solarwinds

eBOOK

The Unifying Force of Data

Bringing DBAs and Storage Admins Together solarwinds

Table of Contents

Introduction

IT managers look at the relationship between DBAs and storage administrators and scratch their heads. Of all of the disciplines in the data center, how is it that DBAs and storage admins don't understand each other, let alone get along? Both disciplines care deeply about data. They both want to understand the nature of the data so they can understand its impact on business requirements and operations. However, they approach data management from two completely different directions.

The goal of this eBook is to provide a practical roadmap to improving collaboration between these two groups. We'll do this by restating the problem, establishing a common list of metrics to work toward, and suggesting operational improvements.

Strangers in a Strange Land

DBAs and storage admins are sovereign within their own domains. They know the landscape, speak the language, and have a reasonable idea of the issues they face on a daily basis. But when they journey out into each other's domains, they become strangers in a strange land. They both have the same goals; they just approach them differently.

This problem is as old as IT, as DBAs and infrastructure types have always been at odds. System administrators have always wanted to blame performance problems on anything but the server or storage in their charge. "The problem is under the floor," was an old phrase in the data center, meaning it was a network problem and not a server/storage problem. DBAs, on the other hand, want to blame performance problems on anything but the code. It's easier to blame something that doesn't create additional work for you. The reality is that both DBAs and infrastructure administrators can be right; sometimes poor infrastructure can slow down good code, and sometimes bad code can slow down the best infrastructure.

SAME GOALS, DIFFERENT PRIORITIES

From back-office services like IT and HR to frontline functions such as customer service and sales, all applications have the same end goal: accomplish the mission. Every organization from a government agency, non-profit, to the traditional for-profit enterprise have stated missions. The mission may differ from organization to organization, but the goal within every department remains to reach the overall organization's goals.

The advantages to accomplishing the mission are obvious. For most reading this eBook, that means personal satisfaction from accomplishing some social good or simply supporting our families. Without accomplishing the organization's mission, there is no continuing concern. If the funding for your organization comes from donations, donors stop giving. If funding comes from consumers purchasing products, consumers stop purchasing products. If you are a government agency and don't perform your duties… well, at least for most organizations, you must meet the organization's mission to continue as a going concern.

However, this drive to accomplish the mission doubles as a stress source for much of IT. If the server that provides the customer relationship management (CRM) software experiences performance issues, orders can't be processed or donations can't be pledged. The executive over at customer service will take issue with IT not supporting the mission.

Taking that use case further, the executive responsible for customer service will in turn call the director of IT to ask why IT is inhibiting the mission. I'm sure most readers have heard the phase, "it rolls downhill." This eBook is about the bottom of that hill: storage.

The goal of both the DBA and the administrator is to meet the needs of the mission. Once IT takes its focus off the mission, it becomes only about managing disk pools or ensuring logs are written. If there ever were a time when technologists could only concern themselves with technology, those days have long passed. The technology is now only there as a tool to support the mission.

ADD DIGITAL TRANSFORMATION

To bring the challenges to life, we'll take a real-world application design and apply a typical set of operational challenges faced by the teams, such as an Enterprise Resources Planning (ERP) application called SAP® ERP Central Component (ECC). As the name indicates, SAP ECC is the primary component of a typical mission-critical workload. ECC comprises a set of application servers and a database. SAP is a great use case, as ECC supports a wide range of databases from Microsoft® SQL Server®, Oracle® DB, and DB2® running on an array of operating systems and processor platforms.

You may think there's not much digital transformation in a traditional ERP such as SAP ECC, but this couldn't be further from the truth. SAP executives have appeared at the most cutting-edge conferences. Developers are building front-end applications using platforms like Fiori® that integrate with SAP ECC.

An example is a new business initiative to increase the number of relationships with every customer. A financial services customer may want to encourage customers with a brokerage account to purchase wealth management services. As a result, each customer representative must know the status of each account and existing services of a customer when they contact the call center.

What happens when the executive responsible for customer relationship sponsors a new CRM application built using Fiori? The CRM has hooks back to SAP ECC. IT must now operate at two speeds. One speed ensures the system of record is reliable and performs. The other speed ensures rapid application development cycles while performing at the speed of business—all the while, not impacting the system of record or mission-critical back office processes.

The end result? Even more pressure on the DBA and storage administrator to work well together.

