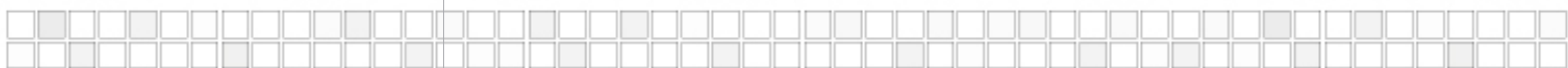




ENTERPRISE MESSAGING MIT STÄNDIGER VERFÜGBARKEIT

Reduzierung operativer Risiken und einfachere Administration





INHALTSVERZEICHNIS

Einführung	2
Enterprise Messaging als wichtige Voraussetzung	4
Java Message Service	5
Voraussetzungen für Enterprise Messaging	6
Traditionelle fehlertolerante Architekturen	8
Hardware-Hochverfügbarkeitsarchitektur	8
Software-Hochverfügbarkeitsarchitektur	9
Sonic Continuous Availability Architecture™	11
Broker-Replikation	12
Replikationsverbindungen	15
Client-Verbindungen mit ständiger Verfügbarkeit	16
Nachrichtenzuverlässigkeit	17
Zuverlässigkeit von Transaktionen	18
Flexible Implementierungsmöglichkeiten	19
Zusammenfassung	20

EINFÜHRUNG

Immer häufiger müssen Unternehmen in der Lage sein, geschäftliche Aktivitäten rund um die Uhr in Echtzeit zu unterstützen. Entsprechend steigt auch die Nachfrage nach hoch verfügbaren Unternehmens-Systemen. Diese Systeme bestehen aus mehreren Klassen von Servern, die durch unterschiedliche Verfügbarkeitsmerkmale gekennzeichnet sind und jeweils potenzielle Quellen für Systemausfälle darstellen können. Deshalb stehen IT Organisationen vor dem Problem, eine optimale Verfügbarkeit der Systeme sicherzustellen und gleichzeitig einzelne Ausfallursachen zu eliminieren.

In zunehmendem Maße verlangen Unternehmen, dass ihre Systeme ununterbrochen verfügbar sind und weder durch geplante noch durch ungeplante Ausfälle beeinträchtigt werden. Stillstandszeiten, schlechte Performance und geplante Systemunterbrechungen stören den Geschäftsbetrieb, erhöhen die Kosten und wirken sich negativ auf die Zufriedenheit der Kunden aus. Die Kosten von Systemausfällen äußern sich nicht nur in den Produktivitätsverlusten der IT-Organisation und der von ihr versorgten Geschäftsbereiche, sondern auch in Umsatzeinbußen, Konventionalstrafen und mitunter auch Bußgeldern.

Die Kosten eines Systemausfalls setzen sich folgendermaßen zusammen:

Produktivität der betroffenen Benutzer = Stundensatz der betroffenen Benutzer x Anzahl der Stunden mit Systemunterbrechung

+ Verlorene IT-Produktivität = Stundensatz der betroffenen Mitarbeiter x Anzahl der Stunden mit Produktivitätsausfall

+ Auswirkungen auf Kundenservice und Glaubwürdigkeit

+ Umsatzverlust = Umsatzverlust pro Stunde x Anzahl von Stunden mit Systemausfall

+ Weitere geschäftliche Einbußen

Kosten für Überstunden = Stundensatz der Mitarbeiter x Anzahl Überstunden

+ Verdorbene Waren

+ Finanzielle Strafen oder Bußgelder

In der Regel stellen Umsatzeinbußen das größte zugleich aber auch am schwierigsten zu quantifizierende finanzielle Risiko dar. Wie wichtig ein Echtzeitbetrieb von Unternehmenssystemen ist, hängt dabei von der jeweiligen Branche ab. Die folgende Tabelle zeigt die entsprechenden Kosten für unterschiedliche Arten von Unternehmen.

Durchschnittliche Kosten eines ungeplanten Systemausfalls für amerikanische Unternehmen aus verschiedenen Branchen¹

Börsenmakler	\$6,45 Millionen pro Stunde
Kreditkartenprüfung	\$2,6 Millionen pro Stunde
Info-Dienste / Verkaufsdienste über kostenfreie Rufnummern	\$199.500 pro Stunde
Katalogvertriebszentrale	\$90.000 pro Stunde
Flugreservierung	\$89.500 pro Stunde
Betreiber von Geldausgabeautomaten	\$14.500 pro Stunde

Darüber hinaus sind die Auswirkungen auf Benutzer bei unerwarteten Systemausfällen wesentlich schwerwiegender als bei planmäßigen Ausfällen. Arbeitswissenschaftler haben festgestellt, dass die Konzentration des Benutzers sofort nachlässt, wenn das System länger als 2 Sekunden braucht, um auf seine Eingabe zu reagieren. Da sich der Benutzer anschließend neu konzentrieren muss, entsteht ein Produktivitätsverlust, der sich bereits in Minuten beziffern lässt. Bei Systemstörungen mit einer Dauer von mehr als 20 Minuten wenden sich Benutzer einer anderen Aufgabe zu, was zu einer Unterbrechung von Geschäftsprozessen und aus der Sicht des Benutzers zu effektiven Ausfallzeiten von mehreren Stunden führt.²

Darüber hinaus können Systemausfälle langfristige negative Auswirkungen auf den Ruf eines Unternehmens haben, was nicht nur dazu führen kann, dass sich Kunden vom Unternehmen abwenden, sondern auch gerichtliche Schritte und Geldbußen nach sich ziehen kann.

Um Umsatz und Gewinn durch eine unterbrechungsfreie Unterstützung von Geschäftsprozessen zu maximieren, müssen Unternehmen sowohl intern als auch bei ihren externen Verbindungen ein hohes Maß an Verfügbarkeit gewährleisten, um somit eine kontinuierliche Betreuung ihrer Geschäftspartner und Kunden sicherzustellen. Eine wesentliche Maßnahme zur Lösung dieses Problems ist die Festigung der Datenbanken, der Netzwerke, der Web-Servers und der physischen Hardware. Dieses Whitepaper beschäftigt sich mit den Problemen, die mit den derzeitigen Ansätzen für unternehmensweite, fehlertolerante Messaging-Lösungen verbunden sind, und stellt ein neues und besseres Paradigma vor, mit dem Unternehmen ihre operative Verfügbarkeit verbessern können: die Sonic Continuous Availability Architecture™.

1. InternetWeek 4/3/2000 and "Fibre Channel: A Comprehensive Introduction", R. Kembel, 2000, Seite 8; basierend auf einer Befragung, die im Rahmen einer Untersuchung zum Thema Notfallplanung durchgeführt wurde.

