Large-Scale Road Network Simulations for Smart Cities

Peter Heywood, Paul Richmond, Steve Maddock, Rob Chisholm, James Pyle
The University of Sheffield

Smart City Simulation

- Smart Cities
- Transport Simulation
- Computational Challenges

Smart Cities

- Data-driven transport management
- **Improve** utilisation and efficiency
- **Reduce** congestion and pollution

- 31% growth in UK motorway traffic by 2041 [1]

- High Congestion in Cities
  - Travel Speeds reduced by 58% in London [2]

Peak	15.6 mph
Free Flow	36.9 mph

- New modes of transport appearing
  - I.e Connected Autonomous Vehicles (CAVs)

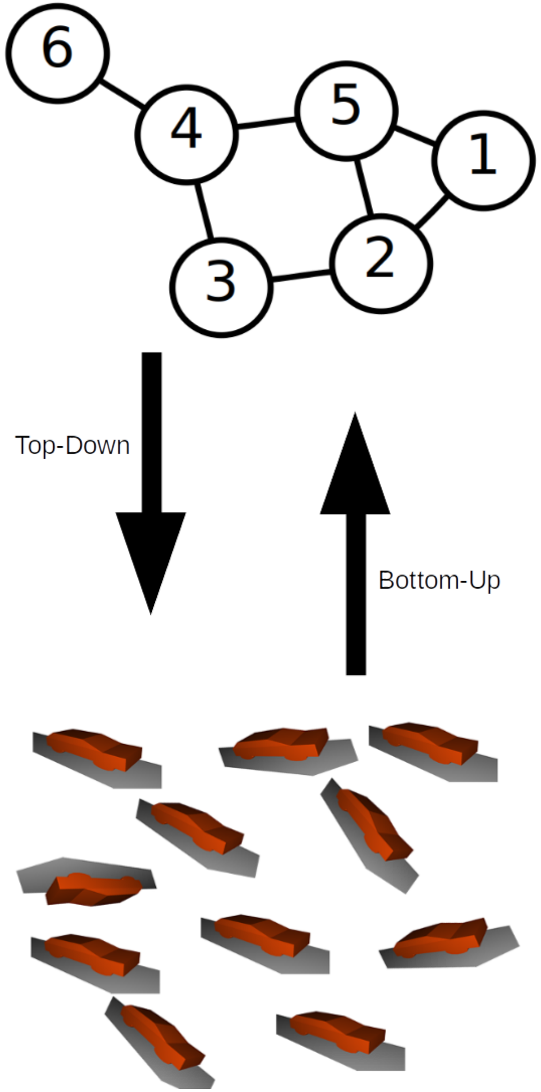
[1] Highways England Strategic Road Network Initial Report December 2017
[2] Inrix 2018 Traffic Scorecard for London

Smart City Transport Simulation

- Goals can be achieved through **simulation**
  - Planning
  - Management
- Cities are challenging
  - High population density
  - Co-located modes
- Limitations on possible interventions
  - Space
  - Air Quality
  - Money

Transport Network Simulation Resolution

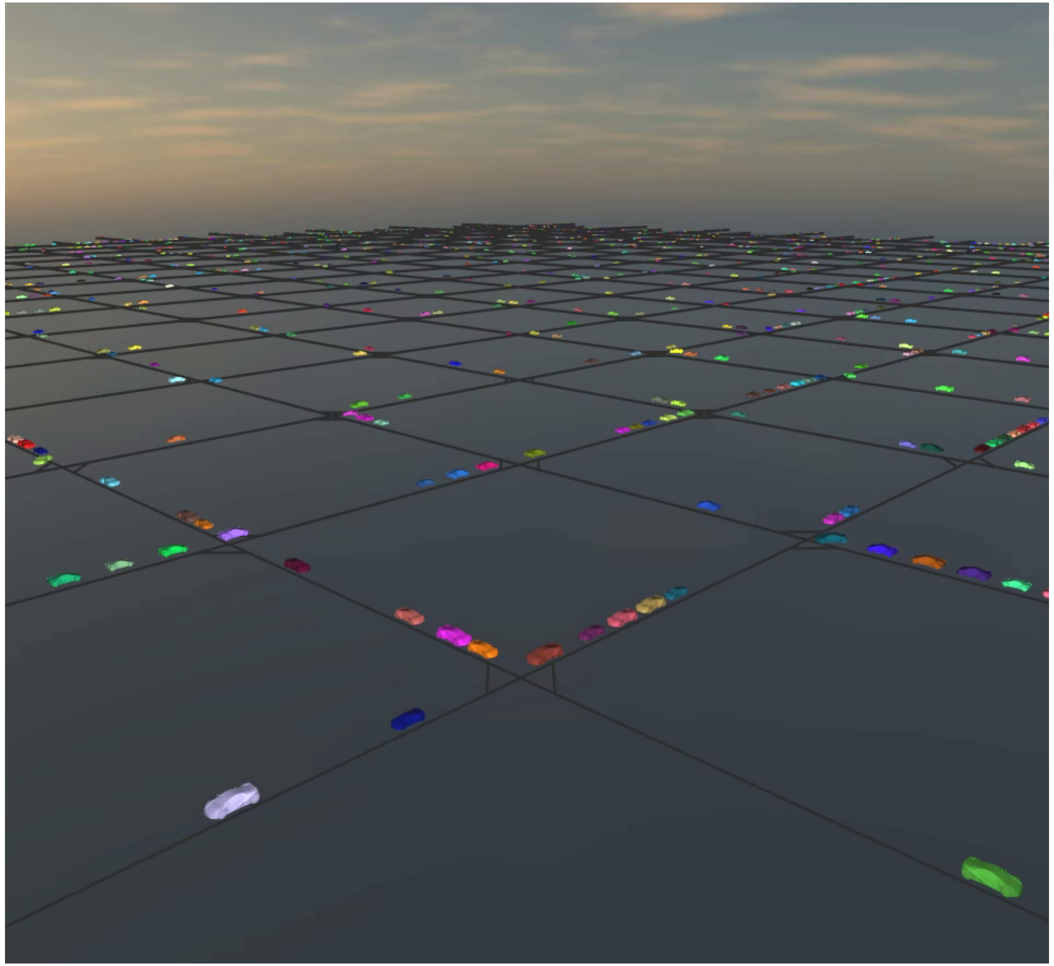
- **Macroscopic** (Top-Down)
  - Simulate aggregate flows across links
  - *Low Resolution*
  - *Lowest Computational Cost*
- **Mesoscopic**
  - Simulate platoons consisting of multiple vehicles
- **Microscopic** (Bottom-up)
  - Simulate individual vehicles or people
  - *High Resolution*
  - *Very High Cost*



Agent Based Modelling (ABM)

- An approach for Microsimulation
- Individuals (agents) with properties
- Describe simple rule-based behaviours
- Interactions with others agents and environment
- Complex behaviours emerge

- **Huge computational cost**
- Large volumes of data required
- Many simulations required

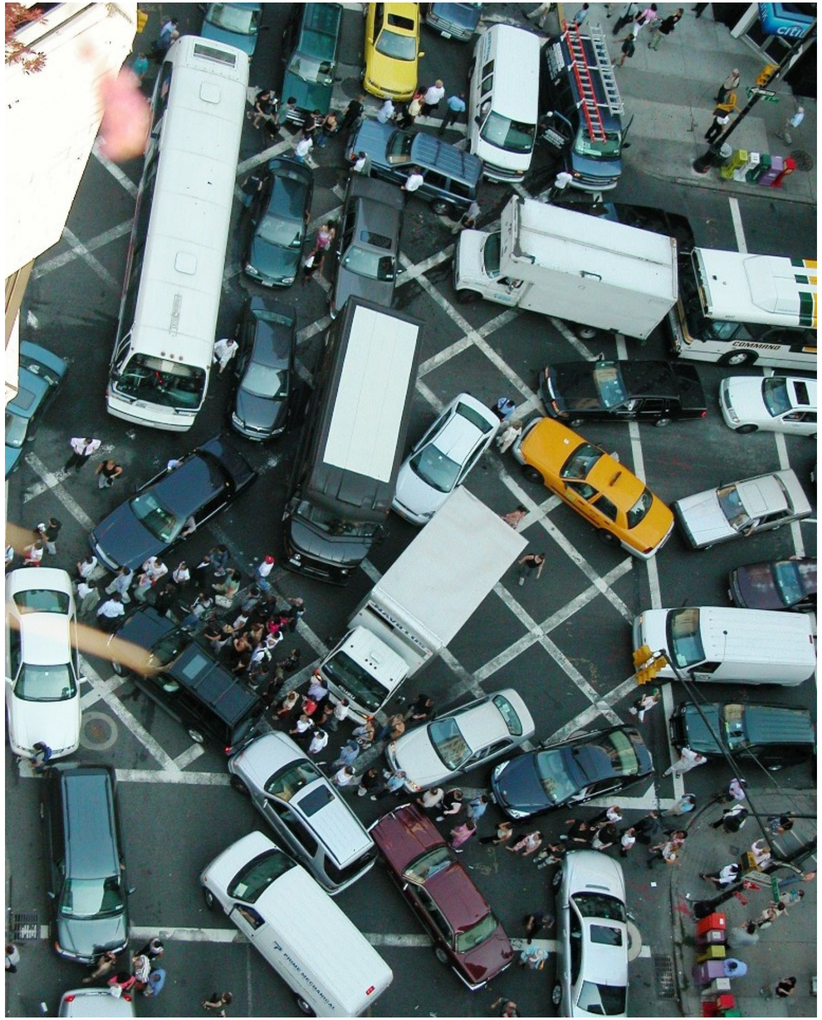


FLAME GPU Road Network Microscopic Simulation

Computational Challenges

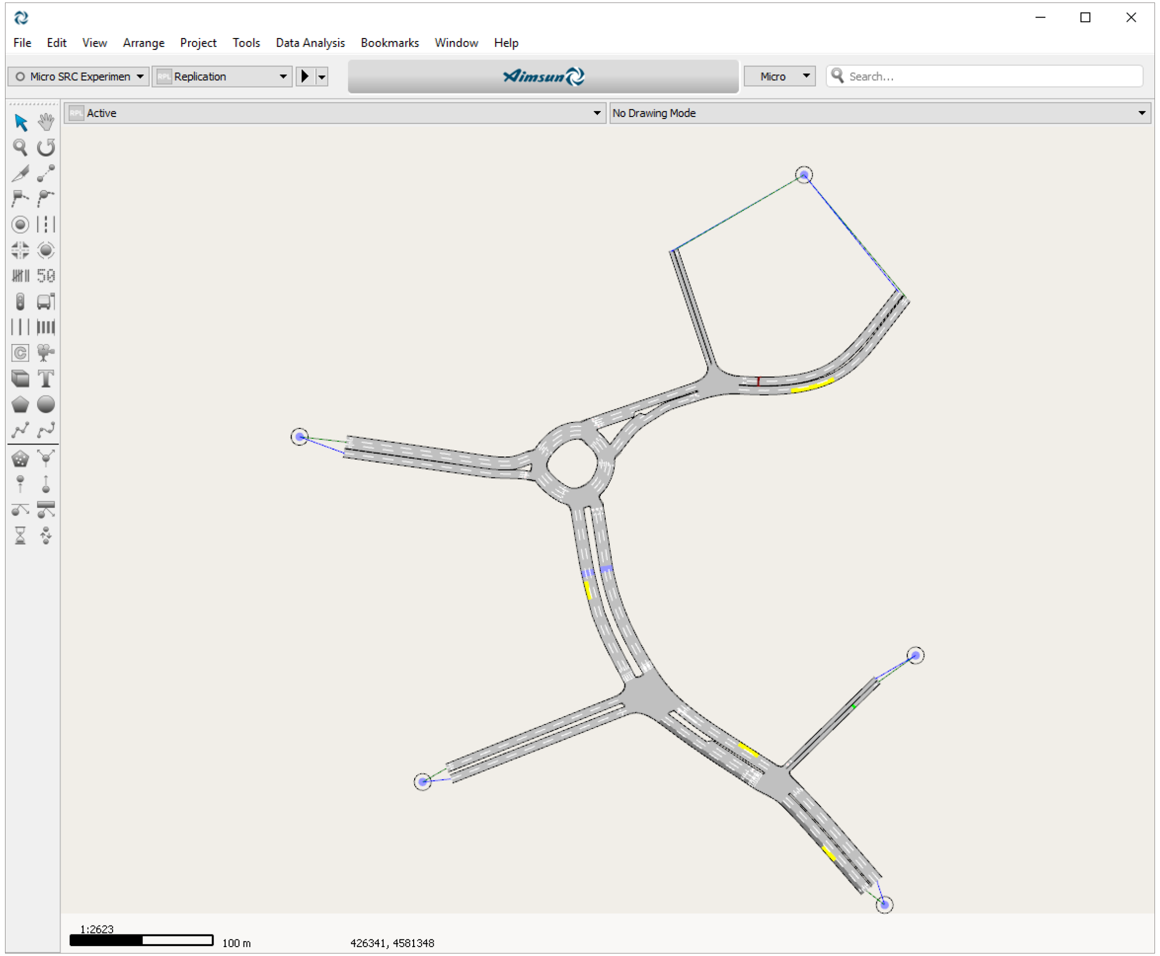
- Smart city simulations are **massively** computationally expensive
  - *Millions* of individuals
  - Many modes of transport
  - Many permutations required
    - Weather, Demand, etc.
- Performance is limiting the use of simulation in industry [1]
- **Faster simulators are required**

[1] Brackstone, M., & Punzo, V. (2014). Traffic Simulation: Case for guidelines. European Commission, Joint Research Centre, Luxembourg, 100.



