Channel Bonding Notes

Luke Hindman
Department of Computer Science
College Of Engineering
Boise State University, Boise, Idaho 83725

This material is based upon work supported by the National Science Foundation under grant No. 0321233, Development of Tools to Enable the Port of Software to a Beowulf Cluster.

Installation and Configuration Details:

I initially began by attempting to use the Linux kernel bonding driver (bonding.o) to implement channel bonding on the master node. However, each time I would bring up the bonded interface (bond0) I would loose communication with the remainder of the cluster. After a lot of searching, I came across a post in which someone was experiencing the same problem as myself. They attributed it to a conflict between the Broadcom Gbit drivers and the bonding driver.

Not to be dissuaded, I began looking for other options when I discovered that Broadcom provided channel bonding support of their own in a package called Broadcom Advanced Server Program (BASP). This package provides many features including a channel bonding solution they call teaming.

BASP Install:

(1) download BASP and unzipped to /home/lhindman/install
(2) tar –xzf basplnx-6.1.2.i386.tgz
(3) cd basplnx-6.1.2
(4) make // this compiles basp.o
(5) su – root
(6) make install // creates device file and copies files
(7) depmod –a
(8) insmod basp // generates warning about proprietary code

BASP Configuration:

(1) cd /etc/sysconfig/network-scripts
(2) edit ifcfg-eth1 as follows, remove all additional entries
    a. DEVICE=eth1
    b. ONBOOT=yes
    c. BOOTPROTO=static
(3) Repeat step 2 for eth2 and eth3
(4) cp /etc/basp/samples/team-gec /etc/basp
(5) edit /etc/basp/team-gec as follows
    a. First Adapter: eth1
    b. Second Adapter: eth2
    c. Third Adapter: eth3
    d. Virtual Interface: bond0
        i.ip: 192.168.1.1

ii.mask: 255.255.255.0

(6) Manually starting BASP:
   a. /etc/init.d/basp start
(7) Automatically starting BASP at system startup:
   a. (1) chkconfig --add basp // adds basp to runlevels 2,3,4,5

Switch Configuration:
I configured port 1 on each switch for etherchannel.

(1) Log into switch web interface (http://192.168.1.254)
(2) Click ports and select EtherChannel
(3) Click create
    a. group name: 1
    b. Select port 1 on each switch (Gi 1/0/1, Gi 2/0/1, Gi 3/0/1)
    c. Set mode for each port to on
(4) Save running config to startup config

Additional Installation Notes:
- Created backup copy of original ifcfg-eth1 called bkp.ifcfg-eth1
- Install Webmin to access firewall rules
    - /usr/local/webmin
    - Did NOT configure to start at boot
    - User: Admin
    - Port: 10000
- Added rule to IPTables to accept all traffic from bond0
- Added rule to IPTables to reject tcp traffic to port 10000 from eth0

Steps to disable Channel Bonding:
(1) Log into switch and delete EtherChannel group
    a. From web console click ports and select EtherChannel
    b. Select the group and click delete
    c. Save Running Config to Startup Config
    d. NOTE: This might need to be done from a separate machine because the master node will loose network access when EtherChannel is disabled.
(2) On master node, stop basp
    a. /etc/init.d/bast stop
(3) Remove basp from startup folders (rc2.d, rc3.d, rc4.d, rc5.d)
(4) Restore backup copy of ifcfg-eth1
(5) Edit ifcfg-eth2, ifcfg-eth3 as follows
    a. ONBOOT=no
(6) Restart network
    a. /etc/init.d/network restart

Benchmarking Details:

In order to benchmark the performance of the channel bonding, I performed the following tests using the NetPipe utility. Since NetPipe works in a point to point manner, running a single instance of NetPipe would not be an accurate measure of the channel bonded connection since the bottle neck would be the machine on the other end of the connection. It get around this issue, I ran multiple simultaneous instances of NetPipe (thanks to Kevin's script) to multiple nodes within the cluster.

