

Eliminating the I/O Blender Effect and Realizing World Class Storage Performance in Virtual Storage Environments

DECEMBER 2014

WHITE PAPER BY JON TOIGO

Managing Principal
Toigo Partners International LLC

Chairman
Data Management Institute LLC



Introduction

Storage has long been a stumbling block in server virtualization initiatives. A few years ago, stories began to appear in the trade press about virtualization plans “stalling out” in some companies well short of initial workload consolidation goals, in many cases because no one had anticipated the impact of hypervisor technology on application performance. Moreover, the solutions that were being recommended by vendors at the time, which usually entailed the ripping and replacing of “legacy” SAN and NAS storage infrastructure, carried a hefty price tag. In fact, such strategies were often so costly that they actually offset whatever cost savings that were supposed to accrue to server consolidation in the first place.

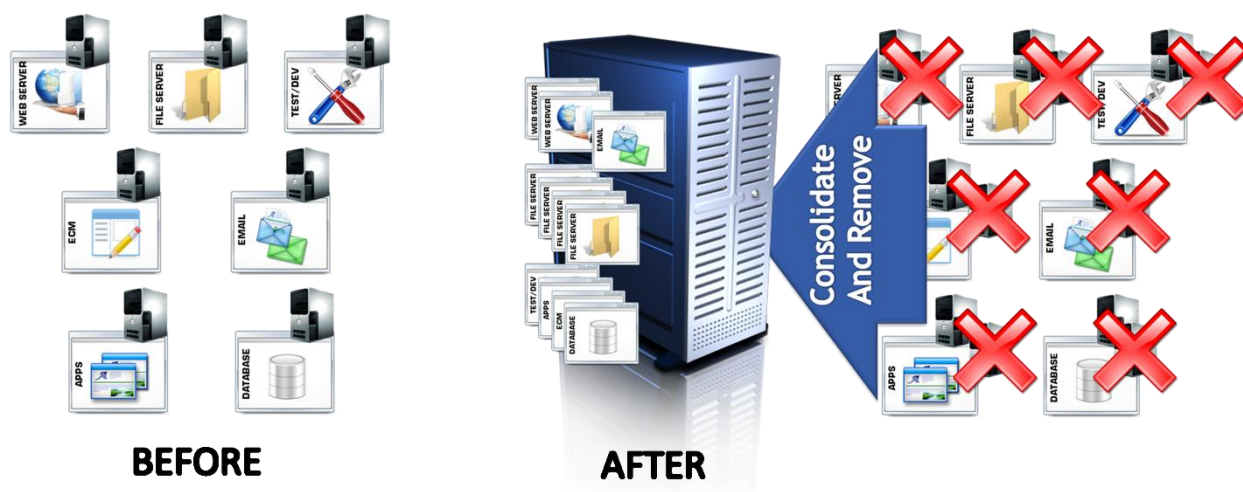
Basically, hypervisor software and storage vendors all sought to address the issue of application slowdowns with strategies and workarounds that aimed at minimizing effects, attacking symptoms rather than addressing the root problem: something we now call the “I/O Blender Effect.” Consequently, none of the strategies actually worked very well.

This paper examines the root cause of poor application performance, the I/O Blender Effect, and how it is brought about by server virtualization. The paper will then survey some of the approaches that have been tried for containing the problem and will discuss a technology strategy that has been implemented by StarWind Software that shows considerable promise for dealing effectively with the I/O Blender.

Historical Perspective

Not long ago, virtualization took hold as a data center technology based on a compelling business value case. First, it promised to reduce costs by reducing CAPEX and OPEX spending on sprawling server infrastructure.

