LA TOLÉRANCE AUX PANNES

DANS LES SYSTÈMES RÉPARTIS

G. Florin

Laboratoire CEDRIC

CNAM

PLAN DE L'EXPOSE

- Introduction: concepts de base de la sûreté de fonctionnement

- Classification des pannes.

- Différents types de redondances.

- Principaux problèmes de la tolérance aux pannes.

- Conclusion.

INTRODUCTION: CONCEPTS DE BASE

DES SYSTÈMES REPARTIS

SURS DE FONCTIONNEMENT

Système sûr ("dependable")

Qualitativement on peut avoir **"confiance"** dans le service qu'il rend.

Système réparti sûr

- Ensemble de plusieurs calculateurs reliés en réseau qui collaborent pour des traitements à contrainte de sûreté

En contrôle de processus industriel
. fortes contraintes de sûreté
. contraintes d'échéances fortes/faibles
. support de dispositifs spécifiques

Mais aussi en gestion transactionnelle
. faibles contraintes de sûreté
. faibles contraintes d'échéance

Tout système réparti doit assurer la prise en compte des contraintes de sûreté (algorithmique répartie des systèmes et des réseaux, calcul intensif, ...)

Terminologie de la sûreté de fonctionnement ("Dependability")

Les fautes (erreurs de conception, accidents, malveillances)

L'existence d'**erreurs de conception** ("design errors") ou **d'accidents** physiques ("physical damage") ou de **malveillances** est inévitable.

- Erreurs de conception, programmation ou de saisie.
- Accidents dus à l'environnement.
- Malveillances intentionnelles.

Les états erronés

L'une des circonstances précédentes peut ne pas gêner le système ou le gêner en conduisant à un état erroné (non prévu).

Cet état peut rester indétecté longtemps (latence de la faute) mais conduit à court ou long terme à une défaillance ou panne.

Problème: définir la faute qui à conduit à l'état erroné.

Défaillances, pannes ("Failures")

Le système est **en panne** si suite à l'un des phénomènes précédents il ne respecte pas l'une de ses spécifications.

La panne est la manifestation au niveau du service rendu d'une faute.

Systèmes de sécurité, systèmes critiques "Safety critical systems"

Le domaine d'utilisation du système est particulièrement dangereux et met en jeu des vies humaines avec des coûts liés aux pannes qui peuvent être immenses.

Domaine des transports
- Conduite automatique de trains.
- Systèmes de contrôle en avionique.

Domaine de la production d'énergie
- Conduite de centrales nucléaires.
- Conduite de barrages.

Classification des pannes

- Pannes catastrophiques
  Elles sont inacceptables

- Pannes non catastrophiques
  Elles sont acceptables.
  Notion de système à défaillance saines
  ("Failsafe")

Techniques pour la construction de systèmes sûrs

L'évitement des fautes ("Fault avoidance")

L'ensemble des techniques de conception, de fabrication qui permettent de produire des composants informatiques de très bonne qualité (très fiable).

Le composant ne doit pas tomber en panne

- bonne conception
- bonne fabrication (génie logiciel)

=> peu d'erreurs.

La tolérance aux fautes ("Fault tolerance")

L'ensemble des techniques de conception des systèmes qui continuent de fonctionner même en présence de la panne de l'un de leurs composants.

L'ensemble de l'architecture, considérée comme un tout, continue par l'utilisation de redondances de rendre le service attendu en dépit de l'existence de fautes.

Techniques pour la validation de systèmes sûrs

L'élimination des fautes

L'ensemble des techniques permettant de minimiser le nombre de fautes résiduelles présentes dans un système (par le test).

Techniques de validation, de tests
=> moins d'erreurs.

La prévision des fautes ("Fault measurement") ("Dependability evaluation")

L'ensemble des techniques de l'évaluation prévisionnelle de la sûreté de fonctionnement (de l'existence de fautes).

L'ensemble de l'architecture considérée comme un tout continue par l'utilisation de redondances de rendre le service attendu en dépit de l'existence de fautes.

Les composantes quantitatives de la sûreté de fonctionnement

- La fiabilité ('Reliability') -

R(t) = Probabilité pour qu'un système soit continûment en fonctionnement sur une période donnée (entre 0 et t).

- La disponibilité ('availability') -

D(t) = Probabilité pour qu'un système soit en fonctionnement à un instant t donné.

- La maintenabilité ('Maintainability) -

M(t)= Probabilité pour qu'un système en panne à l'instant 0 soit réparé à l'instant t.

- La sécurité ('Security')-

S(t) = Probabilité pour qu'un système soit continûment en fonctionnement non catastrophique sur une période donnée (entre 0 et t).

Grandeurs moyennes caractéristiques de la sûreté de fonctionnement

- La disponibilité asymptotique -

La disponibilité après une longue durée:

$$D^* = \lim_{t \to +\infty} D(t)$$

- Le MTTF (Mean Time To Failure) -

Le temps moyen jusqu'à la première panne (si R(t) est la fonction de fiabilité) :

$$MTTF = \int_0^{+\infty} R(t)dt$$

- Le MUT (Mean Up Time) -

Le temps moyen de bon fonctionnement avant la prochaine panne (en régime stationnaire).

- Le MDT (Mean Down Time) -

Le temps moyen en panne avant la réparation (en régime stationnaire).

- Le MTBF (Mean Time Between Failure) -

Le temps moyen entre pannes (en régime stationnaire).

$$MTBF = MUT + MDT$$

Sûreté de fonctionnement et systèmes répartis

- **La sûreté de fonctionnement** d'un système réparti doit être étudiée.

Exemple: Architecture de n sites en coopération : chaque site est indispensable

- Si le taux de panne d'un site est λ.
- Le taux de panne du système est $n\lambda$.
- Hypothèse exponentielle.

