An Introduction to Parallel Computing in Econometrics

Michael Creel

Universitat Autònoma de Barcelona

Master en Economia de la Empresa y Metodos Cuantitativos, UC3M, April 2008

What this presentation does:

- The purpose of this presentation (in 3 two hour sessions) is to explain enough basic concepts and to provide enough examples so that students are able to:
  - run parallel code of interest to economists
  - create their own cluster
  - write new parallel code for basic econometric methods
  - know where to look for more information
- a link to this presentation is on my homepage (`http://pareto.uab.es/mcreel/`) in the "presentations" section

A disclaimer

- the presentation is strongly derivative of my own work, and does not make much effort to explain alternatives.
- This is certainly a biased and narrow perspective. The only justification is that time is limited, and I am able to make these examples work.
- There very well may be better ways to do some of this, though I do believe that what I present is the fastest and easiest way to get started.

Outline

What is parallel computing?

Comput Econ
DOI 10.1007/s10614-008-9143-5

Multi-core CPUs, Clusters, and Grid Computing: A Tutorial

Michael Creel · William L. Goffe

Accepted: 7 April 2008

Abstract The nature of computing is changing and it poses both challenges and opportunities for economists. Instead of increasing clock speed, future microprocessors will have "multi-cores" with separate execution units. "Threads" or other multiprocessing techniques that are rarely used today are required to take full advantage of them. Beyond one machine, it has become easy to harness multiple computers to work in clusters. Besides dedicated clusters, they can be made up of unused lab computers or even your colleagues' machines. Finally, grids of computers spanning the Internet are now becoming a reality.

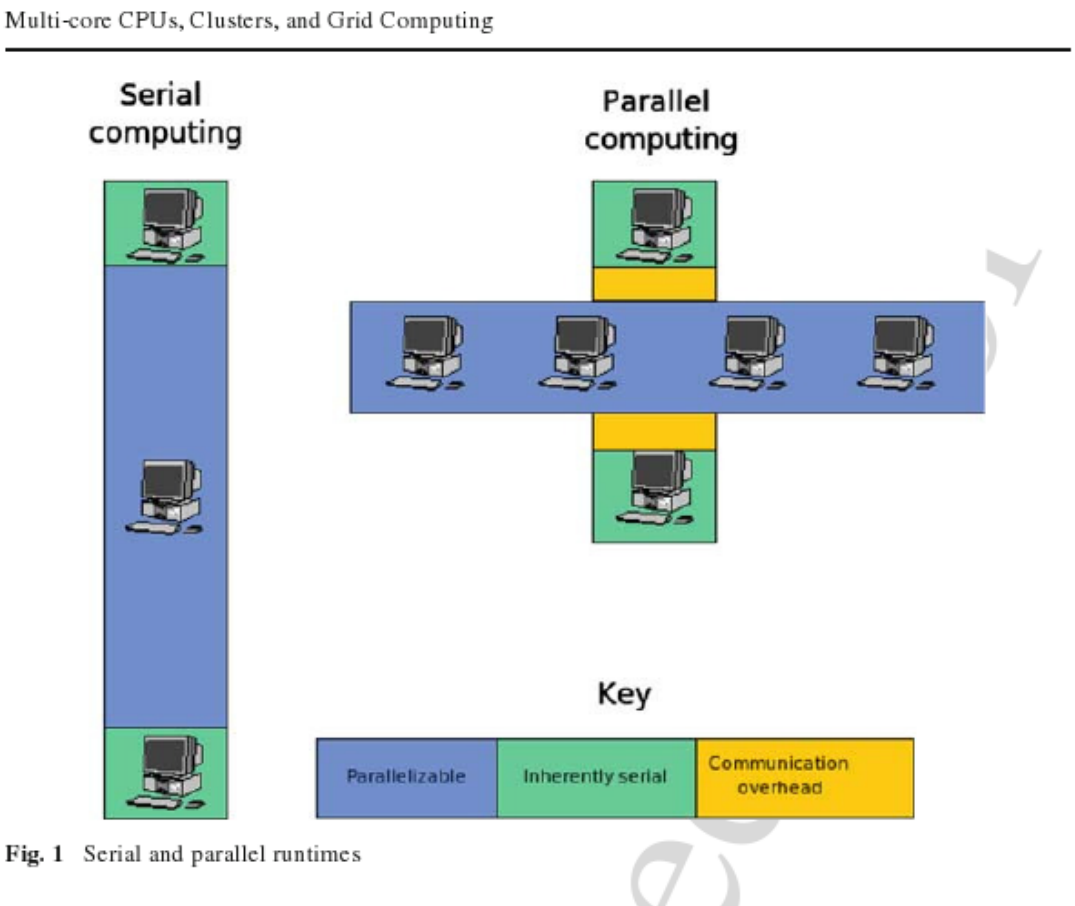
Keywords Multi-core · Cluster · OpenMP · MPI · Grid

What is parallel computing (2)

https://computing.llnl.gov/tutorials/parallel_comp/

Share work to get it done faster

Figure: Share the work



Some factors affect the performance of a parallelized version

- how much of the work is blue/green?
- how much communication overhead is introduced?
  - there may be different ways to parallelize an application, that lead to different amounts of communications overhead.
  - making a good parallelized application often requires understanding both the application and the technology for parallelization. A single person who understands both will probably do a better job than 2 people who each understand one of the parts.

