https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

MERIT-FlowTopo v1.0: a reusable computational foundation for hyperresolution hydrology on the global 90 m drainage network

Lulu Jiang[1,2], Huan Wu[3,4*], Weitian Chen[5], Zhijun Huang[1,2], Dai Yamazaki[6], Ting Yang[7], Lingfeng Li[1,2]

[1] Southern Marine Science and Engineering Laboratory (Zhuhai), and School of Atmospheric Sciences, Sun Yat-sen University, Zhuhai 519082, China

[2] Guangdong Province Key Laboratory for Climate Change and Natural Disaster Studies, Sun Yat-sen University, Zhuhai 519082, China

[3] State Key Laboratory of Climate System Prediction and Risk Management/Collaborative Innovation Center on Forecast and Evaluation of Meteorological Disasters, Nanjing University of Information Science and Technology, Nanjing, Jiangsu 210044, China

[4] Key Laboratory of Hydrometeorological Disaster Mechanism and Warning of Ministry of Water Resources/School of Hydrology and Water Resources, Nanjing University of Information Science and Technology, Nanjing, Jiangsu 210044, China

[5] Future Water Laboratory, Innovation Center of Yangtze River Delta, Zhejiang University, Jiaxing 314102, China

[6] Institute of Industrial Science, The University of Tokyo, Tokyo 153-8505, Japan

[7] CAS Engineering Laboratory for Yellow River Delta Modern Agriculture, Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences, Beijing 100101, China

Correspondence: Huan Wu (huanwu_flood@nuist.edu.cn)

Abstract.

Global hydrological models still run at tens of kilometers, too coarse to resolve local floods; hyperresolution, about 1 km globally and 100 m continentally, remains the goal. With the network already mapped, the cost has shifted from mapping it to computing over it: refining a 10 km grid to 1 km multiplies cells a hundredfold, and deeper, longer flow paths make routing climb several

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



hundredfold. Yet how a flow network should be organized has rarely been studied, though that organization governs the cost as much as the hardware. On the global 90 m MERIT Hydro network of 22.2 billion land pixels, we show how three structural choices do so: ordering sets locality in a core's private cache, layering in the shared cache, and partitioning balances load across processors without communication. Because the network is static, each structure is precomputed once from D8 flow directions and released as MERIT-FlowTopo v1.0, a reusable foundation any model traversing the network in drainage order reads instead of rebuilding. Across four terrain kernels (the elementary memory-bound step shared by routing computations on the fixed network), depth-first ordering cuts serial runtime up to 5.1-fold and lowers the simulated L3 cache-miss rate from about 37% to 6%, while a conflict-free downstream layering introduced here keeps converging flows lock-free and deterministic without atomic operations. MERIT-FlowTopo extends MERIT Hydro from a map to a basis for computing on it, fixing in the data the locality, scheduling, and load balance hyperresolution would otherwise demand as parallel-computing expertise.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

1. Introduction

Rivers integrate the terrestrial water cycle, draining the continents through a branching network that governs where water gathers and where floods and droughts strike (Allen & Pavelsky, 2018; Wu et al., 2014). Predicting these dynamics rests on simulating water moving over this network step by step through time. Mainstream global models still do so at the tens-of-kilometer scale: H08 at 1° (about 110 km; Hanasaki et al., 2008), WaterGAP at 0.5° (Müller Schmied et al., 2021), CaMa-Flood at 0.25° to 0.1° (Yamazaki et al., 2011, 2013), and PCR-GLOBWB 2 at 5 arcmin (Sutanudjaja et al., 2018). Operational forecasting remains coarse too: the Global Flood Monitoring System at 0.125° (Wu et al., 2014) and the Global Flood Awareness System at 0.1° (Alfieri et al., 2013), only recently upgraded to 0.05° (GloFAS v4.0; Grimaldi et al., 2023). At this scale a grid cell spans an entire landscape, averaging over the heterogeneity that controls runoff and the local detail that flood, drought, and water-resource prediction depend on. Wood et al. (2011) framed hyperresolution modeling (about 1 km globally, approaching 100 m at the continental scale) as a grand challenge for monitoring Earth's terrestrial water. The community took up the agenda (Bierkens et al., 2015), even as some asked whether resolution alone would suffice without better subgrid processes (Beven & Cloke, 2012; Melsen et al., 2016; Wood et al., 2012).

This agenda rests on a drainage network that is already mapped at the resolution it calls for. That network is encoded by a single primitive: the D8 flow-direction scheme, which assigns each pixel one steepest-descent receiver (O'Callaghan & Mark, 1984) and has been the backbone of digital hydrography ever since (Wu et al., 2012). Dispersive alternatives such as the multiple-flow-direction scheme (MFD; Quinn et al., 1991) and D∞ (Tarboton, 1997) resolve hillslope flow more realistically, but D8 remains the continental- and global-scale standard for its unambiguous connectivity. On this primitive, global hydrography has sharpened in step with its underlying

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

digital elevation models (DEMs): HydroSHEDS at 15 arcsec (about 500 m) from the Shuttle Radar Topography Mission (SRTM) DEM (Farr et al., 2007; Lehner et al., 2008), and MERIT Hydro at 3 arcsec (about 90 m) from the MERIT DEM (Yamazaki et al., 2017, 2019), the finest released global flow-direction product to date. A 1 arcsec (about 30 m) global successor on the TanDEM-X DEM is in production (HydroSHEDS v2), and the National Hydrography Dataset Plus resolves the contiguous United States at about 10 m from 3D Elevation Program (3DEP) elevation data (Moore et al., 2019). To run at a coarser grid, the network is upscaled by established algorithms (Eilander et al., 2021; Wu et al., 2011; Yamazaki et al., 2009), preserving the drainage structure but discarding native resolution. The network every flow computation depends on is therefore already mapped globally at hyperresolution, and finer where data allow. The mapped network still has limitations: the single-receiver D8 network omits bifurcations and distributaries, and the hydraulic parameters a routing scheme needs, such as channel width and depth, remain uncertain at every resolution. This study addresses the cost of computing over the network as mapped.

For hyperresolution global hydrology (Wood et al., 2011), one binding constraint has shifted from mapping the network to computing over it. Recent work has pushed finer: ParFlow resolves the contiguous United States at 1 km (Maxwell et al., 2015); PCR-GLOBWB has reached 30 arcsec (about 1 km), first over Europe (Hoch et al., 2023) then globally (van Jaarsveld et al., 2025), with a groundwater counterpart at the same resolution (Verkaik et al., 2024); and grid-based flow routing has reached 90 m (Jiang et al., 2025a). Computing at these resolutions is costly. The first global 30 arcsec model, a 35-year daily run, would take roughly 4.5 years on a single core; even across 6,528 cores it ran 17 days, against under 3 days on 1,696 cores at the coarser 5 arcmin (van Jaarsveld et al., 2025). Its groundwater companion is no lighter: the largest continental run would

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

take about four and a half months on one core, and a 16-hour target took 12 nodes, 384 cores (Verkaik et al., 2024). Each step finer compounds it: refining today's 10 km grids to 1 km multiplies the cells a hundredfold, while the deeper, longer flow paths and the shorter time steps that finer cells require make the total routing cost climb several hundredfold. Making this computation efficient is the next step.

That efficiency has been pursued on two fronts: more hardware, and the parallel algorithms to exploit it. The parallelism is inherent: D8 flow directions give each pixel a single downstream receiver and a strict upstream-to-downstream dependency, yet parts that share no dependency advance concurrently, across processors and within each. Across processors, the division takes two forms. The first divides the domain along the drainage hierarchy: each processor receives whole basins with no flow between them, so processors never communicate, as in PCR-GLOBWB, whose 5 arcmin domain splits into 53 basin groups (Sutanudjaja et al., 2018), a design carried to its global 30 arcsec successor (van Jaarsveld et al., 2025). The second cuts across it: a basin is split between processors, which then exchange information at subdomain boundaries by message passing (MPI; RAPID: David et al., 2011; J. Liu et al., 2016; mizuRoute: Mizukami et al., 2021). Within a processor, finer schemes order the network into layers and advance a basin's independent pixels together across cores (Jiang et al., 2025a; J. Liu et al., 2014; LISFLOOD: van der Knijff et al., 2010), and most recently routing has moved to graphics processing units (GPUs) (Freitas et al., 2022; Kang et al., 2025; Y.-H. Liu et al., 2023). Adding cores, though, reaches a limit the hardware cannot cross: each kernel touches every pixel with little computation, so cost follows data movement rather than arithmetic, set by how the network moves through the memory hierarchy. How the traversal beneath these designs is organized governs that movement, yet it has owed more

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

to developer experience than to any measured account of its cost, which coarse resolution long kept negligible.

That organization takes three forms, each meeting the machine at a different level. At hyperresolution a single layer of the network runs to several gigabytes, far past a cache of kilobytes to megabytes, so what governs the cost is where the traversal sits against the machine, not how much arithmetic it does. The kernels benchmarked here are light, but they are the elementary step of the heavier computations the framework targets, where the pressure only grows. On one core, the order in which pixels are visited sets cache locality: a network far larger than cache leaves the order to decide whether a pixel's neighbor is still cached when reached, or must be fetched from main memory. Across the many cores of a processor, how the network is layered sets locality in the cache they share, whether each layer's gigabytes fit it or spill to main memory. Across several processors, how the domain is partitioned sets load balance and communication: split along the drainage hierarchy, the pieces stay independent, yet an oversized basin can leave other processors idle. The propagation manner, whether receivers gather values or donors scatter them, is a fourth choice left implicit in the designs above: push keeps only the downstream receiver that D8 already encodes but collides where tributaries meet, and a conflict-free downstream layering resolves the collision so push stays lock-free. At coarse resolution the network fits in cache, spreads over a few processors, and the choices cost little; at 90 m, all of them bind.

The few studies comparing these choices confirm they matter: depth-first ordering leaves fewer cross-domain dependencies than breadth-first (Y.-H. Liu et al., 2023), outlet-anchored layering balances the load better than headwater-anchored (Jiang et al., 2025a), and atomic push runs faster

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

than pull or task scheduling for accumulation (Kotyra et al., 2021). Each isolated a single axis for a single kernel within its own cost model, observing that organization matters without tracing it to the data locality behind it. We are not aware of a study that compares these choices together across multiple kernels at global scale, or that asks whether precomputing these structures once could make them reusable across tools and runs. Table 1 surveys how terrain-analysis software, flow-routing models, landscape-evolution codes, and recent parallel-algorithm studies resolve the D8 flow-direction dependency, organized by the granularity of the precomputed structure.

Table 1. Methods for resolving the D8 flow-direction dependency, organized by the granularity of the precomputed structure. Entries span terrain-analysis software, flow-routing models, landscape-evolution codes, and recent parallel-algorithm studies; software appearing in multiple rows implements multiple methods. Entries marked "this work" are implemented in this study.

Class	Method	Software & references
Order-free (no precomputed order)	Locality (packing) order	CaMa-Flood (Yamazaki et al., 2011); RVIC (Hamman et al., 2018); mizuRoute-IRF (Mizukami et al., 2016); MOSART (Li et al., 2013); WRF-Hydro (Gochis et al., 2018); xanthos (Li et al., 2017)
Fine-grained, sequential (pixel-level order)	Topological sort from the sources	WhiteboxTools (Lindsay, 2016); RichDEM (Barnes, 2017); TauDEM (Tarboton, 1997); TopoToolbox (Schwanghart & Scherler, 2014); pysheds (Bartos, 2020); mRM (Thober et al., 2019); SWAT+ (Bieger et al., 2017); PRMS (Markstrom et al., 2015); HYPE (Lindström et al., 2010); HRR (Beighley et al., 2009); MGB (Collischonn et al., 2007); RHESSys (Tague & Band, 2004); WBM (Wisser et al., 2010); MODFLOW 6-SFR (Langevin et al., 2017);

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

		this work
	Breadth-first search from the pit	pyflwdir (Eilander et al., 2021); Y.-H. Liu et al. (2023); this work
	Depth-first search from the pit	Landlab (Hobley et al., 2017); fastscapelib (Braun & Willett, 2013); LSDTopoTools (Mudd et al., 2014); PCRaster (Karssenberg et al., 2010); melfp, meshed (Cho, 2025b, 2025a); PCR-GLOBWB (Sutanudjaja et al., 2018); t-route (NOAA Office of Water Prediction, 2023); mizuRoute-KWT (Mizukami et al., 2016); wflow (van Verseveld et al., 2024); DRYP (Quichimbo et al., 2021); Barnes (2019); Y.-H. Liu et al. (2023); this work
Fine-grained, parallel (pixel-level layers)	As-soon-as-possible layering	mefa (Cho, 2023); Kotyra et al. (2021); Sten et al. (2016); RAPID (David et al., 2011); Richardson et al. (2014); Jiang et al. (2025a); this work
	Conflict-free downstream layering	this work
	As-late-as-possible layering	LISFLOOD (van der Knijff et al., 2010); J. Liu et al. (2014); Barnes (2019); Jiang et al. (2025a); this work
Coarse-grained (subregion partition)	Partition across the drainage hierarchy	TauDEM (Wallis et al., 2009); RichDEM (Barnes, 2017); mizuRoute (Mizukami et al., 2016, 2021); RAPID (David et al., 2011); HRR (Beighley et al., 2009); WRF-Hydro (Gochis

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

	(runtime exchange, typically MPI)	et al., 2018); asynch (Small et al., 2013); J. Liu et al. (2016); Avesani et al. (2021); Y.-H. Liu et al. (2023)
	Partition along the drainage hierarchy (communication-free)	PCR-GLOBWB (Sutanudjaja et al., 2018; van Jaarsveld et al., 2025); this work

This study provides a systematic account of how a flow network's organization governs the cost of computing over it. From the static D8 field, the framework precomputes a reusable library of structures: three serial orderings; three parallel layerings, one of them conflict-free downstream layering, introduced here, which removes the write conflict by scheduling, so push runs lock-free and deterministic for any kernel, even Strahler stream order. Two partitions keep every piece independent without message passing: a basin-level partition, and a subbasin-level partition, introduced here, that splits a region's dominant basin along its mainstem to balance the load across processors. Each is stored as a raster. Across four kernels (flow length downstream, flow length upstream, Strahler stream order, and upstream drainage area) on the global 90 m network of 22.2 billion land pixels in 65 regions, we measure how ordering, layering, and partition together set performance, and distill recommendations for the best combination at each kernel and scale. The structures are released openly as MERIT-FlowTopo v1.0, and the kernel products derived from them as MERIT-DrainAttr v1.0. Just as shared flow-direction products spared every study its own mapping, this precomputed topology spares every run its own organization: the locality, scheduling, and load balance that otherwise demand parallel-computing expertise come fixed in the data, and a kernel simply reads them.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

2. Explicit topological representations of the drainage network

Every structure in this study is built from a single static, two-dimensional D8 flow-direction raster. The raster is first flattened into a compact set of one-dimensional arrays. From these, the framework builds three serial sequences, each ordering the pixels into one upstream-to-downstream pass, and three parallel layerings, each grouping independent pixels into layers a machine processes at once. A spatial decomposition then splits the domain into balanced, basin-based subregions for multi-processor machines. A running example, a 10 × 10 grid with two basins (basin 1 in blue, basin 2 in green), carries through Figures 1–4. Each structure below is described in the direction it is built, either upstream-to-downstream from the headwaters or downstream-to-upstream from the outlet, so its construction reads directly from the figures.

2.1 Grid-based dependency representation

Figure 1 shows the data structures that make the flow-direction dependencies explicit. The two-dimensional D8 flow-direction raster (dir; Figure 1a) stores at each pixel a code for the neighbor it drains to, under the standard power-of-two encoding. This code is relative: finding a pixel's receiver means decoding the direction and locating that neighbor, so the dependency is only implicit and ill-suited to traversal and accumulation. Flattening the grid makes it explicit. Each pixel takes an absolute linear index, row × ncols + col, forming the array idxs (Figure 1b), and a pixel at index N is written iN. Resolving its D8 code against these indices gives the downstream pointer idxs_ds (Figure 1c), where idxs_ds[i] is the index of the pixel that i drains to, its receiver; the pixels that drain into i are its donors (after Braun & Willett, 2013). Outlets, having no receiver, point to themselves; each is the pit at the root of its D8 tree. These pointers make the network a directed forest rooted at the outlets, the explicit dependency every ordering in this section traverses. In the example, the arrays span all 100 pixels so any one is indexed directly, but only the 73 inside the two basins hold data; the rest are NoData and stay out of every traversal.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

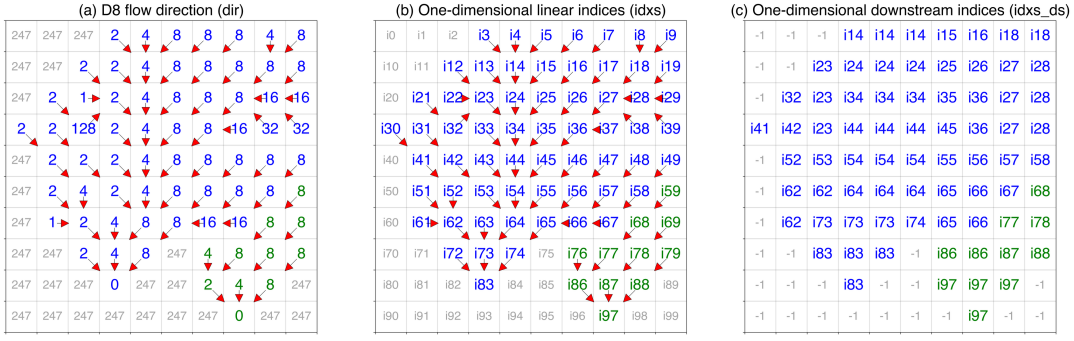


Figure 1. Common representations of flow-direction dependencies for an example with two basins (basin 1, blue; basin 2, green). (a) D8 flow-direction codes (dir) in the standard power-of-two encoding on the two-dimensional grid, with red arrows from each pixel to its receiver. (b) Linear pixel indices (idxs = row × ncols + col), with the same receiver arrows. (c) Downstream pointers (idxs_ds): each pixel stores the linear index of its receiver (pits point to themselves). Gray pixel values are invalid (no-data) locations outside the drainage network; this convention, and the basin colors, recur in the figures that follow.

2.2 Serial traversal sequences

Building on these arrays, a serial sequence linearizes the example's 73 valid pixels into a single order consistent with the upstream-to-downstream dependency. Figure 2 shows three topological orderings that construct it, each drawn in the direction it is built. The topological sort from the sources (Figure 2a) advances upstream-to-downstream: it counts each pixel's in-degree and seeds a queue with the zero-in-degree headwaters. As each pixel is dequeued and emitted, its downstream neighbor's in-degree drops by one, and the neighbor joins the queue once the count reaches zero. This is Kahn's algorithm, which emits pixels directly in topological order. The breadth-first search from the pit sequence (Figure 2b) starts at the outlets and expands upstream through donor lists, emitting all pixels at one distance from the outlet before the next, so the sequence is built downstream-to-upstream, with the outlet first. The depth-first search from the pit sequence (Figure

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

2c) also starts at the outlets but follows each tributary to its headwater before backtracking, descending the largest-area tributary first; like the breadth-first search from the pit, it is built downstream-to-upstream. The three orderings traverse the same 73 pixels but arrange siblings at equal topological depth differently, producing the distinct memory-access patterns that govern serial performance.

