Response to reviewers for the revised manuscript:

"CaMa-Flood-GPU: A GPU-based hydrodynamic model implementation for scalable global simulations"

Manuscript ID: egusphere-2025-6500

We thank the reviewers for their careful reading of the manuscript and for their constructive comments and suggestions. In response, we have substantially revised the manuscript and the response documents.

The revised version presents the study more clearly and consistently, with improved organization, clearer explanations, and a more focused presentation of the main contributions. We have also refined the manuscript text, figures, captions, and response documents so that the relationship between the reviewers' comments and the corresponding revisions is easier to follow.

Overall, the revision strengthens the clarity of the manuscript, improves the transparency of the methodology and evaluation, and addresses the issues raised during review. We believe that the revised manuscript has been significantly improved and that the reviewers' concerns have been addressed, and we hope that it will now be suitable for publication in Geoscientific Model Development.

On behalf of all authors,

Sincerely,

Jiabo Yin and Dai Yamazaki

Reply to Reviewer 1's comments

Manuscript: CaMa-Flood-GPU: A GPU-based hydrodynamic model implementation for scalable global simulations (egusphere-2025-6500)

Legend

- Reviewers' comments
- Authors' responses
- Unchanged context from the manuscript
- New or changed text in the revised manuscript

This manuscript presents and evaluate a new implementation of the CaMa-Flood global river model. With this new implementation, the original model, written in Fortran and ran on a CPU multi-core architecture, is rewritten for a GPU-based architecture with adapted libraries and kernels with the objective to speedup global scale high resolution simulations without degrading performances. As a first step, the authors carefully analyze the main challenges behind this transposition, including the irregularity of the network topology, the interpolation of runoff inputs, the non linear relationship between water depth and river storage, and the handling of memory and communications between GPUs. Methods adapted to massive parallelism are proposed at each step. The new model, called CaMa-Flood-GPU, is then compared to the original CPU-based CaMa-Flood in terms of computation time and reproducibility. Results show a significant gain in computation time (more than 3 times quicker for a simulation at 1 arcmin resolution) with negligible differences in the outputs (river discharge and depth, flood outlow). The manuscript is well written and organized, figures are of good quality although some could be improved (see comments bellow). I have a few remarks that could further improve the manuscript, remarks that should be easily handled.

To reviewer: We thank the reviewer for the positive assessment of the manuscript and for the detailed comments. In response, we have made five main groups of revisions. First, we expanded the domain-decomposition description to explain how unit-catchments are grouped into basin groups formed by merging primitive river-mouth basins through cross-basin bifurcation links, and then assigned to GPU ranks without splitting such a group across GPUs. Second, we clarified that backwater coupling is local to each GPU rank and that the only runtime inter-GPU communications are the three collectives now named explicitly. Third, we revised the forcing, dataloader, block-size and memory-footprint descriptions so that the performance comparison is reproducible and the OOM entries in Table 3 are explained. Fourth, we redrew Fig. 5, revised Fig. 6, and changed Fig. 6 to display river-mouth locations as a primary map layer so the selected drainage units are visually explicit. Finally, we simplified the numerical-stability presentation by retaining the bifurcation-enabled comparison, adding relative differences, adding 0.1 ° ERA5-Land runoff over the unified 1980-2014 period, and broadening the discharge comparison across rivers of different scales. We appreciate that these comments helped us make the implementation choices and evaluation design much clearer. The specific manuscript changes and their locations are identified in the itemized responses below.

Comment 1. Ordering catchments and assigning them to dedicated GPUs is particularly important for efficient parallelism in terms of memory and communications, but it is not clear how this first

step is elaborated. More detailed could be provided, for example in section 2.1.1. This could also include how catchments are assigned to one GPU or another in a multi-GPU configuration (L172).

To reviewer: Thank you for raising this important point. We agree that the submitted manuscript did not explain the first decomposition step in enough detail. The implementation does not simply divide the global array into equal contiguous chunks. Instead, the preprocessing first traces each unit-catchment to its river mouth to obtain primitive river-mouth basins. It then uses the cross-basin bifurcation links to merge primitive basins that are hydrologically connected by bifurcating channels, producing larger basin groups. Only after this grouping step are the unit-catchments topologically ordered and packed into the global state vector.

This distinction is important for the multi-GPU design. Load balancing is applied to whole bifurcation-formed basin groups rather than to individual unit-catchments, using a longest-processing-time-first (LPT) greedy rule. As a result, a GPU subdomain is the union of several complete basin groups, and no such group is split across ranks. This preserves all main-channel and bifurcation connections inside a rank while still giving a practical load-balance strategy. The detailed paragraph below gives the grouping definition, the state-vector ordering and the LPT rank-assignment rule. The manuscript change is shown below for Section 2.1.4, replacing the current communication-overhead paragraph that only described domain decomposition in general terms.

Revised manuscript text:

A key challenge in scaling hydrodynamic models to multiple GPUs is managing communication and data handling overhead. For multi-GPU runs, we decompose the reordered river network by basin group rather than by individual unit-catchment. Let C be the set of all unit-catchments and let B_m be the primitive basin draining to river mouth m. Cross-basin bifurcation links define an undirected graph among these primitive basins; each connected component of this graph is treated as one basin group G_k. The global state vector is then ordered as $\mathbf{x} = [\mathbf{x}_{G_1}, \mathbf{x}_{G_2}, \ldots, \mathbf{x}_{G_K}]$, where basin groups are sorted by decreasing size and the entries within each G_k follow the upstream-to-downstream topological order of the main river network. For P GPU ranks, basin groups are assigned by a longest-processing-time-first greedy rule: processing G_k in decreasing $|G_k|$, we assign it to the rank with the smallest current load. The subdomain owned by rank p is therefore the union of all basin groups assigned to that rank. Because no bifurcation-formed basin group is split across ranks, all main-channel and bifurcation links used by the local inertial update connect unit-catchments inside the same rank. The layout keeps state arrays contiguous, balances the number of local unit-catchments among GPUs, and removes the need for peer-to-peer halo exchange during time stepping. With this multi-GPU decomposition in place, the remaining scalability bottlenecks no longer come from neighboring subdomains exchanging halo data, but from the global data movement and synchronization that still couple the ranks.

Comment 2. How are communications between neighbor catchments handled to account for backwater effects (impact of downstream water level on the surface profile and flow dynamics)? In other terms, are there some tricks with the arrangement of catchments into the memory to limit communication time (see also previous comment)?

To reviewer: Thank you. Backwater effects in CaMa-Flood-GPU enter through the local inertial update, where each unit-catchment uses the downstream water state along the river-network connection to compute the water-surface gradient and discharge. This is a network-neighbour dependency,

but it is not handled through a peer-to-peer exchange at run time. The reason is the decomposition described in Main Remark 1: all unit-catchments connected by retained main-channel and cross-basin bifurcation links remain inside the same bifurcation-formed basin group, and each such group is owned by a single GPU rank.

In implementation terms, the downstream index lookup, the bifurcation receiver lookup and the atomic accumulation path therefore operate on local state arrays after the preprocessing and rank assignment have been completed. The only data that cross GPU ranks during time stepping are the global collectives needed for forcing distribution, adaptive time-step synchronization and diagnostic reduction. We added the paragraph below to make this local-memory interpretation explicit before listing the collectives. The manuscript change is shown below for Section 2.1.4, immediately before the paragraph that enumerates the inter-GPU collectives.

Revised manuscript text:

For output, we offload file writing to separate threads, preventing the main simulation loop from stalling. By construction of the basin-group decomposition described above, every pair of unit-catchments coupled by the local inertial step, whether through the main network or through a bifurcation link, lies inside a single bifurcation-formed basin group, and every such group is owned by a single GPU rank. Reading the downstream water state therefore reduces to a contiguous local memory access, and no peer-to-peer halo exchange between GPUs is required at run time. The only inter-GPU traffic per step is the three collective operations enumerated below.

Comment 3. Can floods represented in 2D introduce water exchanges between neighbor catchments that are not directly connected through the river network? What would be the implications for memory exchanges?

To reviewer: Thank you. This is an important distinction between CaMa-Flood's catchment-based macro-scale floodplain (CMF) formulation and a genuine two-dimensional floodplain solver. The main CaMa-Flood approximation is the "uniform water surface" assumption within each unit-catchment: floodplain storage and inundated area are diagnosed locally from total storage and the precomputed sub-grid topographic profile. This local diagnosis changes water depth and flooded fraction inside the unit-catchment, but it does not by itself solve arbitrary lateral 2-D exchange between neighbouring floodplain areas.

At the same time, CaMa-Flood does include a bifurcation scheme to represent channel bifurcations and overbank-flow paths that are outside the main downstream river-network map. These pathways are treated as additional predefined routing links rather than as a 2-D floodplain stencil. In CaMa-Flood-GPU, they are therefore included in the same routing graph and basin-group decomposition as the main river links, so they do not introduce a separate halo-exchange pattern. If the model were extended to a true 2-D floodplain dynamic, with water exchanged across adjacent floodplain cells or catchment boundaries not represented by those links, then the GPU decomposition would need a stencil- or halo-style exchange between neighbouring ranks. The manuscript change is shown below for Section 2.1.3, appended to the paragraph that describes local floodplain-depth diagnosis from storage.

Revised manuscript text:

This diagnostic step is an embarrassingly parallel problem, because the depth calculation for each catchment is entirely self-contained and does not depend on any other. Because the inundation diagnosis is based on the uniform-water-surface assumption within each unit-catchment, it adds no inter-GPU communication. Water exchange outside the main downstream river path is represented only where the CaMa-Flood bifurcation and overbank-flow scheme defines additional routing links; those links are treated as explicit edges of the routing graph and are included in the basin-group decomposition.

Comment 4. Could you briefly describe what the `scatter_add` and `atomic_add` operations do?

To reviewer: Thank you for the suggestion. We have added a brief functional description that frames the two operations by what they achieve in the model rather than by their argument lists. The manuscript change is shown below for Section 2.1.1, at the end of the paragraph introducing the `scatter_add` and `atomic_add` operations.

Revised manuscript text:

Fortunately, this challenge is a well-studied problem in computational science, allowing us to reframe the algorithm by adapting established parallel computing solutions. To solve this, we leverage the highly-parallelized and extensively-optimized `scatter_add` operation available in the Triton language. In brief, a `scatter_add` distributes outflows from many source catchments into their downstream destinations in a single parallel pass; whenever several sources write to the same destination, as at confluences and bifurcation receivers, `atomic_add` serializes those individual increments at the hardware level, so the accumulated inflow is order-independent and mass-conserving without explicit locks.

Comment 5. It is not clear how the global scale state array is constructed (see major comment 1).

To reviewer: Thank you. The global state array is built in two preprocessing stages. First, every unit-catchment is traced to its river mouth to form primitive river-mouth basins; cross-basin bifurcation links then merge primitive basins that exchange water through bifurcating channels into larger basin groups G_k. Second, the global state vector is laid out as $\mathbf{x} = [\mathbf{x}_{G_1}, \mathbf{x}_{G_2}, \ldots, \mathbf{x}_{G_K}]$, with basin groups sorted by decreasing size and entries within each G_k ordered upstream-to-downstream along the main river network. For multi-GPU runs, each rank's subdomain is the union of basin groups assigned to it by the longest-processing-time-first rule; because no group is split across ranks, every main-channel and bifurcation link used by the local inertial update connects unit-catchments inside the same rank, so state arrays remain contiguous in GPU memory. The full revised paragraph is given under Comment 1 above (Section 2.1.4, Communication and overhead) and is therefore not repeated here.

