Containerized services for FEL data processing

Anton Teslyuk, Sergey Bobkov, Viacheslav Ilyin

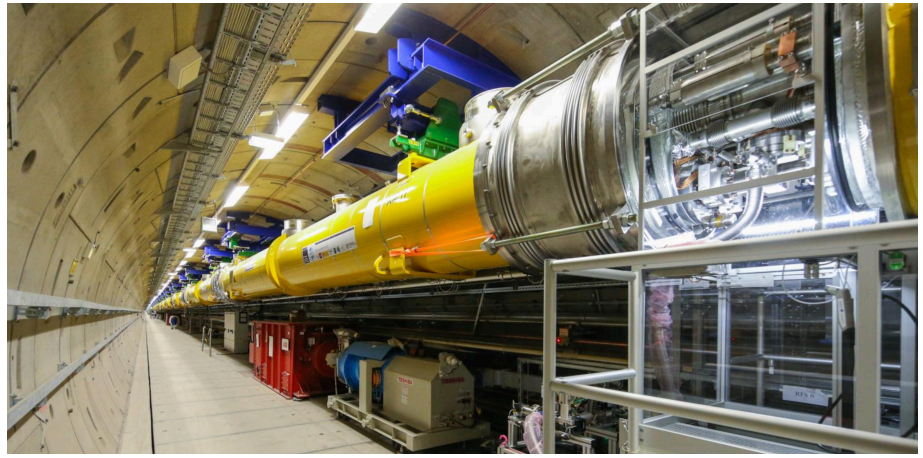
NRC Kurchatov Institute

NEC'2019

30 Sep – 4 October 2019, Budva, ME

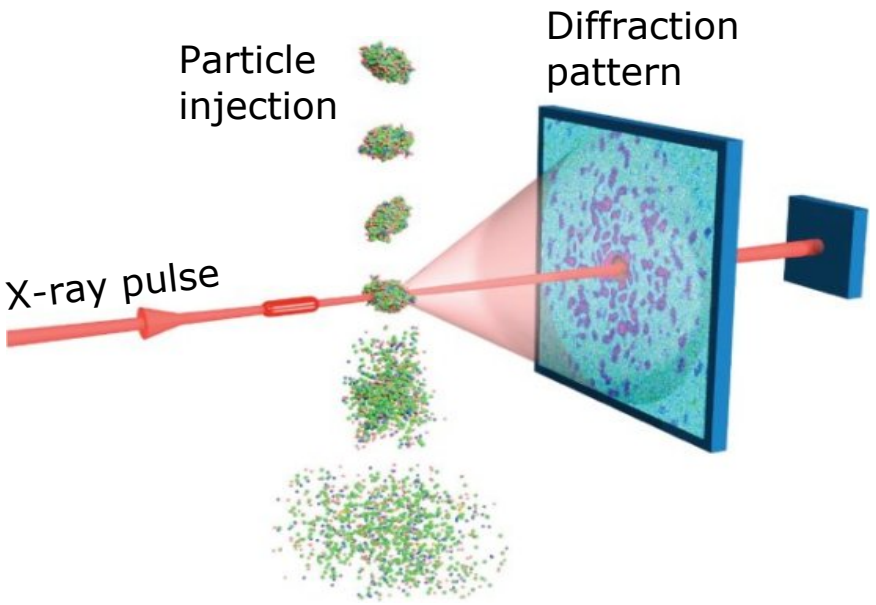
European XFEL

- X-Ray Free-Electron Laser - mega science research facility
  - High brilliance (10^9 times more than conventional X-ray source)
  - High frequency: up to 27000 flashes per second
  - Wavelength range: 0.05-4.7 nm
  - Short pulses: less than 100 fs
- Construction start – Jan 2009
- First experiments – Sep 2017



