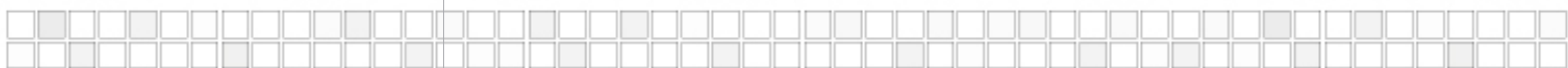




DISPONIBILITE CONTINUE POUR LA MESSAGERIE D'ENTREPRISE

Réduction du risque opérationnel et de la complexité d'administration





SOMMAIRE

Introduction	2
Nécessité d'une messagerie d'entreprise	4
JMS (Java Message Service)	5
Exigences d'une messagerie d'entreprise	6
Architectures traditionnelles à tolérance de pannes	8
Architecture matérielle à haute disponibilité	8
Architecture logicielle à haute disponibilité	9
Sonic CAA (Continuous Availability Architecture™)	11
Réplication de courtier	12
Connexions de réplication	15
Connexions client à disponibilité permanente	16
Fiabilité des messages	17
Fiabilité des transactions	18
Solutions de déploiement souples	19
Résumé	20

INTRODUCTION

L'importance accrue d'une activité fonctionnant 24h/24 x 7j/7, et le besoin croissant d'opérations en temps réel favorisent la demande de systèmes d'entreprise haute disponibilité. Ces systèmes se composent de plusieurs niveaux de serveurs, chacun d'entre eux ayant des caractéristiques de disponibilité spécifiques et représentant des points de défaillance possibles. Les départements informatiques doivent relever le défi d'optimiser la disponibilité des systèmes et d'éliminer tout point de défaillance unique.

Les entreprises exigent de plus en plus une disponibilité continue des services sans aucune immobilisation, qu'elle soit ou non planifiée. Les pannes, baisses de performances et interruptions programmées perturbent les activités de l'entreprise et affectent négativement la satisfaction des clients. Les coûts d'immobilisation se calculent par la perte de productivité engendrée tant au sein du département informatique que des entités commerciales qui en dépendent, la perte de revenu subie, ainsi que par les pénalités appliquées, telles que les sanctions administratives.

Le coût d'une immobilisation est la somme des éléments suivants:

Productivité des utilisateurs affectés = coût horaire des utilisateurs affectés x nombre d'heures d'interruption

+ Perte de productivité informatique = coût horaire des utilisateurs affectés x nombre d'heures de perte de productivité

+ Impact sur le service client et la crédibilité

+ Perte de revenu = perte de revenu par heure x nombre d'heures d'immobilisation

+ Autres pertes engendrées

Rémunération des heures supplémentaires = salaires horaires x nombre d'heures supplémentaires

+ Marchandises gaspillées

+ Sanctions ou pénalités financières

Généralement, la perte de revenu constitue la plus grosse perte financière, mais peut s'avérer la plus difficile à quantifier. Certains secteurs sont plus dépendants du fonctionnement en temps réel de leurs systèmes d'entreprise. Le tableau suivant définit ces coûts pour différents types d'activité.

Coûts moyens des immobilisations non planifiées pour l'industrie américaine¹	
Courtage de détail	6,45 millions USD par heure
Autorisation de carte de crédit	2,6 millions USD par heure
Publireportage/promotion par numéro gratuit	199.500 USD par heure
Centre de ventes sur catalogue	90.000 USD par heure
Réservations aériennes	85.000 USD par heure
Fournisseurs de services ATM	14.500 USD par heure

Par ailleurs, en cas d'immobilisation non planifiée, l'impact sur l'utilisateur est beaucoup plus important qu'une immobilisation planifiée. "Les spécialistes des facteurs humains ont remarqué qu'un temps de réponse de plus de 2 secondes à une requête utilisateur affecte la concentration du demandeur, ce qui nécessite une réinitialisation mentale avec perte de productivité mesurée en minutes. Les immobilisations système supérieures à 20 minutes environ incitent l'utilisateur à changer d'activité, ce qui perturbe les processus et entraîne des arrêts effectifs mesurés en heures du point de vue utilisateur"².

Enfin, les immobilisations système peuvent avoir un effet à long terme affectant la réputation d'une entreprise, et l'exposer ainsi à la défection des clients et à des risques juridiques, tels que les actions judiciaires ou pénalités. Pour optimiser leurs revenus et leur rentabilité via la continuité de leur activité, les entreprises doivent concevoir une architecture à haute disponibilité au sein même de la société et au-delà, afin de traiter sans interruption les demandes et besoins de leurs partenaires et clients. L'une des clés de la solution consiste à renforcer la base de données, les réseaux, les serveurs Web et les équipements physiques. Le présent document traite des problèmes associés aux approches actuelles en matière de solutions de messagerie à tolérance de panne, et présente l'architecture Sonic CAA (Sonic Continuous Availability Architecture™), un paradigme innovant et inégalé permettant aux entreprises d'améliorer leur disponibilité opérationnelle.

1. InternetWeek 4/3/2000 et "Fibre Channel: A Comprehensive Introduction", R. Kembel, 2000, p.8, basé sur une enquête par recherche de planification de contingence

2. Forrester Research: 15 mars 2004 "An Executive Guide To High Availability" par Bob Zimmerman



NECESSITE D'UNE MESSAGERIE D'ENTREPRISE

L'importance de l'infrastructure de messagerie pour la communication d'entreprise augmente au fur et à mesure que les sociétés cherchent à améliorer leur productivité opérationnelle. Même pendant l'immobilisation planifiée des systèmes, il est indispensable que les autres équipements, départements, entités commerciales et partenaires puissent continuer à travailler. En outre, en cas de panne d'un système, il est primordial qu'il n'y ait aucune perte de données.

