01 Jan 2004

A Taxonomy of Network Computing Systems

Muthucumaru Maheswaran

Shoukat Ali

Recommended Citation

M. Maheswaran and S. Ali, "A Taxonomy of Network Computing Systems," *Computer*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2004.
The definitive version is available at https://doi.org/10.1109/MC.2004.148

A Taxonomy of Network Computing Systems

Muthucumaru Maheswaran, McGill University
Shoukat Ali, University of Missouri-Rolla

A proposed taxonomy encompasses all NC systems yet is widely applicable.

Rapid advances in networking and microprocessor technologies have led to the emergence of Internet-wide distributed computing systems ranging from simple LAN-based clusters to planetary-scale networks. As these network computing systems evolve by combining the best features of existing systems, differences among NCs are blurring. However, a clear differentiation is essential to assess new contributions.

Currently, NC systems are grouped in an ad hoc fashion—for example, as single-site clusters, multisite clusters, peer-to-peer networks, generalized P2P overlays, grids, or computing utilities. However, a given system can fall into more than one category, while another may not fully fit into any. For example, a geographically distributed NC system can be a multisite cluster or grid. Likewise, one NC system can be classified as P2P if its nodes enjoy high levels of autonomy, while another can be classified as a multisite cluster due to the participating nodes' overall distribution.

To address this problem, researchers have proposed formal taxonomies of NC systems. However, most such groupings focus on different aspects of the systems, such as their approach to resource management, or specialized forms of NC systems—for example, heterogeneous computing systems. We propose a new taxonomy that is both broad enough to encompass all NC systems and simple enough to be widely used.

TOWARD A NEW TAXONOMY

Following the work of T.D. Braun and colleagues ("A Taxonomy for Describing Matching and Scheduling Heuristics for Mixed-Machine Heterogeneous Computing Systems," *Proc. 17th IEEE Symp. Reliable Distributed Systems,* IEEE CS Press, 1998, pp. 330-335), we have identified common characteristics of NC systems with respect to applications, platforms, and management.

Applications

NC systems can be characterized using application-based parameters such as the number of applications the system supports and its communication and performance models.

Some NC systems are designed for a fixed number of applications, while others support multiple applications. P2P systems such as Gnutella (www.gnutella.com) and Kazaa (www.kazaa.com) are designed solely for file swapping.

The communication model determines the communication primitives an NC system supports. Multimedia-intensive systems such as Access Grid (www.accessgrid.org) rely on multicasting-based models to improve efficiency, whereas less communication-intensive systems use easier-to-deploy models. For example, high-performance distributed computing systems can use simple broadcasting-based models to scatter and gather computations within the system.

The performance model determines application performance aspects designed into the NC system. Volunteer systems such as SETI@home (http://setiathome.ssl.berkeley.edu) are designed with the fewest application-based performance requirements, while Access Grid and other QoS-sensitive systems have stringent requirements.

The number of applications is the most important of these parameters because it influences many other factors including the communication and performance models. For example, resource management is simpler with a single application due to the lack of interapplication competition.

Platforms

NC systems can also be characterized using different platform-based parameters such as the number of owners/machines, resource hetero-

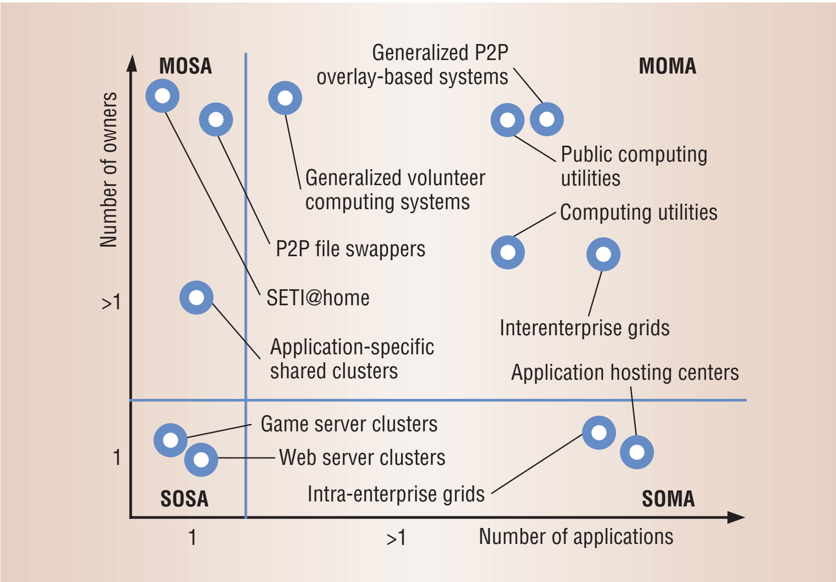

Figure 1. Mapping of sample NC systems: single-owner, single application (SOSA); single-owner, multiple-application (SOMA); multiple-owner, single-application (MOSA); and multiple-owner, multiple-application (MOMA).

geneities, and trust among resources.

NC systems can consist of resources belonging to a single owner or to multiple owners. Some systems, such as P2P file-sharing systems, can have millions of owners.

In addition, different system architectures (single instruction, multiple data versus multiple instruction, multiple data), instruction set architectures (x86, Sparc, Power PC, and so on), and software configurations (operating system, compiler, and so on) cause resource heterogeneities in NC systems.

Trust is another key platform-based issue for NC systems that indicates the level of cooperation among resources.

The number of owners is a key factor because it largely determines the impact of other parameters. For example, with single ownership, it is possible to effectively manage trust across the system as well as control factors such as resource heterogeneity due to software and policy characteristics.

