Accelerating Road Network Simulations using GPUs

Peter Heywood

The University of Sheffield

Table of contents

Road Network Simulation

Road Network Simulation

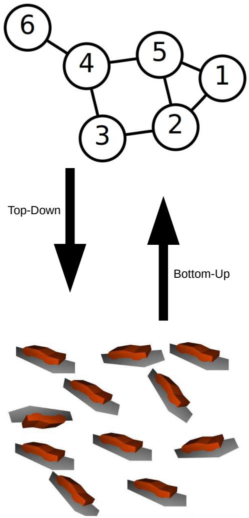
- Global transport demand is increasing [4]
- Many constraints on transport networks
- Simulation can improve use of limited resources
  - Planning
  - Management

Road Network Simulation

- Simulations are becoming more computationally expensive
  - **Larger** - City-scale, National-scale
  - **More Complex** - CAVs, Smart Motorways, ...
  - **More Permutations** - weather, demand, ...
- **Better than real-time** simulations required for active management
- Performance is limiting the use of simulation [1]
- Need higher performance simulators!

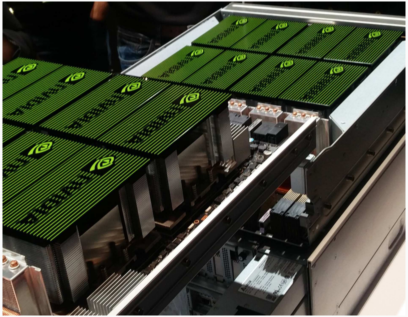
Road Network Simulation Categories

- **Macroscopic** Simulation
  - Top-Down
  - High level, flow simulation
- **Mesoscopic** Simulation
  - Mid-level
  - Fine-grained macrosimulation or Platoons/groups
- **Microscopic** Simulation
  - Bottom-Up
  - Low level, individual vehicles



Graphics Processing Units (GPUs)

- Massively parallel, many-core co-processors
- Data-parallel algorithms and data structure
  - Possibly very different to CPU
- Suitable for all scales of road network simulation
  - Different degrees of parallelism expressed
  - Different levels of performance improvement

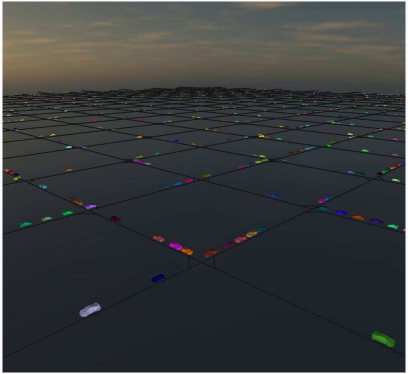


NVIDIA DGX-2

GPU Accelerated *Microscopic* Simulation

Microscopic Simulation

- Bottom-up Simulations
- Individual vehicles
- Agent Based Modelling (ABM) [6]
  - Intuitive descriptions of behaviour and interactions
    - with other vehicles
    - with the environment
  - Complex behaviour emerges from simple rules
- Very computationally expensive
- Large volume of data required and generated



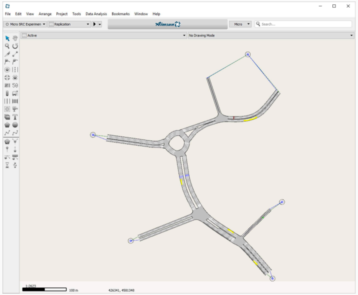
FLAME GPU Road Network Microscopic Simulation

Our Aims

Aims

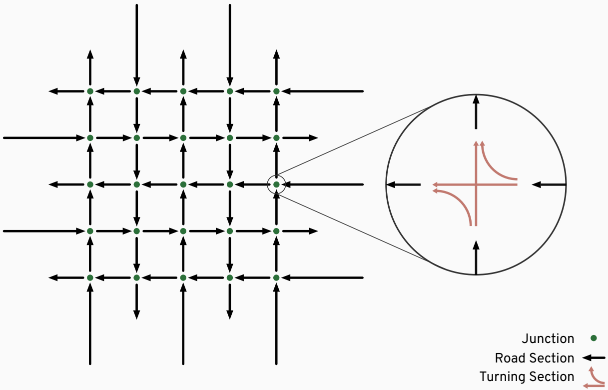
- Demonstrate GPUs are suitable and performant
    - Implement a subset of models from commercial tool
    - Cross-validate GPU implementation
    - Benchmark using a scalable model

- **Aimsun [2]**
    - Commercial, multi-core CPU, microscopic simulator
    - Used globally within the transport industry
    - Can simulate a broad array of transport networks and infrastructure



