High-Performance Simulations of Turbulent Clouds on a Desktop PC

Exploiting the GPU

BY JERÔME SCHALKWIJK, ERIC J. GRIFFITH, FRITS H. POST, AND HARM J. J. JONKER

COMPUTATIONAL ATMOSPHERIC SCIENCE. Advances in atmospheric science have been strongly coupled with technological advances in computational resources. This started early in the twentieth century when it became apparent that an analytical solution to the Navier-Stokes equations in the context of atmospheric weather prediction would not be feasible. Richardson pioneered the numerical approach to weather prediction in 1922 using pencil and paper, but the first successful forecasts were not made until the 1950s, when digital computers became available. Since that time, the field of numerical weather and climate modeling has continued to grow, and its predictive accuracy has increased, as it has continued to take advantage of increasing computer power.

Today, atmospheric science relies heavily on numerical modeling on a variety of scales. Climate and weather predictions cover entire continents in large-scale models, whereas mesoscale models provide more detailed simulations of selected regions. On smaller scales, turbulent boundary layer processes and clouds are studied in high-resolution models like large-eddy simulations. Regardless of the scale, all of these models are computationally intensive.

AFFILIATIONS: SCHALKWIJK AND JONKER—Department of Multi-Scale Physics (Clouds, Climate and Air Quality Group), Delft University of Technology, Delft, the Netherlands; GRIFFITH—Petrotechnical Data Systems, Rijswijk, the Netherlands; POST—Department of Mediamatics (Data Visualization Group), Delft University of Technology, Delft, the Netherlands
CORRESPONDING AUTHOR: J. Schalkwijk, Delft University of Technology, Faculty of Applied Physics, Department of Multi-Scale Physics, Lorentzweg 1, 2628 CJ Delft, the Netherlands
E-mail: J.Schalkwijk@tudelft.nl

DOI:10.1175/BAMS-D-11-00059.1

Processor clock speeds have increased exponentially over the last several decades. This has gone a long way toward supplying the necessary computational power for running these numerical simulations. Yet the computational demands of the atmospheric models have outpaced even this exponential growth. Most numerical simulation codes have been parallelized so that they can take advantage of the extra computational power provided by supercomputers or computational clusters.

After years of predictable evolution, though, the high-performance computing landscape is now changing. Computer central processing units (CPUs) are now increasing in number of cores rather than clock speed. Specialized processing units such as graphics processing units (GPUs, or, more commonly, video cards) and field-programmable gate arrays (FPGAs) are being used increasingly for general-purpose numerical computing. Computational clusters and supercomputing facilities now have computing nodes with traditional processors, specialized processors, or both. The number of processing cores in computing nodes is also increasing. These changes mean that adapting numerical codes will be increasingly important if they are to get the most out of the available computing facilities.

However, these changes also offer new opportunities. By taking advantage of specialized processing units, some simulations may no longer need a cluster at all, which would allow them to return to the realm of the desktop computer. The GPU, in particular, is becoming a mature platform for running numerical simulations. It was designed to perform the intensive matrix projection calculations associated with gaming graphics. In order to efficiently and quickly perform such calculations, modern GPUs are designed as massively parallel calculating devices. Aided by the vast commercial market for visually high-performing computer games, these GPUs have experienced tremendous development while remaining reasonably priced. NVIDIA's current top-of-the-line gaming video card, the GeForce GTX 580, has 512 parallel computing cores, and is available off-the-shelf for less than \$500. Compared to CPU cores, which are traditionally designed for complex serial work, GPU cores are simpler at heart. Nevertheless, in the case of the GTX 580, their combined computing power reaches 1.58 TFLOPS (1.58 x 10^{12} floating point operations per second) in single precision. For comparison, the Dutch supercomputer facility SARA-HUYGENS reaches a total peak performance of 60 TFLOPS (albeit in double precision), but requires all its 1,728 dual-core processors to do so.

UTILIZING THE GPU. In order to effectively utilize the full power of today's supercomputer through large-scale parallelization, relatively large problem sizes are required. A "large" problem, in this context, should have a high ratio of time spent computing data to time spent communicating data when run on the supercomputer. The exact definition of "large" is hardware- and simulation-specific, but a CFD with $1{,}024^3$ or more grid cells should generally qualify.

As was recently noted by Michalakes and Vachharajani (2008), the strategy of increasing problem size is not always effective for problems that need fast time-to-solution ratios. They argued that for these problems, the kind of parallelism that the GPU offers has large advantages. To demonstrate this, they ported the time-consuming microphysics module of the Weather Research and Forecast (WRF) model to the GPU, which yielded a speed-up by a factor of 10 for this module and a factor of 1.23 for the weather model as a whole.

