Compiling Amber8 on AMD Opteron Machines with both Intel and Pathscale Compilers

Compilation of amber on a linux platform may be challenging due to a great variety of linux distributions, compilers, system and MPI libraries. Here, I summarized all the problems during my compilations of Amber8, the latest version of a molecular dynamics simulation application on our cluster. Our cluster, with a beautiful name---pegasus, is equipped with SuSE Linux 9.0, MPICH (both gnu and Intel versions), Pathscale 2.2 release and Intel Fortran 9 EM64Tcompilers. As for the hardware, processors information is as follows:

node01:~ # cat /proc/cpuinfo
processor : 0
vendor_id : AuthenticAMD
cpu family : 15
model : 5
model name : AMD Opteron(tm) Processor 246
stepping : 8
cpu MHz : 1993.821
cache size : 1024 KB
fpu : yes
fpu_exception : yes
cpuid level : 1
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx mmxext lm 3dnowext 3dnow
bogomips : 3923.96
TLB size : 1088 4K pages
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management: ts ttp

processor : 1
vendor_id : AuthenticAMD
cpu family : 15
model : 5
model name : AMD Opteron(tm) Processor 246
stepping : 8
cpu MHz : 1993.821

cache size : 1024 KB
fpu : yes
fpu_exception : yes
cpuid level : 1
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx mmxext lm 3dnowext 3dnow
bogomips : 3981.31
TLB size : 1088 4K pages
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management: ts ttp

I have to admit that both Pathscale and Intel are very excellent compilers, but when used to compile applications, Amber8 at least, on 64 bit AMD Opteron processors, some problems will appear; in fact, in many situations, maybe what you can do is only being confused, totally. So, very careful investigation on the error message and necessary modifications of some configuration files, even of some Makefiles in the application itself, are usually required. Turn to some computer books to get some basic knowledge will be good for you.

Problems from Intel compilers will be presented to you first, and then those from Pathscale.

The Intel X86-64 compilers, the one designed for the Pentium 4 EM64T chips (the new P4's with 64 bit extensions) work very well on Opterons. I assume this is because Intel used AMD's X86-64 instruction set for their new Pentium 4's. (Please make sure you get the X86-64 version, the regular 32 bit version will work fine but you won't get 64 bit support. Also the Itanium version will not work at all on Opterons). If you misuse the Intel compiler for 32 bit processors, during the compilations the machines will tell you that your compiler should be changed to the 64bit one, so just pay a little attention on the error message and this will be settled.

Now, I give you the compilation process by Intel X86-64 compiler here step by step:

REMEMBER to set **AMBERHOME** and **MPICH_HOME** (for parallel compilations) environmental variables before you start compiling (export is used for setting environmental variables in the following examples; use setenv for csh variants; # is prompt).

#export AMBERHOME="/local/user/amber8" (bash, sh, zsh, etc)

or

#setenv AMBERHOME local/user/amber8 (csh,tcsh, etc)

#export MPICH_HOME="/opt/mpich/intel"

then

#. **/opt/intel/fce/9.0/bin/ifortvars.sh** (for bash)

or

#**source /opt/intel/fce/9.0/bin/ifortvars.sh** (for csh)

Then run this command to make sure the environment settings are correct

#which ifort

You should have the following output

#/opt/intel/fce/9.0/bin/ifort

If not, do not continue! Make sure the environment settings are correct first. If this step is ignored by you, then during the compilations, you will get some error reports, which indicate that sander binary can not find Intel Library (libimf.so) to be link, so you need to set your environment variable point to intel compiler directory here.

#cd $AMBERHOME/src

#vi leap/src/leap/Makefile

edit from :

XALEAP_LIB = ../Xraw/libXaw.a ../Wc/libWcLeap.a ../Xpm/libXpm.a \
    ../Xmu/libXmu.a -L$(XHOME)/**lib** -lXt -lXext -lSM -lICE -lX11 -lm -lpthread

to

XALEAP_LIB = ../Xraw/libXaw.a ../Wc/libWcLeap.a ../Xpm/libXpm.a \
    ../Xmu/libXmu.a -L$(XHOME)/**lib64** -lXt -lXext -lSM -lICE -lX11 -lm -lpthread

---

The Makefile for leap is originally made for 32 bit machines, now it should be adjusted to fit for our 64 bit ones, so MAKE SURE of this here, or your leap will never be installed properly.

#./configure -opteron -static ifort

Here, make sure to use the *–static* parameter, to find more details, just issue command #./configure –help in $AMBERHOME/src directory.

