MAXELER Technologies Maximum Performance Computing
2012 Symposium on Application Accelerators in High Performance Computing

Acceleration of a Meteorological Limited Area Model with Dataflow Engines

Diego Oriato, Simon Tilbury Application Acceleration Maxeler Technologies London, UK
diego@maxeler.com, simon@maxeler.com

Marino Marrocu, Gabriella Pusceddu Environmental Sciences Research Program CRS4 Pula, Italy marino@crs4.it

Abstract— Climate and weather modeling is a significant consumer of High Performance Computing due to the hard deadlines inherent in predicting weather. Given the large data volumes and runtimes involved, climate and weather modeling is ideally suited for dataflow computation. In this paper, we demonstrate a dataflow implementation of the dynamic core of a meteorological limited area model. To achieve maximum performance we transform the computation by reordering operations and encoding the data. We present results for a domain of 13,600 x 3,333 x 30 km with 620 thousand grid points, and show speedups of up to 74x comparing an x86 CPU node to a dataflow node. Keywords: Macceleration, dataflow, weather, meteorology, hydrostatic, climate, performance, paralellism

I. INTRODUCTION

As microprocessors reach the limits of attainable clock frequencies and power consumption, hardware vendors are scaling up the number of cores per node, thereby enabling higher degrees of parallelism [1]. To take advantage of potential and only modest speed improvements, existing software must be modified. Alternative computing technologies are emerging as a powerful solution to overcome the slowdown of scaling with traditional processors. In this paper we investigated the feasibility of implementing a realistic full dynamic core of a meteorological model using Dataflow Engines (DFE). Dataflow programming focuses on modeling a program as a directed graph of operations by arranging the calculation in a parallel pipeline where thousands of operations are computed every cycle at relatively low clock frequencies. We focused on the dynamic core of a Limited Area Model (LAM) derived from the BOLAM model [2], which is a research-oriented hydrostatic LAM developed by ISAC-CNR (Bologna, Italy) and parallelized using domain decomposition and message passing libraries [3]. A complete version of a hydrostatic LAM is made up essentially of three blocks: initialization, post-processing and the meteorological model that numerically solves the prognostic equations governing atmospheric circulation. The last part, the most complex and computationally expensive, is composed of two macro blocks: the dynamic core and the physical parameterizations routines (of which the most computational expensive is the radiation). The aim is to develop a computationally fast version of the dynamic core of the hydrostatic model to satisfy requirements of higher spatial resolution for regional weather forecast, large domain dimension for global models, and very long time integration typical of climate simulations. Our work can be extended to the dynamic core of a non-hydrostatic model, as long as explicit time integration schemes are used for the primitive equations.

II. LIMITED AREA MODEL

For the Limited Area Model (LAM) we utilize a numerical meteorological model which integrates the primitive equations (PE) in time over a regional horizontal domain covering part of the Earth. The primitive equations are a set of nonlinear 3D partial differential equations that approximate global atmospheric flow. They consist of equations for the conservation of momentum, continuity and thermal energy. The two equations for the horizontal conservation of momentum for the zonal velocity u and the meridional velocity v are

$$\frac{\partial u}{\partial t} = -\frac{u}{ah_x}\frac{\partial u}{\partial \lambda} - \frac{v}{a}\frac{\partial u}{\partial \varphi} - \dot{\sigma}\frac{\partial u}{\partial \sigma} - \frac{R_d T_v}{ah_x}\frac{\partial \ln p_s}{\partial \lambda} - \frac{1}{ah_x}\frac{\partial \Phi}{\partial \lambda} + F_u \quad (1)$$

$$\frac{\partial v}{\partial t} = -\frac{u}{ah_x}\frac{\partial v}{\partial \lambda} - \frac{v}{a}\frac{\partial v}{\partial \varphi} - \dot{\sigma}\frac{\partial v}{\partial \sigma} - \frac{R_d T_v}{a}\frac{\partial \ln p_s}{\partial \varphi} - \frac{1}{a}\frac{\partial \Phi}{\partial \varphi} + F_v, \quad (2)$$

for time t, latitude and longitude coordinates and , geopotential field , mean earth radius a, scale factor hx, gas constant for dry air Rd and virtual temperature Tv. As vertical coordinate (altitude) the a-dimensional terrainfollowing variable =p/ps (sigma coordinate) is used, with pressure p and surface pressure ps. The continuity equations for surface pressure ps and specific humidity *q* are

