Parallelism Day

Avi Kulkarni

Dartmouth College

July 2, 2022

Goals

Understand how to use parallelism within magma
Understand how parallelism works in magma

Agenda

This Primer
Exercises + Workshop time

Resources

```
git clone https://github.com/a-kulkarn/magma-parallel-cookbook.git
```

Why Parallelize (in magma)?

Most loops

```
   [Do_Something(x) : x in S]
```

are sped up considerably.

Why do this *in magma*?

- Automatic management to avoid idle CPUs
- Magma objects passed between subprocesses

Some intrinsics (builtin magma functions) have native parallelism support.

```
SetNthreads(n);
SetGPU(true);    // Doesn't work on MacOS.
```

Things Improved by `SetNthreads(n)`:

- Matrix multiplication
- Groebner basis calculations
- Short vector enumeration in lattices

As well as anything dependent on the above.

Warning 1: Your milage per machine may vary.

```
SetNthreads(N); // Where N  = 1, 4
X := Random(MatrixRing(GF(5), 10000));
time P := X*X;
```

Macbook pro, M1 chip, 8 cores	$N = 1$	5.450 (s)
Macbook pro, M1 chip, 8 cores	$N = 4$	6.870 (s)
Toby, AMD Ryzen Threadripper, 48 cores	$N = 1$	5.780 (s)
Toby, AMD Ryzen Threadripper, 48 cores	$N = 4$	2.120 (s)

Warning 2: Threads $\neq$ processors

```
SetNthreads(N); // Where N = 1, 40, 1000, 4096
X := Random(MatrixRing(GF(5), 10000));
time P := X*X;
```

Toby, AMD Ryzen Threadripper, 48 cores

$N = 1$	5.780 (s)
$N = 40$	0.820 (s)
$N = 1000$	1.380 (s)
$N = 4096$	3.740 (s)

User-Implemented parallelism in Magma

Don't use Magma's User-Implemented parallelism on a multi-user machine

(There is a security issue)

User-Implemented parallelism in Magma

- One process opens a channel

```
socket := Socket(...);
DistributedManager(socket, jobs); // Dangerous
```

- Other processes connect to this channel

```
the_results := DistributedWorker(host, port,
    your_function);
```

- Input/Output is transmitted over the channel (serialization)

Directory: cookbook/lecture-examples/

```
// collatz_manager.m
socket := Socket(: LocalHost := "localhost",
                   LocalPort := 10000);
for i in [1..10] do
    System("magma collatz_worker.m &"); // Launch and detach the workers.
end for;
DistributedManager(socket, [1..10]);
delete socket;
```

```
// collatz_worker.m
host := "localhost"; // Same as above.
port := 10000;

function collatz_info(one_arg)
    ...
end function;

DistributedWorker(host, port, collatz_info);
quit;
```

Important takeaways

- Don't need the `SetNthreads` call.
- Manager and workers are separate processes
  - Source code needs to be in separate files.
  - No shared memory
  - Worker is run as a script. (Or interactively)
  - Worker needs to quit.
- Host and port information must be the same
- Only one argument can be passed.

Restrictions of Magma’s parallelism:

- No shared memory between processes.
- Transmitting of objects not faithful.
- Transmitting user defined objects very restrictive.

Warning 3: Orphans

An orphan process is a process whose parent process has terminated.

- Easy to accidentally create when parallelizing code
- Eat resources or block ports

Example: How to "rescue" orphans

Directory: cookbook/lecture-examples/

```
$ magma dyad_institute.m // Spawns children and exits
```

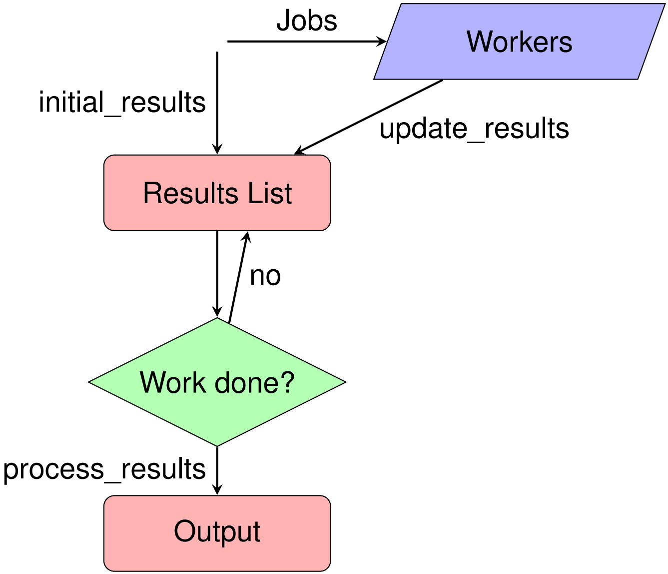
Catch-all tool:

```
$ top
$ htop // Even better if you have it.
```

