Creating and Using a Non-Dedicated HPC Cluster with ParallelKnoppix

Universitat Autònoma de Barcelona

February, 2006

Abstract

This note describes ParallelKnoppix, a bootable CD that allows econometricians with average knowledge of computers to create and begin using a high performance computing cluster for parallel computing in very little time. The computers used may be heterogeneous machines, and clusters of up to 200 nodes are supported. When the cluster is shut down, all machines are in their original state, so their temporary use in the cluster does not interfere with their normal uses. An example shows how a Monte Carlo study of a bootstrap test procedure may be done in parallel. Using a cluster of 20 nodes, the example runs approximately 20 times faster than it does on a single computer.

Keywords: parallel computing; high performance computing; cluster; Monte Carlo; bootstrap

JEL classifications: C15; C87; C88

1 Introduction

A high performance computing (HPC) cluster is a group of computers that are networked together, and which have a software environment that allows given a computational task to be split up into parts that are evaluated on more than one CPU simultaneously. This can lead to dramatic reductions in the time needed to complete computations, in comparison to running them on a single CPU. ParallelKnoppix[1] (P-KPX) is a bootable CD that allows users with average computing skills to create a HPC cluster in very little time - about 10 minutes. The computers used in a P-KPX cluster may be heterogeneous, and the cluster is temporary, in the sense that nothing is installed on the computers that are used in the cluster - they are not altered in any way. Thus, for example, the computers in a university computer room that are used for students' work during the day can be converted into a HPC cluster for nighttime research work, without affecting their use by students the next day.

This note describes what P-KPX is, how it works, how to set it up and use it, and presents an example of how it can be used to accelerate a Monte Carlo study of a bootstrap testing procedure. The intention is to make econometricians aware of the existence of the tool and to give basic information of how to use it, along with pointers to additional sources of information. Creel (2005) presents additional examples of parallel programs and results that may be of interest to econometricians, and that were obtained on a P-KPX cluster. The example presented in this paper as well as the examples from Creel (2005) are all contained on the latest version of the CD.

In what follows I first give a more detailed description of what P-KPX is and how it works. Then I describe the setup process. The fourth section gives an example that shows how a Monte Carlo study that involves 2,001,000 nonlinear optimizations may be completed in less than 4 hours, when it would take roughly 80 hours to complete running on a single computer. The fifth section provides a reference to additional information about software contained on the P-KPX CD, and the sixth section gives information about similar projects.

2 Description

ParallelKnoppix is a modified version of the Knoppix[2] distribution of the GNU Linux operating and software system. Knoppix is a CD image that may be downloaded and then burnt to a CD. The resulting CD can be used to boot a computer of the IA-32 architecture[3]. Knoppix contains hardware detection and automatic configuration tools that determine the computer's graphical display type, the networking cards installed, etc. This process only makes use of the computer's CPU and RAM - the hard disk(s) are not used unless the user chooses to do so. Once the computer is halted, it is in its original state. Thus, Knoppix can be used to run GNU Linux on a computer that has its entire hard disk occupied another operating system.

P-KPX is a modification of the Knoppix CD image that adds software needed to perform parallel computations as well as tools for rapid configuration and creation of a HPC cluster. Once

[1] ParallelKnoppix may be freely downloaded from the homepage at http://pareto.uab.es/mcreel/ParallelKnoppix. This page also provides a tutorial, a user forum, and additional information.

[2] http://www.knopper.net/knoppix/index-en.html

[3] Most if not all desktop computers that have 32 bit CPUs by Intel and AMD will work properly with P-KPX. It has not yet been tested with 64 bit CPUs.

the "master node" is booted using the CD, a configuration script is used to boot the remaining "slave" nodes of the cluster across the network[4]. The slave nodes receive their operating system from the P-KPX CD, which is in the CD tray of the master node. Automatic hardware detection also occurs on the slave nodes, and for this reason, they may be heterogeneous, with different CPU types, different network cards, etc. The fact that all the computers in the cluster receive their filesystems from using the same CD ensures that all of the computers in the cluster have access to exactly the same software. Once the slave nodes are booted, a directory on the master node is shared across the network with all the slave nodes, so that user programs and data can be made available to all the nodes of the cluster. This directory may be on a USB memory stick, to avoid using the master node's hard disk in any way. This directory may be re-used every time P-KPX is used, so work and results may be stored there, and additional software packages that are not on the CD may be installed there. The entire setup routine is mouse-based, and is designed to be a simple and quick as possible. An experienced user can complete the setup process in less than five minutes.

