2014 Stata Conference

Boston, Massachusetts

Distributed computations in Stata

Michael Lokshin Sergiy Radyakin

mlokshin@worldbank.org sradyakin@worldbank.org

Research Department (DECRG),
The World Bank

August 1, 2014

DEVELOPMENT RESEARCH

THE WORLD BANK

Motivation

- there is a number of computational tasks, which clearly consist of a large number of repeating and isolated steps;
- one example is bootstrap;
- another example is simulation;

While the data may be available, it is the computational power restrictions that challenge researchers. Some simulations can run for weeks.

Increasing performance

- First choice is usually a CPU upgrade = get more MHz. (saves licenses);
- Modern computers are commonly equipped with multiple processors (cores); Stata/MP can make use of multiple CPUs or cores (up to 32 on a single machine);
- Not all commands benefit from parallelization (see MP Report for details);
- When a command can't be parallelized, resources are effectively idling, and are available to other programs on the same computer. This can be exploited by running multiple Stata instances on the same computer. This is the idea behind the `PARALLEL` package by George Vega Yon. `PARALLEL` is limited to one computer.

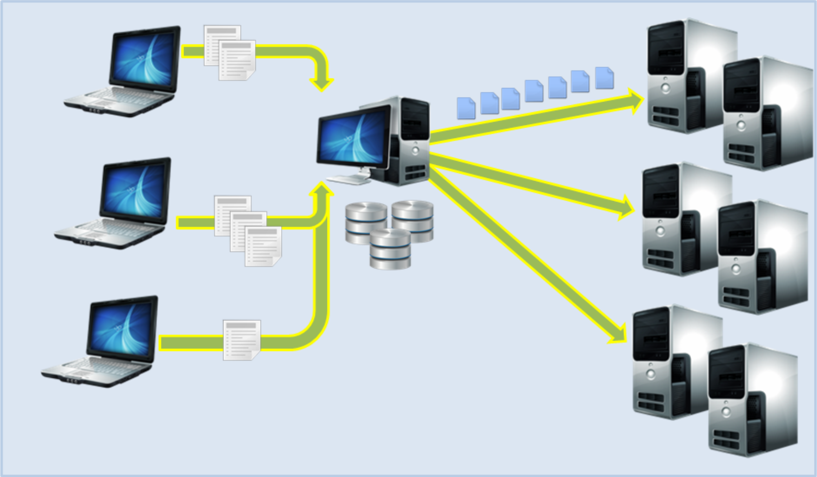
Introducing HPCCMD

- We want to move further and join the power of multiple computers in a network. This effort requires coordination of multiple computers for distributing tasks and collecting results.
- HPCCMD is a collection of software components to implement distributed computations.
- Stata is just one of many possible applications. The software itself is written generically enough to permit other applications.

HPCCMD: Components and terminology

- Aserver - coordinator server, which receives tasks and distributes them among the computational servers. Aserver must have a fixed IP to be found by other components. Aserver usually does not perform any other tasks (to maximize the response rate) and doesn't have to have Stata installed.
- Bserver - a network node, receiving computational tasks. Bservers execute an engine and host a performer. The higher the performance of each of such station, the higher is the overall performance.
- Performer - a software component actually responsible for performing computations. [e.g. Stata]
- Engine - an adapter interfacing the bserver with the performer.
- Client - user's machine that submits the jobs for computations. May or may not have Stata installed depending on use and configuration.
- Job - an assignment of the user to the cluster. Usually a collection of multiple tasks.
- Task - an individual assignment of the aserver to the bserver. Usually a part of a larger set of tasks, a job.

Overview of the system



User workstations Cluster gateway and data storage Server farm

Aserver

Aserver performs the following operations:

- coordinating the pool of the available computational nodes by registering new nodes and unregistering exiting or dead nodes.
- receiving the jobs from the clients, and managing the queue of jobs;
- coordination of distribution of tasks among the bservers;
- coordinating access to shared resources: data and code;
- collecting results of individual tasks and compiling the results of jobs;
- storing the job results until they are collected by the clients.