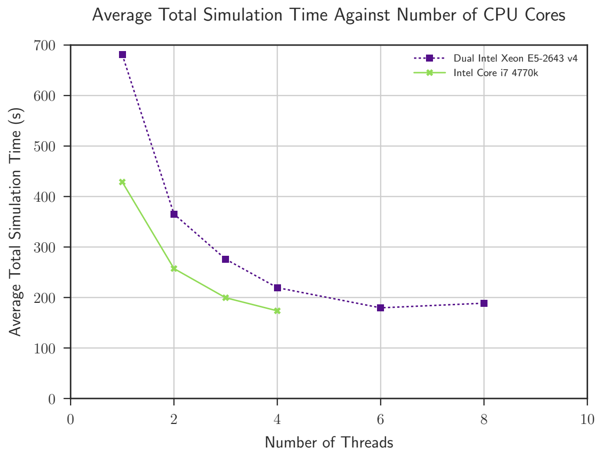
Procedurally Generated Network

- Manhattan-style grid network
- Single lane, one-way roads
- Stop-signs at junctions
- Entrances and Exits at the edge of the simulated grid



Aimsun 8.1 CPU Performance

- Single size of grid network
- 3 repetitions
- **Diminishing Returns** from additional cores



Models and Functionality

- Gipps' Car Following Model [9, 14]
- Aimsun Gap Acceptance Model [2]
- Turning Probability based Routing [13]
- Simulated Vehicle Detectors [13]
- Constant Vehicle Arrival [13]

Gipps' Car Following Model

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

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

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

FLAME GPU

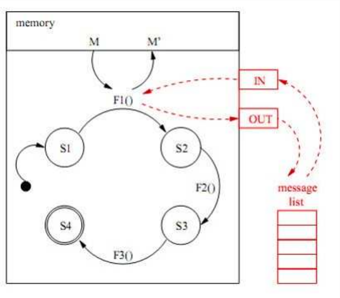
- Flexible Large-scale Agent Modelling Environment for the GPU [11]
- Template-based simulation environment for high performance simulation
- Agents represented as X-Machines
  - with *message lists* for communication
- Abstracts the CUDA programming model away from the user
  - I.e. A modeller writes an XML file and simple C/C++ code

flamegpu.com

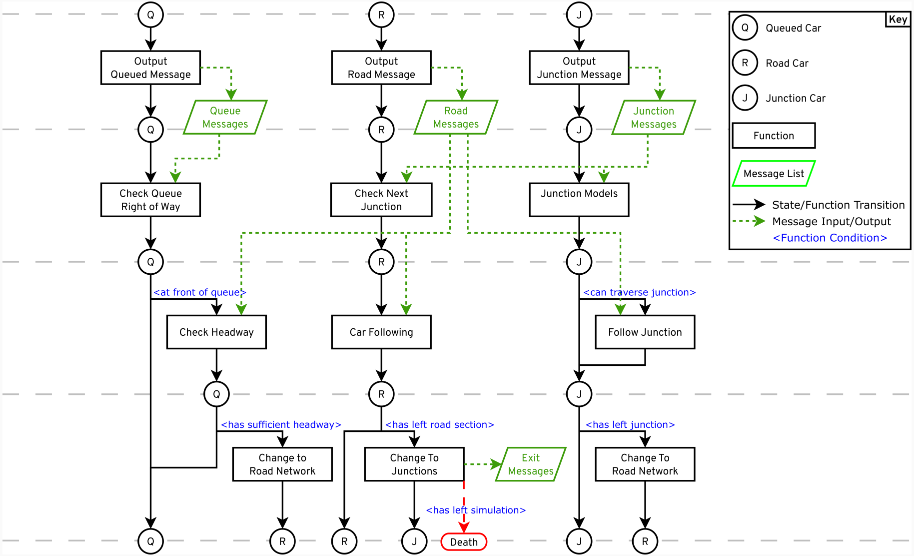
github.com/flamegpu

FLAME GPU

- State-based representation minimises divergence
- SoA per state list - improves data access pattern
- Message lists avoid race-conditions
    - Natural synchronisation barriers
- Reduce global reads via shared memory



FLAME GPU Road Network Simulation State Diagram



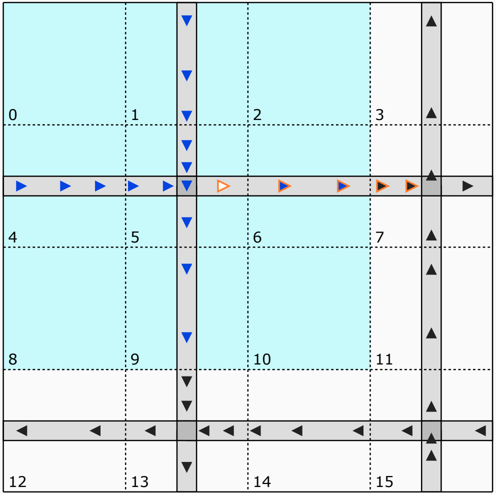
FLAME GPU Communication

- Message lists enable high performance memory access pattern
  - avoids issues with concurrent access to agent memory
- Typically the performance-limiting factor in large-scale simulations
- Specialisation for typical communication patterns [12]
  - All-to-All
  - Discrete Partitioned Messaging (2D Cellular Automata)
  - Spatially Partitioned Messaging (2D & 3D Continuous Agents)
- Non-optimal for road network models