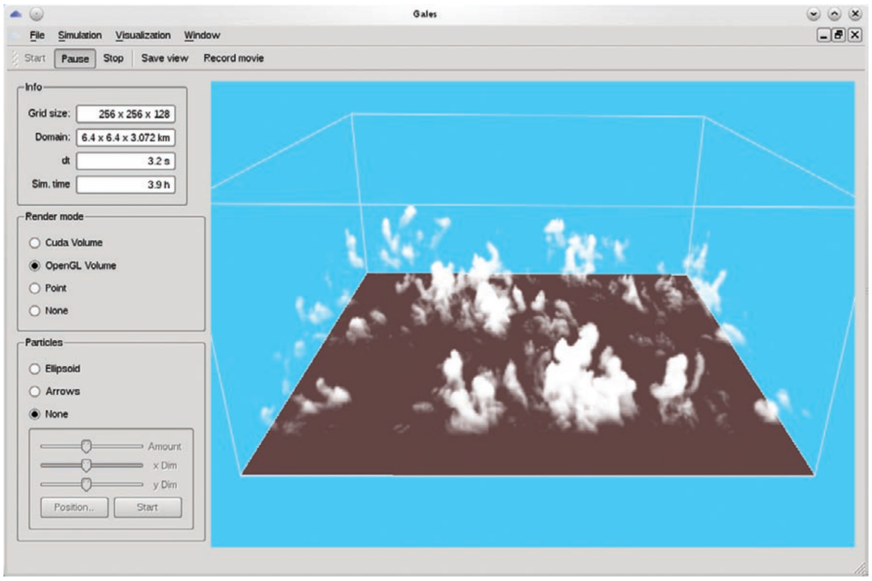
This is a promising development, especially since CPU-to-GPU transfer rates are rather slow. Michalakes and Vachharajani used a so-called accelerator strategy, in which a time-consuming module is off-loaded to the GPU. A disadvantage of this method is that the relevant data also has to be transferred to and from the GPU. In their case, this took 30% of the total time required by the GPU module.

Cohen and Molemaker (2009) remarked that the relative cost of data transfer will most likely continue to increase, and therefore proposed that even larger, and more sustained, speed-ups are possible in a "full implementation" design. In such a design, hereafter referred to as GPU-resident, the simulation data resides on the GPU, such that (almost) no transfer is needed between the CPU and the GPU and all heavy calculations can be performed on the GPU. They demonstrated the viability of such a design by creating a GPU-resident CFD code, which showed a performance increase of more than 800% compared to an eight-core CPU.

GALES: GPU-RESIDENT ATMOSPHERIC LARGE-EDDY SIMULATION. We have used the aforementioned full implementation strategy in our GPU-resident Atmospheric Large-Eddy Simulation (GALES). GALES is based on the Dutch Atmospheric LES (DALES) of which an extensive description is given by Heus et al. (2010). The general philosophy of LES (e.g., Deardorff 1970) is to explicitly resolve the larger, most energetic scales of turbulence motion and to model the smaller scales. By doing this, DALES can simulate daytime and nighttime atmospheric boundary layers, as well as cloud-topped boundary layers such as shallow cumulus and stratocumulus. A typical atmospheric LES encompasses domains of about 10 km wide and a few kilometers high, discretized on a grid containing 128^3 or 256^3 cells and with time steps on the order of a second. Such simulations can be typically run on a supercomputer or cluster using a number (32–64) of computational cores in parallel.

While supercomputers can provide the necessary computational capacity for such runs, their use also presents some disadvantages. As supercomputers typically use a batch-job queuing system, both pre- and postprocessing is required for a case study to be performed. The preprocessing involves preparing and submitting the simulation job to the queue. When finished, the interpretation of the simulation involves (sometimes extensive) postprocessing to visualize and interpret the produced data. These steps are time consuming and can be a serious bottleneck in research workflow when handling large amounts of data.

Although several ways exist to deal with or circumvent these limitations, a GPU inherently possesses the power to combine all the needed steps in a user-friendly process. GPU-equipped PCs are available at low cost and require relatively little power and maintenance. As will be shown below, GALES is able to handle problems, which would otherwise require 32–64 processors in a supercomputer or cluster, on a single GPU with comparable speed. Moreover, as the simulation data and results continuously reside on the GPU, it is possible to directly (during the simulation) render and interpret simulation data, thereby combining all data processing steps into one.


