Validating Software-Defined Storage Operating Models for The Enterprise

EBOOK CATEGORY: TECHNICAL ANALYSIS

Release 1.0
JUNE 2021

Validating SDS Operating Models for the Enterprise 2021 - First Edition

Published by Brookend Limited.

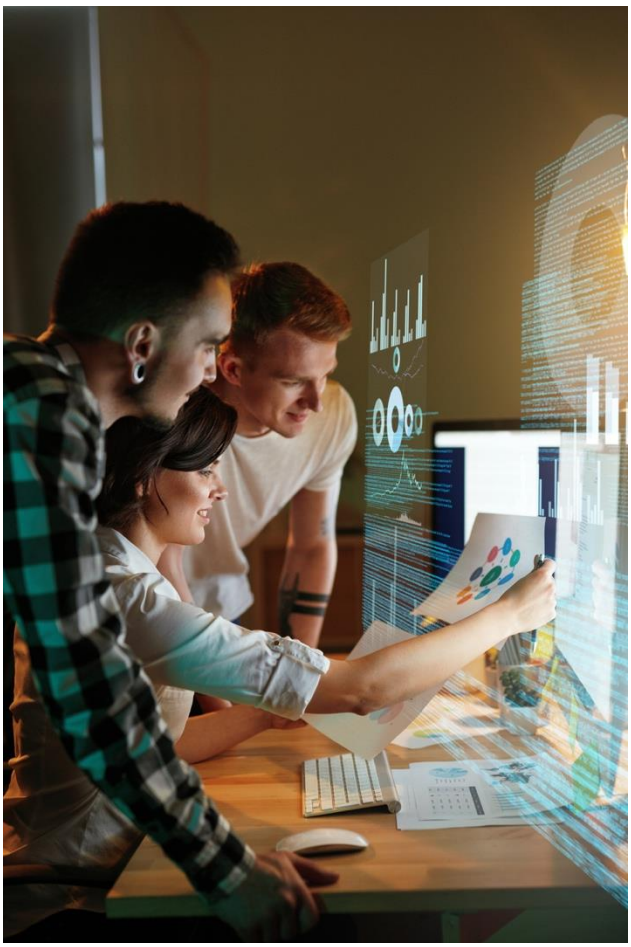
Document reference number BRKSW0140.

CONTENTS

Introduction 3
The Evolution of SDS 4
The Rise of Open Source 5
The Phases of SDS 6
Requirements 8
The Four Operating Models 9
- The Proprietary Model 9
- DIY (Self-design and Build) 10
- Supported Solution 10
- Task Specific 11

Conclusions 12
More Information 13
- The Author 13

Introduction



Software-defined storage (SDS) has become the de-facto standard for modern storage solutions. Hardware standardisation and commoditisation provides the framework for software to take centre stage and be a driver for storage solutions innovation. As the SDS market has evolved and matured over the last ten years, implementation models have iterated through the use of self-build commodity hardware towards more tightly specified hardware options.

In the early years of SDS, software and hardware disaggregation provided the basis to buy and build from cheap and readily available hardware components. Modern solutions are much more sophisticated, making use of new technology like power-efficient Arm cores, persistent memory, and process offload devices, including FPGAs. Software is now written specifically to take advantage of the hardware capabilities, with improved programming interfaces and APIs. The benefit of this revolution has been to deliver much more reliable and enterprise-capable storage solutions to businesses.

In this eBook, we look at the choices available for IT organisations looking to make a strategic move to software-defined storage. We examine the deployment models that encompass the implications of choosing between proprietary and open-source software and between proprietary and entirely self-designed and built hardware. Our conclusion is that while self-build has a place, enterprise organisations will ultimately want the reliability of engineered task-specific solutions with the flexibility of open-source software. This combination capitalises on the integration skills of the vendor and the depth of software development expertise brought by Open Source.

The Evolution of SDS

Software-defined storage or SDS has transformed from a niche option to one that dominates today's mainstream storage market. Twenty years ago, vendors often differentiated their products through bespoke hardware. Most of that advantage has now gone, and legacy vendors use mostly the same hardware components. Few vendors build their solutions entirely from scratch but choose to take advantage of mature and sophisticated components that now include all-flash media and PCIe offload cards. Storage hardware has standardised on the rack-mounted server (typically 1/2/4U) with high-performance networking and hot-swappable media. This transition covers both appliance-based and SDS solutions.

