APPLICATION NOTE

Intel Corporation

Intel® Speed Select Technology - Base Frequency (Intel® SST-BF) with Kubernetes*

Authors

Louise Daly

David Hunt

Chris MacNamara

Kuralamudhan Ramakrishnan

1 Overview

Select SKUs of 2nd generation Intel® Xeon® Scalable processor (5218N, 6230N, and 6252N) offer a new capability called Intel® Speed Select Technology - Base Frequency (Intel® SST-BF). This document describes how to utilize the Intel® SST-BF feature in a cluster managed by Kubernetes* (K8S*) container orchestration. It includes steps to configure the frequency (and by association, the power consumption) of CPU cores on servers, expose feature capabilities, and provision workloads.

It's important to note that Kubernetes existing methodologies and architecture are leveraged and extended in support of Intel® SST-BF provisioning, feature exposure, and workload deployments. For example, a key element is assigning pods to nodes using the existing nodeselector mechanism.

This document is part of the Network Transformation Experience Kit, which is available at: https://networkbuilders.intel.com/

Primary use cases include the following:

- Deployment of NFV infrastructure containers to high-frequency or standard-frequency cores. For example, deploying Open vSwitch* [OVS] or load distribution functions to higher-frequency cores.
- Deployment of containers to high-frequency or standard-frequency cores. In this case, it is not necessary to have any knowledge within the workload to select the high or standard-frequency cores. Hence, the workload does not have to change to comprehend the specific cores.
- Workloads may be aware of the high-frequency and standard-frequency cores and may of course decide to latch software to the appropriate cores. Hence, within a workload, the pinning of virtual CPUs to physical CPUs, or the pinning of a container to a core. This case involves the deployment of selected cores of a workload or a container as high- frequency or standard-frequency cores. For example, freeing a workload bottleneck by selecting a core(s) from within a workload as a high-frequency core. Or configuring a core running a distribution thread in a container as a high-frequency core, where the workers are on standard-frequency cores.

Table of Contents

Figures

Tables

1.1 Terminology

Table 1. Terminology

ABBREVIATION	DESCRIPTION
CMK	CPU Manager for Kubernetes*
DPDK	Data Plane Development Kit
Intel® SST-BF	Intel® Speed Select Technology – Base Frequency
K8S*	Kubernetes*
NFD	Node Feature Discovery
NFV	Network Functions Virtualization
OVS	Open vSwitch*
Pod	Group of Kubernetes containers that are deployed together on the same host.
VM	Virtual Machine

1.2 References

Table 2. References

REFERENCE	SOURCE
Kubernetes – Production-Grade Container Orchestration	https://kubernetes.io/
Enhanced Platform Awareness in Kubernetes Application Note	https://builders.intel.com/docs/networkbuilders/enhanced-platform-awareness-in-kubernetes-application-note.pdf
CPU Pinning and Isolation in Kubernetes Application Note	https://builders.intel.com/docs/networkbuilders/cpu-pin-and-isolation-in-kubernetes-app-note.pdf
Node Feature Discovery for Kubernetes (NFD)	https://github.com/kubernetes-sigs/node-feature-discovery#overview
CPU Manager for Kubernetes* (CMK)	https://github.com/intel/CPU-Manager-for-Kubernetes
DPDK tstpmd Application User Guide	https://doc.dpdk.org/guides/testpmd_app_ug/

2 High-level steps to deployment

Figure 1 shows the high-level steps for deploying the solution.

