Eleventh Floor, Menzies Building
Monash University, Wellington Road
CLAYTON Vic 3800 AUSTRALIA

Telephone:	from overseas:
(03) 9905 2398, (03) 9905 5112	61 3 9905 2398 or 61 3 9905 5112
Fax:	
(03) 9905 2426	61 3 9905 2426
e-mail:	impact@buseco.monash.edu.au
Internet home page:	http//www.monash.edu.au/policy/

Paper presented at the 9th Conference on
Global Economic Analysis
Addis Ababa, Ethiopia, June 2006

Running Simulations Faster on Multi-Processor or 64-Bit PCs via GEMPACK

by

J. Mark HORRIDGE

And

Ken PEARSON

Centre of Policy Studies
Monash University

Computing Document No. C15-01 April 2006

ISSN 1 031 9034 ISBN 0 7326 1562 3

Running Simulations Faster on Multi-Processor or 64-Bit PCs via GEMPACK

J. Mark Horridge and Ken Pearson

Abstract

Many modern Windows PCs now have two or more processors. These PCs cost little more than a PC with a single processor.

If you are solving a model accurately using GEMPACK, you will usually extrapolate from 2 or 3 multi-step calculations (for example, from Gragg 2-step, 4-step and 6-step calculations). The separate multi-step calculations are independent of each other. For example, if you are doing Gragg 2,4,6-step calculations, the 6-step calculation can be done independently of the 4-step calculation.

If you wish to solve a model by extrapolating from 2 or 3 multi-step calculations, you can now ask GEMPACK to carry out one or two of these multi-step calculations in parallel. In this way you may be able to solve the model significantly more quickly than previously.

Suppose, for example, you are solving using Gragg 2,4,6-step calculations. If you have two processors, you can ask the main program to carry out the 2-step and 4-step calculations. And you can ask the main program to run (on the other processor) a separate job which carries out the 6-step calculation. When that 6-step calculation is finished, the main program will read the results from the 6-step calculation and then combine these results with the results of the 2-step and 4-step calculations to do the extrapolation and produce the usual results. We refer to the main program as the "master" and to the run which carries out the 6-step calculation in parallel as "the servant". Since the 6-step calculation probably only takes as long as the 2-step followed by the 4-step, this will approximately halve the total elapsed time required to solve the model.

If you have more than two processors, you can ask the master to run two servant programs.

When you run two or three calculations in parallel (one master and one or two servants), each program requires about the same amount of memory. If your model takes about 500MB of memory to solve normally, you will need about 1000MB to run one servant or about 1500MB to run two servants. If you don't have this much memory, the servants and/or the master will be running in virtual memory (which is very slow).

The new 64-bit processors (combined with a 64-bit version of Windows) are interesting in this context since they are able to address more than 2GB of memory. [2 gigabytes of memory is effectively the maximum which can be addressed by any program under 32-bit Windows XP and Windows 2000.]

The version of GEMPACK which allows master/servant solving is available as a best test version for customers from organisations which have a multi-user Source-Code Version of GEMPACK with expiring annual licences.

TABLE OF CONTENTS

1. Press Release

The following "press release" summarises the situation well.

WHEN TWO HEADS ARE BETTER THAN ONE: PROFESSORS PROMOTE PARALLEL PROCESSING

Professors Kenneth and Robert Pearson of Monash University's BusEco's Centre of Policy Studies have developed a method of speeding up complex economic calculations.

Their software system, GEMPACK, is used to solve many of the world's biggest economic models. Complex simulations can take many hours to complete.

Modern PC's often contain two CPUs, effectively doubling processing power. The challenge is to design software that can take advantage of both CPUs at once.

"We divided the calculations into parts which could be executed independently by either processor -- so halving compute time," chorused the twins. "Faster computations will enable economists to conduct more simulations, or to solve even bigger models," claimed Robert. "Or they could spend more time playing golf," echoed Ken.

2. Solving in Parallel

Many modern Windows PCs now have two or more processors. These PCs cost little more than a PC with a single processor.

