1 Invoking DDT

Compile your program with debug enabled; setting a low optimization level can also be a good idea: `mpicc -O0 -g yourprogram`. (Higher optimization levels work, but the compiler can 'optimize away' some variables and code, which may confuse the novice.)

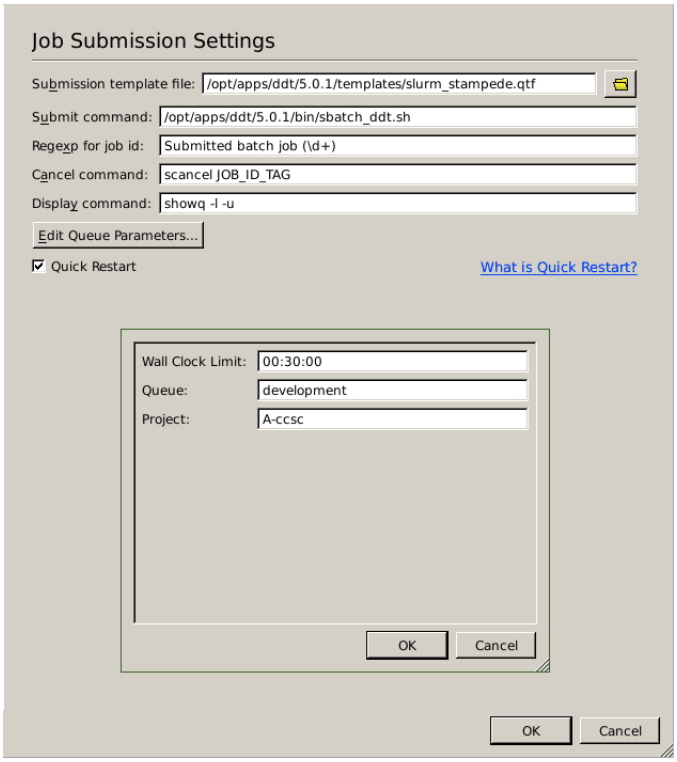
Load the module:

```
module load ddt
```

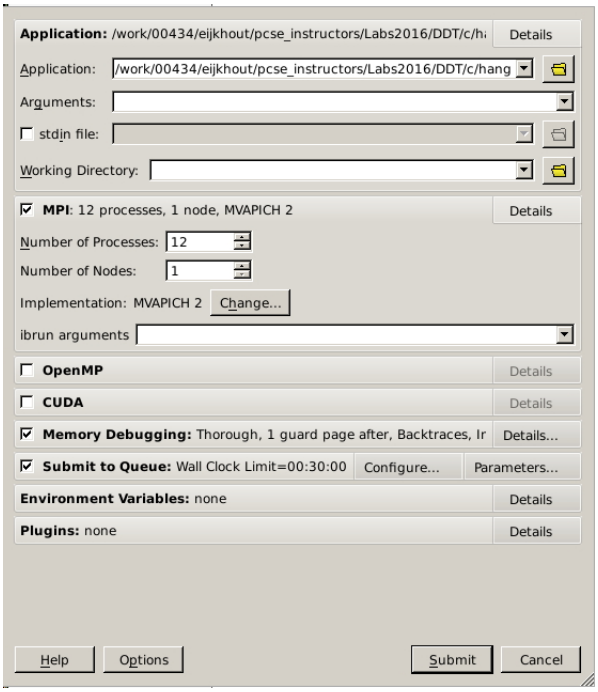
and call ddt with the name of your executable `ddt yourprog`. Make sure you have a connection with X11 forwarding: `ssh -X`.

2 Setup for your run

Clicking `Options > Queue parameters` takes you to some options that you can usually leave untouched.



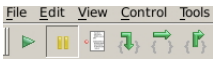
Run parameters include the name of your program, commandline arguments, whether the program is MPI or OpenMP and how many processes/threads it takes.



When you submit your program you may have to wait a little while because this in effect submitting a batch job. When the program starts, execution is paused at the `MPI_Init` statement:

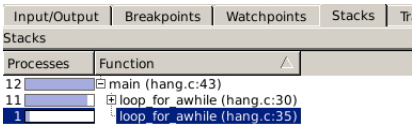
```
37 int main(int argc,char **argv) {
38   MPI_Comm comm;
39
40   MPI_Init(&argc,&argv);
41   comm = MPI_COMM_WORLD;
42
43   loop_for_awhile(comm);
44
45   MPI_Finalize();
46   return 0;
47 }
48
```

You can then use the controls for start all, pause all, add breakpoint at current line, enter a subprogram, step to next statement, leave this subprogram:

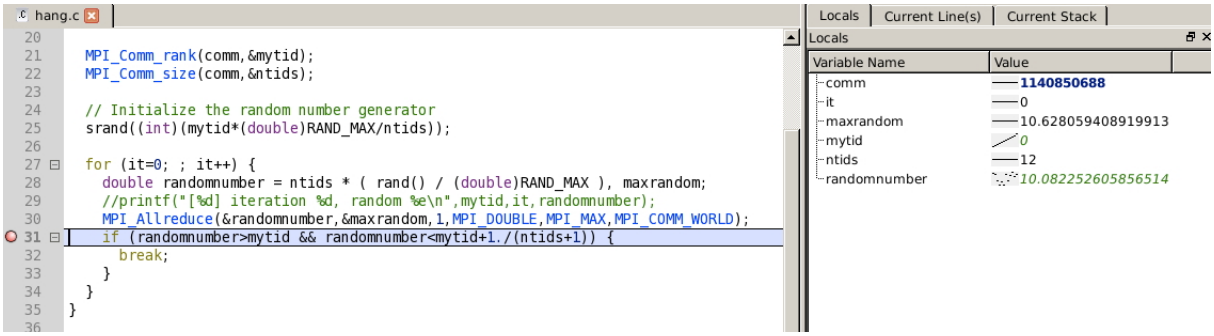


A hanging program typically looks like this: one process is red, meaning stopped or completed, and all others are green, meaning still running.

When you hit the pause button all processes are halted, and you can see in the 'stacks' panel on what line they are.



The best way to diagnose a program is to set breakpoints and (after possibly restarting the run) investigating the values of variables at the breakpoint:



At this particular breakpoint all processes have the same value for `it`, their values are linearly increasing for `mytid`, and they are all over the place for `randomnumber`.