Notion de système en série: la fiabilité est le produit des fiabilités:

$$R(t) = \prod_i R_i(t)$$

- Au contraire une organisation efficace d'un système réparti (en tolérance aux pannes) atteint des niveaux de sûreté de fonctionnement qu'aucune approche d'évitement ne peut atteindre à un coût comparable.

Notion de système en parallèle:

$$R(t) = 1 - \prod_i (1 - R_i(t))$$

I

Classification des pannes

Modèle des systèmes répartis étudiés

Un composant (matériel ou logiciel) est considéré comme correct s'il se comporte **de manière consistante avec ses spécifications.**

Un modèle formel (ex: automate) décrit le comportement correct du composant. Pour toute situation on connaît :

. Un ensemble possible d'événements entrants.

. Les traitements à réaliser

. Les ensembles de messages produits en résultat

. Les contraintes de délais de réponse (dans de nombreux cas cette donnée est fondamentale).

Un composant en panne ne se comporte pas selon ses spécifications de comportement et de performances

Conséquences sur les équipements matériels

Processeurs corrects

Exécution du jeu d'instruction respecté.

Respect de l'intégrité des données en mémoire.

Temps de traitement conformes aux spécifications.

Réseau de communication correct

- Topologie quelconque permettant tous les échanges nécessaires à l'application.
- Délai de transmission des messages conformes aux spécifications.

Horloges physiques correctes

. Dérive bornée (par rapport à un temps universel utilisé à titre de comparaison)

Ex. : u, v dates universelles

c(u), c(v) valeurs lues

r dérive maximum

$$(1 - r)(u - v) < c(u) - c(v) < (u - v)(1 + r)$$

Panne franche ("Crash")

Une fois le composant en panne franche il cesse immédiatement et de façon indéfinie de répondre à toute sollicitation ou de générer de nouvelles requêtes (jusqu'à une réparation).

Une panne franche est **une panne permanente.**

Ex. : .Panne franche de processeur.
. Coupure de voie physique.
. Certains types de programmes erronés (exemple boucle)
. Système d'exploitation interbloqué.

Distinction

Système silencieux sur panne "Fail-silent"

En panne silencieuse le système ne produit plus aucune sortie.

Système stoppé sur panne "Fail-stop"

Modèles de systèmes relativement au temps

Systèmes synchrones

Idée de base

Deux systèmes ne peuvent se mettre à agir à des vitesses relatives non prévues.

. Les délais de transmission des messages sont bornés (par une valeur D).

. Il existe une borne supérieure pour le temps d'exécution d'une étape par un processus.

. Les horloges matérielles ont une dérive bornée.

Hypothèses temporelles synchrones

La réponse à une sollicitation s'effectue toujours dans un délai borné ou pas du tout.

Systèmes asynchrones

On ne connaît pas de borne au temps de réponse à une requête qui peut-être arbitraire.

Aucune hypothèse temporelle n'est formulée.

Détecteurs des pannes franches.

Les hypothèses synchrones permettent l'utilisation de délais de gardes pour la construction de détecteurs de pannes par scrutation d'un processus.

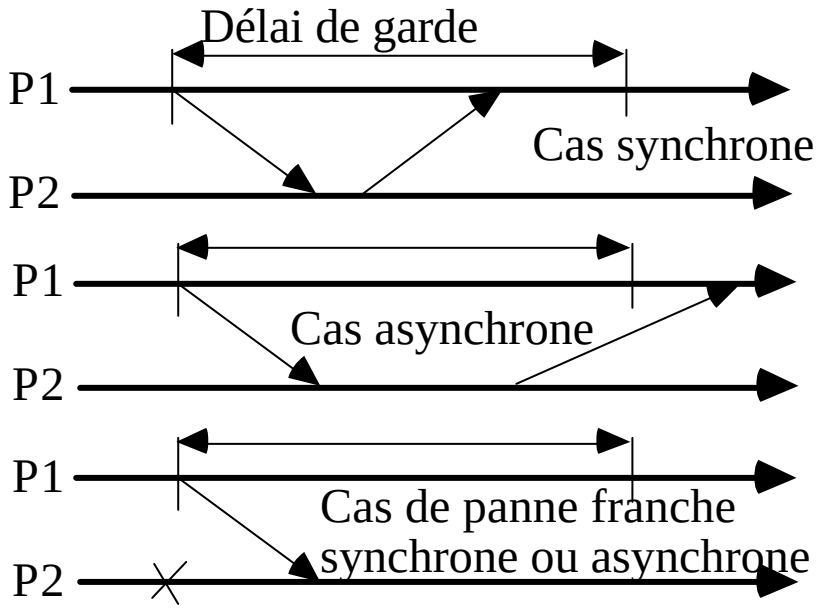
Cas des systèmes synchrones

Si le réseau ne perd pas de messages cette solution détecte les pannes franches.

Si le réseau est soumis à des pertes de messages cette solution est probabiliste.

Cas des systèmes asynchrones

On ne peut pas distinguer le cas d'un processus extrêmement lent de celui d'une panne franche.



Propriétés d'un détecteur de pannes franches.

Idée de la complétude ("Completeness")

Un processus en panne franche doit être détecté en panne.

Idée de la précision ("Accuracy")

Un processus correct ne doit pas être considéré en panne franche.

Raffinement des propriétés des détecteurs de pannes franches (Chandra et Toueg).

Complétude ("Completeness")

Complétude forte

Inévitablement tout processus en panne franche est suspecté de manière permanente par tout processus correct

Complétude faible

Inévitablement tout processus en panne franche est suspecté de manière permanente par un processus correct.

Précision ("Accuracy")

Précision forte ("Strong Accuracy")

Aucun processus correct n'est suspecté avant de tomber en panne franche.

Précision faible ("Weak Accuracy")

L'un des processus correct n'est jamais suspecté avant de tomber en panne franche.

Précision forte inévitable ("Eventual strong Accuracy")

Il existe une date au delà de laquelle aucun processus correct n'est suspecté avant de tomber en panne franche.

