Cluster Computers and Grid Automata

M. Burgin
Department of Mathematics
University of California, Los Angeles
Los Angeles, CA 90095

Abstract
A new model for computers and networks is developed and studied. The model has the two-folded form: a grid array, which gives a direct representation for different kinds of computers and networks and its theoretical counterpart, a grid automaton. Grid arrays are aimed at computer/network design, description, verification, adaptation, maintenance, and reusability. Grid automata are aimed at the development of a theoretical technique for computer/network studies. Different properties of grid arrays and automata are obtained, demonstrating their usefulness for computers and their applications.

Key words: network, grid, grid array, grid automaton, grid computation

1 INTRODUCTION
There are many models and types of computers. They reflect different aspects of computers, as well as various computer architectures. Classical Turing machine with one tape and one head embodies the centralized computing architecture that realizes sequential paradigm of computation. This is a model for the classical von Neumann computer architecture that was suggested and implemented on the first stage of the computer development.

A complementary for centralized computing is distributed computing architecture. It has two types. Turing machine with many heads gives an example of the controlled distributed computing architecture that realizes parallel paradigm of computation. This is a model for the multiprocessor computer architecture. This has been the second stage of the development of computer architecture.

Neural networks, Petri nets, and cellular automata give examples of the autonomous distributed computing architecture that realizes concurrent paradigm of computation.

A new approach in information technology recently appeared. It is called cluster computation. In it computers are combined in clusters that work as a single system, which is called a cluster computer. The demands of the high performance computing market cause an outward expansion from localized clusters of computers to clusters of clusters shared between different departments and even between remote sites within the organization. But clusters are relatively limited systems. They tend to have homogenous hardware and software platforms, existing at a single site and having one specific function that they were built to accomplish [7].

A more advanced level of this approach is grid computation, which, on the other hand, is a natural evolution of the Internet. Grid computing is realized by grid information processing systems or grid arrays. Such system is the organization of computers into interlinked hierarchical networks that can be tapped for processing and communication power [3, 4, 15]. According to California Institute for Telecommunications and Information Technology director Smarr, grid computing has even more growth potential than the Internet explosion of the 1990s. He expects that grid computing will provide an infrastructure to support the entire economy and foretells a future of interconnected grids of all sizes, running the gamut from supercomputer clusters to mid-sized nodes of desktops to PC-based "micronodes." Grid computing projects are initially being established to provide data-crunching power for scientific and academic research, but Foster of Argonne National Laboratory believes that they will also serve as testbeds for commercial applications [15]. At this time, grid computing is still a work in progress.

However, as writes Lumb [7], grid computing is not a matter of some distant future. It is presently helping many organizations dynamically integrate their disparate, heterogeneous compute resources. Organizations who do not want to wait to reap the benefits of the global Grid have already built their own enterprise grids and partner grids.

Both approaches of cluster computers and grid computing are synthesized in the concept of grid array. Grid arrays embody the autonomous distributed computing architecture and realize the concurrent paradigm of computation in the most complete form. Grid automaton is its theoretical model. This model allows one to study properties of grid arrays related to design, maintenance, and utilization.

While the classical scheme of computing emphasizes transformation of information, grid computing shifts the emphasis, making communication and interaction as important in computation as it is information processing.

Interactions between computers, as well as interactions between users and computers have been studied for a definite period of time. Now we have such areas in computer science as human-computer interaction (cf. [9]) and interactive computations (cf. [6, 8, 10]). The advent of INTERNET intensified research in this direction. Grid automata provide a relevant context for these studies.

2 GRID COMPUTATION AND GRID ARRAYS
To achieve a adequate, efficient, and flexible representation of modern information processing systems, we introduce the following concept.
**Definition 1.** A grid array is a system of information processing systems (computers, networks, imbedded systems, etc.), which are situated in a grid, called nodes, are optionally connected and interact with one another.

The concept of grid array is very general, allowing one to include into the grid people and even non-living (natural and artificial) systems when they are considered as information processing systems. One grid array can be a node of another grid array. However, the most direct application of the grid approach is to computers and their conglomerates as it is stated above.

Grid arrays have distinct layers. At the foundation, there are the tangible resources to be shared - servers, desktops, networks, storage, software licenses, data, and so on. In the emerging nomenclature of grid computing, these physical resources are talked about in terms of the services that they can add to the grid's capabilities. At the top of the grid hierarchy, users and applications get transparent access to the grid services. In comparison with contemporary local and global networks, the Grid will have much more services for users and much more options for computation and communication.

It is natural to separate three types of grid arrays:

1. An actual grid array, which constitutes a unified organized system of collaborating information processing systems as nodes.
2. A virtual grid array, which is a set of connected information processing systems as nodes.
3. A potential grid array, which is a set of information processing systems that are connected or have a potency to be connected.