Dans l'environnement concurrentiel actuel, un contact permanent avec vos partenaires commerciaux est indispensable. L'intégration des systèmes permet d'accélérer les délais de réponse aux demandes de devis, paiements et rapports d'état. La nécessité de connecter les systèmes à l'extérieur du pare-feu et de transférer les données stratégiques a rendu incontournable la mise en place d'un échange d'informations sécurisé, disponible et fiable.

Pour relever ces défis, la plupart des entreprises ont mis en œuvre une infrastructure informatique basée sur un système de messagerie de type RPC (Remote Procedure Call). Les systèmes RPC, tels que ceux fournis par les serveurs d'applications, se sont répandus dans les années 1990 dans leurs variantes CORBA, COM/DCOM et RMI. Malheureusement, ces systèmes sont très peu fiables, car les applications sont fortement couplées aux communications synchrones. Ils utilisent des connexions point à point, et n'ont pas d'interface commune.

Un système de messagerie de type RPC exige que les applications client soient développées selon des directives très limitées. La nature synchrone de ce système exige que l'ensemble des systèmes et applications soient disponibles chaque fois que cela s'avère nécessaire. Chaque application client exige une connaissance approfondie des API associées aux autres applications. Lorsque les applications évoluent ou que de nouvelles sont intégrées dans le système, le nombre d'interfaces nécessaires augmente très rapidement. Alors que la connexion d'un nombre limité d'applications à l'aide de cette approche peut s'avérer raisonnable, un environnement hétérogène de grande taille peut être affecté par ces interactions synchrones à couplage fort.

Dans les années 1990, les systèmes MOM (Message-Oriented Middleware) sont passés à une approche plus évolutive, impliquant une architecture asynchrone à couplage faible. Un système MOM gère des fonctions telles que la fourniture garantie des messages, et le traitement des erreurs via une interface de messagerie standardisée. Les applications pouvant communiquer les unes avec les autres de façon asynchrone, il n'est pas nécessaire que tous les systèmes s'exécutent pour assurer l'intégrité du réseau d'applications. Globalement, les systèmes MOM constituent une amélioration significative par rapport au modèle de communication RPC car ils impliquent moins de connexions réseau, offrent une souplesse accrue dans les déploiements (meilleure fiabilité), et requièrent moins de codage lors de changements de configuration.

JMS (JAVA MESSAGE SERVICE)

En 1998, Sun Microsystems a introduit JMS (Java Message Service). L'API JMS, développée par Sun en collaboration étroite avec les principaux fournisseurs de système de messagerie, combinait des éléments clés des technologies RPC et MOM. La messagerie d'entreprise est maintenant considérée comme un outil indispensable pour le développement d'applications client. Pour plus d'informations sur JMS, visitez le site <http://java.sun.com/products/jms/>. Progress® Software a développé Progress SonicMQ®, un produit de messagerie basé sur JMS et qui est devenu le système de messagerie standard le plus robuste et le plus résilient. Pour plus d'informations sur SonicMQ, visitez le site <http://www.sonicsoftware.com/products/sonicmq>.

La messagerie d'entreprise, et JMS en particulier, fournit un service souple et fiable pour l'échange asynchrone d'événements et de données stratégiques dans l'ensemble d'une société. JMS permet aux applications client de communiquer les unes avec les autres à l'aide d'un protocole de messagerie à couplage faible bien défini. L'API JMS offre une API commune et un cadre fournisseur permettant de développer des applications à base de message portables, sécurisées et fiables.

De nombreuses applications client ne prennent pas en charge les messages perdus ou dupliqués. Il est essentiel pour de nombreuses applications JMS de garantir la fourniture d'un message "une et une seule fois". Ce niveau de service est désigné par le terme "fiabilité des messages de type *exactement une fois*".

JMS définit plusieurs mécanismes de fiabilité des messages. La méthode la plus fiable pour générer un message consiste à l'envoyer en tant que message persistant au sein d'une transaction. La méthode la plus fiable pour consommer un message consiste à le recevoir provenant d'une file d'attente ou d'une souscription durable au sein d'une transaction.

EXIGENCES D'UNE MESSAGERIE D'ENTREPRISE

Dans des conditions de fonctionnement normales, JMS fournit une fiabilité de type "exactement une fois". Cependant, aucun des mécanismes JMS ne peut fournir ce type de service en cas de défaillance des équipements, du réseau ou du système d'exploitation. Il existe quatre catégories de défaillances possibles ayant un impact significatif sur les activités de l'entreprise.

1. MESSAGES PIEGES

Après une panne système, les messages sont perdus dans le système de messagerie défaillant et ne sont jamais reçus par l'application client.

2. MESSAGES EN DOUBLE

Après une panne système, l'état des messages est inconnu. Lors de la reprise, cela peut se traduire par l'envoi de messages en double par l'application client ou le courtier de messagerie (par exemple, l'envoi à trois reprises d'un ordre de débit du service financier).

3. MESSAGES DESORDONNES

Après une panne système, l'état du transit et de la fourniture des messages n'est pas fiable. Lors du redémarrage, cela peut se traduire par la réception de messages désordonnés par l'application (par exemple, la réception d'un ordre d'annulation ou de mise à jour avant celle de l'ordre initial).

4. TRANSACTIONS ALTEREES

Après une panne système, l'état des transactions est incomplet. Lors du redémarrage, les transactions altérées doivent être annulées et supprimées.

Ces pannes peuvent provoquer un retard important dans le traitement des processus opérationnels. Dans tous les cas évoqués, une "équipe de secouristes" est nécessaire pour intervenir et reconstruire l'état des messages. Cela implique l'annulation des transactions, la récupération des messages piégés, l'identification et la suppression des messages en double, ainsi que la reconstruction des messages désordonnés. Si cette reconstruction n'est pas effectuée, les applications client d'entreprise des flux de messages altérés se traduisant par des transactions incomplètes ou inexacts.



