HiDRA

High Data Rate Access

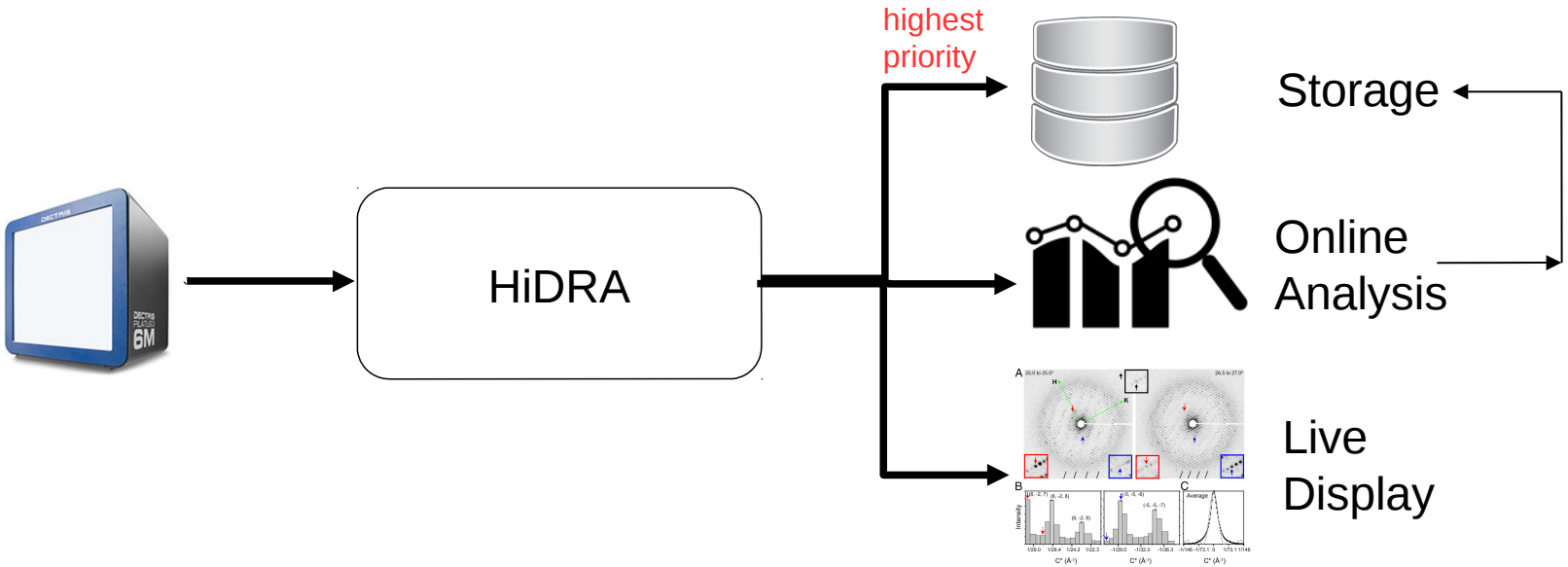
Manuela Kuhn
DESY FS Detector Systems

New concepts in ultra fast data acquisition
PSI - April 2018

HELMHOLTZ GEMEINSCHAFT

DESY

HiDRA


highest
priority
Storage
HiDRA
Online
Analysis
Live
Display

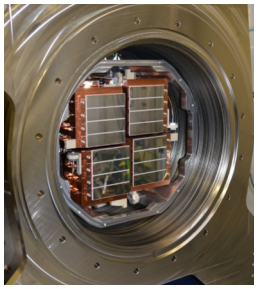
Challenges: New Detectors

> Current and future detectors have increasing demands: Eiger, Lambda, Percival, AGIPD

