Introduction to Benchmarking

Rocks-A-Palooza II

Lab Session

SDSC

CPU Benchmark

Linpack

- Linpack is an MPI application that reports the sustained floating-point operations per second on a machine
- It performs an LU-decomposition
  - CPU-intensive
  - But also exercises the network

Linpack

- Linpack is used to sort the machines on the Top500 list
  - www.top500.org

1-100 | 101-200 | 201-300 | 301-400 | 401-500

Rank	Site	Computer	Processors	Year	R_{max}	R_{peak}
1	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution IBM	131072	2005	280600	367000
2	IBM Thomas J. Watson Research Center United States	BGW - eServer Blue Gene Solution IBM	40960	2005	91290	114688
3	DOE/NNSA/LLNL United States	ASC Purple - eServer pSeries p5 575 1.9 GHz IBM	10240	2005	63390	77824
4	NASA/Ames Research Center/NAS United States	Columbia - SGI Altix 1.5 GHz, Voltaire Infiniband SGI	10160	2004	51870	60960
5	Sandia National Laboratories United States	Thunderbird - PowerEdge 1850, 3.6 GHz, Infiniband Dell	8000	2005	38270	64512
6	Sandia National Laboratories United States	Red Storm Cray XT3, 2.0 GHz Cray Inc.	10880	2005	36190	43520
7	The Earth Simulator Center Japan	Earth-Simulator NEC	5120	2002	35860	40960

Running Linpack

- Linpack is part of the HPC Roll
  - The executable is named ‘xhpl’
- Login to your frontend as a non-root user
  - To create a user account, execute:
    - # useradd <username>

Running Linpack

- Get the linpack configuration file:

```
$ cp /var/www/html/rocks-documentation/4.1/examples/HPL.dat .
```

- Then, create a file named ‘machines’
  - This file should contain:

```
compute-0-0
compute-0-0
```

  - This tells xhpl that it should launch two process, one on compute-0-0 and another on compute-0-0

Running Linpack

- Launch the job

```
$ ssh-agent $SHELL
$ ssh-add
$ /opt/mpich/gnu/bin/mpirun -nolocal -np 2 -machinefile machines /opt/hpl/gnu/bin/xhpl
```

- This tells ‘mpirun’:
  - Don’t start a job on the frontend (-nolocal)
  - Use two processors (-np 2)
  - The names of those two processors are in the file ‘machines’
  - Start the program ‘xhpl’ on both processors

Linpack Results

- View the results

```
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
W11R2L8         1000    64     1     2               0.30           2.261e+00
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.9803216 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0237937 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0057484 ..... PASSED
================================================================================
```

- In this example, sustained 2.2 gigaflops
  - If we add 280,598 more, we'd be #1 on the Top500 list!

Linpack Results

- This configuration of linpack runs so fast, can't really view the results on the 'Cluster Status' page
  - So, let's scale linpack up!
- We'll increase size of matrix
  - Open file 'HPL.dat
  - Change line:
    - 1000 Ns
  - To:
    - 6000 Ns
- Relaunch linpack

Linpack Results

- You may see this error message:

```
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
W11R2L8         6000    64     1     2              52.12           2.764e+00
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0115600 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0241085 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0047482 ..... PASSED
p1_24309: (55.624154) xx_shmalloc: returning NULL; requested 3072560 bytes
p1_24309: (55.624323) p4_shmalloc returning NULL; request = 3072560 bytes
You can increase the amount of memory by setting the environment variable
P4_GLOBMEMSIZE (in bytes); the current size is 4194304
p1_24309:  p4_error: alloc_p4_msg failed: 0
p1_24309: (57.626802) net_send: could not write to fd=5, errno = 32
```

- This error message is common to MPI programs that have large memory footprint
- To fix, edit file '.bashrc' and append the line:
  - Export P4_GLOBMEMSIZE=200000000

