Parallelization of the Nanoscale Device Simulator nanoMOS2.0 Using a 100 Nodes Linux Cluster

Sébastien Goasguen, Ali. R. Butt, Kevin D. Colby and Mark S. Lundstrom
Purdue University, 1285 Electrical Engineering Building, West Lafayette, IN 47907-1285
sebgoa@purdue.edu

Abstract- We use a state of the art linux cluster for quantum simulations of nanoscale devices. The simulator nanoMOS2.0 can be accessed through the Purdue University Network Computing Hub (PUNCH) that interoperates with the cluster through the Portable Batch System. NanoMOS2.0 is also modified to speed up the energy integration by distributing the energy grid over several processors. A 88% speed-up is achieved using the Parallel Matlab Interface .

I-Introduction

The program, nanoMOS2.0, simulates quantum transport in nanoscale MOSFETs using ballistic or dissipative transport models based on the non-equilibrium Green's function (NEGF) formalism [1]. NanoMOS has been written in Matlab, and it has been a very easy way to investigate the physics of nanoscale double gate MOSFETs [2]. Parallelizing a Matlab code does not produce the most efficient simulations, but it is the best way for us to focus on the physical phenomena at this stage of its development.

II-Linux cluster description and performance

Our Linux cluster is composed of 100 nodes dual-Athlon 1.2 GHz with 1 GB of RAM and 10 GB of disk. The head node is a quad Xeon 700 MHz with 2 GB of RAM and 160 GB of RAID storage. The operating system is Red-Hat 7.2 with the 2.4.17 kernel. PBS is installed to schedule batch jobs and monitor the usage of the cluster. Parallelization is done through the Message Passing Interface (MPI) routines. The Automatically Tuned Linear Algebra Sub-programs (ATLAS) has been used to create the BLAS and LAPACK library optimized for the slave nodes. The High Performance Linpack (HPL) is used to benchmark the performance of the cluster. After careful tuning of the HPL benchmark we obtain around 130 GFLOPS for the whole cluster. These performances unofficially put the cluster within the TOP500 supercomputers. Fig. 1 shows the cluster in its final configuration.

III-Parallelization strategies

This cluster can be used in different ways to run our quantum simulations of double gate MOSFETs (Fig. 2) using nanoMOS2.0. One way is to run multiple simulations of nanoMOS using different input decks. Each simulation is sent to the cluster through the PBS that assigns this job to a free processor. One can also decide to truly parallelize the code. In this way, one simulation of nanoMOS will use several processors at the same time. Both approaches can also be combined, shipping nanoMOS runs to the cluster each run using several processors. To implement the first strategy we decided to make use of PUNCH, a web portal that allows users to run all types of tools through the internet [3]. A tool implemented in PUNCH can be setup to interoperate with PBS [4]. We setup a new tool on the NanoHUB: nanoMOSCluster that ships his jobs to the cluster through the PUNCH-PBS interface. Figure 3 shows a snapshot of the nanoMOSCluster tool as displayed in PUNCH.

Parallelization of nanoMOS can be very advantageous. In the NEGF formalism the electron density is obtained by integration over energy. This integration is the most CPU time expensive operation in the code as shown in Fig. 5. When multiple valleys and multiple subbands are added this integration becomes even more CPU time intensive (Fig. 6). Several toolboxes are available to do parallelization in Matlab. A MPI toolbox and a Parallel Virtual Machine (PVM) toolbox are available. For our first attempt at parallelizing the energy integration in NEGF formalism we used the Parallel Matlab Interface (PMI) that uses the Matlab engines API to manage remote Matlab sessions. Figure 6 shows the speed up obtained by parallelizing the energy integration over processors. A 88% speed-up can be achieved for a fine energy grid, resulting in an overall speed up of 52 %.

IV-Conclusions

We have successfully setup a powerful Linux cluster to provide enormous computing power through the web-computing portal PUNCH. An initial parallelization of nanoMOS2.0 distributing the energy grid in the NEGF formalism over the processors has led to a 88% speed-up of the energy integration and an overall 52 % speed up of nanoMOS2.0.

References

[1]-S. Datta " Nanoscale device modeling: the Green's function method" *Superlattices and Microstructures*, 28, pp. 253-278, 2000.

[2]-Z. Ren et al. "Examination of Design and Manufacturing Issues in a 10 nm Double Gate MOSFET using Non equilibrium Green's Function Simulation" *IEDM, Tech. Dig.*, pp. 2001.

