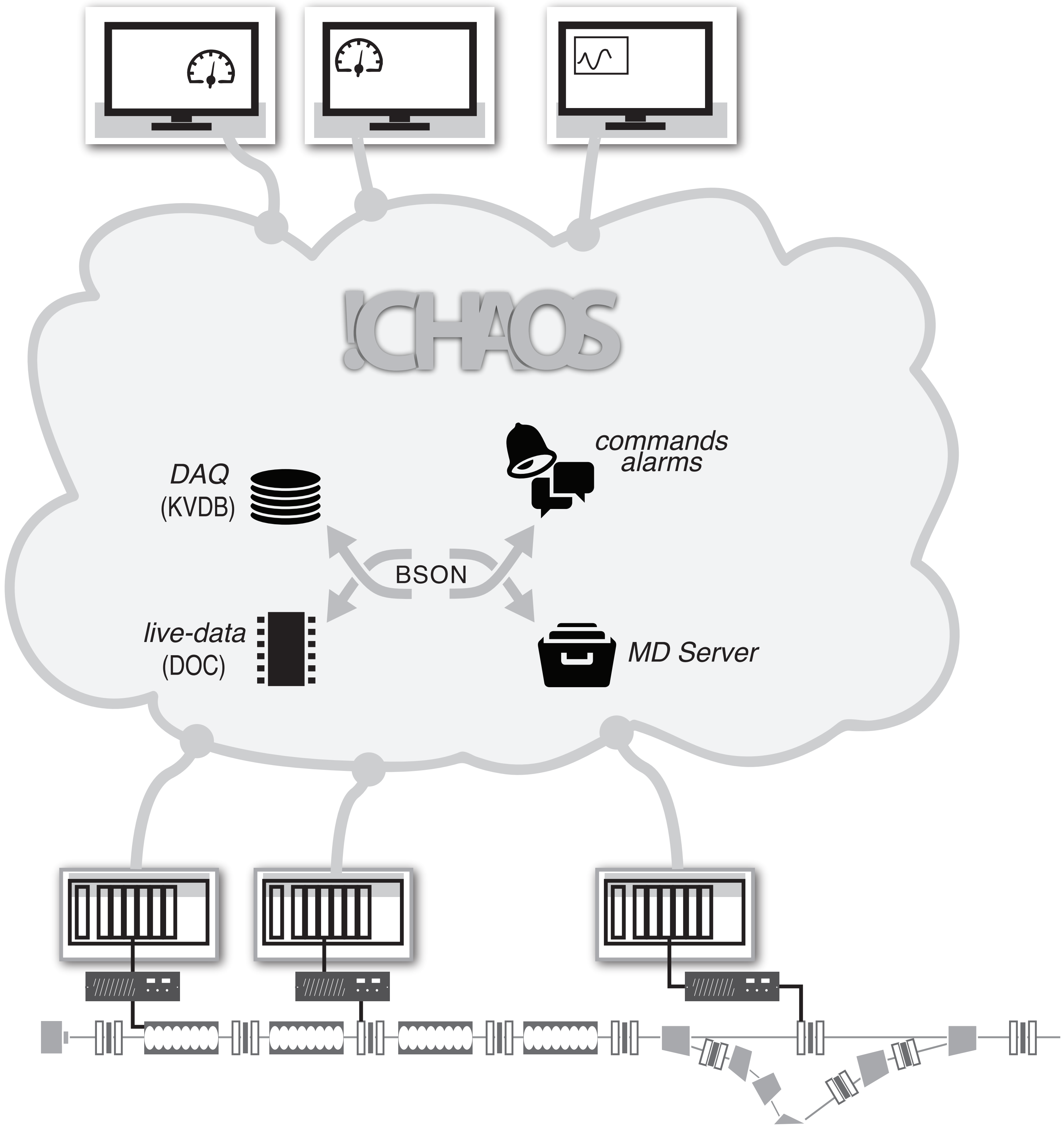
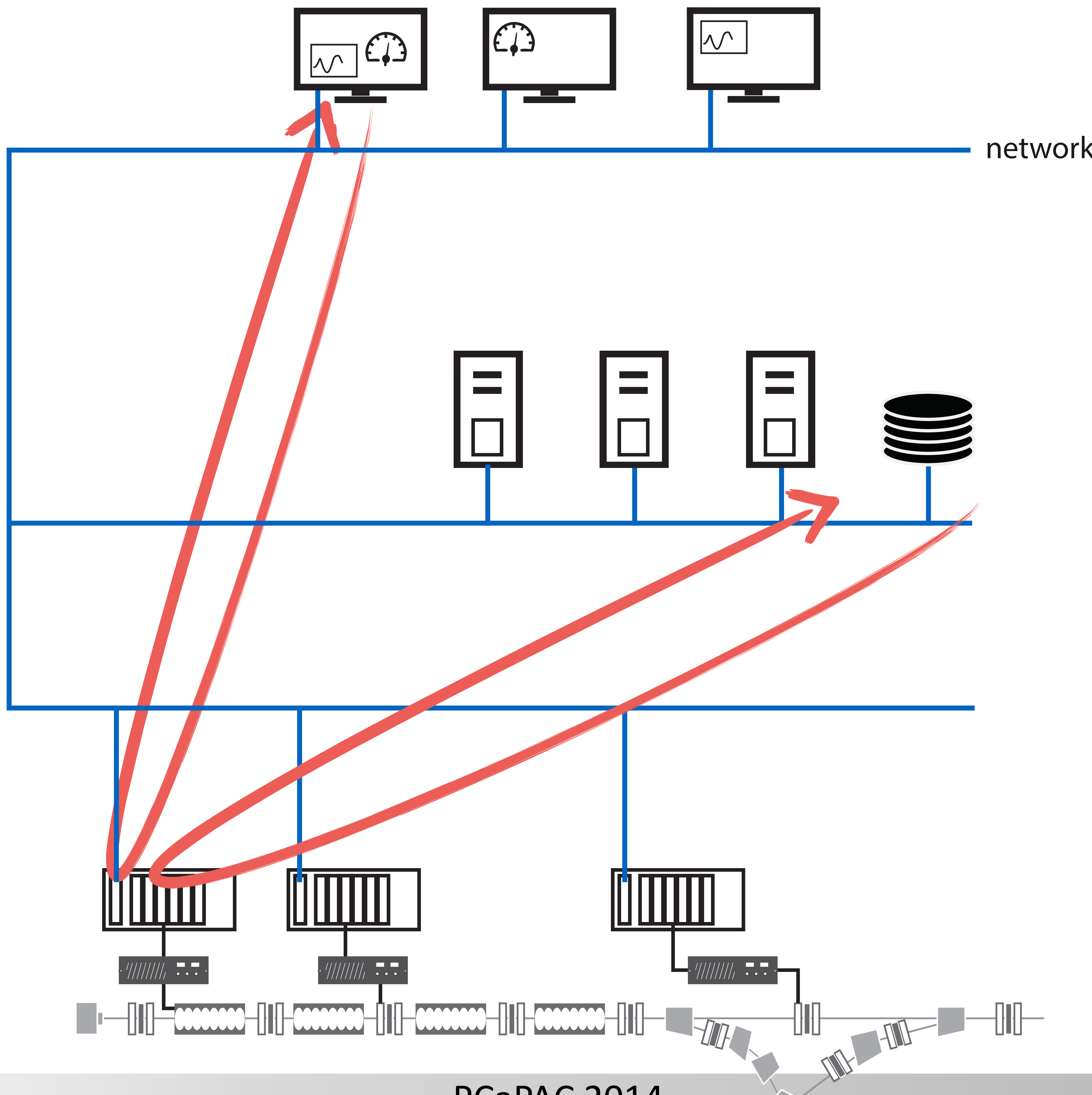
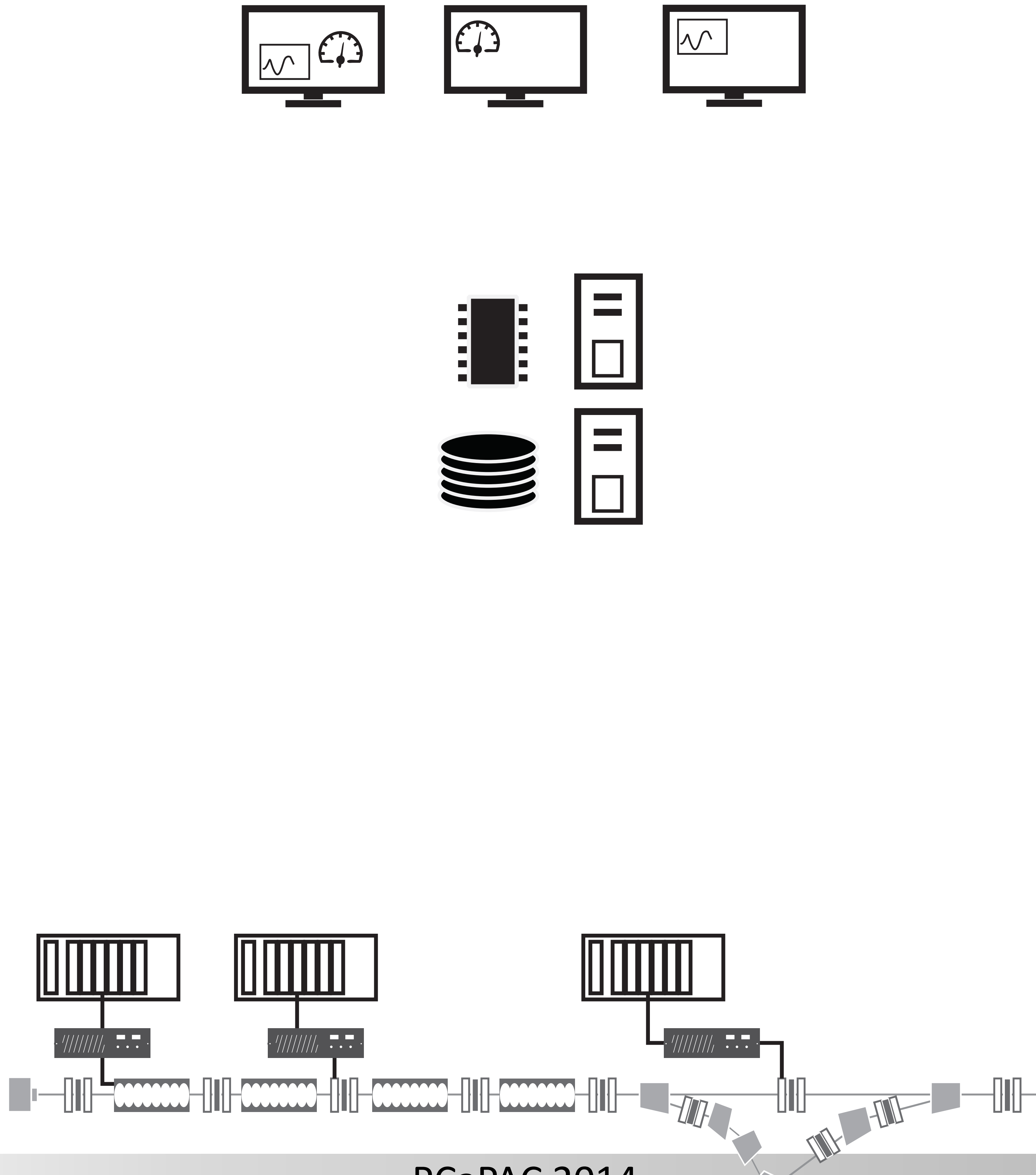
First Operational experience with the !CHAOS Framework

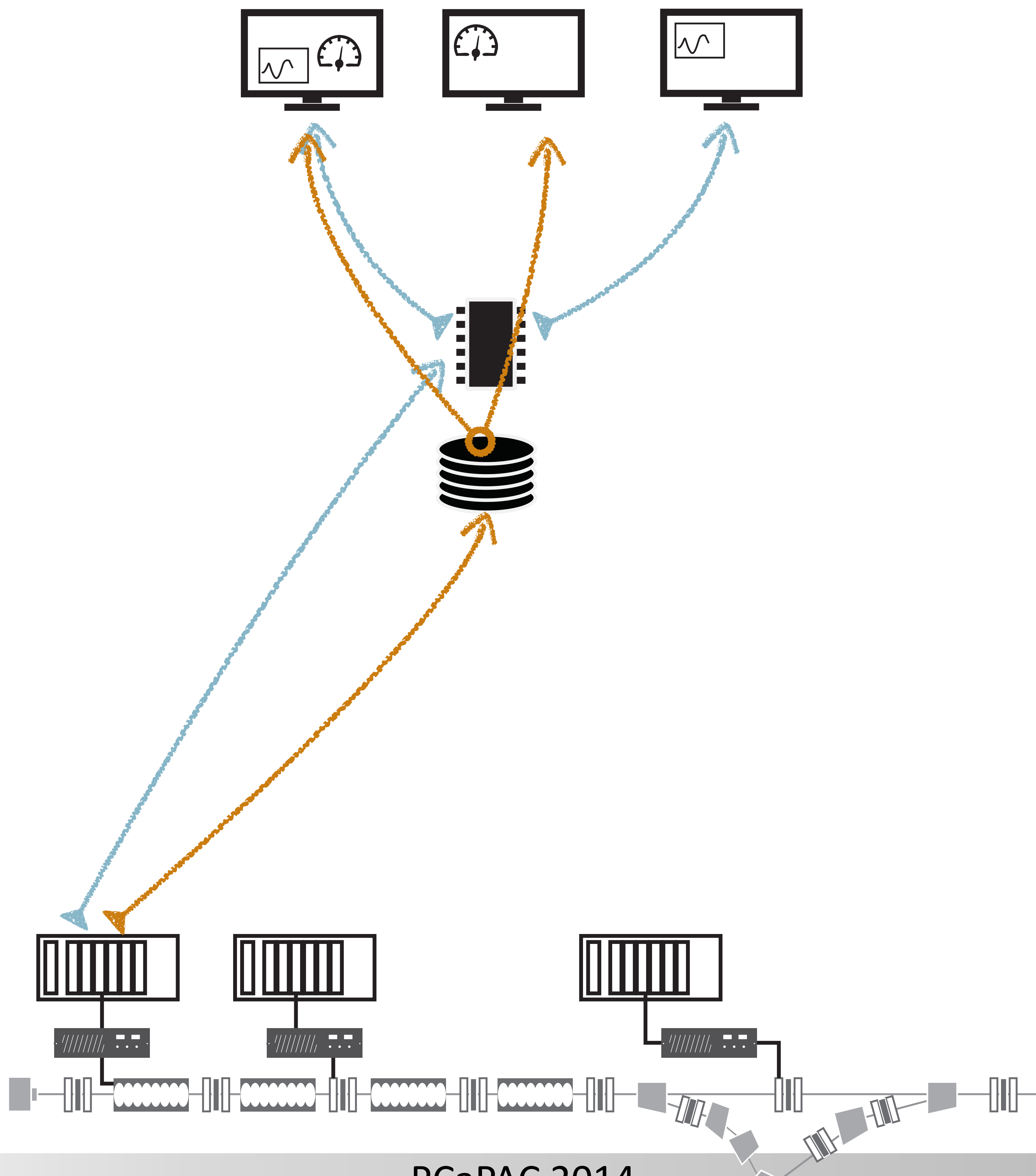
C. Bisegni, L. Catani, P. Ciuffetti, G. Di Pirro, L. Foggetta, F. Galletti,
R. Gargana, E. Gioscio, G. Mazzitelli, A. Michelotti, A. Stecchi
INFN-LNF & INFN-Roma2