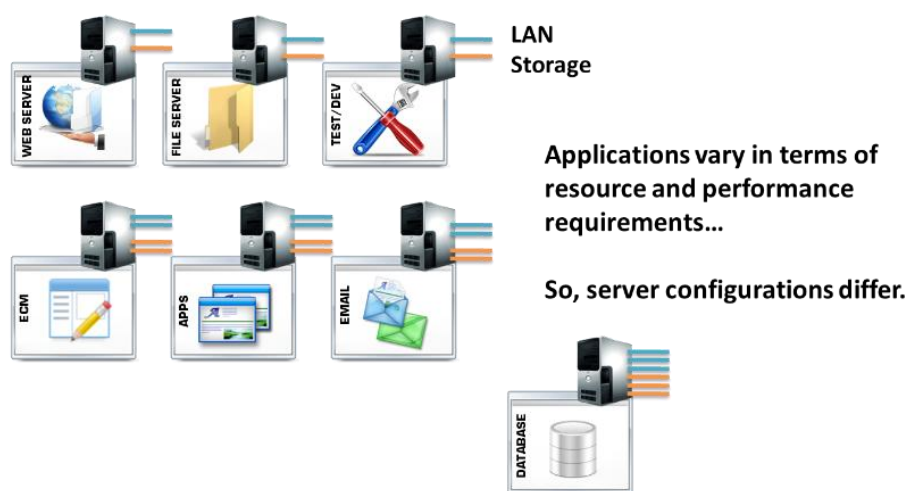
Basically, most companies had deployed many physical servers in their corporate IT environments – with each server typically being used to host a single application. Hypervisor computing advocates argued that such infrastructure was cost-inefficient. Server resources were rarely fully utilized, so the infrastructure was, per the old saw, “overprovisioned and underutilized.” In addition to wasting server technology resources, companies were also wasting energy, floor space and environmental conditioning resources. And, of course, labor costs were disproportionately high given the number of servers that needed to be maintained, the paucity of centralized tools for management, and the need to grow staff with technology scaling.



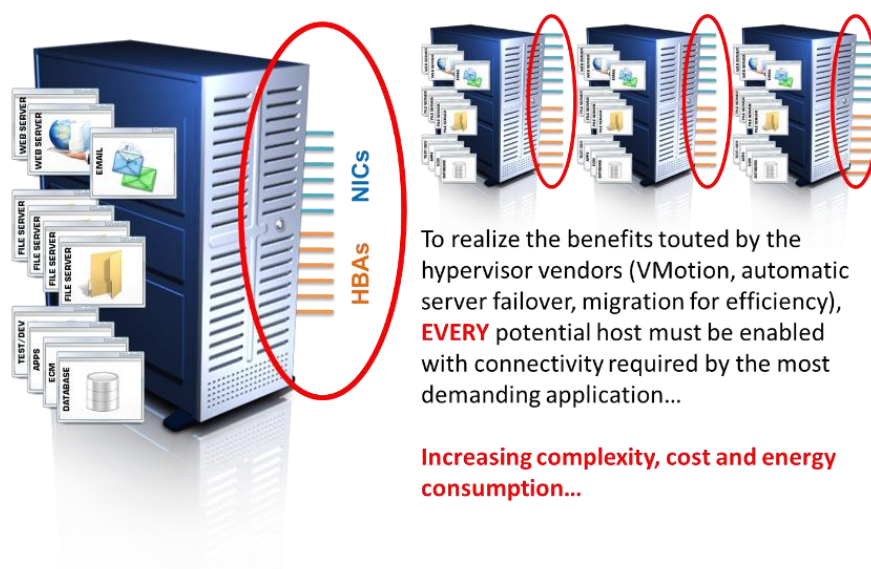
With the advent of chip architectures from Intel and others that were designed to support multi-tenancy (multiple workloads), the idea took hold that server virtualization technology could be used to consolidate many wasteful physical application server environments onto fewer physical hosting platforms by first converting the applications and their operating systems into virtual machines. Once consolidated, the hardware that was originally used to host individual workload could either be repurposed or donated to a recycling plant.

The upside effect of this strategy seemed boundless. Businesses would have lower energy costs, lower IT costs generally, improved utilization efficiency of hardware investments, more uptime, and less hardware to manage. Vendors also insisted that IT professionals would be able to provision IT resources more quickly, enhancing “IT agility” and responsiveness to business needs. The net effect would be improved service levels at a lower cost to the firm – the ultimate “win-win.”

There were some challenges fairly early on. First, the configurations of servers that were being virtualized differed from one another – some only in nuanced ways, others in very significant ways. Some workloads required a substantial physical machine resources to accommodate their performance requirements – for example, different numbers of I/O paths to data storage taking the form of different combinations of LAN connections and storage fabric connections.



This complicated the task of enabling the movement of workload from one virtual server host to another, a promised benefit of server virtualization. Basically, differences in the hosting requirements of applications led to a server configuration model that was driven by the highest common denominator. In other words, the configuration of the host machine had to accommodate the most demanding application workload that might possibly be hosted there. So, administrators found themselves having to overbuild their virtual machine hosting environments, which had an impact on the complexity, cost and energy consumption of each server.