On-Graph Communication

- Communication between vehicles is based on the transport network
- I.e. Gipps' car following model only involves the lead vehicle
- Associate messages to the graph data structure
- Reduce the number of messages to be iterated
  - by accessing messages from the relevant edge(s) or vertices

Communication	Messages
All-to-All	42
Spatial	18
Graph	5



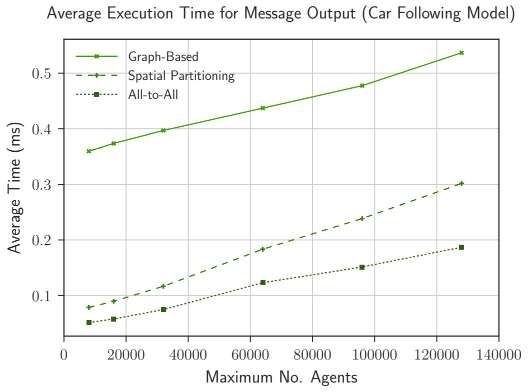
Example highlighting FLAME GPU Communication strategies

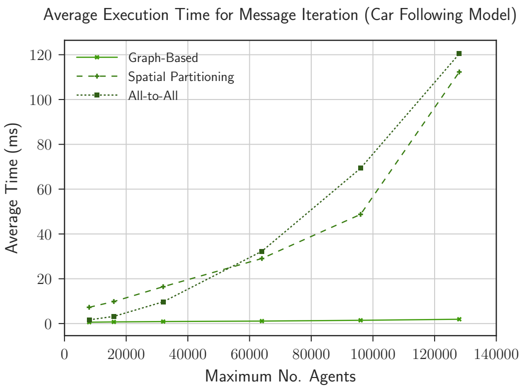
On-Graph Communication

- Compressed Sparse Row (CSR) representation of graph
- Messages contain edge or vertex index
- Sort message list based on edge (or vertex) index
  - *Counting Sort*
  - Shared-memory atomics
  - Builds data structure to access messages whilst sorting
- Can access a single edge, or use the CSR to explore the message-list
- Available in the next release of FLAME GPU (1.5)

On-Graph Communication Performance

- Measured performance of message list output and input for car-following
- Higher output cost, **much** cheaper message input cost.





Performance Benchmarking

1. Scale population and environment
2. Scale population for fixed size environment

- 3 repetitions
- 1 hour of simulated time
- Multiple hardware configurations

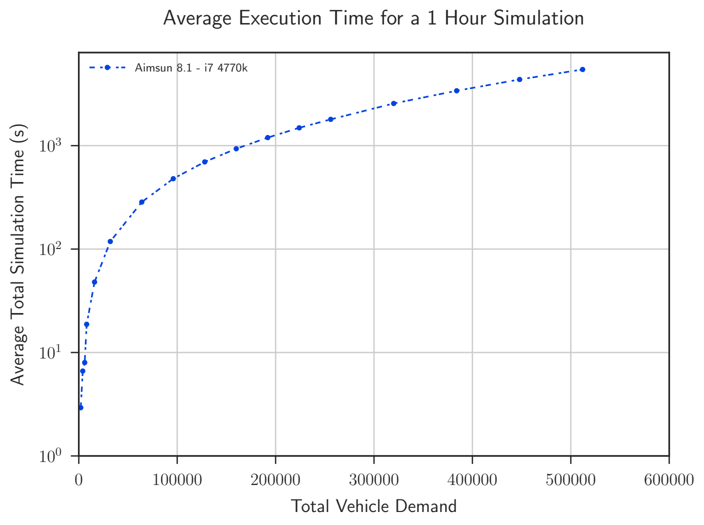
Workstation

- Windows and Linux
- i7 4770k (4 Cores)
- GTX 1080
- Titan X (Pascal)
- Titan V

Nvidia DGX-1

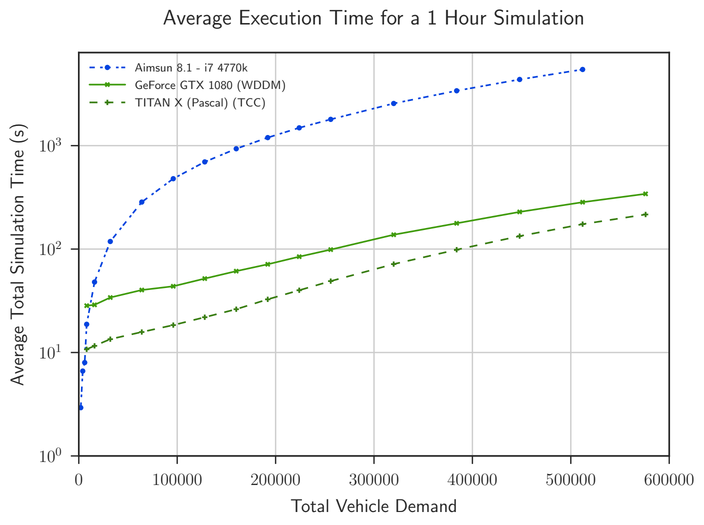
- Linux
- 2x Xeon E5 2698 v4 (20 cores each)
- 8x Tesla P100