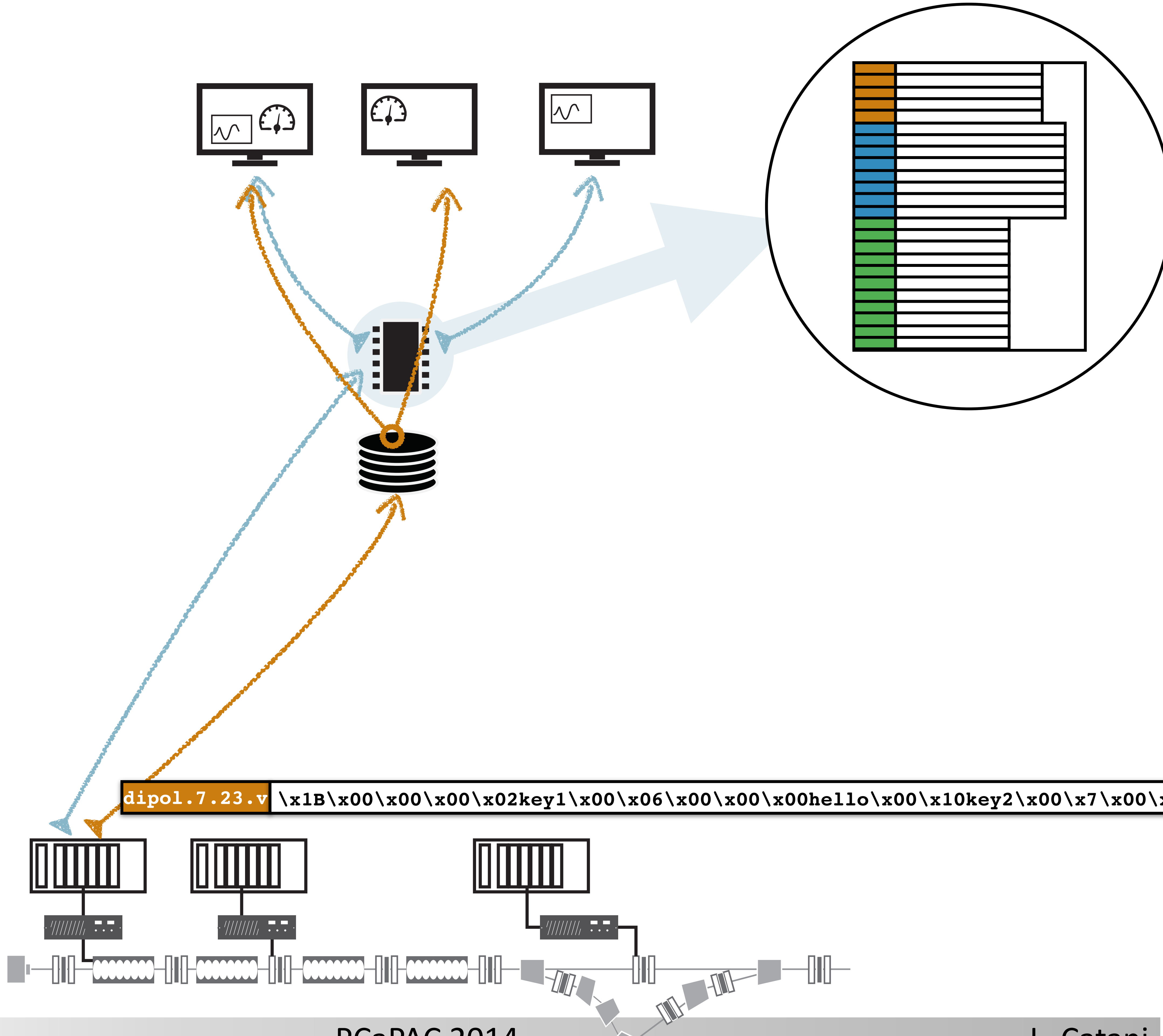
INFN
Istituto Nazionale
di Fisica Nucleare

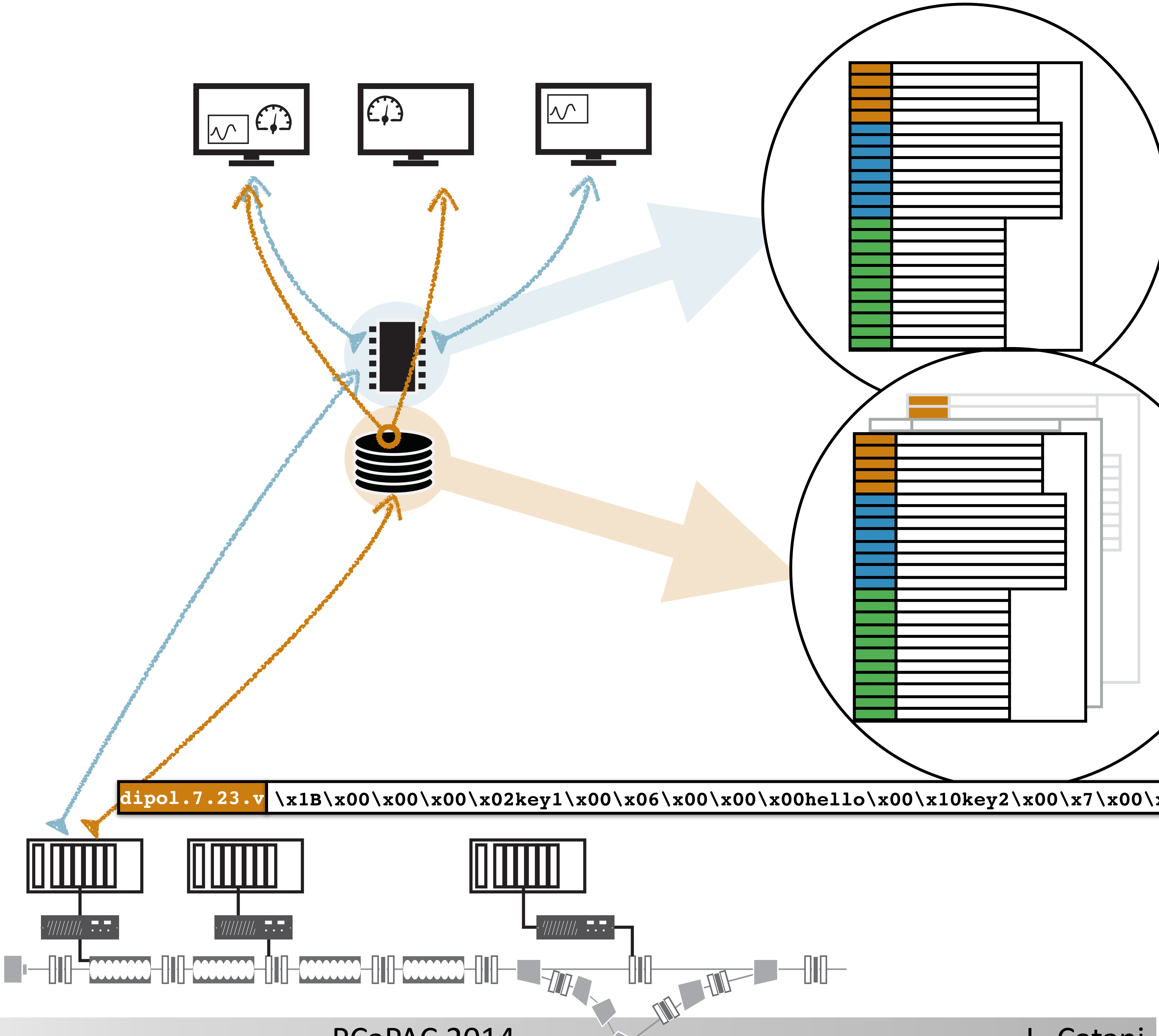

!CHAOS
DAQ
(KVDB)
commands
alarms
BSON
live-data
(DOC)
MD Server

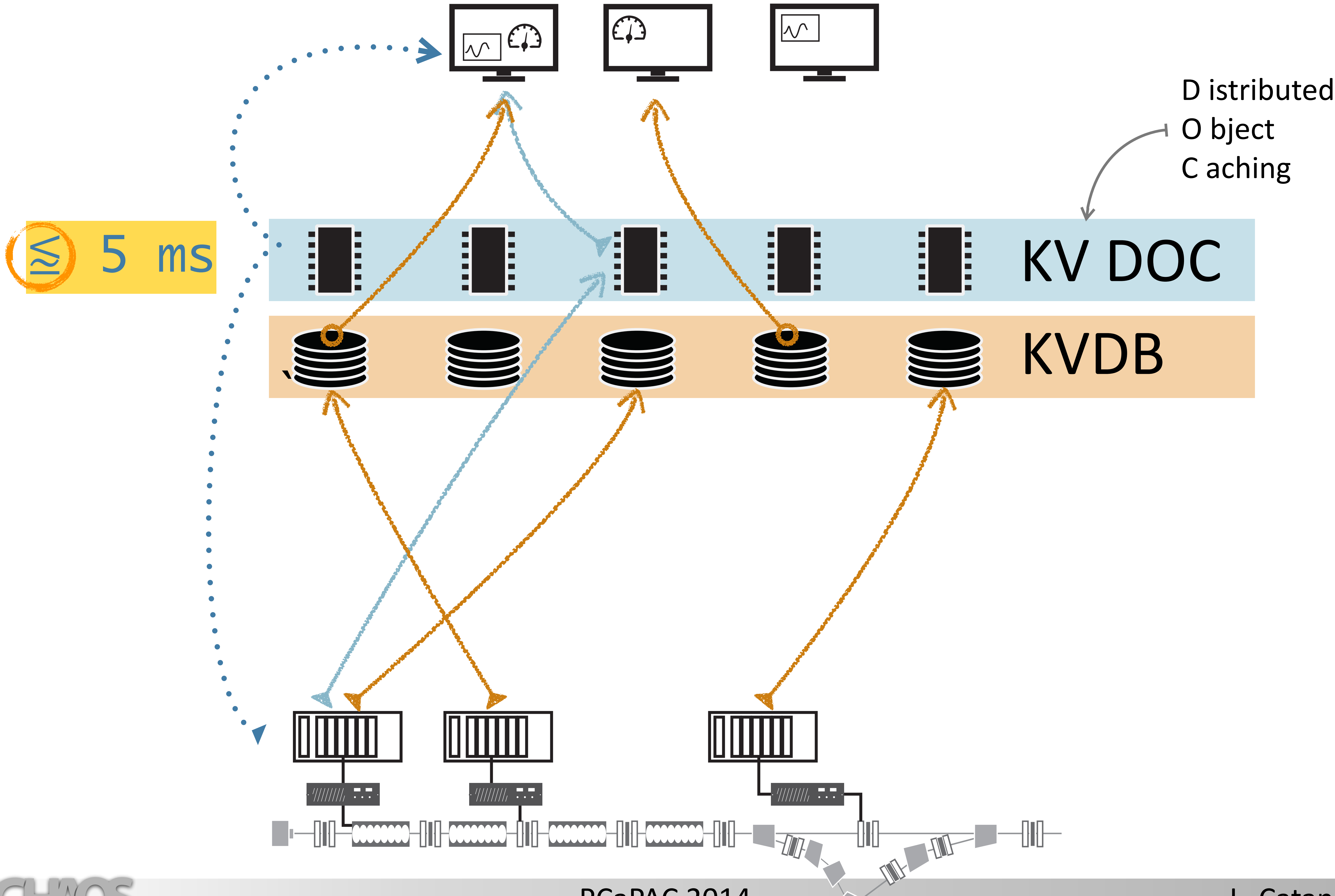

network






dipol.7.23.v
\x1B\x00\x00\x00\x02key1\x00\x06\x00\x00\x00hello\x00\x10key2\x00\x7\x00\


dipol.7.23.v
\x1B\x00\x00\x00\x02key1\x00\x06\x00\x00\x00hello\x00\x10key2\x00\x7\x00\