> Support of the next generation detectors

→ Data has to be drained from detectors before memory exhausted

Pilatus



AGIPD

Eiger

Lambda

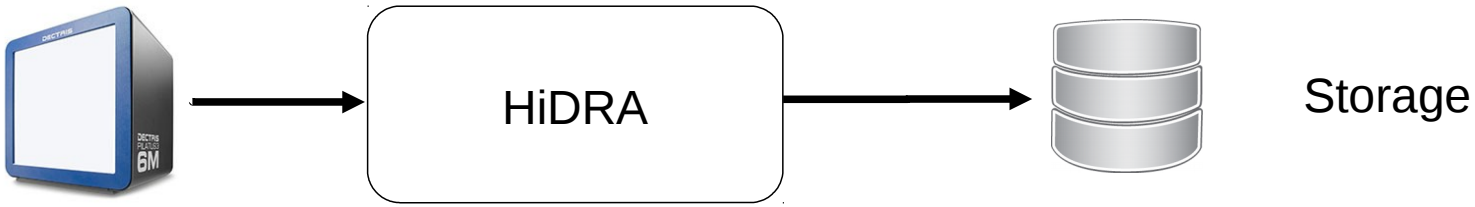
Detectors

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
AGIPD	Linux	16 x 260 MB @ >4.5 MHz	15 GB/s

Challenges: New Detectors

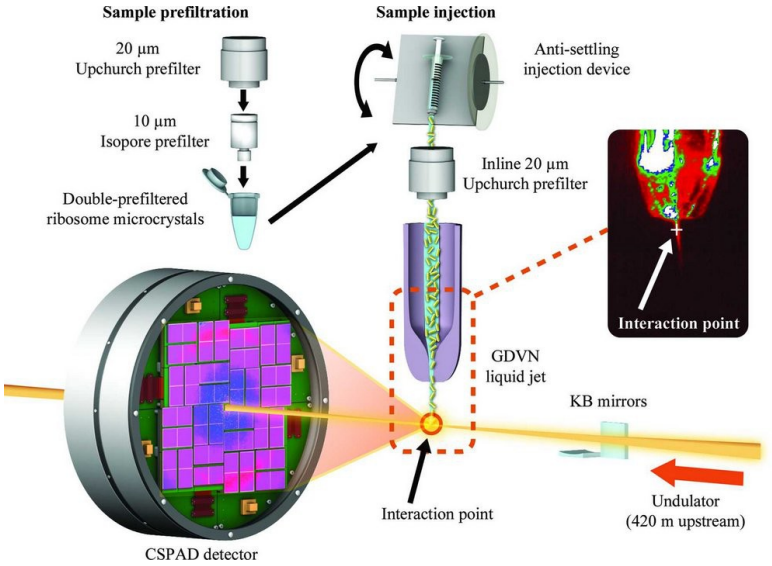
- Data rates exceeds 10GE network connection
- Images generated at kHz frequencies
  - Millions of files per experiment
- Wide variety of file and data formats
- SMB, NFS Network File System not fast enough (regardless of network technology)

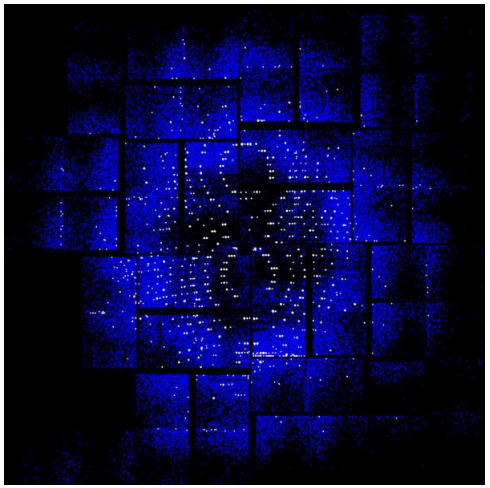
HiDRA



Challenges: Experimental conditions and setup

- Support for next generation experiment setups (e.g. more than one detector per beamline,...)
- Decouple persistent storage and selective image collection
- Experiment conditions have to be monitored/analyzed in near-real time to avoid the collection of unfavorable data