Scale Population and Environment



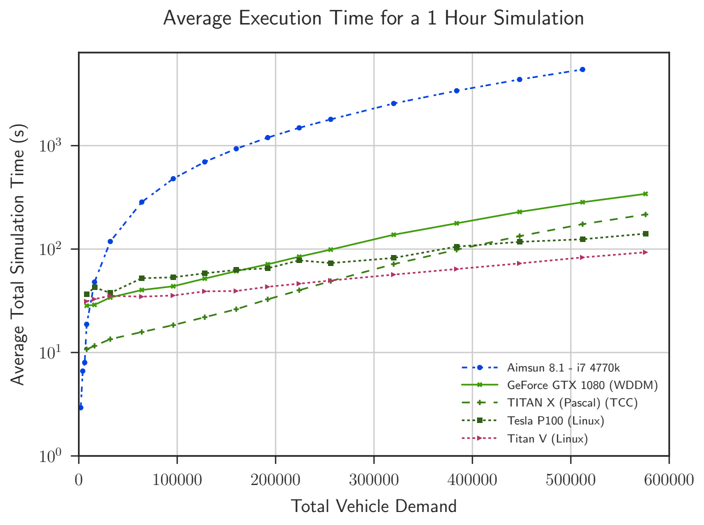
- 0.5 Million Vehicles:
- CPU - Windows
  - 5447s

Scale Population and Environment



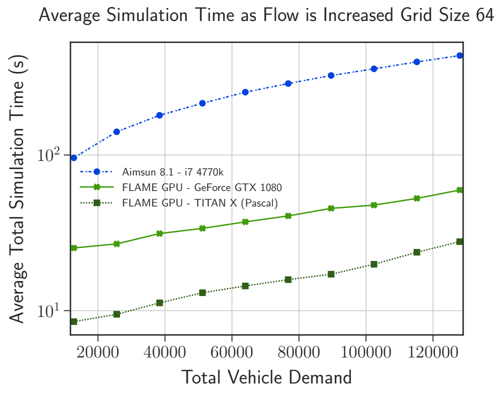
- 0.5 Million Vehicles:
- CPU - Windows
  - 5447s
- GPU - Windows
  - 174.2s
  - 31x speed up (Titan X (Pascal))

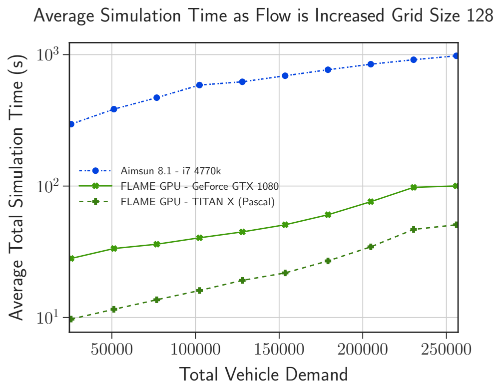
Scale Population and Environment



- 0.5 Million Vehicles:
- CPU - Windows
  - 5447s
- GPU - Windows
  - 174.2s
  - 31x speed up (Titan X (Pascal))
- GPU - Linux
  - 82.04s
  - 66x speed up (Titan V)

Scale Population for Fixed Environment





GPU Accelerated *Macroscopic* Simulation

Macroscopic Simulation

- Top-Down Simulations
- Models networks as flows on roads (i.e pipes)
- High level of abstraction from reality
- Relatively long time steps
  - Misses short-term events
- Low data requirements
- Lower computational cost
  - But still expensive for large scale simulations

SATURN

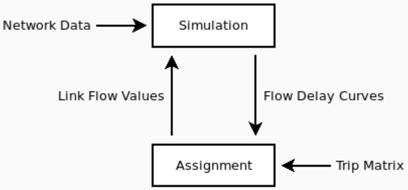
- Simulation and Assignment of Traffic to Urban Road Networks [16]
- Commercial, multi-core CPU software
- Originally Developed in the 1970s
- Used by companies and governments for (mostly) planning
    - Highways England, Transport for London (TfL), ...
- Fortran 77 with OpenMP

