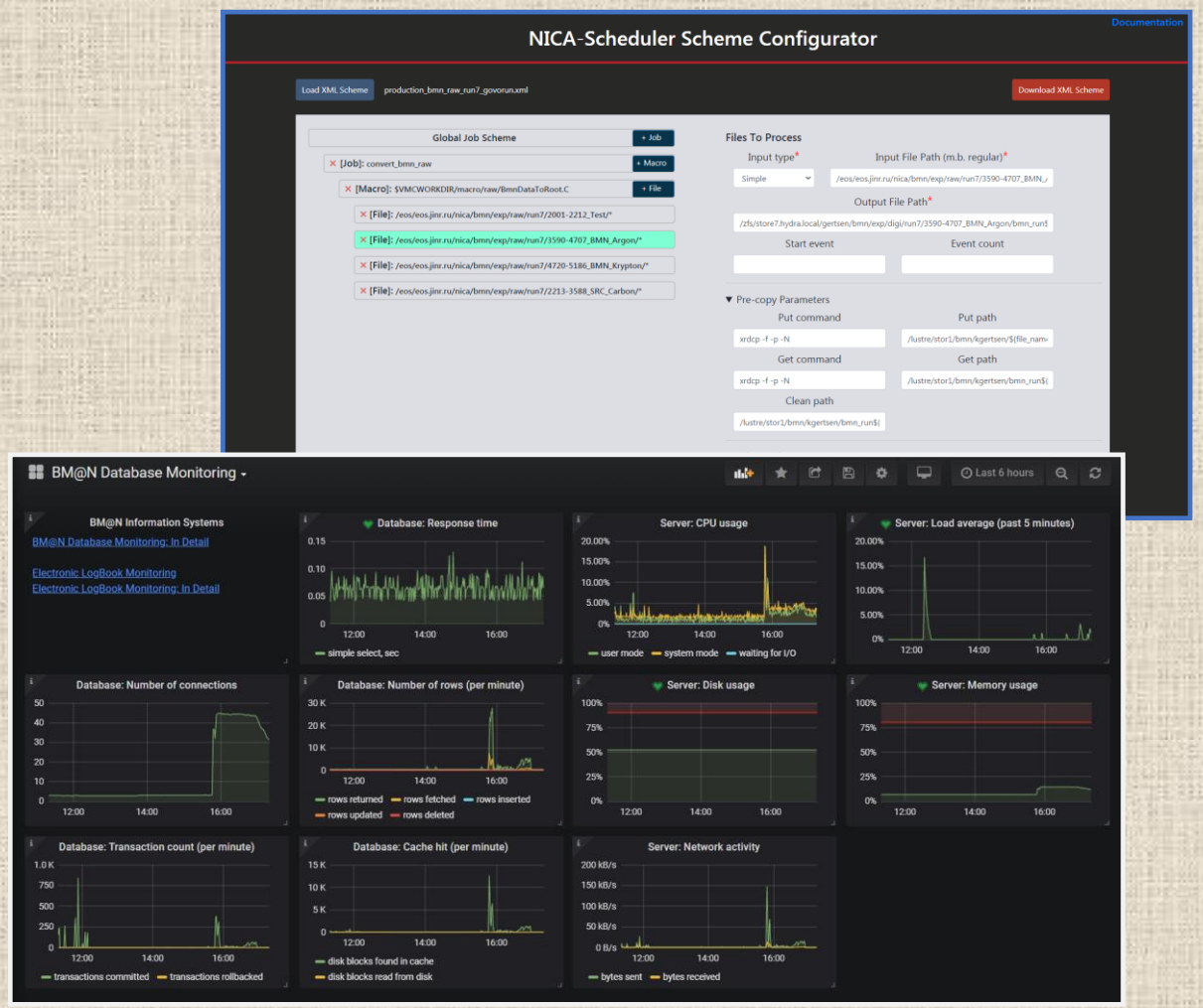
Software contribution from MIPT: implementation of systems and services for BM@N

Peter Klimai

MIPT Projects for BM@N

MIPT projects for BM@N include services for:

- Visualization,
- Monitoring,
- Statistics collection, and more



About MIPT-NPM:

- http://npm.mipt.ru/ru/
- https://research.jetbrains.org/groups/npm

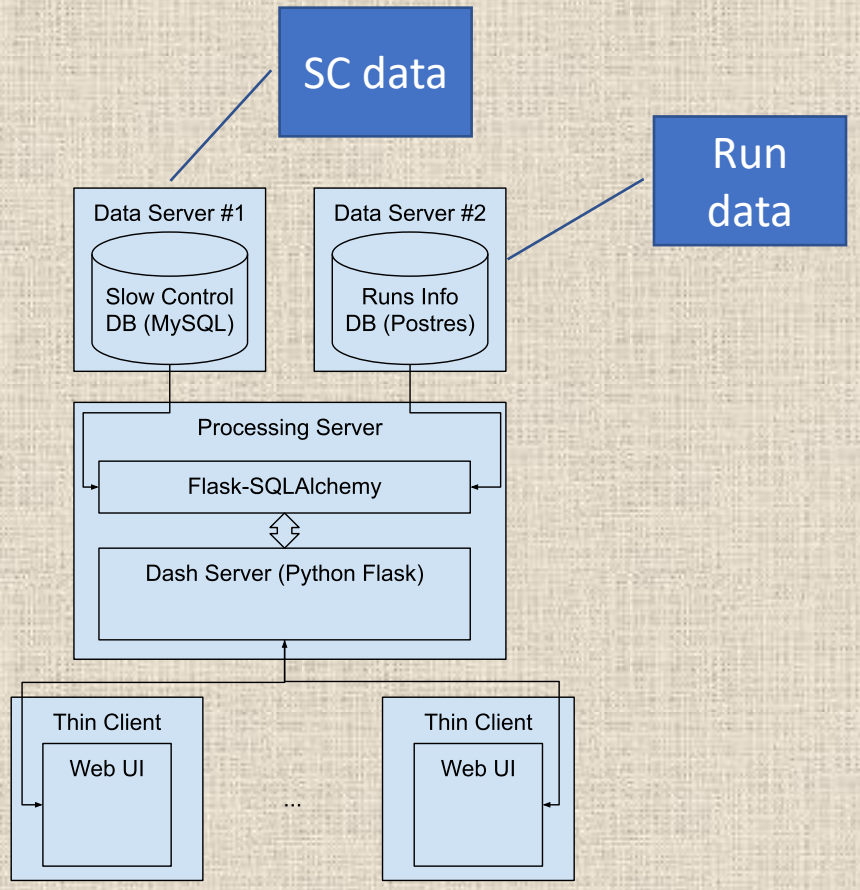
Project Summary

Project	Status
NICA Scheduler Configurator GUI https://bmn-scheduler.jinr.ru	In production
Monitoring service https://mon-service.jinr.ru	In production TODO: Add HTTP(S) monitoring
Next-gen event display https://github.com/mipt-npm/visionforge	Working on ROOT integration
Slow control system viewer https://bmn-tango.jinr.ru	Bugs fixed, deployed to production
Statistics collection script	Available as part of BmnRoot
Event indexing service	Pgsql and Cassandra db tests, system design
Experimental data migration tool (text to database)	New

