SPD Information systems

Current status

SPD Information Systems

- Hardware Database 
- Mapping Database 
- Conditions and Calibration Databases
- Physics Metadata Databases
- Event Index 
- Distributed Computing Information System 
- Distributed Data Management 
- Monitoring information system
- Logging and Bookkeeping
- Personal and publication databases

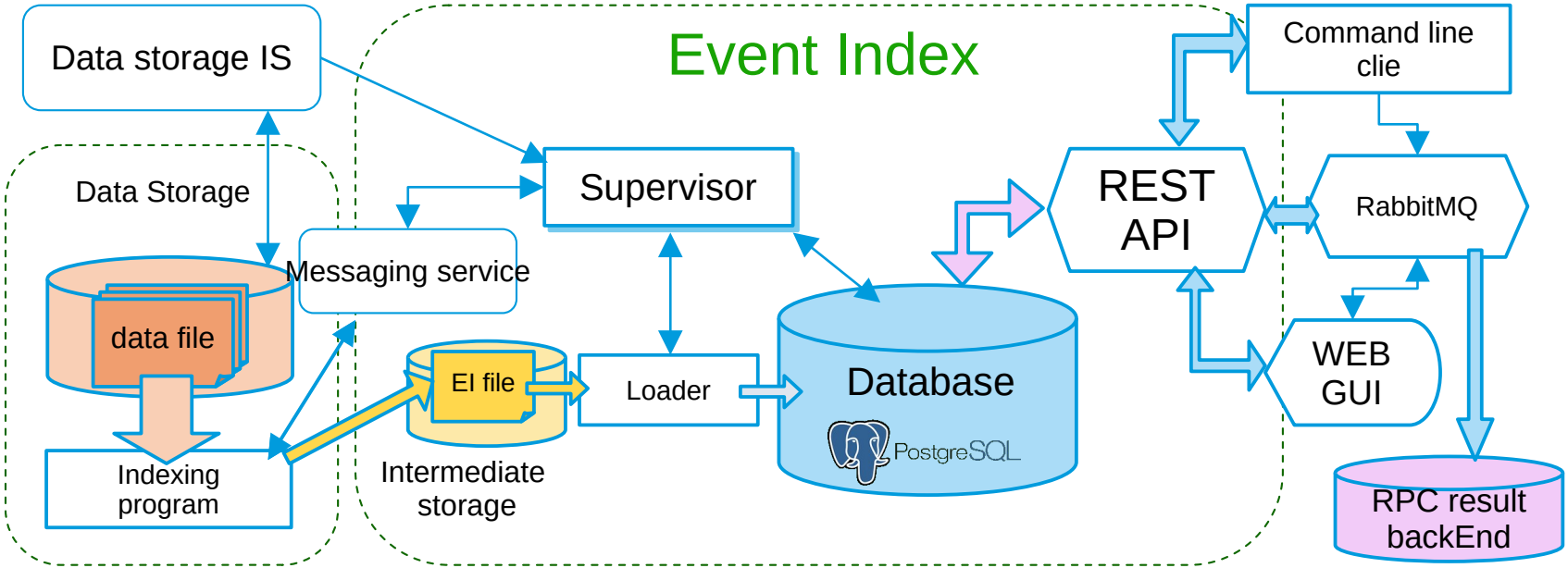
Event Index

Event Index

- EventIndex is a system designed to be a complete catalog of SPD events, real and simulated data
- SPD EventIndex is being developed as a comprehensive information system that should provide:
  - obtaining information about experimental events and simulated data by indexing data files containing information about these events;
  - transfer of this information and write to databases;
  - access to information for data processing and analysis programs via API and applications;
  - access to information to users through interactive and asynchronous interfaces.

