Debugging on Ra and Mio

Timothy H. Kaiser, Ph.D.

Revised January 2010 for:
OpneMPI 1.4.x
ddt 2.6

Available Debuggers

- Portland Group
  - pgdb
  - X11- GUI
- Intel
  - idb
  - Command line only

Available Debuggers

- gdb
- /usr/bin/gdb
  - Command line only
- Allinea ddt
  - Currently on Mio
  - X11- GUI
  - /opt/ddt/bin/ddt

Not a big fan

- End up debugging the debugger
- Steep learning curve
- Can be misleading
- Difficult for large processor count and the problem might only show up there
- My favorite debuggers are
  - printf
  - write

However...

- I recently used ddt to find a problem for which printf did not work. It might have taken me weeks.
- Print statements might make the problem go away
- Debuggers are useful for learning a program that you have never seen

Running GUI based debuggers

- Compile the program with debugging enabled
- Launch an interactive session
- Running

1. Launch the debugger along with MPI program
2. Launch the MPI program from within the debugger
3. Connect to a running MPI program

Compiling

- Need to use a compiler build with debugger support
  - /opt/lib/openmpi/1.4.2/intel/11.1db/bin/mpicc
- Fortran
  - mpif77 -g stf_03.f90 -o stf_03.g
  - mpif90 -g stf_03.f90 -o stf_03.g
- C
  - mpicc -g stc_03.c -o stc_03.g

Launching Debug Session

- Start two Xterm windows
  - ssh -Y ra in both
- In one window launch an interactive session:
  - msub -q INTERACTIVE -I -V -l nodes=1

```
[tkaiser@ra ~]$ qsub -q INTERACTIVE -I -V -l nodes=1
qsub: waiting for job 8760.ra.mines.edu to start
qsub: job 8760.ra.mines.edu ready

[tkaiser@compute-3-1 ~]$
```

