Systèmes à haute disponibilité

Définitions et solutions

Plan

L'objectif de cette présentation est d'introduire les définitions propres aux systèmes à haute disponibilité et de présenter ensuite des solutions utilisées dans le cadre de serveurs pour des systèmes d'information

Plan

- **Définitions**
  - Sûreté de fonctionnement
  - Coût de l'indisponibilité
  - Classification des systèmes
  - Concepts et terminologie
  - Modes de défaillance
  - Grandeurs caractéristiques
  - Principes de conception
- **Solutions**
  - Solutions au niveau du matériel
  - Solutions au niveau du logiciel

Définitions

Sûreté de fonctionnement -1-

- **Notion de sûreté de fonctionnement : plusieurs définitions**
  - Propriété qui permet à un utilisateur de placer une **confiance justifiée** dans le **service** que le système délivre
  - C'est la propriété et les caractéristiques d'une entité ayant rapport au temps qui lui confèrent **l'aptitude** à satisfaire les besoins exprimés ou implicites pour un intervalle de temps donné et des conditions d'emploi fixées (X50-125, norme de management de la qualité et de l'assurance de la qualité, standard ISO 8402).
  - C'est l'ensemble **des aptitudes** d'un produit lui permettant de disposer des **performances fonctionnelles spécifiées**, au moment voulu, pendant la durée prévue, sans dommage pour lui-même et pour son environnement

- **Propriétés liées a la sûreté de fonctionnement** :

1. Fiabilité *(reliability) :* Correspond à la continuité du service rendu.
2. Disponibilité *(availability) :* Correspond à l'aptitude du système a être prêt a rendre le service pour lequel il a été conçu.
3. Maintenabilitè *(maintenability) :* C'est l'aptitude d'un système a être maintenu en condition opérationnelle.
4. Innocuité *(safety) :* ou encore sécurité vis-à-vis de l' environnement.
5. Immunité *(immunity) :* Correspond à la résistance d'un système aux agressions externes.
6. Pour les systèmes informatiques, *l'immunité* correspond aux 3 critères suivants :
   - ✓ *disponibilité* du système,
   - ✓ *intégrité*
   - ✓ *et confidentialité* des données.

coût de l'indisponibilité

- Coût moyen d'une heure d'indisponibilité du système (Source Contingency Planning Research)

Application	Secteur d'activité	Coût de l'indisponibilité
Courtage	Finance	$6,45 millions
Ventes par carte de paiement	Finance	$2,6 millions
Films a la demande	Loisirs	$150 000
Téléachat	Distribution	$113 000
Ventes sur catalogue	Distribution	$90 000
Réservation aérienne	Transport	$89 500

Classification des systèmes -1-

❑ **D'après [GRA91], classification des systèmes en fonction de leur disponibilité**

- **Hypothèse : un système doit accomplir sa mission 7 jours sur 7, 24 heures sur 24 (7x7, 24x24)**
- **Attention, la terminologie peut varier, il convient de s'en tenir aux chiffres**

Type de système	Indisponibilité (minutes par an)	Disponibilité (%)	Classes de disponibilité
Non géré (Unmanaged)	50000	90	1
Géré (Managed)	5000	**99**	2
Bien géré (Well Managed)	500	**99,9**	3
Tolérant les fautes (Fault-tolerant)	50	**99,99**	4
Haute disponibilité (High Availability)	**5**	**99,999**	**5**
Très haute disponibilité (Very High Availability)	0,5	**99,9999**	6
Ultra haute disponibilité (Ultra High Availability)	0,05	**99,99999**	7

- **Notion d'arrêt programmé (l'arrêt planifié de l'exploitation du système qui n'est pas pris en compte dans le calcul de la disponibilité)**
- **Nécessité de définir, au cas par cas, les limites du système**

Classification des systèmes -2-

❑ **Exemple :** La disponibilité d'un système de classe 5 se caractérise par :

- ➢ une indisponibilité de cinq minutes par an,
- ➢ soit **60 x 24 x 365** - **5** minutes sur un temps théorique total de **60 x 24 x 365**,
- ➢ soit **525 595** minutes de disponibilité sur **525 600** minutes de temps théorique,
- ➢ c'est-a-dire un rapport de **99.999**.

♦ Ce rapport est la mesure de la disponibilité d'un système.

❑ **Remarques :**

- ▪ La **classe de disponibilité** d'un système correspond au **nombre de 9** que comprend la valeur de sa disponibilité
- ▪ Aujourd'hui, un certain nombre de constructeurs s'engagent sur des chiffres de disponibilité des systèmes, classe 4 ou classe 5 (de cinquante minutes a cinq minutes d'indisponibilité par an)

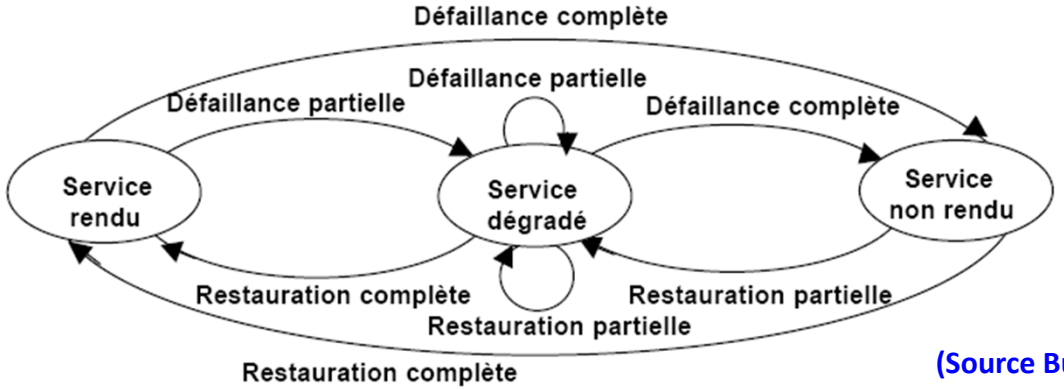
Concepts et terminologie -1-

- Un ***système informatique***, ou système (Computer System, ou System) est un ensemble cohérent et autonome
  - ✓ **d'éléments matériel,**
  - ✓ **logiciel de base** et d'application,
  - ✓ placé éventuellement dans un **environnement réseau.**
- Son comportement est décrit dans un ***document de référence*.**
  - ✓ Ce document revêt une importance capitale, car c'est justement sur les **écarts** par rapport a cette spécification que lon va juger de la **disponibilité du système**.
- **Définitions :**
  1. ***Notion de service attendu*** est l'ensemble des résultats et des conditions de leur délivrance, défini dans un document de référence, que le système informatique doit fournir à l'utilisateur dans un environnement donné.
  2. Notion de ***service fourni*** est l'ensemble des résultats et des conditions de leur délivrance que le système informatique fournit a l'utilisateur dans un environnement donné.

