ASU Arizona State University

Building a Large-Scale Microscopic Road Network Traffic Simulator in Apache Spark

Zishan Fu **Jia Yu** Mohamed Sarwat

Arizona State University

Traffic Congestion

Traffic Figure: https://www.google.com/url?sa=i&source=images&cd=&ved=2ahUKEwjRuLKfwMbhAhWRu54KHaDiCI8QjRx6BAgBEAU&url=https%3A%2F%2Fwww.straitstimes.com%2Fasia%2Fse-asia%2Fthailand-has-worlds-most-congested-roads-survey&psig=AOvVaw3IU1ztCJ-gcsvD3y5f9rk1&ust=1555[illegible]

97

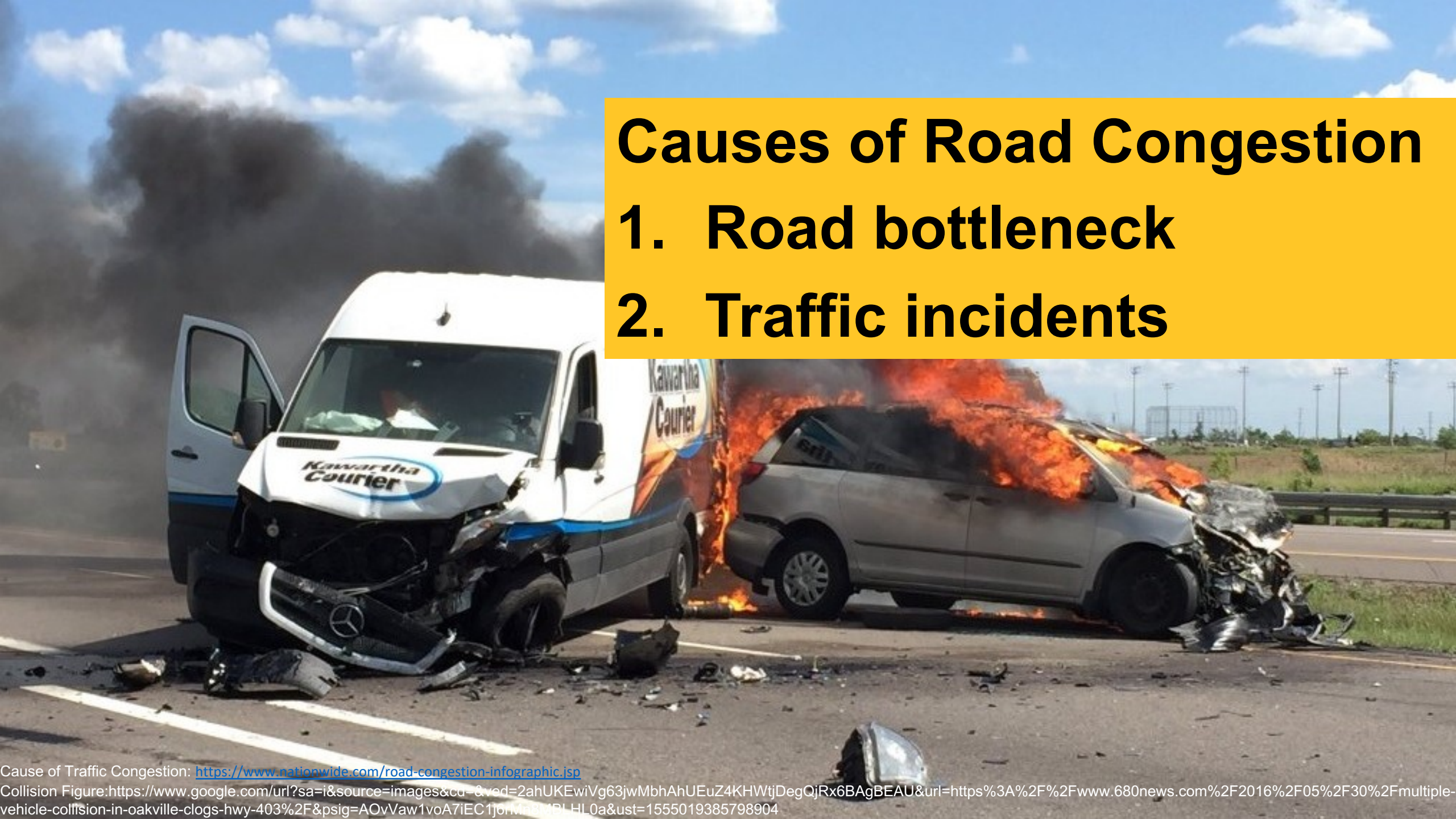
hours were lost in traffic congestion on average

87 **billion** total cost for traffic in 2018

United States traffic statistics 2018: https://www.statista.com/chart/12855/americas-most-congested-cities/

Causes of Road Congestion

1. Road bottleneck
2. Traffic incidents



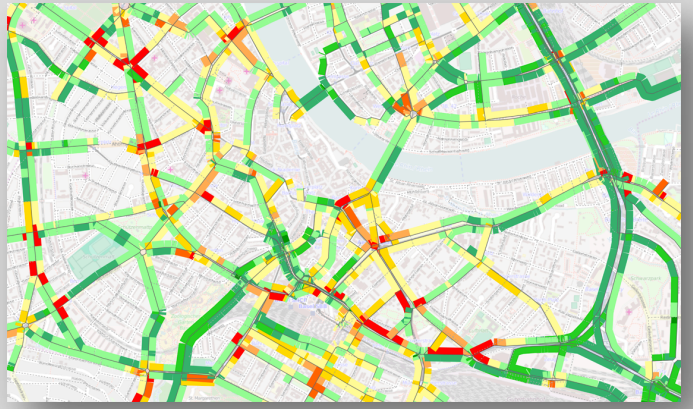
Cause of Traffic Congestion: https://www.nationwide.com/road-congestion-infographic.jsp

Collision Figure:https://www.google.com/url?sa=i&source=images&cd=&ved=2ahUKEwiVg63jwMbhAhUEuZ4KHWtjDegQjRx6BAgBEAU&url=https%3A%2F%2Fwww.680news.com%2F2016%2F05%2F30%2Fmultiple-vehicle-collision-in-oakville-clogs-hwy-403%2F&psig=AOvVaw1voA7iEC1jorMa8MBLHL0a&ust=1555019385798904

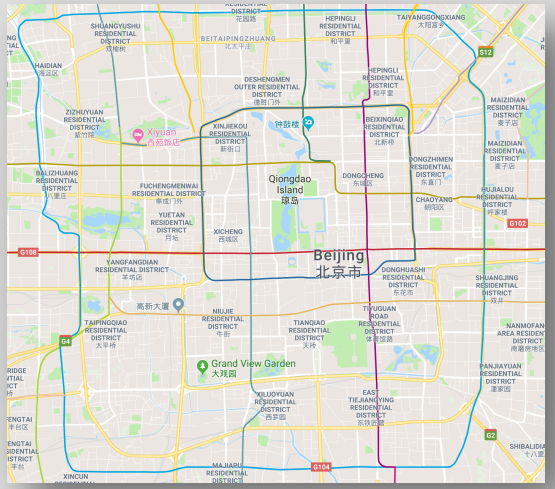
Traffic data can help

- Urban planning
- Traffic prediction
- Spatial-temporal databases

Traffic data



Road network



Where is the bottleneck?

What if we add a new road?

What if a road is blocked?

Left figure: https://vision-traffic.ptvgroup.com/nl/products/ptv-visum-data-analytics/use-cases/fusion-and-visualization-of-map-data-and-historical-traffic-data/
Right figure: Google maps

Collecting real traffic data is NOT EASY

- Problems
  - Small-scale: requires extra monitoring devices / privacy
  - Inaccurate: many trajectory data points deviate from roads
  - Road planning: the road is not built yet
- What we want
  - Large-scale traffic data: various driving behaviors and road situations
  - High-precision: follow on the road network
  - Test planned roads
- Synthetic but simulates realistic driving behaviors and road situations

GeoSparkSim

ASU Arizona State University

GeoSparkSim: Large-Scale Microscopic Road Network Traffic Simulator

- Scalable
  - big traffic trajectories
  - big road network
- Microscopic traffic simulation:
  - driving behaviors: speed, lane changing, car following
  - traffic signals
  - traffic rules
- Extensible: customizable driving behaviors
- Traffic visualization

GeoSparkSim

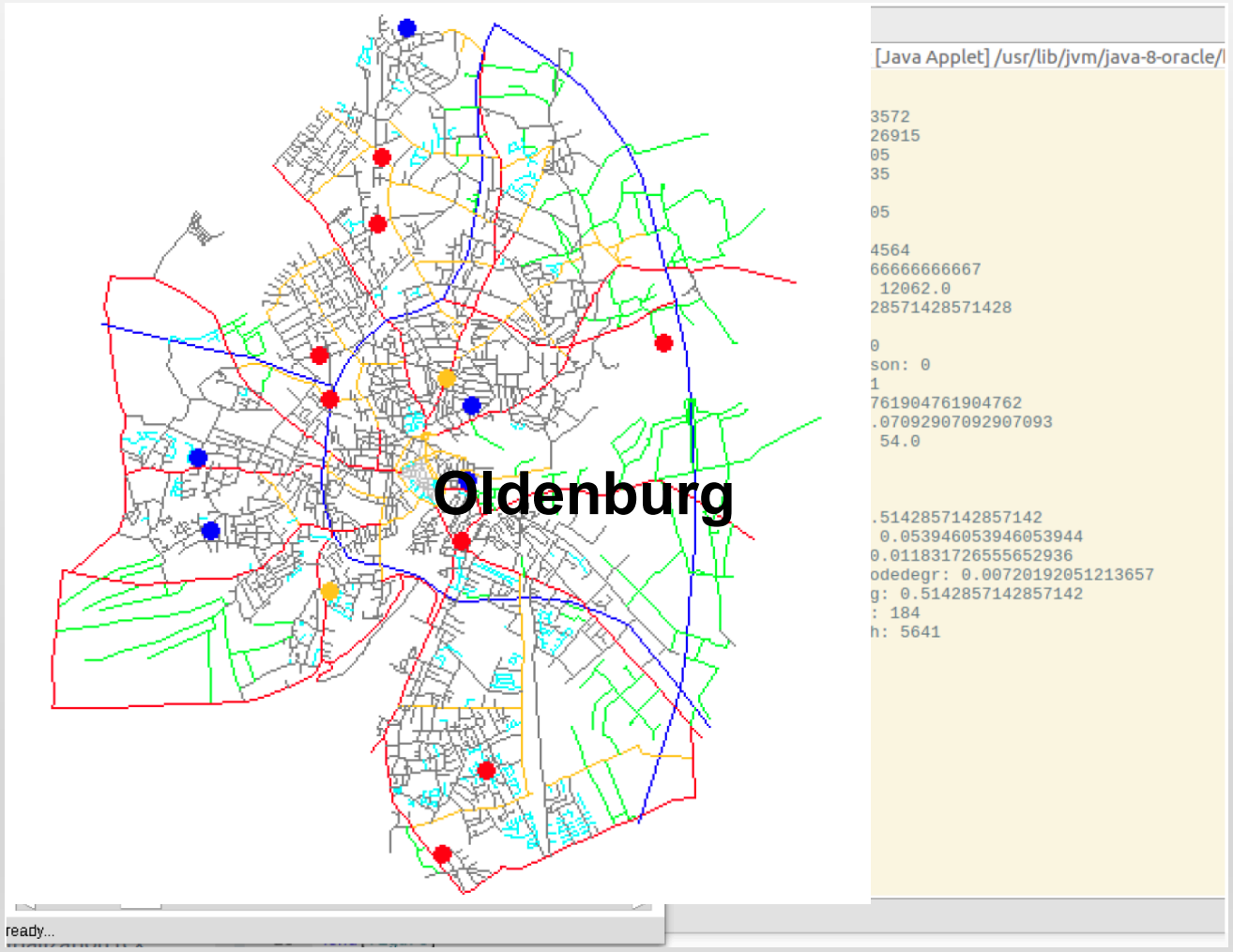
- Related Works
- Architecture
- Simulation
- Performance
- Demo

GeoSparkSim

- **Related Works**
- Architecture
- Simulation
- Performance
- Demo

Brinkhoff [1] and BerlinMOD [2]

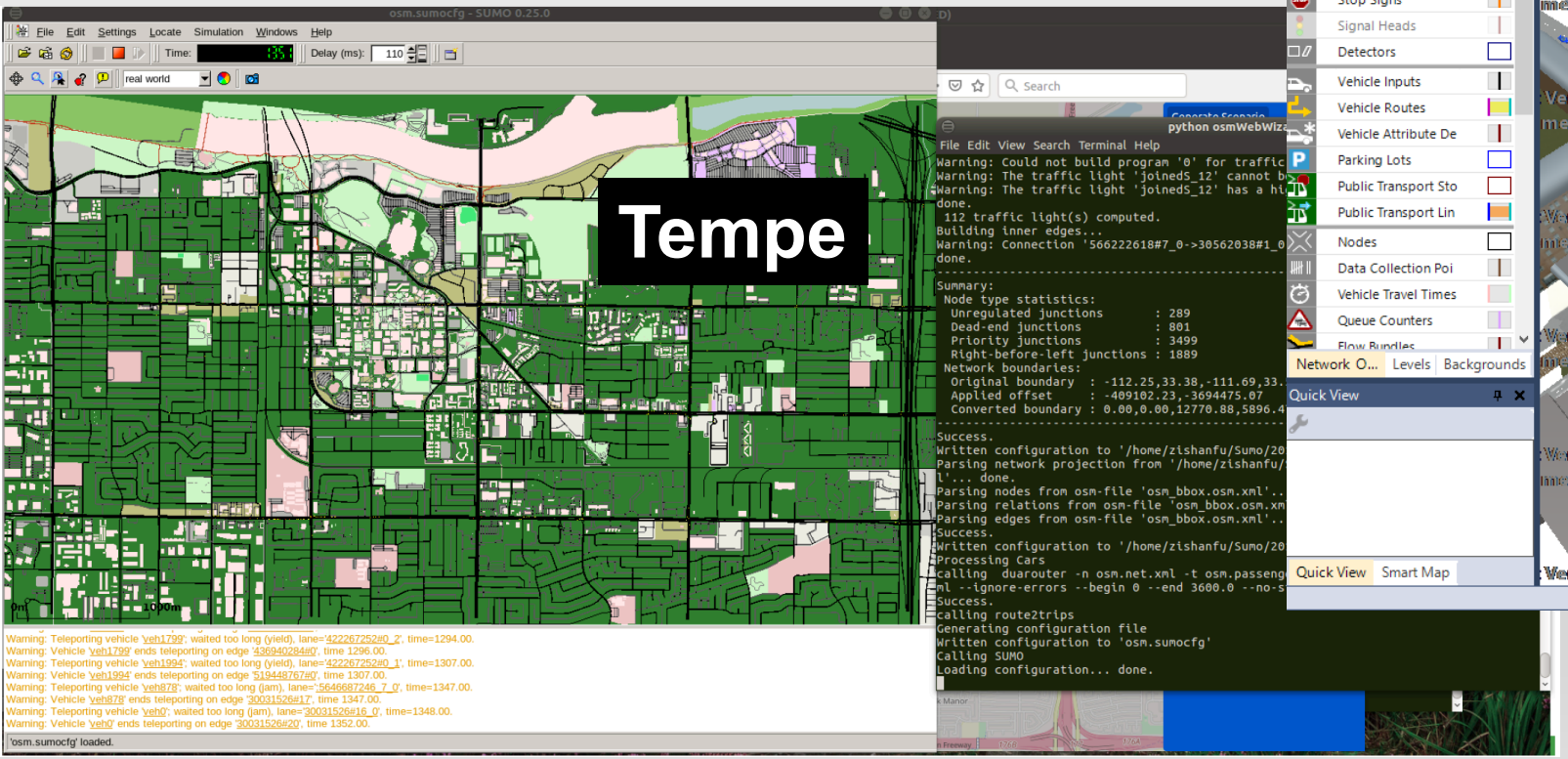
- Macroscopic Traffic Simulator
- Driving Behaviors



