# PVFS2 Experiments

## (1)POSIX interface pvfs2 installation

**configure pvfs2**

hec-42,hec-44 down

**kernel**

hec-40: different kernel I compiled before

**pvfs-server:** 8 nodes

hec-40

hec-41

hec-43

hec-45

hec-46

hec-47

hec-48

hec-49

**pvfs-client:** hec-41

Parallel file system: **OrangeFS** 2.8.6

HEC’s **kernel heads** path: /usr/src/linux-2.6.28.10

**To build PVFS2:**

./configure --with-kernel=/usr/src/linux-2.6.28.10 --prefix=/mnt/common/shuibing/install

make

make install

**To produce kernel module**

make kmod

/mnt/common/shuibing/software/orangefs-2.8.6/src/kernel/linux-2.6/pvfs2.ko

**copy to:**

/mnt/common/shuibing/research/iocon/confs/pvfs2.ko

**To produce pvfs2-server config file**

pvfs2-genconfig --quiet --protocol tcp --ioservers "hec-{40-41,43-43,45-49}" --metaservers "hec-{40-41,43-43,45-49}" --storage /home/shuibing/iocon-pvfs2 confs/8N.conf

**To prepare the storage space**

/home/shuibing/iocon-pvfs2

or script:

tests/scripts/mk-pvfs2-space.sh

#!/bin/bash

#prepare PVFS2 storage spaces

#when we use sudo, we use the absolute path of executable file

for i in 40 41 43 `seq 45 49`

do

echo "hec-$i mkdir iocon-pvfs2"

ssh hec-$i sudo /bin/mkdir -p /home/shuibing/iocon-pvfs2

ssh hec-$i sudo /bin/ls -lt /home/shuibing/iocon-pvfs2

done

**To start PVFS2 server**

pvfs2-server /mnt/common/shuibing/research/iocon/confs/8N.conf -f

pvfs2-server /mnt/common/shuibing/research/iocon/confs/8N.conf

**or use a script**

tests/scripts/start-pvfs2-8N.sh:

#!/bin/bash

#prepare PVFS2 storage spaces

#when we use sudo, we use the absolute path of executable file

for i in 40 41 43 `seq 45 49`

do

ssh hec-$i sudo /mnt/common/shuibing/install/sbin/pvfs2-server /mnt/common/shuibing/research/iocon/confs/8N.conf -f

done

**#start PVFS2 server program**

for i in 40 41 43 `seq 45 49`

do

ssh hec-$i sudo /mnt/common/shuibing/install/sbin/pvfs2-server /mnt/common/shuibing/research/iocon/confs/8N.conf

done

**To start pvfs2 client:**

**node：**

hec-41

ssh hec-41: # is the client of pvfs2

sudo insmod /mnt/common/shuibing/research/iocon/confs/pvfs2.ko

cd /mnt/common/shuibing/software/orangefs-2.8.6/src/apps/kernel/linux/

./pvfs2-client -p ./pvfs2-client-core

sudo mount -t pvfs2 tcp://hec-41:3334/pvfs2-fs /mnt/shuibing/iocon-pvfs2

## To uninstall pvfs2 clients

**stop-pvfs2-clients.sh**

#!/bin/bash

CLIENTDIR=/mnt/common/shuibing/software/orangefs-2.8.6/src/apps/kernel/linux

for i in 41

do

ssh hec-$i sudo umount /mnt/shuibing/iocon-pvfs2

sleep 2

ssh hec-$i sudo killall pvfs2-client

ssh hec-$i sudo rmmod pvfs2

done

**To uninstall pvfs2 servers**

**stop-pvfs2-8N.sh**

#!/bin/bash

#kill PVFS2 server

#when we use sudo, we use the absolute path of executable file

for i in 40 41 43 `seq 45 49`

do

ssh hec-$i sudo killall pvfs2-server &

done

sleep 2

#remove PVFS2 storage space

for i in 40 41 43 `seq 45 49`

do

ssh hec-$i sudo rm -rf /home/shuibing/iocon-pvfs2 &

done

**(2)MPI interface pvfs2 installation**

**1.Enter the software dirmak**

$cd /mnt/common/shuibing/software/mpich2-1.4.1p1

**To patch romio dir:(we do not need here)**

**cd mpich2-1.4.0p1/src/mpi/romio # change to ROMIO dir**

**prompt% patch -p1 < ~/src/romio-<CORRECT\_VERSION>.diff #apply patch**

**To configure MPI installation**

./configure --with-pvfs2=/mnt/common/shuibing/install/ -prefix=/mnt/common/shuibing/install/

make

make install

## 2.To start pvfs2 server with the same steps mentioned above.

**To produce pvfs2-server config file**

pvfs2-genconfig --quiet --protocol tcp --ioservers "hec-{40-41,43-43,45-49}" --metaservers "hec-{40-41,43-43,45-49}" --storage /home/shuibing/iocon-pvfs2 confs/8N.conf

**To prepare the storage space**

/home/shuibing/iocon-pvfs2

or script:

tests/scripts/mk-pvfs2-space.sh

#!/bin/bash

#prepare PVFS2 storage spaces

#when we use sudo, we use the absolute path of executable file

for i in 40 41 43 `seq 45 49`

do

echo "hec-$i mkdir iocon-pvfs2"

ssh hec-$i sudo /bin/mkdir -p /home/shuibing/iocon-pvfs2

ssh hec-$i sudo /bin/ls -lt /home/shuibing/iocon-pvfs2

done

**To start PVFS2 server**

pvfs2-server /mnt/common/shuibing/research/iocon/confs/8N.conf -f

pvfs2-server /mnt/common/shuibing/research/iocon/confs/8N.conf

**or use a script**

tests/scripts/start-pvfs2-8N.sh:

#!/bin/bash

#prepare PVFS2 storage spaces

#when we use sudo, we use the absolute path of executable file

for i in 40 41 43 `seq 45 49`

do

ssh hec-$i sudo /mnt/common/shuibing/install/sbin/pvfs2-server /mnt/common/shuibing/research/iocon/confs/8N.conf -f

done

**#start PVFS2 server program**

for i in 40 41 43 `seq 45 49`

do

ssh hec-$i sudo /mnt/common/shuibing/install/sbin/pvfs2-server /mnt/common/shuibing/research/iocon/confs/8N.conf

done

## 3.Client configuration

client node:

hec-41

hec-50

## sudo vi /etc/pvfs2tab, add entry

in hec-41:

tcp://hec-40:3334/pvfs2-fs /mnt/shuibing/iocon-pvfs2 pvfs2 defaults, noauto 0 0

in hec-50:

tcp://hec-45:3334/pvfs2-fs /mnt/shuibing/iocon-pvfs2 pvfs2 defaults, noauto 0 0

when use commands such as pvfs2-ls, pvfs2-ping and etc, these commands will access the io servers through the **path：**

**/mnt/shuibing/iocon-pvfs2**

according the entry in：

/etc/pvfs2tab or /etc/fstab

## （3）Run MPI program

**compiling IOR**

cd /mnt/common/shuibing/benchmarks/IOR

make mpiio (posix, mpiio, hdf5,ncmpi,all)

**run IOR**

/mnt/common/shuibing/benchmarks/IOR/src/C/IOR -a MPIIO -m -t 32k -i 1 -o pvfs2:/mnt/shuibing/iocon-pvfs2/test -b 32m

note: like this will result errors

IOR -a MPIIO -m -t 32k -i 1 -o pvfs2:/mnt/shuibing/iocon-pvfs2/test -b 32m

reference:

/export/home/huaiming/script/layout-adp/run-test.sh

**MPI multiple proceses tests:**

**shuibing@hec-50:**/mnt/common/shuibing/research/iocon/tests$

mpirun -np 8 /mnt/common/shuibing/benchmarks/IOR/src/C/IOR -a MPIIO -m -t 4k -i 1 -o pvfs2:/mnt/shuibing/iocon-pvfs2/test -b 8m

**-i N -m: 控制重复实验的次数，并且每次采用不同的文件。**

**-F：控制每个进程访问几个文件； 缺省情况所有进程共享一个文件。**

**设置文件属性**

shuibing@hec:/mnt/common/shuibing/research/iocon$ pvfs2-xattr

Please specify key if getting extended attributes

Usage: pvfs2-xattr -s {set xattrs} -k <key> -v <val> -t {print attributes} filename

**宋怀明设置不同文件夹布局属性：**

shuibing@hec-50:/export/home/huaiming/script/layout-adp$ vi setup-root.sh

#!/bin/bash

copy\_files(){

# for ((i=1;i<=9;i++))

# do

# scp pvfs2-io.conf hec-$i:/etc/

# done

for ((i=33;i<=64;i++))

#!/bin/bash

copy\_files(){

#!/bin/bash

#!/bin/bash

copy\_files(){

# for ((i=1;i<=9;i++))

# do

# scp pvfs2-io.conf hec-$i:/etc/

# done

for ((i=33;i<=64;i++))

do

scp pvfs2tab hec-$i:/etc/

done

}

create\_dir(){

pvfs2-mkdir /mnt/pvfs2-8ib/simple-${1}k

pvfs2-xattr -s -k user.pvfs2.dist\_name -v simple\_stripe /mnt/pvfs2-8ib/simple-${1}k

ssize=`expr ${1} \\* 1024`

pvfs2-xattr -s -k user.pvfs2.dist\_params -v strip\_size:${ssize} /mnt/pvfs2-8ib/simple-${1}k

pvfs2-chmod 777 /mnt/pvfs2-8ib/simple-${1}k

}

create\_dir2(){

pvfs2-mkdir /mnt/pvfs2-shm/simple-${1}k

pvfs2-xattr -s -k user.pvfs2.dist\_name -v simple\_stripe /mnt/pvfs2-shm/simple-${1}k

ssize=`expr ${1} \\* 1024`

pvfs2-xattr -s -k user.pvfs2.dist\_params -v strip\_size:${ssize} /mnt/pvfs2-shm/simple-${1}k

pvfs2-chmod 777 /mnt/pvfs2-shm/simple-${1}k

}

#copy\_files

hahaha(){

hms -n 1-9 killall -9 pvfs2-server

sleep 2

hms -n 1-9 rm -rf /home/pvfs2-storage-io

sleep 2

hms -n 1-4 pvfs2-server -f /etc/pvfs2-io.conf

hms -n 6-9 pvfs2-server -f /etc/pvfs2-io.conf

sleep 2

hms -n 1-4 pvfs2-server /etc/pvfs2-io.conf

hms -n 6-9 pvfs2-server /etc/pvfs2-io.conf

}

create\_dir 1

create\_dir 4

create\_dir 16

create\_dir 64

create\_dir 256

create\_dir 1024

create\_dir 4096

**My own setup\_path.sh**

#!/bin/bash

create\_dir(){

pvfs2-mkdir /mnt/shuibing/iocon-pvfs2/simple-${1}k

pvfs2-xattr -s -k user.pvfs2.dist\_name -v simple\_stripe /mnt/shuibing/iocon-pvfs2/simple-${1}k

ssize=`expr ${1} \\* 1024`

echo ${ssize}

pvfs2-xattr -s -k user.pvfs2.dist\_paramsi -v strip\_size:${ssize} /mnt/shuibing/iocon-pvfs2/simple-${1}k

pvfs2-chmod 777 /mnt/shuibing/iocon-pvfs2/simple-${1}k

}

create\_dir2(){

#for infiniband access

pvfs2-mkdir /mnt/pvfs2-8ib/simple-${1}k

pvfs2-xattr -s -k user.pvfs2.dist\_name -v simple\_stripe /mnt/pvfs2-8ib/simple-${1}k

ssize=`expr ${1} \\* 1024`

pvfs2-xattr -s -k user.pvfs2.dist\_params -v strip\_size:${ssize} /mnt/pvfs2-8ib/simple-${1}k

pvfs2-chmod 777 /mnt/pvfs2-8ib/simple-${1}k

}

hahaha(){

hms -n 40,41,43,45-49 killall -9 pvfs2-server

sleep 2

hms -n 40,41,43,45-49 rm -rf /home/shuibing/iocon-pvfs

sleep 2

hms -n 40,41,43,45-49 pvfs2-server -f /mnt/common/shuibing/research/iocon/confs/8N.conf

sleep 2

hms -n 40,41,43,45-49 pvfs2-server /mnt/common/shuibing/research/iocon/confs/8N.conf

