Firebird White Paper

Anforderungen für eine virtuelle Maschine mit guter Firebird Performance

Holger Klemt, Februar 2024

Entscheidend sind die folgenden Faktoren für die Host-Konfiguration

1. Single Dedicated VM, es sollten keine anderen I/O-relevanten VM auf der gleichen Hardware/CPU laufen.
2. CPU Auswahl: Wichtig: Möglichst ca. 5 GHz Maximalspeed verfügbar und aktiviert auf mindestens einem Kern.
3. CPU Auswahl: Wie viele Cores die CPU hat, ist weniger wichtig, es sei denn, es sind sehr viele User mit Firebird mit sehr hoher Schreib-/Leselast verbunden. Beim Single Dedicated VM Betrieb sollten einfach alle in der CPU-Hardware verfügbaren Cores in der VM aktiviert sein.
4. Datenträgeranbindung: Maximum IOPS bei minimaler Latenz, am besten High Speed NVMe direkt auf dem Mainboard, wo die Host CPU läuft.
5. Ein Betrieb auf einer externen Storage ist selten schnell genug.

Warum ist das so?

Firebird ist ein Open Source Datenbanksystem, welches sich durch extreme Stabilität bei sehr hoher Geschwindigkeit auszeichnet. Extrem wichtig für diese Geschwindigkeit ist dabei in der Architektur ein Merkmal, was man als Careful Write bezeichnet.

Alle in der Datenbankdatei schreibenden Operationen passieren nicht in irgendeinem externen Transaktionslog-Bereich, der im Fehlerfall dann nach einem Reboot für eine Datenbanksperre sorgt, solange bis das dann hoffentlich noch vollständige Transaktionslog in der Datenbankdatei umgesetzt wurde, bevor neue Client-Verbindungen wieder aufgenommen werden können (der ältere Leser dieses Dokuments erinnert sich gerne auch an die beim Windows- oder DOS-Systemstart erforderliche chkdsk-Ausführung, bis man mit dem Computer wieder arbeiten konnte). Genau dieses Verfahren setzen die meisten anderen Datenbankhersteller ein.

Bei Firebird hingegen sorgt der Firebird-Server-Prozess dafür, dass alle Schreiboperationen in der Datenbankdatei in einer exakt definierten Reihenfolge stattfinden müssen, damit der Inhalt dieser Datei auch im Worst Case (unerwarteter Reboot oder sonstiger Absturz) sofort nach wieder Anlaufen des Betriebssystems und damit auch des Firebird-Server-Prozesses wieder ohne jegliche Reparatur als Datenbank geöffnet werden kann und sofort für alle Schreib/Lese-Operationen verfügbar ist. Alles, was

jemals den Transaktionsstatus COMMIT in der Datenbank erreicht hat, ist mit 100% Sicherheit dann auch in der Datenbank-Datei so enthalten.

Begriffe aus anderen Welten wie "Index reparieren" gibt es in der Firebird-Welt nicht, es sei denn, das benutzte Betriebssystem mischt sich durch ungewünschte Änderungen der Schreibreihenfolge in die Careful Write Reihenfolge ein und schreibt das andersherum. Das lässt sich leider nicht bei jedem Betriebssystem ausschließen und gerade eine virtuelle Host-Konfiguration mit externer Storage bietet da oft mit besserer Performance begründete Optimierungen an, deren Auswirkungen man aber erst im Worst Case erfährt.

Grundlagen

Der Firebird Serverprozess sorgt über das übliche Attribut Forced Writes dafür, das Schreiboperationen in der korrekten Reihenfolge an das Betriebssystem übermittelt werden und der Serverprozess auf die erfolgreiche Abschlussmeldung vom Betriebssystem wartet, bevor die Client Applikation die Meldung bekommt, dass der Vorgang erfolgreich war. Ein Caching ist dabei explizit nicht gewünscht, weil, wenn es einen Fehler gab, soll der Anwender nicht den Eindruck bekommen, dass seine Daten bereits erfolgreich geschrieben wurden.

Für diese Schreiboperation hat der Firebird Serverprozess die gesamte Datenbankdatei in Pages organisiert.

Während vorne in der Datei Informationen wie Metadaten, Transaktionszustände und Seiteninhalts-verzeichnisse zu finden sind, findet man weiter hinten in der Datenbank-Datei zu den Tabelle relevante Daten Pages und Index Pages.

Mit einem Tool wie dem Process Monitor von syinternals.com kann man zum Beispiel auf einem Windows-Server die genaue Schreibreihenfolge beobachten. Die Positionen sind dabei also sehr stark auf der Datenbank-Datei in relativ kleinen Blöcken verteilt; vorne/hinten/vorne/hinten... usw. Und je mehr Clients gleichzeitig auf der Datenbank arbeiten, umso häufiger muss das in der Datei koordiniert stattfinden.

