InfiniBand at Home

Matteo Gandolfi

ERLUG

Cosa è / Storia

- InfiniBand (IB) è uno standard di comunicazione ad alte prestazioni, principalmente usato nell'High Performance Computing, caratterizzato da una bassa latenza ed un alto troughput.
- Nasce nel 1999 dalla fusione di altri due standard in competizione.
- I primi adattatori 10Gb/s arrivano sul mercato nel 2001
- Nel 2004 OpenFabrics Alliance sviluppa un sotware stack per linux e nel 2005 viene rilasciata la prima versione di OFED (OpenFabrics Enterprise Distribution).

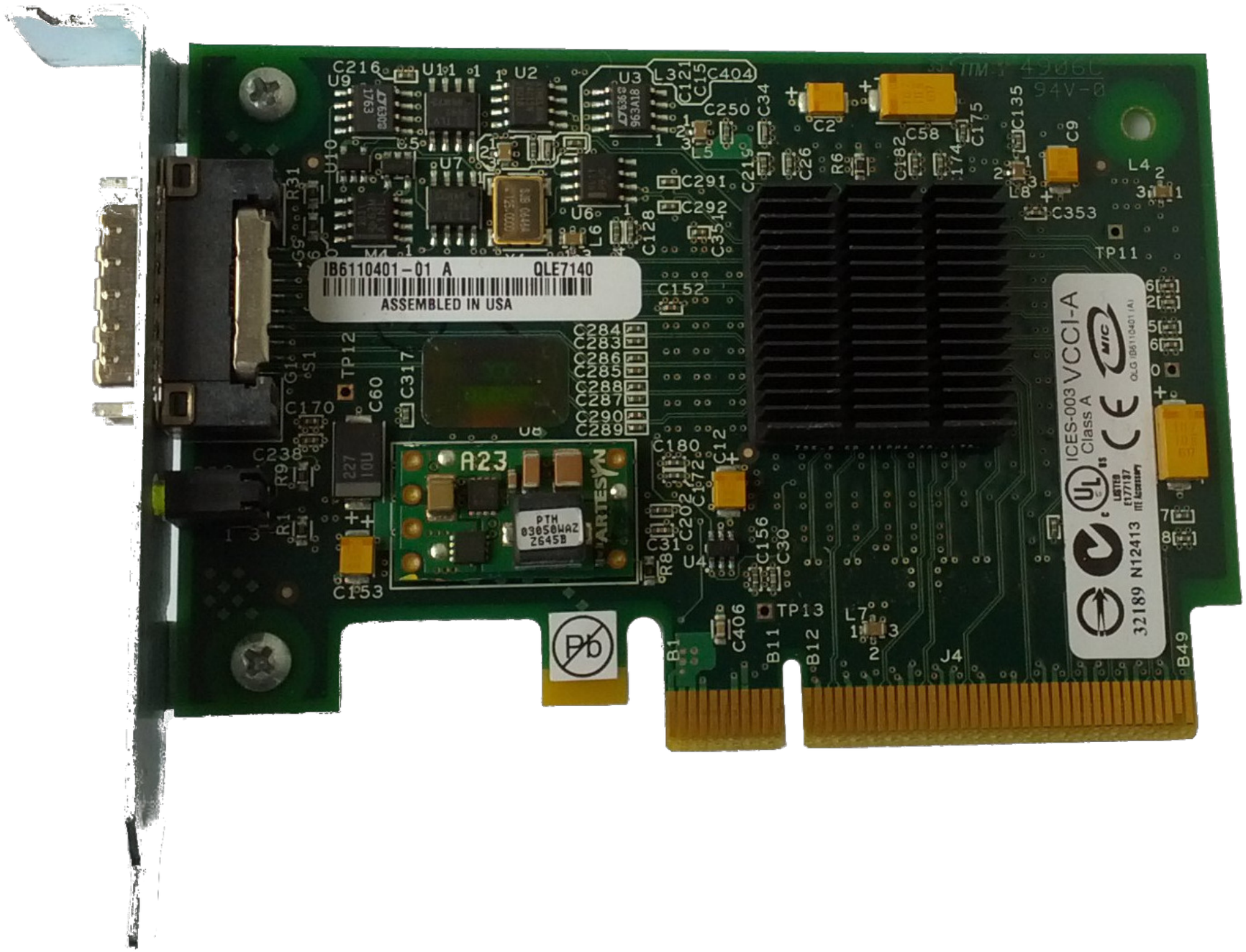
Prestazioni

- Ogni porta di un adattatore IB è costituita da 4 o 12 linee di comunicazione. Vengono espresse con 4x e 12x
- La velocità della porta è data dall'aggregazione delle linee:

\Data Rate:	**SDR (Single)**	**DDR (Double)**	**QDR (Quad)**	**FDR (Fourteen)**	**EDR (Enhanced)**
Gb/s teorico per canale	**2.5**	**5**	**10**	**14.0625**	**25**
Gb/s effettivo per canale	2	4	8	13.64	24.24
Gb/s teorico 4x	**10**	**20**	**40**	**56.25**	**100**
Gb/s effettivo 4x	8	16	32	54.54	96.97
Gb/s effettivo 12x	24	48	96	163.64	290.91
Encoding	8/10	8/10	8/10	64/66	64/66
Latenza µs	**5**	**2.5**	**1.3**	**0.7**	**0.5**
Anno	**2001**	**2005**	**2007**	**2011**	**2014**

Hardware

- Adattatore IB 10Gb:



Hardware - Cavi

- Gli adattatori IB “economici” hanno solitamente connettori SFF-8470 “Latched” cioè con i ganci e non con le viti
- Quando si scelgono i cavi bisogna anche fare attenzione che il “keying” (dentini di allineamento) corrisponda sia sul cavo che sulla scheda.

IPoIB

- IP over Infiniband, o IPoIB, è il protocollo che definisce l'incapsulamento e la trasmissione dei protocolli IPv4, IPv6 dell'ARP sull'InfiniBand.

Supporta due modalità:

- Datagram: usa l' IB UD (Unreliable Datagram), l'MTU è uguale all'MTU del L2 IB meno i 4 bytes dell'header di encapsulation dell'IPoIB, quindi in un fabric standard avente MTU 2k l'MTU dell'IPoIB sarà 2044 bytes.
- Connected: usa l'IB RC (Reliable Connected), l'MTU dell'IPoIB può essere portata fino a 64k (65520 bytes)

Sistema di test

- 2x Dell SC1435
  2x Opteron Dual-core 2218
  8GB ram (velocità 658 MB/s)
  1x IB DDR 20Gb Mellanox MHGA28-XTC [InfiniHost III Ex] dual port
  CentOS 6.7 (da minimal) e OFED 3.18
- 1x HP ML110 G6
  1x Xeon Quad-core X3430
  8GB ram (velocità 2.5 GB/s)
  1x IB DDR 20Gb Mellanox MHGA28-XTC [InfiniHost III Ex] dual port
  CentOS 6.7 (da DVD) e OFED 3.18
- 1x Switch SilverStorm 9024 24-Port Infiniband DDR Switch
- Varie ed eventuali per la parte elettrica e di rete ethernet

E un rack elegantissimo:

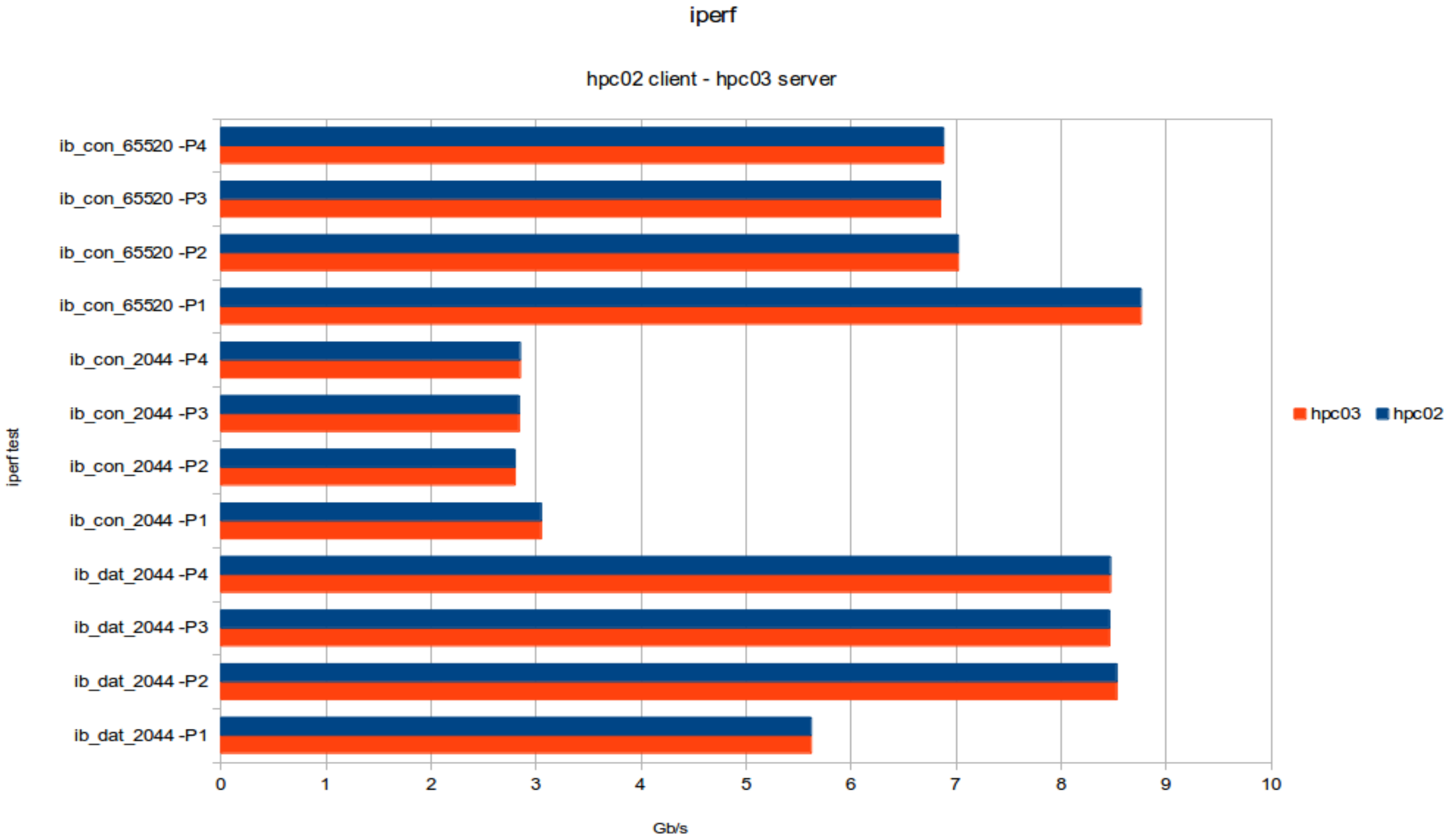
Installazione driver (su CentOS)

- I driver OFED sono scaricabili da https://www.openfabrics.org/downloads/OFED/
- Basta estrarli ed eseguire install.pl
- Esperienza personale con CentOS 6.7 e OFED 3.18:
  Procedere con l'installazione custom dei pacchetti selezionandoli manualmente e non installare:
  - compat-rdma
  - compat-rdma-devel
  - 32-bit packages
- Aggiungere ai moduli da caricare:
  ib_sa ib_cm ib_umad ib_addr ib_uverbs ib_ipoib ib_ipath ib_qib ib_mthca svcrdma
- Subnet Manager, si occupa di sviluppare una routing table per gli adattatori collegati al fabric IB, va fatto partire su un host.
  #service opensm start

Test con iperf

- I primi test sono stati fatti con iperf usando hpc03 come server e hpc02 come client.
- Sono state testate varie configurazioni delle interfacce IB:
  - Datagram MTU 2044 (ib_dat_2044)
  - Connected MTU 2044 (ib_con_2044)
  - Connected MTU 65520 (ib_con_65520)
- Con diverse impostazioni del client di iperf, da 1 a 4 test paralleli: -P 1-4

Risultati iperf



Osservazioni iperf

- Possiamo notare le diverse performance tra la modalità connected con mtu di default e connected con l'mtu massimizzata
- Possiamo anche notare come i test parallelizzati non sempre corrispondono ad un incremento prestazionale.
  Ho potuto osservare che il carico sulle cpu aumentava, in particolare in modalità connected, all'aumentare dei test paralleli, ritengo quello il motivo del calo di prestazioni.
  (non sono riuscito a riportare i dati di carico delle cpu per questo test)

RDMA

- “Remote Direct Memory Access” è la funzionalità di un device di accedere direttamente alla memoria dell'host senza andare a gravare sulla cpu.