ParallelKnoppix is focused on parallel computing using the Message Passing Interface (MPI) specification, though the Parallel Virtual Machine (PVM) is also supported. The LAM/MPI, MPICH, and openMPI implementations of the MPI specification are included on the P-KPX CD. These are libraries of compiled C and FORTRAN functions which may be included in users' programs to allow the use of message passing between compute nodes to achieve parallelism. Higher level programming languages can also make use of MPI, through binding functions that provide access to the lower level C and FORTRAN functions contained in a given implementation library. The P-KPX CD contains MPI-enabled versions of GNU Octave[5], R[6] and Python. One of the advantages of using a high level language is that the underlying parallelism of a program may be almost entirely hidden from end users (Creel, 2005).

3 Setup and use

A detailed tutorial with many screenshots that shows how to set up a cluster with P-KPX is available at http://econpapers.repec.org/paper/aubautbar/626.04.htm. The setup process is done using a graphical interface that is initiated using an entry in the Desktop menu list. This setup process:

- detects network cards, and configures a private network using the selected network card
- prompts the user for the number of computers to include in the cluster, and for the type of network cards that are to be used
- boots the slave nodes across the network, using the PXE capabilities of their network cards
- mounts a mass storage device for use as working space, and NFS shares a directory on the device with all of the nodes in the cluster. The mass storage device may be a USB storage device, to avoid using the hard disk(s) of the master node.

[4] The network can be a pre-existing ethernet network, or one specially constructed using cables and a switch, for example. If an existing network is used, any other DHCP servers must be disabled while the slave nodes are booting to ensure that they receive their IP addresses from the DHCP server running on the master node.

[5] Octave is a free software package that is very similar to MATLAB.

[6] R is a free software package that is similar to the S language.

- copies a number of example programs for C, FORTRAN, Octave and other languages to the working space
- mounts the shared working directory on all of the slave nodes. Data and programs placed in the working directory are available to all of the nodes of the cluster.
- optionally secures the cluster by changing passwords and generating new RSA keys, so that it becomes safe to connect the cluster to the Internet. This allows for simple transfer of data onto and off of the cluster.

The most important hardware requirement for a trouble-free experience is that the slave nodes' network cards be capable of booting across the network (PXE boot). This is the case for all newer network cards, though the feature must often be enabled using the computers' BIOS setup routines. It is possible to work around this requirement, but beginning users are advised to start with machines that can boot using PXE. The most important stopping block in terms of users' knowledge is determining the models of network cards used by the computers in the cluster, so that the correct modules may be selected in order to boot the slaves across the network. To find out what model of network card a given computer uses, one may consult the information made available by the operating system that is installed on the computer, or if doubts persist, boot the computer using the P-KPX CD, and examine the output of the automatic configuration process.

At this point, the cluster is ready to use for parallel computing. The whole process takes an experienced user about five minutes to complete. The master node has the IP address 192.168.0.1, and the nodes have IP addresses that run up to 192.168.0.X, where X is the total number of nodes in the cluster ($X \leq 200$). Each of the nodes may be accessed using the names `node`x, $x = 1, 2, ..., X$. One may copy data onto and off the cluster across the network using the `scp` protocol or graphically using a web browser and the `fish://` protocol. Alternatively one may use a USB storage device or any of the hard drives available on any of the computers in the cluster. It is worth emphasizing that a user who has physical access to the cluster can potentially access or destroy information on any of the hard disks of the computers in the cluster. By default, the hard disks are not used in any way, but they can be accessed as an option. If the administrator performs the optional step of changing passwords and re-generating RSA keys, then users without physical access to the cluster will not be able to access hard drives.

The working directory contains examples for C, FORTRAN, GNU Octave, and other languages and programs. These can be run immediately after creation of the cluster. The web browser provided on the CD contains bookmarks for tutorials for MPI and other relevant information. Creel (2005) presents several examples of econometric problems that are parallelized in a manner that is mostly transparent to the end user, using GNU Octave. The following section presents an example that does a Monte Carlo study of a bootstrap test statistic.

