Two-Stage Least Squares algorithms with QR decomposition for Simultaneous Equations Models on heterogeneous multicore and multi-GPU systems

Carla Ramiro[a], José J. López-Espín[b], Domingo Giménez[c] and Antonio M. Vidal[a]

a Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia
b Centro de Investigación Operativa, Universidad Miguel Hernández
c Departamento de Informática y Sistemas, Universidad de Murcia

ABSTRACT

This paper analyzes the use of a multicore+multiGPU system for solving Simultaneous Equations Models by the Two-Stage Least Squares method with QR decomposition. When working on a heterogeneous system it is necessary to design dynamic and hybrid algorithms to exploit the full potential of the machine but the heterogeneity makes it difficult. Our contribution shows that we can efficiently exploit the resources of the machine even for dense linear algebra problems of double data type where GPUs do not offer good performance, as occurs in some highly optimized libraries that use the hybrid programming CPU with GPU, such as CULA or MAGMA, where the speedup achieved is far from the theoretical.

INTRODUCTION

Simultaneous Equations Models (SEM) are a statistical technique which has traditionally been used in economics although nowadays it is widely used in an increasing number of fields and often in large scale problems. SEM can be solved through a variety of methods, Two-Stage Least Squares (2SLS) is one of the most used methods because it can be used in all identified equations and is computationally less expensive than other methods.

Consider **N interdependent endogenous** variables which depend on **K independent exogenous** variables and **white noise** that represents stochastic interference. The relation between these variables can be expressed as follows:

$$\mathbf{Y} = \mathbf{Y}\mathbf{B}^{\mathbf{T}} + \mathbf{X}\boldsymbol{\Gamma}^{\mathbf{T}} + \mathbf{u}$$

where $\mathbf{Y} \in \mathbf{R}^{dxN}$; $\mathbf{X} \in \mathbf{R}^{dxK}$ and $\mathbf{u} \in \mathbf{R}^{dxN}$ are matrices, d is the sample size, and elements $\mathbf{Bii} = 0$

Solving a SEM is equivalent to obtaining $\mathbf{B}$ and $\boldsymbol{\Gamma}$, from a representative sample of the model (a set of values of the data variables **X** and **Y**) in order to explicitly ascertain a matrix equation which represents the relationship between both sets of variables.

2SLS Method

1. OBTAIN $\mathbf{X} = \mathbf{QR}$ (QR DECOMPOSITION OF X) AND $\tilde{\mathbf{Y}} = \mathbf{Q}^{\mathbf{T}}\mathbf{Y}$
2. SELECT COLUMNS $[\mathbf{R}_{i,1}|\tilde{\mathbf{Y}}_{i,1}]$ FROM $[\mathbf{R}_1|\tilde{\mathbf{Y}}_1]$
3. OBTAIN $[\mathbf{R}_{i,1}|\tilde{\mathbf{Y}}_{i,1}] = \tilde{\mathbf{Q}}_i\tilde{\mathbf{R}}_{i,1}$ AND $\tilde{\mathbf{y}}_{i,1} = \tilde{\mathbf{Q}}_i^{\mathbf{T}}\tilde{\mathbf{y}}_{i,1}$
4. SOLVE $\tilde{\mathbf{R}}_{i,1}\hat{\eta}_i = \tilde{\mathbf{y}}_{i,1}$

FOR i = 1 … N

PARALLEL ALGORITHMS FOR 2SLS

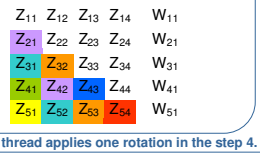
Parallelism by equations

To solve the SEM problem we need to solve N independent equations, so we can parallelize their computation. By using OpenMP API, we can create several threads. Some of these threads (depending on the number of GPUs the platform has) can work in the GPUs and are responsible for transferring data to the GPU global memory and processing the data back, while the other threads work on the CPU cores in parallel.

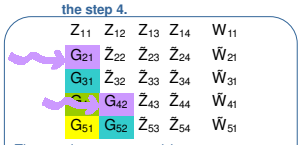
Parallel QR decomposition on GPU (QR-SPAN)

We have implemented a parallel algorithm in CUDA, based on the triangularization of a matrix in NS sequential steps by using a parallel scheme, which simultaneously nullifies elements placed in several main diagonals. To calculate and apply a number of Givens rotations simultaneously the rows involved must be disjoint. QR-SPAN computes the QR decomposition of a matrix **Z** and applies $\mathbf{Q}^{\mathbf{T}}$ on another matrix **W** by using this method:

Triangularization of a matrix **Z** in 7 sequential steps by using parallel scheme, which simultaneously nullifies elements placed in several main diagonals.