Précision faible inévitable ("Eventual weak Accuracy")

Il existe une date au delà de laquelle l'un des processus correct n'est jamais suspecté avant de tomber en panne franche.

Catégories de détecteurs de pannes franches selon Chandra et Toueg

Combinaison des quatre niveaux de précision et des deux niveaux de complétude.

Précision / Complétude	Forte	Faible	Inévitablement Forte	Inévitablement Faible
Forte	Parfait P	Fort S	Inévitablement Parfait ◇ P	Inévitablement Fort ◇ S
Faible	Q	Faible W	◇ Q	Inévitablement Faible ◇ W

Avant Chandra et Toueg

Définition de problèmes de sûreté de fonctionnement résolus sous hypothèses synchrones ou asynchrones.

Après Chandra et Toueg

Définition de problèmes de sûreté de fonctionnement résolus sous hypothèses de fonctionnement des détecteurs de pannes.

Panne transitoire ou intermittente "Transient, Intermittent, Omission failure"

En réponse à un événement en entrée un composant ne délivre **jamais** la réponse attendue (le composant ne peut fournir son service habituel pendant une certaine période => perte de quelques données).

Ultérieurement il peut fonctionner à nouveau de façon correcte.

Il n'y a pas déviation par rapport aux spécifications sur ces autres réponses.

Distinction possible

Panne transitoire

Apparaît une seule fois puis disparaît.

Panne intermittente

Apparaît plus ou moins périodiquement.

Exemples. :

. Perte de messages dues au bruit.

. Destruction de message pour éviter la congestion.

. Destruction d'activités pour éviter l'interblocage ou les problèmes de contrôle de concurrence.

Panne temporelle
"Timing, Performance failure"

. Une sortie correcte associée à une requête entrante se manifeste de façon incohérente avec les spécifications temporelles.

- Trop tard.

- Trop tôt.

Le cas le plus fréquent est celui d'une manifestation trop tardive.

Ex. : . Surcharge d'un processeur.
. Horloge trop rapide.
. Délai de transmission trop long.

Problème traité dans les cours de **temps réel** (stratégies d'ordonnancements temps réel) ou dans les cours de **réseaux à QOS** (respect de latence ou de gigue).

Panne quelconque ou byzantine ("Malicious, byzantine Failures")

Tout comportement s'écartant des spécifications (principalement en ce que les résultats sont non conformes) est qualifié de comportement byzantin (Lamport).

On distingue quelquefois :

a) Fautes byzantines "naturelles"

Ex : . Erreur physique non détectée (sur une transmission de message, en mémoire, sur une instruction).

. Erreur logicielle amenant une non vérification des spécifications.

b) Fautes byzantines "malicieuses"

Ex : . Comportement visant à faire échouer le système (sabotage, virus ....).

Classification complémentaire des pannes byzantines

- **La signature des mes**sages et la vérification de signature (authentification et intégrité) entraîne une résistance aux pannes byzantines bien meilleure (surtout pour ce qui concerne les modifications quelconques qui pourraient être effectuées sur les messages du fait de la transmission via des sites malicieux).

- **De manière plus générale** l'usage des fonctions cryptographiques simplifie les protocoles de communication tolérant les pannes byzantines.

On distingue donc parfois:

1) **La classe des fautes byzantines** précédemment décrites (pour lesquelles les communications sont **non authentifiées**).

2) **La classe des fautes byzantines** qui apparaissent malgré les signatures ("pannes byzantines **authentifiées**").

Hiérarchisation des classes de pannes

Les 4 classes sont hiérarchisées.

Panne franche:

Pas de réponse à une entrée

=> Panne transitoire

Panne transitoire:

Un délai de réponse infini.

=> Panne temporelle

Panne temporelle:

Non respect d'une échéance (spec)

=> Panne quelconque

Tolérance à une classe de pannes

- On réalise des composants pour tolérer l'une des classes de pannes précédentes.
- Restent non tolérées les pannes de la classe supérieure.
- Cas les plus fréquents:

Tolérance aux pannes franche

Tolérance aux pannes d'omission.

