Toward High-Resolution Global Atmospheric Inverse Modeling Using Graphics Accelerators

Frédéric Chevallier, Zoé Lloret, Anne Cozic, Sakina Takache, Marine Remaud

ADVANCING EARTH AND SPACE SCIENCE

Geophysical Research Letters®

RESEARCH LETTER
10.1029/2022GL102135

Toward High-Resolution Global Atmospheric Inverse Modeling Using Graphics Accelerators

Frédéric Chevallier[1], Zoé Lloret[1], Anne Cozic[1], Sakina Takache[1], and Marine Remaud[1]

[1]Laboratoire des Sciences du Climat et de l'Environnement, LSCE/IPSL, CEA-CNRS-UVSQ, Université Paris-Saclay, Gif-sur-Yvette, France

Key Points:

- The workload in a Eulerian transport model on a longitude-latitude grid can be embarrassingly parallel enough to run efficiently on a Graphics Processing Unit
- The transport model of the Laboratoire de Météorologie Dynamique was run in a new grid of 1,626,768 3D cells
- The calculation time in the direct version of the model is now less than the time to read the input meteorological data

Correspondence to:

F. Chevallier,
frederic.chevallier@lsce.ipsl.fr

Citation:

Chevallier, F., Lloret, Z., Cozic, A., Takache, S., & Remaud, M. (2023). Toward high-resolution global atmospheric inverse modeling using graphics accelerators. *Geophysical Research Letters*, *50*, e2022GL102135. https://doi.org/10.1029/2022GL102135

Received 17 NOV 2022
Accepted 5 JAN 2023

Author Contributions:

Conceptualization: Zoé Lloret
Methodology: Zoé Lloret, Anne Cozic, Marine Remaud
Software: Zoé Lloret, Anne Cozic, Sakina Takache, Marine Remaud
Writing – original draft: Sakina Takache
Writing – review & editing: Zoé Lloret, Anne Cozic, Sakina Takache, Marine Remaud

Abstract The ability of global transport models to go up in resolution becomes discriminating for greenhouse gas atmospheric inversions. This paper describes the porting on Graphics Processing Units of the global transport model currently used in the European operational Copernicus Atmosphere Monitoring Service (CAMS) for CO_2 and N_2O inversions. It represents an important milestone to achieve sub-degree resolution. The code includes not only the direct model but also its tangent-linear and its adjoint versions which are needed in variational inversions. Tests were carried out for CO_2 at a resolution of 2.50° in longitude, 1.27° in latitude and 79 layers in the vertical, corresponding to 1,626,768 3D cells, 4.5 times more than the current standard resolution of the model used in the CAMS reanalyzes. A month's worth of computation of the tangent-linear and of the adjoint versions now takes 2.5 min, including 50 s for reading meteorological data.

Plain Language Summary Atmospheric transport models are intensively used to infer global greenhouse gas emissions and removals, from atmospheric measurements: a single global analysis involves repeated and long transport simulations. This intensive use has limited the spatial resolution of such analyses despite an increasing need for national greenhouse gas budgets, despite an increasing number of corresponding space-based observations at kilometer resolution, and despite an increasing number of high-quality surface measurements made in sites marked by strong local influences. For the global transport model currently used in the European operational atmosphere monitoring service, our objective within the next 5 years is to make it reach a resolution of about 50 km over the whole globe. This paper describes an important milestone in this direction with the porting of this transport model on hardware components initially developed for video display but now used for high performance computing. Tests were carried out at a resolution of 2.50° in longitude, 1.27° in latitude and 79 layers in the vertical, corresponding to 4.5 times more 3D cells than the current standard resolution of the model. The direct model itself now takes less time than reading the input meteorological data.

1. Introduction

The recent focus on national greenhouse budgets for the preparation of the first Global StockTake of the United Nations Framework Convention on Climate Change (UNFCCC) has renewed the incentive toward higher spatial resolutions of the transport models embedded in atmospheric inversions (Chevallier, 2021; Deng et al., 2022). For instance, despite its two parallelization layers, the transport model in the LSCE inversion system, which derives from Remaud et al. (2018), currently achieves a modest 3.75° in longitude × 1.90° in latitude. This limits the comparison of the inversion results with UNFCCC national inventory reports to large countries or groups of countries (Chevallier, 2021). A specific feature of the LSCE system up to now has been its use in the time-critical environment of the operational Copernicus Atmosphere Monitoring Service (CAMS, https://atmosphere.copernicus.eu/) for carbon dioxide (CO_2) and nitrous oxide (N_2O): the "time to solution" of the LSCE system, and therefore its spatial resolution, have been constrained by the desire to keep its latest products to a maximum of a few months from real time. However, this resolution challenge is shared by the current generation of global transport models used for inverse modeling. In version 10 of the Model Intercomparison Project (MIP) of the second Orbiting Carbon Observatory (OCO-2), six out of the 11 participating models were run at 5° in longitude × 4° in latitude or coarser; the CAMS/LSCE system at its standard 3.75° × 1.90° resolution was another participant; the resolution of three other participants was between 2.5° and 3.0° in longitude × 2.0° in latitude globally (with a regional zoom for one of them) (https://ceos.org/gst/carbon-dioxide.html, accessed 8 January 2023).

The last participant to the OCO-2 MIP (NIES, Maksyutov et al., 2021) was the only one to generate sub-degree resolution results. To achieve this performance, this contributor coupled a global Eulerian model at medium resolution $3.75° \times 3.75°$ with a $0.1° \times 0.1°$ Lagrangian model, which directly parallelizes along a large number of independent air particle trajectories generated for each assimilated observation. Such an "embarrassingly parallel" workload illustrates a major software trend in high-performance computing (HPC) in general, but the ensemble of air parcel trajectories run for each observation mobilizes large Central Processing Unit (CPU) resources. In order to increase the resolution of the transport model in the LSCE inversion system, a more compact and economical approach is favored here which keeps the model within the Eulerian framework using Graphics Processing Units (GPUs).

