Efficient Parallel and Adaptive Partitioning for Load-balancing in Spatial Join

Jie Yang
PhD Advisor: Dr. Satish Puri
Department of Computer Science
Marquette University

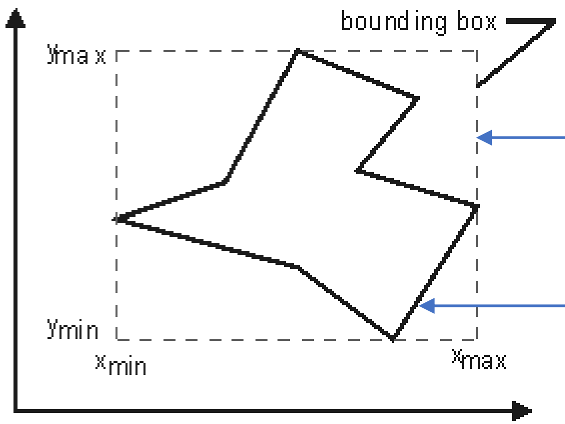
Overview

- Application: Geographic Information System (GIS)
  Spatial Database
- Novelties:
  - An adaptive partitioning method for spatial datasets (ADP)
  - A parallel version of ADP (ParADP)
- Experimental results

Background: Geometry and its bounding rectangle

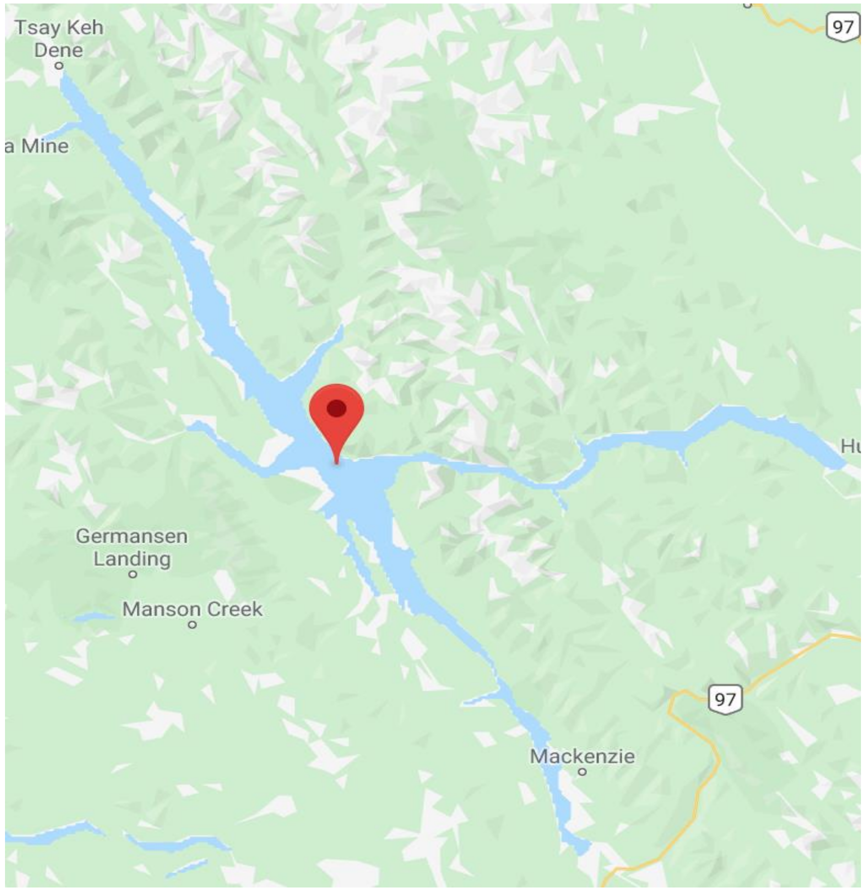
Outline of WI, USA

Outline of USA



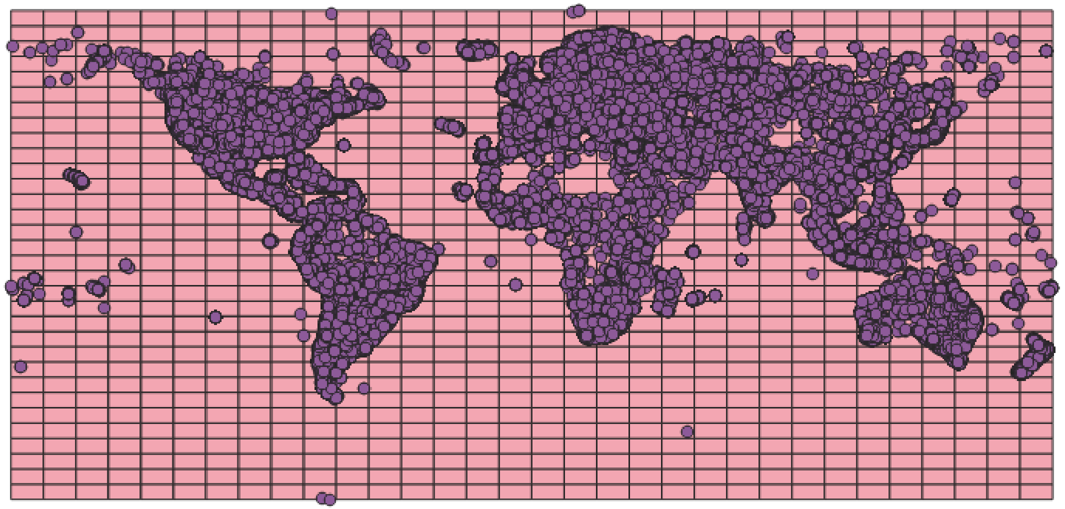
MBR = Rectangle
4 points in 2D

Polygon(20 30, 25 40, ..,)