}

create\_dir 1

create\_dir 4

create\_dir 16

create\_dir 64

create\_dir 256

create\_dir 1024

create\_dir 4096

**Run different IOR tests under different layout configures**

**scripts/ior/run\_test.sh**

#!/bin/bash

cd /mnt/common/shuibing/benchmarks/IOR/src/C

IOR-test(){

#note: awk use the output of former appliaciton as input

#so the 1 is the first field of the record from the output of pvfs2-ls

/usr/local/shm/mpich2-install/bin/mpirun -np ${1} ./IOR -a MPIIO -m -t ${2} -i 1 -o pvfs2:/mnt/pvfs2-shm/${4}/ior -b ${3}m >>/export/home/huaiming/layout-adp/lay1.log

pvfs2-ls /mnt/pvfs2-shm/${4} | awk -v DIR=${4} '{print "pvfs2-rm /mnt/pvfs2-shm/" DIR "/" $1 }' |sh

}

#IOR-test 8 4k 8 simple-4k

#cd -

test1(){

#for procnr in 2 8 32 128

for procnr in 8 32 128

do

for xsize in 64 256 1024

do

IOR-test ${procnr} ${xsize} 8 simple-4k

IOR-test ${procnr} ${xsize} 8 simple-16k

IOR-test ${procnr} ${xsize} 8 simple-64k

IOR-test ${procnr} ${xsize} 8 simple-256k

IOR-test ${procnr} ${xsize} 8 simple-1024k

IOR-test ${procnr} ${xsize} 8 simple-4096k

done

done

}

hahaha(){

for procnr in 8 32 128

do

for xsize in 4k 64k 256k

do

IOR-test ${procnr} ${xsize} 64 simple-4k

IOR-test ${procnr} ${xsize} 64 simple-16k

IOR-test ${procnr} ${xsize} 64 simple-64k

IOR-test ${procnr} ${xsize} 64 simple-256k

IOR-test ${procnr} ${xsize} 64 simple-1024k

IOR-test ${procnr} ${xsize} 64 simple-4096k

done

for xsize in 1m 4m 64m

do

IOR-test ${procnr} ${xsize} 512 simple-4k

IOR-test ${procnr} ${xsize} 512 simple-16k

IOR-test ${procnr} ${xsize} 512 simple-64k

IOR-test ${procnr} ${xsize} 512 simple-256k

IOR-test ${procnr} ${xsize} 512 simple-1024k

IOR-test ${procnr} ${xsize} 512 simple-4096k

done

done

}

hahaha

cd -

**analysis the results**

1. 打印需要的行，生成临时文件

sed -n -e /"test filename"/p -e /"clients"/p -e /"xfersize"/p -e /"blocksize"/p -e /"aggregate filesize"/p -e /"Max Write"/p -e /"Max Read"/p lay1.log > lay1.log.tmp

1. 替换临时文件中相关内容，以便更方便的输出数据

sed -i -e s/"pvfs2:\/mnt\/shuibing\/iocon-pvfs2\/simple\-"/""/g -e s/"\/ior"/""/g -e s/"Max\ Write:"/"

Write"/g -e s/"Max\ Read:"/" Read"/g lay1.log.tmp

3. 从临时文件中最后打印出按表格排列的直观的结果

awk 'BEGIN{

print "Strip\tProcNR\tXferSZ\tBlkSZ\tAGGSZ\tWrite\tRead";

}{

if($1=="test") printf("%s\t", $4);

if($1=="clients") printf("%d\t", $3);

if($1=="xfersize") printf("%d %s\t", $3,$4);

if($1=="blocksize") printf("%d\t", $3);

if($1=="aggregate") printf("%d %s\t", $4,$5);

if($1=="Write") printf("%f\t", $2);

if($1=="Read") printf("%f\n", $2);

}' lay1.log.tmp >lay1.txt

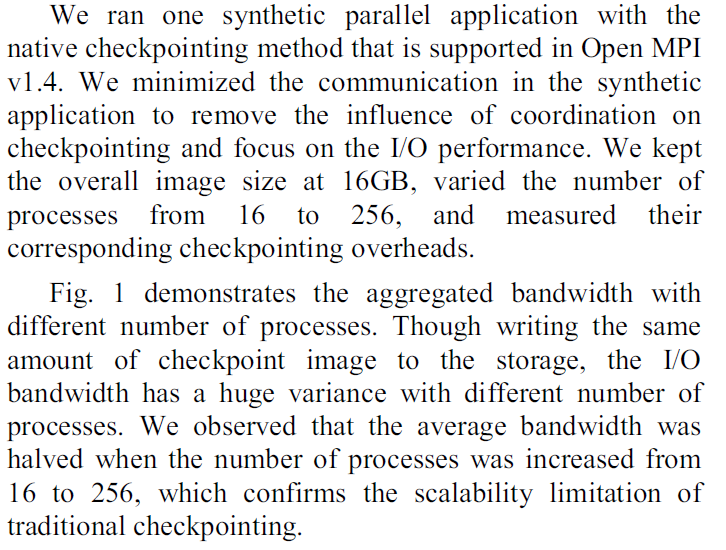
## (4)To test the configuration impact on performance

1. **~~stripe\_size~~**
2. **~~blocks size~~**
3. **~~transfersize~~**

# concurrency

测试方式：

（1）金辉：固定4 I/O server, 和数据总量，变化进程数。



**To assign clients(compute nodes), and servers( I/O nodes)**

**clients.txt:指定客户端**

hec-40

hec-41

hec-43

hec-45

**servers.txt**

hec-46

hec-47

hec-48

hec-49

**（1）To generate pvfs2-configuration file**

servers=`awk '{printf("%s,",$1)}' servers.txt`

number=`awk 'END{print NR}' servers.txt`

pvfs2-genconfig --quiet --protocol tcp --ioservers $servers --metaservers $servers --storage /home/shuibing/iocon-pvfs2 pvfs2-${number}N.conf

**（2）setup-root.sh**

#!/bin/bash

servers=`awk '{print $1}' servers.txt`

firstnode=`awk 'NR==1' servers.txt`

clients=`awk '{print $1}' clients.txt`

number=`awk 'END{print NR}' servers.txt`

hahaha(){

hms -n 46-49 sudo killall -9 pvfs2-server

sleep 2

hms -n 46-49 rm -rf /home/shuibing/iocon-pvfs2

sleep 2

sleep 2

}

start\_server(){

echo $node

sleep 2

for node in $servers

do

ssh $node sudo rm -rf /home/shuibing/iocon-pvfs2

done

do

done

sleep 2

for node in $servers

do

done

}

#copy\_file()

start\_client()

{

for node in $clients

do

echo $node

sudo scp pvfs2tab $node:/etc/

ssh $node sudo cat /etc/pvfs2tab

#echo "pvfs2 root dir"

ssh $node pvfs2-ls /mnt/shuibing/iocon-pvfs2/

done

}

create\_dir(){

pvfs2-mkdir /mnt/shuibing/iocon-pvfs2/simple-${1}k

pvfs2-xattr -s -k user.pvfs2.dist\_name -v simple\_stripe /mnt/shuibing/iocon-pvfs2/simple-${1}k

ssize=`expr ${1} \\* 1024`

pvfs2-xattr -s -k user.pvfs2.dist\_params -v strip\_size:${ssize} /mnt/shuibing/iocon-pvfs2/simple-${1}k

pvfs2-chmod 777 /mnt/shuibing/iocon-pvfs2/simple-${1}k

}

create\_dir2(){

#for infiniband access

pvfs2-mkdir /mnt/pvfs2-8ib/simple-${1}k

pvfs2-xattr -s -k user.pvfs2.dist\_name -v simple\_stripe /mnt/pvfs2-8ib/simple-${1}k

ssize=`expr ${1} \\* 1024`

pvfs2-xattr -s -k user.pvfs2.dist\_params -v strip\_size:${ssize} /mnt/pvfs2-8ib/simple-${1}k

pvfs2-chmod 777 /mnt/pvfs2-8ib/simple-${1}k

}

create\_client\_dir(){

create\_dir 1

create\_dir 4

create\_dir 16

create\_dir 64

create\_dir 256

create\_dir 1024

create\_dir 4096

}

start\_server

start\_client

算术运算：

(1)expr

CNUM=`expr $GNUM \\* $GSIZE`

io-schedule/run.sh: CNUM=`expr $GNUM \\* $GSIZE`

io\_test/run16n.sh: reqs=`expr 1600 / ${clients}`

io\_test/run8n.sh: reqs=`expr 1600 / ${clients}`

io\_test/run.sh: reqs=`expr 3200 / ${clients}`

io\_test/run64n.sh: reqs=`expr 6400 / ${clients}`

io\_test/run32n.sh: reqs=`expr 3200 / ${clients}`

layout-adp/setup-root.sh: ssize=`expr ${1} \\* 1024`

layout-adp/setup-root.sh: ssize=`expr ${1} \\* 1024`

注意：乘时需要加转义字符：\

(2)bc -l

childnum=$(echo "2 ^ $j"| bc -l)

multiprocess/concurrency.sh: TotResptime=$(echo "$TotResptime + $ChildResptime" | bc )

multiprocess/concurrency.sh: #IPS=`echo "$Tot\_ins / $Wall\_time"|bc -l`

multiprocess/concurrency.sh: #IOPS=`echo "$Tot\_ios / $Wall\_time"|bc -l`

**(3)To run test**

**run-test.sh &&run-test2.sh(multiple clients)**

#!/bin/bash

exec=/mnt/common/shuibing/benchmarks/IOR/src/C/IOR

IOR-test(){

mpirun -np ${1} -f clients.txt ${exec} -a MPIIO -m -t ${2} -i 1 -o pvfs2:/mnt/shuibing/iocon-pvfs2/ior -b ${3}m >>/mnt/common/shuibing/research/iocon/script/concur2.log