DBA Priorities

DBAs tend to focus mainly on customer satisfaction and application efficiency. The CRM use case provides insights into this relationship. Once IT is contacted as a result of a slow CRM application, the operations team begins troubleshooting. DBAs are normally part of the rapid response team assembled to resolve the performance issue. The DBA is close to the pain point inhibiting the mission.

With a deeper understanding of the business process, the DBA knows that if the website is slow because the database behind the website is slow, then it will take longer for customers to purchase their products. In fact, some customers may get frustrated with how long a purchase is taking and decide to go to a competitor. When that happens, the company loses more than the revenue from that sale; it potentially loses future revenue from the customer. A good DBA knows a poorly performing database can be an Achilles' heel to the company's website. A slow database isn't just an inconvenience, it's a vital business concern.

The same is true of databases that support company processes, such as customer service. If a customer calls in, it should take only a few seconds to match that customer to their record so that a phone support person can begin supporting them. Databases can also be used to drive automated support information systems, including websites and automated phone systems. Poorly performing databases behind these critical systems can remove any loyalty a customer has with your product or company, working against the goals discussed above.

Couple these challenges with the digital transformation that's happening in many enterprises and the demands increase. Business leaders want to become more agile and remove friction between the back office functions, such as distribution and the end consumer of a product. By moving business processes closer to the end consumer, the pressures on the backend become more pronounced.

Suffice it to say, any good DBA knows their databases must perform well, and they're often told their databases aren't performing fast enough. Application managers are constantly wanting to make improvements to the front-end systems that use the databases, sometimes using agile methods. While laudable, these often increase the size and quantity of records, as well as the complexity and number of queries against the company's databases. This means a DBA is playing constant catch up between database performance and application requirements.

Storage Admin Priorities

Storage admins are also concerned about performance, but their primary concern is cost. DBAs are their customers, but the CFO and CIO are their bosses. Anyone familiar with IT infrastructure costs know storage costs can be the single biggest component of an IT budget. These costs come from multiple things, including the cost of the storage itself, the cost to manage it, and the cost to back it up. Storage admins are continually reminded by upper management that storage is the fastest-rising cost in the data center. So, cost tends to be the storage administrator's focus.

When a DBA comes to them and asks for another 20 TB of storage, they immediately ask themselves how much that DBA used of the last 20 TB of storage they gave them. This is especially true if they are able to track actual usage versus requested storage. If they are able to track the number of IOPs a given application requests, they will also be running that through their head as the DBA requests storage capable of a certain number of IOPs.

Storage admins know DBAs tend to ask for much more than they need, and for good reason. It can take a very long time to provision additional storage or to change the characteristics of storage that is already provisioned. Overprovisioning represents the dance between the DBA and storage admins. DBAs will commonly overprovision storage within the database to mitigate the risk of running out of physical space on the file system hosting database.

But in a self-perpetuating cycle, the storage admin becomes difficult to work with because they know the DBA tends to ask for more than they need and feel the need to counteract that by making sure these are “real” requirements. Then the DBA asks for more than they need because the storage admins are difficult to work with.

Storage admins want to give the application owners what their application really needs, but they know they have to do that in a way that reduces cost as much as possible. First, they must reduce hard costs by eliminating storage that is allocated but not used. Second, they must reduce soft costs by eliminating wasted effort. To further complicate matters, which of these two costs is more important to them will depend on what their upper management is telling them at any given time.

But it’s important to understand that while the storage admin may get a reputation of being “Dr. No,” he or she is doing that for the same reasons the DBA is pushing for more and faster storage: to make the company more profitable. Both the DBA and the storage admin want the same thing; they just go about it differently. Maybe if each of them understood that, things would be a little better.

Storage Challenges

There are three primary challenges with storage: capacity, performance, and cost. A storage manager’s job is to make sure there is enough storage that performs fast enough for the applications that need it—without wasting money to do it.

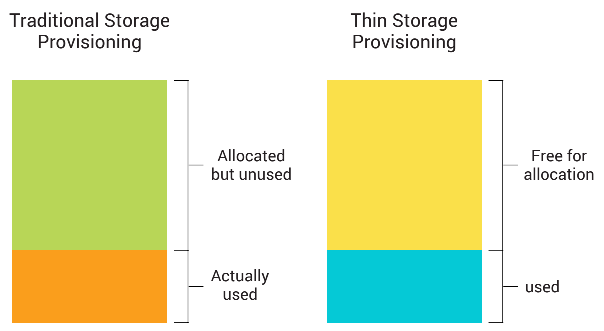
CAPACITY

Database capacity seems like a fairly straightforward measurement from both the storage perspective and the database. Without getting into the details into the initial storage request process, storage provisioning remains a pain point between the DBA and the storage admin. Unlike most applications, databases consume all provisioned space on disk even if the tables aren’t full with data. Oddly, this represents one of the basic cultural differences between the DBA and other application owners.