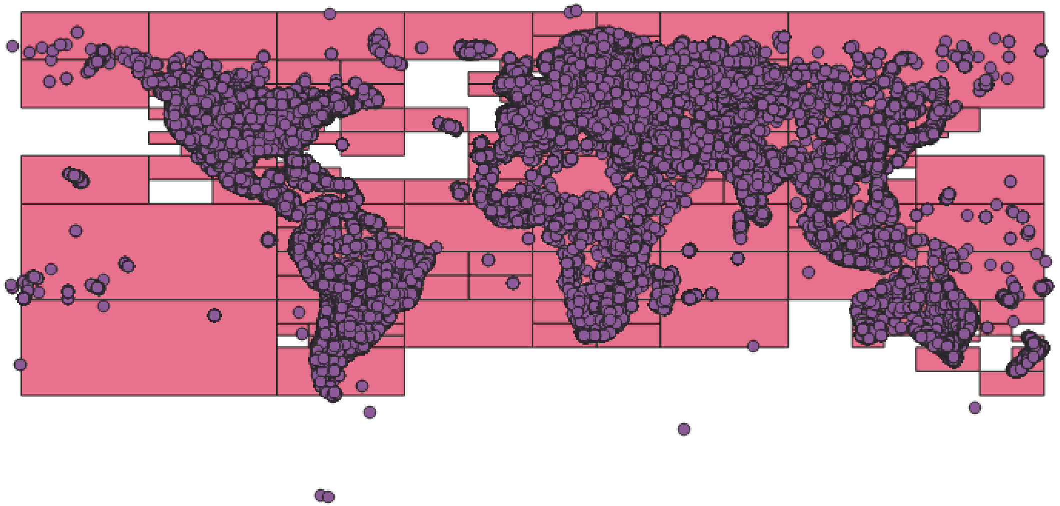


A lake with more than 100K points

Background: Spatial Data Partitioning



Uniform partitioning

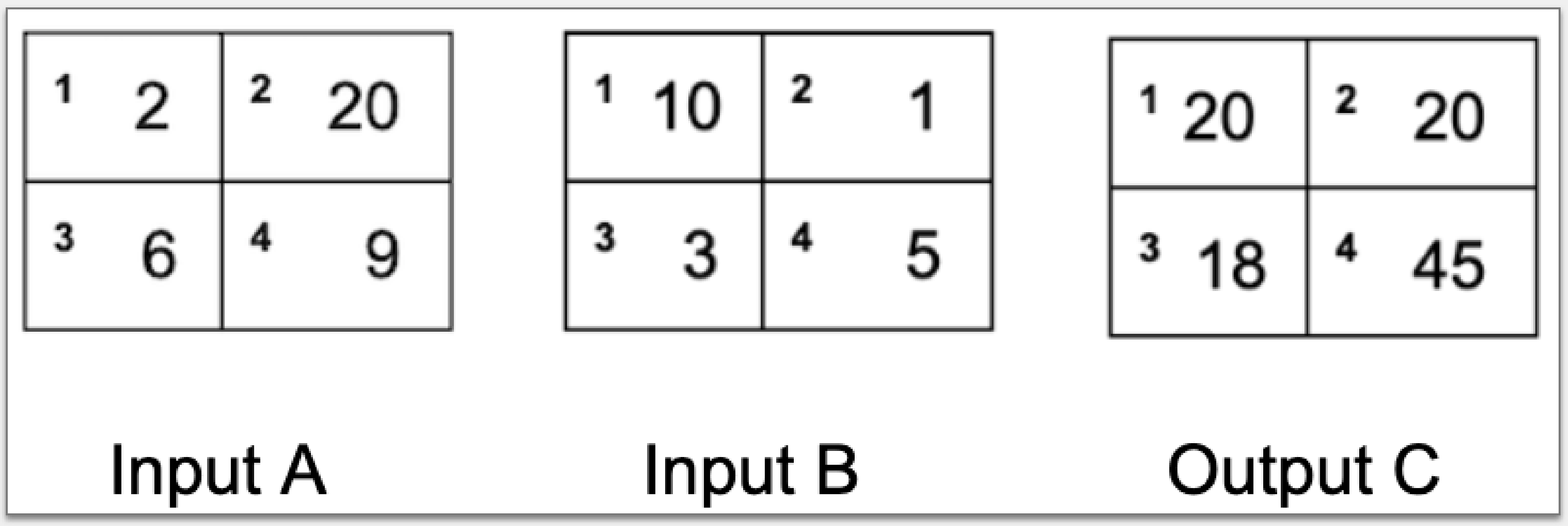


Quadtree partitioning

Background: Spatial Data Join

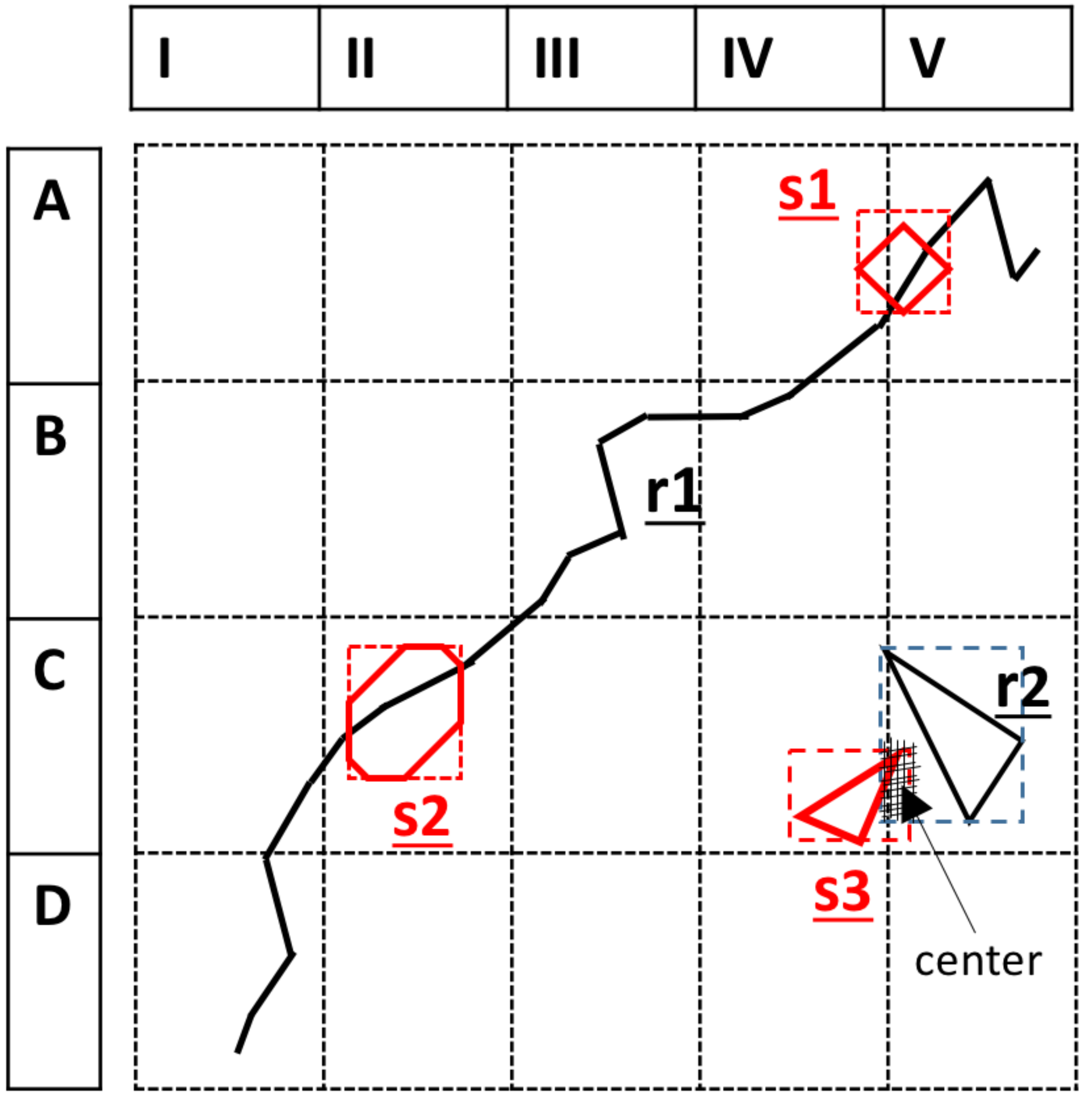
```
SELECT
    polyTable.[PolygonID] ,
    pointTable.[PointID]
FROM
    [PolygonTable_Name] polyTable WITH(INDEX([SPATIAL_INDEX_NAME]))
INNER JOIN
    [PointTabl_Name] pointTable
ON
    polyTable.Geog.STIntersects(pointTable.Geog) = 1
```

