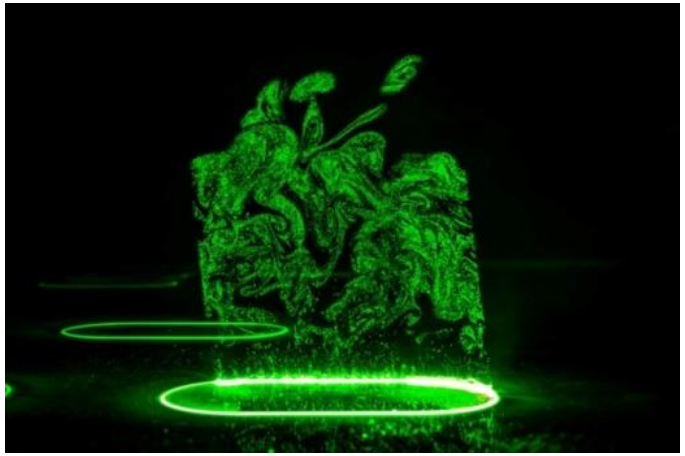
Fusion Middleware.

A comparison of state of the art systems in research, industrial manufacturing and cloud.

LLT Lehrstuhl für Lasertechnik | RWTH AACHEN UNIVERSITY

Digital Photonics at RWTH Aachen LLT



Chair:
LLT RWTH Aachen

Group:
Digital Photonics

Professor:
Constantin Häfner

Fusion Research:
- Datainfrastructure for laser systems
- Cloud based control systems and experiment execution systems
- Integration of Datascience and online Optimization into large scale lasers

LLT Lehrstuhl für Lasertechnik | RWTH AACHEN UNIVERSITY

Strategic initiatives at the innovation hot spot Aachen, Germany

Research Building DPP

Fraunhofer ILT

Industry Building DPP

Research Center DPP (RWTH I³)

Forschungscampus DPP (BMBF)

Collaborative Research areas
DFG funded research groups
Excellence Cluster Production Technology
DFG funded projects

BMBF funded joint projects
AiF funded projects
European funded projects
Fraunhofer internal research

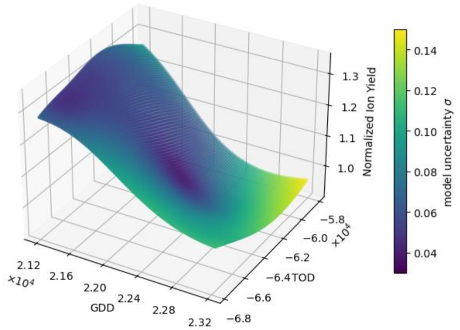
Prototype manufacture
Contract Research
Education and training

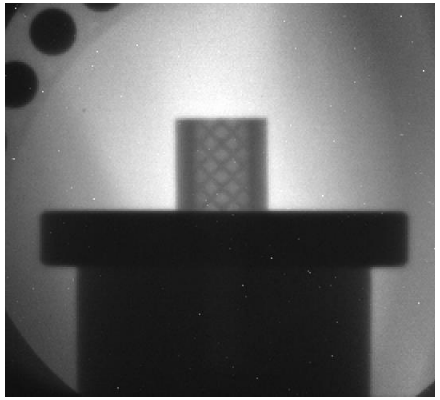
Production
Spin-off
Consortia research projects
Feasibility studies

Basic Digital Engineering Focus: AI-Driven Optimization of Laser Sources for Fusion Applications

Conducted Experiments

- AI-based optimization of laser-driven ion sources at ELI Beamlines (November 2023)
- AI-based optimization of laser-driven X-ray sources for radiography and CT applications at ELI Beamlines (August 2025)

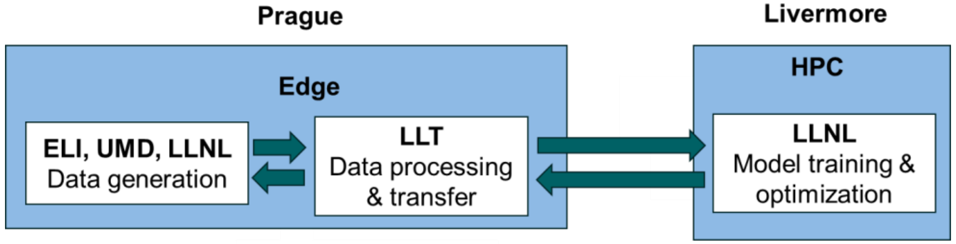




Relevance to Fusion Research

- Ion sources are fundamental to the *fast-ignition* concept, where a compressed fusion target is ignited using an internal particle beam.
- Laser-driven X-ray sources can be used for radiography of exploding fusion targets, enabling scientists to visualize the laser–target interaction

Lawrence Livermore National Laboratory | eli | University of Maryland 1856



LLT Lehrstuhl für Lasertechnik | RWTH AACHEN UNIVERSITY

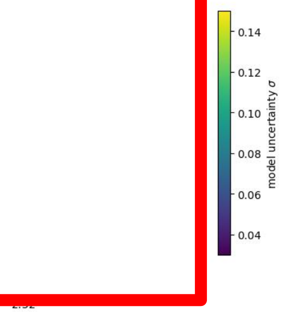
Basic Digital Engineering Focus: AI-Driven Optimization of Laser Sources for Fusion Applications

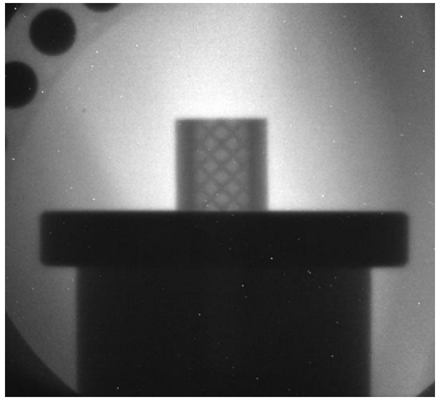
Conducted Experiments
- AI-based optimization of laser-driven ion sources at ELI Beamlines (November 2023)
- AI-based optimization of laser-driven X-ray sources for radiography and CT applications at ELI Beamlines (August 2025)

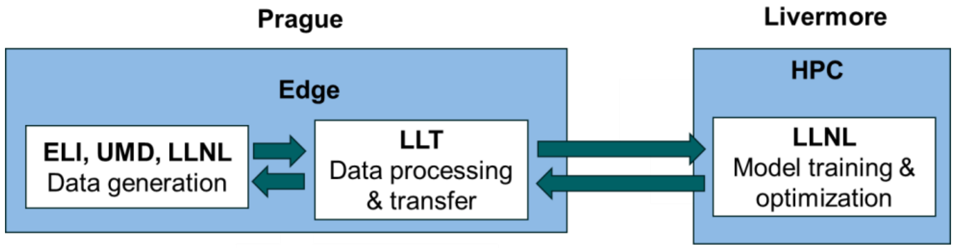
Relevance to Fusion Research
- Ion sources are fundamenta… a compressed fusion target … beam.
- Laser-driven X-ray sources … exploding fusion targets, enabling scientists to visualize the laser–target interaction

Poster Session:
Integrating Edge Analytics and HPC for Autonomous HED Experimental Workflows

Abhik Sarkar





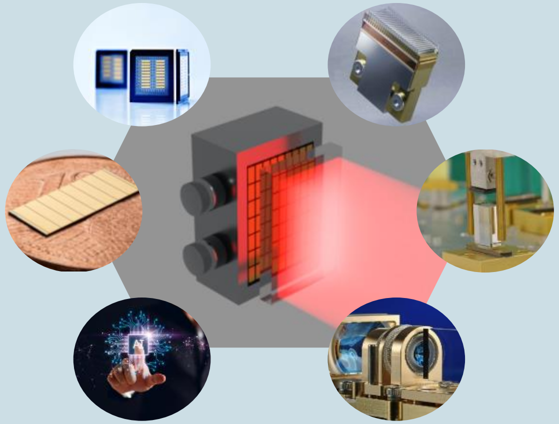


Lawrence Livermore National Laboratory | eli | University of Maryland 1856

LLT Lehrstuhl für Lasertechnik | RWTH AACHEN UNIVERSITY

Fraunhofer ILT is a leading Partner for Industry Participation for Fusion Energy Research

DIOHELIOS

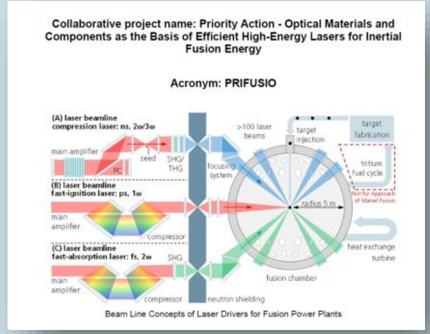


Diode laser pump source for high energy lasers in fusion power plants (started in 10/2024)

Consortium:

JENOPTIK, FBH Ferdinand Braun Institut, laserline, TRUMPF, Fraunhofer ILT, ams OSRAM

PriFUSIO

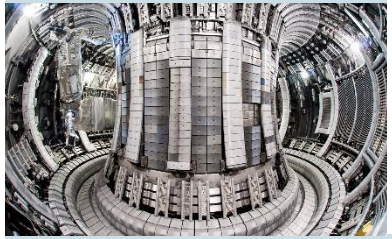


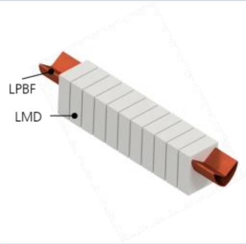
Development of Optical Materials and Components for Efficient High Energy Lasers for IFE

