Implementing Connected Component Labeling as a User Defined Operator for SciDB

Amidu Oloso[1,2], Kwo-Sen Kuo[1,3], Thomas Clune[1], Paul Brown[4], Alex Poliakov[4]

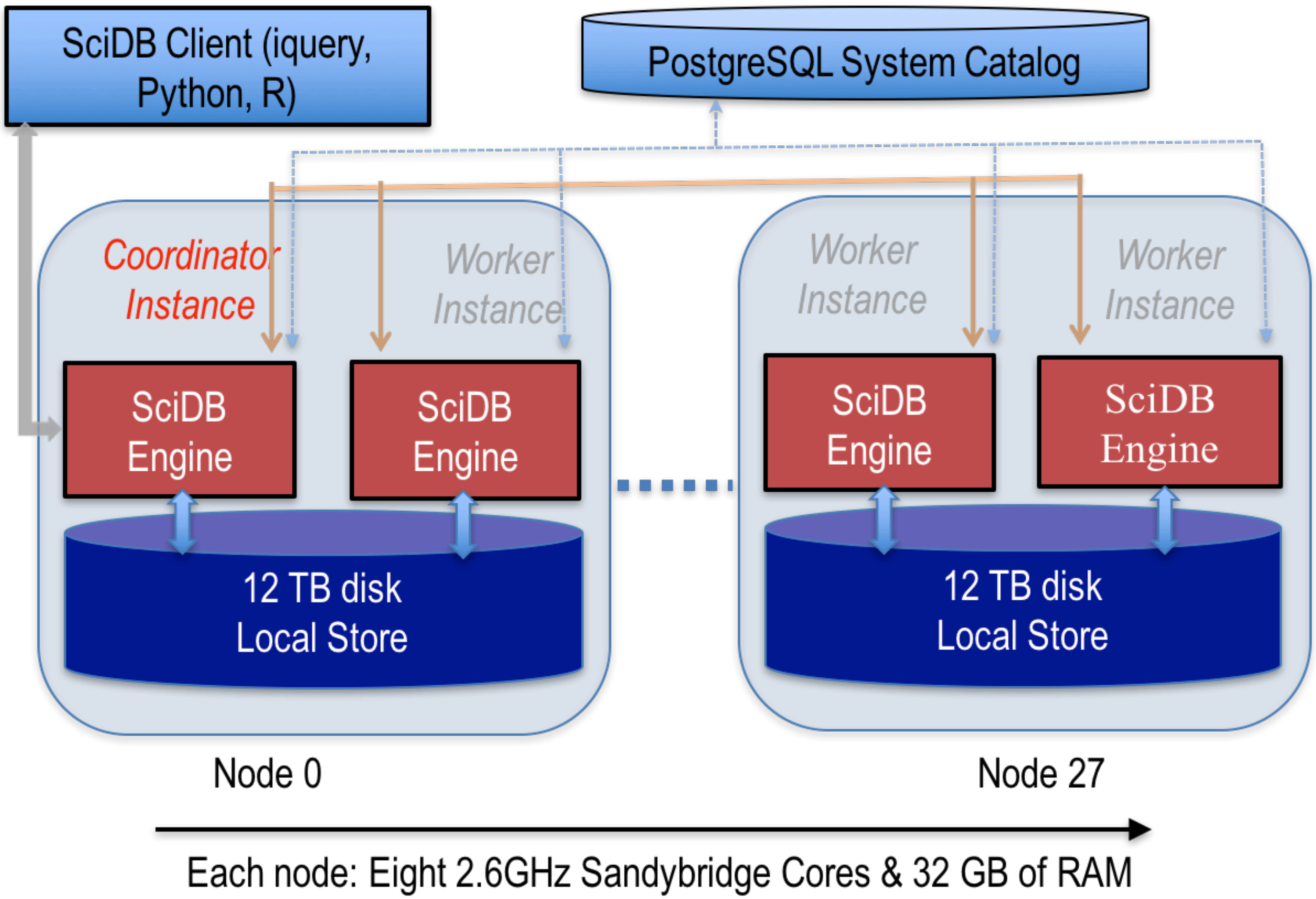
1. NASA Goddard Space Flight Center, Greenbelt, Maryland, USA
2. Science Systems and Applications, Inc., Greenbelt, MD, USA
3. Bayesics, LLC, Bowie, MD, USA
4. Paradigm4, Waltham, MA, USA

SciDB

SciDB is an all-in-one data management and advanced analytics platform that features:

- Complex analytics inside a next-generation parallel array database, i.e. not row-based or column-based like RDBMS's based on table data model.
    - Supports extensive and flexible algebra operators that can be efficiently "wired" together for more complex operations.
- Based on the "shared nothing architecture" for data parallelism, data versioning and provenance to support science applications.
- Open source.
- A better performer than Hadoop (MapReduce), 2-10 times faster, in almost all benchmarks that we have performed so far.
- Extensible through user-defined types, functions and operators

System Layout



Event Identification needs CCL

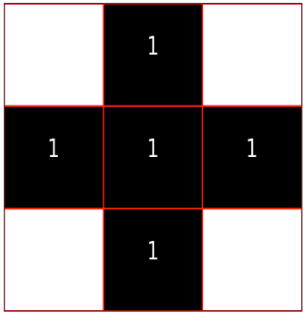
- Our interest: Identifying weather events as connected spatio-temporal regions in large multi-year climate datasets.
    - For example, snowstorm climatology in thirty-plus (30+) years of GEOS-5 MERRA (http://gmao.gsfc.nasa.gov/research/merra/)
    - Thresholding to generate a mask array (foreground & background mesh points).
    - Grouping of (adjacent) foreground mesh points into uniquely labeled connected regions, hence CCL

CCL implementation in SciDB

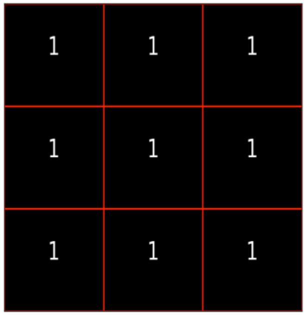
- Requirements:
  - Flexible array dimensionality
  - Flexible connectivity in any of the array dimensions
  - Periodic Boundary Conditions - BCs (due to need to support earth science applications)
- Algorithm - Weighted Quick Union with Half-Path compression (WQU) e.g. as in http://algs4.cs.princeton.edu/15uf/:
  - Efficient
  - Easy implementation considering the requirements.
- SciDB MemArray (data structure for handling larger-than-RAM arrays)
  - Used extensively for efficient indexing and searching throughout.

CCL implementation in SciDB

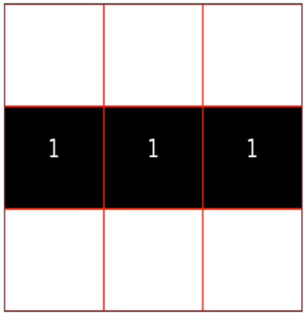
- Flexible connectivity achieved by an auxiliary array of the same dimension as the mask array but length 3 in each dimension e.g. for 2-D