#note: awk use the output of former appliaciton as input

#so the $1 is the first field of the record from the output of pvfs2-ls

pvfs2-ls /mnt/shuibing/iocon-pvfs2/ |awk '{print "pvfs2-rm /mnt/shuibing/iocon-pvfs2/"$1 }' |sh

}

hahaha(){

#total 8G size file

aggsize=8192

for xsize in 4k 64k 256k 1M

do

for procnr in 8 32 64 128

do

bsize=`expr 8192 / ${procnr}`

# echo "IOR-test ${procnr} ${xsize} ${bsize} "

IOR-test ${procnr} ${xsize} ${bsize}

done

done

}

hahaha

**(4)get the clear result from the log file**

**sort2.sh**

#!/bin/bash

create\_tmp(){

sed -n -e /"clients"/p -e /"xfersize"/p -e /"blocksize"/p -e /"aggregate filesize"/p -e /"Max Write"/p -e /"Max Read"/p concur2.log > concur2.log.tmp

}

replace\_tmp(){

sed -i -e s/"Max\ Write:"/" Write"/g -e s/"Max\ Read:"/" Read"/g concur2.log.tmp

}

sort\_result(){

awk 'BEGIN{print "ProcNR\tXferSZ\tBlkSZ\tAGGSZ\tWrite\tRead";}

{

if($1=="clients") printf("%d\t", $3);

if($1=="xfersize") printf("%d %s\t", $3,$4);

if($1=="blocksize") printf("%d %s\t", $3,$4);

if($1=="aggregate") printf("%d %s\t", $4,$5);

if($1=="Write") printf("%f\t", $2);

if($1=="Read") printf("%f\n", $2);

}' concur2.log.tmp >concur2.txt

rm -rf concur2.log.tmp

}

create\_tmp

replace\_tmp

sort\_result

**(5)The primary results**

**a:concurr1.txt**

**ProcNR XferSZ BlkSZ AGGSZ Write Read**

8 4096 bytes 1 GiB 8 GiB 45.080000 59.840000

32 4096 bytes 256 MiB 8 GiB 74.370000 109.980000

64 4096 bytes 128 MiB 8 GiB 56.540000 109.680000

128 4096 bytes 64 MiB 8 GiB 35.430000 72.420000

8 65536 bytes 1 GiB 8 GiB 104.700000 112.520000

32 65536 bytes 256 MiB 8 GiB 111.330000 112.740000

64 65536 bytes 128 MiB 8 GiB 109.080000 112.790000

128 65536 bytes 64 MiB 8 GiB 110.150000 112.620000

8 262144 bytes 1 GiB 8 GiB 109.380000 112.850000

32 262144 bytes 256 MiB 8 GiB 110.800000 112.700000

64 262144 bytes 128 MiB 8 GiB 110.770000 112.390000

128 262144 bytes 64 MiB 8 GiB 112.160000 110.880000

8 1 MiB 1 GiB 8 GiB 112.870000 112.900000

32 1 MiB 256 MiB 8 GiB 112.580000 112.660000

64 1 MiB 128 MiB 8 GiB 112.790000 112.470000

128 1 MiB 64 MiB 8 GiB 111.690000 112.310000

**b:concurr2.txt**

**ProcNR XferSZ BlkSZ AGGSZ Write Read**

8 4096 bytes 1 GiB 8 GiB 42.830000 51.040000

32 4096 bytes 256 MiB 8 GiB 71.670000 138.700000

64 4096 bytes 128 MiB 8 GiB 59.420000 134.340000

128 4096 bytes 64 MiB 8 GiB 55.440000 74.590000

8 65536 bytes 1 GiB 8 GiB 129.600000 250.160000

32 65536 bytes 256 MiB 8 GiB 144.950000 440.400000

64 65536 bytes 128 MiB 8 GiB 136.860000 444.370000

128 65536 bytes 64 MiB 8 GiB 142.420000 444.520000

8 262144 bytes 1 GiB 8 GiB 135.660000 435.070000

32 262144 bytes 256 MiB 8 GiB 145.370000 447.530000

64 262144 bytes 128 MiB 8 GiB 133.220000 447.150000

128 262144 bytes 64 MiB 8 GiB 133.550000 447.490000

8 1 MiB 1 GiB 8 GiB 197.260000 417.120000

32 1 MiB 256 MiB 8 GiB 169.190000 440.270000

64 1 MiB 128 MiB 8 GiB 197.460000 442.640000

128 1 MiB 64 MiB 8 GiB 208.660000 444.450000

**c: concur1\_1.txt**

**1.一个进程在一个客户端上，访问令一个节点组成的pvfs**

ProcNR XferSZ BlkSZ AGGSZ Write Read

1 4096 bytes 1 GiB 1 GiB 8.110000 8.820000

1 65536 bytes 1 GiB 1 GiB 37.340000 46.010000

1 262144 bytes 1 GiB 1 GiB 59.630000 74.790000

1 1 MiB 1 GiB 1 GiB 91.020000 86.900000

~

**2. 多进程一个客户端**

ProcNR XferSZ BlkSZ AGGSZ Write Read

1 4096 bytes 1 GiB 1 GiB 7.680000 8.280000

4 4096 bytes 256 MiB 1 GiB 21.900000 28.450000

8 4096 bytes 128 MiB 1 GiB 18.270000 48.890000

32 4096 bytes 32 MiB 1 GiB 16.240000 54.370000

128 4096 bytes 8 MiB 1 GiB 19.690000 54.800000

1 65536 bytes 1 GiB 1 GiB 35.300000 38.430000

4 65536 bytes 256 MiB 1 GiB 47.000000 111.090000

8 65536 bytes 128 MiB 1 GiB 38.020000 111.270000

32 65536 bytes 32 MiB 1 GiB 33.390000 111.550000

128 65536 bytes 8 MiB 1 GiB 35.950000 110.990000

1 262144 bytes 1 GiB 1 GiB 59.790000 75.090000

4 262144 bytes 256 MiB 1 GiB 77.900000 112.820000

8 262144 bytes 128 MiB 1 GiB 46.300000 112.640000

32 262144 bytes 32 MiB 1 GiB 65.330000 112.430000

128 262144 bytes 8 MiB 1 GiB 47.090000 112.420000

1 1 MiB 1 GiB 1 GiB 89.090000 86.270000

4 1 MiB 256 MiB 1 GiB 59.960000 112.970000

8 1 MiB 128 MiB 1 GiB 47.720000 111.460000

32 1 MiB 32 MiB 1 GiB 58.300000 112.940000

128 1 MiB 8 MiB 1 GiB 70.710000 108.010000

**3.多进程多个客户端**

ProcNR XferSZ BlkSZ AGGSZ Write Read

1 4096 bytes 1 GiB 1 GiB 7.810000 8.730000

4 4096 bytes 256 MiB 1 GiB 20.050000 28.080000

8 4096 bytes 128 MiB 1 GiB 17.370000 48.120000

32 4096 bytes 32 MiB 1 GiB 17.810000 53.830000

128 4096 bytes 8 MiB 1 GiB 19.430000 51.800000

256 4k 4MB 1GB 4.65 39.47

512 4K 4MB 2GB 2.36 42.42

1 65536 bytes 1 GiB 1 GiB 36.750000 39.310000

4 65536 bytes 256 MiB 1 GiB 40.800000 112.480000

8 65536 bytes 128 MiB 1 GiB 36.120000 112.050000

32 65536 bytes 32 MiB 1 GiB 33.420000 109.940000

128 65536 bytes 8 MiB 1 GiB 35.830000 111.800000

256 64k 4MB 1GB 39.08 111.44

512 64K 4MB 2GB 18.26 110.75

1 262144 bytes 1 GiB 1 GiB 55.960000 69.630000

4 262144 bytes 256 MiB 1 GiB 52.070000 112.450000

8 262144 bytes 128 MiB 1 GiB 47.420000 112.460000

32 262144 bytes 32 MiB 1 GiB 43.170000 112.900000

128 262144 bytes 8 MiB 1 GiB 56.230000 112.660000

256 256K 4MB 1GB

512 256K 4MB 2GB 18.35 111.09

1 1 MiB 1 GiB 1 GiB 90.270000 90.600000

4 1 MiB 256 MiB 1 GiB 58.630000 112.960000

8 1 MiB 128 MiB 1 GiB 47.900000 112.820000

32 1 MiB 32 MiB 1 GiB 61.940000 112.810000

128 1 MiB 8 MiB 1 GiB 69.470000 110.790000

256 1M 4MB 1GB

512 1M 4MB 2GB 5.81 107.89

**从上面可以看出，单个传输请求特别小时，并行的影响特别大；**

**当单个传输请求大小比较大时，这种干扰影响可以降低。**

# 进入核心并行化测试

**(1)固定xfersize=4k, 总大小8G，改变进程数，进行只与并行性相关的测试效果，将问题进一步简单化：**

run-test4.sh

#!/bin/bash

exec=/mnt/common/shuibing/benchmarks/IOR/src/C/IOR

IOR-test(){

#note: awk use the output of former appliaciton as input

#so the $1 is the first field of the record from the output of pvfs2-ls

pvfs2-ls /mnt/shuibing/iocon-pvfs2/ |awk '{print "pvfs2-rm /mnt/shuibing/iocon-pvfs2/"$1 }' |sh

}

hahaha(){

#total 8G size file

aggsize=8192

for xsize in 64k

do

for procnr in 1 4 8 32 64 128 256 512

do

bsize=`expr 8192 / ${procnr}`

IOR-test ${procnr} ${xsize} ${bsize}

done

done

}

rm -rf /mnt/common/shuibing/research/iocon/script/concur1.log

hahaha

**（2）SSD性能测试**

**有SSD**

**hec-40 45 46 47 50**

**servers-ssd.txt:**

**hec-45**

**hec-46**

**hec-47**

**hec-50**

**clients-ssd.txt**

**hec-40**

**hec-41**

**hec-43**

**hec-48**

**清空cache指令：**

**sudo su -c " echo 3 > /proc/sys/vm/drop\_caches "**

**a) generate pvfs2-configure**

**genconfig-ssd.sh**

servers=`awk '{printf("%s,",$1)}' servers-ssd.txt`

number=`awk 'END{print NR}' servers-ssd.txt`

pvfs2-genconfig --quiet --protocol tcp --ioservers $servers --metaservers $servers --storage /mnt/ssd/iocon-pvfs2 pvfs2-${number}N-ssd.conf

