Proceedings of the 1998 Winter Simulation Conference
D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.

SCALABLE MEANS MORE THAN MORE: A UNIFYING DEFINITION OF SIMULATION SCALABILITY

Darren R. Law

Science Applications International Corporation
12479 Research Parkway
Orlando, Florida 32826-3248 U.S.A.

ABSTRACT

The word "scalability" is used in a variety of ways by different simulation communities. This paper describes some of the more common usages and presents a general, unifying definition of simulation scalability which addresses the intent of these differing usages. Many common definitions of scalability can be viewed as simple restrictions of this multivariate scalability function to some subset of the variables in its domain. The quantitative nature of this definition allows systems to be compared based on their scalability instead of their relative performance at some level of capability. The utility of the proposed general and restricted definitions of scalability is discussed.

1 INTRODUCTION

In recent years, the military simulations community has placed an increasing emphasis on scalability and scalable simulation systems. Much of this increased emphasis is in response to the need to simulate increasingly large numbers of entities in the battlefield environment. In fact, the word "scalable" is often used as shorthand for "simulates more entities." In comparing the differing interests of the largely academic parallel and distributed simulation (PADS) community and the military simulation community, Fujimoto identifies speedup as the primary metric for simulation performance in the PADS community and scalability as a primary metric in the military simulation community (Fujimoto 1995).

Speedup is a well defined, quantifiable measure; in contrast, scalability is widely understood but rarely quantified. It is often said that a particular simulation is scalable, but no one can say how scalable it is. This paper presents a very general definition of scalability which addresses the intent of the most commonly used definitions while providing a unifying set of terms and relationships which allow us to quantify scalability. The definitions presented here were developed to quantify the scalability of simulations, but nothing in these definitions explicitly constrains them to the simulation domain.

2 COMMON USAGE

Steinman describes scalability as it might be viewed by individuals with backgrounds in parallel simulation, networking, distributed optimistic simulation, or software engineering (Steinman 1995). Implicit in each of the descriptions presented is the assumption that scalability is simply a property which something must have in order to be called scalable. This usage of scalability is certainly correct, but it is not quantifiable and clearly depends on the user's definition of scalable.

In the domain of parallel computation, "An algorithm is scalable if the level of parallelism increases at least linearly with the problem size. An architecture is scalable if it continues to yield the same performance per processor, albeit used on a larger problem size, as the number of processors increases" (Quinn 1994). This definition of scalable is ubiquitous in the parallel computation community. It is clear from this definition that an algorithm or architecture either has scalability or it does not. There is no explicit quantification of scalability, though one could reasonably argue that the efficiency of a parallel computation suffices to describe the scalability.

The military simulation community uses several definitions for scalability. The Defense Modeling and Simulation Office (DMSO) defines scalability as "the ability of a distributed simulation to maintain time and spatial consistency as the number of entities and accompanying interactions increase" (DoD M&SMP 1995). This definition specifically identifies scalability as a property of distributed simulation, but leaves the quantifiability of scalability open to interpretation. It explicitly addresses the spatial and temporal consistency which are implicit requirements in the PADS community (Fujimoto 1995), but makes no distinction between the scalability of an algorithm and an architecture since it merges the two in the term "distributed simulation." This

definition also implies that scalability is a property particular to distributed simulations.

The most common use of scalable in military simulations is that given in the introduction; a system is described as scalable if it can simulate more entities than some alternative system. This definition makes the implicit assumption that any interesting properties of the simulation are entirely a function of the number of simulated entities. This is clearly not the case, as observed in (Pratt 1996, Cavitt *et al.* 1996, Harless and Rogers 1995). The researchers using this definition frequently conclude that one of two compared algorithms or systems is more scalable than the other.

