MADA - An adaptive distributed architecture for RoboCup Rescue Simulation

Pooya Deldar Gohardani, Vahid Amiri, Sajjad Rostami
Mechatronics Research Laboratory, Islamic Azad University, Qazvin Branch, Qazvin, Iran
{pooya.gohardani, vahid.amiry, sajadrostami19}@gmail.com

Abstract

RoboCup Rescue Simulation intends to promote research and development in a socially significant domain such as disaster rescue by creating a standard simulator and forum for researchers and practitioners. The simulator has changed a lot to get closer to the reality but it cannot support large scale domains well. In this paper we are going to introduce a new architecture for the simulator to address known issues such as scalability and failure management much better than before.

Introduction

RoboCup Rescue Simulation System (RCRSS) has changed a lot since its first release in 2000 [1, 2]. One of the most obvious changes is related to the number of possible entities in a scenario, such that, in the current version there can be about 20,000 buildings and roads, 150 rescue agents and 1000 civilians. These number of entities are much more than the number of entities of prior versions.

Considering the scenarios close to the reality, the number of possible entities are still far less than what it should be. Actually there are two main problems in the way of RCRSS to be close to reality. One of the problems is the slowness of the system by growing the number of possible entities; where in a more real scenario there can be about a million or more entities. This limitation is because of the resource consumption of the system, that is, when the number of entities in the current system is more than the mentioned numbers, CPU and memory consumption of the system gets very high so that a single step of the scenario may take about a minute or so to complete. Although the architecture of the system is modular, it is considered to be a monolithic architecture. In this monolithic architecture there is no feasible approach for the system to support scalability. For clarification consider the collapse simulator which is one of the main simulators of the RCRSS; The system has only one instance of the simulator for the whole simulation and in case of a high load; e.g. scenarios close to reality, it has to process everything singly.

The second problem is a famous problem for a monolithic architecture which is called single point of failure. As the system uses only one instance for each component, the failure of any components; e.g. simulators, breaks down the whole simulation and it needs to be started from the beginning.

To address these problems, MRL team had proposed a distributed architecture using Hadoop Yarn framework in 2015 [3] but this architecture could only support scalability in a static manner where the system could calculate the needed resources before starting the simulation and no more dynamic resource allocation could happen during the simulation. On the other hand, if we wanted to design such dynamic resource allocation using Yarn, it could be something time consuming and difficult because Yarn is not designed to support such functionality.

This year, our concentration is on designing a new architecture separate from a specific framework or any utility. We named this new architecture as MADA which stands for MRL Adaptive Distributed Architecture.

Distributed multi-agent system

A multi-agent system is a computerized system composed of multiple interacting intelligent agents within an environment. Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve. Intelligence may include some methodic, functional, procedural or algorithmic search, find and processing approach. Topics where multi-agent systems research may deliver an appropriate approach

include online trading, disaster response, and modeling social structures. Many new multi-agent applications are currently being developed and launched such as Jade [4] and Pandora [5]. In [6], a comparative review of 24 multi-agent platforms, frameworks, and simulators is reported using 28 universal criteria.
Multi agent System has been brought in research and development of distributed systems. Distributed multi-agent system is another form of multi-agent systems aims to provide scalability, fault management and other features of a distributed architecture into the multi-agent environment.
Although a multi-agent system inherits some features of distribution but most of them are composed of some monolithic components interacting separately using message passing. So to bring the most benefits of a distributed environment they should support some form of scalability, fault management and so on which we cannot see in RCRSS yet.
In the next section we are going to introduce MADA, our proposed architecture for RCRSS.

Proposed Architecture

As we discussed in previous sections, scalability and fault management are our goals for the new architecture. Rescue Simulation System consist of a kernel, different simulators and lots of agents. For this system, there are two obstacles to our goals using the current architecture of the RCRSS, the first one is the tight coupling between components of the system specially between simulators and kernel and the second one is about states of the components, their variables.

The tight coupling between simulators and the kernel comes to be a problem when one of them crashes so the system becomes useless at that situation and the second problem keeps the components away from replication.

Thus to overcome the mentioned obstacles, we proposed the use of a micro service based architecture. In a micro service based architecture loosely coupling and stateless components play important roles. Considering these hypothesis, whenever the load of the system gets high, it can simply control the situation by adding new resources and put the load on different components using its load balancer.

In the current architecture of the system, the kernel and other components of the system can work on separated workstations and this happens to agents as well. Figure.1 shows the current architecture.

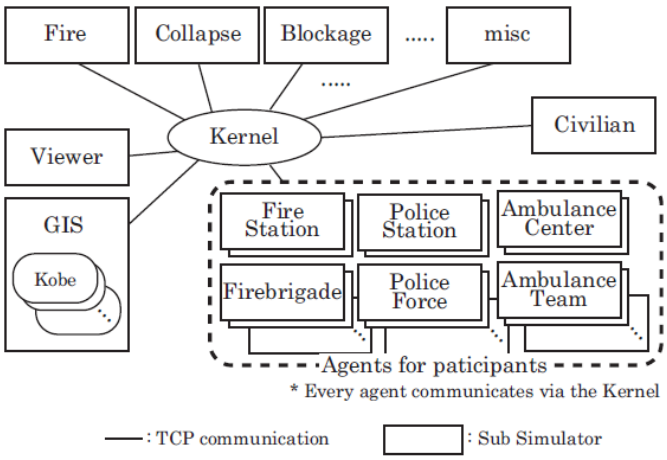

Figure.1 current architecture of RCRSS

In this architecture whenever we use a map with lots of entities and agents, the load of the system gets aggressively high and most of the time we need a strong computer to handle it. As the system is monolithic, running a more real scenario – a scenario with millions of entities – drops the speed of the system sharply, that is, there is no way for the

system to balance the load using multiple instances of the same components, this causes us to run the system with minimum amount of entities which is far away from real scenarios. On the other hand, whenever a part of the system crashes or stops accidentally, the whole system gets into trouble and needs a restart.