**b)setup-root-ssd.sh**

#!/bin/bash

servers=`awk '{print $1}' servers-ssd.txt`

firstnode=`awk 'NR==1' servers-ssd.txt`

clients=`awk '{print $1}' clients-ssd.txt`

number=`awk 'END{print NR}' servers-ssd.txt`

#pvfs2-genconfig --quiet --protocol tcp --ioservers $servers --metaservers $servers --storage /home/shuibing/iocon-pvfs2 pvfs2-${number}N.conf

stop\_server(){

for node in $servers

do

echo $node

ssh $node sudo killall -9 pvfs2-server

done

sleep 2

for node in $servers

do

ssh $node sudo rm -rf /mnt/ssd/iocon-pvfs2

done

}

start\_server(){

for node in $servers

do

ssh $node sudo /mnt/common/shuibing/install/sbin/pvfs2-server -f /mnt/common/shuibing/research/iocon/script/pvfs2-${number}N-ssd.conf

done

sleep 2

for node in $servers

do

ssh $node sudo /mnt/common/shuibing/install/sbin/pvfs2-server /mnt/common/shuibing/research/iocon/script/pvfs2-${number}N-ssd.conf

done

}

#copy\_file()

start\_client()

{

echo "tcp://${firstnode}:3334/pvfs2-fs /mnt/shuibing/iocon-pvfs2 pvfs2 defaults,noauto 0 0" >pvfs2tab

for node in $clients

do

echo $node

sudo scp pvfs2tab $node:/etc/

ssh $node sudo cat /etc/pvfs2tab

#echo "pvfs2 root dir"

ssh $node pvfs2-ls /mnt/shuibing/iocon-pvfs2/

done

}

create\_dir(){

pvfs2-mkdir /mnt/shuibing/iocon-pvfs2/simple-${1}k

pvfs2-xattr -s -k user.pvfs2.dist\_name -v simple\_stripe /mnt/shuibing/iocon-pvfs2/simple-${1}k

ssize=`expr ${1} \\* 1024`

pvfs2-xattr -s -k user.pvfs2.dist\_params -v strip\_size:${ssize} /mnt/shuibing/iocon-pvfs2/simple-${1}k

pvfs2-chmod 777 /mnt/shuibing/iocon-pvfs2/simple-${1}k

}

create\_dir2(){

#for infiniband access

pvfs2-mkdir /mnt/pvfs2-8ib/simple-${1}k

pvfs2-xattr -s -k user.pvfs2.dist\_name -v simple\_stripe /mnt/pvfs2-8ib/simple-${1}k

ssize=`expr ${1} \\* 1024`

pvfs2-xattr -s -k user.pvfs2.dist\_params -v strip\_size:${ssize} /mnt/pvfs2-8ib/simple-${1}k

pvfs2-chmod 777 /mnt/pvfs2-8ib/simple-${1}k

}

create\_client\_dir(){

create\_dir 1

create\_dir 4

create\_dir 16

create\_dir 64

create\_dir 256

create\_dir 1024

create\_dir 4096

}

}

stop\_server

start\_server

start\_client

**c)运行测试程序**

**run-ssd.sh**

#!/bin/bash

exec=/mnt/common/shuibing/benchmarks/IOR/src/C/IOR

IOR-test(){

mpirun -np ${1} -f clients-ssd.txt ${exec} -a MPIIO -m -t ${2} -i 1 -o pvfs2:/mnt/shuibing/iocon-pvfs2/ior -b ${3}m >>/mnt/common/shuibing/research/iocon/script/concur1.log

#note: awk use the output of former appliaciton as input

#so the $1 is the first field of the record from the output of pvfs2-ls

pvfs2-ls /mnt/shuibing/iocon-pvfs2/ |awk '{print "pvfs2-rm /mnt/shuibing/iocon-pvfs2/"$1 }' |sh

}

hahaha(){

#total 8G size file

aggsize=8192

#for xsize in 4k 16k 64k 256k

for xsize in 4k 16k 64k 256k 1m 4m 16m

#for xsize in 1m 4m 16m

do

for procnr in 1 4 8 32 64 128 256 512

do

bsize=`expr 8192 / ${procnr}`

IOR-test ${procnr} ${xsize} ${bsize}

done

done

}

#rm -rf /mnt/common/shuibing/research/iocon/script/concur1.log

hahaha

）

**d）结果和分析**

1.结果见[iocon-result1-20120729](https://docs.google.com/spreadsheet/ccc?key=0AmYrx4wQIG0edEN5R2owa0puVnVmUnljcGVXRFktaFE#gid=0)

**发现hdd和ssd性能基本一致**

怀疑服务器端用了cache导致的（本地文件系统的cache）

2. 测试本地文件系统性能和本地直接磁盘操作性能

修改各节点上home/shuibing/读写权限

for i in 40 41 43 `seq 45 50`;do ssh hec-$i sudo chown -hR shuibing /home/shuibing;ssh hec-$i ls -al /home/;done

1.带缓冲的顺序和随机差距不大

2.直接IO两个有较大差距，但是随机读写差距（-z）更大。

4k顺序写 顺序读 随机写 随机读：

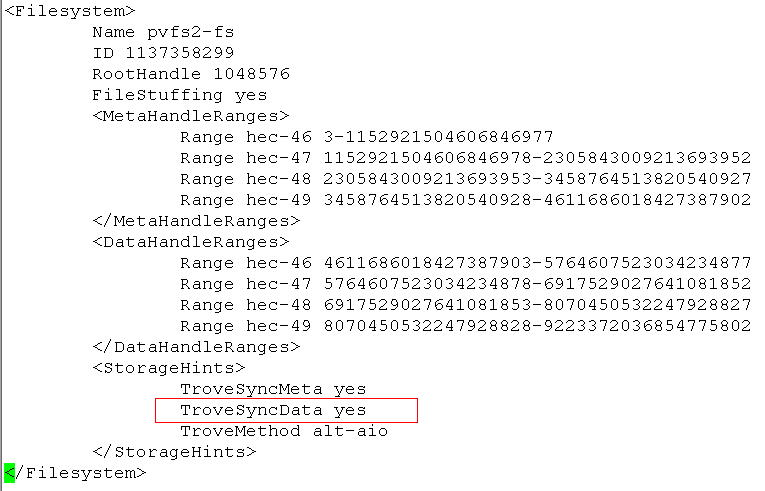
ssd 30 21.16 21.48 31.47M

hdd 5.72 10.31 5.82 11.53M

**3pvfs2下面，缓冲的顺序和随机区别不大，ssd和hdd区别不大**

# pvfs2测试数据同步性能

1：将trovedatasync set true



?? pvfs2tab: parameters

2. [result](https://docs.google.com/spreadsheet/ccc?key=0AmYrx4wQIG0edEN5R2owa0puVnVmUnljcGVXRFktaFE#gid=0)

observations: 在同步条件下ssd性能和hdd性能依然差不多

3. iozone , ior 本地direct测试，可以看出并行条件下效果变差

sudo /mnt/common/shuibing/install/bin/iozone\_ori -s 64m -r 4k -I -i 0 -i 1 -+n -w -t 1

。。。

sudo /mnt/common/shuibing/install/bin/iozone\_ori -s 1m -r 4k -I -i 0 -i 1 -+n -w -t 64

4.实验设计有误：

# iocon实验并行性干扰测试1

**(1)当文件大于mem大小时，降低cache影响**

ProcNR XferSZ BlkSZ AGGSZ Write Read

4 262144 bytes 10 GiB 40 GiB 68.560000 63.000000

8 262144 bytes 5 GiB 40 GiB 109.490000 63.590000

16 262144 bytes 2 GiB 40 GiB 117.100000 56.140000

32 262144 bytes 1 GiB 40 GiB 109.410000 53.890000

64 262144 bytes 640 MiB 40 GiB 113.590000 44.720000

128 262144 bytes 320 MiB 40 GiB 120.100000 41.420000

4 1 MiB 10 GiB 40 GiB 154.790000 121.550000

8 1 MiB 5 GiB 40 GiB 144.020000 89.310000

16 1 MiB 2 GiB 40 GiB 140.750000 77.740000

32 1 MiB 1 GiB 40 GiB 143.880000 61.360000

64 1 MiB 640 MiB 40 GiB 151.660000 56.450000

**（2）先写一个大点的文件，不删除，然后读，在刷新cache，重新读。**

**IOR -k:保存文件不删除**

**sudo su -c "sync; echo 3 >/proc/sys/vm/drop\_caches"**

**+++测试脚本：**

run-test.sh

#!/bin/bash

exec=/mnt/common/shuibing/benchmarks/IOR/src/C/IOR

servers=`awk '{print $1}' servers.txt`

fm=/mnt/common/shuibing/research/iocon/script/fm.sh

IOR-read(){

for node in $servers

do

#ssh $node sudo su -c "sync; echo 3 >/proc/sys/vm/drop\_caches"

ssh $node ${fm}

done

mpirun -np ${1} -f clients.txt ${exec} -a MPIIO -m -t ${2} -i 1 -r -o pvfs2:/mnt/iocon-pvfs2/ior -b ${3}m -k >>/mnt/common/shuibing/research/iocon/script/concur1.log

}

prepare(){

mpirun -np 4 -f clients.txt ${exec} -a MPIIO -m -i 1 -o pvfs2:/mnt/iocon-pvfs2/ior -w -t 1m -b 256M -k

}

clean(){

pvfs2-ls /mnt/iocon-pvfs2/ |awk '{print "pvfs2-rm /mnt/iocon-pvfs2/"$1 }' |sh

}

hahaha(){

#total 4\*8G size file, remove the cache effect

for xsize in 64k 256K 1m

do

for procnr in 4 8 16 32 64 128

do

bsize=`expr 1024 / ${procnr}`

IOR-read ${procnr} ${xsize} ${bsize}

done

done

}

#rm -rf /mnt/common/shuibing/research/iocon/script/concur1.log

prepare

#IOR-read

hahaha

clean

**+++测试结果：结果文件：**[**read1Ggood.txt**](https://docs.google.com/spreadsheet/ccc?key=0AmYrx4wQIG0edHF6aHBfVnJkVzlOaEZTU0R4THZrX3c#gid=0)

**+++结论：相当好，进程越多，性能越差**

# iocon实验并行性干扰测试2

**1. 指定pvfs2目录下面某一个目录的布局（比如不stripe），这样改路径下的文件操作将局限于一个固定的I/O server, 不会分散到多个I/O servers.**

pvfs2-xattr -s -k user.pvfs2.dist\_name -v simple\_stripe /mnt/pvfs2-8ib/simple-${1}k

ssize=`expr ${1} \\* 1024`

pvfs2-xattr -s -k user.pvfs2.dist\_params -v strip\_size:${ssize} /mnt/pvfs2-8ib/simple-${1}k

pvfs2-chmod 777 /mnt/pvfs2-8ib/simple-${1}k

## 2. 4hdd+1ssd测试情况

测试结果 hybrid421.txt

## 3. 3hdd+1ssd测试情况

**client端可以完全不改变，而是把server端的某一个存储路径改掉。**

****

storage server: /home/shuibing/iocon-pvfs2

**let ssd is this path**

**we choose node hec-46:**

cd /mnt/ssd

mkdir iocon-pvfs2-ssd

ln -s iocon-pvfs2-ssd /home/shuibing/iocon-pvfs2

and in setup-root.sh

we do not remove this path:/home/shuibing/iocon-pvfs2

测试结果：

hybrid321.txt

# Benchmark mpi-tile-io

README:

#

# Rob Ross, 10/05/2001

#

# mpi-tile-io.c - a tile reading MPI-IO application

The purpose of this application is to test the performance of an

underlying MPI-IO and file system implementation under a noncontiguous

access workload.