Commercial Microsimulation tools

- Many commercial and open-source simulators. I.e.
  - Aimsun
  - PTV Vissim
  - Parasim
  - SUMO
- All CPU-based simulators
  - Single-threaded or Multi-threaded
- Poor scaling
  - with additional processor cores
  - with problem size



Aimsun 8.1 Microsimulation User Interface

Our Aims

Our Aims

- Enable GPU Accelerated Smart City Simulations
  - Suitable for city-scale networks
  - Better-than-real-time performance

1. Implement subset of models from a commercial tool
2. Cross-validate GPU implementation
3. Benchmark using a scalable model

Nvidia Titan Xp and Titan V GPUs

Microsimulation Benchmark

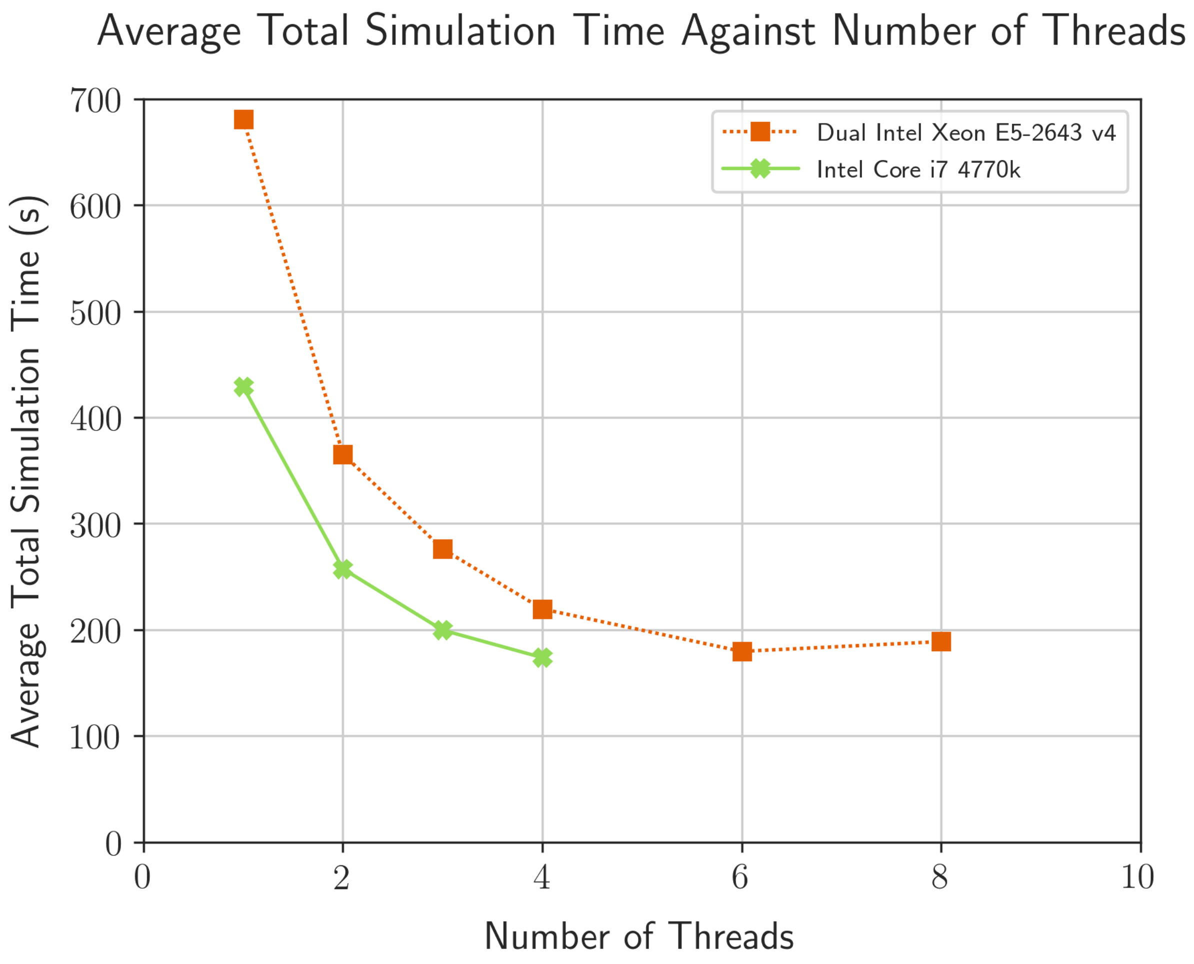
- Reference Simulator
- Scalable Network
- CPU Benchmarking

Reference CPU Simulator

- Simulator to re-implement and compare
- **Aimsun 8.1**
  - Multi-core CPU simulator
  - Used globally
  - Suitable for a wide range of transport modelling tasks
  - Diminishing returns over 8 cores

aimsun.

www.aimsun.com



Aimsun Performance using different numbers of CPU cores

Benchmark Microsimulation Models

- Gipps' Car Following Model
- Aimsun Gap Acceptance Model
- Probability-based routing
- Constant Vehicle Arrival
- Stop-Sign Yellow-box junctions
- Simulated Detectors

Gipps' Car Following Model

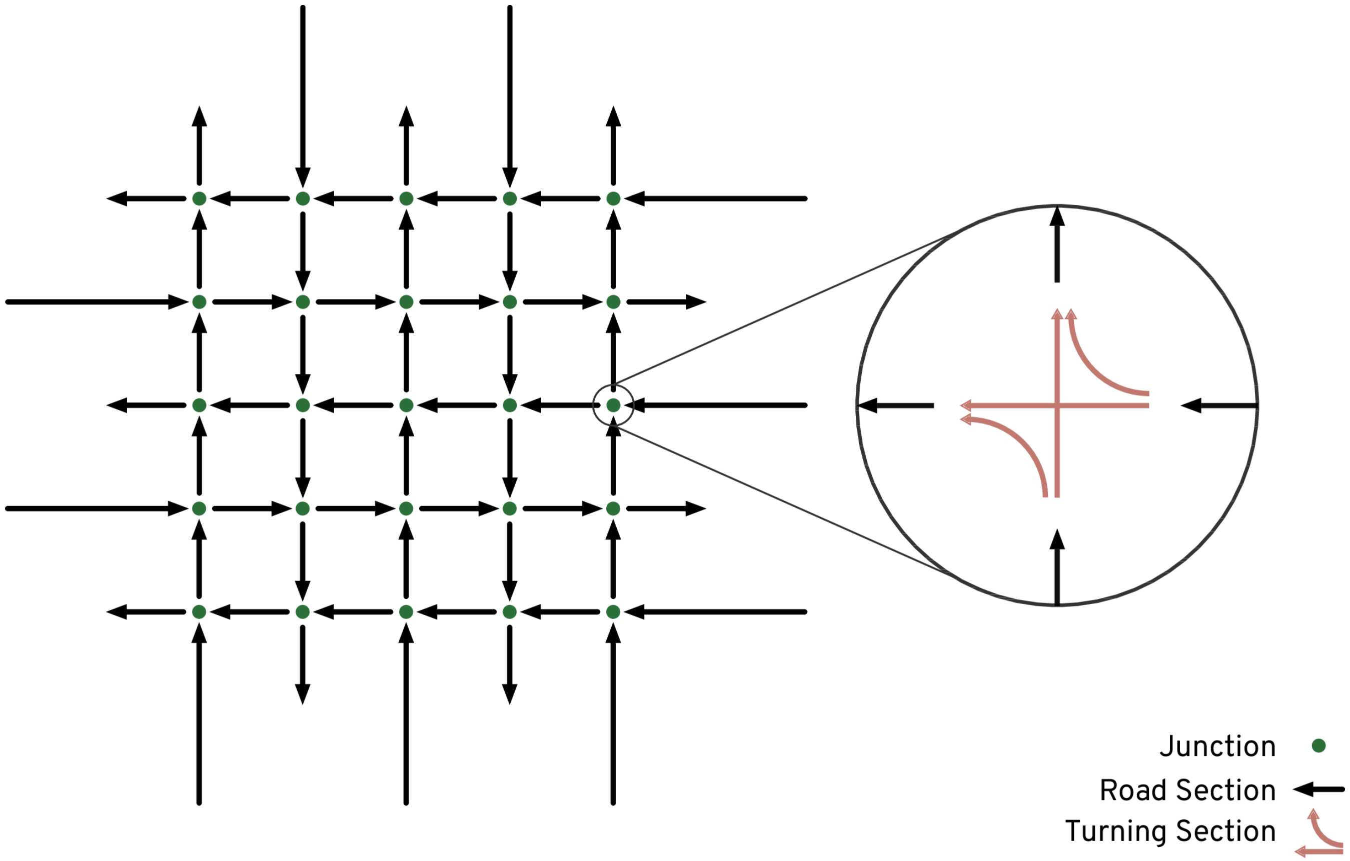
$$v_{free}(n, t+\tau) \leq v(n, t) + 2.5a(n)\tau(1 - v(n, t)/V(n))(0.025 + v(n, t)/V_t(n)^{\frac{1}{2}}$$

$$v_{safe}(n, t+\tau) \leq d(n)\tau + \sqrt{d(n)^2\tau^2 - d(n)(2[x(n-1, t) - s(n-1) - x(n, t)] - v(n, t)\tau - \frac{v(n-1, t)^2}{\hat{d}(n)})}$$

$$v(n, t+\tau) = \min\left\{v_{free}(n, t+\tau), v_{safe}(n, t+\tau)\right\}$$

Benchmark Network

- Manhattan-style grid network
- Single lane, one-way roads
- Stop-signs at junctions
- Entrances and exits at the edges of the simulation



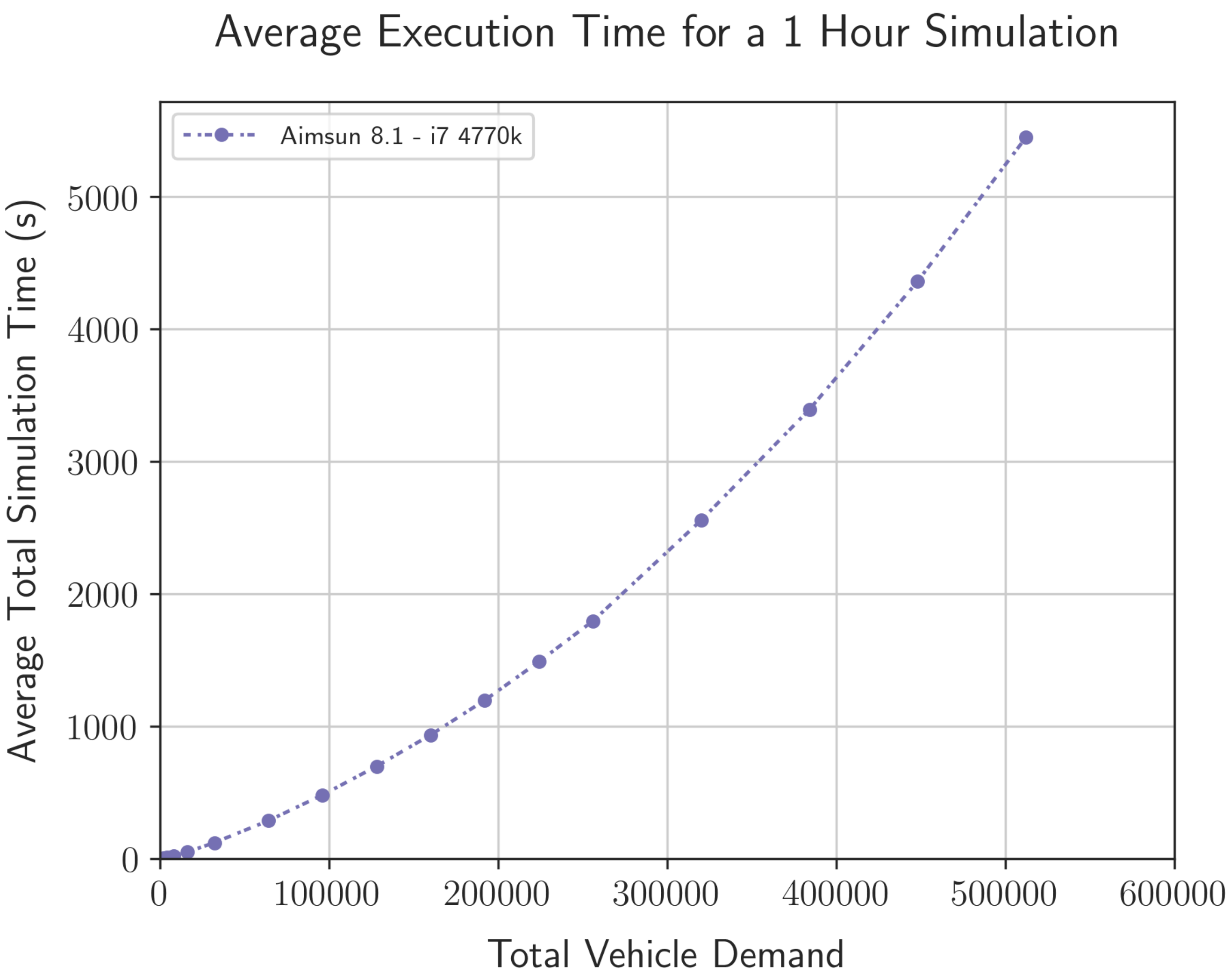
Aimsun CPU Benchmarking

- Aimsun 8.1
- 4 core Intel i7 4770k CPU

- 3 repetitions
- 1 hour simulation
- Varied environment and population sizes

- Largest Simulation
  - ~ 500,000 vehicles
  - ~ 2,000,000 detectors
  - 5447 seconds
  - **1.5x slower than real time**

