Data Farming in Support of HLA Performance Assessment

Anthony Cramp, Tom van den Berg, Wim Huiskamp
TNO
Oude Waalsdorperweg 63
2597 AK The Hague
The Netherlands
{anthony.cramp, tom.vandenberg, wim.huiskamp}@tno.nl

ABSTRACT: *Performance assessment is a key factor in designing distributed simulation environments that are fit-for-purpose and cost-effective. Simulations used for training applications should provide the required level of responsiveness and interactivity. Simulations used for analysis or decision support should execute as fast as possible to enable quick results on large numbers of scenarios and variables. There are many parameters that have an impact on the performance of a typical High Level Architecture (HLA) federation. We describe a structured process for using data farming (experiment design, simulation, cloud computing, and data mining) in order to identify key parameters affecting the performance of HLA federations. A parameterized and instrumented federation is designed with parameters covering areas suspected of impacting HLA performance and instrumentation measuring key values of HLA performance. One key parameter set for HLA performance is the computing platform that hosts the federation. Varying such a platform in an efficient and cost effective manner is possible using cloud computing via Infrastructure-as-a-Service providers. Finally, the performance measures captured during federation execution are collated and analysed using data mining techniques to identify key parameters and their effect on the performance of the federation. By including the Runtime Infrastructure (RTI) as a parameter in the federation design, it may also be possible to identify where the HLA itself (instead of particular RTIs or computing platforms) is impacting on simulation performance. Initial results from this process are presented and future work, including the creation of a HLA performance model from identified parameters, is discussed.*

1 Introduction

Performance of simulations can be vitally important. Human-in-the-loop simulations for training require an appropriate amount of responsiveness to ensure seamless interactivity. Any extra computational time taken by the simulation infrastructure is time not available to the simulation models potentially restricting the fidelity of the models in order to achieve simulation timing requirements. The latency introduced by the infrastructure also affects the responsiveness of the system and possibly the stability of the models, for example, due to operator induced oscillations.

Similarly, constructive simulations used for analysis or decision support need to be performant. A saving of as little as a few seconds per simulation iteration can lead to vast savings over the life of an analysis covering multiple scenario configurations and multiple Monte Carlo iterations per configuration. For example, the authors have been involved in a number of studies involving analysis simulations. One of the analysis simulations consists of 100 different simulation configurations with each configuration being run over 400 Monte Carlo iterations. Each configuration had a runtime of, on average, 130 simulation seconds and ran at a wallclock rate of 0.1 or 1300 wallclock seconds (21 minutes, 40 seconds). Executing these simulations sequentially would take approximately 600 days. Increasing the wallclock rate to even 0.11 would reduce the execution time to less than 550 days allowing answers to be produced earlier or providing more time for extra scenario configurations to be executed[1,2].

The performance of this simulation was measured by running the simulation. However, this leads to a chicken and egg problem whereby the amount of computing resources required for the simulation is dependent on the performance of the simulation but the performance of the simulation is only measured by running the simulation on the computing resources.

Being able to predict performance is one way to break this circular dependency. Accurate prediction of the performance of a simulation provides the simulation designers scope to efficiently allocate simulation participants, simulation support staff and computing resources prior to the final completion of the simulation. Of course, some measure of agility will be required to adapt to circumstances falling outside the performance

[1] Wallclock runtime in days = #configurations x #iterations x simtime / wallclock rate / seconds per day

[2] The numbers presented for the analysis simulation are representative but not exact. Plus, the actual simulation execution didn't happen sequentially; rather multiple machines were employed to execute different scenario configurations in parallel. Also, startup and shutdown times are not taken into account in the above calculations, which add more time. The full suite of iterations in the real simulation on the real computing hardware took about one month of continuous execution.

predication. However, some predication, assuming it is correct, is better than none.

Despite the body of research that exists on High Level Architecture (HLA) performance, prediction is still somewhat hit or miss due, in part, to the large parameter space that can affect performance. The parameter space for HLA performance spans the simulation stack from the computing infrastructure (computers and networks, including encryption) through the base platform environment (operating systems, programming language) into the HLA Runtime Infrastructure (RTI) being used and the ways in which the federates interact with the RTI, and finally into the domain being simulated and the domain coupling that arises across the federates interacting during simulation execution.

Exploring this huge parameter space requires techniques from experiment design (to identify the most important parameters and the important value ranges of those parameters), high performance computing (to be able to execute all the simulation configurations within a reasonable amount of time) and data mining and analysis (to be able to search through the data and reveal interesting results and/or correlations).

Data farming is the term given to this holistic approach to simulation derived data exploration. Horne and Meyer [1] presents data farming as an iterative approach covering question/topic research and definition, model development and gaming, parameter space exploration, and data exploration and analysis. They note *"in Data Farming, there is not necessarily a predefined hypothesis that is being confirmed. The data is being 'explored'. Trends and relationships may become exposed, but outliers, cusps and saddle points may also be found in the n-dimensional output space"*.

The North Atlantic Treaty Organisation (NATO) is one organization exploring the use of Data Farming to be used for decision support. The NATO Science and Technology Organisation (STO) Modelling and Simulation Group's (NMSG's) Task Group MSG-088 delivered its final technical report [2] in which are defined six aspects of data farming: rapid scenario development, model development, design of experiments, high performance computing, analysis and visualization, and collaborative processes. The final report notes *"... the essence of data farming is that it is first and foremost a question-based approach. The basic question that is asked over and over again in different forms and different contexts is: What if?"*.

Cloud computing [3] is of particular importance to data farming as it offers an expandable computing resource that can scale in or out depending on the requirements of the experiment being analysed. Cloud computing offers an additional benefit in terms of using data farming for HLA performance analysis. Since one of the parameter sets with a potential impact on HLA performance is the computing and network infrastructure, cloud computing and, in particular, the Infrastructure-as-a-Service model can provide a cost effective way of accessing different computing platforms with configurable interconnections.

