White-Paper: SQL Anywhere® Performance-Tuning

Änderungsstand: 2015-01-26
Autor: Markus KARG (karg@quipsy.de)

Zusammenfassung

Dieses White-Paper erläutert Maßnahmen zur Erhöhung der Geschwindigkeit bei Nutzung des relationalen Datenbank-Management-Systems (RDBMS) "Sybase SQL Anywhere®" der SAP AG.

Gilt für

- QUIPSY® CAQ, PMV, EMP, FMEA, RB 4.35
- QUIPSY® DMS und Audit bei Nutzung des SQL Anywhere®-Backends
- Sybase SQL Anywhere® 11.0.1

Einführung

Mit zunehmendem Umfang und Zugriffshäufigkeit auf die durch die QUIPSY®-Software verarbeiteten Informationen steigt der Bedarf nach administrativen Maßnahmen zur Verbesserung der Zugriffsgeschwindigkeit.

Im Folgenden sind beispielhafte Optimierungsmaßnahmen beschrieben, welche geeignet und von der QUIPSY QUALITY GmbH & Co. KG empfohlen sind, wenn SQL Anywhere® 11.0.1 zum Einsatz kommt.

Über dieses Whitepaper hinausgehend sind im Handbuch dieses RDBMS weitere Informationen zur Erhöhung der Zugriffsgeschwindigkeit beschrieben (siehe http://dcx.sybase.com/index.html#1100/en/dbadmin_en11/dbadmin_en11.html).

Dieses Whitepaper verbindet fachspezifisches Allgemeinwissen über Datenbankadministration mit den speziellen Belangen der Produkte SQL Anywhere® und QUIPSY® CAQ. Es ist grundsätzlich jedoch auch geeignet, um mit anderen Produkten angewendet zu werden.

Hintergrund

Die QUIPSY®-Software nutzt eine Datenbank, welche auf dem Sybase SQL Anywhere® Server in typischerweise zwei, gelegentlich auch mehr, Dateien vorliegt. Diese Dateien werden nicht direkt von der QUIPSY®-Software angesprochen, sondern stehen unter alleiniger Kontrolle des Datenbank-Serverdienstes (`dbsrv11.exe`), mit welcher die QUIPSY®-Software kommuniziert.

Theorie

Der Zugriff auf Festplatten (Hard Disk Drives, HDDs) ist auch auf moderner Hardware immer noch etwa 1.000 mal langsamer als der Zugriff auf Arbeitspeicher (Random Access Memory, RAM). Sogenannte Solid-State Discs (SSD) sind durch den Verzicht auf mechanische Bauteile zwar erheblich schneller als mechanische Festplatten, trotzdem immer noch um mehrere Größenordnungen langsamer als RAM.

Optimal wäre somit, Daten ausschließlich im RAM zu halten (In-Memory). Leider führt dies bei Stromausfällen zum Datenverlust, was gerade bei CAQ-Anwendungen untragbar ist. Daher führt kein Weg daran vorbei, alle Daten sofort und dauerhaft auf nicht-flüchtige Medien zu schreiben. Da SSDs

erheblich schneller sind als HDDs, wäre die ausschließliche Nutzung von SSDs optimal. Leider sind diese um Größenordnungen teurer als HDDs.

Innerhalb der Gruppe der Festplatten sind SAS-Festplatten (Serial Attached SCSI) zumeist schneller als SATA-Festplatten (Serial ATA), was hauptsächlich ein Nebeneffekt der höheren Umdrehungsgeschwindigkeit ist: SAS-Festplatten drehen sich mit etwa 15.000 $^{1}/_{min}$, SATA-Festplatten typischerweise nur mit maximal der Hälfte. Leider sind SAS-Festplatten teurer als SATA-Festplatten, und nur mit kleinerer Kapazität verfügbar. Folglich entsteht beim Zugriff (vor allem beim Schreibenden Zugriff) auf SATA-Festplatten eine höhere Wartezeit als bei SAS-Platten.

Bei SSDs ist bezüglich der Geschwindigkeit üblicherweise kein eklatanter Unterschied zwischen SAS- und SATA-Varianten bekannt, sehr wohl jedoch zwischen SAS/SATA-konnektierten und PCIe-konnektierten. PCIe-Varianten dabei sind um Zehnerpotenzen schneller als SAS/SATA-Varianten, aber auch ungleich teurer und von gängigen Betriebssystemen schlechter unterstützt.

Technisch bedingt sind Schreibzugriffe, egal ob auf SSD oder HDD und unabhängig von der Anschlussart, *immer* erheblich langsamer als Lesezugriffe. Dies ergibt sich alleine bereits aus der Tatsache, dass zum Schreiben grundsätzlich stets zunächst gelesen werden muss. Zudem erfolgen Schreibzugriffe oftmals "schubweise", d. h. nicht sofort sondern nach einer absichtlichen oder technisch bedingten Wartezeit.

