Master Seminar Paper

Christian Hüning, christian.huening@haw-hamburg.de

Scalability of the Massive Multi-Agent System MARS

Christian Hüning, christian.huening@haw-hamburg.de

Scalability of the Massive Multi-Agent System MARS

Master Seminar Paper eingereicht im Rahmen der Master Seminar Paper

im Studiengang Master of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Thiel-Clemen

Eingereicht am: 27. Februar 2015

Christian Hüning, christian.huening@haw-hamburg.de

Thema der Arbeit

Scalability of the Massive Multi-Agent System MARS

Stichworte

Multiagentensystem, Ökologie, Verteilte Systeme, Lastverteilung, MMAS, MSaaS

Kurzzusammenfassung

Christian Hüning, christian.huening@haw-hamburg.de

Title of the paper

Scalability of modern Multi-Agent Systems

Keywords

Multi-Agent-System, Ecology, Distributed Systems, Load Balancing, MMAS, MSaaS

Abstract

Contents

1 Introduction

The identified gap to build a scalable and yet accessible multi-agent system (Hüning *et al.*, 2014), still is very present, as it has moved into focus of several researchers at last years WinterSim conference (Taylor *et al.*, 2012, 2013; Tolk & Mittal, 2014). Especially the definition of MSaaS (Modeling and Simulation as a Service) by Cayirci (2013) and first follow-up papers (Padilla, 2014) expresses the interest in and necessity of cloud-based simulation. It is whished for MSaaS to deliver scalable simulation execution by means of a simple enough user interface suitable for domain experts.

The MARS simulation framework is very well aligned with the trend to move large-scale simulations into the cloud and offer a user friendly web interface for modelers, in that MARS is conceptualized as a chain of processes. Theses processes are either user-centric and thus integrated into a web application or part of the simulation runtime environment and thus deployed into the distributed cloud environment.

To achieve fast, scalable cloud-based large-scale simulation execution the underlying distribution and communication mechanism is crucial. It requires an efficient way to send and retrieve messages, allow to easily locate other entities (agents) and should address usability issues like communication transparency.

This paper introduces the Agent Shadowing (AS) approach, which is used in the MARS framework to achieve both, scalability and usability alike. Chapter 2 provides a brief conceptual and technological overview, chapter 3 presents the simulation model and how it will be used as the benchmark for the proposed solution. First results are presented in chapter 4, risks and chances are discussed in chapter 5. Finally in chapter 6 a conclusion is drawn and an outlook towards the author's thesis is provided.

2 Concept & Technology

2.1 MARS LIFE Overview

The actual simulation component in the overall MARS architecture (see Hüning *et al.* (2014)) is called LIFE. It consists of two main processes which make up the distributed simulation system. An exemplary deployment of the system's components is shown in figure 2.1.
The SimulationManager is the controlling application for the simulation. It manages the model, calculates the distribution and scheduling pattern, takes care of the distributed initialization and finally controls the simulation run.
The LayerContainer houses layers and agents, which are the two primary components MARS simulations are made of. A LIFE system may be composed of any number of LayerContainers among which layers and agents are distributed.

