Flow Computation on Massive Grids

Laura Toma Rajiv Wickremesinghe
Lars Arge Jeffrey S. Chase Jeffrey S. Vitter

Patrick N. Halpin Dean Urban

NICHOLAS
School of the ENVIRONMENT

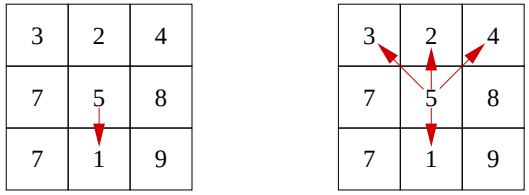
Duke University

Flow Modeling on Terrains

- ☆ Terrain represented as a grid
- ☆ Flow modeled by two basic attributes
  - Flow direction: The direction water flows at a point in the grid
  - Flow accumulation value: Total amount of water which flows through a point in the grid
- ☆ Objective: Compute flow directions and flow accumulation values for the entire grid
  - Flow routing
  - Flow accumulation

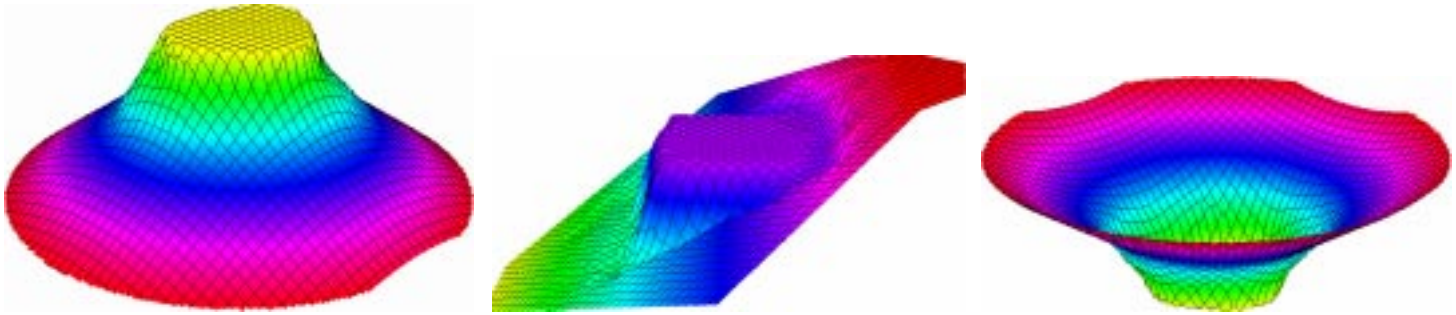
Flow Routing

☆ Water flows downhill.



☆ Compute flow directions by inspecting 8 neighbor points.

☆ Flat areas: plateaus and sinks.