II

Les différentes catégories

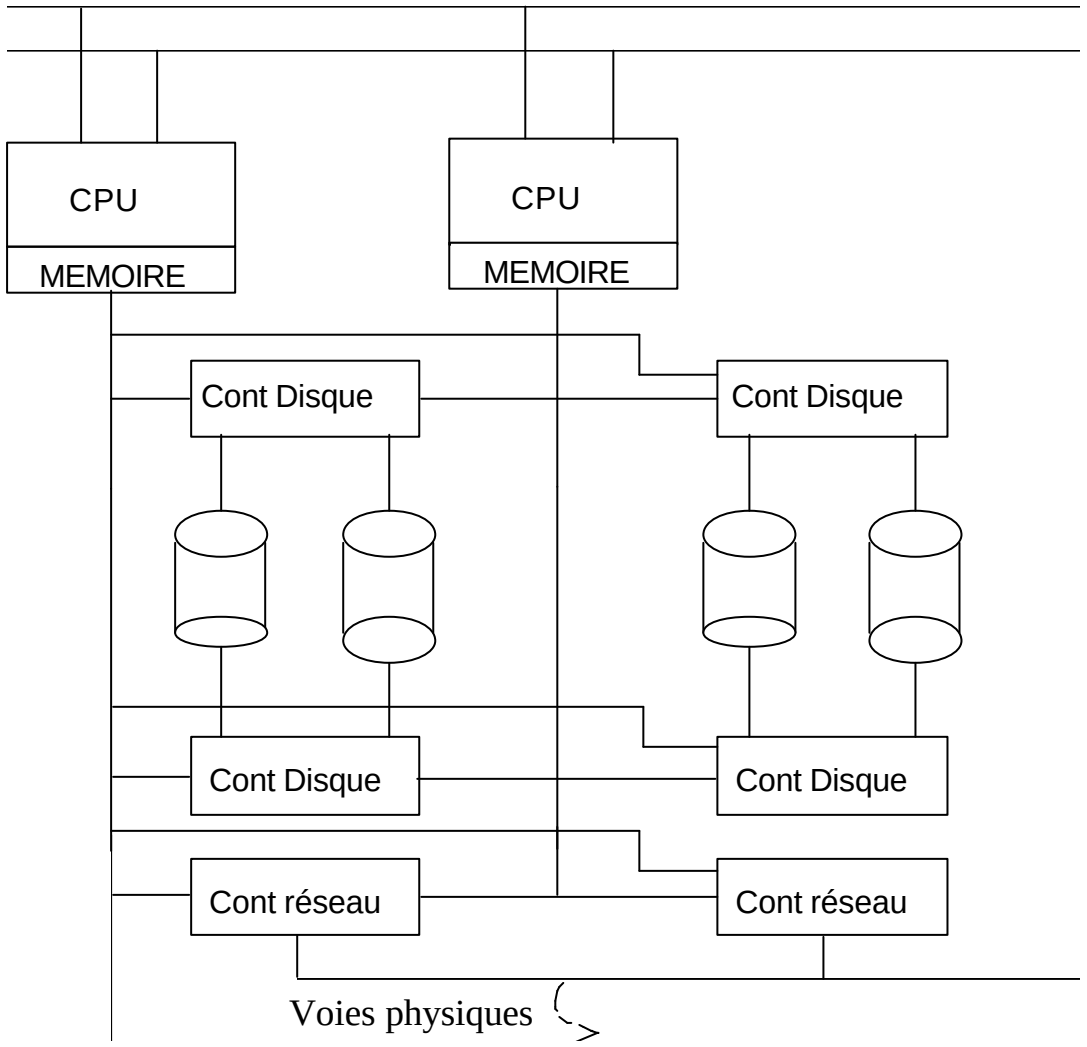
de redondance

Rappel : Architectures à redondances matérielles

Exemple type : Architecture TANDEM

- Tolérance à une panne franche matérielle destinée aux applications de gestion

- Système orienté disponibilité



Les différents types de redondances

Nombreuses propositions
Nombreux points de vue

Techniques de recouvrement d'erreurs ("Error recovery")

1) Existence d'un détecteur de panne.
2) D'un état erroné on peut retrouver un état correct puis délivrer un service correct.

Reprises arrières
Redondances temporelles
("Backward Recovery")

- Pour un composant soumis à des pannes transitoires il est courant de tenter de corriger cette panne par un nombre fixé de tentatives successives.

Nécessite la pose de points de reprise.

Z Cette technique est utilisée assez systématiquement pour les serveurs uniques et ce n'est qu'après l'échec de cette approche que l'on déclare une panne non temporaire.

Reprises avants / Traitement des exceptions / Poursuite ("Forward Recovery")

- Pour un composant soumis à des pannes il est courant de tenter de traiter cette panne en recherchant un état de cohérence du système n'ayant jamais existé (futur) ou n'ayant pas existé dans un passé récent (qui s'apparenterait à une reprise).

La reprise avant évite la pose de points de reprise mais nécessite de déterminer l'étendue des dommages causés aux système par la faute jusqu'à la détection de la défaillance.

Exemple : traitant/récupérateur d'exception

Techniques de compensation ou de masquage d'erreurs

Un état erroné comporte des redondances permettant au moyen d'un algorithme rapide de délivrer un service correct.

Redondances de données

- Pour une donnée soumise à des erreurs (stockage, transmission) il est d'usage de rajouter des informations de redondance selon un code correcteur d'erreur qui permet de corriger certaines erreurs.

Redondances spatiales ou redondances de groupes

- Un groupe de serveurs redondants g en redondance spatiale est conçu pour tolérer la panne de certains de ses membres.

En dépit de la panne à un certain niveau de certains membres de g, le service offert du point de vue global par g continue en masquant le niveau de panne visé.

Par exemple on observe des pannes quelconques si l'on masque les pannes temporelles, transitoires et franches.

- Éventuellement certaines performances sont dégradées.
- Certains membres du groupe g clairement détectés en panne sont isolés et retirés du groupe de serveurs redondants.

=> on reconfigure le groupe

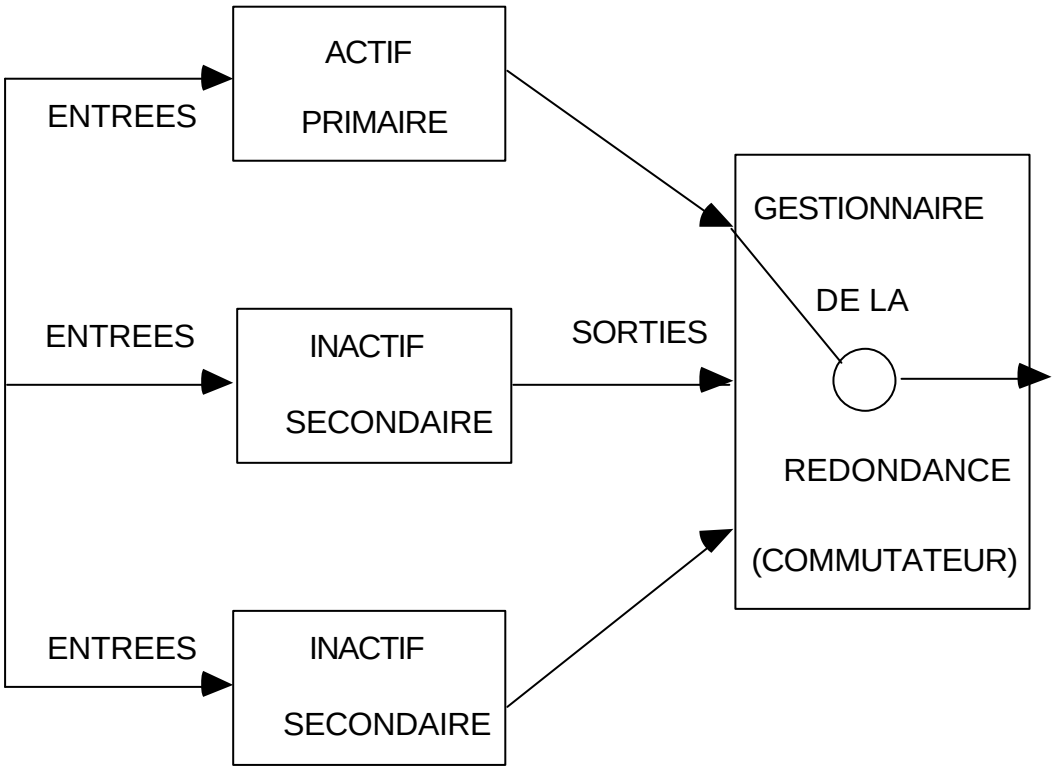
=> jusqu'à ce que la reconfiguration qui maintienne un service acceptable devienne impossible.

