HALO: A New System for Lustre HA Management

Thomas Bertschinger, LANL

04/29/2026

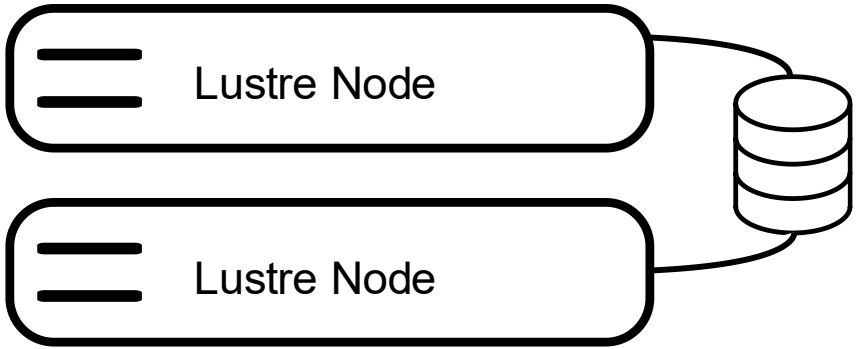
LA-UR-26-23244

Agenda

1. Review the state of the art
2. Why make a new system?
3. How HALO works
4. How you can get involved

HA in Lustre - Review

- Lustre filesystems support HA with hardware that allows multiple servers to connect to the same storage.
  - At any given time, one server can serve the storage.



HA in Lustre - Review

- A software solution is typically used to monitor the filesystem and perform migration of services between nodes in a pair.

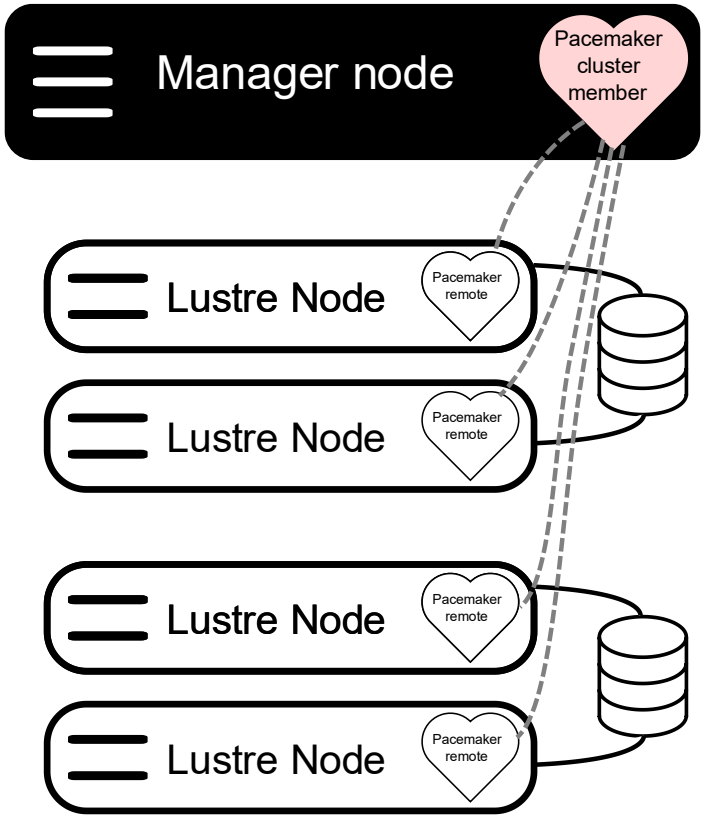
- Pacemaker is the standard software solution, recommended by the Lustre documentation at https://wiki.lustre.org/Creating_a_Framework_for_High_Availability_with_Pacemaker.

HA in Lustre - Review

- Pacemaker is commonly used in two architectures:
  - Single management node running a management service which manages the entire cluster in a single “pacemaker unit”
  - Separate “pacemaker units” for every HA pair in the filesystem

