Zehn Tipps für die Optimierung der SQL Server Leistung

Autoren: Patrick O'Keefe und Richard Douglas

Zusammenfassung

Die Optimierung der SQL Server Leistung kann eine enorme Herausforderung darstellen. Bei einer Recherche finden Sie sicherlich umfangreiche Informationen dazu, wie Leistungsprobleme ganz allgemein behoben werden können. Es gibt jedoch nicht viele detaillierte Informationen zu spezifischen Problemen und noch weniger Informationen dazu, wie Sie dieses spezifische Wissen konkret auf Ihre individuelle Umgebung anwenden.

Das vorliegende Dokument gibt Ihnen zehn Tipps an die Hand, mit denen Sie die SQL Server Leistung optimieren können. Dabei konzentrieren wir uns auf die Versionen SQL Server 2008 und 2012. Es gibt natürlich keine absoluten Listen der wichtigsten Tipps, aber mit den folgenden Ratschlägen bieten wir Ihnen einige hilfreiche Anhaltspunkte.

Als Datenbankadministrator verfügen Sie zweifellos über eigene Erfahrungen und haben Ihre eigenen Tipps, Tricks und Skripte zur Optimierung entwickelt. Wenn dem so ist, dann würden wir uns freuen, wenn Sie an den Diskussionen auf SQLServerPedia teilnehmen würden.

Einführung

Welche Ziele möchten Sie mit der Optimierung erreichen?

Jedes Unternehmen möchte seine SQL Server Bereitstellung optimal nutzen. Eine gesteigerte Effizienz Ihrer Datenbankserver setzt Systemressourcen frei, die für andere Aufgaben wie die Erstellung von Unternehmensberichten oder die Bearbeitung von Ad-hoc-Abfragen eingesetzt werden können. Damit Ihre Organisation optimal von ihren Hardwareinvestitionen profitiert, müssen Sie sicherstellen, dass SQL Server Rechenlasten und Anwendungsrechenlasten auf den Datenbankservern so schnell und effizient wie möglich ausgeführt werden.

Welchen Schwerpunkt Sie bei der Optimierung setzen, hängt ganz von Ihren Zielen ab. Vielleicht steht für Sie die Frage im Vordergrund, ob Sie mit Ihrer SQL Server Bereitstellung die beste Effizienz erzielen. Ein anderes Unternehmen fragt sich hingegen eventuell, ob sich seine Anwendung auch skalieren lässt. Die wichtigsten Schwerpunktvarianten bei der Optimierung sind folgende:

- Optimierung, um Servicestufenvereinbarungen (Service Level Agreements, SLAs) oder Vorgaben für die wichtigsten Leistungsindikatoren (Key Performance Indicators, KPIs) zu erfüllen
- Optimierung, um die Effizienz zu verbessern und dadurch Ressourcen für andere Aufgaben freizusetzen
- Optimierung, um Skalierbarkeit zu gewährleisten und dadurch in Zukunft SLA- und KPI-Vorgaben zu erfüllen

Versuchen Sie immer, die Skalierbarkeit und Effizienz aller Ihrer Datenbankserver zu maximieren – auch wenn Ihre Geschäfts-anforderungen derzeit erfüllt werden.

Denken Sie immer daran, dass es sich bei einer Optimierung um einen fortlaufenden Prozess handelt, nicht um eine einmalige Maßnahme.
Leistungsoptimierung ist ein kontinuierlicher Prozess. Wenn Sie beispielsweise eine Optimierung anstreben, um SLA-Ziele zu erfüllen, dann haben Sie diese Aufgabe irgendwann abgeschlossen. Wenn Sie jedoch optimieren, um die Effizienz zu verbessern oder Skalierbarkeit zu gewährleisten, sind Sie damit niemals wirklich fertig. Diese Art der Optimierung muss so lange fortgesetzt werden, bis die Leistung „gut genug" ist. Ist die Leistung der Anwendung zu einem zukünftigen Zeitpunkt nicht mehr gut genug, so muss der Optimierungszyklus erneut beginnen.

Was „gut genug" konkret bedeutet, ist für gewöhnlich durch unternehmensbezogene Vorgaben wie SLAs oder Anforderungen an den Systemdurchsatz definiert. Über diese Anforderungen hinaus sollten Sie jedoch stets bemüht sein, die Skalierbarkeit und Effizienz aller Datenbankserver zu maximieren – auch wenn Ihre Geschäftsanforderungen derzeit erfüllt werden.

Über dieses Dokument

Die Optimierung der SQL Server Leistung ist eine enorme Herausforderung. Es stehen umfangreiche allgemeine Informationen zu verschiedenen Datenpunkten zur Verfügung, so beispielsweise zu Leistungsindikatoren und dynamischen Verwaltungsobjekten (Dynamic Management Object, DMOs). Aber es gibt nur wenig Informationen zur weiteren Verwendung dieser Daten oder zu ihrer Auslegung. Dieses Dokument erläutert zehn wichtige und nützliche Tipps für die Praxis, mit deren Hilfe Sie einige dieser Daten in verwertbare Informationen umwandeln können.

Tipp Nr. 10. Mit der Methodik des Baselining und Benchmarking können Sie Probleme erkennen.

Überblick über den Baselining- und Benchmarking-Prozess Abbildung 1 veranschaulicht die einzelnen Schritte des Baselining-Prozesses. Im folgenden Abschnitt wird näher auf die wichtigsten Schritte in diesem Prozess eingegangen.

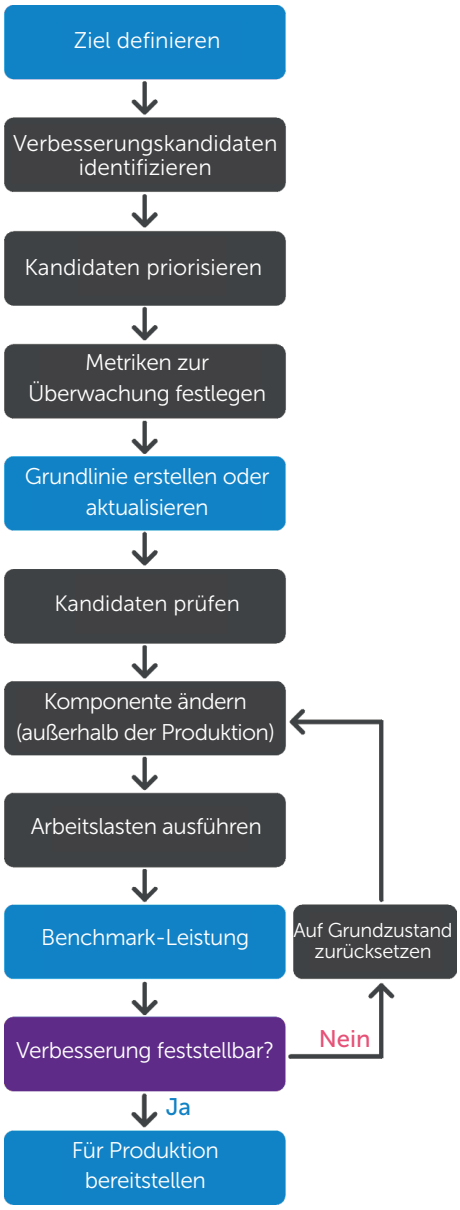

Abbildung 1: Der Baselining-Prozess

Bevor Sie Änderungen vornehmen, sollten Sie zunächst einen Bericht zur aktuellen Systemleistung erstellen. Wenn es um die Optimierung von Systemen geht, sind die meisten Administratoren versucht, ohne Umschweife mit dem Projekt zu beginnen. Vielleicht haben auch Sie schon einmal Ihren Wagen aus der Werkstatt abgeholt und hatten den Eindruck, dass er noch schlechter läuft als vor der Reparatur? In einem solchen Fall würden Sie sich gern beschweren, sind sich aber doch nicht wirklich sicher, ob Sie recht haben. Sie fragen sich eventuell auch, ob der Automechaniker das Problem, das Sie jetzt bemerken, verursacht hat oder ob es erst später aufgetaucht ist. Geht es um die Datenbankleistung, können Sie vor genau demselben Dilemma stehen.

Wenn Sie dieses Whitepaper gelesen haben, werden Sie viele neue Ideen haben und Ihre Baselining- und Benchmarking-Strategie unverzüglich implementieren wollen. Auch wenn der erste Schritt in Richtung Ziel nicht besonders aufregend ist, so ist er doch einer der wichtigsten: Sie müssen ermitteln, inwiefern Ihre aktuelle Umgebung die Kriterien erfüllt, die Sie ändern möchten.

Definieren Sie Ihre Ziele.
Bevor Sie etwas an Ihrem System ändern, müssen Sie entscheiden, was Sie erreichen möchten. Gibt es Aspekte Ihrer SLA, die hinsichtlich Leistung, Ressourcenverbrauch oder Kapazität optimiert werden müssen? Gibt es ein aktuelles Problem in der Produktion? Gab es Beschwerden in puncto Ressourcenzeit? Legen Sie klare Ziele fest.

Die meisten Administratoren sind für viele Datenbanken und Instanzen verantwortlich. Um folglich maximalen Nutzen aus jeder Maßnahme zu ziehen, sollten Sie sorgfältig überlegen, welche Anforderungen an ein bestimmtes System gestellt werden müssen, damit es optimale Leistung bringt und die Erwartungen der Benutzer erfüllt. Bei einer übermäßigen Analyse und Optimierung kann es sein, dass Sie unverhältnismäßig viel Zeit auf weniger wichtige Systeme verwenden, was negative Auswirkungen auf Ihre wesentlichen Produktionssysteme haben kann. Verschaffen Sie sich Klarheit darüber, was genau Sie mit Ihren Bewertungs- und Optimierungsmaßnahmen erzielen möchten. Priorisieren Sie Ihre Maßnahmen und sichern Sie sich idealerweise die Zustimmung und Unterstützung eines Stakeholders in Ihrem Unternehmen.

