

Distributed Platform for Large-Scale Agent-Based Simulations

David Šišlák, Přemysl Volf, Michal Jakob, and Michal Pěchouček

Agent Technology Center, Department of Cybernetics,
Faculty of Electrical Engineering, Czech Technical University in Prague
sislakt@fel.cvut.cz

Abstract. We describe a distributed architecture for situated large-scale agent-based simulations with predominately local interactions. The approach, implemented in AGLOBEX SIMULATION platform, is based on a spatially partitioned simulated virtual environment and allocating a dedicated processing core to the environment simulation within each partition. In combination with dynamic load-balancing, such partitioning enables virtually unlimited scalability of the simulation platform. The approach has been used to extend the AGENTFLY air-traffic test-bed to support simulation of a complete civilian air-traffic touching National Air-Space of United States. Thorough evaluation of the system has been performed, confirming that it can scale up to a very high number of complex agents operating simultaneously (thousands of aircraft) and determining the impact of different configurations of the simulation architecture on its overall performance.

1 Introduction

Agent-based simulations have recently become a popular way to model and study complex multi-actor systems, complementing the long-established system dynamics and discrete-event simulation approaches [3]. For a large number of agents, high complexity of their reasoning and/or high complexity of agent-environment interaction, a single machine might not be able to host the whole simulation. In such a case, an effective distribution schema is required that would enable the simulation to scale.

In this paper, we propose a particular distributed agent-based simulation architecture and describe AGLOBEX SIMULATION, a simulation platform implementing the architecture. The development of the platform has been motivated by the need to simulate, with great level of detail, full civilian air-traffic touching National Air-Space of the United States. The design and the implementation developed, however, provide a more universal recipe and can be applied to other large-scale (analytical) simulations requiring high simulation fidelity, scalability and maximum simulation speed.

The proposed approach is based on dynamic partitioning of the simulation of the virtual world into a variable number of dynamically-sized partitions, each handled by a dedicated environment simulation component. Thanks to the prevailing locality of agent-to-agent and agent-to-environment interactions, such

partitioning allows to effectively distribute the required computation between a number of machines while keeping the communication overhead low.

The proposed architecture has been applied for the simulation of the full civilian air-traffic touching National Air-Space of United States, requiring high-fidelity simulation involving the total of 52,799 airplanes with up-to 6,500 simultaneous aircrafts operating over the whole Earth – long flights departing, arriving or flying over the United States. The AGENTFLY [18] system implementing the proposed architecture is used for a thorough empirical evaluation.

The paper is organized as follows. After a review of the related work in Section 2, we define the class of simulation problems addressed and state the requirements of the suitable solution in Section 3. In Section 4, we describe the proposed approach in detail, starting with the architecture of the simulation and the structure of the simulation up to the description of the core spatial partitioning idea and its dynamically re-balanced extension. In Section 5, we briefly describe the implementation of the approach utilizing the high-performance AGLOBE multi-agent platform. Section 6 is the dedicated to an in-depth evaluation of the approach on the civilian air-traffic simulation domain. The effect of system load and the distribution of hardware resources among different components of the architecture are evaluated. Section 7 concludes with a summary and suggestions for future work.

2 Related Work

There are two basic approaches to agent-based simulations – *analytic simulation* and *distributed virtual environments*. The analytic simulation is typically used for high fidelity simulation of a given problem with a maximum *detail* and *accuracy* to provide results for further analysis of the problem. Analytic simulation runs typically without human interaction and thus a repeated simulation run with the same initial configuration leads to the same results. Furthermore, the relation to real-world time (referred to also as *wall clock*) is not important in analytic simulation and the simulation is therefore executed at maximum speed to deliver the results as soon as possible (this mode is also referred to as *as fast as possible* [6] mode). The simulation is synchronous and usually deterministic and can be based either on time steps or on events. The *time-stepped simulation* advances by predefined equally-sized (virtual simulation) time steps [8, 24] and new states of the simulation are computed after each such a step. The *event-driven simulation* uses events as a basis for its execution [2, 15]. Each event is scheduled with a time-stamp and the simulation framework advances to the time of the event with the earliest time-stamp and determines the new states.

The distributed virtual environments (DVEs) [23] are used to create a realistic illusion of real-world environment to a human user for purposes such as training or entertainment. Because of involvement of human users, DVEs are usually paced with wall clocks and do not produce the same results when repeated. They also usually do not support absolute synchronization and ordering of messages [12, 13, 25].

High level architecture (HLA) [5, 11] is an industry standard developed as an extension of DVE simulation. The architecture has been defined by a working group formed by the Defense Modeling and Simulation Office in order to standardize simulation development and allow inter-operability between various simulators produced by multiple vendors by using standardized simulation roles, predefined communication protocols and the data exchanging format.

The AGLOBEX SIMULATION described in this paper uses a combination of both the analytic and DVE approach. Agent’s communication, internal processes and non-physical interaction with other agents is asynchronous, non-deterministic, possibly containing a human in the loop, and the implementation of these processes therefore uses the DVE approach. The simulation of the environment, situated entities and physical interactions in the virtual world is synchronous and deterministic, and the analytic approach with maximal accuracy is therefore used. The simulations built on the platform can run in the as-fast-as-possible mode preserving the accuracy of the simulation together with asynchronous and non-deterministic behavior of the agents in faster than wall clocks paced time.