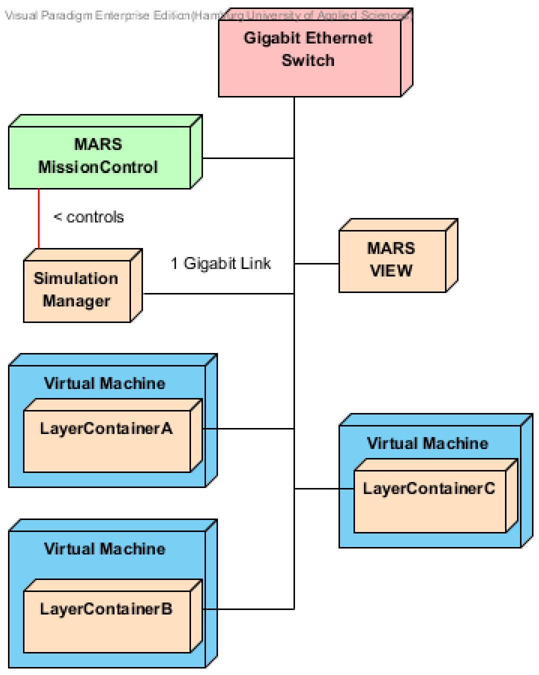

Figure 2.1: Exemplary deployment of MARS system

2.2 Distribution

Distribution and thus communication are two key aspects of scalability. In a very early version of the MARS system layers were only distributable as a whole, so each LayerContainer needed to take care of one ore more complete layers. The author's findings however have shown, that one layer may be too complex for a single computer or there may be some rather slow compute nodes involved. So the new approach will also allow to distribute each layer across several LayerContainers, which resembles true horizontal scalability. Since distributing layers has direct influence on the agents living on them, the approach for layer distribution is tightly coupled with the distribution of agents and is meant to make the overall system scalable. That approach is called Agent Shadowing (AS).

The problem of synchronization and communication between agents arises, especially when it comes to agent interaction or movement across boundaries (Vigueras *et al.*, 2013). Specifically whenever an agent wants to communicate (that is: call a method) with another agent, it first has to check whether the desired agent is present locally or remotely and, if remotely, obtain a reference through which it may perform the actual communication. By that each method call results in a (potentially slow) remote message.

2.3 Concept

Agent Shadowing is the depiction of an agent living on layer instance A having its shadow cast onto layer instance B, where it is not actually instantiated, but instead is represented by a stub-like object as in remote communication concepts like RPC/RMI.

2.3.1 Agents

In RPC/RMI each agent's methods and properties are callable by third parties through its stub object. Usually a stub just provides the capabilities to establish an interface-bound communication with the remote object. If the remote reference changes, in classic RPC/RMI the stub simply becomes useless, since its reference is not updated. The protocol then has to notice the broken link and re-establish a new one.

A shadow agent stub (SAS) is extended by the ability to hold cached attributes like its environmental position or any other attribute. The real agent object updates both, the attributes and the remote reference, whenever a change occurs. These updates are delivered via multicast when in LAN to reduce the amount of traffic. The initial remote references can be provided when the overall system is initiated since some kind of distribution information has to be provided at that state. This results in each container node containing the full environment as well as all agents, but with the difference, that only numberOfAgents / numberOfNodes (given an even partitioning) agents are really instantiated and thus have to be computed. The remaining agents are only instantiated as SASs and do not contain any agent behavior logic. An increase in container nodes would reduce the amount of agents per node that have to be actively computed, while the memory footprint per node would also potentially decrease, assuming that a SAS consumes less RAM than a full-fledged agent.

2.3.2 Layers

Calling or referencing another layer, works by the same pattern of either having a local instance of that layer to address directly or a stub to communicate with a remote reference. Environmentally it does not matter which remote reference to a layer is provided, since each layer instance locally has a full view of its state, even if distributed.

2.4 Communication & Network

Each SAS requires a unique connection to the real agent. This results in a unique port and socket being used on the client machine. Depending on which OS you are on, the maximum number of simultaneous TCP connections is limited. For example on Windows 7 you could theoretically create up to 16 million concurrent connections, but are limited by port restrictions to something closer to several thousand connections 3. It thus is obvious that this is not a suitable approach for a massive scale multi-agent simulation with possibly a million agents. Other approaches towards the network implementation need to be taken.

An alternative is to change the communication technology from TCP Unicast to UDP Multicast. A real agent would send its state update only one time per multicast to a group of potentially interested listeners, who then all receive the message and decide individually whether or not to consume it. This solution requires the creation of multicast groups, which could be created either on a per agent, per agent type or per layer base.

A group per agent wouldn't do much to the port problem mentioned above, since each multicast socket listener would still require its own port. Grouping by layer could result in too many messages received by too many SAS which don't need it (if a layer holds more than one agent type), or could create a bottleneck if the amount of agents is very high on each layer instance. Possibly the best result can be achieved with a grouping based on agent type, since the recipient group is narrowed down to those agents (and hosts) who can use the message. Through that kind of interest management the bottleneck problem should directly scale with further distribution of the layer itself.