Comment 6. By "land grid cells", do you mean "grid cells from the Land Surface Model that produces runoff"?

To reviewer: Yes, thank you for asking. By "land grid cells" we meant the runoff-generating cells delivered by an upstream land-surface model. The manuscript change is shown below for Section 2.1.2, in the runoff-aggregation paragraph that defines the runoff vector.

Revised manuscript text:

Another challenge arising from the irregular network topology is the mismatch between gridded external forcing and our catchment units (Figure 2b). Most runoff products are delivered on latitude-longitude grids that do not align with catchment boundaries. At each time step, the forcing reader supplies runoff from an upstream land-surface model (LSM) to the dataset object. If using a global grid, N_r is the number of runoff-generating land grid cells, i.e., the output cells of an upstream LSM. The resulting runoff vector is then passed to `shard_forcing`, which maps the runoff-generating cells to the local GPU catchments.

Comment 7. I guess the `shard_forcing` interface could easily integrate a specific method to couple CaMa-Flood with a Land Surface Model, right? It might be worth mentioning it.

To reviewer: Yes, thank you. This is exactly the design intent. We have added a sentence clarifying that `shard_forcing` is the hand-off point for online or offline coupling with a land-surface model. The manuscript change is shown below for Section 2.1.2, at the end of the paragraph introducing the `UserDefinedDataset` and `shard_forcing` interface.

Revised manuscript text:

Beyond this optimized implementation, we provide a flexible interface for runoff inputs to handle various data sources. By abstracting the data loading and mapping procedure in a general `shard_forcing` interface, we allow for considerable customization. A user can create a new dataset subclass with a custom `shard_forcing` method that defines how to broadcast and map their particular data source onto the catchments. In particular, `shard_forcing` is intentionally designed as the coupling hand-off point for online or offline coupling with an LSM: an LSM can pass its per-time-step runoff tensor, in PyTorch or any compatible array, directly to `shard_forcing`, which performs the grid-to-catchment aggregation on-device without going through the disk. The core model remains unaware of these preprocessing details—as long as the dataset object supplies catchment runoff values R_c for each time step, the model will route them.

Comment 8. The figure is not clear and could be improved. For instance, what does the columns (3 in batched runoff, fluxes and errors) represent? Catchments? And the lines? Computation (sub-)time steps? Where is the synchronization between GPUs, does it allow to advance to the next time step? What are dataloader 0 and 1?

To reviewer: Thank you for pointing this out. We redrew Fig. 5 as a wall-clock timeline with separate rows for CPU loader processes and GPU ranks, so that the per-step tasks, the three inter-GPU collectives and the adaptive sub-steps are now labelled directly in the artwork. The figure caption and surrounding paragraph are unchanged. The redrawn figure is shown below.

Revised manuscript text:

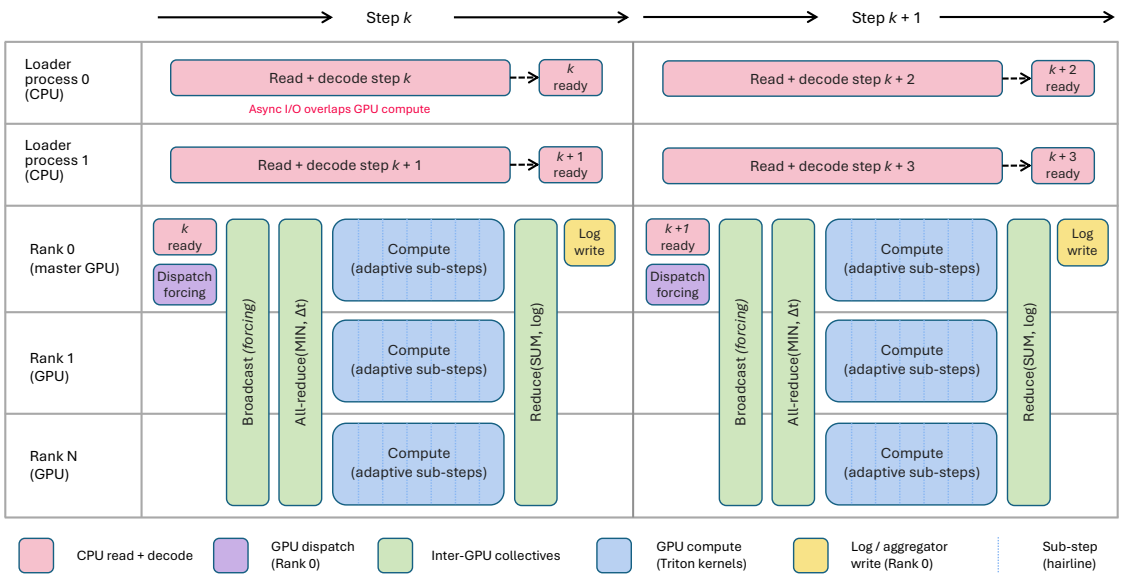

Figure 5. Runtime workflow for asynchronous input preparation, multi-GPU computation, and logging over two consecutive time steps.

To address these overheads, we implemented a suite of asynchronous and optimized data handling strategies (Figure 5). For input data, we use a multi-process data loader. While the GPUs are computing the current step, background worker processes are already pre-fetching and preparing the data for the next step, placing it into a memory buffer. This ensures that when the GPUs are ready for new input, the data is already in memory and can be quickly broadcast, effectively hiding the I/O latency.

Comment 9. Isn't the input broadcast also a collective communication? This would give three collective communications at each time step.

To reviewer: You are correct. The input broadcast is also a collective communication. We have updated the count from two to three and now name the three collectives explicitly in the revised text. The manuscript change is shown below for Section 2.1.4, replacing the sentence that previously counted only two collectives per time step.

Revised manuscript text:

Only at the end of a full time step is a single, collective communication performed to aggregate the results for logging. This minimizes cross-GPU traffic, ensuring that communication occurs only when necessary. As a result, each time step involves three collective operations: the runoff broadcast, the all-rank minimum reduction used to select the global adaptive sub-step, and the end-of-step gather/reduction of diagnostic output. The runoff broadcast carries only N_r floats per step, amounting to on the order of tens of megabytes at the resolutions tested, and is dominated by inter-GPU bandwidth, so its cost stays a small fraction of the per-step compute even at multi-GPU scale.

Comment 10. How is the flexible time step implemented/parallelized? I understand that the same sub-step is chosen for all the catchments of the globe, is that right? Since each GPU works asynchronously, could it be possible to choose a different sub-step for each GPU?

To reviewer: Thank you for the question. We use one globally synchronized sub-step so that the GPU trajectory remains comparable to the CPU baseline regardless of GPU partitioning. The revised manuscript text below adds this point to the adaptive time-step discussion. The manuscript change is shown below for Section 2.1.4, in the paragraph explaining adaptive time-step synchronization.

Revised manuscript text:

For logging and synchronization, we employ a local accumulation and end-of-step reduction strategy. Instead of performing a global reduction at every sub-step, each GPU accumulates diagnostic variables in a local buffer. All GPUs therefore share the same global sub-step, taken as the minimum required Δt across the whole domain, so that the GPU integration trajectory remains identical to the reference CPU run regardless of the number of GPUs.

Comment 11. The figure could be improved and enlarged: gauge dots are not clearly visible except with a very high zoom, star symbols are not visible at all. Also, in the figure caption, it is written that the catchment outlines are shown; I understand that they are represented by the shaded colors, but in the text, the term catchment corresponds to base unit while in the figure it is more likely the entire basin. Is that right? Finally, why some catchments/basins are so large, encompassing several basins (like orange in South America, pink in Asia, green in Africa or brown in North America)?

To reviewer: Thank you for these comments. We agree that the previous Fig. 6 description could be misread. In the revision, we changed the figure concept to directly show retained-basin river-mouth locations together with cross-basin bifurcation links. This makes the selected drainage units and their correspondence to the point-of-interest filtering step easier to interpret at normal zoom. We also clarify that GPU-rank assignment is described separately in the domain-decomposition section. The revised Fig. 6 is shown below.

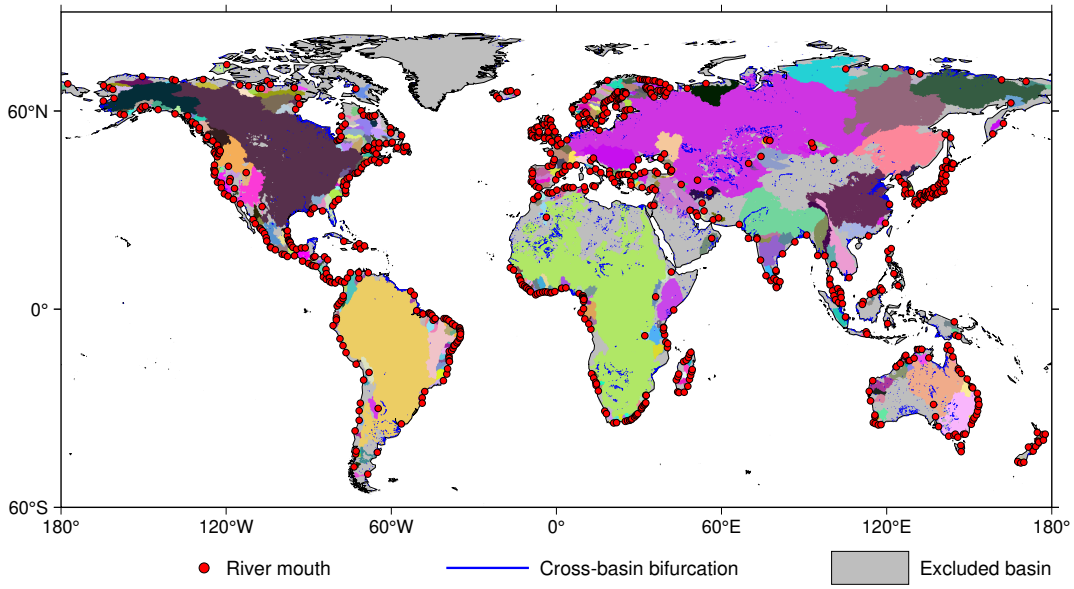

Figure 6. Example of a customized simulation domain constructed from bifurcation-formed basins after the point-of-interest filtering and domain-decomposition step. Each retained basin contains at least one point of interest. Coloured patches denote the retained basins, while grey areas mark basins excluded from the selected domain; markers indicate river-mouth locations and blue lines denote cross-basin bifurcation links.

Comment 12. It seems from Table 3 that the CPU configuration was not used for the first three machines (4070 Ti, V100 and A100). Why then fill in the CPU and CPU Cores columns for these machines? Also, would it be possible to add the available memory for each machine and node?

To reviewer: Thank you for this useful suggestion. The CPU and CPU-core columns are retained for the GPU machines because these CPUs are the host processors for the GPU runs: they launch the GPU program, handle CPU-side preprocessing and I/O, coordinate the data loader, and provide host memory around GPU execution. The CPU-only benchmark uses the separate CPU row, but the host CPU configuration is still needed to reproduce the GPU cases in Table 3. We therefore kept the CPU information for all machines and added CPU memory per node to Table 1.

Revised manuscript text:

Table 1. Summary of computing nodes and their hardware configurations.