[3]-N.H Kapadia, R. J. Figueiredo and J.A.B. Fortes "PUNCH a Web Portal for Running Tools" *IEEE Micro. In special issue on Computer Architecture Education*, May-June 2000.

[4]-S.Adabala, N. H Kapadia and J.A.B. Fortes "Performance and interoperability issues in incorporating cluster management systems within a wide area network computing environment" *Supercomputing 2000: High Performance Networking and Computing*. Dallas, Texas.

This work was supported by the ARO DURINT program

Fig. 1 Purdue Computational Electronics Laboratory Linux cluster "Superman"

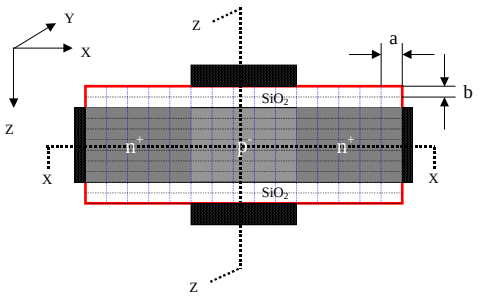

Fig. 2 Structure of the Double-Gate MOSFET simulated in nanoMOS.

PUNCH NanoMOS on Superman Cluster

Hub Directory

User: sebgoa | Run NanoMOS on Superman Cluster. Help Available.

NanoMOS on Superman Cluster-Related Information

Description
First Time User's Guide
Manual
Examples
Source
Questions

Run Status

Run NanoMOS on Superman Cluster

1. Modify/Create NanoMOS on Superman Cluster Input Files
2. Execute NanoMOS on Superman Cluster
3. View/Download NanoMOS on Superman Cluster Output Files

Nanotechnology Simulation Hub
PUNCH User's Manual | PUNCH-Related Questions/Comments

Fig. 4 Sample screen shot of NanoMOSCluster webpage on the nanoHUB

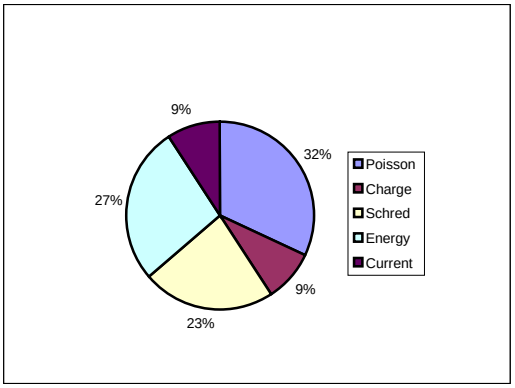

Fig. 4 Repartition of the CPU time in nanoMOS simulations for a quantum ballistic simulation with only one subband and unprimed valley.

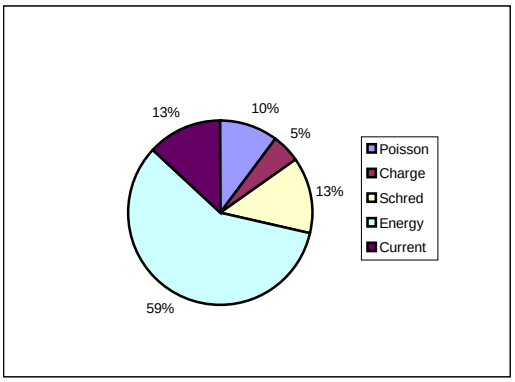

Fig. 5 Repartition of the CPU time in nanoMOS simulations for a quantum ballistic simulation with three subbands and all valleys.

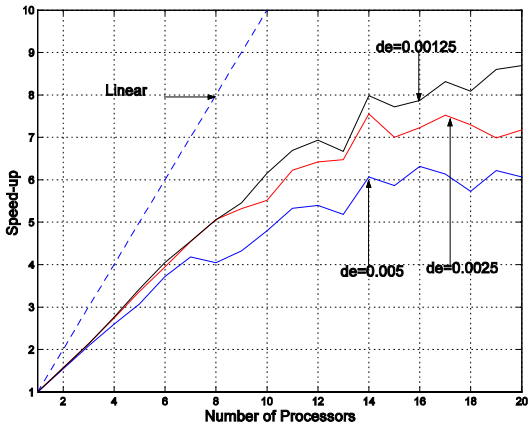

Fig. 6 CPU time speed-up by distributing the energy grid among processors. Three grid spacing have been used to investigate the parallelization algorithm.