Just tired of endless loops!

or parallel: Stata module for parallel computing

George G. Vega
gvega@spensiones.cl

Chilean Pension Supervisor

Stata Conference New Orleans
July 18-19, 2013

Thanks to Damian C. Clarke, Félix Villatoro and Eduardo Fajnzylber, Tomás Rau, Eric Melse, Valentina Moscoso, the Research team of the Chilean Pension Supervisor and several Stata users worldwide for their valuable contributions. The usual disclaimers applies.
Agenda

1. Motivation
2. What is and how does it work
3. Benchmarks
4. Syntax and Usage
5. Concluding Remarks
Motivation

- Despite the availability of administrative data, its exploitation is still a novel issue.
Motivation

• Despite the availability of administrative data, its exploitation is still a novel issue.

• At the same time, currently home computers are arriving with extremely high computational capabilities.
Motivation

• Despite the availability of administrative data, its exploitation is still a novel issue.
• At the same time, currently home computers are arriving with extremely high computational capabilities.
• Given its nature, matching both (big data problems and HPA) sounds straightforward.
Motivation

• Despite the availability of administrative data, its exploitation is still a novel issue.
• At the same time, currently home computers are arriving with extremely high computational capabilities.
• Given its nature, matching both (big data problems and HPA) sounds straightforward.
• But, implementing parallel computing for the social scientiest is not easy,
Motivation

- Despite the availability of administrative data, its exploitation is still a novel issue.
- At the same time, currently home computers are arriving with extremely high computational capabilities.
- Given its nature, matching both (big data problems and HPA) sounds straightforward.
- But, implementing parallel computing for the social scientist is not easy, most of this due to lack of (user-friendly) statistical computing tools.
Motivation

- Despite the availability of administrative data, its exploitation is still a novel issue.
- At the same time, currently home computers are arriving with extremely high computational capabilities.
- Given its nature, matching both (big data problems and HPA) sounds straightforward.
- But, implementing parallel computing for the social scientist is not easy, most of this due to lack of (user-friendly) statistical computing tools.
- parallel aims to make a contribution to these issues.
1 Motivation

2 What is and how does it work

3 Benchmarks

4 Syntax and Usage

5 Concluding Remarks
What is and how does it work

What is?

• Inspired in the R package “snow”
What is and how does it work

What is?

- Inspired in the R package “snow” (several other examples exists: StataMP, Condor HTC, C’s Ox library, Matlab’s Parallel Toolbox, etc.)
What is and how does it work

What is?

• Inspired in the R package “snow” (several other examples exist: StataMP, Condor HTC, C’s Ox library, Matlab’s Parallel Toolbox, etc.)
• Is designed to be used in multicore CPUs (dualcore, quadcore, etc.).
What is and how does it work

What is?

- Inspired in the R package “snow” (several other examples exists: StataMP, Condor HTC, C’s Ox library, Matlab’s Parallel Toolbox, etc.)
- Is designed to be used in multicore CPUs (dualcore, quadcore, etc.).
- It implements parallel computing methods through an OS’s shell scripting (using Stata in batch mode) to accelerate computations.
What is and how does it work

What is?

- Inspired in the R package “snow” (several other examples exist: StataMP, Condor HTC, C’s Ox library, Matlab’s Parallel Toolbox, etc.)
- Is designed to be used in multicore CPUs (dualcore, quadcore, etc.).
- It implements parallel computing methods through an OS’s shell scripting (using Stata in batch mode) to accelerate computations.
- Depending on the task, can reach near to (or over) linear speedups proportional to the number of physical cores of the computer.
What is and how does it work

What is?

- Inspired in the R package “snow” (several other examples exist: StataMP, Condor HTC, C’s Ox library, Matlab’s Parallel Toolbox, etc.)
- Is designed to be used in multicore CPUs (dualcore, quadcore, etc.).
- It implements parallel computing methods through an OS’s shell scripting (using Stata in batch mode) to accelerate computations.
- Depending on the task, can reach near to (or over) linear speedups proportional to the number of physical cores of the computer.
- Thus having a quad-core computer can lead to a 400% speedup.
What is and how does it work

How does it work?