$$\frac{\partial p_s}{\partial t} = -\int_0^1 \nabla \cdot (\vec{V}_h \frac{\partial p}{\partial \sigma}) d\sigma \quad (3)$$

$$\frac{\partial q}{\partial t} = -\frac{u}{ah_x}\frac{\partial q}{\partial \lambda} - \frac{v}{a}\frac{\partial q}{\partial \varphi} - \dot{\sigma}\frac{\partial q}{\partial \sigma} + F_q. \quad (4)$$

where Vh is the horizontal wind in vector form, for which Cartesian components are u and v. Finally the thermodynamic equation expressing Mconservation of potential temperature is

$$\frac{\partial \theta}{\partial t} = -\frac{u}{ah_x}\frac{\partial \theta}{\partial \lambda} - \frac{v}{a}\frac{\partial \theta}{\partial \varphi} - \dot{\sigma}\frac{\partial \theta}{\partial \sigma} + F_\theta. \quad (5)$$

The PE (1-5) describe the time evolution for the five prognostic variables u, v, ps, q and However a number of other diagnostic equations are used to express functional relations between the prognostic variables.

Terms Fu, Fv, Fq and Fθ in (1), (2), (4) and (5) represent contributions to the tendencies from the parameterization of physical processes such as radiation [4], convection [5], dry adiabatic adjustments, surface friction, soil water and energy balance, large scale precipitation and evaporation. The contribution due to physical parameterization has been omitted in this job and our migration is therefore limited to m the dynamics of the model.

The system of differential equations is solved through integration in time, achieved by using a leapfrog scheme based on the staggered Arakawa C-grid [6]. First, time derivatives (tendencies) are calculated, at time t, and the prognostic variables are advanced to time t+t, where t is the time step of integration. Asselin time filtering [7] is then applied to suppress the computational noise arising from the three time level integration scheme. To suppress non linear instability and to limit energy concentration on grid scales, all prognostic variables except ps are subjected to a horizontal fourth-order diffusion operator.

Both zonal and meridional velocity are further processed using a divergence damping operator of the second order, solved using an Euler time scheme.

III. APPLICATION ANALYSIS

The process of accelerating a software application using Dataflow Engines consists of several stages:

A. Structural Analysis

The BOLAM algorithm uses an explicit finite difference time domain scheme to solve (1-5). The time step computation can be decomposed into five logical blocks:

a) Pressure Tendency: Calculation of the continuity equation (3) for the surface pressure integrated over the vertical dimension.

b) Diaged: Calculation of the logarithm of the surface pressure, vertical profile of the pressure, virtual temperature, geopotential and vertical velocity.

c) Tendency: Computation of the dynamic contributions to the prognostic equations as in (1), (2), (4) and (5).

d) Diffusion: Computation of the artificial horizontal diffusion for the prognostic variables.

e)Divergence & Damping: Calculation of the horizontal divergence and damping with a diffusion coefficient function of time.

Each block is structured as a 3D loop where longitude, latitude and altitude are respectively fast, medium and slow dimensions. Incrementing counters are employed for longitude and latitude corresponding to west-to-east and south-to-north movements.

The vertical dimension is described by the sigma coordinate which represents the ratio of pressure over surface pressure; its value is highest at the surface level.

Consequently a decreasing counter is employed for altitude. Due to the tight coupling nature of the PE, each block either modifies the prognostic variables or computes intermediate variables required by the next block making the computation of the blocks a strictly orderly serial process. Parallelization of the algorithm was implemented using 2D domain decomposition in the horizontal plane [3]. A full 3D decomposition is not convenient due to the integration over the vertical dimension of the surface pressure ps in the Pressure Tendency block. MPI was used for the exchange of the ghost regions, the size being determined by the stencil used in the Diffusion block.

B. Partitioning

We used the Maxeler Parton toolset to analyze the CPU time spent on each logical block of the application when run on an Intel Xeon core. The Tendency block is responsible for over 40% of the compute time, followed by the Divergence-Damping and the Diffusion. A relatively large overhead is caused by some memory management, treatment of boundaries conditions and by the advancing time integration scheme. This cost is evenly spread over all the five blocks described previously.