The application logically divides a data file into a dense two dimensional

set of tiles, as shown below:

---------------------------------------------

| | | | |

| | | | |

| | | | |

---------------------------------------------

| | | | |

| | | | |

| | | | |

---------------------------------------------

| | | | |

| | | | |

| | | | |

---------------------------------------------

The parameters controlling the interpretation of the file are as

follows:

nr\_tiles\_x - number of tiles in the X dimension (rows)

nr\_tiles\_y - number of tiles in the Y dimension (columns)

sz\_tile\_x - number of elements in the X dimension of a tile

sz\_tile\_y - number of elements in the Y dimension of a tile

sz\_element - size of an element in bytes

overlap\_x - number of elements shared between adjacent tiles in the X

dimension

overlap\_y - number of elements shared between adjacent tiles in the Y

dimension

header\_bytes - number of bytes at beginning of file to consider as

header data (skipped over)

filename - name of file to operate on

Additionally the following parameters control the behavior of the

application:

collective - perform I/O collectively

cb\_config\_list - control aggregation

All of these parameters may be passed using the double-dash "--"

notation; for example:

**mpirun -np 100 mpi-tile-io --nr\_tiles\_x 25 --nr\_tiles\_y 4 --sz\_tile\_x \**

**100 --sz\_tile\_y 100 --sz\_element 32 --filename /pvfs/foo**

Alternatively all caps versions of these same parameter names may be

used as environment variables for passing in parameters. Command line

parameters take precedence over environment variables.

The application will report the min, mean, max, and variance for open,

close, and read times.

Finally, the parameter "write\_file" should be used to create a datafile for

subsequent use in testing. This mode will report statistics as well,

but note that the application overwrites the overlap regions and

generally doesn't try to do the smart thing in this case; it is just

trying to get a datafile created for later use.

The application does some checking on the number of processes available

and the number of tiles to write. It will throw out extra processes,

removing them from collective operations. It will return an error if

fewer processes are available than tiles requested.

mpi-tile-io does not assume that all processes are passed command line

arguments (because MPI does not guarantee this); instead the command

line is parsed on process 0, and all pertinent parameters are broadcast

from there. Likewise environment variables are parsed in the same way.

# IOsig

shuibing@hec:/export/home/yyin/research/iosig/sc11\_experiments/benchmarks/mpi-tile-io/test$ cat trace\_r0.out

（1）编译：

mpi-tile-io: mpi-tile-io.c

$(CC) $(CFLAGS) mpi-tile-io.c -o $@ **-L/export/home/yyin/mpich2-install/lib -lpushio**

#-L/home/yanlong/data/installed\_programs/mpich2-1.1.1p1/lib -lpushio

MPI\_Rank File # File Pos # of Bytes Time(s) I/O Op

=====================================================================================

0 0 0 0 0.165454 MPI\_OPEN

0 32606 0 32000000 0.200615 MPI\_READ

0 0 0 0 20.250397 MPI\_CLOSE

MPI\_Rank File # File Pos # of Bytes Time(s) I/O Op

=====================================================================================

0 0 0 0 0.148651 MPI\_OPEN

0 32618 0 32000000 0.175565 MPI\_READ

0 0 0 0 1.052824 MPI\_CLOSE

MPI\_Rank File # File Pos # of Bytes Time(s) I/O Op

=====================================================================================

0 0 0 0 0.536149 MPI\_OPEN

0 32647 0 32000000 0.583743 MPI\_READ

0 0 0 0 12.524716 MPI\_CLOSE

MPI\_Rank File # File Pos # of Bytes Time(s) I/O Op

=====================================================================================

0 0 0 0 0.258164 MPI\_OPEN

0 32601 0 32000000 0.292484 MPI\_READ

0 0 0 0 6.419643 MPI\_CLOSE

MPI\_Rank File # File Pos # of Bytes Time(s) I/O Op

=====================================================================================

0 0 0 0 0.135479 MPI\_OPEN

0 32543 0 32000000 0.164667 MPI\_READ

0 0 0 0 1.712290 MPI\_CLOSE

MPI\_Rank File # File Pos # of Bytes Time(s) I/O Op

=====================================================================================

0 0 0 0 0.154180 MPI\_OPEN

0 32679 0 32000000 0.181009 MPI\_READ

0 0 0 0 1.495386 MPI\_CLOSE

MPI\_Rank File # File Pos # of Bytes Time(s) I/O Op

=====================================================================================

0 0 0 0 0.153184 MPI\_OPEN

0 32739 0 32000000 0.182074 MPI\_READ

0 0 0 0 1.768756 MPI\_CLOSE

（2）重新编译带iosig的IOR版本

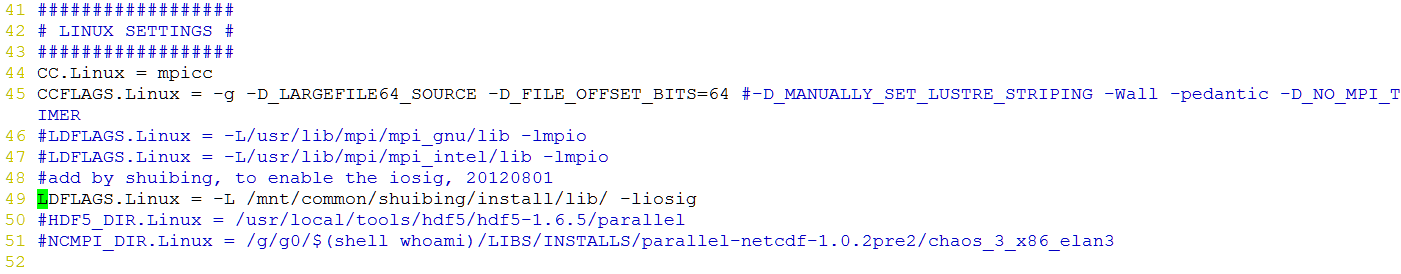
注意新选项在src/C/Makefile.config中设置，该文件被包含在Makefile中:

a:cd xx/src/C/,

vi Makefile.config, modify line 49 as follow

+++:

**-L /mnt/common/shuibing/install/lib/ -liosig**



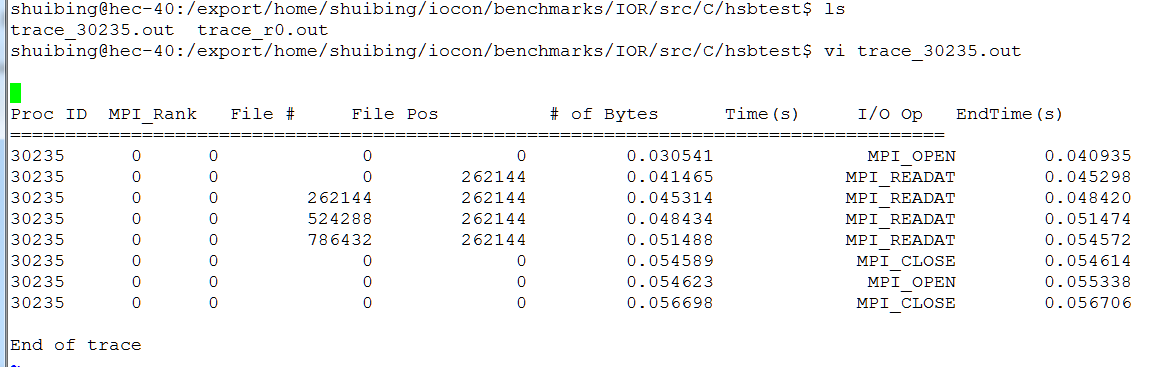
b.在新编译后，将生成文件重命名为：IOR-iosig以示区别 mv IOR IOR-orig

c:mkdir hsbtest;

d: 进行测试

cd hsbtest

../IOR-iosig -a MPIIO -o pvfs2:/mnt/iocon-pvfs2/ior.0 -r -k



Proc ID MPI\_Rank File # File Pos # of Bytes Start Time(s) I/O Op EndTime(s)

=====================================================================================

30235 0 0 0 0 0.030541 MPI\_OPEN 0.040935

30235 0 0 0 262144 0.041465 MPI\_READAT 0.045298

30235 0 0 262144 262144 0.045314 MPI\_READAT 0.048420

30235 0 0 524288 262144 0.048434 MPI\_READAT 0.051474

30235 0 0 786432 262144 0.051488 MPI\_READAT 0.054572

30235 0 0 0 0 0.054589 MPI\_CLOSE 0.054614

30235 0 0 0 0 0.054623 MPI\_OPEN 0.055338

30235 0 0 0 0 0.056698 MPI\_CLOSE 0.056706

End of trace

# NPB BTIO

(1)修改make.def

shuibing@hec:/export/home/shuibing/hsb/benchmarks/NPB2.4/NPB2.4-MPI$ cat config/make.def