Vantaggi:

- Zero-copy: le applicazioni possono trasferire i dati senza il software dello stack di rete.
- Bypass del kernel: le applicazioni possono trasferire i dati direttamente dallo userspace.
- Carico nullo sulla cpu: la memoria sulla macchina remota sarà letta senza alcun intervento della cpu.
- Message based transaction: i dati sono gestiti come messaggi discreti e non come un flusso eliminando la necessità dell'applicazione di separare il flusso in differenti messaggi/transazioni.

Test con mount points NFS

- Gli altri test che ho avuto modo di eseguire sono stati effettuati con 3 diversi mount point NFS impostati sullo stesso “client”, uno ethernet, uno in IB diretto ed uno in RDMA.
- La scelta è ricaduta sull'NFS poiché è possibile configurare il server NFS per accettare connessioni RDMA.
- Il test è stato eseguito scrivendo un file da 2 GB da /dev/zero sul mountpoint relativo al test.
- Anche in questo caso i test sono stati eseguti con le diverse combinazioni delle modalità datagram/connected e le diverse mtu.

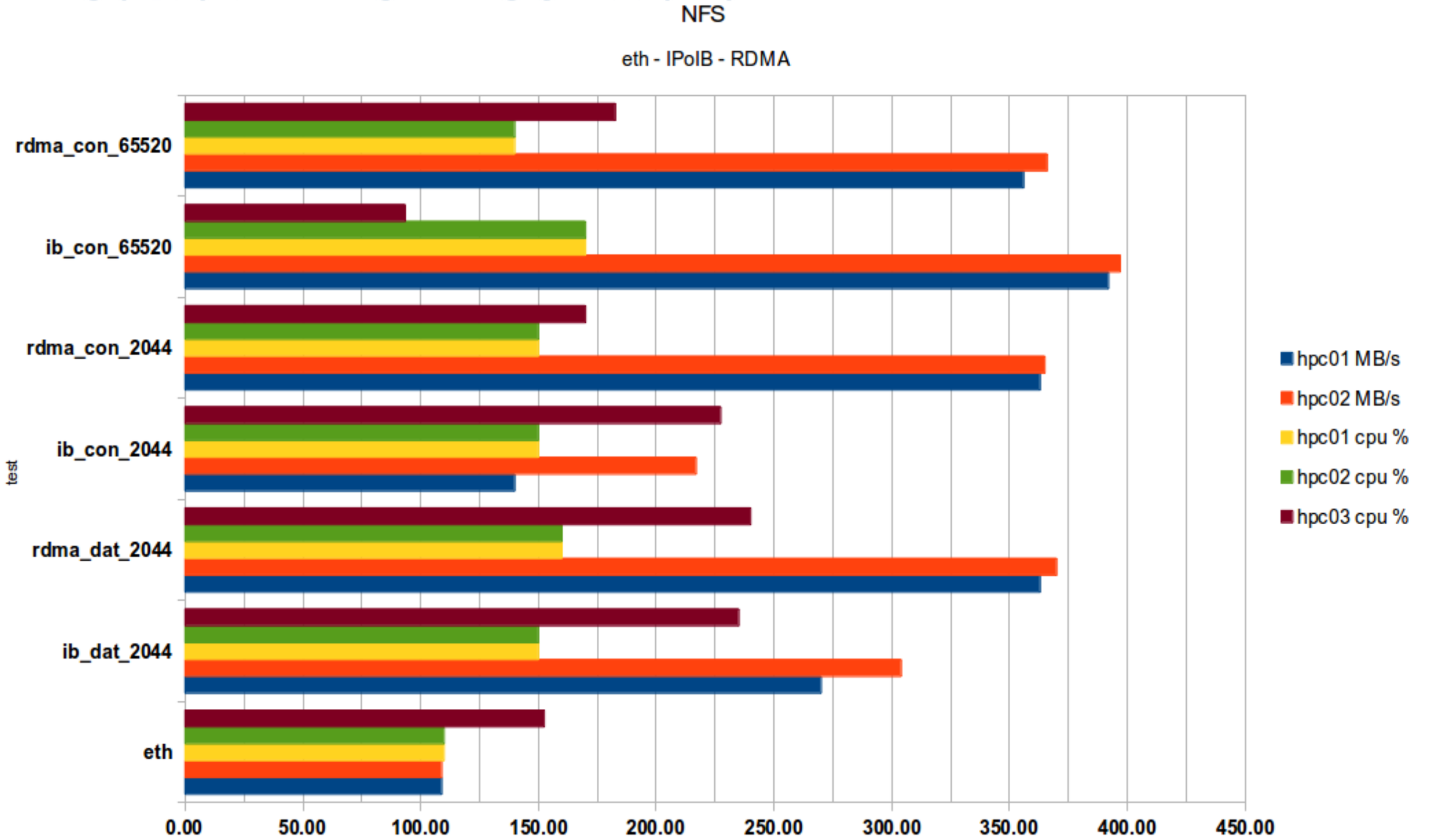
Ramdisk

- Non avendo sottosistemi dischi che andassero a velocità tali (anche in raid 0) da non essre un collo di bottiglia per i trasferimenti, i test sono stati effettuati su ramdisk.
- Un ramdisk è una porzione della memoria ram che viene utilizzata come se fosse un disco.

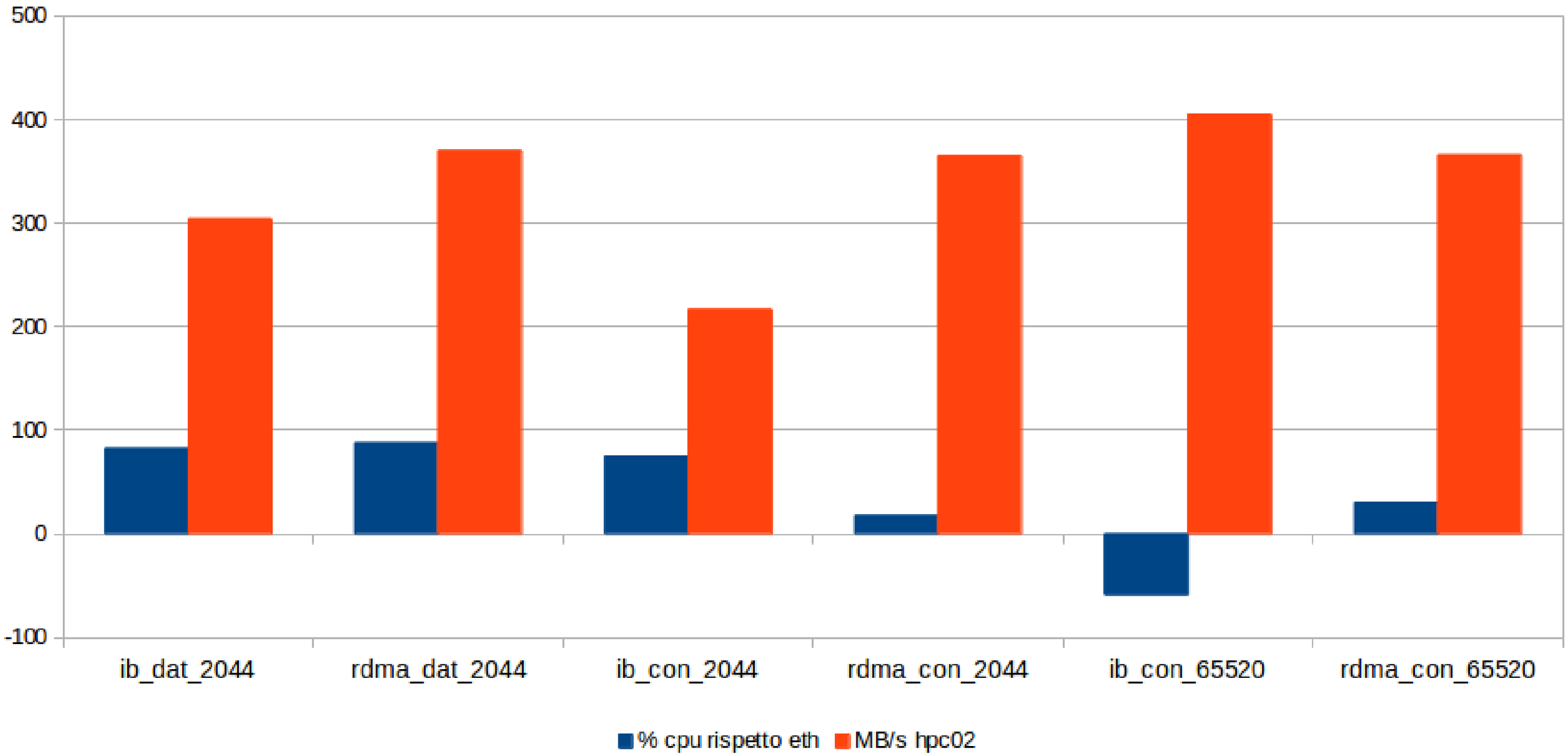
Su linux si può fare in due modalità:

- **RamFS**: cresce dinamicamente senza un limite imposto, quindi bisogna stare attenti a non “sforare” la ram fisica poiché si bloccherebbe il sistema
  #mount -t ramfs -o size=1g ramfs /storage/ramfs
- **TmpFS**: ha un limite imposto quindi non c'è pericolo di riempire accidentalmente la ram, al contrario di RamFS impatta sullo swap
  #mount -t tmpfs -o size=4g tmpfs /storage/tmpfs

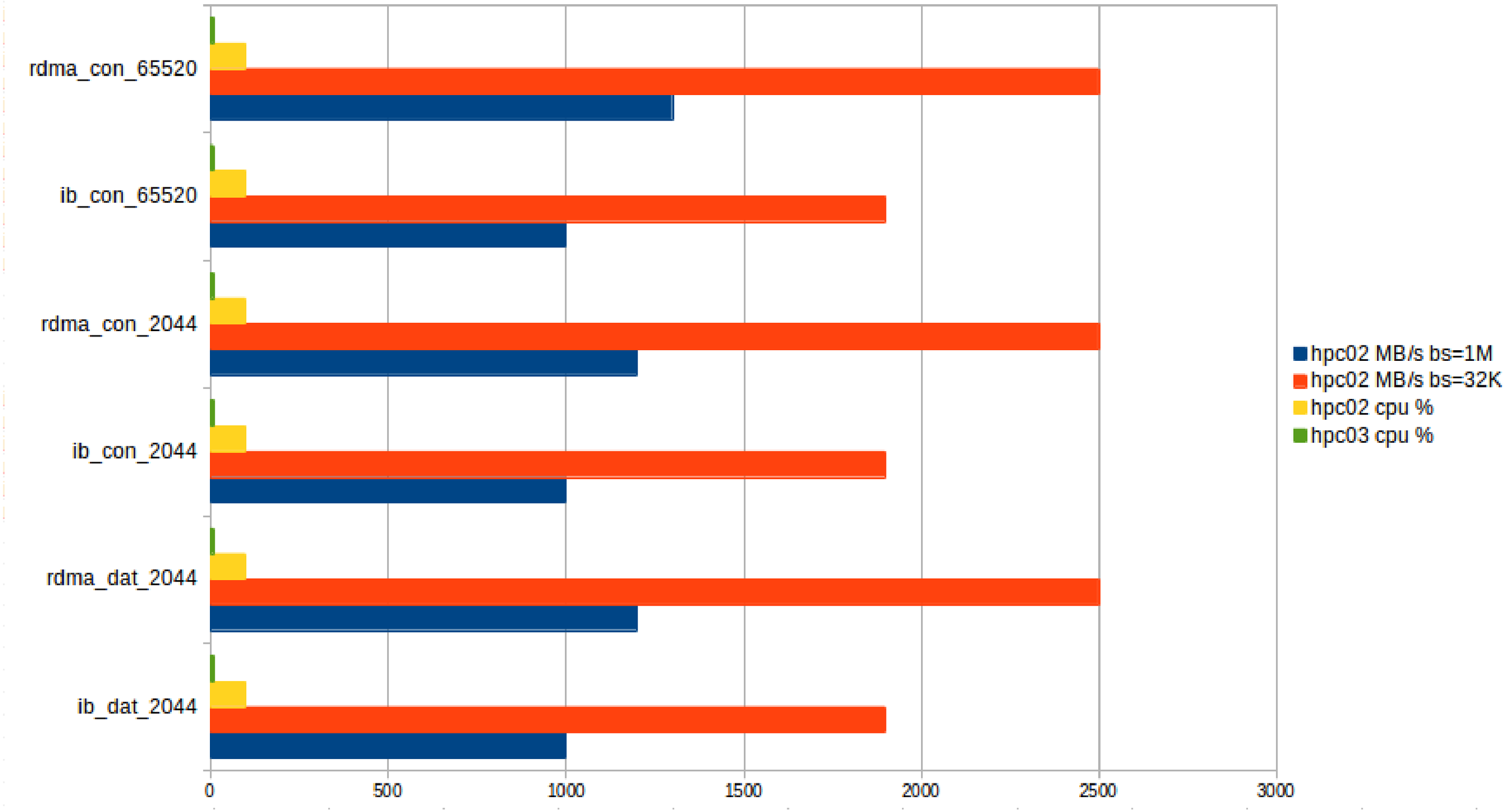
Risultati NFS in scrittura



Carico cpu su hpc03 rispetto all'ethernet (in scrittura)