For example, if a virtual machine was a SQL server database and application workload that required six LAN connections to accommodate user access and six storage I/O pathways to the back end storage infrastructure to accommodate data access, the server host needed to provide all of these connections by creating all of the related host bus adapters and network interface cards. That configuration then needed to be replicated on every server that might eventually host the app either as a function of workload cut-and-paste, vMotion or high availability failover.

Some early efforts to minimize the impact of this resource provisioning requirement included efforts by Xsigo and others to virtualize NICs and HBAs, a forerunner of the current software-defined networking architectures that have come to dominate the trade press lately. The idea was to direct LAN and storage I/O traffic to virtual I/O cards and ports rather than to physical equipment. This traffic would then be forwarded via software to an off-server director switch via a wideband interconnect such as Infiniband. From there, the director switch would send the I/O wherever it needed to go.

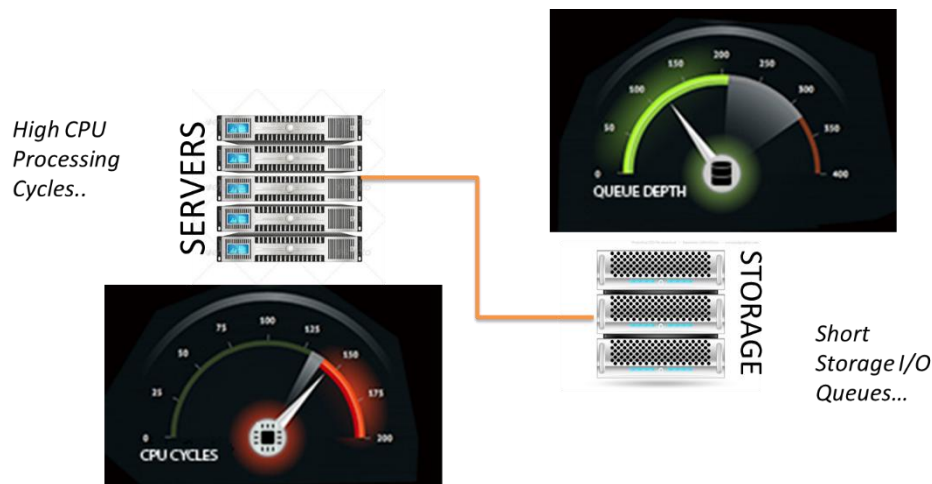
While an interesting counter to the cost problem of highly configured servers, the solution did nothing to fix the problem of application performance slowdown. Therefore, attention turned to another possible explanation for the performance issue: slow storage.

Some hypervisor vendors argued that the nature of centralized and shared legacy SAN storage were creating barriers to I/O performance. There was a bit of conflation going on in this claim. While server virtualization advocates identified a legitimate issue with fabric-based storage, namely that its hard-coded LUN addressing was interfering with the dynamic movement of workload between different server platforms, this issue had nothing to do with I/O performance.

It was true that moving an application from one host to another necessitated the reconfiguration of LUN addresses, so that the re-hosted application could find its LUNs in the legacy SAN infrastructure. The need to manually reconfigure application LUN addressing could be said to be a barrier to the rapid (and “agile”) re-allocation of storage resources to the workload. However, it had nothing to do with application performance once connections to storage were made.

In fact, a simple check of storage I/O performance showed that, in most cases, the processors on virtual servers hosting workload were running “hot” – that is, using a lot of processor cycles to process application instructions and data. However, looking at storage queue depths – a measure of the number of I/Os waiting to be written or read to disk storage – one would likely find that queues are actually quite small. Simply put, the storage was operating just fine, responding to read and write requests without delay.

So, application slowdowns were not the result of slow storage failing to keep up with application I/O. In fact, this situation described a problem with the software – the hypervisor software – that was responsible for organizing I/Os for processing rather than any problem with the storage infrastructure itself.



What followed was a line of experiments to try to work around rather than actually resolve the performance problem. By 2010, it was clear to most hypervisor software vendors that users were experiencing application performance issues linked to hypervisor software itself. Moreover, surveys demonstrated that performance issues were stalling server virtualization programs.

For a time, hypervisor vendors continued to attribute the problem to the uneven performance of legacy storage arrays or to their proprietary and monolithic architecture. Each array had its own array controller operating its own value add software and management utilities. As in pre-virtualization days, proprietary storage was creating management headaches, adding expense to infrastructure and administration.

