Parallel Real-time Implementation of Large-scale, Route-plan-driven Traffic Simulation

M. Rickert[a,b,c] and P. Wagner[a]

[a] Center for Parallel Computing, Cologne University, D-50923 Köln, Germany,
mr@zpr.uni-koeln.de and p.wagner@zpr.uni-koeln.de

[b] Los Alamos National Laboratory, TSA-DO/SA MS M997, Los Alamos NM 87545, USA
rickert@tsasa.lanl.gov

[c] The work of MR was supported in part by the
"Graduiertenkolleg Scientific Computing Köln/St. Augustin"

April 1, 1996

Abstract

This work is part of our ongoing effort to design and implement a traffic simulation application capable of handling realistic problem sizes in multiple real–time. Our traffic simulation model includes multi-lane vehicular traffic and individual route-plans. On a 16–CPU SGI Power Challenger and a 12-CPU SUN workstation-cluster we have reached real–time for the whole German Autobahn network.

Keywords: **traffic, cellular automata, complex systems, parallel computing**

1 Introduction

The importance of car and truck traffic increasingly demands tools capable of realistic traffic simulation. Conventional applications, however, usually fail to reproduce the phenomena found in real life traffic because either (a) the resolution of the simulation is not fine enough or (b) the considered area is too small.

Only high-end computer architectures including parallel computing together with high-speed physical methods can deliver the computational performance necessary to tackle these problems.

This work is part of our ongoing effort of the project "Nordrhein–Westfalen Research Cooperative Traffic Simulation and Impacts on the Environment (NRW–FVU)" [1] at the Center of Parallel Computing [2] to design a flexible, high–performance simulation tool for vehicular traffic. The problem size was given by the current Autobahn network of Germany which amounts to approximately 75,000 kilometer of road lane. Estimating an average occupancy of 10% results in 1,000,000 vehicles, each following an individual route-plan during the simulation. We also used the Autobahn network of Nordrhein–Westfalen (a sub–set of the latter) with approximately 11,500 kilometer of road lane.

The traffic model used in this implementation was developed by several authors, at first as a single-lane version [3] and later extended to a two-lane version [4]. Rickert [5] implemented a parallelized traffic simulation using cellular automata (CA) techniques running at multiple real–time[1] for the German Autobahn network on an Intel Paragon. This simulation, however, was not capable of executing individual routes, but used turning probabilities at intersections instead. Moreover it did not include dynamic load balancing.

In the remaining part of this section we will give an overview of the CA in single-lane and multi-lane traffic simulation. The next section describes the network model and necessary modifications of the original CA. We continue by outlining the parallelization scheme of the current implementation in section three, followed by the results of the performance benchmarks in section four.

1.1 Traffic Simulations

Generally, all road traffic simulation models can be classified into *microscopic* or *macroscopic* (fluid-dynamical) models. The microscopic (high-resolution) simulation uses individual cars, each equipped with a route-plan it wants to follow. Their dynamics are modelled on very different levels of fidelity[2]. One of the most detailed models uses a complicated delay differential equation for every car [6], where the acceleration of the car depends on the distance and velocity difference to the car ahead. Although such models perform nicely when compared to measurements, they are computationally very demanding, and have a large set of parameters to be adapted to reality.

At the low end of complexity, we arrive at a model in which both space and velocity are discrete. The dynamics of the cars are reduced to a few simple rules, which are controlled by a small set of parameters. Models of this kind are called cellular automata (CA) for traffic simulation, and can be understood as a minimal microscopic model for simulating traffic. To have only a small set of parameters to calibrate a CA turns out to be a tremendous advantage when coping with more complicated situations such as the lane changing behavior, where it is difficult to calibrate the more complicated models [7].

The macroscopic (low-resolution) models [8] describe the movement of blocks of cars, according to some rules which utilize the continuity equation, together with the empirically measured relation between the car density ρ and flow q. They can be understood as a spatial and time discrete version of a partial differential equation describing a particle flow. Macroscopic models are able to simulate road traffic on very large networks, e.g. it is possible to simulate the German freeway network on a single workstation [9] in real-time. Their main disadvantage, in our view, is their inability to handle a large number of route plans. So, any investigations which rely on route plans, such as in telematics applications, can not be done.

From the considerations made above, we think that the microscopic modeling is the most natural way to simulate road traffic.

1.2 Traffic Simulation using CA

Let us briefly summarize, how to simulate traffic [3]. Space, time and velocity are discrete, each cell is either occupied by a car or is empty. The length of a cell is $7.5m$, which is interpreted as the length of a car plus the gap between cars in a traffic jam. One time-step lasts $1sec$, which is of the order of the reaction time of humans. Velocity ranges from $0, \ldots, v_{\max} = 5$, corresponding to a maximum velocity of approximately $120km/h$.

Let n denote the current time-step, $(\Delta x)_n$ the front bumper to front bumper distance between the car we are looking at and the car ahead, and let v_n, x_n be the current speed and position, respectively. Then we have the following set of rules, which are updated in parallel:

[1] Multiple real-time means that several simulation seconds can be computed in one wall-clock second.

[2] In this context *fidelity* is often used as a synonym for *accuracy*.

$$
\begin{aligned}
v &= \min(v_n + 1, (\Delta x)_n - 1, v_{\max}), && (1)\\
v_{n+1} &= \max(0, v-1) \quad \text{with} \quad \text{probability} \quad p_{\text{brake}}, && (2)\\
x_{n+1} &= x_n + v_{n+1}. && (3)
\end{aligned}
$$

The first rule summarizes the interaction between two cars and their tendency to drive with maximum speed, if there is no other car ahead. The interaction is constructed to avoid any accident. The second rule accounts for the different kind of inaccuracies in human driving behavior, making the model a stochastic CA. The third rule simply advances the cars v_{n+1} sites.

The original model works with one maximum velocity only. However, it is simple to introduce a distribution of velocities and different car types (e.g. trucks).

Despite its simplicity, the model yields quite realistic behavior. It describes the spontaneous generation of traffic jams, it gives space-time plots of traffic flow, which look very similar to aerial views of real traffic, and it yields realistic fundamental diagrams. A fundamental diagram is the graphical representation of the relation between average local speed $\langle v_l \rangle$ and flow q. An example is shown in figures 1, where it is difficult to distinguish between the simulated and the measured fundamental diagrams (see [10]), showing that the CA is capable of reproducing the observed macroscopic behavior.

Even in more complicated situations the CA-model displays remarkably realistic behavior. Examples are the mixing or weaving of two traffic flows or the lane changing behavior of the model. We have found a set of rules, which leads to the correct lane-usage behavior: on German freeways, the left lane has a higher occupancy than the right lane, even for moderate values of the flow. This behavior can be reproduced with the CA-model. More details can be found in [8].