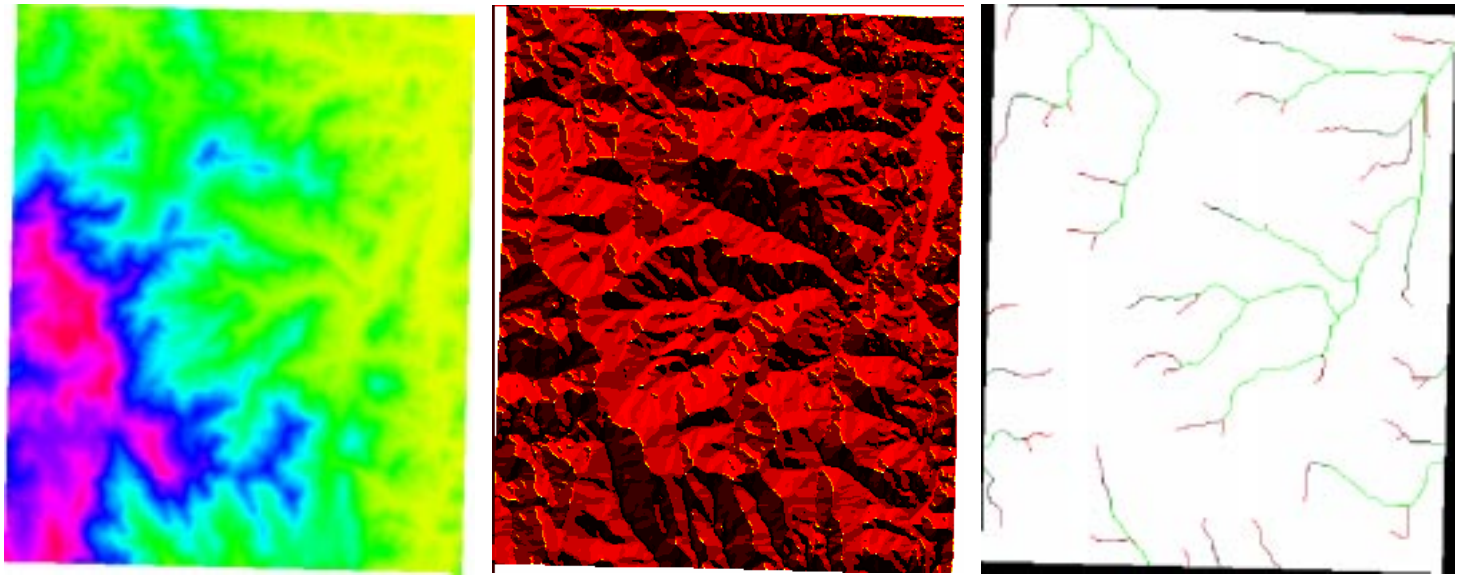


Flow Accumulation

- ☆ Water flows following the flow directions
- ☆ Compute the total amount of flow through each grid point
  - Initially one unit of water on each grid point
  - Every point distributes water to the neighbors pointed to by its flow direction(s)

Applications

- ☆ Watersheds, drainage network
- ☆ Erosion, infiltration, drainage, solar radiation distribution, sediment transport, vegetation structure, species diversity



Massive Data

- ☆ Massive remote sensing data available
  - USGS (entire US at 10m resolution)
  - NASA's SRTM (whole Earth, 5TB)
  - LIDAR
- ☆ Example: Appalachian Mountains (800km × 800km)
  - at 100m resolution: 500MB
  - at 30m resolution: 5.5GB
  - at 10m resolution: 50GB
  - at 1m resolution: 5TB !!

Scalability to Massive Data

☆ Existing software

- GRASS *r.watershed*
  - killed after 17 days on a 50MB dataset
- ArcInfo *flowdirection, flowaccumulation*
  - can handle the 50MB dataset
  - cannot process files $> 2GB$

☆ Current GIS algorithms minimize CPU time

☆ I/O is the bottleneck in computation on massive data!

Our Results: TerraFlow

- ☆ Collection of theoretical algorithms and practical implementations for flow routing and flow accumulation on massive grids.
- ☆ Available at `http://www.cs.duke.edu/geo*/terraflow/`

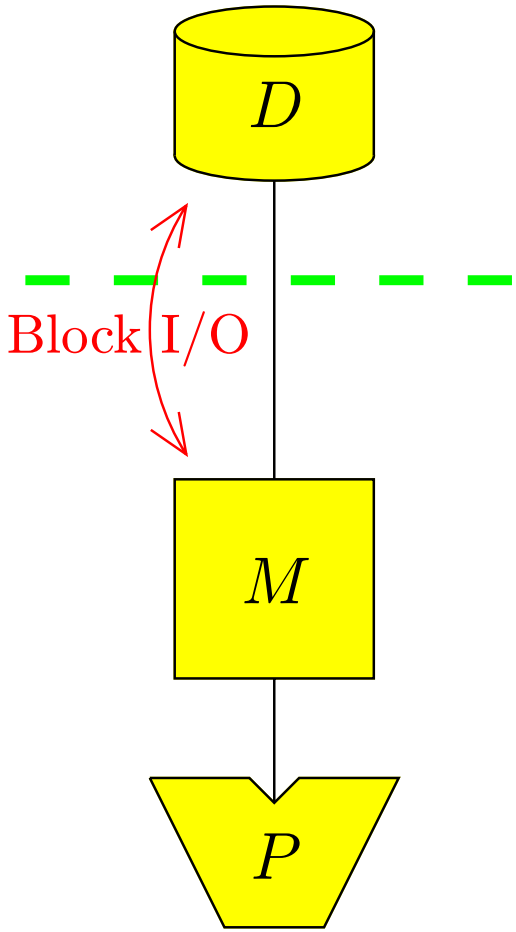
- ☆ Efficient
  - 2-1000 times faster on massive grids than existing software
- ☆ Scalable
  - 1 billion elements! (> 2GB)
- ☆ Flexible
  - different flow models