Risultati NFS in lettura



Osservazioni NFS

- Il trasferimento su ethernet ci può dare dei valori base da considerare quando andiamo a osservare gli altri risultati.
- Il carico delle cpu è espresso in % su un massimo di 400%, ovvero 100% x 4 core.
- Al contrario di quanto mi aspettassi (con la premessa dell'RDMA), le migliori performance di trasferimento sono state ottenute in modalità connected, senza RDMA, con MTU massimizzata, arrivando a 405 MB/s ovvero 3.2 Gb/s
- I test sono stati fatti con dd con BS=1M, abbassando il blocksize (l'ottimale è a 32K) incrementano le prestazioni del trasferimento di circa un 20/25%. Il carico sulla cpu del server incrementa del 50%.

  Essendo che i test sono stati abbastanza empirici non ho ritenuto importante rifarli con il blocksize a 32K.

Bonus: interconnessione Ethernet-IPoIB

- Funziona senza problemi

Dubbi

- Non ho ancora capito quanto impatti l'hardware che sta sotto le schede IB, al momento immagino molto. Dai test che avevo fatto con i soli server Dell le performance (in modalità Connected MTU 65520) erano inferiori del 50% per l'IPoIB (405 vs 202 MB/s) e del 22% sull'RDMA (366 vs 288 MB/s). Vorrei riuscire a testare con hardware più recente e confrontare i dati.

Next steps

- Replicare device a blocchi tramite IB, con DRBD direttamente e con iSER (iSCSI Extensions for RDMA).
- Filesystem con supporto ad RDMA tipo GlusterFS
- Capire se ci sono altre best-practices/ottimizzazioni da applicare che mi sono sfuggite.

- Conquista del globo terraqueo, instaurazione governo distopico, grande massacro delle stampanti.

Test iperf

- Server: iperf -s
- Client: iperf -c ip_server -P 1-4

Test dd

- Server: iperf -s
- Client: iperf -c ip_server -P 1-4

Test su mountpoint NFS

- Scrittura: dd if=/dev/zero of=/mountpoint/test.img bs=1M count=2048
- Lettura: dd if=/mountpoint/test.img of=/dev/null bs=1M

Dipendenze OFED 2.18 su CentOS 6.7

- yum groupinstall 'Development Tools'
- yum install wget zlib-devel tcl tcl-devel tk tcl tcsh expat-devel imake libtool glibc-devel.i686 libnl-devel libnl glib2-devel libudev-devel pciutils

Aggiunta moduli su CentOS 6.7

- edit /etc/rc.modules:
  modprobe ib_sa
  modprobe ib_cm
  modprobe ib_umad
  modprobe ib_addr
  modprobe ib_uverbs
  modprobe ib_ipoib
  modprobe ib_ipath
  modprobe ib_qib
  modprobe ib_mthca
  modprobe svcrdma
- chmod +x /etc/rc.modules (il file verrà letto in avvio e i moduli caricati)

Fonti

- OFED Download: https://www.openfabrics.org/downloads/OFED/
- Open Fabrics Alliance: https://www.openfabrics.org
- RDMA: http://www.rdmamojo.com/2014/03/31/remote-direct-memory-access-rdma/
- NFS-RDMA: https://www.kernel.org/doc/Documentation/filesystems/nfs/nfs-rdma.txt
  NB: in CentOS mi è bastato editare: /etc/sysconfig/nfs scommentando “RDMA_PORT=20049”
  come indicato: https://lists.fedoraproject.org/pipermail/users/2014-November/455615.html
- http://www.davidhunt.ie/enabling-infiniband-on-ububtu-10-10/

GRAZIE!

Le slides e le riprese audio/video

dell'intervento saranno disponibili su:

http://erlug.linux.it/linuxday/2015/