SPE-173223-MS

Will GPGPUs be Finally a Credible Solution for Industrial Reservoir Simulators?

A. Anciaux-Sedrakian, IFP Energies nouvelles; J. Eaton, NVIDIA; J. Gratien, T. Guignon, P. Havé, C. Preux, and O. Ricois, IFP Energies nouvelles

This paper was prepared for presentation at the SPE Reservoir Simulation Symposium held in Houston, Texas, USA, 23–25 February 2015.

Abstract

Heterogeneous supercomputers combining “general-purpose graphic processor”, “many integrated cores” and general-purpose CPUs promise to be the future major architecture, because they deliver excellent performance with limited power consumption and space occupancy. However, developing applications that achieve “real” scalability remains a subject of several research fields. This paper studies the possibility of exploiting the power of heterogeneous architecture for Geoscience dynamic simulations for real large-scale models. Geoscience dynamic simulations need scalable linear solvers to realize the potential of the latest high-performance-computing architectures, since it represents the most expensive part of the whole simulation. Several studies have already been done to accelerate sparse linear solver’s preconditioners and they illustrated comparable or even improved performance for the academic test cases. However, for a real industrial test case with highly heterogeneous data, these preconditioners do not offer a robust solution, as CPR-AMG does. We study several implementations of AMG and CPR-AMG for heterogeneous architectures in this paper. We illustrate the obtained gain using one real field thermal model and two academic test cases all using a fully implicit scheme. For these models, the total simulation time is reduced by a significant factor when running on n CPU + n GPU, compared to n CPU only, without any accuracy loss.

Introduction

Basin modeling or reservoir simulations with highly heterogeneous petrophysical data and complex geometries are based on the resolution of complex non-linear Partial Differential Equations -PDEs-. These PDEs are discretized with a finite volume scheme in space, leading to non-linear systems, which are generally solved with an iterative Newton solver. At each Newton step, a linear system is built and solved with an iterative algorithm, such as Bi-Conjugated Gradient Stabilized, well suited for large, sparse and unstructured systems. Since this linear solution phase constitutes the most expensive part of the simulation, representing nearly 60% to 80% of the total simulation time, the efficiency of linear solvers is therefore a key point of the simulator’s performance. The convergence rate of iterative linear solvers depends on the spectral properties of the matrix. In Geoscience applications, due to the heterogeneities of the geological data, the linear system may be ill conditioned which has an important impact on the

robustness of the used iterative methods. They may indeed fail to converge. This spectrum is particularly difficult to predict -either not attainable or need so many iterations to converge- due to the coefficients. Hence naturally, the iterative solver cost depends directly on the number of iterations required for convergence and each iteration cost. Appling preconditioning techniques is a well-known way to improve this rate. Thus in order to reduce the cumulative number of iterations for the whole simulation, the choice of a robust parallel preconditioner becomes important mainly for heterogeneous elliptic problems which is the case of most real simulations.

Several studies have been done already to accelerate sparse linear preconditioners such as BILU(0), BILU(T), SSOR, AMG [2,7,9,13] and they illustrated comparable or even improved performance for the academic test cases. However, for real industrial test cases, which often have highly heterogeneous data, these preconditioners do not offer a robust solution, as CPR-AMG does. This paper studies and reviews different implementations of preconditioners as Polynomial, ILU0 and CPR-AMG used in different reservoir simulations for both homogenous and heterogeneous architectures. It shows the advantages and the inconvenients of each algorithm for the different contexts of Geoscience dynamic simulations, in particular in reservoir simulation.

The first section reviews hardware and software considerations in a context of heterogeneous super-computers. The second section presents the preconditioned BiCGStab with general-purpose graphic processors and describes the common kernels among all the preconditioned iterative Krylov subspaces methods. In the third section, we present different preconditioners as polynomial, BSSOR, ILU(0) and AMG preconditioner algorithms -used in Krylov solvers- using classical and heterogeneous architecture. In section four, we present the CPR-AMG algorithm and NVIDIA's AmgX algorithms. Results are gathered and analyzed in the fifth section, while section six concludes this work.

Hardware and software context

In the exascale computing roadmap, using accelerators -GPUs and many-core- is a key point. These kinds of architectures offer a very high peak of floating-point operations per second, a very high peak memory bandwidth, for very interesting ratios of FLOPS per watt and FLOPS per euro.

In the past twenty years, GPUs have evolved from fixed-function processors to massively parallel floating-point engines. Hence, the idea of General-Purpose Computation on Graphic Processing Units (GPGPU) emerged to take advantage of the processing power of GPUs for non-graphical tasks. General purpose programming tools like BrookGPU[4], CUDA[16] or OpenCL [18], ease GPU use for all programmers.

The GPU is a massively parallel architecture that allows the user to run several 10s of thousands of threads concurrently. In order to use the GPU to its full potential we need to run a sufficient amount of work on it such that we are fully utilizing its resources. For example, if we are running a large ZGEMM -say 4096x4096x4096- then we have enough work to saturate any of today's GPUs and achieve their peak FLOPS -which in the case of the NVIDIA K40 GPU for example is 1.43TFLOPS per GPU-. On the other hand, if we run a much smaller sized ZGEMM, then we do not run enough work to utilize all of the GPU's resources and we become latency bound. In order to run such small ZGEMMs, or in fact any GPU kernel, at higher efficiency we can try to run several of them at the same time on the GPU. Therefore, in order to fully benefit from the GPU, computational power still requires that algorithms exhibit a high degree of parallelism and regular data structures. There are two key issues in developing high performance implementations for heterogeneous architectures:

- Maximizing the occupancy of the GPU;
- Minimizing the high transferring data cost between the GPU and CPU node(s).