Each of the following tests were performed before (base) and after (bond) the channel bonding was installed.

Single Node (test 1):
- Test two-way communication between master node and node 01
- NPtcp –h node01 –b 131072

Multi Node (test 2):
- Test two-way communication between master node and nodes 01 and 02
- NPtcp –h node1,2 –b 131072

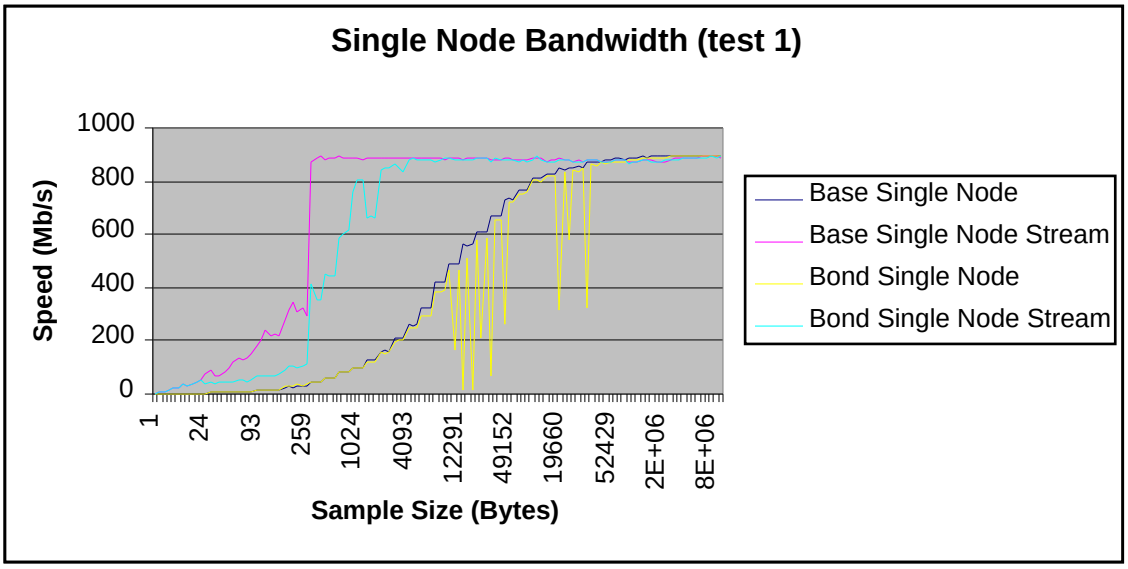
Multi Node (test 3):
- Test two-way communication between master node and nodes 01,02, and 03
- NPtcp –h node1,2,3 –b 131072

Single Node Stream (test 1):
- Test one-way communication between master node and node 01
- NPtcp –h node01 –b 131072 -s