Cleanup when an MPI Program Crashes

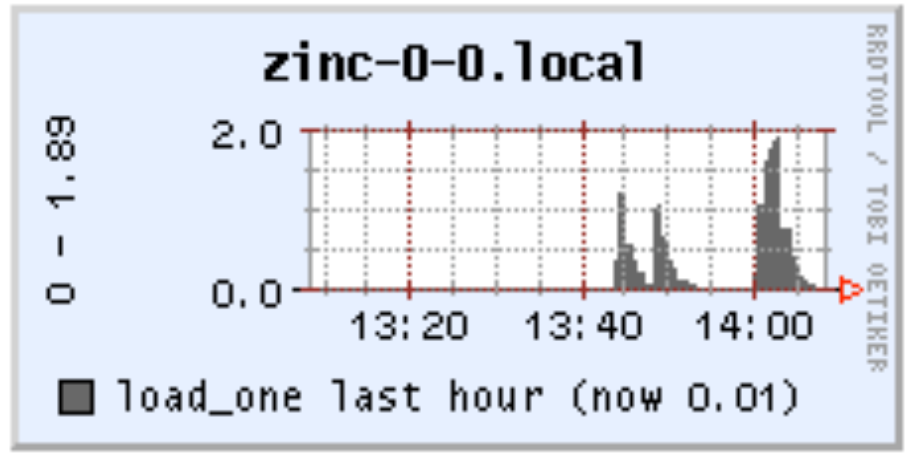
- MPICH in Rocks uses shared memory segments to pass messages between processes on the same node
- When an MPICH program crashes, it doesn't properly cleanup these shared memory segments
- After a program crash, run:

```
$ cluster-fork sh /opt/mpich/gnu/sbin/cleanipcs
```

- NOTE: Be aware that this removes all shared memory segments for your user id
  - If you have other live MPI programs running, this will remove their shared memory segments too and cause that program to fail

Monitoring a Job

- Point web browser to:
  - http://localhost
- Click ‘Cluster Status’ tab
- Specific node CPU stats look like:



Using more CPUs

- To use more CPUs, edit ‘HPL.dat’ and go to the section:

  1 Ps
  2 Qs

- Linpack uses P x Q processors
  - In the above example, 1 x 2 = 2 processors
- To use 4 processors
  - Change “1 Ps” to “2 Ps”, or
  - Change “2 Qs” to “4 Qs”

- Remember to also add entries to your ‘machines’ file!

Using more CPUs

- Relaunch for 4 CPUs:

$ /opt/mpich/gnu/bin/mpirun -nolocal -np 4 -machinefile machines /opt/hpl/gnu/bin/xhpl

Disk Benchmark

Iozone

- File system benchmark tool
  - http://www.iozone.org/
- Distributed with Rocks

Iozone

- Lots of flags
  - For example:
    - Can make Excel spreadsheets
    - Parallel I/O
    - Asynchronous system calls
    - And many, many more
- We'll walk through only a few

Iozone

- '-a'
  - Auto Mode
  - Runs all tests with increasing buffer size
  - Sample output

```
Auto Mode
        Command line used: /opt/iozone/bin/iozone -a
        Output is in Kbytes/sec
        Time Resolution = 0.000001 seconds.
        Processor cache size set to 1024 Kbytes.
        Processor cache line size set to 32 bytes.
        File stride size set to 17 * record size.
                                                    random  random    bkwd  record  stride
              KB  reclen   write rewrite    read    reread    read   write    read rewrite    read   fwrite frewrite   fread  freread
              64       4  238761  639793 1051057 1208283  841626  614716  940908  621006 1104577   283187   533423  983515 1122603
              64       8  304728  726843 1455473 1491271 1254099  752203 1277037  703783 1491000   328129   718441 1364053 1424647
              64      16  294886  736136 1425825 1488521 1361539  761455 1360805  571711 1254222   385577   832346 1390223 1424315
```

