R at the U of R:
Benchmarking Computing Resources

1 Introduction

The purpose of this executive summary is to briefly describe the star lab benchmarking project, highlight the results of the project, and offer suggestions based on these results.

A full technical report detailing the exact protocol for the tests, relevant details for replication, description of necessary source code, and providing a technical discussion of results and suggestions is available on the star lab's website\footnote{http://www.rochester.edu/college/psc/thestarlab/main/} and from the author's website\footnote{http://www.rochester.edu/college/gradstudents/jolmsted/}. If for whatever reason the report cannot be obtained from these locations, send an email to the author.

This report is best viewed in color. Though all attempts have been made to represent the information presented here in monochrome (in addition to color), they are not always as clear.

2 Project Motivation

The star lab benchmarking project has a deliberate and practical goal:

> to provide a current and comprehensive understanding of how the various computing resources available to the Political Science Department at the University of Rochester perform against each other for the benefit of the administrators and users.

3 Tests

To evaluate an environment (a term I define below), five relatively short tests are run. Each test is run multiple times by repeatedly invoking a function provided by the R package SLbench which was written for the purposes of this benchmarking project. See the author's website for more information and documentation of the SLbench package. A description of each test follows.

IPE ideal point estimation applied to a standard set of votes from the U.S. Senate, estimated using the wnominate package

Matrix inversion, transposition, and multiplication applied to a large matrix

Optim a poisson statistical model of US executive agency creation estimated with via MLE

RNG generation of a large number of random numbers from various distributions

Standard a standard benchmarking script comprising a battery of tests including matrix calculations, linear regression, FFT

This selection of tests includes both basic tasks (e.g. random number generation, matrix operations) and practical political science computations (e.g. ideal point estimation, optimization). Each test is repeated a fixed number of times in each environment. The total wall-time required to complete the computation is recorded.
4 Environments

For this project, I define an environment as a unique combination of software (i.e. OS, numerical libraries, and R) and hardware.

4.1 Hardware

There are five different hardware configurations covered by this benchmarking project. I will use the bold abbreviations to refer to these hardware configurations throughout the rest of this report.

**HP** this is the author’s main person computer, an HP dv6000 with two 2.0 GHz processors, with 2.0 Gb of RAM (circa 2007)

**Mars1** this is the Mars cluster in the star lab using the hardware prior to the November/December 2010 upgrade

**Mars2** this is the Mars cluster in the star lab using the hardware after to the November/December 2010 upgrade

**P690** this is one of the more powerful desktop workstations in the starlab and has eight processors

**Bluehive** this is the standard cluster maintained by the CRC for use by the university community

*Comparisons across hardware are of particular interest.*

4.2 Operating System

Both Windows XP and Ubuntu 10.10 run on **HP**. **Mars1** runs Fedora Core 6, as does **Mars2**. **Bluehive** runs Red Hat Enterprise Linux 5.5. And, **P690** runs Windows XP. *Comparisons across operating systems are possible only on **HP** and are not of particular interest.*

4.3 R

The **HP** machine runs CRAN R 2.11.1 under Ubuntu, CRAN R 2.11.1 under Windows XP, and a version of Revolution R which is based on CRAN R 2.11.1. **Mars1** runs CRAN R 2.9.2 and **Mars2** runs CRAN R 2.11.1. **P690** runs CRAN R 2.11.0 and a version of Revolution R which is based on CRAN R 2.11.1. **Bluehive** runs CRAN R 2.11.1. No difference in R versions is known to be responsible for any observed performance differences. *Only comparisons of CRAN R versus Revolution Analytics R are of interest.*

4.4 Numerical Libraries

Basic Linear Algebra Subroutines (BLAS) refers to a bundle of linear algebra subroutines that is external to R, itself. There are various tuned BLAS implementations which are often built with specific hardware in mind. R has the ability to use a BLAS other than the reference BLAS that comes with R by default. Several are considered in this project. Their descriptions are below. The listed abbreviations are suggestive of the labels used when presenting results.

**Ref.** the reference BLAS library is compiled into the R binary and any additional executable code that may be created by packages; not tuned in any way

**Sh. Ref.** the shared reference BLAS library is contained in executable code and all calls flow through the single external file; not tuned in any way

**Sh. Intel (1)** the shared Intel MKL BLAS implementation is tuned for Intel chips, but generally fast; supports multi-threading, of which only one thread is used; has a restrictive EULA; comes by default in Revolution R

---

3Currently, the cluster has 24 physical nodes with 4 processors per node and 4 GB of RAM per node.
Sh. Intel (2) the shared Intel MKL BLAS implementation is tuned for Intel chips, but generally fast; supports multi-threading, of which two threads are used; has a restrictive EULA; comes by default in Revolution R

Sh. MKL the shared Intel MKL BLAS implementation is tuned for Intel chips, but generally fast; has a restrictive EULA; this is not the version distributed with Revolution R

Sh. Goto (1) the shared GotoBlas BLAS implementation is released by UT Austin, but development has ceased since the beginning of the benchmarking project; the license is free to academic users; this is a tuned BLAS; supports multi-threading, of which only one thread is used

Sh. Goto (2) the shared GotoBlas BLAS implementation is released by UT Austin, but development has ceased since the beginning of the benchmarking project; the license is free to academic users; this is a tuned BLAS; supports multi-threading, of which two threads are used

Sh. Atlas the shared Atlas BLAS implementation is designed to have good performance across a wide array of hardware setups because it dynamically determines optimal configurations and compile-time settings for the BLAS

Comparisons across different BLAS implementations are of interest.

