Revisiting the ESGF Node Manager for Federation Scalability

P. Dwarakanath, S. Ames

LIU/LLNL

December 10 2014

Contents

1. Background
2. Desirable features for next-gen Node Manager
3. Node Manager types
4. View of federated Node Managers
5. Node Manager components
6. Node Manager component diagram
7. Node Manager design considerations

Background

The current ESGF Node Manager handles:

- Capturing metrics
- Sharing node information across federations: certs, endpoints etc
- A mechanism to share common configuration files.

Drawbacks

- Limited scalability.
- P2P file/data exchange could be more secure, particularly configuration files.

Desirable features for next-gen Node Manager

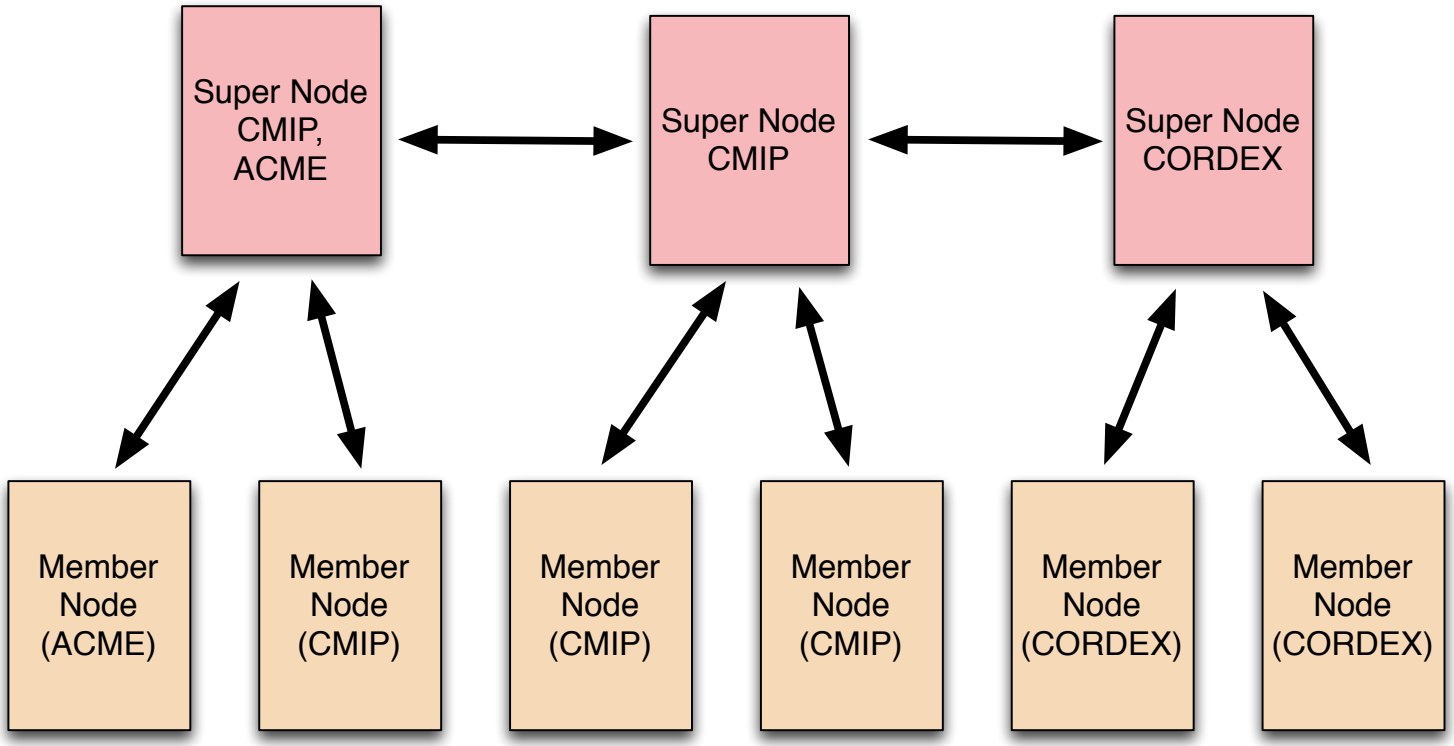
- Fault-tolerant distributed system, without a single point of failure.
- High scalability without overloading resources.
- Minimise communication overheads.
- PAN federation administration: handling cert requests, node memberships etc
- Consistent and highly available common configuration files
- Mechanism to ensure replica consistency across federation.

Node Manager types

Node managers can be of three different types.

1. Supernodes
   - A validated and reliable source for configuration directives, metrics, information about components etc, at project level.
   - Multiple concurrent supernodes for scalability, fault tolerance and load sharing.
   - Supernodes query other Node Managers for metrics and status.
   - A single Node Manager can serve as supernode to multiple projects or even as supernode to one and membernode to another etc.
2. Membernodes
   - Default Node Manager configuration
   - Cannot query other Node Managers.
3. Standby supernodes
   - The Node Managers run as supernodes only when too few supernodes are operational.

