Node Feature Discovery
NFD and My Adventure in the Cloud Native Project Jungle
Markus Lehtonen
• My Journey

• Resource Management in Kubernetes

• Node Feature Discovery (NFD)

• Demo

• Future of NFD
Introduction

Markus Lehtonen
Cloud Software Engineer, Intel
@marquiz
How I Got Involved

• Background in embedded
• Hopped in the K8s wagon in 2018
• NFD needed care
• Fun and welcoming year
Resource Management in K8s

**POD SPEC**
name: my-pod
...
resources:
  requests:
    memory: 512Mi

**NODE**
NATIVE RESOURCES
  cpu
  memory
  hugepages

**EXTENDED RESOURCES**
gpu
fpga
...

**MASTER**
API SERVER

**SCHEDULER**

**ETCD**

**NODE**
KUBELET

**Allocate resources**

Create object

Advertise resources

Allocate resources

Schedule pod

Schedule pod
What About Node Features?

WHAT
Platform capabilities
Non-allocatable, ”unlimited” resources

WHY
Heterogenous clusters
Strict workload requirements
Workload performance improvements
What About Node Features?

HOW

Node Feature Discovery – NFD
  • Fills the gap
  • Running one instance per Node
  • Advertise features as Node labels

Node labels can be used in workload spec
Node Feature Discovery

• Sponsored by SIG Node

• Under Kubernetes-SIGs in Github
  
  • github.com/kubernetes-sigs/node-feature-discovery
NFD – Architecture

[Diagram showing NFD architecture with MASTER, NODE, NFD MASTER, API SERVER, and gRPC connections]
NFD-Master

NFD-MASTER

gRPC server

- mTLS
- Label filtering
- Node Update
  - annotations
  - labels

API-SERVER

gRPC

updateNode
NFD-Worker

NFD-WORKER

Feature Sources
- cpu
- iommu
- kernel
- memory
- network
- pci
- local
- storage
- system

gRPC client

NFD-MASTER
Feature Sources

Discovery organized into a hierarchy of sources

<table>
<thead>
<tr>
<th>HW SOURCES</th>
<th>OTHER SOURCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kernel</td>
</tr>
<tr>
<td>PCI</td>
<td>System</td>
</tr>
<tr>
<td>IOMMU</td>
<td>Local (custom hooks)</td>
</tr>
<tr>
<td>Memory</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td></td>
</tr>
</tbody>
</table>

Node feature labels

```
feature.node.k8s.io/<source name>-<feature name>[.<attribute name>]=<value>
```

Re-discovery/re-label every 60s (by default)
<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpuid</td>
<td>&lt;cpuid flag&gt;</td>
<td>CPU capability is supported</td>
</tr>
<tr>
<td>hardware_multithreading</td>
<td>n/a</td>
<td>Hardware multithreading, such as Intel® HTT, enabled</td>
</tr>
<tr>
<td>power</td>
<td>sst_bf.enabled</td>
<td>Intel® SST-BF (Intel Speed Select Technology - Base frequency) enabled</td>
</tr>
<tr>
<td>pstate</td>
<td>turbo</td>
<td>Turbo freq. are enabled in pstate driver</td>
</tr>
<tr>
<td>rdt</td>
<td>RDTMON</td>
<td>Intel® RDT Monitoring Technology</td>
</tr>
<tr>
<td></td>
<td>RDTCMT</td>
<td>Intel® Cache Monitoring</td>
</tr>
<tr>
<td></td>
<td>RDTMBM</td>
<td>Intel® Memory Bandwidth Monitoring</td>
</tr>
<tr>
<td></td>
<td>RDTL3CA</td>
<td>Intel® L3 Cache Allocation Technology</td>
</tr>
<tr>
<td></td>
<td>RDTL2CA</td>
<td>Intel® L2 Cache Allocation Technology</td>
</tr>
<tr>
<td></td>
<td>RDTMBA</td>
<td>Intel® Memory Bandwidth Allocation Technology</td>
</tr>
</tbody>
</table>
# Feature Sources – Kernel

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config</td>
<td>&lt;option name&gt;</td>
<td>Kernel config option is enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true for bool/tristate options (set 'y' or 'm')</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• config value for other options (str, int, ...)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defaults: NO_HZ, NO_HZ_IDLE, NO_HZ_FULL, PREEMPT</td>
</tr>
<tr>
<td>selinux</td>
<td>enabled</td>
<td>Selinux is enabled on the node</td>
</tr>
<tr>
<td>version</td>
<td>full</td>
<td>Full kernel version (e.g. '4.5.6-7-g123abcde')</td>
</tr>
<tr>
<td>version</td>
<td>major</td>
<td>First component of the kernel version (e.g. '4')</td>
</tr>
<tr>
<td>version</td>
<td>minor</td>
<td>Second component of the kernel version (e.g. '5')</td>
</tr>
<tr>
<td>version</td>
<td>revision</td>
<td>Third component of the kernel version (e.g. '6')</td>
</tr>
</tbody>
</table>

