THE BUSINESS VALUE OF PORTWORX PX ENTERPRISE

A CONTAINER-NATIVE STORAGE FABRIC FOR STATEFUL APPLICATIONS ON CONTAINER PLATFORMS

EXECUTIVE SUMMARY

Enterprises are rapidly embracing containers as a key technology enabler of cloud-native development methods. Containers unleash agility and rapid application development, increasing the rate of innovation and reducing time-to-market both for new products and high value applications lifted and shifted into containers.

To develop and operate applications in containers at scale, enterprises deploy open source container orchestrators, such as Kubernetes, both on-premises and in the cloud. While some enterprise organizations deploy the open source version of Kubernetes, many increasingly opt to deploy it via a broader container-based platform. These platforms include software products deployable across multiple public and private cloud infrastructures, such as Red Hat OpenShift and Mesosphere DC/OS. Others include managed services specific to a public cloud, such as Google Kubernetes Engine (GKE) and Amazon Elastic Container Service for Kubernetes (EKS). Deployment of a container management platform is a big step forward for enterprise development and operations teams as they gain development speed and operational maturity managing a growing container-based application portfolio.

However, constraints emerge showing container management is just a first step as enterprises deploy applications to production. While the container platforms are a natural fit for *stateless* applications, these platforms do not provide container-native persistent storage necessary to effectively operate *stateful* applications such as databases. Lack of storage availability, performance, and features that support snapshots, autoscaling, and encryption leave teams with two sub-optimal options.

Enterprises can continue to run stateful workloads outside of a container platform, but they lose out on the cloud-native automation and agility the platform provides for operational availability and scaling. This option also constrains application development speed as developers must work through slower, less automated processes in iterating on their data stores compared to their rapid development of stateless applications on a container platform.

Enterprises can instead run stateful workloads within a container platform for the benefits of automation and agility, but this puts them at high risk of recurring operational incidents from storage issues. These operational incidents risk impact to customer experience, drive up infrastructure and operations team costs, and exact an opportunity cost as developers spend time troubleshooting data store issues instead of coding. The subtlety of the technical

challenges encountered when applying common storage options to support stateful applications on container platforms surprises many enterprises.

For example, consider the use of traditional network-attached storage (NAS) with the widely popular cloud storage option of Ceph. A Ceph cluster is operated without the automation benefit of container orchestration when deployed outside of a container platform. In addition, the cluster suffers a significant performance penalty in I/O latency due to its block storage on object storage implementation and network transit latency between the application in the Kubernetes cluster and the Ceph storage nodes. As an alternative, enterprises can deploy Ceph within a Kubernetes cluster to gain the benefit of automation; however, Kubernetes is not able to coordinate placement of application containers with their data in Ceph replicas. As a result, I/O performance still suffers from the increased network transit latency versus locally attached storage as well as Ceph's block-on-object storage I/O path latency.

Enterprises commonly expect cloud block storage, such as Amazon EBS, to better enable automation and deliver the performance desired for container storage, but what they often find is an operational mismatch. Kubernetes is designed to automate rapid, repeated scheduling and replacement of containers to address cluster health issues. Cloud providers designed cloud block storage for switching volume connections across compute nodes at virtual machine (VM) provisioning rates where delays of seconds to minutes in switching connections, including with occasional failures, have normally been tolerable. At the scale of use and frequency of volume to node connection switching associated with distributed applications on container platforms, these delays and failures undermine the platform's automation value as they result in recurring operational issues.

These sub-optimal storage options for container-based stateful applications create a set of risks for enterprises deploying container platforms. These risks include potential cost increases for operations teams and reduced developer productivity – as developers spend time aiding operations instead of coding. Over-provisioning of infrastructure to work around storage availability and performance challenges can also prove to be costly. In addition, there is a platform integration risk resulting from the mismatch of cloud infrastructure platform focus on machine-centric capabilities (e.g. volume-based backups, single availability zone resources) versus container platform focus on application-centric capabilities (e.g. application-level backups, multi-availability zone scheduling).

We believe Portworx provides a compelling, container-native, persistent storage solution that enables operation of stateful applications within the same container platform as an enterprise's stateless applications. Portworx's unique storage solution delivers the availability, performance, and features necessary to minimize stateful application operations costs which, as this paper

shows, can easily exceed $1 million per year across staff and support contracts. These capabilities can also reduce compute infrastructure costs for some workloads by 40-60 percent for savings that can grow to more than $1 million per year. Most importantly, improved revenue growth from an optimal customer experience and increased developer productivity can also be achieved, although they are more difficult to quantify in impact.

This paper provides an in-depth review of these topics as indicated below.