Management

NC system management can be characterized using various parameters including organization, level of site autonomy, and management policies. However, management largely depends on the number of applications and owners. Site autonomy may not be an issue in single-owner NC systems, while resource heterogeneities can be significantly reduced. We therefore opted to exclude management-based parameters from our taxonomy.

FOUR-WAY CLASSIFICATION

Using the number of applications and number of owners as the grouping criteria, we developed the four-way classification of NC systems shown in Figure 1.

The single-owner, single-application class encompasses the simplest NC systems. Examples of SOSA systems include game and Web server clusters. Trust relationships among the resources are known, management objectives are well defined and less conflicting, and management systems do not handle site autonomies.

The SOMA class denotes NC systems with a single owner that run multiple applications in space or time. Examples of such systems include application hosting centers and intra-enterprise grids. SOMA systems are typically more complex than SOSA systems with respect to application performance and access.

P2P file-sharing systems, application-specific shared clusters, and SETI@home are examples of MOSA systems, on which multiple owners run a single application. These systems add another layer of complexity by introducing various trust, commitment, and QoS requirements. Although MOSA systems are widely deployed, several aspects of these systems are still under development.

The most general and complex class of NC systems involves multiple owners running multiple applications in space or time. Examples of MOMA systems include interenterprise grid computing systems, public computing utilities, volunteer computing systems, and P2P overlay-based systems. Ideally, MOMA systems should implement a generalized hosting facility atop a variable resource base, but in practice they can restrict variability in resource ownership composition.

Figure 2 shows how the four different classes of NC systems relate to one another. It also illustrates how introducing new levels of complexity can transform one type of NC system into another—in this case, a SOSA system into a MOMA system.

SYSTEM PARAMETERS

Unlike current ad hoc groupings of NC systems, the four-way classification shown in Figure 1 is

- disjoint—a given system cannot fit into more than one class; and
- complete—a given system should fit into at least one of the classes.

However, our proposed taxonomy goes beyond merely organizing NC systems by the number of owners and applications. The classes also differ with respect to three important system parameters: performance, trust, and flexibility.

Performance

A key objective of SOSA systems is to maximize the performance delivered to an application. A single application can have multiple components, such as a Web-serving application that receives numerous requests from different users. For this class of systems, performance can be measured using application-centric parameters such as throughput or average response times, or system-centric parameters such as utilization.

SOMA systems add fairness, utility, and QoS to SOSA system metrics. These parameters reflect the desire to prevent a few applications from monopolizing resources.

MOSA systems add participation, autonomy, and trust to the metrics that SOSA systems use. Participation can be measured by the total amount of useful work the different resources do on behalf of the system. A resource with strong participation will contribute more toward overall performance than a resource with weak participation. Mandating that each owner supply a predefined level of service can simplify participation.

MOMA systems are the most general and embrace the other three classes' performance considerations.

Trust

SOSA systems have few or no dynamic trust considerations because resource ownership is fixed and the application is known. However, the combination of multiple owners and applications make trust definition and monitoring difficult in MOMA systems.

In SOMA systems, trust issues can arise between the resource provider and the different clients launching the applications. Accountability is one way to deal with these concerns. Static trust requirements can limit the number of owners in MOSA systems. Accommodating a large number of owners requires dynamic trust modeling.

Flexibility

Single-owner resource platforms are easy to deploy but likely to provide only a local-area presence. Deploying resource platforms with multiple owners is more difficult, but they are likely to provide a wide-area presence. MOMA systems provide the most flexibility for multiplexing resources among different applications.

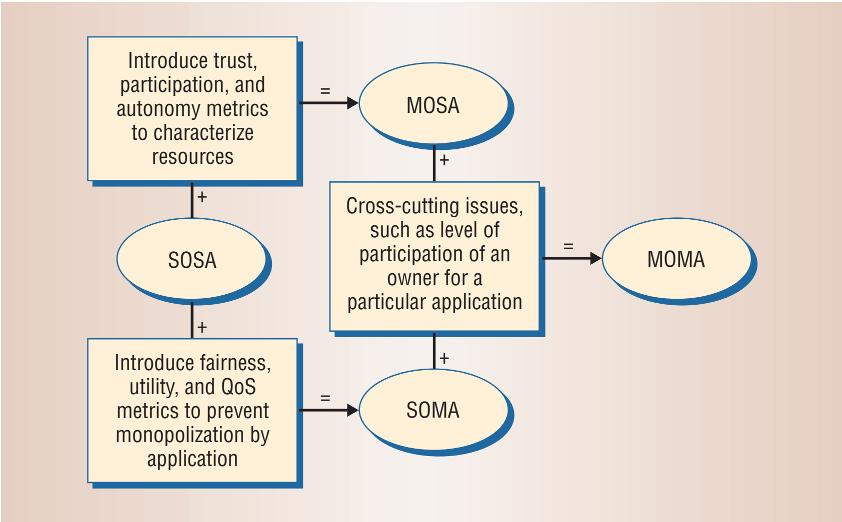

Figure 2. Relationship among NC system classes. Introducing new levels of complexity can transform one type of system into another—in this case, a SOSA to a MOMA system.

Our proposed taxonomy of NC systems currently uses only two criteria—number of users and applications—for simplicity and broad applicability. However, some situations may warrant a finer classification. One way to extend the proposed taxonomy is to include criteria based on management systems and users. ■

***Muthucumaru Maheswaran** is an assistant professor in the School of Computer Science at McGill University. Contact him at maheswar@cs.mcgill.ca.*

***Shoukat Ali** is an assistant professor in the Department of Electrical and Computer Engineering at the University of Missouri-Rolla. Contact him at shoukat@umr.edu.*

Editor: Upkar Varshney, Department of CIS, Georgia State University, Atlanta; uvarshney@gsu.edu