The disconnect in used database table capacity vs. actual disk usage causes untold numbers of priority-1 (P1) tickets. Some unexpected change in the business or a bad script was run against the production database and the database tablesspace fills. As a result, the DBA opens a P1 ticket to expand storage. The question is, will there be enough of the same production class storage available to fulfill the request? Even more importantly, how do you prevent this scenario from even happening?

When speaking of capacity, we must also speak of the idea of thin provisioning, which is a technology that allows a storage administrator to allocate all the storage an application could eventually need without actually having that much storage available at the time. Without thin provisioning, there are two capacity metrics: how much storage has been allocated to an application, and how much storage that application actually used. This leads to the scenario described above.

THIN PROVISIONING SIMPLIFIED!



One potential solution is a thin provisioning system that would give every database its "dream" amount of storage, while allocating blocks as the database actually uses them. The challenge then becomes watching for race conditions, where too many applications start using too much storage all at once, as a basic aspect of thin provisioning is that you allocate way more disk space than you actually have. This gives you another metric to track.

Thin provisioning on spinning media comes with another issue for a storage admin to consider: fragmentation. Instead of having a contiguous disk in a typical storage allocation, thin provisioning a database server will cause the database files themselves to grow in small increments. This fragmentation must be tracked for legacy storage. On newer all-flash arrays, fragmentation is no longer an issue, but managing used storage versus what could be used remains a concern.

Whether using traditional disk allocation or thin provisioning, the key to preventing problems is monitoring actual usage. Storage should always be able to be bought in advance far before it's actually needed.

PERFORMANCE

A database performance issue can manifest in several ways. An example would be manufacturing work orders not printing from a manufacturing resource planning (MRP) system connected to SAP. The MRP system may rely on a batch process that runs in SAP ECC. After investigating, the SAP BASIS team determines the application isn't performing due to a failed batch job that's timing out and failing. But why is it failing?

The DBA suspects the culprit is under-performing storage. Now the real work begins. In an application as complex as SAP, answering that question can be quite difficult.

What data points can the DBA feed to storage administrator to isolate the storage issue? If it is a storage issue, what's the best way to resolve the issue? How do you prevent a similar problem occurring in the future, or at least proactively detect similar occurrences before a P1 is created? You must answer these questions in order to solve this problem.

COST

Budgets are not unlimited. If they were, everyone would probably buy the best all-flash array they could get and put everything there. So, you could easily solve the capacity and performance challenges if you had an unlimited budget—but you don't.

This means most storage administrators find themselves looking for ways to get additional capacity or performance without paying for it. They are constantly looking for storage that isn't being used so it can be reallocated to applications that need it. They are also on the lookout for storage with performance capabilities that are not being fully utilized. If there's an application with a performance profile more appropriate for rotational disk, then they'd like to find that application and move it there. That frees up space in the all-flash array, which is the most expensive resource. By shrinking a volume that's not being used or moving a slower application from flash to rotational disk, they save the company money by deferring additional purchases.

One thing that's often overlooked when examining storage costs, however, is the cost of all of the management. In fact, many organizations say the cost to manage storage is much greater than the cost to purchase it. Therefore, anything they can do to reduce the cost of management will also reduce the total cost of ownership. If it truly is the lion's share of the cost, perhaps it should take a greater priority in the purchasing process. Self-managing, self-healing storage would take priority.

The Reality

While DBAs and storage administrators have the same goals—achieving the mission of the organization—their areas of influence are different, and therefore their concerns and priorities are different. They also speak completely different languages. The DBA is concerned with query speeds and SQL compilations, whereas the storage admin is focused on IOPs and overall utilization of the underlying infrastructure.

Expecting each discipline to understand the other might be a bit of a stretch, but they should be able to appreciate the challenge the other one faces. They can recognize each discipline has unique capabilities that can be brought to bear, and each is uniquely capable of managing its requirements. If each discipline managed their own requirements and communicated external requirements to other disciplines, things could be much better.