Slow Control System Viewer

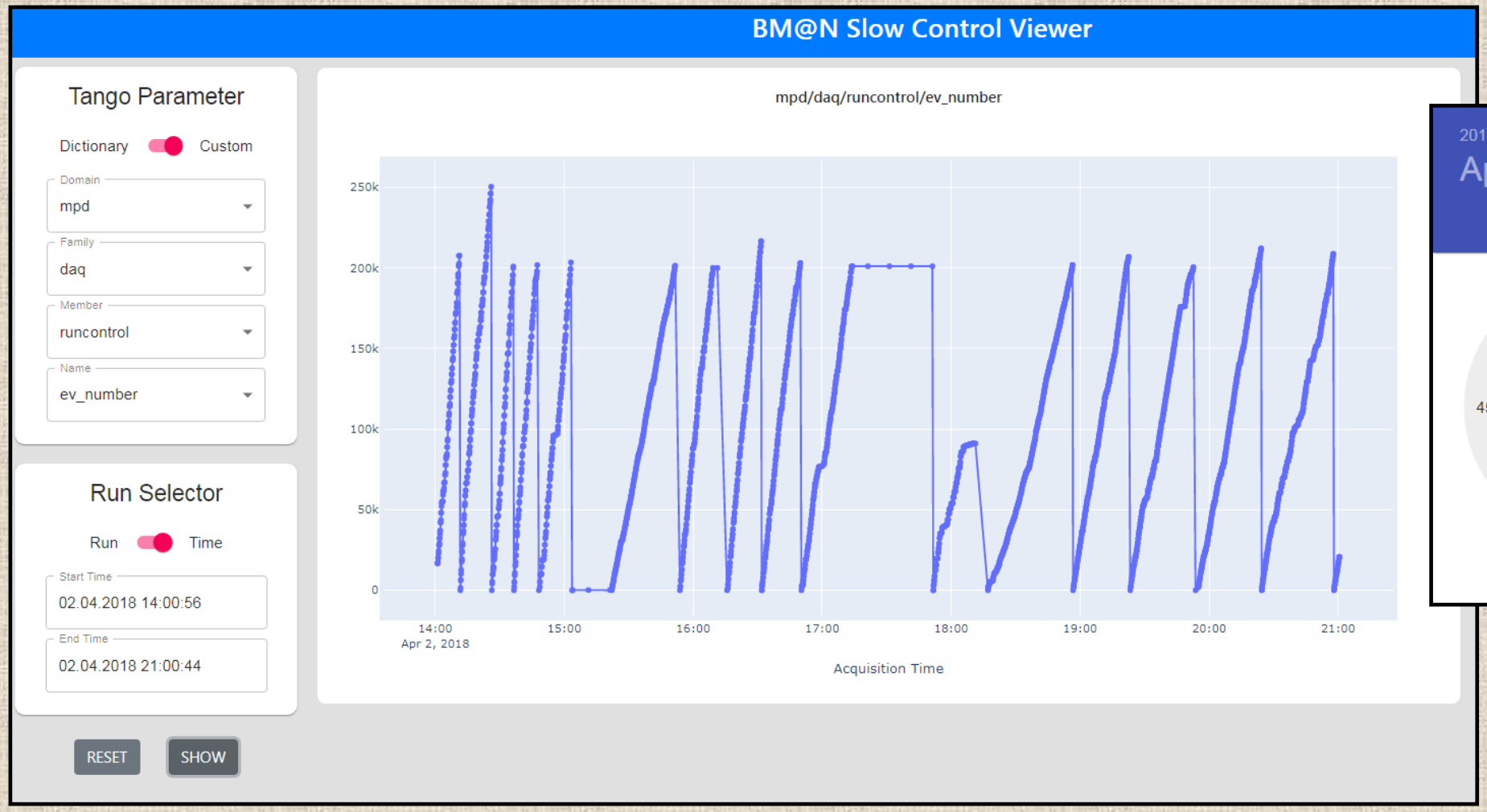
Slow Control Viewer – Task

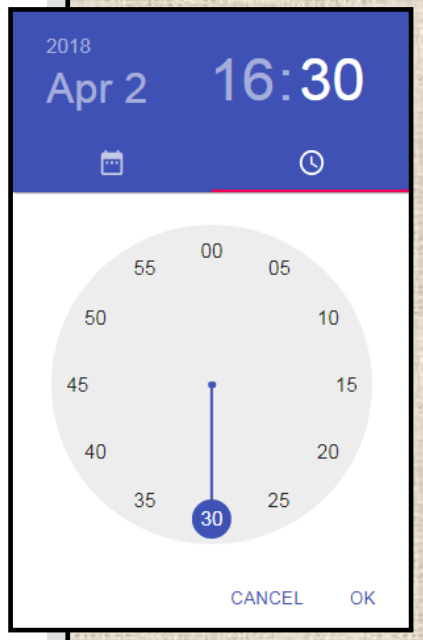
- Task
  - Web interface for slow control system of BM@N
  - Existing SCS Tango has MySQL DB with all sensor data obtained during runs
  - Show sensor data graph based on:
    - User-friendly alias for parameter name OR domain/family/member/name as stored in DB
    - Run no. OR time interval
  - Parameter can be 1d array – in this case a multigraph is displayed
- Implementation
  - Uses Python-based Dash framework, all packed in Docker container
  - Sources: https://git.jinr.ru/nica_db/tango_web