**Examples:** A cluster computer is an actual grid array. The INTERNET, as a whole, is a virtual grid array. All computers in the world form a potential grid array. It shows what a high potential that may be achieved through a synthesis of all computers into one grid array.

Grid arrays open new perspectives for building more exact models of computers than those that are give by conventional abstract automata such as Turing machines or finite automata. Elements of a grid array can be devices of different kinds. This allows us to consider any computer as a grid array that consists of microprocessors, the system bus, random access memory (RAM), the read only memory (ROM), cache memory (in some cases), CMOS memory, the expansion card, disk controllers, a video display adapter, a video display a mouse, the keyboard, and so on. Such grid array is called a computer representational grid array. On the next level of the representational grid array each of the microprocessors is subdivided into the ALU, a math coprocessor, cache memory, and the control unit; the system bus is subdivided into the data and address buses; RAM is subdivided into the conventional, upper and extended memory [12], expansion card is subdivided into modem, slots, and ports and so forth.

Another level of the representational grid array consists of integrated circuits. The lowest level of the representational grid array consists of logical elements and similar primitive automata as its nodes.

One more advantage of grid arrays stems from the fact that any information processing system is represented by its hardware, software, and processes that are realized by this system. Consequently, we have hardware, software, and process grid arrays. A computer representational grid array is an example of a hardware grid array. A grid array that consists of programs and their components is a software grid array. On the lowest level, a software grid array consists of separate instructions or operators. A process grid array consists of different processes, operations and transformations.

### 3 GRID AUTOMATA

Grid arrays, as physical systems, are modeled by grid automata, as abstract theoretical systems.

**Definition 2.** A grid automaton is a system of automata, which are situated in a grid, are called nodes, are optionally connected and interact with one another.

Thus, the difference is that a grid array consists of real/physical information processing systems and connections between them, while a grid automaton consists of other abstract automata as its nodes. Connections in a grid automaton can be simple ties as in neural networks. In other cases, these connections can include special automata as transitions in Petri nets [13]. Nodes in a grid automaton can be Boolean elements, Boolean circuits, artificial neurons, finite automata, Turing machines, vector machines, array machines, RAM, inductive Turing machines, neural networks, cellular automata etc. Even more, some of the nodes can be also grid automata.

A grid automaton $G$ is described by three grid characteristics and three node characteristics.

**The grid characteristics are:**

1. The space organization or structure of the grid automaton $G$. This space structure may be in the physical space, reflecting where the corresponding information processing systems (nodes) are situated, or it may be a mathematical structure defined by the geometry of node relations. There are three kinds of space organization of a grid automaton: static structure that is always the same; persistent dynamic structure that eventually changes between different cycles of computation; and flexible dynamic structure that eventually changes at any time of computation. Persistent Turing machines [5] have persistent dynamic structure, while reflexive Turing machines [2] have flexible dynamic structure.

2. The topology of $G$ is determined by the type of the node neighborhood. A neighborhood of a node is the set of those nodes with which this node directly interacts. In a grid, these are often the nodes that are physically the closest to the node in question.
3. The dynamics of $G$ determines by what rules its nodes exchange information with each other and with the environment of $G$. For example, when the interaction of Turing machines in a grid automaton $G$ is determined by a Turing machine, then $G$ is equivalent to a Turing machine. At the same time, when the interaction of Turing machines in a grid automaton $G$ is random, then $G$ is much more powerful than any Turing machine.

Interaction with the environment separates two classes of grid automata/arrays: open grid automata/arrays interact with the environment through definite connections, while closed grid automata/arrays have no interaction with the environment. For example, Turing machines are usually considered as closed automata because they begin functioning from some start state and tape configuration, finish functioning in some final state and tape configuration, and do not have any interactions with their environment.

**The node characteristics are:**

1. **The structure** of the node. For example, one structure determines a finite automaton, while another structure is a Turing machine.

2. **The external dynamics** of the node determines interactions of this node. According to this characteristic there are three types of nodes: accepting nodes that only accept or reject their input; generating nodes that only produce some input; and transducing nodes that both accept some input and produce some input. Note that nodes with the same external dynamics can work in grids with various dynamics.

3. **The internal dynamics** of the node determines what processes go inside this node. For example, the internal dynamics of a finite automaton is defined by its transition function, while the internal dynamics of a Turing machine is defined by its rules. Differences in internal dynamics of nodes are very important because a change in producing the output allows us to go from conventional Turing machines to much more powerful inductive Turing machines of the first order [1].

