Theoretical Nabladot Analysis of Amdahl's Law for Agent-Based Simulations

Claudio Cioffi-Revilla

Center for Social Complexity and Department of Computational Social Science,
George Mason University, Fairfax, Virginia 22314, USA
ccioffi@gmu.edu
http://krasnow.gmu.edu/socialcomplexity/faculty/csc-faculty-dr-cioffi/

Abstract. Amdahl's Law states that execution speedup S is nonlinearly proportional to the percentage of parallelizable code P and the number N of processors. Additional terms must be added to Amdahl's Law when applied to agent-based simulations, depending on how synchronization is implemented. Since P is continuous but N is discrete, traditional multivariate operators based on nabla or del ∇ are applicable only for P, not for N, regardless of synchronization architecture (linear, logarithmic, constant, among other). Moreover, relatively low values of N (bound by Miller's number 7 ± 2) are common in some cases. Here I apply a novel and exact operator, called "nabladot" and denoted by the symbol $\nabla\cdot$, that is defined for hybrid function such as Amdahl's Law. The main results show how exact solutions using nabladot differ from traditional approximations, particularly in the logarithmic case that is characteristic of hierarchical synchronization. Improvements in precision are inversely proportional to P and N, converging to 0.8 as $N \to 2$.

Keywords: Amdahl's Law, nabladot, multivariate vector analysis, distributed systems, concrete mathematics, hybrid functions.

1 Introduction

Agent-based simulations represent an emergent field of computational science, including the convergence of natural, social, and engineering sciences to advance our understanding of complex systems (Cioffi 2014a). This section provides basic motivation and background for the theoretical analysis that follows, focusing on Amdahl's Law for distributed systems and its application to a set of important cases in agent-based simulations.

1.1 Amdahl's Law

Consider a computer program operating as a distributed system over a number N of processors and let P denote the relative proportion of code that can be parallelized, where $N \geq 2$ and $0 \leq P < 1$. Accordingly, $1 - P$ is the proportion of code that remains serialized.

L. Lopes et al. (Eds.): Euro-Par 2014 Workshops, Part I, LNCS 8805, pp. 440–451, 2014.

The following question is of enduring interest in the theory of computing, especially with regard to time and space (or spatio-temporal) complexity: What is the speedup S resulting from different values of N and P? Amdahl's Law (1967) states that S, the amount of time that can be gained by parallelization, is given by the following formula:

$$S = \frac{1}{(1-P) + \frac{P}{N}} \tag{1}$$

$$= \frac{N}{N + P - NP} \,. \tag{2}$$

Several improvements to Amdahl's Law have advanced the field during the past decades (Eyerman and Eeckhout 2010; Gustafson 1988; Hill and Marty 2008), so formal analysis of laws of parallel and distributed systems remains significant in the theory of computation.

Equation 1 is the most common form of Amdahl's Law found in the literature, including most formal interpretations of Amdahl's statement in his original historic paper.[1] This equation for Amdahl's Law also provides a nice interpretation in terms of sequential and parallel components, corresponding to the denominator terms $(1-P)$ and P/N, respectively. By contrast, Eq. 2 is a simpler formula obtained through easily verifiable algebraic manipulation.

The advantage of Eq. 2 over Eq. 1 is that the latter is simpler and highlights the multiple dependencies of the mapping between speedup and P and N, or the multivariate function $f : (P, N) \to S$. With respect to P, S shows an inverse additive dependency ($S \propto 1/N$, based on the first term in the denominator) as well as an inverse multiplicative or inverse interactive dependency ($S \propto 1/NP$, the last term in the denominator). The same is also true with respect to N but, *in addition*, S has a linear dependency ($S \propto N$, based on the numerator), which is lacking with respect to P. Therefore, speedup has three different dependencies on N versus two for P—a feature that is missed when viewing Amdahl's Law simply as a function of two variables.