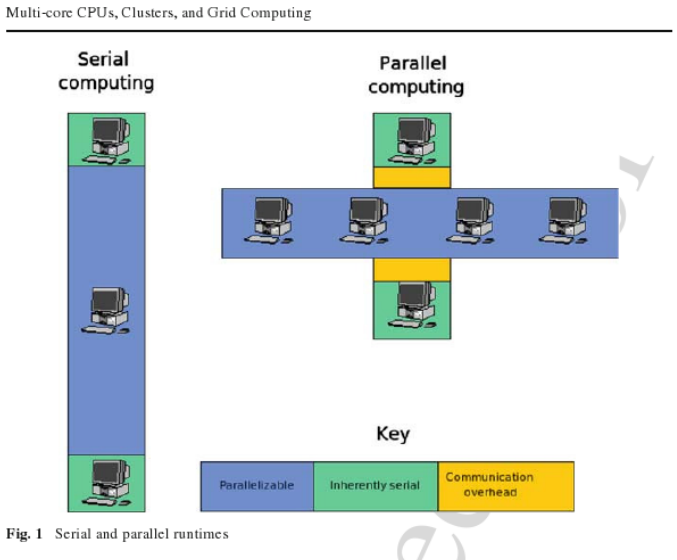


Fig. 1 Serial and parallel runtimes

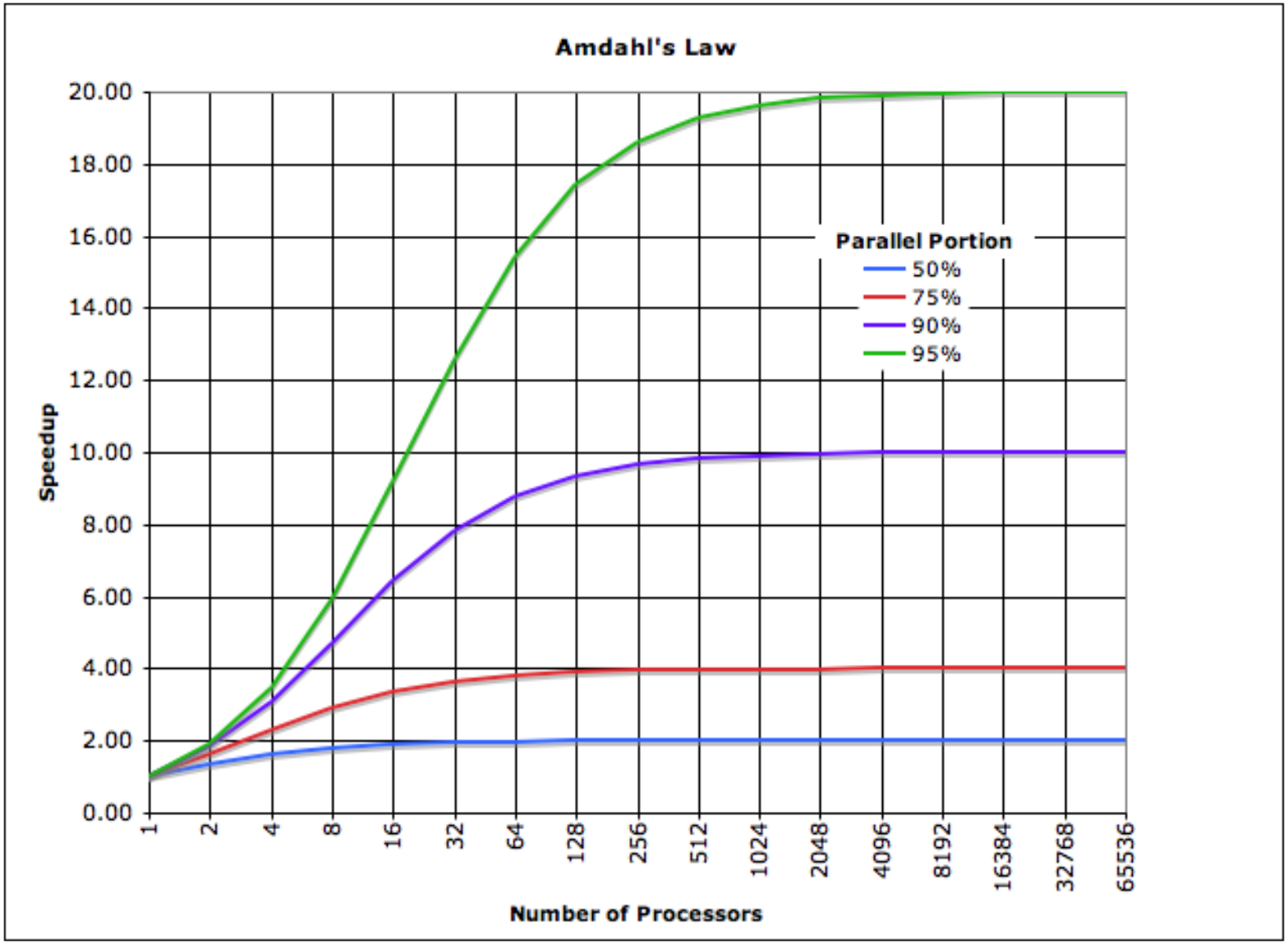
Amdahl's Law

In the case of parallelization, Amdahl's law states that if P is the proportion of a program that can be made parallel (i.e. benefit from parallelization), and $(1-P)$ is the proportion that cannot be parallelized (remains serial - the green in the previous figure) then the maximum speedup that can be achieved by using N processors is

$$\frac{1}{(1-P)+\frac{P}{N}}$$

In the limit, as N tends to infinity, the maximum speedup tends to $1/(1-P)$. In practice, performance/price falls rapidly as N is increased once there is even a small component of $(1-P)$.
source: `http://en.wikipedia.org/wiki/Amdahl's_law`

Amdahl's Law (2)



source: http://en.wikipedia.org/wiki/Amdahl's_law

Multiple cores, clusters, and grids

- There are many ways to do parallel computing. A general overview is here: `http://en.wikipedia.org/wiki/Parallel_computing`
- We'll focus on multiple cores and clusters
- Grids could also be used for econometrics, but this is outside the scope of this course. A grid is a large group of heterogeneous machines that don't trust eachother. They share CPU capacity when it is available. There is no guarantee that resources will be available on any specific machine at any particular time. seti@home or folding@home are wll known examples. A promising project that used virtual machines is `http://www.grid-appliance.org/`

Multiple cores

- A multi-core CPU has two or more cores, which essentially act as separate CPUs
- tasks will ordinarily run on a single core
  - run the sorry1core.m example and use htop to show that only 1 core is used while running a matrix inversion loop in octave
- Using more than 1 core requires specialized methods, such as multi-threading. Much currently available software will not use threads, so will run on only one core. Newer software will certainly start to use threads more widely.
- using only 1 core to crunch numbers is very annoying if you have 3 other mostly idle cores (quad core example)

High performance computing clusters

- A HPC cluster is a set of computers connected with a reasonably high speed networking technology, *that are prepared to work together to solve a given task*.
- The goal is to finish the given task more quickly by sharing the work using parallel computing methods.
- Working together on a problem means that the same data and software must be available to each node in the cluster.
  - Ensuring that this is the case is not always easy, especially as the cluster ages
  - shared filesystems that reside on a single server is one way to syncronize the nodes
  - installing and maintaining a cluster is usually a nontrivial job
  - if you use commercial software, license fees can be very expensive if you use many nodes

Parallel computing and econometrics

Not the only source, but a good place to start (gives references):

Computational Economics (2005) 26: 107–128
DOI: 10.1007/s10614-005-6868-2

User-Friendly Parallel Computations with Econometric Examples

MICHAEL CREEL
Department of Economics and Economic History, Edifici B, Universitat Autònoma de Barcelona, 08193 Bellaterra, Barcelona, Spain; E-mail: michael.creel@uab.es

Accepted 3 May 2005

Abstract. This paper shows how a high-level matrix programming language may be used to perform Monte Carlo simulation, bootstrapping, estimation by maximum likelihood and GMM, and kernel regression in parallel on symmetric multiprocessor computers or clusters of workstations. The implementation of parallelization is done in a way such that an investigator may use the programs without any knowledge of parallel programming. A bootable CD that allows rapid creation of a cluster for parallel computing is introduced. Examples show that parallelization can lead to important reductions in computational time. Detailed discussion of how the Monte Carlo problem was parallelized is included as an example for learning to write parallel programs for Octave.

Key words: bootstrapping, GMM, kernel regression, maximum likelihood, Monte Carlo, parallel computing

How to parallelize a nonlinear estimation routine?

Let's consider a paradigmatic problem in econometrics: estimation of the parameter of a nonlinear model. Supposing we're trying to minimize $s_n(\theta)$. Take a second order Taylor's series approximation of $s_n(\theta)$ about θ^k (an initial guess).

$$s_n(\theta) \approx s_n(\theta^k) + g(\theta^k)' \left(\theta - \theta^k\right) + 1/2 \left(\theta - \theta^k\right)' H(\theta^k) \left(\theta - \theta^k\right)$$

To attempt to minimize $s_n(\theta)$, we can minimize the portion of the right-hand side that depends on θ, *i.e.*, we can minimize

$$\tilde{s}(\theta) = g(\theta^k)'\theta + 1/2 \left(\theta - \theta^k\right)' H(\theta^k) \left(\theta - \theta^k\right)$$

with respect to θ. This is a much easier problem, since it is a quadratic function in θ, so it has linear first order conditions. These are

$$D_\theta \tilde{s}(\theta) = g(\theta^k) + H(\theta^k) \left(\theta - \theta^k\right)$$

So the solution for the next round estimate is

$$\theta^{k+1} = \theta^k - H(\theta^k)^{-1} g(\theta^k)$$

How to parallelize it?

