

# Scaling Transportation Simulations Through Declarative Processing

Alan Demers\* Oliver Gao# Johannes Gehrke\*  
Christoph Koch\* Marcos Vaz Salles\* Walker White\*

\*Cornell Department of Computer Science #Cornell Department of Civil Engineering

## 1 Introduction

As we acquire an increasing amount of information on the physical world, we have the opportunity to develop new methods for analyzing it, and to develop new transportation simulation models that can capture the underlying properties that shape large-scale, long-term processes such as transportation, pollution, and the impact of new technologies or policy decisions on our environment. The interesting challenge here is that much work in science and engineering has developed sophisticated methods for either (1) handling the complex, detailed questions concerned with small or moderate-sized micro-simulations (in the tens, hundreds, or small number of thousands of simulated entities), or (2) handling questions at the aggregate level concerned with macro-simulations where higher-level concepts such as flows abstract away individual simulated entities (SEs), enabling simulations of much larger phenomena. Historically, scaling up micro-simulations to address detailed questions about large-scale transportation systems, when feasible at all, has required supercomputer-class facilities and much specialized hand-optimization of the code.

Our research makes a first step towards tackling the grand challenge of truly scalable transportation micro-simulations. We take the computational ideas that allow database systems to scale to petabytes and apply them to the programming and processing of large-scale transportation micro-simulations. Thus, the use of our computer science techniques will enable detailed simulations on a standard cloud computing infrastructure (a shared nothing cluster) with SEs numbering in the tens of thousands to millions. In particular, we will enable scalable *declarative* processing of such transportation simulations, which will enable us to use optimizations from databases such as set-at-a-time processing, indexing, and parallel processing. Our declarative *processing* model does not come with a declarative *programming* model, but instead allows transportation simulation designers to stay in a traditional imperative programming model as long as they follow certain high-

level program design patterns. Thus we use computational ways of thinking about transportation simulations that are only subtly different from existing ways of programming transportation simulations, but that enable researchers to perform transportation micro-simulations at scales that were never imagined before.

## 2 Transportation Simulations

Transportation simulations can be classified as *microscale*, *macroscale*, or *mesoscale*. Microscale transportation simulations (for short, micro-simulations) operate at the level of individual simulated entities (SEs), for example an individual vehicle of a person, or the individual herself crossing the street and entering a bus. Transportation micro-simulations address very detailed questions, but are often limited in scale to at most thousands of SEs. Macroscale simulations operate at the aggregate level, abstracting away from individual SEs in favor of concepts like flow. They can scale to much larger systems, but cannot address detailed questions involving individual SEs. Unfortunately there is a big gap between the accuracy of micro-simulations and the scalability of macro-simulations which limits the types of interesting simulations that engineers and scientists are able to perform.<sup>1</sup> We scale up transportation micro-simulations by moving the designers of simulations towards *design patterns* motivated by computational thinking about scalability, we will scale transportation micro-simulations to macroscale, thus allowing scientists to address detailed research questions at a truly large scale.

### 2.1 Emissions Exposure Estimation

Air pollution is an issue of significant importance to public health, climate, visibility, and the ecosystem in the U.S. and across the world. The literature linking air pollution to adverse impacts is extensive and

---

<sup>1</sup>Mesoscale simulations attempt to circumvent the scale limitations of micro-simulations, informally, performing a micro-simulation in an evolving environment determined by a macro-simulation. However, they usually require some deep analysis, make many approximations and are very difficult to program.

growing. Exposure to airborne particulate matter and ground-level ozone increases the risk of cardiovascular disease and lung cancer [7], raises children’s asthma rate [11, 17], and leads to premature death [19]. Transportation emission is the single largest source of air pollution in most urban areas, especially megacities [13]. Numerous studies have found that traffic-related air pollution can trigger heart attacks [15] and lead to cancer [6]. In fact, more people are killed by air pollution from vehicles than by traffic accidents [10]. Thus, air pollution is a top environmental priority, and transportation emission control is crucial to winning the battle against air pollution.