Gales
File Simulation Visualization Window
Start Pause Stop Save view Record movie

Info

Grid size: 256 x 256 x 128

Domain: 6.4 x 6.4 x 3.072 km

dt 3.2 s

Sim. time 3.9 h

Render mode: Cuda Volume [✓ OpenGL Volume] Point None

Particles: Ellipsoid Arrows [✓ None]

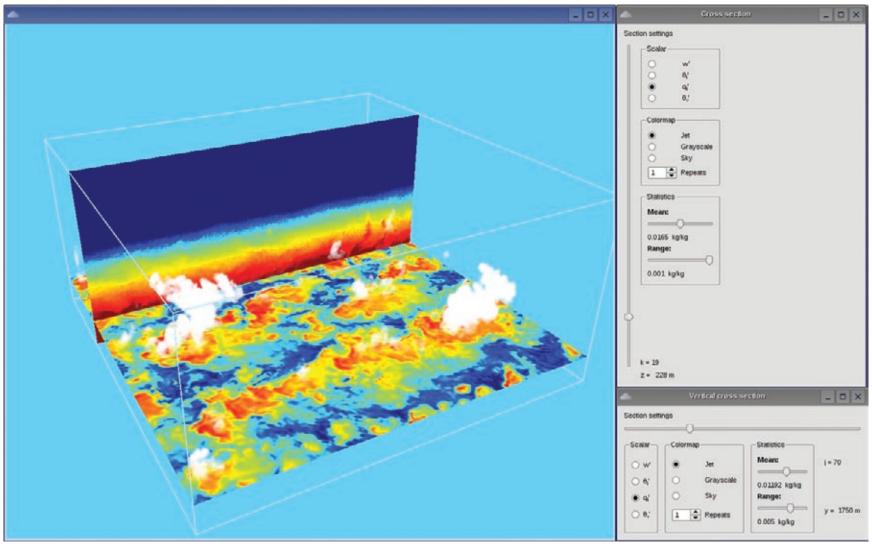
Amount
x Dim
y Dim
Position...
Start

FIG. 1. A screenshot of an interactive simulation with GALES. The 3D cloud field visualization is shown using volume rendering. During the simulation, the visualization can be actively zoomed and rotated to directly obtain insight into the simulation process.

GALES has a graphical user interface which shows statistical information as well as three-dimensional visualizations of the running simulation. An example of the 3D visualization is given in Fig. 1, which shows how GALES visualizes the current cloud field, volume-rendered for more realistic cloud appearance. Since all simulation data natively reside on the video card, these kinds of visualizations can be activated with minor impact on performance. Moreover, the three-dimensional visualization can be interactively navigated by rotating and scaling. This provides the possibility to study in detail the turbulent cloud processes happening in the simulation, without having to wait for the simulation to finish, or even having to write 3D fields to disk (which can severely reduce performance due to relatively slow disk access). Other processing possibilities include visualization of scalar fields such as temperature or humidity by an interactively placed 2D-cross section (Fig. 2), as well as live statistics plots for mean vertical profiles or time series. During the simulation, the user can also release Lagrangian particles to study dispersion characteristics or to investigate cloud-mixing properties. In addition, the simulation view, including the user's navigation actions, can be stored as an MPEG movie that can be played back later for review of the simulation or for demonstration purposes.

Working with full three-dimensional datasets is important for more than just a nice visual display: it provides a wealth of information that might otherwise be missed. An example of this is the shell of subsiding air surrounding cumulus clouds, which is vital to understanding dispersion in shallow cumulus fields but does not show up in ordinary cloud statistics.

GALES requires no further expensive software packages, nor does it require dedicated clusters or network facilities. Indeed, at the time of writing,



Cross section

Scalar: w' θ' [✓ q'] θ'

Colormap: [✓ Jet] Grayscale Sky

1 Repeats

Statistics

Mean: 0.0165 kg/kg

Range: 0.001 kg/kg

k = 19

z = 228 m

Vertical cross section

Scalar: w' θ' [✓ q'] θ'

Colormap: [✓ Jet] Grayscale Sky

1 Repeats

Statistics

Mean: 0.01192 kg/kg

Range: 0.005 kg/kg

i = 70

y = 1750 m

FIG. 2. While the simulation is active, users can add horizontal and vertical cross sections to visualize the 3D fields of velocity, temperature, or (in this case) humidity. The cross sections can be interactively positioned and the color scale adjusted.

