RenPar'15 / CFSE'3 / SympAAA'2003
La Colle sur Loup, France, 15 au 17 octobre 2003

Pérennité dans les systèmes de stockage pair-à-pair*

Antoine Vernois

École Normale Supérieure de Lyon,
LIP, 42 allée d'Italie,
69007 LYON - France
antoine.vernois@ens-lyon.fr

Résumé
Dans les systèmes de stockage pair-à-pair, la pérennité des données est un enjeu important. En effet, lorsque la durée de vie des noeuds est courte (quelques heures), il devient nécessaire d'être fortement tolérant aux pannes. Pour cela, les systèmes existants se basent sur des mécanismes de redondance des données. Dans cet article, après avoir rapidement présenté plusieurs systèmes pair-à-pair de partage et de stockage de données, nous présentons des mécanismes de redondance permettant d'améliorer le temps de survie des données. Une analyse quantitative de ces mécanismes nous permet de voir l'impact de leur mise en œuvre sur la pérennité des données mais aussi de quantifier les coûts de réparation qu'ils induisent pour assurer l'intégrité des données.

Mots-clés : pair-à-pair, stockage, redondance, fiabilité, distribué

---

1. Introduction

Aujourd'hui les systèmes pair-à-pair sont très largement utilisés pour partager des ressources à travers l'Internet. Certains ont pour but de partager du temps CPU (Seti@Home, XtremWeb, Entropia...), d'autres de mettre des données à disposition d'autres utilisateurs (Gnutella, Freenet, KaZaa, ...). Parallèlement, des systèmes pair-à-pair dédiés au partage d'espace disque ont vu le jour tel Intermemory ou Oceanstore. Leur but premier est d'offrir un accès transparent à un service de stockage distribué. Cette dernière catégorie de systèmes partage de nombreuses bases communes avec les autres, comme les mécanismes de découverte de réseau ou de localisation de ressource. Mais si les systèmes de partage de données ou de CPU peuvent assez facilement faire face à la disparition d'un noeud, en relançant un calcul sur un autre noeud par exemple, pour un système de stockage, la défaillance d'un noeud implique la perte définitive des données qu'il stockait. Ainsi, il est indispensable de mettre en place des méthodes de redondance dont le rôle est d'assurer la pérennité des données.
Cet article propose une étude théorique sur l'efficacité des mécanismes de redondance pour le projet US (Underground Storage) qui a pour objectif de définir un système de stockage pair-à-pair dans le cadre de l'ACI GRID CGP2P. Après une rapide présentation des systèmes existants, nous nous intéresserons à la manière de mettre en oeuvre des mécanismes de redondance afin d'assurer la pérennité des données. Enfin, nous présenterons une analyse quantitative de l'efficacité de ces mécanismes sur la persistance des données, ainsi que des coûts qu'ils induisent.

2. État de l'art

Nous allons présenter succintement quelques systèmes pair-à-pair. D'un point de vue de la survie des données, nous pouvons distinguer 2 catégories : les systèmes pair-à-pair dont le but est de permettre la diffusion de documents à travers l'internet, et les systèmes pair-à-pair dédiés au stockage intégrant des mécanismes pour assurer la pérennité des données.

* Ce travail fait parti du projet US (http://www.ustorage.net) supporté par l'ACI GRID CGP2P du MENRT

Les systèmes pair-à-pair se caractérisent principalement par la mise en place d'un "sur-réseau" basé sur des connections point-à-point formant un graphe faiblement connexe dont l'objectif principal est d'offrir des mécanismes de localisation et de routage.

Depuis l'apparition de Napster, le précurseur, de nombreux systèmes pair-à-pair de diffusion et partage de documents ont vu le jour. Gnutella[9] est l'un d'eux et pose les spécifications d'un protocole de partage de documents. La recherche de documents dans le système est effectuée par inondation du graphe de connections. Freenet[4, 3] est un projet dont l'ambition est de garantir l'anonymat des utilisateurs. Il propose des mécanismes de cryptographie, de routage auto-adaptatif pour la recherche de documents ainsi qu'une méthode de réplication dont le but est d'assurer la survie et une meilleure disponibilité des données les plus populaires.

