Accelerating Scientific Discovery via In Situ Computation of Merge Trees

Dmitriy Morozov, Data Analytics and Visualization, CRD

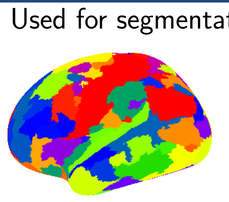
This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, through the grants "Scalable Data-Computing Convergence and Scientic Knowledge Discovery," "Scalable Analysis Methods and In Situ Infrastructure for Extreme Scale Knowledge Discovery," and RAPIDS SciDAC Institute; and by the use of resources of the National Energy Research Scientific Computing Center (NERSC).

Efficient Merge Tree Computation

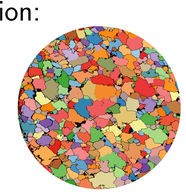
Merge trees have applications in Cosmology, Neuroscience, Materials Science, Combustion, Climate, Biochemistry, among many others, for clustering, threshold selection, distribution and stability of features; machine learning.

Merge Trees

Used for segmentation:

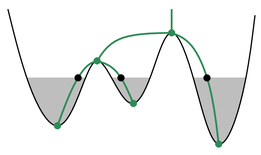


neuroscience



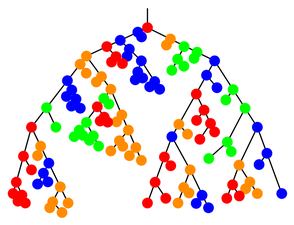
materials science

Track sublevel sets of a function.



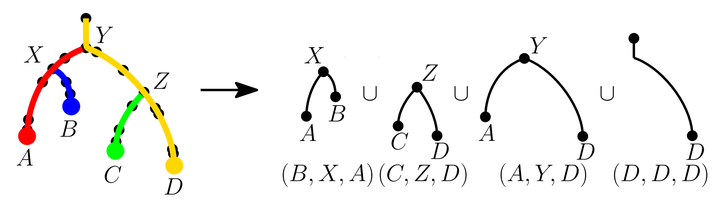
Problem:

- Expensive to compute global information in parallel (too much communication)
- Difficult to post-process (one large tree in one place)

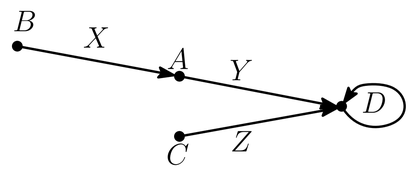


Triplet Merge Trees (+ D. Smirnov)

Decompose the tree into branches. The nesting structure forms a new tree.

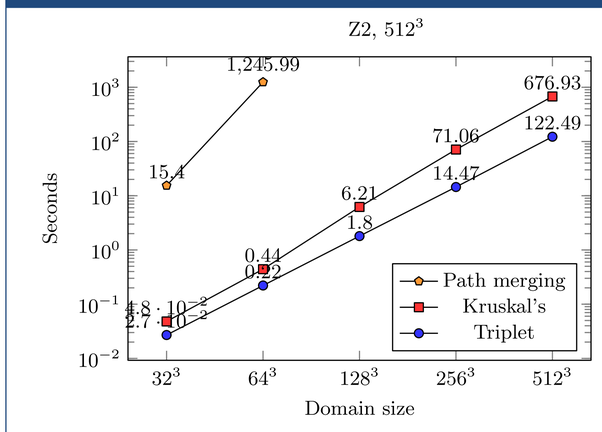


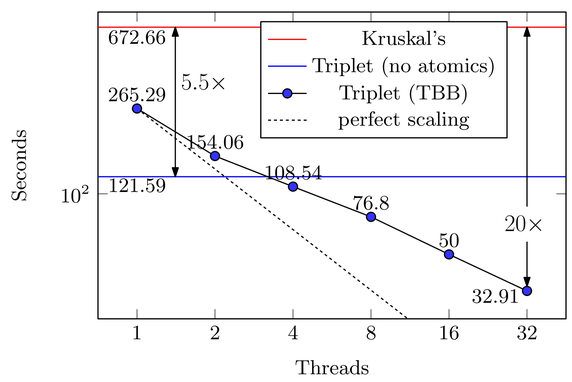
Compute directly. Much more efficient in practice. Easy to parallelize in shared memory using double-word compare-and-swap (DWCAS) operations (lock-free).