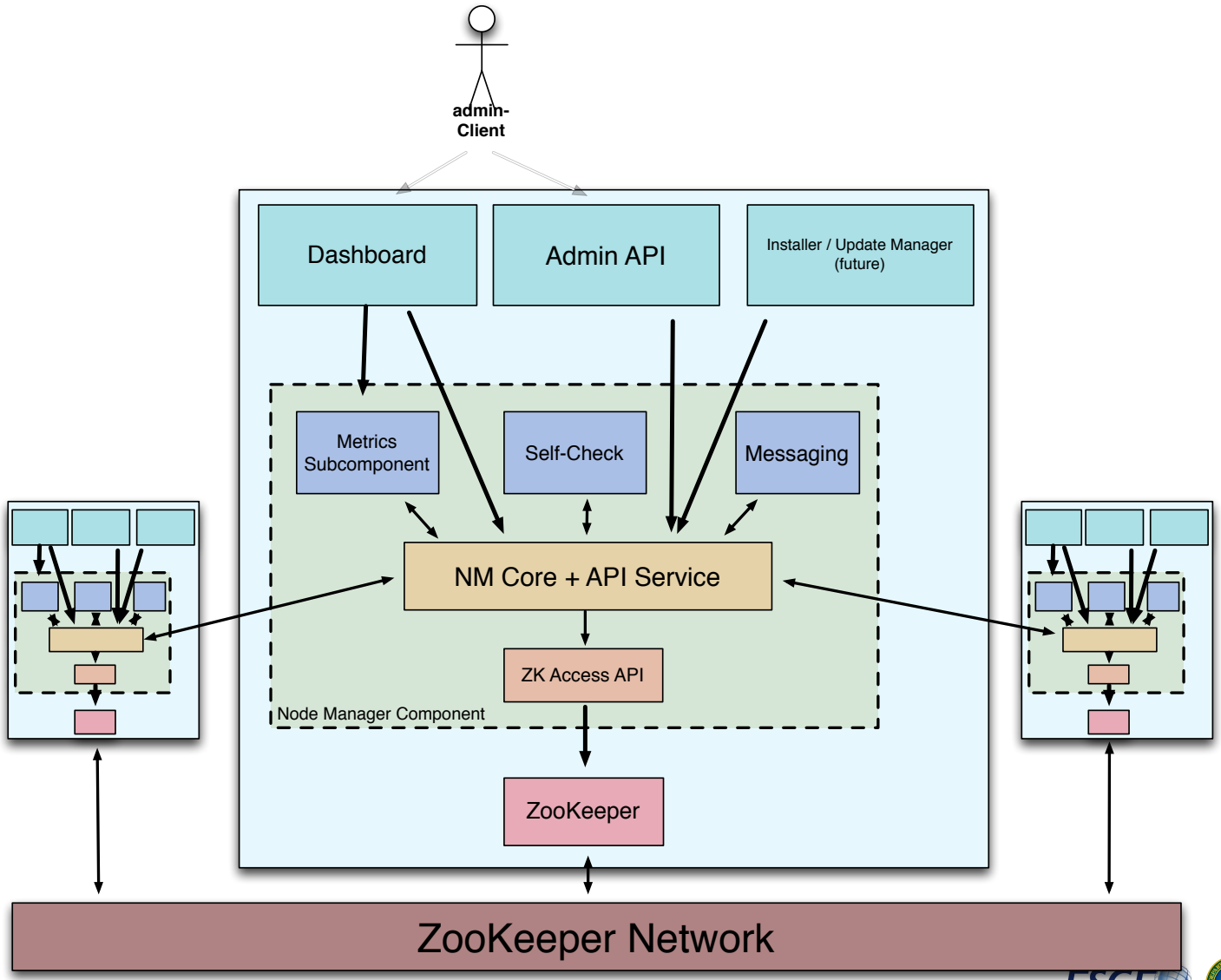
View of federated Node Managers



Node Manager components

- **Apache ZooKeeper**, to take care of group management (project), leader election for supernode management etc
- **ESGF Node Manager API**, to serve as a wrapper, for ZooKeeper and modify node's availability to be considered for use as supernode etc.
- **Metric collector**: query member nodes and aggregate them (supernodes)
- **Self-check component**: run sanity checks on self.
- **Messaging component**: alert notifications for local admins when services fail.
- **Admin console (local node)**: submit membership requests, CSRs, volunteer for supernode role etc
- **Admin console (supernode admin mode)**: sign CSRs, manage membership and volunteering requests etc.

Node Manager component diagram



Node Manager design considerations

- Security: factor for both user/machine executed elevated privilege operations.
- Design to guard against spoofing of membernodes/supernodes etc.