**Configurable:**
- Kconfig file to read
- Kconfig options to discover
<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>os_release</td>
<td>ID</td>
<td>Operating system identifier (e.g. 'centos')</td>
</tr>
<tr>
<td></td>
<td>VERSION_ID</td>
<td>Operating system version identifier (e.g. '7.6')</td>
</tr>
<tr>
<td></td>
<td>VERSION_ID.major</td>
<td>First component of the OS version id (e.g. ’7’)</td>
</tr>
<tr>
<td></td>
<td>VERSION_ID.minor</td>
<td>Second component of the OS version id (e.g. ’6’)</td>
</tr>
</tbody>
</table>
### Feature Sources – Memory

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numa</td>
<td>n/a</td>
<td>Multiple memory nodes i.e. NUMA architecture</td>
</tr>
<tr>
<td>nv</td>
<td>present</td>
<td>NVDIMM device(s) are present</td>
</tr>
<tr>
<td>dax</td>
<td></td>
<td>NVDIMM DAX mode regions are present</td>
</tr>
</tbody>
</table>
## Feature Sources – PCI

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;device label&gt;</code></td>
<td>present</td>
<td>PCI device is detected. Defaults: GPU and accelerator cards are detected</td>
</tr>
</tbody>
</table>

`<device label>` is composed of raw PCI IDs, separated by underscores.

**Configurable:**
- Fields that `<device_label>` contains
- Device classes that are detected
## Feature Sources – Network

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sriov</td>
<td>capable</td>
<td>Single Root Input/Output Virtualization (SR-IOV) enabled Network Interface Card(s) present</td>
</tr>
<tr>
<td></td>
<td>configured</td>
<td>SR-IOV virtual functions have been configured</td>
</tr>
</tbody>
</table>
### IOMMU FEATURES

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>n/a</td>
<td>An IOMMU is present and enabled in the kernel</td>
</tr>
</tbody>
</table>

### STORAGE FEATURES

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonrotationaldisk</td>
<td>n/a</td>
<td>Non-rotational disk, like SSD, is present in the node</td>
</tr>
</tbody>
</table>
Feature Sources – Local

- User-specific feature detection
- Custom feature sources in a pluggable way
- Create new labels / override existing labels
- Two mechanisms
  - Hooks
  - Feature files
Feature Sources – Local Hooks

- Execute files from 
  /etc/kubernetes/node-feature-discovery/source.d/

- Stdout is turned into labels
  
  [[<label ns>]/<source name>-]<feature name>[=<value>]

- stderr directed to NFD logs

STDOUT FROM  my_source

my_bool
my_non_bool=myvalue
/override_src-my_feature_1
/override_src-my_feature_2=123
my.namespace/value=k8s

NODE LABELS CREATED

feature.node.kubernetes.io/my-source-my_bool=true
feature.node.kubernetes.io/my-source-my_non_bool=myvalue
feature.node.kubernetes.io/override_src-my_feature_1=true
feature.node.kubernetes.io/override_src-my_feature_2=123
my.namespace/value=k8s
Read files from
/etc/kubernetes/node-feature-discovery/features.d/

File content is turned into labels

Format is similar to hooks
[[<label ns>]/<source name>-]<feature name>[=<value>]]
Configuration Options

- nfd-worker has (optional) configuration file

- Three configurable sources
  - cpu
  - kernel
  - pci
NFD Deployment

**DS+DS**
- **Node #1**: NFD-Master, NFD-Worker
- **Node #2**: NFD-Master, NFD-Worker

**DS+JOB**
- **Node #1**: NFD-Master, NFD-Worker
- **Node #2**: NFD-Master, NFD-Worker

**DS COMBINED**
- **Node #1**: NFD-Master, NFD-Worker
- **Node #2**: NFD-Master, NFD-Worker
nodeSelector:
  <label>: <value>
nodeAffinity:
  requiredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
      - matchExpressions:
        - key: <label>
          operator: {In|NotIn|Exists|DoesNotExist|Gt|Lt}
          values: [<list of values>]

  preferredDuringSchedulingIgnoredDuringExecution:
    weight: <integer>
    preference:
      matchExpressions:
        - key: <label>
          operator: {In|NotIn|Exists|DoesNotExist|Gt|Lt}
          values: [<list of values>]
Workload Deployment – nodeAffinity

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - image: k8s.gcr.io/pause
    name: pause
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: feature.node.kubernetes.io/kernel-version.major
            operator: Gt
            values: ['3']
          - key: feature.node.kubernetes.io/kernel-version.minor
            operator: Gt
            values: ['4']
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          preference:
            matchExpressions:
            - key: feature.node.kubernetes.io/cpu-hardware_multithreading
              operator: NotIn
              values: ['true']
```

Demo Time

- Deploy NFD v0.4.0 on a cluster
- Deploy Intel® GPU device plugin on GPU-capable node(s)
The (Near) Future

• NFD operator
• Multi-arch builds
• Usability and configurability improvements
• Support for Taints (?)
• Support Extended Resources (?)

• Project logo ;)
• The usual boring
  • Usage examples
  • Documentation
Want To Help?

• Feature requests

• Patches

• Tell us your story

https://github.com/kubernetes-sigs/node-feature-discovery
THANK YOU