- **Too slow for real-time management**



GPU Accelerated Microsimulation

- FLAME GPU
- Implementation Details
- Cross Validation

FLAME GPU

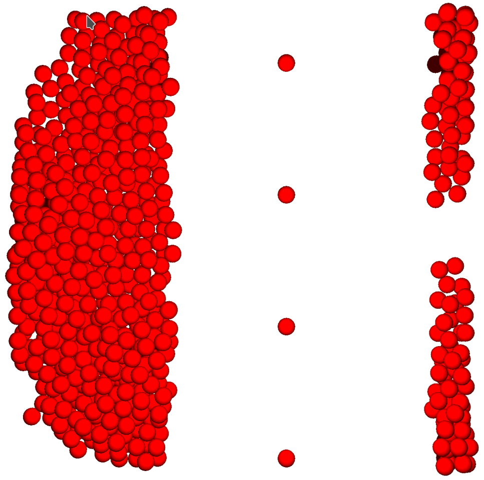
Flexible **L**arge-Scale **A**gent **M**odelling **E**nvironment for the **GPU**

- Template-based simulation environment
- High performance agent-based simulation
- Agents modelled using X-Machines
  - with *message lists* for communication
- Abstracts the CUDA programming model away from the user

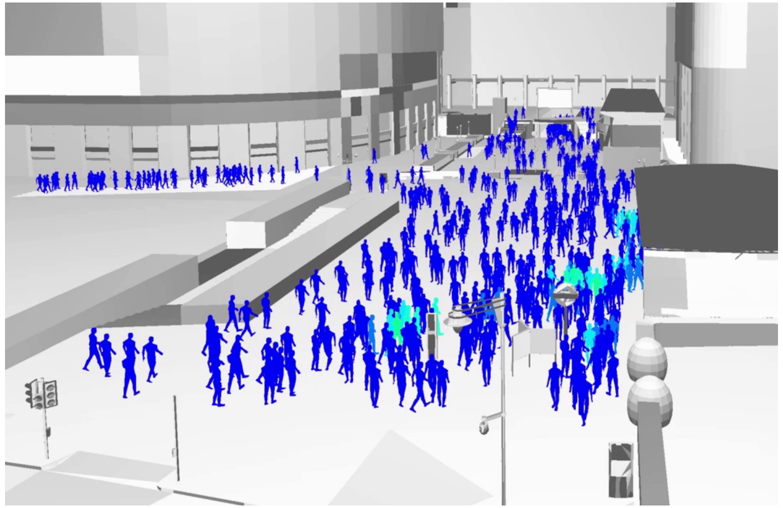
FLAME GPU

github.com/flamegpu/

flamegpu.com

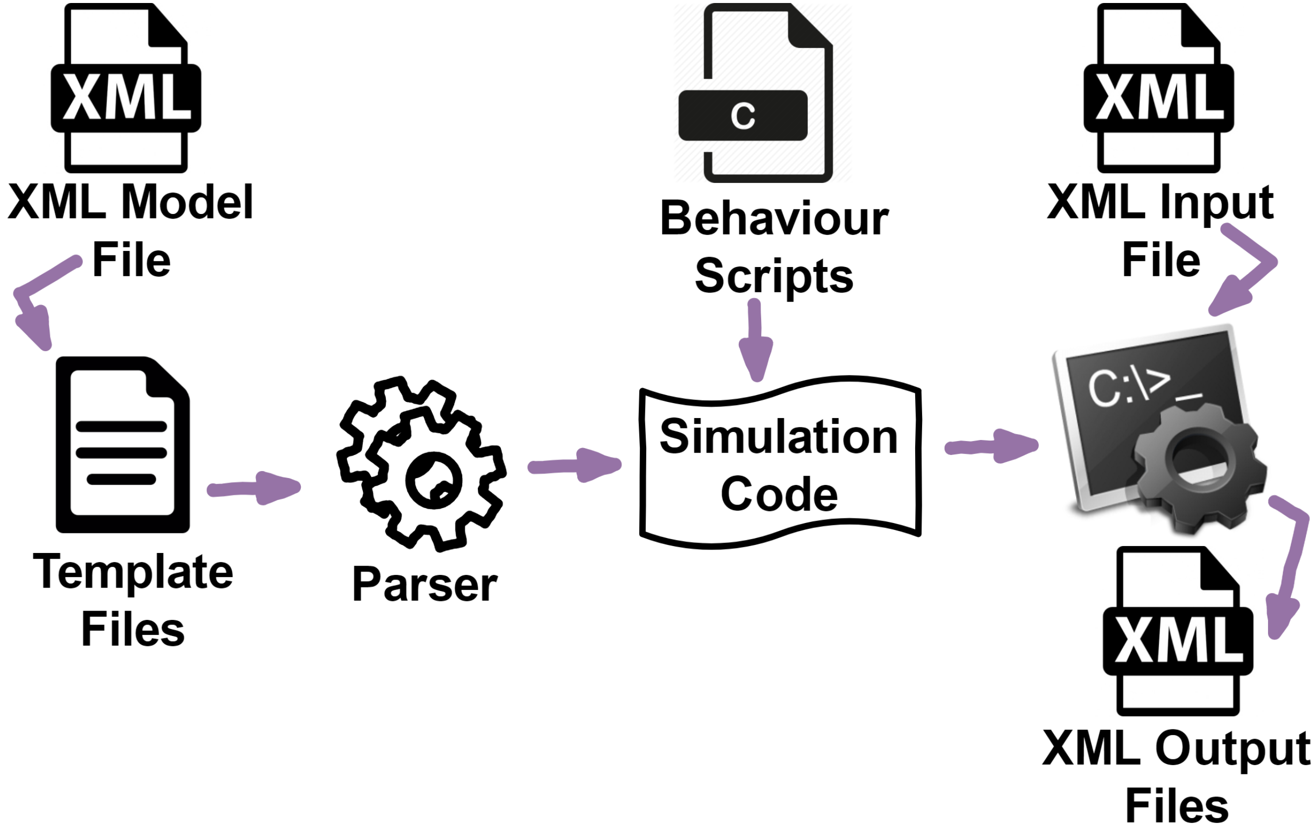


Keratinocyte Cell Simulation



Pedestrian Simulation

FLAME GPU Code Generation Process



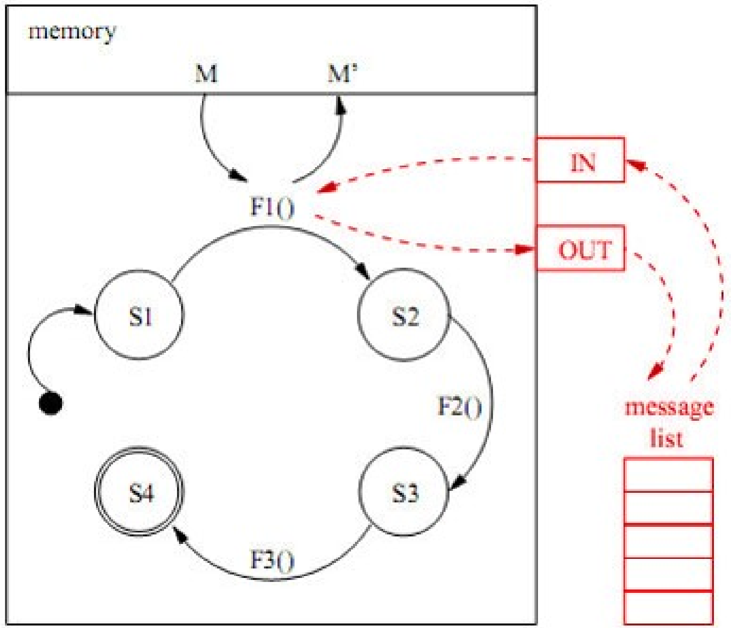

```xml
<gpu:function>
  <name>outputdata</name>
  <currentState>default</currentState>
  <nextState>default</nextState>
  <outputs>
    <gpu:output>
      <messageName>location</messageName>
      <gpu:type>single_message</gpu:type>
    </gpu:output>
  </outputs>
  <gpu:reallocate>false</gpu:reallocate>
  <gpu:RNG>false</gpu:RNG>
</gpu:function>
```

```c
__FLAME_GPU_FUNC__ int outputdata(
    xmachine_memory_Boid* xmemory,
    xmachine_message_location_list* location_messages
    )
{
    add_location_message(location_messages, xmemory->
        id, xmemory->x, xmemory->y, xmemory->z,
        xmemory->fx, xmemory->fy, xmemory->fz);

    return 0;
}
```

Why use FLAME GPU?

- GPU knowledge not required
- Divergence minimised by state-based representation
- Efficient memory access
  - SoA, neighbouring threads
  - Shared memory used where appropriate, authomatically
- Race conditions avoided
  - Message-lists
  - Natural synchronisation barriers



Benchmark Microsimulation Models

- Gipps' Car Following Model
- Aimsun Gap Acceptance Model
- Probability-based routing
- Constant Vehicle Arrival
- Stop-Sign Yellow-box junctions
- Simulated Detectors

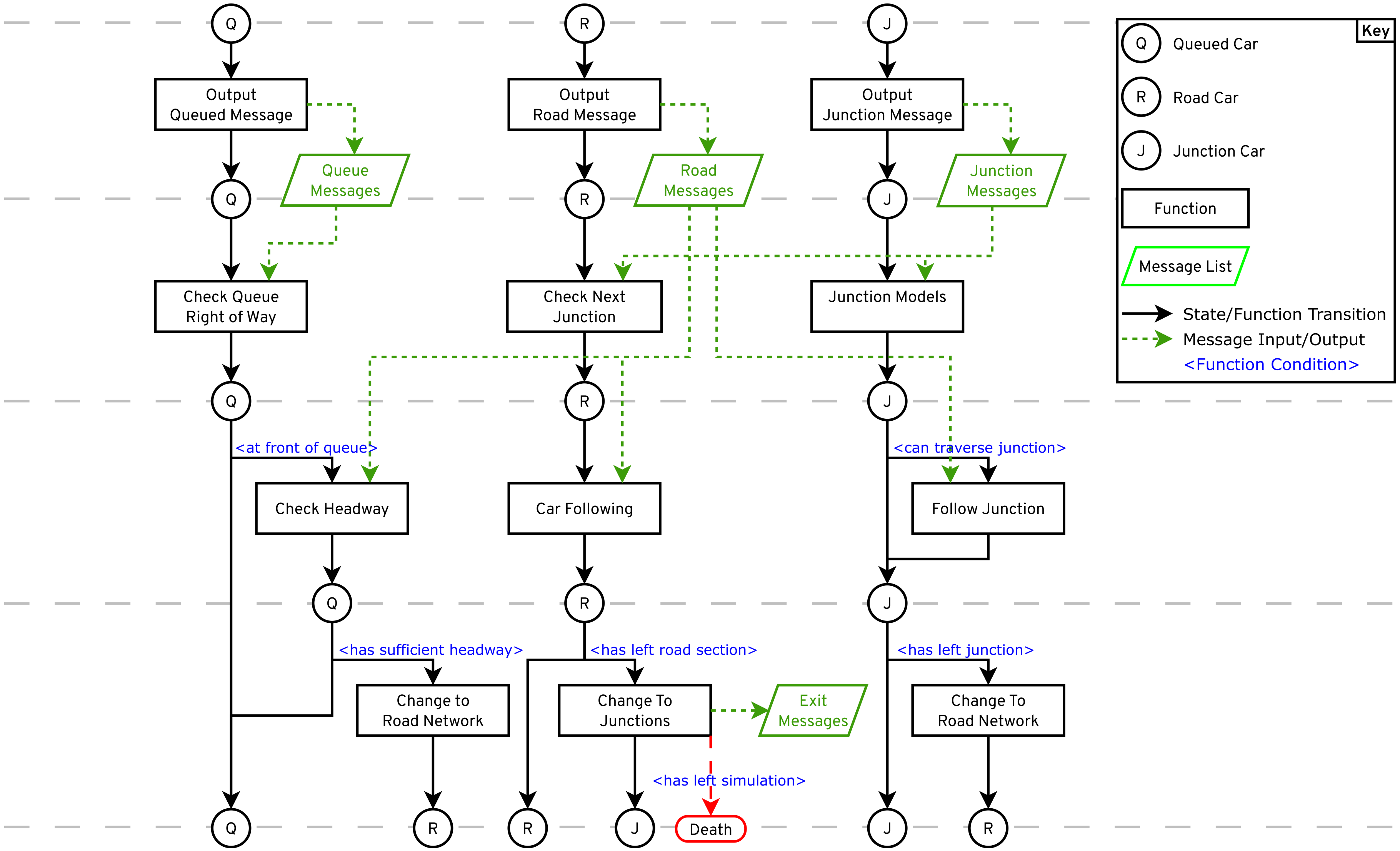
Gipps' Car Following Model

$$v_{free}(n, t+\tau) \leq v(n, t) + 2.5a(n)\tau(1 - v(n, t)/V(n))(0.025 + v(n, t)/V_t(n)^{\frac{1}{2}}$$

$$v_{safe}(n, t+\tau) \leq d(n)\tau + \sqrt{d(n)^2\tau^2 - d(n)(2[x(n-1, t) - s(n-1) - x(n, t)] - v(n, t)\tau - \frac{v(n-1, t)^2}{\hat{d}(n)})}$$