ATKINS
Member of the SNC-Lavalin Group

SATURN

SATURN Simulation-Assignment Loop

- Iterative equilibrium-based algorithm of Assignment and Simulation
  - Wardrop's Equilibrium [17]
- **Assignment Phase**
  - Network + Demand Matrix -> Flow-per-edge
  - Vehicles types are considered independently (*User Classes*)
    - Cars, Taxis, Buses, HGVs, ...
  - Trip Matrix contains many *Origins* and *Destinations*
    - Known as *Zones* or *Centroids*



Assignment-Simulation Loop in SATURN [16]

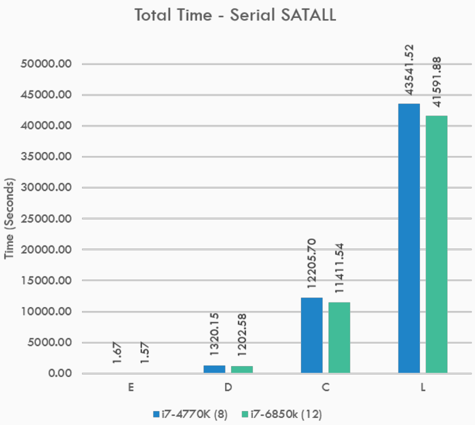
SATURN Models

- Road networks are very sparse graphs
  - Preprocessing step to create a denser representation
  - Referred to as "*Spider Network*"
    - Contraction Hierarchies
- These are **very sparse** graphs, even when preprocessed
- Range of scales from *tiny* to *very very large*

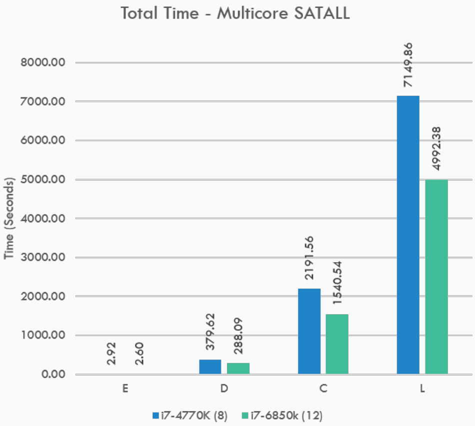
Model	Size	User Classes	Centroids	Vertices	Edges
E	Town	2	12	17	74
D	Small City	13	547	2700	25385
C	Large City	5	2548	15179	132600
L	Metropolitan	5	5194	18427	192711

CPU Performance - Serial and OpenMP

Single Core CPU

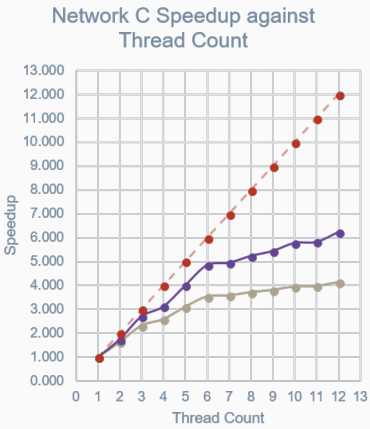


Multi-Core CPU

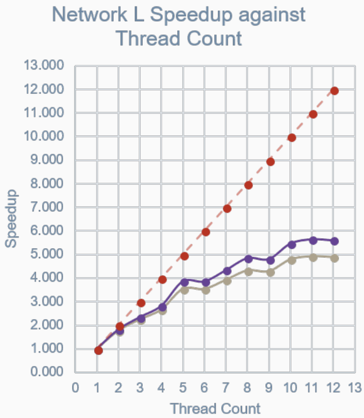


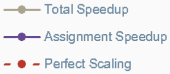
CPU OpenMP Scaling

Single Core CPU



Multi-Core CPU

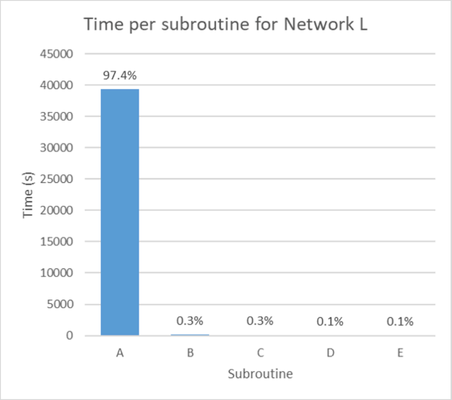




- i7 6850k
- 6 cores
- 12 threads
- 3 Repetitions
- **Diminishing Returns**

SATURN Profiling - Serial

- Serial version of SATALL
- Largest available model (L)
  - London + surrounding area
- > 11 Hour Runtime
- 97.4% in a single subroutine
- Computes shortest paths for an origin centroid
  - Accumulates flow for each trip from the origin
- Most time spent calculating paths



