AN UPDATE ON CAFE, A C++ CHANNEL ACCESS CLIENT LIBRARY, AND ITS SCRIPTING LANGUAGE EXTENSIONS

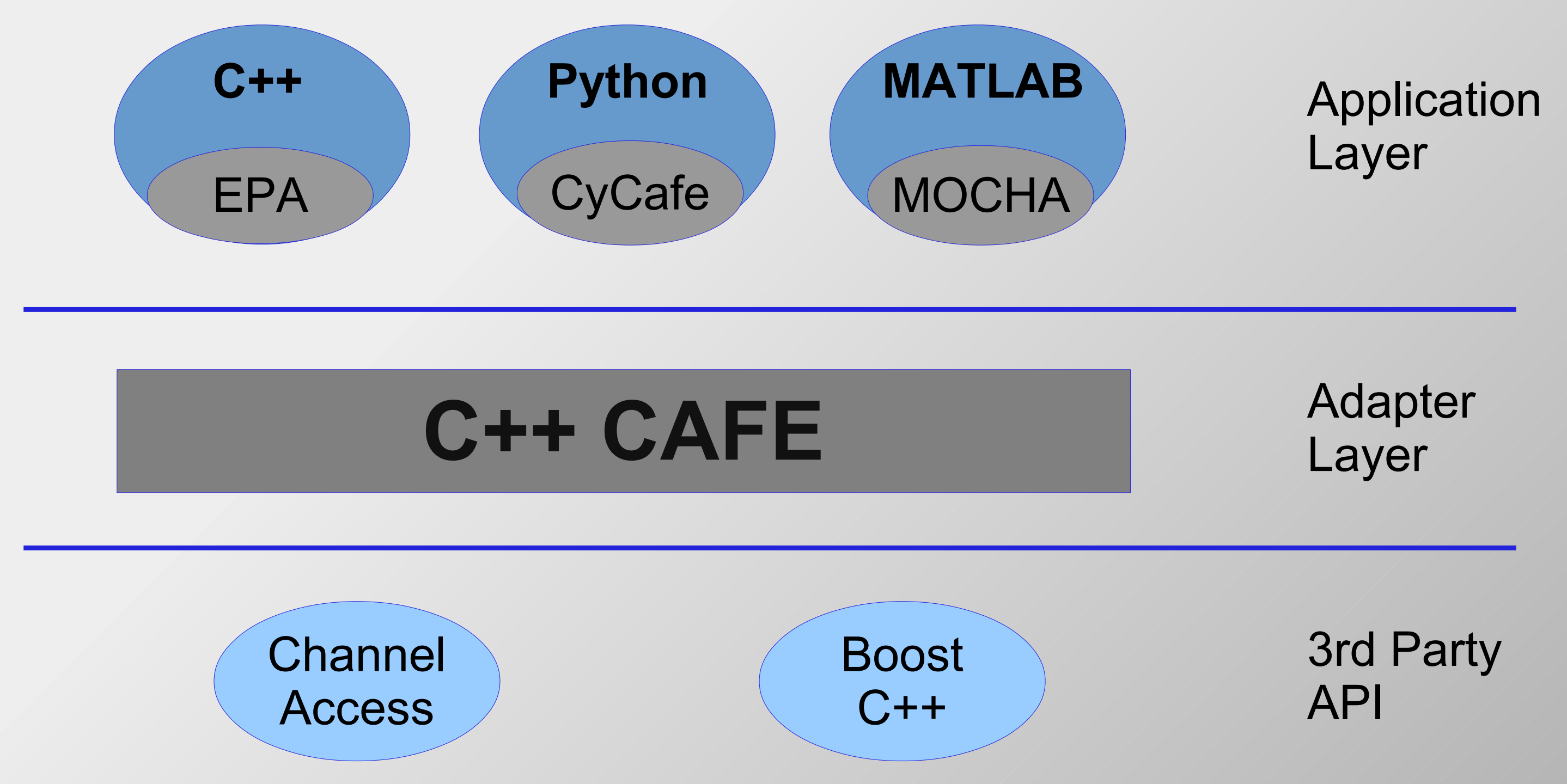
J. Chrin, Paul Scherrer Institut, 5232 Villigen PSI, Switzerland

ABSTRACT

CAFE (Channel Access interFacE) is a C++ client library that offers a comprehensive and easy-to-use interface to EPICS (Experimental Physics and Industrial Control System). Functionality is provided for the synchronous and asynchronous interaction of individual and groups of low-level control data, coupled with an abstraction layer to facilitate development of high-level applications. The code base has undergone major refactoring to make the internal structure more comprehensible and easier to interpret, and further interfaces have been implemented to increase its flexibility, in readiness to serve as the CA host in fourth-generation and scripting languages for use at the SwissFEL, Switzerland's X-ray Free-Electron Laser facility. An overview of the structure of the code is presented, together with an account of newly created bindings for the Cython programming language, which offers a major performance improvement to Python developers, and an update on the CAFE MATLAB Executable (MEX) file.