Splitting the data set

Task (stata batch-mode)

Appending the data set
What is and how does it work

Sounds “pretty” but...
What is and how does it work

Sounds “pretty” but...is this for real!?
What is and how does it work
Parallel’s backend

When the user enters

```
parallel:  gen n = _N
```

parallel takes the command and writes something like this
What is and how does it work
Parallel’s backend

When the user enters

```
parallel: gen n = _N
```

`parallel` takes the command and writes something like this

```plaintext
cap clear all
cd ~
1 set seed 34815
   set memory 16777216b
   cap set maxvar 5000
   cap set matsize 400
2 local pll_instance 1
   local pll_id efcql2tspr
capture {
   noisily {
3 use __pplefcql2tsprdataset if _ecfql2tsprcut == 1
   gen n = _N
   }
}
4 save __pplefcql2tsprdata1, replace
   local result = _rc
cd ~
5 mata: write_diagnosis(st_local("result"),
>"__pplefcql2tsprfinito1")
```
What is and how does it work
Parallel's backend

When the user enters

```plaintext
parallel: gen n = _N
```

parallel takes the command and writes something like this

```plaintext
cap clear all
cd ~
1 set seed 34815
set memory 16777216b
cap set maxvar 5000
cap set matsize 400
2 local pll_instance 1
   local pll_id efcql2tspr
capture {
   noisily {
3 use __plefcql2tsprdataset if _ecfql2tsprcut == 1
   gen n = _N
   } }
4 save __plefcql2tsprdata1, replace
   local result = _rc
cd ~
5 mata: write_diagnosis(st_local("result"),
   "__plefcql2tsprfinito1")
```

```plaintext
cap clear all
cd ~
1 set seed 98327
set memory 16777216b
cap set maxvar 5000
cap set matsize 400
2 local pll_instance 2
   local pll_id efcql2tspr
capture {
   noisily {
3 use __plefcql2tsprdataset if _ecfql2tsprcut == 2
   gen n = _N
   } }
4 save __plefcql2tsprdata2, replace
   local result = _rc
cd ~
5 mata: write_diagnosis(st_local("result"),
   "__plefcql2tsprfinito2")
```
Ok, it works but...
Ok, it works but... it must be really hard to use!
1 Motivation

2 What is and how does it work

3 Benchmarks

4 Syntax and Usage

5 Concluding Remarks
Serial fashion

do mydofile.do

Parallel fashion

parallel do mydofile.do

Figure: mydofile.do

```
local size = _N
forval i=1/`size’ {
    qui replace x = ///
        1/sqrt(2*`c(pi)‘)*exp(-`x^2/2‘) in ‘i’
}
```

Table: Serial replacing using a loop on a Linux Server (16 clusters)

<table>
<thead>
<tr>
<th></th>
<th>100.000</th>
<th>1.000.000</th>
<th>10.000.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1.43</td>
<td>16.94</td>
<td>144.68</td>
</tr>
<tr>
<td>Total</td>
<td>0.34</td>
<td>3.20</td>
<td>12.49</td>
</tr>
<tr>
<td>Setup</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Compute</td>
<td>0.32</td>
<td>3.07</td>
<td>11.54</td>
</tr>
<tr>
<td>Finish</td>
<td>0.02</td>
<td>0.12</td>
<td>0.95</td>
</tr>
<tr>
<td>Ratio (compute)</td>
<td>4.50</td>
<td>5.51</td>
<td>12.53</td>
</tr>
<tr>
<td>Ratio (total)</td>
<td>4.22 (26%)</td>
<td>5.30 (30%)</td>
<td>11.58 (72%)</td>
</tr>
</tbody>
</table>

Tested on a Intel Xeon X470 (hexadeca-core) machine
Benchmarks
Monte Carlo simulation (Windows Machine)

Serial fashion

do myexperiment.do

Parallel fashion

parallel do myexperiment.do, nodata

Table: Monte Carlo Experiment on a Windows Machine (4 clusters)

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>111.49</td>
<td>114.13</td>
</tr>
<tr>
<td>Total</td>
<td>58.02</td>
<td>37.48</td>
</tr>
<tr>
<td>Setup</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Compute</td>
<td>58.02</td>
<td>37.48</td>
</tr>
<tr>
<td>Finish</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Ratio (compute)</td>
<td>1.92</td>
<td>3.04</td>
</tr>
<tr>
<td>Ratio (total)</td>
<td>1.92 (96%)</td>
<td>3.04 (76%)</td>
</tr>
</tbody>
</table>