This paper presents a data farming approach to analyzing the performance of HLA federations. No hypotheses are made regarding what are the key parameters of HLA performance. Instead, a 'performance federation' is constructed and run over a pre-identified space of parameter values. Performance measures recorded by the federation are then analysed to identify the effect the parameters have on HLA performance.

The structure of this paper follows the data farming steps developed in the NATO MSG TG 088 final report. These steps are iterative in nature and illustrated in Figure 1.

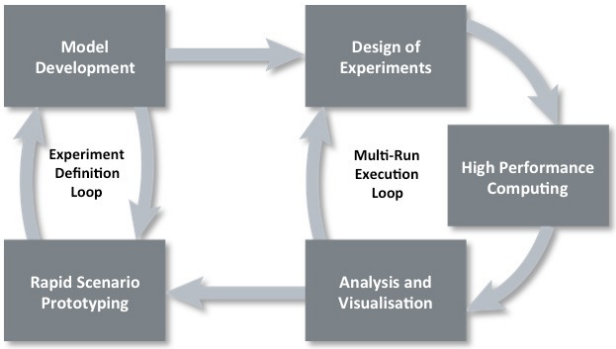

Figure 1: Data Farming "Loop of Loops" (MSG-088 Report)

We begin with Rapid Scenario Prototyping in Section 2 wherein the question to be answered is defined along with input parameters and measures of effectiveness. Section 3 presents some details on the Model Development for the analysis of HLA federation performance. Section 4 outlines the design of the performance experiments including the selection of parameter values. Section 5 provides information on the computing environment in which the performance experiments are run. Section 6 documents the analysis process of the data generated by the experiment runs and presents some visualisations from that data analysis. Section 7 derives answers to the original questions from the data analysis. Section 8 proposes future work to be conducted to continue the discovery of parameters affecting HLA federation performance. Section 9 summarises the data farming process as applied to exploration of HLA federation performance and proposes that a collaborative effort be established within the community to continue the research

for the purpose of making performance-based suggestions in the evolution of the HLA standard.

2 Rapid Scenario Prototyping

We begin with the top-level question: "How do HLA federation parameters affect the performance of those HLA federations?"

This is a very broad question requiring answering at least two sub-questions: "What is HLA federation performance?" and "What are the HLA federation parameters?" To begin answering these questions we first conduct a review of HLA performance research.

2.1 Background on HLA Performance Research

Most HLA performance research has focused on the "infrastructure"—communications, computers, HLA RTI—aspects in terms of measures such as latency and throughput. For example, Knight et al [4] researched the effect on latency and throughput across twelve independent parameters:

- RTI
- Number of federates
- Distribution of federates
- Data Distribution Management
- Network transport mode
- Objects per federate
- Attributes per object
- Interactions per federate
- Parameters per interaction
- Attribute buffer size
- Interaction buffer size
- Data bundling

Drake et al [5] extends this work by adding a Load parameter to the test federates to simulate actual work performed by federates. Also added was an additional metric of one way latency and the calculation of latency jitter rather than just presenting average latency over a number of test iterations.

Malinga and Roux [6] compared round trip latency across two parameters: RTI (three choices) and transport type (reliable or best effort).

Morse et al [7] discuss the impact of attribute level advisories on the scalability of a federation and highlighted the importance of understanding how an RTI implements its features and how those features may be configured to impact performance for the better (or worse).

Burks et al [8] used a one-way latency metric to compare different RTI implementations and, in particular, 'tuned' versions of those RTIs to identify configurations that yielded optimal performance.

Fujimoto and Hoare [9] compared attribute update latency and time advance rate across RTIs using traditional TCP/UDP networking on an Ethernet LAN and an RTI using Myrinet.

Van den Berg et al [10] analysed the performance of a time-managed simulation over a Wide Area Network. A performance model was constructed showing the general time advancement pattern of the federation, the major components, the processing by each component and the messages that are exchanged between the components.

Murray and Faier [11] took a different approach to performance and looked at how aspects of Federation Object Model design impacted processing time and network bandwidth. They found that a single record type attribute, as opposed to multiple attributes representing the fields of the record type, was more efficient in terms of both measures.

Cramp [12] analysed latency, throughput and network utilization across three different methods for implementing an HLA federation supporting the simulation of a domain decomposed into multiple systems of systems.

Karlsson and Johansson [13] suggest that RTI performance is wider than just speed metrics and should also capture aspects related to fault tolerance, stability, development and deployment.

Möller et al [14] argue that this infrastructure, while important, is probably not the limiting factor in the performance of federations. Instead, they propose a conceptual model for causes of federation performance problems that classify issues in terms of the process model, data rates, high-level synchronization, and domain model dependencies.

Watrous et al [15] propose that HLA federation performance is impacted at multiple levels including specific HLA requirements, RTI implementation choices, federation requirements, and physical resource constraints.

2.2 Performance Measures of Effectiveness

From this background research we can identify a few key measures of effectiveness when looking at performance of HLA federations:

- Latency and Throughput of RTI communication: these measure how long it takes a message to propagate between federates (latency) and how many messages (of a given message length) are able to be communicated within a fixed time window (throughput).
- Network overhead of RTI communication: This measure can capture the amount of management data added by the RTI when sending messages. It can also reflect the amount of unnecessary data communicated, e.g., messages that are sent only to be discarded by the receiver.
- Time advance rate: Applicable to time managed federations, this measure captures the speed at which the HLA federation progresses in relation to wallclock time.
- Total time: This is an aggregate measure capturing the total time to complete a simulation and can give a high level view of the performance of a HLA federation.
- Resource usage: The amount of disk space, memory, computing time, network activity, etc used by the HLA federation.
- Other issues: graceful degradation under high load; 'deadtime': is there a window right after a processed interaction during which time the simulation is non-responsive to new events. May be relevant for some (engineering) applications.