Architecture

General architecture: components and data flow



Event Record

- An entry for an event in EventIndex contains the following fields:
  - event IDs: Run number (run_number) and event number (event_number)
  - information about online filter decisons, encoded as bit mask (olf_result)
  - unique identifier of the RAW data file containing this event (fuid_raw). Using this UUID, you can access it through a distributed storage system.
  - ID of the dataset this file is included in (dsid_raw)
- As the data is processed, new instances of the recovered events will be created in a format optimized for physical analysis (AOD).
  - Pointers to different versions of such files will be added to the event record.
- Also, important event parameters can be added to the record, which can be used for classification and selection.

Backend

- The choice of a platform for data storage and management was made based on the expected flows and volumes of information, the content of the record and the expected use cases
- The estimated data flow at the output of the online filter:
  - from tens thousand events per second in the early stage
  - up to 150 thousand at maximum machine performance,
  - from hundreds of billion (10^{11}) to few trillion (10^{12}) events per year
- PostgreSQL DBMS is used for storage and processing of data:
  - ability to process large amounts of data, multithreading
  - high performance in data ingestion by support of bulk loading
  - open source, widely used, employed in other IS of NICA experiments

Front-end

- A convenient and efficient program interface was developed and implemented that performs data exchange using the RESTful API.
- The frontend part of the client interface was developed using the Angular framework
  - Provides tools for creating modern dynamic user interfaces, as well as provides effective interaction with the server and data manipulation.
  - An Angular application is built from a set of components, each representing a specific part of the user interface.
  - Components can be nested into each other, forming a hierarchy.
  - Each component includes an HTML template for displaying the user interface, TypeScript code for logic and data structure, as well as CSS styles for appearance.

Server side

- A microframework for the Python language, Flask, was chosen for the server side for the first prototype
- Further studies lead to the using of other solution, fastAPI: a light weighted asynchronous RESTful framework for Python
  - Fast, highly flexible, well documented
  - Automatic Open API documentation from the box
  - Asynchronous. Fast API uses ASGI servers by default
- For asynchronous task processing RabbitMQ and Celery were used to improve system performance.
  - RabbitMQ is a message broker that allows you to send, receive and route messages between application components asynchronously.
  - Celery is a system for performing operations in the background.

Interaction between components

- When a user interacts with a web application, for example, requests to receive fuid_raw after entering data parameters, the following happens:
  - FastAPI accepts a request from the user and then passes it to the RabbitMQ message queue.
  - Since Celery works asynchronously and is designed to perform tasks, it retrieves the task from the RabbitMQ queue and performs the necessary operations to get fuid_raw from the database.
  - After successful completion of the task, Celery sends the result (for example, fuid_raw) back to FastAPI, which can send it to the Angular application in response to a user request.
  - Further, the queue in which it was located can self-destruct, that is, the queue can be automatically deleted or released from the task.

Interaction between components

- Thus, an efficient asynchronous system is created in which FastAPI sends tasks to the RabbitMQ queue using Celery for asynchronous execution.
- This allows you to process many requests from users without blocking the application and provides scalability.

- The whole stack has been assembled and the request sequence was tested for the picking of single event
- Further studies underway

Data loading performance studies

- To test the prototype of the system, sets of generated intermediate Event Index data are created.
- The format of these sets is JSON, and it does not depend on the format in which the data from the detector will be stored.
- For each event, identifiers are generated (*run_number* and *event_number*), a random *of_result*, as well as *fuid_raw* and *dsid_raw*.
- This data is then written to tables in the database: event records are stored in the “events” table, and information about datasets is stored in the “datasets” table. The dataset ID serves as a foreign key for the event table.

Data loading performance studies

- The procedure used in the first version of the interface for recording each event with a separate PUT instruction showed insufficient speed when testing even on relatively small data arrays.
  - For example, 1000 events, on average, are recorded for 1 minute 55 seconds.
- A system capable of coping with a flow of tens of thousands of events per second is needed.
- To solve this problem, various optimization methods were investigated to speed up the process of writing data to tables..