1.3 Other Models and Implementations

Several other research groups are currently involved in large scale traffic simulation. The TRANSIMS [11][12] group at the Los Alamos National Lab simulates large urban areas down to individual intersections on workstation clusters. The PARAMICS [13][14] group in Edinburgh (EPCC) simulates the whole national federal road network on a Cray T-3D.

A comprehensive summary about how the traffic CA has been used in simulation models can be found in [15].

2 Model Description

Using the criteria introduced above, we would like to characterize our simulation (henceforth called *MicroSim*) as a low fidelity (CA), high resolution (both in time and space) traffic simulation using individual route-plans as routing method. This section will give a detailed description of how we extended the original CA to cover net simulation.

2.1 Overview

The main objective of MicroSim is to execute route-plans in a realistic street network. Each route-plan is defined by a source, a destination, a list of intermediate net points, and a departure time which are used as follows: After the vehicle has been instantiated[3] at the given departure time,

[3] In object-oriented programming languages the term *instantiate* is used for the dynamic creation of a memory object (e.g. vehicle). These objects usually have a limited life-time before they are *deleted* or *disposed*.

it will be inserted into the simulation network at the origin. It will then execute the route-plan until it reaches its destination. Finally, it will be removed from the system after statistics about its actual travel time have been collected. For the time being, the route-plan is to be regarded as *static* for each simulation. There is no online rerouting being performed. This will be covered by future versions of the simulation.

A simulation run is initiated by supplying a map defining the geometry of the street network and a list of routes. Vehicles will be instantiated according to their departure times until all routes have been processed. The simulation will continue until a given percentage of all instantiated vehicles have reached their destination. This percentage should be chosen to find a reasonable compromise between the duration of the simulation (small percentage) and a comprehensive overview over successful route-plan execution (large percentage).

In addition to the traffic volume generated by routes MicroSim provides a mechanism to generate background traffic (see 2.6) of a given density. In this case vehicles will be homogeneously instantiated across the network before the actual simulation is started resulting in the selected density. In contrast to their routed counterparts they do not carry route-plans at all. Instead, they change directions at intersections according to a given turning probability. In case an un-routed vehicle is about to leave the network, it will be *reflected* with its current lane and velocity leading into the opposite direction.

The network representation used in MicroSim is a graph in which each intersection represents a node[4] and each street segment between intersections corresponds to two edges[5]. Moreover, there are nodes defined by the *natural* boundaries of a road network with node degree one, called *terminators*, and additional nodes with degree two where vehicles can enter or exit the network, called *ramps*[6]. This network is usually supplied in two sets of objects: (a) the *set of nodes*, each of which has a unique number (id) and the geometric location of the object, given in rectangular coordinates relative to an arbitrary, but fixed point, and (b) the *set of edges*, each of which has two references (by id) to nodes and optional information such as name, number of lanes, or speed limit.

This representation is an over-simplification of the motorway-intersections found in real networks. Figure 2 depicts an intersection of degree four with its given *substructure* and the corresponding simplified graph structure.

The reason why MicroSim is still based upon such a simple input data format is simply due to its general availability. More elaborate data formats — especially for the German street network — have only started to emerge. It will probably take another five years or so before a comprehensive data base will be available. Meanwhile, the concept is to provide a set of substructures which can be used as templates to model the real intersections. Currently available substructures are covered in 2.5.

2.2 CA Rule Extension

The original CA model for traffic simulation was defined for a single lane and periodic boundary conditions [3]. One of the most important aspects of this approach was to keep the model as simple as possible to allow for an efficient implementation. Three rules proved to be sufficient for single-lane traffic. Later [4] the model was extended to include two-lane traffic featuring realistic lane changing behavior. The CA model used in MicroSim corresponds to the asymmetric version presented in [4]. An additional extension was made to allow for more than two lanes: in such a system, all center lanes are possible candidates for collision whenever vehicles separated by a single site (orthogonal

[4] vertex

[5] A segment can correspond to one edge or two edges depending on whether the two directions are equivalent, or not. MicroSim uses the latter, even if all characteristics of both directions are identical, since this symmetry is broken during simulation, anyway.

[6] The segments feeding the ramps are not part the network. Therefore they do not increase the degree of a ramp. If the map were extended to include lower hierarchies, ramps would also have degrees larger than two.

to flow of traffic) decide to change to that very empty site between them. One way to resolve this collision is to update the lanes from left to right or vice versa sacrificing the concept of a purely parallel update.

Let us now outline how route-plans are integrated into the CA behavior. It is important to realize that a vehicle within the CA context does not really "see" a street network while it travels through it. Rather, its "view of reality" is always restricted to the current grid. Likewise, the vehicle does not *actively exit* a grid. Whenever it has to change grids to fulfill its route-plan, it is *passively absorbed* from its current location and *emitted* into the destination grid preserving its current velocity. Thus, the only requirement for a vehicle to be absorbed correctly turns out to be in the correct lane or possibly correct group of lanes when approaching a network node.

In the original model each vehicle evaluates its state deducing a truth value for the following statements:

- *change to left lane required:* small gap on current lane, sufficient gap on left lane
- *change to right lane required:* sufficient gap on right lane
- *change to left lane prohibited:* vehicle within certain distance when looking over left shoulder, or left neighboring site occupied
- *change to right lane prohibited:* vehicle within certain distance looking over right shoulder, or right neighboring site occupied

Suppose a vehicle on lane l has to move to the group of lanes denoted by $l_l \ldots l_r$. In order to include lane changing induced by route-plans, the statements above can logically OR-ed with the corresponding statement in the following table:

- *change to left lane required:* $l_r < l$
- *change to right lane required:* $l < l_l$
- *change to left lane prohibited:* $l_l \leq l$
- *change to right lane prohibited:* $l \leq l_r$

Note that the parameters describing these conditions can easily be coded with a few bits of a `long` variable.

In "classical" cellular automata all state information was usually kept in one computer memory unit (*long* or *word*). This was advantageous both for accessing state information (evaluation of rule set) and moving state information (moving particles). If we applied this principle to our routed vehicles, a site would include all CA vehicle information: the current velocity v, the identity, and the route-plan. Unfortunately, the route-plan information is considerably longer than the original CA. Moreover it does not have a constant length which would complicate the handling of the grid considerably. Thus it was obvious to chose a different approach.

