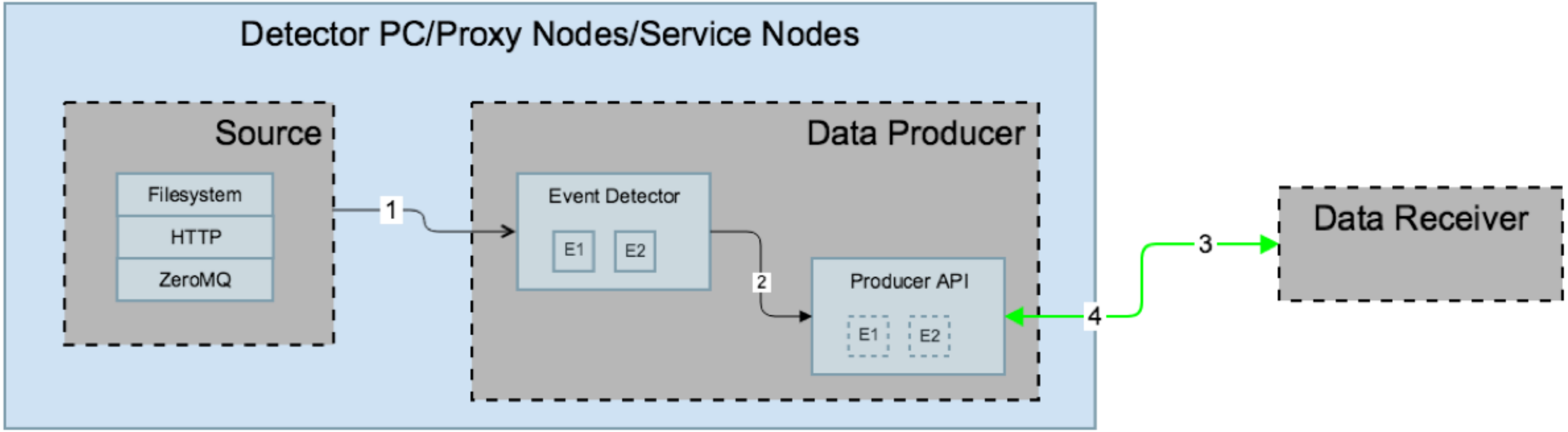
ASAP::O - Addressing the Challenges

- A new middleware platform for high-performance next-generation detector data analysis
- Supports data analysis synchronous and asynchronous to data taking
- Scalable (N detectors, K network links, L service nodes, M analysis nodes)
- Highly available (services in Docker containers managed by Kubernetes)
- Efficient (C++, multi-threading, RDMA, …)
- Provides user friendly API interfaces (C/C++, python, REST API)
- Supports various platforms and OS
- Compatible with HiDRA API

