IMPLEMENTATION GUIDE

Data Center
Rack Scale Design

Dynamic Container Scaling with Intel® Rack Scale Design

Summary

This document describes several ways that container environments like Kubernetes can take advantage of the capabilities of Intel® Rack Scale Design (Intel® RSD) compatible hardware to deliver more dynamic scaling of container resources.

Why Containers?

Containers are similar to virtual machines, but abstract the operating system environment down to the very specific components needed to run the application or service rather than virtualization of hardware. Containers provide several advantages: they are consistent across environments, enable deployment automation, and make efficient use of resources. Container hosting systems, such as SUSE CaaS Platform*, provide an environment for creating and running containers. They typically include a cluster of worker servers for running containers, and one (or more) servers for administration. The SUSE CaaS Platform includes orchestration (e.g., Kubernetes), hardware management, software deployment, and other services.

How Intel® RSD Complements Containers

Although containers and other forms of virtualization provide much better use of resources than the old "one-application-one-server" paradigm, they are still constrained by the lack of flexibility of traditional server hardware. A hyperscale data center based on racks of preconfigured servers is still limited in its ability to achieve full utilization while adapting to changing requirements. Data center operators need a more flexible hardware architecture that is open, composable and interoperable, plus they need it to work with their orchestration software.

This is exactly what Intel® Rack Scale Design (Intel® RSD) provides: a new, open architecture that improves productivity, performance, agility and efficiency. Intel RSD is an industry-aligned architecture for hyperscale data centers that provides physically disaggregated, modular building blocks that can be managed as resource pools and composed on demand to meet specific workload requirements. It also exposes open, scalable and secure management APIs that support interoperability across hardware and software vendors.

Intel RSD delivers "hyper flexibility," which enables operators to reduce overprovisioning and achieve higher utilization and reduced capex. It also makes equipment refresh more economical by avoiding the need to replace entire servers. Intel RSD is a natural complement to a container hosting platform, resulting in more efficient utilization of hardware, and increased automation of hardware configuration tasks. Intel RSD makes the data center more economical, flexible, simpler to manage, and easier to scale out on demand. Products based on the Intel® RSD architecture are available today from major OEMs and ODMs.

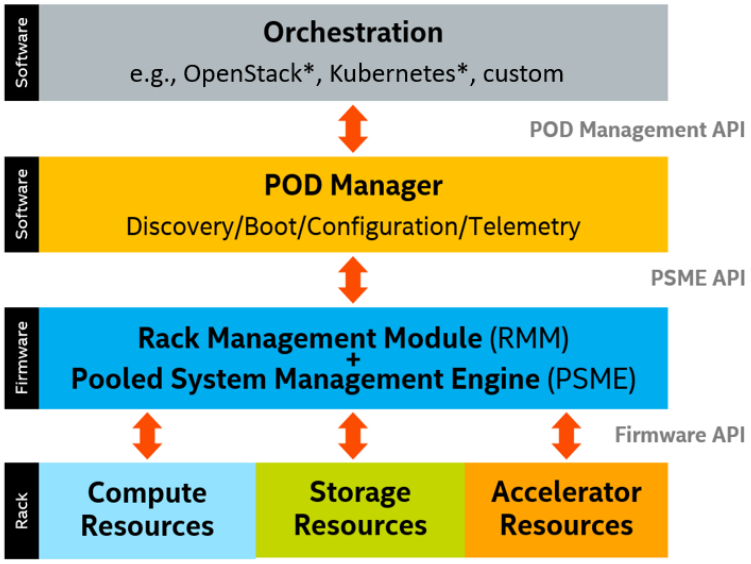

Figure 1. Intel Rack Scale Design Architecture

Dynamic Container Scaling with Intel® RSD

Intel® RSD supports dynamic composition of physical hardware resources in the most optimal configuration for a targeted workload. Intel RSD RESTful APIs allow orchestration software to implement a variety of dynamic use models in a containerized environment.

Use Case 1 – on-demand addition of servers to scale out a container cluster (dynamic scaling)

In this scenario, Kubernetes (sometimes abbreviated as K8s) requests deployment of additional containers, but the existing cluster doesn't have adequate capacity. Based upon an interactive reaction to an alert or, if allowed, full-stack automation with K8s, Intel® RSD can automate the process of adding a worker server to the cluster:

1. A composition request is sent to Intel RSD Pod Manager (PODM) using Intel RSD APIs.
2. PODM composes a new node, presenting it to the container hosting platform.
3. The container hosting platform deploys the new node, installs the necessary software, incorporates the node into the container cluster, and then deploys the requested containers.