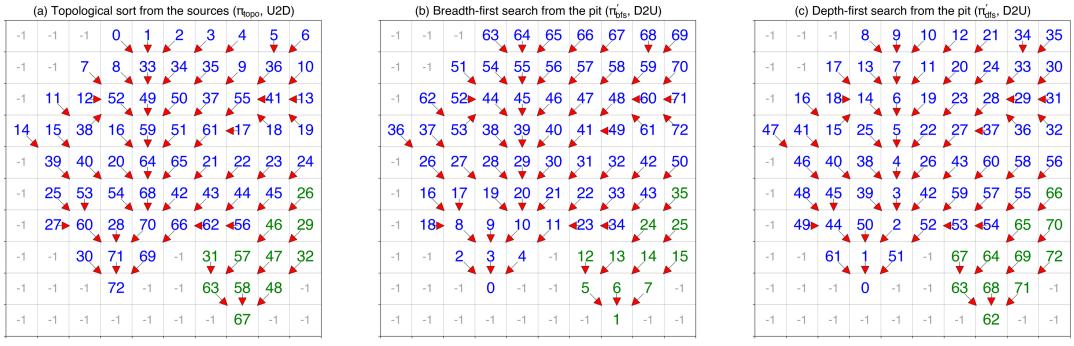


Figure 2. Three serial traversal orders of the same network, each drawn as a raster of per-pixel visit rank in the order's natural direction, with the D8 receiver arrows overlaid. (a) Topological sort from the sources, built by Kahn's algorithm from zero-in-degree pixels and traversed upstream-to-downstream, so rank 0 is a headwater. (b) Breadth-first search from the pit, and (c) depth-first search from the pit, both traversed downstream-to-upstream, so rank 0 is the outlet; in (c) the donors at each confluence are visited in order of decreasing upstream drainage area.

2.3 Parallel layering schemes

Where a serial sequence handles pixels one at a time, a parallel layering groups pixels into ordered layers whose members are mutually independent and run concurrently. Figure 3 shows three such layerings, each again drawn in the direction it is built. As-soon-as-possible layering (Figure 3a) places every headwater pixel in layer 0 and expands downstream, assigning each pixel to one layer beyond the highest layer among its upstream neighbors, its earliest legal layer. This front-loads the hierarchy: 33 of the 73 pixels sit in layer 0 and 59 (81%) within the first three layers. Conflict-free

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

downstream layering (Figure 3b) adds a single constraint to the as-soon-as-possible construction, that no two pixels in a layer share a receiver, deferring any conflicting pixel to the next layer. Pixels i3, i4, and i5, for instance, all drain into i14, so conflict-free downstream layering keeps i3 in layer 0 and moves i4 and i5 to later layers. Removing within-layer write conflicts in this way permits lock-free push updates at confluences, at the cost of more layers. As-late-as-possible layering (Figure 3c), which anchors each pixel at its outlet (outlet-anchored), is built the other way, from the outlet up: each pixel takes its step count to the outlet, the outlet at 0 and the count rising toward the headwaters. Because each pixel has a single downstream path to the outlet, this count is unambiguous. The result spreads pixels far more evenly across layers than the two headwater-anchored schemes. The three layerings group the same 73 pixels into layers differently, nine for as-soon-as-possible and as-late-as-possible layering, twelve for conflict-free downstream layering, producing the locality, load balance, and lock-free concurrency that govern parallel performance.

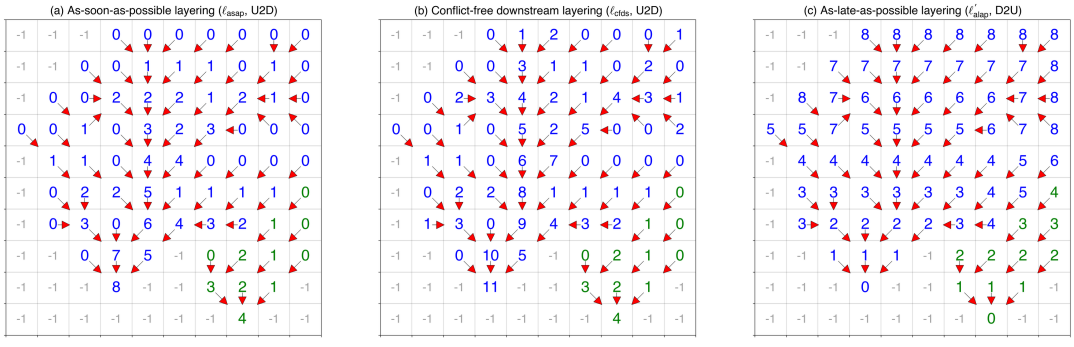


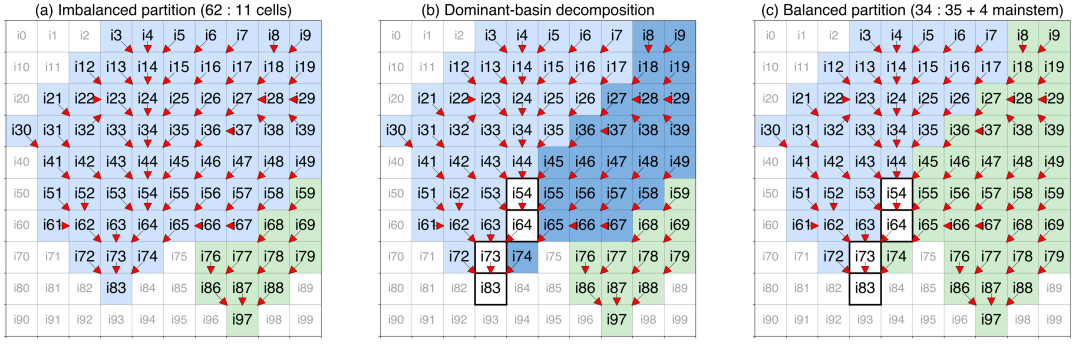
Figure 3. Three parallel layering schemes for the two-basin example, each a raster of the per-pixel layer index with the receiver arrows overlaid. (a) As-soon-as-possible layering: a breadth-first sweep from the headwaters, so layer 0 is at the headwaters and the index grows upstream-to-downstream. (b) Conflict-free downstream layering: as-soon-as-possible with the added constraint that no two pixels in a layer share a receiver, so a layer can be updated by push without conflict.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

(c) As-late-as-possible layering, drawn as the per-pixel rank (step count to the outlet), so layer 0 is at the pit.

2.4 Spatial partitioning domains

Parallel execution over the global domain also requires splitting it into subregions, each processed independently by a separate processor. Two strategies do this, shown in Figure 4 on the two-basin example mapped onto two subregions, with the intermediate decomposition step in panel (b). Basin-level partitioning (Figure 4a) weights each basin by its pixel count and assigns whole basins to subregions; when one basin dominates, this balances poorly: basin 1 (62 pixels) and basin 2 (11 pixels) cannot be split evenly, leaving one subregion nearly idle. Subbasin-level partitioning (Figure 4b, c) eases this imbalance by decomposing the dominant basin along its mainstem. Walking upstream from the outlet and taking the larger-area tributary at each confluence isolates the mainstem and its tributary subtrees; a subtree can then be reassigned to the lighter subregion (Figure 4b), yielding a balanced partition (Figure 4c). The mainstem is processed after its tributary subtrees complete; it holds a negligible share of the pixels, so this final stage adds little to the runtime. Because both strategies partition along the drainage hierarchy (whole basins, or subbasins staged along the mainstem), no values cross subregion boundaries at runtime, and the subregions are computationally independent.



https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

Figure 4. The load-balancing problem and its remedy, on the two-basin example (pixel labels are linear indices; background color gives the subregion a pixel is assigned to). (a) Imbalanced partition: keeping each basin whole leaves the two subregions badly unbalanced when one basin dominates. (b) Dominant-basin decomposition: the dominant basin is split along its mainstem and its tributary subtrees are reassigned; mainstem pixels are outlined in black and processed as a final stage after both subregions complete. (c) Balanced partition: the reassignment evens the two subregions, the basis of subbasin-level partitioning.

3. Benchmark kernels and propagation manners

Four kernels run on the precomputed structures and span the dependency patterns that arise in flow-network computation. Flow length downstream propagates from the outlet upward: each pixel takes its receiver's value and adds the center-to-center step, equal on the schematic grid to the pixel size in the cardinal directions and $\sqrt{2}$ times the pixel size in the diagonal directions, and on the geographic grid to the distance between pixel centers from the latitude-dependent meters per degree, so every pixel depends on exactly one downstream pixel. The remaining three propagate upstream-to-downstream, where several upstream donors meet at one receiver. Flow length upstream takes, over a receiver's donors, the greatest of each donor's path length plus its own donor-to-receiver step. Strahler stream order keeps the largest order among a receiver's donors and increments it only when two or more donors carry that largest order. Upstream drainage area starts each pixel at its own cell area and sums the contributing areas of all donors into the receiver. This downstream accumulation of an additive quantity is the upstream-inflow step that routing on a fixed single-receiver network repeats as it carries water from headwaters to outlets, so upstream drainage area is the closest of the four to one step of that routing. Schemes that resolve backwater add a dependence on downstream state, and flow reversal and bifurcation break the single-receiver

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

assumption; a one-way accumulation represents none of these (Sect. 6.4). Figures A1–A4 walk through one kernel each, step by step over the layers of the two-basin example: flow length downstream (Figure A1), flow length upstream (Figure A2), Strahler stream order (Figure A3), and upstream drainage area (Figure A4).

These kernels can be computed under three propagation manners, contrasted at a confluence in Figure 5. Pull (Figure 5a) has every receiver gather the values of its donors and write only itself, so concurrent updates never collide. The other two scatter, with every donor writing into its shared receiver, where simultaneous writes would race: atomic push (Figure 5b) serializes the colliding writes through hardware atomics, whereas push (Figure 5c) relies on the conflict-free downstream layering, which separates donors that share a receiver into different layers so the writes stay lock-free. The kernel's operator decides which manners apply. Flow length downstream, with a single downstream neighbor per pixel, has nothing to scatter and uses pull alone. The maximum and the sum are associative and commutative, so flow length upstream and upstream drainage area admit all three. Atomic push, however, sums in a run-dependent order, so for these accumulations push under conflict-free downstream layering is the deterministic alternative. The Strahler rule is not associative (its result depends on all donors at once), so no atomic addition can express it, and it admits only pull (Figure 5a) or push (Figure 5c).

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

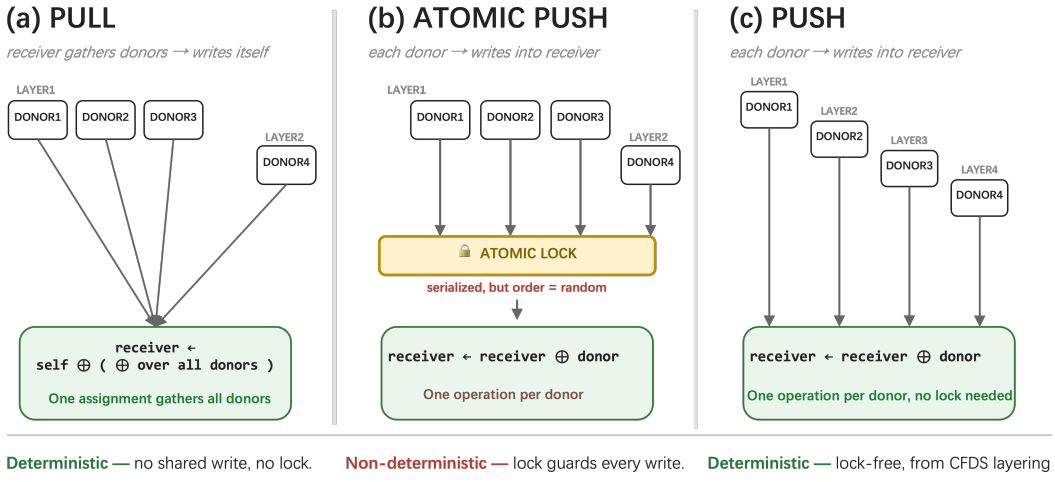


Figure 5. Schematic of the three propagation manners at a D8 confluence, where a receiver accumulates from its donors (up to eight neighbors, here four carrying flow; empty slots gray). (a) Pull: the receiver reads its donors and writes only itself, so the gather has no write conflict and is deterministic. (b) Atomic push: each donor adds into the receiver through a hardware atomic that serializes the writers in an unspecified order, so the result can vary across runs when the operation is non-associative, as in floating-point summation. (c) Push under conflict-free downstream layering: each donor reaches the receiver through its own layer, so at most one writer arrives per layer and the same push is lock-free and deterministic. Each panel gives the update rule for the receiver and the resulting property.

4. Application to the global 90 m MERIT Hydro network

Sections 2 and 3 introduced the representations and the four kernels on a small two-basin example; this section applies them at scale to the global 90 m MERIT Hydro network and measures what each choice of organization costs. Each kernel, flow length downstream, flow length upstream, Strahler stream order, and upstream drainage area, is run serially over each of the three sequences and in parallel over each of the three layerings under the propagation manners it admits (Sect. 3);

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

upstream drainage area is additionally scaled across the four processors under the two partitions. All runs cover the global network, with per-region times summed over the 65 regions, and we report both runtime and the memory-locality metrics that explain the differences among the representations.

4.1 Input data

The global 90 m MERIT Hydro flow-direction dataset (Yamazaki et al., 2019) provides the input topology. It spans 180° W to 180° E and 85° N to 60° S, a grid of roughly 174,000 rows by 432,000 columns (75.17 billion pixels), of which 22.24 billion (29.6%) are land pixels that define the continental domain. Kernels defined on the drainage network instead use the roughly 0.44 billion pixels whose upstream drainage area exceeds 10 km², the threshold HydroSHEDS adopts to delineate the river network (Lehner et al., 2008). The global domain is partitioned into 96 basin-consistent hydrological regions, each enclosing only complete drainage basins and bounded within 38° × 38° (Jiang et al., 2026a); this bound keeps every region's array dimensions and index ranges within 32-bit integer limits. The 65 continental regions (Figure 6a) are the ones analyzed and released here. Region 43, in southern China (Figure 6b), serves as the example region for the per-region memory-locality analyses, while an example basin within it (Figure 6c), a 731 km² coastal catchment ranked 8,595th by drainage area worldwide, is used for visualization.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

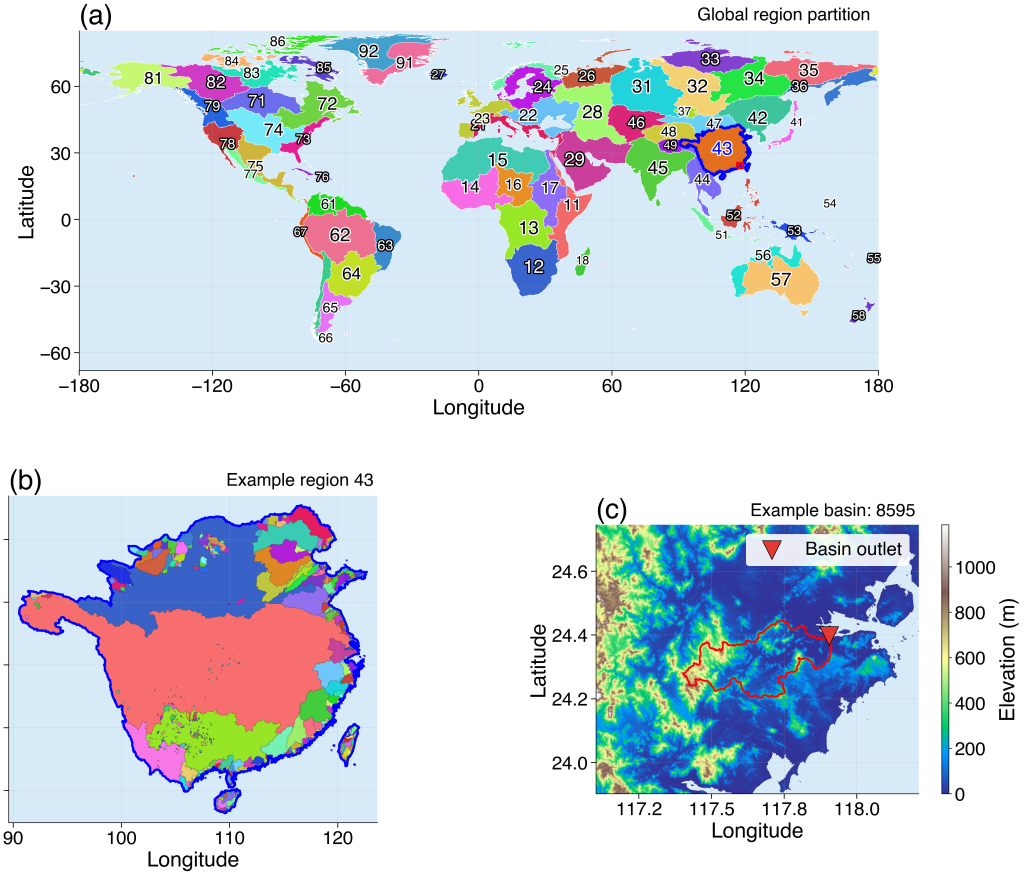


Figure 6. Study region. (a) Global partition of the land surface into hydrological regions, each in a distinct color and labeled by its identifier; Region 43 (southern China) is outlined in blue and the example basin is marked by a red rectangle. (b) The basins of Region 43, each in a distinct color. (c) Elevation of the example basin (drainage area ≈ 731 km²) from the 90 m MERIT Hydro digital elevation model (DEM); the basin outline is red and the outlet is a red downward triangle.

4.2 Workflow

The framework converts the input flow-direction grid (Sect. 4.1) into a small library of reusable traversal structures in a single preprocessing pass, run once before any kernel and independent of the target machine. It first flattens the two-dimensional D8 raster into the shared one-dimensional

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

index arrays of Figure 1: a linear index for every pixel and a pointer to its downstream receiver. From these it builds the six structures introduced in Sect. 2, the three serial sequences of Figure 2, each an ordered sequence for one-pass traversal, and the three parallel layerings of Figure 3, each grouping mutually independent pixels for concurrent processing.

The pipeline runs on each of the 65 hydrological regions independently, so the framework never holds the global domain in one array, and the global maps in this paper are assembled from the per-region outputs for display only. Every structure is computed once and stored in a common upstream-to-downstream orientation, with those built from the outlet reordered into it beforehand, so that the three accumulating kernels read them forward and flow length downstream reads them in reverse, without re-ordering.

One further structure depends on the machine. To run each region across the server's four processors at once (Sect. 4.3), the graph partitioner METIS (Karypis & Kumar, 1998) divides it into four parts, one per processor. It treats every local basin as a node weighted by its pixel count, a proxy for runtime since each kernel scans every pixel once per layer or sweep. The four parts come out contiguous and near-equal in weight, with no basin split across them. Basin-level partitioning stops there, so when one basin dominates a region no cut can balance the parts and its processor carries a disproportionate load. Subbasin-level partitioning adds a second stage that splits the dominant basin along its mainstem (Figure 4), distributing the resulting subbasins across the processors.

4.3 Computing environment and evaluation methods

All experiments ran on a shared-memory server at Sun Yat-sen University with four Intel Xeon Platinum 8270 processors. Each processor holds 26 physical cores at a 2.70 GHz base frequency

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

(4.00 GHz turbo) with hyper-threading, for 104 physical and 208 logical cores across the four processors, each a non-uniform memory access (NUMA) node. Each core has a 32 KB L1 data cache and a 1 MB L2 cache, and each processor shares a 35.75 MB L3 cache.

The code is written in C and compiled with GCC at -O3, and every run uses a NUMA-local configuration with fixed OpenMP thread placement and static scheduling. Every reported time measures the kernel computation alone: the propagation pass over the precomputed sequence or layering (two passes for flow length upstream, Sect. 5.4), timed by a wall-clock timer placed immediately around that pass. Reading the flow-direction grid and the stored structure, preparing the traversal order, and writing the output raster all lie outside the timed region. Three thread settings are reported: a single-thread serial baseline, parallel execution at 13 threads (half of one processor), and a strong-scaling experiment that sweeps 4 to 13 threads and adds 26 threads at full processor occupancy. Each setting repeats the kernel five times within a single launch; because other users share the platform, the two largest elapsed times are discarded and the remaining three averaged. This trimming suppresses contention spikes but cannot remove all run-to-run variability; the residual is small relative to the between-representation differences the comparisons rest on, so it does not change the reported rankings.