Based on the CPU profiling report, the Tendency block is the first candidate for acceleration with 160 floating point operations. Since we aim for acceleration of 100x or more, all the blocks would have to be migrated to the dataflow engine (DFE). By moving all the computation to the DFE we also minimize data transfer between CPU and DFE. We kept the time step control loop in the CPU and invoked the DFE computation per time step.

C. Transformation

In dataflow computing operations are implemented spatially as part of a pipeline through which data is streamed rather than each instruction being executed temporally on a new piece of data. For this reason a dataflow algorithm is inherently parallel.

The aim is therefore to group all the computational logic inside a single loop and to replace the loop with a deterministic data access pattern. This requires an understanding of the data dependency among the different parts of the algorithm. For example the Pressure Tendency block consists of a 3D loop whereas the surface pressure is reduced over the altitude in 2D. Before the surface pressure can be used in the successive Diaged block, the reduction has to be concluded, prohibiting merging the 3D loops in the Pressure Tendency and the Diaged blocks.

In order to avoid buffering the full three dimensional data between the two blocks or executing two streaming phases, which would double the time to stream the data, we exploited a feature of the order in which the functional blocks are executed in the algorithm. We observed that the "Pressure Tendency" block is actually the first computation done in the time step. Therefore computing the Pressure Tendency block at time step t only requires , data at time step t-1. As a consequence it is possible to precompute the Pressure Tendency at the end of time step t-1.

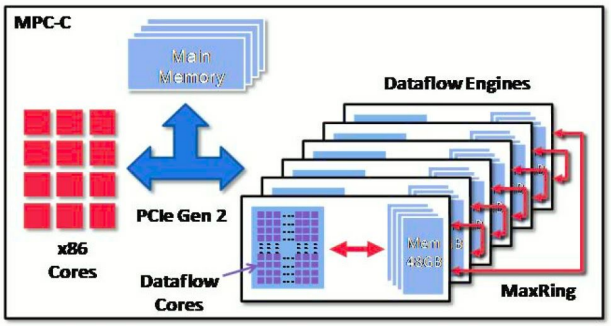

Figure 1. Hardware architecture of the Maxeler MPC-C series compute node used in this work

D. Parallelization

A dataflow engine comprises memory coupled to a chip implementing many dataflow cores, as shown in Fig.1. The dataflow cores are arranged in a pipeline which processes one item of data per clock cycle. To compute one grid point we need the five prognostic variables, their previous and next time step values and the constants describing the geometry and physics of the problem; we calculated that a single pipeline would use only 50% of the available memory bandwidth in a DFE, which suggests a pipeline of double capacity should be the target.

To facilitate this we converted part of the computation to use fixed-point arithmetic, which is more efficient in terms of silicon area per operation than the equivalent floating-point arithmetic. Analysis was performed on each block of the algorithm to establish the optimal fixed-point representations to maximize precision and avoid overflow, applying scaling coefficients where prognostic variables exhibited high dynamic range. In select places floating-point was retained to allow for high dynamic variability. With these modifications we were able to replicate the pipeline logic to compute two items of data in parallel.

The last axis of parallelism is derived from the six DFEs available in an MPC-C series compute node. One option was to decompose the problem among the DFEs. Since the grid of real-world datasets have a contained size (< 1 million points) we opted for a one-to-one mapping of serial simulation to DFE, giving the capability of running six independent simulations in parallel. If tackling larger problem sizes in future we could utilize a decomposition approach as shown possible by a recent study [9].

IV. EXPERIMENTAL SETUP

We ran the dataflow application on a Maxeler MPC-C series dataflow node (eight Intel Xeon E5506@2.13GHz CPU cores and six MAX3 Dataflow Engines connected to the CPUs via PCI Express as shown in Fig.1). Each MAX3 DFE utilizes a Xilinx Virtex 6 SX475T FPGA to implement the dataflow cores and 48GB of memory. Maxeler's MaxCompiler development environment was used to implement the dataflow pipeline and integrate it into the original FORTRAN application. The tested data flow design with a double capacity pipeline occupies 80% of the chip's internal memory and 58% of the DSPs (multipliers), and was set to run at 150MHz. All the variables needed during the computation were stored on the DFE. The prognostic variables were initialized from CPU and transferred out at the last time step of the simulation. The CPU controlled the time step loop by triggering the dataflow process and waiting for it to finish before repeating the trigger. No computation was executed on the CPU other than for initialization and post processing.