2. Forrester Research: 15. März 2004 "An Executive Guide To High Availability", von Bob Zimmerman



ENTERPRISE MESSAGING ALS WICHTIGE VORAUSSETZUNG

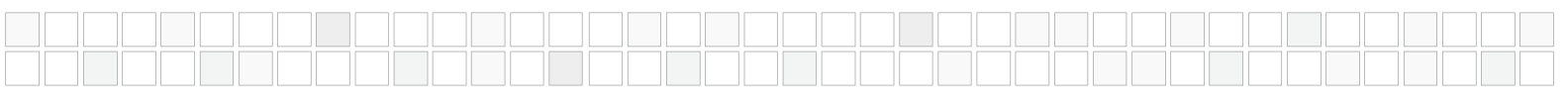
Die Bedeutung einer unternehmensweiten Messaging Infrastruktur für die geschäftliche Kommunikation wächst, wenn Unternehmen versuchen, ihre operative Produktivität zu verbessern. Auch bei eingeplanten Systemunterbrechungen müssen die anderen Systeme, Abteilungen, Geschäftsbereiche und Partner weiterarbeiten können. Darüber hinaus darf ein Systemausfall nicht zum Verlust von Daten führen.

Im hart umkämpften Wettbewerbsumfeld von heute müssen Unternehmen und ihre Geschäftspartner miteinander verbunden sein. Die Integration von Systemen ermöglicht eine schnellere Beantwortung von Angebotsanfragen, eine schnellere Abwicklung von Zahlungen und eine schnellere Bereitstellung von Informationen zum aktuellen Status eines Vorgangs. Aufgrund der Notwendigkeit, eine Anbindung der außerhalb der Firewall vorhandenen Systeme zu realisieren und unternehmenskritische Daten zu übertragen, ist es unabdingbar, einen zuverlässigen, verfügbaren und sicheren Informationsaustausch zu unterstützen.

Um diesen geschäftlichen Anforderungen gerecht zu werden, haben viele Unternehmen eine IT Infrastruktur geschaffen, in deren Mittelpunkt ein RPC-artiges (Remote Procedure Call) Messaging-System steht. RPC-basierte Systeme (wie die von Applikationsservern bereitgestellten) wurden in den neunziger Jahren populär und umfassten die Varianten CORBA, COM/DCOM und RMI. Leider sind diese Systeme sehr brüchig, da die Applikationen dabei über eine synchrone Kommunikation eng miteinander verkoppelt werden. Entsprechende Systeme basieren auf Punkt-zu-Punkt-Verbindungen und verfügen über keine einheitliche Schnittstelle.

Ein weiteres Merkmal RPC-artiger Messaging-Systeme ist, dass bei der Entwicklung von unternehmensweiten Client-Applikationen sehr strenge Vorgaben eingehalten werden müssen. Aufgrund der synchronen Eigenschaften des Systems müssen alle Systeme und Applikationen verfügbar sein, wenn sie benötigt werden. Jede Client Applikation muss die mit den anderen Applikationen verbundenen APIs genau kennen. Ändert sich eine Applikation oder werden neue Applikationen in das System integriert, wächst die Anzahl der benötigten Schnittstellen exponentiell. Wenngleich dieser Ansatz bei der Anbindung einer begrenzten Anzahl von Applikationen sinnvoll sein mag, wirkt sich eine eng gekoppelte synchrone Interaktion auf eine größere Umgebung mit heterogenen Systemen eher negativ aus.

In den neunziger Jahren entstand mit der Einführung von MOM-Systemen (Message-Oriented Middleware) ein stärker skalierbarer Ansatz, der auf einer lose gekoppelten, asynchronen Architektur basiert. Ein System auf MOM-Basis nutzt eine standardisierte Messaging-Schnittstelle, um Funktionen wie Fehlerbehandlung oder die garantierte Zustellung von Nachrichten zu koordinieren. Da Applikationen hierbei asynchron miteinander kommunizieren können, müssen nicht alle Systeme verfügbar sein, damit das Netz von Applikationen weiter



funktionieren kann. Insgesamt stellt MOM eine deutliche Verbesserung gegenüber dem RPC-Kommunikationsmodell dar, da weniger Netzwerkverbindungen benötigt werden, eine flexiblere Implementierung ermöglicht wird und weniger Programmieraufwand erforderlich ist, um Konfigurationsänderungen zu unterstützen.

JAVA MESSAGE SERVICE

1998 führte Sun Microsystems den Java Message Service (JMS) ein. Bei der JMS API, die von Sun in enger Zusammenarbeit mit führenden Anbietern von Enterprise-Messaging-Lösungen entwickelt wurde, wurden die Hauptelemente von RPC und MOM miteinander kombiniert. Enterprise Messaging gilt inzwischen als essenzielles Tool für die Entwicklung von unternehmensweiten Client-Applikationen. Detaillierte Informationen zu JMS sind im Internet unter <http://java.sun.com/products/jms/> zu finden.

Mit Progress® SonicMQ® hat Progress Software ein Enterprise-Messaging-Produkt entwickelt, das auf JMS basiert und als das robusteste und flexibelste auf Standards basierende Enterprise-Messaging-System gilt, das auf dem Markt erhältlich ist. Informationen zu SonicMQ sind im Internet unter <http://www.sonicsoftware.com/products/sonicmq> zu finden.

Enterprise Messaging und JMS im Besonderen stellen einen zuverlässigen und flexiblen Service für den unternehmensweiten Austausch wichtiger Daten und Ereignisse bereit. Durch JMS können Client Applikationen über ein gut definiertes und lose gekoppeltes Messaging-Protokoll miteinander kommunizieren. JMS API erweitert dieses Protokoll durch ein gemeinsames API- und Anbieter-Framework, das die Entwicklung portabler, sicherer und zuverlässiger nachrichtenbasierter Applikationen ermöglicht.

Bei vielen unternehmensweiten Client-Applikationen dürfen keine Nachrichten verloren gehen oder doppelt zugestellt werden. Deshalb muss bei vielen JMS Applikationen sichergestellt werden, dass eine Message mindestens einmal, aber auch nicht häufiger zugestellt wird. Diese Zuverlässigkeitsstufe bei der Zustellung von Nachrichten trägt die Bezeichnung 'Exactly Once'.

JMS definiert mehrere Zuverlässigkeitsmechanismen für Nachrichten. Die zuverlässigste Möglichkeit der Produktion einer Nachricht besteht darin, die Nachricht als persistente Nachricht innerhalb einer Transaktion zu senden. Die zuverlässigste Möglichkeit, eine Nachricht zu konsumieren, besteht darin, die Nachricht aus einer Queue oder über ein dauerhaftes Abonnement innerhalb einer Transaktion zu empfangen.



VORAUSSETZUNGEN FÜR ENTERPRISE MESSAGING

Im normalen Betrieb verwendet JMS die Nachrichtenzuverlässigkeitsstufe *Exactly Once*. Gleichwohl kann diese Zuverlässigkeitsstufe bei einem Hardware-, Netzwerk- oder Betriebssystemausfall mit keinem JMS-Zuverlässigkeitsmechanismus realisiert werden. Es gibt vier Klassen von Nachrichtenfehlern, die sich kritisch auf die Arbeit des Unternehmens auswirken können.

1. HÄNGEN GEBLIEBENE NACHRICHTEN

Nach einem Systemausfall gehen Nachrichten im ausgefallenen Messaging-System verloren und werden von der Client-Applikation niemals empfangen.

2. DOPPELTE NACHRICHTEN

Nach einem Systemausfall verbleiben Nachrichten in einem unbekanntem Zustand. Dies kann bei einer Wiederinbetriebnahme dazu führen, dass die Client-Applikation oder der Messaging Broker Nachrichten doppelt verschickt, so dass beispielsweise eine Lastschrift zweimal verschickt wird.

