2024 OFA Virtual Workshop

HOW TO SETUP RDMA CI USING THE FSDP CLUSTER

Doug Ledford, Chair OFA

Red Hat, Inc.

- Create your tests
- Add your tests to the kpetdb
- Add your pipeline to the ofa pipeline in kci
- Sit back and wait for results

- **Watch the first video on the left to see how to get setup with your account and access the FSDP**
- **Watch the second and third videos for an introduction to the testing API (everything is based on the Restraint test harness as shipped with Beaker: https://restraint.readthedocs.io/en/latest/)**
- **Clone the FSDP test repo:**
  - https://github.com/OpenFabrics/fsdp_tests.git
  - git@github.com:OpenFabrics/fsdp_tests.git
  - gh repo clone OpenFabrics/fsdp_tests
- **Add your own unique directory for your tests**
  - You will need 1 directory per unique test, mainly for selecting the pool specific to how you want to run your tests, may use a shared directory for the majority of the test itself
  - Eg.
    - Redis/common <- Generic copy of the test that can be shared
    - Redis/ib-mlx <- Could use either generic or specific pool here as there is only one IB vendor in the cluster
    - Redis/roce-mlx <- Need to use a specific pool to limit devices to mlx roce
    - Redis/roce-qedr <- Need to use a specific pool to limit devices to qedr roce
    - Redis/iwarp <- Use the generic iwarp pool and utilize any available iwarp devices, may end up with either heterogeneous or homogeneous mix of devices

Webinar Video

Webinar Video

Webinar Video

```
function client {
    echo "--- wait for server to get ready - ${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER} ---"
    rhts_sync_block -s "server-ready_${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER}" ${SERVERS}
    rhts_sync_set -s "client-ready_${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER}"

    # do the sanity test
    bash -x ./tier1.sh client

    # Report the result
    echo "--- client finished - ${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER} ---"
    rhts_sync_set -s "client-done_${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER}"
}
```

```
function server {
    # server get ready
    echo "--- server is ready - ${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER} ---"
    rhts_sync_set -s "server-ready_${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER}"
    rhts_sync_block -s "client-ready_${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER}" ${CLIENTS}

    # do the sanity test
    bash -x ./tier1.sh server

    # Report the result
    echo "--- server finishes - ${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER} ---"
    rhts_sync_block -s "client-done_${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER}" ${CLIENTS}
}
```

```
#--- Start test -------------------------------------------------

bash -x ./tier1.sh common

if hostname -A | grep ${CLIENTS%%.*} >/dev/null ; then
    echo "******** client test start ********"
    echo "**** ${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER} ***"
    client
    TEST=${TEST}/sanity/client
elif hostname -A | grep ${SERVERS%%.*} >/dev/null ; then
    echo "******** server test start ********"
    echo "******** ${TNAME}-${RDMA_NETWORK}-${RUN_NUMBER} ********"
    server
    TEST=${TEST}/sanity/server
fi
```

```
function common_tests {
    # ensure appropriate ib/iw/en/opa drivers are
available in this kernel
    driver_modules=$(RQA_get_driver_modules)
    for module in $driver_modules; do
        lsmod | grep -i "^${module} "
        RQA_check_result -r $? -t "load module
${module}"
    done
```

- **This is the appropriate place to build/install user space packages needed for your test**

Specifically, for the redis packages Jeremy has built:

wget --no-check-certificate https://builder-00.ofa.iol.unh.edu/~jspewock/server.rpm && yum install -y ./server.rpm

And

wget --no-check-certificate https://builder-00.ofa.iol.unh.edu/~jspewock/client.rpm && yum install -y ./client.rpm

This command should be added to the common section. After the common section, but before the tests are run, we wait for an “all ready” sync up between server and client, so building and installing our code we wish to test during the common section works fine

```
function server_tests {
    # ping client. If it fails, the following multi-host tests would
fail.
    if [[ -z $CLIENT_IPV4 ]]; then
...

function client_tests {
    # ping server. If it fails, the following multi-host tests would
fail.
    if [[ -z $SERVER_IPV4 ]]; then
...
```