- In second window:
  - ssh -Y compute-3-1

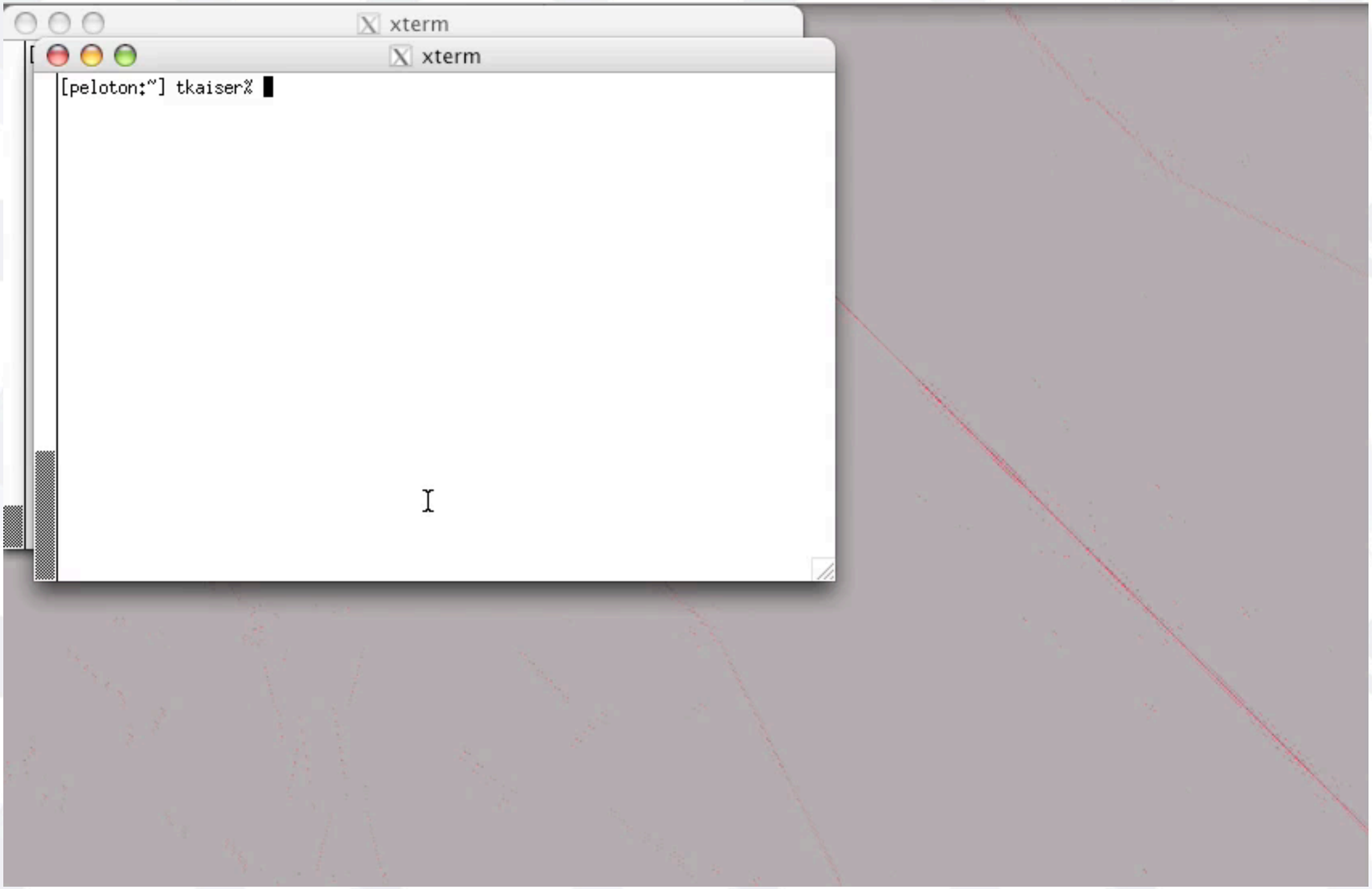
Launching Portland Group Debugger

- In the second window
  - cd to the directory containing your program
    - Launch pgdbg giving full path to mpiexec and your program
      - May want to use “which mpiexec”

```
pgdbg -mpi:/lustre/home/apps/mpi/openmpia/bin/mpiexec
      -np 4 /lustre/home/tkaiser/stommel/stf_03.g
```

Things to show you

- Launch Process
- Setting break points
- Starting the application
- Using the commands window
  - Non-local variables
  - Changing proc
  - Changing values
  - Halting while running


xterm
xterm
[peloton:~] tkaiser%

Portland Group Notes

- Does not work with ifort/OpenMPI
- Have not been able to get standard input using OpenMPI
- Might need to click a few times to start it
- Use the command window
  - To see non-local variables
  - To change variables

ddt

MPI versions for debugging

- Have tested on Mio with:
  - Intel 11
  - OpenMPI 1.4.2
- Should work with:
  - Intel version 12.x
  - Portland Group 11.x

These versions are subject to change

Allinea DDT debugger

- Initial setup with ssh -Y is the same
- An initial setup is done the first time you run
- Works with both Portland Group and Intel Fortran
- Better support for Fortran modules
- Syntax highlighting

Settings

Edit your .bashrc file and add the following

```
###  ddt settings  ###
export PATH=/opt/ddt/bin:$PATH
export DMALLOCPATH=/opt/ddt
export DMALLOC
export D_LIBRARY_PATH=${DMALLOCPATH/lib/64}:$LD_LIBRARY_PATH
```

Debug Compile with memory trace

```
mpicc -g \
stc_06.c \
$DMALLOCPATH/lib/64/libdmalloc.a  -o \
stc_06.g
```

Here we link to the debug memory library. This is required if you want to track memory usage in ddt.

Note it must library be last on the list.

Fortran Compile with memory trace

```
#include <stdlib.h>
dummy_(){
 char *x;
 x=(char*)malloc(1);
 free(x);
}
```

```
icc dummy.c -c -o dummy.o

mpif90 -g \
stf_03.f90 \
dummy.o \
$DMALLOCPATH/lib/64/libdmalloc.a \
-o stf_03.g
```

Here we link to the debug memory library. This is required if you want to track memory usage in ddt.
Note it must library be last on the list.

Initial ddt setup

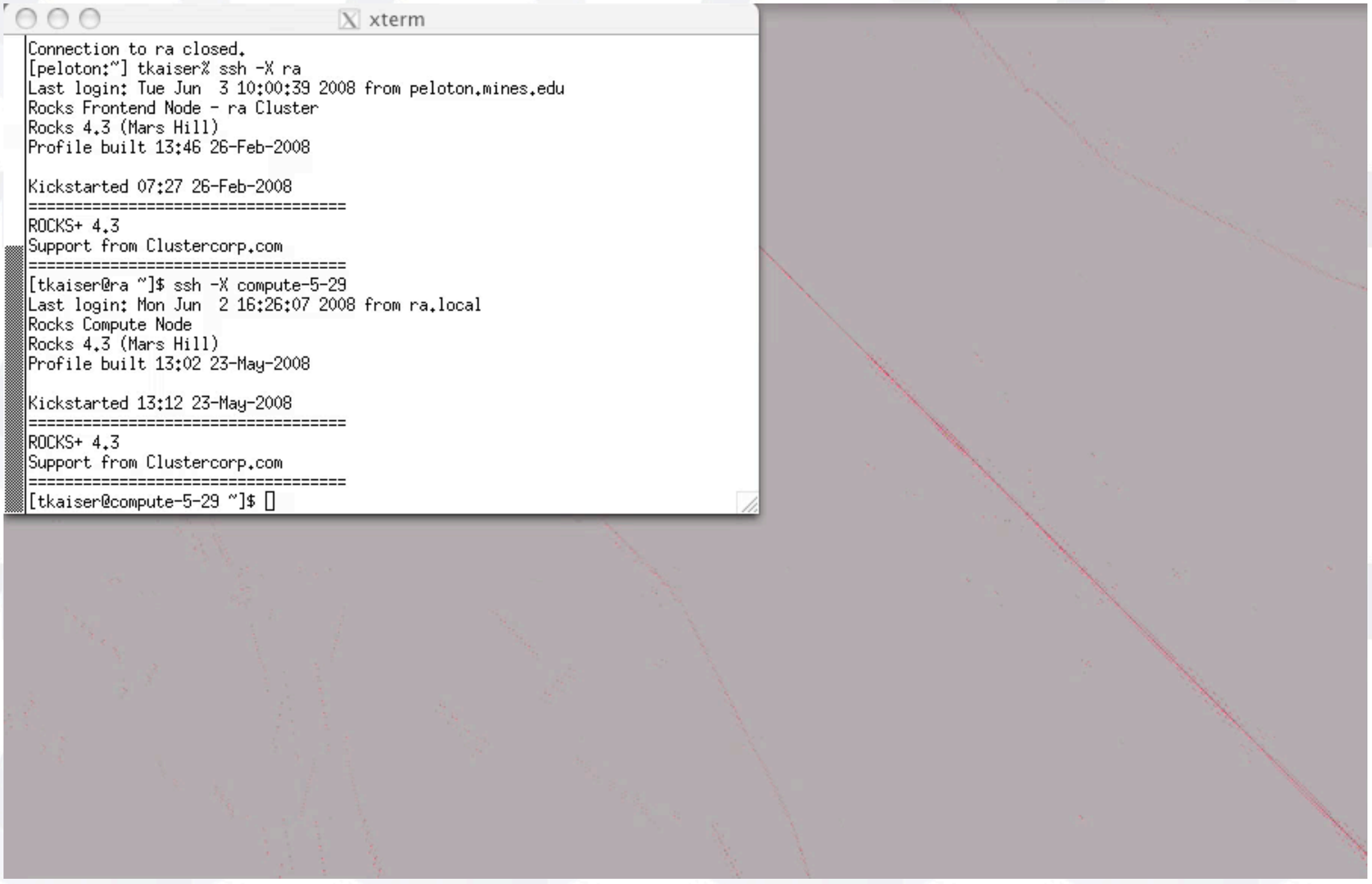
- Run first time, creates a directory ~/.ddt
- type ddt
- Choose a MPI version (OpenMPI)
- Choose a list of nodes (Default)
  - Note location of this file
  - Need to change this list to connect to running process
- Wait a few seconds

