Using numa=fake and CPUSets for Resource Management
Written by David Rientjes <rientjes@cs.washington.edu>

This document describes how the numa=fake x86_64 command-line option can be used
in conjunction with cpusets for coarse memory management.  Using this feature,
you can create fake NUMA nodes that represent contiguous chunks of memory and
assign them to cpusets and their attached tasks.  This is a way of limiting the
amount of system memory that are available to a certain class of tasks.

For more information on the features of cpusets, see
Documentation/cgroups/cpusets.txt.
There are a number of different configurations you can use for your needs.  For
more information on the numa=fake command line option and its various ways of
configuring fake nodes, see Documentation/x86/x86_64/boot-options.txt.

For the purposes of this introduction, we'll assume a very primitive NUMA
emulation setup of "numa=fake=4*512,".  This will split our system memory into
four equal chunks of 512M each that we can now use to assign to cpusets.  As
you become more familiar with using this combination for resource control,
you'll determine a better setup to minimize the number of nodes you have to deal
with.

A machine may be split as follows with "numa=fake=4*512," as reported by dmesg:

        Faking node 0 at 0000000000000000-0000000020000000 (512MB)
        Faking node 1 at 0000000020000000-0000000040000000 (512MB)
        Faking node 2 at 0000000040000000-0000000060000000 (512MB)
        Faking node 3 at 0000000060000000-0000000080000000 (512MB)
        ...
        On node 0 totalpages: 130975
        On node 1 totalpages: 131072
        On node 2 totalpages: 131072
        On node 3 totalpages: 131072

Now following the instructions for mounting the cpusets filesystem from
Documentation/cgroups/cpusets.txt, you can assign fake nodes (i.e. contiguous
memory
address spaces) to individual cpusets:

        [root@xroads /]# mkdir exampleset
        [root@xroads /]# mount -t cpuset none exampleset
        [root@xroads /]# mkdir exampleset/ddset
        [root@xroads /]# cd exampleset/ddset
        [root@xroads /exampleset/ddset]# echo 0-1 > cpus
        [root@xroads /exampleset/ddset]# echo 0-1 > mems

Now this cpuset, 'ddset', will only allowed access to fake nodes 0 and 1 for
memory allocations (1G).

You can now assign tasks to these cpusets to limit the memory resources
available to them according to the fake nodes assigned as mems:

        [root@xroads /exampleset/ddset]# echo $$ > tasks
        [root@xroads /exampleset/ddset]# dd if=/dev/zero of=tmp bs=1024 count=1G
        [1] 13425

Notice the difference between the system memory usage as reported by
/proc/meminfo between the restricted cpuset case above and the unrestricted
case (i.e. running the same 'dd' command without assigning it to a fake NUMA
cpuset):

|  | Unrestricted | Restricted |
|---|---|---|
| MemTotal: | 3091900 kB | 3091900 kB |
| MemFree: | 42113 kB | 1513236 kB |

This allows for coarse memory management for the tasks you assign to particular
cpusets.  Since cpusets can form a hierarchy, you can create some pretty
interesting combinations of use-cases for various classes of tasks for your
memory management needs.