Existing Problems: Load Imbalance



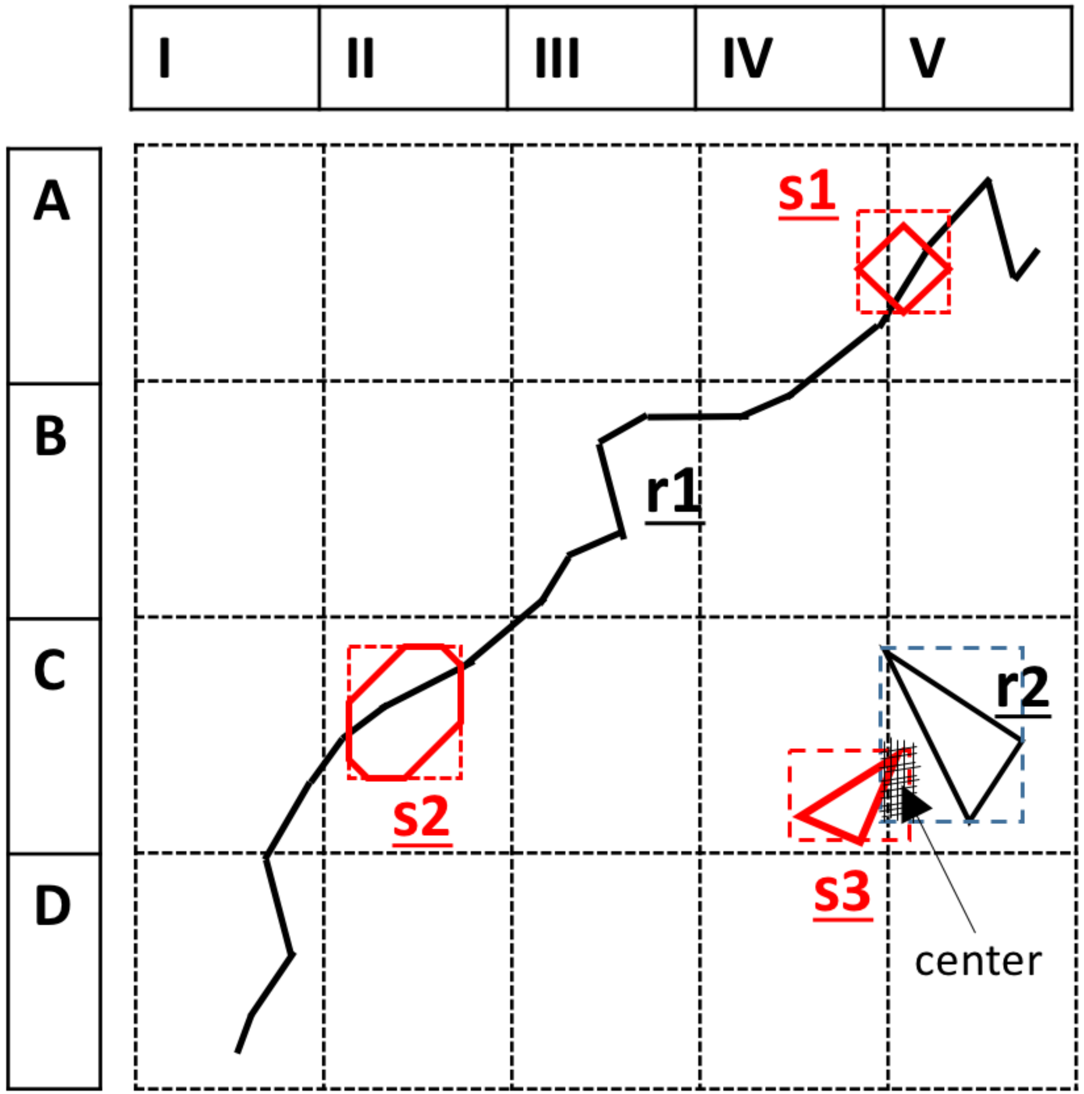
Number of geometries shown in each grid cell. The workload of a cell in grid C is the product of the number of geometries present in corresponding cells in A and B (e.g., workload in the fourth cell is 9*5).

Existing Problems: Duplications



Mapping of candidates to grid cells. Using current partitioning methods, r1 is kept in cell ids (I, C), (I, D), (II, C), (III, B), (IV, A), (IV, B), and (V, A).

ADP: Duplications Avoidance



Mapping of candidates to grid cells.(r1, s1),(r1, s2),(r2, s3) are candidates. Geometry r1 is not stored in cell ids (D, I), (C, I), (B, III), (B, IV), and (A, IV) even though it passes through these grid cells. Instead, r1 is stored in cells (C,II) and (A,V) because it is part of two candidates (r1, s1) and (r1, s2) only.

ADP: Load Balancing

- Partitioning based on both layers.
- Considering each geometry's weight.
- Using a quadtree approach
- Weight of a candidate

$$W = ((n + m)\log(n + m))$$

Where *n* and *m* are numbers of vertices in each geometry.

ADP algorithm

Algorithm 1 Algorithm for finding candidates

Input: Two collections of spatial objects R and S.
Output: Candidate set denoted by C,
Build Rtree index RI using MBRs of R
for MBR s_j in S **do**
    $results \leftarrow RI.query(s_j.MBR)$
    for r_k in $results$ **do**
        Find the intersection of $r_k.MBR$ and $s_j.MBR$
        Calculate center point of intersection denoted by p_{jk}
        Calculate weight w_{jk} using weight equation.
        $C \leftarrow C \cup tuple(r_k, s_j, p_{jk}, w_{jk})$
    end for
end for