Consortium

Marvel Fusion, FOCUSED ENERGY, TRUMPF, LAYERTEC Optical Coatings · Optics, LZH Laser Zentrum Hannover e.V., Heraeus, LASEROPTIK, SCHOTT, Fraunhofer ILT, Fraunhofer IOF

DURABLE





Material development for novel first wall and divertor solutions through LAM

Consortium:

GAUSS FUSION, ModuleWorks, REUTER Technologie, access, research instruments, AMCM, H.C.Starck, toolcraft, Fraunhofer ILT, JÜLICH Forschungszentrum

IFE Target Hub

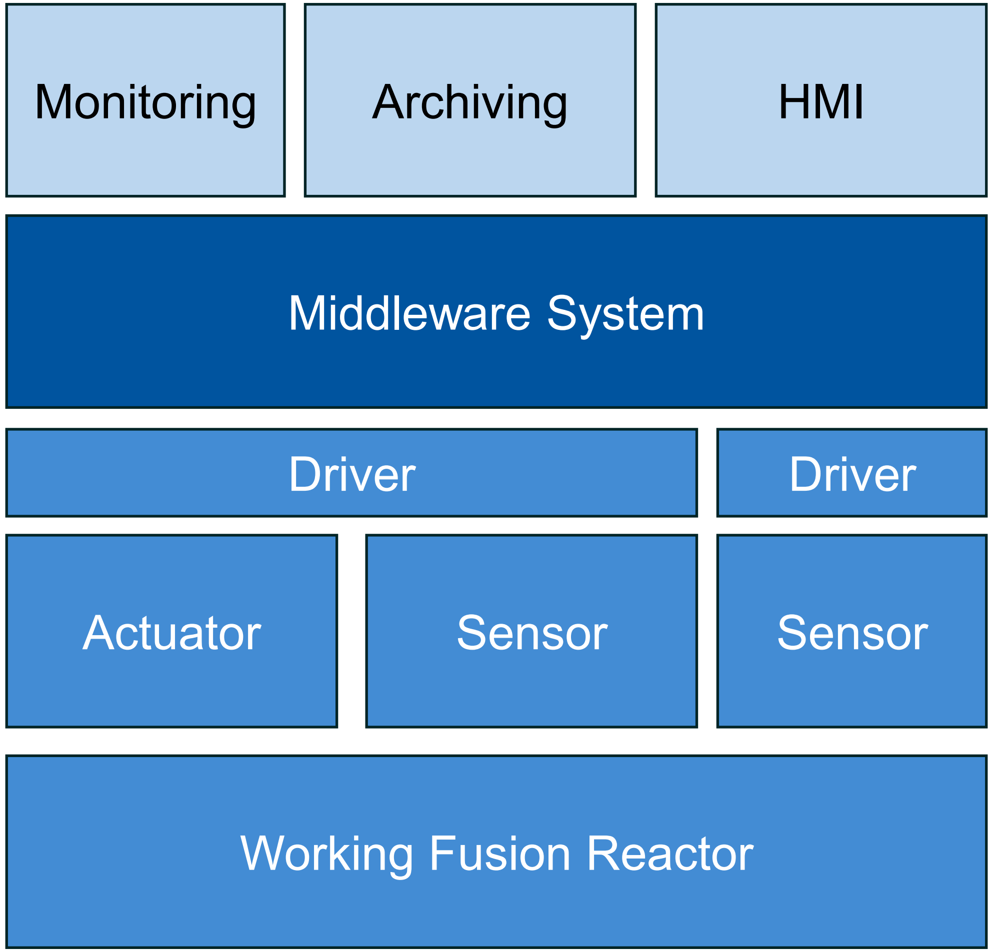
Developing materials and methods for functional, cost-effective scaling and characterization of targets IFE

Consortium:

FOCUSED ENERGY, plasmatreat, Diamond Materials Advanced Diamond Technologies, HERKULA High-Chem Speciality Inks, LightFab, KERN, ModuleWorks, KIT Karlsruher Institut für Technologie, Fraunhofer ILT, IFAM, IAF, IIS, Technische Universität Darmstadt

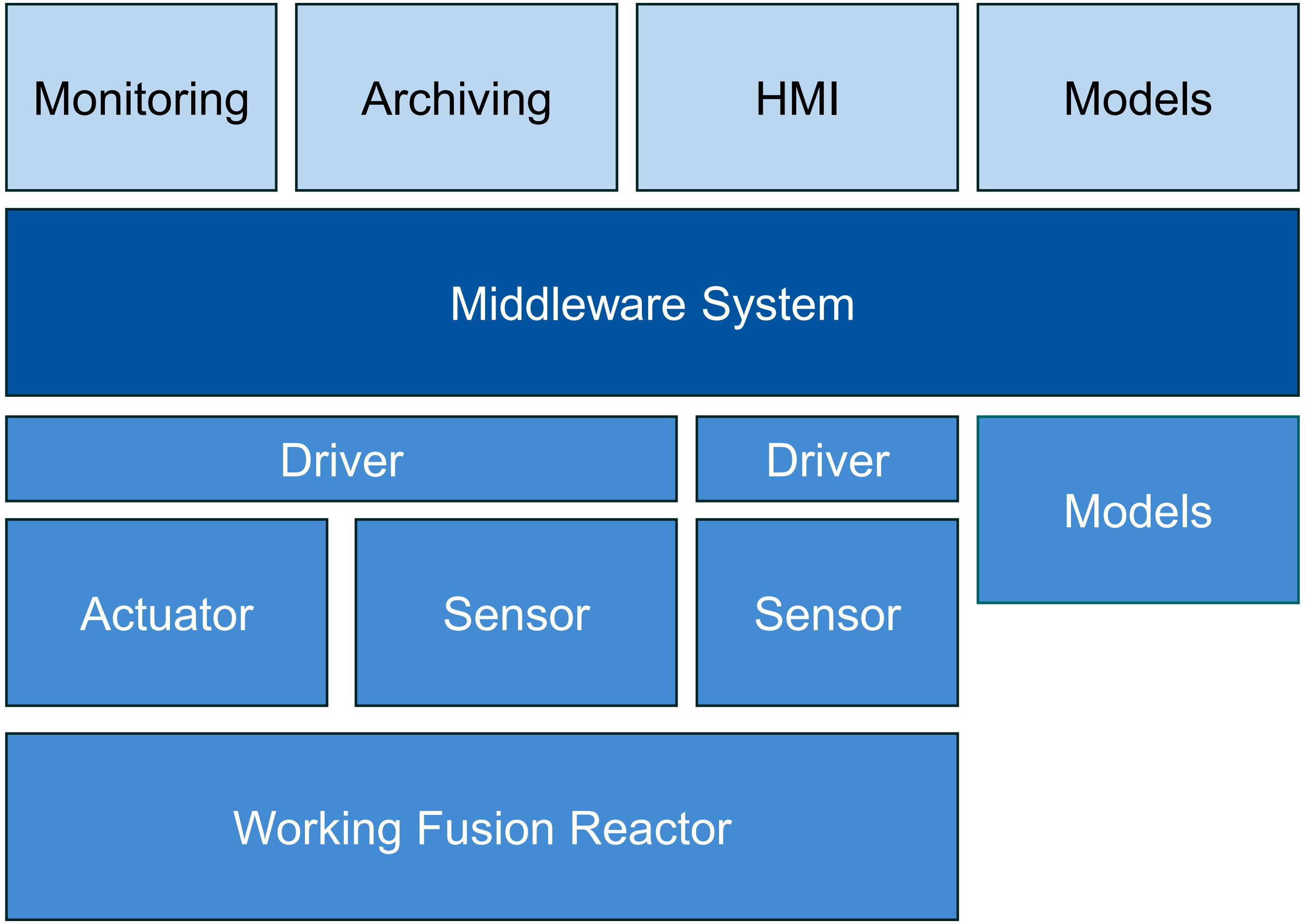
1. What is Middleware
2. EPICS and Tango Basics
3. Workflow and Messaging
4. What's missing
5. Summary

Example Fusion Control System



Example Fusion Control System

Middleware to be



What is Middleware

Frameworks of this talk

Scientific technologies:
- EPICS
- Tango

Manufacturing technologies:
- Closest is ROS
- OPC/UA only a protocol

Cloud technologies:
- Cloud native computing foundation (CNCF)
- Apache foundation

