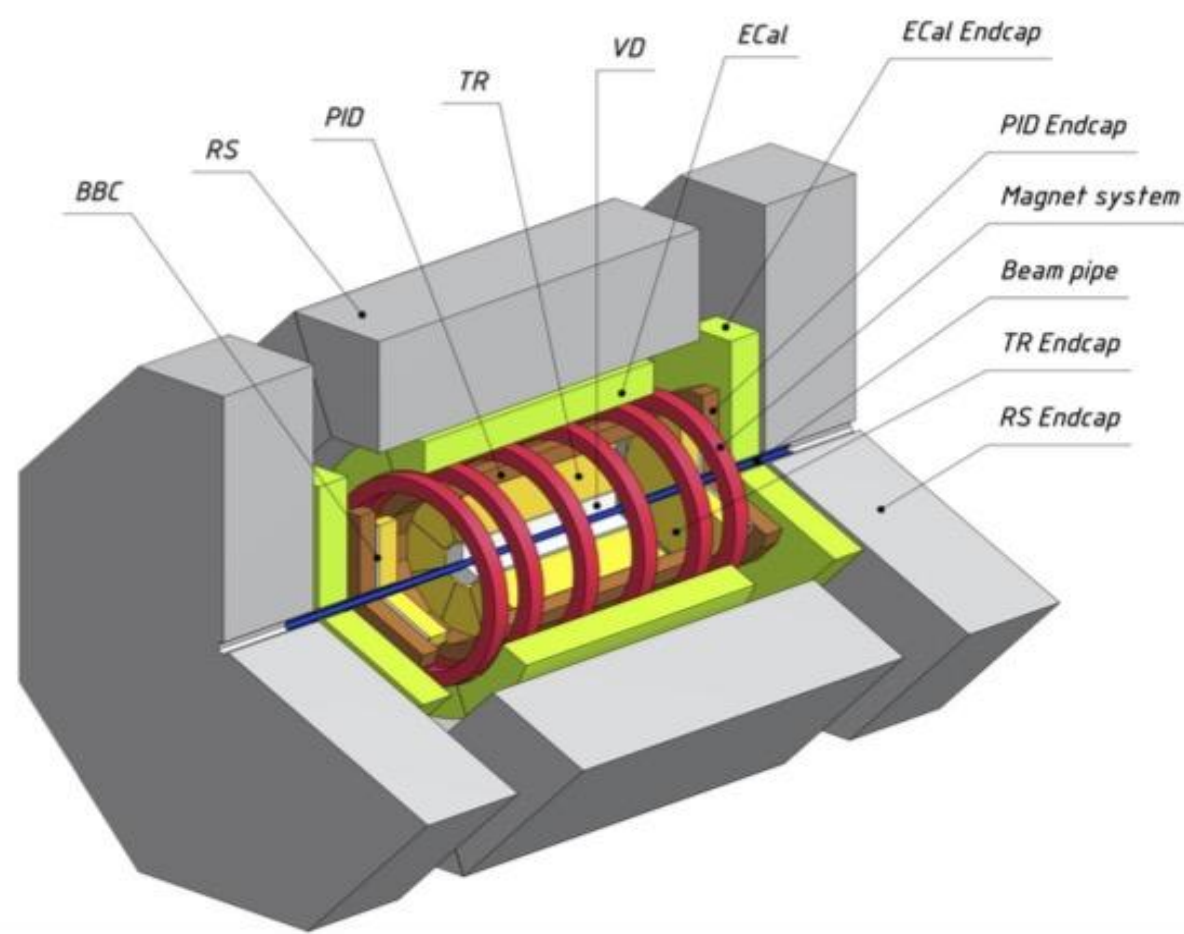


Status of the offline computing system

A. Petrosyan, D. Oleynik, A. Zhemchugov, A. Kiryanov,
A. Zarochentsev

SPD Collaboration Meeting
December 13, 2021

SPD data amount

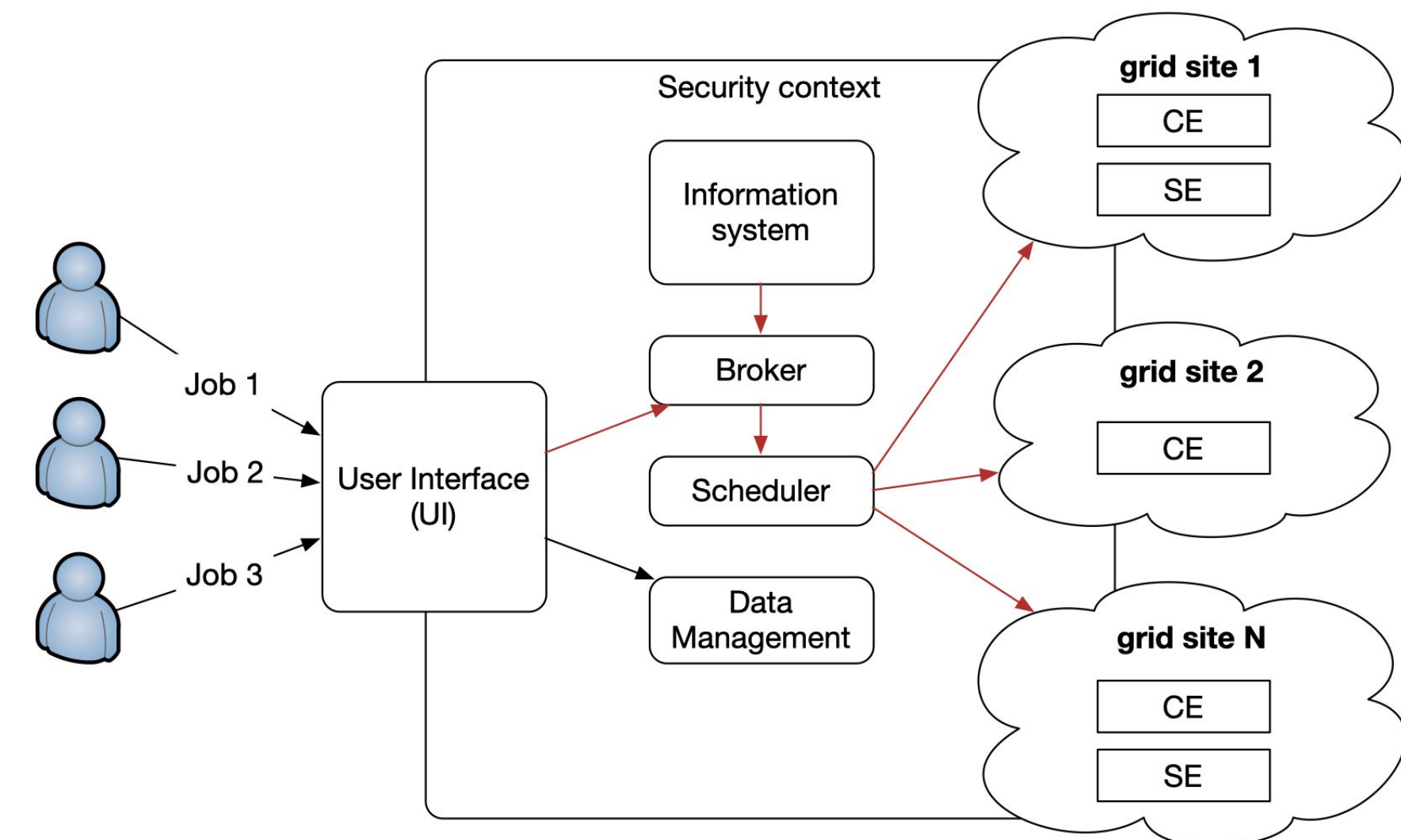


- Bunch crossing every 76.3 ns = crossing rate 13 MHz
 - ~ 3 MHz event rate (at $1032 \text{ cm}^{-2}\text{s}^{-1}$ design luminosity) = pileups
 - **Estimated data rate:** 20 GB/s (or 200 PB/year (raw data), 3×10^{13} events/year) before On-line filter
 - *We still have no full understanding of realistic, expected data rate: frontend electronics and DAQ on initial design step*
 - Required number of simulated events will be in order of magnitude comparable with collected.
-