Enterprise Challenges Using Non-Container-Native Storage 3
Problems with Traditional NAS Storage for Container Workloads 5
Problems with Local Disk Storage for Container Workloads.....6
Problems with Cloud Block Storage for Container Workloads 7
The Business Risks Of Using Non-Container-Native Storage 9
Platform Integration Risk.....9
The Cost of Over-Provisioning Infrastructure.....10
The Cost of Increased Operations and Reduced Developer Productivity 11
Portworx PX Enterprise: Container-Native Storage 11
Business Value in Reduced Operations and Infrastructure Cost 14
Business Value in Increased Revenue 16

Enterprise Challenges Using Non-Container Native Storage

The challenges with running stateful containerized applications in a performant, operationally reliable way are often not well recognized. In many cases, enterprises are still early enough in their use of containers that there is lack of awareness of the challenges they will need to overcome to run container-based stateful applications in production. As an enterprise's container use matures into adoption of a full-fledged container platform, there is a common expectation that either the platform will provide persistent storage or that existing NAS capabilities will suffice.

Lack of awareness is understandable since enterprises have been conditioned for decades to expect they can operate their data stores (e.g. databases, big data workloads, queues, key-value stores, etc.) with in-house operations expertise. This was achievable in the days of big bets on a small set of core relational databases (e.g. Oracle Database, Microsoft SQL Server, etc.). It became more of a stretch as the secondary set of more organically adopted open source databases grew to include MySQL, Redis, and MongoDB, which was common even up until recent years in cloud computing.

However, application development in new areas in the past few years has furthered a proliferation in the types of data stores to include high performance open source options like

Cassandra, Kafka, and Spark. These new fields include big data analytics, machine-learning based artificial intelligence (ML based AI) and Internet of Things (IoT). A typical enterprise is likely to run at least 10 to 20 different data store technologies as part of its application portfolio. Developing the expertise necessary to operate these many, increasingly complex data stores becomes difficult from staffing and cost perspectives. Recruiting and retaining this expertise is challenging as staff costs can exceed $1 million[1], and vendor support subscriptions to aid these teams can cost hundreds of thousands of dollars.[2] Additionally, the operational requirements of containerized data stores are different than traditional VM-based databases, further straining operations teams. Enterprises must recognize the business value of empowering developers to use the best tool for the job and consider options that help manage such a diverse set of data stores. This is critical to minimizing staff and support subscription costs as well as to avoid constraining the options of application development teams.

Enterprises turn to container platforms to deploy and manage stateful services because these platforms have captured application-specific operational best practices for operating compute resources in scripted automation conducted by the container orchestrator, whether using Kubernetes, DC/OS's Marathon, or other container orchestrators. This automation of compute resources greatly eases their organization's empowerment of developers to experiment with new data stores and easily operate those taken into production use, including deployment, scaling, updating, and maintaining availability through infrastructure failures.

While the container platform automates the management of compute, it does not automate the management of storage. Enterprises commonly expect their existing persistent storage options, such as clustered NAS solutions like Ceph or GlusterFS, or cloud block storage, such as Amazon EBS, will meet their needs for containerized applications as well. These options present challenges that tend to be subtle performance and availability issues that cause intermittent, unexpectedly painful incidents in production operation much more than show up as functional problems in development and testing environments. It is critical that enterprises understand the limitations inherent in running mission critical stateful applications using non-container optimized storage.

[1] For example, 10 database operations staff members could be necessary to manage 10 different types of data stores 24x7 (e.g. MySQL, MongoDB, Cassandra, etc.), including multiple database instances of each type and assuming each team member is cross-trained across multiple data store types. At $150,000 in average fully loaded (compensation plus benefits, office space, equipment, etc.) annual cost per team member, the total annual cost for the team would be $1,500,000.

[2] Software support subscriptions for data stores commonly cost $1,000 to $10,000 per year depending on type and scale of nodes supported. An example cost would be 10 types of data stores with an average of five database instances per type over an average of six nodes per database instance at an average of $1,000 per node, resulting in $300,000 per year in subscription support cost.

Problems with Traditional NAS Storage for Container Workloads

Some enterprises try to use traditional storage they have already deployed as the persistent storage for containerized applications. For example, Ceph is considered a top clustered NAS option for block storage for containers. Ceph is a relatively mature open source storage technology that delivers on features, including encryption and replication for availability.

Enterprises typically deploy Ceph separate from compute on storage-specific server nodes to supply high density storage and support Ceph's significant CPU resource requirements. The CPU requirements are, in part, driven by its multi-stage data path. Ceph must conduct a block to object mapping step as it implements block storage by mapping it into object storage. In addition, Ceph uses the Linux file system to handle reads and writes to disk, so its file system hands off I/O operations to the Linux file system instead of handling them directly.

