UNB

# Query Compilation based Distributed Morsel-driven Parallel Spatial Query Processing

Rahul Sahni, Xiaozheng Zhang, Sudip Chatterjee, Suprio Ray
University of New Brunswick, Fredericton

Compiler-Driven Performance Workshop @ CASCON, 2024

CAS–Atlantic

UNB

# Outline

- Motivation
- Background
- Our approach
- Evaluation
- Conclusions



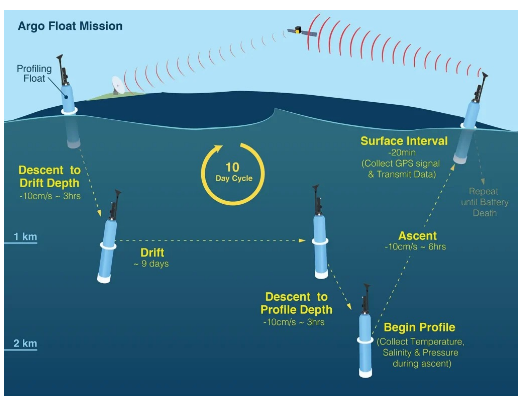
CAS–Atlantic

UNB

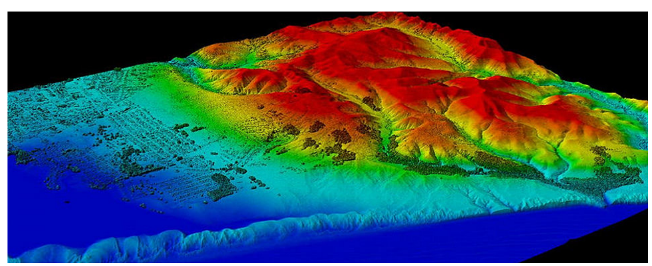
# Motivation

- Spatial Data is increasing at a rapid rate.
- Map based app – Google Maps
- Geo-Social app - Facebook
- Ocean data – Argo Data
- Remote sensing data
- Agriculture Tech – Picketa Systems

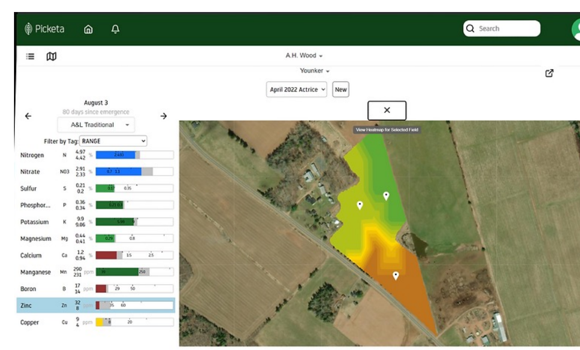
[1]




[2]



[3]




[4]

[1] https://www.flaticon.com/free-icon
[2] https://oceanservice.noaa.gov/facts/remotesensing.html
[3] https://globalocean.noaa.gov/research/argo-program/
[4] https://www.picketa.com/

CAS–Atlantic

# Motivation

- RDBMS are popular, partly due to SQL.
- RDBMS follow Volcano Model for query execution.
- Main focus was to minimize disk I/O and CPU utilization was less important, bottleneck on modern CPUs
- "To go 10X faster, the engine must execute 90% fewer instructions and yet still get the work done. To go 100X faster, it must execute 99% fewer instructions" - Hekaton

UNB

# Motivation

- Query Compilation based query processing, offers significant benefits but complicated to incorporate.
- PostgreSQL supports just-in-time (JIT) query compilation for tuple materialization and expression evaluation only.
- Query compilation for spatial workload could add more complexities. LB-2 Spatial proposed a generative query compilation approach but only support MBR based spatial query execution and is based on single node.



CAS–Atlantic

UNB

# Outline

- Motivation
- Background
- Our approach
- Evaluation
- Conclusions



CAS–Atlantic

UNB

# Models of Query Execution

- Pull-Based Query Execution
  - Volcano/Iterator model

- Push-Based Query Execution

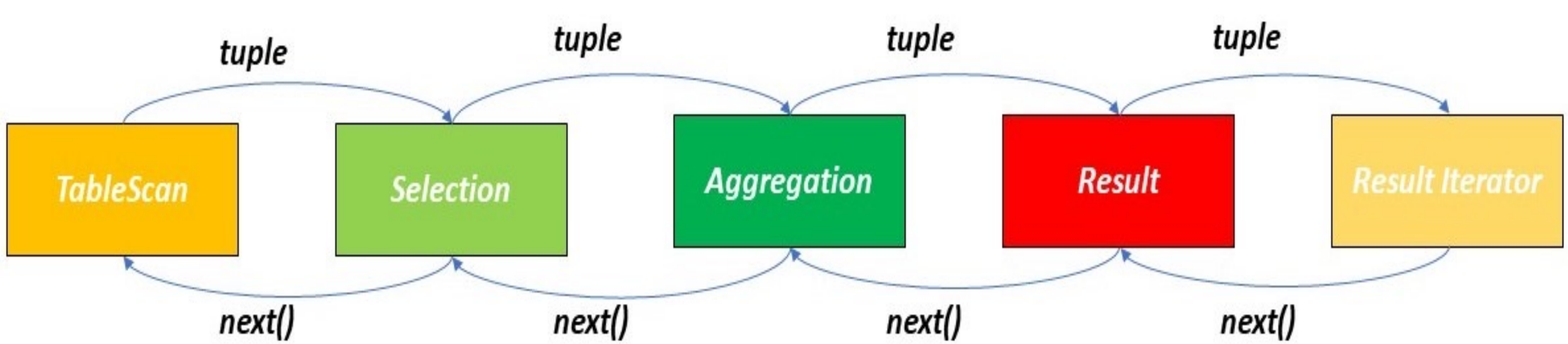


CAS–Atlantic

UNB

# Pull-Based Model

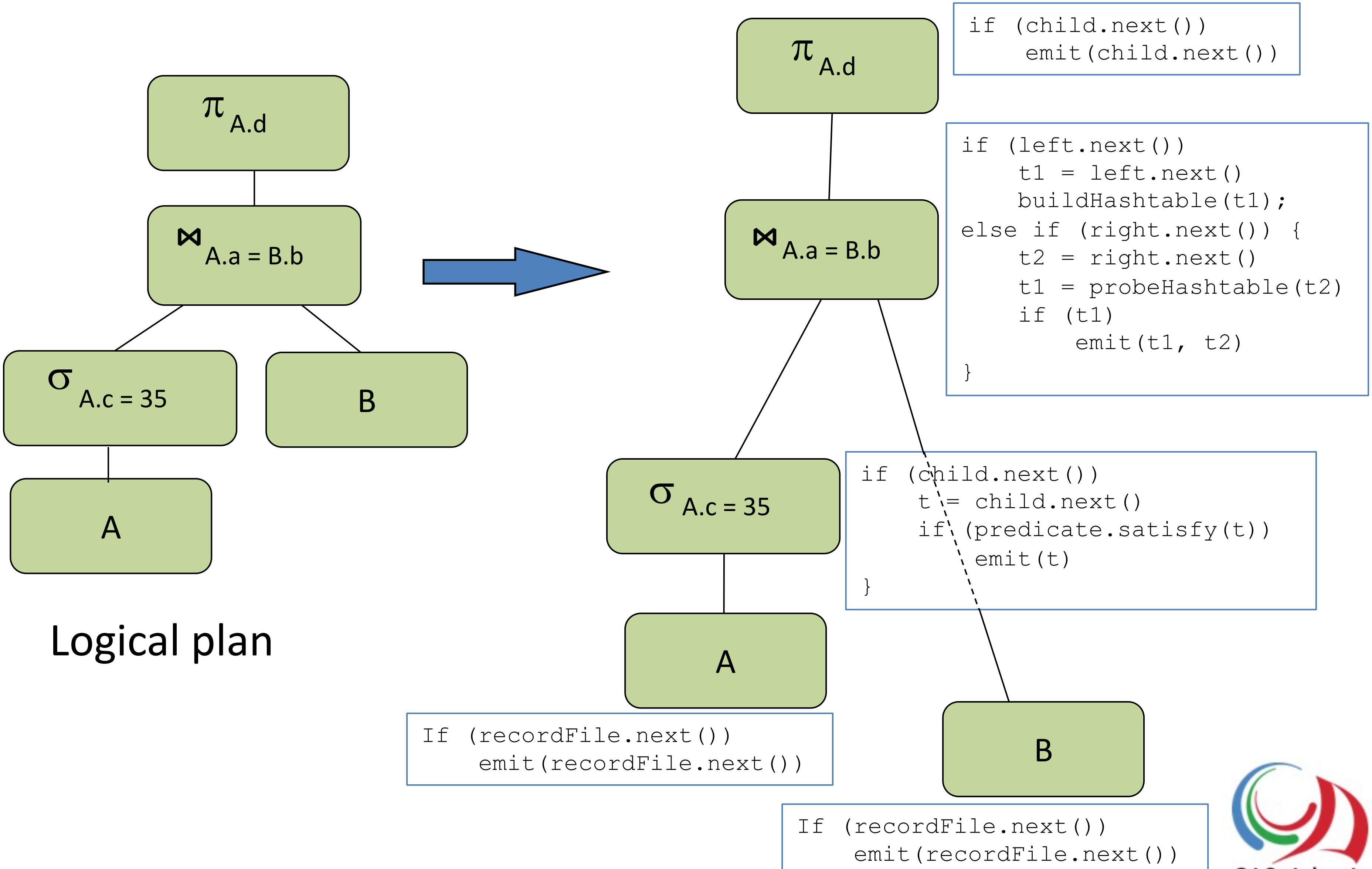
- Tuple-at-a-time
- Each operator implements a common interface:
  - `open()`
  - `next()`
  - `close()`






CAS–Atlantic

# Sample query - Pull-Based



UNB

# Pull-Based Model - limitations

- Millions of virtual function calls
- Control flow constantly changes between operators
- Generated code can be too big with many conditions and branches
- Branch prediction and cache locality suffer
- Solution?
  - Push-Based Model

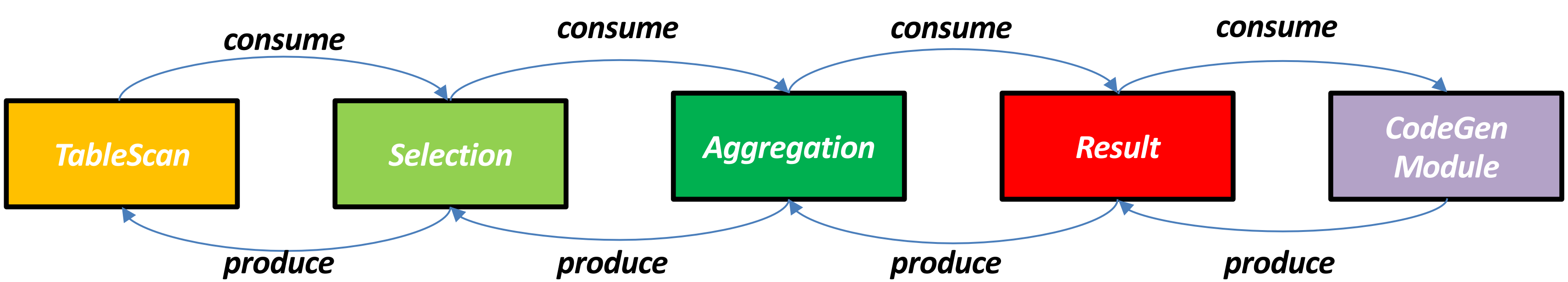


CAS–Atlantic

UNB

# Push-Based Model

- Each operator has two interfaces:
  - `produce()`: asks the operator to produce tuples and push it up
  - `consume()`: accepts the tuples and pushes further up
- The functions are not really called
  - they only exist conceptually during code generation