Concepts et terminologie -2-

- **Concepts de service :** La vie opérationnelle d'un système informatique est perçue, par ses utilisateurs, comme une alternance de trois états :

1. ***Service rendu*** *(Proper Service) :* lorsque les **résultats fournis** et leurs **conditions de délivrance** sont **conformes** à ceux du **service attendu**.
2. ***Service dégradé*** *(Degraded Service) :* lorsque les **résultats fournis** sont **conformes** à ceux du **service attendu** mais que leurs **conditions de délivrance** ne sont pas **conformes** à celles du **service attendu.**
3. ***Service non rendu*** *(Improper Service) :* lorsque les **résultats fournis** ne sont pas **conformes** à ceux du **service attendu**.



(Source Bull)

Concepts et terminologie -3-

- La figure précédente introduit le concept **de défaillance :**
  - la défaillance *(failure)* est une **discordance** observée entre le service fourni a l'utilisateur et le service attendu.
- **La défaillance peut être détectée :**
  - **par l'utilisateur** (humain ou autre système) du système
  - ou bien par le **système lui-même**.
- La figure met aussi en évidence **plusieurs niveaux de défaillance** :
  1. ***Défaillance complète*** *(Complete Failure).* C'est une discordance sur **les résultats du service**, par exemple blocage (« plantage »), ou *crash* du système d'exploitation.
  2. ***Défaillance partielle*** *(Partial Failure).* C'est une discordance **sur les conditions de la délivrance des résultats**, par exemple perte de performance ou support d'un nombre moindre d'utilisateurs.

Concepts et terminologie -4-

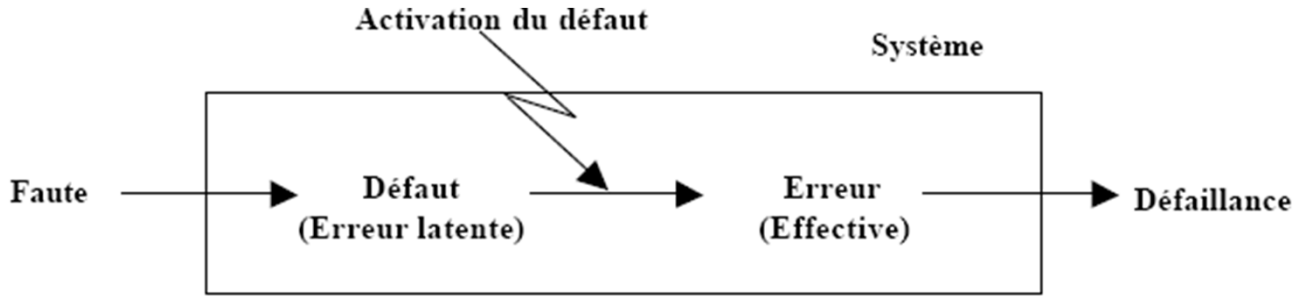
- Les **défaillances** sont causées par les **erreurs**.
  - **Une erreur** *(Error),* a l'intérieur d'un système informatique, est la **discordanc**e entre une valeur ou une condition, **calculée** ou **observée**, et la valeur ou la **condition théorique correspondante.**
  - ➢ **Une erreur** peut entraîner une défaillance.

- Un **défaut** ou **erreur latente** *(Defect* ou Latent Error) est une partie du système informatique **inadaptée ou manquante**.
  - ➢ **L'activation** d'un défaut produit une erreur.
  - ➢ Il est habituel de distinguer des **classes de défauts**,
    - défauts physiques,
    - défauts de conception, etc.
  - ➢ Il convient, dans une étude de sûreté de fonctionnement, de préciser à quelles **classes de défauts** on s'intéresse et comment on placera la limite entre les classes.

- La **faute *(Fault)*** est une défaillance du système ou des êtres humains chargés de la production du système informatique ou du système environnant le système informatique considéré
  - La faute peut introduire un défaut dans le système informatique.

Concepts et terminologie -5-

- **Figure suivante illustre ces concepts :**



- **Cette figure montre que :**
  - ⇨ La **conséquence d'une faute**, qu'elle soit de nature humaine (faute commise par un programmeur dans la conception ou la rédaction d'un programme, par exemple) ou non, est l'introduction **d'un défaut dans le système**.
  - ⇨ Ce défaut est aussi **appelé erreur latente**, car tant que ce défaut n'est pas **activé** n'a pas d'effet)
    - Exécution d'une séquence de code erronée, par exemple,
    - Ou bien utilisation de la valeur d'une variable initialisée avec une valeur incorrecte ou d'une constante dont la valeur n'a pas été fixée correctement),

    il ne se passe rien au niveau du système.
  - ⇨ Dès que le défaut est **active**, **il se produit une erreur** : c'est-à-dire que l'on est passé d'une erreur latente a une erreur effective.
  - ⇨ Une erreur **effective** peut conduire à une **défaillance du système**

Concepts et terminologie -6-

- **Types de défauts :** Dans l'analyse des causes de défaillance des systèmes, on est amené faire une **classification des défauts.**
  - **Dans la pratique, on constate que** :
    - Certaines erreurs se reproduisent de ***façon systématique*****,** quelles que soient les conditions dans lesquelles le module contenant le défaut est appelé
    - D'autres erreurs ***ne sont pas reproductibles de façon systématique*****.** De tels défauts se rencontrent surtout au **niveau du logiciel**.
    - On rencontre aussi des défauts de ce type au niveau du matériel : **on parle alors de *défauts transitoires.***
  - D'une façon schématique, on peut considérer que l'on est en présence
    - soit de **défauts parfaitement déterministes,**
    - soit de **défauts aléatoires**.
  - Les auteurs anglo-saxons ont proposé une terminologie pour ces défauts : respectivement les **Bohrbugs** et les ***Heisenbugs.***
    - ***Bohrbug :*** un réessaie conduit de nouveau à une erreur (et éventuellement a une défaillance).
    - ***Heisenbug :*** un réessaie ne conduit pas systématiquement à une erreur.

