How to bind Dockers to NUMA nodes

Overview

In this document I will introduce how to run the database dockers, e.g. redis, memcached, mongodb, on the specified NUMA node. Procedures should be similar if you want to run other kinds of dockers.

Step1. Change the docker - install `numactl`

Specify the dockers you want to bind to a specific NUMA node. For example, I want to bind the redis server, memcached, and mongodb to the remote node. We should first launch the docker in interactive mode to see if `numactl` tool is supported:

```
sudo docker run -it <image_id> /bin/bash
```

You can use the following command to see the image ID:

```
sudo docker ps
```

Check if `numactl` is installed. If not, install it.

```
numactl -H
# if numactl is not found
apt-get update
apt-get install numactl
```

After you made the change, use the following commands to exit the docker and commit the changes:

```
exit
sudo docker ps -a # to find the container ID
sudo docker commit [CONTAINER_ID] [new_image_name] # e.g. redis-numa
```

Note: There is a special case for `mongo` docker. You may not commit now because further changes will be made.

Step2. Change the docker - find the entrypoint

After you make sure `numactl` is installed, we should find the entrypoint of this docker. As a practice, the first command dockers will run is usually like `docker-entrypoint.sh redis-server`. You should check dockerhub to check the docker command used for those dockers that are directly pulled from dockerhub. As an example, the entry command for `redis` docker is here. For a better understanding of the difference between `ENTRYPOINT` and `CMD` commands, you can check this answer.

The entry command you find will later be added to the `docker-compose.yml` file.

Step3. Modify `docker-compose.yml`

This file resides in the root path of a specific benchmark, e.g. `socialNetwork/`. It is good practice to "save as" the file and work on its copy. Modify a service in the file from:

```
social-graph-redis:
    image: redis
    hostname: social-graph-redis
    restart: always
```

to:

```
social-graph-redis:
    image: redis-numa // the image you created
    hostname: social-graph-redis
    privileged: true // numactl requires root privilege
    entrypoint: numactl --cpunodebind=0 --membind=1 redis-server // the entry command for the database docker with the numactl as prefix
    restart: always
```

You might have noticed that we did not use the `docker-entrypoint.sh` to start the database application. This depends on the docker. For `redis` as an example, the shell script does not really do anything, thus it is fairly safe to ignore it. However, for `mongo` the shell script does a lot of initialization jobs so it should not be ignored. In that case, you should modify the `docker-entrypoint.sh` file inside the docker to **implant** the `numactl --cpunodebind=0 --membind=1` commmand.

Note: I will skip the steps to change `docker-entrypoint.sh` here, but as a hint the `numactl` is part of the mongo docker already. The final entrypoint command is something like `docker-entrypoint.sh mongod`.

Step4. Run the dockers

Since you might want to switch between local and remote nodes, a shell script to rewrite `docker-compose.yml` is helpful. That is, you can create two `yml` files like `docker-compose-local.yml` and `docker-compose-remote.yml`, then use `cp` command to substitute the real `docker-compose.yml`.

Use the following command to set up the dockers.

```
sudo docker-compose up -d
```

Check the status of the dockers to make sure they are running normally (no restarting) and are actually started by a `numactl` command.

```
sudo docker ps
```