4-connectivity



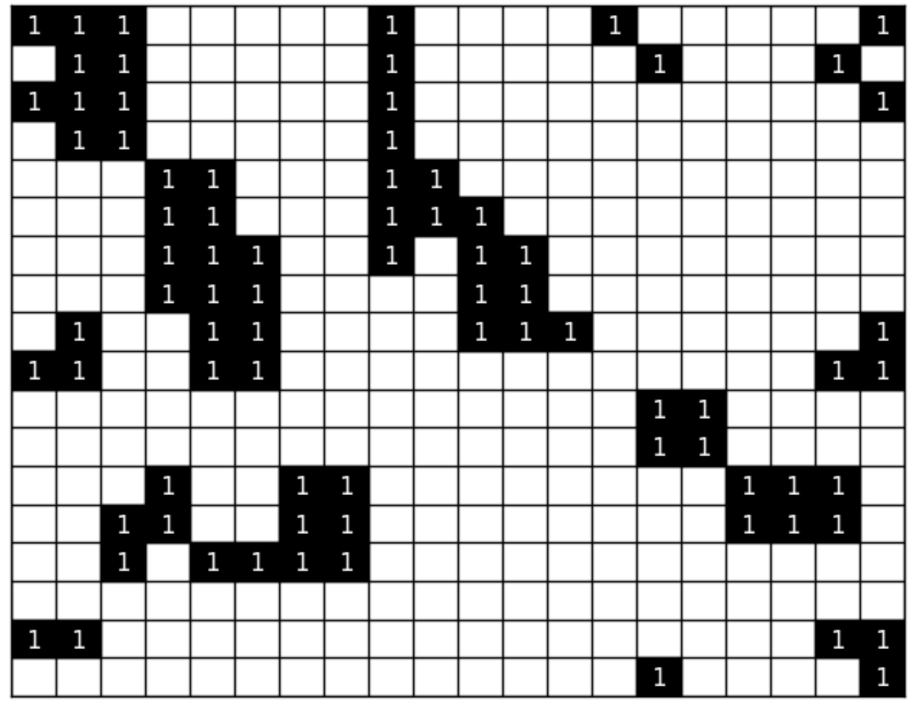
8-connectivity



X-connectivity

Preparation steps

1. Thresholding already done to prepare the mask array



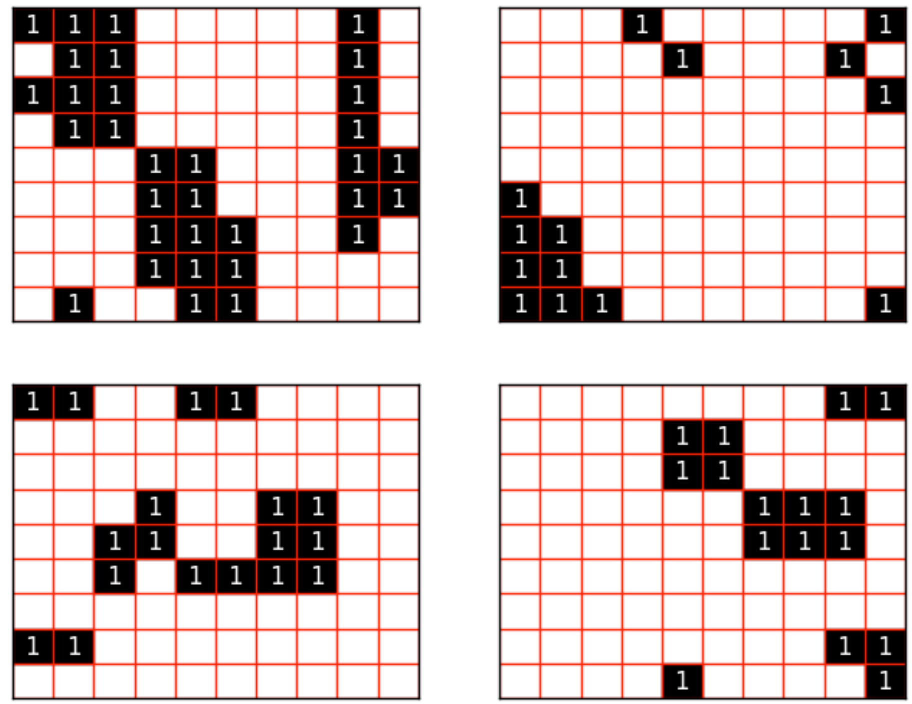
Preparation step

Partition mask array into SciDB chunks

- Schema supplied by the user to set chunk sizes (and overlap if any) e.g.

  <mask:int8> [x=0:17,9,1,y=0:19,10,1]

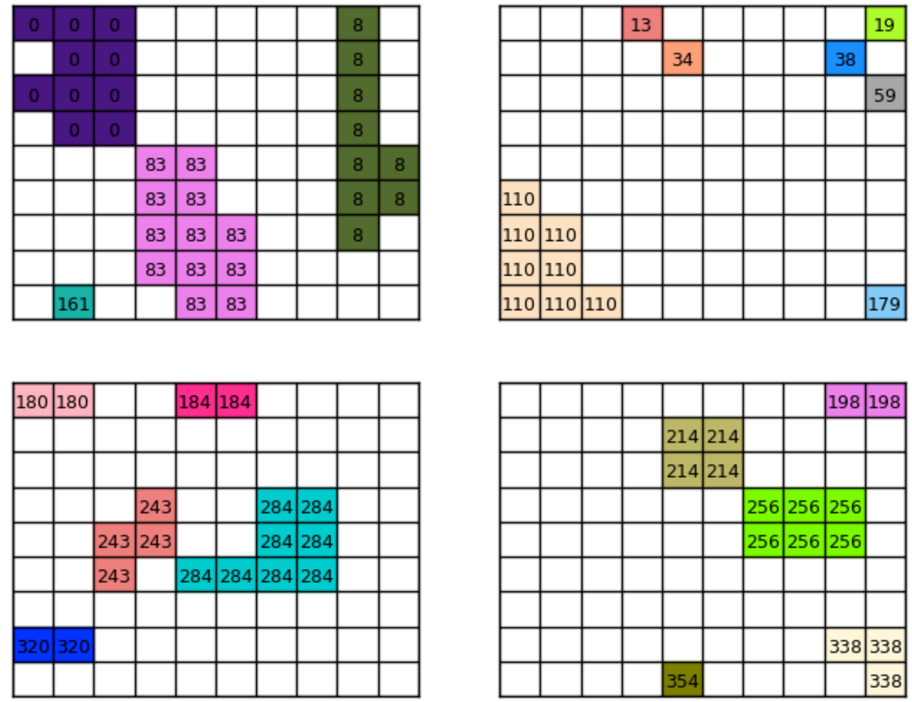
- Chunk assignment to instances is managed by SciDB using a heuristic hash function