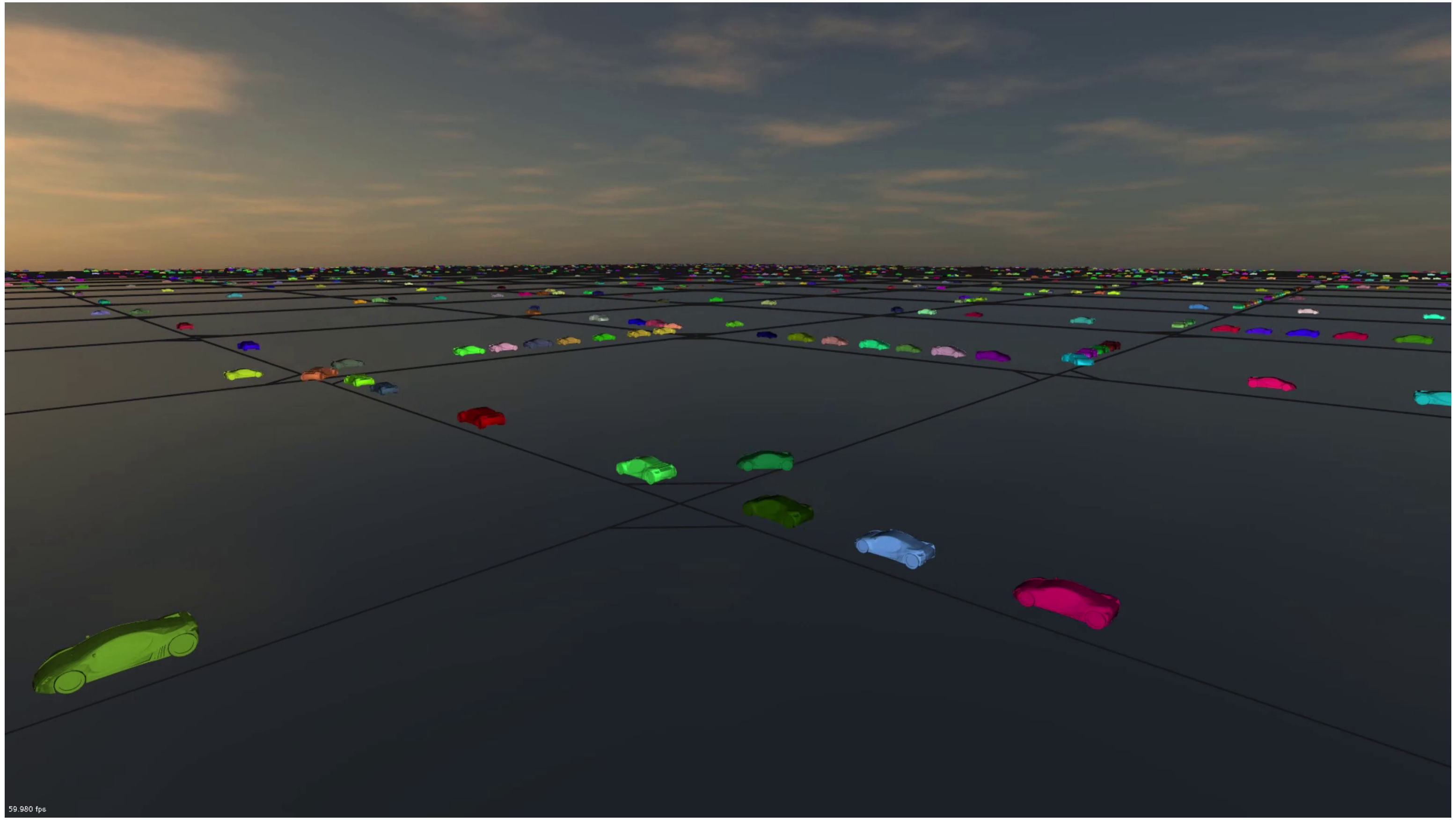
$$v(n, t+\tau) = \min\left\{v_{free}(n, t+\tau), v_{safe}(n, t+\tau)\right\}$$

Implementation State Diagram



Validation of GPU Implementation

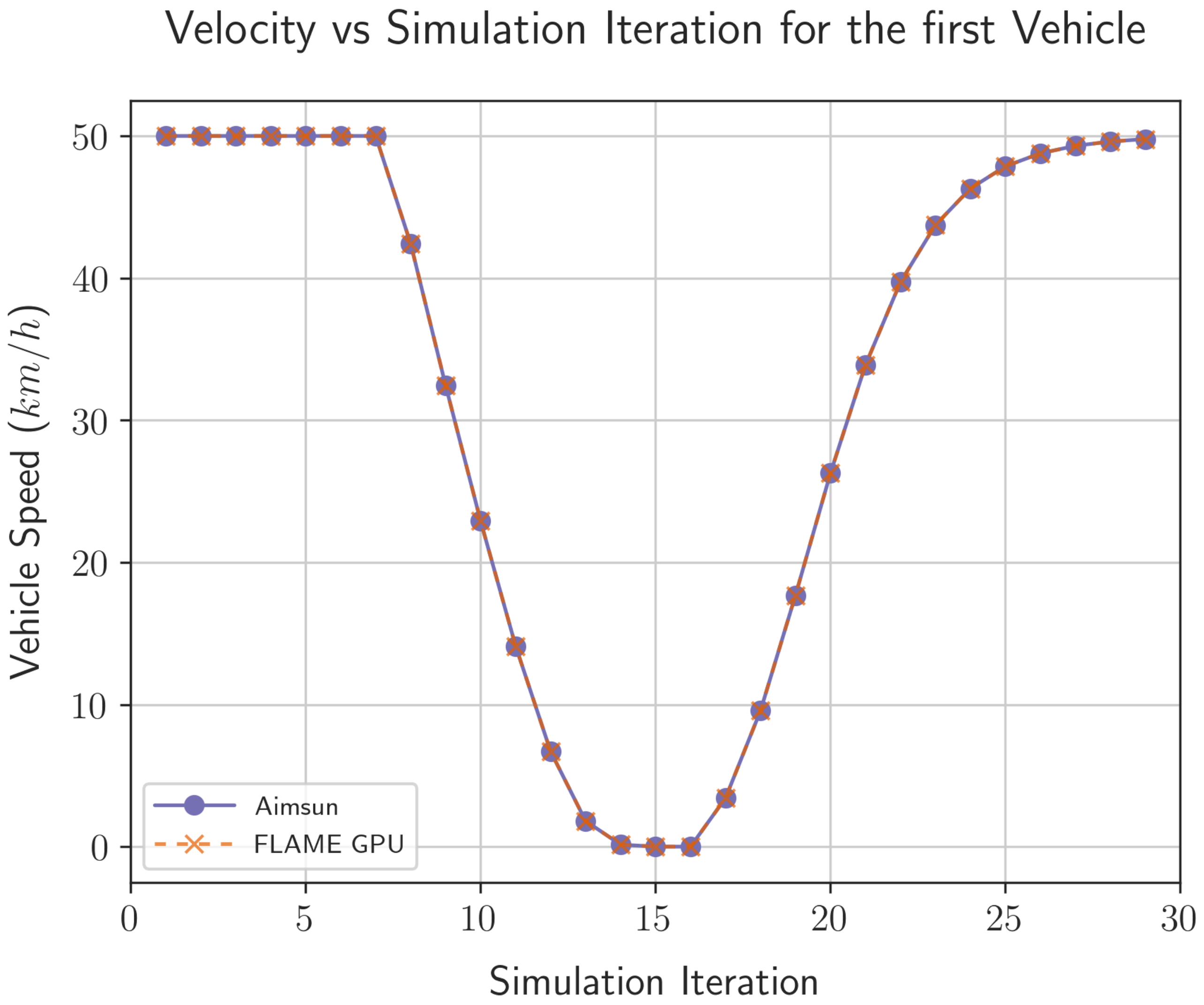
- Cross validated vs Aimsun 8.1
- 6 sets of validation networks
- Individual behaviours
- Combined Effects



Validation of GPU Implementation

- Cross validated vs Aimsun 8.1
- 6 sets of validation networks
- Individual behaviours
- Combined Effects

- Deterministic tests reproduced exactly
- Stochastic test reproduced within acceptable range



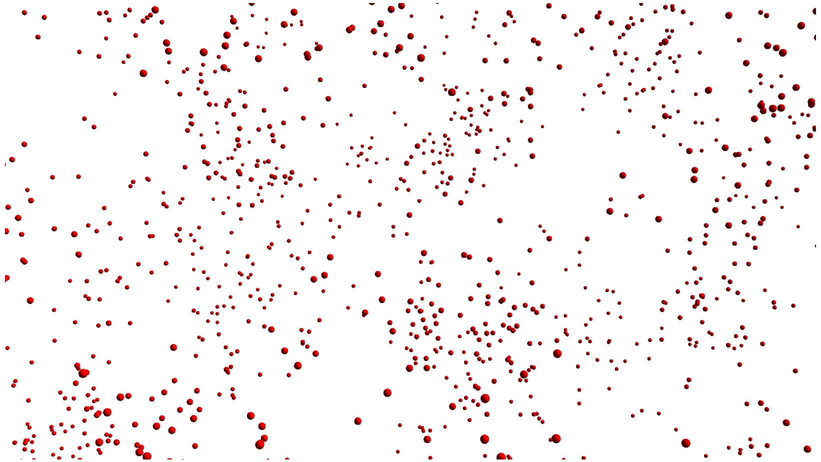
Agent Communication

- Existing Communication Strategies
- Graph Based Communication
- Benchmark Results

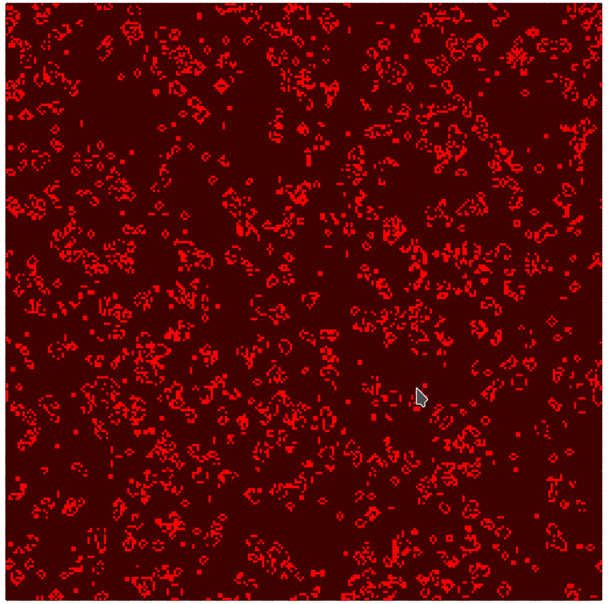
FLAME GPU Communication

- Message lists used to communicate
  - Avoids Race Conditions
  - Good cache utilisation
  - Memory hierarchy optimisations
- Message iteration often limits performance
- Specialise communication pattern for efficiency

- Communication Patterns in FLAME GPU 1.4
  - All-to-All
  - Spatially Partitioned Messaging
  - Discrete Partitioned Messaging
- Non-optimal for road network models



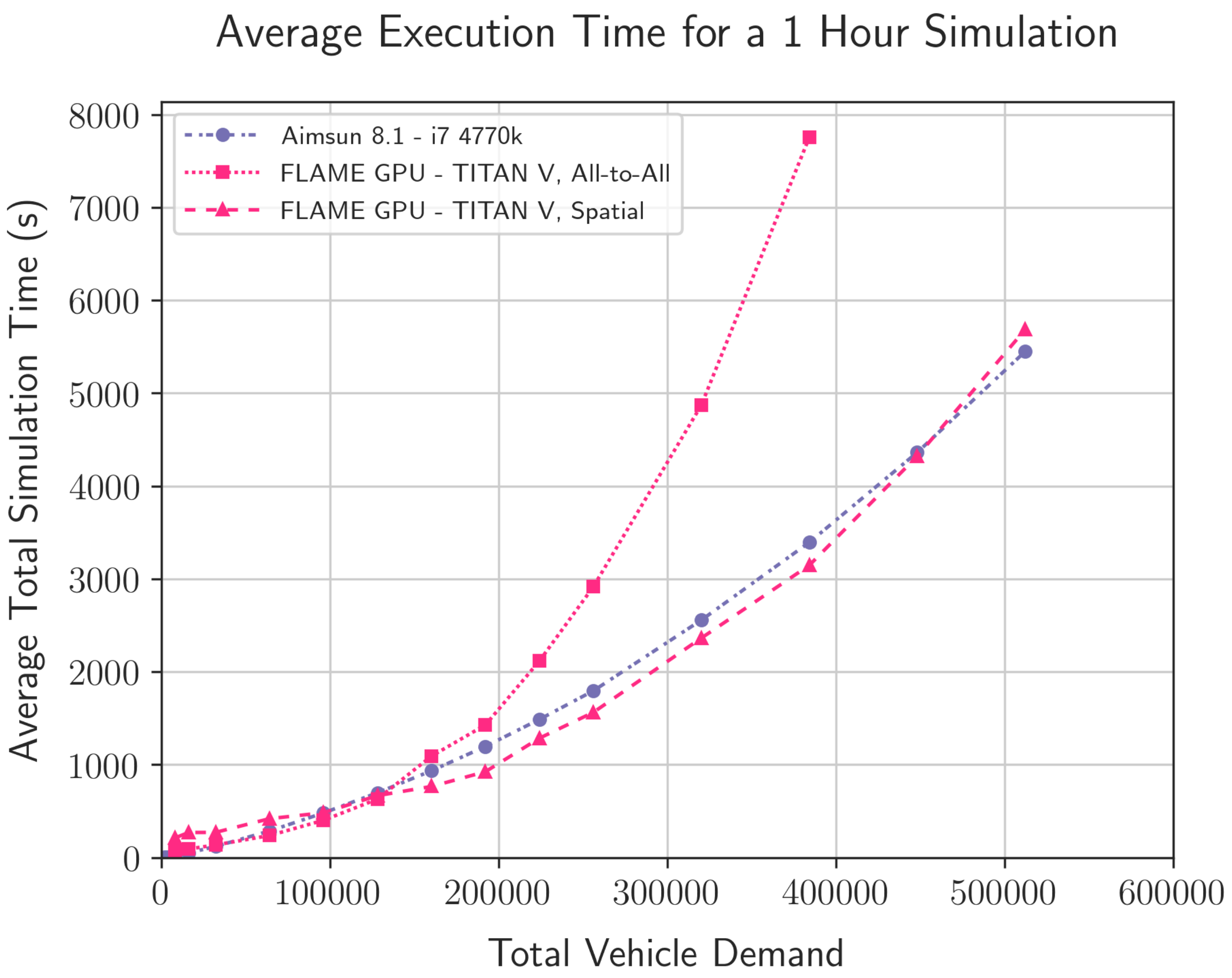
Boids Flocking Model - Spatial Partitioning



