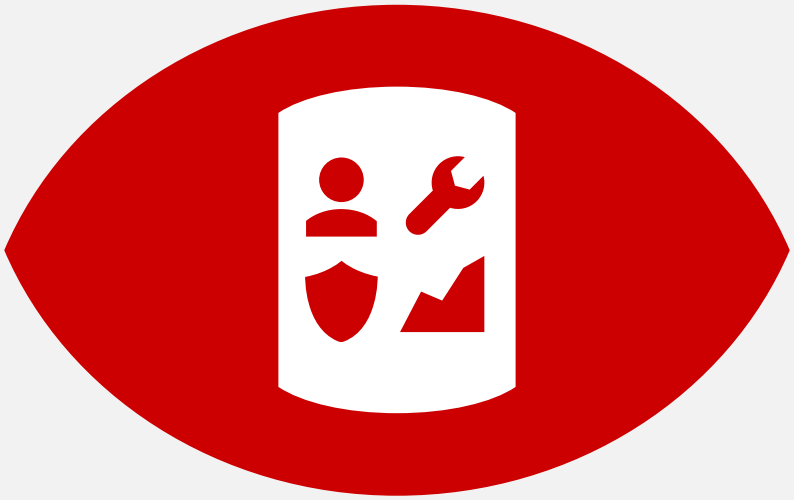


4 pillars of SQL Server Monitoring

By 5 SQL Server experts

Tony Davis

Rodney Landrum

Phil Factor

Kathi Kellenberger

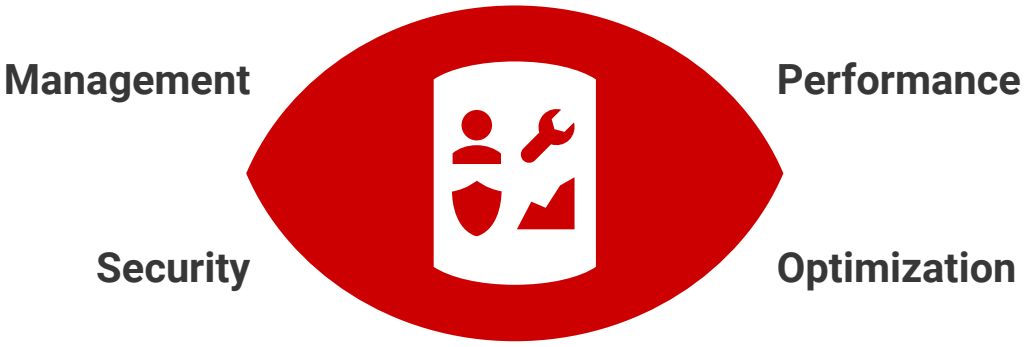
Grant Fritchey

Compliant Database DevOps

redgate

The Four Pillars of SQL Server Monitoring

The four contributors to this whitepaper, Rodney Landrum, Phil Factor, Kathi Kellenberger and Grant Fritchey each tackle one of the four strategies required for a successful, estate-wide SQL Server monitoring solution:



In each case, they write directly from many years of experience working and maintaining SQL Server data systems and describe what they believe is required, minimally, of an enterprise database monitoring solution.

Regardless of the logistics of your monitoring solution, this whitepaper outlines the principal services and features it ought to provide, to ensure that your servers are manageable, secure, and efficient.

The Four Pillars

In '**Management of the Server Estate**', Senior DBA Consultant and Microsoft Data Platform MVP Rodney Landrum describes how a modern database monitoring tool should make management of a growing number of SQL Server easier. He describes the need to keep track of the cost of Cloud-based database services, be alerted to unscheduled code and configuration changes, and have the tool provide some degree of automated performance tuning. He also reveals what he believes would be the "best feature ever" in a monitoring solution.

In '**Monitoring SQL Server Security**', legendary database expert Phil Factor examines the essential topic of monitoring for compliance. He establishes the need to be prepared for every piece of legislation an organization may be bound by, and how you can only be sure of protection with a robust access-control system in place. He also gives recommendations on how to be alerted and prepared for an attempted attack on your systems, so that you can always be ready to respond and protect your data.

In '**Monitoring SQL Server Performance**', Editor of Simple Talk and Microsoft Data Platform MVP Kathi Kellenberger describes the ways in which an effective performance monitoring tool will allow the team to shift away from 'firefighting' problems, as they are reported, to adjusting and fine-tuning the databases based on business priorities. The tool must help predict when server stress conditions will escalate into a performance problem that will affect the business, and it must lead the investigator intuitively towards the root cause.

Finally, in '**Monitoring for Optimization**', Redgate Evangelist and Microsoft Data Platform MVP, Grant Fritchey, explains how a monitoring tool supports a process of continuous optimization, by helping the team manage all resources efficiently, right across the server estate. He focusses on the analytical capabilities of a monitoring tool, such as trend analysis, baselines and variance. These will help the team understand normal and predicted patterns of behavior, in the set of database and server-level performance, availability and security metrics, so that they can respond in a controlled way, when they spot sudden changes or worrying trends.

Tony Davis
Editor, Redgate Software

Contents

Management of the Server Estate

Rodney Landrum

Back in 2012, I was transitioning into the role of 'Director of Data Services', at the Fortune 500 insurance organization where I worked. It wasn't that easy. Previously, as a DBA, I'd spent time developing my own database monitoring and documentation solutions. Now, I needed to alter my perspective to take in the broad landscape of all the database systems we maintained. I needed to seek out the 'best of breed' SQL Server monitoring tool; the one that could help me ensure the uptime and performance of the whole range of data services we supported, across the enterprise.

Many companies like mine were getting their first foothold into "the Cloud", even if it was to test out a development environment. Virtualization for SQL Server was becoming commonplace. Even so, the database monitoring and documentation solution that my team and I had crafted focused mainly on monitoring the infrastructure that I was used to, which was on-premises and was managed locally by a team of skilled and dependable DBAs. The broad view across the organization wasn't like that and required a different solution.

Six years later, I'm now a database consultant. Data is still data, and I am still fortunate enough to be working with it daily, but a lot else has changed. I work with many different customers whose SQL estates are now spread across a range of hosting environments, including on-premises data centers, hosted facilities, Microsoft Azure and Amazon AWS. The logistics of data connectivity has changed and hybridized. Data services have multiplied; if you are a DBA in 2018 you will more than likely be familiar with terms like Data Lake, NoSQL, Data Factory, Containers, Machine Learning and Power BI. Many of these systems have their own monitoring as part of the service but as whole, it is still a challenge to see the big picture for all services combined.

So, what features of a monitoring solution make it appropriate for covering a polyglot estate of data services? Every feature I identified back in 2012, when I wrote The Strategic Value of Monitoring SQL Servers, still holds true. More than ever, I need the tool to be low in both its impact and its demand on services, and which isn't intrusive. I still need precise, configurable alerting, and it's become even more important that I can adapt the metrics, and associated alerts, as required for specific business processes that are supported, or the data that is stored. However, I'd also add some new features to the list, essential to managing the uptime, performance and security of a growing number of database servers, spread across an ever-diversifying range of data platforms.

Support for Categories and Groups of Servers

Most of the organizations that do internal database development have created processes and environments that support their development and testing activities. With several teams working on different applications, there could be specific development, QA, staging and production servers for each application.

Production is always "king". The other environments must be monitored but have different requirements. One would not, for example, expect to be woken one night from a deep dream of peace to deal with a runaway query on a Development server.