The retrospective implication of these multiple and concurrent dependencies is that Amdahl's Law is deceptively simple in its algebraic form. In fact, speedup is anything but intuitive as a multivariate function of P and N except in trivial cases, such as $P = N = 1$ or 0. Figure 1 shows the graph of Amdahl's Law for values of P and a range of low values of N. The general intuition that S should increase with both P and N is correct—as shown by the graph—but details matter greatly, given the nonlinear structure of Amdahl's Law.

Given these properties of Amdahl's Law, additional questions arise. Arguably, the following is fundamental: What is speedup more sensitive to, code parallelization P or processor parallelization N? Alternatively, viewed as an optimal allocation problem: in terms of speedup and given finite resources available, is it better to invest resources in P or in N? The complex functional dependencies

[1] Amdahl's 1967 paper contains the original statement of his law of speedup, but did not state a specific mathematical function.

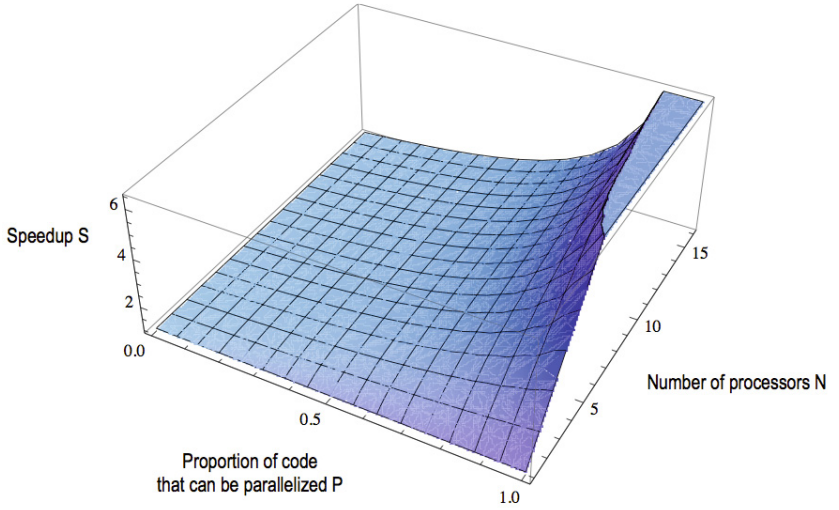

Fig. 1. Graph of Amdahl's Law for up to sixteen processors in the domain of N

of speedup on P and N prevent straightforward answers to these and related questions. This is a problem in multivariate calculus for functions of the form $g(x_1, x_2, x_3, \cdots)$, one that should be tractable with traditional methods from vector analysis for computing gradient fields, norm values, and the like (i.e., gradient, divergence, curl, and related functions).

Before attempting to answer this problem, however, Amdahl's Law needs to be understood at a more fundamental level as a function of two different *kinds* of variables with interesting properties. As a relative proportion, P is continuous over the [0,1.0) interval, assuming the program (or process) is sufficiently long that portions of it can be approximated as a continuous variable. By contrast, N is strictly discrete and assumes values on the interval $[1, n]$, where n is finite. Therefore, in terms of the optimal allocation problem just stated, traditional multivariate methods from vector analysis are not applicable, except as approximations for high values of N.

A hybrid function consisting of a mix of continuous and discrete variables—such as Amdahl's Law—may be called a "concrete function," paraphrasing Graham, Knuth, and Patashnik (1994). A combined differential-difference multivariate calculus operator for analyzing and better understanding such hybrid or concrete functions has not been available, until recently (Cioffi 2013, 2014).

Definition 1 (Hybrid concrete function). *Let* $\varphi(X_1, X_2, X_3, \cdots, X_N; Y_1, Y_2, Y_3, \cdots, Y_M) = \varphi(X_i, Y_j)$ *denote a multivariate function in* $N + M$ *variables. The function* $\varphi(\cdot)$ *is called a concrete function if and only if its* N

independent variables (X_i) are continuous with real values, and its M independent variables (Y_j) are discrete with integer values.