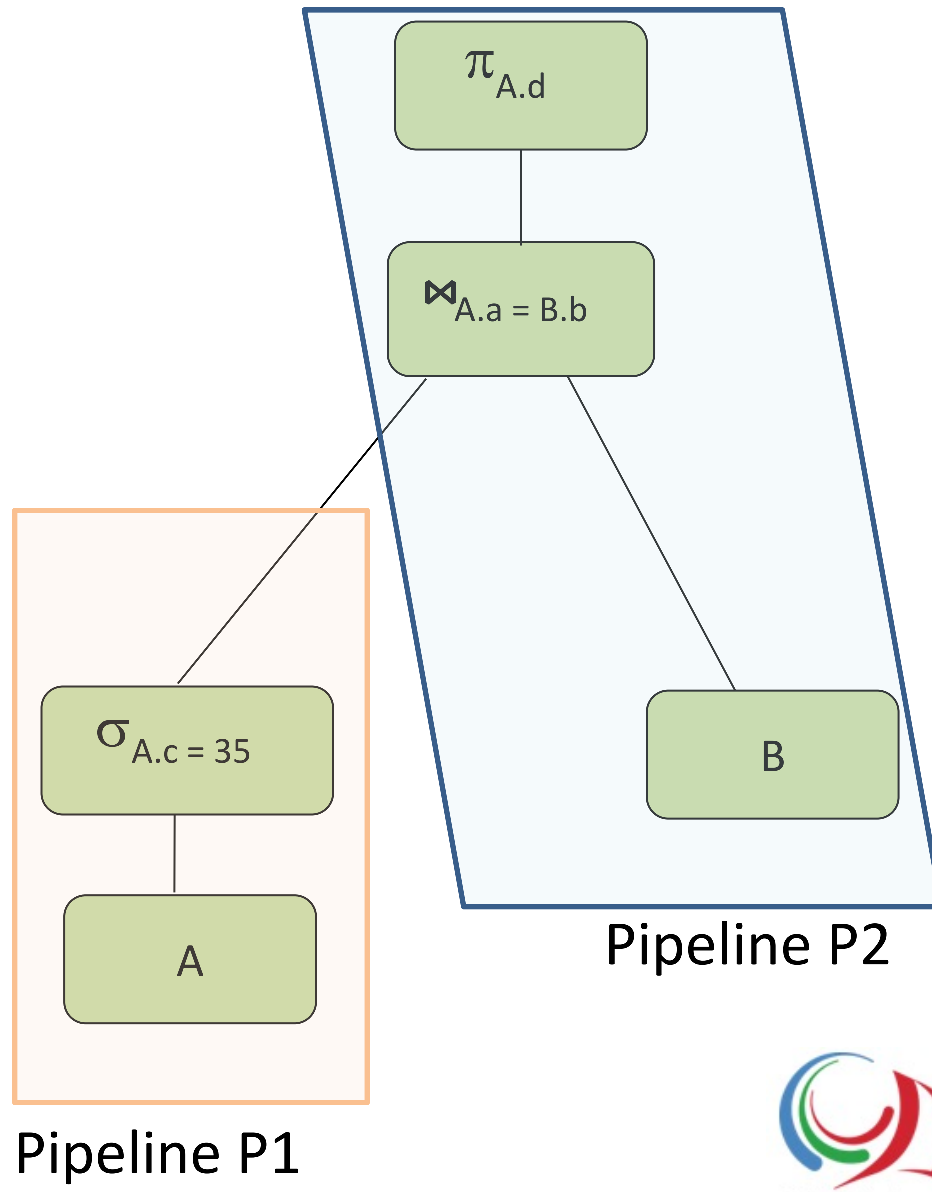



CAS–Atlantic

UNB

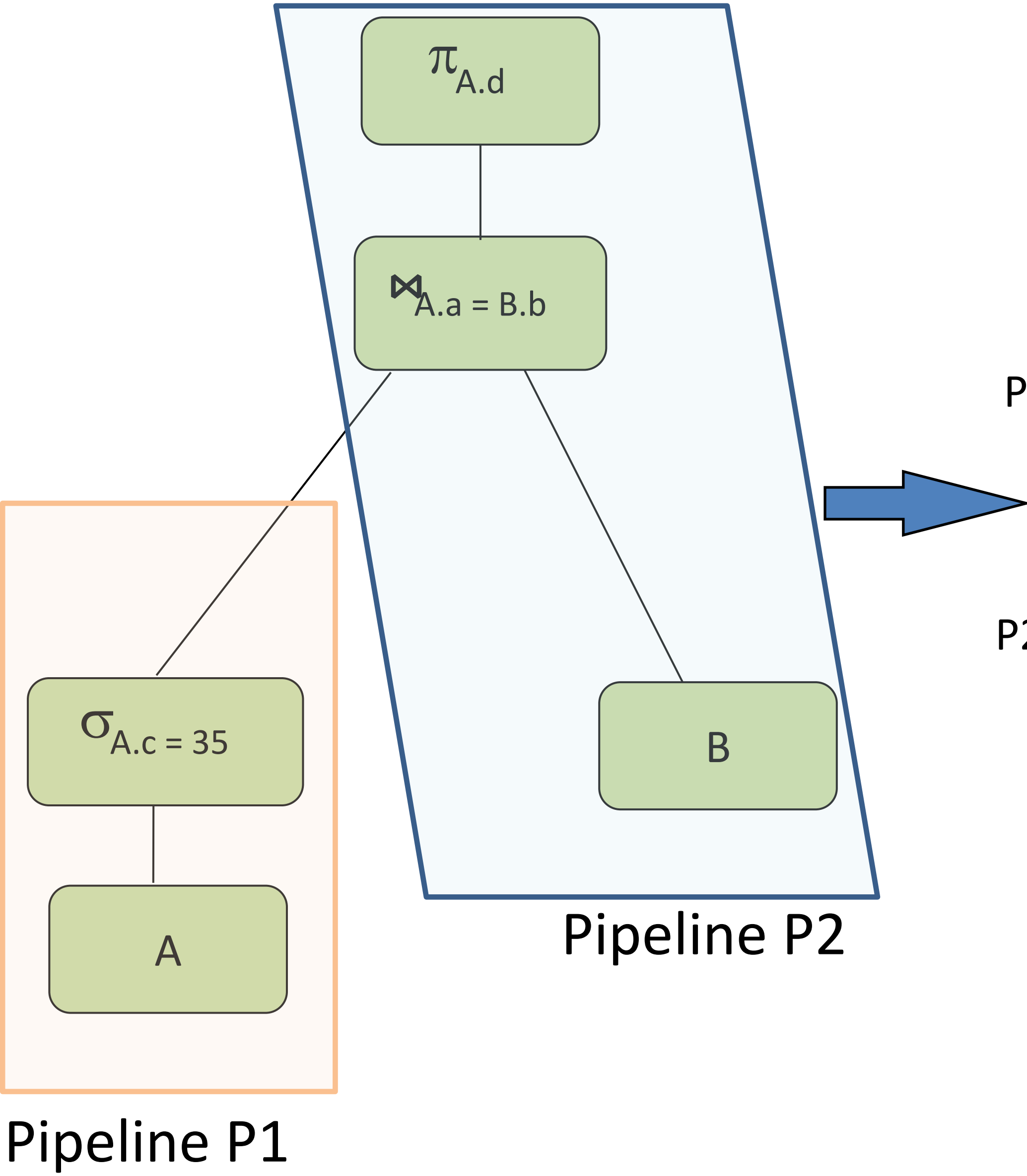
# Push-Based Model

- Operator boundaries are determined by conceptual “pipelines”
  - Instead of iterating, we push up the pipeline
- Within a pipeline, a tight loops performs a number of operations
- Data is taken out at a pipeline breaker and materialized into the next




CAS–Atlantic

# Sample query - Push-Based




P1

```
for t in A:
    if (t.c == 35)
        materialize t in Hashtable HT
```

P2

```
for t2 in B:
    t1 = probeHashtable(HT, t2)
    if (t1)
        emit(t1.d)
```

UNB

# Query Compilation – Using Push-based Model

- No virtual functions calls
- Better data locality – “real code inlining”
  - Operators act upon data in CPU registers
- Operator fusion
- Code specialization
- Compiler optimizations, like loop unrolling and loop-invariant code motion



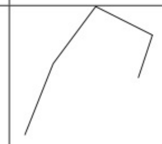
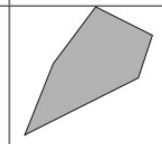
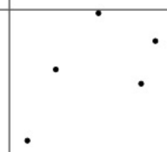
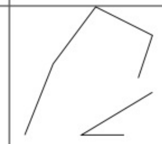
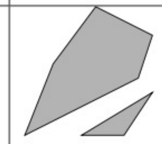
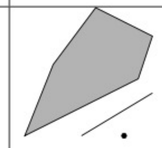
CAS–Atlantic

# Spatial Data

- Data representing location, shape, relationship with other object in a space
- Vector Data : points, lines and polygons
- Raster Data: grid data, with each grid represent some value

UNB

# Spatial Data

| Geometry Type | WKT representation | Geometry |
|---|---|---|
| Point | Point(10,10) |  |
| LineString | LineString(10 10, 20 35, 35 55, 55 45, 50 30) |  |
| Polygon | Polygon((10 10, 20 35, 35 55, 55 45, 50 30, 10 10)) |  |
| MultiPoint | MultiPoint((10 10),(20 35),(35 55),(55 45),(50 30)) |  |
| MultiLineString | MultiLineString((10 10, 20 35, 35 55, 55 45, 55 30),(45 30, 30 10)) |  |
| MultiPolygon | MultiPolygon(((10 10, 20 35, 35 55, 55 45, 55 30, 10 10)), ((55 25, 30 10, 45 10)) |  |
| Geometry Collection | GeometeryCollection(Point(45, 10), Linestring(55 25, 30, 10), Polygon(10 10, 20 35, 35 55, 55 45, 55 30, 10 10)) |  |



CAS–Atlantic

UNB

# Spatial Functions

| Function Type | Function Name | Desciption |
|---|---|---|
| Conversion | ST_GeomFromText | Converts a WKT to geometry |
| Conversion | ST_GeomFromWKB | Converts a binary object to geometry |
| Conversion | ST_GeomFromGeoJSON | Converts a GeoJSON to geometry |
| Geometric | ST_Length | Calculates length of a LineString or MultiLineString |
| Geometric | ST_Area | Calculates area of a geometry |
| Geometric | ST_Perimeter | Calculates 2D perimeter of a geometry |
| Geometric | ST_Distance(A,B) | Calculate the shortest distance between geometry A and B |
| Relational | ST_Equals(A,B) | Checks if geometry A is identical to geometry B |
| Relational | ST_Intersects(A,B) | Checks if geometry A and geometry B's boundary or interiors intersects |
| Relational | ST_Disjoint(A,B) | Checks if geometry A and geometry B are disjoint, i.e. if they have no point in common |
| Relational | ST_Crosses(A,B) | Checks if geometry A and geometry B's interior intersects with the interior of A at some but not all points |
| Relational | ST_Overlaps(A,B) | Checks if geometry A and B overlaps. Two geometries overlap if they have the same dimension, their interiors intersect in that dimension and each has at least one point inside the other |
| Relational | ST_Touches(A,B) | checks if geometry A and B touch their boundaries, but do not intersect in their interior |
| Relational | ST_Within(A,B) | checks if geometry A is completely within geometry B |
| Relational | ST_DWithin(A,B, x) | checks if geometry A is with x meters of geomtery B |



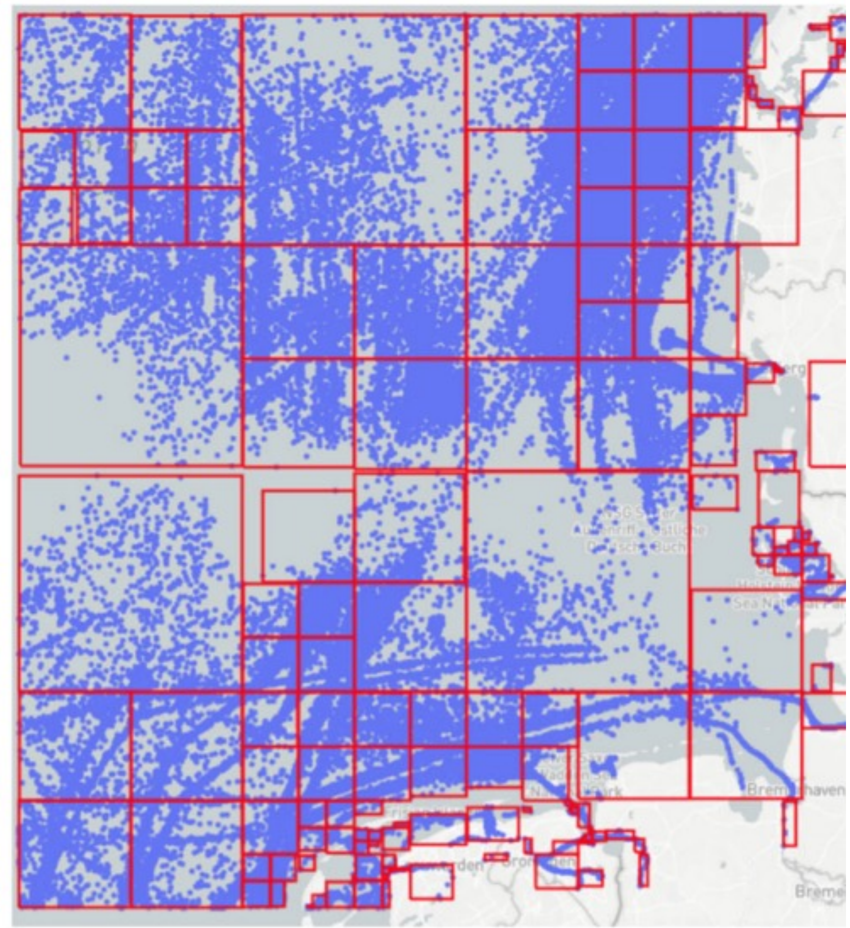
CAS–Atlantic

UNB

# Spatial Partitioning

- Dividing a spatial region into sub-regions (partitions/tiles)

- Efficiently organize and process the data



* Image from Filipiak, Dominik & Węcel, Krzysztof & Stróżyna, Milena & Michalak, Michał & Abramowicz, Witold. (2020). Extracting Maritime Traffic Networks from AIS Data Using Evolutionary Algorithm. Business & Information Systems Engineering. 62. 10.1007/s12599-020-00661-0.