In part, this change has derived from the commoditisation of technology onto the Intel x86 architecture, standardised server designs and the reliability of modern storage media. Hardware prices have continued to drop, year on year, with storage leading the race. As prices have declined, so component capacities and performance have increased. This evolution makes it easier than ever to build customised storage solutions at a fraction of what they cost a decade ago.

In parallel with hardware commoditisation, the implementation of new storage features is now achieved primarily in software. Even the traditional enterprise market incumbents have moved to a model of standardised hardware and feature-rich software. Intel has improved and expanded the x86 instruction set, enabling storage software vendors to implement resource-hungry tasks like deduplication and encryption directly in software.

Gartner predicts the SDS revolution to reach 50% of the storage market by 2024 (from 15% in 2020). In reality, this figure is arguably already much higher if we include the appliance-based solutions from major vendors that are, to all intents and purposes, already SDS under the covers.

The Rise of Open Source

A significant factor in the prevalence of SDS has been the success of Open Source. Open-source software evolved from freemium and shareware models of the 1990s to become a widely accepted solution for infrastructure and application deployment. Many of the core components of modern computing derive from open-source projects. Linux is the most remarkable example of the success of the open-source model. The operating system has developed from a desktop x86 alternative to commercial Unix to become the dominant platform for applications in both the enterprise data centre and the public cloud.

As a foundation, Linux supports open-source applications across the IT landscape, including databases, infrastructure tools, storage, networking, and infrastructure management. The CNCF Cloud Native Landscape demonstrates the breadth of open-source solutions that are freely available and widely supported today. In the storage world, Ceph is one of the most mature leading open-source offerings available.

In the storage market, most appliance-based storage platforms, including proprietary storage operating systems, are based on the Linux kernel. Most storage solutions (both proprietary and Open Source) are developed and only run on the Linux platform. This standardisation demonstrates the power of Linux and Open Source in general. The use of open-source storage software has been a game-changer for many organisations. The academic and scientific community were early adopters in this space, taking advantage of the low cost needed to develop petabyte-scale archives.

Why Ceph?

The object storage market has been a perfect entry-point for software-defined storage, and no other solution exemplifies this more than Ceph. From early origins as a scale-out object platform, Ceph now offers file, block and object storage from the same infrastructure.

Ceph is widely used across commercial governmental and academic institutions that include the United States Department of Defense, Intel, ARM and CERN. In 2018, the Ceph Foundation was established as a successor to the Ceph Advisory Board, providing a solid foundation and funding under the direction of the Linux Foundation. This approach demonstrates an industry commitment to the Ceph platform within the boundaries of traditional open-source control. Ceph differs from other open-source software-defined storage projects in its diversity of delivery and support options from vendors like Red Hat, Canonical, and SoftIron and others.

The open-source model enables the end-user to have a greater influence on the direction and features of storage software, which could even include contributing and reviewing code. The end-user can choose a support model appropriate to the value of the application, from no support to complete enterprise support contracts. Typically (and especially in the market verticals mentioned), building large-scale storage solutions on Open Source has required investment in low-level technical skills and significant operational support and testing. This model isn't generally acceptable to the enterprise market. We will discuss the implications of this in a moment.

The Phases of SDS

Software-defined storage has evolved through several phases. Each generation demonstrates a leap forward in both the adoption of SDS solutions and the acceptance of the proprietary and open-source development models.

Hardware Separation

This step represents the genesis of software-defined storage as vendors started the transition by offering existing storage solutions that could be deployed on commodity hardware. End-users had the capability to re-use old technology or do their own appliance design. Hardware separation represented a fundamental change in the cost model for storage appliances, as they provided greater transparency to the buyer.

Bespoke SDS

The move to storage software designed explicitly around an SDS model of disaggregated hardware and software was arguably driven by the object storage market. Object stores are a natural fit for scale-out commodity hardware and don't have the strict performance characteristics required by block storage platforms (although this position is changing). Outside of object stores, vendors developed solutions that could be deployed on any hardware, including virtual machines. These solutions were specifically marketed as virtual storage appliances or VSAs.

Infrastructure Abstraction

The third phase of SDS development saw a greater emphasis on hardware abstraction and multi-tenancy. Performance capabilities became a function of quality of service (QoS), providing greater maturity and flexibility to the process of delivering persistent storage. At this point in the evolution of SDS, vendors had developed functionality suitable for enterprise or managed service providers (including the multi-tenancy and QoS features already highlighted). Prior to this, SDS was arguably more widely adopted by cost-conscious small/medium enterprises and the academic/scientific communities.