Testing DB interface modules

- Along with using the common psycopg2 driver for interacting with PostgreSQL, the following modules were tested: asyncpg, pg8000, PyGreSQL, SQLAlchemy.
- To further optimize the data write speed, a COPY query was used instead of an INSERT query.
  - Using COPY accelerated the writing of data to the table by about 2 times.
  - When working with a small number of events (from 10 to 100000), using the PyGreSQL module is effective
  - For a large number of events (from 1000000), the asynchronous asyncpg module should be used.
- The loading time for the 10 million sample is less then 6 minutes

Further performance studies

- Work is underway to optimize the data write speed by changing PostgreSQL parameters, the effect of block size on download speed is being investigated.
  - The system tries to load all the data at once, but at high load it adapts using a cluster approach for efficient data processing.
- Additionally, the possibility of parallel data loading is being considered to improve performance.
- Currently use of special files for bulk loading is investigated
  - Data from the JSON file are ceonverted to the temporary file that is loaded to the tables and then cleaned
  - Only the slight pereformance increase detected
- It should be noted that performance can be limited by the VM

Further development

- In the course of further development of the Event Index project, the following tasks are expected to be performed:
  - Development of user authorization and authentication mechanisms using single sign-on technology and group access policies;
  - API development (REST, Python, C++, ...);
  - Optimization of user request processing, with synchronous or asynchronous output of results, depending on the volume of requested data;
  - Development of mechanisms for transmitting “EventIndex” data obtained by indexing files located on remote nodes of a distributed computing network;
  - Development a supervisor - software for managing, collecting and importing data into “EventIndex”;
  - Development of an EventIndex component monitoring system, with graphical representation of data based on popular platforms (Grafana, etc.).
- The implementation of these tasks will be carried out in parallel with the development of other Information Systems of the experiment.

Hardware and mapping database

Hardware Database

- A catalog of hardware components that SPD detector consist of.
- It should contain the information about the detectors and the electronic parts, cables, racks, and crates, as well as the location history of all items
- It include equipment models, provider, parameters and other (semi)permanent characteristics
- This should help in maintenance of the detector systems and especially helpful in knowledge transfer between team members.

Hardware ID

- An unique identifier (Hardware ID, HWID) is assigned to each device
  - HWID identifies a specific instance of the device, if the device is replaced, its HWID will change
  - The values of the parameters and characteristics of this particular device (s/n, threshold values, bin length) are associated with the specific HWID
  - Label with the HWID value is placed on the surface of the equipment, in the form of a number and a bar code
  - For devices connected to the DAQ system, it is desirable to implement an automatic system that allows you to receive HWID in response to a special control signal
  - The HWID ranges are distributed between subsystems, the HWIDs subsystem components are selected within these ranges

Type ID

- Each type of device is assigned a unique identifier (Type ID), which is the key of the type table.
  - TypeID defines the type of homogeneous components with the same set of parameters and characteristics
  - For each type of device, a set and type of parameters are defined that are common to all devices of this type, as well as acceptable values or intervals
  - Parameter values are determined separately for each device (in general).
  - Parameter values that are common to all devices of this type can be specified in the type table
  - The parameter sets are individual for each device type.

Record schema

- The schemes for device records are determined by device type and are created automatically based on information on that type
- Two ways of implementing different schemes for different data types were considered:
  - To create separate tables for each type of equipment.
  - To organize parameters in JSON structures that are stored in universal tables containing records of different types
- The second solution was chosen, as the selected DBMS (PostgreSQL) allows handling of JSON structures,

Tables schema

HWID	type	S/N	Name	state	parameters
0007-015b2e3488ac	04fd15c3	PG1342		OK	{JSON}
0007-015b2e3488ad	04fd15c3	PG1344		FUBAR	{JSON}
0007-01fe47adf301	0368eba1	164756		FU	{JSON}
0007-01fe47adf301	0368eba1	164756		OK	{JSON}

```
{
  ...
  "ov": "37",
  "dcr": "253",
  ...
}
```

TypeID	Name	Description	parent	parameters
04fd15c3	PG2T390	PANGOMICRO Titan-2 PG2T390 FPGA Board	000013f0	{JSON}
0368eba1	EQR151110	EQR15 11-1010D-S SiPM	00001134	{JSON}
000013f0	Concentrator			
00001134	SiPM			

```
{
  "rov": {
    "description":"rec. oper. V",
    "value":"38"
  },
  "ov": {
    "description":"oper. V",
    "type":"dec_1",
    "lo":"36",
    "hi":"40"
  }
  ...
}
```

