Control system based on a
Highly
Abstracted and
Open
Structure

WORKSHOP CCR 2013

http://chaos.infn.it

C. Bisegni

Objective

One ring to rule them all

Objective

What we want

- a design that permits to scale to achieve high speed in the data acquisition and elaboration and for fault tolerant;
- possibility to attach and detach an entire subsystem;
- possibility to “configure” a specified instrument without halt the full system;
- possibility to hot insert an instrument or a subsystem;

Objective

What we want

- give the flexibility to implements control algorithm on abstracted instrument class;
- work on data abstraction, without fixed schema;
- give a data system that permit to monitor the instrument channels and query the history instruments data;
- the data can be managed into domains
- every domain can be implemented with different technology

Objective

Objective


ALICE
CMS
ATLAS
LHCb

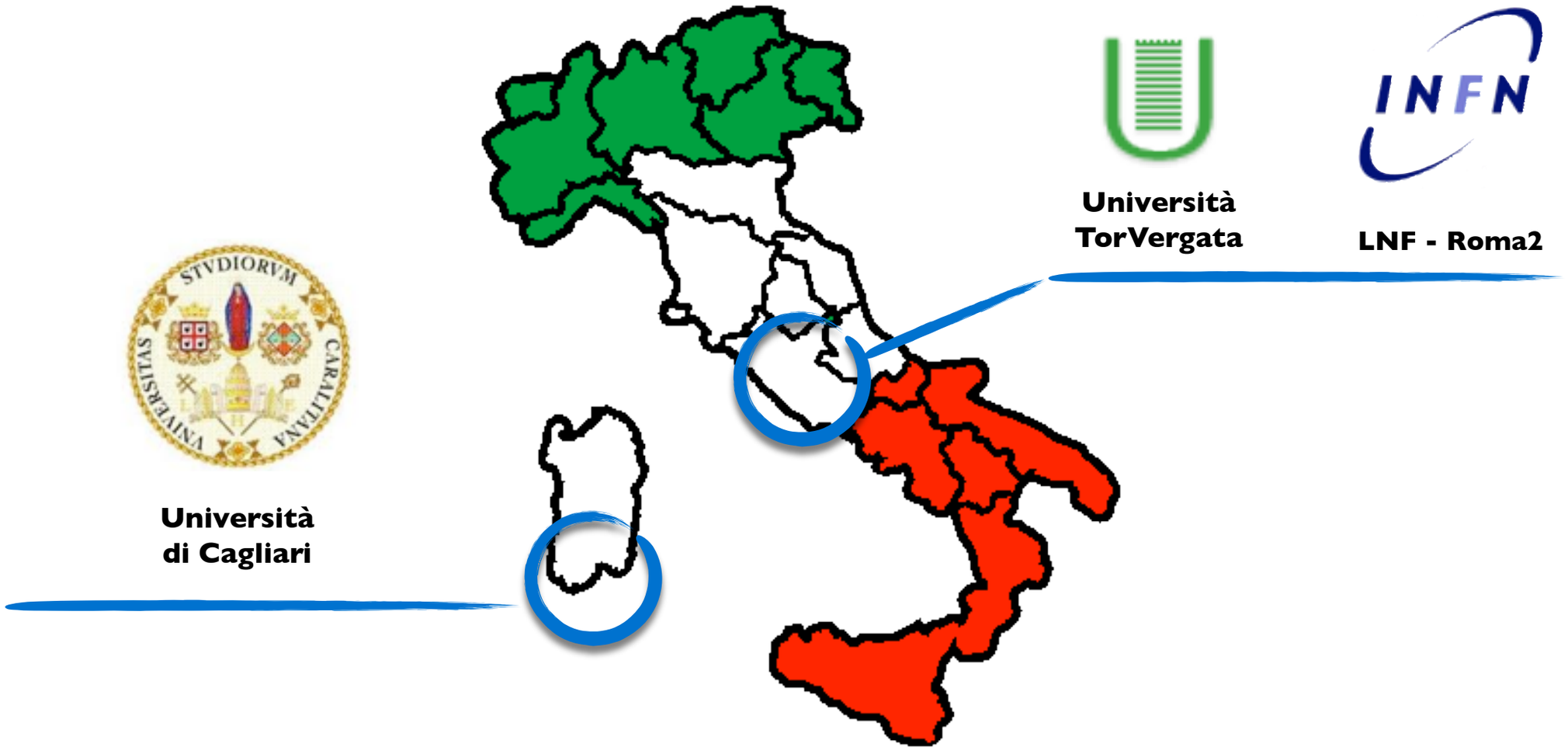

!CHAOS


U.S.S. VOYAGER
NCC-74656
KEY LOCATIONS CUTAWAY

What is

!CHAOS (**C**ontrol system based on **H**ighly **A**bstracted and **O**pen **S**tructure), is an experiment of CSN 5(technological research experiments) at the **LNF** and **Roma-TV** section.

The project has been appreciated by educational world and the University of Tor Vergata and Cagliari have joined the development team.



What is

Premiale INFN "!CHAOS: A Cloud of Controls"

INFN-LNF (Laboratori Nazionali di Frascati)
INFN-TV (Sezione di Tor Vergata)
INFN-PG (Sezione di Perugia)
INFN-CNAF (Centro Nazionale Tecnolgie Informatiche)
INFN-PD (Padova)
INFN-LNS (Laboratori Nazionali di Catania)

National Instruments (NI)

ADF Solaris

Framework and Services

A CPP Framework and services that permit to have:

- **90%** of adaptation time of instrument to !CHOAS done by framework
- simplify the creation of the user interfaces
- simplify the creation of the control algorithms basing it on an abstraction of the I/O channels or full dataset not on physical device
- All service for data management and monitoring already done and fully customizable

!CHAOS Simple Vision

Simple Vision

!CHOAS Common Framework

Simple Vision

!CHOAS CU Framework

!CHOAS Common Framework

Simple Vision

Device Driver
(Control Unit)

!CHOAS CU Framework

!CHOAS Common Framework

Simple Vision

Device Driver
(Control Unit)

!CHOAS CU Framework

!CHOAS EU Framework

!CHOAS Common Framework

Simple Vision

Device Driver
(Control Unit)

Algorithm

!CHOAS CU Framework

!CHOAS EU Framework

!CHOAS Common Framework

Simple Vision

Device Driver
(Control Unit)

Algorithm

!CHOAS CU Framework

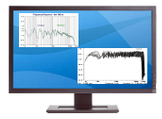
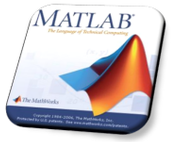
!CHOAS EU Framework

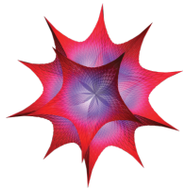
!CHOAS UI Framework

!CHOAS Common Framework

Simple Vision



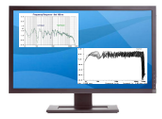
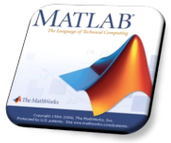



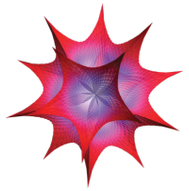
Device Driver (Control Unit)	Algorithm	UI Application
!CHOAS CU Framework	!CHOAS EU Framework	!CHOAS UI Framework

!CHOAS Common Framework

Simple Vision





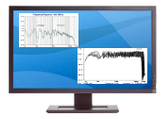
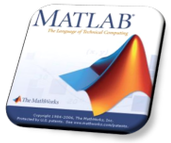
Device Driver (Control Unit)	Algorithm	UI Application
!CHOAS CU Framework	!CHOAS EU Framework	!CHOAS UI Framework

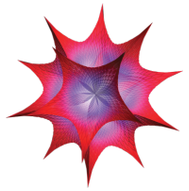
!CHOAS Common Framework

!CHOAS MetaData Services

Simple Vision





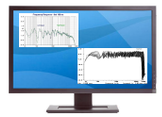
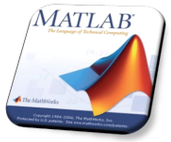
Device Driver (Control Unit)	Algorithm	UI Application
!CHOAS CU Framework	!CHOAS EU Framework	!CHOAS UI Framework