CPU Algorithm

- **Single Source Shortest Path (SSSP)**
  - Uses the D'Esopo-Pape algorithm [10]
    - An efficient, *highly-serial* algorithm
    - Algorithmic decision in the 1970s, due to benchmarking at the time [15]
    - A modern implementation of Dijkstra's algorithm [5, 8] is up to 50% faster
- **Flow Accumulation**
  - Trace all routes from an origin to destination zones
  - Update per-edge flow value at each step
  - Double precision to avoid numerical precision loss
- Calculated per-origin centroid, per-userclass, at each iteration

GPU Shortest Path Algorithm

- Need data-parallel algorithms for the GPU
  - Sacrifice efficiency to enable parallelism
  - More work, but in parallel

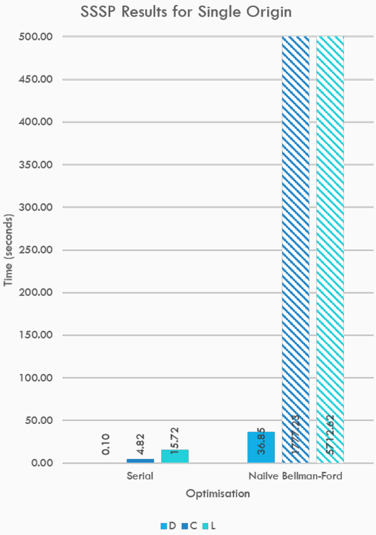
Bellman-Ford Algorithm [3, 7]

- For up to $|v| - 1$ iterations
  - For each Edge in the network
  - If the edge is a cheaper route to the destination node, update the route.

- Significant changes required to provide a performance improvement for road networks vs Dijkstra or D'Esopo-Pape

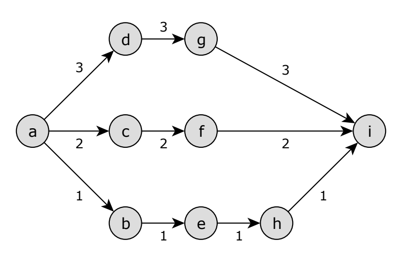
Initial GPU Implementation

- Naive version of the Bellman-Ford Algorithm
- **Much, Much, Much, Much Slower...**
- 364*x* slower
- Inefficient use of compute
- Inefficient data transfer
- Lots of unnecessary work



Multiple Source Bellman-Ford

- *Frontier-based* implementation of Bellman-Ford
- Solve for **multiple origins** concurrently
- Threads co-operate to balance work-load
- Solve for **multiple independent user-classes** concurrently



Iteration	Frontier Vertices
0	a
1	b c d
2	e f g
3	h i
4	i
5	

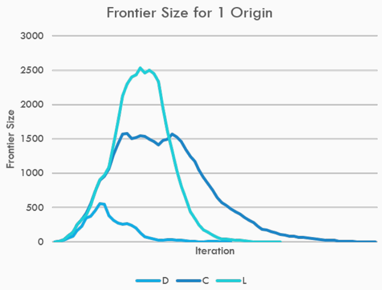
Origin-Vertex Frontier

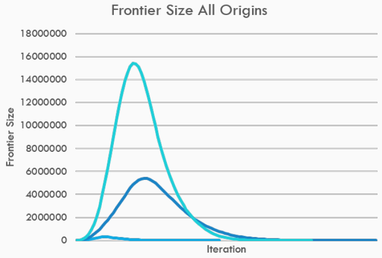
Vertex Frontier

- Tracks which vertices could cause an update
  - Increases efficiency, but decreases parallelism
- Not enough work
  - Latency bound, Low number of threads ($<$ 2500 for network L)

Origin-Vertex Frontier

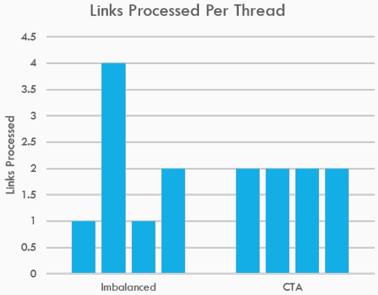
- Multiple origins concurrently
- Track which origin each fronter vertex belongs to
  - Increases parallelism
  - Significantly increases memory requirements





Block-level Load Balancing

- Number of edges per vertex varies
  - Co-operative Thread Array (CTA)
  - Threads in a block collectively work on the same portion of the origin-vertex frontier
  - Balances work load across threads (and warps) in the block
  - Improved L2 bandwidth 4.8x (148GB/s to 716GB/s)
  - CUDA 9.0 introduces Cooperative Groups API