Pour les systèmes pair-à-pair dédiés au stockage de données, le précurseur est Intermemory[2, 6]. Il intègre un mécanisme de protection des données par fragmentation et réplication. Dans PAST[5, 10], noeuds et fichiers possèdent un identifiant unique. Les fichiers sont placés sur le noeud dont l'identifiant est le plus proche de celui du fichier. PASTRY est le système de routage et de localisation de données tolérant aux fautes mis en oeuvre dans PAST. En ce qui concerne les données, la tolérance aux pannes s'effectue par réplication des données. Oceanstore[1, 11] est un vaste projet de stockage pérenne de données, probablement le plus abouti. Les données sont stockées sur des serveurs dédiés ayant une forte connectivité et une large bande passante situés, par exemple, chez les fournisseurs d'accès à Internet ou au niveau des backbones. Dans Oceanstore, le routage et la localisation des données sont assurés par Tapestry[8, 12]. Oceanstore gère également l'existence de multiples versions d'un même fichier ainsi que la tolérance aux pannes assurée par de la redondance.

Techniques de routage et méthodes de redondance sont deux éléments qui caractérisent les systèmes de stockage pair-à-pair. On trouve deux systèmes primitifs : Gnutella dans lequel les documents restent chez les propriétaires et qui peut être vu comme un Napster décentralisé et Intermemory qui ne possède pas de mécanisme de routage mais un répertoire global. Pour ce qui est du routage des requêtes, une forme générale se dégage : les requêtes sont dirigées suivant un identifiant unique qui caractérise un fichier. Chaque noeud est spécialisé pour un certain identifiant de fichier. Dans PAST et Oceanstore, les noeuds ont une connexion stable et un identifiant unique. Dans PAST, le fichier est stocké sur le noeud d'identifiant le plus proche tandis que, dans Oceanstore, ce n'est qu'un lien vers le noeud de stockage. Ces deux systèmes garantissent l'aboutissement de la recherche.

Freenet est basé sur des noeuds plus volatiles n'ayant pas d'identifiant. Par le jeu des insertions et des recherches, certaines parties du réseau se spécialisent dans certains types de fichiers par l'adaptation dynamique de la table de routage des noeuds. Il n'y a cependant pas la connaissance globale de cette spécialisation et certains noeuds peuvent ne pas trouver un fichier demandé. Pour la gestion de l'espace de stockage, Freenet se comporte comme un cache, lorsqu'un noeud n'a plus de place, les données les plus anciennement accédées sont supprimées. Dans Oceanstore et Intermemory, les mécanismes d'indirection permettent de placer les fichiers sur n'importe quel noeud disposant d'un espace suffisant. Le système statique de PAST complique l'équilibrage de charge car les fichiers doivent être stockés sur le noeud dont l'identifiant est le plus proche de celui du fichier.

En ce qui concerne la pérennité, Freenet n'offre aucune garantie. Dans PAST, elle est assurée par un système de réplication simple. Pour Intermemory et Oceanstore, les garanties sont basées sur des systèmes plus sophistiqués à base de redondance et fragmentation que nous étudierons plus en détail dans la suite.

3. Les mécanismes de la redondance

La redondance est la clé de la pérennité des données dans un système de stockage distribué. L'exemple le plus simple de redondance est celui de la réplication des données, c'est-à-dire que l'on se contente de recopier les données telles quelles sur une ou plusieurs machines comme dans PAST ou Freenet. On notera r le facteur de réplication. Pour $r = 1$, un seul réplicat, la tolérance est donc de 1 noeud, pour $r = 10$, il sera de 10 noeuds.

Dans un système basé sur la fragmentation, les blocs de données de taille B sont découpés en s blocs de taille $\frac{B}{s}$. A partir de ces s fragments, on détermine r fragments de redondance eux aussi de taille $\frac{B}{s}$. Les $s + r$ fragments sont tels qu'à partir de s fragments quelconques, il est possible de reconstruire le bloc