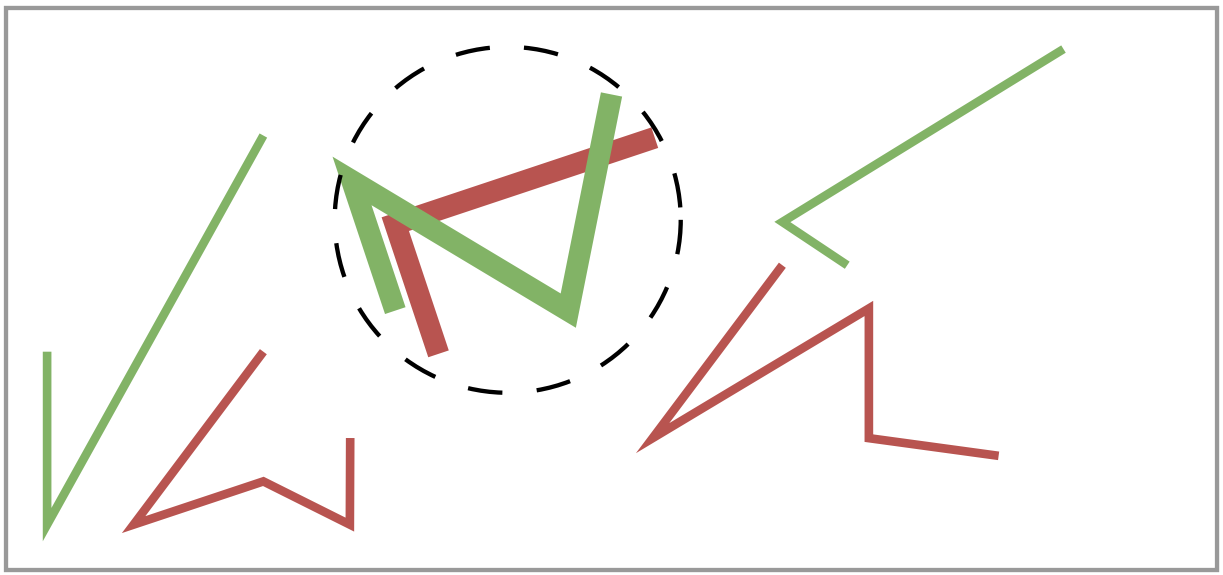


CAS–Atlantic

UNB

# Spatial Query Processing

**Spatial Join**: finds object pairs from two tables, which satisfies a spatial predicate like *ST_INTERSECTS*

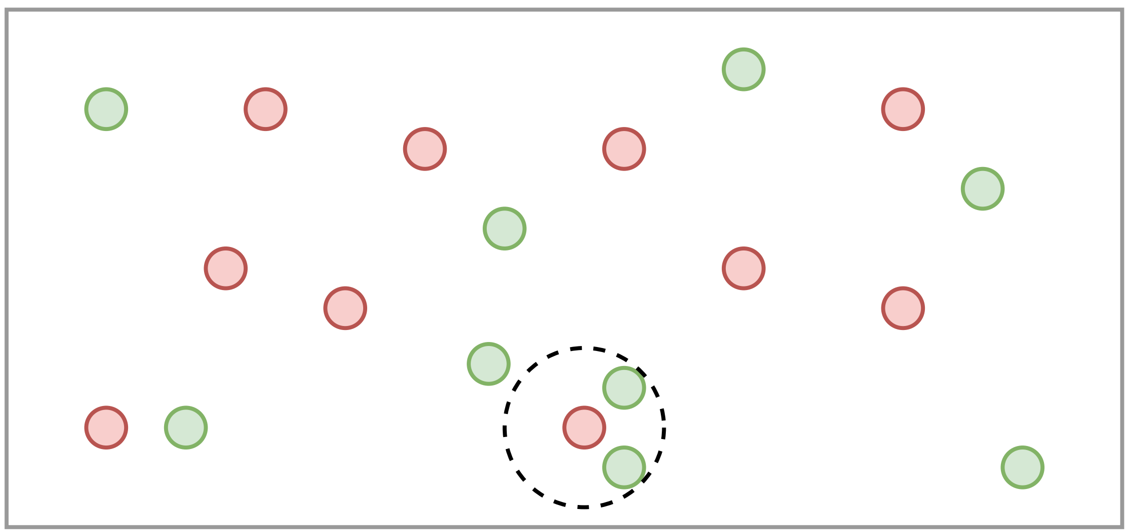




CAS–Atlantic

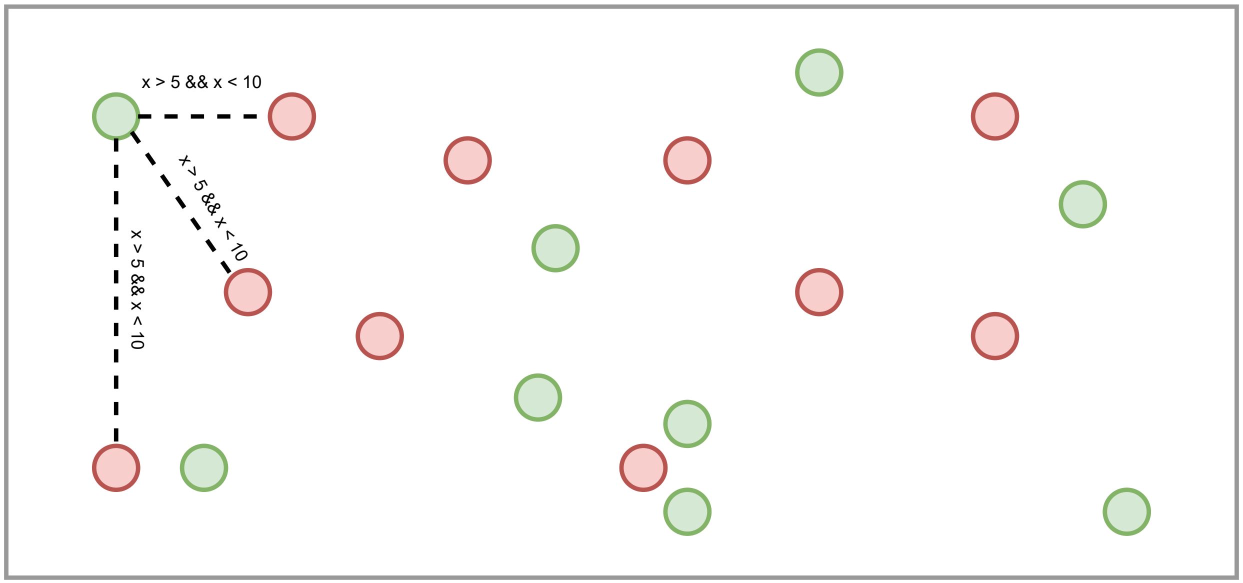
# Spatial Query Processing

- **Spatial Range Join**: finds object pair where the objects are within a defined radius of the other object (query object) from the other table. *ST_DWITHIN*



# Spatial Query Processing

**Spatial Distance Join**: finds object pairs that satisfy a particular distance unit. *ST_DISTANCE*



UNB

# Outline

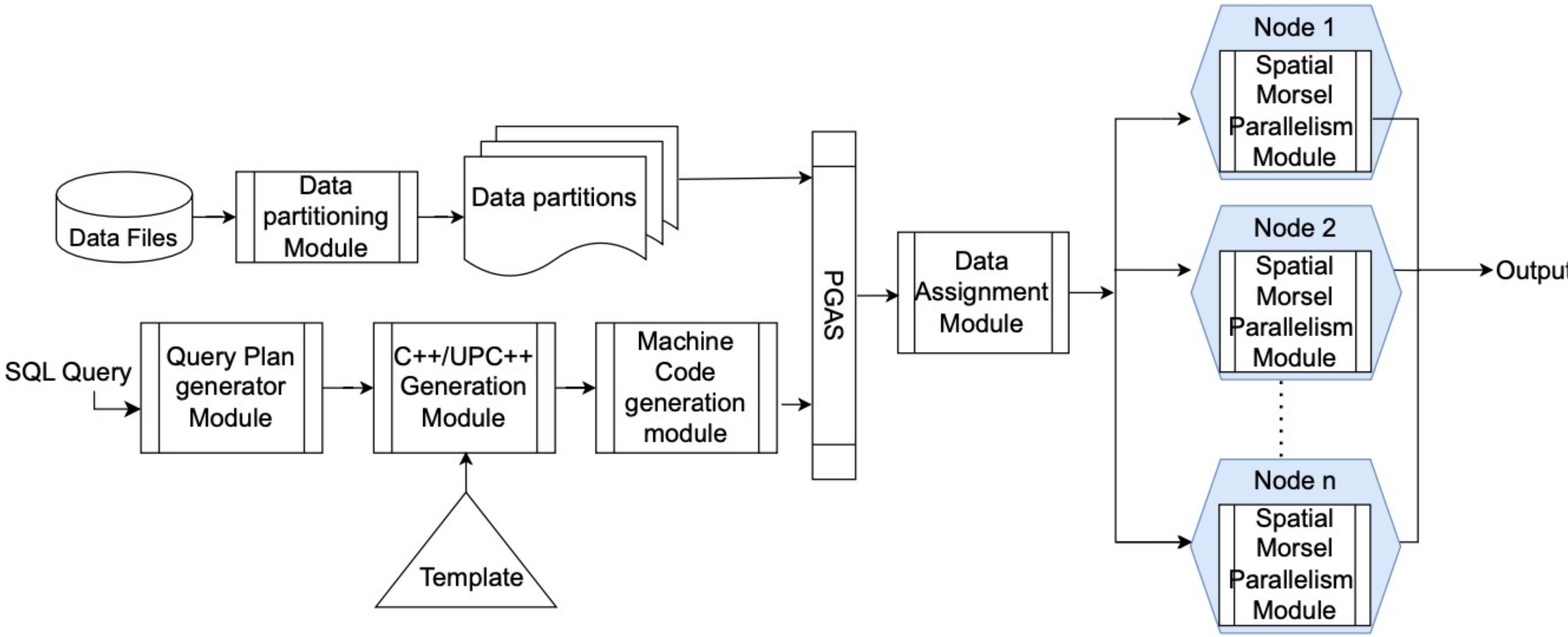
- Motivation
- Background
- Our system
- Evaluation
- Conclusions



CAS–Atlantic

UNB

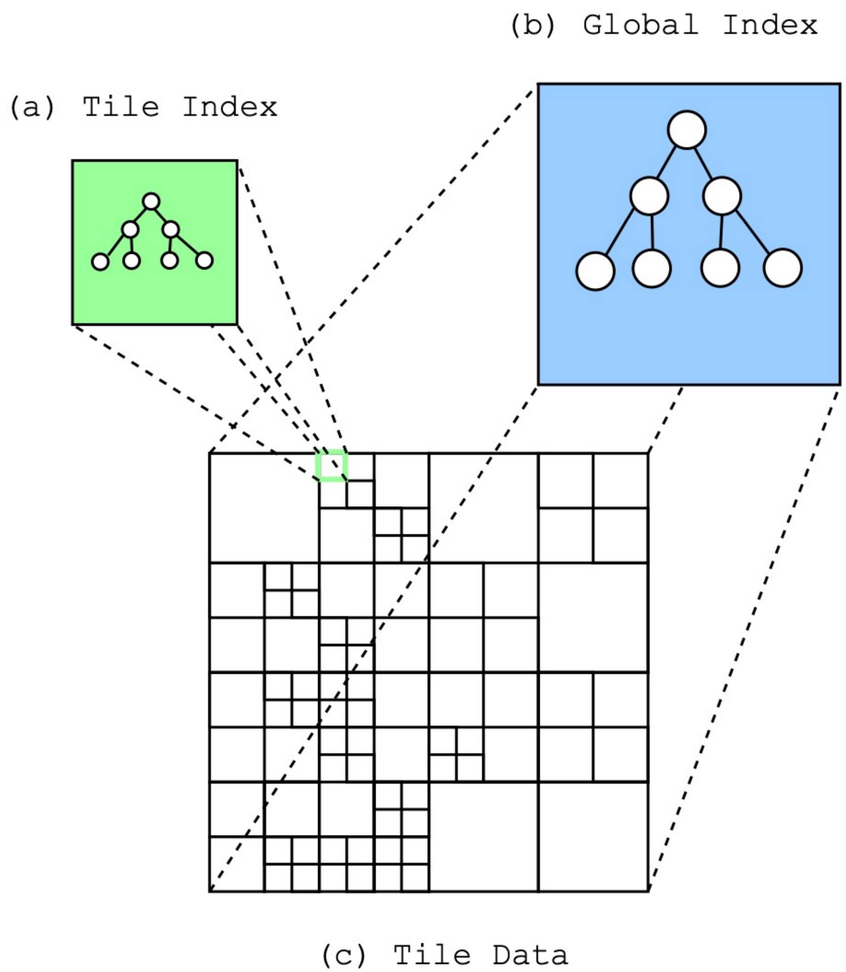
# Our System - CasaDB






CAS–Atlantic

# Index Organization



# Morsel-driven Parallelism (MDP)

- Parallelism by running the operator pipelines in parallel on separate threads.
- MDP divides data into small chunks called “morsels”.
- MDP’s dispatcher spawns a fixed number of workers and each of these workers is assigned a morsel.
- MDP’s dispatcher provides dynamic task scheduling, load balancing and parallelism.

UNB

# Morsel Driven Parallelism (MDP) for Spatial Data

- MDP works well for non-spatial workload.

- For a tile-based Spatial workload
  - How to define morsel?
  - How to handle processing skew within morsels?