As an additional side effect of the per agent type multicast messaging, updating the remote references of stubs in case of cross-boundary movement of agents, will not be necessary anymore. Since each individual agent can be reached by a multicast group uniquely generated from the agent's type and multicast messaging inherently does not care about endpoints, real agents may be moved across nodes without regard to references.

3 A model to benchmark LIFE

To test-drive MARS LIFE including agent shadowing, a real-world scenario (Scholes *et al.*, 2001) from a model under active research in the MARS group has been selected. The model incorporates a 4 km by 4 km area located in Skukuza, which is the administrative headquarters of the Kruger National Park in South Africa. In a first iteration the model is populated with 4,000 tree agents representing the Marula tree species. The trees feature a complex life-cycle which changes depending on the four seasons as well as the tree's sex and will eventually lead to fruiting and thus the creation of new trees (Helm *et al.*, 2009, 2011; Helm & Witkowski, 2012).

In the first version of the model a simplified seed dispersal algorithm will be used, which simulates the interaction of elephants and other herbivores with the trees and its fruits. The animals eat the fruits and release them later somewhere alongside their trails. By that around 2 % of all seeds grow into seedlings and an even smaller fraction will become adult trees. Also animals and especially elephants do - sometimes fatal - damage to the trees, thus decreasing the overall biomass in the area.

This behavior will be added later by means of an elephant model simulating the effects of water point closure in the Kruger National Park (Hilbers *et al.*, 2014; Scheiter & Higgins, 2012), which is currently under development with MARS LIFE at the Department of Agricultural and Biological Engineering at the University of Florida in Gainesville.

Synthetic benchmarks are good to explore lower and upper limits as well as extreme setups, but how LIFE performs in a real scenario is much more significant to the majority of end-users. So once the elephant model is merged with the Marula model, complexity will have reached a serious threshold, which makes it reasonable to properly parameterize, explore and scale the model up to the whole Kruger National Park. That model will be tested both in single instance runs as well as in multi-node distributed runs, where the scaling factor of adding hardware can be measured.

4 First Results

Since the models described in section 3 are not yet finished, the author conducted some baseline tests with a very basic version of the model. The model includes a simple tree implementation containing equations for tree growth used in the Abdoulaye model (Pereki *et al.*, 2013). The tree agents are placed on random positions in the Kruger National Park (KNP) and perform a spatial query on a KNP GIS elevation layer to fetch their height above sea-level, which is used in the equations. The resolution of the GIS data is 3 arcseconds (90m). The model features no interaction between the agents.