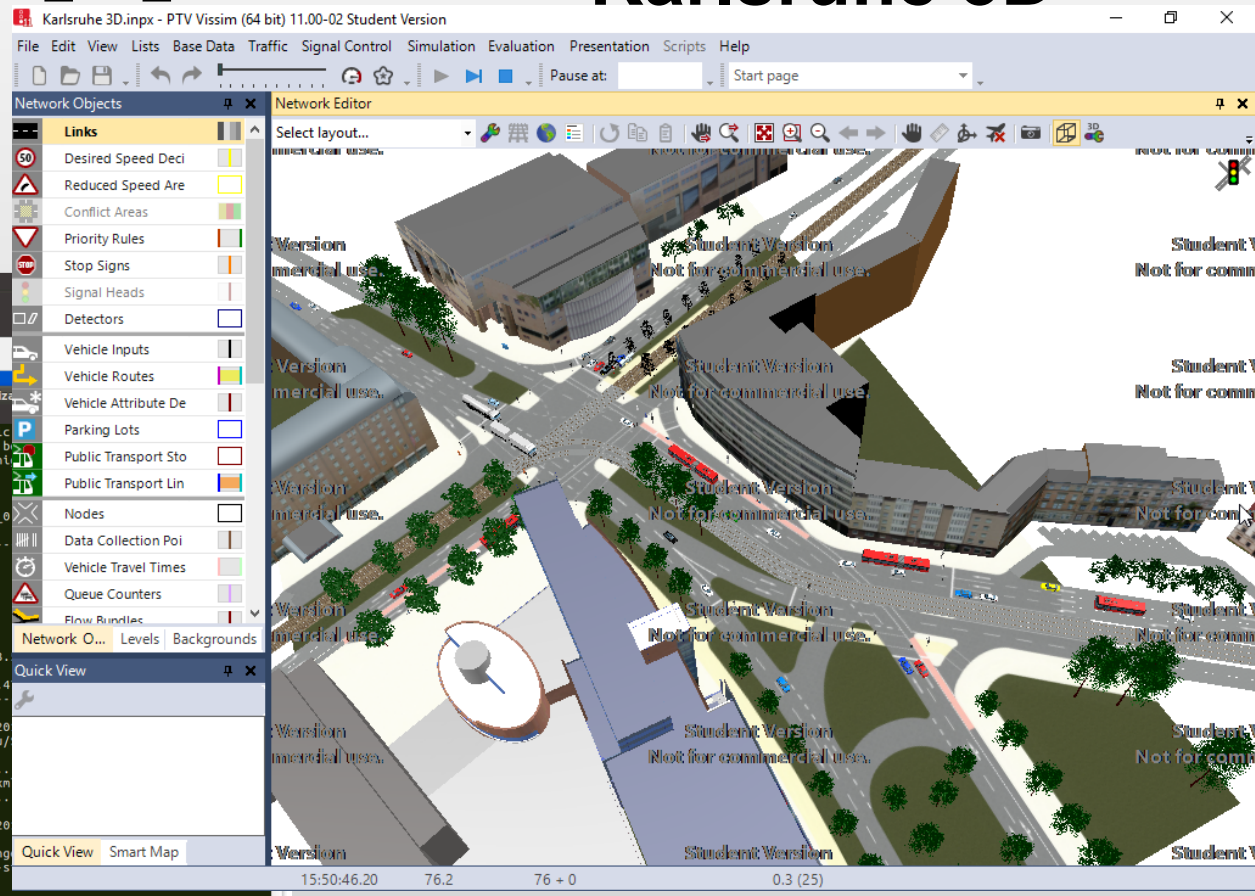
[1] Brinkhoff T. A framework for generating network-based moving objects[J]. GeoInformatica, 2002, 6(2): 153-180.
[2] Düntgen C, Behr T, Güting R H. BerlinMOD: a benchmark for moving object databases[J]. The VLDB Journal—The International Journal on Very Large Data Bases, 2009, 18(6): 1335-1368.

SUMO[1] and Vissim[2]

- Microscopic Traffic Simulator



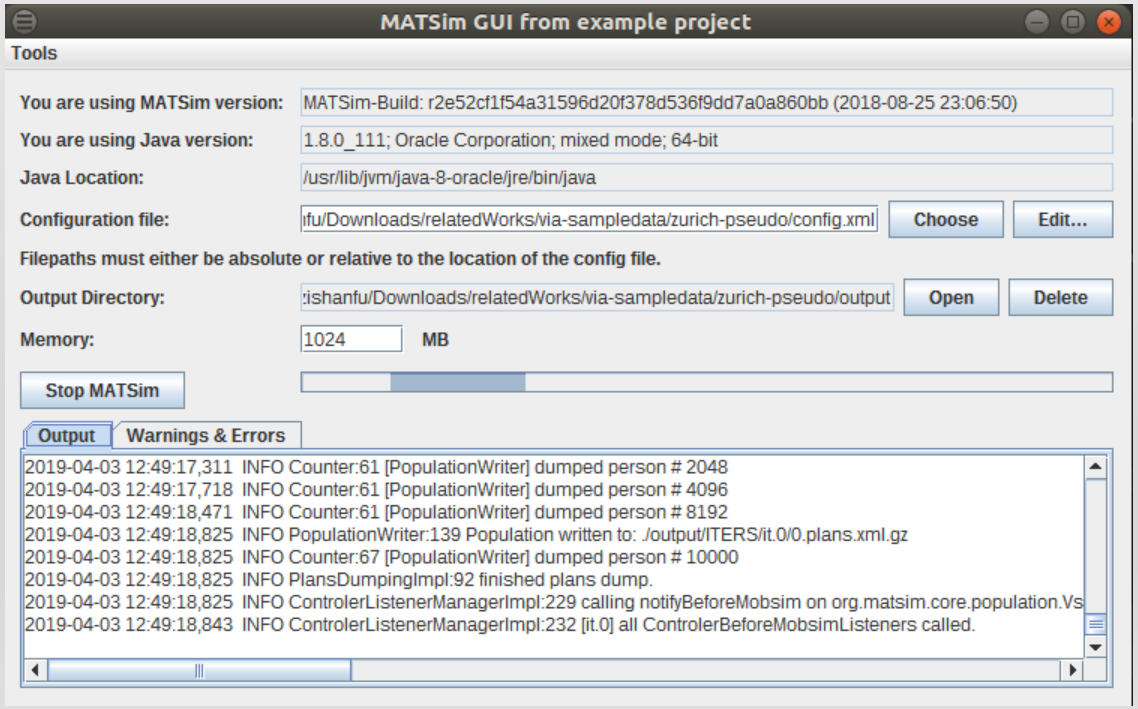
Karlsruhe 3D



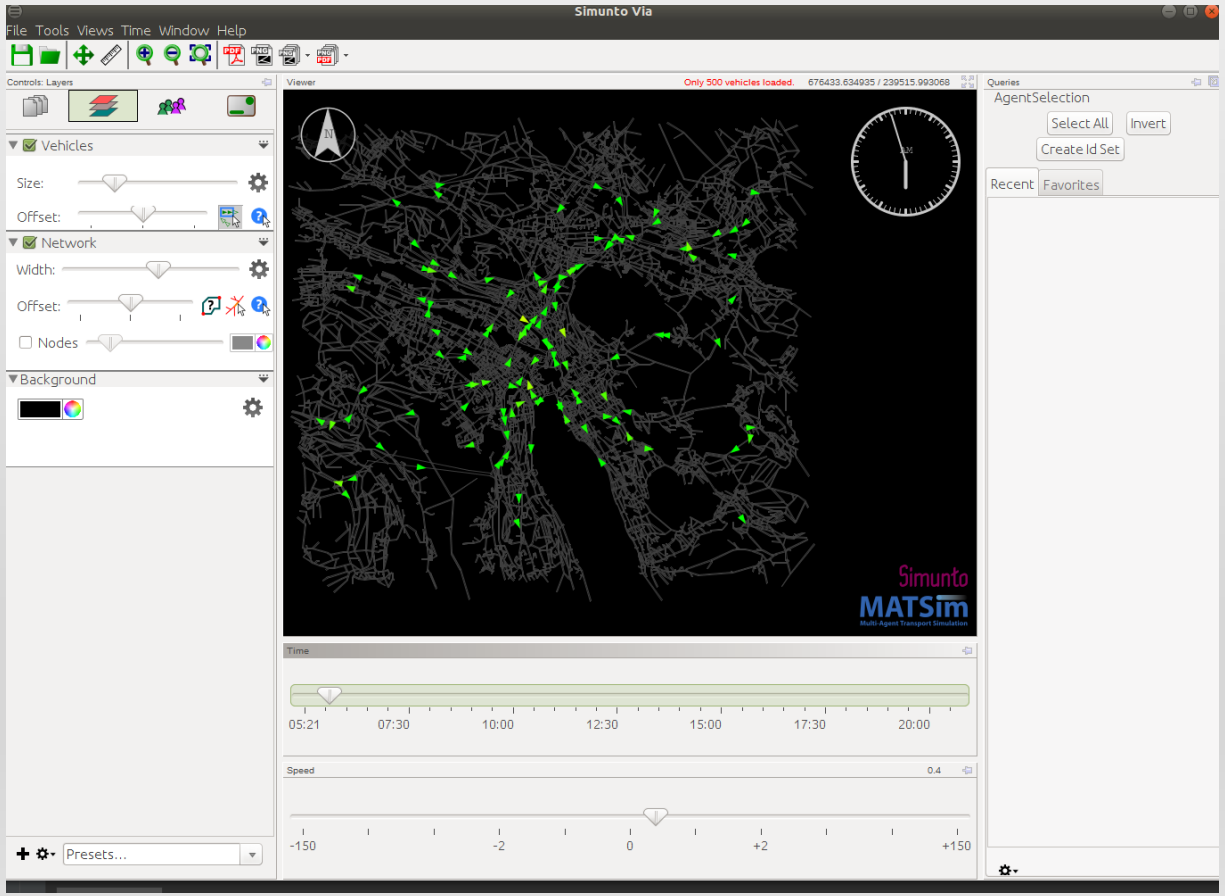
[1] Krajzewicz D, Hertkorn G, Rössel C, et al. SUMO (Simulation of Urban MObility)-an open-source traffic simulation[C]//Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002). 2002: 183-187.
[2] Vissim, http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/

MATSim[1]

- Parallel traffic microscopic simulator



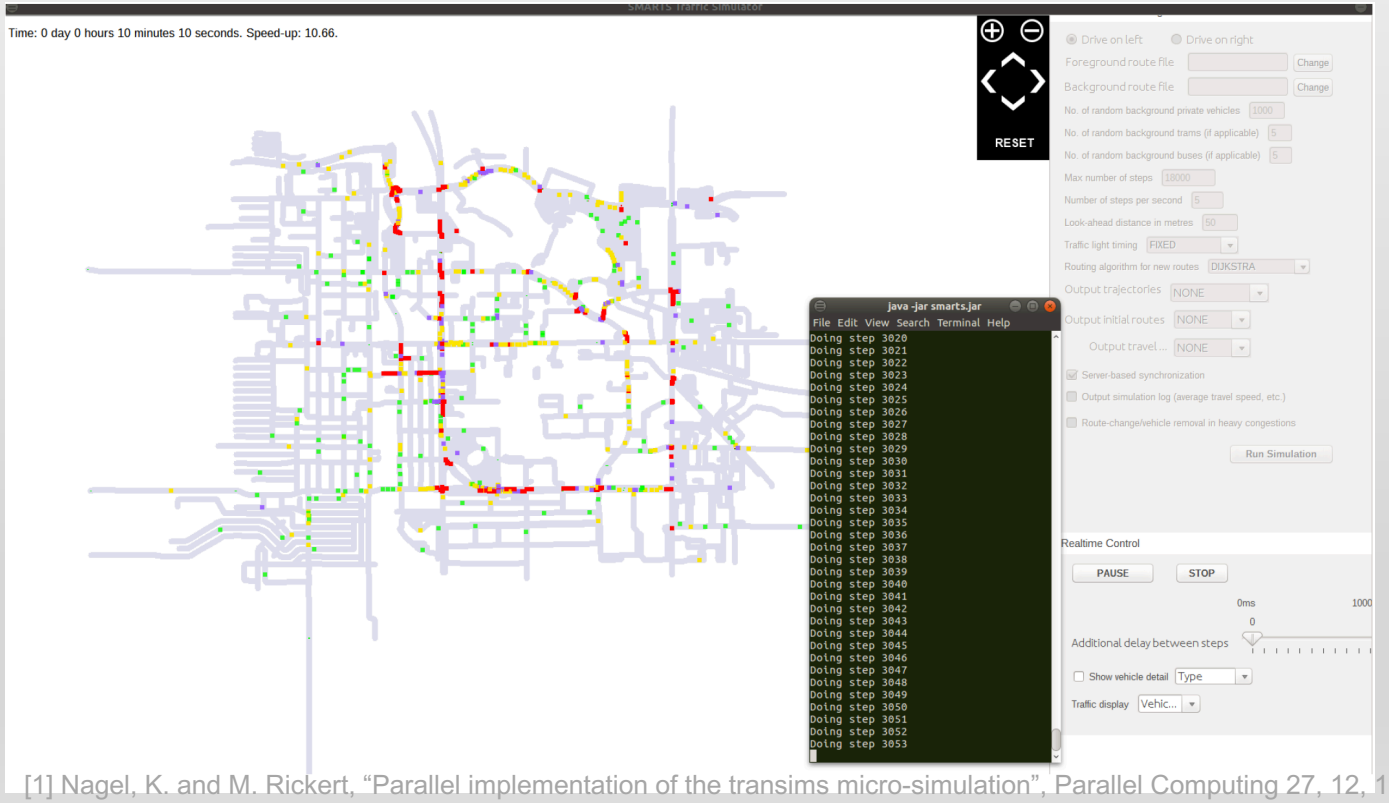
Zurich

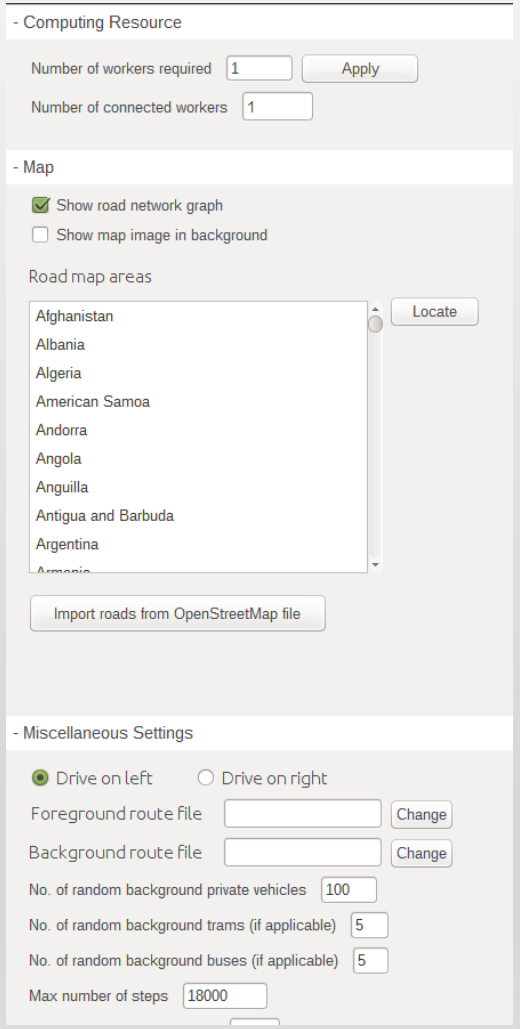


[1] Balmer M, Rieser M, Meister K, et al. MATSim-T: Architecture and simulation times[M]//Multi-agent systems for traffic and transportation engineering. IGI Global, 2009: 57-78.