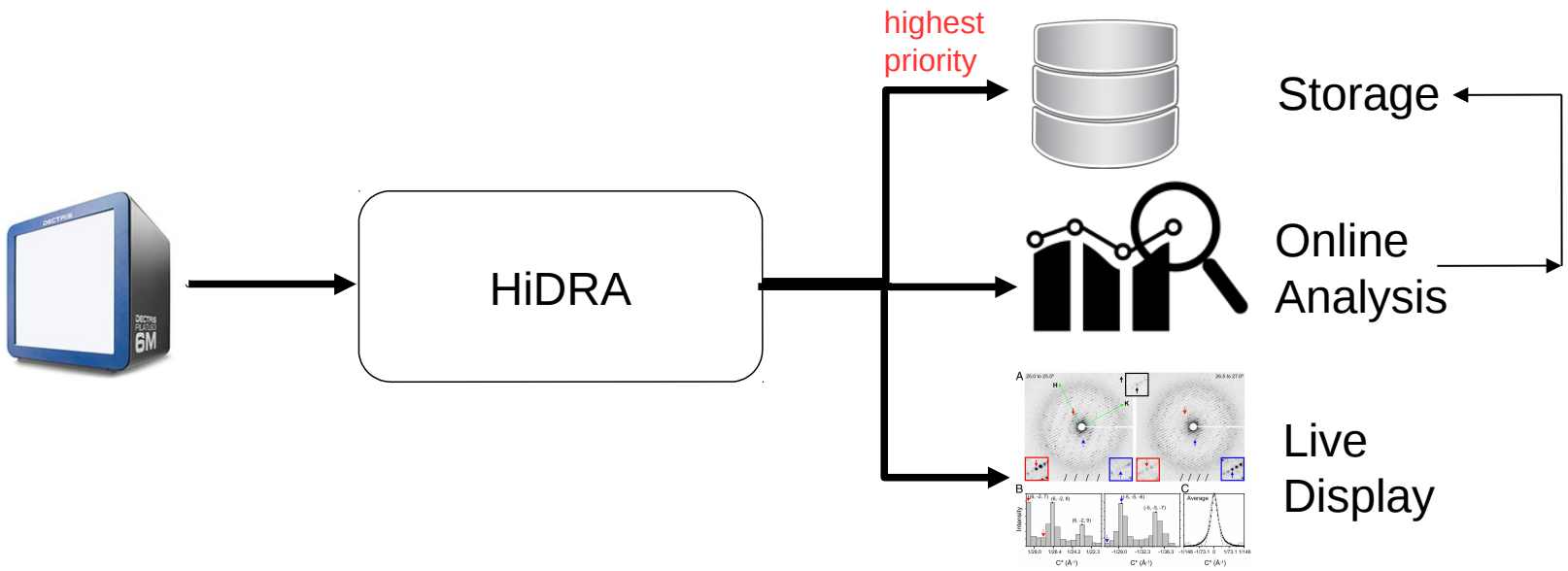


HiDRA

Development of HiDRA

> Client-server concept based on python and ØMQ

> Generic tool set for high performance data multiplexing with different qualities of service



Developed as a common project between Central-IT, FS-EC and CFEL with the help of PETRA III beamlines