Aserver requires admin rights to be installed and operate.

Bserver

Bserver is a computational server in the network. It performs the following functions:

- registering with the aserver;
- acquiring the engine;
- initializing the engine with stored settings;
- waiting for incoming tasks;
- acquiring code and data for tasks;
- unpacking/unfolding incoming tasks;
- responding to healthcheck quiries;
- unregistering at the end of service.

Bserver requires admin rights to be installed and operate.

Engine

- Stata is not the engine. Stata's installation and license are not sent over the network.
- The engine is a bridge component (library) that interfaces the parallelization system with a particular task performer, which is Stata.
- Each bserver is responsible for acquiring, unpacking and attaching the engine upon successful registration with the aserver, and by doing that: contracting to perform the requested tasks.
- It is not even required that each bserver employs the same performer. For example, if you believe Stata is backward compatible, then some of the performers might be Stata 12s, and some might be Stata 13s, and the whole cluster work as one fast Stata 12.
- Each computational server must run at least one bserver component. It can run multiple bservers. In fact it can run multiple bservers belonging to different clusters.

Engine

- It is up to the engine to decide how to exploit a performer. For example, it may decide to launch a new instance of Stata for every task in batch mode (safer), or keep one always on and send execution commands via OLE Automation (faster).
- Aserver provides the same engine to all bservers. However its implementation can be sophisticated enough to react to the bservers' particular situation and it's behavior controlled by various local parameters.

Communication subsystem

There are two communication channels in the system: 'thin' and 'thick':

- thin is implemeted as exchange of [short] messages directly between the nodes through TCP/IP;
- thick is implemented as exchange of files through a shared storage;

Shared resources

- A network connected storage provides source for data and code.
- Each cluster has a function to inform the client about the location of such shared storage (in our experiments a network drive mounted on all the network machines).
- Each job submission may be accompanied by a zip archive, containing a collection of `*.ado` and other files necessary for the execution of the tasks.
- There is no automatic way to decide, which files are necessary. The person preparing the job should determine the set of files.
- Each bserver acquires a copy of data and code from the shared storage upon receiving <u>the first task</u> of the job.
- Locally cached copies of shared resources are flushed when the job is complete (last task of the job is done).
- Bservers are logically isolated and don't know about each other: 2 bservers running on the same computer currently require 2 data transfers.

Jobs, tasks, parameters

- Jobs are collections of tasks (at least 1, usually hundreds). Tasks are independent.
- Tasks consist of parameters.
- There are 3 parametes: command, data, and results

Parameters

Do what? (command)
Do with what? (data)
What do you want? (results)

- Technically there are other parameters of task, such as instance number in a batch, seed, etc

Examples

A few examples of outsourcing computing power to grid for an 8-core machine with StataMP4 license and another single core performer X.

1. One bserver instance, outsourcing Stata. 4 cores may be occupied by the grid, 4 are always available to the local tasks.
2. Two bserver instances, one outsourcing Stata with 4 cores to one cluster, and one outsourcing X with a single core to a different cluster. Three cores are always available for local tasks.
3. Five bserver instances, one outsourcing Stata with 4 cores to one cluster, and 4 outsourcing X to a different cluster.
4. Ten bservers: two outsourcing Stata with 4 cores to one cluster and 8 outsourcing X to a different cluster. (Extremely busy server). Performers will compete for power, but can make sense if the nature of them is different, e.g. one needs lots of CPU, another is mostly IO operations.

Reading StataMP report

Table 1. Stata/MP performance, command by command

Command	Speed relative to a single core[a]: Number of cores: 2	4	8	16	Percentage parallelized[b]
mleval	2.0	4.0	8.0	15.7	100
mleval, nocons	2.0	4.0	7.9	15.6	100
mlmatbysum	1.7	3.3	6.3	11.6	98
mlmatsum	2.0	3.9	7.8	15.3	100
mlogit	1.6	2.3	2.9	3.4	75
mlsum	1.8	3.4	4.9	8.0	94
mlvecsum	2.0	3.8	7.3	13.2	99
mprobit	1.0	1.0	1.0	1.1	5

Fragment from: `http://www.stata.com/statamp/statamp.pdf` page 25.