What is Middleware

Frameworks of this talk

Scientific technologies:
- EPICS
- Tango

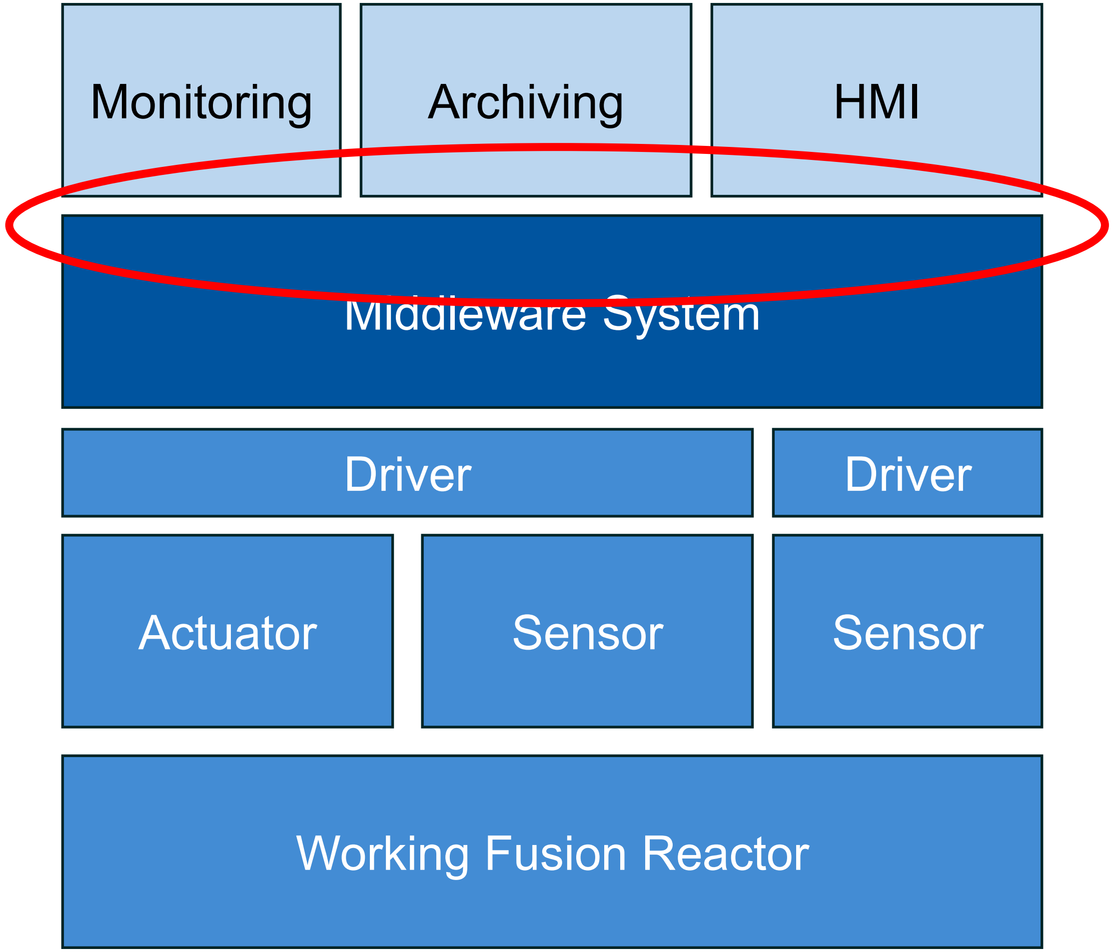
Manufacturing technologies:
- Closest is ROS
- OPC/UA only a protocol

Cloud technologies:
- Cloud native computing foundation (CNCF)
- Apache foundation

Focus on concepts behind the frameworks as well as the general architecture.

No in depth analysis or performance analysis!

Concept Example Fusion Control System



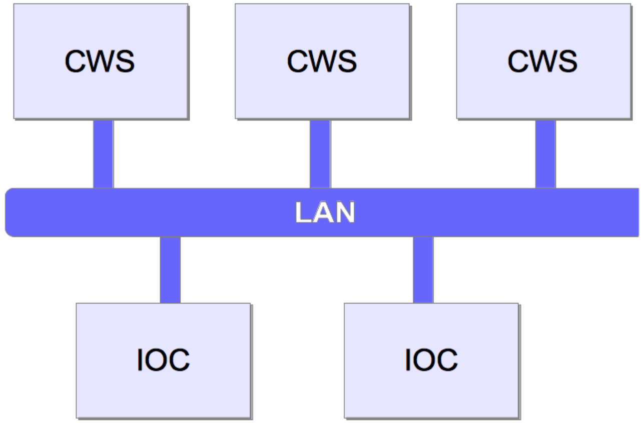
1. What is Middleware
2. EPICS and Tango Basics
3. Workflow and Messaging
4. What's missing
5. Summary

EPICS

The Experimental Physics and Industrial Control System

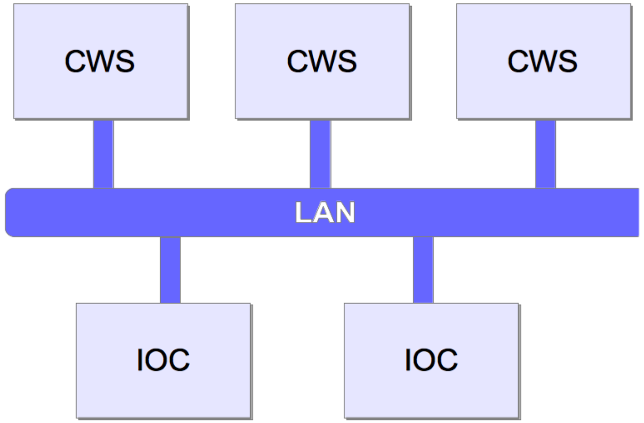
- Inital Release: 19.01.1994
- License: EPICS Open License
- Founded by : LANL and APS
- Framework for communication between distributed Client Work Stations (CWS) and Input Output Controller (IOC)
- IOC are central interface for device representation

- Concept Paper:
- McDowell, William P., et al. *Standards and the design of the Advanced Photon Source control system.* No. ANL/CP--74764. Argonne National Lab., IL (United States), 1991.



Adopters

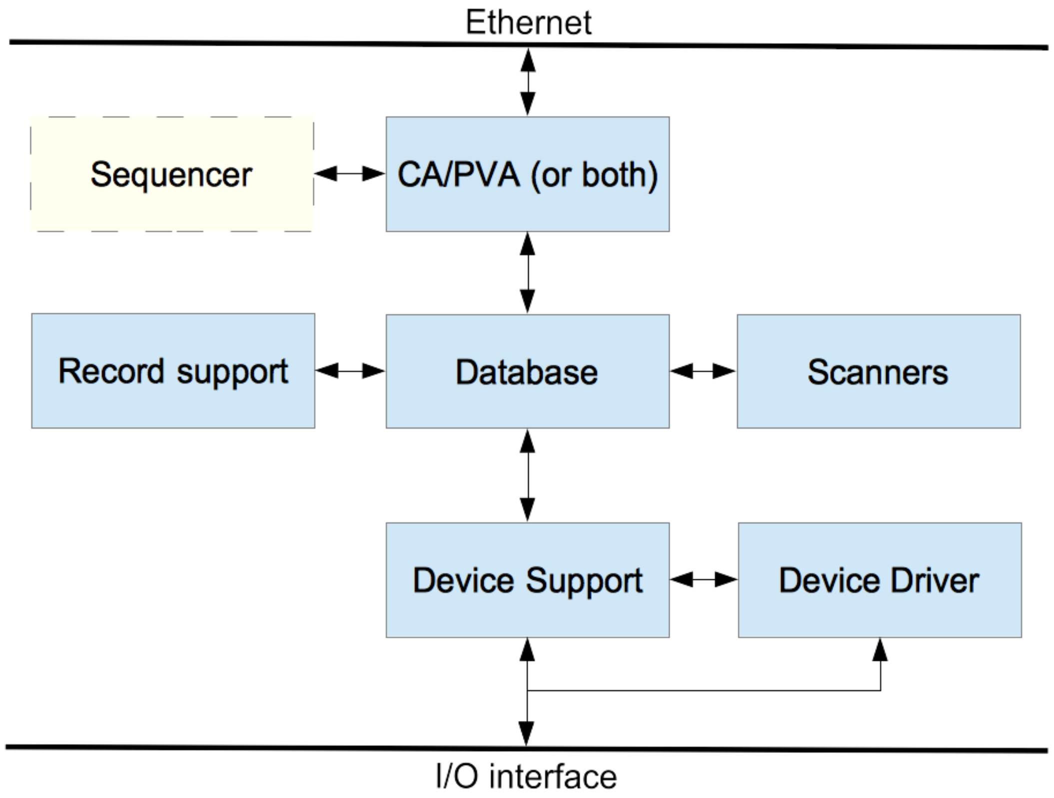
- ITER
- SLAC National Accelerator Laboratory
- Advanced Photon Source
- KSTAR
- ELI Beamlines
- …