4 Monte Carlo study of a bootstrap test

Creel (2005) provided several examples of how certain computations routinely done by econometricians can be parallelized and executed on a cluster. One of these was the case of Monte Carlo

studies, which are quite obviously parallelizable[7]. The example discussed there was designed to introduce parallel computing, keeping the discussion as simple as possible. However, code of that example has some weaknesses that make it unsuitable for research work. The most serious is that the failure of any of the nodes in the cluster will cause the program to interrupt, loosing the results of all calculations. Since a cluster may be made up of a large number of nodes, the probability that one fails (or is accidentally turned off) during the course of time-consuming computations may not be negligible. Another problem is that the overall number of Monte Carlo replications is simply evenly divided amongst the available nodes. If the nodes are not evenly-matched in terms of their CPU speeds, this allocation of work is not efficient. The `montecarlo.m` and `montecarlo_nodes.m` functions provided on the current version of the P-KPX CD have been modified to allow work to proceed without interruption in the event that one or more nodes fail, and do load-balancing to ensure that all nodes are kept fully occupied up to the point that the desired number of replications are obtained. Results are also flushed to an output file as they are generated to allow for the possibility that the master node might fail. These new versions are extensively commented, for readers who wish to know the details. The older, more didactic versions are also archived on the CD.

A number of studies have shown that bootstrap test procedures often work substantially better than do procedures that rely on asymptotic critical values (for example, Horowitz, 1999; MacKinnon, 2002). Yet as pointed out by MacKinnon, there are often a number of ways of conducting a bootstrap test, the specifics of how the procedure is performed can be important, and there is presently little knowledge of which procedures are most accurate for typically encountered sample sizes. Here, I conduct a Monte Carlo study of a single procedure for testing hypothesis about the parameter of a simple probit model, comparing it to the performance of the usual test procedures that rely on asymptotic results.

The data generating process (DGP) is a simple probit model:

$$y = 1\left(\epsilon < \Phi(\alpha + \beta x)\right)$$

where $\epsilon \sim U(0,1)$, $x \sim U(-1,1)$, $\Phi(\cdot)$ is the standard normal distribution function and $1(\cdot)$ is the indicator function. I set $\alpha = 0$, and β is drawn randomly from a $U(0,1)$ distribution. I test the hypotheses that β is equal to its true value (size) and that it is equal to zero (power). The test is done both using standard asymptotic results and using a bootstrap procedure that draws the bootstrap data from the DGP under the null hypothesis. The sample size is $n = 15$. With such a small sample, we may anticipate that the asymptotic procedure may not perform well. I use 1000 Monte Carlo replications, and each bootstrap is done using 1000 replications. For each of the 1000 Monte Carlo replications there is a single estimation using the "true" data, 1000 estimations for the bootstrap samples generated under the true null hypothesis, and 1000 estimations for the bootstrap samples generated under the false null hypothesis. This gives a total of 2,001,000 nonlinear optimizations.

To reiterate one of the points made in Creel (2005), end users do not need to know hardly anything about parallel computing in order to use a parallel program that has a friendly interface.

[7] Bruche (2003) and Doornik *et al.* (2002, 2005) give additional examples of Monte Carlo studies of econometric problems done in parallel on distributed clusters.

The script used to conduct the Monte Carlo study is presented in Listing 1. Lines 1-7 define the particulars of what is to be done, and how. Line 8 calls the function `montecarlo.m`, which does the work. If this line were identical, but without the last two arguments, the Monte Carlo study would be done serially on the computer that was booted using the P-KPX CDROM. As it is presented, the Monte Carlo study is done on the master node plus the 19 slave nodes specified in line 5 of Listing 1.

In line 8, the last argument, `n_pooled`, the value of which is specified in line 6, is only relevant when execution is in parallel. This is the number of Monte Carlo replications that the slave nodes should do before passing the results back to the master node. The master node writes all results to the output file as they are received, but when there are no results ready to write to file, it performs a replication of the function under study. `n_pooled` should be set high enough so that the CPU overhead due to network traffic as results are passed to the master node does not interfere substantially with the CPU time of the master node being devoted to calculations. The computational intensity of the function being studied (the first argument in line 8, in this case, the function `bootstrap_tests.m`), determines the appropriate value of `n_pooled`. For a time consuming problem, as is the case with this example, `n_pooled` can be quite small. For less computationally demanding problem, it could be much larger. The guiding factor is that the loss, due to the failure of a slave node, of the results for which writing to file is pending should not be excessively annoying. A bit of experimentation before launching a definitive run can be used to determine an appropriate value. To replicate the results I report below, one only needs to create a cluster using P-KPX, and then run the script `mc_example3.m`, found in the `./Octave/montecarlo` subdirectory of the Examples directory that is found on the Desktop. The timings that one obtains will of course depend upon the size of the cluster and the speed of the computers that make it up.