The four kernels are light: each does only an addition or a comparison per pixel, so their cost lies in moving data through the memory hierarchy rather than in arithmetic, and memory locality governs performance. Two families of metrics quantify it: for the serial sequences, a dependency distance, a row-jump fraction, and L1, L2, and L3 miss rates from a set-associative cache simulation; for the parallel layerings, a cache-line footprint, with the L1–L3 miss rates from the

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

same cache simulation applied to the layered order. Their definitions (Eqs. B1–B4) and the cache simulation are given in Table B1.

5. Results

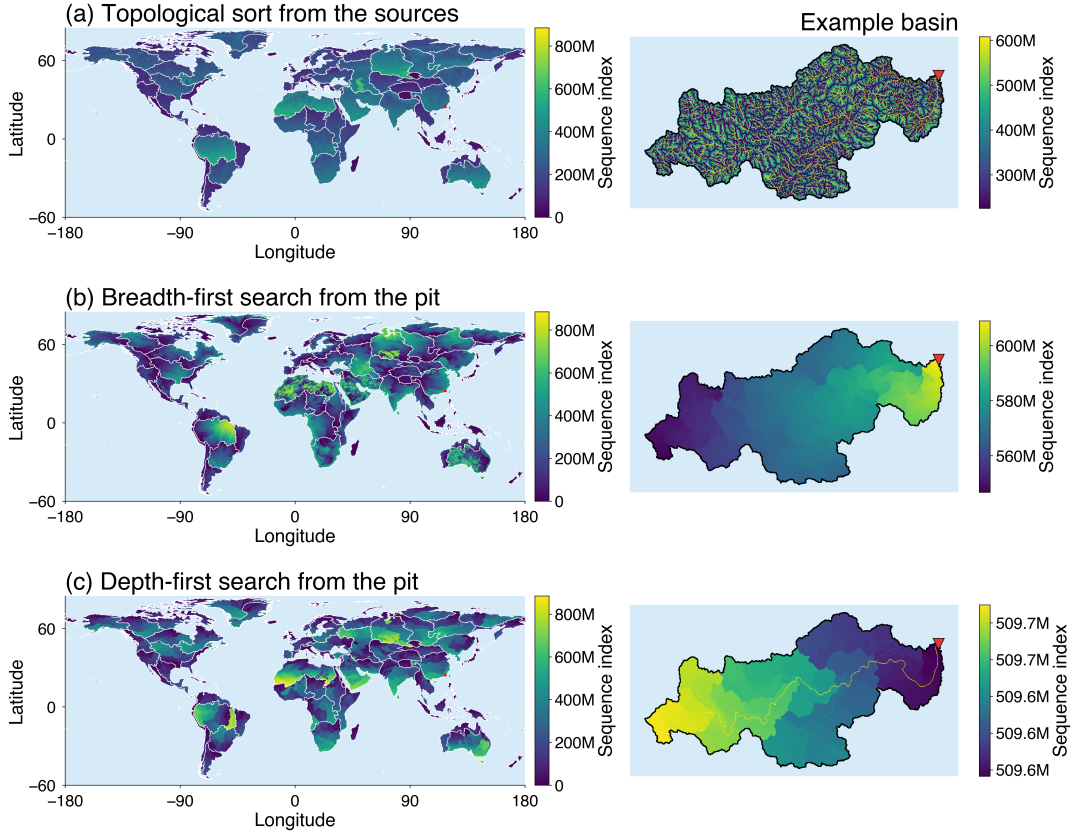
The six topological representations (three serial orderings and three parallel layering schemes) and the two domain partitions (basin-level and subbasin-level) are constructed once from the 90 m MERIT Hydro dataset over the full continental domain of about 22.2 billion land pixels, of which about 0.44 billion are river-channel pixels above a 10 km² upstream-area threshold. Sections 5.1 and 5.2 characterize the spatial structure and memory locality of the serial orders and the parallel layerings, Sect. 5.3 the load balance of the two partitions, and Sect. 5.4 the serial, parallel, and multi-processor runtime of the four benchmark kernels these properties govern. Locality is quantified on the example Region 43, while the global maps and the load-balance statistics span the full set of regions.

5.1 Serial ordering: global maps and memory locality

The three serial orders impose distinct spatial structures on the same flow network (Figure 7). The topological sort from the sources (Figure 7a) increases smoothly from headwaters to outlets, tracing the upstream-to-downstream propagation of the sort; within the example basin it resolves into a fine dendritic pattern that follows the channel network. The breadth-first search from the pit sequence (Figure 7b) advances level by level from the outlet, forming broad bands that widen upstream and track flow length downstream. The depth-first search from the pit sequence (Figure 7c) follows each tributary to its head before backtracking, so the basin breaks into compact blocks, one per subtributary, with sharp index jumps between adjacent tributaries but strong coherence

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

414 within each. Because a sequential kernel walks the pixels in this order and each pixel accesses the
415 value at its downstream receiver, these structures govern how often that value is still cache-resident.



416

417 Figure 7. Sequence index of the three serial orders, mapped at the global scale (left) and for the
418 example basin (right): (a) topological sort from the sources, (b) breadth-first search from the pit,
419 and (c) depth-first search from the pit sequences. Color gives the sequence index (viridis); the
420 global maps share a single scale from 0 to the total pixel count, while each basin map is scaled to
421 itself. On the global maps, white lines are region boundaries and a red rectangle marks the example
422 basin; on the basin maps, the outline is black and the outlet is a red downward triangle. All three
423 sequences are stored in upstream-to-downstream order, so the maps share a single orientation.
424 Breadth-first search from the pit and depth-first search from the pit are built outward from the

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

outlet but re-stored upstream-to-downstream for comparison, so the color shows each order's spatial structure, not where its construction began.

Among the three sequences, the depth-first search from the pit sequence is the most cache-friendly, and the ranking follows the dependency distance (Figure 8, computed for Region 43, about 609 million pixels). This quantity counts the traversal steps between a pixel and its downstream receiver; a small distance means the receiver was accessed recently and is likely still cached. The depth-first search from the pit sequence places over 70% of pixels at distance ≤ 10 (Figure 8a), because depth-first traversal keeps each pixel and its downstream receiver close together in the sequence. The breadth-first search from the pit sequence places a pixel and its receiver in adjacent levels, so distance stays bounded below about 3×10^5, whereas the topological sort from the sources scatters receivers across the whole sequence, with distance reaching 6×10^8. The row-jump fraction (Figure 8b) shows that this ranking is not a matter of spatial contiguity: the depth-first search from the pit lands in a different raster row most often (84.8%), yet its short dependency distances keep it the most efficient. The simulated miss rates (Figure 8c) match this prediction: the depth-first search from the pit misses only 8.8% at L1 and 6.4% at L3; the breadth-first search from the pit overflows L1 and L2 (about 48% and 26%) but its bounded distance holds the working set within L3 (6.5%); the topological sort from the sources stays unfriendly at every level, missing 37.4% even at L3.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

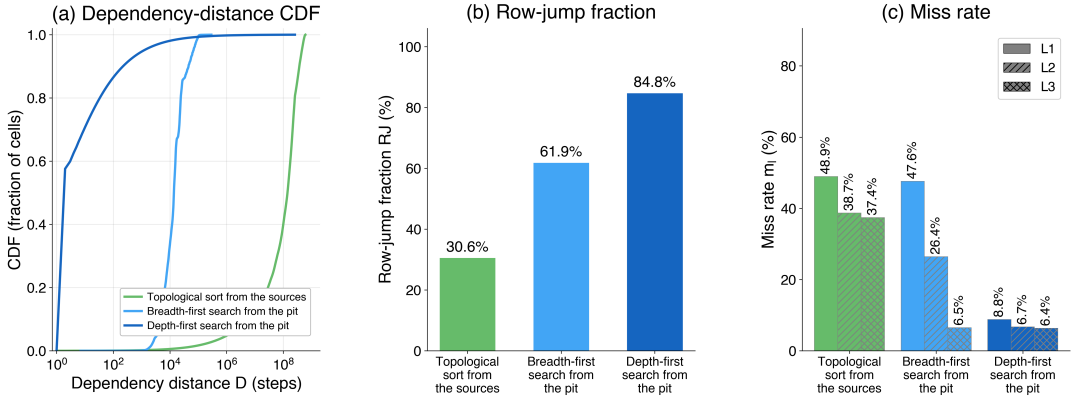


Figure 8. Memory-locality characteristics of the three serial sequences in Region 43. (a) Cumulative distribution of the dependency distance (small distance means a pixel's receiver was visited recently). (b) Row-jump fraction (a high fraction means consecutive visits often jump to a different raster row, a long stride in memory). (c) Simulated L1, L2, and L3 cache miss rates (lower rates mean more reads come from fast cache rather than slower main memory).

5.2 Parallel layering: global maps and memory locality

The three layering schemes fill their layers in very different ways (Figure 9). The as-soon-as-possible and conflict-free downstream schemes (Figure 9a, b) look near-identical at both global and basin scales: most pixels sit in the earliest layers, and only the major river corridors reach high layer indices. The as-late-as-possible scheme (Figure 9c) instead uses the full range of layer indices, so its map varies smoothly across the basin rather than saturating in the lowest layers. This contrast in how pixels fill the layers, spread evenly under as-late-as-possible layering but heavily front-loaded under as-soon-as-possible and conflict-free downstream layering, governs how much memory each layer must touch at once, and how evenly that work divides across threads.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

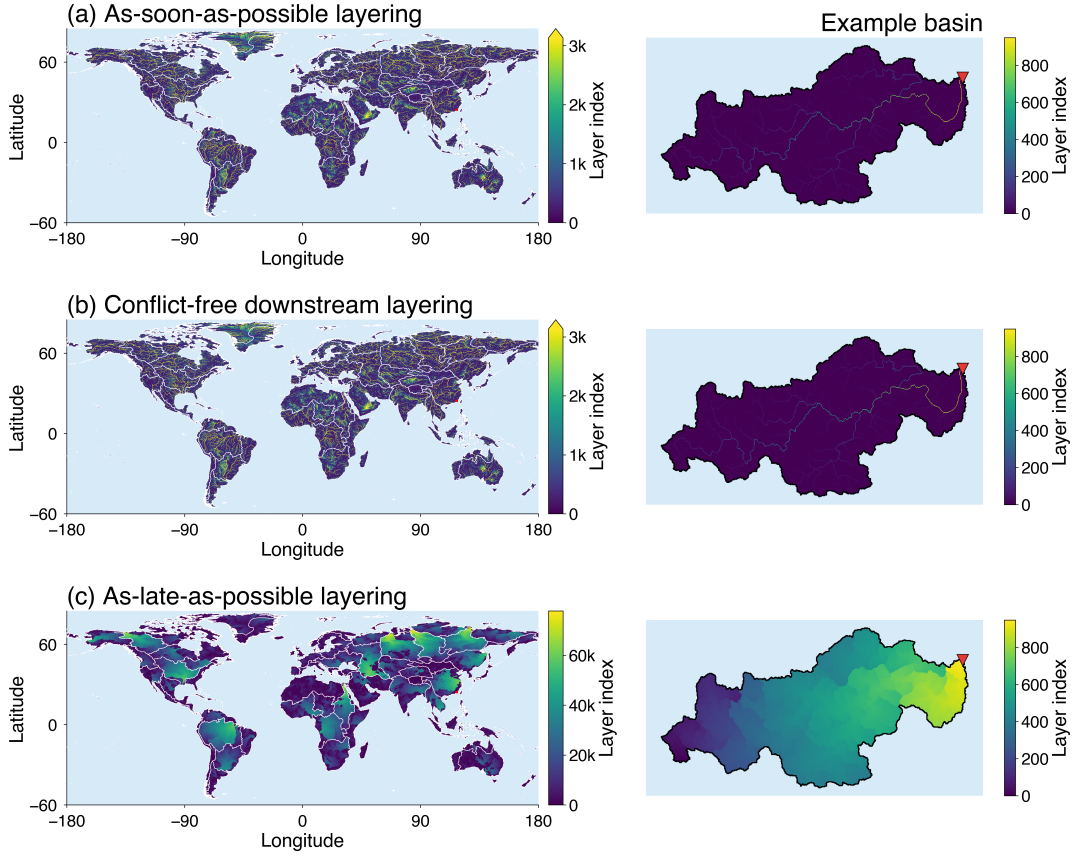


Figure 9. Layer index, mapped at the global scale (left) and for the example basin (right), for (a) as-soon-as-possible, (b) conflict-free downstream, and (c) as-late-as-possible layering. Color gives the layer index (viridis); in (a) and (b) the scale is truncated at the 95th percentile (arrow on the colorbar), while (c) spans the full range. On the global maps, white lines are region boundaries and a red rectangle marks the example basin; on the basin maps, the outline is black and the outlet is a red downward triangle. All three layerings are stored in upstream-to-downstream order, so the maps share a single orientation. The three panels use different color scales, so a given color marks different layer indices across panels; compare the schemes through the cumulative distribution in Figure 10a rather than by color.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

Among the three schemes, as-late-as-possible layering is the most load-balanced and the most cache-friendly (Figure 10, Region 43). The three span nearly the same depth, since the longest flow path sets a floor on the number of layers: as-soon-as-possible and as-late-as-possible layering reach exactly that floor (63,982 layers), and conflict-free downstream layering at least as many, here 63,984, because its conflict-free rule (one donor per receiver per layer) can only push a pixel to a later layer, never an earlier one. What separates the schemes is how pixels populate that depth. The cumulative distribution of the layer index (Figure 10a) shows as-soon-as-possible layering placing 42% of all pixels in layer 0 and conflict-free downstream layering 33%, both reaching about 90% within the first ten layers, whereas as-late-as-possible layering fills the full set of layers far more evenly. This front-loading sets how much memory a single layer must touch at once: the peak per-layer cache-line footprint (Figure 10b) reaches about 37.5 million cache lines under as-soon-as-possible and conflict-free downstream layering (roughly 2.4 GB, far beyond the 35.75 MB L3), so each layer's working set spills to main memory, whereas as-late-as-possible layering touches only about 141,000 (about 9 MB, within one processor's L3) and stays cache-resident. Feeding each layering's visitation order through the same cache simulation as the serial sequences (Figure 10c) confirms the consequence: as-late-as-possible layering misses only 6.5% at L3, close to the depth-first search from the pit sequence, whereas as-soon-as-possible and conflict-free downstream layering miss 37.4% and 42.6%, their oversized first layers overflowing the cache much as the topological sort from the sources does.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

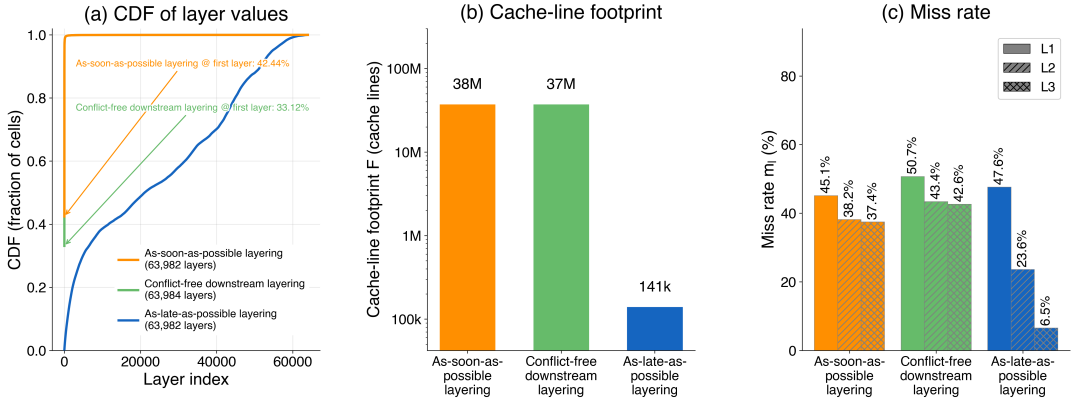


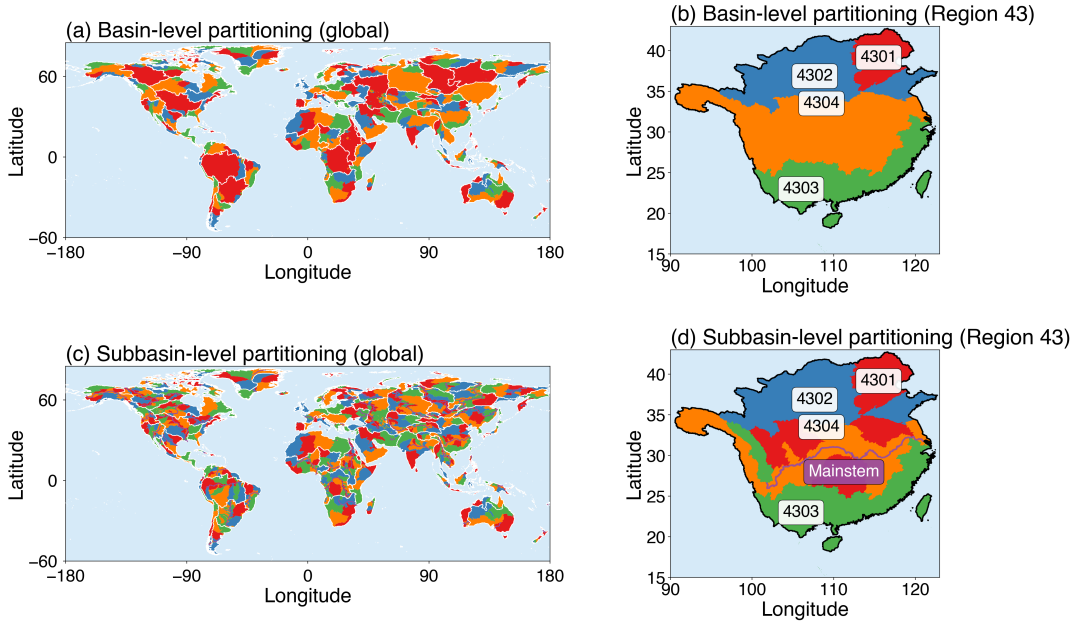
Figure 10. Memory-locality characteristics of as-soon-as-possible, conflict-free downstream, and as-late-as-possible layering in Region 43. (a) Cumulative distribution of the layer index, with each scheme's total layer count in the legend (a steep early rise means the work concentrates in the first layers, an imbalanced load; a gradual, even rise means the layers are balanced). (b) Maximum per-layer cache-line footprint (log scale), the most memory any single layer spans (smaller fits within cache). (c) Simulated L1, L2, and L3 cache miss rates on the layered visitation order (lower means more reads come from cache).

5.3 Spatial partitioning: global maps and load balance

Beyond locality within one processor, parallel execution across the four processors (one per NUMA node, Sect. 4.3) requires splitting each region into four balanced subregions; Figure 11 contrasts two ways of forming them. Basin-level partitioning (Figure 11a, b) treats each drainage basin as indivisible and groups whole basins into the four subregions, so every basin's flow network stays within one subregion. This works where a region holds many comparable basins, but it breaks down when one basin dominates: in Region 43, keeping that basin intact leaves one subregion well above the others (Figure 11b). Subbasin-level partitioning (Figure 11c, d) removes the limit by cutting each region's dominant basin along its mainstem into subbasins that spread

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

504 across the four subregions (flow order preserved by the staging in Sect. 2.4; Figure 4), yielding the
505 even partition of Region 43 in Figure 11d.