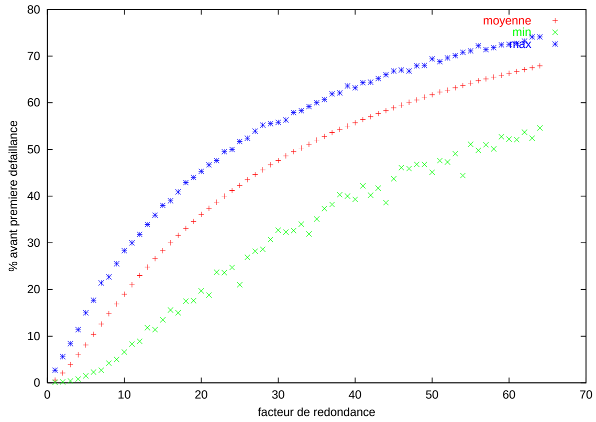

FIG. 1 – Tolérance aux pannes en fonction du facteur de redondance r dans le cas où $s = 16$, le réseau est constitué de 1000 noeuds et 250000 blocs stockés

initial. Ces $s + r$ fragments seront bien évidemment placés sur $s + r$ noeuds différents.

Le ratio $\frac{s}{s+r}$ détermine l'espace utile c'est-à-dire le rapport entre la taille de la donnée et l'espace de stockage. À tolérance équivalente, la redondance par fragmentation permet de gagner en espace utile par rapport à la simple réplication.

Pour déterminer ces fragments de redondance, une méthode couramment utilisée est l'emploi du codage de Reed-Solomon[7]. Considérons un bloc de données D comme un vecteur $D = (d_j)_{j \in [1..s]}$ de dimension s où les d_j sont les fragments. Le vecteur $E = (e_i)_{i \in [1..s+r]}$ des fragments à disperser sur les noeuds est défini comme le produit d'une matrice A de dimension $(s + r) \times s$ avec D, où la matrice A est telle que s vecteurs lignes quelconques soient linéairement indépendants : $E = AD$. Grâce à la propriété de A, à partir de s éléments de E et les s vecteurs lignes de A correspondants, il est possible de reconstruire un fragment perdu. La matrice A peut être obtenue, par exemple, à partir d'une matrice de Vandermonde ou de Cauchy dans un corps de Galois (avec un nombre fini d'éléments).

La figure 1 représente le pourcentage de noeuds défaillants avant la première perte d'un bloc de données en fonction du facteur de redondance dans le cas d'un réseau constitué de 1000 noeuds dans lequel 250000 blocs sont stockés et $s = 16$. La courbe du milieu représente la moyenne sur 100 simulations et les deux autres le minimum et le maximum obtenus au cours de ces mêmes simulations. Nous pouvons y voir que le gain apporté par la redondance augmente rapidement au début. Par exemple, pour $r = 1$, la tolérance est proche de 0% alors que pour $r = 16$ il est déjà, en moyenne, à 30%. Par contre, la tolérance croît de moins en moins avec l'augmentation de r. En effet, augmenter le facteur de redondance de 1, revient à multiplier ce facteur de redondance par $\frac{r+1}{r}$. Plus r augmente, plus ce coefficient est faible, donc moins l'effet de l'augmentation du facteur de redondance est important.

Toute augmentation du facteur de redondance augmente l'espace occupé par un bloc dans le système. Nous nous sommes donc penchés cette fois sur la tolérance aux pannes en fonction de l'espace utile. Cet espace représente le volume de données utilisateur qui pourra être inséré dans le système. Il s'agit d'un pourcentage de l'espace totale de stockage.

Pour calculer le pourcentage avant la première défaillance en fonction de l'espace utile, nous avons fixé la valeur de s et fait varier r, ce qui a pour conséquence de faire varier l'espace utile. La figure 2 représente le résultat de cette simulation pour trois valeurs de s. Nous constatons que bien évidemment la tolérance augmente avec la diminution de l'espace utile (qui correspond à une augmentation du facteur de redondance).

Mais nous voyons aussi que s influe sur la tolérance : pour un même espace utile, il semble qu'une grande valeur de s soit préférable. En effet, dans ce cas, une plus grande valeur de s implique une plus grande valeur de r donc une plus grande tolérance aux pannes. Il ne faut cependant pas oublier que s est le nombre de fragments nécessaire pour reconstruire le bloc, ainsi les coûts de reconstruction