Ermitteln Sie die aktuellen Durchschnittswerte.
Nachdem Sie Ihre Ziele genau formuliert haben, müssen Sie entscheiden, nach welchen Kriterien der Erfolg gemessen werden soll. Welche Betriebssystemindikatoren, SQL Server Indikatoren, Ressourcenkennzahlen und anderen Datenpunkte können Ihnen den erforderlichen Einblick geben?

Nachdem Sie diese Liste erstellt haben, müssen Sie die typische Leistung Ihres Systems in Bezug auf die von Ihnen gewählten Kriterien definieren, die sogenannte Baseline. Sie müssen über einen angemessenen Zeitraum hinweg genügend Daten zusammentragen, um repräsentative Werte für die typische Leistung des Systems ermitteln zu können. Liegen Ihnen diese Daten vor, können Sie die durchschnittlichen Leistungswerte für den Zeitraum errechnen, um Ihre erste Baseline festzulegen. Nach der Modifikation Ihres Systems können Sie die neuen Benchmark-Werte mit den ursprünglichen Werten vergleichen und auf diese Weise objektiv den Effekt Ihrer Änderungen bewerten.

Untersuchen Sie nicht nur die Durchschnittswerte, sondern auch die Abweichungen von der Norm.
Behandeln Sie Durchschnittswerte stets mit Vorsicht. Berechnen Sie zumindest die Standardabweichung jedes einzelnen Indikators, damit Sie eine Vorstellung von der Variation im Laufe der Zeit erhalten. Denken Sie dabei an einen Bergsteiger, dem gesagt wird, sein Seil habe einen durchschnittlichen Durchmesser von 1 cm. Darauf verlässt er sich und lässt sich nach unten fallen. Er baumelt hundert Meter oder mehr über dem Abgrund und lächelt selbstgefällig. Dann sagt man ihm, dass das Seil an der dicksten Stelle 2 cm und an der dünnsten Stelle 0,1 cm misst. Er wird nicht mehr lächeln!

Wenn Sie sich mit Standardabweichungen nicht auskennen, ziehen Sie ein Einführungsbuch zum Thema Statistik zurate. Sie müssen nicht zu sehr ins Detail zu gehen, aber ein solides Grundwissen ist durchaus hilfreich.

Der wichtigste Grundsatz lautet also: Untersuchen Sie nicht nur die Durchschnittswerte, sondern auch die Abweichung von der Norm (Mittelwert). Definieren Sie die Normwerte. (Entsprechende Informationen finden Sie oftmals in Ihren SLAs). Ihre Aufgabe ist es nicht, die maximal mögliche Leistung zu erzielen. Vielmehr sollten Sie versuchen,

> Benchmarking gibt Ihnen gute Anhaltspunkte darüber, welches Verhalten als normal betrachtet werden kann. So lässt sich unnormales Verhalten leichter identifizieren.

> Ihre Aufgabe ist es nicht, die maximal mögliche Leistung zu erzielen. Vielmehr sollten Sie versuchen, Ihren Leistungszielen so nahe wie möglich zu kommen, und anschließend Abweichungen von diesen Zielen auf ein Minimum reduzieren.

Ihren Leistungszielen so nahe wie möglich zu kommen, und anschließend Abweichungen von diesen Zielen auf ein Minimum reduzieren. Alles andere wäre vergeudete Zeit und Mühe und könnte zudem eine unzureichende Auslastung der Infrastrukturressourcen bedeuten.

Wie viele Daten sind für eine Baseline erforderlich? Wie viele Daten Sie benötigen, um eine Baseline zu definieren, hängt davon ab, wie sehr Ihre Rechenlast im Laufe der Zeit variiert. Sprechen Sie mit Systemadministratoren, Endbenutzern und Anwendungsadministratoren. Sie haben in der Regel eine gute Vorstellung von den Nutzungsmustern. Sie sollten genug Daten zusammentragen, um Zeiten mit niedriger, durchschnittlicher und hoher Auslastung abzudecken. Außerdem müssen Sie ermitteln, in welchem Umfang und mit welcher Häufigkeit die Rechenlast variiert. Bei Systemen mit vorhersehbaren Mustern müssen Sie weniger Daten sammeln. Je größer die Variabilität ist, desto kürzer ist auch das einzelne Messintervall und desto länger ist der erforderliche Messzeitraum, um eine zuverlässige Baseline ausarbeiten zu können. Kommen wir hier noch einmal auf unseren Bergsteiger zurück: Je mehr von dem Seil Sie untersuchen, desto größer ist die Wahrscheinlichkeit, dass Sie die Unterschiede im Durchmesser bemerken. Auch die Frage, wie wichtig ein System ist und welche Folgen sein Ausfall hätte, ist ausschlaggebend dafür, wie sorgfältig Sie es untersuchen müssen und wie zuverlässig Ihre Testdaten sein müssen.

Speichern Sie die Daten effizient. Je mehr Parameter Sie untersuchen und je häufiger Sie sie erfassen, desto größer ist der gesammelte Datensatz. Es mag wie eine Binsenweisheit klingen, aber Sie müssen immer die Kapazität bedenken, die für die Speicherung Ihrer Messdaten erforderlich ist. Sobald Sie über die ersten Daten verfügen, dürfte es kein Problem sein, das zukünftige Wachstum des Repositorys einzuschätzen. Bei einer Überwachung über einen längeren Zeitraum hinweg sollten Sie eine regelmäßige Aggregation historischer Daten in Betracht ziehen, damit das Repository nicht zu umfangreich wird.

Um Leistungseinbußen zu vermeiden, sollte sich das Repository der Messdaten nicht am gleichen Speicherort befinden wie die überwachte Datenbank.

Nehmen Sie jeweils nur eine begrenzte Anzahl an Änderungen vor. Begrenzen Sie nach Möglichkeit die Anzahl an Änderungen, die Sie zwischen den einzelnen Benchmarks vornehmen. Planen Sie Ihre Modifikationen so, dass Sie immer nur eine bestimmte Hypothese prüfen. Auf diese Weise können Sie jeden einzelnen Verbesserungskandidaten optimal untersuchen und gegebenenfalls ausschließen. Wenn Sie dann eine Lösung ins Auge fassen, können Sie genau nachvollziehen, warum Sie eine Änderung im Verhalten sehen. Oft ergeben sich so potenzielle Möglichkeiten für weitere Verbesserungen.

Analysieren Sie die Daten. Nachdem Sie Änderungen an Ihrem System vorgenommen haben, müssen Sie natürlich ermitteln, ob diese Änderungen auch die erwünschte Wirkung hatten. Dafür müssen Sie die Messungen, die Sie für Ihre ursprüngliche Baseline vorgenommen haben, über einen ähnlich repräsentativen Zeitraum hinweg wiederholen. Anschließend können Sie die zwei Baselines vergleichen und

- **Bestimmen, ob Ihre Änderungen die erwünschte Wirkung hatten:** Wenn Sie eine Konfigurationseinstellung justieren, einen Index optimieren oder SQL-Code ändern, können Sie anhand der Baseline beurteilen, ob die jeweilige Änderung den erhofften Effekt hatte. Wenn Sie Beschwerden über langsame Systemleistung erhalten, können Sie nachprüfen, ob diese Aussage in Bezug auf die Datenbank tatsächlich zutrifft.

Der häufigste Fehler unerfahrener Datenbankadministratoren ist es, zu früh Schlussfolgerungen zu ziehen. Es kommt nur zu oft vor, dass Administratoren sich in Sicherheit wähnen, wenn sie eine sofortige und erkennbare Leistungssteigerung beobachten, nachdem sie eine oder mehrere Änderungen vorgenommen haben. Sie stellen die Änderungen in der Produktion bereit und senden sofort E-Mails an alle Benutzer, in denen sie verkünden, das Problem sei gelöst. Aber die Freude über die schnelle Behebung kann schnell wieder verfliegen, wenn dieselben Probleme kurze Zeit später erneut auftauchen oder bisher unbekannte Nebenwirkungen neue Probleme verursachen. Nicht selten ist die Situation schlussendlich schlimmer, als sie es ursprünglich war. Wenn Sie der Meinung sind, die Lösung für ein Problem gefunden zu haben, testen Sie sie und bewerten Sie die Ergebnisse in Bezug auf Ihre Baseline. Nur so können Sie mit Sicherheit wissen, dass Sie Ihr System tatsächlich verbessert haben.

- **Bestimmen, ob eine Änderung unerwartete Nebenwirkungen hatte:** Anhand einer Baseline sind Sie ebenfalls in der Lage, objektiv zu erkennen, ob eine Änderung einen Indikator oder einen Messwert in einer Weise beeinflusst hat, die Sie nicht erwartet hatten.
- **Probleme erkennen, bevor sie entstehen**: Eine Baseline ermöglicht es Ihnen, akkurate Leistungsnormen für typische Rechenlastbedingungen festzulegen. So können Sie auf der Grundlage von derzeitigen Trends der Ressourcenauslastung oder auf der Grundlage von Rechenlastprognosen für zukünftige Szenarien besser vorhersagen, ob und wann Sie in Zukunft auf Probleme stoßen werden. Nehmen wir beispielsweise an, Sie erstellen einen Kapazitätsplan. Wenn Sie die aktuelle typische Ressourcenauslastung pro verbundenem Benutzer hochrechnen, können Sie vorhersagen, wann es auf Ihren Systemen zu einem Engpass bei den Benutzerverbindungen kommen wird.
- **Probleme effektiver beheben:** Haben Sie auch schon einmal viele Tage und Nächte bei dem Versuch zugebracht, ein Leistungsproblem Ihrer Datenbank zu beheben, um schließlich herauszufinden, dass die eigentliche Ursache überhaupt nicht bei der Datenbank lag? Wenn Sie eine Baseline festlegen, ist es viel einfacher, Ihre Datenbankinstanz als Verursacher auszuschließen und den tatsächlichen „Schuldigen" zu ermitteln. Nehmen wir beispielsweise an, die Arbeitsspeicherauslastung schnellt plötzlich in die Höhe. Sie bemerken, dass die Anzahl an Verbindungen stark ansteigt und einen Wert erreicht, der deutlich über Ihrer Baseline liegt. Ein kurzer Anruf beim Anwendungsadministrator bestätigt, dass im E-Store ein neues Modul bereitgestellt wurde. Es dauert nicht lange und Sie haben herausgefunden, dass der neue Junior-Entwickler Code schreibt, der Datenbankverbindungen nicht wie erforderlich freigibt. Ich bin sicher, dass Ihnen noch viele ähnliche Szenarien einfallen würden.