- The SPD offline data processing system should be able to deal with trillions of events per year.
 - Dozens of thousands of CPUs will be required

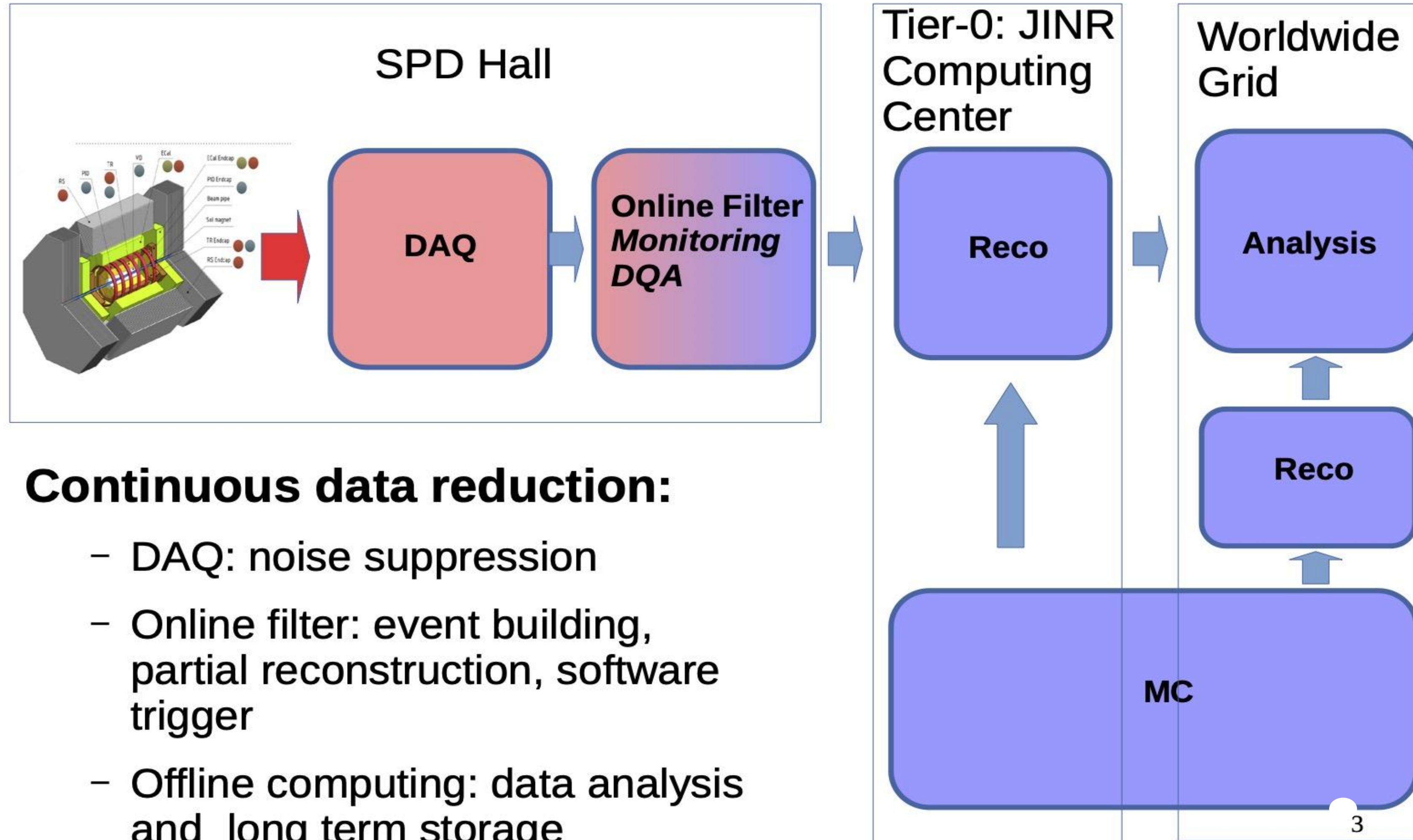
HTC: grid computing

Grid computing is the collection of computer resources from multiple locations to reach a common goal. The grid can be thought of as a distributed system with non-interactive workloads that involve a large number of files. Grid computing is distinguished from conventional high performance computing systems such as cluster computing in that grid computers have each node set to perform a different task/application.

We encourage to organize a GRID computing system for offline processing of SPD data by incorporation of resources of experiment collaborators.