Schreib- und Lesezugriffe stören sich gegenseitig. Dies ist gerade bei HDDs offensichtlich, da typischerweise nur *ein* Schreib-Lesekopf pro Medium vorhanden ist, somit also zu jedem Zeitpunkt nur auf *einen* Sektor der Platte positioniert sein kann - entweder auf den zu lesenden, oder auf den zu schreibenden. Gleichzeitiges Lesen und Schreiben führt im Extremfall dazu, dass mehr Zeit für das Positionieren verstreicht, als für den eigentlichen Medienzugriff. Auch SSDs stellen hier *keine* Ausnahme dar, wenngleich aufgrund der wegfallenden mechanischen Bewegung das Problem weniger Stark zutage tritt.

Aus algorithmischer Sicht geht das Anhängen an bestehende Dateien sehr schnell, das Ändern in der Mitte von Dateien benötigt dagegen relativ viel Zeit. Umgekehrt ist das Auffinden einer bestimmten zu lesenden Information in einer *strukturierten* Datei ("Archiv") sehr schnell (Direktzugriff an die "richtige" Position), während das Suchen der gleichen Information in einer *chronologisch* fortgeschriebenen Datei ("Journal") sehr langsam ist.

SQL Anywhere® kombiniert, wie die meisten relationalen Datenbank-Management-Systeme, ein Journal mit einem Archiv. Hierbei kommt beim Schreiben das WAL-Verfahren (Write Ahead Logging) zur Anwendung. Zunächst werden zu schreibende Informationen an das Ende des Journals angehängt. Zu einem *wesentlich* späteren Zeitpunkt wird (vom Anwender unbemerkt, da im Hintergrund) dann das Archiv geändert. Grundsätzlich liest SQL Anywhere® zur Betriebszeit ausschließlich per Direktzugriff aus dem Archiv und puffert so viel Information wie nur möglich im Hauptspeicher (RAM). Somit wird das jeweils optimale Zugriffsverfahren auf das jeweilige Medium angewendet.

Die Trennung in Journal und Archiv hat primär Sicherheitsgründe. Siehe hierzu das White-Paper "Datensichermit mit SQL Anywhere®". Sie bietet jedoch zusätzlich auch einen Ansatzpunkt für die getrennte Geschwindigkeits-Optimierung der beiden Dateien.

Durch Lagerung von Journal und Archiv auf getrennten Medien ist es möglich, Schreib- und Lesezugriffe zu entkoppeln (kein gegenseitiges Warten), sowie die Schreiben bzw. Lesen optimierte Medien zu verwenden.

Üblicherweise wird ein *Journal* zur Reduktion von Wartzeiten auf einem für hohe Schreibgeschwindigkeit optimierten Volume abgelegt. *Die Verfügbarkeit sollte dabei jedoch nicht außer Acht gelassen werden (d. h. kein RAID 0 sondern RAID 1).*

Ein *Archiv* wird zur Reduktion von Wartezeiten üblicherweise auf einem für hohe Lesegeschwindigkeit optimieren Volume abgelegt. *Aus Gründen der Datensicherheit sollte jedoch nicht RAID 1 sondern besser RAID 6 zum Einsatz kommen.*

Bei allen Performance-Optimierungen sollte immer bedacht werden, dass im CAQ-Umfeld die Datensicherheit (möglichst geringer Datenverlust) die höchste Priorität haben muss. In vielen Betrieben ist ebenso die Verfügbarkeit (d. h. möglichst kurze Ausfallzeiten) als auch die Wirtschaftlichkeit höher zu bewerten als die Performance. Daher schließt sich RAID 0 typischerweise ebenso aus wie Einzelplatten (Datenverlust, Verfügbarkeit), reine SSD-Lösungen ebenso wie "Unmengen" an Festplatten oder die pauschale Nutzung von RAID 60 (Kosten, Performance).

Eine sinnvolle Lösung besteht darin, für Journal und Archiv physische zu entkoppeln, die jeweils richtigen RAID-Level zu finden und möglichst viel RAM einzusetzen.

Wie die meisten RDBMS prüft SQL Anywhere® vor einem Datenzugriff, welche der möglichen Vorgehensweisen die optimale ist (z. B. "Alle Daten sequenziell lesen" statt "Die benötigten Daten über einen Index ermitteln, nur diese lesen."). Dieses Verfahren ist kostenbasiert. Daher benötigt SQL Anywhere® Informationen darüber, wie hoch die Kosten (allen voran für Plattenzugriffe als höchste Kosten) tatsächlich auf dem jeweiligen Gerät sind.

Da diese Kosten maßgeblich von der lokalen Hardware abhängig sind, der eingestellte Standardwert jedoch in etwa "Laptop mit einer einzigen Festplatte" entspricht, ergibt sich ein Verbesserungspotential durch Messung der *tatsächlichen* Hardware-Geschwindigkeit.

Abgesehen von diesen Hauptfaktoren gibt es noch weitere Aspekte zu beachten: Übertragungsgeschwindigkeit (LAN), Fragmentierung von Dateien, etc. Diese wirken sich im Normalfall aber nur sehr viel geringer aus als Hauptspeicher und Medienzuschnitt und sind daher nachrangig.

