Simulation Modelling Practice and Theory 000 (2017) 1–13

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

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

Peter Heywood[a,*], Steve Maddock[a], Jordi Casas[b], David Garcia[b], Mark Brackstone[b], Paul Richmond[a]

[a] Department of Computer Science, The University of Sheffield, United Kingdom
[b] Transport Simulation Systems Ltd (TSS), Spain

ARTICLE INFO

Article history:
Available online xxx

Keywords:
Agent-based simulation
GPU
Simulation framework
Transport microsimulation

ABSTRACT

Road network microsimulation is computationally expensive, and existing state of the art commercial tools use task parallelism and coarse-grained data-parallelism for multi-core processors to achieve improved levels of performance. An alternative is to use Graphics Processing Units (GPUs) and fine-grained data parallelism. This paper describes a GPU accelerated agent based microsimulation model of a road network transport system. The performance for a procedurally generated grid network is evaluated against that of an equivalent multi-core CPU simulation. In order to utilise GPU architectures effectively the paper describes an approach for graph traversal of neighbouring information which is vital to providing high levels of computational performance. The graph traversal approach has been integrated within a GPU agent based simulation framework as a generalised message traversal technique for graph-based communication. Speed-ups of up to 43 × are demonstrated with increased performance scaling behaviour. Simulation of over half a million vehicles and nearly two million detectors at a rate of 25 × faster than real-time is obtained on a single GPU.

1. Introduction

Simulations of road networks are used during the development and management of transport networks around the globe. Microscopic road network simulations are fine-grained simulations which simulate individual vehicles within the system, capturing low level behaviours. Agent Based Modelling (ABM) is one microscopic approach, where relatively simple individual behaviours are defined, which combined with interactions between agents and the environment, allows the emergence of complex behaviours. However, microscopic simulations are much more computationally expensive than the more traditional higher-level macroscopic simulations, which use a higher level of abstraction consisting of network flows rather than individual vehicles. Nonetheless, the level of detail captured by microscopic simulations is much greater than that of macroscopic and mesoscopic simulations, including the emergent behaviours enabled by the use of ABM.

* Corresponding author.
E-mail addresses: p.heywood@sheffield.ac.uk (P. Heywood), p.richmond@sheffield.ac.uk (P. Richmond).

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

As microscopic simulations are computationally expensive, current commercial and open-source tools such as Aimsun [1], Vissim [2], MATsim [3] and SUMO [4] can have considerable execution run-times, especially for large scale simulations. This limits the overall effectiveness and uptake of microscopic simulation within the transport sector [5]. To increase simulator performance, existing simulation tools use parallel processing applied to multi-core processors to reduce the run-time of simulations. Task-parallel and coarse-grained data-parallel approaches are typically applied within existing tools. Task-parallelism distributes independent processing tasks to separate processing threads. Coarse-grained data-parallelism applies the same algorithms to different units of data, where the individual units of data are relatively large. These approaches are well-suited to multi-core Central Processing Units (CPUs), but can result in poor performance scalability.

Many-core architectures such as Graphics Processing Units (GPUs) offer significantly greater levels of parallelism than multi-core architectures, and significantly greater levels of raw compute performance. To access the high levels of performance, algorithms and data structures must expose high levels of parallelism and enable good memory-access patterns. Typically fine-grained data-parallelism is used, where the same algorithms are applied to relatively small individual units of data.

To demonstrate the simulation performance improvements which could be achieved by using many-core GPUs for microscopic road network simulations, a single-lane transport model with stop-sign-based yellow-box junctions has been implemented using FLAME GPU (Flexible Large Scale Agent Modelling Environment for Graphics Processing Units). FLAME GPU is the only general-purpose GPU-accelerated ABM framework [6]. The performance of the simulation is benchmarked using a procedurally generated artificial grid-based road network, and the simulation performance is compared to a high-performance multi-core CPU microscopic road network simulation tool, Aimsun 8.1, which implements the same models.

This paper presents two contributions:(i) a GPU accelerated data-parallel agent-based road network microsimulation model is evaluated against an equivalent model in a commercial multi-core CPU software tool, demonstrating considerable improvements to simulation performance and performance scalability; and (ii) a general-purpose graph-based communication strategy is presented for high performance agent communication for fine-grained data-parallel agent based simulations, implemented for the FLAME GPU ABM framework which enables high performance agent based simulations of transport networks on GPUs.

Section 2 provides a summary of related work. Section 3 details the implemented model, the benchmark network used and the FLAME GPU implementation. Section 4 describes the graph-based communication strategy to enable the high levels of performance in this model. Section 5 provides the results of a set of application benchmarks used to assess the performance impact of GPUs on microscopic road network simulation using ABM. Section 6 concludes the paper.

2. Related work

Microscopic simulation of transport simulation involves the simulation of individual vehicles and pieces of road network infrastructure to predict the effects of changes in vehicle behaviour or changes in the road network infrastructure. This is used as a tool in the planning and management of transport networks to improve the effectiveness of infrastructure and minimise the impact of changes in conditions on the transportation network. Microscopic simulations are computationally expensive, which has limited the adoption of microsimulation compared to higher level mesoscopic and macroscopic simulations [5]. A micro-scale simulation must include behavioural models for vehicles to follow as well as models for any dynamic infrastructure such as traffic signals and vehicle detectors, which attempt to accurately capture the behaviour of the real world. ABM is a powerful approach to defining microscopic models, which provides a natural method of describing individual behavioural models and then simulating the interaction between individuals in the simulation [7]. Some of the most important vehicle behavioural models are: (i) Car Following Models [8,9]; (ii) Lane Changing Behaviour [10,11]; and (iii) Gap Acceptance Modelling [1,12].

These behavioural models make use of several sets of input data, including: the attributes and structure of the transport network; transport demand which is used to populate the simulation and allow alternate scenarios to be simulated for predictive use; and simulation parameters which are used to modify the models within the simulator. Parameters can include distributions from which vehicle properties can be sampled, such as vehicle length, rate of acceleration or even properties such as pollution emission rates. These parameters can be manipulated to replicate observed behaviour [13,14].

Leading commercial software packages such as Aimsun and Vissim use multi-core CPU architectures to increase simulation performance through task-parallelism and coarse-grained data-parallelism [15,16], reducing the time required for simulations to execute. The work-load of the simulation is distributed as individual tasks or coarse-grained units of data across the available processing hardware, but, as with many task-parallel and coarse-grained data-parallel multi-core software applications, the performance improvement from each additional processing core reduces as the number of cores and threads is increased. Fig. 1 shows the application run-time of Aimsun 8.1 for a simulation containing approximately $25,000$ concurrent vehicles as the simulation thread count is increased for two multi-core CPU systems. The diminishing returns of additional processor threads are shown, with no significant increases in performance observed beyond six threads. This limits the performance scalability of the application, with large simulations requiring considerable amounts of time to execute even using CPUs with high numbers of processing threads.

