Considerations for, and an example of, sizing OpenShift for virtual machines

TL;DR: It's the same as any other OpenShift sizing. Each VM has a request for the full amount of memory plus overhead. There is no memory overcommitment (yet). Memory will be your biggest constraint and is how the cluster/nodes will be sized for the majority of deployments.

At a high level, the process is to determine the amount of virtualization resources needed (VM sizes, overhead, burst capacity, failover capacity), add that to the amount of resources needed for cluster services (logging, metrics, ODF/ACM/ACS if hosted in the same cluster, etc.) and customer workload (hosted control planes, other Pods deployed to the hardware, etc.), then find a balance of node size vs node count.

Getting started

To size an OpenShift cluster based on the number of virtual machines, we need a few important bits of information:

1. The size of the VMs, specifically CPU and memory.
2. The number of VMs.
3. The amount of overhead for the VMs. This is static(ish).

With that information we can come up with various scenarios that will both affect, and be affected by, further decisions such as cluster architecture, HA capacity, and special resource requirements.

To begin, we need to understand the desired workload. In the absence of valid VM sizing data, we will use "t-shirt" sizing for VMs:

- X-Small = 1 vCPU, 1GiB memory
- Small = 1 vCPU, 4GiB memory
- Medium = 2 vCPU, 8GiB memory
- Large = 4 vCPU, 24GiB memory
- X-Large = 8 vCPU, 48GiB memory

Step 1: Determine the raw capacity required for workloads

To set the stage for the remainder of the example, let's identify a desired workload. The customer wants to host this many VMs...

- 1000 small = 1000 vCPU, 4000GiB. Plus 159 GiB memory overhead.
- 300 medium = 600 vCPU, 2400GiB. Plus 53 GiB memory overhead.
- 200 large = 800 vCPU, 4800GiB. Plus 45 GiB memory overhead.
- 200 x-large = 1600 vCPU, 9600GiB. Plus 60 GiB memory overhead.

This brings us to a total of 1700 VMs using 4000 vCPUs and 20800 GiB memory, plus 317 GiB overhead. The average VM will have 2.4 vCPUs and use 12.4GiB memory, which is a convenient way of estimating capacity for nodes.

Step 2: Decide node size

Now we need to determine node size. This includes a lot of factors, but the two that we will focus on are 1) failure domain, which is subjective, and 2) density, which can be driven by max Pod count and/or physical resources.

For density, we know that the current max pod count is 500. At a minimum this deployment will need 4 nodes to be able to host the VM count. This would result in nodes that have `1700 / 4 = 425` VMs each. With the average VM size determined above, we see that the hosts would need capacity for at least `2.4 * 425 = 1020` vCPUs and `12.4 * 425 = 5270` GiB memory. Since we cannot do memory overcommitment, this means that memory becomes the limiting factor for *most* servers, which means a 4 node cluster is not feasible and, therefore, maximum density / fewest nodes in the cluster is not an optimal choice.

Without memory overcommitment, this will almost certainly be the limiting factor for server size. If we assume that the typical/average virtualization server has 768GiB of memory, then we can determine the max VM count by dividing the total server memory by the average VM memory. In this case `768 / 12.4 = 62(ish)` VMs per node. This would result in the customer needing `1700 / 62 = 28 (rounded up)` nodes in the cluster based on memory. Each of those 28 servers would need to host `2.4 * 62 = 149` vCPUs. A modest 4:1 CPU overcommit would result in the servers needing only 38 physical cores to host the workload. A dual socket 22-core server would have a small amount of unused, or “stranded”, CPU resources in this instance.

A secondary factor is to attempt to avoid stranding resources by balancing CPU and memory according to requirements. This is as simple as keeping the physical CPU:memory ratio inline with the overall requirement - 4000:21000, which reduces down to 1:5.25.

To balance the ratio, we also need to keep the CPU oversubscription factor in mind when choosing a physical server size. Jumping to 2TiB of memory per server, we can now host `2048 / 12.4 = 165(ish)` VMs per node based on memory, which would need 396 vCPUs. A dual socket server with 24 cores per socket would have an overcommit ratio of `396 / (2 * 24) = 8.25:1`. This both falls within the expected/allowable CPU overcommit ratio for OpenShift Virtualization (defaults to 10:1) and is inline with our overall CPU:memory ratio at 1:5.17.

165 VMs per node means our customer would need 11 (actual = 10.3, but you can’t buy .3 of a device) servers.

If we factor in failure domain size, we need to understand the amount of capacity lost when a server goes offline. 28 servers means losing a single server would result in losing just 3.6% of total capacity. 11 servers means losing a single server would reduce total capacity by approximately 9%. The lost server would also result in 28 or 165 VMs needing to be rescheduled. Is either of these numbers too large of an impact? If so, then we need to adjust the size of the nodes downward to increase node count. This is a risk decision that can only be made by the customer.

Thus far the above calculation has not taken into account node overhead. If we allow the nodes to automatically configure system and kube reserved resources, then there is a formula to follow. For a node with 768GiB of memory, 22GiB will be reserved for the system. With 2TiB of memory, 48GiB will be reserved. In both instances of a 44 or 48 core server, the CPU reservation is modest at .18 and .19 cores respectively.

OpenShift Virtualization itself has node overhead as well. This is a negligible 360MiB memory and a flat 2 cores per compute node.