2.3 Performance Parameters

The extant body of knowledge reveals a large number of parameters that have been identified and tested for their effects on HLA federation performance. The following subsections attempt to summarise and categorise these parameters. These lists of parameters are, undoubtedly, incomplete and only gain true meaning when the domain of parameters values that are applicable are explored.

Some of these parameters are likely to be invariants in any performance experiment. For example, the infrastructure parameters are probably limited to those available, although infrastructure-as-a-service providers increase the scope to be able to test across these parameters as well.

2.3.1 Infrastructure Parameters

Parameters in this category relate to the type and number of computing and network resources available to run the federation. Such parameters include:

- The computing core
- The memory infrastructure
- The permanent storage infrastructure
- The network infrastructure
- Encryption infrastructure

2.3.2 Platform Parameters

This category captures the software platform running upon the computing infrastructure. This includes all software required to create the runtime environment for a federate. These parameters include:

- The operating system (OS) and configuration
- Runtime support environment (e.g., Java Virtual Machine, Python interpreter)
- Support software
- The RTI

2.3.3 RTI Parameters

The choice of RTI is a parameter in the Platform category. Individual RTIs will also have a large number of parameters that can be used to configure their performance. Historically, these parameters are defined in a file called RTI.rid. A generic list of such parameters is presented below but actual parameters will be dependent on the specific RTI.

- Centralized vs decentralized
- Use of declaration management advisories
- Data Distribution Management (DDM) configuration
- Network message bundling and transmit options

2.3.4 Federate Parameters

These parameters capture how individual federates are developed and, in particular, how the federate operates and interacts with the RTI. This operation and interaction with the RTI is referred to as the federate's process model. It captures details like the work the federate does and how the federate receives information from the RTI, i.e., via evoked callbacks or asynchronous callbacks [16]. Different types of federates will have different impacts on federation performance. For example, a federate designed to log federation data to disk will incur the relatively slow disk access performance and, potentially, pass that on to the overall federation performance. These federate level parameters are of particular importance since they are under direct control of the federate developer. Such parameters include:

- The HLA standard used (dependent on RTI support)
- The programming language and compiler used to develop the federate and interact with the RTI (may differ from the language the RTI is written in)
- The process model used to interact with the RTI. The process model encompasses a number of sub parameters including:
    - Timestep
    - Lookahead

- Time management
- Disk/OS calls
- Number of objects per federate
- Number of interactions per federate
- Size of attribute instances
- Size of parameter instances
- Frequency of updates and sends

2.3.5 Federation Parameters

Most parameters fall into this category as it relates to how a federation is designed and how federates interact with each other during a running federation execution. These parameters include:

- Federation Object Model (FOM) design
- Number of federates
- Distribution of federates across available computing resources
- Distribution of RTI components
- Domain coupling between federates
- Simulation running time (to capture any effects caused by accumulation of factors)
- Time representation

2.4 Scope of Parameters and Measures for this Paper

The previous sections highlighted the large number of measures and parameters related to HLA federation performance.

For the purpose of this paper we concentrate on a constructive, time managed federation with measures being rate of time advance grants compared to wallclock time and overall simulation wallclock time. The parameters and metrics proposed and researched in the literature will indirectly impact these higher level, time management metrics. We choose to start at the top level and, in the future, delve down to the lower more fine grained parameters and metrics.

For this paper, the parameters of interest for their impact on the metrics are:

- Number of federates
- Simulation running (end) time
- Timestep per federate
- Lookahead per federate
- Time parameters for the evokeCallback and evokeMultipleCallbacks based federate process model

These measure and parameter choices dictate the development of the model to be used to explore the parameter value space and the resulting captured metrics.

3 Model Development

One approach to the development of a model for testing HLA federation performance is to develop a conceptual representation of the HLA and conduct the testing within this representation. Gianni et al [17] take this approach by representing a HLA federation as an Extended Queueing Network (EQN) and performing performance simulations on this model.

We take the approach of building a testing federation that can be run to test input parameters by analyzing captured output.

The testing federation is composed of multiple test federates that implement the test logic and log the measured metrics. The lifecycle of these test federates is managed by a single execution manager federate that uses interactions and federation sychronisation points to manage the start of the test execution. A management script captures the scenario parameters and iterates over all combinations of the parameter values and controls the starting and stopping of the federate processes. The source code for this test environment is available on Github (https://github.com/anthonycramp/hla-performance). Each configuration of parameter values is iterated multiple times to average any transient work load resident on the host computer. In particular, the first iteration is usually ignored due to it incurring 'warm-up' artefacts that won't exist for subsequent iterations.

The architecture of the performance test environment is illustrated in the sequence diagram presented in Figure **2** and described in the following subsections.

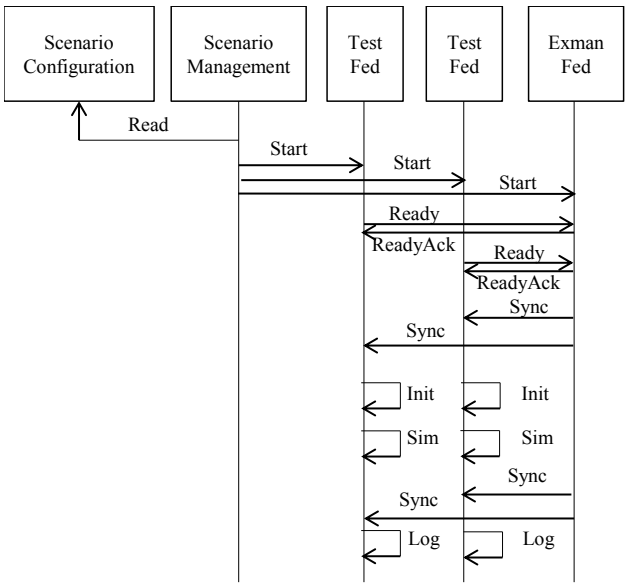

Figure 2: Sequence diagram for the performance test environment.