Commercial use of the term "scalability" is increasingly commonplace and very ill defined. In (IBM 1997) scalability is defined as "the ability to incrementally grow a company's information systems to handle dramatic increases in usage…". This definition, while not quantifiable, is rigorous by advertising standards. The more general usage simply defines a system as scalable if the customer can solve larger problems by buying more systems or more powerful systems of the same type. No particular relationship between system cost and system performance is inherent in this definition.

Many researchers have evaluated the scalability of simulation systems and reached the conclusion that the examined system is or is not scalable. Blais recognizes that scalability has architectural and algorithmic components, but states that there is "no clear criteria by which this judgement can be made" in describing the issue of evaluating system scalability (Blais 1995). Smith also identifies the need for a "simple, compact characterization of inherent scalability" (Smith *et al.* 1996). The following section presents a definition for scalability that is both general and quantifiable.

3 DEFINITION OF SCALABILITY

The most naïve definition of simulation scalability is "the ability of a simulation to get bigger;" this simplistic definition captures the essential goal of most scalability research. Unfortunately, this definition is neither quantifiable nor particularly useful.

It is widely (but not universally) recognized that scalability is not a characteristic of hardware or software, but of both (Greenfeld 1997, Blais 1995, Quinn 1994). The DMSO definition of scalability presented in Section 2 dealt with this issue by encapsulating hardware and software into the term "simulation", but many researchers hold hardware constant and evaluate software, or vice versa. To accommodate both of these views, our definition of scalability will address hardware and software separately, then characterize scalability in terms of their relationship.

Simulation can be informally defined as "the cost-effective use of something in place of something else." Some measurement or characterization of simulation cost or performance is at the core of every definition of scalability; the tradeoff between system resources (cost) and system performance lies at the heart of simulation scalability. Definition 1 provides the framework for a later definition which quantifies this tradeoff by adding the notion of cost-effectiveness to the naïve definition presented earlier in this section.

Definition 1:

A *scalable* simulation is one that exhibits improvements in simulation capability in direct proportion to improvements in system architectural capability.

Definition 1 corresponds closely to the definition of hardware scalability given by (Blais 1995). This focus on system scalability with respect to hardware is often the most appropriate, since it most closely approximates a "performance per dollar" metric. In the design stage, we can use performance per dollar to predict the relationship between system cost and system capability. After a system is implemented, we can continue to use this information to estimate the costs of improving system capability.

This definition of scalable focuses our evaluation on tradeoff costs at the interface between the architecture (hardware and operating system) and the simulation. This is not unreasonable, since the architecture for many simulation systems is purchased off the shelf, but it does force us to view scalability as a function of the tradeoffs between architectural capability and simulation capability. It is important to note that architectural capability and simulation capability are not necessarily independent variables.

It is clearly difficult to quantify any of the variables in Definition 1. By refining this definition into more quantifiable elements, we will develop a measure of the relative scalability of two systems. To achieve this, we must quantify simulation capability and architectural capability.

3.1 Simulation Capability

A simulation exists to satisfy some cognitive purpose; to satisfy this purpose, it must compute some set of values that characterizes the state of the simulation model. We will refer to these values as the simulation's measures of interest. The purpose of the simulation dictates the fidelity and resolution requirements for the computation of these measures of interest.

In addition to computing the measures of interest, a simulation system must often satisfy other performance requirements related to these computations. Human-in-the-loop simulations often place some upper bound on the time

allowed to communicate information but tolerate occasional message loss. Discrete-event simulations often demand perfect communications reliability but accept arbitrarily long communication delays to achieve it (Fujimoto 1995). An analytical model and a training simulation may compute the same measures of interest but have very different performance requirements. If a simulation system computes its measures of interest with the required fidelity and resolution and meets these computational performance requirements, we will say that the simulation system has achieved acceptable performance.

To help illustrate the subtler aspects of simulation capability, let us consider the "Dynamic Line-of-Sight (LOS) Problem". The Dynamic LOS Problem is the problem of evaluating lines of sight among a collection of n tanks. These tanks move through a simulated terrain and attempt to detect each other visually. If any of these tanks is controlled by a human operator, we must display what the operator can see at a reasonable level of fidelity to make that control possible. This additional performance requirement demands significantly more computational resources for the same measures of interest.