#make serial (It will take you several minutes, just be patient)

#cd $AMBERHOME/test

#make test.leap (or test.sander **Note**: this sander is a single-threaded one)

Then back to the src directory

#cd $AMBERHOME/src

and

#make clean

#./configure -opteron -mpich -static ifort

#make parallel

Most of the problems appeared in this step during my compilations, so go through each parameter before this step, including the **MPICH_HOME** environment variable, the **#. /opt/intel/fce/9.0/bin/ifortvars.sh** operation and the **–static** parameter things.

Actually, when you go to /opt/mpich directory, you will see that there are two folders there; one is gnu and the other intel. And I have tried to compile Amber8 with both of them, when I used the gnu one, during my compilations, I got the following message:

```
ifort   -FR  -o sander trace.o lmod.o decomp.o icosasurf.o egb.o
findmask.o pb_force.o sander.o cshf.o noecalc.o noeread.o caldis.o calrate.o dinten.o
drates.o indexn.o kmat.o pearsn.o plane.o remarc.o nmrcal.o nmrred.o restal.o getnat.o
nmrnrg.o modwt.o disnrg.o angnrg.o tornrg.o nmrprt.o nmrgrp.o nmrcms.o nmrcmf.o
impnum.o nmrsht.o at2res.o chklin.o opnmrg.o printe.o runmin.o ndvprt.o force.o
rdparm.o mdread.o locmem.o runmd.o getcor.o r6ave.o r6drv.o aveint.o degcnt.o corf.o
threeb.o tripl.o nmrrad.o decnvh.o fastwt.o echoin.o parallel.o jnrg.o shake.o ene.o
mdwrit.o minrit.o set.o setmm.o dynlib.o mdfil.o nmlsrc.o ew_force.o ew_setup.o
ew_box.o ew_bspline.o ew_fft.o ew_direct.o ew_recip.o pcshift.o align.o rstack.o
istack.o rfree.o rgroup.o random.o lsqfit.o amopen.o debug.o ew_recip_reg.o
ew_handle_dips.o ew_dipole_recip.o mexit.o new_time.o extra_pts.o thermo_int.o
matinv.o assert.o mmtsb.o mmtsb_client.o erfcfun.o veclib.o mdm.o pb_init.o
constantph.o prn_dipoles.o  \
        ../lmod/lmod.a -lpbs -lnsl -L/opt/lammpi-7.1.1/lib -llammpio -llamf77mpi -lmpi -
llam -laio -laio -lutil -ldl  \
        ../lapack/lapack.a ../blas/blas.a ../lib/nxtsec.o ../lib/sys.a
egb.o: In function `genborn_mp_egb_':
```

egb.o(.text+0x210b): undefined reference to `mpi_allreduce_'
egb.o(.text+0x30ff): undefined reference to `mpi_allreduce_'
sander.o: In function `MAIN__':
sander.o(.text+0x2d): undefined reference to `mpi_init_'
sander.o(.text+0x56): undefined reference to `mpi_comm_rank_'
sander.o(.text+0x77): undefined reference to `mpi_comm_size_'
sander.o(.text+0x90): undefined reference to `mpi_barrier_'
sander.o(.text+0x104): undefined reference to `mpi_bcast_'
...
sander.o(.text+0x4b2): undefined reference to `mpi_barrier_'
sander.o(.text+0x518): undefined reference to `mpi_comm_split_'
sander.o(.text+0x664): undefined reference to `mpi_comm_size_'
sander.o(.text+0x685): undefined reference to `mpi_comm_rank_'
...
(much more of this)

Do **NOT** be confused just like I was, that is only because that you are not building Amber with the exact same compiler as you are building MPICH or LAM with. Actually, our administrator maybe compiled MPICH with Pathscale before, so error was showed to me on the screen. Do not twist the problem as I did, just change to the Intel compiler, then all of this will disappear. So you should set the MPICH_HOME and DO_PARALLEL (below) to /opt/mpich/intel….

#cd $AMBERHOME/test

#export DO_PARALLEL="/opt/mpich/intel/bin/mpirun -np 4" (This variable is demanded by the installation manual)

#export P4_GLOBMEMSIZE=256000000 (Will talk about it latterly)

#make test.sander

Be sure to pass this step smoothly before you submit your own long-time jobs to the cluster. When I executed the make test.* commands, some trouble just emerged.