The bridge between air pollution and health effects in advanced simulation models is the notion of *exposure*, which combines ambient and indoor air quality with the activities of individuals—where, when, and how they travel or play—to predict health effects. Accurate exposure estimation is an essential input to a scientifically robust environmental management policy. It is not feasible to measure exposure of a large or widespread population with accuracy; instead, exposure estimates must be obtained by simulation. These simulations must be done at fine granularity for several reasons. First, the air quality model is more accurate with finer spatial resolution. The National Research Council reports that large grid sizes smooth out predicted concentrations of pollutants [10]; another study found significant improvements in prediction of wind components by reducing grid size to  $200 \times 200$  meters [9]. The existence of small sources of concentrated vehicular emissions (e.g., intersections with traffic signals) suggests that even finer granularity may be needed. Second, the effect of exposure is captured more accurately with finer temporal resolution. This is true even if average exposure is the same—a coarse-grained simulation cannot distinguish between an hour of exposure to 5 units of pollutant and an hour during which exposure increases from 0 to 10 and returns to 0.

To achieve high-quality exposure analysis by micro-simulation requires four fundamental component models, together with a high-performance simulation infrastructure to tie them together. An *air quality model*, such as the the Community Multiscale Air Quality model [12], delivers estimates of the levels of ozone, particulates, toxics and their evolution over time. An *exposure model* estimates health effects of levels of exposure to pollution over time. An *emissions model* describes the relationships between transportation behavior (e.g., vehicle speed, acceleration, drivers behavior) and emissions. Finally, a *transportation activity model* for micro-simulation needs to provide second-by-second transportation activity patterns, including speed

and acceleration of individual vehicles. All these components are now available, except for detailed activity patterns. For over a decade, these have been impossible to obtain for larger areas by simulations because of the difficulty of computing fine-grained micro-simulations on a large scale.

We have developed a transportation simulation framework (of both activities and networks) for more accurate, high-resolution, and realistic second-by-second transportation activity data. As a case study, we are planning to develop and implement activity models for the New York City metropolitan area, which has the most diversified and complicated transportation network in the world. We hope that this application of large scale and finely resolved transportation activity simulation will (1) provide fundamental insight on the influence of transportation systems on urban ozone and particulate matter pollution; (2) provide more accurate spatio-temporal mobile emissions data for the city of New York; and (3) advance transportation-air quality modeling in terms of understanding the implications of socio-economic, land-use data and traveler’s decision making for air quality and public health.

### 3 Transportation and Data Mining

There is a rich bi-directional connection between transportation simulations and data mining. We will outline one such example.

Mobile emission inventory models require transportation activity inputs such as traffic volume, vehicle-miles traveled (VMT), travel speed and so on by vehicle class (e.g., light-duty auto, heavy-duty truck) to estimate running exhaust emissions, since emission rates (grams/mile) vary dramatically across different vehicle classes [8]. In particular, emissions estimates used in advanced air quality simulation and exposure assessment models desire the emissions inventory, hence transportation activity data, to be computed at fine temporal (e.g., hourly) and spatial (e.g., a 4km by 4km grid) resolution. Therefore, the accuracy of this whole chain of models is heavily dependent on the accuracy of transportation activity inputs, of which truck transportation data is the focus of this study.

In practice transportation activity data of light-duty vehicles (e.g., passenger cars) are available, to certain degree of accuracy, in period-based outputs from travel demand models or from highway performance monitoring data. Traffic on roads consists of a mix of different classes of vehicles, which produce different levels of emissions per mile driven. Vehicle mix is therefore important for accurate emissions estimation. Vehicle mix/classification data is important, yet very hard to obtain, and even harder if fine spatial and temporal reso-

lution is desired. Oftentimes the default values in emission models, which may not be representative of a particular area, are used [4, 2, 3]. Trucks, especially heavy duty-diesel trucks, tend to contribute more than their fair share to emissions, accounting for approximately 60% of nitrogen oxides and 40% of particulate matter in running exhaust emissions in 2008 in California [5]. Emissions factors (grams/mile) of trucks are tens of times higher than those for passenger cars. If the default vehicle mix in an emission model under-estimates the proportion of truck traffic, adoption of the model default will lead to under-estimation of the emissions inventory.

