Using Virtualization to Simulate Biological Cells

Ahmad Bazzi
Gunma University
1-5-1 Tenjincho,
Kiryu, Gunma 376-8515
+81-277-30-1837
abazzi@acm.org

Yoshikuni Onozato
Gunma University
1-5-1 Tenjincho,
Kiryu, Gunma 376-8515
+81-277-30-1835
onozato@nztl.cs.gunma-u.ac.jp

Rihito Saito
Gunma University
1-5-1 Tenjincho,
Kiryu, Gunma 376-8515
+81-277-30-1837
saito@nztl.cs.gunma-u.ac.jp

ABSTRACT
In this paper, we discuss the virtualization technologies and the characteristics of biological cells. We notice that both virtual machines and cells are usually "self contained," "logically independent" and relatively "mobile." We work to show the analogy between the two. With this level of common characteristics we conclude that it would be very beneficial for us to use virtual machines to simulate biological cells in general and suggest using "teams" of virtual machines to simulate biological organ functionality. In our opinion, this makes it easier to put our knowledge of biological systems into use.

Keywords
biology, cell, biologically-inspired, virtualization, simulation.

1. INTRODUCTION
Biological systems have evolved across ages to survive the challenges imposed by their harsh environments. During the course of time, a diversity of species could not survive till the present day. Hence we believe that there is a lot to learn from natural systems in general and biological systems in particular to further develop information systems.

With the advance in the processing power of modern computers, complex biologically inspired systems are now being not only researched and simulated but also realized. This varies from self-healing systems to biologically inspired networking [4].

A cell can be defined as the "smallest living biological functional and structural unit of an organism that displays the properties of growth, metabolism, energy cycles, and reproduction."[5] While a virtual machine is a software imitation of a computer that can provide the functionality of a real machine. In this paper we work to establish the analogy that exists between biological cells on one side and virtual machines on the other side.

The objective is to provide a new valid ground that can aid in creating computer simulation of biological systems; this in turn would allow an easier realization of biologically inspired systems. We believe that the suggested analogy can be a promising field for further research and simulation

Bionetics '08, November 25-28, 2008, Hyogo, Japan

We start in section 2 by defining virtualization in general and continue in section 3 to explain the common virtualization approaches and technologies that are in use nowadays. Section 4 provides a brief description of a biological cell and defines the two cell types recounting the main four "specializations" of cells. In section 5, the core of this paper, we discuss the different points that allow us to build an analogy between biological cells and virtual machines. Section 6 provides a short comparison between our analogy with biological cells and previous similar ideas. We end with the conclusion in section 7.

2. DEFINING VIRTUALIZATION
Computer virtualization is a technique where a software layer lies directly or indirectly on top of the hardware to allow multiple guest operating systems to run in *virtual machines*. A virtual machine (VM) will appear like a real computer to the guest operating system, but it is in fact an abstraction of the underlying resources.

Figure 1 shows the different components of a virtual system. We will go through them from bottom to top. At the base level, we have the *physical hardware* that is the real computer hardware. On top of the physical hardware, we can either have an operating system with the *virtualization software* installed on it as an application, or alternatively the virtualization software can be installed directly on top of the physical hardware thus replacing the operating system on the physical machine.

Virtualization software, as the name indicates, provides us with the ability to create virtual machines. The virtual machine, in turn, appears and functions as a real computer for the *guest operating system (OS)*, which can be installed and run on top of it the same way it can be installed and run on ordinary hardware.

Hence, as shown in Figure 1, virtualization allows us to run more than one operating system on the same hardware where each *guest OS* can have its private independent virtual hardware.

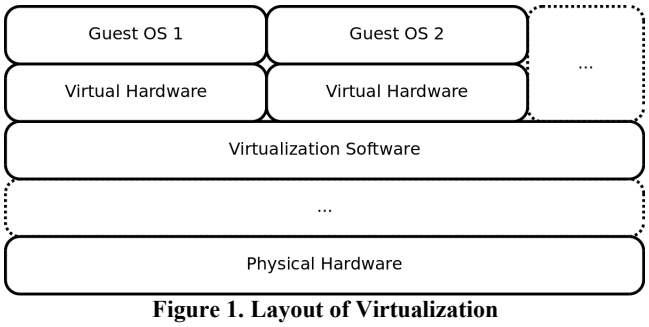

Figure 1. Layout of Virtualization

3. VIRTUALIZATION TECHNIQUES

There are different approaches used to achieve virtualization of computer hardware. Below, we discuss four common techniques that are in use nowadays with their advantages and disadvantages.