This data path complexity results in latencies that can quickly run from single-digit into double-digit milliseconds, especially when using synchronous replication across nodes. This delivers average and worst-case latencies on the upper end of what high performance databases under high production application load can tolerate. As Ali Zaidi, director of Platform Services at DreamWorks Animation, explains, "Before deploying Portworx, we struggled to find a data services layer that allowed us to take advantage of Kubernetes automation without using a centralized storage solution like a SAN or Ceph, an anti-pattern for most of our databases."

Beyond the performance challenges, the separate deployment of storage from the container cluster requires enterprise operations teams to provision and manage the storage themselves. This forgoes the automation benefit of the container platform in common operations such as upgrades, scaling, node failures, and network failures, which introduces infrastructure and operational resource complexity and costs that enterprises would much rather avoid.

Attempts exist, for example via the Rook project, to bridge NAS software like Ceph into a more container-native model by deploying the storage cluster using block volumes of the container cluster nodes (NAS or local disk). This approach can address operations pain by utilizing the container platform to manage some of the storage cluster operations; however, the performance benefit is more limited. The solution lacks container cluster scheduler integration necessary to land replacement containers (i.e. after failures) on cluster nodes containing the storage volumes with their data. This results in varied I/O latency per container depending on the distance between the container and its data within the cluster.

Even assuming the Rook project adds scheduler integration in the future, the project will improve the I/O latency for reads by removing the network transit latency but still leave the single-digit millisecond level latency impact from the complexity of Ceph's I/O mapping steps

versus less than 1 millisecond latencies typical of I/O direct to local solid-state drives (SSDs). It would not improve write latency since those operations must still transit the network for synchronous replication, including across Availability Zones (AZs) or their equivalents in multiple on-premises data centers designed for high availability applications.

Enterprises sometimes consider GlusterFS as an alternative to Ceph for container-native NAS, but GlusterFS is designed for file storage, not as a block store for databases. It can satisfy file sharing when monolithic applications are refactored to pull file data and database components out from the stateless applications themselves, but as its documentation explains, "Gluster does not support so called 'structured data', meaning live, SQL databases".[3] GlusterFS can also be sub-optimal for data-intensive data store applications outside of databases for various architectural reasons. For example, it scales sub-optimally with high volume counts per node because each GlusterFS volume has its own process consuming memory and CPU resources, resulting in linear resource consumption with volume count.

Problems with Local Disk Storage for Container Workloads

An alternative approach to clustered NAS is to use the local disks on each server of a container cluster. This can be useful when optimizing heavily for high performance via low I/O latency and high I/O operations per second (IOPS). However, the first challenge with this option is that it requires custom feature implementation or additional software to address lack of encryption, snapshots, replication, class of service, and other common enterprise features.

The second challenge is the performance risk to critical applications during recovery from failures or updates that result in scheduling replacement containers on new nodes. When using local storage, many modern databases will recover by recreating the lost dataset on the new node. This "application level replication" has potentially significant impact to performance during the restoration of the data to the new container node's local disks, risking transient impact to the customer experience, which can be difficult to monitor, measure, and mitigate.

Increasing the replication count in the data store application with additional compute nodes and/or more resources provisioned per node (for example, network I/O bandwidth) can help mitigate the performance impact during recovery. However, each option involves multiplying infrastructure costs and increasing operational costs of managing what could be a 50-100 percent increase in compute resources. This also typically requires an operationally intensive approach to the handling of upgrades, failures, and scaling to maintain the data store's performance. Therefore, local disk is normally an option of last resort.

[3] Per Gluster's documentation at https://docs.gluster.org/en/v3/Install-Guide/Overview

Problems with Cloud Block Storage for Container Workloads

To address the challenges of clustered NAS and local disk use, enterprises often expect to be able to rely on persistent block storage from their cloud infrastructure such as Elastic Block Storage (EBS) volumes on Amazon Web Services (AWS). This option is programmatically straightforward, provides strong per-volume feature support (encryption, snapshots, etc.), and can deliver consistent performance, though at a latency in single-digit milliseconds.

This can be an acceptable solution for statically provisioned databases when the application performance impact during recovery and operational work is tolerable after each compute node – or storage volume failure in cases of compute detachment failures or lost volumes per EBS's 0.1-0.2% annual failure rate (AFR), for example. This can apply to a small scale relational database that is rarely updated, not scaled up or down elastically, and whose performance is not critical to business delivery of customer experience or internal productivity. However, this normally represents only a fraction of an enterprise's production data store footprint.

A greater need for consistent performance and operational automation exists for the larger enterprise footprint of relational databases operated at greater scale and in situations critical to business delivery. This is particularly true when operating data stores that have more complex updating schemes and/or employ elastic scaling to address difficult to predict application workload peaks while optimizing for cost, such as NoSQL stores like Elasticsearch and Kafka. This dynamic provisioning of containers and the scale of containers subject to infrastructure failures multiplies the frequent provisioning and movement of cloud block storage volumes, a time consuming and fragile process.