The first issue boils down to expressing enough parallelism to feed the tens of thousands of GPU threads. The second issue is addressed by keeping data resident on the GPU for as long as possible. This

can mean porting many intermediate, non-performance critical methods to the GPU, and carefully choosing or designing low-communication algorithms. “Batching” and “streaming” [17] could help to generate large GPU-ready tasks.

Preconditioned BiCGStab and GPU acceleration

In multiphase porous media flow models, the discretization and the linearization of the PDEs lead to large linear systems to compute the pressure or/and the temperature which are elliptic/parabolic unknowns and the saturations or/and the compositions which are hyperbolic unknowns. These linear systems, often ill conditioned due to the heterogeneous and anisotropic geological data, are classically solved with preconditioned iterative Krylov subspaces methods such as GMRES, CGS or BiCGStab.

Most of preconditioned iterative Krylov subspaces methods, are based on the following kernels : i) basic vector algebra operations, AXPY, SCAL, DOT and copy, ii) sparse matrix vector product -SpMV- and iii) preconditioners, polynomial, BSSOR, ILU(0), etc. . .

Basic vector algebra operations

These operations are usually vendor-supplied through optimized BLAS packages. For example, NVIDIA provides the CUBLAS library, which contains the full BLAS1, 2 and 3 implementations plus some extensions. Nevertheless, BLAS operations do not map directly to iterative Krylov methods’ needs. Thus, sequences of BLAS1 operations are used even for simple linear combinations with more than two vectors. These sequenced BLAS1 calls do not have any impact on linear solver performance when using large linear systems. However, if we consider the smaller cases or a size per process in hybrid MPI+GPU cases, these sequenced calls tend to be inefficient due to the GPU kernel call overhead. This situation leads us to develop new GPUs kernels that exactly fit BiCGStab operations.

Sparse matrix Vector product

Sparse matrix vector product -SpMV- on a GPU has been widely studied since introduction of General-Purpose GPU; GPGPU computation [3]. Although, SpMV exhibits simple and massive parallelism in $y=A.x$ each y_i computation is independent from each other whereas each y_i is a dot product that also exhibits parallelism.

The before-mentioned investigations always show the importance of choosing the right sparse format for the right sparse structure due to the need of massively fine-grain and regular parallelism for efficient GPU use. Thus, unstructured sparse formats like CSR or COO do not suit well for GPU computation.

NVIDIA now provides in the CUDA toolkit cuSparse library a SpMV for different sparse formats and this should be considered as a reference.

Although NVIDIA provides efficient SpMV for Ellpack and Hybrid format -mix of Ellpack and COO-, these SpMV are still designed for generic purpose and do not use linear specific systems that come from the reservoir simulator. To use these specifics, we propose a SpMV that explores the sparse block structure of the matrix A where each non-null element is a small 4x4 3x3 or 2x2 blocks, depending on the underlying Black-Oil simulation.

By using blocks, we reduce the indirection cost in SpMV: within the sparse matrix A, a column index represents 2, 3 or 4 contiguous elements of x so that the effort for loading column indices is reduced by a factor up to 4 to 16 for each block copaired to a point sparse approach. Furthermore, the data reuse is better in block matrices and the access patterns are more regular. Performance improvements provided by this approach are shown in [1].

Preconditioners

The efficiency of preconditioners depends numerically on the problem size, on the heterogeneity, the discontinuities and the anisotropies of the problem data. However, the efficiency of their implementation depends on the target hardware architecture. Some algorithms may be numerically efficient and help to

reduce the number of iterations considerably, but they may be difficult to parallelize and not efficient in terms of FLOPS on certain hardware configurations. Even more, to have performance on hardware configurations like GPGPUs, it is preferable to have algorithms offering a high degree of parallelism at a fine-grain level. Therefore, the compromise between the numerical efficiency of a preconditioner and the performance of its algorithm's implementation for each hardware configuration must be considered.

Overview of available preconditioners on GPGPU

The family of proconditioners based on Polynomial and SSOR algorithms is usually poor for standard industrial study cases. The preconditioners based on incomplete factorization ILU(k,t) with k as the level of factorization and t as the threshold parameter are interesting for easy test cases with low heterogeneity and common grid block size. Finally, for highly heterogeneous data and huge grid block size the best preconditioners are based on the CPR-AMG algorithms. Nevertheless the incomplete factorization or AMG algorithms is unfortunately not naturally parallel at a fine grain size, and they need sophisticated parallelism techniques to be implemented efficiently on GPGPUs.

The **Neumann Polynomial preconditioner** consists simply in applying several time the SpMV to compute an approximation of the inverse of a matrix A as a polynomial Pm(A) of degree m. The efficiency of the preconditioner relies therefore directly on the efficiency of the SpMV GPGPU implementation.

The **Block Symmetric Successive Over-Relaxation preconditioner** consists in applying one of several iterations of the Symmetric Successive Over Relaxation method well described in [19]. To parallelize the backward and forward substitution algorithms which are sequential, the block version is usually used, version which consists in introducing a standard block-coloring renumbering procedure enabling to parallelize at fine grain size level the back and forward substitutions on different sets of independent rows and columns.