Und die Inhalte der Datenbank werden von der Datenbanksoftware selbst im RAM vorgehalten, um die Lesezugriffe zu minimieren. Ab der Firebird Version 3 wird dabei auch ein gemeinsamer Cache für alle Clients im RAM vorgehalten und daher kann dieser bei sehr viel höherer Anzahl an Clients deutlich weniger Lesezugriffe machen. Ebenso wird ab Firebird 3 dann auch jeder Core sinnvoll benutzt, aber nur, wenn die Anzahl der aktiven Clients, die auch wirkliche SQL-Befehle machen, der Anzahl entspricht. Eine höhere Anzahl wird automatisch sauber verteilt. Es ist aber eindeutig besser, wenn auf einem Server z.B. 8 Cores je 4 GHz bei Bedarf erreichen können, als wenn man 64 Cores hat, aber keiner mehr als 2 GHz liefert. Das gilt auch insbesondere, weil sich sämtliche Cores die Datenleitungen im CPU-Socket teilen müssen und bei speicherintensiven Operationen führt es sonst im Multicore-Betrieb schnell zu Serialisierungen, obwohl die gesamten CPU-Cores eigentlich mehr könnten.

Die Geschwindigkeit, mit der dann der Server intern in der CPU die Daten verarbeiten kann, bis diese wieder in den RAM zurückgehen, ist dabei explizit abhängig von der maximalen Taktung des Prozessors. Da es sich um wenig echte CPU Load handelt, sondern eher um sehr intensive Byte-Schieberei, kann man davon

ausgehen, dass wichtige Operationen innerhalb der Datenbank, die noch nicht auf den externen Datenträger gehen, bei einem Takt von 2 GHz die doppelte Zeit benötigen im Vergleich zu einer CPU, die es mit 4GHz abarbeitet.

Unterschied zu anderen Systemen

Im Gegensatz zu einem transaktionslogbasierten System, das nur ein Transaktionslog als große Einzeldatei vorhalten muss und aus diesem und den Daten der restlichen Datenbank-Datei bei einem Systemabsturz wieder eine funktionsfähige Datenbank erzeugen muss, bevor man damit wieder arbeiten kann, ist also das I/O-Verhalten von Firebird mit höchster Priorität auf Zuverlässigkeit und sofortiger Benutzbarkeit auch nach Systemabsturz völlig anders.

Ein transaktionslogbasiertes System pflegt sozusagen nur eine große Datei beliebiger Länge, die es irgendwann dann wieder in die Datenbankdatei einarbeitet und in der Zwischenzeit den realen Inhalt der Datenbankdatei im Speicher vorhält.

Man kann das Verhalten dieser Architektur also mit einem Kopiervorgang einer großen Datei von 5 GB vergleichen, die von einem Pfad in einen anderen Pfad kopiert wird. Bei Firebird hingegen ähnelt das Profil eher einem parallelen Kopiervorgang, der gleichzeitig versucht, 100000 kleine Dateien in 10000 verschiedene Pfade zu kopieren.

Unter Windows sollte man das im NTFS Filesystem nicht wirklich vergleichen, weil das jede Windows-Maschine eh an die Leistungsgrenze bringt, aber da der Firebird-Server den Dateiinhalt selbst verwaltet ohne dafür einzelne Windows-Dateien zu erzeugen, ist da dieser Teil vom Problem zu vernachlässigen. Und dabei ist dann noch ganz wichtig, dass der Firebird-Server aufgrund der Forced Writes Anforderung das Betriebssystem beauftragt, den Schreibvorgang sofort durchzuführen und Erfolg zu melden, bevor der nächste kommt. Daher auch die wichtige Latenzanforderung.

IOPS und Latenz

Geeignete moderne Serverhardware, wie unsere üblicherweise nicht virtualisierten IFS-Server, erreichen dabei auf NVMe-Laufwerken direkt auf dem Mainboard Schreib/Leseleistungen von 4-5 GB pro Sekunde und selbst in Benchmarks mit kleineren Paketen von 16 oder 32 k multithreaded sind das noch 1-1,5 GB pro Sekunde.

Der ATTO Disk Benchmark ist dabei ein gutes Werkzeug. Für den effektiven Betrieb mit Firebird empfehlen wir aber den IBExpert Benchmark, den man kostenlos unter ibexpert.com/benchmark herunterladen kann.

Unsere IFS-Server sind dabei mit Hardware ausgestattet, die dabei der Software nahezu latenzfrei 300000 - 450000 IOPS bereitstellen können und damit auf diese Werte kommen.

