ITk DAQ Workshop (28/05 - 31/05)

Local Database in YARR SW

Arisa Kubota (Tokyo Tech)
Tokyo Tech: Osamu Jinnouchi, Hideyuki Oide, Eunchong Kim, Hiroki Okuyama
Osaka Univ.: Minoru Hirose, Shohei Yamagaya
Bologna: Giuseppe Carratta, Antonio Sidoti

YARR: https://gitlab.cern.ch/YARR/YARR
(currently LocalDB developed in devel-localdb branch)
LocalDB-tools (Viewer Application): https://github.com/jlab-hep/localDB-tools
Wiki: https://github.com/jlab-hep/Yarr/wiki

Overview of Local Database

Referential talks about Local DB

ITk SW meeting on 16 Nov 2018 talked by Eunchong
https://indico.cern.ch/event/766838/contributions/3219312/attachments/1754554/2844168/181116_software_meeting_kim.pdf

ITk SW meeting on 14 Dec 2018 talked by Minoru
https://indico.cern.ch/event/766840/contributions/3252338/attachments/1771915/2879675/181214_00_hirose.pdf

QA/QC workshop on 23 Jan 2019 talked by Jannicke
https://indico.cern.ch/event/779148/contributions/3291012/attachments/1784022/2904428/JP_24_01_19.pdf

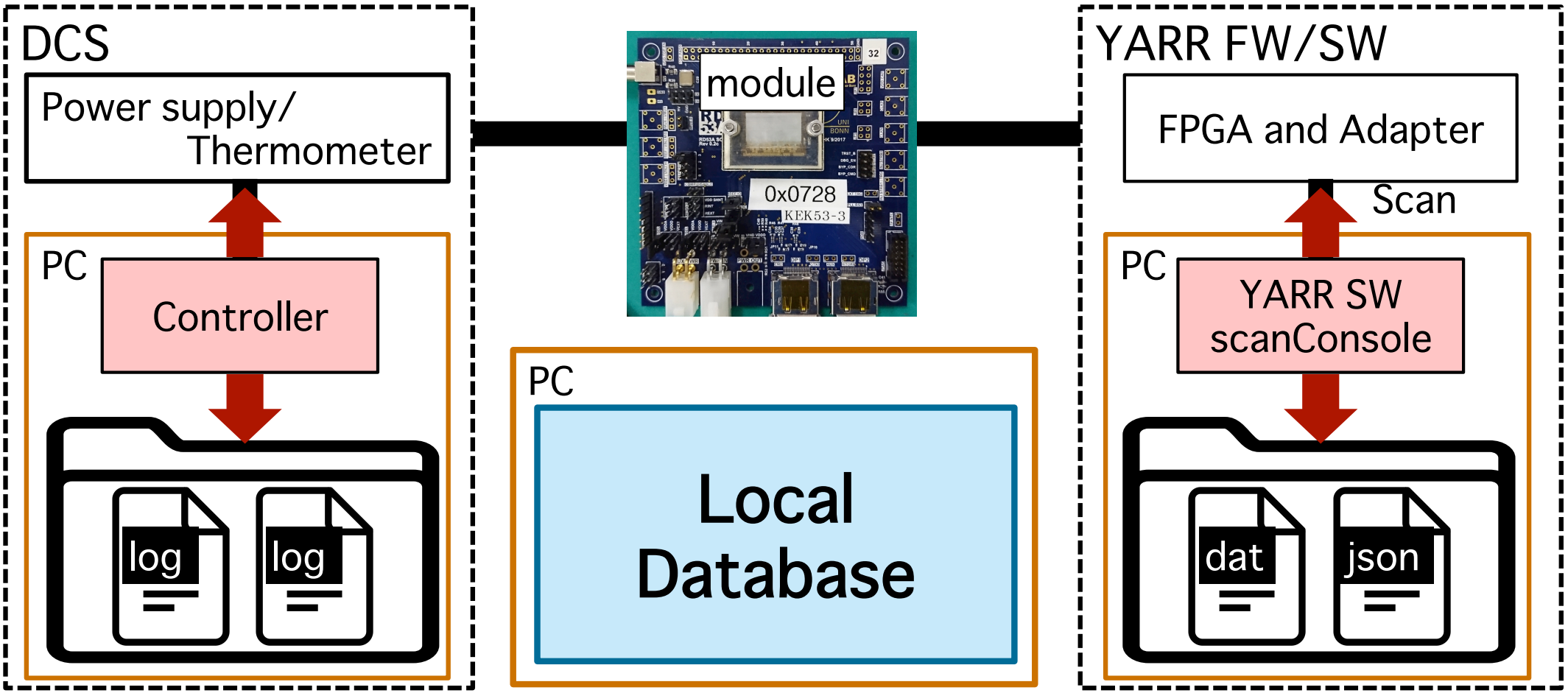
ITk SW meeting on 9 Feb 2019 talked by Arisa
https://indico.cern.ch/event/790727/contributions/3314465/attachments/1793277/2922200/ITkSW_0209_kubota.pdf

ITk SW meeting on 22 Mar 2019 talked by Arisa
https://indico.cern.ch/event/790730/contributions/3366823/attachments/1816927/2969907/ITk_SW_meeting_0322.pdf

Overview of Local DB (1/3)

Test data and results are uploaded into LocalDB automatically

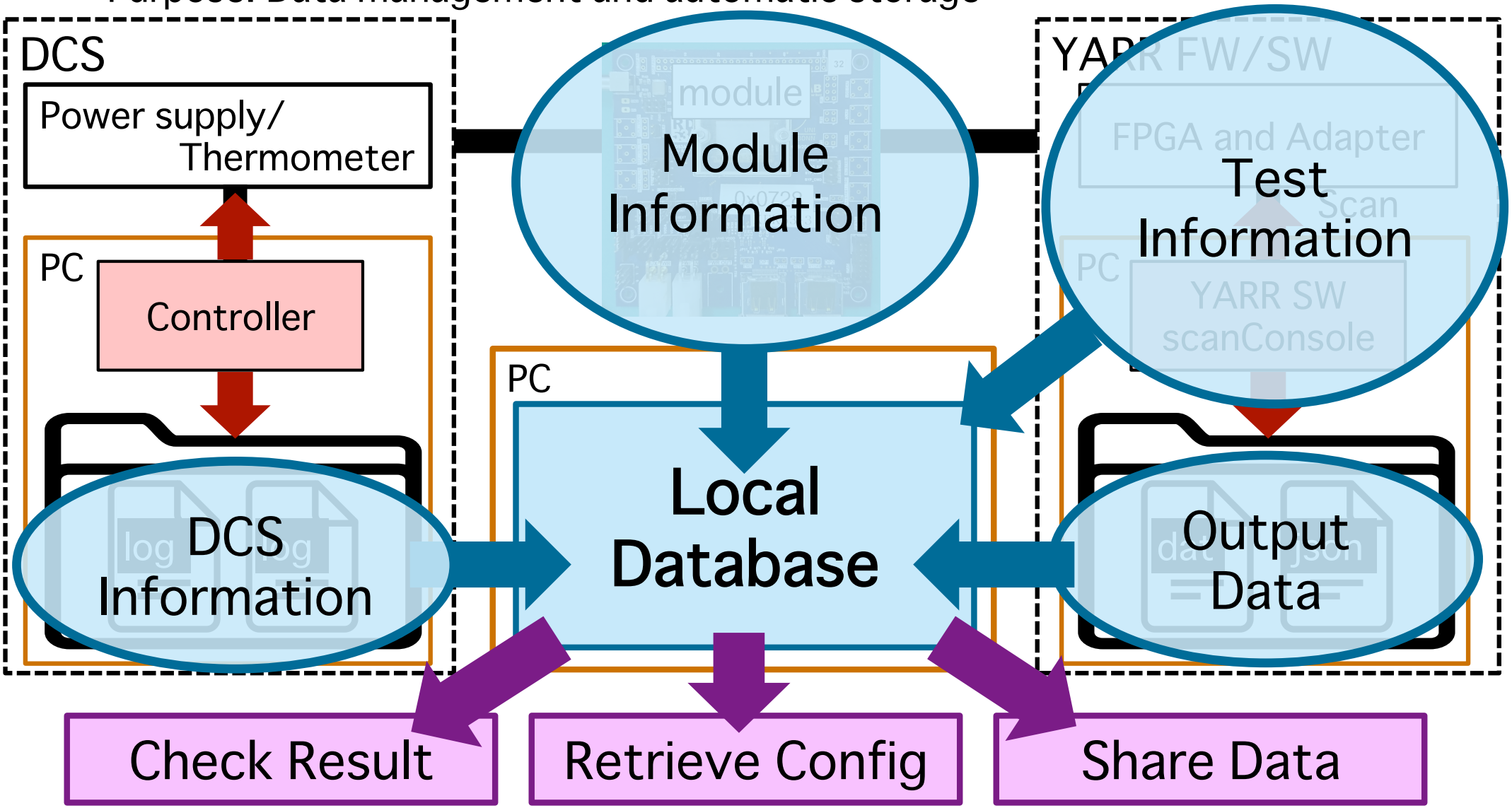
- Purpose: Data management and automatic storage



Overview of Local DB (2/

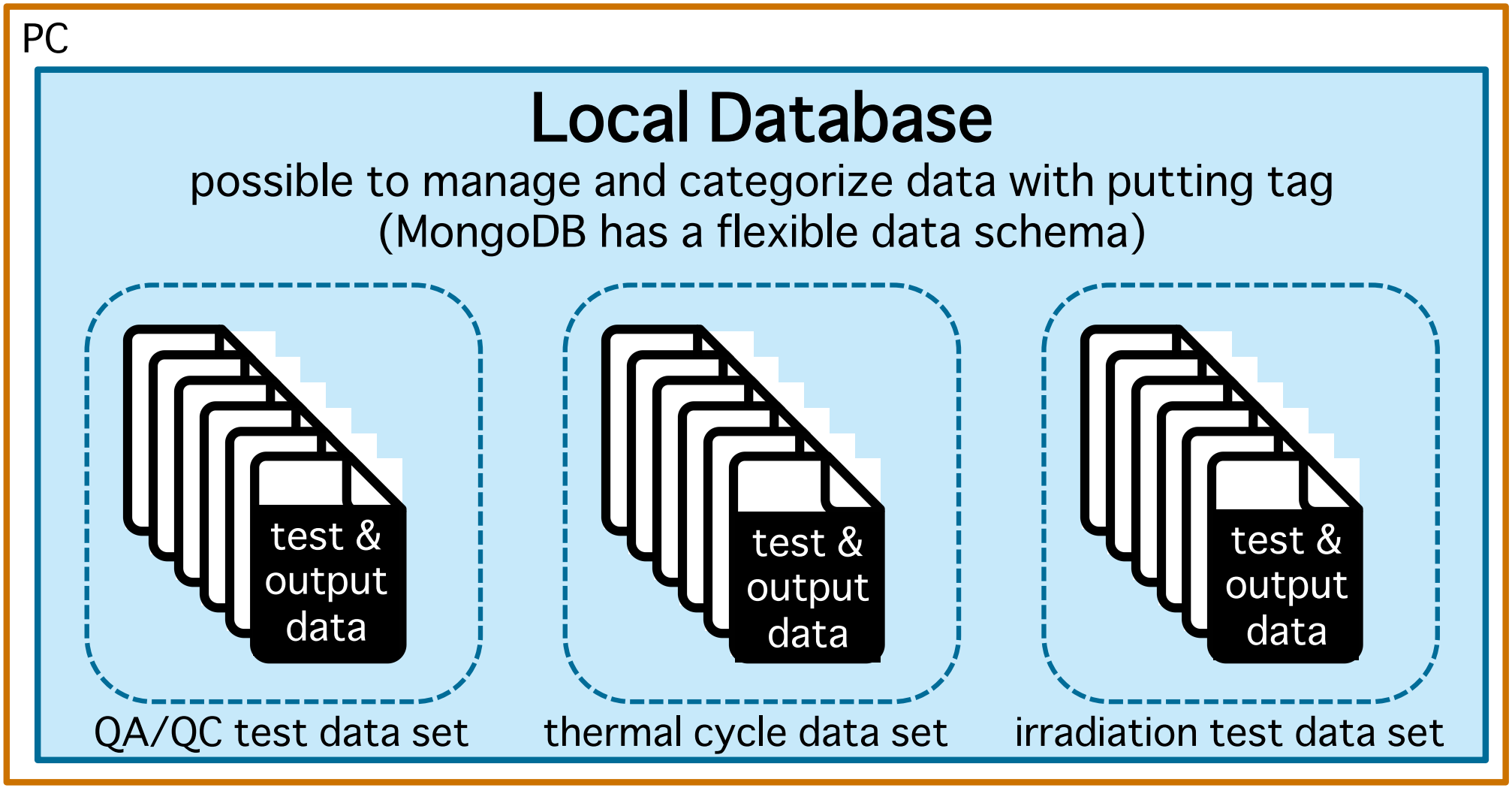
Data in LocalDB can be retrieved and checked with some tools immediately

- Purpose: Data management and automatic storage



Overview of Local DB (3/3)

- Local Database could be helpful to manage data not of only QA/QC but also other tests such as irradiation tests



Open features in development

- welcome to receive feedback! -

- DCS Storing
- Config Retriever
- Synchronization Tool
- Display by Viewer Application

DCS Storing

- Main design concern

  DCS setup (power supply, temperature… how to communicate with them, etc.) will be very different setup-by-setup!
- From LocalDB point of view:
  - Provide **a generic format** to associate any number/type of DCS information to each scan result
  - **User specified which information to be recorded**

- (Work in progress)
  - Specify **a log file format** that DCS controllers should write out (developer: S. Yamagaya)
  - The path to the log file and DCS type are described in a DCS configuration file in Yarr
  - Each scan result will be pushed to the local DB without DCS
  - User specifies the algorithm to find out the corresponding timestamp of the scan from the DCS log file
  - The series of DCS log will be stored by a separated script than scanConsole

DCS Storing

- Enable to store DCS information while scanning into Database automatically if DCS log file is prepared in the specific format

DCS log file

```
6
hv_v hv_i VDDD_v VDDD_i VDDA_v VDDA_i
null null CV CV CV CV
-80 5e-06 1.2 0.2 1.5 0.5
19-04-25_21:11:51 +900 -79.9716 -1.05318e-06 1.19936 0.07946 1.50016 0.04028
19-04-25_21:12:01 +900 -79.9709 -1.04204e-06 1.19939 0.07945 1.50024 0.04025
19-04-25_21:12:11 +900 -79.9705 -1.04654e-06 1.1993 0.07946 1.50034 0.04027
19-04-25_21:12:20 +900 -79.9724 -1.05562e-06 1.19942 0.07946 1.50036 0.0403
19-04-25_21:12:30 +900 -79.9704 -1.04843e-06 1.19928 0.07942 1.50022 0.04027
19-04-25_21:12:40 +900 -79.9708 -1.05706e-06 1.1993 0.07943 1.50018 0.04026
19-04-25_21:12:50 +900 -79.9714 -1.06087e-06 1.19933 0.07944 1.50025 0.04027
19-04-25_21:13:00 +900 -79.9706 -1.03796e-06 1.19919 0.07946 1.50032 0.04029
19-04-25_21:13:10 +900 -79.971 -1.05728e-06 1.19915 0.07944 1.50019 0.04028
19-04-25_21:13:20 +900 -79.9701 -1.03978e-06 1.19918 0.07946 1.50026 0.04029
```

<YARR SW>

scanConsole -I DCSCfg.json

DCSCfg.json

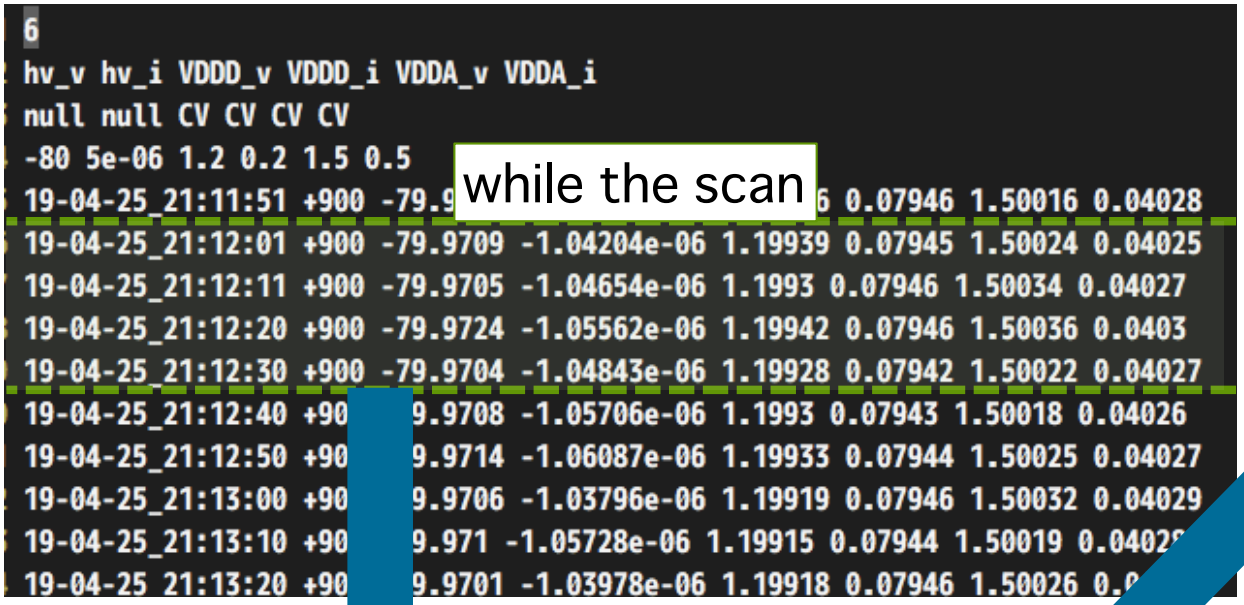
```
{
    "assembly": {
        "stage": "Testing"
    },
    "environments": [
        {
            "key": "lv",
            "value": 1.8,
            "description": "Low voltage [V]",
            "path": "/tmp/lv.dat"
        }
    ]
}
```

DCS keyword

path to log file

DCS Storing

- Enable to store DCS information while scanning into Database automatically if DCS log file is prepared in the specific format



<YARR SW>

scanConsole -I DCSCfg.json

upload test data into DB with generating cache to upload DCS info

testRun : Data ID
DCS : path to log file

DCS data while scan is uploaded following cache by specific tool

Local DB

dcs data

data_id: XXX
lv : [{timestamp, value},
{timestamp, value}]

linked by id

test run data

start time : …
finish time : …
env_id: XXX

DCS Storing

- Enable to check stored DCS data in browser

Local DB

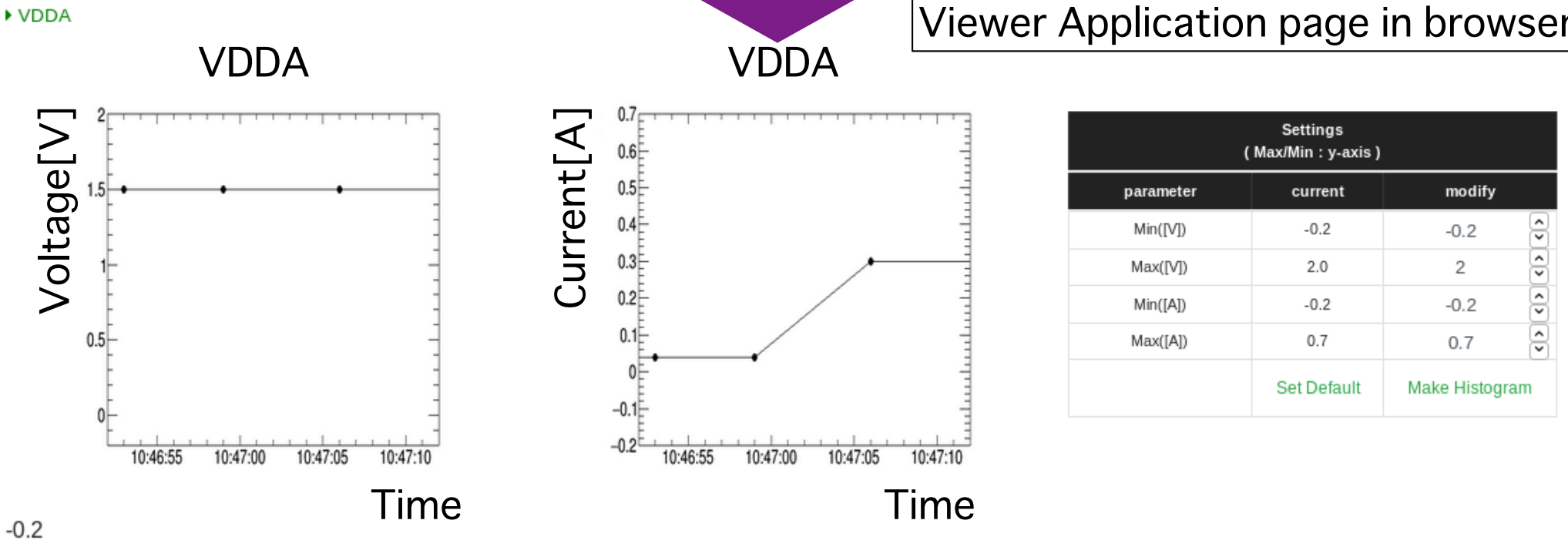
dcs data

data_id: XXX
lv : [{timestamp, value},
{timestamp, value}]

test run data

start time : …
finish time : …
env_id: XXX

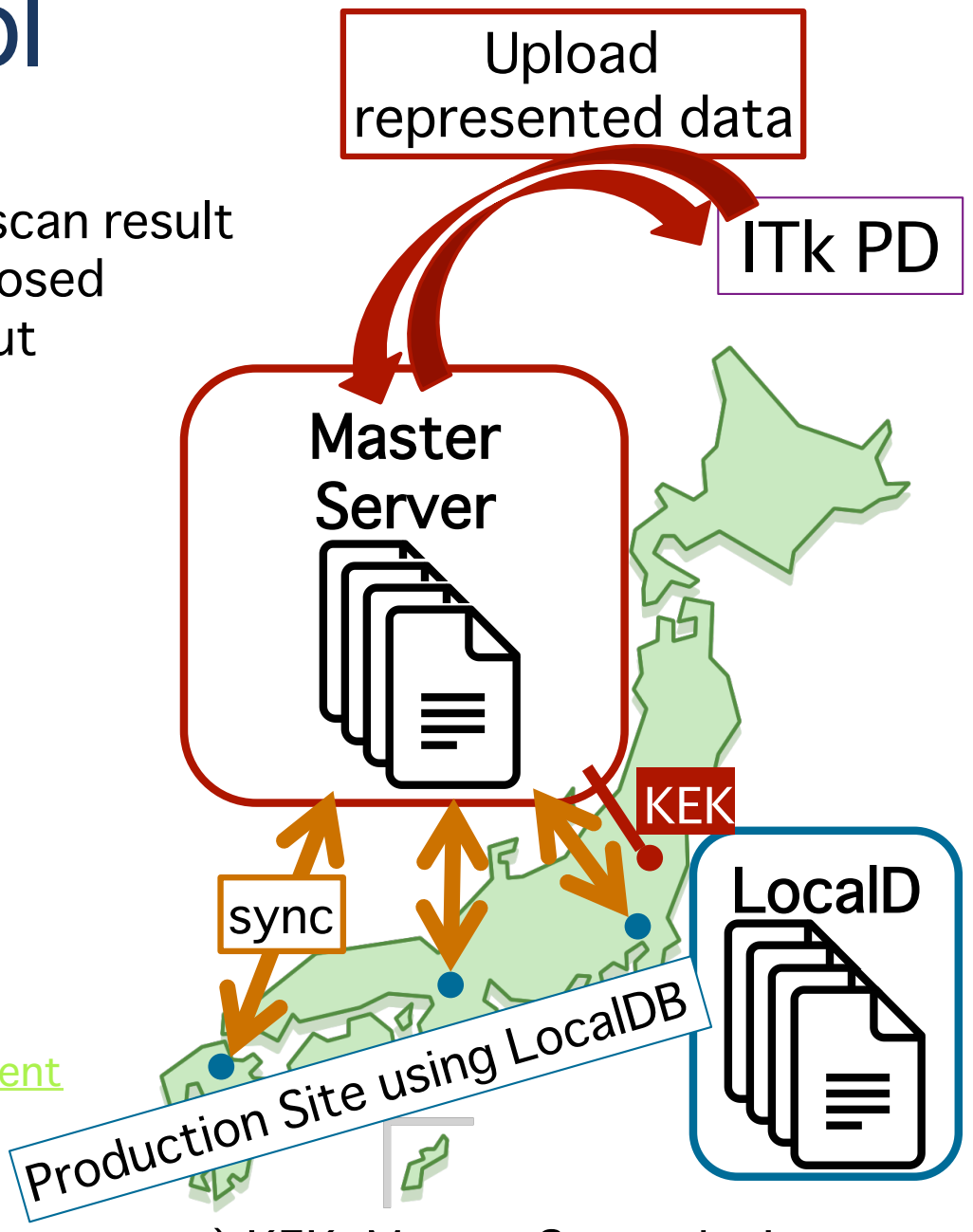
Viewer Application page in browser



Settings (Max/Min : y-axis)		
parameter	current	modify
Min([V])	-0.2	-0.2
Max([V])	2.0	2
Min([A])	-0.2	-0.2
Max([A])	0.7	0.7
	Set Default	Make Histogram

