The Design of Data Management for a New Plasma Control System

J.Q. Zhu[1*], Q.P. Yuan[1,], Z.M. Huang[1,2], J. J. Huang[1,2], R.R. Zhang[1], G. Xu[1,2]

[1] *Institute of Plasma Physics, Hefei Institutes of Physical Science, Chinese Academy of Sciences*

[2] *University of Science and Technology of China*

*Email: jqzhu@ipp.ac.cn

Outline

Motivation

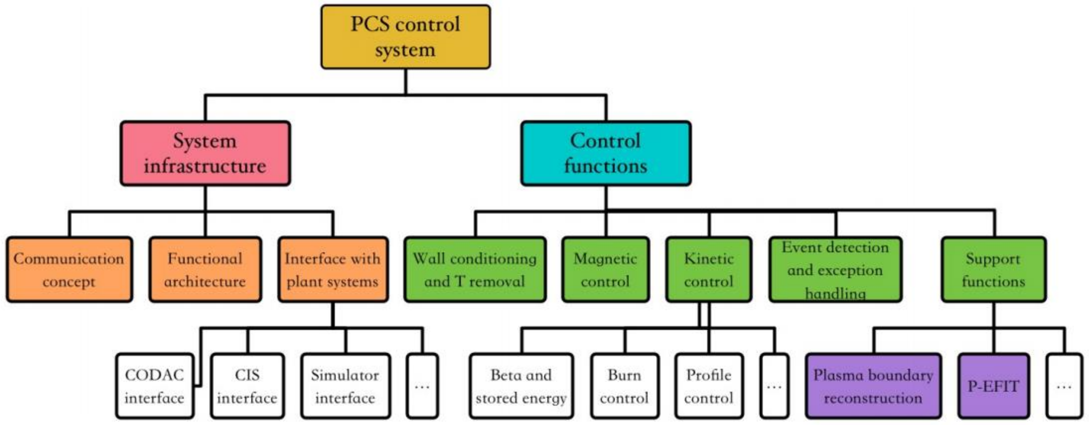
Data Management Design

Test Results

Summary

Motivation

□ Plasma Control System

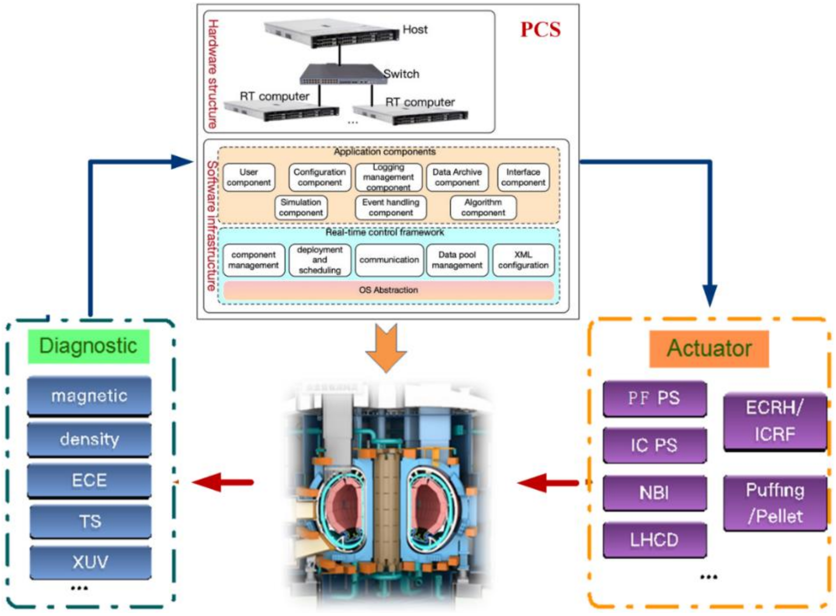


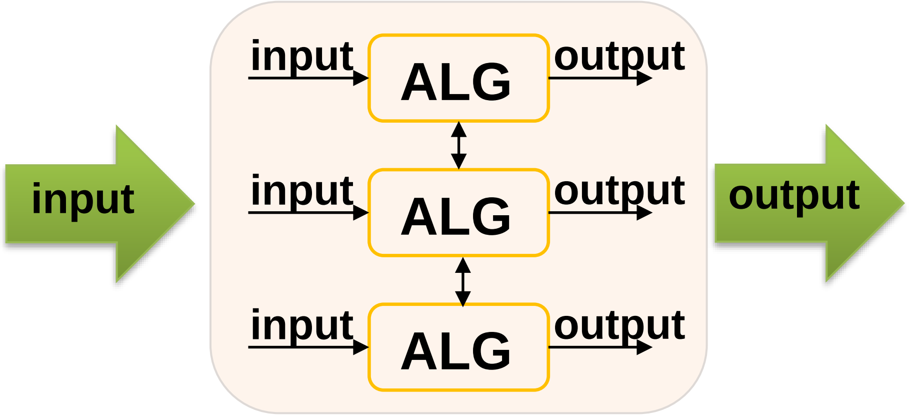
An overall PCS include:

- ➢ infrastructure development
- ➢ control algorithms development.

From a data perspective, the role of PCS is to obtain input data, process data, and output data, which also applies to individual algorithm in the system

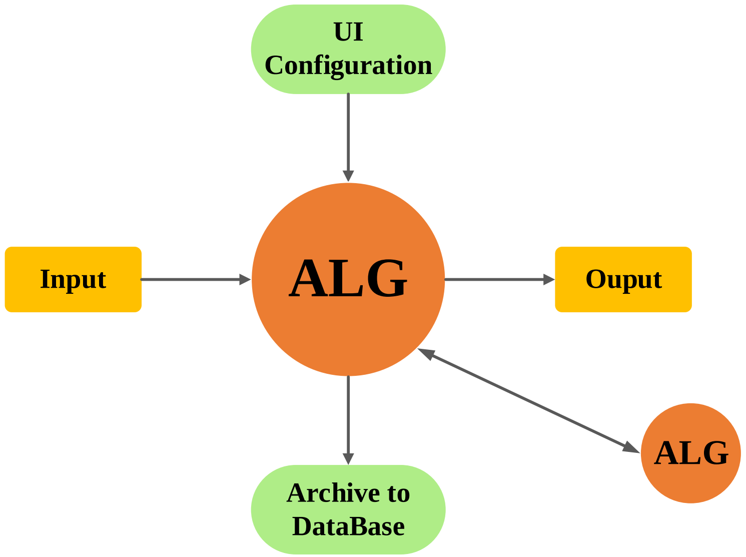
□ Data standpoint

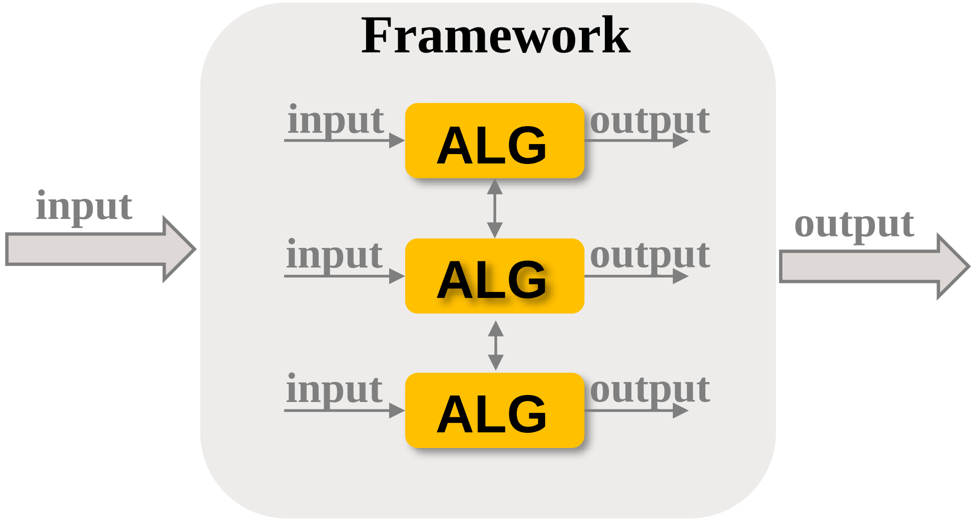




What needs to be done for Data management in PCS

The PCS infrastructure should provide data flow management functionality to support algorithms for real-time data processing and computation.