Modes de défaillance -1-

- **Défaillance = non conformité de la réponse d'un système par rapport a ses spécifications.**

- **On rappelle l'importance que revêt la spécification du comportement d'un système**
  - En effet, en l'absence d'une telle spécification, il n'est pas possible de différencier les **comportements normaux** de beaucoup des **comportements anormaux**

- **La définition du comportement d'un système doit comprendre la définition de la réponse du système :**
  - La réponse d 'un système est fonction de **l'état initial** du système et des **données d'entrée**
  - La réponse doit être fournie à l'intérieur **d'un intervalle de temps**,
  - Le système doit se trouver dans un état résultant **spécifié,**
  - Les valeurs des données fournies par le système sont **spécifiées**

- **Modes de défaillance : on distingue quatre classes de défaillances :**

1. **Défaillance par omission (Omission Failure) :**
   - Cette classe de défaillance se caractérise par **une absence de réponse** du système suite a une sollicitation.

2. **Défaillance temporelle (Timing Failure) :**
   - Cette classe de défaillance se caractérise par le fait que le système ne répond pas à une sollicitation dans **l'intervalle de temps spécifié**.
   - On peut distinguer :
     - ✓ **Une défaillance hâtive** *(early* timing *failure)*
     - ✓ **Une défaillance tardive** *(late timing failure),*

     selon que le système réponde trop tôt ou top tard.

3. Défaillance de réponse (Response Failure) :

- Cette classe de défaillance se caractérise par le fait que le système fournit une valeur incorrecte

  Défaillance de valeur, (ou *value failure)*

- ou par le fait que la transition d'état du système qui se produit est incorrecte

  Défaillance de transition d'état, (ou *state transition failure).*

4. Défaillance de type «plantage» (Crash Failure) :

- Si, après la première omission à produire une réponse, un système ne répond plus aux sollicitations

Grandeurs caractéristiques de la disponibilité des systèmes -1-

- **Il s'agit de grandeurs accessibles et mesurables (comme toute mesure), elles doivent vérifier les propriétés suivantes :**
  - **représentativité**, **interprétables**, **fidèles**, **précises** et **utiles.**

A. Mesures liées au concept de défaillance

- **MTTF :** ou Temps moyen jusqu'à défaillance *(Mean Time To Failure) :*
  - Esperance mathématique de la durée de fonctionnement jusqu'a **la première défaillance**.
  - Il est estimé **par le rapport** des **cumuls de durées de service rendu** au **nombre de défaillances** pendant l'intervalle de temps considéré.

- **MTBF :** ou Temps moyen entre défaillances *(Mean Time Between Failures) :*
  - Espérance mathématique de la durée entre deux défaillances.
  - Il est estimé par **le rapport** des cumuls de **durées de service rendu** et des **temps de restauration** au **nombre de défaillances pendant** l'intervalle de temps considéré.

B. Mesures liées au concept de maintenabilitè

- **MTTRes : ou Temps moyen jusqu'a restauration *(Mean Time To Restore* ou *Mean Time to Recover) :***
  - Esperance mathématique de la durée jusqu'a restauration.
  - Ce temps est estimé par **le rapport** du cumul des **durées de restauration** au **nombre de restaurations** pendant l'intervalle de temps considéré.

- **Note :** si pas de redondance MTBF = MTTF + MTTRes (MTTRes étant le temps moyen nécessaire a la remise en service du système : voir ci-après)

- **MTTRep : ou Temps moyen jusqu'a réparation (élément) *(Mean Time To Repair element) :***
  - Esperance mathématique de la durée jusqu'a réparation.
  - Ce temps est estimé par le rapport du cumul des **durées de réparation** au **nombre de réparations** pendant l'intervalle de temps considéré

C. Mesure liée au concept de disponibilité

- ***Disponibilité (Availability).*** C'est la probabilité, exprimée en pourcentage, que le service soit rendu a l'instant.

- Connaissant le **MTTF** et le **MTTRes**, la disponibilité technique (**At**) s'exprime de la façon suivante :

$$At = MTTF/(MTTF + MTTRes)$$

Principes de conception -1-

- **Concept d'état d'un système** :
  - ⇨ **Hypothèse : D'un point de vue externe** (c'est-à-dire visible des utilisateurs du système) , un système **à haute disponibilité** doit **masquer les défaillances** auxquelles il est sujet vis-à-vis de son environnement.
  - ⇨ Conséquence : un système, suite a une défaillance, doit continuer les opérations a partir **d'un état défini**.
- **Comment ? Deux** approches sont possibles :
  1. Le système maintient *en permanence* **plusieurs « versions » de l'état courant** du système et continue, en cas de défaillance, l'exécution à partir de l' un des états ***courants disponibles*** (**c'est le cas des systèmes proposés par Stratus**).
  2. Le système maintient **des « états » à partir desquels le traitement peut être repris**. C'est la technique **des points de reprise** (**cas des systèmes proposés par Tandem**, ainsi que des solutions a base de clusters).

- **Remarques:**

1. La solution qui consiste à maintenir en permanence **plusieurs contextes** implique une **redondance,** au moins **au niveau du matériel.**

2. La **reconstitution d'un contexte** permettant la reprise des opérations impose des contraintes
   - ☞ **Soit au niveau de la programmation** (⇔ approche explicite de la définition du point de reprise),
   - ☞ **Soit au niveau du matériel** (⇔approche implicite de la définition du point de reprise)

3. les **systèmes distribués** présentent une difficulté majeure, qui est la reconstitution d'un **état global cohérent** (c'est-à-dire pour l'ensemble des systèmes composant le système distribué).

- **Principes de conception :** La conception des systèmes a haute disponibilité obéit a un certain nombre de principes

1. Modularité :

- C'est un principe très classique dans la conception des systèmes, dont les avantages sont bien plus généraux que la seule conception des systèmes a haute disponibilité.
  - ⇨ Un système est constitué par un ***ensemble de modules.***
  - ⇨ La décomposition d'un système en un ensemble de modules est le résultat de la ***conception fonctionnelle*** du système.
  - ⇨ Un module est une unité de ***service***, de ***cloisonnement des fautes*** *(fault containment)* et de ***réparation.***

2. Fail Fast : ⇔ fonctionnement correct ou arrêt immédiat :

- Le respect de ce principe implique que tout module fonctionne de façon correcte ou arrête immédiatement son exécution **des la détection d'une erreur.**
- Un certain nombre de terminologies sont utilisées autour de ce principe, en particulier *:*

  A. Fail Safe

  B. Fail Silent

  C. Fail Stop

A. **Un système (ou un sous-système) est dit « *fail safe* »** si **les conséquences** de ses défaillances sont, du point de vue de l'utilisateur, uniquement du même ordre de grandeur que les bénéfices que l'utilisateur peut retirer d'un fonctionnement normal.