Pour qu'un système de messagerie d'entreprise fournisse une fiabilité de type "exactement une fois" en cas de défaillance matérielle, réseau ou système, il doit fournir une redondance suffisante. Cette redondance est assurée via une paire de courtiers principal et secondaire (secours), qui doit répondre aux exigences suivantes en terme de tolérance de panne:

1. RECUPERATION DE L'ETAT COMPLET:

En cas de défaillance d'un système, le courtier secondaire doit être capable d'assumer le rôle de son partenaire principal défaillant. Le courtier secondaire doit récupérer l'état complet des messages avant l'incident.

2. REPRISE AUTOMATIQUE:

En cas de défaillance d'un système, le courtier secondaire doit passer à l'état actif dans un délai minimal. Un tel délai minimal garantit que la mémoire tampon des applications client ne sera pas saturée ou que ces dernières ne se bloqueront pas.

3. REPRISE TRANSPARENTE:

En cas de défaillance d'un système, les applications doivent passer de manière transparente sur le courtier de messagerie secondaire. La connexion des applications au système de messagerie reste active tant que la transition sur le courtier secondaire n'est pas terminée.



ARCHITECTURES TRADITIONNELLES A TOLERANCE DE PANNES

Lorsqu'un message transite d'un client vers une destination, il peut traverser de nombreux réseaux, systèmes et applications. Des niveaux supérieurs de disponibilité peuvent être atteints en renforçant divers composants, tels que les clients, middleware et bases de données. La tolérance de panne pour l'infrastructure de messagerie a traditionnellement été basée sur l'existence de plusieurs serveurs à couplage fort et gérés de façon centralisée. Les serveurs sont configurés pour fournir toute application client un serveur secondaire en cas de défaillance des systèmes. Ce processus est désigné par le terme de "reprise" (ou "failover") d'un serveur sur un autre.

Un système à tolérance de panne et disponibilité élevée est accessible aux utilisateurs et applications sous la forme d'un environnement unique. Outre la disponibilité accrue des applications, la technologie de clustering permet également d'augmenter la capacité des systèmes et d'améliorer l'efficacité d'administration. Les solutions HA (High Availability) sont disponibles depuis les années 1980, où elles étaient utilisées dans les systèmes VMS de DEC. Le sysplex d'IBM est une approche HA destinée à un système mainframe. Microsoft, Sun Microsystems, ainsi que d'autres éditeurs de logiciels et constructeurs de matériel de pointe proposent des packages HA réputés pour leur évolutivité et leur disponibilité. A mesure que le trafic et la garantie de disponibilité augmentent, tout ou partie du système peut être augmentée en taille ou en nombre.

Les architectures à haute disponibilité garantissent que la panne d'un système d'exploitation ne provoquera pas d'immobilisation de longue durée des applications, et fournissent un environnement prenant en charge la maintenance en ligne et la mise à niveau de chaque système informatique composant l'infrastructure. Une disponibilité élevée peut être implémentée via des solutions matérielles et logicielles.

ARCHITECTURE MATERIELLE A HAUTE DISPONIBILITE

La haute disponibilité matérielle (évolution verticale) est une méthode basée sur le matériel dans laquelle un groupe de serveurs se comporte comme un système unique. Dans la pratique courante, cette architecture est créée en installant un certain nombre de serveurs en lame sur la machine qui contrôlera le système. Chacune des lames fonctionne indépendamment des autres, même si elles répondent tous aux mêmes requêtes. Le système d'exploitation du serveur de contrôle est chargé de surveiller le système et d'exécuter des tâches d'administration, comme, par exemple, déterminer à quel moment le failover est nécessaire et affecter la charge d'un nœud défaillant vers un serveur opérationnel.

Les architectures à haute disponibilité matérielle peuvent être actives-passives, auquel cas les serveurs redondants sont réservés pour les tâches de reprise et n'exécutent pas eux-mêmes d'application. Elles peuvent également être actives-actives, auquel cas tous les serveurs exécutent eux-mêmes leurs propres applications, mais réservent également des ressources afin de pouvoir exécuter les tâches de reprise les uns pour les autres. La tolérance de panne matérielle impliquant l'achat et la configuration de matériel spécialisé coûteux, elle est essentiellement utilisée dans les systèmes d'entreprise haut de gamme.

ARCHITECTURE LOGICIELLE A HAUTE DISPONIBILITE

La haute disponibilité logicielle (évolution horizontale) est une méthode qui consiste à transformer plusieurs serveurs en une solution à tolérance de panne. Le logiciel de clustering du système d'exploitation est installé sur chacun des serveurs du système, et les serveurs principal et secondaire ont généralement leurs applications mises en miroir. Chaque serveur gère les mêmes informations et ils exécutent collectivement des tâches d'administration telles que l'équilibrage de charge, la détermination des défaillances de nœud, et l'affectation des tâches de reprise.