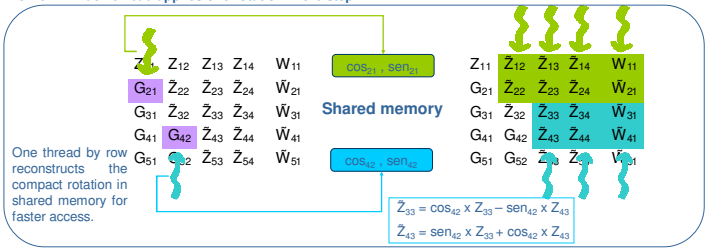


Kernel 1 : Each thread calculates a rotation in the step 4.



The rotations are stored in compact way at the bottom of the matrix.

Kernel 2 : Each thread applies one rotation in the step 4.



One thread by row reconstructs the compact rotation in shared memory for faster access.

IMPLEMENTED ALGORITHMS

Let us suppose that the platform has **p CPU cores** and **n GPU installed**. The N equations of model can be distributed among **p + n heterogeneous cores**:

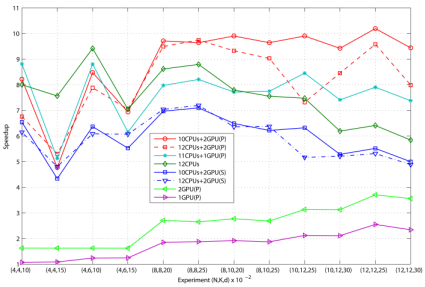
- ✓ **pCPUs**: Each core computes a QR with a sequential algorithm.
- ✓ **nGPU(S):** Each GPU thread calculates a complete decomposition.
- ✓ **nGPU(P):** All threads of GPU collaborate to compute a QR by applying the **QR-SPAN** algorithm.

The computer used in our experiments has:

- 2 Intel Xeon X5680 hexacore processors at 3.33 GHz.
- 2 Nvidia Tesla C2070 GPU with 14 SM and each one includes 32 cores.

RESULTS

- The results with 12CPUs are good, and programming is very simple, yet they are far from the theoretical speedup. But, if we apply the same parallelization scheme (12CPUs+2GPU(S)) without taking into account the specificities of the GPU, we get a negative effects on performance (excessive number of memory accesses, uncooperative threads and a higher CPU speed with regard to GPU).
- The use of the GPU as a standalone tool (nGPU(P)) provides benefits but does not even reach the performance obtained when using parallelism in the CPU. The use of two GPUs does not imply a fifty per cent reduction in the execution time, this is because the transferences to two GPUs do not overlap.
- The best results are obtained when using the system as a single heterogeneous computer (pCPUs+nGPU(P)), particularly for the case 10CPUs+2GPU(P). Note that the use of the 12CPUs in 12CPUs+2GPU(P) and 12CPUs+2GPU(S) versions does not achieve better speedups except for some small sizes. This is because two of these cores are busy sending and receiving data to and from the GPU, and therefore remain occupied for some time.



CONCLUSIONS

- Computers with multicore+multiGPU architecture are useful when working with big Simultaneous Equations Models built from a set of lower dimension models, i.e the world mode managed by the LINK project.
- When we are working on a heterogeneous system it is necessary to design dynamic and hybrid algorithms to exploit the full potential of the machine but the heterogeneity makes it difficult. The problems should be suitable and programming must be performed carefully.
- Our contribution shows that we can efficiently exploit the resources of the machine even for dense linear algebra problems of double data type where GPU do not offer good performance, as occurs in some highly optimized libraries that use the hybrid programming CPU with GPU, like CULA or MAGMA, where the speedup achieved is far from the theoretical.

REFERENCES

- W. Greene, Econometric Analysis, 3rd Edition, Prentice Hall, 1998.
- R. Henry, I. Lu, L. Beightol, D. Eckberg, Interactions between CO2 Chemoreflexes and Arterial Baroreflexes, Am. Journal of Physiology 274 (43) (1998) H2177–H2187.
- W. Ressler, M. Waters, Female earnings and the divorce rate: a simultaneous equation model, Applied Economics 32 (2000) 1889–1898.
- J. J. López-Espín, A. M. Vidal, D. Giménez, Two-stage least squares and indirect least squares algorithms for simultaneous equations models, Journal of Computational and Applied Mathematics (0) (2011) .
- Project LINK Research Centre, www.chass.utoronto.ca/link.

ACKNOWLEDGEMENTS

Spanish Ministerio de Ciencia e Innovación (Projects TIN2008-06570-C04-02, TEC2009-13741 and CAPAP-H3 TIN2010-12011-E), Universidad Politécnica de Valencia (PAID-05-10), Fundación Séneca from C.A.Regi´on de Murcia (08763/PI/08) and Generalitat Valenciana (project PROMETEO/2009/013).