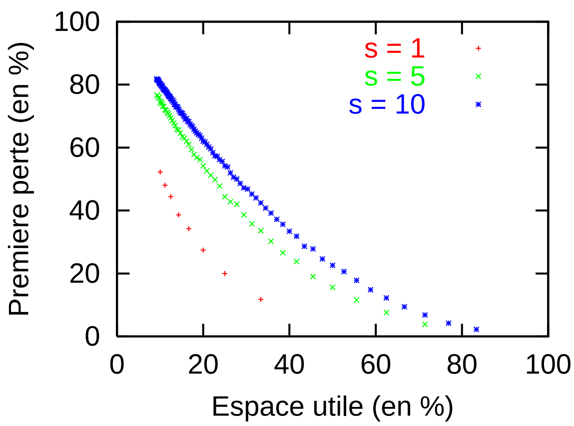

FIG. 2 – Tolérance aux pannes en fonction de l'espace utile r

augmentent avec s. Il ne faut donc pas choisir une valeur trop grande.

4. Les mécanismes de la réparation

Les mécanismes de redondance permettent de tolérer un certain nombre de défaillance, mais dans un environnement fortement dynamique, un mécanisme statique ne peut pas assurer convenablement la pérennité des données.

4.1. Durée de vie des noeuds

Considérons que la durée de vie des noeuds suit une loi de probabilité exponentielle de paramètre τ.

$$P(X < t) = (1 - e^{-\frac{1}{\tau}t}) \tag{1}$$

Supposons maintenant que tous les noeuds soient indépendants, la probabilité $P_k(X < t)$ que k noeuds soient en panne dans le temps t :

$$P_k(X < t) = (1 - e^{-\frac{1}{\tau}t})^k \tag{2}$$

Le MTTF (Mean Time To Failure) de k noeuds peut alors se calculer comme suit :

$$MTTF_k = \frac{1}{\tau}k\int_0^{+\infty} t * (1 - e^{-\frac{1}{\tau}t})^{k-1} * e^{-\frac{1}{\tau}t}dt \tag{3}$$

La figure 3 représente la durée de vie de k noeuds dans le cas où le paramètre de la loi exponentielle est $\tau = 1$. Nous pouvons constater que la vitesse de disparition des noeuds est assez rapide. Donc, bien que l'on soit assez résistant, les défaillances surviennent rapidement et donc la perte de données arrivera très vite si l'on n'agit pas suffisamment vite.
Pour assurer la pérennité, il est donc nécessaire d'introduire un mécanisme de réparation qui détecte et reconstruit les fragments de données perdus. Dans ce qui suit nous allons étudier comment évolue le MTTF lorsque des réparations sont effectuées. Pour cela nous devons poser quelques hypothèses. Tout d'abord, nous supposons que lorsqu'un noeud est défaillant nous disposons de suffisamment d'espace libre dans le système pour pouvoir reconstruire les données manquantes et les affecter à un nouveau noeud. Nous considérons que la durée de vie des noeuds suit une loi exponentielle de temps moyen τ. Enfin, nous supposons disposer d'un mécanisme qui vérifie le bon fonctionnement des noeuds à intervalle régulier et effectue les réparations nécessaires en un temps inférieur à $\epsilon.\tau$. C'est à dire que si une panne survient à l'instant t, au temps $t + \epsilon.\tau$ celle-ci aura été détectée et réparée.

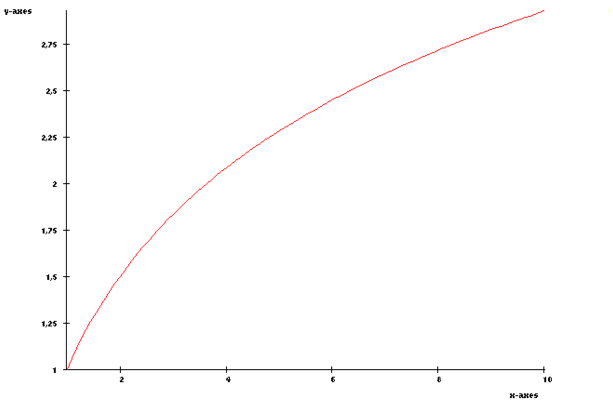

FIG. 3 – Temps de disparition de k noeuds pour $\tau = 1$

La probabilité qu'un noeud soit toujours actif après un temps t est :

$$p(X > t) \quad = \quad \int_{t}^{+\infty} \frac{1}{\tau} e^{-\frac{1}{\tau}y} dy \tag{4}$$