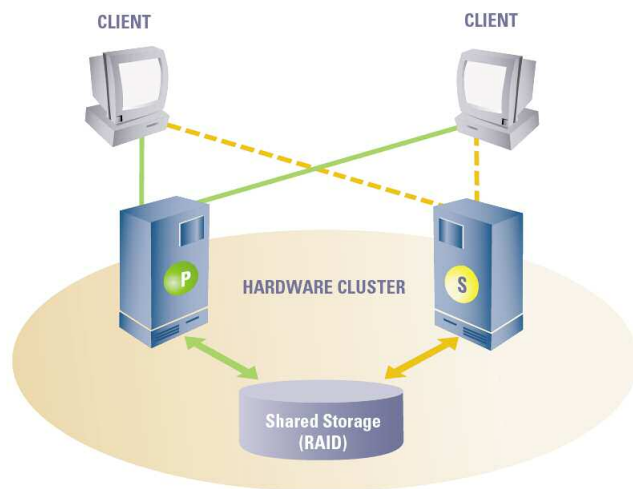


Figure 1: Ce diagramme présente une solution à tolérance de panne standard. En cas de défaillance du serveur principal, le serveur secondaire (secours) démarre et commence le processus de récupération à partir du système de stockage partagé. Des problèmes de messagerie peuvent survenir: messages piégés sur le serveur défaillant, messages en double envoyés et reçus, messages désordonnés et transactions altérées

Les serveurs pouvant être facilement ajoutés ou supprimés du cluster en fonction des besoins, la tolérance de panne logicielle est une solution évolutive. Toutefois, l'architecture à tolérance de panne logicielle standard (Figure 1) pour la messagerie d'entreprise exige l'utilisation d'une base de données partagée afin de garantir des informations actualisées et complètes sur l'état des messages. Deux serveurs de messagerie (principal et secondaire) désignent cette base de données à un emplacement accessible au réseau. En l'absence de défaillance, seul le serveur principal dispose d'un accès en lecture/écriture. Un système de base de données sur modules RAID permet de garantir que la base ne présente pas de point de défaillance unique.



En cas de défaillance du serveur principal, le système tiers initie un processus de reprise sur le serveur secondaire. Le serveur secondaire commence à lire les informations sur l'état des messages à partir de la base de données partagée, et initialise des fichiers et journaux internes. Ce processus de reprise peut prendre plusieurs minutes, voire davantage. Le système à tolérance de panne notifie également les applications d'entreprise de la défaillance et fournit des informations de reconnexion. Les applications client se connectent au serveur secondaire et exécutent le processus de récupération.

Malgré sa complexité, le processus de reprise et de réinitialisation ne fournit toujours pas de fiabilité des messages de type "exactement une fois". Les applications client peuvent encore rencontrer des messages piégés, en double, désordonnés et des transactions altérées. Ces quatre catégories de défaillance peuvent toutes affecter le système opérationnel.

SONIC CAA (CONTINUOUS AVAILABILITY ARCHITECTURE™)

Progress Software a développé Sonic CAA (Continuous Availability Architecture), une architecture de messagerie innovante et unique dont les brevets sont en cours d'homologation. Supérieure aux solutions HA traditionnelles proposées par les fournisseurs de système de messagerie actuels, Sonic CAA offre aux utilisateurs des avantages inégalés en termes de disponibilité opérationnelle et de réduction des coûts de développement, de déploiement et d'administration.

L'architecture Sonic CAA fournit une disponibilité élevée à la couche messagerie, et notamment aux courtiers de messagerie Sonic, clients Sonic et communications entre les clients, courtiers et destinations, en garantissant une fiabilité de type "exactement une fois", aussi bien en conditions de fonctionnement normales qu'en cas de défaillance. Sonic CAA élimine la nécessité de modules RAID coûteux, logiciels de clustering OS ou cadres HA tiers dans la couche messagerie. Quelle que soit leur complexité, les transactions en cours de traitement continuent jusqu'à leurs destinations sans aucun temps coûteux de rollback ou de récupération.

L'architecture Sonic CAA (Figure 2) est basée sur la réplication des courtiers primaire/secondaire via la synchronisation "backchannel" et les connexions à tolérance de panne. Cette architecture prend en charge l'entreprise étendue d'une façon jusqu'à présent impossible, et complète totalement les solutions HA actuellement disponibles sur le marché, telles que celles pour base de données, serveur d'applications et serveur Web.

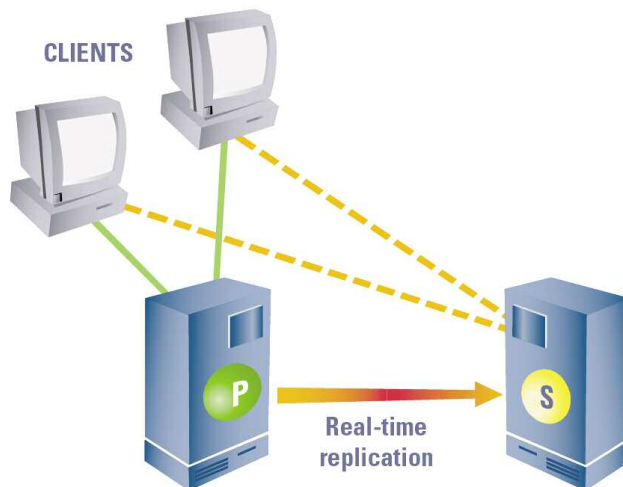


Figure 2: Ce diagramme présente l'architecture Sonic CAA fournissant une réplication en temps réel entre les serveurs principal et secondaire, sans risque de messages piégés, en double, désordonnés ou de transactions altérées



REPLICATION DE COURTIER

Le concept central de l'architecture Sonic CAA est la réplique de courtier "backchannel". Le courtier principal fournit des services de messagerie aux applications client. Parallèlement, il utilise un canal de réplique backend pour transférer l'état de la messagerie vers un courtier secondaire. Ce canal de réplique est pris en charge sur un réseau privé dédié à la synchronisation des données de messagerie et d'état des courtiers. La paire de courtiers principal/secondaire utilise le canal de réplique pour rechercher régulièrement le "heartbeat" (signal de présence) de l'autre et identifier les interruptions dans la connexion ou le flux de données. Le courtier secondaire n'accepte pas de connexion client pendant son rôle de secours automatique, mais est préparé à assumer immédiatement le rôle actif en cas d'indisponibilité du courtier principal.