⇨ **On parle alors de défaillances bénignes *(benign failures).***

⇨ C'est une notion assez subjective ;

- En effet, la ***perte d'une opération*** peut avoir des conséquences très différentes selon qu'il s'agit ***d'opérations statistiques portant sur un grand nombre d'expériences*** ou ***de transactions financières***.

B. **Un système (ou un sous-système) est dit « fail silent »** si ses défaillances sont uniquement, du point de vue de l'utilisateur, **des défaillances de type *crash* du système.**

⇨ En d'autres termes, **une défaillance provoque « un *crash* » (ou « plantage ») du système**.

C. **Un système (ou un sous-système) est dit « *fail stop* »** si ses défaillances sont uniquement, du point de vue de l'utilisateur, **des défaillances de type « arrêt » du système.**

⇨ En d'autres termes, **une défaillance provoque « l'arrêt » du système.**

- La notion de « **fail fast »** implique qu'un systeme, dans la mesure ou il est **muni de redondance,** surveille en permanence l'ensemble de ses constituants pour déterminer les constituants défaillants et ceux qui sont en état de bon fonctionnement.
  - Cette surveillance est assurée par **l'échange périodique de messages** entre les constituants du système.
  - On désigne souvent ce type de surveillance sous le nom de **battement de cœur (Heartbeat).**

3. Indépendance des modes de défaillance :

⇨ **Les différents modules constituant un système et leurs interconnexions doivent être conçus de façon que la défaillance d'un module *n'affecte pas les autres modules.***

4. Redondance et réparation

- La redondance est l'une des techniques de base pour la construction de systèmes à haute disponibilité.
- **L'installation d**e modules de rechange (niveau physique) et leur **prise en compte** dans la configuration du système (niveau logique) doivent pouvoir se faire pendant le **régime de fonctionnement normal du système**.
  - ⇨ En d'autres termes, cette installation doit pouvoir se faire en avance.
- De cette façon, lorsqu'un module défaille, le système peut avoir immédiatement recours au module de secours
- La r**edondance** et la **réparation** sont des éléments clés de la **disponibilité des systèmes.**
  - ⇨ En plus de la possibilité de retirer les éléments défaillants et de procéder a leur réparation alors que le système continue a fonctionner, c'est, d'une façon plus générale, **la strategie de reparation** qui contribue, de façon significative, à **la disponibilité d'un systeme**.

5. Elimination des points de défaillance uniques :

- Un **point de défaillance unique** *(**S**ingle* **P**oint **O**f ***F**ailure,* ou **SPOF**) est un élément dont la défaillance entraine la **défaillance complète du système**.
- L'un des principes de la conception des systèmes à haute disponibilité consiste à ***identifier*** et ***à éliminer*** les points de défaillance uniques.

- **Quelques exemples simples de points de défaillance uniques :**
  a) L'alimentation en énergie électrique pour un système ne disposant pas d'une alimentation secourue (par exemple une batterie).
  b) Le bus système pour un **SMP** (**S**ymmetric **M**ulti**P**rocessing).
  c) Le système d'exploitation dans le cas d'un système monoprocesseur ou d'un système multiprocesseur symétrique : la défaillance du système provoque l'arrêt du système, et le retour a l'état opérationnel nécessite une réinitialisation du système.

- **Mise en œuvre des principes de conception :**
  - Le tableau ci-dessous passe en revue la façon dont les principes énoncés ci-dessus sont mis en ouvre au niveau **du logiciel** et au niveau **du matériel**

Concept	Matériel	Logiciel
Modularité	**Classique**	**Classique**
Fail Fast	• Auto vérification • Comparaison	• Auto vérification • Programmation en *N* versions
Independence des modes de défaillance	***Mode de couplage*** entre les modules	***Mode de couplage*** entre les modules
Redondance et réparation	Modules de rechange approche *N+1* Echange de modules en fonctionnement	Redondance de certains modules logiciel Remplacement du logiciel en marche
Elimination des points de défaillance uniques	Redondance	Redondance

- **Dans le cas de la modularité :**
  - La notion de module facilite **la conception**, la **mise au point**, la **maintenance** et la **réutilisation**,
  - le principe de **modularité** est généralement appliqué tant **pour le matériel que pour le logiciel**.
  - Elle est exploitée de façon plus systématique **dans le materiel**, où le concept de module correspond à **une realite physique**.
  - Le principe de modularite consiste à concevoir un systeme (ou une application) comme un ensemble, **d'unités coopérants** ;
    - une unité réalisant une fonction ou un ensemble de fonctions bien déterminé.

- **Dans le cas de Fail Fast :**

A. *Dans le cas du matériel,* les deux approches peuvent se caractériser de la façon suivante :

- **L'auto vérification (self checking)** : un module implémente non seulement *une fonction* mais aussi effectue des travaux *complémentaires permettant de s'assurer de la validité de l'état obtenu*.
  - Les **codes détecteurs et correcteurs d'erreur des mémoires** (**ECC**, pour **E**rror **C**orrecting ***C**ode)* et **les codes cycliques utilisés dans les protocoles de communication** sont des exemples de ces techniques.

- **La comparaison** est réalisée par :
  - ✓ la multiplicité des éléments implémentant une fonction (par exemple plusieurs processeurs)
  - ✓ et *le test de l'identité* des résultats obtenus par ces différents éléments.
  - Cette approche peut aller **jusqu'au vote majoritaire.**

☞ **Les techniques d'auto vérification** sont largement utilisées. Ainsi, la quasi-totalité des serveurs ont une mémoire munie d'un **ECC** permettant, en utilisant 72 bits pour représenter 64 bits d'information, de corriger une erreur simple (un bit erroné) et de détecter des erreurs doubles (deux bits erronés).

☞ **Les techniques a base de comparaison** au niveau du matériel sont assez communes dans les systèmes a **forte criticité**, comme les systèmes proposés par Stratus et Tandem (Himalaya).