1.2 Amdahl's Law for Agent-Based Simulations

Amdahl's Law is highly relevant in the context of agent-based simulations, because such models have additional requirements for executing in "lockstep" to update the state of agents. Accordingly, Amdahl's Law can be modified by adding a third term in the denominator, a function $\psi(N)$, representing synchronization across N processors:

$$S = \frac{1}{(1-P) + \frac{P}{N} + \psi(N)}. \tag{3}$$

The following three cases are significant in agent-based simulation:

Case 1: Linear synchronization function Let $\psi(N) \propto N$. Then

$$S_\lambda = \frac{1}{(1-P) + \frac{P}{N} + \lambda N}, \tag{4}$$

where $\lambda > 0$ is a constant (scale parameter).

Case 2: Logarithmic synchronization function Let $\psi(N) \propto N$. When synchronization is hierarchical, then

$$S_{log} = \frac{1}{(1-P) + \frac{P}{N} + \log N}. \tag{5}$$

Case 3: Constant synchronization function When synchronization is insensitive to N, then

$$S_k = \frac{1}{(1-P) + \frac{P}{N} + k}, \tag{6}$$

where $k > 0$ is a constant.

The three cases are illustrated in Figures 2–4. Surprising, the linear case shows that speedup is not significant, in clear contrast with the basic Amdahl Law (Fig. 1). The logarithmic case (Fig. 3) shows gains in speedup, with a more complex surface that includes a saddle-point for null baseline of single processing ($N = 1$). Speedup remains significant for high parallelizable programs, but decreases with increasing values of N. The constant case shows no significant gains in speedup. Therefore, only case 2 (logarithmic synchronization) seems interesting.

Remark The equations in each case, as well as in subsequent extensions of Amdahl's law (e.g., Eyerman and Eeckhout 2010; Gustafson 1988; Hill and Marty 2008), retain the hybrid combination of discrete and continuous variables and parameters that is the defining feature of concrete equations.

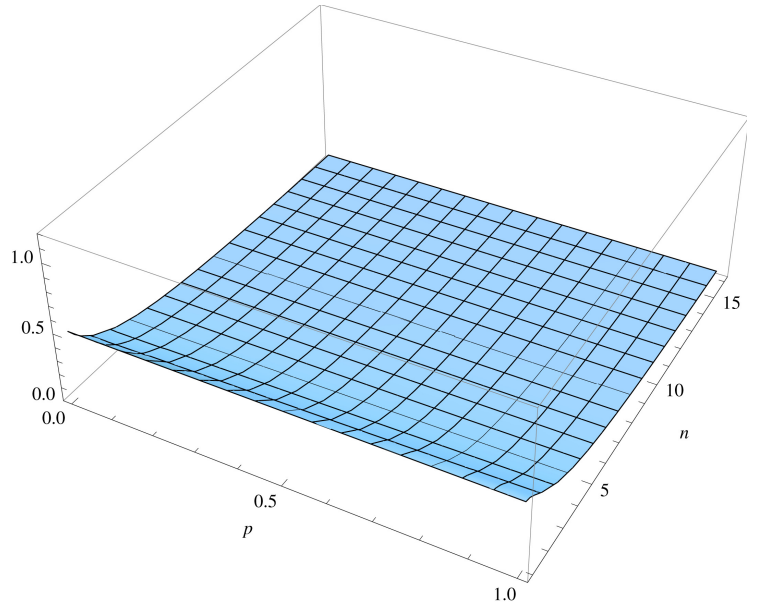

Fig. 2. Graph of Amdahl's Law for linear synchronization (Case 1)