En cas de défaillance d'un système, le courtier secondaire assume le rôle actif. Toutes les applications client se reconnectent au courtier de secours secondaire. Ce processus de reprise automatique est immédiat et transparent pour les applications client. Dans le rôle actif, le courtier secondaire est sensible au rétablissement du canal de réplique. Cette reconnexion peut résulter d'une récupération ou du remplacement du courtier principal. Une fois reconnectés, les courtiers se répliquent activement à des fins de synchronisation, jusqu'à ce que le courtier principal soit à nouveau prêt à reprendre l'état actif.

DIAGRAMME D'ETAT

Le comportement du courtier est déterminé par son état de réplication. Un diagramme d'état (Figure 3) permet d'illustrer les rôles, états et événements associés à la paire de courtiers à tolérance de panne. Les états des courtiers sont regroupés en deux rôles principaux, le rôle actif et le rôle de secours. Dans le rôle actif, un courtier fournit des services de messagerie aux applications client. Dans le rôle de secours, le courtier absorbe les informations d'état répliquées et est continuellement préparé pour une reprise automatique.

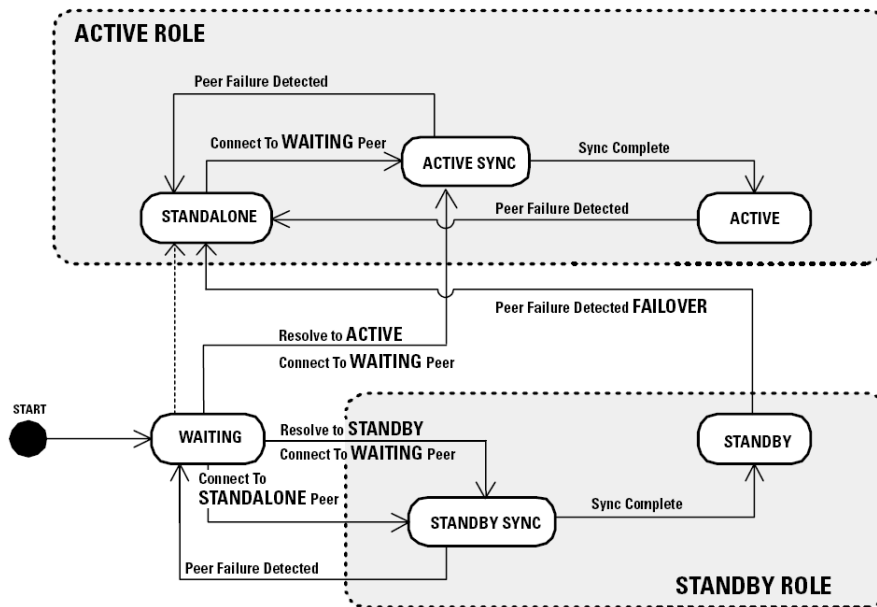


Figure 3: Diagramme d'état présentant les rôles, état et événements associés à la paire de courtiers à tolérance de panne



Lors du démarrage, un courtier passe à l'état **WAITING** (attente) et n'accepte pas les connexions client. Il changera d'état si l'un des événements suivant se produit:

- > Un courtier à l'état d'attente peut passer directement à l'état **STANDALONE** (autonome) en réponse à une intervention administrative, ou s'il est configuré pour le faire.
- > Si une connexion de réplication est établie, et que l'autre courtier est à l'état **STANDALONE** (autonome), le courtier passera à l'état **STANDBY SYNC** (synchronisation de secours) et commencera la synchronisation d'exécution.
- > Si une connexion de réplication est établie, et que l'autre courtier est également à l'état **WAITING** (attente), les courtiers choisissent des rôles en fonction de leur rôle précédent, de l'état de synchronisation et des préférences définies administrativement.

Un courtier principal est à l'état **STANDALONE** (autonome) s'il traite activement des opérations de client et de cluster mais qu'aucun courtier secondaire ne s'exécute. Lorsqu'il est à cet état, si une connexion de réplication est établie et que l'autre courtier est à l'état **WAITING** (attente), le courtier principal passe à l'état **ACTIVE SYNC** (synchronisation active).

A l'état **ACTIVE SYNC** (synchronisation active), le courtier synchronise l'état avec le courtier de secours tout en servant les applications client. L'exécution du protocole de synchronisation le fait passer à l'état **ACTIVE** (actif).

Après avoir exécuté la synchronisation, le courtier principal passe à l'état **ACTIVE** (actif) et commence activement le transfert des données et de l'état de messagerie vers le courtier secondaire correspondant qui est à l'état **STANDBY** (secours). En cas de défaillance du courtier principal à ce stade, le courtier secondaire passe immédiatement à l'état **STANDALONE** (autonome).

Lorsqu'un courtier secondaire passe à l'état **STANDBY SYNC** (synchronisation de secours), il synchronise dynamiquement son état avec le courtier actif qui est à l'état **ACTIVE SYNC** (synchronisation active) tout en absorbant le flux de réplication principal. L'exécution du protocole de synchronisation le fait passer à l'état **STANDBY** (secours). Une fois dans cet état, le courtier secondaire absorbe les données et l'état de messagerie du courtier principal qui lui permet d'assumer immédiatement un rôle **STANDALONE** (autonome) en cas de défaillance de celui-ci. La transition de l'état **STANDBY** (secours) à l'état **STANDALONE** (autonome) est désignée par le terme de "reprise du courtier".

CONNEXIONS DE REPLICATION

L'architecture Sonic CAA a été mise en œuvre à l'aide d'un protocole de réplication unique prenant en charge la synchronisation dynamique et le flux d'états entre une paire de courtiers à tolérance de panne. Sonic CAA prend en charge et encourage l'utilisation de plusieurs connexions de réplication afin de fournir un support réseau redondant entre les courtiers principal et secondaire. Une seule connexion est utilisée pour gérer à tout moment le trafic de réplication, mais en cas d'échec d'une connexion, la paire de courtiers bascule automatiquement sur la connexion de réplication suivante sans interrompre le processus de réplication.