Transims [1], ParamGrid [2] and SMARTS [3]

- Distributed Microscopic Simulator





[1] Nagel, K. and M. Rickert, “Parallel implementation of the transims micro-simulation”, Parallel Computing 27, 12, 1611–1639 (2001).
[2] Klefstad, R., Y. Zhang, M. Lai, R. Jayakrishnan and R. Lavanya, “A distributed, scalable, and synchronized framework for large-scale microscopic traffic simulation”, in “Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE”, pp. 813–818 (IEEE, 2005).
[3] Ramamohanarao, K., H. Xie, L. Kulik, S. Karunasekera, E. Tanin, R. Zhang and E. B. Khunayn, “Smarts: Scalable microscopic adaptive road traffic simulator”, ACM Transactions on Intelligent Systems and Technology (TIST) 8, 2, 26 (2017).

ASU Arizona State University

Comparison

GeoSparkSim

- Brinkhoff, BerlinMOD — Driving behaviors
- SUMO, Vissim — Scalable
- MATSim — Workload balancing
- Transims, ParamGrid, SMARTS — Spatial distribution over time

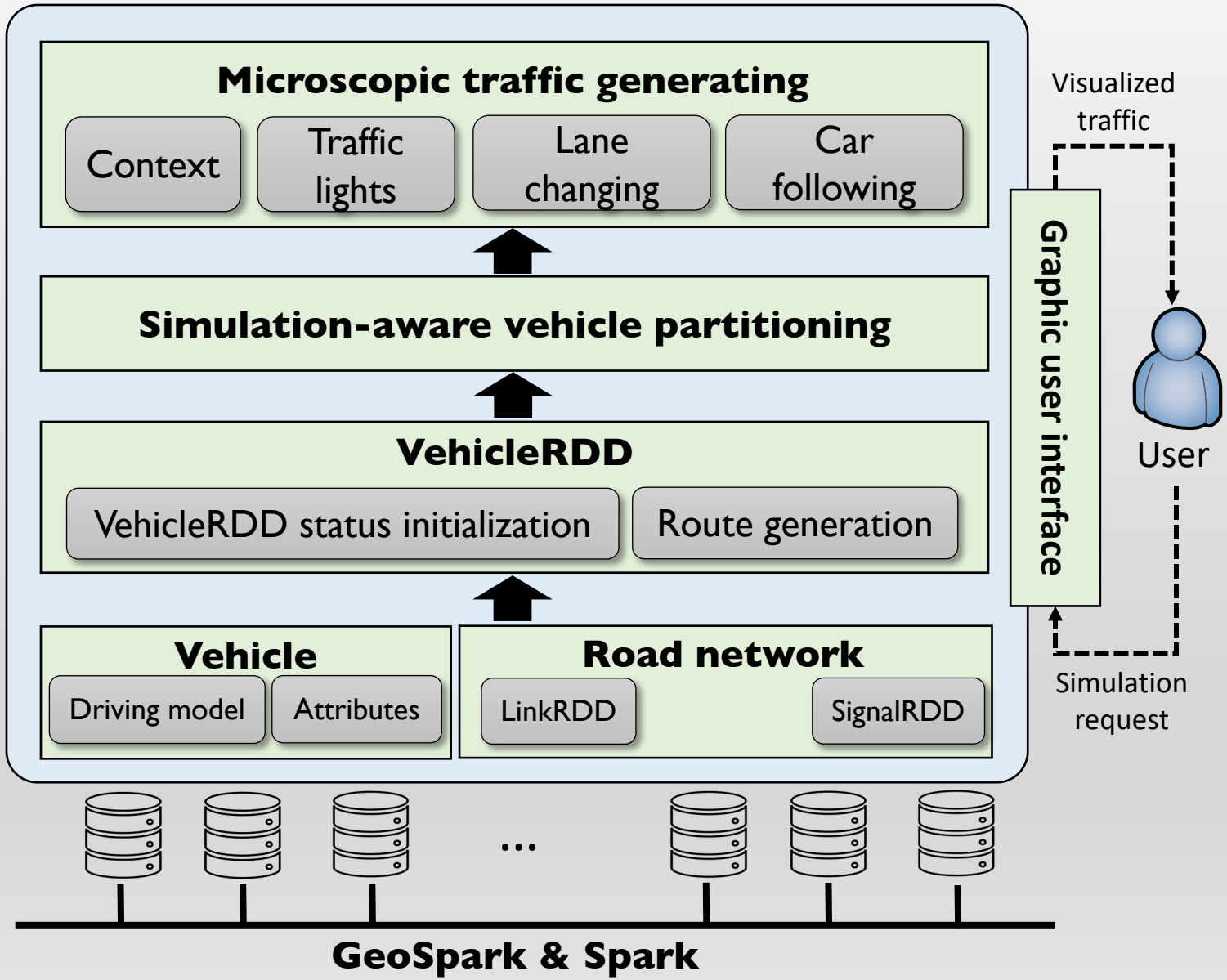
Large-scale data analytics in Apache Spark and GeoSpark

GeoSparkSim

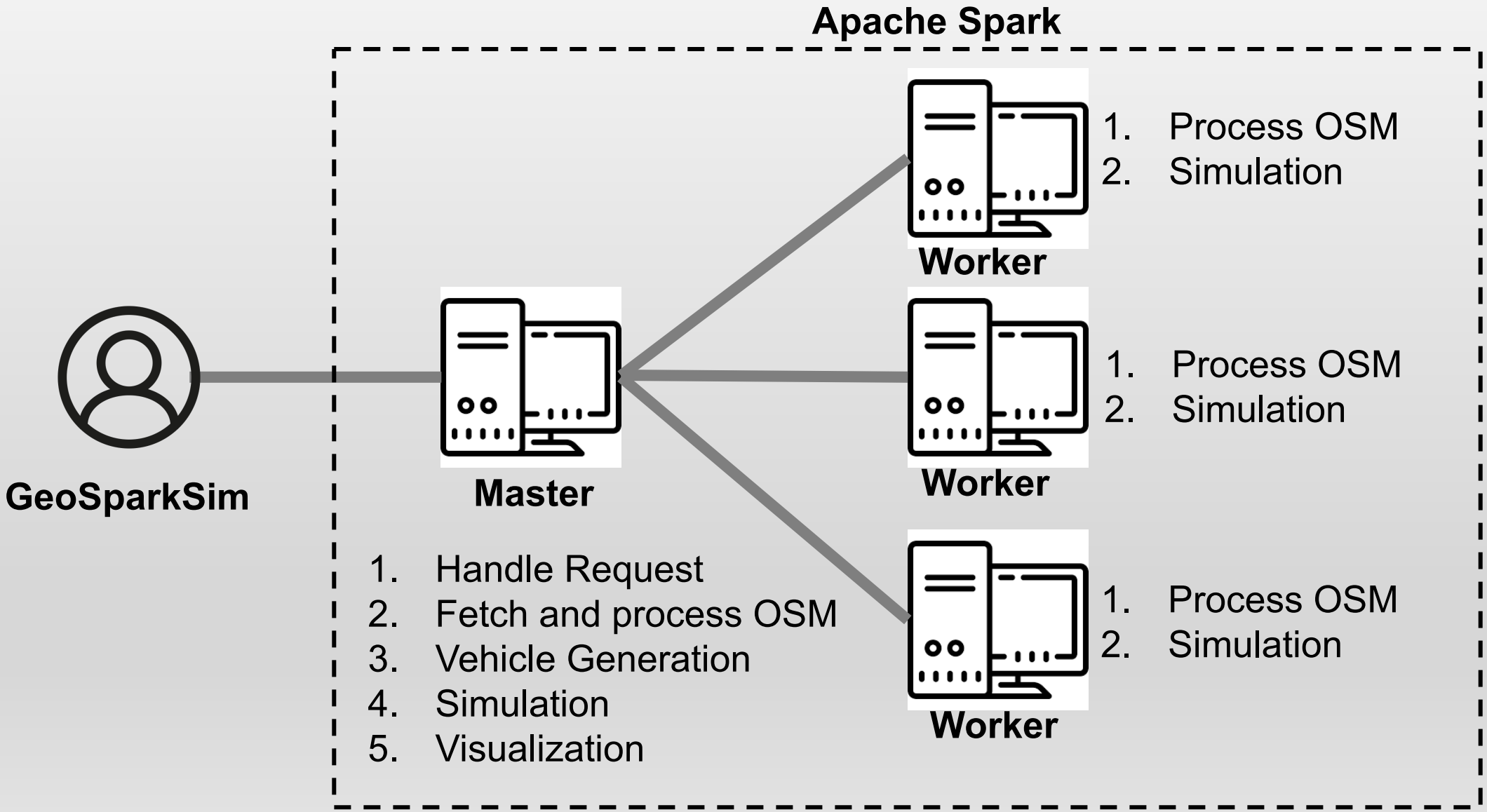
- Related Works
- **Architecture**
- Simulation
- Performance
- Demo

ASU Arizona State University

Architecture

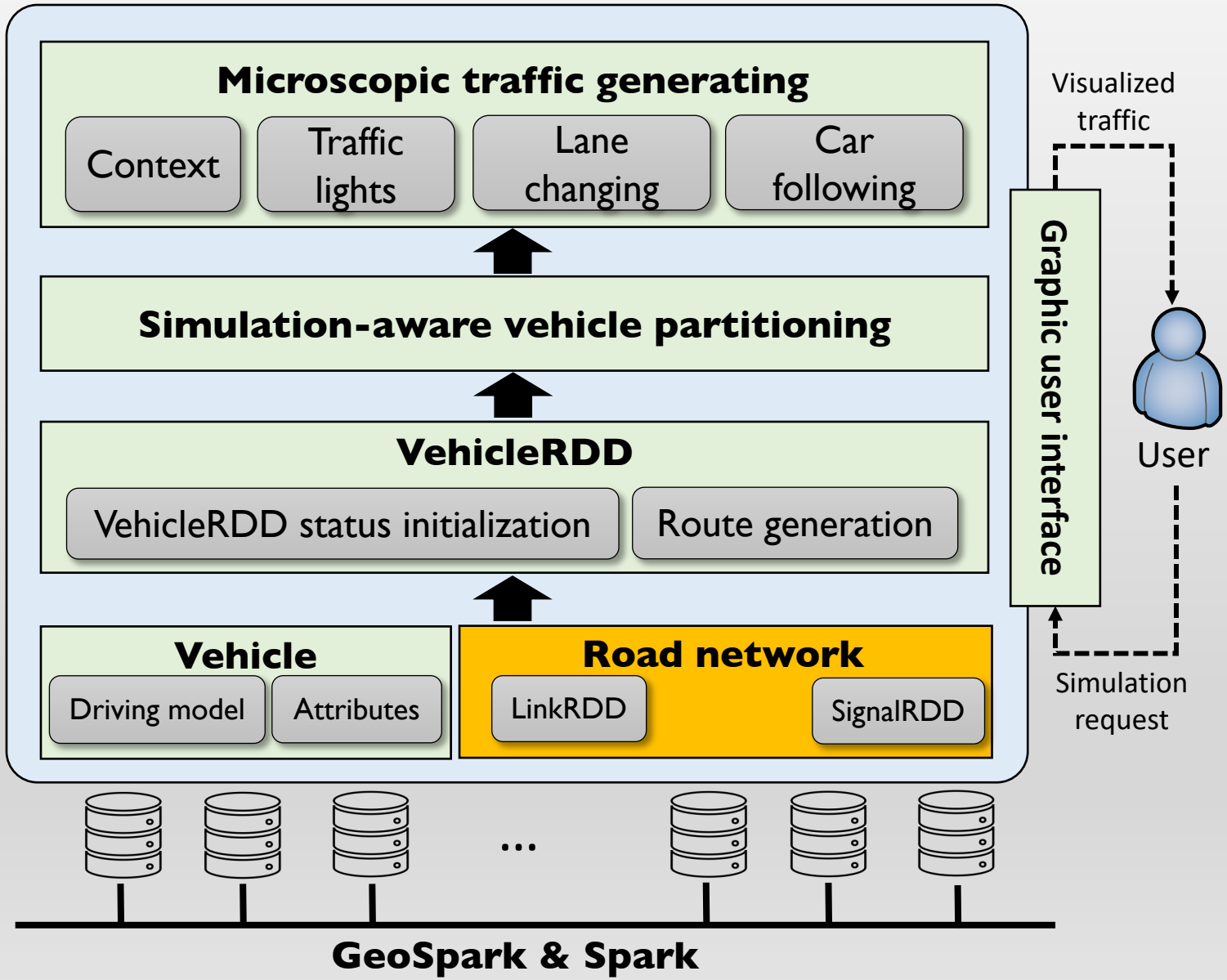


Distributed Architecture

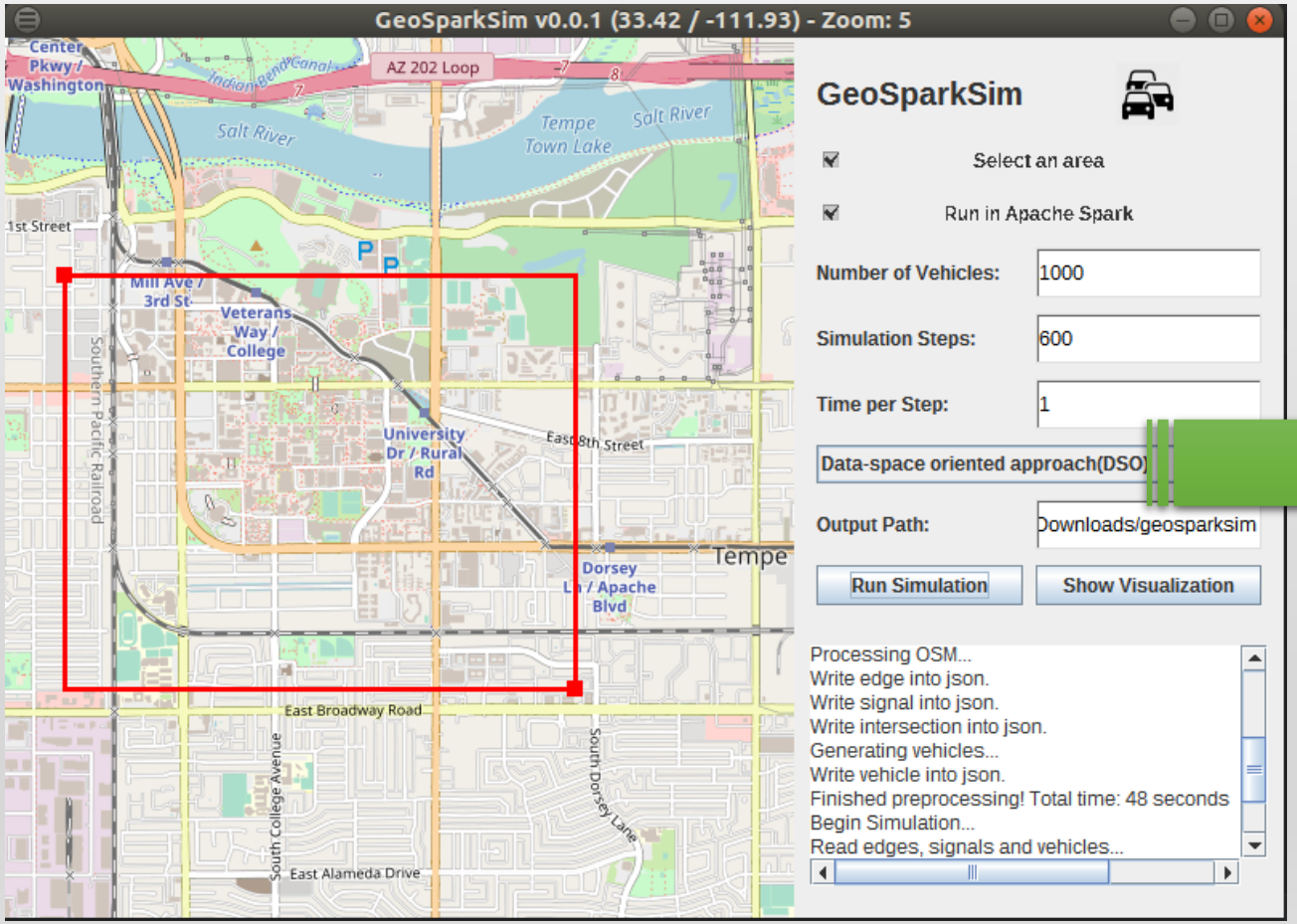


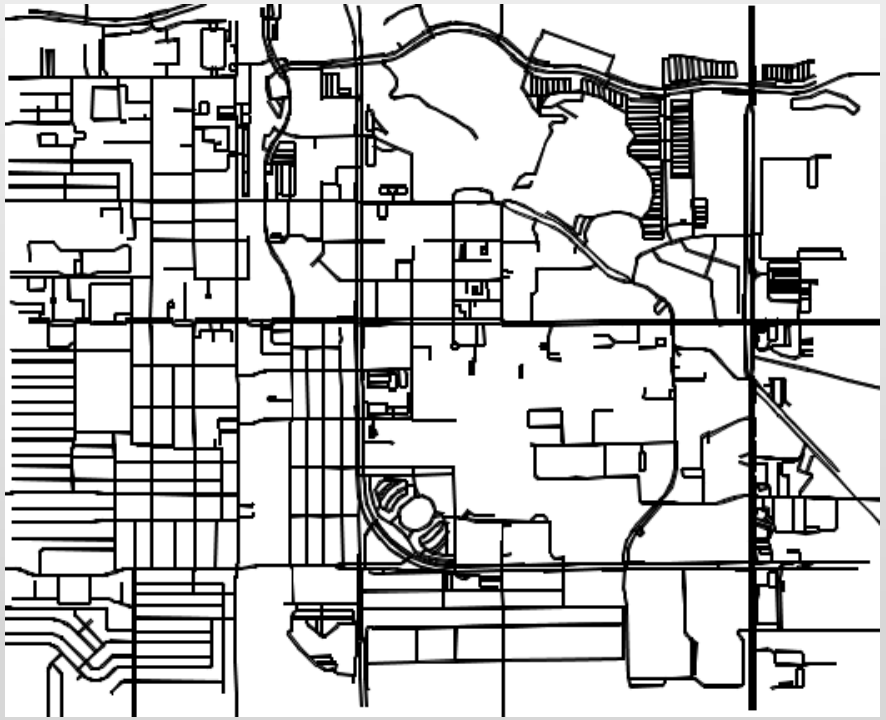
ASU Arizona State University

Architecture



Road Network





ARIZONA STATE UNIVERSITY

Road Network

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="Overpass API">

<node id="1" lat="33.4231268" lon="-111.9304264">
   <tag k="highway" v="traffic_signals"/>
</node>
<node id="2" lat="33.4245016" lon="-111.9275146" />

