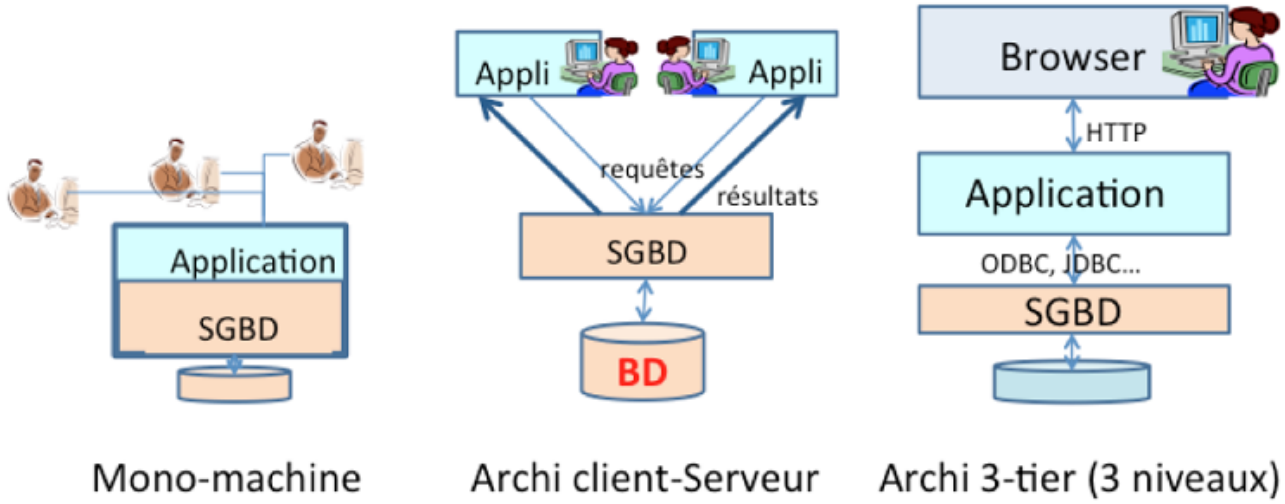
Cloud computing

Table des matières

Le cloud computing est l'exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement internet. Ces serveurs sont loués à la demande, le plus souvent par tranche d'utilisation selon des critères techniques (puissance, bande passante, etc.) mais également au forfait. Il s'agit donc d'une délocalisation de l'infrastructure informatique.

1. Les données dans le Cloud et les Data centers

1.1. architecture client-serveur



Il est intéressant de remarquer que la gestion de données dans un système de gestion de base de données (SGBD) n'a cessé de se compliquer. On pourra suivre cette évolution sur la figure ci-dessus. Considérons par exemple une application Toque qui utilise les données sur des recommandations de restaurants. Au départ, les données (les recommandations) et l'application (le code de *Toque*) sont sur le même ordinateur (architecture mono-machine). Dans les années 70-80, on est passé à une architecture client-serveur. On a placé le serveur (le SGBD et ses données) sur une machine dédiée, dotée des ressources adaptées, et les clients (les applications qui accèdent à ces données) sur d'autres, souvent des postes (les fameux micro-ordinateurs) remplaçant les terminaux passifs. Cette architecture a encore évolué à partir des années 90 par l'ajout d'un troisième niveau (« 3-tier ») en séparant la couche applicative et l'interface utilisateur.

Les raisons principales de cette complexité croissante tiennent essentiellement à l'augmentation des ressources nécessaires pour chaque composant :