PORTING TO THE GPU

GALES is based on the existing FORTRAN90-based Dutch Atmospheric LES [DALES; see Heus et al. (2010) for more detail]. All physical considerations and numerical algorithms are identical in nature, although sometimes implemented somewhat differently in GALES. This section gives a short summary of the considerations and requirements involved with porting to the GPU, but attempts to avoid technical details as much as possible.

GALES uses NVIDIA's CUDA (Compute Unified Device Architecture) to perform computations on the GPU. Other GPU programming models such as OpenCL work similarly. CUDA provides extensions to C/C++ allowing certain functions (*kernels*) to be executed on the GPU. These kernels divide the computation over a very large number of GPU *threads*. The GPU is optimized for data-parallel computation, in which each thread executes the exact same code but acts on a different portion of data. Many (atmospheric) modeling schemes are essentially data-parallel, which means that they can make good use of the GPU's power.

GPU parallelization is achieved through threading. A thread, in this case, can be seen as a virtual task that will be run on some processor when scheduled and is itself not parallel. For instance, four tasks might run in parallel on four processors, yet would run sequentially on a single processor. GPUs differ from CPUs in that they tend to run best with a very large number of threads—many more threads than processors—as they are efficient in scheduling and swapping threads to best utilize the GPU's architecture.

For a much more complete CUDA overview, the reader is referred to Sanders and Kandrot (2011) or the CUDA programming guide.

There are two typical strategies for using the GPU for numerical computation. The simplest is the "accelerator" strategy, which leaves the main code as is and moves only the most costly computations to the GPU. This provides a relatively easy speedup for many applications. However, it also requires the relevant data to be copied to the GPU and back, which requires the data to be transferred over the PCI-E bus every calculation step, seriously limiting the total speedup.

To avoid this bottleneck, GALES uses the other strategy, which is to completely redesign and rewrite the code and move (nearly) all computation to the GPU. Note that this required a full rewrite of the FORTRAN code in C++, after which the code was adapted stepwise for GPU computation. The end result is that data reside on the GPU throughout the simulation and not on the CPU, thus minimizing data transfer from GPU to CPU. In this design, the CPU code is used to direct the GPU to perform the needed calculations.

The GALES code is generally written such that, for each update to the grid, a single thread is initialized for each grid cell. For example, the advection code on a 128^3 grid initializes 128^3 (roughly 2 million!) threads, each updating the state of the particular cell it is assigned to. When calculating slab-averages or sums, each thread calculates the sum of a single row, after which the threads communicate to find the slab-average. Note that all threads can access the GPU's main memory, such that no MPI-like data segmentation is necessary.

As GPUs (at the time of writing) still perform significantly better in single precision than in double precision, sensitivity tests were done to implement as much as possible of the LES simulation in single precision. The pressure solver, which uses a Fast Fourier transform to determine the pressure fluctuations in order to enforce an incompressible flow, was found to be the most numerically sensitive step. For this reason, it is the single module in GALES which is performed in double precision.

In GALES, rendering of 3D information is done using OpenGL. CUDA can cooperate with either OpenGL or Microsoft's DirectX. In GALES, the cloud field is realistically rendered with CUDA, using the information already resident on the GPU. The resulting perspective image is then handled by OpenGL and displayed on screen. Similar protocols are followed for 2D slices or particles. The use of OpenGL has the advantage that GALES is compatible with Windows, Mac OS, and Linux systems.

simulations of 128^3 grid cells can be performed on the NVIDIA GeForce® line, available in computers costing less than $1,000, with a total power consumption of less than 500 W (such a machine is used for testing as described below). This makes high-performance simulations available at unprecedented low costs and ease of implementation.

Currently, the most important limitation of the GPU-resident implementation is memory. At the time of writing, the maximum amount of available memory available on an NVIDIA GPU is 6 GB, which is available in the second Tesla GPU line. As GALES needs more than 2 GB of memory to simulate a grid of 256^3 cells, 6 GB is insufficient to simulate a 512^3 cell grid; 16 GB would be required. Supercomputers are therefore a necessity for large problem sizes. For this reason, GALES has been designed such that it can start from or produce the exact same input files as its supercomputer-based equivalent DALES, allowing simulations to be easily scaled up and run on a supercomputer.