#---------------------------------------------------------------------------

#

# SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS.

#

#---------------------------------------------------------------------------

#---------------------------------------------------------------------------

# Items in this file will need to be changed for each platform.

# (Note these definitions are inconsistent with NPB2.1.)

#---------------------------------------------------------------------------

#---------------------------------------------------------------------------

# Parallel Fortran:

#

# For CG, EP, FT, MG, LU, SP and BT, which are in Fortran, the following must

# be defined:

#

# MPIF77 - Fortran compiler

# FFLAGS - Fortran compilation arguments

# FMPI\_INC - any -I arguments required for compiling MPI/Fortran

# FLINK - Fortran linker

# FLINKFLAGS - Fortran linker arguments

# FMPI\_LIB - any -L and -l arguments required for linking MPI/Fortran

#

# compilations are done with $(MPIF77) $(FMPI\_INC) $(FFLAGS) or

# $(MPIF77) $(FFLAGS)

# linking is done with $(FLINK) $(FMPI\_LIB) $(FLINKFLAGS)

#---------------------------------------------------------------------------

#---------------------------------------------------------------------------

# This is the fortran compiler used for MPI programs

#---------------------------------------------------------------------------

MPIF77 = mpif77

# This links MPI fortran programs; usually the same as ${MPIF77}

FLINK = ${MPIF77}

#---------------------------------------------------------------------------

# These macros are passed to the linker to help link with MPI correctly

#---------------------------------------------------------------------------

#FMPI\_LIB = -L/usr/local/lib -lmpi

#shubing 20120801

FMPI\_LIB = -L/mnt/common/shuibing/install/lib -lmpich -liosig

#---------------------------------------------------------------------------

# These macros are passed to the compiler to help find 'mpif.h'

#---------------------------------------------------------------------------

#FMPI\_INC = -I/usr/local/include

FMPI\_INC = -I/mnt/common/shuibing/install/include

#---------------------------------------------------------------------------

# Global \*compile time\* flags for Fortran programs

#---------------------------------------------------------------------------

FFLAGS = -O3

# FFLAGS = -g

#---------------------------------------------------------------------------

# Global \*link time\* flags. Flags for increasing maximum executable

# size usually go here.

#---------------------------------------------------------------------------

FLINKFLAGS =

#---------------------------------------------------------------------------

# Parallel C:

#

# For IS, which is in C, the following must be defined:

#

# MPICC - C compiler

# CFLAGS - C compilation arguments

# CMPI\_INC - any -I arguments required for compiling MPI/C

# CLINK - C linker

# CLINKFLAGS - C linker flags

# CMPI\_LIB - any -L and -l arguments required for linking MPI/C

#

# compilations are done with $(MPICC) $(CMPI\_INC) $(CFLAGS) or

# $(MPICC) $(CFLAGS)

# linking is done with $(CLINK) $(CMPI\_LIB) $(CLINKFLAGS)

#---------------------------------------------------------------------------

#---------------------------------------------------------------------------

# This is the C compiler used for MPI programs

#---------------------------------------------------------------------------

MPICC = mpicc

# This links MPI C programs; usually the same as ${MPICC}

CLINK = ${MPICC}

#---------------------------------------------------------------------------

# These macros are passed to the linker to help link with MPI correctly

#---------------------------------------------------------------------------

#CMPI\_LIB = -L/usr/local/lib -lmpi

CMPI\_LIB = -L/mnt/common/shuibing/install/lib -lmpich -liosig

#---------------------------------------------------------------------------

# These macros are passed to the compiler to help find 'mpi.h'

#---------------------------------------------------------------------------

CMPI\_INC = -I/mnt/common/shuibing/install/include

#---------------------------------------------------------------------------

# Global \*compile time\* flags for C programs

#---------------------------------------------------------------------------

CFLAGS = -O3

# CFLAGS = -g

#---------------------------------------------------------------------------

# Global \*link time\* flags. Flags for increasing maximum executable

# size usually go here.

#---------------------------------------------------------------------------

CLINKFLAGS =

#---------------------------------------------------------------------------

# MPI dummy library:

#

# Uncomment if you want to use the MPI dummy library supplied by NAS instead

# of the true message-passing library. The include file redefines several of

# the above macros. It also invokes make in subdirectory MPI\_dummy. Make

# sure that no spaces or tabs precede include.

#---------------------------------------------------------------------------

# include ../config/make.dummy

#---------------------------------------------------------------------------

# Utilities C:

#

# This is the C compiler used to compile C utilities. Flags required by

# this compiler go here also; typically there are few flags required; hence

# there are no separate macros provided for such flags.

#---------------------------------------------------------------------------

CC = mpicc -g

#---------------------------------------------------------------------------

# Destination of executables, relative to subdirs of the main directory. .

#---------------------------------------------------------------------------

BINDIR = ../bin

#---------------------------------------------------------------------------

# Some machines (e.g. Crays) have 128-bit DOUBLE PRECISION numbers, which

# is twice the precision required for the NPB suite. A compiler flag

# (e.g. -dp) can usually be used to change DOUBLE PRECISION variables to

# 64 bits, but the MPI library may continue to send 128 bits. Short of

# recompiling MPI, the solution is to use MPI\_REAL to send these 64-bit

# numbers, and MPI\_COMPLEX to send their complex counterparts. Uncomment

# the following line to enable this substitution.

#

# NOTE: IF THE I/O BENCHMARK IS BEING BUILT, WE USE CONVERTFLAG TO

# SPECIFIY THE FORTRAN RECORD LENGTH. IT IS A SYSTEM-SPECIFIC VALUE.

# UNCOMMENT THE SECOND LINE AND SUBSTITUTE THE CORRECT VALUE FOR

# "length".

# IF BOTH 128-BIT DOUBLE PRECISION NUMBERS AND I/O ARE TO BE ENABLED,

# UNCOMMENT THE THIRD LINE AND SUBSTITUTE THE CORRECT VALUE FOR

# "length"

#---------------------------------------------------------------------------

# CONVERTFLAG = -DCONVERTDOUBLE

# CONVERTFLAG = -DFORTRAN\_REC\_SIZE=length

# CONVERTFLAG = -DCONVERTDOUBLE -DFORTRAN\_REC\_SIZE=length

#---------------------------------------------------------------------------

# The variable RAND controls which random number generator

# is used. It is described in detail in Doc/README.install.

# Use "randi8" unless there is a reason to use another one.

# Other allowed values are "randi8\_safe", "randdp" and "randdpvec"

#---------------------------------------------------------------------------

RAND = randi8

# The following is highly reliable but may be slow:

# RAND = randdp

（2）一直出现错误：

# apt-get install fort77 libmpich-mpd1.0-dev

（1）./configure --prefix=/mnt/common/shuibing/install --enable-fc --enable-f77 F77=gfortran FC=gfortran --with-pvfs2=/mnt/common/shuibing/install |tee config.hsb,

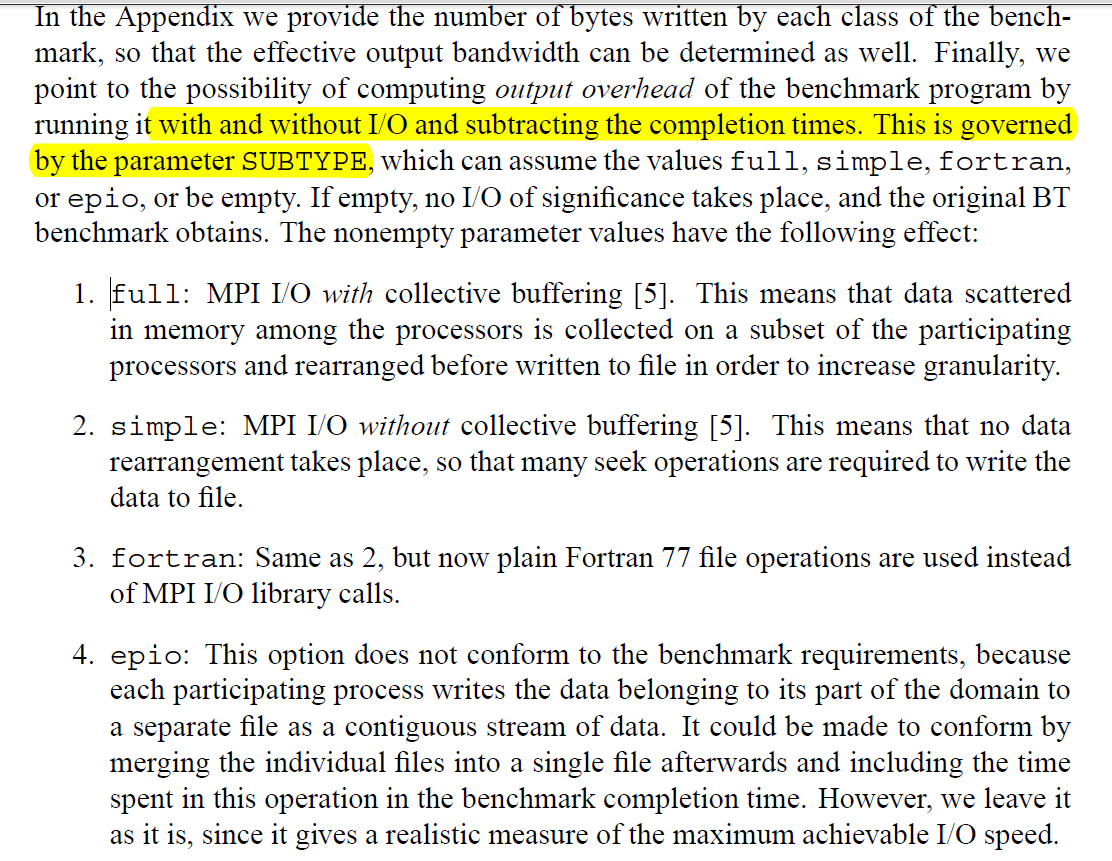
安装后出现mpif90

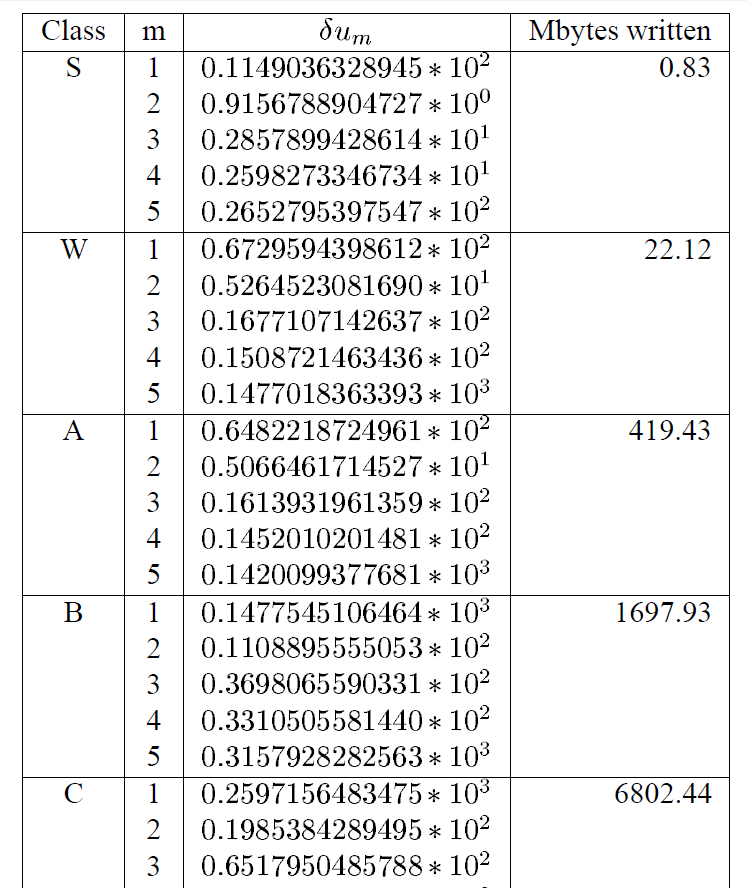
（2）cd NPB-MPI/

将MPIF77指定为mpif77,编译后没有问题

（3）运行：

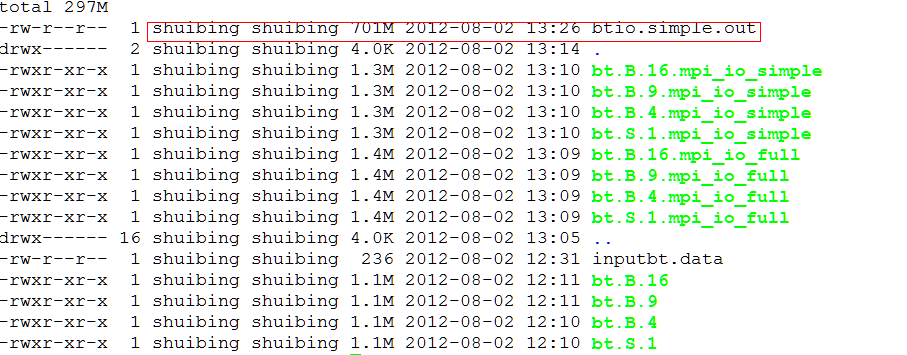
bin/...





