Measurement of RTI Performance for Tuning Parameters to Improve Federation Performance in Real-time War Game Simulation

Su-Youn Hong*, Jae-Hyun Kim**, Tag Gon Kim***
Department of EECS
KAIST
373-1 Kusong-dong, Yusong-gu
Taejon, Korea 305-701

Tel: +82-42-869-3454
Fax: +82-42-869-8054
syhong@smslab.kaist.ac.kr*; jhkim@smslab.kaist.ac.kr**; tkim@ee.kaist.ac.kr***

Keywords: HLA, RTI, performance, real-time war game simulation

Abstract
Real-time war game simulation often employs High-Level Architecture (HLA) for interoperability. Performance of a HLA federation is dependent on efficiency of Run-Time Infrastructure (RTI) which provides HLA services to federates joining the federation. This paper investigates performance of RTI to recommend guidelines which can improve federation performance in real-time war game simulation. Measurement of RTI NG1.3v6 performance under the combination of a variety of options and different workloads is present. A set of optimistic options for improvement of federation performance is proposed.

1. INTRODUCTION

Recently interoperable simulators have much attention in large scale simulation area, especially in real-time war game simulation area. An interoperable federation consists of a collection of interoperable federates which communicates with each other. The main function of interoperation is synchronization of simulation time and information exchange between federates. Thus, execution time of the federation not only depends on execution times of federates but communication times between federates. Real-time simulation of such federation requires that both the execution times and the communication times should satisfy a specified time constraint for each simulation cycle.

High-Level Architecture(HLA) and its implementation of Runtime Infrastructure(RTI) was proposed as a standard for simulators interoperation[1]. But before applying RTI for real-time war game simulation area, it is needed to check whether RTI satisfies the performance requirements of such a simulation. Performance tests have been a frequent topic of HLA/RTI area. Generally, the performance test of RTIs compare differences between the RTIs with respect to simulation design and analyze the testing strategies used[3]. In addition to the general performance test, performance of a real-time simulation system with RTI NG 1.3 from DMSO was presented[4].

What this paper proposes is different from above researches in searching optimistic options and parameters for RTI. The performance of federation can be improved by these RTI characteristics. This paper shows a variety of experimental results for RTI NG 1.3v.6. An experimental frame federate has been implemented that generates different workloads for information exchanges through RTI. The experimental results suggest optimistic options and parameters for a use of RTI NG 1.3v.6.

The remainder of the paper is organized as follows. A brief introduction of HLA/RTI and the structure of federation are presented in section 2. Section 3 discusses factors for performance of federation and RTI options and parameters affiliated those factors. Section 4 gives the concepts and structure of a developed experimental frame federate. In section 5, we present the compared results with different RTI options and parameters, and section 6 contains conclusions of this paper.

2. HLA/RTI BASED INTEROPERABLE SIMULATION

The HLA mandate establishes common high-level simulation architecture to facilitate the interoperability of all types of models and simulations among themselves. The HLA is designed to promote standardization in the modeling and simulation (M&S) community and to facilitate the reuse of M&S components.

The RTI software implements the interface specification and represents one of the most tangible products of HLA. It provides services in a manner that is comparable to the way a distributed operating system provides services to applications [1].

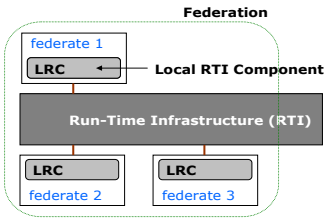


Figure 1. Typical form of federation

Within the HLA, federations are comprised of federates that exchange information in the form of objects and interactions through RTI, which communicates in the end via TCP/IP[2].

3. FACTORS FOR FEDERATION PERFORMANCE AND MEASURED RTI OPTIONS AND PARAMETERS

Performance of federation in overall simulation time may be improved one of the following three ways. The first method is to reduce communication time for information exchange between federates. The second is to reduce local simulation times within federates. The third is to reduce an amount of information exchange between federates. The first can be achieved by tuning RTI parameters in optimal manner. The second can be achieved by use of a multi-resolution modeling approach. The third can be achieved by exploiting Data Distribution Management service provided in RTI. This paper deals with the first approach to improve federation performance. To do so a variety of measurements of RTI performance is made with different options and parameters setting.