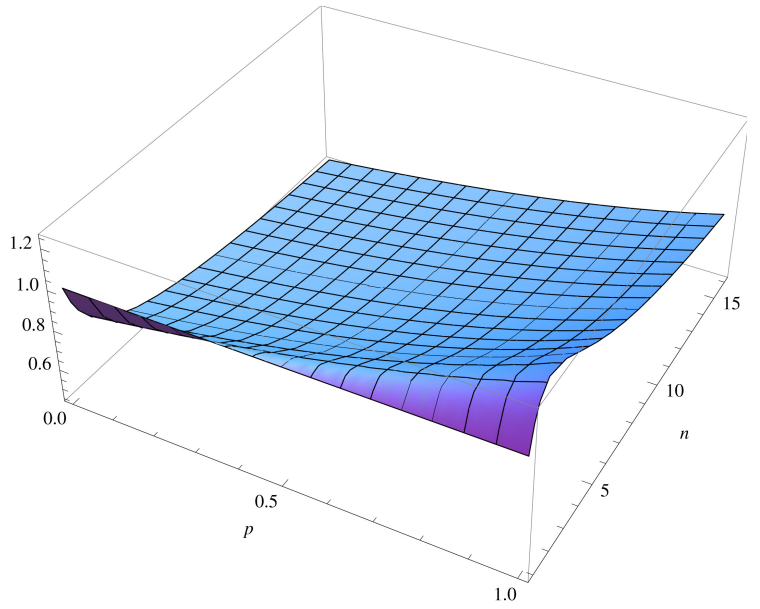

Fig. 3. Graph of Amdahl's Law for logarithmic synchronization (Case 2)

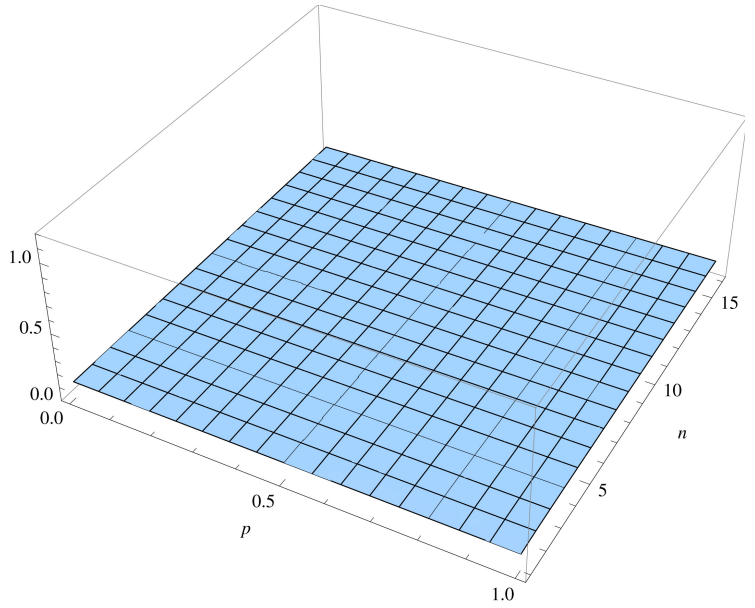

Fig. 4. Graph of Amdahl's Law for constant synchronization (Case 3)

2 The Nabladot Operator

The "nabladot" operator is strictly defined for a concrete function $\varphi(\cdot)$. Further, this operator is defined in both absolute and relative (i.e., normalized) terms, corresponding to unit-based and dimensionless scales, respectively.

2.1 Absolute Nabladot Operator ∇

Definition 2 (Nabladot; absolute nabladot operator ∇). *The nabladot operator ∇ is a vector associated with a concrete function φ and defined as*

$$\overrightarrow{\nabla}\varphi \equiv \frac{\partial\varphi}{\partial X_1}\hat{\mathbf{x}}_1 + \frac{\partial\varphi}{\partial X_2}\hat{\mathbf{x}}_2 + \cdots + \frac{\partial\varphi}{\partial X_N}\hat{\mathbf{x}}_N + \Delta_{y_1}\varphi\hat{\mathbf{y}}_1 + + \Delta_{y_2}\varphi\hat{\mathbf{y}}_2 + \cdots + \Delta_{y_M}\varphi\hat{\mathbf{y}}_M \tag{7}$$

$$= \sum_{i=1}^{N}\frac{\partial\varphi}{\partial X_i}\hat{\mathbf{x}}_i + \sum_{j=1}^{M}\Delta_{Y_j}\varphi\hat{\mathbf{y}}_j, \tag{8}$$

where $\partial/\partial X_i$ and Δ_{y_j} denote first-order derivatives and first-order differences w.r.t. X_i continuous variables and Y_j discrete variables, respectively; and $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{y}}_j$ are unit vectors codirectional in their respective dimensions.[2]

Note that, in general, $\nabla\varphi \neq \nabla\varphi$, as shown by analysis of Amdahl's Law in the next section.

