Parallel computing in Julia

Jasper Boomer

August 2, 2014

Table of Contents

Background(1)

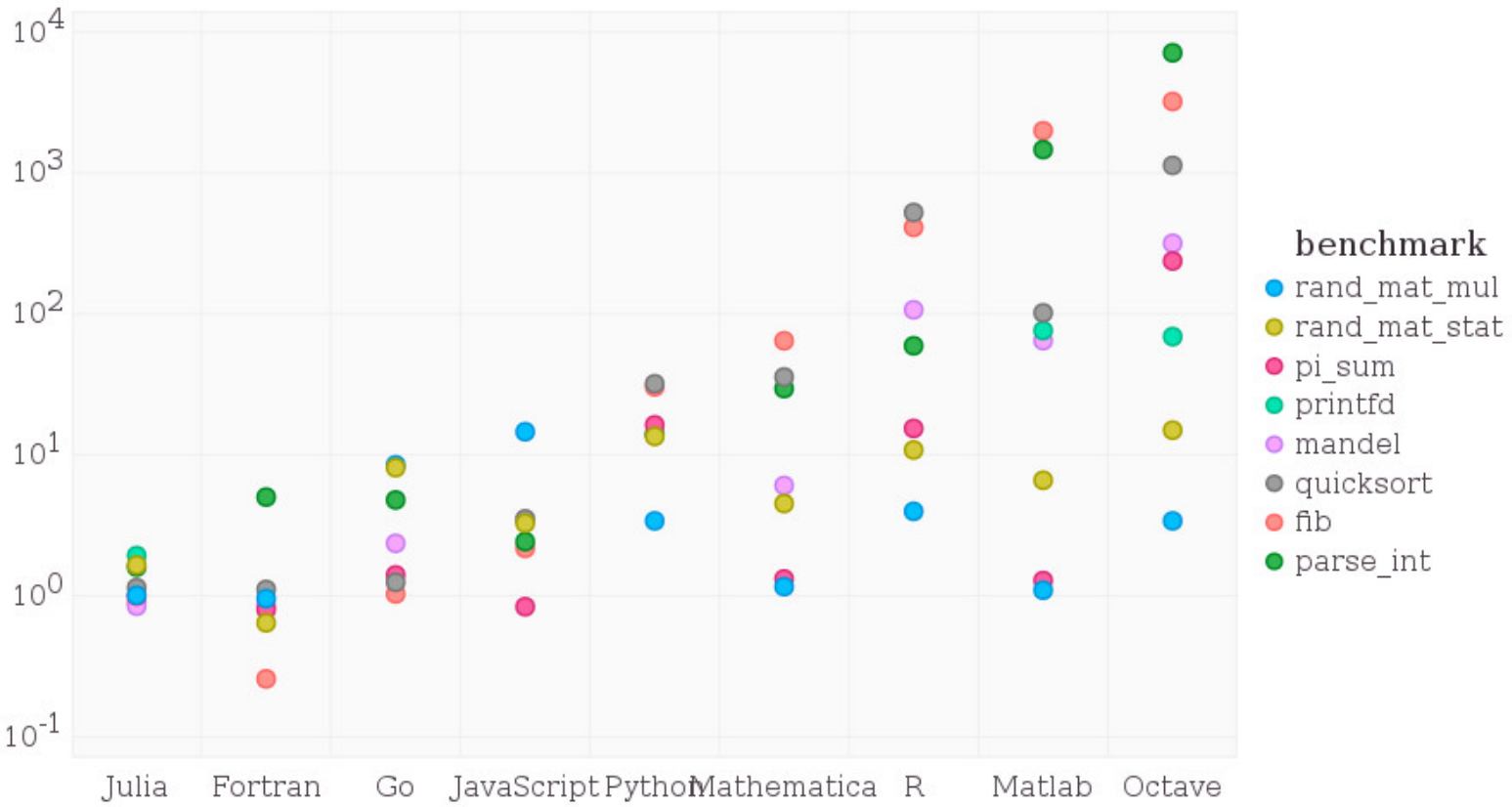
" We want a language thats open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language thats homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled."

Julia developers on "Why we created Julia"

Background(2)

Julia is:

- Free & Open source (MIT licensed)
- Development started in 2009, v0.1 in 2012
- General purpose high-level dynamic programming language
- Designed for parallelism & distributed computation
- Core written in C/C++, STL in Julia
- JIT Compiled
- Optimized for speed of calculation



1

[1] Speed relative to C

Some syntax

Examples

```
julia> [i*2 for i=1:5]
[1 4 6 8 10]

julia> function printargs(args...)
         for a in args
             println(a)
         end
       end

julia> map(x->(x%3==0), [3 4 12])
[true, false, true]
```

Macros

Expressions are Julia objects. These can be modified and then run.

