

CCD Image Server

Brian Tieman

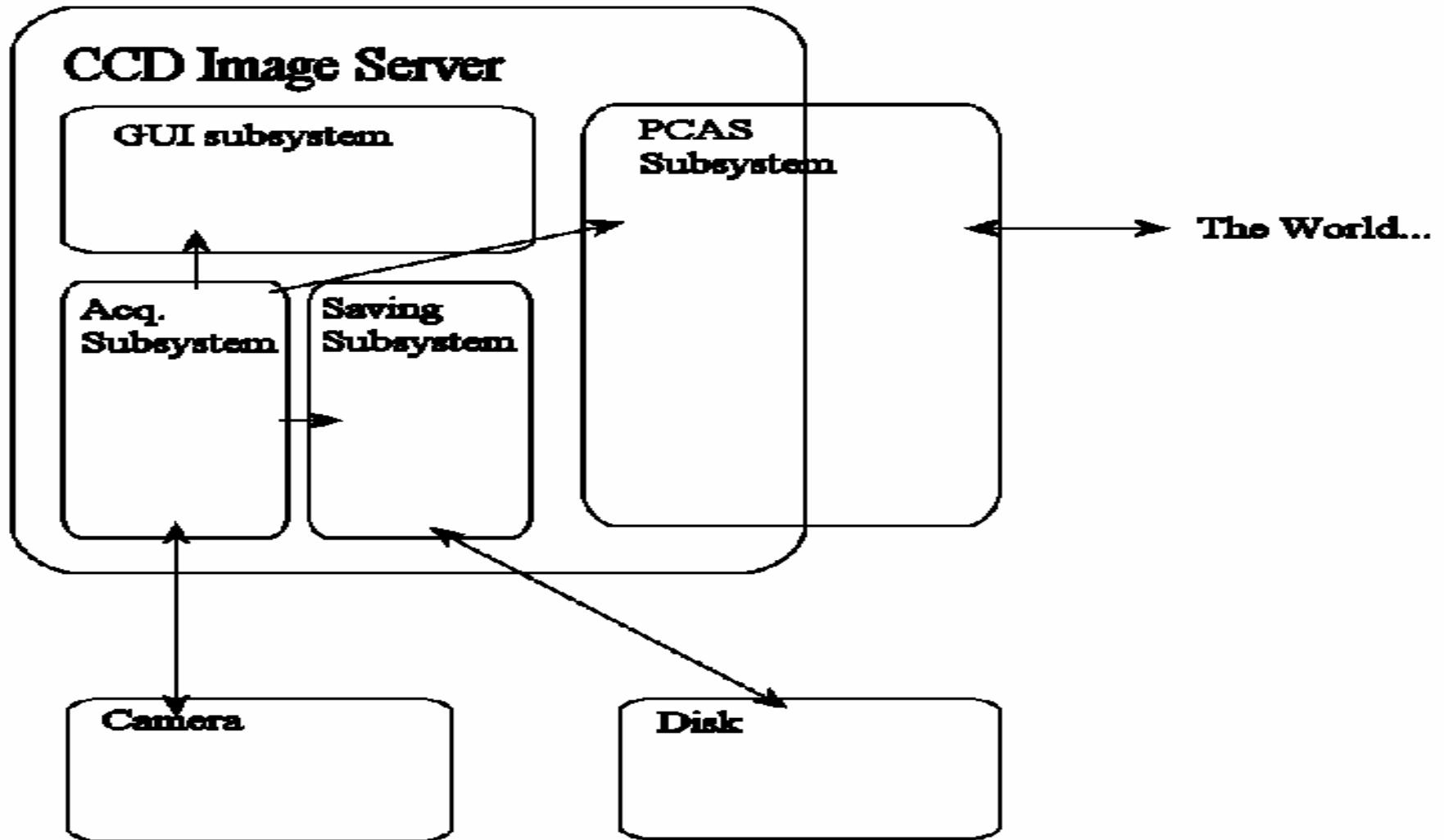
History

- Started in 1997
 - Tried IDL first
 - 2IDB beamline and Grand Challenge
- PCAS
 - Needed EPICS Server capabilities
 - Most cameras were PCI wintel only
- FEL
 - Very remote cameras...
 - ...and lots of them!
 - Resulted in FEL Image Server
- Tomography
 - 1st PCAS version of Image Server in 1998
 - Oriented specifically towards tomography (Grand Challenge)
 - Not very extensible
 - Only Princeton Instruments cameras
 - Only HDF file format
- Other users
 - Integration...
 - New cameras
 - New file formats
 - New ROI types
 - New GUI features
 - Etc...

Today

- Supports 8 different camera interfaces
- Supports 4 ROI types
- Supports 3 file systems
- Very robust
- Used by
 - FEL/SPIRIT
 - Topology Lab
 - Sectors
 - 1
 - 2
 - 4
 - 7
 - 8
 - 33
 - 34

Block Diagram



Two Interesting Subsystems

- File Subsystem
 - HDF and meta-data
 - High performance buffering
- EPICS Subsystem
 - PCAS implementation