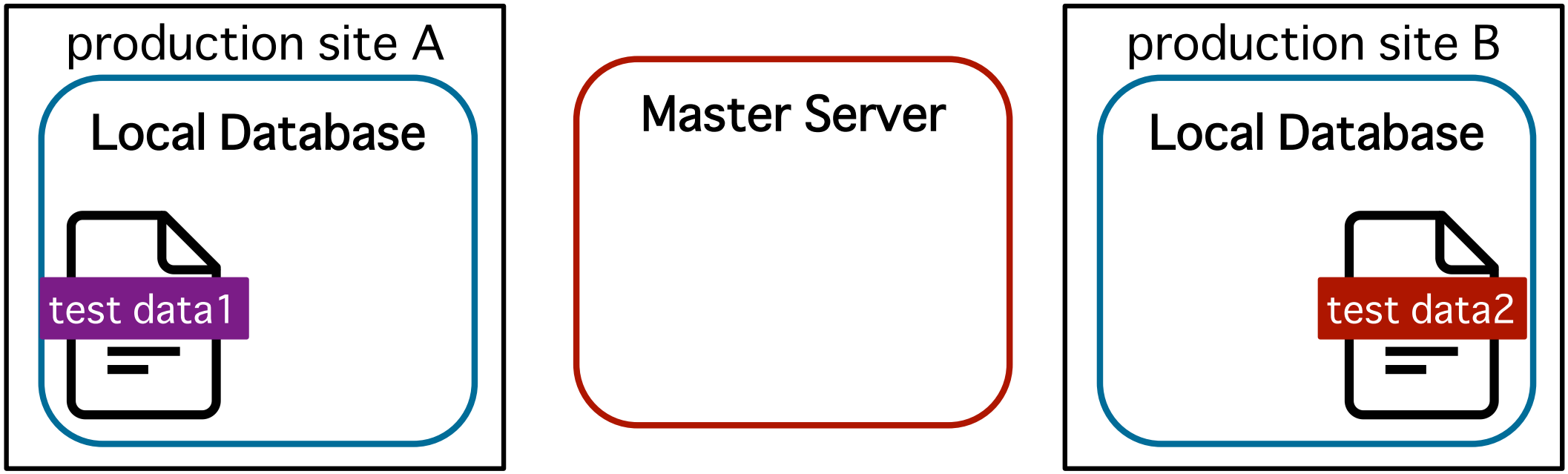
Synchronization Tool

- Main design concern
  - Local DB : primary data buffer of the scan result before uploading to ITk PD → be supposed to be built for each production site, but might not support large-scale storage
- Master server (special big local DB) : aggregation of distributed Local DB data (e.g. hosting >>10TB) → upload represented data into ITk PD
- Production site using LocalDB can upload and retrieve test data to Master Server directly
- Also production site can check whole data of Master Server in browser
  viewer) http://atlaspc5.kek.jp/yarrdb/development



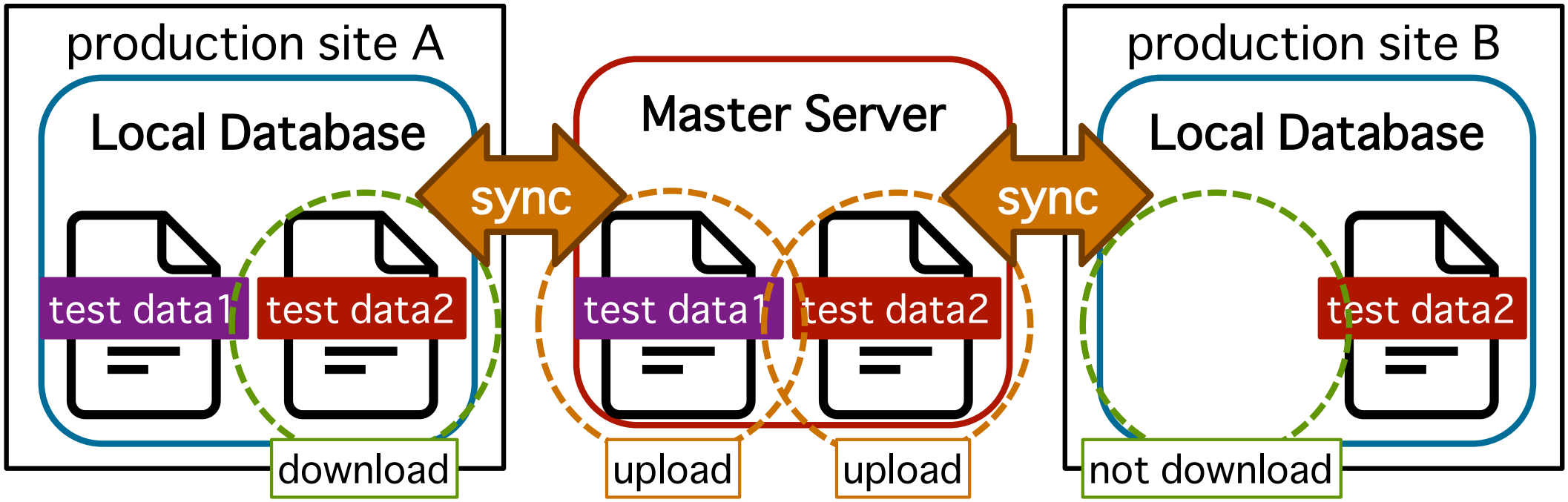
e.g.) KEK: Master Server in Japan

Synchronization Tool



- (Work in progress)
  - Master Server has whole data by synchronizing from each production site
  - Synchronization Tool (developer: E. Kim) transfer only the difference data between Local DB and Master Server like `**rsync**`
  - Master Server should have whole data set, but Local DB is not, so Local DB can download data partially from Master Server if required

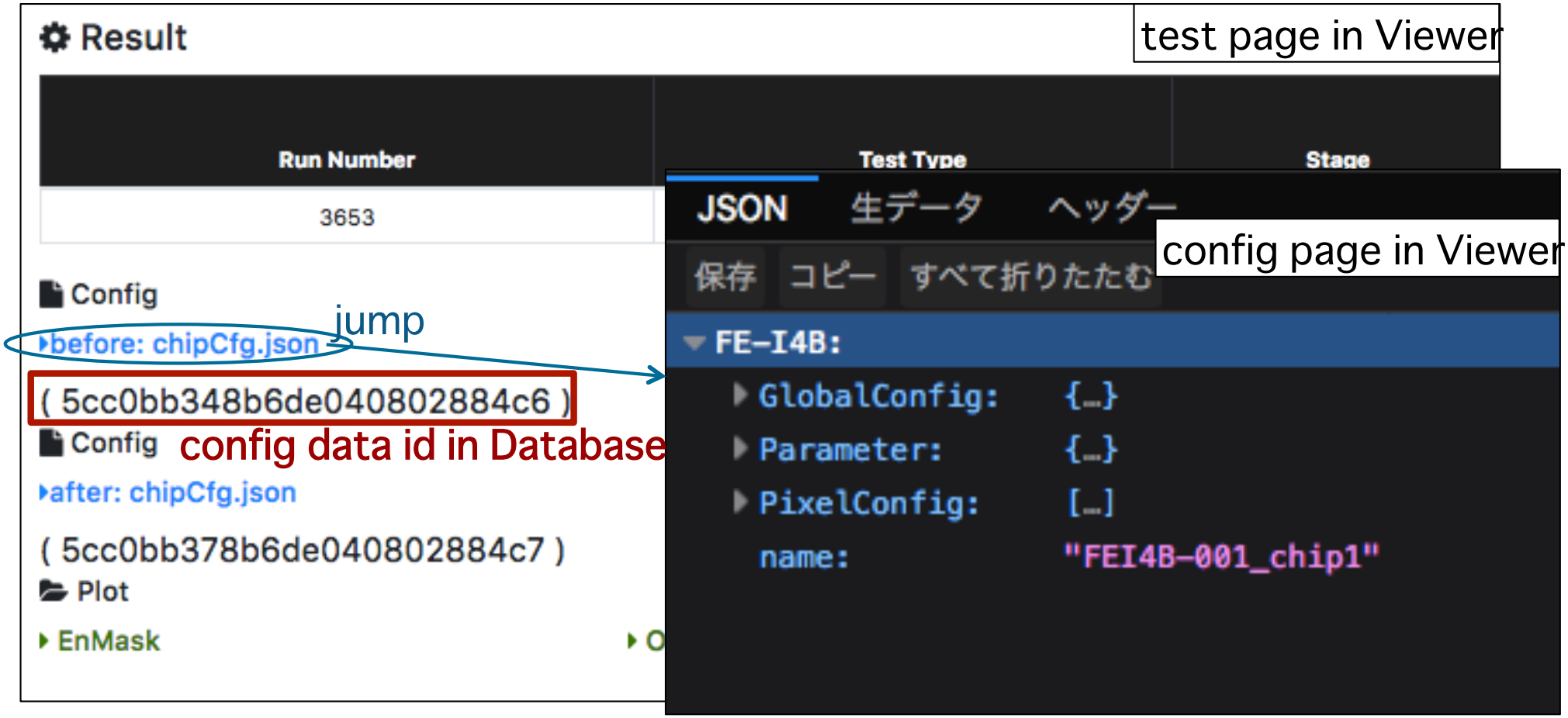
Synchronization Tool



- (Work in progress)
  - Master Server has whole data by synchronizing from each production site
  - Synchronization Tool (developer: E. Kim) transfer only the difference data between Local DB and Master Server like `**rsync**`
  - Master Server should have whole data set, but Local DB is not, so Local DB can download data partially from Master Server if required

Config Retriever

- (Work in progress)
  - The chip config: displayed with data id in browser
    → retrieve it from Local DB by querying with data id and overwrite some parameters (currently chipId and chipName)



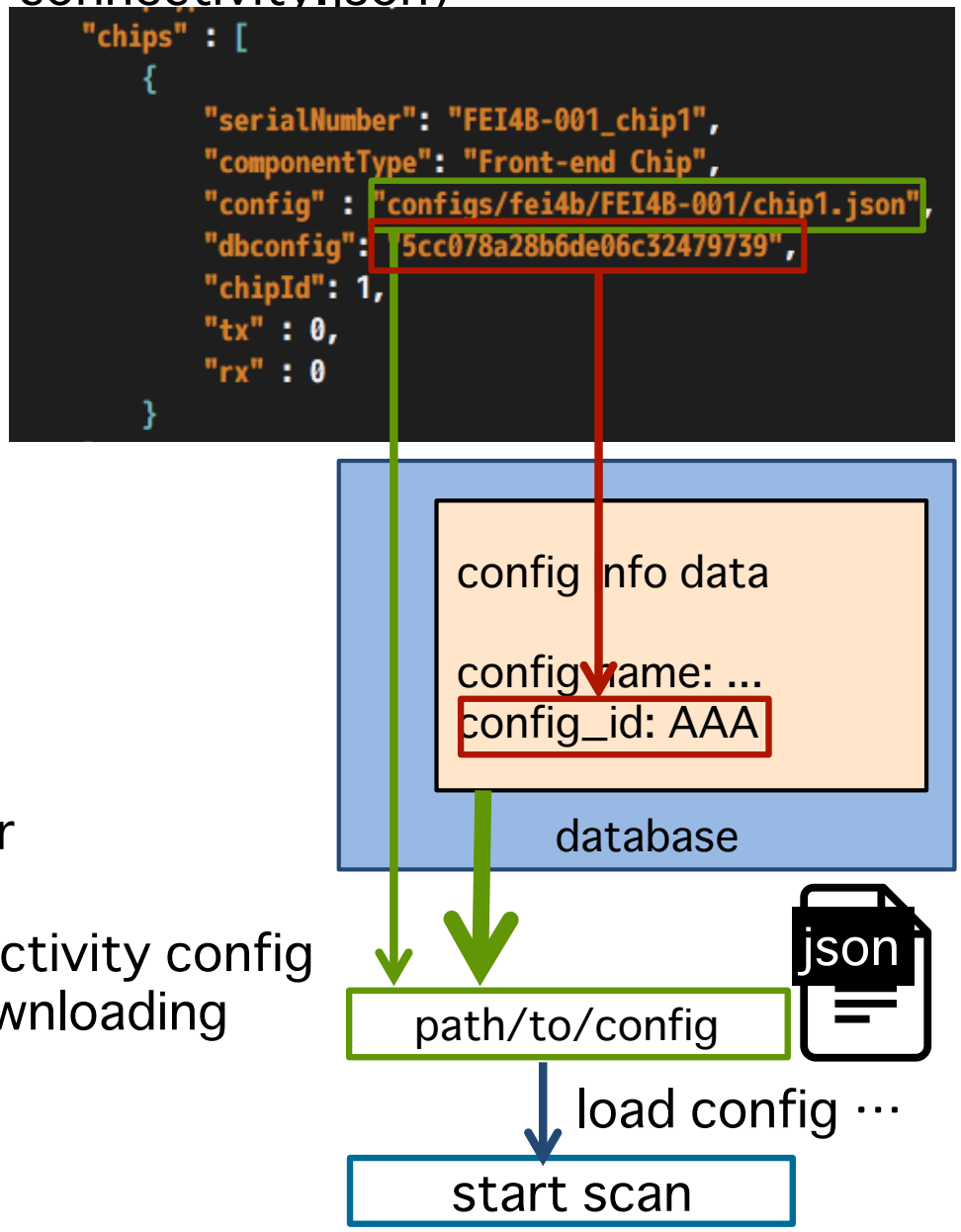
Config Retriever

- scanConsole
  - ⓪ Give config id in Viewer (dbconfig)
  - ① download config file in DB by config id
  - ② Create config file from DB
    → path/to/config
  - ③ Load config file (path/to/config)

- config creation
  - ◆1. Get config data from DB
  - ◆ 2. Modify some values in config data for
    - chip name → serial number
    - chipId → change to ID written in connectivity config

    so enable to scan continuously after downloading

connectivity.json)



Viewer Application (1/2)

- Enable to check the results in LocalDB for each Modules using Viewer Application
- App is running in local machine so the results can be checked only in internal if not setting to open the page

PIXEL MODULE DB　Toppage　Sign in

ITk database for Yarr

Summary

the asic type tested is showed

the table is hidden by default, and opened by click the asic type ⬇

FE-I4B (8 modules)

Module	Chips				Created Date	Cuurent Stage
QU-09	QU-09_chipId1	QU-09_chipId2	QU-09_chipId3	QU-09_chipId4	2018/11/01 01:15:04	None
QU-10	QU-10_chipId1	QU-10_chipId2	QU-10_chipId3	QU-10_chipId4	2018/11/01 02:09:20	None
KEK-145	KEK-145_chipId1	KEK-145_chipId2	KEK-145_chipId3	KEK-145_chipId4	2018/11/01 02:09:32	None
KEK-151	KEK-151_chipId1	KEK-151_chipId2	KEK-151_chipId3	KEK-151_chipId4	2018/11/01 02:17:36	None
KEK-149	KEK-149_chipId1	KEK-149_chipId2	KEK-149_chipId3	KEK-149_chipId4	2018/11/01 02:19:42	None
QU-11	QU-11_chipId1	QU-11_chipId2	QU-11_chipId3	QU-11_chipId4	2018/11/01 02:20:25	None
KEK-150	KEK-150_chipId1	KEK-150_chipId2	KEK-150_chipId3	KEK-150_chipId4	2018/11/01 02:22:22	None
	_chipId1	_chipId2	_chipId3	_chipId4	2018/11/03 00:04:36	None

RD53A (1 modules)

Module	Chips				Created Date	Cuurent Stage
0x0728_module	0x0728				2019/01/18 23:26:54	None

Viewer Application (2/2)

scan result page

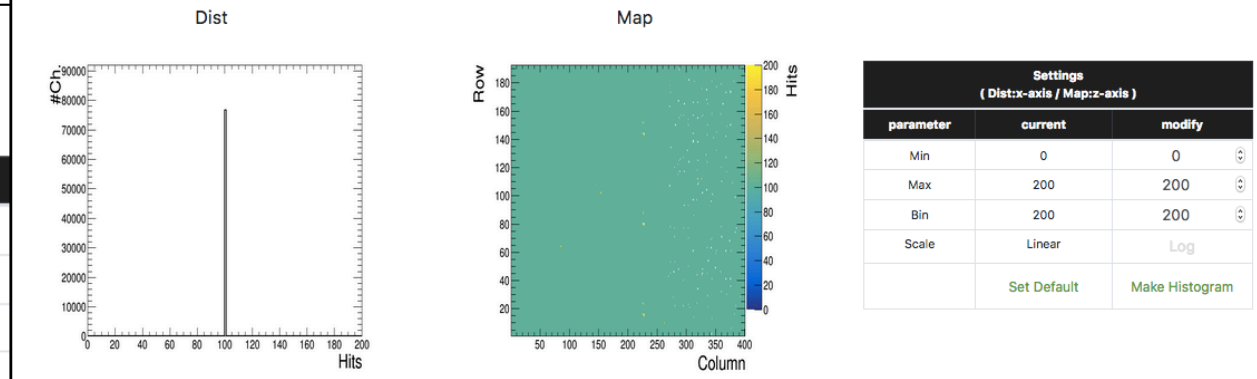
Result

Run Number	Test Type	Stage	Environment: Low voltage [V]	Institution	Operator
1155	std_analogscan		1.8	Tokyo_Institute_Of_Technology	Arisa_Kubota

Plot

ROOT

OccupancyMap



Settings (Dist:x-axis / Map:z-axis)		
parameter	current	modify
Min	0	0
Max	200	200
Bin	200	200
Scale	Linear	Log
	Set Default	Make Histogram

module Summary : JP-2

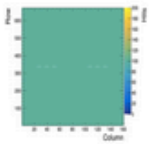
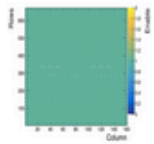
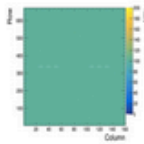
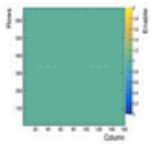
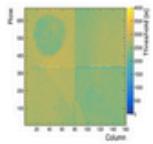
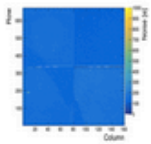
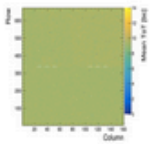
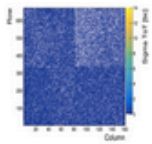
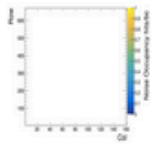
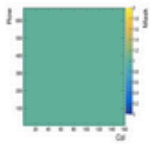
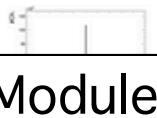
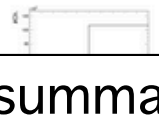
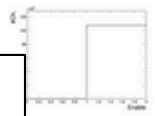
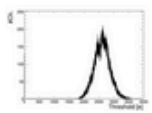
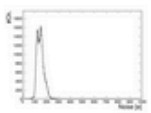
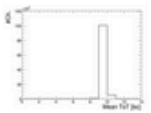
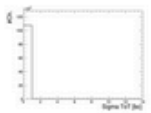
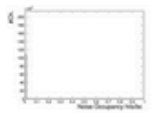
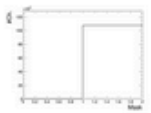
Index

Item	
Serial Number	JP-2
FE type	FE-I4B
Module	JP-2
Chips	JP-2_chipId1 JP-2_chipId2 JP-2_chipId3 JP-2_chipId4

Test summary

demonstrator

representing plots for each stage

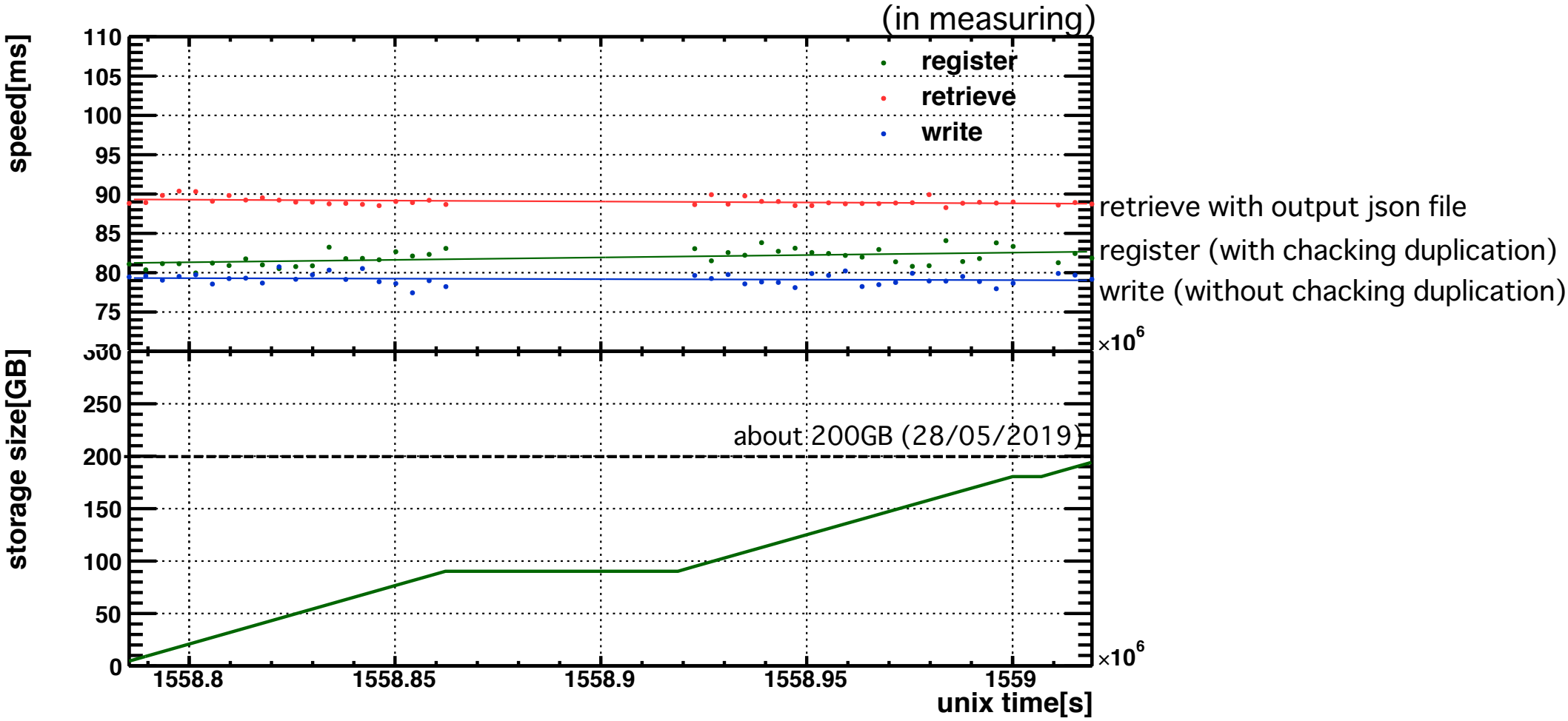
digitalscan		analogscan		thresholdscan		totscan		noisescan		selftrigger
OccupancyMap	EnMask	OccupancyMap	EnMask	ThresholdMap	NoiseMap	MeanTotMap-0	SigmaTotMap-0	NoiseOccupancy	NoiseMask	None.
										None.
										None.