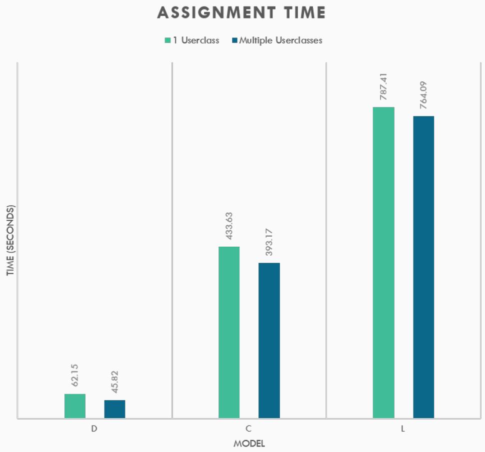


Flow Accumulation

- For each trip from origin to destination:
  - Trace the shortest path, atomically updating per-link flow
- Good performance on Pascal and Volta
- **But** `atomicAdd(double)` not available on Maxwell and older
  - `atomicCAS` implementation very slow due to high *atomic contention*
  - Complex workaround:
    - Device-wide sort
    - Block-wide key-value reduction
    - Single global `atomicAdd` per edge in the block
  - Faster the naive algorithm on Maxwell, but slower than Pascal

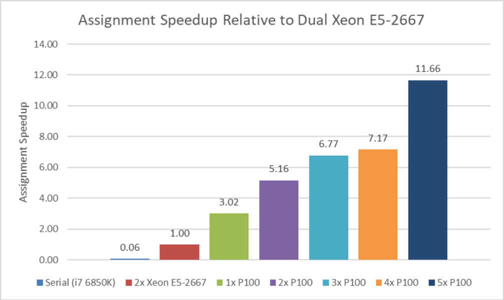
Multiple User Classes

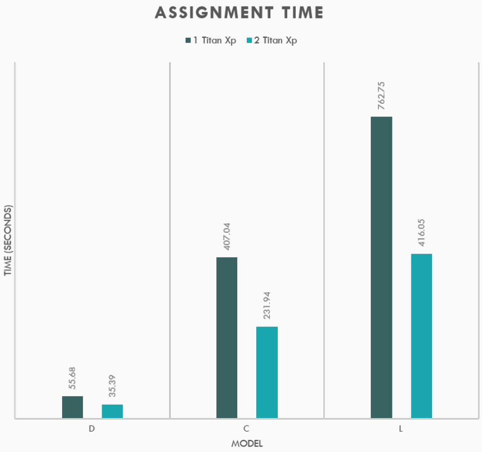
- User classes can processed independently
- CUDA stream per user-class
- Increases parallelism
- Not a significant speed up
    - Serialisation when device oversubscribed
- *Enables the use of Multiple GPUs*



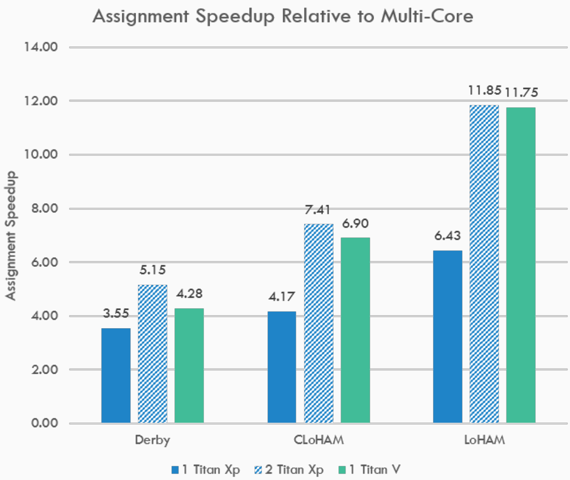
Multiple GPUs

- Distribute user classes between GPUs
- Imbalanced workload between devices
    - Only assign whole user classes





Volta GPU Architecture

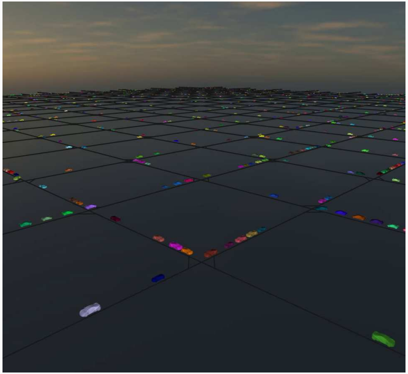


- Up to 80% performance improvement vs 1 Titan Xp
- Speed up relative to 6 core i7
- No source code changes
  - Other than updating libraries (CUB) and CUDA version.