3. NACHRICHTEN IN DER FALSCHEN REIHENFOLGE

Nach einem Systemausfall ist unsicher, ob die Nachricht zugestellt worden ist oder sich noch auf dem Transportweg befindet. Bei einem Neustart des Systems kann dies dazu führen, dass die Client Applikation Nachrichten in einer falschen Reihenfolge erhält, beispielsweise ein "Stornierungs-" oder "Aktualisierungs"-Auftrag wird vor dem ursprünglichen Auftrag empfangen.

4. ABGEBROCHENE TRANSAKTIONEN

Nach einem Systemausfall bleiben Transaktionen in einem unvollständigen Zustand. Nach dem Neustart müssen die transaktionsbezogenen Nachrichten zurückgerollt und verworfen werden.

Diese Störungen können zu erheblichen Verzögerungen bei der Abwicklung operativer Geschäftsprozesse führen. In allen Fällen ist ein "forensisches" Team erforderlich, das einschreitet und den Zustand der Nachricht wiederherstellt. Hierzu gehört das Zurückrollen transaktionsbezogener Nachrichten, die Wiederherstellung hängen gebliebener Nachrichten, die Identifizierung und Beseitigung doppelter Nachrichten und die Wiederherstellung der Reihenfolge von Nachrichten. Wird der Zustand der Nachricht nicht wiederhergestellt, so erhalten die Client-Applikationen verfälschte Nachrichtenströme, was wiederum zu unvollständigen und fehlerhaften Transaktionen führt.



Damit ein Enterprise-Messaging-System bei einem Hardware-, Netzwerk-oder Systemausfall die Nachrichtenzuverlässigkeitsstufe Exactly Once realisieren kann, muss es eine Systemredundanz gewährleisten. Erreicht wird diese Redundanz durch einen primären und sekundären (Backup) Broker, die gemeinsam die folgenden Fehlertoleranzanforderungen erfüllen müssen:

1. FULL-STATE RECOVERY:

Bei einem Systemausfall muss der sekundäre Broker in der Lage sein, die Rolle des ausgefallenen primären Brokers zu übernehmen. Dabei muss er den Status der Nachricht vollständig wiederherstellen.

2. HOT-FAILOVER:

Bei einem Systemausfall muss der sekundäre Broker mit minimaler Failover-Latenz in einen aktiven Zustand übergehen. Durch die Reduzierung der Latenzzeit auf ein Minimum wird verhindert, dass die Puffer der Client-Applikationen überlaufen oder Applikationen ausfallen.

3. CLIENT FAILURE TRANSPARENCY:

Bei einem Systemausfall müssen die Enterprise-Applikationen transparent auf den sekundären Messaging Broker umgeschaltet werden. Dabei bleibt die Verbindung der Enterprise-Applikation mit dem Messaging-System so lange erhalten, bis der Wechsel zum sekundären Broker abgeschlossen ist.



TRADITIONELLE FEHLERTOLERANTE ARCHITEKTUREN

Auf ihrem Weg vom Client zum Zielort durchläuft eine Nachricht möglicherweise eine Vielzahl von Netzwerken, Systemen und Applikationen. Dabei kann die Verfügbarkeit der Systeme gesteigert werden, indem die einzelnen Komponenten - Clients, Middleware und Datenbanken - gestärkt werden. Eine traditionelle Methode, mit der die Messaging-Infrastruktur fehlertolerant gemacht wurde, besteht darin, einen oder zwei Server miteinander zu kombinieren, indem sie eng aneinander gekoppelt und zentral verwaltet werden. Dabei werden die Server so konfiguriert, dass die Client-Applikation bei einem Systemausfall mit einem sekundären Server bereitgestellt wird. Dieser Vorgang des Umschaltens von einem Server auf einen anderen wird als "Failover" bezeichnet.

Ein hoch verfügbares bzw. fehlertolerantes System erscheint den Benutzern und Applikationen gegenüber als eine einzige Umgebung. Neben der Steigerung der Verfügbarkeit von Applikationen kann mit der Clustertechnologie auch eine Steigerung der Systemkapazität und eine effizientere Administration realisiert werden.

Hochverfügbarkeitslösungen gibt es bereits seit den achtziger Jahren, als die Digital Equipment Corporation (DEC) entsprechende Technologien in ihren VMS-Systemen einsetzte. Auch in der Mainframe-Welt bot IBM mit sysplex ein Hochverfügbarkeitskonzept an. Microsoft, Sun Microsystems und andere führende Hardware- und Softwarehersteller bieten skalierbare Hochverfügbarkeitslösungen an, deren Skalierbarkeit und Verfügbarkeit angepriesen wird. Bei einer Zunahme des Datenverkehrs oder einem Anstieg der Verfügbarkeitsanforderungen kann die Größe oder Anzahl aller oder einiger Teile des Systems gesteigert werden.

Mit Hochverfügbarkeitsarchitekturen wird verhindert, dass ein Ausfall des Betriebssystems nicht zu einem längeren Ausfall von Applikationen führt. Außerdem wird damit eine Umgebung geschaffen, in der die einzelnen Komponenten, aus denen sich das System zusammensetzt, online gewartet und aufgerüstet werden können. Eine hohe Verfügbarkeit lässt sich sowohl mit Hardware- als auch mit Softwarelösungen realisieren.

HARDWARE-HOCHVERFÜGBARKEITSARCHITEKTUR

Hardware-Hochverfügbarkeit (vertikale Skalierung) ist eine hardwarebasierte Methode, bei der eine Gruppe von Servern als einzelnes System agiert. In der Praxis wird eine solche Architektur geschaffen, indem mehrere Blade Server auf dem Rechner installiert werden, der das System steuert. Jeder Blade Server arbeitet völlig unabhängig von den anderen, obwohl alle auf die gleichen Datenanforderungen reagieren. Dabei ist das Betriebssystem des steuernden Servers für die Überwachung des Systems und die Erledigung administrativer Aufgaben verantwortlich und

muss beispielsweise entscheiden, wann eine Umschaltung erforderlich ist, und die Arbeitslast vom ausgefallenen Knoten auf einen funktionierenden Server verlagern.

Hardware-Hochverfügbarkeitsarchitekturen können aktiv-passiv sein, wobei einige Server für Failover Aufgaben reserviert werden und keine eigenen Applikationen betreiben. Alternativ kann eine solche Architektur auch aktiv-aktiv sein, wobei alle Server ihre eigenen Applikationen betreiben, gleichzeitig aber auch Reserveressourcen vorhalten, um bei Bedarf Failover-Aufgaben für andere Server übernehmen zu können. Die Realisierung einer entsprechenden Fehlertoleranz über die Hardware erfordert den Kauf spezieller und teurer Hardwareprodukte. Deshalb wird dieser Ansatz hauptsächlich bei High-End-Unternehmenssystemen verfolgt.