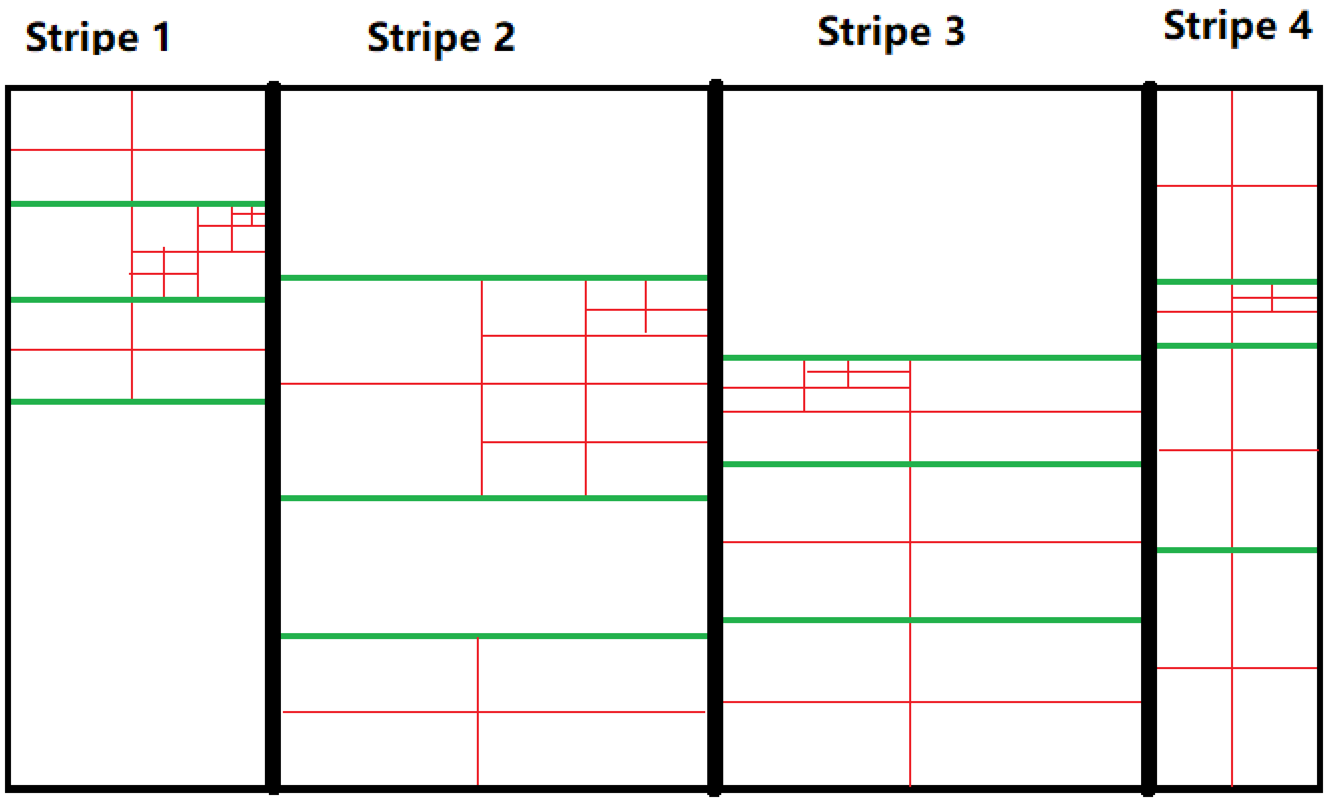
Parallel Adaptive Partitioning (ParADP)

- ADP is slow.
- Each step in ADP relays on the result from its prior step.
- Load balancing problem.

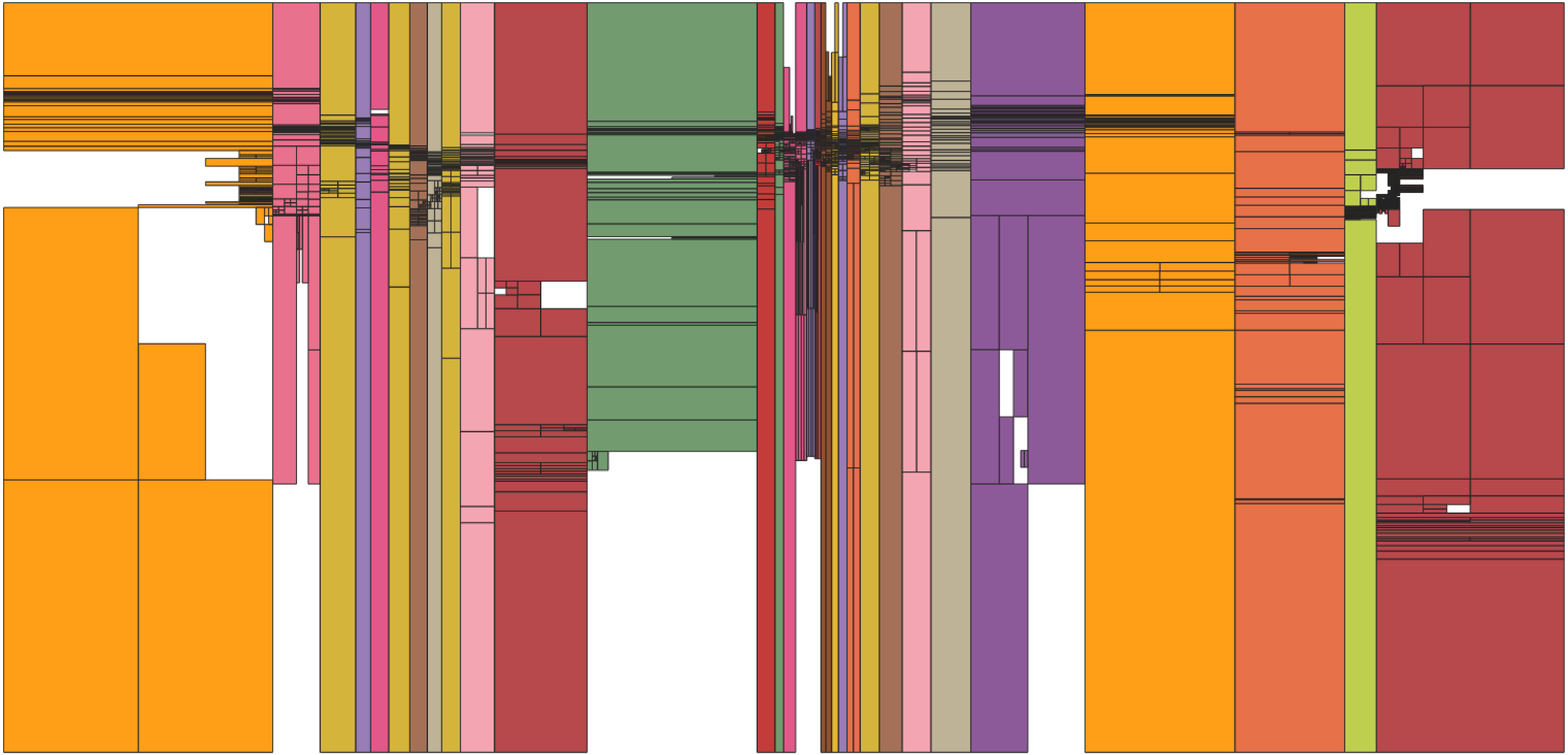
ParADP

- Load balancing problem for ParADP is a chicken-egg problem.
- Each compute node takes part of data from two input datasets.
- Using *Parallel Sorting by Regular Sampling* to sort candidates.
- Utilizing shared memory in a distributed memory architecture.