Example 1 – Parameter and Time Selection

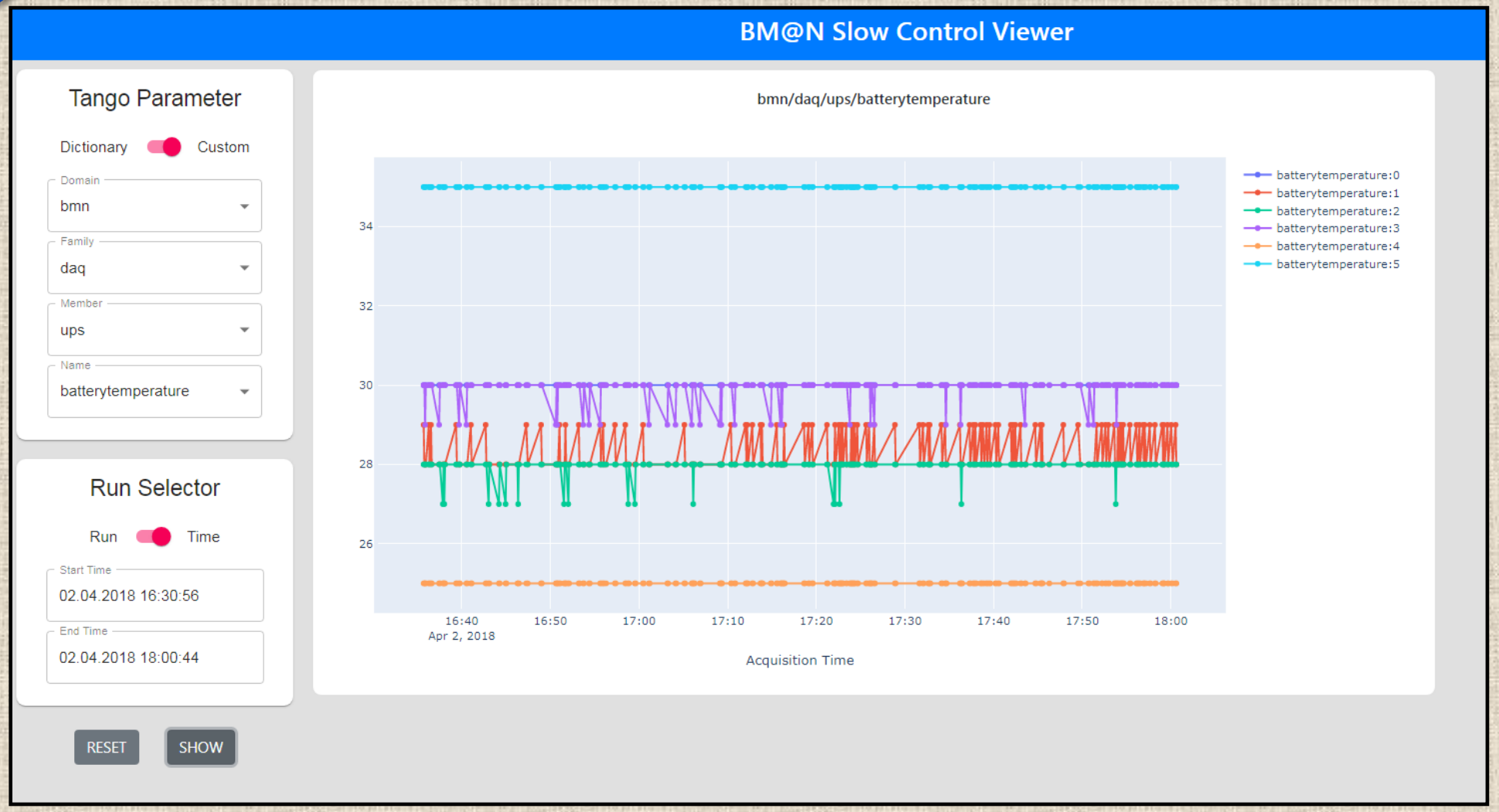
https://bmn-tango.jinr.ru





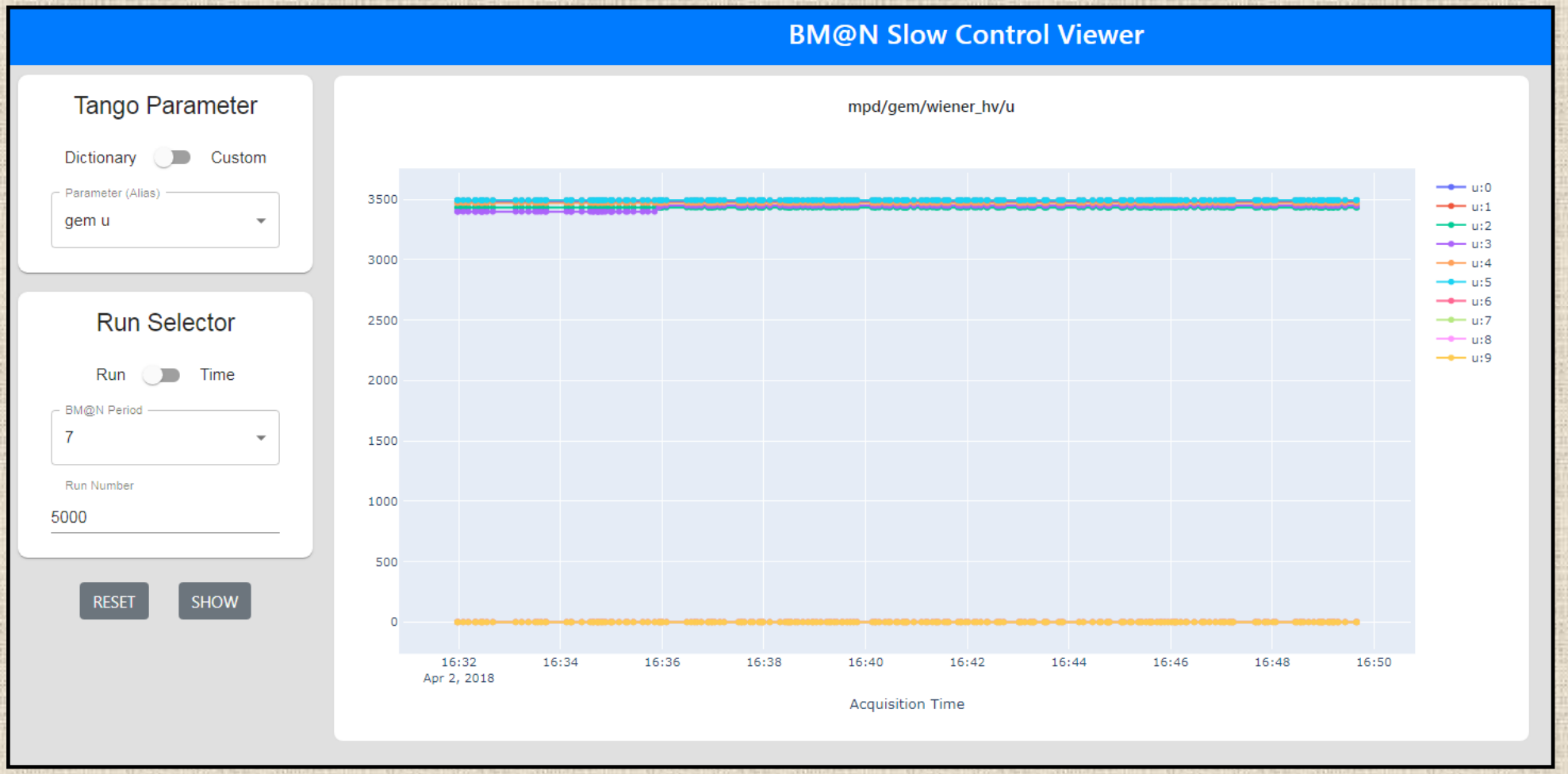
Software contribution from MIPT: implementation of systems and services for BM@N

