# WIDAR Correlator Backend:

## Requirements,
## Status,
## &
## Design Update



M. Pokorny

- Stage 2 acceptance testing

  Current software deployed at DRAO is sufficient.

- Stage 2 acceptance testing

  Current software deployed at DRAO is sufficient.

- Hardware/software integration testing

  Possibly the following:

  – backend pipeline startup, configuration and shutdown (backend pipeline M&C interface);
  – lag set assembly (requires configuration);
  – pipeline processing functionality: Fourier transform, data valid normalization, *etc.*; and
  – BDF output.

- OTS critical tests

  All of the above, with the following possible additions:

  - master node interface with test executor;
  - master node applications for monitor and control of backend pipeline(s);
  - auxiliary data distribution to backend pipelines (*e.g.*, state counts);
  - additional pipeline processing functionality (van Vleck correction?);
  - parallel I/O (*i.e.*, fast formatter) capability;
  - staging area for BDF files; and
  - master node interface with MCAF.

- OTS critical tests

  All of the above, with the following possible additions:

  - master node interface with test executor;
  - master node applications for monitor and control of backend pipeline(s);
  - auxiliary data distribution to backend pipelines (*e.g.*, state counts);
  - additional pipeline processing functionality (van Vleck correction?);
  - parallel I/O (*i.e.*, fast formatter) capability;
  - staging area for BDF files; and
  - master node interface with MCAF.

- Limited observing and integration with EVLA M&C

  All of the above, with the addition of

  - master node interface with MCCC;
  - additional capabilities for master node M&C of backend pipelines;
  - additional pipeline processing functionality; and
  - archive interface.

## Pipeline (compute node) software status

- Architecture/framework complete.
- M&C interface complete.
- Needs lag set processing element implementations (and pipeline data type definitions), including, in the short term,
  - FFT element,
  - normalization element,
  - integration element, and
  - BDF output element.
- Needs additional monitor properties and diagnostic capabilities.
- Needs improved logging capabilities.

## Master node software status

- Interface to backend pipelines complete.
- Needs interface to test executor and MCCC.
- Needs backend applications, such as
  - compute node resource allocation,
  - compute node/pipeline M&C (including configuration and failover),
  - auxiliary data distribution to compute nodes,
  - BDF output coordination, and
  - interface to MCAF (and archive?).

- Cluster management

  Still using Rocks with good results.

- Parallel I/O

  Still using Lustre; no critical tests done yet. Recent developments in the field include

  – pNFS, a standardized parallel filesystem interface, which will be implemented by Lustre and other parallel filesystems;
  – Cluster File Systems (primary Lustre developer) acquired by Sun Microsystems — Lustre code remains open source.

- High availability and failover

  Investigating Heartbeat from Linux-HA project.

## Design update: compute nodes

Backend (compute node) software has been redesigned and reimplemented since last face-to-face meeting.