3.1 Hardware Emulation

As the name indicates, this technique allows the emulation of a complete set of hardware; in particular, it allows the emulation of a different processor. The virtualization software, emulator in this case, translates the issued machine instructions by the guest operating system to virtual hardware and executes them on the physical hardware in real time. Due to the translation of every instruction, this will inevitably lead to a relatively slower performance; however, this can also provide new potentials. Emulation allows us to execute binary instructions on a non-existent processor. In other words, both an obsolete out-of-production processor and a future processor - that has not been manufactured yet - can be emulated.

Examples of virtualization software that provide emulation capabilities are QEMU and BOCHS.

3.2 Full Virtualization

The second type is *full virtualization* which allows creating complete virtual machines. When properly implemented, this type of virtualization can provide near native speed.

Unlike "hardware emulation" virtualization, this technique imposes a limitation; the virtualized processor must be the same type as the physical machine. The reason for this limitation is that the virtualization software tries to avoid instruction translation in order to provide a higher execution speed. Hence, it favors passing the instructions issued to the virtual machine directly to the physical processor if possible – this is also known as *direct execution*. However, when this is not feasible, the virtualization software will resort to binary translation. Now, the benefit would be clear, as most binary instructions issued to the virtual processor are passed directly to the physical CPU, instead of being translated, we can achieve a near-native speed. [7] This technique is currently implemented using two different approaches:

3.2.1 Hosted Architecture

In this case, the virtualization software will install and run as a privileged application on the host operating system as shown in figure 2. The virtualization software allows creating one or more virtual machines each with its own set of hardware on which guest operating system(s) can be installed. As shown in figure 2, the virtualization software can run alongside other applications installed on the host operating system. Using full virtualization, it allows most instructions to pass directly to the physical CPU. An examples of this approach is VMware Workstation.

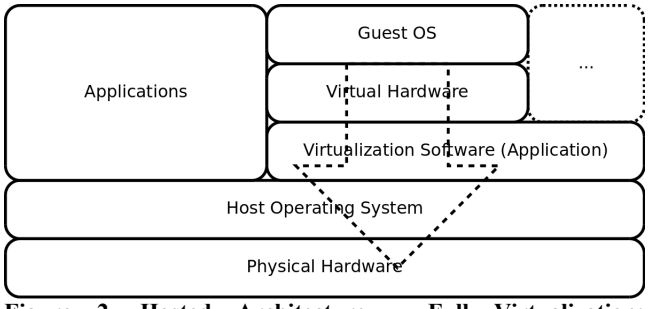

Figure 2. Hosted Architecture – Full Virtualization; Virtualization Software running as an application.

3.2.2 Hypervisor (Bare-Metal) Architecture

To provide further stability when using virtualization in more critical environments, different types of virtualization software have been engineered that can run directly on top of the physical hardware. Thus this approach eliminates the need for another host operating system. As shown in figure 3, the virtualization software will function as the host operating dedicated for the purpose of hosting virtual machines and monitoring them. As the virtualization software has now full control of the physical hardware, this approach can provide greater scalability, robustness and performance compared to the hosted architecture. Hence, it is more suitable for business applications. [7] An example of business solutions utilizing this approach is VMware ESXi.

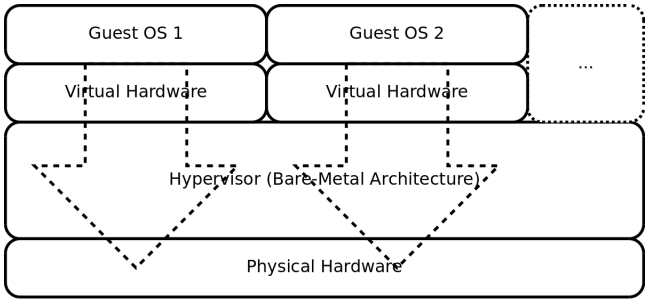

Figure 3. Hypervisor Architecture – Full Virtualization; Virtualization Software replaces the Host Operating System.