2.2 Relative (Normalized) Nabladot Operator $\boldsymbol{\nabla^*}$

Definition 3 (Nabladot-star; normalized nabladot ∇^*). *The normalized nabladot operator ∇^*, also called nabladot-star, is defined for a concrete function φ in terms of normalized sensitivities (Definition 1), as*

$$\overrightarrow{\nabla}^*\varphi \equiv \sum_{i=1}^{N} \frac{\partial\varphi}{\partial X_i}\frac{X_i}{\varphi}\hat{\mathbf{x}}_i + \sum_{j=1}^{M} \Delta_{Y_j}\varphi\frac{Y_j}{\varphi}\hat{\mathbf{y}}_j, \tag{9}$$

where $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{y}}_j$ are unit vectors codirectional in their respective dimensions.

In turn, nabladot operators (absolute and relative, nabladot ∇ and nabladot-star ∇^*, respectively) can be used to define the corresponding operators (Laplacian $\nabla^2\varphi$, divergence $\nabla\cdot\varphi$, curl $\nabla\times\varphi$) analogous to those in the standard vector calculus for continuous multivariate fields. The advantage is that nabladot operators are well-defined for hybrid functions such as Amdahl's Law and, therefore, the resulting vectors and fields are exact, whereas traditional methods yield approximations with varying degrees of error.

3 Nabladot Analysis of Amdahl's Law

This section provides an analysis of the Amdahl Law equations 2 and 4–6, based on nabladot's exact solutions, rather than through traditional approximations that would treat the number of processors N as if it were a continuous variable. Given that all Amdahl Law equations are nonlinear multivariate functions of two independent variables, its first-order rates of change with respect to each variable is of basic interest. This is possible using the nabladot operator for hybrid functions such as Amdahl's.

3.1 General Parallelization

The objective is twofold, based on Def. 2 and Eqs. 8–9: (i) to obtain the first-order derivative of speedup S with respect to continuous variable P and, similarly, (ii)

[2] By convention (Grady & Polimeni 2010; Hamrick 2011), the first-order finite difference is defined as $\Delta_y\varphi = \Delta\varphi/\Delta y \equiv \varphi(y+1) - \varphi(y)$, which is the `DifferenceDelta` operator in Mathematica.

to obtain the first-order difference with respect to N (discrete). The first part yields

$$\frac{\partial S}{\partial P} = \frac{N(N-1)}{(N+P-NP)^2}, \tag{10}$$

while the second part yields

$$\Delta_N S = \frac{P}{(NP-N-1)(NP-P-N)}. \tag{11}$$

These two equations are exact results, not approximations. The gradient of speedup with respect to its two independent variables is obtained by simply substituting Eqs. 10 and 11 into continuous and discrete components of Eq. 8, respectively, we obtain the following new expression for the vector field of $S = f(P, N)$:

$$\nabla S = \frac{\partial S}{\partial P}\hat{\mathbf{x}}_p + \Delta_N S\hat{\mathbf{y}}_n \tag{12}$$

$$= \frac{N(N-1)}{(N+P-NP)^2}\hat{\mathbf{x}}_p + \frac{P}{(NP-N-1)(NP-P-N)}\hat{\mathbf{y}}_n. \tag{13}$$

Equation 13 brings us a significant step closer to answering the question concerning which variable has the greater effect on speedup. Finally, nabladot-star provides the normalized gradient field. This is obtained by applying Eq. 9 containing dimensionless elasticities.