There is no one-size-fits-all monitoring and alerting strategy for a modern SQL Server estate. Of course, the tool should collect a sensible, base set of metrics and alerts, for all servers: Beyond that, however, I need the ability to group servers by application, and then further by their specific usage, to allow for a singular set of rules and alerts to be applied to each group. This makes it easy to manage the monitoring strategy across the estate, and adapts it according to, for example, the tier-level of support required for a group of servers.

Cost-based Computing: Counting the Cost

When companies make the switch to a service-based Cloud offering, they are going to be billed monthly for the services they use, beyond the basic database usage tier. They suddenly need to be aware of fluctuations in cost as service demands change. They will often have applications that they support on premises, as well as those that are hosted in a Cloud service such as Azure. Ideally. They would want one single tool that will monitor both the usage patterns, and the cost of the resources.

Fortunately, both AWS and Azure have billing and cost APIs that can be used to pull down and store this information for analysis. It's possible to do this as a separate, manual process, but ideally it would be done automatically by the monitoring tool. Having daily aggregations of usage and cost for the entire estate, and to drill down into the detail to find the high-cost services, will help DBAs and managers predict future costs and make appropriate adjustments.

Synthetic Workloads and Query Baselines

With a growing server estate to manage, the administration team are required to be "proactive" in their approach to the twin objectives of ensuring uptime and providing database systems that perform well. Their oft-quoted frame of reference is that they need to know that there is a problem before the user does. In some respects, a standard monitoring solution makes this proactive approach easier to achieve. It is easy, for example, to monitor TempDB space usage and predict when the server will run out of space. If I'm alerted to runaway growth, I can identify the offending query and, with business sign-off, kill it.

In reality, of course, it's impossible to be aware of every problem before the user does. There are just too many variables that can cause applications to slow to a crawl or even bring the SQL Server down altogether. It's hard to predict, for example, that a stored procedure will suddenly start running 10 to 20 times slower than normal, at 2.35PM tomorrow.

However, there are several ways a modern SQL monitoring tool can help the team manage a larger number of servers, while still minimizing nasty surprises. First, it needs to establish baselines that define the normal patterns of behavior for each metric, and to alert when significant deviations are detected. When this happens, it ought to lead the user towards a possible cause, such as by tapping into the Query Store to find out if it was caused by a change in execution plan.

As the server estate grows, more of its administration will be done by staff who are skilled but not database specialists. Therefore, it's important that this data is presented in a simple, graphical way so that any user can understand what has changed, and the possible causes.

As well as giving warning of specific problems, the monitoring tool ought to make it easy to get a measure of how well the overall system is performing, via support for synthetic workloads that we can use to verify query performance. This could work almost like an application "speedo". For example, I'd like to be able to create an application-based performance metric, such as to measure the time taken from order submission to order confirmation. One or more queries could be developed to run at a set interval, and if a duration threshold is met, the monitoring solution would trigger an alert. The alert details should include the current state of resource usage, any blocking and session counts along with the duration of the query in question.

Automated Performance Tuning

Monitoring systems aren't really supposed to resolve problems, just report them. The idea of moving beyond diagnosis to cure is one that alarms most DBAs. However, as SQL Server estates grows, so does the desire for some of the very basic tuning to be automated by the monitoring tool.

For example, I see no harm in a monitoring solution that can identify poorly performing queries and add a missing index to resolve the problem. After all, this service already exists, at least for Azure SQL Databases. It will identify, and create, indexes that it detects will improve query performance. Having used this service myself in the past I am comfortable with letting a tool perform this work for me, with the obvious stipulations that it will remove the index if it proved to not have beneficial impact.

A SQL Server monitoring tool ought to make a similar service available to all databases. The feature would be configurable to only report on the missing indexes initially and if approved, create the index at the appropriate non-peak time, which it will have also identified, from its knowledge of resource usage patterns throughout the day. Of course, the DBA will still maintain full management over the systems and can override and rollback any decision the service makes.

Alerting on Code or Configuration Changes

In my former life as a DBA, our team were often burned by code or configuration changes that made it to production, either not fully tested, or outside of our change management processes, or often both. This should not be allowed to happen, but I have seen it often enough to know that it is something I would like monitored in real time.

I really want to know when changes happen, what changed and who did it. I should be able to add to the tool's configuration settings, both the times of the scheduled deployment windows, and the users that have the authority to make changes. That way, the next time an unauthorized user does an ad-hoc alteration of a stored procedure, an alert will fire, and that employee will be fired!

Of course, I am being too harsh. However, I do want monitoring tool that makes it easy to connect events such as authorized or unauthorized deployments directly to their consequences for the performance and stability of the server. Or, to put it more bluntly, it would be cool to be able to point the finger and have the monitoring system back me up.

The Best Feature Ever: Monitoring Migratory Data

As DBAs we know that data migrates around the enterprise, like herds of wildebeests across the savannas. It moves about from one system to another, being shaped and re-shaped, aggregated, cleansed, de-duped and otherwise exhaustively transformed. Sometimes it undergoes this amazing change every night, and even every minute. It may be an Excel file one minute, and a Power BI graph the next.

If at any time during this astonishing feat, a transformation process breaks, then other processes downstream are going to be very upset that they did not get their turn to play with the data and they will complain, in turn, by shutting down, throwing error tantrums and breaking other downstream processes that want the data.

End-to-end, this would make a very complex dependency diagram, and that is what I want the monitoring tool to provide, in visual form. I would like an integrated, interactive and configurable dependency map of all my data flows, from one system to another, and to be able to see when any one breaks, or, better still, when it is about to break.

Furthermore, I would like to be able to predict the impact to processes later in the workflow. For example, what if a file copy process that moves 60 Gigabytes of Excel files fails because the drives ran out of space? How much time does it take to recoup that space and restart that process? How long can the business expect to wait for those reports that were dependent on that data?

Conclusion

I know that everyone has their own ideas for what makes for an ideal monitoring solution for their SQL Server estates. I've outlined some of my ideas for how a monitoring tool can help us monitor, analyze and predict resource usage, including costs, across a growing and diversifying estate, and to help the organization to connect server resource usage and error conditions directly to their impact on business processes.

However, if it were possible for a list to scratch, this list here would only scratch the surface of what is possible to do with an enterprise database monitoring solution. In 6 to 10 years from now, I predict there will not even be a need for monitoring data estates, or whatever they will be called at that time, because they will all be self-healing and living in a server-less, containerized world. I suspect that the good old days will be recalled with nostalgic sighs by old DBAs.

I really hope, though, that there is still a role for the expert, because I plan on working with and monitoring SQL Server until I, in turn, need to be monitored for slow or failing internal processes.

Monitoring SQL Server Security

Phil Factor

A characteristic theme of all the current legislation on the responsible curation of data is that there should be a system for continuously monitoring and analyzing database activity, in real-time. This Database Activity Monitoring (DAM) system must detect anyone, inside or outside the organization, accessing or tampering with information in any database-driven applications that handle personal, financial, HR, or other business data.

The monitoring system will collect metrics such as syntax errors, denials of access, permission changes and unusual access patterns, and then aggregate them, and report on them. Although this sort of database security monitoring is part of the broader activity of database auditing, and real-time protection, it is quite separate from, and independent of, any defensive techniques to prevent attack. The aim is to be immediately aware if this layer of defense has been avoided, sidestepped or penetrated, and to detect any unusual activities on the server, including database read and update activity, server or database settings, or permission changes.