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002

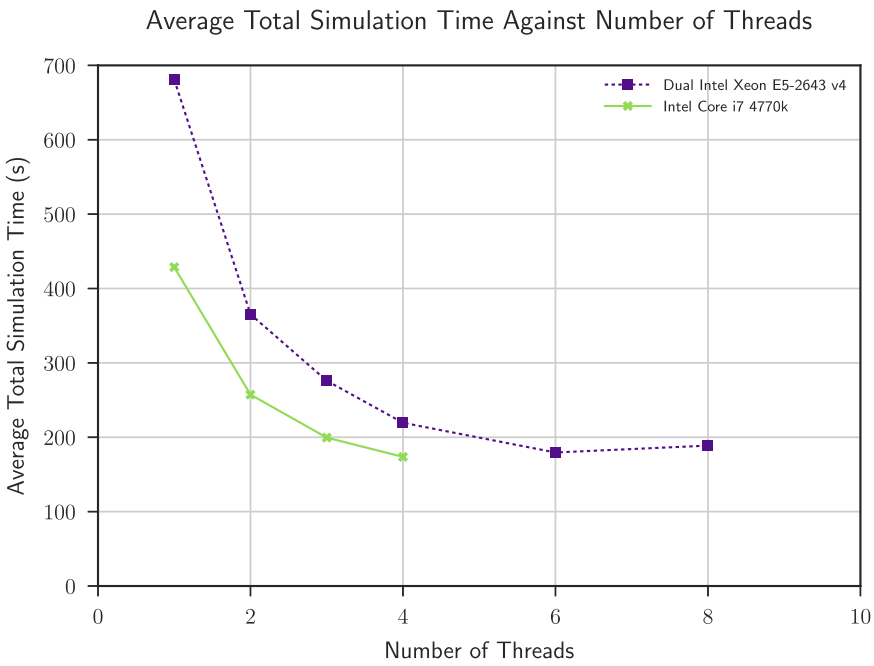

Fig. 1. The average run-time from 3 repetitions for a 60 minute Aimsun 8.1 simulation containing approximately 25,000 vehicles, for different processor thread counts. As processor thread count increases, total simulation time decreases, but with diminishing returns. Note that the Intel Xeon E5-2643 is a dual-socket system. Additionally, no performance improvements were observed when using more processing threads than physical cores (Hyper-threading).

In contrast to multi-core processor architectures, highly parallel many-core processing architectures such as GPUs enable the use of fine-grained data-parallelism to offer high levels of performance. Originally developed for 2D and 3D computer graphics applications, GPUs are now commonly used to accelerate general purpose computing through data-parallelism, as they offer significant computational power. For instance, a single state of the art NVIDIA P100 GPU has a peak theoretical performance of 5.3 TFLOPS (trillion floating point operations per second) in double precision [17], compared to Intel Broadwell (22 core) Xeon CPUs which are capable of less than 1.0 TFLOPS [18], with even greater levels of raw compute performance for single and half precision floating point operations. From a device perspective, GPUs use a form of Single Instruction, Many Data (SIMD) parallel architecture, where the same instructions are applied to many items of data concurrently. Additionally, a large portion of the GPU die space is used for the many processing cores, leaving a relatively small portion of die space available for on-chip cache so to achieve high levels of performance the use of memory must be carefully considered. Along with data access patterns, the SIMD model used by GPUs often requires alternate algorithms and data structures to software designed for multi-core CPUs, in order to leverage the high levels of performance required. Additionally, GPUs are typically more energy-efficient when compared to CPUs, offering greater levels of performance for the same power usage, i.e. GPUs typically offer a higher number of FLOPS per watt. This is demonstrated by the high proportion of GPU enabled super-computers in the upper ranks of the Green 500, the ranking of the world's most power-efficient super-computers [19].

Although current commercial and open source microscopic road network simulators such as Vissim or Aimsun use multi-core CPU parallelism to increase simulator performance, GPUs have yet to be adopted within transport modelling software as a whole. GPUs have been used previously in microscopic transport network simulation, as both cellular automata [20] and also using multi-agent systems [21]. Additionally other tasks associated with microscopic simulation have been accelerated using GPUs, such as the generation of demand data using activity plan generation [22], traffic signal optimisation [23] and dynamic route assignment [24]. Recently, outside of microscopic simulation, GPUs have been utilised within both mesoscopic and microscopic road network simulation tools and performance improvements have been demonstrated [25–27], further highlighting the demand for reduced software run-times within the transport modelling sector.

Several ABM frameworks exist which use distributed and parallel computing to provide increased levels of simulation performance, whilst abstracting the complexities of the parallel computing paradigm away from the end user. D-Mason [28], Repast HPC [29] and FLAME [30] use distributed or CPU-based task-parallelism or coarse-grained data-parallelism to offer improved performance compared to single threaded applications. This is achieved through the distribution of simulation work-load across the available processors or cores, as in current commercial microscopic road network simulators such as Aimsun. In contrast, FLAME GPU is a GPU accelerated ABM framework. FLAME GPU uses the CUDA (Compute Unified Device Architecture) API for NVIDIA GPUs and a state-based representation of agent dynamics to leverage the high levels of performance from the GPU. The complexities of the CUDA programming model are abstracted away from the end-user,

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002

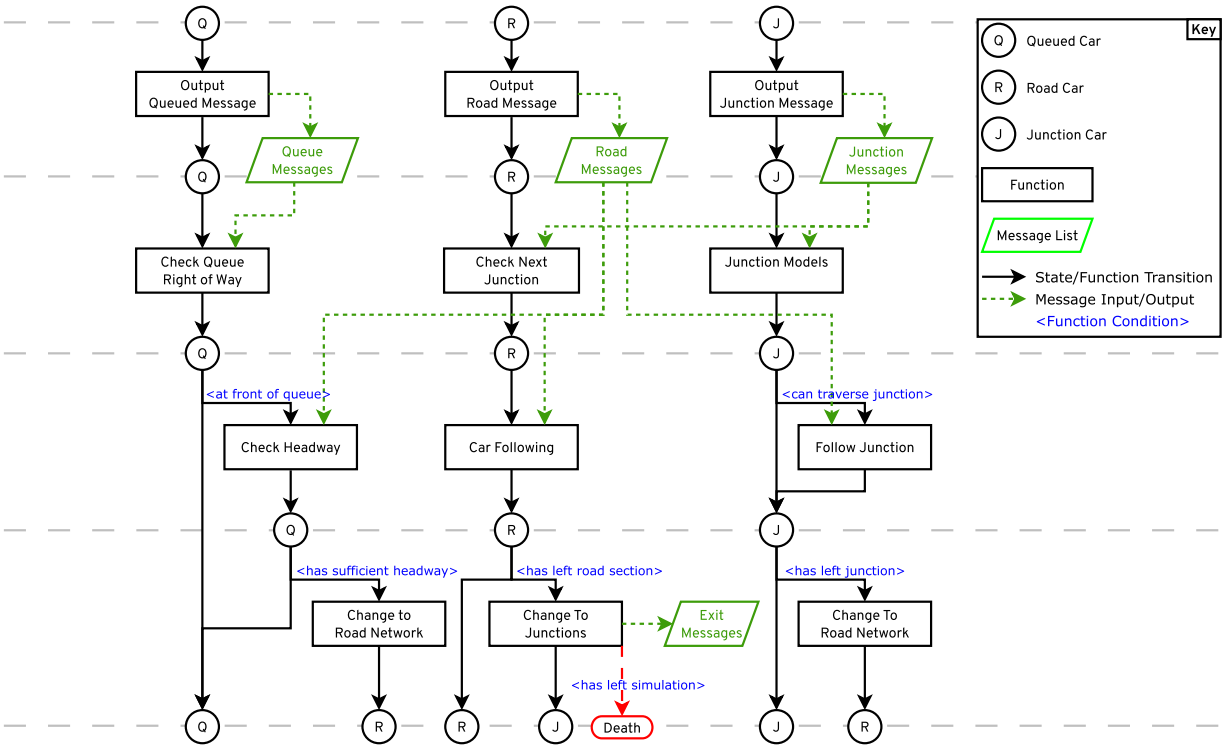

Fig. 2. FLAME GPU State-based representation for vehicle agents within the simulation, illustrating the logical control-flow of the agents within a single iteration. Interaction with detector agents has been omitted for clarity.

enabling high performance simulations without explicit knowledge of the parallel processing model [6,31,32]. FLAME GPU has successfully been used in several modelling domains such as pedestrian crowd simulation [33] and cellular simulations [31,34], however the use of FLAME GPU for road network simulation has been limited, due to the unsuitability of fixed radius communication for use within transport networks [35].