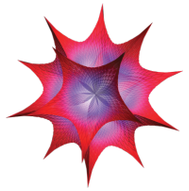
!CHOAS Common Framework

!CHOAS MetaData Services	!CHOAS Data Services

Simple Vision





Device Driver (Control Unit)	Algorithm	UI Application
!CHOAS CU Framework	!CHOAS EU Framework	!CHOAS UI Framework

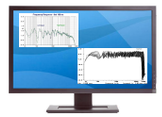
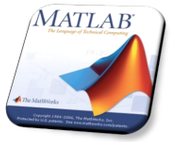
!CHOAS Common Framework

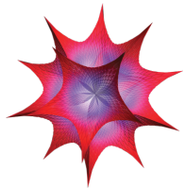
!CHOAS MetaData Services	!CHOAS Data Services

FS Driver

Simple Vision



Device Driver (Control Unit)	Algorithm	UI Application
!CHOAS CU Framework	!CHOAS EU Framework	!CHOAS UI Framework

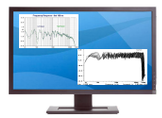
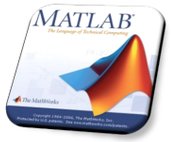
!CHOAS Common Framework

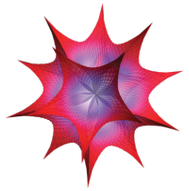
!CHOAS MetaData Services	!CHOAS Data Services	
	FS Driver	Index Driver



Simple Vision





Device Driver
(Control Unit)

Algorithm

UI Application

!CHOAS CU Framework

!CHOAS EU Framework

!CHOAS UI Framework

!CHOAS Common Framework

!CHOAS MetaData Services

!CHOAS Data Services

FS Driver

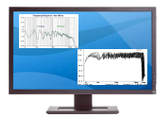
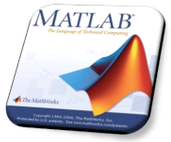
Index Driver

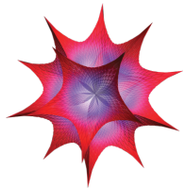


Framework Boundary

Simple Vision





Device Driver
(Control Unit)