506

507 Figure 11. Two partitioning strategies, shown for the global drainage network and for Region 43.
508 Color gives the subregion each pixel is assigned to; every region is split into four, so the four colors
509 recur region by region. Top row, basin-level partitioning, which keeps each drainage basin intact:
510 (a) global and (b) Region 43. Bottom row, subbasin-level partitioning, which additionally splits
511 each region's dominant basin along its mainstem (purple line): (c) global and (d) Region 43. On
512 the Region 43 panels the four subregions are labeled 4301–4304, and the mainstem is labeled in
513 (d).

514 Only subbasin-level partitioning balances the four subregions within each region (Figure 12),
515 measured by the load ratio between the busiest and least-busy subregion, where load is pixel count
516 and a ratio of one is perfect balance. This ratio is what the partition predicts; the runtime payoff
517 appears in Sect. 5.4. Basin-level partitioning leaves this ratio above the imbalance threshold of two

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

(beyond which the busiest processor dominates the parallel runtime) in 35 of the 65 benchmark regions; when a single basin fills most of a region, it cannot be split, so partitioning gains nothing over computing the whole region on one processor. Splitting the dominant basin pulls almost every region toward balance, dropping the median ratio from 2.47 to 1.08 and leaving only eight regions above two. These eight are mostly of two kinds: small or island regions too small to divide evenly into four, which include the worst residual at 6.2, and large regions whose dominant basin stays too big to balance even after the mainstem split. Region 43 shows the typical outcome, tightening from 5.1 to 1.06 once its dominant basin is cut. This balance is the prerequisite for the four-way partition to deliver parallel speedup.

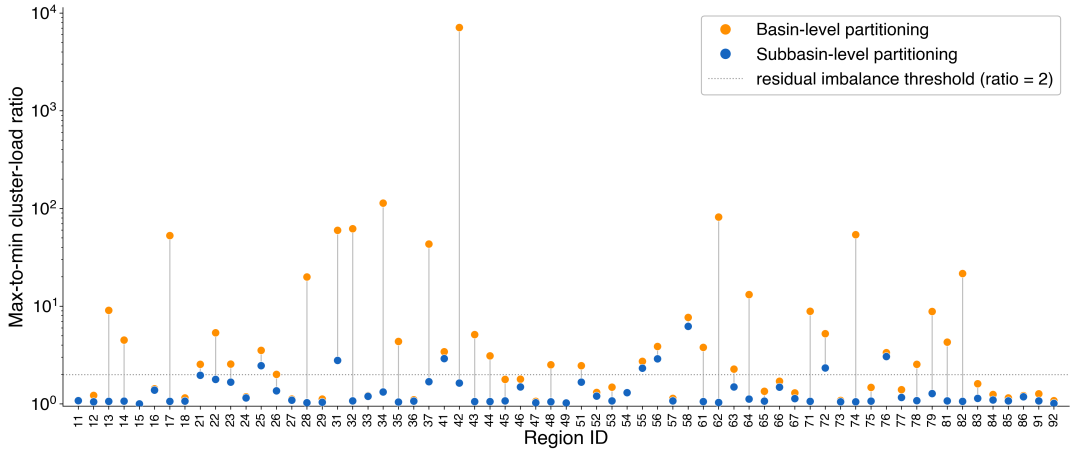


Figure 12. Four-way partition load balance per region, comparing basin-level partitioning (orange) and subbasin-level partitioning (blue) across the 65 benchmark regions. Each region is one paired marker joined by a gray connector, giving the maximum-to-minimum subregion load ratio (load is the subregion pixel count) under each strategy, ordered by region identifier (logarithmic y-axis). A ratio of 1 is perfect four-way balance; the dotted line marks a residual-imbalance threshold at 2.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

5.4 Runtime: single-core, multi-core, and multi-processor

Because each kernel is memory-bound (Sect. 4.3), the locality and balance of Sects. 5.1–5.3 set the wall-clock cost directly. Figure 13 reports each kernel's serial runtime, summed over the 65 benchmark regions, under the three serial sequences. The depth-first search from the pit sequence is fastest for every kernel, confirming the locality analysis of Sect. 5.1, and runs 3.4 to 5.1 times faster than the slowest sequence. The kernels span a wide range of cost: Strahler stream order is lightest at 26.8 s, since it is defined only on the channel network and runs on the 0.44 billion drainage-network pixels rather than the full 22.2 billion land pixels traversed by the other kernels; flow length downstream (132.6 s) and upstream drainage area (147.2 s) are moderate, each a single sweep over the full grid; and flow length upstream is heaviest at 303.7 s because it takes two such sweeps: one propagates the maximum flow length downstream up the network, the other subtracts each pixel's own. These depth-first search from the pit times serve as the serial baseline for the speedups below.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

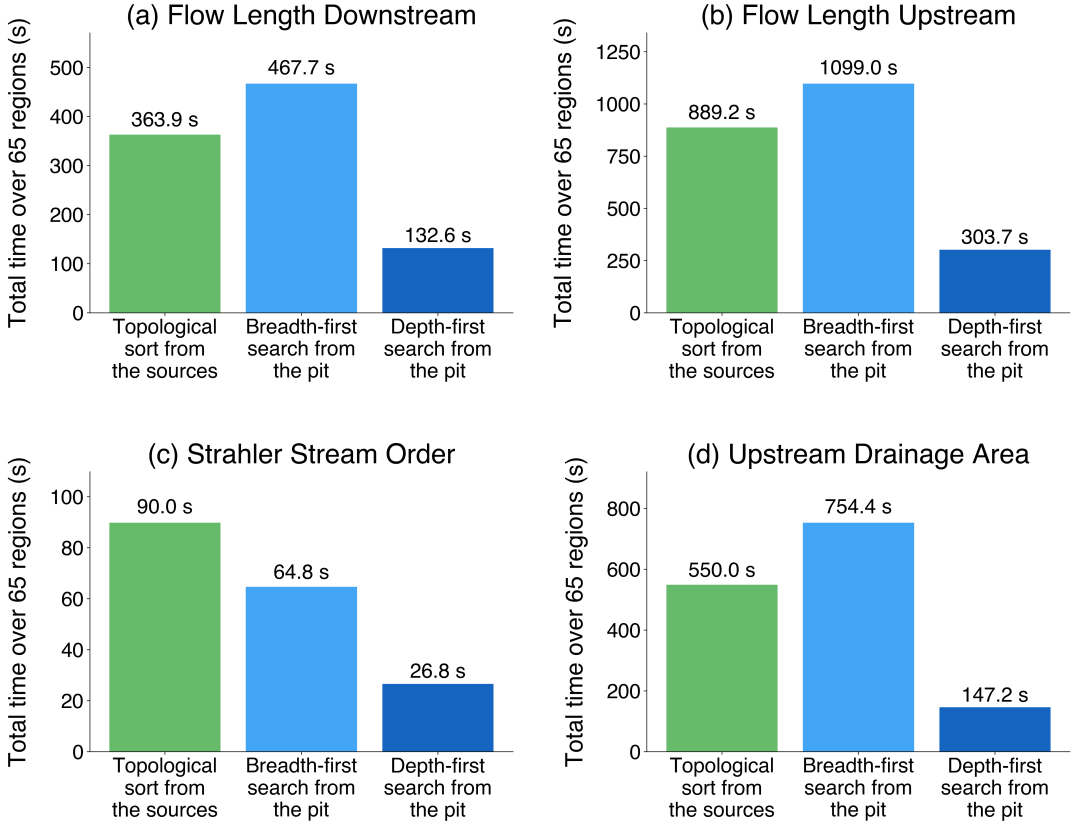


Figure 13. Serial performance of the four kernels across the 65 benchmark regions for the three serial sequences (topological sort from the sources, breadth-first search from the pit, depth-first search from the pit): (a) flow length downstream, (b) flow length upstream, (c) Strahler stream order, (d) upstream drainage area. Bars give the total elapsed time summed over all regions; each region's time is the mean of its three fastest of five runs. The depth-first search from the pit sequence is fastest for every kernel.

At 13 threads, as-late-as-possible layering gives the fastest parallel time for every kernel (Figure 14), in line with its locality and load-balance advantage in Sect. 5.2. These best configurations stay well short of the 13-thread ideal: speedups of 4.9 for flow length downstream (27.0 s), 4.3 for flow length upstream (71.1 s), 3.8 for upstream drainage area (38.9 s), and 5.6 for Strahler stream order

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

(4.8 s), so parallel efficiency runs only 0.43 for Strahler stream order down to 0.29 for upstream drainage area. With so little arithmetic per pixel, the shared memory bandwidth rather than the thread count sets this ceiling. The propagation manner that wins depends on the kernel's dependency rule. Flow length downstream, with one downstream dependency per pixel, admits only the pull manner. Flow length upstream and upstream drainage area aggregate from several upstream neighbors, so atomic push beats pull (71.1 s against 85.1 s for flow length upstream under as-late-as-possible layering). Strahler stream order increments only where two or more equal-order tributaries meet, a non-linear rule that no single atomic addition can express, so under as-late-as-possible layering it runs with the pull manner (4.8 s); the only collision-free push alternative, on conflict-free downstream layering, is slower (9.1 s).

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

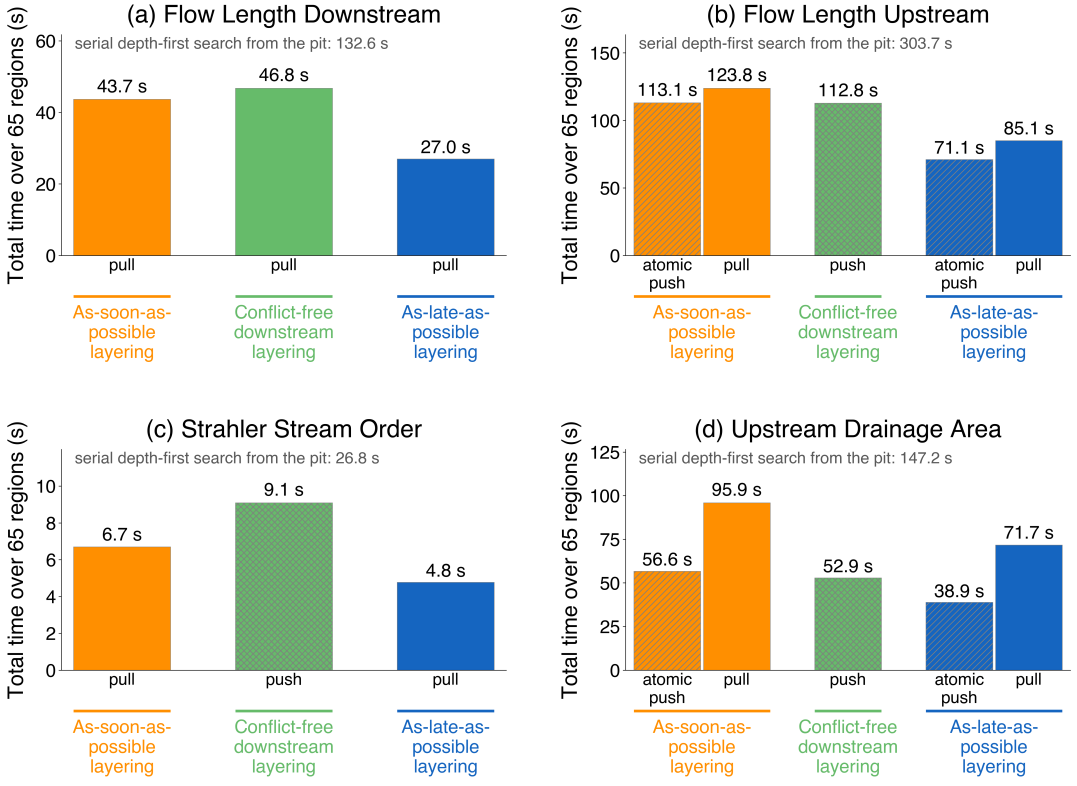


Figure 14. Parallel performance of the same four kernels on 13 threads across the 65 benchmark regions; panels (a)–(d) follow the kernel order of Figure 13. Bars are grouped by layering scheme, namely as-soon-as-possible, conflict-free downstream, and as-late-as-possible, then subgrouped by propagation manner (pull, push, atomic push), distinguished by hatching and named under each bar; each bar is annotated with its total time. The serial reference (depth-first search from the pit) is printed in each panel. As-late-as-possible layering is fastest for every kernel.

Figure 15 separates two sources of speedup for the upstream drainage area kernel: spreading a region across the four processors (the three lines) and threading within each partition (the x-axis). At one thread per partition there is no within-subregion threading, so these leftmost points isolate partitioning alone: with no partitioning the kernel is a plain serial run on one core (142.2 s, an independent run about 3.4% below the 147.2 s in Figure 13), whereas the four-processor partitions

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

run their four subregions concurrently, so balanced subbasin-level partitioning already cuts this to 66.7 s, while basin-level reaches only 112.5 s because its oversized dominant basin keeps one processor busy and the others idle. Threading with as-late-as-possible layering then drives every line down (after a small startup cost at four threads on the unpartitioned run), with subbasin-level fastest throughout and bottoming at 19.6 s at 13 threads per partition. The unpartitioned kernel scales only within its one processor, reaching 25.0 s at full occupancy but never using the other three. From 13 to 26 threads per partition the three nearly converge (25.0, 24.0, and 20.5 s at 26 threads): the partitioned curves flatten, each processor bandwidth-bound by about 13 threads (subbasin-level even rises slightly), while the unpartitioned run keeps improving on its single processor and catches up to them. The speedup therefore scales with the number of processors, not the number of threads: on this server each processor saturates its memory bandwidth near 13 threads, so balanced subbasin-level partitioning across all four processors, threaded to about 13 on each, is fastest at every thread count. The global fields these four kernels produce are mapped in Figures A5–A8, one per kernel in the same order: flow length downstream (Figure A5), flow length upstream (Figure A6), Strahler stream order (Figure A7), and upstream drainage area (Figure A8).

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

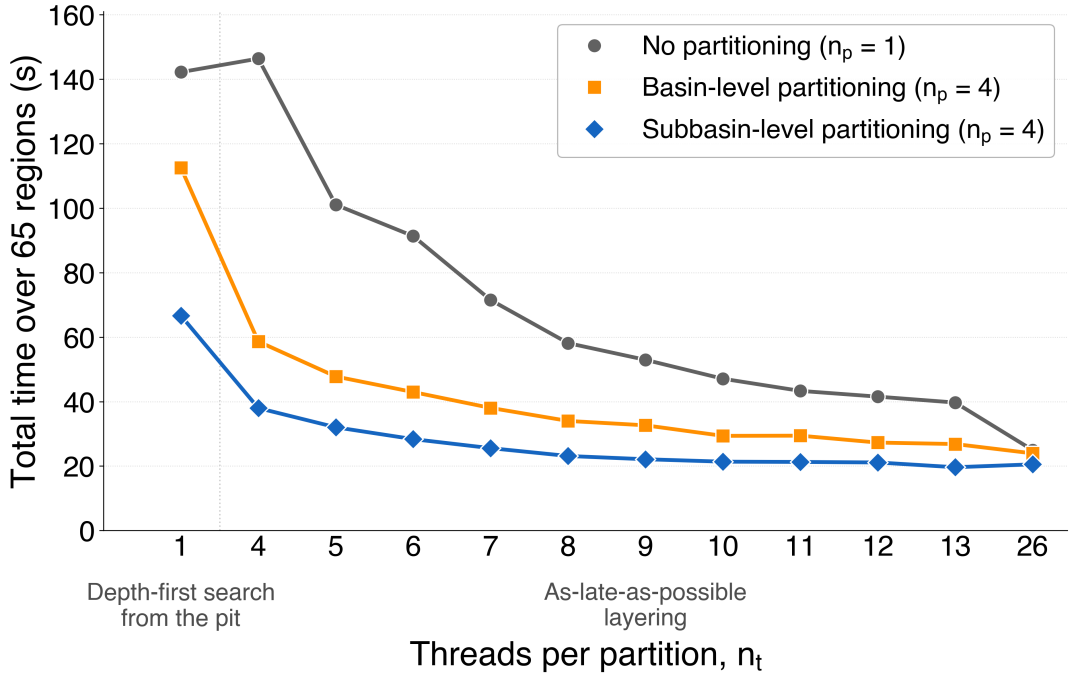


Figure 15. Thread scaling of the upstream drainage area kernel for three partitioning strategies, on a common axis of total wall time summed over the 65 benchmark regions against threads per partition: no partitioning (gray circles, confined to one processor), basin-level partitioning (orange squares), and subbasin-level partitioning (blue diamonds), the latter two splitting each region into four subregions, one per processor. At one thread per partition the kernel runs as the depth-first search from the pit serial sweep (a single-core run with no partitioning, four concurrent subregions otherwise), and at four or more threads as-late-as-possible layering parallelizes within each subregion. Each region's time is the mean of the three fastest of five runs, and each partitioned time is the slowest of its four subregion ranks. Only partitioning engages all four processors, and balanced subbasin-level partitioning is fastest at every thread count.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

6. Discussion

For hyperresolution hydrology, mapping the drainage network is no longer the hard part. Computing over it is one of the parts that remain, alongside the representation of the network itself and the hydraulic parameters it carries (Sect. 6.4). That cost lies not in the static flow network but in the processing order every computation in drainage order must rebuild, here across MERIT Hydro's 22.2 billion land pixels. Computed once and distributed in a representation matched to the demand, that order turns a repeated, machine-bound cost into a one-time read.

6.1 From implicit traversal to explicit reusable structures

The global hydrography datasets in widest use, HydroSHEDS (Lehner et al., 2008) and MERIT Hydro (Yamazaki et al., 2019), distribute the static inputs of flow routing: a conditioned elevation grid and a D8 flow-direction grid. Producing the quantities applications need, such as upstream drainage area, the stream network and its Strahler order, or watershed boundaries, first requires a processing order consistent with the drainage dependencies. That order is a topological ordering of the flow-direction grid, fixed by the static D8 field but distributed nowhere. Every order-based tool in Table 1 reconstructs it by recursive tracing, queue-based traversal, or repeated grid sweeps (TauDEM, GRASS GIS (Metz et al., 2011), RichDEM (Barnes et al., 2014; Barnes, 2017), PCRaster (Karssenberg et al., 2010), MATLAB TopoToolbox (Schwanghart & Scherler, 2014), pyflwdir (Eilander, 2020), ArcGIS). The recent analyses that advanced continental-scale hydrography each build on one of these engines and inherit the same step: MERIT Hydro river networks on TauDEM (Lin et al., 2019, 2021), Hydrography90m on GRASS GIS (Amatulli et al., 2022), a global drainage-shape dataset on TopoToolbox (He et al., 2024), and hydrography upscaling on pyflwdir (Eilander, 2020).

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

The dataset presented here is that order for MERIT Hydro, stored as explicit reusable structures: pixel-index layers for parallel processing and one flattened topologically ordered sequence for serial use. Computed once and reused, it changes downstream practice in four ways. (i) It removes the software barrier: the topological ordering is already done, so a user computes a flow-routing variable from a single linear pass over the ordered array, with no GIS engine. (ii) It lightens distribution: one order ships next to the direction and elevation grids, released as MERIT-FlowTopo, instead of a separate product for every derived variable (Amatulli et al., 2022) or the 4.1 TB of the 104-variable Environment90m (García-Márquez et al., 2026); users compute what they need on demand. (iii) It removes the compute barrier: building the order is the costly step, with a peak memory footprint of tens of gigabytes even for a single region and beyond any desktop at the global 90 m scale (Barnes, 2017; Cho, 2025b), and expensive even on a cluster, where the Hydrography90m suite took 12,418 core-hours (Amatulli et al., 2022). Reading the published order avoids that peak, holding only one copy of the index array, so a desktop that cannot build the order can still read it; in interpreted tools repeated access is cheaper still, with caching cutting 1,398-watershed reruns from 58 to 6 minutes (Xie et al., 2022). (iv) It can be clipped to any basin or region and used directly, with no re-ordering, so a study reads only its own area. Just as HydroSHEDS and MERIT Hydro made the elevation and flow-direction grids a shared foundation, the processing order is the next such layer, distributed not as another derived variable but as the structure from which any variable is computed, for the community to read rather than reconstruct.

