Stockage des données XML

Didier Courtaud

CEA DAM Ile-de-France

Courriel : Didier.Courtaud@cea.fr

- Ce cours a pour objectifs
  - De vous faire comprendre pourquoi le problème du stockage est essentiel pour les performances
  - De vous expliquer qu'il est discriminant entre les différentes bases XML natives existantes
  - De vous montrer comment il a été traité dans XediX
  - De vous présenter les résultats obtenus sur des cas tests significatifs

Positionnement du problème

- Lorsqu'on importe des données XML dans une base de données, il faut
  - Vérifier leur validité
    - C'est le rôle du parseur
  - Décider de leur stockage
    - C'est le rôle du gestionnaire physique de données
- Rappel du fonctionnement de XediX
  - C'est le parseur OpenSP utilisant la méthode SAX qui
    - Vérifie la validité XML du document importé
    - Déclenche
      - L'indexation par les moteurs de recherche
      - Le stockage

Positionnement du problème (suite)

- La façon de stocker les éléments XML détermine
  - L'espace disque nécessaire à la base
  - Les opérations nécessaires à l'extraction de l'information
  - Les performances obtenues lors de l'extraction de l'information
- C'est donc l'opération cruciale pour optimiser les performances générales du système

Différentes approches

- Les principales solutions se différencient sur "l'atomicité" requise :
  - Soit tout le document XML
  - Soit l'élément XML
- Document XML
  - Choix de Tamino (Software AG)
    - Stockage des documents XML entiers dans Adabas (base hiérarchique de Software AG)
  - Choix de Exist (Base NXD Open Source)
    - Stockage des arbres DOM sur disque
- Élément XML
  - Choix fait par XediX

Différentes approches (suite)

- **Répercution sur l'extraction de données :**
  - Une requête-sélection classique nécessite
    - Un parsing systématique dans le cas d'un stockage du document entier
    - Un adressage direct dans le cas du stockage individuel de chaque élément
      - "*Parse one Read Many*"
    - Équivalent pour des petits volumes de données
    - Fondamentalement différent lorsqu'on adresse des volumes plus grands

- Un des gros avantages des bases XML natives vis à vis des SGBDR est qu'elles ne nécessitent pas de systèmes physiques spécialisés
  - Toute "l'intelligence du stockage" est logicielle
  - Cout moindre des surfaces de stockage
  - Cout moindre de l'administration de base
  - Pas besoin de "tuning" système (sauf pour de très grandes volumétries)

Support matériel de l'information (suite)

- De fait n'importe quel système de gestion physique est acceptable
  - Système de fichiers
    - De préférence journalisés pour pouvoir facilement faire des retours arrière
  - Autres systèmes de bases de données
    - Hiérarchiques
      - Cas de Tamino
    - Relationnelles
      - Nécessite d'avoir des APIs de bas niveau
      - SQL ne convient pas

- Cas de XediX
  - Par défaut, XediX s'appuie sur un système de gestion de données de type hiérarchique "intelligent"
    - CLIO ™ développé à l'origine par le CEA pour le Programme Simulation Numérique
      - Portable sur toutes cibles Unix
      - S'appuie sur des outils standards d'Unix
        - XDR
        - CPIO
      - Optimisé pour la gestion de grands volumes
      - Sécurité par l'ignorance

Support matériel de l'information (fin)