MONITORING IS THE SOLUTION

Each DBA and storage admin should decide on the metrics they could use to best track the needs of their discipline, and then make sure they track those metrics. Continuous monitoring of all levels of the environment can provide three-dimensional information, because each metric can be viewed over time, and all metrics can be viewed in relation to each other. Correlation between trends is the key to understanding any performance challenges.

These metrics need to go from the top of the stack to its very bottom. Environments should be tracking whatever creates a database operation (e.g., web form), the database operation itself, the computing that happens as a result, the "read or write to" or from storage and its associated latency, which array that I/O operation went to, and eventually what storage device actually received that operation. Correlating events from top to bottom is the only way to get at the heart of random performance issues.

For example, suppose you had a "hot disk" receiving many more reads or writes than any other device. First, you would only know that if you are continually monitoring your storage. Second, you would need to know why it keeps getting those operations. If you're tracking and monitoring all applications up and down the stack, it should be easy to identify the application creating all the operations. Once the application is identified, you can find out if the operations are normal or not. Once they are identified as normal, you can make a decision as to whether or not that application is appropriate for rotational disk, or whether or not it should be migrated to flash media.

The same is true of an application suddenly seeing decreased performance. Queries that used to take one second suddenly take 10 seconds. What happened? Monitoring everything up and down the stack might allow you to see that the only change is that the media the application is using is now also being used by several of the replications, and it is now experiencing much greater latency than it used to. Clearly, you need faster storage or greater distribution of the workload.

What if you see the media is still performing the same way it was performing before? All reads and writes are being responded to at the same speed as they were before. This might cause you to look up higher in the stack, eventually finding someone changed the code and the queries are now 10 times more complicated than they were before the change. If this is the case, you would probably expect to see a spike in CPU utilization for that application. Again, this would only be possible if you are monitoring the entire stack. The solution might be to change the code, or to move the workload to infrastructure that can handle it.

WHERE TO START

Many people would agree that monitoring is a great practice; what they might not agree on is what to monitor. Each organization should come up with a set of metrics appropriate to their environment that would indicate how the infrastructure is responding to the needs of each application. Organizations need to develop a common set of taxonomy between the DBA and storage administrator. While this seems fairly straightforward on paper, it's a rather difficult undertaking. Organizers should create baselines both for performance and capacity.

Cross-functional teams should start at application performance and work their way down to the database and storage. They should agree on what metrics will be used and then create baselines both for performance and capacity. If this process is not clearly understood, then it can be difficult to create the details of the correct baselines.

Even in our example of a single application in SAP ECC, this may prove difficult. In a typical SAP environment, ECC include a minimum of three different landscapes. These include production, pre-production, and development. The same query may exist across each landscape. However, the underlying infrastructure and database sizes may result in different baselines. Changing workloads and timing of establishing baselines even in the same instance can cause variations. To address this, think of a one week baseline at month end vs. middle of month. These challenges compound as you add landscapes such as development, test, and quality. Each landscape comes with its unique set of attributes.

You can learn a lesson from the SAP sizing tool. In a naming convention worthy of an Abbott & Costello® bit, SAP has a measurement called SAP. The SAP application has a quick performance calculator that helps determine how many SAPs are required to run a given SAP module based on the number of users and database size. Unfortunately, the SAP calculator helps to determine the CPU and memory required to run the module, but not the storage IOPS and latency.

The lesson learned for the methodology is that database and storage teams can work together to create a measurement to convert database performance requirements to storage specifications such as latency and IOPS. If you think this sounds like an incredible amount of work, you're right. The relationship between the DBA and storage administrator is not a simple problem to solve. At the start, you may work with back-of-the-napkin calculations and iterate on it until the taxonomy and resulting metrics are reliable for your environment.

This should be a collaborative process, with each side contributing suggestions. It might be a DBA who remembers to track I/O latency, and it might be a storage admin who suggests tracking the number of SQL compilations per second. Each metric should be discussed and agreed upon by all prior to embarking on the monitoring project.

EXAMPLES OF THINGS TO MONITOR

The following is a short list of metrics one might include to give you a complete picture of performance from the top to the bottom of the stack. In addition, we've included a few storage capacity metrics that will be useful as well. It is not even close to being an exhaustive list of things you should consider monitoring in an application environment. It is merely an example you can use to generate your own discussions. Let's first start with database metrics.

Database Metrics

The point of monitoring the database is to be able to see what's changed. The aim is to determine if the data is healthy or to identify areas of concern. If a benchmark metric changes on the database side, then something has changed on the storage side. If a storage metric changes, but nothing has changed on the storage side, one would look at the database metrics to see if any of them changed.