Der Ausschluss aller Faktoren, die nicht für ein Problem verantwortlich sind, spart eine Menge Zeit ein. Sie erhalten einen viel besseren Überblick und können sich voll und ganz auf die tatsächliche Ursache des Problems konzentrieren. Es gibt zahlreiche Beispiele, bei denen sich Systemindikatoren mit SQL Server Leistungsindikatoren vergleichen lassen, um schnell zu beurteilen, ob die Datenbank an dem Problem beteiligt ist oder nicht. Wenn die häufigsten Fehlerkandidaten ausgeschlossen sind, können Sie konzentriert nach wesentlichen Abweichungen von der Baseline suchen, relevante Indikatoren zusammentragen und der eigentlichen Ursache des Problems auf den Grund gehen.

Wiederholen Sie den Baselining-Prozess so oft wie nötig.
Eine gute Optimierung ist ein iterativer und wissenschaftlicher Prozess. Die in diesem Dokument vorgestellten Tipps stellen einen optimalen Ausgangspunkt dar – sind aber auch tatsächlich nicht mehr als das. Die Leistungsoptimierung ist ein in hohem Maß individueller Prozess, der von Design, Zusammensetzung und Nutzung jedes einzelnen Systems abhängt.

Die Methodik des Baselining und Benchmarking ist der Grundpfeiler für eine gute und zuverlässige Leistungsoptimierung. Sie liefert den Plan, die Referenz und die Meilensteine, die Sie benötigen, um zu wissen, was Sie erreichen möchten und wie Sie es erreichen können. Nur so können Sie sicher sein, dass Sie während des Projekts nicht den falschen Weg einschlagen. Mit einem strukturierten Ansatz können Sie zuverlässige und konsistente Leistung für Ihr gesamtes Datenbankportfolio garantieren.

Tipp Nr. 9. Leistungsindikatoren liefern Ihnen schnell nützliche Informationen über aktuelle Abläufe.

Gründe für die Überwachung von Leistungsindikatoren

Oftmals wird in Bezug auf die SQL Server Leistungsoptimierung die folgende Frage gestellt: „Welche Indikatoren muss ich überwachen?" Es gibt zwei gute Gründe, die Leistungsindikatoren bei der SQL Server Verwaltung immer im Blick zu haben:

- Verbesserung der betrieblichen Effizienz
- Vermeidung von Leistungsengpässen

Auch wenn Überschneidungen bestehen, können Sie anhand dieser zwei Gründe problemlos eine Reihe von Datenpunkten zur Überwachung auswählen.

Überwachung von Leistungsindikatoren zur Verbesserung der betrieblichen Effizienz Bei der Betriebsüberwachung wird die allgemeine Ressourcennutzung geprüft. So können Sie u. a. folgende Fragen beantworten:

- Verfügt der Server nur noch über geringe Ressourcen bei Prozessor (CPU, Central Processing Unit), Festplattenspeicher oder Arbeitsspeicher?
- Können die Datendateien wachsen, wenn sie es müssen?
- Verfügen Datendateien mit fester Größe über ausreichend Speicherplatz für alle Daten?

Wie viele Daten Sie benötigen, um eine Baseline zu definieren, hängt davon ab, wie sehr Ihre Rechenlast im Laufe der Zeit variiert.

Nehmen Sie zwischen den Benchmarks jeweils nur eine begrenzte Anzahl an Änderungen vor, um die Wirkungen der einzelnen Änderungen genau bewerten zu können.

Sie können die Daten auch zur Bestimmung von Trends nutzen. Ein anschauliches Beispiel hierfür ist die Erfassung der Größen aller Datendateien, um einen Trend bezüglich ihrer Wachstumsraten zu ermitteln und zukünftige Ressourcenanforderungen zu prognostizieren.

Bei der Beantwortung dieser Fragen helfen die folgenden Indikatoren::

Indikator	Zweck
Prozessor\% Prozessorzeit	Überwachung des CPU-Verbrauchs auf dem Server
Logischer Datenträger\Freie MB	Überwachung des freien Speicherplatzes auf den Datenträgern
MSSQL$Instance:Datenbanken\Größe der Datendateien (KB)	Wachstumstrend im Laufe der Zeit
Arbeitsspeicher\Seiten/s	Überprüfung auf Auslagerungen, ein starker Hinweis darauf, dass die Arbeitsspeicherressourcen eventuell nicht ausreichen
Arbeitsspeicher\Verfügbare MB	Anzeige des physischen Arbeitsspeichers, der für die Systemnutzung verfügbar ist

Überwachung von Leistungsindikatoren zur Vermeidung von Leistungsengpässen
Bei der Überwachung von Engpässen stehen eher leistungsbezogene Aspekte im Mittelpunkt. Anhand der gesammelten Daten können Sie u. a. folgende Fragen beantworten:

- Gibt es einen CPU-Engpass?
- Gibt es einen E/A-Engpass?
- Sind die wichtigsten SQL Server Subsysteme in einwandfreiem Zustand, beispielsweise der Puffer-Cache und der Plan-Cache?
- Gibt es Konflikte in der Datenbank?

Bei der Beantwortung dieser Fragen helfen die folgenden Indikatoren:

Indikator	Zweck
Prozessor\% Prozessorzeit	Die Überwachung des CPU-Verbrauchs ermöglicht Ihnen die Ermittlung von Engpässen auf dem Server (erkennbar an anhaltend hoher Nutzung).
Hoher Prozentwert bei der Signalwartezeit	Die Signalwartezeit ist die Zeit, die ein Arbeits-Thread auf CPU-Zeit warten muss, nachdem andere Wartezeiten (für Sperren, Latches usw.) beendet wurden. Müssen Prozesse auf die CPU warten, deutet das auf einen CPU-Engpass hin. Signalwartezeiten können in SQL Server 2000 durch die Ausführung von „DBCC SQLPERF(waitstats)“ und in SQL Server 2005 durch die Abfrage von „sys.dm_os_wait_stats“ ermittelt werden.
Physischer Datenträger\Durchschnittl. Warteschlangenlänge der Festplatte	Überprüfung auf Datenträgerengpässe: Ist der Wert größer als 2, so besteht wahrscheinlich ein Datenträgerengpass.
MSSQL$Instance:Puffer-Manager\Lebenserwartung von Seiten	Die Lebenserwartung einer Seite ist die Anzahl an Sekunden, die sie im Puffer-Cache verbleibt. Ein niedriger Wert zeigt an, dass Seiten bereits nach kurzem Verbleib im Cache entfernt werden, was dessen Effektivität mindert.
MSSQL$Instance:Plan-Cache\Cache-Trefferquote	Eine niedrige Cache-Trefferquote für den Plan-Cache bedeutet, dass Ausführungspläne nicht wiederverwendet werden.
MSSQL$Instance:Allgemeine Statistiken\Blockierte Prozesse	Lange Blockierungen weisen auf Ressourcenkonflikte hin.

Tipp Nr. 8. Die Änderung von Servereinstellungen kann für eine stabilere Umgebung sorgen.

Dass eine Änderung der Einstellungen innerhalb eines Produkts zu einer verbesserten Stabilität führen soll, klingt zunächst paradox, ist jedoch in diesem Fall tatsächlich zutreffend. Als Datenbankadministrator ist es Ihre Aufgabe, konsistente Leistung für Benutzeranfragen an Anwendungen sicherzustellen. Wenn Sie die Einstellungen nicht wie nachfolgend in diesem Dokument erläutert ändern, kann es unter Umständen zu Szenarien kommen, die die Leistung für Ihre Benutzer ohne Vorwarnung beeinträchtigen. Diese Optionen finden Sie in der Ansicht „Sys. Configurations", die serverweite Konfigurationsoptionen sowie zusätzliche Informationen auflistet. Das Attribut „Is_Dynamic" in „Sys.Configurations" zeigt an, ob die SQL Server Instanz nach einer Konfigurationsänderung neu gestartet werden muss. Um die Änderungen zu implementieren, rufen Sie die gespeicherte Prozedur „sp_configure" mit den relevanten Parametern auf.

Die Einstellungen für den minimalen und den maximalen Serverarbeitsspeicher gewährleisten ein bestimmtes Leistungsniveau.
Gehen wir einmal von einem Aktiv/Aktiv-Cluster aus (oder sogar von einem einzigen Host mit mehreren Instanzen). Wir können bestimmte Konfigurationsänderungen vornehmen, die im Fall eines Failovers, bei dem sich beide Instanzen auf demselben Rechner befinden, die Erfüllung unserer SLAs sicherstellen.

In diesem Szenario würden wir die Einstellungen für den minimalen und den maximalen Serverarbeitsspeicher ändern, damit der physische Host über ausreichend Arbeitsspeicher für beide Instanzen verfügt, ohne ständig versuchen zu müssen, den Arbeitssatz der anderen Instanz rigoros zu begrenzen. Eine ähnliche Konfigurationsänderung kann vorgenommen werden, um bestimmte Prozessoren zu verwenden und dadurch ein bestimmtes Leistungsniveau zu gewährleisten.

Bitte beachten Sie, dass die Einstellung des maximalen Serverarbeitsspeichers nicht nur für Instanzen in einem Cluster eine geeignete Option ist, sondern auch für Instanzen, die Ressourcen mit anderen Anwendungen gemeinsam nutzen. Wenn SQL Server zu viel Arbeitsspeicher nutzt, so begrenzt das Betriebssystem unter Umständen die ihm zugewiesene Arbeitsspeichergröße drastisch, damit für das Betriebssystem selbst oder für alle anderen Anwendungen weiterhin ausreichend Arbeitsspeicherkapazität zur Verfügung steht.