Label	CPU	CPU Memory (per node)	GPU	CPU Cores (per node)	GPUs (per node)
4070 Ti	Intel Core i7-13700	64 GB	NVIDIA GeForce RTX 4070 Ti (12GB)	16	1
V100	Dual Intel Xeon E5-2640 v4	128 GB	NVIDIA Tesla V100 (16GB)	20	4
A100	Dual AMD EPYC 7543	256 GB	NVIDIA A100 (40GB)	64	4
CPU	Dual Intel Xeon 6248R	256 GB	—	48	—

Comment 13. I understand the idea of using a coarse resolution forcing (1° runoff) to focus more on computation performances. But running a global scale simulation at very high resolution (e.g. 1 arcmin, that is typically not achievable with the current CPU version) would also require high resolution forcing. Maybe an additional experiment would help quantify the added simulation time due to the reading and broadcasting of high resolution forcing.

To reviewer: Thank you for raising this. We did not run a forcing-resolution sweep for the speed benchmark, and we would like to explain why we believe the current setup already isolates the comparison properly. CaMa-Flood-GPU uses a multi-process asynchronous dataloader that reads and decodes the next time step's forcing while the GPU is integrating the current step. In our GPU tests, using NetCDF forcing instead of CaMa-Flood's native binary forcing produced no measurable change in wall-clock time, because the file-format cost was hidden behind the GPU computation. The CPU version, however, reads forcing synchronously, so NetCDF decoding competes with the routing calculation for CPU resources and increases the CPU wall time. The deliberate choice of the 1 ° binary sample forcing distributed with CaMa-Flood therefore keeps the wall-time comparison in Table 3 focused on routing-side performance rather than file-format overhead.

At the same time, we agree that the revised manuscript should demonstrate that CaMa-Flood-GPU can ingest and route higher-resolution gridded forcing. We therefore added a separate numerical-stability experiment based on a daily forcing series prepared from 0.1 ° ERA5-Land runoff over 1980-2014. This experiment is not used to claim a speed impact of high-resolution forcing; instead, it verifies the high-resolution NetCDF input path, forcing-to-unit-catchment mapping and routing kernels under identical CPU and GPU parameter states. We mention this distinction in the revised setup text so the reader can separate the routing-focused speed benchmark from the high-resolution-forcing stability test.

A second consideration is where the resolution mismatch is actually paid for. Interpolating high-resolution runoff onto CaMa-Flood-GPU's irregular unit-catchment network is a preprocessing step rather than a per-time-step cost, and Python's geospatial library ecosystem makes implementing such regridders straightforward. In CaMa-Flood-GPU, the forcing wrapper classes offer optional pre-interpolation and in-memory caching. When this option is enabled, regridding is performed once and reused across runs, whether the source is a structured high-resolution grid or another unstructured mesh. We recommend this opt-in pattern for high-throughput use cases such as parameter calibration and sensitivity analysis, where hundreds to thousands of replicate runs may share the same forcing. As a result, the wall-time impact of high-resolution forcing on the simulation itself is bounded by the asynchronous dataloader pipeline rather than by the regridding cost. The revised Figure 5 further illustrates this loading mechanism, and the expanded numerical-stability experiment exercises this path with long-period high-resolution forcing.

Comment 14. Could you explain what the block size is? Is this related to the catchment assignment (see major remark 1)?

To reviewer: Thank you for raising this. We now define the Triton block size as a kernel granularity and distinguish it from the basin-group assignment described above under Main Remark 1. The manuscript change is shown below for the new experimental-setup paragraph at the start of Section 3.1.

Revised manuscript text:

To make the subsequent performance and numerical-stability analyses reproducible, we first summarize the common experimental setup and then distinguish the choices specific to the wall-clock benchmark and to the stability comparison. All performance benchmarks use CaMa-Flood global parameter sets derived from MERIT Hydro (Yamazaki et al., 2019) at four spatial resolutions, 15-, 6-, 3- and 1-arcmin; the corresponding numbers of unit-catchments, bifurcation links and approximate adaptive sub-steps are listed in Table 2. The same model options are enabled in both implementations: adaptive sub-step integration, bifurcation routing and on-the-fly logging of the selected diagnostic output. The hardware configurations used in the benchmarks are summarized in Table 1.

For the speed benchmark, the runoff forcing is the daily 1 ° binary sample runoff distributed with the CPU CaMa-Flood release; the sample forcing includes year 2000 and was prepared from the output of the Ensemble Land State Estimator (ELSE) (Kim et al., 2009). To limit non-computational bottlenecks, we also control the I/O path so that the wall-clock comparison reflects routing performance rather than file-format overhead. The GPU implementation overlaps forcing input with computation through its asynchronous dataloader, whereas the CPU reference reads forcing synchronously. Each run therefore saves only one key output variable, and the CPU reference writes that output in binary format. The CPU reference is CaMa-Flood v4.23 compiled with Intel Fortran using the flags "-O3 -fp-model precise" and run in hybrid MPI/OpenMP mode; we tested several MPI/OpenMP combinations and used the fastest configuration for each CPU benchmark, with 16 MPI ranks following the official CaMa-Flood recommendation on the multi-core hosts. GPU runs use a Triton block size of 128, which specifies the number of unit-catchments processed by one kernel instance. All reported wall-times are means over five repeated runs. Table 3 summarizes the benchmark timing results for a 1-year simulation period at the four resolutions on various hardware configurations: a personal workstation with NVIDIA RTX 4070 Ti, the V100 server with 1–4 GPUs, the A100 server with 1–4 GPUs, and the CPU-only server (four CPU nodes) with different numbers of CPU cores.

For the numerical-stability comparison, we use daily forcing series prepared from 0.1 ° ERA5-Land NetCDF runoff (Muñoz-Sabater et al., 2021) and from the eartH2Observe (E2O) Tier-1 ensemble runoff (Schellekens et al., 2017). The 1980-2014 period is adopted to remain consistent with the temporal coverage of the E2O Tier-1 dataset. For each numerical-stability comparison, CPU and GPU runs use the same 6-arcmin parameter set, identical forcing, initial states and adaptive sub-step settings, with the bifurcation module enabled in both runs.

Comment 15. The amount of memory is also a very important aspect in global scale and high resolution simulations. Could you explain why some configurations encountered lack of memory problems and not others?

To reviewer: Thank you for raising this. We added a short explanation of the 1-arcmin GPU memory footprint and how it relates to the OOM entries in Table 3. The manuscript change is shown below for Section 3.1, appended to the paragraph discussing the 1-arcmin performance results.

Revised manuscript text:

The gap widens at finer resolutions: at 3-arcmin, the CPU assumes 2 h 45 m 19 s, compared to 11 m 47 s on 4×V100 and 7 m 21 s on 4×A100; at 1-arcmin, the CPU needs 140 h 58 m 20 s, versus 6 h 51 m 36 s on 4×V100 and 3 h 51 m 24 s on 4×A100. At the 1-arcmin resolution, the global

state requires about 30 GB of GPU memory in total. This footprint is partitioned across ranks by the LPT assignment of bifurcation-formed basin groups, so the per-GPU peak roughly halves with each doubling of the rank count. Configurations whose per-card GPU memory in Table 1 is below the resulting per-GPU peak are marked OOM in Table 3.

Comment 16. In the third column, it could be preferable to show the relative difference. In that case, values bellow 10^{-6} could be attributed to numerical errors only (floating-point precision).

To reviewer: Thank you for the suggestion. We accepted the suggestion and redrew the retained numerical-stability figure so the third column reports relative differences. We note, however, that the floating-point noise floor in this comparison is somewhat above 10^{-6}: each CaMa-Flood time step is composed of many adaptive sub-steps, and within each sub-step the routing kernel evaluates several physical processes (flow inertia, friction, floodplain exchange and, when enabled, bifurcation accumulation) in different orders on CPU and GPU. The accumulated round-off across sub-steps and across these operations therefore exceeds the per-operation floating-point precision and is reflected in the relative-difference panel.

Comment 17. What is the period of the simulation?

To reviewer: Thank you for the question. The benchmark period is calendar year 2000 (365 days, 2000-01-01 to 2000-12-31). Year 2000 was selected because it is included in the sample runoff forcing distributed with the CPU CaMa-Flood release, which makes the benchmark straightforward to reproduce. This is already stated in the new experimental-setup paragraph in Section 3.1 shown above.

Comment 18. What is the added value of showing both simulations, with and without the activation of the bifurcation module?

To reviewer: Thank you for the question. We took this comment as an opportunity to substantially expand the numerical-stability section rather than to keep two parallel figures. In the revision, we retain the bifurcation-enabled comparison as the primary case because it includes the base routing operations and additionally activates the bifurcation-specific atomic accumulation path, so CPU-GPU agreement in this configuration also validates the simpler no-bifurcation path. We then expand the retained analysis by adding 0.1 ° ERA5-Land runoff and eartH2Observe Tier-1 ensemble runoff comparisons over the unified 1980-2014 period, and by broadening the station-level discharge comparison to rivers of different scales to test whether CPU-GPU agreement holds beyond the largest channels. The manuscript change is shown below for Section 3.2.

Revised manuscript text:

The numerical-stability comparison follows the setup described in Section 3.1 and covers 1980–2014 with two daily runoff forcing series: E2O Tier-1 ensemble runoff at 0.25 ° and a daily series prepared from ERA5-Land surface runoff at 0.1 °. Using these two forcing datasets, we compare CaMa-Flood-GPU with the reference CPU implementation through spatial mean fields and station hydrographs, and assess whether floating-point differences remain bounded over the multi-decadal simulation.

Figure 8 presents the CPU–GPU comparison under E2O runoff. The CPU and GPU mean river outflow fields (Figure 8a, b) are visually indistinguishable, and the relative-difference map in Figure 8c is dominated by the gray "below-noise-floor" band of $\pm 0.001\%$, with non-zero values appearing mainly on the largest river stems. Figure 8d–f repeats the comparison for floodplain outflow and Figure 8g–i for river depth, both with the same noise-floor behaviour and the same concentration of residual signal on the main channels of large basins. Across the three variables, the field-mean relative difference remains below $10^{-3}\%$. Figure 9 reproduces the same comparison under ERA5-Land runoff. The residual magnitude and spatial pattern remain essentially unchanged between Figures 8 and 9, although the two forcing products differ in resolution and temporal variability. This consistency indicates that the residuals arise from the floating-point ordering of the routing calculation rather than from the runoff forcing itself.

Figures 10 and 11 compare simulated discharge for the final four years of the full simulation at six GRDC gauge locations spanning four orders of magnitude in drainage area. At every gauge location, the CPU and GPU curves overlay each other almost exactly, and the right-hand red axis shows the day-by-day CPU–GPU difference at a magnified scale. The residual amplitude follows the depth of the upstream reduction at the station location rather than the absolute discharge magnitude. On the large main stems, GPU `atomic_add` operations are associative and commutative but their execution order is not deterministic, whereas the CPU reduction follows a deterministic order. The two reductions therefore return slightly different floating-point bit patterns, producing a small bounded difference at the m^3/s level, three to four orders of magnitude smaller than the simulated discharge. At the mid-scale tributary and headwater gauge locations, the reduction is shallower and the red difference curve collapses toward zero for most of the displayed period. Figure 11 reproduces the same behaviour under ERA5-Land runoff, and neither forcing shows a monotonic component in the residual over the multi-year window. We therefore do not see evidence of numerical drift accumulating through the integration.