B. *Dans le cas du logiciel*, les deux approches peuvent se caractériser de la façon suivante :

- **L'auto vérification *(self-checking)* : correspond a une conception particulière des modules logiciels, avec :**
  - ✓ **un test a l'entrée du module vis-à-vis des paramètres**
  - ✓ **et un test de vérification de la validité des résultats à la sortie, portant sur les valeurs restituées.**
  - ✓ **Le code du module peut aussi être muni d'assertions permettant de vérifier, en cours d'exécution, la validité de certaines hypothèses concernant le fonctionnement du module.**
    - ⌦ **On parle de programmation défensive *(Defensive* Programm*ing)***

Principes de conception -14-

- **La multiplication des modules qui consiste a réaliser la même fonction de différentes façons et a comparer les résultats.**
  - **Avec cette technique,** le système est constitué d'un ensemble **de modules redondants.**
  - **Au sein de chacun de ces modules, la même fonction** est implémentée de **différentes façons.**
  - **A la fin de l'exécution d'une fonction, les résultats produits par les différents modules sont comparés.**
    - ⇨ **En cas de divergence, l'exécution est suspendue.**
    - ⇨ **Si l'on dispose d'un nombre impair de modules (au moins supérieur ou égal a 3), on peut procéder a une comparaison sur les sorties et choisir, a l'aide d'un vote majoritaire, la réponse.**
  - **Le ou les composants ayant fourni une valeur divergente sont écartés de la configuration opérationnelle.**
  - **On désigne ce type de programmation *N-version Programming* (programmation en *n* versions) ou « diversité de conception ».**

- **L'indépendance des modes de défaillance implique** que le degré de couplage entre les modules soit ***nul ou minimal.***
  - ⇨ Ainsi, toute interaction entre modules, que ce soit au moyen de messages ou par mémoire partagée, doit être protégée.
  - ♦ Une telle approche a des implications évidentes sur les coûts de développement et sur les performances :
    - – ***La communication par mémoire partagée***, ***au niveau du matériel***, doit faire l'objet de mécanismes de contrôle (par exemple des protections mémoire telles que celles que l'on trouve dans les dispositifs implémentant la mémoire virtuelle).
    - – ***La sécurisation des communications par mémoire partagée*** –***au niveau du logiciel*** – revient pratiquement à transformer cette communication en une ***communication par message.***
      - ⇨ De cette façon, on perd l'efficacité de la communication qu'apporte la mémoire partagée.

- **Le principe de redondance et de réparation** est très souvent mis en œuvre au niveau du matériel.
  - Pour augmenter la disponibilité des systèmes, on s'oriente de plus en plus vers des systèmes où les éléments peuvent être élimines, échanges, ajoutés sans ***interrompre le fonctionnement du système.***
  - On utilise parfois le vocable de **« tolérance à la réparation »** pour designer de tels systèmes.
  - On désigne de tels éléments sous le nom de FRU (Field *Replaceable Unit)* ou de CRU *(Customer Replaceable* Unit).
  - On élimine (ou on limite) ainsi les interventions du personnel de maintenance sur le site.

- **L'élimination complète des SPOF se révèle souvent très coûteuse.**
  - C'est pourquoi les constructeurs de systèmes s'orientent vers des solutions dans lesquelles des SPOF peuvent subsister pour des ***éléments de très haute fiabilité*** (c'est-a-dire des éléments dont la probabilité de défaillance soit quasiment négligeable vis-à-vis du taux de défaillance des autres composants).
  - **En ce qui concerne le logiciel,** élimination des point de défaillance uniques passe aussi par la redondance

Notes :

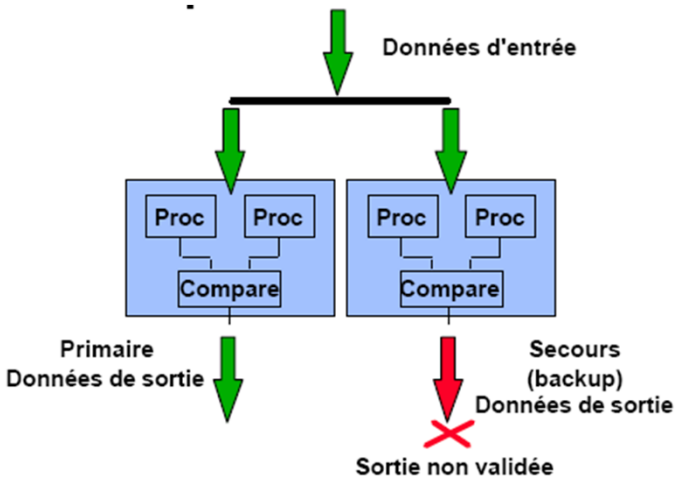
- *Les solutions proposées sont conçues pour résister à une défaillance. Pour la commodité de l'exposé, on a séparé les solutions en deux familles*
  - *Solutions au niveau du matériel;*
  - *Solutions au niveau du logiciel.*

- *De fait, une solution peut* ***combiner les deux approches.***

Solutions au niveau matériels

❑ « Pair and Spare» de Stratus (ancien système) (⇔ doublement et rechange)

A. Illustration du concept



Principe de l'architecture de Stratus concernant les unités de traitement

❑ Principe de fonctionnement :

1. *Appariement (Pair) des organes actifs :*

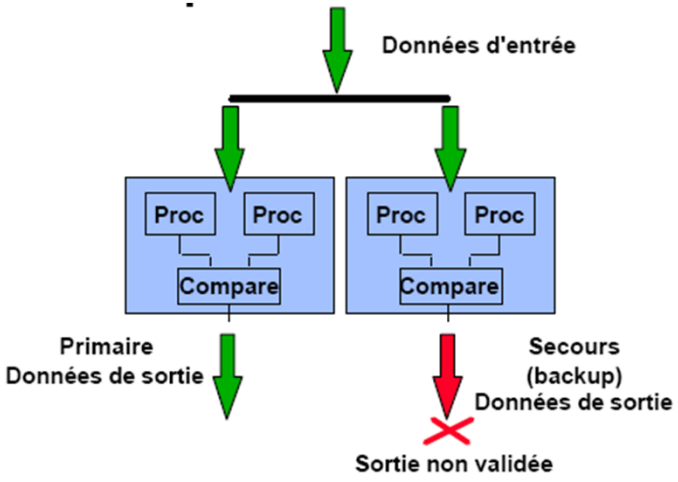
- Le même processus est exécute en parallèle par deux processeurs au sein d'un même organe

2. *Doublement (Spare) des organes actifs :*

- Les deux organes exécutent le même processus, l'un des deux est dit organe primaire.
- Seules les sorties de l'organe primaire sont validées.
- En cas de défaillance de l'organe primaire, l'organe de secours prend le relais et ses sorties sont validées.
- Les différents organes peuvent être échangés sans interrompre le fonctionnement du système.