Algorithm

UI Application

!CHOAS CU Framework

!CHOAS EU Framework

!CHOAS UI Framework

!CHOAS Common Framework

!CHOAS MetaData Services

!CHOAS Data Services

FS Driver

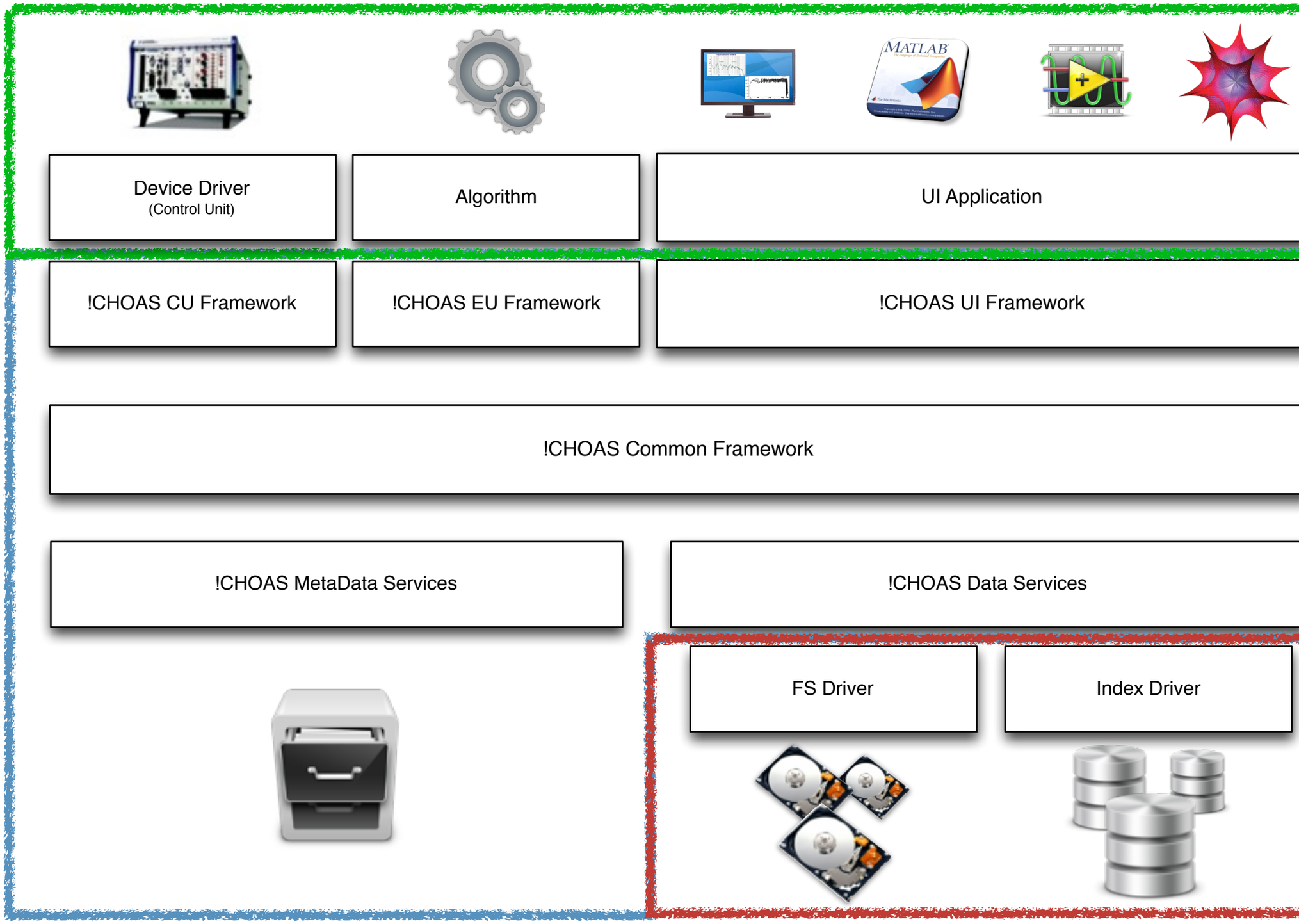
Index Driver



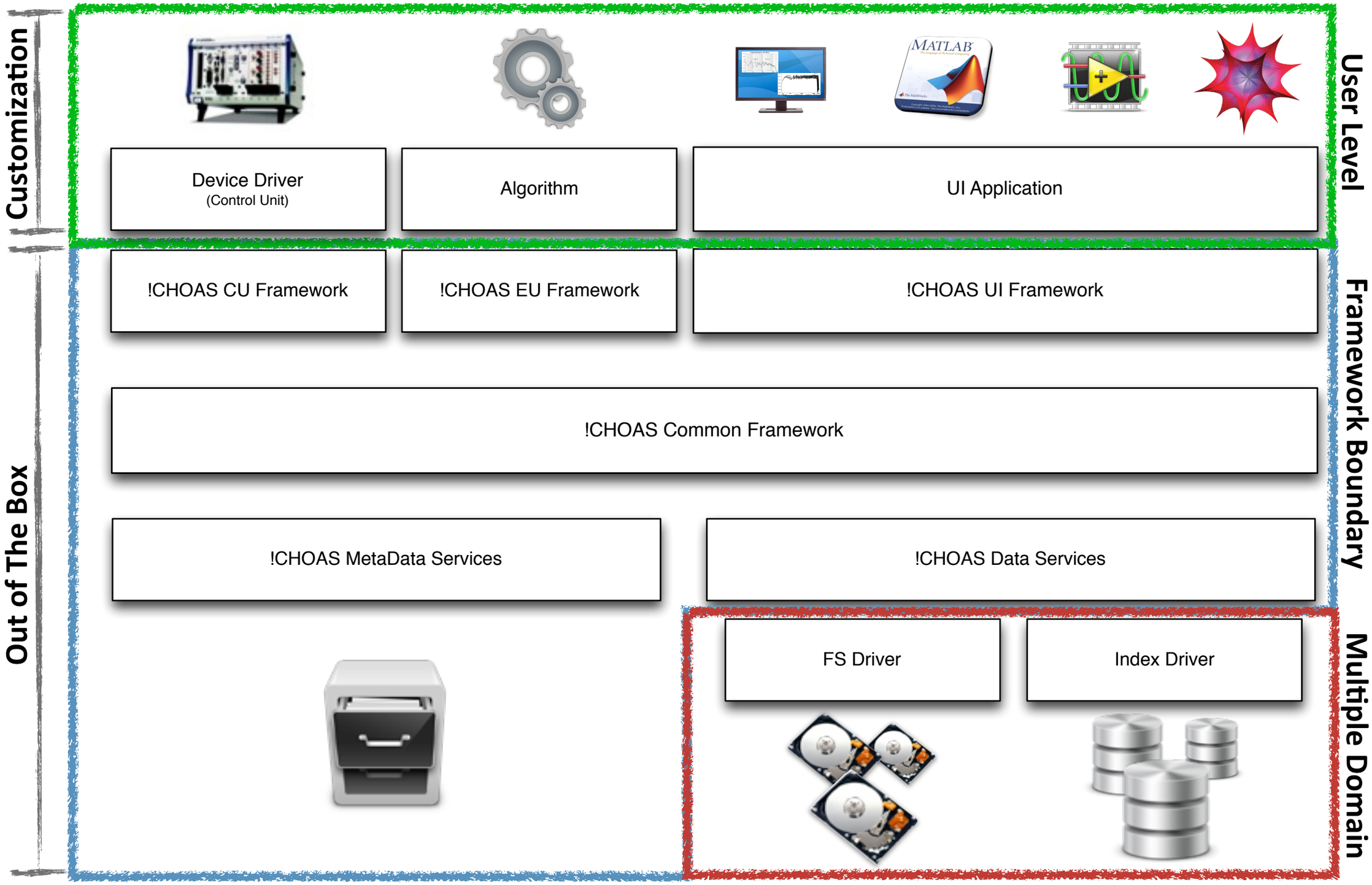
Framework Boundary

Multiple Domain

Simple Vision



Simple Vision



!CHAOS Real Model

User Customization Boundary

User Interface for monitoring and controls

Abstraction Boundary

User Interface Toolkit

Common Toolkit

RPC - Event - Direct I/O

Controls Algorithm

Execution Unit Toolkit

Common Toolkit

RPC - Event - Direct I/O

Data Boundary

CQL Proxy Nodes

Indexer Nodes

Management Nodes

Live / History domain

Live / History domain

Information Boundary

Chaos Directory Nodes

Node Information

Security

Node Activity

HTTP - API

RPC - Event - Direct I/O

Common Toolkit

Control Unit Toolkit

Driver for sensor and instruments

!CHAOS Framework Toolkit

!CHAOS Framework Toolkit

Common Toolkit

implements the fundamental software's layer to RPC, I/O, Event and other's utility code and for logic

CU Toolkit

abstracts the !CHAOS resources to the device drivers developers.

UI Toolkit

abstracts the CHAOS resources to the developers that need to develop control and monitoring user interface

EU Toolkit

abstracts the CHAOS resources to the developers that need to develop control and computing algorithms

!CHAOS Services

!CHAOS Services

Metadata Server

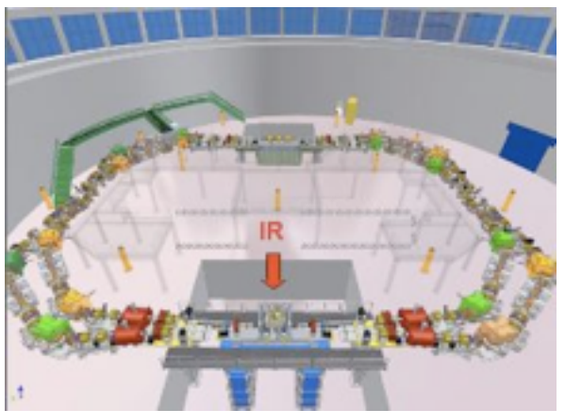
The information system for the setup and the topology of the !CHOAS installation. COntains information about all device connected and for all data domains. Performs the load balance logics in static mode

Data Cloud Domain

Implements the live and historical data management logics. All internal service (Proxy, Storage, Indexer) are scalable. Every domain can be iplemented on different technolgies (FS, IndexDB)

!CHAOS Services

Metadata Servers



Slow Control Data

SC Data Cloud
[live/HST]

Slow Control Data

Detector Data

Experiment Cloud
[HST]

!CHAOS Communication Systems

Communication Systems

!CHOAS has three different communication system, implemented via plug-ins:

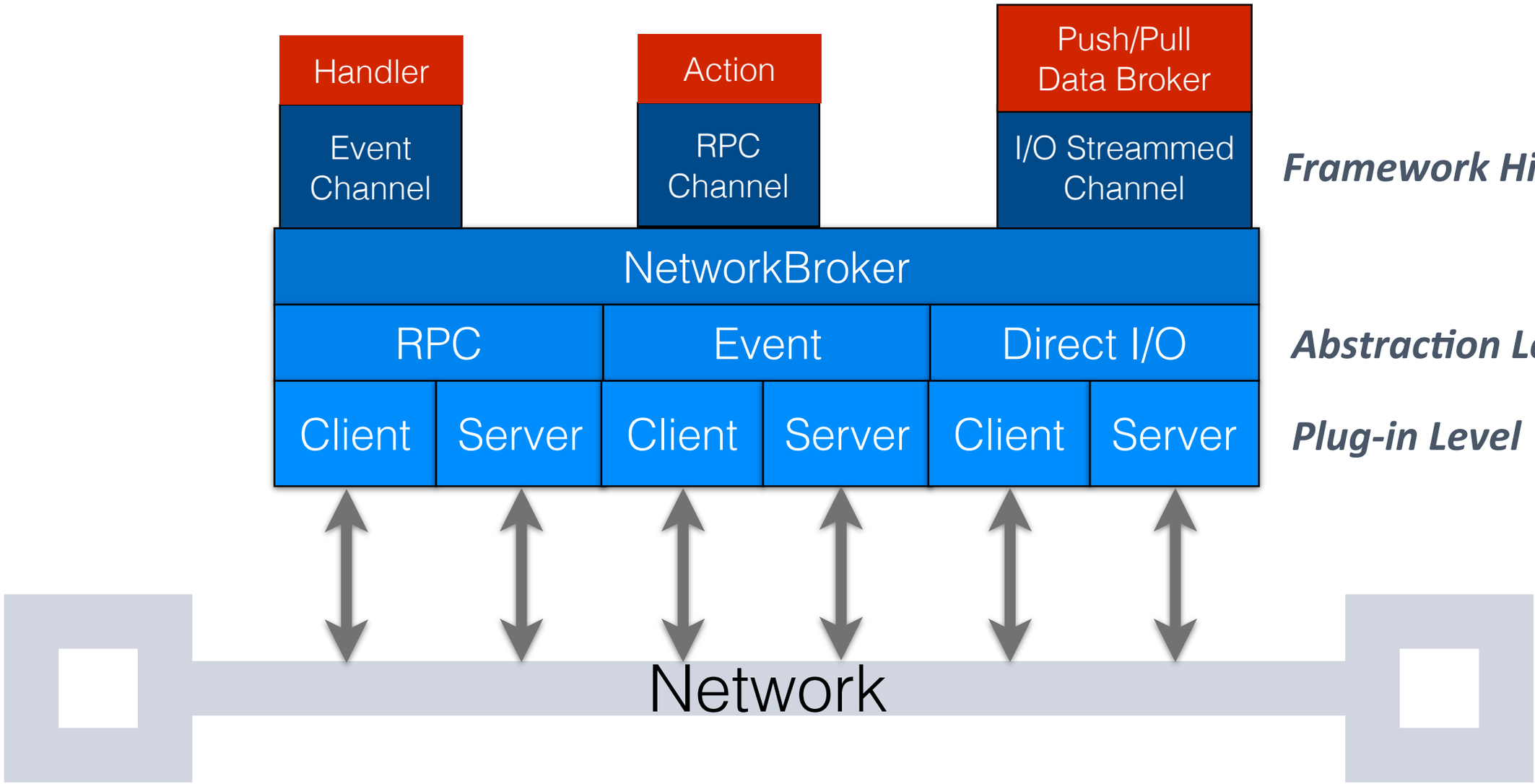
RPC, used for command (messages that need to be securely received and recorded and for message that need a response [asynchronous])

Events, used for inform to a wide range of nodes, that something has happened (device heartbeat, alert on device channel, received command, etc..)