As mentioned in [16][17], we distinguish between so-called *primary* vehicle data and *secondary* vehicle data (see fig. 3). The CA update-logic actively accesses the primary data only, consisting of the original CA state (current velocity) and parameters required for the lane selection as described above[7]. The secondary vehicle data (route-plan) is kept in a data structure dynamically allocated when the vehicle is instantiated. A pointer to that data is kept as the second component of each grid site. During a CA update the primary data and the pointer to the secondary data are always moved simultaneously.

[7] The third important parameter is a random value (see 3.4).

Near network nodes, the intersection control scans predefined areas for occupied sites (see 2.4). Now, in contrast to the CA logic, the pointer to the secondary data set is actually evaluated to retrieve routing information. In case a vehicle has to exit, the intersection control will change the primary CA data to influence subsequent CA behavior accordingly, until the vehicle is absorbed.

2.3 Route-plans

The first attempt of including route-plans into a medium scale traffic simulation was done in one of the early versions of the TRANSIMS project [18]: the interstate traffic of the city of Albuquerque was simulated on a single workstation to show the general feasibility of this approach. Nagel [16][19][20] used a parallel computer with two CPN[8] to run a parallel net simulation based upon the single-lane CA with individual route-plans. He examined iterative route-selection behavior of a group of drivers travelling through the network. The NRW-FVU [1], TRANSIMS [12], and PARAMICS [13] groups are currently designing large-scale traffic simulations that include route execution.

For MicroSim, route-plans represent the third major input for the simulation beside nodes and edges. Each route-plan entry contains information about the node id of the origin, the scheduled departure time-step from the origin, the estimated travel time in simulation time-steps, a list of node id including the destination as its last entry, and optional vehicle data.

MicroSim expects the routes to be sorted according to their departure time step. Thus the evaluation of the route-plan is reduced to the following scheme: at every time step routes are sequentially read until a departure time step is found that is larger than the current time step. For each route a vehicle is created and loaded with that route. The vehicle is appended to the insertion queue of the source associated with the given origin id. If there is more than one source per id (e.g. ramps) also the second node id of the route-plan is scanned to determine the outgoing segment and thus the specific source. Note that the scheduled time of insertion may differ from the actual time of insertion, since the source can only add vehicles to the grid if there are vacant sites. During intervals of high insertion rates there will be a certain number of vehicles waiting (or 'pending') in the source queue.

2.4 Basic Network Elements

We designed eight basic network elements which serve as building blocks for the intersection templates. Similar to the design of the traffic CA, the basic elements are discrete in time and space. Their rule-sets are of comparable complexity.

Transfer Segment (TS) Transfer segments are multi-lane CA grids of length $l_{transfer}$ used to let vehicles travel from absorption ranges to emission ranges. They can have any number of lanes. In case of $n_{lanes} = 1$ the CA rule set is equal to the single CA of Nagel and Schreckenberg, otherwise the lane-changing behavior described in 2.2 applies. The beginning and the end of a transfer segment is either defined by a connector or a block preventing vehicles from advancing any further.

Source (SRC) A source is associated with a network node and the outgoing lanes of a street segment. It carries a queue of vehicles that are to be inserted into the street segment. Every time-step it scans the first site on each lane for vacancy. Each vacant site is then filled by those vehicles in the queue which have been waiting the longest time. Compared to the "look back" rule of the two-lane rule-set for lane changing, this is a rather simple behavior. A more elaborate rule-set can be found in [16].

[8] Computational Node: a more general term than processor for one unit of large computer system solving a part of a distributed parallel problem.

Sink (SNK) A sink is associated with a network node and the incoming lanes of a street segment. Every time step it scans the last (closest to the node) v_{max} sites of each lane to detect vehicles that carry a route-plan whose destination node is the same as its own network node. Whenever such a vehicle is found it is removed from the site and deleted after some information about the route-plan execution has been collected.

Marking Range (MR) The marking range is used to mark vehicles for exit. It is associated with a source street segment and a destination street segment. Every time-step it scans a range of sites of length v_{max} on the source segment for vehicles carrying route-plans that require exiting to its destination segment. Whenever such a vehicle is found it will be marked with a flag which influences CA lane-changing behavior.

Absorption Range (AR) An absorption range is associated with a source street segment, a destination street segment and a transfer segment. At every time-step it scans each lane (see fig. 4) over a range of length l_{merge} on the source segment for vehicles that are marked for exit to its destination segment. Whenever such a vehicle is found and the corresponding site on the transfer segment is vacant, it will be moved from the source segment to the transfer lane preserving its current velocity. At the same time the flag which caused the transfer will be reset.

Deceleration Range (DR) A deceleration range is associated with a street segment. At every time-step it scans a certain set of lanes (not necessarily all lanes) over a range of length v_{max} (see fig. 4) for vehicles marked for exit and currently located in a lane leading them to a wrong destination. Whenever such a vehicle is found, its maximum velocity is decreased as follows: Let v_i denote its current velocity, $v_{i,max}$ its maximum velocity, and p_i the position of the vehicle, where $p_i = 0$ ($p_i = v_{max} - 1$) represents the first (last) site of the range of the range, respectively. Then

$$\begin{aligned} v_{i,new} &= \min(v_i, v_{i,max}, v_{max} - p_i - 1) \\ v_{i,max,new} &= \min(v_{i,max}, v_{max} - p_i - 1) \end{aligned}$$

represent the new current velocity and new maximum velocity, respectively. The vehicle will never be able to leave the range in forward direction and will eventually stall at the end of the range. It can only proceed by changing lanes and thus restoring its maximum velocity to its old value.

Emission Range (ER) The emission range is the counterpart to the absorption range. It is associated with a transfer segment and a destination street segment. At every time step it scans each lane at the end of the transfer segment over a range of length l_{merge} for vehicles. Whenever a vehicle is found and the corresponding site on the rightmost lane of the destination segment is vacant the vehicle is transferred from the transfer segment to the destination segment.

Connector (CN) The connector is associated with a source street segment and a destination street segment. At every time step it copies boundary information from the outgoing lanes of the destination segment to the incoming lanes of the source segment and vice versa. The associated CA grids can thus be regarded as uninterrupted so that no special rules are needed to guarantee consistent update between segments.

2.5 Composite Network Elements (Substructures)

We hope to be able to capture the main characteristics of an intersection like throughput and capacity by defining adequate template substructures and choosing available parameters accordingly.

Composite Element	TS	SRC	SNK	MR	DR	AR	ER	CN
terminator	0	1	1	0	0	0	0	1
ramp	4	2	2	2	2	2	2	2
intersection degree 3	4	0	0	4	4	2	2	2
intersection degree 4	8	0	0	8	8	8	8	4

Table 1: *Composite Network Elements*

These are: the number of lanes $n_{transfer}$ of the transfer segment, the maximum velocity and related CA parameters of the transfer segments, the lengths and positions of the merge, absorption, and emission ranges, as well as the relative position of the ranges with respect to each other. Throughput measurements of real intersections will be used to validate the templates and compare them to other models that have a higher fidelity of their intersection representation [14][21].

