

# ENTERPRISE STRATEGY GROUP POSITION PAPER



## Virtualization and the Case for Universal Grid Architectures

By Steve Duplessie, Founder and Senior Analyst  
November 2012

# Contents

Abstract .....	4
History .....	4
The Problem Exposed: Virtualization .....	5
Virtualization Today and What It Broke .....	6
What Are the Application-related Problems We Face in a Virtualized World? .....	7
Virtualization Tomorrow and What Will Break .....	10
Grid—Compute, Network, and Storage: Past, Present, and Future .....	12
The Bigger Truth .....	15

All trademark names are property of their respective companies. Information contained in this publication has been obtained by sources The Enterprise Strategy Group (ESG) considers to be reliable but is not warranted by ESG. This publication may contain opinions of ESG, which are subject to change from time to time. This publication is copyrighted by The Enterprise Strategy Group, Inc. Any reproduction or redistribution of this publication, in whole or in part, whether in hard-copy format, electronically, or otherwise to persons not authorized to receive it, without the express consent of The Enterprise Strategy Group, Inc., is in violation of U.S. copyright law and will be subject to an action for civil damages and, if applicable, criminal prosecution. Should you have any questions, please contact ESG Client Relations at 508.482.0188.

“Moore’s law allowed IT vendors to keep pace with the lion’s share of demand from the buying community. Until now.”



# ABSTRACT

Traditional computing architectures have been based about 99% on monolithic systems clustered together to form larger virtual systems or to create a system designed for a higher level of availability. These very architectures are hampering the mobility and dynamism that virtualization enables. For virtual environments to reach their full potential, infrastructure must move beyond monolithic clustered architectures to far more extensible grids.

## HISTORY

Since the beginning of the industrial computing era, systems have been designed in a monolithic fashion. Namely, they are, effectively, self-contained compute, memory, and I/O systems “in a box.”


Although those “boxes” stretch out internally and externally, and although the capabilities of those elements and connections have ridden the technology advancement wave, fundamentally, they continue to operate the same old way. Data lives on some storage device; it is fed into memory where it is processed, and the pattern is reversed.

Thus, to accommodate more of anything—more users on the system, more data to process, more transactions, faster processing, etc., the industry has responded by constantly developing bigger, faster, more capable systems ... that still remain largely monolithic.

As systems became more crucial to the operation of various business functions, secondary or redundant (highly available) systems were required. Enter the era of “clustering”—namely, where one monolithic system can take over for another monolithic system should a failure occur.

Clusters have grown in sophistication and size (as have their monolithic components), but they remain comparatively small and confined when compared with the alternative approach—grid.

## THE PROBLEM EXPOSED: VIRTUALIZATION



Moore's law has meant that we have been able to effectively double our capabilities (processing and capacity capabilities anyway, not I/O) every 18 months. That doubling has, by and large, allowed IT vendors to keep pace with the lion's share of demand from the commercial computing buying community.

Until now.

Monolithic architectures, either clustered or standalone, have historically been finite and static. To execute an application on a system, one must run that application on that system. The system is configured with an operating system (overall stack controller) and applications beneath that OS. The applications execute under rigid, specific conditions directly related to that OS and that infrastructure stack.

In such a situation, clustering is normally relegated to having System A take over the application workload of System B if and when System B goes down. The process happens in many different ways, but basically, that's it. Sometimes, we have more than a 1:1 cluster relationship—sometimes we can have 4:1 or even 8:1. But we never have 1,000:1 or more.

As long as the world was comfortable in the knowledge that an application could only execute under those physical parameters, clustering has been fine.

Virtualization changes all of that.

## VIRTUALIZATION TODAY AND WHAT IT BROKE

Today, server virtualization has allowed customers to make one physical stack of hardware appear to the OS/application environments as many individual stacks, allowing much better hardware utilization, efficiency, etc.

Building an N-node cluster of individual hardware stacks with high availability is great. It enables much better operating efficiency. Often, users can eliminate many of their previous smaller stacks of kit and push all their application environments onto virtual machines.

But the applications reap no benefit—actually, they often lose benefit—by doing this. You save money on hardware and operations, but your applications don't perform better. Nor do they become more available or scalable on virtual hardware than they would on their own dedicated hardware. This is simple reality.