Implementation steps

1. Stage 1 - CCL is computed by each instance, one chunk at a time

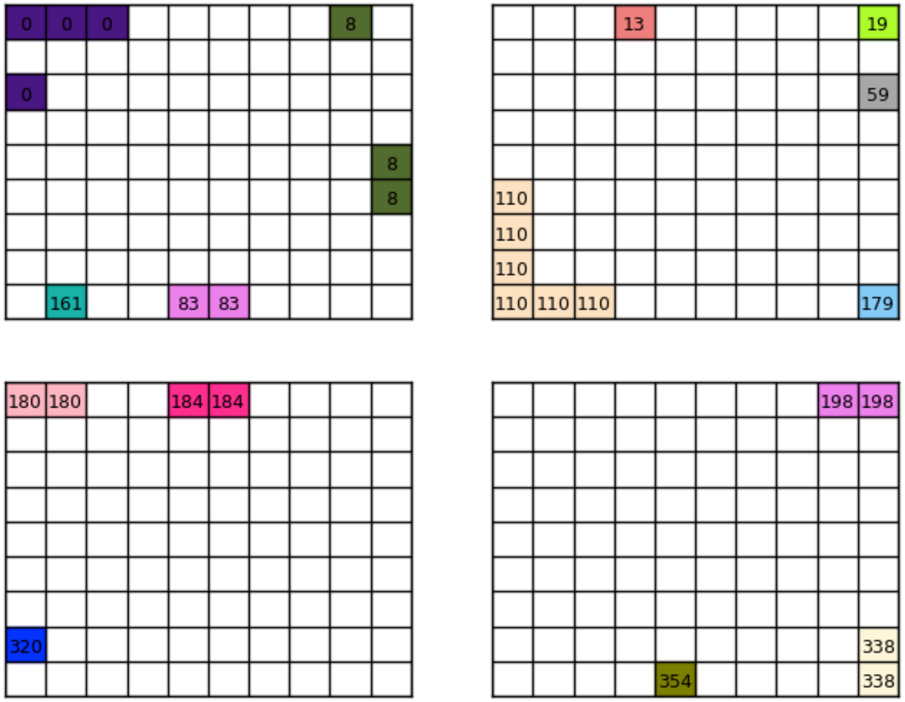
- Apply the auxiliary connectivity array (4-connectivity in this case) to each foreground mask (using “flattened” coordinates) to determine pairs of adjacent cells.
- Eliminate duplicate pairs
- Apply WQU to each pair to determine CCLs for the chunk
- Write chunk CCLs to MemArray



Implementation steps

2. Stage 2a - Equivalency resolution

- Write chunks' boundary labels to MemArray
- Replicate the MemArray on all instances so that each instance now has all boundary labels