3.1 Factors for Federation Performance

As explained in sections 2, RTI provides and manages the communication interface for federates. All communication messages of federation are performed through RTI. Since RTI is like a gateway, it produces some latency and throughput degradation because of these messages and internal mechanism. Therefore the performance of federation with RTI can be improved by using optimal options and parameter settings in a RTI usage.

There is another considerable element for federation performance except for the performance of RTI. That is the cycle of calling ticks; a service tick() is invoked by a federate to yield processor time to the LRC associated with the federate. During a tick() invocation, the LRC will process incoming traffic, deliver callbacks to the federate, and perform various functions for internal RTI maintenance essential to the operation of the a federation. It is essential that all federates invoke tick() frequently so that internal RTI communications may be serviced in a timely fashion[1]. RTI would be starved if the tick is not called appropriately; if a federate too often calls the tick, it may consume more time than required.

3.2 Measured RTI Options and Paramet -ers

For systematical measurement we divided parameters which related to federation performance into three categories: RTI options, a tick usage and a number objects and their updates. Measurements are performed by changing parameters in one category while those in two other categories are fixed.

3.2.1 RTI Options

The first category is parameters in the RID file, which allows us to select different options for simulation management within RTI. Among these options, there are two options which can improve the performance of federation by the contents in section 3.1. Those are Process model's strategy and Mom service reporting.

There are currently two process model strategies that are supported by the RTI: a polling process model and an asynchronous I/O process model. The polling process model uses a single thread of execution shared between RTI and a federate. Only when the federate calls a tick is the RTI able to perform work. This strategy can starve RTI if the tick is not called appropriately. The asynchronous I/O process model uses an internal thread within RTI to avoid starvation. This thread will periodically wake up and determine if it can perform any internal RTI work. In the asynchronous I/O strategy the federate only needs to invoke ticks when it is prepared to handle callbacks[1].

Although MOM services are very useful for federation monitoring and control, there is an overhead associated with MOM. This parameter, FederationSection.MOM.MomServiceAvailable, allows a federation to turn off MOM services if they are not being used within federation[1].

3.2.2 Tick Usage

The second category is options for use of ticks within RTI which can be used by the experimental frame federate to perform callback functions, to update objects and to send interactions.

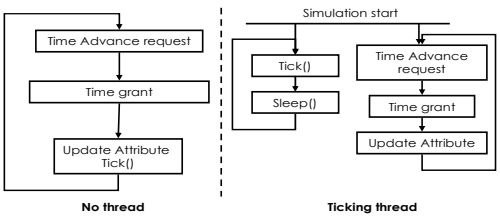


Figure 2. Tick usage

A federate must call a tick for a RTI callback,

for which this section suggests two types of calling ticks. The first type is to call ticks in a federate program while the federate being executed. The second type is to call ticks using a thread which is different from a simulation thread.

3.2.3 Number of Objects and Updates

The third category is a number of objects and their updates. This category is connected directly to wire traffic and RTI load. Fixing the other parameters in the previous two categories, an experiment is performed to find a critical boundary about the number of registered objects. As a case of a number of updates, the execution times are measured as a number of updates vary.

3.3 Reference for comparison

To measure RTI performance with different parameters a simulation ratio is used as a reference for comparison. The ratio is defined by simulation time divided by wall clock time, where simulation time is federates' logical time and wall clock time is physical time to be spend in simulation execution.

$$ratio = \frac{\partial(simulation\_time * 60.0)}{\partial(wallclock\_time)}$$

What the constant 60.0 means is that one simulation time step is equal to one minutes.

4. EXPERIMENTAL FRAME FEDER -ATE AND MEASUREMENT ENVIRO -NMENT

4.1 Experimental Frame Federate

The main purpose of measurement is first to find RTI parameters which play an important role in information exchange through RTI and then to tune them optimally. Such tuning leads to improve federation performance for the same amount of information exchange. To do so this paper first develops an experimental frame federate (Fig 3) which generates different workloads for information exchanges through RTI.

Then, measurements are made at different workloads using the developed federate to analyze a relationship between RTI parameters and an amount of information exchanges between federates. The environment of developing experimental frame federate is as follows.

