Adventures in HPC and R: Going Parallel

Justin Harrington & Matias Salibian-Barrera

University of British Columbia

The R User Conference 2006

Outline

What is Parallel Computing?

Implementation

Examples

Closing Remarks

What is Parallel Computing?

• From Wikipedia:

  “Parallel computing is the simultaneous execution of the same task (split up and specially adapted) on multiple processors in order to obtain faster results.”

• Two specific situations:
  • A multiprocessor machine
  • A cluster of (homogeneous or heterogeneous) computers.

• R is inherently concurrent, even on a multiprocessor machine.

• S-Plus does have one function for multiprocessor machines.

Goal for today's talk:

To demonstrate the potential of incorporating parallel processing in tasks for which it is appropriate.
What is Parallel Computing?

Example - Multiprocessor Machine

Features:
- Each process has the same home directory.
- Architecture is identical.
- R has the same libraries in the same locations.
- Data is passed through resident memory.

Implementation

- Tasks have to be appropriate.
  - Concurrent, not sequential.
  - It is possible sometimes to take a process inherently sequential, and approximate with a concurrent process e.g. simulated annealing.
- In order to do parallel computation, two things are required:
  - An interface on the O/S that can receive and distribute tasks; and
  - A means of communicating with that program from within R.

Example - Heterogeneous Cluster of Machines

Features:
- Each process may not have the same home directory.
- Architecture might be different.
- R may not have the same libraries in the same locations.
- Data is passed through the network.

PVM & MPI

- There are two common libraries:
  - PVM: Parallel Virtual Machine
  - MPI: Message Passing Interface
- Both are available through open-source for different architectures.
- Which to use? From Geist, Kohl & Papadopoulos (1996):
  - MPI is expected to be faster within a large multiprocessor.
  - PVM is better when applications will be run over heterogeneous networks.
- One of these programs need to be running on the host computer before R can send them tasks.
What is Parallel Computing?

Implementation

- In R there are three relevant packages:
  - Rmpi - the interface to MPI;
  - rpvm - the interface to PVM;
  - snow - a "meta-package" with standardized functions.
- snow is an excellent introduction to parallel computation, and appropriate for "embarrassingly parallel" problems.
- All of these packages are available from CRAN.
- In a environment where the home directories are not the same, the required libraries have to be available on each host.

Commands in snow

Administrative Routines

- makeCluster
- stopCluster
- clusterSetupSPRNG

High Level Routines

- parLapply
- parSapply
- parApply

Basic Routines

- clusterExport
- clusterCall
- clusterApply
- clusterApplyLB
- clusterEvalQ
- clusterSplit

Examples

Closing Remarks
## Implementation

### Commands in snow
- **makeCluster**
  - create a new cluster of nodes
- **stopCluster**
  - shut down a cluster
- **clusterSetupSPRNG**
  - initialize random number streams

### High Level Routines
- **parLapply**
  - parallel lapply
- **parSapply**
  - parallel sapply
- **parApply**
  - parallel apply

### Basic Routines
- **clusterExport**
  - export variables to nodes
- **clusterCall**
  - call function to each node
- **clusterApply**
  - apply function to arguments on nodes
- **clusterApplyLB**
  - load balanced clusterApply
- **clusterEvalQ**
  - evaluate expression on nodes
- **clusterSplit**
  - split vector into pieces for nodes

---

## Example

### Bootstrapping MM-regression estimators

- The function `roblm` (from the library of the same name) calculates the MM-regression estimators.
- Is also available in the library `robustbase` (see talk by Martin Mächler and Andreas Ruckstuhl).
- Can use bootstrapping to calculate the empirical density of $\hat{\beta}$.

```r
library(roblm)
X <- data.frame(y=rnorm(500),
                x=matrix(rnorm(500*20), 500, 20))
samples <- list()
for (i in 1:200)
  samples[[i]] <- X[sample(1:500, replace=TRUE),]
rdctrl <- roblm.control(compute.rd=FALSE)
```

---

### Non-parallel - Takes 196.53 seconds

```r
lapply(samples,
        function(x, z)
        roblm(y~., data=x, control=z), z=rdctrl)
```
Example
Bootstrapping MM-regression estimators

Non-parallel - Takes 196.53 seconds

```r
lapply(samples,
function(x,z)
  roblm(y~., data=x, control=z), z=rdctrl)
```

Parallel - 4 CPUS - Takes 54.52 seconds

```r
cl <- makeCluster(4)
clusterEvalQ(cl, library(roblm))
clusterApplyLB(cl, samples,
  function(x, z)
    roblm(y~., data=x, control=z), z=rdctrl)
stopCluster(cl)
```

Example
Linear Grouping Analysis (LGA)

- Is a clustering algorithm that finds $k$ groups around hyperplanes of dimension $d - 1$ using orthogonal regression.

The algorithm is given by

1. **Initialization**: Initial hyperplanes are defined by the exact fitting of $k$ sub-samples of size $d$.
2. **Forming $k$ groups**: Each data point is assigned to its closest hyperplane using Euclidean distances.
3. **Computing $k$ hyperplanes**: New hyperplanes are computed applying orthogonal regression to each group.
4. Steps 2) and 3) are repeated several times.

- This process is started from a number of random initializations, and the best result selected.
- The number of starts depends on $k$, the relative sizes of the groups, and $d$. 
### Example

**Linear Grouping Analysis (LGA)**

- We have a list `hpcoef` containing `m` matrices that specify each of the starting `k` hyperplanes.
- We wish to iterate from these starting hyperplanes with the function `lga.iterate`.
- In this example, the dataset has `n = 10,000`, `k = 4`, `d = 2`.

#### Non-parallel - Takes 851 seconds

```r
outputsl <- lapply(hpcoef, lga.iterate, xsc, k, d, n, niter)
```

#### Parallel - 4 CPUs - Takes 230 seconds

```r
cl <- makeCluster(4)
outputsl <- clusterApplyLB(cl, hpcoef, lga.iterate, xsc, k, d, n, niter)
stopCluster(cl)
```

### Closing Remarks

- With a small amount of preparation, it is relatively simple to implement parallel programming for suitable problems.
- The technology for small scale implementations is available to most researchers.
- The efficiency gains versus effort expended makes parallel computation something to seriously consider.
- However, when working in a heterogeneous computing environment, care needs to be taken!