3. Mulit-agent GPU microsimulation

To evaluate the potential performance impact of GPU accelerated ABM on microscopic road network simulation a single-lane road network model was implemented. The implemented models include: car following model [1,15,36] (based on Gipps' car following model[8]); constant vehicle arrival [15,36]; virtual queues [15,36]; detectors [15,36]; a yellow-box model [15,36] and a gap acceptance give-way model [1,15,36] combined with stop-signs for junctions. These features and models were selected to produce comparable behaviour for the procedurally generated grid road network described in Section 5.3, without the complexities of modelling the broad range of road network infrastructure present in real world road networks.

FLAME GPU simulations make use of a state-based representation of agent dynamics to simplify development and enable key performance optimisations on behalf of the modeller. Fig. 2 shows the state machine for the vehicular agents used to implement this model; it represents the logical process followed by each agent at each iteration of the simulation, showing the relationship between agent states, functions and message-lists. Agents enter the state machine for the current iteration in one of the possible initial states. Agent functions allow agents to transition from one state to another, with the order of these functions controlled by *execution layers*. Communication between agents is in the form of *message lists*, which are used to highlight dependencies in the state-based representation. The use of indirect communication prevents race conditions in the parallel processing environment. Agent functions can output messages to the relevant message list data structure, which is then used as input by a different agent function. The message lists are accessed using specialised message access patterns to reduce the performance impact of message list parsing, a new graph-based communication pattern is used for agent to agent communication, as described in Section 4.

For instance, in the first execution layer, the agents in each of the three possible states for vehicles output a message to a message list. Those message lists are then iterated by the different agent states in layers two and three, dependant upon the agent state. Vehicle agents which are deemed to have left the simulation are identified as *dead* and subsequently excluded from further iterations of the simulation.

The *Queued* state of *Car* agents is used to store the virtual queue of vehicles entering the simulation. New agents generated at each iteration of the simulation are initially places in the Queued state. The series of agent functions which use this

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002

state implement the constant or exponential vehicle arrival algorithm from Aimsun [15], ensuring that the same conditions required for an agent to be inserted into the simulation such as order of arrival and sufficient headway are met.

The *Road* and *Junction* states are used to represent vehicles which use the two graphs used to implement the road network data structure, with vehicles in the Road state exhibiting primarily the car following behavioural model, and interaction with stop signs. Agents in the Junction state execute the gap acceptance give-way model and yellow-box model used for this study, and subsequently if movement is allowed the vehicle will use the car following model to traverse the junction.

Additional to the state machine for cars, a linear state machine exists for *detector* agents. Detector Agents represent real-world vehicle detection systems such as induction loops placed within road network infrastructure. These detection loops are used to measure traffic conditions, the output of which can be used for calibration, validation or interactive behaviour. The detector state machine is linear, consisting of four agent functions in separate layers. These functions each iterate one of the message lists produced by the vehicle agents, and capture any statistics relevant to the individual detector for the current interaction.

The transport network is represented using a pair of graphs, represented using the Compressed Sparse Row (CSR) data structure. The first graph represents the roads of the transport network as edges, and the junctions as vertices. The second graph models the connections between roads within each junction. Separate graphs are used to simplify the specification of the road network.

4. Network communication

Within FLAME GPU, communication between agents uses message-list traversal algorithms providing specialised access patterns to enable efficient, indirect, data-parallel agent communication. The algorithms and data-structures used can have considerable impact on the performance of a model, and the development of specialised high-performance access to messages is vital to achieve high levels of performance for large-scale ABMs. This section describes a new specialisation for graph-based messages within FLAME GPU, which enables high performance for network-based communication patterns. The performance of the communication strategy is evaluated compared to existing FLAME GPU communication patterns.

4.1. FLAME GPU message-based communication

The communication method used in a FLAME GPU model can be specialised to offer increased levels of performance for different communication patterns. Existing techniques for message-based communication available within FLAME GPU allow global communication (all-to-all) or partitioned communication based on spatial locality [6]. All-to-all message specialisation enables indirect global communication between agents, while spatially partitioned messaging restricts the communication to the immediate region surrounding the agent in 2D or 3D space. These communication patterns are non-optimal for several core vehicle models for road network microsimulation, where the communication between agents is restricted to the road network infrastructure. Typically models such as car following and lane changing models only require communication with other agents on the same road section or directly connected road sections within the transport network. Other models such as give way modelling for junctions may require communication within the junction and with other vehicles on the connected road sections. In these cases both all-to-all or spatially partitioned communication strategies can result in the iteration of large message lists to find the relevant message or messages. Spatially partitioned communication would offer an improvement over all-to-all communication, however in areas with a large number of agents, or in dense sections of road network such as urban city centres, the number of messages each agent must iterate can still be significant for even relatively small communication radii, resulting in a negative impact on performance [35].

4.2. Graph-based messaging

Ideally communication should only need to occur within the constraints of the network, which is typically represented using a graph or series of graphs. For the purposes of this benchmark model, which consists of single lane sections of roads and stop-sign-based, yellow-box junctions, communication is limited to the current road network edge, the next road network edge, or the current junction (graph vertex). Existing techniques for graph-based communication exist within parallel and distributed agent based frameworks such as D-Mason and Repast HPC, however these strategies use individual agents as the vertices of the graph and the relationships between agents are the edges of the graph [29,37].