If you are solving a model accurately using GEMPACK, you will usually extrapolate from 2 or 3 multi-step calculations (for example, from Gragg 2-step, 4-step and 6-step calculations). The separate multi-step calculations are independent of each other. For example, if you are doing Gragg 2,4,6-step calculations, the 6-step calculation can be done independently of the 4-step calculation.

We are grateful to Hom Pant who pointed out that these calculations are independent of each other and who encouraged us to implement parallel calculations to take advantage of this.

If you wish to solve a model by extrapolating from 2 or 3 multi-step calculations, you can now ask GEMPACK to carry out one or two of these multi-step calculations in parallel. In this way you may be able to solve the model significantly more quickly than previously.

Suppose, for example, you are solving using Gragg 2,4,6-step calculations. If you have two processors, you can ask the main program to carry out the 2-step and 4-step calculations. And you can ask the main program to run (on the other processor) a separate job which carries out the 6-step calculation. When that 6-step calculation is finished, the main program will read the results from the 6-step calculation and then combine these results with the results of the 2-step and 4-step calculations to do the extrapolation and produce the usual results. We refer to the main program as the "master" and to the run which carries out the 6-step calculation in parallel as "the servant". Since the 6-step calculation probably only takes as long as the 2-step followed by the 4-step, this will approximately halve the total elapsed time required to solve the model.

If you have more than two processors, you can ask the master to run two servant programs.

Of course, there is no gain from using a master and servants if you are only carrying out a single multi-step calculation – for example, if you are solving the model using just an Euler 4-step calculation.

This ability to run separate multi-step calculations in parallel is a new GEMPACK feature which will be available in Release 10 of GEMPACK (see section 5).

When you run two or three calculations in parallel (one master and one or two servants), each program requires about the same amount of memory. If your model takes about 500MB of memory to solve normally, you will need about 1000MB to run one servant or about 1500MB to run two servants. If you don't have this much memory, the servants and/or the master will be running in virtual memory (which is very slow).

The new 64-bit processors (combined with a 64-bit version of Windows) are interesting in this context since they are able to address more than 2GB of memory. [2 gigabytes of memory is effectively the maximum which can be addressed by any program under 32-bit Windows XP and Windows 2000.]

If your model takes about 1.5GB of memory to solve normally, there is no point in running a servant as well under 32-bit Windows XP since the master and servant will be competing for memory. But, if you are running 64-bit Windows XP on a 64-bit PC and have say 4GB of memory, you can happily run one master and one servant (each using 1.5GB) simultaneously. Or, if you have more than two processors and 6GB of memory, you can happily run a master and two servants (each accessing 1.5GB of memory). And this remains true even if the EXEs you are running are made with a 32-bit compiler such as the Lahey compiler LF95.

The version of GEMPACK which allows master/servant solving is available as a best test version for customers from organisations which have a multi-user Source-Code Version of GEMPACK with expiring annual licences (see section 5 for details).

2.1 Several Subintervals, Automatic Accuracy, Complementarities

Running a master and one or two servants is allowed in more complicated situations including

- when you have two or more subintervals. Then each servant is called to carry out the multi-step calculation it is responsible for on each subinterval.

- when you are using automatic accuracy. As when there are several subintervals, each servant is called to carry out the multi-step calculation it is responsible for on each subinterval. The master controls the testing of the accuracy (has it been sufficient?) on each subinterval. If the values of a Coefficient go out of range, the master is informed and repeats the subinterval as usual.
- when there are Complementarity statements in the model. Then the master has to carry out all of the approximate run in each subinterval. The master enlists the aid of servants when doing the accurate run in each subinterval (since this probably requires two or three separate multi-step calculations). The time saving will be less in this case since the master must do all of the approximate run. There is only time saving on the accurate run.

2.2 Multi-Processor PCs

For some years Windows PCs have been available which incorporated 2 or more CPUs. The PC's motherboard has 2 or 4 CPU sockets. Windows NT (and now XP) is able to schedule tasks to either processor. Specially written programs (as described here) are able to speed execution by using both processors.

These traditional multi-processor PC's are usually large, noisy and expensive, with a rather limited market (server applications). Not much software was adapted to use multiple CPUs.