Direct I/O, used for streams to and from the live, historical data or inter node data exchange

Communication Systems

CommonToolkit, implementa a software switch that permit, to all other class, to access all three protocoll.



Framework High Level

Abstraction Level

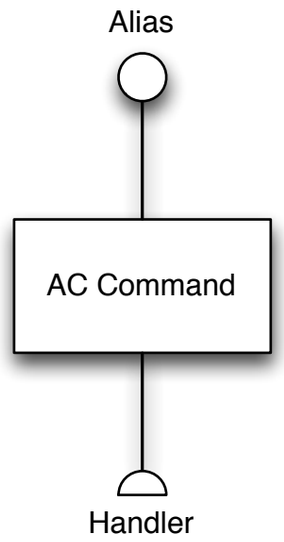
Plug-in Level

!CHAOS ControlUnit toolkit

CU Toolkit

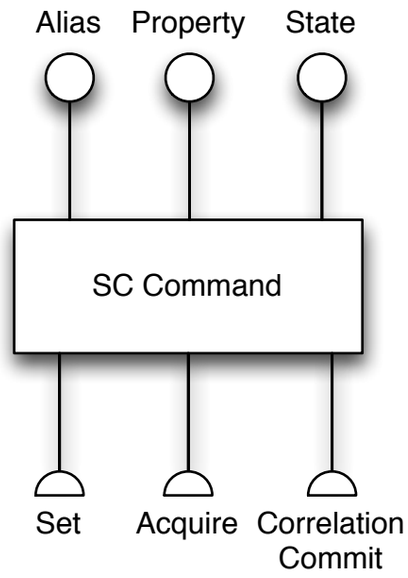
- CUToolkit help the Control Unit (Device Driver) development
- A ControlUnit attach an hardware to !CHAOS
- Abstract the Command Implementation as AC (Action) and SC (SlowControl)
- Complex schedule for SL Command
- Implement the main control run-loop (Acquisition/Control)

CU Toolkit Command



- AC Command is a simple RPC call for an action attacched to an alias
- it is executed in a different thread than the main run-loop, the appropriate synchronizations need to be done by developer

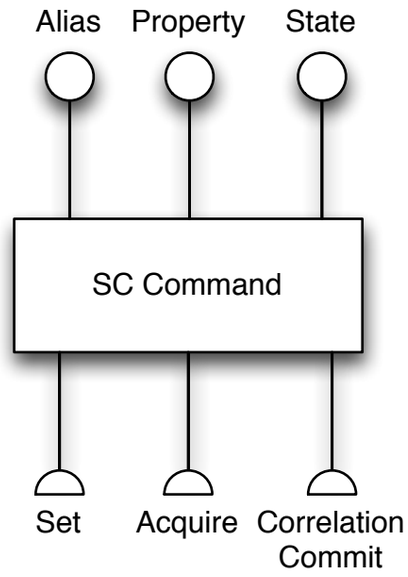
CU Toolkit Command



Slow Control Command implements up to three handler

- **Set Handler** (mandatory), need to set the device register for achieve the command purpose
- **Acquire Handler** (optional) need to acquire the data needed by next handler
- **Correlation and Commit** (optional), need to check the current value and execute hardwre control for achieve the command purpose

CU Toolkit Command



Slow Control can have also property and state:

- **property**, are used to tweek framework behaviour within command execution, this also contai the input valu efor the command;

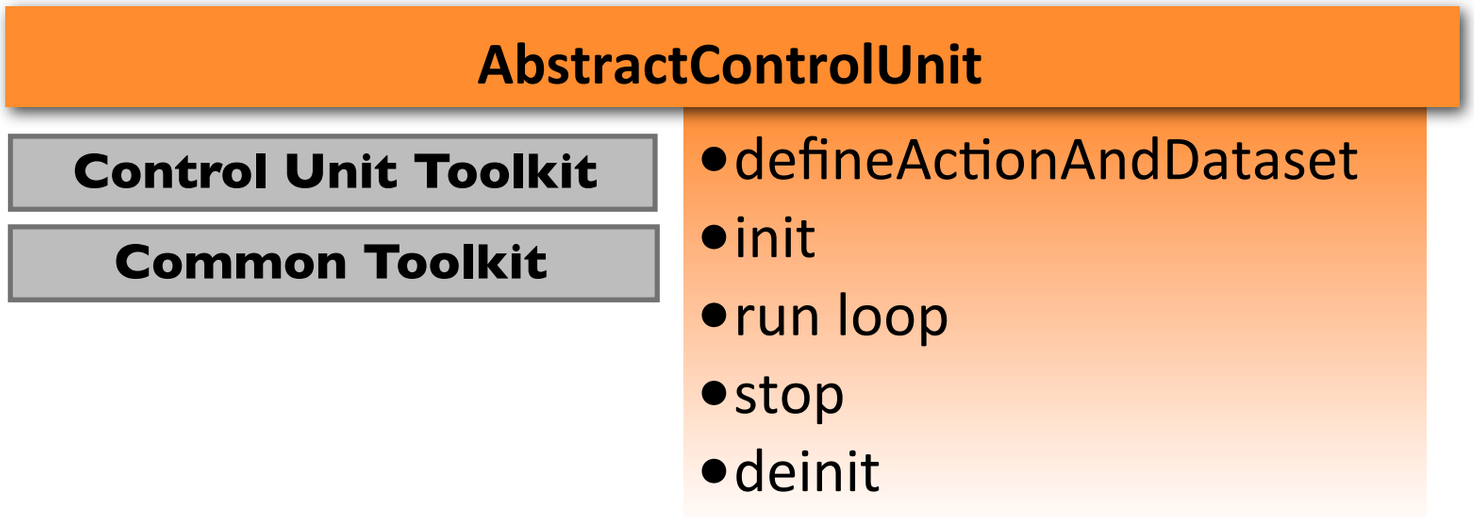
- **state**, tracks the current state of the command {fault, end, killable, overridable, stackable}

CU Toolkit Scheme

The Control Unit developed by the user need to extend a C++ class

The ACU implements the mandatory methods used by !CHOAS to get device information and for control it

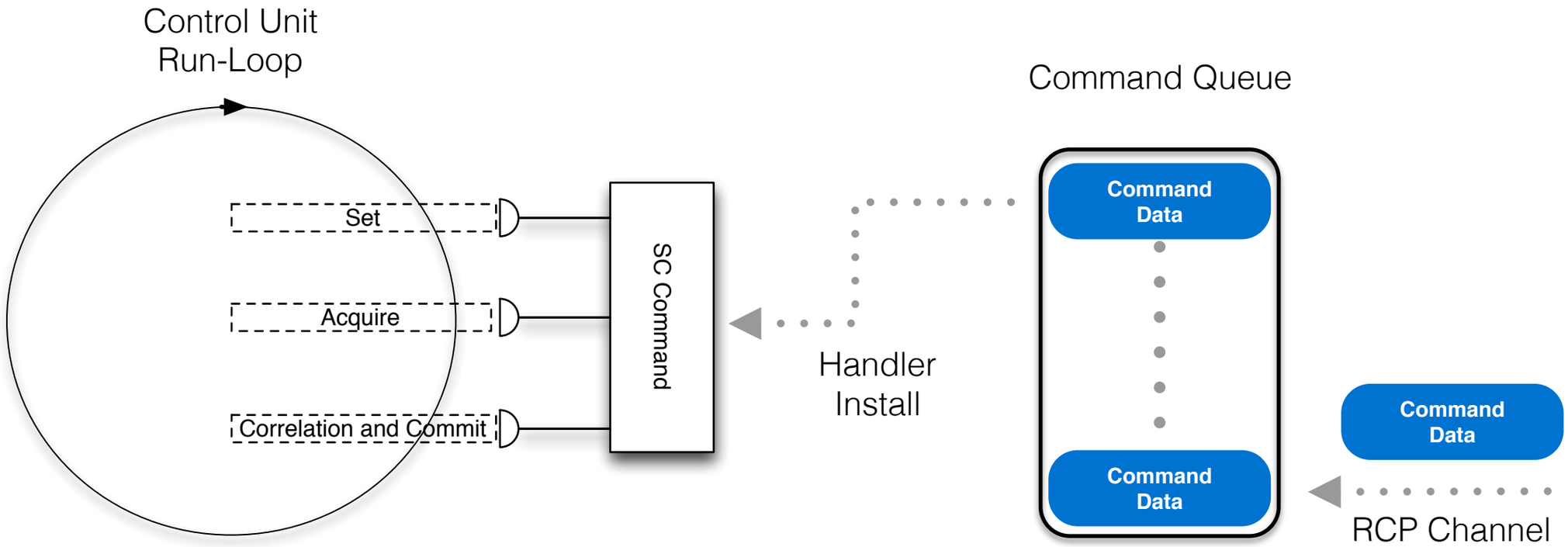
CU Toolkit Scheme



The Control Unit developed by the user need to extend a C++ class

The ACU implements the mandatory methods used by !CHOAS to get device information and for control it