VERIFICATION. Although GALES is identical to DALES in terms of its algorithms

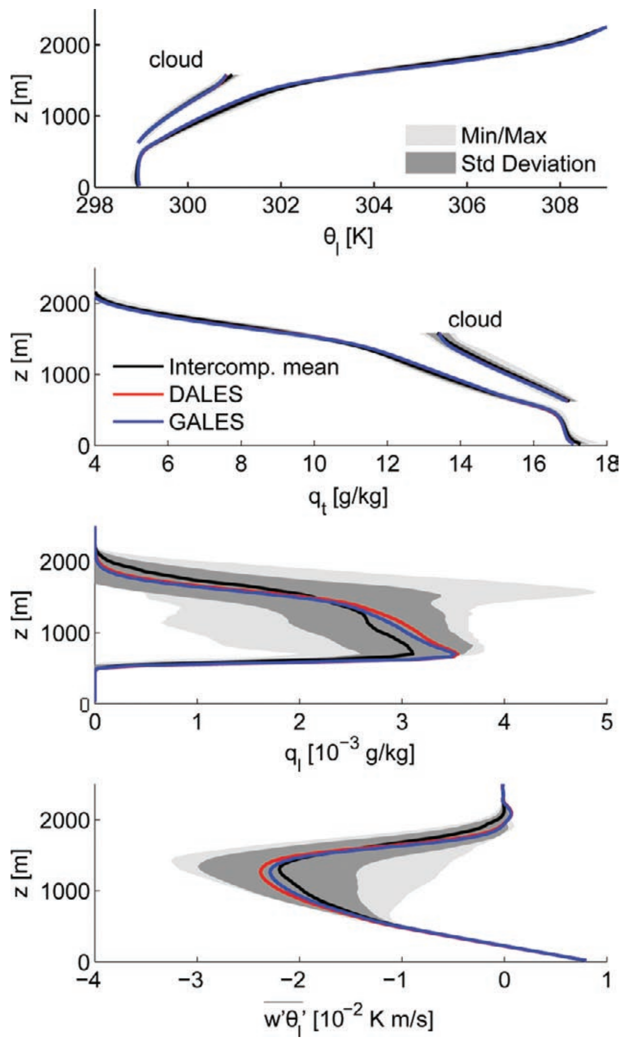


FIG. 3. Results from DALES (red) and GALES (blue) are compared in the context of the BOMEX intercomparison case. The results of DALES and GALES are 50-run ensemble averages; the shaded areas denote standard deviation and minimum/maximum of the BOMEX intercomparison cases.

and physics, it required a full rewrite of the code. Also, GALES performs computations in single precision where possible, while DALES performs all computations in double precision. The issues involved in porting the model are further explained in the "Porting to the GPU" sidebar at left; suffice it to say that the rewrite is so far-reaching that verification is in order. To verify whether GALES produces correct output with respect to DALES (in a statistical sense), the models are compared in the context of the BOMEX (Barbados Oceanic Meteorological Experiment) shallow cumulus case. BOMEX was used as a benchmark case to compare several different LES models. Figure 3 shows a summary of the results of 11 LES simulations that participated. The intercomparison mean of the profiles of liquid water potential temperature (θ_l), total water content (q_t), and liquid water content (q_l) are shown in black, complemented with cloud-sampled ($q_l > 0$) profiles. The dark gray area indicates the standard deviation, while the light gray area shows the minimum and maximum.

The profiles of DALES and GALES are shown in red and blue, respectively, denoting an ensemble average of 50 runs each, with each run having a slightly perturbed initial condition. From these figures, it can be concluded that GALES results do not deviate significantly from any LES model that participated in the intercomparison study, especially DALES.

SPEED. To put the performance of our GPU-resident implementation in some perspective, the performance of GALES is compared with that of DALES running in parallel on a 32-core IBM p575 (4.7-GHz) node of the Dutch supercomputer Huygens. In this test, GALES was running on an Intel Core i5 K655 quad-core (3.20-GHz) desktop PC equipped with an NVIDIA Tesla C1060 GPU. Secondly, GALES was tested on a budget (under $1,000) PC equipped with an Intel Core i5 K655 and an NVIDIA GTX 460 GPU. Figure 4 shows the wall clock time per time step per