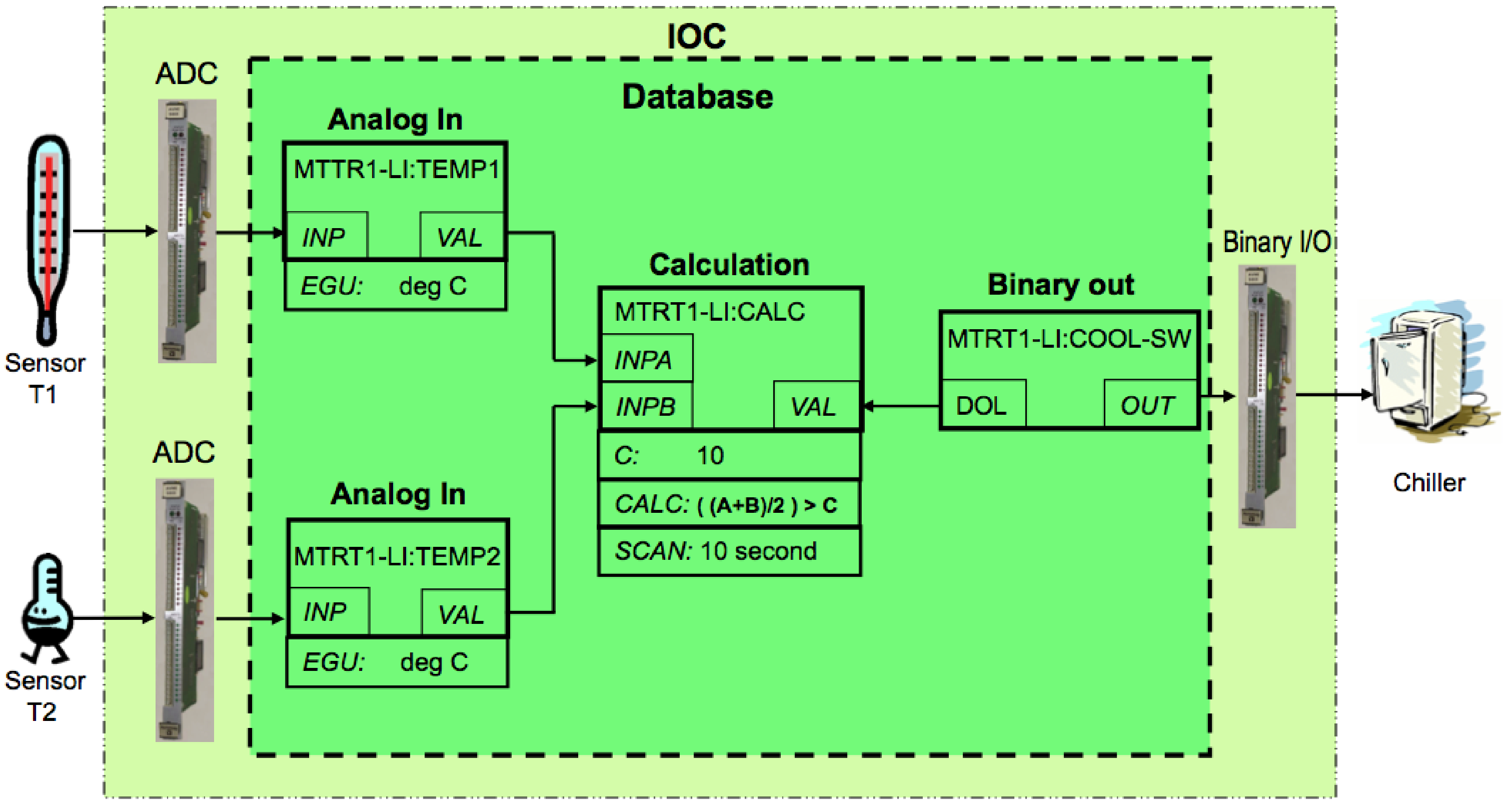
LLT Lehrstuhl für Lasertechnik | RWTH AACHEN UNIVERSITY

Epics IOC Design in a Nutshell

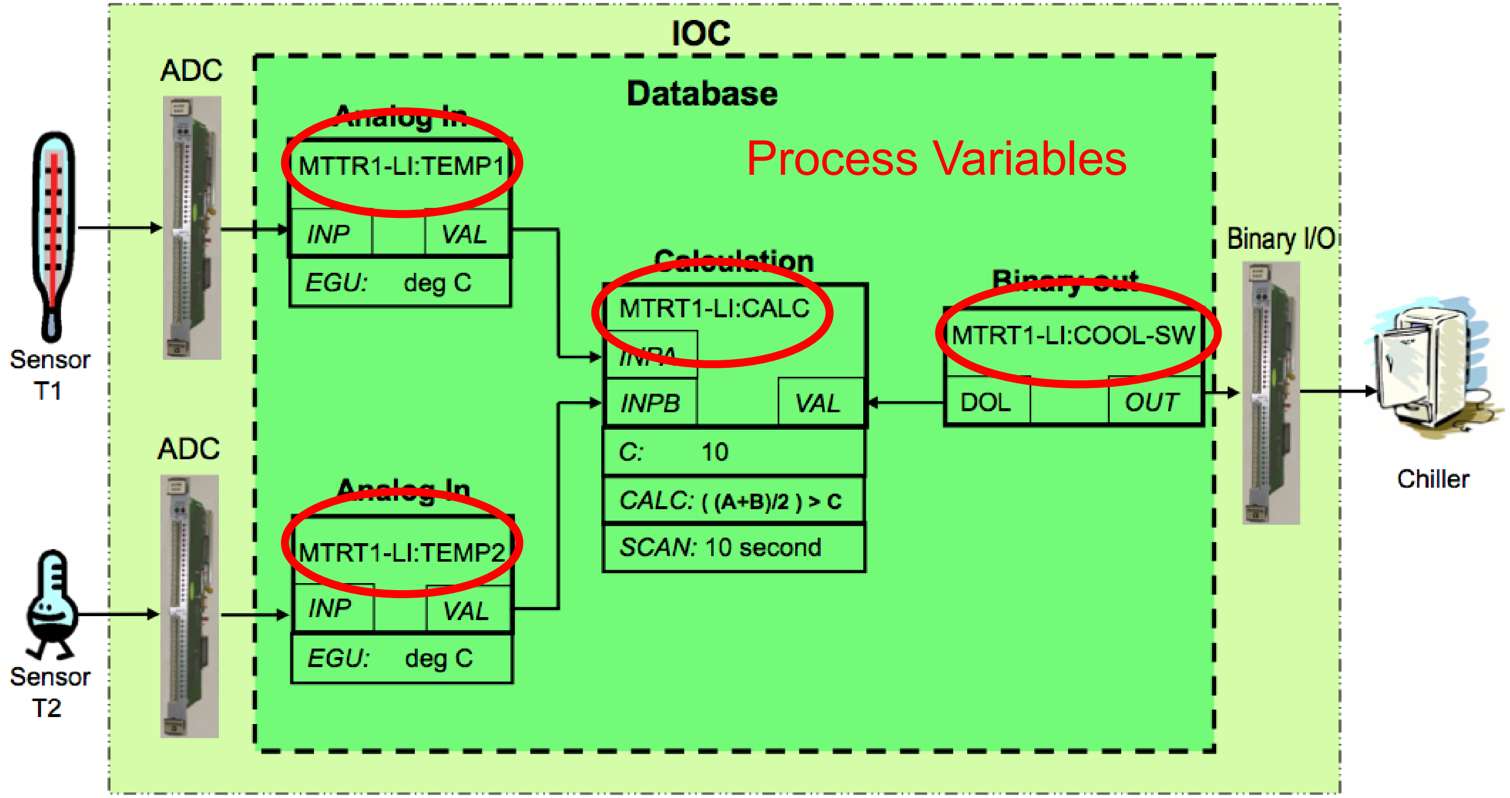
- Main component of an EPIC‘s Server is the Database:
  - Not a common Database like SQL
  - Defines records, record types and metadata like names, scan times, descriptions etc. and connection between them.
  - Every IoC has it‘s own database
- Three record types:
  - Analog Input Output
  - Binary Input Ouput
  - Calc and Calcout
- Records can be accessed by ChannelAcces or pvAccess (newer)
  - Search (Find a PV)
  - Get (Get PV Value)
  - Put (Set PV Value)
  - Add Monitor (Get Notified on change)
  - PutGet (Put and Get Atomic)
  - ChannelRPC (Run a remote function)



Epics IOC Design in a Nutshell



Epics IOC Design in a Nutshell



Tango Control System

- Inital Release: 19.04.2001
- License: GNU GPL v3
- Device-oriented controls toolkit for controlling any kind of hardware or software and building SCADA (supervisory control and data acquisition) systems

- Concept Paper:
  Chaize, Jean-Michel, et al. "TANGO: An Object Oriented Control System Based on CORBA." (1999).

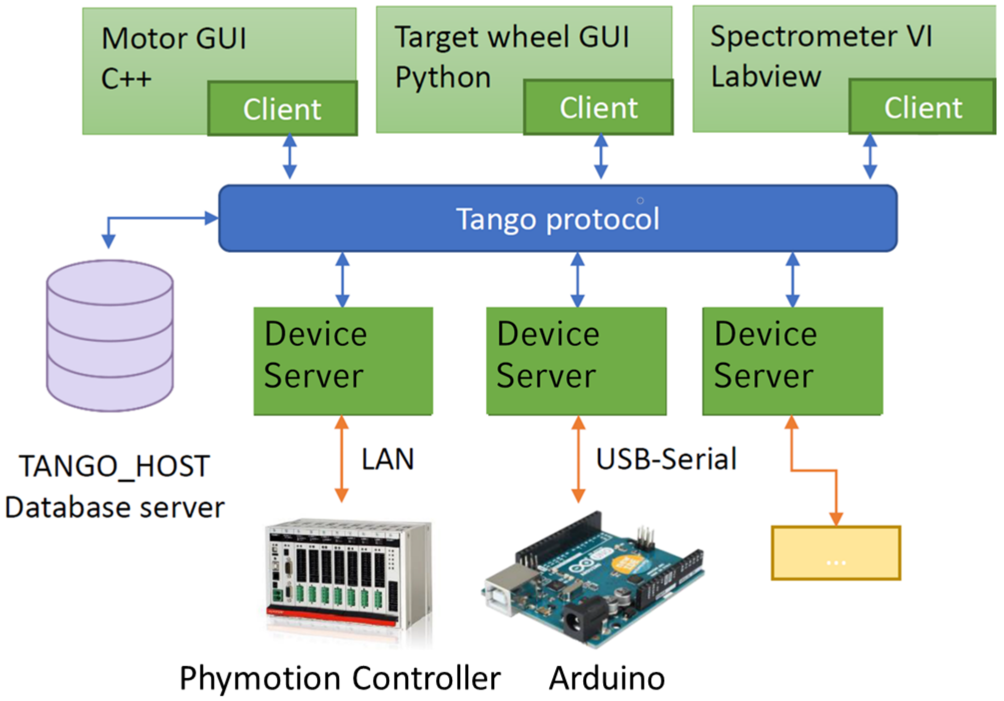
Tango

Adopters

- ESRF
- DESY
- ALBA
- ELI
- HZG
- …

Tango Design in a Nutshell

- Main component is the Database Server:
  - Database manages Devices in the Network
  - Databases manages creation of new connections inside the network. Therefore clients connect to the database before connecting to the devices.
- Exposed Device Servers Datatypes:
  - Commands
  - Device Properties
  - Attributes
  - (Pipes)
  - States