We have stated that simulation capability is related to its measures of interest and system performance requirements. To relate simulation capability to scalability, it is clear that system capability must include some measure of problem size. Certainly, we view a system that can simulate "more" than another does as being more capable. In the Dynamic LOS Problem n, the number of tanks, is the most obvious characterization of simulation size. It is clear, however, that various factors relating to the terrain and its representation may affect the computational time and space required to simulate the described situation.

The measures of interest and system performance requirements define the underlying problem that our simulation must solve. If we can characterize the size of this problem, then we can derive analytical results about its time and space complexity. The complexity of the underlying problem dictates the best performance that we can hope to achieve with any algorithm. If it is mathematically impossible to evaluate lines of sight among n tanks in less than kn^2 time, it is useful to know that before we attempt to develop algorithms to solve the problem.

It is obvious that the size of a simulation problem may not be a simple function of a single variable. For purposes of definition, let us suppose that a simulation's size can be characterized as a function of some set of s variables, $n_1...n_s$. We will denote the size of the simulation by the function $S(n_1...n_s)$. We will also assume that this size function corresponds to our intuitive notion of size, i.e. that the function is monotonically increasing for each n_i so that $S(n_1...n_j....n_s) < S(n_1...n_j+\varepsilon...n_s)$ for each value of j and positive value of ε. The simulation size defined here is essentially the value of the simulation's complexity as a function of the input parameters, and is not simply the size of the input to the simulation.

In our Dynamic LOS example, the simulation size might be given by $j(n^2T^{0.5} +T+n)$, where n is the number of tanks, T is the number of polygons in the terrain database, and j is some constant. Since simulation size embeds the size of the inputs to the simulation and the complexity of the simulation, it is important to recognize that there are several characterizations of complexity.

We may consider the complexity of the simulation problem, or the average-case or worst-case complexity of an algorithm chosen to solve it. We may consider complexity in terms of computational time, space, or both. Since we can often make tradeoff decisions between computational time and space, the most general formulation of simulation size should consider both. The appropriateness of using less general characterizations is discussed later in this paper; in any case, we must hold faithfully to the characterization of complexity chosen throughout our analysis.

A simulation may have quantifiable performance characteristics that are of interest but not directly related to a performance requirement. For example, we may want to minimize network message delays even in the absence of a specifically stated maximum delay requirement. Even if such a requirement exists, we may be interested in knowing how much the system surpassed the requirement. We will refer to any quantifiable performance characteristic whose value is of interest as a performance metric. Performance metrics may or may not relate specifically to a system requirement, but are distinguished from performance requirements by the fact that we are interested in their precise values, not just their acceptability. If the cognitive purpose of our simulation can be satisfied only by recomputing the lines of sight between every pair of tanks at least twice each second, we must do so in order to achieve minimum acceptable performance. We may still be interested in how often the lines of sight are computed when we evaluate the system.

$S(n_1...n_i....n_s)$ quantifies the simulation's measures of interest, performance requirements, and size; to completely quantify simulation capability it is necessary to consider performance metrics. The values of some performance metrics may only be permitted to vary within the ranges defined as acceptable by performance requirements, but the behavior of these performance metrics may still be of great interest. S embeds all the requirements placed on the simulation; performance metrics describe the amount by which the simulation exceeds these requirements.

For some metrics, a higher value may indicate better performance (e.g. display update rate), while some metrics may decrease as performance improves (e.g. message

latency). For convenience, we will assume that the value of every performance metric increases as performance improves. We can ensure this by using alternative characterizations of those metrics whose values decrease as performance improves (e.g. we can represent message latency by its reciprocal).

