MESOSPHERE

DC/OS Master and Agent Node Recommendations

Arthur Johnson,
Solutions Architect, Mesosphere
ajohnson@mesosphere.io

Table of Contents

Masters and Master Networking

The master nodes in a DC/OS cluster are a critical component for cluster health and availability-- in reality, you cannot have a DC/OS cluster without masters, they are the 'brain' of the cluster and control resource assignment and overall cluster operations. Master nodes may be placed in different subnets/zones as long as latency is <10ms between all nodes. It is generally recommend that the masters exist in the same protected network zone, yet be placed in different racks with different network uplinks and power-distribution-units (PDUs).

Master Nodes - Resources

Best practice is to deploy a DC/OS production cluster with multiple master nodes in order to be resilient in the event of a failure and avoid a single point of failure (SPOF). A DC/OS master node is a host that runs a collection of DC/OS components which work together to manage the rest of the cluster. Key attributes of the master nodes are:

- Runs many different DC/OS components, most notably a Mesos master process
- A Mesos master is a process that runs on master nodes to coordinate cluster resource management and facilitate orchestration of tasks.
- Mesos depends on ZooKeeper, a high-performance coordination service to manage the cluster state. Exhibitor automatically configures and manages ZooKeeper on the master nodes.
- Master nodes work in a quorum to provide consistency of cluster coordination. To avoid split brain cluster partitioning, clusters should always be an odd number count in cluster size (e.g. 3, 5, 7, etc. nodes)

Resiliency of Master Nodes

Number of Master Nodes	Quorum Size	Failures Tolerated
1	1	0
3	2	1
5	3	2
(2 * F) + 1	F+1	F

A DC/OS Mesos master is mostly stateless, that is, (mostly) everything is held in memory. State is retrieved from the agents themselves (which are stateful). If all the Mesos masters are unavailable (e.g., crashed or unreachable), the cluster should continue to operate: existing Mesos agents and user tasks should continue running. However, new tasks cannot be scheduled, and frameworks will not receive resource offers or status updates about previously launched tasks.

Recommended Hardware (Baseline) for Masters

The following is a basic recommendation for hardware to run and support masters in a resilient, performant production environment. Site specific data and/or tasks might influence the requirements; depending on workloads, expectations, and other factors, experiences may vary. This provides our best practice for a production cluster starting point. While bare metal environments are preferred, virtual environments which meet these requirements are considered best starting points:

1. Master count: 5
2. Master Hardware profile:
    a. >= 16-cores in a two socket configuration
    b. >= 64GB RAM (NUMA-balanced) -- '/state' is maintained in the masters so **memory is critical for growth and performance**
    c. "Fast Disks" or SSD for `/var/lib/dcos`
    d. 10Gb NICs in (ideally) a non-blocking network topology (leaf-spine, low over-subscription ratios) between masters and agents. Bonding and LACP across the NICS and switch stacks (as possible)
    e. Mechanism to provide assisted or unassisted backup of replication log data and Zookeeper state

Recommended System Directories

Mesosphere does not have recommendations for sizing on these directories- However, the following are considered best practices: Increased spindles or devices, and higher performing, are generally preferred. This is at customer discretion considering prices of devices on individual servers.

Isolation for System Directories

In an ideal configuration, you consider separate partitions for the following filesystem paths, again, if possible. Additionally, these directories may appear on Masters and Agents-- On the Masters, it is highly recommended that `/var/lib/dcos` be hosted on a separate partition backed by fast, locally-attached storage (SSD/NVMe)

Replicated State Store Locations

Mount Partition	***Purpose***
`/var/lib/dcos/mesos/master/replicated_log`	Mesos Paxos replicated log
`/var/lib/dcos/mesos/master/overlay_replicated_log`	Navstar Overlay replicated log

/var/lib/dcos/cockroach	CockroachDB distributed database
/var/lib/dcos/navstar/mnesia	Navstar Mnesia distributed database
/var/dcos/navstar/lashup	Navstar Lashup distributed database
/var/lib/dcos/secrets/vault	Secrets Vault
/var/lib/dcos/exhibitor/zookeeper	Exhibitor Zookeeper distributed database
/var/lib/dcos/dcos-history	History Service cache

In addition to the above recommendations, the following should be planned for as well:

- All filesystems/mount points should ideally be on their own isolated IO path, directly to the physical devices to minimize IO contention
- The `/var/lib/mesos/slave/meta/resources`, `/var/lib/mesos/slave/volumes` & `/dcos/volume<N>` (`MOUNT` volumes, if in use) directories MUST all be preserved (or restored from backups, if present) for reservations to be re-advertised to the frameworks for operations and tasks to recover successfully
- DC/OS is installed on /opt/mesosphere-- /opt/mesosphere must be on the same mount point as "/". This is required because DC/OS installs systemd unit files under /opt/mesosphere and all systemd units must be available for enumeration during the initialization of the RAMdisk at boot. If /opt is on a different partition, or volume, systemd will fail to discover these units during the initialization of the RAMdisk and DC/OS will not automatically start at boot time.
- Do not mount /tmp with NOEXEC. This will prevent Exhibitor and ZooKeeper from running.

Final Partition Layout

Masters

Mount Point	***Size***	***Notes***
/var/lib/dcos	10 GB	LVM

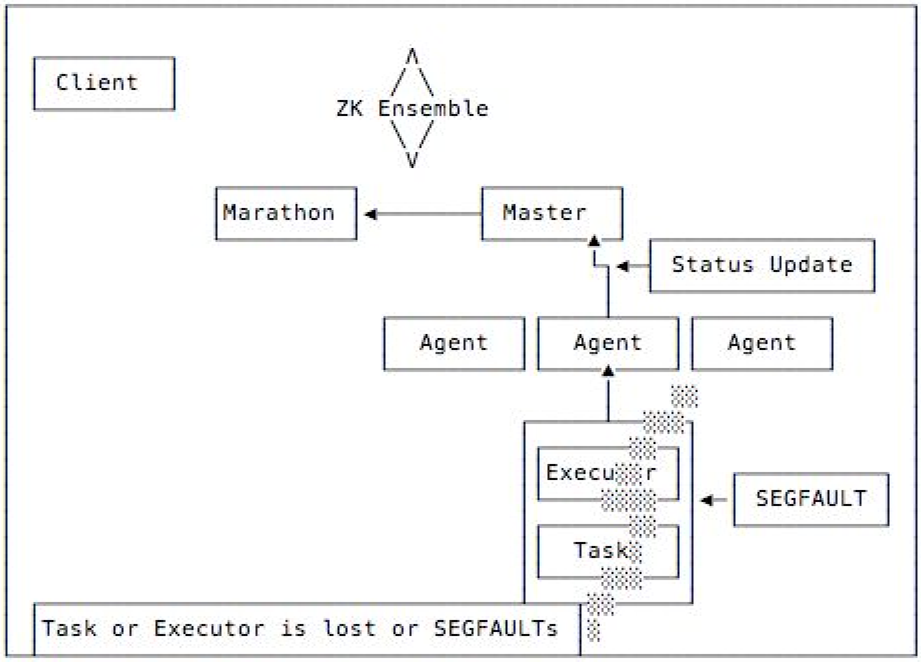
Network Partitions and Loss of Connectivity between Masters and Agents