Each composite network element is composed from several basic network elements described above. In addition to their individual functionality they share the ability to forward vehicles with respect to the *through-lanes* of a *through-segment.* This is done by using connectors to pass boundary information of an incoming segment to the corresponding outgoing segment and vice versa. Which lanes are to be regarded as through-lanes will be described below. Table 1 shows an overview over how many basic elements constitute one composite element.

Terminator A terminator consists of a source and sink. They are necessary to define boundary conditions for the traffic volume generated by routes. Since there is only one incident street segment it is defined to be its own through-segment.

Ramp The ramp serves as an origin and destination for route-plan execution. Since vehicles can leave into two distinct directions there are two sources (and, of course, two sinks) per ramp. The through-direction is trivially defined.

Intersection of Degree 3 The substructure of an intersection of degree 3 (see fig. 5) is designed to be asymmetric: two of the three incident street segments are regarded as through-directions, one is subordinate. Since information about the actual structure of those intersections is usually not available, the geometric graph information is used to determine the through-direction as follows: for each pair of incident segments the enclosing angle is calculated. Of those the pair with the angle having the smallest deviation from π is regarded as through-direction, the remaining segment as subordinate.

Due to the discrete nature of the CA there is another characteristic: vehicles coming in on the subordinate segment have to change lanes to reach either the left $\lfloor n_{lanes}/2 \rfloor$ lanes for a left turns or the right $\lceil n_{lanes}/2 \rceil$ lanes for a right turn. Suppose two vehicles A and B end up on the very last sites of the incoming segment with A bound to make a right turn, but located on one of the left lanes, and B bound to make a left turn, but located on one the right lanes. In such a case the vehicles will block each other from changing lanes, resulting in a dead lock which is not automatically resolved through the CA rule set. Therefore at every time these locations are scanned for pairs of vehicles which fulfill the above condition. In case such a pair is found the vehicles will simply be switched.

Intersection of Degree 4 The substructure of an intersection is regarded to be completely symmetric resembling a *clover.* Opposing incident street segments are regarded as through-directions. Figure 6 shows the structure of this type of intersection. Note that due the symmetry only 2 out of 8 transfer segments are displayed.

2.6 Background Traffic

In addition to the network traffic generated by routes it is possible to select a certain density of un-routed vehicles as background. These vehicles are generated automatically during startup and are homogeneously distributed across the network. Their CA behavior is exactly equivalent to that of routed vehicles except at network nodes: at terminators un-routed vehicles are not removed from the network but reinserted into the opposite direction using connectors. Therefore un-routed vehicle do not "see" terminators at all. They also completely disregard ramps. As for intersections of degree three and four, a certain ratio of un-routed vehicles behave as though they had temporary route-plans: With a given turning probability a vehicle is marked and later absorbed at the associated absorption range.

Note that for each vehicle this probability is only applied once even though it may remain on the marking range for more than one time step. This is to maintain a density-independent exit behavior.

As a side effect the background traffic can be used to check the consistency of the simulation especially for the distributed version: running the simulation without routes, that is, with background traffic only, the number of vehicles is constant. Inconsistencies due to the intersection or parallel functionality can thus easily be detected since those usually result in a loss of vehicles or additional (phantom) vehicles.

3 Parallelization

The inherent structure of a traffic microsimulation favors a *domain composition* as the general approach to parallelization:

- The street network can easily partitioned into tiles of equal or almost equal size. A realistic measure for size is not the number of net elements (nodes and segments), but the CA grid lengths associated with those elements (see 3.5). Tiles are then assigned to processors.
- The range of interdependencies between network elements are restricted to the interaction range of the CA. All current rule sets have a interaction range of either $v_{max} \equiv 35[m]$ or $2v_{max} \equiv 70[m]$ which is a short distance compared to the average length of the edge segments (e.g. $484[site] \equiv 3630[m]$ for map FRG, see table 3) in a motorway network[9]. Therefore, the most straightforward approach is to cut the network at the middle of street segments. As a consequence the tiles exchange boundary information containing all data necessary for the evaluation of the CA rule sets (see 3.4), resulting only in local communication between neighboring tiles.

3.1 Parallel Toolbox

The parallelization of MicroSim was done by defining descendent C++ classes of the C++ base classes provided in the Parallel Toolbox. A description of the toolbox is beyond the scope of this paper. More information can be found in [22].

However, we would like to outline how the traffic network is projected onto the structural elements supplied by the toolbox. These are *nodes*, *edges*, and *boundaries*:

- The *node* class was used to represent exactly one node of the traffic network. The toolbox guarantees that nodes exactly reside on one CPN. This is advantageous for the substructures (see 2.5) associated with a node: all elements can assume that other related elements of the same substructure reside on the same CPN. If an incident edge happens to be split (see below),

[9] This is an obvious difference to a city street network which has considerably shorter segment lengths.

at least the half of the edge next the node can be assumed to be local. Other edges of the subnetwork are local anyway.

- The *edge* was used to represent a bidirectional multi-lane street segment. For each direction a multi-lane CA grid was used. In contrast to nodes, an edge may be duplicated by the toolbox in case that the incident nodes reside on different CPN. Such an inter-CPN edge is split exactly in the middle[10]. On one CPN the first half is active and on the other CPN the second half. Exactly in the middle boundaries are retrieved from the grids and transferred to the remote CPN (see 3.4).

3.2 Initial Distribution and Load Balancing

The toolbox handles the initial distribution and subsequent load balancing if requested. The geometric node locations are used to perform a recursive orthogonal bisection of the traffic network. Since no topological aspects are considered, the resulting tiles may not be connected anymore. Nevertheless, each CPN can reestablish a single connected component by casting off all superfluous, not connected components to neighbors and keeping the largest one only.

During the course of the simulation dynamic load balancing is performed. The implemented method corresponds to a *local decision, local migration* ($LDLM_S$, see [23]) strategy applied to the network nodes. Incident edges are transferred or split accordingly. When a part of a local network has to be off-loaded, nodes are sequentially transferred along the boundaries with the node furthest away from the center of the subnetwork being selected first. As an optional restriction only those nodes can be selected that maintain one connected component on the CPN. See [24] for a more detailed description of the dynamic load balancing.

3.3 Timing

The simulation uses a parallel update with a global time-step. However, synchronization of all CPN is only performed after a *simulation–sequence* comprising approximately 10-20 time–steps. In between, there is only an implicit synchronization through the exchange of boundaries.