Différentes redondances spatiales

Redondance passive ("Standby redundancy") ("Primary backup")

Objectif poursuivi : tolérance aux pannes franches de calculateurs.

- Un seul des composants réalise effectivement les traitements et est affecté aux sorties (le primaire).
- En cas de panne du primaire l'un des calculateurs inactifs (secondaire) est sélectionné et activé pour prendre en charge le service.



Problèmes de synchronisation en redondance passive

- Problème de détection de panne du primaire.
- Problème de détermination d'un nouveau primaire parmi les alternants.
- Pour le nouveau primaire il faut reconstituer un contexte d'exécution correct.

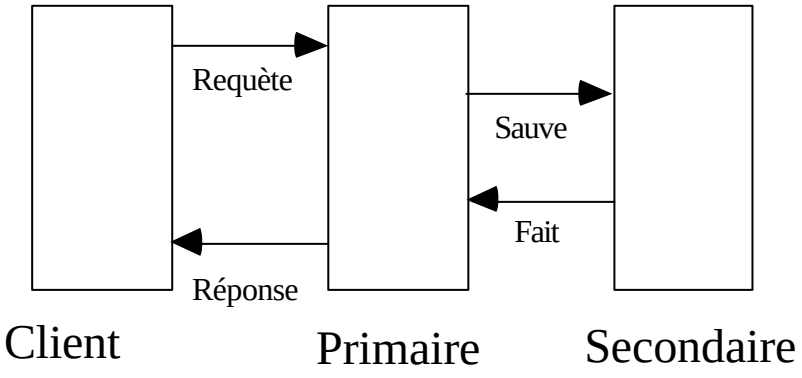
Solutions possibles

. Recopie périodique d'informations de reprise constituées par le primaire pour les secondaires.

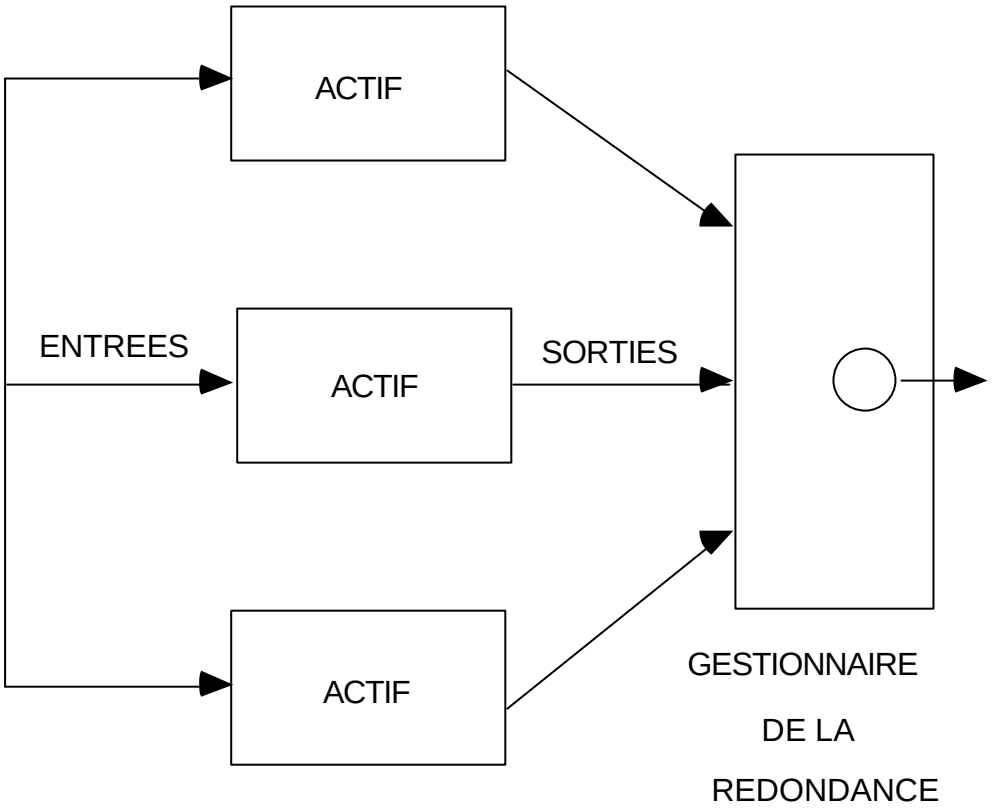
Périodes statiquement prédéterminées

Points de reprise applicatifs.

. Rééxécution des services fournis depuis le dernier point de reprise.