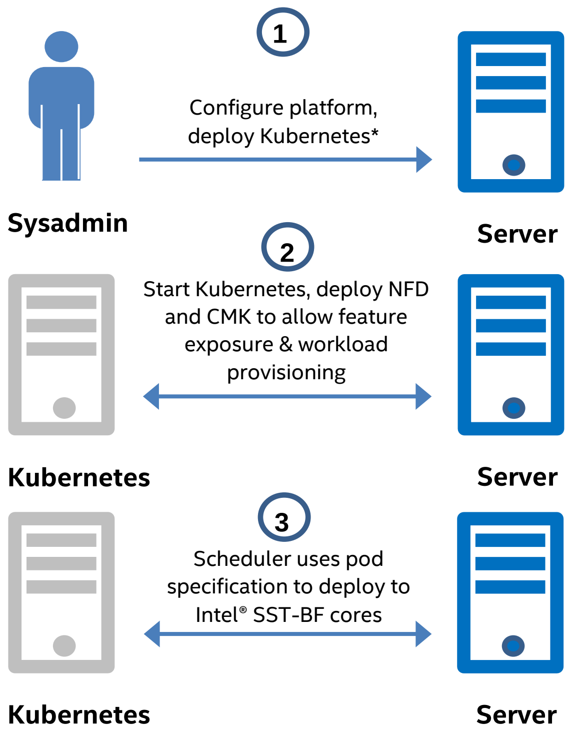

1
Configure platform, deploy Kubernetes*
Sysadmin
Server
2
Start Kubernetes, deploy NFD and CMK to allow feature exposure & workload provisioning
Kubernetes
Server
3
Scheduler uses pod specification to deploy to Intel® SST-BF cores
Kubernetes
Server

Figure 1. High-level deployment steps

1. The sysadmin configures the platform using the BIOS and scripts that set up Intel® SST-BF core frequencies. The sysadmin then deploys Kubernetes and the Kubernetes cluster is configured. See: https://github.com/intel/CommsPowerManagement for more information.
2. Kubernetes starts enabling feature exposure using Node Feature Discovery (NFD), and workload provisioning using CPU Manager for Kubernetes (CMK). On completion, the Kubernetes cluster is ready for workload deployments.
3. The Kubernetes scheduler uses the pod specification to deploy to Intel® SST-BF high- or standard-frequency cores. On completion, containers run on high-frequency or standard-frequency cores.

3 Key elements

The following elements are key to the proposed solution:
1. A new script that enables the exposure and provisioning of the Intel® SST-BF feature on selected Intel® Xeon® processors. See: https://github.com/intel/CommsPowerManagement/
2. Node feature discovery capability using NFD. See: https://github.com/kubernetes-sigs/node-feature-discovery
3. CPU Manager for Kubernetes* (CMK) to provide core affinity for NFV-style workloads. See: https://github.com/intel/CPU-Manager-for-Kubernetes