In detail， the data management system should provide

- ➢ a means for algorithms to input and output data.
- ➢ communications services between algorithms.
- ➢ configuration parameters for algorithm.
- ➢ data archiving services for algorithm.

The desired performance characteristics include:

□ Requires high real-time performance.

□ Support for long-pulse operation.

Outline

Motivation

Data Management Design

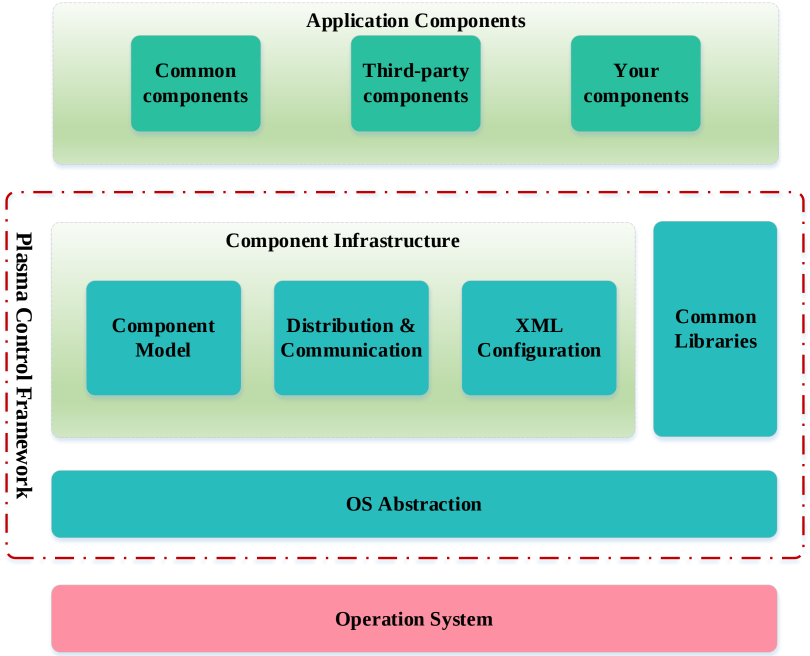
- Software Infrastructure Design
- Data management and Exchange
- Configuration service
- Data Archiving

Test Results

Summary

Software Infrastructure of the new PCS

□ Software infrastructure design



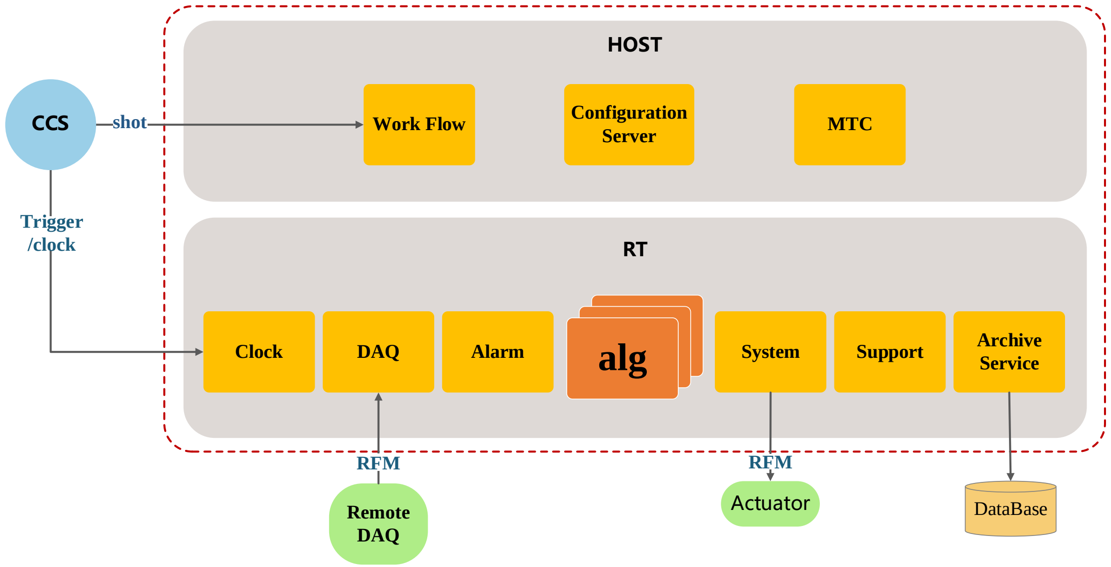
The new PCS software infrastructure is divided into three layers:

① **Operation system**

② **Plasma control framework**: data stream management, execution operations management and interface to other system

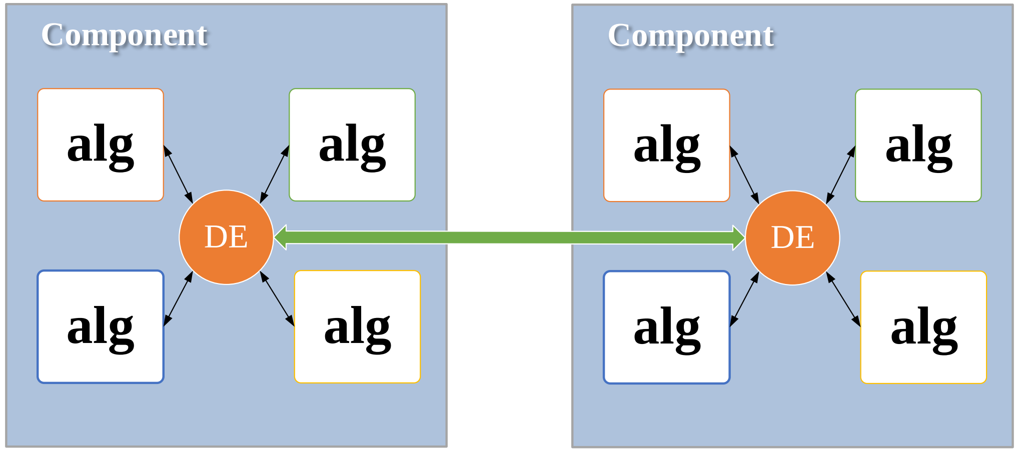
③ **Application components**

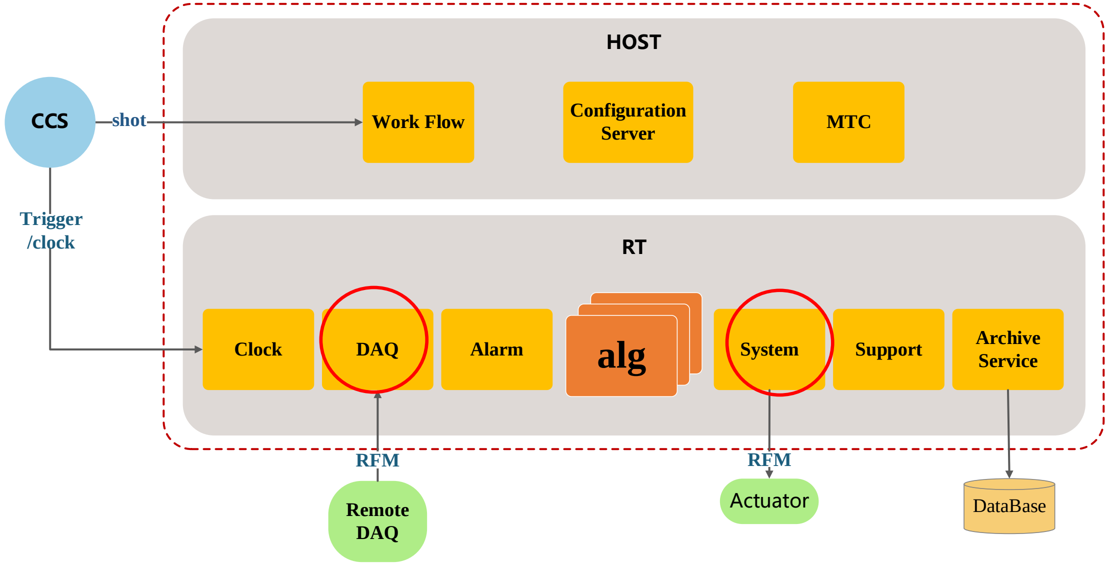
□ Component-based distributed real-time control framework



- The application components is used to implement specific functionalities and operates as an independent process running on a specific cycle.
- A component may contain one or more algorithms that are executed based on a certain time sequence.
- All components of PCS are deployed on two servers (HOST and RT) based on their functional differences.