The **ILU(k,t) preconditioners** commonly used in basin and reservoir simulation have been implemented on GPGPUs more recently [1]. As the LU factorization is naturally sequential and difficult to parallelize at a fine grain size, a graph coloring technique is needed to exhibit such parallelism. However, such renumbering algorithms have a well-known impact on the shape of the matrix and on the fill-in phenomena of the factorization. As the incomplete factorization algorithm is associated with various criteria to drop fill-in matrix entries, these renumbering techniques have an impact on the amount of dropped entries and therefore on the numeric robustness of the preconditioner: the greater the amount of dropped entries the poorer becomes the preconditioner. This problem becomes a great handicap for the GPGPU version of such algorithms regarding their standard CPU version as the first one needs often much more iterations than the second one on CPU.

The **Algebraic MultiGrid -AMG- preconditioner** [21,15] is a robust preconditioner for elliptic problems. It is appreciated for its extensibility qualities with M-matrix systems: the number of iterations required to converge only depends minimally on the size of the problem and can be entirely size-independent. This is an important feature for massive parallel applications. The main idea of Algebraic MultiGrid is to remove smooth error components on the coarser grid. This is done by solving the residual equation on a coarse grid and then interpolating the error correction onto the finest grid. The coarsening scheme is the major and crucial part of the AMG preconditioner for both the sequential and the parallel version. AMG is based on different coarsening heuristics.

The traditional coarsening scheme proposed by Ruge and Stuben -RS-, tends to work well for two spatial dimensions -2D- problems. Several works, concerning coarsening algorithms, have been done in order to adapt them to 3D problems. The following coarsening algorithms -PMIS and HMIS- well-known for three-dimensional -3D- problems, are implemented for the parallel environment. Parallel Modified Independent Set -PMIS- algorithm [6], is a modification of an existing parallel algorithm; CLJP. In this algorithm, the coarse-grid result is independent of the number of processors and the distribution of the

points across processors. The Hybrid Modified Independent Set -HMIS- algorithm [6], is obtained by combining the PMIS algorithm with a one-pass RS scheme.

The unsmoothed aggregation AMG method takes a very different approach. Instead of identifying 'coarse' and 'fine' points from the fine grid, aggregation attempts to pair up strongly connected points to form new degrees of freedom, or 'aggregates'. One pass produces aggregates of size 2, and additional passes produce aggregates on average of size 2n, for n passes. In the 'unsmoothed' version, these aggregates are only represented logically: each fine point is assigned to an aggregate using an integer index. The P matrix has a special form, where each column on P corresponds to an aggregate in the coarse problem. The P matrix consists in a 1 in column 'I' and row 'J' if fine point 'J is assigned to aggregate 'I', and 0 everywhere else. R is taken as the transpose of P. Forming the RAP product then has a special optimization, because only the sum of the entries of A is needed without any multiplication. This aggregation approach is very efficient to parallelize using a 'one-phase handshake' [5] and for constant-coefficient problems aggregation AMG delivers the best overall time to solution.

The difficulty of implementing AMG algorithms on GPGPU architectures lies on two main facts: firstly, most of the algorithms are naturally sequential and usually parallelized only at a coarse grain level; secondly, it is not obvious to design data structures shared between all the kernels used to implement the AMG ingredients that fit in memory for each of these kernels. The implementation of AMG on GPGPU requires to adapt each ingredient standard algorithms or to find new ones more appropriate to fine grain parallelism. This has been realized only in few projects like GAMPack from Boston University [9] and AmgX by NVIDIA [8].

CPR-AMG Preconditioners on GPGPU

CPR-AMG -Constrained Pressure Residual Algebraic MultiGrid- is an effective preconditioner of iterative Krylov solvers, for reservoir simulators. This two-stage preconditioner, introduced by Lacroix, Vassilevski and Wheeler [14] is based on the CPR[1] acceleration with an efficient solver such as AMG solvers on the elliptic sub system, in our case the pressure part, then in the resolution of the full system with a simpler but efficient preconditioning step. CPR-AMG often combines one V cycle of AMG on a pressure equation with an ILU(0), Polynomial, etc. preconditioner on the full system. This two-stage preconditioner may be defined as follows:

Let $A\ x = b$ be the linear system to solve, where A represents the full system with A_{pp} pressure block coefficients, A_{ss}; non-pressure block coefficients -saturation block in Black-oil context or saturation and concentration block in compositional context-, and A_{ps}, A_{sp}; the coupling coefficients:

First stage:

- Apply ILU(0) (or polynomial) on the full system; $ILU0(A)\ x^1 = b$
- Compute the new residual; $r_p = b_p - A_{pp}x_p^1 - A_{ps}x_s^1$

Second stage:

- Apply AMG-Vcycle on the residual pressure equation; $AMG(A_{pp})x_p^2 = r_p$
- Correction of the pressure unknowns; $x_p = x_p^1 + x_p^2$

The CPR acceleration with an AMG solver enables to solve efficiently large-scale and complex reservoir problems, in fully implicit formulation.