xterm

```
Connection to ra closed.
[peloton:~] tkaiser% ssh -X ra
Last login: Tue Jun  3 10:00:39 2008 from peloton.mines.edu
Rocks Frontend Node - ra Cluster
Rocks 4.3 (Mars Hill)
Profile built 13:46 26-Feb-2008

Kickstarted 07:27 26-Feb-2008
================================
ROCKS+ 4.3
Support from Clustercorp.com
================================
[tkaiser@ra ~]$ ssh -X compute-5-29
Last login: Mon Jun  2 16:26:07 2008 from ra.local
Rocks Compute Node
Rocks 4.3 (Mars Hill)
Profile built 13:02 23-May-2008

Kickstarted 13:12 23-May-2008
================================
ROCKS+ 4.3
Support from Clustercorp.com
================================
[tkaiser@compute-5-29 ~]$
```



Running ddt

- Select “Run and Debug a Program”
- Select the program that you will run
- Set number of processes
- Most likely Set threads to “off”
- Click Run
- Details to follow...

To show you...

- Routine required for correct stdio with Portland Group compiler
- Setting stdin
- Module support
- Changing values
- Locals / Current Line

Note about the movie:

This following movie show ddt being started on a compute node.

Our current license requires that ddt be started on the head node and then you connect to processes running on the compute node.

You can also have ddt submit a batch script for you and then connect automatically when the job starts.

Slides and movies showing how to do this follow.

xterm

```
[tkaiser@compute-5-29 ~/stommel]$ mpif90 -g -L/lustre/home/apps/gdb-6.8/lib64 -l
iberty stf_03.f90 -o stf_03.g
```

Option: Let ddt submit a batch job

- Your run script becomes a template which ddt fills in the arguments at submit time
- Tell ddt the particulars
  - Program
  - Input
  - # processors <= 16
- ddt will watch the queue for your job to start and then connect

Let ddt submit a batch job

- Change your run line to run ddt with your program as an argument

```
mpiexec -n 8 stf_03.g < st.in
```

becomes

```
mpiexec  -n NUM_PROCS_TAG DDTPATH_TAG/bin/ddt-debugger
DDT_DEBUGGER_ARGUMENTS_TAG PROGRAM_ARGUMENTS_TAG
```

- Add (Not required but useful for attaching to already running jobs)

```
sort -u $PBS_NODEFILE  > mynodes.$PBS_JOBID
cp mynodes.$PBS_JOBID ~/.ddt/nodes
```

A simple script

```
#!/bin/csh
#PBS -l nodes=1:ppn=8
#PBS -l walltime=00:10:00
#PBS -N testIO
#PBS -o stdout
#PBS -e stderr
#PBS -r n
#PBS -V
#-----------------------------------------------
cd $PBS_O_WORKDIR

#save a nicely sorted list of nodes
sort -u $PBS_NODEFILE  > mynodes.$PBS_JOBID
cp mynodes.$PBS_JOBID ~/.ddt/nodes

#for openmpi
#mpiexec -n 8 stf_03.g < st.in

#Old style run line syntax
#DDTPATH_TAG/bin/ddt-client DDT_DEBUGGER_ARGUMENTS_TAG mpiexec -np \
#NUM_PROCS_TAG EXTRA_MPI_ARGUMENTS_TAG PROGRAM_TAG  \
#PROGRAM_ARGUMENTS_TAG

#current run line syntax
MPIRUN_TAG AUTO_MPI_ARGUMENTS_TAG EXTRA_MPI_ARGUMENTS_TAG \
DDTPATH_TAG/bin/ddt-client DDT_DEBUGGER_ARGUMENTS_TAG PROGRAM_TAG \
PROGRAM_ARGUMENTS_TAG
```

Note this line is commented out.