As far as our primary application area of interest, i.e. air-traffic simulation is concerned, both the analytic and DVE approaches are used. The DVE are primarily used for training purposes and not directly relevant to our objectives. Analytic simulation, on the other hand, is used for the purposes of obtaining insight into various processes and phenomena related to air traffic. Numerous simulations with varying level of detail, different purposes and coverage have been developed. Agogino [1] uses agent-based simulation for incremental enhancements of the current *air-traffic management* (ATM). Krozel [10] uses the *FACET* simulation system to model air-traffic in inclement weather. Gorodetsky [7] uses an agent-based simulation for ATM within the air-space of large airports. Hwang [9] uses simulation for verification of collision avoidance algorithms. All of the above systems focus on simulating a particular subset of the overall air traffic. On the other hand, the AGENTFLY system built using the AGLOBEX SIMULATION platform aims to provide a model of the entire air traffic. The huge computational requirements resulting from this goal were consequently one of the driving forces behind the development of the presented distributed simulation platform.

3 Simulation Problem and Requirements

In this section, we describe the class of simulations addressed by this work and we specify the requirements on the solutions.

3.1 Simulation with Localized Interactions

We are addressing simulations involving large numbers of *situated entities*¹, both *autonomous* and *passive*, operating and interacting in a realistically modeled

¹ By situated entities we mean that they are embedded in a synthetic model of a 3D physical space.

large-scale *virtual world*. Given our motivating application, we further assume that the virtual world is an Earth-like 3D environment limited by a maximum altitude. The entities need not be present during the whole simulation but can be dynamically introduced or removed based on the simulation needs.

Each situated entity in the simulation carries a *state*, components of which can be either (i) *observable* i.e. public and external or (ii) *hidden* i.e. non-observable, private and internal. The fundamental component of the entity's observable state present in the simulation is its position and orientation in the virtual world.

The evolution of the entity's state is governed by the defined entity's *physics*, e.g. entity movement is driven by its motion dynamics typically defined by a set of differential equations, which can also refer to entity's hidden state components. Physics can be divided to *intra-entity* and *inter-entity*. The intra-entity physics capture those aspects of physical dynamics that can be fully ascribed to a single entity. Although the equations of the intra-entity physics can refer to any states in the simulated world, they only govern the states carried by its respective entity. In contrast, the inter-entity physics captures the dynamics that affects multiple entities simultaneously and which cannot be fully decomposed between them (consider e.g. the effects of a physical collision of two entities).

In addition to physical interactions, autonomous entities can also interact via communication, i.e. exchange electronic messages, and by using *sensors* to perceive their surrounding virtual environment. Communication between entities can be restricted by the inter-entity physics simulating required communication medium, e.g. wireless network. Similarly, each sensor has its capabilities defined which may restrict the observable state it can perceive only to a defined subset, typically based on the distance from the sensor's location (e.g. a limited range of a radar).

3.2 Requirements

Having defined the type of simulations addressed and the necessary terminology, we can now state the desired properties of a simulation platform:

- *High fidelity* – The simulation has to be accurate with respect to the defined physics. Simulation *level of details* (LOD) [4] techniques cannot be used to reduce the complexity of used intra- or inter-physics as this could lead to significant errors in the simulation output.
- *Scalability* – The simulation platform has to be scalable in terms of the number of simulated situated entities (both the number of simultaneously running entities and their total number) and the virtual world dimension. Distributed architecture of the simulation should allow to scale the simulation up simply by increasing the number of computing resources used for the simulation.
- *Maximum simulation speed* – The high-performance simulation platform should be able to produce results for a given simulation scenario in minimum possible time without sacrificing the accuracy of the high fidelity simulation. In other words, the speed with which the simulation is executed should not affect simulation results.

4 Approach and Architecture

We now describe the key components of the approach to large-scale simulations which we implemented within the AGLOBEX SIMULATION platform.

4.1 Situated Entities

In our approach, any entity consists of up to three subcomponents:

- *Body* – The entity’s body encapsulates entity’s intra-physics which governs the evolution of entity’s state as a function of other, possibly external states. Typically, the body defines motion dynamics for the entity.
- *Reactive control* – The entity’s reactive control contains real-time loop-back control for the entity. The loop-back control affects entity’s states based on the observation of other states. Reactive control e.g. handle urgent reactive behaviors (such as avoiding an obstacle) triggered in emergency situations.
- *Deliberative control* – The entity’s deliberative control contains complex algorithms employing planning, prediction and communication with other entities. The outputs of deliberative control typically feed back into and controls the entity’s reactive control module. Typically, complex deliberative algorithms providing high-level entity control are invoked based on the sensing perceptions.

Due to their principal similarity in loop-back approach, the body and the reactive control are collectively referred to as entity’s *state updater*.

Entities can be either *autonomous* or *passive*. Autonomous entities, termed *situated agents*, have at least one and typically both control modules. Situated agents are used to represent pro-active, goal-oriented actors in the simulation. Passive entities only possess the body and thus they represent non-autonomous objects whose behavior in the virtual world is fully defined only by their physics.

4.2 System Architecture

All components in the simulator architecture are divided into two groups, see Figure 1:

- *Environment simulation components* (ESCs) – These components are responsible for application of all entities’ state updaters and also for the updates resulting from the inter-physics.
- *Deliberative control components* – The deliberative control part of each situated agent is encapsulated in a separate deliberative control component. Deliberative parts can communicate via component-to-component message transmission if this is allowed by inter-physics governing the communication medium.