3.1 Test Federate

The test federates accept command line arguments in order to be able to set the parameter values for timestep, lookahead, min and max time for evokeCallback, and the end time of the simulation.

The test federate algorithm proceeds as follows:

1. Connect to the RTI, attempt to create the federation execution (using RPR-FOM as the Federation Document Data (FDD)), attempt to join the federation execution (using a name specified on the command line)
2. Periodically send a Ready interaction, with the name of the federate as a parameter instance, until a ReadyAck interaction is received, with a matching federate name (these interactions are defined in a custom exman.xml FOM module)
3. Wait for the announcement of a synchronization point and, once announced and achieved, wait for the federation to be synchronized. The execution manager federate is responsible for registering this synchronization point.
4. In order to exercise the RTI a little, the test federate publishes and subscribes to the Spatial attribute of the BaseEntity.PhysicalEntity.Platform.SurfaceVessell object class. For this paper, test federates don't register/discover object instances, update/reflect attribute values or send/receive interactions. These will be added in the future.
5. The federate enables time constraint and regulation using the default HLAfloat64 time representation.
6. A simulation loop is entered whereby a time advance grant to current time plus timestep is requested and the grant waited upon. The waiting is executed either via the single argument call to evokeCallback or the two argument call to evokeMultipleCallbacks. The loop exits once a predetermined simulation end time is reached.
7. After exiting the simulation loop, the test federate waits for a synchronization point to be announced (registered by the execution manager federate). Once it has been announced, the test federate achieves the synchronization point and waits for the federation to be synchronized.
8. Finally, the test federate attempts to resign from and destroy the federation execution and then terminates.

The test federate records the current system time[3] in memory at key points. This logging is carefully managed so as to impose minimal overhead on federate performance. For example, the memory for the logging structure is pre-allocated so that no dynamic memory allocations (due to logging) occur.

System time is recorded before entering the simulation loop and after exiting the loop. This time delay is the simulation time measure. Also, system time is recorded each pass through the simulation loop in order to measure the time advance rate. These measures, along with the parameter values for this test configuration, are written to a text file before the federate terminates but after all simulation measurements have occurred so as to not capture disk access time in the measured values.

3.2 Execution Manager Federate

The execution manager federate is simply responsible for the registration of the synchronization points that act as barriers for the test federates to beginning simulation execution and shutting down. The execution manager federate is provided with a list of federate names it expects will join the federation. At startup, the execution manager listens for Ready interactions that are sent from the test federates. These interactions contain the name of the federate sending the interaction and are sent periodically by each test federate. Once a Ready interaction is received, the execution manager records the presence of that test federate and sends a ReadyAck interaction to tell the test federate to stop sending the Ready interactions[4]. Once all expected federates have been registered, the execution manager registers the first synchronization point indicating the start of the simulation. Once the federation is synchronized at this point, the execution manager immediately registers the end synchronization point and waits for all the federates to achieve it. The execution manager then attempts to resign and destroy the federation execution and then terminates.

3.3 Scenario Management

A script program manages the test federation. The values of the test parameters are captured in this script. The script iterates over all valid combinations of these parameter values (the validity conditions are also captured in the script) and launches the test federates and execution manager federate with the appropriate command line arguments. Each valid combination is executed multiple

[3] The test federate is written in Java and System.nanoTime() is used to query the system timer. All measures are time periods between two calls to this method.

[4] This process could be implemented using the Management Object Model (MOM). However, the RTI under test did not, at the time of writing, provide MOM functionality.

times in order to gain an average performance across any computing artefacts that may occur during execution.

The script monitors the 'liveness' of each of the federates and begins the next combination of parameter values once it has detected that all federates have terminated.

4 Design of Experiments

Choice of the values for these parameters needs to be informed so as to limit the need to test all possible combinations, the number of which can rapidly explode. Sanchez [18] and NATO MSG-088 [2] describe various methods for intelligently choosing values. The simplest approach is referred to as a 2^k factorial design or a coarse grid. This approach basically takes a low value and a high value for each of the k parameters so as to generate a general first order effect arising from that particular parameter. This is the approach taken for this paper.

In addition to the 2^k factorial design, some parameter values impact on the choice of other parameter values. For example, the timestep value of a federate must be greater than or equal to the lookahead of that federate since it is not possible to request a time advance to a point in time less than the federate's current time plus its lookahead. Also, the minimum value of the evokeMultipleCallbacks method must be less than the maximum value.

The following table presents the values for the parameters. These are chosen somewhat arbitrarily since the purpose is to look for answers in the data instead of predicting how the values will impact performance.

Parameter	Low Value	High Value
#Federates	2	8
End simulation time	20	100
Timestep	0.01	1.0
Lookahead	0.01	1.0
evokeCallback	0.001	0.1
evokeMultipleCallbacks min time	0.001	0.1
evokeMultipleCallbacks max time	0.1	1.0

These parameter values give a total of 96 combinations. The first five parameters yield 32 (2^5) combinations. The final two parameters yield three combinations due to the restriction that the min time be less than the max time. Thus, the total combinations are 32x3=96.

In addition, each of these 96 combinations are run five times yielding a total of 480 federation executions.

The RTI and computing configuration are kept as invariants. The RTI is kept anonymous as the RTI's developer has not (at time of writing) been consulted about the performance results. Suffice it to say that the RTI is a Java implementation and supports the IEEE 1516-2010.1 interface specification.

The computing configuration is named Machine M: Apple MacBook Pro with a 2.8GHz Intel Core i7, 16GB of 1600MHz DDR3 RAM and running Mac OS X 10.9.3 and Java Version 1.7.0_55.