Module summary page

Future Plan

Performance Test

- Performance of Local DB is currently devently benchmarked and optimized via stress tests
  - measure the speed to register/retrieve config file to/from Local DB



Feedbacks and TODOs

- Make the introduction in Wiki page more user friendly and clearly

- Operation model of Local DB in production is still quite undetermined and further discussions will be needed (depending on the storage size available to support in each local instance)
  - Nevertheless the basic embedding of localDB in Yarr is becoming matured.
- Component (Chip/Module) Registration:
  - (currently) Components are registered by manually and there should be many mistakes and troubles
  - (suggestion) Components would be registered in ITk PD so Local DB just retrieve the components data and test following these data without registration
- Config retriever:
  - (currently) We can retrieve config from Local DB with config id
  - (suggestion) Enable to retrieve config from Master Server directly without storing into Local DB (just retrieve and load it!)

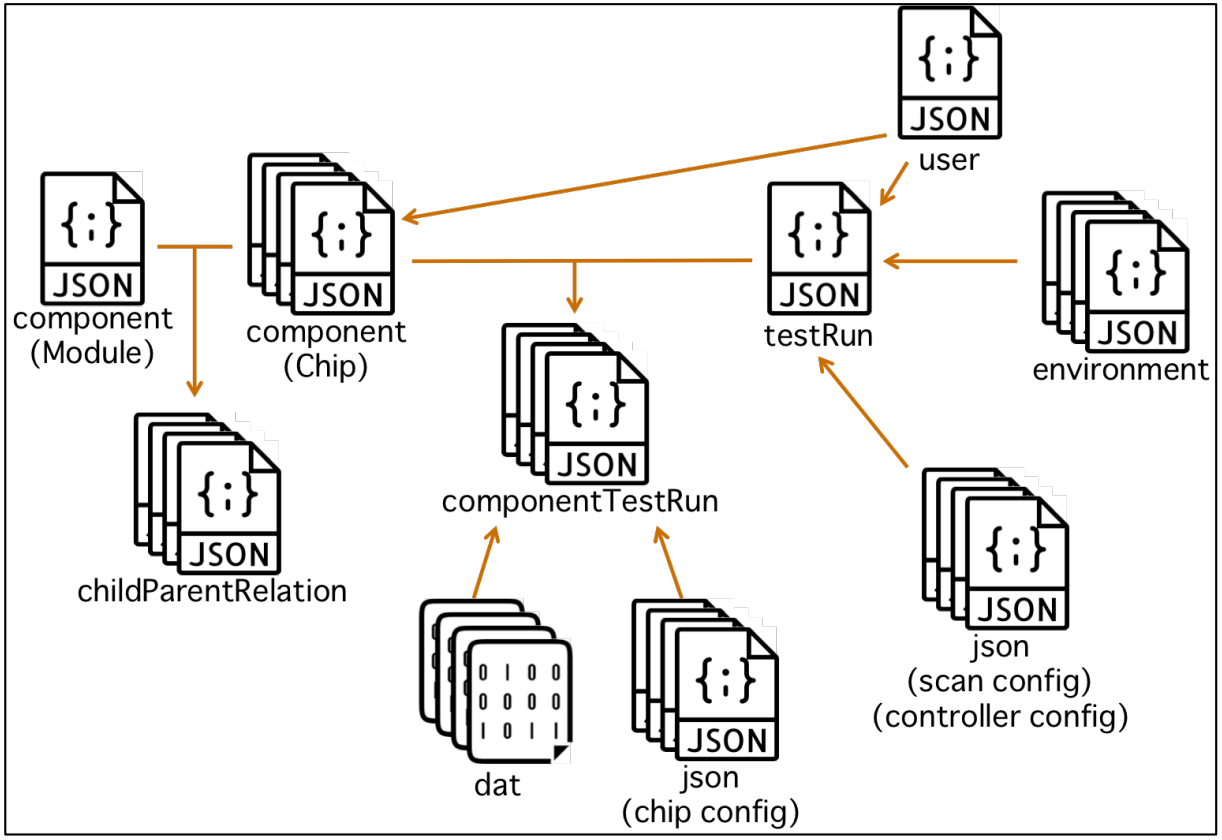
Backup

Short Introduction about MongoDB and Data Structure

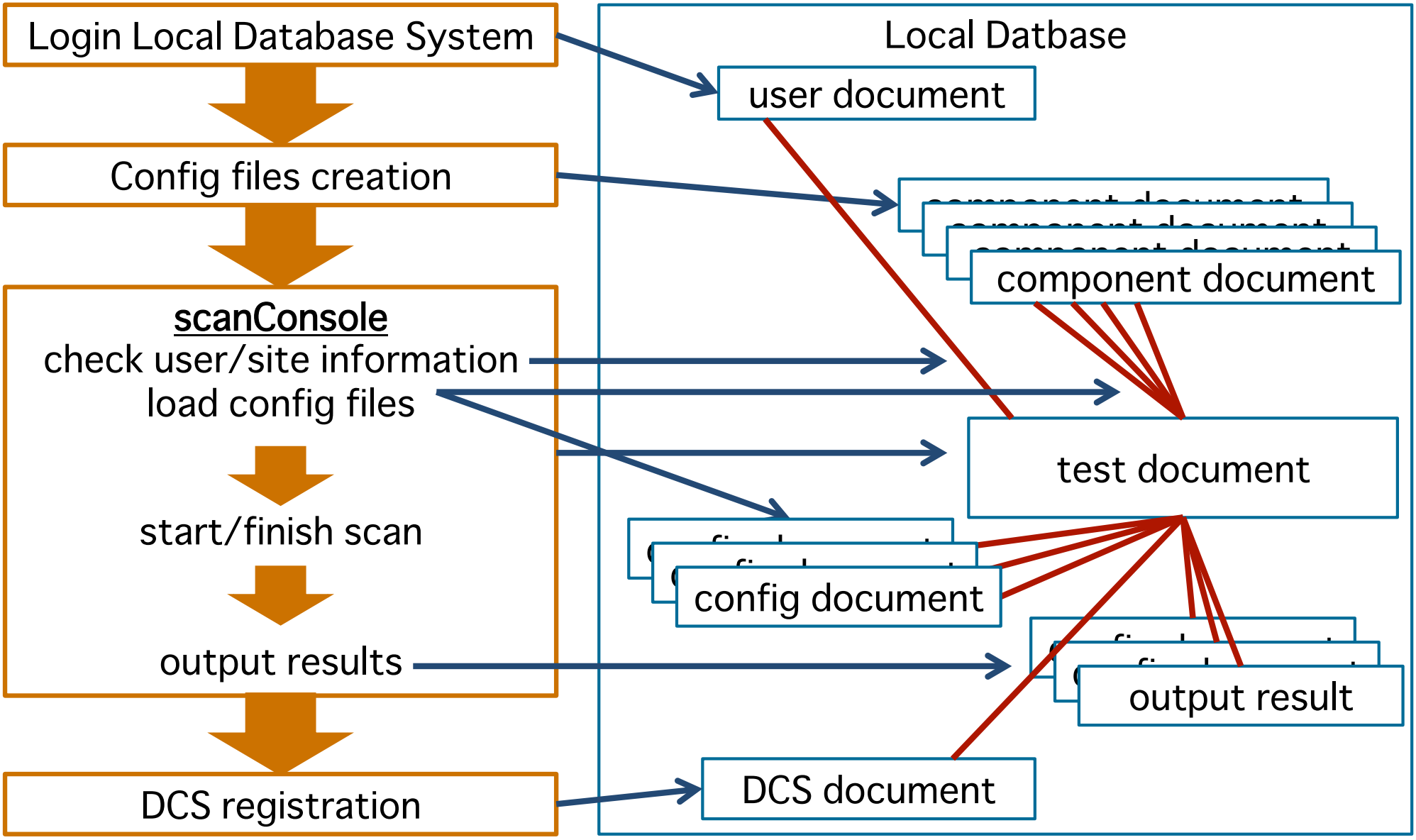
MongoDB https://docs.mongodb.com/

- Open source NoSQL Database
  - Flexible data schema so not need to determine schema before storing data
  - The documents (data) as json like format
  - It can be used as like relational Database with connecting by id
  - Many Drivers (Python, C++, JavaScript …)

Local DB Data Schema



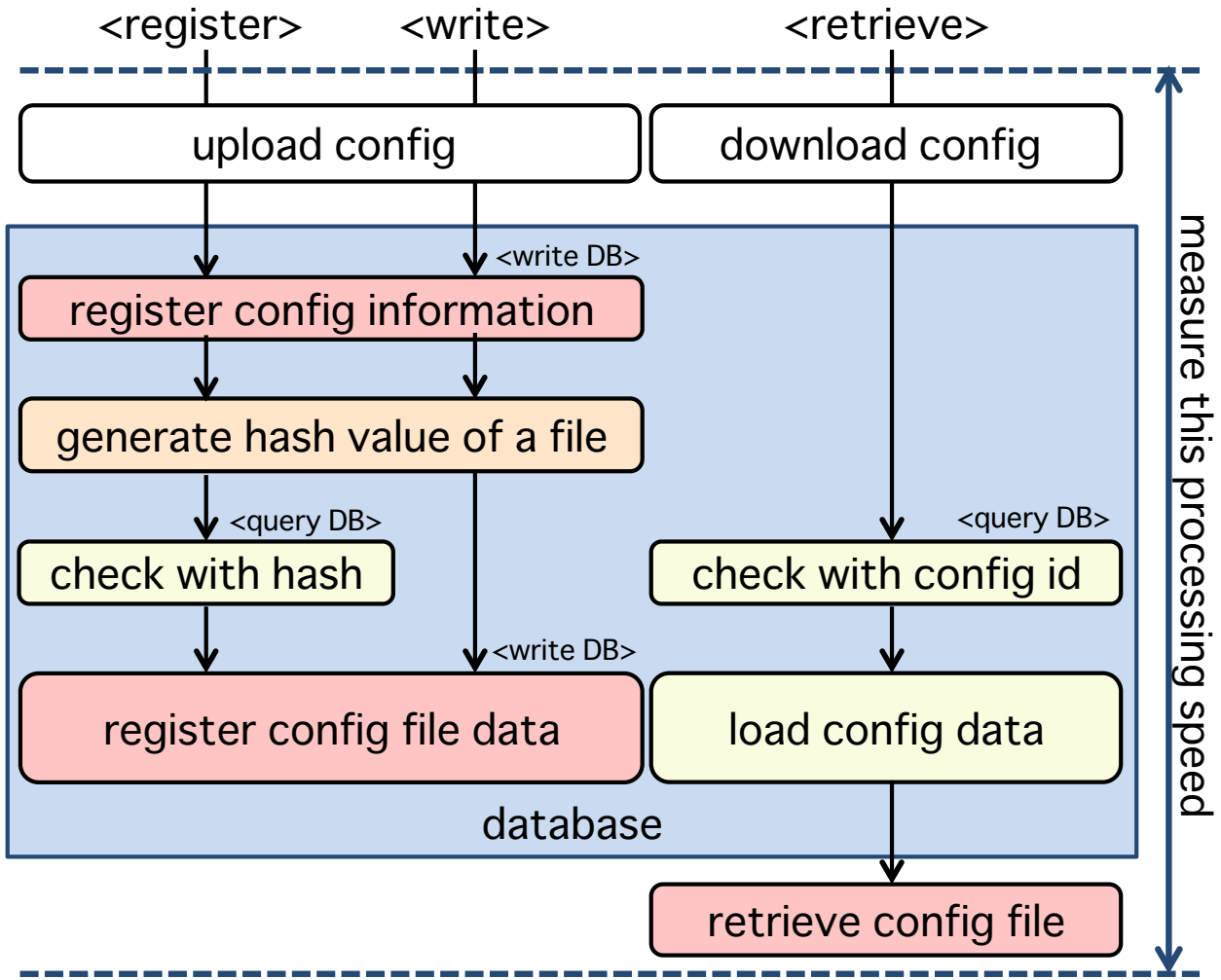
YARR scanConsole with LocalDB SW



Database Performance

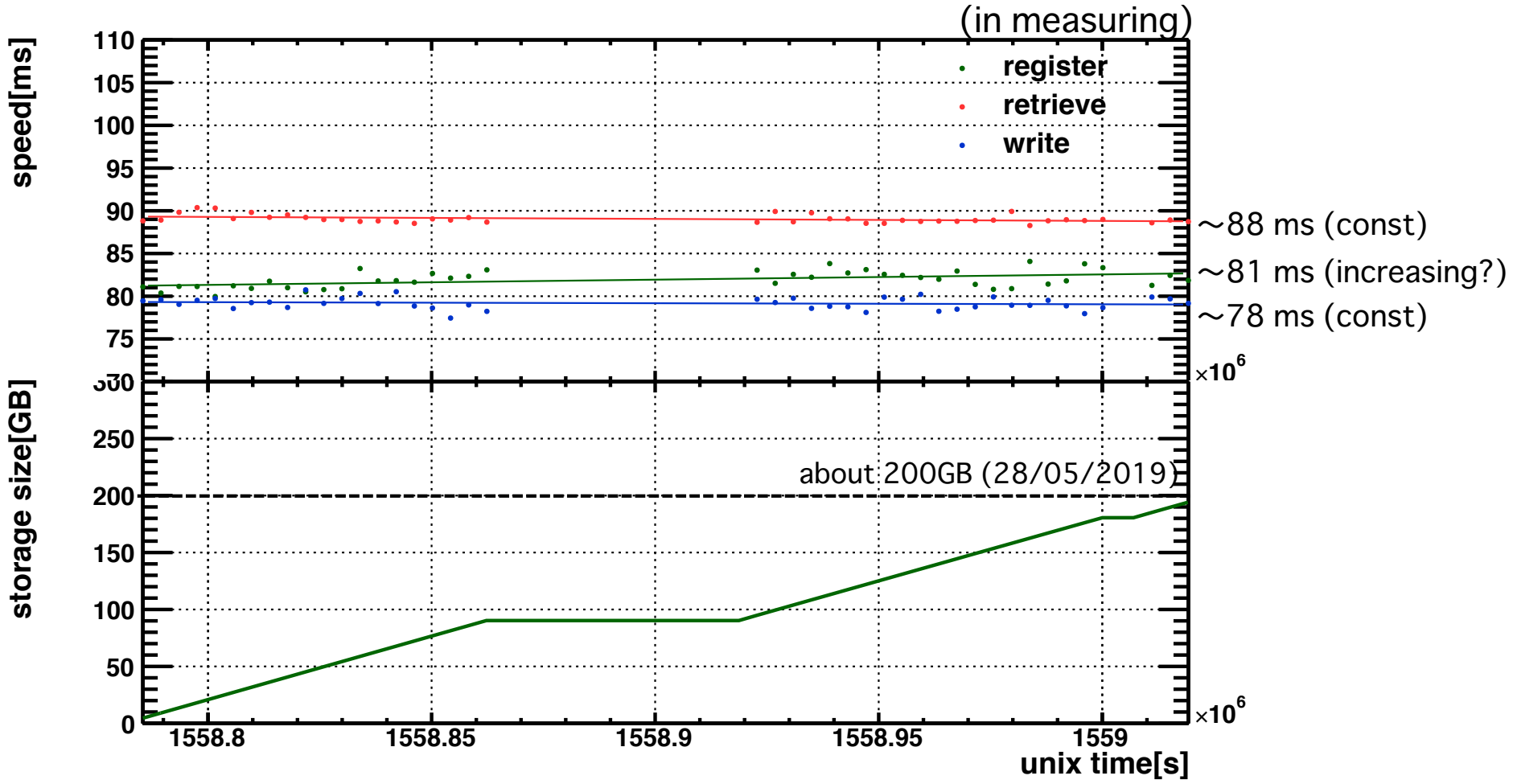
Stress test

- Measuring the speed to upload/download config file into/from Database with increasing storage size to check the performance of the Database



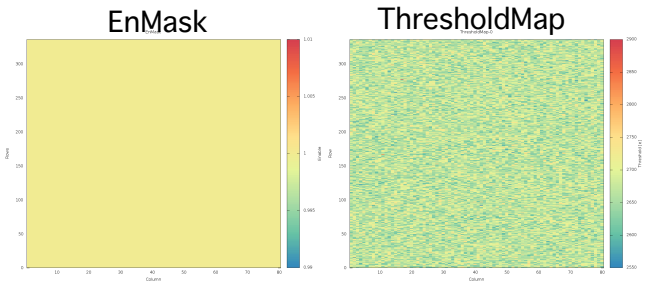
Stress test

- Measuring the speed to upload/download config file into/from Database with increasing storage size to check the performance of the Database

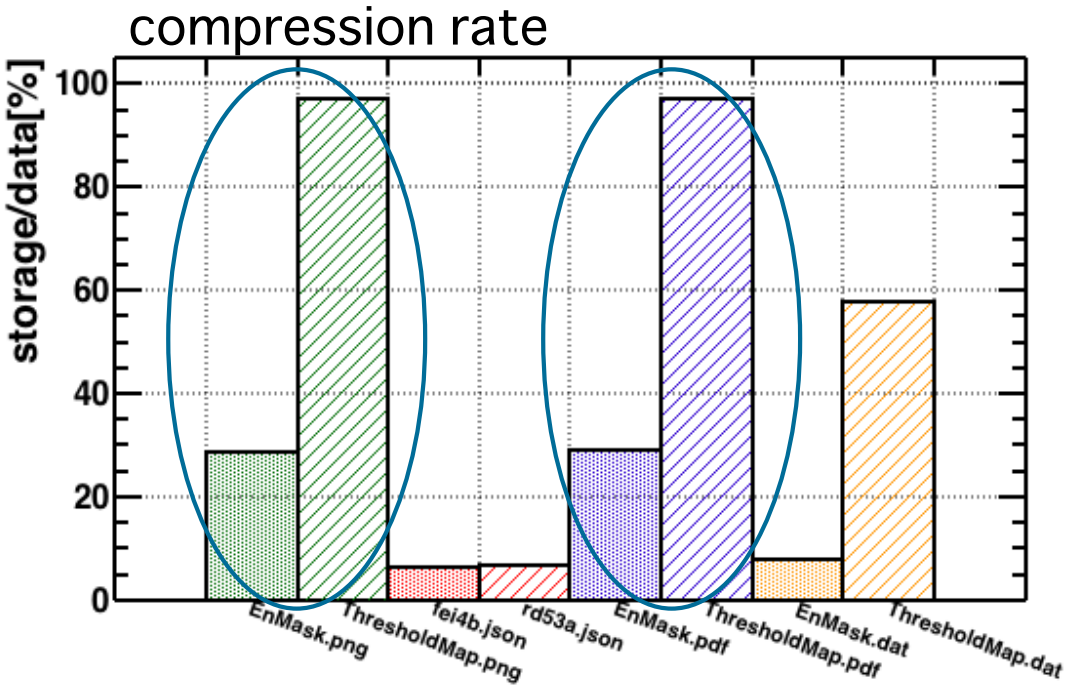


Data Size

- Measure data size after storing 100,000 documents for each data format
  - png/pdf data can't be compressed much
    → not need to store because they can be created by dat file