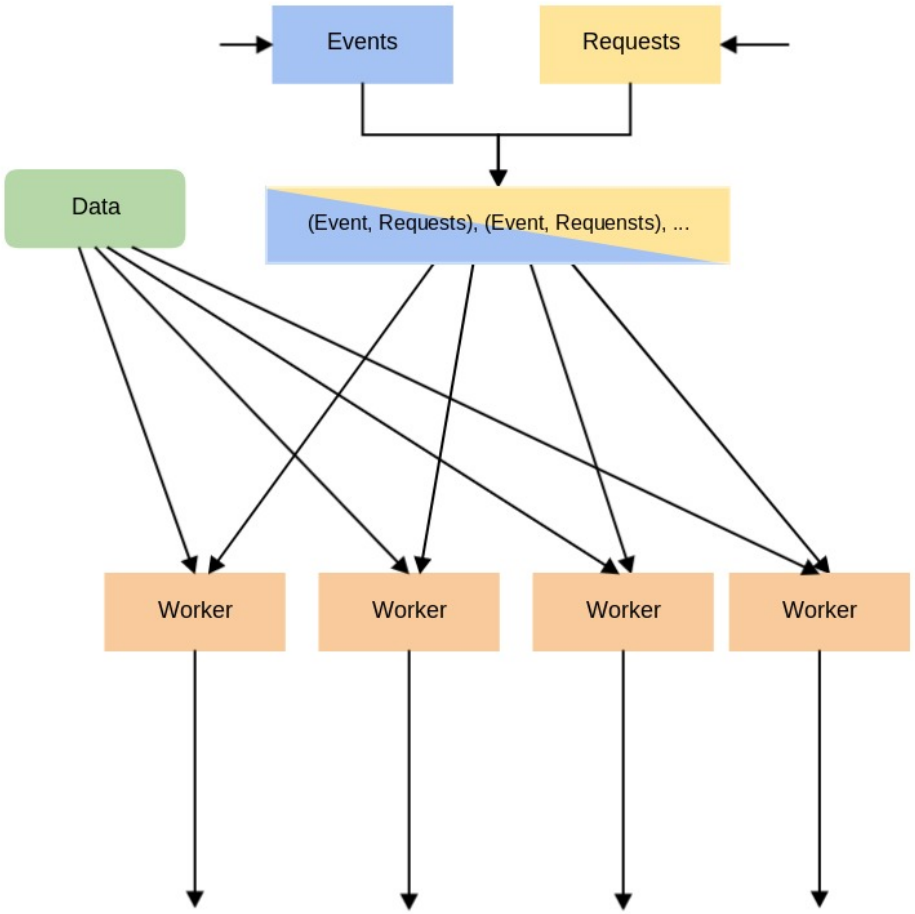
HiDRA

- Directly store the data in the storage system
- Send data to online monitoring or analysis framework
- Modular architecture (divided into event detectors, data fetchers and receivers)

→ This gives the possibility to adapt the software to specific detectors directly

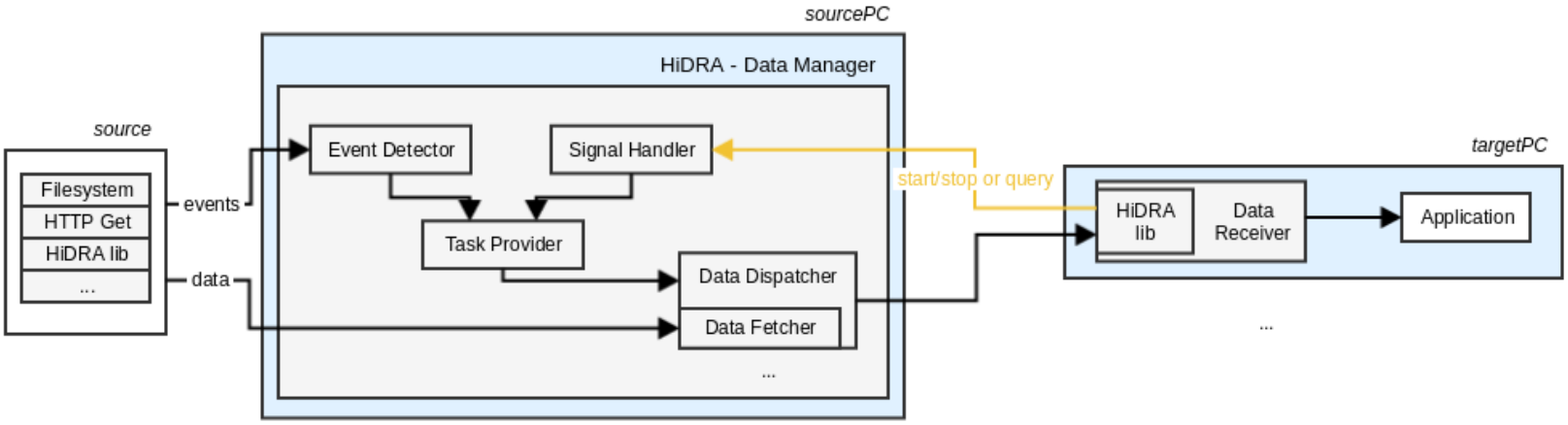
- Facility independent: Adaptable to other photon sources and storage systems
- Open source
- Performance limits not yet hit (saturate a 10 GE link, is able to handle 2000 Hz)
- Successfully used in multiple experiments

Parallelization



- Fan-out
- Events are coupled with requests and send to workers
- Workers process events:
  - Get data corresponding to the event
  - Send data to the targets which sent the requests

HiDRA Architecture



Available event detectors:

- Based on inotifyx library (Linux)
- Based on watchdog library (Linux/Windows)
- Get events via HTTP
- Get events via an API (C\Python)