Game of Life - Discrete Partitioning

FLAME GPU Communication Benchmarking

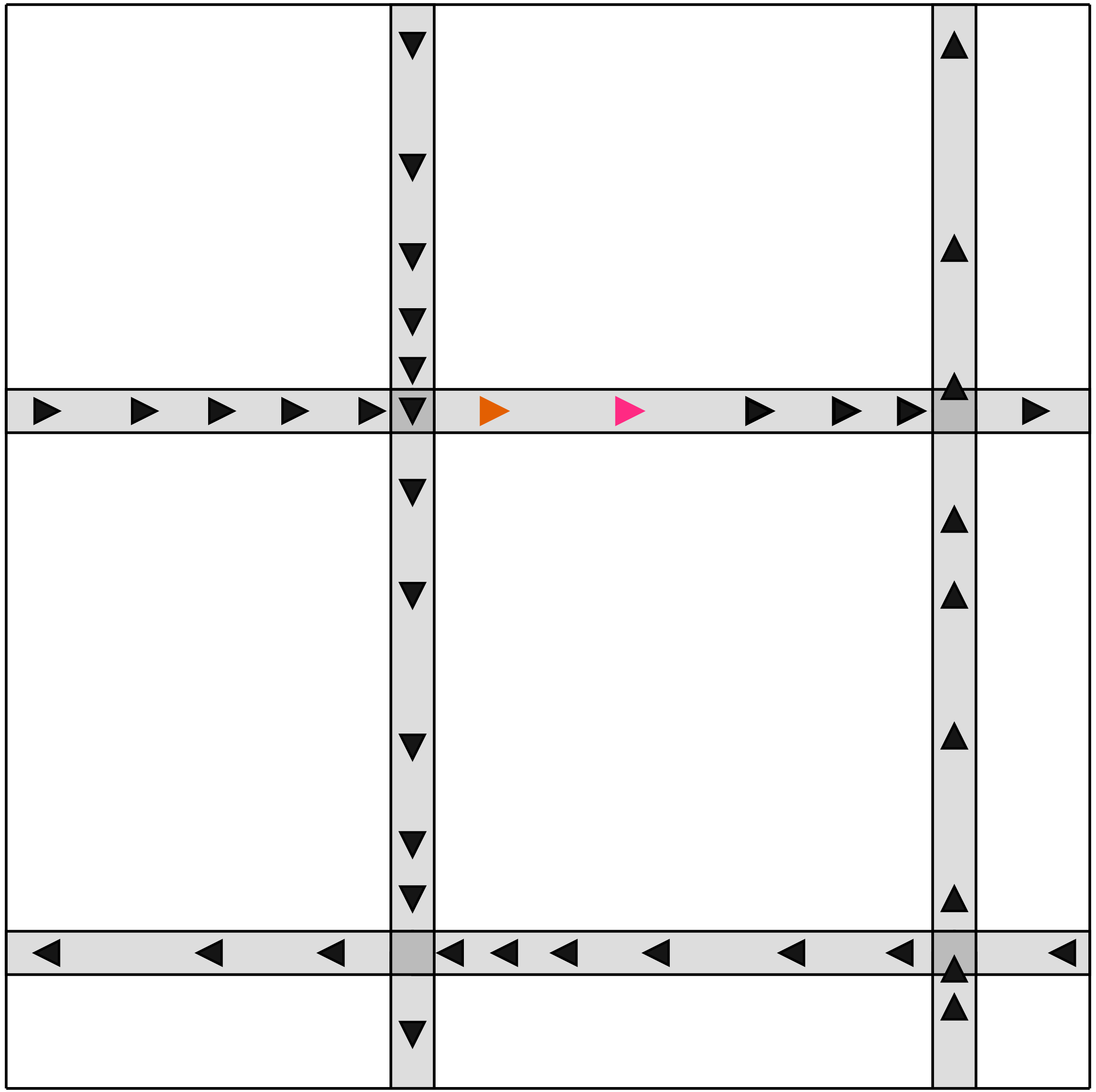
- Benchmarked existing communication strategies
- 1 hour simulation
- 3 repetitions
- Titan V
- Poor Performance
- **Majority of runtime spent iterating messages!**



Communication Example

Gipps' Car Following Model

- Agent only requires information from the lead vehicle to calculate new speed
- I.e. ▶ requires information from ▶

