Glossary for JuliaCon ParallelComputing Workshop 2016

- @async - execute a block of julia code in a separate task. Returns immediately without waiting for completion (see async map to map multiple asyncs)
- @sync - wait till all enclosed asynchronous code blocks complete execution
- @parallel for - partition the range of the loop among all workers and execute in parallel. Used with a reduce (e.g. @parallel (+) for) with the assumption the data is mostly computed on the fly or is available through shared memory, and the result is small.
- @spawn - spawn a task for possible remote running
- addprocs - basic command to add processors
- ArrayFire - seamless link between Julia and the GPU ArrayFire library
- Channel - A queue for sharing data between tasks within a Julia process
- Closures - function that captures its environment for later execution (including references to objects on stack)
    - Localizing references - makes copies of references so that it captures the environment at the time of definition.
- Cluster (Julia) - the collection of processes (master + workers) connected to each other in a all-to-all or a master-slave topology
- Communication
    - Inter-task - passing data between coroutines in the same process, usually via Channels
    - Inter-process - passing data between different processes on the same machine or within a cluster of machines
- ComputeFramework - Package for out of core computing that automatically reorganizes code so that only needed pieces are in main memory at a time.
- CPU bound - Tasks which load the CPU
- DAG - Directed Acyclic Graph -- a representation of the dependencies in a computation
- DArray - a distributed global array with one name (e.g. A). Ideally all array functions would some day work on DArray's exactly as they work in serial
- DistributedArrays - the package that allows use of DArrays
- Distributed memory - Memory referenced by all participating processes of a Julia cluster across nodes
- Distributed arrays - see DArray
- Elemental -- state of the art linear algebra package
- Event loop - Multiplexing multiple I/O, timers, etc in a single OS thread of execution
- GPU - Graphical Processing Unit, a popular computational accelerator
- I/O bound - Tasks which have I/O calls either to read/write data or wait for results from external services. Often best called asynchronously
- Iterator - A means of representing a possibly large iteration space without storing every value of the iteration variable in memory, therefore improving performance
- map (on a DArray)

- Mapreduce - The julia function that allows a map and a reduction on a vector in what is known as the “functional form.” The more famous mapreduce tends to be geared to a less regular framework
- Master Process - The initial Julia process which orchestrates work among available workers or external services
- MPI -- the traditional message passing interface, usable with the MPI.jl package
- Out-of-core computation - Ability to perform computation of data too big to fit into RAM by processing it in smaller chunks and aggregating individual results
- pmap - A map implementation that executes the mapping function in parallel on available workers. The data is assumed available on the master process at the start and at the end, therefore it is not recommended if large amounts of data need to move to the processors.
- Remote calls - API to execute functions on remote workers
    - remotecall - execute a function remotely, returns immediately with a reference to the result
    - remotecall_fetch - blocking version of remotecall. Returns the result
- Remote reference - A reference to a Julia object on a different process
- Shared memory - Same block of memory mapped by multiple Julia processes on a single node.
- Star topology - A research project framework that expresses a means of passing results of an embarrassing parallel operation to a master process, and then back again for further embarrassing parallel operations. Also called master-slave topology in Julia
- Tasks (Julia) - lightweight co-routines within a Julia process. All tasks execute on the same OS thread.
- Threads - threads of execution scheduled by the OS.
- Topology - how the workers in a Julia cluster are connected to each other. Can be connected in a all_to_all mesh, master_slave star topology or any other custom topology.
- Worker Process - Offload computational work to external Julia processes. Leverage multiple cores on a node.