Some Notes on Parallel Computing in R

Luke Tierney

Department of Statistics & Actuarial Science
University of Iowa

January 26, 2015

Introduction

Two approaches I have used to parallel computing in R:

- Explicit paralellization
  - uses multiple R processes, possibly on different machines
  - user code explicitly specifies parallel computation
  - available in packages `snow` and `parallel`
- Implicit parallelization of vectorized operations
  - uses threads and shared memory (ideally via OpenMP)
  - requires no explicit user action once enabled
  - available in an experimental package `pnmath`
- The two approaches can be used together.

This talk presents some notes on the two approaches and issues that arise.

- `snow` stands for Simple Network of Workstations.
- Originally designed for utilizing a group of single-core workstations.
- Motivated by COW (Cluster of Workstations) for Python.
- Design goals:
  - support distributed computing
  - simple to use
  - hard to misuse or get into trouble (e.g. no deadlocks)
  - support reproducible parallel simulations

Explicit Parallel Computing in `snow`
Basic Usage

- Basic usage:
  - start a collection of R worker processes
  - initialize parallel random number streams if needed
  - perform one or more scatter-compute-gather operations
  - shut down the workers
- Workers can maintain state between parallel operations

- Two levels of parallel functions:
    - clusterApply assigns a worker to each element.
    - parLapply, parCapply, parRapply assign equal size chunks to each worker.
    - The par functions are implemented using the cluster functions.
- clusterApplyLB provides load balancing at the cluster level.
    - Proper load balanging is not available at the par level.
    - Load balancing complicates making simulations reproducible
- These operations
    - transfer functions and data to the workers
    - wait for the workers to complete and return their results
    - return the merged results
- Data are transmitted as serialized R objects.

Explicit Parallel Computing in `snow`

Some Details

- The design allows for several back end implementations:
  - sockets
  - pvm
  - MPI
  - NWS
- Some back ends allow heterogeneous machines to be used
- Some also provide tools for visualizing parallel computations
- `snow` contains experimental visualization support that works independently of back-end support

Explicit Parallel Computing in `snow`

Possible Directions

- A sequence of scatter-compute-gather operations can express many parallel computations, but not all
- Is it possible to allow more communication while
  - maintaining safety
  - not increasing complexity too much
- The BSP (Bulk Synchronous Parallel) model may be promising
- Some work on BSP in Python may be useful.

Explicit Parallel Computing in `snow`

Some Issues

- A mechanism for out of band communication might help with
  - interrupting or abandoning computations
  - assessing worker progress
  - maybe BSP communication.
- Asynchronous use of worker clusters
- More flexible handling of R level errors on workers
- Fault tolerance
- Cleaner way to maintain state on workers
- Handling RNG streams in the context of load balancing

Implicit Parallelization of Vectorized Operations
Basic Ideas

- R is a vector-oriented language
- For sufficiently long vectors, performing vectorized computations in parallel makes sense.
- Use of parallelization is implicit:
  - the user specifies a maximal number of threads to use
  - R decides internally when to use more than one thread, if allowed
  - no further user specification is needed
- OpenMP can be used to implement this.
- An experimental version is available as a package.
- This may be merged into R this calendar year.

Implicit Parallelization of Vectorized Operations

Some Issues

- Care is needed in accounting for, and minimizing, synchronization costs.
- Work on compilation may help to fuse vector operations and reduce the importance of synchronization.
- A cost model would be useful to allow new functions to be parallelized.
- Work is also needed to allow package writers to use the parallelization infrastructure.
- Reduction operations may also benefit from parallelizaiton, but care is needed to ensure identical floating point results in sequential and parallel reductions.

Useful Additions to R

- Convention for controlling the number of threads used by R.
- Asynchronous events mechanism
- Proper server sockets
- Incomplete data structures
- Distributed data structures
- Better character vector representation