[1] CPR intends to extract the pressure part -elliptic unknowns- and approximately solve it. The pressure solution is used to constrain the residual on the full system, thus achieving a more robust and flexible overall solution strategy

The first stage preconditioner choice

The choice of an efficient relaxation solver, which depends tightly on the target architecture, is critical for this stage, since its cost must be moderate. The polynomial preconditioner exhibits a high level of parallelism. Nevertheless, it is not numerically robust because it does not reduce significantly the condition number of the matrix. On multi-core architectures, this preconditioner is usually not competitive with other preconditioners. However, as its construction is not expensive and it can be parallelized easily at a fine-grain level, its algorithm is well suited for the GPGPU's data parallel programming model. As a result, even if it requires many more iterations to converge than most of other preconditioners, it offers a good performance on the GPUs.

On the other side, ILU(0) is numerically robust but not naturally parallel, because of its recursive algorithm. ILU(0) is based on a triangular solve with the incomplete factors L and U. The parallelism of the sparse triangular depends strongly on matrix ordering and if we consider "natural" ordering coming from the Geoscience simulator, the degree of parallelism in the triangular solve is at most the size of the largest sparse row. The graph coloring technique, generally used to improve the level of parallelism at a fine-grain level, is efficient in iterative solution methods for the SPMD context. However, in the ILU(0) context, this renumbering technique, modifying the shape of the original matrix graph, has an influence on the rate of dropped fill-in entries during the incomplete factorization. This well-known renumbering effect may decrease the numerical efficiency of the preconditioner, as it may increase the number of iterations to achieve convergence compared to the factorization with the natural ordering [1].

By taking into account these considerations, in heterogeneous architecture context the first stage of CPR-AMG could benefit from the polynomial scalable parallel -MPI+CUDA- implementation. Besides, in homogenous multi-core architecture parallel ILU(0) -MPI+Multi-thread, or MPI- implementations could be good candidates.

AmgX: Used AMG in second stage of CPR-AMG

For the CPR stage on GPGPU, we use the NVIDIA AmgX package. Matrices are implemented with a CSR and BSR formats that enable both coarse-grained parallelism (MPI) and fine-grained parallelism for the GPU. The use of a high-quality graph partitioner, such as METIS, SCOTCH or ParMETIS, is a critical point as it provides both load balancing of the partitions and some attempt to minimize communication costs. The BSR format with block-structured PDEs allows fewer indirections per flop, improving overall performance.

Coarsening is accomplished using the PMIS or aggressive PMIS algorithm for classical AMG, and by a one-phase handshaking algorithm for aggregation AMG. Both AMG versions have a fully parallel setup phase in AmgX. A key component in the setup phase is a memory-pool, which enables fast memory allocation and re-use of de-allocated blocks by condensing small blocks into larger contiguous chunks automatically.

To discover parallelism on the GPU, AmgX provides fast, parallel multi-coloring algorithms on the GPU, which can produce high-quality graph colorings in minimal time. The coloring can then be used to induce a permutation of the unknowns, which exposes large amounts of parallel work for normally serial smoothers like Gauss-Siedel or ILU. Often block-Jacobi is effective for block systems in BSR format, and because of the inherent parallelism in the Jacobi method, it is a good first choice. AmgX also provides a group of polynomial smoothers, which leverage a fast SPMV operation to achieve high performance and parallelism.

For coarse solves, AmgX can use a small dense LU factorization on a single GPU, or apply any of the available Krylov methods or smoothers as a solver on the coarse level in multi-GPU cases. When taking advantage of coarse-grid consolidation, the dense LU solver can even be used in multi-GPU cases. Often the LU factorization provides a robust solution, but a simple smoother or Krylov method may deliver higher performance depending on the system to be solved.

MCGSolver

MCGSolver -MultiCore multiGpu Solver- is a software package developed by IFP Energies nouvelles -IFPEN- to provide linear solver algorithms for its industrial simulators in reservoir simulation, basin simulation, or engine combustion simulation. The provided algorithms are designed to be efficient on customer hardware configurations such as Linux cluster with multi-core nodes partially accelerated with GPGPU or Windows 64 platform. This package contains a Krylov BiCGStab solver, some specific domain partitioner and matrix graph renumbering algorithms. It is interfaced with NVIDIAs libraries like CUBLAS, CUSPARSE and AmgX. All the preconditioners algorithms mentioned previously like ILU(0), BSSOR, Neumann polynomial and CPR-AMG are developed in this package.

Numerical Experimentation

In this section, we evaluate the performance of the different linear solver features presented in the previous section on various hardware configurations. We focus on the total solving time -setup + solve time- for several complete reservoir simulations to identify the performance improvement, using GPU acceleration, that can be expected compared to the legacy CPU parallel solver.

For our experimentations, we use our in-house research reservoir simulator to illustrate the MCGSolver impact. This simulator is developed with ArcGeoSim^{TM}, which is a C++ framework dedicated to provide the foundations of the new generation of Geoscience simulators at IFPEN. It is built over Arcane [11], an HPC framework co-developed by IFPEN and CEA since 2007. The combined Arcane/ArcGeoSim^{TM} framework aims to focus application developers to their core business and to delegate to the framework many low level features : I/O, service plugins, parallelism management including load-balancing, distributed mesh and variables data structures, standard finite volume numerical schemes, linear solvers. . .

The linear solver features presented in this work are provided by our in-house IFPSolver and MCGSolver libraries.