Data workflow

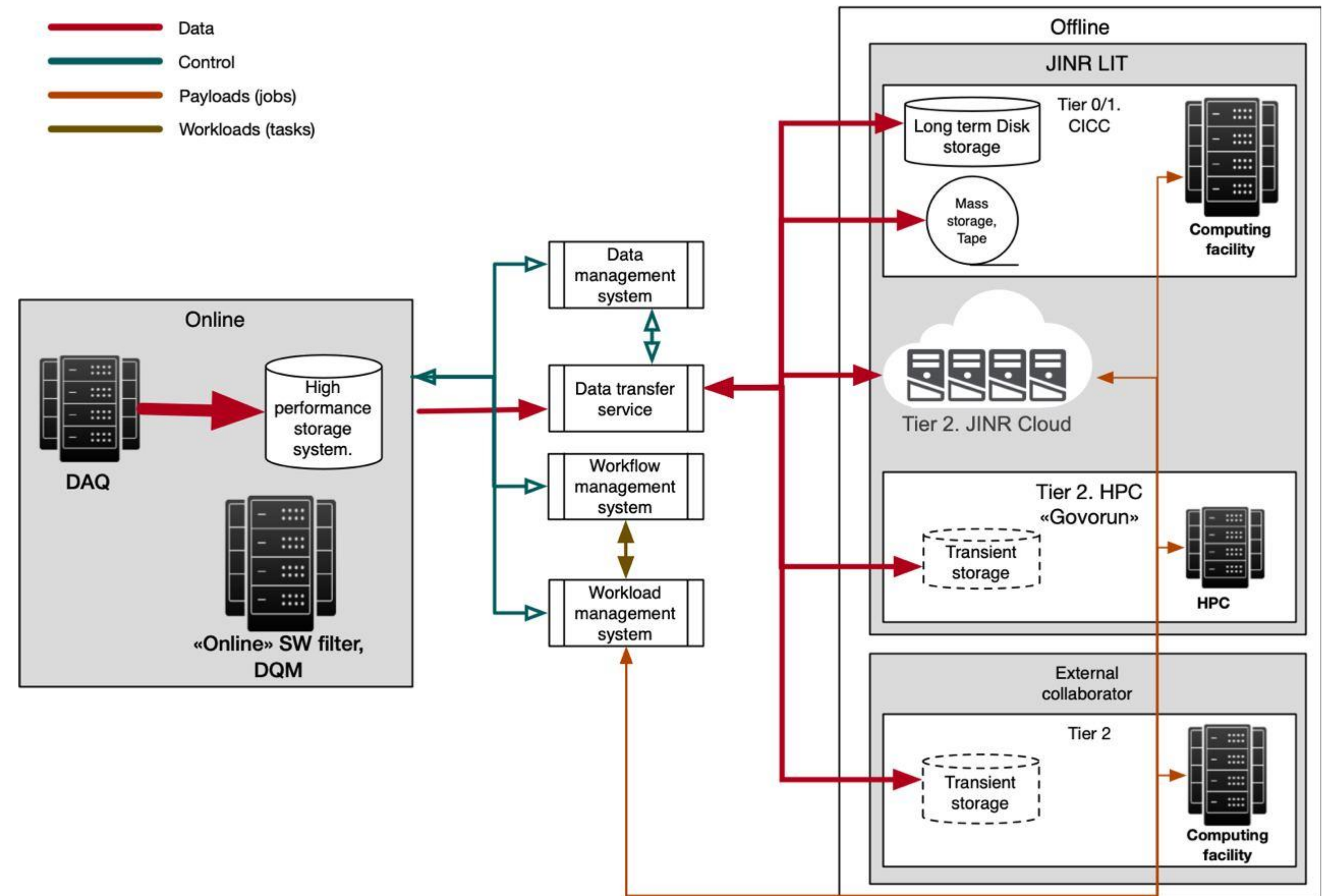


Continuous data reduction:

- DAQ: noise suppression
- Online filter: event building, partial reconstruction, software trigger
- Offline computing: data analysis and long term storage