While semiconductor miniaturization is stalling, accelerators like GPUs are currently sustaining the development of supercomputer performance (e.g., Leiserson et al., 2020). The word "graphics" in GPU refers to the history of this hardware component for video display, but GPUs are now more generally *manycore processors*, that is processors containing thousands of cores which are dedicated to synchronously running the same series of instructions on different input data and on independent parallel execution entities (the *Single-Instruction Multiple Thread* concept). This specialization allows smaller device size and much faster execution times than the general-purpose CPUs for repetitive and computation-intensive tasks. However, the independency requirement is limiting and favors some types of applications, like deep learning. In the case of our subject here atmospheric global transport models, many instructions may be structured in loops over the model 3D cells. However, the latter are not independent of each other because they exchange tracer mass at each model time step: the adaptation to GPU is therefore not straightforward. The specialization of the GPUs also means that they cannot work alone: the different tasks of an entire program must be distributed between CPU and GPU according to their respective capacity, adding another layer of complexity. The increasing investment of HPC centers in hybrid partitions including GPUs forces the adaptation of computer programs that were written for the general-purpose CPU to this custom hardware, at least partially. On the positive side, this development also represents a strategic opportunity to increase the complexity of models like the one in the LSCE inversion system without increasing execution time.

This paper describes the adaptation of the transport model in the LSCE inversion system to the GPU environment at the Centre de Calcul Recherche et Technologie (CCRT, https://www-ccrt.cea.fr/) supercomputing facility. Tests have been made at the resolution of 2.50° in longitude times 1.27° in latitude, with 79 layers in the vertical, hence with 4.5 times more 3D cells than the existing standard configuration. This resolution corresponds to that of its reference general circulation model (GCM) used in the sixth phase of the international Coupled MIP as the atmospheric component of the Institut Pierre Simon Laplace coupled model (Hourdin et al., 2020). The text is structured as follows: the next section describes the transport model; Section 3 describes the algorithm adaptation to GPU; Section 4 presents the validation of the adaptation; conclusions and prospects are drawn in the last section.

2. Model Description

2.1. Variational Framework

In order to meet one of the needs of what was to become the European CAMS operational service, the LSCE developed in 2004 a "variational" global atmospheric inversion system for long-lived trace gases like CO_2, and N_2O (Chevallier et al., 2005; Thompson et al., 2014). Outside CAMS, this same system was used for methane (CH_4) and related molecules (Berchet et al., 2021; Pison et al., 2009). The advantage of this variational system compared to the previous analytical approach developed at LSCE that explicitly used a matrix of derivatives of the whole transport model (Bousquet et al., 2000), lies in its scalability as to the size of the observation vector (i.e., the number of observations to assimilate) and of the control vector (i.e., the number of variables to be optimized). Indeed, with a variational formulation of the Bayesian inversion problem, the calculation of derivatives can be made at the level of the lines of code, using the chain rule, and not at the level of the whole model, thus allowing a considerable gain of computer time and memory.

2.2. Transport Model

Most of the computation time spent by the LSCE global variational system is invested in three configurations of its Eulerian atmospheric transport model: a standard direct version, a tangent-linear version which implements a first-order Taylor expansion of the model, and an adjoint version which propagates sensitivities backward in time in the model. These three versions are based on the GCM of the Laboratoire de Météorologie Dynamique (LMDz, Hourdin et al., 2020) nudged to a Numerical Weather Prediction (NWP) re-analysis and reduced to tracer transport equations through the use of meteorological variables precomputed with the full GCM run at the needed resolution (Hourdin et al., 2006).

The model addresses tracer transport by large-scale advection, deep convection, thermal plumes and boundary layer turbulence with the same equations used in the full GCM for tracer transport and using a series of meteorological variables computed at exactly the same spatial resolution from the full LMDz model: the 3D mass of air, 3D large-scale atmospheric mass fluxes for the advection, vertical turbulent exchange coefficients and temperature for the boundary-layer turbulence, vertical mass fluxes for the thermal scheme, various updraft and downdraft mass fluxes and exchange coefficients to drive the deep convection scheme. The values of these variables are 3-hr averages stored in single precision in NetCDF4 format, while the model uses double precision to obtain accurate gradients for the variational application. An illustration for the need of accurate gradients is given by Figure 2 from Chevallier (2013), with an outlier point (in 1990) which disappears when the gradient is filtered.

2.3. Direct Model

The horizontal grid is regular in both latitudinal and longitudinal degrees, with grid points duplicated for computational convenience at the poles along the longitudes, and at the change-of-date line. The pressures of the vertical levels combine a term that varies with the surface pressure and a fixed term that dominates above the tropopause.

The dynamic part contains the numerical resolution of the fluid mechanics equations on a three-dimensional grid, described by Hourdin and Armengaud (1999). The tracer transport equations are iteratively solved over the longitude-latitude grid following the Van Leer I (1977) scheme. Paired with a slope-limiting algorithm, where slopes are computed by finite differences of the tracer distribution, this scheme preserves tracer monotonicity and does not incur spurious numerical oscillations. However, it is non-linear by construction. While the Van Leer scheme is one-dimensional in space, it can be used over a 3D grid by treating the flux in each of the three dimensions alternatingly. The advance in time is accounted for by integrating the fluxes in each dimension over a half or full time step. Hourdin and Armengaud (1999) use the sequence proposed by Russell and Lerner (1981) which consists of undertaking half a time step in the longitudinal direction, half a time step in the latitudinal direction, then a full time step in the vertical direction, followed again by half a time step in latitude, then half in longitude. This sequence of operations has the advantage of preserving a uniform tracer field regardless of the divergent character of the wind field (Lin & Rood, 1996).

In each direction, the concentration slopes are first calculated, then slope-limiting is applied. The air mass fluxes in the considered direction are then calculated based on the wind field. The mass fluxes and limited slopes are then used to evaluate the tracer concentration reaching the cell boundary in the given direction following the Van Leer I scheme. The boundary value, multiplied by the mass flux, gives the tracer flux traversing the given edge. The tracer amount entering the cell is added to the existing amount in the cell for an updated value in the given direction in time (see, e.g., Figure 1 in Hourdin & Armengaud, 1999). The flux contribution is accumulated in the cell in three dimensions from the three upwind directions.