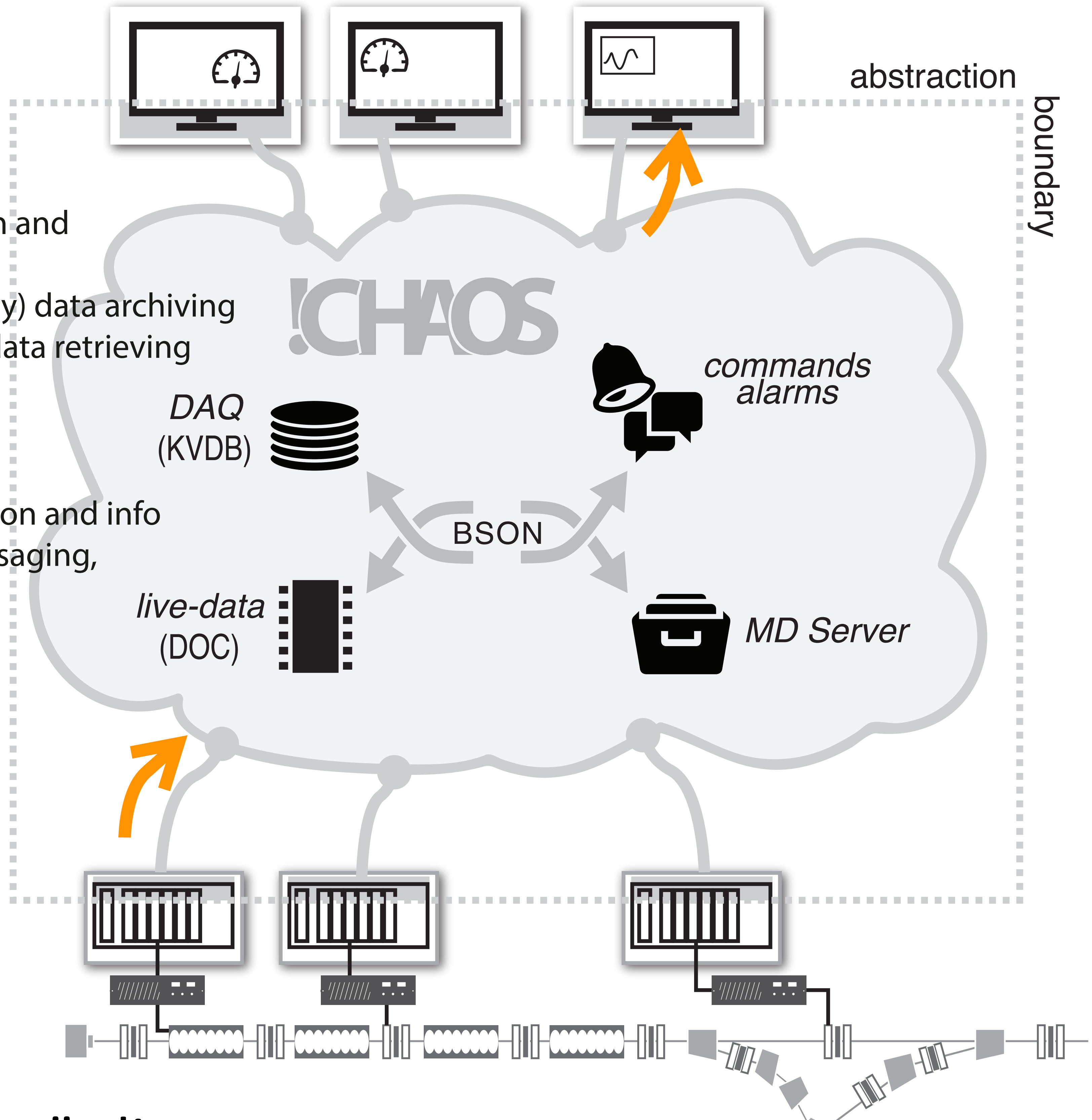


≤ 5 ms
D istributed
O bject
C aching
KV DOC
KVDB

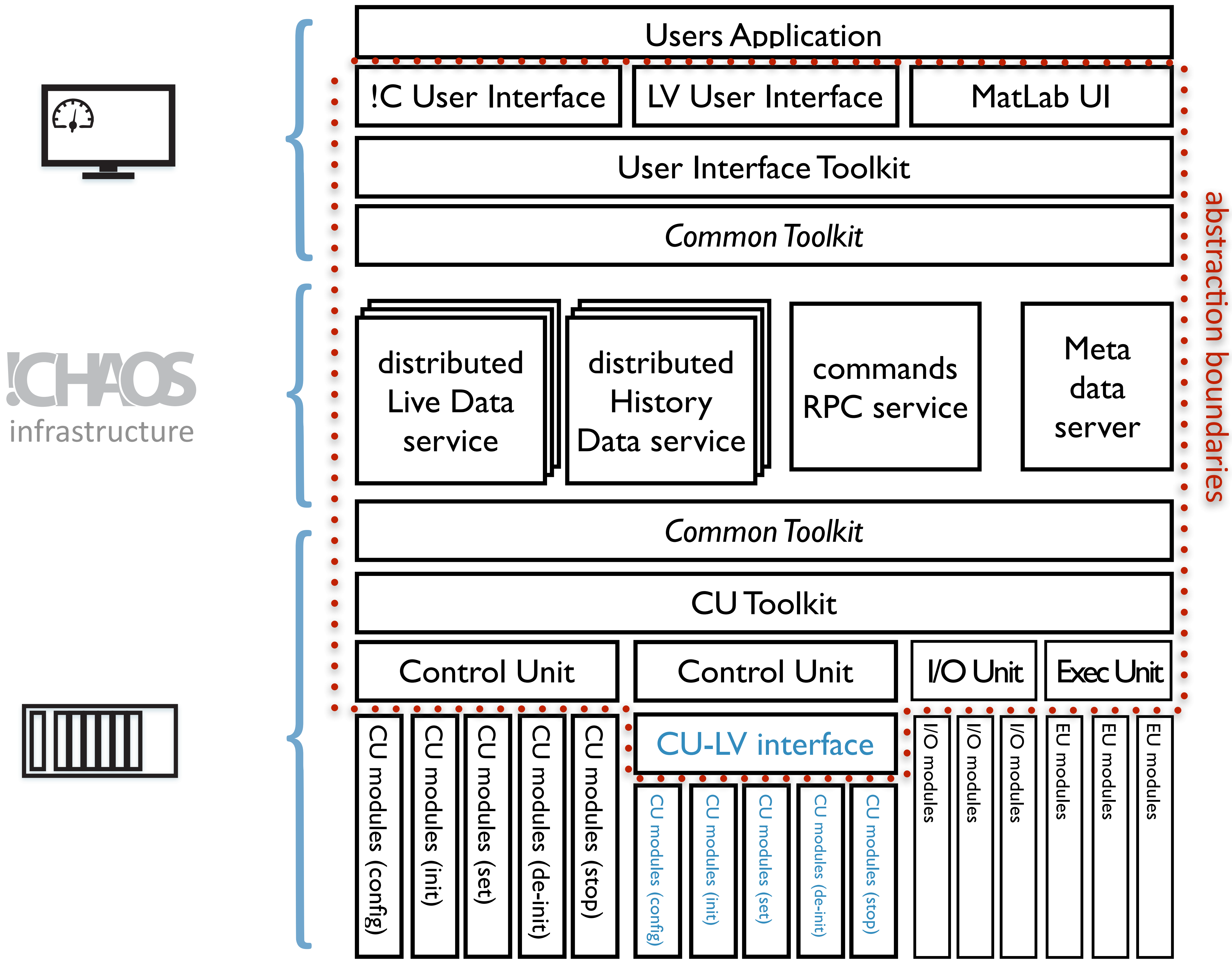
“data consumer” clients

- live data collection and distribution
- continuous (history) data archiving (DAQ) and history data retrieving

- system configuration and info management - messaging, alarms, timing



“data producer” clients



Users Application
!C User Interface
LV User Interface
MatLab UI
User Interface Toolkit
Common Toolkit
!CHAOS
infrastructure
distributed Live Data service
distributed History Data service
commands RPC service
Meta data server
Common Toolkit
CU Toolkit
Control Unit
Control Unit
I/O Unit
Exec Unit
CU modules (config)
CU modules (init)
CU modules (set)
CU modules (de-init)
CU modules (stop)
CU-LV interface
CU modules (config)
CU modules (init)
CU modules (set)
CU modules (de-init)
CU modules (stop)
I/O modules
I/O modules
I/O modules
EU modules
EU modules
EU modules
abstraction boundaries

serialization

formatting data for k/v

key1, key2
hello, 123
world, 456

CSV

key/value data storage:
JSON serialisation

JSON

{{“key1” : “hello”
“key2” : 123}}

BSON

- binary version of JSON
- key - value syntax
- serialised data described by means of:
    - key name
    - type and dimension
    - value

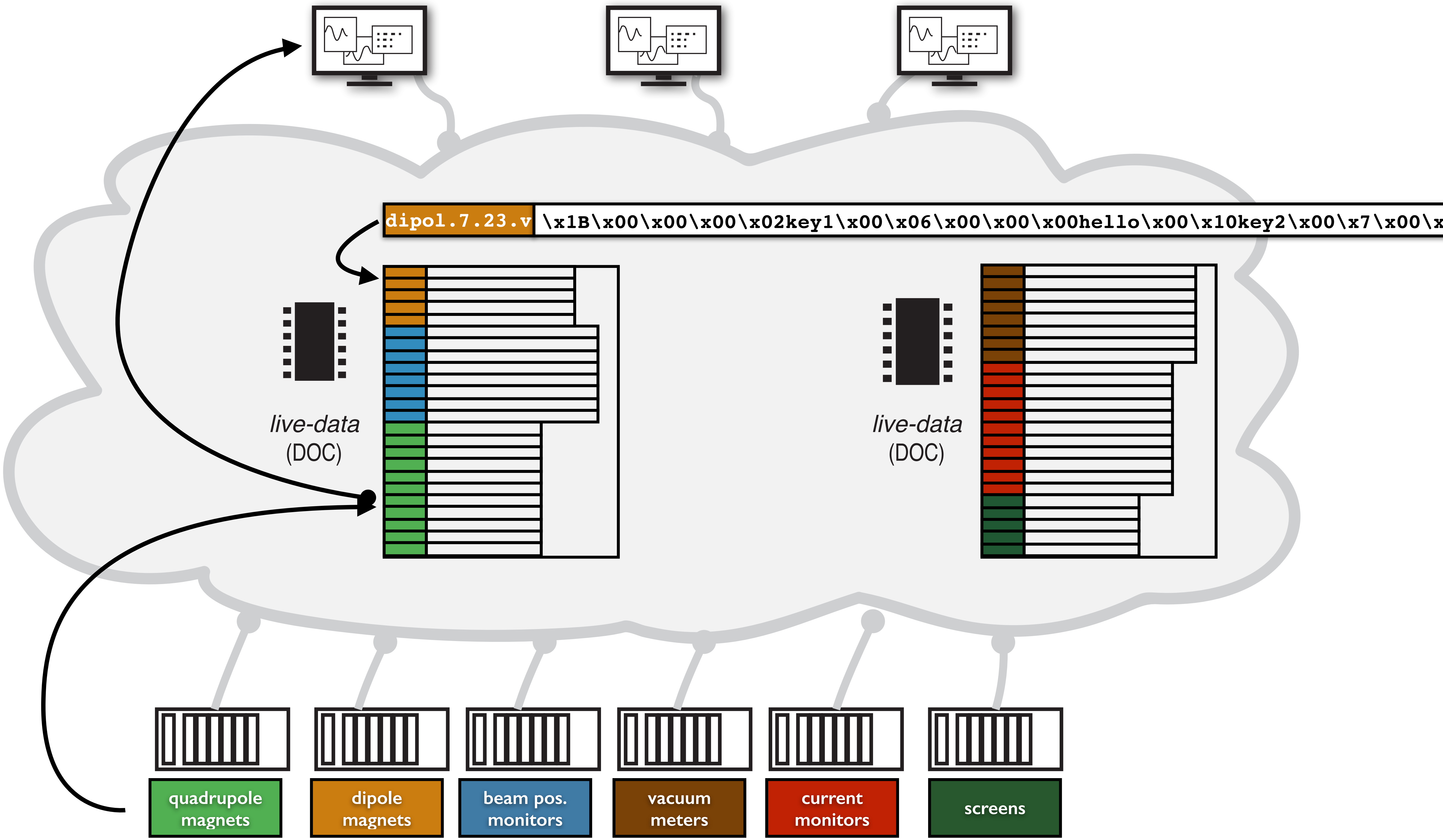
\x1B\x00\x00\x00
\x02key1\x00\x06\x00\x00\x00hello\x00
\x10key2\x00\x7D\x00\x00\x00
\x00

or

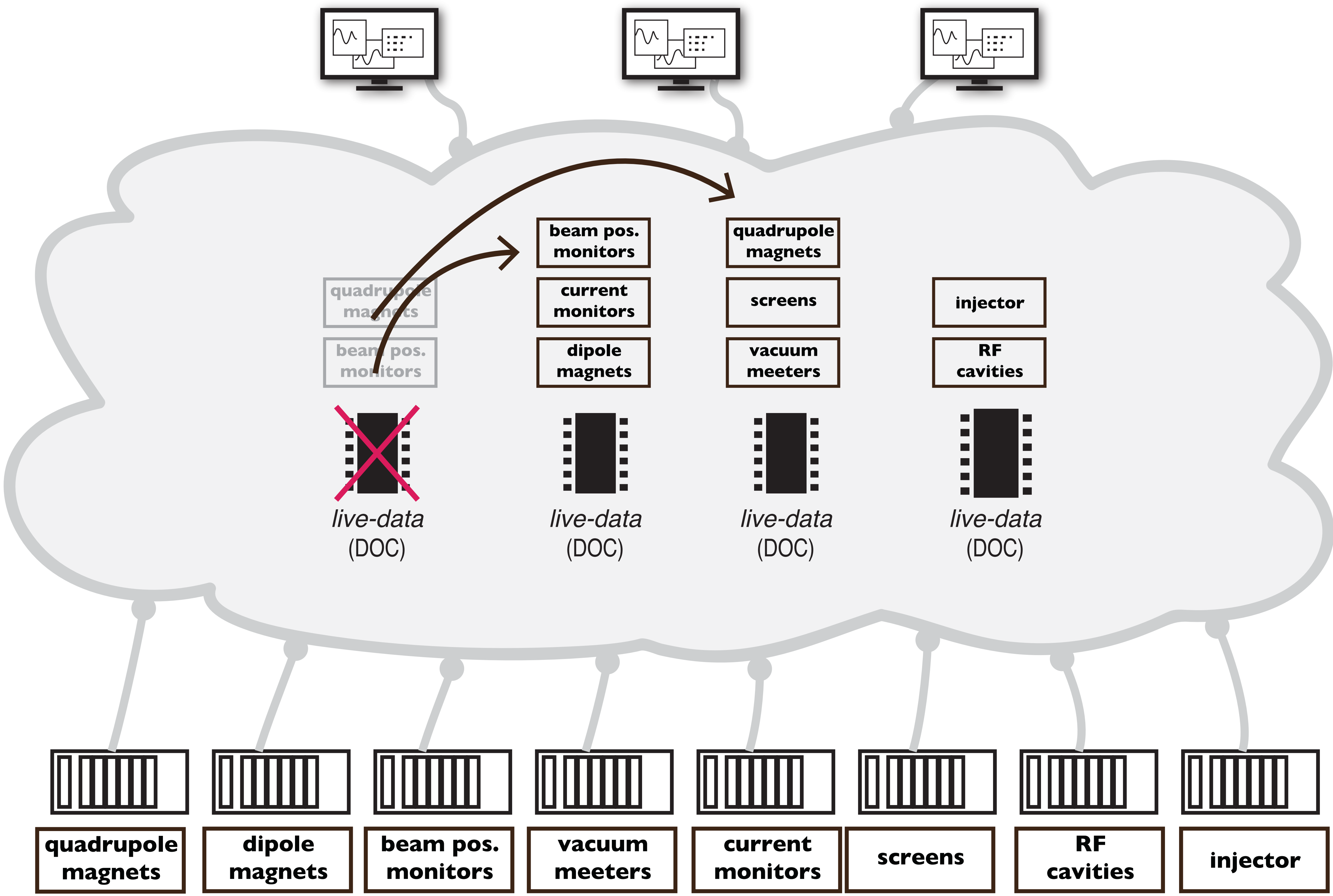
{“key1” : cstring : dataoffset
“key2” : int32 : dataoffset}