5 High Performance Computing

The scope of the performance experiments for this paper have been deliberately kept small to focus on the data farming process. As such, the size of the parameter space is limited and a high performance computing environment is not needed. Thus, the test federation is able to be run manually. While not employed for the performance results in this paper, extra computing configurations could be accessed through Infrastructure-as-a-Service providers such as Amazon Web Services (AWS) or Google Compute Engine (GCE).

These services provide the ability to provision virtual machines of specific configurations (although, of these two, only Amazon provides Windows virtual machines). Both services also provide the ability to create a virtual network of virtual machines that is partitioned off from the rest of the Internet. This could be employed for testing the performance across a network of computers. Although, care would need to be taken in the interpretation of results due to the performance penalty of using virtualized and shared resources. At best, performance should be reported relative to a baseline established on the virtualized resources.

However, both AWS and GCE do not allow multicast or broadcast network traffic within these virtual networks, which would limit applicability since most RTIs use multicast at least during the initial discovery phase. As such, in order to achieve a similarly virtualized test environment, other infrastructure-as-a-service providers or the establishment of a private data farming environment will be researched. With respect to the latter approach, the NATO MSG-088 report [2] describes three such environments established by United States, Germany and Singapore.

6 Analysis and Visualisation

Each of the test federates record the performance measures in text files. These text files are written to a directory created per federation configuration.

The data analysis process begins with data collection and curation. A script traverses all the data output and records it into a relational database table with columns for each of the parameters and a column each for the total simulation time, the average time advance rate, the minimum time advance rate and the maximum time advance rate. Having all this data collated in a central database allows for efficient querying and visualization of results across different parameter dimensions.

The data in the resulting database is effectively an n-dimensional cube that can be analysed and visualized using typical online analytical processing (OLAP). The cube can be sliced and diced to reduce its dimensionality by picking particular values for the parameter values. If the raw step-by-step time advance rates were retained or linked then that data could be drilled down to from the top-level aggregate data. This would be particularly important if the data revealed a wide range between the minimum and maximum time advance rate.

Manipulation and visualisation of this n-dimensional data is best done via an interactive tool allowing a user to step in and out of the data space. Such a tool would have support for quickly identifying outliers and other interesting data features. Machine learning and data mining algorithms such as classification, clustering and regression can assist in this process.

It is probably more efficient for this small testing federation to have each of the federates write their results directly to the final database. However, as the parameter space increases and the number of simultaneous federation executions grows with the employment of high performance computing resources, writing text files to a local directory is an efficient and scalable way of capturing results. The subsequent analysis step becomes more complex but can be facilitated by technologies and tools like MapReduce[5] and Hadoop[6].

6.1 Example Analysis and Visualisation

Data from performance runs are presented in Table 1 in Section 12 at the end of this paper. The PM column refers to the process model used by the federate and is coded as follows:

- P1: evokeCallback(0.001)
- P2: evokeCallback(0.1)
- P3: evokeMultipleCallbacks(0.001, 0.1)
- P4: evokeMultipleCallbacks(0.001, 1.0)
- P5: evokeMultipleCallbacks(0.1, 1.0)

The Average Simulation Time is the average over all federates in each federation and then over iterations 2, 3, and 4 (results from iterations 1 and 5 are discarded as warm up and cool down iterations respectively). The Minimum and Maximum Simulation Time columns are calculated similarly.

The parameters end time and timestep affect how many calls each federate makes to request a time advance during a simulation. As such, it is not possible to directly compare simulation times where the end time and timestep differ. Therefore, a secondary parameter of "number of frames" is introduced to capture how many time advance requests are made by a federate. The simulation time is divided by the number of frames to get an average simulation time per frame metric that is suitable for comparison.

The plot below (Figure 3) presents the average simulation time (seconds) per frame (y-axis) for the five different process models (x-axis) across all values of end time, timestep and lookahead. Two series of values are presented: one for federations with two federates and the other for federations with eight federates.

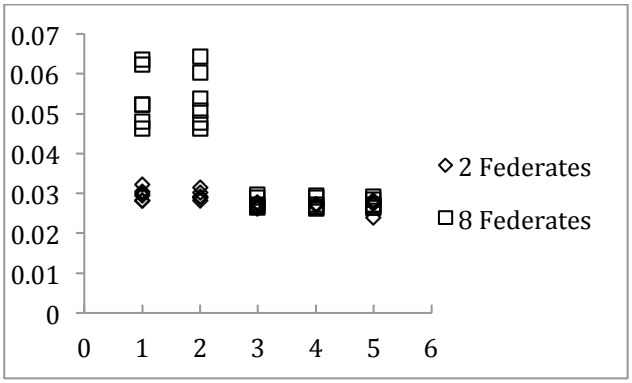

Figure 3: Average simulation time per frame vs process model for 2 and 8 federate federations

From this plot it is seen the that P1 and P2 process models result in longer simulation times. However, the simulation times for P1 and P2 in the 8 federate case are almost twice the simulation times for the P3, P4 and P5 process models. In the two federate case the extra simulation time for the P1 and P2 process models is only around 4%.

Surprisingly, for the P3, P4 and P5 process models, the simulation time is largely similar across two or eight federates indicating decent scalability (in this simplistic federation) for this RTI. It also indicates that perhaps the values for the number of federates should be changed and further experimentation undertaken to identify the scalability inflection point.

[5] http://research.google.com/archive/mapreduce.html
[6] http://hadoop.apache.org/

The variance in the results for P1 and P2 for the eight federate case indicates that there is likely some dependency between those results and the hidden parameters of end time, timestep and/or lookahead. A look through the data implies that a smaller timestep yields a lower average simulation time per frame for the same end time. Also, a larger end time yields a lower average simulation time frame for the same time step. Different lookahead values seem to have no impact on average simulation time per frame. However, smaller end times and larger end times yield more frames resulting in greater overall simulation time so care needs to be taken in applying these results directly.