Even though the deliberative control part and the state updater are decoupled in the simulator architecture, the deliberative control and the state updater of each entity is connected by a signal channel through which sensing perceptions (one way) and control commands to the reactive control (the other way) are transmitted.

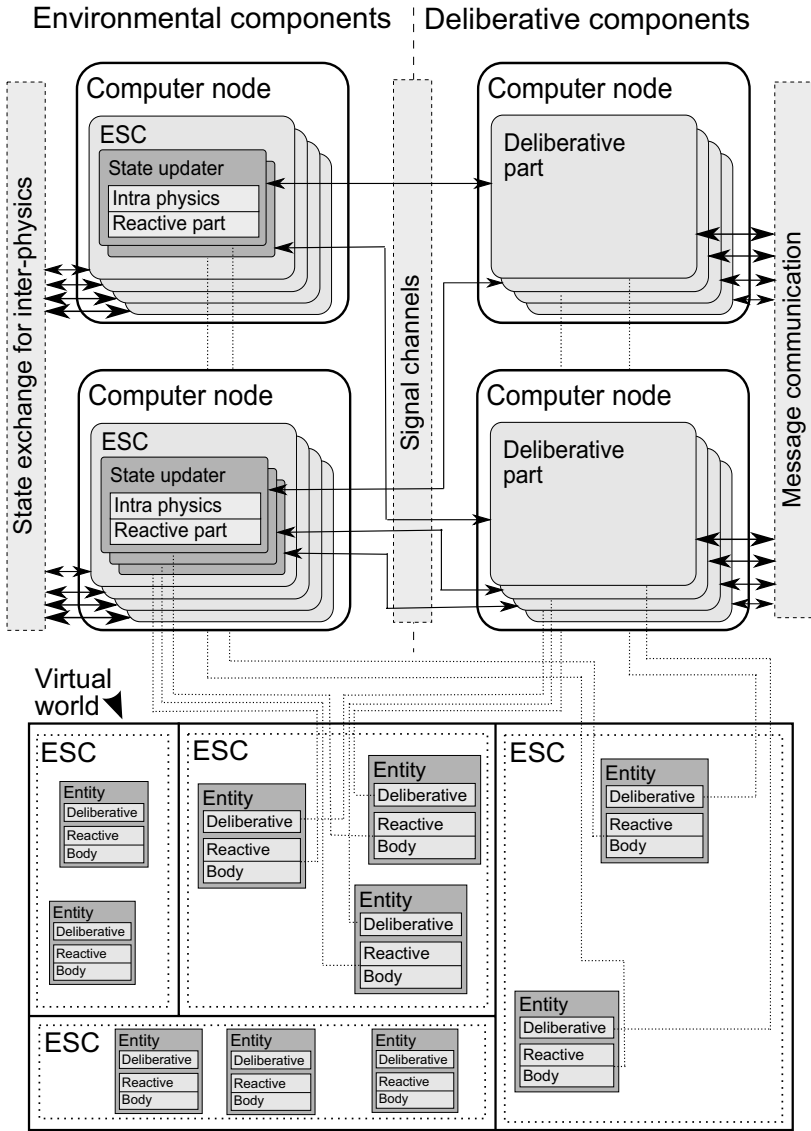


Fig. 1. Architecture overview of the distributed AGLOBEX SIMULATION platform

4.3 Time-Stepped Simulation

The presented simulation platform employs a time-stepped simulation approach [6]. The virtual simulation time driving the dynamic processes of the simulation entities is incremented uniformly by a constant time step. Defined physics and loop-back control from the reactive parts are expressed in terms of difference equations; such equations define the entity’s state (after the time increment) as

a function of the current state only. All these difference equations, and therefore all physics and reactive parts, have to be evaluated synchronously in order to provide consistent evolution of the virtual world. Synchronous evaluation means that states are updated only after difference equations (relevant in the given moment in time) have been evaluated.

The above virtual time steps and thus the resulting state updates can be applied regularly with respect to the external wall clock time, or in a *as-fast-as possible* manner. In the *as-fast-as possible* mode, the next time step is initiated just immediately after the previous one has been completed. This mode therefore allows to complete a given simulation scenario in the shortest possible time.

4.4 Distributed Simulation and Environment Partitioning

Both the environment simulation and deliberative control components can be distributed among multiple computer nodes during the simulation. The whole virtual simulation world is spatially divided into a number of partitions. The partitions are mutually disjoint except for small overlapping areas around partitions' boundaries. Each partition has a uniquely assigned environment simulation component (ESC) which is responsible for updating states (i.e applying of relevant difference equations) corresponding to all entities located within the partition. The application of intra-physics may require exchange of state information between multiple partitions, in case the entities it affects are not located within the same partition.

In contrast with the state updaters, entity's deliberative control modules are deployed in the simulation grid (computer nodes) irrespective of the partitions and corresponding entity's position in the virtual world.

The above virtual world partitioning implies that whenever the location of an entity changes so that the entity moves (spatially) to a new partition, the entity's state updater needs to be migrated to ESC responsible for the new partition. In contrast, the relatively heavy-weight deliberative control modules are never moved between computer nodes. The signal channel between the entity's reactive control and its deliberative control module is transparently reconnected and it is guaranteed that no signal transmission is lost during the migration of a state updater between two ESCs. The small overlapping areas around partition boundaries introduced above are used to suppress oscillatory migration of state updaters between neighboring ESCs in the case when the corresponding entity is moving very close along partition borders.