CAS–Atlantic

UNB

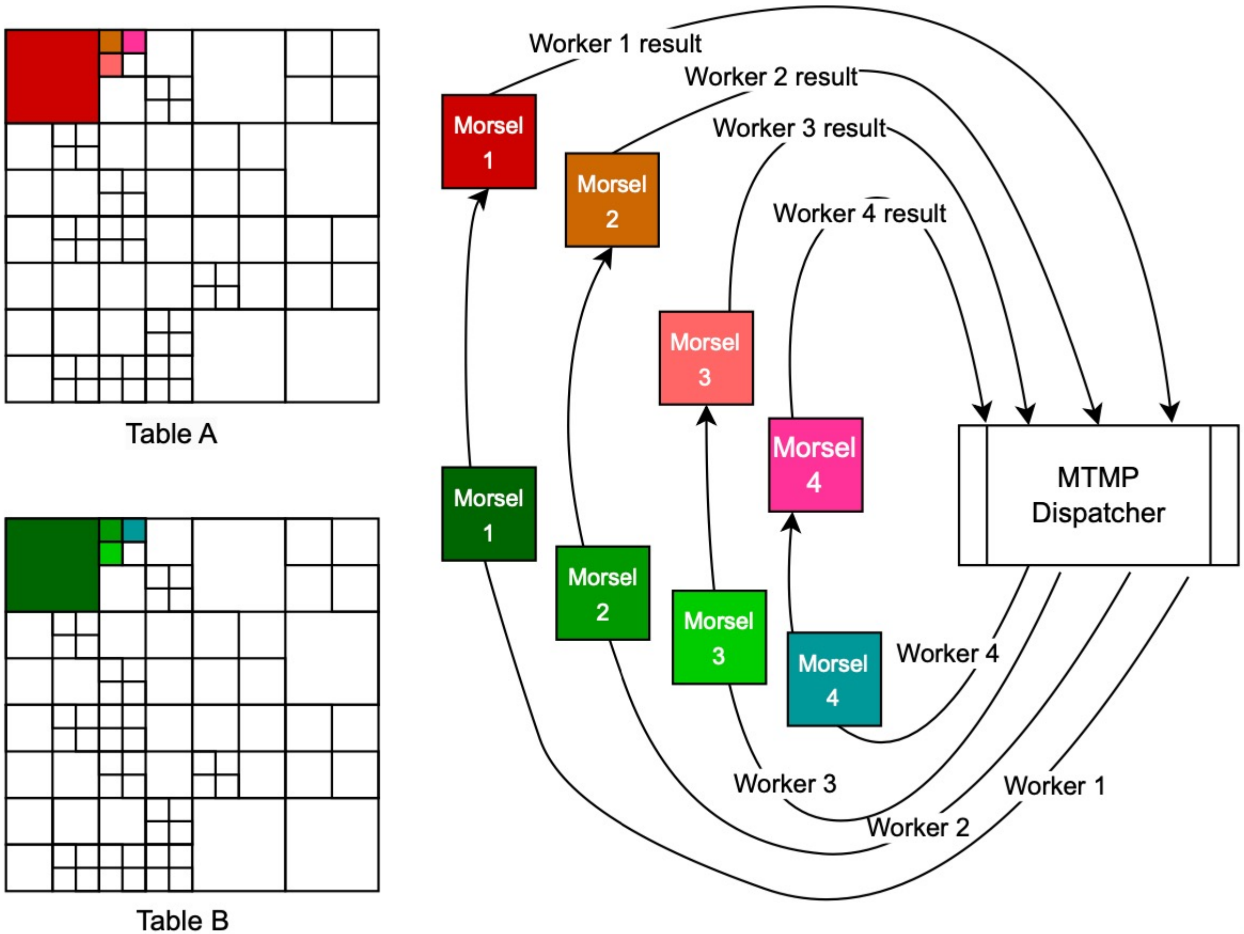
# Morsel Driven Parallelism (MDP) for Spatial Data

- Monolithic Tile-based Morsel Driven Parallelism (MTMP)
- Granular Tile-based Morsel Driven Parallelism (GTMP)



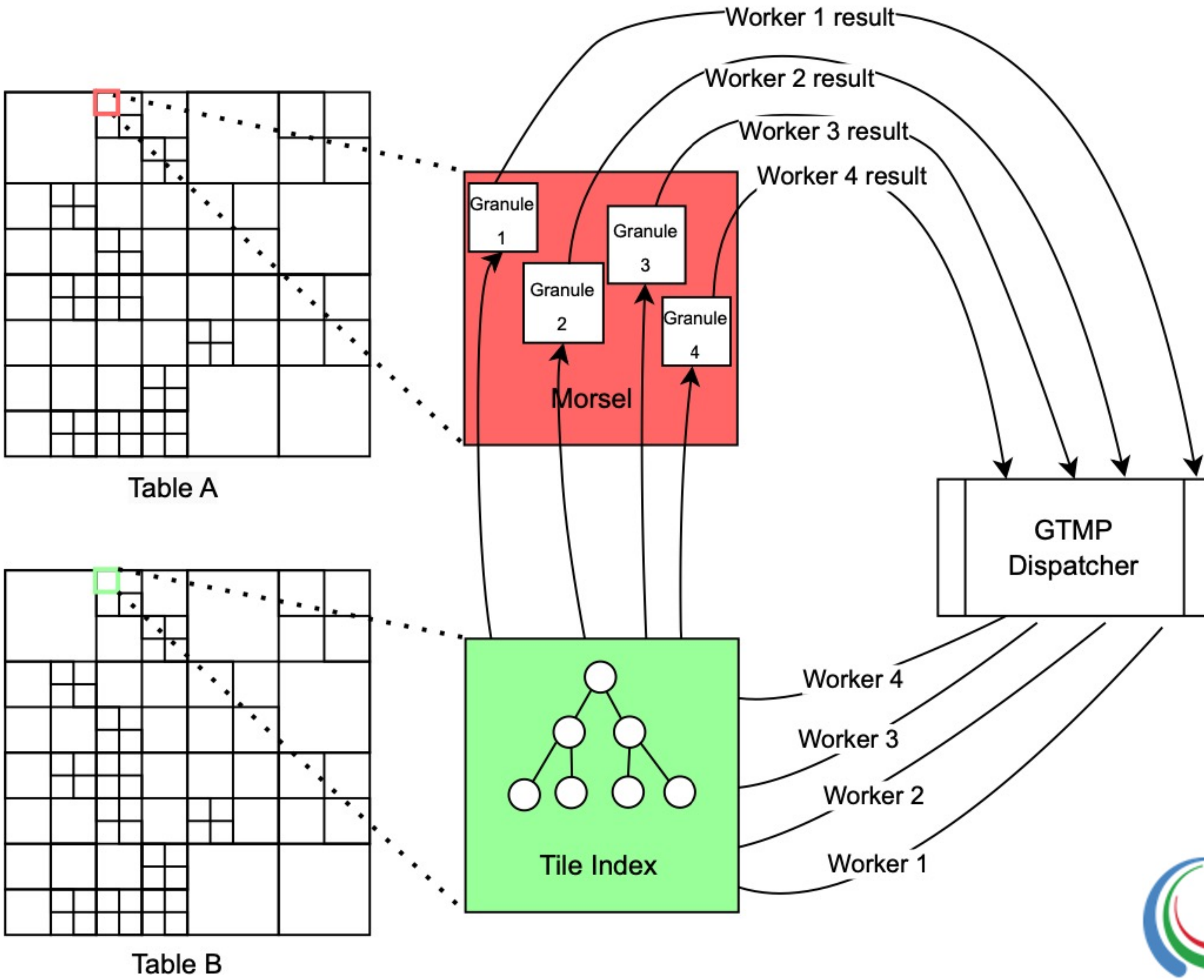
CAS–Atlantic

# Monolithic Tile-based Morsel Driven Parallelism (MTMP)



UNB

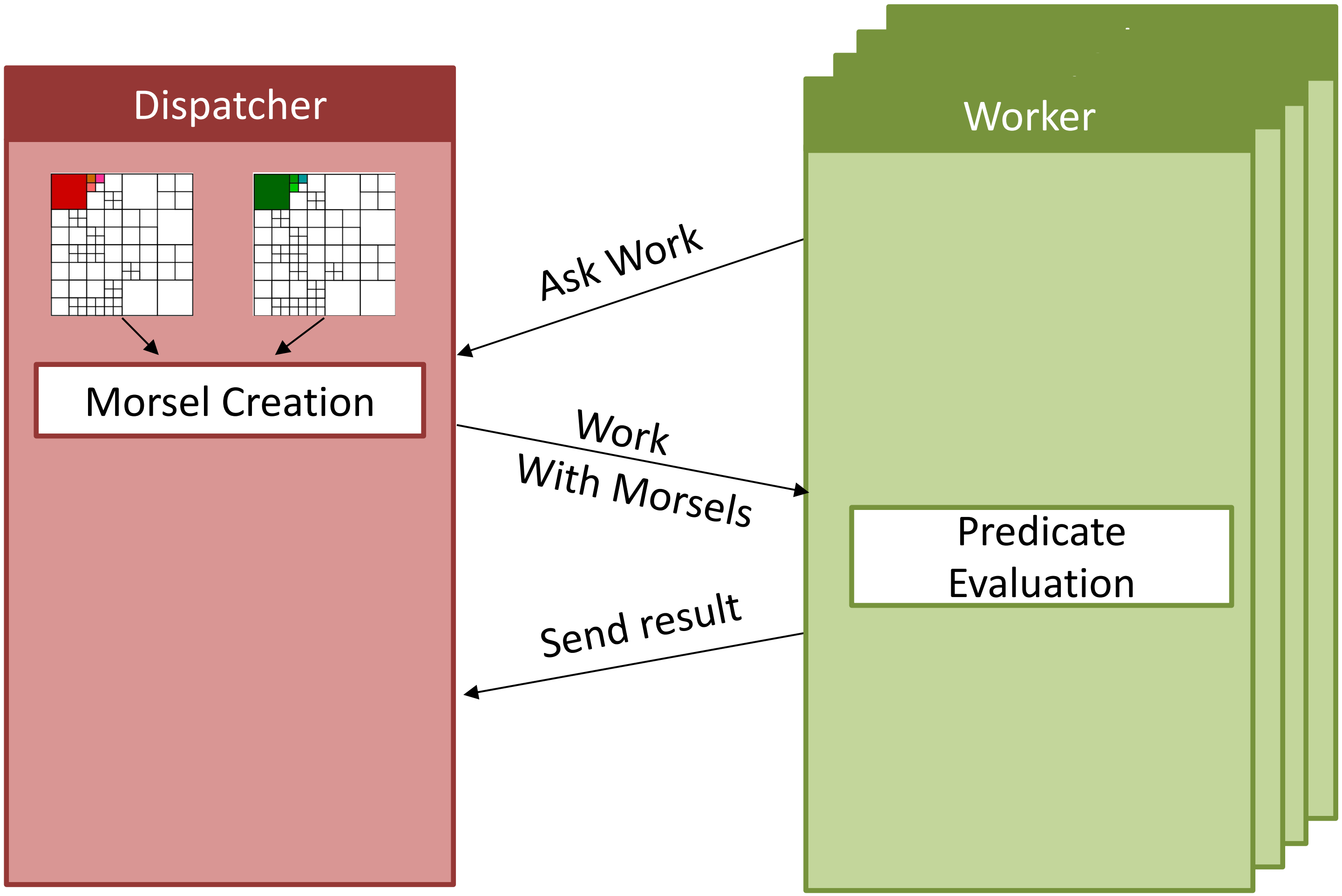
# Granular Tile-based Morsel Driven Parallelism (GTMP)




CAS–Atlantic

UNB

# Spatial Join Processing - MTMP

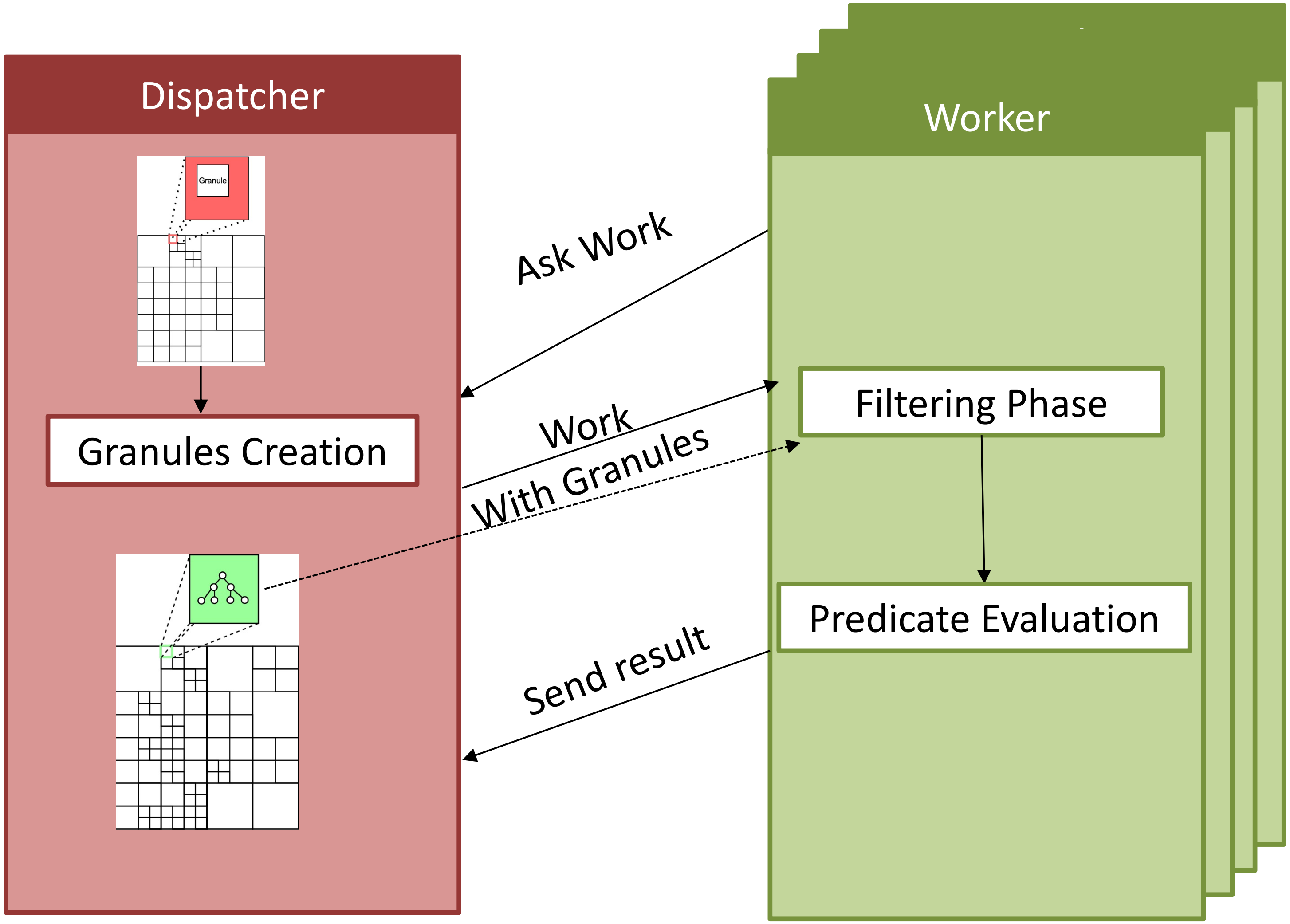





30
CAS–Atlantic

UNB

# Spatial Join Processing - GTMP






CAS–Atlantic

UNB

# Outline

- Motivation
- Background
- Out System
- Evaluation
- Conclusions



CAS–Atlantic

UNB

# Experimental Setup

- Cluster of 6 machines, Dual-Core AMD Opteron Processor 2222 clocked at 3 GHz and 16 GB main memory with Ubuntu 16.06