A summary of the requirements

A DAM system will need to ensure that it can detect and log any user activity that violates the security policies for the data, or that indicates unusual or suspicious user activity. This system must operate as independently as possible of the RDBMS, to prevent anyone who is attempting to tamper with the system from also disabling the audit and control system. For example, any attacker who gains administration rights can then use it to disable triggers and constraints and stop both SQL Server Audit and SQL Server Agent.

We need an independent process, such as a custom monitor, that regularly checks that the audit and control system that you are using is still working.

The monitoring system must provide:

Data privacy – by ensuring that only authorized applications and users can view sensitive data. With a good access control system in place that matches the users to their requirements for data access, It is then wise to log any failed attempts to access data. The DBA must aim to be able to report a list of every person who has read access to sensitive data.

Database authorization – by checking the activities of privileged 'superusers' and reporting on anomalous activities, and by making sure that all administrative changes to access control and authorization (DCL) and metadata (DDL) can be traced to an individual user, where possible, rather than a shared SQL Server admin login.

Data governance – by checking that no changes are made to critical database metadata without the relevant change control, and by checking that no unauthorized changes are being made to data values.

End-user accountability – ensuring that unauthorized or suspicious application-based user activity can be tracked and audited with sufficient veracity and detail to be usable as legal evidence. This must be sufficient to associate specific database transactions with individual application end users.

Fraud detection – monitoring and recording unusual patterns of usage by authorized users, and particularly changes to historic data, or business processes, made outside the normal procedures.

Cyber-attack alerts – detect an intrusion based on a divergence from normal patterns of activity, by first establishing a baseline of normal user behavior, the background level of database errors, patterns of access, denials of permission, and so on. The system should also be able to detect intra-database attacks and back-door attacks, in real time

Configuration auditing – detecting changes in database and server configuration that aren't covered by change management procedures, to comply with audits required by the U.S. Sarbanes-Oxley Act (SOX)

A system that also works with Virtual and Cloud environments – many systems rely on in-house hosted databases, whereas cloud and virtualized environments are just as much in need of monitoring database activity

What is needed in a monitoring system?

All organizations that handle data do so in the face of the wide-ranging requirements, and growing legislation, such as embodied by the GDPR, the Payment Card Industry Data Security Standard (PCI DSS), the Health Insurance Portability and Accountability Act (HIPAA), the Sarbanes-Oxley Act (SOX), U.S. government regulations such as NIST 800-53, and many others.

Therefore, any monitoring system that aspires to compliance must allow for a wide range of custom monitoring, to allow monitoring of the processes of a database application, as well as the database system. It must be able to monitor errors, sessions and individual SQL Statements that access database objects that allow access to sensitive information, as well as changes to database objects. It must also be able to monitor database server settings and database settings to detect changes.

However, of course, all monitoring activity comes with a technical cost in terms of CPU, memory, potential attack-surface and network traffic. The more you require of your monitoring software, the greater that cost. Monitoring must be as lightweight as possible and must be careful to avoid locking or blocking. For SQL Server, the use of Extended Events is an obvious way of keeping this overhead low. The two principles of filtering-at-source and being selective about the metrics collected are always important.

Any security monitoring system, itself, presents a potential security risk because of the sensitive nature of what it must check. The monitoring system must have had rigorous security checking and use the best of current security practices. Ideally, the database and monitoring system should engage in mutual security checking. The storage of the monitoring system should be well tied-down for security.

How to monitor for security

The great value in a monitoring tool, such as SQL Monitor, is in 'baselining' any metric you provide and allowing you to set alerts when there is a substantial variance from this baseline. In SQL Monitor, we install each custom metric, which is simply the integer value returned by a SQL Query, separately. This gives a wide range of possibilities for alerting. More advanced reporting and checking is better done by a separate application, because requirements vary so widely.

For tracking the business processes of an application – there is no industry standard, so SQL monitor relies on an interface with the application, via the database. This would need a custom monitor, or set of them, that can produce graphs that plot current activity against a baseline, and mange alerts.

For detecting intrusions from outside the organization – the easiest course is to monitor the characteristic errors found in both in-band and blind SQL Injection, to monitor all denials of access to database objects, and to check any attempts to use the system stored procedures that are used by hackers to extract data.

For detecting changes in server and database settings – the monitoring tool should monitor for specific informational messages relating to changes to the server and database configuration. However, if we only monitor changes, it is easy to miss the fact that a configuration has drifted from the value at the time of release. The agreed settings should be stored within a DCMA (Database Configuration Management Archive). There are many ways of using this data, but we can, for example, load these agreed settings into an extended property that the monitor can compare them to the current, 'live' settings

For checking on the session-level activity of users – SQL Monitor, as shipped, can only show general trends and manage alerts. The only practical way of checking on session-level user activity is to use Extended Events and a custom monitor. We could supplement this with a database-level process that monitors for any unusual chain of processes, to use a custom monitor to alert the administrator to check it for validity.

For changes to permissions and users – Ideally, all users would be administered via an LDAP or Domain, but there should be checks on the DCL within the database, by comparing with the access control code for the release, to ensure that no unauthorized changes have happened, such as to add a new user or role, or to change access rights to existing users. It is easy to use a range of Extended Events to audit all attempts, successful or not, to make changes to permissions and logins.

For database drift – The monitoring system should check for any changes to the schema of the database that have been made without any authorization or change-control. It should alert the administrator whenever a change happens, and record the nature of the change, who did it, when, and to which database object.

Security monitoring of a database application is far more effective if the database has an interface that reports on security issues within the application. This has only to report a numeric summary of the state for each security metric, such as the number of anomalous user interactions, or number of failed front-end injections, or where a single user makes many requests for lists of data. It must allow a DBA to query for details. This relies on the application having an effective logging system for user behavior, within the database itself. With a system like this, SQL Monitor can be used along with a custom metric, to do the graphing and alerting.

Promoting a culture of alertness

Whenever I've been asked to assess the security of a website or application, I've found that I can usually predict what I'll find by first assessing the attitude of a team to designing for security, and to security monitoring. On one occasion, a successful penetration of the network happened whilst I was being briefed by the team leader on their splendid security efforts!

Over the years, I've found that developers and admins have needed encouragement to develop an interest in defensive security design, least-privilege access, and fine-grained access control.

Database security has to be tested in the same way that one tests the software processes, by using penetration tests as part of the test regime. The best strategy for focusing effort on security defenses, and monitoring, is to make the security requirements an intrinsic part of the documented user requirements. With the arrival of GDPR, it is now far easier to persuade the business to do this.

Conclusion

In reading through the requirements of the various regulations that are imposed on the processing and holding of data, you'll find a broad range of monitoring requirements. Where one is certain of the sort of tampering, penetration or copying of data that is being attempted, or is likely to be attempted, then it is relatively easy to detect it. The odds are stacked against the attacker with the well-known attack vectors, such as SQL Injection. If the type of likely attack is unknown and difficult to imagine, then then we need to monitor a more diffuse collection of metrics and determine the reason for all sudden changes in the patterns of access, even if often the cause is innocuous, such as a popular football or baseball game.

With any system that is concerned with security, the principle is to flag up all changes that do not correspond with agreed changes that are logged in a source control system as being authorized. Otherwise it is difficult to tell whether a change is intentional or malicious. For this sort of monitoring, nothing beats a visual representation of database metrics that supports the intuition of the database professional.

Monitoring SQL Server Performance

Kathi Kellenberger

During normal day-day-day business, end users only care about SQL Server when they find that performance of the application is so slow that it impedes their work. At that point, they will rightly complain. Slow applications can cause companies to lose customers and it will make business users less productive. In either case, it costs the organization money. Performance matters.

