Condor Tutorial
V 1.0
Tristan Ziska 12/09/05
Justin Ross 7/28/09 Updated

The purpose of this tutorial is to allow a new user to become familiar with the condor system, and provide a look at some of the tools and commands.

Using condor presents the user with a choice of two universes to work in, a vanilla and a standard. The vanilla universe is for programs that are not expected to take extended periods of time to run. They execute on one machine only. As such, the finial compiled binary is smaller. The standard universe is for long running programs, as programs compiled this way are automatically saved, stopped and restarted on the fly, allowing near uninterrupted work. (The system automatically saves the progress of a program, and stops it on one computer, only to resume it on another from the same point) This also allows a program to be resumed from a saved state if the computer that was working on it becomes unavailable. Aka not all work is lost. The catch is that files are larger as they are statically compiled.

Vanilla Universe:

Before you can submit a job to Condor, you need a job. For this tutorial I will create a very simple C program. First, create a file called simple.c. In that file, put the following text:

```
#include <stdio.h>

main(int argc, char **argv)
{
    int sleep_time;
    int input;
    int failure;

    if (argc != 3) {
        printf("Usage: simple <sleep-time> <integer>\n");
        failure = 1;
    } else {
        sleep_time = atoi(argv[1]);
        input      = atoi(argv[2]);

        printf("Thinking really hard for %d seconds...\n", sleep_time);
        sleep(sleep_time);
        printf("We calculated: %d\n", input * 2);
        failure = 0;
    }
    return failure;
}
```

Now compile that program:

```
[ziska@mcfarm ~]$ gcc -o simple simple.c
[ziska@mcfarm ~]$ ls -lh simple
-rwxrwxr-x    1 ziska    ziska         14k Dec  9 14:15 simple
```

Finally, run the program and tell it to sleep for four seconds and calculate 10 * 2:

```
[ziska@mcfarm ~]$ ./simple 4 10
Thinking really hard for 4 seconds...
We calculated: 20
```

Submitting your job

Now that you have a job, you just have to tell Condor to run it. Put the following text into a file called *submit*:

```
Universe   = vanilla
Executable = simple
Arguments  = 4 10
Log        = simple.log
Output     = simple.out
Error      = simple.error
Queue
```

Let's examine each of these lines:

- **Universe**: The vanilla universe means a plain old job. Later on, we'll encounter some special universes.
- **Executable**: The name of your program
- **Arguments**: These are the arguments you want. They will be the same arguments we typed above.
- **Log**: This is the name of a file where Condor will record information about your job's execution. While it's not required, it is a *really good idea* to have a log.
- **Output**: Where Condor should put the standard output from your job.
- **Error**: Where Condor should put the standard error from your job.

Next, tell Condor to run the job:

```
[ziska@mcfarm ~]$ condor_submit submit
Submitting job(s).
Logging submit event(s).
1 job(s) submitted to cluster 37.
```

Now, watch the job run:

```
[ziska@mcfarm ~]$ condor_q

-- Submitter: mcfarm.physics.smu.edu : <192.168.0.254:32783> :
mcfarm.physics.smu.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  39.0   ziska          12/9  14:20   0+00:00:00 I  0   0.0  simple 4 10

1 jobs; 1 idle, 0 running, 0 held
[ziska@mcfarm ~]$ condor_q

-- Submitter: mcfarm.physics.smu.edu : <192.168.0.254:32783> :
mcfarm.physics.smu.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  39.0   ziska          12/9  14:20   0+00:00:00 I  0   0.0  simple 4 10

1 jobs; 1 idle, 0 running, 0 held
[ziska@mcfarm ~]$ condor_q


-- Submitter: mcfarm.physics.smu.edu : <192.168.0.254:32783> :
mcfarm.physics.smu.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD

0 jobs; 0 idle, 0 running, 0 held
```

In this example there only the job that I ran displayed. However if multiple people were running jobs they would all appear. To simply see jobs submitted by a single user, use the -sub option. Eg:

```
[ziska@mcfarm ~]$ condor_q -sub ziska
```