These libraries are available via ALIEN library which is a part of the ArcGeoSim^{TM}. This library provides an abstract layer to handle algebraic objects independently of their implementation. This abstract layer is implemented with different libraries like PETSc, Hypre, PMTL4 and our in-house libraries. . . available via a plugged-in mechanism.

In our experimentations, we use the following preconditioners:

- ILU(0) preconditioner -ILU0 CPU- for multi-core architecture provided by the IFPSolver library;
- CPR-AMG preconditioner -CPR-AMG CPU- for multi-core architecture provided by the IFP-Solver library using a Hypre AMG solver for the CPR stage and a parallel ILU(0) for the relaxation solver;
- CPR-AMG preconditioner -CPR-AMG GPU- for GPGPU architecture provided by the MCG-Solver library based on the NVIDIA AmgX solver for the CPR stage and a GPU Neumann Polynomial preconditioner for the relaxation solver;
- Neumann Polynomial preconditioner -Poly GPU- for GPU architecture provided by the MCG-Solver library.

In our experimentations we don't evaluate the GPU version of BSSOR and ILU(0), which are already discussed in a previous work [1].

Platform Description

We used the following platforms to present our work: one workstation and a GPU cluster. The workstation includes 2 Ivy Bridges E5-2680v2@2.8Ghz, each processor holds 10 cores and two Tesla K40c GPU cards. The GPU part of ENER110; IFPEN cluster contains 5 nodes; each node includes 2 Sandy Bridge

E5-2470@2.3Ghz processors; each holds 8 cores; and 2 K20m GPU cards. The nodes are interconnected with FDR infiniband links (56 Gbit/s). This cluster allows tests up to 80 cores or 10 GPUs.

Study Case Description

In our experimentations, we simulate two academic test cases, spe10[20] and pab[10] and one real field thermal model, all with fully implicit scheme.

The tenth SPE comparative solution -Spe10 [20]- is an incompressible water-oil, black-oil, two-phase flow problem. The geological model consists in a part of a Brent sequence. It includes two units. The top part of the model −21.336 m (70 ft), 35 layers- belongs to the Tarbert formation and corresponds to a prograding near shore environment. Its lower part -Upper Ness, 30.48 m (100 ft), 50 layers- is fluvial. The model is discretized over a regular Cartesian grid and contains 1,122,000 cells with heterogeneous porosity and permeability, with no top structure or faults. It simulates a 3D waterflood in a geostatistical model over a 2000 day period. In this model the production scheme is based upon a classical inverted "5-spot" pattern with four producers located in the corners and an injector in the center defined and specified with bottom hole pressure.

The pab test case is a synthetic reservoir model built with the geological data of a reservoir located in the turbidite system of the PAB formation studied in [10]. The test case is composed of a 30x51x494 reservoir grid filled with porosity and permeability data obtained by an upscaling procedure of a geological model on 25 million cells grid. The fluid system is described by a standard triphasic Black-Oil PVT model, and the poro-hydrodynamic system, by a set of five rock type, defined by five KrPc model based on tabulated relative permeability curves. We simulate, over a period of 30 days, a classical "5 spots pattern" with 4 water injection wells in the 4 corners of the reservoir and a production well in the center, with max limit bottom hole pressure and max water volume rate constraints.

The Hangingstone oil sands reservoir is located near Fort McMurray in Canada. Indeed, the bitumen viscosity at reservoir condition is over 1,000,000 mPa.s and will not flow naturally. A Steam Assisted Gravity Drainage -SAGD- process is used to extract bitumen from the Hangingstone reservoir. Based on the analysis of the data, it was decided to drill two pairs of 500 m horizontal wells in 1997 and to start operating the SAGD process in 1998. The expected well performance calculated using a thermal simulator has been presented in [12]. The simulation sector model used for this study has 203,770 active cells and one pair of two horizontal wells. One year of the SAGD process is simulated.

Performance results

In this section, we present the results of several simulations generated from the reservoir models presented with different preconditioners. Then, we analyse the obtained performance gain due to GPU implementations of the preconditioned linear solver. Since we want to know if using GPUs for linear system resolution can really improve solver performance we have to take into account the balance between the number of processor cores and the number of GPUs. This means that if we consider a system with 1 GPU per socket we cannot compare 1 GPU to 1 core, but to the full socket with 8, 10 or more cores. Therefore, in order to have a fair comparison, the CPU based simulations were run using all socket's cores -8, 10 or more- while simulations with GPU solver were run using 1 core and 1 GPU per socket.

The simulations performed, using the mentioned preconditioners of the IFPSolver and MCGSolver, are performed without any accuracy loss. All the figures present the total solver time -in seconds- including both solver setup and iterative solve time for the global simulation.

Figure 1, Figure 2 and Figure 3 present the results obtained the SPE10 study case with 10^{-4} convergence tolerance and 5000 maximum iterations for BiCGStab. We illustrate both single-socket -presented in Figure 1- and dual-socket -presented in Figure 2- configurations on the workstation. Moreover in Figure 3 we present the the results obtained on ENER110 GPU part. In these figures we show results with GPUs for Polynomial and CPR-AMG preconditioner and CPU results with ILU0 and CPR-AMG preconditoners.