---

**“Users can push all their application environments onto virtual machines, but the applications reap no benefit. Actually, they often lose benefit.”**

---

## What are the application-related problems we face in a virtualized world?

---

“Today, an Exchange instance may exist unknowingly beside 18 other apps that may alter their behavior and affect that poor Exchange instance. Only now, the admin is likely unable to take corrective action easily.”

---

**Visibility:** We have none. When Exchange was the only thing that ran on a physical stack and we had a problem, we knew where the problem existed: inside that self-contained unit of computing. It was an imperfect science, but we knew the problem had to be there.

There could certainly be interdependencies outside of that stack: The network could be screwed up by another system or switch port, or remote calls to other systems/databases could be hosed up by those systems, etc. But no interdependencies existed within it. Thus, if tuning, optimizing, or problem solving was required within that operating environment, it was possible to solve the problem.

Today, that same Exchange instance exists unknowingly beside 18 other Exchange instances, next to file serving apps, rendering apps, etc. Any of them may suddenly alter their behavior and dramatically affect that poor Exchange instance. Only now, the Exchange administrator is most likely unable to find the cause and take corrective action easily.

Compound this situation with the possibility of external interdependencies (tied to the database server, the network, etc.), and the complexity of the overall system can increase by orders of magnitude. This is where we find ourselves today.

**I/O:** Today, IT can “containerize” an application with an operating system to create a virtual machine that executes on a physical server. The server’s processor, memory, and other resources are partitioned or provisioned to accommodate each virtual machine as long as the resources exist.

Because most data is now housed externally in virtual environments, those environments create I/O-related interdependencies on each other. If one physical server contains ten virtual machines vying for data access on one physical disk array, the likelihood of varied and unpredictable I/O clearly arises. And it is causing performance problems throughout the industry today.

How does the industry respond? By making already-expensive arrays even more expensive—adding higher-performing components (such as flash) as a tier or a cache. And when you think about the intent of server virtualization, such activities bring about the exact opposite consequences that were intended—namely, better asset utilization and lower costs.

All we did was shuffle the problem. Server cores are cheap and getting cheaper, but storage is actually becoming more expensive. We are trying to fix the problem the same way we fixed monolithic infrastructure problems: Make it bigger and faster ... and destroy whatever efficiency and utilization gains we picked up elsewhere.

---

“If one physical server contains ten virtual machines vying for data access on one physical disk array, the likelihood of varied and unpredictable I/O clearly arises. And it is causing performance problems throughout the industry today.”

---



---

“If it’s hard to plan for appropriate I/O capabilities on a single physical stack with multiple virtual machines, how does one plan for an environment in which virtual machines may move across physical boundaries?”

---

**Mobility:** Virtual machines are not restricted to one physical box; they can move. (They must do so holistically, but they can move.) That capability creates several problems, and the biggest ones are I/O related. If it’s hard to plan for appropriate I/O capabilities on a single physical stack with multiple virtual machines, how does one plan for an environment in which virtual machines may move across physical boundaries?

A virtual machine that moves to another physical machine (because, perhaps, it can get on a much faster machine to execute its program, then hop back to a slower machine while it’s idle—good idea!) must continue to be able to access its data seamlessly. Otherwise, that VM is dead.

One big problem (but a boon for industry) is that the data must be on a networked device that can connect to the VM no matter where it moves. When a VM moves, we open ourselves to the same unpredictable I/O issues as before—we can’t control what processes other VMs are executing on the new physical machine and what their I/O patterns are, nor how they will affect us. Very few storage offerings have true QoS across every node in the data center, and across every VM.

## VIRTUALIZATION TOMORROW AND WHAT WILL BREAK

As great as virtualization 1.0 is—and as difficult as the problems it creates are—we are in the “easy” phase. Things are going to get much more difficult.

Fundamentally, we’re in 1972. The mainframe of that era was a single big box with a ton of resources that allowed us to create virtual machine instances: We carved out some resources and dedicated them to specific virtual machines. If one VM/application environment needed more of anything and we had more to give, we’d allocate it.

It’s essentially the same as what we do today. But back then, we could even do it, to some degree, with I/O.

To summarize the problem: Being able to make one physical box look like ten is interesting and compelling. Making ten physical boxes look and act like one is far more valuable. That is where we are heading.