ASAP::O - Architecture



ASAP::O - Producer



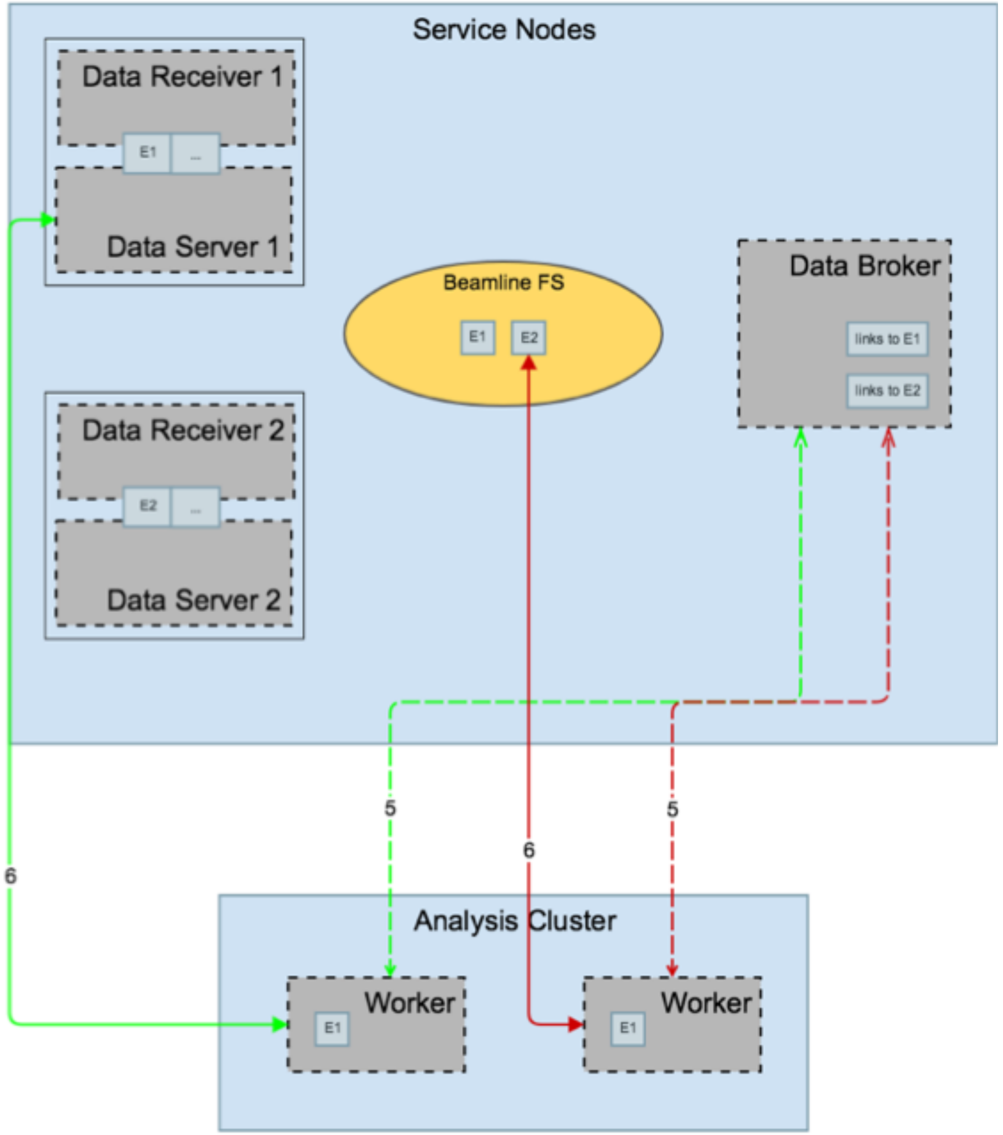
- Event Detector to be written for various detectors
- Python interface to be discussed :)
- Interface to HiDRA

Get data from a detector and send it to the Receiver using Producer API

ASAP::O - Producer Use Cases

- Location
  - Detector (blackbox) writes locally to the NFS/SMB mount point
  - Detector writes/sends data to the detector's PC
  - Detector (blackbox) sends data to proxy node
- Data granularity
  - Sub-images
  - Single images
  - HDF5 files with multiple images
- Detector/user metadata
  - No metadata
  - Metadata is periodically written

ASAP::O - Worker



- Request/Reply pattern
- Same user code for offline/online
- Python interface will be provided
- Can/will run in isolated Docker containers on Maxwell nodes of certain type (memory, GPU,…) with access to Beamline FS
- Interface to HiDRA

Use Worker API to retrieve data

ASAP::O - Worker Use Cases

> Software

- User code (Linux, Windows), C++/Python
- MatLab, IDL, commercial applications

- reduced performance possible

> Analysis

- online
- offline
- testing (online with virtual detectors/offline)
- at 3rd party sites/at home

> Parallelization

- All images can be processed independently
- Sequential processing (with chunks of N images)

> HDF5 writer

- pack images into HDF5 files
- add metadata to HDF5 files

> Using filters/queries

- process only images with a specific condition (from time A to time B, from frame N to frame M, with metadata X, …)

> Multi-stage processing/pipelining

- Worker as Producer
- Connectors to Apache Storm, …

ASAP::O - Current Status and Development Process

> Work in progress (first version this year)

> Agile project (almost)

- Following Kanban approach, without time tracking

> Using Atlassian Tools provided by DESY IT for CI/CD.

- Confluence for documentation (development workflow, code conventions, design sketches, …)
- Bitbucket for source code
- Jira for task tracking
- Bamboo for testing/deployment

> Heavily tested

- Unit tests (sometimes written first)
- Integration tests
- Test coverage close to 100%
- Memory leaks testing
- Testing on both Windows and Linux

Conclusions

- Data analysis infrastructure in current state supports high rate data transfer to the compute center and is able to support online data analysis
- A new platform ASAP::O is being developed
  - Even higher rate data transfer to beamline and core filesystem
  - High performance online and offline data analysis using same user code
  - Transparent access to images (no need to know directory structure, file names)
  - Various modes to access images: last image, arbitrary image, ordered sequence, in parallel
  - Select images based on metadata
  - Multi-stage processing of data streams and connections to other datastream processing frameworks
  - Prepare everything at home, deploy on Maxwell (e.g. Docker)
  - …
- Close(r) coordination with users is essential for success. Participation in development is possible/welcome.