The new architecture is based on micro services. In this new architecture by redesigning the main components to be loosely coupled, we can simply replicate the overloaded components in many different machines. Furthermore, by separating the states of the components form their functionalities, the system can be scaled easily and it will provide higher degrees of system resilience.

In the proposed architecture all the states and information are kept in a distributed in-memory database and there is no component with states in it. This is why we call them stateless components. Figure.2 shows the proposed architecture.

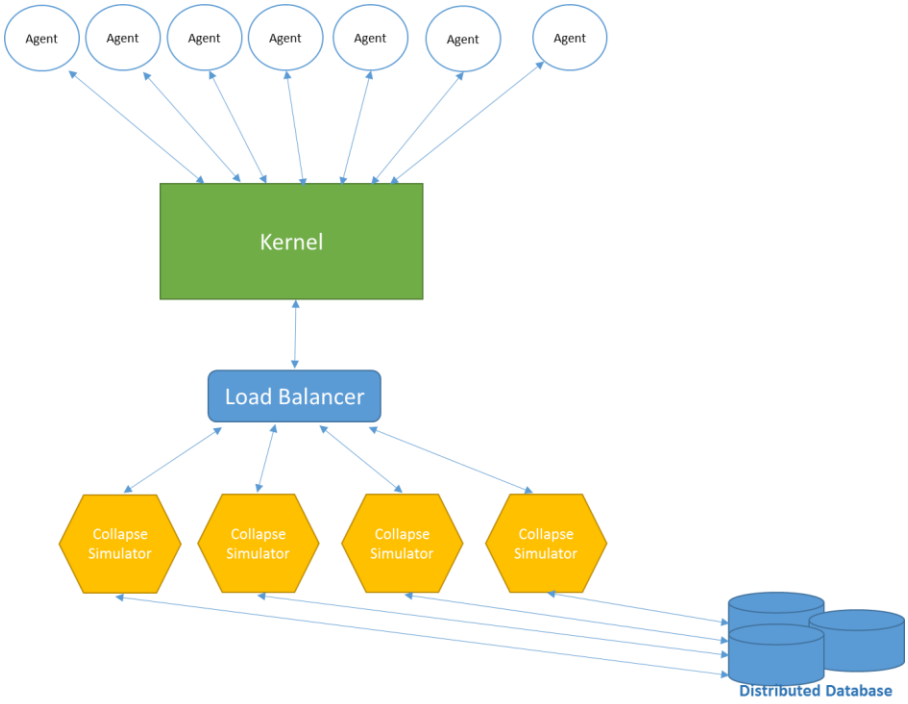

Figure.2 MADA architecture

As it is shown in the figure.2 when the load on any components gets high; e.g. Collapse Simulator, the system replicates those components and uses a load balancer in between to route new loads on proper component replicas.

According to the use of that distributed database to keep the inner states of the components, failure of any components cannot hurt the system, because the system can easily make another replica and uses proper states in the database to start it from the latest stable states.

Using the distributed databases provide the ability to the system to get started from previous time steps so this feature can be handy feature for debugging and development processes.

Conclusion and Future work

Solving issues such as scalability and failure management for a sophisticated simulator like RCRSS encouraged us to think out of the box and think about an improved architecture. We called the new architecture MADA and its aim is to handle scalability and failure management issues using micro services together with a distributed database to keep inner states of separate components (simulators).

This year we focused on collapse simulator as an initial point for our investigation and off course we will extend our workarounds to other parts of the simulator.

Reference

[1] T. Takahashi, I. Takeuchi, F. Matsuno and S. Tadokoro, (2000) "Rescue simulation project and comprehensive disaster simulator architecture," Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113), Takamatsu, pp. 1894-1899 vol.3.

[2] Visser A., Ito N., Kleiner A. (2015) RoboCup Rescue Simulation Innovation Strategy. In: Bianchi R., Akin H., Ramamoorthy S., Sugiura K. (eds) RoboCup 2014: Robot World Cup XVIII. RoboCup 2014. Lecture Notes in Computer Science, vol 8992. Springer, Cham

[3] Deldar Gohardani P., Amiri V., Rostami S.,(2015) RoboCup Rescue Simulation 2015, Introducing a New Distributed Multi Agent System Architecture in Rescue Simulation, MRL team description

[4] Bellifemine F., Poggi A., Rimassa G. (1999). JADE–A FIPA-compliant agent framework. Telecom Italia Internal Technical Report. Part of This Report Has Been Also Published in Proceedings of PAAM'99.

[5] Pandora FMS . Retrieved from pandorafms.com: https://pandorafms.com/

[6] Kravari K., Bassiliades N., (2015). A Survey of Agent Platforms. Journal of Artificial Societies and Social Simulation. 18.10.18564/jasss.2661.