Example 2 – Multigraph



Software contribution from MIPT: implementation of systems and services for BM@N

Example 3 – Parameter Dictionary



Software contribution from MIPT: implementation of systems and services for BM@N

Statistics Collection

Statistics Collection Script

- Task
  - Show histograms, and summary data, for:
    - File size and size per event in a given directory (.data, .root files – configurable)
    - Parse logs to get time of processing for macros (including time per event)
  - Implemented as Python script **stats.py**
  - https://git.jinr.ru/nica/bmnroot/-/tree/dev/uni_db/services/statistics

```
python3 stats.py [-h] [--dir [dir]] [--size] [--time]
                 [--config [CONFIG]] [--output [OUTPUT]]


optional arguments:
  -h, --help            show this help message and exit
  --dir [dir], -d [dir]
                        Name of directory to explore
  --size, -s            Compute size statistics
  --time, -t            Compute time statistics
  --config [CONFIG], -c [CONFIG]
                        Path to config file, default is ./config.txt
  --output [OUTPUT], -o [OUTPUT]
                        Path to output file, default is ./output.png
```

Other parmeters are specified in JSON config file.

required parameters:

- `extensions` - data or log file extensions (use ["*"] to include all files)
- `db_user`, `db_pass`, `db_name`, `db_host` - database credentials

optional parameters:

- `dpi` - dpi for the generated diagram
- `folders_ignore` - directories that must be ignored while processing
- `file_size_limit` - constrain min/max size of files to be processed (for example, `50kb:500gb`)
- `event_size_limit` - constrain min/max file size per event (for example, `1kb:1mb`)

JSON Config Examples

- Example JSON configurations
  - For `--size`:

```
{
  "extensions": [".data", ".root"],
  "db_user": "db_reader",
  "db_pass": "********",
  "db_name": "bmn_db",
  "db_host": "vm221-53.jinr.ru",
  "file_size_limit": "50kb:500gb",
  "event_size_limit": ""
}
```

  - For `--time`:

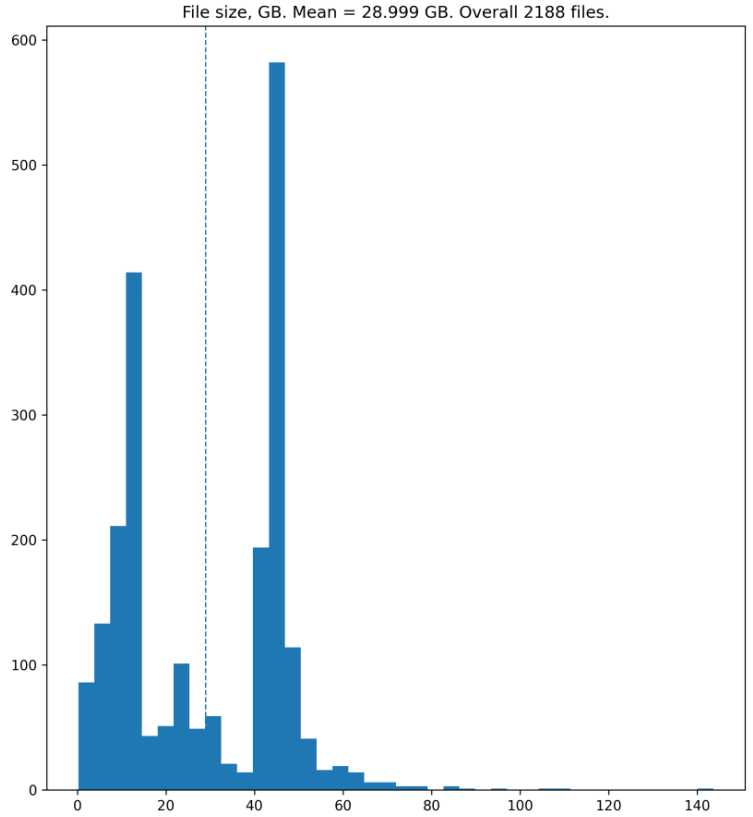
```
{
  "extensions": ["*"],
  "db_user": "db_reader",
  "db_pass": "********",
  "db_name": "bmn_db",
  "db_host": "vm221-53.jinr.ru"
}
```

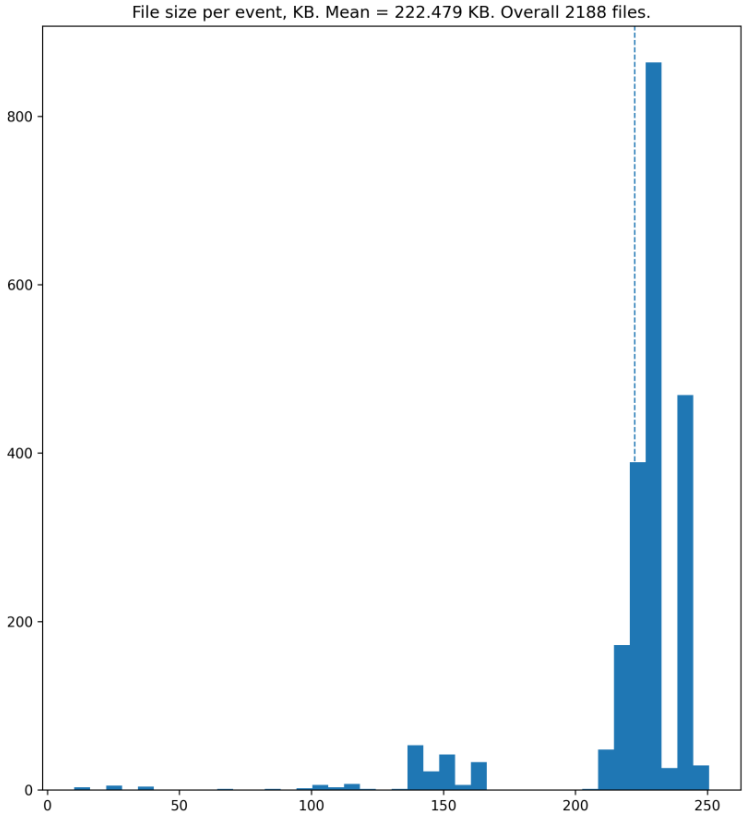
Script Execution Example for File Size (raw)

```
[pklimai@ncx105 statistics]$ python3 stats.py --size --dir /eos/nica/bmn/exp/raw/run7/ --config config-size.json --recursive
Statistics calculation script started.
Calculating file size statistics...
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…
++++++++++++++++++++++++++++++++++++++++++++
Total files parsed: 2188

All files processed successfully.


Obtained characteristics:
  File statistics: min = 0.261 GB, avg = 28.999 GB, max=143.620 GB, summary=63450.440 GB
  File statistics per event: min = 10.221 KB, avg = 222.479 KB, max=250.615 KB

Writing output to ./output.png
Trying to open graphics...   ...ok
Script execution finished.
```

Note: for experimental files, run number is extracted from the file name, e.g. mpd_run_trigCode_**4986**.data

File Size Histogram Example





Feature: Event number check

- For ROOT DST files, event number check is implemented
  - Event count from database is compared against actual number of events in a file
  - Event count is extracted using uproot 4 Python package
  - If numbers don’t match, warning is displayed and file name is added to unsuccessful_list.txt