SPI Experiments

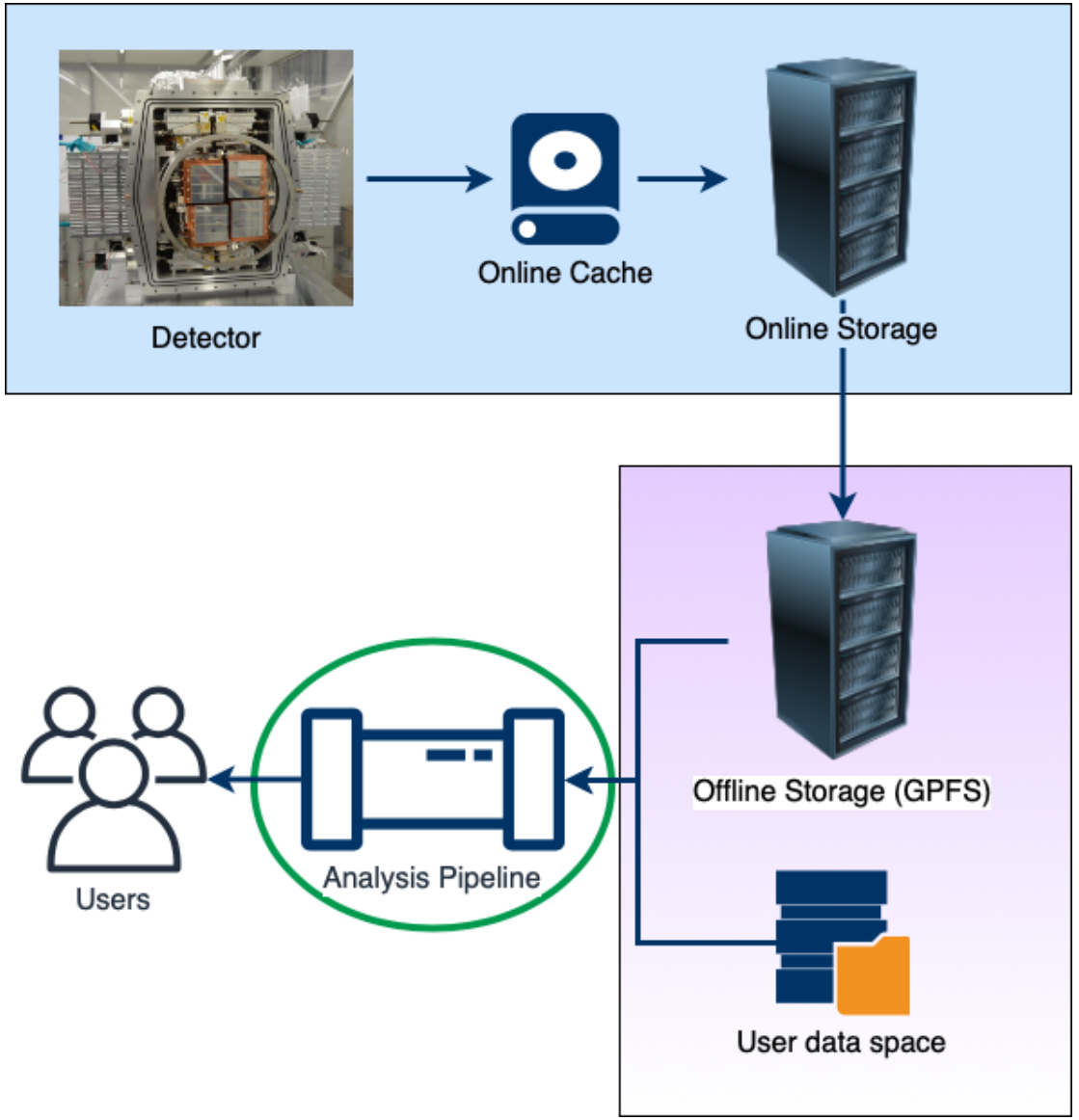
- The goal: Molecule structure at atomic level (1A)
- Big data:
  - 120 Tb per experiment (Dec 2017)
  - 360 Tb per experiment (May 2019)
  - expected to be increased **100x** times!
- Experiments evolve rapidly
- Data Analysis is also under intensive development:
  - Algorithms
  - Software
  - IT services



*Gaffney K. J. & Chapman H. N.// *Science*, 2007.

