TooManyOpenFiles

Too Many Open Files

You can see this on Linux machines in client-side applications, server code or even in test runs.

It is caused by per-process limits on the number of files that a single user/process can have open, which was introduced in the 2.6.27 kernel. The default value, 128, was chosen because "that should be enough".

In Hadoop, it isn't. To fix this log in/su/ssh as root and edit `/etc/sysctl.conf`

add the line

```
fs.epoll.max_user_instances = 2048
```

Then reboot. Different numbers may be chosen.

There is an immediate shortcut: `echo 2048 > /proc/sys/fs/epoll/max_user_instances` . This setting will be lost on the next reboot, but is handy for trying out different values.

limits.conf

Another limit on the number of files open may be the file `/etc/security/limits.conf`

It has a setting on the number of files a user or group may have, `nofile`.

To set this, as root edit `/etc/security/limits.conf`

and add a line such as

```
*               soft    nofile            2048
```

Then restart.

To see the current/default limits, run the command `ulimit -a`. This should print something like

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 20
file size               (blocks, -f) unlimited
pending signals                 (-i) 16382
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 4096
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) unlimited
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

You can dynamically up the limits until the next reboot with the same command. Specifically

```
ulimit -n 8192
```

The updated value can then be printed

```
# ulimit -n
8192
```