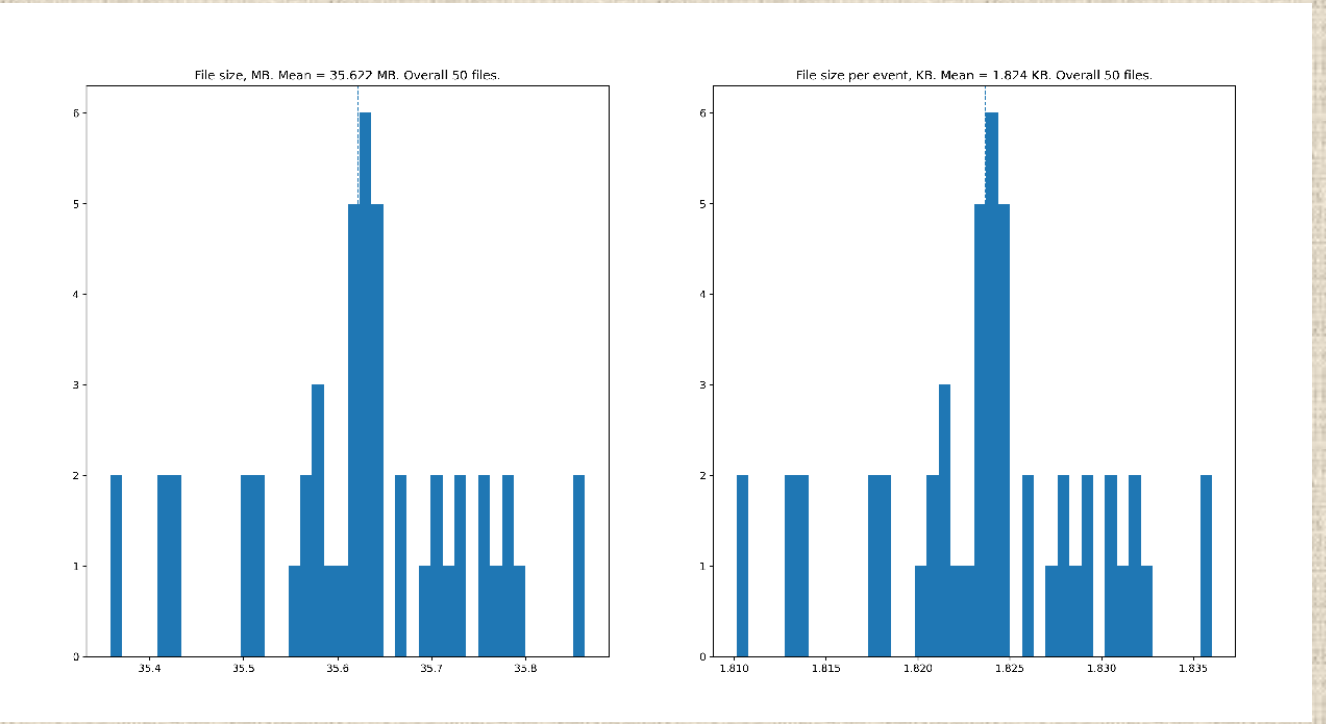
Simulated Files

- For simulated files, similar processing can be performed
  - Specify `"source": "sim"` in JSON config
  - Event count is taken from `simulation_file` table of Unified database

```
$ cat config-size-sim.json
{
        "extensions": [".r12"],
        "db_user": "db_reader",
        "db_pass": "********",
        "db_name": "bmn_db",
        "db_host": "vm221-53.jinr.ru",
        "file_size_limit": "",
        "event_size_limit": "",
        "source": "sim"
}
```

Simulated Files Example

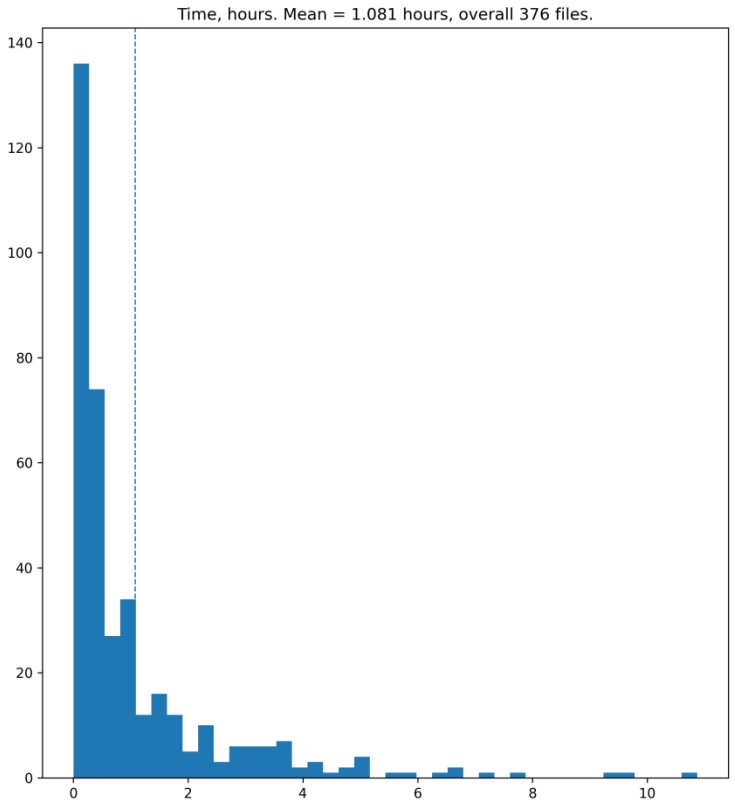
```
[pklimai@ncx105 statistics]$ python3 stats.py --size --dir /eos/nica/bmn/sim/gen/DCMQGSM/CC_3.5_mb_20k/ --config config-size-sim.json --recursive
Statistics calculation script started.
Calculating file size statistics...
++++++++++++++++++++++++++++++++++++++++++++++++++
Total files parsed: 50

All files processed successfully.


Obtained characteristics:
  File statistics: min = 35.358 MB,
    avg = 35.622 MB, max=35.863 MB,
    summary=1781.093 MB
  File statistics per event: min = 1.810 KB,
    avg = 1.824 KB, max=1.836 KB

Writing output to ./output.png
Trying to open graphics...
```