{hellox7D\x00\x00\x00}

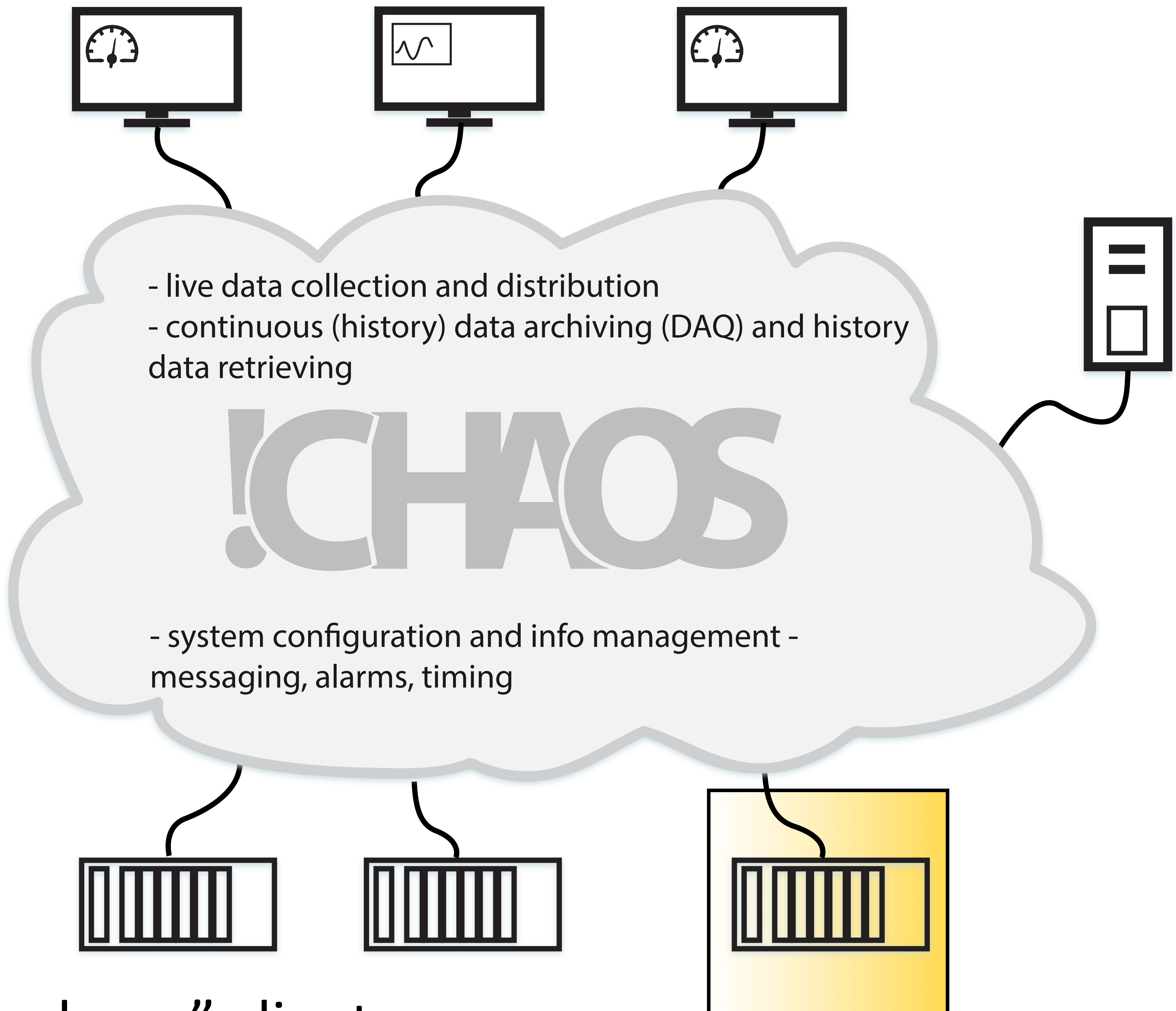
!CHAOS *live-data* service



failover strategies

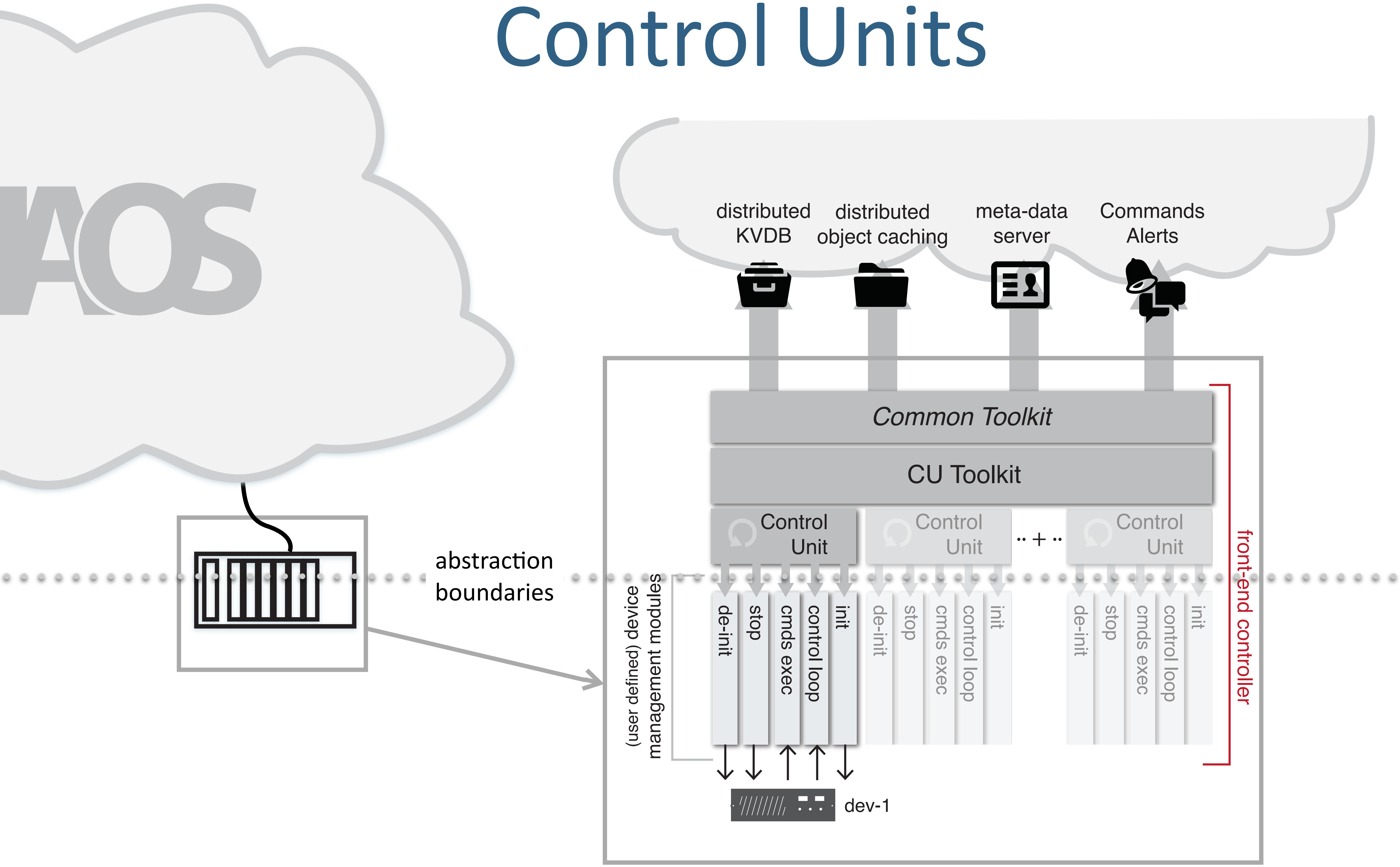


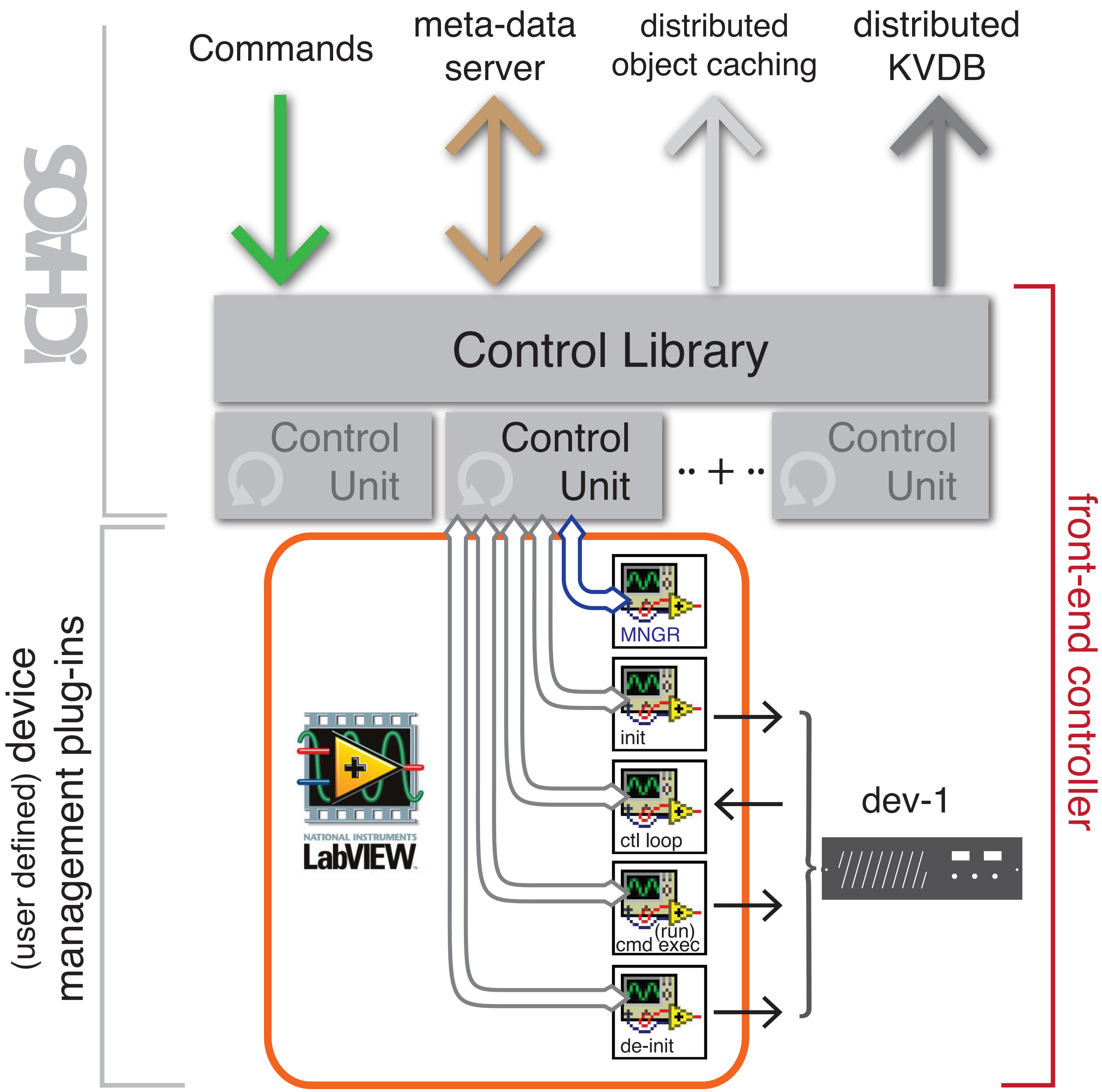
“data consumer” clients

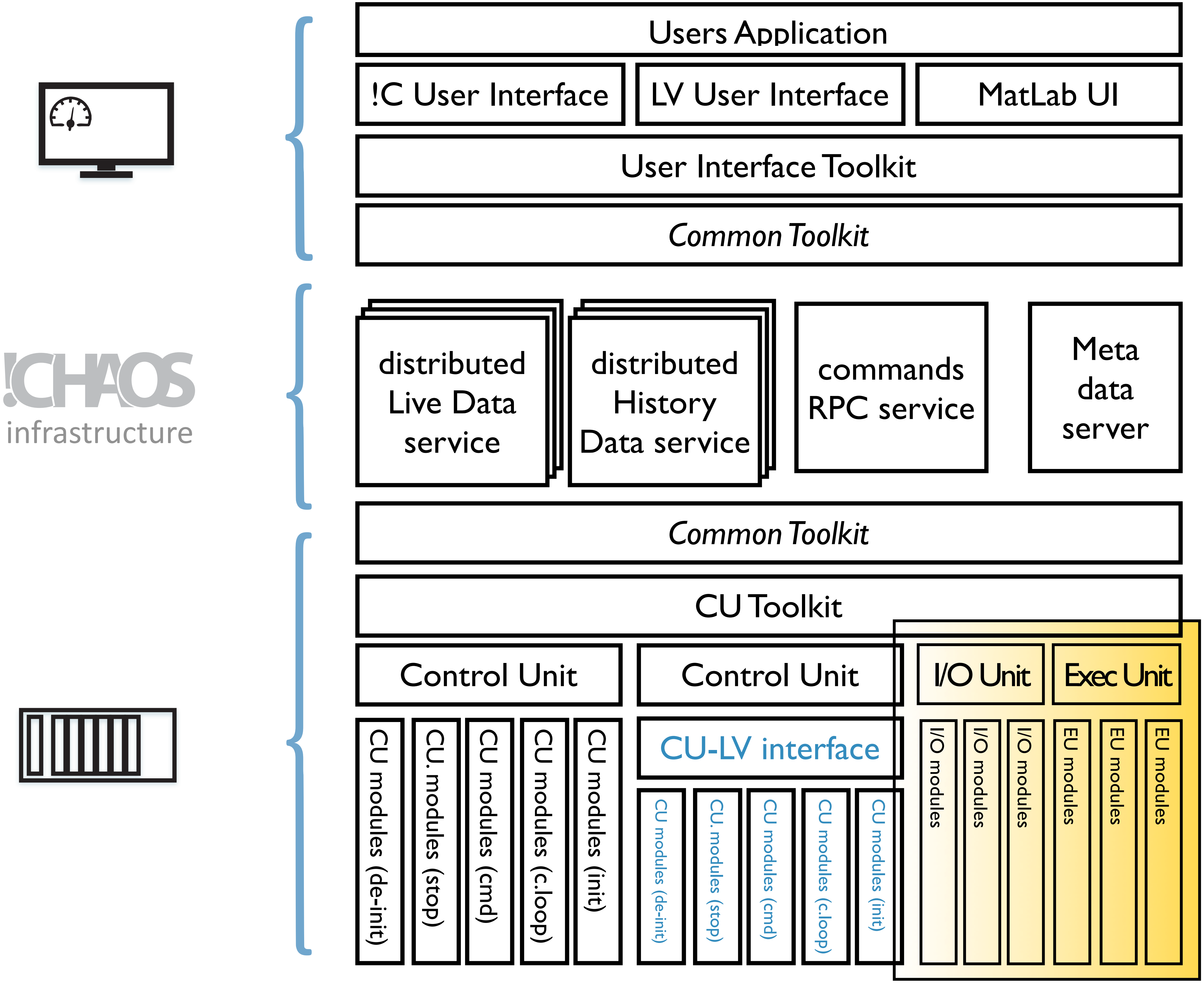


“data producer” clients

Control Units




Commands
meta-data server
distributed object caching
distributed KVDB
Control Library
Control Unit
Control Unit
Control Unit
front-end controller
(user defined) device management plug-ins
MNGR
init
ctl loop
(run) cmd exec
de-init
dev-1
NATIONAL INSTRUMENTS
LabVIEW

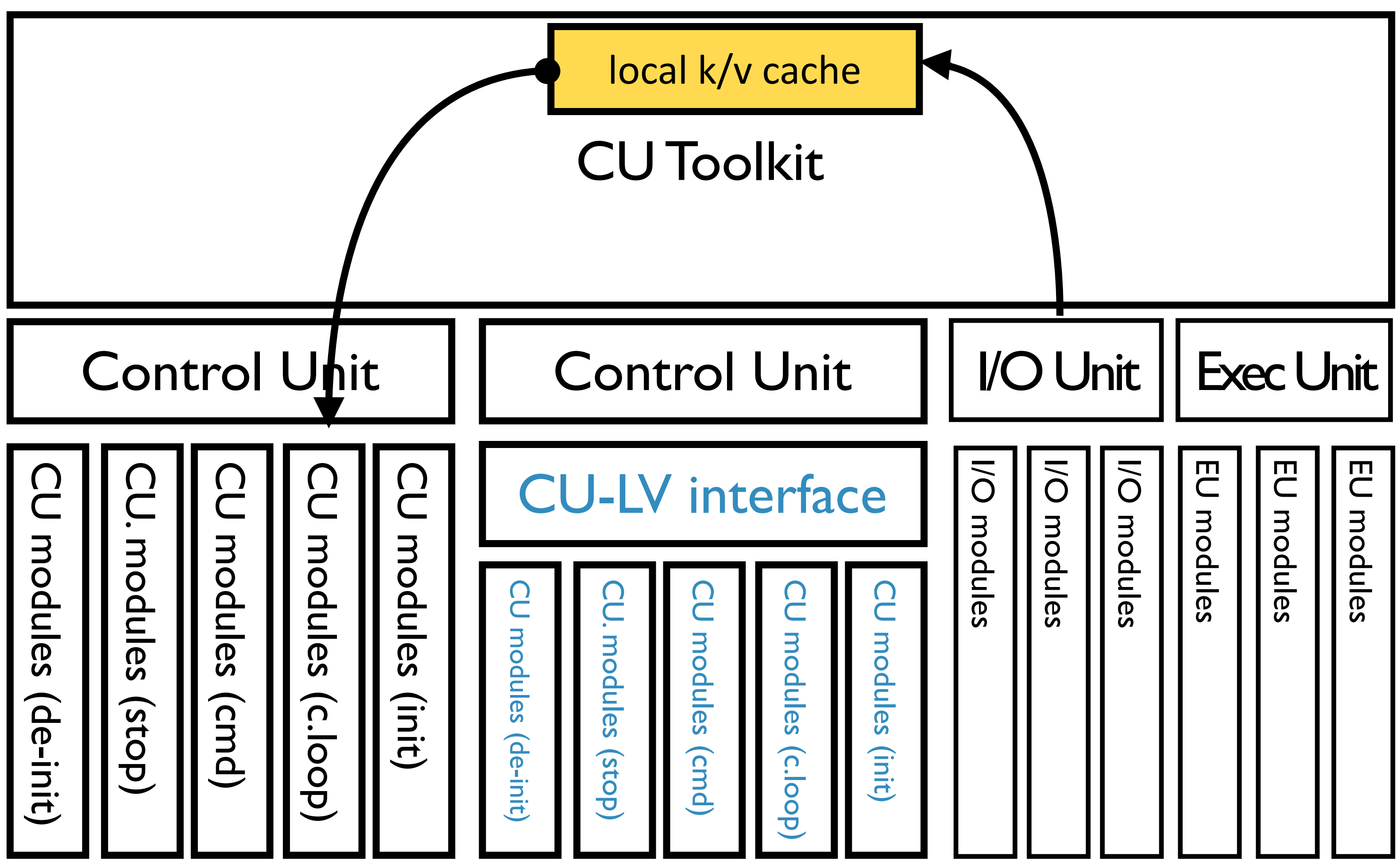

Users Application
!C User Interface
LV User Interface
MatLab UI
User Interface Toolkit
Common Toolkit
!CHAOS
infrastructure
distributed Live Data service
distributed History Data service
commands RPC service
Meta data server
Common Toolkit
CU Toolkit
Control Unit
Control Unit
I/O Unit
Exec Unit
CU modules (de-init)
CU. modules (stop)
CU modules (cmd)
CU modules (c.loop)
CU modules (init)
CU-LV interface
CU modules (de-init)
CU. modules (stop)
CU modules (cmd)
CU modules (c.loop)
CU modules (init)
I/O modules
I/O modules
I/O modules
EU modules
EU modules
EU modules

I/O Units

I/O Unit allows decoupling device logic from signal acquisition

I/O Units, programmed by hw experts, will read Input channels and store data in a local k/v cached (the same technology as for live-data)

CU programmers will get data from local k/v cache instead of I/O modules directly