Da typischerweise viele Anwender gleichzeitig auf den Datenbank-Server zugreifen, ist dessen relative Priorität gegenüber anderen Prozessen zu beachten: Ohne manuelle Justierung behandelt das Betriebssystem den Datenbankserver exakt gleich wie eine Software, die nur von einem einzigen Anwender ausgeführt wird (beispielsweise "NOTEPAD"). Das bedeutet, *alle Datenbankanwender zusammen* würden sich die gleiche Rechenzeit und den gleichen Anteil an Zugriffszeit auf die Festplatten und das Netzwerk teilen, *wie ein einziger Anwender*, der beispielsweise lediglich "NOTEPAD" öffnet. Da dies sicherlich in den wenigsten Fällen gewünscht ist, sollte *unbedingt* dem Datenbankserver (wie typischerweise allen von mehreren Anwendern gleichzeitig genutzten Prozessen) eine höhere Priorität eingeräumt werden. Dies bezieht sich sowohl auf die Priorität innerhalb des Betriebssystems, als auch die Priorität einer virtuellen Maschine gegenüber anderen virtuellen Maschinen. Bei der Balancierung von virtuellen Maschinen ist zudem zu beachten, dass die Leistungsfähigkeit von SQL Anywhere hauptsächlich vom Arbeitsspeicher und dem Festplatten-Zugriff abhängt, während die Rechenzeit (CPU) eine eher untergeordnete Rolle spielt, somit insbesondere also die Zuteilung von Festplatten-Zugriffsanteilen und Arbeitsspeicher-Priorität erhöht werden sollte.

Dateien

QUIPSY.log

Die Datei `QUIPSY.log` enthält das *Journal* der Datenbank und sollte sich auf einem für hohe Schreibgeschwindigkeit optimierten Volume befinden (z. B. **RAID 1 aus 2x SAS**). Die logische(*) Position der Datei ist in binärer Form in der Datei `QUIPSY.db` gespeichert und kann über das Werkzeug `dblog` verändert werden.

QUIPSY.db

Die Datei `QUIPSY.db` enthält das *Archiv* der Datenbank und sollte sich auf einem für hohe Lesegeschwindigkeit optimierten Medium befinden (z. B. **RAID 6 aus 6x SATA**). Die logische(*) Position der Datei wird dem Serverdienst über einen Kommandozeilen-Parameter mitgeteilt und kann über das Werkzeug `dbsvc` verändert werden.

QUIPSY.mlg (opt.)

Die Datei `QUIPSY.mlg`, sofern genutzt, enthält eine *Kopie* des Journals, welche zeitgleich mit dem

Original-Journal geschrieben wird. Die logische(*) Position der Datei ist in binärer Form in der Datei `QUIPSY.db` gespeichert und kann über das Werkzeug `dblog` verändert werden. Das Journal sollte sich ebenso auf einem für hohe Schreibgeschwindigkeit optimierten Volume befinden (z. B. **RAID 1 aus 2x SAS**).

Temporäre Dateien

Temporäre Dateien werden vom Datenbankserver verwendet, wenn für bestimmte Operationen wie Sortieren oder Gruppieren nicht genügend Arbeitsspeicher frei ist. Sie sollten auf einem Medium liegen, welches höchstmögliche Schreibgeschwindigkeit aufweist (z. B. **RAID 1 aus 2x SSD**). Die Position von temporären Dateien wird über die Umgebungsvariable `SATMP` festgelegt. Sofern diese nicht vorhanden ist, wird das allgemeine Temporär-Verzeichnis des Betriebssystems genutzt (Umgebungsvariable `TEMP`).

(*)Tip: Nicht die logische Position der Dateien ändern (`dblog` / `dbsvc`), sondern statt dessen die physische. Hierzu die Dateien zunächst physisch verschieben (`MOVE`) und danach über `MKLINK` auf Windows® bzw. `link` auf Linux® einen neuen Eintrag im ursprünglichen Directory erstellen. Dies erleichtert die Übersichtlichkeit über die logische Zusammengehörigkeit der Dateien erheblich, da sie logisch gesehen weiterhin im gleichen Ordner liegen, physisch jedoch auf getrennten Volumens).

Maßnahmen

Zur Erzielung möglichst optimaler Geschwindigkeit sollten alle folgenden Maßnahmen angewendet werden. Aus Gründen der Wirtschaftlichkeit kann es jedoch angebracht sein, einige Maßnahmen nicht anzuwenden, oder statt dessen gleichwertige Ersatzmaßnahmen anzuwenden.

Einige der genannten Maßnahmen können nur von QUIPSY®-Servicepersonal durchgeführt werden.

Generelle Maßnahmen zur Erhöhung der Geschwindigkeit

Generell ist für ein performantes Arbeiten mit QUIPSY® **obligatorisch**, die folgenden Maßnahmen pauschal anzuwenden.