Recently, dual-core processors which incorporate 2 processing units in a single package have become popular. AMD has led the market, producing 2-processor chips which are plug-in replacements for existing (single-processor) CPUs. No special motherboard is required, leading to a cheap, quiet PC with multi-processing capacity.

Intel is rushing to catch up, and has already released a dual-core version of its popular Pentium M laptop CPU (christened "Core Duo"). Desktop versions will follow. Within 1 or 2 years the majority of new PCs will contain such chips. 4-core and even 8-core chips are planned. Multi-processing, once a niche specialty, is becoming ubiquitous.

One reason for this trend is that traditional single-core CPUs have reached a performance plateau (or at least improve more slowly than before). Multi-processing offers the prospect of continued performance increases – if software can take advantage of it.

Although Windows allows for virtual memory (when the hard disk emulates RAM), good performance requires that each running task has ample access to RAM. However, ordinary 32-bit CPUs and 32-bit Windows can only manage 4GB of RAM. This limits the possibility of running several memory-intensive tasks simultaneously. 64-bit CPUs and Windows versions are now available which can manage much more memory. Indeed, all the dual-core AMD chips are 64-bit capable. Hence, there is a connection between the transition to multicore PCs and the transition to 64-bit computing. That connection is explored in section 3 below.[1]

2.3 Telling the Program to Run in Parallel

You can do this by including the following statement in your Command file.

```
servants = NO|1|2 ;         ! NO is the default
```

- "**servants = 1 ;**" tells the program to use one servant. That will do the longest multi-step calculation.
- "**servants = 2 ;**" tells the program to use two servants. Those two servants will do the longer two multi-step calculations. This only makes sense if you have three or more processors on your machine since the main program will use one processor and each of the servants ideally has its own processor. This only makes sense if you are extrapolating from three (not two) multi-step calculations.

[1] When you run computational-intensive and memory-intensive jobs simultaneously on two or more processors, it is important that each processor has quick access to the part of memory allocated to the task it is performing. In some cases, each processor has its own channel to memory while in other cases, two processors may share a common channel to memory.

- "**servants = no ;**" tells the program not to use servants. That may be appropriate even if you have two processors because you want to do serious word-processing while the simulation is running and you want the second processor to concentrate on the word processing.

2.3.1 Problems with the Program Deciding How Many Servants

The program may be able to tell how many processors are available on your PC (though this is not easy to find out reliably).

The program could find out how much memory is available on your PC. However, the program cannot know early on how much memory will be required to solve your model. Accordingly, the program cannot sensibly decide whether you have sufficient memory for one or more servants.

That is why, at least for the present, we require you to tell the program whether or not to use servants and, if so, how many servants to use.

2.4 Some Elapsed Times

In this section we report some elapsed times with and without servants for various models with different compilers.

All times reported here were obtained on a Windows XP PC with two dual-core AMD64 chips[2] (that is, 4 processors) with 8Gb of physical memory.

The times for the LF90, LF95 and Intel-32 compilers were obtained running under the Windows XP Professional (32-bit) operating system. The times for the Intel-64 compiler were obtained running under the Windows XP Professional x64 Edition (a 64-bit operating system).

In each case, the default compiler options as supplied with Release 10 of GEMPACK were used. For LF90 and LF95 this is basically O1 optimisation options (the same as for Release 9.0 of GEMPACK with these compilers). For the Intel compilers, this is /O2 optimisation option (which is also the default recommended by Intel).

The times reported are elapsed times. Even for the same compiler and Command file, the elapsed time varies from run to run.[3] The times reported are the average of 3 runs in each case.