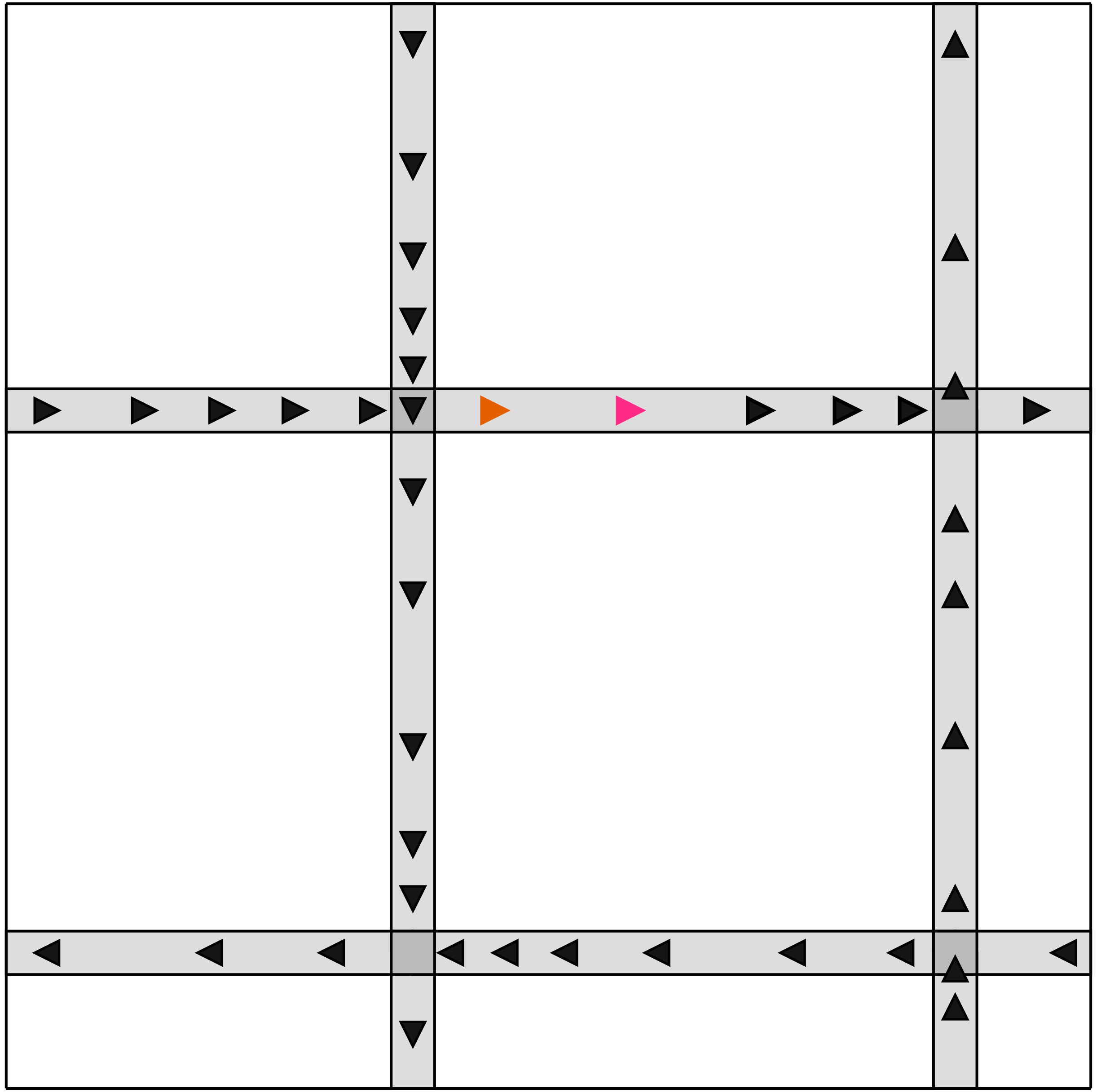


Communication Example

All-to-All Communication

- Each agent reads every message
- Agent ▶ reads **42** messages
  - From ▶ ▶ ▶

Communication Strategy	# Messages
All-to-all	42



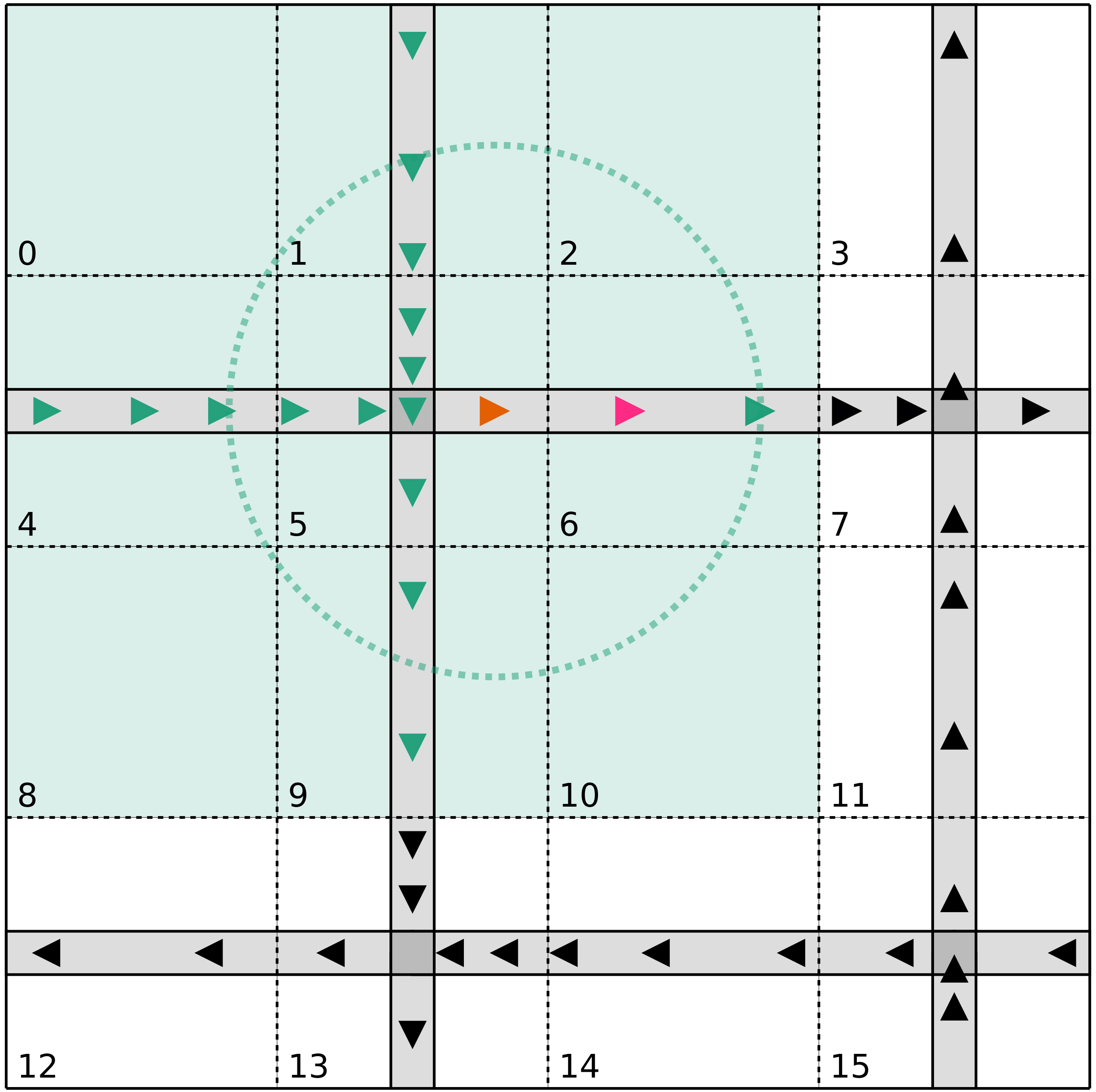
Communication Example

Spatially Partitioned

- Radius-based
- Partition the environment
- Read from Moore's Neighbourhood

- Agent ▶ reads **18** messages
  - From ▶ ▶ ▶

Communication Strategy	# Messages
All-to-all	42
Spatial	18

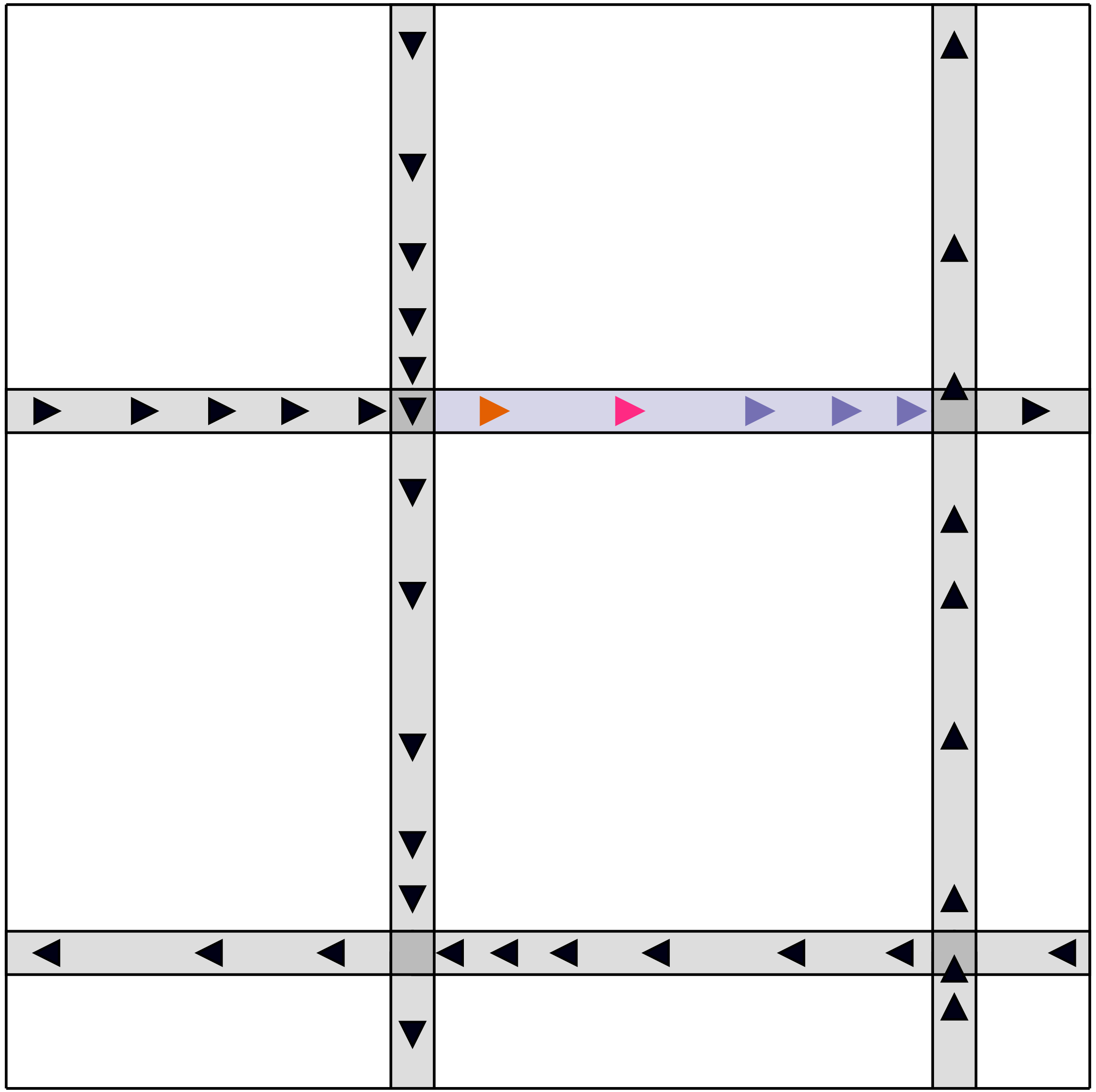


Communication Example

Graph Based

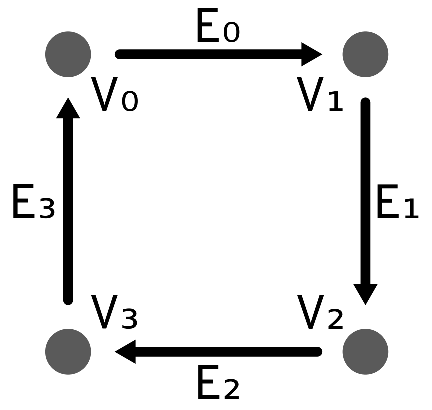
- Couple messages to graph data structure
- Read from relevant part of graph
- Agent ▶ reads **5** messages
  - From ▶ ▶ ▶

Communication Strategy	# Messages
All-to-all	42
Spatial	18
Graph	5



Graph Based Communication Implementation

- Compressed Sparse Row (CSR) representation of graph
- Messages contain *edge* or *vertex* index
- Sort message list based on index
  - *Counting Sort*
  - Shared-memory atomics
  - Builds histogram to access messages
- Can access a single edge, or explore using CSR
- Neighbouring threads access same messages
- Available from FLAME GPU 1.5.0

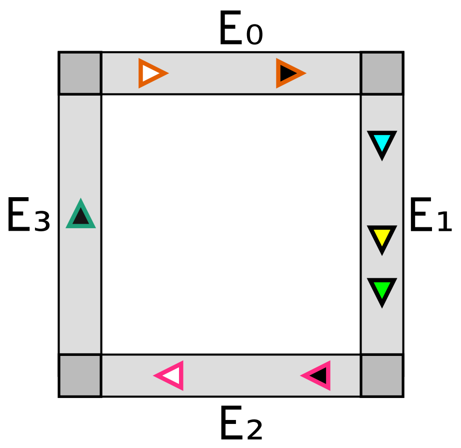


Compressed Sparse Row

$A = [E_0 E_1 E_2 E_3]$

$IA = [0\ 1\ 2\ 3\ 4]$

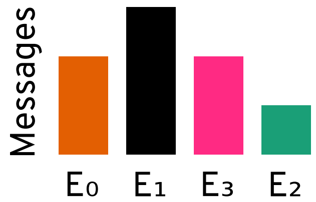
$JA = [1\ 2\ 3\ 0]$



Output: 

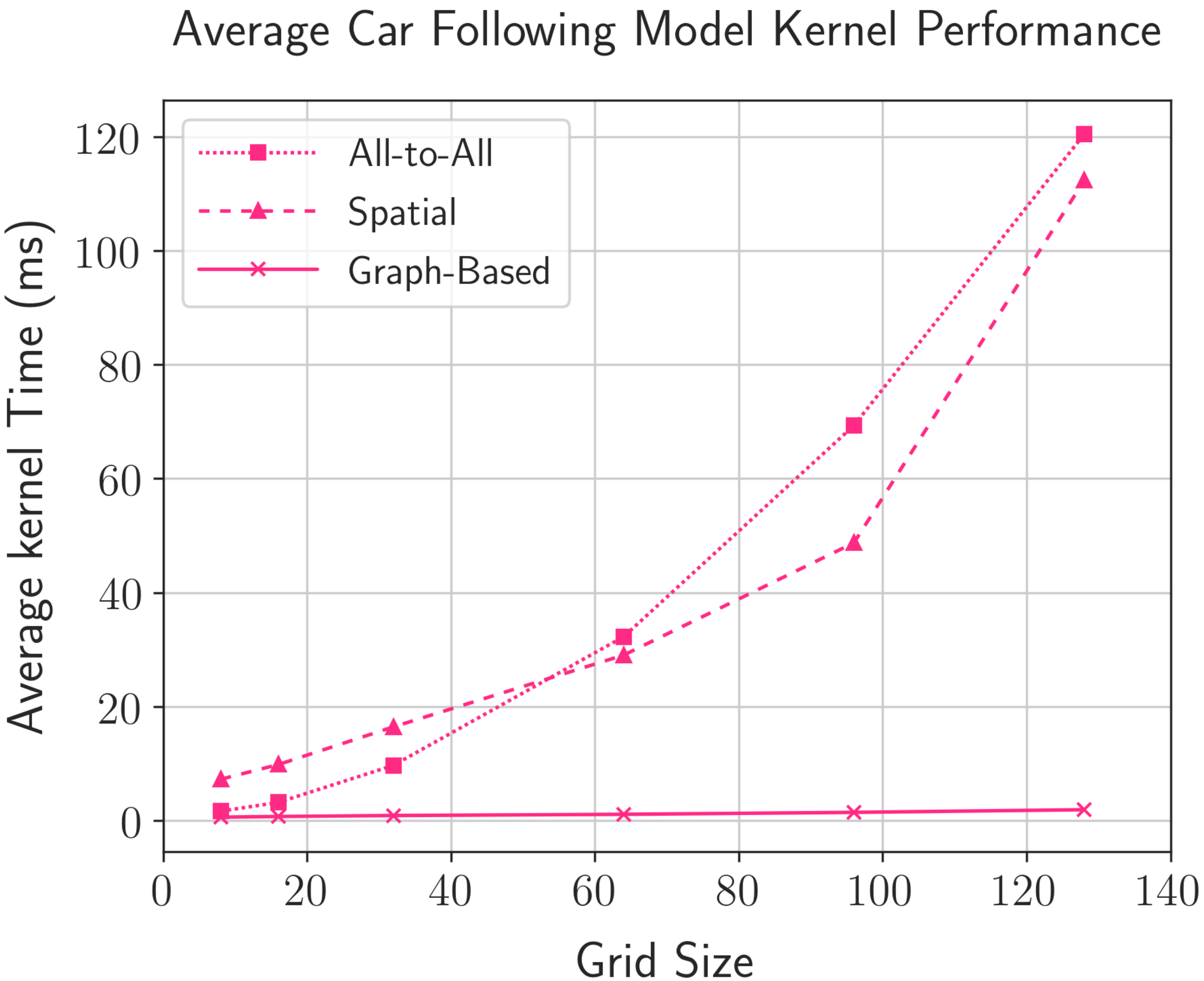
Sorted: 

Histogram



Graph Based Communication Performance

- Measured performance of message input/output
- Higher output cost
  - ~0.5ms vs ~0.2ms
- **Much Lower** iteration cost
  - ~ 1ms vs ~ 120ms



Graph-based Communication Benchmarking

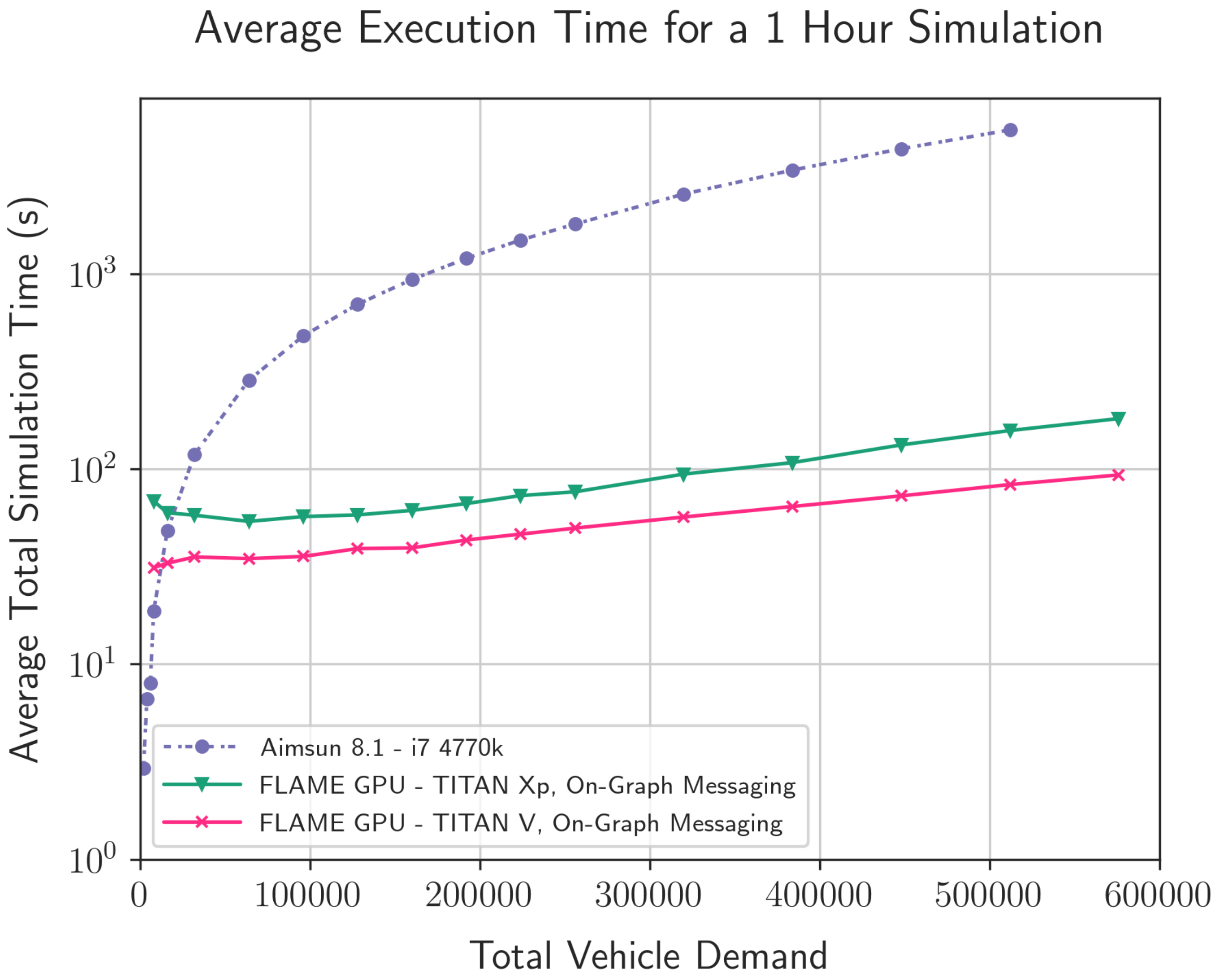
- Benchmarked graph-based communication
- 1 hour simulations
- 3 repetitions
- Titan V
- **Significant performance improvement!**