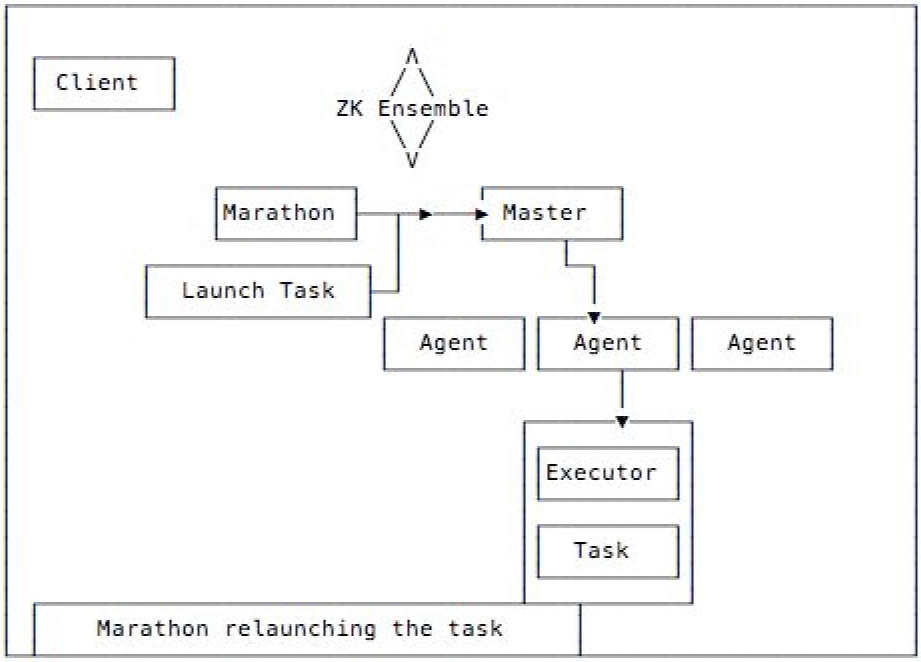
- In the event of a loss in network connectivity between the master and agents, Mesos agents and user tasks continue to run on the agent nodes. However, new tasks cannot be scheduled, and frameworks will not receive resource offers or status updates about previously launched tasks unless a quorum of master nodes is restored.
- In the event an agent loses connectivity to the network or is unresponsive, the default Mesos health check will fail due to lost heartbeats between the master and agent nodes. Unreachable nodes are marked for removal from the cluster and the scheduler (Marathon or Metronome) is responsible for starting the task on another agent node.

Task Failure Recovery

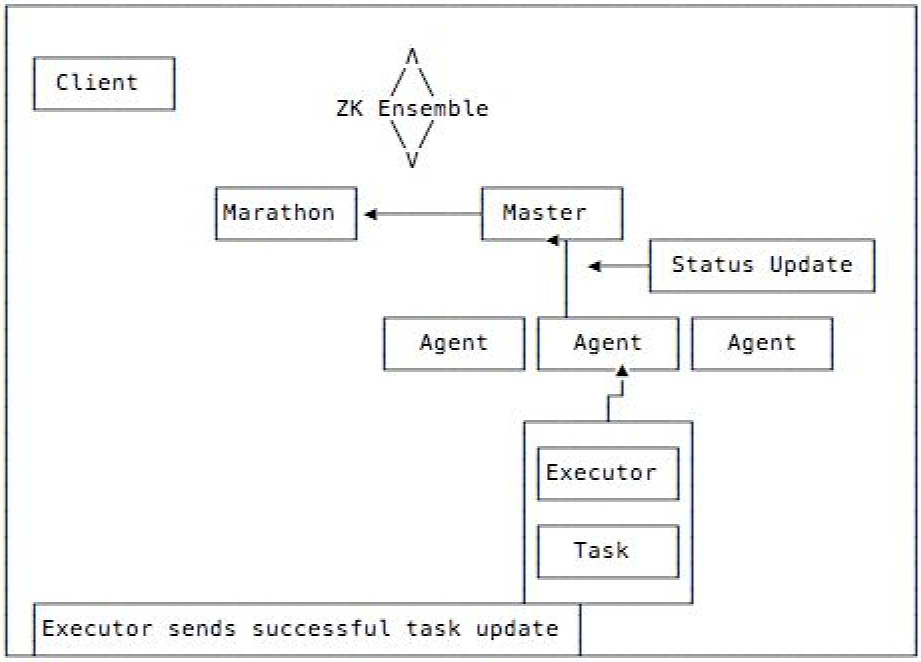
Task recovery in Mesos (and DC/OS) has the follows this sequence (*It should be noted that this is purely high-level and does not cover all the failure scenarios):

- Task or the executor is lost or segfaults
- The agent node updates the leader master which forwards the update to the framework
- Marathon relaunches the task on an agent with available capacity and after accounting for any constraints that were defined as part of the application specification
- Executor sends a task status update to the agent which is forwarded on to the master and Marathon





Marathon relaunching the task



Executor sends successful task update

Agents and Agent Networking

Recommended Hardware (Baseline) for Agents

The intention with agents is to 'scale-out' rather than 'scale-up'. Although, with recent advances in hardware, with densely packed chassis with many cores/CPU & memory, there may be cases where agent nodes may not exist in great numbers. If this condition exists, it is important to consider the loss of a single agent node and the percentage impact it has on the environment (e.g. loss of an agent node, out of 5, is a 20% loss of agent capacity).