<way id="1">
   <nd ref="1"/>
   <nd ref="2"/>
   <tag k="highway" v="primary"/>
   <tag k="lanes" v="8"/>
   <tag k="oneway" v="no"/>
</way>

</osm>
```

```
Node: {
    ID: 1,
    Latitude: 33.4231268,
    Longitude: -111.9304264,
    Tags: {
        tag1: {
            key: highway,
            value: traffic_signals
        },
        tag2: …
    }
}
```

SignalRDD

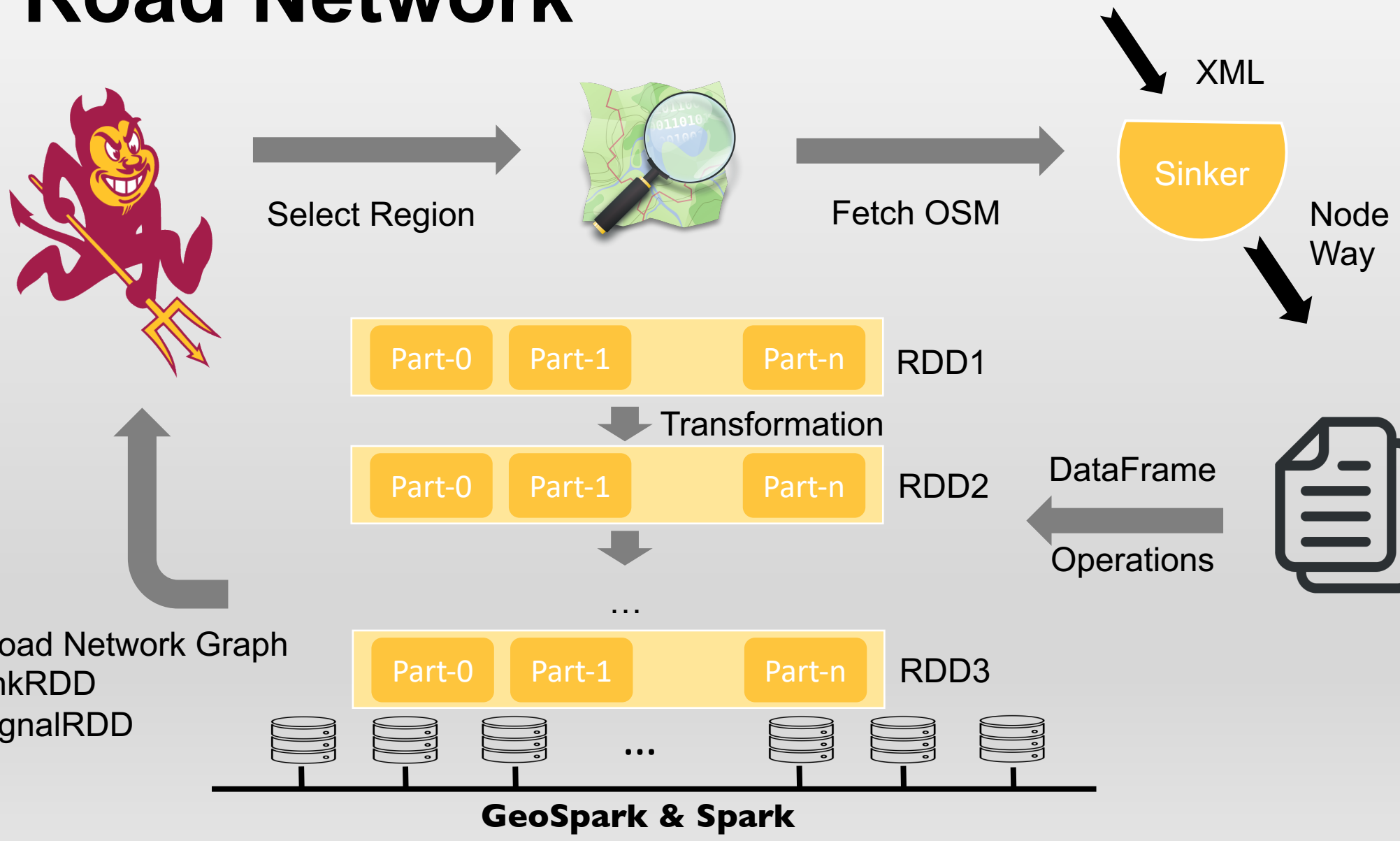
```
Node
ID: long
Tags: array
    element: struct
        key: string
        value: string
Latitude: double
Longitude: double
```