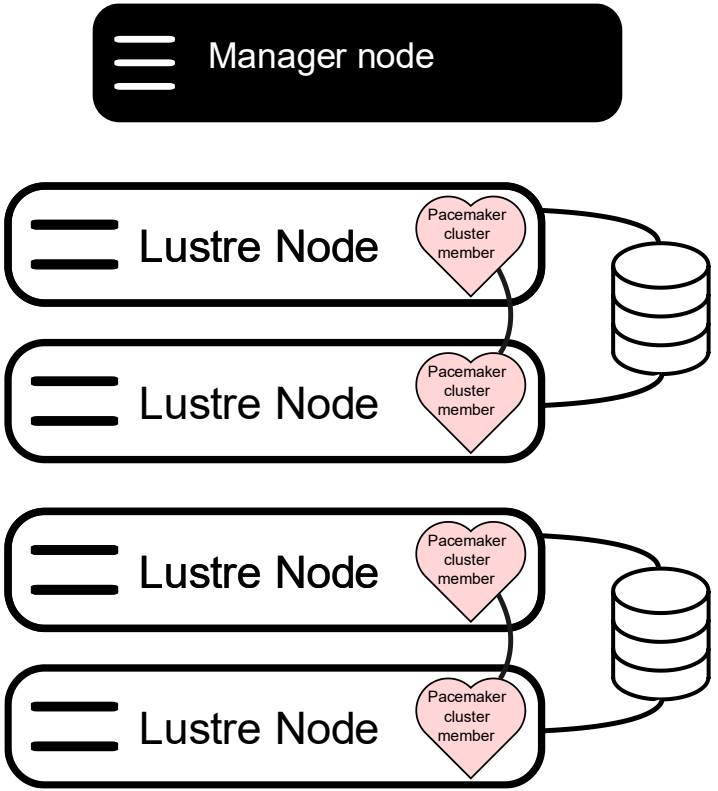
HA in Lustre - Review

- Single pacemaker unit for entire filesystem
- Pro:
  - Easier to administer: doesn't require external tooling to see the entire cluster state at once.
- Con:
  - Performance suffers, especially at large scale. Pacemaker is apparently not designed for clusters with dozens of nodes.



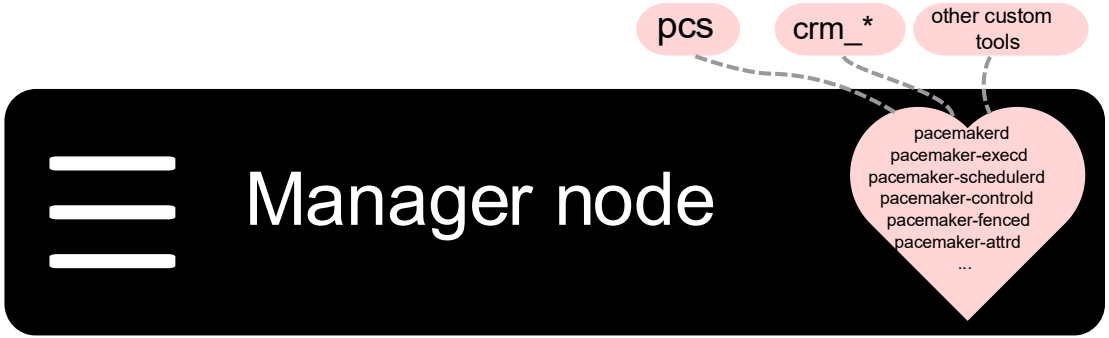
HA in Lustre - Review

- Pacemaker unit per HA pair
- Pro:
  - Doesn't run into scalability limitations as cluster size increases.
- Con:
  - No way to get an overview of the entire cluster state without external tooling / scripting.



HA in Lustre - Review

- Pacemaker usage and configuration is complex and unintuitive. New Lustre admins in particular struggle to learn Pacemaker, in my experience.
  - Multiple frontends.
  - Config file is not human-readable or human-editable.
  - Management service is made up of numerous separate daemons.



HA in Lustre - Review

- Pacemaker is opaque:
  - Getting Pacemaker to do a failover and migrate resources when desired can be surprisingly difficult.
  - It also seems to be a common experience that Pacemaker stops services and halts nodes when it shouldn’t.

HA in Lustre - Review

- The reality is that:
  - Some admins report disabling Pacemaker during outages because it is more likely to get in the way.
  - Some admins report not using Pacemaker at all, because it’s easier and more reliable to perform failovers and migrate resources by hand.

Why make a new system?

- I don’t want to be perceived as suggesting that Pacemaker is a bad system.

- Pacemaker (I presume) works well for its intended use-case, but that use-case is much different than what Lustre HA needs.

Why make a new system?

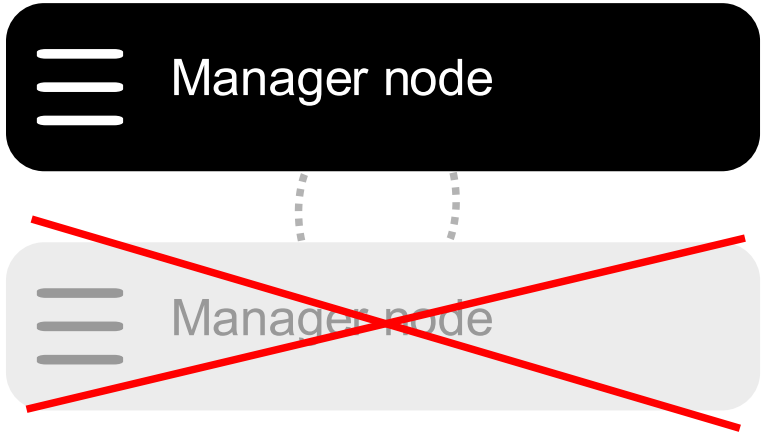
- Pacemaker is designed for a very different kind of HA system than Lustre.