Redondances actives ou dynamiques ("Active redundancy")

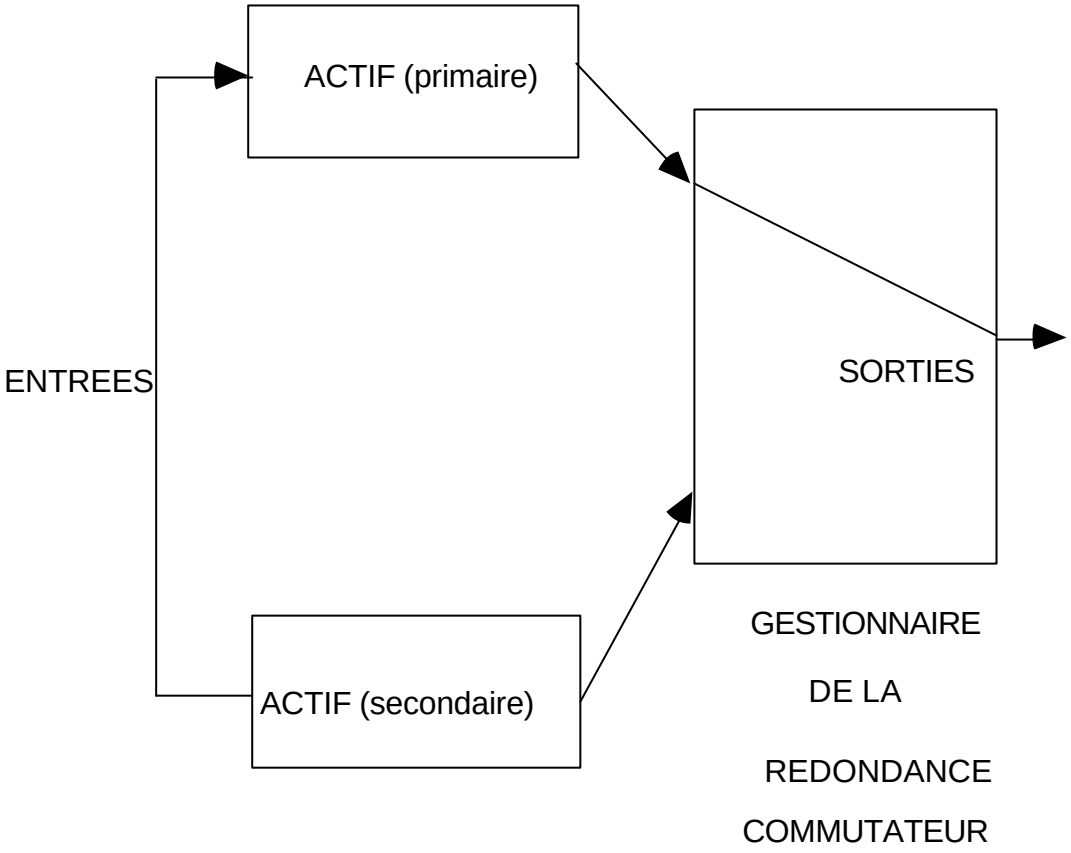


- Dans la redondance active tous les composants réalisent les traitements.

- Le gestionnaire de la redondance traite les sorties pour tolérer différentes classes de panne des serveurs.

Redondance sélective active

Tolérance des pannes franches



- Un seul serveur est affecté aux sorties

- En cas de panne du primaire le secondaire prend le contrôle (avec le contexte d'exécution complet de l'activité).

Problèmes de synchronisation en redondance sélective active

- Tout le monde doit recevoir les entrées en diffusion.

- Lors d'un basculement, l'alternant actif doit être réélu.

- Lors d'un basculement, le contexte de l'alternant actif doit être cohérent avec celui laissé par l'ancien actif: besoin d'une technique pour traiter dans le même ordre et exhaustivement les mêmes données.

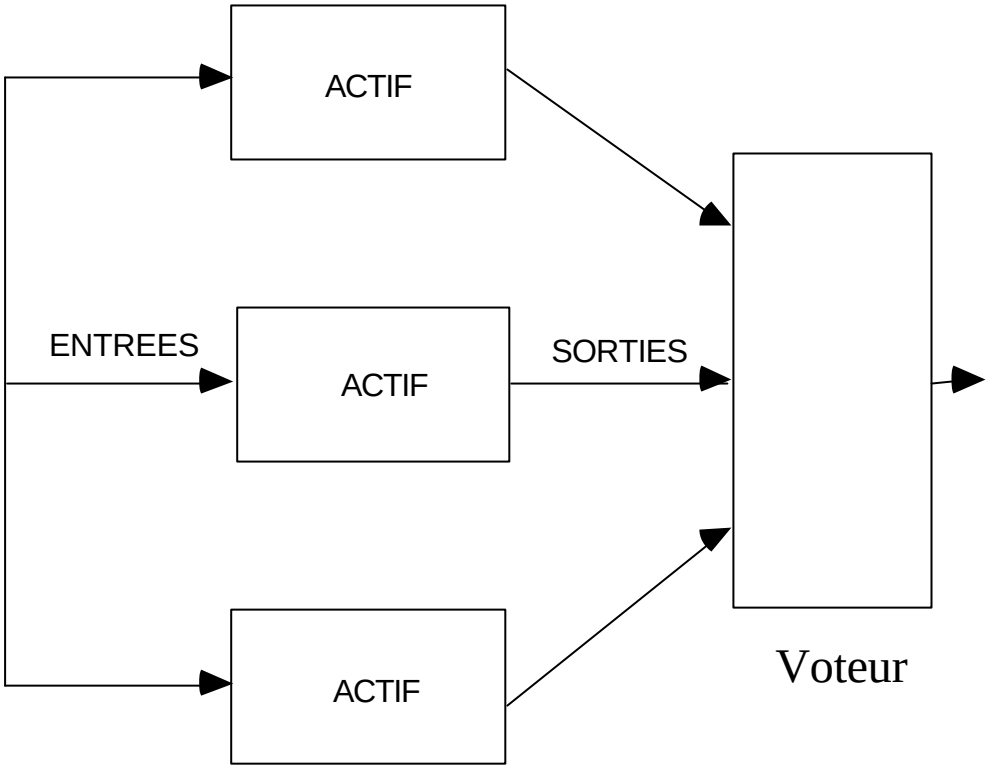
Diffusion ordonnée totalement.