Reading and writing of data

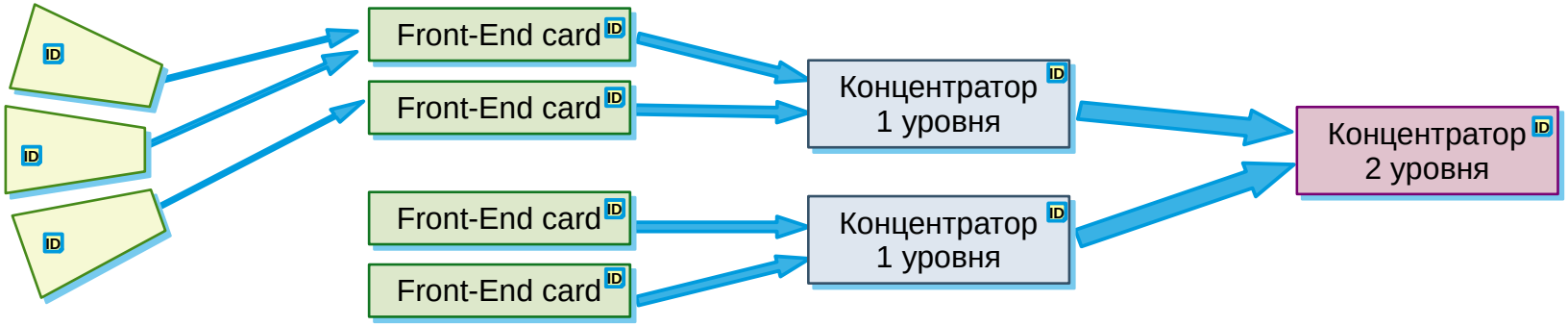
- The hardware database must support the following data loading methods:
  - via the Web interface,
  - from a JSON file
- The information should be available through
  - various APIs (REST, Python, C++) for DAQ systems and
  - Web interface programs that allow you to get information about a specific device or search by parameters
- It is necessary to organize access rights in such a way that only those responsible for individual subsystems can enter and edit data related to them

Прототип ИС БД оборудования

- A prototype of the equipment information system is being developed
- PostgreSQL DBMS is used for data storage
  - JSON support, which allows you to use schema-less data. Built-in specialized JSON operators and functions
  - You can set up synchronous data duplication, and use separate copies for DAQ, OnlineFilter, offline processing
- For the data exchange and processing the framework using RESTful API has been created, using solutions similar to the ones used for the EventIndex
- It should be deployed in a VM and presented for testing

DAQ Mapping

- The number of data collection channels of the SPD installation will be several hundred thousand
- The signals from the detector will pass through several communication devices



- It is necessary to have a mapping of the data collection system that establishes the correspondence of the channel addresses at the DAQ outputs with the devices from which this signal came

Building of mapping

- Due to the large number of elements in the system, it is almost impossible to build mapping manually
- For the elements involved in the transmission of digital signals, an automatic mapping procedure should be implemented
  - The element must issue a HW ID over the data channel in response to a special signal
- For parts of the system that are not equipped with automatic source ID recognition, an interface must be provided that allows data entry by groups.