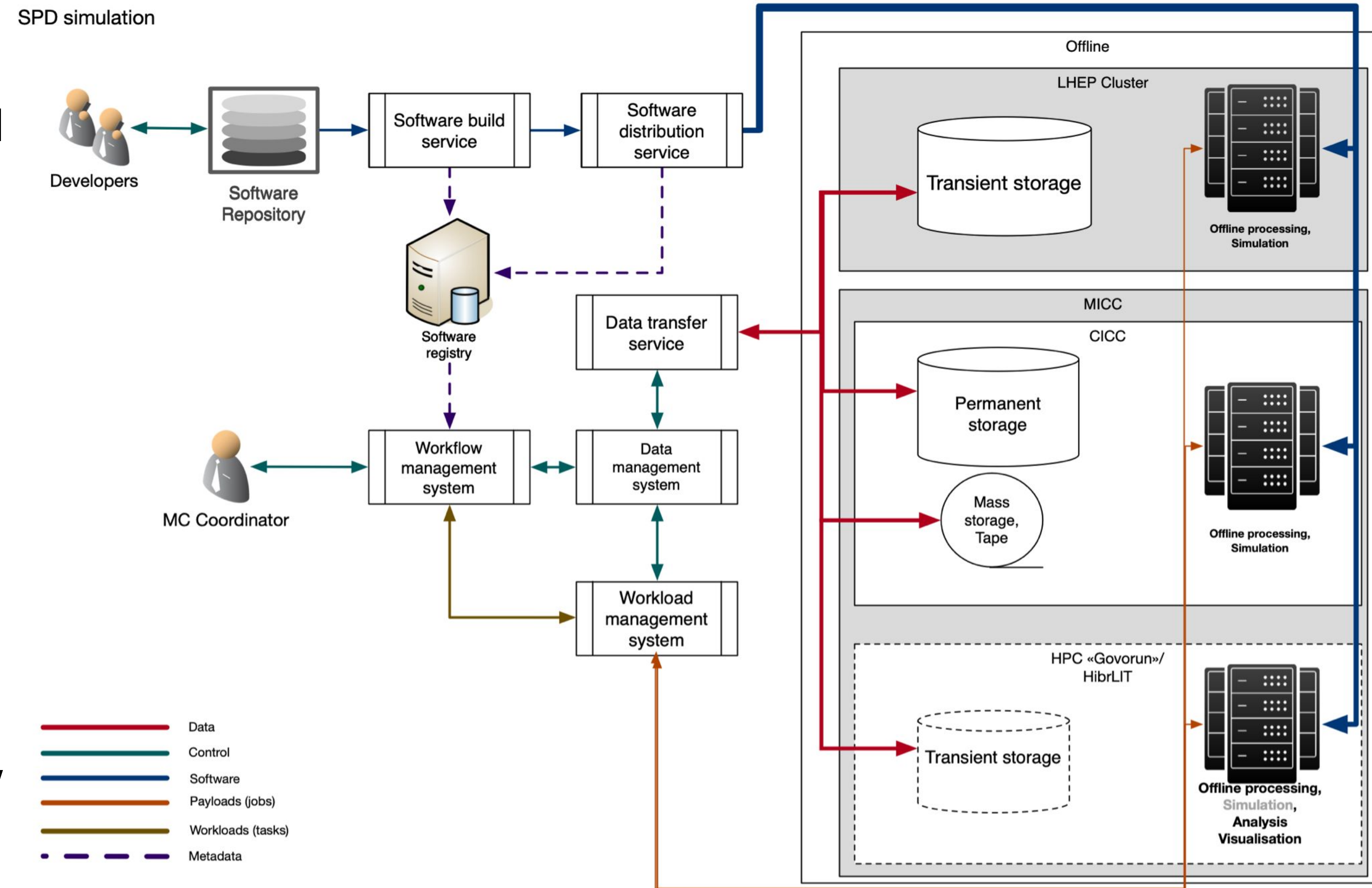
Use case: mass data processing

- The main consumer of distributed computing resources
- It is very important to find the most optimal ratio of the size and number of files to be processed by the system for the most efficient use of the available computing infrastructure
- Key components:
 - WFMS
 - WMS
 - DMS & DTS
 - Software distribution service - service which allow automatic deployment of new versions of SW in heterogeneous environment



Use case: simulation

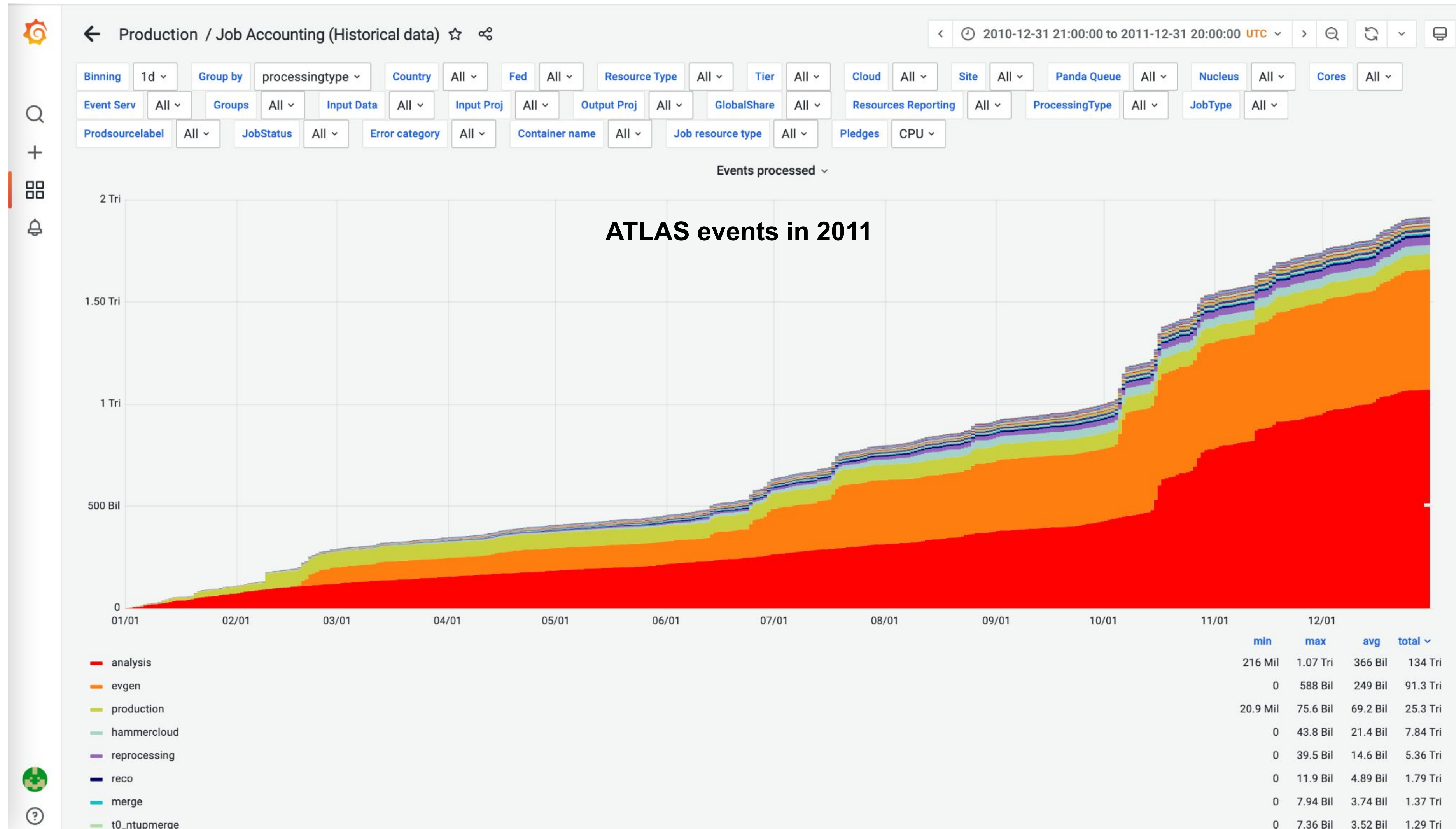
- Simulation – another huge consumer of computing resources
- Can be (should be) started before facility will be ready to collect data
- Accompanied by intensive software development
- Key components:
 - WFMS
 - WMS
 - DMS & DTS
 - Software build service – required for automation of building of new releases of SW
 - Software distribution service - service which allow automatic deployment of new versions of SW in heterogeneous environment