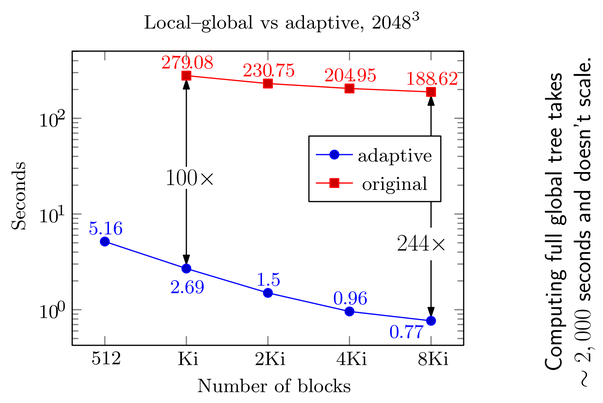
Results

Triplet merge trees





Adaptive Local–global Trees



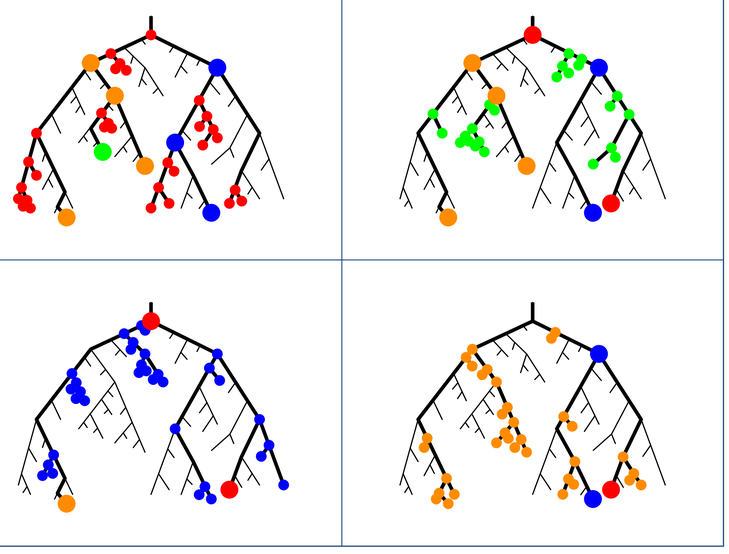
Computing full global tree takes $\sim 2,000$ seconds and doesn't scale.

Data: 2048^3 snapshot of a Nyx simulation.

Local–global Trees (+ G. Weber)

Idea: For each domain region, compute how its local vertices fit into the global tree.

Benefit: Enough information locally to answer queries without communication.

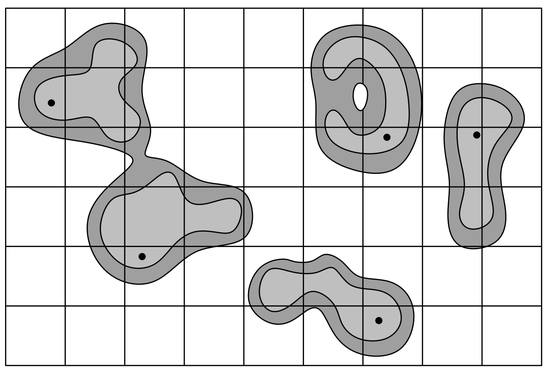


Domain Adaptation (+ A. Nigmetov)

Idea: In many applications, interested in a tree, restricted to a specific sublevel set, $f^{-1}(-\infty, a]$.

Benefit: When the sublevel set is broken into many smaller components, we can compute the tree using only a local exchange. Much less communication, much faster.

Especially useful for halo (and subhalo) finding in cosmology, where there is a well-defined threshold beyond which, there can be no halos.



Also works on AMR data. (First such code.)

Henson: Cooperative Multitasking for In Situ Processing

(+ Z. Lukić)

Problem

- Simulations are getting larger.
- High cost of I/O (growing with every new system).
- No disk space to store simulation results. (A single Nyx snapshot is ~ 56 TB; scratch space at NERSC is 20 TB.)
- Queue wait times favor large jobs (penalizes ensembles of smaller runs, e.g., for exploring the parameter space).

Consequences:

- Can't analyze every time step of the simulation ($\Rightarrow$ can't augment the simulation with analysis).
- Difficult to couple simulations together.
- Complicated to do surrogate modeling (difficulty coordinating individual runs).
- Tight coupling of codes requires complicated maintenance, coordination between many parties, lots of wasted programmer time.

Solution

Main ingredients:

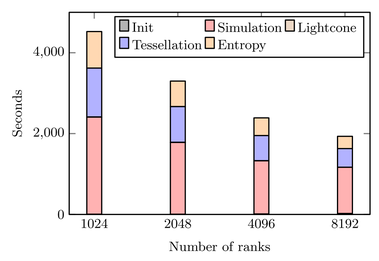
- Position-independent executables = code is an executable and a shared library ($\Rightarrow$ can be opened via `dlopen`)
- Coroutines = codes cooperatively switch execution (e.g., a simulation yields after each time step)

Henson:

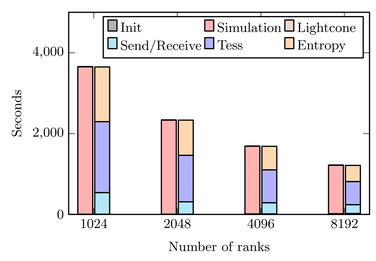
- Codes are loaded into the **same address space** without having to modify their memory management.
- Codes can exchange data (including pointers to their internal arrays) via a shared table.
- User specifies as a script how execution is to alternate between the codes.
- Using PMPI wrappers restrict codes to subsets of the full MPI job:
  - in transit analysis (including $M : N$)
  - dynamically scheduled ensembles of simulations (e.g., guided by a surrogate model)

Results

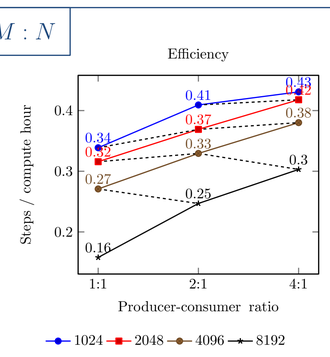
in situ:



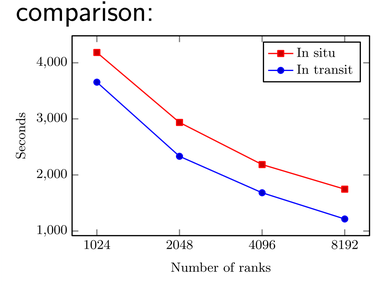
in transit:



$M : N$



comparison:



Different codes scale differently; none perfectly.
The optimal configuration depends on relative scaling.

Contact: dmorozov@lbl.gov