multishell
Running Stata parallel efficiently
... or ...
I was a final year PhD student and needed computational power....

Jan Ditzen

Heriot-Watt University, Edinburgh, UK
Center for Energy Economics Research and Policy (CEERP)

September 7, 2018
Introduction

Time is limited....

- Simulations to assess bias of an estimator run over a huge variety of different parameters. This is very time consuming.
- Likewise, running many large do files to process datasets or create tables can take a lot of time. Often it does not matter in which order they run.
- Does Stata help with it?
  - Only possible to run a single do file at a time.
  - Multi-core systems would allow parallel computing. Stata/IC and Stata/SE use only one core. Stata/MP supports multiple cores, but only commands are speeded up.
- Why not run a simulation or do files parallel.
Example: Monte Carlo to assess bias of an estimator with 5 parameterisations for number of time periods (T) and cross sections (N).

5 * 5 runs with 1000 repetitions necessary to generate this table, with no other parameters changed.

Assume 1 estimation takes 1 second, 1000 seconds needed for one parametrisation, 25,000 seconds or ~7 hours required for all simulations.

If 5 runs could be run parallel, 5000 seconds or ~1.5 hours would be needed.
Introduction - What exists?

- **parallel**
  - Inspired by R library "snow" implements parallel computing through Stata’s batch mode.
  - Can be used to speed up commands like simulate or bootstrap and speeds up computations on datasets.

- **qsub**
  - Queues a list of jobs and submits them to different Stata instances.

- **multishell**
  - A mix of both routines with the extension to use it across computers.
  - Loops (forvalues and foreach) are dissected into variations and queued.
  - The queue is then processed on multiple instances of Stata on one or more computers.
Example

- Assume a Monte Carlo to assess the bias of the OLS estimator is planned with an increasing number of observations. Results for each run are saved in a separated dataset.
- A straightforward way to code this would be:

```
simulation.do

program define MCprog, rclass
    syntax anything(name = N)
    clear
    set obs ‘N’
    drawnorm x e
    gen y = 2 + 0.5*x + e
    reg y x
    return scalar x = _b[x]
end

clear
forvalues n = 10 (10) 130 {
    simulate bx = r(x), reps(1000): MCprog ‘n’
    save results_‘n’, replace
}
```

```
 multishell.do

multishell exepath "C:/Stata/Stata.exe"
multishell path "C:/documents/multishell/temp"
multishell add ".../simulation.do"
multishell start , threads(3) sleep(2000)
```
Example

What does multishell do?

- multishell will create a new sub folder for each variation, i.e. $n = 10, n=20, n=30, ..., n=130$. In each folder a do file with the corresponding variation, a log file and file containing the status are saved.
- The do files are queued and further do files can be added.
- The running Stata instance acts as the main multishell instance. It creates a batch file for each job and coordinates the number of parallel Stata instances.
- As soon as a job (or variation) is completed, the status in the sub folder is changed and the instance will be closed.
- multishell main instance will scan the folders and check if additional instances can be started.
Single Computer

starts instances

reports

<table>
<thead>
<tr>
<th>id</th>
<th>Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>i = 10</td>
</tr>
<tr>
<td>#2</td>
<td>i = 20</td>
</tr>
<tr>
<td>#3</td>
<td>i = 30</td>
</tr>
<tr>
<td>#13</td>
<td>i = 130</td>
</tr>
</tbody>
</table>
. multishell run, threads(4) sleep(2000)