Hardware Realignment

While SDS provided the ability to detach the closely coupled link between hardware and software, advancements in performance with technologies such as NVMe and persistent memory have made the integration of hardware and software more critical than ever before. SSDs deliver I/O latencies at multiple magnitudes better than hard drives (around 100µs compared to 5-10ms) while delivering hundreds of thousands of IOPS. Intel Optane SSDs provide greater performance with significantly higher write endurance than SSDs. NVMe has brought in a device API that overcomes the scalability challenges of SCSI and SAS with fast media. Hardware innovations result in year-on-year improvements. Modern SDS needs to fully understand and exploit the capabilities of these new and complex technologies, otherwise the result is wastage and inefficiency.

The SDS market has reached an inflection point where the relationship between hardware and software is more critical than ever. A great example of this is the transition to SMR hard drives. SMR (Shingled Magnetic Recording) is an internal HDD feature for delivering improved I/O density but results in extremely poor random write I/O performance. Storage software needs to be able to identify and write to SMR drives sequentially to mitigate the random I/O performance problem. Poorly designed systems that don't understand the characteristics of hardware media risk wasting resources and losing the flexibility and cost savings that SDS was born to solve. This capability is now being introduced with SSDs using ZNS.

Another aspect of hardware re-alignment is the dependency on some hardware designs. NVMe and persistent memory (generally Intel Optane) have introduced new programming models and architectural designs that create hardware and software dependencies. As an example, Optane SSDs provide a high level of endurance compared to NAND flash and can therefore be used for write-intensive I/O. Optane persistent memory in the DIMM format only supports Intel Xeon 3rd scalable processors, creating a limitation on the processor architecture. Neither technology has a mature alternative, which results in a dependency for solutions that use these technologies exclusively.

However, new technology also offers a chance to optimise solutions in other ways. Arm processor designs, for example, provide greater power and cooling efficiencies than Intel x86 CPUs. Software and solution vendors have started to take advantage of processor choices in the design of new products, and this is reflected in a broader range of solutions in many storage portfolios.

Requirements

What does it take to build a reliable and efficient SDS solution? The four categories of infrastructure deployment include design, build and deploy, operate and maintain.

DESIGN

In SDS, the end-user has the option to be fully in control of the design or to delegate to a solution vendor. The design process can be complicated, with many choices of components and configurations. Getting the right design requires a deep knowledge of the hardware and components being used, as well as how the software will exploit those resources. In the proprietary and task-specific operating models, the vendor bears the burden of designing, building, (and sometimes manufacturing) the combination of hardware and software. In both the DIY and supported solution models, the customer bears this burden which should include life cycle considerations over the life of the investment.

TEST AND DEPLOY

Integrated appliance vendors build, and stress test their products during the development cycle and before shipping to reduce the risk of DoA (dead on arrival) or unreliable systems. As a result, hardware and software come prepared for deployment into a cluster and often with tools to simplify the installation process. With SDS solutions where the buyer selects hardware and software separately, the end-user must decide whether to take on the risk of buying and maintaining component inventory as part of their solution. Many aspects influence component availability, which is why appliance vendors establish direct relationships with the manufacturers. During deployment, the user must ensure correct and consistent firmware, OS, and software throughout. Some software vendors and communities provide detailed prerequisites about hardware state prior to installation.

OPERATE

Once in place, solutions must be managed, including allocating capacity and deploying services and users, capacity expansion, and ongoing data movement or recovery activities. Depending on the global nature of the solution, the SDS end-user may need to keep inventory in multiple global locations. This process can be a logistical challenge and result in increased costs if not managed effectively.

MAINTAIN

The process of support is distinct from operation, as it covers the issues of managing software bugs, deploying new software updates and ensuring continuous uptime of the system, which includes hardware and software repair or replacement. SDS introduces the potential for a split support model, which can leave the end-user as the coordination point between suppliers in the DIY and support solution cases.

It's clear there are many considerations to make when choosing the most suitable operational model for SDS. Picking the correct approach is essential as it has impacts on availability, cost and the ongoing practicality of the solution. As we will discuss, implementing SDS offers choices, each of which has strengths and weaknesses.

The Four Operating Models

In a recent paper, SoftIron reviewed four support models for modern software-defined storage solutions. We review these here, with an eye on the evolution and requirements of SDS we've already discussed.

Picking the right solution is an exercise in delivering the right time to value for the business. The proprietary appliance model is being subsumed by self-build and task-specific designs that need to deliver quickly to the business and faster than ever before.