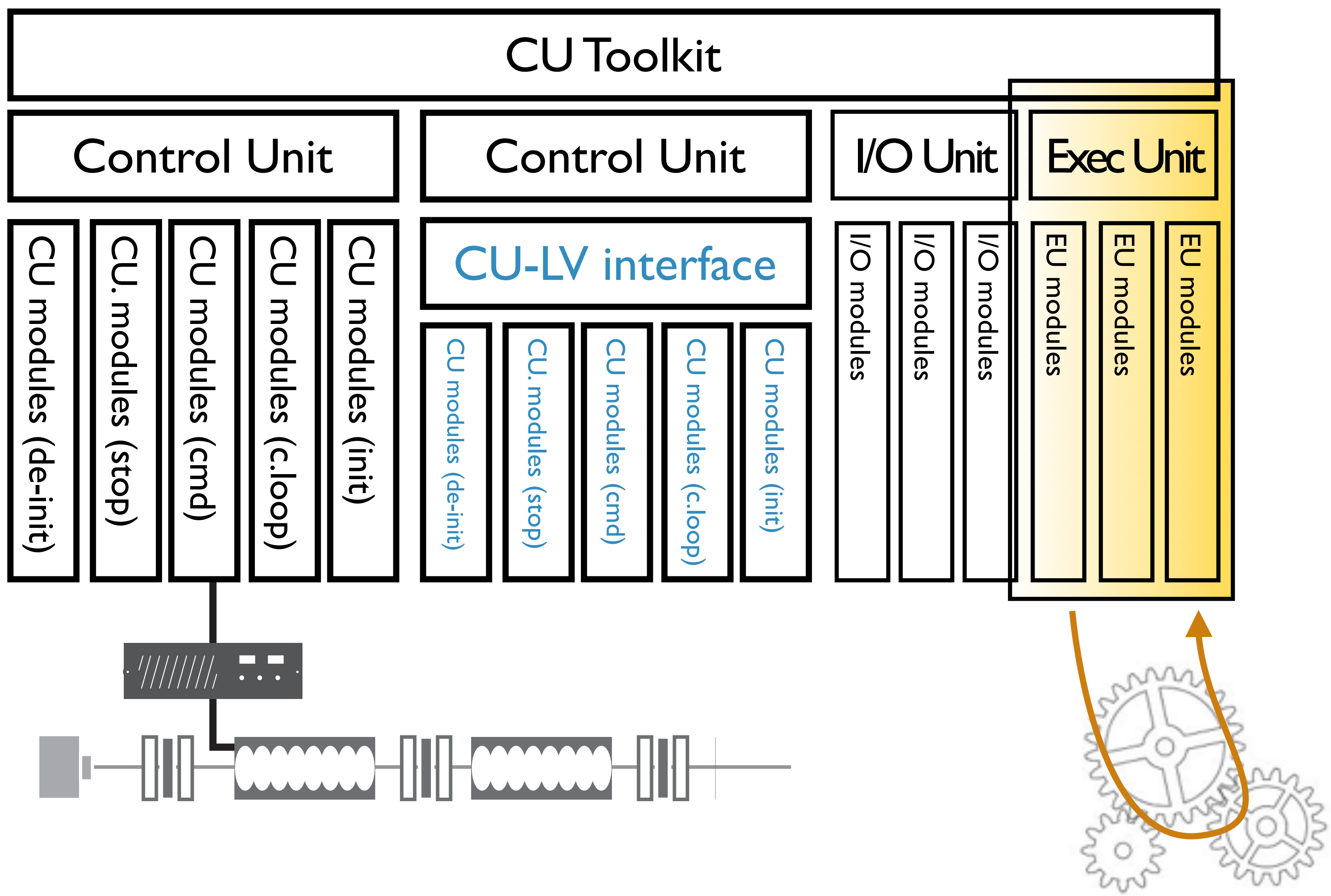
Execution Units

Execution Unit extend the concept of Control Unit to computational tasks

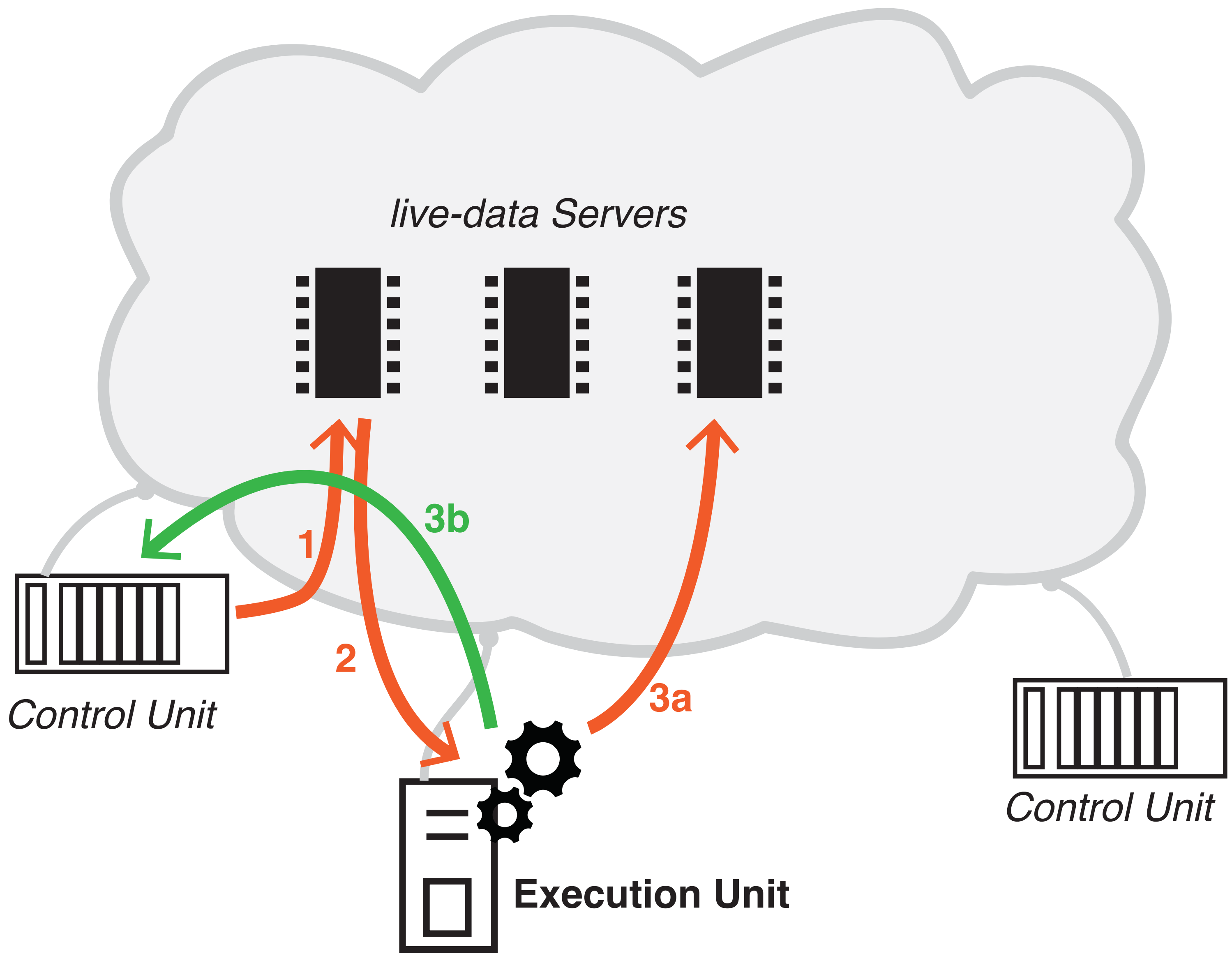
EU are not connected to any device, instead they produce data as result of a logic or calculations.

They produce an output data class as consequence of the value defined by the input data class.

EU wil be used to implement analysis, measurement procedures, feedbacks



Execution Units

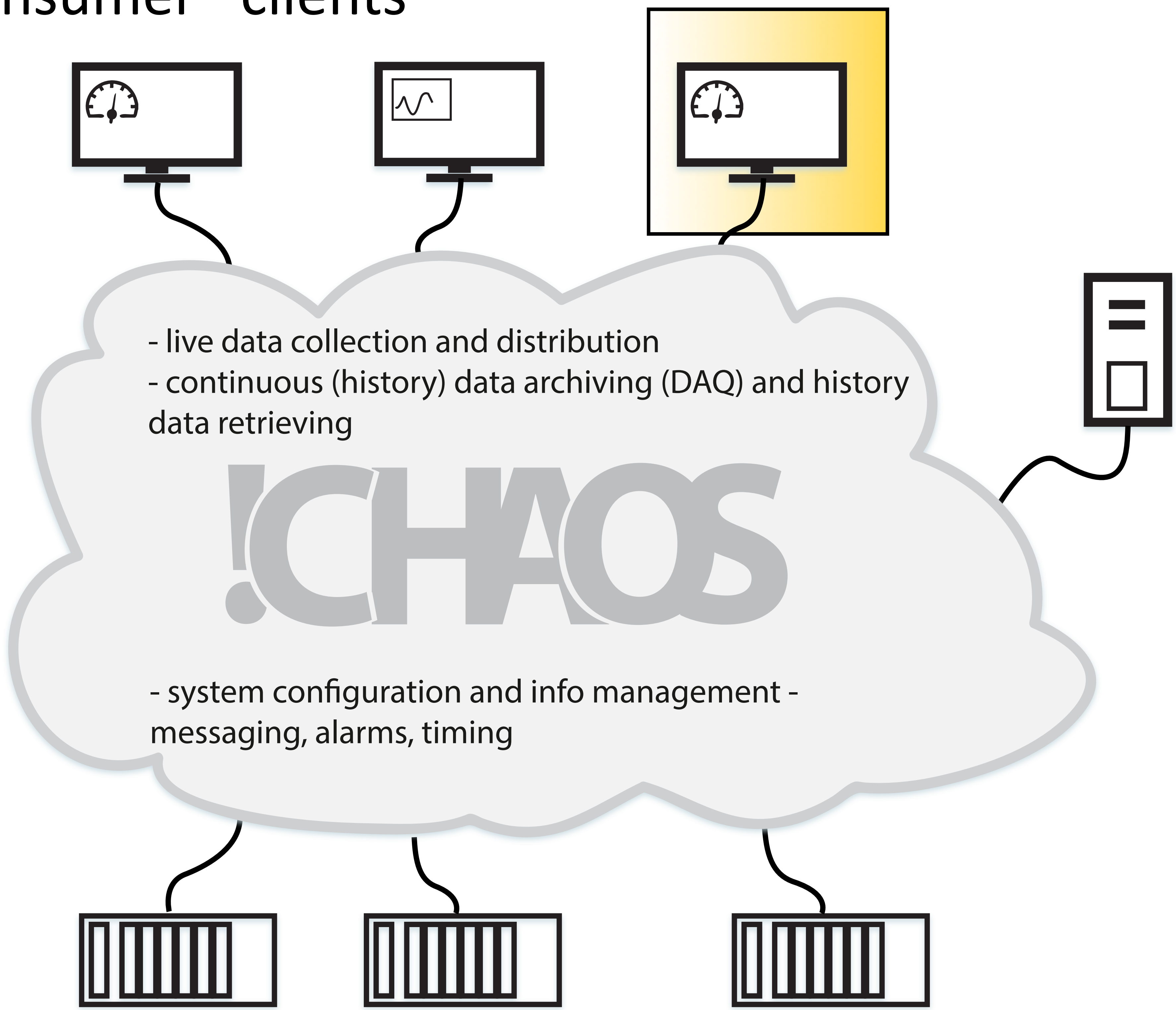


1 dataset containing Input data is written as k/v into the live-data cluster

2 Execution Unit reads the dataset and performs calculations

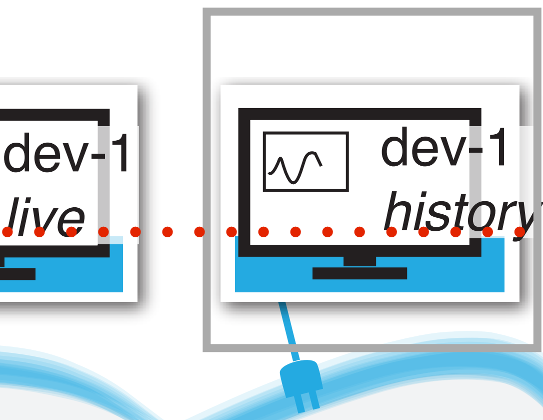
3a results is provided to the rest of the world as a k/v in the live-data cluster

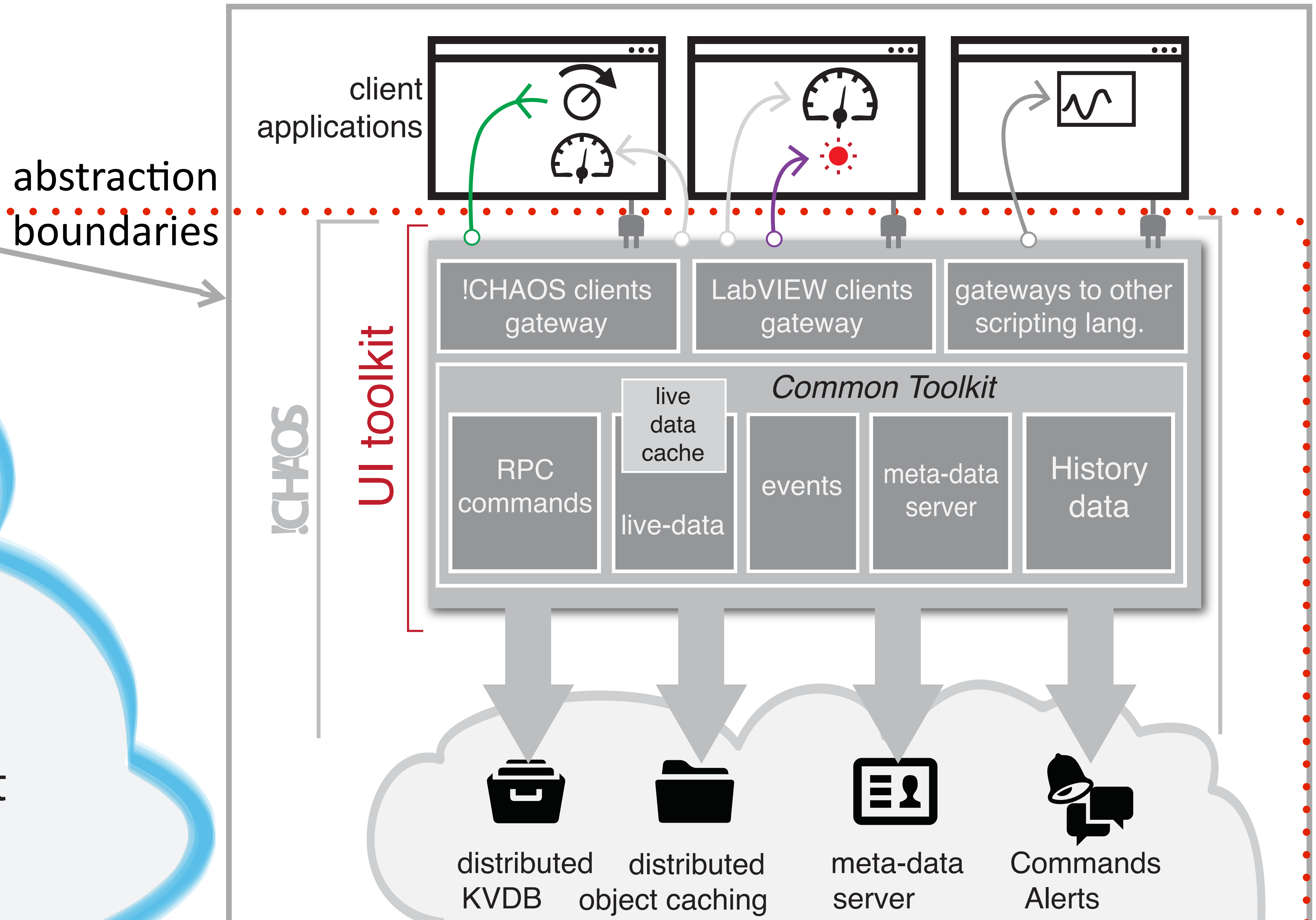
3b if, for instance, the Exec. Unit is running a feedback then the results is also sent as commend to the Control Unit in charge for the device “data consumer” clients