```
Way: {
    ID: 1,
    Nodes: {
        node1: {
            index: 1,
            nodeID: 1
        },
        node2: {
            index: 2,
            nodeID: 2
        }
    },
    Tags: {
        tag1: {
            key: highway,
            value: traffic_signals
        },
        tag2: …
    }
}
```

LinkRDD

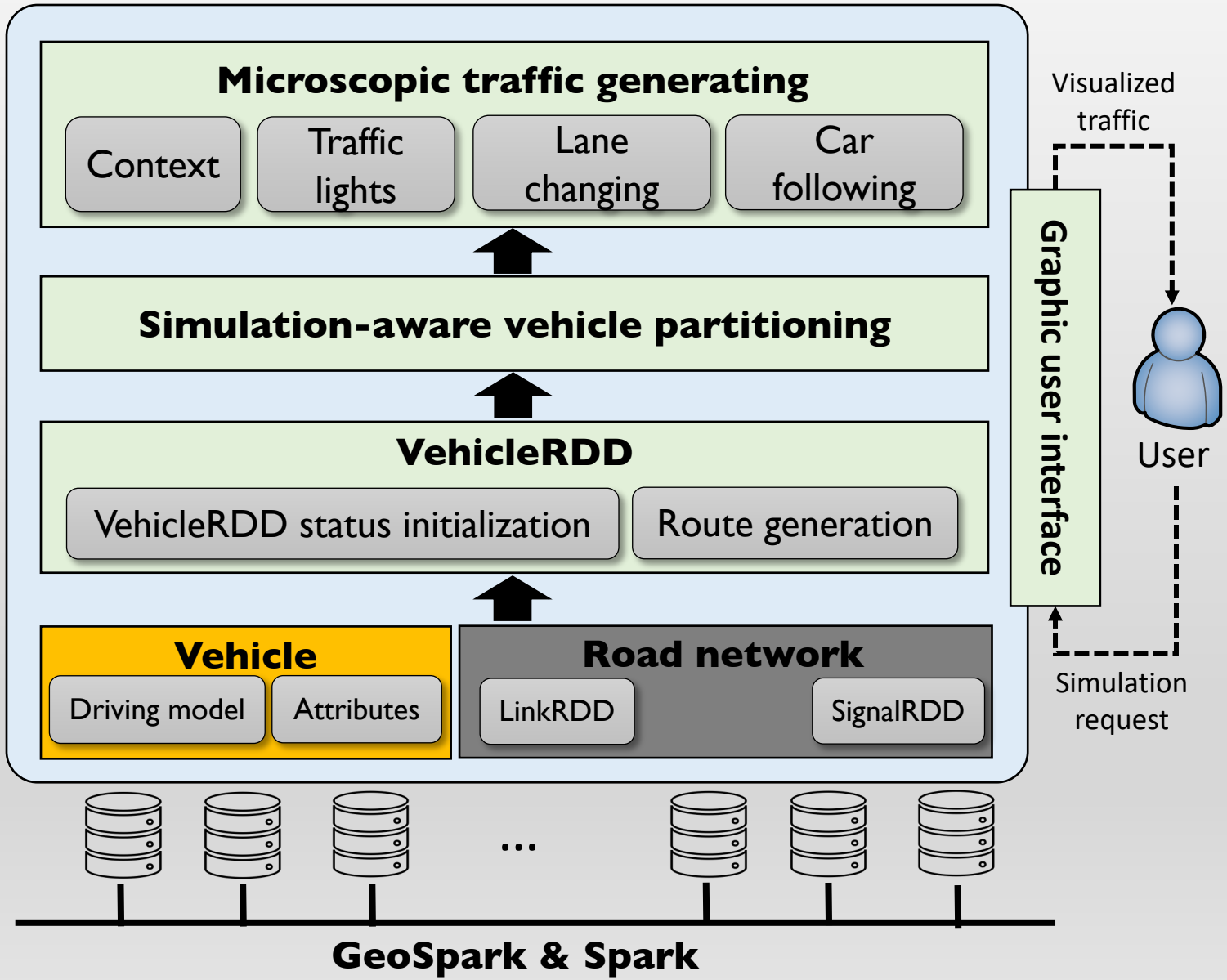
```
Way
ID: long
Tags: array
    element: struct
        key: string
        value: string
Nodes: array
    element: struct
        index: integer
        nodeID: long
```

Road Network



ASU Arizona State University

Architecture



Vehicle

Black box: field
Red box: method

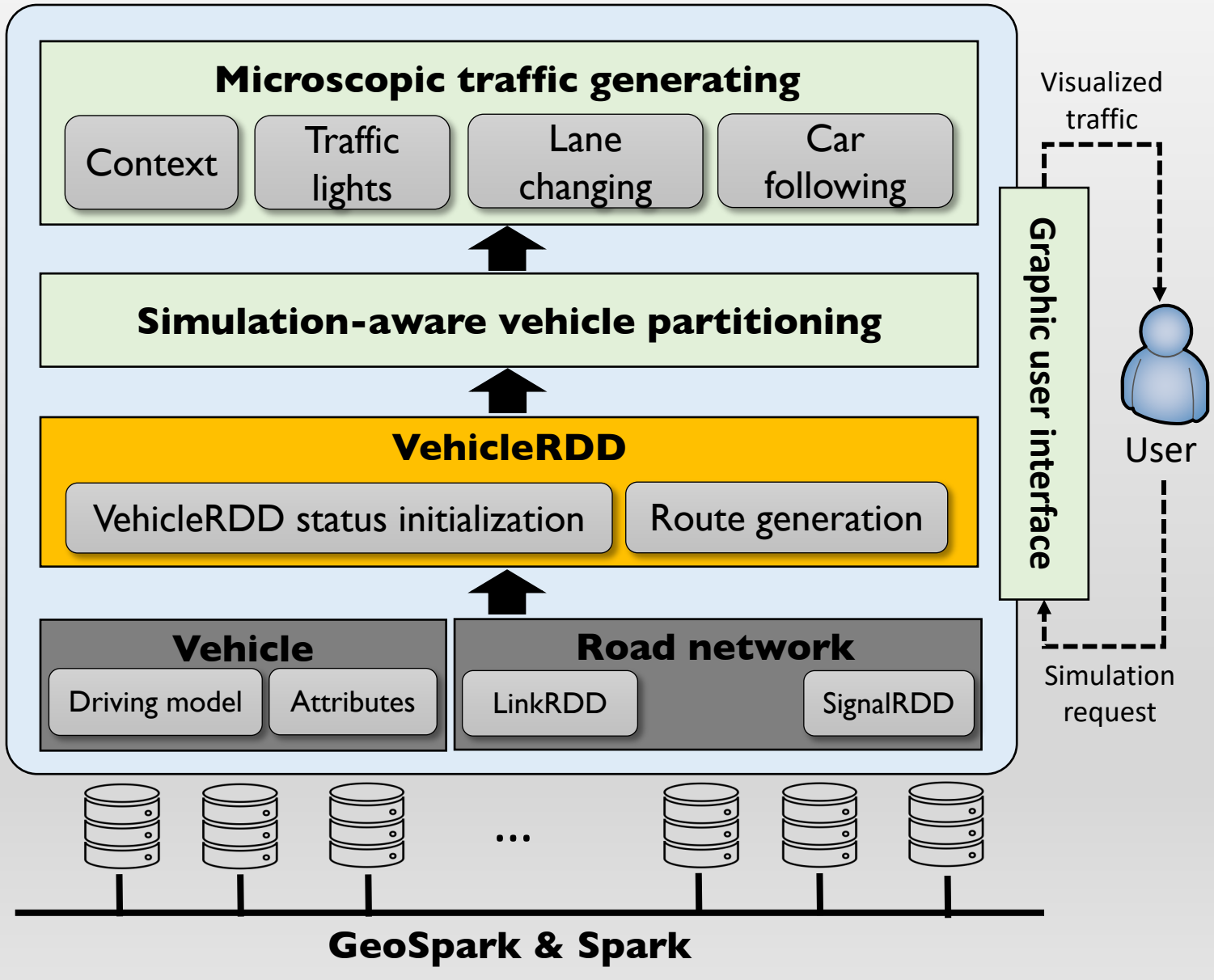
Vehicle
ID: String
Source: Coordinate
Target: Coordinate
EdgePath: Long Array
Costs: Double Array
FullPath: Coordinate Array
Front: Coordinate
Rear: Coordinate
edgeIndex: int
carLength: 3
currentLane: int
isArrive: boolean
currentLink: Link

getLinkByHead
headWayCheck
basicMovement
born

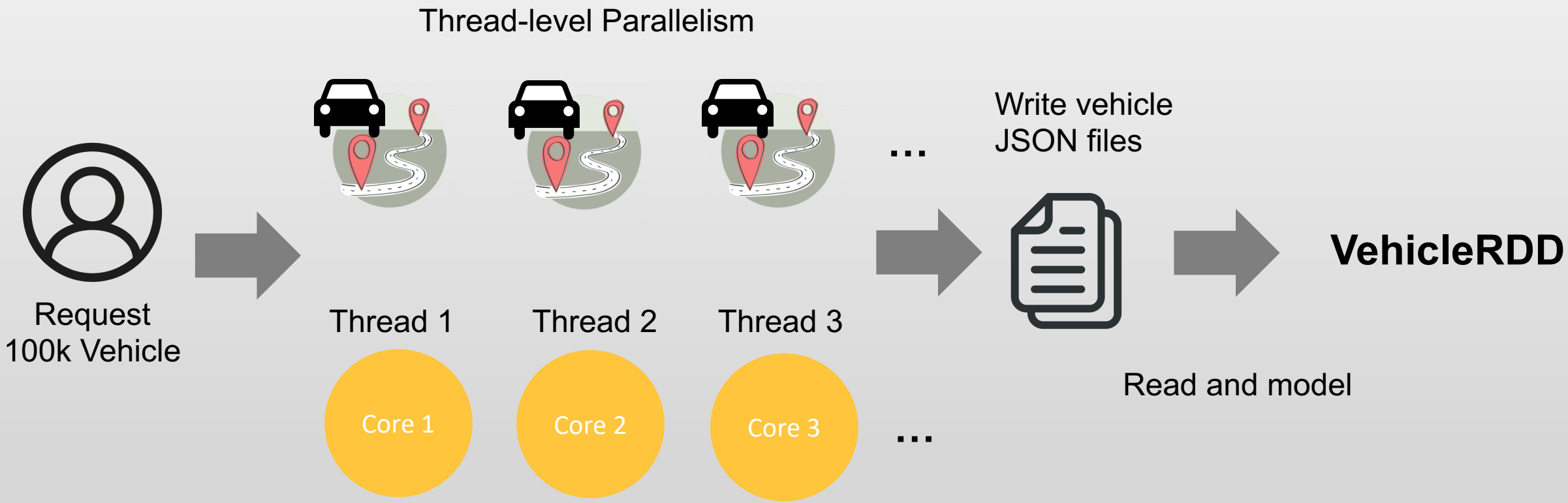
- Vehicle ID
- Source coordinate
- Destination coordinate
- Compute path trajectory
- Array of link ID
- Array of link distance cost
- Other attributes (car length, etc)

ARIZONA STATE UNIVERSITY

Architecture



Vehicle to VehicleRDD



List of Vehicles -> vehicle JSON -> VehicleRDD

GeoSparkSim

- Related Works
- Architecture
- **Simulation**
- Performance
- Demo

ASU Arizona State University

Simulation

Input
Simulation Period
LinkRDD
SignalRDD
VehicleRDD

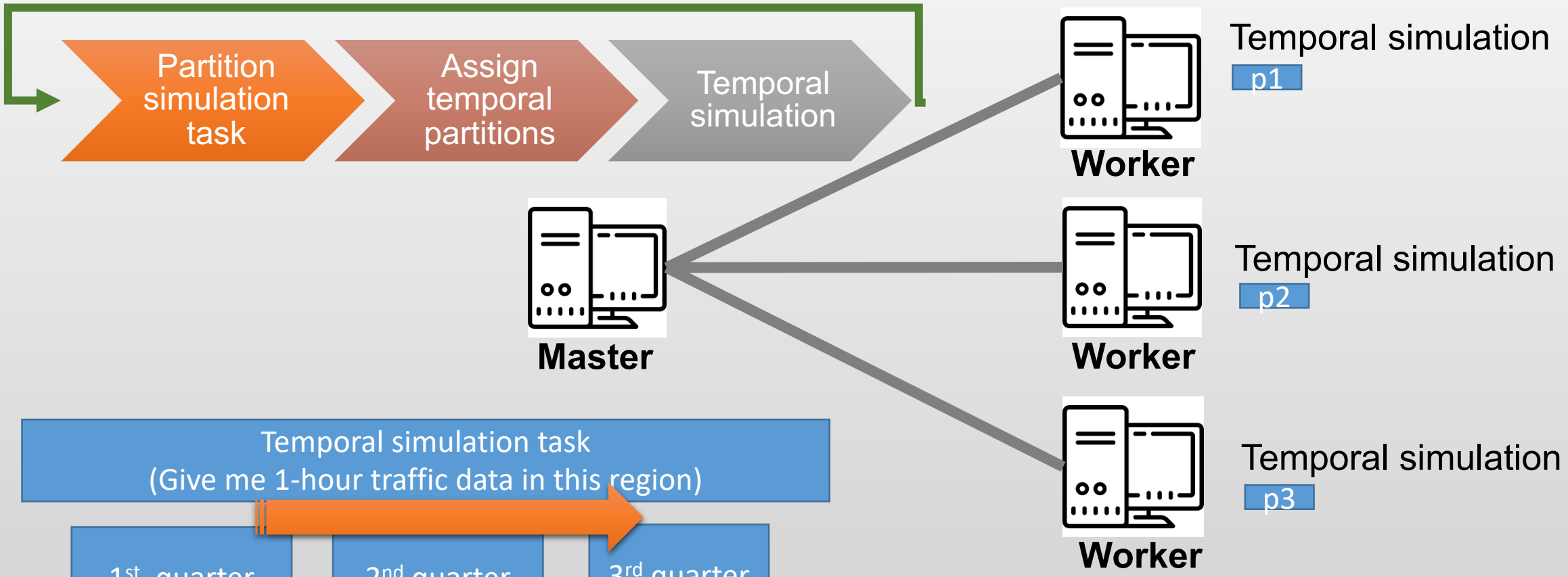
Simulation Steps

Timing (say, every 1 second)

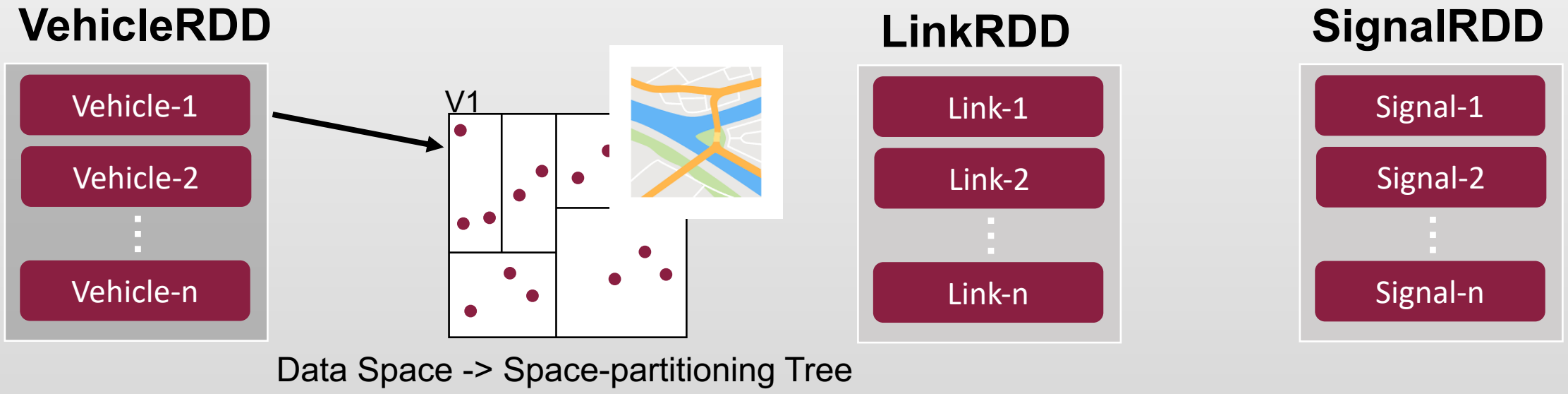
1. **Check nearby objects**
2. **Update acceleration**
3. **Check lane change**
4. **Update new coordinate**

Output (StepReportRDD)
Signal
Vehicle

Workload Distribution



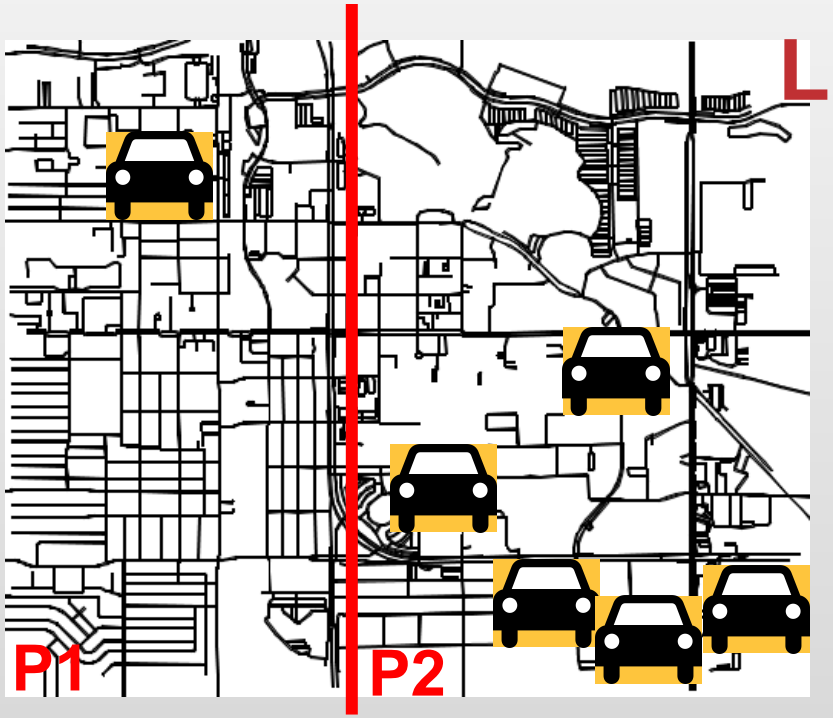
Spatial Resilient Distributed Dataset



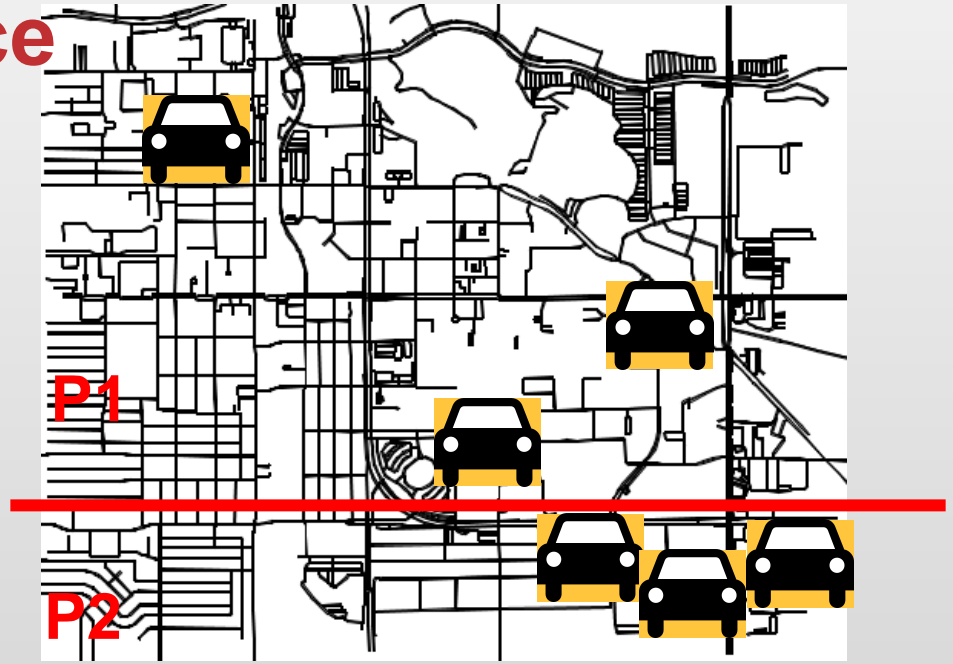