4.5 Simulation Cycle

The consistent evolution of the whole virtual world is achieved by coordinated operation of all ESCs. Each simulation cycle is decomposed into the following phases:

1. *Evaluation of state updaters* – During this phase, ESCs determine new state values as a result of difference equations defined in intra-physics and reactive control modules of respective entities.

2. *Evaluation of inter-physics* – In this phase, ESCs exchange state information required to calculate state updates controlled by inter-physics (and therefore possibly crossing partition boundaries)
3. *Migration of state updaters* – Entities whose location after the state update no longer belongs to the partition corresponding to their hosting ESCs are migrated to the appropriate ESCs.
4. *Application of new states* – From this moment, all pre-computed new state values are considered as current ones.

The next phase of the simulation cycle is invoked only once the current phase is completed by all ESCs.

4.6 Dynamic Load-Balancing

In order to provide maximum performance throughout the whole simulation, the proposed architecture implements load-balancing. Load-balancing related to both environment simulation components (through dynamic re-partitioning) and deliberative control parts (through balanced startup deployment) is supported.

The world decomposition to partitions is not fixed during the simulation but is dynamically updated depending on each partition's host's load. Based on the time required for processing of the first two phases of the simulation cycle (see above) by each ESCs, the world is re-partitioned so that the number of entities belonging to partitions is proportional to the measured times. ESC performing the first two phases faster will be assigned a larger area with more situated entities and thus more state updaters and vice versa. If there are no situated entities in the simulation, partitions are equally distributed covering the whole virtual world.

The above described re-partitioning is performed before the migration of state updaters (phase 3 of the simulation cycle). After re-partitioning, all state updaters which no longer belong to their currently assigned ESC partition are subsequently migrated to the appropriate ESC. Re-partitioning is not performed during each simulation cycle but only if the difference in state update evaluation times by all ESCs exceed a pre-defined threshold. The re-partitioning is also triggered whenever a new computer node is allocated to environment simulation.

The load of computer nodes running deliberative components is balanced only when new deliberative components are instantiated. Deliberative parts of newly introduced entities are spread proportionally among computer nodes according to individual nodes' average load in several previous simulation cycles, expressed as the sum of each node's idle time over those simulation cycles. The computer node with the highest idle time will receive more deliberative control modules and vice versa. There is no dynamic re-balancing of already running deliberative control modules because it would introduce issues with component-to-component addressing and proper conversation protocol consistency during the migration.

5 Implementation

The distributed simulation architecture described in the previous section has been implemented on top of the AGLOBE multi-agent platform [21] and is part of the the AGENTFLY [18] multi-agent system for air-traffic control and planning.

AGLOBE is a multi-agent middleware supporting agent encapsulation, effective agent-to-agent communication, high-throughput message passing, message topicking, yellow pages services, agent full migration, but also agent life-cycle management. AGLOBE is well suited for scalable multi-agent simulation as it supports integration of simulation components. AGLOBE requires limited computational resources and its operation is very efficient. AGLOBE also facilitates modeling communication inaccessibility and unreliability in ad-hoc networking environment. AGLOBE has been used for number of applications and agent system prototypes in the area of air traffic control, car electronic modeling, design process simulation or network intrusion detection.

In the designed distribution simulation platform, each ESC and each deliberative control module are encapsulated as an AGLOBE software agent. Beside agents that represent ESCs and deliberative control modules, there are also other agents which take care of simulation maintenance and coordination among distributed system.

AGENTFLY [18] is a complex multi-agent system for modeling and simulation of air-traffic, involving both manned and unmanned aircraft. The framework provides models of aircraft physics, advanced trajectory path planning algorithms [22], multi-layer collision avoidance implementing both cooperative and non-cooperative collision avoidance [20], integration interfaces for external data sources, high-fidelity 3D visualization and data collectors for post-run analysis tools. AGENTFLY has been integrated with the algorithms for planning tactical missions, information collection and surveillance [19]. The system scalability has been dramatically enhanced, by integration of the presented distributed simulation platform, in order to answer commercial traffic simulation requirements.

6 Evaluation

We have evaluated the proposed distributed simulation architecture on a large-scale simulation of civilian air-traffic (see the domain description below). The objective of the experiments were two-fold: (i) to understand the performance of the simulation platform with constant computing resources under varying level of system load, i.e. the size/complexity of the simulation scenario (Section 6.4), and (ii) to understand the performance of the simulation system under varying distribution of the available resource between key platform components (Section 6.5).