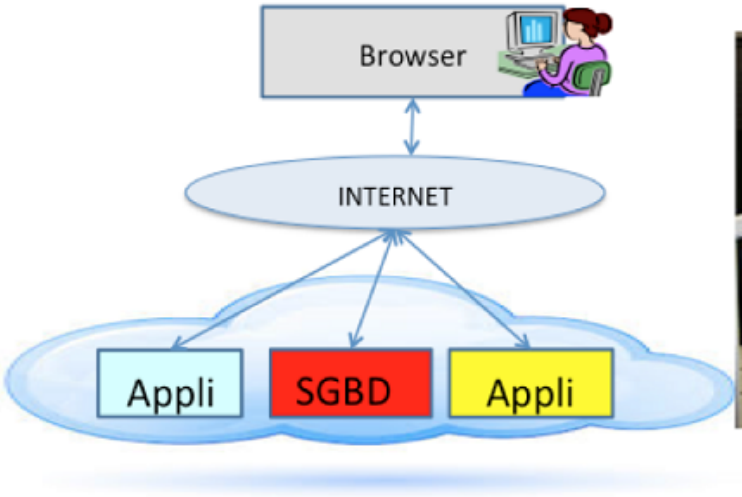
- Les serveurs de données sont confrontés à des volumes de données sans cesse croissants, et doivent communiquer avec un nombre d'applications clientes elles-mêmes en constante progression. On parlait en Mégaoctets dans les années 80, en Téraoctets de nos jours, soit une croissance de l'ordre d'un million. Un serveur de données dans une grande entreprise doit satisfaire des milliers de requêtes simultanées. Ces tendances (et quelques autres, comme la sécurité) ont dicté l'affectation de ressources de calcul importantes aux SGBD.
- Les applications elles-mêmes ont grossi et se sont complexifiées. Notre application Toque, initialement écrite en Fortran et effectuant quelques opérations simples, est devenue une plate-forme Java complexe, utilisant des bibliothèques extérieures, fournissant des services Web, mettant en jeu un paramétrage lourd. L'installation et la maintenance d'une telle plate-forme sur les postes clients sont devenues des tâches très complexes : on préfère ne laisser au poste client qu'une couche logicielle mince dédiée à l'interface utilisateur.

Nous sommes en 2015, l'application Toque est devenue une référence avec une base de données ambitionnant de couvrir tous les restaurants du monde, des logiciels internes d'analyse et de recommandation gourmands en ressources, une interface sur le Web et des applications mobiles. De

plus, le nombre d'utilisateurs de l'application varie assez fortement dans le temps, avec un pic de fréquentation le samedi en fin d'après-midi. Les dirigeants se posent la question : comment faire face à l'augmentation des ressources matérielles nécessaires, à leur maintenance, à leur renouvellement, tout en ne surdimensionnant pas leur système. Le cloud offre une solution.

1.2. Le cloud computing

Le cloud computing, ou l'informatique en nuage, est l'exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement l'internet. Ces serveurs sont loués à la demande, le plus souvent par tranche d'utilisation selon des critères techniques (puissance, bande passante, etc.) mais également au forfait. Il s'agit donc d'une délocalisation de l'infrastructure informatique. [A partir de Wikipédia, Cloud computing, 2015]



SGBD et Applications dans les nuages Data center

Dans le cas de la gestion de données, on va déporter le SGBD et l'application dans le cloud, c'est-à-dire dans des fermes de serveurs connectées à l'Internet (figure ci-dessus). Les données passent sur des machines, dont la localisation quelque part dans le monde n'a a priori pas d'importance, et idem pour les applications. Ce qui a permis de le faire c'est :

1. l'augmentation des vitesses et capacité des réseaux informatiques, et
2. la disponibilité d'immenses ressources interconnectées de calcul et de stockage – les fermes de serveurs – dotées de la technologie permettant de les découper en machines virtuelles, configurables à la demande, pour optimiser leur temps d'utilisation.

De nombreuses questions, en particulier juridiques, apparaissent lors de l'utilisation du cloud. En effet, comme les données sont localisées dans plusieurs pays, quelles lois s'appliquent alors ? Par exemple, les lois sur la gestion des données personnelles diffèrent entre l'Europe et les Etats-Unis.

On sait maintenant faire fonctionner des fermes de serveur avec des millions de processeurs, des PétaOctets (1 Po = 1 000 To) de données. On peut construire, temporairement, à la demande, une infrastructure virtuelle pour exécuter à faible coût de très gros traitements, sans avoir à effectuer d'investissement lourd. Cette flexibilité permet également de répondre à un problème de charge variable au cours du temps.