Outline

- ☆ The I/O-Model
- ☆ Flow routing: Previous work and I/O-efficient algorithm
- ☆ Experimental results
- ☆ Open problems

Disk Model

[Aggarwal & Vitter '88]



N = # of points in the grid

M = # of vertices/edges that fit in memory

B = # of vertices/edges per disk block

- ☆ I/O complexity
- ☆ Basic bounds
  - $\text{scan}(N) = \frac{N}{B} \ll E$
  - $\text{sort}(N) = \Theta(\frac{N}{B} \log_{M/B} \frac{N}{B}) \ll E$

I/O-Efficient Flow Routing

★ Flow routing: assign flow direction to every point such that

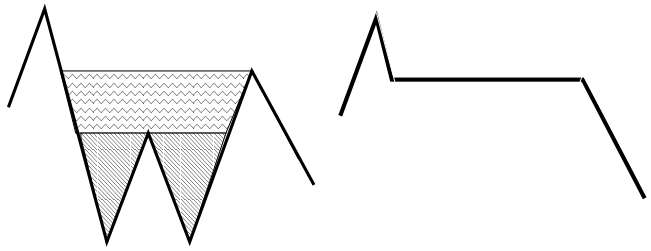
- Flow directions do not induce cycles $\Longrightarrow$ every point has a flow path to the edge of the terrain

Steps:

1. Assign flow directions to all points which have downslope neighbors.
2. Identify flat areas, their boundaries and spill points.
3. Assign flow directions on plateaus.
4. Remove sinks.
5. Assign flow directions to the entire terrain (repeat steps 1-3).

Removing Sinks

- ☆ Sinks are removed by flooding [Jenson & Domingue '88]
- ☆ Flooding fills the terrain up to the steady state level reached when an infinite amount of water is poured onto the terrain and the outside is viewed as a giant ocean.

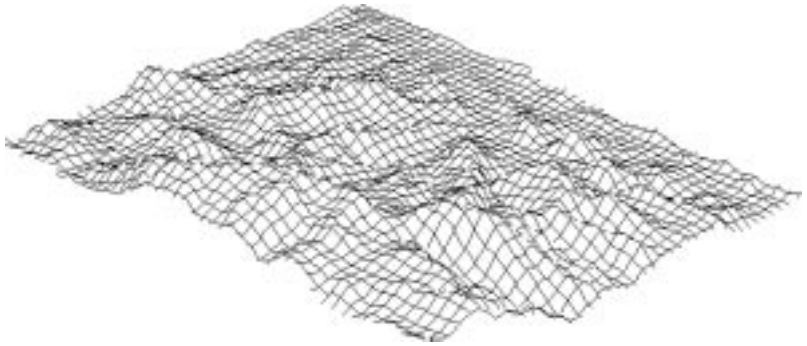


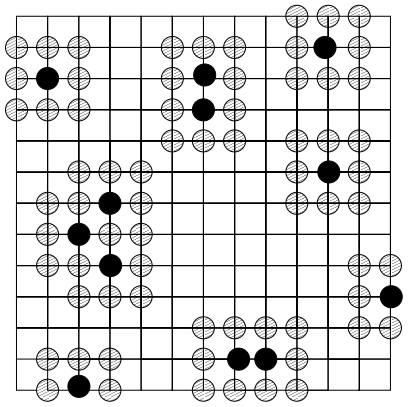
A watershed u is *raised* to height h by raising every point in u of height lower than h to height h.