The following metrics are essential to monitor to make sure that all systems are a "go" in our environment.

Benchmark Queries

Think of these queries as things that don't change. You should design a number of queries that reflect the types of queries that the application generates. These queries will be run periodically to create a benchmarked response time to compare against the current response. If you are having performance problems with the application and this metric hasn't changed, the most likely culprit is some design change on the database itself.

Read/Write Ratio

To help understand how an application is going to impact the storage environment, it is necessary to know its read/write ratio. This can be used when designing future applications with similar read/write ratios. If feasible, keeping the raw data used to calculate this ratio would be helpful as well.

EXAMPLE SQL SERVER METRICS

The following are metrics found specifically with the SQL Server application. If you are using a different application, your metrics will be similar but probably named differently.

Forwarded Records/Sec

This metric is about SQL Server tables without a clustered index, referred to as *heaps*. If SQL Server has to use a forwarding pointer to get a record from a different page from a new location, it creates an additional I/O. If the number of forwarded records has increased, that is going to increase your impact to the storage system, and you should look into defragmenting your table.

Page Splits/Sec
This is another metric to help you understand how fragmented your indexes are. If SQL Server doesn't have the room it needs when it wants to insert a row onto a page, SQL Server will split the page into multiple pages, balance the number of rows out between the pages, then insert the row. This is a very time consuming and CPU- and I/O-intensive process that should be minimized as much as possible. This metric will tell you how often this is happening per second.

Processes Blocked
Blocked processes can't be avoided, as it's the nature of multithreading. The purpose of this metric is to monitor what the normal number of blocked processes are for your system. When the number of blocked processes goes beyond the normal number, there is cause for concern. One thing that might cause this is page escalation, which is when SQL Server locks entire tables instead of rows or pages.

Locking is a normal activity in a database; it protects data integrity and read consistency. Blocking means contention on a specific resource and is not desired on any level. Of course, while locking is normal, it doesn't mean it's always desired.

Batch Requests/Sec
This metric is one of the best indicators of how busy your SQL Server application is. If there are more batch requests than usual, something has changed. It could be a change in user behavior or a change in how the batch requests are being generated. Of course, this can be a good thing. It can mean that the database engine or queries are more efficient now and can just process more workload. Like blocked processes, it's just important to track this number over time to look for times when it changes.

SQL Compilations/sec and SQL-Recompilations/Sec
When SQL Server has to compile or recompile query plans because the plan in cache is invalid or not there, this is called a SQL Compilation. This may be a very small number for some applications, or a high one. Again, the point is to track over time to see if this value changes.

SQL Server Waits
Taking this a step further, we introduce waits into the monitoring picture to measure actual performance achievement. This helps us see how our systems are doing from an end-user perspective.

A thread can be in one of several states. The three most common are the following:

» **Running** - Queries are in active mode of execution

» **Runnable** - Query is queued and waiting on CPU cycles to execute

» **Suspended** - Query is waiting on a resource to execute

There are several I/O resources that may impact a query to enter running state. SQL provides several information of several types of I/O. Understanding the I/O types and their status can prove critical to troubling shooting SQL performance. Here's an excerpt of some common types.

» **ASYNC_NETWORK_IO** - The async_network_io (in SQL 2005 and up) and networking (in SQL 2000) wait types can point to network related issues, but most often are caused by a client application that is not processing results from the SQL Server quickly enough.

» **CXPACKET** - This wait type is involved in parallel query execution, and indicates that the SPID is waiting on a parallel process to complete or start. Excessive CXPacket waits may indicate a problem with the WHERE clause in the query.

» **OLEDB** - This wait type indicates that a SPID has made a function call to an OLE DB provider and is waiting for the function to return the required data. This wait type may also indicate that the SPID is waiting for remote procedure calls, linked server queries, BULK INSERT commands, or full-search queries.

» **PAGEIOLATCH_EX** - Buffer latches including the PAGEIOLATCH_EX wait type are used to synchronize access to BUF structures and associated pages in the SQL Server database. The most frequently occurring buffer latching situation is when serialization is required on a buffer page.

» **SOS_SCHEDULER_YIELD** - SQL Server instances with high CPU usage often show the SOS_SCHEDULER_YIELD wait type, which can indicate a need for further research and action.