Therefore, developing a model for better estimation of hourly truck traffic on road networks is important for improving the accuracy of mobile emissions inventory calculation. Thus we need statistical models to improve estimation of truck traffic for emissions modeling. But the feedback also goes the other way: We can mine the results from simulations to understand the root-causes of large emission concentrations, and we can use simulations to guide data mining techniques towards the right parameter space. However, all of these ideas require scalable transportation micro-simulations, the topic of the next section.

## 4 Scaling Transportation Simulations

Fundamentally, the computational problems of transportation micro-simulations lie in lack of scalability: increasing the size or granularity of a transportation simulation also increases the amount of memory and processor performance needed to run the simulation in a reasonable amount of time. In prior work, we have developed the scripting language SGL for increasing the number and expressiveness — meaning the level of sophistication of behaviors — of nonplayer characters (NPCs) in real-time strategy games [18]. In our approach, the programmer specifies the behavior of NPCs at the unit level in SGL. We then compile the behavior of all NPCs into a huge extended relational algebra plan which we can optimize and process efficiently. Specifically, we can take this query plan and (1) add indices which give us asymptotic performance improvements, (2) rewrite the query plan into a more efficient one, and (3) generate an execution plan that scales to the number of cores and available main memory across a cluster. We call this style of processing which isolates the programmer from implementation and execution details *declarative processing*. The programmer scripts only the behavior of an *individual* NPC; efficient processing of *all* NPCs is left to the game engine. Our work on SGL has shown that declarative processing can result in orders of magnitude performance improvement for sim-

ulation games [18].

### 4.1 Declarative Processing

There is an impedance mismatch between the declarative processing model we presented and the imperative programming languages that transportation simulation designers are used to. When designing transportation simulations, it is natural to think in terms of the behavior of an individual SE that is simulated, and we would like our programming model to reflect this level of abstraction. It is only natural to write procedural code, and not try to express the transportation simulation behavior in a declarative language such as SQL. In other words, we want to give the engineer the abstraction of a traditional imperative programming model with sequential execution — but at the same time we would like to be able to automatically scale the transportation simulation across many cores and across a cluster; that is, behind the scenes we would like to enable declarative processing of the transportation simulation.

One of the main ways in which declarative processing achieves performance gains is by intentionally limiting expressiveness. Consider relational algebra, the algebra behind SQL [16]. Relational algebra does not support arbitrary iteration: it has no *for* or *while* loops and no recursion. The only type of iteration that it does support is the *for-each* loop, which iterates through the elements in a list. This type of iteration — a “join” — has been heavily researched and highly optimized. Join iteration is capable of expressing a wide range of complex algorithmic behavior, in particular, it is sufficient for all scenarios that we have encountered in practice. Thus the performance benefits that we obtain by restricting ourselves to join iteration outweigh the small loss of expressiveness.

In order to implement this restriction on expressiveness, we could try to identify and separate joins from arbitrary iteration in the program logic. However, this is a very difficult problem in general. A better approach is to identify *design patterns* amenable to declarative processing and then to provide language constructs and tools based on them. Let us present a first such basic design pattern that allows for extensive declarative optimizations. We will then discuss processing optimizations that this design pattern enables

### 4.2 The State-Effect Design Pattern

We call our first design pattern the *state-effect pattern*. It ensures that imperatively-specified transportation simulations can be executed declaratively. We separate the attributes of simulated entities into two disjoint classes: *states* and *effects*. Our simulation proceeds in steps. State values change only at the end of each step; effects can change within the step as SEs interact with

other SEs and the environment. Having classified all attributes as states or effects, we put restrictions on how states and effects can interact; these restrictions will be enforced by the script compiler. Essentially, every SE has an associated script, which loops repeatedly through these phases:

- (1) It queries its own state attributes and those of some (potentially *all*) other SEs.
- (2) It computes new values for effect attributes.
- (3) It updates the state attributes using their old values and the newly-computed effect values.