- **SQL Server 2008**: In SQL Server 2008 R2 und niedriger begrenzten die Einstellungen für den minimalen und den maximalen Serverarbeitsspeicher nur die Menge an Arbeitsspeicher, die der Pufferpool verbrauchte – genauer gesagt nur Zuordnungen einzelner Seiten mit 8 KB. Wenn also Prozesse außerhalb des Pufferpools durchgeführt wurden (z. B. erweiterte gespeicherte Prozeduren, CLR oder andere Komponenten wie Integration Services, Reporting Services oder Analysis Services), musste der Wert noch einmal heruntergesetzt werden.
- **SQL Server 2012**: In SQL Server 2012 sieht dies etwas anders aus, da ein zentraler Speicher-Manager eingeführt wurde. Dieser Speicher-Manager ermöglicht die Zuordnung mehrerer Seiten, z. B. von Seiten mit umfangreichen Daten, und die Zuweisung zwischengespeicherter Pläne mit mehr als 8 KB. Der Speicherplatz umfasst jetzt zudem einige CLR Funktionen.

Zwei Serveroptionen können indirekt die Leistung optimieren.
Es gibt keine Optionen, die auf direkte Weise die Leistung verbessern, aber es gibt zwei Optionen, die indirekt zur Leistungsoptimierung beitragen können.

- **Standardmäßige Komprimierung von Sicherungen**: Diese Option bewirkt, dass Sicherungen standardmäßig komprimiert werden. Auch wenn es dadurch zu zusätzlichen CPU-Zyklen während der Komprimierung kommen kann, so werden in der Regel im Vergleich zu einer unkomprimierten Sicherung weniger CPU-Zyklen insgesamt verwendet, da weniger Daten auf die Festplatte geschrieben werden. Je nach Ihrer E/A-Architektur kann die Aktivierung dieser Option auch E/A-Konflikte reduzieren.
- Die zweite Option wird eventuell in einem zukünftigen Tipp zum Plan-Cache näher erläutert. Sie müssen sich gedulden und abwarten, ob dieser Tipp es in die Liste unserer Top 10 geschafft hat.

Ein weiser Rat für die Fehlerbehebung: mögliche Verursacher ausschließen oder den Schuldigen finden!

Tipp Nr. 7. Ermitteln Sie störende Abfragen im Plan-Cache.

Sobald Sie einen Engpass identifiziert haben, müssen Sie ermitteln, welche Rechenlast diesen Engpass verursacht. Dies ist seit der Einführung von DMOs in SQL Server 2005 viel einfacher geworden. Benutzern von SQL Server 2000 und älteren Version steht nur Profiler oder die Ablaufverfolgung zur Verfügung (mehr dazu in Tipp Nr. 6).

Die Überwachung von Leistungsindikatoren hilft Ihnen, die betriebliche Effizienz zu verbessern und Leistungsengpässe zu vermeiden.

Diagnose eines CPU-Engpasses Wenn Sie in SQL Server einen CPU-Engpass identifiziert haben, müssen Sie zunächst ermitteln, wer die größten CPU-Verbraucher auf dem Server sind. Dies ist durch eine sehr einfache Abfrage von „sys.dm_exec_query_stats" möglich:

```
SELECT TOP 50
    qs.total_worker_time / execution_count AS avg_worker_time,
    substring (st.text, (qs.statement_start_offset / 2) + 1,
       (  ( CASE qs.statement_end_offset WHEN -1
            THEN datalength (st.text)
            ELSE qs.statement_end_offset END
         - qs.statement_start_offset)/ 2)+ 1)
      AS statement_text,
   *
FROM sys.dm_exec_query_stats AS qs
   CROSS APPLY sys.dm_exec_sql_text (qs.sql_handle) AS st
ORDER BY
avg_worker_time DESC
```

Der größte Vorteil dieser Abfrage besteht darin, dass Sie „CROSS APPLY" und „sys.dm_exec_sql_text" verwenden können, um eine SQL-Anweisung zu erhalten, die Sie analysieren können.

Diagnose eines E/A-Engpasses Ähnliches gilt für die Diagnose eines E/A-Engpasses:

```
SELECT TOP 50
      (total_logical_reads + total_logical_writes) AS total_logical_io,
      (total_logical_reads / execution_count) AS avg_logical_reads,
      (total_logical_writes / execution_count) AS avg_logical_writes,
      (total_physical_reads / execution_count) AS avg_phys_reads,
      substring (st.text,
            (qs.statement_start_offset / 2) + 1,
            ((CASE qs.statement_end_offset WHEN -1
               THEN datalength (st.text)
              ELSE qs.statement_end_offset END
              - qs.statement_start_offset)/ 2)+ 1)
            AS statement_text,
    *
FROM sys.dm_exec_query_stats AS qs
    CROSS APPLY sys.dm_exec_sql_text (qs.sql_handle) AS st
ORDER BY total_logical_io DESC
```

Tipp Nr. 6. SQL Profiler kann eine große Hilfe sein.

SQL Server Profiler und erweiterte Ereignisse

SQL Server Profiler ist ein natives Tool, das im Lieferumfang von SQL Server enthalten ist. Mit ihm können Sie eine Ablaufverfolgungsdatei erstellen, die Ereignisse in SQL Server erfasst. Die Ablaufverfolgung liefert Ihnen wertvolle Informationen zu Ihrer Rechenlast und zu leistungsschwachen Abfragen. In diesem Whitepaper kann nicht genauer auf die Verwendung des Profiler Tools eingegangen werden. Weitere Informationen zu SQL Server Profiler finden Sie in unserem Video-Tutorial auf SQLServerPedia.

Auch wenn SQL Server Profiler als veraltet gilt und in SQL Server 2012 durch die erweiterten Ereignisse ersetzt wurde, so soll darauf hingewiesen werden, dass dies nur für die Datenbank-Engine und nicht für die SQL Server Analysis Services gilt. Profiler kann nach wie vor für viele SQL Server Umgebungen wichtige Echtzeiteinblicke in die Abläufe von Anwendungen bereitstellen.

Auf die Verwendung erweiterter Ereignisse einzugehen, würde den Rahmen dieses Dokuments sprengen. Eine detaillierte Beschreibung dieser Funktion finden Sie im Whitepaper „How to Use SQL Server's Extended Events and Notifications to Proactively Resolve Performance Issues" (Verwenden der erweiterten Ereignisse und Benachrichtigungen von SQL Server zum proaktiven Lösen von Leistungsproblemen). Hier soll nur gesagt sein, dass erweiterte Ereignisse in SQL Server 2008 eingeführt und in SQL Server 2012 aktualisiert wurden, um mehr Ereignisse und eine lang erwartete Benutzeroberfläche bereitzustellen.

Bitte beachten Sie, dass Profiler die „ALTER TRACE"-Berechtigung erfordert.

Verwendung von SQL Profiler

Hier haben wir Ihnen Schritt für Schritt beschrieben, wie Sie auf der Datenerfassung durch den Systemmonitor (Perfmon) aufbauen können und die Informationen zur Ressourcennutzung mit den Ereignisdaten aus SQL Server in Verbindung setzen:

1. Öffnen Sie Perfmon.
2. Falls auf Ihrem System noch kein Datensammler konfiguriert ist, erstellen Sie jetzt einen mithilfe der erweiterten Option. Die Indikatoren aus Tipp Nr. 9 können Ihnen als Richtlinien dienen. Starten Sie den Datensammler noch nicht.
3. Öffnen Sie Profiler.
4. Erstellen Sie eine neue Ablaufverfolgung, indem Sie festlegen, welche Instanz, welche Ereignisse und welche Spalten Sie überwachen möchten, und das Ziel definieren.
5. Starten Sie die Ablaufverfolgung.
6. Wechseln Sie zurück zu Perfmon, um den Datensammler zu starten.
7. Lassen Sie beide Sitzungen laufen, bis die erforderlichen Daten erfasst wurden.
8. Beenden Sie die Profiler Ablaufverfolgung. Speichern Sie die Ablaufverfolgung und schließen Sie sie.
9. Wechseln Sie zu Perfmon und beenden Sie den Datensammler.
10. Öffnen Sie die zuvor gespeicherte Ablaufverfolgungsdatei in Profiler.
11. Klicken Sie auf „Datei" und anschließend auf „Leistungsdaten importieren".
12. Navigieren Sie zur Datendatei der Datensammlung und wählen Sie die gewünschten Leistungsindikatoren aus.

Jetzt können Sie die Leistungsindikatoren in Kombination mit der Profiler Ablaufverfolgungsdatei anzeigen (siehe Abbildung 2) und zueinander in Bezug setzen, was eine deutlich schnellere Behebung der Engpässe ermöglicht.

Zusatztipp: Die oben aufgeführten Schritte verwenden die Client-Oberfläche. Zur Einsparung von Ressourcen wäre eine serverseitige Ablaufverfolgung effizienter. Informationen zum Starten und Beenden der serverseitigen Ablaufverfolgung finden Sie in der Onlinedokumentation.

Bei der Betriebsüberwachung wird die allgemeine Ressourcennutzung geprüft. Bei der Überwachung von Engpässen stehen eher leistungsbezogene Aspekte im Mittelpunkt.

Das Attribut „Is_ Dynamic“ in „Sys. Configurations“ zeigt an, ob die SQL Server Instanz nach einer Konfigurationsänderung neu gestartet werden muss.