V. RESULT

The LAM dataset used is a medium size domain of 13600 km (160 points) in longitude, 3333 km (120 points) in latitude and 30 km (32 points) in altitude with a baroclinic atmosphere for the initial condition. Although boundary conditions limit the clear development of the baroclinic instability [8], we believe that this initial condition, together with the fixed boundary conditions, is a representative test to evaluate numerical stability of the migrated application. Spatial resolution chosen was of

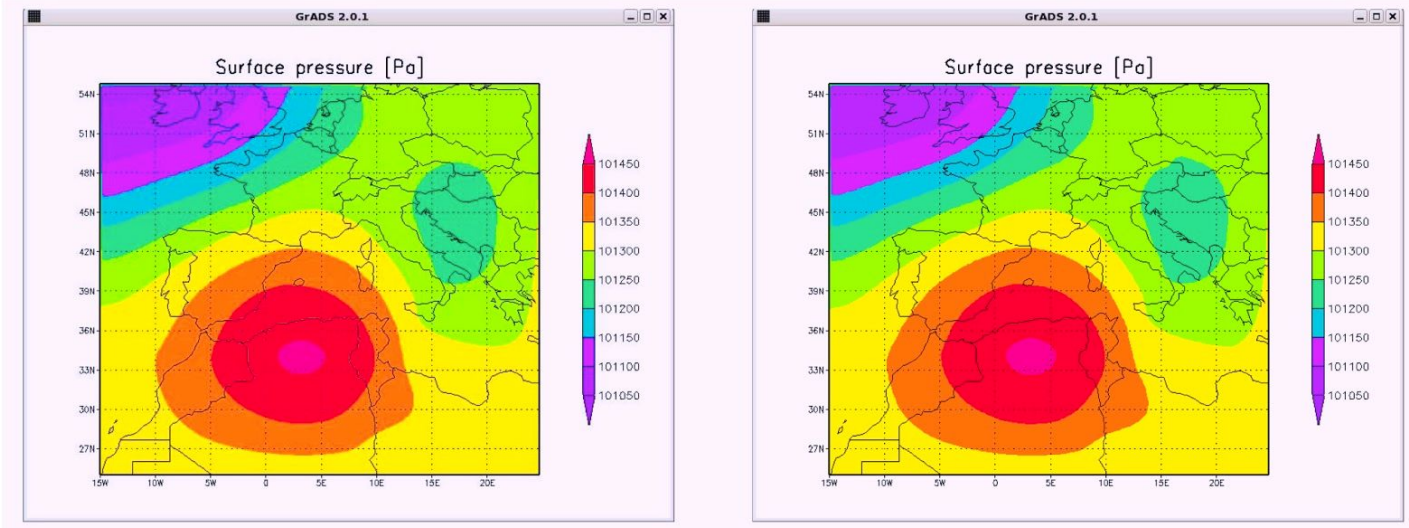

Figure 2. Surface pressure result after 500 time steps run on software (left) and Dataflow Engine (right)

0.25° both in latitude and longitude with a time step t = 15s to satisfy the CFL criteria [6], which guarantees the stability of the solutions. Fig.2 reports the surface pressure after 500 time steps for both the original software and the dataflow engine. The isobars are similarly placed with small differences, arising from the number representations used, only visible for low values of pressure and are at an acceptable level.

VI. SPEEDUP RESULTS

To accurately determine the speedup of the dataflow implementation we analyzed the dynamic behavior of the parallel software version to understand how efficiently the software exploits a multi-core node. In this contest we used an independent simulation of 5000 time steps.

The parallel software was run on a dual Intel Xeon X5650@2.7Ghz six-core CPU coupled to 192GB DDR3 memory. Results showed that the software does not scale very well when using more than two cores; four cores give a speedup of only 2.8x compared to a single core. Using all twelve cores becomes less efficient than running with four cores with only a 2.6x speedup. The highest speedup of 3.1x is reached with 8 cores.

Previous results showed that the software is limited by the speed of the memory bus. To understand whether it is the memory bandwidth or the contention in accessing the same data that constitutes the bottleneck we tested the performance by varying the number of independent tasks concurrently run. Fig.3 shows the core efficiency when running tasks using a single core each (blue bars); two cores each (green bars) and four cores each (red bars).

