CPU & Memory limits in LSF with cgroups

Introduction

We want to improve the robustness and reliability of the batch system by applying tighter resource controls. The goal is to isolate jobs from each other and prevent them from consuming all the resources on a machine. LSF version 9.1.2 makes use of linux Control Groups (AKA cgroups) to limit the CPU cores and memory that a job can use. These cgroup-based restrictions are currently not in our production configuration. We want to understand the potential impact to users and get feedback from stakeholders. I have outlined some examples below using our test cluster.

CPU core affinity

Take a look at this simple script named **mploadtest.csh**. The script explicitly forks several CPU-bound tasks. In reality many users submit "single-slot" jobs that may behave in a similar manner or call API functions that spawn multiple child threads or processes:

```
-----start of mploadtest.csh -----------

#!/bin/csh
dd if=/dev/urandom bs=1M count=80 | bzip2 -9 >> /dev/null &
dd if=/dev/urandom bs=1M count=80 | bzip2 -9 >> /dev/null &
dd if=/dev/urandom bs=1M count=80 | bzip2 -9 >> /dev/null &
dd if=/dev/urandom bs=1M count=100 | bzip2 -9 >> /dev/null

-----end of mploadtest.csh -----------
```

Run the script as a single slot job on a specific idle host, for example: "**bsub -q mpitest -m bullet0019 ./mploadtest.csh" .** Open up a terminal session on the chosen host and observe the load across the CPU cores. You can do this running "top" in interactive mode and pressing "1" for the per-core load view. You'll notice the child processes associated with your running job are distributed across several cores, even though the job is "single-slot".

Wait for the job to finish then resubmit to a host that has cgroups enabled. This time also we request CPU affinity in the job submission command: **"bsub -q mpitest -m bullet0019 -R 'affinity[core]' ./mploadtest.csh".** Observe this job again using the per-core load view with top. This time you should see all of the load is associated with a single core. The number of assigned cores will match the number of job slots so submitting the job with "-n 3" will result in the job using 3 CPU cores.

Memory limit enforcement

This C program named **memeater** consumes some physical memory in a fairly short time period. We basically allocate and reference 100MB every 2 seconds for 10 iterations = 1GB:

```
------ start of memeater.c -------------

#include <stdio.h>
#include <stdlib.h>

/* allocate memory in 100MB chunks */

 #define HUNDRED_MB 104857600



main(){

    int x;
    for( x = 1; x <= 10 ; x++){
        char *myptr;
        myptr = malloc(HUNDRED_MB );
        int y;
        /* Dereference memory pointer so we use physical RAM */
        for(y = 0; y < HUNDRED_MB; y++){
            myptr[y] = 0;
        }
        printf("%d MBs allocated\n", x * HUNDRED_MB / 1048576);
        sleep(2);
    }
}

------ end of memeater.c -------------
```

Compile the executable and submit as a job for a specific host that does not have cgroups enabled. We will also specify a physical memory limit of 350MB: **"bsub -q systems -m pinto21 -M 350 ./memoryeater".** Open a terminal session on the target batch host and run "top -a -u <your_uid>" to monitor processes listed in order of memory usage. While the job is running you should see the amount of physical resident memory it is using ("RES") increase. The job will run beyond the requested 350MB limit and probably complete and exit normally. This is because LSF is simply checking memory use periodically and jobs may exceed memory limits between these check intervals. Next, try submitting the same job with the 350MB limit but specify a host with cgroups enabled: **"bsub -q systems -m bullet0020 -M 350 ./memoryeater".** Observe the memory use on your chosen host as we did before using "t op -a -u <your_uid>". This time you'll see that LSF kills the job as soon as the memory limit is exceeded.