This explains why performance problems are usually the main reason given to start monitoring SQL Server. The idea is a good one: to spot impending performance problems before the users do. However, even with a monitoring system in place, it isn't easy to spot an issue before it becomes a problem. It is hard enough to pinpoint the cause of poor performance once the phones have started ringing. It isn't that you lack the basic information; there is no shortage of performance metrics available in SQL Server, or ways to get them. The difficulty is often more one of seeing the wood for the trees. It's often hard to determine the root cause from the vast sets of server metrics that indicate resource contention, error conditions, and bottlenecks on the SQL Server.

Effective performance monitoring and diagnosis requires a tool that not only collects all the required performance metrics, over time, but also presents the relevant metrics, graphs and summaries, at the right point in the performance investigation.

Proactive monitoring plus fast reaction times

DBAs find that they are more effective when they can spot, and work on, impending performance problems, rather than to each crisis. They want to be able to prevent performance issues from reaching the point where anyone complains, because then they can prioritize and plan their work.

As Rodney Landrum noted, in the Management section, an enterprise SQL Server monitoring tool will make it easy to categorize servers into Groups, whose SQL Server instances share a common set of properties or requirements. For example, you might have common requirements for production servers, staging or test or development servers, or you might group servers according to service level requirements, or to the sensitivity of the data they store. You can adapt the monitoring strategy for each group and prioritize the issues that have the biggest impact on the business.

Of course, once a call is logged and is in a tracking system, DBAs begin to lose their freedom to prioritize, because issues must be dealt with within a timeframe. A monitoring tool also has an important role to play in improving the time taken to resolve real-time performance problems.

It's important that the DBAs do not become so immersed in dealing with individual calls about performance that they can't attend to preventing problems in databases that are more critical to the enterprise.

Tell-tale signs of potential performance problems

There are several trends that will give you advanced warning of performance problems. Amongst the more important are:

- Current performance is unusually slow
- There is high processor pressure on the server
- There are more incidents than usual of blocking, deadlocks and long-running queries
- There are an increasing number of waits of certain types
- Performance is degrading significantly as the database size increases
- Use of memory and disk-space for logging is rising rapidly

As Grant Fritchey suggests, a good test of any monitoring tool is to write scripts to reproduce a couple of the main performance problems from which your systems suffers, and then assess how quickly the tool helps you spot the problem and determine a possible cause.

Collecting the diagnostic data

With responsibility for just a few servers, it is perfectly fine to use SQL Server's own tools to collect the diagnostic data you need to help you spot worrying changes or trends. The information you get from monitoring is always better than the subsequent description you get from angry users, once the trend has hatched into a problem.

Windows Server and SQL Server have several built-in tools that can help DBAs figure out what is going on at present. They can use extended events, perfmon, dynamic management views (DMVs), traces, and tools from the community, such as `sp_whoisactive`, to troubleshoot performance. If the issue continues after the DBA is notified of the problem, they may be able to find the root cause using these tools. If the problem is intermittent, then by the time the issue is reported, it's too late, and they must wait until it occurs again.

The DBA needs to collect this data continuously, at periodic intervals, so they can troubleshoot problems quickly, regardless of when they occur. Before effective database monitoring tools became available, DBAs had no choice but create homebrew solutions, but it was always a distraction and it is an intrinsically difficult job to create a tool to monitor the performance of SQL Server.

The DBA had to write scripts to collect performance metrics from all the sources mentioned and store them in a dedicated database so that there is a history in place that can be researched and played back to the point when slowness occurs. Writing the scripts and developing processes to gather the information requires a huge effort, especially for multiple versions of SQL Server, that takes away from other important responsibilities and tasks. The data from each of these sources is in a different format, and it's also difficult to know exactly what to collect. One set of metrics will be great for troubleshooting I/O issues while another set will be better for discovering the cause of blocking.

As the number of monitored servers and databases grows, and server workload increases, you need new ways of spotting these tell-tale signs, and then diagnosing the cause, quickly. A good monitoring system allows the DBA to do just this across a large group of servers. In other words, it will help you join all the dots, across the entire landscape, to reveal the cause of the problem, quickly

Requirements of the performance monitoring tool

By using a packaged monitoring tool, the DBA will spend less time figuring out which metrics to collect and writing those scripts, and more quickly getting to the root causes. It provides a ready-made solution to spotting potential problems and leaves the existing range of diagnostic tools to assist the DBA to drill deeper into the difficult problems, to establish the cause and provide a remedy.

There are any number of ways in which you'd want your monitoring tool to help you identify and diagnose performance problems, using the diagnostic data it collects, and here I can cover only a few 'highlights'.

Connect server stress conditions quickly to their cause

Why collect and record so many metrics? That's because it's easy to misdiagnose server stress conditions, if you only have one or two performance metrics to go on. High CPU use isn't necessarily indicative of CPU pressure, just as high disk IO doesn't necessarily indicate an overwhelmed or underpowered disk subsystem. There will be an often-complex inter-relationship between CPU, IO and memory use on the database server. If users run complex queries that request large volumes of data, this will place demands on CPU, disk IO, and on memory required in the buffer cache to store all the data. Any or all of these could become a performance-limiting factor. You need to look at how individual metrics inter-relate.

Let's say users are reporting slow database performance, and your monitoring tool tells you that CPU use is "high". Does this mean that "processor pressure" is causing the performance issue? It depends! After all, if the server is busy then your processors are supposed to be working hard, to process all the users' queries! A good monitoring tool will help you quickly correlate this one metric with others that will confirm (or rule out) CPU pressure, and to the specific queries running at the time of the problem.

Is the CPU busier simply because the server is busier? Are the number of connections, or number of user requests per second higher than normal? You can only know for sure if your monitoring tool has a **baseline** for each of these metrics; in other words, if it enables you to quickly compare current levels to levels for the same metrics at the same time yesterday, or the same time over the past several days or weeks. Grant Fritchey will cover baselines in detail in the Optimization section. They are essential to helping the DBA understand if performance degrades predictably during certain time periods or if the current performance of the server is typical or out of the ordinary. They can also watch for changes as the data grows or the system becomes busier over time.

Is the CPU increase accompanied by a significant increase in the time user processes spend simply waiting to get on the CPU (called signal waits)? Higher than normal signal waits, plus high CPU usage, might indicate CPU pressure. Again, you'll need a baseline for signal waits to establish this, and your monitoring tool should make it very easy to add a metric to collect this data, if required.

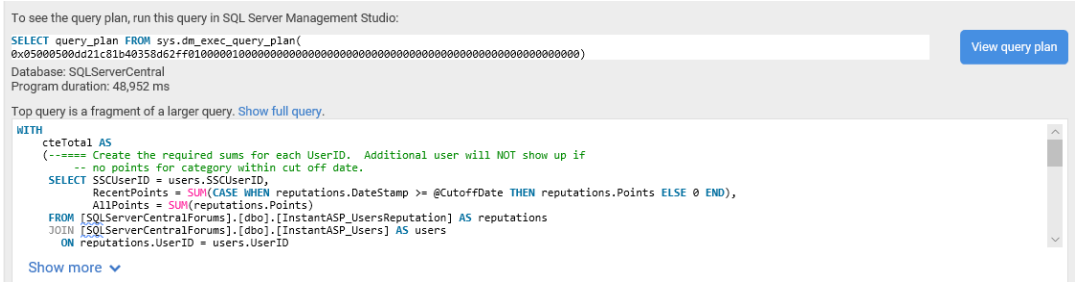
Your performance metric baselines might reveal that CPU pressure is 'normal' in the last week of every month. At that point in the performance investigation, you'll want to connect the server stress condition, be it high CPU, memory, or I/O utilization (or all three), directly to long-running queries over that period. Often, excessive CPU use is caused by long-running, complex queries that simply require a lot of processing power. Poorly tuned T-SQL statements, and poorly designed indexes, can cause severe overutilization of all resources.