These issues, which were well known prior to the advent of hypervisor computing, had little to do with the apparent I/O handling problems in the hypervisor software itself. Yet, VMware tried to address its application performance issues by introducing nine non-standard SCSI commands to work around the problem by offloading certain storage operations to the controllers of attached legacy storage arrays. The vendor said that doing so would remove as much as 80 percent of I/O workload that was then handled by the hypervisor – not exactly fixing the I/O bottleneck but reducing its severity and impact on hosted application performance.

Despite the fact that these new SCSI commands had not been submitted for consideration or approval by the ANSI T-10 Standard committee responsible for maintaining the canonical primitives of SCSI, hardware vendors worked diligently to support them anyway in order to support their customers...but not without considerable grumbling about the hypervisor vendor's failure to notify the storage industry or ANSI about their proprietary commands.

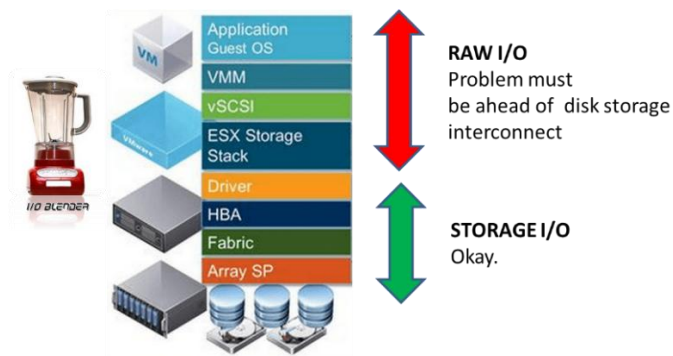
A year later, the vendor changed these arbitrary commands for a second time, again without notification to the industry. But they finally relented and sought formal approval from the ANSI T-10 committee.

Offloading some I/O processing did yield a modest improvement in the performance of some applications, but the effect did not last, especially as more workload was piled on to a given server, mainly because I/O offload did not address the real problem. A generous reading of history saw the hypervisor vendor working on the core problem of I/O throttling, but at the same time capitalizing on its new found “muscle” in the storage market to convince them to build their own storage appliance: a VSAN array. In the end, this proprietary storage array product did not deliver general purpose storage behind virtual servers but instead a limited storage solution for the vendor’s own VMDK files. The VSAN appliance was quickly panned by reviewers.

Still suggesting that the solution to I/O-related application performance issues would come in the form of a change in storage hardware, the hypervisor vendors began pursuing a new “hyper-converged” server-storage architecture that featured a return to direct-attached storage topology combined with server clustering and ongoing data replication between cluster nodes, all orchestrated and controlled by the hypervisor. Efforts produced a set of software and APIs intended to facilitate the use of commodity storage hardware behind clustered servers to create a more responsive, easily deployed and readily scaled cluster storage architecture called a Virtual SAN. Unfortunately, none of these workarounds (not even the virtual SAN) by themselves attacked the root cause of application performance issues: the I/O blender.

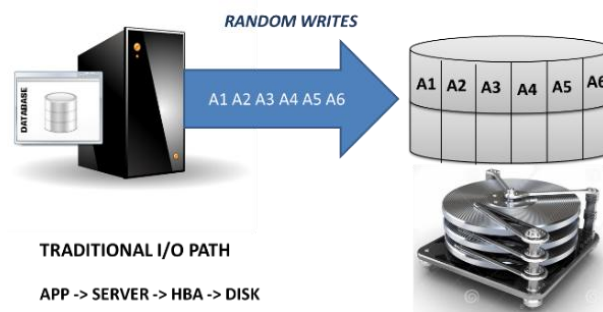
Enter the I/O Blender

In some post-2013 presentations by VMware, it was acknowledged that I/O “queues” or bottlenecks could develop at different locations in the hardware/software stack. One visual showed that chokepoints or queues can and do occur prior to the storage hardware layer. This was the beginning of the vendor’s acknowledgement of something most users with even rudimentary performance monitoring and measurement tools had already figured out: that the application performance problem wasn’t storage at all. It had to do with the multiple processes involved in handling RAW I/O occurring in the software stack ahead of disk reads and writes.