A special treatment is required in the high latitudes as the cells become thinner with the converging longitudes by nature of the longitude-latitude grid. Given constant time steps δt, the Courant number $u \cdot \delta t/\delta x$—$u$ being the speed of wind in the longitudinal direction x—may exceed 1 due to the diminishing value of δx. This means that the wind field along x traverses multiple cells in one time step. The cell reached at the end of this time step thus has accumulated tracer mass from multiple upwind cells. The tracer fluxes from the traversed cells are summed to form the flux traversed through the edge of the destination cell following Equation 17 of Hourdin and Armengaud (1999) which is non-linear. In practice, this treatment reduces the effective number of grid cells toward the pole.

The representation of unresolved subgrid scales in the model is based on Hourdin et al. (2006). Tracer transport by turbulent diffusion in the planetary boundary layer is represented by vertical exchange coefficients (Equation 6 in Hourdin et al., 2006). It is completed by a thermal model, formulated in terms of mass fluxes, for the case of convective boundary layers (Rio & Hourdin, 2008). Transport by deep convection is derived from the Emanuel (1991) scheme (see, e.g., Pilon et al., 2015, and references therein), which, in practice, links all vertical levels between the cloud base and the level of neutral buoyancy together in a convective system. This link makes the data volume of convective mass fluxes quadratically dependent on the number of vertical levels of the model. However, in practice, most of the global exchange matrix is made of zeros. In order to save disk space and reduce the input data flow, the matrix is stored in compressed sparse row format. The three subgrid-scale parameterizations redistribute tracer mass in the vertical in a strictly linear way.

The code of the transport model is written in Fortran and is run in chunks of 1 month. The inversion system, which is a code in Python language (see Berchet et al., 2021, for the latest version), manages the connection between the months for the three model versions.

2.4. Tangent-Linear and Adjoint Versions

As expected for a Bayesian inversion system, the LSCE system minimizes the usual Bayesian cost function, which represents the balance between observations and prior information in the inversion solution (see, e.g., Equation 9 in Rayner et al., 2019). The LSCE system is "variational" in that it exploits the gradient of the cost function in an iterative way, rather than by an analytical expression or by a Monte Carlo search. The minimizer itself is either a limited memory quasi-Newton method, the M1QN3 software from Gilbert and Lemaréchal (1989) or, systematically for CO_2, a Lanczos version of the conjugate gradient algorithm developed at ECMWF (Fisher, 1998). If the operator that links the variables to estimate (greenhouse gas surface fluxes) and the observations to assimilate (atmospheric concentrations) is linear, the cost function is quadratic and the conjugate gradient converges super-linearly, thereby reducing the number of needed iterations and saving computing time. For this motif, an exact tangent-linear and its transpose (adjoint) were developed for the LMDz off-line transport model.

The tangent-linear is the first-order Taylor expansion of the model. It is used to compute the difference between the model state and the observations in the Bayesian cost function, with the linearization point obtained from the prior value of the control vector. Its transpose, or adjoint, appears in the expression of the gradient of the cost function and is therefore also important for a variational inversion system (see, e.g., Equation 6 of Chevallier et al., 2005).

In practice, the tangent-linear and adjoint codes were derived manually from the reference direct model line by line, in order to directly control the computational efficiency. The adjoint derivation was particularly delicate as it requires the availability of the forward state each time there is a non-linearity in the model: state-dependent conditional instructions coming from the slope limiters or from the special treatment of advection in the high latitudes; updated values of the mass of the atmosphere to multiply with; a profile scaling when assimilating satellite retrievals in order to conserve the column-average mixing ratio while interpolating to the retrieval grid (Chevallier, 2015). This availability of the forward state is not straight-forward because the adjoint runs backward in time. In order to save computer memory, a mixed approach which combines checkpoint files—containing the tracer state every 3 hr—and re-computations in-between was chosen. When entering a 3-hr slot (from its end), the adjoint code reads the tracer state at the start of the slot, runs the forward subroutines for 3 hr forward, while keeping in memory the tracer state at the start of each call to an advection subroutine within the Russell and Lerner (1981) sequence described above (60 times with a 15-min time step for the 3D advection).

2.5. Parallelization Strategies

Since the inception of the LSCE variational system, various opportunities have been taken to increase the speed of the three configurations of this offline model to allow for longer assimilation windows, increased spatial resolutions, and increased model accuracy.

In 2008, the transport model was parallelized with Message Passing Interface (MPI), allowing multi-year inversions (Chevallier et al., 2010). The globe was split into a series of latitude bands that run on different cores. The bands communicate through MPI for the advection along the latitudes. For the two horizontal resolutions discussed here (using respectively 96 and 143 grid cells along the latitudes), the best speed is achieved already at about 10 cores. Every 3 hr, each core reads the meteorological variables of its own latitude band.

In 2012, another layer of parallelization, based on physical considerations, was introduced in the tangent-linear and adjoint versions (Chevallier, 2013). The tangent-linear and the adjoint versions are run in parallel temporal segments which partially overlap. The tracer increments in the tangent-linear and the adjoint sensitivities are carried out from one parallel segment to the next through a global bias term. The simulation that provides the linearization point is only parallelized through MPI and does not use the bias-term simplification. This bias term assumes that all mole fraction increments are uniformly mixed in the global atmosphere. Unwanted side effects of this approximation are damped by a months-long overlap period and by an update of the linearization point during the minimization. The increased speed was successively invested into decadal inversions (Le Quéré et al., 2015), then into increased spatial resolution (2.5 more model 3D cells) and increased complexity of the subgrid parameterizations (Locatelli et al., 2015).