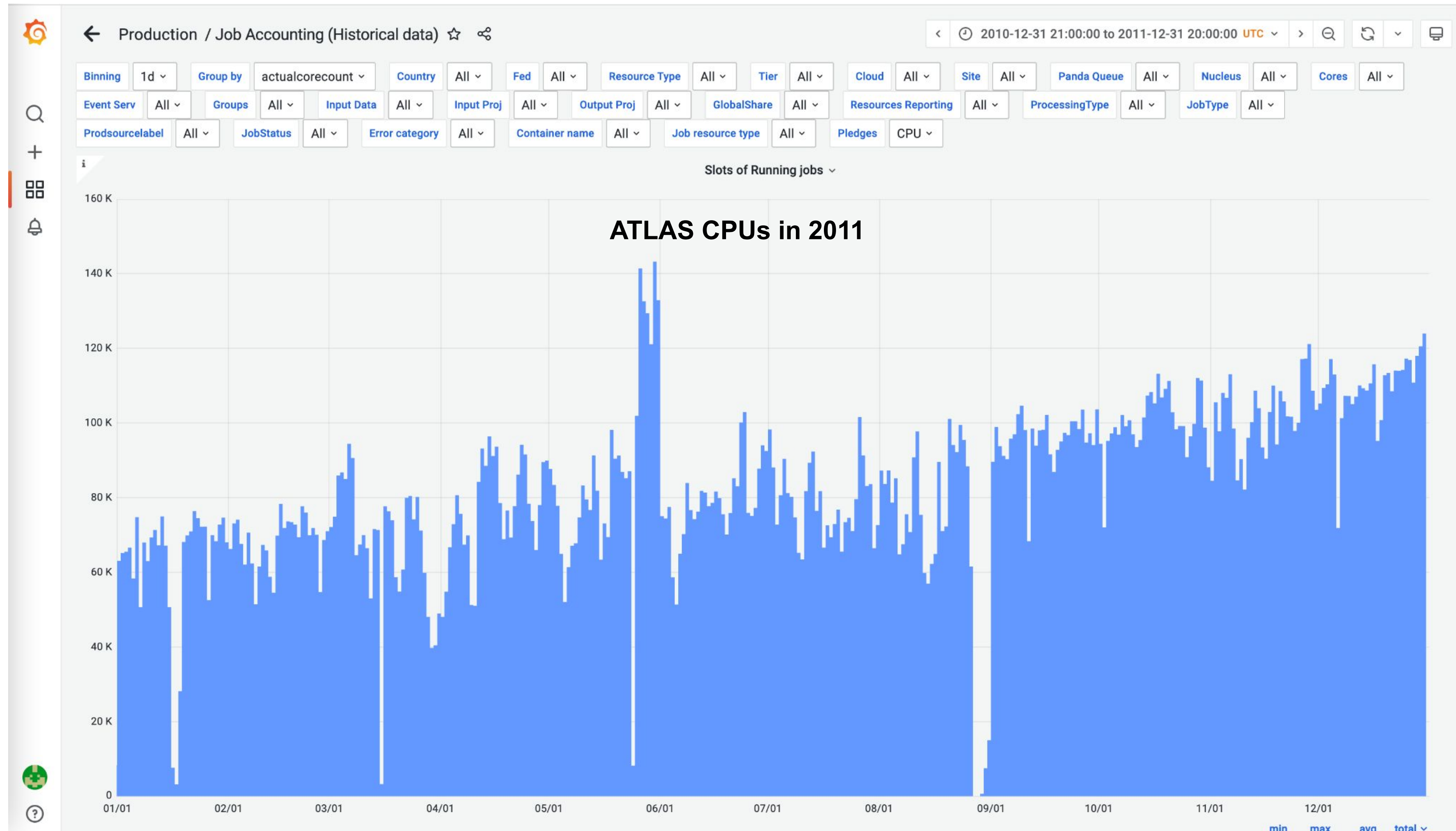
Data volumes in numbers

- Now we expect that facility will generate $2 \cdot 10^{12}$ events per year (EPY)
- We expect to process one event per one second
- There are 31536000 seconds per year (SPY)
- $EPY/SPY = 63419$ fully loaded CPUs
- Now we spend ~5-6 seconds to process one event, it is a crucial point – **serious improvements of the applied software must be done to reduce this number**
- At the moment in LIT we have ~8000 CPU shared between many experiments
- To handle load of such level we need to be ready to process our data at any available computing resource like remote cluster, cloud or HPC

Existing experiments with similar data volumes 1/2



Existing experiments with similar data volumes 2/2



Services of the offline computing system

- Some services, like authorization/authentication, VOMS and CVMFS were already in place and supported by LIT as central and common services for JINR
- Information system (CRIC): deployed, integrated with JINR SSO, being filled with data
- WFMS (Apache Airflow): deployed, integrated with JINR SSO, integrated with WMS
- WMS (PanDA): deployed, integrated with CRIC
- WMS monitoring: deployed, integration with JINR SSO is ongoing
- Pilots delivery to the computing resources (Harvester, HTCondor): deployed, integrated with CRIC, deliver pilots to the computing resources (LIT MICC)
- Pilots, pilot wrappers: container-ready, adopted to our environment, integrated with CRIC, tested on work with JINR EOS, ready to run payloads with real data

NICA CRIC Web UI

The screenshot displays the NICA CRIC Web UI interface. At the top, there is a navigation bar with the following items: Home, Core, Core API, NICA (highlighted), NICA API, Admin, and Logs. On the right side of the navigation bar, there are icons for Help, a user profile (virthead), a search icon, and a power icon.

The main content area features the CRIC logo (Computing Resource Information Catalog) on the left. Below the logo, there is a welcome message: "Welcome to the CRIC Web Portal". To the right of the logo, there is a "Resources" section with a list of links: Federations, Resource C, Service list, and ...

A dropdown menu is open over the "NICA" navigation item, listing the following options: Topology, NICA Sites, PanDA Sites, PanDA Queues, DDMEndpoints, PanDA Queue parameters (with a calendar icon), Blacklisting, PanDA Queue blacklisting, PanDA Queue Status History, PanDA Queue availability (with a calendar icon), DDMEndpoint blacklisting, DDMEndpoint Status History, Operations, Create NICA Site, Create new RSE, Create new PandaQueueStatus, and Create new DDMEndpointStatus.

The main content area is titled "WebUI" and contains a section titled "Operations" with a list of links: Create new RC site, Create new Federation, Create Service, and ...

Apache Airflow Web UI

The screenshot displays the Apache Airflow Web UI interface for a DAG named 'pandaJobsPrototype'. The top navigation bar includes the Airflow logo, 'DAGs', 'Security', 'Browse', 'Admin', 'Docs', and 'About' menus. The current date and time are '2021-12-10, 12:23:38 UTC', and the user is 'Artem Petrosyan'. The DAG is currently 'On' and has a 'schedule: None'.

The toolbar offers various views: Graph View, Tree View (selected), Task Duration, Task Tries, Landing Times, Gantt, Details, Code, Trigger DAG, Refresh, and Delete. Below the toolbar, there are input fields for 'Base date: 2021-11-17T15:32:45Z' and 'Number of runs: 25', with a 'Go' button.

A legend for task statuses is provided: success (green), running (light green), failed (red), skipped (pink), upstream_failed (orange), up_for_reschedule (cyan), up_for_retry (yellow), queued (grey), and no_status (white). The selected operator is 'PythonOperator'.