If you confirm CPU pressure, for example, you'll want your monitoring tool to reveal the most CPU-intensive queries that were running at the time. In other words, you'll want to see the costliest queries in the cache, ordered by total worker time (i.e. CPU time):

TOP 10 QUERIES TOP 10 WAITS

Query Text	Execution count	Duration (ms)	CPU time (ms)
WITH cteTotal AS (--==== Create the required sums for each UserID. Ad…	5	9,790	27,953
SELECT TOP 5 ck.[_Name] [ClusterName], sk.[_Name] [SqlServerName], tq…	3	8,831	24,184
WITH cteTotal AS (--==== Create the required sums for each UserID. Ad…	5	6,789	18,968
INSERT INTO [#Cluster_SqlServer_TopQueries_StableSamples] SELECT [Id]…	2	3,236	12,992
SELECT InstantForum_Users.UserID, 0 FROM InstantForum_Users INNER JOI…	62	8,613	7,798
SELECT target_data FROM sys.dm_xe_session_targets xet WITH(nolock) JO…	22	3,661	3,453

Perhaps they turn out to be the end-of-month business reports? You'll want to see the SQL text and execution plans for these queries:



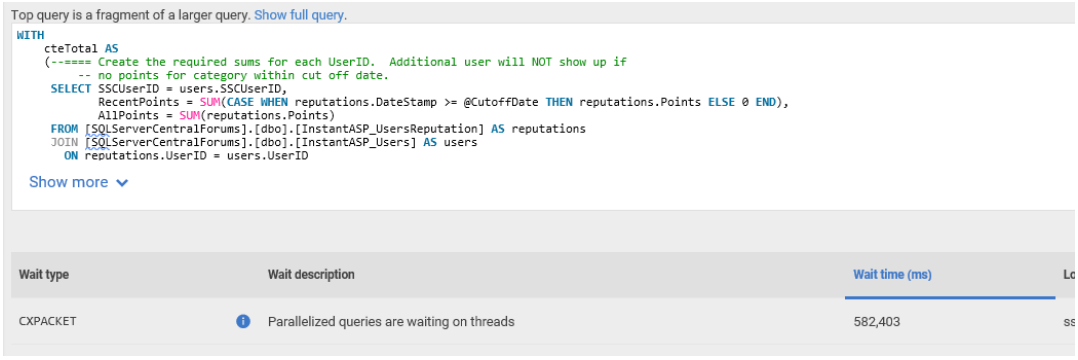
Are there opportunities to relieve the pressure by tuning these queries, or the database indexes? If not, can you take steps to optimize CPU allocation for that period. For example, if these are cloud-based servers, can you elastically expand to use more processors, and so optimize the performance of your important reporting queries?

Without the tools to see if the bottlenecks are caused by poorly written queries or missing indexes, the knee-jerk reaction is typically to increase the number of cores. With the help of a monitoring tool, a small number of queries can often be found that are causing most of the bottlenecks. Once those queries are identified, steps can be taken to tune the queries and indexes involved.

Analyze server waits

Wait statistics are a powerful tool in establishing the most significant bottlenecks on your SQL Servers. Every time a request is forced to wait, SQL Server records the length of the wait, and the cause of the wait (the wait type), which generally indicates the resource that the request was waiting for but couldn't acquire. Waits may be related to parallelism, I/O slowness, or locking, for example. This is reported as overall aggregate values, collected since the last time the server was restarted. DBAs typically review waits to understand the stresses on the server.

Again, knowing that a certain wait type metric is "high" does not on its own indicate a problem. Your monitoring tool must help you correlate these with other metrics, and to drill-down on these waits to find the actual queries affected by them:



A good monitoring tool will also report the top waits over a period, so that the DBA can easily spot trends or outliers.

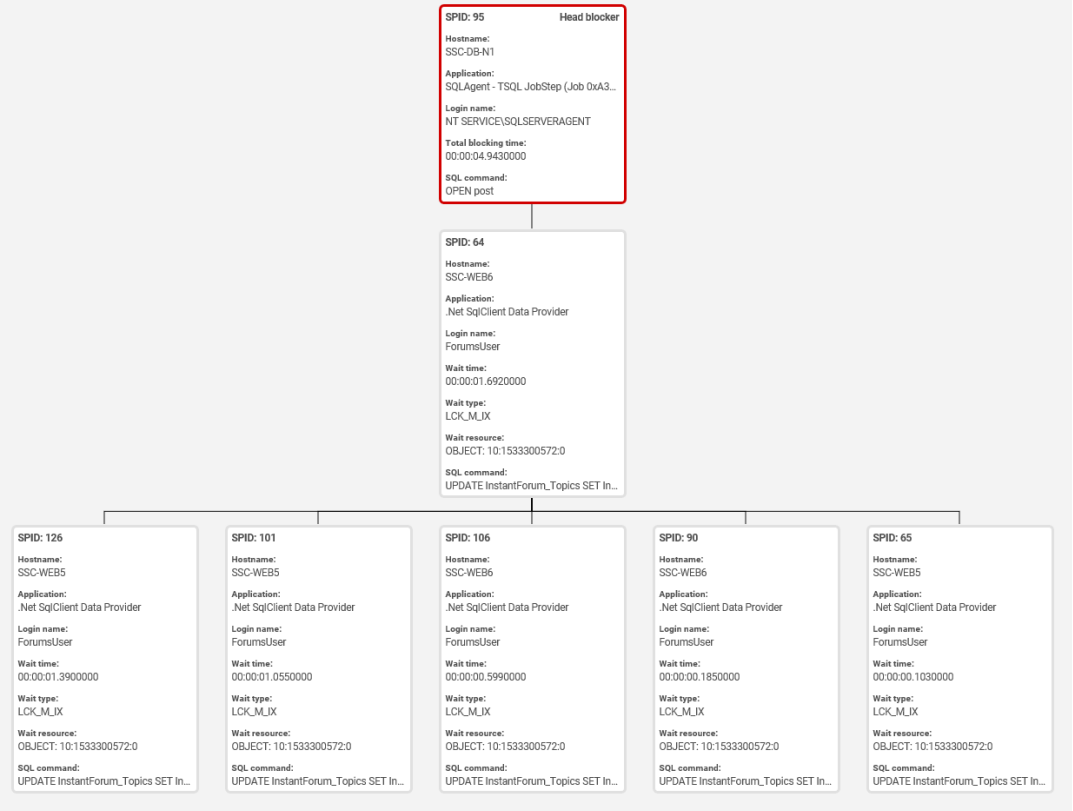
Wait type	Wait description	Waiting task	Wait time (ms)
CXPACKET	Parallelized queries are waiting on threads	2,016,975	103,784,012
SOS_SCHEDULER_YIELD	A process has yielded CPU	9,711,806	19,174,571
BACKUPIO	Backup task waiting for data or an available buffer	462,594	9,360,120
MSQL_XP	Waiting for an extended stored procedure to finish	1,784	4,557,538
PAGEIOLATCH_EX	Waiting on a latch to read data from disk into memory	24,967,043	3,997,567
LATCH_EX	Waiting for non-buffer page latches to clear	725,505	3,817,002
BACKUPBUFFER	See MSDN for this wait description	342,637	3,345,465
ASYNC_IO_COMPLETION	Waiting for asynchronous I/Os to complete	32	3,070,164
HADR_SYNC_COMMIT	Waiting to commit due to synchronous replication processes	790,163	2,139,859

Let's say you observe 'high' CXPACKET waits. Is this a problem? This type of wait will always occur when queries are running in parallel across multiple processors. A common misdiagnosis on seeing very high CXPACKET waits is that the parallel workload is saturating the disk IO subsystem. However, first, check the queries affected by these waits. If this is an OLTP system, where transactions ought to be short and fast, you'll want to investigate those queries that are getting parallel plans and look for tuning possibilities.