- Apache Sedona v1.4.0, Citus v12.1



CAS–Atlantic

UNB

# Experimental Setup - Dataset

**TIGER Dataset**

- U.S. Census Bureau's geographic spatial data
- Contains spatial information about various geographic features in the United State, such as roads, rivers, railroads, boundaries, landmarks, and more

| Table Name | Geometric Shape | No. of tuples |
|---|---|---|
| Arealm | Polygon | 6,708 |
| Areawater | Polygon | 40,799 |
| Pointlm | Point | 49,837 |
| Edges | Line | 4,251,911 |



CAS–Atlantic

UNB

# Experimental Setup - Queries

**TIGER Queries**

| Query Name | Query |
|---|---|
| TIGER_Q1 | select * from arealm join areawater on ST_TOUCHES(arealm.geom, areawater.geom) |
| TIGER_Q2 | select * from edges join arealm on ST_CROSSES(edges.geom, arealm.geom) |
| TIGER_Q3 | select * from edges join edges as edges2 on ST_CROSSES(edges.geom, edges2.geom) |
| TIGER_Q4 | select * from edges join areawater on ST_CROSSES(edges.geom, areawater.geom) |
| TIGER_Q5 | select * from areawater join areawater as areawater2 on ST_OVERLAPS(areawater.geom, areawater2.geom) |
| TIGER_Q6 | select * from pointlm join areawater on ST_WITHIN(pointlm.geom, areawater.geom) |
| TIGER_Q7 | select * from pointlm join areawater on ST_DWITHIN(pointlm.geom, areawater.geom, 1) |
| TIGER_Q8 | select * from pointlm join areawater on ST_DISTANCE(pointlm.geom, areawater.geom) >= 1 |



CAS–Atlantic

UNB

# Experimental Setup - Dataset

**OSM UK Dataset**

- OpenStreetMap(OSM) is a free map data provided by the website openstreetmap.org.
- Contains spatial information about various geographic features in the United Kingdome, such as roads, buildings, waterways, water bodies, etc.

| Table Name | Geometric Shape | No. of tuples |
|---|---|---|
| poi_point_uk | Point | 907,914 |
| bld_poly_uk | Polygon | 12,982,472 |
| lwn_poly_uk | Polygon | 2,742,757 |
| rds_line_uk | Line | 593,706 |



CAS–Atlantic

UNB

# Experimental Setup - Queries

**OSM**

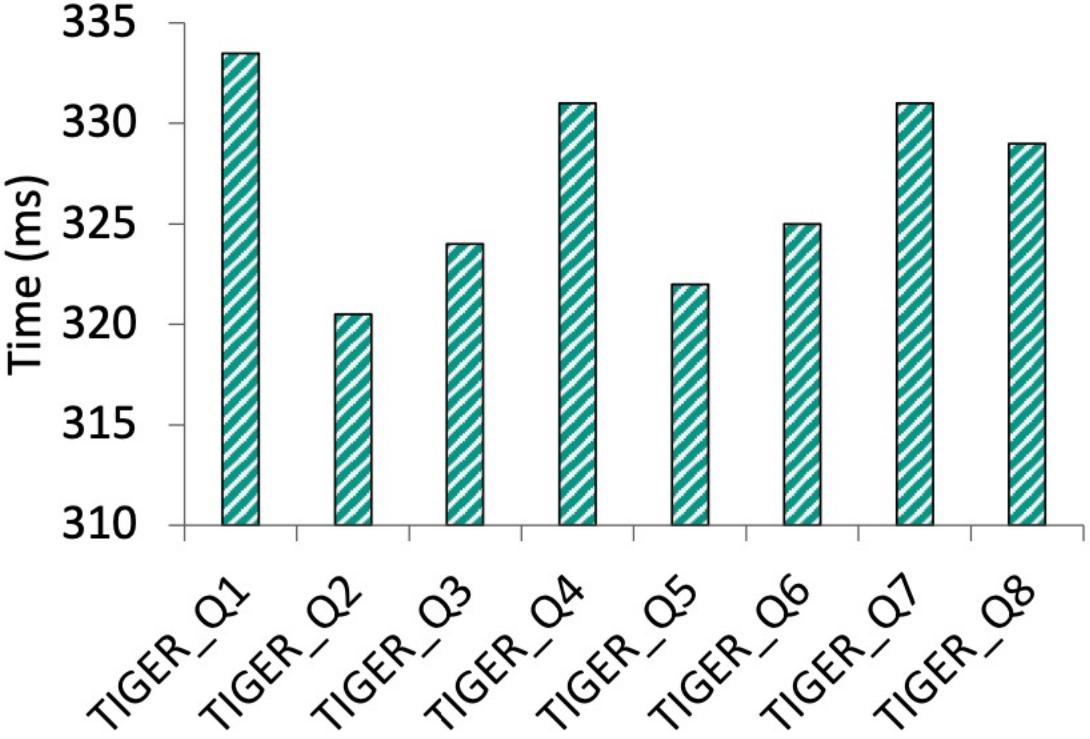
**Queries**

| **Query Name** | **Query** |
|---|---|
| OSM_Q1 | select * from rds_lin_uk join bld_poly_uk on ST_TOUCHES(rds_lin_uk.geom, bld_poly_uk.geom) |
| OSM_Q2 | select * from rds_lin_uk join lwn_poly_uk on ST_CROSSES(rds_lin_uk.geom, lwn_poly_uk.geom) |
| OSM_Q3 | select * from poi_point_uk join lwn_poly_uk on ST_WITHIN(poi_point_uk.geom, lwn_poly_uk.geom) |
| OSM_Q4 | select * from bld_poly_uk join lwn_poly_uk on ST_OVERLAPS(bld_poly_uk.geom, lwn_poly_uk.geom) |



CAS–Atlantic

# Code Generation Time

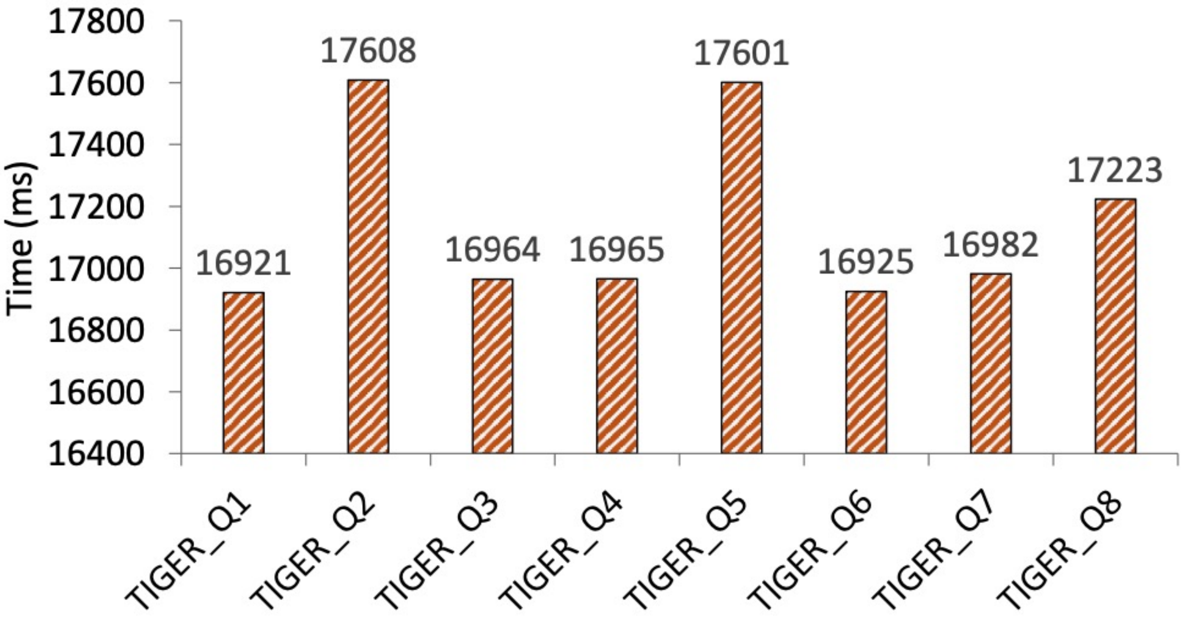





CAS–Atlantic

UNB

# Code Compilation Time






CAS–Atlantic

UNB

# GTMP vs MTMP

**Performance analysis of GTMP vs MTMP – Conclusion**

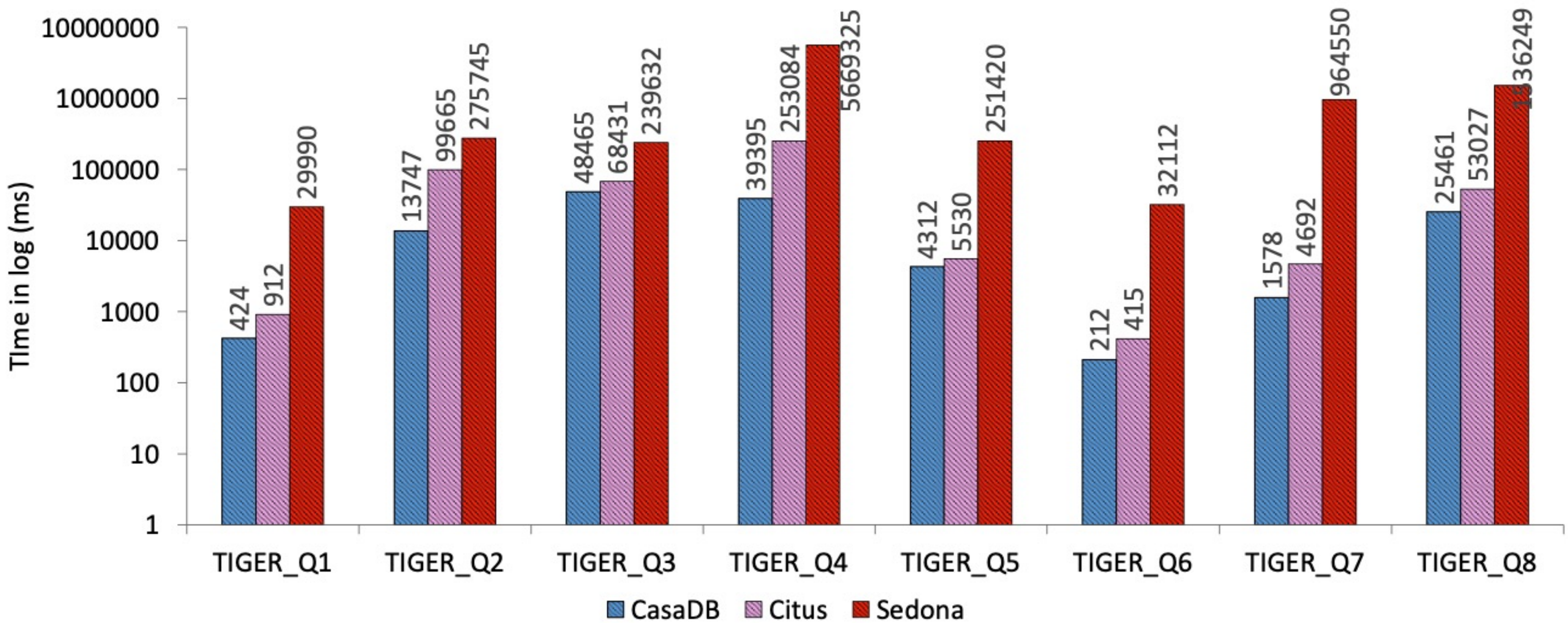
- For most of the queries GTMP and MTMP performs similar
- In case of skew GTMP performs better than MTMP
- Spatial Range Join and Spatial Distance Join performs better on GTMP than MTMP



CAS–Atlantic

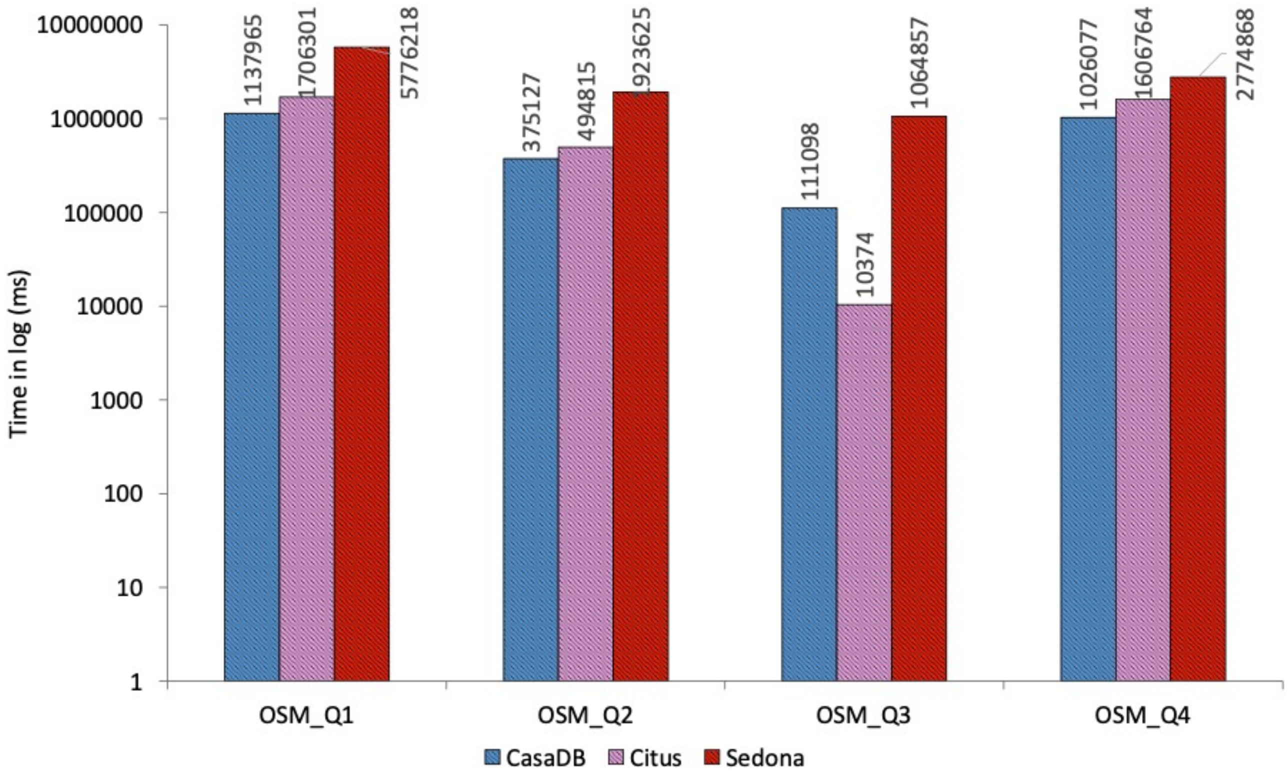
UNB

# CasaDB vs Citus vs Apache Sedona TIGER Dataset




CAS–Atlantic

# CasaDB vs Citus vs Apache Sedona OSM Dataset






CAS–Atlantic

UNB

# CasaDB vs Citus vs Apache Sedona

- CasaDB is **4x faster** on average, than **Citus** and **308x faster** on an average, than **Apache Sedona** on **TIGER** dataset.
- CasaDB is at least **7.3x faste**r on an average, than **Sedona** and atmost **1.5x faster** than **Citus** on **OSM** dataset.