La probabilité p qu'un fragment survive après un temps $\epsilon.\tau$:

$$\begin{aligned} p(X > \epsilon.\tau) &= \int_{\epsilon.\tau}^{+\infty} \frac{1}{\tau} e^{-\frac{1}{\tau}y} dy && (5) \\ &= e^{-\epsilon} && (6) \end{aligned}$$

La probabilité p_b qu'un bloc soit encore en vie après l'écoulement d'une période dépend donc de la probabilité qu'il reste au moins s fragments du bloc après le temps $\epsilon.\tau$. La probabilité que exactement k noeuds soient en vie après le temps $\epsilon.\tau$ correspond au nombre de façon d'arranger ces k noeuds dans l'ensemble des noeuds multiplié par la probabilité que k noeuds soient en vie et par la probabilité que les $(s + r) - k$ fragments du blocs aient disparu. Ainsi, la probabilité p_b est la somme des probabilités que exactement m noeuds soient en vie après le temps $\epsilon.\tau$ pour $m \in [s..s + r]$

$$p_b(X > \epsilon.\tau) = \sum_{m=s}^{s+r} C_m^{s+r} p(\epsilon.\tau)^m \left(1 - p(\epsilon.\tau)\right)^{s+r-m} \tag{7}$$

A partir de là, nous pouvons calculer le MTTF en fonction de $\epsilon.\tau$:

$$\begin{aligned} MTTF(\epsilon.\tau) &= \epsilon.\tau \sum_{i=0}^{+\infty} i \left(1 - p_b(\epsilon.\tau)\right) \left(p_b(\epsilon.\tau)\right)^i && (8) \\ &= \epsilon.\tau \frac{p_b(\epsilon.\tau)}{1 - p_b(\epsilon.\tau)} && (9) \end{aligned}$$

La figure 4 représente le MTTF d'un système avec $\tau = 1$ pour différentes valeurs de r avec s fixé à 16. L'échelle sur l'axe vertical est logarithmique. Cette courbe est intéressante car elle nous permet de voir que la MTTF chute très rapidement avec l'augmentation du facteur $\epsilon.\tau$. Malgré tout, pour une valeur faible de $\epsilon.\tau$, la durée de vie du système atteint des valeurs très élevées qui nous permettent de penser qu'un tel système peut fonctionner à condition d'être fortement réactif. Bien sur le degré de réactivité dépend de la durée moyenne de vie des noeuds (le paramètre τ). Mais dans le cas où celle-ci est brève (quelques heures), la phase de détection et réparation des pannes doit être effectuée rapidement.

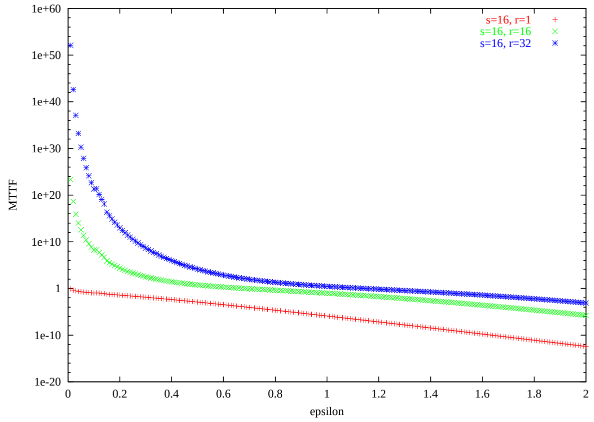

FIG. 4 – MTTF avec réparation avec $\tau = 1$

4.2. Coût des réparations

La redondance permet donc d'assurer une bonne pérennité des données, mais les résultats précédents ne sont valables que dans le cas où l'hypothèse d'un processus balai capable de détecter et réparer les pannes en $\epsilon.\tau$ soit réaliste. La reconstruction des données implique un coût en calcul et en bande passante : pour reconstruire un fragment, il est nécessaire de reconstruire le bloc en entier, c'est-à-dire de récupérer s fragments valides puis de les combiner. Si l'on considère que la méthode de Reed Solomon est utilisée pour créer les fragments, l'opération de reconstruction consiste en une multiplication matrice-vecteur et s'effectue en $O((s * B)^2)$.