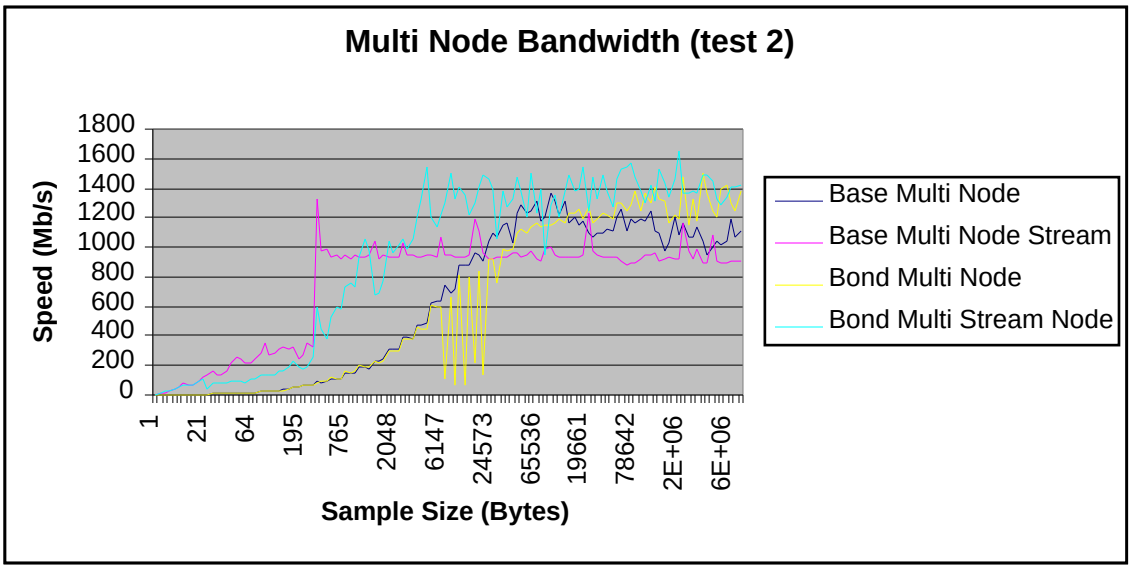
Multi Node Stream (test 2):
- Test two-way communication between master node and nodes 01 and 02
- NPtcp –h node1,2 –b 131072 -s

Multi Node Stream (test 3):
- Test two-way communication between master node and nodes 01,02, and 03
- NPtcp –h node1,2,3 –b 131072 –s

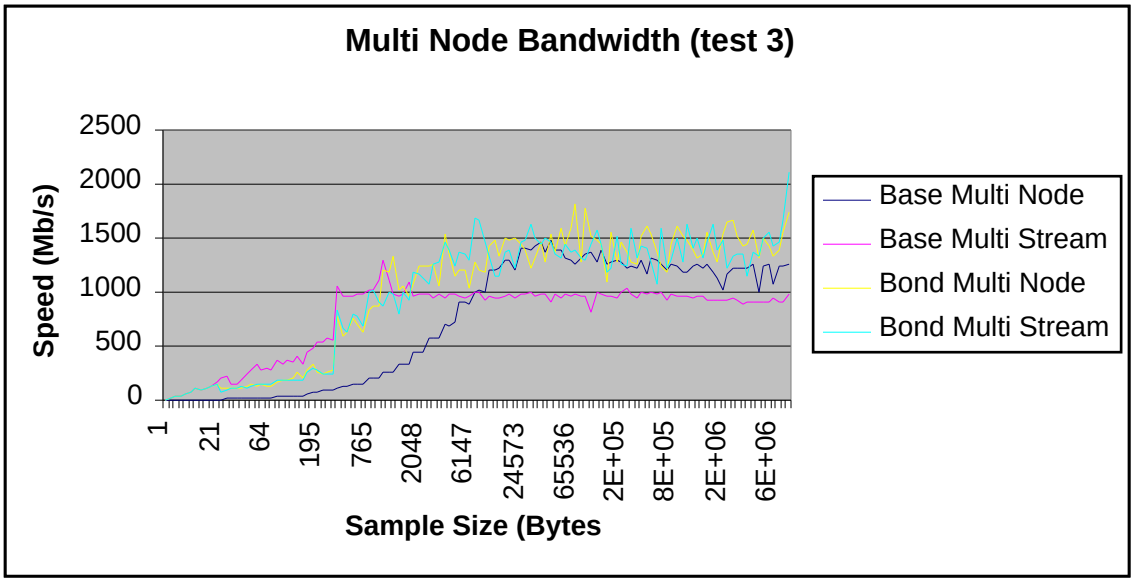
Additional Testing Notes:
- To calculate the total bandwidth for the multimode tests, I added together the bandwidth values from corresponding sample sizes then graphed the results: Since the tests did not run in perfect synchronization, I do not believe that the bandwidth values for an individual sample rate are entirely accurate. However I do believe that the general trends of the graphs reflect actual performance increase.
- To calculate the latency for the multi node tests, I calculated the average latency of all the tested nodes. I did not graph the latency information from the Stream tests because according to the NetPipe docs, this value is not accurate.