Src: Weiße N, Doyle L, Gebhard J, et al. Tango Controls and data pipeline for petawatt laser experiments. *High Power Laser Science and Engineering*. 2023;11:e44. doi:10.1017/hpl.2023.17

Workflow Comparison – Creating a new EPICS IoC – Cpp

Epics Cpp IOC

1. Decide if you want a Soft or Hard IoC
   – Hard IoC allows direct access to Hardware I/O
2. Download and setup EPICS Base
3. Get the records that you need for you application
   – Check if somebody built a module that you can download
   – Write your own module and compile it
4. Add the modules to your IoC application by adding it to your startup script
5. Define your IoC application by creating a database file
6. Start your IoC application
7. Process variables show up in your network

```
record(ai, "Cavity1:T") #type = ai, name = “Cavity1:T”
{
  field(DESC, “Cavity Temperature”) #description
  field(SCAN, “1 second”) #record update rate
  field(DTYP, “XYZ ADC”) #Device type
  field(INP, “#C1 S4”) #input channel
  field(PREC, “1”) #display precision
  field(LINR, “typeJdegC”) #conversion spec
  field(EGU, “degrees C”) #engineering units
  field(HOPR, “100”) #highest value on GUI
  field(LOPR, “0”) #lowest value on GUI
  field(HIGH, “65”) #High alarm limit
  field(HSV, “MINOR”) #Severity of “high” alarm
}
```

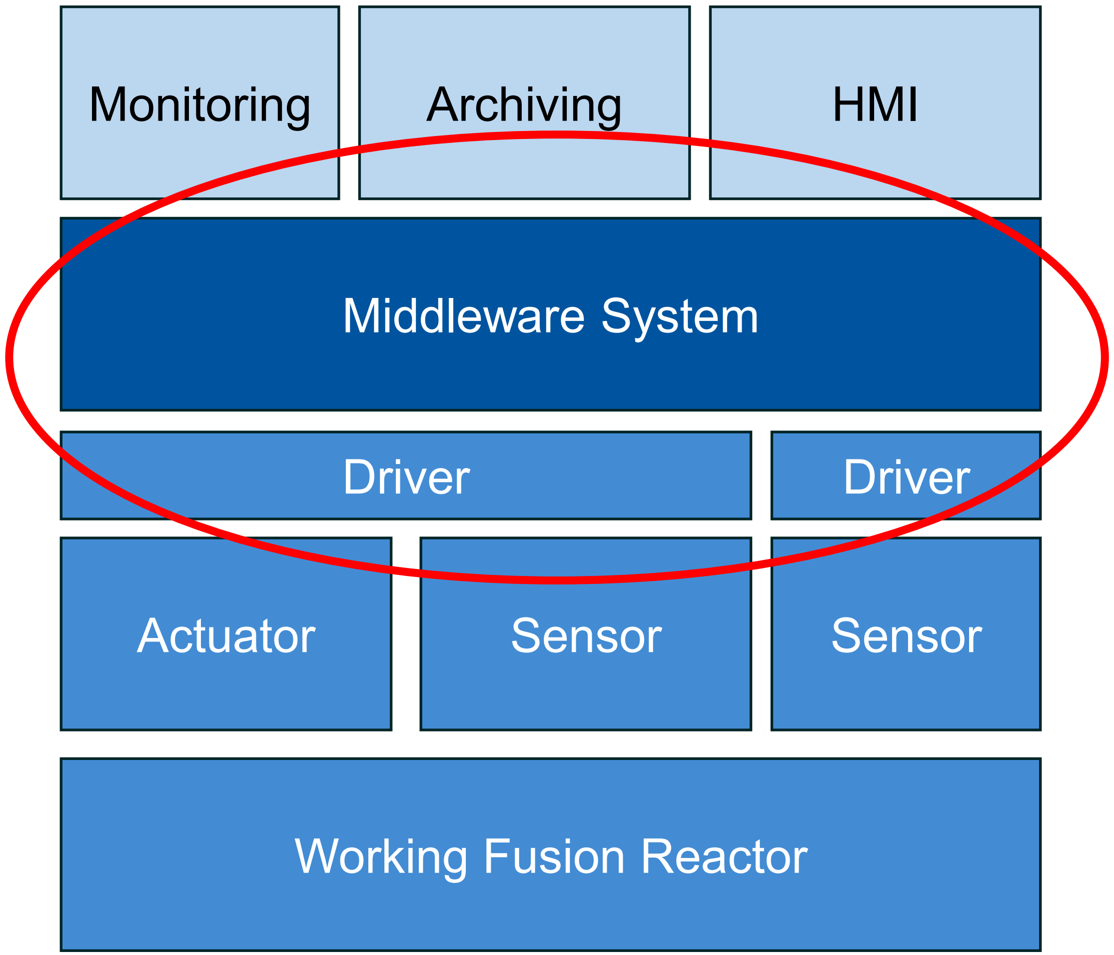
Example record definition in database file

Workflow Comparison – Creating a new EPICS IoC – Python

Epics Python IOC

1. Decide on your Python Lib
   - pythonSoftIoC: Implements record-like PV Definition (No modules though)
   - Caproto: Pure Python implementation of EPICS CA protocol
2. Write your IoC application in Python
3. Start your IoC application
4. Process variables show up in your network

Epics not only focuses on the message delivery of a middleware but also on the driver management



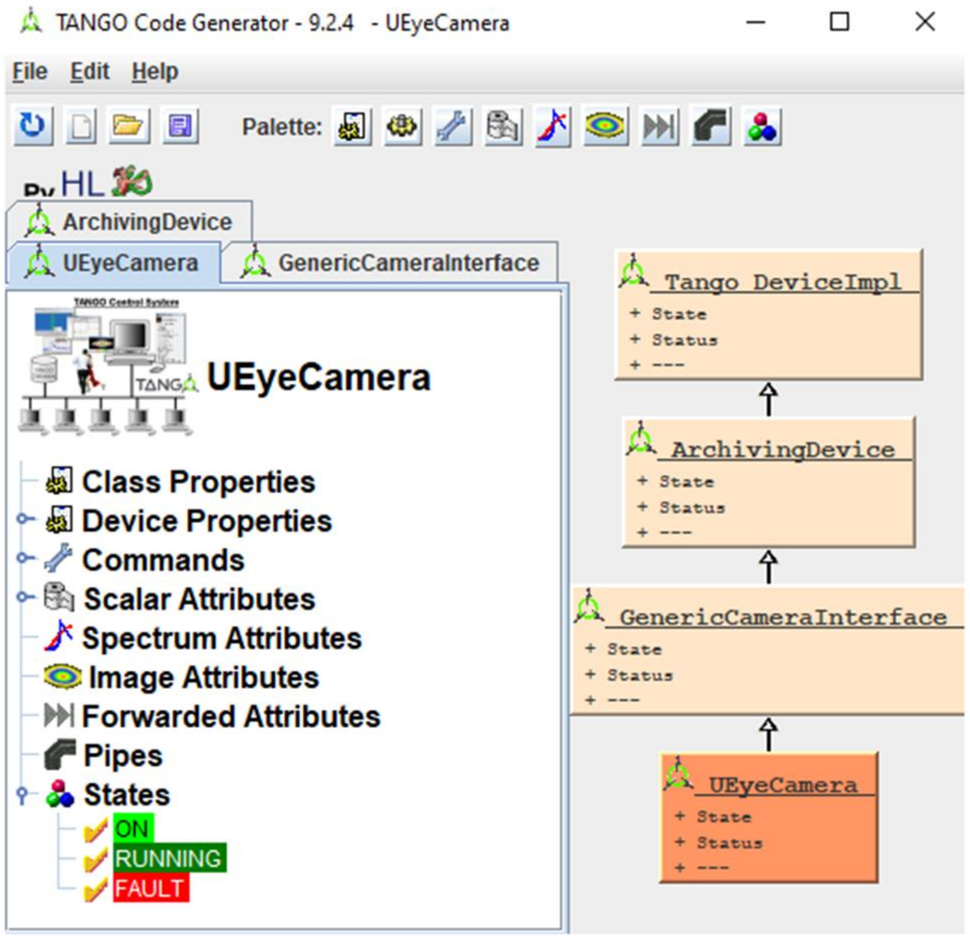
Definition of a new Tango Device

1. Setup a Tango Database Server
2. Install Tango in your favorite language
3. Check if somebody already implemented your device
4. Create a new Device class in Pogo
5. Register the device class to the database server
6. Generate Device Interface Code Skeleton from your device class definition
7. Implement your new device in your language of choice
8. Register your device to the server
9. Start your device application

