Developing A Universal Radio Astronomy Backend

Dr. Ewan Barr,
MPIfR Backend Development Group
Overview

- Why is it needed?
- What should it do?
- Key concepts and technologies
- Case studies:
  - MeerKAT FBF and APSUSE instruments
  - EDD, TNRT and SKA Prototype systems
Why is a URAB needed?

- **Adaptability:**
  - Rapid development and deployment of new (and old) processing algorithms
  - Adapt to meet new needs (e.g. real-time transient detection using ML)

- **Commensality:**
  - The ability to observe for multiple distinct disciplines simultaneously

- **Simplification:**
  - The move from custom hardware to COTS-based systems opens the developer pool lowering the cost of development
What should it do?

- Satisfy basic and not so basic telescope processing needs:
  - Digitisation, channelisation (Hz - MHz), Stokes detection, integration
  - VLBI, pulsar timing, real-time transient search
  - Dumpable voltage buffers, online RFI flagging (e.g. SK)
- Produce standard-format science-ready outputs (FITS, VDIF, FIL, etc.)
- Run out-of-the box (no installation necessary)
- Provide rich feedback to operators and astronomers
Physical view
Functional view
Requirements

- High-performance
- Flexible data transport
- Fast disk I/O
- Reproducibility
- Monitoring & Reporting
- Hardware agnostic(ish)
- Standardised interfaces
GPUs are the preferred due to flexibility, cost and FP32 performance.

PCle mounted FPGAs are interesting possibility due to native 100 GbE support (e.g. Nallatech 520N).
Flexible data transport

- Ethernet data backbone
- Data streams split across multiple groups
- Low traffic per group (6 Gb/s)
- Highly scalable
- Self load balancing
Network data: **VDIF, SPEAD**

Control interface: **Tango, KATCP**

Application data: **PSRDADA, HASHPIPE**

Metadata: **KATCP, redis, etcd**
Processing nodes are also storage nodes

Performance increases with number of spindles

Infiniband or Ethernet (w/ RoCE) interconnect

Cheap storage nodes can be added to improve performance

Fast disk I/O
Hardware agnostic(ish) + Reproducibility

- Containerisation
- Resource virtualisation
- Version control
- Environment control
Unified logging: **Elasticsearch, Logstash, Kibana**

Hardware monitoring: **Grafana, Collectd, Prometheus, Heapster**

Everything run as services using **Kubernetes**

Application monitoring: ???
Case study I: MeerKAT
MeerKAT CBF switch
FBFUSE
Multi-beam beamformer

APSUSE
Binary pulsar search

TUSE
Fast transient search

S-band Digitiser/
Packetiser

BLUSE
SETI

PTUSE
Pulsar timing
System Design:
  B. Klein, I. Kraemer, S. Hochgürtel
MeerKAT interfacing:
  C. Connot, E. Nussbaum
Controlling:
  A. Bell
Basic Concept by G. Knittel
# FBFUSE Cluster Specifications

## Hardware – Compute Node (32x)

- **Huawei FusionServer 2288H V5**
- 2x Xeon Gold 6134
- 2x 40 GbE NIC
- 384 GB RAM (transient buffer)
- 2x GTX 1080 Ti

## Performance

- **1 Petaop**

## Transient buffer

- **12 TB (~50 seconds)**
FBFUSE - BENCHMARKING

- DP4A support gives 4x performance over f32 (4 Tflops to 16 Tops on Titan X Pascal)

<table>
<thead>
<tr>
<th># Antennas</th>
<th>Beamformer benchmarking (Nbeams 85% real-time)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>856 MHz</td>
</tr>
<tr>
<td>4</td>
<td>2944</td>
</tr>
<tr>
<td>8</td>
<td>3008</td>
</tr>
<tr>
<td>16</td>
<td>2272</td>
</tr>
<tr>
<td>32</td>
<td>1760</td>
</tr>
<tr>
<td>64</td>
<td>960</td>
</tr>
</tbody>
</table>

Original prototype: [https://github.com/ewanbarr/beanfarmer](https://github.com/ewanbarr/beanfarmer)
Integrated version: [https://github.com/ewanbarr/psrdada_cpp](https://github.com/ewanbarr/psrdada_cpp)
<table>
<thead>
<tr>
<th>APSUSE Cluster Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware – Capture Node (8x)</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Hardware – Compute Node (60x)</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Storage volume</strong></td>
</tr>
<tr>
<td><strong>Write speed</strong></td>
</tr>
</tbody>
</table>
MKSEND / MKRECV

- Based on SPEAD2 library:
  - https://casper.berkeley.edu/wiki/SPEAD
  - https://github.com/ska-sa/spead2

- **MKRECV**: Captures SPEAD stream(s) and saves to PSRDADA buffers
- **MKSEND**: Converts PSRDADA buffer to SPEAD stream(s)
- ASCII configuration file to support arbitrary SPEAD streams
- Uses Infiniband Verbs for kernel bypass
MKSEND / MKRECV
MKRECV configuration

# Network connection
PACKET_SIZE 1500
IBV_IF 192.168.2.20
PORT 7148
MCAST_SOURCES 224.2.1.150,224.2.1.151,224.2.1.152,224.2.1.153
DADA_KEY dada
SYNC_TIME 1231235243.0000000
SAMPLE_CLOCK 1750000000.0
NTHREADS 32

# MeerKat F-Engine
NINDICES 3

# The first index item is the running timestamp
IDX1_ITEM 0
IDX1_STEP 2097152 # The difference between successive timestamps

# This second index is the F-engine ID
IDX2_ITEM 1
IDX2_LIST 0,1,2,3,5,6,7,8,9,10,11,12,13,14,15 # Antennas to capture

# The second index item is the frequency
IDX3_ITEM 2
IDX3_LIST 0,256,512,768,1024,1280 # List of frequency partitions
Case study II:
Eifelsberg Direct Digitisation Backend (EDD)

- The Effelsberg realisation of the Universal Backend concept
- Intended to support the new range of direct digitisation receivers:
  - K, C+, UBB, Q, Ka and Ku bands
- Must integrate with the telescope control systems and provide real-time feedback for pointing and focus calibration
- Intended as the prototype for TNRT and SKA Prototype dish (slightly different functionality, same framework)
Control Plane
Monitoring interface (iGUI)
Thoughts going forward

- Many risks (noise diodes!), we expect to learn a lot in the coming months.
- Future k8s updates will simplify system management across the board.
- We are happy to collaborate with any and all. Code is MIT licensed, k8s configs and cluster configs can be made available.
- Dependency chains for specific needs should be developed with back-pressure deployment.
- Need to understand lifetimes of COTS backends better (rolling replacement/upgrade, accelerator changes, etc. etc.).
- Before we needed hardware experts, now we need sys admins!