- **This is the appropriate place to hook in your test commands**

Specifically, for the redis packages Jeremy has built:

Server test command:

```
redis-server --port 6379 --loadmodule
/etc/redis/redis-rdma.so port=6379
bind=XX.XX.XX.XX --loglevel verbose --
protected-mode no --server_cpulist 2 --
bio_cpulist 3 --aof_rewrite_cpulist 3 --
bgsave_cpulist 3 --appendonly no
```

Client test command:

```
redis-benchmark -h xx.xx.xx.xx -p 6379 -c 30 -
n 10000000 --threads 4  -d 1024 -t
ping,get,set --rdma
```

Creating host_types in kpetdb/hosts_types (if needed)

- **Existing host_types:**
  - **ofa-ib.host_requires.xml.j2**
  - **ofa-iwarp.host_requires.xml.j2**
  - **ofa-opa.host_requires.xml.j2**
  - **ofa-roce.host_requires.xml.j2**
- **This only pull in generic hosts on a fabric. For specific hosts on a specific fabric (eg. for mlx specific roce), you will need to create a host type that pulls the rights hosts in using a beaker pool specifier specific to the hosts you want**

Example creation of roce-mlx hosts_type file

ofa-roce-mlx.host_requires.xml.j2:

<pool op="=" value="roce-mlx"/>

You can see all of the available pool types by checking the pools out on the beaker web interface for the FSDP cluster. If we need to create additional pools to specific specific host hardware, that is easily doable.

If you created a new hosts_types, you also need to link it into the host_types in index.yaml

 # OFA RoCE tests

ofa-roce_1:

domains:

ofa host_requires: host_types/ofa-roce.host_requires.xml.j2

ofa-roce_2:

domains:

ofa host_requires: host_types/ofa-roce.host_requires.xml.j2

Example linking of roce-mlx hosts_type file, add the following to index.yaml

 # OFA RoCE-MLX tests

ofa-roce-mlx_1:

domains:

ofa host_requires: host_types/ofa-roce-mlx.host_requires.xml.j2

ofa-roce-mlx_2:

domains:

ofa host_requires: host_types/ofa-roce-mlx.host_requires.xml.j2

And you nbeed to create a recipeset for the new hosts in index.yaml:

```
recipesets:
  multihost:
      ...
    ofa_roce:
      - ofa-roce_1
      - ofa-roce_2
    ofa_roce_mlx:
      - ofa-roce-mlx_1
      - ofa-roce-mlx_2
```

Then create your test entry in the cases directory

mkdir –p cases/ofa/redis/roce <-Please keep ofa tests under the ofa parent directory

vi cases/ofa/redis/roce/index.yaml <- Yes, Vi does in fact rule the universe!

Sample cases/ofa/redis/roce/index.yaml

```
name: ‘Redis RoCE’
location: redis
target_sources:
  - drivers/infiniband/core/.*
  - drivers/infiniband/hw/.*
max_duration_seconds: 7200
environment:
  ENV_DRIVER: ""”
cases:
  RoCE: cases/ofa/hosts/RoCE/index.yaml
sets:
  ofa-roce
```

Fields

Name: Name of your test, and needs to be unique

Location: top level directory in the fsdp_tests repo for your test (this need not be fabric specific)

Target Sources: kernel directories that, if they change, should trigger retesting (not used on user space repos)

Max Duration Seconds: how long until a hung test times out

Environment: Random items you need set for your test

Cases: Can point to sub-tests in the kpetdb cases subtree. HOWEVER! All sub-tests need to have the same machine requirements. If you mix a test that requires IB with a test that requires RoCE, the scheduler will be unable to resolve a machine to satisfy the test and the tests will just get skipped

Sets: Upper level test sets that this test claims to be a part of. Test sets are defined in the main index.yaml file