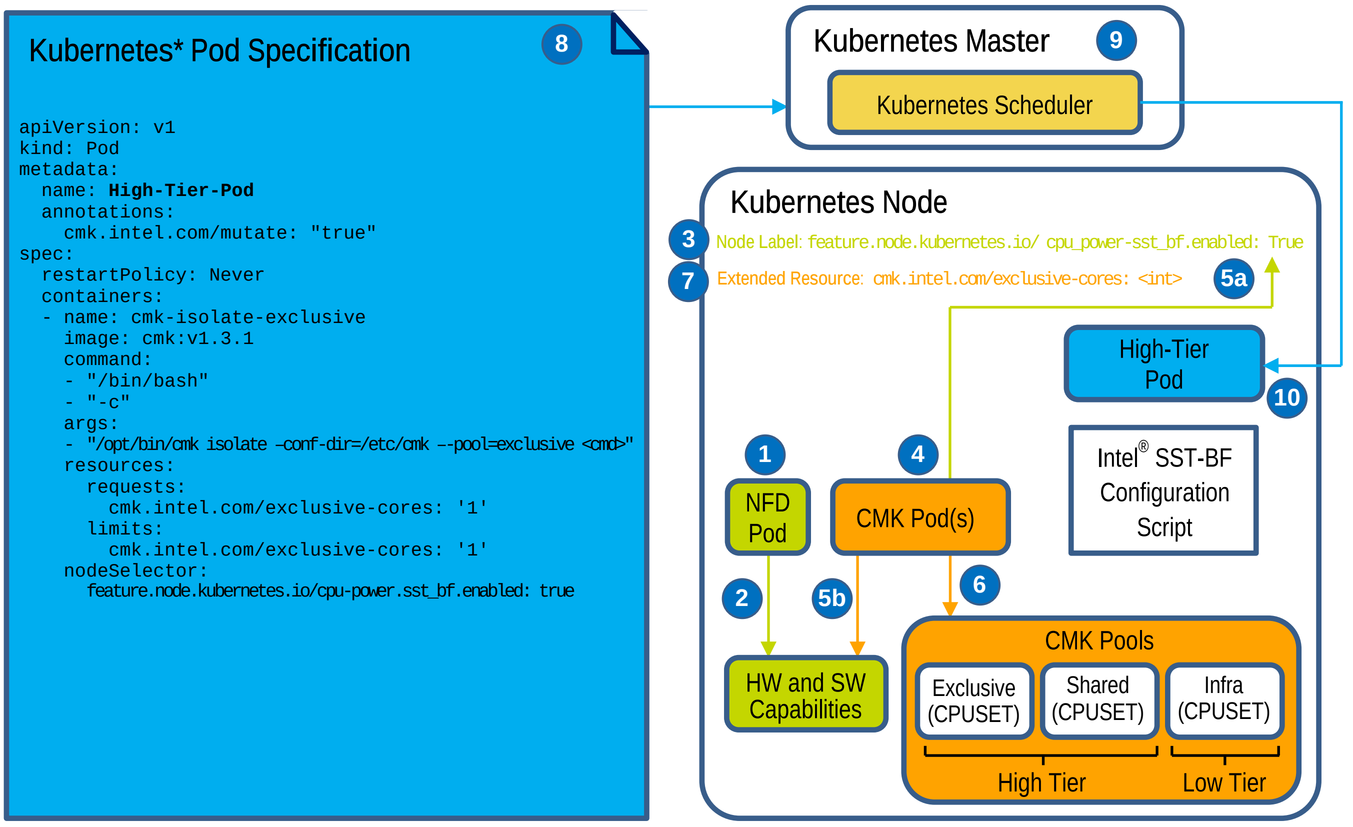
4 Architecture

4.1 Prerequisites

Prerequisites in this context are used to apply a level of platform configuration before deploying Kubernetes.
- Intel® SST-BF is supported on select SKUs of 2nd generation Intel® Xeon® Scalable processor (5218N, 6230N, and 6252N).
- Configure Intel® SST-BF via a Python* script that has several options. See: https://github.com/intel/CommsPowerManagement/
- If CPU isolation is necessary at the system level, a node must be pre-configured using `isolcpus`, where Intel® SST-BF cores are a subset of the cores specified using the `isolcpus` command.
- A basic Kubernetes cluster that can deploy pods is up and running. The cluster contains a master node (host) with several agents (daemons). See https://kubernetes.io/docs/concepts/overview/components/

More information on the components and Kubernetes installation can be found at:
https://networkbuilders.intel.com/network-technologies/container-experience-kits

Figure 2 shows the sequence of steps in a typical deployment.



```
apiVersion: v1
kind: Pod
metadata:
  name: High-Tier-Pod
  annotations:
    cmk.intel.com/mutate: "true"
spec:
  restartPolicy: Never
  containers:
  - name: cmk-isolate-exclusive
    image: cmk:v1.3.1
    command:
    - "/bin/bash"
    - "-c"
    args:
    - "/opt/bin/cmk isolate -conf-dir=/etc/cmk --pool=exclusive <cmd>"
    resources:
      requests:
        cmk.intel.com/exclusive-cores: '1'
      limits:
        cmk.intel.com/exclusive-cores: '1'
    nodeSelector:
      feature.node.kubernetes.io/cpu-power.sst_bf.enabled: true
```

Figure 2. High-level architecture and flow diagram

4.2 Deployment

Refer to the corresponding step numbers in

Figure 2.

1. At Kubernetes start, deploy an NFD `daemonset` on the cluster. This configures each node in the cluster to run an NFD pod.
2. NFD discovers if the node is Intel® SST-BF capable and sets a label to "`true`", for example: `feature.node.kubernetes.io/cpu-power.sst_bf.enabled`
3. NFD advertises this to Kubernetes using a node label (which is `true` if Intel® SST-BF is possible or `false` otherwise).
4. Deploy the CMK `cluster-init` pod on the Kubernetes cluster. This pod creates additional pods on any nodes in the cluster that must have the CMK configured. The `cluster-init` pod is a YAML configuration file created by the sysadmin to deploy on the cluster.
5. CMK performs platform discovery as follows:
   a. The CMK communicates with the Kubernetes `api-server` master node and checks for the presence of the Intel® SST-BF capable label on each node in the cluster.
   b. On detection of the Intel® SST-BF label, CMK interacts with `sysfs` to identify the cores operating as Intel® SST-BF cores.