The Octave script in Listing 1 was run on a cluster of 20 Pentium IV desktop computers, each with a 2.8 GHz processor with 1 MB of level 2 cache. The total time to complete the 1000 Monte Carlo replications was 3.96 hours. For comparison, the same script, modified to complete only 10 Monte Carlo replications, using serial computing, was run on a computer with a 3.0 GHz Pentium IV CPU, using the Ubuntu Linux distribution. The time to complete the 10 replications was 0.857 hours, so the time to complete 1000 replications would be roughly 85 hours. The 20 node cluster achieves a better than 20X speedup. The comparison is not entirely fair, since the versions of Octave on P-KPX and Ubuntu Linux were compiled using software libraries, compilers, and compiler switches that may differ somewhat, and the 10 replications done on the single computer may not be representative. Nevertheless, the results clearly show that an overnight 12 hour run on the P-KPX cluster would do an amount of work that would require more than 10 days on a single computer.

Turning to the results, in Figure 1, we see a plot rejection frequencies as a function of nominal size, with a 45 degree line to facilitate comparison. The bootstrap test has quite accurate size, with a slight bias below the nominal size. The asymptotic test has a tendency to over-reject, and the bias is more pronounced than is the case for the bootstrap test. Looking at the lines for false null hypotheses, the bootstrap test has power somewhat below that of the asymptotic test, but the difference is mostly due to the size bias of the asymptotic test. It is important to keep in mind that this is an average power curve, over all true parameter values, where the true parameter is uniformly distributed on [0,1]. When the true parameter value is close to zero, power

is low, whereas if the true parameter value is close to one, power is higher. Figure 2 plots rejection frequencies for the 106 replications where the true parameter value is greater than 0.9. The lines are much less smooth, since there is more Monte Carlo noise due to fewer replications, but we see that power is considerably higher when the null hypothesis is more radically false.

5 Conclusion

The example presented here is additional evidence of the well-known fact that the computationally demanding work implied by Monte Carlo studies can be sped up considerably if it is done on a cluster of computers. ParallelKnoppix is a rapid means of gaining access to such a cluster. It may be used on banks of computers that are temporarily idle, without disrupting their normal use when they are not idle.

The P-KPX CD contains a number of additional examples. It also contains easy to use scripts that allow one to modify the CD itself, so that additional programs and data may be permanently added. Contributions to the project in are welcome. Creating new examples, writing documentation, and extending the functionality are all possibilities.

6 A selective list of resources

This section briefly mentions some resources relevant to parallel computing that are contained on the P-KPX CD, along with URLs that provide additional information.

GNU Octave (http://www.octave.org)

Gnu Octave is a high level matrix programming language that is mostly compatible with MATLAB (http://www.mathworks.com). MATLAB scripts generally run unmodified on Octave, though they may fail if they make use of 3rd party functions that are not available in Octave. A collection of add on software for Octave is available at http://octave.sourceforge.net.

LAM/MPI (http://www.lam-mpi.org)

LAM/MPI is an implementation of the entire MPI-1.2 specification, and much of the MPI-2 specification. MPITB for GNU Octave is by default compiled linking to LAM/MPI, though it can be compiled linking to other implementations.

MPI (http://www-unix.mcs.anl.gov/mpi)

MPI is a standard for message passing. Message passing is one means of parallelizing programs. Libraries that implement message passing that adhere to the MPI standard can be optimized for best performance on particular platforms. Software that uses MPI can be compiled linking to any of the implementations, and is thus portable across platforms.

MPICH (http://www-unix.mcs.anl.gov/mpi/mpich)

MPICH is another widely-used implementation of the MPI standard. MPITB can be compiled linking to MPICH, instead of LAM/MPI.

MPITB (http://atc.ugr.es/javier-bin/mpitb_eng)

MPITB, by Fernández Baldomero *et al.* (2004), is a set of binding functions for GNU Octave that allow use of MPI functions inside Octave scripts. This is done by creating compiled dynamically loaded Octave functions that call the C functions provided by an MPI implementation library. Most users will not need to be concerned with the details of how this is done.

openMPI (http://www.open-mpi.org/)

openMPI is a new implementation of the MPI-2 standard. Members of a number of previous projects that implement MPI have worked together to create this new implementation. It is provided on the P-KPX CD, though currently no examples of its use are available.