Link the cases/ofa/redis/roce/index.yaml into the cases/ofa/index.yaml file

```
sets:
 - ofa
cases:
 ofa_kernel_rdma: cases/ofa/kernel-rdma/index.yaml
 ofa_user_rdma: cases/ofa/user-rdma/index.yaml
 ofa_libfabric: cases/ofa/libfabric/index.yaml
 ofa_redis_roce: cases/ofa/redis/roce/index.yaml
```

Note: The existing kernel_rdma, user_rdma, and libfabric entries combine all the different fabrics under a single grouping and are, as a result, all skipped. This is known. Don't follow their example if you want your tests to actually be run. We will split them out to the individual fabric types when we enable these tests.

Link new test(s) into the ofa pipelines by adding the follow to the ofa.yaml file

redis-roce: **<- keep the fabric name in the test name since you will need a separate test for each fabric you want to run on**

git_url: https://git.kernel.org/pub/scm/linux/kernel/git/rdma/rdma.git **<- should point to the upstream rdma kernel tree. This item is used to build the latest kernel for your tests**

tests_regex: 'OFA – Redis RoCE.*' **<- This name must match OFA - <test name> from your keptdb index.yaml name entry**

.branches:

- for-next **<- This is the kernel to build and test against**

watch_url: https://github.com/redis/redis

watch_branch: <branch to watch>

- The CKI pipeline will watch your watch_url repo on watch_branch branch. When new commits are added, it triggers a run.
- First step in the run is to build the kernel specified at the git_url and .branches option (or reuse pre-built artifacts if they exist)
- When the artifacts are ready, the system pulls up your test in kpetdb to get the necessary test info
- The system then uses that to find your actual test files and pull out any needed beaker xls/xml files
- The system then attempts to process the test info to generate beaker xml files suitable for automated job submission
- The system will use the requirements and restrictions in the hosts_types and other fields to select the right hosts and the right cluster for the job
- The system then attempts to submit the job to the beaker controller
- Once submitted, beaker attempts to run the jobs, which will result in the tests tarball being downloaded and installed
- From the contents of the tarball, your specific runtest.sh script will be ran using the top level dir. you specified in the kpetdb entry for your test
- At this point, your test script is in control and can free-form run the tests as you see fit\

2024 OFA Virtual Workshop

THANK YOU

2024 OFA Virtual Workshop

How to do Manual RDMA Testing Using the FSDP Cluster

Jeremy Spewock, Lead Developer

UNH InterOperability Lab (IOL)

University of New Hampshire
InterOperability
Laboratory

Overview

- **Introduction To The Cluster**
  - Brief overview of the hardware available
  - Showcase of pre-configured fabrics
- **Tour of Beaker**
  - What is Beaker?
  - Why do we use Beaker?
  - How do I reserve nodes using Beaker?
- **Redis and RDMA**
  - Brief explanation of Redis
  - RDMA support in redis
- **Demo**

THE CLUSTER

The FSDP Cluster

Hardware

- **Consists of 10 nodes and a build server**
  - Nodes
    - Bare-metal hosts with RDMA cards
    - Grouped by RDMA fabric
    - Provisioned by Beaker
  - Builder-00
    - Static environment for building binaries
    - Persistent home directory for your user
    - public_html for easy transportation of binaries

