

Survey of scalability of computational systems

David Keil

Framingham State College dkeil@framingham.edu

Abstract. Notions of scalability have helped in evaluation of the performance of algorithms and of parallel and distributed systems, including multi-agent systems. *Algorithm analysis* and *computational complexity* address primarily the *scalability* of solutions, where we may intuitively associate the scalability of a solution with its “reasonable” performance regardless of problem size. Performance and scalability of concurrent systems and models is a research area lacking a standard authoritative body of literature.

Some algorithms are favored precisely because they scale better than others, and problems are called *intractable* precisely because their best solutions don’t scale well. $O(n)$ solutions scale better than $O(n^2)$ ones; problems in class P (polynomial-time) have solutions that scale acceptably, whereas NP-hard problems have no known solutions that scale acceptably.

The objective of previous work on scalability of parallel systems was to use scalability analysis to select algorithm-architecture combinations, to predict performance of systems or algorithms, to determine optimal number of processors for a task, or to predict the effect of changes in hardware technology on performance [2]. Previous work on scalability in distributed systems and MASs has been based on the work on scalability in parallel systems.

Notions of scalability to date have been based on the assumption that the complexity of *algorithm execution* is the object of study.

Scalability notions are also related to *efficiency*, which may be defined as speedup divided by the number of processors, so that ideal efficiency is 1.0. *Speedup* for an algorithm running on a system is serial time (running time on one processor) divided by parallel time for the algorithm.

Research has identified a metric to determine scalability, *isoefficiency*. The isoefficiency of an algorithm executing on an architecture is the ratio of problem size to the minimum number of processors p necessary to obtain an increase in speedup proportional to p . For cost-optimal algorithms, fixed running time for arbitrary problem size is obtainable if and only if the isoefficiency of the algorithm is $\Theta(p)$ [4]. A system may be described as *scalable* if an isoefficiency function exists; i.e., when efficiency can be maintained as p rises if problem size W rises sufficiently.

Communication overhead among processors may prevent linear speedup in parallel systems, hence isoefficiency, hence scalability. In particular, *global communication*, which includes broadcast, one-to-all, and all-to-all personalized communication, is an obstacle. On parallel machines with message-passing latency,

it is not possible to solve arbitrarily large instances of problems in constant time if the solutions involve global communication [4].

In distributed systems, the factor of communication overhead is all the more important than in parallel systems. Here it is clearest that performance of concurrent systems is not only *algorithm* related but *interaction* related. In fact, what most affects the performance of parallel and distributed computing is the role of interaction *within* the systems. Thus, because communication may become a paramount issue in distributed computing, research on scalability of these systems must make a departure from the conceptual foundations in research on scalability of parallel systems. Moreover, communication and CPU resources vary widely [1, 3].

As early as 1990, the sharp distinction between performance analysis of *algorithms* and of *interactive systems* was recognized in the difference between the notions of a *protocol* in communication (rules specifying order and semantics of messages) and *algorithms* in traditional complexity theory [6]. Thus the theory of the performance of parallel and distributed *algorithms* must depart somewhat from the familiar terrain of what is known as computational complexity. In effect, *communication* has become a basic class of resource for computation, alongside time, processor cycles, and storage.

MASs are distinguished from distributed computing systems in general in that MASs are composed of *autonomous* agent entities [5]. Thus MASs, as sets of software entities executing on distributed but communicating hardware, are a step closer than parallel and distributed computing systems to our more general notion of multi-stream interaction. As late as 1998, “nonfunctional properties” of MASs, such as performance, stability, and scalability, had been given “scant attention” in research [5].

The literature on scalability in multi-agent systems goes beyond previous work on scalability of distributed systems by calling attention to *simultaneity* and *synchrony/asynchrony* in systems. *Simultaneous* actions by agents are actions that conceptually occur at the same time; they may be executed physically on a single processor if it is hosting the MAS [10].

Synchronous agent platforms support simultaneous actions by causing all “messages” to be exchanged at a (clocklike) “tick” when no further processing can occur without communication. Asynchronous platforms support communication as needed by individual pairs of agents. Most MASs require this [9].

Some previous work on scalability of MASs can be summarized by enumerating some informal definitions of scalability found in the literature. Scalability of a system means that its capacity to do work rises with the number of agents in the system [5]. Scalable systems work effectively without respect to size [7].

MAS scalability is associated with the system’s ability to reach a minimum degree of utility *to users*, as population sizes rise [9]. Another definition identifies scalability with supporting more agents without users incurring significant service delays [8]. A quantitative definition of scalability is average performance degradation of agents as their environmental load rises due to expansion of the population [5].

In much of the literature on MASs, including on MAS scalability, an “agent” is a software agent hosted on a server. Scalability issues are discussed from that perspective. Examples are a trader system [9] and a system that matches jobs with job seekers [8].

References

1. A. Helsinger, R. Lazarus, W. Wright, and J. A. Zinky. Tools and techniques for performance measurement of large distributed multiagent systems. *AAMAS 2003*, pages 843–850.
2. L. Hu and I. Gorton. Performance Evaluation for Parallel Systems: A Survey. *Tech. Rept. UNSW-CSE-TR-9707, University of NSW, Sydney, Australia*, 1997.
3. P. Jogalekar and M. Woodside. Evaluating the Scalability of Distributed Systems. *IEEE Trans. on Parallel and Distributed Systems*, 11(6):589–603, June 2000.
4. V. Kumar and A. Gupta. Analyzing scalability of parallel algorithms and architectures. *Journal of Parallel and Distributed Computing*, 22(3):379–391, 1994.
5. L. Lee, H. Nwana, D.T.Ndumu, and P. D. Wilde. The stability, scalability and performance of multi-agent systems. *BT Technol J*, 16(3), 1998.
6. L. Lovasz. Communications complexity: A survey. In B. Korte, editor, *Paths, Flows, and VLSI layout*, Springer, 1990.
7. P. Marrow. Scalability in Multi-Agent Systems: The DIET Project. *Agents 01 Workshop on Infrastructure and Scalability for Agents*, 2001.
8. R. Song and L. Korba. The Scalability of a Multi-Agent System in Security Services. *NRC/ERB-1098*, August 2002.
9. P. J. Turner and N. R. Jennings. Improving the scalability of multi-agent systems. In *Proc. 1st Intl. Wkshp. on Infrastructure for Scalable MASs, LNAI 1887*, 2000.
10. D. Weyns and T. Holvoet. A Model for Simultaneous Actions in Situated Multi-Agent Systems. *1st Intl. German Conference on Multi-Agent System Technologies, LNCS 2831*, 2003.