CU Toolkit Scheme



The Control Unit developed by the user need to extend a C++ class

The ACU implements the mandatory methods used by !CHOAS to get device information and for control it

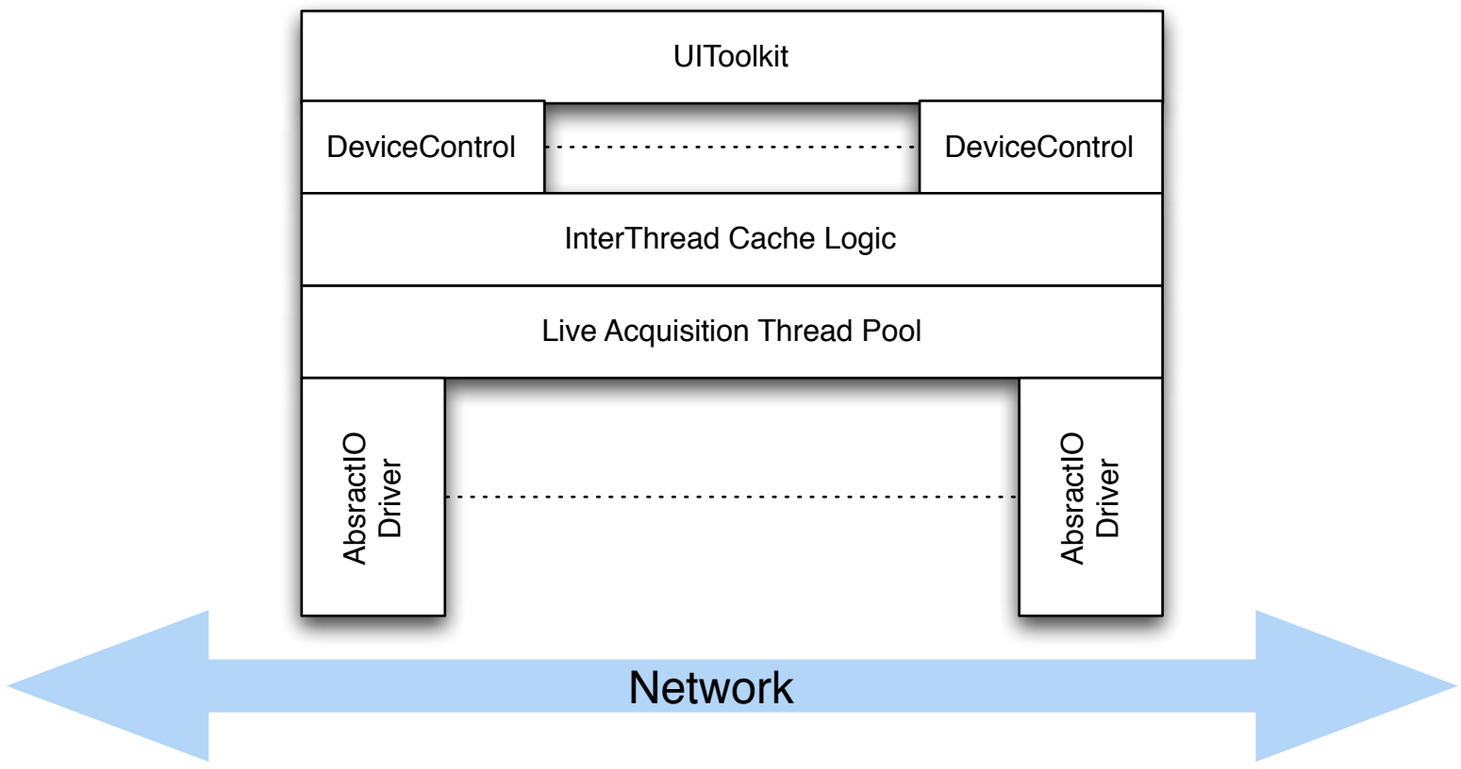
!CHAOS UI Toollkit

UI Toolkit

UIToolkit helps the user interface development

Gives API that help to develop: monitoring processes, device controller etc...

Gives an high performance queue using "locking free" design. The queue can manage quantizations , over time, for the dataset data acquisition



UI Toolkit

A “c” interface is provide to access UIToolkit into software that don’t permit to use C++ as developer language.

This permit to realize an user interface or controller algorithm into another software as plugin, as for example LabView.

The “C” proxy has a minimal set of interface to simplify the device controller allocation, deallocation and usage.

!CHAOS EU Toolkit

EU Toolkit

- helps the development of distributed algorithms (Execution Unit) that can be used to compute or automate the devices control

- The EU can have, in input the device channels or all the Dataset description, so it work on a "Class" of device not directly on one device.

- The EU can be also used to perform calculation instead control algoritm;

- on EU can be preset in many instance distributed on different fiscal server

- Still under design

Execution Unit Example 1

Device Driver
(Control Unit)

Algorithm
(Custom Execution Unit)

Device Driver
(Control Unit)

CU_1 DATA BLOCK

live data

Execution Unit Example 1

this is the real data flow

Output channel data is
read from device

Device Driver
(Control Unit)

Algorithm
(Custom Execution Unit)

Device Driver
(Control Unit)

CU_1 DATA BLOCK *live data*

Execution Unit Example 1

this is the real data flow

Device Driver
(Control Unit)

Algorithm
(Custom Execution Unit)

Device Driver
(Control Unit)

Output channel data
is pushed
on live memory block
from CU_1

CU_1 DATA BLOCK *live data*

Execution Unit Example 1

this is the real data flow

Device Driver
(Control Unit)

Algorithm
(Custom Execution Unit)

Device Driver
(Control Unit)

Output CU_1 channel
data is read from live

CU_1 DATA BLOCK *live data*

Execution Unit Example 1

this is the real data flow

some logic is done

Device Driver
(Control Unit)

Algorithm
(Custom Execution Unit)

Device Driver
(Control Unit)

CU_1 DATA BLOCK

live data

Execution Unit Example 1

this is the real data flow

Device Driver
(Control Unit)

Algorithm
(Custom Execution Unit)

Device Driver
(Control Unit)

The Input channel on CU_2
is set with RPC command

CU_1 DATA BLOCK *live data*

Execution Unit Example 1

this is the real data flow

The input channel is set on
HW
by CU_2

Device Driver
(Control Unit)

Algorithm
(Custom Execution Unit)

Device Driver
(Control Unit)

CU_1 DATA BLOCK

live data

!CHAOS Data Cloud & Services

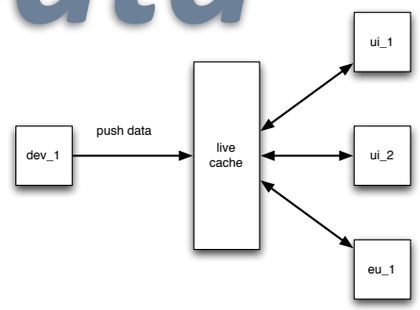
Data Management

!CHOAS has two different management system for the data apparatus controlled environment data, managed by the “cloud” paradigma