Graph-based Communication Benchmarking

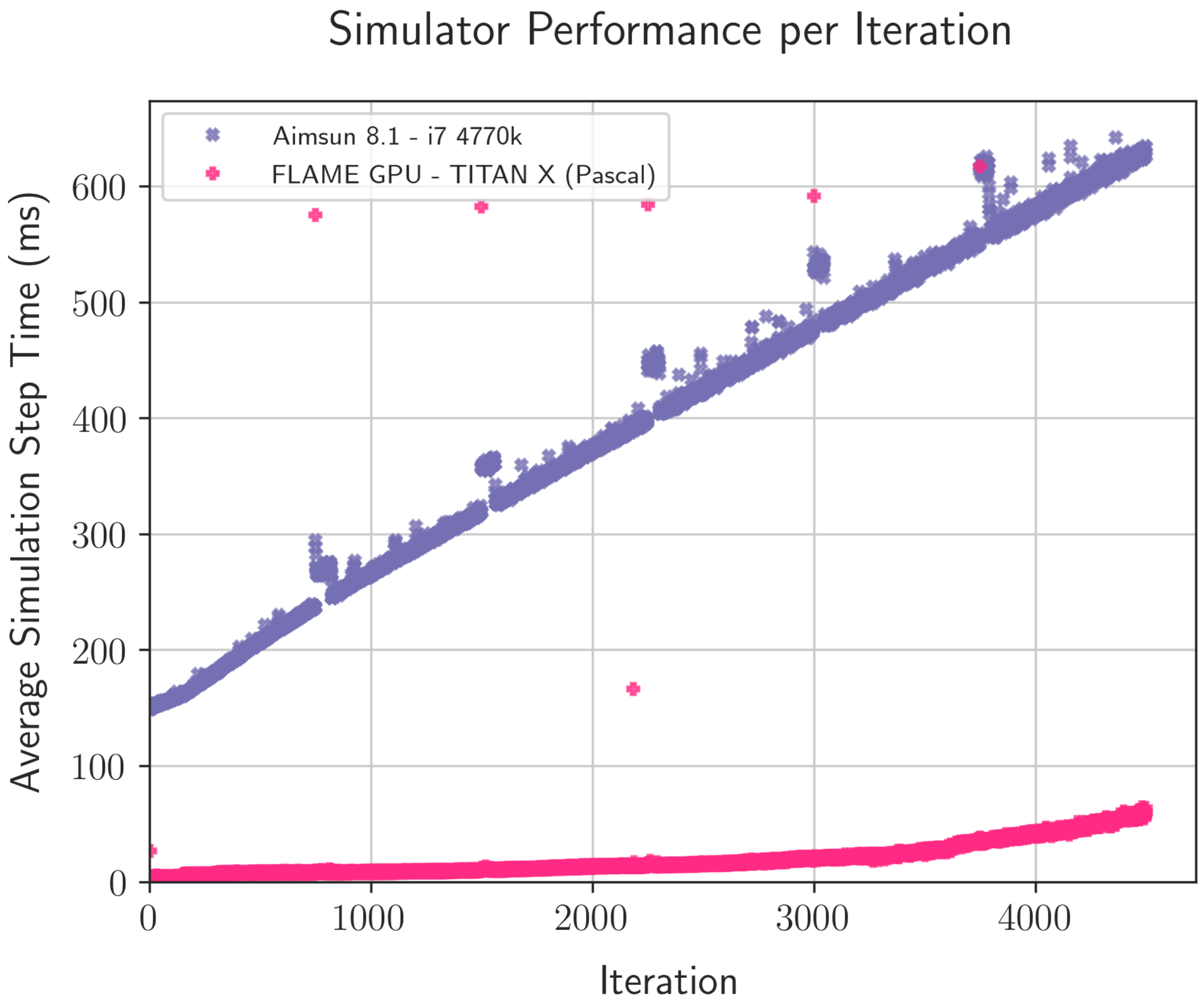
- Benchmarked graph-based communication

- 1 hour simulations
- 3 repetitions

- Titan V
- 0.5 million vehicles
  - 82.04 seconds
  - **66x faster than CPU**
  - **44x faster than real-time**
  - 1.9x faster than Titan Xp

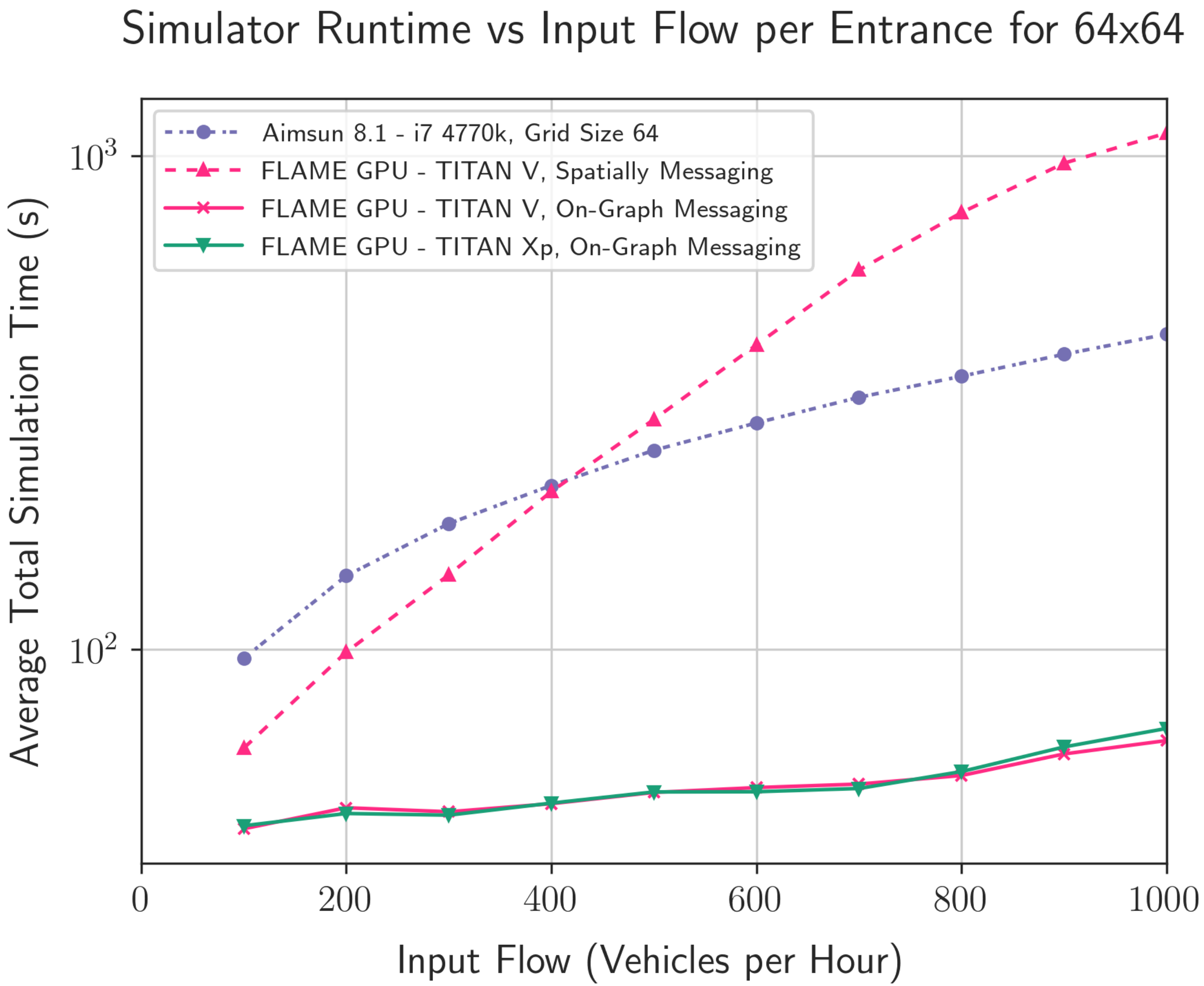


Run-time per Simulation Iteration



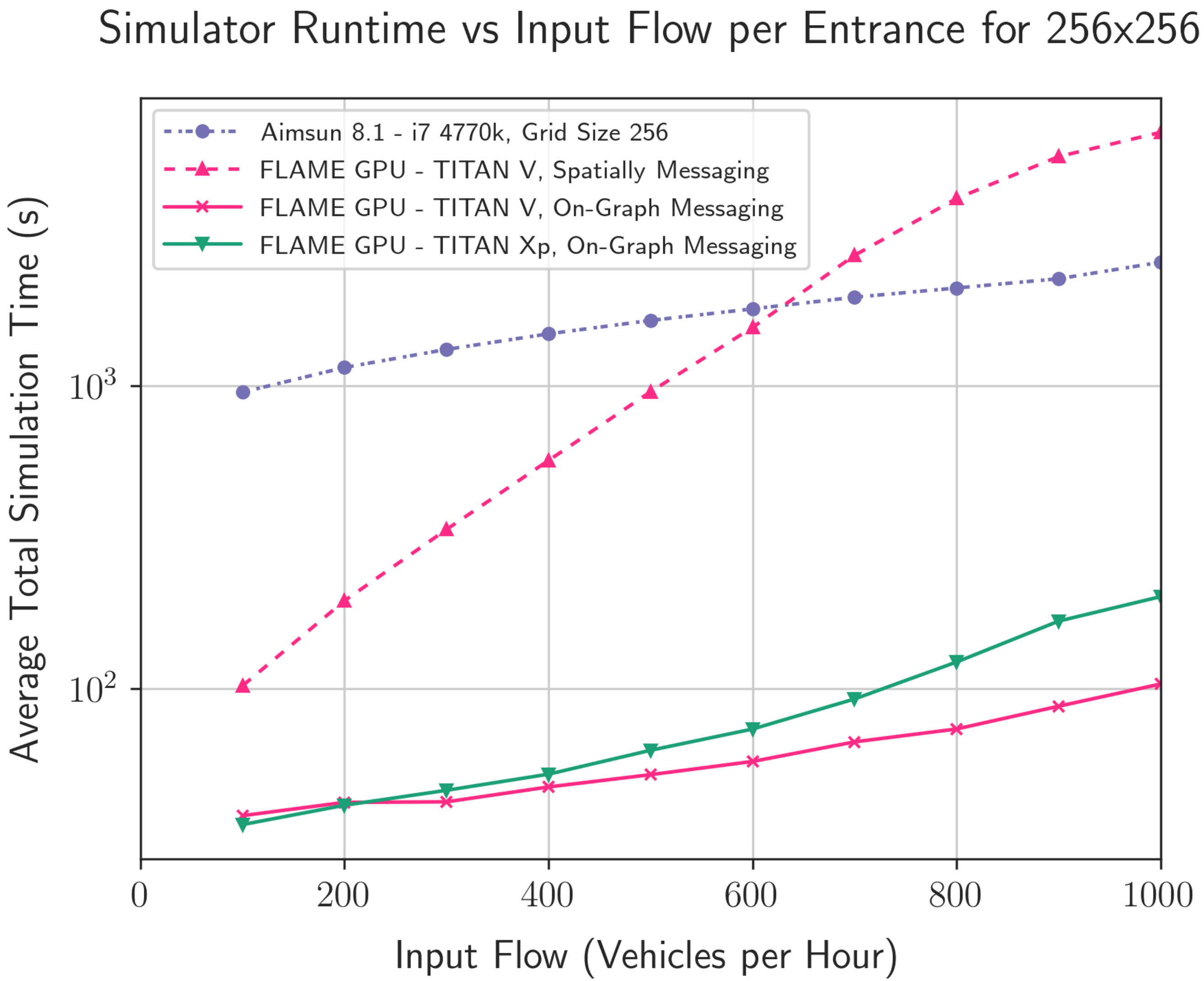
- Timed individual iterations
- 256 x 256 grid
  - Up-to 256,000 vehicles
- i7 4770k
- Titan X (Pascal)
- Runtime increases as population grows
- Anomalous Values from periodic detector behaviour

Input Flow Benchmarking: 64x64 grid



- Varied input flow of vehicles per edge
  - I.e. vehicle density
- 64 x 64 grid
- Spatially Partitioned Messaging
  - Low: **1.5x** faster than CPU
  - High: **2.6x** slower than CPU
- Graph Partitioned Messaging
  - Up to **6.6x** faster than CPU
  - Up to **17.0x** faster than SPM
- Titan V up to **5%** faster than Titan Xp

Input Flow Benchmarking: 256x256 grid



- Varied input flow of vehicles per edge
  - I.e. vehicle density
- 256 x 256 grid
- Spatially Partitioned Messaging
  - Low: **9.3x** faster than CPU
  - High: **2.7x** slower than CPU
- Graph Partitioned Messaging
  - Up to **24.7x** faster than CPU
  - Up to **66.3x** faster than SPM
- Titan V up to **94%** faster than Titan Xp

Other Work

- Additional Functionality
- Multi-Mode GPU Simulations
- Machine Learning Surrogate Models
- FLAME GPU 2