- Entkopplung von Lesen und Schreiben

  Grundsätzlich sollten Journal und Archiv niemals auf dem selben, physikalischen Medium liegen. Da QUIPSY® CAQ aufgrund der Sammlung von Messdaten sehr viele kleine Schreibzugriffe aufweist, gleichzeitig wenige dafür sehr umfangreiche Lesezugriffe für Auswertungen ausführt, ist es unbedingt notwendig, diese Zugriffe voneinander zu isolieren. Der wirtschaftlichste und gleichzeitig aus Gründen der Datensicherheit optimale Weg ist die Nutzung von **getrennten *physikalischen* Medien**, mindestens also zwei echte Festplatten, besser noch zwei getrennte RAID-Sets, optimal sogar getrennte RAID-Controller.

  Zur weiteren Entkopplung können neben Journal und Archiv auch temporäre Dateien und ggf. Journal-Spiegel (`.mlg`) auf separaten Medien abgelegt werden, was je nach Umfeld und Nutzungsweise weitere Geschwindigkeits- und Sicherheitsvorteile bringen kann, ohne erhebliche Mehrkosten zu erzeugen.

- Viren-Scanner korrekt konfigurieren

  Keine der vom SQL Anywhere® genutzten Dateien (auch nicht dessen temporäre Dateien) können nach aktuellen Stand der Technik von Viren befallen werden. Ein Schutz durch einen Viren-Scanner ist daher nicht notwendig. Umgekehrt führt jedoch das Abscannen von Archiv, Journal, Journal-Spiegel und temporären Dateien zu möglicherweise erheblichen Verzögerungen sowohl beim Lesen als auch beim Schreiben.

  Es ist ratsam, den Viren-Scanner so zu konfigurieren, dass diese Dateien niemals gescannt werden (weder zur Laufzeit noch während Routine-Scans).

- Betriebssystem-Priorität erhöhen

  Der Startparameter `-gb` setzt die beim Start von `dbsrv11.exe` gültige Prozesspriorität und sollte stets auf den Wert `high` gesetzt werden.

  Der Wert `maximum` ist zu vermeiden, da die Stabilität des Gesamtsystems gefährdet werden könnte. Ebenso ist der Wert `idle` zu vermeiden, da der Datenbankserver dadurch möglicherweise nicht mehr funktioniert.

  Startparameter wirken sich nur bei einem Neustart des Dienstes aus. Während der Laufzeit kann daher über die Prozesssteuerung des Betriebssystems (z. B. Windows® Task Manager oder `renice` und `ionice -p` auf Linux®) die Prozesspriorität angepasst werden. Hierbei sollte der Wert `Hoch` gesetzt werden, auf keinen Fall jedoch `Echtzeit`.

- Balance der virtuellen Maschinen justieren

  Wird der Datenbankserver in einer virtuellen Maschine betrieben, ist deren Priorität wie folgt zu justieren:
  - Festplatten, auf welche der Datenbank-Server zugreift (d. h. auf welchen Archiv, Journal oder temporäre Dateien liegen), sind mit der Priorität `Hoch` zu versehen.
  - Die Zuteilungspriorität für Arbeitsspeicher ist als `Hoch` festzulegen.

  Die Priorität aller anderen, virtuellen Maschinen, ist entsprechend deren Optimierungsziel *wahrheitsgemäß* zu reduzieren: Eine virtuelle Maschine, welche beispielsweise hauptsächlich Berechnungen durchführt (z. B. CAD), benötigt virtuelle Festplatten nur sporadisch. Daher ist dort die Festplatten-Nutzung als `Niedrig` zu priorisieren.

  Für alle weiteren Optimierungen im Bereich der Virtualisierung verweisen wir auf die entsprechenden Unterlagen des jeweiligen Herstellers.

- Messung der tatsächlichen Hardware-Geschwindigkeit

  Nach jedem "Umzug" der Datenbank-Dateien auf anderen Platten bzw. des Datenbank-Servers auf andere Hardware, Änderungen an der Hardware selbst, Justierung virtueller Hardware, oder Änderungen an der Verkabelung zwischen Datenbank-Server und Festplatten, sollte eine erneute Messung der Hardware-Geschwindigkeit vorgenommen werden.

  Die "Kalibrierung" genannte Geschwindigkeits-Justierung ist wie folgt auszuführen:

  - System muss in absoluter Ruhe sein

    Es darf keinerlei Last auf der gesamten Hardware-Kette, also auf dem Server / Host, Verbindung zum SAN, SAN, Platten, etc., anliegen!

  - Mittels `dbisql` sind diese Befehle ausführen

    - `ALTER DATABASE CALIBRATE SERVER`
    - `ALTER DATABASE CALIBRATE GROUP READ`
    - `ALTER DATABASE CALIBRATE PARALLEL READ`
    - `ALTER DATABASE CALIBRATE DBSPACE TEMPORARY`

Maßnahmen zur Erhöhung der Lesegeschwindigkeit

- **Top-Tip: Arbeitsspeicher vergrößern (RAM)**

  Optimale Leseperformance wird erreicht, wenn im laufenden Betrieb so viel Arbeitsspeicher (RAM) frei ist, dass das Archiv *zwei Mal* komplett hinein passt.