Today, if you run out of processing capability on your VM, you can either give it more cores within your physical machine if you have them, or you can move the VM to a bigger, more powerful physical machine and let it run on those cores—the very definition of monolithic computing.

---

**“Being able to make one physical box look like ten is interesting and compelling. Making ten physical boxes look and act like one is far more valuable. That is where we are heading.”**

---

Tomorrow, you will have the ability to distribute, or federate, your application processing across cores, across systems, across boxes as you need to (and shrink back accordingly) —all without an application knowing or caring.

Years ago I wrote something to this effect, and I remain committed to the concept of “liquid” computing.

In short, we will eventually live in a world where physical boxes represent nothing more than carrying containers of valuable resources. All the resources in a data center (and conceivably beyond) will be pooled, merged, and utilized for as long as required, then relinquished back to the pool from which they came.

We'll have a pool of processing capability, memory, cache, and I/O from which VM applications will carve out their requirements for the job at hand, and then disappear until needed again.

“We will eventually live in a world where physical boxes represent nothing more than carrying containers of valuable resources.”

# GRID—COMPUTE, NETWORK, AND STORAGE PAST, PRESENT, AND FUTURE

That container idea is not farfetched. We have examples where, while not entirely automated, the concept already works.

High-performance computing (HPC) environments have existed for years doing just this. A single “job” or application is massively parallelized to execute small pieces across thousands of individual physical servers, performing a task thousands of times faster than if the jobs were executing serially on a single processor.

To the application, it’s one machine—one really big machine with a ton of cores.

If you want to span an application executable across physical nodes to process, you don’t use a “cluster.” You use a grid.

Know what the bottleneck in HPC environments is most of the time? It’s I/O. Because although the compute side may be grid, the storage side is normally a big, fast, fat, shared monolithic storage instance. Guess what has to change?

Networks used to be monolithic. Many still are. Traits such as resiliency can be considered in much the same way as clustering. We can fail over links. But most networks remain monolithic. They have some big cores and pipes, but when one pipe is overused, the system can’t easily move or absorb excess capacity from other underused pipes. This is, however, the promise of software-defined networks and other technologies: grow, shrink, and expand across pipes and boxes on demand.

Grid networking: What do we do today? We add faster, bigger, more expensive monolithic boxes. We replace a 1Gb with a 10Gb infrastructure—and along the way, we propagate Moore's law negatively. We adversely increase the amount of unused, inaccessible resource bandwidth in our data center, all to provide the few big hogs the pipe they need.

Storage is the final frontier. We adopted storage clustering soon after server clustering and never really looked back. Today 99% of all storage arrays are monolithic, two-controller (clustered) boxes. Run out of stuff in one box, bring in another, maybe even cluster those together.

Certain storage arrays can support more than two controller clusters today, but they are uncommon. And even those tend to just be larger clusters—four pairs of two-controller clusters, for example. They are still monolithic.

Finally, a grid is a federation of resources, unconstrained by traditional architectures. In my grid computing/HPC example, 1,000 servers with 1,000 network connections being squeezed down through two (or eight or 16 or 32) storage controllers only to then connect to 1,000 disk drives makes no sense. Why aren't there 1,000 disk controllers, virtual or otherwise? Eventually, there will be.