6.2 Choosing a representation, and the baseline it is measured against

Across terrain-analysis software, flow-routing models, landscape-evolution codes, and algorithmic studies, the single-receiver D8 dependency is resolved by developer convention rather than systematic comparison; Table 1 organizes the options by granularity. Several routing models avoid a per-pixel order altogether, advancing each pixel from the previous timestep's value or summing

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

upstream contributions by convolution, so the within-step update order is immaterial (CaMa-Flood, Yamazaki et al., 2011, among the order-free models in Table 1). Every other approach first arranges the pixels into a processing order, serially from the headwaters downward or depth-first from the pit, and in parallel groups them into concurrent layers built from the headwaters (as-soon-as-possible layering) or from the outlet (as-late-as-possible layering). These layerings have been compared only in part: Jiang et al. (2025a) found as-late-as-possible layering faster and attributed the gain to load balance, and Kotyra et al. (2021) compared six flow-accumulation implementations, pitting pull by recursion from the outlets against atomic push scheduled by in-degree counting and finding the latter fastest; yet each held the kernel fixed and varied a single axis. At the largest scale the domain is split across machines by MPI or staged along the drainage hierarchy (Table 1). We are not aware of a prior study that spans this design space on a common network and kernel set; we do so here.

We recommend matching the representation to the computational demand. On a single core the depth-first search from the pit is the fastest of the three serial orderings across all four kernels, because its subtree contiguity holds each pixel's downstream dependency within a few cache-resident traversal steps. It is therefore also the baseline against which parallel speedup must be measured: a slower topological-sort-from-the-sources reference would inflate the same speedup into an apparent superlinear result, crediting to parallelism a gain that originates in memory locality. Parallel execution becomes worthwhile only when the network is traversed repeatedly, as in calibration or ensemble simulation; as-late-as-possible layering then distributes work evenly across threads and keeps each basin's working set cache-resident, the advantage reported regionally by Jiang et al. (2025a) and confirmed here at global scale. Additional threads yield diminishing

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

returns once a single dominant basin governs runtime: its mainstem is the longest dependency chain, and left undivided it serializes work the other threads would share, a limit set by the critical path rather than memory capacity. Partitioning that basin into subbasins across the processors removes most of the imbalance within one machine's shared memory, without message passing. Even then the ceiling shifts rather than disappears: with so little arithmetic per pixel, each processor's memory bandwidth saturates well before all of its cores are in use, so further threads on a processor stop helping and the speedup tracks the number of processors rather than the threads on each. The fastest configuration is therefore all available processors, each threaded to the point at which its memory bandwidth saturates. Where that point lies depends on the processor, on the kernel's arithmetic intensity, and on how much of the traversal stays in cache; Sect. 5.4 reports it for the server tested, and it has to be measured anew on other hardware. Multiple machines are required only when turnaround is the binding constraint, as in operational forecasting; because complete basins are independent, they distribute with negligible communication. Distributed-memory MPI with halo exchange, which basin partitioning does not require, is reserved for the case in which a coupled host model dictates the partition. When routing shares an Earth system model's MPI-decomposed grid, as in MOSART within CESM (Li et al., 2013), E3SM's shared Voronoi mesh (Feng et al., 2025), or WRF-Hydro's inherited atmospheric domain (Gochis et al., 2018), tile boundaries are fixed externally and downstream neighbors may reside on different ranks, so topological reordering cannot avoid halo exchange at each timestep.

6.3 Propagation manner and lock-free concurrency

Propagating a value along the network takes one of two directions: a pixel can push its value onto the receiver it drains into, or pull the values it needs from the pixels that drain into it. In serial execution the manner follows each kernel's natural direction and affects only storage, not correctness. It becomes consequential only in parallel, and only for many-to-one propagation,

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

where several upstream pixels drain into one receiver and their writes collide; one-to-one propagation such as flow length downstream writes each pixel once and parallelizes trivially either way. Existing parallel implementations resolve the collision two ways: by pull, where each pixel gathers from its donors and writes only itself, removing the conflict but, in the implementations evaluated here, requiring a stored upstream donor structure, or by atomic push, where each donor adds to its receiver through a hardware atomic (Kotyra et al., 2021). Atomic push carries three costs: contending writes are serialized; the resolution work grows with the confluence operator, from a single addition for drainage area to a compound update for Strahler stream order; and floating-point sums become non-reproducible, since addition is non-associative and thread arrival order is unspecified, so sums vary across runs while integer pixel counts stay reproducible. In the implementations surveyed in Table 1 the collision is avoided by pulling or absorbed by atomics; neither removes it by scheduling, so none of them offers a manner that is at once a push, free of atomics, and deterministic.

The conflict-free downstream layering introduced here fills that gap: it schedules each layer such that no two concurrent pixels share a receiver, so a push writes to the receiver yet never collides. Without atomics it stays deterministic and resolves even the non-linear confluence rule of Strahler stream order which no atomic addition can express, while storing only the receiver pointer. The choice between the two atomic-free manners is a trade-off between RAM and cache. Pull with as-late-as-possible layering gives the best cache locality, its peak per-layer footprint staying resident in the L3 (Sect. 5.2), and the fastest run overall, but, as implemented here, it must materialize the donor structure as a per-pixel donor list or compressed sparse row index, a cost on a dense grid beyond the single receiver pointer that push requires. Conflict-free downstream push spares that

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

structure, making it RAM-economical, deterministic, and able to express Strahler stream order, but it is front-loaded, so a single layer can span about 2.4 GB and overflow the L3, weakening its cache locality. The binding resource therefore decides. When many state variables or ensemble members saturate RAM, each carrying its own per-pixel state array, the donor structure that pull requires cannot fit, and conflict-free downstream push is the feasible choice of the two atomic-free manners evaluated here, the regime of spatial-temporal-multimember scheduling (Jiang et al., 2025a) that already holds routing state for many steps and members at once. When the donor structure fits and cache throughput dominates, as in single-variable static-topology products, as-late-as-possible pull wins.

A different approach recasts accumulation as an implicit sparse linear system solved iteratively (Richardson et al., 2014), scaling to many processors without materializing an explicit order. It resolves the ordering implicitly rather than removing it, but because the formulation is linear it computes additive quantities such as drainage area while leaving the conditional, non-linear update of Strahler stream order beyond reach; a single pass over an explicit, reusable order resolves both kinds of kernel alike.

6.4 Implications and outlook

The four kernels are at once the most widely used variables in hydrology and a representative span of the dependency patterns of single-receiver flow-network computation. Flow length downstream is the one-to-one case, the pattern of steady backwater and subcritical-flow profiles solved upstream from a downstream boundary, and of reverse flood routing (Pati et al., 2023), for which as-late-as-possible layering, anchored at the outlet, preserves the physical order of solution; with the flow length upstream it sets the travel distances behind time of concentration and the unit hydrograph (Hu et al., 2024). Upstream drainage area and the flow length upstream are the many-

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

to-one case, each gathering contributions from all upstream pixels at every confluence. These underlie flow accumulation itself, the most optimized target in terrain analysis (Barnes, 2017), and the transport of real quantities along the network: discharge in floods together with the heat, solutes, and sediment that set water temperature and quality. These are the processes the field is now pushing to 1 km and finer (Nguyen-Quang et al., 2018; Yang et al., 2025). Strahler stream order (Strahler, 1957), which classifies the network for hydraulic-geometry and scaling analyses, is the non-linear case, where the update is conditional rather than additive and push resolves it without atomics only under conflict-free downstream layering, among the layerings evaluated here. Upstream drainage area is therefore not only a static map but the elementary upstream-to-downstream step of routing on a fixed single-receiver network: in CaMa-Flood, for instance, every timestep sums each cell's outflow into its downstream cell's inflow in exactly this many-to-one pattern (Yamazaki et al., 2011). The framework that accelerates this step accelerates that part of the simulations built on it. Fully dynamic routing is not reducible to this step. The outflow being summed is computed from the water-surface slope between a cell and its downstream cell, so under backwater a reach depends on the downstream water level as well as on upstream inflow; the sign of that flow can reverse; and distributary channels connect cells outside the single-receiver network. These dependencies run in both directions, which is why such models update every cell from the previous timestep's state rather than in drainage order (Table 1). For them the precomputed order fixes the traversal of the one-way accumulation, and the two-way hydraulic coupling remains a cost that a static accumulation benchmark does not measure.

The nearest frontier of this work is real-world dynamic and coupled routing (Chen et al., 2022). The kernels here run once, but routing a moving state in drainage order repeats the same upstream-

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

to-downstream pass thousands of times, and adds whatever hydraulic coupling the scheme carries on top of it. A ten-year daily routing of the single Pearl River basin already takes about 22 hours on one core and close to two on thirteen (Jiang et al., 2025a), so a global, multi-decade, or ensemble run of the kind operational systems and large-sample studies require (Jiang et al., 2025b; Mizukami et al., 2016; Wu et al., 2019) lies far beyond one sweep. Routing several constituents together compounds the demand and brings the RAM-versus-cache trade-off of Sect. 6.3 to bear, with the full coupled-routing benchmark left to future work. The hardware path is throughput computing: flow-network workloads are moved to GPUs for flow accumulation (Qin & Zhan, 2012; Sten et al., 2016), watershed delineation (Kotyra, 2023), the longest flow path (Kotyra, 2025), and real-time ensemble flood inundation (Wei et al., 2023). The conflict-free downstream layering suits them, exposing broad within-layer concurrency and removing write conflicts by construction, though efficient use will require layer layouts that coalesce memory access. Underpinning both is a reusable, deterministic topological order, computed once and distributed with the flow-direction grid, that any such computation reads instead of rebuilding, so that as resolutions rise and ensembles grow these gains accumulate across models and runs rather than being recomputed for each.

7. Conclusion

This study presents six reusable topological representations (three serial orderings and three parallel layerings) and two domain partitions, derived once from the static D8 flow-direction field and benchmarked on the 90 m MERIT Hydro network of 22.2 billion land pixels across 65 continental regions. The systematic comparison shows that how the computation is organized governs its cost as much as the hardware it runs on. In serial, depth-first ordering runs up to 5.1-fold faster than the slowest serial order, because its subtree contiguity keeps each pixel's

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

downstream dependency cache-resident, lowering the L3 miss rate from 37.4% to about 6%. In parallel, as-late-as-possible layering is fastest, keeping each layer's working set within the shared cache, whereas conflict-free downstream layering is, among the schemes evaluated here, the one that allows atomic-free push updates for non-linear kernels such as Strahler stream order, and it suits throughput architectures such as GPUs. A communication-free subbasin-level partition balances the load across processors without message passing, scaling upstream drainage area 7.3-fold on the four-processor server tested.

These findings reduce to a simple guide: a single sweep over the network needs only depth-first ordering; repeated traversal, as in calibration or ensembles, is best served by as-late-as-possible layering; a deterministic push on non-linear kernels calls for conflict-free downstream layering; and the largest basins are handled by subbasin-level partitioning, run across all available processors, each threaded only to the point at which its memory bandwidth saturates, since beyond it the bandwidth rather than the algorithm bounds the speedup. Because the D8 field does not change, each structure is computed once and reused without limit, and all are released openly: the recommended structures as MERIT-FlowTopo, with all eight for one region, and the kernel products as MERIT-DrainAttr (flow length downstream, flow length upstream, and Strahler stream order for every region, upstream drainage area for Region 43). As global hydrography advances to finer resolutions, distributing such precomputed topology alongside each new flow-direction dataset would spare the field the redundant preprocessing that now burdens every study, establishing explicit topological organization as shared infrastructure for the next generation of hyperresolution hydrological computation.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

Appendix A

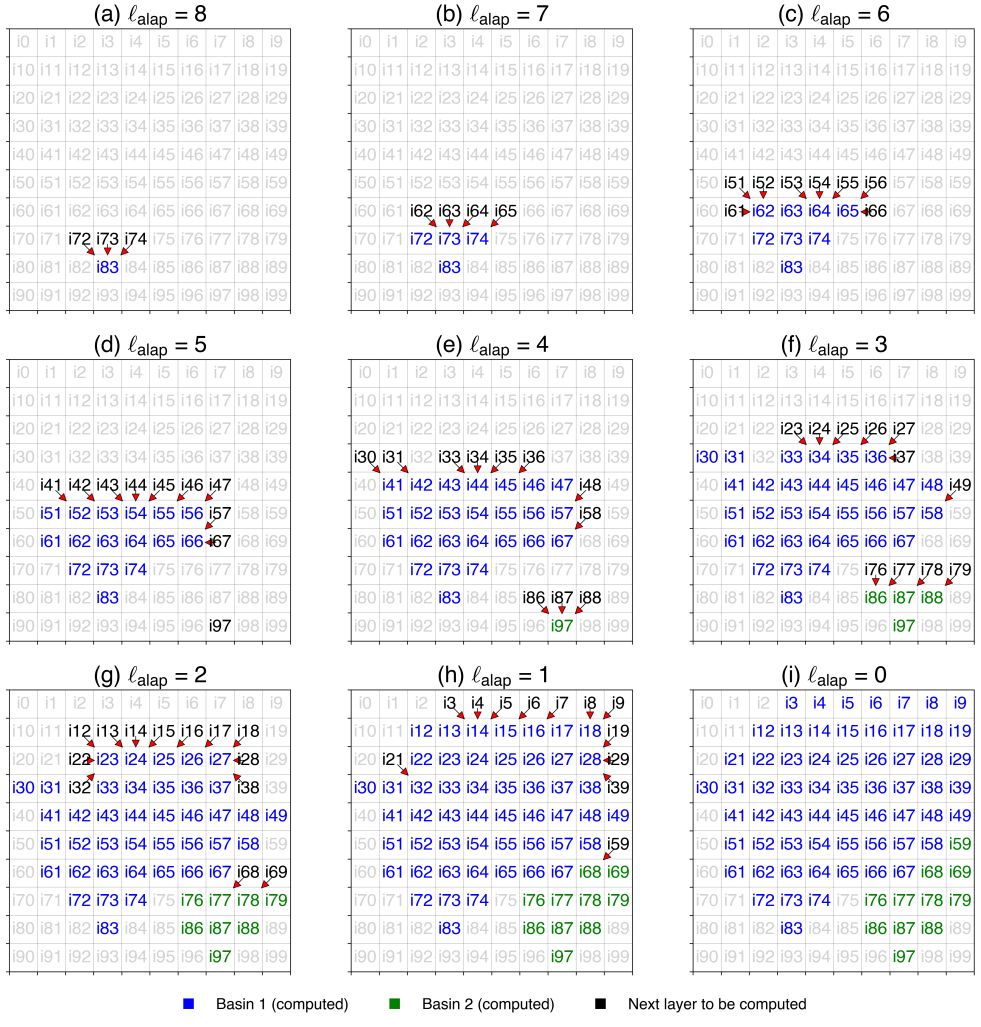


Figure A1. Step-by-step flow-length-downstream computation on the two-basin example using as-late-as-possible layering, processed downstream-to-upstream, one layer per panel from the outlet layer (a) to the headwater layer (i). Each pixel is labeled by its linear index; computed pixels are colored by basin once their distance is finalized. Each pixel takes the distance already finalized at its receiver; basin 2 begins at its own outlet and propagates independently. Pixels of the next layer are black, with arrows to their receivers.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

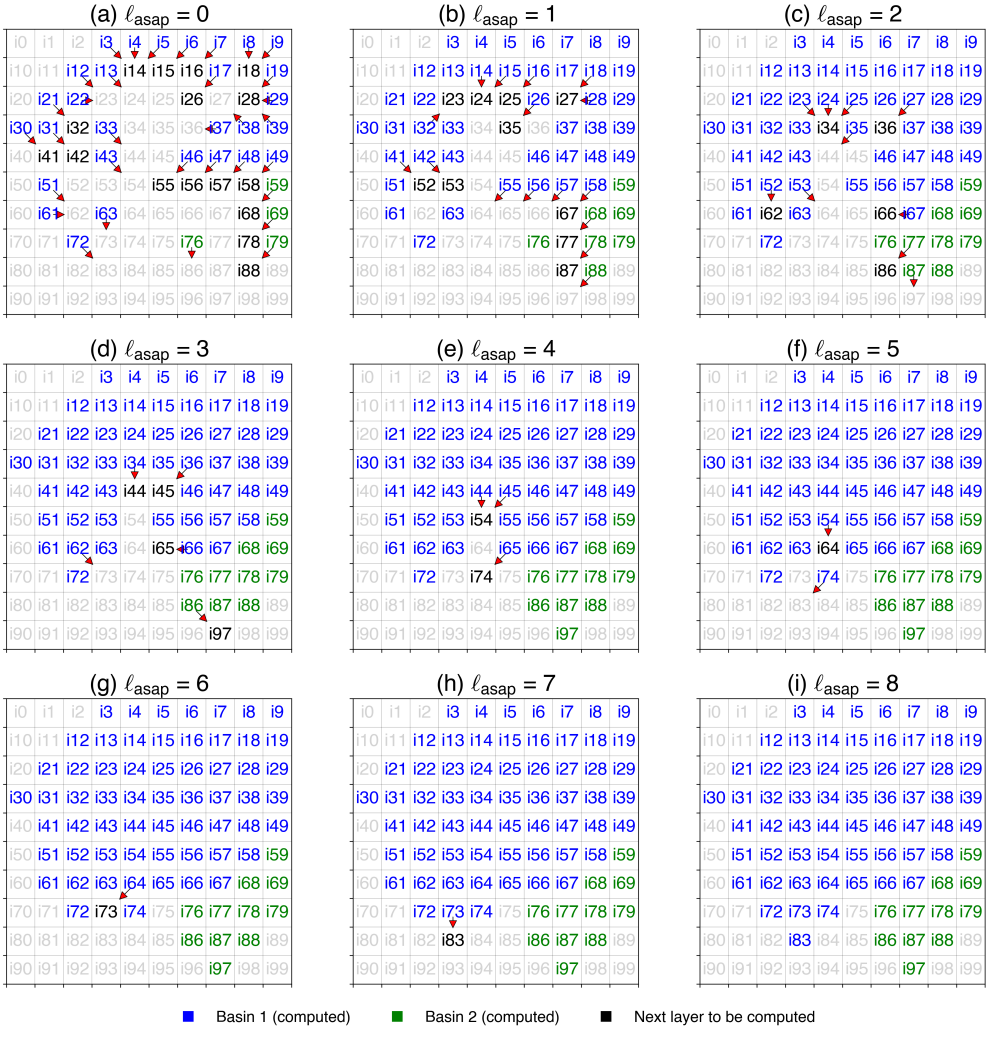


Figure A2. Step-by-step flow-length-upstream computation on as-soon-as-possible layering for the two-basin example, one layer per panel (layer 0 at the sources to the maximum layer at the pit), processed upstream-to-downstream. Each pixel is labeled by its linear index; computed pixels are colored by basin and the next layer is black. Red arrows on the current layer point to each pixel's receiver; a pixel takes the maximum over its donors of the donor value plus the donor-to-pixel distance.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



Figure A3. Step-by-step Strahler-order computation on conflict-free downstream layering for the two-basin example, one layer per panel processed upstream-to-downstream. Stream pixels (upstream pixel count above the threshold) are labeled by their linear index, with background shading giving the current order (1, 2, 3). At each confluence the downstream pixel takes the maximum upstream order, or one greater when two or more tributaries share that maximum. Computed pixels are colored by basin; pixels of the next layer are black, with arrows to their receivers.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