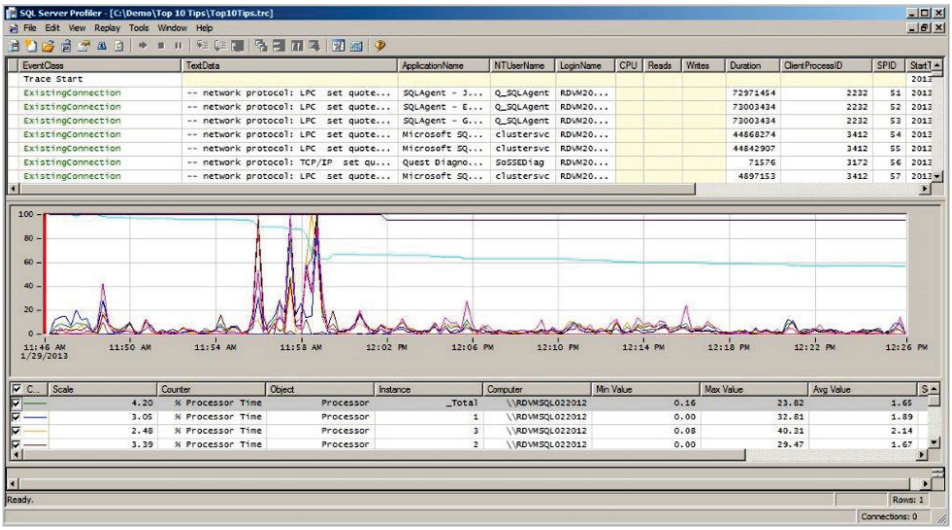

Abbildung 2: Eine korrelierte Ansicht der Leistungsindikatoren und der Profiler Ablaufverfolgungsdatei

Tipp Nr. 5. Konfigurieren Sie SAN-Systeme für optimale SQL Server Leistung.

SAN (Storage Area Network)-Speichersysteme sind extrem nützliche Systeme. Mit ihnen können Sie Massenspeicher auf ganz einfache Weise bereitstellen und verwalten. SAN-Speichersysteme können so konfiguriert werden, dass sie schnelle SQL Server Leistung garantieren – oftmals sind sie es jedoch nicht. Organisationen implementieren SAN-Systeme in der Regel, um Massenspeicher zu konsolidieren und die Verwaltung zu vereinfachen, nicht zur Leistungsoptimierung. Erschwerend kommt hinzu, dass sie für gewöhnlich keine direkte Kontrolle darüber haben, wie die Provisionierung auf den SANs abläuft. Folglich ist das SAN-System oftmals mit einem einzigen logischen Datenträger (Volume) konfiguriert, in dem alle Datendateien abgelegt werden müssen.

Best Practices für die Konfiguration von SAN-Systemen für optimale E/A-Leistung

Die Speicherung aller Dateien auf einem einzigen Volume ist im Allgemeinen keine gute Idee, wenn Sie optimale E/A-Leistung erreichen wollen. Empfehlenswert sind die folgenden Best Practices:

- **Legen Sie Protokolldateien auf eigenen Volumes** und getrennt von den Datendateien ab. Bei Protokolldateien werden fast ausschließlich sequenzielle Schreibvorgänge vorgenommen und keine Lesevorgänge. (Zu den Ausnahmen gehören Datenbankspiegelung und AlwaysOn Verfügbarkeitsgruppen.) Die Konfigurationen sollten daher immer auf eine schnelle Schreibleistung ausgerichtet sein.
- **Legen Sie „tempdb“ auf einem eigenen Volume ab**. Die Datenbank „tempdb“ wird innerhalb von SQL Server für zahlreiche verschiedene Zwecke verwendet, daher ist es von Vorteil, wenn sie sich auf einem eigenen E/A-Subsystem befindet. Für eine weitere Optimierung der Leistung sind zunächst weitere Statistikwerte erforderlich.
- **Ziehen Sie die Erstellung mehrerer Datendateien und Dateigruppen in VLDBs in Betracht**, um von parallelen E/A-Abläufen zu profitieren.
- **Legen Sie Sicherungen auf eigenen Laufwerken ab** , um die Redundanz zu optimieren und E/A-Konflikte mit anderen Volumes während Wartungsperioden zu reduzieren.

Sammlung von Daten

Natürlich gibt es die Windows Leistungsindikatoren, die Ihnen einen Überblick über Ihr System geben, so wie Windows es sieht. (Vergessen Sie nicht, die Rohwerte gemäß der RAID-Konfiguration anzupassen.) SAN-Anbieter stellen oftmals ihre eigenen Leistungsdaten bereit.

SQL Server liefert ebenfalls E/A-Informationen auf Dateiebene:

- **Versionen vor SQL 2005**:Verwenden Sie die Funktion „fn_virtualfilestats“.
- **Höhere Versionen**:Verwenden Sie die dynamische Verwaltungsfunktion „sys.dm_io_virtual_ file_stats“.

Durch die Verwendung dieser Funktion im folgenden Code können Sie:

- E/A-Raten für Lese- und Schreibvorgänge ableiten
- Den E/A-Durchsatz ermitteln
- Durchschnittszeiten pro E/A ermitteln
- E/A-Wartezeiten anzeigen

```
SELECT db_name (a.database_id) AS [DatabaseName],
      b.name AS [FileName], a.File_ID AS [FileID],
      CASE WHEN a.file_id = 2 THEN 'Log' ELSE 'Data' END AS [FileType],
      a.Num_of_Reads AS [NumReads],
      a.num_of_bytes_read AS [NumBytesRead],
      a.io_stall_read_ms AS [IOStallReadsMS],
      a.num_of_writes AS [NumWrites],
      a.num_of_bytes_written AS [NumBytesWritten],
      a.io_stall_write_ms AS [IOStallWritesMS],
      a.io_stall [TotalIOStallMS],
      DATEADD (ms, -a.sample_ms, GETDATE ()) [LastReset],
      ( (a.size_on_disk_bytes / 1024) / 1024.0) AS [SizeOnDiskMB],
      UPPER (LEFT (b.physical_name, 2)) AS [DiskLocation]
  FROM sys.dm_io_virtual_file_stats (NULL, NULL) a
    JOIN sys.master_files b
      ON a.file_id = b.file_id AND a.database_id = b.database_id
ORDER BY a.io_stall DESC;
```

Analyse der Daten

Bitte beachten Sie besonders den Wert für „LastReset" in der Abfrage. Er gibt an, wann der SQL Server Service zuletzt gestartet wurde. Da Daten von dynamischen Verwaltungsobjekten nicht persistent sind, sollten Sie für alle Daten prüfen, wie lange der Service zum Zeitpunkt ihrer Erfassung bereits ausgeführt worden ist. Tun Sie dies nicht, kann es zu falschen Annahmen kommen.

Anhand dieser Zahlen können Sie schnell die Dateien einkreisen, die für den Verbrauch von E/A-Bandbreite verantwortlich sind, und folgende Fragen stellen:

- Sind diese E/A-Operationen notwendig? Fehlt ein Index?
- Ist eine bestimmte Tabelle oder ein bestimmter Index in einer Datei für den Verbrauch verantwortlich? Kann ich diesen Index oder diese Tabelle in eine andere Datei oder auf ein anderes Volume verschieben?

Tipps für die Hardwareoptimierung Wenn sich die Datenbankdateien am korrekten Speicherort befinden und alle Objekt-Hotspots identifiziert und auf verschiedenen Volumes abgelegt wurden, müssen Sie sich eingehend Ihrer Hardware zuwenden.

Die Optimierung der Hardware ist ein eigenes Thema, das über den Rahmen dieses Dokuments hinausgehen würde. Es gibt jedoch einige Best Practices und Tipps, die sie erleichtern und die ich hier an Sie weitergeben möchte:

- Verwenden Sie nicht die standardmäßige Größe der Zuordnungseinheiten, wenn Sie Volumes für SQL Server erstellen. SQL Server verwendet Erweiterungen mit 64 KB, daher gilt dieser Wert als Mindestwert.
- Stellen Sie sicher, dass Ihre Partitionen korrekt ausgerichtet sind. Jimmy May hat ein erstklassiges Whitepaper zu diesem Thema verfasst. Falsch ausgerichtete Partitionen können die Leistung um bis zu 30 % senken.
- Führen Sie mit Tools wie SQLIO Benchmarks für die E/A Ihres Systems durch. Zu diesem Tool ist ein Video-Tutorial verfügbar.

Die Aktivierung der standardmäßigen Komprimierung von Sicherungen reduziert E/A-Konflikte.

DatabaseName	FileName	FileID	FileType	NumReads	NumBytesRead	IOStallReadsMS	NumWrites	NumBytesWritten	IOStallWritesMS	TotalIOStall...	LastReset	SizeOnDiskMB	DiskLocation
AWTarget2012	AdventureWorks2012_Data	1	Data	91	5709824	10626	8	73728	2	10628	31/01/201...	205.000000	C:
NewTest	NewTest	1	Data	36	2113536	9963	1	8192	19	9982	31/01/201...	5.000000	C:
AWSource2012	AdventureWorks2012_Data	1	Data	84	5251072	9731	2	16384	0	9731	31/01/201...	205.000000	C:
AW20121	AdventureWorks2012_Data	1	Data	74	4677632	9151	1	8192	0	9151	31/01/201...	205.000000	C:

Abbildung 3: E/A-Informationen auf Dateiebene von SQL Server

SQL Profiler kann nach wie vor für viele SQL Server-Umgebungen wichtige Echtzeiteinblicke in die Abläufe von Anwendungen bereitstellen.

Tipp Nr. 4: Cursor und andere schlechte T-SQL-Skripte können zu Anwendungsproblemen führen.

Ein Beispiel für schlechten Code

In einer früheren Position ist mir einmal der schlechteste Code begegnet, den ich in meiner Laufbahn je gesehen habe. Das System wurde bereits vor langer Zeit ersetzt. An dieser Stelle möchte ich jedoch kurz den Prozess aufführen, den die Funktion durchlaufen hat:

1. Parameterwert für Entfernen akzeptieren
2. Parameterausdruck für Entfernen akzeptieren
3. Länge des Ausdrucks ermitteln
4. Ausdruck in eine Variable laden
5. Für jedes Zeichen in der Variable eine Schleife durchführen und prüfen, ob es einem der zu entfernenden Werte entspricht. Wenn ja, Variable aktualisieren und Zeichen entfernen
6. Fortfahren mit dem nächsten Zeichen, bis der Ausdruck vollständig geprüft wurde

Sicherlich werden Sie jetzt ebenso entsetzt sein wie ich, als ich diesen Code zum ersten Mal gesehen habe. Offensichtlich hat hier jemand versucht, seine eigene „T-SQL-REPLACE"-Anweisung zu schreiben.