- Language : C++
- Environment
  - Window XP: visual studio .NET 2003
  - Linux Redhat 9.0 : gcc- 3.0.2

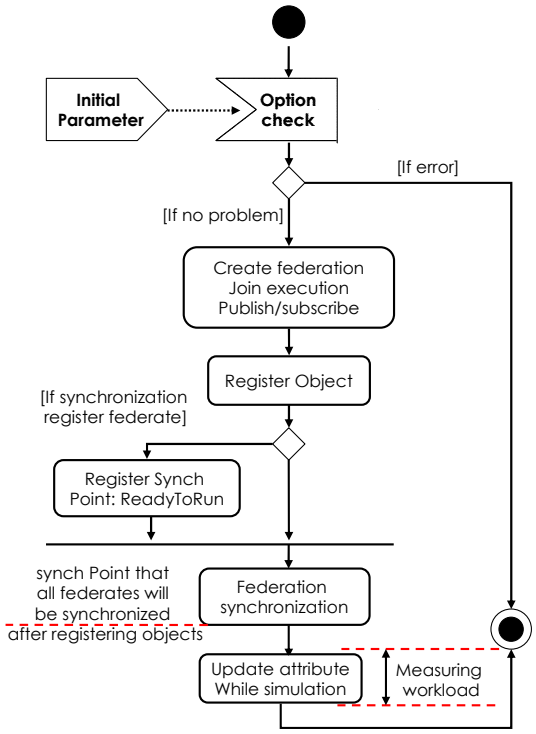


Figure 3. Experimental frame federate for workload measurement

For real-time simulation, a time-stepped advance of time is assumed[4]. In case of an experimental frame federate, each time step consists of three phases as shown in Figure 4. Time interval and lookahead are fixed at 1 min and 59.0/60.0, respectively.

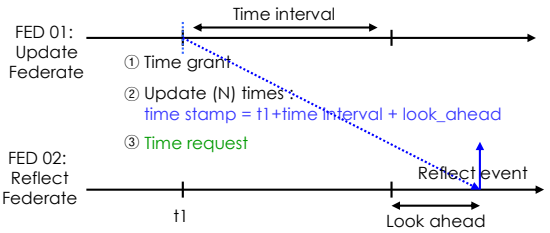


Figure 4. Progress of simulation time advance

FED01 and FED02 in Figure 4 are simplified in which FED01 and FED02 only consider updates and reflect, respectively. However, a real experimental frame federate can perform both update and reflect. Updating and reflecting are repeated per each time step and measurement of an execution time(sec) is obtained 1,500 min of simulation time. If the ratio of simulation time to real time exceeds 1.0 this simulation satisfies real-time execution. Our simulation ratio satisfies this condition.

The experimental frame federate named "MeasureLoad", can be controlled by parameters. This program is started with the following example using parameters in Table 1.

Table 1. MeasureLoad parameters

MeasureLoad <FederateName> <-m/s> [-synchPointRegister] …	
<Federate Name>	Federate name
<-m/s>	-m: multi-federates/federation -s: single federates/federation
[-synchPoint Register]	This federate will register synchronization point.
[-tickOption <0>/<min,max>]	tick() arguments setting
[-TAR/-NER]	TAR: TimeAdvanceRequestAvailable NER:nextEventRequestAvailable
[-Attr <1~10>]	Attribute's number of registered object
[-Update <1~>]	Update number/simulation time
[-Object <0~...>]	Total number of registered objects
[-AttrSize <1~>]	Attribute's size of registered object
[-A/-P]	Cycle of tick calls (3.2) -A: asynchronous -P: polling
[-Real]	- real time simulation

4.2 Measurement Environment

The measurements environment is fixed on the basis of log files that have specification about an amount of information used in a practical war game simulation. The federation using these log files consists of four federates and one federate registers objects and updates these objects. The number of objects is about 12,000 and the number of updates during simulation is shown in Figure 5.