1. Before running parallel (MPI) amber8 tests, set **DO_PARALLEL** environmental variable. It is a good idea to set the full path to your mpirun to prevent problems/confusion locating mpirun executable, e.g.:

#export DO_PARALLEL="/usr/local/mpich/bin/mpirun -np 2 "

#export P4_GLOBMEMSIZE=256000000

Also make sure you can 'rlogin' to the nodes where you are testing without a password and that your -machinefile mfile is set properly. If during "make test" you get an error message similar to:

cd cytosine; ./Run.cytosine
Unit 5 Error on OPEN: in.md
[0] MPI Abort by user Aborting program
[0] Aborting program
p0_4302: p4_error: : 1
./Run.cytosine: Program error

It usually means that you compiled parallel version of sander but forgot to set DO_PARALLEL before running the tests (in other words, you are running MPI version of sander without mpirun).

2. However, error messages like the following are not due to the **DO_PARALLEL** things:

Bad : modifier in $ (/).
Warning: Permanently added 'node00' (RSA) to the list of known hosts.
rm_24403: p4_error: semget failed for setnum: 0
p0_7245: (0.082031) net_recv failed for fd = 5
p0_7245: p4_error: net_recv read, errno = : 104
Killed by signal 2;

They are induced (most likely) by your improper job-killing operations when you are testing your programs, so using up all available IPCs. Should use cleanipcs to fix it. The basic of what is going on is that MPICH is trying to allocate shared memory in large blocks for parallel execution. If you need more memory than it can handle, it tries to allocate too many SHMIDS and fails. When you up the GLOBMEMSIZE, each chunk is larger, and thus you need less IDS. If MPICH dies in your testing, though, or you terminate it, the shared memory is not correctly cleaned up. Fortunately, cleanipcs command can free the shared memory segments and semaphores. It is normally not needed, but if MPI programs exit abnormally, the program may be unable to free the System V IPCs that it held (this is a feature of System V IPCs). In that case, cleanipcs can be used to recover the IPCs that are no longer needed. Note that this command releases all IPCs held by the user. This is the correct behavior for most users, but may cause problems for users of other programs or systems that rely on the persistence of System V IPCs.

If you're doing all of this and it fails, it could be a problem with the program you're running or the data you're passing to it. So change a direction to find things out, please.

As for our machines, you have to clean up the rubbish from one node to the other. Be sure of this, or you will still have the same error report. When you login to the pegasus cluster, you are operating everything on the administrative node, so when issue the command cleanipcs, it cleans the administrative node for you only. I don’t know how to

clean up all the nodes with only one script or command so far ☹, so I use these to achieve it: (supposing to start our rubbish-cleaning work from pegasus)

 # ssh node00 (node01, 02……09…..)

#cleanipcs

Then go back to pegasus,

#ssh pegasus

3. If you got this:

p0_15783: p4_error: exceeding max num of P4_MAX_SYSV_SHMIDS: 256

or

p0_4393: p4_error: interrupt SIGSEGV: 11

or something like these:

p8_12489: (6.243597) xx_shmalloc: returning NULL; requested 3801600 bytes
p8_12489: (6.243731) p4_shmalloc returning NULL; request = 3801600 bytes
You can increase the amount of memory by setting the environment variable
P4_GLOBMEMSIZE (in bytes); the current size is 4194304
p8_12489: p4_error: alloc_p4_msg failed: 0

Then, the **P4_GLOBMEMSIZE** variable must be set to much larger than what the program is requesting. The current default size is 4194304. You should edit/create your ~/.tcshrc (or .bashrc or others) to add, for example,

#export P4_GLOBMEMSIZE=256000000 (bash, sh, zsh, etc.)
(For our system 256MB is large enough, anyway, you can put this into your batch scripts which will be listed below)

or

#setenv P4_GLOBMEMSIZE 256000000 (csh,tcsh)
then

#. ~/.bashrc

or

#source ~/.tcshrc

Pathscale part:

So much for Intel compilers trouble, now I give you some information on Pathscale.
Before the latest version of Pathscale v2.2beta (or v2.1.99) comes, the old version has some bugs, so you have to fix them firstly. However, after the repairs, when you submit your jobs to the machines, running error likes this:

 # Warning: Permanently added 'node00' (RSA) to the list of known hosts.
Warning: Permanently added 'node01' (RSA) to the list of known hosts.
Warning: Permanently added 'node02' (RSA) to the list of known hosts.

lib-4014 : UNRECOVERABLE library error
The requested action requires that unit 6 be connected to a file.

Encountered during an I/O operation on unit 6
Fortran unit 6 is not connected

lib-4014 : UNRECOVERABLE library error
The requested action requires that unit 6 be connected to a file.