Worst Case Szenarios

Je nach Hersteller sind dabei auch bei den großen Herstellern transaktionsbasierter Datenbanksysteme nicht nur im Fehlerfall sehr gute administrative Kenntnisse erforderlich, um das System auch mit sehr vielen Benutzern und Daten zuverlässig wieder anzufahren, wenn es konkrete Fehler gab. Oder wie es in einem Kundenprojekt mit einem Test der Oracle Datenbank ergeben hat: Einfach mal nach dem Absturz während des Reparaturvorgangs den Server erneut zum Absturz bringen. Der Test bei einer großen Presseagentur erbrachte dabei eine Mindestwartezeit von 30 Minuten, bis das System wieder erreichbar war.

Ein auch immer wieder gern aufgeführtes Beispiel ist ein System, das u.a. in amerikanischen Panzern verbaut ist und die Position der nächsten Artillerie-Angriffe anzeigt, wo man auch in einem amerikanischen Panzer dann nicht rumstehen sollte. Das basiert auf der Technik vom Firebird-Vorgänger InterBase und zeichnet sich dadurch aus, dass auch nach dem bis heute nicht vermeidbaren elektromagnetischen Impuls, der in einem Panzer auftritt, wenn man da vorne eine Granate abschießt, sämtliche Computer aufgrund von Antenneneffekten auf der Leiterbahn abstürzen lässt und die Frage nur ist, wann sind nach dem Reboot wieder Daten verfügbar. Ein erforderliches Warten mit dem nächsten Schuss, bis der Computer wieder gebootet hat, wäre dabei sicherlich taktisch ein Problem.

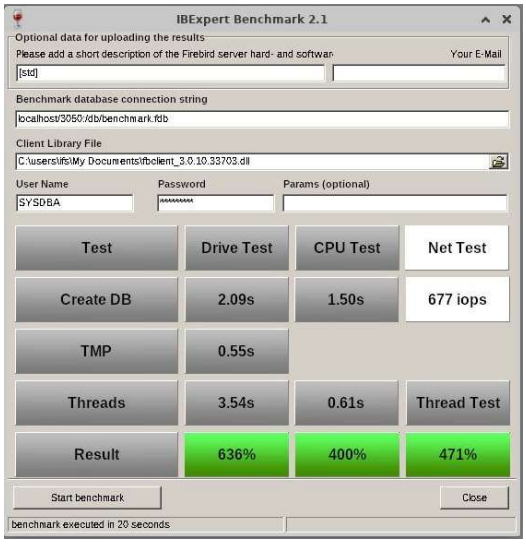
Fazit

Wenn aber ein Server für die Virtualisierung eingesetzt wird und dieser mit einer dafür geeigneten Hardware, wie oben beschrieben ohne virtuellen Host drumherum sehr gute Benchmarkwerte liefert, wird die Endgeschwindigkeit dann nur noch durch die der VM Host Softwarelösung und der dabei gewählten Konfiguration beeinflusst. Wenn es aber ungeeignete Hardware ist, hilft auch die beste Anpassung der VM Host Konfiguration nicht dabei, das System auch nur annähernd auf ein geeignetes Geschwindigkeitslevel zu bringen.

Selbst die in einem großen deutschen Krankenhaus eingesetzte Storage, die mehrere Petabyte Speicher hatte und sämtliche Caching-Verfahren mit eingebauter RAM/SSD/NVME Technik konnte, war für den Einsatz als Storage für eine Firebird VM deutlich zu langsam. Da half es auch nicht, dass die Storage ca. 1 Mio. Euro gekostet haben sollte. Weniger IOPS-lastige oder latenzkritische Anwendungen waren mit dem System sehr gut einsetzbar. Für den Einsatz als Firebird-Server war dieses System aber ungeeignet und jede 100 Euro Sata SSD direkt in der Hosthardware eingebaut, hätte dafür deutlich bessere Leistungen erbracht.

Es gilt also, unabhängig von Virtualisierung eine geeignete Hardware auszuwählen. Ob diese geeignet ist oder nicht, kann man mit dem IBExpert Benchmark unter Windows jederzeit selbst testen. Wenn das Ergebnis deutlich im grünen Bereich liegt, kann man diese Hardware dafür einsetzen, wenn die Werte unter 100% liegen, dann eher nicht. Mit 100% haben wir einen Server referenziert, den wir vor 13 Jahren an Kunden ausgeliefert haben. Unsere aktuellen Server erreichen da für ein Budget von ca. 5000 Euro pro Server 400%-650%.

https://ibexpert.com/benchmark



Haben Sie Fragen?

IBExpert GmbH
Oldenburger Straße 233
26203 Wardenburg
Deutschland

www.ibexpert.com
sales@ibexpert.biz

Telefon: +49 4407 3148770