On the left, a tree view shows the DAG structure with tasks: [DAG], init_job, 4_job, 2_job, 0_job, 1_job, and 3_job. On the right, a Gantt chart shows the execution timeline from Nov 07 to Nov 14. The chart consists of seven rows of colored blocks representing task execution. The top row shows a sequence of green (success) and red (failed) blocks. The second row is entirely green. The third and fourth rows show a mix of white (no_status) and green blocks. The fifth and sixth rows show a mix of green and red blocks. The seventh row is entirely green.

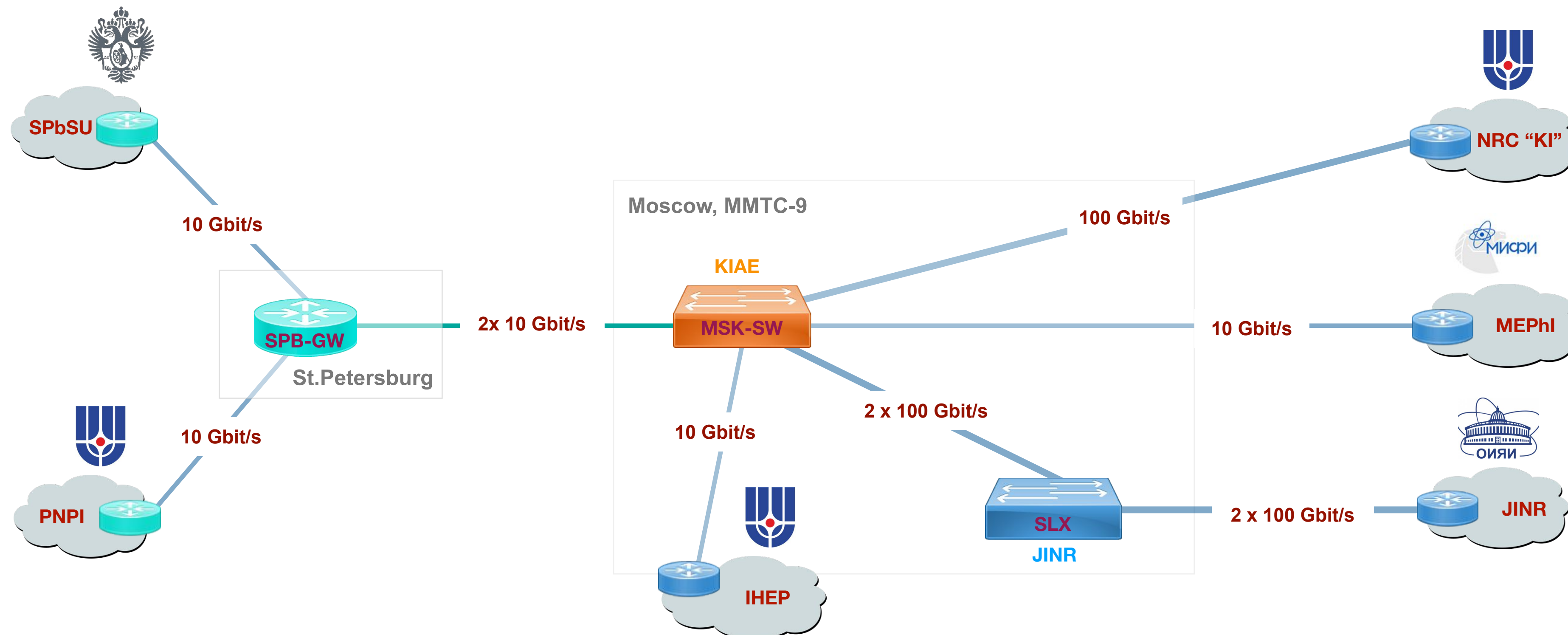
Distributed computing for HEP in Russia

JINR, PNPI, SPbSU, IHEP – already have experience of supporting own Data Processing Centers and participation in the distributed computing for LHC

- *JINR – Data Center for LHC with ~23000 CPU and 25PB Disk storage and 55PB Tape storage*
- *PNPI – Data Center for own experiments and LHC with ~15000 CPU and 5Pb Disk storage*
- *SPbSU – Tier2 ALICE Computing facility*
- *IHEP – Tier2 WLCG site*

Russian network infrastructure for Megascience projects

JINR, PNPI, SPbSU, IHEP data centers already settle high throughput network, which is one of the keys for creation of distributed system



Status and plans

- A prototype, based on the components, which have proven their ability to handle expected data volumes, is ready for real data
- We're ready to start implementing workflows
- Integration with Korolev HPC at Samara University now is in technical stage: we're working on the resource
- Several institutes, like PNPI and SPbSU, are considering the possibility of participating in data processing
- We're looking forward to integrate external computing and storage resources, which members of our collaboration will be ready to provide

Thank you!