» **WRITELOG** - When a SQL Server session waits on the WRITELOG wait type, it is waiting to write the contents of the log cache to disk where the transaction log is stored.

» **LCK*** - When a SQL Server session is waiting for a lock to be released by another session before it can proceed.

SYSTEM METRICS

This paper will assume that applications are running in a virtualized environment, as that is pretty commonplace. In such an environment, there are challenges in monitoring both the CPU and memory resources of each individual VM, as well as looking at the overall utilization of a virtual cluster.

Total Megahertz/Gigahertz

This is the total amount of CPU resources of all hosts in the cluster and is simply the frequency of all of the processors in the cluster multiplied by the number of course. Think of this as the total available CPU.

Usage

The percentage of usage of each individual virtual CPU from the host view—not the guest operating systems view.

Total Memory

This is the total amount of memory available on the host running the hypervisor. This is the memory equivalent of total megahertz.

Active Memory
This is the amount of memory actively used, as estimated by the VM kernel, based on the current workload of the VM or host machine.

Network Usage
If appropriate, the usage of LAN and WAN connections should be monitored as well. One of those metrics is the number of packets sent and received.

Network Latency
One of the best metrics to monitor performance is how long it takes to transmit and receive data.

Packet Loss
This is another great metric. If data is being sent and received between multiple hosts, is it being sent successfully, or is some of it getting lost along the way?

STORAGE METRICS

It’s important to mention that in a modern infrastructure, these metrics need to be monitored on multiple levels. Where appropriate and possible, these metrics should be monitored inside VMs and the virtualization host, at the storage system level (e.g., array, volume, or file system), and eventually at the individual device level (e.g., actual disk drive).

Write Latency
How long does it take to get an acknowledgment that a write operation has succeeded? Since a database will not move forward until it has been told its writes have been successfully written to storage, higher latency values are a significant deterrent to good performance.

Read Latency
How long does it take to successfully receive a read request? Depending on the read/write ratio, the ability for a storage subsystem to respond to a read request might actually be a bigger deterrent to good performance.

Writes Per Second
This metric helps track how busy an individual storage system is. It can also be used to show how capable the system is, or at what point it stops performing. For example, you might notice that at N writes per second, write latency is fine, but once you go to *Nx2*, write latency increases.

Reads Per Second
Depending on the workload, a given application might request more reads than writes. With both writes and reads per second, a particular storage system might be performing differently because the number of writes or reads per second it is not receiving has significantly increased.

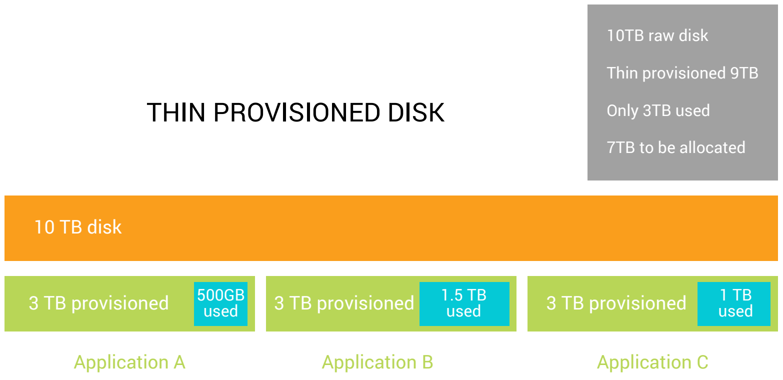
Total Usable Capacity (Per System)

This metric tracks the total capacity available to applications, which is a subset of the total capacity of the system. The reason for the difference is the overhead involved in mirroring or parity-based data protection. For example, a RAID 10 system uses mirroring and therefore has a 100% overhead. 10 TB of disk would provide roughly 5 TB of usable capacity. This metric would need to be tracked per volume, array, or clustered storage system, because the system could run out of available storage at each of these levels. For example, a 5 TB system might have 51 TB filesystems, each of which is only using a few gigabytes of capacity. Looking only at the available storage at the volume level would lead one to believe you have available capacity in the system. Just try to add an additional volume in that 5 TB system and you will find out you have no capacity left, hence the need to monitor the capacity at multiple levels.

Storage Usage

This number would be either a number representing the number of terabytes or petabytes actually consumed, or a percentage of the total usable capacity consumed. This also must be monitored at multiple levels to ensure no level runs out of capacity in an untimely manner. If possible, you should also attempt to associate this metric with the application actually using the storage. Both of these capacity metrics should also be tied back to the media type (e.g., flash, SCSI disk, sat a disk) so storage administrators will know exactly how much of each type of media they have available.