- The version of **GTAP** is one with 38 regions and 39 tradeable commodities. This is the same 38x39 aggregation we reported about in sections 4.2.9, 5.2.1 and 5.3.1 of the Release 9 version of GPD-5. The simulation in **SIM1.CMF** is a Gragg 2,4,6-step simulation.
- The version of **GTEM** used for the report below has 23 regions and 29 tradeable commodities. [This is the same version of GTEM we reported about in section 5.3.1 of the Release 9 version of GPD-5.] **GTEMSIM3.CMF** is an Euler 3,5,7-step simulation. We are grateful to Guy Jakeman for supplying us with this version of ABARE's GTEM model.
- The **TERM-WATER** version is a version of TERM with 48 commodities and 20 regions aimed at water-related applications. This is the TERM model reported in the tables in sections 5.2.1 and 5.3 of the Release 9 version of GPD-5. This model contains Complementarity statements. **TERMSIM1.CMF** does 3-step Euler approximate run followed by Euler 2,3,4-step accurate run. There are no Complementarity state changes in this simulation.

[2] Two Dual Core AMD Opteron Processors 270 2.01GHz.

[3] For example, with Intel 32, the 3 elapsed times for the 2-servant GTAP simulation were 24 m 48s, 21m 58s and 23m 40s, while for Intel 64, the 3 elapsed times for the same 2-servant simulation were 21m 31s, 21m 30s and 21m 36s.

Simulation and Compiler	**Simulation (no servants)**	**1 Servant**	**2 Servants**	**Longest multi-step calculation**
GTAP38X39 SIM1.CMF	**Gragg 2,4,6**	**Gragg 2,4,6**	**Gragg 2,4,6**	**Gragg 6**
LF90	48m 35s[4]	32m 18s	26m 39s	20m 57s
LF95	66m 24s	35m 19s	34m 58s	33m 58s
Intel 32	48m 10s	26m 28s	23m 49s	23m 40s
Intel 64	41m 23s	24m 55s	21m 32s	20m 48s
GTEM GTEMSIM3.CMF	**Euler 3,5,7**	**Euler 3,5,7**	**Euler 3,5,7**	**Euler 7**
LF90	17m 31s	10m 51s	10m 21s	10m 14s
LF95	46m 8s	25m 32s	24m 57s	24m 19s
Intel 32	19m 43s	11m 3s	10m 57s	10m 45s
Intel 64	18m 20s	10m 39s	10m 38s	9m 54s
TERM-WATER TERMSIM1.CMF	**Euler 2,3,4 (plus 3-step approx run)**	**Euler 2,3,4 (plus 3-step approx run)**	**Euler 2,3,4 (plus 3-step approx run)**	**Euler 4 (plus 3-step approx run)**
LF90	6m 33s	4m 38s	4 m 32s	4 m 33s
LF95	7m 34s	5m 28s	5m 32s	5m 15s
Intel 32	5m 23s	3m 38s	3m 27s	3m 37s
Intel 64	3m 58s	2m 50s	2m 50s	2m 45s

Table 2.4 : Elapsed Times for Typical Simulations With and Without Servants

2.4.1 Comments on the Elapsed Times Reported in Table 2.4

The best you can hope for is that, by using 2 servants, the elapsed time will be the same as the elapsed time for the longest of the 3 separate calculations. As you can see from the last two columns in Table 2.4, that pretty much happens (except for the LF90 GTAP simulation).

With one servant, the master does the shorter two multi-step calculations while the servant does the longest multi-step calculation.

In the case of the Gragg 2,4,6 GTAP simulation, this means that the master does something like 2+4=6 passes while the servant does 6 passes. [A Gragg 4-step has 5 passes. The first pass of all multi-step calculations is done by the master before the servant starts. So the servant does the last 6 passes of the 7-pass Gragg 6-step while, starting at the same time, the master does the last 2 passes of the Gragg 2-step and then the last 4 passes of the Gragg 4-step.] On the basis of this (crude) analysis, you might expect that the master would finish its two tasks (Gragg 2 and 4) at about the same time as the servant finished its one task (Gragg 6). In fact, in this simulation, the master takes longer than the servant because of variations between the times taken for the different steps.[5]

The TERM-WATER simulation has a 3-step Euler approximate run followed by an Euler 2,3,4-step accurate run. The master does all of the approximate run (the servants are not able to help with that). Hence the overall time saving is not as great as for the other simulations reported.

[4] This is an abbreviation for 48 minutes, 35 seconds.