Mass conservation on the irregular unit-catchment graph is preserved by construction. The GPU implementation applies the same source-to-target routing fluxes as the CPU algorithm: main-channel fluxes are accumulated through `scatter_add` reductions, and bifurcation fluxes are accumulated through `atomic_add` reductions rather than overwriting destination storage. Each outgoing flux is therefore added to the corresponding downstream storage term, with the CPU–GPU difference limited to the floating-point summation order discussed above. The bifurcation-enabled configuration exercises both the base routing operations and the bifurcation-specific `atomic_add` path. The no-bifurcation configuration is a strict subset of these operations, so agreement with bifurcation enabled also supports agreement for the simpler routing case. Taken together, the field comparisons, station hydrographs and routing-flux accounting show that CaMa-Flood-GPU reproduces the CPU reference without detectable numerical drift over multi-decadal integrations, across both forcing products and with bifurcations enabled.

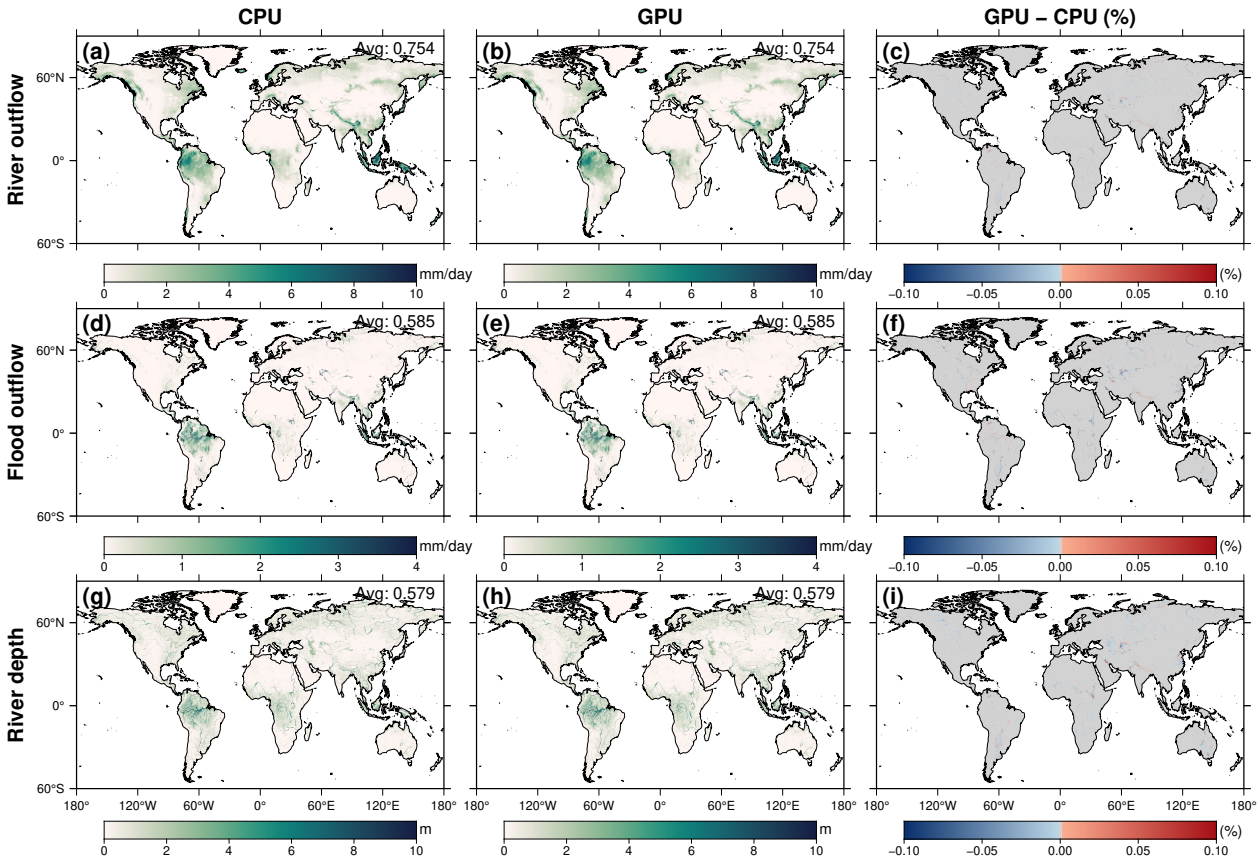

Figure 8. CPU vs. GPU mean fields with the bifurcation module enabled, under E2O Tier-1 ensemble-mean runoff at 0.25 ° for 1980–2014. **(a–c)** Mean river outflow for CPU, GPU and their relative difference. **(d–f)** Mean floodplain outflow for CPU, GPU and their relative difference. **(g–i)** Mean river depth for CPU, GPU and their relative difference. The third-column colour bar is bipolar with a flat gray band at ±0.001% to mark the floating-point noise floor.

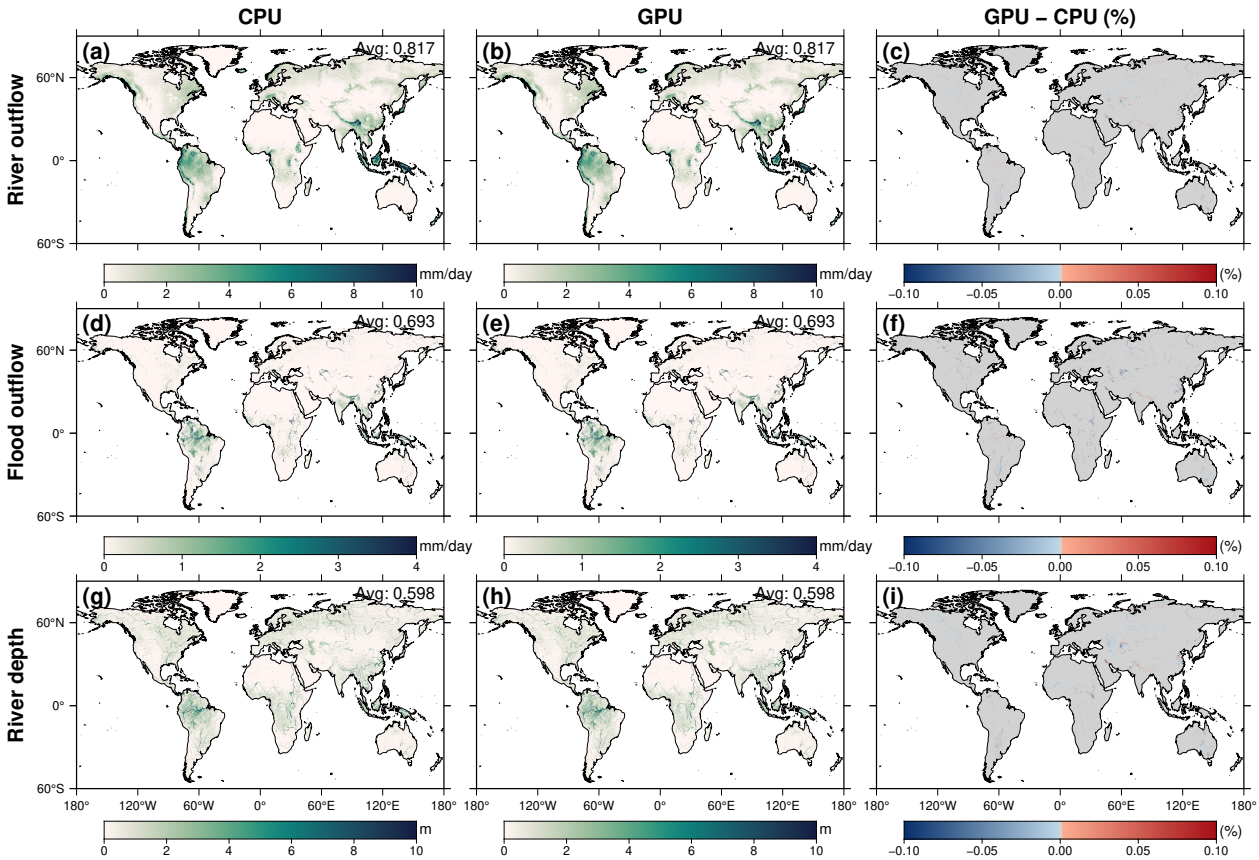

Figure 9. Same as Figure 8, but with ERA5-Land surface runoff at 0.1 °.

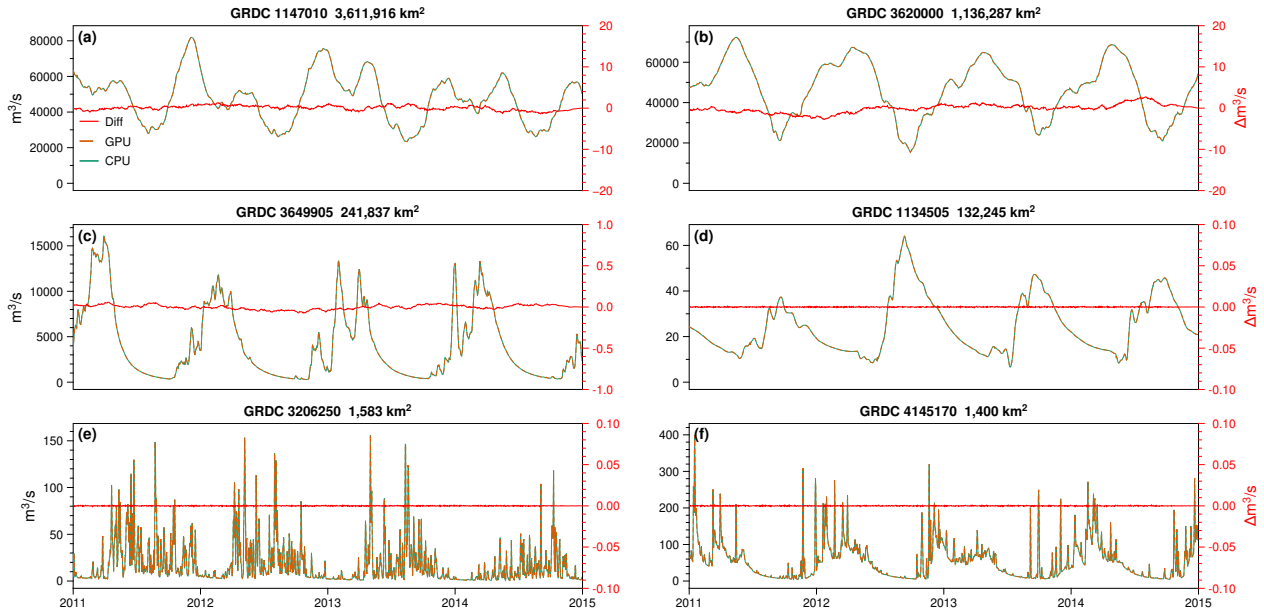

Figure 10. Simulated daily river discharge at the locations of six GRDC gauges spanning four orders of magnitude in drainage area, for the last four years of the 1980–2014 simulation under E2O Tier-1 runoff at 0.25 °. The gauge panels are ordered by drainage area from large to small. CPU (green solid) and GPU (orange dashed) curves overlap; the difference is plotted as a red solid line with values given on the right-hand axis.