This one is alive

A simple script

Save file as pbsfile.qtf

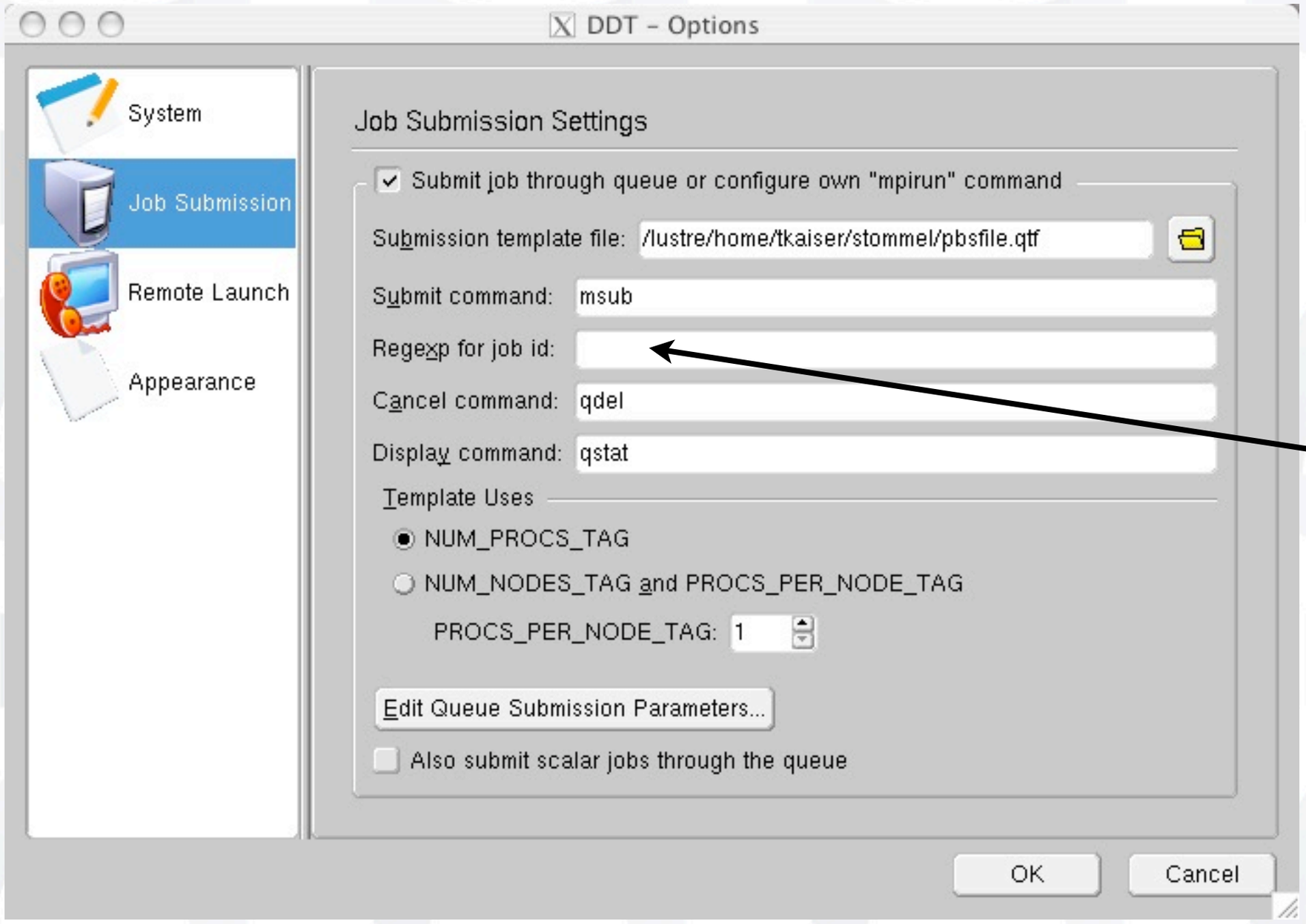
```
#!/bin/bash
#PBS -l nodes=1:ppn=8
#PBS -l walltime=00:10:00
#PBS -N testIO
#PBS -o stdout
#PBS -A myaccount
#PBS -e stderr
#PBS -r n
#PBS -V
#-------------------------------------------
cd $PBS_O_WORKDIR

#save a nicely sorted list of nodes
sort -u $PBS_NODEFILE  > mynodes.$PBS_JOBID
cp mynodes.$PBS_JOBID ~/.ddt/nodes

#for openmpi and ddt
MPIRUN_TAG AUTO_MPI_ARGUMENTS_TAG EXTRA_MPI_ARGUMENTS_TAG DDTPATH_TAG/
bin/ddt-debugger DDT_DEBUGGER_ARGUMENTS_TAG PROGRAM_TAG
PROGRAM_ARGUMENTS_TAG
```