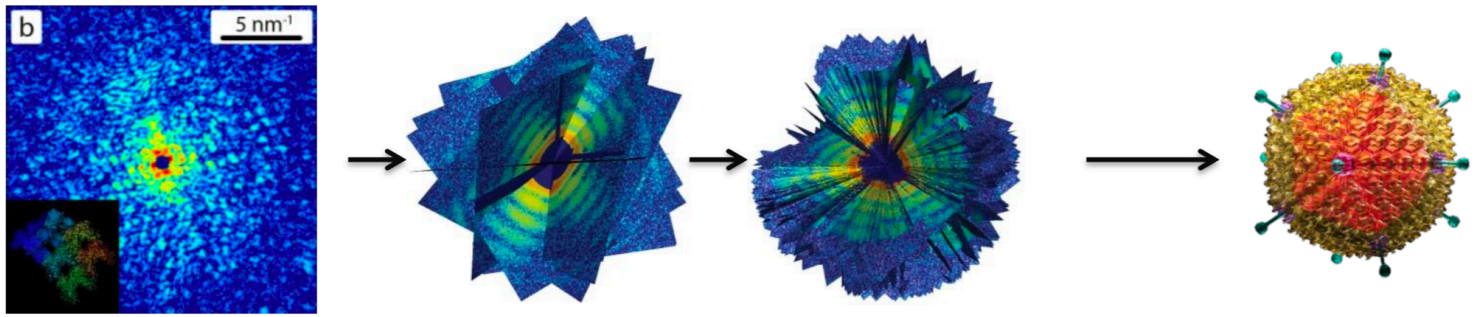
The Goal of the Project

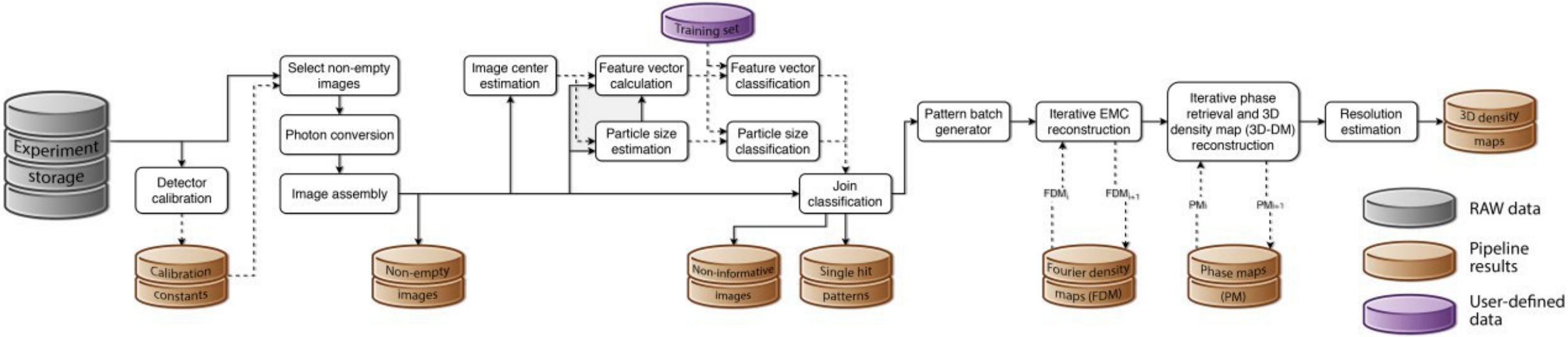
- Software Pipeline for automated data processing
- From diffraction patterns to 3D structure in near real-time
- Core Ideas:
  - Integration of software packages for various stages of data analysis in analysis pipeline
  - Simple configuration and deployment
  - Scalability
  - Extensibility, modular architecture
  - Various workflows



Structure Reconstruction Pipeline

Briefly

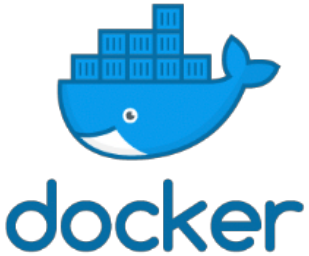


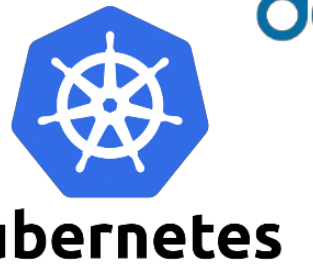


A little bit more detailed

Realization Strategy

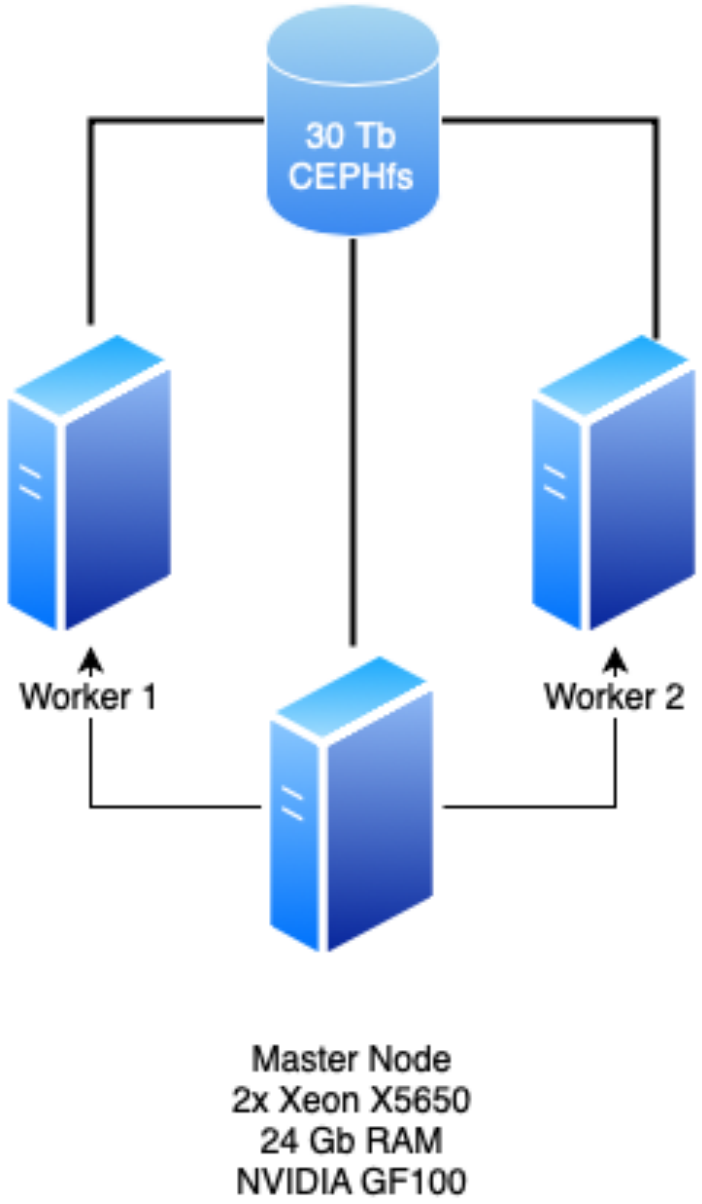
- Container technology for easy deployment
- Microservices for individual stages of analysis
- Container orchestration for scalability and management
- Shared network filesystem to reduce data transfers