“data producer” clients

client abstraction

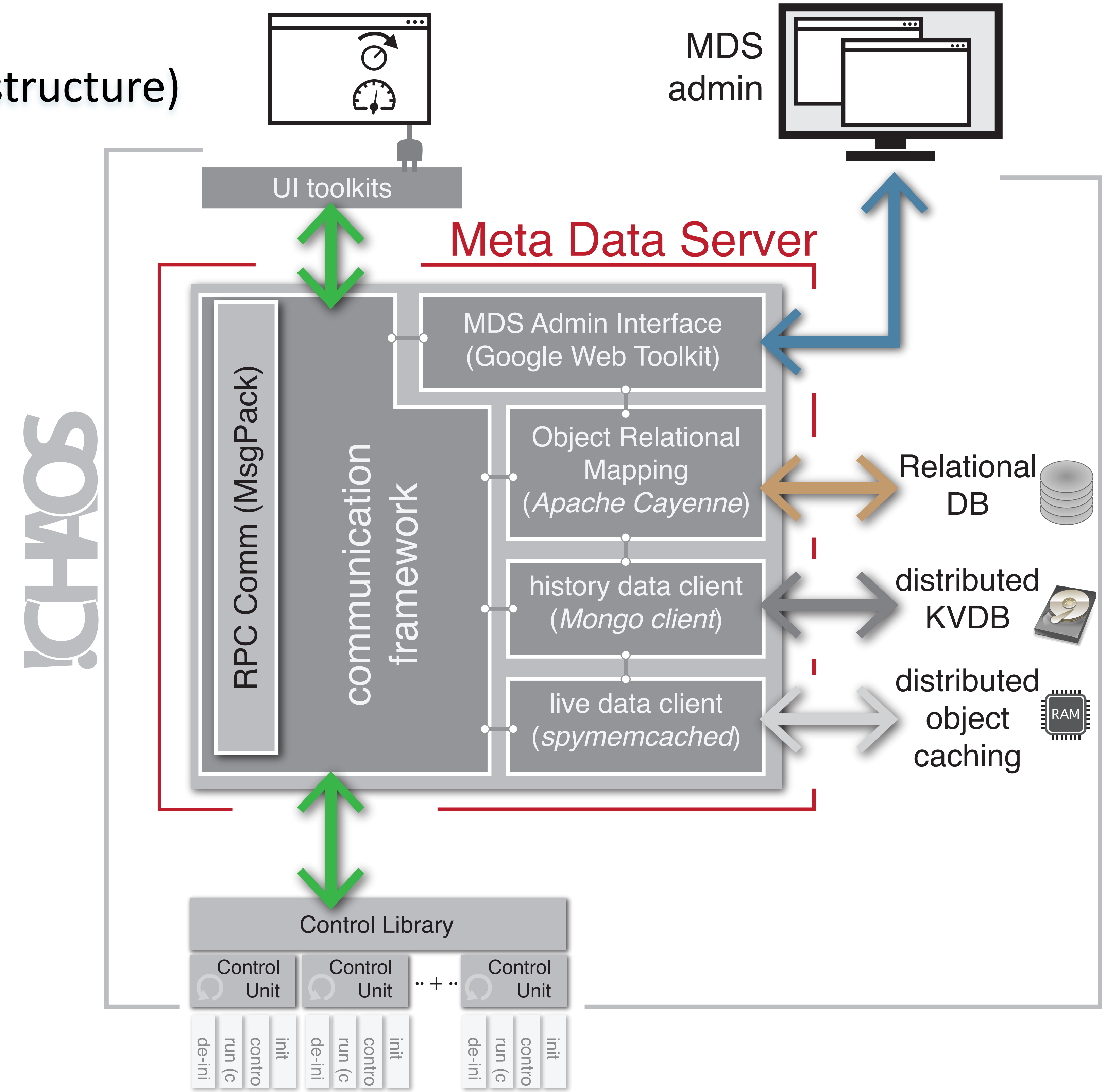




ion and distribution
tory) data archiving (DAQ)
retrieving
ation and info management
ms, timing

Meta-data Server

- CU configuration manager
  (e.g. managing of pushing data rate)
- Semantic of data (e.g. dataset's structure)
- Command's list and semantic
- Naming service



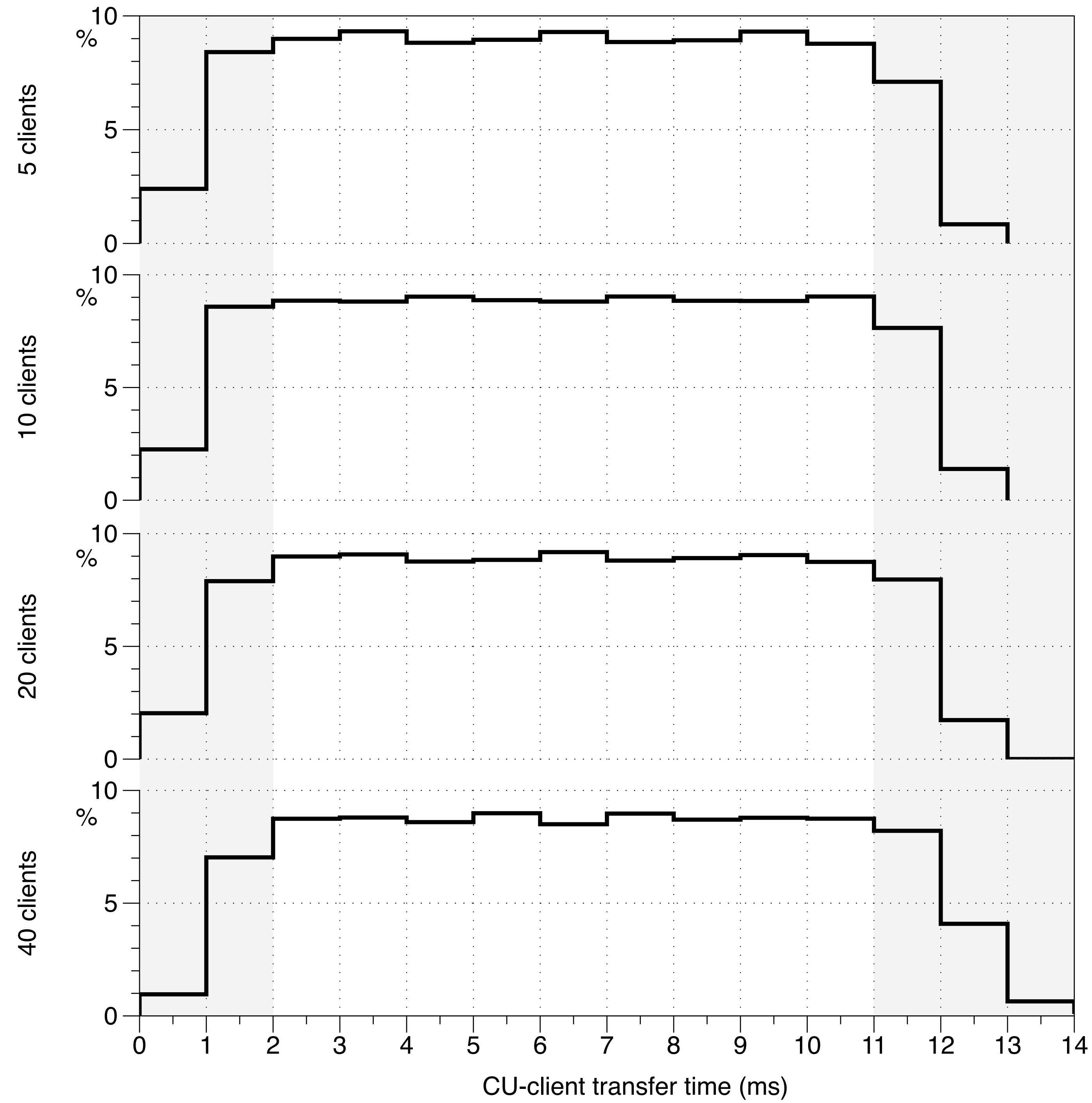
multi-client test

Measured transfer time between front-end CU and a client application via DOC for a different number of concurrent clients reading the same key/value being continuously updated by the CU.

