Parallel 3D Sweep Kernel with PARSEC

Salli Moustafa, Mathieu Faverge, Laurent Plagne, Pierre Ramet

Parallel 3D Sweep Kernel with PaRSEC

Salli Moustafa*, Mathieu Faverge†, Laurent Plagne*, and Pierre Ramet†
*EDF R&D 1, Av du Général de Gaulle F92141 CLAMART CEDEX France
†INRIA-University of Bordeaux, France

I. Introduction

The Boltzmann Transport Equation (BTE) is a fundamental law governing the evolution of large sets of particles. BTE numerical solvers are involved in various physical contexts such as nuclear core simulation. A large class of BTE solvers are based on a so-called sweep algorithm that represents the vast majority of the computing operations. In a previous work, we have presented its parallelization for shared-memory architectures through Intel TBB library inside the DOMINO solver [1]. In this work, we show how we used the PaRSEC framework to dynamically schedule the task graph of the sweep operation on distributed memory architectures. Taking advantage of SIMD units of interconnected shared-memory nodes, we are able to fully exploit the computational power of modern supercomputers.

II. The Cartesian Transport Sweep

The considered algorithm consists in multiple sweeps across the cells of a rectangular grid that correspond to a set of angular directions. In 2D, these directions are grouped into quadrants. Fig. 1 represents such a sweep that corresponds to a quadrant gathering right-top angular directions. In this case, each cell has two input dependencies located on its left and bottom faces (IN_L and IN_B). The left and bottom faces of the grid being given, the corner cell (0,0) happens to be the first cell that can be processed. This cell process produces two outputs located on the right and top faces (OUT_R and OUT_T) that satisfy the dependencies of the neighboring cells (0,1) and (1,0). Consequently, all cells belonging to the same diagonal of the grid may be processed in parallel, and step by step all grid cells are evaluated for the considered quadrant. This operation is identically repeated for all the 4 quadrants. In the following, we consider that one can perform the four sweeps simultaneously.

III. Sweep Implementation on top of PaRSEC

The PaRSEC [2] runtime system is a generic data-flow engine supporting a task-based implementation and targeting distributed hybrid systems. The main concept behind this framework is the scheduling of an abridged representation of the Direct Acyclic Graph (DAG) describing the algorithm among available computing resources. That means that the user only needs to: 1) transcript his problem in a DAG, using the symbolic representation, and 2) provide the data distribution. PaRSEC is then in charge of distributing the tasks onto the underlying architecture with the classic owner compute rule.

PaRSEC requires to describe the DAG using the Job Data Flow (JDF) symbolic representation internal to the framework. It consists in defining several parameters: the execution space of the task graph; the incoming data of each task, and the outgoing data released by these tasks. Within the PaRSEC framework, there is a separation between the algorithm, the data distribution, and the computation placement. This can be very useful as it allows easy integration of several data/task distributions with minimal effort. A distribution is specified by implementing a given API. This API is then used by the runtime to compute the data/task placement and then manage data movements between the computing resources if needed. For example, network communications, through MPI calls, are issued when a task mapped on a given node needs data located on another node. In our current implementation, the data distribution of the 3D cube used is a block distribution of the cells over the (P, Q, R) grid defined by the $N = PQR$ nodes.

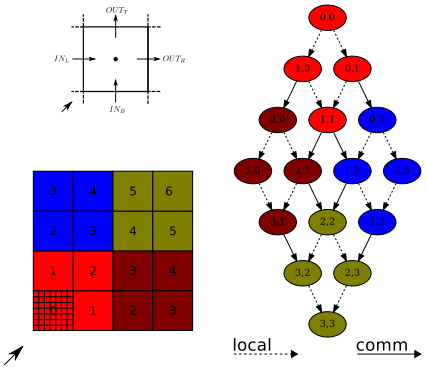

Fig. 1: Sweep over a 4×4 grid cells for a single direction. Two input data, IN_L and IN_B, are needed to start the treatment processing each cell. Cells of the same color are located on the same node. On the task graph to the right, tasks belonging to the same horizontal line can be processed in parallel.

IV. Performance Measurements

Performance measurements were carried out on shared and distributed memory settings using two test cases `small` and `big` (see Table I).

	BIGMEM	IVANOE
Nodes	1	64
N_{core}	32	768
Processor	Intel X7560	Intel X5670
Th. peak	0.57 Tflop/s	18 Tflop/s
Network	–	QDR IB

	small	big
N_{dir}	288	288
$ncells_x$	120	480
$ncells_y$	120	480
$ncells_z$	120	480
$TFlop$	0.012	0.796

TABLE I: Characteristics of the supercomputers and test cases used. On the right, $TFlop$ gives the number of floating point operations required for one complete sweep.

A. Shared Memory Results

We conducted a comparative study of the performances obtained with the two implementations of the sweep: the one with INTEL TBB versus the one with PARSEC. This study was conducted on the shared memory supercomputer `BIGMEM`, using the test case `big`, and results are shown on Fig. 2. The performance of the sweep implementation on top of PARSEC reached 291 Gflop/s (see Fig. 2), corresponding to 51% of the theoretical peak performance of this supercomputer. PARSEC implementation scales better and is 8% faster than our previous INTEL TBB code. The parallel pattern being constant, we interpret this improved speed-up as a sign for a reduced scheduling overhead for the PARSEC framework.