Dies erhöht die Performance stärker als alle anderen Maßnahmen zusammen.

Der RAM muss deshalb doppelt so groß sein als das Archiv, damit das Archiv komplett hineingeladen werden kann (Verzicht auf jeglichen Archiv-Medienzugriff), und gleichzeitig nötigenfalls alle Daten im Arbeitsspeicher gruppiert / sortiert werden können (Verzicht auf jeglichen Zugriff auf temporäre Dateien).

- Multi-Core-CPUs

  Der Einsatz von Multi-CPU-Cores (z. B. 2x SixCore) bringt mehr Leistung, als der Einsatz von mehreren CPUs mit wenigen Cores (z. B. 3x QuadCore).

  Die im Lieferumfang von SQL Anywhere® enthaltene Lizenz nutzt automatisch alle CPU-Cores von zwei CPU-Sockeln. Alle anderen Sockel werden nicht genutzt. Stehen keine Multi-Core-CPUs zur Verfügung kann ersatzweise ein zweiter Sockel genutzt werden, jedoch kein dritter.

  Zudem sind technisch bedingt Daten zwischen Cores im gleichen Sockel schneller zu transferieren als über mehrere Sockel hinweg (Busgeschwindigkeit, Cache-Lokalität).

  Optimale Leseperformance wird erreicht, wenn zwei Sockel mit den "größtmöglichen" CPUs belegt sind, d. h. sehr viele Cores aufweisen (z. B. 12) die mit sehr schnellen Takt (z. B. 3 GHz) laufen.

  ***Hinweis:** "HyperThreading" stellt diesbezüglich keinen echten CPU-Core dar, sondern erlaubt es lediglich, einige wenige Aufgaben parallel auf dem gleichen CPU-Core zu betreiben.*

- Striping (RAID 6, nicht RAID 0)

  Werden Festplatten per RAID 0, RAID 5 oder RAID 6 zusammengeschaltet, kann parallel von diesen gelesen werden. Dies erhöht somit die *Lesegeschwindigkeit* nahezu linear mit der Festplattenzahl.

  ***Hinweis:** Aufgrund der Datensicherheit sollte auf RAID 0 komplett **verzichtet** werden. Beim Einsatz moderner RAID-Controller ist RAID 6 typischerweise üblich, da es im Vergleich zu RAID 5 keine nennenswerten Performance-Nachteile mehr liefert, die Daten aber grundsätzlich besser schützt.*

  RAID 6 ist somit optimal für das Archiv (`*.db`).

  ***Hinweis:** RAID 5 und RAID 6 benötigen eine sehr lange Zeit für das **Schreiben**. Daher sollte das Journal (`*.log`) **auf keinen Fall** auf einem RAID 5 oder RAID 6 abgelegt werden. Ebenso wenig ist RAID 6 für Journal-Spiegel (`*.mlg`) oder temporäre Dateien geeignet.*

- Read Cache einschalten

  Die Nutzung von Read Cache beschleunigt das gleichzeitige Lesen, vor allem, wenn wenig RAM zur Verfügung steht und sehr langsame Festplatten genutzt werden. Der effektive Performance-Zugewinn ist hingegen bescheiden, wenn viel RAM zur Verfügung steht, oder SSDs zum Einsatz kommen.

  Da kein negativer Effekt von Read Cache bekannt ist, sollte dieser stets eingeschaltet bleiben.

- Richtige Festplattenwahl

  Für die Ablage des Archivs genügen SATA-Festplatten vollauf. Die Nutzung von SSDs und SAS-Festplatten ist für die Beschleunigung der Lesegeschwindigkeit nicht erforderlich (sehr wohl jedoch für die Schreibgeschwindigkeit). Gerade wenn genug RAM vorhanden und Read Caches eingeschaltet sind, spielt die Festplattentechnologie aus Lesesicht keine nennenswerte Rolle.

Maßnahmen zur Erhöhung der Schreibgeschwindigkeit

- **Top-Tip: Entkopplung von Log, Mirror Log und Temp**

  Damit sich konkurrierende Zugriffe nicht gegenseitig ausbremsen ist es für die Verbesserung der Schreibgeschwindigkeit **absolut erforderlich**, dass Journal und Archiv, optimaler Weise auch Journal-Spiegel und Temporäre Dateien, auf **getrennten, physischen Medien** abgelegt werden.

- Write Cache nicht verwenden

  Aus Gründen der Datensicherheit **muss** der Write Cache, welcher eigentlich optimale Schreibperformance bietet, **zwingend ausgeschaltet** werden, sofern keine funktionsfähige BBU (Battery Backup Unit) direkt auf dem RAID-Controller verbaut ist (eine UPS genügt nicht).

  Vor dem Einschalten von Write Caches zur Verbesserung der Schreibgeschwindigkeit wird ausdrücklich gewarnt, da die Datensicherheit in hohem Maße gefährdet ist!

