

Parallel Computing in MARCC

Mateo Velásquez-Giraldo

Workshop for the Department of Economics
Johns Hopkins University
February, 2021

What is parallel computing?

Parallel computing is a problem-solving technique in which

- ▶ A problem is divided into parts.
- ▶ Each part's solution does not depend on the others'.
- ▶ Multiple parts are solved *simultaneously* by *different* processing units.

Example: baking 400 muffins in 1 vs. 2 vs. 4 ovens.

The gains are easy to understand:

- ▶ Get many processing units (n).
- ▶ Make them all work simultaneously.
- ▶ Speed up your solution by a factor of n .

... not so fast.

Caveats — What *can* you parallelize?

Idea from Economics:

$$V_t(m_t) = \max_{0 \leq c_t \leq m_t} u(c_t) + V_{t+1}(m_t - c_t), \quad t = 1, \dots, T.$$

Can we solve each t in a different processor *simultaneously*?

No! notice we can't find $V_t(\cdot)$ until we have found $V_{t+1}(\cdot)$.

Things we *could* do:

- ▶ If we have heterogeneous agents, solve each type in its own processor.
- ▶ If we are using a grid for m , given $V_{t+1}(\cdot)$
 - ▶ Solve $V_t(m)$ for different m 's simultaneously in different processors.
 - ▶ Join the solutions to form $V_t(\cdot)$, move on to $V_{t-1}(\cdot)$.

Caveats — What *should* you parallelize?

Example: 400 muffins again. Should you simultaneously use 400 ovens?

At some point it becomes impractical.

- ▶ Coordinating parallel tasks has some costs. E.g.
 - ▶ Setting up the processing units.
 - ▶ Dividing the problem.
 - ▶ Transferring information to and from the units.
 - ▶ Combining the independent solutions.
- ▶ This is often called *overhead*.
- ▶ Overhead can easily make the parallel version of your solution slower than the sequential version!
- ▶ Therefore, test and optimize your parallelization strategy.

MARCC — MD Advanced Research Computing Center

- ▶ Computing facility.
- ▶ Funded by the State, managed by JHU and UMD-CP.
- ▶ Two purposes:
 - ▶ High performance computing. > 23,000 cores.
 - ▶ Reliable data storage.

Its resources are organized in *Partitions*

- ▶ You select which partition to send your jobs to.
- ▶ Each has its own characteristics:
 - ▶ Type of processor, number of available cores.
 - ▶ GPUs.
 - ▶ Time allowances.
 - ▶ Memory.
- ▶ Find detailed descriptions [here](#).

MARCC — Resources

Example, each node in parallel partition

- ▶ 24 cores, 128GB RAM.
- ▶ Type of processor: Intel Haswell, 12-core, 2.5GHz, 30MB cache (2x).
- ▶ Can be used for up to 72 hours.

Software:

- ▶ Packaged in “modules” that you must load.
- ▶ See them with command: `module avail`.
- ▶ Matlab, Mathematica, Python, R, Julia, gcc (C++, Fortran), and many more.

Availability:

- ▶ Currently Econ. Dep. has 1,000 CPU-Hours/quarter. **Free**.
- ▶ Going mostly unused.

Getting Access to MARCC

- ▶ MARCC's usage is regulated through *allocations* of resources.
- ▶ "PIs" control who can access their allocations.
- ▶ Our allocation's PI (as of 2021) is Prof. [Jonathan Wright](#).

Steps

1. Contact Professors Hwang, Moffitt, or Wright to request access.
2. Make sure Prof. Wright is aware that you will request an account.
3. Request an account [here](#).
4. Wait for an approval email from MARCC.

Connecting to MARCC

MARCC uses the *Secure Shell (SSH)* protocol.

Options:

▶ **Command ssh in a terminal:**

- ▶ Available by default in Unix and MacOS.
- ▶ Available in Windows ≥ 10 . Might need [setup](#).
- ▶ Find instructions [here](#).

▶ **SSH client:**

- ▶ A program dedicated to ssh connections. [List of available ones](#).
- ▶ For Windows: **PuTTY** is the most popular. I like **Bitvise**.

In both cases you will need [Google Authenticator](#):

- ▶ Register the app with your MARCC account. [Instructions](#).
- ▶ Use it to get your `Verification` code.

Example — Terminal

Command:

```
ssh -X login.marcc.jhu.edu -l user@jhu.edu
```

```
mvelasq2@jhu.edu@bc-login02:~
mv77@LAPTOP-DAU3RLPU:~$ ssh -X login.marcc.jhu.edu -l mvelasq2@jhu.edu
Verification code:
Password:
Last login: Mon Dec 21 17:17:40 2020 from 172.16.0.9
-----
Blue Crab Project: jwrigh10
=====
  1,000 shares (hours/quarter)
    0 used (hours) since 2021-01-01
    0 used recently (hours)
  1,000 available (hours)
  100% AVAILABLE

      USER      USED    RECENT
* mvelasq2@jhu.edu    271     205

see this link for more info:
https://marcc-hpc.github.io/esc/common/allocations
-----
[mvelasq2@jhu.edu@bc-login02 ~]$
```

Example — Client (Bitvise)

Configure your client as in the left panel.

The image shows two windows from the Bitvise SSH Client. The left window is the 'Default profile' configuration panel, and the right window is a terminal session.