In comparison with cellular automata, a grid automaton can contain different kinds of automata as its nodes. For example, finite automata, Turing machines and inductive Turing machines can belong to one and the same grid. In comparison with systolic arrays, connections between different nodes in a grid automaton can be arbitrary like connections in neural networks. In comparison with neural networks and Petri nets, a grid automaton contains, as its nodes, more powerful machines than finite automata. Consequently, neural networks, cellular automata, systolic arrays, and Petri nets are special kinds of grid automata. An important property of grid automata is a possibility to realize hierarchical structures, that is, a node can be also a grid automaton. In grid automata, interaction and communication becomes as important as computation. This peculiarity results in a variety of types of automata, their functioning modes, and space organization.

5 **SUBROUTINES, AUTONOMOUS AGENTS, AND VIRTUAL MACHINES IN GRID ARRAYS**

Grid automata as the most advanced abstract information processing systems have different categories of resources: memory, interface (input and output) devices, control devices, operating devices, software, data/knowledge bases.

Modes of resource utilization yield interdependence classification of automata in a grid:
1. **Autonomous automata** with independent resources.
2. **Automata with shared resources**, in which some resource, for example, memory, belongs to one node, but some other nodes from the grid can also use it.
3. **Automata with common resources**, for example, common memory or a database, which belong to two or more nodes from the grid.

Each type of automata implies specific styles of exchange in the grid. For example, there are three layers of exchange for autonomous automata:
1. **Data and program exchange** (distributed storage of information).
2. **Task and workspace exchange** (distributed computation and intelligent agent systems).
3. **System exchange** (data, knowledge, tasks, programs, agents are specific systems in such exchange).

**Definition 3.** A resource of a node is called open when it is accessible for utilization by other nodes.

Example of such resources are: time for execution of its own program; time for execution of some other program; some space in memory (for information storage, for utilization during computation etc.); some software; some device.

Open resources may be used in three modes: for free, for rent, and for lease. There are three techniques to use open resources: to send a task(s) for execution, to send an autonomous agent(s) for work with the resource, and to get direct access to the resource. Each of these techniques has its advantages and shortcomings. The first two techniques are less time and energy consuming for a user, but they are not interactive, that is, they do not allow the user of the resource to make changes in the process of computation until some task is completed. As a result, interaction goes in quanta of actions. A mathematical model for such style of computation is given by persistent Turing machines [5].

Agent technology allows interference, but it is mediated as changes are executed only by the agent who/that is performing the task.

Access to an open resource can be given in three forms: *autonomous access* when the resource is given exclusively to one user; *shared access* when several users, including the owner of the resource, can use this resource; *free access* when anybody can use this resource.

There are many problems with open resources. Now many hackers and other indecent people break into others unprotected computers and use any resources they want without permission. They can even damage the resource and its host.
This is similar to the situation in human society at the beginning of its development when the brute force ruled. However, this state of affairs demonstrated many shortcomings and proved to be an obstacle to progress. Consequently, the rules changed and now society more or less protects its members. Similar situation will come to the world of computers and networks when the assets of people in this world will be protected and people will be able to decide themselves how to use their computing resources – making them open or preserving them to themselves, and when making them open, people will chose in what mode they open a resource (for free, for rent or for lease). Thus, it will be necessary not only to be able to make resources open, but also to be able to close definite resources and to regulate access to them.

The open resource technology allows us to change and to extend the concept of a subroutine. Now a subroutine is a specifically organized sequence of instructions in a program for performing a separate task. This allows the subroutine code to be called from multiple places of the program, even from within itself, in which case the form of computation is called recursive. In the grid array/automaton, it is not necessary to have a subroutine as a part of a program $P$. A virtual subroutine may be a part of a different program, which in its turn may be at a different computer/node than the program $P$ that uses this subroutine.

Using agent technology, we can aggregate computers that use different devices and programs from the grid into one virtual computer system. This approach extends the concept of reconfigurable computer, synthesizing it with the idea of cluster computer. In its turn, virtual computers may be combined into a virtual network or grid.

Grid automata are sufficiently powerful to model different algorithmic structures. It is evident when grid automaton contain as a node such structure. However, even with much weaker nodes grid automata are able to simulate advanced types of algorithms. For example, results of [11] imply the following theorem.

**Theorem 1.** For any finite automaton $A$, there is a grid automaton $G$ that has only Boolean elements as nodes and simulates $A$.

Results of Siegelman and Sontag [14] imply the following theorem.

**Theorem 2.** For any Turing machine $T$, there is a grid automaton $G$ that has only neurons with rational weights as nodes and simulates $T$.

Besides, we have the following result.

**Theorem 3.** For any inductive Turing machine $M$ of order $n$ [1], there is a grid automaton $G$ with output exchange that has only inductive Turing machines of the first order as nodes and simulates $T$.

Here a grid automaton $G$ has output exchange if its nodes can exchange their outputs.

5 CONCLUSION

As any separate computer or, in more generality, any information processing system can be treated as a degenerate grid array, all previous considerations allow us to formulate the following conjecture.

*Any information processing system can be modeled by an extended grid automaton.*

6 REFERENCES