- parallelize the gradient calculation over the parameters? If θ is k dimensional, we could use up to $2k+1$ computers to calculate a central difference gradient
- use a parallel search algorithm to find the best step size?
- these ideas work, but poorly.
  - why?
  - because they generate much inter-node communication. The yellow areas in Figure 1 become large, and the speedup is small.
  - you can't use more than $2k+1$ computational cores.

Econometric models use data

this provides a simple and natural way to parallelize: do calculations on different blocks of data on different computers

For a sample $\{(y_t, x_t)\}_n$ of n observations of a set of dependent and explanatory variables, the maximum likelihood estimator of the parameter θ can be defined as

$$\hat{\theta} = \arg\max s_n(\theta)$$

where

$$s_n(\theta) = \frac{1}{n} \sum_{t=1}^{n} \ln f(y_t | x_t, \theta)$$

Here, y_t may be a vector of random variables, and the model may be dynamic since x_t may contain lags of y_t.

MLE, continued

As Swann (2002) points out, this can be broken into sums over blocks of observations, for example two blocks:

$$s_n(\theta) = \frac{1}{n}\left\{\left(\sum_{t=1}^{n_1} \ln f(y_t|x_t,\theta)\right) + \left(\sum_{t=n_1+1}^{n} \ln f(y_t|x_t,\theta)\right)\right\}$$

Analogously, we can define up to n blocks. Again following Swann, parallelization can be done by calculating each block on separate computers.

For a sample as above, the GMM estimator of the parameter θ can be defined as

$$\hat{\theta} \equiv \arg\min_{\Theta} s_n(\theta)$$

where

$$s_n(\theta) = m_n(\theta)' W_n m_n(\theta)$$

and

$$m_n(\theta) = \frac{1}{n} \sum_{t=1}^{n} m_t(y_t | x_t, \theta)$$

GMM, continued

Since $m_n(\theta)$ is an average, it can obviously be computed blockwise, using for example 2 blocks:

$$m_n(\theta) = \frac{1}{n}\left\{\left(\sum_{t=1}^{n_1} m_t(y_t|x_t,\theta)\right) + \left(\sum_{t=n_1+1}^{n} m_t(y_t|x_t,\theta)\right)\right\}$$

Likewise, we may define up to n blocks, each of which could potentially be computed on a different machine.

Data parallelism

- The idea of performing the same computations on different parts of a data set is known as *data parallelism*, and it is a well-known strategy for parallelization.
- See `http://en.wikipedia.org/wiki/Data`
- since we always use data in econometrics, this is a natural place to start thinking about how to parallelize our work.

Monte Carlo fits a job-based framework

- A Monte Carlo study involves repeating a random experiment many times under identical conditions. Several authors have noted that Monte Carlo studies are obvious candidates for parallelization (Doornik *et al.* 2002; Bruche, 2003) since blocks of replications can be done independently on different computers.
- Job-based parallelism is the idea of executing independent parts of an overall computation on different CPUs. See `http://en.wikipedia.org/wiki/Task`

Monte Carlo, continued

- if we need 1000 Monte Carlo replications, we can do half on one computer and half on another
  - it would be nice to have a convenient way of organizing this
    - use many computers
    - centralized reporting of results
    - recovery if some particular computer fails or crashes

The is more than 1 way to think about this, and there is more than 1 thing to think about

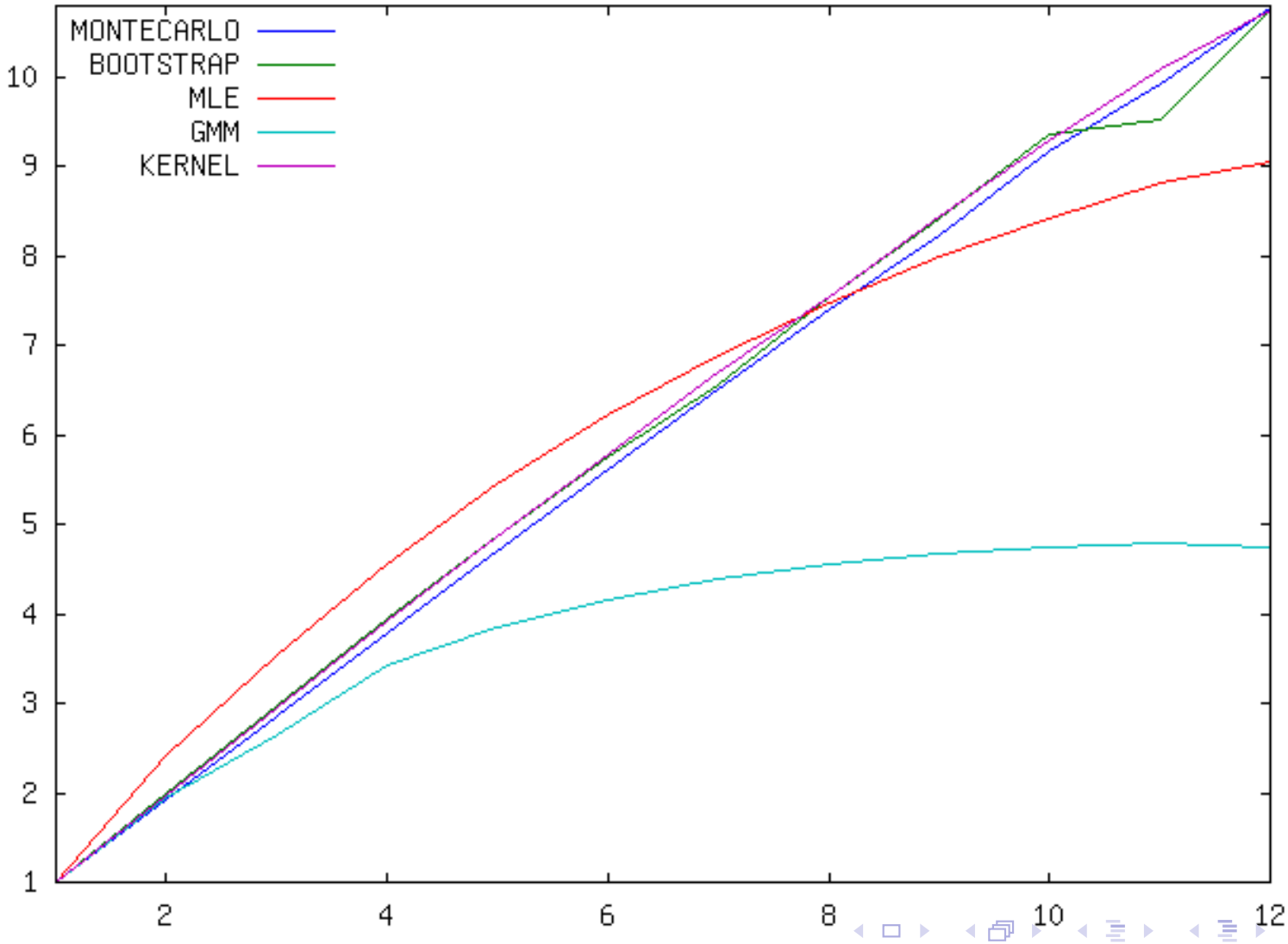
- the concepts of data parallelism and job-parallelism are not entirely separate. A real problem usually has elements of each.
- The important thing is to think carefully about your problem to find the best way to parallelize it
- what does "best" mean? There are a number of competing factors:
  - code that runs fast
  - code that is easy to write and maintain
  - code that can be re-used
  - code that is easy enough to understand so that other econometricians can use it
- experience is a good guide, but the only way to get it is to make some mistakes

Advantages and drawbacks of parallel computing

- parallelized code is obviously more difficut to write than normal serial code. This will become clear soon.
- new compilers may automatically parallelize part of code to use multiple core CPUs
- many (most?) interesting research problems are not so computationally demanding so as to require parallel computing
- however, once you know how to use parallel computing, maybe your idea of what is a feasible and interesting research project might change. The tools we have influence the research agenda.