   The CMK also checks for CPU isolation:
   a. If `isolcpus` is configured, the CMK reads these CPU IDs and uses them to create the exclusive and shared pools.
   b. If not, the CMK creates exclusive and shared pools based on the lowest CPU IDs.
6. The CMK creates the core pools based on the information gathered in step 5. The categories of pools created are:
   – Exclusive (isolated cores)
   – Shared (isolated cores)
   – Infrastructure (shared cores)

   Intel® SST-BF aligns with CMK pool categories as follows:
   – Exclusive pool = High-frequency cores (Linux* scheduler isolated cores)
   – Shared pool = Standard-frequency cores (Linux scheduler isolated cores)
   – Infrastructure pool = Standard-frequency cores (no isolation by Linux scheduler)

Figure 3 shows the CMK pools and how they can apply in a Kubernetes implementation.

Note: CMK uses a file system hierarchy for tracking pools, their assigned CPUs, pool exclusivity, and the processes currently allocated to CPUs. See Section 4.6 for details.

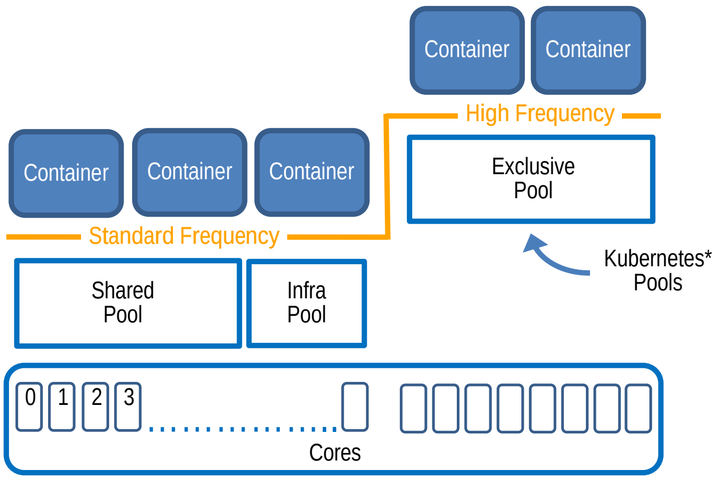


Figure 3. CMK pool assignment and Intel® SST-BF core association

7. The CMK advertises the number of cores in the exclusive pool using extended resources. For example:
`Extended Resource:  cmk.intel.com/exclusive-cores: '1'`
8. The Kubernetes pod specification (a user-created YAML file) contains the following:
   – The extended resource request for 1 to N cores (if desired).
   – The CMK command line arguments including desired pool.
   – The node selector for an Intel® SST-BF capable node.
9. The user does a `kubectl` create of the pod. The Kubernetes scheduler detects the pod, looks at the extended resource request and node selector, and schedules the pod on the most appropriate node in the cluster.
10. The CMK on the host platform pins the incoming pod to the requested pool and the requested number of cores. The high-tier cores and the infrastructure cores are advertised to the container using environment variables.

4.3 Configuring a container to use standard- and high-frequency cores

Optionally, a workload may be aware of the cores and may want to pin its own threads to standard- and high-frequency cores as shown in Figure 4. The workload can achieve this using a query with the environment variables that CMK provides.

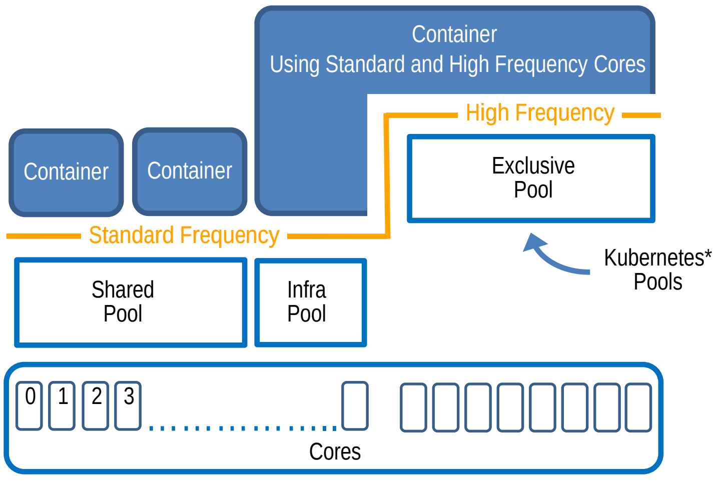


Figure 4. Container using standard- and high-frequency cores

The following bash script is an example.

```
#!/bin/bash
CORES=`printenv CMK_CPUS_ASSIGNED`
INFRACORES=`printenv CMK_CPUS_INFRA`
PSR=`ps -ax -o pid,psr,comm | grep $$ | awk '{ print $2 }'`
```

```
if [ -z "$CORES" ]
then
      echo "CPU cores not isolated by CMK! Used core: $PSR"
else
      echo "CPU cores isolated: $CORES. Used core: $PSR"
fi
echo "Infrastructure cores: $INFRACORES"
COMMAND=${@//'$CORES'/$CORES}
$COMMAND
```