CAS–Atlantic

# Outline

- Motivation
- Background
- Out System
- Evaluation
- Conclusions

# Conclusion

- Presents a compilation-based distributed spatial query processing engine for CasaDB.
- Proposed two new morsel parallelism-based algorithms, Monolithic Tile based Morsel Parallelism (MTMP) and Granular Tile based Morsel Parallelism (GTMP).
- Presented two Index organization techniques, Global Index and Tile-based Index and how they can be used with different kinds of spatial joins.

UNB

# Conclusion

- **CasaDB** is **4x faster** than Citus and **308x** faster than Apache Sedona on TIGER dataset, and on OSM dataset CasaDB is **almost 1.5x faster** than Citus and **7.3x faster** than Apache Sedona.

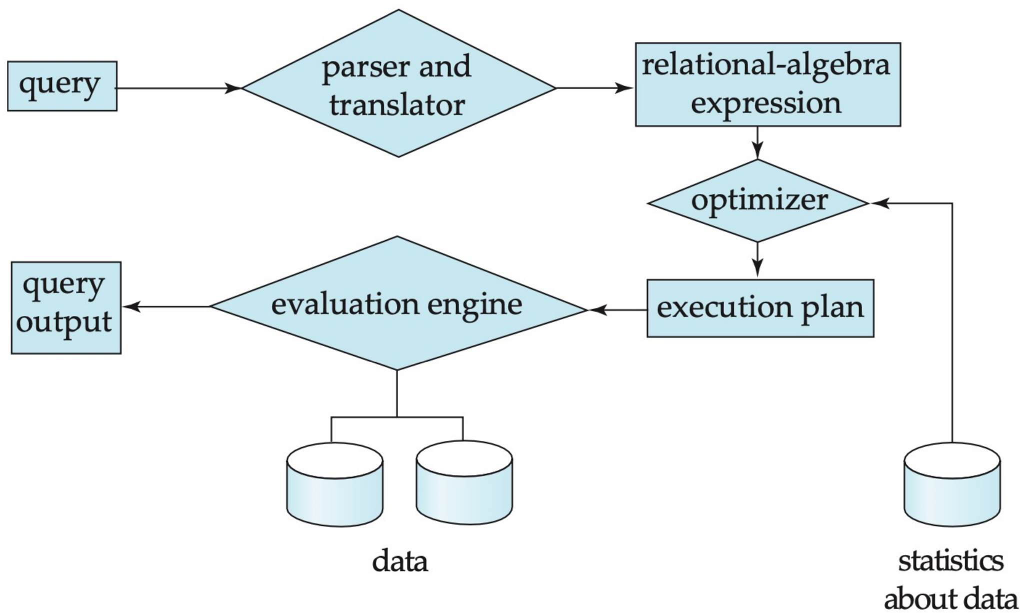


CAS–Atlantic

Thanks!

# Query Processing

- What happens with SQL query?
  - Goal: translate SQL query to an executable plan and run it
  - Steps:
    - Parsing and validation
    - Optimization
    - Execution




* Image from Abraham Silberschatz, Henry F. Korth, and S. Sudarshan, Database system concepts, 6 ed., McGraw-Hill, New York, 2010.

# Query Processing - Sample query

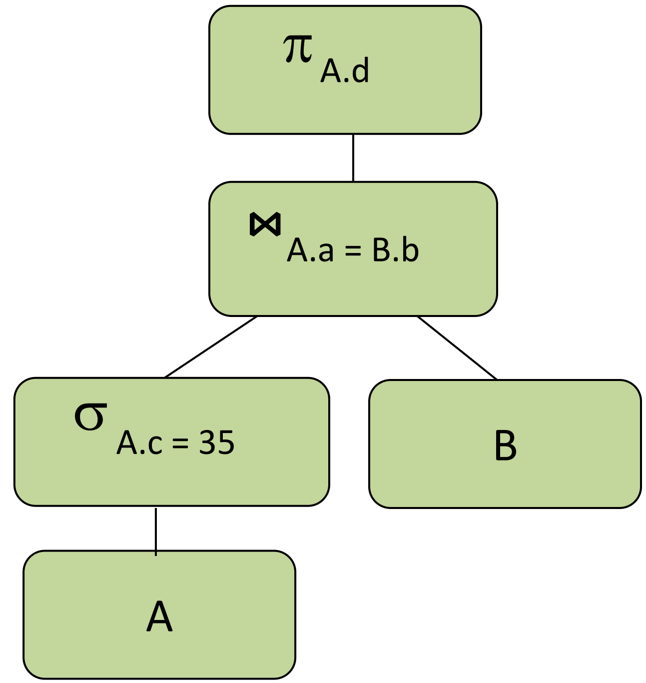
**TABLES:**

A(a,c,d)

B(b,e,f)

**QUERY:**

SELECT A.d
FROM A, B
WHERE A.a = B.b
AND A.c = 35




Logical plan

UNB

# Sample query - Push-Based

**TABLES:**

A(a,c,d)

B(b,e,f)

**QUERY:**

SELECT A.d
FROM A, B
WHERE A.a = B.b
AND A.c = 35

```
for t in A:
    if (t.c == 35)
        materialize t in Hashtable HT


for t2 in B:
    t1 = probeHashtable(HT, t2)
    if (t1)
        emit(t1.d)
```

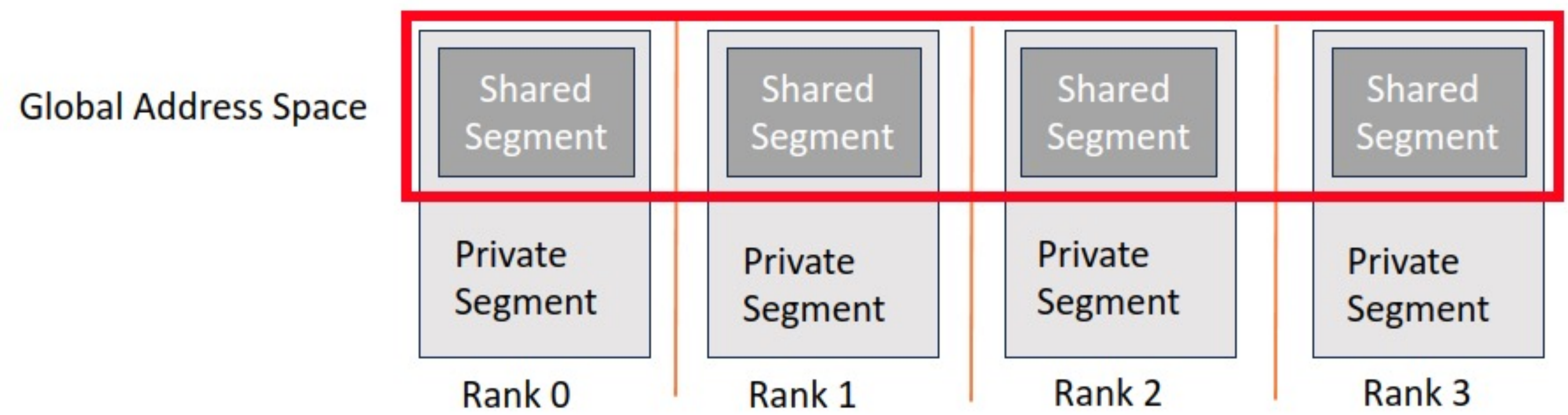
- Compiled code generated from SQL query

CAS–Atlantic

UNB

# PGAS

- Distributed-Shared Memory
  - Easy programmability and data referencing (global address space)
  - Good performance and data locality (partitioning)

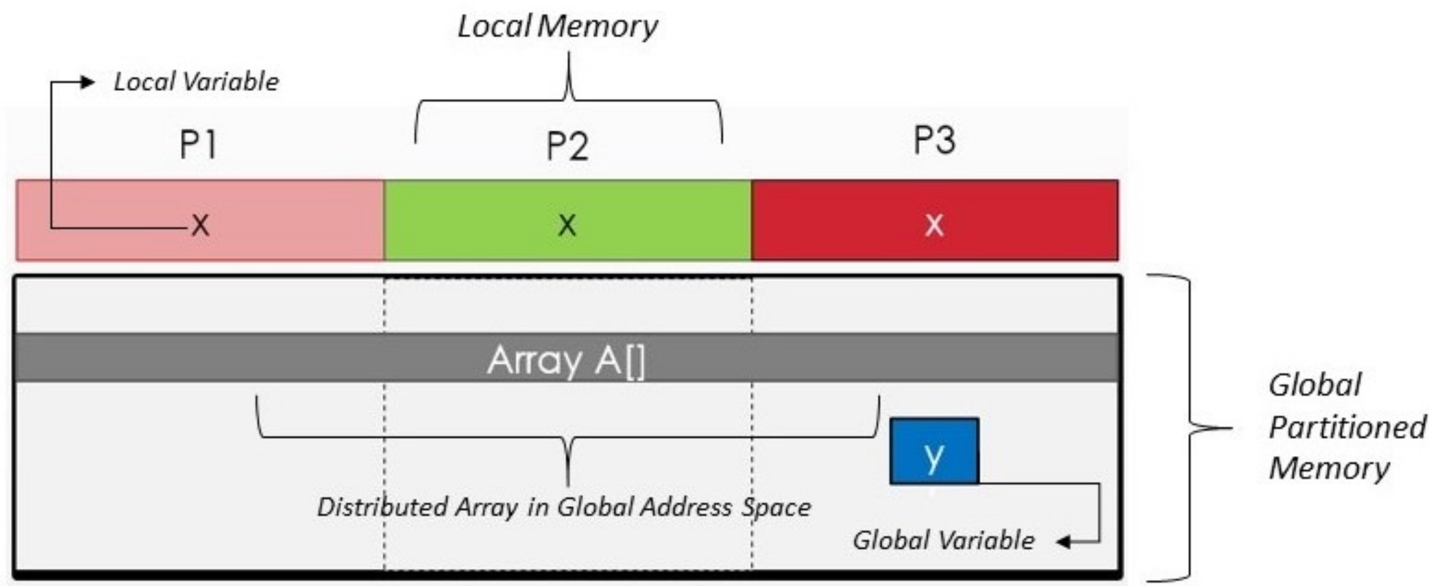





CAS–Atlantic

UNB

# PGAS






CAS–Atlantic

UNB

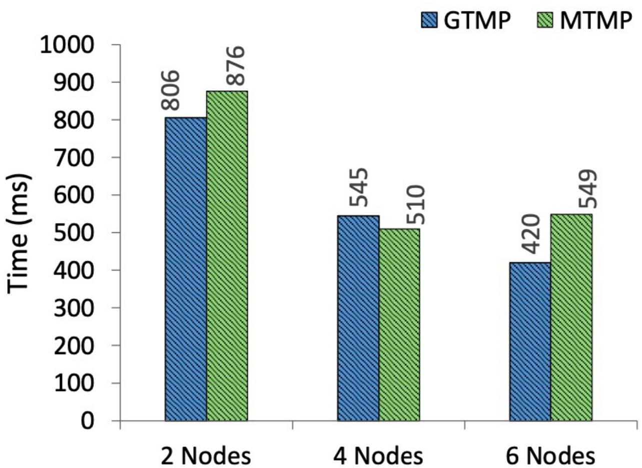
# UPC++

- Why UPC++-based Compiled Query Plans?
  - UPC++ is a C++ library that supports PGAS programming model
  - All accesses made to remote memory are explicit
  - All remote memory access operations are asynchronous
  - Enable developers to write code that performs well at scale
  - Minimal changes in the generated query plan code
    - Scalable to hundreds of nodes