ParADP

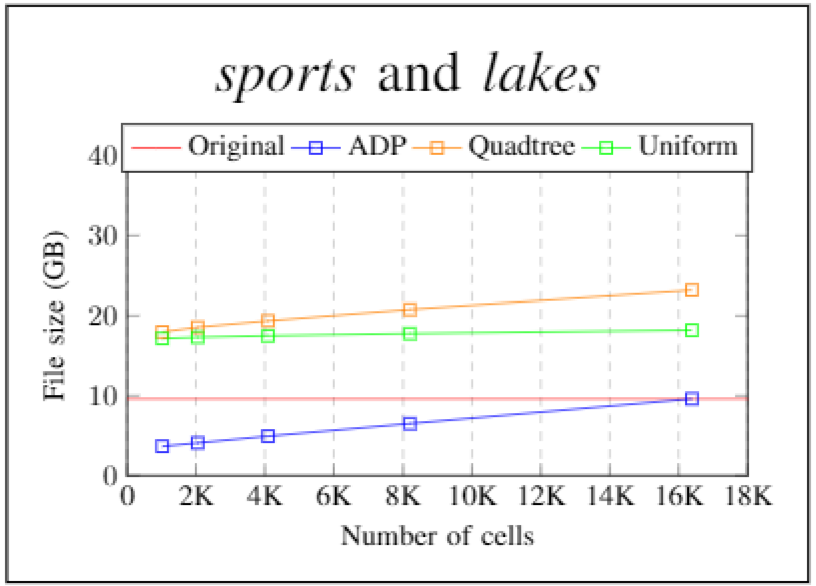


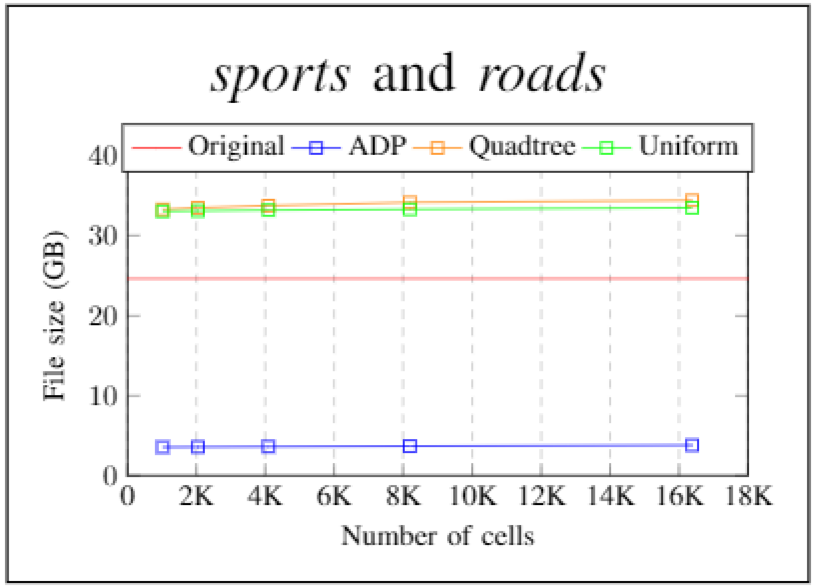
ParADP using 4 compute nodes with 4 cores in each node.
Longitudinal thick black lines are for rearranging data between nodes.
The green lines are for data distribution within a node.