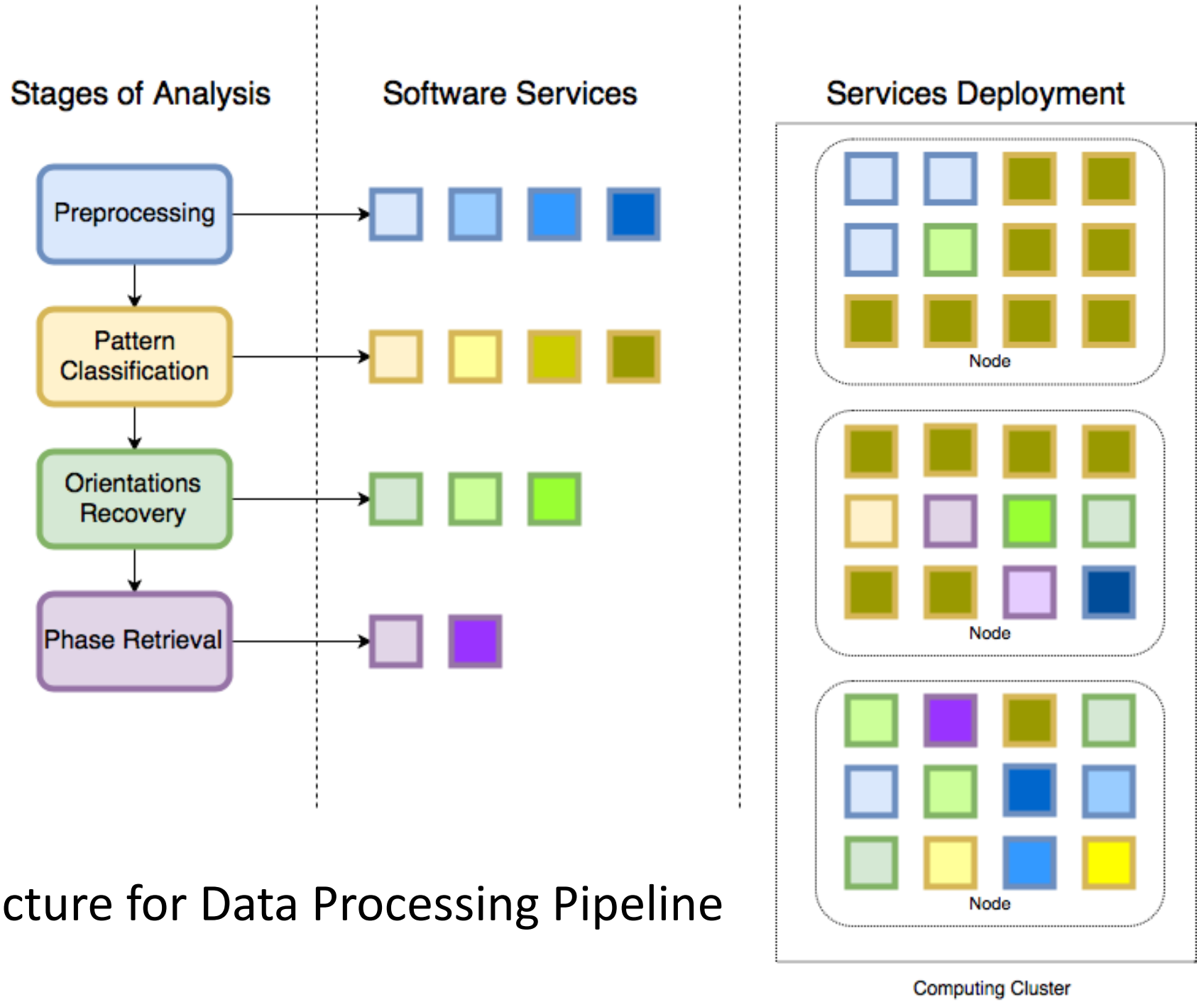


Testbed

- Dedicated K8s cluster (version v1.15.3) with three nodes
- Dedicated CEPHfs storage
- 1Gbps interconnect