Remarque :

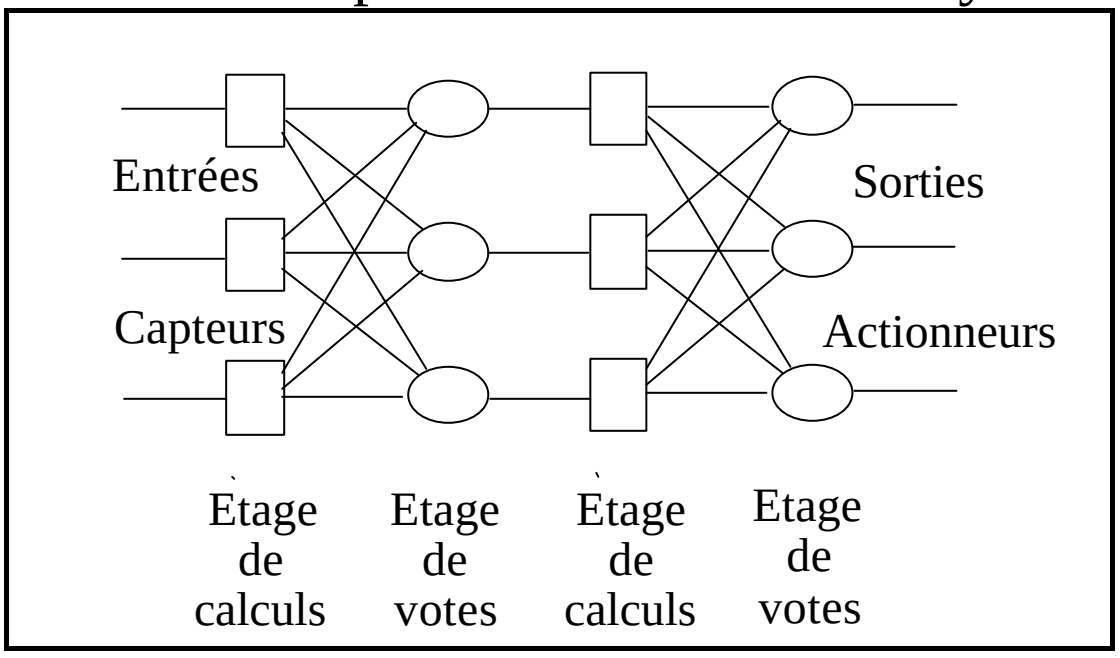
Le gestionnaire de la redondance peut-être plus complexe qu'un simple commutateur. Quand les deux composants sont actifs il peut choisir d'utiliser les résultats de l'un ou de l'autre selon les sites de résidence.

Redondance massive

Tolérance des pannes quelconques.



TMR "Triple Modular Redundancy"



Système TMR avec réplication des voteurs

Problèmes de synchronisation en redondance massive

- Tout le monde doit recevoir les entrées en diffusion dans le même ordre pour traiter les données dans le même ordre et fournir les résultats dans le même ordre aux voteurs.

- La synchronisation entre les productions de résultats doit permettre la réalisation du vote majoritaire.

Remarques:

Si l'on veut simplement tolérer des pannes temporelles on peut prendre la réponse disponible le plus rapidement.

Si l'on veut tolérer des pannes quelconques on doit voter pour exclure aussi bien les sorties non majoritaires que celles qui apparaissent trop tardivement.

Tolérance aux pannes logicielles

Si les calculateurs redondants ont la même panne le système de redondance massive ne fonctionne pas (de même que les solutions de reprise arrière).

Notion de **panne de mode commun**

Typiquement les pannes logicielles.

Solution: la **diversification** (fonctionnelle)

"N-Version programming"

- Diversification/isolement des équipes
  Spécifications détaillées.
  Développement des codes.
- Diversification des processeurs
- Diversification des systèmes.
- Diversification des compilateurs.

Variantes

- Dans le cadre des techniques de redondances temporelles

("**recovery blocks**" Randell)

- Dans le cadre des techniques de redondance massive.

Solutions coûteuses et difficiles à réaliser

III

Les problèmes fondamentaux des systèmes répartis tolérants les pannes

Rappel des différentes étapes d'un mécanisme de tolérance (1)

Les principaux mécanismes de la tolérance reposent sur l'existence de **services redondants** dont les éléments peuvent tomber en panne.

a) Il faut tout d'abord détecter les pannes d'un ou plusieurs serveurs.

b) Si l'on peut formuler une hypothèse de panne transitoire et si les contraintes de l'application sont compatibles avec les techniques de recouvrement d'erreur (pas trop temps réel fortement contraint) il faut appliquer une technique de rééxécution.

c) Si les redondances temporelles sont insuffisantes il faut mettre en place des redondances spatiales.

=> décider de la suite des situations d'appartenance au groupe de serveurs redondants.

Les situations successives résultent:

- des pannes des membres
- des réinsertions de composants ou des adjonctions de serveurs nouveaux.

Rappel des différentes étapes d'un mécanisme de tolérance (2)

d) Il faut assurer des transmissions fiables vers des groupes de composants redondants (diffusion d'informations de chacun des clients vers le groupe de serveurs redondants implémentant un service).

Ces diffusions doivent être réalisées sous les différentes hypothèses de pannes et satisfaire des propriétés d'ordre.

e) Pour la redondance massive il faut rechercher un consensus sur une valeur calculée n fois (vote réparti).

f) Il faut éventuellement tolérer les pannes temporelles (satisfaction de contraintes temporelles pouvant être sévères) => l'ordonnancement temps réel réparti des tâches.