The parallelization with MPI in the LMDz transport model could be extended to longitudes and the vertical axis. However, the distributed-memory approach of MPI does not leverage the shared intra-node memory of current multicore nodes. The parallelization of the LMDz transport model therefore needs to be upgraded for better computational efficiency. A solution which takes a single CPU core but parallelizes across GPU threads using OpenACC directives is described in the next section.

3. Exploiting GPU Hardware

We have chosen to keep the LMDz transport model in Fortran but we have restructured and augmented it with OpenACC directives throughout the code, particularly before each loop. These generic directives guide the compiler in the management of the GPU, specifying each step of the data flow between the memory of the CPU, the memory of the GPU and the GPU threads.

As explained in the introduction, codes running on GPUs primarily modify different locations of the computer memory in parallel threads that execute the same series of instructions. The LMDz code has therefore been restructured in order to expose such situations as much as possible. For many loops, independent parallel threads can be identified across all 3D cells of the global atmosphere, as with initializations. Where a single loop modified the variables from two cells at once, we applied loop fission and split it into two loops (Case 1 in Table 1). Recursive statements exist in the sub-grid-scale parameterizations or for pressure computations (from 3D mass to 2D surface pressure, or from 2D surface pressure to 3D pressures), but only along the vertical axis. These recursive loops were placed inside the loops across the horizontal directions so that they would be executed sequentially within the threads, thus avoiding the latency incurred by a loop on the GPU instructions (Case 2 in Table 1). Similarly, averages are performed in the two polar rows after the advection along the latitudes: the corresponding loop over the polar longitudes was also set as an inner sequential one. Along the longitude axis, the advection can happen over more than one cell in a time step with the formulation mentioned above. This is not an issue for the forward and tangent-linear versions because the initial code was already localized, leaving only a sum over several grid cells (in Equation 17 of Hourdin & Armengaud, 1999). This was already the inner loop, but had simply to be made sequential within each thread. However, in the adjoint version, the process is reversed: the adjoint variable of this sum feeds tracer-mass adjoint variables in up to several neighboring cells at once. We have reformulated this part by temporarily storing the tracer-mass adjoint variables in a matrix. This matrix is of dimension longitudes $\times$ longitudes $\times$ N, with N being the number of cells in the 3D atmosphere, in which the tracer mass is fully advected in less than one time step. Once the matrix is built, the tracer-concentration adjoint variables can be updated by the GPU, with the loop over N performed sequentially on each thread (Case 3 in Table 1).

For the linearization points in the adjoint computations, we noticed that recomputing the 3-hr tracer states is slightly faster than reading them from the disk, so we chose this option. The memory volume needed to save the tens of intermediate states between two consecutive 3-hr states (see Section 2d) would exceed resources so we keep the re-computation in 3-hr chunks there. In total, running the adjoint code therefore implies two full direct simulations, in addition to the adjoint computations themselves.

The transfer time between CPU and GPU is significant in general, so we structured the code in order to avoid back-and-forth transfers: only the input data of the code is pushed from the CPU to the GPU and only the output data comes out of the GPU (Case 4 in Table 1). Since the CPU role is essentially reduced to input/output tasks, we removed all MPI instructions and code instructions related to latitude banding. The code is made much simpler, but we lose the parallel (but memory-intensive) reading of the meteorological variables which was distributed along the cores. The data flow therefore directly uses a single CPU core and one GPU.

4. Evaluation

Our tests were performed on the hybrid partition of the Topaze supercomputer of CCRT. Topaze is a BullSequana XH2000 made by Atos and delivered in 2021. Its hybrid partition is composed of 48 nodes, each of which has four

Table 1
Four Transformations of the Code for an Efficient Use by the GPU

Case	Initial code	New code

1. Loop fission in vlz_ad() – to update q_ad_glo(ij,l-1,iq) and q_ad_glo(ij,l,iq) separately. Subroutine vlz_ad() deals with the adjoint of the vertical advection.	DO l=2,llm
	DO ij=ijb,ije
	IF (dzqw(ij,l) > 0.) THEN
	dzqw_ad(ij,l)= dzqw_ad(ij,l) + adzqw_ad(ij,l)
	adzqw_ad(ij,l) = 0.
	ELSE
	dzqw_ad(ij,l)= dzqw_ad(ij,l) - adzqw_ad(ij,l)
	adzqw_ad(ij,l) = 0.
	ENDIF
	q_ad(ij,l-1) = q_ad(ij,l-1) + dzqw_ad(ij,l)
	q_ad(ij,l) = q_ad(ij,l) - dzqw_ad(ij,l)
	dzqw_ad(ij,l) = 0.
	ENDDO
	ENDDO
		!$acc parallel loop collapse(2)
		DO l=2,llm
		DO ij=1,ip1jmp1
		IF (dzqw(ij,l) > 0.) THEN
		dzqw_ad(ij,l)= dzqw_ad(ij,l) + adzqw_ad(ij,l)
		ELSE
		dzqw_ad(ij,l)= dzqw_ad(ij,l) - adzqw_ad(ij,l)
		ENDIF
		q_ad_glo(ij,l-1,iq) = q_ad_glo(ij,l-1,iq) + dzqw_ad(ij,l)
		ENDDO
		ENDDO
		!$acc parallel loop collapse(2)
		DO l=2,llm
		DO ij=1,ip1jmp1
		q_ad_glo(ij,l,iq) = q_ad_glo(ij,l,iq) - dzqw_ad(ij,l)
		ENDDO
		ENDDO

2. Sequential inner loop in timeloop() – to minimize the number of GPU calls. Subroutine timeloop() manages the time axis.	DO l=1,llm
	DO j=jj_begin,jj_end
	DO i=1,iip1
	smass(i,j)=smass(i,j)+masse(i,j,l)
	ENDDO
	ENDDO
	ENDDO
	DO j=jj_begin,jj_end
	DO i=1,iip1
	ps (i,j)=smass(i,j)/aire(i,j)*g
	END DO
	END DO
		!$acc parallel loop collapse(2)
		DO j=1,jjp1
		DO i=1,iip1
		smass=0.
		!$acc loop seq
		DO l=1,llm
		smass=smass+masse_glo(i,j,l)
		ENDDO
		ps_glo(i,j)=smass/aire_glo(i,j)*g
		END DO
		END DO