[5] For the Intel-32 compiler, the single step times taken through this simulation vary from about 2m 35s to about 4m 56s. In this case, it just happens that more of the longer ones occur in the parts done by the master. For another simulation, the reverse might happen.

Summary

- If you have 3 or more processors and run with 2 servants, you can expect that the elapsed time will be roughly that for the longest multi-step calculation.
- If you have only 2 processors (or run with only one servant), you still can expect to make considerable time savings. And you can estimate roughly how much by considering how many total passes the master and servant must each make (once the servant is started).

3. 64-bit Processors, Operating Systems on Windows PCs

Until recently the mainstream desktop PC environment has been a Pentium-compatible 32-bit (IA-32) processor running Microsoft Windows. Ordinary 32-bit Windows supports up to 4Gb RAM, although each running program may use at most 2Gb (in practice 1.5Gb) for data.

Intel promoted the **Itanium** 64-bit (**IA64**) processor, and Microsoft provided a special version of Windows (Server 2003 Itanium) to run it. However, the Itanium is internally quite different from the Pentium and so existing 32-bit programs (compiled for 32-bit Windows) can run only relatively slowly, using an emulation layer. The difficulty of using older programs has restricted the Itanium to a shrinking niche market.

Rival firm AMD has produced **Opteron** and **Athlon** 64-bit processors which, like the Itanium, can access much more than 2Gb RAM. The AMD chips, however, are closer to the older Pentium architecture, and offer a 32-bit mode which runs older 32-bit programs at full speed. AMD calls this CPU architecture X-86-64 or **AMD64**.

Intel has rushed to imitate the AMD64 system, which it terms **IA32e** or **EM64T**. Both AMD and Intel chips are available in dual-core versions. In performance, the Intel chips almost match AMD (but they fry eggs much faster). To software, the AMD64 and EM64T chips appear the same. Their high backwards compatibility allows AMD64/EM64T chips to run 32-bit Windows very well -- but without any 64-bit advantages.

Now Microsoft offers "x64" editions of Windows Server 2003 and Windows XP to suit AMD64 or EM64T chips. Programs compiled especially for x64 Windows can access all available RAM (the 2 Gb limit is broken). But even older, 32-bit, programs run as fast under x64 as they did in 32-bit Windows. Indeed x64 Windows is the best way to simultaneously run several 32-bit programs which each use up to 2Gb of RAM. In total, 16 or 128 Gb of RAM may be used.

In common parlance, 64-bit now refers to AMD64 or EM64T chips and x64 Windows, not to the Itanium IA64 and its special versions of Windows.

The main benefit of 64-bit technology is the greater RAM access – but it offers other potential advantages. For example, the speed penalty for using double-precision is reduced (but not removed). There are more high-speed registers, which a very few especially-tuned programs may use. But in general we would not expect that a program originally compiled for a 32-bit CPU would run faster when re-compiled for x64. Indeed, the 64-bit program will require a little more RAM than its 32-bit counterpart.