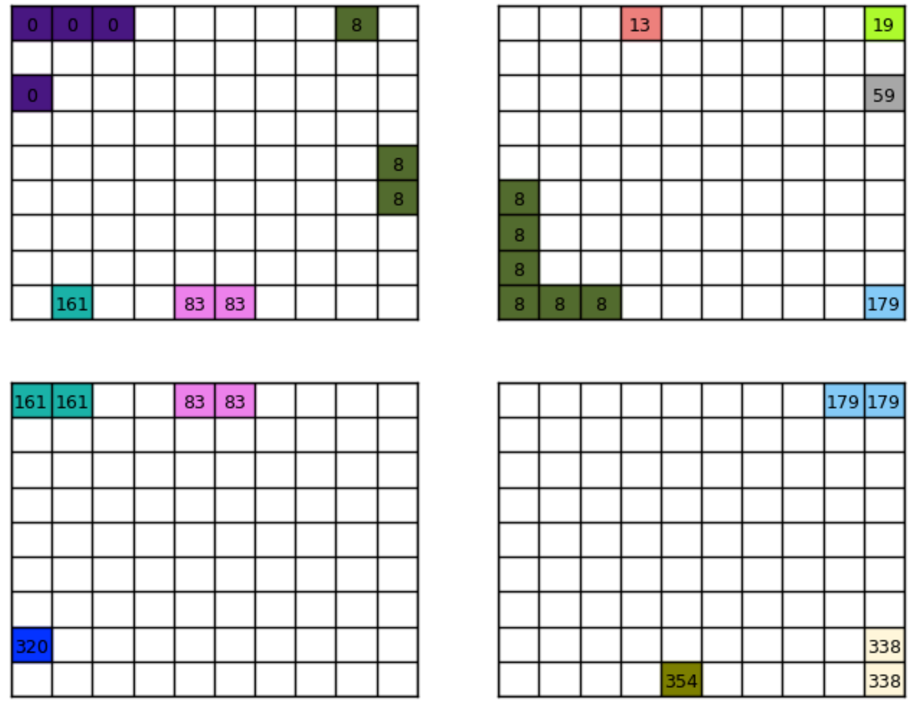


Implementation steps

3. Stage 2b - Equivalency resolution (contn'd)

On each instance

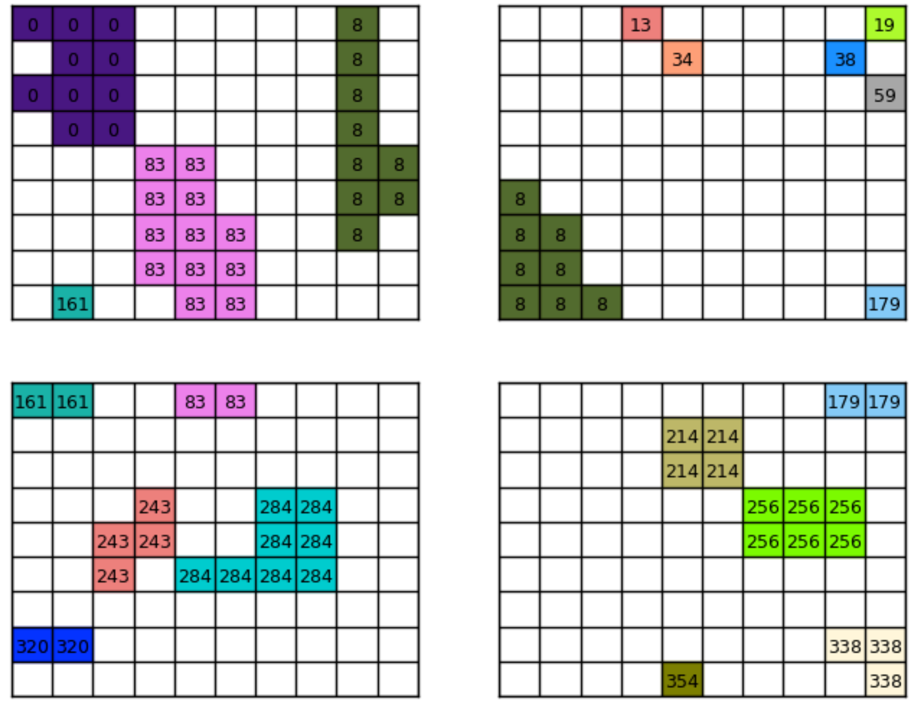
- Set up pairs of adjacent boundary labels
- Apply WQU to each pair to resolve equivalencies
- Write resolved boundary labels into 1-D MemArray using boundary labels from the Stage 1 as index (for later fast access in Stage 3)



Implementation steps

4. Stage 3 - Relabeling of Stage 1 using resolved labels from Stage 2b
   - On each instance, iterate over each chunk and relabel:

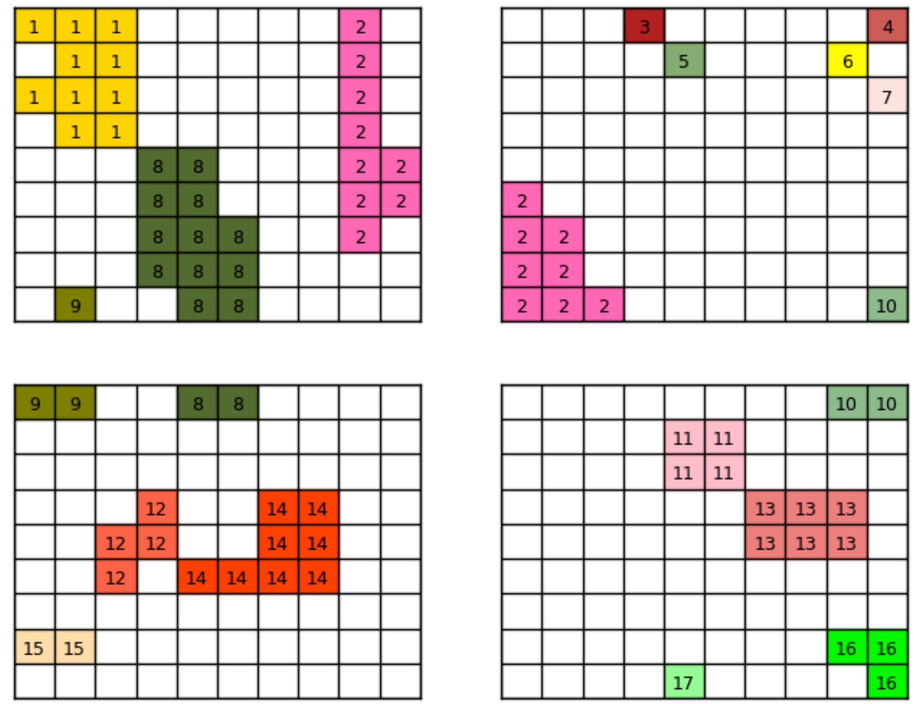
```
if stage2b_array[stage1_label] is not NULL :
        new_label <- stage2b_array[stage1_label]
else :
        new_label <- stage1_label
```



Implementation steps

5. Stage 4 – Final Relabeling

- Replace out-of-order labels from Stage 3 (represented by "flattened" coordinates from the original mask array) with sequential integers for legibility



Operator invocation

The operator is invoked (using the Array Functional Interface - AFL - interface via the iquery client):

```
iquery -anq “store(ccl(mask_array, con_array), ccl_array)”
```

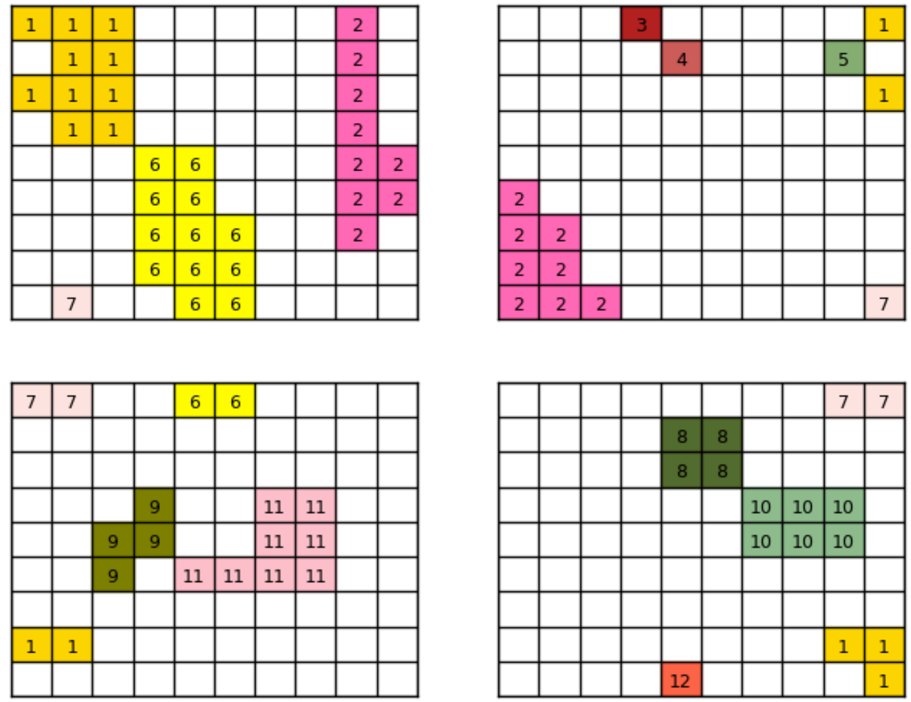
where

- ccl : the new CCL operator
- store : the SciDB native operator to “write” results into an array
- mask_array : the binary mask array
- con_array : the connectivity array
- ccl_array : array into which the resulting labels are written

Operator invocation with periodic BCs

Periodic BCs in both X- and Y-directions for our example:

iquery -aq “store(ccl(mask_array, con_array, 1, 2), ccl_array)”