5 Results

5.1 A Single Process in each Environment

By design, the tests were created to take roughly one hour in a baseline environment (i.e. the HP machine with Ubuntu and the reference BLAS library). The exception to this is the standard test which takes roughly two hours. All that matters are times relative to each other. The times for the tests when run on a single processor in the different environments are show in Figure 1.

A great deal of information is contained in these results, so I outline the key features:

• Mars2 is roughly twice as fast as Mars1

• Mars2, per process, is considerably faster than Bluchive

• Mars2 is fast even relative to the desktop machines (HP and P690)

• using a performance BLAS increases performance (see XP REF vs. XP SH_INTEL1 on P690 or UB REF vs. UB SH_GB2 on HP) only on matrix-heavy operations (e.g. the Matrix and Standard tests)

• still, using a performance BLAS doesn’t decrease performance on non-matrix-heavy operations

• the performance realized by using Revolution R on Windows XP (see XP SH_INTEL1 and XP SH_INTEL2 on HP) can be realized within CRAN R by changing the BLAS —which is what Revolution does—(see UB SH_GB1 and UB SH_GB2 on HP)

• the installation of R packages that use external compiled code (see the IPE test) on Unix-like machines creates a dependency of performance on the actual compilation (e.g. compiler flags, compiler choice), whereas Windows machines all download and use the same binary file (see the decreased performance of UB REF compared to XP REF on HP)4

5.2 Revolution vs. CRAN

The question of performance of Revolution R vs. CRAN R is a particularly important one for the star lab. For this reason, we will take a closer look comparing these two environments. Figure 2 contains the wall-times for the five different tasks using multiple processors in three different environments, each on two machines.

The important features of these results are highlighted below:

It is not known whether this is a known issue, common unidentified problem among users, or a problem only on HP.
Results from running each test in each different environment using a single processor. The units are in minutes to completion after the process has started. Any queuing problems on clusters, for example, would only add to this time. Each environment specification is grouped by machine. For each environment an abbreviation of the operating system and then the numerical library constitutes the label. Comparisons across tests are not meaningful. Comparisons within tests are. This figure is best viewed in color.
Results from running each test in three different kinds of environments using multiple processors on two different machines. The units are minutes until completion once the process has started. Each environment is on Windows XP. CRAN R is marked by red, circles, and a solid line. Revolution R with the single-threaded BLAS is marked by green, triangles, and a dotted line. Revolution R with a multi-threaded BLAS is marked by blue, squares, and a dashed line. Comparisons across tests are not meaningful. Comparisons within tests are meaningful. This figure is best viewed in color.
• for tests during which no calls are made to the external libraries (i.e. IPE, Optim, RNG) Revolution R should not and does not out-perform CRAN R

• for tests that do call the external BLAS libraries (i.e. Matrix, Standard) Revolution R with its performance BLAS is faster

• bottlenecks can occur when calling external BLAS libraries; more processors only helps computation that occurs within R and does not affect those that are passed to an external library

• parallel execution does not result in perfect scaling, but it certainly improves performance when the computation occurs within R (see the IPE, Optim, and RNG tests)

5.3 Parallelization

Because we have already compared the relative merits of the hardware setups, BLAS implementations, and the differences between Revolution R and CRAN R, what remains to be discussed is how conducive the environments are to parallelization. See Figure 3 for these results.

Many of the observations are similar to the presentation of CRAN vs. Revolution R wall-times (Figure 2), but they are included anyway:

• these tests parallelize quite well (note the logarithmic scale), but these tests aren’t heavy on communication or data transfer which are the major hold-ups in parallel processing

• the exceptions to the above statement are those tests run on Windows which are matrix-heavy

6 Conclusions

I conclude the report with several of the most important observations and some suggestions.

First, Mars2 is a substantial upgrade to Mars1. Jobs run almost twice as fast. The nodes on this cluster are the single fastest processors in this project. The upgrade represents a considerable increase in the computing power available to the University of Rochester Political Science Department.

Second, parallel evaluation of R code is not reserved for clusters or Revolution R, it can be achieved on any hardware that runs R with more than one processor. And, moreover, this can be achieved with open software. Because of this, writing new R code in a way that uses multiple processors is a profitable step. Most users are probably not aware of how to do this, education is critical.

Third, high performance BLAS implementations are not reserved for Revolution R. Similarly fast or even faster BLAS libraries can be used by R on any platform. These should be installed and used by default on users’ machines and on the machines in the star lab (both desktops and the cluster). Most users are probably not aware of how to do this, education is critical.

Fourth, Bluehive is an attractive resource insofar as one desires a large number of processors. Beyond that, the same work can be done in-house on Mars2 and sometimes P690 in less time per processor. For the time being, over-selling the merits of Mars2 is probably difficult.
Results from running each test in different hardware-BLAS combinations. Each test is performed in each environment using 1, 2, 4, and 8 processors. The units are minutes until completion once the process has started. **Bluehive** runs RHEL. **Mars1** and **Mars2** run Fedora Core. **P690** runs Windows XP. Comparisons across tests are not meaningful. Comparisons within tests are meaningful. Comparisons within environment across processors are the focus. This figure is best viewed in color.