-100 Hz data refresh rate

-1 kByte data

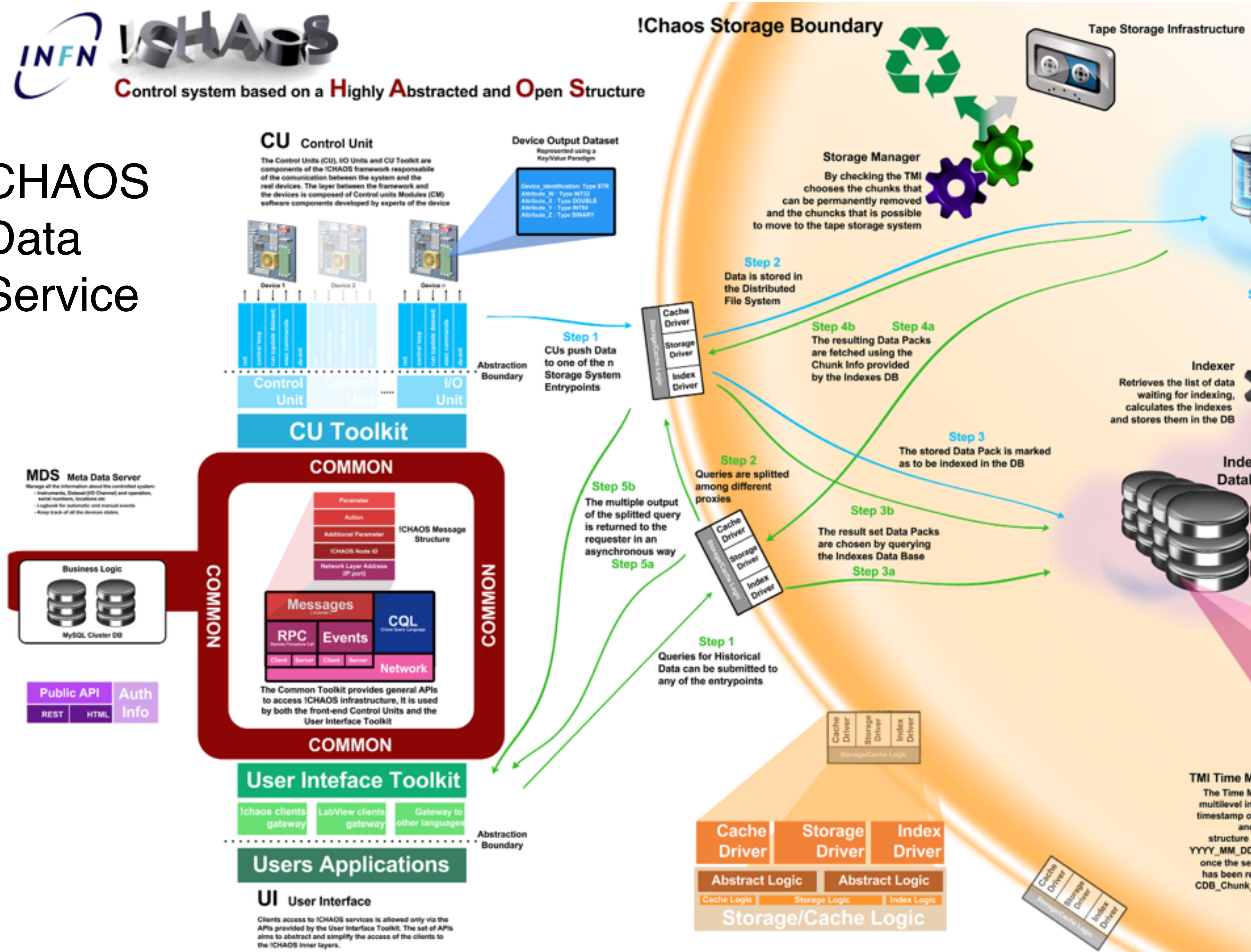
-5 to 40 concurrent clients



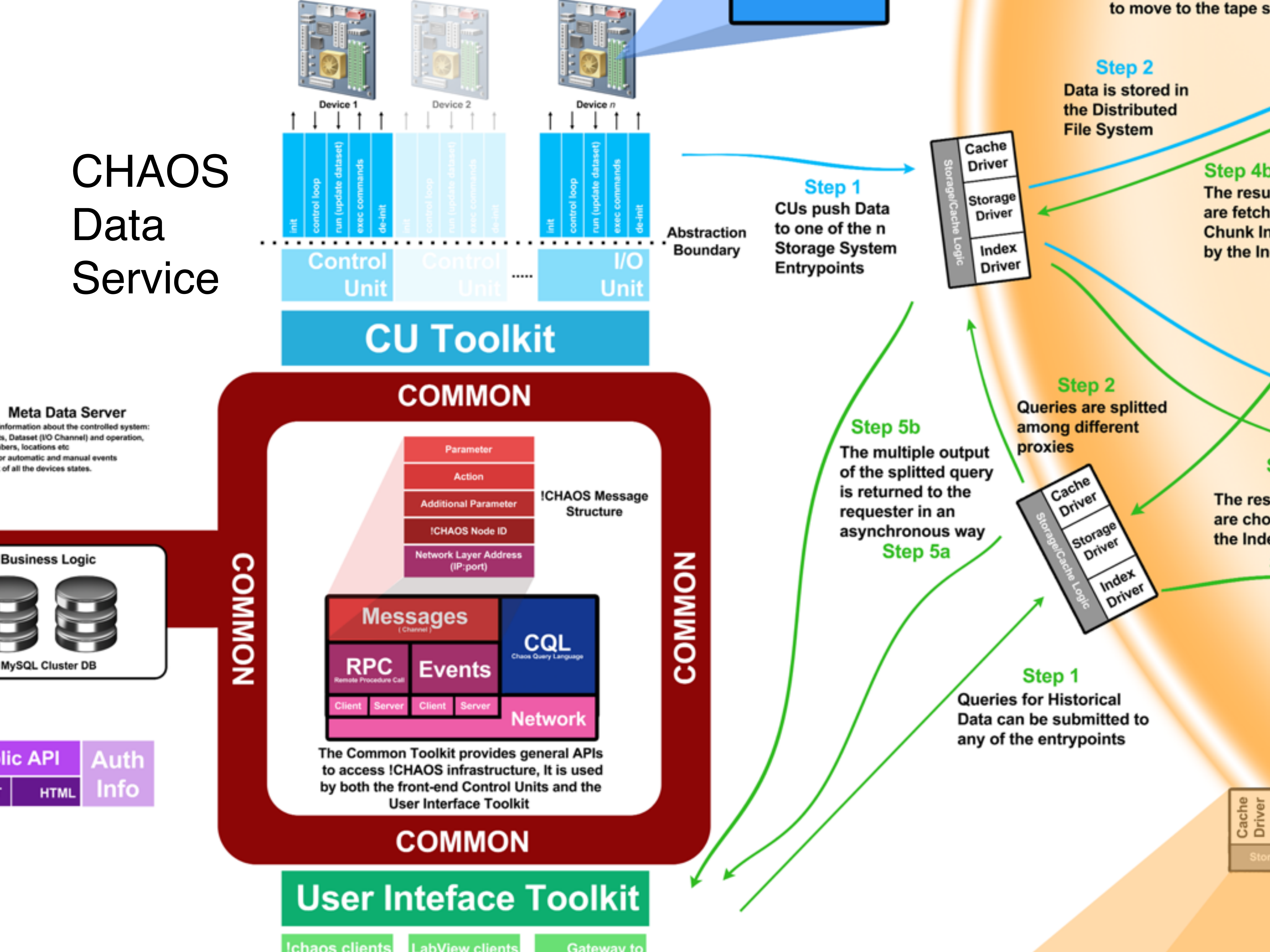
new developments

- ✓ CDC: !CHAOS Data Service
- ✓ BTF installation
- ✓ non-HEP application

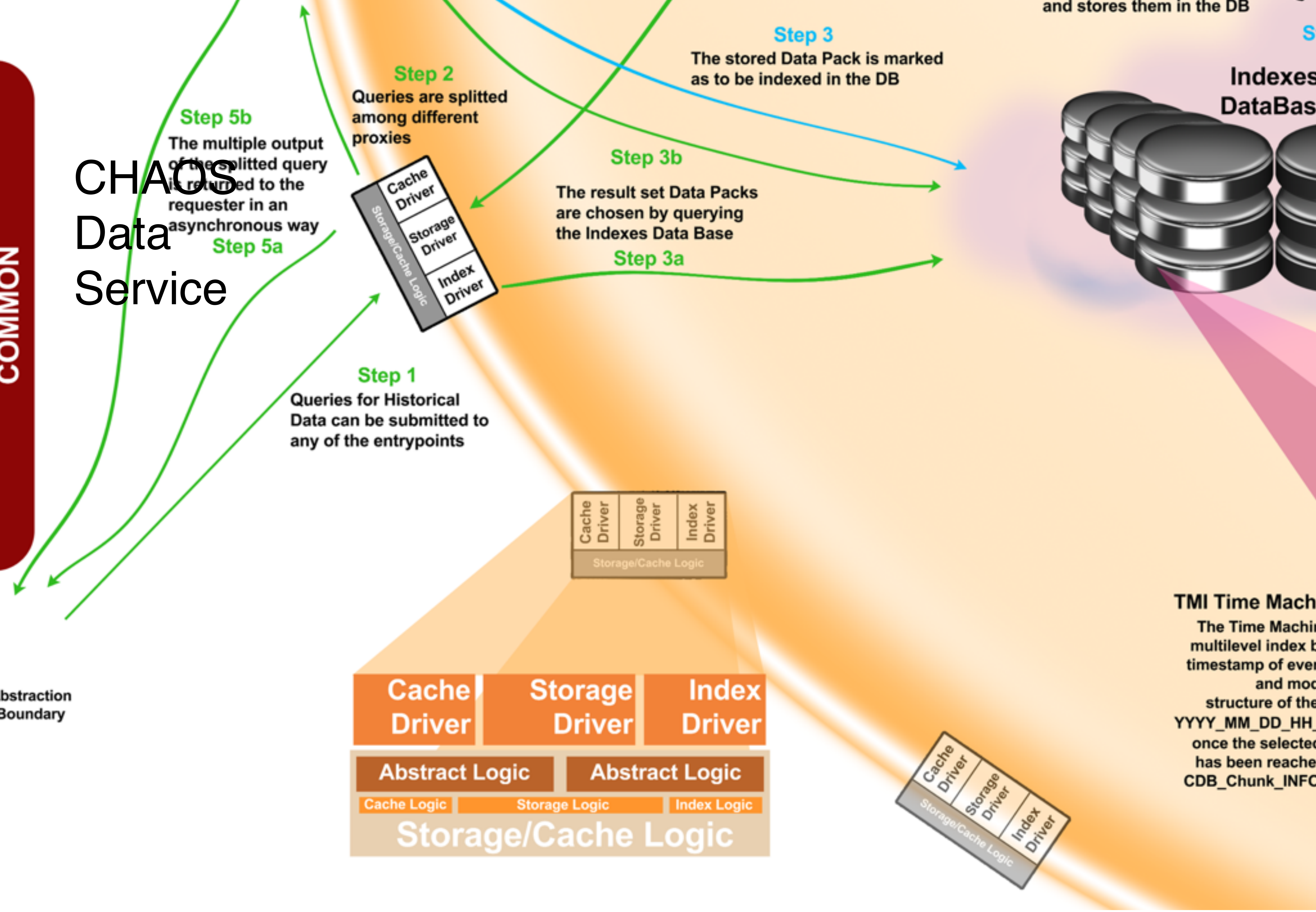
CHAOS Data Service

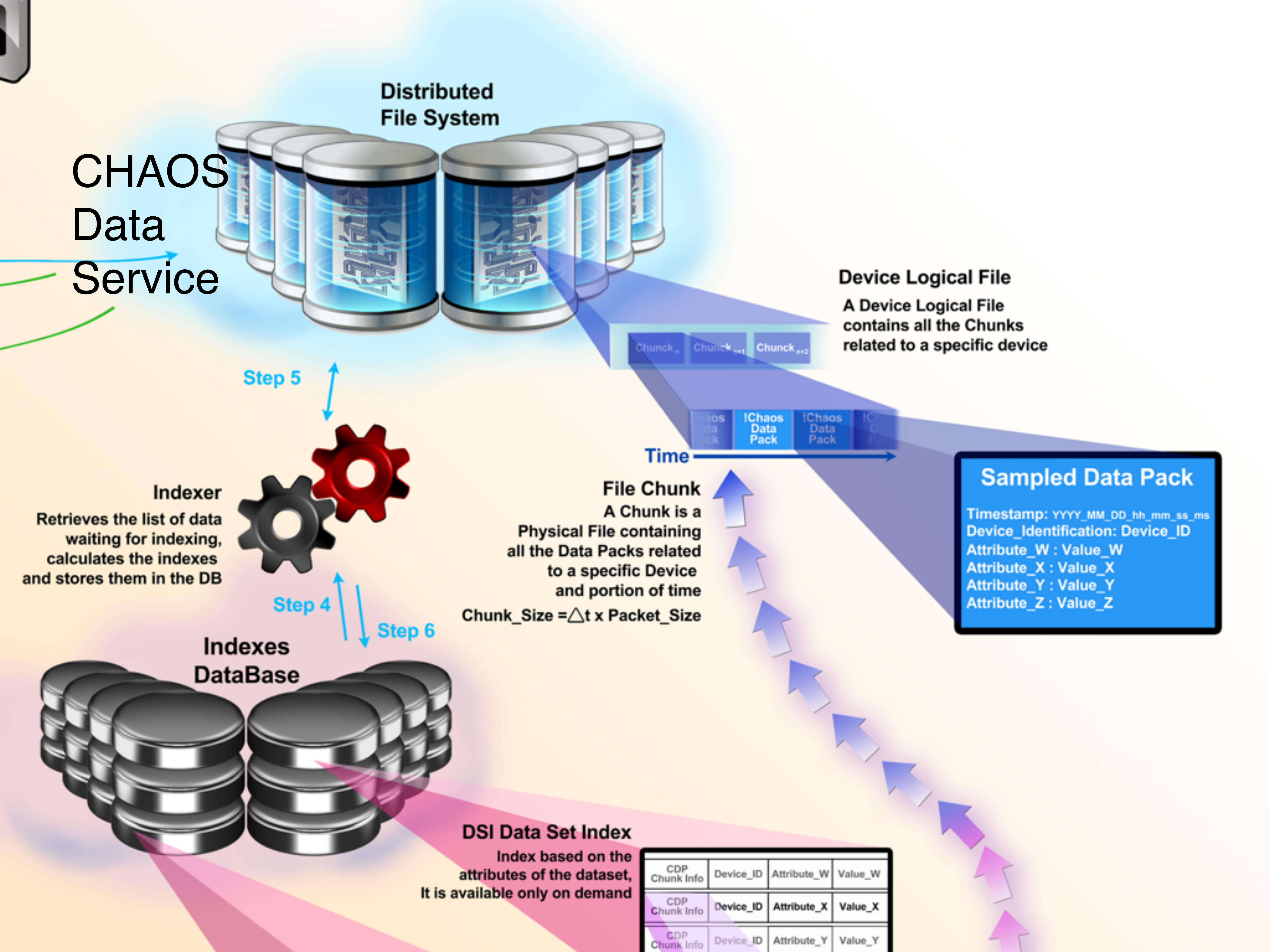


CHAOS Data Service



CHAOS Data Service

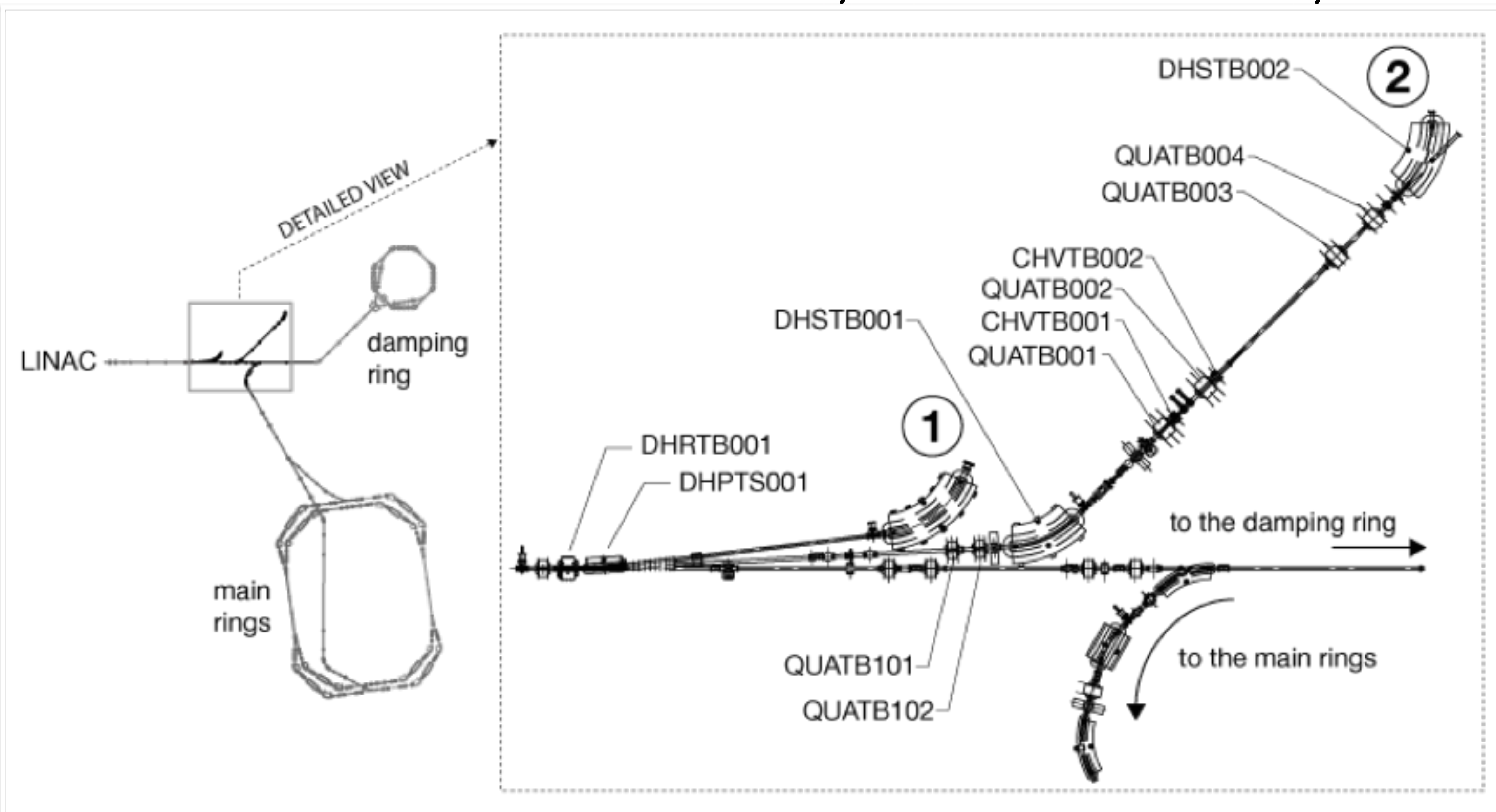



CHAOS Data Service
Distributed File System
Device Logical File
A Device Logical File contains all the Chunks related to a specific device
Chunck n
Chunck n+1
Chunck n+2
!Chaos Data Pack
!Chaos Data Pack
Time
Sampled Data Pack
Timestamp: YYYY_MM_DD_hh_mm_ss_ms
Device_Identification: Device_ID
Attribute_W : Value_W
Attribute_X : Value_X
Attribute_Y : Value_Y
Attribute_Z : Value_Z
Step 5
Indexer
Retrieves the list of data waiting for indexing, calculates the indexes and stores them in the DB
File Chunk
A Chunk is a Physical File containing all the Data Packs related to a specific Device and portion of time
Chunk_Size =△t x Packet_Size
Step 4
Step 6
Indexes DataBase
DSI Data Set Index
Index based on the attributes of the dataset, It is available only on demand
CDP Chunk Info
Device_ID
Attribute_W
Value_W
CDP Chunk Info
Device_ID
Attribute_X
Value_X
CDP Chunk Info
Device_ID
Attribute_Y
Value_Y

!CHAOS BTF installation

Dafne Beam Test Facility is designed for experiments (e.g. detectors' R&D and calibration) with the electron beam of the injector linac.

The BTF beamline is controlled by a dedicated control system.

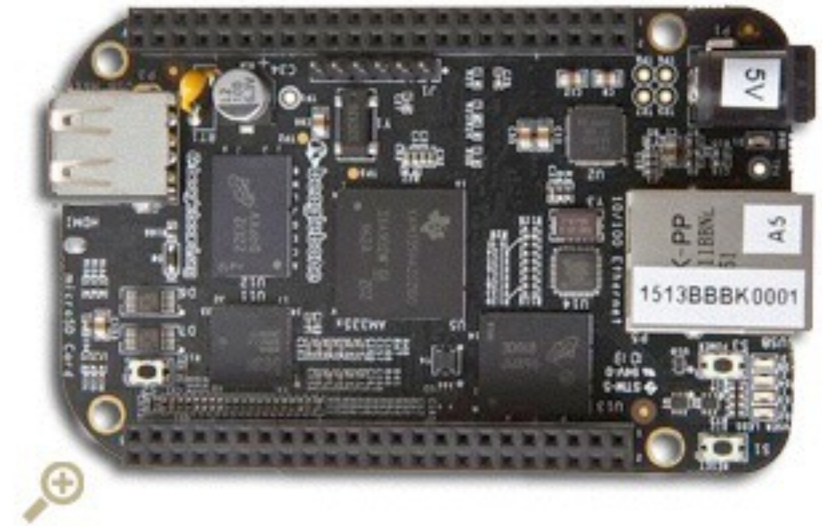


BB Control Unit

!CHAOS CU on Beagle Bone

+

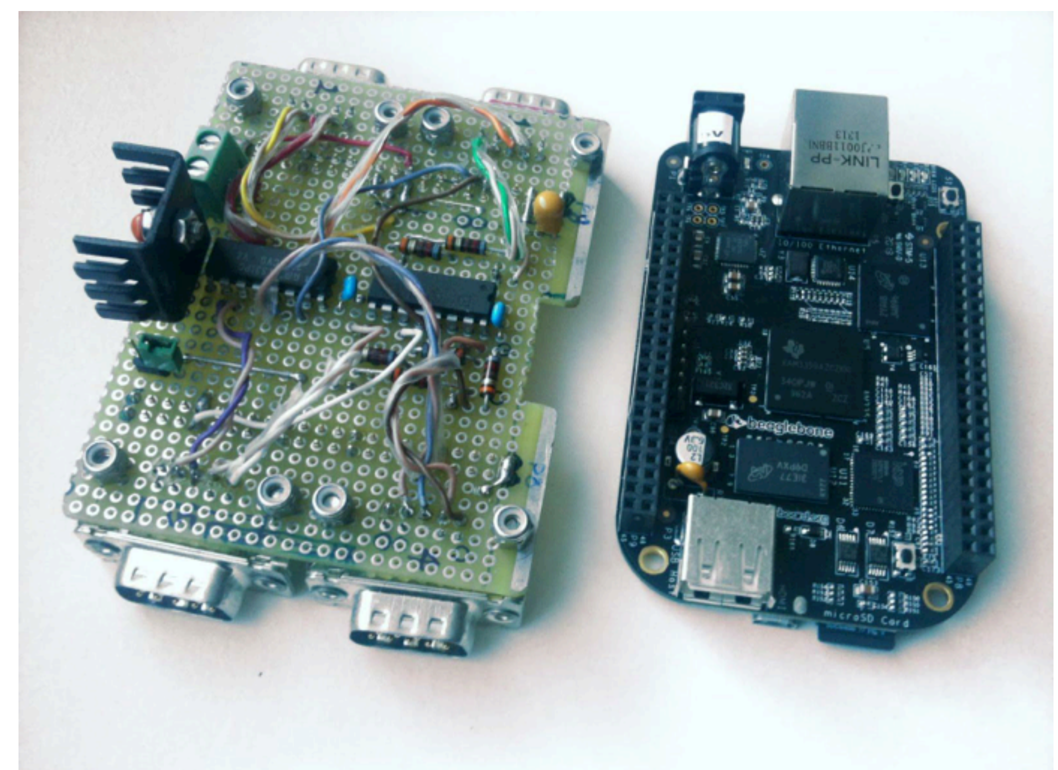
custom CAPE to drive OCEM power supplies