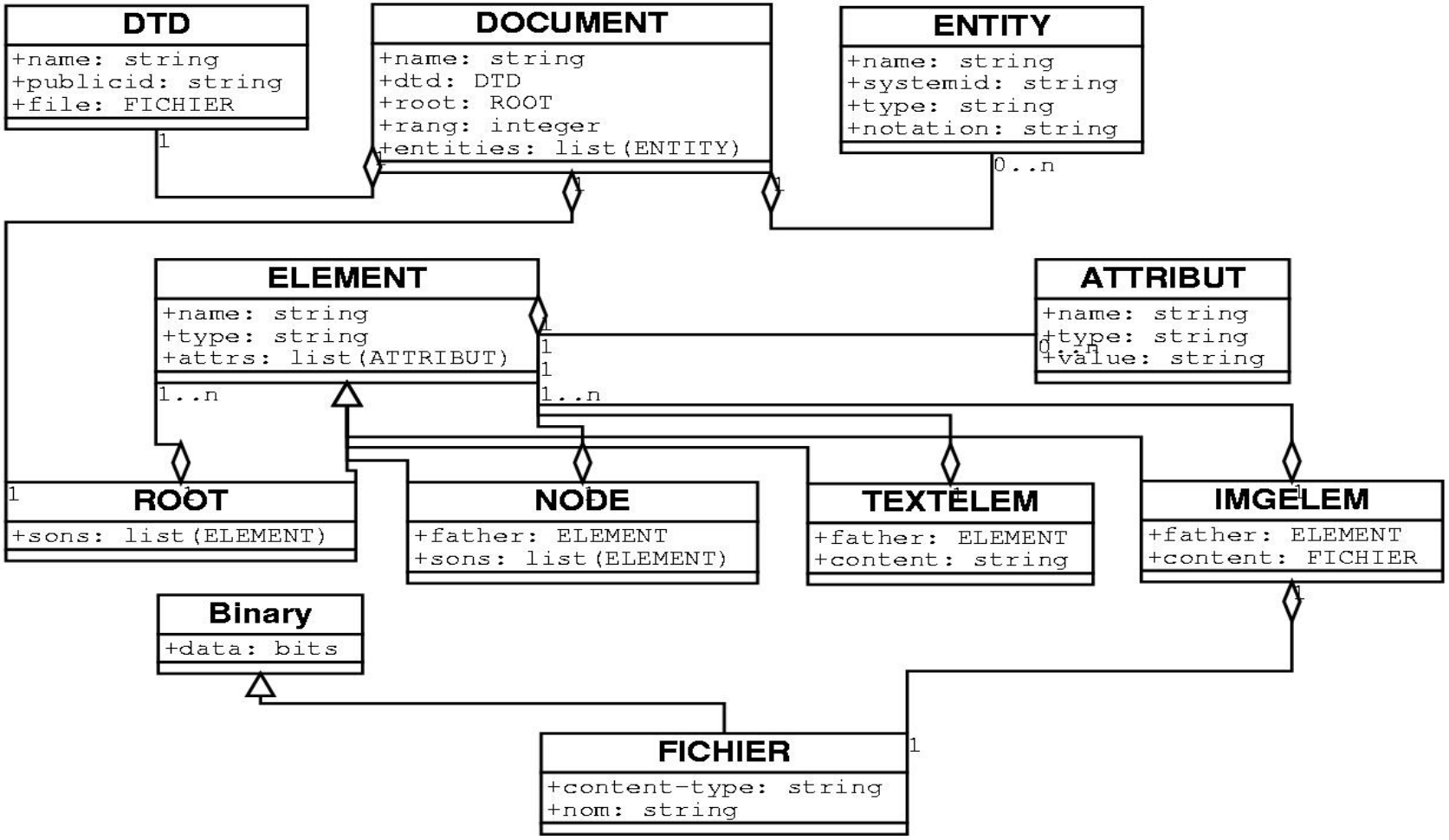
- Mais XediX a aussi été testé sur
  - MySQL
    - Portage réalisé par des étudiants de M2 MIAGE il y a 3 ans
    - Utilise des primitives de bas niveau disponibles sur la version Open Source de MySQL
  - Oracle
    - Portage test
    - En utilisant l'interface ODBC
  - ReiserFS3 et ReiserFS4
    - En utilisant le driver File développé pour cet usage
    - Disponible en standard
    - Fonctionne avec ext3, ..

Stratégie de stockage

- Comment doit-on stocker les éléments XML pour pouvoir les retrouver rapidement ?
- Deux approches sont possibles
  - En s'appuyant sur la structure des documents importés
    - Avantages
      - On connait parfaitement la structure
    - Inconvénients
      - Il faut refaire le "schéma" de base pour chaque DTD
  - En s'appuyant sur une structure générique
    - Avantages
      - Plus de "schéma" de base
    - Inconvénients
      - Moins optimisé

- XediX utilise la deuxième approche (*générique*)

Diagramme de classe UML de la description générique d'un document dans Xedix



Stratégie de stockage (suite)