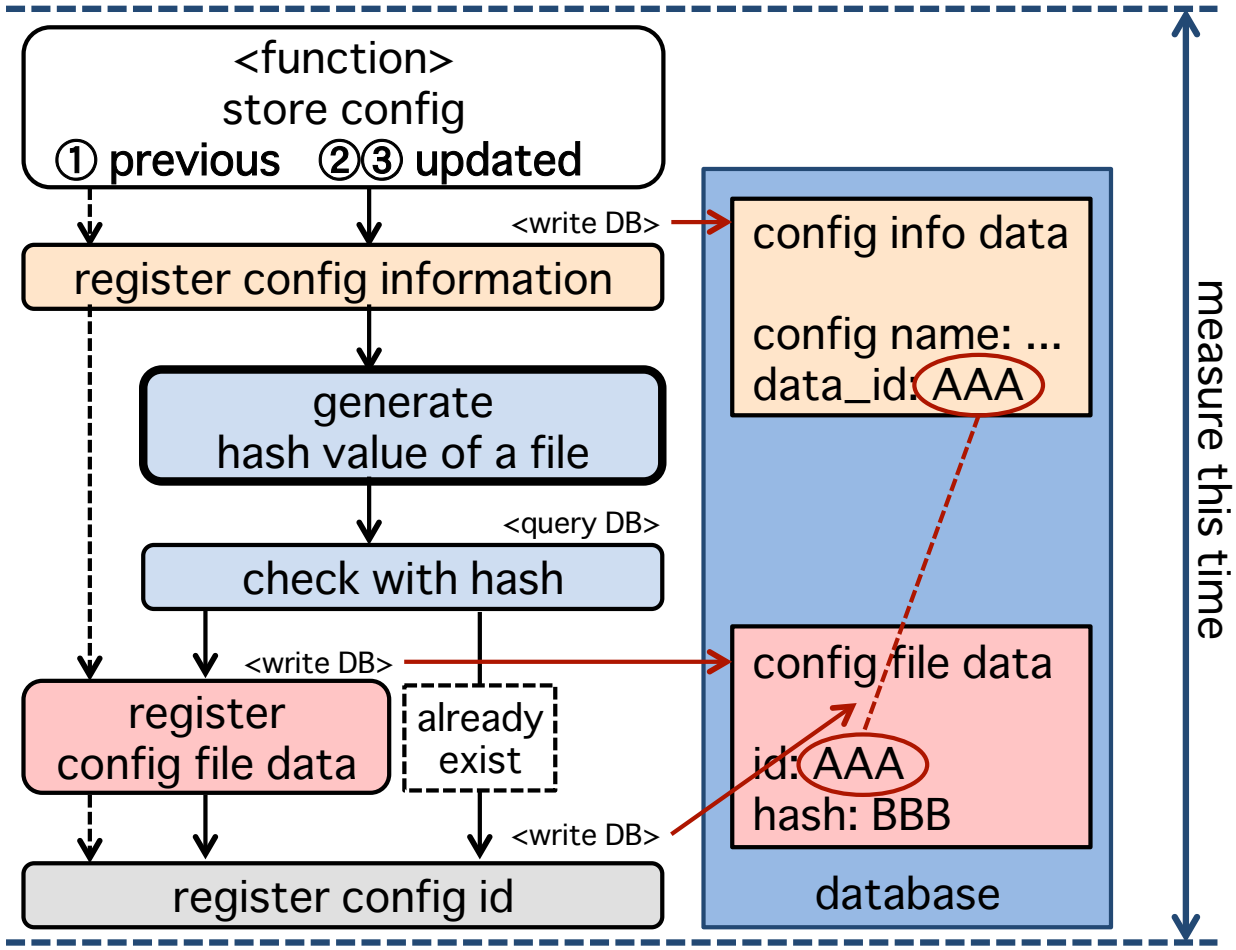
- data
  - dat
    - EnMask.dat (53 KB)
    - ThresholdMap.dat (207 KB)
  - json
    - default_fei4b.json (3.7 MB)
    - default_rd53a.json (670 KB)
  - pdf
    - EnMask.pdf (9 KB)
    - ThresholdMap.pdf (61 KB)
  - png
    - EnMask.png (9 KB)
    - ThresholdMap.png (61 KB)



Config Storing

Config storing -Motivation and Methods-

- Generate a hash value of file in storing config for checking whether the data is already registered by querying with the hash value
- Compare some parameters in defference ways to save config
  - ① previous
  - ② updated
  - ③ updated using index
  - → check …
    - ❖ speed
    - ❖ data size

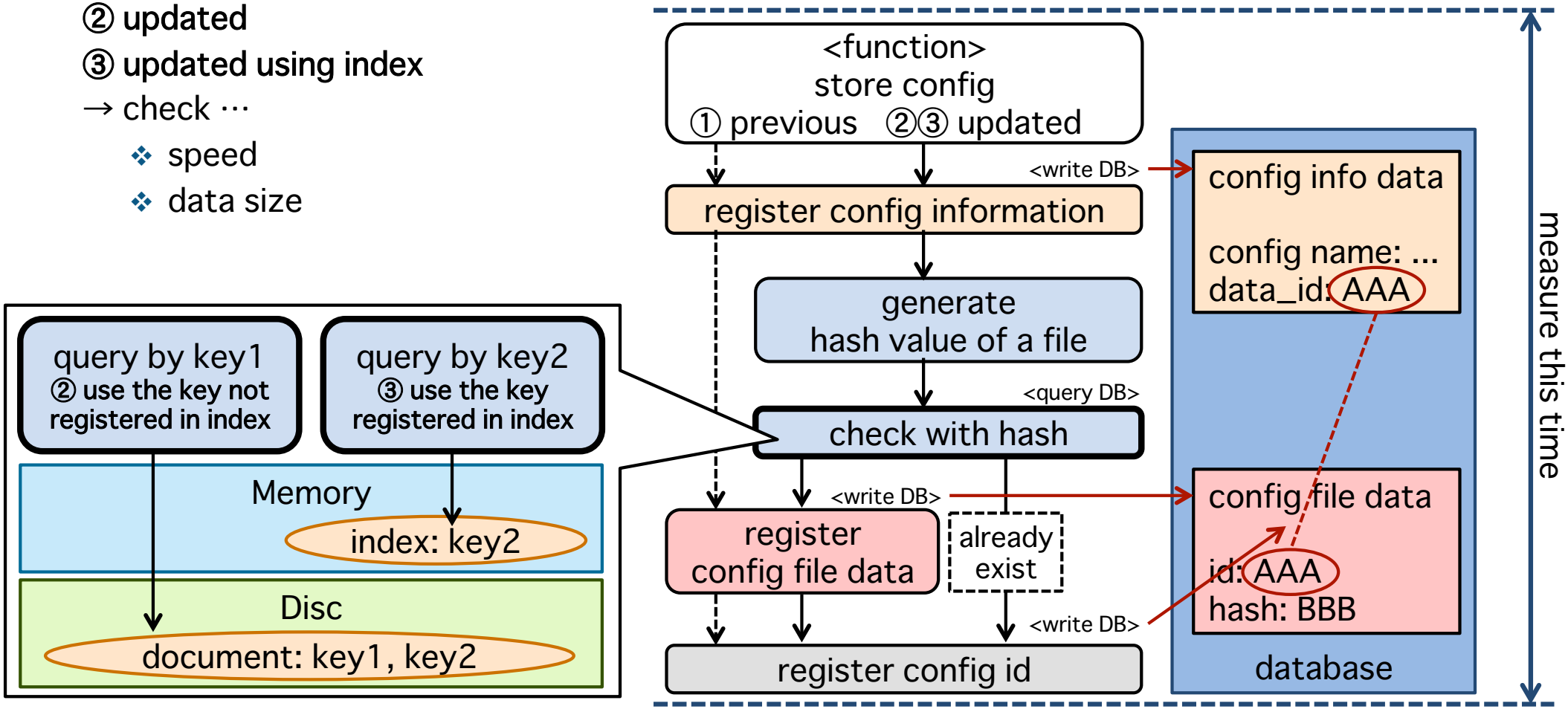


Config storing -Motivation and Methods-

- Generate a hash value of file in storing config for checking whether the data is already registered by querying with the hash value
- Compare some parameters in defference ways to save config

① previous
② updated
③ updated using index
→ check …

- speed
- data size



Config storing -Results-

- Generate a hash value of file in storing config for checking whether the data is already registered by querying with the hash value
- Compare some parameters in defference ways to save config

after 14 scan (full scan for FEI4B) ...

① previous

registered config file data	28 files
data size (before compression)	11,895 KB
storage size	233 KB

② updated

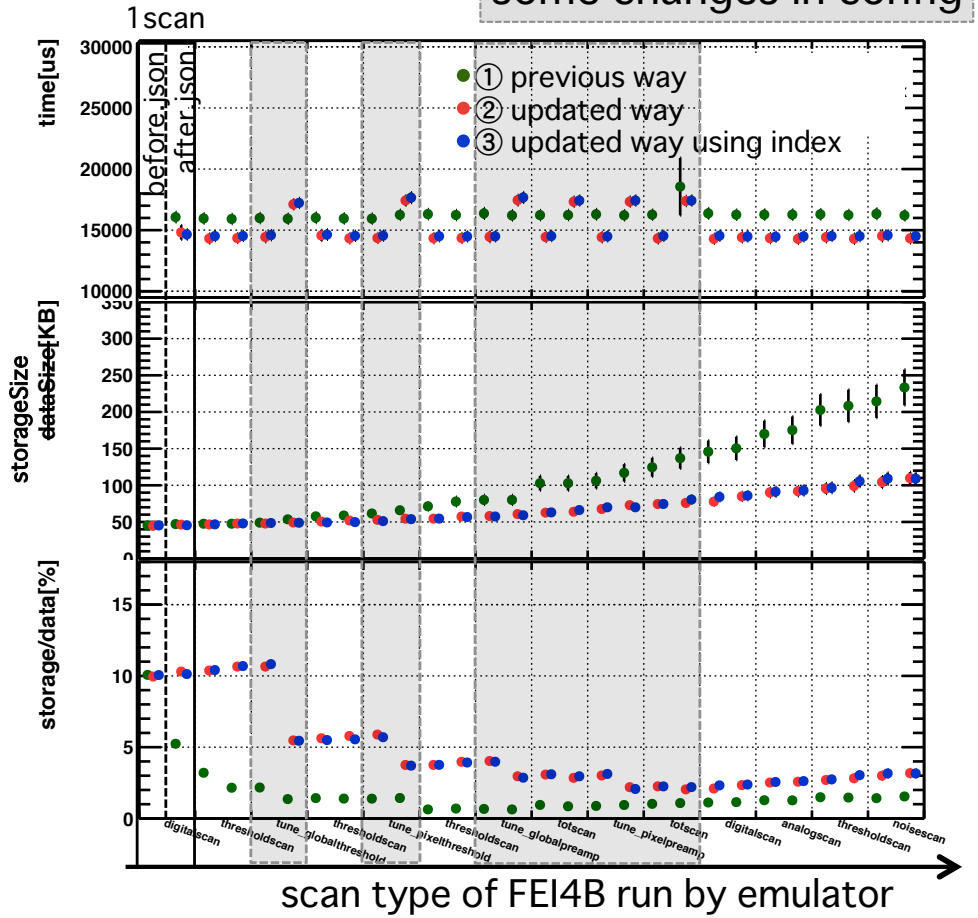
registered config file data	7 files
data size (before compression)	2,978 KB
storage size	110 KB

③ updated using index

registered config file data	7 files
data size (before compression)	2,978 KB
storage size	110 KB

→avoid storing config file after scan which doesn't change config

there was some changes in config



Config storing -Results-

- Generate a hash value of file in storing config for checking whether the data is already registered by querying with the hash value
- Compare some parameters in defference ways to save config

after 14 scan (full scan for FEI4B) ...

① previous

average speed 15.97±0.18 ms

② updated

average speed 15.28±0.17 ms

(register speed 17.34±0.04 ms)

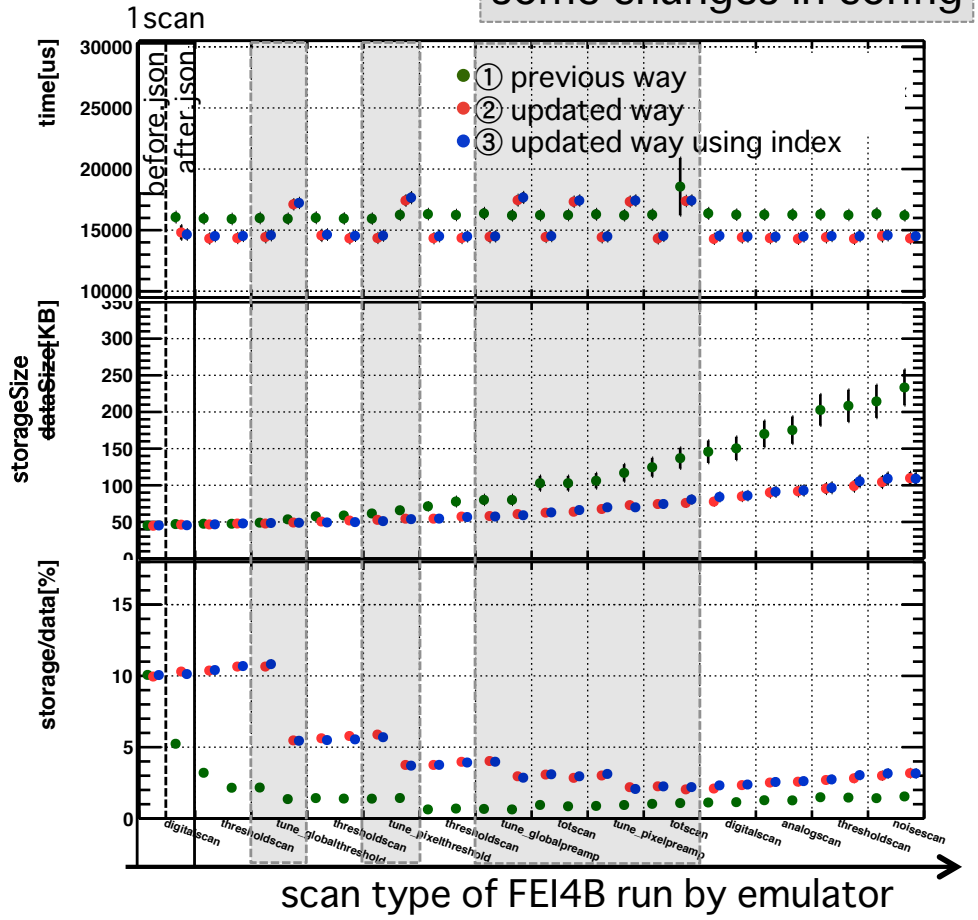
③ updated using index

average speed 15.33±0.17 ms

(register speed 17.47±0.07 ms)

→ there are not much difference in speed

there was some changes in config



Config storing -Results-

- Generate a hash value of file in storing config for checking whether the data is already registered by querying with the hash value
- Compare some parameters in defference ways to save config

after storing 500,000 config files

① previous

average speed --- ms

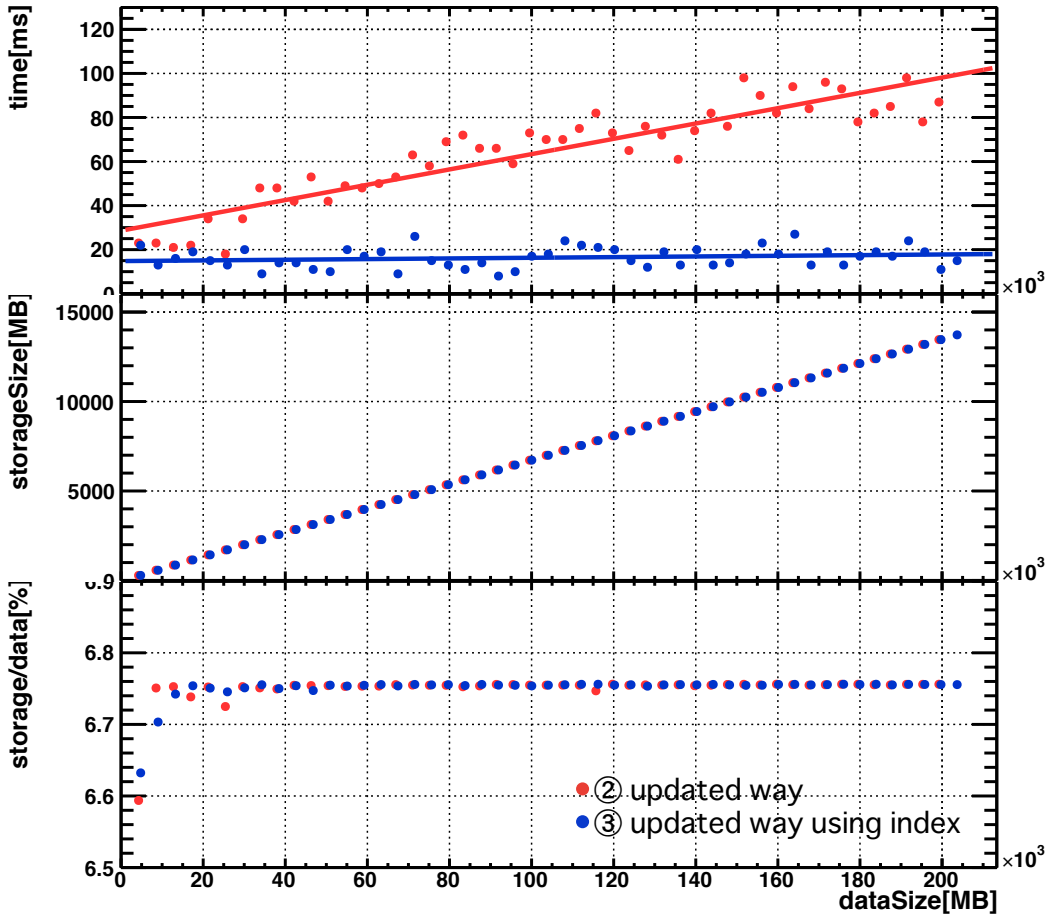
② updated

speed $(1.4\times10^{-4})\times$files+29.0 ms

③ updated using index

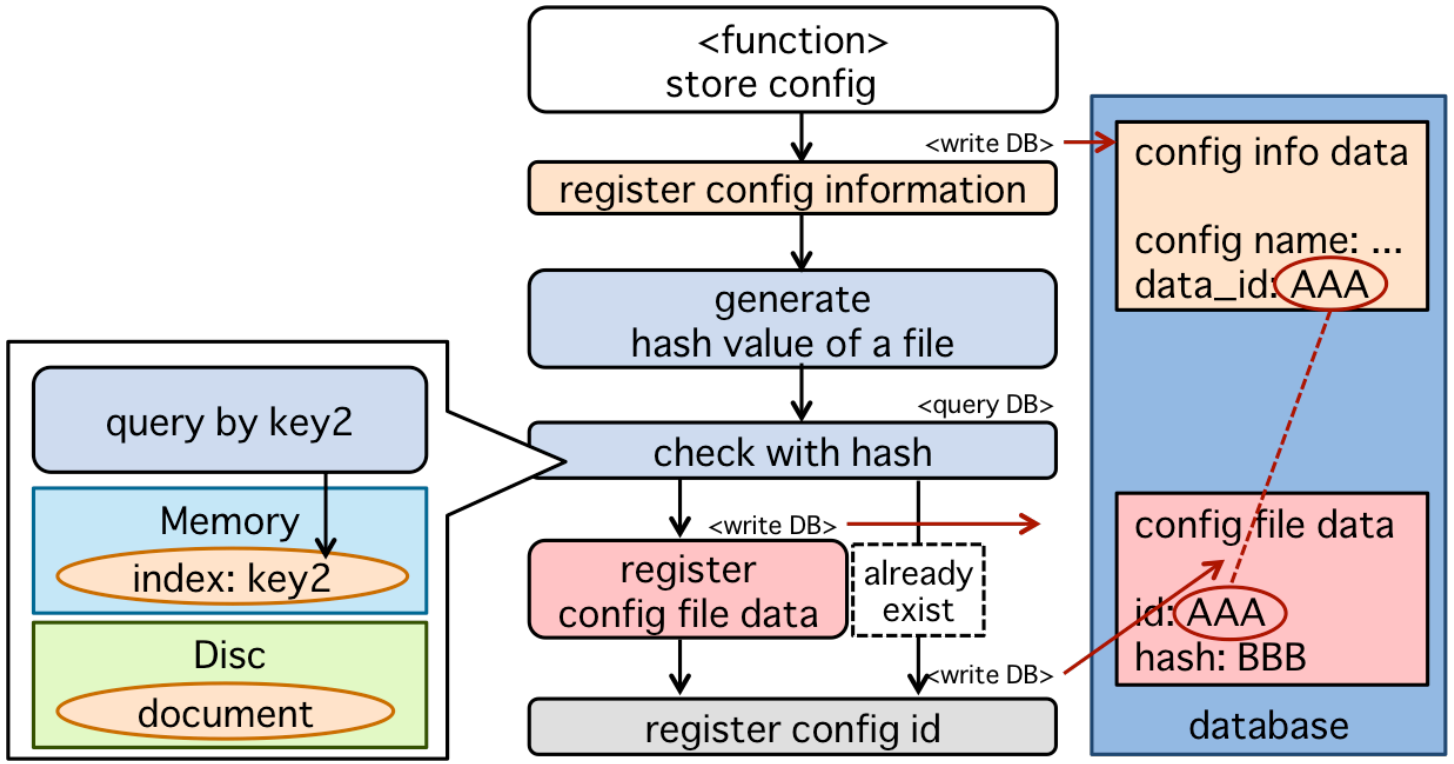
speed $(6.1\times10^{-6})\times$files+14.8 ms

→ the difference increases as the amount of data increases



Config storing -Conclusion-

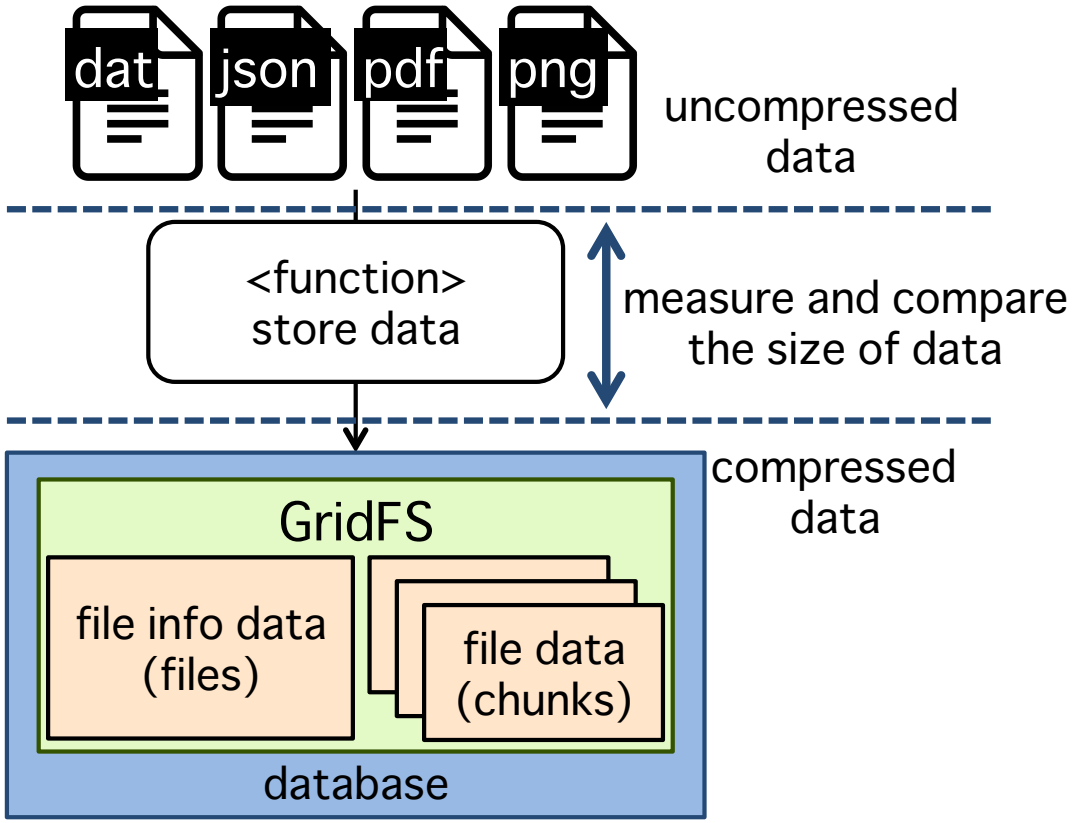
- Checking with the hash value of file can avoid storing data already registered:
  storage size after full scan: 233 KB→110 KB
- The query by the key registered in index is hardly delayed as the amount of data increases:
  register speed after storing 10^6 docs: 169 ms→21 ms