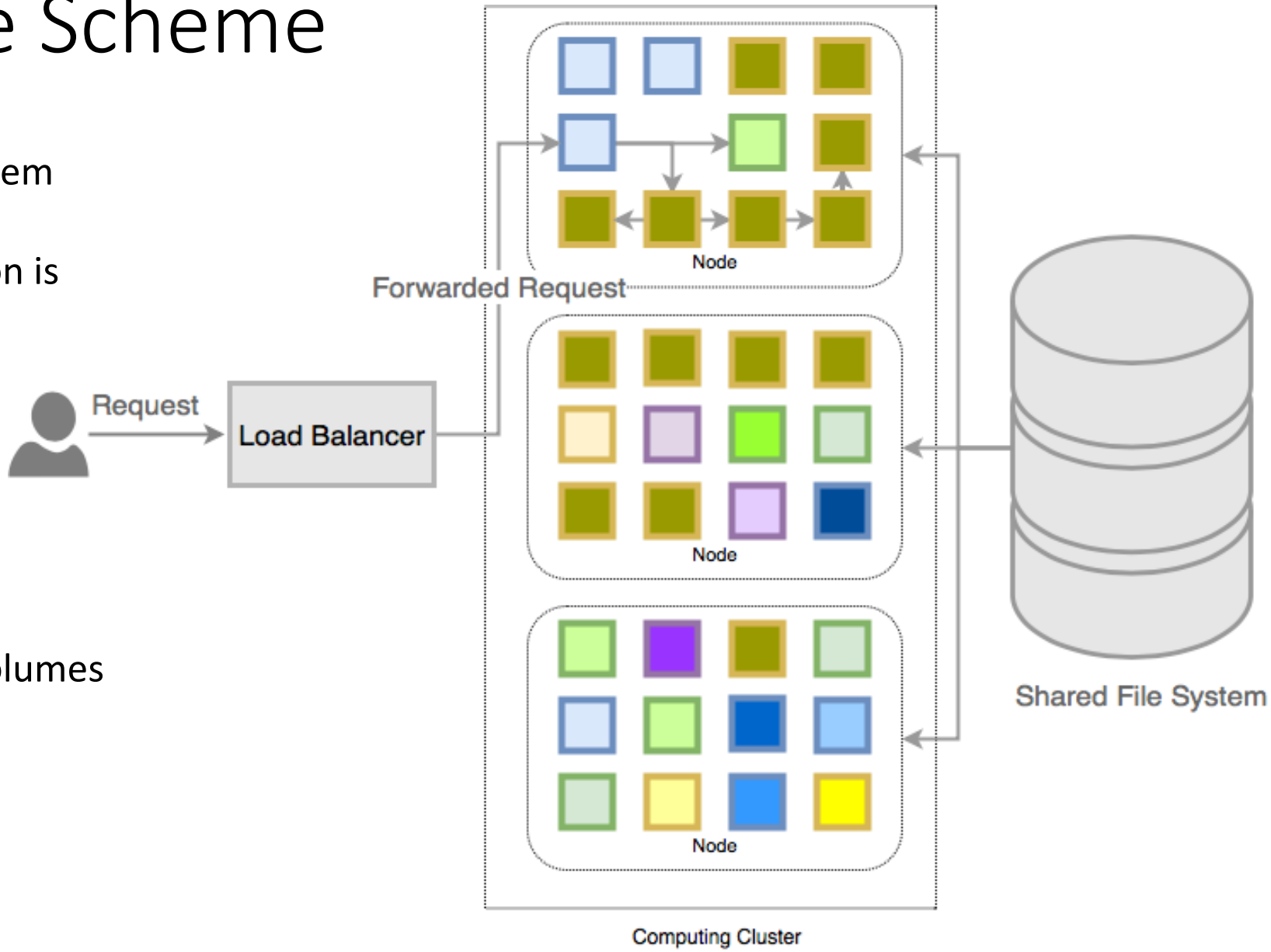
- Split analysis pipeline into small components
- Pack components into containers
- Use container orchestration infrastructure to distribute containers across computing cluster