Baseline Agent Recommendations (considered minimum requirements):

Component	***Recommendation***
Nodes	6 or more - 2 as public, 4 as private (minimum)
Processor	2 cores
Memory	16 GB RAM
Hard Disk	60 GB additional storage from what is required for the operating environment (Linux)

On agent nodes, best practice is to make sure the following requirements are met:

- `/var` with 60 GB or more free space. This directory space is used by the sandbox
- Directory: `/var/lib/mesos/slave` should be on a separate devices. This protects all the other services from a task filling a system disk
    - You can independently enable resource enforcement by inserting the environment variable: `MESOS_ENFORCE_CONTAINER_DISK_QUOTA=true` into one of the Mesos agent extra configuration files (e.g. `/var/lib/dcos/mesos-slave-common`) to have this take effect.

On the Agent, under `/var/lib/dcos`, persistent configuration override files are stored and it is a bit of an overkill to have it on its own partition.

Persistent Configuration Override Locations

Mount Partition	***Purpose***
`/var/lib/dcos/mesos-slave-common`	Configuration Overrides
`/var/lib/dcos/mesos-resources`	Resource Overrides

The disk space that Apache Mesos advertises in its UI is the (sum of the) space advertised by filesystem(s) underpinning `/var/lib/mesos`

Note: This space is further rolled up if there are MOUNT volumes (`/dcos/volume<N>`) present

On the Agents, the following directories under `/var/lib/mesos` should ideally be on distinct partitions, but if that's too much to ask, please ensure that `/var/lib/mesos` is hosted on a separate partition at the very least.

Distinct Mount Points under /var/lib/mesos

Mount Partition	***Purpose***
`/var/lib/mesos/slave/slaves`	This is used to house the sandbox directories for tasks
`/var/lib/mesos/slave/volumes`	This is used by frameworks that consume ROOT persistent volumes
`/var/lib/mesos/slave/docker/store`	This is used to store Docker Image Layers that are used to provision UCR containers
`/var/lib/mesos/slave/provisioner`	This is used to store the overlay filesystem layers for Docker containers provisioned with the UCR

Distinct Miscellaneous Mount Points

Mount Partition	***Purpose***
`/var/lib/docker`	This is used to store Docker Image Layers and by Containers launched with the Docker Engine

`/dcos/volume<N>`
`(e.g., /dcos/volume0, /dcos/volume1 ...)`
	This is used by frameworks that consume MOUNT persistent volumes.

Note: `/opt/mesosphere` could (should) also ideally be on its own partition but doing so requires newer builds of DC/OS (1.11+) that have all the commits from this PR: https://github.com/dcos/dcos/pull/2383

All of the aforementioned filesystem paths (mount points) should ideally be on their own isolated I/O path, all the way down to the physical devices to minimize the effects of noisy neighbors.

Note: The `/var/lib/mesos/slave/meta/resources`, `/var/lib/mesos/slave/volumes & /dcos/volume<N>` (MOUNT volume) directories MUST all be preserved (or restored from backups, if present) for reservations to be re-advertised to the frameworks for operations and tasks to recover successfully.

Final Partition Layout

Agents

Mount Point	**Size**	**Notes**
`/var/lib/mesos/slave/slaves`	`250 GB`	Local Volume
`/var/lib/mesos/slave/volumes`	`250 GB`	Local Volume
`/var/lib/mesos/slave/docker/store`	`250 GB`	Local Volume, formatted with: `mkfs.xfs -ftype=1 /dev/<>`
`/var/lib/docker`	`250 GB`	Local Volume, formatted with: `mkfs.xfs -ftype=1 /dev/<>`

Other considerations include:

- Network access to a public repository or internal Docker registry for all agents
- On RHEL 7, `firewalld` must be stopped and disabled. There are known Docker issues that firewalld interacts poorly with.