This might change as Libraries and ddt get updated

Under Session - Options

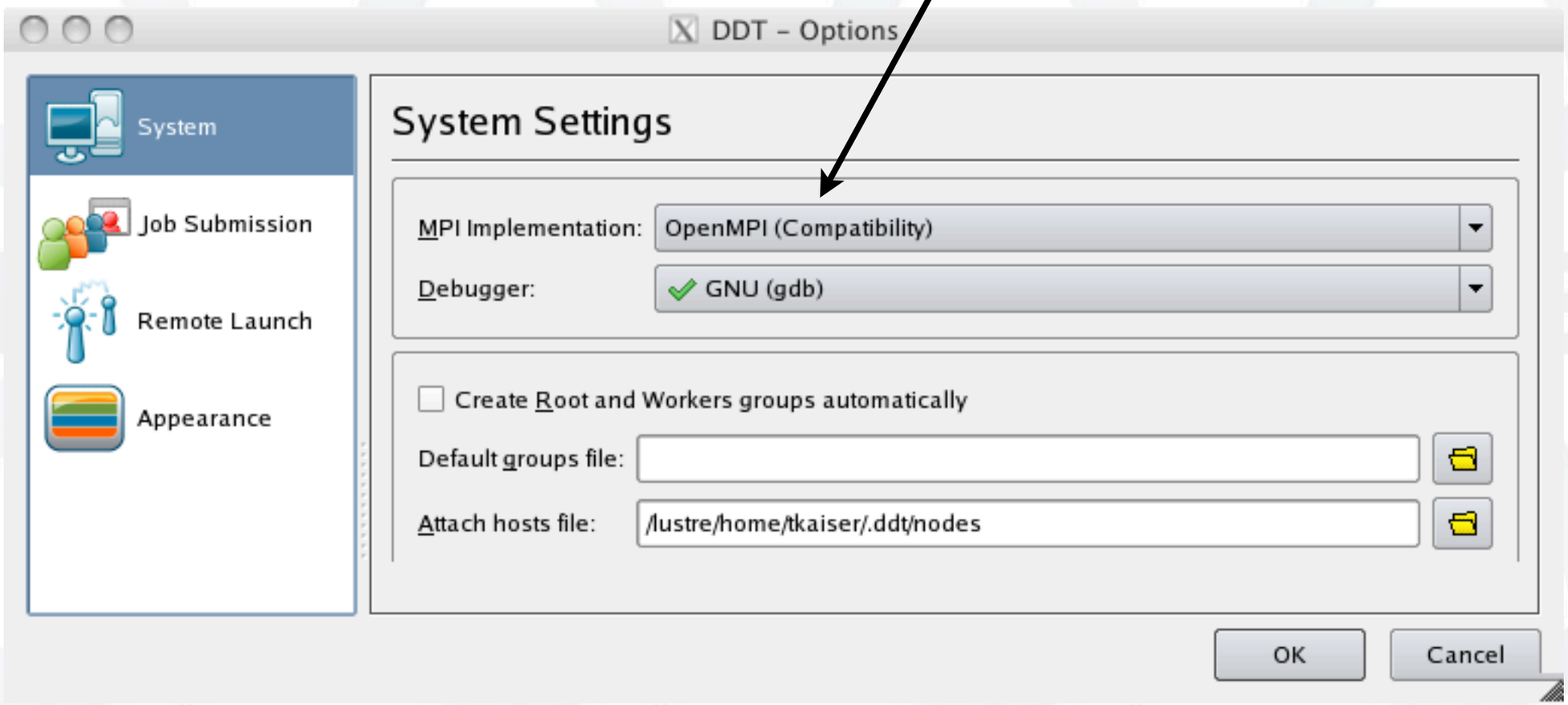


Note this is blank

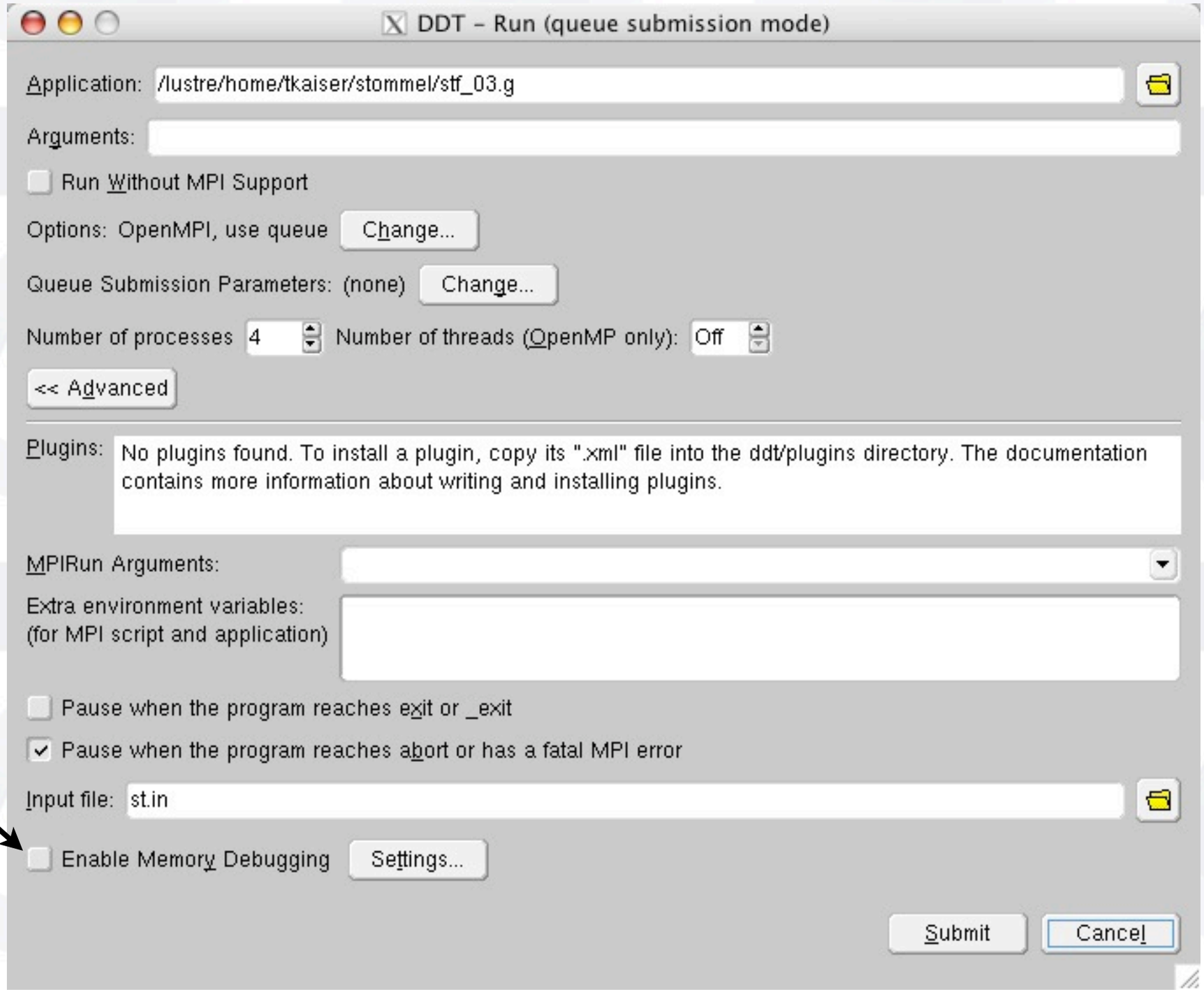
MPI Implementation