4. Intel Fortran Compiler (32-bit and 64-bit) Supported

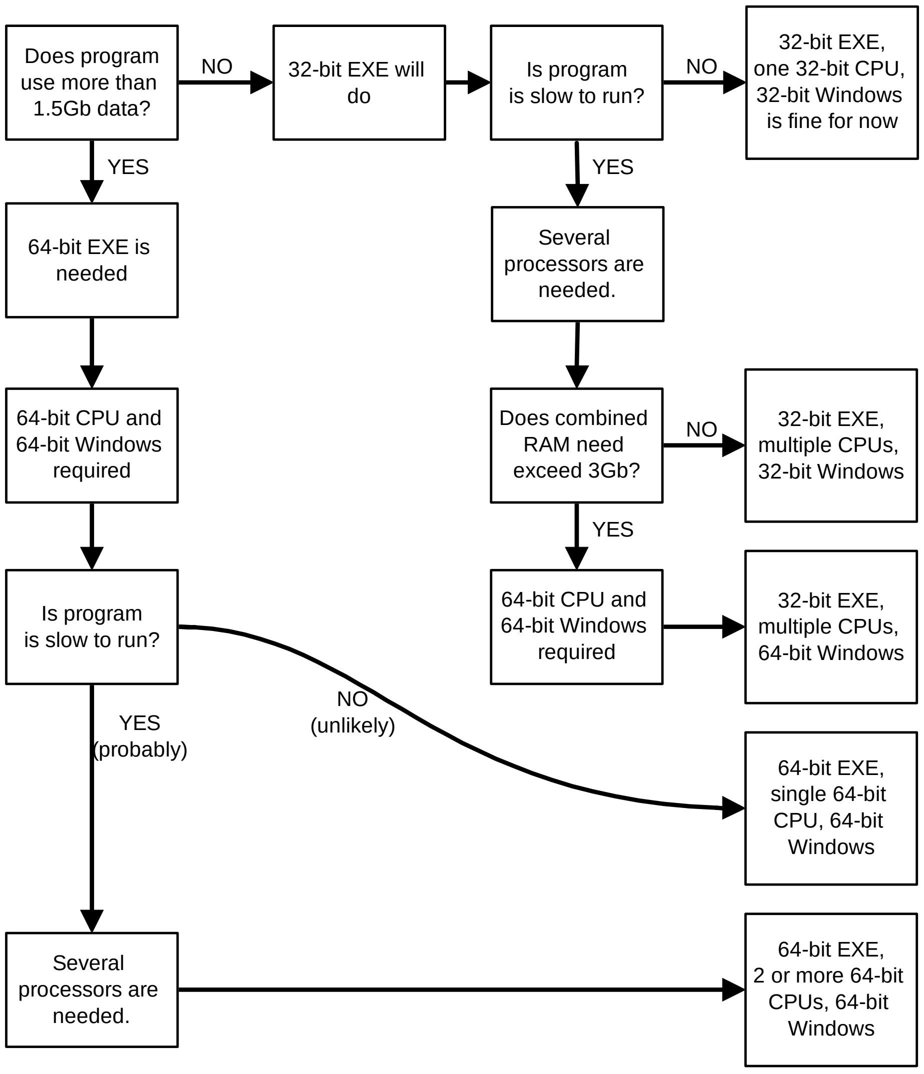
GEMPACK will support the Intel Fortran compiler from Release 10 (as well as the Lahey compilers currently supported).

The Intel compiler is of particular interest on 64-bit PCs since it comes in a 64-bit version which can write code which will only run on a 64-bit processor. This produces executable images which can access more than the 2GB limit which applies on 32-bit operating systems. Modellers who have large models which are currently close to the 2GB limit will be free of this limit if they use GEMPACK with the Intel compiler on a 64-bit PC.

The Intel compiler also comes in a 32-bit version.

4.1 Should I Consider Moving To 64-bit Intel Now?

The following flow chart will help you decide if you should consider moving to the 64-bit Intel compiler now. If 32-bit is still ok for you, you can continue to use the Lahey compilers supported by GEMPACK – there is no need to switch to the Intel compiler.


Does program use more than 1.5Gb data?
NO
32-bit EXE will do
Is program is slow to run?
NO
32-bit EXE, one 32-bit CPU, 32-bit Windows is fine for now
YES
YES
64-bit EXE is needed
Several processors are needed.
64-bit CPU and 64-bit Windows required
Does combined RAM need exceed 3Gb?
NO
32-bit EXE, multiple CPUs, 32-bit Windows
YES
Is program is slow to run?
64-bit CPU and 64-bit Windows required
32-bit EXE, multiple CPUs, 64-bit Windows
NO (unlikely)
YES (probably)
64-bit EXE, single 64-bit CPU, 64-bit Windows
Several processors are needed.
64-bit EXE, 2 or more 64-bit CPUs, 64-bit Windows

5. Beta Test Version is Available

We are making a beta test version of Release 10 of GEMPACK (this includes the master/servant machinery described in this paper) available to GEMPACK users who work in an organisation which has a multi-user Source-Code Version of GEMPACK with expiring annual licences.

If you work in such an organisation and would like to try the master/servant machinery in your work, please email Ken Pearson <Ken.Pearson@buseco.monash.edu.au>.