Conversely, if the needed resource capacity subsides, these steps can be done in a reverse order: the extra node can be cordoned from the K8s cluster scheduling, the container workloads are drained off, the node is removed from the cluster, and the node's resources are returned to the available resource pools.

Use Case 2 – on-demand deployment of servers with pre-defined roles (template-based dynamic scaling)

Usually, a cluster deployment requires a number of worker nodes for running containers, as well as an administration server, which typically only needs modest hardware (e.g., fewer cores and DIMMs). We assume the container hosting platform has templates for two roles: one for worker servers, and one for administration servers. The Intel® RSD composition request API accepts parameters that describe a composed server in terms of CPU, DRAM, storage, and other requirements. So an administrator or K8s (if allowed to request such infrastructure changes) can use the role templates to determine the correct composition parameters to send to PODM, which then creates a new node from the available recourse pools with the desired characteristics.

We can extend this scenario by assuming that the data center manager plans to deploy ten container clusters over time, each with 10 workers and one admin server. She might purchase Intel RSD-compatible hardware for one hundred workers and ten administration servers, and initially deploys the ten clusters with one administration server and six worker servers each to meet current workload demand. The remaining forty servers are placed in available resource pools, which makes them immediately available for on-demand expansion as needed, whether the need is for more clusters, workers or admin servers.

A data center operator can set up a script to automate the deployment of these Intel RSD based clusters:

1. Request PODM (via Intel RSD APIs) to compose a server using the administration template.
2. Install the operating system and other software (via network/PXE or other means) for the administration node.
3. Request PODM to compose 6 servers using the worker template.
4. Install operating system and request container hosting platform software from the administration node to incorporate the worker servers into the container hosting cluster.

Other templates could define composed servers for other roles, for example, servers with higher storage capacity (or multiple storage tiers) for software defined storage (SDS) clusters.

Use Case 3 –on-demand reconfiguration to scale performance of a containerized, data intensive workload (dynamic optimization)

In this scenario, we assume that a Kubernetes container cluster is already installed and running on an assortment of hardware including an Intel RSD compatible rack. Kubernetes runs a container with a data intensive application. K8s has a template associated with the container image that specifies a worker with additional persistent storage volumes (NVMe) and uses it to request Intel RSD Pod Manager (PODM) to attach two NVMe drives to the node on which the container is running. PODM reconfigures the node as requested and the container now has the resources needed to deliver the required performance and resources.

There are many other dynamic use models that are enabled by Intel RSD, but let's turn now to an exploratory concept demo that shows how a container orchestration environment might interact with Intel RSD.

Intel RSD and the SUSE CaaS Platform—Concept Demo

Intel RSD can work with a wide range of orchestration software to manage bare metal, virtual machine and containerized environments. In this concept prototype, SUSE and Intel collaborated on a simple interface between Intel RSD APIs and the SUSE CaaS Platform* to demonstrate the third use case described above, i.e., adding resources to an already composed and running node in a Kubernetes cluster.

In this demonstration SUSE CaaS Platform accesses Intel RSD Pod Manager (PODM) APIs to add two host storage drives to an existing composed node connected over PCIe for a containerized SMuFin* (Somatic MUtation FINder) genomics application[i]. It also demonstrates how these resources can be easily returned to resource pools, ready for use elsewhere.