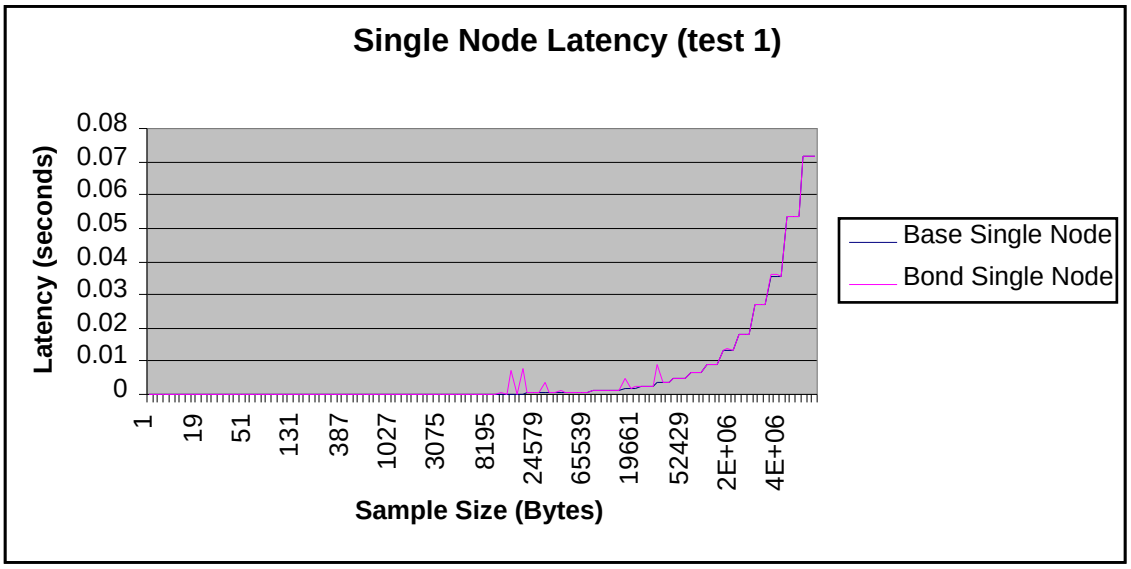
Evaluation: In this case, it appears that the channel bonding performs worse than the base tests. Why does the 'Bond Single Node' test have so many low values in the 12291 – 49152 range?



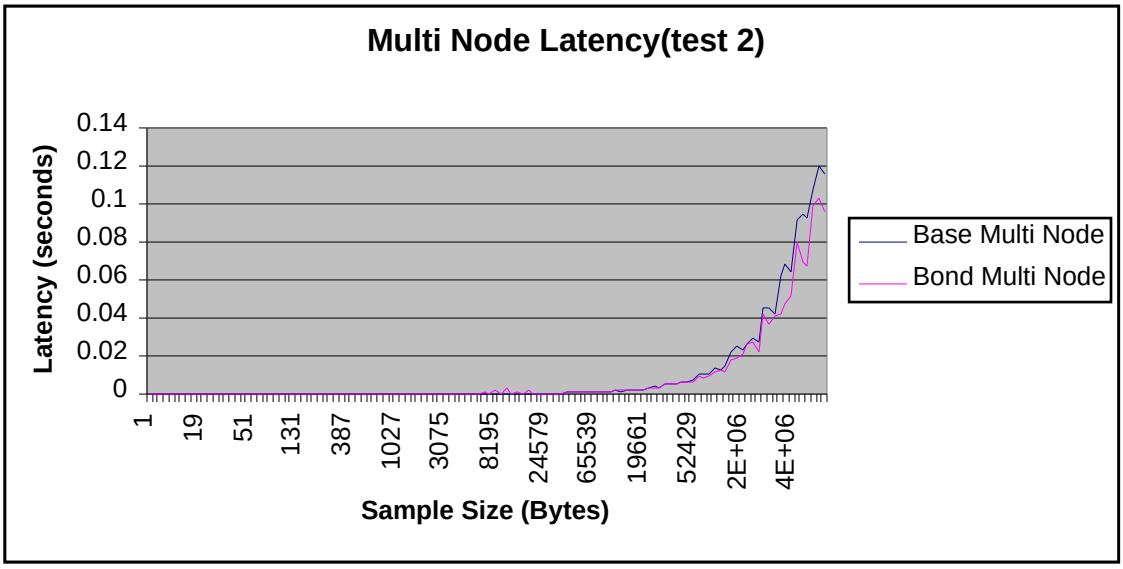
Evaluation: How is it possible that the 'Base Multi Node' test performed above 1000 Mb/s?