Available data fetcher:

- Read from file system
- Get data via HTTP
- Get data via an API (C\Python)

→ easily expandable

Available receiver types:

- Store as files
- Forward to an application

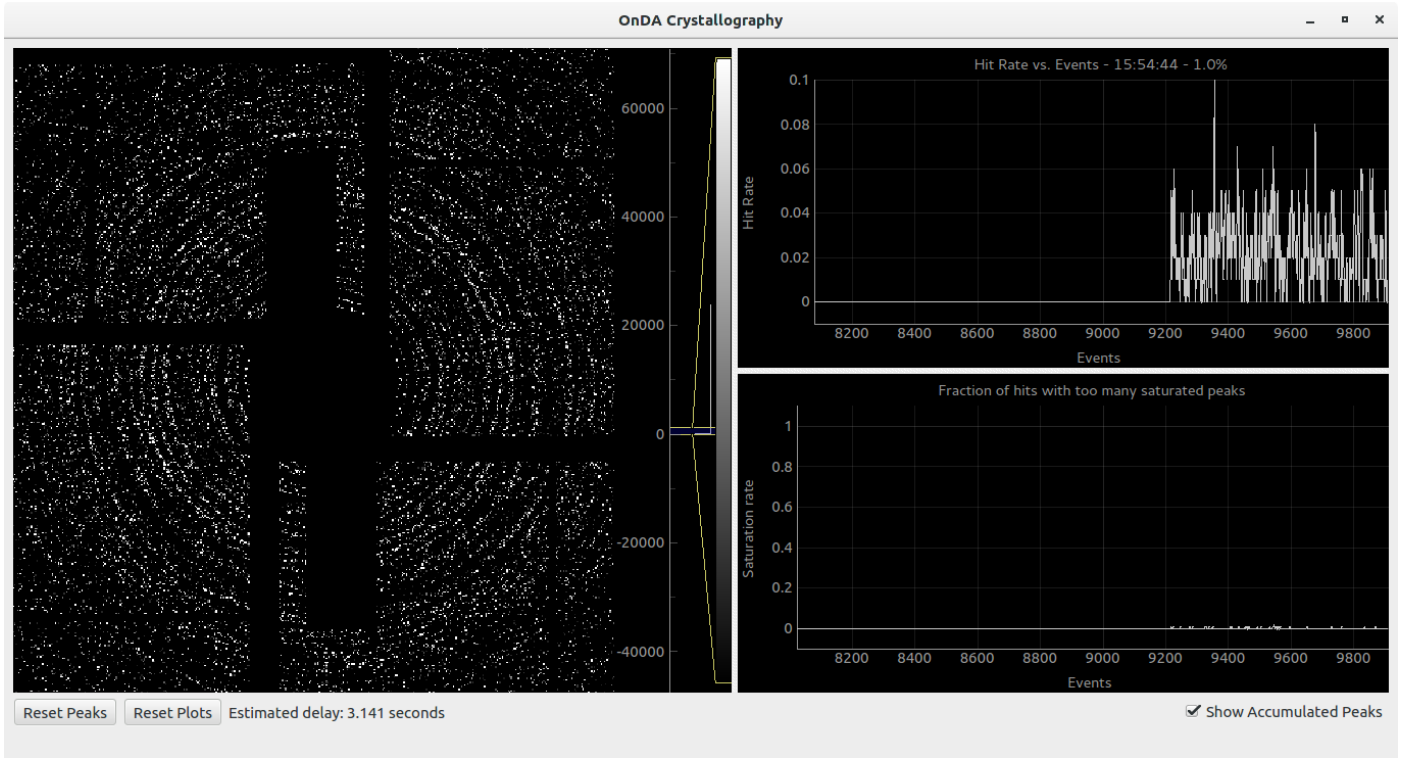
Connecting to HiDRA – HiDRA library

- Connecting to HiDRA
  - Handling the authentication
  - Signal handling
- Requesting data
  - Stream or query
  - Data+metadata or metadata only
  - Priority
  - Choosing File type

```python
from hidra import Transfer


if __name__ == "__main__":

    signal_host = "host-where-hidra-is-running.desy.de"
    target_host = "host-where-data-should-be-send-to.desy.de"
    target_port = "50101"

    targets = [[target_host, target_port, 1, ".*(tif|cbf)$"]]
#    targets = [[target_host, target_port, 1, [".tif", ".cbf"]]]

    query = Transfer("QUERY_NEXT", signal_host)

    query.initiate(targets)

    query.start()

    [metadata, data] = query.get()

    query.stop()
```