SOFTWARE-HOCHVERFÜGBARKEITSARCHITEKTUR

Software-Hochverfügbarkeit (horizontale Skalierung) ist eine Methode, mit der mehrere Server zu einer fehlertoleranten Lösung zusammengeschaltet werden, indem auf jedem einzelnen Server im System eine Betriebssystem-Clustersoftware installiert wird und die auf den primären und sekundären Servern betriebenen Applikationen in der Regel gespiegelt werden. Jeder der Server führt die gleichen Informationen. Administrative Aufgaben wie Lastverteilung, Ermittlung ausgefallener Knoten und Zuordnung von Failover-Aufgaben werden von allen Servern gemeinsam erledigt.

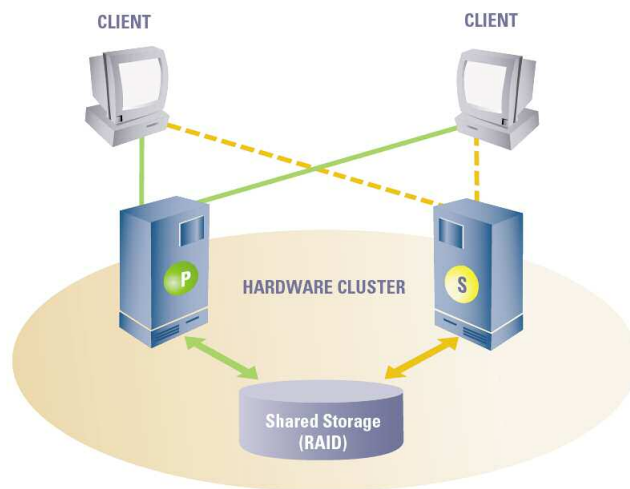


Abbildung 1. Diese Abbildung zeigt eine fehlertolerante Lösung. Bei einem Ausfall des Primärservers wird der sekundäre (Backup) Server aktiviert und beginnt mit der Wiederherstellung der Systeminformationen aus dem gemeinsamen Speicher. Dabei können verschiedene Messaging-Probleme auftreten, z.B. auf dem ausgefallenen Server hängen gebliebene Nachrichten, doppelt oder in falscher Reihenfolge verschickte oder empfangene Nachrichten sowie abgebrochene Transaktionen.

Bei Bedarf können dem Cluster problemlos, Server hinzugefügt oder auch aus dem Cluster entfernt werden. Deshalb ist Software-Fehlertoleranz eine skalierbare Lösung. Die typische Software-Fehlertoleranzarchitektur (siehe Abbildung 1) für Enterprise Messaging erfordert jedoch eine gemeinsame Datenbank, in der Informationen zum aktuellen Status von Nachrichten sowie zum Status erledigter Nachrichten gespeichert werden. Zwei Messaging-Server (ein primärer und ein sekundärer) bestimmen diese Messaging-Datenbank an einem Ort mit Netzwerkzugang.



Bis zu einem Ausfall hat nur der primäre Server read/write-Zugriff. Durch den Einsatz eines RAID-Datenbanksystems wird sichergestellt, dass die Datenbank keine Fehlerquelle darstellt, die zu einem Ausfall des gesamten Systems führen könnte.

Bei einem Ausfall des primären Servers initiiert das Drittsystem automatisch eine Umschaltung zum Sekundärserver, woraufhin der Sekundärserver damit beginnt, Nachrichten-Statusinformationen aus der gemeinsamen Datenbank auszulesen und interne Aufzeichnungsprotokolle und Dateien zu initialisieren. Dieser Failover-Prozess kann sich über mehrere Minuten erstrecken oder auch länger dauern. Darüber hinaus benachrichtigt das Fehlertoleranzsystem die Enterprise Applikationen über den Ausfall und liefert ihnen Informationen für einen Neuaufbau der Verbindung. Die Client-Applikationen werden mit dem Sekundärserver verbunden und führen den Wiederherstellungsprozess durch.

Trotz dieses komplexen Vorgangs wird durch den Failover- und Neuinitialisierungsprozess nicht die Nachrichtenzuverlässigkeitsstufe *Exactly Once* erreicht, da die Enterprise-Client-Applikationen nach wie vor auf hängen gebliebene, doppelt oder in der falschen Reihenfolge empfangene Nachrichten sowie abgebrochene Transaktionen treffen können. Alle vier Arten von Nachrichtenfehlern können das Betriebssystem beeinträchtigen.

SONIC CONTINUOUS AVAILABILITY ARCHITECTURE™

Mit der patentierten Sonic Continuous Availability Architecture (CAA) hat Progress Software eine neue und einzigartige Messaging-Architektur entwickelt, die den traditionellen Hochverfügbarkeitslösungen der derzeitigen Anbieter von Enterprise-Messaging-Lösungen deutlich überlegen ist und nicht nur eine höhere operative Verfügbarkeit gewährleistet, sondern auch eine kostengünstigere Entwicklung, Implementierung und Administration ermöglicht. Die Sonic CAA sorgt für hohe Verfügbarkeit in der Messaging-Schicht, wozu die Sonic Message-Broker, die Sonic Clients und die Kommunikation zwischen Clients, Brokern und Zielsystemen gehören. Dabei wird sowohl im normalen Betrieb als auch bei einem Ausfall die Nachrichtenzuverlässigkeitsstufe Exactly Once gewährleistet. Durch den Einsatz der Sonic CAA entfällt die Anschaffung kostspieliger RAID-Systeme, Betriebssystem-Clustersoftware oder Hochverfügbarkeitsplattformen von Drittanbietern für die Messaging-Schicht. Unabhängig von der Komplexität werden alle im Prozess befindlichen Transaktionen ohne zeit- und kostenintensive Roll-Back- und Wiederherstellungsmaßnahmen zu ihren Zielen geführt.

Die Sonic CAA (vgl. Abbildung 2) basiert auf einer Replikation zwischen primärem und sekundärem Broker, die durch eine Backchannel-Synchronisierung und fehlertolerante Client-Verbindungen realisiert wird. Diese Architektur unterstützt das erweiterte Unternehmen in einer bisher unmöglichen Art und Weise und ergänzt vorhandene Hochverfügbarkeitslösungen wie jene für Datenbanken, Applikationsserver und Web Server.

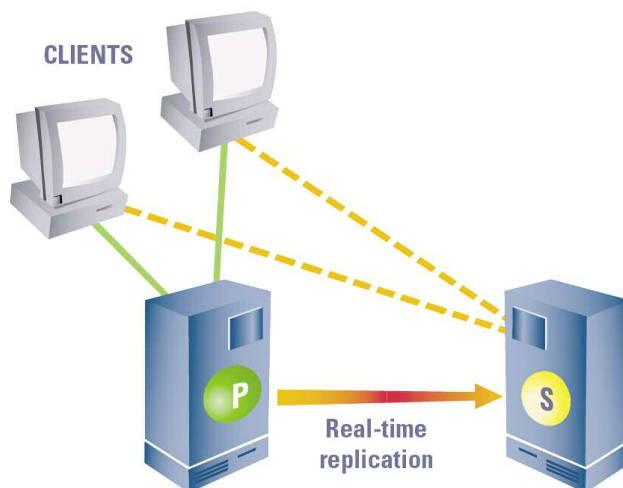


Abbildung 2. Diese Abbildung zeigt die Sonic CAA, die eine Echtzeitreplikation zwischen primären und sekundären Servern ermöglicht und das Risiko hängen gebliebener, doppelt oder in falscher Reihenfolge verschickter bzw. empfangener Nachrichten oder abgebrochener Transaktionen beseitigt.