The global time–step is used to guarantee consistent collection of statistical data: Although partial results from the CPN may not be collected at the same physical wall-clock time due to a potential time-step gradient (see [25]), they always belong to the same logical time-step. The master CPN takes care of combining partial results.

Each global time-step is subdivided into two sub-time-steps. The first sub-time-step is used for lane changing, while the second sub-time-step is used for forward motion. Each sub-time-step requires the exchange of boundaries between CPN, although they are of different resolution: the first time-step only requires the transfer of primary vehicle data, while the second sub-time-step also comprises the secondary data.

Each sub-time-step is subdivided into a preparation phase (P) and an execution phase (E) preceded by the implicit local synchronization (IS) through boundary exchange as summarized in table 2.

3.4 Boundaries and Object Migration

Due to definition of the CA rule-set, there cannot be any concurrent computation of subnetworks without local exchange of boundaries before every time-step. As described above, the subdivision of a time-step even results in two exchanges for each time-step. So, in contrast to other mechanism

[10] Note that, of course, a discrete CA grid of odd length l has to be handled with care by assigning $\lfloor l/2 \rfloor$ sites to one and $\lceil l/2 \rceil$ sites to the other CPN after breaking their symmetry.

sub-time-step	IS/P/E	Action
1	IS	exchange primary vehicle data, gather statistics
1	P	CONN, DR, MR, resolve dead-locks
1	E	lane change
2	IS	exchange all vehicle data
2	P	CONN, ER, AR
2	E	motion, migration

Table 2: *Timing*

(e.g. statistics) that require only *regular* but not necessarily *frequent* communication, the *frequency* of communication cannot be changed for boundaries.

As to the length of boundaries, an optimization can be made by taking advantage of specific characteristics of the CA rule set. Usually the boundary that has to be transferred is as large as the *interaction range*[11] of the CA rules, which is currently v_{max}. This would result in encoding and decoding of all vehicle data that are located within a range of v_{max} sites from a boundary. If the local density is high, that is, the boundary is located within a traffic jam, there may be more than one vehicle per lane. The CA rules, however, only refer to the *immediate* predecessor or successor on each lane, reducing the maximum number of vehicles in a boundary to one per lane. Moreover, only the *primary* vehicle data is needed and not the secondary data including route-plan. This is true, at least for the first sub-time-step. In the second sub-time-step all vehicle data is needed to guarantee a consistent vehicle migration across CPN boundaries, which we will describe next.

After each second sub-time-step there is a certain chance for vehicles to have crossed the inter-CPN boundaries. Let us consider the case of a vehicle migrating from origin CPN A to destination CPN B: The vehicle is updated by *both* CPN in an *identical* fashion. Afterwards CPN A disposes the vehicle, while CPN B converts it from a boundary vehicle to a local vehicle. In a non-deterministic simulation this requires that the random numbers defining the stochastic part of the CA rule-set must be reproduced on both CPN. This can easily be achieved by including the random number into the primary vehicle data which is transferred in each sub-time-step anyway. In MicroSim a single bit is used to carry the stochastic information.

3.5 Load Estimate

Since the performance of the CA only weakly depends on the number of vehicles in a grid we use a value proportional to the number of grid sites handled by a segment as a measure for its computational load. Measurements (see [24]) confirm that the time required for updating one million sites [MUP] only varies by a factor of two for densities between $\varrho = 0.003 \ldots 0.3$

As for the nodes their load was estimated by first summing up the sites on all transfer segments and secondly weighing this value by a factor f_{node} to include the increased computational load generated by the additional intersection functionality.

[11] Actually, the real value that defines the boundary length is the maximum of both *interaction range* and *maximum velocity*, but in a consistent, collision-free CA update the first is always at least as large as the second. In our current rule-set they happen to be equal.

	NRW	FRG
nodes	549	3,307
edges	1,160	6,860
terminators	19	46
ramps	349	1,568
intersections (degree=3)	39	176
intersections (degree=4)	21	58
nodes (degree≠2)	79	280
transfer segments (TS)	1,720	7,440
lane-kilometer	11,712	74,844
sites	1,561,600	9,979,200
average edge length [sites]	448	484

Table 3: *Network Sizes*

4 Performance

We did measurements using different street network sizes on two different computer architectures, both with PVM (see [26]) as underlying communication library. One platform was a workstation cluster of 12 SUN SPARC-station (5, 10, and 20 with performances between 133 and 135.5 MIPS) connected through 10 Mbit Ethernet (PVM architecture SUN4SOL2). The other one was a 16 processor SGI Challenger (SC 900 XI) Shared Memory system (PVM architecture SGI64). On the latter we did not explicitly use the shared memory architecture, but the PVM architecture `SGI64` which is based on UNIX sockets for inter-process communication.

The networks consisted either of the whole German Motorway network (FRG) or an excerpt thereof, namely the sub-network of the federal state Nordrhein-Westfalen (NRW). Table 3 gives an impression of their respective sizes.

Since the available data did not contain any information about street segment characteristics nor substructures of intersections we used the defaults $n_{lane} = 3$, $v_{max} = 5$, and $p_{brake} = 0.5$ on all segments, as well as $n_{transfer} = 1$, $l_{merge} = 5$, $l_{transfer} = 200[m] \equiv 26[site]$ for all intersections and ramps. The density of background traffic was set to $\varrho = 0.1$ for measurements on the SGI and $\varrho = 0.05$ on the workstation cluster. Note that we did not actually use route-plans since filling the network with the help of route-plans would take considerably longer and vary the computational load until a constant density is reached. As the performance only suffers less than 5% during vehicle insertion activity the measured values are also true for routed simulations. Figure 7 shows that after approximately 200 time-steps of load balancing the simulation reaches its optimal performance. Since there was not any other activity on the computer systems during the measurements, this load balancing activity is only due to the imbalances caused by the initial distribution: The work station cluster was running idle and the SGI was only loaded with jobs running at a high nice-level. Judging from the load reported by the UNIX-utility `top` the performance on the SGI may be improved by another 5% to 10% if the machine were not loaded otherwise.

5 Summary

In this paper, we presented a new framework to perform high-speed traffic microsimulation. The concept is based upon cellular automata as underlying traffic model and parallel computer architectures for efficient implementation. The CA simulation part (500 lines of C-code) was embedded in a C++ class library handling the net simulation (10,000 lines of C++-code). Our toolbox for parallel distribution and dynamic load balancing (25,000 lines of C++-code) served as a link to the parallel message passing library PVM.