Data management and Exchange





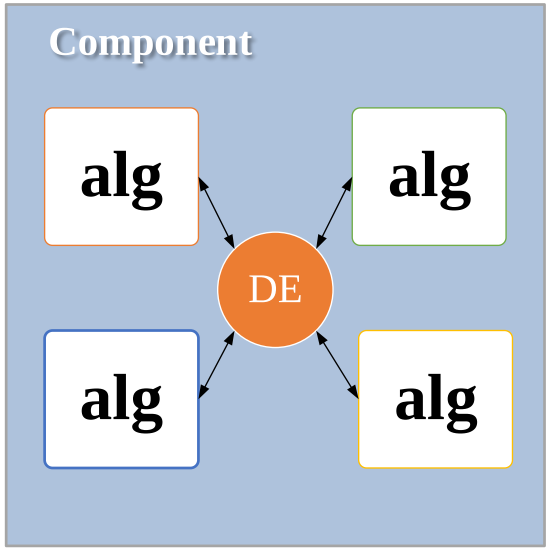
Data exchange

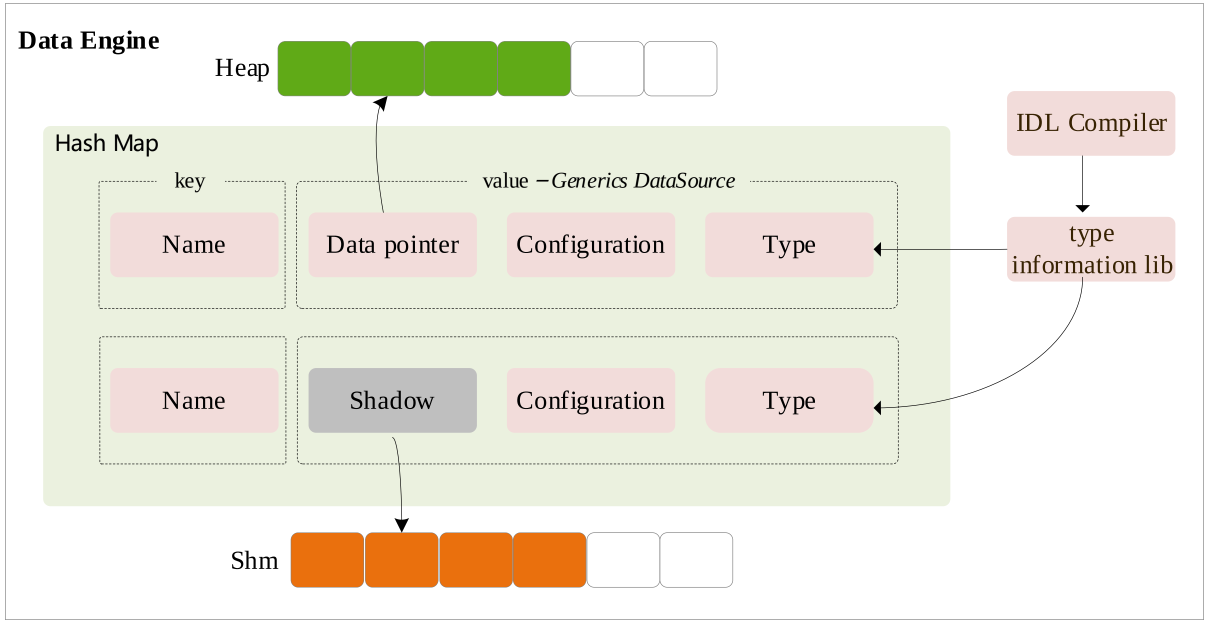
- **Between algorithms within the same component**
- **Between different components**

Since the PCS's data acquisition and output are deployed in the DAQ and System component, respectively, the data acquisition and output processes of the components can be classified as data communication between the component.

Data management and Exchange

□ Data exchange within the same component





Data Engine（DE） deployed within a component,

□ serves as a data management container

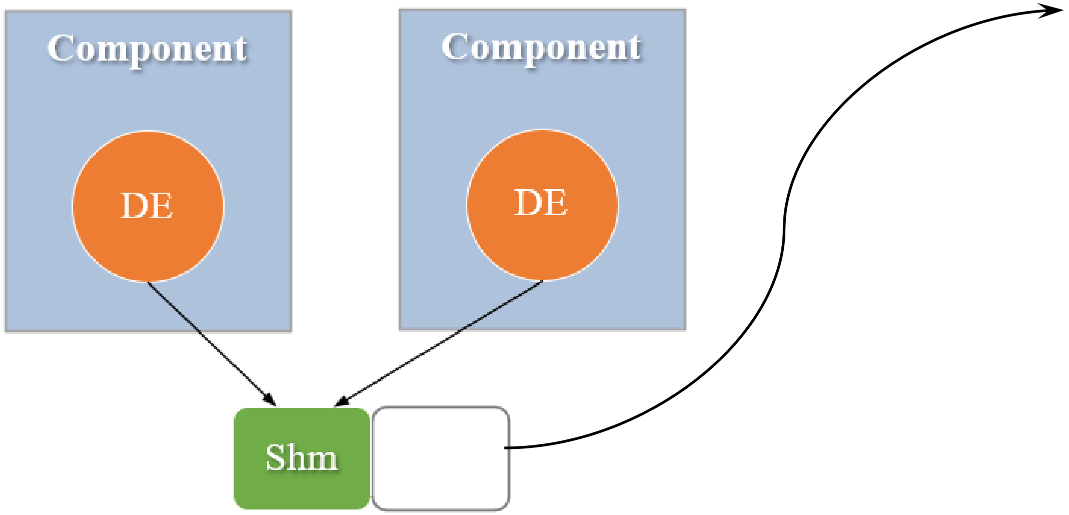
□ facilitates the exchange of data among algorithms

The data engine efficiently organizes data using a hash table, and generic programming is used in implementation to support various types of data.

The data engine supports storing data in the heap as well as other forms of memory, such as shared memory. This flexibility allows for efficient data storage and retrieval, which enable data exchange and interaction among components.

Data management and Exchange

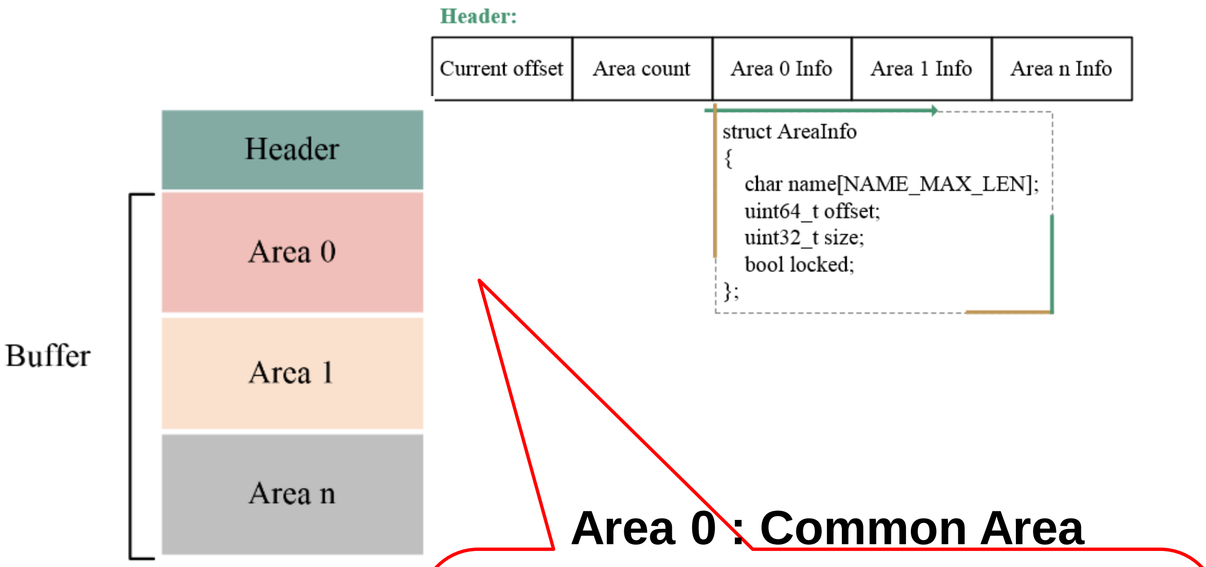
□ Data exchange between different components



Data can be registered within the data engines of different components, where the data is stored in the same shared memory region. Therefore, algorithms can obtain data from other components through data engines

A shared memory management library is provided to

- handle the SHM creation,
- conflict management,
- utilization of shared memory.