Options for Distributed manager

```
DistributedManager(server::IO, inputs::List) -> .
[
initial_results, // Initial value for accumulator
update_results,  // Update on intake
process_results, // Post-process
group_tasks,     //
update_group,    //
incomplete_group //
]
```

Options without Task Groups



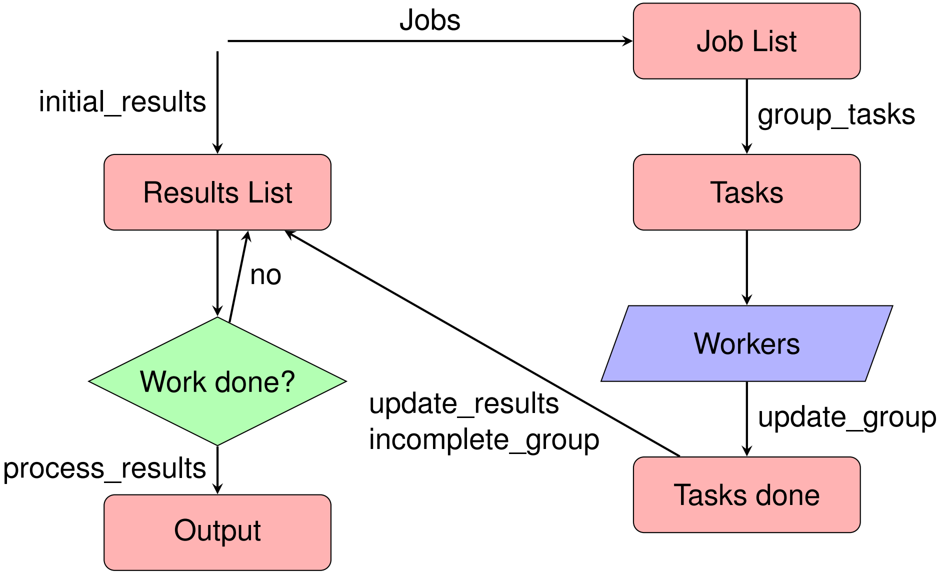
Task Groups

- Organize a Job into parallelized sub-jobs (task groups)
- Common use cases:
  - Try many things, return first-to-finish
  - Searching in a box
  - Dividing a computation among primes

```
DistributedManager(server::IO, inputs::List) -> .
[
initial_results, // Initial value for accumulator
update_results,  // Update on intake
process_results, // Post-process

group_tasks,     // From jobs, create task groups
update_group,    // Update task when workers finish
incomplete_group // Handle incomplete tasks.
]
```

Options with Task Groups



Directory: cookbook/lecture-examples/

```
// squares_manager.m
break_into_local_tasks := function(pairs)
    a, n := Explode(pairs); // Unpack
    default_val := true;
    ...
    // Divide into tasks
    facts := Factorization(n);
    return default_val, [<a, p[1]^p[2]> : p in facts];
end function;

combine_local_info := function(item, task, tresult,
    gresult)
    ...
end function;

...
issquare_input := [<-1, n> : n in [1..100]];

results := DistributedManager(socket, issquare_input :
                    group_tasks := break_into_local_tasks,
                    update_group := combine_local_info);
```

Summary (Thanks!)

- Parallelizing code is *sometimes* faster.
- Some Magma built-ins support parallelism.
- Distributed computing can use one or many machines.
- User-implemented parallelism is based on manager-worker model.
- The `DistributedManager` class can implement different flavours of this model.
- `top, htop` can get you out of trouble.
- Examples have been provided in the cookbook.

Magma also allows for a lower level interface (managing the I/O channel directly). We did not cover this today.

Warning!

The following slide is dangerous.

On toby

```
$ hostname // Prints "toby"
$ magma
> socket := Socket(: LocalHost:="toby",
                     LocalPort:=10000);
> DistributedManager(socket, [1..10]);
```

Meanwhile... on doob

```
// collatz_worker.m
host := "toby"; // Name of host
port := 10000;  // Note: Receiving Port needs to be open
                //       in the firewall.

function collatz_info(one_arg)
    ...
end function;

DistributedWorker(host, port, collatz_info);
quit;
```

```
$ magma collatz_worker.m // Interacts with toby
```