- DTD ou schéma XML ?
- Intérêt des schémas XML
  - Ajoute une information de typage à l'information de structure
  - Mais en contrepartie fige le type d'un champ (élément XML)
- Intérêt des DTDs
  - Ne décrit que l'information de structure
  - La seule utile dans le cas documentaire
  - Permet de laisser l'application fixer le type du champ
  - Exemple : date
- XediX travaille avec des DTDs XML ou SGML
  - Fournit des opérateurs typés pour *caster* l'élément au niveau applicatif

Stratégie de stockage (suite)

- Chaque élément XML stocké se voit affecter une URI interne à XediX préfixée par "xedix:"
  - Permet de référencer directement chaque élément individuellement
  - Autorise un accès très rapide à chaque élément
  - Propre à XediX
- Chaque URI peut donc être récupérée par HTTP
  - Soit en totalité
    - Cas classique
  - Soit partiellement
    - Particulièrement intéressant pour les images et la vidéo

- **Utilisation de HTTP Range**
  - Inclus dans le protocole HTTP depuis la version 1.1 (RFC 2616 Juin 1999)
  - Entête permettant de spécifier la "portion" d'élément qu'on veut récupérer en octets

```
ranges-specifier = byte-ranges-specifier
byte-ranges-specifier = bytes-unit "=" byte-range-set
byte-range-set  = 1#( byte-range-spec | suffix-byte-range-spec )
byte-range-spec = first-byte-pos "-" [last-byte-pos]
first-byte-pos  = 1*DIGIT
last-byte-pos   = 1*DIGIT
```

```
- The first 500 bytes (byte offsets 0-499, inclusive):  bytes=0-
  499

- The second 500 bytes (byte offsets 500-999, inclusive):
  bytes=500-999

- The final 500 bytes (byte offsets 9500-9999, inclusive):
  bytes=-500

- Or bytes=9500-

- The first and last bytes only (bytes 0 and 9999):  bytes=0-0,-1

- Several legal but not canonical specifications of the second 500
  bytes (byte offsets 500-999, inclusive):
   bytes=500-600,601-999
   bytes=500-700,601-999
```

Stratégie de stockage (suite)

- La plupart des serveurs Web actuels supportent cette version du protocole
  - Apache depuis les versions 1.x
- Les applicatifs eux aussi supportent cette version du protocole sans forcément l'afficher
  - Real Player
    - Depuis la version 10
  - Adobe Acrobat Reader
  - La plupart des logiciels libres

Stratégie de stockage (suite)

- Où stocker les objets non-XML ?
  - Images
  - Sons
  - Vidéos
  - Données numériques
  - …
- Les SGBDR (et la plupart des NXDs) stockent
  - Les objets eux-mêmes sur un système de fichiers
  - Les métadonnées les concernant dans la base
  - Et Établissent un lien "hard" entre les deux

Stratégie de stockage (suite)

- **Avantages**
  - Volume de données à traiter faible
  - Métadonnées très structurées
- **Inconvénients**
  - Sécurité
    - Deux emplacements à protéger
  - Cohérence
    - Il faut déplacer en même temps les deux emplacements
    - Sinon liens cassés
  - Récupération partielle
    - Difficile et pas optimisée
    - Repose sur un outil tiers

Stratégie de stockage (suite)

- XediX a contrario stocke toutes les données binaires en base
  - Meilleure cohérence
    - On peut déplacer la base "en bloc" et tout remarche instantanément
  - Meilleure sécurité
    - Les images, les vidéos font partie du système de sécurité de la base
  - Meilleure utilisation
    - Chaque objet a une URI interne
    - Il peut donc être exploité par HTTP

Stratégie de stockage (suite)

- En pratique, il suffit d'indiquer à XediX où se trouvent les objets externes
- A l'import
  - XediX analyse le fichier XML à la recherche des balises référençant les objets externes
  - Quand il en trouve une
    - Il récupère l'objet externe en tant que train de bits
    - Il lui affecte une URI interne
    - Il modifie l'import XML pour faire pointer le lien vers l'URI interne

Stratégie de stockage (suite)

- **Exemple**
  - Fichier XML

```
<?xml>
<DOCTYPE SYSTEM "madtd.dtd"/>
<doc id="abc123"/>
        <par> Ceci est un exemple </par>
        <image src="maphoto.jpg"/>
        <video mpg="http://www.youtube.com/mavideo.mpeg"/>
</doc>
```