@time macro

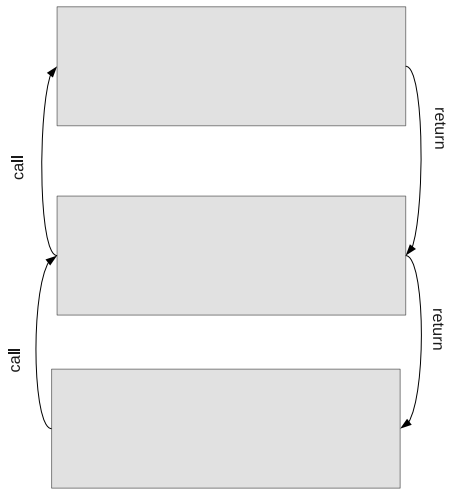
```
macro time(d::Expr)
  local t0 = time()
  local val = $d
  tocal t1 = time()
  println("elapsed time: ", t1-t0, " seconds")
  return val
end

julia> @time rand(100,100)
elapsed time 0.00679384 seconds
100x100 Array{Float64}
```

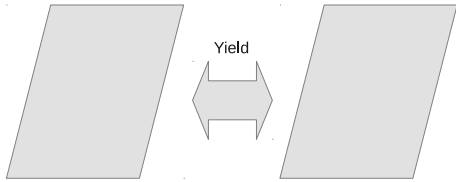
Table of Contents

Tasks(1)

Functions (Sub-routines)
Call / Return



Tasks (Co-routines)
Yielding to other tasks (saving state)



Tasks(2)

Example: Integrator

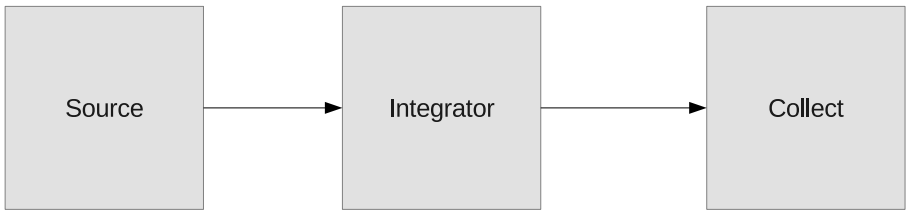


Table of Contents

Parallel implementation in Julia

- No native threads (yet?)
- Uses tasks/coroutines for scheduling
- Message passing interface for data communication
- Implementation of message passing is one-sided

Worker processes

Syntax:

```
julia -p N [filename]
```

- Launches `N` worker processes
- Runs `filename` or interactive session
- Workers are numbered $2 : N + 1$
- Can run on multiple cores or cluster of computers (specified using a machinefile)

Remote calls & references

Executing statements on specific worker processes

```
julia -p 2
julia> r = remotecall(2, sin, pi/4)
RemoteRef(3,1,2)

julia> fetch(r)
0.7071067811865475

julia> remotecall_fetch(2, myid)
2

julia> @spawn cos(pi/4)
RemoteRef(3,1,3)
```

Table of Contents

MapReduce(1)

`pmap(function, collection)`

Performs `function` on `collection`. Uses a 'feeder' task for each worker process to serve data, while waiting for result the task yields to next feeder task until work is complete.

Example: list of expressions

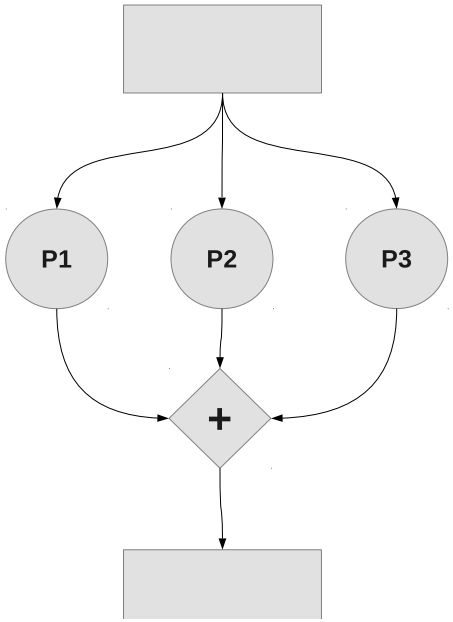
```
julia> pmap(eval, {:(4+4), :(4*4), :(7-2), :(6/2)} )
[8, 16, 5, 3]
```

MapReduce(2)