Data Storing

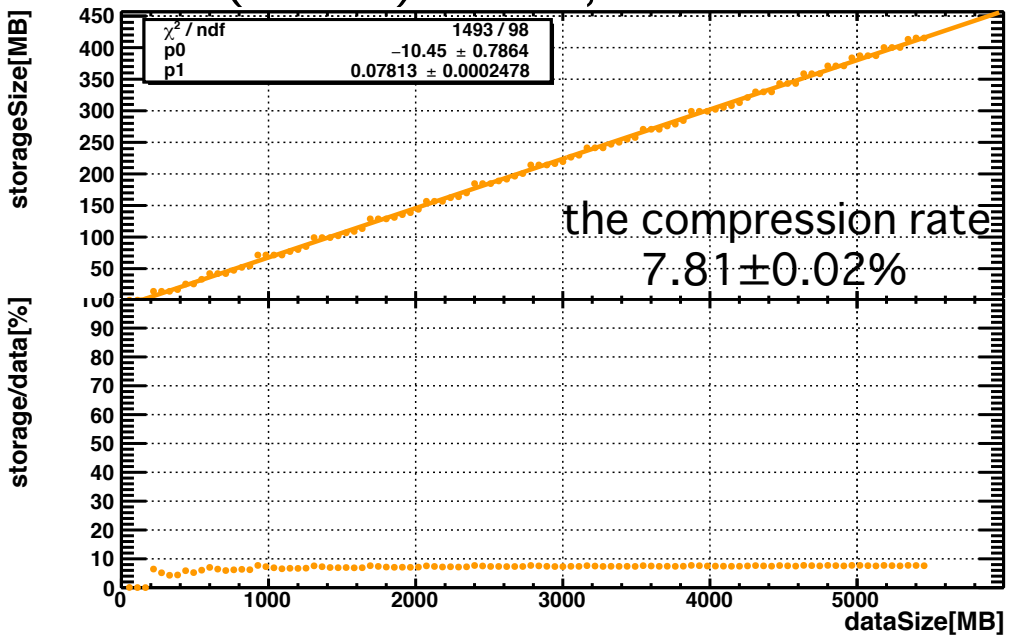
Data storing -Motivation and Methods-

- Compare the compression rate after storing data: png, pdf, dat and json
- Storage engine
  - WiredTiger
  - compression algorithm: snappy
  - interface: GridFS
- data
  - dat
    - EnMask.dat (53 KB)
    - ThresholdMap.dat (207 KB)
  - json
    - default_fei4b.json (3.7 MB)
    - default_rd53a.json (670 KB)
  - pdf
    - EnMask.pdf (9 KB)
    - ThresholdMap.pdf (61 KB)
  - png
    - EnMask.png (9 KB)
    - ThresholdMap.png (61 KB)

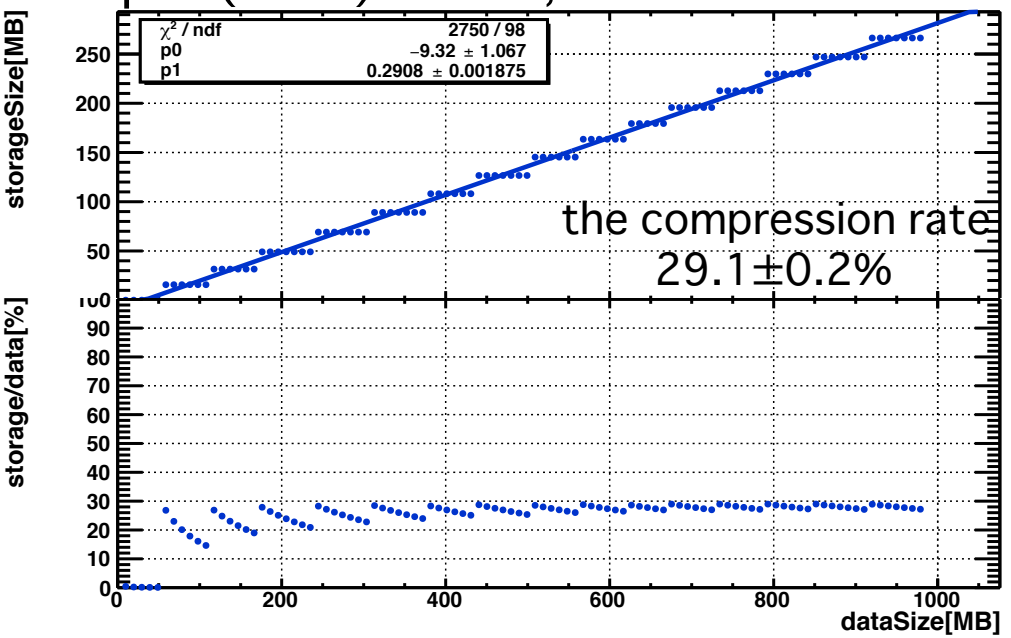


Data storing

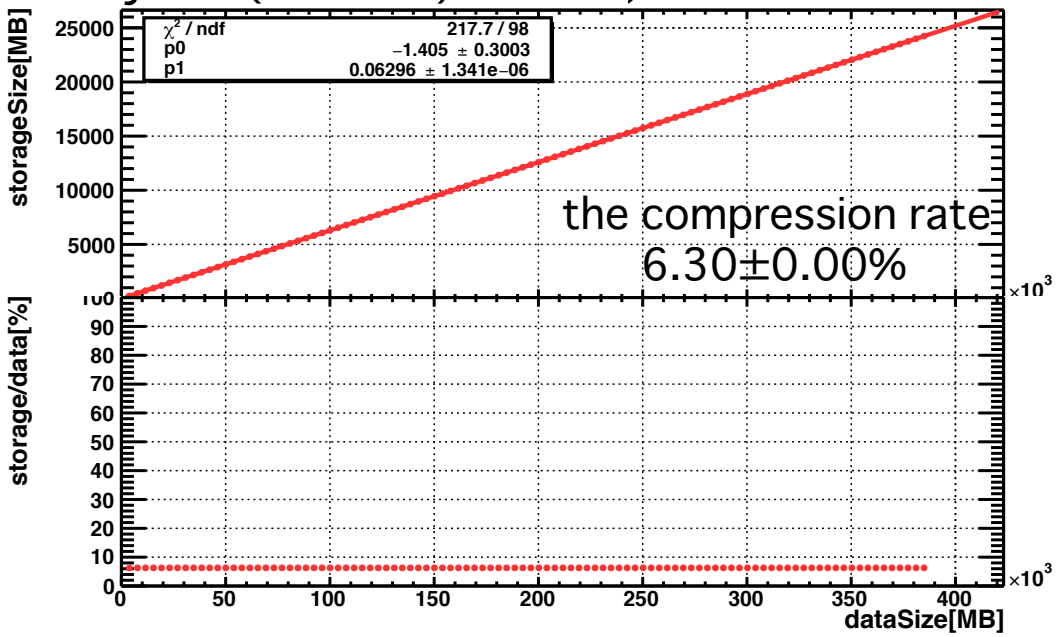
dat (53 KB) × 100,000 docs



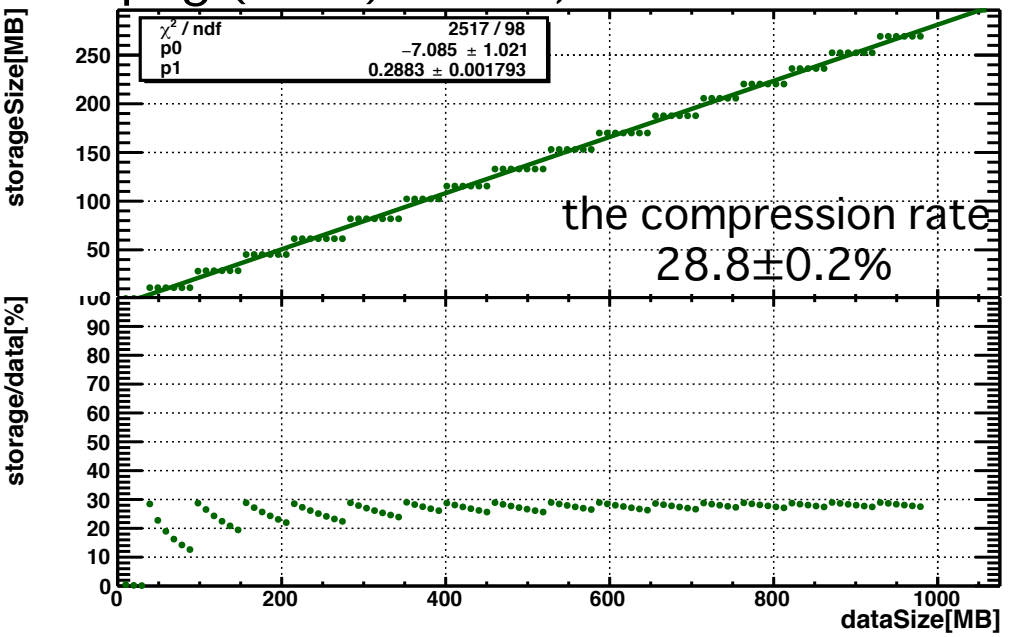
pdf (9 KB) × 100,000 docs



json (3.7 MB) × 100,000 docs

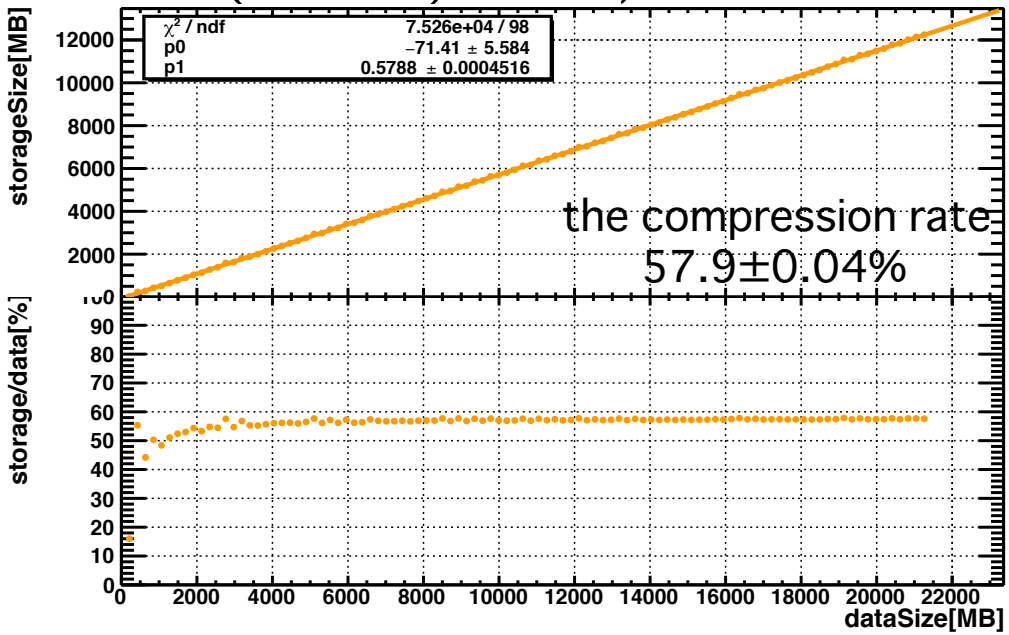


png (9 KB) × 100,000 docs

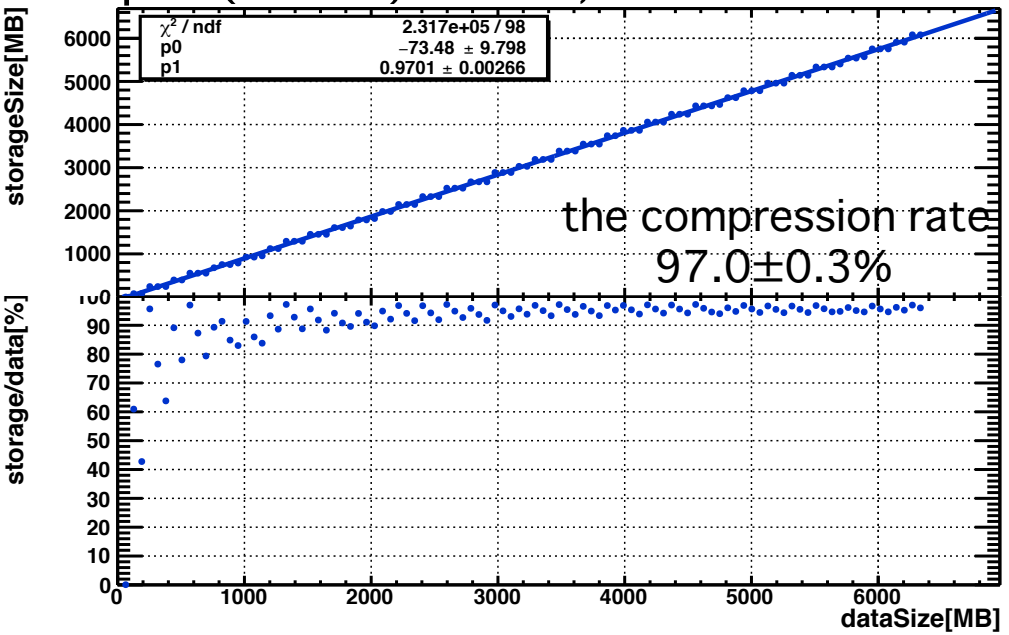


Data storing

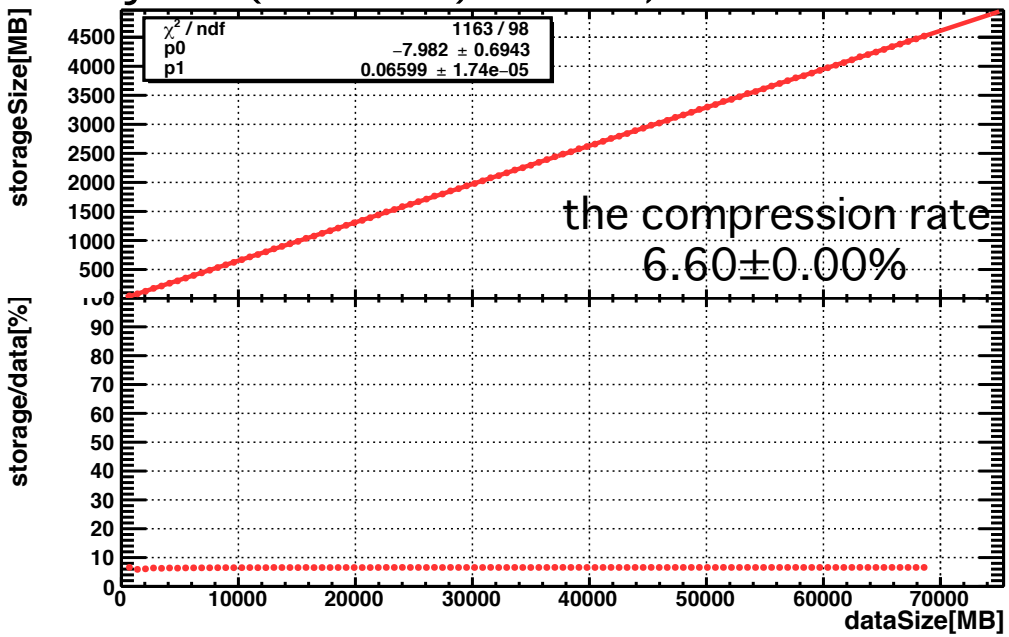
dat (207 KB) × 100,000 docs



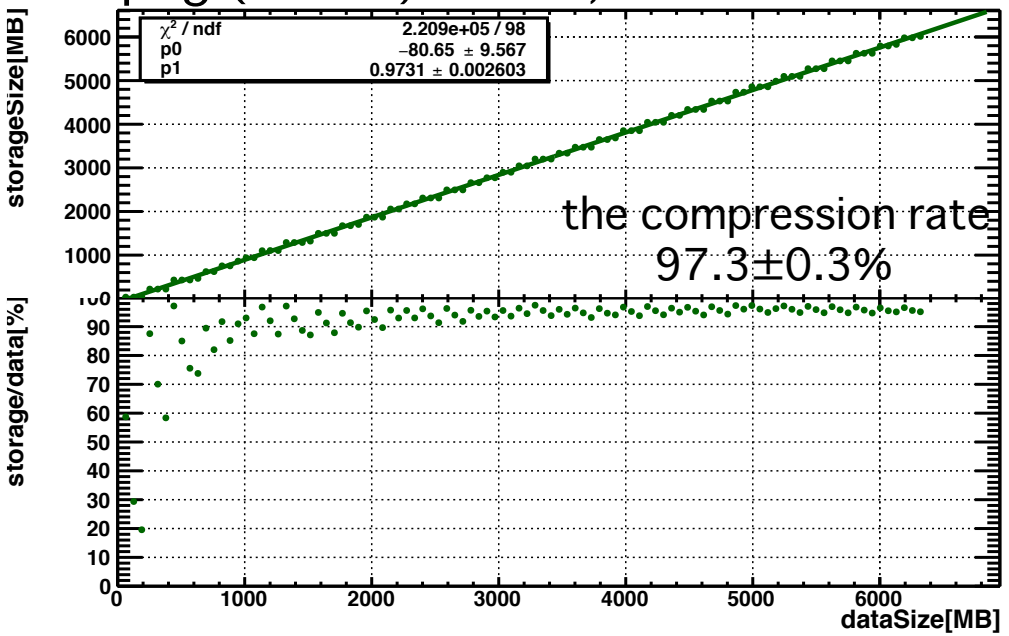
pdf (61 KB) × 100,000 docs



json (670 MB) × 100,000 docs

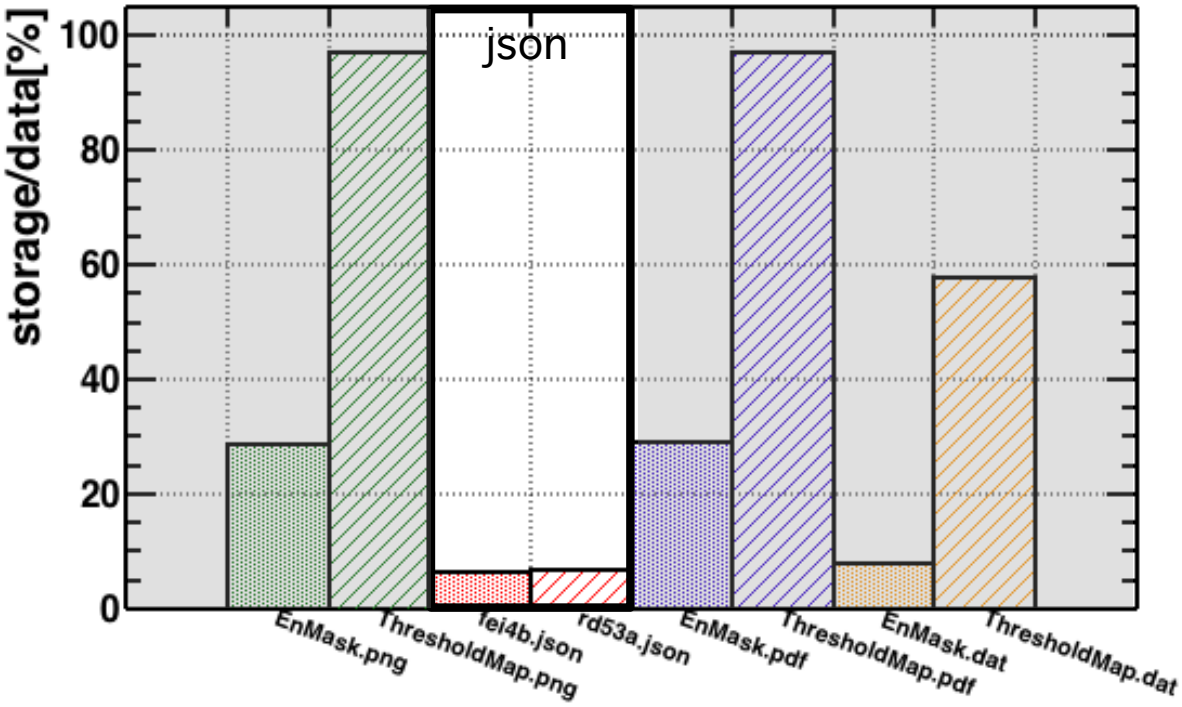


png (61 KB) × 100,000 docs



Data storing -Results-

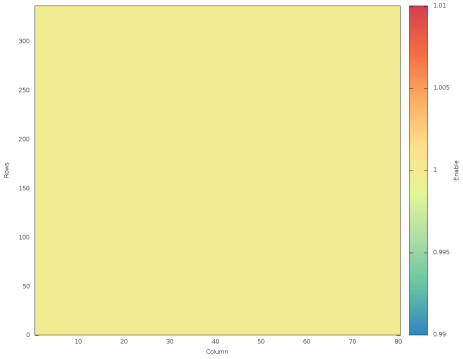
- Compare the compression rate after storing data: png, pdf, dat and json
- Storing json file
  - size: fei4b.json ⋯ 3,7MB, rd53a.json ⋯ 670KB
  - total storage size after storing 100,000 documents:
    fei4b.json ⋯ 24 GB, rd53a.json ⋯ 4.5 GB
  - high compression ratio: storage/data = 6%