6.1 U.S. National Airspace Traffic Domain

The proposed simulation platform has been deployed for the simulation of civilian air-traffic within U.S. National Airspace (U.S. NAS). The resulting airspace

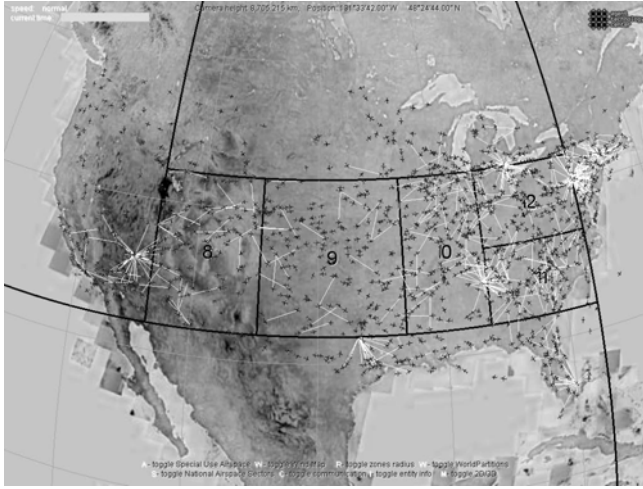


Fig. 2. Airplanes over U.S. NAS and world partitioning

simulator is used for studying concepts for the Next Generation Air Transportation Systems (NGATS) [14] in cooperation with the Federal Aviation Administration (FAA).

Each civilian flight operated under Instrumented Flight Rules (IFR) [16] is represented as a situated entity within the simulation. One simulation run typically covers one day and simulates operation of all civilian IFR flights which traverse, fully or partially, U.S. NAS. Even though the area of the study is limited to U.S. NAS, some effects external to U.S. NAS are also studied, covering almost the whole Earth for long-distance flights. The overall number of simulated airplanes is 52,799 (up to 6,500 are simultaneous) corresponding to real air-traffic for a particular day in 2007, see Figure 2. Depending on the airplane type, each entity uses a precise flight model based on the BADA airplane performance models [17].

The distribution of the number of airplanes during the simulation day time is depicted in Figure 3. Note that there are significant peaks every half an hour of the simulation time; these peaks lead to extreme load of the system with more than 300 airplanes created at one simulation cycle in the highest peak of the day. For each such a newly created airplane, a corresponding situated entity needs to be spawned and the computation-intensive flight trajectory planning and flight control algorithms executed.

The simulation cycle is 12 seconds long, which is suitable for this domain. A scalable simulation platform is required because the system is used for analysis of concepts related to the predicted growth of air traffic in the upcoming decades (i.e. a simulation of the expected density of traffic in 5, 10, 15, and 20 years).

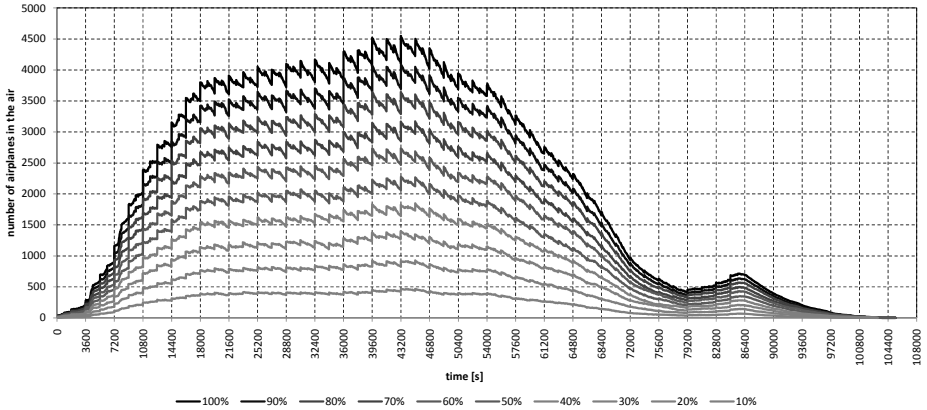


Fig. 3. The number of simultaneous airplanes during the simulation

6.2 Metrics

The *run-time* is the total real (wall clock) time needed for the simulation multiplied by the number of processor cores used during the simulation.

The *normalized airplane* is a fictive airplane that is flying during the whole simulation from its start to the end. The number of airplanes is changing during the simulation, thus the mean of the number of airplanes is used as the number of normalized airplanes running during the whole simulation. This number is used to normalize run-time to study requirements per each situated entity.

6.3 Testing Configuration

The configuration contained a set of several heterogenous (both in terms of the operating system and hardware) computer nodes. During experiments, nodes were dedicated for ESCs or deliberative controllers, see computer node allocation in Figure 4:

The testing configuration consisted of 9 cores with 9GB RAM for ESCs and 11 cores with 15GB RAM for the flight control integrated in deliberative controllers. All computers were interconnected through a 1 Gbit Ethernet network (connection through a single switch) using the TCP/IP protocol. All systems were running Sun Java 64-bit 1.6.0_14 environment. All empirical values are averaged from five independent measurements with identical initial conditions, e.g. a freshly launched JVMs, no allocated memory etc.

6.4 Overall System Load

The objective of the first experiment was to explore how the speed of the simulation changes with the changing level of system load. The experiment was performed using all computer nodes. The variations in system load were introduced by varying the number of airplanes simulated, ranging from 10% up to 100% of all air-traffic (with 10% step).