BROKER-REPLIKATION

Das zentrale Konzept der Sonic CAA ist die Backchannel-Replikation von Brokern. Der primäre Broker stellt Nachrichtendienste für die Enterprise-Client-Applikationen bereit. Über einen Backend-Replikationskanal überträgt der primäre Broker gleichzeitig einen kontinuierlichen Datenstrom mit Statusinformationen zu Nachrichten an einen sekundären Broker. Dieser Replikationskanal wird auf einem privaten Netzwerk betrieben, der eigens für die Synchronisierung der Broker- und Nachrichtenzustandsdaten eingerichtet wurde. Über den Replikationskanal überwachen sich beide Broker gegenseitig, indem sie in regelmäßigen Abständen den Heartbeat des anderen Brokers abfragen, um Störungen beim Datenfluss oder bei der Verbindung zu ermitteln. Wenn der sekundäre Broker im Hot-Standby-Modus ist, nimmt er keine Client-Verbindungen entgegen, ist jedoch bereit, sofort in den aktiven Zustand zu wechseln, wenn der primäre Broker nicht verfügbar ist.

Bei einem Systemausfall übernimmt der sekundäre Broker die aktive Rolle. Alle Client-Applikationen werden vom primären Broker auf den jeweils konfigurierten sekundären Backup-Broker umgeschaltet. Dieser Hot-Failover-Prozess wird sofort durchgeführt und erfolgt für die Client Applikationen in transparenter Weise. In der aktiven Rolle wartet der sekundäre Broker auf eine Wiederherstellung des Replikationskanals. Diese kann durch die Wiederherstellung des primären Brokers, aber auch durch die Konfiguration eines anderen Systems als primärem Broker erfolgen. Sobald die Verbindung wieder hergestellt ist, beginnen die Broker mit einer aktiven Replikation, um die Systeme so weit zu synchronisieren, dass der primäre Broker wieder bereit ist, in den aktiven Zustand zu wechseln.

DIAGRAMME D'ETAT

Das Verhalten des Brokers wird durch den Replikationsstatus des Brokers bestimmt. Das folgende Statusdiagramm (Abbildung 3) veranschaulicht die für das fehlertolerante Broker Paar relevanten Rollen, Zustände und Ereignisse.

Beim Broker-Status wird zwischen zwei Hauptrollen unterschieden, der aktiven Rolle und der Standby-Rolle. In der aktiven Rolle stellt ein Broker Nachrichtendienste für die Enterprise-Applikations-Clients bereit. In der passiven Rolle nimmt der Broker replizierte Statusinformationen auf und ist ständig bereit, über einen Hot-Failover-Prozess in den aktiven Zustand zu wechseln.

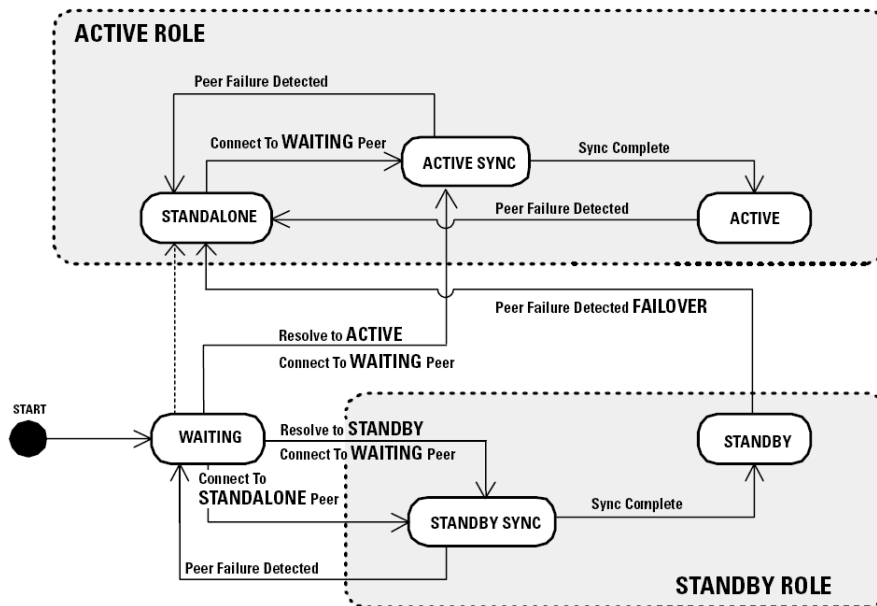


Abbildung 3. Dieses Statusdiagramm veranschaulicht die Rollen, Zustände und Ereignisse des fehlertoleranten Broker-Paars.



Nach dem Hochfahren geht ein Broker zunächst in den **WAITING**-Status und nimmt keine Client-Verbindungen entgegen. Ein Broker wechselt seinen Status, wenn eines der drei folgenden Ereignisse eintritt:

- > Ein Broker im Wartezustand kann bei administrativen Eingriffen oder bei einer entsprechenden Konfigurationsänderung direkt zum **STANDALONE**-Status übergehen.
- > Wird eine Replikationsverbindung hergestellt und befindet sich der andere Broker im **STANDALONE**-Status, wechselt der Broker in den **STANDBY SYNC**-Status und beginnt mit der laufenden Synchronisierung.
- > Wird eine Replikationsverbindung hergestellt und der andere Broker befindet sich ebenfalls im **WAITING**-Status, wählen die Broker ihre Rollen entsprechend ihrer vorherigen Rolle, ihres Synchronisationszustands und administrativ definierter Präferenzen.

Ein primärer Broker befindet sich im **STANDALONE**-Status, wenn er aktiv am Betrieb von Clients und Clustern mitwirkt, aber kein sekundärer Broker in Betrieb ist. Wenn in diesem Status eine Replikationsverbindung hergestellt wird und sich der andere Broker im **WAITING**-Status befindet, wechselt der primäre Broker in den **ACTIVE SYNC**-Status.

Im **ACTIVE SYNC**-Status synchronisiert der Broker den Zustand mit dem Standby-Broker und bedient gleichzeitig Client-Applikationen. Sobald das Runtime-Synchronisationsprotokoll durchlaufen worden ist, wechselt er in den **ACTIVE**-Status.

Nach Abschluss der Synchronisierung wechselt der primäre Broker in den **ACTIVE**-Status und beginnt damit, Nachrichtendaten und Daten zum Status von Nachrichten an den jeweiligen sekundären Broker zu übertragen, der sich momentan im **STANDBY**-Status befindet. Sollte sich zu diesem Zeitpunkt ein Ausfall des primären Brokers ereignen, wechselt der sekundäre Broker sofort in den **STANDALONE**-Status.

Wenn ein sekundärer Broker in den **STANDBY SYNC**-Status eintritt, synchronisiert er dynamisch seinen Zustand mit dem aktiven Broker, der sich im **ACTIVE SYNC**-Status befindet, und nimmt gleichzeitig den laufenden Replikationsdatenstrom auf. Nachdem das Runtime-Synchronisationsprotokoll durchlaufen worden ist, wechselt der sekundäre Server in den **STANDBY**-Status.