Summary

Conclusion

- Microscopic Simulation
  - Up to 66x speed up using a Titan V
  - Real-time-ratio of 39x for up to 576000 vehicles
- Macroscopic Assignment
  - Up to 11.7x speed up on 1 Titan V vs 6 core i7
  - Up to 11.8x speed up on 5 P100 vs 2 CPUs
- More simulations in less time
- Large simulations feasible
- Better-than-real-time microsimulation of 0.5 million vehicles is achievable



Thank You

Supported By

- DfT Transport Technology Research Innovation Grant (T-TRIG July 2016)
- EPSRC fellowship "Accelerating Scientific Discovery with Accelerated Computing" (EP/N018869/1)
- Support from Atkins, STFC, TSC & Aimsun

Contact

- p.heywood@sheffield.ac.uk
- @ptheywood
- ptheywood.uk
- rse.shef.ac.uk

More Information

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

`https://doi.org/10.1016/j.simpat.2017.11.002`

References i

[1] C. Antoniou, J. Barcelò, M. Brackstone, H. Celikoglu, B. Ciuffo, V. Punzo, P. Sykes, T. Toledo, P. Vortisch, and P. Wagner.
Traffic simulation: Case for guidelines.
2014.

[2] J. Barceló and J. Casas.
Dynamic network simulation with aimsun.
In *Simulation approaches in transportation analysis*, pages 57–98. Springer, 2005.

[3] R. Bellman.
On a routing problem.
Quarterly of applied mathematics, pages 87–90, 1958.

[4] Department for Transport.
Road traffic forecasts 2015.
https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/260700/road-transport-forecasts-2013-extended-version.pdf, Mar. 2015.

[5] E. W. Dijkstra.
A note on two problems in connexion with graphs.
Numerische mathematik, 1(1):269–271, 1959.

References ii

[6] G. Eliasson.
Modeling the experimentally organized economy, 1991.

[7] L. R. Ford Jr.
Network flow theory.
Technical report, DTIC Document, 1956.

[8] M. L. Fredman and R. E. Tarjan.
Fibonacci heaps and their uses in improved network optimization algorithms.
Journal of the ACM (JACM), 34(3):596–615, 1987.

[9] P. Gipps.
A behavioural car-following model for computer simulation.
Transportation Research Part B: Methodological, 15(2):105–111, 1981.

[10] U. Pape.
Implementation and efficiency of Moore-algorithms for the shortest route problem.
Mathematical Programming, 7(1):212–222, 1974.

[11] P. Richmond.
Flame gpu technical report and user guide (cs-11-03).
Technical report, Technical report, Department of Computer Science, University of Sheffield, 2011.

References iii

[12] P. Richmond and D. Romano.
Template-driven agent-based modeling and simulation with cuda.
GPU Computing Gems Emerald Edition, Applications of GPU Computing Series, pages 313–324, 2011.

[13] Transport Simulation Systems.
Aimsun 8 Dynamic Simulators Users' Manual, 2014.

[14] M. Treiber, A. Hennecke, and D. Helbing.
Congested traffic states in empirical observations and microscopic simulations.
Physical review E, 62(2):1805, 2000.

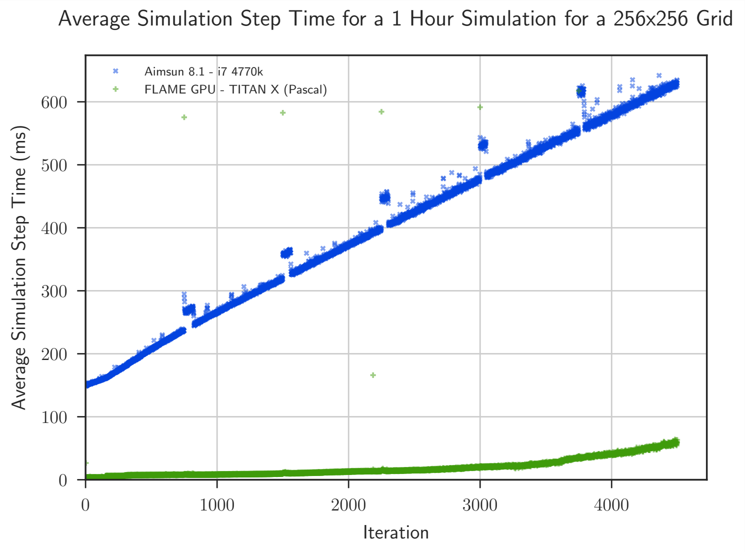
[15] D. Van Vliet.
Improved shortest path algorithms for transport networks.
Transportation Research, 12(1):7–20, 1978.

[16] D. Van Vliet.
SATURN - a modern assignment model.
Traffic Engineering & Control, 23(HS-034 256), 1982.

[17] J. G. Wardrop.
Some theoretical aspects of road traffic research.
Proceedings of the institution of civil engineers, 1(3):325–362, 1952.

Backup Slides

Microsimulation: Runtime per Iteration



- Population grows as time progresses
- Anomalous values correlate with detector outputs
- Every 800 iterations (10 minutes)

About Me

- MComp Computer Science & Artificial Intelligence at Sheffield (2010-2014)
- PhD Student at Sheffield (2014 - 2018)
- Research Software Engineer (RSE) and PhD Candidate at Sheffield (2018-2021)

Research
Software
Engineering
Sheffield.