These two overheads combined - approximately 2.2 cores and 22 or 48 GiB memory per node - may affect the size or density of the nodes. In this example, both of our node count estimates were rounded up (27.4 -> 28 and 10.4 -> 11), therefore it is expected that there is enough excess resources to accommodate these overheads already.

Step 3: Adjust for cluster architecture

There are three architecture options:

- SNO
- Compact
- Full

SNO has some clear limitations and sizing is relatively straightforward, so we can ignore it for this exercise.

Compact and full are two sides of the same coin, but we need to know other info too…

- Does the customer want homogenous node sizes or are they willing to use a different node size for control plane and infra?
    - If yes, then consider having schedulable control plane nodes that also host infra workload. Depending on the node size/capacity, there may still be capacity for customer workload on these nodes.
- Will this cluster be using hosted control planes, where there are no control plane nodes?
- Which of the infra qualified add-on features, e.g. logging, etc., will be hosted by the cluster?
    - If the cluster does not have dedicated infra (or combined infra + control plane) nodes, this capacity will need to be added to the overall requirement.
- Is the customer planning on using features, such as pipelines and/or GitOps, which will increase the API server, and other control plane component, utilization?
    - Each of these places additional burden on the control plane components. This may affect the size of those nodes if they're dedicated or how many VMs can be hosted on the control plane nodes.

Let's say that the customer has chosen the 11-node cluster size from step 2. They want to have schedulable control plane nodes that are able to host their workloads, multiple additional OpenShift Platform Plus features, and intend to use homogeneous nodes. For the sake of

simplicity, let's assume that the combined resource requirements for this totals approximately 15% of the three control plane nodes. This means that the cluster needs to have and additional `(3 * 15) / (11 * 100) = 4%` total cluster capacity for non-VM workloads. Since we do not already have this much excess capacity available (we rounded up from 10.4 -> 11 nodes, which gives us 60% of one node distributed across the cluster, however that capacity is consumed by overhead), this would require a 12th node to be added to the cluster.

Step 4: Adjust for failover / HA and resource balancing

We factored in the size of failure domain above, however we have not taken into account how to accommodate that capacity in the event of a failure scenario - or even when taking nodes offline for, for example, cluster updates and upgrades. With a 12 node cluster, each node represents 8% of cluster capacity, so we need to accommodate for `8 * #_of_node_failures` worth of capacity to be rescheduled in the surviving nodes.

- Tolerating one node failing would limit hosts to 92% maximum utilization.
- Two node tolerance would limit hosts to 83%.
- Three node tolerance would limit hosts to 75%.

Furthermore, we want to prevent things like out of memory conditions. When to trigger evictions will depend on a number of factors, most importantly we want to set the value low enough where soft evictions will have time to happen before hard evictions and/or any actions are taken by the OOM_kill process. A secondary consideration is that the soft eviction will also become the threshold where a node's workload triggers rescheduling for things like resource balancing as well.

Depending on the size of the nodes, the size of the VMs, and the length of time to live migrate virtual machines, it may make sense to have the soft eviction threshold set anywhere between 85% - 98%. Splitting this down the middle for our example, if we choose 92% memory utilization on servers with 2TiB of memory, that would trigger soft evictions (i.e. live migrations) for VMs when the host has approximately 160GiB of free memory remaining. This may or may not be enough for the workload and would need to be monitored and adjusted over time.

However, for the sake of sizing, we can now assume that for a 12-node cluster where we want to tolerate two nodes failing with a soft eviction threshold of 8% remaining capacity, we can math out that we are increasing overall resource requirements by 24% (16% for node failover capacity, 8% for eviction). This means we would want to increase the node count by 24%, `12 + (12 * .24) = 15` nodes total.

Step 5: Storage, network, special resources, and more

There are multiple other aspects to take into account that may affect specific node hardware. An incomplete list includes:

- CPU, memory, network, and disk requirements for ODF, Portworx, and other Kubernetes-native storage solutions
- Network throughput for cluster functions, SDN, live migration, storage traffic, hosted applications, etc.
- High IO/throughput workloads that may cause resource starvation for co-hosted workloads.
- GPUs, SR-IOV devices, and other “special” resources that may be allocated to virtual machines such as dedicated CPUs, NUMA regions, etc.
- Regulatory, or self, imposed scheduling requirements/restrictions. Are there requirements which affect the ability to maximally utilize the resources in the cluster? For example, “team A’s VMs cannot be co-hosted on the same servers as team B’s”. This includes not just (anti)affinity rules, but any taints/tolerations and special resources that the customer desires to protect. These will affect overall resource utilization and may impact sizing.
- Third party agents and services, such as monitoring and log forwarding.
- Will there be excessively large virtual machines that might be a challenge to schedule? Will there be VMs with substantial CPU requirements that may need special accommodations and/or have limited scheduling opportunities?

These can be difficult to accommodate in a generic sizing exercise like this, instead it may be helpful to size them separately in an exercise dedicated to those workloads. The customer can choose to have one or more dedicated clusters for those workloads or use features such as taints and node selectors to keep cluster count to a minimum.

Step 6: Disaster recovery

See the OpenShift Virtualization Disaster Recovery Guide for strategies and methods to recovery virtual machines at the destination site. For workloads which are identified to be recovered, follow the above steps at that site with the appropriate considerations.