Apply same partition to linkRDD and signalRDD

ASU Arizona State University

Spatial Workload Distribution



Load balance



Partition by road network

P1 Simulation

P2 Simulation

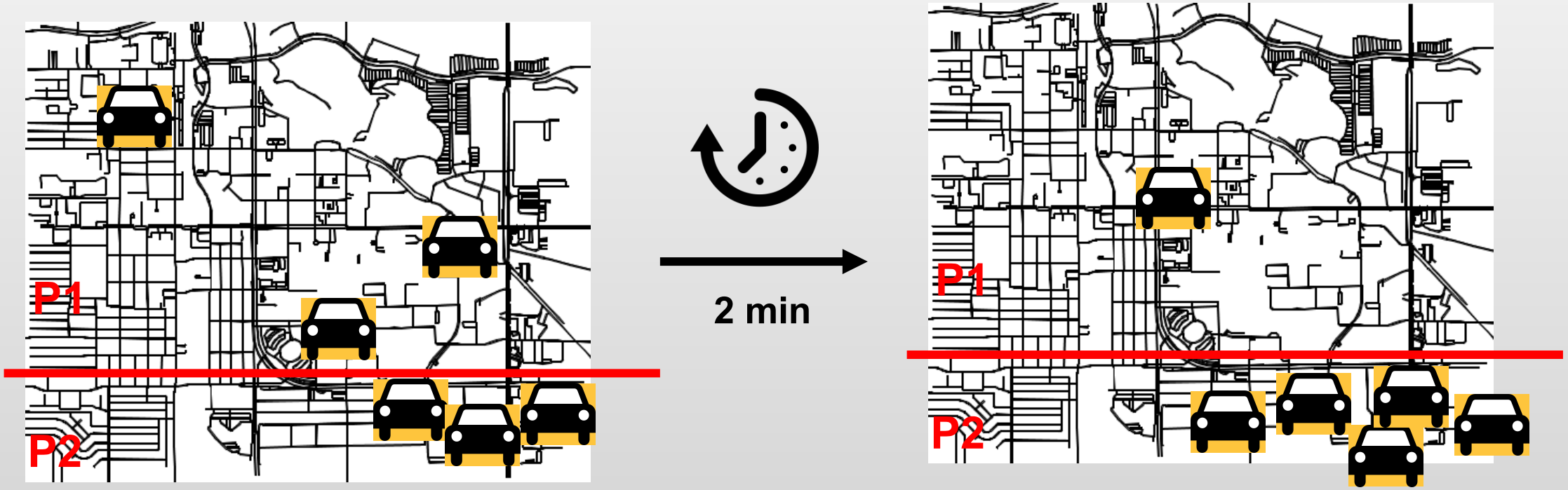
Partition by vehicle density

P1 Simulation

P2 Simulation

ASU Arizona State University

Spatial Workload Balancing



P1 Simulation

P2 Simulation

Uneven!

P1 Simulation

P2 Simulation

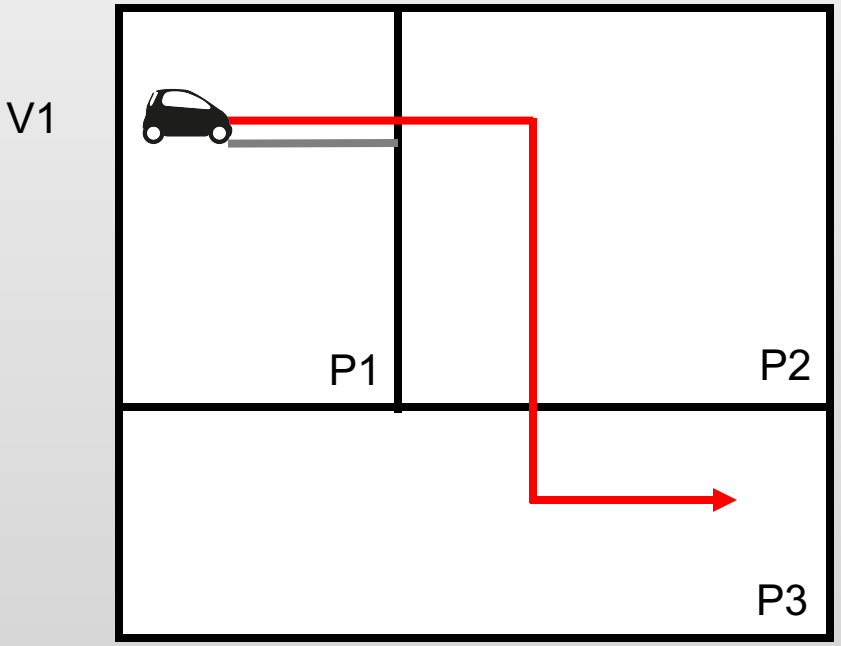
How to do spatial partition in the simulation?

Vehicles are moving!

Source coordinate?

Planned route?

Partition by Source coordinate

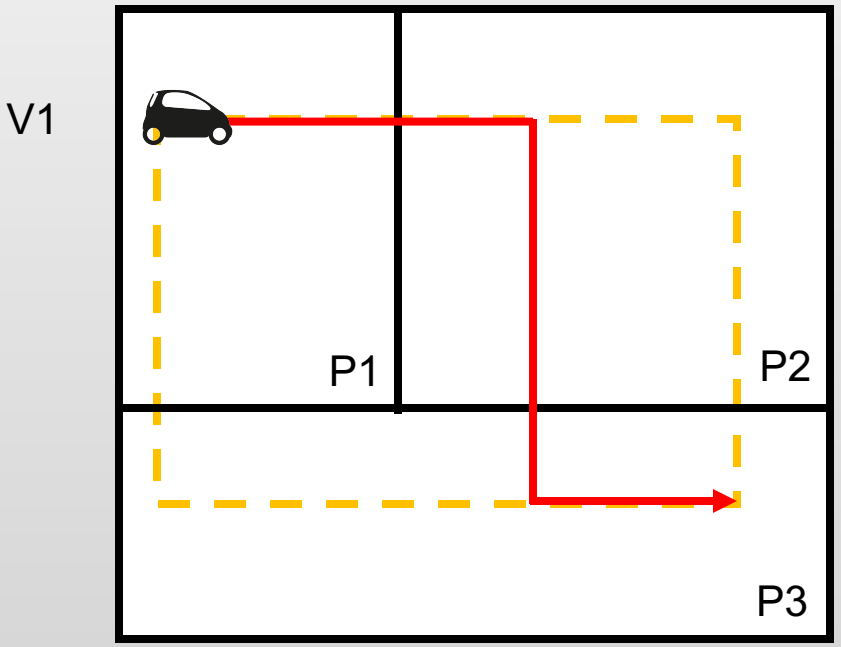


Update

P1: V1 ⟶ **P2: V1**

Message exchange, inefficient!

Partition by planned routes



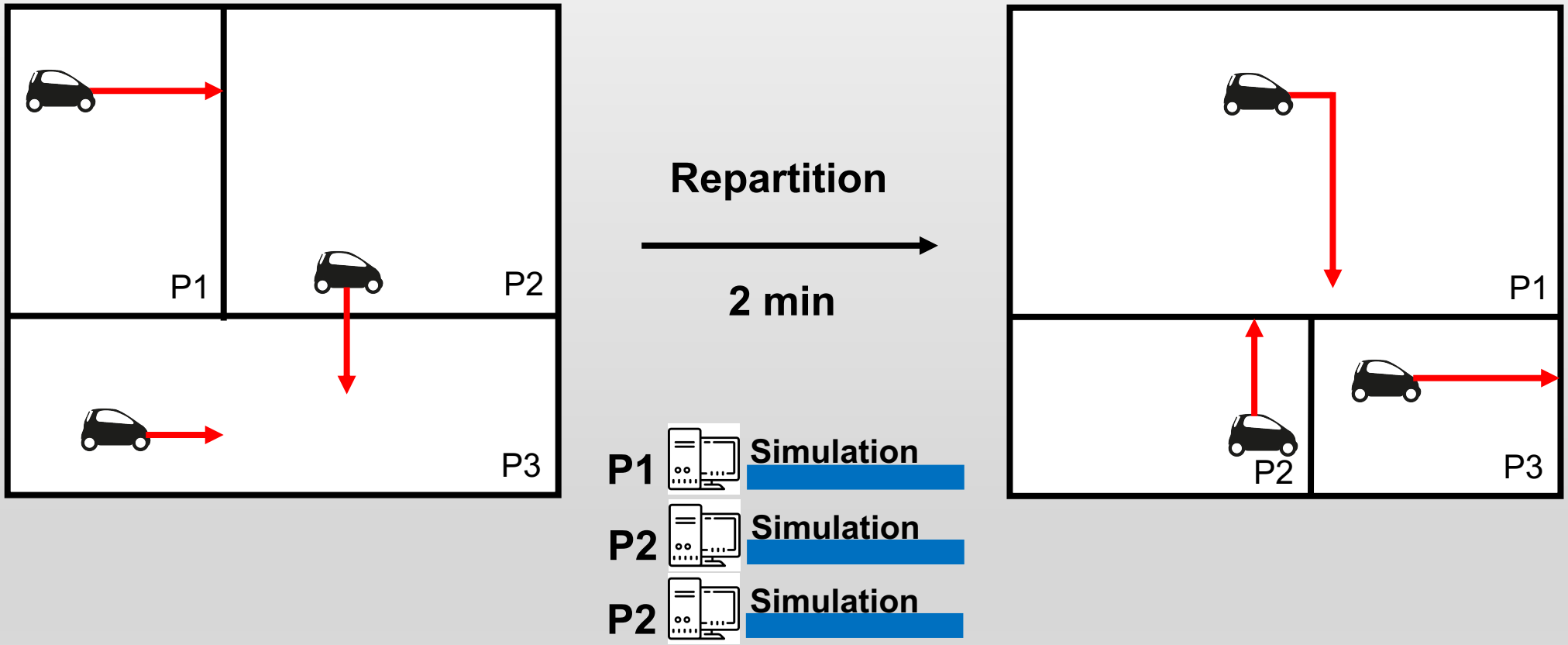
Minimum Bounding Rectangle (MBR)

P1: V1
P2: V1
P3: V1

Routes are too long
Overlaps, intersections!

ARIZONA STATE UNIVERSITY

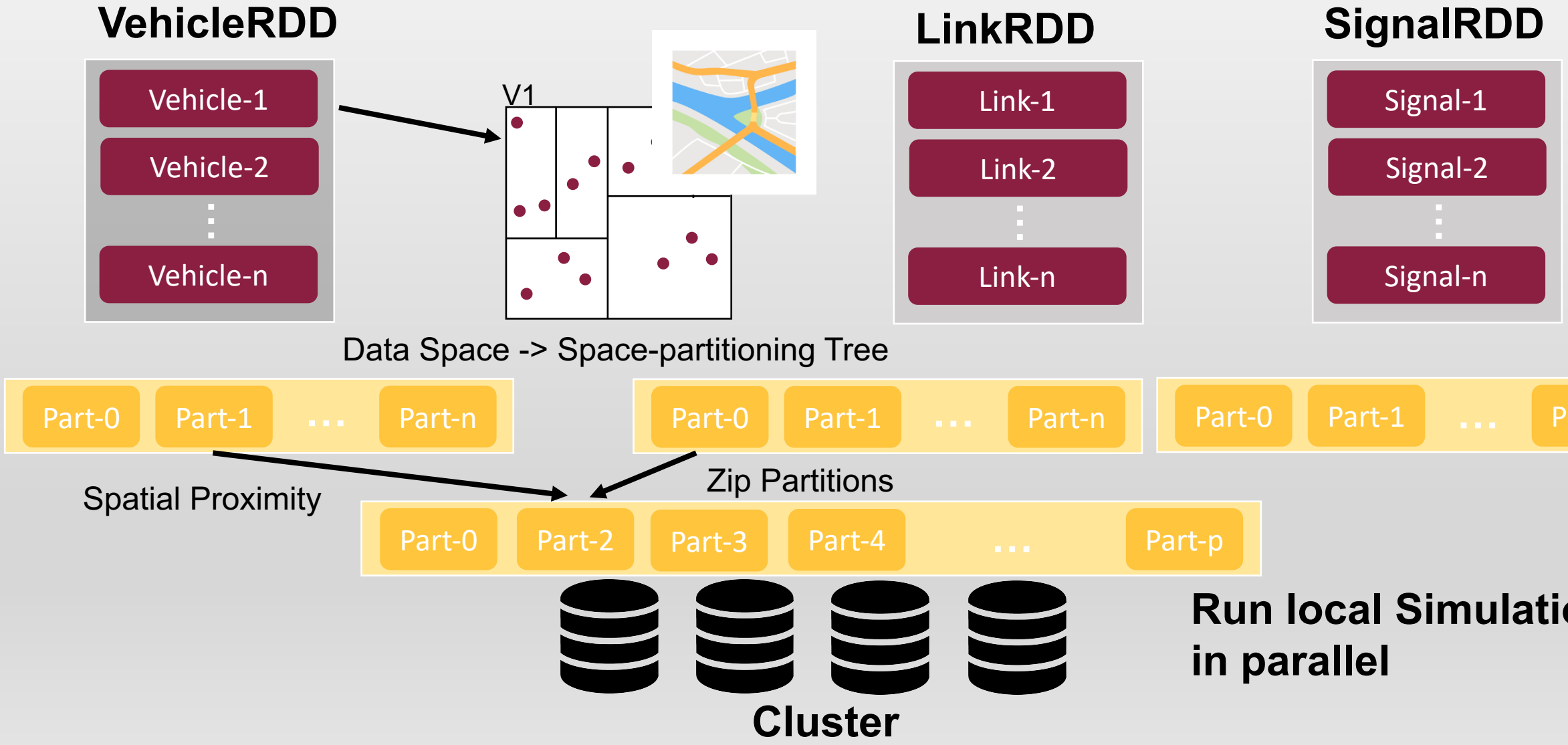
Spatial Workload Balancing



Update partition by spatial distribution over time!

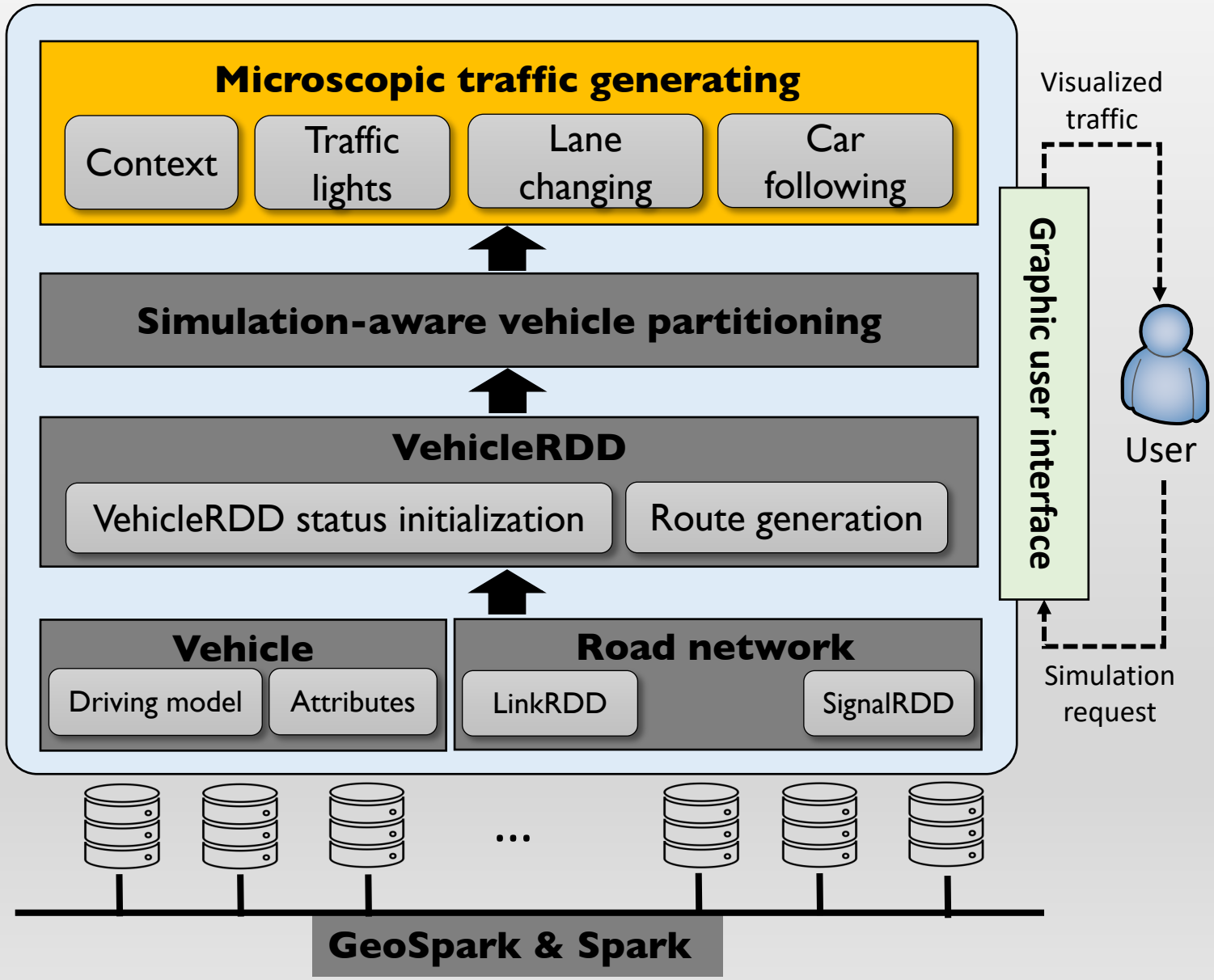
ARIZONA STATE UNIVERSITY

Spatial Resilient Distributed Dataset



ASU Arizona State University

Architecture



Simulation

Input
Simulation Period
LinkRDD
SignalRDD
VehicleRDD

Initialize RDDs

Partition

Assign Temporal Simulation

Partition Iterations

Zipped vehicleRDD, linkRDD and signalRDD

Part-0 | Part-2 | Part-3 | Part-4 | ...

Parallel

Temporal Simulation Iterations

Signal | Surroundings | Acceleration | Lane Change

StepReports

Simulation-aware repartition

Update RDDs last status | Partition

Zip

Write Step Reports to file in each iterations

Simulation

In-memory vehicleRDD, linkRDD and signalRDD

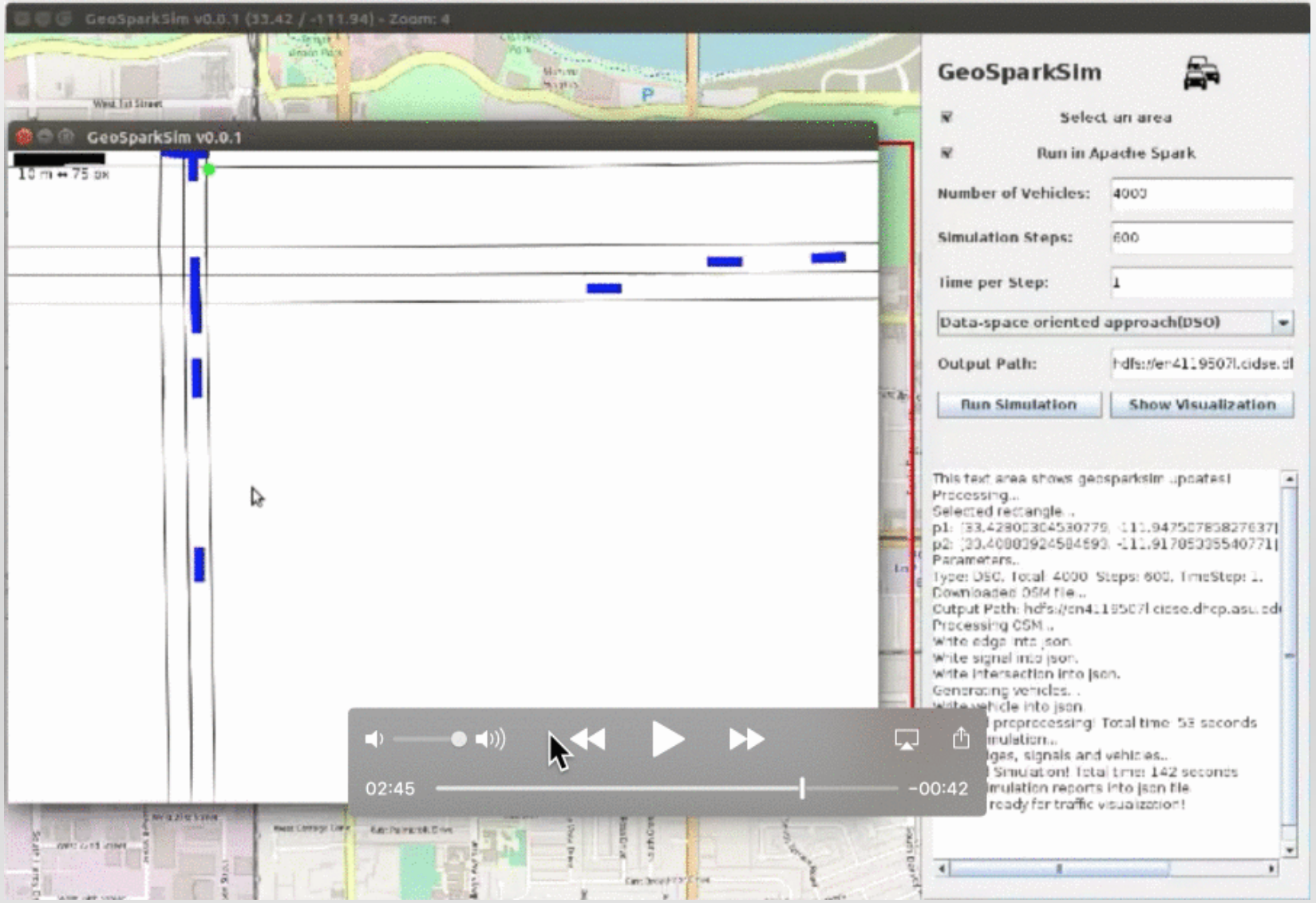
Apache Spark & GeoSpark

Output (StepReportRDD)
Signal
Vehicle

GeoSparkSim

ASU Arizona State University

Visualized Traffic Data



GeoSparkSim