Real life example

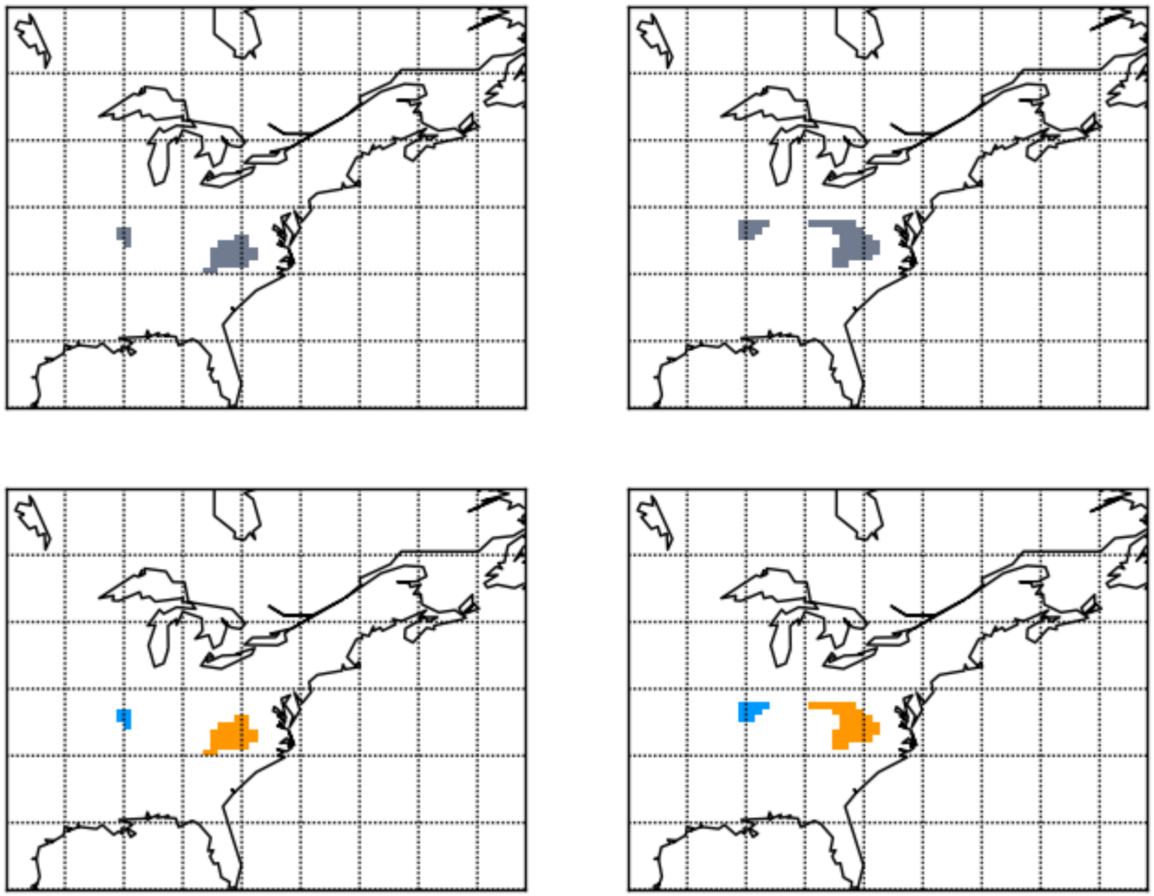
- Thirty-plus (30+) years of potential blizzard conditions on the globe
- Binary mask created by applying thresholds to GEOS-5 MERRA
- Total grid (array) size: $O(10^{11})$
- Foreground cells: $O(10^{8})$
- Spatio-Temporal CCLs (with periodic BCs in longitudinal direction): $O(10^{6})$

Sample results (from Winter 2010 blizzard over Eastern U.S.)