Microservices Architecture for Data Processing Pipeline

Data Exchange Scheme

- Data is stored in a shared filesystem (GPFS, Lustre, CEPH) in HDF5
- K8s based container orchestration is used for:
  - containers deployment
  - load balancing
  - internal and external communications
  - services monitoring and management
- Native K8s support for CEPHfs volumes



Services and Jobs in K8s

Services model of operation

```
{
    "srand": 1,
    "data": "data/sample.bin",
    "data_dim": "2d",
    "number_of_outputs_images": 10,
    "number_of_outputs_scores": 10,
    "support_algo": "static",
    "support_size": 30,
    "algo": [
        { "name": "hio", "number_of_iterations": 500, "beta_init": 0.9, "beta_final": 0.9},
        { "name": "er", "number_of_iterations": 100}
    ],
    "out_file": "data/sample_out.h5"
}
```

```
root@c003f610f7a5:/opt/wsgi_app# curl  -H "Content-Type: application/json" -d @test.json localhost/phase
[8.3701095581054688, 0.090004101395606995, 0.049716383218765259, 0.12340822815895081, 0.084595672786235809, 0.077844396233558655,
0.065120011568069458, 0.068547621369361877, 0.024556649848818779, 0.024488534778356552]
```

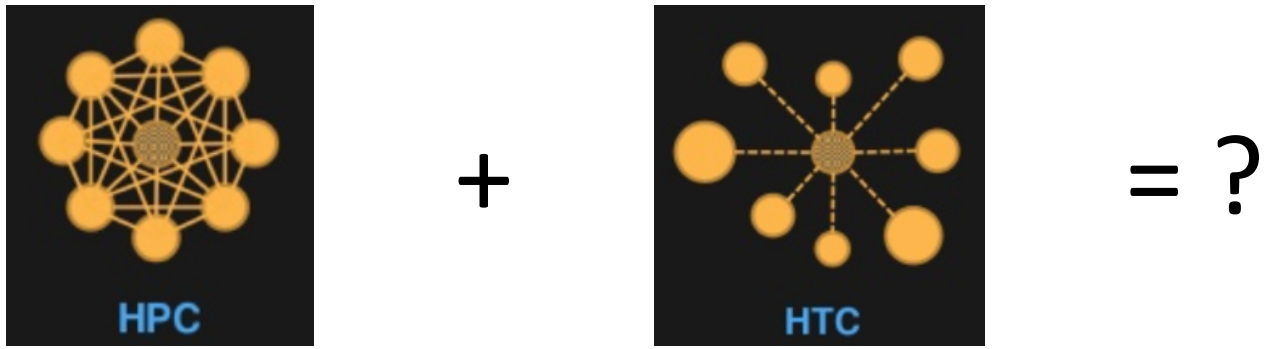
Jobs

```
root@c003f610f7a5:/opt#  kubectl create -f phaser-job.yaml
```

Use Cases: Orientations Determination

- Dragonfly
  - EMC algorithm for orientations reconstruction
  - High quality code
  - MPI
  - GUI interface

- It is the bright case where HPC application meets HTC (Cloud)!



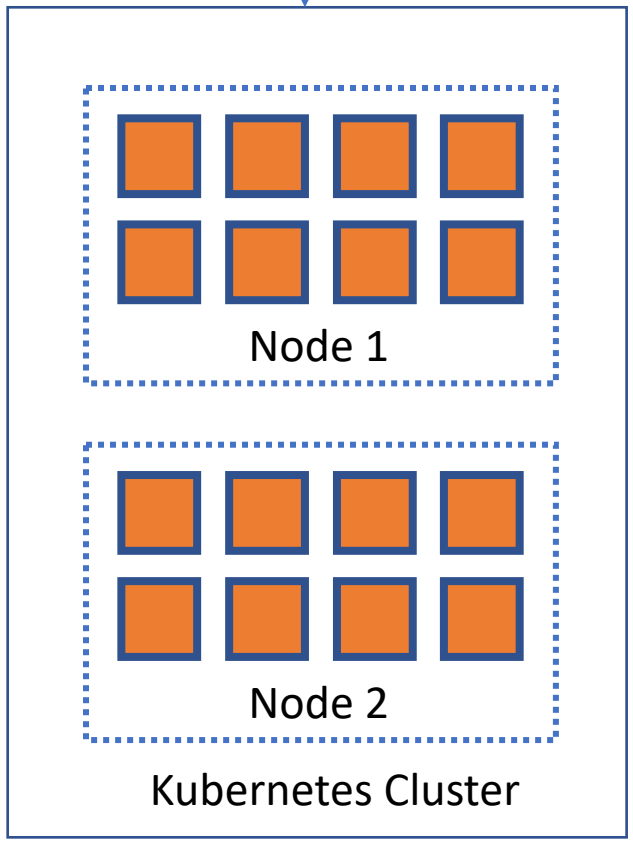
HPC vs HTC

- Different focus, history, architecture, ecosystem
  - HPC – parallel computing. Intensive communications between nodes
  - HTC – data and services centric. Loosely coupled services
- Possible scenarios of combined usage
  - application code refactoring
  - run HTC workloads in HPC systems (Singularity, Shifter)
  - virtualize HPC infrastructure in HTC systems
  - maintain separate infrastructures