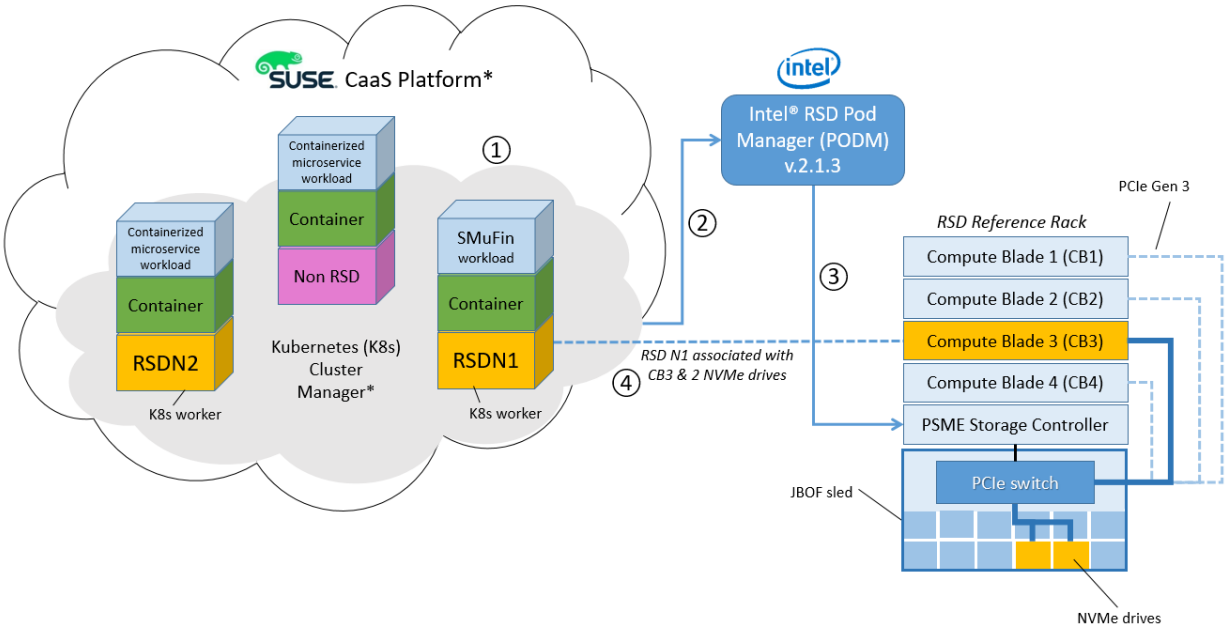

Figure 2. Composing a node to add NVMe drives to a containerized workload.

The interaction goes like this:

1. SMuFin app running on node RSDN1 requires hot storage.
2. K8s requests PODM to add two host storage drives to the existing composed node RSDN1.
3. PODM sends "AttachEndpoint" POST action to the PSME storage controller, which configures the PCIe switch in the JBOF chassis.
4. 2 NVMe drives are attached to CB3 via PCIe3, and associated with RSDN1 node.

From an operator's point of view, here are the underlying steps and commands issued to the SUSE CaaS Platform (this is just a demo—in an integrated production solution, these steps could be automated and transparent to the operator):

1. The first step is to minimize the impact on the SUSE CaaS Platform cluster by removing the node to be modified from the Kubernetes scheduler's view to make sure Kubernetes doesn't try to schedule the node during the reconfiguration.

   ```
   kubectl cordon RSDN1
   ```

2. Next the running containers are moved off the target node.

   ```
   kubectl drain RSDN1
   ```

3. SUSE CaaS requests PODM software to add two host storage drives from a JBOF sled in the rack to the existing composed node RSDN1, which is using compute blade 3 (CB3).

   ```
   curl -k -u admin:admin -X POST -H "Content-Type: application/json" -d
   '{"Resource" : {"@odata.id" :
   "/redfish/v1/Chassis/<JBOF_Chassis_ID>/Drives/<Drive-ID>"}}'
   https://<PODM_IP>:8443/redfish/v1/Nodes/RSD1/Actions/ComposedNode.Attac
   hEndpoint
   ```

4. PODM send commands to the JBOF controller to attach two NMVe drives to compute blade CB3; the PCIe Gen 3 switch in the JBOF connects the drives to CB3 via PCIe Gen 3 cables.

5. Enable Kubernetes to utilize these new volumes for the containerized SMuFin workload.

For example:

- Create a PersistentVolume (keying off capacity for differentiation and matching the mount points)
- Create a PersistentVolumeClaim (again keying off capacity and setting read/write attributes)

6. Re-enable the node for scheduling of container workloads.

   ```
   kubectl uncordon <node>
   ```

7. Launch the containerized workload (and specifying the volume claim mounts to match the container workload's configuration needs) with the K8s manifest for the SmuFin application container.
8. Let the workload run to completion.

The cool thing about Intel RSD is that the resources of a composed nodes can be returned to the available resources pools when they are no longer needed for a specific use, allowing them to be recomposed to a different node later, or for a different set of requirements. In this case, the NVMe drives can be detached and returned to the available resource pools, while the node remains intact. The SUSE CaaS Platform to Intel RSD interactions are:

1. SMuFin workload completes execution in container hosted on RSD node RSDN1.
2. SUSE CaaS Platform requests PODM to “detach” 2 NVMe drives from RSDN1.
3. PODM sends “DetachEndpoint” POST action to storage controller.
4. The connection between the 2 NVMe drives and RSDN1 is removed while maintaining RSDN1 node association with CB3.