Individual agents are coupled with the graph, maintaining their location within the graph at all times. For a network-based communication strategy, the messages are also coupled to the underlying graph. When an agent outputs a message, it must include the edge or vertex for which the message is coupled. The message list of all messages can then be sorted by the edge or vertex to align memory and a sparse data structure is constructed to enable access to relevant messages of the data structure. The data-structure construction is similar to that used for fixed radius communication [32] commonly applied to many particle-based GPU simulations [38], which identifies the position within the message list of the first message for each edge or vertex in the network, and the number of messages for each edge or vertex. The sorting of the message list and construction of the message delimiting data structure can be performed efficiently using a *counting sort*. High performance implementation of counting sort is reliant upon GPU fundamentals like cache-aware atomic operations. As the number of edges or vertices within the network are known, the counting sort constructs a histogram of the messages for each edge or

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002

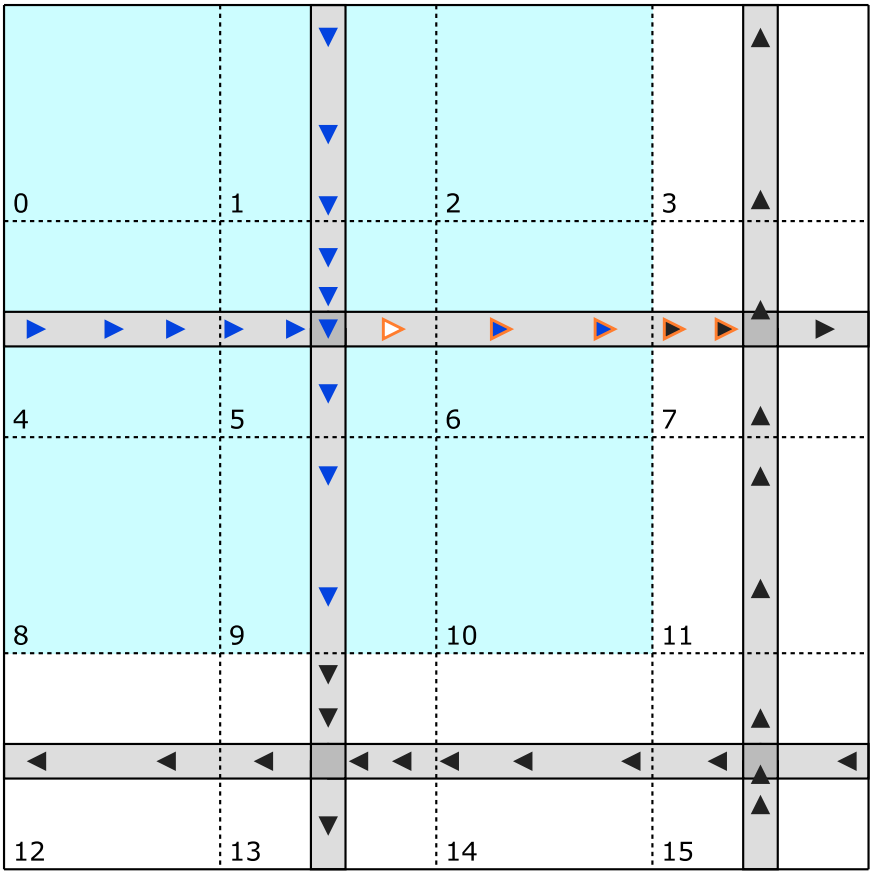

Fig. 3. The effects of alternate FLAME GPU communication strategies on car following models is shown for a section of a grid-based road network, for a single agent represented in white. All-to-all communication results in 42 messages being parsed by the single agent. Spatially partitioned messaging results in the 18 messages from the agents in the blue shaded region, while the graph-based communication strategy results in only 5 messages from the orange bordered agents being processed. The spatially partitioned radius used for this illustration would be insufficient for accurate modelling, and is only used for illustrative purposes.

vertex within the network. A scan can then be performed on the histogram to provide the output position for the messages, allowing messages to be sorted in parallel. The histogram which is constructed during the sorting algorithm can also be used as a part of the data structure for message access. The network-based communication strategy is general enough to be applied to any GPU multi-agent simulation, where communication along a graph-based data structure is required, such as social network modelling.

4.3. Application to road network microsimulation

Fig. 3 illustrates the impact of the FLAME GPU message specialisation techniques on vehicle behavioural models such as the car following model. The figure shows a portion of a grid-based transport network, with grey rectangles used to represent road sections and coloured triangles to represent individual vehicle agents. The colour of the triangle shows the messages processed by a single agent, shown in white in cell 5, for each message partitioning technique. Using all-to-all messaging, the agent will process a total of 42 messages, from each of the agents in the simulated region. Using spatially partitioned messaging the target agent will process messages from the 18 agents in the shaded region of 9 cells, with agents highlighted in blue. For the proposed graph-based communication strategy, the target agent will only process a list of 5 messages, for the vehicles on the same section of road as the target agent. These agents are highlighted with orange borders. This reduction in the number of messages parsed by individual vehicles leads to considerable performance improvements for this type of model, with even greater reductions in the number of messages to be processed for denser areas of transport networks when using the graph-based communication strategy.

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002

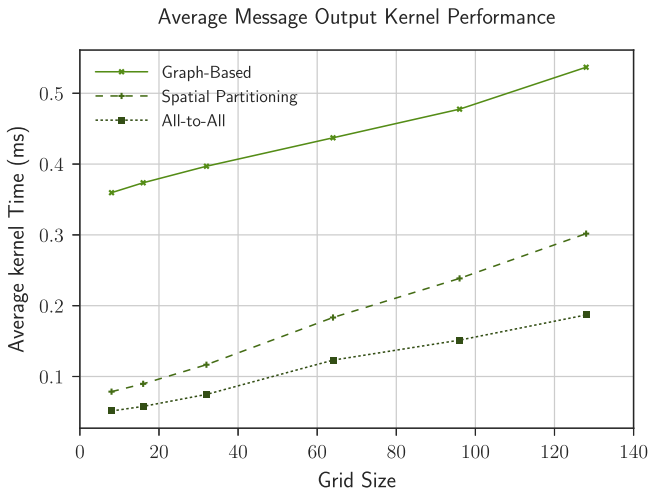

Fig. 4. The average time taken for vehicle agents to output road messages are shown for three communication strategies and different scales of model. This shows the overhead cost of each communication specialisation. The application was benchmarked on a Nvidia Titan X (Pascal).

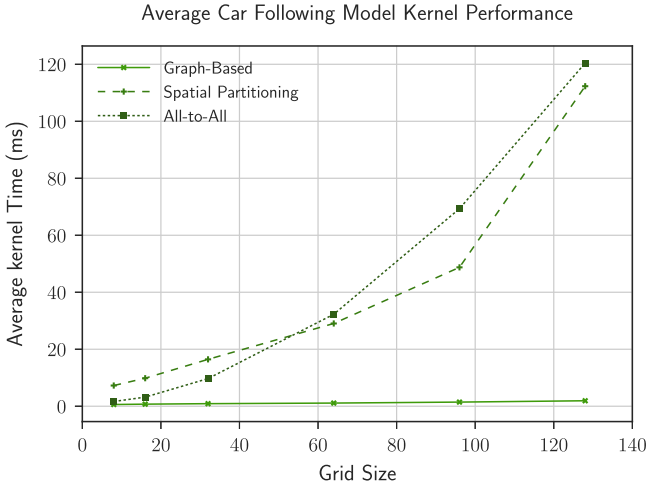

Fig. 5. The average time taken for vehicle agents to execute the GPU kernel which includes the car following model is shown for three communication strategies and different scales of model. This kernel iterates all messages in the appropriate list to find the lead vehicle. Graph-based communication shows the highest level of performance. The application was benchmarked on a Nvidia Titan X (Pascal).

4.4. Performance impact of the communication strategy on the car following model

To evaluate the performance improvement provided by graph-based communication for agent-based road network models, a set of road networks were benchmarked using three communication strategies. The message list output by vehicles travelling along the road sections within the road network were specialised using all-to-all, spatially partitioned and graph-based messaging. The performance of the relevant GPU kernels is extracted and compared.