Our realization

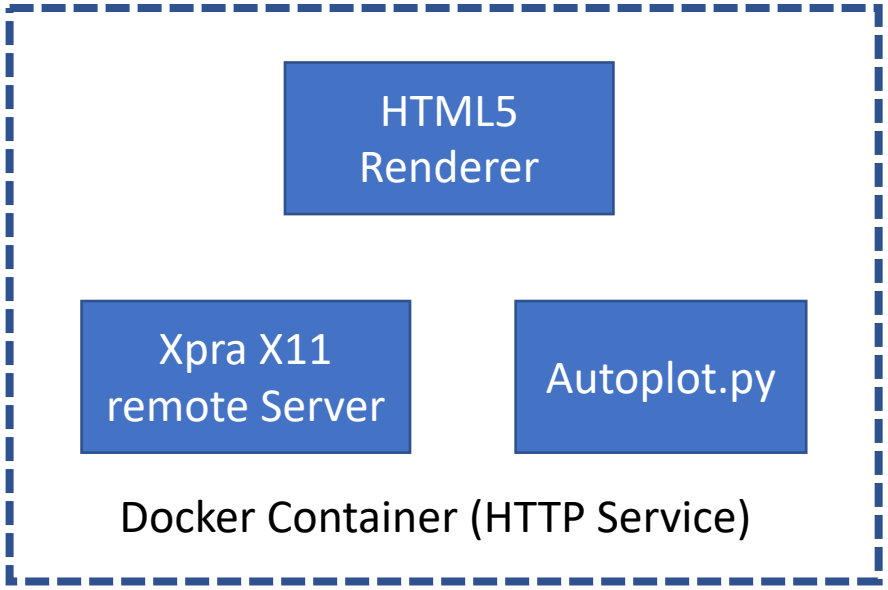
- Kube-openmpi, MPI infrastructure for Kubernetes
  - MPI nodes as Docker containers
  - OpenMPI 2.1.2
- Run Dragonfly as a native MPI application inside virtualized HPC cluster

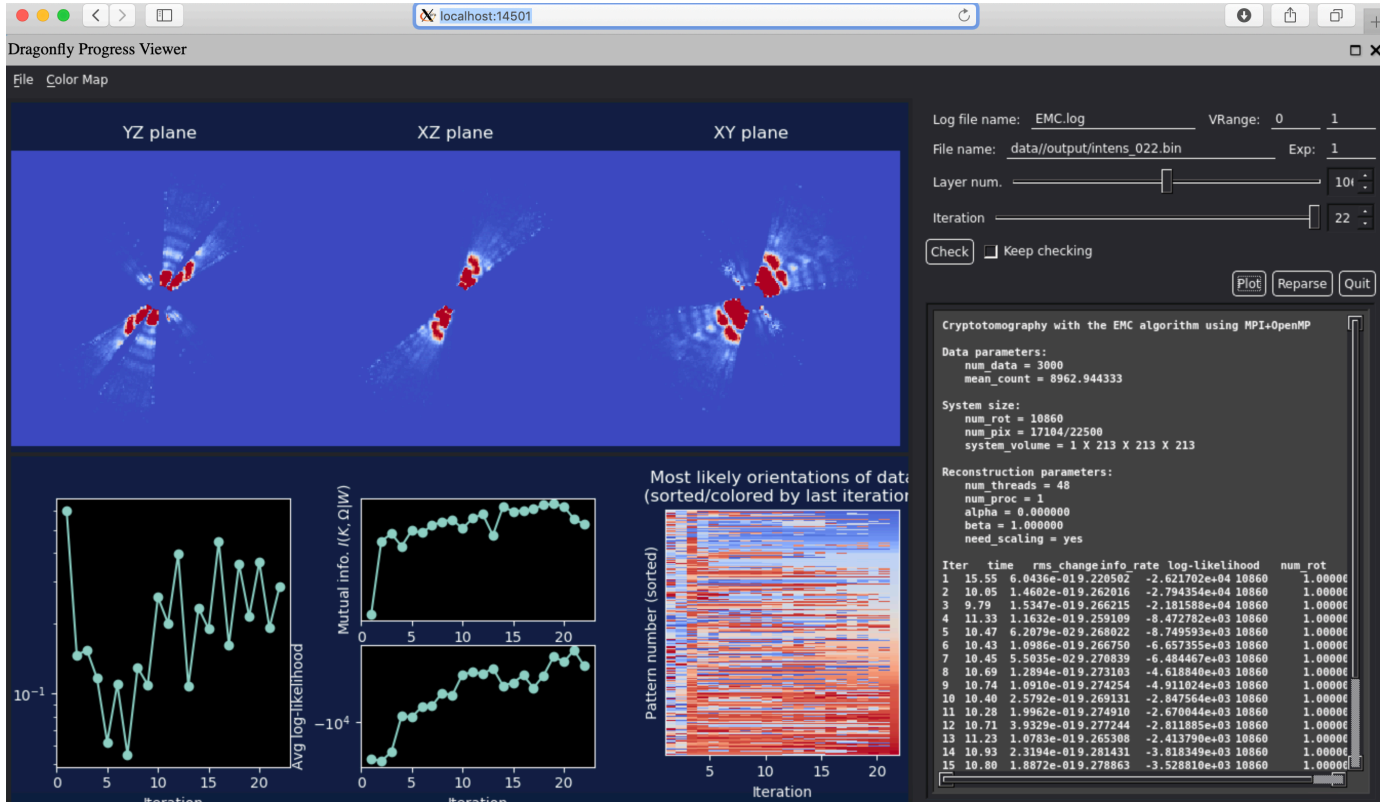
kubectl mpiexec –n 16
./run_emc.py –f emc.ini