3.3 Paravirtualization (OS Assisted Virtualization)

In this technique, the guest OS will be constantly communicating with the *hypervisor* to improve both performance and efficiency. Unfortunately, for this technique to work with the current operating systems, the guest OS has to be modified in order to be made "*aware*" of the hypervisor's existence and consequently to be able to communicate and "coordinate" with it. [7]

In the case of full virtualization, the guest OS is not modified and hence it is totally *unaware* of the underlying virtualization software. By modifying the operating system to achieve a paravirtualization implementation, there is a relative gain in performance; yet, this comes at a maintenance cost where the guest OS has to be modified and this limits portability.

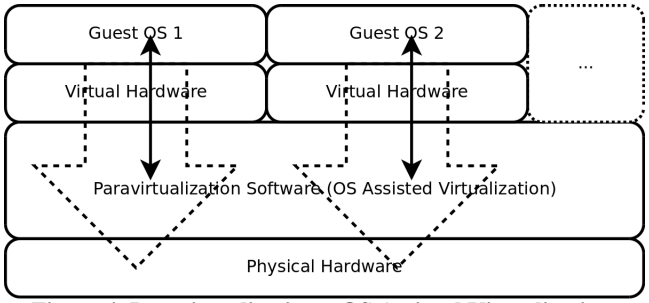

Figure 4. Paravirtualization – OS Assisted Virtualization; Guest OS can communicate with the virtualization software

3.4 Hardware Assisted Virtualization

With the further increase in the popularity of virtualization applications, leading CPU manufacturers are working vividly to provide better support for virtualization technologies in newer generation processors. Eventually CPUs that support virtualization were released in the market as of 2006; however, this could not outperform the classical virtualization approaches

such as the full virtualization and the paravirtualization.[1] This promising technique has yet to bear fruits in the future.

4. BIOLOGICAL CELLS

The cell (biology) is the "basic unit of life" first observed by Robert Hooke in the 17th century and later it was studied further with the advance of microscopy.[3]

Despite the diversity in shapes and sizes of cells, there are only two types of cells: Prokaryotic cells and Eukaryotic cells.

Prokaryotic is the Greek for "before nucleus" as this type of cells does not have a nucleus and has very little visible internal organization. These cells are relatively small with the majority between 1-2 µm in length. The most common example of prokaryotic cells is bacterial cells.[3]

Eukaryotic is the Greek for "with a nucleus" as it is structurally more complex. This type is generally larger with 5-100 µm in length. This is the kind of cells that we find in most organisms from mammals to plants and fungi.[3]

The cells of plants and animals are organized into different *tissues*. Tissues are groups of cells that are specialized to carry out a common function. In animals, for example, there are four major tissue types: epithelium, connective tissue, nervous tissue and muscle [3].

5. CELLS AND VIRTUAL MACHINES

There are different aspects of resemblance between the biological cells and virtual machines; we start listing them below then we summarize these points in Table 1.

5.1 Specialization

As mentioned in section 4.2, there are different types of tissue cells each specialized in a certain function; an analogy holds when it comes to computer systems and virtual machines. For example, a virtual machine can practically have any function; this ranges from web servers, database servers to security systems and firewalls. For available example applications, the reader is referred to VMware Virtual Appliance Marketplace [8].

5.2 Inter-communication

It is a necessity for life that the millions of cells that compose a multi-cellular organism to exchange chemical messages (transmitters); this is called inter-cellular communication. [3] The same holds true for computer systems where the communication between the diverse information systems is achieved through the implemented network protocols.

To further elaborate on this idea, let's take a small sized network as shown in Figure 5. This network has four servers: User directory server, DNS (domain name system) server, mail server and a firewall. We also assume that the administrators have opted to host all these servers on virtual machines.