Une ferme de serveurs, c'est d'abord une masse considérable de matériel informatique, des baies de processeurs, des baies de stockage, des réseaux informatiques, des réseaux de télécoms, un système

électrique fournissant une puissance considérable, un système de climatisation consommant une part importante de cette électricité…

- L'architecture est évidemment distribuée pour gérer les millions d'utilisateurs et de restaurants, les milliards de recommandations et de commentaires.
- La gestion efficace de ces grandes quantités de données conduit à l'optimisation de requêtes et aux index.
- Les données ont une valeur considérable et doivent être très protégées dans le cloud. Cela passe bien sûr par la sécurité, et par exemple, la prévention d'accès non autorisés aux données.
- Cette protection passe aussi par la résistance aux pannes afin d'assurer une disponibilité et une pérennité totales. Les données sont par exemple répliquées pour être accessibles même quand certaines machines, certains réseaux sont en panne.

Avec un nombre massif de machines, de disques, des pannes arrivent statistiquement souvent. Un des défis techniques est le maintien opérationnel d'un système en dépit de la fréquence d'incidents plus ou moins graves. Un autre est la gestion de la climatisation : comment arriver à maintenir à faible coût les machines à une température raisonnable alors qu'elles dissipent une énergie considérable et qu'il peut faire chaud en dehors du centre. Des efforts importants sont consacrés à la baisse du « coût écologique » de tels centres.

Alors, doit-on recourir au Cloud pour nos données et nos applications ? Pesons le pour et le contre :

- Pour : l'absence d'investissement dans une infrastructure matérielle coûteuse, d'obsolescence rapide, dont la qualité de service (disponibilité, efficacité) est techniquement difficile à garantir.
- Pour : la délégation de la gestion des problématiques informatiques « bas niveau » (matériel, systèmes) à des compétences extérieures, afin de mieux pouvoir se concentrer sur son cœur de métier.
- Contre : le prix à payer est une dépendance accrue vis-à-vis des sociétés de service, et une certaine incertitude sur la sécurité des données confiées à une entité externe.

Recourir au cloud, pour une entreprise, c'est donc choisir une forme de spécialisation économique, avec les avantages et inconvénients connus et discutés depuis longtemps.

Pour conclure, observons qu'à une échelle plus modeste, le cloud c'est aussi la possibilité pour une PME, un individu, de disposer d'un serveur à très faible coût (quelques Euros par mois), connecté sur le Web, hébergeant un site web, un blog ou des applications plus ambitieuses. C'est la démocratisation de l'accès au Web en tant qu'acteur.

2. Moteur de recherche et recherche plein texte

Quand on parle de requête dans un contexte de base de données, la référence est SQL. Une requête dans ce langage exprime des conditions sur les nuplets à sélectionner, sous la forme d'égalité entre des attributs et des valeurs simples comme des chaînes de caractères ou des numériques. Cette recherche est de nature booléenne : un nuplet est ou n'est pas dans le résultat (il satisfait ou non les critères), il n'y a pas d'alternative.

Que peut SQL quand les données à traiter ne sont plus des valeurs simples mais des documents textuels, écrits dans une langue particulière, de taille très variable et dont le contenu n'obéit en principe à aucune règle précise ? Que peut-on faire dans le cadre du Web qui contient des milliards

de tels documents ?

Il existe depuis très longtemps un opérateur LIKE qui permet de rechercher des documents contenant une ou plusieurs sous-chaînes. Supposons que l'on veuille sélectionner les nuplets dont le champ `'contenu'` (que l'on suppose de type TEXT) contient les sous-chaînes « MOOC » et « bases de données ». On pourrait essayer d'exprimer cela en SQL :

```
select * from docs
where contenu like('%MOOC%base de données%')
```

Cette requête est très insatisfaisante pour satisfaire ce besoin. Voyons les points qui fâchent :

- L'opérateur LIKE va trouver les documents contenant 'MOOC' avant 'base de données', et pas l'inverse. Il faudrait faire une seconde requête (ou utiliser une disjonction).
- Les sous-chaînes sont recherchées littéralement, au caractère près. Un document qui contient 'Mooc' ou 'mooc', ou contient deux espaces ou un retour à la ligne dans l'expression 'base de données', ne sera pas sélectionné.
- Pour la même raison, SQL ne prendra pas en compte les variantes telles les pluriels ('bases de données'), les conjugaisons de verbe, les acronymes ('BD'), les synonymes ('cours en ligne'), le multilinguisme ('Flot'), etc.