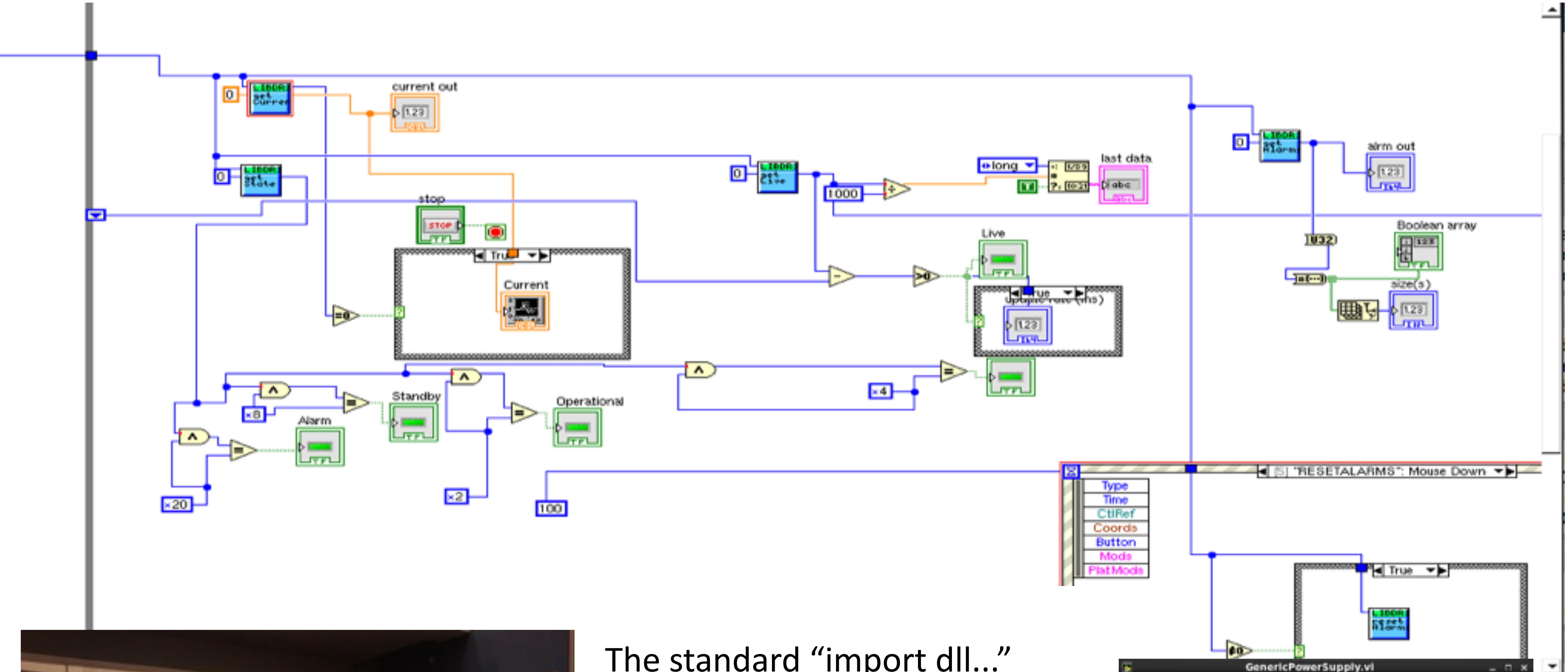


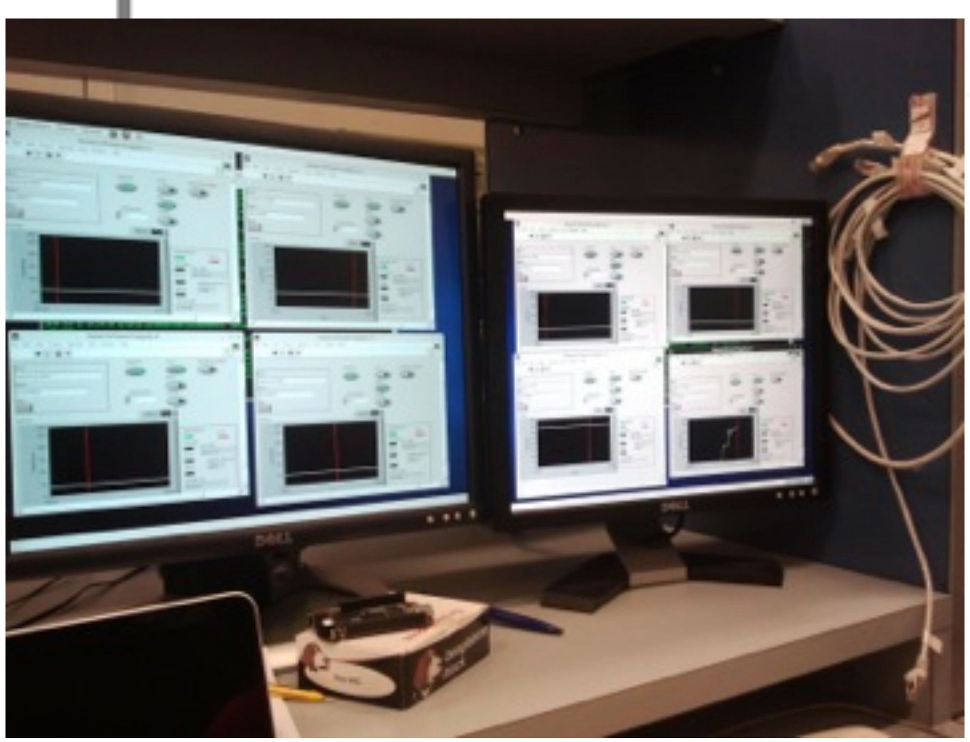
We realised a custom cape with 4x485 serial interfaces to drive point to point 4 power supplies.

The original configuration was based on a single multi-drop serial 485 line connected with MOXA.

1. dramatically cut the cost of the HW
2. reduced latency by a factor 4
3. Demonstrated that !CHAOS can be easily retargeted to run on Linux based cheap embedded platforms.





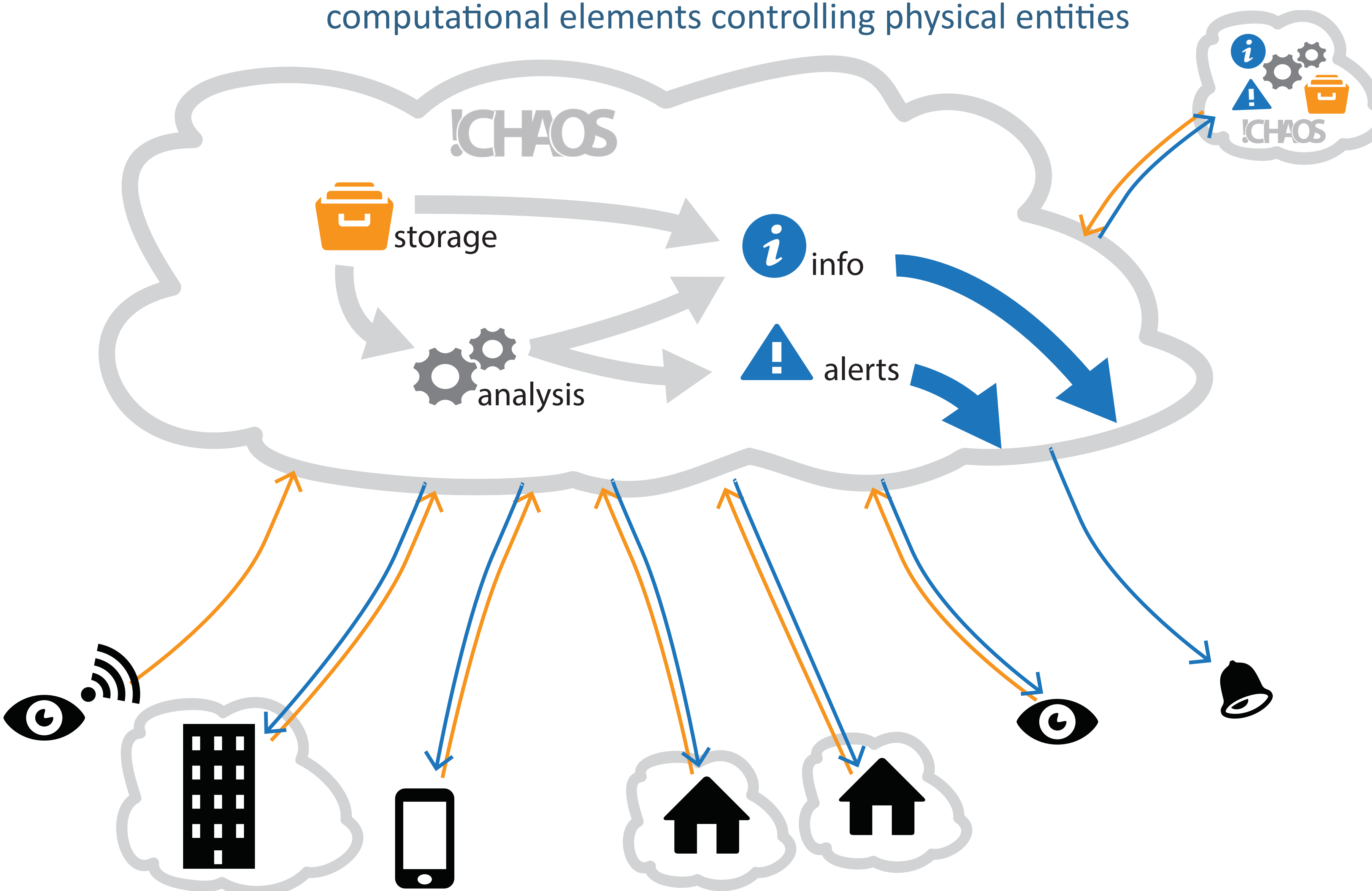


The standard “import dll...” LabVIEW flow has been used to create VI APIs (blue boxes) from !CHAOS C/C++ UI APIs.

Through these VIs we can access !CHAOS resources and quickly create a LabVIEW interface from them.

!CHAOS interdisciplinary application

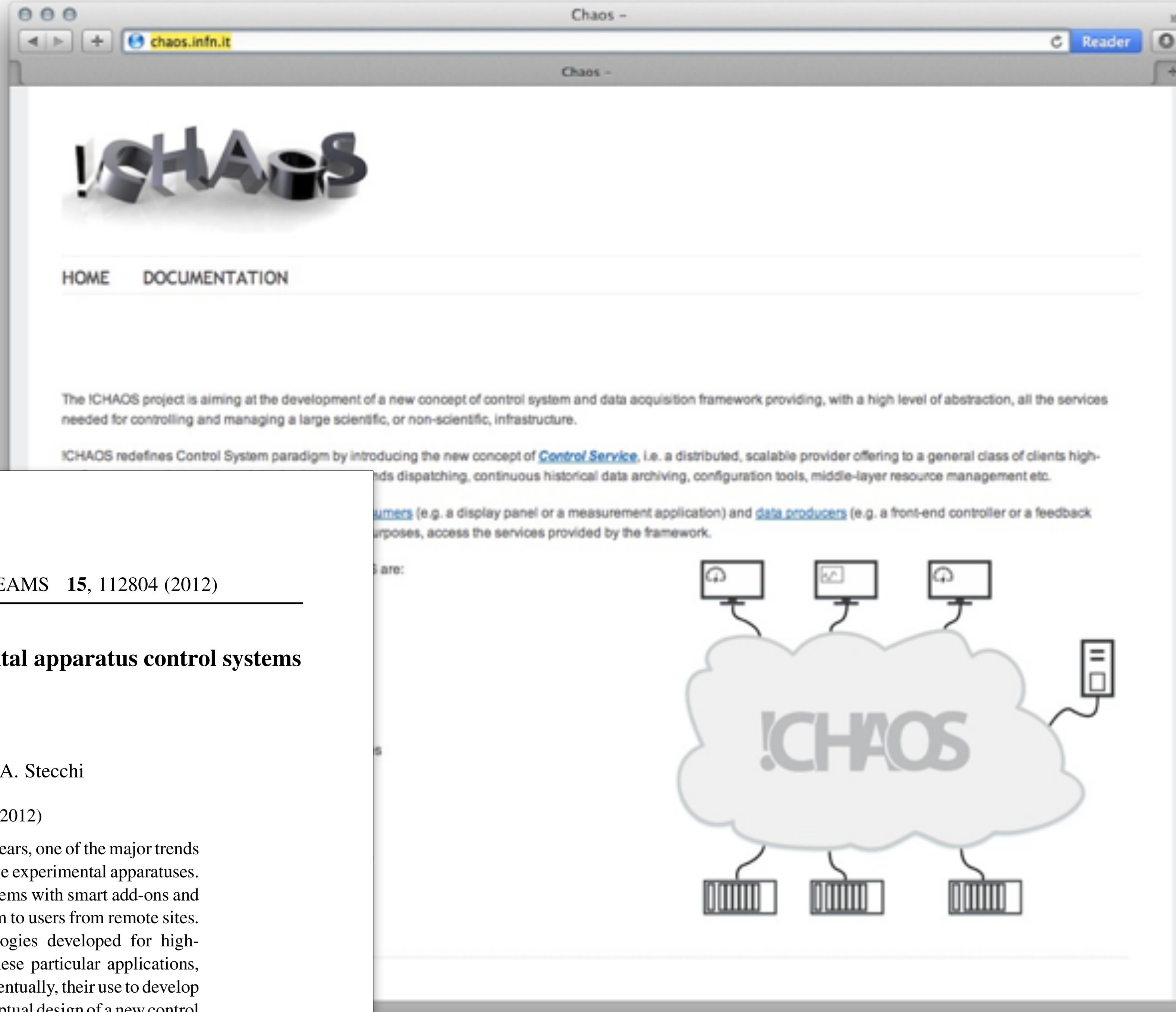
acquisition&control infrastructure of sensors, appliances, collaborating computational elements controlling physical entities



Conclusions

- !CHAOS design completed
- main services completed…
- …but still exploring new solutions
- all-!CHAOS control system is operational
- compatibility with many OS and HW
- non-HEP application under development

more Info



PHYSICAL REVIEW SPECIAL TOPICS - ACCELERATORS AND BEAMS **15**, 112804 (2012)

Introducing a new paradigm for accelerators and large experimental apparatus control systems

L. Catani and F. Zani
INFN-Roma Tor Vergata, Roma, Italy

C. Bisegni, G. Di Pirro, L. Foggetta, G. Mazzitelli, and A. Stecchi
INFN-LNF, Frascati, Italy
(Received 19 December 2011; published 29 November 2012)

The integration of web technologies and web services has been, in the recent years, one of the major trends in upgrading and developing distributed control systems for accelerators and large experimental apparatuses. Usually, web technologies have been introduced to complement the control systems with smart add-ons and user friendly services or, for instance, to safely allow access to the control system to users from remote sites. Despite this still narrow spectrum of employment, some software technologies developed for high-performance web services, although originally intended and optimized for these particular applications, deserve some features suggesting a deeper integration in a control system and, eventually, their use to develop some of the control system's core components. In this paper, we present the conceptual design of a new control system for a particle accelerator and associated machine data acquisition system, based on a synergic combination of a nonrelational key/value database and network distributed object caching. The use of these technologies, to implement respectively continuous data archiving and data distribution between components, brought about the definition of a new control system concept offering a number of interesting features such as a high level of abstraction of services and components and their integration in a framework that can be seen as a comprehensive service provider that both graphical user interface applications and front-end controllers join for accessing and, to some extent, expanding its functionalities.

DOI: 10.1103/PhysRevSTAB.15.112804 PACS numbers: 07.05.Dz, 07.05.Bx, 07.05.Hd, 29.20.−c

I. INTRODUCTION

Two main motivations support the decision to start investigating a new approach in the design and development of distributed control systems (DCS) for particle accelerators.

New developments in this field, similar to what has happened in recent years, will be basically directed towards of new accelerator DCS may profit from solutions borrowed from cutting-edge Internet services. To fully profit from these new technologies the DCS model has to be reconsidered, thus leading to the definition of a new paradigm.

The second strong motivation for this development follows the recent approval, by the Italian Ministry for

Thank you