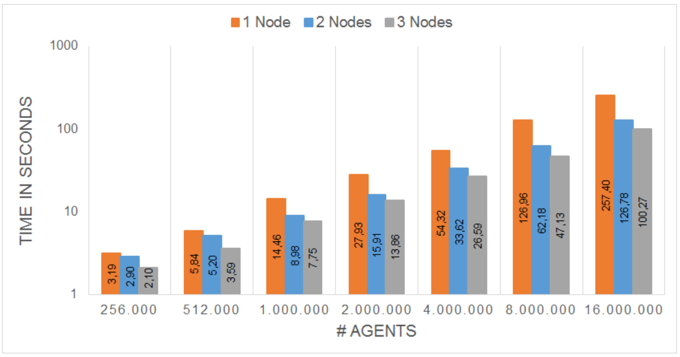

Figure 4.1: Execution duration for 100 ticks

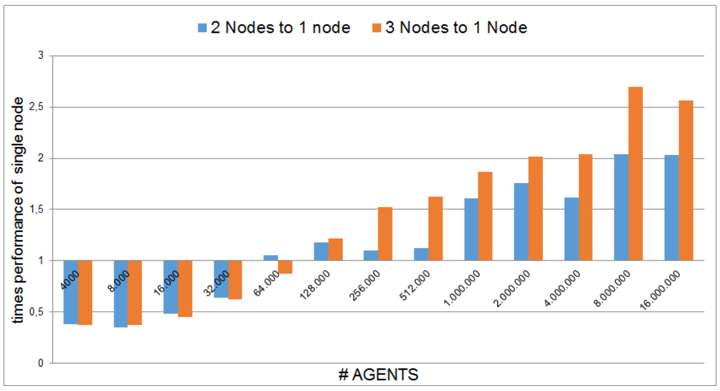

Figure 4.2: Performance gain compared to single node execution

The hardware used for this experiment consisted of one bare-metal node with an Intel Core i7, 3,5 Ghz and 16 GB Ram as well as two virtual nodes with an Xeon 12 Core x 3,5 Ghz and 8 GB Ram each. Tests were conducted from 4,000 to 16,000,000 tree agents with logarithmic distances in between. Each test was run for 100 ticks, which represents 100 weeks in wall clock time, on one, two and three nodes. Figure 4.1 shows the execution duration in the upper range from 256,000 to 16,000,000. The range from 4,000 up to 256,000 is spared here, since the measured durations in that range would not be visible on the scale needed to show the higher values. Figure 4.2 depicts the performance gain when two and three nodes were used compared to the single node run and also highlights the lower range of agent amounts.

The results show that MARS LIFE is capable of running large amount of agents in a very acceptable time and that this performance can even be significantly improved by more than 2 times if two or three nodes are used. Figure 4.2 specifically outlines the turning point at around 64,000 agents, where the performance advantage of distribution overcomes the overhead.

It also became evident that at around 1,000,000 agents the initialization of the SAS turns into a very long (»1h) running task. Performance tracing with dotTrace revealed that the .NET event system seems to be responsible, since the time it takes to add a listener to events in the SAS's underlying implementation increases with each agent. Therefore it should be considered to use full Agent Shadowing up to 1,000,000 agents per agent type and to switch over to on-demand SAS creation if more agents are to be used.

The experiments showed as well that one has to be careful when using the AS caching feature. Activating the cache in simulations which are executed completely locally or run distributed but have very little interaction between agents, leads to worse execution performance depending on the amount of agents (A) multiplied by the amount of changing properties (P). Effectively this results in A x P messages being send without cause or little effect respectively. Thus running the simulation with 100,000 agents on a single node takes around 14 seconds with distribution features deactivated. When distribution features are activated, the same simulation on a single node takes around 84 seconds.

5 Risks & Chances

5.1 Risks

5.1.1 Memory Consumption

Among the greatest risks for the distribution concept of MARS LIFE is the memory consumption. The basic Agent Shadowing demands for every agent to become a proxy on every remote host. So basically every remote host will have to create as many proxies as the layers it hosts contains and are not present as real agents on these hosts. This limits the overall distribution capability to the smallest amount of RAM found on any of the host participating in the simulation.

However this only applies to the basic version of Agent Shadowing. Due to the way Agent Shadowing is implemented, the memory consumption can be significantly reduced by means of a garbage-collection like mechanism. Since MARS LIFE uses multicast messages to communicate between its nodes, each agent can be uniquely addressed by its type and ID. So if memory is becoming an issue, proxies can be spared by just storing agent types and corresponding IDs. This approach could also be combined with some kind of real interest management, which would result in only making those proxies available which are currently in use on a particular node and removing unused stubs respectively.

5.1.2 UDP Multicast

Another risk is the possible loss of messages when using udp multicast as the transport solution for agent communication, because no guarantees for message order or message arrival are made with that protocol. Since MARS is meant to either be executed on local computers or in cloud-like environments with a very controlled and capable network, this shouldn't be too much of an issue. In later version however there should be either a TCP based fallback option or an implementation of one of the newer approaches towards reliable multicast messaging.

5.2 Chances

If the results of the author's master thesis are satisfying, there is a good chance to make a significant contribution to the world of MMAS. Given the demand for cloud-based systems, large-scale models and distributed approaches by Taylor *et al.* (2013), Tolk & Mittal (2014) and Padilla (2014) the MARS system is designed to provide a solution to them by covering all major needs of simulation. The LIFE system is the central component performing the actual simulation as depicted by figure 5.1.