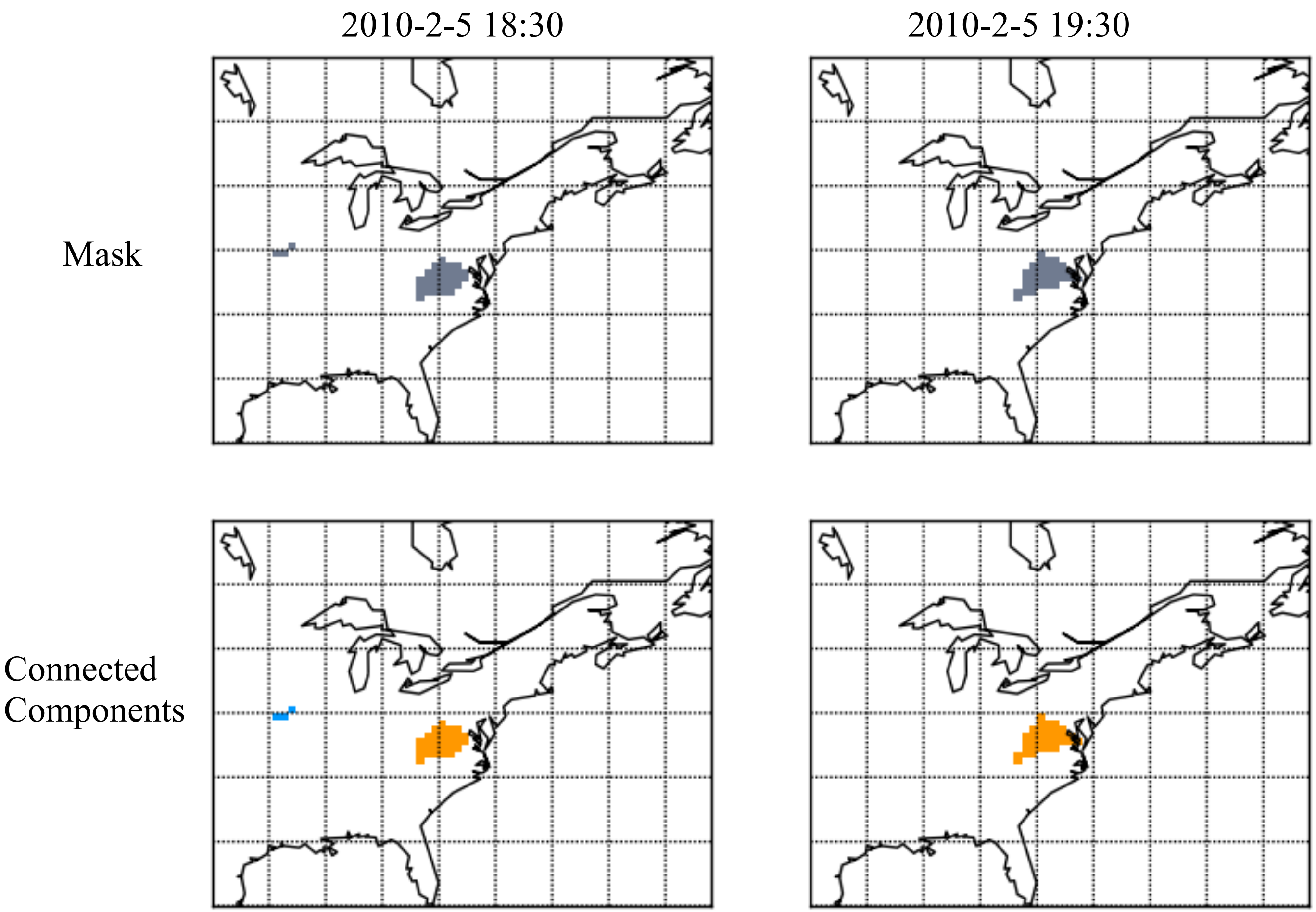
2010-2-5 13:30 | 2010-2-5 14:30

Mask

Connected Components



Sample results (from Winter 2010 blizzard over Eastern U.S.)



Wall time - seconds min(max)

	84 instances	56 instances	28 instances	14 instances
Stage 1 (local chunk labeling)	12(18)	21(27)	41(54)	88(114)
Stage 2a (replicate)	6(14)	6(12)	7(9)	9(12)
Stage 2b (resolve)	705(975)	705(935)	715(950)	725(940)
Stage 3 (relabeling)	33(55)	85(120)	115(160)	240(320)
Stage 4 (Final sequential labeling)	49(81)	90(125)	180(240)	364(475)

Comments

- Array schema - need to choose with care – significant effect on performance

  <ccl:int64> [t=0:323591,720,0,z=0:0,1,0,y=0:360,91,0,x=0:539,135,0]

- Good scaling for Stages 1, 3 & 4
- Boundary labels replication (Stage 2a) not terrible; dependent on array & schema
- As expected, no scaling for Stage 2b; further work needed to improve efficiency

Concluding remarks

- Array grows as time slices are added
- Expensive to re-compute labels for whole array each time new time slices are added
- Working on incremental implementation so that completed labels are excluded from new computation

Acknowledgement

This work was funded by the NASA Earth Science Technology Office (ESTO) Advanced Information Systems Technology (AIST) Program.