HDF Files and Meta-Data

- Template Files
 - PVs
 - Internal Values
 - Constants
- File compression

Sample Template

- ;This is a template file to save a bare minimum of info.
- <Facility:facility>
 - [facility_name,CONST,"APS",NX_CHAR]
 - [facility_sector,CONST,"XOR",NX_CHAR]
 - [facility_beamline,CONST,"2ID",NX_CHAR]
 - [facility_station,CONST,"B",NX_CHAR]
- <Facility:End>
- <NXentry:entry1>
 - <NXdata:data>
 - [data,VAR,"ROIData",NX_UINT16]
 - {signal,CONST,"1",NX_INT32}
 - [Sample_X,PV,"2idb:m13.VAL",NX_FLOAT]
 - [Sample_Y,PV,"2idb:m14.VAL",NX_FLOAT]
 - [Sample_Z,PV,"2idb:m15.VAL",NX_FLOAT]
 - <NXdata:End>
- [program_name,VAR,"ProgramName",NX_CHAR]
 - {program_version,VAR,"ProgramVersion",NX_CHAR}
 - [program_author,VAR,"ProgramAuthor",NX_CHAR]
 - [start_time,VAR,"AcquisitionTime",NX_CHAR]
- <NXentry:End>
- <EndTemplate>
- ;End of file

High Performance Circular Buffering

- User specified size FIFO buffer to disk
 - Images enter FIFO at acquisition speed
 - Dumped to disk at file write speed
 - The two are asynchronous until FIFO full
- Nearly 5x faster!
- Drawbacks
 - No meta-data...yet...
 - May require long “downtime” at end of experiment to “catch up”

Simple Server API

- C callable wrapper around PCAS
- Only a handful of functions to support EPICS serving of PVs
- Supports
 - Nearly all DBR types
 - Synchronous/Asynchronous PVs
 - Callbacks
- Looks exactly like an IOC Server...well almost...

API Header File

- `#ifndef CASInterface_H`
- `#define CASInterface_H`
- `#include "pv_info.h"`
- `extern "C" int __declspec(dllexport) __cdecl CreateServer (char *ioc_name, char *prefix, unsigned alias_count);`
- `extern "C" int __declspec(dllexport) __cdecl StartServer (void);`
- `extern "C" int __declspec(dllexport) __cdecl StopServer (void);`
- `extern "C" int __declspec(dllexport) __cdecl DestroyServer (void);`
- `extern "C" int __declspec(dllexport) __cdecl CreatePV (PV_INFO *parms);`
- `extern "C" int __declspec(dllexport) __cdecl ForceScan (char *device, char *name, void *data);`
- `extern "C" int __declspec(dllexport) __cdecl ForceTimestampScan (char *device, char *name, unsigned long seconds, unsigned long nano_seconds, void *data);`
- `extern "C" int __declspec(dllexport) __cdecl AsyncCompletion (char *device, char *name, void *data);`
- `#endif`

PV_INFO structure

```
• typedef struct {
•     int      async_pv;           //0 for synchronous IO, 1 for asynchronous
•     float    scan_period;        //the scan period
•     char     name[50];          //string to use in the FIELD portion of the PV
•                           device[50],          //string to use in the DEVICE portion on the PV
•                           units[50];          //String specifying the PVs units
•     float    lopr,              //low operational limit
•             hopr;              //high operational limit
•     int      io_type,           //PV type (synchronous/asynchronous)
•             scan_type,          //scan the PV at scan_period
•             num_enum_types;      //Number of enumerated types if ENUMPV
•     char     *enum_types[10];    //List of string to use for the enumerated types
•     unsigned elements;         //How many elements in the PV
•     int      native_type;       //PVs native type
•     void    *address;           //Address of application variable to use for PV
•     void    *user_value;         //A user value which will be passed to process
•     void    *preread;           //Address of a function to call at beginTransaction
•     void    *postread;           //Address of a function to call at endTransaction
•     void    *process;            //Address of a routine to call on a put command
• } PV_INFO;
```

Creating a Server

- CreateServer ("mycomputer", "bjt", 150);
- //int InstallSyncPV (char *device, char *name, int scan_type, double period, float lopr, float hopr, unsigned elements, int native_type, int numEnums, char **enum_types, char *units, void *user_val, void *data_address, void *preread, void *process)
- InstallSyncPV ("ccd", "NumExposures", FORCED_SCANNING, 0.1, 0.0, 100000.0, 1, INT32PV, 0, empty_list, "", NULL, CCD_camera>GetParameterAddress (CCD_NUM_EXPOSURES), NULL, NumberExposures_Process);
- InstallAsyncPV ("ccd", "AcquireCLBK", FORCED_SCANNING, 0.1, 0.0, 1.0, 1, ENUMPV, 2, onoff_list, "", NULL, (void *) &server_pvs.acquire_callback, NULL, AcquireCallback_Process);
- StartServer ();
- ...Do something interesting here...
-
- StopServer ();
- DestroyServer ();

Sample Process Routines

```
• //-----
• void _cdecl NumberExposures_Process (long int *new_value, void *usr_val)
• {
•     CCD_camera->PutParameter(CCD__NUM_EXPOSURES, *new_value);
•
•         return;
• }
• //-----
• void _cdecl AcquireCallback_Process (unsigned short int *new_value, void *usr_val)
• {
•     if ((*new_value) && (acquire_thread->IsImageComplete ()))
•     {
•         server_pvs.acquire_callback = *new_value;
•         CCD_Main->StartAcquisitionThread ();
•     }
•     else
•         AsyncCompletion (server_pvs.device, "AcquireCLBK", &server_pvs.acquire_callback);
•
•         return;
• }
• //-----
```

The Future

- Immediate plans
 - Add Epix frame grabber
 - Upgrade to 3.14
 - Meta-data in buffered mode
 - Numerous misc. requests
 - Whatever comes along...
- Long term speculations
 - Port to IOCCore