- In an HA system, there are two classes of service:
  - The manage***ment*** service
  - The manag***ed*** service

- Pacemaker is designed to make both classes of service highly available. This is truly needed for some kinds of systems, like air traffic control.

Why make a new system?

- It's common to run Lustre with only a single management node.
  - Some installations use a pair of management nodes, but I don't think this provides much value.



Why make a new system?

- Isn’t having a non-redundant management service a problem?
  - In my experience, no. In my years working with Lustre, I have never experienced an outage due to the management node failing.

Why make a new system?

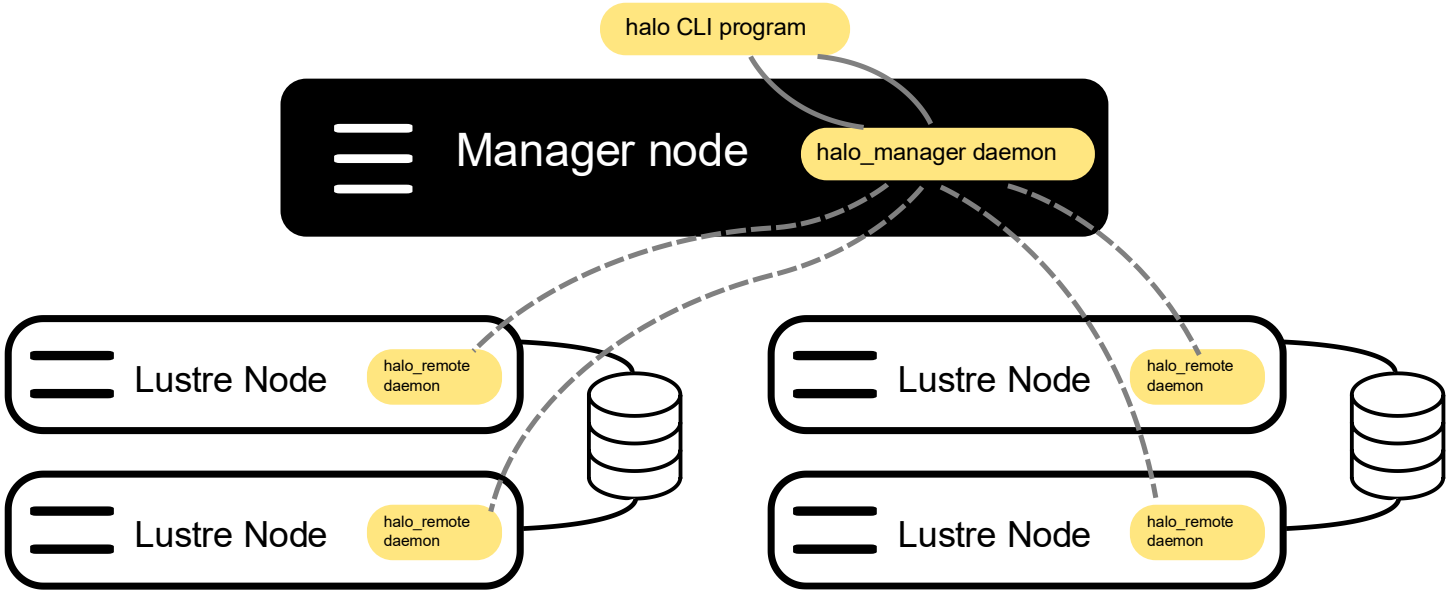
- Given the dissatisfying state of Lustre HA management currently, I think we can improve the admin experience by managing Lustre with software that is designed specifically to provide what Lustre HA needs.

How HALO works - Overview

- HALO aims to improve the admin experience for Lustre HA management by implementing a system that is:
  - Simple: no features that aren't useful for Lustre administration.
  - Transparent: easy to understand why HALO performed an action.
  - Unobtrusive: gets out of the admin's way when the admin needs to take action manually.

How HALO works - Overview

- HALO consists of three components: `halo_manager` (daemon), `halo_remote` (daemon), and `halo` (CLI utility)



How HALO works - Principles

- HALO accepts the reality that the *management* service is non-redundant in a typical Lustre installation.
  - A single management service instance runs on the single management node.
  - Not worrying about complex but irrelevant issues like quorum and distributed consensus dramatically simplifies the system.