Live Data

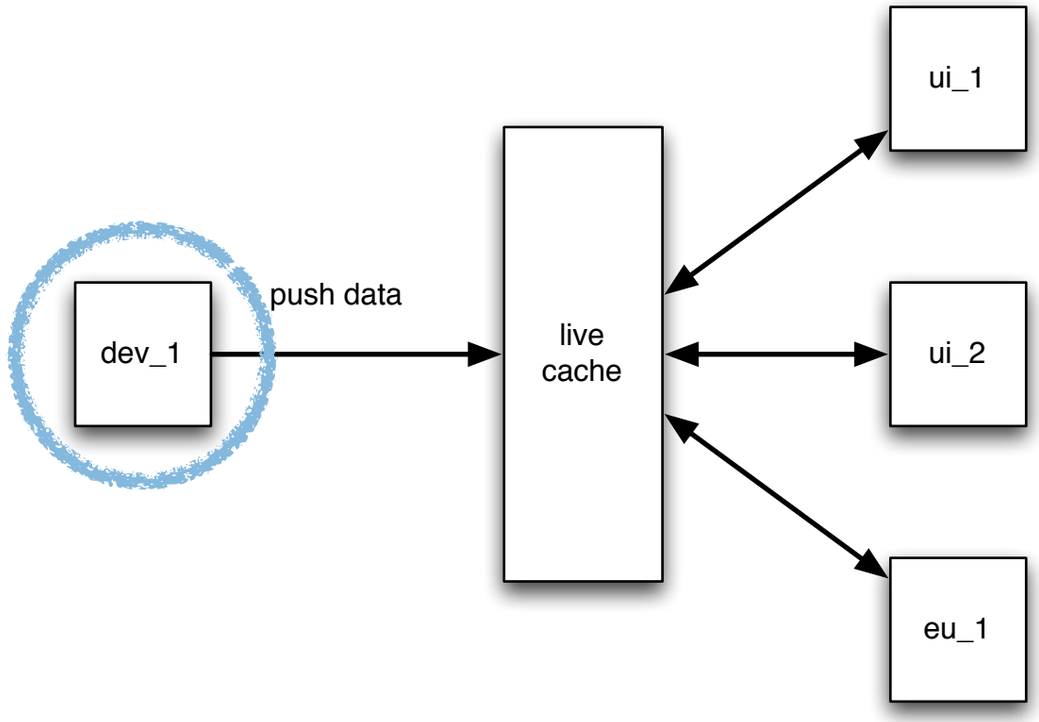
History Data

!CHAOS Live Data



Live Data

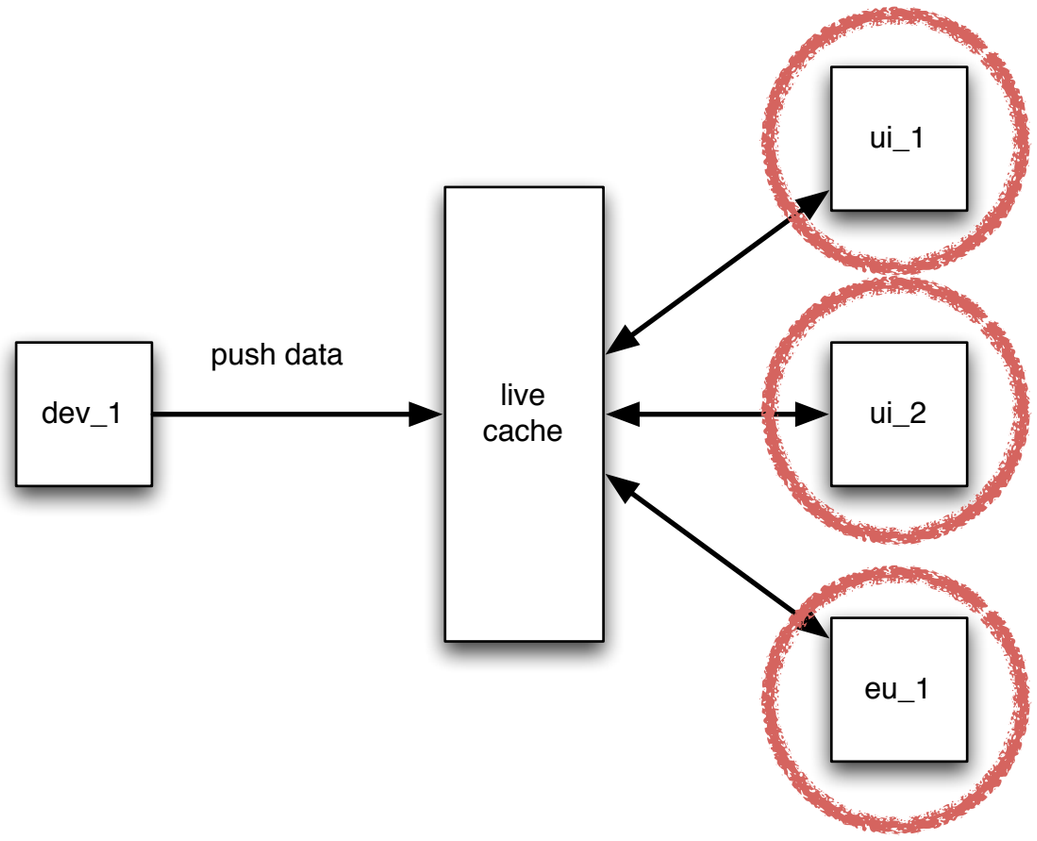
Every instrument pushes the data of his I/O channels into the shared cached through the controller CU



Live Data

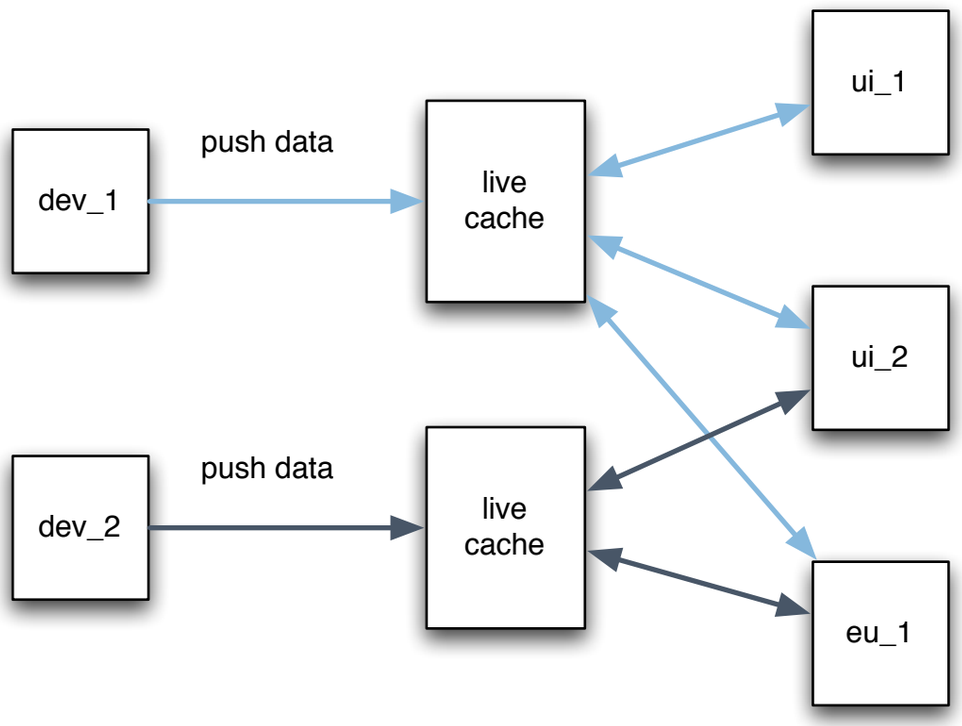
1. every node can read data from the central cache at any time[polling]

2. the nodes can register to central cache for receive update[push]