3. Temporary transfer matrix in vlx_ad() – to create independent parallel threads for the update of q_ad_glo(ijq,l,iq) and dxq_ad(ijq,l). Subroutine vlx_ad() deals with adjoint of the longitudinal advection.	DO l=llm,1,-1
	IF(nl(l)>0) THEN
	DO iju=1,niju(l)
	ij=indu(iju,l)
	j=(ij-1)/iip1+1
	zu_m=u_m(ij,l)
	IF(zu_m > 0.) THEN
	ijq=ij
	i=ijq-(j-1)*iip1
	DO WHILE(zu_m > masse(ijq,l))
	q_ad(ijq,l) = q_ad(ijq,l) + u_mq_ad(ij,l)*masse(ijq,l)
	zu_m=zu_m-masse(ijq,l)
	i=MOD(i-2+iim,iim)+1
	ijq=(j-1)*iip1+i
	ENDDO
	q_ad(ijq,l) = q_ad(ijq,l) + zu_m * u_mq_ad(ij,l)
	dxq_ad(ijq,l) = dxq_ad(ijq,l) &
	+ zu_m*0.5*(1.-zu_m/masse(ijq,l))*u_mq_ad(ij,l)
	ELSE
	ijq=ij+1
	i=ijq-(j-1)*iip1
	DO WHILE(-zu_m>masse(ijq,l))
	q_ad(ijq,l) = q_ad(ijq,l) – u_mq_ad(ij,l)*masse(ijq,l)
	zu_m=zu_m+masse(ijq,l)
	i=MOD(I,iim)+1
	ijq=(j-1)*iip1+i
	ENDDO
	q_ad(ijq,l) = q_ad(ijq,l) + zu_m * u_mq_ad(ij,l)
	dxq_ad(ijq,l) = dxq_ad(ijq,l) &
	- zu_m*0.5*(1.+zu_m/masse(ijq,l))*u_mq_ad(ij,l)
	ENDIF
	u_mq_ad(ij,l) = 0.
	ENDDO
	ENDIF
	ENDDO
		(…)
		!$acc parallel loop collapse(3)
		DO l=1,llm
		DO j=2,jjp1-1
		DO i1=1,iip1
		IF (indjl(j,l) > 0.) THEN
		ijq=(j-1)*iip1+i1
		!$acc loop seq
		DO i2=1,iip1
		q_ad_glo(ijq,l,iq) = q_ad_glo(ijq,l,iq) + qtmp_ad(i1,i2,indjl(j,l))
		dxq_ad(ijq,l) = dxq_ad(ijq,l) + dxqtmp_ad(i1,i2,indjl(j,l))
		ENDDO
		ENDIF
		ENDDO
		ENDDO
		ENDDO

4. All arrays are on the GPU in vlx() – to minimize transfer times to and from the GPU. Subroutine vlx() deals with the longitudinal advection.	SUBROUTINE vlx_p(q,pente_max,masse,u_m,ijb_x,ije_x) USE Parallel IMPLICIT NONE INCLUDE "dimensions.h" INCLUDE "paramet.h" INCLUDE "logic.h" INCLUDE "comvert.h" INCLUDE "comconst.h" REAL :: masse(ip1jmp1,llm),pente_max REAL :: u_m(ip1jmp1,llm) REAL :: q(ip1jmp1,llm) INTEGER :: ij,l,j,i,iju,ijq,indu(ip1jmp1),niju INTEGER :: n0,iadvplus(ip1jmp1,llm),nl(llm) REAL :: new_m,zu_m,zdum(ip1jmp1,llm) REAL :: dxq(ip1jmp1,llm),dxqu(ip1jmp1) REAL :: zz(ip1jmp1) REAL :: adxqu(ip1jmp1),dxqmax(ip1jmp1,llm) REAL :: u_mq(ip1jmp1,llm) REAL :: SSUM EXTERNAL :: SSUM INTEGER :: ijb,ije,ijb_x,ije_x	SUBROUTINE vlx(iq,pente_max,zzpbar) USE modglob ! contains the arrays already on the GPU IMPLICIT NONE INCLUDE "dimensions.h" INCLUDE "paramet.h" INCLUDE "logic.h" INCLUDE "comvert.h" INCLUDE "comconst.h" REAL :: pente_max REAL :: u_m(ip1jmp1,llm) REAL :: zzpbar, zzw INTEGER :: iq INTEGER :: ij,l,j,i,ijq REAL :: new_m(ip1jmp1,llm),zu_m,zdum(ip1jmp1,llm) REAL :: dxq(ip1jmp1,llm),dxqu(ip1jmp1,llm) REAL :: adxqu(ip1jmp1,llm),dxqmax(ip1jmp1,llm) REAL :: u_mq(ip1jmp1,llm) !$acc data create(u_m,zdum,u_mq,new_m,dxqu,adxqu,dxqmax,dxq)

Note. Some loop indices have been changed because the initial subroutines are called per latitude band whereas the new code, which no longer uses MPI, processes the entire globe at once. MPI, Message Passing Interface; GPU, Graphics Processing Unit.