How HALO works - Principles

- HALO decouples the management and managed services.
  - This means that the management service can be started and stopped independently from the managed services.
  - If the admin wants to stop the management service to perform a complex manual action, just run `systemctl stop halo.service`.
    - HALO will not try to halt the entire cluster. Only the management service will halt.
  - When the admin is ready for HALO to resume cluster management, just run `systemctl start halo.service`.
    - HALO detects the state of the cluster when it starts and learns about any changes in state that occurred while it was stopped.

How HALO works - Integration

- HALO integrates with existing OCF resource agent and fence agent scripts:
  - `/usr/lib/ocf/resource.d/lustre/Lustre`
  - `/usr/sbin/fence_{powerman,ipmilan,redfish}`
- If you use Pacemaker, you should already have these programs installed.

How HALO works - Configuration

- Manager config file is in yaml format:

```
hosts:
- hostname: lu-mds00
  resources:
    mgt:
      kind: lustre/Lustre
      parameters:
        mountpoint: /mnt/mgt
        target: mgt
        type: mgt
      requires: zpool_00
    zpool_00:
      kind: heartbeat/ZFS
      parameters:
        pool: zpool_00
      requires: null
  …
```

- `kind`: the path to the OCF resource agent script for this resource
- `parameters`: the parameters expected by the OCF resource agent script
- `requires`: specifies dependencies (i.e., Lustre targets depend on underlying zpool)

How HALO works - Configuration

- Fencing and failover configuration:

```
hosts:
- hostname: lu-mds00
  …
  fence_agent: powerman
  fence_parameters: null
  …


failover_pairs:
- - lu-mds00
  - lu-mds01
- - lu-oss00
  - lu-oss01
```

- `failover_pairs`: self-explanatory
- `fence_agent`: name of OCF fence agent script
- `fence_parameters`: additional parameters for fence agent script, if applicable

How HALO works - Configuration

- Create a config automatically based on an already running Lustre filesystem:

```
halo discover lu-mds[00-01],lu-oss[00-01] > /etc/halo/halo.conf
```

- The config will contain the state of currently running services.

- Fencing and failover information is not included in the generated config file; that has to be added by hand.

How HALO works – Starting the service

- In a normal installation, HALO is managed via systemd:

```
lu-mgmt$ clush -g mds,oss systemctl enable --now halo-remote.service

lu-mgmt$ systemctl enable --now halo.service
```

- When the manager service starts, it checks the status of all resources (on both nodes in the pair) and tries to start those that are running nowhere.

How HALO works - Management tasks

- Checking status:

```
lu-mgmt$ halo status
test_zpool_00   (heartbeat/ZFS):        Running on lu_mds00
test_mgt        (lustre/Lustre):        Running on lu_mds00
test_zpool_01   (heartbeat/ZFS):        Running on lu_mds01
test_mdt        (lustre/Lustre):        Running on lu_mds01
Connected hosts:        lu_mds[00-01]
Disconnected hosts:
```

- Checking status, only printing abnormal results:

```
lu-mgmt$ halo status -x
```

How HALO works - Management tasks

- Trigger a failover:

```
lu-mgmt$ halo fence lu-mds01 --reason "node is misbehaving"
```

- Status after:

```
lu-mgmt$ halo status
test_zpool_00   (heartbeat/ZFS):        Running on lu_mds00
test_mgt        (lustre/Lustre):        Running on lu_mds00
test_zpool_01   (heartbeat/ZFS):        Running (Failed Over) on lu_mds00
test_mdt        (lustre/Lustre):        Running (Failed Over) on lu_mds00
Connected hosts:        lu_mds00
Disconnected hosts:     lu_mds01
Events:
2026-04-15 15:40:45.186271 event=fence object=lu_mds01 comment="node is misbehaving"
```

How HALO works - Management tasks

- To prevent excessive reboots in case of an unhealthy node, once a node has been fenced HALO will not automatically fence that node again until an admin intervenes to manually reset it.

```
lu-mgmt$ lu-mgmt$ halo reset lu-mds01 --reason "confirmed node is healthy again"
```

- After being reset, now HALO can automatically fence the node again if it detects a problem.

- HALO does not automatically power on nodes. After a node has been fenced, the admin must power it on.

How HALO works - Management tasks

- Unmanage a resource:

```
lu-mgmt$ halo unmanage test_zpool_01 --reason "zpool is unhealthy; needs manual fixing"
```

- Think of this like the management service releasing a mutex on the resource, making it safe for the admin to do things like manually stopping or migrating it.