Live Data

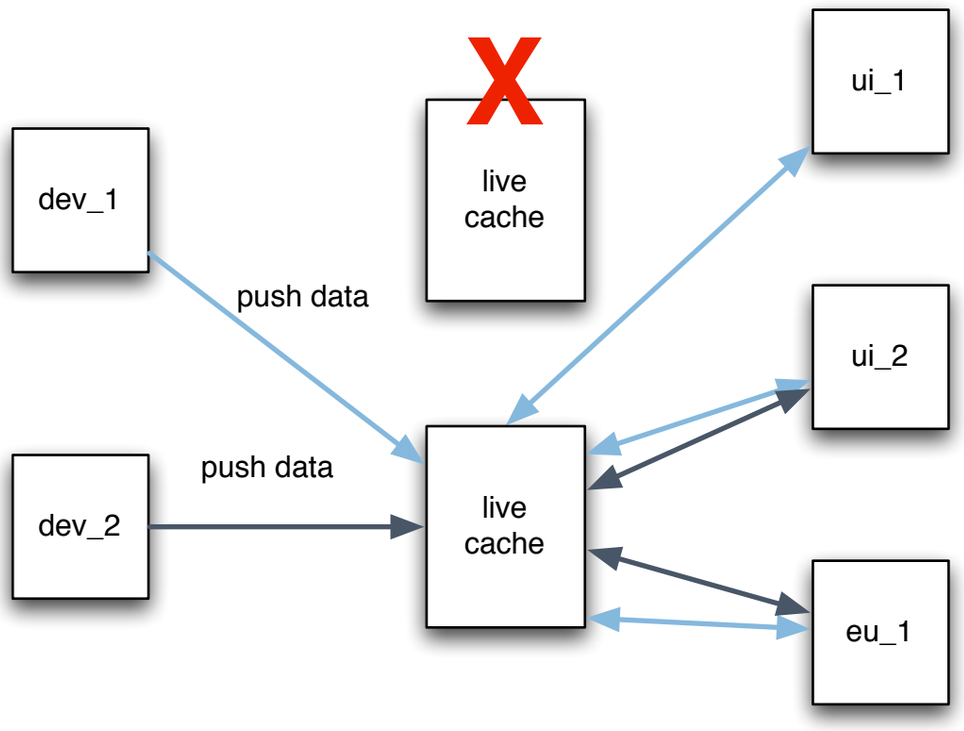
scalability & reliability



All data of all instruments are distributed across many cache server

Live Data

scalability & reliability



with the algorithm previously shown, when one server goes down, all client that push's, or read, data on that server, use another server, with highest priority

Push data service

every node, if authorized, can register itself on the Control Unit for “push” data service.

In !CHOAS this mode is permitted only for those nodes that need to get the device value in a deterministic time.

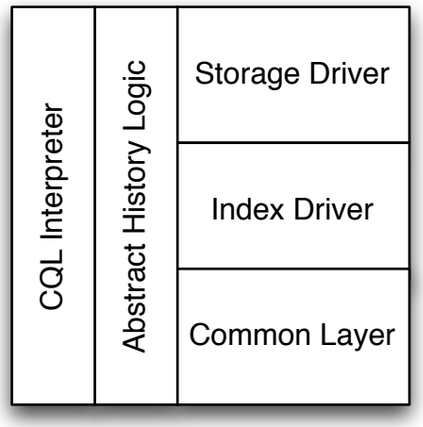
The number of “Push” client is regulated by the control managment console.

!CHAOS History Data

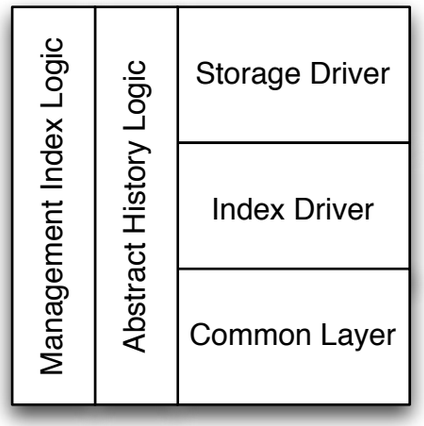
History Data

It is realized by three different **services** and two **driver**

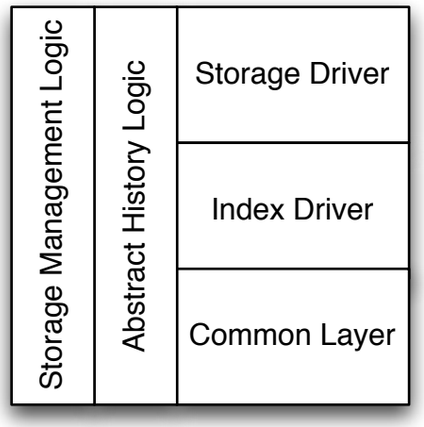
ChaosQL Proxy



Indexer Node



Management Node



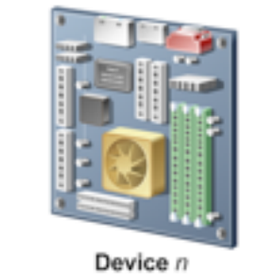
Storage Driver abstraction

Index Driver abstraction

Chaos Query Language

ChaosQL implement the following operation on data

- push history/live data per device
- create index on device::attribute::{rule}
- delete index on device::attribute
- retrieve data with logic operation on attribute and index