Shifting the storage volume connection for each replacement container after an infrastructure failure or data store application update involves:

- Unmounting the storage volume(s) from the impacted container(s);
- Detaching the volume(s) from the impacted compute instance(s);
- Reattaching the volume(s) to the new compute instance(s); and
- Mounting the volume(s) to the new container(s).

Note, these actions are required for each scheduled replacement container. (Newly scheduled containers for scale-up events only require attachment and mounting.) Some operational issues, for example partial network failures (e.g. network partitions) and cluster affinity setting updates, can result in the scheduling of many replacement containers at once. Each of these "day 2" operational issues can be low in frequency, but issues can easily result in one to two orders of magnitude in aggregate more storage volume shifts than in the case of a simple

relational data store running on VMs when each involves a multi-step impact to the storage volume(s) across all impacted compute instances.

This increase in volume shifts across compute nodes would be acceptable if these actions were reliably flawless and occurring within fractions of a second. Instead, these actions involve enough exposure to delays and failures that they result in regularly occurring operational issues when conducted at the frequency and scale of use with distributed applications on containers.

Delays in detaching from the old node and attaching to the new node can vary from seconds to full minutes. An example failure occurs when the volume detachment application programming interface (API) call fails (the volume is “stuck”). Force detaching a stuck volume involves risk of data corruption or data loss on top of the risk of volume loss from the block storage service’s baseline AFR. These risks exist regardless of whether the containers are operated by a container-based cloud platform (e.g. Red Hat OpenShift, Mesosphere DC/OS, etc.) or a container-based cloud service (e.g. AWS EKS, GKE from Google Cloud, Microsoft AKS, etc.), because they all depend on the infrastructure platform’s block storage capability.

It can be easier to discard a stuck or lost volume and instead restore the data at the application level from the rest of the cluster just as when operating on bare local disks when running the common open source data stores with distributed architectures offering application level data redundancy. This approach avoids having to scan the volume for corruption or restoring from backup then replaying a transaction log. This is also true when the four unmount, de-attach, attach, and mount steps are delayed.

However, avoiding stuck volumes by relying on application level replication can have a service level agreement (SLA)-impacting performance cost from the perspective of the enterprise applications depending on the stateful application. Imagine a Cassandra or MongoDB database stalling its response for tens of seconds or even minutes. This delay risks impact to the customer’s application experience and possibly even the stability of the data store and its dependent applications if they cannot gracefully handle a significant resulting back up in request queues for such periods. While it can be easiest to discard and rebuild the volume via application level replication, this path results in the same challenges as in the local disk case of risking performance impact to the data store cluster and forces consideration of over-provisioning infrastructure resources.

THE BUSINESS RISKS OF USING NON-CONTAINER NATIVE STORAGE

Enterprises relying on these sub-optimal persistent storage options for stateful applications operated in containers can see various types of business impact including platform integration risk; the cost of over-provisioning infrastructure; the cost of increased operations; and reduced developer productivity.

Platform Integration Risk

The first business risk to consider given the technical strengths and weaknesses of the above options is the integration risk they imply for the breadth of stateful applications addressable with a container platform. The limitations above were specific to the storage architecture. There are two additional types of platform integration risks not addressed well by any of those options: application-level consistency in backups and storage volume alignment with application container scheduling across AZs.

The clustered NAS and cloud block storage options allow taking snapshots of storage volumes but require the enterprise operations team to capture the full set of point-in-time snapshots when recovering from a database backup. This makes taking application consistent snapshots a challenge and risks unsuccessful recovery if inconsistent snapshots lead to data corruption when trying to recover.

To understand why this is more of a challenge than in a VM-based architecture, it is important to understand that cloud-native applications operated in container clusters are often composed of many microservices (i.e. component applications) that are distributed across multiple containers. If an enterprise can't capture the entire distributed state in an application-consistent method at a single point in time, then integration of the container platform with supporting storage is difficult to deliver with an effective backup / recovery and disaster recovery (DR) capability from a performance (i.e., time to recovery) and operational workload perspective.

This leaves application owners to decide whether the integration risk of a challenging recovery in the case of a large-scale infrastructure failure (e.g. multiple AZs or a region) is worth the automation benefits of running their stateful applications in the container cluster. If the value does not justify the risk, it reduces the value of the container platform because it manages fewer applications.

The second type of platform integration risk concerns availability feature support. Cloud providers guide customer use of multiple AZs for optimal application availability based on their data center fault domains, so applications operated in container clusters should be deployed across AZs. As a result, the scheduling of new and replacement containers on compute nodes

occurs dynamically across AZ boundaries. The challenge of maintaining storage volume scheduling (i.e. placement) across AZs in alignment with changes in application container placement is daunting and typically not feasible without container-native storage, resulting in a significant application availability risk. For example, moving clustered NAS volumes deployed on cloud infrastructure between AZs corresponding to each replacement container scheduled to a different AZ would be a significant operational effort in custom automation to develop and operationally support.