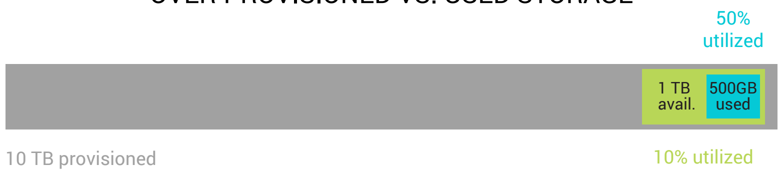
THIN PROVISIONED DISK



Overprovisioned vs. Used Storage

This metric is only appropriate for thin-provisioned environments and speaks to the amount of storage actually used out of the provisioned storage. It's basically the same as the storage usage metric, with a slight difference. Depending on how you count things, this number may be very different than storage percentage used. Suppose you've used thin provisioning to provision 10 TB for a database, but actually only have 1 TB of actual disk available. Now suppose you actually use about 500 GB. Depending on how you look at things, that's either 50% utilized or 10% utilized. The latter is more reflective of reality in a thin-provisioned environment.

OVER-PROVISIONED VS. USED STORAGE



Future State

Mature organizations have monitoring in place to gather these basic metrics up and down the stack. However, there's a gap. Let's examine "free space." As explained in an earlier section, mature operations will understand how much space is physically in use vs. the amount of space provisioned. As a reminder, this is called thin provisioning. Thin provisioning represents a disconnect between the DBA and storage admin. Many databases have a fixed filesystem size regardless of the physical amount of data in the database.

If a DBA creates a new 500 GB DB, regardless of the actual amount of data in the tables, the DB will have a fixed file system size of 500 GB. Looking at storage monitoring tools, the storage administrator has no insight into the actual used in the DB. The storage administrator sees 500 GB of data consumed on the file system and physical disk.

Many major database vendors (including SQL Server) leverage an autogrow feature that will allocate space to a file when needed. You can set a max file size, but that is not what will show as consumed on the filesystem. Think of this as thin provisioning at the database level.

Still, in traditional infrastructure environment, the best world-class organizations struggle to answer the basic business question of how much capacity is left in the system. Many P1 tickets originate from the simple challenge that a database fills up unexpectedly. Best case scenario, additional file system space exists to expand the database. In most cases, the DBA must clean up space to bring the database back online. In the very worst case, storage administrators must scramble to find LUNs to expand the file system. We've seen file systems span several arrays not due to redundancy but rather a desperate effort to bring a mission-critical application back online due to capacity management issues. Hint: this is bad.

In the world of DevOps, the application provides the monitoring hooks to provide metrics for used space. API-driven infrastructures can autoscale the filesystem and database to react to capacity challenges. While applications such as SAP in theory support such auto-provisioning features, few storage administrators trust the DBA with that level of control. Also, capacity planning is just one challenging area.

Remember our CRM use case? The goal is to prevent repeated performance issues. A consistent challenge within organizations is correlating a problem in the DB to the result of an infrastructure issue. Ideally, a DBA is alerted that the primary disk pool of a production DB has contention. Once alerted, the DBA may run a script to relocate log files to prevent performance issues. Again, in most traditional environments, the DBA doesn't have such insights. Applications built on API-driven infrastructure have the ability to self-heal. Most of us don't operate in such environments.

The future state is refactoring applications for a DevOps model of operations. While that's great for the webscale companies of the world, what practical steps are available for most other enterprises dealing with legacy applications and tools?

SOLUTIONS

Having information about all of the metrics on the various levels of the stack—and being able to associate those metrics with the corresponding application and its resources—can solve all sorts of problems. The best way to help DBAs and storage admins come together and solve the capacity and performance problems of their environments is to have them agree on a common set of metrics to monitor. Once these metrics are identified and monitored, we can solve a number of problems.

Ongoing Capacity Management

A storage administrator should never run out of storage. Knowing exactly what storage capacity is being used and how it is being used is the secret to making sure this never happens. A good monitoring system should be able to detect wasted storage and improperly used storage. Specifically, it should be able to identify if the database that asked for 100 TB is only using five. It should also be able to identify an application sitting on flash storage that would be perfectly fine on something much less expensive. Cleaning up wasted space and making sure every application is on the right storage is the first place to start with capacity management.