PVM (http://www.csm.ornl.gov/pvm/)

The Parallel Virtual Machine is a parallelization method that is different from MPI. It allows machines of different architectures to work together in parallel. There are no examples on the P-KPX CD at present.

7 Similar projects

This section mentions some projects that can give relatively rapid access to a HPC cluster for parallel computing, and notes the relevant differences with respect to P-KPX. For some uses, one or another of these projects may be a more appropriate tool than is P-KPX. In particular, a cluster created with P-KPX is temporary. This is convenient in certain circumstances, but in others a more permanent multiuser cluster may be desired[8].

BCCD (http://bccd.cs.uni.edu/)

The Bootable Cluster CD is the most similar project to P-KPX. It pre-dates P-KPX[9], and also allows use of MPI and PVM. The main difference in the two projects is that the BCCD project focuses on teaching the creation and use of clusters, while P-KPX simply makes a cluster available for use as quickly and simply as is possible. Thus, setup and configuration of BCCD is considerably less automated than is the case with P-KPX. Also, BCCD uses the Linux 2.4.x kernel, which has considerably less support for newer hardware than does the 2.6.x kernel used by P-KPX.

[8] Work in progress is directed to allow multiple users under P-KPX. This could be of use when teaching MPI in a computer laboratory, for example. When this will become a stable feature is not yet known.

[9] The initial version of P-KPX was released in June, 2004.

ROCKS (http://www.rocksclusters.org/)

ROCKS is a distribution of Linux specialized to the creation of permanent, multi-user HPC clusters. It is a complete and full-featured distribution, with queuing and monitoring packages. Rocks is similar to P-KPX in that the slave nodes boot across the network using PXE, and their filesystems are obtained from the master node. The difference is that this filesystem is more easily modifiable since it is not on a CD. Installation of ROCKS is reasonably straightforward, but is considerably more complicated than is the use of P-KPX. ROCKS is a very attractive option for migrating to a permanent cluster installation.

Warewulf (http://www.warewulf-cluster.org)

Warewulf is a package that adds on to an existing Linux installation. Warewulf allows nodes to be booted using PXE, just as does P-KPX. As it the case of ROCKS, the filesystem that is provided to the slave nodes is contained on the hard disk of the master node. This has the advantage that the cluster's configuration may more easily be modified. Warewulf, in common with P-KPX, does not use the hard disks of the slave nodes, whereas ROCKS does. Thus Warewulf may be used to create a temporary cluster in the same way that is possible with P-KPX. Warewulf's disadvantage, compared to P-KPX, is that a Linux distribution must be installed to the master node's hard drive first, and the Warewulf package must then be installed and configured. Warewulf is designed to work with RPM-based Linux distributions.

```
reps = 1000; # number of monte carlo replications
bs_reps = 1000; # number of boostrap replications
n = 15; # sample size
iter_limit = 50; # limit BFGS iterations, if hit, the draw is not used
nslaves = 19; # how many slave nodes?
n_pooled = 10; # slaves do this many reps before sending results back to master
outfile = "bootstrap_test_07_feb_2006"; # where to write results
montecarlo("bootstrap_tests", {n, bs_reps, iter_limit},reps,outfile,nslaves,n_pooled);
```

Listing 1: mc_example3.m

Figure 1: Rejection frequencies versus nominal size, all replications (reps = 1000)