We have shown that realistic system sizes can be computed in real-time or faster. The complete motorway network of a medium-sized country like Germany can be computed in real-time on hardware that is already available at many institutions. For a smaller network (map NRW) the real-time-ratio reaches almost factor four.

The original concept of cellular automata was preserved wherever possible resulting in a high performance. Route-plans were added to the traffic model by using secondary vehicle data to give vehicles individual properties. We provided a simple set of structural elements to build intersections which can be used as templates to model intersections found in real networks. The next step must be to validate these structural elements. Moreover the CA-rules (especially the lane-changing) have to be reconsidered to show a benign behavior close to intersections.

An immediate application of our microsimulation is the iterative computation of origin-destination matrices using individual route-plans. With map NRW, one iteration for a rush-hour of four hours simulation time can be computed in one hour real-time. Therefore, even if convergence is slow (e.g. 20 iterations), results could be obtained within a day. Conventional systems which are at least one order of magnitude slower would require at least a week.

6 Acknowledgements

We thank A. Bachem, R. Schrader and C. Barrett for supporting MR's and PW's work as part of the traffic simulation efforts in Cologne ("Verkehrsverbund NRW") and Los Alamos (TRANSIMS). We thank K. Nagel for helping us with this paper. We also thank them, plus Ch. Gawron, K. Nagel, S. Krauß, S. Rasmussen, and M. Schreckenberg for help and discussions. Computing time on the SGI-1 Challenger of the Regionales Rechenzentrum Köln and on the workstation cluster at TSA-DO/SA is gratefully acknowledged. We further thank all persons in charge of maintaining the above mentioned machines.

The work of MR was supported in part by the "Graduiertenkolleg Scientific Computing Köln/St. Augustin".

References [11], [15], [16], and [20] can be retrieved from [12]. References [3], [5], [9], [22], and [25] can be retrieved from [2]. See [1] for further examples of traffic maps.

References

[1] NRW-FVU Home Page. http://www.zpr.uni-koeln.de/Forschungsverbund-Verkehr-NRW/.

[2] Center of Parallel Computing Home Page. http://www.zpr.uni-koeln.de/.

[3] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *J. Physique I*, 2:2221, 1992.

[4] M. Rickert, K. Nagel, M. Schreckenberg, and A. Latour. Two lane traffic simulations using cellular automata. accepted by Physica A, 1995.

[5] M. Rickert. Simulation zweispurigen Verkehrsflusses auf der Basis zellularer Automaten. Master's thesis, Universität zu Köln, 1994.

[6] R. Wiedemann. Simulation des Straßenverkehrsflusses. Technical Report Heft 8, Institut für Verkehrswesen der Universität Karlsruhe, 1974.

[7] M. McDonald and M. A. Brackstone. Simulation of lane usage characteristics on 3 lane motorways. In *Proceedings of the 27th International Symposium on Automotive Technology and Automation (ISATA)*, 1994.

[8] P. Wagner. Traffic simulation using cellular automata: Comparison with reality. In *Proceedings of the conference "Traffic and Granular Flow"*, 1995. to be published.

[9] J.T. Pfenning. *Beiträge zum Einsatz von "Workstation Clustern" als Parallel-Rechner*. PhD thesis, University of Cologne, 1994.

[10] Caltrans Home Page. http://www.scubed.com/caltrans/transnet.html/.

[11] C. Barret and D.J. Roberts. The TRANSIMS microsimulation status. Technical report, TSA-DO/SA, Los Alamos National Lab, New Mexico, USA, 1994.

[12] TRANSIMS Home Page. http://studguppy.tsasa.lanl.gov/.

[13] Paramics Home Page. http://www.epcc.ed.ac.uk/epcc-projects/paramics/.

[14] D. McArthur. The *PARAMICS* Model: Present and Future Directions. Technical report, SIAS Ltd., Edinburgh, 1994.

[15] K. Nagel, C. Barrett, and M. Rickert. Parallel traffic micro-simulation by cellular automata and application for large scale transportation modeling. Technical report, TSA-DO/SA, Los Alamos National Lab, New Mexico, USA, and Santa Fe Institute, Santa Fe, New Mexico, USA, and Center for Parallel Computing, University of Cologne, Germany, 1996. submitted to Transportation Research C.

[16] K. Nagel. *High–speed Microsimulations of Traffic Flow*. PhD thesis, Universität zu Köln, 1994.

[17] A. Bachem, K. Nagel, and M. Rickert. Ultraschnelle mikroskopische Verkehrssimulationen. In R. Flieger and R. Grebe, editors, *Parallele Datenverarbeitung Aktuell TAT*, 1994.

[18] C.L. Barrett. personal communication.

[19] K. Nagel and M. Schreckenberg. Traffic jam dynamics in stochastic cellular automata. In J. I. Soliman and D. Roller, editors, *Proceedings of the 28th International Symposium on Automotive Technology and Automation (ISATA)*. Automotive Automation Ltd, Croydon, England, 1995.

[20] K. Nagel, S. Rasmussen, and C. L. Barrett. Network-traffic as a self-organized critical phenomenon. In F. Schweitzer, editor, *International Conference "Self-organization of complex structures: From individual to collective dynamics"*, 1995.

[21] Ch. Gawron, P. Oertel, and P. Wagner. The Lego traffic simulation. Technical report, Traffic Group at the Center of Parallel Computing, University of Cologne, 1996.

[22] M. Rickert. Parallel Toolbox 1.0. Technical report, Center for Parallel Computing, Cologne, Germany, and TSA-DO/SA, Los Alamos National Lab, USA, 1995.

[23] R. Lüling, B. Monien, and F.Ramme. Load balancing in large networks: A comparative study. In *3rd IEEE Symposium On Parallel And Distributed Processing*, pages 686–689, 1991.

[24] M. Rickert, P. Wagner, and Ch. Gawron. Real-time traffic simulation of the German Autobahn Network. In *Proceedings of the 4th PASA Workshop*, 1996. to be published.

[25] K. Nagel and A. Schleicher. Microscopic traffic modeling on parallel high performance computers. *Parallel Computing*, 20:125–146, 1994.

[26] Parallel Virtual Machine Home Page. http://www.epm.ornl.gov/pvm/pvm_home.html.