Sobald sich der sekundäre Broker im **STANDBY**-Status befindet, nimmt er die laufenden Nachrichten- und Nachrichtenstatusdaten des primären Brokers auf, so dass er in der Lage ist, bei einem Ausfall des primären Brokers sofort eine **STANDALONE**-Rolle zu übernehmen. Der Wechsel von **STANDBY** zu **STANDALONE** wird als Broker Failover bezeichnet.

REPLIKATIONSVERBINDUNGEN

Die Sonic CAA wurde mit einem einzigartigen Replikationsprotokoll implementiert, das eine dynamische Synchronisation und die laufende Übertragung von Status-Datenströmen zwischen einem fehlertoleranten Broker Paar ermöglicht. Dabei erlaubt die Sonic CAA die Verwendung mehrerer Replikationsverbindungen, um eine redundante Netzwerkunterstützung zwischen den primären und sekundären Brokern zu gewährleisten. Für den Transport des Replikationsdatenverkehrs wird jeweils nur eine Verbindung verwendet; beim Ausfall einer Verbindung wechselt das Broker Paar jedoch automatisch zur nächsten Replikationsverbindung, ohne dass der Replikationsprozess unterbrochen wird.

Durch mehrfache Replikationsverbindungen (siehe Abbildung 4) wird die Elastizität des fehlertoleranten Broker-Paars erhöht. Sollten alle Replikationsverbindungen ausfallen, wird von einem Ausfall des primären Brokers ausgegangen, so dass der sekundäre Broker die aktive Rolle übernimmt. Diese Redundanz ist von großer Bedeutung, da sich der primäre und der sekundäre Server in keinem Fall gleichzeitig im aktiven Zustand befinden sollten - einem Zustand, der als Dual Active Partition bezeichnet wird. Jeder Replikationsverbindung wird eine Gewichtung zugeordnet, mit der bestimmt wird, über welche Verbindung der Replikationsdatenverkehr vorrangig abgewickelt werden soll. Replikationsverbindungen mit der Gewichtung 0 werden nicht für den Replikationsdatenverkehr, sondern nur für die Übermittlung des Heartbeats verwendet, um sicherzustellen, dass Clients keinen Zugang zum anderen Broker erhalten. In der Regel wird eine Verbindung mit der Gewichtung 0 im Public Network eingerichtet, um das System vor einem Ausfall des Private Networks zu schützen, was zu einer Partition führen würde.

Falls eine Replikationsverbindung ausfällt und eine Replikationsverbindung mit einer niedrigeren Gewichtung zur Verfügung steht, wird die Replikation ohne Unterbrechung über diese Verbindung fortgesetzt. Sobald die Verbindung mit der höheren Gewichtung wieder zur Verfügung steht, wird der Datenverkehr über diese Verbindung fortgesetzt. Dieses Konzept ermöglicht Administratoren eine flexible Auswahl des Netzwerkpfads, der für den Replikationsdatenverkehr die höchste Bandbreite bietet.

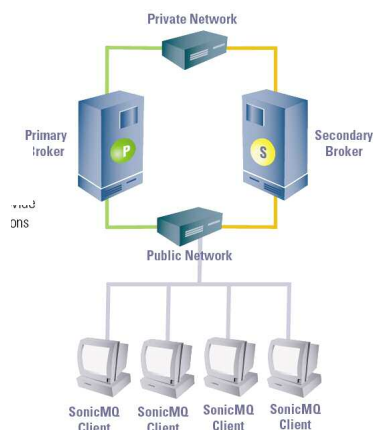


Abbildung 4. Mehrfache Replikationsverbindungen sorgen für eine Netzwerkkommunikation mit ständiger Verfügbarkeit

CLIENT-VERBINDUNGEN MIT STÄNDIGER VERFÜGBARKEIT

Die Sonic CAA stellt robuste Verbindungen für Enterprise-Client-Applikationen bereit. Bei einem Ausfall des Brokers oder des Netzwerks wird automatisch eine Standard-JMS-Verbindung abgesetzt. Diese abgesetzte Verbindung zwingt die Client-Applikation, explizit auf dieses externe Ereignis zu reagieren.

Was dabei in Wirklichkeit geschieht, ist folgendes: Wenn eine fehlertolerante Verbindung einen Ausfall des Brokers oder Netzwerks bemerkt, versucht sie sofort, die Verbindung wiederherzustellen. Dabei kann das Protokoll zur Wiederherstellung der Client-Verbindung unterschiedlich reagieren, was einerseits von der Art des Fehlers und andererseits von der Konfiguration des Brokers abhängt.

- > Bei einem vorübergehenden Ausfall des Netzwerks versucht die fehlertolerante Verbindung immer wieder, die Verbindung wiederherzustellen, bis das Netzwerk wieder in den Normalzustand zurückkehrt.
- > Wenn der Client-Applikation redundante Netzwerkpfade zum Broker zur Verfügung stehen und einer der Pfade ausfallen sollte, verwendet die fehlertolerante Verbindung einen der anderen Pfade, um die Verbindung fortzusetzen.
- > Wenn die Client-Applikation mit einem sekundären Broker verbunden ist und dieser ausfällt, versucht die fehlertolerante Verbindung immer wieder, die Verbindung zum Broker wiederherzustellen, bis dieser wieder in Betrieb ist.
- > Wenn die Client-Applikation mit einem fehlertoleranten Broker Paar verbunden ist und der primäre Broker ausfällt, stellt die fehlertolerante Verbindung sofort eine Verbindung zum sekundären Broker her.

Hat die Client-Applikation die Verbindung erfolgreich wiederhergestellt, werden sofort mehrere Status- und Synchronisationsprotokolle ausgetauscht, damit der Zustand von Client und Broker synchronisiert und der Status ungeklärter Nachrichten ermittelt wird. Während die Verbindung den Zustand von Client und Broker erfolgreich synchronisiert, arbeiten die Client-Applikationen weiter.

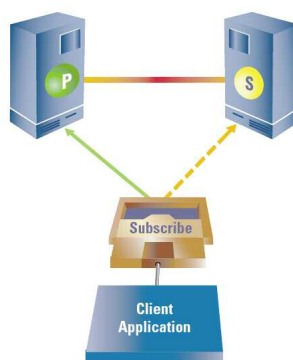


Abbildung 5. Ständig verfügbare Client-Verbindungen

NACHRICHTENZUVERLÄSSIGKEIT

Mit der Sonic CAA können verschiedene Nachrichtenzuverlässigkeitsstufen für die mit einem Message-Broker verbundenen Client-Applikationen realisiert werden. Diese Service-Qualitätsniveaus beschreiben die Anzahl von Nachrichten, die eine Client-Applikation bei einem Systemausfall erhält. Dabei können drei verschiedene Servicestufen zugrunde gelegt werden: "At Most Once" (AMO), "At Least Once" (ALO) und "Exactly Once" (EO). Festgelegt werden diese Servicestufen von den Client Applikationen. Tabelle 1 zeigt eine Zuordnung dieser Nachrichten-Servicestufen.