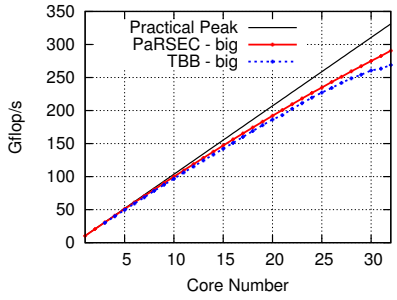

Fig. 2: Comparison between INTEL TBB and PARSEC on the `BIGMEM` supercomputer using the `big` test case.

B. Distributed Memory Results

Associating one MPI process per node and as many threads as available cores to execute local tasks, PARSEC allows us to use *Hybrid* parallelism for our solver. In this case, only inter-nodes data transfers are achieved through MPI calls, while shared-memory is used within the nodes. This approach differs from the classic one-level, or *Flat*, approach, where all data movements are achieved through MPI calls. The following sections details differences between these two approaches for the sweep operation.

The PARSEC framework allows us to establish priorities on the task scheduling made by the runtime. As shown in Fig. 3, we define priorities in order to first execute tasks belonging to the same z plane. This priority allows us to reduce the idle time for central nodes.

1) Hybrid: The implementation of the sweep on top of PARSEC scales up to 768 cores of the `IVANOE` supercomputer and produces a sustained performance of 6.2 Tflop/s (see Fig. 4), corresponding to 34.4% of the theoretical peak performance of this supercomputer. We see that without priorities the performance is reduced by more than 26%.

2) Hybrid versus Flat approaches: Performances measurements discussed in this section are obtained using `small` test case on the `IVANOE` supercomputer. Indeed, considering the `big` test case requires more cores in order to bring out differences between *Hybrid* and *Flat* approaches. Distribution of nodes used in this case are those described by Adams et al. [3]. As shown on the Fig. 5, up to 96 cores, both approaches exhibit similar performances. Above 100 cores, the

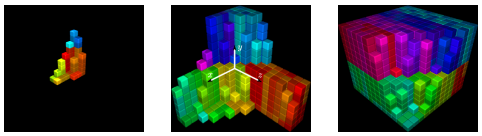

Fig. 3: Snapshots showing the progress of the sweep throughout the spatial domain. The distribution of nodes used is $(4, 2, 1)$. Threads belonging to the same node have similar colors.

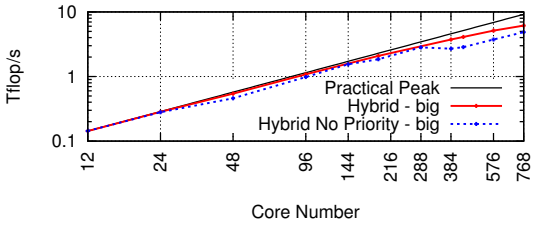

Fig. 4: Strong scalability of the sweep operation on the `IVANOE` supercomputer using the test case `big`.

Hybrid approach outperforms the *Flat* one. Indeed, distribution of nodes used at 384 cores being $(4, 4, 4)$ for *Hybrid*, and $(16, 12, 2)$ for *Flat*, it is clear that the latter involves much more MPI communications.

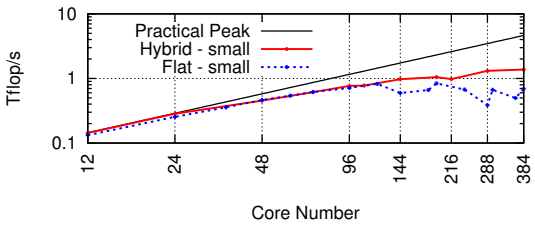

Fig. 5: Strong scalability of the sweep operation on the `IVANOE` supercomputer.

V. CONCLUSION

We have presented an implementation of the sweep operation on top of the PARSEC runtime. This implementation combines efficiently three levels of parallelism: SIMD, Multi-Threads, and Message Passing. Measured performance reaches 6.2 Tflop/s, and corresponds to 34.4% of the theoretical peak performance of the `IVANOE` supercomputer.

REFERENCES

[1] S. Moustafa, I. Dutka-Malen, L. Plagne, A. Ponçot, and P. Ramet, "Shared memory parallellism for 3D cartesian discrete ordinates solver," in *Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo (SNA + MC 2013)*, Paris, October 2013.

[2] G. Bosilca, A. Bouteiller, A. Danalis, T. Hérault, P. Lemarinier, and J. Dongarra, "DAGuE: A generic distributed DAG engine for High Performance Computing," *Parallel Computing*, vol. 38, no. 1-2, 2012.

[3] M. P. Adams, M. L. Adams, W. D. Hawkins, T. Smith, L. Rauchwerger, N. M. Amato, T. S. Bailey, and R. D. Falgout, "Provably optimal parallel transport sweeps on regular grids," in *Proc. International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering, Idaho*, 2013.