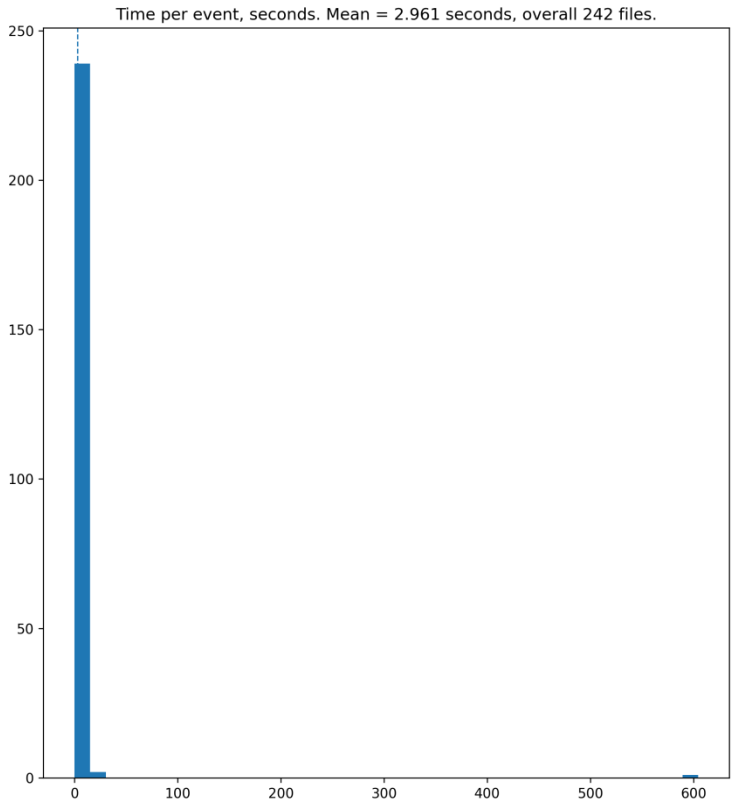


Script Execution Example for Time Stats

```
[pklimai@ncx105 statistics]$ python3 stats.py --time --dir /eos/nica/bmn/users/gertsen/logs/batch_raw_run5/
--config config-time.json
Statistics calculation script started.
Calculating time statistics...
+++  File has time and run number, but not ended successfully - skipping
/eos/nica/bmn/users/gertsen/logs/batch_raw_run5/convert_bmn_raw.o2017539.100
+++  Warning: more than one run records: the latest period number is selected for
/eos/nica/bmn/users/gertsen/logs/batch_raw_run5/convert_bmn_raw.o2017539.104
+++ Can't get events count from the database - skipping file
/eos/nica/bmn/users/gertsen/logs/batch_raw_run5/convert_bmn_raw.o2017539.110   …

Total files parsed: 398

Unsuccessfully ended runs:
/eos/nica/bmn/users/gertsen/logs/batch_raw_run5/convert_bmn_raw.o2017539.100    …

Unsuccessfully processed files list (22/398, 5.5%) was saved to unsuccessful_list.txt

Obtained characteristics:
  Mean time = 1.081 hours.
  Summary time = 406.323 hours.
  Mean time per event = 2.961 seconds.

Writing output to ./output.png
Trying to open graphics...              ...ok
Script execution finished.
```