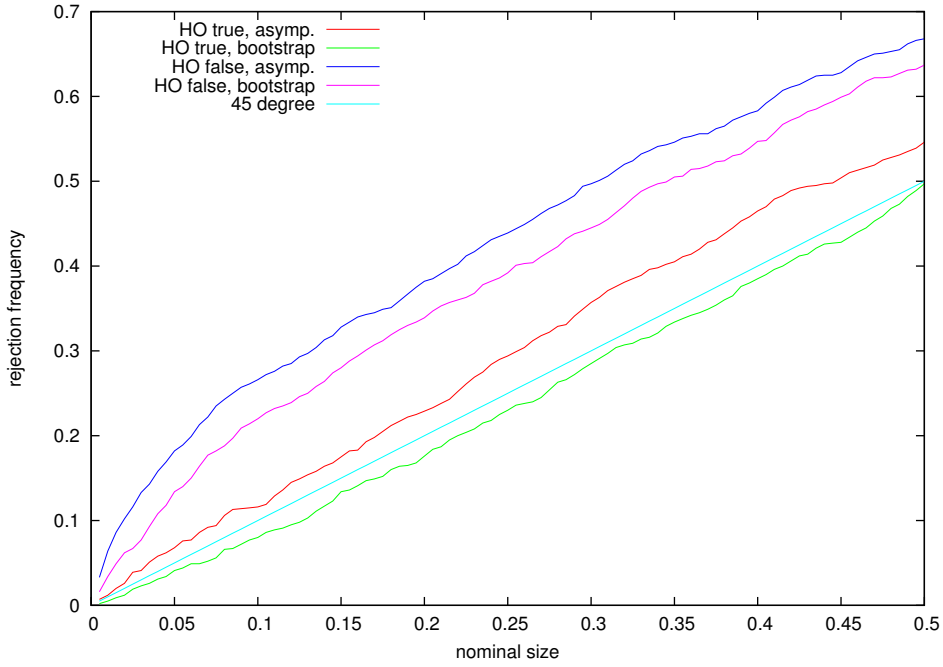
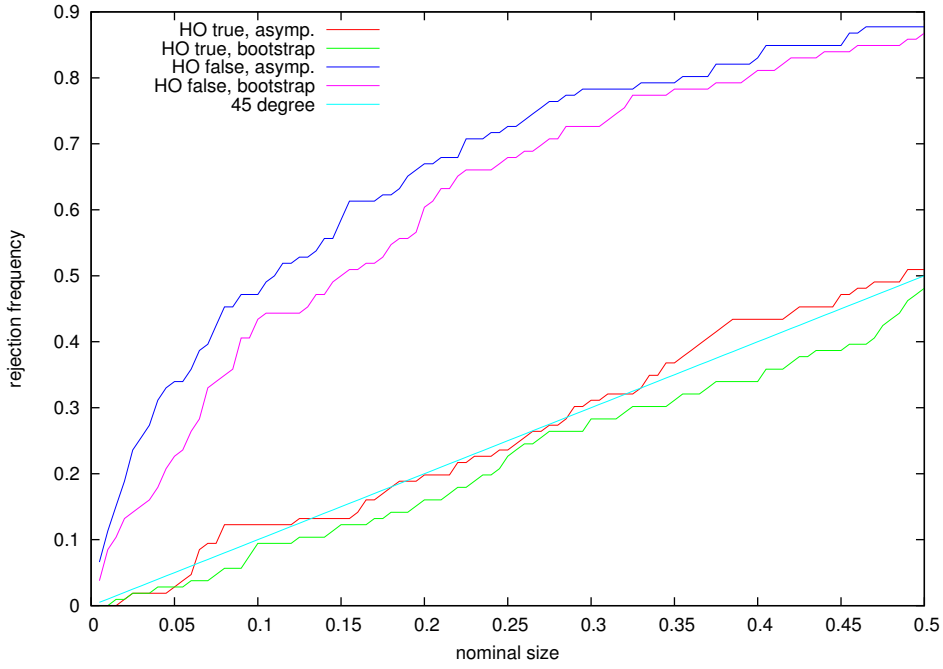


Figure 2: Rejection frequencies versus nominal size, replications where $\beta > 0.9$ (reps = 106)



References

[1] Bruche, M. (2003) A note on embarrassingly parallel computation using openMosix and Ox, working paper, Financial Markets Group, London School of Economics.

[2] Doornik, J.A., D.F. Hendry and N. Shephard (2002) Computationally-intensive econometrics using a distributed matrix-programming language, *Philosophical Transactions of the Royal Society of London, Series A*, 360, 1245-1266.

[3] Doornik, J.A., D.F. Hendry and N. Shephard (2005) Parallel computation in econometrics: a simplified approach, E. Kontoghiorgies (ed.) *Handbook on Parallel Computing and Statistics*, Marcel Dekker.

[4] Fernández Baldomero, J. *et al.*, (2004) MPI toolbox for Octave, VecPar'04, Valencia, Spain, June 28-30 2004, http://atc.ugr.es/~javier/investigacion/papers/VecPar04.pdf

[5] Horowitz, J. (2001) The bootstrap, *Handbook of Econometrics*, Vol. 5, J.J. Heckman and E.E. Leamer, eds., Elsevier Science B.V., Ch. 52, pp. 3159-3228.

[6] MacKinnon, J. (2002), Bootstrap inference in econometrics, Canadian Journal of Economics, **35**, 615-645.