Header:

Current offset	Area count	Area 0 Info	Area 1 Info	Area n Info

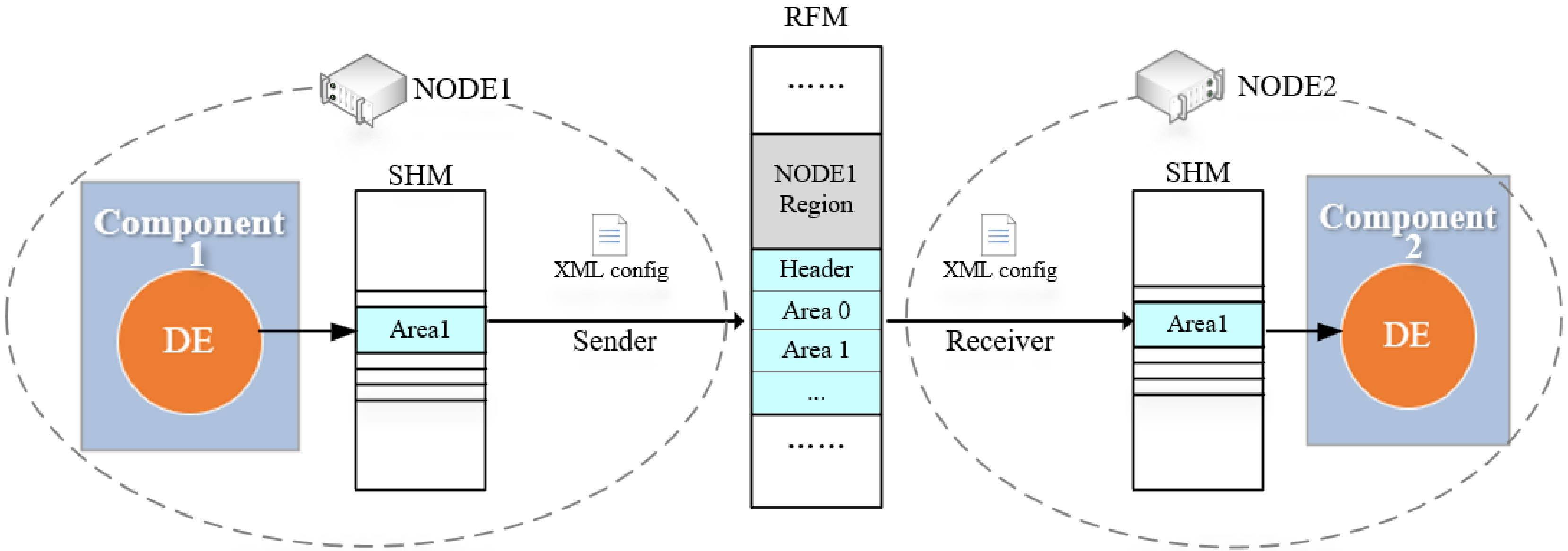
```
struct AreaInfo
{
    char name[NAME_MAX_LEN];
    uint64_t offset;
    uint32_t size;
    bool locked;
};
```

Area 0 : Common Area

a) Shot Information： shot number, stop time…
b) System Deployment: cpu cycle, category count…
c) System state: cpu heartbeat ..

Data management and Exchange

□ Data exchange between components on different servers



To enable communication between components on different servers, specific software programs and hardware communication devices are required.

① Hardware communication devices: RFM(Reflective Memory)

② Software programs: Sender & Receiver

Data management and Exchange

□ Applications of Data Engine

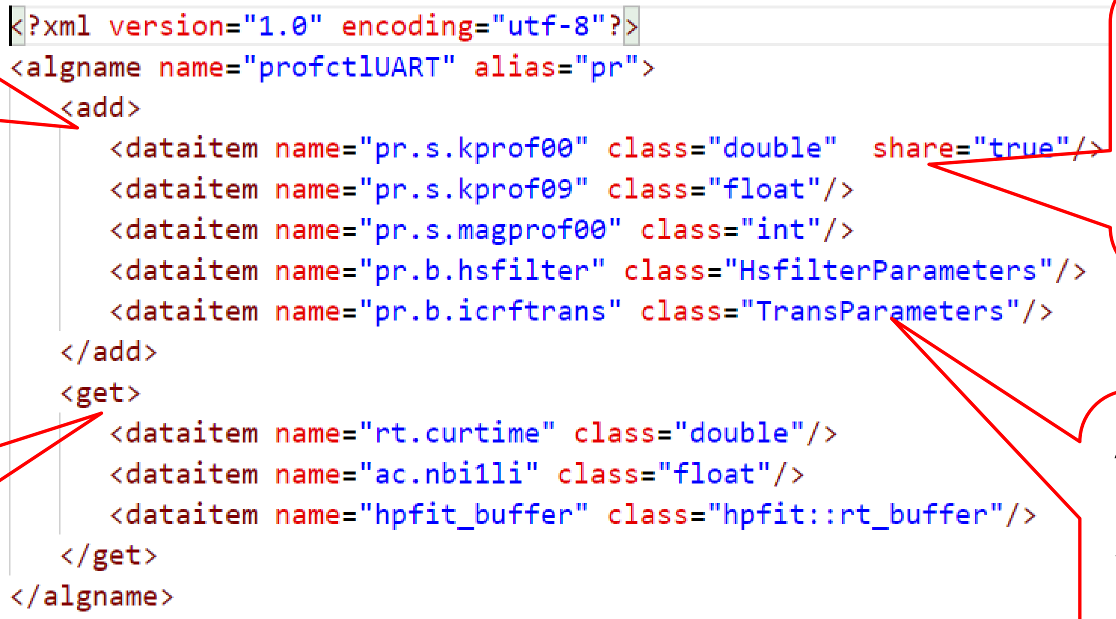
Only an **XML file for each component** is required.

the XML is divided into two sections:

- <add> the data registration area
- <get> the data retrieval area.

The data newly registered in the data engine by the algorithm.

The data is desired to be gotten from other components.

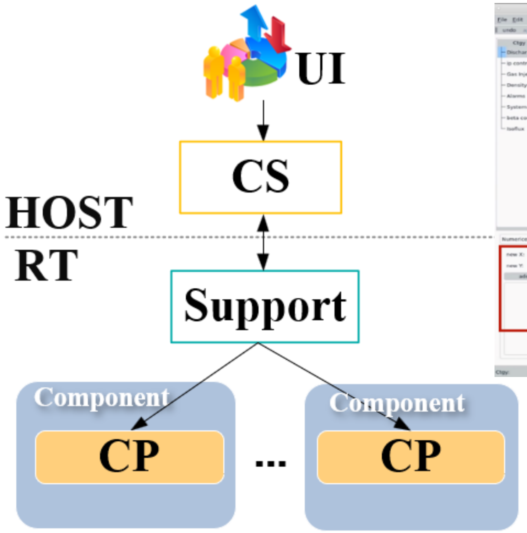


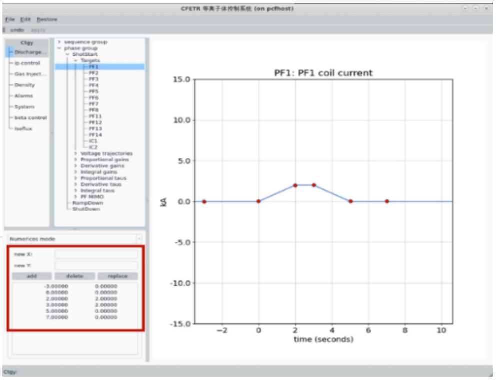
```
<?xml version="1.0" encoding="utf-8"?>
<algname name="profctlUART" alias="pr">
   <add>
      <dataitem name="pr.s.kprof00" class="double"  share="true"/>
      <dataitem name="pr.s.kprof09" class="float"/>
      <dataitem name="pr.s.magprof00" class="int"/>
      <dataitem name="pr.b.hsfilter" class="HsfilterParameters"/>
      <dataitem name="pr.b.icrftrans" class="TransParameters"/>
   </add>
   <get>
      <dataitem name="rt.curtime" class="double"/>
      <dataitem name="ac.nbi1li" class="float"/>
      <dataitem name="hpfit_buffer" class="hpfit::rt_buffer"/>
   </get>
</algname>
```

“share” flag indicates that the data is located in SHM and can be gotten by other components.