Similarly, cloud providers specifically define block storage volumes as confined to a single AZ, so moving them between AZs requires development of custom automation triggering volume to volume copying or re-creating a volume after taking a snapshot. Latency in the scheduled replacement container's availability to the data store cluster is also increased while waiting on the movement of the storage volume in both the clustered NAS and cloud block storage cases.

It is tempting to expect this to be addressable by forcing scheduling of containers to be within the same AZ as the storage volumes on which they depend. In fact, this can address individual compute and storage resource failures but leaves applications vulnerable to large-scale resource failures such as network partitions and full AZ service outages where replacement compute and storage must be re-provisioned in another AZ. These incidents are infrequent, but each of the top cloud providers has had AZ-level network and service outages and warns customers to design with an expectation that those large-scale incidents will recur. Failing to architect for them with container-based applications risks eventual mass application outages.

The Cost of Over-Provisioning Infrastructure

The infrastructure cost risk – whether in fixed, private resources or on-demand, public cloud resources – results from attempts to mitigate the application performance issues described above. The simplest way application operators mitigate risk is to avoid adoption of a container platform for operating stateful applications. This leaves the applications running in VMs at a larger compute cost due to the inefficiency of compute resource utilization when compared to containers.

For teams who do run their stateful applications on the container platform, they often solve performance issues that accompany common failure modes by creating a greater number of nodes so that in the case of failures, each data set that needs to be recovered is smaller. This minimizes the I/O thrashing that occurs when distributed data stores such as Cassandra, Elasticsearch, or Kafka rebuild the data for a node. However, it comes at a significant infrastructure cost since scaling from four to six application level nodes, for example, increases infrastructure costs by 50 percent.

An additional, more subtle case of infrastructure over-provisioning risk exists when application operators over-provision the number of containers to manage availability risk instead of performance. For example, a MongoDB database supporting a low scale application workload may be able to service all requests via a single container, but the operator may over-provision resources by running multiple replica containers to ensure performance headroom for quick scaling and availability in case of an AZ level failure. Operators may take this approach due to a lack of storage replication across AZs for availability and allowing ease of quick scale-out. Each potential single node data store deployment deployed instead as two or three nodes is 200 or 300 percent higher compute cost across each low scale workload database.

The Cost of Increased Operations and Reduced Developer Productivity

Operations cost risk exists when application operators either stick with more manual operation of their stateful applications in VMs or provision more replicas requiring operation of a larger infrastructure footprint. The more data stores operators keep off the container platform, the more the enterprise loses out on the benefits of container-based operations automation.

Container platform-based automation not only automates operation of stateful applications, it absolves the development and operations teams from having to develop expertise in each type of data store they deploy. As mentioned previously, the operations staff and support costs can run well over $1 million per year for a set of 10 different data stores – a situation that is both costly and impractical for many enterprises.

As developers experiment with a wider variety of data stores when developing new applications, particularly as they are empowered with open source options that accelerate application creation, lack of ability to operate the data stores constrains their ability to move fast. Once deploying a new data store, developers are more frequently pulled into operational incidents to fight availability and/or performance fires due to issues with data stores instead of creating value coding. Container platforms, particularly when combined with container-native storage, encode operational best practices in automated handling of these data stores in deployment, updates, scaling, and failure handling. This greatly reduces the operational burden not just on the operations team but on the development team to improve their productivity as well.

PORTWORX PX ENTERPRISE: CONTAINER-NATIVE STORAGE

What would a block storage capability offer if purpose-built for container-based cloud platforms? Ideally, it would:

- deliver immediate availability of dynamically provisioned storage volumes;
- synchronously replicate copies of data spread across the cluster in case of failures;
- guarantee hyperconvergence of containers and their volumes on the same host, even after recovery from failure;
- deliver performance approaching that of the native cloud block storage or local disks;
- deliver multi-cloud portability across public and private cloud infrastructures; and
- deliver application-specific point-in-time snapshots along with typical persistent storage features such as encryption.

Past attempts to deliver these capabilities in projects such as Flocker, RexRay, and others succeeded on some aspects, but sacrificed on others. They struggled to offer reliable, immediate availability of pre-populated storage volumes for each new data store container.