PanDA Workload Management System

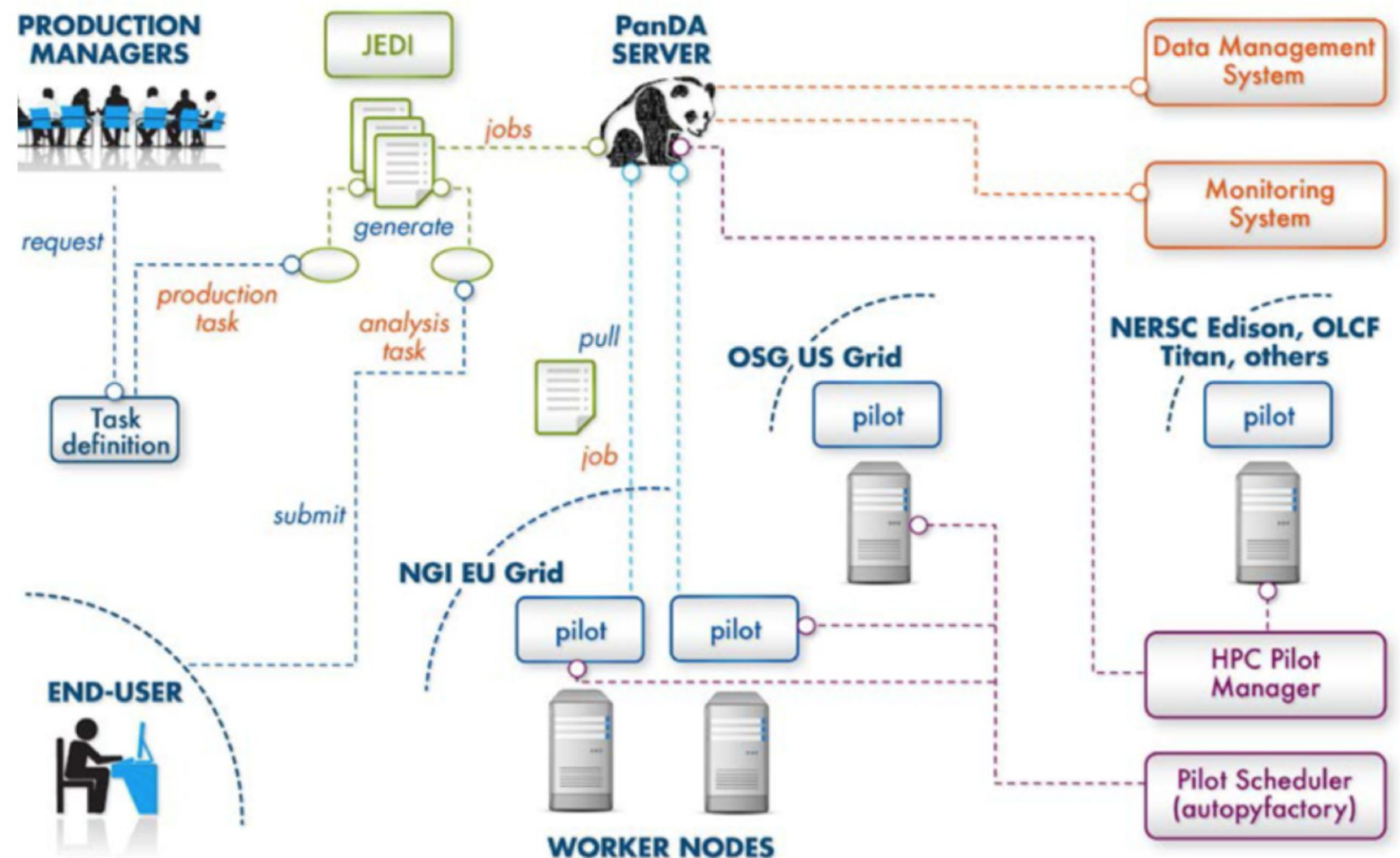
- The PanDA workload management system was developed for the ATLAS experiment at the Large Hadron Collider
- A new approach to distributed computing
 - A huge hierarchy of computing centres and opportunistic resources working together
 - Main challenge – how to provide efficient automated performance
 - Auxiliary challenge – make resources easily accessible to all users
- Core ideas :
 - Make hundreds of distributed sites appear as local
 - Provide a central queue for compute jobs – similar to local batch systems
 - Reduce site related errors and reduce latency
 - Build a pilot job system – late transfer of payloads
 - Crucial for distributed infrastructure maintained by local experts
 - Hide middleware while supporting diversity and evolution
 - PanDA interacts with middleware – users see high level workflow
 - Hide variations in infrastructure
 - PanDA presents uniform ‘job’ slots to user (with minimal sub-types)
 - Easy to integrate grid sites, clouds, HPC sites ...
 - Data processing, Production and Analysis users see same PanDA system
 - Same set of distributed resources available for different processing types
 - Highly flexible – instantaneous control of global priorities by experiment

PanDA Workload Management System

- The PanDA Production ANd Distributed Analysis system has been developed by ATLAS since summer 2005 to meet ATLAS requirements for a data-driven workload management system for production and distributed analysis processing capable of operating at LHC data processing scale. ATLAS processing and analysis places challenging requirements on throughput, scalability, robustness, efficient resource utilization, minimal operations manpower, and tight integration of data management with processing workflow.
- PanDA throughput has been rising continuously over the years. In 2009, a typical PanDA processing rate was 50k jobs/day and 14k CPU wall-time hours/day for production at 100 sites around the world, and 3-5k jobs/day for analysis. In 2017, PanDA processes about a 1,2M jobs per day, with about 120,000 jobs running at any given time. The PanDA analysis user community numbers over 1400.