Default profile configuration:

- Server:** Host: login.marcc.jhu.edu, Port: 22, Enable obfuscation:
- Authentication:** Username: YOUR_ID@jhu.edu, Initial method: keyboard-interactive, Submethods: bsauth,pam,totp,pw,pv, Elevation: Default
- Kerberos:** SPN: [empty], GSS/Kerberos key exchange: , Request delegation: , gssapi-keyex authentication:
- Proxy settings:** Proxy settings, host key manager, Client key manager, Help
- Log:**
 - 14:57:46.582 Received keyboard-interactive info request.
 - 14:57:53.391 Attempting keyboard-interactive authentication. Sending info response.
 - 14:57:53.485 Authentication completed.
 - 14:57:53.518 Host key has been erased from the global database. Algorithm: Ed25519, size: 255 bits, SHA-256 fingerprint: ZiawYeg/FjIeFCdeN9TZUX+6GTZLas2PV5nIzIEpU.
 - 14:57:53.522 Host key has been erased from the global database. Algorithm: RSA, size: 2048 bits, SHA-256 fingerprint: 6vz247GeFACzkd6FwI3MvtgRdJWGabmMvHHEEz0IM.
 - 14:57:53.522 Host key synchronization completed with 2 keys erased from global settings. Number of keys received: 2, rejected: 1.
 - 14:57:53.548 Terminal channel opened.
 - 14:57:53.548 SFTP channel opened.
 - 14:57:53.549 SFTP channel opened.
 - 14:59:10.107 Changes made on the Login tab will have no effect over the established connection, even in case of auto reconnection.

Terminal session:

```

Blue Crab Project: jwrigh10
=====
1,000 shares (hours/quarter)
 0 used (hours) since 2021-01-01
 0 used recently (hours)
1,000 available (hours)
100% AVAILABLE

      USER      USED      RECENT
* mvelasq2@jhu.edu      271      205

see this link for more info:
https://marcc-hpc.github.io/esc/common/allocations

[mvelasq2@jhu.edu@bc-login03 ~]$
  
```

SFTP window:

The SFTP window shows local files on the left and remote files on the right. The remote files list includes .cache, .emacs.d, and java.

Transferring files — Globus

MARCC recommends a system called [Globus](#). Instructions [here](#).

Details:

- ▶ You need to set-up “endpoints”.
- ▶ Your computer:
 - ▶ Endpoints → “Create a personal endpoint”.
 - ▶ Download and install the client.
 - ▶ Will appear at Endpoints/Administered By You.
- ▶ MARCC:
 - ▶ Appears as `marcc#dtn`.
 - ▶ Select it and log in with your marcc credentials.
- ▶ Select both in File Manager and use the interface.
- ▶ You will get emails when transfers complete.

Examples — Matlab

- ▶ Different languages do parallel computing differently.
- ▶ Basic examples in R, Matlab and Python [here](#).

... However, they share the same structure. Some subset of:

1. Create a pool or cluster.
2. Export the tools workers need. E.g., data, self-defined functions.
3. Execute the parallel operations.
4. “Paste” and format their results.
5. Shut down pool.

Lets go through the [Matlab example](#).

SLURM — Workload Manager

Would be a mess to have everyone running jobs at the same time.

That is why **SLURM** exists.

- ▶ It is a resource administrator.
- ▶ You don't *just run* your jobs.
 - ▶ You write your instructions as a **bash** script.
 - ▶ Specify what resources you would like to use.
 - ▶ Send the request to SLURM.
 - ▶ It will assign your resources and run your script.
- ▶ Pros: order and predictability!
- ▶ Cons: there is a queue for resources.

SLURM Scripts

- ▶ You configure your request with lines starting with `#SBATCH`.
- ▶ Place them at the top of your bash script.

Some examples:

```
#SBATCH --job-name :
```

```
#SBATCH --partition : MARCC partition. Type of resources.
```

```
#SBATCH --time : maximum time allowed for your job.
```

```
#SBATCH --cpus-per-task : CPU cores to allocate to each task.
```

[More here.](#)

Confusing: nodes vs cpus vs cores vs tasks. [Best guide I've found.](#)

Let's see an [example script for our Matlab job.](#)

Hopefully we will do this part live!

For posterity:

- ▶ Get familiar with [basic Linux commands](#): `cd`, `ls`, `cat`...
- ▶ Navigate to your script.
- ▶ Submit it with `> sbatch YOUR_SCRIPT.sh`

After that:

- ▶ You can see your jobs' status using command `sqme`.
- ▶ This will also give you your job's "id".
- ▶ `seff id` details your job's use of resources.
- ▶ `scancel id` cancels your job.

More [here](#).

You can log off. Your job won't be stopped.

Additional resources and troubleshooting

- ▶ MARCC's help desk: marcc-help@marcc.jhu.edu.
- ▶ MARCC's website: <https://www.marcc.jhu.edu>.
- ▶ MARCC's own introductory session:
 - ▶ Go to [this link](#).
 - ▶ Click on "This video".
 - ▶ Log-in with your JHU credentials.
- ▶ [MARCC's guide for Matlab](#).
- ▶ [Course](#) from Princeton.
- ▶ My [Github repository with Matlab, R, and Python examples](#).
- ▶ My own email: mvelasq2@jhu.edu.

Thank you and best of luck!