La plupart des systèmes relationnels proposent une extension consistant à exprimer des critères sous forme d'expressions régulières, ce qui ne résout que très partiellement les problèmes ci-dessus. De plus, il faut mentionner une limitation supplémentaire qui les rend vraiment impropres à une recherche « plein texte ». Dans la mesure où on effectue une recherche en fonction d'un besoin exprimé informellement, un processus de nature booléenne qui classe les candidats en deux catégories (sélectionné ou pas) n'est plus satisfaisant ; il est plus naturel d'essayer d'estimer dans quelle mesure un document satisfait le besoin, et de classer les résultats en fonction de cette estimation.

Les moteurs de recherche (des systèmes de recherche plein texte), notamment du Web, tiennent compte de ces spécificités. Ils ont connu un développement considérable depuis 20 ans et la nécessité d'effectuer une recherche rapide et pertinente dans d'énormes masses de documents textuels. Mais même si le web est l'exemple qui vient immédiatement à l'esprit, les moteurs de recherche sont partout : sur votre ordinateur personnel par exemple (Spotlight sous Mac) ou sur votre mobile, dans votre gestionnaire de courrier électronique, etc. Tout le monde a pris l'habitude, dans tous ces contextes, d'utiliser le confort d'une recherche par quelques mots-clés qui ramène une liste classée des documents, ou plus généralement des objets (contacts, rendez-vous, fichiers, articles de blog, etc.), les plus pertinents. On cherche d'instinct la petite loupe, on entre quelques mots dans la fenêtre, et on trouve souvent ce que l'on cherche. Pas de syntaxe compliquée, pas d'hypothèse sur la forme ou l'emplacement des données. Simple et efficace !

3. Sécuriser les données par l'anonymisation

3.1. La problématique

La sécurité des bases de données ne dépend pas seulement du contrôle d'accès. Dans certains cas, par exemple celui d'un administrateur de SGBD malhonnête prêt à diffuser le contenu de la base, ces dispositifs sont insuffisants. Il est donc préférable d'avoir réfléchi en amont aux informations que l'on souhaite conserver et qui sont susceptibles d'être communiquées frauduleusement ou par

erreur. Une attention particulière doit être portée à la gestion de données personnelles, comme le nom, prénom, numéro de sécurité sociale, etc. La loi demande une déclaration à la CNIL de tout fichier (i.e., de toute base de données) traitant de données personnelles. Toutefois, si les données sont *anonymes*, alors plus aucune déclaration n'est nécessaire. Dit autrement, le risque associé au vol d'une donnée anonyme est supposé sinon nul, tout du moins très faible. Dans cet article, nous nous intéressons aux diverses techniques permettant de rendre une base de données anonyme.

Il existe légalement deux types de données : les données à caractère personnel, et les données anonymes. Les données sont à caractère personnel « *dès lors qu'elles concernent des personnes physiques identifiées directement ou indirectement* » pour citer la CNIL. Au contraire, toute donnée qu'il est impossible d'associer avec une personne physique sera dite « *anonyme.* » Il est intéressant de constater que la loi française définit une impossibilité forte, puisqu'elle précise que « *pour déterminer si une personne est identifiable, il convient de considérer* ***l'ensemble des moyens*** *en vue de permettre son identification dont dispose ou auxquels peut avoir accès le responsable du traitement ou toute autre personne.* » Plus mesuré, le projet de règlement Européen prévoit que « *pour déterminer si une personne est identifiable, il convient de considérer* ***l'ensemble des moyens susceptibles d'être raisonnablement mis en œuvre****, soit par le responsable du traitement, soit par une autre personne, pour identifier ladite personne.*

Ces définitions sous entendent qu'il existe plusieurs méthodes d'anonymisation, plus ou moins *efficaces,* au sens d'une protection plus ou moins « forte ». Pourquoi n'utiliserait-on pas toujours « la meilleure » ? Parce qu'il y a un coût à payer pour une anonymisation forte. Calculer une « bonne » anonymisation coûte en temps de calcul. Et puis, plus les données sont anonymes, moins elles sont précises. Par exemple, si on publie des informations au niveau d'un département on est beaucoup moins anonyme que si on les publie au niveau d'une région. Bien que tous ces facteurs entrent en jeu, le facteur déterminant reste cependant le *type de modèle d'anonymisation* utilisé.Comme nous allons le voir, il conditionne l'exploitation des données anonymes et peut la restreindre à certains types de calculs.