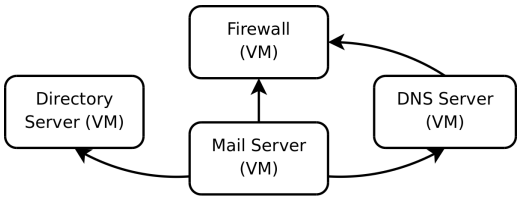

Figure 5. Inter-Communication between Virtual Machines; Like biological cells, VMs communicate with each other.

During the course of normal utilization of this network, the mail server will constantly need to query the DNS server to be able to deliver emails, the mail server also needs to constantly communicate with the directory server in order to authenticate users before granting them access to their email inboxes.

5.3 Intra-signaling

In intracellular signaling, cells use internal signaling mechanisms that allow them to alter their behavior in response to internal changes or to external events. Biologically speaking, this is achieved through intracellular messengers which is an intracellular solute whose concentration changes in response to cell stimulation [3]. On the other hand information systems uses similar techniques such as interrupts and signals. For example, when the user presses a key on the keyboard or clicks the mouse button, the system should respond accordingly.

5.4 Self-Containment

According to [6], a cell is "the fundamental unit of living organisms; a structure that is capable of independent reproduction and that consists of cytoplasm and a nucleus, or a nuclear zone, surrounded by a cell membrane." Hence, we can say that a cell is self contained. A virtual machine has its operating system running on its own virtual CPU with its virtual memory on a virtual hard disk, etc. Like a cell, a virtual machine can carry out its basic operations without the need of the external system and hence it can be said to be self-contained as well.

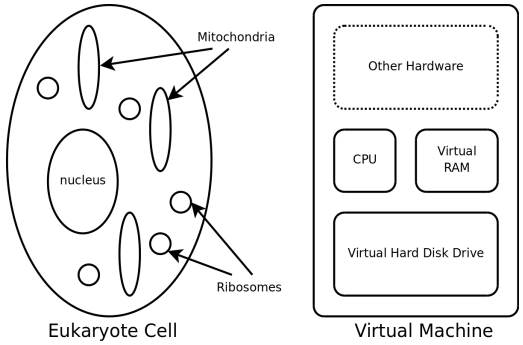

Figure 6. Cells and Virtual Machines are Self-Contained.

5.5 Self-Maintaining

A cell can take in nutrients, convert them into energy and carry out its normal function, the function that was encoded in its DNA [2]. Every information system, on the other hand has a certain task to execute depending on how it was configured by the system administrator. With the progress in information technology, the system is becoming more self-maintaining. For example, an operating system (installed on a VM in this case) running GNU/Linux or MS Windows can be easily configured to check for security patches and other system updates on a regular basis. The system can further be configured to install the updates at specified times to avoid or minimize user interruption, occasionally it can even restart and resume operation without any user interference. For an example update configuration, the reader is referred to [9] which details the case of Windows XP. This can go even further to "self-healing."

5.6 Logical Disconnection

A cell can be logically disconnected from and totally unaware of the host as it carries out its own basic operations. Similarly, a virtual machine "lives" without the need to be aware of the host existence.

The cell is logically disconnected from the host organism and practically unaware of the host being. For example, a bacteria can invade a certain host, carry out its function that it is

"programmed" to do without being really aware of the host organism.

The same goes for the virtual machine. It just carries out its function using its set of allocated virtual hardware without being aware of the real details of the host system. For example, the host server can be a personal computer with average hardware specifications or it can be a multi-processor server with a fail-proof RAID configuration; either case, the guest virtual OS is unaware of the underlying host.