Average simulation time per frame hides any anomolies that may occur in the actual simulation time per frame encountered by a federate. Figure 4 below (data is inTable 2 in Section 12) scatter plots the actual simulation time (seconds) per frame for the P1 and P3 process models in a federation with eight federates. The simulation time per frame (basically the time taken from time advance request to receiving a time advance grant in this test environment) is very jittery for process model P1. The values for process model P3 are much more consistent. Similar results to P1 could be plotted for P2. Also, process models P4 and P5 yield similar results to P3.

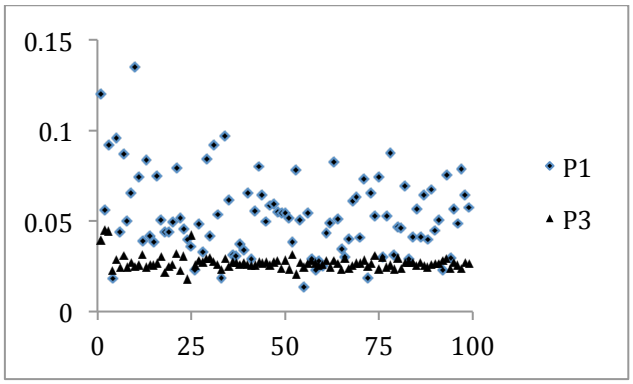

Figure 4: Actual simulation time (seconds) per frame for process models P1 and P3, 8 federate federation, end time 100s, timestep 1.0s, lookahead 0.01s

The reason for the difference in performance of P1 and P2 compared to P3, P4 and P5 is due to it taking a lot more calls to evokeCallback (P1 and P2) before a time advance grant is returned. Compared to the number of calls required of evokeMultipleCallbacks (P3, P4 and P5). For the RTI under test, it takes over 100,000 calls to evokeCallback on one frame, but the very next frame can take as little as three calls. The following frame goes back to requiring over 100,000. The up and down pattern for evokeMultipleCallbacks also exists but the larger number of calls required is only around 30.

7 Answers

Given the analysis so far it is evident that for the used RTI, process models P1 and P2 result in poorer performance than for performance models P3, P4 and P5. Also, in regards to performance models P3, P4 and P5, the actual timing values supplied seem to not matter.

8 Future Work

The intent is to expand the parameter set and range of values per parameter to generate a fuller data space from which to derive correlations (and, hopefully, causations) between parameter values and HLA federation performance. With this expanding scope, a new test environment (either using an existing public cloud or developing an internal private cloud) will be used to allow completion of the performance runs in a timely manner. Additional performance measures will be captured to provide more insight into the tradeoffs engendered in the selection of various parameter values. Additionally, non-federation metrics will be captured to identify second order effects on performance such as third-party application CPU load, memory usage, disk access, and network use. These secondary metrics are particularly important to capture in public, virtualized, shared computing platforms.

The increased data space will require the employment of data mining algorithms to identify key parameter sets and, finally, in the derivation of a performance model that is representative of the data collected.

9 Summary

We have presented a data farming approach to exploring and analyzing the space of parameters that potentially affect the performance of HLA federations. The parameter space is potentially huge and experimenting in such a space requires support from several fields captured in the data farming methodology: rapid scenario prototyping, model development, design of experiments, high performance computing and analysis and visualization.

One step of the data farming process as reported by the NATO MSG-088 task group has not been discussed yet in this paper. Collaborative processes covers the teamwork required to maintain a healthy data farm from the experiment designers through the computing team and on to the analysts. We end this paper by proposing a performance team with the necessary skills be established within the Simulation Interoperability Standards Organisation (SISO) community to continue the research

on HLA performance with a view to making performance based suggestions on the evolution of the HLA. In order to establish a level playing field the team would start by defining agreed methods and procedures to measure HLA performance.

10 References

[1] Gary E. Horne, Ted E. Meyer, *"Data Farming: Discovering Surprise"*, Proceedings of the 2004 Winter Simulation Conference.

[2] NATO Science and Technology Organisation Modelling and Simulation Task Group 088, *"Data Farming in Support of NATO"*, STO Technical Report TR-MSG-088, AC/323(MSG-088)/TP/548, March 2014.

[3] Peter Mell and Timothy Grance, *"The NIST Definition of Cloud Computing"*, National Institute of Science and Technology, U.S. Department of Commerce, Special Publication 800-145.

[4] Pamela Knight et al, *"Analysis of Independent Throughput and Latency Benchmarks for Multiple RTI Implementations"*, Simulation Interoperability Workshop, September 2002. 02F-SIW-068.

[5] Ray Drake et al, *"Independent Benchmarking of RTI Real-Time Performance"*, Simulation Interoperability Workshop, September 2003. 03F-SIW-024.

[6] L. Malinga and Willem H. le Roux, *"HLA RTI Performance Evaluation"*, Simulation Interoperability Workshop, July 2009. 09E-SIW-005.

[7] Katherine L. Morse, Frank J. Hodum and Peter M. Wickis, *"Attribute Level Advisories and Scalability"*, Simulation Interoperability Workshop, September 2001. 01F-SIW-066.

[8] Terrell Burks et al, *"Latency Performance of Various HLA RTI Implementations"*, Simulation Interoperability Workshop, April 2001. 01S-SIW-015.

[9] Richard Fujimoto and Peter Hoare, *"HLA RTI Performance in High Speed LAN Environments"*, Simulation Interoperability Workshop, September 1998.

[10] Tom van den Berg et al, "*Execution Management Solutions for Geographically Distributed Simulations*", Simulation Interoperability Workshop, Spring 2009. 09S-SIW-008.