- ☆ Watershed: part of the terrain that flows into the sink.
- ☆ Partition the terrain into watersheds $\longrightarrow$ watershed graph

Computing Watersheds
Internal Memory Algorithm

★ Assign to each sink a unique watershed label. Process (sweep) points in reverse topological order of the flow directions (increasing order of height), assigning labels to incoming neighbor points $\Longrightarrow$ $O(N)$ time





★ Problem: algorithm uses $O(N)$ I/Os if directions and labels stored as grids (not fitting in memory)

- Points with same height are distributed over the terrain $\Longrightarrow$ scattered accesses

Computing Watersheds
I/O-Efficient Algorithm

- ☆ Eliminate scattered accesses to watershed-label grid
  - Idea: neighbor only needs the label when the sweep plane reaches its elevation
  - Use a $O(\frac{1}{B}\log_{M/B}\frac{N}{B})$ priority queue [A95, BK98]
    - Assign label by inserting it in priority queue with priority equal to neighbor's height
    - Trade space for I/Os: Augment each height with heights of neighbors which flow into it
  - $O(N)$ priority queue operations $\Rightarrow$ $O(\frac{N}{B}\log_{M/B}\frac{N}{B})$ I/Os

Flooding
Internal Memory Algorithm

- ⋆ Watershed graph
  - W watersheds
  - Edges between adjacent watersheds (labeled with height)
- ⋆ Identify and merge watersheds that spill into each other
  - For each watershed follow the steepest edge downslope; if it leads back into itself, merge all watersheds on the cycle
  - Repeat until there are no cycles
- ⋆ $O(W^2)$ time, I/Os

Flooding
I/O-efficient Algorithm

★ Plane sweep algorithm

- Introduce a watershed representing the outside of the terrain
- Initially only the "outside" watershed is done
- Sweep watershed graph bottom-up with a horizontal plane. When hit edge (u, v):
  - If both watersheds are done, ignore
  - If none is done, union them
  - If precisely one is not done, raise it and mark it as done