Des connexions de réplication multiples (Figure 4) augmentent la résilience de la paire de courtiers à tolérance de panne. En cas d'échec de l'ensemble des connexions de réplication, il est supposé que le courtier principal est défaillant et que le courtier secondaire passera à un rôle actif. Cette redondance est importante car en aucun cas les courtiers principal et secondaire ne doivent passer simultanément à l'état actif, condition désignée par le terme de *partition active double*.

Chaque connexion de réplication est affectée d'un coefficient de pondération permettant de déterminer sa priorité dans le traitement du trafic. Les connexions de réplication affectées d'un coefficient de pondération de 0 ne traiteront pas le trafic de réplication et seront uniquement utilisées pour le "heartbeating" afin de confirmer que les clients ne peuvent pas non plus accéder à l'autre courtier. Généralement, un coefficient de pondération de 0 est créé sur le réseau public afin de se protéger contre une défaillance du réseau privé provoquant une partition. Si une connexion de réplication échoue et qu'une connexion de réplication affectée d'un coefficient de pondération moindre est disponible, la réplication se poursuit sans interruption sur cette connexion. Si la connexion affectée d'un coefficient de pondération supérieur est rétablie, le trafic de réplication reprend sur celle-ci, permettant ainsi aux administrateurs de choisir le chemin de réseau offrant la largeur de bande la plus élevée pour le trafic de réplication.

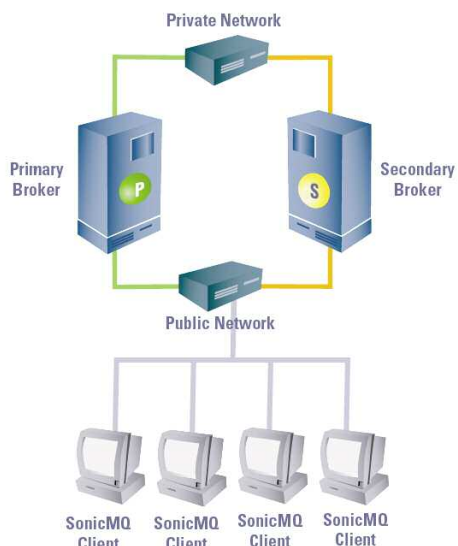


Figure 4: Des connexions de réplication multiples assurent la disponibilité continue des communications réseau

CONNEXIONS CLIENT A DISPONIBILITE PERMANENTE

L'architecture Sonic CAA (Continuous Availability Architecture) fournit des connexions résilientes aux applications client. Une connexion JMS standard est immédiatement perdue en cas de défaillance du courtier ou du réseau. La connexion perdue oblige l'application client à gérer explicitement cet événement externe.

Une connexion à tolérance de panne tentera de se reconnecter immédiatement en cas de défaillance d'un courtier ou d'un réseau. Le protocole de reconnexion client répond de différentes manières selon la configuration du courtier et la nature de la défaillance.

- > En cas de panne transitoire du réseau, la connexion à tolérance de panne tente à plusieurs reprises de rétablir la connexion jusqu'à ce que le réseau revienne à l'état normal.
- > Si l'application client dispose de chemins réseau redondants vers le courtier et que l'un d'entre eux est défaillant, la connexion à tolérance de panne utilise les autres pour reprendre la connexion.
- > Si l'application client est connectée à un courtier secondaire défaillant, la connexion à tolérance de panne tente à plusieurs reprises de se reconnecter au courtier jusqu'à sa récupération et son redémarrage.
- > Si l'application client est connectée à une paire de courtiers à tolérance de panne et que le courtier principal est défaillant, la connexion à tolérance de panne se connecte immédiatement au courtier secondaire.

Lorsque l'application client parvient à se reconnecter, elle exécute immédiatement plusieurs échanges de protocole de synchronisation et d'état, ce qui lui permet de resynchroniser l'état des clients et des courtiers, et de résoudre les messages douteux. Lorsque la connexion parvient à resynchroniser l'état des clients et des courtiers, les applications client poursuivent leurs opérations.

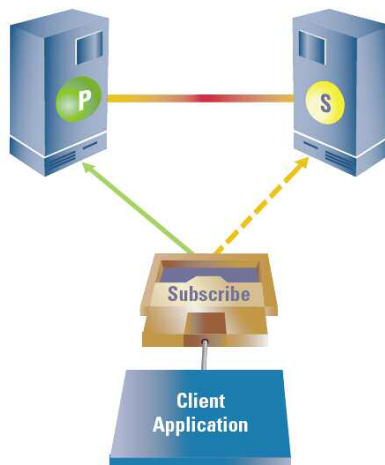


Figure 5: Connexions client disponibles en continu

FIABILITE DES MESSAGES


L'architecture Sonic CAA offre différents niveaux de fiabilité aux applications client connectées à un courtier de message. Ces qualités de service décrivent le nombre de messages qu'une application client recevra en cas de défaillance d'un système. Ces trois niveaux de service sont les suivants: "au plus une fois" (AMO), "au moins une fois once" (ALO) et "exactement une fois" (EO). Ils sont contrôlés par les applications client. Le Tableau 1 fournit une correspondance de ces niveaux de service.

- > Les applications peuvent choisir d'utiliser une messagerie de type publier-souscrire ou point à point.
- > Les applications client produisant des messages peuvent être persistantes ou non persistantes.
- > Les applications client consommant des messages peuvent sélectionner des souscriptions durables ou non durables.
- > Les applications client peuvent se connecter à une paire de courtiers à tolérance de panne.
- > Les applications client peuvent sélectionner des connexions standards ou à tolérance de panne.