Running twelve independent tasks reduces the efficiency of the single core to 43%. However the speedup equivalent is 5.2x; the highest achieved by a single node. This is preferable to using all the cores of the node for fewer tasks.

In absolute terms the problem executes in an average of 960 seconds on the CPU-only node. To compare this with dataflow node performance we use the configuration achieving the maximum overall throughput, and compare the throughputs achieved. Twelve tasks, each using one core, give an average execution time of 2235 seconds.

We ran the same problem on the DFE in 15.1 seconds.

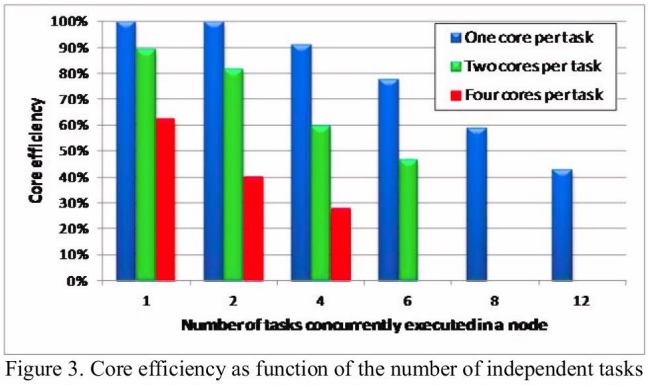

Figure 3. Core efficiency as function of the number of independent tasks running in a single 8-core node. Blue bars are for tasks using a single core, red bar for two cores per task, green bar refers to task using 4 cores.

Table 1 Speedup Results: Single Node

Speedup	Intel Xeon X5650@2.7Ghz	
	Single Core	*Node (12 cores)*
Single DFE	64x	12x
MPC-C Series Node	381x	74x

Exactly the same time was achieved with six tasks running concurrently. Since the computation is performed entirely within the DFE, there is no contention for the CPU resources of the MPC-C series node. Table 1 shows the speedup measurements relative to the CPU architecture. We compare one DFE against one core, measuring 64x performance speedup and the MPC-C series node, containing six DFEs, against the twelve-core Intel node, the fastest CPU solution, measuring 74x speedup. Peak power usage of a MPC-C series node was measured to be around 900W. Considering a peak value of 400W per Intel node, we need 74 twelve-core nodes and therefore around 30kW to match the performance of the dataflow solution.

VII. CONCLUSION

We migrated the dynamic core of a meteorological limited area model to Dataflow Engines. We analyzed the application and transformed the algorithm to best exploit the dataflow architecture. We optimized the number representations for various parts of the computation and parallelized each computational block. Results presented in this work for a domain of realistic dimensions have a good match with software. The speedup obtained between a six-DFE MPC-C series node and a twelve-core Intel node it is 74x.

[1] H Esmaeilzadeh, E Blem, R S Amanat, K Sankaralingam, D Burger, “Dark Silicon and the End of Multicore Scaling”, Proc. 37th International Symposium on Computer Architecture, (ISCA 2011).

[2] A. Buzzi, M. Fantini, P. Malguzzi, F. Nerozzi, “Validation of a limited area model in cases of mediterranean cyclogenisis: surface fields and precipitation scores”, Meteorolog. Atmos. Phys. 53 (1994) 137-153.

[3] M. Marrocu, R. Scardovelli, P. Malguzzi, “Parallelization and performance of a meteorological Limited Area Model”, Parallel Computing 24 (1998), 911-922.

[4] B. Ritter, J.-F. Geleyn, “A comprehensive radiation scheme for numerical weather prediction models with potential applications in climate solutions“, Mon. Wea. Rev. 120 (1992) 303-325.

[5] K. A. Emanuel, “A scheme for representing cumulus convection in large scale models“, J. Atoms. Sci. 48 (1991) 2313-2335.

[6] F. Mesinger, A. Arakawa, “Numerical methods used in atmospheric models", GARP Publications Series no,17 (1976), Volume 1.

[7] A. Asselin, “Frequency filter for time integration”, Mon. Wea. Rev. 100 (1972) 487-490.

[8] J. G. Charney, “The dynamics of long waves in a baroclinic westerly current.” J. Meteor., 4 (1947.) 135-162

[9] D. Oriato, O. Pell, C. Andreoletti, N. Bienati, “FD Modeling beyond 70hz with FPGA,” SEG 2010 HPC Workshop.