Go to:
Session
Options
System

OpenMPI (Compatibility)



Finally select Session - New Session - Run



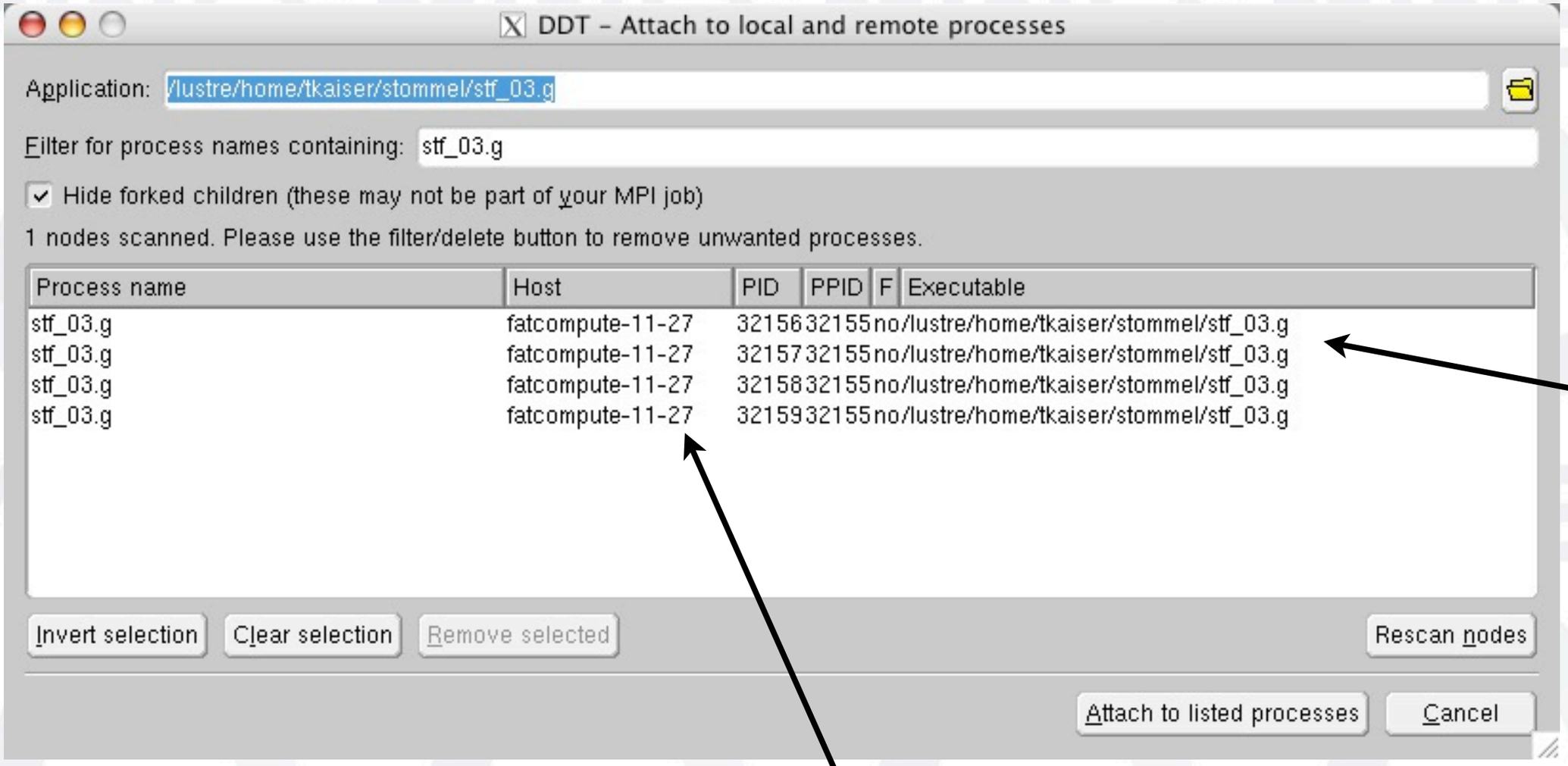
Also select this if your built with memory debugging support