Tested on a Intel i3 2120 (dual-core) machine
Serial fashion

do myexperiment.do

Parallel fashion

parallel do myexperiment.do, nodata

**Table: Monte Carlo Experiment on a Linux Server (16 clusters)**

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>164.79</td>
<td>164.04</td>
<td>162.84</td>
<td>163.89</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>69.85</td>
<td>34.28</td>
<td>19.00</td>
<td>10.78</td>
</tr>
<tr>
<td><strong>Setup</strong></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td><strong>Compute</strong></td>
<td>69.85</td>
<td>34.28</td>
<td>19.00</td>
<td>10.78</td>
</tr>
<tr>
<td><strong>Finish</strong></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Ratio (compute)**

<table>
<thead>
<tr>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.36</td>
<td>4.78</td>
<td>8.57</td>
<td>15.21</td>
</tr>
</tbody>
</table>

**Ratio (total)**

<table>
<thead>
<tr>
<th>2 (118%)</th>
<th>4 (120%)</th>
<th>8 (107%)</th>
<th>16 (95%)</th>
</tr>
</thead>
</table>

Tested on a Intel Xeon X470 (hexadeca-core) machine
Serial fashion

\[
\text{reshape wide tipsolic rutemp opta derecho ngiros, ///} \\
i(id) j(time)
\]

Parallel fashion

\[
\text{parallel, by(id) : reshape wide tipsolic rutemp opta derecho ngiros, ///} \\
i(id) j(time)
\]

**Table:** Reshaping wide a large database on a Linux Server (8 clusters)

<table>
<thead>
<tr>
<th></th>
<th>100,000</th>
<th>1,000,000</th>
<th>5,000,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>5.51</td>
<td>72.70</td>
<td>392.97</td>
</tr>
<tr>
<td>Total</td>
<td>2.33</td>
<td>17.46</td>
<td>86.44</td>
</tr>
<tr>
<td>Setup</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Compute</td>
<td>1.83</td>
<td>12.42</td>
<td>57.93</td>
</tr>
<tr>
<td>Finish</td>
<td>0.50</td>
<td>5.04</td>
<td>28.51</td>
</tr>
<tr>
<td>Ratio (compute)</td>
<td>3.01</td>
<td>5.85</td>
<td>6.78</td>
</tr>
<tr>
<td>Ratio (total)</td>
<td>2.37 (29%)</td>
<td>4.16 (52%)</td>
<td>4.55 (57%)</td>
</tr>
</tbody>
</table>

Tested on a Intel Xeon X470 (hexadeca-core) machine
1 Motivation

2 What is and how does it work

3 Benchmarks

4 Syntax and Usage

5 Concluding Remarks
Setup

```
parallel setclusters # [, force]
```
Syntax and Usage

Setup

```
parallel setclusters # [, force]
```

By syntax

```
parallel [, by(varlist) programs mata seeds(string) randtype(random.org|datetime) processors(integer) nodata]: stata_cmd
```
Syntax and Usage

Setup

```
parallel setclusters # [, force]
```

By syntax

```
parallel [, by(varlist) programs mata seeds(string) randtype(random.org|datetime)
processors(integer) nodata]: stata_cmd
```

Do syntax

```
parallel do filename
[, by(varlist) programs mata seeds(string) randtype(random.org|datetime)
processors(integer) nodata]
```
**Syntax and Usage**

Recomendations on its usage

parallel suit ...

- Montecarlo simulation.
Recomendations on its usage

parallel suit ...

- Montecarlo simulation.
- Extensive nested control flow (loops, while, ifs, etc.).
parallel suit ...

- Montecarlo simulation.
- Extensive nested control flow (loops, while, ifs, etc.).
- Bootstraping/Jacknife.
parallel suit ...

- Montecarlo simulation.
- Extensive nested control flow (loops, while, ifs, etc.).
- Bootstraping/Jacknife.
- Simulations in general.
parallel suit ...

- Montecarlo simulation.
- Extensive nested control flow (loops, while, ifs, etc.).
- Bootstraping/Jacknife.
- Simulations in general.

parallel doesn’t suit ...

- (already) fast commands.
**Syntax and Usage**

**Recommendations on its usage**

---

**parallel suit ...**

- Montecarlo simulation.
- Extensive nested control flow (loops, while, ifs, etc.).
- Bootstrapping/Jacknife.
- Simulations in general.

---

**parallel doesn’t suit ...**

- (already) fast commands.
- Regressions, ARIMA, etc.
Syntax and Usage
Recomendations on its usage

parallel suit ...

- Montecarlo simulation.
- Extensive nested control flow (loops, while, ifs, etc.).
- Bootstraping/Jacknife.
- Simulations in general.

parallel doesn’t suit ...