Fast real-time troubleshooting

As proactive as a DBA would like to be in heading off issues, before they become real problems, the monitoring tool will always still have an important role to play in helping resolve real-time performance issues.

If an important business server is experiencing severe performance problems, right now, then you'll want to know very quickly what connections and sessions are affected, and what activity is occurring on the server. One of the ways a professional monitoring tool can really help, is not just detecting excessive 'blocking', or reporting that a deadlock occurred, for example, but also providing simple visualizations of the blocking or deadlocking chain. The following image visualizes a blocking chain:



Blocking in SQL Server is like an intersection in a traffic jam. The query at the heart of the problem affects other queries, making them wait until the problem query is done running or killed. Frequent causes are excessive I/O because of scans against large tables or transactions left open after errors. Solving the problem in the short term involves finding the offending query and possibly killing the connection. A good monitoring tool will identify the cause of the blocking so that appropriate action can be taken.

A deadlock is a circular locking chain, because every process in the blocking chain will be waiting for one or more other processes in that same blocking chain, such that none can complete. This is a serious error condition that SQL Server resolves by killing and rolling back one of the processes. The DBA will need to investigate immediately to find out what caused the deadlock and take steps to prevent it recurring.

Conclusion

If your monitoring tool merely detects problems as they occur, then the DBA will simply spend a lot of time fighting fires, as they occur, often at the expense of having the time to determine the root cause of the most frequent performance problems that occur on the server. By the time a performance deficit becomes a problem so extreme that it prevents the users doing their work, it will have affected customer loyalty and employee efficiency. Most DBAs want to work proactively, to adjust and fine-tune the databases rather than to just react to calls from users and first-line support. The trick is to notice that performance on one database, amongst many, needs improving, before users complain. The right monitoring solution will go a long way towards helping the DBA be proactive and spending less time fighting fires.

Monitoring for Optimization

Grant Fritchey

Frequently when people think about optimization, they focus on enhancing performance. After all, if we make a query run faster, then application users have a better experience, or the company's end-of-month reports are ready sooner. However, performance is only a small part of optimization; we also need to optimize database availability, connectivity, network communication, security and so on. To do all this, we need to ensure we're using memory effectively, and not overloading the CPUs, that our drives are not filling up, and lots more.

To help us make optimizations, our monitoring tool must help manage these resources efficiently, right across the server estate, in such a way as to improve performance, minimize downtime and fix all those other problems caused by a lack of resources. As Rodney hinted in his Management section, this boils down to being aware of worrying trends, or abrupt changes, in resource use as early as possible, so that we spot pending issues before they become a problem that affects the business, or its customers. Ideally, the tool will help us relate any changes in server resource usage, and what caused it, directly to changes in the performance or behavior of dependent application and business processes. This makes it much easier for the team to spot opportunities to optimize the processes the company cares most about.

Trends and Trend Analysis

A range of system resources will need to be managed and optimized, from the bandwidth of your disk controllers, to the number of transactions that your replication distribution server can support. We could monitor the growth trends of any of these resources over time. However, to keep things simple, we'll focus here on those resources that generally cause the most pain within SQL Server: These are CPU, Memory, I/O and disk usage.

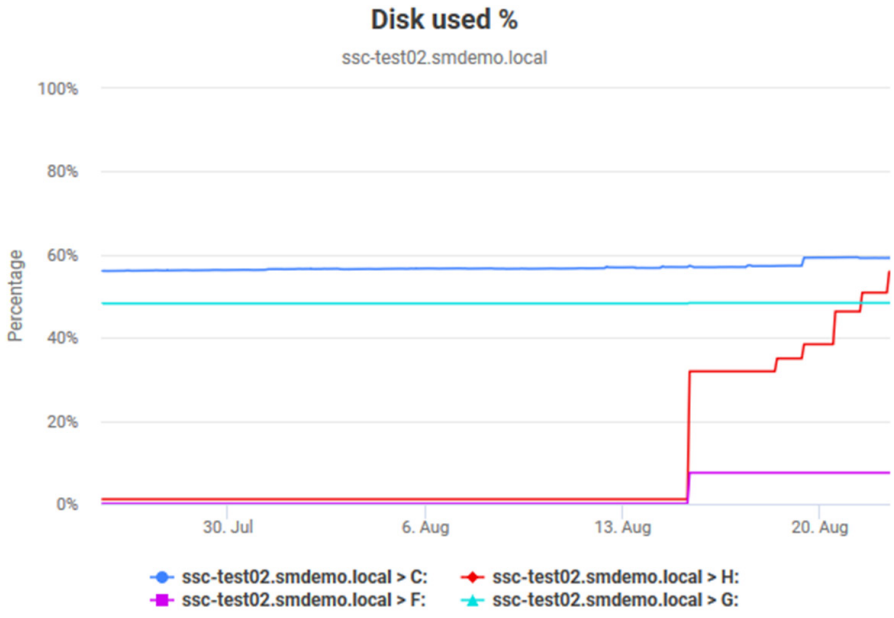
It's rare for one of these resources to suddenly be completely consumed, by one or more processes, on a SQL Server instance. Consumption tends to creep up quietly, over time. If you're suddenly confronted with a full drive, then you're immediately pitched into firefighting mode, attempting to deal with an emergency that would never have happened, had you been monitoring the space consumed on that drive, and the rate of growth of your data, and in the size of your database files.

However, we learn nothing by finding out that we're using 50% of the amount of space on a given disk until we know whether this amount is increasing or decreasing. To establish significant trends for each of these resources, we must first capture sufficient measurements over a period.

We also can't assume that resource use is linear: it could be exponential or follow any number of different types of curve. This means that it isn't enough to know the rate of change over a short period. What if our database gathers data during the week and then on weekends ships it to a data warehouse or data mart. If we measure the amount of space used on Monday, 5%, and then on Friday, 75%, we may panic. In short, to establish true trends, we not only have to compare day-to-day behaviors, but also weekly and monthly behaviors, to be confident that we can then make accurate projections, based on the data at hand.

Having captured these metrics multiple times, over a suitable period, and retained the information, our monitoring tool should use it to create graphs of data, so any user can easily see the usage patterns and trends. After all, this diagnostic data is not so useful to the business if no-one, or no-one but the DBA, can use it to make optimizations.