Encountered during an I/O operation on unit 6
Fortran unit 6 is not connected

lib-4014 : UNRECOVERABLE library error
The requested action requires that unit 6 be connected to a file.

Encountered during an I/O operation on unit 6
Fortran unit 6 is not connected

was reported. Actually, according to answer from Pathscale Company, this is a known problem (bug #6433) which occurs when running AMBER8 with MPICH on multiple nodes, and the problem has been fixed in Release 2.2. Beta-2.2. So I skipped the bugfix things here, just assuming that you have got the beta2.2 version Pathscale.

The compilation procedure is similar with that of Intel compiler's. But when edit the $AMBERHOME/src/leap/src/leap/Makefile, the modification is somewhat different from the former one, just instead the L114:

../Xmu/libXmu.a -L**$(XHOME)/lib** -lXt -lXext -lSM -lICE -lX11 -lm –lpthread

with

../Xmu/libXmu.a -L**$(XLIBS)** -lXt -lXext -lSM -lICE -lX11 -lm –lpthread

And be sure to get the new configure file from Pathscale Company and put it into **$AMBERHOME/src** directory before you start your compilations. When compiling

Amber8 with Pathscale compiler, I used **gnu** MPICH. 'Coz this MPICH has been compiled with Pathscale compiler and I have explained why before.

Then, steps are:

#export AMBERHOME="/local/user/amber8" (bash, sh, zsh, etc)

#export MPICH_HOME="/opt/mpich/gnu"

#cd $AMBERHOME/src

#./configure -opteron -static pathscale

#make serial

#cd $AMBERHOME/test

#make test.*

#cd $AMBERHOME/src

#make clean

#./configure -opteron -mpich -static pathscale

#make parallel

OK! Till now, we can say everything is ok for the compilation and test steps. What we should do next is to submit our jobs to queues.

Our system is managing by the SUN Grid Engine (SGE) scheduler. You should use SGE to submit all your jobs. Running your program interactively should be limited only for developing, testing, and debugging purposes.

Here, just a very simple sample of the submission procedure, if you want to know more, just turn to www.google.com. I skipped the serial job submission part; only present you the batch ones. Batch jobs are submitted to SGE via scripts. Here is an example of a serial job script, **zb.sh**. It executes the sander command.

#!/bin/sh

cd $HOME/submit

#$ -cwd
#$ -j y
#$ -S /bin/bash

export all environment variables to SGE
#$ -V
#$ -v LD_LIBRARY_PATH=/opt/intel/fce/9.0/lib:$LD_LIBRARY_PATH

 # command
/opt/mpich/intel/bin/mpirun -np $NSLOTS -machinefile /gpfs/home/nnizb/submit/machines $HOME/amber8/exe/sander -O -i rosnpt.in -p ros.prmtop -c rosvmd.rst -o rosnpt00.out -r rosnpt00.rst -x rosnpt00.mdcrd < /dev/null

Note: Entries which start with #$ will be treated as SGE options.

- -cwd means to execute the job for the current working directory.
- -j y means to merge the standard error stream into the standard output stream instead of having to separate error and output streams. (Here, if you didn't choose these two parameters, you will get four output files, like: zb.sh.eJOBID, zb.sh.oJOBID, zb.sh.peJOBID, zb.sh.poJOBID; if you chose them, then output files are: zb.sh.oJOBID and zb.sh.poJOBID)
- -S /bin/bash specifies the interpreting shell for this job to be the Bash shell.
- ## line is an explanation line.
- -V export all environment variables to SGE.
- -v line ensure that the dynamic libraries used by locally installed packages cvan be found. You should leave it unchanged.
- The last line is the most important one in this simple script; the call to mpicun itself. /opt/mpich/intel/bin/mpirun is the location of the command mpirun; $NSLOTS is the number of slots for this MPI job, which corresponds to the value given by the user as the second argument to the –pe option(you will meet it soon). The $TMPDIR is a temporary directory which will contain a file titled *machines* , itself containing an automatically-generated list of nodes on which the MPI job will be run. The temporary directory and its contents will be automatically removed upon completion of the MPI job. Then sander comes.

After you prepared your batch script, then just do this:

#qsub -pe mpich 4 zb.sh

Do you want to monitor your jobs? Ok, two simple commands:
To monitor your jobs:
#qstat (you can issue it with or without options, but –f and –ext are too useful ones)

Want to kill your jobs? Just do this:
#qdel JOBID (JOBID can be obtained from qstat)