Table 1. Analogy between Cells and Virtual Machines

Property	**Cells**	**Virtual Machines**
Specialization	Every cell has a certain role to carry out.	Every virtual machine functions as a particular server.
Inter-Communication	Cells uses chemical signals to communicate between each other.	VMs use network protocols to communicate between each other.
Intra-Signaling	Through intracellular messengers, cells alter their behavior.	Through interrupts and internal signals, a VM OS alters its behavior.
Self-Containment	A cell has its own nucleus or nuclear zone, cytoplasm, etc.	A VM has its own OS, applications, etc.
Self-Maintaining	A cell takes its nutrients converts them into energy and carries out its function encoded by its DNA.	A VM OS can handle a variety of tasks from running certain services to downloading and installing updates automatically.
Logical Disconnection	The cell carries out its function without being really aware of the host organism.	A VM does not need to be aware of the host machine existence.
Mobility	A cell can move or be moved from one host organism to another.	A VM can be moved or even copied from one host machine to another.
Physical Dependence	A cell usually needs to be inside a host organism.	A VM has to be in a host in order to run.

5.7 Mobility

Being logically disconnected from the host system gives rise to an interesting feature, mobility. This useful characteristic offers great utility in both biology and virtualization. In biology, this leads to the possibility of blood transfusion, organ transplant and a variety of other medically beneficial operations.

In virtual machines (VMs), this provides one interesting benefit. A VM that "lived" on one set of hardware (host system) can be easily moved (or copied) to another host system. Practically once a hardware becomes obsolete or possibly defective, the system administrator can easily move the VM to another more recent and robust host. Interestingly enough, this resembles organ donation; only in this case, cloning (copy) is as simple as moving (cut).

5.8 Physical Dependence

Most kinds of cells cannot live by themselves, they are often found inside an organism. For example, blood cells, nerve cells, etc. On the other hand, it is possible to store blood, for example, in controlled environments. However, during the storage period the blood won't be accomplishing its main tasks, such as providing oxygen to the cells, instead, it will be in a stagnant state.

The same holds true for virtual machines. Although we can back up a VM to a storage media, it won't be operational by that. For a VM to be on, it has to be run on a computer.

6. COMPARISON

The idea of simulating a biological cell might not be new; similar ideas were proposed in grid computing and agent systems. However, the VM approach can offer several subtle advantages that makes it more versatile for simulating biological cells.

From grid computing approach, a computer, for example, can be used to represent a biological cell. The main limitation in this approach is the lack of mobility. While a VM can be moved from one system to the other using basic software instructions, any relocation a computer generally requires human interference. Another convenience would be the ability to copy a VM; something which cannot be done to a grid's computer-cell.

Compared to agent systems, the VM has the advantage of being self-contained. It can save inside itself any configuration or customization issued by the user as well any data saved. The physical representation of the VM will continue to be a set of previously defined files that can be conveniently copied or backed up. On the other hand, an agent, being a program or a script, it needs to save data in separate files or on an external database.

7. CONCLUSION

Virtualization technology has armed us with another tool to create a digital form of "life" that mimics biological systems in general and cells in particular. In our opinion, the strong analogy that holds between the cells and virtual machines provides a valid basis for further research. This in turn can offer potentials for creating a digital form that can further mimic the way that biological systems work in.

8. REFERENCES

[1] Adams, K. and Agesen, O., "A Comparison of Software and Hardware Techniques for the x86 Virtualization", Proceedings of the 12th international conference on Architectural support for programming languages and operating systems, October 21-25, 2006, San Jose, California, USA.

[2] Alberts, B., et al., "Molecular Biology of the Cell", Fifth Edition, Garland, December 2007.

[3] Bolsover, S. R., et al., "Cell Biology: A Short Course", Wiley-Liss 2004.

[4] Leibnitz, K., et al., "Biologically Inspired Networking", Cognitive Networks: Towards Self-Aware Networks, Chapter 1, pp. 1-21, John Wiley & Sons, September 2007.

[5] Mai, L. L., et al., The Cambridge Dictionary of Human Biology and Evolution, Cambridge University Press, 2005.

[6] Stenesh, J., Dictionary of Biochemistry and Molecular Biology, John Wiley & Sons, 1989.

[7] "VMware, Understanding Full Virtualization, Paravirtualization, and Hardware Assist." http://www.vmware.com/solutions/whitepapers/virtualization.html

[8] VMware Virtual Appliance Marketplace, http://www.vmware.com/vmtn/appliances/directory/

[9] Windows XP, How to Configure and Use Automatic Updates, http://support.microsoft.com/kb/306525