In our Dynamic LOS example, let A denote the average time between LOS computations over all tank pairs. We could use A as a performance metric, but system performance is better when A decreases. We can correct this to match our "higher is better" requirement for performance metrics by using (0.5–A) instead. As our performance improves, (0.5-A) increases. Similarly, we could use 1/A as a performance metric to capture the same information.

Let us denote the performance metrics of interest by $m_1...m_q$ and let $M(m_1...m_q)$ denote a performance metrics function that characterizes simulation performance in terms of these metrics. We will assume that M is monotonically increasing with respect to each performance metric so that an improvement in any performance metric improves the overall system performance if all other metrics remain constant.

3.2 Architectural capability

It is difficult to compare the capability of hardware systems in absolute terms. One cannot say that a system with a 200 MHz CPU clock speed and 16 MB of RAM is faster or slower than a system with a 100 MHz CPU clock speed and 32 MB for every application. Greenfeld points out that system scalability is a function of the application, since each application may stress different system resources (Greenfeld 1997).

If we were evaluating a well-defined application, we would quantify architectural capability with respect to that application. At the design stage of a simulation system, it is probable that the simulation is still loosely defined. This may force us to evaluate the architecture in terms of abstractions such as computing power, computing space, and communications speed.

Let us assume that we can somehow quantify architectural performance as a function of some set of p variables $h_1...h_p$ which quantify the relevant aspects of the system's capability with respect to the problem being solved. We will denote the architectural capability of the system by $P(h_1...h_p)$.

We will assume that the function P is monotonically non-decreasing for each h_i so that $P(h_1...h_i....h_p) \leq P(h_1...h_i+\varepsilon...h_p)$ for each value of j and non-negative value of ε. This monotonicity assures us that improving the performance of any single architectural element will not decrease the overall performance of the system.

3.3 Simulation Scalability

We have defined simulation capability as a function of measures of interest, performance requirements, simulation size, and performance metrics. The measures of interest, their required fidelity, and the performance requirements depend entirely on the simulation's cognitive purpose. If we fail to meet these requirements, our simulation's performance is unacceptable, so its scalability is irrelevant. The performance metrics function captures the amount by which the simulation's capability exceeds the minimum requirements placed upon it.

We have now reduced our characterization of simulation capability to a function of the simulation size $S(n_1...n_s)$ and the performance metrics function $M(m_1...m_q)$. Let $C((n_1...n_s), M(m_1...m_q))$ denote simulation capability. We will abbreviate $S(n_1...n_s)$ by S and $M(m_1...m_q)$ by M whenever possible for compactness. We will similarly abbreviate the architectural capability $P(h_1...h_p)$ by P whenever this is unambiguous.

We will assume that C is a monotonically increasing function of S and M; this corresponds to the intuitive notion that a simulation that simulates a larger problem with the same performance as another or simulates the same size problem with better performance than another is more capable. In our Dynamic LOS example, we might determine that simulating 100 tanks and updating every line of sight 4 times per second requires the same computational resources as simulating 200 tanks and updating every line of sight once per second. Our capability function should capture the equivalence of these two different situations.

With these simplified characterizations of simulation capability and architectural capability, we are now prepared to give a quantifiable definition of simulation scalability. It is clear that scalability is a relative measure, so we must first identify some benchmark system to measure against. Let us suppose that we have selected a benchmark architecture with architectural capability P_1. Suppose also that we have defined the domain of capabilities for a simulation problem; i.e. we have specified the set of simulation capabilities (sizes, resolutions, and fidelities) that are of potential interest to us.

Let us assume for simplicity that our simulation size S(N,T) is given by $n^2T^{0.5}$ where T is the total number of terrain polygons in our simulation space and N is the number of tanks. We will assume that our performance metrics function is $M(A_1) = A_1$, where $A_1 = 0.5/A$ and A is the average time required to update lines of sight between every pair of tanks. Suppose that we have determined that C(S,M) = SM for all values of S and M which are of interest.