- Once the resource is healthy, HALO can resume management:

```
lu-mgmt$ halo manage test_zpool_01 --reason "zpool has been fixed"
```

How HALO works - Management tasks

- Deactivate a node to prevent HALO from running resources on it:

```
lu-mgmt$ halo deactivate lu-mds00 --reason "node needs hardware replaced"
```

- When a node is deactivated, any resources currently running there are gracefully stopped and started on the partner.

- Once the node is healthy, reactivate it to allow HALO to run resources on it again:

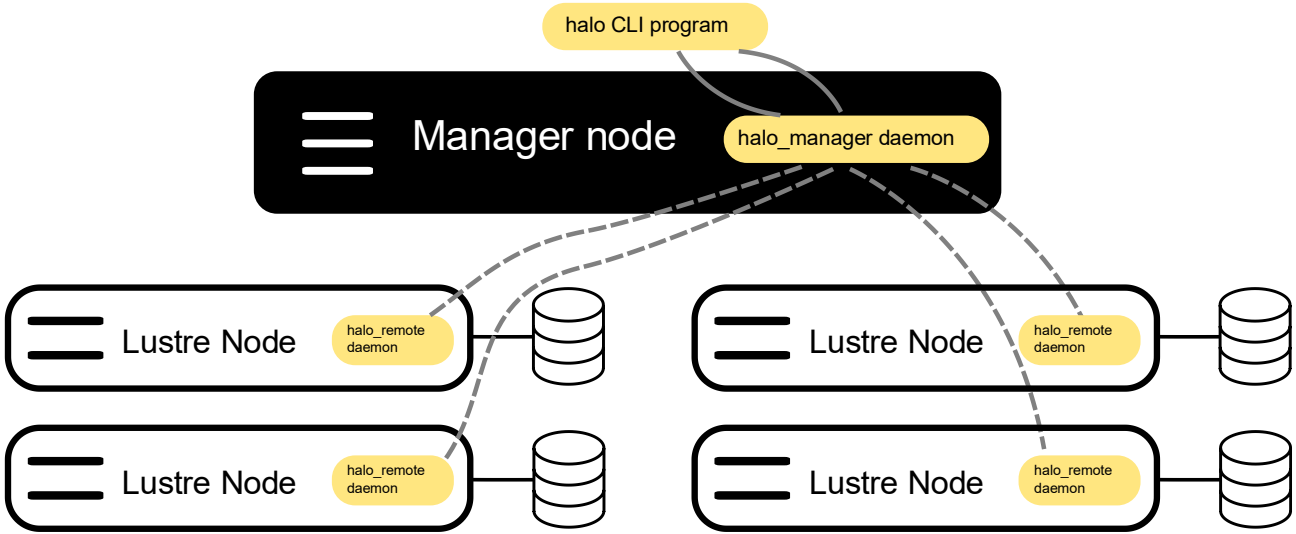
```
lu-mgmt$ halo activate lu-mds00 --reason "node has been fixed"
```

How HALO works - Persistent State

- HALO does not store cluster state persistently. It discovers it dynamically when it starts.
- HALO stores a log of events that have occurred.
- When the manager services starts, it replays the log to learn which resources are (un)managed, which hosts are (de)activated, etc.
  - It does NOT use the log to figure out where resources are running!

How HALO works - Non-HA Clusters

- HALO supports clusters without failover pairs, too.
- It won't perform failovers (obviously), but it can monitor and report on cluster status.



Interested in trying HALO—or contributing?

- HALO is written in Rust, and is open-source with an MIT license:
  - https://github.com/lanl/halo

- The source code is commented, has a test suite, and extensive admin and developer documentation is available in the repo.

- New contributors are welcome; there are plenty of "good first issues" to work on. Reach out to me if interested.

Production Readiness

- HALO in HA mode is experimental; we aren't yet using in production at LANL but hope to soon.

- HALO has a passive "observe-only" mode where it reports on cluster status but does not perform actions. It is already used in production in this capacity at LANL.

- HALO has mainly been tested with the ZFS backend. ldiskfs should work equivalently, however.

Trying HALO for yourself

- If you're adventurous and willing to be an active participant in getting HALO production ready, let me know.

- Want to try HALO but not ready to replace your existing solution?
  - The "observe-only" mode can be installed alongside your existing cluster management software, allowing you to get a feel for how HALO works.

Acknowledgements

- Thanks to my LANL management for entertaining the idea of creating an alternative HA manager for Lustre.

- Thanks to Trevor Bautista and Noah Jones of LANL for assisting with the development.

Any questions?