All Tests, One Buffer Size

- Buffer size (‘-s #’)
  - Where ‘#’ can be ‘1m’ for 1 MB file size

Iozone - All Tests, One Buffer Size

```
$  /opt/iozone/bin/iozone -s 1m
        Iozone: Performance Test of File I/O
                Version $Revision: 3.233 $
                Compiled for 32 bit mode.
                Build: linux

        Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
                     Al Slater, Scott Rhine, Mike Wisner, Ken Goss
                     Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
                     Randy Dunlap, Mark Montague, Dan Million,
                     Jean-Marc Zucconi, Jeff Blomberg,
                     Erik Habbinga, Kris Strecker.

        Run began: Tue May  9 15:37:15 2006

        File size set to 1024 KB
        Command line used: /opt/iozone/bin/iozone -s 1m
        Output is in Kbytes/sec
        Time Resolution = 0.000001 seconds.
        Processor cache size set to 1024 Kbytes.
        Processor cache line size set to 32 bytes.
        File stride size set to 17 * record size.
                                                            random  random    bkwd  record  stride
              KB  reclen   write rewrite    read    reread    read   write    read rewrite    read   fwrite frewrite   fread  freread
            1024       4  240146  501003   735634   819185  818477  527264  746861  672816  744710   241619   468667  689533   786499
```

Iozone - Write/Read Tests, One Buffer Size

- ‘-i 0 -i 1’
  - Run test 0 (write) and test 1 (read)
  - Must always run write test
    - It lays down a file in which to perform other operations upon
- Available tests:

-i # Test to run (0=write/rewrite, 1=read/re-read, 2=random-read/write
3=Read-backwards, 4=Re-write-record, 5=stride-read, 6=fwrite/re-fwrite
7=fread/Re-fread, 8=random_mix, 9=pwrite/Re-pwrite, 10=pread/Re-pread
11=pwritev/Re-pwritev, 12=preadv/Re-preadv)

Iozone - Write and Read Tests, One Buffer Size

```
$  /opt/iozone/bin/iozone -i 0 -i 1 -s 1m
        Iozone: Performance Test of File I/O
                Version $Revision: 3.233 $
                Compiled for 32 bit mode.
                Build: linux

        Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
                     Al Slater, Scott Rhine, Mike Wisner, Ken Goss
                     Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
                     Randy Dunlap, Mark Montague, Dan Million,
                     Jean-Marc Zucconi, Jeff Blomberg,
                     Erik Habbinga, Kris Strecker.

        Run began: Tue May  9 16:20:06 2006

        File size set to 1024 KB
        Command line used: /opt/iozone/bin/iozone -i 0 -i 1 -s 1m
        Output is in Kbytes/sec
        Time Resolution = 0.000001 seconds.
        Processor cache size set to 1024 Kbytes.
        Processor cache line size set to 32 bytes.
        File stride size set to 17 * record size.
                                                            random  random    bkwd  record  stride
              KB  reclen   write rewrite    read    reread    read   write    read rewrite    read   fwrite frewrite
            1024       4  247521  517713   744171   906975
```

Iozone - Write and Random Read Tests, One Buffer Size

- Write test (‘-i 0’)
- Random read test (‘-i 2’)
- Buffer size 1 MB (‘-s 1m’)

Iozone - Write and Random Read Tests, One Buffer Size

```
$  /opt/iozone/bin/iozone -i 0 -i 2 -s 1m
        Iozone: Performance Test of File I/O
                Version $Revision: 3.233 $
                Compiled for 32 bit mode.
                Build: linux

        Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
                     Al Slater, Scott Rhine, Mike Wisner, Ken Goss
                     Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
                     Randy Dunlap, Mark Montague, Dan Million,
                     Jean-Marc Zucconi, Jeff Blomberg,
                     Erik Habbinga, Kris Strecker.

        Run began: Tue May  9 16:21:35 2006

        File size set to 1024 KB
        Command line used: /opt/iozone/bin/iozone -i 0 -i 2 -s 1m
        Output is in Kbytes/sec
        Time Resolution = 0.000001 seconds.
        Processor cache size set to 1024 Kbytes.
        Processor cache line size set to 32 bytes.
        File stride size set to 17 * record size.
                                                            random  random    bkwd  record  stride
              KB  reclen   write rewrite    read    reread    read   write    read rewrite    read   fwrite frewrite
            1024       4  234645  499747                    776990  563849
```

Use Iozone to Test NFS

- Login to compute node as non-root user
- Write/read to home directory
  - NFS mounted back to frontend