The output from the example bash script is the following.

```
# CPU cores isolated: 0,44. Used core: 0
Infrastructure cores:
5,49,6,50,7,51,8,52,9,53,10,54,11,55,12,56,13,57,14,58,15,59,16,60,17,61,18,62,19,63,20,64,21,65,22
,66,23,67,24,68,25,69,26,70,27,71,28,72,29,73,30,74,31,75,32,76,33,77,34,78,35,79,36,80,37,81,38,82
,39,83,40,84,41,85,42,86,43,87
```

The CMK provides configurable user-supplied options for the CPU pinning policy for a workload. The options are:

- `--affinity`: The CMK handles the CPU pinning of the workload to the allocated requested cores, in this context, high-tier Intel® SST-BF cores.
- `--no-affinity`: The CMK allocates desired cores to the workload, however, it does not do the actual pinning. The workload has access to the assigned cores (high-tier Intel® SST-BF) and infrastructure (standard tier) cores and takes care of the required workload pinning.

The following are example commands showing the use of the options in a Kubernetes pod specification.

```
/opt/bin/cmk isolate --conf-dir=/etc/cmk
--pool=exclusive --affinity <cmd>

/opt/bin/cmk isolate --conf-dir=/etc/cmk
--pool=exclusive --no-affinity <cmd>
```

If a workload requires two-tier pinning, meaning the threads must be pinned to high-frequency and standard cores, then the environment variables and the `--no-affinity` option are necessary. For example, DPDK applications can instruct (in the launch command) that the starting threads run on specific CPU cores, meaning workload changes are not necessary.

If the entire workload must be pinned to a high-frequency core, then it is not necessary to use environment variables and the default affinity option is used.

4.4 Incorporating a DPDK command in a pod specification

The following is an example of a Data Plane Development Kit (DPDK) command in a Kubernetes pod specification.

```
# /opt/bin/cmk isolate --conf-dir=/etc/cmk --pool=exclusive --no-affinity /etc/cmk/use_cores.sh
'testpmd -l \$CORES -m 512 -w 0000:0c:00.1 -- -i'
```

Note: The $CORES variable in this command comes from the bash script provided earlier.

4.5 Kubernetes scheduler result for insufficient resources

If there are no nodes in the cluster that have enough resources available for the incoming pod requests, the Kubernetes scheduler provides a result like the following.

```
# NAME               READY     STATUS      RESTARTS    AGE
cmk-isolate-pod-2    0/3       Pending     0           3s

Warning  FailedScheduling  1m (x2 over 1m)  default-scheduler  0/1 nodes are available: 1
Insufficient cmk.intel.com/exclusive-cores.
```

4.6 Tracking of pools, CPU exclusivity, and processes

Figure 5 shows the file system hierarchy that CPU Manager for Kubernetes* uses to track pools, their assigned CPUs, pool exclusivity, and the processes currently allocated to CPUs.

```
etc
 |
 + - cmk
      |
      + - lock
      |
      + - pools
            |
            + - shared
            |      |
            |      + - 0                 # Socket ID
            |      |   |
            |      |   + - 0,1           # logical core IDs
            |      |        |
            |      |        + - tasks    # contains PIDs
            |      |
            |      + - exclusive         # Determine exclusivity
            |
            + - exclusive
            |      |
            |      + - 0
            |      |   |
            |      |   + - 2,3
            |      |   |    |
            |      |   |    + - tasks
            |      |   |
            |      |   + - 4,5
            |      |        |
            |      |        + - tasks
            |      |
            |      + - exclusive
            |
            + - infra
                   |
                   + - 0
                   |   |
                   |   + - -6
                   |        |
                   |        + - tasks
                   |
                   + - exclusive
```

Figure 5. CPU Manager for Kubernetes* file system hierarchy

5 Summary

Select SKUs of 2nd generation Intel® Xeon® Scalable processor (5218N, 6230N, and 6252N) offer a new capability called Intel® Speed Select Technology – Base Frequency (Intel® SST-BF). This document described how to utilize the Intel® SST-BF feature in a cluster managed by Kubernetes* (K8S*) container orchestration.