By contrast, the continuous approximation used by traditional methods that ignore the discreteness of N would have resulted in the following (erroneous) vector component in Eq. 11:

$$\frac{\partial S}{\partial N} = \frac{N(N-1)}{(N+P-NP)^2}, \tag{14}$$

rather than the correct component.

How does the exact solution given by Eq. 13 compare with the approximation in Eq. 14? This is shown in Figure 5. The graph on the left was drawn by the traditional approach using Eq. 14 (ignoring discreteness), while the graph on the right was drawn by the exact method provided by Eq. 11 based on the new nabladot operator (respecting discreteness). The visual similarity between the two gradient fields seems unquestionably close, but appears misleading once greater precision is used to compare the two.

Let ϵ denote the difference between the two functions plotted in Fig. 5. Arguably the simplest viable way to specify ϵ is by subtracting one graph from the other, as in a difference map. This is shown in Fig. 6, which shows the result of subtracting the difference from the derivative of Amdahl's Law in both cases with respect to N to see if they are in fact the same.

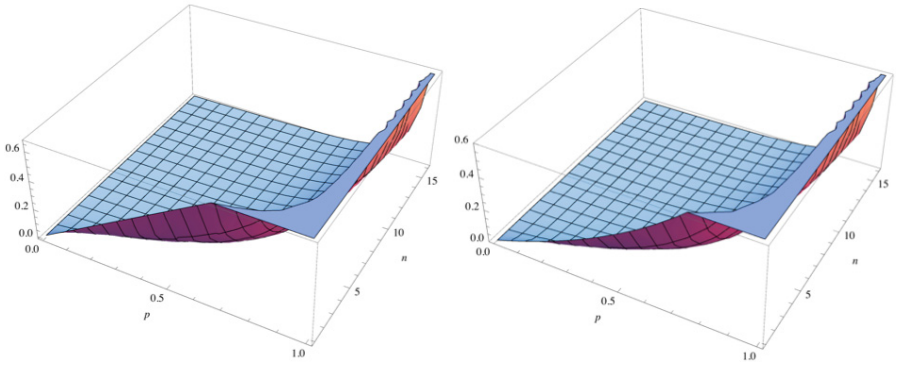

Fig. 5. Multivariate analysis of Amdahl's Law. First-order derivative function (left - incorrect approximate solution) and first-order difference function (right - correct exact solution) shown practically identical results. However, see next figure.

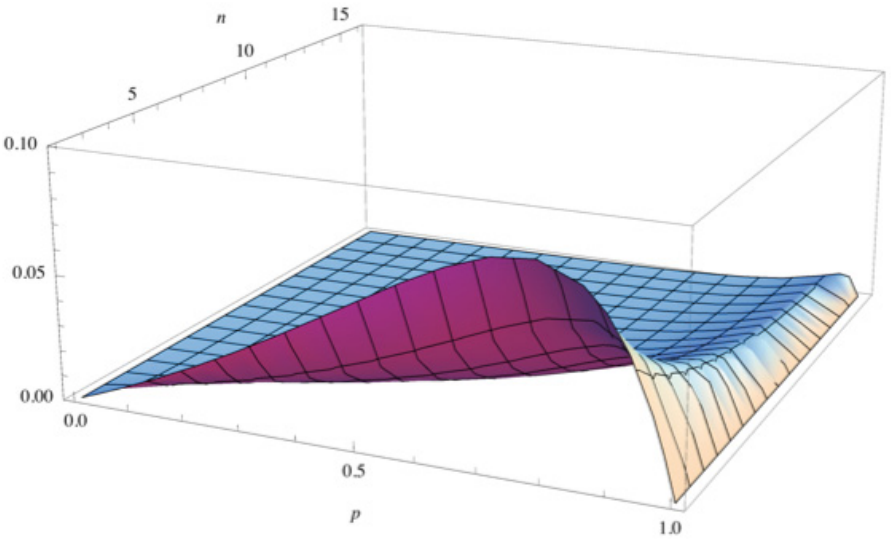

Fig. 6. Difference graph of the first-order derivative function minus the first-order difference function in Fig. 5

Clearly the two graphs in Fig. 5 are not the same, although they are similar. Several features observed in Fig. 6 merit highlighting:

1. The error surface $\epsilon(P, N)$ is not flat (which would indicate a low or constant error of the derivative approximation Eq. 14). Instead, the graph shows significant curvature.
2. The size of the largest error is approximately 7 percent.