Wow, NFS is Fast!

```
$ /opt/iozone/bin/iozone -i 0 -i 2 -s 1m
        Iozone: Performance Test of File I/O
                Version $Revision: 3.233 $
                Compiled for 32 bit mode.
                Build: linux

        Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
                     Al Slater, Scott Rhine, Mike Wisner, Ken Goss
                     Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
                     Randy Dunlap, Mark Montague, Dan Million,
                     Jean-Marc Zucconi, Jeff Blomberg,
                     Erik Habbinga, Kris Strecker.

        Run began: Tue May  9 16:23:36 2006

        File size set to 1024 KB
        Command line used: /opt/iozone/bin/iozone -i 0 -i 2 -s 1m
        Output is in Kbytes/sec
        Time Resolution = 0.000001 seconds.
        Processor cache size set to 1024 Kbytes.
        Processor cache line size set to 32 bytes.
        File stride size set to 17 * record size.
                                                            random  random    bkwd  record  stride
              KB  reclen   write rewrite    read    reread    read   write    read rewrite    read
    fwrite frewrite   fread  freread
            1024       4  511258  564486                   704804  611009
```

Wow, NFS is Fast!

- Benefiting from caching effects
- Need to write/read a file that is larger than the memory size

```
$ cat /proc/meminfo | grep MemTotal
MemTotal:      2074480 kB
```

- Above machine has 2 GB
  - Need to write a file that is at least 2 GB

Realistic NFS Numbers

```
$ /opt/iozone/bin/iozone -i 0 -i 1 -s 4g
        Iozone: Performance Test of File I/O
                Version $Revision: 3.233 $
                Compiled for 32 bit mode.
                Build: linux

        Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
                     Al Slater, Scott Rhine, Mike Wisner, Ken Goss
                     Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
                     Randy Dunlap, Mark Montague, Dan Million,
                     Jean-Marc Zucconi, Jeff Blomberg,
                     Erik Habbinga, Kris Strecker.

        Run began: Tue May  9 17:06:40 2006

        File size set to 4194304 KB
        Command line used: /opt/iozone/bin/iozone -i 0 -i 1 -s 4g
        Output is in Kbytes/sec
        Time Resolution = 0.000001 seconds.
        Processor cache size set to 1024 Kbytes.
        Processor cache line size set to 32 bytes.
        File stride size set to 17 * record size.
                                                            random  random    bkwd  record  stride
              KB  reclen   write rewrite    read    reread    read   write    read rewrite    read
         4194304       4   50955   37263   20867    24051
```

Bonnie

- Another file system benchmark
- Not bundled in Rocks
  - See ‘Cluster Management and Maintenance Lab’ in order to deploy bonnie

Bonnie

- Execute bonnie

/share/apps/benchmarks/bonnie++/sbin/bonnie++ -s 4096 -n 0 -f -d ~/output_files

- Flags
  - '-s 4096' - write a 4 GB file
  - '-n 0' - skip the 'file creation' test
  - '-f' - fast mode, don't do character (one byte) tests
  - '-d ~/output_files' - put all temporary files in ~/output_files
    - If ~/output_files is mounted on NFS, then this tests NFS

Bonnie Output

```
Writing intelligently...done
Rewriting...done
Reading intelligently...done
start 'em...done...done...done...
Version  1.03       ------Sequential Output------ --Sequential Input- --Random-
                    -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
rocks-45.sdsc.ed 4G           36597  15 17056   5           38552   6 156.6   0
rocks-45.sdsc.edu,4G,,,36597,15,17056,5,,,38552,6,156.6,0,,,,,,,,,,,,
```

- Measurements for sequential output/input
- Last line is comma-separated values
  - Can be used import values into analysis program

Network Benchmark

Using iperf

- On one node, run the ‘server’

```
# ssh compute-0-0
# /opt/iperf/bin/iperf -s
```