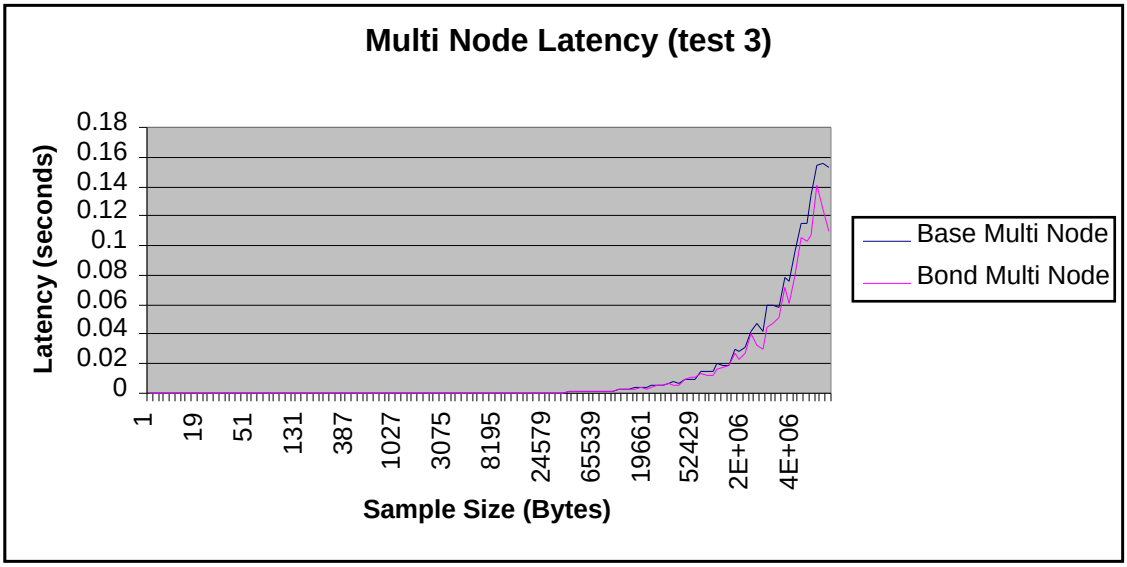
Evaluation: Again, how is it that the 'Base Multi Node' test show a speed of over 1000 Mb/s? Since the one-way 'Base Multi Stream' is right at 1000 Mb/s perhaps it has something to do with the two-way communication.



Evaluation: In this test it appears the latency between the base and bond test are almost identical.



Evaluation: Here the Average Latency of the bond test is lower than the corresponding base latency.



Evaluation: While the average latency values overall are higher than test 2, it still appears that the bond test has a lower latency than the base test.

Conclusion:

To be honest, I expected much higher performance from the channel bonded tests. The limited performance shown above could simply caused by the way I ran my tests, or it may indicate a bottle neck somewhere else on the system. In either case, I feel that more testing would yield a better picture of what is actually going on.

Also, the BASP package has several configuration options which may help to improve the performance the bonded cards.