Some speedup results

Taken from Michael Creel "User-Friendly Parallel Computations with Econometric Examples" *Computational Economics*, 2005, vol. 26, issue 2, pages 107-128



Speedup for kernel regression

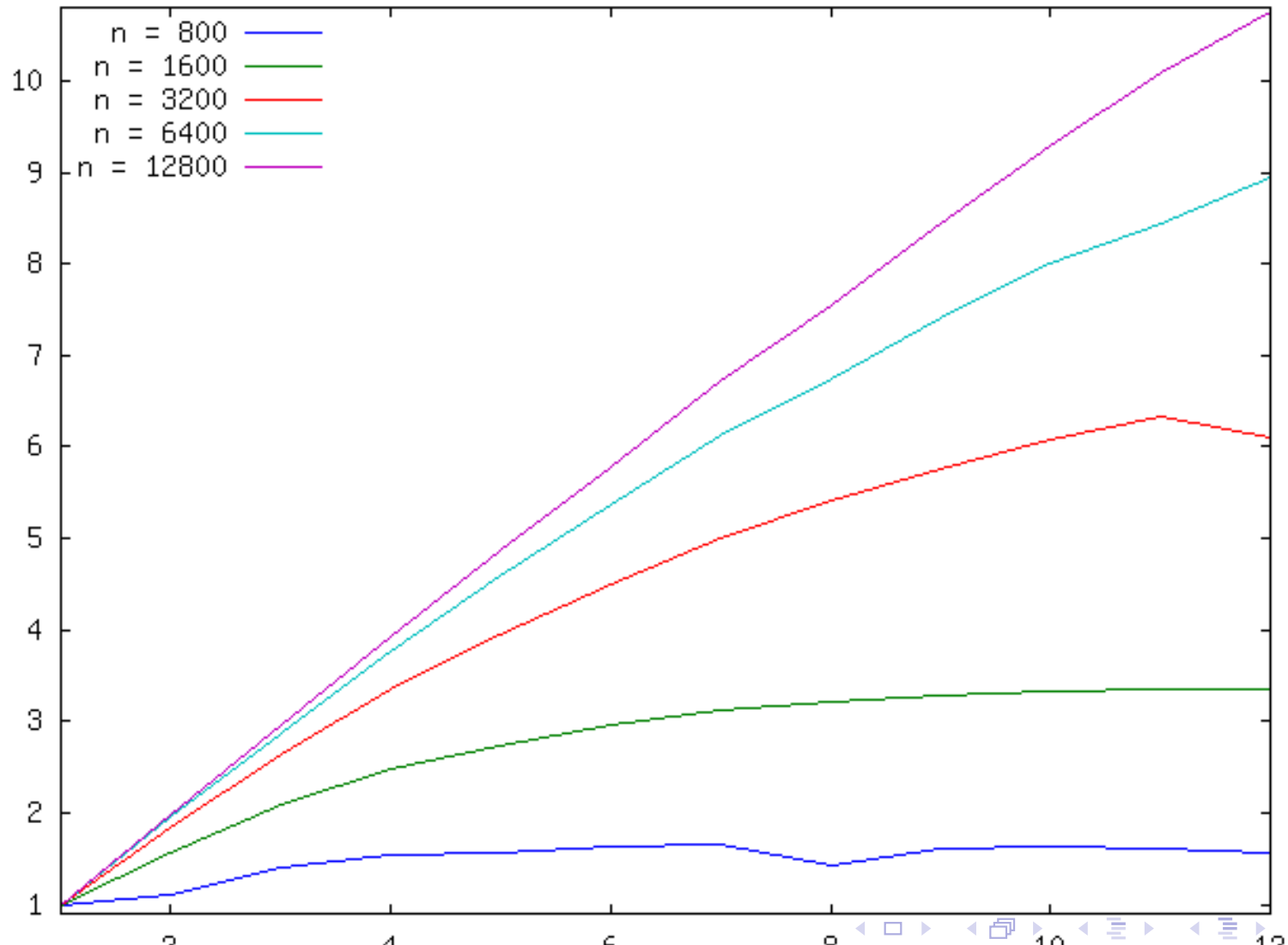
The Nadaraya-Watson kernel regression estimator of a function $g(x)$ at a point x is

$$\begin{aligned} \hat{g}(x) &= \frac{\sum_{t=1}^{n} y_t K\left[(x - x_t)/\gamma_n\right]}{\sum_{t=1}^{n} K\left[(x - x_t)/\gamma_n\right]} \\ &\equiv \sum_{t=1}^{n} w_t y_y \end{aligned}$$

We see that the weight depends upon every data point in the sample. To calculate the fit at every point in a sample of size n, on the order of $n^2 k$ calculations must be done, where k is the dimension of the vector of explanatory variables, x. Racine (2002) demonstrates that MPI parallelization can be used to speed up calculation of the kernel regression estimator by calculating the fits for portions of the sample on different computers.

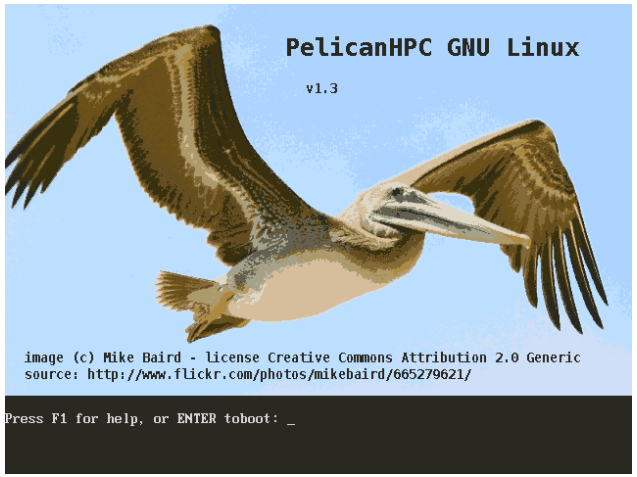
kernel regression, continued

Taken from Michael Creel "User-Friendly Parallel Computations with Econometric Examples" *Computational Economics*, 2005, vol. 26, issue 2, pages 107-128



PelicanHPC

- PelicanHPC is a live CD image that let's you set up a HPC cluster for parallel computing in about 10 minutes.



- The homepage is `http://pareto.uab.es/mcreel/PelicanHPC/` (make sure to look at the tutorial)
- I will use this to present examples. You can get the examples and run them from the CD.

ParallelKnoppix

- You might prefer to work with ParallelKnoppix (`http://pareto.uab.es/mcreel/ParallelKnoppix/`) which is a bit more user friendly.
- However, it is older and no longer developed, and will eventually become obsolete.

Pelican - basic use

- boot up a virtual master node (with home already set up on hard disk)
- do setup, and boot a real compute node
- run a basic example
- encourage students to try it out before the next session

What is going on here?