THE CAFE MODEL

The C++ Channel Access Client library serves as host to scripting and domain-specific languages, and event processing agents that aggregate and analyze data.



The Advantages

- The inherent simplicity and convenience of maintaining a single CA interface code.
- New CA functionalities from future EPICS 3 releases need only be integrated into a single base library.
- A uniform response to errors and exceptions that facilitates trace-backs.
- The CA class is well separated from the internals of the domain language meaning that bindings to other scripting and domain-specific libraries are vastly simplified.

CAFE C++ IMPLEMENTATION

- Management of client-side CA connections.
- Memory optimization, particularly when connections are restored.
- Separation of data retrieval from its presentation.
- Strategies for converting between requested and native data types.
- Caching of pertinent data related to the channel and its state.
- Aggregation of requests for enhanced performance.
- Adaptive correction procedures, e.g. for network timeouts.

The outcome of each method invocation is captured with integrity in every eventuality, ensuring reliability and stability

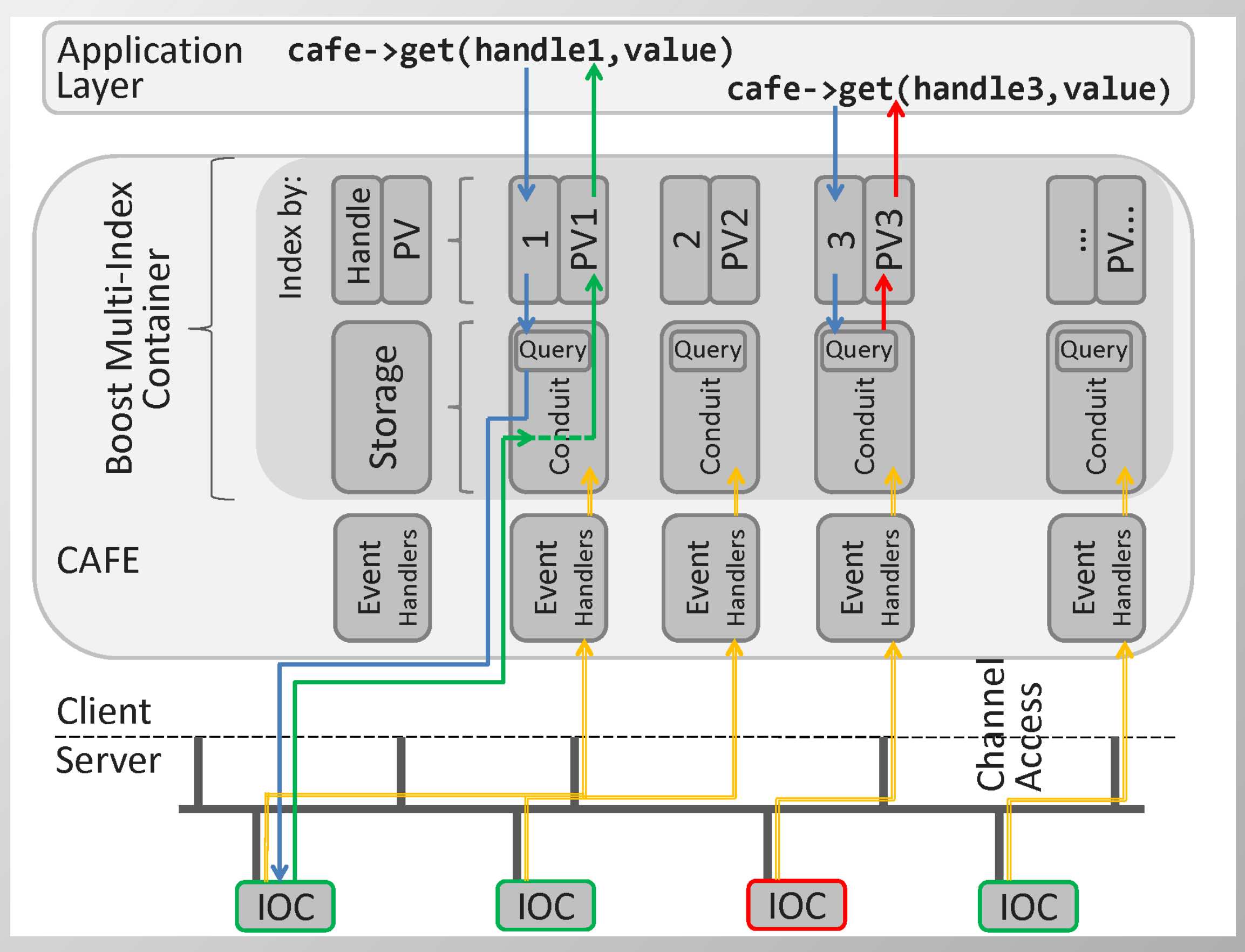


Figure: The information flow for a cafe method invocation in the case of a connected channel, PV1 (green), and a disconnected channel, PV3 (red). The multi-index container (`"Conduit" object) serves as the data store for the full complement of the PV's data, whether static or dynamic. The handle (index) is the reference to the resource's data. PV data emitted from the IOC is recorded within the container (yellow); cafe method invocations first query the container to assess whether the prerequisites for launching a message have been met.

CyCafe: Syphoning CAFE with Cython

Code Listing: PyCafe Read/Write/Monitor Examples

CyCafe is 4x faster than pure Python Channel Access clients.

Python Protocol Buffer. Data shared without copying gives a much improved performance.

Monitors made easy. No Python dictionary to interrogate.

```
import PyCafe
cafe = PyCafe.CyCafe()
cyca = PyCafe.CyCa()

#handlePV=<handle/'pvName'>