<table>
<thead>
<tr>
<th>#</th>
<th>do-file</th>
<th>State</th>
<th>Time</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>simulation.do</td>
<td>queued and running</td>
<td>2 Sep 2018 - 15:25:29</td>
<td>HPJD</td>
</tr>
<tr>
<td>1</td>
<td>n = 10</td>
<td>finished</td>
<td>2 Sep 2018 - 15:25:29</td>
<td>HPJD</td>
</tr>
<tr>
<td>2</td>
<td>n = 20</td>
<td>finished</td>
<td>2 Sep 2018 - 15:25:29</td>
<td>HPJD</td>
</tr>
<tr>
<td>3</td>
<td>n = 30</td>
<td>finished</td>
<td>2 Sep 2018 - 15:25:29</td>
<td>HPJD</td>
</tr>
<tr>
<td>4</td>
<td>n = 40</td>
<td>finished</td>
<td>2 Sep 2018 - 15:25:29</td>
<td>HPJD</td>
</tr>
<tr>
<td>5</td>
<td>n = 50</td>
<td>running</td>
<td>2 Sep 2018 - 15:25:31</td>
<td>HPJD</td>
</tr>
<tr>
<td>6</td>
<td>n = 60</td>
<td>running</td>
<td>2 Sep 2018 - 15:25:31</td>
<td>HPJD</td>
</tr>
<tr>
<td>7</td>
<td>n = 70</td>
<td>running</td>
<td>2 Sep 2018 - 15:25:31</td>
<td>HPJD</td>
</tr>
<tr>
<td>8</td>
<td>n = 80</td>
<td>running</td>
<td>2 Sep 2018 - 15:25:31</td>
<td>HPJD</td>
</tr>
<tr>
<td>9</td>
<td>n = 90</td>
<td>queued</td>
<td>2 Sep 2018 - 15:25:14</td>
<td>HPJD</td>
</tr>
<tr>
<td>10</td>
<td>n = 100</td>
<td>queued</td>
<td>2 Sep 2018 - 15:25:14</td>
<td>HPJD</td>
</tr>
<tr>
<td>11</td>
<td>n = 110</td>
<td>queued</td>
<td>2 Sep 2018 - 15:25:14</td>
<td>HPJD</td>
</tr>
<tr>
<td>12</td>
<td>n = 120</td>
<td>queued</td>
<td>2 Sep 2018 - 15:25:15</td>
<td>HPJD</td>
</tr>
<tr>
<td>13</td>
<td>n = 130</td>
<td>queued</td>
<td>2 Sep 2018 - 15:25:15</td>
<td>HPJD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Machine</th>
<th>Queued</th>
<th>Assigned</th>
<th>Running</th>
<th>Finished</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>This Computer</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Total</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Computername: HPJD
next refresh in 2s.
Cluster of Computers

- In case of multiple computers, one computer acts as the server.
- Prerequisite: the computers must have shared access to the folder `multishell` uses to save do files.
- The main instance of the server allocates tasks to the clients, so a cluster is set up.
- Each computer has a main instance, which then starts new instances of Stata processing the allocated tasks.
Cluster of Computers

- - - starts instances
- - - reports
- - - assigns tasks

<table>
<thead>
<tr>
<th>id</th>
<th>Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>i = 10</td>
</tr>
<tr>
<td>#2</td>
<td>i = 20</td>
</tr>
<tr>
<td>#3</td>
<td>i = 30</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>#13</td>
<td>i = 130</td>
</tr>
</tbody>
</table>
Cluster of Computers
In Stata (from help file)

<table>
<thead>
<tr>
<th>#</th>
<th>do-file</th>
<th>State</th>
<th>Time</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MonteCarloSimulation.do</td>
<td>running and finished</td>
<td>17 Jul 2018 - 14:26:50</td>
<td>HPJD</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>finished</td>
<td>17 Jul 2018 - 14:26:50</td>
<td>HPJD</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>finished</td>
<td>17 Jul 2018 - 14:26:50</td>
<td>HPJD</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>finished</td>
<td>17 Jul 2018 - 14:26:51</td>
<td>HPJD</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>finished</td>
<td>17 Jul 2018 - 14:26:52</td>
<td>HPJD</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>running</td>
<td>17 Jul 2018 - 14:26:50</td>
<td>HPJD</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>finished</td>
<td>17 Jul 2018 - 14:26:41</td>
<td>Research181</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>finished</td>
<td>17 Jul 2018 - 14:26:41</td>
<td>Research181</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>finished</td>
<td>17 Jul 2018 - 14:26:50</td>
<td>Research181</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>do-file</th>
<th>State</th>
<th>Time</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>MonteCarloSimulation_panel.do</td>
<td>queued and running</td>
<td>17 Jul 2018 - 14:26:43</td>
<td>Research181</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>running</td>
<td>17 Jul 2018 - 14:26:53</td>
<td>HPJD</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>assigned</td>
<td>17 Jul 2018 - 14:26:31</td>
<td>HPJD</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>assigned</td>
<td>17 Jul 2018 - 14:26:32</td>
<td>HPJD</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>running</td>
<td>17 Jul 2018 - 14:26:52</td>
<td>Research181</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>assigned</td>
<td>17 Jul 2018 - 14:26:32</td>
<td>HPJD</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>assigned</td>
<td>17 Jul 2018 - 14:26:32</td>
<td>HPJD</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>queued</td>
<td>17 Jul 2018 - 14:26:33</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>17 Jul 2018 - 14:26:33</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Machine</th>
<th>Queued</th>
<th>Assigned</th>
<th>Running</th>
<th>Finished</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPJD</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>This Computer</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

Total          2 4 4 8 16

Computername: Research181
as of 17 Jul 2018 - 14:26:54; started at 17 Jul 2018 - 14:26:33
next refresh in 2s.
Syntax and set-up I

1 Set paths.
   - multishell path "C:/Documents/Multishell"
     Path for folder to store files.
   - multishell exepath "C:/Programs/Stata/Stata.exe"
     Path to Stata exe.