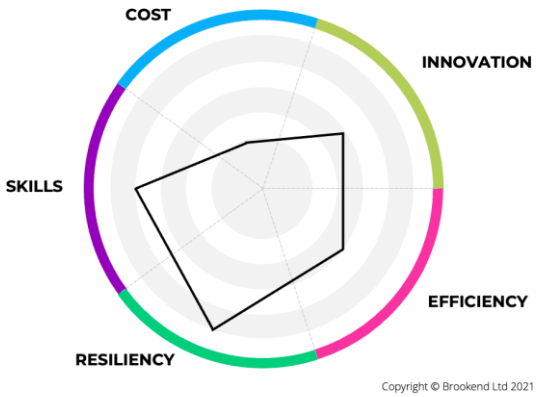
Typically, vendors don't offer proprietary storage software with a wide choice of customer hardware, as this is a complex model to support. Instead, they tend to provide a hardware compatibility list which narrows customer choice and limits the scope of vendor testing. We do, however, see entirely custom (DIY) options, supported solutions that span Open Source and proprietary software, entirely proprietary solutions, and a deployment model SoftIron describes as "Task Specific".

In the definitions that follow, we compare each one using a spider diagram of five metrics, scoring from one (lowest) to five (highest). The metrics are based on personal experience and understanding of each model in the enterprise. The metrics are:

- Cost – is the model likely to be more or less cost-efficient compared to its peers?
- Efficiency – does the model drive efficiency both from an operational and resources perspective?
- Innovation – does the model offer room for innovation, both by the vendor and the customer?
- Resiliency - Does the model provide long-term resiliency through design and ongoing support?
- Skills – does the model require significant investment in non-transferable skills?

The Proprietary Model

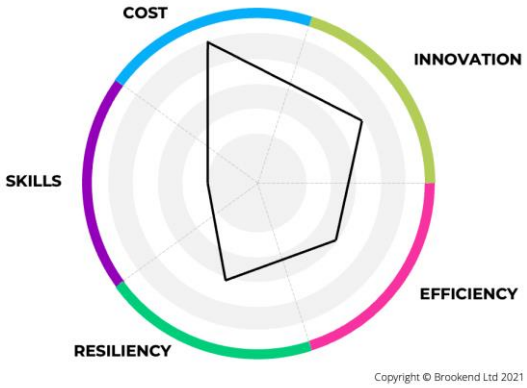
In this scenario, the customer continues to buy tightly integrated hardware and commercial software. The vendor is the sole developer behind the software components. As an integrated package, the vendor can continue to obfuscate the underlying cost of the solution. The customer has no option to customise the product (outside of model variants being offered) and must take the hardware provided by the vendor. This, in many cases, could be at least one generation behind current technology because vendors want products in the market for multiple years.



This option scores poorly for cost, as the vendor continues to sell the solution as an entire package but does score highly for resiliency, as the vendor has access to data across the whole customer base. From an innovation perspective, choice is dictated by the vendor and may be slow to take advantage of new media and systems hardware. With regards to skills, the customer doesn't need intimate knowledge of the solution, other than at the operational level. However, any skills built up in operating the platform are unlikely to transfer usefully elsewhere. This risk can be a big challenge if a vendor chooses, for example, to discontinue an entire product line. This solution is not particularly efficient as the hardware can't be used for other purposes and is based on a design that suits the vendor's ecosystem, not the customer.

DIY (Self-design and Build)

In this option, the customer does all of the design and build work in-house. While this offers more flexibility than proprietary models, the end user is now the system integrator, taking commodity components and validating their use with the solution. This model is found across both open-source and proprietary software where there is no prescribed or validated HCL (hardware compatibility list). As a result, every end-user of the solution will go through a process of discovering what hardware works well and what doesn't (reinventing the wheel).



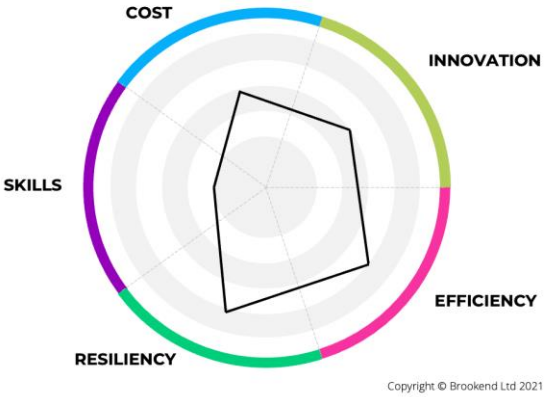
In the DIY model, the end-user must maintain inventory and have alternative hardware components when supplies dry up or for when the vendor releases new models. There's no relationship between the hardware supplier and the software developers. Therefore, if bugs are encountered, the end-user may (or may not) be able to get support to fix the problem. Any issue that results in O/S or driver patches could take months to filter through to the latest software release.