  - dtds.ini

```
[madtd.dtd]
.......
REFTAG="image;video"
REFATTR="src;mpg"
.........
```

  - Résultat

```
<?xml>
<DOCTYPE SYSTEM "madtd.dtd"/>
<doc id="abc123"/>
        <par> Ceci est un exemple </par>
        <image src="xedix:2567"/>
        <video mpg="xedix:8995"/>
</doc>
```

Stockage de grands volumes

- Une suite d'expérimentation baptisées Xtera a été menée depuis quelques années pour tester les performances de stockage de XediX sur de très grandes volumétries
- Chaque situation est particulière
  - Ces tests servent à donner des points de repère des performances
    - Sur des jeux de tests normalisés
    - Aisément reproductibles
  - Cela ne prouve pas que la base ne peut avoir de meilleures performances encore

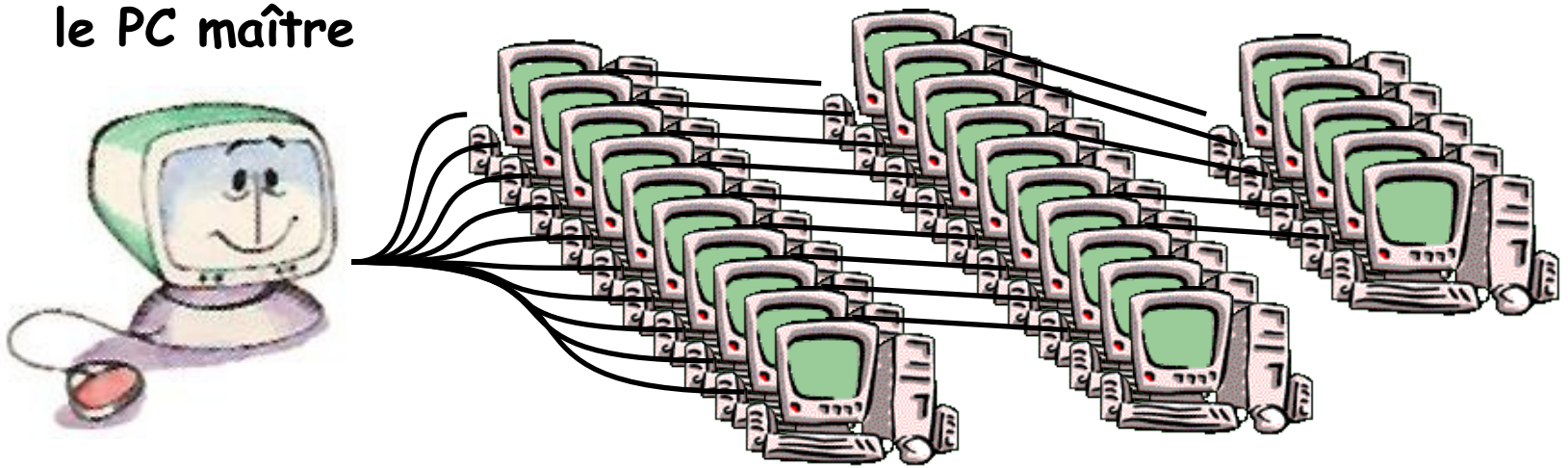
Le projet Xtera 1 (2003)

- Projet de benchmark pour comparer les performances
- de XediX à celles de ses principaux concurrents
- Alors que la plupart des implémentations actuelles des NXDs saturent au voisinage de 10 Go, nous voulions démontrer que XediX peut gérer un volume de 1 To de documents XML !
- Mesures
  - des performances d 'import
  - des performances d 'indexation
  - des performances d 'export