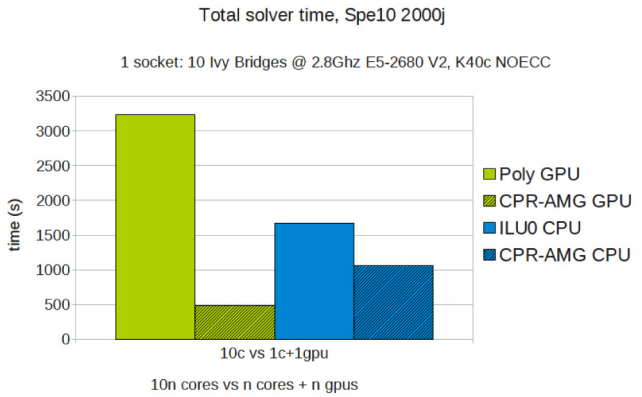

Figure 1—CPU and GPU version of CPR-AMG with PMIS comparing with other preconditioners running on single- socket (Ivy Bridge workstation)

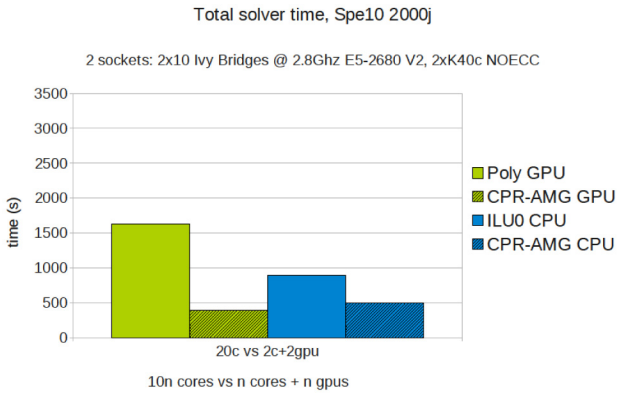

Figure 2—CPU and GPU version of CPR-AMG with PMIS comparing with other preconditioners running on dual- sockets (Ivy Bridge workstation)

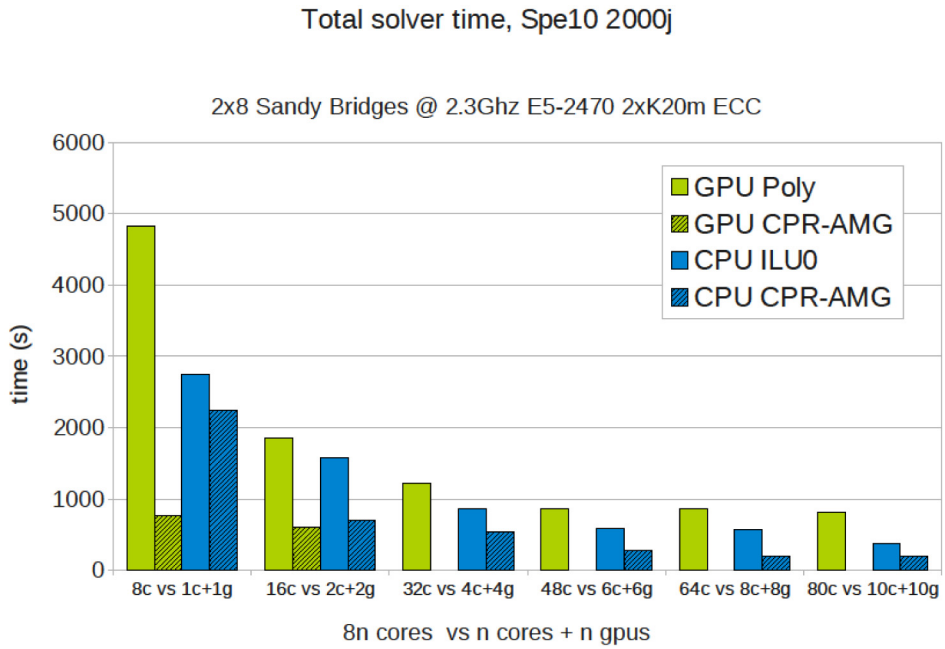

Figure 3—CPU and GPU version of CPR-AMG with PMIS comparing with other preconditionners running on ENER110

For the Polynomial preconditioner these results show that even if we have a good intrinsic scaling: acceleration is close to 2 for 2 GPUs compared to 1 GPU and close to 6 with 6 GPUs. This precontionner does not compete with CPU preconditioners because it gives very poor convergence rates on SPE10: the total number of solver iterations varies from 177000 to 194000 while CPU ILU0 performs from 38000 to 47000 iterations.

Figures 1 and 3 show that in the single-socket, the performance is improved by a factor 2 when using the GPU version of CPR-AMG -using one core and one GPU- with respect to the CPU version -using all the cores of one socket-.