Time Histogram Example

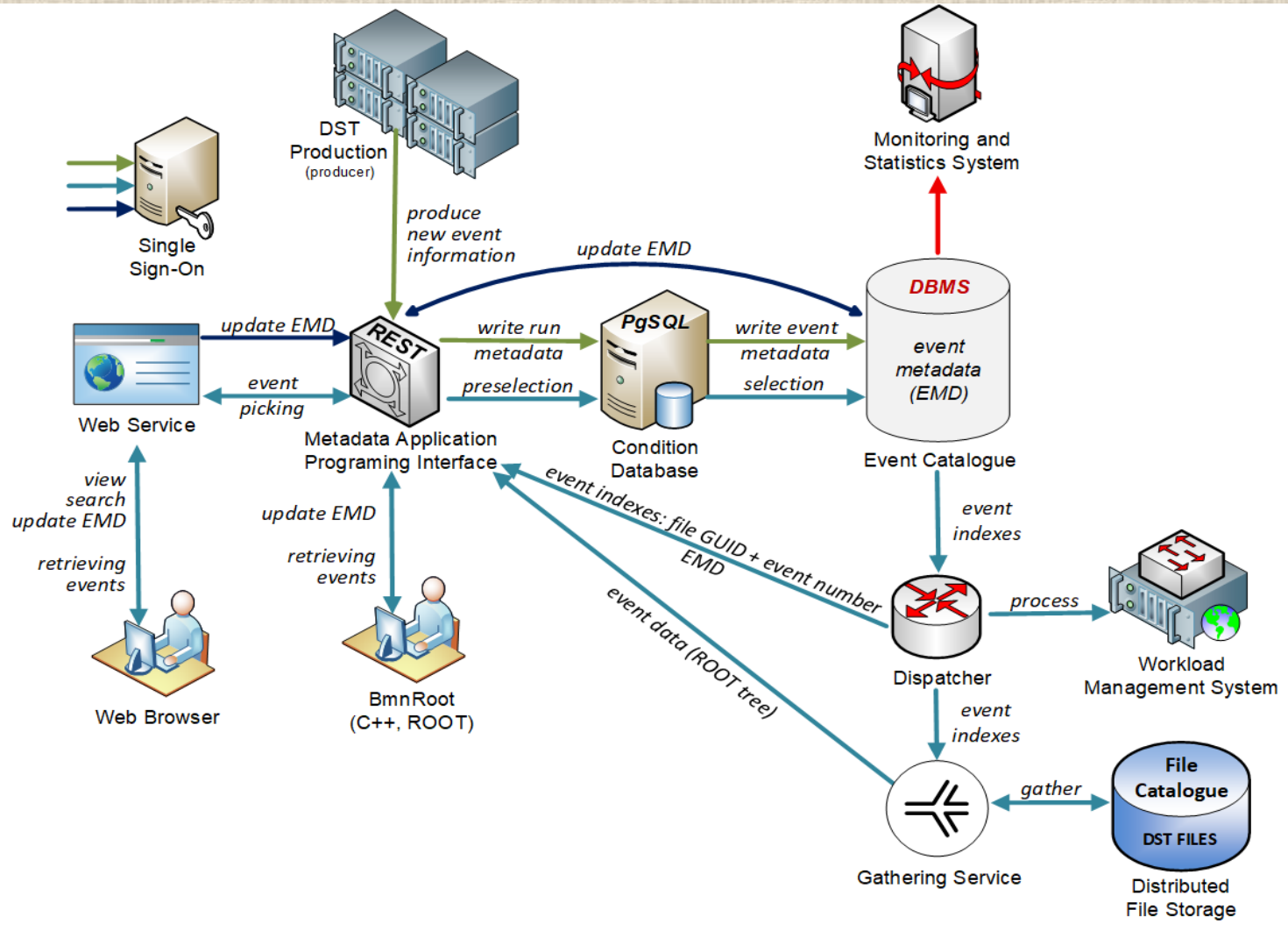




Software contribution from MIPT: implementation of systems and services for BM@N

Event Indexing

Event Metadata Architecture



Software contribution from MIPT: implementation of systems and services for BM@N

EMD DBMS Choice – Task

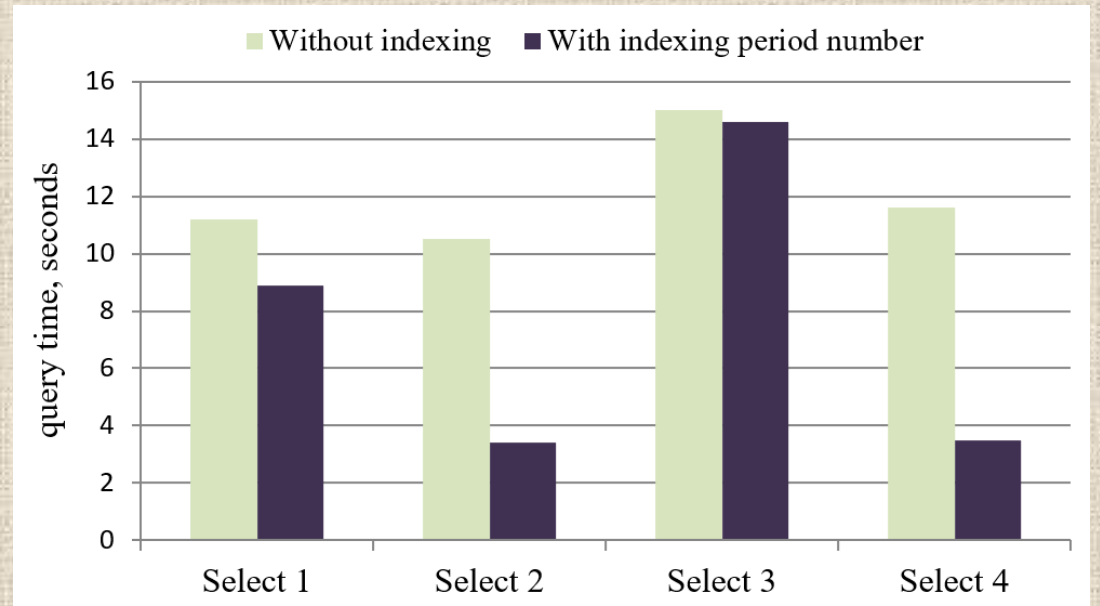
- Proposed DB schema (https://git.jinr.ru/nica/bmnroot/-/issues/58):
  - period_number (int, 4 byte)
  - run_number (int, 4 byte)
  - software_id (int, 2 byte)
  - file_id (int, 8 byte)
  - event_number (int, 4 byte)
  - primary_vertex (boolean)
  - primary_tracks (int, 4 byte)
  - all_tracks (int, 4 byte)
  - positive_tracks (int, 4 byte)
  - detector_hit (int, 4 byte)
  - particles (int, 4 byte)

Example Requests to DB

- It is desirable to have queries similar to (SELECT … FROM … WHERE …):
  - `period = 7 AND software_version = 20.02.0 AND primary_tracks > 5`
  - `period = 6 AND software_version = 19.05.0 AND primary_vertex = false AND detector_hit[bits:0-5] > 20`
  - `period = 5 AND run_number > 100 AND primary_vertex = true AND particles[bits:5-9] > 4`
  - `period = 4 AND software_version = 19.10.0 AND primary_tracks > 6 AND all_tracks > 40 AND particles[bits:10-14] > 6`

DBMS Tests – PostgreSQL

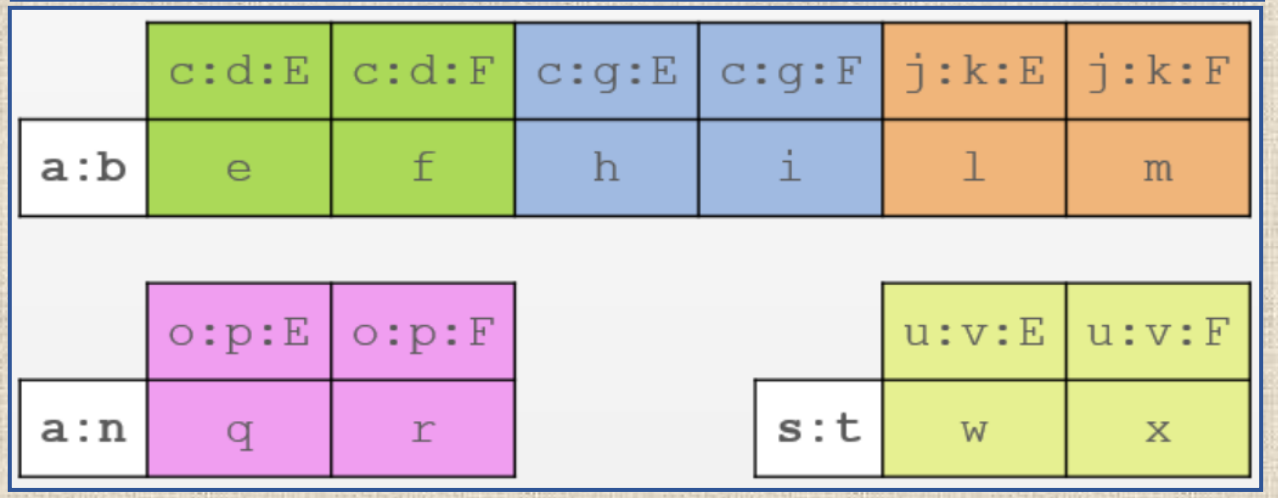
- One of PostgreSQL tests (500M events)
  - Intel Core i9-10900F 2.80GHz (20 CPU cores), 64 GB RAM, 1TB NVMe SSD disk, CentOS 8.2, PostgreSQL 12.5
  - Time typically grows linearly with db size



DBMS Tests – Cassandra

- Cassandra
  - Overall, Cassandra / NoSQL requires us to think differently
  - First requests, then data
  - Both key-value and column-based database
  - Need to plan requests very carefully. Normally, new request = new table and data duplication
  - ALLOW FILTERING exists, but is killing performance
  - CQL has many limitations compared to SQL
    - E.g. comparison can only be applied to last element of partition or clustering key
    - Comparison is only <, > (no bitwise operators etc.)

```
CREATE TABLE example (
  A text,
  B text,
  C text,
  D text,
  E text,
  F text,
  PRIMARY KEY ((A,B), C, D)); -- составной распределительный ключ (A,B) и кластерные ключи (C,
D)
INSERT INTO example (A, B, C, D, E, F) VALUES ('a', 'b', 'c', 'd', 'e', 'f');
INSERT INTO example (A, B, C, D, E, F) VALUES ('a', 'b', 'c', 'g', 'h', 'i');
INSERT INTO example (A, B, C, D, E, F) VALUES ('a', 'b', 'j', 'k', 'l', 'm');
INSERT INTO example (A, B, C, D, E, F) VALUES ('a', 'n', 'o', 'p', 'q', 'r');
INSERT INTO example (A, B, C, D, E, F) VALUES ('s', 't', 'u', 'v', 'w', 'x');
```