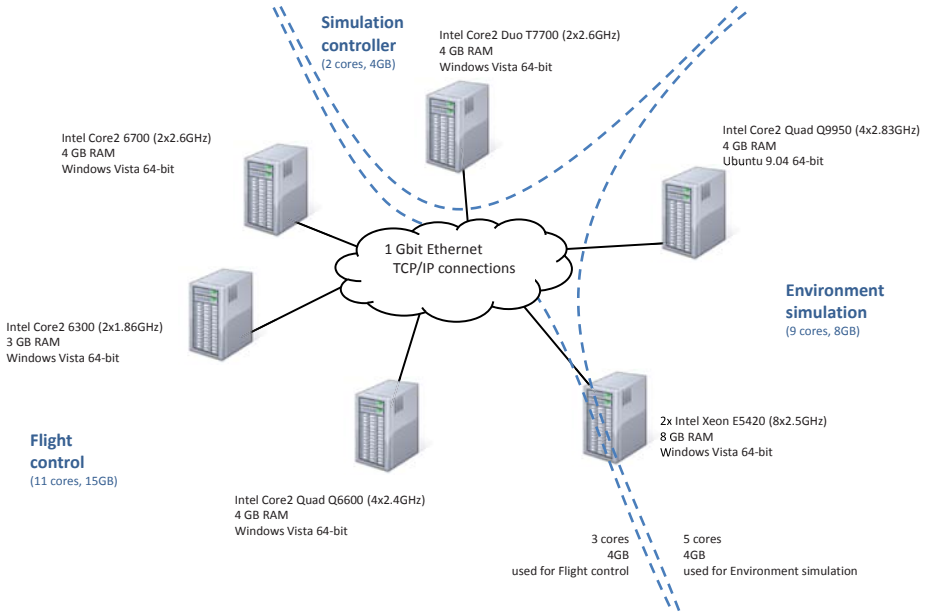


Fig. 4. Computer nodes allocation during experiment: *environment simulation* – hosts environmental components, *flight control* – hosts all deliberative control parts and *simulation controller* – other components for simulation control and input/output interfacing to the simulation

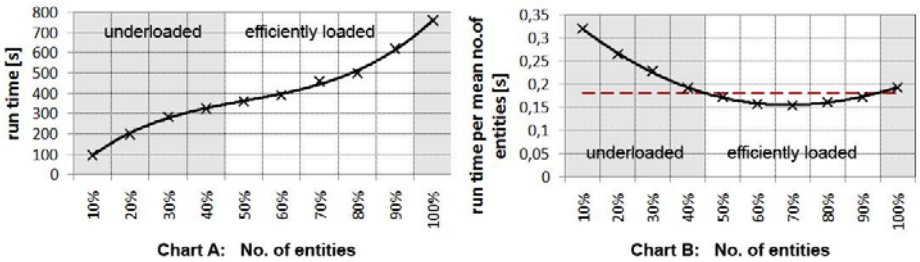


Fig. 5. Effect of the increasing number of flights simulated with fixed hardware resources, measured in terms of the total duration of the simulation (Chart A) and run-time per normalized airplane (Chart B)

The aggregated results of the experiment are depicted in Figure 5 showing the correspondence between the total duration of the simulation and the number of flights. Chart A shows the maximum duration is less than 13 minutes for 100% of flights. Chart B shows run-time needed to simulate one normalized airplane.

The experiment demonstrates properties and effectiveness of the distributed simulation. The measured characteristics can be divided into three regions

depending on the system load (the number of airplanes): (i) underloaded, (ii) optimally loaded and (iii) overloaded simulation. The regions can be determined by the specified threshold, a dashed horizontal line at 0.18 s in Figure 5, Chart B.

The upper-left part of the plot above the threshold corresponds to the underloaded simulation, the part below the threshold is the optimally loaded simulation and the upper-right part above the plot is the overloaded simulation.

Underloaded Simulation. At low levels of system load (few situated entities), the simulation is underloaded and thus inefficient. The hardware resources cannot be effectively used because the load of the simulation is too low (e.g. the simulation running 5 airplanes cannot effectively use 20 processor cores), but there is an overhead given by the maintenance of distributed architecture which depends both on the number of processors cores and the number of airplanes in the system. The shape of the load line in Figure 5, Chart B is concave as the overhead of the distributed simulation is split among the increasing number of airplanes. The underloaded simulation leads to higher run-time per normalized airplane.

Optimally loaded Simulation. In the optimally loaded configuration, the simulation is able to use all available hardware resources effectively. This section begins at the moment when all hardware can be used and ends when some components start to be overloaded. Within the optimally loaded region, the increasing load of the simulation is distributed regularly among all resources, leading to almost linear increase of the simulation run-time.

Overloaded Simulation. Moving towards the highest load of the fixed computer configuration, the hardware resources become overloaded. The overload has three main reasons:

CPU overload. The simulation management is designed to run the simulation as fast as possible considering the CPU load. If the CPU performance is not sufficient, the simulation continuously slows down until the whole simulation almost stops.

Memory overload. Once the simulation requires more than available physical memory, this results in higher probability of page faults and causes swapping which leads to decrease of the simulation performance.

Network overload. Despite the distributed design which inherently aims to minimize the network traffic, the network can get overloaded. If the network is not able to transmit all traffic required by the simulation especially related to the environment simulation, the simulation is paused and waits for their delivery.

In the experiments, usually both CPU and memory resources were overloaded. Thus the simulation gradually slows down at the beginning of the overload and with higher load it slows down quickly.

6.5 Varying Resource Distribution

The aim of the second group of experiments was to assess the performance impact of different distributions of available hardware resources between environmental and deliberative control components in the simulation. Both experiments used a simulation load consisting from 40% of the flights.

The first experiment was performed using the full test hardware configuration for the deliberative control parts (see Figure 4) and varying hardware configuration for the environmental components. The second experiment was performed using the full test hardware configuration for the environmental components and varying hardware configuration for the deliberative control parts.