- Related Works
- Architecture
- Simulation
- **Performance**
- Use

Cluster: One master and four worker

Machine: 48 cores, 100 GB memory, 4TB HDD

ASU Arizona State University

Parameters

Parameters	Range
Number of vehicles (thousand)	100, 200, 300
Time step (second)	1, 0.8, 0.6, 0.4, 0.2
Simulation period (minute)	10, 30, 60, 120
Number of Partitions	1000, 1500, 3000
Repartition period (minute)	1, 2, 4, 8

ARIZONA STATE UNIVERSITY

The Number of Vehicles

Number of Vehicles: **100k, 200k, 300k**

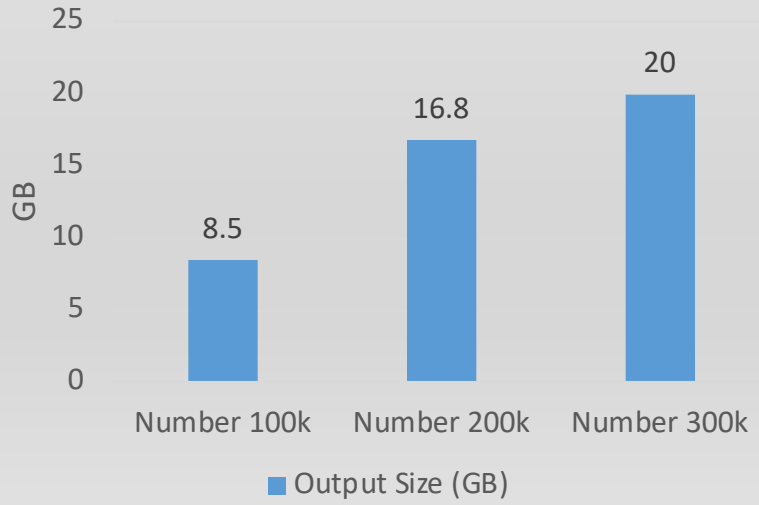
Time Step: **1 sec**

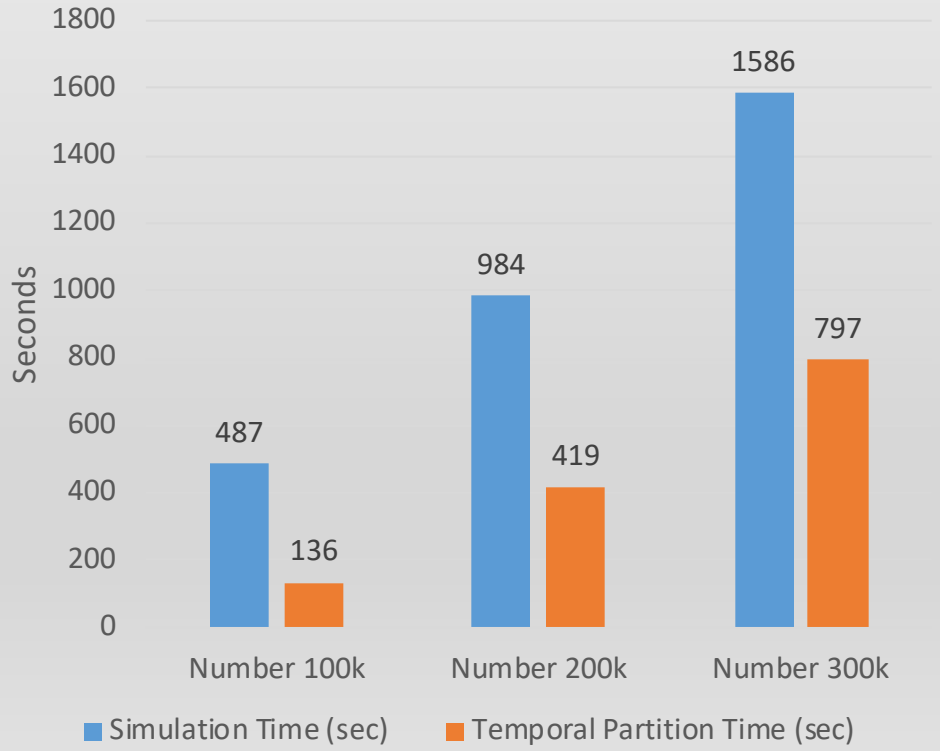
Simulation Period: **10 min**

Repartition Period: **2 min**

Number of Partition: **1500**

Simulation, temporal partition, output size linear increase





ARIZONA STATE UNIVERSITY

Simulation Period

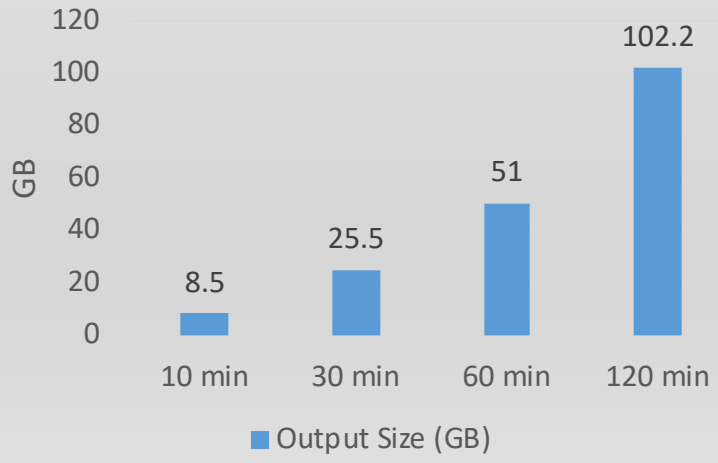
Number of Vehicles: **100k**

Time Step: **1 sec**

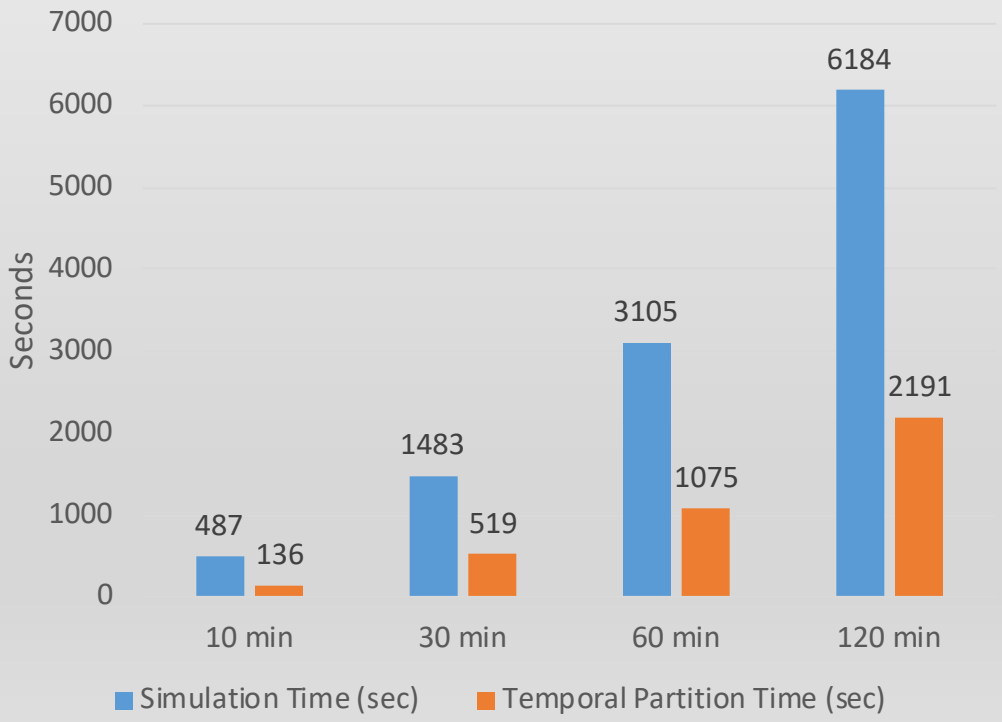
Simulation Period: **10, 30, 60, 120 min**

Repartition Period: **2 min**

Number of Partition: **1500**



Linear increase



ARIZONA STATE UNIVERSITY

Time per Steps

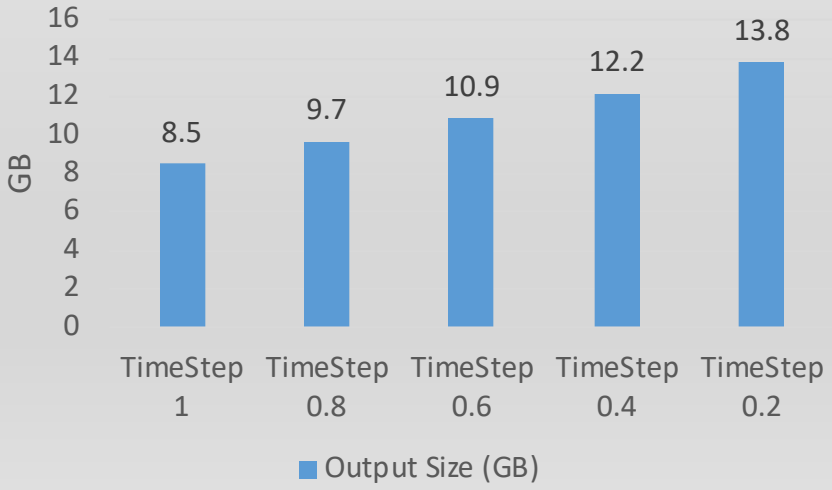
Number of Vehicles: **100k**

Time Step: **1, 0.8, 0.6, 0.4, 0.2 sec**

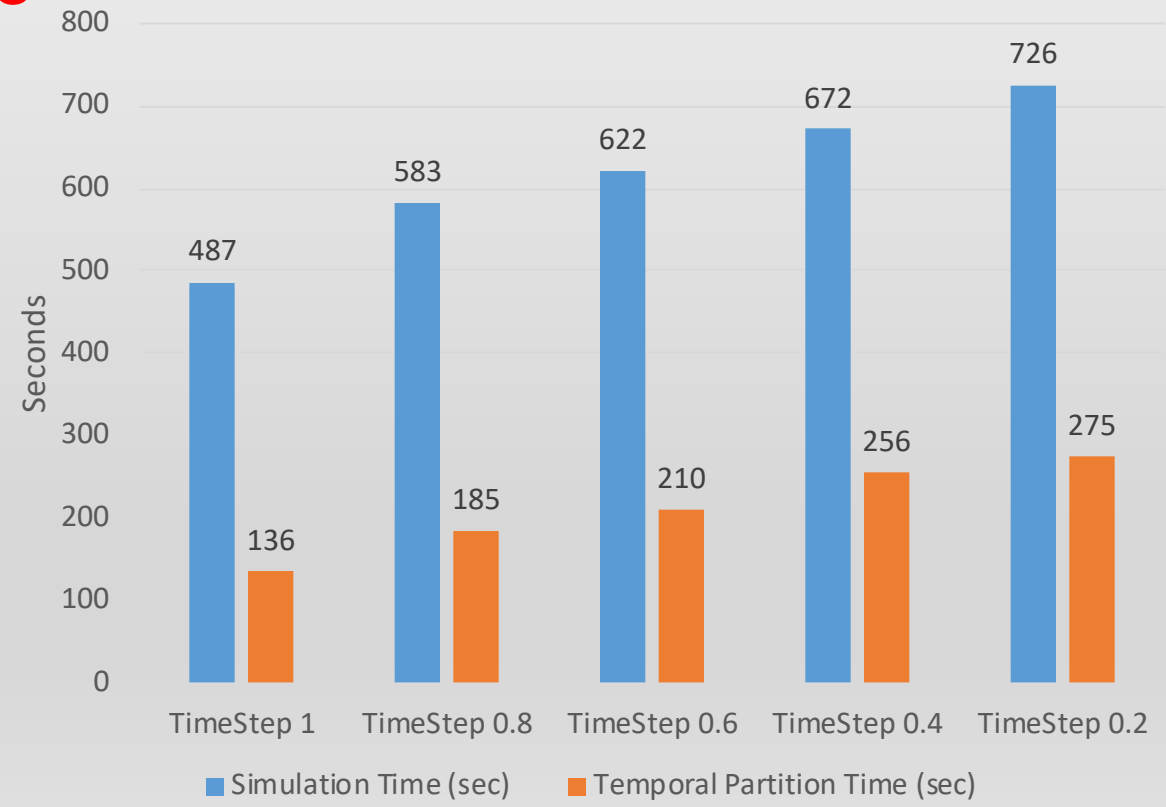
Simulation Period: **10 min**

Repartition Period: **2 min**

Number of Partition: **1500**



Linear increase



ASU Arizona State University

The Number of Partitions

Number of Vehicles: **100k**

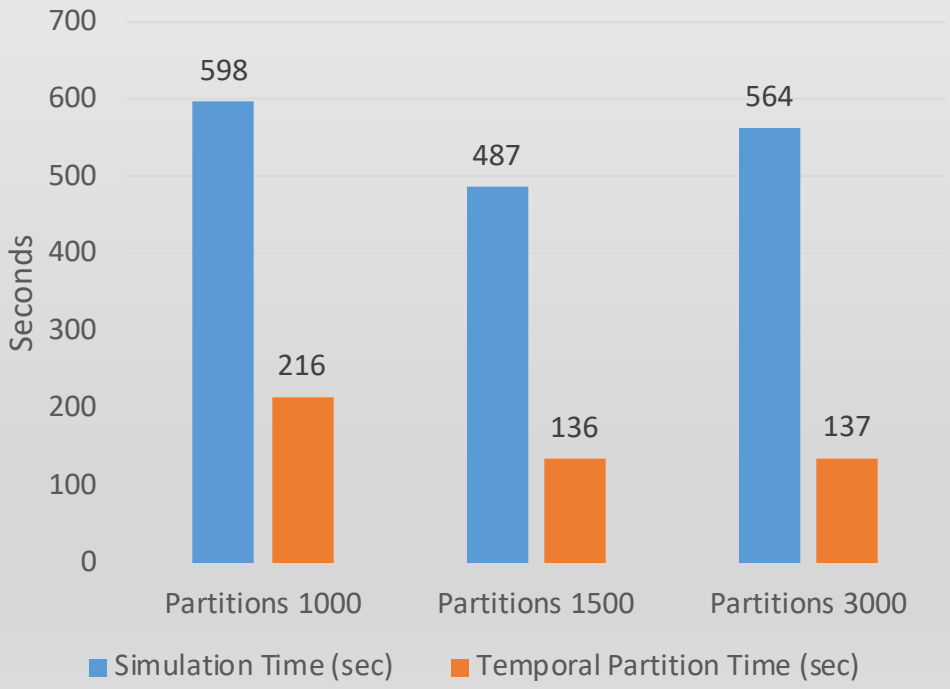
Time Step: **1 sec**

Simulation Period: **10 min**

Repartition Period: **2 min**

Number of Partition: **1000, 1500, 3000**

More parallelism and more data shuffle costs



ASU Arizona State University

Repartition Period – Temporal

Number of Vehicles: **100k**

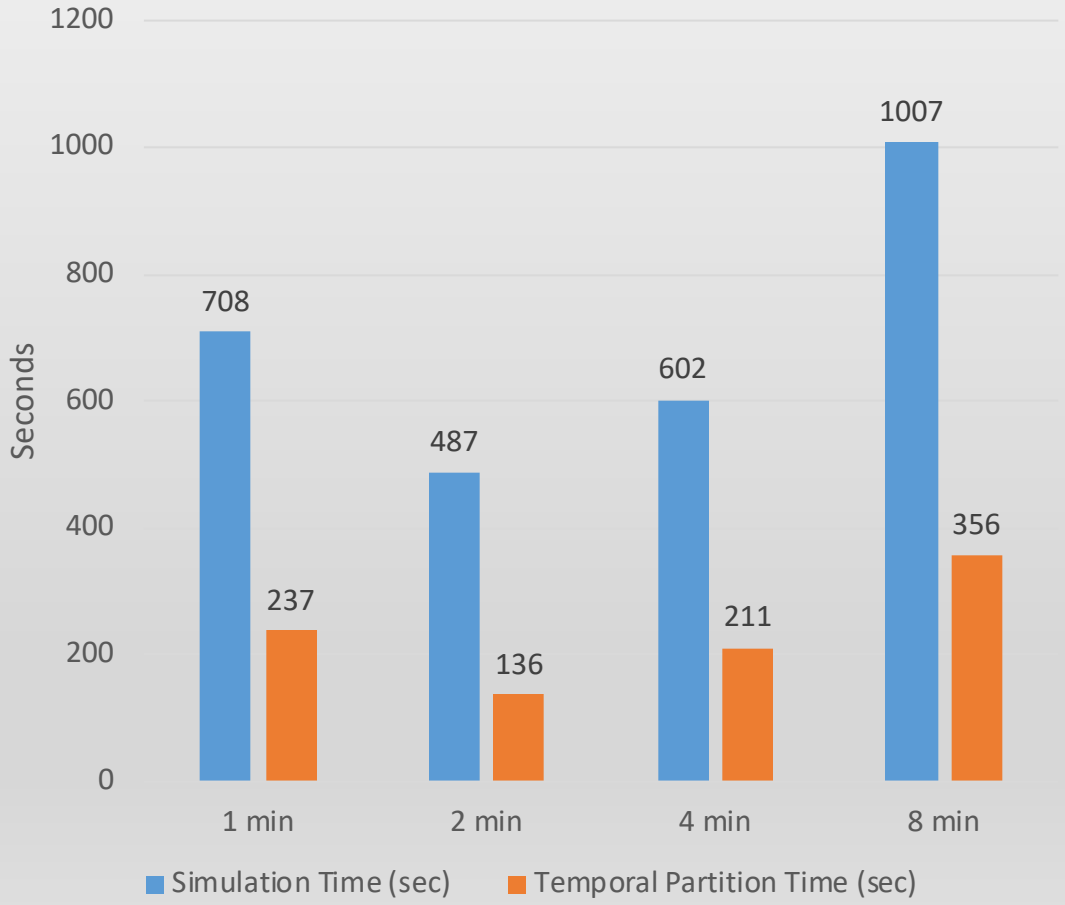
Time Step: **1 sec**

Simulation Period: **10 min**