Pour illustrer ces techniques, considérons une base de données d'opinions politiques, composée d'une table dont un échantillon est donné ci-dessous. On repère dans un *n*-uplet des données dites *sensibles* comme une opinion politique. De manière évidente, si on connaît le n° de sécu d'une personne, on peut accéder à son opinion politique.

Numéro de sécurité sociale (Identifiant)	**Age**	**Code postal**	**Sexe**	**Parti politique (Donnée sensible)**
2401075123123	75	75005	F	Les Républicains
2750875123123	40	75012	F	Parti socialiste
1931175123123	12	78000	M	Parti socialiste

Figure 1. Une base de données personnelles

3.2. La pseudonymisation

La pseudonymisation consiste à supprimer les champs *directement* identifiants des enregistrements, et à ajouter à chaque enregistrement un nouveau champ, appelé *pseudonyme*, dont la caractéristique est qu'il doit rendre impossible tout lien entre cette nouvelle valeur et la personne réelle. Pour créer ce pseudonyme, on utilise souvent une *fonction de hachage* que l'on va appliquer à l'un des champs identifiants (par exemple le numéro de sécurité sociale), ce qui rend impossible (ou tout du moins extrêmement difficile) la déduction de la valeur initiale. On voit ainsi que deux entités possédant des informations sur une même personne, identifiée par son numéro de sécurité sociale, pourraient

partager ces données de manière anonyme en *hachant* cet identifiant.

Le gros avantage de la pseudonymisation est qu'elle n'impose aucune limite au traitement subséquent des données. Tant que l'on traite des champs qui ne sont pas directement identifiants, on pourra exécuter exactement les mêmes calculs qu'avec une base de données non-anonyme. Ainsi, la Figure 2 illustre un exemple de calcul de la moyenne d'âge pour une opinion politique. L'utilisation de données pseudonymisées ne nuit pas à ce calcul.

Pseudonyme

Donnée sensible

ID	Age	CP	Sexe	Parti politique
1	75	75005	F	Les Républicains
2	40	75012	F	Parti Socialiste
3	22	75019	M	Parti Socialiste

Parti Politique	MOY(age)
L.R.	75
P.S.	31

Figure 2. Pseudonymisation et exemple de calcul

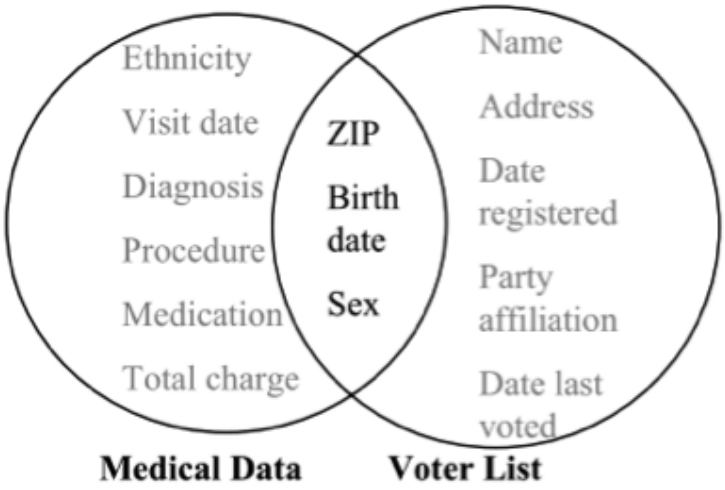


Figure 3. Un exemple de recoupement d'une base anonyme

Toutefois, la pseudonymisation n'est pas reconnue comme un moyen d'anonymisation, car elle ne donne pas un niveau de protection suffisamment élevé : la combinaison d'autres champs peut permettre de retrouver l'individu concerné. Latanya Sweeney l'a mis en évidence aux États-Unis en 2001 en croisant deux bases de données, l'une, médicale, pseudonymisée et l'autre, électorale, avec