All data types, including custom data structures, are allowed to be added to Data Engine.

Configuration service



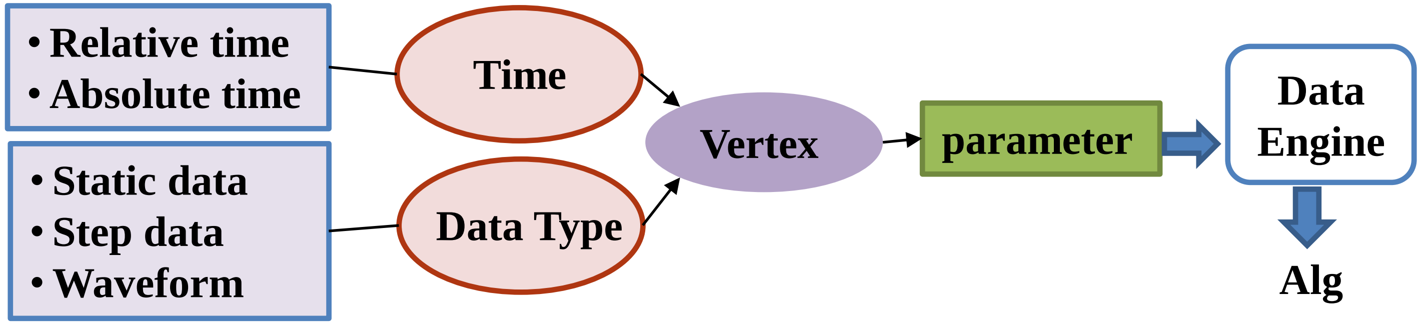


-3.00000	0.00000
0.00000	0.00000
2.00000	2.00000
3.00000	2.00000
5.00000	0.00000
7.00000	0.00000

Before shot: All configuration parameters, including target waveforms, execution sequences, and static parameters are set in UI and managed by CS, saved as vertex groups.

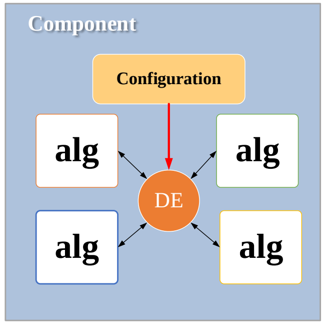
Ready for shot: Support component requests vertex groups from CS and distributes it to various components.

During shot: CP parses vertex into the parameter required by each algorithm at each moment.



Configuration service is composed four pars:

- **UI:** user interface
- **CS:** configuration server, managing UI data & responding to RT configuration requests.
- **Support:** requesting configuration from CS and distributing it to component.
- **CP:** configuration parsing, deployed in each component. Real-time parameter parsing during shot.



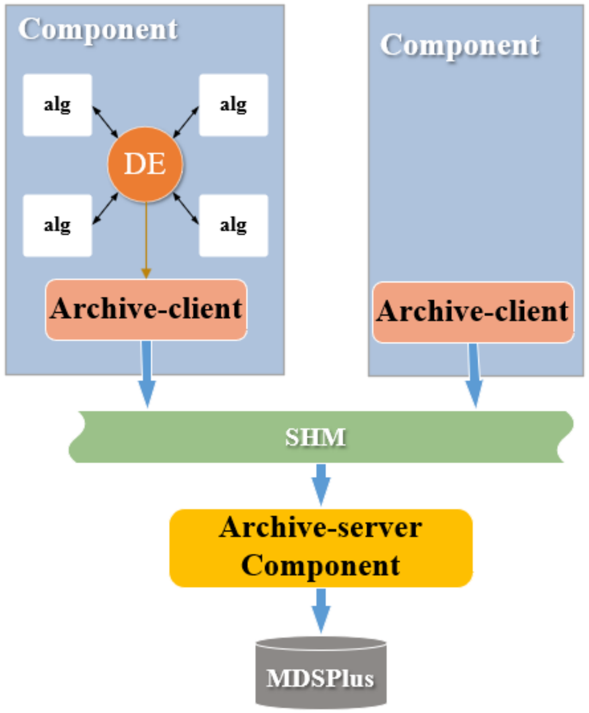
✓ Improved real-time responsiveness

✓ supports long-pulse discharge

✓ Dynamic parameter parsing according algorithm change.

Data Archiving

□ Data archived in real-time mode

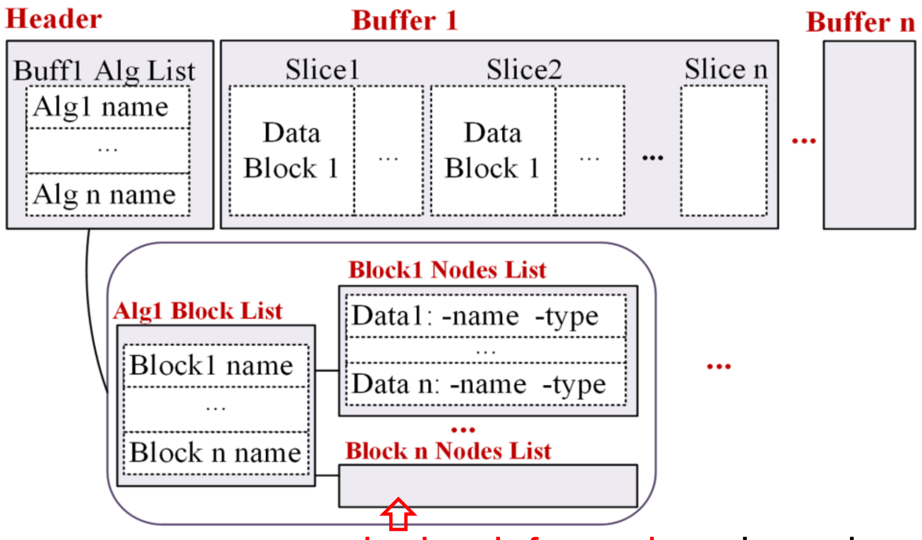


In the new PCS, archiving service is divided into two parts:

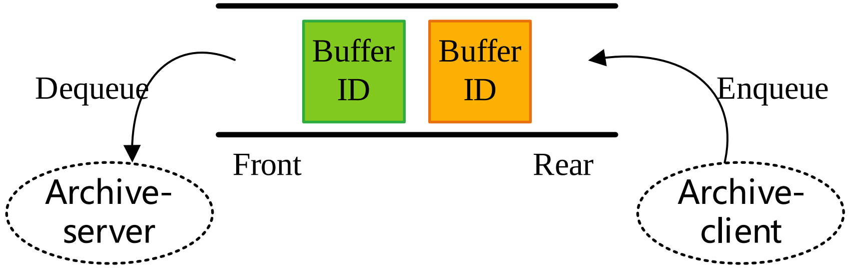
Archive-client: deployed within the algorithm component, responsible for collect the data in real-time and transmit it to SHM.

Archive-server: deployed within a separate component, responsible for organizing a continuous segment of data from SHM and storing it into a database.

Structure of SHM: dynamic and multi-buffered.



The names, types, and other information about the algorithm's data are stored in the shared memory before shot. During shot, the data values are stored in SHM, to ensure the real-time archiving of the data.



A queue-based approach to control the order of buffer in SHM and manage conflicts.