Device *n*

init
control loop
run (update dataset)
exec commands
de-init

CUTK

CommonTK

CQL Proxy

CQL Proxy

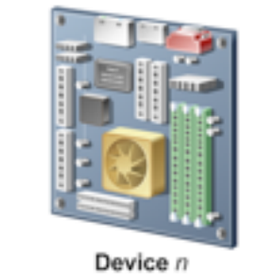
CQL Proxy

Storage

Indexer Cluster

Management Cluster

Index Database



Device *n*

init
control loop
run (update dataset)
exec commands
de-init

CUTK

CommonTK

Push Data

CQL Proxy

CQL Proxy

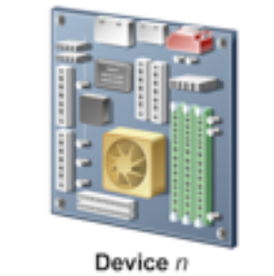
CQL Proxy

Indexer Cluster

Storage

Management Cluster

Index Database



Device *n*

init
control loop
run (update dataset)
exec commands
de-init

CUTK

CommonTK

Push Data

CQL Proxy

CQL Proxy

CQL Proxy

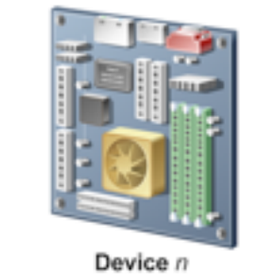
Store Data

Storage

Indexer Cluster

Management Cluster

Index Database



Device *n*

init
control loop
run (update dataset)
exec commands
de-init

CUTK

CommonTK

Push Data

CQL Proxy

CQL Proxy

CQL Proxy

Store Data

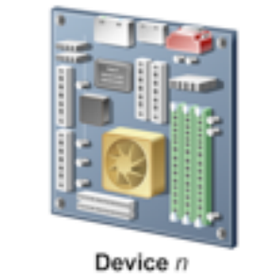
Launch
Index

Storage

Indexer
Cluster

Management
Cluster

Index Database



Device *n*

init
control loop
run (update dataset)
exec commands
de-init

CUTK

CommonTK

Push Data

CQL Proxy

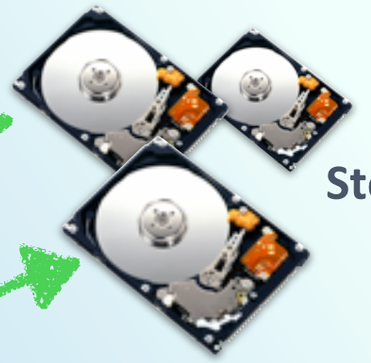
CQL Proxy

CQL Proxy

Store Data

Launch Index

get data pack

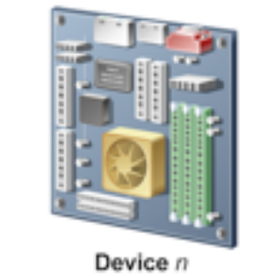


Storage

Indexer Cluster

Management Cluster

Index Database



Device *n*

init
control loop
run (update dataset)
exec commands
de-init

CUTK

CommonTK

Push Data

CQL Proxy

CQL Proxy

CQL Proxy

Store Data

Launch Index

get data pack

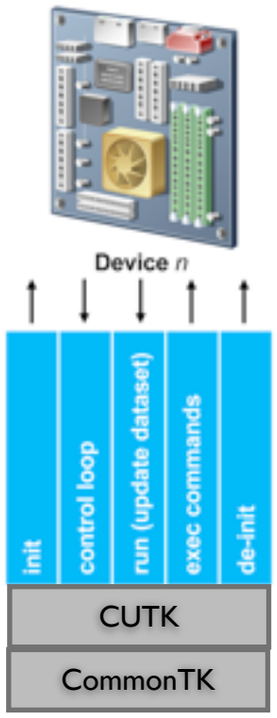
Store Index

Indexer Cluster

Storage

Management Cluster

Index Database



Device *n*

init
control loop
run (update dataset)
exec commands
de-init

CUTK

CommonTK

CQL Proxy

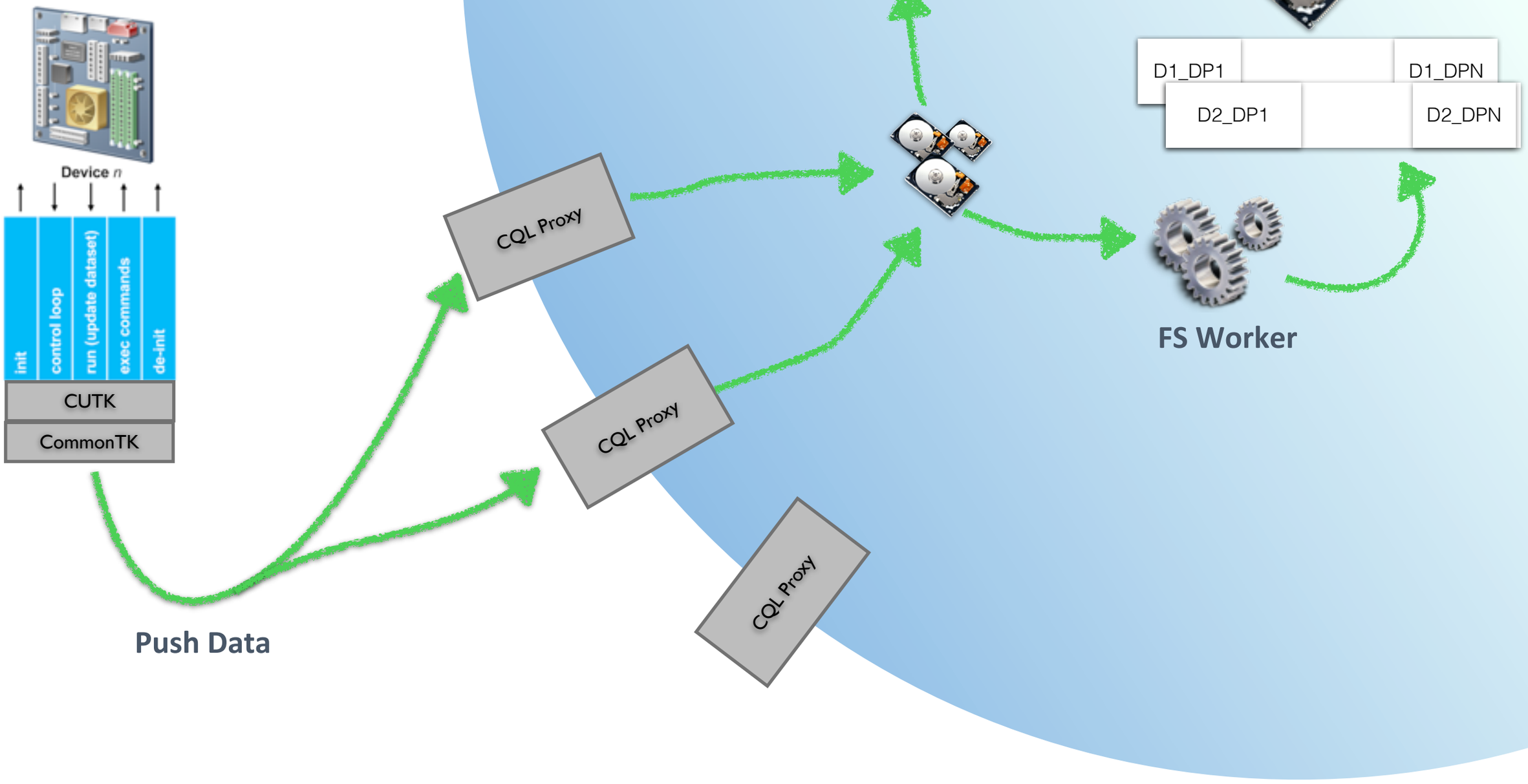
CQL Proxy

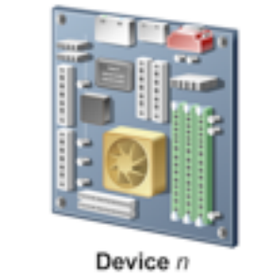
CQL Proxy

D1_DP1 | D2_DP | D1_DPN

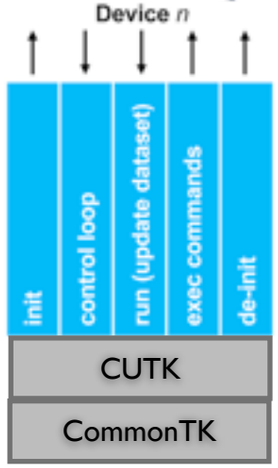
D1_DPN+1 | D2_DPN

Push Data


Device n
init
control loop
run (update dataset)
exec commands
de-init
CUTK
CommonTK
Push Data
CQL Proxy
CQL Proxy
CQL Proxy
FS Worker
FS Worker
Storage
D1_DP1
D1_DPN
D2_DP1
D2_DPN



Device *n*



Management Cluster

Storage

Indexer Cluster

CQL Proxy

CQL Proxy

CQL Proxy

Index Database

Device *n*

init

control loop

run (update dataset)

exec commands

de-init

CUTK

CommonTK

Management Cluster

Storage

Indexer Cluster

CQL Proxy

CQL Proxy

CQL Proxy

Index Database

EUTK

CommonTK

UITK

CommonTK

Device *n*

init | control loop | run (update dataset) | exec commands | de-init

CUTK

CommonTK

Management Cluster

Storage

Indexer Cluster

CQL Proxy

CQL Proxy

CQL Proxy

Query Live Data

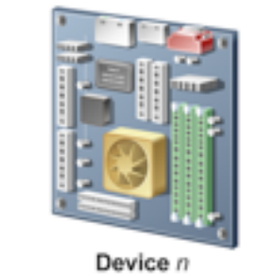
EUTK

CommonTK

UITK

CommonTK

Index Database



Device *n*

init | control loop | run (update dataset) | exec commands | de-init

CUTK

CommonTK

Management Cluster

Storage

Indexer Cluster

CQL Proxy

CQL Proxy

CQL Proxy

Query Live Data

EUTK

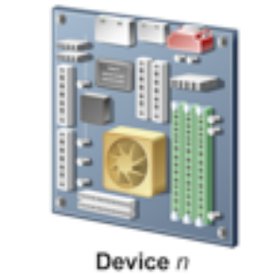
CommonTK

UITK

CommonTK

Query History Data

Index Database



Device *n*

init | control loop | run (update dataset) | exec commands | de-init

CUTK

CommonTK

Management Cluster

Storage

Indexer Cluster

execute search on idx DB

CQL Proxy

CQL Proxy

CQL Proxy

send search on idx

Index Database

EUTK

CommonTK

Query Live Data

UITK

CommonTK

Query History Data


Device n
init
control loop
run (update dataset)
exec commands
de-init
CUTK
CommonTK

Management Cluster
Storage
get pack at location
Indexer Cluster
execute search on idx DB
CQL Proxy
CQL Proxy
CQL Proxy
send search on idx
Query Live Data
EUTK
CommonTK
UITK
CommonTK
Query History Data
Index Database

Device *n*

init

control loop

run (update dataset)

exec commands

de-init

CUTK

CommonTK

Management Cluster

Storage

get pack at location

Indexer Cluster

CQL Proxy

execute search on idx DB

Query Live Data

CQL Proxy

send search on idx

EUTK

CommonTK

Receive History Data

CQL Proxy

UITK

CommonTK

Query History Data

Index Database

...closing

in the next year all development force will be focused to realize the system for the history data and for the metadata server, a prototype for each system will be available for 2014 2Q

CHOAS Group

Claudio Bisegni, Luciano Catani,

Giampiero Di Pirro, Luca Foggetta,

Mazziteli Giovanni, Matteo Mara,

Stecchi Alessandro

thesis's

Claudio Bisegni (Tor Vergata SSMMFF Informatica), Flaminio Antonucci (Tor Vergata Ingengeria), Andrea Capozzi (Tor Vergata Ingengeria), Francesco Iesu (Cagliari Infromatica),

thanks for the time