Workflow-Oriented Cyberinfrastructure for Sensor Data Analytics

Arcot Rajasekar¹, John Orcutt², Frank Vernon²

¹University of North Carolina, Chapel Hill
²University of California, San Diego
## Four Kinds of Big Data

<table>
<thead>
<tr>
<th>Characterization</th>
<th>Archetypal</th>
<th>Crowd-Sourced</th>
<th>Long-tail</th>
<th>Sensor Streams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Velocity</td>
<td>High</td>
<td>Bursty</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Variety</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Veracity</td>
<td>High</td>
<td>Mixed</td>
<td>Low</td>
<td>Mixed</td>
</tr>
<tr>
<td>Value</td>
<td>High</td>
<td>Ephemeral</td>
<td>Unknown</td>
<td>Huge</td>
</tr>
<tr>
<td>Findability</td>
<td>High</td>
<td>High</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Availability</td>
<td>High</td>
<td>Short-term</td>
<td>None</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Archetypal Science Projects**
- LHC, SKA, LSST

**Batcher Industry/Industry**
- Genomics, Finance

**Government**
- NASA, NOAA, DOE

**Crowd-Sourced Social Media**
- Facebook, Twitter

**Recommenders**
- Yelp, Angie, Groupon

**Web Commerce**
- Amazon, Ebay

**Long-tail Science Projects**
- Small orgs, RDM

**Personal**
- Hobbies, Citizen Science/Arts

**Government**
- Internal and unpublished

**Sensor Streams**
- Internet of Things
- Appliances, Homes

**Smart Cities**
- Energy grids, Transportation

**Health**
- Biosensors, ER, OR
Sensor Data

What is a sensor? A sensor acquires a physical parameter and converts it into a signal suitable for processing (e.g. optical, electrical, mechanical)

Sensor data have some peculiar properties:

• Highly distributed network
• Time-related
  – Continuous
• Concept of infinite stream
• Volume – small to large packets
• Velocity – slow mostly
• Variety - Disparate
• Fusion is important
• Metadata is important
• Sensor Concentrators
Sensors & DFC

• Multiple partners use sensor data
  – Marine, Seismic & Environment Science (SciON)
  – Hydrology (Hydroshare)
  – Engineering (Smart Cities)
  – Cognitive Science (TDLC)
  – Biology (CyShare)

• DFC development activities:
  – Access to sensor data
    • Access control, Authentication,…
  – Export to Standard formats
  – Archiving of sensor data
    • Reuse & Repurpose
  – Integrated Metadata & Discovery
  – Integrate into Tools & Workflows
  – Playback: Synchronized
Antelope Real Time System

- Concentrator
  - Used by multiple projects
  - High performance Object Ring Buffer
  - Multiple types of sensor
  - Stream processing
  - Network of ORBs
  - Used by UCSD SIO
DFC & Antelope

- Loosely-coupled federation
- Connection through Microservices
- Can define MSO for each orb stream
- Can be added to Workflows
- Provide access without burdening ARTS Administrators
- Implementation:
  - Reap Sensor Streams
  - Convert Formats
  - Store Streams as Files
  - Access Packets from Files
  - Push Files as Streams
  - Use Rules to Archive
DFC Antelope Microservices

• Single Packet Microservices
  – msiAntelopeGet - get a packet
  – msiAntelopePut - put a packet

• Connection Microservices
  – msiOrbOpen
  – msiOrbClose
  – msiOrbTell - redirect to an orb

• Stream-level Microservices
  – msiOrbSelect - select streams
  – msiOrbReject - reject streams
  – msiOrbPosition - position read pointer by packetid
  – msiOrbSeek - position read pointer by skipping packet
  – msiOrbAfter - seek with time

• Other Helpers
  – convertExec - format conversion
  – readLine

• Packet Low-level Access (read, write) Microservices
  – msiOrbGet - get current packet
  – msiOrbReap - get next packet
  – msiOrbReapTimeout
  – msiOrbPut - push a packet

• Packet Manipulation Microservices
  – msiOrbUnstuffPkt
  – msiFreeUnstuffPkt
  – msiOrbDecodePkt
  – msiOrbStuffPkt
  – msiOrbEncodePkt

• ARTS Heartbeat Microservices
  – msiOrbStat
  – msiOrbPing
Reaping Rules

```c
antelopRule{
    delay("<PLUSET>30s</PLUSET><EF>10m</EF>") {
        msiAddKeyVal(*KVP,"selectCriteria",*pktSelectInfo);
        msiAntelopeGet(*pktSelectInfo,*firstPktId,*lastPktId,
            *NumOfPkts,*outBufParam);
        *SColl = *Coll ++ "/" ++ *Sensor
        *SFile = *SColl ++ "/" ++ "*firstPktId" ++ "_" ++ "*lastPktId" ++ ".data";
        msiCollCreate(*SColl,"1",*STAT_1);
        msiDataObjCreate(*SFile,*Resc,*D_FD);
        msiDataObjWrite(*D_FD,*outBufParam,*WR_LN);
        msiDataObjClose(*D_FD,*STAT_2);
        msiAddKeyVal(*KVP,"firstPktId","*firstPktId");
        msiAddKeyVal(*KVP,"lastPktId","*lastPktId");
        msiAddKeyVal(*KVP,"numOfPkts","*NumOfPkts");
        msiAssociateKeyValuePairsToObj(*KVP,*SFile,"-d");
    }
}
writeLine("stdout", "Delayed Rule Launched");
}
```

```
antelopRule{
    #Get Packet
    msiOrbOpen(*orbHost,*orbParam,*orbId);
    msiOrbSelect(*orbId,*Sensor,*sresOut);
    msiOrbReap(*orbId,*pktId,*srcName,*oTime,*pktOut,*nBytes,*resOut);
    msiOrbDecodePkt(*orbId,*modeln,*srcName,*oTime,*pktOut,*nBytes,*decodeBufInOut);
    msiOrbClose(*orbId);
    #Store Packet
    *SColl = *Coll ++ "/" ++ *Sensor
    *SFile = *SColl ++ "/" ++ "waveform.data";
    msiCollCreate(*SColl,"1",*STAT_1);
    msiDataObjCreate(*SFile,*Resc,*D_FD);
    msiDataObjWrite(*D_FD,*decodeBufInOut,*WR_LN);
    msiDataObjClose(*D_FD,*STAT_2);
}
```