Outline

Motivation

Data Management Design

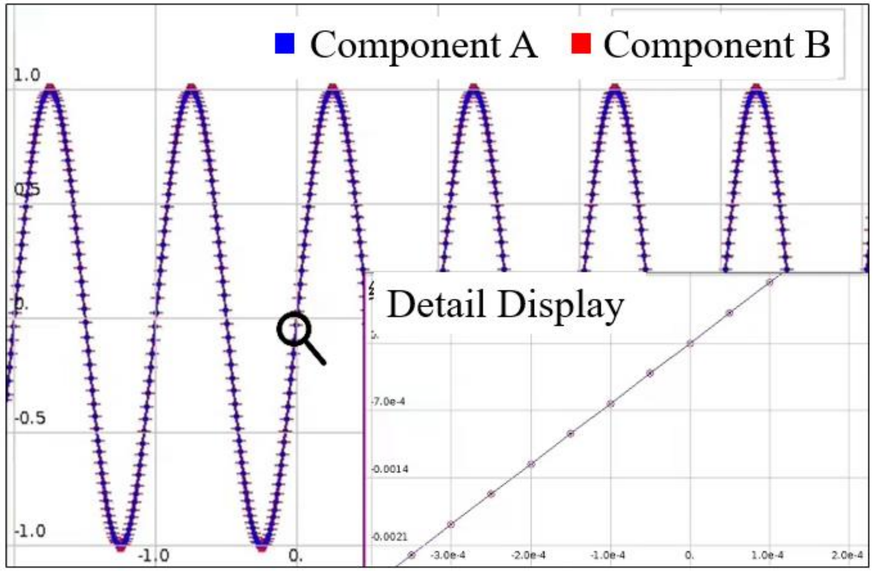
Test Results

Summary

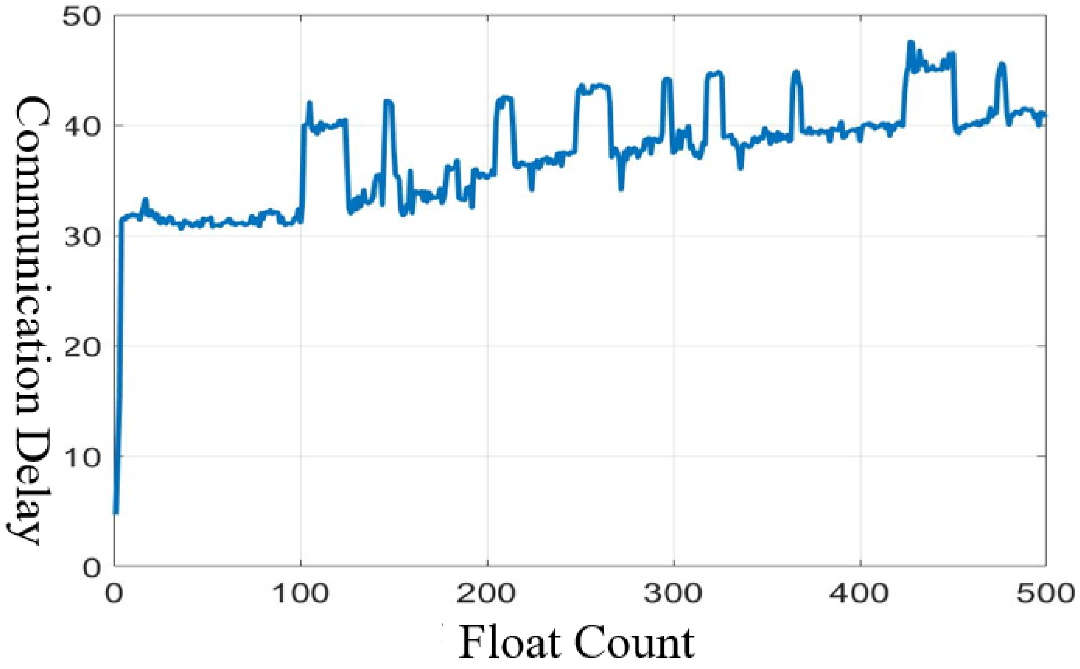
Test Results

The new PCS was successfully applied in the 2023 EAST summer operation campaign, in **total 286 shots with no system failure**.

□ Data exchange latency



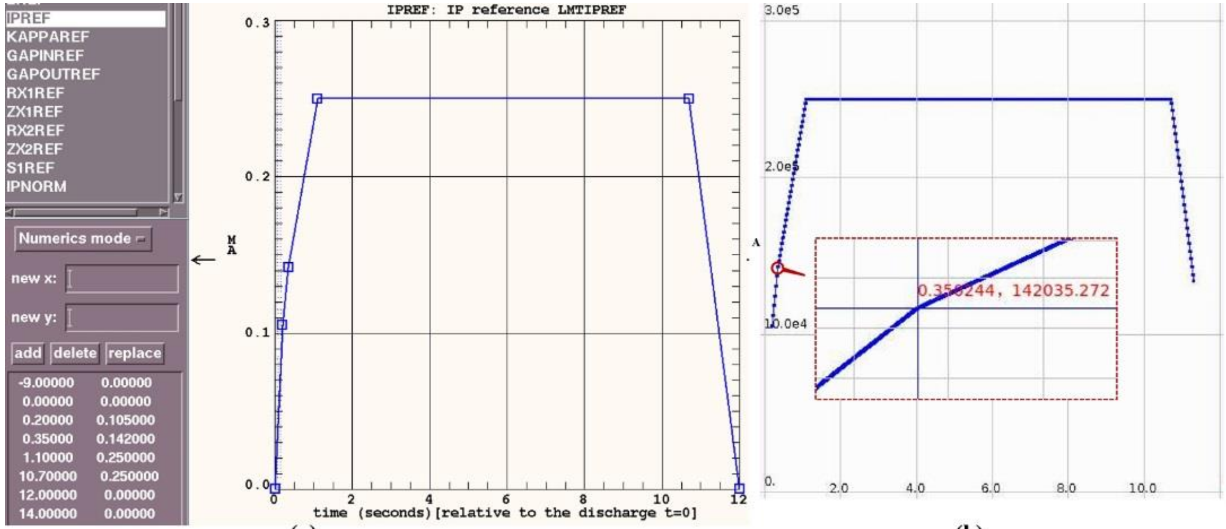
- ✓ The communication latency between components on the same service can be considered negligible.



- ✓ the communication latency between components on different servers is within 50 microseconds When the data size is less than 500 float values, which is the minimum execution cycle of PCS.

Test Results

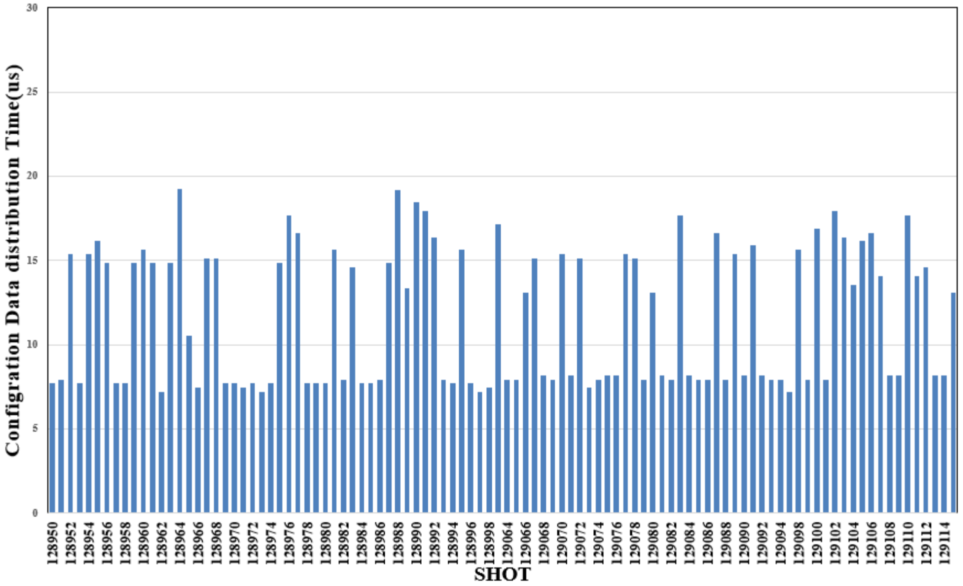
□ Configuration service



-9.00000	0.00000
0.00000	0.00000
0.20000	0.105000
0.35000	0.142000
1.10000	0.250000
10.70000	0.250000
12.00000	0.00000
14.00000	0.00000

(a)

(b)