[11] Bob Murray and Adam Faier, *"Designing FOMs for Performance"*, Simulation Interoperability Workshop, September 2000. 00F-SIW-116.

[12] Anthony Cramp, *"Simulation Multiple Systems of Systems Using The High Level Architecture"*, PhD Thesis, University of Adelaide, 2009.

[13] Peter Karlsson and Magnus Johansson, *"RTI performance in a wider scope"*, Simulation Interoperability Workshop, June 2003. 03E-SIW-074.

[14] Björn Möller, Mikael Karlsson, Björn Löfstrand, *"Common Federation Performance Bottlenecks"*, Simulation Interoperability Workshop, September 2005. 05F-SIW-044.

[15] Ben Watrous, Len Granowetter, Douglas Wood, *"HLA Federation Performance: What Really Matters?"*, Simulation Interoperability Workshop, September 2006. 06F-SIW-107.

[16] Mikael Karlsson and Peter Karlsson, *"An In-Depth Look at RTI Process Models"*, Simulation Interoperability Workshop, March 2003. 03S-SIW-055.

[17] Daniele Gianni, Paolo Bocciarelli and Andre D'Ambrogio, *"Model-Driven Performance Prediction of HLA-Based Distributed Simulation Systems"*, Proceedings of the 2012 Winter Simulation Conference.

[18] Susan M. Sanchez, *"Work Smarter, Not Harder: Guidelines for Designing Simulation Experiments"*, Proceedings of the 2006 Winter Simulation Conference.

11 Author biographies

ANTHONY CRAMP is a Defence Science and Technology Organisation (DSTO, Australia) International Fellow working within the Modeling, Simulation and Gaming department of TNO Defence, Security and Safety, The Netherlands until February 2015 on topics including M&S as a Service and HLA performance. His research interests include distributed systems and simulations, software architectures, and programming languages. From March 2015 he can be contacted via his DSTO email address: anthony.cramp@dsto.defence.gov.au.

TOM VAN DEN BERG is a scientist in the Modeling, Simulation and Gaming department at TNO Defence, Security and Safety, The Netherlands. He holds an M.Sc. degree in Mathematics and Computing Science from Delft Technical University. His research area includes simulation systems engineering, distributed simulation architectures and concept development & experimentation.

WIM HUISKAMP is Chief Scientist Modelling, Simulation and Gaming in the M&S department at TNO Defence, Security and Safety in the Netherlands. Wim leads TNO's research program on Simulation, which is carried out on behalf of the Dutch MOD. Wim is a member of the NATO Modelling and Simulation Group (NMSG) and currently acts as its Chairman. He has also chaired several NMSG Technical Working groups, including the NMSG M&S Standards Subgroup (MS3) and he is the liaison of the NMSG to the Simulation Interoperability Standards Organization SISO.

12 Data

Table 1: Simulation Data Results

#FED	PM	EndT (s)	TS (s)	#Frames	LHD (s)	AVG SIM T (ns)	MIN SIM T (ns)	MAX SIM T (ns)	AVG ST/FRAME (s)
2	P1	20	0,01	2000	0,01	56,46032717	56,252417	56,698317	0,02823
2	P2	20	0,01	2000	0,01	57,903679	57,802833	57,985213	0,028952
2	P3	20	0,01	2000	0,01	54,790349	54,680757	54,852616	0,027395
2	P4	20	0,01	2000	0,01	54,66029583	54,587349	54,713394	0,02733
2	P5	20	0,01	2000	0,01	55,044595	53,826493	55,72255	0,027522
8	P1	20	0,01	2000	0,01	96,02848358	95,530187	96,327429	0,048014
8	P2	20	0,01	2000	0,01	95,37848029	94,772291	96,115905	0,047689
8	P3	20	0,01	2000	0,01	53,31894283	53,124708	53,502852	0,026659
8	P4	20	0,01	2000	0,01	52,65814558	52,216342	53,414581	0,026329
8	P5	20	0,01	2000	0,01	53,08759292	52,189616	53,710031	0,026544
2	P1	20	1	20	0,01	0,641514333	0,582148	0,685257	0,032076
2	P2	20	1	20	0,01	0,629404333	0,547157	0,704944	0,03147
2	P3	20	1	20	0,01	0,553491833	0,550598	0,558404	0,027675
2	P4	20	1	20	0,01	0,544958667	0,524619	0,555382	0,027248
2	P5	20	1	20	0,01	0,479461167	0,376466	0,562291	0,023973
8	P1	20	1	20	0,01	1,246927875	1,170826	1,313135	0,062346
8	P2	20	1	20	0,01	1,203941375	1,167344	1,247632	0,060197
8	P3	20	1	20	0,01	0,577439083	0,554209	0,640085	0,028872
8	P4	20	1	20	0,01	0,582201333	0,562886	0,630617	0,02911
8	P5	20	1	20	0,01	0,566887833	0,553592	0,61027	0,028344
2	P1	20	1	20	1	0,6054065	0,522492	0,663463	0,03027
2	P2	20	1	20	1	0,603123	0,531656	0,658796	0,030156
2	P3	20	1	20	1	0,535834167	0,534216	0,537324	0,026792
2	P4	20	1	20	1	0,542841667	0,533754	0,559383	0,027142
2	P5	20	1	20	1	0,562990833	0,560639	0,564883	0,02815
8	P1	20	1	20	1	1,272072875	1,136138	1,43314	0,063604
8	P2	20	1	20	1	1,285272042	1,199841	1,384903	0,064264
8	P3	20	1	20	1	0,595028792	0,559139	0,656344	0,029751
8	P4	20	1	20	1	0,586511333	0,539842	0,667346	0,029326
8	P5	20	1	20	1	0,5856945	0,546134	0,678086	0,029285
2	P1	100	0,01	10000	0,01	280,1258333	277,624	282,492	0,028013
2	P2	100	0,01	10000	0,01	280,5368333	277,696	283,642	0,028054
2	P3	100	0,01	10000	0,01	267,701	265,298	271,055	0,02677
2	P4	100	0,01	10000	0,01	270,8658333	270,561	271,024	0,027087
2	P5	100	0,01	10000	0,01	274,5218333	274,412	274,594	0,027452
8	P1	100	0,01	10000	0,01	462,4359167	462,049	462,959	0,046244
8	P2	100	0,01	10000	0,01	462,6778333	460,617	463,78	0,046268
8	P3	100	0,01	10000	0,01	264,4117083	263,87	264,832	0,026441
8	P4	100	0,01	10000	0,01	261,104	259,09	264,637	0,02611
8	P5	100	0,01	10000	0,01	264,356375	261,632	265,868	0,026436
2	P1	100	1	100	0,01	2,991847333	2,921354	3,069313	0,029918
2	P2	100	1	100	0,01	2,883334667	2,766244	2,991599	0,028833
2	P3	100	1	100	0,01	2,619080167	2,575113	2,679672	0,026191
2	P4	100	1	100	0,01	2,699565167	2,653196	2,757232	0,026996
2	P5	100	1	100	0,01	2,806422167	2,799595	2,81017	0,028064
8	P1	100	1	100	0,01	5,217799208	5,035349	5,343164	0,052178
8	P2	100	1	100	0,01	5,076578792	4,958876	5,178918	0,050766
8	P3	100	1	100	0,01	2,708294833	2,680744	2,760317	0,027083
8	P4	100	1	100	0,01	2,679911583	2,665196	2,724619	0,026799
8	P5	100	1	100	0,01	2,709734667	2,687948	2,760207	0,027097
2	P1	100	1	100	1	2,948851833	2,874261	3,009218	0,029489
2	P2	100	1	100	1	2,916675167	2,822951	3,019887	0,029167
2	P3	100	1	100	1	2,7529615	2,735937	2,768221	0,02753
2	P4	100	1	100	1	2,7092835	2,595965	2,768161	0,027093
2	P5	100	1	100	1	2,7995955	2,783733	2,809897	0,027996
8	P1	100	1	100	1	5,2292615	5,137961	5,387298	0,052293
8	P2	100	1	100	1	5,3708665	5,144697	5,62323	0,053709
8	P3	100	1	100	1	2,699719667	2,653402	2,786291	0,026997
8	P4	100	1	100	1	2,723542792	2,686884	2,788623	0,027235
8	P5	100	1	100	1	2,696968417	2,67522	2,750964	0,02697