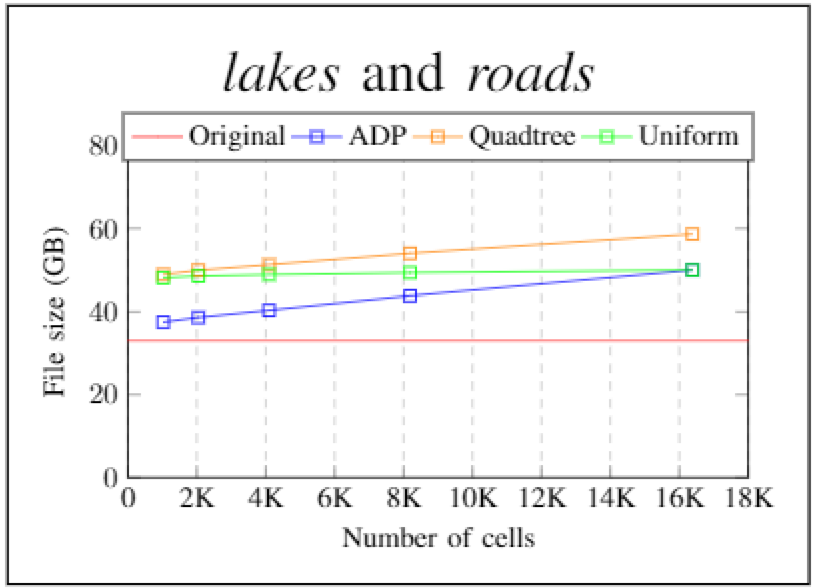


Parallel partitioning of two large datasets into 8192 grid cells using ParADP

Experimental: Storage Space



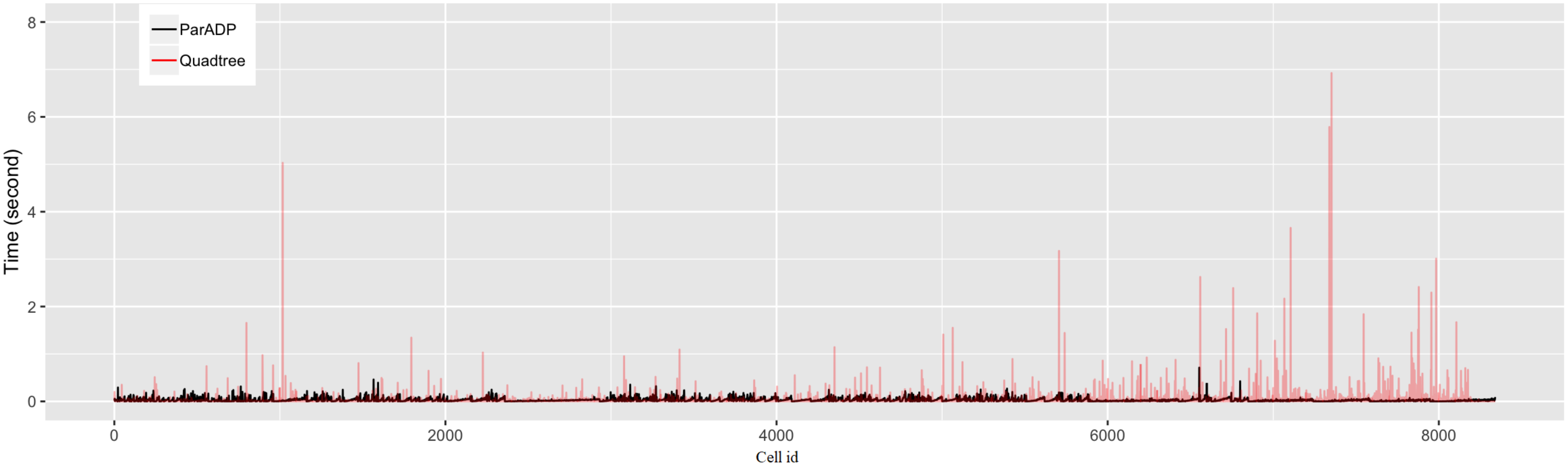




*Attributes of the datasets

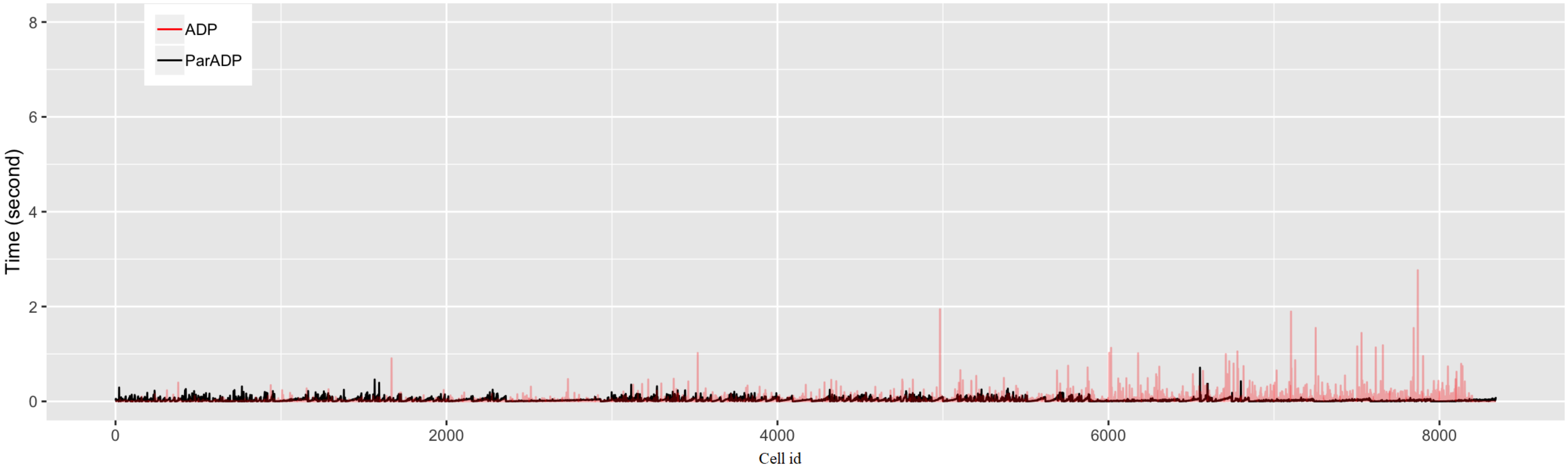
Name	Type	#Geometries	File size
sports	Polygons	1.8 M	590 MB
lakes	Polygons	8.4 M	9 GB
parks	Polygons	10 M	9.3 GB
Roads	Polylines	72 M	24 GB

Experimental: Load balancing



Execution time of applying Intersectson on 8192 different cells of the partitioned parks and sports.
The data sets are partitioned into 8192 cells by ParADP and Quadtree partitioning.