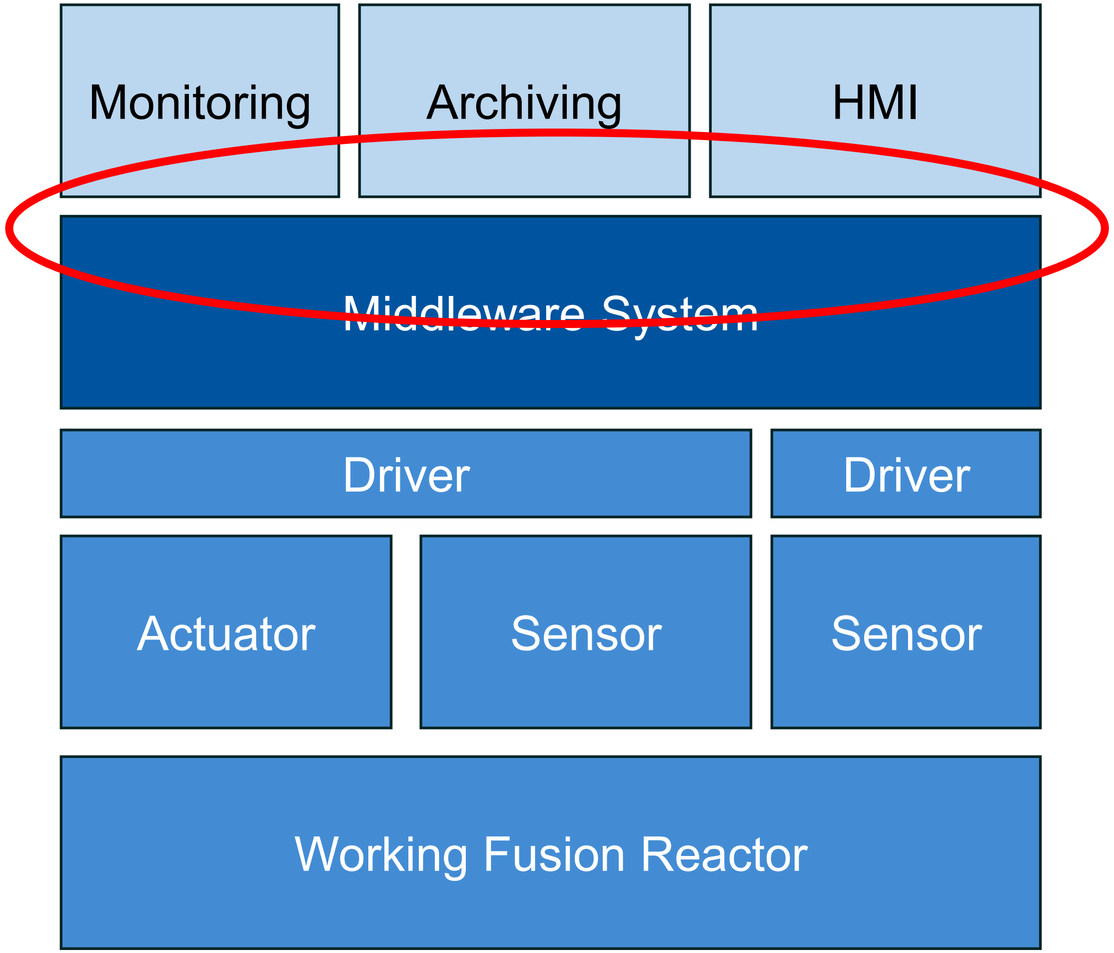
Workflow Comparison – Tango

Definition of a new Tango Device

1. Setup a Tango Database Server
2. Install Tango in your favorite language
3. Check if somebody already implemented your device
4. Create a new Device class in Pogo
5. Register the device class to the database server
6. Generate Device Interface Code Skeleton from your device class definition
7. Implement your new device in your language of choice
8. Register your device to the server
9. Start your device application



Tango focuses more on the client facing part of the middleware. Application is responsible for driver management



Workflow Comparison

Tango

Object oriented approach:

- Defining your Class (Attributes, Properties, Commands)
- Generate device classes for your specific language
- Implement specific logic

Defined static interface

Epics

Implicit device definition:

- IOC‘s are represented by the set of PV‘s it has.
- PV‘s can be added on the fly
- Datastructure is defined in the code itself
- No central „schema registry“ but a set of common types

Flexible self describing interface

Framework Comparison

	Tango	Epics	OPC/UA	ROS2
Schema Definition	Yes - Device Class Model stored in Tango DB + POGO files	No -Structures defined locally	Yes - Global Information Model defined via NodeSet XML	Yes - Message/service definitions in .msg / .srv / .action files
Central Schema Registry	Yes	No	No	No
Auto Discovery	No	Yes	Yes	Yes
Schema Definition Method	IDL + POGO model (fixed type system)	Programmatic definition	NodeSet XML	Message files (.msg, .srv, .action) compiled into code
Code Generation	Yes	No	Yes	Yes

And how does the cloud handle it?

Dynamic Interface

- CNCF
  - OpenTelemetry
    - Used for logs, message tracing and metrics
  - NATS
    - Only sends binary blobs
  - GRPC + Protobuf
    - Only with self reflection

- Apache
  - Kafka
    - Comparable to NATS
  - AMQP

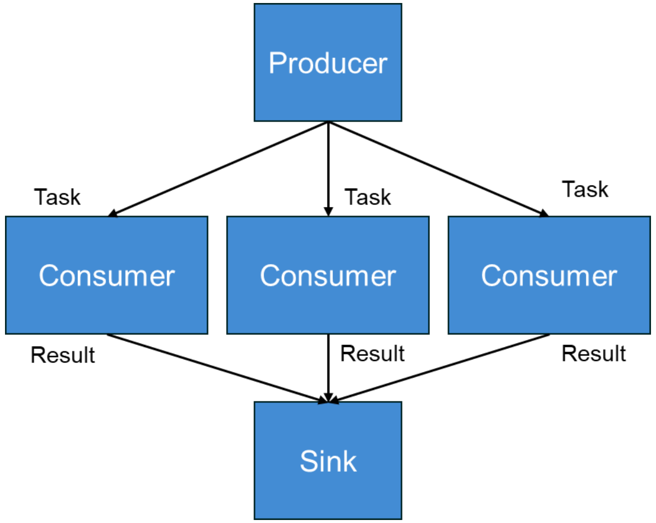
Static Interface

- CNCF
  - GRPC + Protobuf
- Apache
  - Thrift
  - Kafka (with registry)
  - CXF

Data pipelining (ML or Simulation Integration)

Pipeline for large scale batch or stream analytics are not included

- Built for **real-time control**
- Provide **live data**, not large-scale storage or history
- Lack **ML/analytics tooling** and ecosystems
- Don’t scale like **distributed data platforms**
- Mixing analytics with control risks **latency + safety issues**



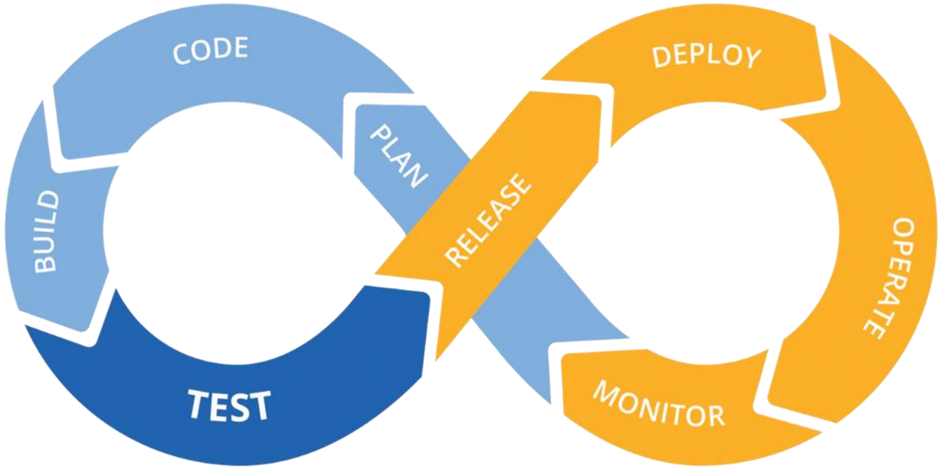
Strong Cloud Contenders

- Apache Kafka
- Apache Flink
- Apache Spark
- Apache Parquet
- CNCF Kubeflow

Deployment and CI/CD

Tango and Epics have no clear way of CI/CD

- Especially Epics is run on a lot of legacy and specialized hardware.
- No best practices for CI/CD in documentation



Strong Cloud Contenders

- Docker/Podman etc...
- CNCF Kubernetes
- CNCF Istio/Linkerd
- CNCF Akri
- CNCF Open Telemetry (Tango already includes this)
- CNCF Prometheus

Security

Authentication and Authorization

Epics and Tango both have internal mechanisms for user authentication and authorization.
Mostly realies on other mechanisms like VPN.

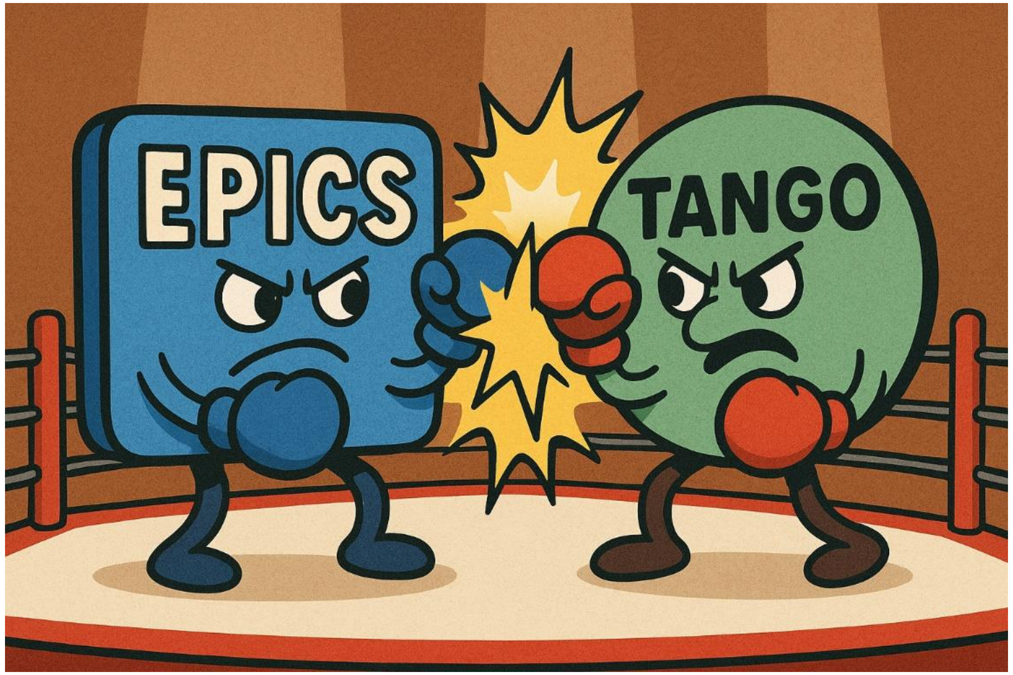
Interesting Cloud Technologies:

- Keycloak and Dex for Identity Managemant
- OPA for fine-grained authorization
- SPIFFE/SPIRE for Zero Trust Networks

Summary

Epics vs Tango

- Both Tango and EPICS allow the use of Request/Reply and Pub/Sub in order to exchange State, Parameters and run Functions.
- EPICS is based on a very liberate self organizing structure. PV‘s can be added and removed on demand
- Tango runs a very object oriented approach. Schemas are designed and organized in a central location
- Both tremendously effect the way you develop software
- Integration of Data Analytics and Simulation Models can be accelerated by the use of cloud domain software systems



Read recommendations

Zero MQ The Guide

Designing Data-Intensive Applications [Book]

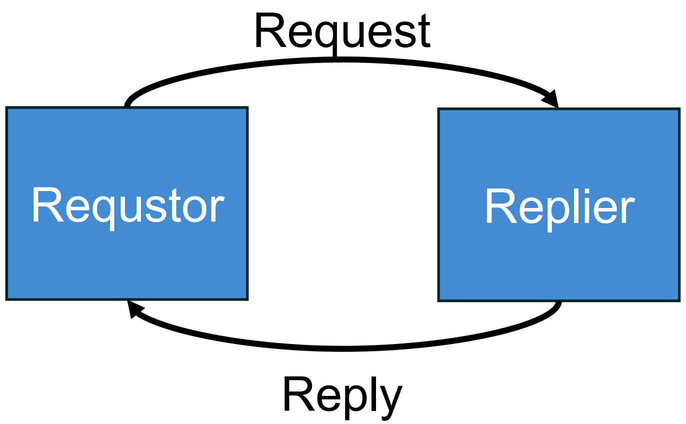
Tango vs EPICS technical comparison ELI Beamlines (2014)

Thank You
For Your Attention

Moritz Kröger
RWTH Aachen University
Lehrstuhl für Lasertechnik LLT
Steinbachstraße 15
52074 Aachen
Telefon: + 49 241 8040417
moritz.kroeger@llt.rwth-aachen.de

Basic of Communication Protocols

Request Reply Pattern

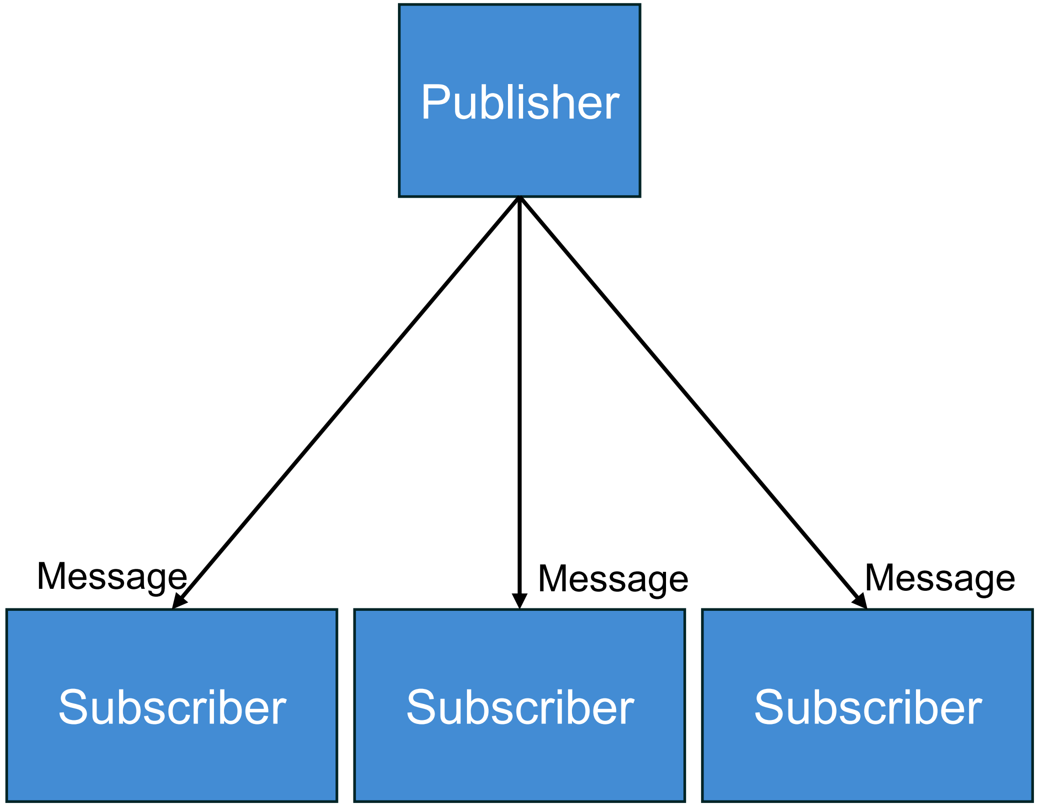


- Client sends to request queue/topic; service consumes and replies
- Requestor get confirmation of execution
- Can be synchronous (blocking) or asynchronous (non-blocking)
- To many requests can overload the replier

- Typical Usage: Webserver, RPC, Health Checks

Basic of Communication Protocols

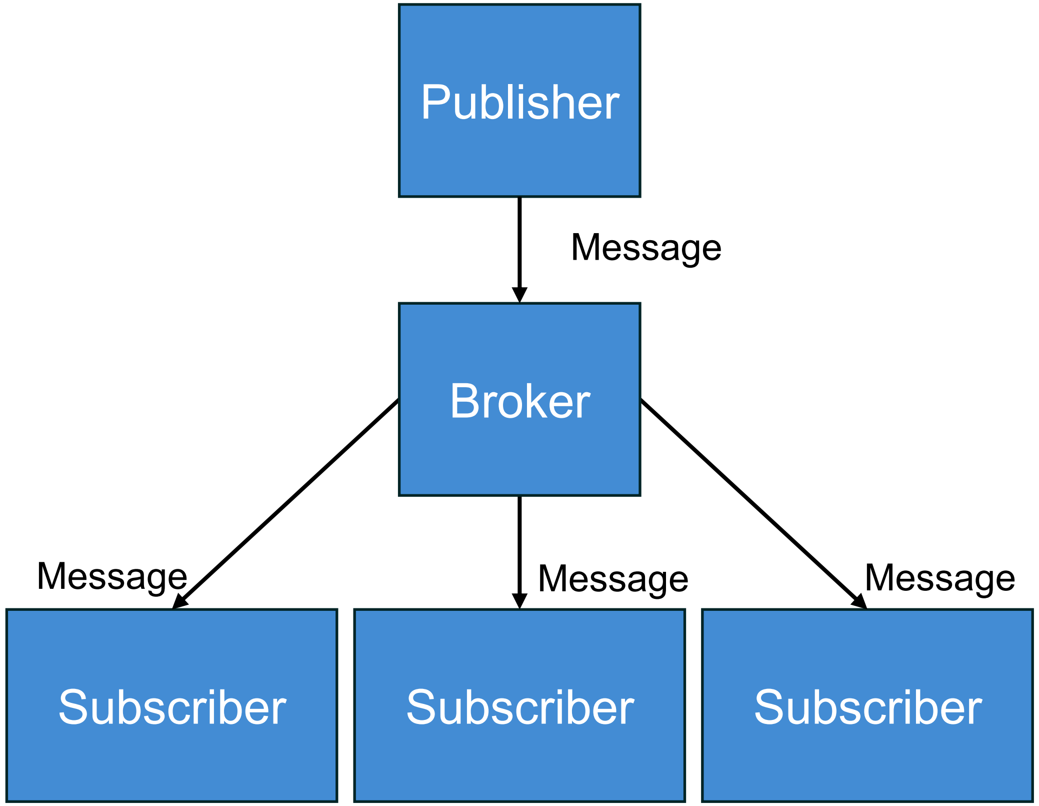
Publish Subscribe Pattern



- Publisher sends message to a broker
- Broker is responsible for delivering messages to subscribers
- Fire and Forget pattern. Publisher does not care about delivery