- the compute nodes boot from the CD image, all nodes have same software
- the /home directory is shared over the network to all nodes
- passwordless ssh is set up between all nodes (shared encrypted keys are used for secure internode connections)
- lamboot is run (needed for the lam/mpi implementation of MPI) **explain this**
- the file /home/user/tmp/bhosts contains the node information, which can be used with OpenMPI, or modified as appropriate

finish first day here?

What is MPI?

- MPI is the "message passing interface". `http://www.mpi-forum.org/` This is a *specification* for a means of parallel computing.
- there are many *implementations* of MPI.
  - LAM/MPI `http://www.lam-mpi.org/` and OpenMPI `http://www.open-mpi.org/` are the ones on PelicanHPC, and are what we will use.
  - An implementation can be optimized for a particular platform. MPI implementations exist for Linux, Windows, MacOS, etc.

MPI is portable

- It will run on a desktop computer, on any of a number of operating systems. You can use all available cores.
- The same code will run on a cluster on PCs
- The same code will run on a supercomputer

Binding functions

- MPI implementations typically provide libraries for Fortran and C/C++
- However, there are numerous collections of bindings for other languages, such as Matlab, Ox, R, Python and GNU Octave.
  - this lets you make calls to MPI functions from the high level languages. The high level language dynamically loads the C functions when needed. The binding functions provide the interface.

GNU Octave and MPITB

- GNU Octave `http://www.gnu.org/software/octave/` is similar to Matlab. Most Matlab code will run on Octave without modification.

Free software (?!)

- Octave is free software. What does "free" mean here?
- most importantly, it means that you (and everyone else) get to have the source code, and you can change it to make it work the way you want to.
- Scientists have an obligation to facilitate replication of their results. Using free software is one means of doing so.
- On an aside, if you have enough interest in this topic to ever read these words, it is almost certainly in your personal interest to switch to free software now. Your productivity will increase.
- "free" also means you don't have to pay for it.
- This is important if you want to run 30 copies on a cluster.

MPITB for GNU Octave

- MPITB is the "message passing interface toolbox" for GNU Octave. `http://atc.ugr.es/javier-bin/mpitb`
- this lets you use MPI in Octave scripts
- this is an example of a set of "binding functions" that make a library of relatively low-level (hard to use) C/Fortran functions available to a high-level (easy to use) package. There are many other examples. MPITB is an outlier for its high quality - it is robust, complete, well-documented, well-maintained, and it offers excellent performance.
- MPITB is developed in Spain! Our tax euros are working!

MPI_Init and MPI_Finalize

- MPI programs start with MPI_Init, and end with MPI_Finalize.
- when using MPITB, we also need to clear variables out of memory between successive calls to MPI_Init,
- see first.m (run these on the Acer laptop, directly, not in VMware - **remember to lamboot**)

MPI_Comm_size and MPI_Comm_rank

- an MPI *communicator* is a set of *ranks* (nodes, processes) that can work in a group. It is isolated from other sets of ranks, so that messages stay within the communicator.
- It is possible to use multiple communicators, but we'll use just one. Its name in MPITB is "MPI_COMM_WORLD"
- the *size* of an MPI communicator is the number of ranks that it holds.
- ranks are processes that run on CPUs. Any number of ranks can run on any number of CPUS. The specific way that ranks are spread over CPUs is part of the design of the parallel program, and can have an important effect on performance. With a homogeneous cluster of single core CPUs, a typical starting point would be to assign one rank to each CPU. For dual core CPUs, one should probably have a rank for each core.
- see second.m

Creating child Octaves

- To increase the size of the communicator, we need to create more threads that run Octave, and integrate them into the communicator. The LAM_Init command will do this for us.
- Using LAM_Init and spawning child Octaves is only one way to use MPITB. We do not have time to explore other options. The method we use here works fairly well, however.
- see third.m **Important reminder to self** - all nodes should be on one machine to see the messages.

NumCmds_Send

- the method uses the NumCmds protocol for communication.
  - NumCmds_Send is used to send Octave objects to all ranks, along with a command to execute
  - after executing a command, the ranks loop, waiting for a new command to execute
- see fourth.m:

Sending and receiving

- once we have more than one rank, we need to know how they can communicate
- the simplest form is point-to-point communication, using MPI_Send and MPI_Recv
- see fifth.m
- you can send Octave objects of any type - **for example, matrices**. See sixth.m

Checking if a message is ready

- sometimes you don't want to block execution to wait for a message to arrive. To check if a message has arrived, you can use MPI_Iprobe (use Octave's help).
- examine and run seventh.m
- this is used in montecarlo.m.

- MPI_Iprobe is used so that results can be received in the order they are ready.
- the frontend node does not participate in the generation of results, it only gathers them.
- for this reason, the number of compute nodes (not counting the frontend) should be set equal to the number of CPU cores available.
  - The frontend will run rank 0, which uses very little CPU time. Mostly it is polling the others to see if they have results.
  - The frontend also runs one of the computational ranks, to keep its CPU busy
  - the computational nodes run a rank on each CPU core they have
- The goal is to keep all CPUs in the cluster fully occupied.

montecarlo_nodes.m

mc_example3

JOURNAL OF APPLIED ECONOMETRICS
J. Appl. Econ. **22**: 215–223 (2007)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/jae.945

I RAN FOUR MILLION PROBITS LAST NIGHT: HPC CLUSTERING WITH PARALLELKNOPPIX

MICHAEL CREEL*
Universitat Autònoma de Barcelona, Spain

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 a given 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] (PK) is a bootable CD that allows users with average computing skills to create an HPC cluster in very little time: about 10 minutes. The computers used in a PK 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 by students during the day can be converted into an HPC cluster for night-time research work, without affecting their use by students the next day.

This note describes what PK is, how it works, and how to set it up and use it. It shows how a Monte Carlo study that involves 4,000,000 nonlinear optimizations may be completed in less than 8 hours on a PK cluster, when it would take roughly 6.25 days on a single computer. The inten-

- who knows what that thing to the R of the logo is?
- look at the code for this

mc_example1.m

mc_example2.m

Homework

- write GNU Octave code to do a Monte Carlo study of the OLS estimator of ρ in the model

$$y_t = \alpha + \rho y_{t-1} + \varepsilon_t$$

- use $n = 30$ observations, with the true parameter values $\alpha = 0; \rho = 0.95$ and $\varepsilon_t \sim \text{IIN}(0,1)$
- your code should make use of montecarlo.m

A real problem and a real cluster

- let's look at a research problem that requires a lot of computational power
- when you have access to computing power your research agenda might change
- once you learn how to do this, there's no reason not to try to get time on a supercomputer like Marenostrum to really increase the scale.