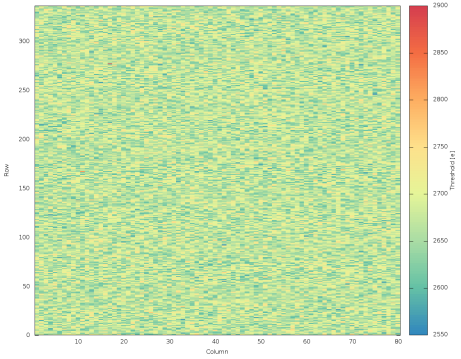
Data storing -Results-

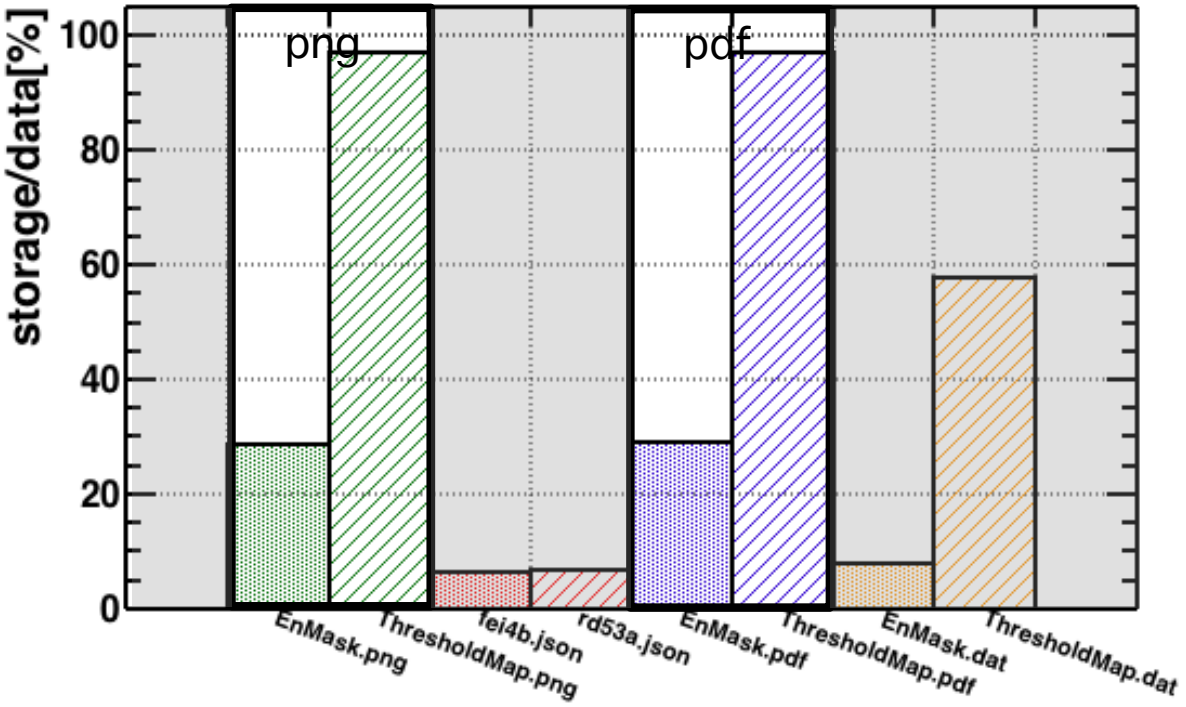
- Compare the compression rate after storing data: png, pdf, dat and json
- Storing png/pdf file
  - size: EnMask ⋯ 9 KB, ThresholdMap ... 61 KB
  - total storage size after storing 100,000 documents:
    EnMask ⋯ 270 MB, ThresholdMap ... 6.0 GB
  - **compression ratio: storage/data = 29%(EnMask), 97% (ThresholdMap)**

EnMask



ThresholdMap

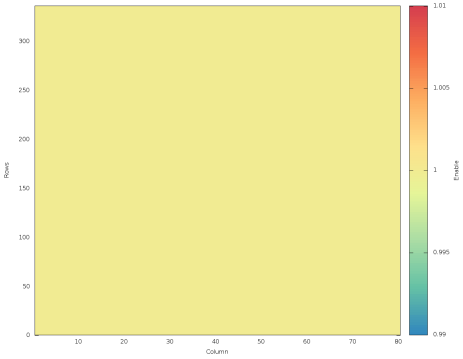




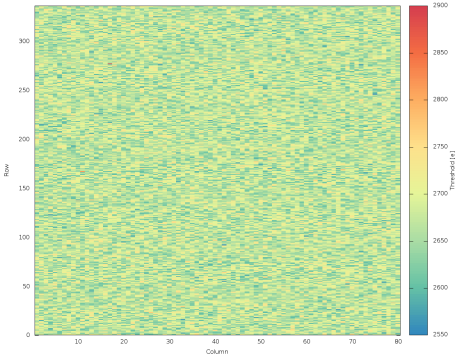
Data storing -Results-

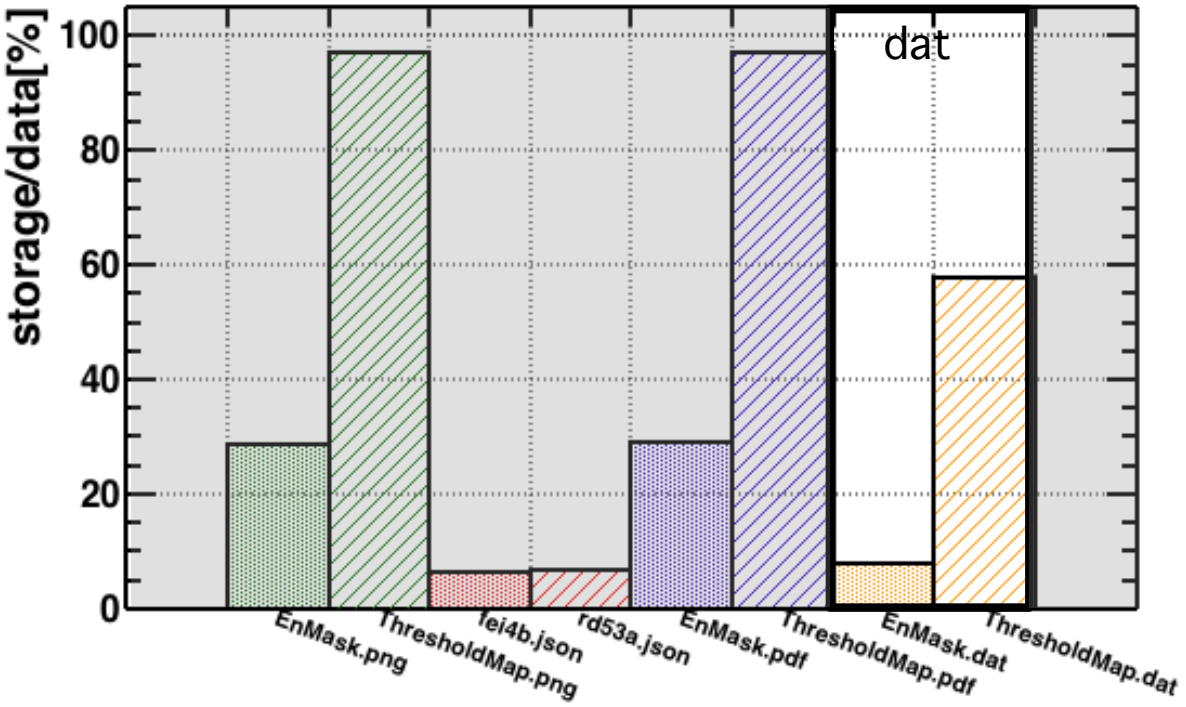
- Compare the compression rate after storing data: png, pdf, dat and json
- Storing png/pdf file
  - size: EnMask ⋯ 53 KB, ThresholdMap … 207 KB
  - total storage size after storing 100,000 documents:
    EnMask ⋯ 415 MB, ThresholdMap … 12.2 GB
  - **compression ratio: storage/data = 8%(EnMask), 58% (ThresholdMap)**

EnMask



ThresholdMap

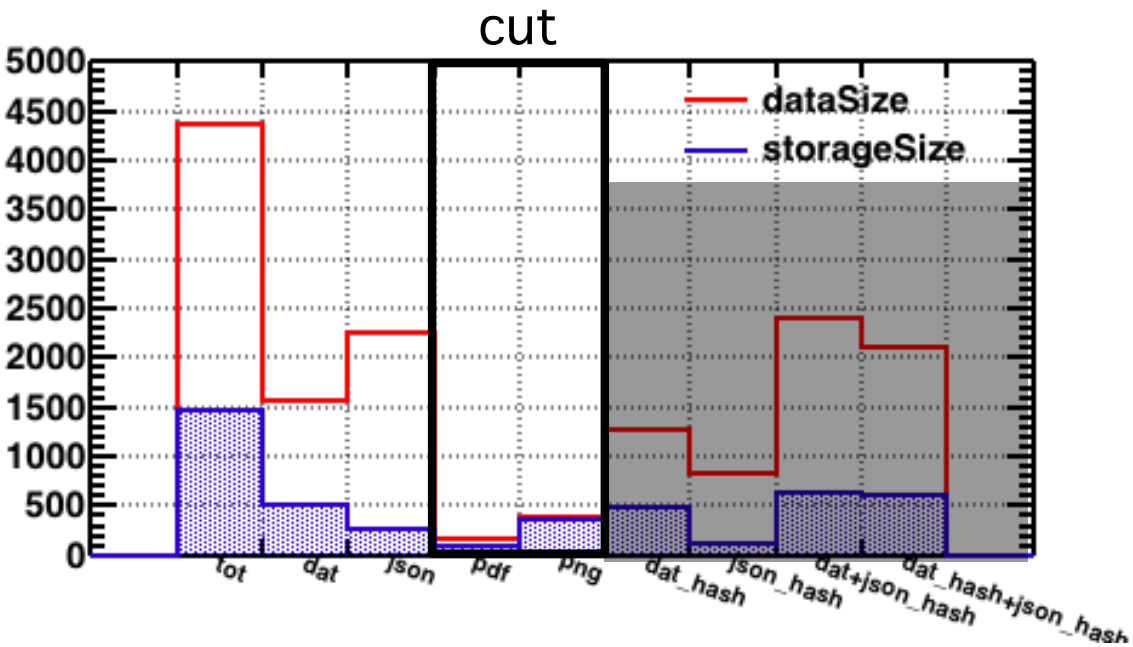




Data storing -Conclusion-

- Compare the compression rate after storing data: png, pdf, dat and json
- Storing json (and dat) file is high performance in compression
- There might be png/pdf file stored without compression
  → <u>They can be created from dat file so there is no need to store them</u>

- i.e.) Dataset of MasterDB in KEK



the data size of docs in MasterDB (KEK)
- ∗ dat 1574 MB (/tot : 36%)
- ∗ json 2251 MB (/tot : 52%)
- ∗ pdf 159 MB (/tot : 3.6%)
- ∗ png 380 MB (/tot : 8.7%)

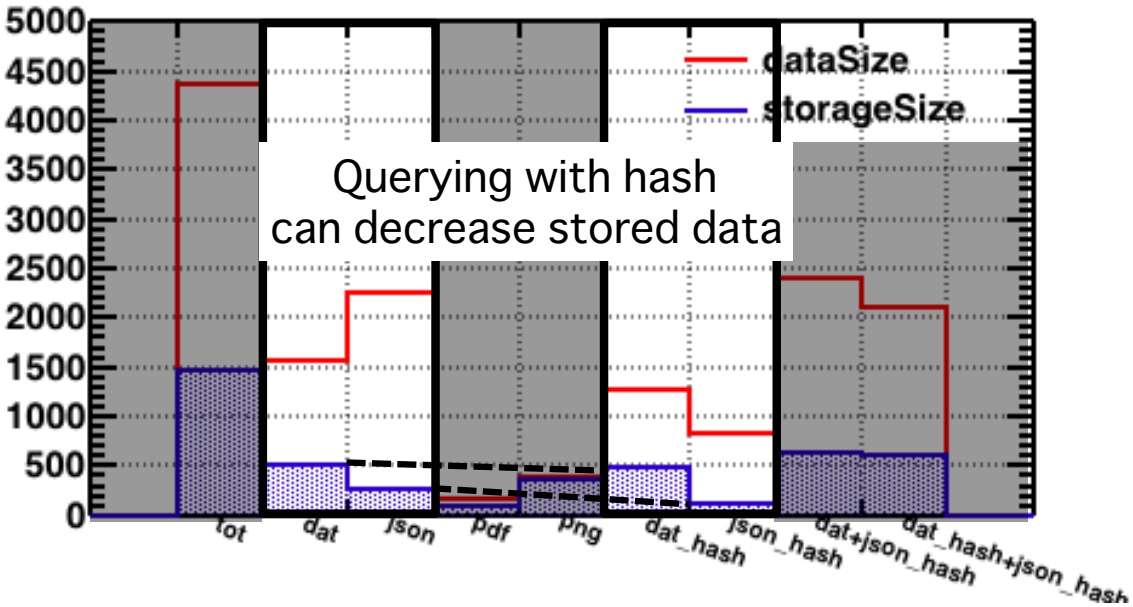
the storage size of docs (KEK)
- ∗ dat 507 MB (/tot : 35%)
- ∗ json 271 MB (/tot : 18%)
- **∗ pdf 93 MB (/tot : 6.3%)**
- **∗ png 352 MB (/tot : 24%)**

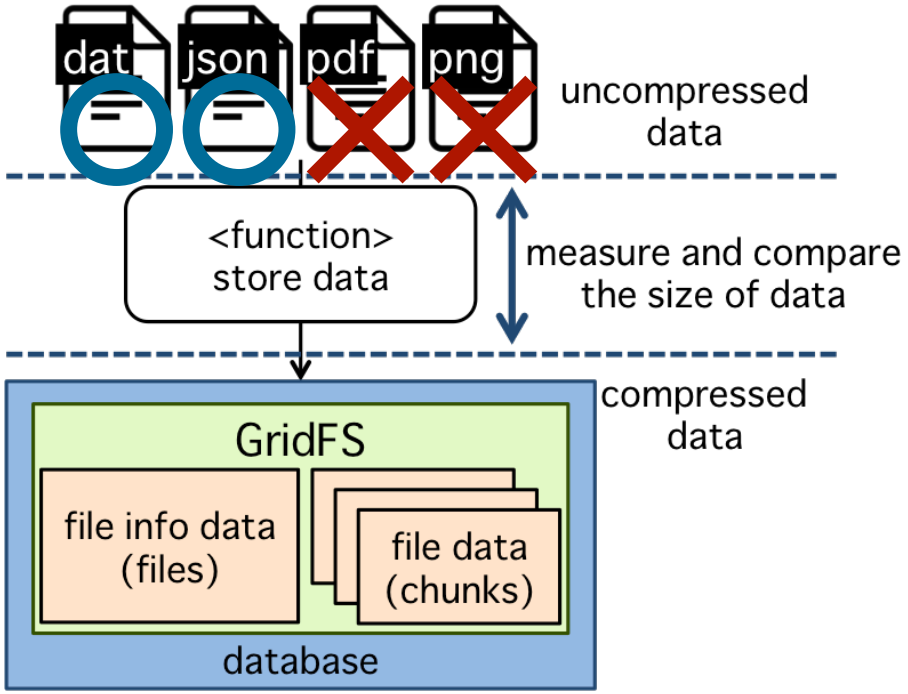
about 30% cut

Data storing -Conclusion-

- Compare the compression rate after storing data: png, pdf, dat and json
- Storing json (and dat) file is high performance in compression
- There might be png/pdf file stored without compression
  → <u>They can be created from dat file so there is no need to store them</u>

- i.e.) Dataset of MasterDB in KEK

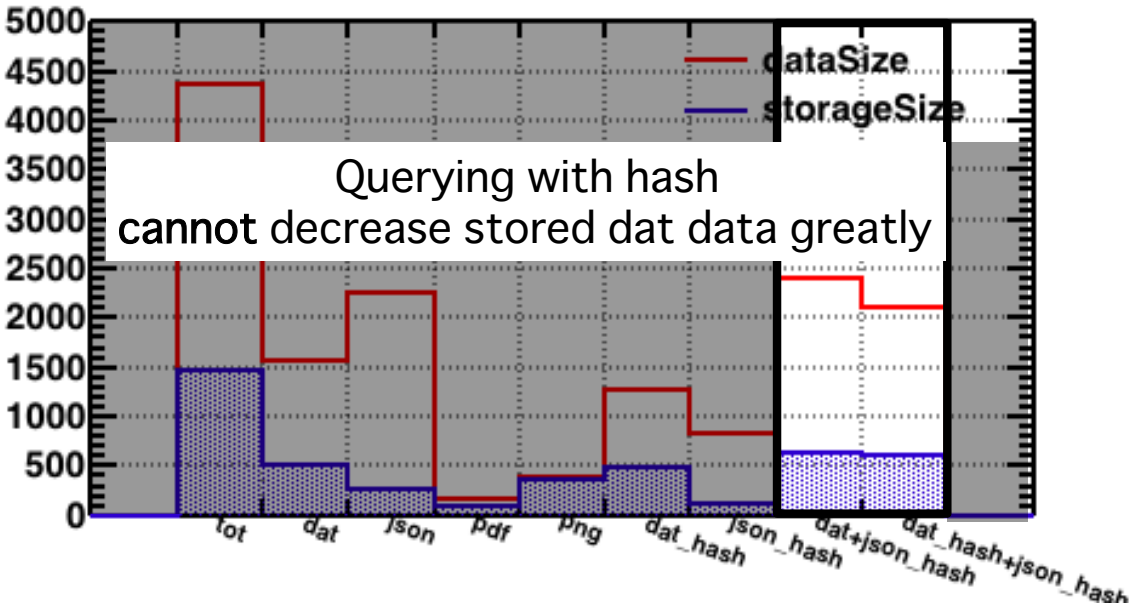




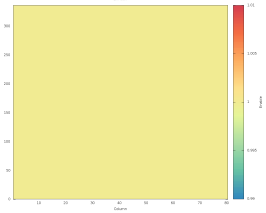
Data storing -Conclusion-

- Compare the compression rate after storing data: png, pdf, dat and json
- Storing json (and dat) file is high performance in compression
- There might be png/pdf file stored without compression
  → <u>They can be created from dat file so there is no need to store them</u>

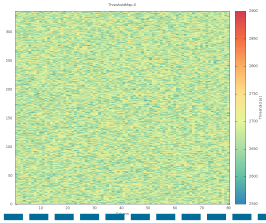
- i.e.) Dataset of MasterDB in KEK



the number of docs in MasterDB (KEK)
<u>80881 → 65084 docs (20% cut)</u>
the storage size of docs (KEK)
<u>625 MB → 605 MB (3% cut)</u>



good/bad EnMask.dat can be cut
→ high compression rate
→ not decrease storage size greatly

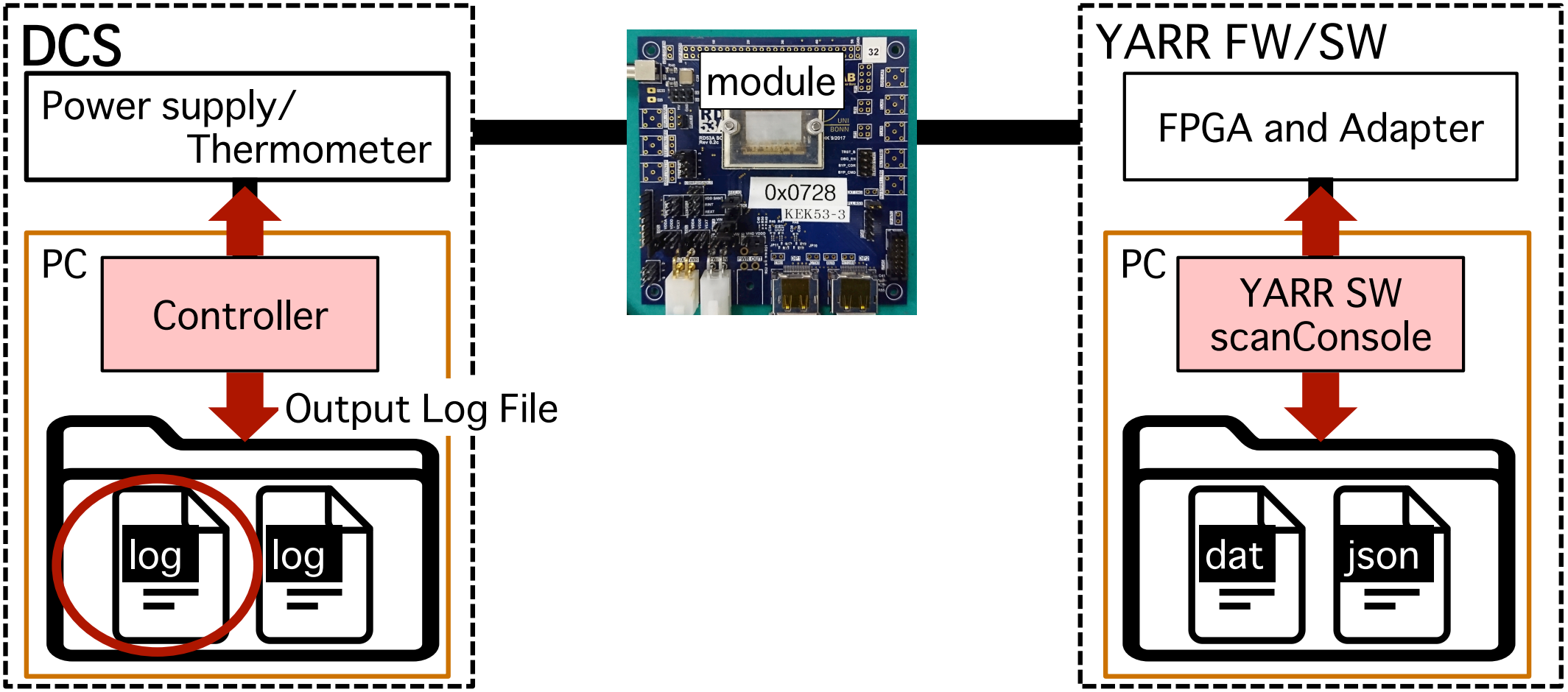


ThresholdMap.dat cannot be cut
→ low compression rate
→ not decrease storage size

DCS Storing

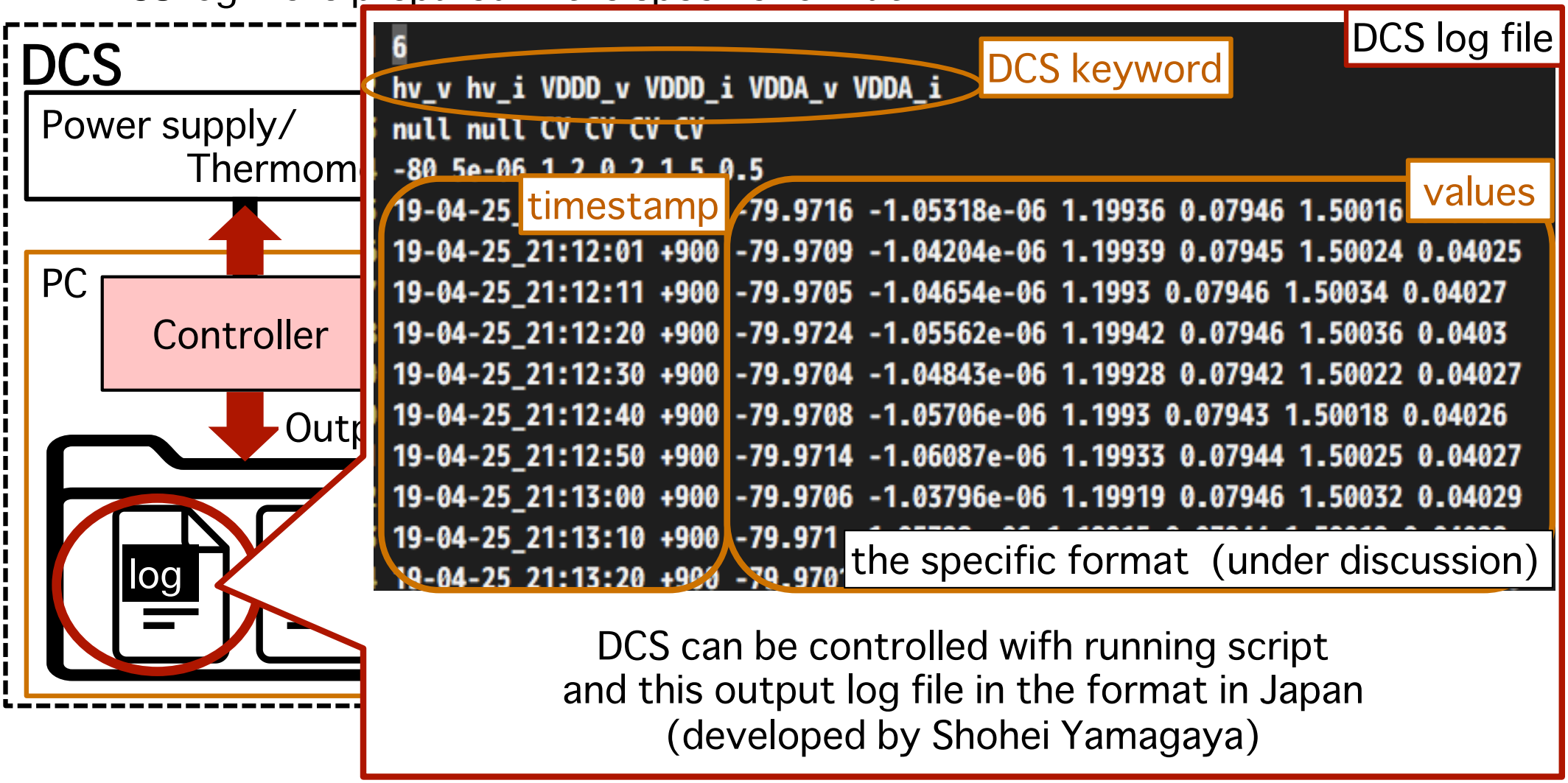
DCS Storing (1/7)