Additional Functionality

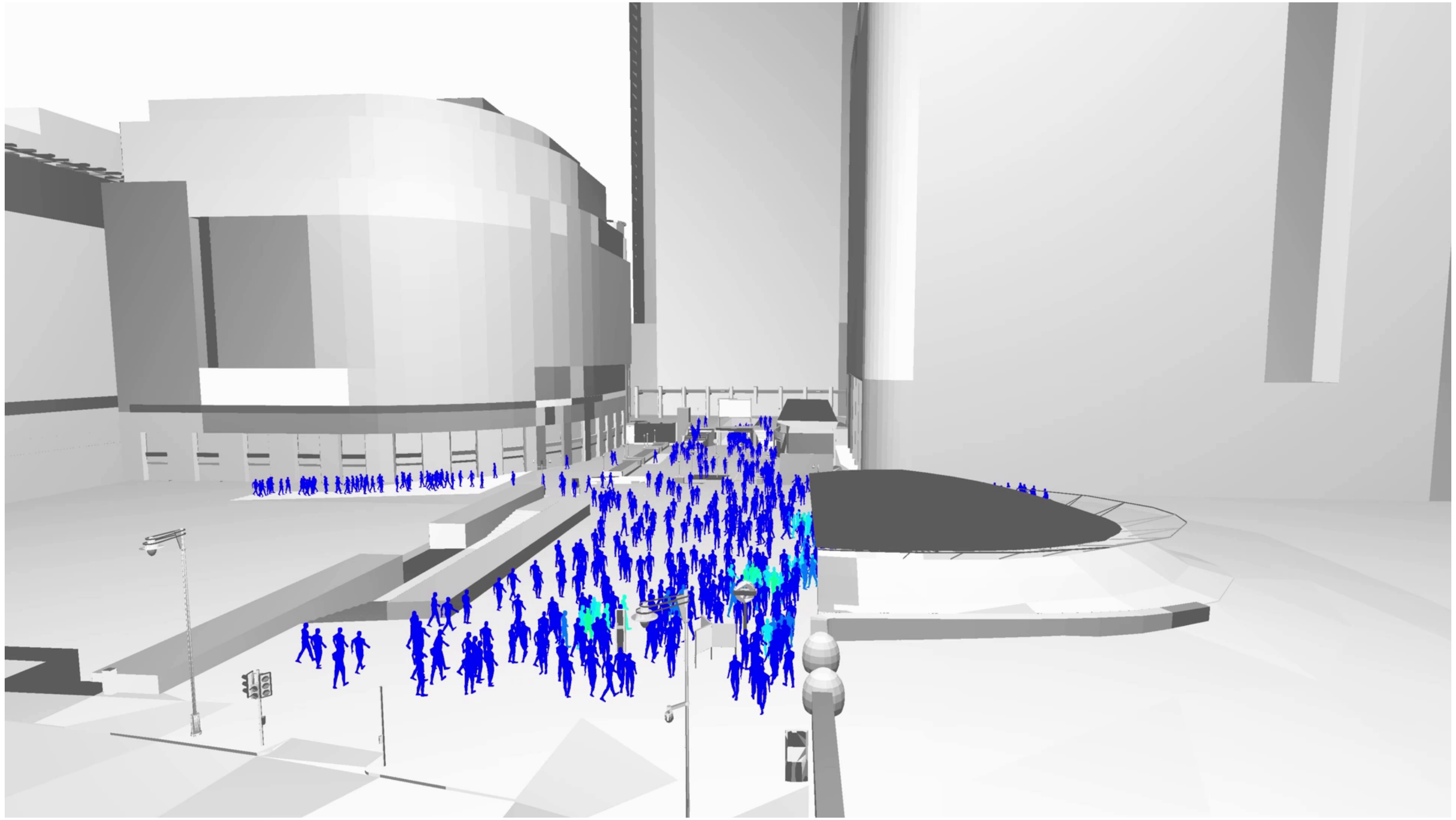
- Real world simulation requires additional functionality
  - Multi-lane roads
  - Dynamic infrastructure
  - O-D Routing
  - Gather additional statistics
- Room for further performance improvements
  - Reduce load on global memory
  - Improve use of CUDA Streams
  - Will be implemented in future versions of FLAME GPU

M24 motorway at night - Bob McCaffrey

https://www.flickr.com/photos/mccaffrey_uk/3207277407

Multi-Mode Simulation: Cars & Pedestrians

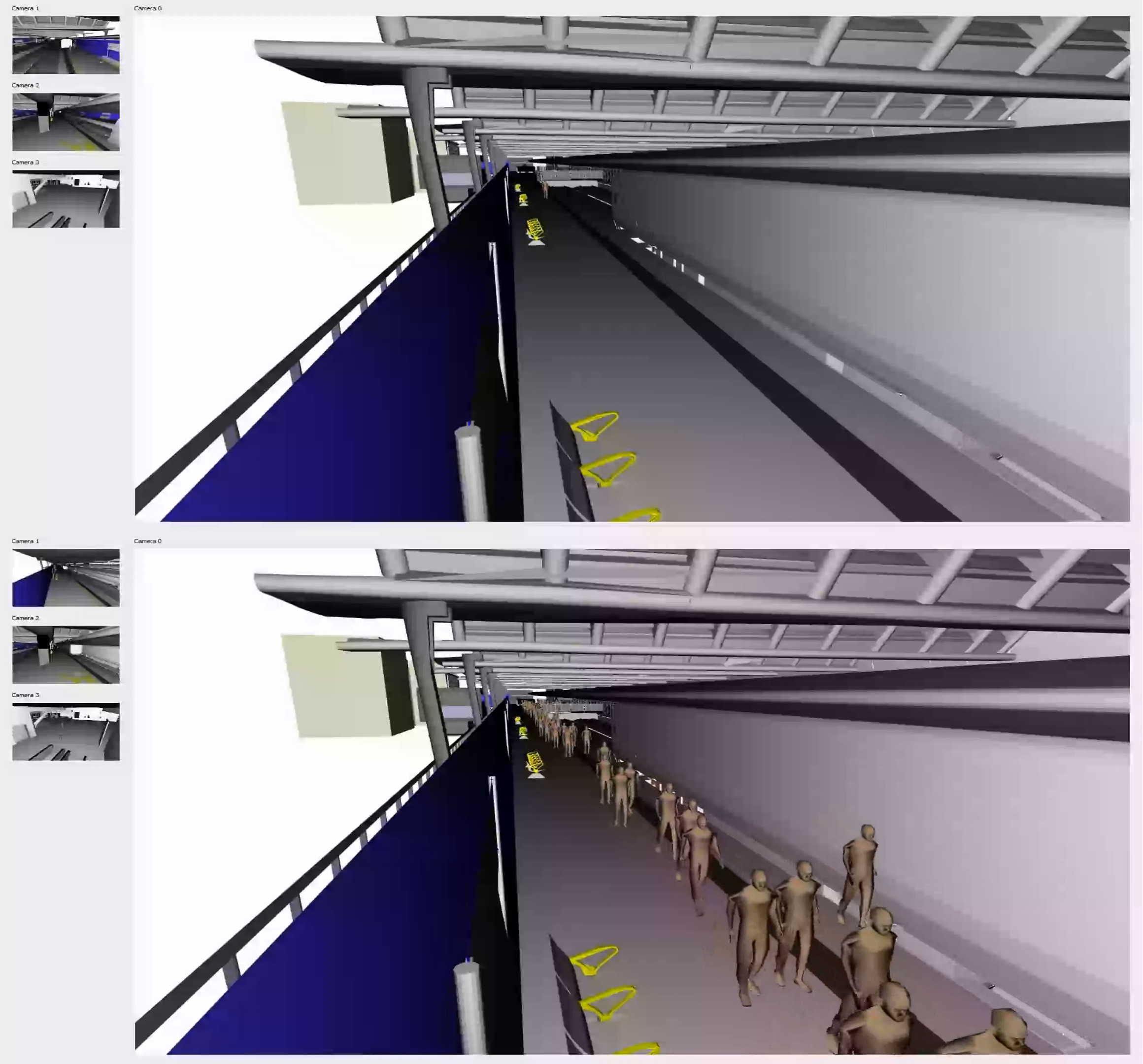
- Simulate Pedestrians and Vehicles on GPU
- Urban shared spaces
- Social Force Pedestrian Simulations icluded in FLAME GPU examples
- Real-time simulations of 100,000s of pedestrians



Multi-Mode Simulation: Cars, Pedestrians & Rail

- SIEMENS Sheffield Advanced Multi-Model Simulator
- Multi-modal Smart-City Simulation
  - GPU accelerated pedestrian simulator
  - CPU rail simulator
  - CPU road network simulator (SUMO)
- Evaluate rail network performance including pedestrian behaviours in station

- More information: youtu.be/Rz_XzbZIMes



- **Robert Chisholm** r.chisholm@sheffield.ac.uk
- **Paul Richmond** p.richmond@sheffield.ac.uk

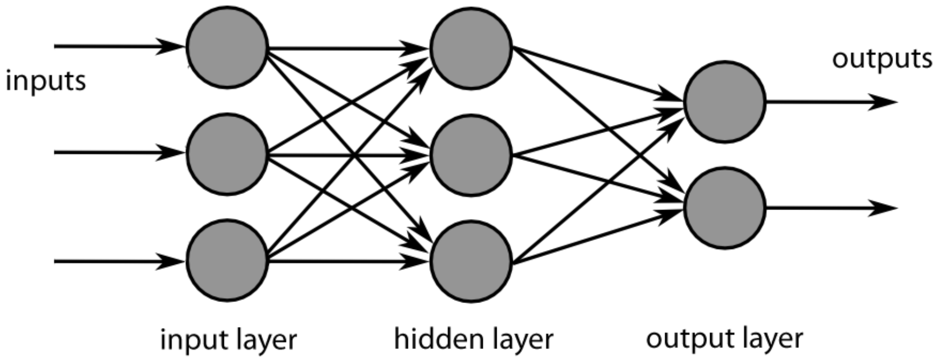
Surrogate Transport Network Models

- Machine Learning inference faster than simulation
- But Networks biased towards training data
  - **Low accuracy for low-frequency events**

1. **Supplement training data with Simulated Data**
   - Improving accuracy for low-frequency events
2. **Surrogate Models**
   - (Deep) Neural Networks to predict simulator output
   - Accelerates Parameter Search
     - Calibration & Validation
     - Optimisation

- Generate huge amounts of training data using GPU accelerated simulations

- **James Pyle** jcbpyle1@sheffield.ac.uk
- **Paul Richmond** p.richmond@sheffield.ac.uk



FLAME GPU 2

- *Under Active Development*

- Ground-up Rewrite
- Modern C++/CUDA
- Improved:
    - Performance
    - Usability
    - Maintainable

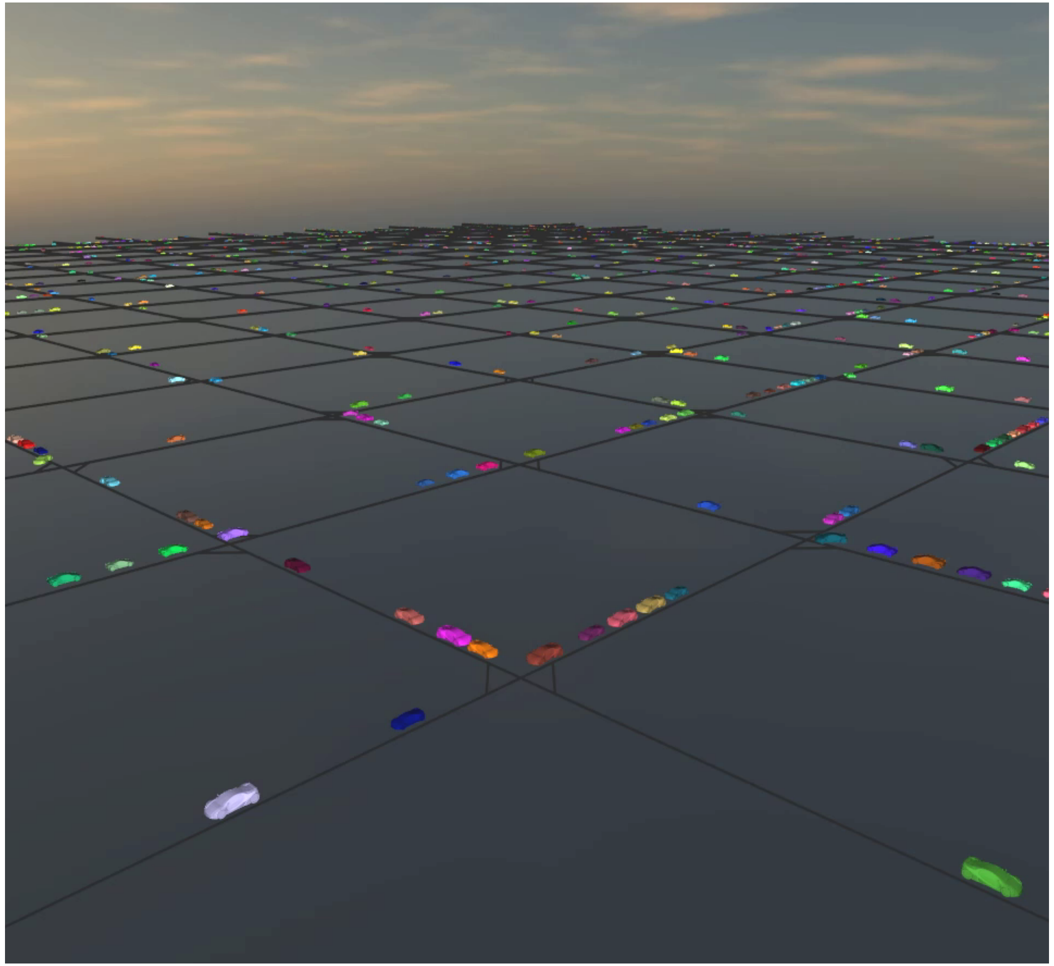
- New Functionality (planned)
    - Automatic parameter exploration
    - Concurrent Batch Simulation
    - Multi-GPU Support & UVM
    - Higher-level language bindings
        - I.e. Python

github.com/flamegpu/flamegpu2_dev

Conclusions

Conclusion

- **Faster-than-real-time city-scale microsimulation**
- Simulation of 500,000 vehicles
  - **44x faster than real-time**
  - **66x faster than Aimsun 8.1 (CPU)**
- Achieved using FLAME GPU
  - New graph-based agent communication strategy
  - Cross-validated implementation
  - FLAME GPU 2 is under development



Thank You

- Peter Heywood
  - p.heywood@sheffield.ac.uk
  - ptheywood.uk
  - rse.shef.ac.uk
- Paul Richmond
  - p.richmond@sheffield.ac.uk
  - paulrichmond.shef.ac.uk
- Robert Chisholm
  - r.chisholm@sheffield.ac.uk
- James Pyle
  - jcbpyle1@sheffield.ac.uk

The University Of Sheffield.

Supported by

- EPSRC fellowship “Accelerating Scientific Discovery with Accelerated Computing” (EP/N018869/1)
- DfT Transport Technology Research Innovation Grant (T-TRIG July 2016)

More Information

"Data-parallel agent-based microscopic road network simulation using graphics processing units"

Peter Heywood, Steve Maddock, Jordi Casas, David Garcia, Mark Brackstone & Paul Richmond. 2017

doi.org/10.1016/j.simpat.2017.11.002