Fiabilité des messages

Producteur du message		Consommateur du message			
Type de connexion	Mode de livraison	Connexion standard		Connexion à tolérance de panne	
		Sujet	Sujet (souscription durable) ou File	Sujet	Sujet (souscription durable) ou File
Connexion standard	DISCARDABLE	Au plus une fois ¹	Au plus une fois ¹	Au plus une fois ¹	Au plus une fois ¹
	PERSISTENT	Au plus une fois ²	Au moins une fois ^{1,2}	Au plus une fois ¹	Exactement une fois ¹
	NON_PERSISTENT	Au plus une fois ¹	Au plus une fois ¹	Au plus une fois ¹	Au plus une fois ¹
Connexion à tolérance de panne	DISCARDABLE	Au plus une fois	Au plus une fois	Au plus une fois	Au plus une fois
	PERSISTENT	Au plus une fois ³	Au plus une fois ²	Au plus une fois	Exactement une fois
	NON_PERSISTENT	Au plus une fois	Au plus une fois	Au plus une fois	Au plus une fois

Ce tableau démontre que lorsque les applications productrice et consommatrice utilisent des connexions standards, les messages sont reçus au plus une fois, excepté pour les messages persistants/durables. Dans le cas des messages persistants/durables, le message est reçu au moins une fois, mais avec duplication possible. Par conséquent, il n'est pas garanti que le message sera reçu exactement une fois lors de l'utilisation de la connexion standard. Si les applications productrice et consommatrice utilisent une combinaison de connexions standards et à tolérance de panne, la qualité de fiabilité des messages ne s'améliore que légèrement. Il existe trois cas spécifiques:

- 
-
1. En cas de panne d'une connexion standard, si le dernier message envoyé était douteux, l'application productrice peut tenter de le renvoyer. Cela entraîne la génération d'un message en double si le courtier a reçu le message initial. Conformément à la spécification JMS, il n'y a pas de nouvelle fourniture du message car il été envoyé à partir d'une nouvelle session. Cette ambiguïté est résolue à l'aide d'une condition à tolérance de panne: les messages PERSISTENT sont de type "exactement une fois"; les messages DISCARDABLE et NON_PERSISTENT sont perdus en cas d'échec de connexion.
 2. En cas d'échec d'une connexion standard, l'accusé de réception associé au dernier message peut être perdu. Dans ce cas, le courtier fournit à nouveau le message conformément à la spécification JMS.
 3. Si une application consommatrice se reconnecte à l'aide d'une connexion standard en même temps qu'un producteur à tolérance de panne se reconnecte, il peut arriver que le producteur renvoie un message déjà fourni au client précédemment connecté. Conformément à la spécification JMS, il n'y a pas de nouvelle fourniture du message car il été envoyé vers une nouvelle session.

La fourniture des messages "exactement une fois" est uniquement garantie si le producteur et le consommateur utilisent des connexions à tolérance de panne et que les messages envoyés sont persistants/durables.

FIABILITE DES TRANSACTIONS

L'architecture Sonic CAA ajoute également une couche supplémentaire de fiabilité à la messagerie transactionnelle. En l'absence de tolérance de panne, une application client doit être capable de gérer l'ambiguïté associée à une défaillance lors d'une validation de transaction. En cas de panne de la connexion ou du courtier, le résultat de l'opération de validation n'est pas fiable, car l'application n'identifie pas clairement si la transaction a été ou non validée.

Cette ambiguïté est levée pour un client qui utilise une connexion à tolérance de panne. Lors de la reconnexion, cette ambiguïté est résolue de manière transparente et la validation réussit. Sonic CAA présente également un grand avantage pour les clients en cas de défaillance dans le cadre d'une transaction, car la transaction peut se poursuivre sans interruption. Pour un client sans tolérance de panne, toutes les tâches effectuées jusqu'au point de défaillance sont annulées, et la transaction doit être redémarrée.

SOLUTIONS DE DEPLOIEMENT SOUPLES

L'architecture Sonic CAA (Continuous Availability Architecture) confère une disponibilité et une résilience élevée à l'infrastructure de messagerie. Pour fournir une disponibilité continue dans les déploiements divers et à grande échelle, des paires de courtiers à tolérance de panne peuvent être configurées sur des plates-formes matérielles hétérogènes. Il n'est pas nécessaire de disposer d'un matériel identique pour les serveurs principal et secondaire. En outre, aucun logiciel particulier de gestion haute disponibilité ou de clustering n'est requis. Cette souplesse permet de configurer les paires de courtiers à tolérance de panne de manière à augmenter l'efficacité d'utilisation des ressources informatiques. La configuration à trois machines présentée dans la Figure 6 héberge trois paires de courtiers à tolérance de panne fournissant un système de messagerie robuste et puissant.

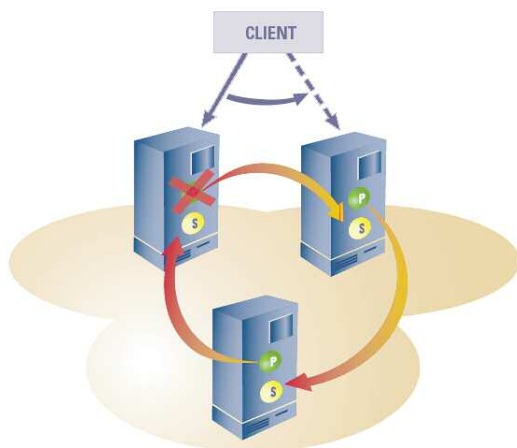


Figure 6: La distribution des paires de courtiers à tolérance de panne sur l'ensemble des machines permet d'utiliser plus efficacement les ressources et d'obtenir des performances de cluster supérieures

Par ailleurs, grâce aux fonctionnalités avancées de l'architecture DRA (Dynamic Routing Architecture[®]) et de clustering de Sonic, la haute disponibilité peut être étendue à l'ensemble de l'entreprise en fournissant une disponibilité continue aux sites distants et aux partenaires commerciaux. (Figure 7)