Fig. 4 shows the average performance of the process for outputting the messages and constructing the relevant data structures. This shows that all-to-all communication has the smallest message-output overhead cost, whilst graph-based communication has the highest overhead-cost. Fig. 5 shows the average performance of the GPU kernel which implements Gipps' car following model for several sizes of procedurally generated road network. The figure shows that graph-based communication provides greater levels of performance than both spatially partitioned messaging and all-to-all communication. The combined effect on performance from graph-based communication provides a significant improvement on performance, as the additional performance provided by the reduction in message-list iteration far outweighs the overhead cost of message output and data structure construction.

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002

Table 1
Computational hardware used for benchmarking Aimsun and FLAME GPU simulations.

Identifier	CPU	CPU cores	CPU frequency	Memory	GPU
System #1	Intel core i7 4770k	4	3.5 GHz	16GB DDR3	NVIDIA Titan X (Pascal) (TCC) & NVIDIA GeForce GTX 1080 (WDDM)
System #2	2 × Intel Xeon E5 2698 v4	40 (20)	2.20 GHz	512GB DDR4	8 × NVIDIA Tesla P100 (linux)

Table 2
Parameters used for validation and all benchmark simulations.

Parameter name	Grid size experiment	Input flow experiment	Units
Time step	0.8	0.8	seconds
Reaction time	0.8	0.8	seconds
Stop reaction time	1.2	1.2	seconds
Detector period	600	600	seconds
Input flow scale factor	1.0	1.0	%
Seed	Random value	Random value	Long integer
Section length	1000	1000	m
Junction length	10	10	m
Grid size	2–576	64, 128 & 256	
Input flow	600	100–1000	
Detectors per section	3	3	

5. Experimental results

This section summarises the cross-validation of the FLAME GPU based simulation against the existing multi-core CPU road network microsimulation tool Aimsun. It provides details of the procedurally generated Manhattan style grid road network used for performance benchmarking and two sets of benchmark experiments are discussed.

5.1. Validation

For the performance evaluation of the two simulators to be comparable it is important that software produces comparable results. As the complex system is stochastic the results will need to be statistically equivalent. The FLAME GPU implementation was cross-validated against Aimsun 8.1 for a series of models to validate model behaviours, such as the vehicle arrival algorithm [15], car following [15] and stop sign interaction [15]. Additionally the overall behaviours and population sizes for the procedurally generated road network are evaluated. The same set of parameters are used for both simulators. The set of validation models showed that vehicle parameter sampling, the vehicle arrival algorithm, the car following model and the junction models were all correctly implemented, with no deviance observed. The procedurally generated networks showed statistically similar population sizes and turning behaviours over an average of multiple replications.

5.2. Benchmarking methodology

Multiple configurations of processing hardware were used to observe the performance impact of the hardware on performance. Table 1 describes the two hardware systems used for these experiments. The Aimsun benchmark models were executed using System #1, a desktop computer. FLAME GPU simulations were carried out using a range of GPUs in System #1, and also on an NVIDIA DGX-1 (System #2), a state-of-the-art multi-GPU server containing NVIDIA Tesla P100 accelerators. All simulations were executed using a single CPU and single GPU and the same parameters (Table 2) were used for both simulations of one hour of vehicle demand.

5.3. Procedurally generated benchmark road network

To benchmark the scalability of performance of both simulators for different sizes of road network, a procedurally generated artificial road network was designed, in the style of a Manhattan grid road network. The artificial network is a 2D grid of single-lane one-way roads, where adjacent rows and columns of the road network are in alternate directions. For each yellow-box junction in the network there are four possible turning sections, each preceded by a stop sign. Fig. 6 shows the arrangement of road sections and junctions, and also the arrangement of turns from one section of road to another within a junction. Three detectors were placed on each section of road. The numbers of sections, junctions and turns in the grid can be calculated based on the *network grid size* parameter. A network with *grid size* (G) contains G^2 junctions, $2G(G+1)$ road sections, $4G^2$ turning sections, $2G$ entrances and $6G(G+1)$ detectors. Table 3 details the parameters involved in network generation.

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002

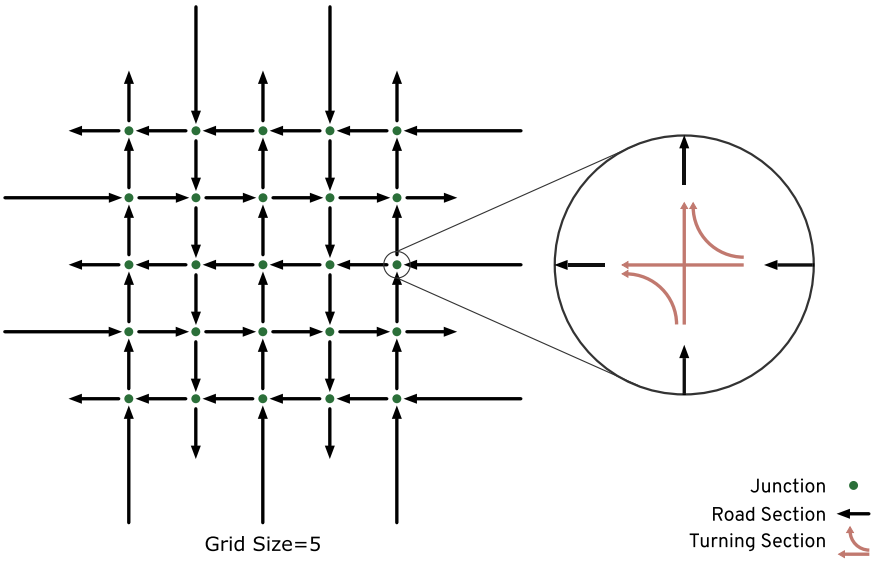

Fig. 6. A 5 × 5 example of the procedurally-generated artificial grid network, showing the overall structure of the network and the arrangement of turning sections within a junction. The network can be scaled to any size, with networks of up to 576 × 576 used during benchmarking.

Table 3
Parameters for generation of the procedurally generated Manhattan-style grid road network.

Parameter Name	Description
Grid size (G)	The number of junctions for each row and column of the grid
Section length	The length of each road section between junctions in metres
Junction length	The length or diameter of junctions within the network
Exit turn proportion	For junctions with 1 exit destination edge, the proportion of vehicles turning onto the exit is reduced to maintain higher vehicle populations within the simulated road network.

5.4. Network scale experiment

To evaluate the performance impact of total problem scale, the *Grid Size* of the procedurally generated network was varied from 2 to 576. Aimsun simulations were only carried out up to a grid size of 512 due to the time taken for a single simulation to complete. As the grid size is increased, the size of the transport network increases, the number of detectors agents in the simulation increases and the vehicle population size increases. Fig. 7 show the average simulation time for 3 repetitions of the simulation. This shows that for larger scale problems the GPU accelerated FLAME GPU simulation outperforms Aimsun 8.1, by up to 37.2 × using *System #1* and 43.8 × using *System #2*. The Real time ratio (RTR) of a simulation is the ratio of execution run-time to simulated time. The largest simulation executed in FLAME GPU of 576,000 vehicles and 1,994,112 detectors shows a RTR of 10.5 × using a NVIDIA GeForce GTX 1080 in *system #1*, 16.6 × using *system #1* NVIDIA Titan X (Pascal) and 25.5 × using the NVIDIA Tesla P100 in *system #2*. For very small simulations, with *grid size* 8 and below containing less than 7396 concurrent vehicles, Aimsun 8.1 shows greater levels of performance than FLAME GPU.