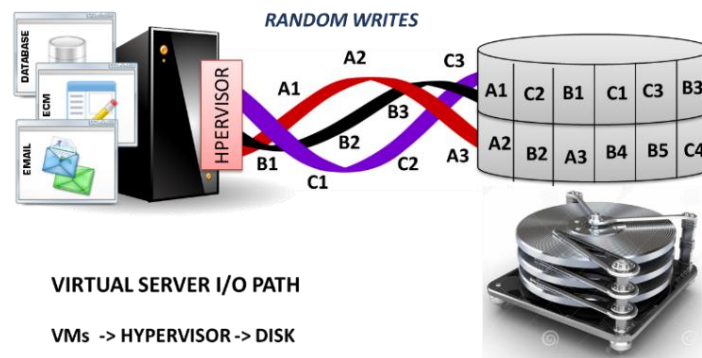


Somewhere between the Virtual Machine Guest OS and the ESX storage stack something was going awry with RAW I/O processing that was creating one or more RAW I/O chokepoints. This throttle was less apparent in the case of applications that created long block reads and writes. But in the case of applications that issued a lot of small reads and writes, VM performance could become noticeably degraded fairly quickly.

The I/O Blender effect is easy to understand. Start simple by considering the basic I/O path in a non-virtualized system: I/O is generated by the application and passed through the server operating system and Basic Input/Output Subsystem (BIOS) as a series of random and sequential writes. These writes are sent, via a host bus adapter, across an interconnect to the controller of either a disk drive or an array of disk drives which then writes the data in a fairly sequential and well-organized way, thereby preserving performance on retrieval.



Of course, this is a very simplistic view. Additional complexity may be introduced by caching data to random access memory or increasingly to flash memory, and of course, many array controllers or server operating systems feature RAID technologies that perform data replication, parity data storage, or other functions to preserve and protect data. However, this is basically the way the I/O path operates: from application to server to HBA to disk.



In the case of a server running a hypervisor and several virtual machines, the I/O path becomes more randomized. The virtual machines are all generating random and sequential I/O. However, the hypervisor aggregates all of this I/O and morphs it into a stream of small random I/O that pass through the HBA to the disk array. The randomized I/O from multiple streams doesn't take long to populate the disk storage with a clutter of randomized bits that take longer to write, retrieve, modify, update and delete. Thus, the Blender Effect, if unchecked, can make storage very unresponsive.

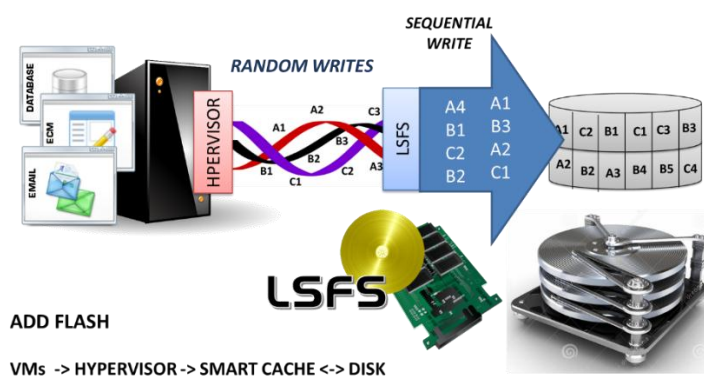
One trick that some vendors promoted for resolving the I/O Blender effect was simply to cache writes: flash memory could be used to augment DRAM to provide a temporary location for holding bits while "spoofing" or concealing from the application the impact of the I/O blender. Designers assumed that silicon storage could find randomly stored bits without the thrashing of mechanical I/O heads working across disk platters and concluded that the impact of lots of small randomized I/O wouldn't matter as much when initial storage was to a memory cache.

That assumption opened the door for a lot of misinformation about the effect of loading up host systems with flash memory on application performance. In fact, the impact of small block writes on Flash memories proved to be a sticking point with this strategy.

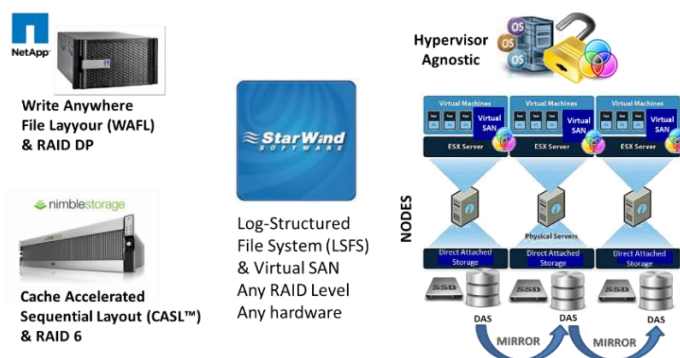
In flash storage, even the best multi-layer cell memories are rated to be able to handle approximately 250,000 writes per cell location before the cell wears out and must be retired together with all of the other cells in its group. Thus, the fit for the flash cache solution was linked to the amount and type of I/O generated by the VMs individually and collectively. More to the point, loading systems with flash usually proved to deliver little improvement, whether in terms of technical or cost performance efficiency over doing nothing at all.