★ $O(W \cdot \alpha(W, N))$ time, I/Os

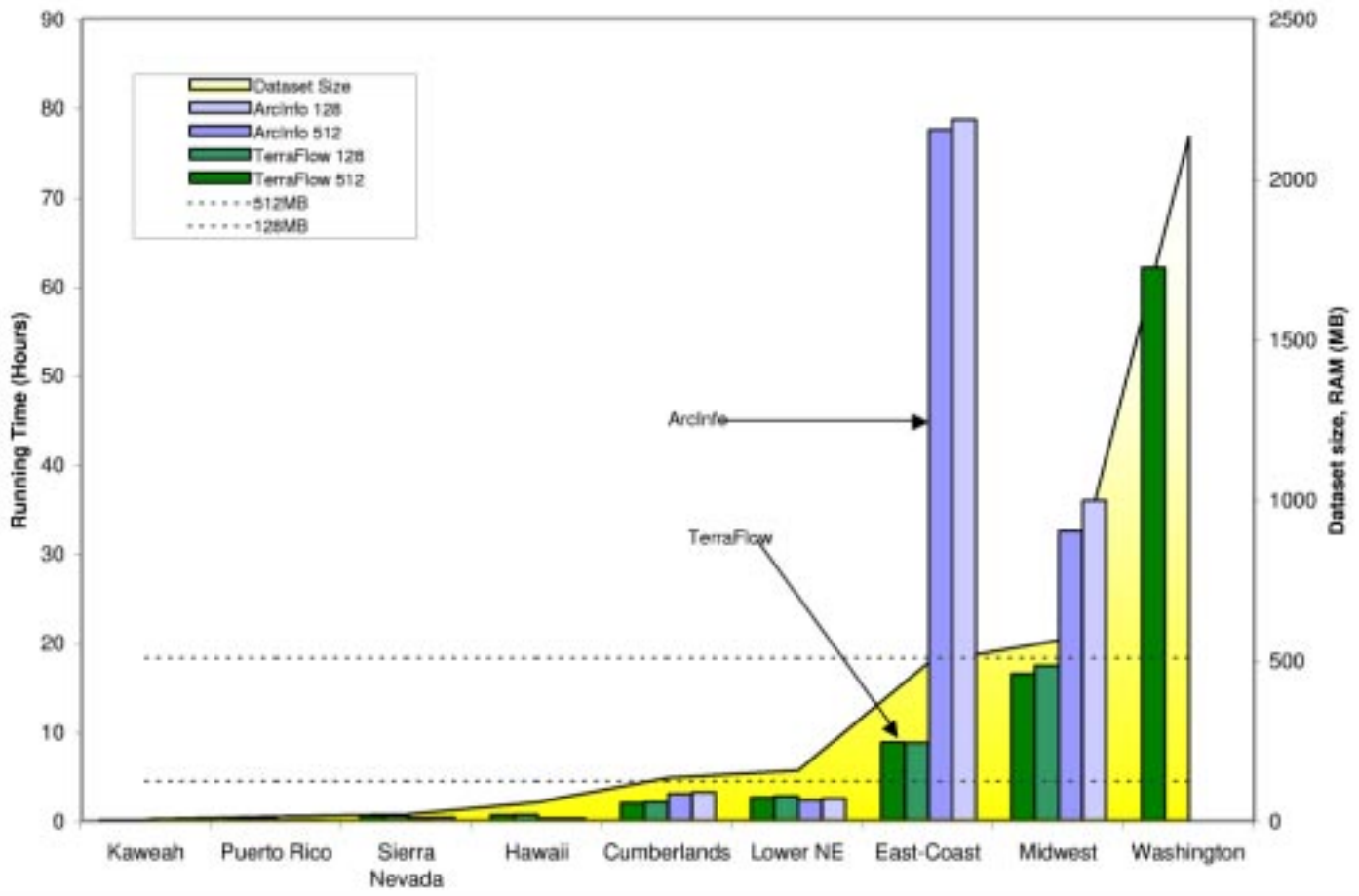
Implementation and Platform

- ☆ `http://www.cs.duke.edu/geo*/terraflow/`
- ☆ C++, TPIE
- ☆ TerraFlow, ArcInfo: 500MHz Alpha, FreeBSD 4.0, 1GB main memory
- ☆ GRASS: 500MHz Intel PIII, FreeBSD 4.0, 1GB main memory
- ☆ TARDEM: 500MHz Intel PIII, Windows, 1GB main memory

Experimental Results: Datasets

Dataset	Resolution	Dimensions	Grid Size
Kaweah	30m	1163 x 1424	3.2MB
Puerto Rico	100m	4452 x 1378	12MB
Sierra Nevada	30m	3750 x 2672	19MB
Hawaii	100m	6784 x 4369	56MB
Cumberlands	80m	8704 x 7673	133MB
Lower New England	80m	9148 x 8509	156MB
Central Appalachians	30m	12042 x10136	232MB
East-Coast USA	100m	13500 x 18200	491MB
Midwest USA	100m	11000 x 25500	561MB
Washington State	10m	33454 x 31866	2GB

TerraFlow Performance



TerraFlow Performance

- ☆ Significant speedup over ArcInfo for large grids
  - East-Coast dataset
    - TerraFlow: 8.7 hours
    - ArcInfo: 78 hours
  - Washington state dataset
    - TerraFlow: 63 hours
    - ArcInfo: cannot process it!
- ☆ Other software
  - GRASS: killed after 17 days on Hawaii
  - TARDEM: Can handle Hawaii. Killed after 20 days on Cumberlands (CPU utilization 5%, 3GB swap file)

Future Directions

- ☆ Flow modeling on TINs
  - Flow along edges. Compute flow accumulation of nodes.
  - Extend grid approach: assign flow at triangle level. Flow across edges and along channel edges. Compute flow accumulation of triangles and channel nodes.
  - Compute contributing area directly: trace steepest downslope paths across triangles.
- ☆ Grid/TIN conversion
  - Maintain global features