Soit N le nombre de noeuds qui composent le réseau. La probabilité d'un noeud d'être encore en vie après le temps $\epsilon.\tau$ est $p(X > \epsilon.\tau)$. Alors, $(1 - p(X > \epsilon.\tau))$ est la probabilité d'un noeud d'être défaillant au temps $\epsilon.\tau$. La probabilité pd_k que k noeuds aient disparu après $\epsilon.\tau$ est donc :

$$pd_k(X > \epsilon.\tau) = C_k^N.p(X > \epsilon.\tau)^{N-k}.(1 - p(X > \epsilon.\tau)^k) \tag{10}$$

d'où en moyenne, le nombre N_d de noeuds défaillants après $\epsilon.\tau$:

$$N_d(\epsilon.\tau) = \sum_{k=1}^{N} k.C_k^N.p(X > \epsilon.\tau)^{N-k}.(1 - p(X > \epsilon.\tau)^k) \tag{11}$$

La Figure 5 représente le nombre de noeuds à réparer en fonction $\tau.\epsilon$ avec $\tau = 1$. Nous constatons que plus ϵ est grand plus le nombre de noeuds à réparer est important.

Supposons que tous les noeuds offrent 10Mo de leur disque dur, que $s = 16$ et $r = 16$. Lorsqu'un noeud est défaillant, le volume qui doit transiter sur le réseau pour reconstruire ses données perdues est de $s * B = 160Mo$ auxquels il faut ajouter, les $10Mo$ de fragments reconstruits à envoyer sur un autre noeud. Pour $\tau = 1h$ et $\epsilon = 0.5$, le nombre moyen de pannes est d'environ 4000 noeuds sur 10000 ce qui implique $640Go$ devant transiter sur le réseau. Bien sûr, il s'agit d'un cas limite dans lequel on considère que la reconstruction est faite pour chaque fragment perdu. Dans le cas réel, les fragments défaillants d'un même bloc peuvent être reconstruits simultanément. Mais même ainsi, la masse de calculs et de données transitant sur le réseau est très élevée lorsque la durée de vie des noeuds est très faible.

4.3. Diminuer le coût de la reconstruction : réplication

Les coûts de reconstruction sont élevés car à chaque fois qu'un noeud tombe en panne, pour reconstruire les fragments qu'il possédait, il est nécessaire de reconstruire l'intégralité des blocs dont provenaient ces fragments. Pour réduire ce coût nous avons imaginé ajouter de la réplication à notre mécanisme de redondance. L'idée consiste à répliquer k fois les $s+r$ fragments de redondance. Ainsi, lors de la défaillance d'un noeud, il n'est plus nécessaire de reconstruire le bloc, il suffit de recopier sur un nouveau noeud l'un des réplicats des fragments.

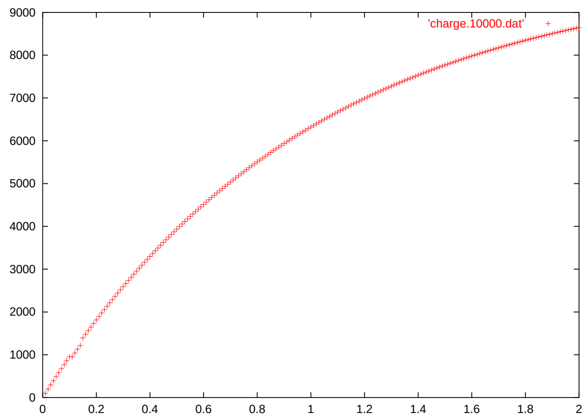

FIG. 5 – Nombre moyen de noeuds en panne en fonction de ϵ pour un réseau de 10000 noeuds pour $\tau = 1$

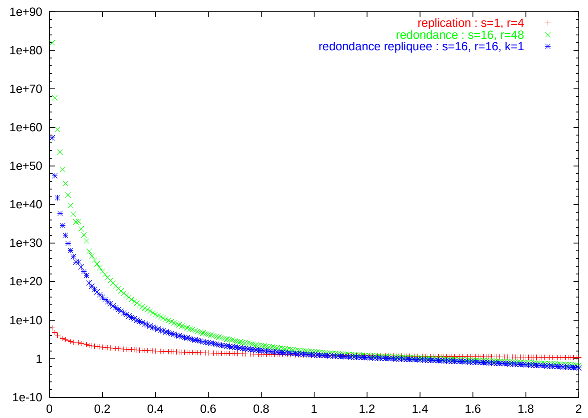

FIG. 6 – Comparaison de la MTTF entre réplication, redondance et redondance répliquée