Suppose that our architectural capability function has been determined to be $P(h1,h2) = (h_1/10)^2h_2$, where h_1 is

the CPU clock rate and h_2 is the amount of system RAM in MB. If we select as benchmark architecture a system with a 400 MHz Pentium II CPU and 64 MB of RAM, our benchmark architecture has capability $(40^2)64 = 102{,}400$. Let us further suppose that our benchmark architecture can execute our simulation and meet acceptable performance standards for any values of A, N, and T in our domain of interest so long as the simulation capability C does not exceed 102,400 (A=0.5, N=32, T=10,000 gives this value).

Let $k = C_1(S,M)/P_1$, where $C_1(S,M)$ is the largest value of C for which a specified system with architectural capability P_1 can compute the simulation's measures of interest and meet all the specified performance and correctness requirements for every element in the simulation capability domain with capability C. The system performance ratio k is the best ratio of simulation capability to architectural capability we can achieve with our benchmark architecture. In our contrived example, the simulation performance ratio is 1.0.

Let $C_i(S,M)$ represent a simulation capability i times greater than $C_1(S,M)$, and let P_i represent an architectural capability i times greater than P_1.

Definition 2

The *scalability of the system* is the largest real-valued j such that $C_i(S,M)/P_i \geq k$ for every value of i in the real-valued interval [1.0,j] where the described architecture with architectural capability P_i can compute the simulation's measures of interest and meet all the specified performance and correctness requirements for every element in the simulation capability domain with capability C_i. A system is *fully scalable* if it has an infinite scalability, i.e. if $C_i(S,M)/P_i \geq k$ for every value of i in the real-valued interval $[1.0,\infty)$.

Definition 2 defines scalability as the size of the interval in which the ratio between simulation capability and architectural capability remains at least as good as it was for the specified benchmark architecture. If we apply this definition to our example, j would tell us how much we could increase the capability of our architecture and still get at least 1.0 "units of simulation capability" per unit of architectural capability. Alternatively, we use these formulae to determine the cost effectiveness of our system in "units of simulation per unit of architecture" given the appropriate parameter values.

Given the definitions for C_i and P_i, it would seem natural to write $iC_1(S,M)/iP_1 \geq k$ rather than $C_i(S,M)/P_i \geq k$. Since $k = C_1(S,M)/P_1$, it is obvious that $k = iC_1(S,M)/iP_1$ for any nonzero value of i, leading to the obvious conclusion that $C_i(S,M)/P_i = k \geq k$ for every value of i in the real-valued interval $[1.0,\infty)$. This would imply that every simulation system is fully scalable by definition if it can perform acceptably for some architectural capability and simulation capability (to establish P_1 and C_1).

The critical point that must be remembered in the preceding discussion is that the system must exhibit acceptable performance, not merely achieve k as its ratio of simulation capability to architectural capability. The use of C_i and P_i instead of iC_1 and iP_1 is intended to prevent the obvious but erroneous conclusion that nearly all simulations are fully scalable under this definition.

It is certainly true that all simulation problems are fully scalable at the most abstract level under this definition. If we express the simulation capability function of the problem in terms of its worst-case space and time complexity, we could simply select that function as our architectural capability function as well. In the more real world of algorithms and architectures, full scalability is much more difficult to achieve.

3.4 Limitations of the Definition

A very general definition of simulation scalability that can be quantified as a function of two variables has been presented. Unfortunately, this general and quantifiable definition has several flaws that must be addressed.

A subtle issue in this definition is the largely hidden relationship between k and P_1. By selecting the benchmark architecture we selected P_1, and implicitly determined $C_1(S,M)$ and k. A system could be scalable over different ranges, depending on the value of k; this means that the same system could receive different values for scalability depending on the architectural capability of the selected benchmark system. Since simulation scalability is necessarily relative, this does not affect the utility of the definition, but the user must recognize this hidden dependency, and remain faithful to the benchmark capability and benchmark architecture when reporting scalability results.