The DLV model

Billio and Monfort (2003)

$$\text{DLV:} \begin{cases} y_t = & r_t\left(y^{t-1}, y^{*t}, \varepsilon_t; \theta\right) \\ y_t^* = & r_t^*\left(y^{t-1}, y^{*t-1}, \varepsilon_t^*; \theta\right) \end{cases} \tag{1}$$

- y^{t-1} is notation for $\left(y_1', ..., y_{t-1}'\right)'$
- $\{\varepsilon_t\}$ and $\{\varepsilon_t^*\}$ are two independent white noises with known distributions
- θ is a vector of unknown parameters

The DLV model

Billio and Monfort (2003)

$$\text{DLV: } \begin{cases} y_t = & r_t\left(y^{t-1}, y^{*t}, \varepsilon_t; \theta\right) \\ y_t^* = & r_t^*\left(y^{t-1}, y^{*t-1}, \varepsilon_t^*; \theta\right) \end{cases} \tag{1}$$

- y^{t-1} is notation for $\left(y_1', ..., y_{t-1}'\right)'$
- $\{\varepsilon_t\}$ and $\{\varepsilon_t^*\}$ are two independent white noises with known distributions
- θ is a vector of unknown parameters

The DLV model

Billio and Monfort (2003)

$$\text{DLV:} \begin{cases} y_t = & r_t\left(y^{t-1}, y^{*t}, \varepsilon_t; \theta\right) \\ y_t^* = & r_t^*\left(y^{t-1}, y^{*t-1}, \varepsilon_t^*; \theta\right) \end{cases} \tag{1}$$

- y^{t-1} is notation for $\left(y_1', ..., y_{t-1}'\right)'$
- $\{\varepsilon_t\}$ and $\{\varepsilon_t^*\}$ are two independent white noises with known distributions
- θ is a vector of unknown parameters

The DLV model

Billio and Monfort (2003)

$$\text{DLV:} \begin{cases} y_t = & r_t\left(y^{t-1}, y^{*t}, \varepsilon_t; \theta\right) \\ y_t^* = & r_t^*\left(y^{t-1}, y^{*t-1}, \varepsilon_t^*; \theta\right) \end{cases} \tag{1}$$

- y^{t-1} is notation for $\left(y_1', ..., y_{t-1}'\right)'$
- $\{\varepsilon_t\}$ and $\{\varepsilon_t^*\}$ are two independent white noises with known distributions
- θ is a vector of unknown parameters

DLV models are often impossible to estimate using classical methods

- Calculation of the likelihood function requires finding the density of y^n
- this involves integrating out all of the y_t^*, of which there are n. It is in general an untractable problem
- Without the density of the observable variables, analytic moments cannot be computed
- Without the density function, maximum likelihood is unavailable
- Without moments, moment-based estimation methods are not available

DLV models are often impossible to estimate using classical methods

- Calculation of the likelihood function requires finding the density of y^n
- this involves integrating out all of the y_t^*, of which there are n. It is in general an untractable problem
- Without the density of the observable variables, analytic moments cannot be computed
- Without the density function, maximum likelihood is unavailable
- Without moments, moment-based estimation methods are not available

DLV models are often impossible to estimate using classical methods

- Calculation of the likelihood function requires finding the density of y^n
- this involves integrating out all of the y_t^*, of which there are n. It is in general an untractable problem
- Without the density of the observable variables, analytic moments cannot be computed
- Without the density function, maximum likelihood is unavailable
- Without moments, moment-based estimation methods are not available

DLV models are often impossible to estimate using classical methods

- Calculation of the likelihood function requires finding the density of y^n
- this involves integrating out all of the y_t^*, of which there are n. It is in general an untractable problem
- Without the density of the observable variables, analytic moments cannot be computed
- Without the density function, maximum likelihood is unavailable
- Without moments, moment-based estimation methods are not available

DLV models are often impossible to estimate using classical methods

- Calculation of the likelihood function requires finding the density of y^n
- this involves integrating out all of the y_t^*, of which there are n. It is in general an untractable problem
- Without the density of the observable variables, analytic moments cannot be computed
- Without the density function, maximum likelihood is unavailable
- Without moments, moment-based estimation methods are not available

Why is SMM inefficient?

- SMM is inefficient for DLV models, since conditional moments can't be used. Why not?
- It's because we can't sample the latent variables conditional on the history of the observed variables. Recall the model:

$$\text{DLV: } \left\{ y_t = \quad r_t\left(y^{t-1}, y^{*t}, \varepsilon_t; \theta\right) \right. \tag{2}$$

  If we could sample from $y^{*t}|y^{t-1}$, we could substitute the draw into the DLV to get a draw from $y_t|y^{t-1}$

- For Markovian models, the Markov chain Monte Carlo method can be used to sample from $y^{*t}|y^{t-1}$, since in this case information about the distant past is not needed to sample the latent variables. Fiorentini, Sentana and Shephard (2004) is an example.

Why is SMM inefficient?

- SMM is inefficient for DLV models, since conditional moments can't be used. Why not?
- It's because we can't sample the latent variables conditional on the history of the observed variables. Recall the model:

$$\text{DLV: } \begin{cases} y_t = & r_t\left(y^{t-1}, y^{*t}, \varepsilon_t; \theta\right) \end{cases} \tag{2}$$

If we could sample from $y^{*t}|y^{t-1}$, we could substitute the draw into the DLV to get a draw from $y_t|y^{t-1}$

- For Markovian models, the Markov chain Monte Carlo method can be used to sample from $y^{*t}|y^{t-1}$, since in this case information about the distant past is not needed to sample the latent variables. Fiorentini, Sentana and Shephard (2004) is an example.

Why is SMM inefficient?

- SMM is inefficient for DLV models, since conditional moments can't be used. Why not?
- It's because we can't sample the latent variables conditional on the history of the observed variables. Recall the model:

$$\text{DLV: } \left\{ y_t = \quad r_t\left(y^{t-1}, y^{*t}, \varepsilon_t; \theta\right) \right. \tag{2}$$

If we could sample from $y^{*t}|y^{t-1}$, we could substitute the draw into the DLV to get a draw from $y_t|y^{t-1}$

- For Markovian models, the Markov chain Monte Carlo method can be used to sample from $y^{*t}|y^{t-1}$, since in this case information about the distant past is not needed to sample the latent variables. Fiorentini, Sentana and Shephard (2004) is an example.

The GMM estimator

- Error functions are of the form

$$\varepsilon(y_t, x_t; \theta) = y_t - \phi(x_t; \theta), \tag{3}$$

- Moment conditions are defined by interacting a vector of instrumental variables $z(x_t)$ with error functions:

$$m(y_t, x_t; \theta) = z(x_t) \otimes \varepsilon(y_t, x_t; \theta) \tag{4}$$

- Average moment conditions are

$$m_n(Z_n; \theta) = \frac{1}{n} \sum_{t=1}^{n} m(y_t, x_t; \theta) \tag{5}$$

- The objective function is

$$s_n(Z_n; \theta) = m_n'(Z_n; \theta) W(\hat{\tau}_n) m_n'(Z_n; \theta) \tag{6}$$

The GMM estimator

- Error functions are of the form

$$\varepsilon(y_t, x_t; \theta) = y_t - \phi(x_t; \theta), \tag{3}$$

- Moment conditions are defined by interacting a vector of instrumental variables $z(x_t)$ with error functions:

$$m(y_t, x_t; \theta) = z(x_t) \otimes \varepsilon(y_t, x_t; \theta) \tag{4}$$

- Average moment conditions are

$$m_n(Z_n; \theta) = \frac{1}{n} \sum_{t=1}^{n} m(y_t, x_t; \theta) \tag{5}$$

- The objective function is

$$s_n(Z_n; \theta) = m_n'(Z_n; \theta) W(\hat{\tau}_n) m_n'(Z_n; \theta) \tag{6}$$

The GMM estimator

- Error functions are of the form

$$\varepsilon(y_t, x_t; \theta) = y_t - \phi(x_t; \theta), \tag{3}$$

- Moment conditions are defined by interacting a vector of instrumental variables $z(x_t)$ with error functions:

$$m(y_t, x_t; \theta) = z(x_t) \otimes \varepsilon(y_t, x_t; \theta) \tag{4}$$

- Average moment conditions are

$$m_n(Z_n; \theta) = \frac{1}{n} \sum_{t=1}^{n} m(y_t, x_t; \theta) \tag{5}$$

- The objective function is

$$s_n(Z_n; \theta) = m_n'(Z_n; \theta) W(\hat{\tau}_n) m_n'(Z_n; \theta) \tag{6}$$