La MTTF du système avec ce mécanisme se calcule pratiquement comme précédemment. Nous pouvons considérer les fragments comme des blocs pour lequel $s = 1$ et $r = k$. Ainsi, à partir de l'équation (7), la probabilité de survie d'un fragment devient :

$$p_r(\epsilon.\tau) = \sum_{m=1}^{k+1} C_m^{k+1} p(\epsilon.\tau)^m \left(1 - p(\epsilon.\tau)\right)^{k+1-m} \tag{12}$$

En insérant cette probabilité de survie d'un fragment à la place de la loi exponentielle dans les équations précédentes nous obtenons le MTTF avec ce mécanisme.
Cette méthode permet de diminuer les coûts de reconstruction mais à tolérance égale, diminue l'espace utile. Ainsi, à espace utile équivalent, la tolérance est moindre que celle obtenue par redondance simple mais reste supérieure à la simple réplication. La figure 6 nous le montre. Il s'agit des courbes de MTTF dans les 3 cas précédents.

5. Conclusion

Dans cet article, après avoir brièvement introduit diffèrents systèmes de stockage pair-à-pair, nous nous sommes intéressés aux mécanismes de redondance qui peuvent être mis en œuvre pour assurer la pérennité des données dans un système fortement dynamique. Nous avons alors présenté une étude quantitative de l'efficacité de ces méthodes. Les techniques de fragmentation ont de très bons résultats en terme de durée de vie des données si elles sont couplées à un système de détection et de réparation des fautes efficace. Cependant, elles sont extrêmement coûteuses en terme de calcul et surtout de communication, ce qui les rend, en l'état, peu envisageables dans un milieu volatile. Pour réduire ces coûts, nous avons alors introduit une autre technique mêlant redondance et réplication. Mais cette étude se base sur l'hypothèse qu'un noeud quittant le système n'y reviendra pas et donc que les données qu'il stockait sont définitivement perdues. Nous travaillons actuellement sur un modèle prenant en compte la possibilité de reconnections des noeuds dans le système. Ce nouveau cas de figure devrait grandement contribuer à réduire les coûts de maintien de l'intégrité des données.

Bibliographie

1. David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Christopher Wells, Ben Zhao, and John Kubiatowicz. Oceanstore : An extremely wide-area storage system. Technical Report Technical Report UCB CSD-00-1102, U.C. Berkeley, May 1999.
2. Yuan Chen, Jan Edler, Andrew Goldberg, Allan Gottlieb, Sumeet Sobti, and Peter Yianilos. A prototype implementation of archival intermemory. In *Proceedings of the Fourth ACM International Conference on Digital Libraries*, 1999.
3. Ian Clarke. A distributed decentralised information storage and retrieval system, 1999.
4. Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet : A distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2000.
5. P. Druschel and A. Rowstron. PAST : A large-scale, persistent peer-to-peer storage utility. In *Procedings of HOTOS*, pages 75–80, 2001.
6. A. Goldberg and Peter N. Yianilos. Towards an archival intermemory. In *Proceedings of IEEE Advances in Digital Libraries, ADL 98*, pages 147–156, Santa Barbara, CA, 1998. IEEE Computer Society.
7. James S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software Practice and Experience*, 27(9) :995–1012, 1997.
8. Sean C. Rhea and John Kubiatowicz. Probabilistic location and routing. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, June 2002.
9. Matei Ripeanu. Peer-to-peer architecture case study : Gnutella network. In *Proceedings of International Conference on Peer-to-peer Computing*, August 2001.
10. Antony I. T. Rowstron and Peter Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Symposium on Operating Systems Principles*, pages 188–201, 2001.
11. Chris Wells. The oceanstore archive : Goals, structures, and self-repair. Master's thesis, University of California, Berkeley, May 2001.
12. Ben Y. Zhao. *A Decentralized Location and Routing Infrastructure for Fault-tolerant Wide-area Applications*. Ph.d. qualifying examination, U.C. Berkeley, April 2001.