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 asses 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 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.

Introduction - Example

(N,T)	Bias (x100)					RMSE (x100)				
	40	50	100	150	200	40	50	100	150	200
$\phi = 1/N \sum_{i=1}^{N} \phi_i$										
40	-42.85	-31.69	-13.52	-8.06	-5.54	18.14	13.53	6.26	4.11	3.02
50	-42.91	-30.28	-13.45	-8.25	-6.33	18.03	12.95	6.11	3.99	3.18
100	-43.33	-31.51	-13.66	-8.83	-6.17	17.86	12.96	5.85	3.90	2.78
150	-42.16	-31.11	-13.73	-8.74	-6.31	17.23	12.70	5.70	3.72	2.75
200	-43.65	-31.43	-13.69	-8.96	-6.19	17.75	12.80	5.67	3.73	2.65
$\beta_0 = 1/N \sum_{i=1}^{N} \beta_{0i}$										
40	4.34	3.37	2.05	1.30	0.67	12.47	9.21	5.62	4.40	3.66
50	4.78	3.10	2.13	1.26	1.24	11.10	8.65	5.32	4.25	3.77
100	4.27	3.71	2.17	1.15	1.18	8.54	6.55	4.10	3.11	2.62
150	3.66	3.96	2.07	1.30	1.02	7.04	5.94	3.47	2.56	2.15
200	5.21	4.07	2.34	1.39	0.96	6.71	5.17	3.18	2.37	1.89
$\beta_1 = 1/N \sum_{i=1}^{N} \beta_{1i}$										
40	-8.61	-6.20	-1.68	-0.91	-0.91	12.07	10.20	6.02	4.31	3.97
50	-6.82	-5.27	-1.83	-1.02	-1.24	11.54	9.25	5.55	4.06	3.67
100	-5.12	-3.04	-1.07	-1.10	-0.16	8.14	6.41	3.75	3.15	2.66
150	-6.78	-2.76	-0.45	-0.39	-0.29	7.50	5.88	3.32	2.65	2.15
200	-5.13	-2.88	-0.44	-0.68	-0.21	6.63	5.07	2.79	2.31	1.85

Table: Monte Carlo Results for coefficients ϕ, β_0 and β_1, estimating a dynamic common correlated effects model using `xtdcce2`. The DGP is $y_{i,t} = c_{yi} + \phi_i y_{i,t-1} + \beta_{0i} x_{i,t} + \beta_{1i} x_{i,t-1} + \gamma_i' f_t + \epsilon_{i,t}$. Example taken from Table 1 Ditzen (2017).

- Example: Monte Carlo to asses bias of an estimator with 5 parametrisations 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 seperated 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