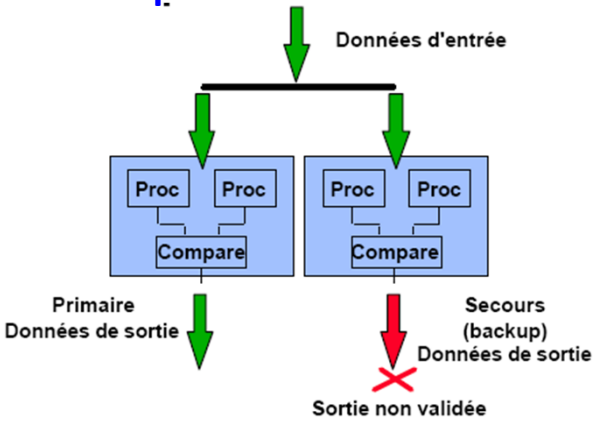
B. Fonctionnement :

- **Dans cette architecture,** une unité de traitement logique est composée de **deux cartes processeur physiquement indépendantes**.
- Ces cartes exécutent le **même processus de façon synchronisée**,
  - ⇔ les mêmes instructions et les mêmes données étant présentées aux deux cartes processeur.
- **L'une des cartes est dite carte primaire**
  - ⇔ Ses sorties (opérations sur la mémoire et les entrées-sorties) sont validées.
- **L'autre carte est dite carte de secours (*backup*)**
  - ⇔ ses sorties ne sont pas validées.



Stratus -3-

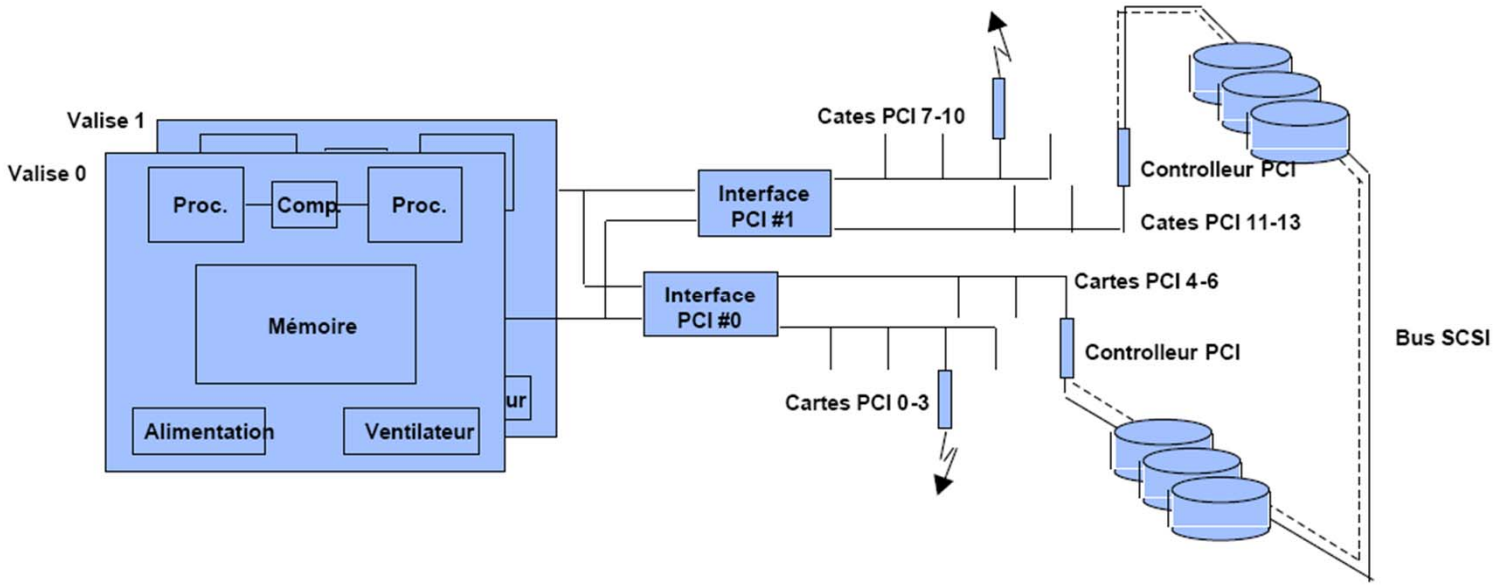
- Les deux microprocesseurs travaillent de façon synchronisée (mode dit *lock step),* ainsi que d'une logique de comparaison.
- Les résultats fournis par les deux microprocesseurs sont comparés.
  - En cas de divergence, la carte processeur est déclarée en défaut et écartée de la configuration.
  - Aucun des résultats produits par chacun des microprocesseurs n'est pris en considération.



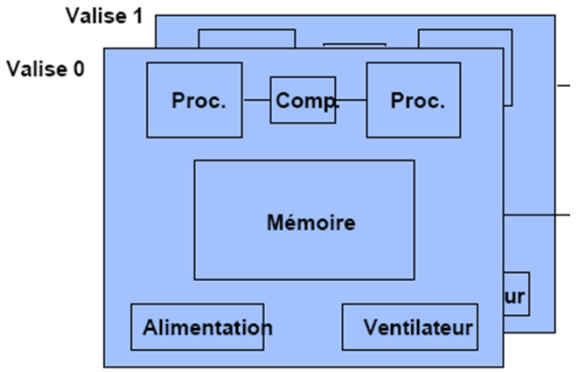
- Comme le même traitement se déroule en parallèle sur la carte de secours, elle aussi munie dune paire de processeurs.
  - Les résultats produits sur cet organe sont pris en considération (en d'autres termes, la carte de secours devient carte primaire) et le traitement continue (de façon transparente).