- Separation between set up on server (initiate) and client (start)
  - To enable master/worker architecture (no data duplication between the workers)

Application: OnDA



> OnDA is an online monitor

> Focus: Speed and stability, Portability, Flexibility

→ Mariani et al., "OnDA: online data analysis and feedback for serial X-ray imaging", J. Appl. Cryst. (2016). 49, 1073–1080

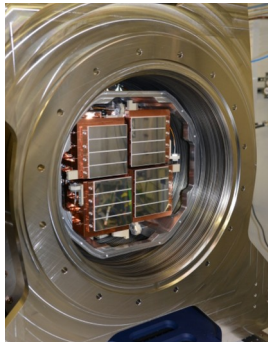
Supported Detectors

- PILATUS
- EIGER
- LAMBDA
- AGIPD
- JUNGFRAU
- PCO EDGE (Windows)
- Perkin Elmer (Windows)

Pilatus

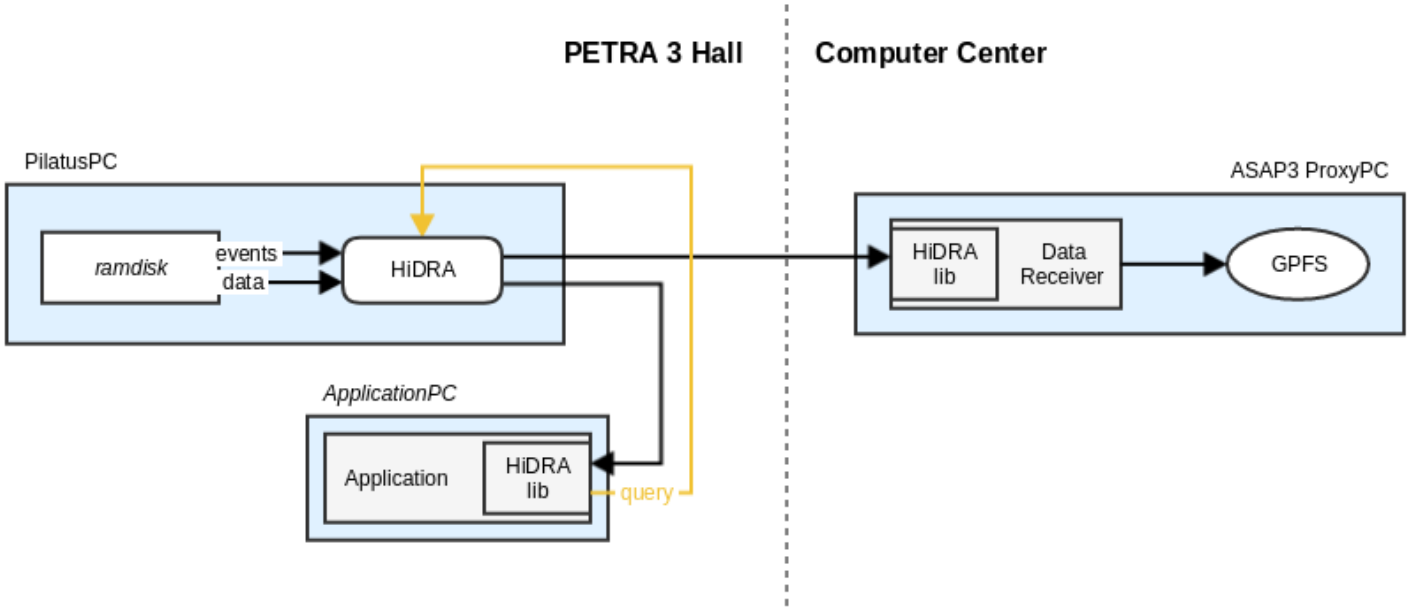
Eiger

Lambda



AGIPD

Pilatus



Constraints:

> Operating systems provided with detector, no updates permitted

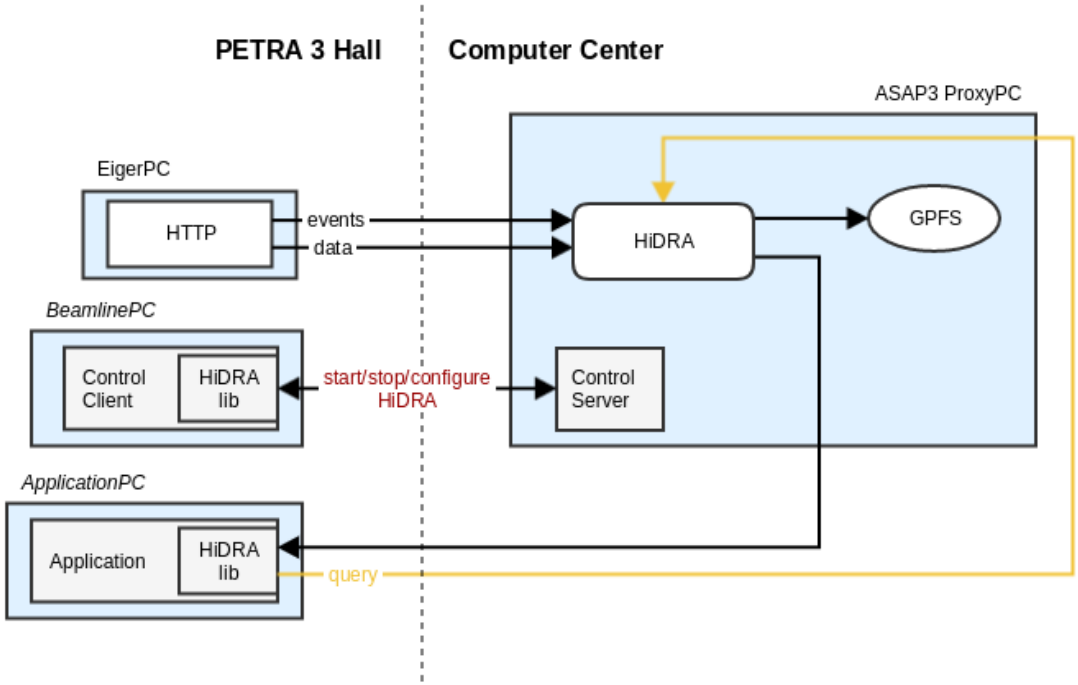
- SuSE 10

> No software allowed to being installed

→ freezing software + deploy as zipped packages

Same approach used for Windows detectors

Eiger

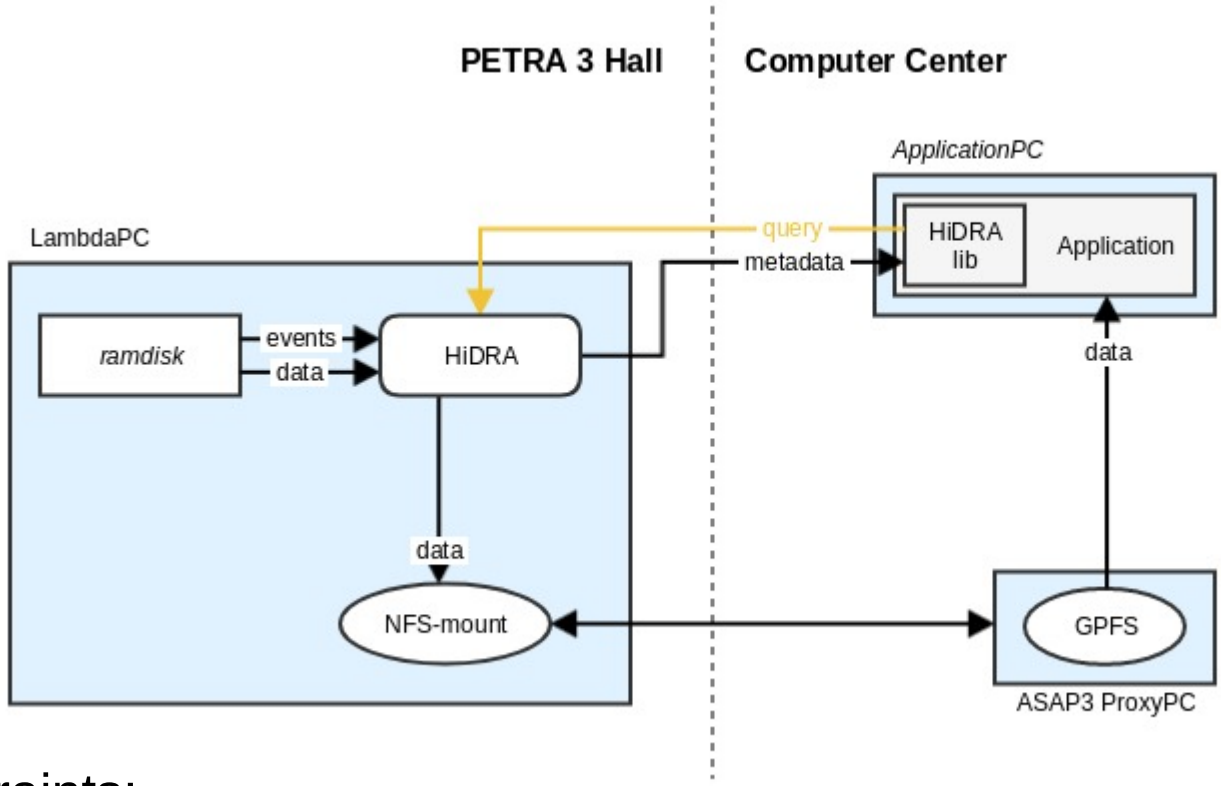


Constraints:

> No Detector-PC

> Data can only be pulled

Lambda, AGIPD and Jungfrau



Constraints:

> Data blocks too big to send to an application directly

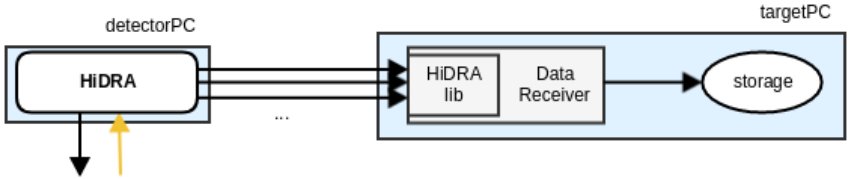
> hdf5 library does not support reading from stream