Figure 2 shows that in the dual-socket configuration, the GPU version of CPR-AMG -using 2 cores and 2 GPUs- enables around 20% gain with respect to the CPU version -using all the dual-socket cores. These results are particularly interesting, when we look at the cumulative number of solver iterations. The GPU version of CPR-AMG performs roughly 8000 solver iterations compared to 4000 for the CPU version of CPR-AMG. The obtained gain could be better if the setup phase was less time consuming. Anyway on

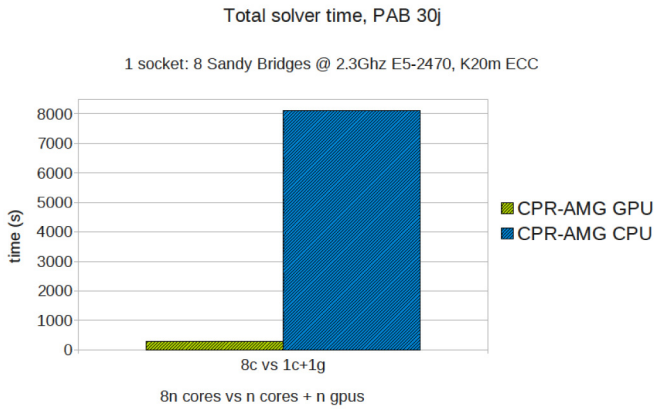

Figure 4—CPU version of CPR-AMG with PMIS and aggregation AMG for GPU version comparing with other preconditioners running on single socket (ENER110 Cluster)

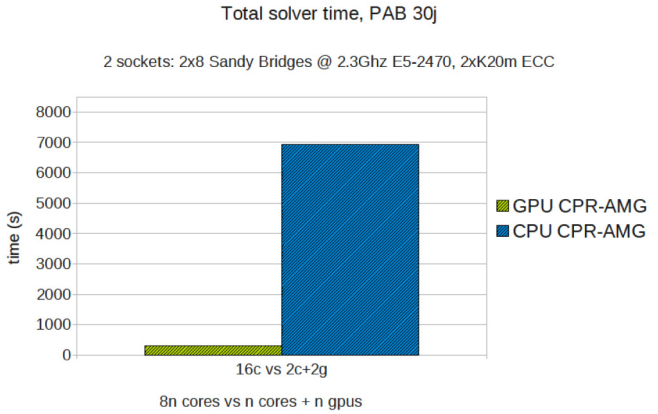

Figure 5—CPU version of CPR-AMG with PMIS and aggregation AMG for GPU version comparing with other preconditioners running on dual sockets (ENER110 cluster)

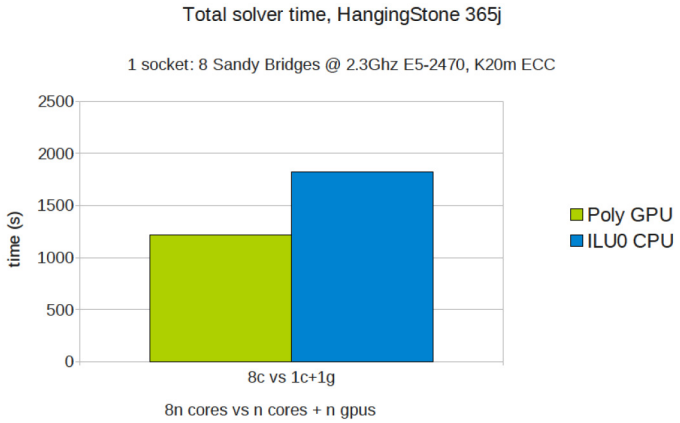

Figure 6—CPU version of ILU(0) comparing with GPU version of Polynomial running on single socket (ENER110 cluster)

this test case and with our configuration GPU CPR-AMG doesn't scale beyond one node while CPU ILU0 scales up to 80 cores and CPU CPR-AMG scales up to 64 cores.

Figure 4 and Figure 5 present pab study case results with 10^{-4} convergence tolerance and 1000 maximum iterations. The simulations are run on the ENER110 platform. We illustrate both single-socket -presented in Figure 4- and dual-socket -Figure 5- configurations on one ENER110 node.

In this test case, which has many degenerate cells, the aggregation AmgX manages better than Hyper AMG as part of the CPR preconditioner. Also the efficient implementation of GPGPU SpMV, used within the Neumann Polynomial preconditioner contributes to this performance gain. Moreover, this GPU version of CPR-AMG performs roughly 13214 linear solver iterations compared to around 78746 for the CPU version of CPR-AMG.

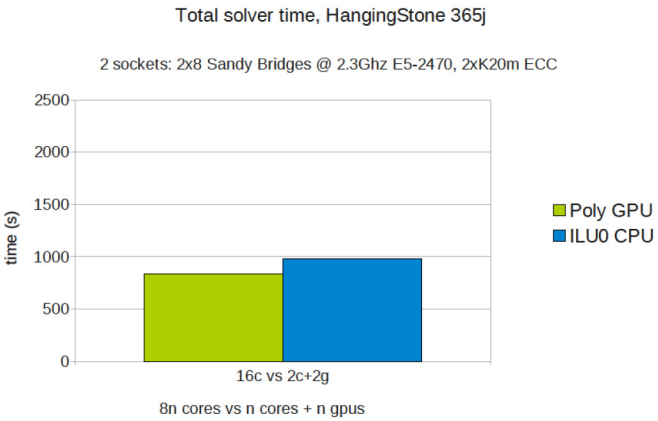

Figure 7—CPU version of ILU(0) comparing with GPU version of Polynomial running running on dual socket (ENER110 cluster)

The experiments performed with spe10 and pab show that thanks to CPR acceleration and to AMG, large-scale and/or complex reservoir problems, in fully implicit formulation, can be solved efficientl.