Execution resources (DC/OS agent nodes)

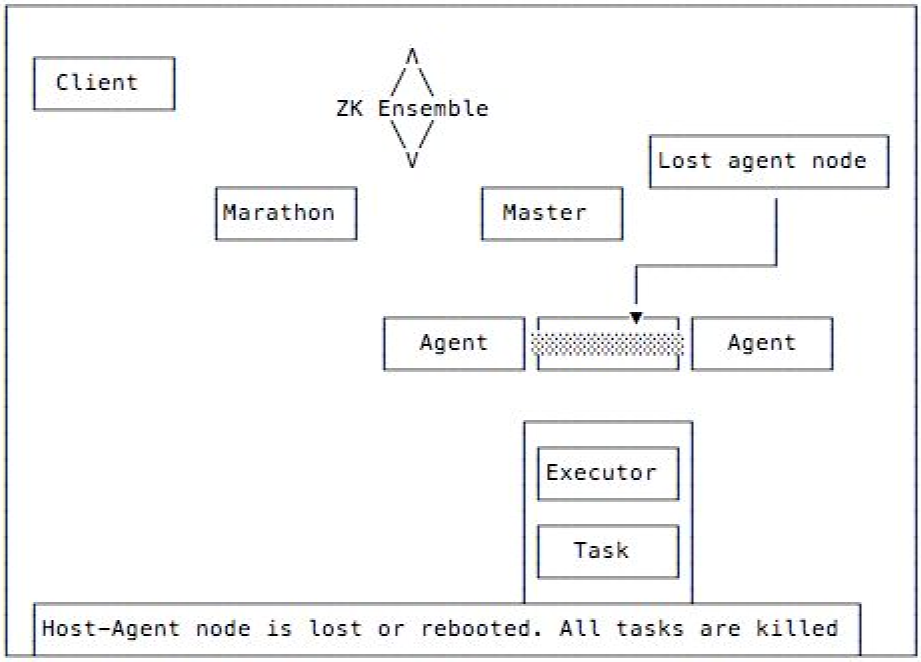
A Mesos agent is a process that runs on agent nodes to manage the executors, tasks, and resources of that node.

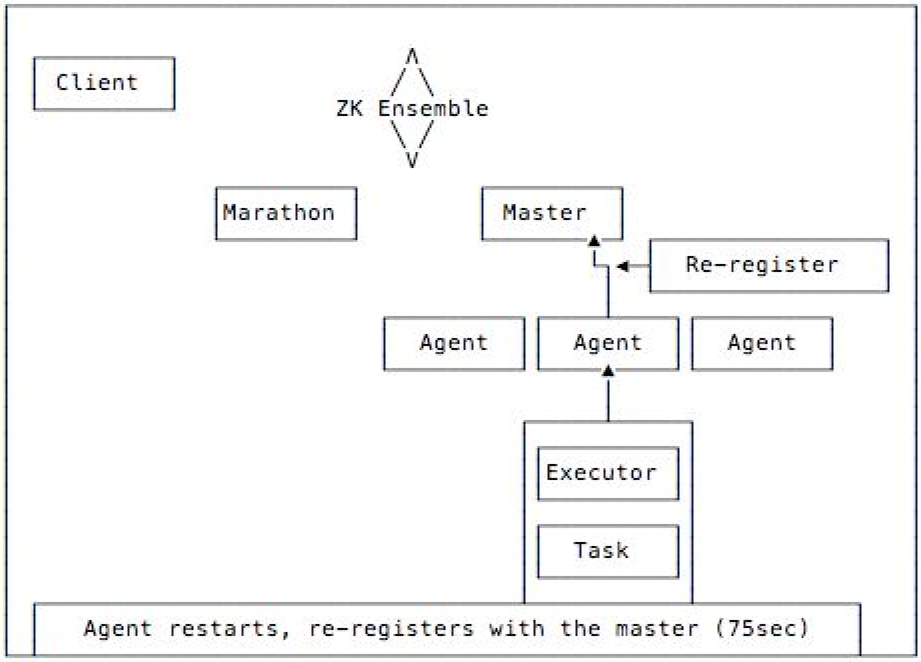
- The Mesos agent registers some or all of the node's resources, which allows the leading Mesos master to offer those resources to schedulers, which decide on which node to run tasks.
- The Mesos agent reports task status updates to the leading Mesos master, which in turn reports them to the appropriate scheduler.