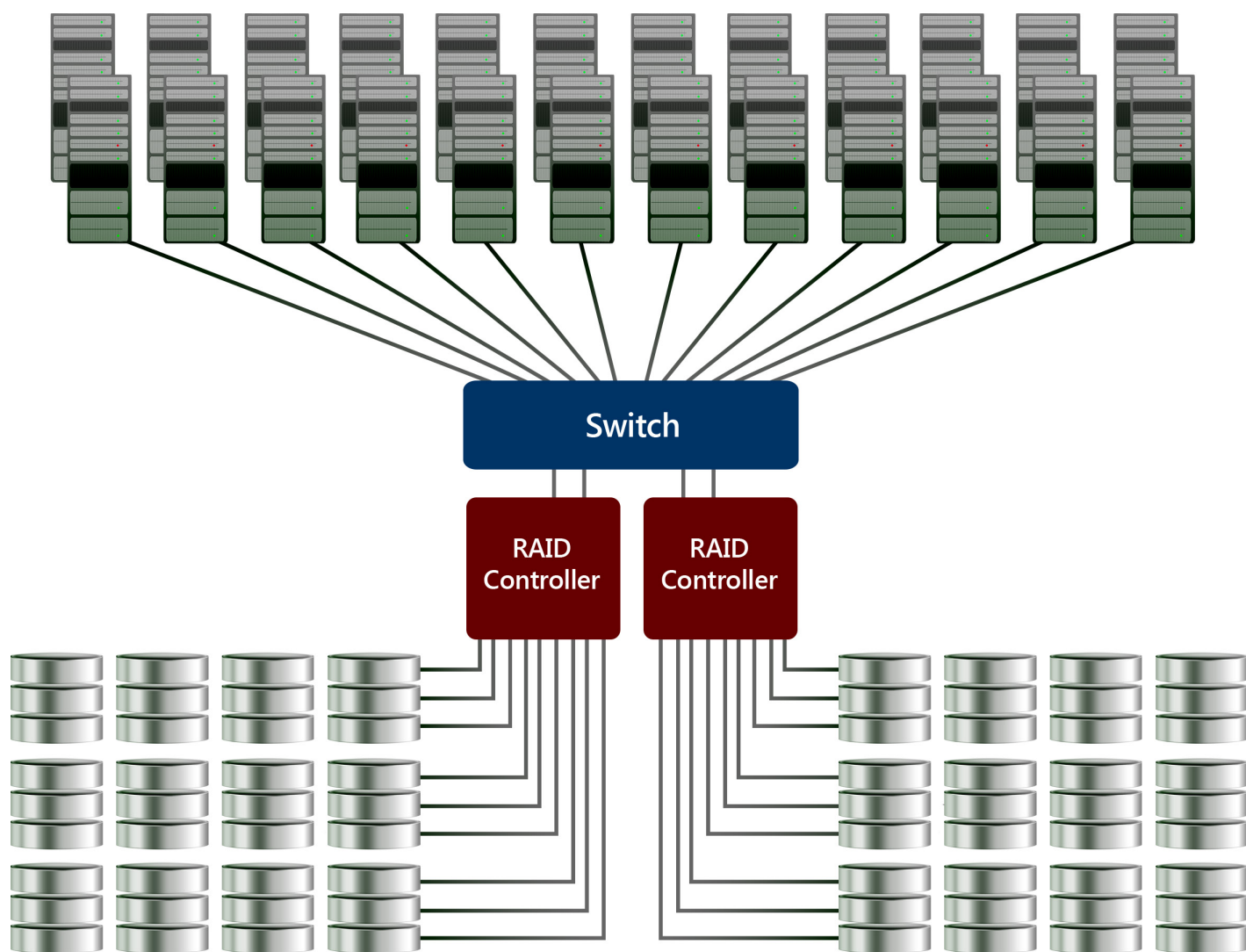
Just as you are restricted to your weakest physical link in a virtual environment today, so you shall remain tomorrow.

---

**“What do we do today? We add faster, bigger, more expensive monolithic boxes—along the way, we propagate Moore's law negatively, adversely increasing the unused, inaccessible resource bandwidth in our data center.”**

---

A grid is a federation of resources,  
unconstrained by traditional architectures.



# THE BIGGER TRUTH

To be able to truly scale to the needs of the virtual machine requirements of the not-too-distant future, here's what the entire IT industry must do:

- Stop developing monolithic answers to grid problems. Face the fact that your R&D should focus on leveraging horizontal, inexpensive componentry that has no physical restrictions.
- Develop your applications to take advantage of multiple physical components ... not just bigger individual ones.
- Build in "self-healing." No physical failure of anything should ever keep an executable from completing its task.

It's also important to realize that we really are in the first inning of the game. Sure it's a new, big game, but look at what has happened since 1972. IBM owned commercial computing, but that didn't stop the IT industry from constantly reinventing itself and creating outrageous opportunity and wealth along the way. If VMware is the equivalent of IBM in 1972, then who will become the next EMC, Oracle, or NetApp? Who will fade away like Digital, Wang, or Prime?

By continuing to develop monolithic infrastructure implementations, we've been doing the same thing architecturally for more than 50 years. Historically, no significant trend lasts much longer. The time is ripe for an upheaval.



Enterprise Strategy Group | **Getting to the bigger truth.**<sup>TM</sup>

20 Asylum Street    Milford MA 01757    P. 508 428 0188    F. 508 428 0128