GPU linear solver for porous media flow simulation.

List of authors:

T. Guignon [1] M. Hacene [2] J.M. Gratien [3]

Porous media flow simulation in petroleum industry is used in reservoir simulation or basin modeling to plan fields development or characterize potential new fields. It's also used in CO2 geological sequestration to prove feasibility and stock safety for long time period. One of the main computation done in theses simulations is solving one or more large sparse unstructured linear systems at each time step. These systems come from finite volume discretization of Darcy flow equation. In reservoir simulation time spent in linear solver can be up to 80% of the total simulation time so solver efficiency is a key point in simulation performance.

The classical method for solving theses systems is an iterative Krylov method like BiCGstab. Since most systems are not well conditioned we use some preconditioner like ILU0, ILU(k) or CPR-AMG for difficult problems which are numerically efficient but not really scalable on distributed memory parallel computers.

Using hardware accelerators like GPU is another way to improve linear solver performance because theses accelerators provide a very high peak floating point and memory performances compared to CPU. But theses improvements comes with some constraints on parallel algorithms we want to deploy on theses accelerators. Mainly datas and computations must have a very regular structure and algorithms must exhibit a very fine grain parallelism to achieve high performance computation. These constraints are very similar to those we have with data parallel programing models and data parallel architectures that where used in the 90's.

Iterative Krylov method like BiCGStab rely mainly on two main software components: a sparse matrix vector product and preconditioner application ,for example solving $L.U.x = y$ for ILU. So the focus is on deploying these components to GPU. Other solver parts, simple vector algebra, can be easily done with a dense BLAS GPU library like CUBLAS.

We will first discuss sparse matrix vector product for unstructured sparse matrix which is now well understood. We present on figure some spmv[4] results with different GPU systems compared to CPU. Linear systems come from reservoir simulation study cases and for most of them there at least is a 10x increase in performance over the fastest CPU tested.

In second we will discuss preconditioning. We will focus on preconditioner application (building is still done on CPU). We will first study $L.U.x = y$ on GPU where L and U are linear system incomplete factors. Unstructured sparse triangular solve required in $L.U.x = y$ does not exhibit much parallelism with the linear system "natural ordering". Thus for an efficient triangular solve on GPU we must reorder system to make it with large diagonal blocks and extra diagonal

[1]IFP Energies Nouvelles
[2]IFP Energies Nouvelles
[3]IFP Energies Nouvelles
[4]sparse matrix vector product

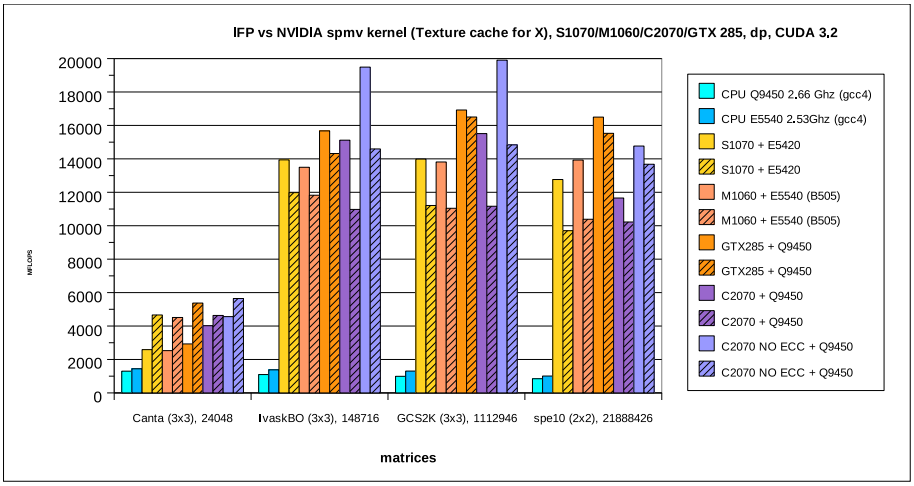

Figure 1: Comparing CPU spmv, IFP spmv and NVIDIA spmv on various GPU configurations.

sparse blocks A_u and A_l:

$$P.A.P^{-1} = \begin{pmatrix} D_1 & & A_u & \\ & D_2 & & \\ & & \ddots & \\ & A_l & & D_n \end{pmatrix}$$

This reordering is obtained by coloring vertices of the adjacency graph such as a vertex (an equation) has no neighbour of his own color and group vertices of the same color. After ILU solving $U.x = y$ (or $L.x = y$, with unit diagonal) expose a high level of fine grain parallelism since we have to chain diagonal inversion and spmv:

$$\begin{pmatrix} D_1 & \dots & U_1 & \dots \\ & D_2 & U_2 & \dots \\ & & \ddots & \\ & & & D_n \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

But reordering linear system reduce significantly ILU numerical efficiency: BiCGstab convergence rates are much slower thus reducing ILU interest.

Since GPU ILU is not efficient as ILU CPU another way is to look at preconditioners that naturally exhibit fine grain parallelism even with lower numerical efficiency. We test polynomial preconditioners and SSOR on CO2 geological sequestration full simulation. Results in table show that highly parallel GPU preconditioner are efficient compared to most advanced CPU preconditioners

190x120x5	BSSOR (GPU)	Polynomial (GPU)	CPR-AMG (CPU)	IFP ILU0 (CPU)	PETSC ILU0 (CPU)
Total simulation time	2100	3914	2964	7032	15563
Solver time	1410	3214	2301	6381	15528
Solver iterations	152483	123037	5069	86353	100159
Time steps	801	801	801	801	784

Table 1: Comparing various CPU and GPU preconditioner on full CO2 sequestration simulation, 114000 grid cells

References

[1] Iterative Methods for Sparse Linear Systems, second edition.
Youssef Saad, *Siam*

[2] Matrix computation, third edition.
Gene H. Golub, Charles F. Van Loan, *John Hopkins University Press*

[3] Implementing Sparse Matrix-Vector Multiplication on Throughput-Oriented Processors.
Nathan Bell and Michael Garland, *Supercomputing '09 proceedings, November 2009*

[4] Scalability and Load Balancing Problems in Parallel Reservoir Simulation.
Gratien J.M. , Guignon T., Magras J.F., Quandalle P., Ricois O., *SPE 106023, 2007 SPE, Reservoir Simulation.Symposium, 2007*