Let ddt submit the job for you


xterm
[tkaiser@ra ~/stommel]$

To Attach to a Running Process

Session - New Session - Attach



List should pop up

Nodes need to be in ~/.ddt/nodes

Attaching to a batch job

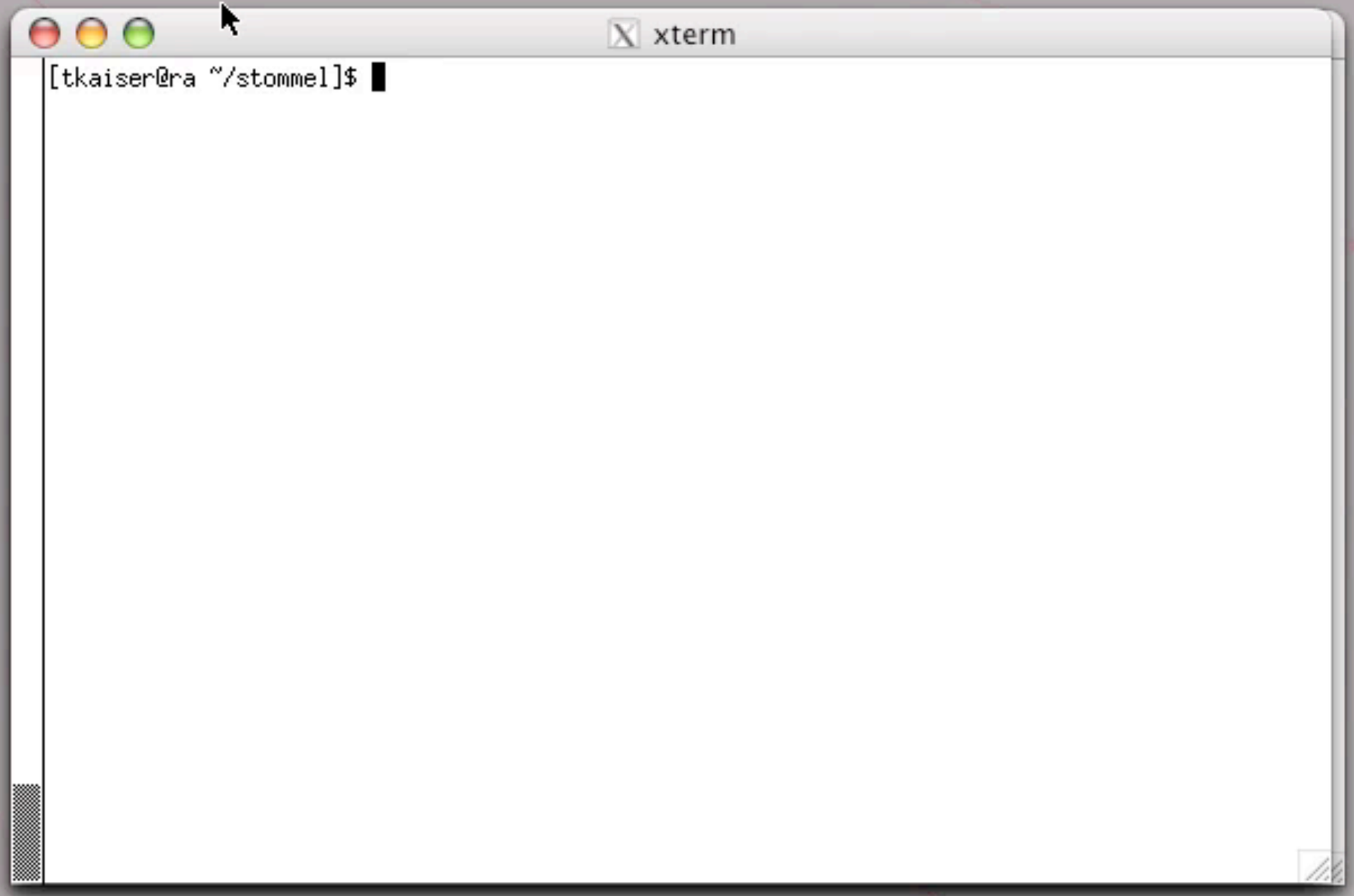
- Key here is that ddt needs to know where your job is running
- Add the following two lines to your script

```
sort -u $PBS_NODEFILE  > mynodes.$PBS_JOBID

cp mynodes.$PBS_JOBID ~/.ddt/nodes
```