A second problem in this definition is that the two variables that k directly depends on are not continuous; P in particular depends on values that may change in very discrete steps. System components are available only in certain sizes and quantities; we specify systems by choosing 16, 32, or 64 MB of RAM -- it is impractical, if not impossible, to obtain a system with 21.4 MB of RAM if we buy components or systems off the shelf. Similar discrete steps are imposed on us in the selection of secondary storage, CPUs, network connections, and every other aspect of architectural specification.

The simulation capability C is likely to be a discrete-valued function as well, since it depends on simulation size. It is possible (though not certain) that our performance metric function M is continuous, but it is very likely that our simulation size S takes on discrete values. Consequently, we cannot determine whether C is discrete-valued or continuous.

If we can analytically compute the maximum simulation capability that a given architecture can support,

the real-world problem that architectures and simulation capabilities may be constrained to discrete values vanishes. In this case, we can simply compute the scalability of the system analytically by determining the range for which $C_i/P_i \geq k$ holds with respect to any benchmark architecture. We must recognize that the system's scalability is applicable only to those discrete values that P and C can assume; two systems with differing scalabilities can be effectively equivalent if their scalability over the set of realizable capabilities is the same.

If we cannot compute the maximum simulation capability that a given architecture can support analytically, we must do so experimentally. In this case, we can estimate scalability by evaluating C_i/P_i at each point (C,P) in our experiment domain and comparing these values to the value of k (C_1/P_1) obtained from our benchmark case. One difficulty that arises from discrete-valued C and P functions is that C_i or P_i may not be a possible value for C or P. In this case, should we "scale" k up or down and evaluate at the nearest realizable (C,P) pair to determine scalability? It is unclear how well this or other methods might serve to approximate simulation scalability, even for simple purposes of comparison.

Discrete-valued C and P functions present another problem in computing scalability. To accurately determine scalability with respect to our discrete-valued C and P we would have to compute C_i/P_i at every point (C,P) that might yield acceptable performance, since C/P is not necessarily monotonic. This does not necessarily require us to compute at every (C,P) pair, since clever ordering of evaluation may allow us to skip certain pairs at which we can guarantee unacceptable performance. Even with carefully planned evaluations, evaluating at all necessary pairs in our sample space could be costly. The obvious solution to reducing this cost is to compute C_i/P_i for some sample collection of points. The difficulty added by discrete-valued C and P functions is that our scalability estimate will be more sensitive to our experiment's sample.

The most significant weakness in the general definition is that it is quantifiable only if one can quantify the variables in its domain: P and C. We have seen that both of these values are particularly difficult to quantify. It is difficult to construct a functional form for P that would correctly express the performance tradeoffs between CPU power, primary storage, secondary storage, operating system, and the many other features that influence architectural capability, even when we are interested only in architectural capability with respect to some single application. Constructing a functional form for C that accurately expresses the relationships between simulation problem sizes, resolutions, and fidelities is similarly difficult. Despite these apparent shortcomings, the definition presented can be fruitfully applied in some circumstances.

4 APPLICATIONS

The definition of scalability presented in the Section 3 has a wide range of applications at various stages in the life cycle of a simulation system. The simulation and architectural capabilities can be evaluated in terms of the space and time complexity of the problem to make decisions at the design stage of a new system. General statements about system requirements as a function of simulation capabilities can be supported by this high-level analysis. Analyzing the complexities of the underlying simulation problem can be invaluable in identifying the boundaries of an optimal implementation.

As system development continues, these capability functions can be made more specific and couched in terms of algorithms, instruction mixes, execution times, and performance requirements to make algorithm selections based on scalability over the range of anticipated uses as well as performance. Rigorous documentation of these scalability analysis efforts may be invaluable when unanticipated requirements changes occur, either to adapt to the new requirements or rapidly evaluate their effects and costs. As the simulation software is implemented, unit, component, and system tests can be conducted to confirm the predictions of the scalability analysis efforts and detect unanticipated obstacles to scalability.