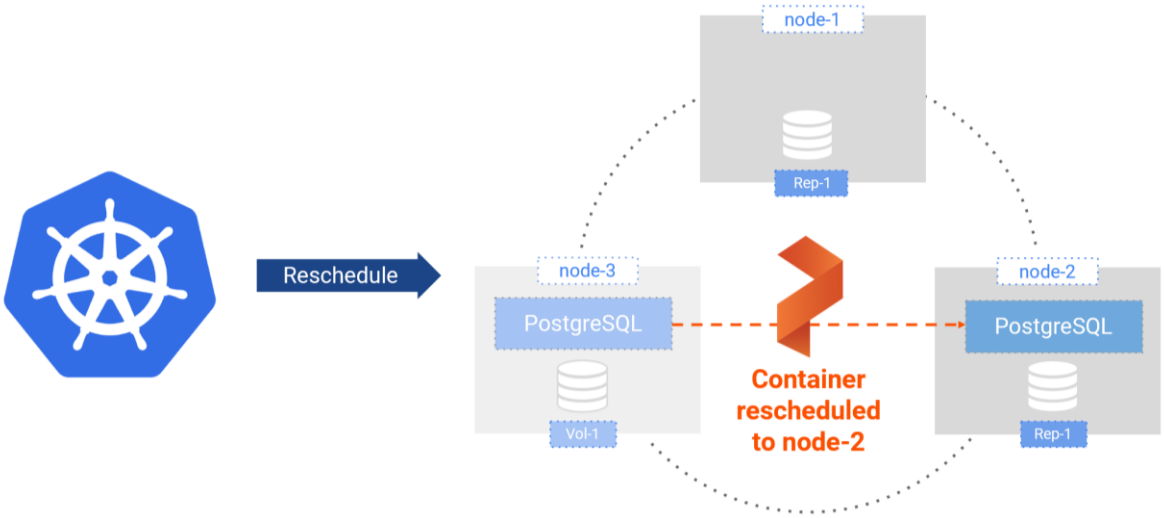
Portworx takes a different approach by providing a storage fabric that synchronously replicates volumes across container cluster nodes to eliminate impact from volume delays or failures and provides immediate pre-populated volumes to new containers. This results in an elegant solution that:

- fully abstracts underlying cloud block storage or local disks to address failure modes;
- automates volume replication to pre-populate storage volumes available immediately to newly scheduled containers;
- automates volume replication across AZs to enable cross-AZ application container scheduling and ensure high availability of the volume data;
- delivers a software-defined storage interface for portable programmatic control across multiple public and private cloud infrastructures; and
- utilizes a lightweight modification to the Linux file system[4] to enable implementation of replication and storage features, such as encryption and application-specific snapshots.

Portworx enables the Kubernetes cluster control system to know where to place new and replacement containers with their associated volume replica to achieve immediate data availability. Only the volume mount step is needed to enable data availability since the volume is already attached to its container cluster node. Figures 1 and 2 illustrate the value of the Portworx storage fabric's replication model that enables swift Kubernetes cluster recovery from node failures for stateful applications, such as PostgreSQL and Cassandra.

[4] The modification implements a virtual block device layer in the Linux namespace on each node via the Linux File system in Userspace (FUSE) interface. Portworx implements its features by operating on the virtual block device layer.

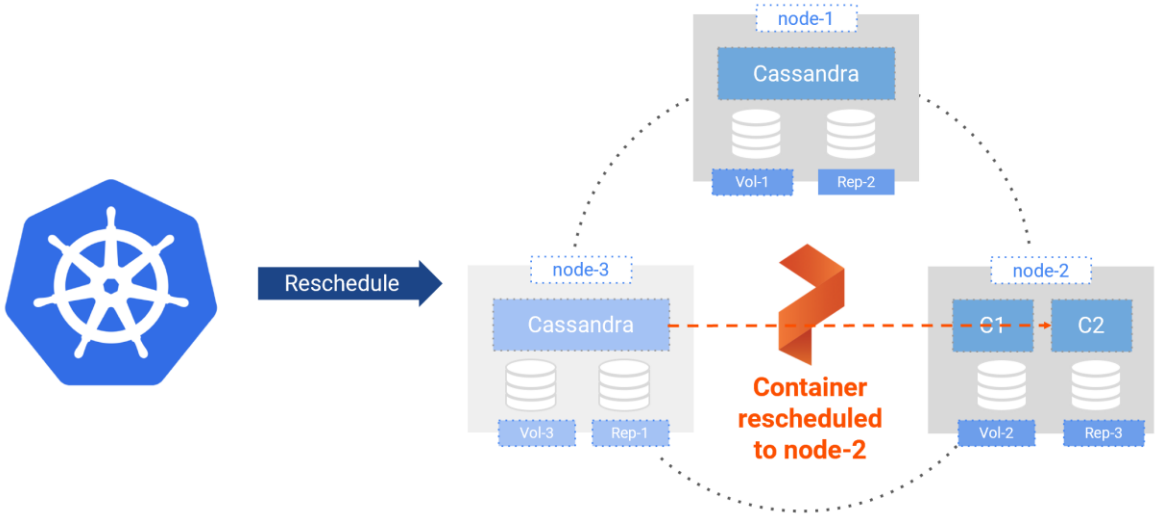
FIGURE 1: KUBERNETES ADDRESSES NODE FAILURE FOR POSTGRESQL



Source: Portworx

Kubernetes detects the failure of node-3 where the single PostgreSQL application container was running. Kubernetes reschedules the PostgreSQL container to node-2 based on guidance from Portworx on where existing replicas of the container's Vol-1 data are immediately available (node-1 and node-2). Portworx speeds up the database's readiness to serve requests by avoiding the delays of volume attachment or data replication.