Capacity management systems can also be used to identify the overall storage growth and when an environment will actually run out of storage of a particular media type. Storage administrators should therefore be able to easily justify the purchase of additional storage by showing how they've already eliminated all possible waste and easily predicting the date when capacity will be exhausted.

Similar to the storage administrator's role of ensuring ample physical disk space, the DBA must ensure there's no wasted space in database files. Proper alerting should exist to ensure the DBA is alerted when a database reaches or exceeds a predetermined threshold. Most importantly, the two groups must communicate. DBAs and storage administrators should have regular capacity-planning meetings. Most database platforms allow for expanding a database file on demand. Likewise, modern operating systems allow for hot expansion of file systems. The output of capacity-planning meetings between the two groups should include a defined process for maintenance of storage provisioning, file system expansion, and database growth.

Documentation of Different Resources

This book previously discussed the importance of making sure applications run on the right media type, but it focused primarily on applications running on media that was more expensive than it needed to be. This was primarily from the storage administrator's point of view, making sure the most expensive storage resources were not being wasted.

But there is another element to this. A DBA should be able to correlate the number of SQL transactions to I/O operations and document that a given application needs faster storage. Once you document applications that exceed a certain number of I/O operations per second really need flash media, it becomes very easy to justify moving an application to a more expensive medium.

Assistance in Future Database Creation
One of the biggest challenges in creating a new database is that there is not a direct correlation between the number of SQL operations for a given application and an associated number of read and write operations the SQL operations generate. The number of high IOPs generated by an application varies based on its read/write ratio and the degree to which queries can be satisfied in RAM. Similarly, there is not an easy correlation between the number of database records and the number of terabytes the database needs to store those records. The main problem here is that records come in a variety of sizes, and that directly affects the amount of storage they're going to use.

But, if you were tracking all of this information from past databases, it should be relatively easy to extrapolate the appropriate number for a future database. The more databases you have and track, the more reliable your estimate should be. Somebody should be able to say, "This application is similar to that application, so let's go look at its read/write ratio and record/GB ratio and use those numbers as a good start for estimating the type and size of storage for the new application."

Workload Placement
Tracking metrics up and down the stack should assist you in more than just deciding whether or not a workload goes on flash, hybrid, or rotating disk. It should also help you understand whether or not it would be appropriate to run an application in the cloud. This would start with tracking the performance of applications that run in the cloud as well as performance of applications running on-prem. Tracking compute and storage usage for each application should allow you to easily decide which workloads would be appropriate for the cloud.

The Original Challenges: Capacity, Performance, and Cost

Unless your company is printing money, there will always be a three-way tug-of-war between capacity, performance, and cost. But information is the secret to making sure your company is not inhibiting its applications' performance or wasting money. That information comes from a variety of metrics your DBAs and storage admins should agree upon in advance and monitor over time.

It should be incredibly easy to document that all capacity is being effectively utilized and yet you're still running out of storage or compute. In addition, it should be easy to document when an application is on the correct media type and computing platform. If it needs more compute or faster storage, performance metrics should be able to document that easily. As for cost, there is nothing better than being able to say all data is in the right place on the right medium, and none of it is being wasted.

SolarWinds Product Solutions

DATABASE PERFORMANCE ANALYZER

Raise the IQ level of DBA/storage admin discussions. Use DPA to understand database performance impacts related to storage. Get the full picture of your database performance story now!

STORAGE RESOURCE MONITOR

Extend your performance visibility from your VMs and datastores all the way to your LUNs and arrays to eliminate bottlenecks with Storage Resource Monitor. Finding storage I/O bottlenecks and planning disk capacity in context of your virtual environment has never been so easy.

About the Authors

KEITH TOWNSEND

In Keith's role as an enterprise architect, he found himself in the middle of sessions that felt like marriage counseling. Both groups have legitimate concerns, well thought-out processes, and proven technologies. The end goals seem to be the same. However, establishing and maintaining a common taxonomy to ensure mission-critical databases receive the storage infrastructure needed to meet business requirements is difficult, to say the least.

STEPHEN FOSKETT

Stephen comes from a storage administrator background. He has over twenty years of experience in enterprise storage, and has seen this animosity first hand. DBAs rarely even speak with storage administrators, and when they do, both parties are skeptical. Neither seems to believe the other knows much about storing and protecting data, and both seem to want the other to stop being so picky about what they are doing.

This book also incorporates the ideas and suggestions of many others, from database designers to data protection experts.