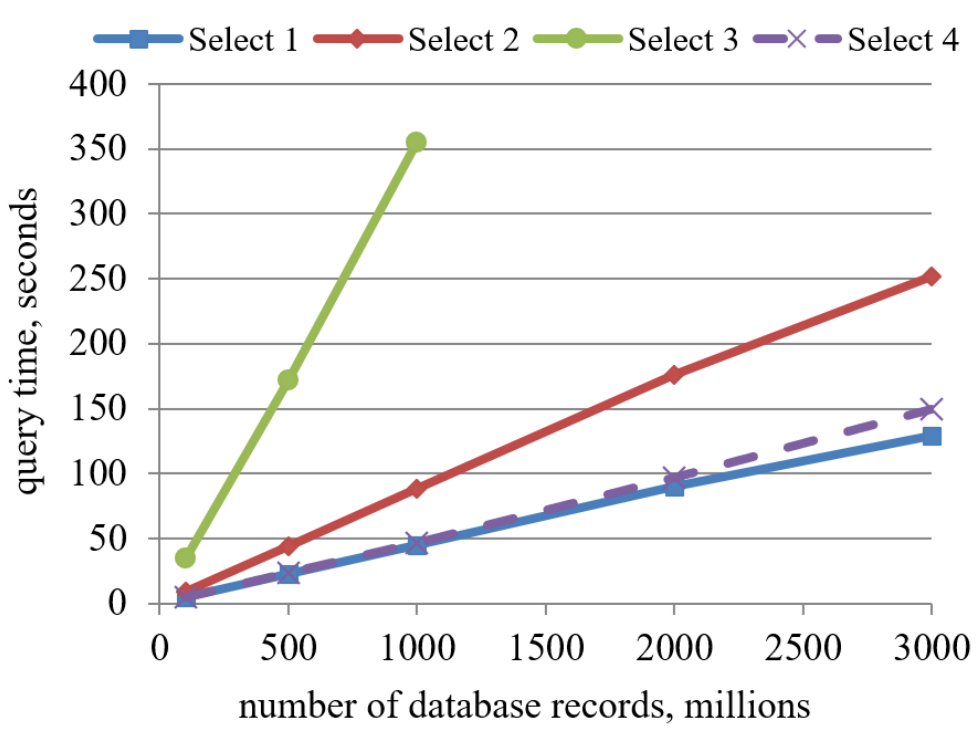


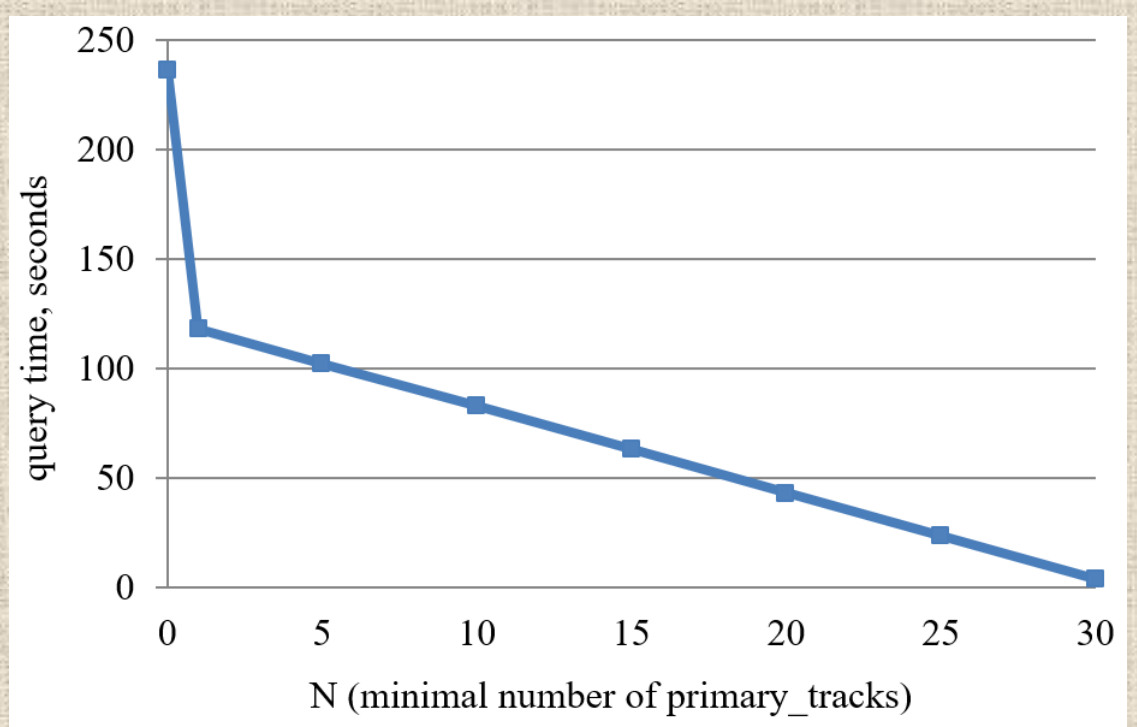
https://habr.com/ru/post/203200/

Cassandra Setup

- Cassandra test
  - Single-node Cassandra 3.11.8
  - Intel Core i9-10900F 2.80GHz (20 CPU cores), 64 GB RAM, 1TB NVMe SSD disk, CentOS 8.2
  - Table with
    - PRIMARY KEY = ((period_number, software_id), primary_tracks, event_number)
      ((partition key), clustering columns)
  - It is projected for requests of the form
    - SELECT... WHERE period_number=... AND software_id=... AND primary_tracks=…
    - Can also use IN instead of =, but time increases linearly
    - Comparison (only <, >) is allowed only for the last part of clustering key

Cassandra Performance Estimates






```
period = 7 AND software_version =
20.02.0 AND primary_tracks > N
```

Improving Performance

- Performance can be improved by
  - Using several subqueries instead of one which returns a big data volume
  - Choosing optimal primary keys

OPTIMAL PRIMARY KEYS

SELECT [...] FROM q.events WHERE	PRIMARY KEY
period_number = 7 AND software_id = 3 AND primary_tracks > 5	((period_number, software_id, primary_tracks), file_id, event_number)
period_number = 6 AND software_id = 2 AND primary_vertex = false	((period_number, software_id, primary_vertex), file_id, event_number)
period_number = 5 AND run_number>100 AND primary_vertex = true	((period_number, primary_vertex), run_number, file_id, event_number)
period_number = 4 AND software_id = 1 AND primary_tracks>6 AND all_tracks>40	((period_number, software_id, primary_tracks), all_tracks, file_id, event_number)

QUERY EXECUTION TIMES

Data volume	SELECT event_number	SELECT COUNT(*)
1st query		
0.5B	5,5	1,7
1B	10,8	3,3
1.5B	17,4	4,9
2B	24,7	6,6
2nd query		
0.5B	6,5	1,8
1B	16,6	3,8
1.5B	21,1	5,6
2B	33,1	7,3
3rd query		
0.5B	77,1	10,5
1B	170,0	10,8
1.5B	242,0	12,2
2B	741,2	13,8
4th query		
0.5B	5,1	1,7
1B	10,6	3,3
1.5B	16,0	4,9
2B	21,4	7,6

Thank You!