Enter Starwind Software Log-Structured File System

However, there was another option. To solve the problem of random writes, StarWind Software is credited with being among the first to develop a solution in the form of a Log-Structured File System or LSFS technology. Simply put, LSFS caches random I/O writes, then converts them into a single sequential write stream. The solution offers multiple advantages over traditional storage. Mainly, it improves the performance of primary storage and significantly reduces wear and tear on Flash storage where this technology is used.



By aggregating small writes in DRAM before writing them to Flash, StarWind Software's LSFS addresses the flash wear problem by significantly decreasing the quantity of erase cycles and rewrite operations that are typically the consequence of erasing cells for each new small block write.



Already a leader in hypervisor agnostic software defined storage, StarWind Software was one of the first to apply Log Structuring to random I/O in connection with hardware agnostic virtual SANs. Nimble Storage, with its adaptive flash technology, had created a similar technology for its all flash arrays, and it could be argued that NetApp's Write-Anywhere File Layout used in conjunction with RAID DP was also an effort to reorganize random I/O for improved performance. But StarWind Software was the first to apply this concept in a manner intended to enable virtual SANs that supports Server all RAID levels, any storage hardware, smart flash caching, and in-line deduplication.

Benefits of Log-Structured File System with Virtual SAN Architecture

With the implementation of StarWind Software's Log-Structured File System, users of hypervisor computing with software-defined storage architecture will finally see the performance bump they have been looking for from their virtual workloads. This involves the elimination of small writes directly to magnetic or flash storage, and the buffering followed by the sequential re-structuring of random I/O in a DRAM cache. In this strategy, pages are simply assembled in memory, then written to the target storage device.

Another benefit of StarWind LSFS™ is its support for all forms of RAID. Other offerings in this space, coming mostly from hardware vendors, tend to support only specific RAID levels used by that vendor's hardware. Since StarWind Software works in the software-defined storage space and is agnostic about the hardware components of storage, it supports all hardware and all RAID levels.

Another benefit is that LSFS minimizes what has come to be called the RAID write penalty – the delay that accrues when a RAID controller lays down parity, striping or mirrored data across different spindles for data protection. StarWind Software's LSFS minimizes the write penalty by writing in full stripes and avoiding stripe overwrite.

While supporting all RAID levels, LSFS also enables other data protection capabilities, such as snapshots. Since snapshots, like parity information, can be made without a write penalty and redirected on write to different storage targets, efficiency is increased without punishing applications with latency. Additionally, deduplication can be done at the LSFS cache level, so data reduction processes are executed more efficiently. This is superior to NetApp's approach since it does the data reduction in line, using minimum write cycles.








Finally, StarWind Software's LSFS is a flash friendly technology. LSFS is the first technology for Virtual SANs that leverages flash caches in a manner that respects the capabilities and limitations of the technology and that minimizes cell wear.

In the final analysis, for those seeking a software-defined storage solution that also addresses the I/O Blender problem smartly and effectively, StarWind Software's LSFS is worth a look. It does not eliminate the I/O Blender Effect, but it does provide an elegant means for reducing its impact on application performance and platform cost.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of StarWind Software.

StarWind Software gives credit to Jon Toigo and confers rights to use the current document as it is, unmodified, for public purposes.

Contacts

US Headquarters	EMEA and APAC
 1-617-449-7717	 +44 20 3769 1857 (UK)
 1-617-507-5845	 +49 302 1788 849 (Germany)
	 +33 097 7197 857 (France)
	 +7 495 975 94 39 (Russian Federation and CIS)
	 1-866-790-2646

Customer Support Portal: <https://www.starwind.com/support>

Support Forum: <https://www.starwind.com/forums>

Sales: sales@starwind.com

General Information: info@starwind.com

In 2016, Gartner named StarWind “Cool Vendor for Compute Platforms”.

Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings or other designation. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.



<https://www.starwind.com> StarWind Software, Inc. 35 Village Rd., Suite 100, Middleton, MA 01949 USA

©2016, StarWind Software Inc. All rights reserved.