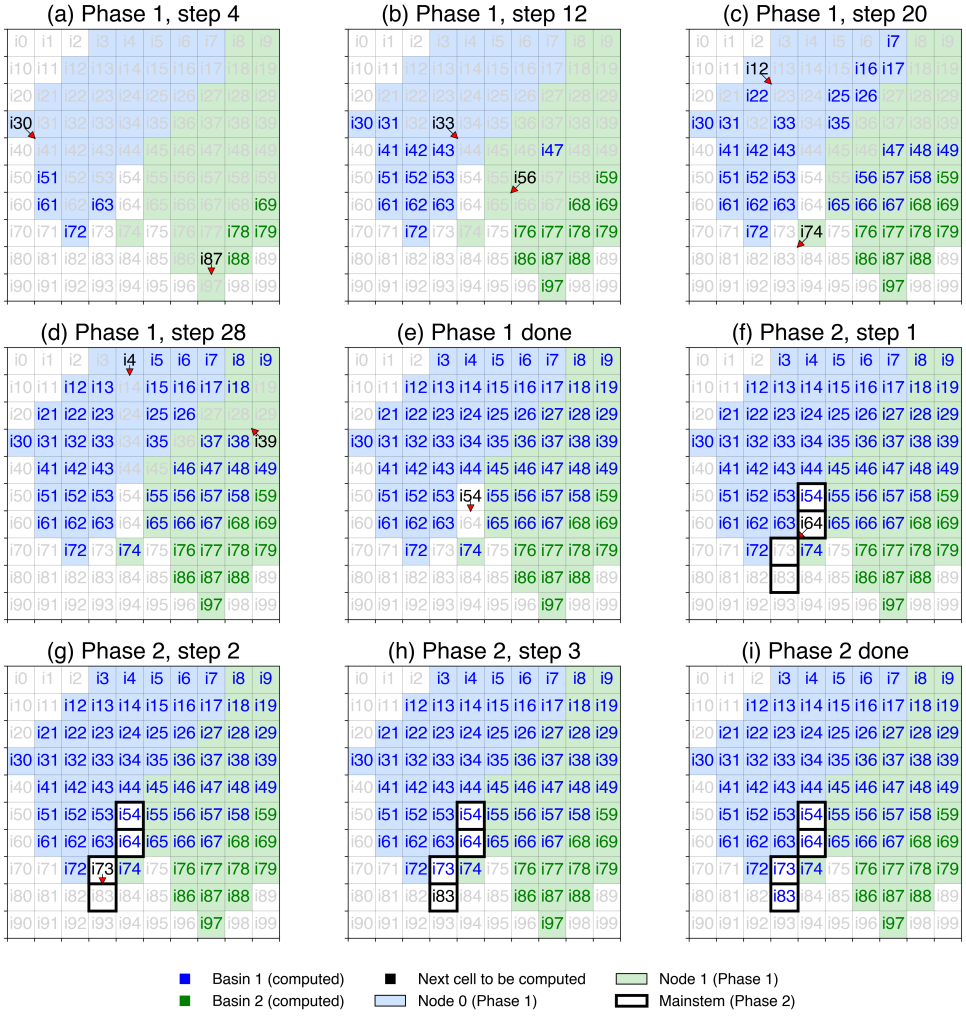


Figure A4. Step-by-step upstream-drainage-area computation under the subbasin-level partitioning scheme, in nine snapshots; each pixel is labeled by its linear index and processed in each node's upstream-to-downstream order. (a–e) Phase 1 accumulates within each node (basin 1 / node 0, light blue; basin 2 / node 1, light green) up to the shared mainstem. (f–i) Phase 2 walks the mainstem to combine the node partial sums into the final area; mainstem pixels carry a thick black border. The next pixel to be computed is black, with an arrow to its receiver.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

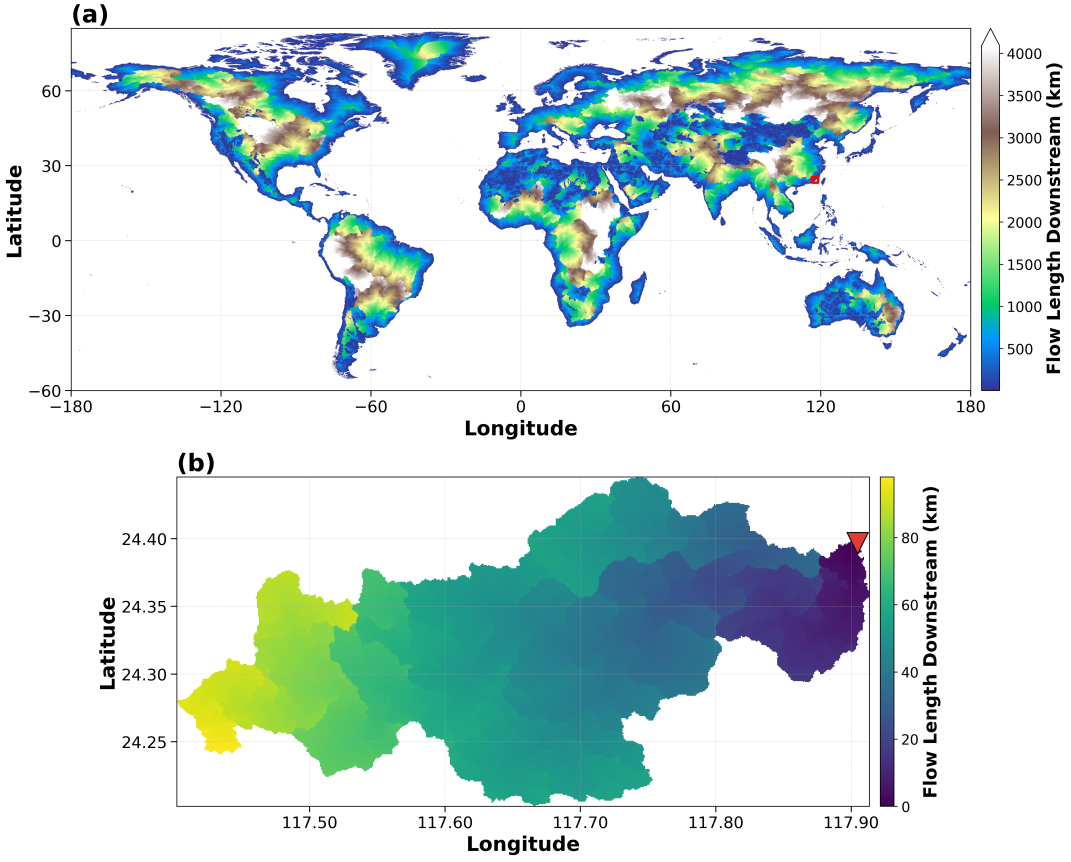


Figure A5. Flow length downstream (km). (a) Global map; a red rectangle marks the example basin. (b) Zoom of the example basin; the outlet is a red downward triangle.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

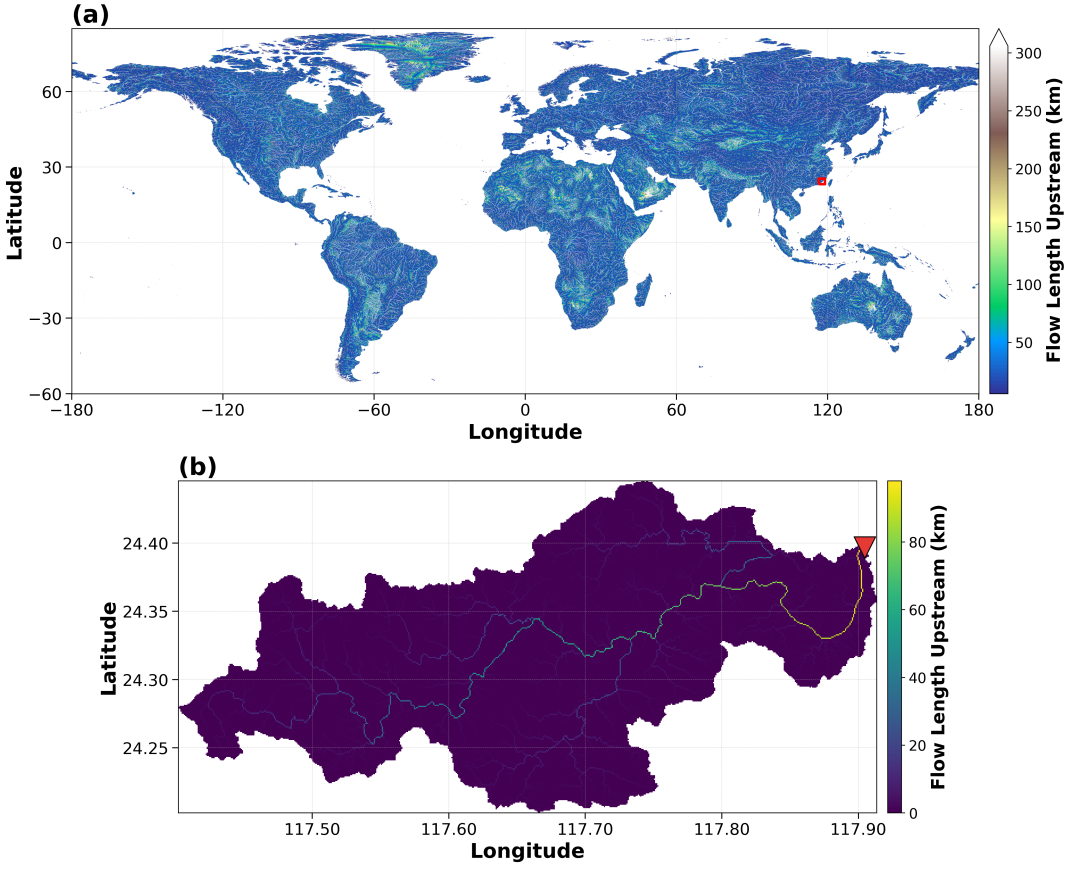


Figure A6. Flow length upstream (km). (a) Global map; a red rectangle marks the example basin. (b) Zoom of the example basin; the outlet is a red downward triangle.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

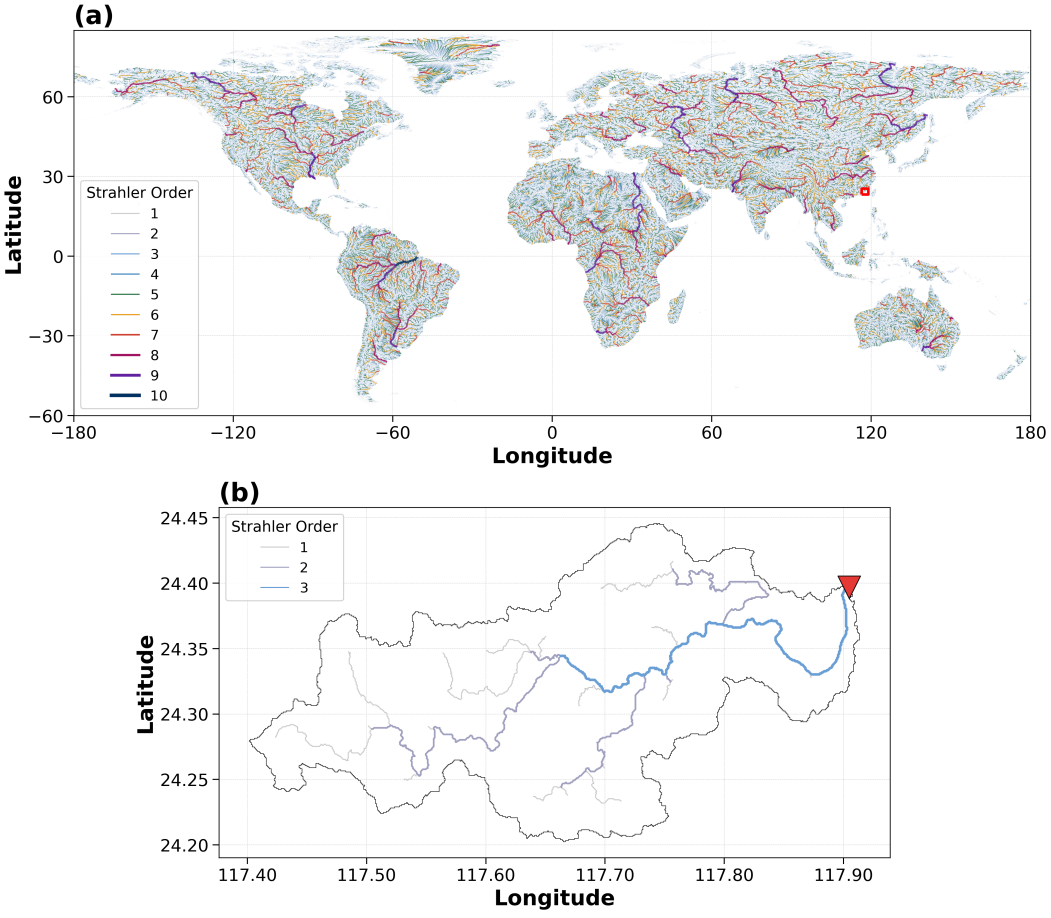


Figure A7. Strahler stream order along the drainage network, at (a) global and (b) example-basin scales. Line color encodes order from a categorical palette and line width increases with order; on the global map opacity also increases with order and a red rectangle marks the example basin. The outlet in (b) is a red downward triangle. (The global network is extracted at an upstream-drainage-area threshold of 10 km².)

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

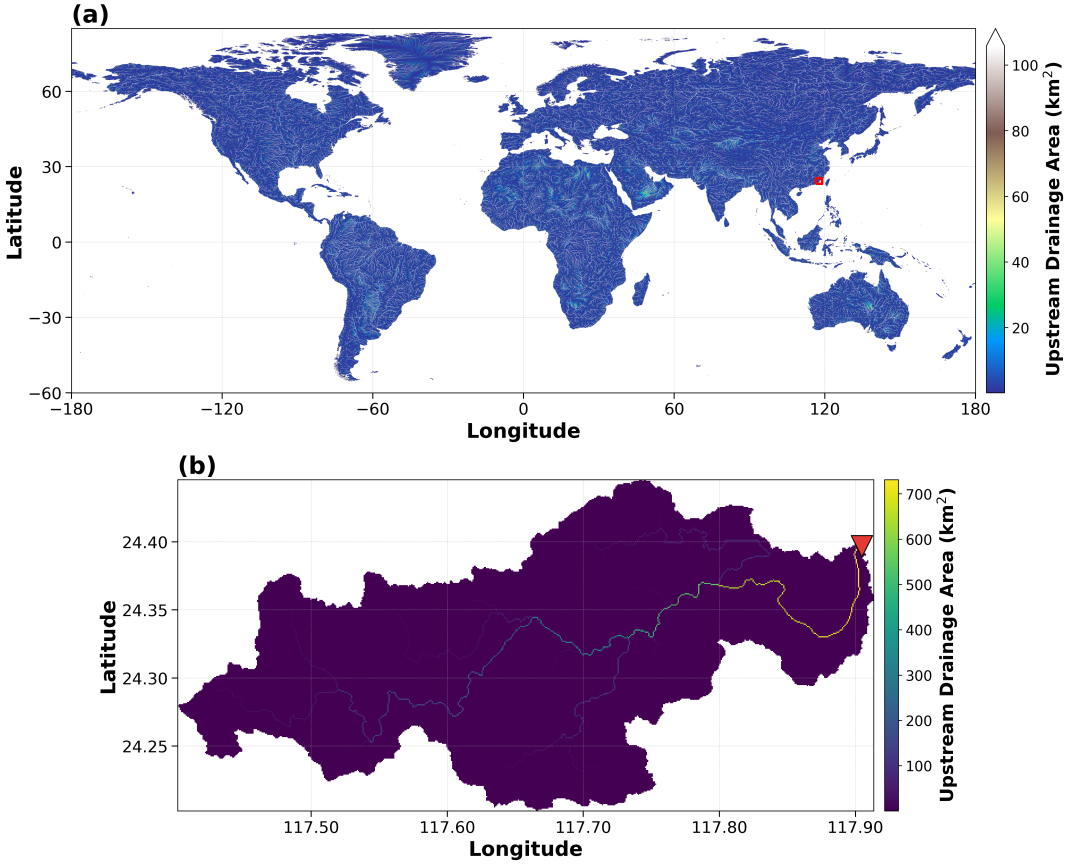


Figure A8. Upstream drainage area (km²). (a) Global map; a red rectangle marks the example basin. (b) Zoom of the example basin; the outlet is a red downward triangle.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

Appendix B

Table B1. Memory-locality metrics reported for the topological representations, evaluated on Region 43 (about 609 million pixels). A sequence visits the N valid pixels in order $p_1, \ldots, p_N$; $r(i)$ is the receiver of pixel i; W is the grid width; seq[i] is the position of pixel i in the visit order; ⌊p/W⌋ is the raster row of pixel p; a value occupies $b = 4$ bytes and a 64-byte line holds $c = 16$ pixels, so $\kappa(p) = \lfloor p \cdot b/64 \rfloor$ is the cache line of pixel p. Within a layer, its m pixels are listed in ascending index order $q_1 < \ldots < q_m$, with gap $g_j = q_{j+1} - q_j$. $\mathbb{1}(\cdot)$ is the indicator function. Sequences are stored upstream-to-downstream, so a receiver follows its donor and D ⩾ 1 for every pixel but the outlets, which point to themselves.

Metric	Family	Definition	Formula	No.
Dependency distance	Serial	Visit-order distance from a pixel to its receiver (shown as a cumulative distribution, Figure 8a)	$D_i = seq[r(i)] - seq[i]$	(B1)
Row-jump fraction	Serial	Share of consecutive visits that cross a raster row (Figure 8b)	$RJ = \frac{1}{N-1}\sum_{k=1}^{N-1} \mathbb{1}\left(\left\lfloor \frac{p_k}{W} \right\rfloor \neq \left\lfloor \frac{p_{k+1}}{W} \right\rfloor\right)$	(B2)
Miss rate (L1, L2, L3) [a]	Serial & parallel	Share of cache-line reads not resident, under set-associative LRU (Figures 8c and 10c)	$m_L = \frac{\text{misses}_L}{\text{accesses}_L}$	(B3)
Cache-line footprint	Parallel	Largest per-layer count of distinct cache lines touched (Figure 10b)	$F = \max_{\text{layer}} \lvert\{\kappa(q): q \in \text{layer}\}\rvert$	(B4)

[a] Miss rates come from a set-associative least-recently-used simulation (Mattson et al., 1970) of the Intel Xeon Platinum 8270 caches (64-byte lines; L1, 512 lines, 8-way; L2, 16,384 lines, 16-way; L3, 585,728 lines, 11-way). Visiting the pixels in visitation order (for a serial sequence, its ordering; for a parallel layering, the layers concatenated in order, each in ascending index): each pixel issues two reads, one at its own index and one at its receiver's index, with the two arrays placed in separate address ranges so their cache lines do not alias; the simulation reports the share of reads not already resident. Simulating the caches rather than reading hardware performance counters isolates the effect of the visiting order from the hardware prefetcher and from contention on the shared platform.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

Code availability

The C sources that produced all timings in this paper are archived as MERIT-FlowTopo code v1.0 on Zenodo (Jiang et al., 2026d; https://doi.org/10.5281/zenodo.22227621) under the MIT license; the concept DOI https://doi.org/10.5281/zenodo.22227620 always resolves to the latest version of this record. The archive holds the four kernels under every propagation manner, the builders of the six ordering structures, the cache simulation behind Figures 8 and 10, and a driver that reproduces the full method on the example basin of Figure 6c with no dependency beyond a C99 compiler and OpenMP. The run checks every combination against the serial depth-first reference and reports PASS or FAIL. The example basin bundled with the code is an excerpt of MERIT Hydro and is redistributed under MERIT Hydro's own terms (CC BY-NC 4.0, with attribution). The same record holds the input data and the plotting script for every figure of the paper, Figures 1–15 and A1–A8, which regenerate the figures as printed with a single command.