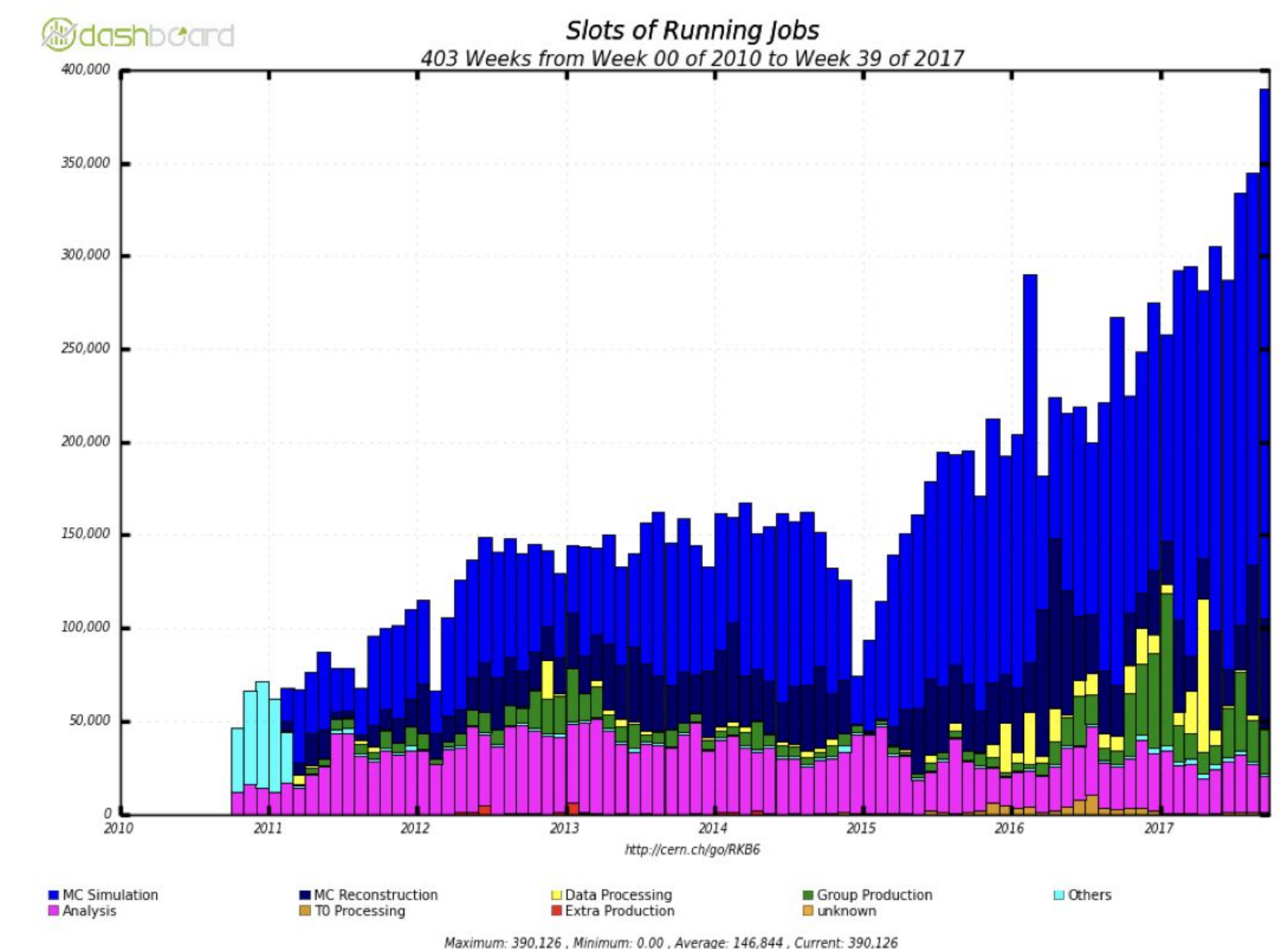
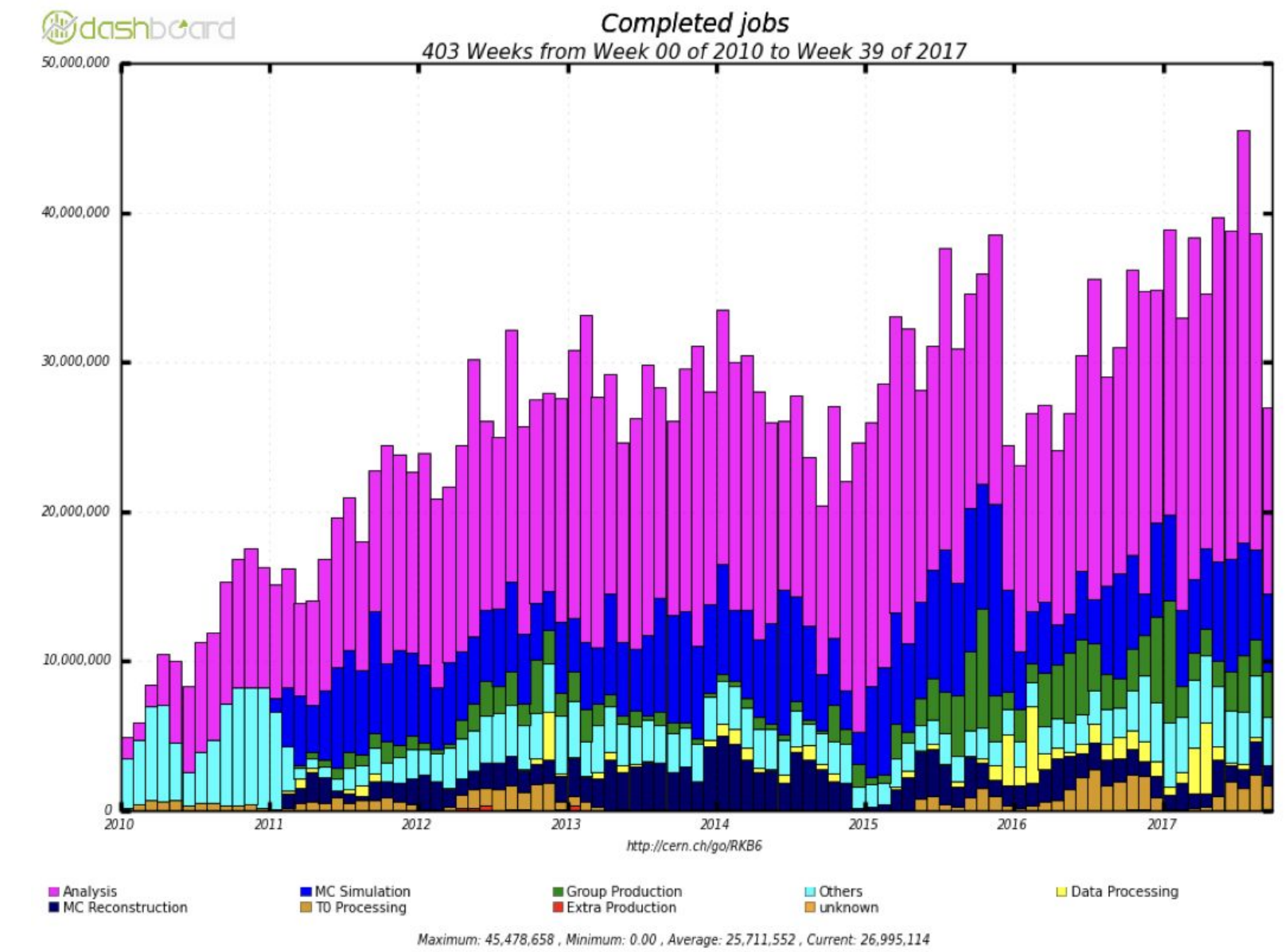
PanDA at nutshell

- Pilot based job execution system
- Pilot manages job execution on local resources, as well as data movement for the job
- Payload is sent only after pilot execution begins on Compute Element
- Minimize latency, reduce error rates



Brief PanDA History

- 2005: Initiated for US ATLAS (BNL and UTA)
- 2006: Support for analysis
- 2008: Adopted ATLAS-wide
- 2009: First use beyond ATLAS
- 2011: Dynamic data caching based on usage and demand
- 2012: ASCR/HEP BigPanDA project
- 2014: Network-aware brokerage
- 2014: Job Execution and Definition I/F (JEDI) adds complex task management and fine grained dynamic job management
- 2014: JEDI-based Event Service
- 2014: megaPanDA project supported by RF Ministry of Science and Education
- 2015: New ATLAS Production System, based on PanDA/JEDI
- 2015: Manage Heterogeneous Computing Resources
- 2016: DOE ASCR BigPanDA@Titan project
- 2016: PanDA for bioinformatics
- 2016: COMPASS adopted PanDA (JINR, CERN, NRC-KI), PanDA beyond HEP : LSST, BlueBrain
- 2017: PanDA instance at OLCF: PanDA for IceCube



Computing Resource Information Catalog (CRIC)

CRIC is the framework to design high-level information systems which describe the topology of the Computing model(s), providing unified description of resources and services used by Experiment applications.

CRIC was chosen to realise the information system for SPD Offline computing infrastructure