More examples

In some cases it can make sense to underuse the available Stata licenses if the expectation is that the tasks are using commands that are difficult to parallelize.

- Four bserver instances: two outsourcing Stata with 3 cores to one cluster, and two outsourcing X to a different cluster. (Stata licenses are underused in this case, but the overall performance may be higher than running 1 Stata instance with 4 cores).
- Eight bserver instances: each outsourcing Stata with 1 core to the cluster (for `mprobit` lovers).

X doesn't have to be a statistical package or have anything to do with the computations, it may just as well be a utility. For example, *Jeph Herrin* describes his workflow of creating 1500 reports in Stata and converting them into PDFs with Adobe: `http://www.statalist.org/forums/forum/general-stata-discussion/general/86543-decreasing-graph-resolution` // in this case X can be a txt to PDF converter. Other similar tasks include preparation of various graphs.

Job Example

The whole job is then a plan, such as:

job example

"Z:\data\country.dta"	econsimulate empl gdp migr, shock(0.05)	e(infl) e(xchrate)
"Z:\data\country.dta"	econsimulate empl gdp migr, shock(0.10)	e(infl) e(xchrate)
"Z:\data\country.dta"	econsimulate empl gdp migr, shock(0.15)	e(infl) e(xchrate)
"Z:\data\country.dta"	econsimulate empl gdp migr, shock(0.20)	e(infl) e(xchrate)

Naturally, the command may include parameters, that vary between the tasks, such as parameter *shock* above.

Currently all tasks receive the same dataset parameter - each job can have one dataset. In the future, a job would be able to have multiple datasets attached, and each task would be able to select one of them.

Queue operation

- Jobs are registered in the catalogue and their tasks are posted into a queue;
- Tasks are posted to bservers sequentially, using the FIFO principle;
- Note, that this does not imply the jobs will be completed in the same order as submitted;
- Aserver maintains the status and monitors the health of all bservers;
- In some cases a bserver may fail to perform a task (e.g. power failure); when this is detected, the task is reassigned to another bserver;
- Computational servers that failed to perform a task are assigned status 'dead' and no longer get any assignments;

Bserver settings

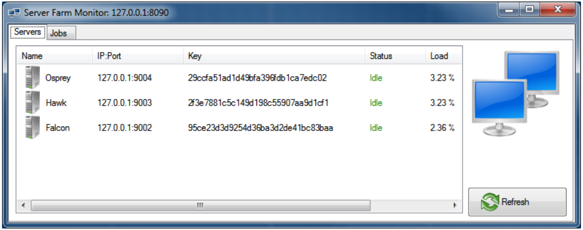
Behavior of the bserver is controlled by a number of parameters saved in a local settings file, they include:

- server name;
- address of the aserver (ipport);
- own address (ipport);
- location of the performer (i.e. path to Stata);
- location to use for temporary files;
- number of cores to be used;
- other parameters.

A particular engine may decide how to use these parameters, for example it may ignore the cores settings in case the performer does not support it.

Monitoring and administration

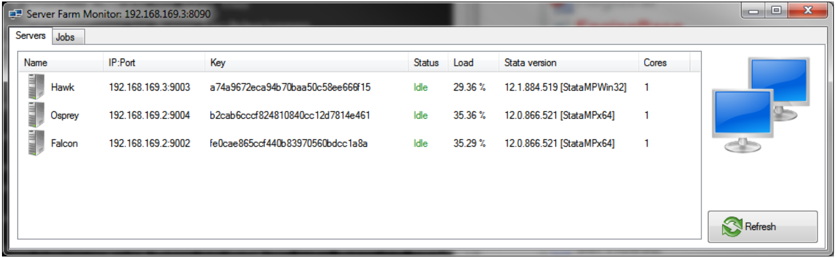
FarmMonitor component provides an overview of what's going on in the cluster. It connects to the aserver and collects statistics on the servers and jobs that are currently queued.