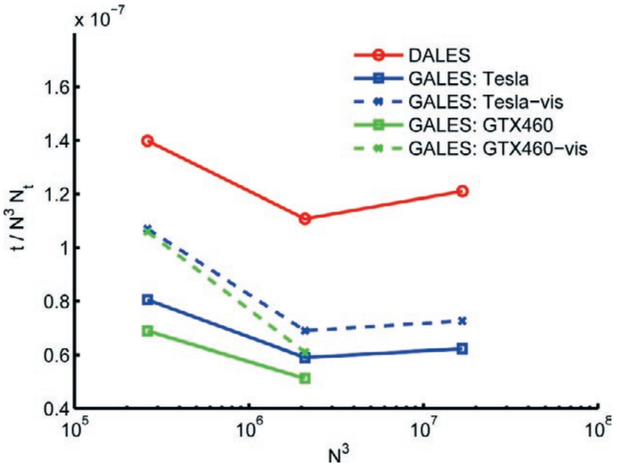


FIG. 4. Wall clock time, per time step per grid cell, plotted against the number of grid cells, for DALES (red) and GALES (blue and green), the latter with visualization deactivated or activated (solid and dashed lines, respectively). DALES is run on a single node of the Dutch supercomputer SARA-HUYGENS, GALES on a Tesla C1090 (blue) and GeForce GTX460 (green). Note that the GeForce is somewhat faster than the Tesla, but does not have the capacity to perform the 256^3 run.

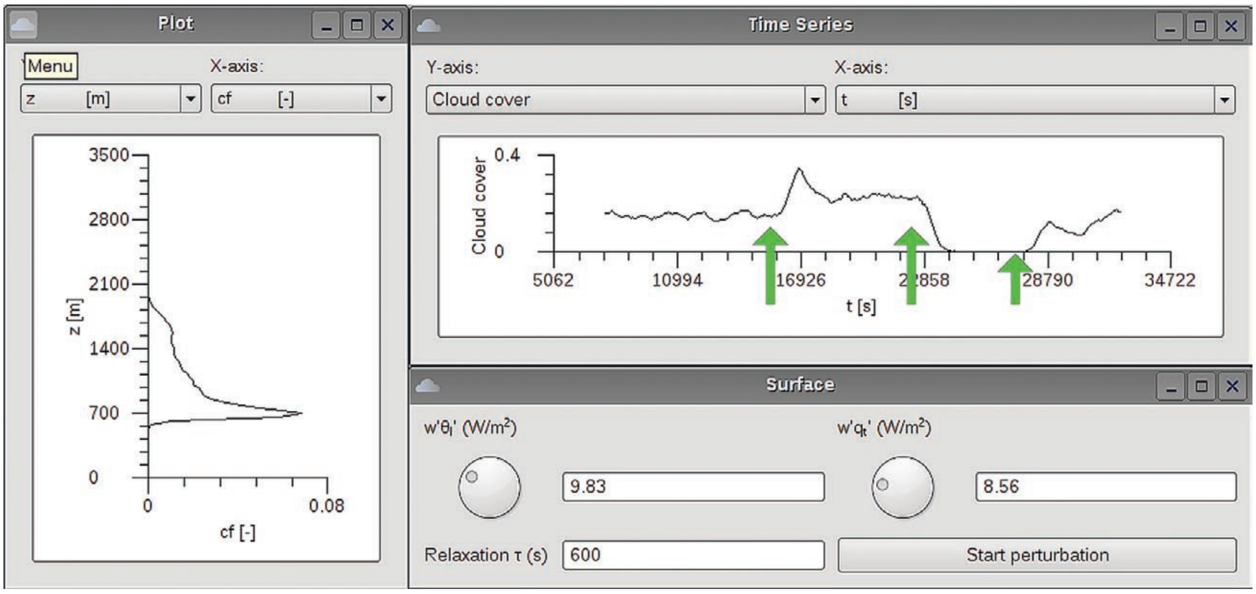


FIG. 5. GALES offers the possibility of direct interaction with the simulation. For example, the surface heat and moisture fluxes can be readily modified during the simulation by turning the "knobs" shown in the lower pane. The resulting effects are directly visible in the shown cloud-fraction profile and time series of cloud cover. The figure shows the response on a user who increased the surface fluxes, subsequently turned them off after a couple of hours, and then restored the fluxes (green arrows).

grid cell. Note that for comparison, the node used by DALES is considered a single computer; otherwise the time would have had to be multiplied by 32. An additional complication to the comparison is that DALES always calculates in double precision, while GALES performs numerically less-sensitive parts of the code in single precision.

Therefore, Fig. 4 should not be interpreted as an exact GPU vs. CPU benchmark; many such benchmarks already exist. Rather, the figure illustrates what size of problems can be handled, and at what speed, by a GPU system. All in all, the speed of our GPU implementation seems more than adequate to allow a dedicated PC to perform simulations up to a 256^3 grid, which would in our case otherwise require 32–64 CPU processor cores. To get a feel for time-to-solution ratios, the numbers in Fig. 4 translate to a speedup (simulated time to wall-clock time) of about 40x at 128^3 (5-s stepping), and 2x at 256^3 (2-s stepping, time steps have to get smaller with higher resolution). In order words, the simulation of 2 h of cloud evolution on a 128^3 grid is performed in about 3 minutes.