#dt=<'int','float','str','native'(default)>
#hpvList=<hList/pvList> i.e. handle/'pvName' list
#s gives overall status, sList is a status list
pvList=['pv1','pv2','pv3','pv4']
try:
  handle= cafe.open('pvName') #returns obj. ref.
  hList = cafe.open( pvList ) #returns obj. ref. list
except Exception as inst:
  print inst

#Synchronous Single Channel Operations
try:
  value = cafe.get(handlePV) #get value in native type
  PvData= cafe.getPV(handlePV,dt='float') #struct
  cafe.set(handlePV, pvData.value+0.001 )
  #waveform, return list in native type
  valList = cafe.getList (handlePV)
  #waveform, return memoryview of floats
  memview = cafe.getArray(handlePV,dt='float')
  #waveform, return numpy.ndarray in native type
  npArray = cafe.getArray(handlePV,asnumpy=True)
  #set waveform; input [values] may be any of
  #list, memoryview, numpy.ndarray, array.array
  cafe.set(handlePV,[values])
  pvCtrl = cafe.getCtrl(handlePV) #Get cached ctrl data
except Exception as inst:
  print inst

#Synchronous Multiple Channel Operations
valList,status = cafe.getScalarList(hpvList)
status,statusList = cafe.setScalarList(hpvList, valList)

#Asynchronous Single/Multiple Channel Operations
status,statusList = cafe.getAsyn(hList)
status,statusList = cafe.waitForBundledEvents(hList)
pvData  = getPVCache(hList[0])

#Synchronous Groups
#gHandleName=<groupHandle/'groupName'>
status = cafe.defineGroup('groupName', pvList)
gHandle = cafe.openGroup('groupName')
valList,status,statusList = cafe.getGroup (gHandleName)
status,statusList = cafe.setGroup (gHandleName, valList)
#returns list of structured data
pvgList = cafe.getPVGroup(gHandleName,dt='str')
cafe.terminate() #close cafe

#Monitors and Callback Functions
def py_callback(handle):
  #Any method that retrieves data from cache
  pvData = cafe.getPVCache(handle)
  return

#Start Monitor
monID=cafe.monitorStart(handle, cb=py_callback)
```

Presented at the 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15), Melbourne, Australia, Oct. 2015, paper WEPGF132

ICALEPCS 2015
International Conference on Accelerator & Large Experimental Physics Control Systems
17-23 OCTOBER 2015 MCEC MELBOURNE