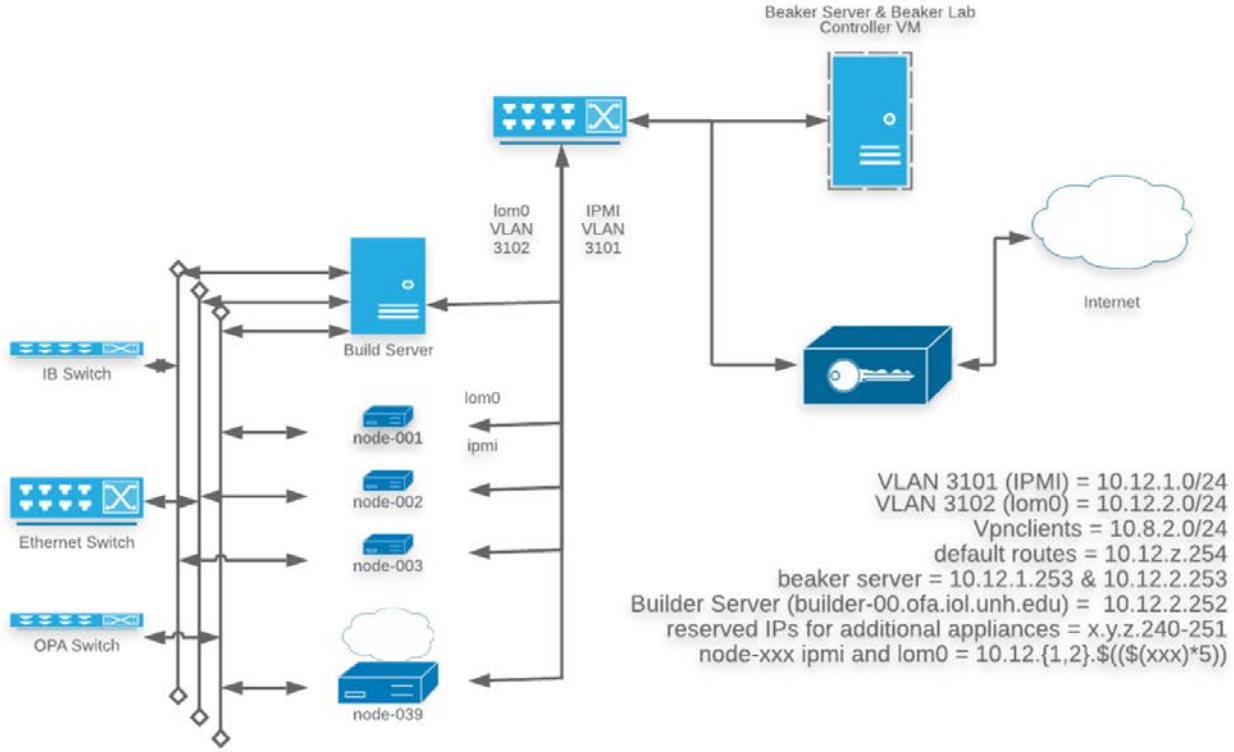


Image is physical section of the cluster network image from the FSDP Docs GitHub repository (https://github.com/OpenFabrics/fsdp_docs)

The FSDP Cluster

Software/Fabrics

- **Nodes can be grouped by fabric**
  - iwarp
  - Infiniband
  - RoCE
  - Omni-Path

Test Networks (exist only on the fabric switches)
ib0 = 172.31.0.0/24
ib0.8002 = 172.31.2.0/24
ib0.8004 = 172.31.4.0/24
ib0.8006 = 172.31.6.0/24
opa0 = 172.31.20.0/24
opa0.8022 = 172.31.22.0/24
opa0.8024 = 172.31.24.0/24
roce = 172.31.40.0/24
roce.43 vlan = 172.31.43.0/24
roce.45 vlan = 172.31.45.0/24
iwarp = 172.31.50.0/24
iwarp.51 vlan = 172.31.51.0/24
iwarp.52 vlan = 172.31.52.0/24
no default routes
build server 172.31.z.252
reserved IPs for DHCP* = 172.31.z.200-239
node-xxx rdma 172.y.z.{$(($(xxx)*5))..$(($(xxx)*5+4))}

Image is software section of the cluster network image from the FSDP Docs GitHub repository (https://github.com/OpenFabrics/fsdp_docs)