4.1 Applying the General Definition

Despite its deficiencies, we can make use of the general definition of scalability to compare the scalability of two simulation programs A and B if those simulations have the same architectural capability domain and the same simulation capability domain.

Let us suppose that we can somehow establish an order of architectures, from least capable to most capable, in the architectural capability domain for each simulation. Our earlier assumption that $P(h_1...h_p)$ is monotonically non-decreasing with respect to each architectural component h_i greatly facilitates this ordering. Let us also suppose that the ordering of architectures is the same for each simulation, so that if simulation A has greater capability on architecture X than on architecture Y then simulation B will also perform better on architecture X. We can identify the architectural capability of the least-capable system as $P_1(h_1...h_p)$ and define it to be one.

Let us suppose also that we can order the elements in the simulation problem domain from least to greatest capability. The assumed monotonicity of the capability function with respect to the size and metric functions simplifies this ordering process. As in the case of the architectural capabilities, we assume that the ordering of simulation capabilities is the same for each simulation. We will refer to the least of the simulation capabilities as $C_1(S,M)$ and define it to be one.

When these orderings are complete, we can characterize simulation capabilities relative to the least-capable simulation in terms of architectural capability P_1 and can characterize architectural capabilities relative to the least-capable architecture in terms of simulation capability C_1. Given these two sets of relative capabilities, we can compute the scalability of each simulation system directly from the definition.

4.2 Restrictions of the Definition

The greatest deficiency in this quantifiable definition of simulation scalability is that it depends directly on variables that are themselves difficult to quantify. This shortcoming can be addressed by restricting the variables; instead of using architectural capability and simulation capability, we allow only some subset or single facet of each capability to vary. By simplifying simulation capability to simulation size or to a single performance metric and leaving all other elements of simulation capability fixed, we can quantify that capability much more easily. Similarly, if we vary only one component of architectural capability, we can more easily quantify that variable.

Let us assume that we have quantified system capability solely in terms of simulation size or some single performance metric; let us call this variable c. Let us also assume that we have characterized architectural capability in terms of some single component h. We can now compute k, the ratio of simulation capability to architectural capability, as we did in our general definition of scalability using some benchmark architecture.

Definition 3

The *scalability of a simulation with respect to (c,h)* is the largest real-valued j such that $C_i(c)/P_i(h) \geq k$ for every value of i in the real-valued interval [1.0,j] where the described architecture with architectural capability P_i can compute the simulation's measures of interest and meet all the specified performance and correctness requirements for every element in the simulation capability domain with capability C_i.

This simplification allows us to compute the simulation's scalability with respect to (c,h), where c is some facet of simulation capability and h is some architectural component. Speedup in parallel computation is exactly the restriction of architectural capability to processor count.

Many researchers have used this restricted definition of scalability with different (c,h) pairs to characterize simulation scalability. When we use scalability with respect to (c,h) to make decisions, it is important to recall that we are fixing many elements that are truly variable to facilitate evaluation. If we say that a simulation has scalability T with respect to (c,h) (and some benchmark architecture with capability P_1), we suggest that architectural component h or simulation component c fails to scale beyond T. It is possible that some other component of the architecture or simulation is the scalability bottleneck; we must recognize this possibility in evaluating our results.

Some researchers have held architectural capability constant and referred to the tradeoff between simulation size and some performance metric as simulation scalability. In essence, such studies are evaluating the shape of the simulation's capability function C(S,M) with respect to some single components of S and M. (Harless and Rogers 1995) mentions the tradeoff between spatial precision, temporal precision, and computational efficiency in simulations that model the interactions of continuously-moving entities. Similarly, (Pratt 1996) discusses the tradeoff relationship between execution time, fidelity, and resolution that occurs when architectural capability is held fixed.