Because the state is not updated until the end of the simulation step, the query and computation phases of different SEs are isolated from one another and can be evaluated in any order, or in parallel, or even unsynchronized. This feature has been exploited by game engineers to parallelize physics computations on modern GPUs [14], and map-reduce and its variants have a similar property. This is the crucial property that allows us to optimize scripts using database techniques. Suppose each of our SEs were represented as a record in a database table. We could compile the query phase into a single database query that returns a new table of effects on each object. The post-processing would then use a standard programming model to update the SEs from the values in this effect table. It is well established that the benefits of using traditional database techniques to speed up the query phase can be enormous [16].

### 4.3 Efficient Declarative Processing

When using the state-effect pattern, we benefit from the following techniques for the evaluation of micro-simulations. These techniques have already shown great benefits for large databases and games:

- *Set-at-a-time processing.* Instead of processing SEs individually, we can use database query processing operations to process the simulation for multiple SEs at once by *sharing computation*. Specifically, we will introduce language features that permit deep analysis of the resulting scripts to facilitate such sharing.
- *Novel index structures and associated processing techniques for aggregate computations in simulations.* Initially we will make use of sophisticated index structures developed by the database community to efficiently retrieve information about the environment of the SEs and to allow sharing of this computation. To further improve performance, we will develop specialized new index structures to support simulations.
- *Parallel processing.* We can adapt parallel query processing techniques from the database community to achieve linear scaling as more hardware is

added. This will be completely transparent to the simulation programmer.

### 4.4 From SGL To Transportation Simulations

While the analogy between real-time strategy games and simulations is very strong, SGL has several limitations that keep it from being applicable in a straightforward way to traffic simulations. First, SGL is designed for games, which have very different performance to expressiveness trade-offs. SGL does not have to support a wide range of behaviors — just those that we have encountered so far in games. Simulations, however, require a much larger space of possible behaviors. For example, preliminary study of the C++ code for a transportation simulation [1] found a huge amount of branching, resulting in DAG-shaped query plans that we need to handle efficiently. Second, the real-time constraints in games prohibit behavior that cannot be processed at the animation frame rate. Game designers often “cheat” and substitute inaccurate behavior that looks acceptable to the user. Simulation designers, on the other hand, may sacrifice some speed to get scientifically accurate results. Thus, while the general processing model may be the same, the design and optimization of the programming language are quite different.

Scientific simulations also have very different main memory requirements than computer games. In games, memory is dominated by graphic objects such as 3D models or texture maps, which are rare in simulations. However, if we simulate enough SEs—say all the vehicles in a megacity—we may not be able to fit them all in memory at once. Additionally, spatial indices and other data structures for improving performance also consume memory. Effective usage of physical memory is crucial to efficient processing of high performance simulations.

Behavior of SEs in simulations will also result in very non-traditional query plans. Traditional database query processors are optimized for disk-resident data whereas to achieve sufficient performance for simulations we will scale across the aggregated memory of a cluster. In addition, as already pointed out, the generated query plans will not be tree-structured but DAG-structured, resulting in very novel challenges for query optimization. Last, our custom in-memory query processor should allow us to make effective use of multi-core processors and all the resources in a cluster.

### 4.5 Tools

Unfortunately, while we can use declarative processing to greatly improve performance in simulations, a side effect is that the simulations can be very hard to debug. The query plans needed to achieve optimal performance and effective use of parallel processors are very

different from the individual behavior programmed by the simulation designer. Hence, while we design our in-memory query processor for performance, we need to design a means to translate the simulation state back into a processing model that the designer understands.

In general, the size and scale of these simulations can make debugging very complicated. The behavior of each SE in the simulation could conceivably depend on the state of every other SE in the simulation. Furthermore, since each step in a simulation depends on an earlier step, small and hard-to-detect errors in the simulation can lead to larger errors later on. Therefore, when unexpected or incorrect behavior occurs in a simulation, it can be very difficult to determine exactly what caused that behavior. We need tools for debugging and visualizing the chain of interactions between SEs in the simulation, to isolate and fix incorrect behavior.

## 5 Ongoing Work and Conclusions

We are currently working on implementing our ideas in a first prototype for truly scalable micro-simulations of large transportation networks.

While we are currently targeting accuracy of micro-simulations in the transportation and environmental modeling communities, there are many other instances where research communities are clamoring for the capabilities of truly large-scale micro-simulations ranging from biologists who want to understand complex, emergent behavior from simple individual behavior such as ants or bees to ornithologists who would like to simulate the migration behavior of birds to scientists who investigate disaster response capabilities.