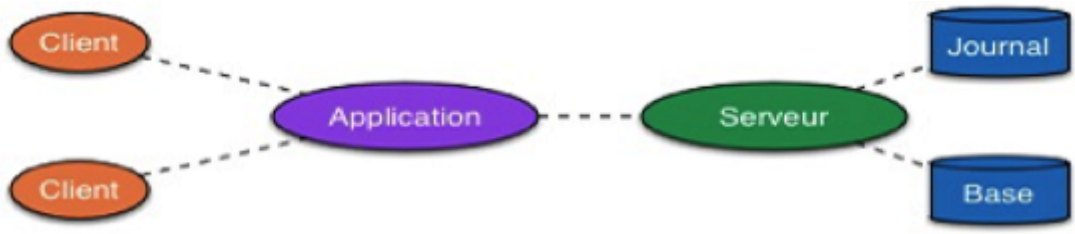
des données nominatives, mais sans valeur sensible. Le croisement a été effectué non pas sur des champs directement identifiants, mais sur un triplet de valeurs : code postal, date de naissance et sexe. Il apparaît qu'un tel triplet caractérise uniquement environ 80% de la population des États-Unis ! L. Sweeneya ainsi pu relier des données médicales à des individus (en l'occurrence le gouverneur de l'état).

4. Vers la très haute disponibilité

La reprise sur panne telle qu'elle est présentée dans le cours permet de s'assurer de l'absence de perte de données, ce qui est déjà beaucoup. En revanche, elle n'assure pas la disponibilité constante du serveur de données puisque ce dernier peut être arrêté pour cause de panne pendant une période allant de quelques secondes (par exemple, relance instantanée du processus serveur après plantage) à quelques heures (par exemple, perte d'un disque suivie d'un remplacement, suivie d'une restauration à partir de la sauvegarde).

De nombreux systèmes applicatifs ne peuvent pas tolérer une telle période d'indisponibilité. C'est le cas par exemple des grands sites de commerce électronique, ou plus généralement des applications dont dépendent de très nombreux utilisateurs (transports, banques, etc.). Il faut dans ce cas développer des méthodes de reprise sur panne pour diminuer le temps de restauration des services et arriver — idéalement — à la garantie d'une disponibilité totale, 24 heures sur 24, 365 jours par an.

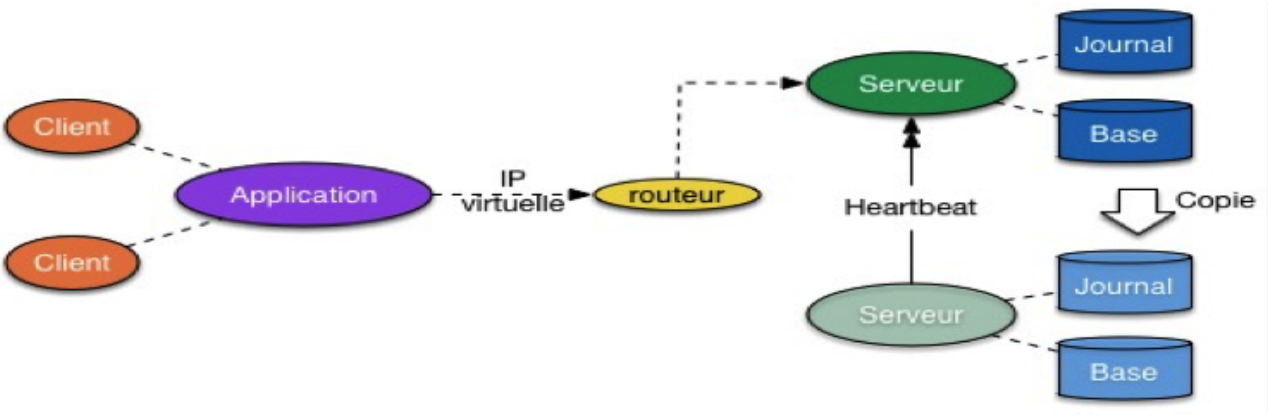
Examinons la chaîne complète des communications entre un client et les données. On peut imaginer par exemple un navigateur web accédant à une application de commerce électronique, lui-même s'appuyant sur un serveur de données gérant le catalogue, le stock, etc. Cette chaîne est résumée sur la figure ci-dessous:



Comment assurer une disponibilité totale de cette chaîne ? Il est clair que la communication entre le client et l'application est hors de notre contrôle. Si le boîtier ADSL de l'utilisateur est défaillant, la transaction sera perdue. Nous pouvons envisager d'agir, en revanche, sur la disponibilité de l'application et du serveur de données.