- > Applikationen können für die Nachrichtenübermittlung entweder Publish-Subscribe- oder Punkt-zu-Punkt-Übertragungen verwenden.
- > Nachrichten produzierende Client-Applikationen können persistent oder nicht persistent sein.
- > Nachrichten konsumierende Client-Applikationen können dauerhafte oder nicht dauerhafte Abonnements verwenden.
- > Client-Applikationen können eine Verbindung zu einem fehlertoleranten Broker-Paar herstellen.
- > Client-Applikationen können zwischen Standardverbindungen und fehlertoleranten Verbindungen wählen.

Nachrichtenzuverlässigkeit

Nachrichtenproduzent		Nachrichtenkonsument			
Verbindungsart	Zustellungsart	Standardverbindung		Fehlertolerante Verbindung	
		Topic	Topic (dauerhaftes Abonnement) oder Queue	Topic	Topic (dauerhaftes Abonnement) oder Queue
Standardverbindung	DISCARDABLE	At Most Once ¹	At Most Once ¹	At Most Once ¹	At Most Once ¹
	PERSISTENT	At Most Once ²	At Least Once ^{1,2}	At Most Once ¹	Exactly Once ¹
	NON_PERSISTENT	At Most Once ¹	At Most Once ¹	At Most Once ¹	At Most Once ¹
Fehlertolerante Verbindung	DISCARDABLE	At Most Once	At Most Once	At Most Once	At Most Once
	PERSISTENT	At Most Once ³	At Least Once ²	At Most Once	Exactly Once
	NON_PERSISTENT	At Most Once	At Most Once	At Most Once	At Most Once

Wenn sowohl die produzierenden als auch die konsumierenden Client Applikationen Standardverbindungen verwenden, werden die Nachrichten höchstens einmal ('At Most Once') empfangen, außer bei persistenten/dauerhaften Nachrichten. Bei persistenten/dauerhaften Nachrichten wird die Nachricht mindestens einmal ('At Least Once') empfangen, wobei jedoch auch die Möglichkeit besteht, dass sie doppelt empfangen wird. Deshalb kann bei einer Standardverbindung nicht garantiert werden, dass eine Nachricht genau einmal ('Exactly Once') empfangen wird.



Wenn die produzierende und die konsumierende Client-Applikation eine Kombination von Standardverbindungen und fehlertoleranten Verbindungen verwendet, wird die Nachrichtenzuverlässigkeit nur geringfügig verbessert. Dabei sind drei Sonderfälle zu unterscheiden:

- 1.** Wenn eine Standardverbindung ausfällt und der Status der letzten gesendeten Nachricht ungeklärt ist, kann die produzierende Client Applikation versuchen, die Nachricht nochmals zu senden. Falls der Broker die ursprüngliche Nachricht bereits erhalten hatte, führt dies dazu, dass eine doppelte Nachricht generiert wird. Nach der JMS Spezifikation gilt dies jedoch nicht als wiederholte Zustellung, da die Nachricht aus einer neuen Sitzung heraus zugestellt wurde. Diese Ambiguität wird durch eine Fehlertoleranz Bedingung aufgelöst: PERSISTENTE Nachrichten werden mit der Zuverlässigkeitsstufe 'Exactly Once' zugestellt, während verworfene (DISCARDABLE) und nicht persistente (NON_PERSISTENT) Nachrichten bei einem Ausfall gelöscht werden.
- 2.** Beim Ausfall einer Standardverbindung kann die Empfangsbestätigung für die letzte Nachricht verloren gehen. In diesem Fall stellt der Broker die Nachricht entsprechend der JMS-Spezifikation erneut zu.
- 3.** Wenn eine konsumierende Client-Applikation eine Standardverbindung zeitgleich mit einer fehlertolerante, produzierende Applikation wiederherstellt, besteht die Möglichkeit, dass die produzierende Applikation eine Nachricht, die dem zuvor verbundenen Client bereits zugestellt wurde, nochmals verschickt. Nach der JMS Spezifikation gilt dies jedoch nicht als wiederholte Zustellung, da die Nachricht für eine neue Sitzung zugestellt wurde.

Die Nachrichtenzuverlässigkeitsstufe 'Exactly Once' kann nur garantiert werden, wenn sowohl die produzierende als auch die konsumierende Applikation fehlertolerante Verbindungen verwendet und Nachrichten als persistent/dauerhaft verwendet werden.

ZUVERLÄSSIGKEIT VON TRANSAKTIONEN

Sonic CAA sorgt für eine zusätzliche Zuverlässigkeitsschicht im direkten/transaktionalen Messaging. Bei nicht vorhandener Fehlertoleranz muss eine Client-Applikation in der Lage sein, die Ambiguität aufzulösen, die verursacht wird, wenn sich beim Ausführen einer Transaktion ein Ausfall ereignet. Fällt der Broker oder die Verbindung aus, so ist das Resultat der Ausführung unsicher; die Applikation weiß nicht, ob die Transaktion abgeschlossen wurde oder nicht. Bei einem Client, der eine fehlertolerante Verbindung verwendet, wird diese Ambiguität beseitigt. Während der Wiederherstellung der Verbindung wird die Ambiguität aufgelöst, und die Ausführung wird erfolgreich durchgeführt. Wenn sich ein Ausfall mitten in einer Transaktion ereignet, ist die Sonic CAA von großem Vorteil für Clients, da die Transaktion nach dem Ausfall ohne Unterbrechung fortgesetzt werden kann. Bei einem nicht fehlertoleranten Client müssen alle Vorgänge, die vor dem Ausfall durchgeführt wurden, zurückgerollt werden, so dass die Transaktion von neuem gestartet werden müsste.

FLEXIBLE IMPLEMENTIERUNGSMÖGLICHKEITEN

Die Sonic CAA stellt eine hoch verfügbare und ausfallsichere Messaging Infrastruktur bereit. Um eine ständige Verfügbarkeit in großen und heterogenen Implementierungen bereitzustellen, können fehlertolerante Broker Paare auf unterschiedlichsten Hardwareplattformen konfiguriert werden. Dabei ist es nicht zwingend erforderlich, für die primären und sekundären Server identische Hardwaresysteme eingesetzt. Außerdem wird keine spezielle Managementsoftware für die Cluster- oder Hochverfügbarkeitssoftware benötigt. Durch diese Flexibilität können fehlertolerante Broker Paare so konfiguriert werden, dass die Nutzung der vorhandenen Rechnerressourcen optimiert wird. Abbildung 6 zeigt eine Konfiguration mit drei Rechnern, auf denen drei fehlertolerante Broker Paare betrieben werden, wodurch ein unternehmensweites, robustes und leistungsfähiges Messaging System geschaffen wird.