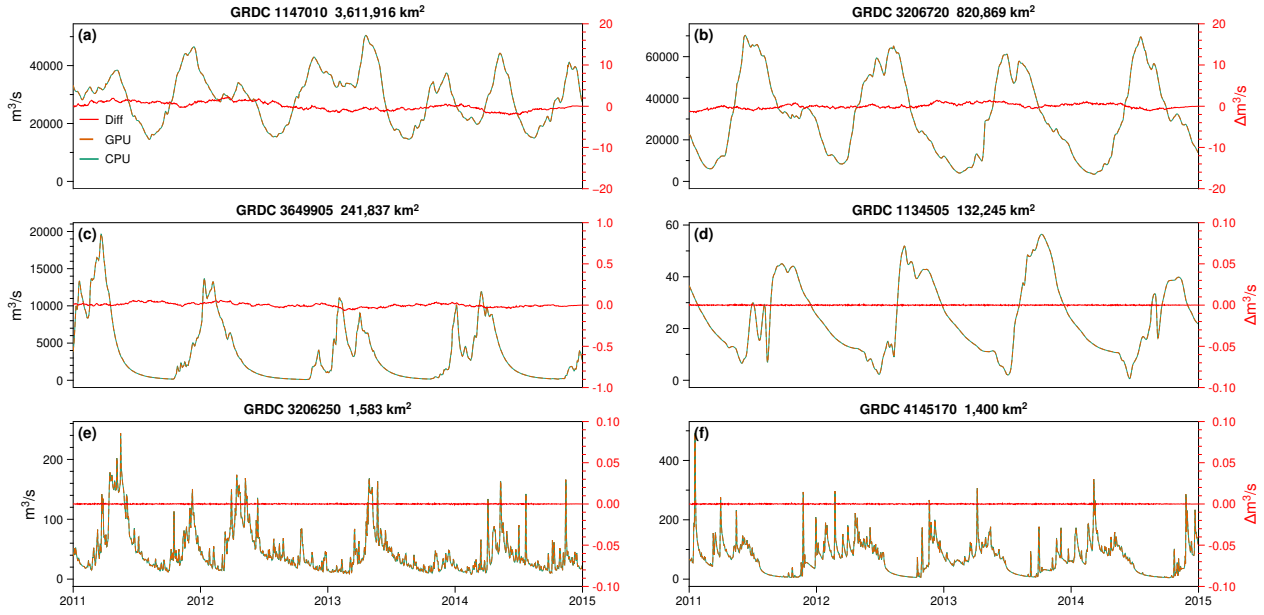

Figure 11. Same as Figure 10, but with ERA5-Land surface runoff at 0.1 °.

References

- Muñoz-Sabater, J., Dutra, E., Agusti-Panareda, A., Albergel, C., Arduini, G., et al. (2021). ERA5-Land: A state-of-the-art global reanalysis dataset for land applications. Earth System

Science Data, 13, 4349-4383. https://doi.org/10.5194/essd-13-4349-2021

- Kim, H., Yeh, P. J.-F., Oki, T., and Kanae, S. (2009). Role of rivers in the seasonal variations of terrestrial water storage over global basins. Geophysical Research Letters, 36, L17402. https://doi.org/10.1029/2009GL039006
- Schellekens, J., Dutra, E., Martínez-de la Torre, A., Balsamo, G., van Dijk, A., Sperna Weiland, F., Minvielle, M., Calvet, J.-C., Decharme, B., Eisner, S., Fink, G., Flörke, M., Peßenteiner, S., van Beek, R., Polcher, J., Beck, H., Orth, R., Calton, B., Burke, S., Dorigo, W., and Weedon, G. P. (2017). A global water resources ensemble of hydrological models: the eartH2Observe Tier-1 dataset. Earth System Science Data, 9, 389-413. https://doi.org/10.5194/essd-9-389-2017
- Yamazaki, D., Ikeshima, D., Sosa, J., Bates, P. D., Allen, G. H., and Pavelsky, T. M. (2019). MERIT Hydro: A high-resolution global hydrography map based on latest topography dataset. Water Resources Research, 55, 5053-5073. https://doi.org/10.1029/2019WR024873

Reply to Reviewer 2's comments

Manuscript: CaMa-Flood-GPU: A GPU-based hydrodynamic model implementation for scalable global simulations (egusphere-2025-6500)

Legend

- Reviewers' comments
- Authors' responses
- Unchanged context from the manuscript
- New or changed text in the revised manuscript

I enjoyed reading manuscript egusphere-2025-6500, which describes the GPU implementation of CaMa-Flood, a popular river routing model typically used in large scale studies—often in concomitance with global hydrologic models. Overall the manuscript is well organized and written, although I believe there are opportunities for improving both quality of the presentation and experimental setup.

To reviewer: We thank the reviewer for the careful reading and constructive suggestions. We have revised the manuscript along the main directions identified in the review. The Introduction now gives more background for both GPU hydrodynamic modelling and CaMa-Flood itself, including a clearer explanation of local inertial routing, sub-grid inundation and CaMa-Flood's adoption as a routing layer. Section 3.1 now starts with an explicit experimental-setup paragraph, separating the 1 ° binary forcing used for speed benchmarks from the 0.1 ° ERA5-Land NetCDF runoff used for numerical stability. Section 3.2 has been expanded by retaining the bifurcation-enabled stability comparison, adding the unified 1980-2014 comparison period, and adding discharge comparisons across rivers of different scales. Finally, the Conclusions now discuss how the modular sub-module interface can support reservoir operation, sediment transport and differentiable global routing. We appreciate that the reviewer's suggestions helped us improve both the context and the experimental transparency of the manuscript. The specific manuscript changes and their locations are identified in the itemized responses below.

Comment 1. Beginning with the Introduction, I think it would be important to provide more context on the implementation of hydrodynamic models in GPU—something that is now limited to just a few lines. In other words, what is the state-of-the-art in the field?

To reviewer: Thank you for the suggestion. We agree that the submitted Introduction underrepresented the state of the art in GPU hydrodynamic modelling and therefore did not clearly show where CaMa-Flood-GPU fits. In the revision, we expanded the discussion from a short citation list into a structured paragraph. The new text distinguishes GPU implementations of full 2-D shallow-water models, GPU ports of simplified local-inertial or kinematic flood models, and GPU acceleration in integrated hydrology or routing workflows.

This broader framing also helps define the specific gap addressed by this work. Many existing GPU hydrodynamic models target regular or quasi-regular meshes and are often applied at urban, regional

or continental scales. CaMa-Flood-GPU instead targets the global CaMa-Flood unit-catchment network, where the graph is irregular, bifurcations must be represented and the calculation must run across multiple GPUs. We therefore revised the Introduction to make the contribution less isolated and more clearly connected to the state of the art. The manuscript change is shown below for the Introduction, replacing the short GPU-hydrodynamic-model context paragraph.

Revised manuscript text:

GPUs have become a cornerstone of scientific computing because of their massive parallelism, and they have been successfully applied to hydrodynamic models in recent years. GPU acceleration is increasingly being adopted across Earth-system model components, from atmospheric chemistry kernels in coupled climate-chemistry models (Alvanos & Christoudias, 2017) to flood hydrodynamics. In the latter, GPU-based modelling has matured along three complementary directions. (i) For the full 2-D shallow-water equations, multi-GPU and single-GPU codes deliver order-of-magnitude speed-ups for high-resolution flood-inundation studies on regular meshes (Morales-Hernández et al., 2021). (ii) For simplified local inertial or kinematic formulations, GPU ports of established CPU floodplain models extend GPU acceleration from urban catchments to continental flood mapping while retaining sub-grid topographic detail (Sharifian et al., 2023; Rong et al., 2024; Caviedes-Voullieme et al., 2023). (iii) For integrated land-surface and routing, GPU acceleration of hydrology models (Hokkanen et al., 2021) and GPU-resident machine-learning surrogates of routing (Zahura et al., 2020) demonstrate that the throughput unlocked by GPUs makes ensemble and global-domain experiments tractable. Most of the above target single-GPU execution on regular or quasi-regular meshes at sub-continental scale. Global river-routing on irregular unit-catchment networks, the regime in which CaMa-Flood operates and which is mandatory for properly representing bifurcations, deltas and floodplain storage on a global mesh, has so far not been ported to GPU. CaMa-Flood-GPU is, to our knowledge, the first multi-GPU implementation of a global, irregular, bifurcation-aware routing model.

Comment 2. A second point I suggest strengthening is the background information on CaMa-Flood; I found it hard to follow the first part of the Introduction, as it assumes the reader is familiar with the model.

To reviewer: Thank you. We agree that the submitted Introduction was too compressed for readers unfamiliar with CaMa-Flood. Because many of the GPU implementation choices only make sense after the reader understands the CaMa-Flood formulation, we inserted a background paragraph before the GPU motivation. The added paragraph explains the CMF discretisation into irregular unit-catchments, the storage partition between channel and floodplain, and the diagnosis of water depth and inundation from sub-grid topography.

We also use this paragraph to connect the physical model to the computational challenges discussed later. The local inertial momentum equation explains why downstream water level and backwater effects matter, while the bifurcation extension explains why a simple single-downstream tree is not sufficient. This gives readers the context needed to understand the subsequent discussion of irregular topology, `scatter_add`/`atomic_add` accumulation and basin-group domain decomposition. The manuscript change is shown below for the Introduction, inserted before the paragraph that motivates GPU implementation.

Revised manuscript text:

CaMa-Flood employs an explicit time integration of this equation, where the new outflow is computed based on the water level difference ($h_c - h_d$) between adjacent catchments. CaMa-Flood is an instance of the catchment-based macro-scale floodplain (CMF) approach (Yamazaki et al., 2011; Yamazaki, 2025), which discretises every river basin into irregular unit-catchments of roughly 5-50 km extracted from MERIT Hydro (Yamazaki et al., 2019) rather than from a regular grid. Within each unit-catchment, water mass is partitioned into a channel storage and a floodplain storage, exchanged according to the channel bank-full geometry. Two simplifying assumptions are imposed: no significant topographic depressions inside a unit-catchment, and a spatially uniform water surface across its floodplain. Under these, sub-grid topographic profiles, precomputed as the cumulative distribution of high-resolution DEM elevations, diagnose water depth and inundation extent directly from total storage, so no 2-D shallow-water solve is needed between neighbouring unit-catchments. Channel routing along the river network follows the local inertial momentum equation (Bates et al., 2010; de Almeida et al., 2012), with the outlet pixel of each unit-catchment supplying an absolute reference elevation so that water-surface gradients, and therefore backwater and flow reversal, can be represented even between coarse-resolution units. Recent extensions remove the single downstream flow path constraint and represent channel bifurcations as additional divergent flows driven by the same local inertial equation (Yamazaki et al., 2014; Mateo et al., 2017).

Comment 3. The "Performance comparison" (Section 3.1) seems strong. In my opinion, it should be complemented by a section / sub-section on the experimental setup, where the authors explain how the runoff data were generated and was CaMa-Flood setup.

To reviewer: Thank you for the suggestion. We agree that the performance results were difficult to interpret without a compact experimental setup. In the revision, a new experimental-setup paragraph at the start of Section 3.1 specifies the river-network resolutions, benchmark period, active model options, CPU and GPU configurations, repetition count and forcing sources before the wall-clock results. This makes the comparison reproducible and separates model-performance choices from later numerical-stability choices.

We also clarified the forcing distinction, because it is important for interpreting Table 3. The speed benchmark uses the 1 ° binary sample runoff distributed with the CPU CaMa-Flood release, which keeps the wall-time comparison focused on routing and GPU execution rather than forcing-format overhead. The numerical-stability comparison adds a 0.1 ° ERA5-Land NetCDF runoff forcing case, so that the GPU and CPU outputs are compared under the same high-resolution forcing and parameter states in that separate analysis. The manuscript change is shown below as a new experimental-setup paragraph at the start of Section 3.1.