  Sofern überhaupt ein Write Cache zum Einsatz kommt (weil eine funktionsfähige BBU vorhanden ist), darf **auf keinen Fall** die Strategie "Write Back" genutzt werden, sondern es muss zum Datenschutz **unbedingt** "Write Through" eingesetzt werden.

  Siehe hierzu auch das White-Paper *Datensicherheit mit SQL Anywhere®*.

- WARNUNG: Kein *reines* Striping (RAID 0) verwenden!

  Aus Gründen der Datenschutzes muss **reines** Striping (RAID 0), welches *eigentlich* höchstmögliche Schreibperformance bietet, **unbedingt vermieden** werden, da bei Defekt einer einzigen Platte sämtliche Daten verloren sind. Dies betrifft beim Journal somit alle Änderungen seit dem letzten Checkpoint, **möglicherweise also mehrere Minuten bis Stunden**.

  Zulässig und sinnvoll ist **reines Mirroring** (RAID 1), vor allem aber Mischformen (**RAID 10**).

  Die Nutzung von RAID 5 und 6 führt hingegen zu *massiven* Geschwindigkeitseinbußen beim Schreibzugriff und sollte daher *für das Journal* unbedingt vermieden werden!

  RAID 5 erreicht auf vielen RAID-Controllern oftmals erst ab etwa fünf Festplatten wieder die gleiche Schreibgeschwindigkeit wie eine einzelne Festplatte!

- Richtige Festplattenwahl

  Maximale Geschwindigkeit erreichen servertaugliche SSDs (Desktop-SSDs sind untauglich, da weder auf 24x7-Betrieb, noch die notwendigen Dauer-Schreibrate ausgelegt).

  Aus Gründen der *Wirtschaftlichkeit* genügen für das Journal jedoch auch SAS-Festplatten mit 15.000 UPM, optimaler Weise im Viererpack gebündelt zu RAID 10 oder aufgesplittet zu je 2x2 (RAID 1) für Journal und Temporären Dateien.

  Tip: Verfügt der Server über eine einzelne SSD als "Booster", können temporäre Dateien und Journal durchaus auf diesem abgelegt werden, was die Schreibleistung erheblich steigert. Davon wird jedoch abgeraten, da die Verfügbarkeit des Gesamtsystems typischerweise höherrangig zu bewerten ist gegenüber der Performance. Die Verfügbarkeit sinkt deutlich, da der "Booster" typischerweise in sich nicht redundant ist! ***Auf keinen Fall darf das Archiv auf dem "Booster" liegen, da dieses zwingend auf einem redundanten Laufwerk abgelegt sein muss!***

- Vorreservierter Festplattenplatz / Defragmentierung

  Zur Maximierung der Schreibleistung, und zur Optimierung der Leseleistung, sollte der verwendete Festplattenplatz von Journal und Archiv vorreserviert sein, da ansonsten stetig zunehmende Fragmentierung die Folge ist. Dies widerspricht der Idee von Thin Provisioning und

setzt die präventive Vergrößerung Journal und Archiv über SQL Anywhere®-Bordmittel voraus.

Alternativ kann auch die Festplatte regelmäßig defragmentiert werden, was jedoch nur bei abgeschaltetem SQL Anywhere®-Dienst möglich und sich mit virtualisierten Festplatten praktisch ausschließt.

Maßnahmen zur Erhöhung der Transportgeschwindigkeit

- Direkt angebundene Median statt SAN

  Direkt angebundene Medien (SAS, SATA) sind typischerweise schneller und agiler als solche, die über ein Speichernetzwerk konnektiert werden, wobei die zunehmende Geschwindigkeit moderner Speichernetzwerke (FC, 10 GbE iSCSI) diesen Nachteil jedoch zunehmend kompensiert.

- Freie Bahn in SANs / Dedizierte SANs

  Kommen SANs zum Einsatz, sollten zwischen HBA (Host -Bus-Adapter) und SAN-Gerät möglichst keine Geräte wie Layer 3 Switches, Router und Firewalls stehen, da jedes Gerät den passierenden Datenpaketen eine zusäzliche Verzögerung (interne Verarbeitung) auflädt.

  SANs sollten niemals über das LAN abgebildet werden, sondern ein exklusives, alleinstehendes Netzwerk bilden, um Störeinflüsse durch den LAN-Verkehr zu verhindern.

- Mindestens 4 Gb/s im SAN

  Beim Einsatz von SANs sollte dieses mindestens 4 Gb/s aufweisen (FC, iSCSI-Bonding, 10 GbE iSCSI).

- Mindestens 1 Gb/s im LAN

  Das zur Kommunikation zwischen Client und Server eingesetzt LAN sollte mindestens 1 Gb/s schnell sein.

  Bei der Auslegung ist zu beachten, dass neben QUIPSY® auch noch weitere Dienste über das gleiche LAN laufen, welche die geplante Bruttokapazität bereits annähernd auslasten.