C. Exemple : Architecture Continuum 4000 de Stratus fondée sur le principe précédent

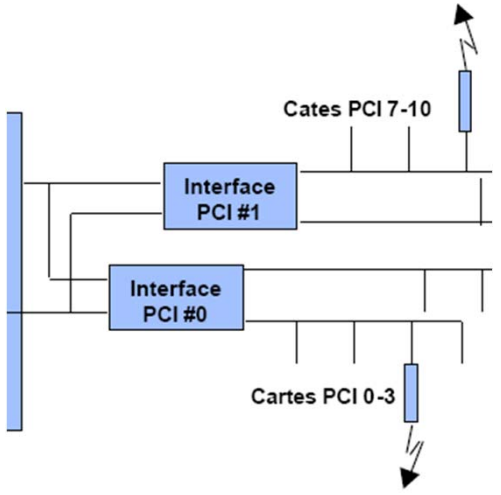
- Fondé sur processeurs HP PA 8000 (Precision Architecture) du type RISC
- Deux systèmes d 'exploitation supportés :
  - VOS Virtual Operating System (Stratus)
  - HP-UX (Unix de HP adapté a PA)
- Prochaine génération à base d'IA-64



- **L'unité de traitement est conditionnée sous forme de valise *(case)*, comprend :**
  - Les deux microprocesseurs,
  - La logique de comparaison,
  - La mémoire (jusqu'a 2 Go),
  - L'alimentation électrique et le ventilateur

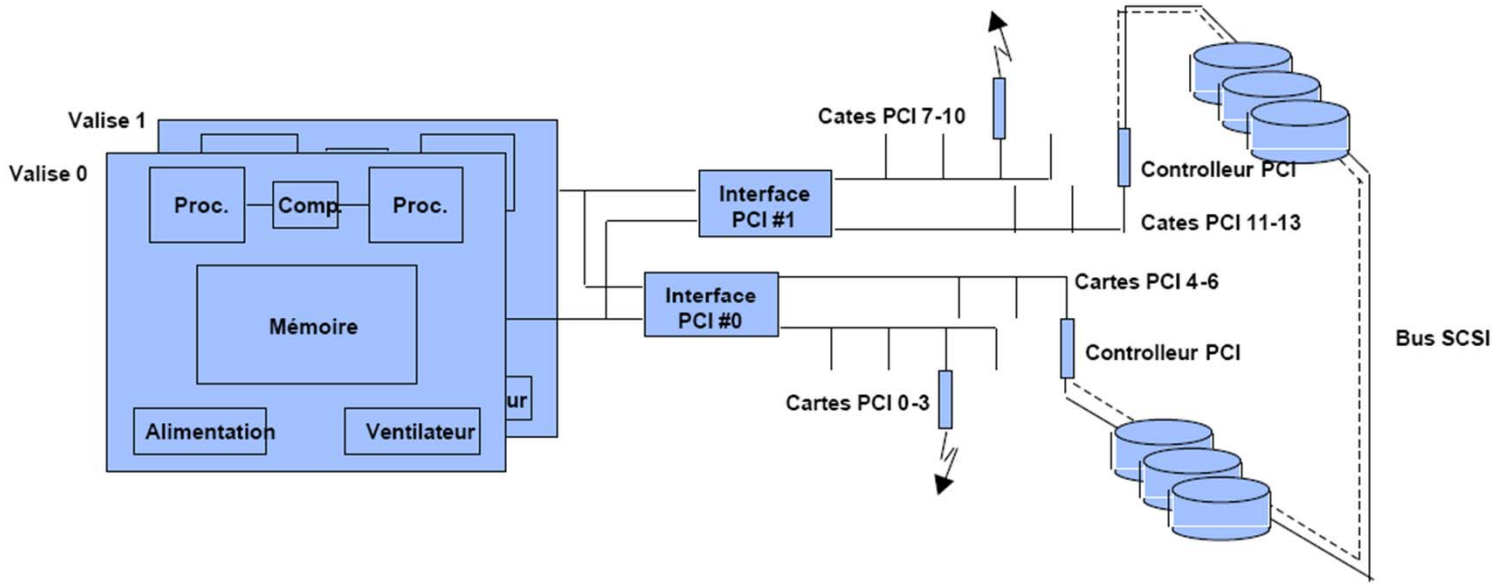


- Chacune des valises peut accéder aux **deux bus d'entrées-sorties au standard PCI** (Peripheral *Component Interconnect,* ou connexion de composants périphériques).



- Les disques sont en **double accès** entre les contrôleurs SCSI, liés chacun a l'un des bus PCI.
  - les disques demeurent accessibles malgré la défaillance de l'un des bus PCI ou de l'un des contrôleurs

- Il y a aussi redondance du côté des contrôleurs de communication (deux bus PCI)



Solutions au niveau du logiciel

Introduction

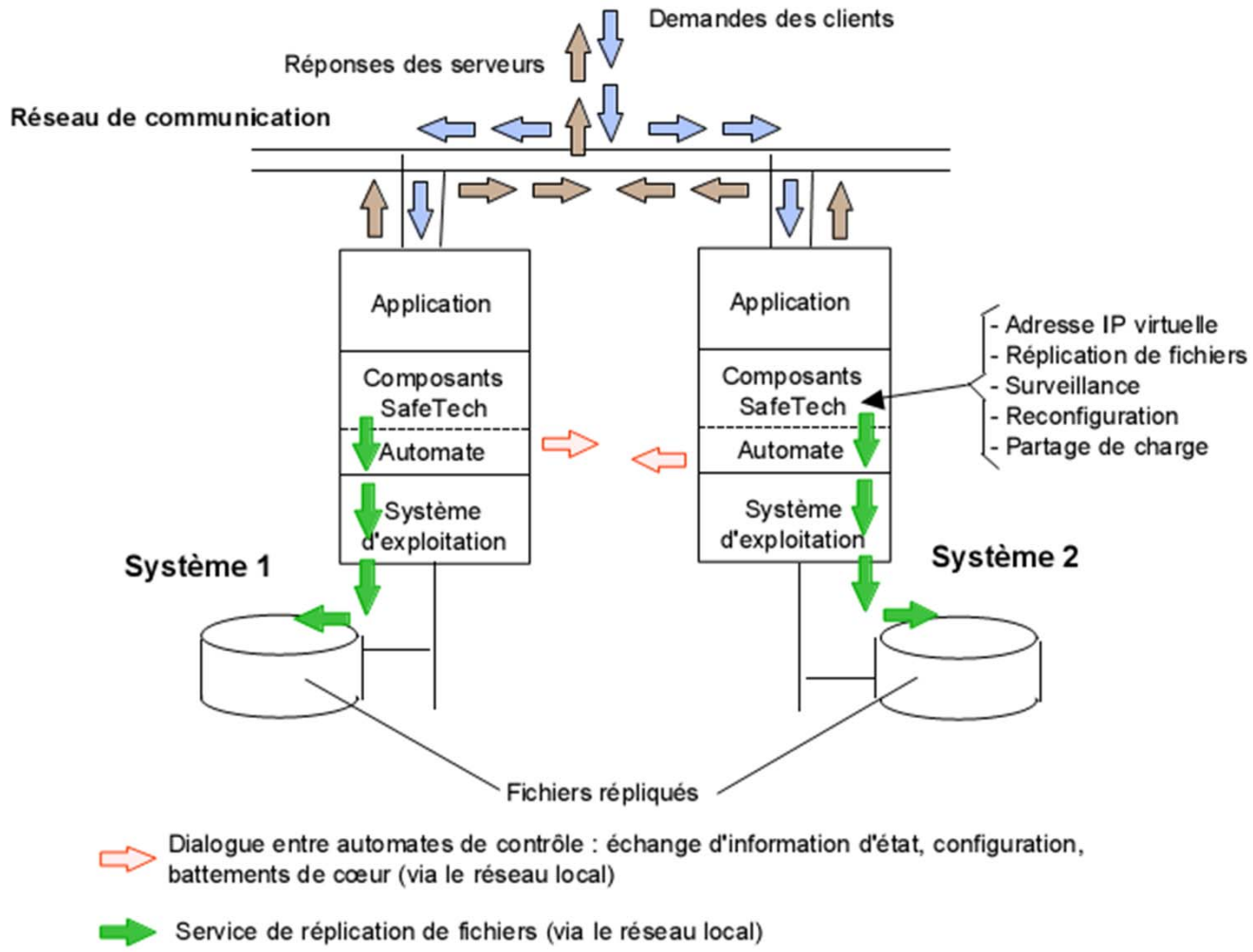
- Le principe des solutions de type logiciel à la **continuité de service** est de **mémoriser des états a** partir desquels le traitement peut être repris suite a une défaillance.
  - La définition de tels points est implicite dans les systèmes transactionnels : s'agit des fins normales des transactions.