Here three servers run on the same (local) machine.

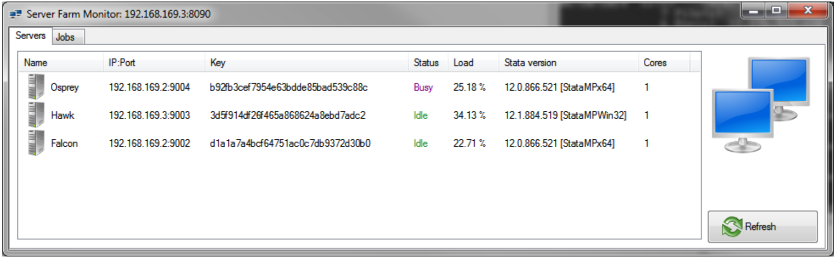
Using this interface the administrator can kick out a server or schedule a maintenance period, during which the server will not receive new tasks.

Monitoring and administration

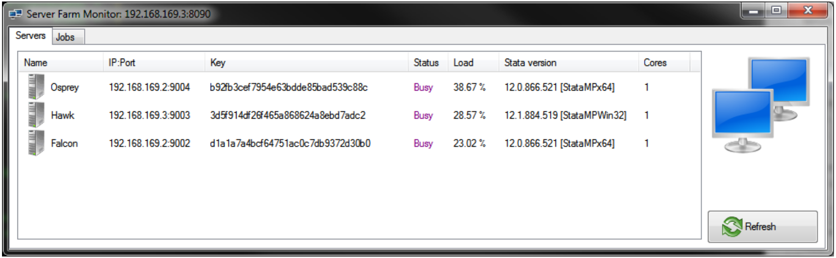


Here three bservers run on two different machines, all servers idle.

Monitoring and administration

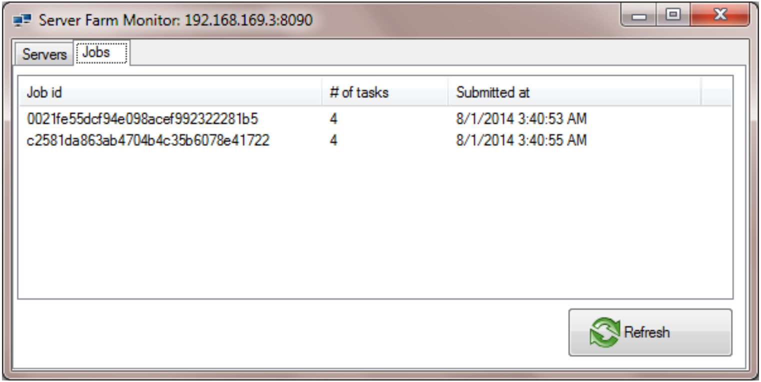


Monitoring and administration



Jobs queue

Similarly the list of jobs in the queue is presented, along with number of tasks and percent complete.



Client

There are two types of clients for the system:

- human oriented - have interface for creating sets of tasks for jobs to follow a particular template.
- automation oriented - provide a possibility to create a job programmatically, say, from Stata, submit to the cluster, wait for the results, and bring the results in when they are ready.