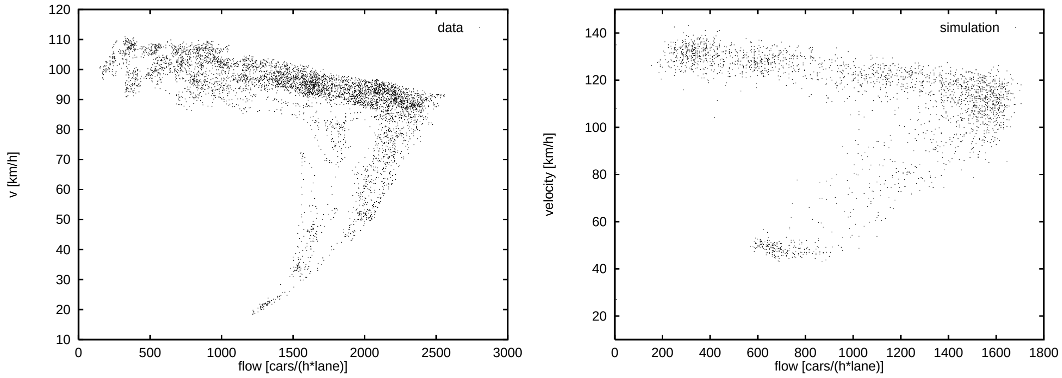

Figure 1: *Left:* Measured fundamental diagram. Data are from a Californian highway. *Right:* Simulated fundamental diagram. Data are a combination of different ρ-values.

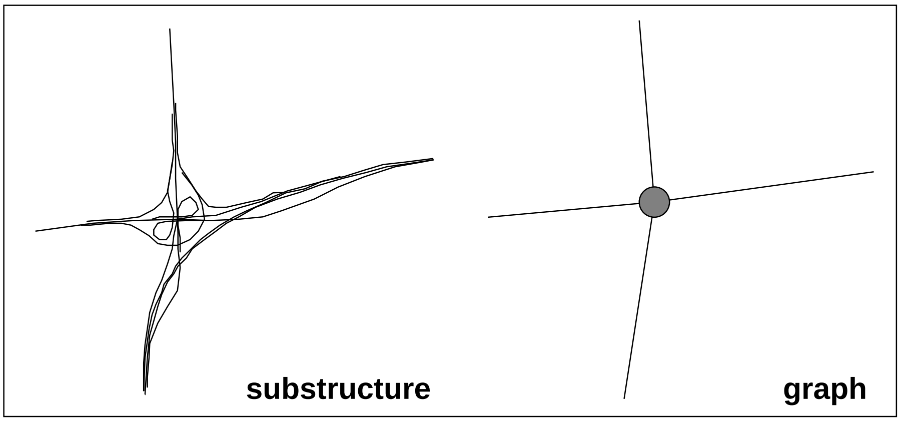

Figure 2: *Substructure of an Intersection of Degree 4* — The left side depicts the substructure of an intersection of degree four. All deceleration, transfer, and acceleration lanes are explicitly coded. The intersection in question is *AK Jüchen* of Autobahn A44 and A46. The right side shows the corresponding graph representation which MicroSim uses as input format.