When the job was done, it was no longer listed. Looking at the log file shows what happened.

```
[ziska@mcfarm ~]$ cat simple.log
000 (040.000.000) 12/09 14:23:41 Job submitted from host: <192.168.0.254:32783>
...
001 (040.000.000) 12/09 14:23:46 Job executing on host: <192.168.0.254:32782>
...
005 (040.000.000) 12/09 14:23:50 Job terminated.
        (1) Normal termination (return value 0)
                Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Total Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local Usage
        0  -  Run Bytes Sent By Job
        0  -  Run Bytes Received By Job
        0  -  Total Bytes Sent By Job
        0  -  Total Bytes Received By Job
...
```

The log would indicate that the program executed sucessfully, and this can be verified by looking at the output in the file previously specified.

```
[ziska@mcfarm ~]$ cat simple.out
Thinking really hard for 4 seconds...
We calculated: 20
```

Success!

Doing a parameter sweep

If you only ever had to run a single job, you probably wouldn't need Condor. Since that is usually not the case, we can modify the submit file to have condor run the program with multiple executions:

```
Universe   = vanilla
Executable = simple
Arguments  = 4 10
Log        = simple.log
Output     = simple.$(Process).out
Error      = simple.$(Process).error
Queue

Arguments = 4 11
Queue

Arguments = 4 12
Queue
```

There are two important differences to notice here. First, the Output and Error lines have the $(Process) macro in them. This means that the output and error files will be named according to the process number of the job. You'll see what this looks like in a moment. Second, we told Condor to run the same job an extra two times by adding extra Arguments and Queue statements.

```
[ziska@mcfarm ~]$ condor_submit submit
Submitting job(s)...
Logging submit event(s)...
3 job(s) submitted to cluster 41.
```

```
[ziska@mcfarm ~]$ condor_q

-- Submitter: mcfarm.physics.smu.edu : <192.168.0.254:32783> :
mcfarm.physics.smu.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  41.0   ziska          12/9  14:26   0+00:00:00 R  0   0.0  simple 4 10
  41.1   ziska          12/9  14:26   0+00:00:00 R  0   0.0  simple 4 11
  41.2   ziska          12/9  14:26   0+00:00:00 R  0   0.0  simple 4 12

2 jobs; 0 idle, 2 running, 0 held

[ziska@mcfarm ~]$ ls simple*out
simple.0.out  simple.1.out  simple.2.out  simple.out

[ziska@mcfarm ~]$ cat simple.0.out
Thinking really hard for 4 seconds...
We calculated: 20

[ziska@mcfarm ~]$ cat simple.1.out
Thinking really hard for 4 seconds...
We calculated: 22

[ziska@mcfarm ~]$ cat simple.1.out
Thinking really hard for 4 seconds...
We calculated: 24
```

Notice that the three jobs with the same cluster number, but different process numbers. They have the same cluster number because they were all submitted from the same submit file. When the jobs ran, they created three different output files, each with the desired output.

Standard Universe:

First, you need a job to run. Again, the simple C program will be used.

```
#include <stdio.h>

main(int argc, char **argv)
{
    int sleep_time;
    int input;
    int failure;

    if (argc != 3) {
        printf("Usage: simple <sleep-time> <integer>\n");
        failure = 1;
    } else {
        sleep_time = atoi(argv[1]);
        input      = atoi(argv[2]);

        printf("Thinking really hard for %d seconds...\n", sleep_time);
        sleep(sleep_time);
        printf("We calculated: %d\n", input * 2);
        failure = 0;
    }
    return failure;
}
```