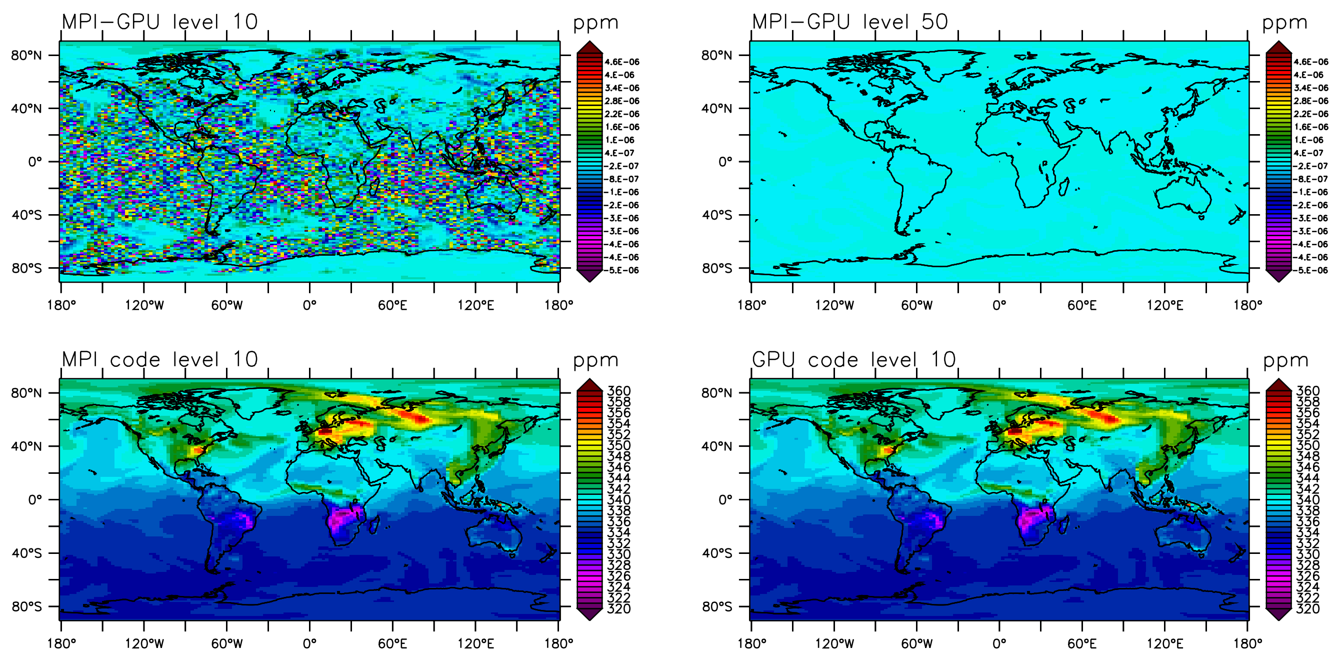

Figure 1. The top row shows differences between the Message Passing Interface-based simulation of CO_2 and the corresponding Graphics Processing Unit-based one at model vertical levels 10 and 50 (around 980 and 7 hPa, respectively). The two simulations at level 10 are shown in the bottom row. The simulations started on 1 January 1979 at 00:00.

NVIDIA A100 GPUs alongside two AMD EPYC Milan 7763 processors. We worked in batch mode on a large "SCRATCH" file system with data transfer rates of about 4 Gb per second per node, by the virtue of Solid-State Drive disks. The compiler was nvfortran version 22.2.

The model was run in a new grid of 2.50° in longitude, 1.27° in latitude and 79 layers in the vertical—that is, 1,626,768 3D cells. The volume of the input meteorological data at that resolution for 1 month is 15 GB, which can be read on a node of Topaze in about 25 s. For tests over a single month, even if the calculations require a single CPU core, it was necessary to reserve four CPU cores from the same node in order to have enough memory. For regular applications over longer times, job submission rules on Topaze require reserving half a processor (32 cores) to access 1 GPU. Note that for the tests, the model was also fed with the location and time of surface measurements so that it simulates them while running.

The direct model was validated against the previous MPI version. Figure 1 shows an MPI-based simulation of CO_2 and the corresponding GPU-based one after 1 month at vertical level 10 of the model (around 980 hPa). It also displays the differences between the two simulations at levels 10 and 50 (around 980 and 7 hPa, respectively): they are marginal, at most of the order of 10^{-6}% close to the surface, as a result of a different data flow in the two codes, in particular in the sub-grid-scale parameterizations.

The GPU version was then modified in order to make the values of the grid-cell surface area consistent throughout the code, which was hardly possible in the previous code structure. The convergence of the tangent-linear version was checked for varying perturbation sizes and the adjoint was validated against the tangent-linear—by developing the expression of the norm of the tangent-linear output in order to make the adjoint appear. An accuracy of 100 times the epsilon of the machine was found.

In terms of pure computing time, the direct model took 15 s for 1 month with the GPU; the tangent-linear version took 30 s and the adjoint 1.3 min. These times are in addition to the 25 s of the mass flux reading in each version. They hardly varied when repeated multiple times. In comparison, the code parallelized with MPI on 15 CPU cores needed 5, 7.5, and 16 min (time to solution) in its direct, tangent-linear and adjoint versions, respectively, reading included. Adding more cores did not make the code run faster. For a variational system that exploits the quadraticity of the cost function with the Lanczos version of the conjugate gradient, one iteration uses one tangent-linear simulation and one adjoint simulation for each month processed: in this case, the gain in speed for the GPU code is eight-fold. Interestingly, the GPU code at this resolution is even 2.6 times faster than the MPI code at the previous resolution of $3.75° \times 1.90° \times 39$ layers.

The new direct model can also be run serially on 1 CPU core by ignoring the OpenACC directives when compiling. In this case, it gives the same results as with the GPU but 10 times slower.

5. Discussion and Conclusions

The growing need for information on regional greenhouse-gas fluxes, for example, within the framework of the Global StockTake of the Paris agreement, the kilometric resolution now achieved by satellite observations of CO_2 and CH_4, the strong local influences at many new greenhouse-gas measurement stations, all constitute a strong incentive toward global transport models in atmospheric inversions at higher resolution than the existing ones. Higher resolutions do not systematically reduce transport model errors and their benefit in inversion systems is damped by prior error correlations, but they allow a finer representation of orography and coastlines and a finer description of emission hotspots, when available. However, for long-lived greenhouse gases like CO_2, N_2O and CH_4, years-long or even decades-long reanalyzes need to be updated frequently as more observations or improved satellite retrievals become available (e.g., Friedlingstein et al., 2022).

