Parallelism

October 24, 2016

1 Levels of Parallelism

Julia offers 3 levels of a parallelism:

- SIMD: Single-Instruction, Multiple Data
- Lets a single instruction calculate multiple things at the same time
- Ex: Multiple multiplications in one CPU instruction
- Threads
- Shared Memory Parallelism
- Small overhead, most share the same memory (multicore processors)
- Processes
- Overhead is much higher
- Can distribute work amongst multiple computers

In addition, Julia can be used with GPGPUs and Xeon Phis

1.0.1 Lowest Level: SIMD

- SIMD parallelization is "processor-level parallelization"
- It uses the processor's AVX (Advanced Vector Extensions)
- The compute cores can process more than 1 number at a time!
- Add @simd to the beginning of an inner loop
- The values must be able to be re-ordered

1.0.2 Middle Level: Threads

- Thread parallelization is shared-memory parallelization
- Threads share the memory on the same computer
- Use Threads.@thread at the beginning of a threadable loop

1.0.3 Highest Level: Processes (Distributed)

- Task parallelization is a general form of parallelization
- Much easier than MPI, though mostly the same capabilities
- Can be used on multiple compute nodes on an HPC
- Memory is not shared, items have to be distributed and passed
- Either open julia with -p numberOfProcs or use addprocs(n)

- One process is the control process, the others are workers
- Use @parallel, pmap, etc. to distribute work to the workers

1.0.4 GPU Example

http://www.stochasticlifestyle.com/julia-on-the-hpc-with-gpus/
http://www.stochasticlifestyle.com/multiple-gpu-on-the-hpc-with-julia/

1.0.5 Xeon Phi Example

http://www.stochasticlifestyle.com/interfacing-xeon-phi-via-julia/