- ddt will look in ~/.ddt/nodes for nodes to search

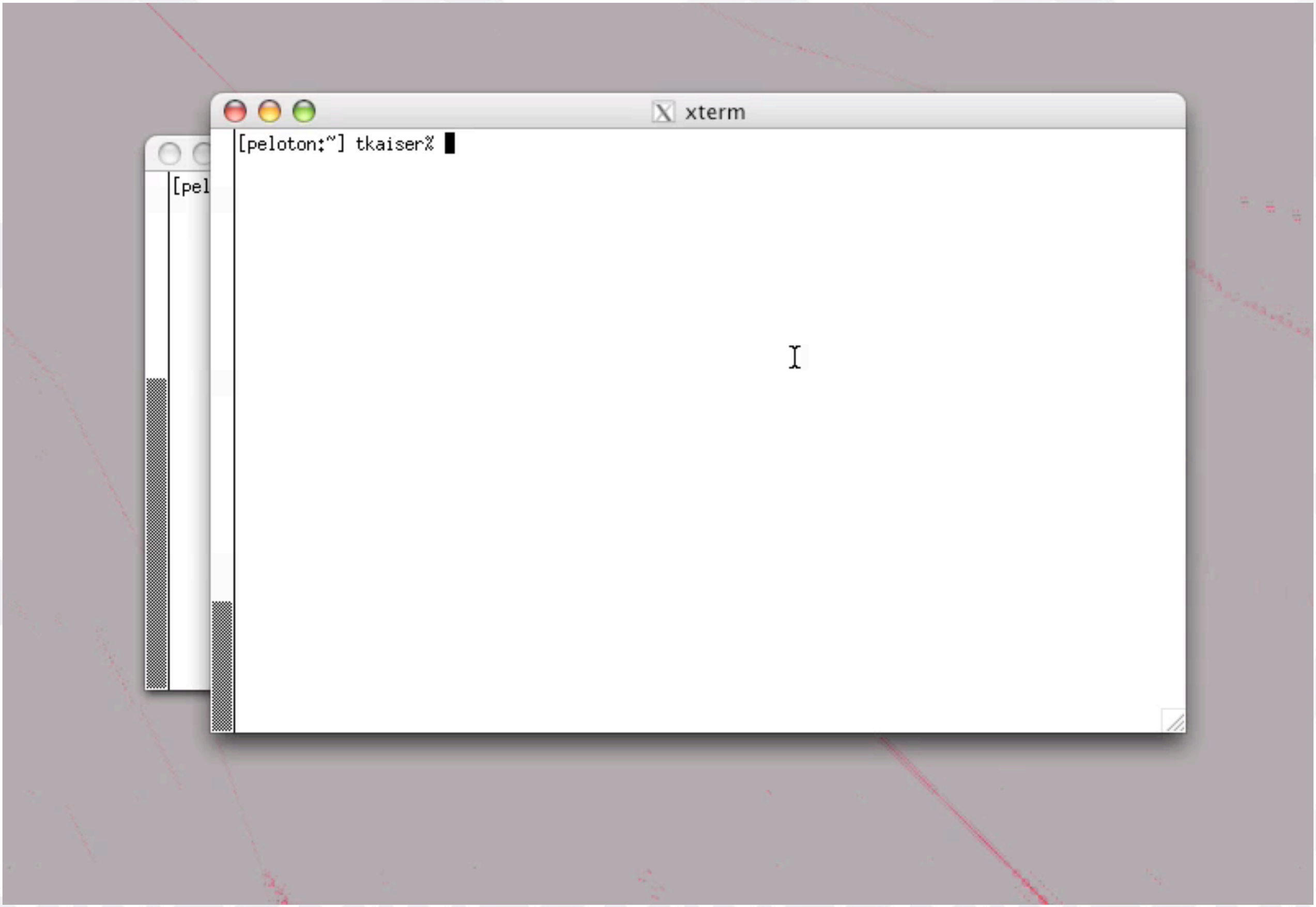
Attaching to a batch job


xterm
[tkaiser@ra ~/stommel]$

Attaching to a interactive job

- Key here is that ddt needs to know where your job is running
  - ddt will look in ~/.ddt/nodes for nodes to search
  - You may need to manually edit this file

Attaching to an interactive job


xterm
[peloton:~] tkaiser%

Let's do it for real

- Set up environment
- Cut and paste "stuff" from the slides
  - Compile a Fortran program
  - Create the ddt batch file
- Do the initial setup
- Submit it
- Show memory usage

A Final Note

Learning to run a debugger is tough.
I have found that actually getting the debugger started is a difficult part of the process.

We should really have a full day(s) on debugging and related topics