A Python implementation of the same method is maintained at https://github.com/hellolulujiang/flowtopo under the MIT license, and the version referred to here is archived alongside the C sources (Jiang et al., 2026d). It is not a wrapper around the C code: the two were written independently and agree on the example basin to floating-point rounding in the accumulated area. It is slower than the C version and is not the code behind the reported timings.

Data availability

The flow-direction field used as input is MERIT Hydro (Yamazaki et al., 2019), available at https://global-hydrodynamics.github.io/MERIT_Hydro/. Its partition into the 65 continental regions analyzed here is provided as part of the MERIT-FullBasin dataset (Jiang et al., 2026a; https://doi.org/10.5281/zenodo.20344113).

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

From this field, the study releases two products as separate Zenodo records, each stored as per-region GeoTIFFs compressed with ZSTD (level 19) for the 65 hydrologically independent regions (Table 2). MERIT-FlowTopo is derived directly from the MERIT Hydro flow-direction field and carries its terms (ODbL 1.0, with attribution and share-alike); MERIT-DrainAttr is released under CC BY 4.0. The licence of each product is stated on its record.

Table 2. Global products released for the 65 main regions of the 90 m drainage network, as two Zenodo records. MERIT-FlowTopo holds the reusable traversal structures. MERIT-DrainAttr holds the four baseline drainage variables (Figures A5–A8). Globally, MERIT-FlowTopo releases the four recommended structures (the depth-first search from the pit serial sequence, the conflict-free downstream and as-late-as-possible layerings, and the subbasin-level partition); Region 43 additionally includes the four alternatives, providing the full set of eight in one region for comparison. MERIT-DrainAttr releases flow length downstream, flow length upstream, and Strahler stream order for every region, whereas upstream drainage area is provided for Region 43 only.

File Name	Variable	Type	NoData	Unit	Figure
MERIT-FlowTopo					
seq_topo_region*_90m.tif	Topological sort from the sources, sequence index	Int32	-1	-	Figure 7a
seq_bfs_region*_90m.tif	Breadth-first search from the pit, sequence index	Int32	-1	-	Figure 7b
seq_dfs_region*_90m.tif	Depth-first search from the pit, sequence index	Int32	-1	-	Figure 7c
lyr_asap_region*_90m.tif	As-soon-as-possible layer index	Int32	-1	-	Figure 9a
lyr_cfds_region*_90m.tif	Conflict-free downstream layer index	Int32	-1	-	Figure 9b
lyr_alap_region*_90m.tif	As-late-as-possible layer index	Int32	-1	-	Figure 9c
part_basin_region*_90m.tif	Basin-level partition ID	UInt16	65535	-	Figure 11a

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

part_subbasin_region*_90m.tif	Subbasin-level partition ID	UInt16	65535	-	Figure 11c
MERIT-DrainAttr					
ldn_region*_90m.tif	Flow length downstream	Float32	-9999	km	Figure A5
lup_region*_90m.tif	Flow length upstream	Float32	-9999	km	Figure A6
ord_region*_90m.tif	Strahler stream order	UInt8	0	-	Figure A7
upa_region*_90m.tif	Upstream drainage area	Float32	-9999	km²	Figure A8

MERIT-FlowTopo v1.0 (Jiang et al., 2026b; https://doi.org/10.5281/zenodo.20653059) holds the reusable traversal structures. For every region it provides the four recommended structures: the depth-first search from the pit serial sequence, the conflict-free downstream and as-late-as-possible parallel layerings, and the subbasin-level partition. For Region 43 it additionally provides the four alternatives (the topological sort from the sources and breadth-first search from the pit serial sequences, the as-soon-as-possible layering, and the basin-level partition), so the full set of eight is available in one region for comparison. It is about 978 GB uncompressed and 50 GB compressed.

MERIT-DrainAttr v1.0 (Jiang et al., 2026c; https://doi.org/10.5281/zenodo.20686665) holds baseline variables computed from those structures: for every region, flow length downstream, flow length upstream, and Strahler stream order; upstream drainage area is provided for Region 43 only, as it is already distributed globally with MERIT Hydro. It is about 622 GB uncompressed and 60 GB compressed.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

Computing with MERIT-FlowTopo requires pairing these structures with the network's pixel-to-receiver connectivity, the D8 flow-direction code that they traverse. Because that field is a MERIT Hydro product, we do not redistribute it; readers download it from the MERIT Hydro archive (Yamazaki et al., 2019). Every released raster is a GeoTIFF on the MERIT Hydro grid itself, 1/1200° with the global origin at (−180°, 85°), already cropped to its region's window, so cropping MERIT Hydro to that same window makes the two line up pixel for pixel. For readers working region by region before downloading, https://fullhydro.org/fullbasin/ lists the MERIT Hydro tiles covering each region together with its row and column offset, and offers the 96 region outlines as a single GeoPackage. Region boxes are aligned to whole degrees and one degree is exactly 1,200 cells at 3 arcsec, so the crop is integer arithmetic with no resampling.

The input data and plotting scripts for every figure in the paper (Figures 1–15 and A1–A8) are archived with the code (Jiang et al., 2026d), and a narrated video overview of the study, which animates the six ordering structures together with the three propagation manners, is available on YouTube at https://youtu.be/tE5K2wM3TTY.

Author contributions

Lulu Jiang: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Huan Wu:** Conceptualization, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Weitian Chen:** Writing – review & editing. **Zhijun Huang:** Writing – review & editing. **Dai Yamazaki:** Resources, Writing – review & editing. **Ting Yang:** Writing – review & editing. **Lingfeng Li:** Writing – review & editing.

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

Competing interests

The contact author has declared that none of the authors has any competing interests.

Acknowledgements

Claude (Anthropic) was used for language editing, file organization, documentation, and plotting scripts; all methods, analyses, core computational code, and conclusions were designed, written, and verified by the authors.

Financial support

This research has been supported by the National Natural Science Foundation of China (grant nos. 42275019 and 42088101), the National Key R&D Program of China (grant no. 2024YFC3013302), the Open Fund Project for Heavy Rain of the China Meteorological Administration (grant no. BYKJ2024Z10), the Hainan Provincial R&D Program (grant nos. CXFZ2022J074 and SCSF202203), and the Hainan Key Research and Development Project (grant no. ZDYF2023SHFZ125).

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

References

Alfieri, L., Burek, P., Dutra, E., Krzeminski, B., Muraro, D., Thielen, J., & Pappenberger, F. (2013). GloFAS – global ensemble streamflow forecasting and flood early warning. *Hydrology and Earth System Sciences*, *17*(3), 1161–1175. https://doi.org/10.5194/hess-17-1161-2013

Allen, G. H., & Pavelsky, T. M. (2018). Global extent of rivers and streams. *Science*, *361*(6402), 585–588. https://doi.org/10.1126/science.aat0636

Amatulli, G., Garcia Marquez, J., Sethi, T., Kiesel, J., Grigoropoulou, A., Üblacker, M. M., Shen, L. Q., & Domisch, S. (2022). Hydrography90m: a new high-resolution global hydrographic dataset. *Earth System Science Data*, *14*(10), 4525–4550. https://doi.org/10.5194/essd-14-4525-2022

Avesani, D., Galletti, A., Piccolroaz, S., Bellin, A., & Majone, B. (2021). A dual-layer MPI continuous large-scale hydrological model including Human Systems. *Environmental Modelling & Software*, *139*, 105003. https://doi.org/10.1016/j.envsoft.2021.105003

Barnes, R., Lehman, C., & Mulla, D. (2014). Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. *Computers & Geosciences*, *62*, 117–127. https://doi.org/10.1016/j.cageo.2013.04.024

Barnes, R. (2017). Parallel non-divergent flow accumulation for trillion cell digital elevation models on desktops or clusters. *Environmental Modelling & Software*, *92*, 202–212. https://doi.org/10.1016/j.envsoft.2017.02.022

Barnes, R. (2019). Accelerating a fluvial incision and landscape evolution model with parallelism. *Geomorphology*, *330*, 28–39. https://doi.org/10.1016/j.geomorph.2019.01.002

Bartos, M. (2020). *pysheds: Simple and intuitive watershed delineation in Python* [Software]. Zenodo. https://doi.org/10.5281/zenodo.3822494

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



988 Beighley, R. E., Eggert, K. G., Dunne, T., He, Y., Gummadi, V., & Verdin, K. L. (2009).
989 Simulating hydrologic and hydraulic processes throughout the Amazon River Basin. *Hydrological*
990 *Processes*, *23*(8), 1221–1235. https://doi.org/10.1002/hyp.7252
991 Beven, K. J., & Cloke, H. L. (2012). Comment on "Hyperresolution global land surface modeling:
992 Meeting a grand challenge for monitoring Earth's terrestrial water" by Eric F. Wood et al. *Water*
993 *Resources Research*, *48*(1), 2011WR010982. https://doi.org/10.1029/2011wr010982
994 Bieger, K., Arnold, J. G., Rathjens, H., White, M. J., Bosch, D. D., Allen, P. M., Volk, M., &
995 Srinivasan, R. (2017). Introduction to SWAT+, A Completely Restructured Version of the Soil
996 and Water Assessment Tool. *JAWRA Journal of the American Water Resources Association*, *53*(1),
997 115–130. https://doi.org/10.1111/1752-1688.12482
998 Bierkens, M. F. P., Bell, V. A., Burek, P., Chaney, N., Condon, L. E., David, C. H., de Roo, A.,
999 Döll, P., Drost, N., Famiglietti, J. S., Flörke, M., Gochis, D. J., Houser, P., Hut, R., Keune, J.,
1000 Kollet, S., Maxwell, R. M., Reager, J. T., Samaniego, L., Sudicky, E., Sutanudjaja, E. H., van de
1001 Giesen, N., Winsemius, H., & Wood, E. F. (2015). Hyper-resolution global hydrological modelling:
1002 what is next?. *Hydrological Processes*, *29*(2), 310–320. https://doi.org/10.1002/hyp.10391
1003 Braun, J., & Willett, S. D. (2013). A very efficient O(n), implicit and parallel method to solve the
1004 stream power equation governing fluvial incision and landscape evolution. *Geomorphology*, *180-*
1005 *181*, 170–179. https://doi.org/10.1016/j.geomorph.2012.10.008
1006 Chen, W., Wu, H., Kimball, J. S., Alfieri, L., Nanding, N., Li, X., Jiang, L., Wu, W., Tao, Y., Zhao,
1007 S., & Zhong, W. (2022). A coupled river basin-urban hydrological model (DRIVE-Urban) for real-
1008 time urban flood modeling. *Water Resources Research,* *58*(11), e2021WR031709.
1009 https://doi.org/10.1029/2021WR031709

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



Cho, H. (2023). Memory-efficient flow accumulation using a look-around approach and its OpenMP parallelization. *Environmental Modelling & Software*, *167*, 105771. https://doi.org/10.1016/j.envsoft.2023.105771

Cho, H. (2025a). Avoid backtracking and burn your inputs: CONUS-scale watershed delineation using OpenMP. *Environmental Modelling & Software, 183*, 106244. https://doi.org/10.1016/j.envsoft.2024.106244

Cho, H. (2025b). Loop then task: hybridizing OpenMP parallelism to improve load balancing and memory efficiency in continental-scale longest flow path computation. *Environmental Modelling & Software, 193*, 106630. https://doi.org/10.1016/j.envsoft.2025.106630

Collischonn, W., Allasia, D., da Silva, B. C., & Tucci, C. E. M. (2007). The MGB-IPH model for large-scale rainfall–runoff modelling. *Hydrological Sciences Journal*, *52*(5), 878–895. https://doi.org/10.1623/hysj.52.5.878

David, C. H., Maidment, D. R., Niu, G. Y., Yang, Z. L., Habets, F., & Eijkhout, V. (2011). River Network Routing on the NHDPlus Dataset. *Journal of Hydrometeorology*, *12*(5), 913–934. https://doi.org/10.1175/2011jhm1345.1

Eilander, D. (2020). *pyflwdir* [Software]. Zenodo. https://doi.org/10.5281/zenodo.4287337

Eilander, D., van Verseveld, W., Yamazaki, D., Weerts, A., Winsemius, H. C., & Ward, P. J. (2021). A hydrography upscaling method for scale-invariant parametrization of distributed hydrological models. *Hydrology and Earth System Sciences*, *25*(9), 5287–5313. https://doi.org/10.5194/hess-25-5287-2021

Farr, T. G., Rosen, P. A., Caro, E., Crippen, R., Duren, R., Hensley, S., Kobrick, M., Paller, M., Rodriguez, E., Roth, L., Seal, D., Shaffer, S., Shimada, J., Umland, J., Werner, M., Oskin, M.,

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

Burbank, D., & Alsdorf, D. (2007). The Shuttle Radar Topography Mission. *Reviews of Geophysics*, *45*(2), 2005RG000183. https://doi.org/10.1029/2005rg000183

Feng, D., Tan, Z., Engwirda, D., Wolfe, J. D., Xu, D., Liao, C., Bisht, G., Benedict, J. J., Zhou, T., Deb, M., Li, H. Y., & Leung, L. R. (2025). Disentangling atmospheric, hydrological, and coupling uncertainties in compound flood modeling within a coupled Earth system model. *Natural Hazards and Earth System Sciences*, *25*(9), 3619–3639. https://doi.org/10.5194/nhess-25-3619-2025

Freitas, H. R., Mendes, C. L., & Ilic, A. (2022). Performance optimization of the MGB hydrological model for multi-core and GPU architectures. *Environmental Modelling & Software*, *148*, 105271. https://doi.org/10.1016/j.envsoft.2021.105271

García-Márquez, J. R., Grigoropoulou, A., Tomiczek, T., Schürz, M., Bremerich, V., Torres-Cambas, Y., Buurman, M., Bego, K., Amatulli, G., & Domisch, S. (2026). Environment90m – globally standardized environmental variables for freshwater science at high spatial resolution. *Earth System Science Data*, *18*(2), 1541–1559. https://doi.org/10.5194/essd-18-1541-2026

Gochis, D. J., Barlage, M., Dugger, A., FitzGerald, K., Karsten, L., McAllister, M., McCreight, J., Mills, J., RafieeiNasab, A., Read, L., Sampson, K., Yates, D., & Yu, W. (2018). *The WRF-Hydro modeling system technical description (Version 5.0)* (NCAR Technical Note). National Center for Atmospheric Research. https://doi.org/10.5065/D6J38RBJ

Grimaldi, S., Salamon, P., Russo, C., Disperati, J., Zsoster, E., Carton de Wiart, C., Mazzetti, C., Choulga, M., Moschini, F., Harrigan, S., Gomes, G., Jesus, C. R., Ramos, A., Barnard, C., Hansford, E., & Prudhomme, C. (2023). GloFAS v4.0: towards hyper-resolution hydrological modelling at global scale [Conference abstract]. EGU General Assembly 2023, Vienna, Austria. https://doi.org/10.5194/egusphere-egu23-3410

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



1054 Hamman, J. J., Nijssen, B., Bohn, T. J., Gergel, D. R., & Mao, Y. (2018). The Variable Infiltration
1055 Capacity model version 5 (VIC-5): infrastructure improvements for new applications and
1056 reproducibility. *Geoscientific Model Development*, *11*(8), 3481–3496.
1057 https://doi.org/10.5194/gmd-11-3481-2018
1058 Hanasaki, N., Kanae, S., Oki, T., Masuda, K., Motoya, K., Shirakawa, N., Shen, Y., & Tanaka, K.
1059 (2008). An integrated model for the assessment of global water resources – Part 1: Model
1060 description and input meteorological forcing. *Hydrology and Earth System Sciences*, *12*(4), 1007–
1061 1025. https://doi.org/10.5194/hess-12-1007-2008
1062 He, C., Yang, C. J., Turowski, J. M., Ott, R. F., Braun, J., Tang, H., Ghantous, S., Yuan, X., &
1063 Stucky de Quay, G. (2024). A global dataset of the shape of drainage systems. *Earth System*
1064 *Science Data*, *16*(2), 1151–1166. https://doi.org/10.5194/essd-16-1151-2024
1065 Hobley, D. E. J., Adams, J. M., Nudurupati, S. S., Hutton, E. W. H., Gasparini, N. M.,
1066 Istanbulluoglu, E., & Tucker, G. E. (2017). Creative computing with Landlab: an open-source
1067 toolkit for building, coupling, and exploring two-dimensional numerical models of Earth-surface
1068 dynamics. *Earth Surface Dynamics*, *5*(1), 21–46. https://doi.org/10.5194/esurf-5-21-2017
1069 Hoch, J. M., Sutanudjaja, E. H., Wanders, N., van Beek, R. L. P. H., & Bierkens, M. F. P. (2023).
1070 Hyper-resolution PCR-GLOBWB: opportunities and challenges from refining model spatial
1071 resolution to 1 km over the European continent. *Hydrology and Earth System Sciences*, *27*(6),
1072 1383–1401. https://doi.org/10.5194/hess-27-1383-2023
1073 Hu, Y., Wu, H., Alfieri, L., Gu, G., Yilmaz, K. K., Li, C., Jiang, L., Huang, Z., Chen, W., Wu, W.,
1074 & Han, Q. (2024). A time-space varying distributed unit hydrograph (TS-DUH) for operational
1075 flash flood forecasting using publicly-available datasets. *Journal of Hydrology, 642*, 131785.
1076 https://doi.org/10.1016/j.jhydrol.2024.131785

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



Jiang, L., Wu, H., Yang, T., Qu, L., & Huang, Z. (2025a). Accelerating large-scale hydrological modeling with stepwise spatial-temporal multimember parallelization. *Environmental Modelling & Software*, *191*, 106495. https://doi.org/10.1016/j.envsoft.2025.106495

Jiang, L., Wu, H., Li, C., Kimball, J. S., Gu, G., Hu, Y., Chen, S., Hu, Y., Dong, S., & Zhou, X. (2025b). Combining climate forcings and initial hydrological conditions for enhanced subseasonal to seasonal flood forecasting. *Journal of Hydrometeorology, 26*(12), 1867–1886. https://doi.org/10.1175/JHM-D-25-0015.1

Jiang, L., Wu, H., Chen, W., Huang, Z., & Yang, T. (2026a). MERIT-FullBasin: A global 90-meter basin dataset with a per-pixel upstream index and morphometric attributes [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.20344113

Jiang, L., Wu, H., Chen, W., Huang, Z., Yamazaki, D., Yang, T., & Li, L. (2026b). MERIT-FlowTopo: A reusable computational foundation for hyperresolution hydrology on the global 90 m drainage network (Version 1.0) [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.20653059

Jiang, L., Wu, H., Chen, W., Huang, Z., Yamazaki, D., Yang, T., & Li, L. (2026c). MERIT-DrainAttr: Baseline drainage attributes for the global 90 m drainage network (Version 1.0) [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.20686665

Jiang, L., Wu, H., Chen, W., Huang, Z., Yamazaki, D., Yang, T., & Li, L. (2026d). MERIT-FlowTopo code: the C and Python implementations and the figure scripts for "A reusable computational foundation for hyperresolution hydrology on the global 90 m drainage network" (Version 1.0) [Software]. Zenodo. https://doi.org/10.5281/zenodo.22227621

Kang, S., Yin, J., & Yamazaki, D. (2025). CaMa-Flood-GPU: A GPU-based hydrodynamic model implementation for scalable global simulations. [Preprint]. Authorea. https://doi.org/10.22541/essoar.176442648.85093032/v2

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



Karssenberg, D., Schmitz, O., Salamon, P., de Jong, K., & Bierkens, M. F. (2010). A software framework for construction of process-based stochastic spatio-temporal models and data assimilation. *Environmental Modelling & Software*, *25*(4), 489–502. https://doi.org/10.1016/j.envsoft.2009.10.004

Karypis, G., & Kumar, V. (1998). A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, *20*(1), 359–392. https://doi.org/10.1137/s1064827595287997