As bugs and vulnerabilities are discovered in software (within the application and O/S), the end-user is responsible for testing, validating, and rolling out new configurations. This process is generally reactive to issues rather than managed proactively with a commercial vendor offering.

The DIY option scores well on cost because users can repurpose old investments or use the cheapest off the shelf hardware, but poorly on skills, where the end-user must invest significant resources. Resiliency, Efficiency and Innovation score just above average, as they are influenced by the degree of skill developed by the end-user.

Supported Solution

Approach three is to use a predefined or supported hardware configuration and take support from a vendor for the software. This model has been widely adopted by solution providers looking to exploit the benefits of open-source development, while finding a commercial model to make money.



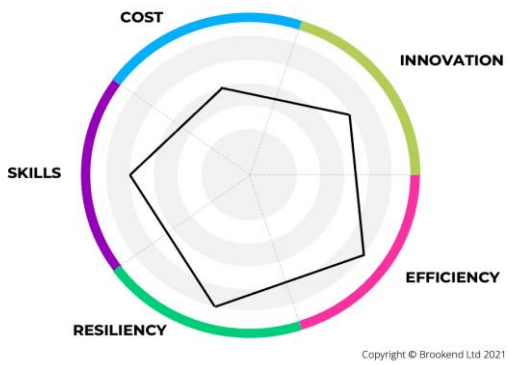
While better than pure DIY, the solution restricts the customer to supported configurations on an HCL. These are the hardware components tested by the software support vendor and could be limited to specific and costly hardware. At any time, the vendor can change the supported configuration, resulting in potentially expensive upgrades. Due to the volume of testing involved, the supported solution can result in a narrow HCL, including one where older hardware drops off the list quickly, forcing the end-user into a more rapid refresh cycle than may be practical or desired. Worse still, the design of a software solution

can dictate specific hardware components (such as Optane SSDs or persistent memory). This results in a solution that is theoretically agnostic to the underlying hardware but in practical terms, is wholly dependent on specific components.

The supported solution scores poorly in skills, as the end-user needs to build up nearly as much knowledge on the solution as the DIY option. The solution cost scores an average mark, as a narrow HCL could impact the ability to use low-cost components. The score for Innovation is also dragged down by the most limited choice of components, while Resiliency and Efficiency score above average due to the restricted nature of the hardware support list.

Task Specific

SoftIron offers what is described as a "Task Specific" solution. In this model, the hardware is designed by SoftIron and configured to work optimally with storage software, in this instance, Ceph. In using a model where the hardware is selected, designed and tested by a vendor, the customer removes many challenges with managing inventory, product lifecycle and component selection. The vendor brings the experience and knowledge of potentially thousands of customers to bear, continually optimising the design of the hardware to match the software. The result is a significant improvement in time to value for the customer. There's no reason why SoftIron couldn't also offer solutions for other open-source platforms. However, Ceph provides a mix of block, object and file protocols and is already widely adopted across the industry.