Experimental: Load balancing



Execution time of applying Intersectson on 8192 different cells of the partitioned parks and sports.
The data sets are partitioned into 8192 cells by ADP and ParADP.

Experimental: ParADP Strong & Weak Scaling

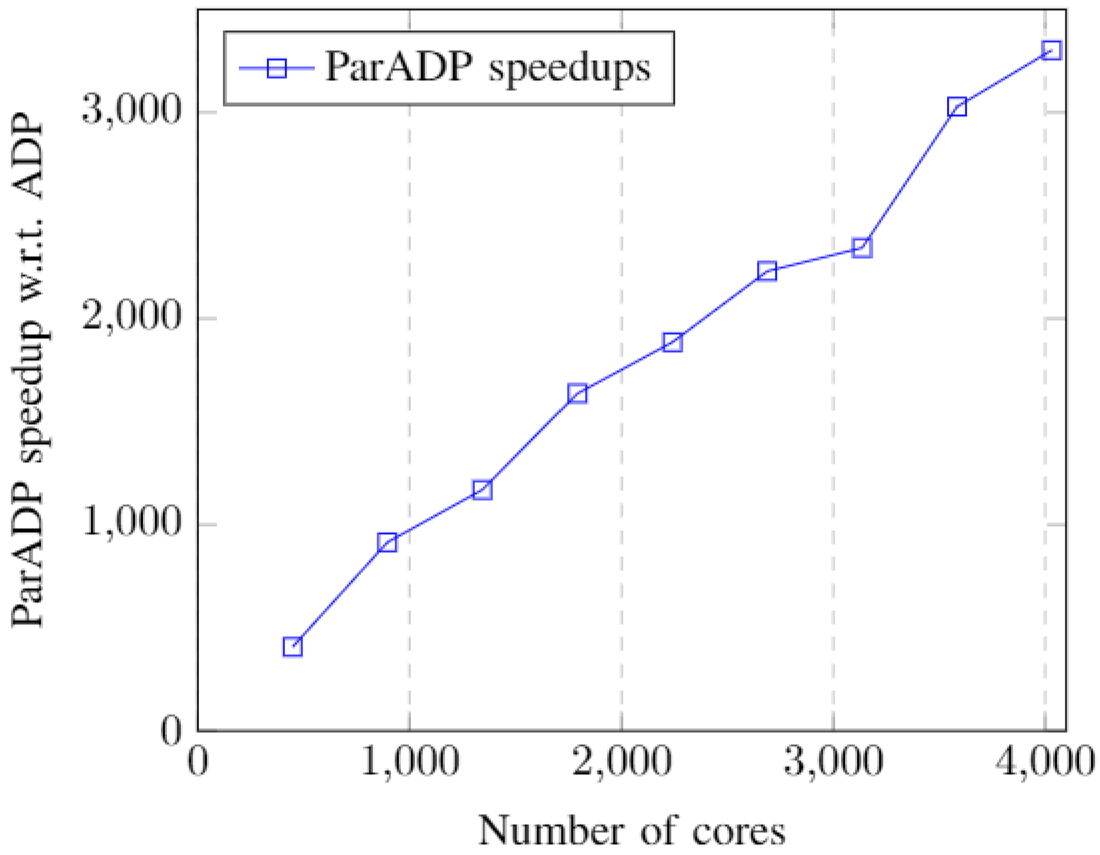
PARADP EXECUTION TIME FOR STRONG SCALING

R	S	Partition Number	Nodes	$T_{total}(s)$	$T_s(s)$
roads	parks	65536	16	55.56	1.82
roads	parks	65536	32	24.69	0.73
roads	parks	65536	48	19.32	0.64
roads	parks	65536	64	13.80	0.38
roads	parks	65536	80	11.98	0.32
roads	parks	65536	96	10.13	0.15
roads	parks	65536	112	9.64	0.15
roads	parks	65536	128	7.46	0.14
roads	parks	65536	144	6.84	0.14

PARADP EXECUTION TIME FOR WEAK SCALING

R	S	Candidates	Nodes	Grid cells	$T_{total}(s)$
roads	*parks*	75 M	16	8192	49.32
roads	2**parks*	150 M	32	16384	38.63
roads	3**parks*	225 M	48	24576	36.89
roads	4**parks*	300 M	64	32768	41.45
roads	5**parks*	375 M	80	40960	40.13
roads	6**parks*	450 M	96	49152	32.26
roads	7**parks*	525 M	112	57344	30.77
roads	8**parks*	600 M	128	65536	31.70
roads	9**parks*	675 M	144	73728	30.56

Experimental: Speedups of ParADP w.r.t. ADP



Time complexity analysis shows that ParADP is *NP* faster than ADP. The experimental result approves it.

Specific analysis is in this paper, Section IV.B.

Reference

- M. Deveci, S. Rajamanickam, K. D. Devine and U. V. Catalyurek,“Multi-Jagged: A Scalable Parallel Spatial Partitioning Algorithm,” inIEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 3,pp. 803-817, 1 March 2016.
- Ahmed Eldawy, Louai Alarabi, and Mohamed F. Mokbel. 2015. “Spatialpartitioning techniques in SpatialHadoop,” Proc. VLDB Endow. 8, 12(August 2015), 1602-1605.
- Satish Puri and Sushil K Prasad. 2015. “A parallel algorithm for clippingpoly-gons with improved bounds and a distributed overlay processingsystem usingmpi,” In2015 15th IEEE/ACM International Symposium onCluster, Cloud and Grid Computing. IEEE, 576–585

Thank you!