5.5. Input flow experiment

Demand on transportation networks is increasing globally [39]. To evaluate the impact of this on microscopic transport network simulation, the demand on the transport network was varied for several fixed sizes of network. Due to the characteristics of our procedurally generated transport network, the vehicle population size increases non-linearly, as vehicles can only enter from the edges of the network. The model parameters used are defined in Table 2.

Figs. 8–10 show the average simulation time for 3 repetitions of each simulation. The simulations for a grid size of 64, FLAME GPU shows average speed-ups of between 3.8 × and 18.9 × compared to Aimsun 8.1. Speed-ups of between 9.3 × and 36.6 × are shown for a grid size of 128, and between 8.1 × and 75.3 × for a grid size of 256.

5.5.1. Discussion of application benchmark results

The simulation results show that the FLAME GPU implementation shows significantly higher levels of performance and scalability for all but very small simulations. Small models with low population sizes do not expose high levels of parallelism, limiting the performance of FLAME GPU, as the highly-parallel GPU is underutilised. That means there are not enough threads to occupy the GPUs many processing cores. As the population of vehicles and detectors increases the level

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002

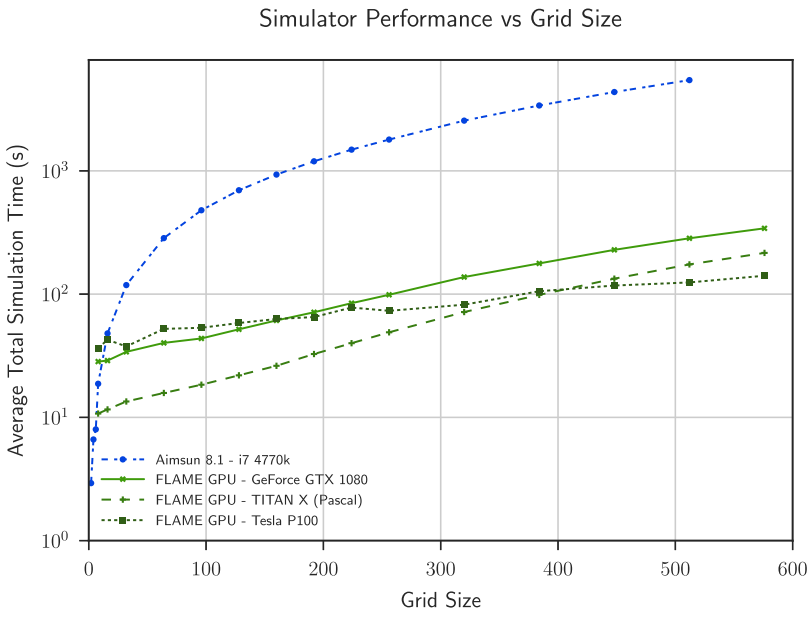

Fig. 7. Total simulation performance as the total scale of the simulation is increased, for each simulator. Values shown are the average from 3 repetitions. A logarithmic scale is used to improve visibility of similar total simulation times.

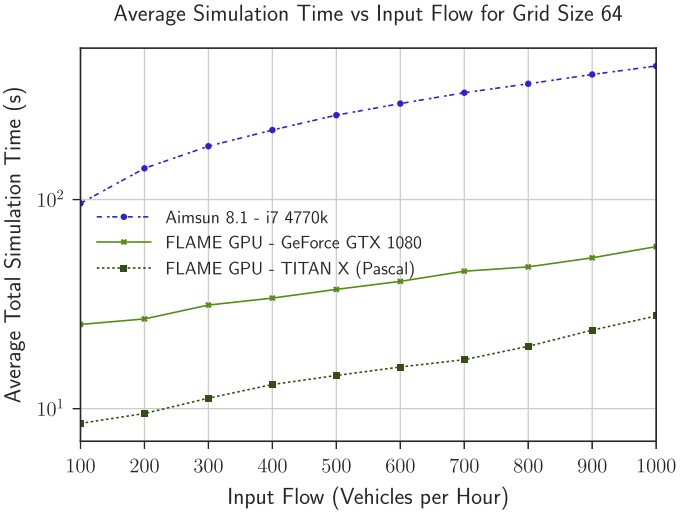

Fig. 8. Total simulation time against input flow for a procedurally generated road network of grid size 64. Run times shown are the average from 3 repetitions. A logarithmic scale is used.

of parallelism and therefore GPU utilisation increases. Additionally, GPU acceleration has associated overhead costs. Data and commands must be transferred from the host computer to the GPU on which code is executed. This is a relatively slow process which must be minimised where possible to achieve high levels of performance. Larger models quickly amortize this cost through performance gains.

The performance of individual simulation iterations, shown in Fig. 11 shows that per-iteration simulation time increases as the simulation progresses. This is due to the increasing number of vehicles within the simulation. The rate of increase of per-iteration simulation time grows at a higher rate within Aimsun than FLAME GPU, demonstrating the increased scalability of the GPU accelerated simulation with respect to population size. For both Aimsun and FLAME GPU the time to simulate a single iteration peaks every 750 iterations. This can be accounted for by the detector and statistical summary data being processed and written out to disk every 600 simulated seconds (with a simulation step of 0.8 s).

Each GPU used within the FLAME GPU benchmarks showed varying levels of performance. This can be attributed to the varying hardware configurations, shown in Table 4, and also the relevant driver model in use (see Table 1). For smaller grid size simulations and smaller input flow values, all GPUs showed similar performance. For larger scale simulations with

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002

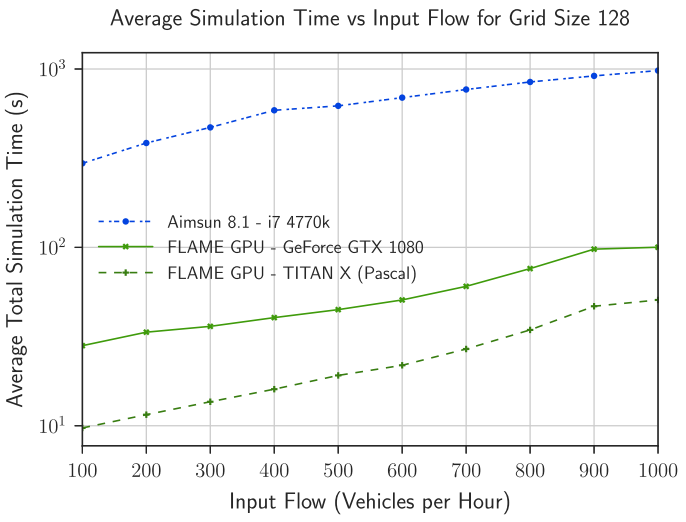

Fig. 9. Total simulation time against input flow for a procedurally generated road network of grid size 128. Run times shown are the average from 3 repetitions. A logarithmic scale is used.