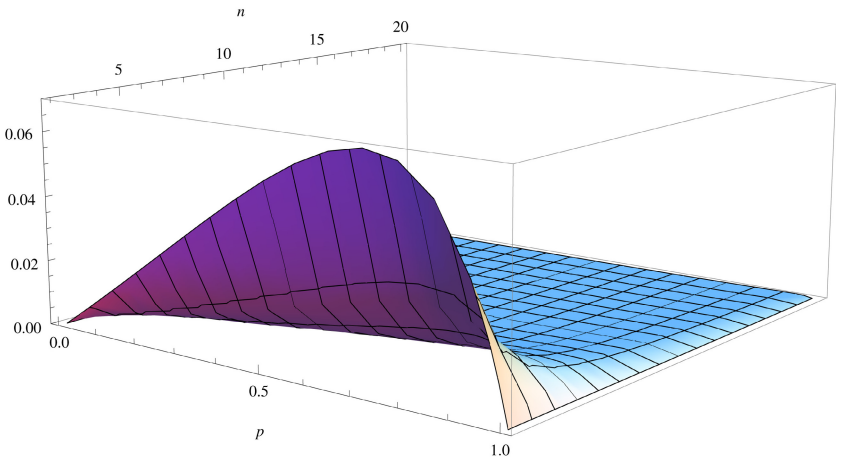

Fig. 7. Difference error graph of the standardized nabla-norm $\| \nabla^* S \|$ minus the standardized nabladot-norm $\| \nabla^* S \|$ of Amdahl's Law.

3. This occurs for $N = 2$, which is clearly where the error is maximal.
4. The error decreases as N increases, but in a non-uniform way.
5. Error increases with increasing values of P, as can be seen by the ridge extending toward the distance as N increases.

Another way to assess the difference between nabla and nabladot gradients of Amdahl's Law is to compare their respective standardized norms:

$$\| \nabla^* S \| = \sqrt{\left(\frac{\partial S}{\partial P}\frac{P}{S}\right)^2 + \left(\frac{\partial S}{\partial N}\frac{N}{S}\right)^2} = \sqrt{\frac{P^2(2-2N+N^2)}{(N+P-NP)^2}} \tag{15}$$

$$\| \nabla^* S \| = \sqrt{\left(\frac{\partial S}{\partial P}\frac{P}{S}\right)^2 + \left(\Delta_N S\frac{N}{S}\right)^2} \tag{16}$$

$$= \sqrt{P^2\left(\frac{1}{(1+N-NP)^2} + \frac{(N-1)^2}{N+P-NP}\right)} \tag{17}$$

The difference (error) function between these two functions (Eqs. 15 and 17) is shown in Figure 6. The topology of this function is the same as the previous error function for gradients, but the geometry differs in terms of the smaller scale of the range (< 0.07) and less pronounced ridge for high values of parallelization P.

3.2 Parallelization in Agent-Based Simulations

The case of Amdahl's Law under logarithmic synchronization (Eq. 5, Fig. 3) is interesting, as was shown earlier (Sec. 1.2). By contrast, linear and constant

cases showed significantly less difference between the two operators. The same result obtains for the difference (error) function between the two norms of the synchronization case, as shown in Figure 3.

4 Discussion and Conclusions

This analysis proposed a new perspective on Amdahl's Law as a hybrid function of continuous and discrete variables. This requires applying mathematically appropriate methods for deepening scientific understanding of distributed systems of computation. The nabladot operator is an improved gradient field operator for understanding hybrid functions commonly found in complex systems. This is because it is based on differential and difference components appropriate for each kind of variable, unlike traditional methods that have ignored discreteness and simply assumed continuity.