The GMM estimator

- Error functions are of the form

$$\varepsilon(y_t, x_t; \theta) = y_t - \phi(x_t; \theta), \tag{3}$$

- Moment conditions are defined by interacting a vector of instrumental variables $z(x_t)$ with error functions:

$$m(y_t, x_t; \theta) = z(x_t) \otimes \varepsilon(y_t, x_t; \theta) \tag{4}$$

- Average moment conditions are

$$m_n(Z_n; \theta) = \frac{1}{n} \sum_{t=1}^{n} m(y_t, x_t; \theta) \tag{5}$$

- The objective function is

$$s_n(Z_n; \theta) = m_n'(Z_n; \theta) W(\hat{\tau}_n) m_n'(Z_n; \theta) \tag{6}$$

The SNM estimator

- Consider a long simulation from the DLV model, $\widetilde{Z}_S$.
- Kernel regression may be used to fit $\phi(x_t;\theta)$, using this simulated data

$$\widetilde{\phi}_S(x_t;\widetilde{Z}_S(\theta)) = \sum_{s=1}^{S} \widetilde{w}_s \widetilde{y}_s(\theta) \tag{7}$$

- the weight $\widetilde{w}_s$ is

$$\widetilde{w}_s = \frac{K\left(\frac{x_t - \widetilde{x}_s(\theta)}{h_S}\right)}{\sum_{s=1}^{S} K\left(\frac{x_t - \widetilde{x}_s(\theta)}{h_S}\right)} \tag{8}$$

- $\widetilde{\phi}_S(x_t;\widetilde{Z}_S(\theta)) \xrightarrow{a.s.} \phi(x_t,\theta)$, for almost all x_t, as $S \to \infty$.
- S can be made as large as we like! (Jump back to last slide)

The SNM estimator

- Consider a long simulation from the DLV model, $\widetilde{Z}_S$.
- Kernel regression may be used to fit $\phi(x_t;\theta)$, using this simulated data

$$\widetilde{\phi}_S(x_t;\widetilde{Z}_S(\theta)) = \sum_{s=1}^{S} \widetilde{w}_s \widetilde{y}_s(\theta) \tag{7}$$

- the weight $\widetilde{w}_s$ is

$$\widetilde{w}_s = \frac{K\left(\frac{x_t - \widetilde{x}_s(\theta)}{h_S}\right)}{\sum_{s=1}^{S} K\left(\frac{x_t - \widetilde{x}_s(\theta)}{h_S}\right)} \tag{8}$$

- $\widetilde{\phi}_S(x_t;\widetilde{Z}_S(\theta)) \xrightarrow{a.s.} \phi(x_t,\theta)$, for almost all x_t, as $S \to \infty$.
- S can be made as large as we like! (Jump back to last slide)

The SNM estimator

- Consider a long simulation from the DLV model, $\widetilde{Z}_S$.
- Kernel regression may be used to fit $\phi(x_t;\theta)$, using this simulated data

$$\widetilde{\phi}_S(x_t;\widetilde{Z}_S(\theta)) = \sum_{s=1}^{S} \widetilde{w}_s \widetilde{y}_s(\theta) \tag{7}$$

- the weight $\widetilde{w}_s$ is

$$\widetilde{w}_s = \frac{K\left(\frac{x_t - \widetilde{x}_s(\theta)}{h_S}\right)}{\sum_{s=1}^{S} K\left(\frac{x_t - \widetilde{x}_s(\theta)}{h_S}\right)} \tag{8}$$

- $\widetilde{\phi}_S(x_t;\widetilde{Z}_S(\theta)) \xrightarrow{a.s.} \phi(x_t,\theta)$, for almost all x_t, as $S \to \infty$.
- S can be made as large as we like! (Jump back to last slide)

The SNM estimator

- Consider a long simulation from the DLV model, $\widetilde{Z}_S$.
- Kernel regression may be used to fit $\phi(x_t;\theta)$, using this simulated data

$$\widetilde{\phi}_S(x_t;\widetilde{Z}_S(\theta)) = \sum_{s=1}^{S} \widetilde{w}_s \widetilde{y}_s(\theta) \tag{7}$$

- the weight $\widetilde{w}_s$ is

$$\widetilde{w}_s = \frac{K\left(\frac{x_t - \widetilde{x}_s(\theta)}{h_S}\right)}{\sum_{s=1}^{S} K\left(\frac{x_t - \widetilde{x}_s(\theta)}{h_S}\right)} \tag{8}$$

- $\widetilde{\phi}_S(x_t;\widetilde{Z}_S(\theta)) \xrightarrow{a.s.} \phi(x_t,\theta)$, for almost all x_t, as $S \to \infty$.
- S can be made as large as we like! (Jump back to last slide)

The SNM estimator

- Consider a long simulation from the DLV model, $\widetilde{Z}_S$.
- Kernel regression may be used to fit $\phi(x_t;\theta)$, using this simulated data

$$\widetilde{\phi}_S(x_t;\widetilde{Z}_S(\theta)) = \sum_{s=1}^{S} \widetilde{w}_s \widetilde{y}_s(\theta) \tag{7}$$

- the weight $\widetilde{w}_s$ is

$$\widetilde{w}_s = \frac{K\left(\frac{x_t - \widetilde{x}_s(\theta)}{h_S}\right)}{\sum_{s=1}^{S} K\left(\frac{x_t - \widetilde{x}_s(\theta)}{h_S}\right)} \tag{8}$$

- $\widetilde{\phi}_S(x_t;\widetilde{Z}_S(\theta)) \overset{a.s.}{\to} \phi(x_t,\theta)$, for almost all x_t, as $S \to \infty$.
- S can be made as large as we like! (Jump back to last slide)

The SNM estimator *is* the GMM estimator if S is large enough

Classical linear model

$$\text{Linear Model:} \quad \begin{cases} y & = \beta_1 + \beta_2 x + \varepsilon \\ x & \sim U(0,1) \\ \varepsilon & \sim N(0,1) \end{cases} \tag{9}$$

- SNM estimation with $S = 500000$, 1000 Monte Carlo reps

The SNM estimator *is* the GMM estimator if S is large enough

Classical linear model

$$\text{Linear Model:} \quad \begin{cases} y & = \beta_1 + \beta_2 x + \varepsilon \\ x & \sim U(0,1) \\ \varepsilon & \sim N(0,1) \end{cases} \tag{9}$$

- SNM estimation with $S = 500000$, 1000 Monte Carlo reps

- $$\widehat{\beta}_1(SNM) = \underset{(0.00023012)}{-0.00106912} + \underset{(0.00030566)}{1.00292}\,\widehat{\beta}_1(GMM) - \underset{(0.00050332)}{0.00267236}\,\beta_1$$
$$T = 1000 \quad \bar{R}^2 = 0.9999 \quad F(2,997) = 8.5632\text{e+}6 \quad \hat{\sigma} = 0.0036169$$
(standard errors in parentheses)

- $$\widehat{\beta}_2(SNM) = \underset{(0.00038392)}{2.50475\text{e-}5} + \underset{(0.00029626)}{1.00389}\,\widehat{\beta}_2(GMM) - \underset{(0.00073023)}{0.000178451}\,\beta_2$$
$$T = 1000 \quad \bar{R}^2 = 0.9999 \quad F(2,997) = 6.9636\text{e+}6 \quad \hat{\sigma} = 0.0061481$$
(standard errors in parentheses)