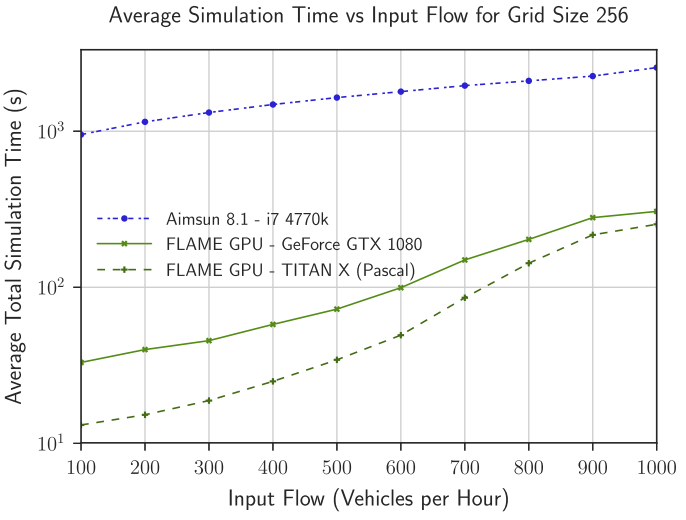

Fig. 10. Total simulation time against input flow for a procedurally generated road network of grid size 256. Run times shown are the average from 3 repetitions. A logarithmic scale is used.

Table 4
GPU specifications [40–42].

GPU	Core count	Core frequency (MHz)	Memory	Memory bandwidth (GB/s)
NVIDIA Geforce GTX 1080	2560	1607 (1733)	8GB GDDR5X	320
NVIDIA Titan X (Pascal)	3584	1417 (1531)	12GB GDDR5X	480
NVIDIA Tesla P100 (SXM2)	3584	1328 (1480)	16GB CoWoS HBM2	732

high populations of agents, the NVIDIA Tesla P100 in *System #2* showed the greatest levels of performance. This can be attributed to the larger number of processing cores, and the greater levels of memory bandwidth provided by the HBM2 memory, compared to GDDR5X and GDDR5. The CPUs in *System #1* & *#2* also differ, however the majority of processing time within the application is executed on the GPU.

6. Conclusion and future work

The GPU accelerated agent based implementation of a single-lane road network microscopic simulation demonstrates considerable performance advantages and improved performance scaling behaviour when compared to an equivalent multi-

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002

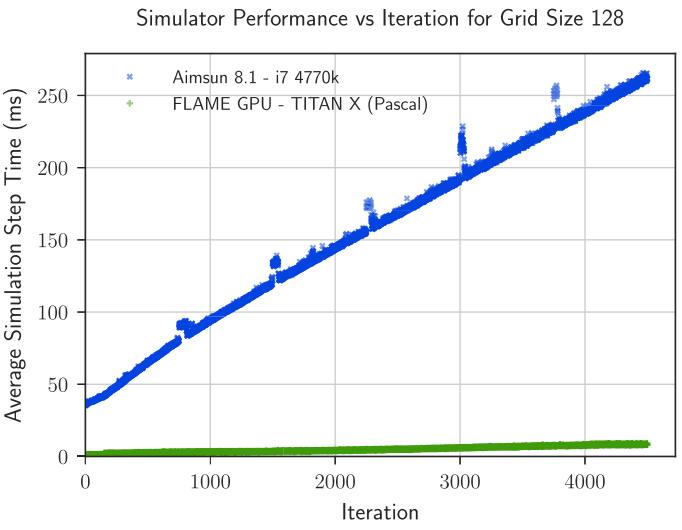

Fig. 11. Per-Iteration simulation time for a model with grid size 128 and input flow 500 vehicles per hour.

core CPU based simulation software tool commonly used within the transport planning industry. To ensure that simulator behaviour is comparable and enable simulator performance to be compared the FLAME GPU based simulation was cross validated against Aimsun using a series of models. As this is a stochastic model and additional features are present in Aimsun which could not be fully mitigated the results were non-exact but were statistically similar. The largest simulation executed using both simulators, a one hour simulation containing up to 512,000 vehicles and 1,575,936 detectors, showed a speed-up of 43.8 $\times$ for the GPU accelerated simulation, with a real-time-ratio of 28.9. The GPU accelerated simulation also demonstrated greatly improved performance scaling behaviour, with the largest simulation of up to 576,000 vehicles and 1,994,112 detectors completing in 49.5% of the time of a much smaller (up to 64,000 vehicles and 24,960 detectors) simulation in Aimsun 8.1.

Critical to the performance of the GPU accelerated agent based microsimulation was the use of data parallel algorithms and data structures employed by FLAME GPU for agent management, abstracting the complexities of the GPU programming model away from the modeller. The proposed graph-based communication strategy for data-parallel agents is also vitally important, as existing methods for agent communication within FLAME GPU were non-optimal, limiting simulation performance. The proposed graph based methods also have more general implications, as other models such as social interaction models rely on graph-based communication. The increased memory-bandwidth of the NVIDIA Tesla P100 GPU, compared to the NVIDIA Titan X (Pascal) and NVIDIA GeForce GTX 1080 proved to have a dramatic increase on the performance of the application, as the application is memory bound.

6.1. Future work

Our FLAME GPU based simulator currently supports a single type of junction to support the benchmark model used. To enable this software to be used for real-world studies of transport systems, additional features must be implemented, such as the inclusion of multiple lanes, traffic signals and alternative types of junction. The impact of these additional behaviours would require further performance evaluation using procedurally generated and real-world transport networks.

The proposed graph-based communication strategy is currently limited to communication along the current edge, or around a single vertex within the graph. Although this can be used to achieve significant performance improvements the strategy is to offer high performance communication for multiple-edges for graph exploration. This would aid in in the wider applicability of the general communication strategy and offer performance improvements to other simulation domains. Furthermore, the existing agent based simulation uses multiple graphs to represent the transport network, combining these graphs into a single graph would simplify the development of additional models, and allow a more-generalised communication strategy to yield further performance increases.

Lastly the benchmark model used in this study imposed limits upon the density of vehicles within the network, as a result of the non linear relationship between vehicle input edges and the total edge count as the network grid size is increased. Future studies into the performance of simulators for higher-density networks (such as urban city centres) will provide further insight into the benefits of many-core processing architectures for agent based road network microsimulation, but would also require the proposed implementation of a broader range of features.

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002

Acknowledgement

This work was supported by a Department for Transport Transport Technology Research Innovation Grant "Accelerating Transport Microsimulation: Demonstrating the impact of future many core simulations" (T-TRIG July 2016) and additional support through EPSRC fellowship "Accelerating Scientific Discovery with Accelerated Computing" (EP/N018869/1).