→ notify application about the newest data to be read from the storage system

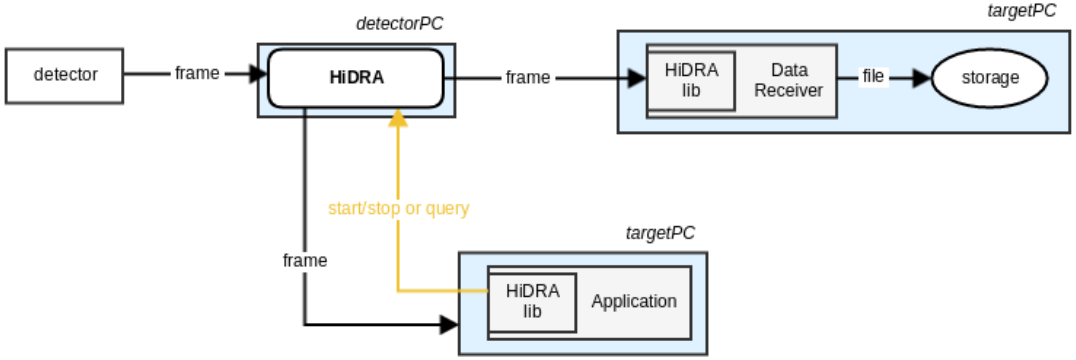
Outlook and Future Development Plans

> Continue to extend HiDRA for use at multiple facilities with upcoming detectors

> Usage of multiple network links in development



> Sending frames directly from the detector and build HDF5 on the receiver side



Further information

Git repository:

https://stash.desy.de/projects/HIDRA/repos/hidra/browse

Documentation:

https://confluence.desy.de/display/hidra/HiDRA