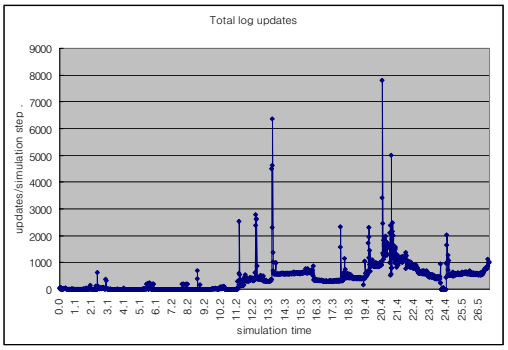


Figure 5. Total updates number of log file

The maximum number of updates was about 8,000. Therefore if RTI persists in real-time with 12,000 objects and 8,000 updates real-time war game simulation can be executed.

Because RTI software implements the interface specification of HLA, many types of RTIs are in place. All of our experiments were done with the RTI NG 1.3v.6 of DMSO.

All the experiments were done on eight Pentium IV 2.8GHz 512MB RAM computers running RedHat Linux 9.0 and communicating via a 100Mb Ethernet network. Each computer uses the standard TCP/IP protocol. We discuss measurement results which were executed with variable RTI options, the number of objects and updates in the following section.

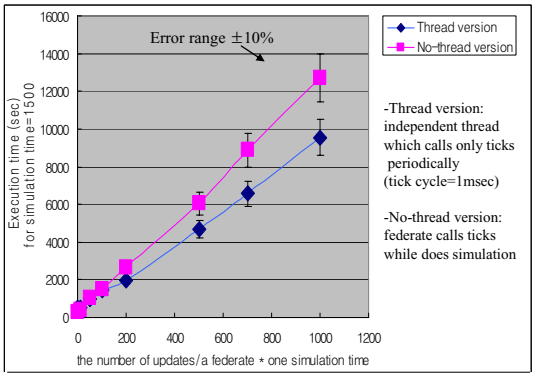
5. MEASUREMENT RESULT AND DISCUSSION

This section discusses a variety of measurements using a practical workload of war game simulation. Measurements of RTI performance with different options and parameter settings give us important guidelines to improve federation performance.

5.1 Tick usage

In section 3.2, we discuss about using the ticking thread which calls ticks periodically. The section compares execution times of two federations: one with federates using a tick thread and the other with federates without using the thread.

The experiments consist of two options. The first option is varying the number of updates with fixed the number of registered objects; the second is varying the number of registered objects with the number of updates fixed. The result of first option is shown in Figure 6.

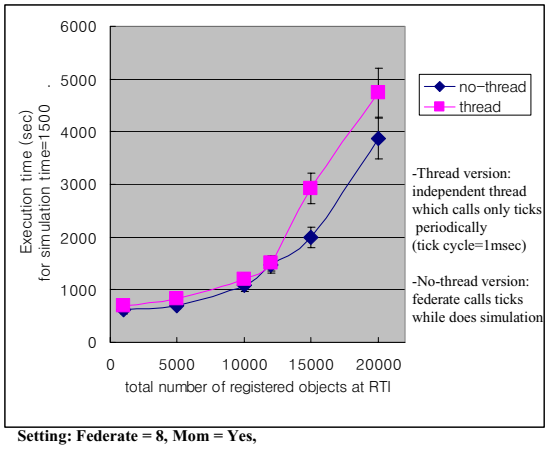


Setting: Federate = 8, Mom = Yes,
Total number of registered Object at RTI=12000

Figure 6. Execution time comparison: thread vs. non-thread version with fixed objects number

As shown in Figure 6, the more the number of updates increases, the less execution time is taken with the ticking thread.

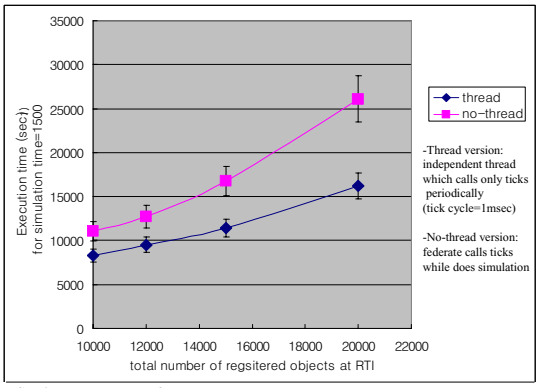
The result of second option is shown in Figure 7. In case that the number of registered objects is small, the performance of a federation using the tick thread and that without using the tick thread is not considerable. However, when the number of registered objects is large, the federation with ticking thread shows better performance.



