Self-Checkpointing

Tim Cartwright
OSG Deputy Executive Director and User School Director
University of Wisconsin–Madison
The Challenge

• Suppose your job will run for a long time
  – Reminder: Look at the “Ideal Jobs” table
  – But let’s say more than about 8 hours

• Likely removed from the Execution Point before done: HTCondor will restart job somewhere else

• **But!** It starts over and loses all progress (*badput*)
Some Solutions

- **Ideal solution:** Break up job into shorter pieces
  - Try to get back into that “Ideal Jobs” column

- But this does not always work; for example, when one iteration depends on the previous one

- **Another solution:** **Self-checkpointing**
  The executable periodically saves its progress to disk – a *self*-made *checkpoint* – so that it can resume from that point if interrupted later, losing minimal progress
Requirements

- Your executable can self-checkpoint and resume progress from checkpoint file(s) upon restart
  - If you have the source code, you can probably do this
  - If not, the code must have the feature already
  - A wrapper script may be able to help, but seems tricky

- Using HTCondor ≥ 9.0.6 is good; ≥ 9.10.0 is best
  - CHTC and OSPool are both ≥ 9.10.0

- Job universe: vanilla (default) or Docker (container)
HTCondor Has 2 Ways to Checkpoint

• Exit-driven self-checkpointing
  – Since HTCondor ≥ 8.9.7
  – Waaaay better for most use cases, esp. in OSG
  – What is shown here

• Eviction-driven self-checkpointing
  – Not even worth talking about for OSG!
  – Documented in the HTCondor Manual
  – But don’t use it 😁
Technical Details
• Tell HTCondor what special exit code your software will use when checkpointing (85 is suggested):

```
checkpoint_exit_code = 85
```

• Tell HTCondor what files (on the Execution Point) to save (on the Access Point) and restore when moved to a new Execution Point — list files and directories, include output file(s) if cumulative:

```
transfer_checkpoint_files = foo.txt, ...
```
Example Submit File

```sh
executable = my_software
transfer_input_files = my_input.txt
transfer_checkpoint_files = my_output.txt, temp_dir, temp_file.txt
transfer_output_files = my_output.txt
request_cpus = 1
request_memory = 1GB
request_disk = 1GB

log = example.log
output = example.out
error = example.err

checkpoint_exit_code = 85

queue
```
Notes About Checkpointed Files

• If you omit `transfer_checkpoint_files`, HTCondor uses `transfer_output_files` (or its defaults)

• Consider Access Point storage needs; can estimate as:

  \[ \text{number of running jobs} \times \text{total size of checkpoint files} \]

  (OSPool uses your `/home` quota; elsewhere: ask admin)

• So, save only what you need! Because it identifies exact files, it can help to use `transfer_checkpoint_files`
Executable (Code) Changes

- Executable may run many times before finishing; external process (HTCondor) reruns it until \textit{done}
- Periodically write state to file(s), then immediately exit with \texttt{transfer_checkpoint_files} (85)
- Any other exit code indicates \textit{done} (good or error)
- At start-up, executable must check for presence of checkpoint file(s); \texttt{if absent}, start at beginning, but \texttt{if present}, read file(s) and resume from that point
Self-Checkpointing Frequency

- Balance overhead versus (risk of) lost computing
  - Writing to disk can be slow and restarts take time
  - Test early! Collect metrics (checkpoint & restart times)

- Look for natural checkpoint times
  - Generally, when there is the least data to write
  - Often between outermost iterations
  - Could use iteration count, time, ...

- As a starting point, checkpoint every 1–2 hours
Debugging Tips

• For testing, you can force HTCondor to stop your job and run again (new sandbox, maybe new EP): condor_vacate_job JobID

• If HTCondor has transferred checkpoint files back to the Access Point, you can get a copy with: condor_evicted_files get JobID
Step-by-Step Example
Submit Directory

```
my_software
my_input.txt
my_submit.sub
```

deleteable = my_software
transfer_input_files = my_input.txt
transfer_checkpoint_files = my_output.txt, temp_dir, temp_file.txt
transfer_output_files = my_output.txt

request_cpus = 1
request_memory = 1GB
request_disk = 1GB

log = zzz.log
output = zzz.out
error = zzz.err

checkpoint_exit_code = 85
queue
Example Step 2: Just Before Execute

Submit Directory

- my_software
- my_input.txt
- my_submit.sub
- zzz.log

Execute Directory

- my_input.txt
- my_software

Spool Directory
Example Step 3: After 1 Minute

Submit Directory

- my_software
- my_input.txt
- my_submit.sub
- zzz.log

Spool Directory

Execute Directory

- my_input.txt
- my_output.txt
- my_software
- _condor_stderr
- _condor_stdout
- temp_dir/1.txt
- temp_dir/2.txt
- temp_file.txt
- trash.txt
Example Step 4: After 1 Hour – exit(85)

Submit Directory

- my_software
- my_input.txt
- my_submit.sub
- zzz.log

Spool Directory

Execute Directory

- my_input.txt
- my_output.txt
- my_software
- _condor_stderr
- _condor_stdout
- temp_dir/42.txt
- temp_dir/43.txt
- temp_file.txt
- trash.txt
Example Step 5: Checkpoint Complete

```
transfer_checkpoint_files = my_output.txt, temp_dir, temp_file.txt
```

**Submit Directory**
- my_software
- my_input.txt
- my_submit.sub
- zzz.log

**Spool Directory**
- my_output.txt
- _condor_stderr
- _condor_stdout
- temp_dir/42.txt
- temp_dir/43.txt
- temp_file.txt
- trash.txt

**Execute Directory**
- my_input.txt
- my_output.txt
- my_software
- _condor_stderr
- _condor_stdout
- temp_dir/42.txt
- temp_dir/43.txt
- temp_file.txt.txt
- trash.txt

Job execute directory is not changed before restart.
Example Step 6: 10 Min. Later – Eviction!

Submit Directory

- my_software
- my_input.txt
- my_submit.sub
- zzz.log

Spool Directory

- my_output.txt
- temp_dir/42.txt
- temp_dir/43.txt
- temp_file.txt

Execute Directory

- my_input.txt
- my_output.txt
- my_software
- _condor_stderr
- _condor_stdout
- temp_dir/51.txt
- temp_dir/52.txt
- temp_file.txt
- trash.txt

Lose changes since last checkpoint
Example Step 7: Restart on New Execute

Submit Directory

- my_software
- my_input.txt
- my_submit.sub
- zzz.log

Spool Directory

- my_output.txt
- _condor_stderr
- _condor_stdout
- temp_dir/42.txt
- temp_dir/43.txt
- temp_file.txt

(New) Execute Directory

- my_input.txt
- my_output.txt
- my_software
- _condor_stderr
- _condor_stdout
- temp_dir/42.txt
- temp_dir/43.txt
- temp_file.txt
Example Step 8: Job Completes Normally

```
transfer_output_files = my_output.txt
```

Submit Directory

```
my_software
my_input.txt
my_output.txt
my_submit.sub
zzz.err
zzz.log
zzz.out
```

(New) Execute Directory

```
my_input.txt
my_output.txt
my_software
_condor_stderr
_condor_stdout
temp_dir/98.txt
temp_dir/99.txt
temp_file.txt.txt
trash.txt
```
Notes & Acknowledgements

• Official documentation:
  – Includes full working example (Python + submit)
  – The exercise is derived from that example

• Many thanks to Todd Miller, Christina Koch, and Jason Patton for their help!

• This work was supported by NSF grants MPS-1148698, OAC-1836650, and OAC-2030508