NEW AVENUES.

GPU implementation can offer a paradigm shift with regard to how the atmospheric science community performs demanding numerical experiments on (atmospheric) turbulence. Simulations can be started at will and analyzed with ease, even while running. Through the direct visual rendering of the 3D cloud field, 2D slices of relevant model variables, and the supplemental graphs of statistical quantities (mean, variances) as a function of time or height, the user obtains immediate feedback on the progress and quality of the simulation. It is even possible to run and visualize a simulation in a virtual-reality environment. Such visual cues turned out to be essential in a recent cloud life-cycle study by Heus et al. (2009) in which a virtual-reality environment enabled human observers to select clouds that were going through a full life cycle. The virtual-reality environment allowed the researchers to interactively select clouds in a time-evolving LES environment and see the relevant statistics of the selected cloud, thereby allowing one to collect an appropriate cluster of cloud samples. With GALES, these types of highly specific statistics can become available on the fly, without the need for heavy data transfer.

Locally running and visualizing high-performance simulations can have educational benefits. In GALES, we have started to experiment with allowing the user to directly interact with the numerical experiment by changing parameters or boundary conditions of the simulation and study their impact. For example, the user can change the current wind speed, subsidence,

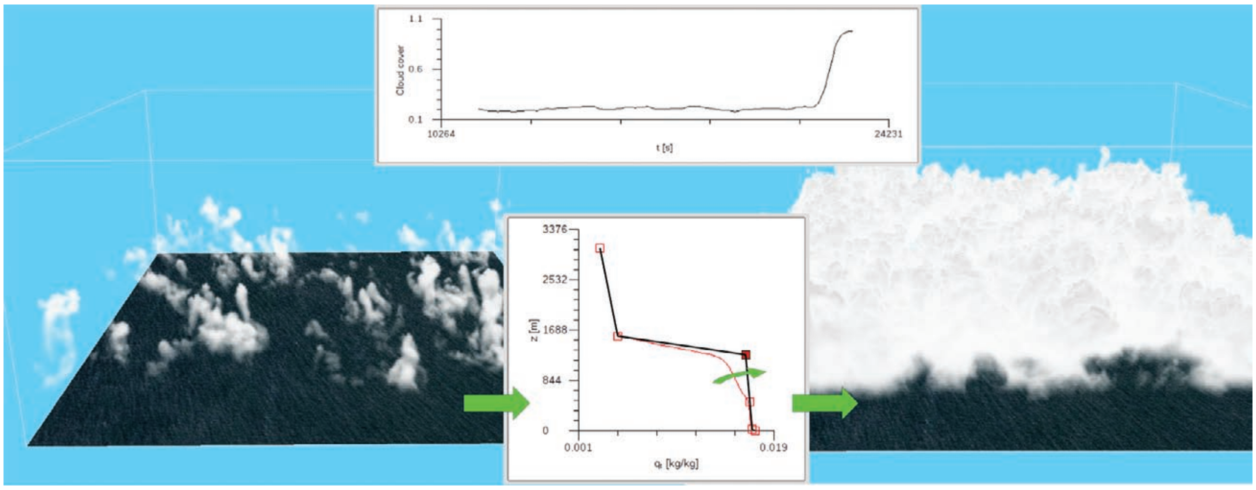


Fig. 6. GALES offers the possibility of direct interaction with the simulation. This figure shows how the user can nudge the current simulation state (red line) toward a user-specified target profile (black line with user-drawn red squares). The nudging can quickly bring the simulation from a cumulus-capped state to a stratocumulus state. A time-series plot of cloud cover, shown above, quantifies the response.

or surface fluxes (Fig. 5) and directly see the consequences. More directly, if less realistically, the user can even "nudge" the mean temperature or humidity profile of the simulation in order to bring the current simulation to a desired (user-specified) profile. Figure 6 illustrates how the user can quickly change the current simulation state from cumulus to stratocumulus using these controls. This could be a great educational tool for new students in boundary layer meteorology. In addition to shallow cumulus situations, stratocumulus or cloud-free situations, such as clear convective boundary layers or neutral or stably stratified boundary layers, can also be studied. The interaction tools allow students to see how changing surface properties or atmospheric conditions can cause clouds to form or break up. Students can acquaint themselves with the different characteristics of turbulent boundary layers with a "learning by doing" approach.