BEAKER

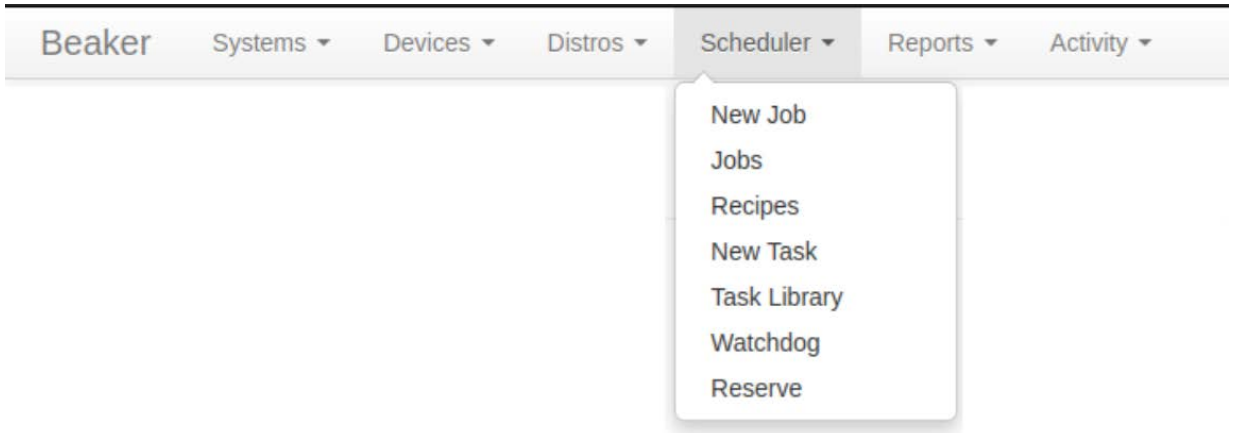
Beaker Overview

- **What is Beaker?**
  - Open-source software for managing automated labs
  - Capable of storing multiple OS distributions to select from when provisioning hosts
  - Maintains inventories of hardware on nodes
  - Able to store a library of tasks for ease of scheduling testing across multiple environments
- **Why do we use it?**
  - Makes scheduling the same test on multiple fabrics simple
    - Hosts can be grouped into “pools” and jobs scheduled from hosts within a pool
      - Allows us to create pools based on fabric for convenience
  - Allows for testing the difference in functionality between distros
  - XML style for writing jobs is easy to follow and reproduce
  - Ability to specify what are called “snippets” in beaker

Beaker Overview

Primary Testing Methods

- **The manual process**
  - Use the built in task called “reserve” in the web GUI
    - Provisions the host for access via SSH
  - Gives you a non-persistent testing environment
  - Able to reach HTTP endpoints on builder-00
- **The automated process**
  - Write XML jobs using beaker’s format
  - Can create custom tasks and package them into RPMs
  - Allows for easy re-use and sharing of tasks
  - Host provisioned for the task, the task is run, and then host is immediately freed*



Images illustrate where to access both the manual and automated processes for tasks, taken from the FSDP beaker instance (https://beaker.ofa.iol.unh.edu/bkr/).

***Some tasks can be used to reserve the system for longer periods of time.**

TESTING RDMA SUPPORT IN REDIS

What is Redis?

- **Redis**
  - High-speed in-memory database
  - A “data structures server”
    - A server that exposes multiple mutable data structures via a set of commands
  - Server-client model that uses a simple TCP protocol for sending commands
  - Non-volatile
    - Even if the data structures are always in-memory, redis still saves them onto disk
    - Maintains speed while still keeping data

Redis and RDMA

- **RDMA support in Redis**
  - Ticket for RDMA support opened in July, 2021 (issue 9292)
  - Pull request (PR 11182) was opened for server-side support with 2 main blockers:
    - A general lack of knowledge on RDMA from Redis maintainers
    - A call for testing the RDMA support across different fabrics
      - Only tested by the submitter on RXE (soft-RoCE) and RoCE using a ConnectX-5 net card
  - AWS has a partial implementation of proprietary RDMA capability
  - Azure only supports Mellanox Infiniband
  - This is where the FSDP cluster comes in!

DEMO

2024 OFA Virtual Workshop

THANK YOU

Jeremy Spewock, Lead Developer

UNH InterOperability Lab

University of New Hampshire
InterOperability
Laboratory