- $$\widehat{\beta}_1(SNM) = \underset{(0.00023012)}{-0.00106912} + \underset{(0.00030566)}{1.00292}\,\widehat{\beta}_1(GMM) - \underset{(0.00050332)}{0.00267236}\,\beta_1$$
  $$T = 1000 \quad \bar{R}^2 = 0.9999 \quad F(2,997) = 8.5632\text{e+}6 \quad \hat{\sigma} = 0.0036169$$
  (standard errors in parentheses)

- $$\widehat{\beta}_2(SNM) = \underset{(0.00038392)}{2.50475\text{e-}5} + \underset{(0.00029626)}{1.00389}\,\widehat{\beta}_2(GMM) - \underset{(0.00073023)}{0.000178451}\,\beta_2$$
  $$T = 1000 \quad \bar{R}^2 = 0.9999 \quad F(2,997) = 6.9636\text{e+}6 \quad \hat{\sigma} = 0.0061481$$
  (standard errors in parentheses)

SNM is computationally demanding

- to fit n points using S simulated points, we need to do approximately knS calculations
- when S is very large, this is a lot of work
- all of this is required for a single evaluation of the GMM criterion function. When this is embedded in a quasi-Newton method, many evaluations are required.
- the GMM criterion is not globally convex - need to use something like simulated annealing to avoid local minima, then use quasi-Newton to refine
- doing Monte Carlo adds a layer of computations

Check the Pelican cluster at the UAB

- made up of two servers, each has two quad core CPUs.
  - A total of 16 cores. 1 server runs the PelicanHPC CD, the other is netbooted from the first
  - cost was about 4500 euros
  - keeping CPU density high is important: energy consumption and the price you pay for housing are highly correlated with the number of things that need to be plugged in. Also, the on-machine inter-core communications is a lot faster than the between-machine communition. Minimize computers - mazimize cores!
- ssh to pareto, then to pelican
- run estimate_fg, and look at htop on second screen
- run mc.m for SV1, look at htop on second screen

Using PelicanHPC

- PelicanHPC is an image file for a bootable CDROM.
  - You can boot a computer using the CDROM
  - using the CDROM does not affect anything installed on your computer, when you reboot, your computer is just as it was before using PelicanHPC
- PelicanHPC allows you to create a cluster for parallel computing using MPI
- other computers in the cluster are booted using their ethernet interfaces
  - netboot must be enabled for this to work, It is an option in the BIOS setup routine
- basic instructions for using PelicanHPC are at `http://pareto.uab.es/mcreel/PelicanHPC/Tutorial/PelicanTutorial.html`

A virtual cluster

- this uses VMware server (`http://www.vmware.com/download/server/`). This is free (as in free beer) software that works on Linux and Windows.
- it lets you use Pelican without actually rebooting your computer
- the virtual frontend node can be used to boot a cluster of real compute nodes
- the compute nodes can also be virtual: it is possible to have an entirely virtual PelicanHPC cluster running on a set of Windows machines
- go on to demonstrate Pelican, running the Monte Carlo examples.

make_pelican

- PelicanHPC ISO images are made by running a single script: make_pelican
- if you use Debian Linux (or a derivative), you can install the live_helper package, and then use the script to make your own version
  - this allows you to specify a password, select software packages you want, place the home directory on permanent storage, etc.
  - this is probably not something to try until you are more familiar with the basics - it's just information to be aware of in case you become interested

GMM

- gmm_results.m
  http://pareto.uab.es/mcreel/Econometrics/MyOctaveFiles/Econometrics/GMM/gmm_results.m
- gmm_estimate.m
  http://pareto.uab.es/mcreel/Econometrics/MyOctaveFiles/Econometrics/GMM/gmm_estimate.m
- gmm_obj.m
  http://pareto.uab.es/mcreel/Econometrics/MyOctaveFiles/Econometrics/GMM/gmm_obj.m
- average_moments.m
  http://pareto.uab.es/mcreel/Econometrics/MyOctaveFiles/Econometrics/GMM/average_moments.m
- sum_moments_nodes.m
  http://pareto.uab.es/mcreel/Econometrics/MyOctaveFiles/Econometrics/GMM/sum_moments_nodes.m

Solving macroeconomic models

An example is a forthcoming paper

Comput Econ
DOI 10.1007/s10614-008-9142-6

Using Parallelization to Solve a Macroeconomic Model: A Parallel Parameterized Expectations Algorithm

Michael Creel

Accepted: 2 April 2008

Abstract Solving nonlinear macroeconomic models with rational expectations can be time-consuming. This paper shows how the parameterized expectations algorithm (PEA) can be parallelized to reduce the time needed to solve a simple model by more than 80%. The general idea of using parallelization applies naturally to other algorithms, as well. This paper is illustrative of the speedup that can be obtained, and it provides computer code that may serve as an example for parallelization of other algorithms. For those who would like to use the parallelized PEA, the implementation does not confront end users with the details of parallelization. To solve a model, it is only necessary to provide ordinary serial code that simulates data from the model. All needed code is available, on a standalone basis, or pre-installed on ParallelKnoppix (Creel, J Appl Economet 22:215–223, 2007).

Keywords Solving macroeconomic models · Parameterized expectations algorithm · Parallel computing · Nonlinear rational expectations

PEA

- uses a long simulation from a model, conditional on a parameterized model of expectations
- the parameterized model of expectations is fit to the generated data
- when the generated data is consistent with the model of expectations, the model has been solved
- both making a long simulation and fitting the model of expectations can easily be parallelized
- is is possible to speed up a simple model by more than 80%. More complicated models should give even better results

Speedups

Table 1 Time to complete 30 iterations (T = 200,000 periods)[a]

Nodes	10 node real cluster	4 node virtual cluster
1	93.55	97.13
2	0.54	0.59
3	0.40	0.41
4	0.31	0.33
5	0.25	na
6	0.21	na
7	0.19	na
8	0.17	na
9	0.16	na
10	0.15	na

[a] For 1 node, the reported time is absolute (s). For more than 1 node, the reported time is relative to the timing for the first node

Go look at the code

- pea_example.m `http://pareto.uab.es/mcreel/Econometrics/Examples/Parallel/pea/example/pea`
- note use of C++ to write the model
- run the examples

Practical projects

The last two hours will be used for discussion and working on the following projects. Please bring your laptop computer! You should have done some parallel computing by the time we are done!

make a real cluster

- Create a real cluster: several students, and several laptop computers. Need a copy of the Pelican CD and a switch

make a virtual cluster

- Create an entirely virtual cluster: at least 1 student with a laptop computer.
- this is more difficult than making a real cluster, and more useful for learning,
- Need a copy of VMware server and the Pelican ISO image.
  1. Install VMware server to allow bridged, NAT and host only networking
  2. create a virtual machine with 2 NICs, first bridged, second NAT. Use it to boot CD image
  3. create a virtual machine with bridged NIC, use it to PXE boot the compute node

Run the examples

- requires a single (hopefully multicore) computer
  - Monte Carlo study: boot up Pelican and use only the frontend. Do the AR1 Monte Carlo suggested above.
  - Do GMM estimation: boot up Pelican and use only the frontend. Make hausman.m run on 2 ranks.

  run pea_example.m

Please feel free to ask questions!

- Write me directly at michael.creel@uab.es, or use the PelicanHPC forum.
- I understand perfectly that this is a big jump for most of you, so please don't worry that your questions are silly. 4 years ago I was asking the same questions.
- If something doesn't work as advertised, I would really appreciate that you let me know.