Das Schlimmste an diesem Beispiel aus der Praxis ist, dass diese Funktion im Rahmen einer Mailing-Routine dazu verwendet wurde, Adressen zu aktualisieren, und daher unzählige Male pro Tag abgefragt wurde.

Mit einer serverseitigen Profiler Ablaufverfolgung können Sie die Rechenlast Ihres Servers anzeigen und häufig ausgeführte Codes identifizieren. (Auf diese Weise sind wir auch auf diesen „Geniestreich" gestoßen.)

Abfrageoptimierung mit Abfrageplänen

Schlechte T-SQL-Skripts können auch als ineffiziente Abfragen erscheinen, die keine Indexe verwenden, was zumeist daran liegt, dass der Index fehlerhaft oder gar nicht vorhanden ist. Es ist von größter Wichtigkeit, dass Sie genau verstehen, wie Sie Abfragen mithilfe von Abfrageplänen optimieren können.

Eine ausführliche Erläuterung der Abfrageoptimierung mithilfe von Abfrageplänen kann im Rahmen dieses Dokuments nicht erfolgen. Die einfachste Methode, um diesen Prozess zu starten, besteht jedoch in der Umwandlung von „SCAN"-Vorgängen in „SEEKs". „SCANs" lesen jede Zeile in der Tabelle oder im Index und verbrauchen daher bei umfangreichen Tabellen zu viel wertvolle E/A-Leistung. Ein „SEEK"-Vorgang hingegen verwendet einen Index, um direkt zu der gewünschten Zeile zu gelangen. Voraussetzung dafür ist selbstverständlich, dass ein Index existiert. Wenn Sie „SCANs" in Ihrer Rechenlast finden, dann könnten fehlende Indexe der Grund dafür sein.

Es gibt eine Reihe informativer Bücher zu diesem Thema, zu denen auch die folgenden Titel gehören:

- „Professional SQL Server Execution Plan Tuning" von Grant Fritchey
- „Professional SQL Server 2012 Internals and Troubleshooting" von Christian Bolton, Rob Farley, Glenn Berry, Justin Langford, Gavin Payne, Amit Banerjee, Michael Anderson, James Boother und Steven Wort
- „T-SQL Fundamentals for Microsoft SQL Server 2012 and SQL Azure" von Itzik Ben-Gan

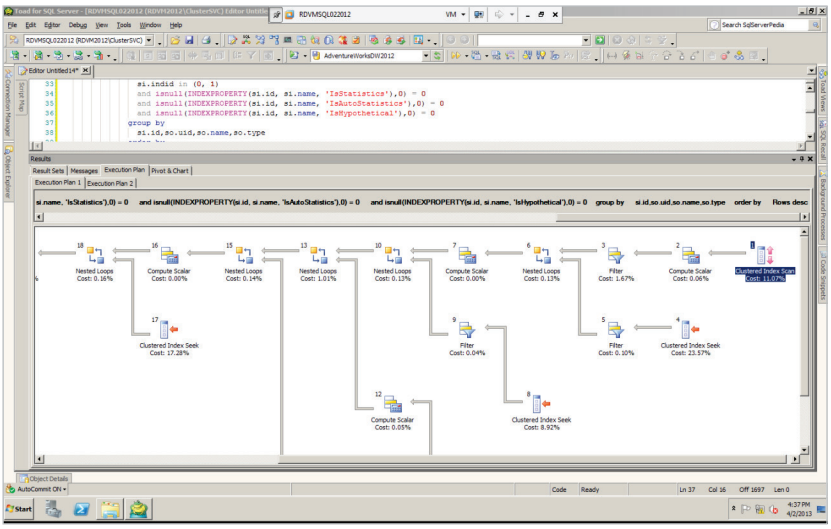

Abbildung 4: Beispiel für einen umfangreichen Abfrageplan

Tipp Nr. 3. Maximieren Sie die Wiederverwendung von Ausführungsplänen für besseres SQL Server Caching.

Wesentliche Gründe für die Wiederverwendung von Abfrageplänen

Vor der Ausführung einer SQL-Anweisung erstellt SQL Server zunächst einen Abfrageplan. Dieser Plan legt die Methode fest, nach der SQL Server den logischen Befehl der Abfrage aufnimmt und ihn als physische Maßnahme in Bezug auf die Daten implementiert.

Die Erstellung eines Abfrageplans erfordert unter Umständen erhebliche CPU-Kapazität. Folglich kann SQL Server effizienter arbeiten, wenn Abfragepläne wiederverwendet werden und nicht bei jeder Ausführung einer SQL-Anweisung ein neuer Plan erstellt werden muss.

Bewertung der aktuellen Wiederverwendungsrate von Ausführungsplänen in Ihrer Umgebung

Es gibt einige Leistungsindikatoren im Leistungsobjekt „SQL-Statistik", an denen Sie ablesen können, ob der Prozentsatz an wiederverwendeten Abfrageplänen ausreichend ist. Die unten aufgeführte Formel gibt das Verhältnis von gesendeten Paketen zu Kompilierungen an:

Der Wert sollte so niedrig wie möglich sein. Ein Verhältnis von 1:1 bedeutet, dass jedes gesendete Paket kompiliert wird und somit keine Pläne wiederverwendet werden.

(Paketabfragen/s – SQL-Kompilierungen/s)/Paketabfragen/s

SAN-Speichersysteme können so konfiguriert werden, dass sie schnelle SQL Server Leistung garantieren – oftmals sind sie es jedoch nicht.

Die Wiederverwendung von Ausführungsplänen verbessern

Oft ist es nicht einfach, zu ermitteln, welche Rechenlast nun genau für die unzureichende Wiederverwendung von Ausführungsplänen verantwortlich ist, da das Problem in der Regel vom Code der Client-Anwendung verursacht wird, die die Abfragen sendet. Unter Umständen müssen Sie den Code der Client-Anwendung untersuchen, von der die Abfragen gesendet werden.

Um Code zu finden, der in eine Client-Anwendung integriert ist, müssen Sie entweder erweiterte Ereignisse oder Profiler verwenden. Wenn Sie das Ereignis „SQL:StmtRecompile" zu einer Ablaufverfolgung hinzufügen, können Sie sehen, wann ein Rekompilierungsereignis abläuft. (Es gibt auch ein Ereignis mit dem Namen „SP:Recompile". Es wurde aus Gründen der Rückwärtskompatibilität eingeführt, da der Kompilierungsvorgang in SQL Server 2005 von der Prozedurebene auf die Anweisungsebene verlegt wurde.)

Ein häufiges Problem liegt darin, dass der Code keine vorbereiteten parametrisierten Anwendungen nutzt. Parametrisierte Abfragen optimieren nicht nur die Wiederverwendung von Ausführungsplänen und reduzieren den Kompilierungsaufwand, sondern verringern auch das Risiko von Angriffen durch Einschleusung von SQL-Befehlen, das in Zusammenhang mit der Übergabe von Parametern über Zeichenfolgenverkettung auftritt. Abbildung 5 zeigt zwei Codebeispiele. Auch wenn diese Beispiele nicht ganz realistisch sind, so veranschaulichen sie doch den Unterschied zwischen der Erstellung einer Anweisung durch Zeichenfolgenverkettung und der Erstellung unter Verwendung vorbereiteter Anweisungen mit Parametern.

Mit E/A-Informationen auf Dateiebene von SQL Server können Sie genau erkennen, welche Dateien viel E/A-Bandbreite verbrauchen.

```
public void ExecuteSomeSQL(int aParam) {
    //cmd is a SqlCommand created somewhere else

    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select foo1, foo2, foo3 from bar where foo1=" + aParam.ToString();

    SqlDataReader dr = cmd.ExecuteReader();
    try {
        while (dr.Read()) {

        }
    } finally {
        dr.Close();
    }

}
```

Schlecht

```
public void ExecuteSomeSQL(int aParam) {
    //cmd is a SqlCommand created somewhere else

    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select foo1, foo2, foo3 from bar where foo1 = @foo1";

    cmd.Parameters["@foo1"].Value = aParam;

    SqlDataReader dr = cmd.ExecuteReader();
    try {
        while (dr.Read()) {

        }
    } finally {
        dr.Close();
    }

}
```

Gut

Abbildung 5: Vergleich von Code, der Anweisungen durch Zeichenfolgenverkettung erstellt, und Code, der vorbereitete Anweisungen mit Parametern verwendet

SQL Server kann den Plan aus dem „schlechten" Beispiel in Abbildung 5 nicht wiederverwenden. Hätte es sich bei einem Parameter um einen Zeichenfolgentyp gehandelt, so könnte mit dieser Funktion ein Angriff durch Einschleusung von SQL-Befehlen vorbereitet werden. Bei dem „guten" Beispiel ist kein Angriff durch Einschleusung von SQL-Befehlen möglich, da ein Parameter verwendet wurde. SQL Server kann den Ausführungsplan wiederverwenden.

Die fehlende Konfigurationseinstellung aus Tipp Nr. 8

Wie Sie sich vielleicht erinnern werden, haben wir Ihnen in Tipp Nr. 8 (in dem es um Konfigurationsänderungen ging) noch einen wichtigen Ratschlag vorenthalten.
In SQL Server 2008 wurde eine Konfigurationseinstellung eingeführt, die „Für Ad-hoc-Arbeitsauslastungen optimieren" lautet. Wenn Sie diese Einstellung aktivieren, so wird SQL Server anstelle eines vollständigen Plans lediglich einen Stub-Plan im Plan-Cache speichern. Diese Option bietet sich insbesondere in Umgebungen an, die dynamisch erstellten T-SQL-Code oder LINQ verwenden, was unter Umständen die Wiederverwendung von Code unterbindet.