- On another, run the ‘client’

```
# /opt/iperf/bin/iperf -c compute-0-0
```

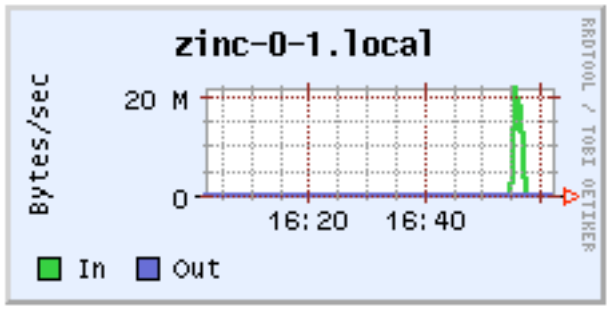
Iperf output

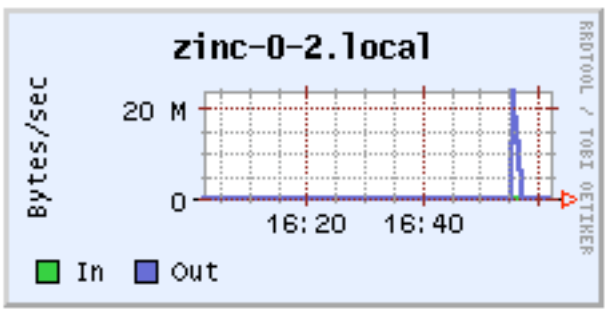
```
# /opt/iperf/bin/iperf -c zinc-0-1
------------------------------------------------------------
Client connecting to zinc-0-1, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.255.255.252 port 33570 connected with 10.255.255.253 port 5001
[  3]  0.0-10.0 sec  1.10 GBytes    941 Mbits/sec
```

View Iperf Network Traffic

- Change ‘Cluster Status report’ to ‘network_report’:

- Then look at ‘server’ and ‘client’





- Client sent data to server at a peak of 25 MB/s
- That doesn’t look right?!?!
  - Need to send more data
  - Ganglia’s sampling is too coarse for this small run

Scaling iperf Up

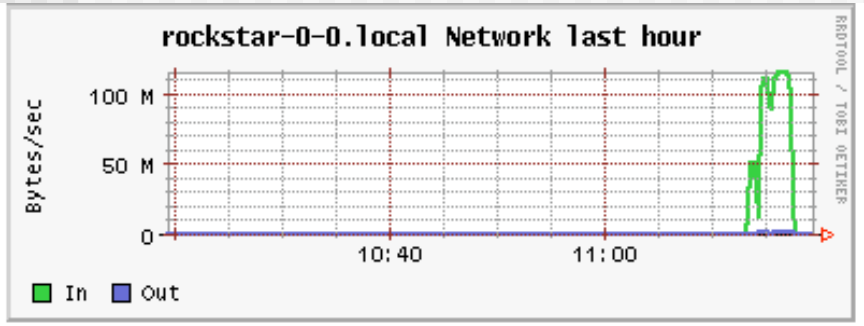
- On one node, run the ‘server’

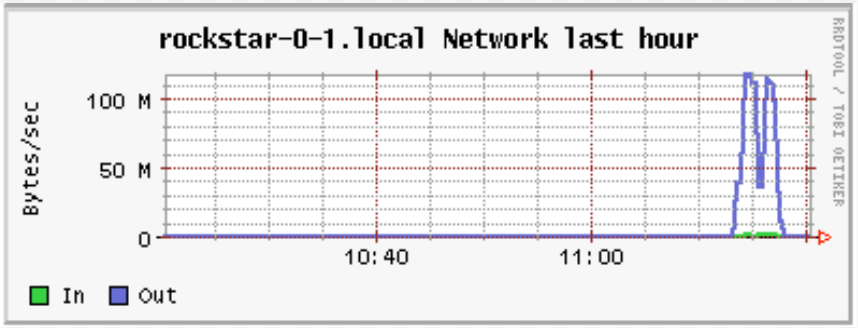
```
# ssh compute-0-0
# /opt/iperf/bin/iperf -s
```

- On another, run the ‘client’, send for 120 seconds and display results every 5 seconds

```
# /opt/iperf/bin/iperf -c compute-0-0 t 120 -i 5
```

‘Better’ Looking Graphs





- Peak of 125 MB/s