Regarding the Hangingstone study case presented in Figure 6 and Figure 7 with 10^{-5} convergence tolerance and 1000 maximum iterations for BiCGStab, since this case is relatively small (203,770 actives cells), tests are only done with 1 and 2 sockets. The results are quite different from the previous tests: These two figures highlight the benefit obtained by using the hybrid CPU+GPU implementation of Polynomial preconditioner. In this test case the GPU version of CPR-AMG decreases the number of iterations to the convergence considerably -23925 for the GPU version of CPR-AMG instead of 115959 for GPU version of Polynomial- but each iteration is more expensive. This permits the Polynomial preconditioner to offer a better performance -around 10% - . Figure 6 demonstrates that in the single-socket configuration, the performance of the GPU preconditioner is around 30% better than complete single-socket cores. Regarding the dual-socket configuration for this case this gain is around 15%, as illustrated in Figure 7.

Conclusion

In this paper, we show the feasibility of exploiting the power of heterogeneous architecture for Geoscience dynamic simulations for real models. We illustrate that, a single GPU may perform around 2x faster than a complete single-socket processor. With large enough problem sizes, with two GPUs, the simulation may be up to 2x faster than with a complete dual- socket cores.

The presented simulations are the early-obtained results with CPR-AMG with AmgX. Hence, in future work, we expect to improve the AmgX use in MCGSolver, by replacing just the modified coefficients in the matrix structure and reusing a certain number of levels from the top to the fine levels of AmgX in order to reduce the setup phase cost.

Acknowledgements

The authors would like to acknowledge Maxim Naumov from NVIDIA for the suggestions and providing the most recent AmgX realeases used in this work.

References

1. Anciaux-Sedrakian, A., Gottschling, P., Gratien, J-M., Guignon, T. (2014). Survey on Efficient Linear Solvers for Porous Media Flow Models on Recent Hardware Architectures. *Oil and Gas Science and Technology, Institut Français du Pétrole,* **69**(4), pp. 753–766.
2. Appleyard, J., Appleyard, J., Wakefield, M., & Desitter, A. (2011, January 1). Accelerating Reservoir Simulators using GPU Technology. *Society of Petroleum Engineers*. doi: 10.2118/141402-MS
3. Bell, N. and Garland, G. (2009). Implementing Sparse Matrix-Vector Multiplication on Throughput-Oriented Processors. SC'09: SC'09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. doi: http://doi.acm.org/10.1145/1654059.1654078
4. Buck I., Foley T., Horn D., Sugerman J., Fatahalian K., Houston M., Hanrahan P. (2004) *ACM Transactions on Graphics* **23,** pp. 777–786
5. Cohen J. and Castonguay P. (2012). Efficient Graph Matching and Coloring on the GPU. GPU Technology Conference, San Jose, CA.,
6. De Sterck H., Yang U.M., Heys J. (2006) Reducing complexity parallel in algebraic multigrid preconditioners, in *Proceeding of SIMAX* **27,** 1019–1039
7. Dogru, A. H., Fung, L. S. K., & Sindi, M. O. (2013, February 18). Multi-Paradigm Parallel Acceleration for Reservoir Simulation. *Society of Petroleum Engineers*. doi: 10.2118/163591-MS
8. Eaton, J. *AmgX: Multi-Grid Accelerated Linear Solvers for Industrial Applications*

9. Esler, K. P., Natoli, V., Samardzic, A., Atan, S., and Ramírez. B. (2012). "Accelerating reservoir simulation and Algebraic Multigrid with GPUs." GPU Technology Conference, San Jose, CA.,
10. Euzen, T., R. Eschard, E. Albouy, and R. Deschamps, 2007, Reservoir architecture of a turbidite channel complex in the Pab Formation, Pakistan, in T. H. Nilsen, R. D. Shew, G. S. Steffens, and J. R. J. Studlick, eds., *Atlas of deep-water outcrops: AAPG Studies in Geology 56,* CD-ROM, 20 p.
11. Grospellier, G. & Lelandais, B. The Arcane Development Framework. POOSC'09, July 7, 2009, Genova, Italy.
12. Ito, Y. and Suzuki, S. 1999. Numerical Simulation of the SAGD Process In theHangingstone Oil Sands Reservoir. *J Can Pet Technol* **38**(9):27–35. PETSOC-99-09-02. http://dx.doi.org/10.2118/99-09-02
13. Klie, H. M., Sudan, H. H., Li, R., & Saad, Y. (2011, January 1). Exploiting Capabilities of Many Core Platforms in Reservoir Simulation. *Society of Petroleum Engineers*. doi: 10.2118/141265-MS
14. Lacroix S., Vassilevski Yu, Wheeler J., Wheeler M.F. (2003) Iterative solution methods for modeling multiphase flow in porous media fully implicitly, *SIAM Journal* **25,** 3, pp. 905–926.
15. Stuben K. (2001) *Algebraic Multigrid: An Introduction for Positive Definite Problems with Applications,* Academic Press, pp. 413–532
16. NVIDIA Corp. http://www.nvidia.com/object/cuda_home_new.html.
17. NVIDIA Corp http://docs.nvidia.com/cuda/cublas/#batching-kernels
18. http://www.khronos.org/opencl/
19. Saad Y. (2001). *Iterative Methods for Sparse Linear Systems,* second edition, SIAM.
20. *SPE10 The 10 th SPE comparative Solution Project* http://www.spe.org/web/csp/datasets/set02.htm
21. W.L. Briggs, V.E. Henson and S. F. McCormick. (2000) *A Multigrid Tutorial,* second Edition, SIAM, 137–159