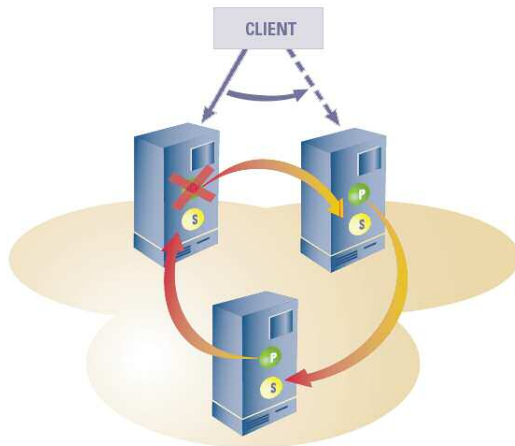


Abbildung 6. Eine effizientere Nutzung von Ressourcen und eine höhere Performance von Clustern werden erreicht, indem fehlertolerante Broker Paare auf mehrere Rechner

Durch das fortschrittliche Clustering und die Möglichkeiten der Dynamic Routing Architecture[®] von Sonic können die Hochverfügbarkeitsfunktionen auch auf das gesamte Unternehmen ausgedehnt werden, so dass auch entfernte Standorte und Geschäftspartner von einer ständigen Verfügbarkeit profitieren (vgl. Abbildung 7).

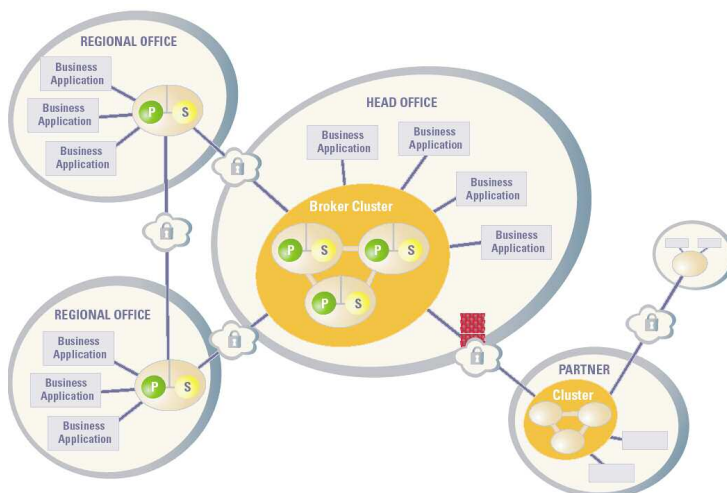


Abbildung 7. Dieses Darstellung veranschaulicht die hoch verfügbare, sichere und zuverlässige Erweiterung des unternehmensweiten Backbone-Systems auf Außenbüros und Geschäftspartner.



“Die derzeit auf dem Markt erhältlichen Hochverfügbarkeitslösungen reichen nicht aus. Wenn die Broker nach einer Störung wieder online sind, können laufende Handelstransaktionen verloren gegangen sein. Dieses Problem lässt sich mit Sonic lösen,” sagt John Brann, Chief Architect bei FXall, dem führenden Devisenhandelsportal. “Mit Sonic war die Konfiguration außergewöhnlich einfach. Bereits nach weniger als einem Tag waren wir mit unseren ersten Tests erfolgreich.”

ZUSAMMENFASSUNG

In zunehmendem Maße verlangen Unternehmen, dass ihre Systeme ununterbrochen verfügbar sind und weder durch geplante noch durch ungeplante Ausfälle beeinträchtigt werden. Um Umsatz und Gewinn durch eine unterbrechungsfreie Unterstützung von Geschäftsprozessen zu maximieren, müssen Unternehmen sowohl intern als auch bei ihren externen Verbindungen ein hohes Maß an Verfügbarkeit bereitstellen, um somit eine kontinuierliche Betreuung ihrer Geschäftspartner und Kunden sicherstellen zu können.

Um die Verfügbarkeit ihrer Messaging-Infrastruktur zu steigern, haben Unternehmen in der Vergangenheit viel Zeit und Aufwand in die Programmierung und Implementierung komplexer Lösungen investiert, die beispielsweise auf redundanten Messaging-Systemen sowie kostspieligen RAID- und SAN-Netzwerken beruhen. Aufgrund der Probleme, die ein Systemausfall mit sich bringen kann (insbesondere auf dem ausgefallenen Server hängen gebliebene Nachrichten, doppelt verschickte bzw. empfangene Nachrichten, in falscher Reihenfolge zugestellte Nachrichten sowie abgebrochene Transaktionen), wird Middleware Lösungen nur wenig Vertrauen entgegengebracht. Vor dem Hintergrund, dass ein Systemausfall von wenigen Minuten Umsatzeinbußen in Millionenhöhe, entgangene geschäftliche Gelegenheiten oder gar Geldbußen nach sich ziehen kann, liegt es auf der Hand, dass eine bessere Lösung erforderlich ist.

Progress Software definiert einen neuen Standard für hohe Verfügbarkeit und fehlertolerantes Messaging, indem nicht nur die operativen Risiken, sondern auch der mit der Entwicklung hoch verfügbarer Lösungen verbundene Zeitaufwand und die administrative Komplexität reduziert werden.

Die patentierte Continuous Availability Architecture (CAA) von Sonic wird den Problemen gerecht, die durch den Ausfall des Messaging-Systems verursacht werden, und gewährleistet, dass die Geschäftsapplikationen des Unternehmens auch bei einem Systemausfall weiter funktionieren.

Die CAA sorgt für eine hohe Verfügbarkeit auf der Messaging-Ebene, welche die Sonic Message Broker, die Sonic Clients und die Kommunikation zwischen Clients, Brokern und Zielapplikationen umfasst. Damit entfällt die Anschaffung kostspieliger RAID-Systeme, Betriebssystem-Clustersoftware oder Hochverfügbarkeitsplattformen von Drittanbietern für die Messaging-Schicht. Laufende Transaktionen werden unabhängig von ihrer jeweiligen Komplexität ohne kostspielige Rollback-oder Recovery-Maßnahmen zu ihren Zielapplikationen geführt.

Unter <http://www.sonicsoftware.com> können Sie mehr darüber erfahren, wie Sie die Verfügbarkeit Ihrer Messaging-Architektur steigern können. Darüber hinaus können Sie unter <http://www.sonicsoftware.com/developer/download> eine Evaluationsversion von Sonic-Produkten, darunter Sonic ESB sowie SonicMQ, herunterladen.



Unternehmenssitz und Zentrale für Nordamerika

Progress Software Corporation, 14 Oak Park, Bedford, MA 01730 USA, Tel: +1 781 280-4000 Fax: +1 781 280-4095

Zentrale für die Region EMEA (Europa, Mittlerer Osten, Afrika)

Progress Software Europe B.V., Schorpioenstraat 67, 3067 GG Rotterdam, Niederlande
Tel: +31 10 286-5700 Fax: +31 10 286-5777

Zentrale für Mittelamerika

8323 Northwest 12 Street, Suite 216, Miami, Florida. 33126 USA, Tel: 305-716-1007 Fax: 305-716-0133

Zentrale für den Raum Asien/Pazifik

Sydney Office, Australia Level 2, 194 Miller Street North Sydney, NSW 2060, Australien
Tel: +61-2-9919-7100 Fax: +61-2-8920-9520

© 2007 Progress Software Corporation. Alle Rechte vorbehalten. Progress, Sonic Continuous Availability Architecture, SonicMQ und Dynamic Routing Architecture sind in den USA und in anderen Ländern (eingetragene) Warenzeichen der Progress Software Corporation. Alle anderen in dieser Publikation aufgeführten Warenzeichen oder Dienstleistungsmarken sind Eigentum der jeweiligen Unternehmen.

prod code 7628



0000113860