References

[1] J. Barceló, J. Casas, Dynamic network simulation with Aimsun, in: Simulation Approaches in Transportation Analysis, Springer, 2005, pp. 57–98.
[2] M. Fellendorf, Vissim: a microscopic simulation tool to evaluate actuated signal control including bus priority, in: 64th Institute of Transportation Engineers Annual Meeting, Springer, 1994, pp. 1–9.
[3] M. Balmer, M. Rieser, K. Meister, D. Charypar, N. Lefebvre, K. Nagel, Matsim-t: architecture and simulation times, in: Multi-agent Systems for Traffic and Transportation Engineering, IGI Global, 2009, pp. 57–78.
[4] D. Krajzewicz, J. Erdmann, M. Behrisch, L. Bieker, Recent development and applications of SUMO - simulation of urban mobility, Int. J. Adv. Syst. Measur. 5 (3&4) (2012) 128–138.
[5] C. Antoniou, J. Barcelò, M. Brackstone, H. Celikoglu, B. Ciuffo, V. Punzo, P. Sykes, T. Toledo, P. Vortisch, P. Wagner, Traffic simulation: case for guidelines, 2014 URL http://publications.jrc.ec.europa.eu/repository/bitstream/JRC88526/2014_multitude_guidelines_on-line.pdf, doi:10.2788/11382.
[6] P. Richmond, FLAME GPU Technical Report and User Guide (CS-11-03), Technical Report, Department of Computer Science, University of Sheffield, 2011.
[7] G. Eliasson, Modeling the experimentally organized economy, 1991. doi:10.1016/0167-2681(91)90047-2.
[8] P. Gipps, A behavioural car-following model for computer simulation, Transp. Res. Part B 15 (2) (1981) 105–111, doi:10.1016/0191-2615(81)90037-0.
[9] M. Treiber, A. Hennecke, D. Helbing, Congested traffic states in empirical observations and microscopic simulations, Phys. Rev. E 62 (2) (2000) 1805.
[10] P. Gipps, A model for the structure of lane-changing decisions, Transp. Res. Part B 20 (5) (1986) URL http://www.sciencedirect.com/science/article/pii/0191261586900123.
[11] M. Brackstone, M. McDonald, J. Wu, Lane changing on the motorway: factors affecting its occurrence, and their implications, in: Road Transport Information and Control, 1998. 9th International Conference on (Conf. Publ. No. 454), IET, 1998, pp. 160–164.
[12] R. Liu, D. Van Vliet, D. Watling, Microsimulation models incorporating both demand and supply dynamics, Transp. Res. Part A 40 (2) (2006) 125–150.
[13] R. Dowling, A. Skabardonis, J. Halkias, G. McHale, G. Zammit, Guidelines for calibration of microsimulation models: framework and applications, Transp. Res. Rec. (1876) (2004) 1–9.
[14] S.-J. Kim, W. Kim, L. Rilett, Calibration of microsimulation models using nonparametric statistical techniques, Transp Res Rec. (1935) (2005) 111–119.
[15] Transport Simulation Systems, Aimsun 8 Users Manual, 2014.
[16] A. PTV, Vissim 5.40 user manual, Karlsruhe, Germany (2011).
[17] NVIDIA Corporation, NVIDIA Tesla P100 (whitepaper), 2016.
[18] Microway, Detailed specifications of the intel xeon e5-2600v4-broadwell-ep-processors, 2016. URL https://www.microway.com/knowledge-center-articles/detailed-specifications-of-the-intel-xeon-e5-2600v4-broadwell-ep-processors/.
[19] Green 500, Green 500 website, 2016. (https://www.top500.org/green500/). Last Accessed 2016-08-01
[20] M. Zamith, M. Joselli, J.R.S. Junior, R.C.P. Leal-Toledo, E. Clua, I. MediaLab, E. Soluri, N. Tecnologia, An approach for traffic forecast with GPU computing & cellular automata model.
[21] D. Strippgen, K. Nagel, Multi-agent traffic simulation with cuda, in: High Performance Computing & Simulation, 2009. HPCS'09. International Conference on, IEEE, 2009, pp. 106–114.
[22] K. Wang, Z. Shen, A GPU-based parallel genetic algorithm for generating daily activity plans, IEEE Trans. Intell. Transp. Syst. 13 (3) (2012) 1474–1480.
[23] K. Wang, Z. Shen, Artificial societies and GPU-based cloud computing for intelligent transportation management, IEEE Intell. Syst. 26 (4) (2011) 22–28.
[24] Y. Sano, N. Fukuta, A GPU-based framework for large-scale multi-agent traffic simulations, in: Advanced Applied Informatics (IIAIAAI), 2013 IIAI International Conference on, IEEE, 2013, pp. 262–267.
[25] Y. Xu, G. Tan, X. Li, X. Song, Mesoscopic traffic simulation on CPU/GPU, in: Proceedings of the 2nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, ACM, 2014, pp. 39–50.
[26] X. Song, Z. Xie, Y. Xu, G. Tan, W. Tang, J. Bi, X. Li, Supporting real-world network-oriented mesoscopic traffic simulation on gpu, Simul. Modell. Practice Theory 74 (2017) 46–63.
[27] R. Bradley, I. Wright, D. Swain, R. Himlin, P. Heywood, P. Richmond, M. Mawson, G. Fletcher, R. Guichard, Accelerating traffic models using GPU-based technology, in: European Transport Conference 2016 Association for European Transport (AET), 2016.
[28] G. Cordasco, R.D. Chiara, A. Mancuso, D. Mazzeo, V. Scarano, C. Spagnuolo, A framework for distributing agent-based simulations, in: Euro-Par Workshops (1)'11, 2011, pp. 460–470.
[29] N. Collier, M. North, Repast HPC: A Platform for Large-Scale Agent Based Modeling, Wiley, 2011.
[30] L. Chin, D. Worth, C. Greenough, S. Coakley, M. Holcombe, M. Gheorghe, Flame-ii: a redesign of the flexible large-scale agent-based modelling environment, Technical Report, 2012.
[31] P. Richmond, D. Walker, S. Coakley, D. Romano, High performance cellular level agent-based simulation with FLAME for the GPU., Briefings Bioinf. 11 (3) (2010) 334–347 URL http://www.ncbi.nlm.nih.gov/pubmed/20123941, doi:10.1093/bib/bbp073.
[32] P. Richmond, D. Romano, Template-driven agent-based modeling and simulation with cuda, in: GPU Computing Gems Emerald Edition, in: Applications of GPU Computing Series, 2011, pp. 313–324.
[33] T. Karmakharm, P. Richmond, D.M. Romano, Agent-based large scale simulation of pedestrians with adaptive realistic navigation vector fields., TPCG 10 (2010) 67–74.
[34] S. Tamrakar, P. Richmond, R.M. DSouza, Pi-flame: a parallel immune system simulator using the flame graphic processing unit environment, SIMULATION (2016). 0037549716673724
[35] P. Heywood, P. Richmond, S. Maddock, Road network simulation using flame GPU, in: Euro-Par 2015: Parallel Processing Workshops, Springer, 2015, pp. 430–441, doi:10.1007/978-3-319-27308-2.
[36] Transport Simulation Systems, Aimsun 8 Dynamic Simulators Users Manual, 2014.
[37] G. Cordasco, C. Spagnuolo, V. Scarano, Toward the new version of d-mason: efficiency, effectiveness and correctness in parallel and distributed agent-based simulations, in: Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International, IEEE, 2016, pp. 1803–1812.
[38] S. Green, Particle simulation using cuda, NVIDIA Whitepaper 6 (2010) 121–128.
[39] Department for Transport, Road traffic forecasts 2015, 2015, (https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/260700/road-transport-forecasts-2013-extended-version.pdf).
[40] NVIDIA Corporation, Nvidia geforce gtx 1080 whitepaper: Gaming perfected, 2016. URL http://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_1080_Whitepaper_FINAL.pdf.
[41] NVIDIA Corporation, Nvidia titan x (pascal) webpage, 2016. URL http://www.geforce.co.uk/hardware/10series/titan-x/.
[42] NVIDIA Corporation, Nvidia tesla p100 data sheet, 2016. URL http://images.nvidia.com/content/tesla/pdf/nvidia-tesla-p100-datasheet.pdf.

Please cite this article as: P. Heywood et al., Data-parallel agent-based microscopic road network simulation using graphics processing units, Simulation Modelling Practice and Theory (2017), https://doi.org/10.1016/j.simpat.2017.11.002