`@parallel` : Macro for MapReduce

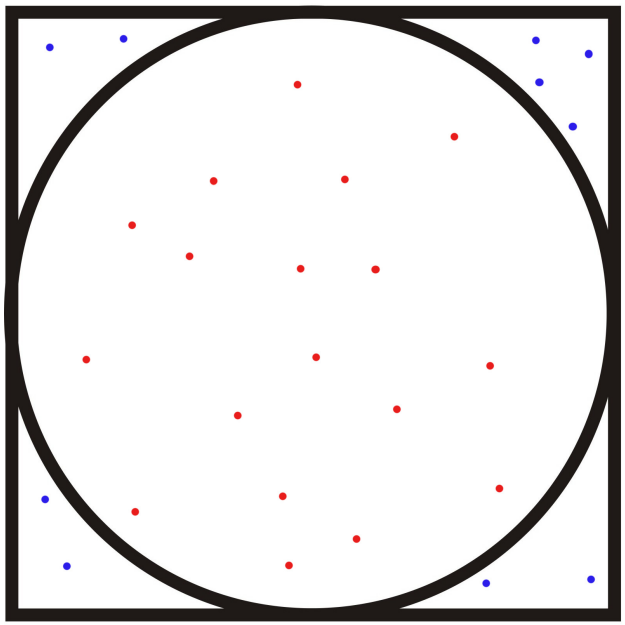
Example: coin toss

```
heads =
@parallel (+) for i=1:10000000
  randbool()
end
```



MapReduce(3)

Example:
A slice of π



Generate random points from uniform distribution, get π from ratio of points in the circle.

$$\begin{aligned} A_{rect} &= (2r)^2 = 4r^2 \\ A_{circle} &= \pi r^2 \\ \pi &= 4\frac{A_{circle}}{A_{rect}} \end{aligned}$$

Table of Contents

Syntax

Constructor: `DArray(init, dims[, procs, dist])`

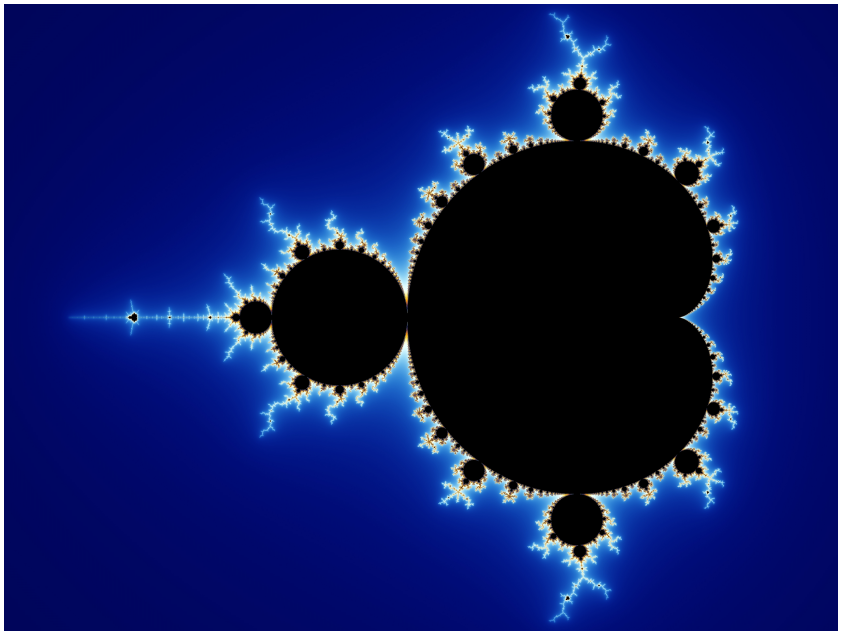
Example with 8 workers:

```
julia> d = DArray(I->rand(length(I[1]),length(I[2])),
(80, 80),  #dimensions
[2:9],     #worker processes
[4,2]);    #distribution

julia> remotecall_fetch(2,localindexes,d)
(1:20, 1:40)

julia> remotecall_fetch(7,localindexes,d)
(21:40, 41:80)
```

Example: Mandelbrot Set



$$
\begin{aligned}
z \in \mathbb{C}: & \\
z_0 &= z \\
z_{n+1} &= z_n^2 + z
\end{aligned}
$$

Set of c for which orbit of z around 0 remains bounded. Colours indicate escape time.

Table of Contents

Links

- http://julialang.org/
- http://learnxinyminutes.com/docs/julia/
- https://github.com/jboomer/parallel-julia/
- http://www.dabeaz.com/coroutines/

Table of Contents