Setting: Federate = 8, Mom = Yes,
Total number of updates/(one federate*one simulation time)=100

Figure 7. Execution time comparison: thread vs. non thread version with fixed updates number

When the fixed number of updates is increased, it is certain. The result is shown in Figure 8.



Setting: Federate = 8, Mom = Yes,
Total number of updates/(one federate*one simulation time)=1000

Figure 8. Execution time comparison: thread vs. non thread version with fixed updates number

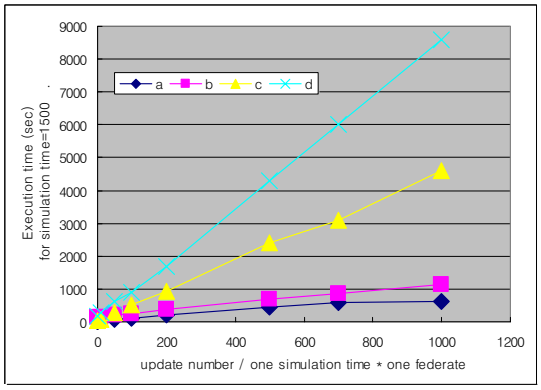
5.2 RTI Options

The performance of federation can be improved by selection of different options that control the operation of RTI.

5.2.1 Mom Service Reporting

Mom service reporting is an optional service for reporting state of each federate to RTI. This option is not required for simulation itself, which has a large amount of wire traffic. This section compares the execution time between federation using Mom service reporting and that without using it for measuring the load of Mom service.

A typical result is shown in Figure 9. The execution time using Mom service is much longer than that not using Mom service. This result shows that the load of Mom service is heavy to a considerable extent.



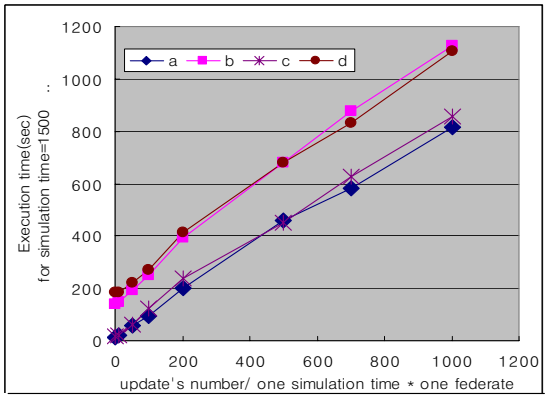
a: Polling Model, Mom Service off, No-thread
b: Polling Model, Mom Service off, Ticking thread
c: Polling Model, Mom Service on, No-thread
d: Polling Model, Mom Service on, Ticking thread
Setting : Federate = 8, Total registered object=12000, Tick arg: (0, 0.1)

Figure 9. Execution time vs. MOM service reporting on/off

Developers usually use Mom service for checking the state of federates and a federation. They should use this service for debugging. However, if the required real-time condition can not be satisfied, this service option may be turned off to meet the condition.

5.2.2 Process Model

RTI supports two types of process models, asynchronous IO and Polling as explained in section 3.2. The process model controls the mechanism used by RTI to obtain processing cycles and support callbacks to the federate during the tick call. This section compares the execution time among federates with different process model, tick usage and Mom service reporting option.



a: Polling Model, Mom Service Off, No-thread
b: Polling Model, Mom Service Off, Ticking thread
c: AsynchronousIO, Mom Service On, No-thread
d: AsynchronousIO, Mom Service On, Ticking thread
Setting: Federate = 8, total registered objects=12000, Tick arg: (0, 0.1)

Figure. 10 Execution time vs. MOM service reporting on/off & process model

The result is shown in Figure 10. The measured execution time using different process

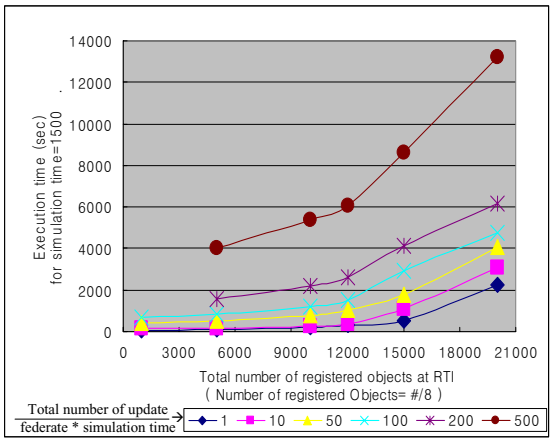
models does not have remarkable difference. The experimental result shows that the process model option cannot affect the execution time against tick usage and Mom service reporting option.