The ability to directly interact with a running numerical experiment may have wider research implications. It could constitute a very interesting research tool for rapidly testing hypotheses or for providing inspiration for new hypotheses. For example, exploring the high-dimensional parameter phase-space around stratocumulus break-up could be sped up by using the direct feedback to more rapidly locate the critical parameter regions.

Also, the ability to directly and continuously interact with the numerical experiment could be a powerful way for models to communicate with each other. For example, a future operational weather model might launch a number of high-resolution "child" (large-eddy) simulations on the GPU for regions dominated by convective boundary layer processes, while continuously feeding these models new boundary conditions as the weather develops. This could provide extremely high-resolution data where they are most needed. A recent step in this direction is described by Neggers et al. (2011, submitted to *BAMS*), where large-eddy simulations as well as single-column models are performed each day on the basis of weather-model data to compare modeling results and investigate the effects of different parameterizations. We are currently investigating the possibility of utilizing GALES in this context.

OUTLOOK. The future of atmospheric GPU computation seems bright. A GPU-resident implementation provides the opportunity to run reasonably high-resolution simulations on a desktop PC. This provides the opportunity to quickly design and start a simulation, directly studying its evolution using the GPU's rendering capabilities. When desired, the user can interactively steer the simulation to further study the phenomenon of interest.

This comes at low cost and energy consumption, making the possibilities of high-performance GPU computing a very attractive and accessible research tool. Furthermore, the possibility of direct visualization and interaction with computationally demanding atmospheric simulations opens new avenues in both education and research.

ACKNOWLEDGMENTS. This work was sponsored by the Netherlands National Computing Facilities Foundation (NCF) for the use of supercomputer facilities, with financial support of the Netherlands Organisation for Scientific Research (NWO). The authors gratefully thank the editor and two anonymous referees for their critical remarks and suggestions to improve the manuscript. The authors also thank Erwin de Beus for his help in realizing GALES.

FOR FURTHER READING

Cohen, J. M., and M. J. Molemaker, 2009: A fast double precision CFD code using CUDA. *Proc. 21st Intl. Conf. on Parallel Computational Fluid Dynamics,* Moffett Field, CA.

Deardorff, J. W., 1970: A three-dimensional numerical investigation of the idealized planetary boundary layer. *Geophys. Astro. Fluid.,* **1,** 377–410.

Heus, T., G. H. van Dijk, H. Jonker, and H. van den Akker, 2008: Mixing in shallow cumulus clouds studied by Lagrangian particle tracking. *J. Atmos. Sci.,* **65**, 2581–2597.

——, H. Jonker, H. van den Akker, E. Griffith, M. Koutek, and F. Post, 2009: A statistical approach to the life cycle analysis of cumulus clouds selected in a virtual reality environment. *J. Geophys. Res.*, **114**, D06208, doi:10.1029/2008JD0109172.

——, and Coauthors, 2010: Formulation of the Dutch Atmospheric Large-Eddy Simulation (DALES) and overview of its applications. *Geosci. Model Dev.,* **3**, 415–414.

Jonker H., T. Heus, and P. P. Sullivan, 2008: A refined view of vertical mass transport by cumulus convection. *Geophys. Res. Lett.,* **35**, L07810, doi:10.1029/2007GL032606.

Lynch, P., 2006: *The Emergence of Numerical Weather Prediction: Richardson's Dream*. Cambridge University Press, 279 pp.

Michalakes, J., and M. Vachharajani, 2008: GPU acceleration of numerical weather prediction. *Paral. Proc. Lett.* **18,** 531–548.

Neggers, R. A. J., A. P. Siebesma, and T. Heus, 2011: Continuous single-column model evaluation at a permanent observation supersite. *Bull. Amer. Meteor. Soc.,* submitted.

Sanders, J. and Kandrot, 2011: *CUDA by Example: An Introduction to General-Purpose GPU Programming.* Addison-Wesley, 312 pp.

Siebesma, A. P., and Coauthors, 2003: A Large Eddy Simulation intercomparison study of shallow cumulus convection. *J. Atmos. Sci.,* **60**, 1201–1219.

Verzijlbergh R., H. Jonker, T. Heus, and J. Vila, 2009: Turbulent dispersion in cloud-topped boundary layers. *Atmos. Chem. Phys.*, **9**, 1289–1302.