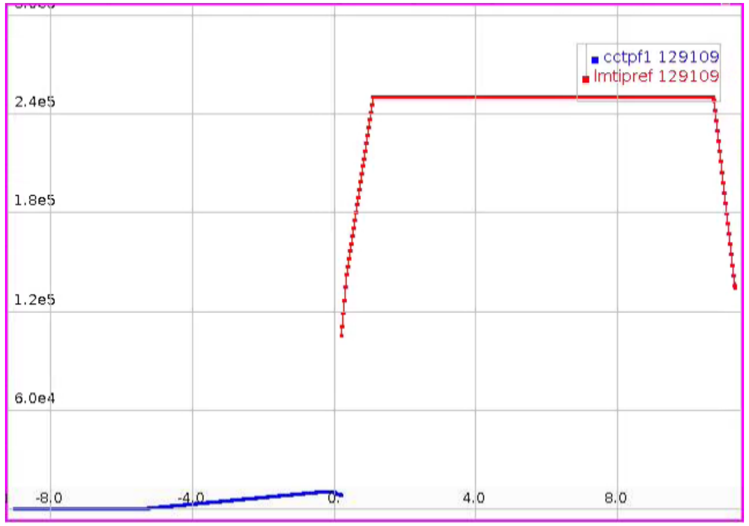


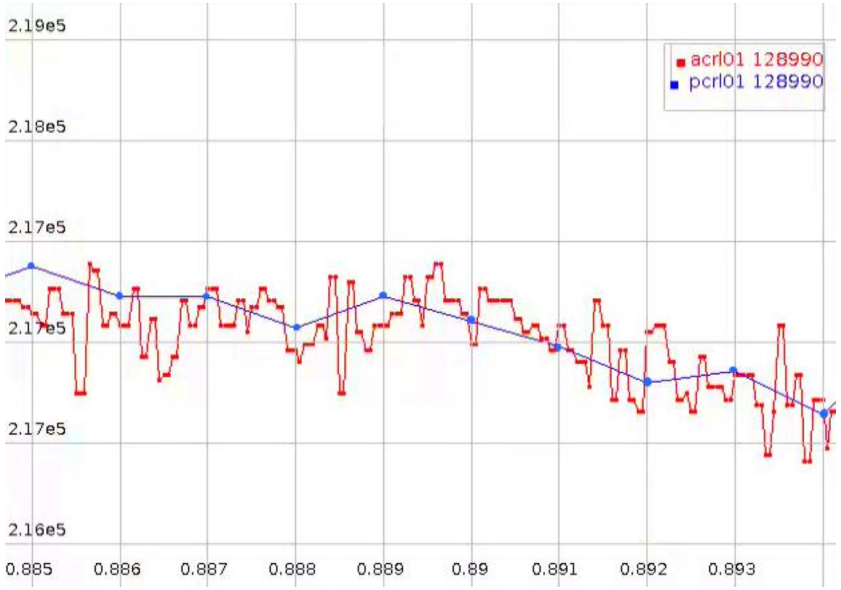
- ✓ The parsed parameters is in agreement to the values set in the user interface.

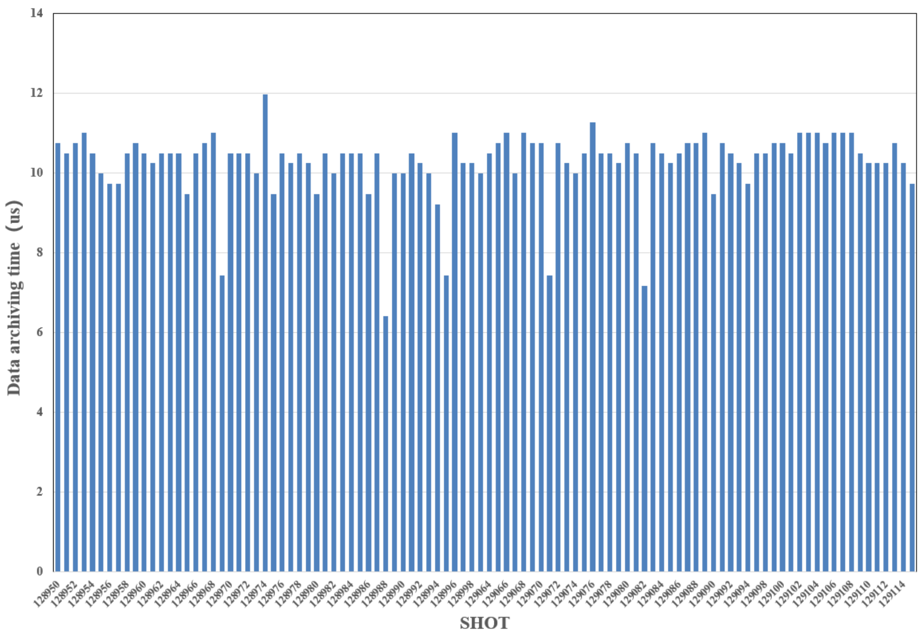
- ✓ The maximum duration in each cycle is less than 20 microseconds.

Test Results

□ Data Archiving

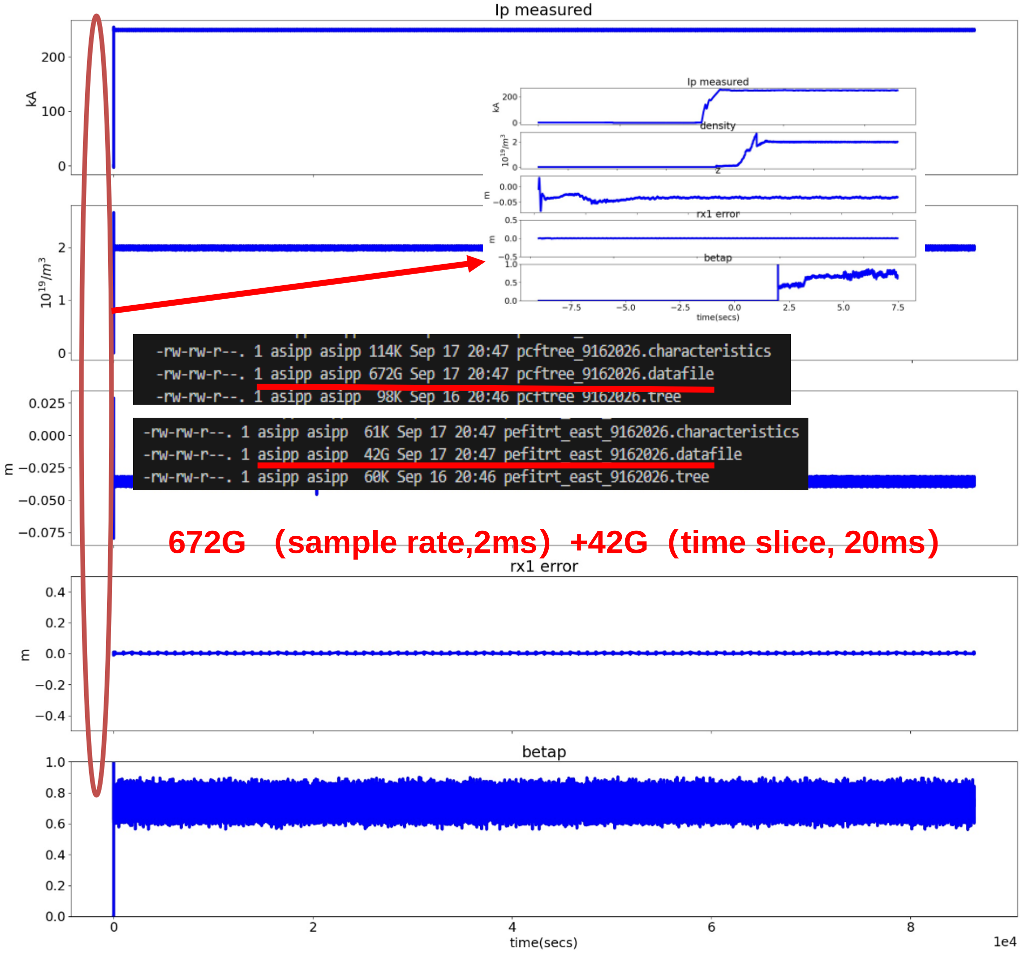






- ✓ The stored data changes as the algorithm changes.
- ✓ All data from each cycle can be saved.
- ✓ With a data archiving service that has a latency of less than 12 microseconds.