- Le système transactionnel est typiquement fondé sur un **système de gestion des transactions** (par exemple CICS, sur les systèmes mainframes d'IBM, Tuxedo, sur les systèmes Unix, Microsoft Transaction Service, ou MTS, sur le système NT, etc.) **et de gestionnaires de données** (comme DB2, Informix, Oracle ou Sybase).
  - ⇨ En cas de défaillance, le système transactionnel se **sert des journaux** pour annuler les effets des transactions qui n'ont pas encore été validées au moment de la défaillance.

SafeTech -1-

- Technologie générique permettant de bâtir des solutions a haute disponibilité sur des systèmes standard (NT ou Unix). Permet de résister aux défaillances du matériel et du logiciel
- Nécessité au moins deux **systèmes interconnectés par un réseau local**
- Ensemble de composants middleware installe sur la plate-forme et fonctionnant sous le contrôle d 'un automate :
  - **Virtualisation de I 'adresse IP**
  - **Système de fichiers redondant distribué**
  - **Mécanisme de surveillance des systèmes**
  - **mécanisme de reconfiguration**
  - **mécanisme de partage de charge**
- Peut nécessiter une adaptation des applications pour pouvoir bénéficier de la continuité de service
- Deux modes de fonctionnement :
  - **primaire-secours (un seul système actif, I 'autre «suit»)**
  - **partage de charge**
- Utilisé dans différents produits (ex pare-feu, serveur Web,...)

- Architecture générale



- **Rôle des différents composants :**
- L'automate de contrôle configure les différents composants logiciels de SafeTech en fonction de l'état des serveurs.
  - → Il est implémenté au moyen d'un **automate a états finis.**
  - → Les transitions d'état sont conditionnées par l**a réception d'événements** : changement d'état de l'autre système ou événement local (défaillance logicielle, par exemple).
  - → Une transition d'état peut provoquer **l'exécution d'un script** destine a reconfigurer les modules du système.
  - → L'automate de contrôle assure également la surveillance de l'état de bon fonctionnement de l'autre système au moyen de la technique des **battements de cœur** *(heartbeats).*

- Des mécanismes **appropries de la gestion de réseau** permettent plusieurs systèmes de partager simultanément une même adresse IP.
  - De cette façon, les clients du serveur virtuel ne connaissent qu'une **seule adresse réseau**.
  - Cette adresse virtuelle possède deux modes de fonctionnement **: *primaire-secours*** et ***partage de charge***.
    1. **Dans le mode primaire secours,** le flux d'information (en provenance des clients) est vu par les deux serveurs mais n'est traité que par le serveur primaire. La tâche du serveur de secours est de surveiller le serveur primaire et de se substituer a lui en cas de défaillance.
    2. **Dans le mode partage de charge**, tous les serveurs voient la totalité du trafic réseau, même si chacun d'eux n'en traite qu'une partie.

- **Cette technique présente les avantages suivants :**
  - **Meilleure isolation** par rapport a la défaillance de l'un des systèmes, temps de reprise rapide
  - **Tolérance aux catastrophes naturelles,** les deux systèmes pouvant être installes dans des bâtiments différents (la limite est celle imposée par la technologie du réseau local utilise).
  - **La réplication peut être limitée aux seuls éléments pertinents** (notons que cela implique une analyse spécifique de ('application pour déterminer ces éléments pertinents).
  - **En addition aux mécanismes classiques de surveillance mutuelle des systèmes** (matériel, logiciel de base et réseau), les mécanismes de surveillance de système a granularité fine peuvent être paramètres de sorte a assurer la surveillance mutuelle (par exemple au niveau du logiciel d'application).

- Les mécanismes de reconfiguration permettent la reconfiguration au sein d'un même système ou entre plusieurs systèmes au travers d'un réseau.
- Lorsqu'une panne est détectée sur le serveur primaire (ou sur l'un des serveurs en mode partage de charge), les applications concernées sont redémarrées sur le serveur de secours.
- L'état des fichiers répliqués correspond a la dernière écriture émise par le serveur défaillant.
- Les connexions TCP/IP avec les applications clientes sont interrompues et doivent être rétablies. II est possible de rétablir ces connexions automatiquement au moyen dune programmation spécifique.
- Si la panne du serveur défaillant a une origine logicielle, le serveur est automatiquement redémarré.

SafeTech -5-

- Après que le serveur a été redémarre (ou réparé dans le cas dune défaillance matérielle), son système de fichier partage est automatiquement resynchronisé avec l'état courant sur le serveur valide.

- Par la suite, ce serveur redevient nœud de secours ou serveur actif (cas du partage de charge), et le système retourne a un état de haute disponibilité.

- Les **mécanismes de partage de charge** entre plusieurs systèmes d'un réseau permettent, outre les caractéristiques de haute disponibilité, **d'optimiser les performances de l'ensemble.**

- La technologie SafeTech a été utilisée pour rendre des applications existantes tolérantes aux fautes.