Kotyra, B., Chabudziński, Ł., & Stpiczyński, P. (2021). High-performance parallel implementations of flow accumulation algorithms for multicore architectures. *Computers & Geosciences*, *151*, 104741. https://doi.org/10.1016/j.cageo.2021.104741

Kotyra, B. (2023). High-performance watershed delineation algorithm for GPU using CUDA and OpenMP. *Environmental Modelling & Software*, *160*, 105613. https://doi.org/10.1016/j.envsoft.2022.105613

Kotyra, B. (2025). A GPU algorithm for identifying the longest flow paths in catchments. *Computers & Geosciences*, *203*, 105961. https://doi.org/10.1016/j.cageo.2025.105961

Langevin, C. D., Hughes, J. D., Banta, E. R., Niswonger, R. G., Panday, S., & Provost, A. M. (2017). *Documentation for the MODFLOW 6 Groundwater Flow Model* (U.S. Geological Survey Techniques and Methods, book 6, chap. A55). U.S. Geological Survey. https://doi.org/10.3133/tm6A55

Lehner, B., Verdin, K., & Jarvis, A. (2008). New Global Hydrography Derived From Spaceborne Elevation Data. *Eos, Transactions American Geophysical Union*, *89*(10), 93–94. https://doi.org/10.1029/2008eo100001

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



1122 Li, H., Wigmosta, M. S., Wu, H., Huang, M., Ke, Y., Coleman, A. M., & Leung, L. R. (2013). A
1123 Physically Based Runoff Routing Model for Land Surface and Earth System Models. *Journal of*
1124 *Hydrometeorology*, *14*(3), 808–828. https://doi.org/10.1175/jhm-d-12-015.1
1125 Li, X., Vernon, C. R., Hejazi, M. I., Link, R. P., Feng, L., Liu, Y., & Rauchenstein, L. T. (2017).
1126 Xanthos – A Global Hydrologic Model. *Journal of Open Research Software, 5*(1), 21.
1127 https://doi.org/10.5334/jors.181
1128 Lin, P., Pan, M., Beck, H. E., Yang, Y., Yamazaki, D., Frasson, R., David, C. H., Durand, M.,
1129 Pavelsky, T. M., Allen, G. H., Gleason, C. J., & Wood, E. F. (2019). Global Reconstruction of
1130 Naturalized River Flows at 2.94 Million Reaches. *Water Resources Research*, *55*(8), 6499–6516.
1131 https://doi.org/10.1029/2019wr025287
1132 Lin, P., Pan, M., Wood, E. F., Yamazaki, D., & Allen, G. H. (2021). A new vector-based global
1133 river network dataset accounting for variable drainage density. *Scientific Data*, *8*(1), 28.
1134 https://doi.org/10.1038/s41597-021-00819-9
1135 Lindsay, J. (2016). Whitebox GAT: A case study in geomorphometric analysis. *Computers &*
1136 *Geosciences*, *95*, 75–84. https://doi.org/10.1016/j.cageo.2016.07.003
1137 Lindström, G., Pers, C., Rosberg, J., Strömqvist, J., & Arheimer, B. (2010). Development and
1138 testing of the HYPE (Hydrological Predictions for the Environment) water quality model for
1139 different spatial scales. *Hydrology Research*, *41*(3-4), 295–319.
1140 https://doi.org/10.2166/nh.2010.007
1141 Liu, J., Zhu, A. X., Liu, Y., Zhu, T., & Qin, C. Z. (2014). A layered approach to parallel computing
1142 for spatially distributed hydrological modeling. *Environmental Modelling & Software*, *51*, 221–
1143 227. https://doi.org/10.1016/j.envsoft.2013.10.005

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



1144 Liu, J., Zhu, A. X., Qin, C. Z., Wu, H., & Jiang, J. (2016). A two-level parallelization method for
1145 distributed hydrological models. *Environmental Modelling & Software*, *80*, 175–184.
1146 https://doi.org/10.1016/j.envsoft.2016.02.032
1147 Liu, Y.-H., Yang, Z. L., & Lin, P. (2023). Parallel river channel routing computation based on a
1148 straightforward domain decomposition of river networks. *Journal of Hydrology*, *625*, 129988.
1149 https://doi.org/10.1016/j.jhydrol.2023.129988
1150 Markstrom, S. L., Regan, R. S., Hay, L. E., Viger, R. J., Webb, R. M. T., Payn, R. A., & LaFontaine,
1151 J. H. (2015). *PRMS-IV, the Precipitation-Runoff Modeling System, Version 4* (U.S. Geological
1152 Survey Techniques and Methods, book 6, chap. B7). U.S. Geological Survey.
1153 https://doi.org/10.3133/tm6B7
1154 Mattson, R., Gecsei, J., Slutz, D. R., & Traiger, I. L. (1970). Evaluation techniques for storage
1155 hierarchies. *IBM Systems Journal*, *9*(2), 78–117. https://doi.org/10.1147/sj.92.0078
1156 Maxwell, R. M., Condon, L. E., & Kollet, S. J. (2015). A high-resolution simulation of
1157 groundwater and surface water over most of the continental US with the integrated hydrologic
1158 model ParFlow v3. *Geoscientific Model Development*, *8*(3), 923–937.
1159 https://doi.org/10.5194/gmd-8-923-2015
1160 Melsen, L. A., Teuling, A. J., Torfs, P. J. J. F., Uijlenhoet, R., Mizukami, N., & Clark, M. P. (2016).
1161 HESS Opinions: The need for process-based evaluation of large-domain hyper-resolution models.
1162 *Hydrology and Earth System Sciences*, *20*(3), 1069–1079. https://doi.org/10.5194/hess-20-1069-
1163 2016
1164 Metz, M., Mitasova, H., & Harmon, R. S. (2011). Efficient extraction of drainage networks from
1165 massive, radar-based elevation models with least cost path search. *Hydrology and Earth System*
1166 *Sciences*, *15*(2), 667–678. https://doi.org/10.5194/hess-15-667-2011

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



1167 Mizukami, N., Clark, M. P., Sampson, K., Nijssen, B., Mao, Y., McMillan, H., Viger, R. J.,
1168 Markstrom, S. L., Hay, L. E., Woods, R., Arnold, J. R., & Brekke, L. D. (2016). mizuRoute version
1169 1: a river network routing tool for a continental domain water resources applications. *Geoscientific*
1170 *Model Development*, *9*(6), 2223–2238. https://doi.org/10.5194/gmd-9-2223-2016
1171 Mizukami, N., Clark, M. P., Gharari, S., Kluzek, E., Pan, M., Lin, P., Beck, H. E., & Yamazaki,
1172 D. (2021). A Vector-Based River Routing Model for Earth System Models: Parallelization and
1173 Global Applications. *Journal of Advances in Modeling Earth Systems*, *13*(6), e2020MS002434.
1174 https://doi.org/10.1029/2020ms002434
1175 Moore, R. B., McKay, L. D., Rea, A. H., Bondelid, T. R., Price, C. V., Dewald, T. G., & Johnston,
1176 C. M. (2019). User's guide for the national hydrography dataset plus (NHDPlus) high resolution.
1177 *Open-File Report*. https://doi.org/10.3133/ofr20191096
1178 Mudd, S. M., Attal, M., Milodowski, D. T., Grieve, S. W. D., & Valters, D. A. (2014). A statistical
1179 framework to quantify spatial variation in channel gradients using the integral method of channel
1180 profile analysis. *Journal of Geophysical Research: Earth Surface*, *119*(2), 138–152.
1181 https://doi.org/10.1002/2013jf002981
1182 Müller Schmied, H., Cáceres, D., Eisner, S., Flörke, M., Herbert, C., Niemann, C., Peiris, T. A.,
1183 Popat, E., Portmann, F. T., Reinecke, R., Schumacher, M., Shadkam, S., Telteu, C. E., Trautmann,
1184 T., & Döll, P. (2021). The global water resources and use model WaterGAP v2.2d: model
1185 description and evaluation. *Geoscientific Model Development*, *14*(2), 1037–1079.
1186 https://doi.org/10.5194/gmd-14-1037-2021
1187 Nguyen-Quang, T., Polcher, J., Ducharne, A., Arsouze, T., Zhou, X., Schneider, A., & Fita, L.
1188 (2018). ORCHIDEE-ROUTING: revising the river routing scheme using a high-resolution

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



hydrological database. *Geoscientific Model Development*, *11*(12), 4965–4985. https://doi.org/10.5194/gmd-11-4965-2018

NOAA Office of Water Prediction. (2023). T-Route: Tree-based channel routing [Software]. GitHub. https://github.com/NOAA-OWP/t-route

O'Callaghan, J. F., & Mark, D. M. (1984). The extraction of drainage networks from digital elevation data. *Computer Vision, Graphics, and Image Processing*, *28*(3), 323–344. https://doi.org/10.1016/s0734-189x(84)80011-0

Pati, A., Sahoo, B., & Perumal, M. (2023). A Physically-Based Reverse-Stage Routing Model Considering Lateral Flow for Establishing Normal Rating Curves at Ungauged Upstream River Sections. *Water Resources Research*, *59*(7), e2022WR034150. https://doi.org/10.1029/2022wr034150

Qin, C. Z., & Zhan, L. (2012). Parallelizing flow-accumulation calculations on graphics processing units—From iterative DEM preprocessing algorithm to recursive multiple-flow-direction algorithm. *Computers & Geosciences*, *43*, 7–16. https://doi.org/10.1016/j.cageo.2012.02.022

Quichimbo, E. A., Singer, M. B., Michaelides, K., Hobley, D. E. J., Rosolem, R., & Cuthbert, M. O. (2021). DRYP 1.0: a parsimonious hydrological model of DRYland Partitioning of the water balance. *Geoscientific Model Development, 14*(11), 6893–6917. https://doi.org/10.5194/gmd-14-6893-2021

Quinn, P., Beven, K., Chevallier, P., & Planchon, O. (1991). The prediction of hillslope flow paths for distributed hydrological modelling using digital terrain models. *Hydrological Processes*, *5*(1), 59–79. https://doi.org/10.1002/hyp.3360050106

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



Richardson, A., Hill, C. N., & Perron, J. T. (2014). IDA: An implicit, parallelizable method for calculating drainage area. *Water Resources Research, 50*(5), 4110–4130. https://doi.org/10.1002/2013WR014326

Schwanghart, W., & Scherler, D. (2014). Short Communication: TopoToolbox 2 – MATLAB-based software for topographic analysis and modeling in Earth surface sciences. *Earth Surface Dynamics*, *2*(1), 1–7. https://doi.org/10.5194/esurf-2-1-2014

Small, S. J., Jay, L. O., Mantilla, R., Curtu, R., Cunha, L. K., Fonley, M., & Krajewski, W. F. (2013). An asynchronous solver for systems of ODEs linked by a directed tree structure. *Advances in Water Resources, 53*, 23–32. https://doi.org/10.1016/j.advwatres.2012.10.011

Sten, J., Lilja, H., Hyväluoma, J., Westerholm, J., & Aspnäs, M. (2016). Parallel flow accumulation algorithms for graphical processing units with application to RUSLE model. *Computers & Geosciences*, *89*, 88–95. https://doi.org/10.1016/j.cageo.2016.01.006

Strahler, A. N. (1957). Quantitative analysis of watershed geomorphology. *Eos, Transactions American Geophysical Union*, *38*(6), 913–920. https://doi.org/10.1029/tr038i006p00913

Sutanudjaja, E. H., van Beek, R., Wanders, N., Wada, Y., Bosmans, J. H. C., Drost, N., van der Ent, R. J., de Graaf, I. E. M., Hoch, J. M., de Jong, K., Karssenberg, D., López López, P., Peßenteiner, S., Schmitz, O., Straatsma, M. W., Vannametee, E., Wisser, D., & Bierkens, M. F. P. (2018). PCR-GLOBWB 2: a 5 arcmin global hydrological and water resources model. *Geoscientific Model Development*, *11*(6), 2429–2453. https://doi.org/10.5194/gmd-11-2429-2018

Tague, C. L., & Band, L. E. (2004). RHESSys: Regional Hydro-Ecologic Simulation System—an object-oriented approach to spatially distributed modeling of carbon, water, and nutrient cycling. *Earth Interactions, 8*(19), 1–42. https://doi.org/10.1175/1087-3562(2004)8<1:RRHSSO>2.0.CO;2

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



Tarboton, D. G. (1997). A new method for the determination of flow directions and upslope areas in grid digital elevation models. *Water Resources Research*, *33*(2), 309–319. https://doi.org/10.1029/96wr03137

Thober, S., Cuntz, M., Kelbling, M., Kumar, R., Mai, J., & Samaniego, L. (2019). The multiscale routing model mRM v1.0: simple river routing at resolutions from 1 to 50 km. *Geoscientific Model Development*, *12*(6), 2501–2521. https://doi.org/10.5194/gmd-12-2501-2019

van der Knijff, J. M., Younis, J., & de Roo, A. P. J. (2010). LISFLOOD: a GIS-based distributed model for river basin scale water balance and flood simulation. *International Journal of Geographical Information Science*, *24*(2), 189–212. https://doi.org/10.1080/13658810802549154

van Jaarsveld, B., Wanders, N., Sutanudjaja, E. H., Hoch, J., Droppers, B., Janzing, J., van Beek, R. L. P. H., & Bierkens, M. F. P. (2025). A first attempt to model global hydrology at hyper-resolution. *Earth System Dynamics*, *16*(1), 29–54. https://doi.org/10.5194/esd-16-29-2025

van Verseveld, W. J., Weerts, A. H., Visser, M., Buitink, J., Imhoff, R. O., Boisgontier, H., Bouaziz, L., Eilander, D., Hegnauer, M., ten Velden, C., & Russell, B. (2024). Wflow_sbm v0.7.3, a spatially distributed hydrological model: from global data to local applications. *Geoscientific Model Development*, *17*(8), 3199–3234. https://doi.org/10.5194/gmd-17-3199-2024

Verkaik, J., Sutanudjaja, E. H., Oude Essink, G. H. P., Lin, H. X., & Bierkens, M. F. P. (2024). GLOBGM v1.0: a parallel implementation of a 30 arcsec PCR-GLOBWB-MODFLOW global-scale groundwater model. *Geoscientific Model Development*, *17*(1), 275–300. https://doi.org/10.5194/gmd-17-275-2024

Wallis, C., Watson, D., Tarboton, D., & Wallace, R. (2009). Parallel flow-direction and contributing area calculation for hydrology analysis in digital elevation models. In *Proceedings of*

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'09) (pp. 467–472). CSREA Press.

Wei, J., Luo, X., Huang, H., Liao, W., Lei, X., Zhao, J., & Wang, H. (2023). Enable high-resolution, real-time ensemble simulation and data assimilation of flood inundation using distributed GPU parallelization. *Journal of Hydrology*, *619*, 129277. https://doi.org/10.1016/j.jhydrol.2023.129277

Wisser, D., Fekete, B. M., Vörösmarty, C. J., & Schumann, A. H. (2010). Reconstructing 20th century global hydrography: a contribution to the Global Terrestrial Network-Hydrology (GTN-H). *Hydrology and Earth System Sciences, 14*(1), 1–24. https://doi.org/10.5194/hess-14-1-2010

Wood, E. F., Roundy, J. K., Troy, T. J., van Beek, L. P. H., Bierkens, M. F. P., Blyth, E., de Roo, A., Döll, P., Ek, M., Famiglietti, J., Gochis, D., van de Giesen, N., Houser, P., Jaffé, P. R., Kollet, S., Lehner, B., Lettenmaier, D. P., Peters-Lidard, C., Sivapalan, M., Sheffield, J., Wade, A., & Whitehead, P. (2011). Hyperresolution global land surface modeling: Meeting a grand challenge for monitoring Earth's terrestrial water. *Water Resources Research*, *47*(5), 2010WR010090. https://doi.org/10.1029/2010wr010090

Wood, E. F., Roundy, J. K., Troy, T. J., van Beek, R., Bierkens, M., Blyth, E., de Roo, A., Döll, P., Ek, M., Famiglietti, J., Gochis, D., van de Giesen, N., Houser, P., Jaffe, P., Kollet, S., Lehner, B., Lettenmaier, D. P., Peters-Lidard, C. D., Sivapalan, M., Sheffield, J., Wade, A. J., & Whitehead, P. (2012). Reply to comment by Keith J. Beven and Hannah L. Cloke on "Hyperresolution global land surface modeling: Meeting a grand challenge for monitoring Earth's terrestrial water". *Water Resources Research*, *48*(1), 2011WR011202. https://doi.org/10.1029/2011wr011202

Wu, H., Kimball, J. S., Mantua, N., & Stanford, J. (2011). Automated upscaling of river networks for macroscale hydrological modeling. *Water Resources Research*, *47*(3), 2009WR008871. https://doi.org/10.1029/2009wr008871

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



Wu, H., Kimball, J. S., Li, H., Huang, M., Leung, L. R., & Adler, R. F. (2012). A new global river network database for macroscale hydrologic modeling. *Water Resources Research*, *48*(9), 2012WR012313. https://doi.org/10.1029/2012wr012313

Wu, H., Adler, R. F., Tian, Y., Huffman, G. J., Li, H., & Wang, J. (2014). Real-time global flood estimation using satellite-based precipitation and a coupled land surface and routing model. *Water Resources Research*, *50*(3), 2693–2717. https://doi.org/10.1002/2013wr014710

Wu, H., Kimball, J. S., Zhou, N., Alfieri, L., Luo, L., Du, J., & Huang, Z. (2019). Evaluation of real-time global flood modeling with satellite surface inundation observations from SMAP. *Remote Sensing of Environment, 233*, 111360. https://doi.org/10.1016/j.rse.2019.111360

Xie, J., Liu, X., Bai, P., & Liu, C. (2022). Rapid Watershed Delineation Using an Automatic Outlet Relocation Algorithm. *Water Resources Research*, *58*(3), e2021WR031129. https://doi.org/10.1029/2021wr031129

Yamazaki, D., Oki, T., & Kanae, S. (2009). Deriving a global river network map and its sub-grid topographic characteristics from a fine-resolution flow direction map. *Hydrology and Earth System Sciences*, *13*(11), 2241–2251. https://doi.org/10.5194/hess-13-2241-2009

Yamazaki, D., Kanae, S., Kim, H., & Oki, T. (2011). A physically based description of floodplain inundation dynamics in a global river routing model. *Water Resources Research*, *47*(4), 2010WR009726. https://doi.org/10.1029/2010wr009726

Yamazaki, D., de Almeida, G. A. M., & 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*(11), 7221–7235. https://doi.org/10.1002/wrcr.20552

https://doi.org/10.5194/egusphere-2026-5497
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



1299 Yamazaki, D., Ikeshima, D., Tawatari, R., Yamaguchi, T., O'Loughlin, F., Neal, J. C., Sampson,
1300 C. C., Kanae, S., & Bates, P. D. (2017). A high-accuracy map of global terrain elevations.
1301 *Geophysical Research Letters*, *44*(11), 5844–5853. https://doi.org/10.1002/2017gl072874
1302 Yamazaki, D., Ikeshima, D., Sosa, J., Bates, P. D., Allen, G. H., & Pavelsky, T. M. (2019). MERIT
1303 Hydro: A High-Resolution Global Hydrography Map Based on Latest Topography Dataset. *Water*
1304 *Resources Research*, *55*(6), 5053–5073. https://doi.org/10.1029/2019wr024873
1305 Yang, C., Jia, Z., Xu, W., Wei, Z., Zhang, X., Zou, Y., McDonnell, J., Condon, L., Dai, Y., &
1306 Maxwell, R. (2025). CONCN: a high-resolution, integrated surface water–groundwater ParFlow
1307 modeling platform of continental China. *Hydrology and Earth System Sciences*, *29*(9), 2201–2218.
1308 https://doi.org/10.5194/hess-29-2201-2025
1309