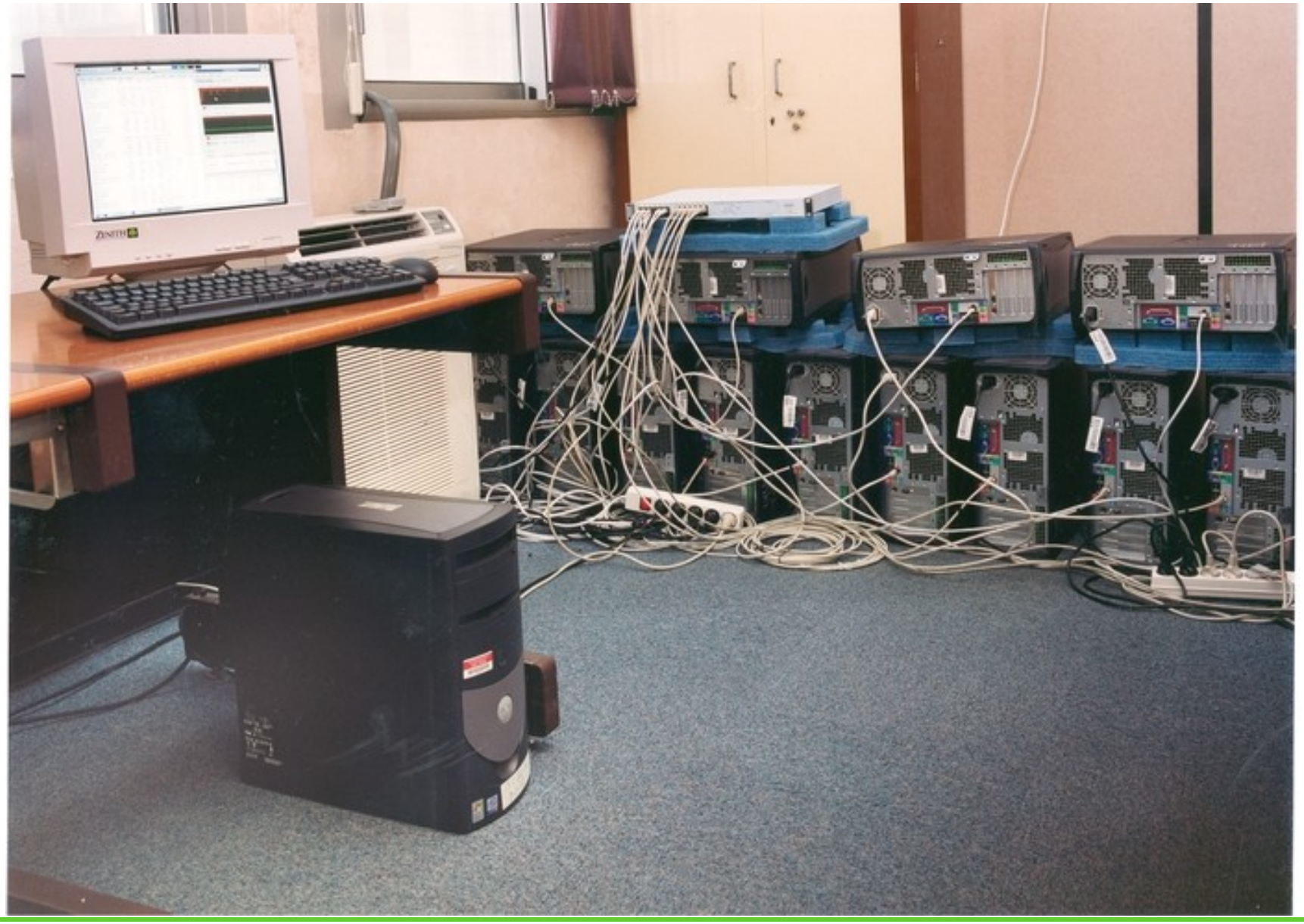
Le projet XTera (II)

- Nécessité de prendre des cas tests reconnus au niveau international
  - Xmach
    - http://dbs.uni-leipzig.de/en/projekte/XML/XmlBenchmarking.html
  - The Michigan Benchmark (Mbench) :
    - http://www.eecs.umich.edu/db/mbench/
  - XBench
    - http://db.uwaterloo.ca/~ddbms/projects/xbench/
  - XMark :
    - http://monetdb.cwi.nl/xml/index.html
  - X007 :
    - http://www.comp.nus.edu.sg/~ebh/XOO7.html

- Architecture du test
  - 25 PCs de 40 Go + 1 PC maître reliés par un switch
    - Chaque PC esclave tourne un système Linux *minimal*
    - Le PC maître tourne une distribution Linux Slackware standard
  - LVM (Logical Volume Manager) + eNBD (extended Network Block Device) nous permettent de voir tous les disques individuels des PCs esclaves comme un seul disque de 1 To sur le PC maître