5.3 Total number of registered objects and updates

In section 5.1 and 5.2, we discuss about RTI options and tick usage which do not change during simulation. The performance of war game simulation using RTI is affected by not only these constant parameters, but also various parameters. Such parameters include a total number of registered objects and their updates that are connected directly with wire traffic. This section discusses about that.

5.3.1 Total number of registered objects

The execution time is measured against the number of registered object with the number of updates fixed. The result is shown in Figure 11. The execution time increases rapidly when the number of registered objects oversteps a critical boundary.



Setting: Federate = 8, Ticking = -P, Tick arg: (0, 0.1), Mom = Yes

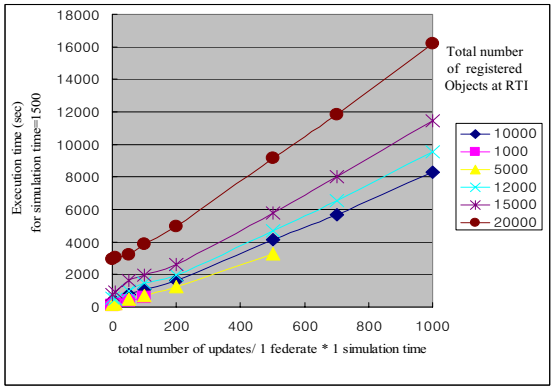
Figure 11. Execution time vs. total number of registered objects at RTI

The more the number of updates is, the less the critical boundary is, but slightly. The critical boundary of registered objects is about 12,000 from experiments.

5.3.2. Total number of updates

The execution time is measured against the number of updates, with the number of registered objects fixed. The result is shown in Figure 12.

The more the number of registered objects increases, the more the execution time for same number of updates increases. But the execution time is taken in proportion to the number of updates. Therefore federation carries out the real-time war game simulation within a limited operational range; the number of updates is 10,000. This limit is satisfied with the range of a log file in section 4.2.



Setting: Federate = 8, Ticking = -thread, Tick cycle= 1msec, Mom = Yes

Figure 12. Execution time vs. total number of updates

6. CONCLUSION

Performance of RTI is crucial to application of HLA/RTI in real-time war game simulation. This paper measured performance of RTI with different RTI options and a variety of workload settings. Experiments were done with a practical log file of war game simulation by using a developed experimental frame federate. All measurements were made under a network of 8 PCs. The measurement results suggested some guidelines to improve performance of federation for real-time war game simulation using RTI. The option of RTI process models did not considerably affect the execution time of federation. However, when federation employed a polling process model the use of the ticking thread option showed better performance. The maximum number of registered objects which showed a reasonable execution in RTI-NG 1.3v6., was about 12,000.

REFERENCES

[1] Defense Modeling and Simulation Office, *High Level Architecture Interface Specification, Version 1.3*: Washington D.C. 1998

[2] Ms. Pamela Knight, Mr. Ron Liedel, Ms. Melanie Klinner, "WBT RTI Independent Benchmark Tests: Design, Implementation, and Updated Results", in *Procs. of 2002 Spring Simulation Interoperability Workshop,* Orlando, Florida, USA, March 10-15, 2002. 02-SIW-081

[3] Brad Fitzgibbons , Thom McLean, Richard Fujimoto, "RTI Benchmark Studies", in *Procs. of 2002 Spring Simulation Interoperability Workshop,* Orlando, Florida, USA, March 10-15, 2002. 02-SIW-105

[4] Dr. Herbert Tietje, "Benchmarking of RTIs for Real-Time Applications", in *Procs. of 2003 Euro Simulation Interoperability Workshop,* Stockholm, Sweden, June 16-19, 2003. 03E-SIW-025