Revised manuscript text:

To make the subsequent performance and numerical-stability analyses reproducible, we first summarize the common experimental setup and then distinguish the choices specific to the wall-clock benchmark and to the stability comparison. All performance benchmarks use CaMa-Flood global parameter sets derived from MERIT Hydro (Yamazaki et al., 2019) at four spatial resolutions, 15-, 6-, 3- and 1-arcmin; the corresponding numbers of unit-catchments, bifurcation links and approximate adaptive sub-steps are listed in Table 2. The same model options are enabled in both implementations: adaptive sub-step integration, bifurcation routing and on-the-fly logging of the selected diagnostic output. The hardware configurations used in the benchmarks are summarized in Table 1.

For the speed benchmark, the runoff forcing is the daily 1 ° binary sample runoff distributed with the CPU CaMa-Flood release; the sample forcing includes year 2000 and was prepared from the output of the Ensemble Land State Estimator (ELSE) (Kim et al., 2009). To limit non-computational bottlenecks, we also control the I/O path so that the wall-clock comparison reflects routing performance rather than file-format overhead. The GPU implementation overlaps forcing input with computation through its asynchronous dataloader, whereas the CPU reference reads forcing synchronously. Each run therefore saves only one key output variable, and the CPU reference writes that output in binary format. The CPU reference is CaMa-Flood v4.23 compiled with Intel Fortran using the flags "-O3 -fp-model precise" and run in hybrid MPI/OpenMP mode; we tested several MPI/OpenMP combinations and used the fastest configuration for each CPU benchmark, with 16 MPI ranks following the official CaMa-Flood recommendation on the multi-core hosts. GPU runs use a Triton block size of 128, which specifies the number of unit-catchments processed by one kernel instance. All reported wall-times are means over five repeated runs. Table 3 summarizes the benchmark timing results for a 1-year simulation period at the four resolutions on various hardware configurations: a personal workstation with NVIDIA RTX 4070 Ti, the V100 server with 1–4 GPUs, the A100 server with 1–4 GPUs, and the CPU-only server (four CPU nodes) with different numbers of CPU cores.

For the numerical-stability comparison, we use daily forcing series prepared from 0.1 ° ERA5-Land NetCDF runoff (Muñoz-Sabater et al., 2021) and from the eartH2Observe (E2O) Tier-1 ensemble runoff (Schellekens et al., 2017). The 1980-2014 period is adopted to remain consistent with the temporal coverage of the E2O Tier-1 dataset. For each numerical-stability comparison, CPU and GPU runs use the same 6-arcmin parameter set, identical forcing, initial states and adaptive sub-step settings, with the bifurcation module enabled in both runs.

Comment 4. A similar comment applies to "Numerical stability" (Section 3.3), which is rather short. Here, there are multiple opportunities for deepening the analysis and demonstrating that the model is indeed stable. For example, you could consider the option of working with runoff data at multiple spatial resolutions (why was a resolution 0.25 degrees adopted?) and using a variety of gauging stations. The current analysis focuses only on three major rivers; how does the two model implementations perform on smaller rivers? How about bifurcation points?

To reviewer: Thank you for these suggestions. We agree that the submitted numerical-stability section was too short and did not explain why the selected setup was informative. In the revision, we focus the main figure on the bifurcation-enabled case because it includes the base routing operations and additionally exercises bifurcation fluxes and atomic accumulation. This avoids duplicating a simpler no-bifurcation figure while still testing the numerically most demanding route through the implementation.

We also explain the forcing choice and broaden the stability evidence. The revised analysis adds a 0.1 ° ERA5-Land runoff comparison because this long, globally available, high-resolution forcing source makes a multi-decadal CPU-GPU comparison possible while exercising NetCDF input, forcing-to-unit-catchment interpolation and the routing kernels together. To address the concern about representativeness, we unified the stability period to 1980-2014 and added discharge comparisons for rivers of different scales, rather than only the three major rivers. The manuscript change is shown below for Section 3.2, replacing the previous short numerical-stability description.

Revised manuscript text:

The numerical-stability comparison follows the setup described in Section 3.1 and covers 1980–2014 with two daily runoff forcing series: E2O Tier-1 ensemble runoff at 0.25 ° and a daily series prepared from ERA5-Land surface runoff at 0.1 °. Using these two forcing datasets, we compare CaMa-Flood-GPU with the reference CPU implementation through spatial mean fields and station hydrographs, and assess whether floating-point differences remain bounded over the multi-decadal simulation.

Figure 8 presents the CPU–GPU comparison under E2O runoff. The CPU and GPU mean river outflow fields (Figure 8a, b) are visually indistinguishable, and the relative-difference map in Figure 8c is dominated by the gray "below-noise-floor" band of ±0.001%, with non-zero values appearing mainly on the largest river stems. Figure 8d–f repeats the comparison for floodplain outflow and Figure 8g–i for river depth, both with the same noise-floor behaviour and the same concentration of residual signal on the main channels of large basins. Across the three variables, the field-mean relative difference remains below 10^{-3}%. Figure 9 reproduces the same comparison under ERA5-Land runoff. The residual magnitude and spatial pattern remain essentially unchanged between Figures 8 and 9, although the two forcing products differ in resolution and temporal variability. This consistency indicates that the residuals arise from the floating-point ordering of the routing calculation rather than from the runoff forcing itself.

Figures 10 and 11 compare simulated discharge for the final four years of the full simulation at six GRDC gauge locations spanning four orders of magnitude in drainage area. At every gauge location, the CPU and GPU curves overlay each other almost exactly, and the right-hand red axis shows the day-by-day CPU–GPU difference at a magnified scale. The residual amplitude follows the depth of the upstream reduction at the station location rather than the absolute discharge magnitude. On the large main stems, GPU `atomic_add` operations are associative and commutative but their execution order is not deterministic, whereas the CPU reduction follows a deterministic order. The two reductions therefore return slightly different floating-point bit patterns, producing a small bounded difference at the m^3/s level, three to four orders of magnitude smaller than the simulated discharge. At the mid-scale tributary and headwater gauge locations, the reduction is shallower and the red difference curve collapses toward zero for most of the displayed period. Figure 11 reproduces the same behaviour under ERA5-Land runoff, and neither forcing shows a monotonic component in the residual over the multi-year window. We therefore do not see evidence of numerical drift accumulating through the integration.

Mass conservation on the irregular unit-catchment graph is preserved by construction. The GPU implementation applies the same source-to-target routing fluxes as the CPU algorithm: main-channel fluxes are accumulated through `scatter_add` reductions, and bifurcation fluxes are accumulated through `atomic_add` reductions rather than overwriting destination storage. Each outgoing flux is therefore added to the corresponding downstream storage term, with the CPU–GPU difference limited to the floating-point summation order discussed above. The bifurcation-enabled configuration exercises both the base routing operations and the bifurcation-specific `atomic_add` path. The no-bifurcation configuration is a strict subset of these operations, so agreement with bifurcation enabled also supports agreement for the simpler routing case. Taken together, the field comparisons, station hydrographs and routing-flux accounting show that CaMa-Flood-GPU reproduces the CPU reference without detectable numerical drift over multi-decadal integrations, across both forcing products and with bifurcations enabled.

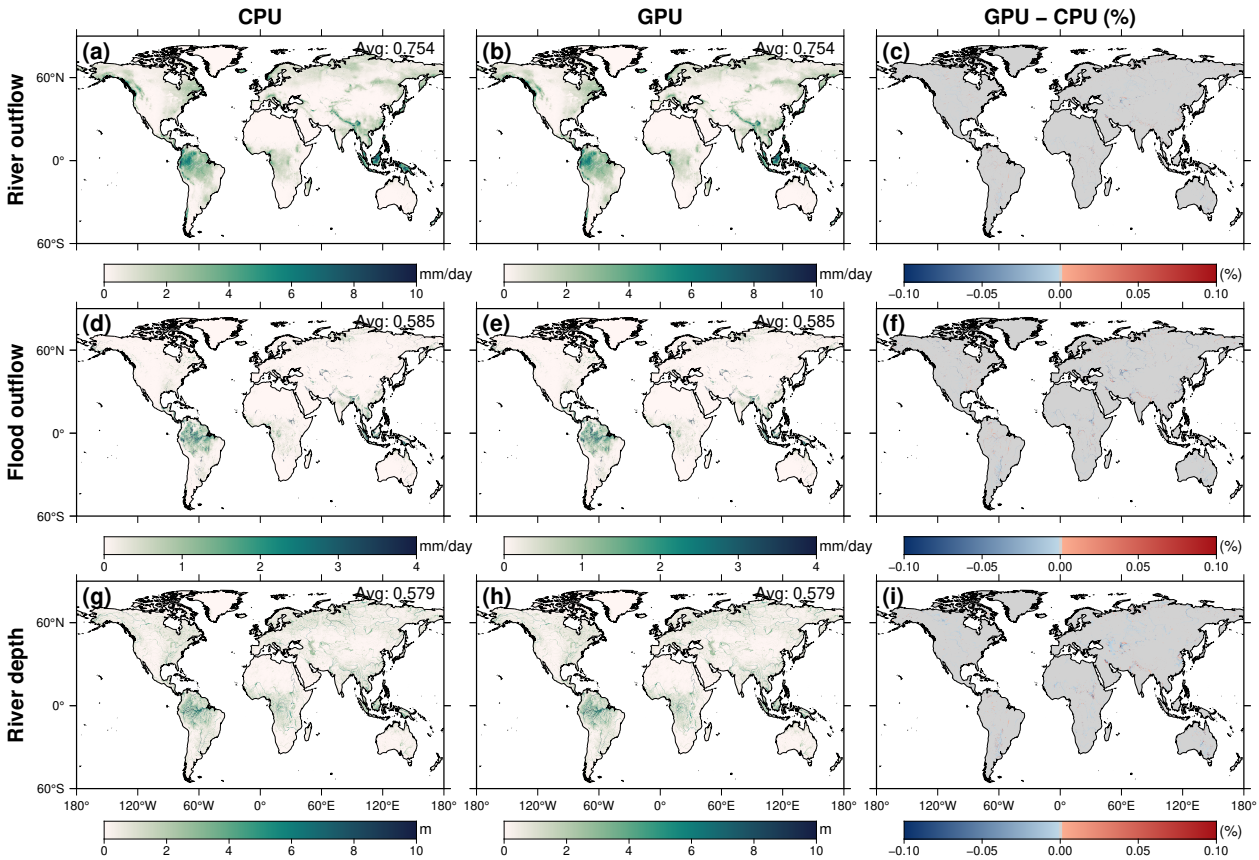

Figure 8. CPU vs. GPU mean fields with the bifurcation module enabled, under E2O Tier-1 ensemble-mean runoff at 0.25 ° for 1980–2014. **(a–c)** Mean river outflow for CPU, GPU and their relative difference. **(d–f)** Mean floodplain outflow for CPU, GPU and their relative difference. **(g–i)** Mean river depth for CPU, GPU and their relative difference. The third-column colour bar is bipolar with a flat gray band at ±0.001% to mark the floating-point noise floor.