Le projet XTera : montage réel



- **Résultats obtenus**
  - La base a importé un peu plus de 1 To sans aucun problème
  - Performances de requêtage
  - <u>Le temps de réponse à une requête est indépendant de la taille de la base et ne dépend que du nombre total de documents retournés</u>
  - Le temps de réponse moyen d'une requête (2/3 de la base) sur la structure est de moins de 10 secondes

Articles dans :

- Le Monde Informatique
- 01 Informatique


XML ● Cinq secondes pour retrouver un document dans une base de 1 Téraoctet : c'est l'exploit réalisé par XediX, la base de données native XML développée par le CEA, dont les performances surpassent largement celles de tous les outils équivalents du marché. AM2 Systems, une start-up de l'INRIA, vient de lancer la commercialisation. Premiers utilisateurs visés : l'archivage, les médiathèques, les projets coopératifs de recherche ; et à terme, tous les gestionnaires de grandes bases de données.

Base de données XediX

Cinq secondes pour retrouver une aiguille dans une botte de foin

POINTS FORTS

- Base de données multimédia sous Unix.
- Cinq secondes pour une requête simple sur 1 Téraoctet de données.
- Base native XML, indépendante des logiciels d'origine des données.
- Détermination et modification libre des structures de données.
- Gestion intégrée et ultra-fine des droits d'accès.

Portage sur architecture parallèle

- XTera 1 avait démontré
  - La robustesse et les performances de la base sur un grand volume
  - Que cette performance pouvait être atteinte avec du matériel standard

- Le portage sur architecture parallèle visait à démontrer
  - Que XediX pouvait tirer parti d'une architecture parallèle distribuée pour augmenter ses performances en
    - Volumétrie
    - Temps d'indexation

Portage : principes

- **Principes**
  - Import et indexation des documents
    - Division et séparation *étanche* du domaine des documents
    - Affectation à chaque processeur d'un sous-ensemble des documents
  - Consultation et requêtage
    - Un serveur XediX (et son serveur Web) est affecté à chaque processeur et contrôle son sous-ensemble de documents
    - L'un de ses serveurs (n'importe lequel) est choisi pour être le serveur de syndication qui dialoguera
      - Avec les autres serveurs pour leur demander les documents ou les fragments de documents
      - Avec l'IHM pour prendre la requête et afficher les résultats

Le projet Xtera10 (2005)

- La deuxième expérimentation Xtera visait à prouver :
  - Que les résultats obtenus sur Xtera 1 étaient extrapolables à de plus grands volumes
  - Que le développement du méta serveur de syndication de XediX
    - L'autorisait à exploiter les possibilités des architectures parallèles en terme de performance
    - Prouvait la scalabilité du produit quant au nombre de noeuds
  - Que la version 2 du moteur de recherche ReX permettait un gain important à l'indexation tant au niveau du temps que de la place disque
  - Permettait d'atteindre la barre symbolique des 10 To (10 x disque dur grand public)

Le projet Xtera10 (II)

- Expérimentation réalisée en partenariat avec Bull sur le cluster TeraNova de Bull installé en zone Ter@tec près du Centre CEA DAM Île-de-France
  - 300 processeurs Pentium Itanium 2
- La base a importé 10 Téraoctets de documents XML sans aucun problème
  - 20 millions de noeuds XML importés
  - Temps de requêtage moyen : ~ 2 secondes

Le projet Xtera 100 (2007)

- Expérimentation réalisée sur le supercalculateur Tera 10 (d'origine Bull) du CEA DAM Ile-de-France en pré-production
- 812 processeurs utilisés
- 200 millions de noeuds XML importés
- Taille de la base : 110 To
- Requêtage moyen
  - 1 seconde sur requête non typée
  - Moins d'une seconde sur requête XML

Conclusion

- Le stockage des informations XML est le point clé des performances des NXDs
  - Preuve d'une meilleure adéquation au monde XML que les SGBDR
  - Offrir des performances impossibles en relationnel
  - Permettre des requêtes impossibles dans un temps raisonnable en relationnel
- XediX a prouvé particulièrement l'efficience de cette approche
  - En implémentant un modèle purement XML pleinement opérationnel
  - En donnant accès à des volumétries inédites