Point essentiel pour la détection des pannes temporelles et pour la satisfaction des contraintes temps réel réparti:

=> l'existence d'une **synchronisation des horloges** entre les différents sites.

Rappel des différentes étapes d'un mécanisme de tolérance (3)

g) Si la programmation de l'application organise des données réparties partagées sur différents sites il faut assurer le **contrôle de l'accès concurrent** aux données (maintien de la cohérence) pour des **données dupliquées**.

h) Si la programmation de l'application comporte encore des sites centraux (redondances sélectives) il faut prévoir la **défaillance de ces serveurs**.

D'ou une liste de problèmes types à résoudre pour une algorithmique répartie de la tolérance aux pannes

LA DÉTECTION DE COMPORTEMENT FAUTIF

Notion de **composant autotestable**:
un composant qui incorpore son propre logiciel de diagnostic => qui fait passer le plus vite possible un composant de l'état de **faute latente** à l'état de **faute détectée**.

Existence de très nombreuses techniques

Quelques exemples

- Utilisant des **programmes de test**

Détection **hors ligne** (diagnostics).
Détection **en ligne** (détection continue).
Chien de garde=>surveillance mutuelle.

- **Détection des erreurs de données**

Codes détecteurs d'erreur.
Assertions/Tests d'acceptance.
Vote entre plusieurs alternants.

- **Détection des erreurs d'enchaînement**

Protection (anneaux, domaines).
Observation de points spécifiques de l'exécution => comparaison à un référentiel.
Signature de séquences
=> comparaison à une tabulation des séquences valides.

REPRISE ARRIERE

Problème classique de l'univers réparti.

Pour une application coopérative faisant intervenir une architecture quelconque de clients et de serveurs (éventuellement redondés):

=> Comment déterminer des points de reprise "cohérents" à une fréquence optimale.

=> Comment assurer si nécessaire la journalisation des messages en transit sur les canaux de communications.

=> Comment effectuer la reprise arrière en cas de panne détectée.

Difficile pour obtenir des performances satisfaisantes

PROTOCOLE D'APPARTENANCE A UN GROUPE

Objectif : Assurer que tous les usagers ayant à connaître la situation d'un groupe de serveurs atteignent un consensus sur la composition du groupe.

- La perception de cette composition est relative à des **communications de groupes** (le protocole d'appartenance à un groupe est en général utilisé conjointement avec un protocole à diffusion).

- Dans un tel protocole on admet souvent que le temps est divisé en **époques** ou la composition du groupe est fixe et identique pour tout le monde.

- A l'intérieur d'une même époque les communications en diffusion atteignent la même liste de processeurs ou échouent ce qui peut advenir en période de changement de liste.

LA DIFFUSION ET LE CONSENSUS

Objectif : Ces deux problèmes précédents sont des variantes peu différentes consistant à faire s'accorder différents composants d'un groupe sur une valeur

- Valeur diffusée par un site.
- Valeur votée après un calcul en redondance massive.

- La valeur est utilisée comme **un signal** pour déclencher un traitement (valeur binaire)

Exemple du problème de validation dans la mise à jour des données ("commit")

- La valeur est utilisée comme **une donnée quelconque**.

Exemple du problème de redondance massive sur des données calculées comme des valeurs à envoyer sur des actionneurs.

LE PROTOCOLE DE SYNCHRONISATION D'HORLOGES

Objectif : Assurer que des horloges situées sur des sites distincts fournissent une datation absolue des événements avec une incertitude définie.

On distingue dans ce contexte deux sous-problèmes :

- Assurer que différents sites arrivent à **démarrer avec la même heure absolue** (au même moment avec une incertitude connue).

- Maintenir aussi longtemps que nécessaire **les différentes horloges** dans une variation relative connue.

. En contrôlant la dérive relative

Solutions par échange de messages.

Solutions par asservissement sur une horloge hertzienne.

LE PROTOCOLE D'ÉLECTION

Objectif : Lorsqu'une solution à un problème est basée sur l'existence d'un site **coordinateur unique** la panne du coordinateur doit être tolérée.

Cas des solutions en redondance sélective passive et sélective active.

Le protocole d'élection vise à désigner un et un seul coordinateur remplaçant.

CONCLUSION

Architecture de systèmes répartis en vue de la tolérance aux pannes
Nécessité de fournir deux catégories de services.

1 Les services standards utilisés dans des systèmes répartis: micro-noyaux, systèmes d'objets répartis

- Gestion des ressources (mémoire, processeur/ordonnancement..)
- Désignation, liaison
- Création des objets, migration,
- Réalisation des interactions de base (IPC, RPC légers/distants)
- Synchronisation **…** etc

La sûreté n'est pas leur objectif principal

La sûreté/tolérance aux pannes de ces algorithmes est fondamentale car ils sont utilisés dans un système dont la sûreté doit être excellente.

2 Algorithmes utilisés dans les systèmes tolérants les pannes pour la tolérance.

Exemples vus:

Détection de panne
Reprises arrière
Élection
Diffusion fiable
Gestion des groupes
Vote réparti
Synchronisation d'horloges
Copies multiples
etc....

- La tolérance aux pannes des solutions présentées dans un système réparti pour la sûreté de fonctionnement est un problème essentiel.

- Certains de ces algorithmes doivent être étudiés pour tolérer tout type de panne.

Bibliographie

1 R. Strong "Problems in fault-tolerant distributed systems", Publication IEEE ISBN 135-/85/0000/0300s01.00

2 F. Christian, "Issues in the design of highly available computing systems", IBM Research Report RJ 5856 (58907) 7/10/1987

3 F. Christian, "Questions to ask when designing or attempting to understand a fault-tolerant distributed system", IBM Research Report RJ 6980 (66517) 4/8/1989

4 J.C. Laprie, B. Courtois, M.C. Gaudel , D. Powell "Sûreté de fonctionnement des systèmes informatiques matériels et logiciels", AFCET Dunod Informatique 1989