Der dem Plan-Cache zugeordnete Arbeitsspeicher befindet sich im Pufferpool. Aus diesem Grund reduziert ein aufgeblähter Plan-Cache die Anzahl an Datenseiten, die im Puffer-Cache gespeichert werden können. So sind mehr Roundtrips erforderlich, um Daten aus dem E/A-Subsystem abzurufen, was sich als ausgesprochen ressourcenintensiv erweisen kann.

Tipp Nr. 2. Lernen Sie, den SQL Server Puffer-Cache richtig zu lesen und somit Cache-Thrashing zu minimieren.

Warum der Puffer-Cache so wichtig ist
Wie oben bereits angedeutet, handelt es sich beim Puffer-Cache um einen umfangreichen Bereich im Arbeitsspeicher, der von SQL Server verwendet wird, um weniger physische E/A-Operationen ausführen zu müssen. Bei der Ausführung einer SQL Server Abfrage werden nie Daten direkt von der Festplatte gelesen. Die Datenbankseiten werden vielmehr aus dem Puffer-Cache ausgelesen. Kann die gesuchte Seite nicht im Puffer-Cache gefunden werden, wird eine physische E/A-Anfrage in die Warteschlange gestellt. Die Anfrage wartet und die Seite wird von der Festplatte abgerufen.

Wenn Daten auf einer Seite eines „DELETE"- oder „UPDATE"-Vorgangs geändert werden, so werden diese Änderungen auch auf den Seiten im Puffer-Cache vorgenommen. Die Änderungen werden später an die Festplatte weitergeleitet. Durch diesen Mechanismus kann SQL Server die physische E/A-Leistung auf verschiedene Weisen optimieren:

- Mehrere Seiten können in einem E/A-Vorgang gelesen und geschrieben werden.
- Read-Ahead kann implementiert werden. Bei bestimmten Arten von Vorgängen erkennt SQL Server unter Umständen, dass es hilfreich sein könnte, sequenzielle Seiten zu lesen. Dahinter steht die Annahme, dass nach dem Lesen der angeforderten Seite vielleicht auch die folgende Seite gelesen werden soll.

Hinweis: Indexfragmentierung hindert SQL Server daran, eine Read-Ahead-Optimierung vorzunehmen.

Evaluierung des Puffer-Cache-Zustands

Es gibt zwei Indikatoren, die Aufschluss über den Zustand des Puffer-Caches geben:

- **MSSQL$Instance:Puffer-Manager/Puffer-Cache-Trefferquote:** Dies ist das Verhältnis von im Cache gefundenen Seiten zu nicht im Cache gefundenen Seiten (d. h. Seiten, die von der Festplatte gelesen werden müssen). Idealerweise sollte dieser Wert so hoch wie möglich sein. Trotz einer hohen Trefferquote kann es zu Cache-Thrashing kommen.
- **MSSQL$Instance:Puffer-Manager\ Lebenserwartung von Seiten:** Dieser Wert gibt an, wie lange SQL Server Seiten im Puffer-Cache zwischenspeichert, bevor sie entfernt werden. Laut Microsoft darf die Lebenserwartung einer Seite länger als fünf Minuten sein. Liegt die Lebenserwartung unter diesem Wert, so kann das ein Indiz für ungenügende Arbeitsspeicherkapazität oder für Cache-Thrashing sein.

Ich möchte diesen Abschnitt mit einem Vergleich abschließen: Oft wird behauptet, dass sich mit der Erfindung von Antiblockiersystemen und anderen unterstützenden Technologien der nötige Bremsweg kürzer ansetzen ließe und demzufolge die Geschwindigkeitsbegrenzungen parallel zur technologischen Weiterentwicklung heraufgesetzt werden könnten. Die Warnschwelle von 300 Sekunden (fünf Minuten) für die Lebenserwartung von Seiten hat zu einer ähnlichen Debatte in der SQL Server Community geführt: Einige halten diesen Wert für eine unantastbare Regel, während andere glauben, dass der Anstieg an Arbeitsspeicherkapazität in modernen Servern eher einen vierstelligen Wert als einen dreistelligen erlaubt. Aus der Perspektive dieses Whitepapers sind diese unterschiedlichen Meinungen nur ein weiterer Beweis dafür, wie wichtig Baselines sind und warum Sie genau verstehen müssen, wo die Warnschwellen der einzelnen Leistungsindikatoren in Ihrer Umgebung angesiedelt werden sollten.

Cache-Thrashing

Während eines umfangreichen Tabellen- oder Indexscans muss jede Seite, die überprüft wird, den Puffer-Cache durchlaufen. Das bedeutet, dass potenziell nützliche Seiten aus dem Cache entfernt werden, um Platz für andere Seiten zu schaffen, die wahrscheinlich nur ein einziges Mal gelesen werden. Dies führt zu einer hohen E/A-Last, da die entfernten Seiten erneut von der Festplatte gelesen werden müssen. Ein derartiges Cache-Thrashing ist in der Regel ein Anzeichen dafür, dass umfangreiche Tabellen oder Indexe gescannt werden.

Um herauszufinden, welche Tabellen und Indexe den meisten Platz im Puffer-Cache einnehmen, können Sie die dynamische Verwaltungssicht (DMV, Dynamic Management View) „sys.dm_os_buffer_descriptors" nutzen. Sie ist ab SQL Server 2005 verfügbar. Das folgende Beispiel einer Abfrage verdeutlicht, wie Sie eine Liste der Tabellen oder Indexe

Wenn sich die Datenbankdateien am korrekten Speicherort befinden und alle Objekt-Hotspots identifiziert und auf verschiedenen Volumes abgelegt wurden, müssen Sie sich eingehend Ihrer Hardware zuwenden.

```
SELECT o.name, i.name, bd.*
  FROM sys.dm_os_buffer_descriptors bd
    INNER JOIN sys.allocation_units a
      ON bd.allocation_unit_id = a.allocation_unit_id
    INNER JOIN
    sys.partitions p
      ON (a.container_id = p.hobt_id AND a.type IN (1, 3))
        OR (a.container_id = p.partition_id AND a.type = 2)
    INNER JOIN sys.objects o ON p.object_id = o.object_id
    INNER JOIN sys.indexes i
      ON p.object_id = i.object_id AND p.index_id = i.index_id
```

Mit einer serverseitigen Profiler-Ablaufverfolgung können Sie die Rechenlast Ihres Servers anzeigen und häufig ausgeführte Codes identifizieren.

abrufen können, die in SQL Server viel Speicherplatz im Puffer-Cache einnehmen. Darüber hinaus können Sie über die Index-DMV herausfinden, für welche Tabellen oder Indexe viele physische E/A-Operationen anfallen.

Tipp Nr. 1. Verstehen Sie, wie Indexe verwendet werden, und ermitteln Sie schlechte Indexe.

SQL Server 2012 stellt ausgesprochen nützliche Daten zu Indexen bereit, die Sie mithilfe einiger ab SQL Server 2005 eingeführter DMOs abrufen können.

Verwendung des DMOs „sys.dm_db_index_operational_stats"

„sys.dm_db_index_operational_stats" enthält Informationen zu aktuellen Low-Level-Aktivitäten in den Bereichen E/A, Sperren, Latches und Zugriffsmethoden für jeden Index. Verwenden Sie diese dynamische Verwaltungsfunktion (DMF, Dynamic Management Function), um die folgenden Fragen zu beantworten:

- Gibt es einen Index, der sehr oft verwendet wird? Gibt es einen Index mit Konflikten? Die Spalten „row_lock_wait_in_ms" und „page_lock_wait_in_ms" geben Auskunft über eventuelle Wartezeiten für diesen Index.
- Gibt es einen Index, der ineffizient verwendet wird? Welche Indexe verursachen derzeit E/A-Engpässe? Die Spalte „page_io_latch_wait_ms" gibt Auskunft darüber, ob eventuell E/A-Wartezeiten anfallen, während Indexseiten in den Puffer-Cache verschoben werden. Dies ist ein zuverlässiger Hinweis auf ein Scan-Zugriffsmuster.
- Welche Arten von Zugriffsmustern werden verwendet? Die Spalte „range_scan_count" und die Spalte „singleton_lookup_count" dokumentieren, welche Zugriffsmuster für einen bestimmten Index verwendet werden.

Abbildung 6 veranschaulicht das Ergebnis einer Abfrage, in der die Indexe nach ihrer „page_io_latch"-Gesamtwartezeit aufgeführt sind. Dies ist sehr hilfreich, wenn Sie ermitteln wollen, welche Indexe für einen E/A-Engpass mitverantwortlich sind.

Verwendung des DMOs „sys.dm_db_index_usage_stats"

„sys.dm_db_index_usage_stats" enthält die Anzahl der verschiedenen Arten von Indexvorgängen und dokumentiert den Zeitpunkt, zu dem die einzelnen Vorgänge zuletzt ausgeführt wurden. Verwenden Sie diese dynamische Verwaltungsfunktion (DMV, Dynamic Management Function), um die folgenden Fragen zu beantworten:

- Wie verwenden die Benutzer die Indexe? Die Spalten „user_seeks", „user_scans" und „user_lookups" geben Auskunft über die Arten und die Bedeutung von Benutzervorgängen in Bezug auf die Indexe.
- Wie hoch sind die Kosten jedes Indexes? Die Spalte „user_updates" zeigt Ihnen, wie hoch der Wartungsaufwand eines Indexes ist.
- Wann wurde ein Index zuletzt verwendet? Die Spalten „last_*" dokumentieren, wann zuletzt ein Vorgang für einen Index ausgeführt wurde.