We have indicated that some value may be obtained from studies that hold one of simulation size, performance metrics, or architectural capability constant and investigate tradeoffs between the other two. An interesting alternative approach could hold two of these variables constant and investigate the effects of varying the third quantity against computing various subsets of the measures of interest. It seems likely that such studies could help identify those measures of interest that impede scalability or otherwise impact simulation capability.

5 CONCLUSIONS

The scalability of a simulation has been defined mathematically as the size of the real-valued interval over which cost-effective improvements in the simulation's capability may be achieved. This definition of simulation scalability captures the intent of most scalability research since the most commonly used definitions of scalability are simple restrictions of this general definition. This definition lends itself well to objective comparisons of simulation scalability.

It is clear that the definition of simulation scalability is both general and quantifiable; it is more important that it is useful. The methodology required to evaluate scalability analytically promotes more extensible system design by distinguishing between system scalability and system performance. Evaluation of a simulation system's scalability can be conducted at varying levels of abstraction throughout the system life cycle to promote extensibility and support the software development process in areas ranging from requirements analysis to algorithm selection.

Recent studies (SAIC 1998) have confirmed that the definitions presented can be usefully applied in a number

of ways to obtain information about simulation system scalability at various stages in the simulation life cycle.

ACKNOWLEDGEMENTS

This paper is based on research (Law and Courtemanche 1997), (SAIC 1998) performed as part of contract N61339-97-C-0017 from the U.S. Army Simulation, Training, and Instrumentation Command. A significant contribution to the development of this definition was made by Anthony J. Courtemanche. Jenifer S. McCormack, Ph.D., and Joe D. Sullivan III made significant contributions in studies which validated the utility of the definitions presented herein.

REFERENCES

Blais, C., 1995. Scalability Issues in Enhancement of the MAGTF Tactical Warfare Simulation System. In *ELECSIM 1995*.

Cavitt, D., Overstreet, C., and Maly, K. 1996. A Performance Analysis Model for Distributed Simulations. In *Proceedings of the 1996 Winter Simulation Conference*, ed. J.Charnes, D. Morrice, D. Brunner, and J. Swain, 629-635. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Department of Defense Modeling and Simulation (M&S) Master Plan, October 1995. DoD reference #5000.59.

Fujimoto, R., 1995. Parallel and Distributed Simulation. In *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexopoulos, K. Kang, W. Lilegdon, and D. Goldsman, 118-125. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Greenfeld, N., 1997. System Scalability. In *Unix Review*, August 1997, pp. 9-12.

Harless, G., and Rogers, R., 1995. Achieving O(N) in Simulating the Billiards Problem in Discrete-Event Simulation. *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexopoulos, K. Kang, W. Lilegdon, and D. Goldsman, 751-756. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

IBM 1997. IBM Makes RS/6000 Even Faster. In *Company Press Release*, September 2, 1997.

Law, D., and Courtemanche, A., 1997. Domain Analysis Report: Simulation Scalability. Science Applications International Corporation Document SAIC-98/7764&00.

Pratt, D., 1996. Next Generation Computer Generated Forces. In *Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation*, pp. 3-8.

Quinn, M., 1994. Parallel Computing: Theory and Practice, 2nd Edition, McGraw-Hill, New York.

SAIC, 1998. Scalability Study Final Report. Science Applications International Corporation Document SAIC-99/7775&00.

Smith, J., Schuette, C., Russo, K., and Crepeau, D. 1996. Rational Characterization of the Performance of Distributed Synthetic Forces. In *Proceedings of the 14th DIS Workshop*, pp. 665-673.

Steinman, J., 1995. Scalable Parallel and Distributed Military Simulations Using the Speedes Framework. In *ELECSIM 1995.*

AUTHOR BIOGRAPHY

DARREN R. LAW is a Scientist with Science Applications International Corporation, a Link Fellow in Advanced Simulation and Training, and a Ph.D. candidate in computer science at the University of Central Florida.