Test Results



```
-rw-rw-r--. 1 asipp asipp 114K Sep 17 20:47 pcftree_9162026.characteristics
-rw-rw-r--. 1 asipp asipp 672G Sep 17 20:47 pcftree_9162026.datafile
-rw-rw-r--. 1 asipp asipp  98K Sep 16 20:46 pcftree_9162026.tree
```

```
-rw-rw-r--. 1 asipp asipp  61K Sep 17 20:47 pefitrt_east_9162026.characteristics
-rw-rw-r--. 1 asipp asipp  42G Sep 17 20:47 pefitrt_east_9162026.datafile
-rw-rw-r--. 1 asipp asipp  60K Sep 16 20:46 pefitrt_east_9162026.tree
```

672G （sample rate,2ms） +42G （time slice, 20ms）

✓ The data management system can effectively support the operation of long pulses lasting over 24 hours.

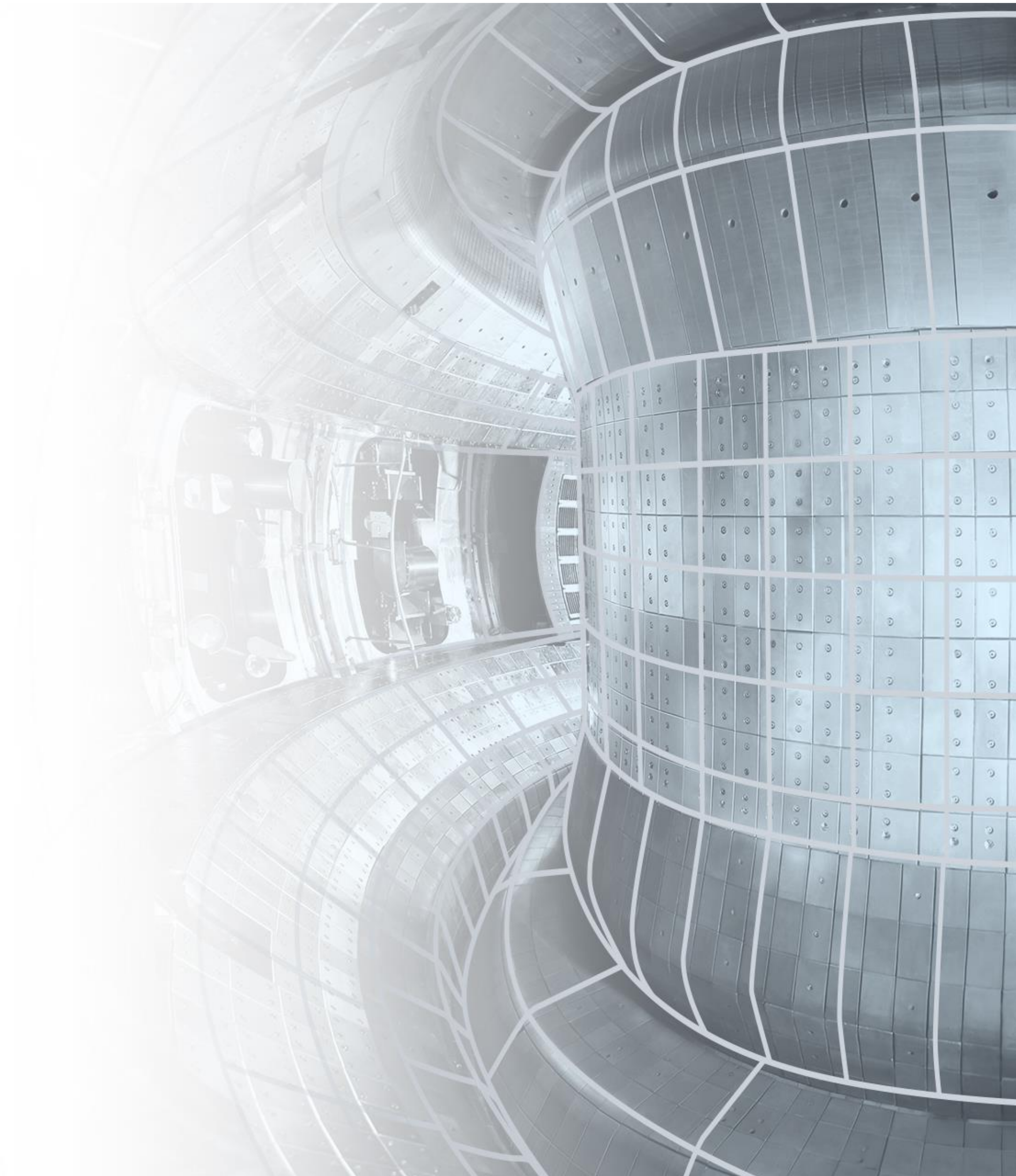
Outline

Motivation

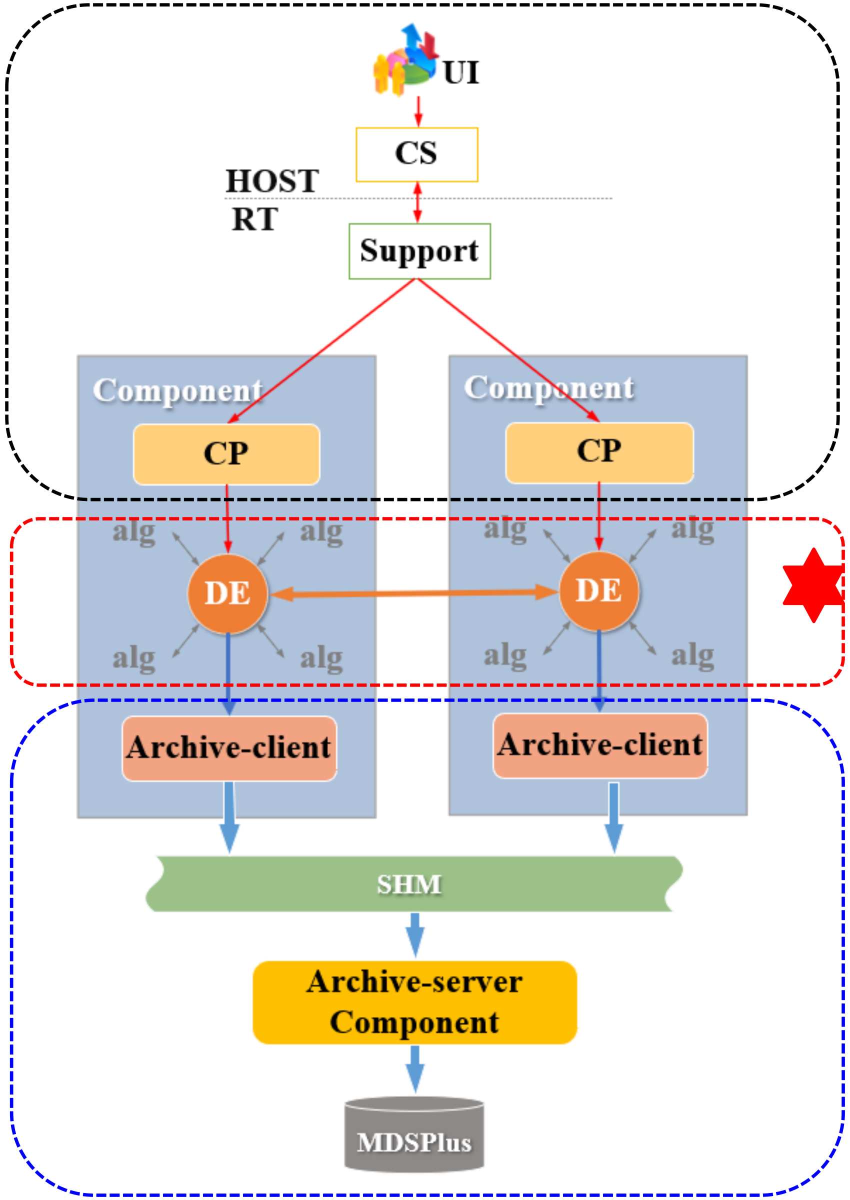
Data Management Design

Test Results

Summary



Summary



- Data communication
- Configuration service
- Data archiving service

CORE: data engine

The Data management system:

① meeting the functional requirements of data management,

② providing excellent real-time capabilities,

③ supporting long pulse discharges.

Thank you !