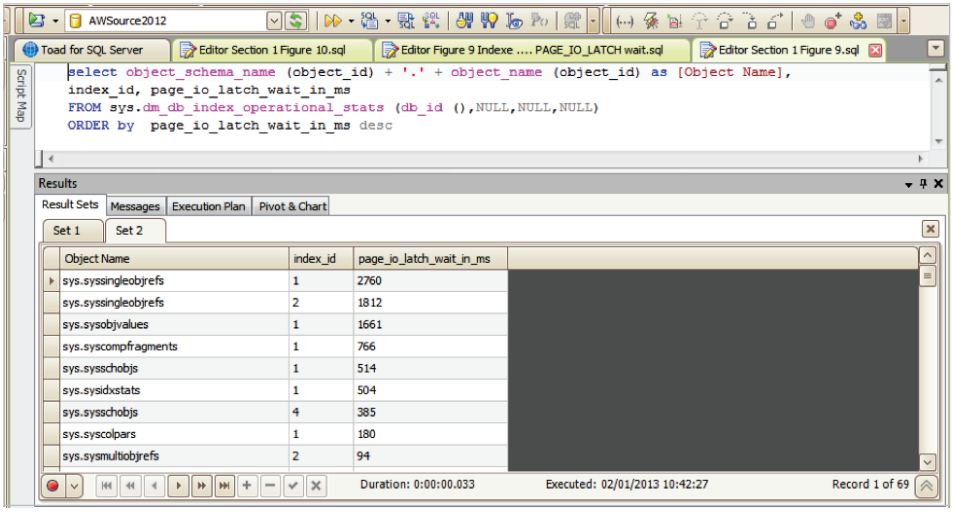

Abbildung 6: Indexe aufgeführt nach der gesamten „page_io_latch"-Wartezeit

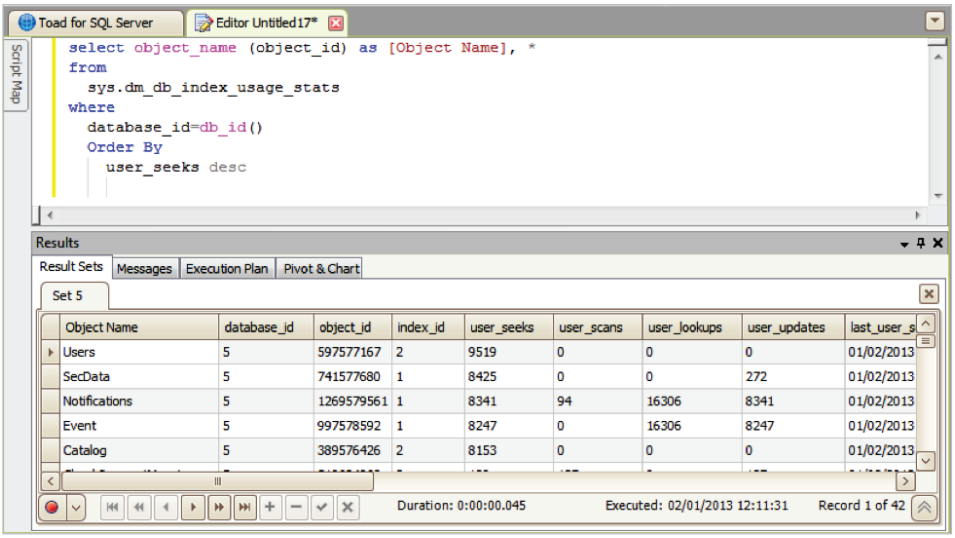

Abbildung 7: Indexe aufgeführt nach der Gesamtzahl an „user_seeks“

Abbildung 7 veranschaulicht das Ergebnis einer Abfrage, in dem die Indexe nach der Gesamtzahl an „user_seeks“ aufgeführt sind. Wenn Sie stattdessen Indexe identifizieren wollen, für die viele Scans ausgeführt wurden, können Sie die Liste nach der Spalte „user_scans“ filtern. Nun, da Sie den Namen des problematischen Indexes kennen, möchten Sie natürlich auch herausfinden, welche SQL-Anweisungen diesen Index nutzen. In SQL Server 2005 und neueren Versionen ist dies möglich.

Selbstverständlich gibt es noch viele weitere Aspekte in Bezug auf Indexes, die Beachtung verdienen, darunter solche wie Designstrategie, Konsolidierung und Wartung. Wenn Sie sich detaillierter über diesen zentralen Bereich der Leistungsoptimierung unter SQL Server informieren möchten, besuchen Sie SQLServerPedia. Oder schauen Sie sich einige der Webcasts oder Whitepaper an, die Dell zu diesem Thema veröffentlicht hat.

Fazit

Natürlich gibt es weit mehr als nur zehn Dinge, die Sie über die SQL Server Leistung wissen sollten. Dieses Whitepaper bietet Ihnen jedoch einen guten Ausgangspunkt und einige praktische Tipps zur Leistungsoptimierung, die Sie auf Ihre SQL Server Umgebung anwenden können. Denken Sie als kurze Wiederholung an diese zehn Ratschläge, wenn Sie versuchen, die Leistung Ihrer SQL Server Bereitstellung zu optimieren:

10. Benchmarking gibt Ihnen gute Anhaltspunkte darüber, welches Verhalten als normal betrachtet werden kann. So lässt sich das Verhalten von Rechenlasten einfacher vergleichen und auf Abweichungen von der Norm untersuchen.
9. Leistungsindikatoren liefen Ihnen schnell nützliche Informationen über aktuelle Abläufe.
8. Die Änderung von Servereinstellungen kann für eine stabilere Umgebung sorgen.
7. DMOs helfen Ihnen bei der schnellen Identifizierung von Leistungsengpässen.
6. Lernen Sie, SQL Profiler, Ablaufverfolgungen und erweiterte Ereignisse richtig einzusetzen.
5. SANs sind nicht nur Systeme zur E/A-Ausführung, sondern bieten auch Optimierungspotenzial für SQL Server.
4. Cursor und andere schlechte T-SQL-Skripte können zu Anwendungsproblemen führen.
3. Maximieren Sie die Wiederverwendung von Ausführungsplänen für besseres SQL Server Caching.
2. Lernen Sie, den SQL Server Puffer-Cache richtig zu lesen und somit Cache-Thrashing zu minimieren.

Und der allerwichtigste Tipp für die Optimierung der SQL Server Leistung:

1. Werden Sie ein Indexspezialist und lernen Sie, wie Indexe verwendet werden und woran Sie einen schlechten Index erkennen.

Beträgt die Lebenserwartung einer Seite weniger als fünf Minuten, kann das ein Indiz für Arbeitsspeicherauslastung (also zu wenig Arbeitsspeicher) oder für das sogenannte Seitenflattern (Thrashing) sein, das entsteht, wenn in kurzer Zeit mehr Daten verarbeitet werden, als Speicherkapazität zur Verfügung steht.

Parametrisierte Abfragen optimieren nicht nur die Wiederverwendung von Ausführungsplänen und reduzieren den Kompilierungsaufwand, sondern verringern auch das Risiko von Angriffen durch Einschleusung von SQL-Befehlen, das in Zusammenhang mit der Übergabe von Parametern über Zeichenfolgenverkettung auftritt.

Maßnahmenplan

Sicherlich möchten Sie das in diesem Whitepaper vermittelte Wissen gleich in die Praxis umsetzen. Die folgende Tabelle nennt Ihnen Aufgaben, die Sie sofort nach der Lektüre in Angriff nehmen können, um Ihre SQL Server Umgebung zu optimieren:

Aufgabe	Teilaufgaben	Zieldatum
Holen Sie die erforderliche Genehmigung ein, um mit dem Projekt zu beginnen.	Sprechen Sie mit Ihrem direkten Vorgesetzten und betonen Sie, dass Sie durch die Umsetzung des Projekts proaktiv anstatt reaktiv sein können.	
Legen Sie Leistungszeile fest.	Sprechen Sie mit Stakeholdern des Unternehmens, um akzeptable Leistungswerte festzulegen.	
Bestimmen Sie eine Baseline für die Systemleistung.	Sammeln Sie relevante Daten und speichern Sie sie in einem speziell dafür eingerichteten Repository oder in einem Drittanbieter-Repository.	
Identifizieren Sie die wichtigsten Leistungsindikatoren und konfigurieren Sie die Ablaufverfolgung und/oder erweiterte Ereignisse.	Laden Sie sich das Dell Poster mit den Perfmon Leistungsindikatoren herunter.	
Überprüfen Sie die Servereinstellungen.	Beachten Sie dabei insbesondere den Arbeitsspeicher und die Einstellung „Für Ad-hoc-Arbeitsauslastungen optimieren“.	
Überprüfen Sie das E/A-Subsystem.	Sprechen Sie bei Bedarf mit Ihren SAN-Administratoren und ziehen Sie E/A-Lasttests mit Tools wie SQLIO in Betracht. Oder ermitteln Sie die Rate, mit der Sie intensive Lese- und Schreibvorgänge ausführen können, z. B. während einer Sicherung.	
Identifizieren Sie Abfragen mit unzureichender Leistung.	Analysieren Sie die Daten aus Ablaufverfolgungen, aus Sitzungen zur Erfassung von erweiterten Ereignissen und aus dem Plan-Cache.	
Verbessern Sie Code mit schlechter Leistung.	Informieren Sie sich im umfassenden Blogservice von SQLServerPedia über die neuesten Best Practices.	
Indexwartung	Stellen Sie stets einen optimalen Zustand Ihrer Indexe sicher.	

Weitere Informationen

Über Dell

Dell Inc. (NASDAQ: DELL) entwickelt innovative Technologien, Business-Lösungen und Services, maßgeschneidert für die spezifischen Anforderungen unserer Kunden. Unsere Produkte garantieren starke Leistung und höchste Zuverlässigkeit. Weitere Informationen finden Sie unter www.dell.com.

Sollten Sie Fragen hinsichtlich der potenziellen Nutzung des Materials haben, wenden Sie sich bitte an:

Dell Software

5 Polaris Way
Aliso Viejo, CA 92656
www.dell.com
Informationen zu unseren regionalen oder internationalen Büros finden Sie auf unserer Webseite.