- Typical Usage: Telemetry, Logs, Notifications, Event distribution

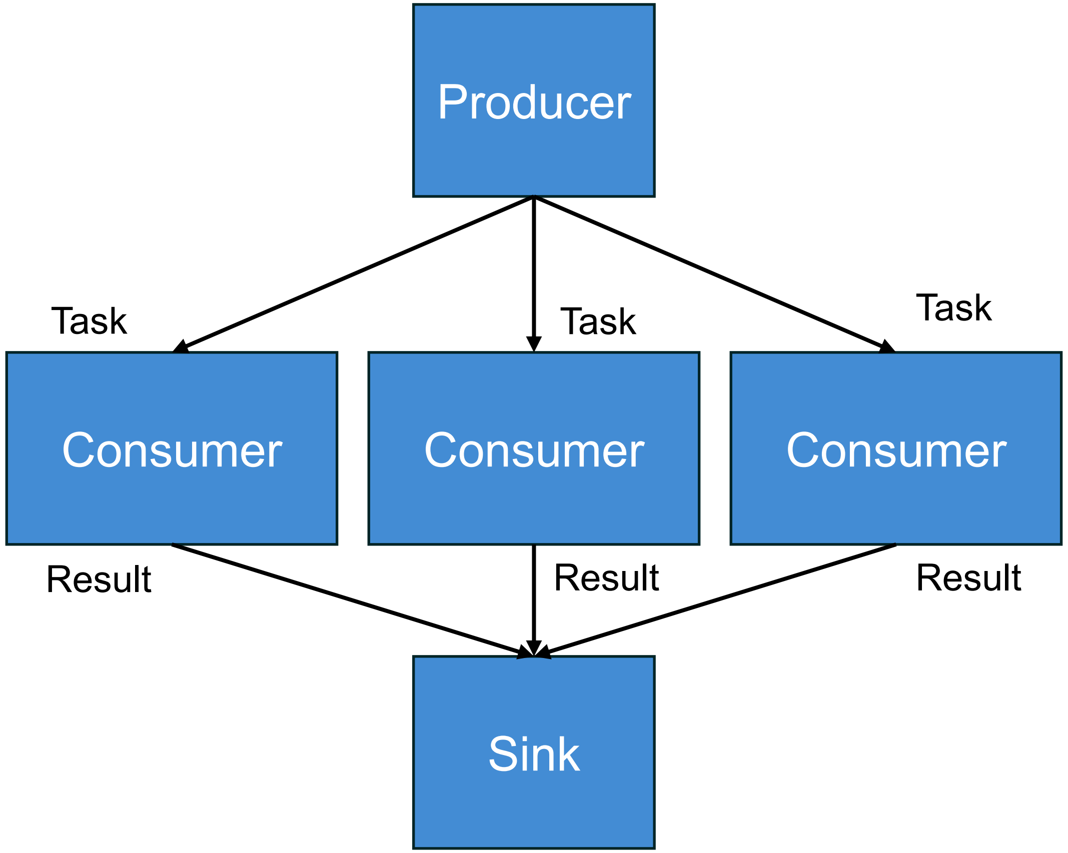
Publish Subscribe Pattern



- Publisher sends message to a broker
- Broker is responsible for delivering messages to subscribers
- Fire and Forget pattern. Publisher does not care about delivery

- Typical Usage: Telemetry, Logs, Notifications, Event distribution

Pipeline Pattern (Push Pull Pattern)

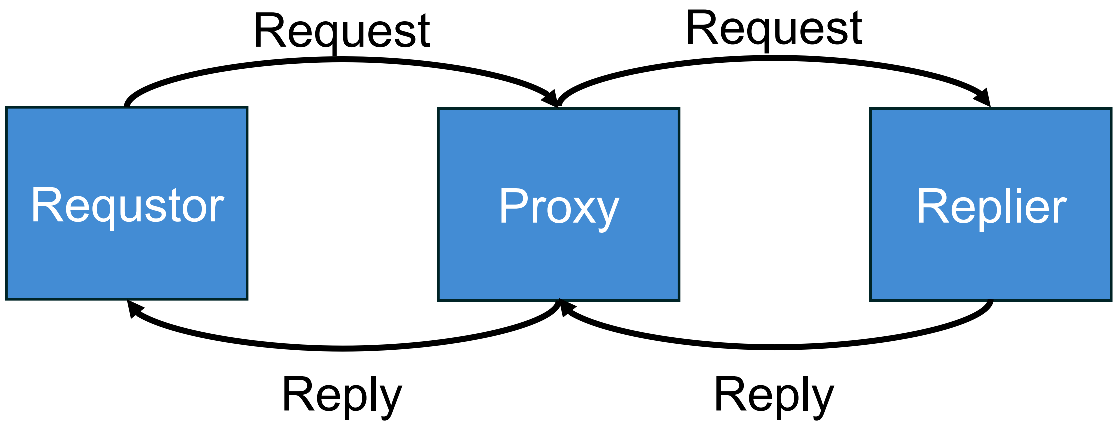


- Publisher sends message to a broker
- Broker is responsible for delivering messages to subscribers
- Fire and Forget pattern. Publisher does not care about delivery

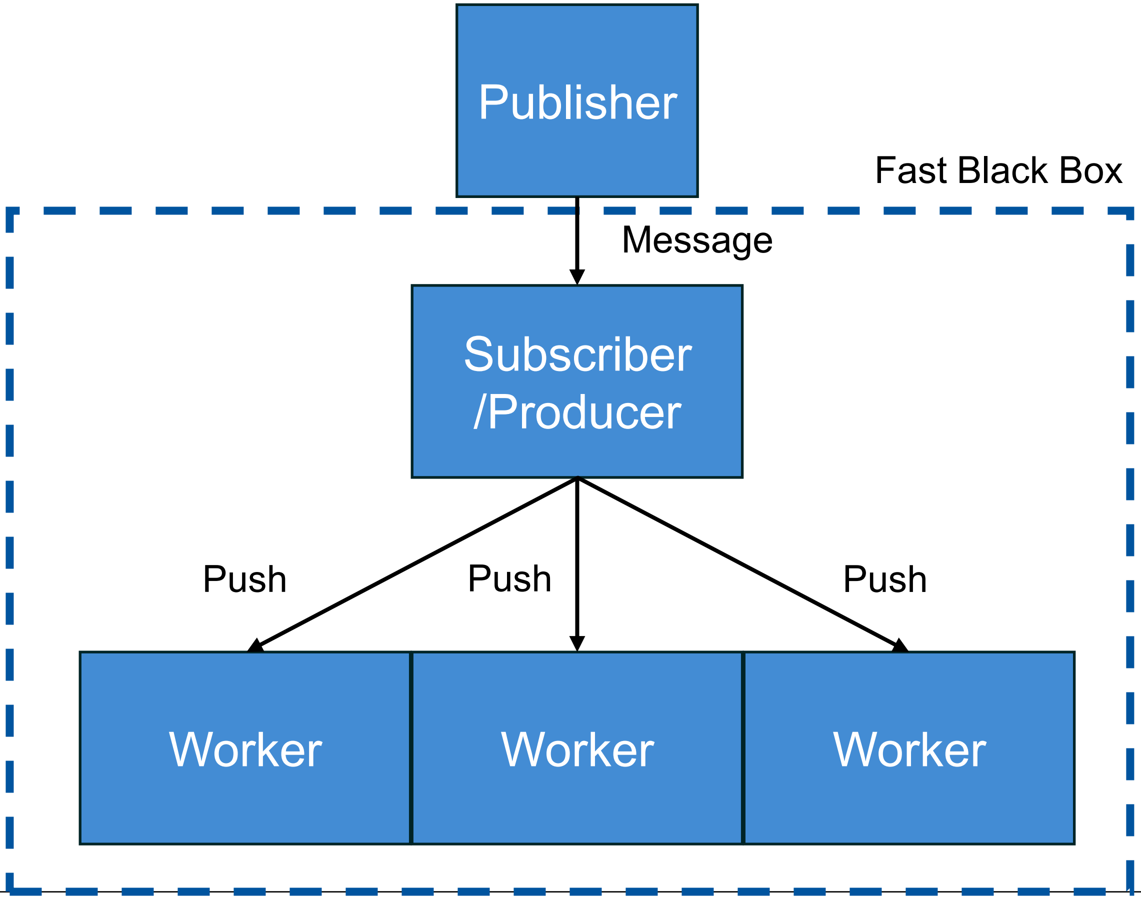
- Typical Usage: Work distribution to parallel workers, Batch processing, ML Inference

Examples of Advanced Combinations of Protocols

Proxy Pattern



Black Box Pattern



Publish Subscribe Communication

Tango

- Events
  - Always generated
    - Change
    - Periodic
    - Archive
    - Alarm
  - User defined:
    - Attribute Configuration
    - Device Interface Change
    - (Pipe)

Epics PV Access

- Get PV Value
- Set PV Value
- PutGet (Put and Get Atomic)
- ChannelRPC (Run a remote function)

Publish Subscribe Communication

Domain	Project Name	Underlying Technology	Message Format
Scientific	Epics	PvAccess	PvData
Scientific	Tango	ZeroMQ	Corba/CDR
Industrial	OPC/UA	UDP/MQTT/ AMQP	UADP Messages Json Messages
Industrial	ROS	DDS	Corba/CDR

Other relevant Projects

- CNCF
  - NATS (Binary PubSub System)
- Apache
  - Kafka
  - Pulsar
  - ActiveMQ

Communication Protocols

Domain	Project Name	Underlying Technology	Message Format	Communication Type
Scientific	Epics PvAccess	TCP	PvData	Request Reply / PubSub
Scientific	Tango	CORBA (omniORB)	Corba/CDR	Request Reply / PubSub
Industrial	OPC/UA Binary	TCP/HTTP	OPC UA Binary DataEncoding / JSON	Request Reply / PubSub
Industrial	ROS	UDP	Corba/CDR	PubSub

LLT Lehrstuhl für Lasertechnik RWTH AACHEN UNIVERSITY

Defining your System

Tango

Object oriented approach:

- Defining your Class (Attributes, Properties, Commands)
- Generate device classes for your specific language
- Implement specific logic

Epics

Implicit device definition:

- IOC‘s are generated by the set of PV‘s it has.
- Datastructure is defined in the code it self
- No central „schema registry“ but a set of common types