Table 2: Actual Simulation Time (seconds) per frame for process models P1 and P3 in an 8 federate federation, end time 100, timestep 1.0, lookahead 0.01

Frame#	P1	P3	Frame#	P1	P3
1	0,120166	0,039268	51	0,051834	0,023605
2	0,056189	0,045185	52	0,038469	0,031157
3	0,091901	0,044554	53	0,078223	0,02076
4	0,018081	0,022407	54	0,050557	0,027199
5	0,095939	0,028863	55	0,013533	0,02467
6	0,044147	0,024363	56	0,054487	0,026227
7	0,08718	0,030903	57	0,028803	0,028015
8	0,049953	0,024443	58	0,022727	0,026629
9	0,065286	0,027228	59	0,027693	0,025817
10	0,13517	0,024959	60	0,024871	0,025898
11	0,07433	0,025522	61	0,043377	0,028255
12	0,038612	0,031374	62	0,048992	0,024431
13	0,083616	0,024624	63	0,082412	0,02812
14	0,04178	0,026148	64	0,051083	0,026211
15	0,038479	0,02561	65	0,034333	0,023054
16	0,07461	0,026558	66	0,029924	0,029333
17	0,050893	0,030336	67	0,040093	0,023949
18	0,044009	0,021406	68	0,060876	0,025426
19	0,043753	0,025099	69	0,063518	0,026558
20	0,04932	0,025951	70	0,040979	0,026739
21	0,079514	0,031891	71	0,073107	0,028445
22	0,051578	0,022522	72	0,018488	0,024803
23	0,045699	0,030613	73	0,065615	0,026395
24	0,039756	0,017859	74	0,053004	0,031009
25	0,035749	0,042144	75	0,074425	0,023585
26	0,022699	0,025798	76	0,029784	0,029766
27	0,048275	0,028081	77	0,052591	0,024724
28	0,032846	0,027362	78	0,087645	0,025828
29	0,084444	0,029198	79	0,031276	0,023537
30	0,041743	0,029511	80	0,046381	0,029477
31	0,092034	0,027916	81	0,046295	0,023986
32	0,053512	0,026133	82	0,069475	0,027097
33	0,018445	0,023152	83	0,028936	0,027397
34	0,096744	0,029149	84	0,041218	0,0269
35	0,061424	0,024881	85	0,056892	0,025145
36	0,031332	0,027455	86	0,041127	0,026944
37	0,030513	0,027072	87	0,064598	0,025362
38	0,037442	0,026051	88	0,039732	0,024759
39	0,03365	0,026881	89	0,067278	0,025879
40	0,065737	0,026139	90	0,044579	0,026424
41	0,029295	0,025365	91	0,050313	0,026215
42	0,055221	0,025659	92	0,022893	0,028216
43	0,079971	0,027335	93	0,075467	0,029063
44	0,06414	0,026825	94	0,029615	0,02386
45	0,049686	0,027415	95	0,056652	0,027053
46	0,058595	0,025798	96	0,048522	0,025604
47	0,059421	0,027529	97	0,078523	0,024051
48	0,055169	0,028101	98	0,064364	0,02709
49	0,05429	0,023897	99	0,057363	0,026525
50	0,054583	0,028325			