- JumboFrames im SAN einschalten

  Gerade die QUIPSY® SPC-Auswertungen laden sehr viele Daten en bloc. Daher ist es ratsam, JumboFrames im SAN generell einzuschalten, sofern alle beteiligten Geräte dies erlauben.

- Kompression bei sehr langsamen LANs / im WAN

  LANs mit einer Geschwindigkeit von 10 Mb/s und weniger können vom Einschalten der SQL Anywhere®-Transferkompression profitieren. Im WAN ist dies obligatorisch, sofern nicht auf CmdSeq verzichtet wird (siehe *RDP statt CmdSeq in WANs*).

  Generell wird vom Einsatz solcher langsamen LANs jedoch dringend abgeraten.

- RDP statt CmdSeq in WANs

  Werden per WAN angebundene Arbeitsstationen betrieben, sollten diese unbedingt per RDP (Remote Desktop Protocol, das native Fernzugriffs-Verfahren von Microsoft Windows®) angebunden werden.

  Generell wird vom Einsatz von CmdSeq (Command Sequence, das native Protokoll von SQL Anywhere®) in WANs dringend abgeraten, da die WAN-Latenz üblicherweise inakzeptabel hohe Wartezeiten erzeugt, die zu unvorhersehbaren Konsequenzen (Verbindungsabbrüchen,

Transaktionsabbrüchen) führen kann.

- iSCSI- und TCP-Offloading

  SANs profitieren üblicherweise vom Einschalten optional vorhandener Offloading-Kapazitäten für iSCSI und TCP, da der hierzu notwendige Datenfluss durch den Server unterbleibt, somit die Server-interne Transferkapazität für einen höheren netto-Durchfluss zur Verfügung steht.

  Dieser Effekt ist jedoch im Umfeld "typisch dimensionierter" QUIPSY® CAQ-Installationen als marginal anzusehen und tritt aufgrund der heutzutage üblichen, massiven internen Bandbreite meist nur bei sehr alten Servern auf.

Sonstige Maßnahmen

- Exklusiver QUIPSY®-Betrieb

  Grundsätzlich stören sich alle Software-Systeme gegenseitig, da z. B. mehr Positionswechsel auf der Festplatte, aber auch mehr Transfers zwischen den CPU-Kernen und CPU-Sockeln nötig werden, als bei exklusivem Betrieb. Optimal ist es daher, die gesamte Hardware (Server, Plattensystem, Netzwerk) **alleine** für QUIPSY® zu nutzen, sofern dies organisatorisch und wirtschaftlich möglich ist.

- GlassFish per SharedMemory

  Der QUIPSY® ApplicationServer (GlassFish) zeigt bessere Performance, wenn er auf der gleichen Maschine wie der QUIPSY® DatabaseServer (SQL Anywhere®) läuft, und per SharedMemory statt TCP/IP auf die Datenbank zugreift.

  *Dies hat leider auf die Geschwindigkeit der grafischen Benutzeroberfläche von QUIPSY® RB **wenig** Auswirkung, sehr wohl jedoch auf die Geschwindigkeit umfangreicher Reklamations-Auswertungen.*

- Keine Virtualisierung

  Die Nutzung von Virtualisierungstechnologie reduziert die Leistung um etwa 5 bis 25% (je nach eingesetztem Hypervisor und dessen jeweiliger Konfiguration) und ist daher *aus Performance-Sicht* nicht optimal.

- VMware Tools, paravirtualisierte Treiber

  Beim Einsatz von VMware ESXi (z. B. vSphere) sollten paravirtualiserten Treiber ("PVSCSI", "VMXNET3" für vHDD und vLAN eingesetzt werden, um massive Performance-Nachteile zu vermeiden.

- SQL Anywhere® kann Lastverteilung bereitstellen (kostenpflichtige Zusatzoption).

  Die Umschaltung erfolgt transparent, erfordert jedoch trotzdem einen Neustart des Clients.

  *Hierdurch werden für die Lesezugriffe andere Server verwendet als für die Schreibzugriffe, wodurch sich die Lesegeschwindigkeit skalieren lässt. Die QUIPSY®-Software ist jedoch **nicht** mit Read-Fan-Out-Konfigurationen getestet und es erfolgt daher keine Unterstützung bei der Einrichtung und Wartung von Spiegel-Konfigurationen durch die QUIPSY QUALITY GmbH & Co. KG.*

Beispielkonfigurationen

Die folgenden Konfigurationen stellen exemplarische (beispielhafte), ausgewogene Server-Konfigurationen dar, die keinen Anspruch darauf erheben, für jeden Kunden, jede Nutzerzahl, und jede Datenmenge absolut optimal zu sein. Vielmehr dienen Sie dazu, ein grobes Verhältnis zwischen

Nutzerzahl, CPU-Zahl und Anzahl der Festplatten zu vermitteln.