In terms of the specific findings of this study, the nabladot analysis of Amdahl's Law revealed a previously unnoticed error in calculating the elasticity and related gradient fields of speedup with respect to parallelization and number of processors. Whereas traditional methods based on continuous operators (nabla) provide approximate results, the newly proposed hybrid operator for concrete functions such as Amdahl's Law provides exact solutions. The distribution of error with respect to the domain of speedup (i.e., the space of $P \times N$) is not uniform, which is explained by the nonlinearities involved in Amdahl's Law and others like it. This is true for the original Amdahl's Law as well as for the logarithmic variant applicable to parallelized ABMs with hierarchical synchronization.

All cases have the same topology, but their geometry differs, depending on each case. The main differences in geometry occur with Miller's number 7 ± 2. For agent-based simulations, the logarithmic case associated with hierarchical synchronization seems most interesting in terms of showing significant difference between approximate nabla-based results and exact nabladot-based results, while results are similar for linear and constant cases.

Error distributions between nabla-based and nabladot-based results (Figs. 6 and 7) seem confined to systems with low number of processors and weakens further with increased potential for parallelization.

All in all, this analysis has brought to light aspects of Amdahl's Law that seem intriguing and potentially worth further exploration. Whereas the mathematical nature of the analysis highlights theoretical features, it may also be the case that computer scientists and engineers may uncover practical aspects through experimental analysis.

With respect to broader issues related to Amdahl's Law, one of the most salient is the fact that computing is only one instance among many others classes of distributed systems. Bureaucracies, investigations, negotiations, and infrastructure systems, are among other systems where the architecture of complexity hinges critically on parallel or distributed systems. Thus, Amdahl's Law and its more recent sucessors—and concepts, features, or principles such as those analyzed here—holds much promise for advancing our understanding of complex systems.

Acknowledgments. Presented at the 2nd Workshop on Parallel and Distributed Agent-Based Simulations (PADABS 2014), Porto, Portugal, August 25, 2014. Thanks to six anonymous referees and workshop participants for helpful comments and discussions, especially Sean Luke, Mark Coletti, and Vittorio Scarano for discussions on Amdahl's Law and distributed systems, and to Keith Sullivan for recommending readings. Funding for this study is provided by CDI grant no. 1125171 from the US National Science Foundation, MURI grant no. N000140810921 from the Office of Naval Research, and by the Center for Social Complexity at George Mason University.

References

Amdahl, G.: Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. AFIPS Conference Proceedings 30, 483–485 (1967)

Cioffi-Revilla, C.: The Nabladot Operator for Hybrid Concrete Functions in Complex Systems. Presented at the Monday Seminar, Krasnow Institute for Advanced Study, George Mason University, Fairfax (2013)

Cioffi-Revilla, C.: Introduction to Computational Social Science: Principles and Applications. Heidelberg, Springer (2014a)

Cioffi-Revilla, C.: The Nabladot Operator for Hybrid Concrete Functions in Complex Systems. In: Proceedings of the Second World Conference on Complex Systems (WCCS 2014), Agadir, Morocco (2014b)

Eyerman, S., Eeckhout, L.: Modeling Critical Sections in Amdahls Law and its Implications for Multicore Design. In: ISCA 2010, Saint-Malo, France, June 19–23 (2010)

Grady, L.J., Polimeni, J.R.: Discrete Calculus: Applied Analysis on Graphs for Computational Science. Heidelberg, Springer (2010)

Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics: A Foundation for Computer Science, 2nd edn. Addison-Wesley, Reading (1994)

Gustafson, J.L.: Reevaluating Amdahl's Law. Comm. ACM, 532–533 (May 1988)

Hill, M.D., Marty, M.R.: Amdahl's Law in the Multicore Era, pp. 33–38. Computer - IEEE Computer Society (July 2008)