Disabling problematic channels

- During the data taking, "dead" or noisy channels may be detected.
- Immediate replacement of a faulty module may not be possible or difficult
- In such cases, problematic channels should be excluded from the dataset, and this should be taken into account during processing.
- Disabling channels in the middle of a run can create problems with data processing
- To disable the channels, the data set is suspended, appropriate changes are made to the installation configuration and the mapping database, and a new RUN is started

Organizing of mapping DB

- The table that holds information on different device types should also contain
  - data on upstream and downstream ports,
  - type of the devices that may connect to them
  - intended “level” of the device (level 1 or 2 concentrator, front-end card)
- The table or tables can be created, using data above, containing
  - Device ID
  - It’s type
  - ID’s of the devices connected to the upstream and downstream ports
- Using this information the mapping of the device ID to the specific channel can be defined

Organizing of mapping DB

- This database architecture allows you to easy replace specific device, change connection between devices, and even add intermediate layers between devices, which makes it highlyly configurable.
- Thanks to the step hierarchy, it is possible to trace the chain from the last device to the first and determine at what stage the channel creates noise or does not work.
- The faulty channel can be masked off either by setting status “block” in the corresponding field or by placing in a separate table
- The faulty device should be marked bad both in the mapping DB and in the hardware database

Performance requirements

- It is expected that the filling of the hardware database will take place gradually, and updates will be rare
- The construction of the connection diagram and its changes will also be performed rarely (no more than once a week)
- The requirements for the speed of recording information in the database are low
- Mapping information may be required when processing each file. It is possible to simultaneously access the system of tens of thousands of processes.
- It is necessary to ensure their processing, avoiding database overload due to too high frequency of requests

База данных персонала и документов

Database of personnel and documents

- About 400 people are currently participating in the SPD project, and the number of participants is expected to grow when approaching to the experiment commissioning and operation
- In order to organize effective cooperation with the shared use of computing and other resources, it is necessary to organize
  - handling of a personnel data
  - working groups organizing
  - accounting of the contribution to their implementation.
- During the experiment, a large number of different documents will appear, both internal and public.
  - It is necessary to implement procedures for its tracking and processing

Personnel information

- ID and status in collaboration (member, student, associated member, user)
- Personal and contact information
- Link to the organization(s), position in it
- Participation in working groups, roles and positions (with history)
  - It is necessary to create an access control system with the implementation of individual and group privileges
- Accounting for the contribution to the experiment
- Information about authorship (with a history)
  - Data on the passage of qualifications, if there will be such feature
- Links to publications, presentations and other documents

Information about organizations

- ID and status of the organization:
  - It is necessary to provide for the possibility of group and delegated membership/participation in the collaboration
- Information about the organization
- Contact information,
  - including links to representatives and participants
- Participating in projects, commitments and their fulfillment
- Accounting for the contribution of employees of the organization
- Participation in organizational events
- Links to organizational document, reports and publications

Authorization, roles and privileges

- To identify employees, JINR authentication services should be used
  - providing possibility of access by external employees who do not have an JINR account
  - introducing group and robot accounts for use in automated tasks
- A role and privilege management system should be implemented as well as group access policies
- Procedures for including users in groups and revoking membership
- The resources of the working groups should use a single authentication and authorization system

Document and publication management

- Implements procedures for creating, editing, and approving documents related to the personnel database
  - Registration and changes of membership in the collaboration
  - Creating and editing lists of groups and privileges
  - Inclusion in the author's lists
- Tracking the progress of publications, organizing the exchange of messages between authors, reviewers and curators, as well as searching through documents
  - It is also desirable to organize tracking of SPD publications in external information systems.
- Obtaining statistical information on the number of publications, broken down by authors, topics...

Organization of reports and presentations

- Organization of presentations and reports at conferences and meetings:
  - Compiling a list of conferences and available reports
  - Organization of call for speakers and selection
  - Acceptance, review and approval of titles, abstracts and slides
  - Tracking the publication of proceedings

BACKUP

Информационные системы SPD

- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nisl tincidunt eget nullam non.
- Quis hendrerit dolor magna eget est lorem ipsum dolor sit:
  - Volutpat odio facilisis mauris sit amet massa.
  - Commodo odio aenean sed adipiscing diam donec adipiscing tristique
  - Mi eget mauris pharetra et. Non tellus orci ac auctor augue. Elit at imperdiet dui accumsan sit
- Ornare arcu dui vivamus arcu felis. Egestas integer eget aliquet nibh praesent. In hac habitasse platea dictumst quisque sagittis purus. Pulvinar elementum integer enim neque volutpat a