FIGURE 2: KUBERNETES ADDRESSES NODE FAILURE FOR CASSANDRA



Source: Portworx

Kubernetes detects the failure of node-3 where a Cassandra application container was running. Kubernetes reschedules the Cassandra container to node-2 based on guidance from Portworx on where the existing replica of its Vol-3 data is immediately available. This dramatically shortens Cassandra's bootstrapping step to move immediately into the cluster node repair step for recovering quickly to full cluster capacity.

Replicas can be deployed across AZs or cloud infrastructures allowing high availability application deployment, and application-specific snapshots allow for data store capture and movement for ease of cross-cloud portability. The storage fabric implements the features of application-specific snapshots, encryption, etc. The control system, file system, and Kubernetes scheduler integration components are implemented in a pluggable model that provides customers with the freedom to replace Portworx's storage fabric implementation in the future.

Business Value in Reduced Operations and Infrastructure Cost

Portworx PX Enterprise delivers business value through both reduced costs and potential for higher revenue. First, operations productivity improvement leads to reduced costs. Achieving operational comfort in running stateful applications on the container platform saves operators significant time – as they no longer have to manage manual changes to individual deployed containers or VMs.

With Portworx enabling fast, reliable storage automation, increased use of the container platform reduces the operations work, mitigating business application performance and customer experience impact during application updates, scaling, and infrastructure failures.

This also allows stateful application operators – whether business application developers or operations staff – to execute full lifecycle management of the application in deploying, updating, taking snapshots, etc. – without the need for traditional specialized storage roles in IT to support them. Traditional storage administration roles can then be repurposed into broader operations or development roles.

Portworx customers cite reduced operations cost as a top benefit. Michael Wilmes, software architect for airline IT services at Lufthansa Systems, sees fewer moving parts resulting in less failover work. "Once Portworx was set up in the Docker cluster, the overhead of managing stateful containers became virtually zero," said Wilmes. For Simon Boisen, CTO of online university study platform Lix, the benefit is clear. "Portworx saves us time. Time is the most important resource in a startup."

Second, operating stateful applications on a container platform using Portworx enables reduced infrastructure cost in multiple ways. Stateful application operators no longer need to over-provision compute and storage resources for their data store. They trust their data store can scale elastically to address peaks in load from dependent business applications and deliver sustained performance during recovery from container failover. For example, a relational database may need a set of four read slaves to service peak read-heavy load at times, but normally only needs two read slaves. Portworx replication allows such a fast, reliable

scaling of read slaves that the operator no longer needs to run the maximum read slave count for periods well under peak load.

Enterprises can save $200 per month[5] when they don't have to run two read slaves, which is a financial savings that can be achieved for even a small relational database resource footprint. This is similarly true with MongoDB and other NoSQL databases, when servicing steady-state load with fewer provisioned replicas per shard. With MongoDB consuming greater compute resources per node at a higher cost than a small relational database, the cost savings of reducing the deployment size by just one replica can exceed $1000 per month[6] per database instance.

Beco, a workforce space analytics provider, found that Portworx reduced I/O flooding during Kafka recovery and enabled failover at least twice as fast as Kafka's application-level replication-based recovery. This is a good example benefit of the sustained performance during recovery from container failover. This improvement enabled Beco to reduce its Kafka broker count from five to three for a 40 percent savings in compute cost. For an organization scaling to 10 types of databases at just 10 instances of each type, a savings of $200 - $1,000 per instance per month represents $240,000 - $1,200,000 in potential savings per year.

Portworx can provide additional cost savings in storage via thin provisioning of storage volumes. Portworx allocates virtual storage to containers that maps to over-subscribed physical storage, which allows provisioning additional storage only when it will be utilized. For example, Portworx can provide four 100 gigabytes (GB) virtual storage volumes based on a single 200 GB physical cloud block storage volume at 2X over-subscription. As the four virtual volumes fully utilize the 200 GB physical space, Portworx can allocate more physical space up to the full 400 GB of the virtual volumes. They provide the additional physical space by automating provisioning of additional volumes or resizing existing volumes (e.g. via elastic volume resizing of cloud block storage).