The results for the first and second experiment are depicted in Figures 6 and 7, respectively. The left plots in both Figures show how the simulation run-time decreases with adding hardware resources. The right plots in both Figures show changes in normalized run-time.

The first experiment (Figure 6) shows the dependency between the overall performance and the number of computational cores assigned to the environment components and thus resulting in different numbers of partitions in the simulation. The left region of the charts (one, two and three partitions) shows a significant speed-up in both the simulation run-time and normalized run-time. This is because the small number of partitions in the simulation are overloaded. Adding more hardware and thus more partitions (three, four and five partitions) corresponds to the optimally-utilized simulation. The simulation run-time is still reduced, however, the normalized run-time stagnates. This is caused by rising demands for partition-to-partition coordination. Adding even more resources brings almost no speed-up in the simulation run-time and even results in a decreasing normalized run-time. The reason is that the simulation configuration is unable to fully-utilize the newly introduced partitions in comparison to increasing overhead for partition-to-partition coordination.

The second experiment (Figure 7) shows a similar picture for the dependency of the simulation run-time on the number of available hardware resources for deliberative controllers. In the left part (two, three and four cores), it shows a

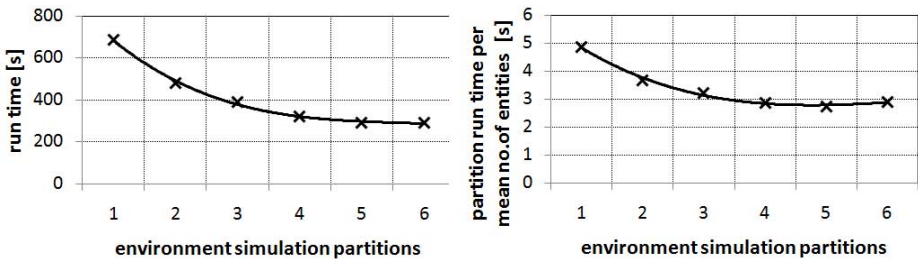


Fig. 6. The effect of changing the hardware resources allocated to environment components while keeping the resources allocated to the rest of the system constant; the effect measured in terms of the total simulation run-time (left) and total run-time per normalized airplane (right)

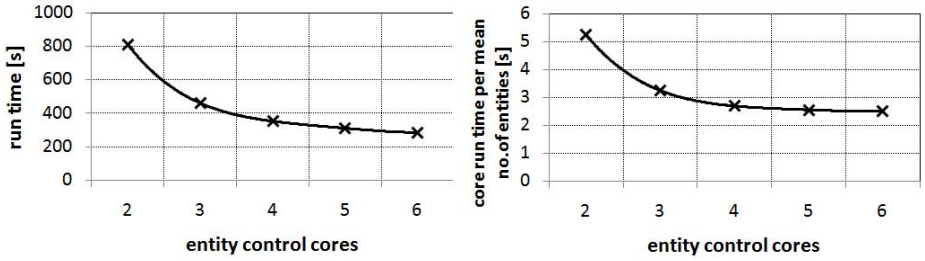


Fig. 7. The effect of changing the hardware resources allocated to deliberative control parts while keeping the resources allocated to the rest of the system constant; the effect measured in terms of the total simulation run-time (left) and simulation run-time per normalized airplane (right)

significant speed-up for the same reasons as in the previous case. The different property is for more hardware resources (four, five and six cores). In such a case, the simulation stays optimally loaded, the simulation run-time decreases and the normalized run-time remains constant. The reason is that deliberative controllers are independent and do not require any additional coordination. All computational resources can be therefore used for parallel computation. At a certain moment, we can identify that there is no use in adding more hardware resources for the deliberative controllers as the bottle neck is in the environment part for the given simulation configuration.

The simulation performance is maximized by the proper distribution of available resources among environmental components and deliberative controllers. Under-resourcing of any part leads to a very significant decrease of the overall system performance. Over-resourcing of the deliberative controllers has no effect on performance, but it can help when the load to the system increases. Over-resourcing of the environment controllers can cause decrease of the overall performance due to an additional coordination overhead induced by a higher number of partitions.

7 Conclusion

We have investigated the problem of efficient simulation of a huge number of situated agents operating in large-scale virtual worlds. A distributed, fully scalable approach with automated load balancing has been proposed to address the problem. The approach efficiently exploits the predominantly local nature of interactions in situated agent-based simulations and uses it to efficiently partition the virtual world and distribute it over multiple computer nodes. The approach has been implemented in a AGLOBEX SIMULATION agent-based simulation platform based on AGENTFLY test-bed, its specific application to the air-traffic domain, and evaluated on the simulation of the complete civilian air-traffic touching the U.S. NAS involving the total of 52,799 flights with up to 6,500 airplanes simulated at one time. The evaluation confirms the ability of

the approach to handle very large-scale agent-based simulations and highlights the need to carefully assign hardware resources to individual components of the simulation platform.

Acknowledgement

The work has been supported by the Federal Aviation Administration (FAA) under project number DTFAC-08-C-00033 and by Czech Ministry of Education under grant number 6840770038. The underlying AGENTFLY system was supported by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-06-1-3073. The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the Federal Aviation Administration, the Air Force Office of Scientific Research or the U.S. Government.