Les solutions de haute disponibilité s'appuient sur la redondance des données et des services. La reprise sur panne telle que nous l'avons exposée en cours s'appuie d'ailleurs également sur ce principe. Pour être capable de pallier immédiatement la défaillance d'un serveur, il faut un serveur de secours ; pour pallier la défaillance d'un disque, il faut un disque de secours.

Le schéma de base est donc celui présenté sur la figure suivante. Le serveur et la base disposent de copies « fantômes », inactives, mais prêtes à prendre le relais en cas de panne. En mode normal, le serveur fantôme se contente de surveiller l'état du serveur principal en échangeant avec lui des messages à intervalles réguliers (mécanisme dit de « *heartbeat* »). Une copie continue et de préférence synchrone de la base et du journal est effectuée par ailleurs depuis les disques principaux vers les disques fantômes.



Un petit composant complémentaire, le routeur, sert à orienter la connexion des applications. En mode normal, l'adresse fournie par le routeur est celle du serveur principal. En cas de panne de ce dernier, le serveur fantôme va détecter la panne grâce à l'absence de réponse aux messages *heartbeat*, et prendre immédiatement le relais en demandant au routeur de lui adresser les requêtes. Si la base et le journal sont des copies fidèles de la base initiale, le service peut continuer sans interruption et sans perturbation des applications.

Conceptuellement, c'est simple, mais en pratique c'est bien entendu assez compliqué, et donc... coûteux. Il faut doubler l'infrastructure (deux fois plus de disques, deux fois plus de serveurs). Il est essentiel de répartir ces infrastructures dans deux *clouds* séparés physiquement, pour éviter la dépendance à de grosses pannes électriques affectant tout un centre de données, des inondations, ou autres catastrophes naturelles. Il faut mettre en place une architecture logicielle assez lourde, avec le personnel, les procédures de contrôle et de test qui vont avec. Il faut enfin assurer la synchronisation des services fantômes et des services principaux grâce à des composants logiciels spécialisés.

Pour les bases de données (qui ne sont qu'un exemple particulier de service dont la disponibilité doit être assurée), la synchronisation des fichiers de base et du journal pose des problèmes spécifiques. Une solution prête à l'emploi est celle des *disk arrays*, qui assurent des écritures en parallèle sur plusieurs disques. Ces dispositifs permettent de se prémunir contre des pannes fréquentes venant de problèmes matériels sur les disques : il est peu probable que plusieurs disques soient affectés simultanément. En revanche, ils sont peu efficaces contre les événements externes plus exceptionnels, qui demandent d'autres solutions. Certains SGBD proposent une réplication basée sur les écritures dans le fichier journal. Les mises à jour de ce fichier sont transmises vers un disque miroir, ce qui assure une reprise sur panne rapide (mais pas instantanée) avec le serveur fantôme. Enfin, on peut ajouter des composants de réplication synchrone au niveau du système de fichiers lui-même.

Les architectures de haute disponibilité sont étroitement liées aux systèmes de gestion de données distribués. Il est en effet tentant d'utiliser le serveur fantôme pour satisfaire des requêtes au lieu de le confiner à un rôle essentiellement passif. Les systèmes relationnels, dans leur version distribuée, associent clairement les deux problématiques. Une des (rares) caractéristiques commune des systèmes dits « NoSQL » est cette combinaison des fonctionnalités de reprise sur panne et de passage à l'échelle par distribution, au prix de l'abandon de certaines fonctionnalités.

Pour conclure, on notera que le sujet de la haute disponibilité date des débuts de l'informatique. Par exemple, la NASA a fait fortement progresser le domaine dans les années soixante avec le projet Apollo. Il faut aussi noter que les exigences en terme de haute disponibilité dépendent de façon directe de l'application et des coûts induits par une interruption de service. Une panne même de moins d'une minute dans une base de données d'une centrale nucléaire peut avoir des coûts

potentiels en vie humaine inacceptables. L'arrêt pendant plusieurs heures du système de paiement d'une entreprise de vente sur Internet peut avoir un coût financier lourd pour l'entreprise. En revanche, la non-disponibilité pendant plusieurs jours de la base de données d'une association locale de protection des grenouilles, si elle peut être gênante pour les membres de l'association, a des conséquences moins sérieuses. La très haute disponibilité coûte cher, on adaptera donc les solutions aux besoins de l'application.