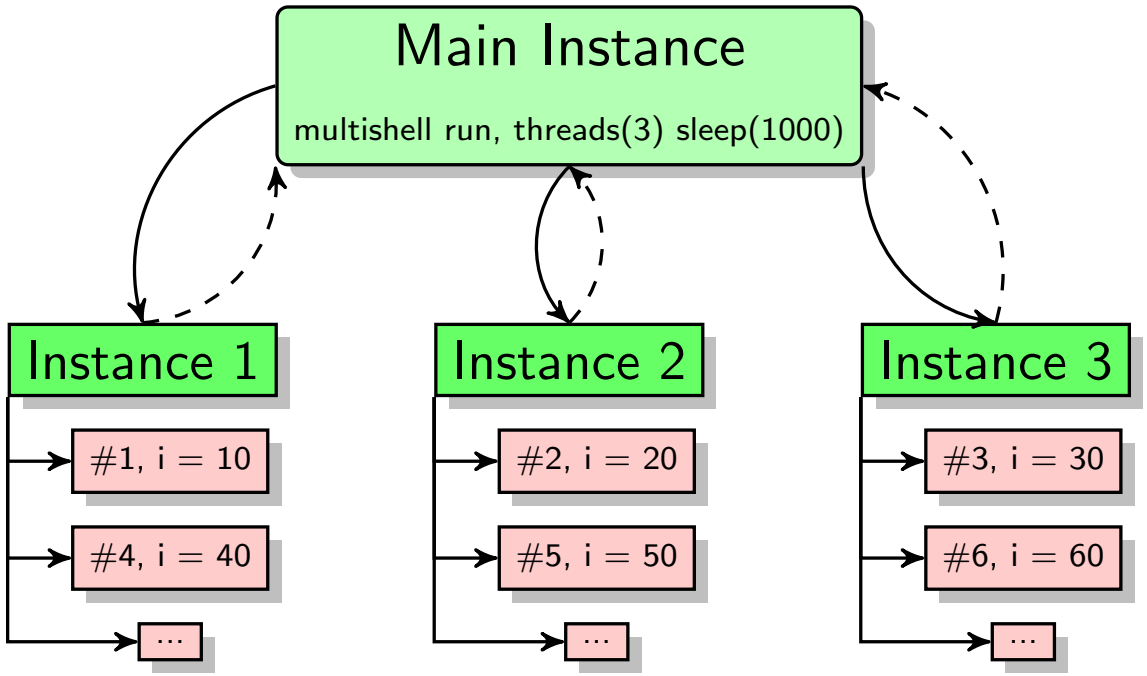
```
multishell exepath "C:/Stata/Stata.exe"
multishell path "C:/documents/multishell/temp"
multishell add ".../simulation.do"
multishell start , threads(3) sleep(2000)
```

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



forvalues i = 10 (10) 130:

id	Variation
#1	i = 10
#2	i = 20
#3	i = 30
⋮	⋮
#13	i = 130

⟶ starts instances

- - - → reports

Single Computer

In Stata

```
. multishell run, threads(4) sleep(2000)
```

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

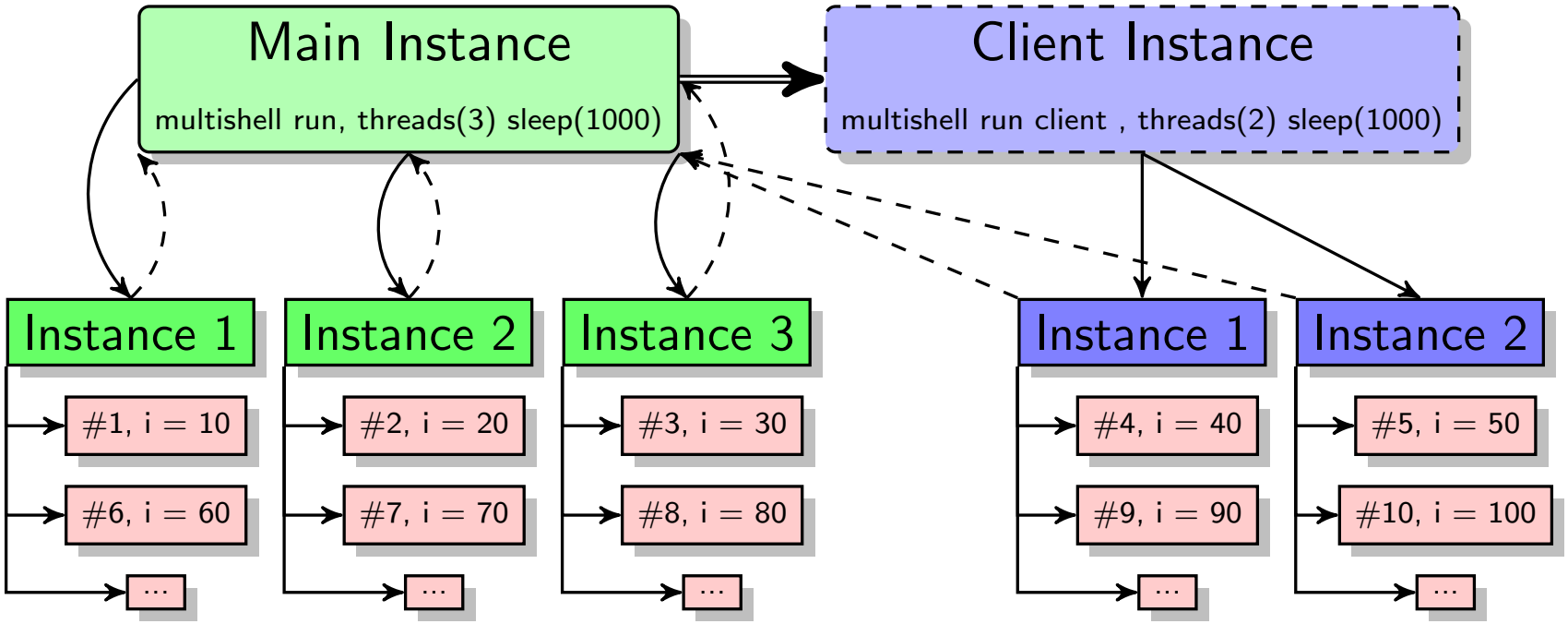
Machine	Queued	Assigned	Running	Finished	Total
This Computer	0	0	4	4	8
Total	5	0	4	4	8

```
Computername: HPJD
as of  2 Sep 2018 - 15:25:32; started at  2 Sep 2018 - 15:25:15
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

forvalues i = 10 (10) 130:

id	Variation
#1	i = 10
#2	i = 20
#3	i = 30
⋮	⋮
#13	i = 130

Cluster of Computers

In Stata (from help file)

#	do-file	State	Time	Machine
	MonteCarloSimulation.do	running and finished		
1	n = 50	finished	17 Jul 2018 - 14:26:50	HPJD
2	n = 60	finished	17 Jul 2018 - 14:26:50	HPJD
3	n = 70	finished	17 Jul 2018 - 14:26:50	HPJD
4	n = 80	finished	17 Jul 2018 - 14:26:51	HPJD
5	n = 90	finished	17 Jul 2018 - 14:26:52	HPJD
6	n = 100	running	17 Jul 2018 - 14:26:50	HPJD
7	n = 110	finished	17 Jul 2018 - 14:26:41	Research181
8	n = 120	finished	17 Jul 2018 - 14:26:41	Research181
9	n = 130	finished	17 Jul 2018 - 14:26:50	Research181
	MonteCarloSimulation_panel.do	queued and running		
10	n = 30 , t = 30	running	17 Jul 2018 - 14:26:43	Research181
11	n = 30 , t = 40	running	17 Jul 2018 - 14:26:53	HPJD
12	n = 30 , t = 50	assigned	17 Jul 2018 - 14:26:31	HPJD
13	n = 40 , t = 30	assigned	17 Jul 2018 - 14:26:32	HPJD
14	n = 40 , t = 40	running	17 Jul 2018 - 14:26:52	Research181
15	n = 40 , t = 50	assigned	17 Jul 2018 - 14:26:32	HPJD
16	n = 50 , t = 30	assigned	17 Jul 2018 - 14:26:32	HPJD
17	n = 50 , t = 40	queued	17 Jul 2018 - 14:26:33	
18	n = 50 , t = 50	queued	17 Jul 2018 - 14:26:33	

Machine	Queued	Assigned	Running	Finished	Total
HPJD	0	4	2	5	11
This Computer	0	0	2	3	5
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(#)`*

Example

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.

Threads	Seconds	Threads	Seconds
1	202.83	6	128.53
2	131.76	7	135.40
3	113.37	8	147.14
4	113.91	9	151.18
5	143.88	10	127.31

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.

References I

Ditzen, J. (2017): "XTDCCE2: Stata module to estimate heterogeneous coefficient models using common correlated effects in a dynamic panel," .