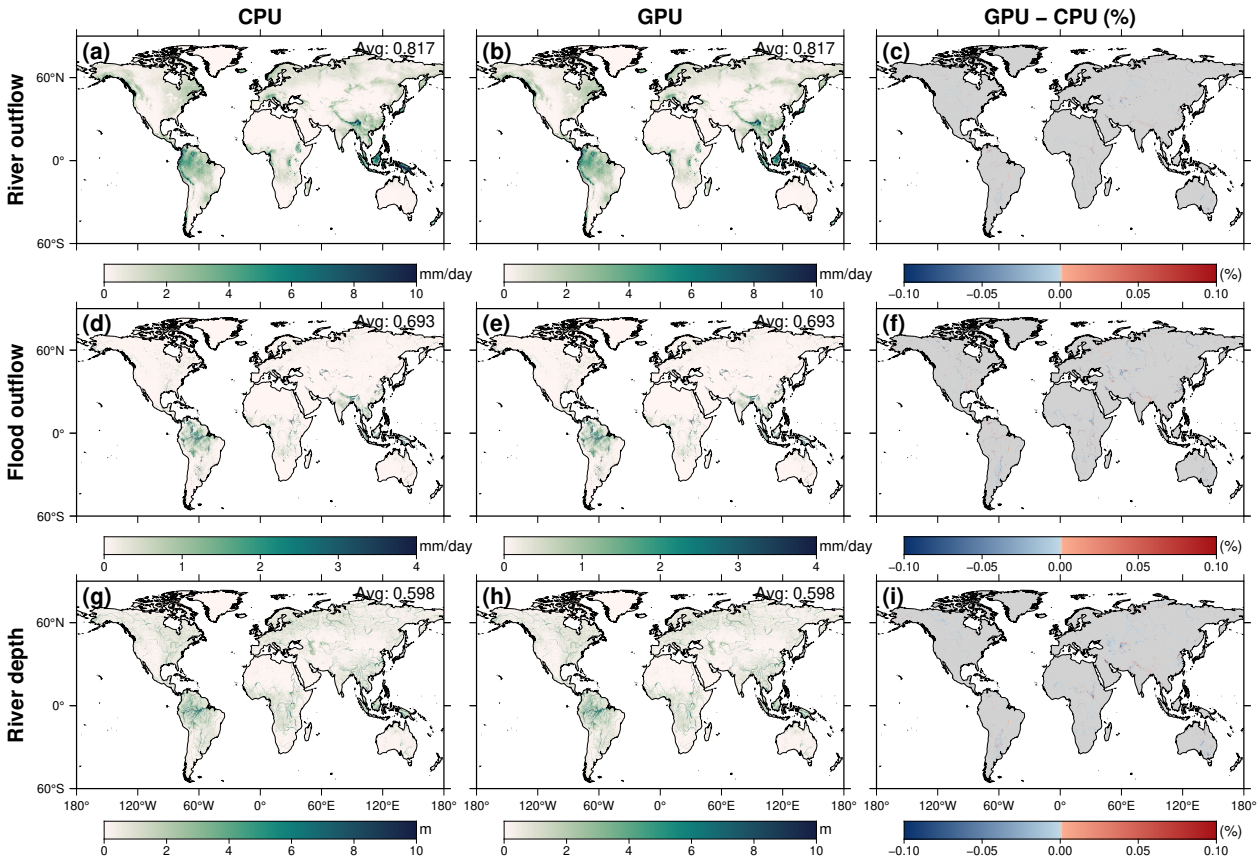

Figure 9. Same as Figure 8, but with ERA5-Land surface runoff at 0.1 °.

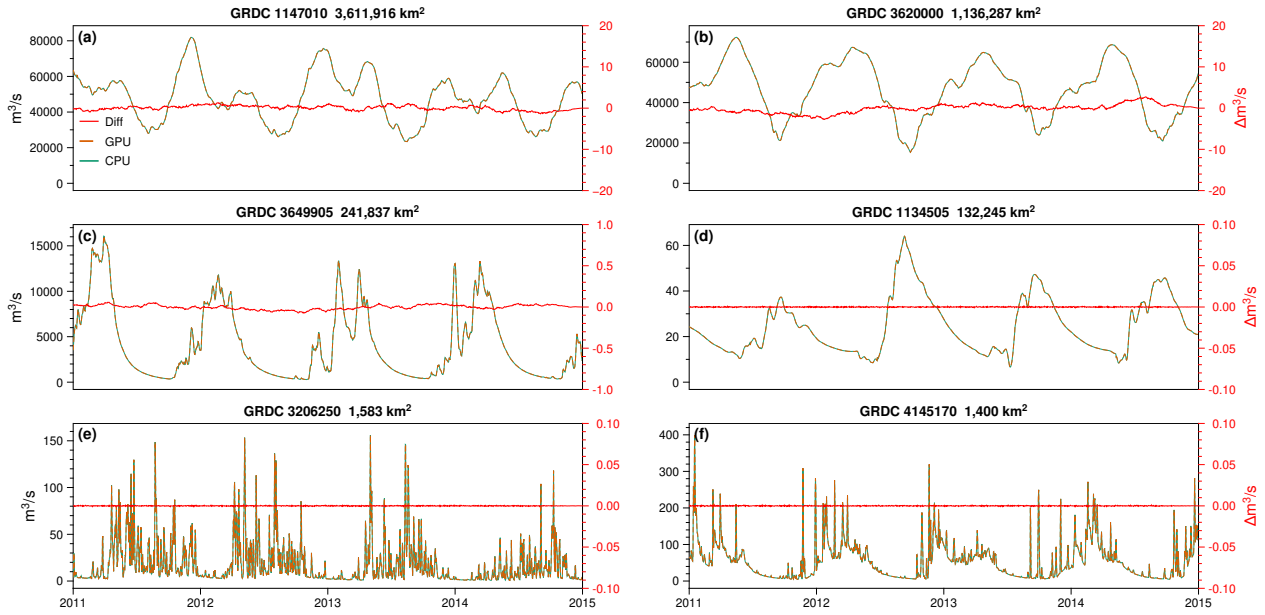

Figure 10. Simulated daily river discharge at the locations of six GRDC gauges spanning four orders of magnitude in drainage area, for the last four years of the 1980–2014 simulation under E2O Tier-1 runoff at 0.25 °. The gauge panels are ordered by drainage area from large to small. CPU (green solid) and GPU (orange dashed) curves overlap; the difference is plotted as a red solid line with values given on the right-hand axis.

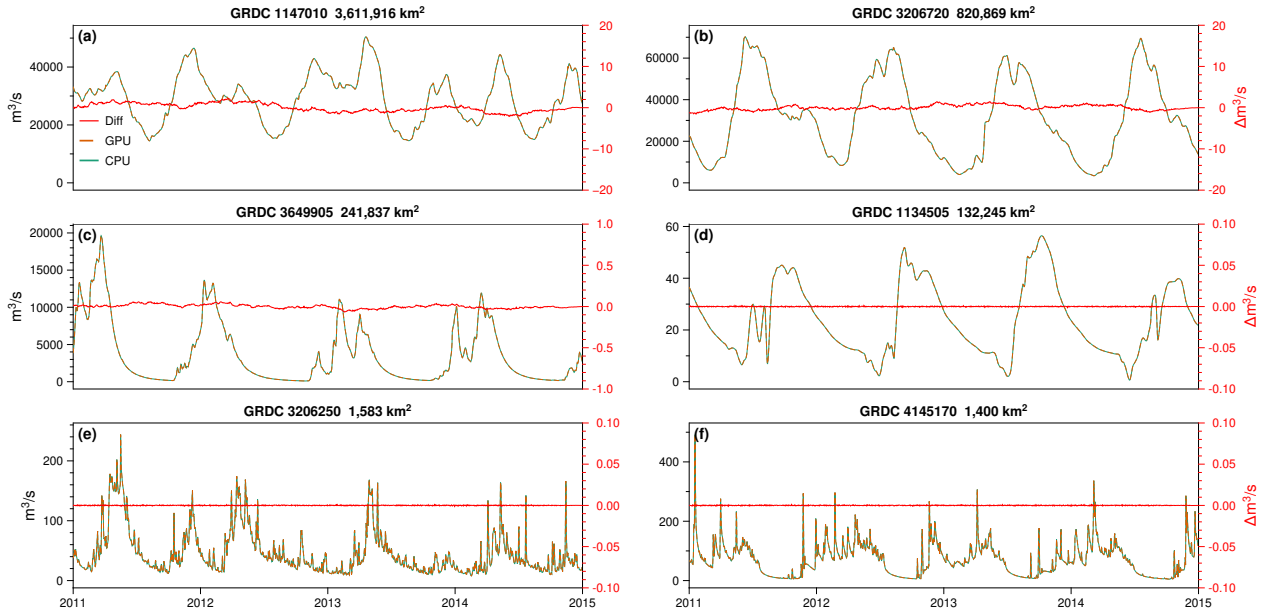

Figure 11. Same as Figure 10, but with ERA5-Land surface runoff at 0.1 °.

Comment 5. Finally, I suggest expanding the Conclusions, which read more like an extended abstract. Specifically, the discussion is now limited to Line 353-356 and could be extended. For example how can the "model modularity" support the integration of reservoir operations and sediment

transport? This should ideally relate to existing / past efforts by the CaMa-Flood community, since model extensions integrating reservoirs already exist.

To reviewer: Thank you for the suggestion. We agree that the submitted Conclusions read too much like a summary and did not make the implications of modularity concrete enough. We therefore expanded the discussion to explain what a new process module would look like in CaMa-Flood-GPU: it declares its own per-unit-catchment state, registers an update kernel and contributes fluxes back through the same routing reduction used by the core solver.

This lets us relate the GPU framework to existing CaMa-Flood community developments rather than describing modularity abstractly. Reservoir operation schemes and sediment-transport extensions can be interpreted as additional catchment-level states and update kernels, which makes them natural candidates for the sub-module interface. We also made the differentiable-routing direction more specific by connecting the PyTorch/Triton design to gradient-based calibration and coupling with PyTorch-native land-surface or AI components. The manuscript change is shown below for the Conclusions, expanding the discussion previously limited to Lines 353-356.

Revised manuscript text:

Looking ahead, there are several avenues for further development. First, additional physics and processes could be incorporated. CaMa-Flood-GPU realizes this through a sub-module layer in which each physical process is encapsulated as a self-contained component. A component declares its own per-unit-catchment state fields, registers one update kernel called once per time step by the main integrator, and contributes any fluxes back to the channel network through the same `scatter_add` reduction used by the core routing. New processes are therefore added by writing one such sub-module rather than by modifying the existing flood-routing kernels or the time-step loop. For reservoir operation, the CaMa-Flood community has already developed schemes such as the global flood-control reservoir module of Hanazaki et al. (2022) and the H08-CaMa-Flood coupling of Shin et al. (2020). These schemes both express reservoir storage and release as per-time-step updates at the unit-catchment level, which maps directly onto this interface. Sediment transport schemes such as the global sediment-dynamics model of Hatono & Yoshimura (2020) similarly reduce, on the GPU side, to a small number of additional catchment-level state fields and one update kernel. These extensions can therefore be ported into the GPU framework as optional modules without rewriting the existing code. Second, a natural research direction for CaMa-Flood-GPU is end-to-end differentiable global routing. The integrator is built in PyTorch and uses custom Triton kernels for the core routing solver: the PyTorch layer already provides automatic differentiation for everything expressed as standard tensor operations, and the Triton kernels can be paired with hand-written backward kernels in the same way as in the wider PyTorch-Triton ecosystem. Once that pairing is in place, parameters such as Manning roughness, river width, floodplain elevation profile and even reservoir operating rules can be calibrated by gradient descent against discharge or altimetry observations, or trained jointly with PyTorch-based land-surface or AI components, in line with the differentiable-geoscience programme advocated by Shen et al. (2023). As part of our commitment to community engagement, we will provide thorough documentation and example cases to encourage broader adoption.

Comment 6. Line 22-23: I would provide more details on "certain terms" as not all GMD readers may be familiar with hydrodynamic modeling.