For the LMDz transport model currently used in the CAMS operational service for CO_2 and N_2O inversions, our goal is to dramatically increase the rate of resolution upgrades over time, with a target resolution of around 1° over the whole globe at the end of 2023 and around 50 km in less than 5 years for routine inversions without lengthening production time. This ambition is supported by the corresponding development of HPC which is moving toward exascale capability, but must be accompanied by a parallelization strategy that fits the corresponding structure of the hardware resources. We showed here that the workload in a Eulerian transport model on a three-dimensional longitude-latitude grid can be made “embarrassingly parallel” enough to run efficiently on a GPU, without any simplification. This efficiency is evidenced by the fact that the calculation time itself in the direct version of the model is now less than the time to read the input meteorological data. The calculation time of the tangent-linear version is comparable to the reading time for the meteorological data, while the adjoint is more than twice as slow, penalized by the management of the linearization points. The code performance not only optimizes the GPU resources, but also leaves room to run the Monte Carlo ensembles of inversions that are critical for the computation of uncertainty in variational systems (Chevallier et al., 2007).

Reading the meteorological data on the node is now the main computation bottleneck. The fact that the transport calculation is taken by the GPU leaves little activity on the CPU: the reading and writing tasks, and the management of the GPU. The code is therefore most suitable for computer partitions with a small ratio of CPU cores to GPU device per node and with large node memory, as is the case for deep learning applications. In comparison, the Topaze machine we tested on is more hybrid and requires reserving tens of cores to access a single GPU.

The next development step for the LMDz transport model is the application to higher-resolution horizontal grids. We expect the speed of the new code to be proportional to the number of atmospheric columns, but note that the time step must be reduced with the grid refinement for the advection along latitudes. In the course of these resolution upgrades, our strategy is to closely follow the scientific and technical developments in the full LMDz GCM, which is part of the Earth system model of Institut Pierre-Simon-Laplace. In particular, the advection scheme of the LMDz GCM is now moving away from regular longitude-latitude grids, in order to avoid their computationally-expensive polar singularity (Dubos et al., 2015). The transport model described here is already evolving accordingly in order to be able to exploit the meteorological variables generated on the new grid, currently made of hexagons. Technically, its developments in the coming years will follow the evolution of Fortran-compiler support for GPU parallelism and will be guided by the evolution of HPC resources toward exascale computing, particularly in France.

Data Availability Statement

This version of the LMDz global transport model, v3.1, is publicly available from https://doi.org/10.5281/zenodo.7324039 (Chevallier, 2022) under the Creative Commons Attribution 4.0 International licence.

Acknowledgments
This work was granted access to the HPC resources of CCRT under the allocation CEA/DRF, and of TGCC under the allocation A0110102201 made by GENCI. It was funded by the Copernicus Atmosphere Monitoring Service, implemented by ECMWF on behalf of the European Commission (Grant: CAMS2 55). The authors would like to thank Yann Meurdesoif (LSCE) for his long-standing support with the model developments and his advice about GPUs in particular.

References

Berchet, A., Sollum, E., Thompson, R. L., Pison, I., Thanwerdas, J., Broquet, G., et al. (2021). The community inversion framework v1.0: A unified system for atmospheric inversion studies. *Geoscientific Model Development*, *14*(8), 5331–5354. https://doi.org/10.5194/gmd-14-5331-2021

Bousquet, P., Peylin, P., Ciais, P., Quere, C., Friedlingstein, P., & Tans, P. (2000). Regional changes in carbon dioxide fluxes of land and oceans since 1980. *Science*, *290*(5495), 1342–1346. https://doi.org/10.1126/science.290.5495.1342

Chevallier, F. (2013). On the parallelization of atmospheric inversions of CO_2 surface fluxes within a variational framework. *Geoscientific Model Development*, *6*(3), 783–790. https://doi.org/10.5194/gmd-6-783-2013

Chevallier, F. (2015). On the statistical optimality of CO_2 atmospheric inversions assimilating CO_2 column retrievals. *Atmospheric Chemistry and Physics*, *15*(19), 11133–11145. https://doi.org/10.5194/acp-15-11133-2015

Chevallier, F. (2021). Fluxes of carbon dioxide from managed ecosystems estimated by national inventories compared to atmospheric inverse modeling. *Geophysical Research Letters*, *48*(15), e2021GL093565. https://doi.org/10.1029/2021GL093565

Chevallier, F. (2022). LMDz transport model (3.1). *Zenodo*. https://doi.org/10.5281/zenodo.7324039

Chevallier, F., Bréon, F.-M., & Rayner, P. J. (2007). The contribution of the Orbiting Carbon Observatory to the estimation of CO_2 sources and sinks: Theoretical study in a variational data assimilation framework. *Journal of Geophysical Research*, *112*(D9), D09307. https://doi.org/10.1029/2006JD007375

Chevallier, F., Ciais, P., Conway, T. J., Aalto, T., Anderson, B. E., Bousquet, P., et al. (2010). CO_2 surface fluxes at grid point scale estimated from a global 21-year reanalysis of atmospheric measurements. *Journal of Geophysical Research*, *115*(D21), D21307. https://doi.org/10.1029/2010JD013887

Chevallier, F., Fisher, M., Peylin, P., Serrar, S., Bousquet, P., Bréon, F.-M., et al. (2005). Inferring CO_2 sources and sinks from satellite observations: Method and application to TOVS data. *Journal of Geophysical Research*, *110*(D24), D24309. https://doi.org/10.1029/2005JD006390

Deng, Z., Ciais, P., Tzompa-Sosa, Z. A., Saunois, M., Qiu, C., Tan, C., et al. (2022). Comparing national greenhouse gas budgets reported in UNFCCC inventories against atmospheric inversions. *Earth System Science Data*, *14*(4), 1639–1675. https://doi.org/10.5194/essd-14-1639-2022

Dubos, T., Dubey, S., Tort, M., Mittal, R., Meurdesoif, Y., & Hourdin, F. (2015). DYNAMICO-1.0, an icosahedral hydrostatic dynamical core designed for consistency and versatility. *Geoscientific Model Development*, *8*(10), 3131–3150. https://doi.org/10.5194/gmd-8-3131-2015