Besides the academic chances, MARS has the ability to become a very supportive and useful tool in ecological modeling, evacuation and urban risk simulations, enterprise architecture analysis and many more fields.

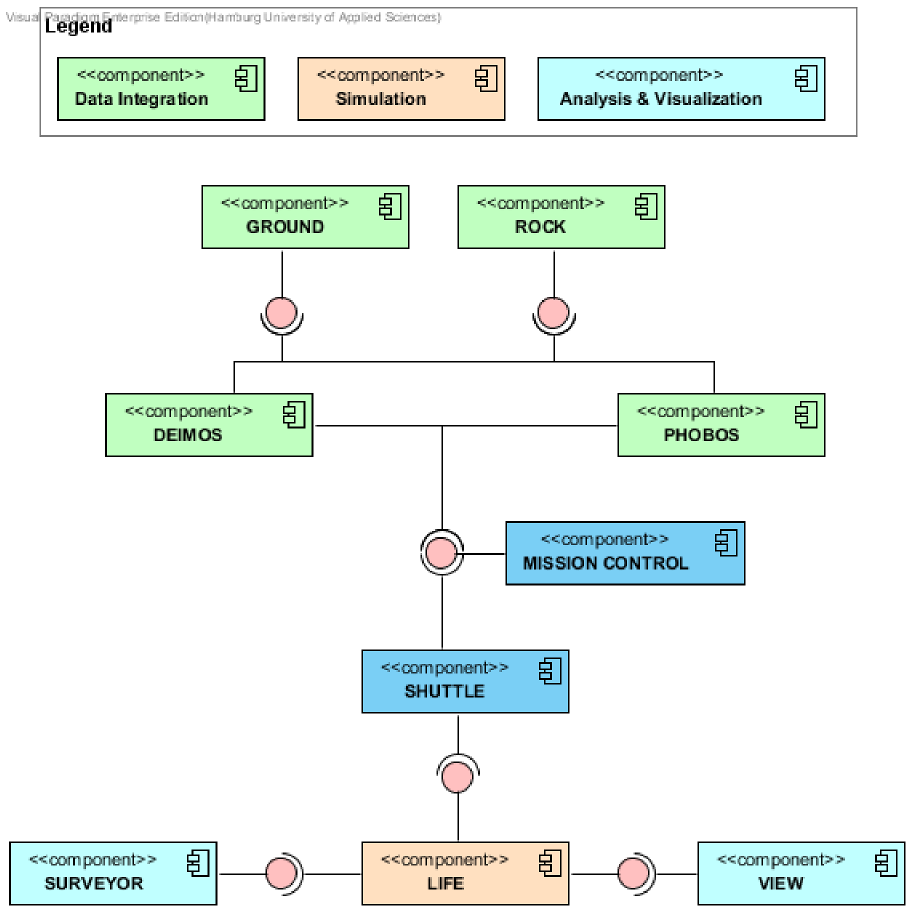

Figure 5.1: MARS System Overview

6 Conclusion & Outlook

The presented concept for the distributed MMAS MARS LIFE is completely implemented and ready to get tested. First results presented in section 4 showed that the system scales in the no-interaction complexity category with more nodes added to the simulation cluster.

The next steps however include additional tests which benchmark the two higher complexity classes of agent-to-environment and agent-to-agent interaction. Also simulation runs with more than 3 nodes, a comparison between bare-metal and virtual nodes as well as monitoring the memory consumption still need to be performed.

It also remains to further develop both the Skukuza and KNP model respectively. Those finished models will include quite a lot of interaction, which is expected to really stress test the Agent Shadowing mechanism. Also a couple of GIS layers will be used in the final models, which imply additional complexity.

The following hypotheses are postulated so far for the author's master thesis:

1. Traffic between agents is minimized by using multicast messaging on a per agent type base.
2. Repeated lookup of remote references is not necessary anymore.
3. Lookup of remote agents only needs the agent's type and ID, but no additional communication.
4. The system is limited by the maximum amount of RAM per node.
5. This limitation can be compensated by introducing on-demand creation of Shadow Agent Stubs.