Zu beachten ist, dass bei intensiver Nutzung (hohe Prüffrequenzen, große Stichprobenumfänge, viele Nutzer) die Schreibleistung des Journals eine zunehmende Rolle spielt. In diesem Fall kann die Nutzung von SSDs als Journal-Medien sinnvoll sein. Umgekehrt können mehr CPU-Kerne bei üppiger RAM-Ausstattung gerade umfangreiche Auswertung deutlich beschleunigen. **Ohne ausreichend RAM bringen diese jedoch keinerlei Zugewinn.**

*Im Grundsatz gilt stets das Prinzip, dass ausgehend von der Standard-Konfiguration **anhand der tatsächlich vom Server gemeldeten Ressourcen-Auslastung** der jeweils im laufenden Betrieb über einen gewissen Zeitraum (z. B. einen Tag oder eine Woche gemittelt) ermittelte Flaschenhalst zu beseitigen ist, unabhängig von den untengenannten Konfigurations-Beispielen.*

Minimal-Konfiguration (bis 5 Nutzer)

Diese Konfiguration stellt aus Performance-Sicht das *absolute Minimum* dar, welches im Hinblick auf vernünftige Arbeitsgeschwindigkeit betrieben werden sollte und ist bei eher *sporadischer* Nutzung durch *maximal* fünf gleichzeitigen QUIPSY®-Anwendern *gerade ausreichend*, sofern das Archiv komplett im RAM Platz findet.

Von der Beschränkung auf diese Mindest-Konfiguration wird abgeraten.

*Aus Gründen der Systemverfügbarkeit sollte diese Konfiguration nur in **absoluten Ausnahmefällen ("Notbetrieb")** genutzt werden: Fällt eine der beiden Platten aus, ist das System nur mit hohem Aufwand wieder in Betrieb zu nehmen, entsprechend sind lange Ausfallzeiten die Folge. Siehe hierzu unbedingt das White-Paper "Datensicherheit mit SQL Anywhere®"*

- 2 CPU Cores @ 2 GHz
- 1x SATA (Journal, temporäre Dateien)
- 1x SATA (Archiv, opt. Journal-Spiegel)

Standard-Konfiguration (bis 10 Nutzer)

Diese Konfiguration stellt eine Empfehlung für *regelmäßige* Nutzung durch etwa fünf bis 10 gleichzeitige Nutzer dar, sofern das Archiv zweimal im RAM Platz findet. Sie berücksichtigt sowohl Performance-Anforderungen als auch Datensicherheit und Systemverfügbarkeit. Fällt eine Festplatte aus, ist das System trotzdem noch weiter *mit geringfügig eingeschränkter Leistung* arbeitsfähig.

Dies stellt die für die meisten Anwender empfohlene Standard-Konfiguration dar.

- 4 CPU Cores @ 2.5 GHz
- 2x SAS @ 15K (RAID 1) (Journal, temporäre Dateien)
- 6x SATA (RAID 6) (Archiv, opt. Journal-Spiegel)

Leistungsfähige Konfiguration (ab mehr als 15 Nutzer)

Dies stellt eine beispielhafte Konfiguration dar, um das Verhältnis von CPU-Kernen zu Festplatten aufzuzeigen. Bei zunehmender Nutzerzahl und Datenmenge sollte entsprechend der *durch den Server gemeldeten, tatsächlichen* Resourcen-Engpässe eine sinnvolle Nachrüstung mit CPU, RAM und Festplatten erfolgen. Ab etwa 24 Festplatten (bei RAM in der Größe eines Mehrfachen der Archiv-Größe bereits schon bei etwa acht Festplatten) ist bei handelsüblichen RAID-Controllern mit keiner *nennenswerten, zusätzlichen* Leistungserhöhung zu rechnen. Spätestens in dieser Ausbaustufe erfolgt oftmals ein Lastwechsel zu *CPU-gebundener* Performance, d. h. *ab diesem Punkt* lohnt es sich eher, in weitere CPU-Kerne zu investieren.

- 8 CPU Cores @ 3 GHz
- 2x SSD (RAID 1) (temporäre Dateien, Spiegel-Journal)

- 4x SAS @ 15K (RAID 10) (Journal)
- 8x SATA (RAID 6) (Archiv)

  Mit sinkenden Preisen von servertauglichen SSDs kann es durchaus Sinn machen, ***statt 4x SAS @ 15K (RAID 10) auf 2x SSD (RAID 1) zu wechseln*** *für das Journal. Die Schreibleistung wird, gerade bei modernen SSDs, wesentlich höher sein als bei HDDs, der Preisunterschied ist möglicherweise marginal. Das Archiv zeigt bei üblichen QUIPSY®-Konfigurationen hingegen wenig Zugewinn durch SSDs, sofern der Server über genug RAM verfügt, jedoch kann es je nach Marktsituation gelegentlich durchaus möglich sein, dass 4x SSD (RAID 6) zeitweilig günstiger zu erwerben ist als 8x SATA (RAID 6), was in jenem seltenen Fall durchaus eine Berechtigung zum Erwerb einer reinen SSD-Lösung darstellt.*

Rechtliche Hinweise

Die genannten Konfigurationsdaten sind lediglich als generelle, beispielhafte Information zu verstehen und stellen keine rechtsverbindlichen Leistungsgarantien dar.