Consider an example of a single MongoDB instance with three read slaves. The application team may anticipate needing up to 500 GB of SSD cloud block storage per node. This would require 2 terabytes (TB) overall. However, the database can begin with just 1 TB for the master

[5] Cost determined based on two read slaves consuming Amazon EC2 m5.xlarge VMs with 16 GB of memory in U.S. East Ohio On-Demand at $0.192 per hour for a total of $281.09 per month for two slaves.
[6] Cost determined based on MongoDB node sizes recommended by AWS for Amazon EC2 with at least 55 GB of memory (r4.2xlarge or c5.9xlarge or larger) in U.S. East Ohio On-Demand at a cost of at least $0.532 per hour for a monthly cost of $389 per replica node. Cutting the replica count by one across three or more shards results in a minimum of $1,167 per month in savings.

and three slaves. This is a savings of $100 per month[7] on the second 1 TB that is not provisioned until needed. While the storage cost savings opportunity is less than the compute savings opportunity per data store instance, the storage savings opportunity can be more consistently available across types of data stores and VMs. The storage savings opportunity can still accumulate into tens to hundreds of thousands of dollars in savings.

Business Value in Increased Revenue

Portworx solutions can provide a more consistent customer experience and improved business productivity in enterprise application development. When a data store is performing more consistently, customers will have a better experience that is likely to drive higher engagement for service adoption or product purchase, which potentially increases revenue. The fast failover of stateful application containers via Portworx storage is critical to mitigating data store performance impact during recovery to preserve consistency in the customer experience. Venkatesh Sivasubramanian, data platform lead, for GE's Predix platform, stated that after container failovers, their customers are "able to go back and look at the system as if there was no change on the infrastructure".

The benefit of improved business productivity in enterprise application development means more developer time spent coding instead of fire-fighting operational issues. Hugo Claria, hosting division head, at IT services provider, Naitways, explains, "The real business value of Portworx is time. We can get to market faster with Portworx than we could if we implemented everything ourselves." As Jeffrey Zampieron, CTO of Beco, describes, "Portworx is a battle-proven, highly reliable storage solution that scales with our increasing demand and enables our development team to focus less on storage and more on building new software services."

Call To Action

Infrastructure, operations, and application development leaders can benefit from early recognition that the stateful applications on which their rapidly developed stateless applications depend will be best deployed and managed in the same container platform. Those recognizing this but delaying prioritization of adoption of a strongly supportive container-native storage solution, are missing out on substantial opportunities for cost reduction and improved revenue growth.

Moor Insights & Strategy (MI&S) recommends that infrastructure and operations (I&O) leaders – as the traditional organizational thought leaders on opportunities for technology-enabled cost

[7] Savings calculated based on Amazon EBS General Purpose SSD volume pricing of $0.10 per GB per month applied to 1 TB of storage for $100 per month in savings.

efficiency and growth – surface the business value of adopting a container-native storage solution for their container platform. It is important that I&O leaders advocate for capabilities for the platform to support optimal long-term stateful application deployment to fully benefit from its value.

MI&S recommends that I&O leaders consider Portworx PX Enterprise as a container-native storage fabric for their container platform. Portworx's unique offering enables storage availability, performance, and features necessary to operate a full portfolio of stateful and stateless applications on the same container platform. As a result, enterprises can reduce operations and compute infrastructure costs, improve revenue growth through optimized customer experience, and increase developer productivity.

IMPORTANT INFORMATION ABOUT THIS PAPER

AUTHOR
Rhett Dillingham, Vice President & Senior Analyst at Moor Insights & Strategy

PUBLISHER
Patrick Moorhead, Founder, President, & Principal Analyst at Moor Insights & Strategy

INQUIRIES
Contact us if you would like to discuss this report, and Moor Insights & Strategy will respond promptly.

CITATIONS
This paper can be cited by accredited press and analysts but must be cited in-context, displaying author's name, author's title, and "Moor Insights & Strategy". Non-press and non-analysts must receive prior written permission by Moor Insights & Strategy for any citations.

LICENSING

DISCLOSURES
This paper was commissioned by Portworx. Moor Insights & Strategy provides research, analysis, advising, and consulting to many high-tech companies mentioned in this paper. No employees at the firm hold any equity positions with any companies cited in this document.

DISCLAIMER
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. Moor Insights & Strategy disclaims all warranties as to the accuracy, completeness, or adequacy of such information and shall have no liability for errors, omissions, or inadequacies in such information. This document consists of the opinions of Moor Insights & Strategy and should not be construed as statements of fact. The opinions expressed herein are subject to change without notice.

Moor Insights & Strategy provides forecasts and forward-looking statements as directional indicators and not as precise predictions of future events. While our forecasts and forward-looking statements represent our current judgment on what the future holds, they are subject to risks and uncertainties that could cause actual results to differ materially. You are cautioned not to place undue reliance on these forecasts and forward-looking statements, which reflect our opinions only as of the date of publication for this document. Please keep in mind that we are not obligating ourselves to revise or publicly release the results of any revision to these forecasts and forward-looking statements in light of new information or future events.