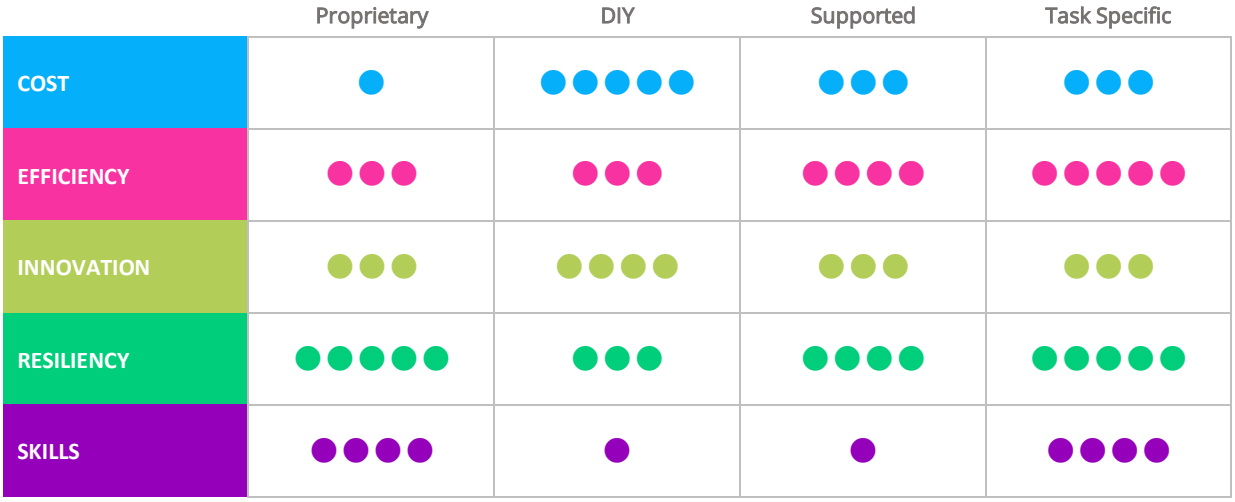
The SoftIron solution is designed to maximize Ceph performance for a given storage class and reduce the skill burden that typically comes with running an open-source storage system. This requires that the customer choose Ceph as the software layer of their SDS solution. However, Ceph is free, has a large and active community, and is designed to work on a wide variety of heterogeneous hardware. Users have the freedom to make future investments in other technology vendors and need not worry about the software disappearing due to changes in a vendor's software strategy. SoftIron adheres to the support of upstream Ceph, which means SoftIron appliances will interoperate with non-SoftIron Ceph clusters.

SoftIron is the only vendor offering support on mixed task specific and generic environments, providing further flexibility, and avoiding vendor lock-in. This approach allows software-defined storage consumers to use multiple deployment models in different parts of the infrastructure. As a result, customers benefit from the marginal cost benefits of the DIY model and the skills and efficiency in the task-specific model based on which is more appropriate for the deployment, with a single operations and support experience across the storage management layer. SoftIron can also provide support for and integrate into other vendors' Ceph distributions, allowing customers to transition to SoftIron without outages or forklift upgrades.

This solution scores average for cost, as the vendor must make money within the model, typically on support. The solution offers good Innovation, as SoftIron provides additional tools to support integration with legacy storage protocols and provides an enterprise-class management experience. Both Resiliency and Efficiency score highly, as the vendor can take advantage of the knowledge from a broad user base while designing for efficient use. The solution scores well on skills, as SoftIron provides additional software and hardware integration features that reduce the specialized skill set required for deployment, operation and support.

Conclusions

Software-defined storage has transformed through multiple stages, from disaggregation to today's "smart" disaggregated solutions that combine the best of hardware and software together. Software is nothing without hardware on which to run it, with the most effective solutions exploiting the best features of hardware without incurring undue lock-in.



Picking the right solution for your business will depend on the sensitivity towards each of the metrics we measure and present. For many companies, the idea of a DIY solution might prove too daunting when the IT teams have been used to direction and assistance by a storage vendor. The DIY model also introduces an increased "time to value" as the IT team has to learn, design and build an entire solution.

The proprietary model represents the opposite side of the equation, giving the customer little or no choice in the platform other than picking a hardware configuration off the shelf. This route is acceptable for customers that have no interest or the resources to look at other alternatives.

The supported model will be a good choice for many businesses, as this solution does provide a fallback support position. However, the user is still dependent on the proprietary software being used, which has a direct impact on hardware choices.

The SoftIron task specific model is attractive in that they design and build the hardware in-house which comes in pre-optimized configurations similar to the proprietary model. But the core software delivering storage services is Ceph, which provides the innovation of the DIY open-source model. Because their appliances are built specifically to run Ceph, they optimize efficiency and have hardware-integrated features which decrease repair time, improving resiliency.

Ultimately the end-user must make a choice based on their own requirements. However, the storage market is inexorably moving to the software-defined model, with Open Source playing a more significant role than ever before.

More Information

Architecting IT is a brand name of Brookend Ltd, an independent consultancy, working to explain technology and business value to the end customer. This is an independently written report, with data from direct discussions with vendors and access to freely available articles and information. No vendors in this document have any editorial control over the content, except in clarifying omissions and errors.

Email: architectingit@brookend.com
Twitter: @architectingit

The Author

Chris M Evans has worked in the technology industry since 1987, starting as a systems programmer on the IBM mainframe platform. After working abroad, he co-founded an Internet-based music distribution company during the .com era, returning to consultancy in the new millennium.

Chris writes a popular blog at https://www.architecting.it/blog, co-hosts the Storage Unpacked podcast, attends many conferences and invitation-only events and can be found providing regular industry contributions through Twitter (@chrismevans) and other social media outlets.