GUI applications inside services

- autoplot.py as a HTTP service
- realtime EMC monitoring from browser



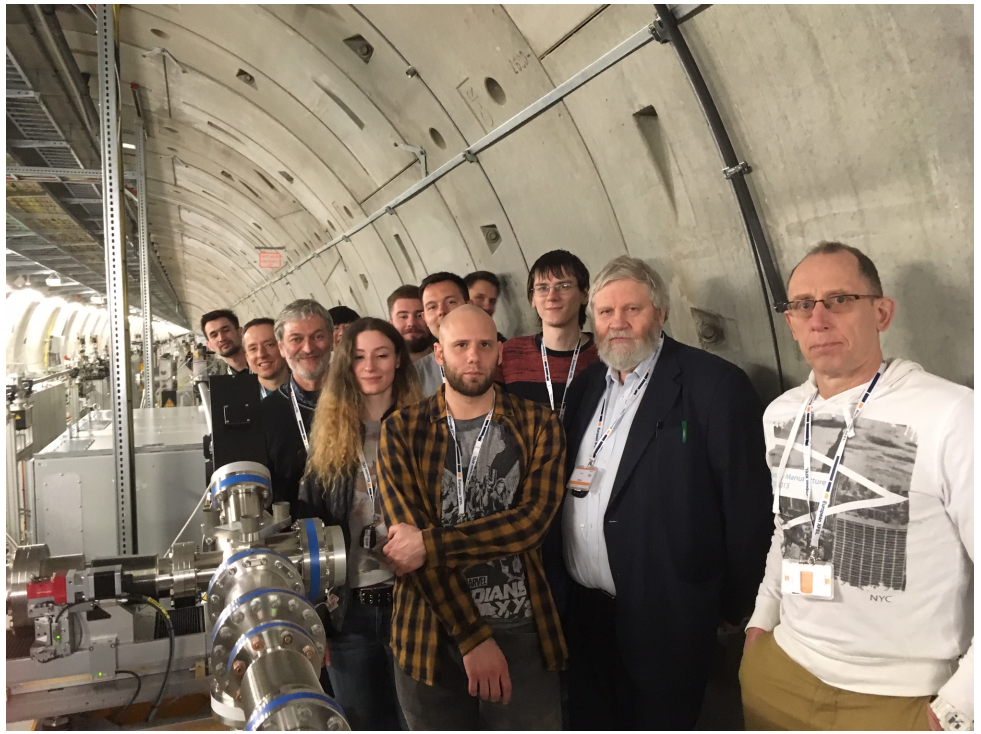


Conclusions and current status

- We have tested existing FEL data analysis software to work in containerized Docker infrastructure. Container technology looks to be effective to integrate heterogenous software with different environment requirements.
- Various patterns of software usage in K8s infrastructure were analyzed and tested: loosely coupled parallel computations, GPU-computing, HPC (MPI) workloads, GUI applications.
- Next step: arrange individual microservices into data analysis pipeline

Acknowledgements

Joined Team from
KI and DESY



Presented results are supported by the Helmholtz Associations Initiative and Networking Fund and the Russian Science Foundation (Project No. 18-41-06001).

Thank you for your attention