Repartition Period: **1, 2, 4, 8**

Number of Partition : **1500**

Increases repartition times leads to longer partition time.
Decreases repartition times causes unbalanced workload



ASU Arizona State University

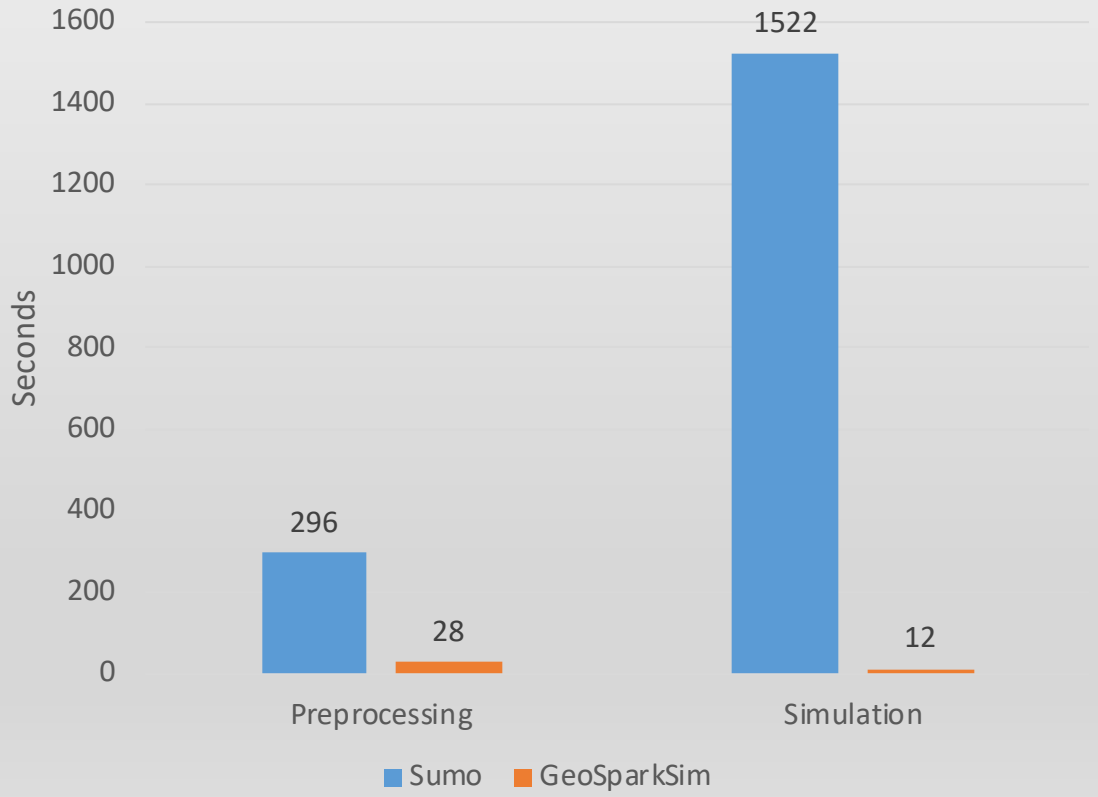
Sumo and GeoSparkSim

Number of Vehicles: **1000**

Time Step: **1 sec**

Simulation Period: **1 min**

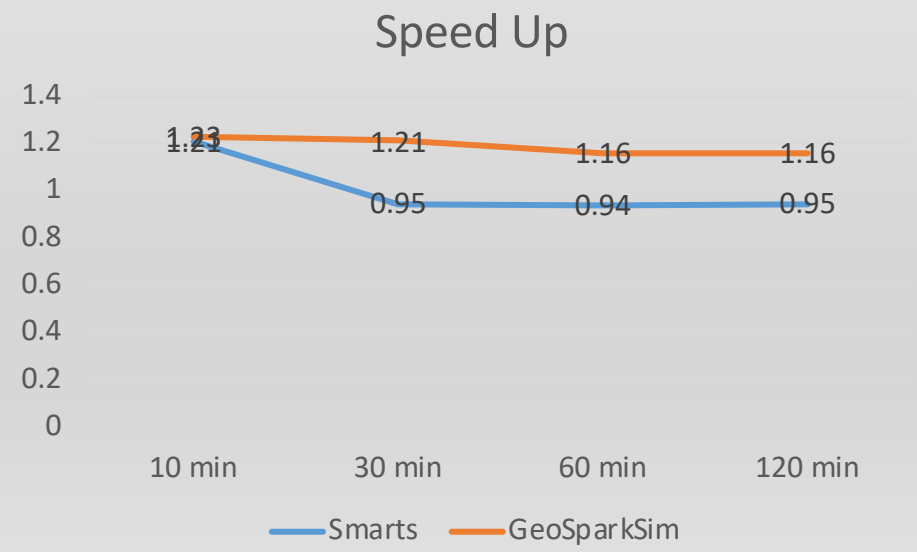
Scalability



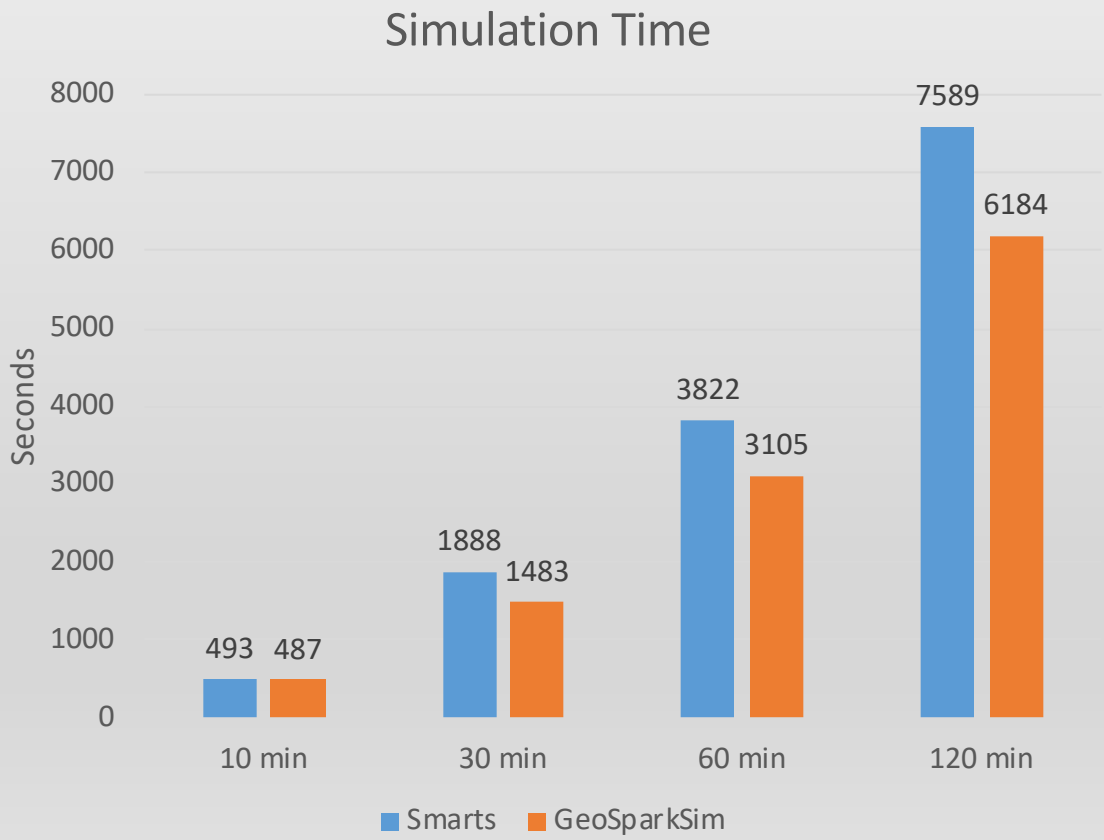
Smarts and GeoSparkSim

100k vehicles, **1 sec** time step,
Simulation Period: **10, 30, 60, 120 min**
Setting: **4** machines

$$Speedup = \frac{Simulation\ period}{Simulation\ time}$$



Workload balancing over time



ASU Arizona State University

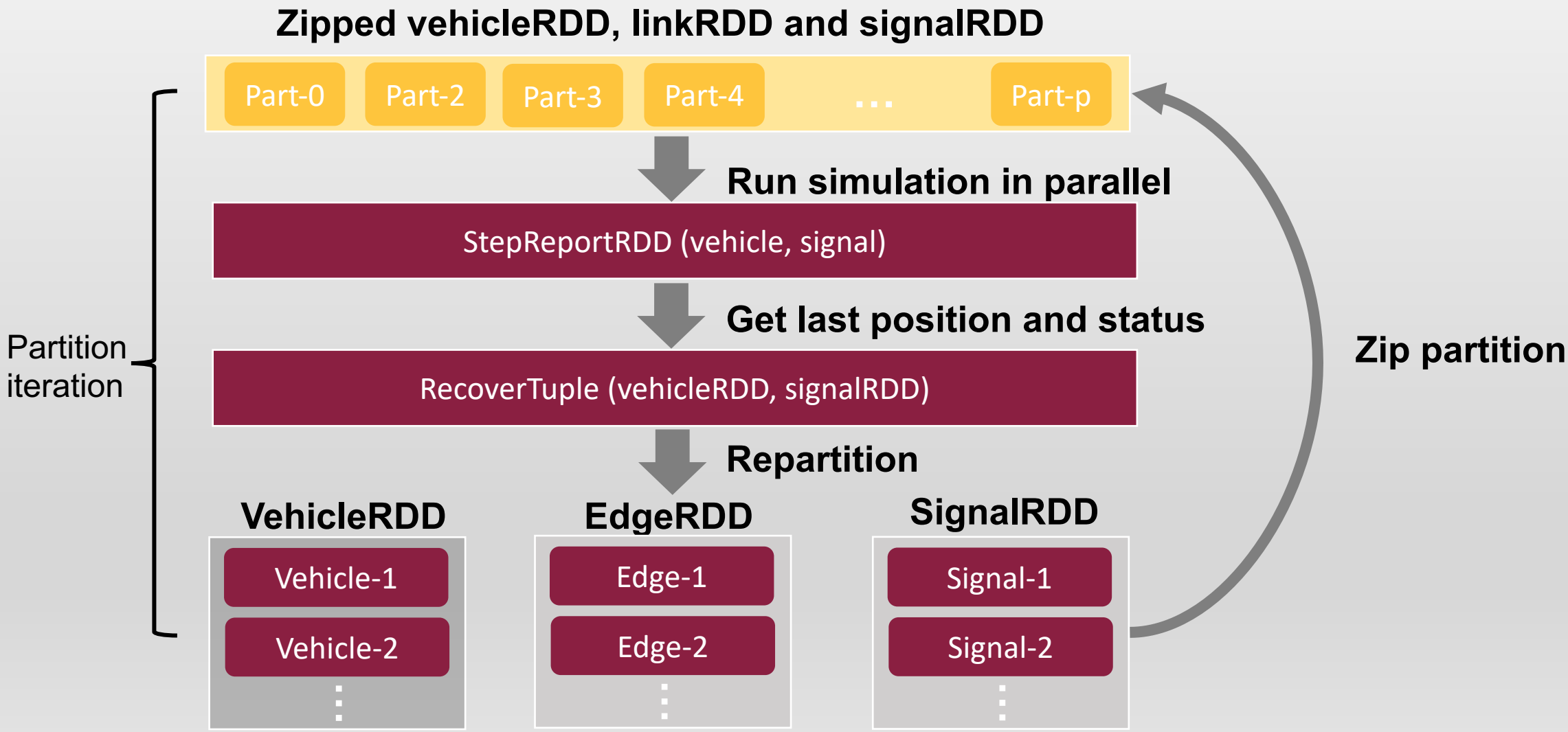
Conclusion

- Scalable – **100X** faster > single-machine solution, **20%** faster > multiple-machine solution
- Comprehensive – **Easy** to use
- Extensible – Driving model **abstraction**
- Large-scale data analysis – with other GeoSpark components

ASU Arizona State University

DEMO

Simulation-aware Partition



ASU Arizona State University

Parallel Microscopic Simulation

Input: TrafficLights, Link, Vehicle

1. **Initialize light and vehicle**
2. **Run simulation steps**

Step: N
Link: E
Vehicle: V
Signal: S
Driving: C

Time Complexity **Spatial Partition: P**

N(S+CV) → $\frac{N(S+VC)}{P}$

Parallel computing part i from 0 -> p

- Loop simulation steps — **N**
  - Update traffic lights — **NS**
  - Loop vehicles — **N(S+V)**
    - If vehicle not arrive
      - Vehicle head way check
      - Move
    - If arrive, reborn vehicle
- End, collect step reports

N(S+VC)

- Get closest vehicle and traffic light
- Update velocity (IDM)
- Remove vehicle from lane
- Compute position by velocity and update vehicle info
- Born vehicle
  - Check lane change (MOBIL)
  - Compute coordinate by position and lane
  - Update vehicle to map