## References

- [1] Federal Highway Administration. Ngsim cooperative lane changing and forced merging model. Technical Report FHWA-HOP-07-078, United States Department of Transportation, 2006.
- [2] U.S. Environmental Protection Agency. Fleet characterization data for mobile6: development and use of age distributions, average annual mileage accumulation rates, and projected vehicle counts for use in mobile6. U.S.S EPA Report EPA420-R-01-047, July 2001.
- [3] U.S. Environmental Protection Agency. Use of locality-specific transportation data for the development of mobile source emission inventories. <http://www.epa.gov/ttnchie1/eiip/techreport/volume04/index.html>, July 2009.
- [4] C.R. Bhat and H. Nair. Vmt-mix modeling for mobile source emissions forecasting: formulation and empirical application. *Journal of the Transportation Research Board 1783*, pages 39–48, 2000.
- [5] California Air Resources Board. 2008 estimated annual average emissions. <http://www.arb.ca.gov/ei/emissiondata.htm>, July 2009.
- [6] U.S. EPA. Particulate matter research needs for human health risk assessment to support future reviews of the national ambient air quality standard for particulate matter. Technical Report EPA/600/R-97/132F, National Center for Environmental Assessment, Research Triangle Park, NC, 1998.
- [7] C. Pope III, T. Burnett, M. Thun, E. Calle, D.D. Krewski, K. Ito, and G. Thurston. Lung cancer, cardiopulmonary mortality, and long-term exposure to fine particulate air pollution. *Journal of the American Medical Association*, 287:pp. 1132–1141, 2002.
- [8] T. R. Jackson. Fleet characterization data for mobile6. U.S. EPA Report EPA420-r-01-047, 2001.
- [9] J. Koupal and S. Srivastava. Moves2004 validation results. Technical Report EPA420-P-05-002, US EPA, 2005.
- [10] N. Kunzli, R. Kaiser, S. Medina, M. Studnicka, O. Chanel, P. Filliger, M. Herry, F. Horak Jr., V. Puybonnieux-Texier, P. Quenel, J. Schneider, R. Seethaler, J. Vergnaud, and J. Sommer. Public-health impact of outdoor and traffic-related air pollution: a european assessment. *The Lancet*, 356:pp. 795–801, 2000.
- [11] R. McConnell, K. Berhane, F. Gilliland, S. London, T. Islam, W. Gauderman, E. Avol, H. Margolis, and J. Peters. Asthma in exercising children exposed to ozone: a cohort study. *The Lancet*, 359:pp. 386–391, 2002.
- [12] Community Modeling and Analysis System Center. Community multiscale air quality. <http://www.cmaq-model.org/>.
- [13] L. Molina and M. Molina, editors. *Air quality in the Mexico Megacity: an integrated assessment*. Kluwer Academic Publishers, 2002.
- [14] H. Nguyen, editor. *GPU Gems*, volume 3. Addison-Wesley, 2007.
- [15] A. Peters, S. von Klot, M. Heier, I. Trentinaglia, A. Hormann, E. Wichmann, and H. Lowel. Exposure to traffic and the onset of myocardial infarction. *The New England Journal of Medicine*, 351:pp. 1721–1730, 2004.
- [16] R. Ramakrishnan and J. Gehrke. *Database Management Systems*. MGH, 3 edition, 2003.
- [17] J. Slaughter, T. Lumley, L. Sheppard, J. Koenig, and G. Shapiro. Effects of ambient air pollution on symptom severity and medication use in children with asthma. *Annals of Allergy, Asthma, and Immunology*, 91:pp. 346–353, 2003.
- [18] W. White, A. Demers, C. Koch, J. Gehrke, and R. Rajagopalan. Scaling games to epic proportions. In *Proc. SIGMOD*, pages 31–42, 2007.
- [19] Working Group on Public Health and Fossil-Fuel Combustion. Short-term improvements in public health from global-climate policies on fossil-fuel combustion: an interim report. *The Lancet*, 350:pp. 1341–1349, 1997.