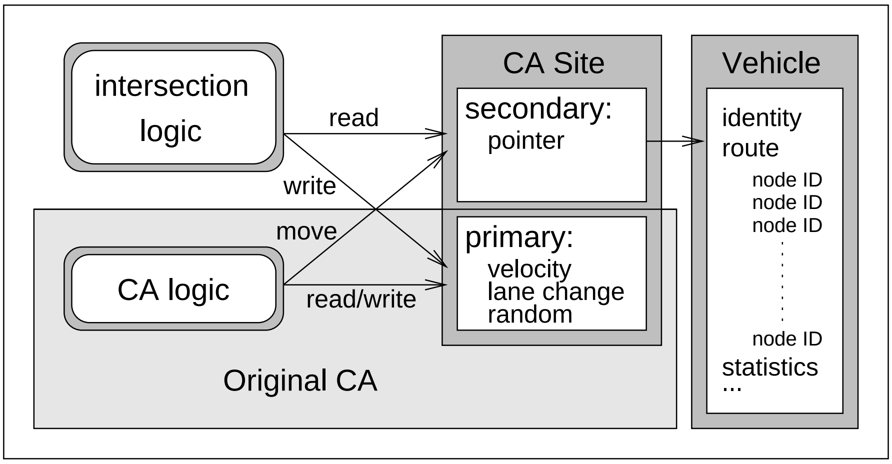

Figure 3: *Primary and Secondary Vehicle Data*

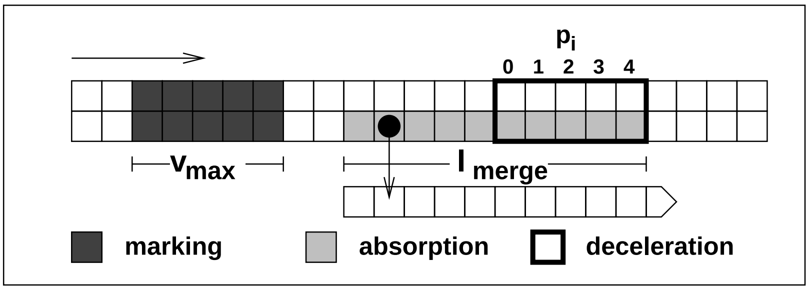

Figure 4: *Marking, Deceleration, and Absorption*

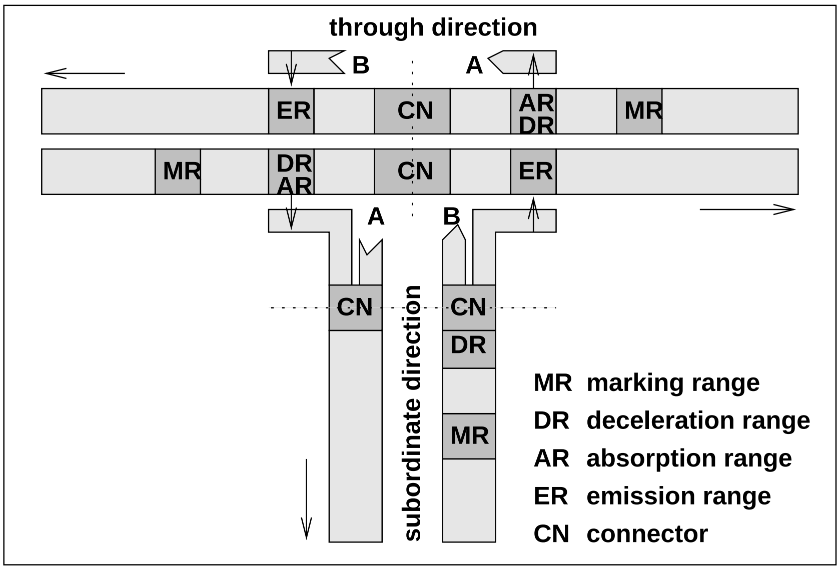

Figure 5: *Intersection of Degree 3*

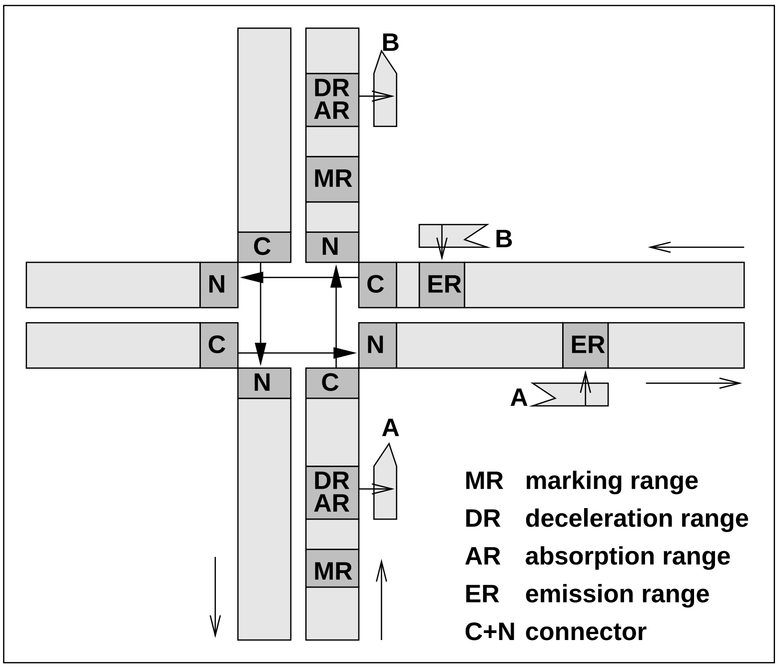

Figure 6: *Intersection of Degree 4*

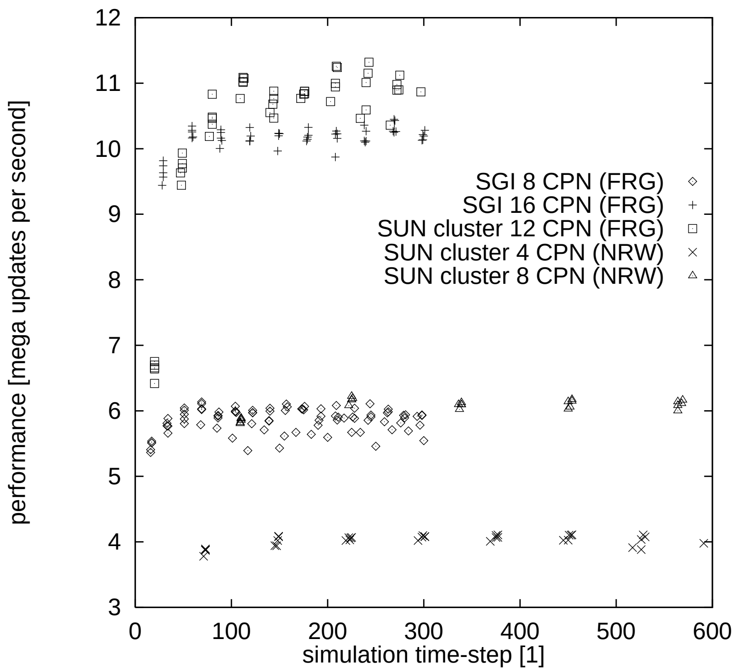

Figure 7: *Performance in MUPS*

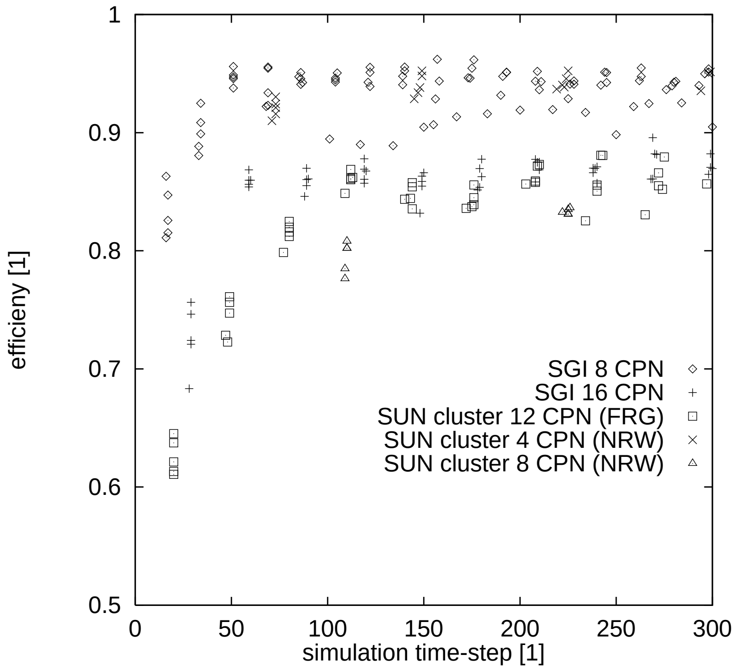

Figure 8: *Efficiency*

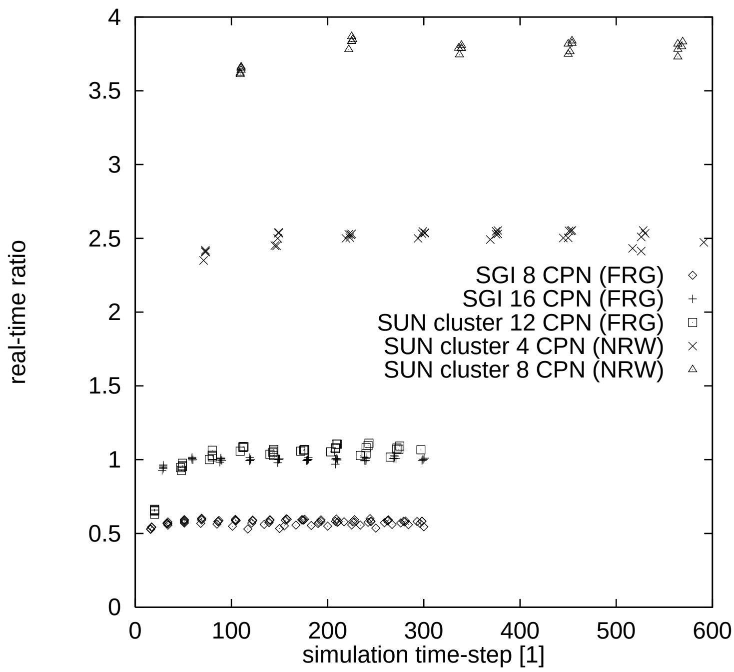

Figure 9: *Real-time Ratio*