Agent Failure Recovery

Agent recovery in Mesos (and DC/OS) has the follows this sequence:

- Agent restarts or segfaults. The executor is forked off as a separate process along with the task and continues running on the node.
- When the agent service comes back up, it reads back state information, runs recovery, reattaches to the executor and re-registers with the master.
- Agent Recovery is a feature of Mesos that allows:
  - Executors/Tasks to keep running when the slave process is down
  - Restarted agent process to reconnect with running executors/tasks
- Agent recovery works by checkpointing enough information (e.g., Task Info, Executor Info, Status Updates) about the running tasks and executors to local disk.
- Once the agent and the framework(s) enable checkpointing, any subsequent agent restarts would recover the checkpointed information and reconnect with the executors/tasks.
- If the host running the agent process is rebooted, all executors/tasks are killed.
- A restarted agent should re-register with the master within a timeout (currently 75s). If the agent takes longer than this timeout to re-register, the master shuts down the agent, which in turn shuts down any live executors/tasks.
- Frameworks should explicitly request checkpointing by setting. FrameworkInfo.checkpoint=True before registering with the master.



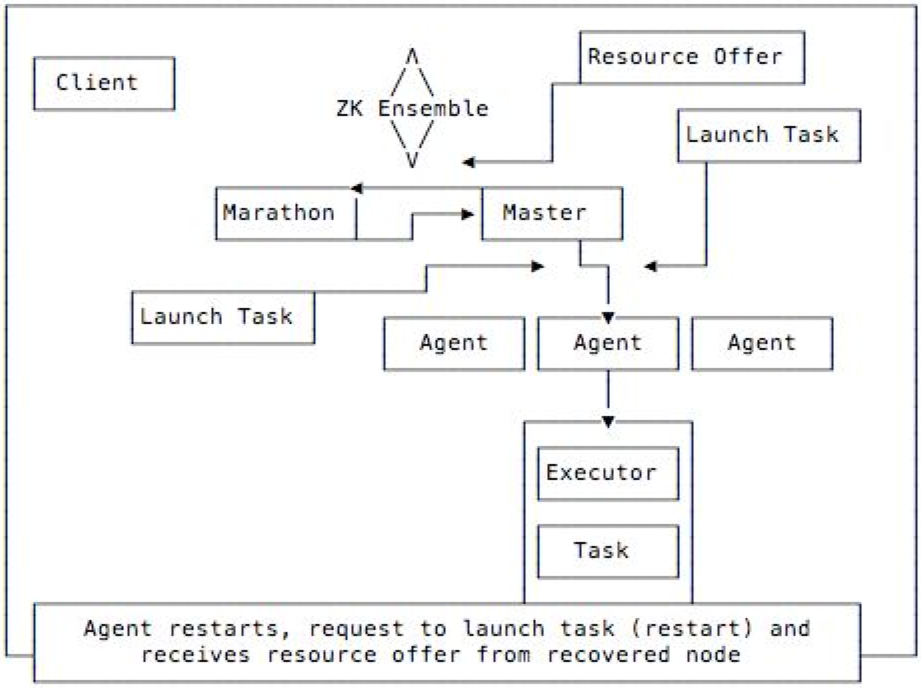


Agent Node Host Failure Recovery

Agent host recovery in Mesos (and DC/OS) has the follows this sequence for applications or services launched in Marathon:

- Host running the agent node in the cluster is network partitioned or goes off the grid

- Mesos health checks will fail (Default 5 pings, 15 seconds each, Total of 75 seconds (5 minutes))
- The host is considered lost after 90 seconds. If the leading master doesn't see a heartbeat from the agent, sends a SLAVE_LOST message to the framework. Unreachable agents marked for removal from cluster. TASK_LOST sent for all tasks on agent nodes.
- Marathon relaunches replacement task on another agent.


Client
ZK Ensemble
Resource Offer
Launch Task
Marathon
Master
Launch Task
Agent
Agent
Agent
Executor
Task
Agent restarts, request to launch task (restart) and receives resource offer from recovered node

Conclusion

Considering planning and design requirements for Master nodes is a critical aspect to planning any DC/OS deployment. The reliability of the cluster, task workload (number of tasks supported), and performance is directly tied to the infrastructure which is implemented for the Master node function. Additionally, in the event that resources for Master nodes need to be reconsidered, changing the underlying infrastructure is tantamount to re-installing the cluster. While it is possible to add Agent nodes, and in various hardware profiles, it is less of a doable proposition for the Master nodes.