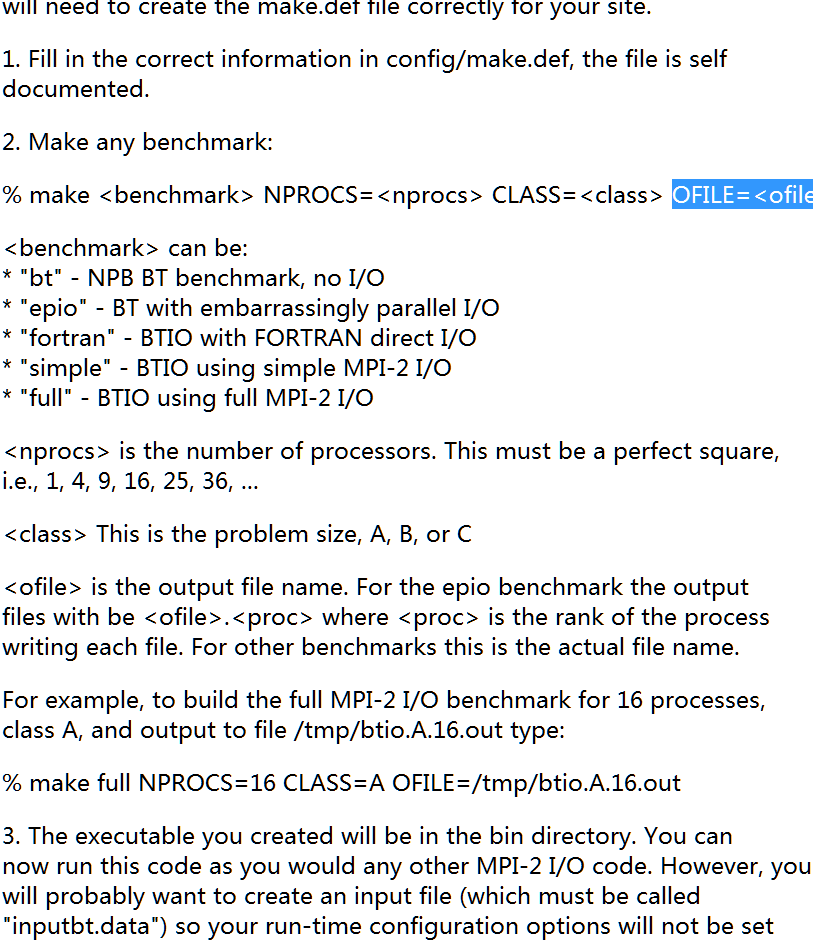
直接运行命令，文件写在当前运行的目录下面：

mpiexec -np 4 ./bt.B.4.mpi\_io\_simple



## 指定输出文件名：

## 在编译时指定：[BTIO编译和运行方法](http://www.mpi-forum.org/archives/mail/mpi-io/0337.html)



# 

# NPB3.3编译和试验方法总结

（1）mpich安装带上 romio pvfs2路径； 同时支持 FC=gfortran , F77=gfortran

./configure --prefix=/mnt/common/shuibing/install --enable-fc --enable-f77 F77=gfortran FC=gfortran --with-pvfs2=/mnt/common/shuibing/install |tee config.hsb,

（2）进入到sys/setparams.c +563: 修改相关的输出文件名路径

if (io) {

fprintf(fp, "%scharacter\*(\*) filenm\n", FINDENT);

switch (io) {

case FULL:

fprintf(fp, "%sparameter (filenm = 'btio.full.out')\n", FINDENT);

break;

case SIMPLE:

//fprintf(fp, "%sparameter (filenm = 'btio.simple.out')\n", FINDENT);

//shuibing 20120802

fprintf(fp, "%sparameter (filenm = 'pvfs2:/mnt/iocon-pvfs2/btio.simple.out')\n", FINDENT);

break;

case EPIO:

fprintf(fp, "%sparameter (filenm = 'btio.epio.out')\n", FINDENT);

break;

case FORTRAN:

fprintf(fp, "%sparameter (filenm = 'btio.fortran.out')\n", FINDENT);

fprintf(fp, "%sinteger fortran\_rec\_sz\n", FINDENT);

fprintf(fp, "%sparameter (fortran\_rec\_sz = %d)\n",

FINDENT, fortran\_rec\_size);

break;

default:

break;

}

(3)make suite

(4)cd bin, 执行可执行文件即可

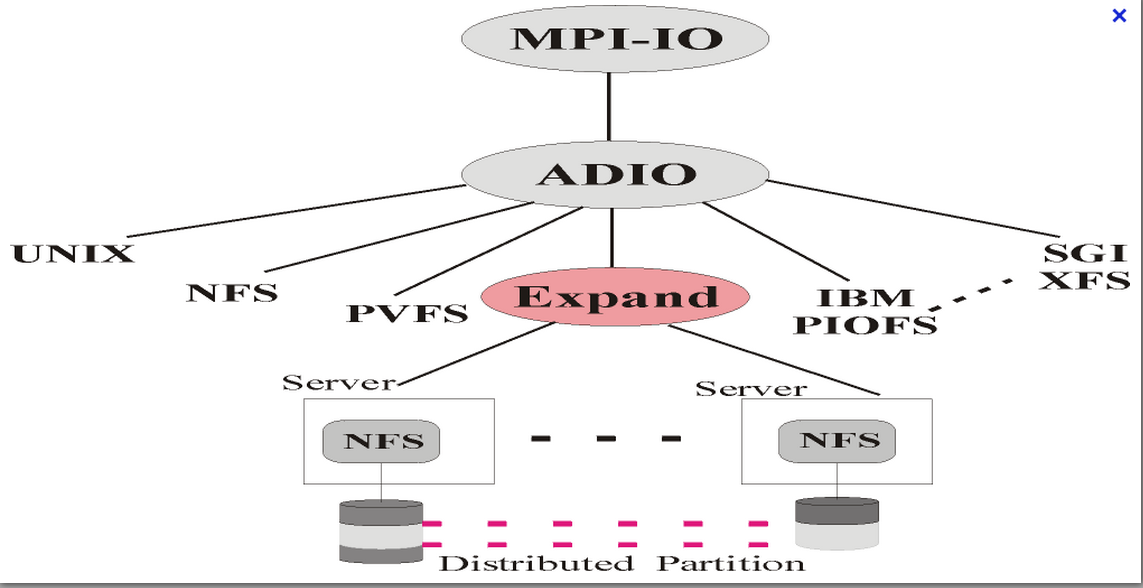
mpiexec -np 4 ./bt.B.4.mpi\_io\_simple

结果见[NPB BTIO](https://docs.google.com/document/d/16ZH-xKYDtOcQzgoQW7XoOf0eyL0vTEsdSrwQdhEAYB4/edit)

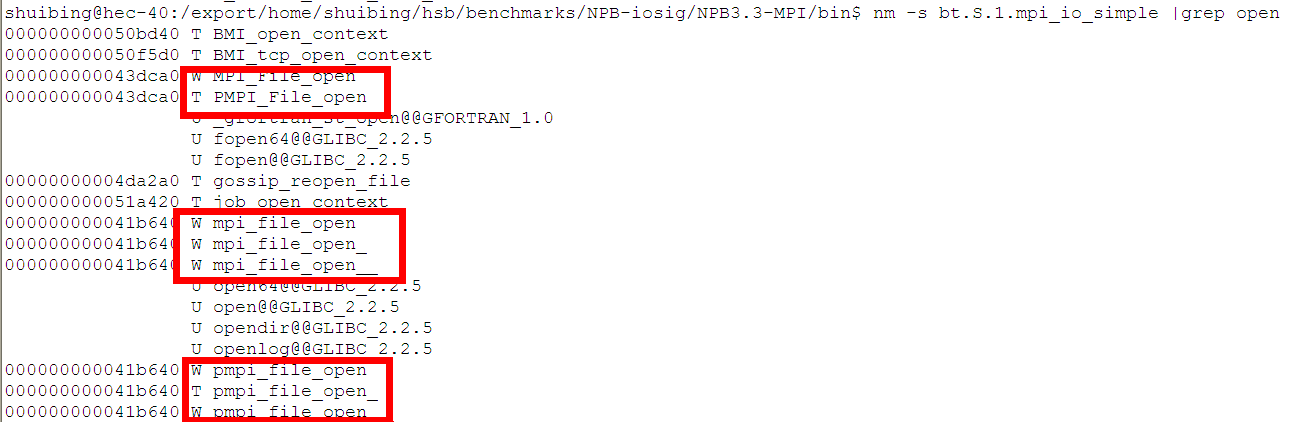
（5）结论：进程越多，性能越好

与IR论文结果相反

原因：各I/O server上内存作用很大，不是实际访问设备。

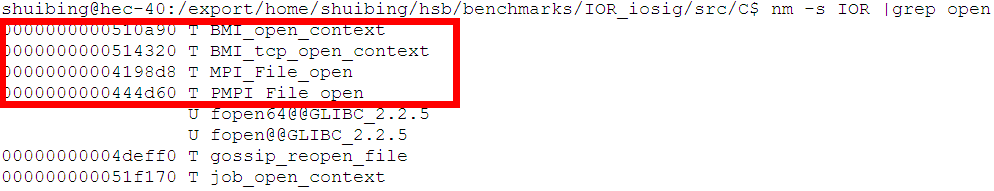


(6)NPB btio +iosig



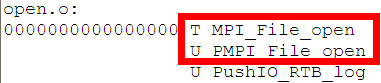
上述MPI\_FILE\_open实际指向PMPI\_File\_open,都一个地址，那么相关的时间截取跳过去了。

2.