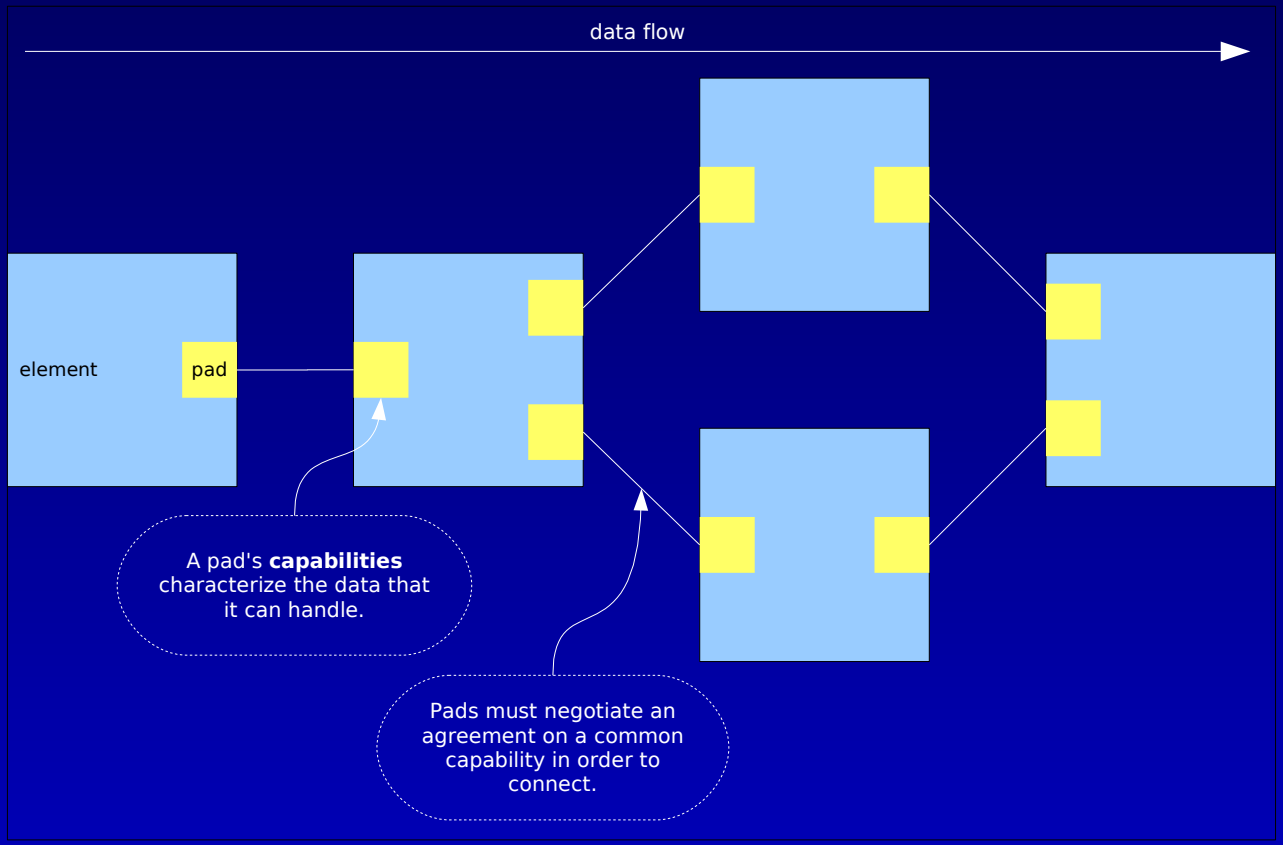
New design based on GStreamer, an open source multimedia framework. A few of the key features of GStreamer:

- construction of arbitrary pipelines based on graph structure;
- simple construction of multi-threaded pipelines;
- lightweight data passing for high performance and low latency; and
- dynamically loaded plugins to provide pipeline elements and media types.

Key concepts of GStreamer:

- element
- pad
- capabilities

# GStreamer key concepts



## Pipeline design features

- Separate input stage and lag set processing stages.
- Lag set processing stage management: startup, shutdown, message/event/error handling, process restarts.
- Supports multiple standby configurations and one active configuration.
- GStreamer-derived features: element properties, pipeline message bus, and pipeline events.
- Pipeline M&C socket interface (datagram-oriented Unix domain or UDP/IP); pipeline essentially runs as a daemon or server.
- Two client programs: *wcbe_console* and *pipelinefs*.

## Master node design

The only software design for the master node that is complete is the master node/pipeline interface, to be used by applications running on the master node to interface with compute node pipelines.

Two options were available:

1. use socket interface (*i.e.*, message protocol), or
2. hide message protocol under another layer.

## Master node design

The only software design for the master node that is complete is the master node/pipeline interface, to be used by applications running on the master node to interface with compute node pipelines.

Two options were available:
1. use socket interface (*i.e.*, message protocol), or
2. hide message protocol under another layer.

*pipelinefs* is an implementation of option 2, using a REST-like style, that is immediately usable by *any* programming language that can read from and write to files — including shell scripts — without requiring additional software.

What is it?

> *pipelinefs* exposes the internal state of pipelines as directory entries in a filesystem. States can be monitored by reading from directory entries, and can be changed by writing to directory entries.