Continuous Reaper
Interactive Packet Ingestion

antelopeRule{
    msiAntelopePut(*orbName, *srcName, *timeStamp, *pktPayLoad);
}

input *orbName="anfdevl.ucsd.edu:demo",
    *srcName="DFC_UNC/ch/T1", *timeStamp="",
    *pktPayLoad="test 3 string"

output ruleExecOut

# get a MGENC packet from cascadia and put it in demo
# also write also in a file to compare
antelopeRule{
    # get the packet and the write into file
    msiAntelopeGet(*pktSelectInfo, *firstPktId, *lastPktId,
        *NumOfPkts, *outBufParam);
    *
    *SColl = *Coll ++ "/" ++ *Sensor
    *SFile = *SColl ++ "/" ++ "*firstPktId" ++ ":_" ++ "*lastPktId" ++ ".data";
    msiCollCreate(*SColl,"1", *STAT_1);
    msiDataObjCreate(*SFile, *Resc, *D_FD);
    msiDataObjWrite(*D_FD, *outBufParam, *WR_LN);
    msiDataObjClose(*D_FD, *STAT_2);
    # write to orb
    msiAntelopePut(*orbName, *srcName, *timeStamp, *outBufParam);
}

input *pktSelectInfo="<ORBHOST>anfexport.ucsd.edu:cascadia</ORBHOST>
<ORBSELECT>TA_J01E/MGENC/SM1</ORBSELECT>
<ORBWHICH>ORBOLDEST</ORBWHICH>
<ORBNUMOFPKTS>1</ORBNUMOFPKTS>
<ORBNUMBULKREADS>1</ORBNUMBULKREADS>
<ORBPRESENTATION>ONEPKT</ORBPRESENTATION>",
    *Resc="destRescName=anfdemoResc++++forceFlag=",
    *Coll="/rajaanf/home/rods/SensorData", *Sensor="TA_J01E_MGENC_SM1",
    *orbName="anfdevl.ucsd.edu:demo",
    *srcName="DFC_UNC/MGENC/T1", *timeStamp="",
output *outBufParam, *firstPktId, *lastPktId, *NumOfPkts, ruleExecOut
Sensor Data in DFC

- Sensor streams are stored as files in DFC:
  - Raw Orb format – buffer
  - CDL format - Common Data form Language a human-readable text representation of netCDF data
  - NC format: NetCDF Format
    - NetCDF 4 – version 4
    - HDF5 compatible
    - Use ‘ncgen’ for conversion
  - JSON – human-readable format

- Multi-type Sensor’s reaped
  - Seismic Sensor
    - 3 sensor measurement per packet
    - North, East, Vertical Movements
  - Pressure Sensor
    - 2 sensor measurement per packet
    - Barometric Pressure, Infrasound
Sample Data Files

netcdf barometric_pressure {

types:
  compound pressure_vector_t {
    double timestamp;
    float pressure;
    float infrasound;
  }; // barometric_vector_t

dimensions:
  time = UNLIMITED;

variables:
  pressure_vector_t barometric(time);
  barometric:standard_name = "two vector barometric pressure data";
  barometric:long_name = "Barometric";

// global attributes:
  :srcname = "TA_003E/MGENC/EP1";
  :packettype = "waveform";
  :net = "TA";
  :sta = "003E";
  :chan = "LDO";
  :loc = "EP";
  :sampratepersec = "1.000";
  :calib = "1.000";
  :calper = "1.000";
  :segtype = "5s";
  :nsamps = "120";
  :epochtime = "1446064294.9710000";
  :epochendtime = "20:33:34.97100";

data:
  barometric =
    {1446064294.9710000, 717022, 10159},
    {1446064295.9710000, 717021, 8821},
    {1446064296.9710000, 717023, 15918},
    {1446064297.9710000, 717026, 21402},
    {1446064298.9710000, 717028, 27201},
    {1446064299.9710000, 717029, 32768},
    {1446064300.9710000, 717031, 38432},
    {1446064301.9710000, 717033, 44104},
    {1446064302.9710000, 717035, 50113},
    {1446064303.9710000, 717037, 56122}.

CDL Format
Pressure Data

JSON Format
Seismic Data
Formats
Types of Operation Supported

- Reap 8 Packets as buffers
- Low level Reap
- Archive One Packet in JSON Format
- Archive Multiple Packet in NetCDF CDL Format
- Archive Pressure Data in NetCDF CDL Format
- Convert CDL to NC Format
- Ingest Character Packet
- Access Ingested Packet
- Orb2Orb Copy of Seismic Waveform Packet
- Access NetCDF File from DFC using Cloud Browser
- Show Plots Using HDFViewer
Real-time Sensor Data from ORBs
Conclusion

- **Real-time Access to Sensor Data**
  - Not just archiving
- **Access to any sensor that is available through an ORB**
  - No need for registration
- **Control sensor data flow**
  - Select Sensor
    - From multi-sensor packet streams
  - Sub sample
    - For high frequency data
    - Eg. Send only one value per second
  - Stretch data flow
    - From multi-value packets
    - Eg. 60 per second values in single packet
- **Provision Access through the web**
  - Using websocketd