2 Add do files.
   - multishell add "C:/Documents/Multishell/simulation.do"
     Do file to be queued. For each job, a sub folder in the path set above is created, do file and status file are saved.

3 Additional Parameters
   - multishell adopath "C:/Documents/myado"
     Load additional ados.
   - multishell alttext "old text @ new text"
     Replace old text in with new text. Possible to adjust paths in the do file for each computer.
Syntax and set-up II

- `multishell seed type filename, [fill]`
  (...yes, I am using Stata 14 and not Stata 15)

  Setting up the seed using dataset `filename`. `type` can be
  - `create` creates a dataset with empty seeds for each variation. If option `fill` is used, then seeds are random numbers.
  - `save` saves the dataset with the seeds used for each variation in `filename`.
  - `load` uses seeds from dataset `filename`.

4 Start the `multishell` server (or client).

- `multishell run [client] , threads(integer) sleep(integer) [nostop networkdrive]`
  Starts the multishell main instance. If option `client` is used, then the instance is started as a client and waits for a server to assign tasks to the computer.

- Options
  - `threads(integer)` Sets the number of parallel Stata instances.
  - `sleep(integer)` milliseconds until status of tasks is refreshed.
  - `nostop` Client is restarted if all tasks are finished.
  - `networkdrive` log file is saved in the path folder.
Syntax and set-up III

5 Diagnosis

- **multishell status**
  Shows the status of the multishell, including the number of tasks, clients and path set up.

- **multishell reset type, computer(Computername)**
  Re-queues tasks for computer.
  where *type* is assigned, running, finished, error, *id(#)*
multishell_server.do

local google_drive "C:/Users//‘c(username)’/Google Drive/Papers/Project"

multishell path "‘google_drive’/Code/simulation/temp" , clear
multishell exepath "C:/Program Files (x86)/Stata14/StataSE-64.exe"
multishell adopath "‘google_drive’/Code/ados/"
multishell alttext "GOOGLE_FILE @ ‘google_drive’"
multishell add "‘google_drive’/Code/simulation/simulation_loop.do"
multishell seed create seed_all , fill
multishell run , threads(7) sleep(1000) network

multishell_client.do

local google_drive "C:/Users//‘c(username)’/Google Drive/Uni/Research/Project"

multishell path "‘google_drive’/Code/simulation/temp"
multishell exepath "C:/Program Files (x86)/Stata14/StataSE-64.exe"
multishell adopath "‘google_drive’/Code/ados/"
multishell alttext "GOOGLE_FILE @ ‘google_drive’"
multishell run client, threads(4) sleep(1000) network
Performance
Is it all worth it?

Simulation from above repeated with varying number of threads on an Intel Core i5-2450M with 4 cores, Windows 7 and Stata 14.2.

<table>
<thead>
<tr>
<th>Threads</th>
<th>Seconds</th>
<th>Threads</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>202.83</td>
<td>6</td>
<td>128.53</td>
</tr>
<tr>
<td>2</td>
<td>131.76</td>
<td>7</td>
<td>135.40</td>
</tr>
<tr>
<td>3</td>
<td>113.37</td>
<td>8</td>
<td>147.14</td>
</tr>
<tr>
<td>4</td>
<td>113.91</td>
<td>9</td>
<td>151.18</td>
</tr>
<tr>
<td>5</td>
<td>143.88</td>
<td>10</td>
<td>127.31</td>
</tr>
</tbody>
</table>
Limitations

(Sadly) there are some limitations

- Only Windows is supported.
- `multishell` only speeds up loops or processing multiple do files. It does not improve the speed of Stata commands.
- If there are synch or speed problems with Cloud services such as Google Backup and Sync, Dropbox, etc. or the local network, `multishell` will slow down or stop. Read/write problems in a local network may occur as well and cause problems.
- If run on a mapped network drive, then the log files may be saved in My Documents or the Stata folder.
- No locals in loops are supported (such as foreach type in ‘one’ ‘two’ ‘three’).
- All loops are dissected.
Conclusion

- multishell helps to speed up simulations or running multiple large do files.
- Parallel instances of Stata can be run on a single machine. Number depends on the number of cores.
- Computational power from multiple machines can be combined by mimicking a cluster.
- On SSC since July.

Outlook
- More robust for networks and less tempfiles.
- Ordering the tasks better.
- Allow to preserve loops.
DITZEN, J. (2017): “XTDCCE2: Stata module to estimate heterogeneous coefficient models using common correlated effects in a dynamic panel,”.