Client

To submit a job from Stata to the cluster one would use the `hpccmd2` command:

```
hpccmd2 tasksfile, data(datafile) cluster(clusterline)
session(sessionfile)
```

where

- *tasksfile* is the list of individual tasks that need to be performed by the cluster;
- *datafile* is the datafile to be processed by the tasks; if your file is not already in the shared location, write the current memory content to a tempfile (*in the shared location*) and pass that name here;
- *cluster* is the cluster connection line, indicating address of the aserver, for example: `"DEMO 192.168.169.99:8888"`
- *session* is a zip file containing a collection of ado files necessary for the tasks to be run;

Tasks file

A task file can be generated:

- programmatically, for bootstrap with command hpccmd_gen_boots
- programmatically, for simulations with command hpccmd_gen_plan
- programmatically, for other types of jobs with a custom script
- manually, by directly creating a list of tasks using a simple plain text editor, e.g. `notepad.exe`;

Explosive substitution

```
hpccmd_gen_plan, planfile("c:\temp\testplan.txt") ///
                 keyfile("c:\temp\testkey.txt") ///
                 cmd("mycmd %X% %Y%, %Z%") ///
                 datafile("Z:\nlsw88abc.dta") ///
                 results("r(product)") ///
                 params("X Y Z") ///
                 p_X("1(1)10") ///
                 p_Y("1(1)10") ///
                 p_Z("@MUL DIV SUM SUB")
```

Plan

```
Z:\nlsw88abc.dta          mycmd 1 1, MUL  r(product)
Z:\nlsw88abc.dta          mycmd 1 1, DIV  r(product)
Z:\nlsw88abc.dta          mycmd 1 1, SUM  r(product)
Z:\nlsw88abc.dta          mycmd 1 1, SUB  r(product)
Z:\nlsw88abc.dta          mycmd 1 2, MUL  r(product)
Z:\nlsw88abc.dta          mycmd 1 2, DIV  r(product)
Z:\nlsw88abc.dta          mycmd 1 2, SUM  r(product)
Z:\nlsw88abc.dta          mycmd 1 2, SUB  r(product)
Z:\nlsw88abc.dta          mycmd 1 3, MUL  r(product)
Z:\nlsw88abc.dta          mycmd 1 3, DIV  r(product)
Z:\nlsw88abc.dta          mycmd 1 3, SUM  r(product)
Z:\nlsw88abc.dta          mycmd 1 3, SUB  r(product)
Z:\nlsw88abc.dta          mycmd 1 4, MUL  r(product)
Z:\nlsw88abc.dta          mycmd 1 4, DIV  r(product)
Z:\nlsw88abc.dta          mycmd 1 4, SUM  r(product)
```

Key

X	Y	Z
1	1	MUL
1	1	DIV
1	1	SUM
1	1	SUB
1	2	MUL
1	2	DIV
1	2	SUM
1	2	SUB
1	3	MUL
1	3	DIV
1	3	SUM
1	3	SUB
1	4	MUL
1	4	DIV
1	4	SUM
1	4	SUB
1	5	MUL
1	5	DIV
1	5	SUM

Generating tasks file:

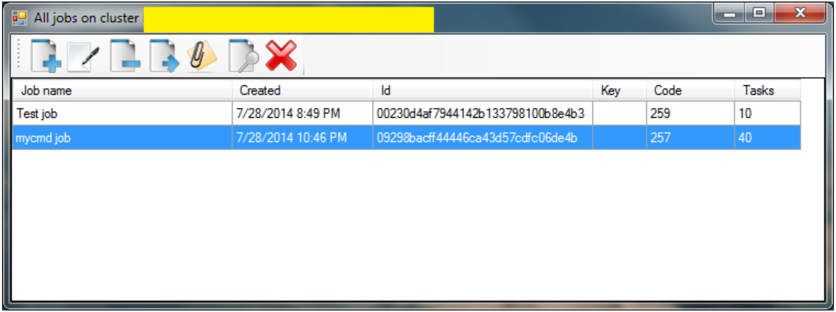
For example, the following command

```
tempfile tmp
hpccmd_gen_boots using "Z:\nlsw88.dta",
cmd(mycmd race wage tenure age)
results("e(chi2) e(ll)") reps(12) saving("`tmp'")
```

creates a temporary task list that can be sent to cluster:

```
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
Z:\nlsw88.dta    mycmd race wage tenure age    e(chi2) e(ll)
```

Client program for manual jobs submission



Client program for manual jobs submission

Creating a new job