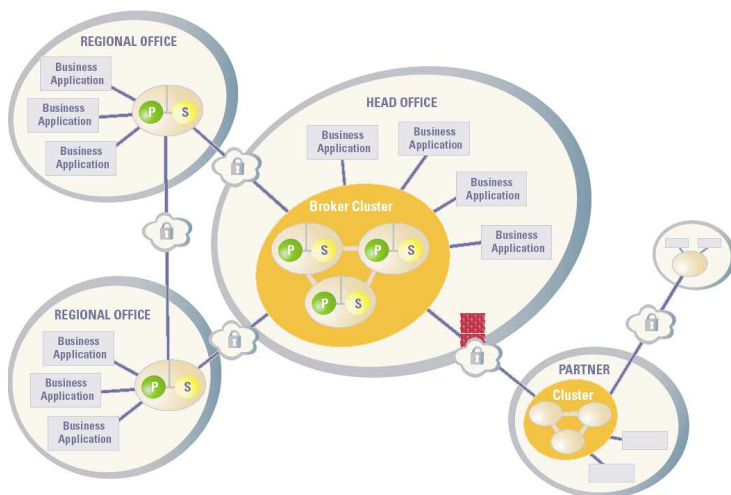


Figure 7: Ce diagramme présente l'extension fiable, sécurisée et hautement disponible du backbone de messagerie aux sites distants et partenaires commerciaux



RESUME

“Les autres solutions haute disponibilité offrent des performances inférieures car même lorsque les courtiers reviennent en ligne, des transactions en vol peuvent être perdues. Sonic résout ce problème” souligne John Brann, architecte en chef chez FXall, le premier portail de trading boursier. “La configuration a été exceptionnellement facile avec Sonic. Nos avons réussi à exécuter nos premiers tests en moins d'une journée”.

Les entreprises exigent de plus en plus une disponibilité continue des services sans aucune immobilisation, qu'elle soit ou non planifiée. Pour optimiser leurs revenus et leur rentabilité via la continuité de leur activité, les entreprises doivent concevoir une architecture à haute disponibilité au sein même de la société et au-delà, afin de traiter sans interruption les demandes et besoins de leurs partenaires et clients.

Pour améliorer la disponibilité de leur infrastructure de messagerie, les entreprises ont investi beaucoup de temps et de ressources en codant et en assemblant des solutions élaborées, telles que les systèmes de messagerie, ou en utilisant des réseaux RAID et SAN coûteux. Un manque de confiance touche les middleware en raison des problèmes causés par les défaillances des systèmes, et en particulier, les messages piégés sur le serveur défaillant, les messages envoyés et reçus en double, les messages désordonnés et les transactions altérées. Lorsque quelques minutes d'immobilisation représentent plusieurs millions en perte de revenus, opportunités manquées ou sanctions administratives, il est évident qu'une solution plus appropriée est nécessaire.

Progress Software repousse les limites de la messagerie à tolérance de panne et à haute disponibilité, en réduisant le risque opérationnel ainsi que les délais de développement et la complexité d'administration des solutions HA. L'architecture Sonic CAA (Continuous Availability Architecture) en cours d'attribution de brevet, résout les problèmes dus à la défaillance des systèmes de messagerie et permet donc aux applications d'entreprise de continuer à s'exécuter. Elle fournit une disponibilité élevée aux courtiers de message Sonic, clients Sonic et communications entre les clients, courtiers et destinations, en éliminant ainsi la nécessité de modules RAID coûteux, logiciels de clustering OS, ou cadres HA tiers dans la couche messagerie. Quelle que soit leur complexité, les transactions en cours de traitement continuent jusqu'à leurs destinations sans aucun temps coûteux de rollback ou de récupération.

Pour plus d'informations sur l'amélioration de la disponibilité de votre architecture de messagerie, visitez le site <http://www.sonicsoftware.com>. Pour télécharger une copie d'évaluation des produits Sonic, et notamment de Sonic ESB ou SonicMQ, visitez le site <http://www.sonicsoftware.com/developer/download>.



Siège social mondial et Amérique du Nord

Progress Software Corporation 14 Oak Park, Bedford, MA 01730,
Etats-Unis Tél: +1 781 280 -4000 – Fax: +1 781 280-4095

Siège social EMEA

Progress Software Europe B.V. Schorpioenstraat 67, 3067 GG Rotterdam, Pays-Bas
Tél: +31 10 286-5700 – Fax: +31 10 286-5777

Progress Software en France

Progress Software France, Tour VISTA - 52 quai de Dion Bouton – 92800 Puteaux, France
Tél: +33 1 41 16 16 00

Agence de Lyon: Progress Software Lyon, Tour Crédit Lyonnais – 129 rue Servient 69326 Lyon cedex 03
Tél: +33 4 78 63 61 65

Siège social monde

Progress Software Corporation. 14 Oak Park Drive, Bedford, MA 01730, Etats-Unis
Tél: +1 781 280 4000 – Fax: +1 781 280 4095

Siège social EMEA

Progress Software Europe B.V., Schorpioenstraat 67, 3067 GG Rotterdam, Pays-Bas
Tél: +31 10 2865700 – Fax: +31 10 2865777

© 2007 Progress Software Corporation. Tous droits réservés. Progress, Sonic Continuous Availability Architecture, SonicMQ et Dynamic Routing Architecture sont des marques commerciales ou déposées de Progress Software Corporation aux Etats-Unis et dans les autres pays. Toutes les autres marques commerciales citées dans le présent document appartiennent à leurs propriétaires respectifs. Ce document n'engage pas la responsabilité de Progress Software et peut être soumis à modification sans information préalable.

prod code 7627



0000113859