To reviewer: Thank you for pointing this out. We rephrased the opening hydrodynamic-model description so the local inertial approximation and sub-grid inundation are defined directly. The manuscript change is shown below for the Introduction, replacing the sentence around Lines 22-23.

Revised manuscript text:

However, applying such 2D models to large river basins is computationally prohibitive for many applications. To improve efficiency, simplified models such as LISFLOOD-FP adopt the local inertial formulation, which neglects the convective-acceleration term in the shallow-water momentum equation while retaining gravity and bed-friction terms; this preserves the propagation of flood waves at a fraction of the cost of the full Saint-Venant equations (Bates et al., 2010; de Almeida et al., 2012; Neal et al., 2021). A further-scalable alternative is the sub-grid inundation approach: instead of solving the 2D shallow-water equations explicitly, the inundated area and depth within each computational unit are diagnosed from a pre-computed sub-grid topographic profile, given the unit's total water storage (Yamazaki, 2025). The Catchment-based Macro-scale Floodplain model (CaMa-Flood) is a leading example of this approach, enabling efficient yet physically-based global flood simulations (Yamazaki et al., 2011).

Comment 7. Line 23-24: Same comment as above.

To reviewer: This point is addressed together with the previous comment by the same revised Introduction passage. The manuscript change is shown below for the Introduction and is the same revised passage used to address Lines 22-23.

Comment 8. Line 25: Can you provide evidence of the "widespread adoption and balanced fidelity and efficiency" of CaMa-Flood?

To reviewer: Thank you for catching this. We retained the sentence, corrected the citation for the unit-catchment-discretization claim, and added benchmark references directly at the adoption claim. The manuscript change is shown below for the Introduction, replacing the sentence around Line 25.

Revised manuscript text:

CaMa-Flood introduced a novel vectorized unit-catchment discretization of the river network, a departure from traditional uniform grids (Yamazaki et al., 2011). Among available global routing models, CaMa-Flood provides an established catchment-based representation of channel storage, floodplain storage and river-network routing. We selected CaMa-Flood as the baseline model for our GPU implementation; it has been adopted as the offline river-routing layer in land-surface and global hydrological models, with benchmark comparisons reporting measurable gains in discharge reproducibility relative to the native routing schemes (Zhao et al., 2017; Heinicke et al., 2024).

Comment 9. Line 66-76: Can you add a few references to support these statements?

To reviewer: Thank you. We added supporting references to the paragraph while keeping the narrative structure unchanged. The manuscript change is shown below for the Introduction, replacing the paragraph around Lines 66-76.

Revised manuscript text:

The remaining gap therefore lies in global-scale river routing models such as CaMa-Flood, whose irregular unit-catchment networks pose challenges that differ fundamentally from those encountered in regional 2D GPU flood models. This likely stems from several intrinsic characteristics of large-scale hydrodynamic models that make GPU computation more challenging: (1) In 2D models, the connectivity between computational cells is uniform and limited to adjacent neighbors on a regular grid, whereas in global river models, the river network topology is defined by irregular upstream–downstream relationships that must be handled explicitly (Mizukami et al., 2016; Yamazaki et al., 2011). (2) In 2D models, the relationship between water storage and water level within each grid cell is generally linear or prescribed by a simple function, but global river models employ sub-grid floodplain topography, resulting in a nonlinear and spatially variable relationship (Yamazaki et al., 2011). (3) While 2D models use uniformly shaped grid cells as computational units, global river models discretize the land surface into irregular catchment-based units. This introduces interpolation across variable areas and greatly increases the total number of computational elements when performing high-resolution global simulations (Yamazaki et al., 2019), making efficient parallelization far more demanding.

Comment 10. Table 3: Is there a specific reason for choosing the Year 2000?

To reviewer: Thank you for the question. The benchmark period is calendar year 2000 (365 days, 2000-01-01 to 2000-12-31). Year 2000 was selected because it is included in the sample runoff forcing distributed with the CPU CaMa-Flood release, which makes the benchmark straightforward to reproduce. This is already stated in the new experimental-setup paragraph in Section 3.1 shown above.

References

- Alvanos, M., and Christoudias, T. (2017). GPU-accelerated atmospheric chemical kinetics in the ECHAM/MESSy Earth system model (version 2.52). Geoscientific Model Development, 10, 3679-3693. https://doi.org/10.5194/gmd-10-3679-2017
- Bates, P. D., Horritt, M. S., and Fewtrell, T. J. (2010). A simple inertial formulation of the shallow water equations for efficient two-dimensional flood inundation modelling. Journal of Hydrology, 387, 33-45. https://doi.org/10.1016/j.jhydrol.2010.03.027
- Caviedes-Voullieme, D., et al. (2023). SERGHEI (SERGHEI-SWE) v1.0: a performance-portable high-performance parallel-computing shallow-water solver for hydrology and environmental hydraulics. Geoscientific Model Development, 16, 977-1008. https://doi.org/10.5194/gmd-16-977-2023
- de Almeida, G. A. M., Bates, P., Freer, J., and Souvignet, M. (2012). Improving the stability of a simple formulation of the shallow water equations for 2-D flood modeling. Water Resources Research, 48, W05528. https://doi.org/10.1029/2011WR011570
- Hanazaki, R., Yamazaki, D., and Yoshimura, K. (2022). Development of a reservoir flood control scheme for global flood models. Journal of Advances in Modeling Earth Systems, 14, e2021MS002944. https://doi.org/10.1029/2021MS002944

- Hatono, M., and Yoshimura, K. (2020). Development of a global sediment dynamics model. Progress in Earth and Planetary Science, 7, 59. https://doi.org/10.1186/s40645-020-00368-6
- Heinicke, S., et al. (2024). Global hydrological models continue to overestimate river discharge. Environmental Research Letters, 19, 074005. https://doi.org/10.1088/1748-9326/ad52b0
- Hokkanen, J., Kollet, S., Kraus, J., Herten, A., Hrywniak, M., and Pleiter, D. (2021). Leveraging HPC accelerator architectures with modern techniques: hydrologic modeling on GPUs with ParFlow. Computational Geosciences, 25, 1579-1590. https://doi.org/10.1007/s10596-021-10051-4
- Kim, H., Yeh, P. J.-F., Oki, T., and Kanae, S. (2009). Role of rivers in the seasonal variations of terrestrial water storage over global basins. Geophysical Research Letters, 36, L17402. https://doi.org/10.1029/2009GL039006
- Mateo, C. M. R., Yamazaki, D., Kim, H., Champathong, A., Vaze, J., and Oki, T. (2017). Impacts of spatial resolution and representation of flow connectivity on large-scale simulation of floods. Hydrology and Earth System Sciences, 21, 5143-5163. https://doi.org/10.5194/hess-21-5143-2017
- Mizukami, N., et al. (2016). mizuRoute version 1: a river network routing tool for continental-domain water resources applications. Geoscientific Model Development, 9, 2223-2238. https://doi.org/10.5194/gmd-9-2223-2016
- Morales-Hernández, M., et al. (2021). TRITON: A multi-GPU open source 2D hydrodynamic flood model. Environmental Modelling and Software, 141, 105034. https://doi.org/10.1016/j.envsoft.2021.105034
- Muñoz-Sabater, J., Dutra, E., Agusti-Panareda, A., Albergel, C., Arduini, G., et al. (2021). ERA5-Land: A state-of-the-art global reanalysis dataset for land applications. Earth System Science Data, 13, 4349-4383. https://doi.org/10.5194/essd-13-4349-2021
- Schellekens, J., Dutra, E., Martínez-de la Torre, A., Balsamo, G., van Dijk, A., Sperna Weiland, F., Minvielle, M., Calvet, J.-C., Decharme, B., Eisner, S., Fink, G., Flörke, M., Peßenteiner, S., van Beek, R., Polcher, J., Beck, H., Orth, R., Calton, B., Burke, S., Dorigo, W., and Weedon, G. P. (2017). A global water resources ensemble of hydrological models: the eartH2Observe Tier-1 dataset. Earth System Science Data, 9, 389-413. https://doi.org/10.5194/essd-9-389-2017
- Neal, J., Hawker, L., Savage, J., Durand, M., Bates, P., and Sampson, C. (2021). Estimating river channel bathymetry in large-scale flood inundation models. Water Resources Research, 57, e2020WR028301. https://doi.org/10.1029/2020WR028301
- Rong, Y., Bates, P., and Neal, J. (2024). GPU-accelerated urban flood modeling using a nonuniform structured grid and a super grid scale river channel. Water Resources Research, 60, e2023WR036128. https://doi.org/10.1029/2023WR036128
- Sharifian, M. K., Kesserwani, G., Chowdhury, A. A., Neal, J., and Bates, P. (2023). LISFLOOD-FP 8.1: new GPU-accelerated solvers for faster fluvial/pluvial flood simulations. Geoscientific Model Development, 16, 2391-2413. https://doi.org/10.5194/gmd-16-2391-2023
- Shen, C., Appling, A. P., Gentine, P., et al. (2023). Differentiable modelling to unify machine learning and physical models for geosciences. Nature Reviews Earth and Environment, 4,

552-567. https://doi.org/10.1038/s43017-023-00450-9

- Shin, S., Pokhrel, Y., Yamazaki, D., Huang, X., Torbick, N., Qi, J., Pattanakiat, S., Ngo-Duc, T., and Nguyen, T. D. (2020). High resolution modeling of river-floodplain-reservoir inundation dynamics in the Mekong River Basin. Water Resources Research, 56, e2019WR026449. https://doi.org/10.1029/2019WR026449
- Yamazaki, D. (2025). Advancing global river hydrodynamics simulations by catchment-based macro-scale floodplain modeling approach. Geoscience Letters, 12, 72. https://doi.org/10.1186/s40562-025-00452-z
- Yamazaki, D., Kanae, S., Kim, H., and Oki, T. (2011). A physically based description of floodplain inundation dynamics in a global river routing model. Water Resources Research, 47, W04501. https://doi.org/10.1029/2010WR009726
- Yamazaki, D., de Almeida, G. A. M., and Bates, P. D. (2013). Improving computational efficiency in global river models by implementing the local inertial flow equation and a vector-based river network map. Water Resources Research, 49, 7221-7235. https://doi.org/10.1002/wrcr.20552
- Yamazaki, D., Sato, S., Kanae, S., Hirabayashi, Y., and Bates, P. D. (2014). Regional flood dynamics in a bifurcating mega delta simulated in a global river model. Geophysical Research Letters, 41, 3127-3135. https://doi.org/10.1002/2014GL059744
- Yamazaki, D., Ikeshima, D., Sosa, J., Bates, P. D., Allen, G. H., and Pavelsky, T. M. (2019). MERIT Hydro: A high-resolution global hydrography map based on latest topography dataset. Water Resources Research, 55, 5053-5073. https://doi.org/10.1029/2019WR024873
- Zahura, F. T., Goodall, J. L., Sadler, J. M., Shen, Y., Morsy, M. M., and Behl, M. (2020). Training machine learning surrogate models from a high-fidelity physics-based model: application for real-time street-scale flood prediction in an urban coastal community. Water Resources Research, 56, e2019WR027038. https://doi.org/10.1029/2019WR027038
- Zhao, F., Veldkamp, T. I. E., Frieler, K., Schewe, J., Ostberg, S., Willner, S., et al. (2017). The critical role of the routing scheme in simulating peak river discharge in global hydrological models. Environmental Research Letters, 12, 075003. https://doi.org/10.1088/1748-9326/aa7250