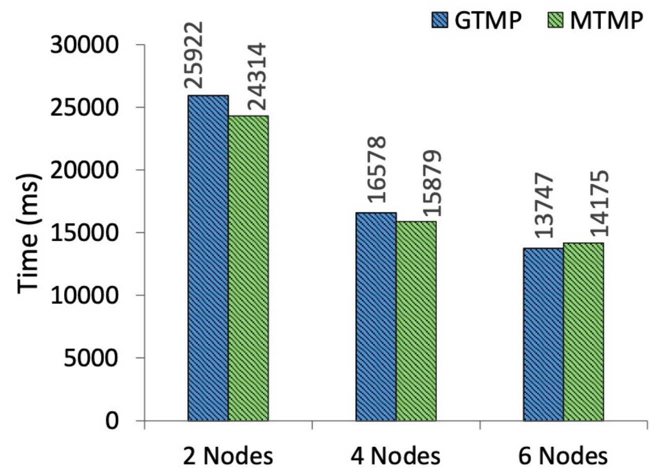


CAS–Atlantic

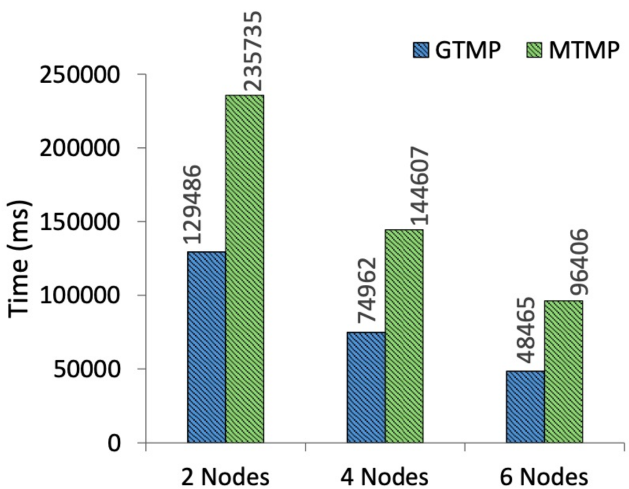
# GTMP vs MTMP



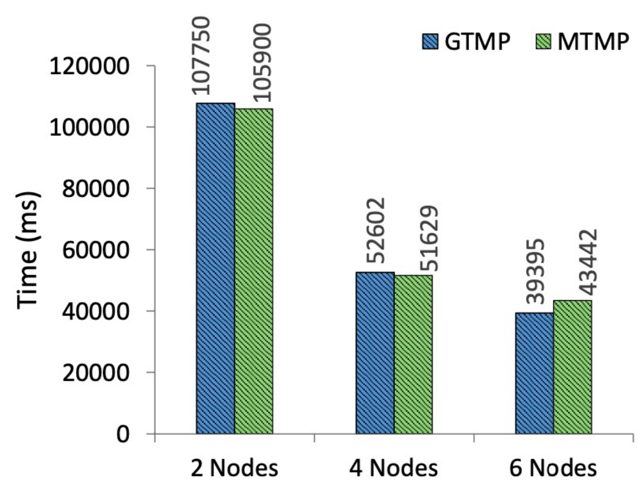

TIGER_Q1




TIGER_Q2



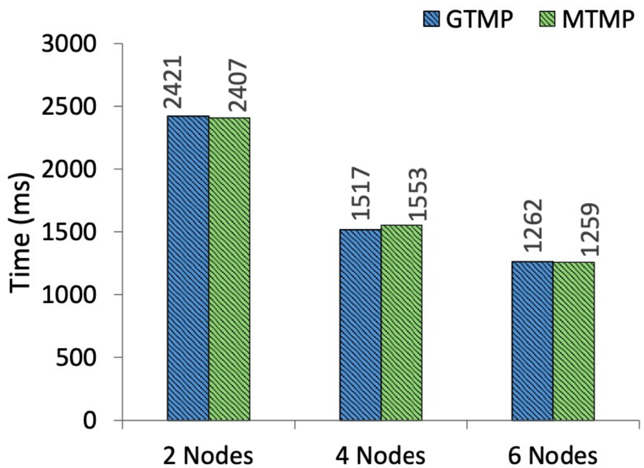

TIGER_Q3



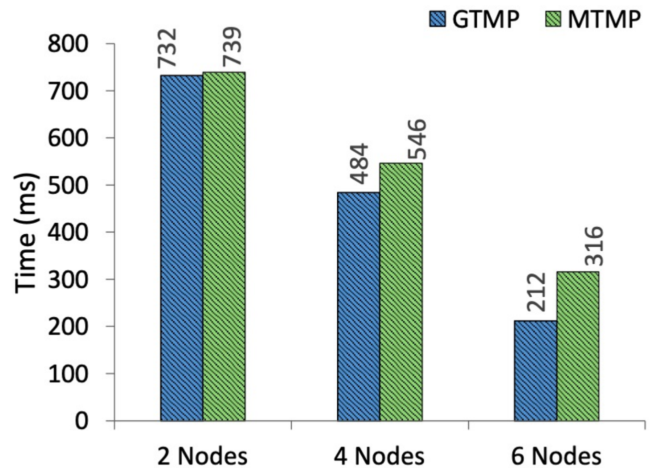

TIGER_Q4

UNB

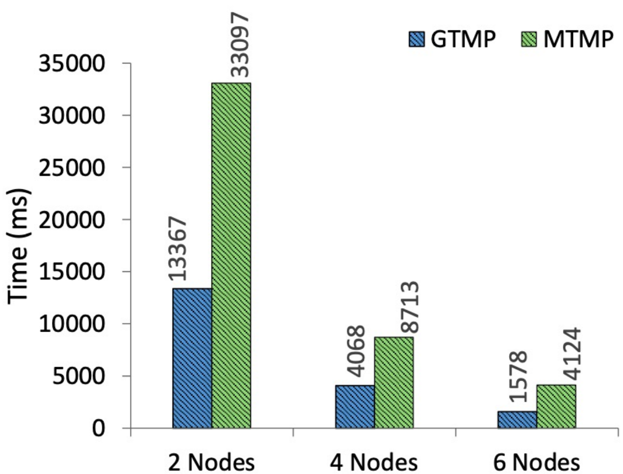
# GTMP vs MTMP



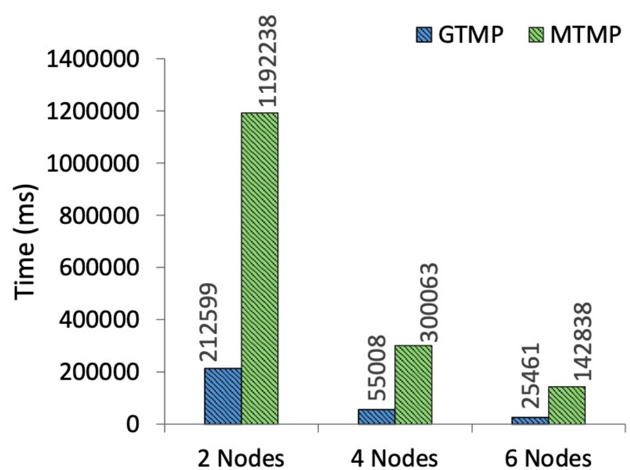

TIGER_Q5




TIGER_Q6




TIGER_Q7



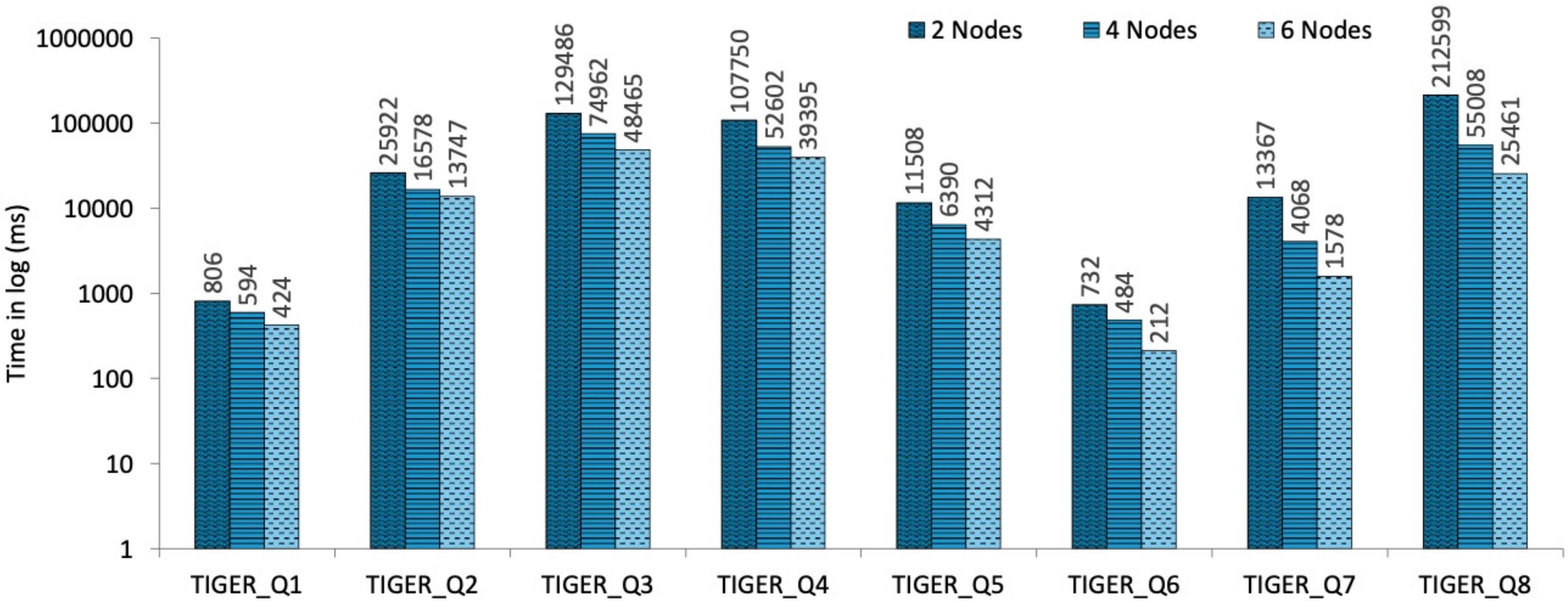

TIGER_Q8



CAS–Atlantic

UNB