References

- [1] Agogino, A., Tumer, K.: Regulating air traffic flow with coupled agents. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, vol. 2, pp. 535–542. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC (2008)
- [2] Banks, J., Carson, J.S., Nelson, B.L., Nicol, D.M.: Discrete-event system simulation. Prentice Hall, Upper Saddle River (2001)
- [3] Borshchev, A., Filippov, A.: From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools. In: Proceedings of the 22nd International Conference of the System Dynamics Society, pp. 25–29 (2004)
- [4] Brom, C., Šerý, O., Poch, T.: Simulation level of detail for virtual humans. In: Pelachaud, C., Martin, J.-C., André, E., Chollet, G., Karpouzis, K., Pelé, D. (eds.) IVA 2007. LNCS (LNAI), vol. 4722, pp. 1–14. Springer, Heidelberg (2007)
- [5] Dahmann, J.S., Fujimoto, R.M., Weatherly, R.M.: The department of defense high level architecture. In: Proceedings of the 29th conference on Winter simulation, pp. 142–149. IEEE Computer Society, Washington (1997)
- [6] Fujimoto, R.M.: Parallel and Distribution Simulation Systems. John Wiley & Sons, Inc., New York (1999)
- [7] Gorodetsky, V., Karsaev, O., Samoylov, V., Skormin, V.: Multi-Agent Technology for Air Traffic Control and Incident Management in Airport Airspace. In: Proceedings of the International Workshop Agents in Traffic and Transportation, Portugal, pp. 119–125 (2008)
- [8] Guo, Y., Gong, W., Towsley, D., Amherst, M.A.: Time-stepped hybrid simulation (TSHS) for large scale networks. In: IEEE INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, vol. 2 (2000)
- [9] Hwang, I., Kim, J., Tomlin, C.: Protocol-based conflict resolution for air traffic control. Air Traffic Control Quarterly 15(1), 1–34 (2007)

- [10] Krozel, J., Doble, N.: Simulation of the National Airspace System in Inclement Weather. In: AIAA Modeling and Simulation Technologies Conference, Hilton Head, SC (2007)
- [11] Kuhl, F., Weatherly, R., Dahmann, J.: Creating computer simulation systems: an introduction to the high level architecture. Prentice Hall PTR, Upper Saddle River (1999)
- [12] Lui, J.C.S., Chan, M.F.: An efficient partitioning algorithm for distributed virtual environment systems. *IEEE Trans. Parallel Distrib. Syst.* 13(3), 193–211 (2002)
- [13] Morillo, P., Ordufia, J.M., Duato, J.: A scalable synchronization technique for distributed virtual environments based on networked-server architectures. In: 2006 International Conference on Parallel Processing Workshops, ICPP 2006 Workshops, p. 8 (2006)
- [14] National Research Council Panel on Human Factors in Air Traffic Control Automation. *The Future of Air Traffic Control: Human Factors and Automation*. National Academy Press (1998)
- [15] Nicol, D.M., Yan, G.: Discrete event fluid modeling of background TCP traffic. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 14(3), 211–250 (2004)
- [16] Nolan, M.S.: *Fundamentals of Air Traffic Control*, 4th edn. Thomson Brooks/Cole, Belmont (2004)
- [17] Nuic, A., Poinot, C., Iagaru, M.G., Gallo, E., Navarro, F.A., Querejeta, C.: Advanced Aircraft Performance Modelling for ATM: Enhancements to the BADA Model. In: *Beitrag zur 24th Digital Avionics System Conference*. DASC, Washington (2005)
- [18] Pěchouček, M., Šišlák, D.: Agent-based approach to free-flight planning, control, and simulation. *IEEE Intelligent Systems* 24(1) (2009)
- [19] Semsch, E., Jakob, M., Pavlíček, D., Pěchouček, M., Šišlák, D.: Autonomous uav surveillance in complex urban environments. In: McGann, C., Smith, D.E., Likhachev, M., Marthi, B. (eds.) *Proceedings of ICAPS 2009 Workshop on Bridging the Gap Between Task and Motion Planning*, Greece, September 2009, pp. 63–70 (2009)
- [20] Šišlák, D., Volf, P., Komenda, A., Samek, J., Pěchouček, M.: Agent-based multi-layer collision avoidance to unmanned aerial vehicles. In: Lawton, J. (ed.) *Proceedings of 2007 International Conference on Integration of Knowledge Intensive Multi Agent Systems, KSCO*. IEEE, Los Alamitos (2007)
- [21] Šišlák, D., Reháček, M., Pěchouček, M., Rollo, M., Pavlíček, D.: Aglobe: Agent development platform with inaccessibility and mobility support. In: Unland, R., Klusch, M., Calisti, M. (eds.) *Software Agent-Based Applications, Platforms and Development Kits*, Berlin, pp. 21–46. Birkhauser Verlag, Basel (2005)
- [22] Šišlák, D., Volf, P., Pěchouček, M.: Accelerated a* path planning. In: *The Eighth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Budapest, Hungary (May 2009)
- [23] Waters, R.C., Barrus, J.W.: The rise of shared virtual environments. *IEEE Spectrum* 34(3) (March 1997)
- [24] Wu, Y., Gong, W.: Time-stepped simulation of queueing systems. In: *Proceedings of SPIE*, vol. 4367, p. 262 (2001)
- [25] Zhou, S., Cai, W., Lee, B.S., Turner, S.J.: Time-space consistency in large-scale distributed virtual environments. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 14(1), 31–47 (2004)