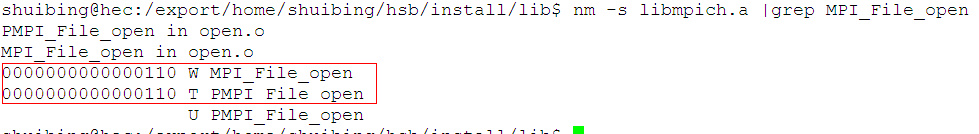


这里IOR\_iosig指向不同地址了。

3. iosig库



4.而PMPI\_File\_open应该是在libmpich.a中定义的

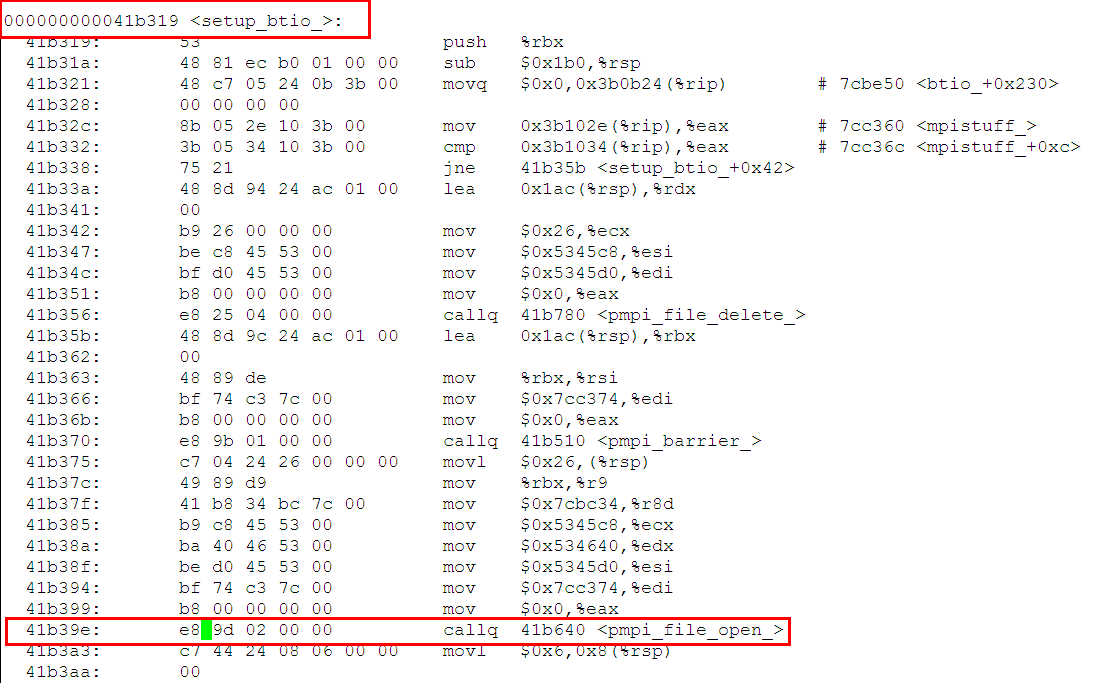


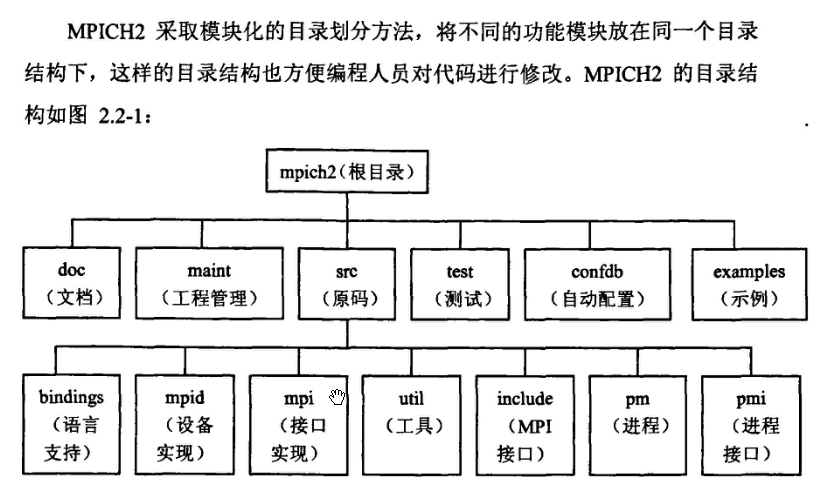
故在运行NPBbenchmark时，需要先包含iosig中的库，后包含libmpich中的库

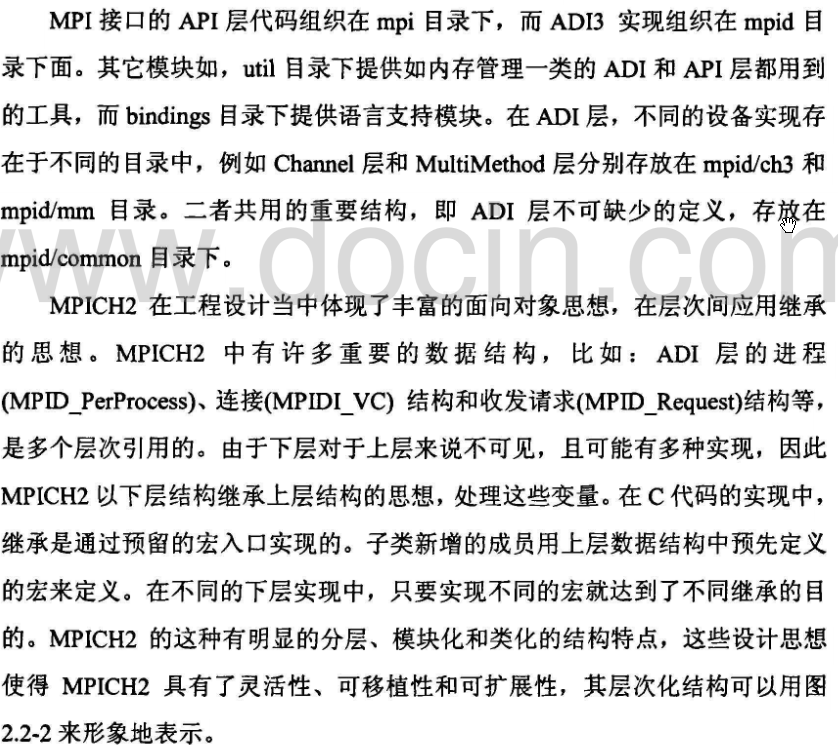
或者其不应该包含mpiioprof.h的头文件，这样函数本身不会内部去转化。

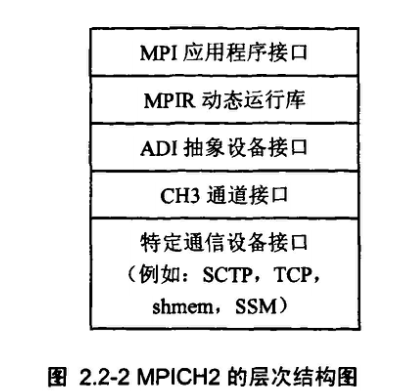
实际上btio程序运行的是：

objdump -d bin/bt.S.1.mpiio\_simple

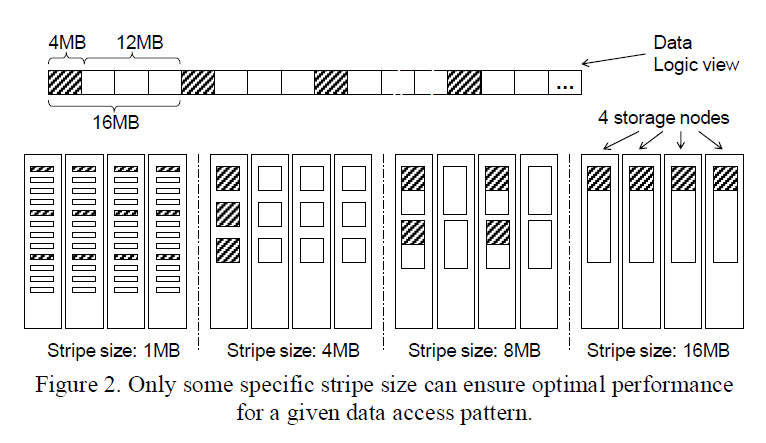


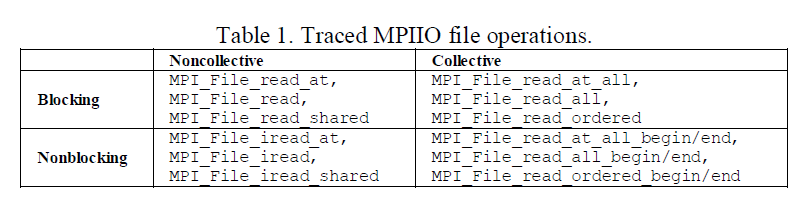






燕龙ccgrid的论文中数据访问图形：





# MPI io test

[pattern](http://institutes.lanl.gov/data/software/src/mpi-io/APexplain.php)



MPIIO ROMIO dir: mpi-io

make

实际干的事情：

（1）生成相应.o文件

CC /home/shuibing/mpich2-source/src/mpi/romio/mpi-io/mpiu\_greq.c

CC /home/shuibing/mpich2-source/src/mpi/romio/mpi-io/mpich2\_fileutil.c

CC /home/shuibing/mpich2-source/src/mpi/romio/mpi-io/register\_datarep.c

CC /home/shuibing/mpich2-source/src/mpi/romio/mpi-io/mpir-mpioinit.c

（2）将这些.o文件生成相关的库文件

ar cr /home/shuibing/mpich2-build/lib/libmpich.a close.o read.o open.o write.o set\_view.o iread.o iwrite.o seek.o get\_posn.o delete.o read\_all.o read\_at.o read\_atall.o iread\_at.o iwrite\_at.o write\_all.o get\_bytoff.o write\_at.o write\_atall.o get\_view.o get\_group.o get\_amode.o get\_extent.o fsync.o get\_atom.o set\_atom.o set\_size.o get\_size.o prealloc.o file\_f2c.o file\_c2f.o set\_info.o get\_info.o rd\_atallb.o rd\_atalle.o read\_allb.o read\_alle.o wr\_atallb.o wr\_atalle.o write\_allb.o write\_alle.o get\_posn\_sh.o iread\_sh.o read\_sh.o write\_sh.o iwrite\_sh.o seek\_sh.o read\_ord.o read\_orde.o write\_ordb.o read\_ordb.o write\_ord.o write\_orde.o mpiu\_greq.o mpich2\_fileutil.o register\_datarep.o mpir-mpioinit.o

（3）生成符号索引

ranlib /home/shuibing/mpich2-build/lib/libmpich.a

ranlib generates an index to the contents of an archive and stores it in the archive. The index lists each symbol defined by a member of an archive that is a relocatable object file

可以查看相关符号表：

nm -s /home/shuibing/mpich2-build/lib/libmpich.a

结果：  [mpiio symbol](https://docs.google.com/document/d/1U9pW80tOgLpVve6hx7i0T7WSBKP7wVk9hlwBm0ISp6M/edit)

注意：Fortran中，所有函数名不区分大小写，但是一般写成与C语言一样的形式，所以MPI\_File\_open=MPI\_FILE\_OPEN=mpi\_file\_open