# GTMP vs MTMP

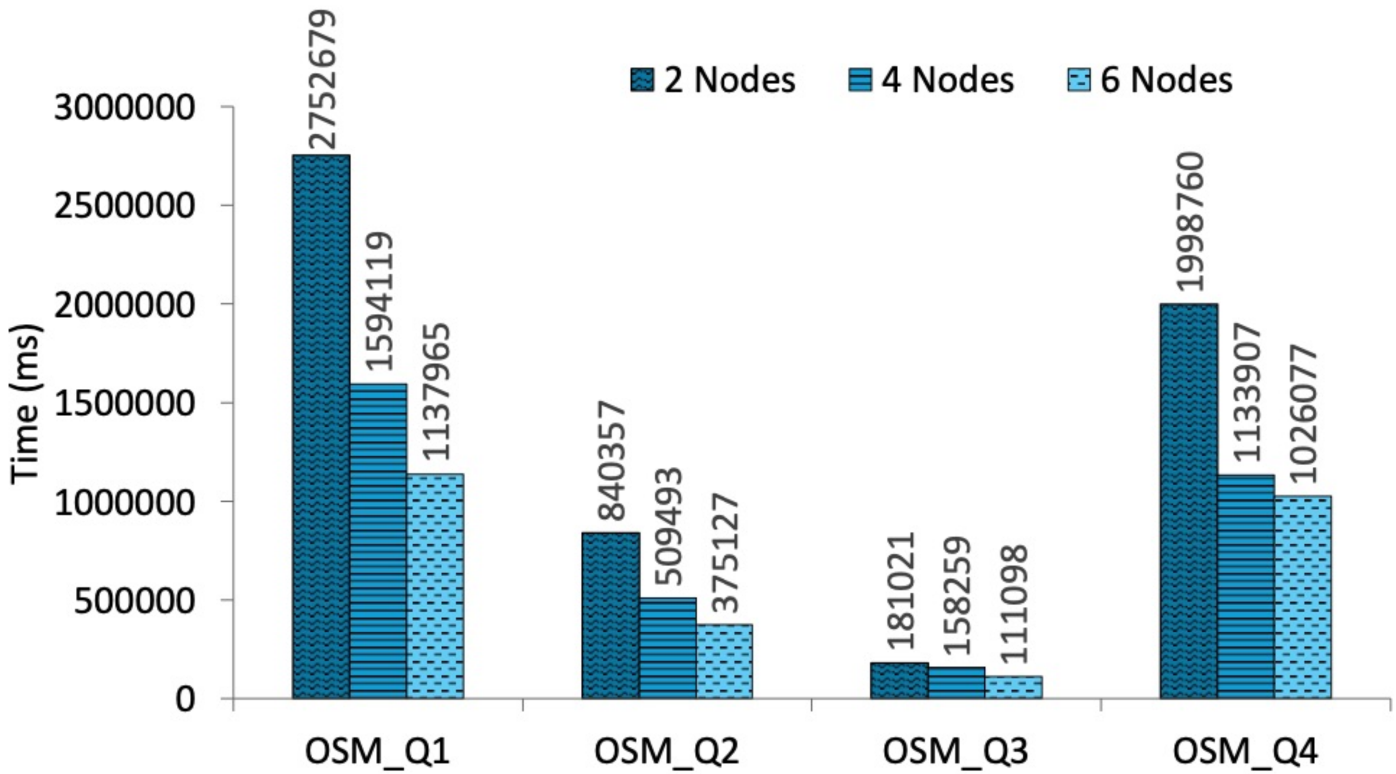





CAS–Atlantic

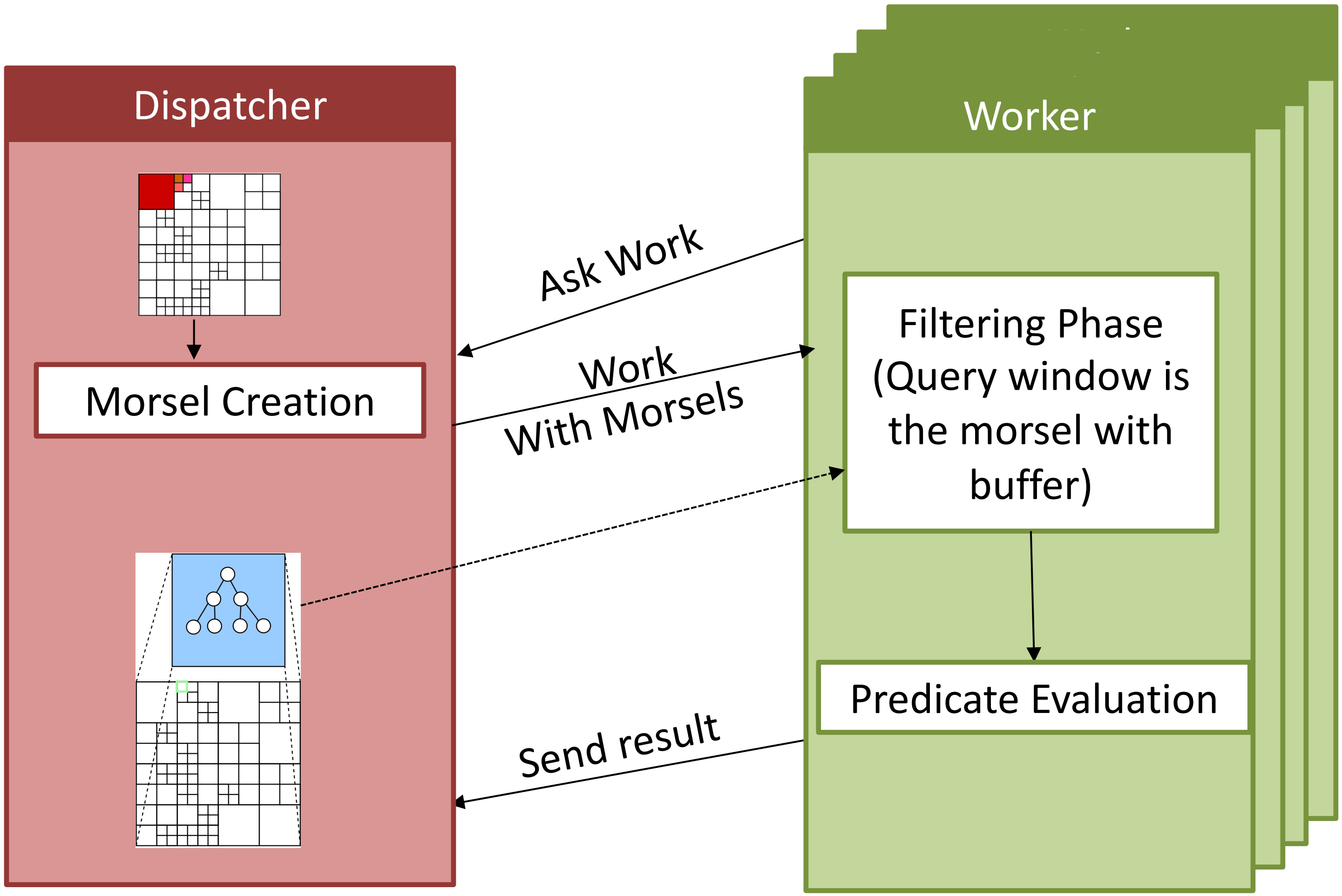
UNB

# GTMP vs MTMP

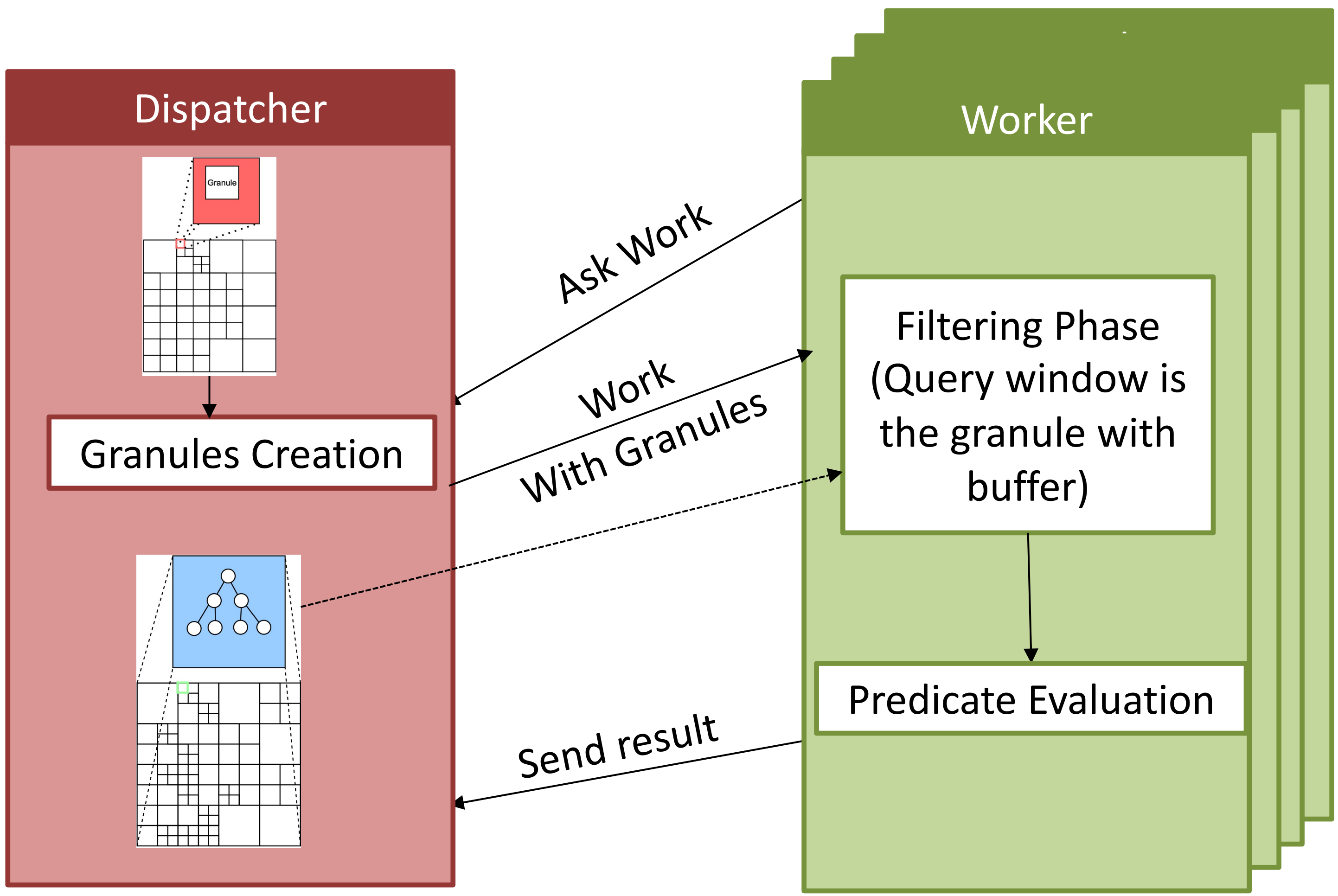



CAS–Atlantic

# Spatial Range Processing - MTMP



# Spatial Range Processing - GTMP



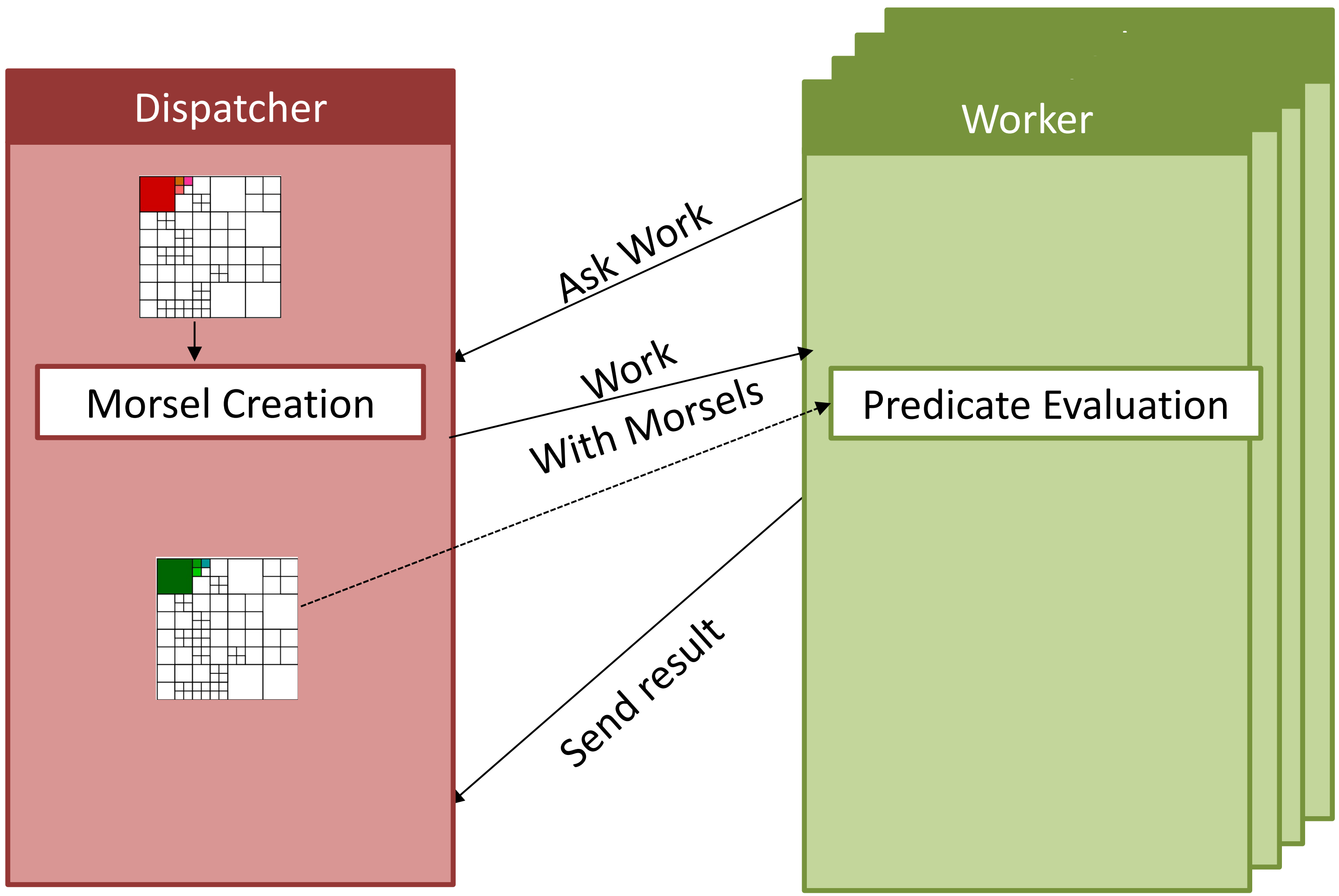



59

CAS–Atlantic

UNB

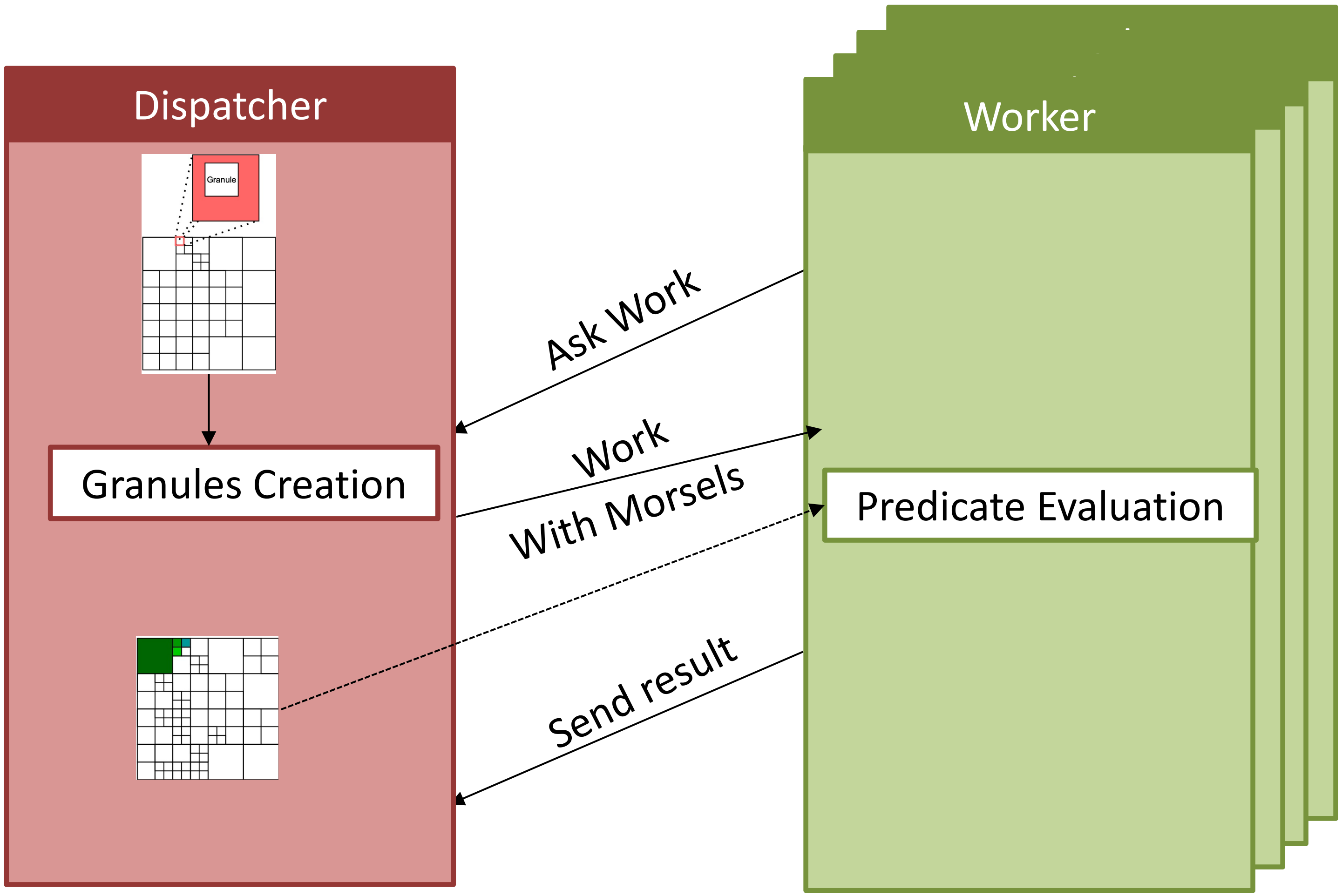
# Spatial Distance Processing - MTMP






CAS–Atlantic

# Spatial Distance Processing - GTMP



# Sample query - Push-Based