- Enable to store DCS information while scanning into Database automatically if DCS log file is prepared in the specific format



DCS Storing (2/7)

- Enable to store DCS information while scanning into Database automatically if DCS log file is prepared in the specific format



DCS Storing (3/7)

- Enable to store DCS information while scanning into Database automatically if DCS log file is prepared in the specific format

DCS log file

```
6
hv_v hv_i VDDD_v VDDD_i VDDA_v VDDA_i
null null CV CV CV CV
-80 5e-06 1.2 0.2 1.5 0.5
19-04-25_21:11:51 +900 -79.9716 -1.05318e-06 1.19936 0.07946 1.50016 0.04028
19-04-25_21:12:01 +900 -79.9709 -1.04204e-06 1.19939 0.07945 1.50024 0.04025
19-04-25_21:12:11 +900 -79.9705 -1.04654e-06 1.1993 0.07946 1.50034 0.04027
19-04-25_21:12:20 +900 -79.9724 -1.05562e-06 1.19942 0.07946 1.50036 0.0403
19-04-25_21:12:30 +900 -79.9704 -1.04843e-06 1.19928 0.07942 1.50022 0.04027
19-04-25_21:12:40 +900 -79.9708 -1.05706e-06 1.1993 0.07943 1.50018 0.04026
19-04-25_21:12:50 +900 -79.9714 -1.06087e-06 1.19933 0.07944 1.50025 0.04027
19-04-25_21:13:00 +900 -79.9706 -1.03796e-06 1.19919 0.07946 1.50032 0.04029
19-04-25_21:13:10 +900 -79.971 -1.05728e-06 1.19915 0.07944 1.50019 0.04028
19-04-25_21:13:20 +900 -79.9701 -1.03978e-06 1.19918 0.07946 1.50026 0.04029
```

<YARR SW>

scanConsole -I DCSCfg.json

DCSCfg.json

```
{
    "assembly": {
        "stage": "Testing"
    },
    "environments": [
        {
            "key": "lv",
            "value": 1.8,
            "description": "Low voltage [V]",
            "path": "/tmp/lv.dat"
        }
    ]
}
```

DCS keyword

path to log file

DCS Storing (4/7)

- Enable to store DCS information while scanning into Database automatically if DCS log file is prepared in the specific format

```
6
hv_v hv_i VDDD_v VDDD_i VDDA_v VDDA_i
null null CV CV CV CV
-80 5e-06 1.2 0.2 1.5 0.5
19-04-25_21:11:51 +900 -79.9716 -1.05318e-06 1.19936 0.07946 1.50016 0.04028
19-04-25_21:12:01 +900 -79.9709 -1.04204e-06 1.19939 0.07945 1.50024 0.04025
19-04-25_21:12:11 +900 -79.9705 -1.04654e-06 1.1993 0.07946 1.50034 0.04027
19-04-25_21:12:20 +900 -79.9724 -1.05562e-06 1.19942 0.07946 1.50036 0.0403
19-04-25_21:12:30 +900 -79.9704 -1.04843e-06 1.19928 0.07942 1.50022 0.04027
19-04-25_21:12:40 +900 -79.9708 -1.05706e-06 1.1993 0.07943 1.50018 0.04026
19-04-25_21:12:50 +900 -79.9714 -1.06087e-06 1.19933 0.07944 1.50025 0.04027
19-04-25_21:13:00 +900 -79.9706 -1.03796e-06 1.19919 0.07946 1.50032 0.04029
19-04-25_21:13:10 +900 -79.971 -1.05728e-06 1.19915 0.07944 1.50019 0.04028
19-04-25_21:13:20 +900 -79.9701 -1.03978e-06 1.19918 0.07946 1.50026 0.04029
```

<YARR SW>

scanConsole -I DCSCfg.json

upload test data into DB
with generating cache
to upload DCS info

testRun : Data ID
DCS : path to log file

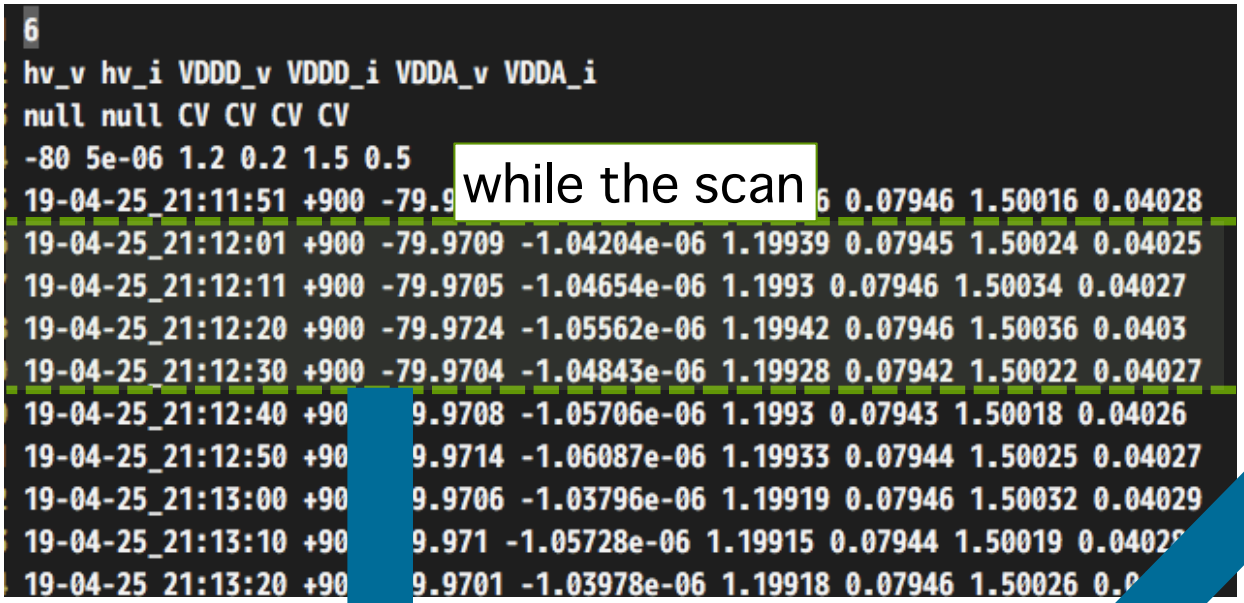
Local DB

test run data

start time : …
finish time : …

DCS Storing (5/7)

- Enable to store DCS information while scanning into Database automatically if DCS log file is prepared in the specific format



<YARR SW>

scanConsole -I DCSCfg.json

upload test data into DB with generating cache to upload DCS info

testRun : Data ID
DCS : path to log file

DCS data while scan is uploaded following cache by specific tool

Local DB

dcs data

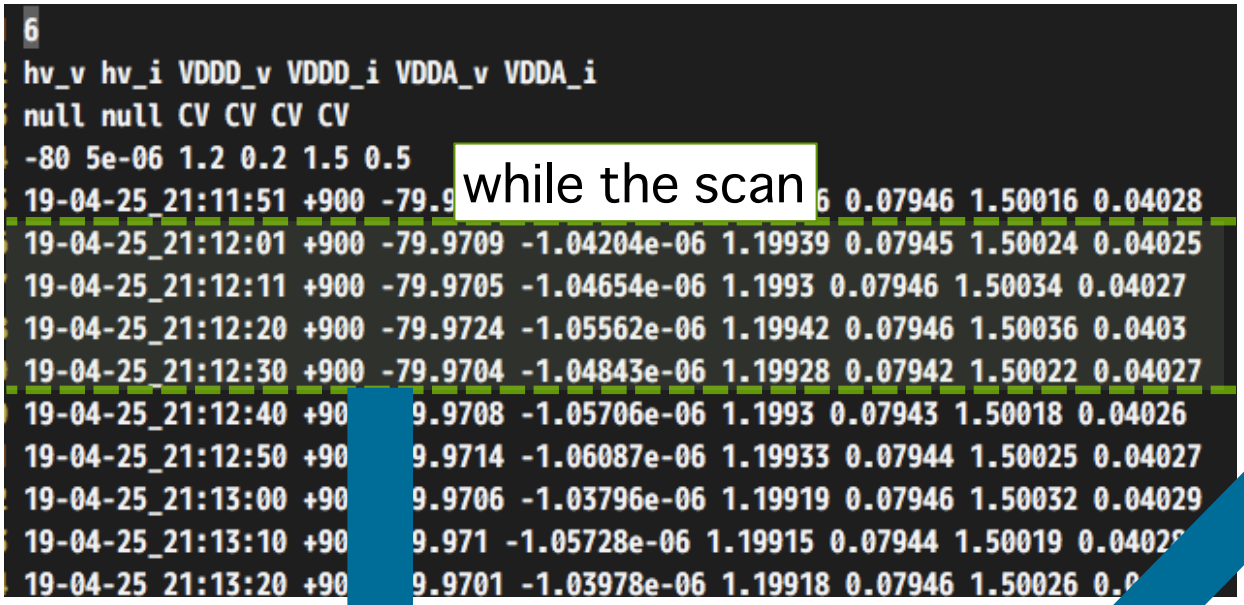
data_id: XXX
lv : [{timestamp, value},
{timestamp, value}]

test run data

start time : …
finish time : …
env_id: XXX

DCS Storing (6/7)

- Enable to store DCS information while scanning into Database automatically if DCS log file is prepared in the specific format



<YARR SW>

scanConsole -I DCSCfg.json

upload test data into DB with generating cache to upload DCS info

testRun : Data ID
DCS : path to log file

DCS data while scan is uploaded following cache by specific tool

Local DB

dcs data

data_id: XXX
lv : [{timestamp, value},
{timestamp, value}]

linked by id

test run data

start time : …
finish time : …
env_id: XXX

DCS Storing (7/7)

- Enable to check stored DCS data in browser (developed by S. Yamagaya)

Local DB

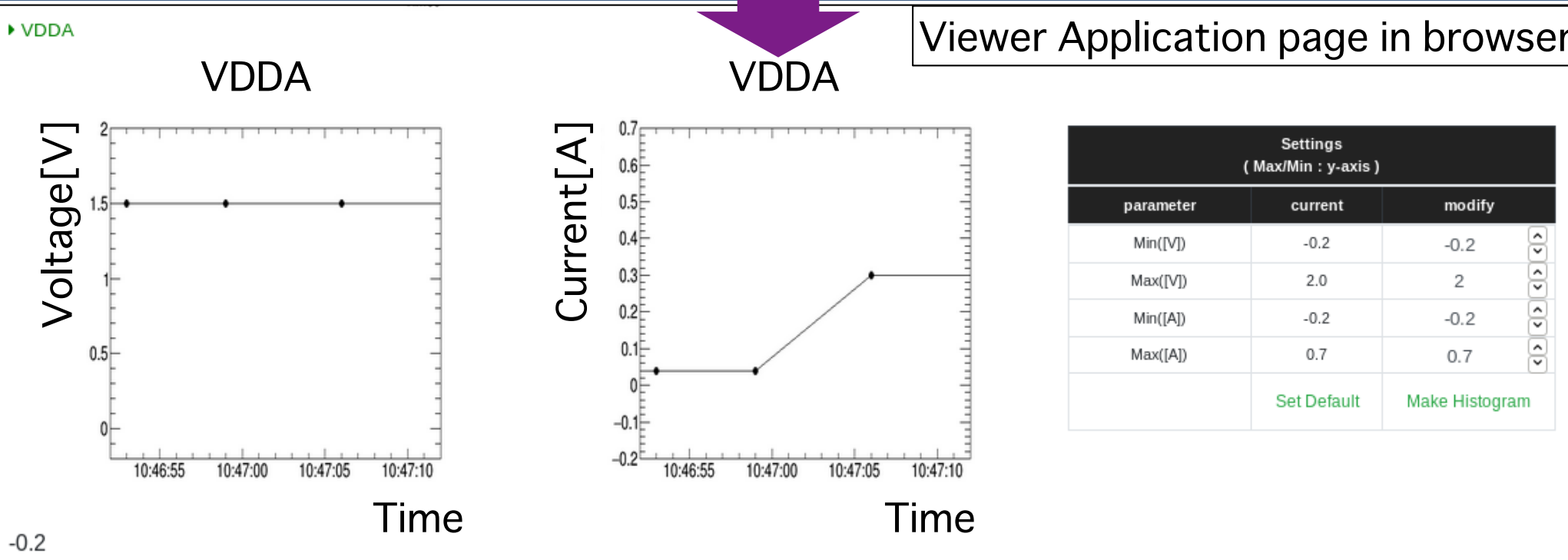
dcs data

data_id: XXX
lv : [{timestamp, value},
{timestamp, value}]

test run data

start time : …
finish time : …
env_id: XXX

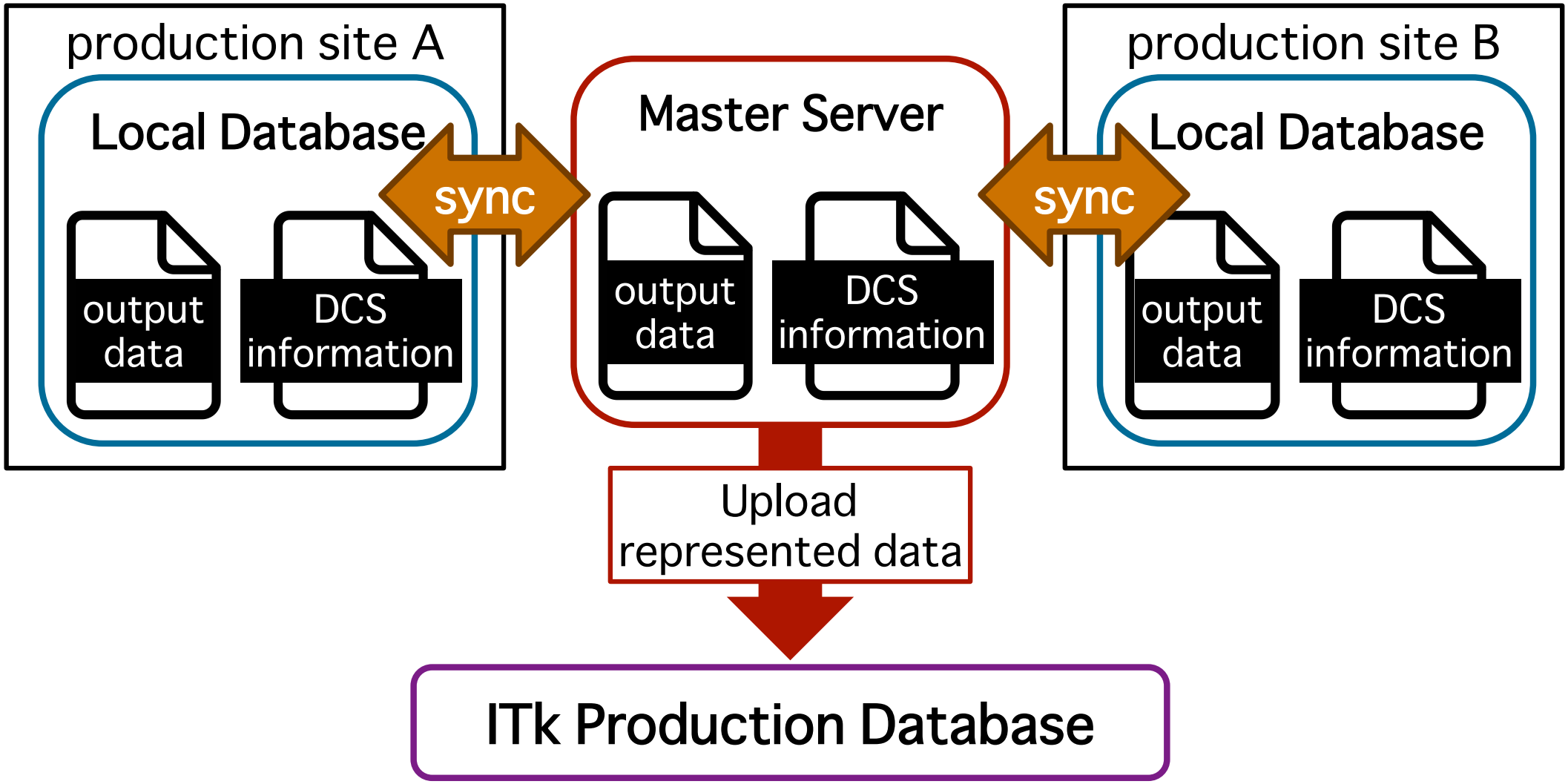
Viewer Application page in browser



Synchronization Tool

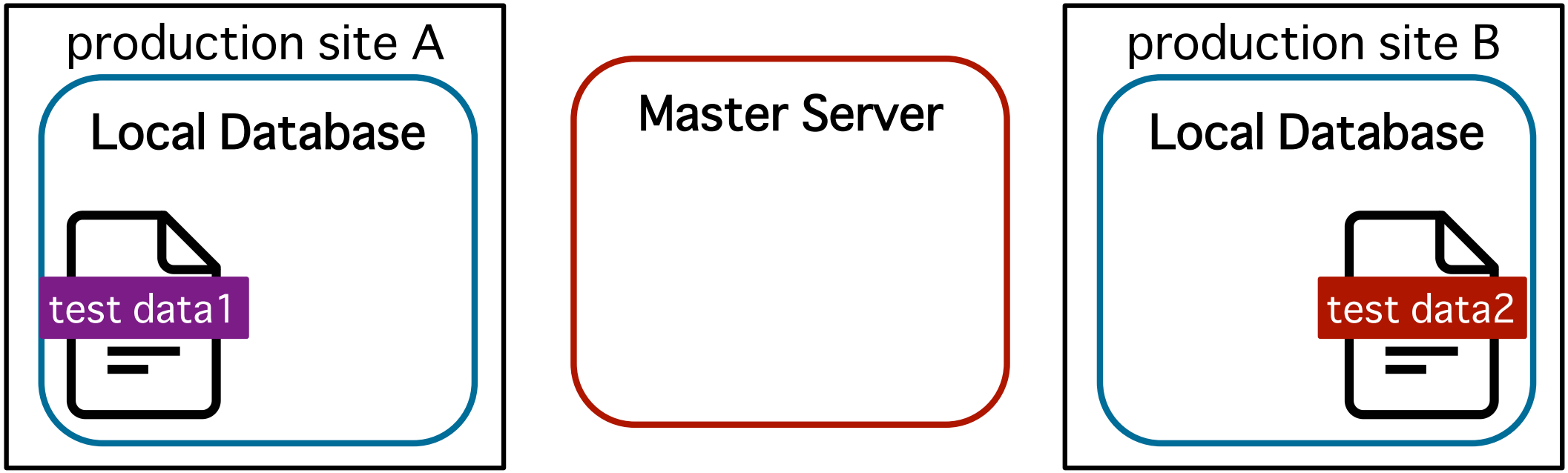
Synchronization Tool (developed by Eunchong Kim)

- Master Server has whole data by synchronizing from each production site, and upload represented data into ITk PD



Synchronization Tool (2/5)

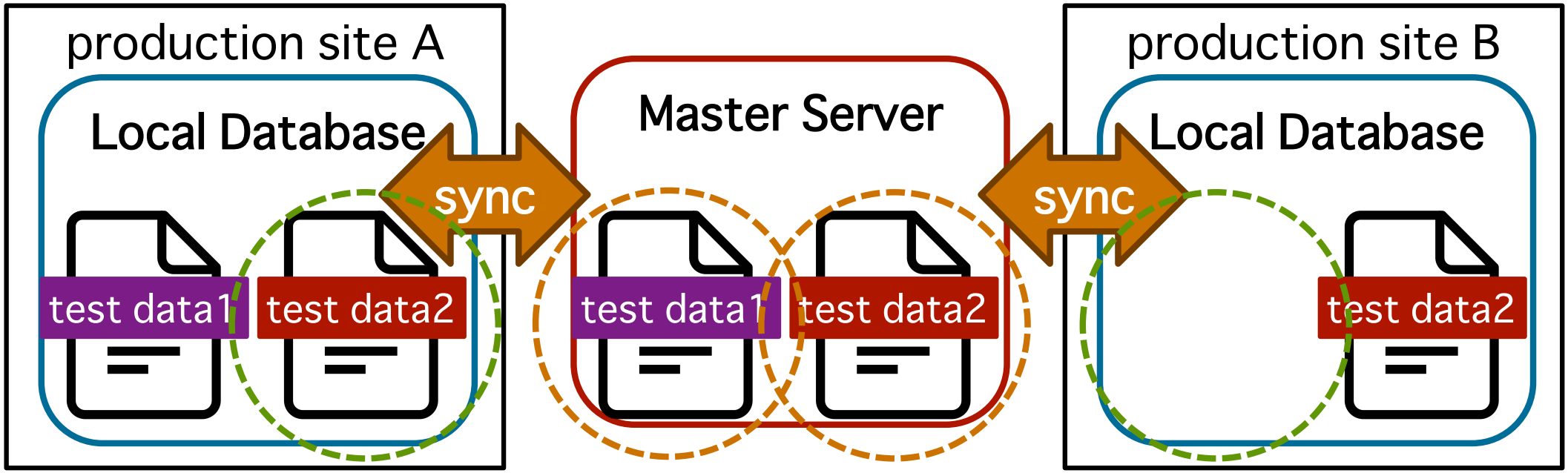
- Master Server has whole data by synchronizing from each production site, and upload represented data into ITk PD



- Synchronization Tool transfer only the difference data between Local DB and Master Server like `**rsync**`
- Master Serevr should have whole data set, but Local DB is not, so Local DB can download data partially from Master Server if required

Synchronization Tool (3/5)