Emanuel, K. A. (1991). A scheme for representing cumulus convection in large-scale models. *Journal of the Atmospheric Sciences*, *48*(21), 2313–2329. https://doi.org/10.1175/1520-0469(1991)048<2313:ASFRCC>2.0.CO;2

Fisher, M. (1998). Minimization algorithms for variational data assimilation. In *Proceedings of seminar on recent developments in numerical methods for atmospheric modelling. 7–11 September 1998* (pp. 364–385). ECMWF. Retrieved from https://www.ecmwf.int/node/9400

Friedlingstein, P., Jones, M. W., O'Sullivan, M., Andrew, R. M., Bakker, D. C. E., Hauck, J., et al. (2022). Global carbon budget 2021. *Earth System Science Data*, *14*(4), 1917–2005. https://doi.org/10.5194/essd-14-1917-2022

Gilbert, J.-C., & Lemaréchal, C. (1989). Some numerical experiments with variable-storage quasi-Newton algorithms. *Mathematical Programming*, *45*(1–3), 407–435. https://doi.org/10.1007/bf01589113

Hourdin, F., & Armengaud, A. (1999). The use of finite-volume methods for atmospheric advection of trace species. Part I: Test of various formulations in a general circulation model. *Monthly Weather Review*, *127*(5), 822–837. https://doi.org/10.1175/1520-0493(1999)127<0822:tuofvm>2.0.co;2

Hourdin, F., Rio, C., Grandpeix, J.-Y., Madeleine, J.-B., Cheruy, F., Rochetin, N., et al. (2020). LMDZ6A: The atmospheric component of the IPSL climate model with improved and better tuned physics. *Journal of Advances in Modeling Earth Systems*, *12*(7), e2019MS001892. https://doi.org/10.1029/2019MS001892

Hourdin, F., Talagrand, O., & Idelkadi, A. (2006). Eulerian backtracking of atmospheric tracers. II: Numerical aspects. *Quarterly Journal of the Royal Meteorological Society*, *132*(615), 585–603. https://doi.org/10.1256/qj.03.198.B

Leiserson, C. E., Emer, J. S., Kuszmaul, B. C., Lampson, B. W., Sanchez, D., & Schardl, T. B. (2020). There's plenty of room at the top: What will drive computer performance after Moore's law? *Science*, *368*, 6495. https://doi.org/10.1126/science.aam9744

Le Quéré, C., Moriarty, R., Andrew, R. M., Peters, G. P., Ciais, P., Friedlingstein, P., et al. (2015). Global carbon budget 2014. *Earth System Science Data*, *7*(1), 47–85. https://doi.org/10.5194/essd-7-47-2015

Lin, S., & Rood, R. B. (1996). Multidimensional flux-form semi-Lagrangian transport schemes. *Monthly Weather Review*, *124*(9), 2046–2070. https://doi.org/10.1175/1520-0493(1996)124<2046:mffslt>2.0.co;2

Locatelli, R., Bousquet, P., Hourdin, F., Saunois, M., Cozic, A., Couvreux, F., et al. (2015). Atmospheric transport and chemistry of trace gases in LMDz5B: Evaluation and implications for inverse modelling. *Geoscientific Model Development*, *8*(2), 129–150. https://doi.org/10.5194/gmd-8-129-2015

Maksyutov, S., Oda, T., Saito, M., Janardanan, R., Belikov, D., Kaiser, J. W., et al. (2021). Technical note: A high-resolution inverse modelling technique for estimating surface CO_2 fluxes based on the NIES-TM–FLEXPART coupled transport model and its adjoint. *Atmospheric Chemistry and Physics*, *21*(2), 1245–1266. https://doi.org/10.5194/acp-21-1245-2021

Pilon, R., Grandpeix, J.-Y., & Heinrich, P. (2015). Representation of transport and scavenging of trace particles in the Emanuel moist convection scheme. *Quarterly Journal of the Royal Meteorological Society*, *141*(689), 1244–1258. https://doi.org/10.1002/qj.2431

Pison, I., Bousquet, P., Chevallier, F., Szopa, S., & Hauglustaine, D. A. (2009). Multi-species inversion of CH_4, CO, and H_2 emissions from surface measurements. *Atmospheric Chemistry and Physics*, *9*(14), 5281–5297. https://doi.org/10.5194/acp-9-5281-2009

Rayner, P. J., Michalak, A. M., & Chevallier, F. (2019). Fundamentals of data assimilation applied to biogeochemistry. *Atmospheric Chemistry and Physics*, *19*(22), 13911–13932. https://doi.org/10.5194/acp-19-13911-2019

Remaud, M., Chevallier, F., Cozic, A., Lin, X., & Bousquet, P. (2018). On the impact of recent developments of an atmospheric general circulation model on the simulation of CO_2 transport. *Geoscientific Model Development*, *11*, 4489–4513. https://doi.org/10.5194/gmd-11-4489-2018

Rio, C., & Hourdin, F. (2008). A thermal plume model for the convective boundary layer: Representation of cumulus clouds. *Journal of the Atmospheric Sciences*, *65*(2), 407–425. https://doi.org/10.1175/2007jas2256.1

Russell, G. L., & Lerner, J. A. (1981). A new finite-differencing scheme for the tracer transport equation. *Journal of Applied Meteorology*, *20*(12), 1483–1498. https://doi.org/10.1175/1520-0450(1981)020<1483:anfdsf>2.0.co;2

Thompson, R. L., Chevallier, F., Crotwell, A. M., Dutton, G., Langenfelds, R. L., Prinn, R. G., et al. (2014). Nitrous oxide emissions 1999–2009 from a global atmospheric inversion. *Atmospheric Chemistry and Physics*, *14*(4), 1801–1817. https://doi.org/10.5194/acp-14-1801-2014

Van Leer, B. (1977). Towards the ultimate conservative difference scheme. Part IV: A new approach to numerical convection. *Journal of Computational Physics*, *23*(3), 276–299. https://doi.org/10.1016/0021-9991(77)90095-x