- (already) fast commands.
- Regulations, ARIMA, etc.
- Linear Algebra.
**Syntax and Usage**

Recomendations on its usage

---

**parallel suit** ...

- Montecarlo simulation.
- Extensive nested control flow (loops, while, ifs, etc.).
- Bootstraping/Jacknife.
- Simulations in general.

**parallel doesn’t suit** ...

- (already) fast commands.
- Regressions, ARIMA, etc.
- Linear Algebra.
- Whatever StataMP does better.
Concluding Remarks

• In the case of Stata, parallel is, to the authors knowledge, the first public user-contribution to parallel computing
In the case of Stata, parallel is, to the authors knowledge, the first public user-contribution to parallel computing. Its major strengths/advantages are in simulation models and non-vectorized operations such as control-flow statements.
In the case of Stata, parallel is, to the authors knowledge, the first public user-contribution to parallel computing.

Its major strengths/advantages are in simulation models and non-vectorized operations such as control-flow statements.

Depending on the proportion of the algorithm that can be de-serialized, it is possible to reach near to constant scale speedups.
Concluding Remarks

• In the case of Stata, parallel is, to the authors knowledge, the first public user-contribution to parallel computing
• its major strengths/advantages are in simulation models and non-vectorized operations such as control-flow statements.
• Depending on the proportion of the algorithm that can be de-serialized, it is possible to reach near to constant scale speedups.
• parallel establishes a new basis for parallel computing in Stata,
Concluding Remarks

- In the case of Stata, parallel is, to the authors' knowledge, the first public user-contribution to parallel computing.
- Its major strengths/advantages are in simulation models and non-vectorized operations such as control-flow statements.
- Depending on the proportion of the algorithm that can be de-serialized, it is possible to reach near to constant scale speedups.
- parallel establishes a new basis for parallel computing in Stata, thus an all new set of algorithms can be implemented:
Concluding Remarks

• In the case of Stata, parallel is, to the authors knowledge, the first public user-contribution to parallel computing
• its major strengths/advantages are in simulation models and non-vectorized operations such as control-flow statements.
• Depending on the proportion of the algorithm that can be de-serialized, it is possible to reach near to constant scale speedups.
• parallel establishes a new basis for parallel computing in Stata, thus an all new set of algorithms can be implemented:
  • parsimulate
Concluding Remarks

- In the case of Stata, `parallel` is, to the authors knowledge, the first public user-contribution to parallel computing.
- Its major strengths/advantages are in simulation models and non-vectorized operations such as control-flow statements.
- Depending on the proportion of the algorithm that can be de-serialized, it is possible to reach near to constant scale speedups.
- `parallel` establishes a new basis for parallel computing in Stata, thus an all new set of algorithms can be implemented:
  - `parsimulate`
  - `parfor`
Concluding Remarks

- In the case of Stata, parallel is, to the authors knowledge, the first public user-contribution to parallel computing.
- Its major strengths/advantages are in simulation models and non-vectorized operations such as control-flow statements.
- Depending on the proportion of the algorithm that can be de-serialized, it is possible to reach near to constant scale speedups.
- Parallel establishes a new basis for parallel computing in Stata, thus an all new set of algorithms can be implemented:
  - parsimulate
  - parfor
  - parbootstrap
Concluding Remarks

- In the case of Stata, parallel is, to the authors knowledge, the first public user-contribution to parallel computing.
- Its major strengths/advantages are in simulation models and non-vectorized operations such as control-flow statements.
- Depending on the proportion of the algorithm that can be de-serialized, it is possible to reach near to constant scale speedups.
- Parallel establishes a new basis for parallel computing in Stata, thus an all new set of algorithms can be implemented:
  - parsimulate
  - parfor
  - parbootstrap
  - parnnmatch
Concluding Remarks

- In the case of Stata, parallel is, to the authors knowledge, the first public user-contribution to parallel computing.
- Its major strengths/advantages are in simulation models and non-vectorized operations such as control-flow statements.
- Depending on the proportion of the algorithm that can be de-serialized, it is possible to reach near to constant scale speedups.
- Parallel establishes a new basis for parallel computing in Stata, thus an all new set of algorithms can be implemented:
  - parsimulate
  - parfor
  - parbootstrap
  - parnmatch
  - ... You name it!
Thank you very much!

George G. Vega

gvega@spensiones.cl

Chilean Pension Supervisor

Stata Conference New Orleans
July 18-19, 2013