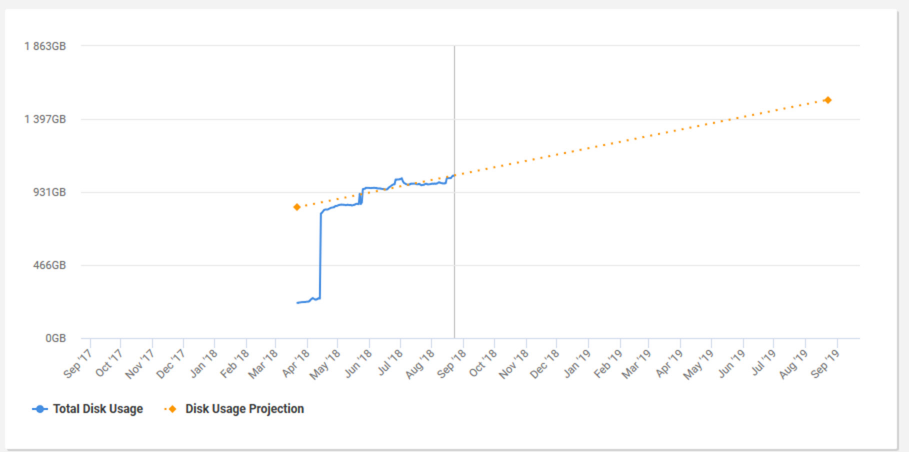
Sticking with our example of disk growth, you may capture the database disk growth for a server daily, for one month. Plotted on a timeline, the information would look like this:



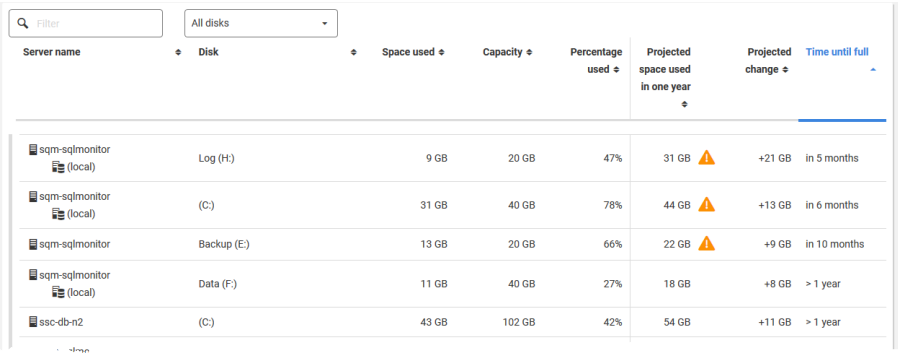
Anyone can see immediately that one of the four disks has experienced a great deal of growth over a short period of time, and clearly has a trend line that suggests within 2-3 weeks, it will be full. That user can then take immediate corrective action, if it happens to be a member of the SAN admin team or inform the person who can.

It's possible, upon investigation, that we may find a cause for the growth of that disk. Maybe a new application behavior was introduced, capturing a lot more data. Change tracking within our monitoring tool would help us here. Maybe this is a drive containing the log file and the log backup job has been failing. Our monitoring tool will both show us the failed job and the lack of backups for a given database. Trends like this are one data point that we can use in combination with other data points from our monitoring tool to arrive at a diagnosis, before we once again enter firefighting mode.

Monitoring the use of disk space server-by-server gets harder the more servers you manage, or at least it gets easier to miss something. Ideally, our monitoring tool will also predict trends for the server estate as a unit, showing the current amount of storage in use, and projecting growth:



It will then show us the breakdown of total disk use by server, so we can see which are the ones that need our urgent attention.



Server name	Disk	Space used	Capacity	Percentage used	Projected space used in one year	Projected change	Time until full
sqm-sqlmonitor (local)	Log (H:)	9 GB	20 GB	47%	31 GB	+21 GB	in 5 months
sqm-sqlmonitor (local)	(C:)	31 GB	40 GB	78%	44 GB	+13 GB	in 6 months
sqm-sqlmonitor	Backup (E:)	13 GB	20 GB	66%	22 GB	+9 GB	in 10 months
sqm-sqlmonitor (local)	Data (F:)	11 GB	40 GB	27%	18 GB	+8 GB	> 1 year
ssc-db-n2	(C:)	43 GB	102 GB	42%	54 GB	+11 GB	> 1 year

We have now moved from a strictly reactive approach to a proactive one. We understand our current server resources, in this example represented by the disk space in use, and the amount of growth we're experiencing.

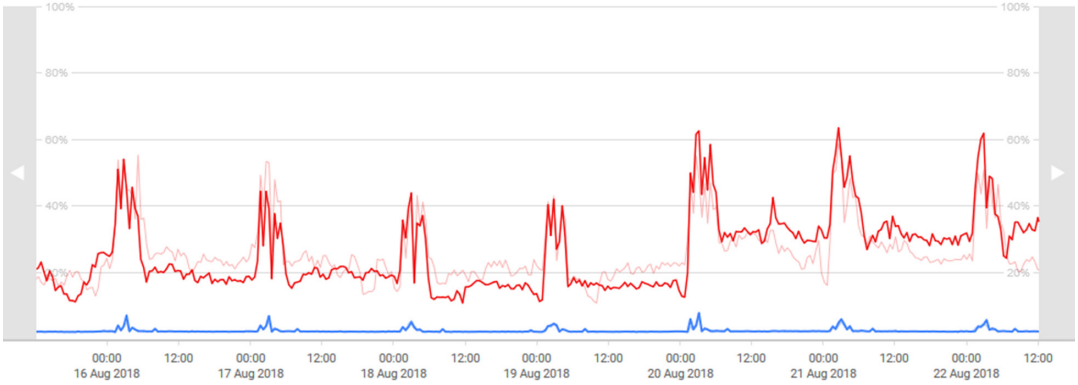
We could take any of the other server resources and apply similar styles of reporting, to identify those resources where current capacity will likely be insufficient to meet satisfy the requirements of the server workload, in short to medium term.

Before and After: Baselines

Another critical use of the monitoring data we collect is to create visual baselines. It is easy to assume that a baseline requires that you know what a metric looks like with no load on it at all, but that's the wrong way to think about it. A baseline for any metric represents at least one reference point, but more often a series of previous reference points, against which we can compare the current value.

How frequently do we hear: *"The server was slow yesterday"* or *"It was faster before we released the upgrade"* or *"It's slow right now"*? To validate the such claims, and find out the cause, we need resource usage data from yesterday in order to be able to compare it to today. For example, this chart shows CPU usage on two different nodes of the same SQL Server cluster, over a seven-day period:

You can see that one node has been running at a very low level, while the other has stepped up from processing at mostly below 20% to having sustained averages nearer 40%. If this trend continues, doubling every three or four days, we might be in trouble. So, let's extend this to see the same information from the preceding week:



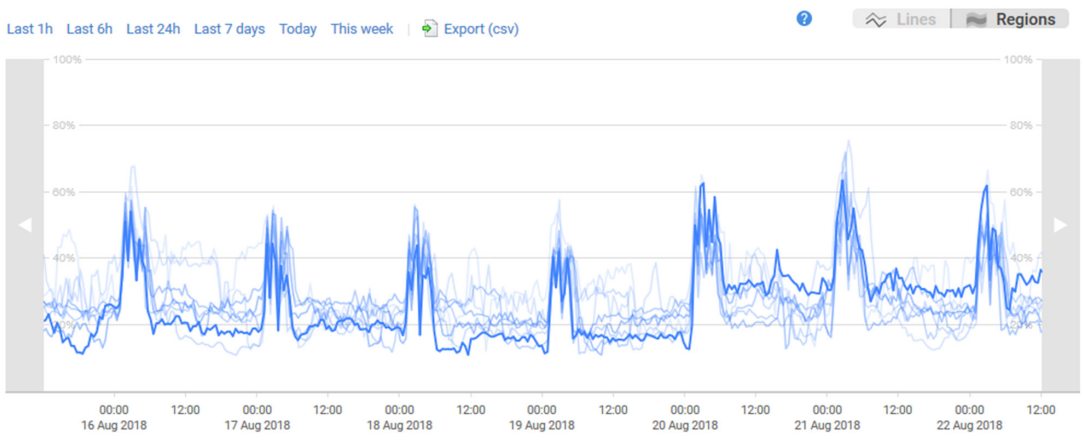
OK, we can relax just a little because we can see from the preceding week's baseline that the behavior then was also characterized by several days of lower CPU use followed by a spike to more sustained use for a few days, on the busy processor.