Now compile the program using condor_compile. This doesn't change how the program is compiled, just how it is linked. Take note that the executable is named differently.

```
[ziska@mcfarm ~]$ condor_compile gcc -o simple.std simple.c
LINKING FOR CONDOR : /usr/bin/ld -L/grid3/condor/lib -Bstatic -m elf_i386 -
dynamic-linker /lib/ld-linux.so.2 -o simple.std /grid3/condor/lib/condor_rt0.o
/usr/lib/gcc-lib/i386-redhat-linux/2.96/../../../crti.o /usr/lib/gcc-lib/i386-
redhat-linux/2.96/crtbegin.o -L/grid3/condor/lib -L/usr/lib/gcc-lib/i386-redhat-
linux/2.96 -L/usr/lib/gcc-lib/i386-redhat-linux/2.96/../../.. /tmp/ccoDdnnK.o
/grid3/condor/lib/libcondorzsyscall.a /grid3/condor/lib/libz.a -lgcc -lc -
lnss_files -lnss_dns -lresolv -lc -lnss_files -lnss_dns -lresolv -lc -lgcc
/usr/lib/gcc-lib/i386-redhat-linux/2.96/crtend.o /usr/lib/gcc-lib/i386-redhat-
linux/2.96/../../../crtn.o /grid3/condor/lib/libcondorc++support.a
/grid3/condor/lib/libcondorzsyscall.a(condor_file_agent.o): In function
`CondorFileAgent::open(char const *, int, int)':
/scratch/cbuild/build/src/condor_ckpt/condor_file_agent.C:77: the use of `tmpnam'
is dangerous, better use `mkstemp'
gcc: file path prefix `/grid3/condor/lib/' never used

[ziska@mcfarm ~]$ ls -lh simple.std
-rwxrwxr-x    1 ziska    ziska        3.4M Dec  9 14:33 simple.std
```

Note, the executable is much bigger now. Partly that's the price of having checkpointing and partly it is because the program is now statically linked, but you can make it slightly smaller if by getting rid of debugging symbols:

```
[ziska@mcfarm ~]$ strip simple.std

[ziska@mcfarm ~]$ ls -lh simple.std
-rwxrwxr-x    1 ziska    ziska        895k Dec  9 14:34 simple.std
```

Submitting a standard universe program

Submitting a standard universe job is almost the same as a vanilla universe job. Just change the universe to standard. Here is a sample submit file, named submit.std

```
Universe   = standard
Executable = simple.std
Arguments  = 120 10
Log        = simple.log
Output     = simple.out
Error      = simple.error
Queue
```

Then submit it, with condor_submit:

```
[ziska@mcfarm ~]$ condor_submit submit.std
Submitting job(s).
Logging submit event(s).
1 job(s) submitted to cluster 42.
[ziska@mcfarm ~]$ condor_q


-- Submitter: mcfarm.physics.smu.edu : <192.168.0.254:32783> :
mcfarm.physics.smu.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  42.0   ziska          12/9  14:35   0+00:00:00 R  0   0.9  simple.std 120 10

1 jobs; 0 idle, 1 running, 0 held

Two minutes pass...

[ziska@mcfarm ~]$ condor_q


-- Submitter: mcfarm.physics.smu.edu : <192.168.0.254:32783> :
mcfarm.physics.smu.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD

0 jobs; 0 idle, 0 running, 0 held
```

And view the log file as to what happened

```
[ziska@mcfarm ~]$ cat simple.log
000 (043.000.000) 12/09 14:41:38 Job submitted from host: <192.168.0.254:32783>
...
001 (043.000.000) 12/09 14:41:42 Job executing on host: <192.168.0.254:32782>
...
005 (043.000.000) 12/09 14:43:43 Job terminated.
        (1) Normal termination (return value 0)
                Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Total Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local Usage
        1162  -  Run Bytes Sent By Job
        917196  -  Run Bytes Received By Job
        1162  -  Total Bytes Sent By Job
        917196  -  Total Bytes Received By Job
...
```

Notice that the log file has a bit more information this time: we can see how much data was transfered to and from the job because it's in the standard universe. The remote usage was not very interesting because the job just slept, but a real job would have some interesting numbers there.

Other useful information

It might be possible for a job to become hung, or unable to execute, or possibly it just needs to be stopped. For this reason, there is a command to remove the job from the queue: condor_rm <job number>

```
[ziska@mcfarm ~]$ condor_submit submit.std
Submitting job(s).
Logging submit event(s).
1 job(s) submitted to cluster 44.
[ziska@mcfarm ~]$ condor_q

-- Submitter: mcfarm.physics.smu.edu : <192.168.0.254:32783> :
mcfarm.physics.smu.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  44.0   ziska           12/9  15:37   0+00:00:00 R  0   0.9  simple.std 120 10

1 jobs; 0 idle, 1 running, 0 held

[ziska@mcfarm ~]$ condor_rm 44.0
Job 44.0 marked for removal

[ziska@mcfarm ~]$ condor_q

-- Submitter: mcfarm.physics.smu.edu : <192.168.0.254:32783> :
mcfarm.physics.smu.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD

0 jobs; 0 idle, 0 running, 0 held
```