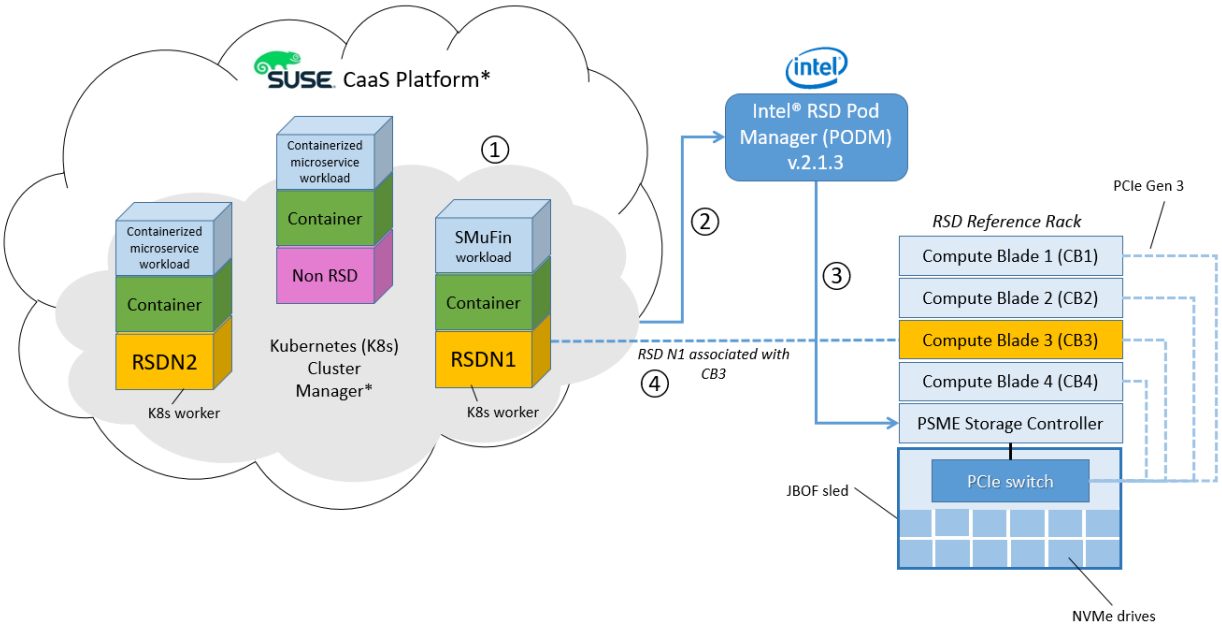

Figure 3. Releasing resource back to the pools when no longer needed.

From the operator’s point of view, the steps involved are:

1. All containers are moved off the target node.

   ```
   kubectl drain RSDN1
   ```

2. Delete the workload pod, volume claim and volumes.

3. Unmount the NVMe drives and detach them from the node RSDN1 via RSD APIs (i.e., request PODM to remove the previously added NVMe drives).

```
curl -k -u admin:admin -X POST -H "Content-Type: application/json" -d
'{"Resource" : {"@odata.id" :
"/redfish/v1/Chassis/<JBOF_Chassis_ID>/Drives/<Drive-ID>"}}'
https://<PODM_IP>:8443/redfish/v1/Nodes/RSD1/Actions/ComposedNode.Detac
hEndpoint
```

4. PODM send commands to the JBOF controller to detach the two NMVe drives from blade CB3.

5. Enable Kubernetes to utilize the node RSDN1 for other workloads.

```
kubectl uncordon <node>
```

Conclusion

By leveraging Intel RSD capabilities, it is possible for container orchestration environments to deliver **dynamic container scaling and optimization** in the data center. That is, container platforms could determine a workload's resource needs, either from a template or performance indicators, and employ RESTful APIs on Intel RSD compatible hardware to dynamically configure a mix of CPUs, DRAM, accelerators (e.g., Intel® FPGAs), hot storage (e.g., NVMe resources), and hard disk storage required to optimize the application's performance or other characteristics. As workloads change, a container platform could reconfigure systems to adapt to the new requirements.

---

[i] SMuFin (Somatic MUtation FINder) is a reference-free method designed to identify somatic variation on tumor genomes by direct comparison with a corresponding normal genome of the same patient. It is available from the Computational Genomics Group at the Barcelona Supercomputing Center. http://cg.bsc.es/smufin/

Product Availability

Intel RSD compatible products are available now from major vendors including Dell EMC*, Ericsson*, HPE*, Huawei*, Inspur*, Quanta*, Radisys*, Supermicro*, Wiwynn* and others. Learn more about how the Intel RSD architecture can accelerate your transition to an open, modular, software-defined Data Center at: https://www.intel.com/intelrsd. Or contact your local Intel representative to discuss how Intel® can help you to meet the demands of the Digital Transformation.