Bibliography

Cayirci, Erdal. 2013. Modeling and simulation as a cloud service: A survey. *Proceedings of the 2013 Winter Simulation Conference - Simulation: Making Decisions in a Complex World, WSC 2013*, 389–400.

Helm, C. V., Witkowski, E. T F, Kruger, L., Hofmeyr, M., & Owen-Smith, N. 2009. Mortality and utilisation of Sclerocarya birrea subsp. Caffra between 2001 and 2008 in the Kruger National Park, South Africa. *South African Journal of Botany*, **75**(3), 475–484.

Helm, C. V., Scott, S. L., & Witkowski, E. T F. 2011. Reproductive potential and seed fate of Sclerocarya birrea subsp. caffra (marula) in the low altitude savannas of South Africa. *South African Journal of Botany*, **77**(3), 650–664.

Helm, Chantal V., & Witkowski, E. T F. 2012. Characterising wide spatial variation in population size structure of a keystone African savanna tree. *Forest Ecology and Management*, **263**, 175–188.

Hilbers, Jelle P, Langevelde, Frank Van, Prins, Herbert H T, Grant, C C, Peel, Mike, Coughenour, Michael B, Knegt, Henjo J De, Slotow, Rob, Smit, Izak P, Gregory, A, Boer, Willem F De, Group, Resource Ecology, Africa, South, Services, Scientific, Park, Kruger National, Resource, Natural, Collins, Fort, Engineering, Biological, & Science, Computer. 2014. Elephant-mediated cascading effects of water point closure in Kruger National Park , South Africa. 1–32.

Hüning, Christian, Wilmans, Jason, Feyerabend, Nils, & Thiel-Clemen, Thomas. 2014. MARS - A next-gen multi-agent simulation framework. *Simulation in Umwelt- und Geowissenschaften, Workshop Osnabrück 2014*, 1–14.

Padilla, Jose. 2014. Cloud-Based Simulators: Making Simulations Accessible To Non-Experts and Experts Alike. *Proceedings - Winter Simulation Conference 2014*, 3630–3639.

Pereki, Hodabalo, Wala, Kperkouma, & Thiel-Clemen, T. 2013. Woody species diversity and important value indices in dense dry forests in Abdoulaye Wildlife Reserve (Togo, West Africa). *International Journal*, **5**(June), 358–366.

Scheiter, Simon, & Higgins, Steven I. 2012. How many elephants can you fit into a conservation area. *Conservation Letters*, **5**(3), 176–185.

Scholes, R. J., Gureja, N., Giannecchinni, M., Dovie, D., Wilson, B., Davidson, N., Piggott, K., McLoughlin, C., Van der Velde, K., Freeman, a., Bradley, S., Smart, R., & Ndala, S. 2001. *The environment and vegetation of the flux measurement site near Skukuza, Kruger National Park.*

Taylor, Simon J E, Fujimoto, Richard, Page, Ernest H., Fishwick, Paul a., Uhrmacher, Adelinde M., & Wainer, Gabriel. 2012. Panel on grand challenges for modeling and simulation. *Proceedings - Winter Simulation Conference.*

Taylor, Simon JE, Khan, Azam, Morse, Katherine, Tolk, Andreas, Yilmaz, Levent, & Zander, Justyna. 2013. Grand Challenges on the Theory of Modeling and Simulation. *Proceedings of the 2013 ACM SIGSIM conference on Principles of advanced discrete simulation.*

Tolk, Andreas, & Mittal, Saurabh. 2014. A necessary paradigm change to enable composable cloud-based M&S services. *Proceedings of the 2014 Winter Simulation Conference*, 356–366.

Vigueras, Guillermo, Orduña, Juan M., Lozano, Miguel, & Jégou, Yvon. 2013. A scalable multi-agent system architecture for interactive applications. *Science of Computer Programming*, **78**(6), 715–724.