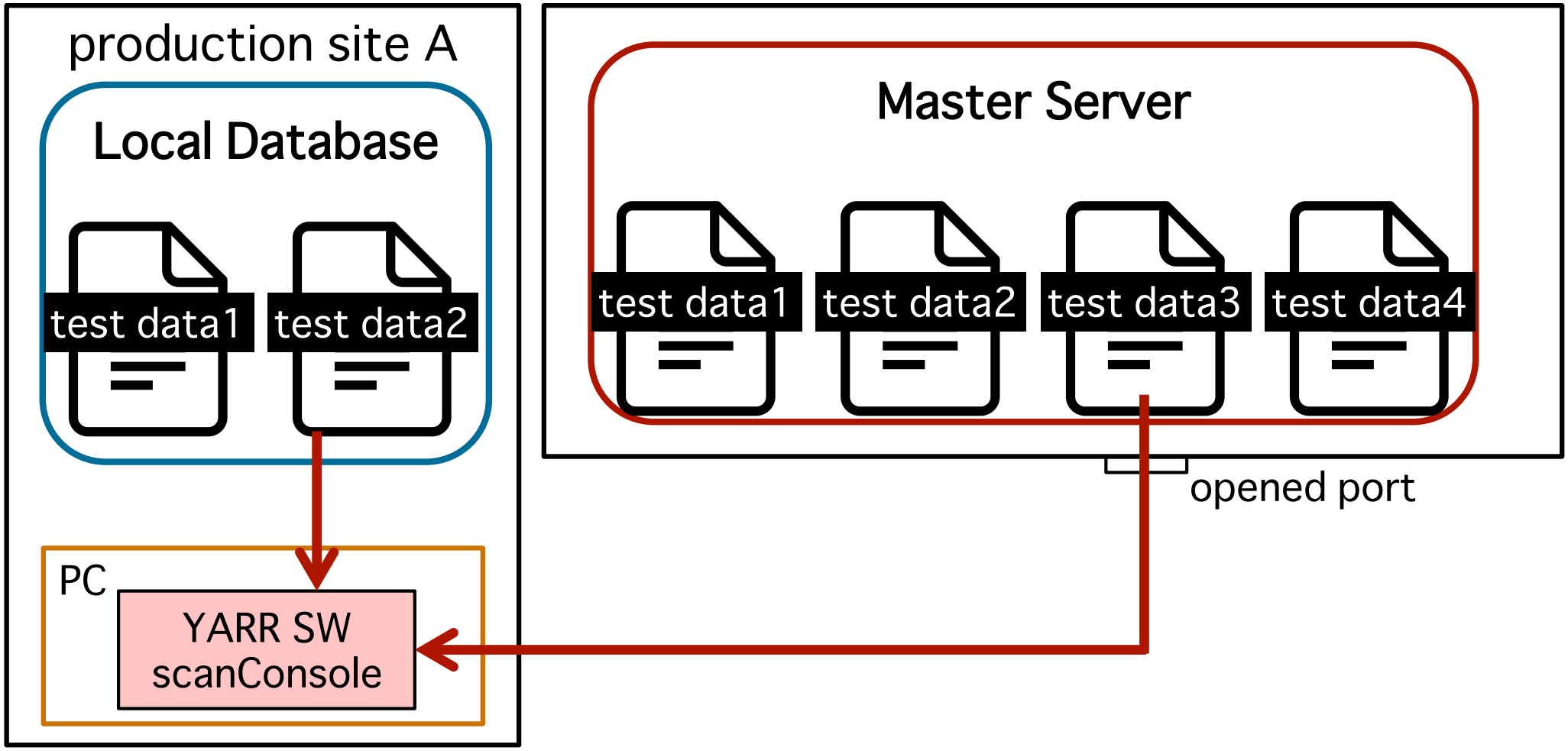
- Master Server has whole data by synchronizing from each production site, and upload represented data into ITk PD



- Synchronization Tool transfer only the difference data between Local DB and Master Server like `**rsync**`
- Master Server should have whole data set, but Local DB is not, so Local DB can download data partially from Master Server if required

Synchronization Tool (4/5)

- Master Server has the same data format as Local DB, so Local DB controller (such as function included in scanConsole) can access Master Server directly to retrieve data stored in other production site

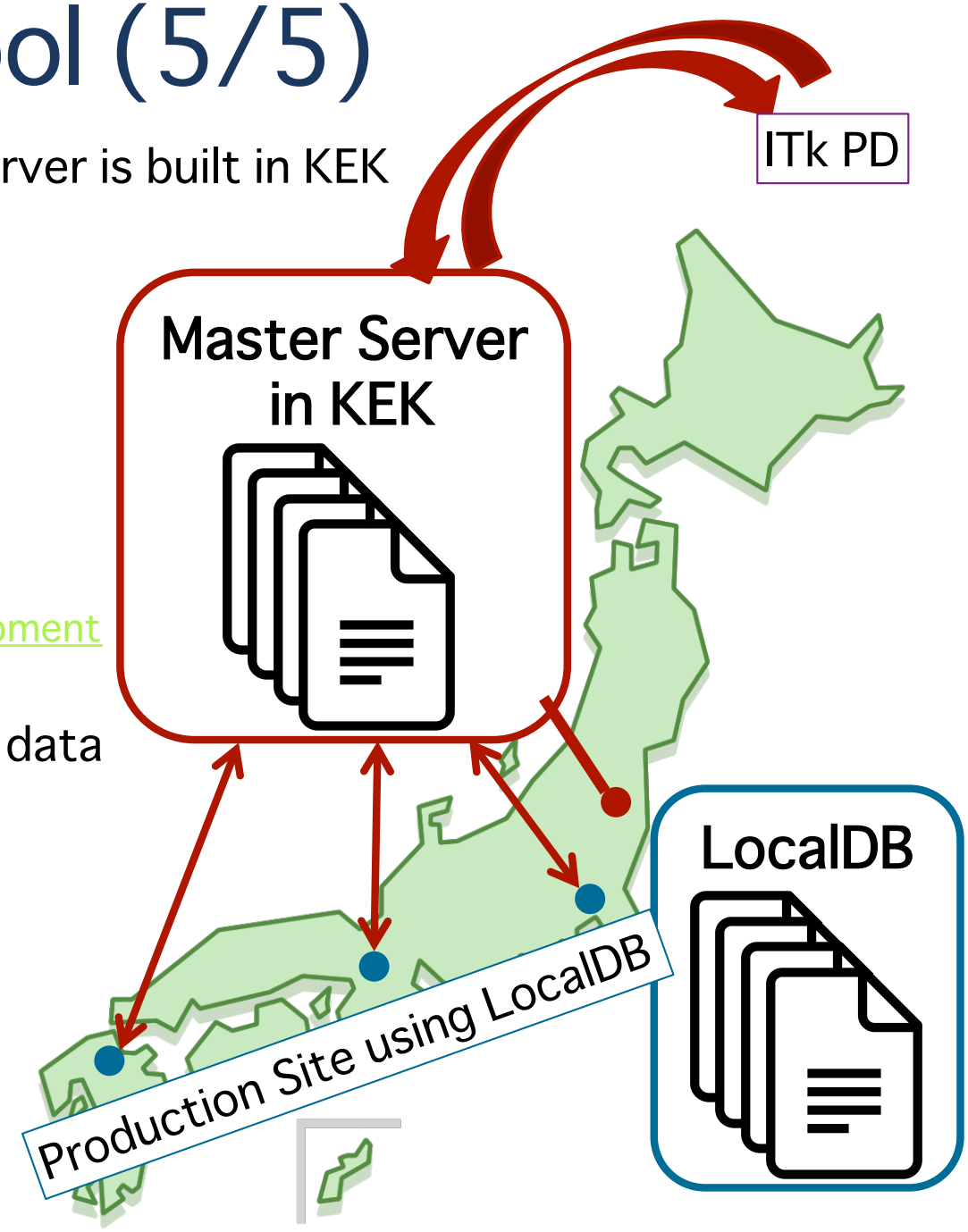


Synchronization Tool (5/5)

- For demonstration in Japan, Master Server is built in KEK

- Production site using LocalDB can upload and retrieve test data to Master Server in KEK with a relatively fast network connection
- Also production site can check whole data in Master Server viewer) http://atlaspc5.kek.jp/yarrdb/development

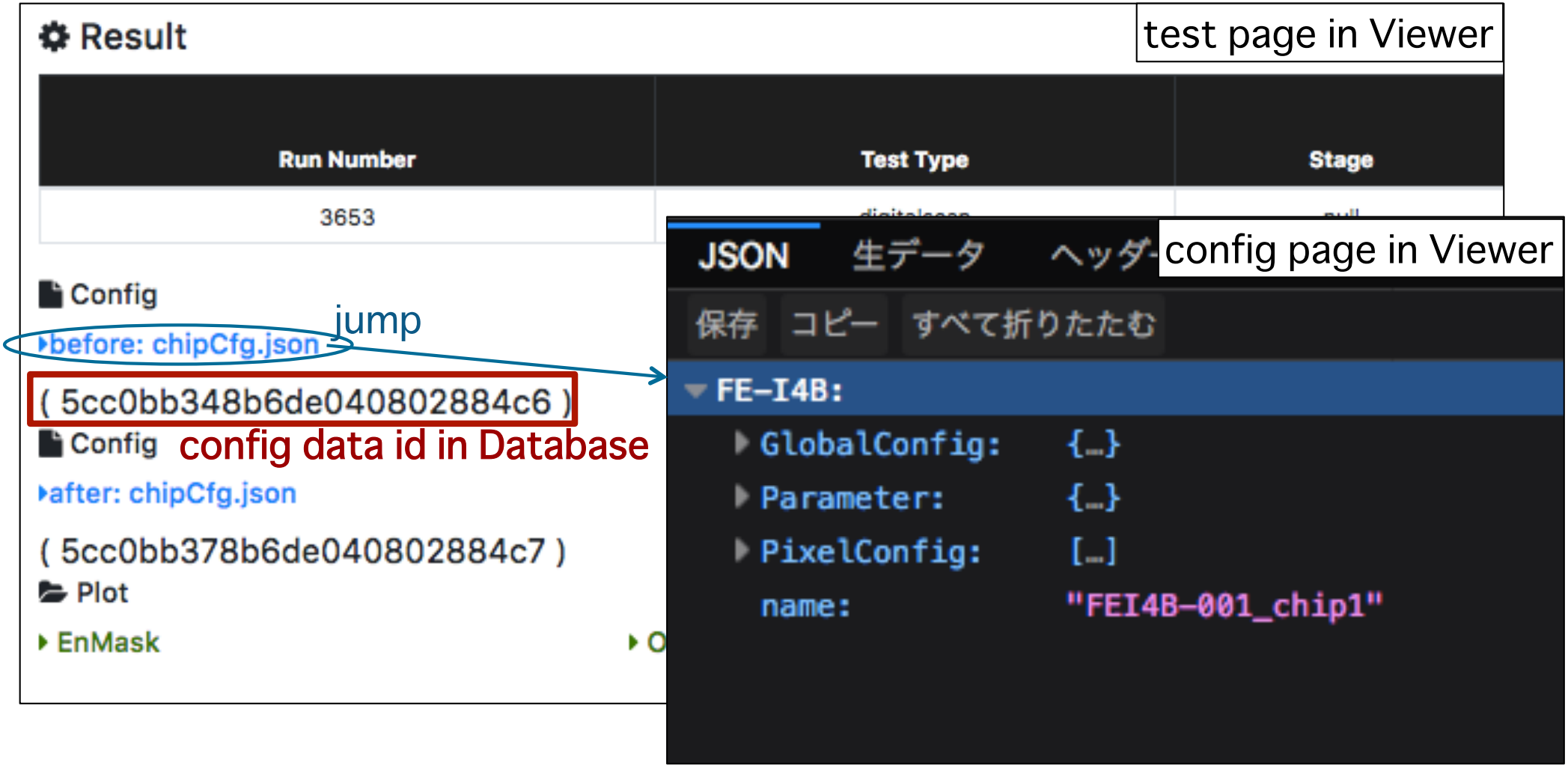
- Master Server can upload represented data into ITk PD



Config Retriever

Config Retriever (1/5)

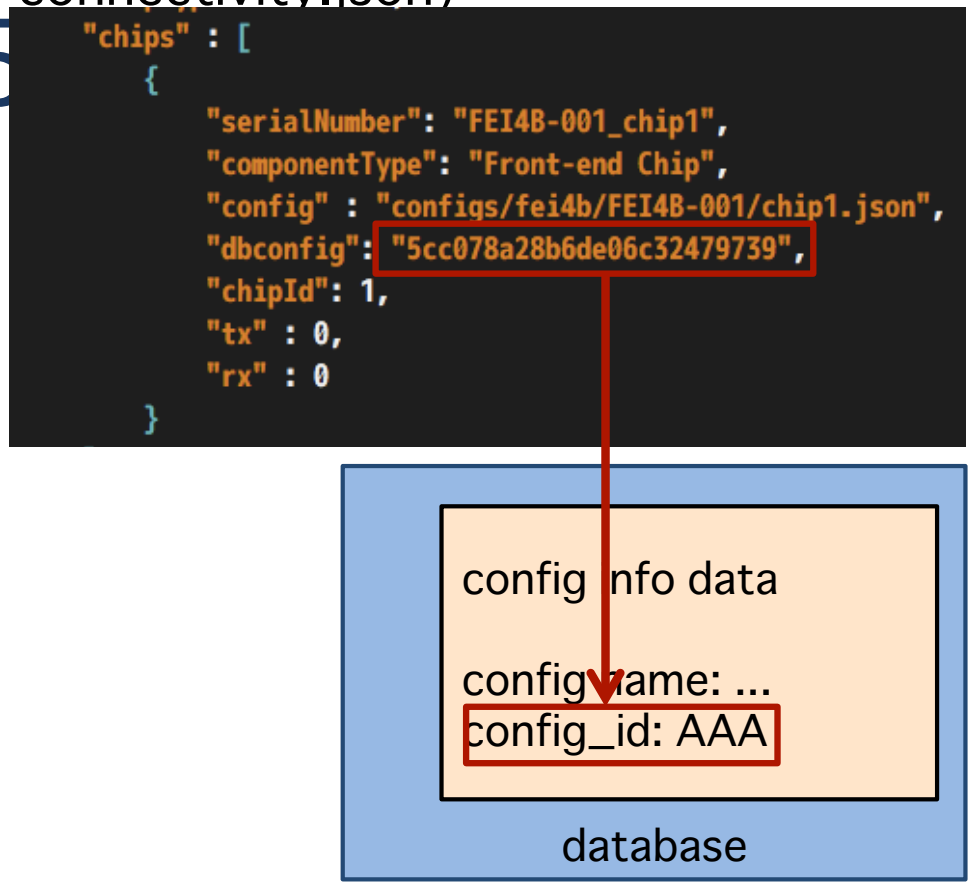
- The chip config can be displayed in Viewer (in development)



Config Retriever (2/5

connectivity.json)

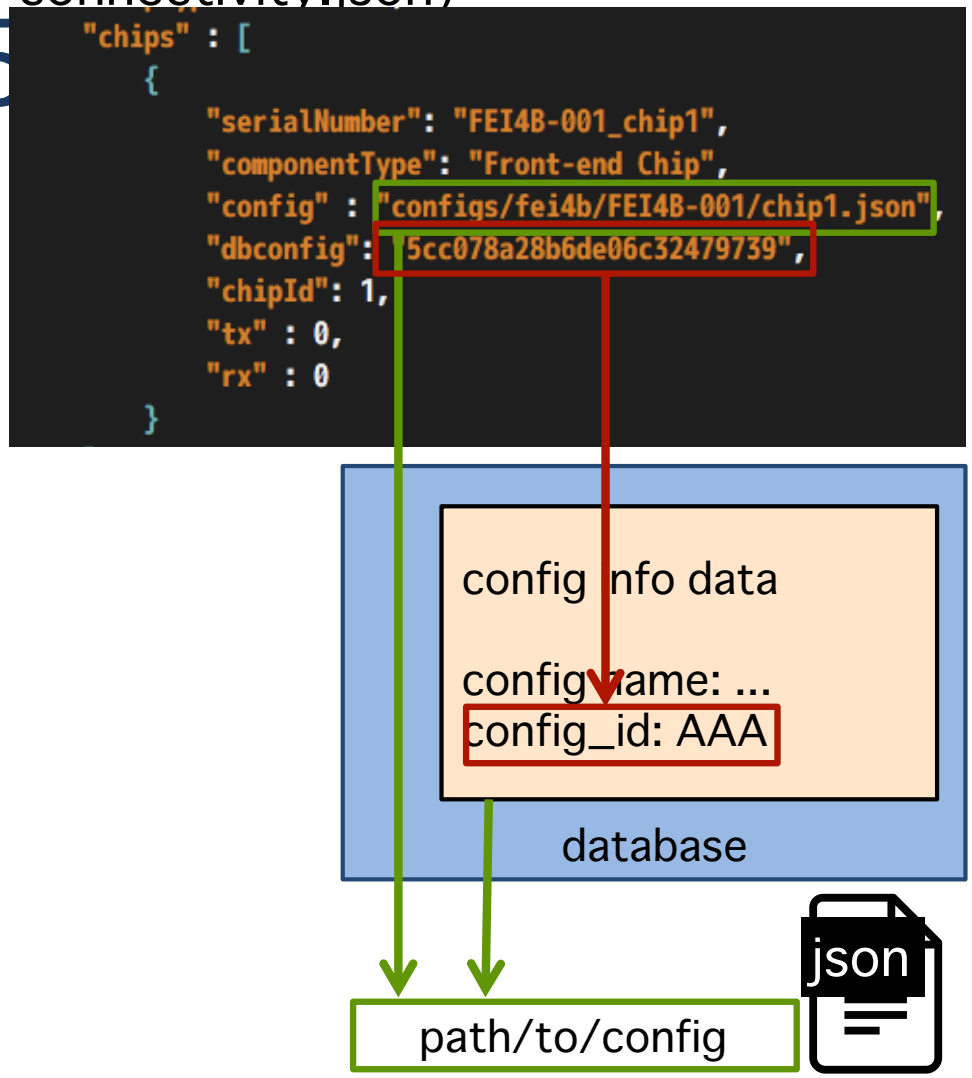
- scanConsole
  - ⓪ Give config id in Viewer (dbconfig)
  - ① download config file in DB by config id



Config Retriever (3/5

- scanConsole
  - ⓪ Give config id in Viewer (dbconfig)
  - ① download config file in DB by config id
  - ② **Create config file from DB**
    - → path/to/config

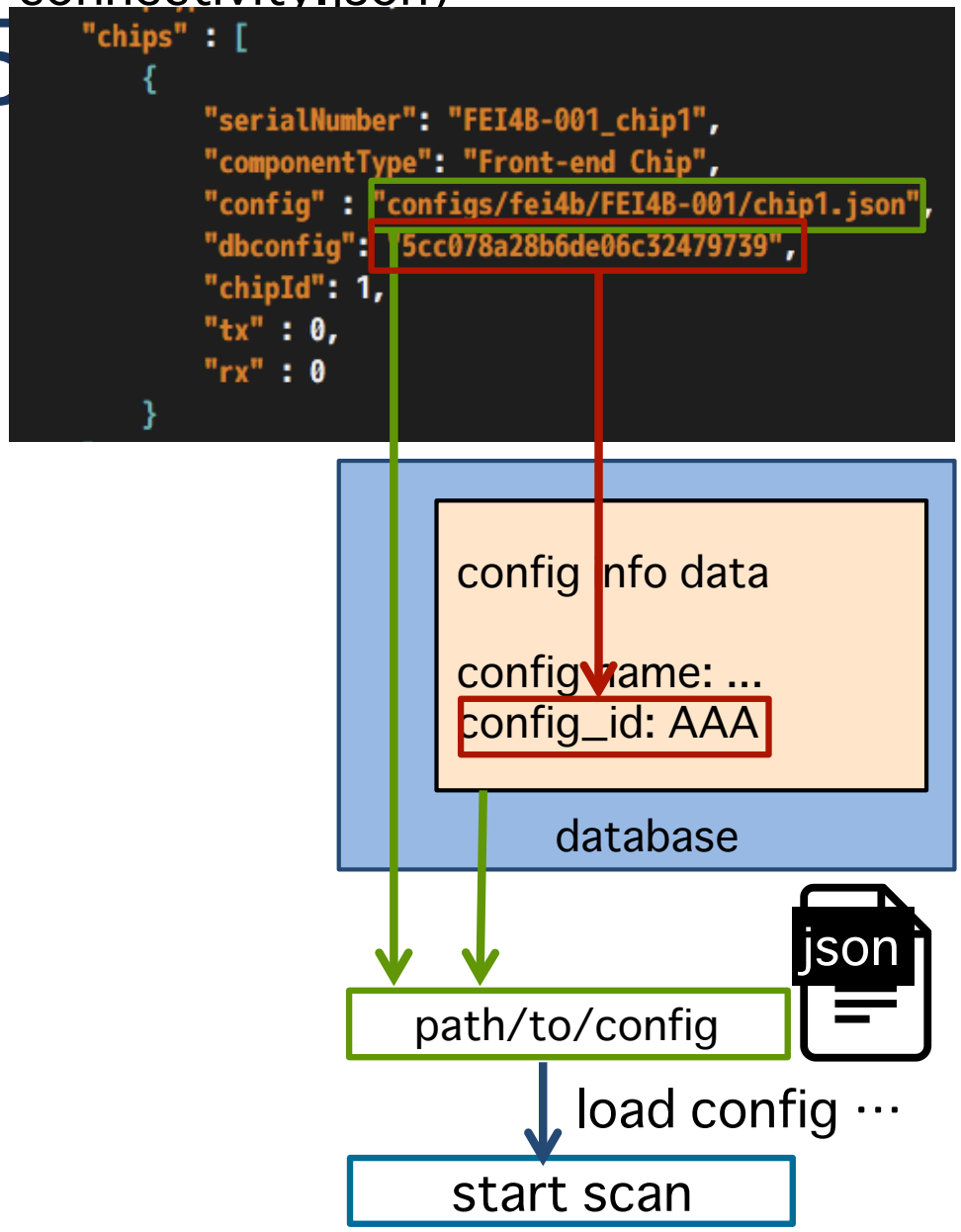
connectivity.json)



Config Retriever (4/5

connectivity.json)

- scanConsole
  - ⓪ Give config id in Viewer (dbconfig)
  - ① download config file in DB by config id
  - ② Create config file from DB
    → path/to/config
  - ③ Load config file (path/to/config)



Config Retriever (5/5

connectivity.json)

```
"chips" : [
    {
        "serialNumber": "FEI4B-001_chip1",
        "componentType": "Front-end Chip",
        "config" : "configs/fei4b/FEI4B-001/chip1.json",
        "dbconfig": "5cc078a28b6de06c32479739",
        "chipId": 1,
        "tx" : 0,
        "rx" : 0
    }
```

- scanConsole
  - ⓪ Give config id in Viewer (dbconfig)
  - ① download config file in DB by config id
  - ② Create config file from DB
    → path/to/config
  - ③ Load config file (path/to/config)

- config creation
  - 1. Get config data from DB
  - 2. Modify some values in config data for
    - chip name → serial number
    - chipId → change to ID written in connectivity config

    so enable to scan continuously after downloading