However, we can see also that CPU use on this processor is higher than for last week. Is this a disturbing trend? In support of possible optimization efforts, we need to determine if we're seeing what can best be described as normal behavior, or if we are in fact seeing a trend that will lead to pressure on this this resource unless we take action, such as looking for ways to spread the load better over available processors during those busier periods.

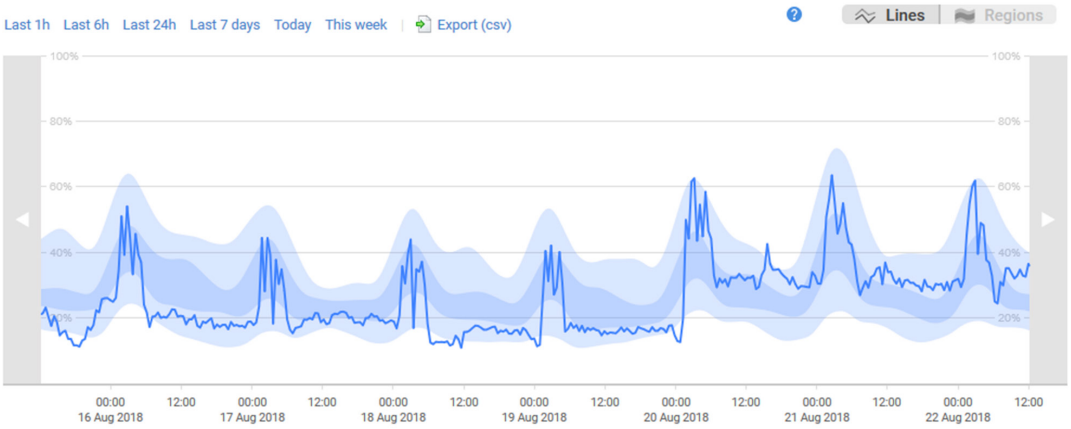
Definitions of Normal

While any resource can experience linear growth, similar to the disk growth in the first example, most systems will have a sustained period where things remain relatively normal, or resource use grows very slowly. Sometimes, if we look at data that is recorded over an insufficient period, we'll fail to distinguish normal behavior, from a worrying trend that requires corrective action.

Continuing our example, one of the processors seemed to indicate that it's behavior was changing. However, with only the two data sets for comparison, all we can really say is that they are different. If we add a bunch more data sets, say, the previous seven weeks' worth of data, we might be able to spot if we're seeing a trend in the information:



The darkest line is the current set of measures. Each dimmer line is going back in time. From this we can start to see that we're probably not seeing a major shift away from normal behavior. Instead of looking all the raw data for the past, we can average the information out to get a better picture of whether the current data set deviates from normal:



With the averages over time, we can tell for certain that the behavior of the last week falls well within normal behaviors. The darker zone represents the average behavior and the lighter zone represents the peaks and valleys. So, while there are some areas where normal behavior comes near the peaks, it doesn't exceed them in any way.

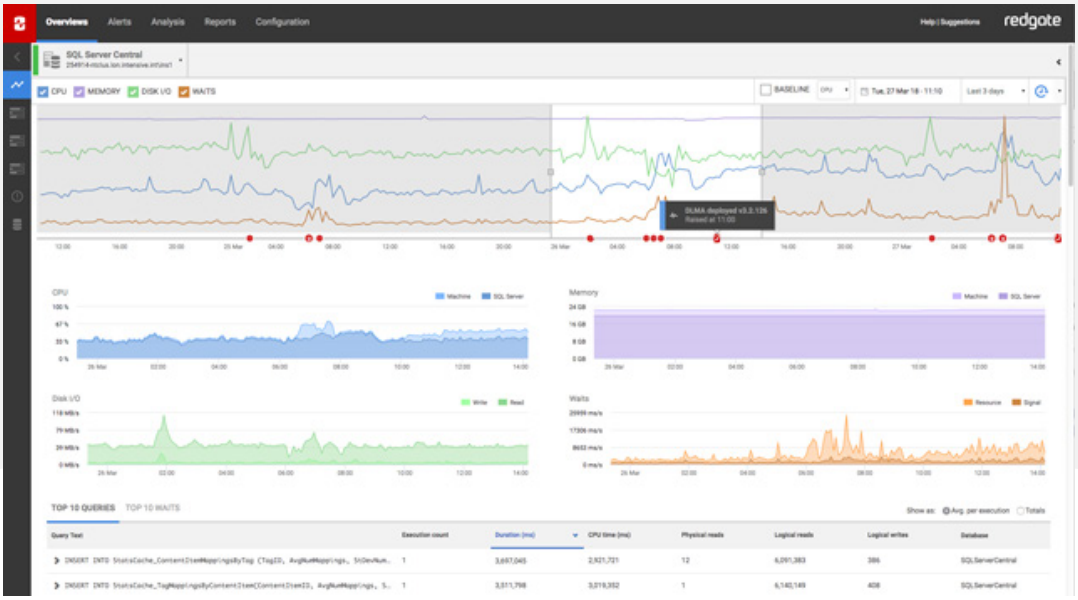
In this case, it's safe enough to say that this is within normal variance and we can focus on other potential issues. Similarly, this understanding of what is normal will also allow you to dismiss anomalies, and odd behavior, when that anomaly is just an outlier and not an indication of a broader problem.

Conclusion

The goal of database optimization is to improve the performance or availability or security of that database. Frequently, we make optimizations by finding ways to make the most efficient use of all available server and database resources, and one of the primary goals of any monitoring tool is to ensure proper management of these resources.

The tool needs to show us what we have, and how it is being used. As we gather the diagnostic data, we will need to retain it over time so that we can predict how usage will change. It also needs to help us understand the patterns of behavior we expect, in the database and server metrics associated with performance (or availability, or security, and so on), using baselines, so they we're able to spot sudden changes or worrying trends quickly.

With all this in place, you can better manage your estate to ensure that you are optimizing the use of resources within that estate.



And finally...

Regardless of how you monitor and manage your SQL Server estate, this whitepaper has helped you understand some of the principals that will ensure you are doing so efficiently, purposefully and with security in mind.

At Redgate, the leading provider of software for professionals working on the Microsoft data platform, we offer an ingeniously simple SQL Server monitoring solution that helps you to proactively monitor your estate so that you and your team can find and fix issues before they become problems.

SQL Monitor has powerful features that cover...

Alerting – Discover issues before they have an impact

Diagnosis – Uncover obstacles and find root causes

Performance – See what has the biggest impact on your system

Overview – View your SQL Server estate at a glance

Reporting – Share tailored reports about your servers' health

...and much more.

SQL Monitor

You can try our live online demo that runs on Redgate's own servers, or download your completely free 14-day trial to test it on your systems:

www.red-gate.com/sqlmonitor