Positionnement des progiciels d'historisation parmi les solutions de gestion de données

Brice Chardin[1], Jean-Marc Lacombe[2] et Jean-Marc Petit[1]

[1] Université de Lyon, CNRS
[1] INSA-Lyon, LIRIS, UMR5205, F-69621, France
[2] EDF R&D, France

Résumé Pour gérer les données de ses systèmes de production d'électricité, EDF a fait le choix de systèmes dédiés à ce cas d'application : les progiciels d'historisation. Ces produits « de niche » ont évolué en parallèle des autres systèmes de gestion de données, en se spécialisant pour ce segment du marché. Dans cet article, nous cherchons à répondre à la question suivante : comment se positionnent les progiciels d'historisation de données parmi les systèmes de gestion de données ? Pour cela, nous examinons les différences avec trois autres catégories de systèmes : SGBDR, systèmes de gestion de flux de données et systèmes NoSQL ; puis définissons un benchmark dérivé du contexte industriel d'EDF.

Mots-clé: Progiciels d'historisation, benchmark, données de capteurs, flux de données

1 Introduction

EDF est un des principaux producteurs d'électricité dans le monde. En France, le groupe exploite un parc constitué de centrales nucléaires, hydrauliques, thermiques à flammes (gaz, charbon, fioul), éoliennes et solaires.

Ces systèmes de production d'électricité, répartis sur plus de cinq cent sites, sont largement instrumentés : au total, environ un million de capteurs constituent les éléments de base des systèmes d'acquisition de données, et échangent de l'ordre du milliard d'enregistrements par jour. L'exploitation de ce volume important de données nécessite dans un premier temps de disposer de moyens d'accès. Pour cela, des systèmes de gestion de bases de données (SGBD) se chargent de leur archivage et de leur mise à disposition. L'archivage de ces données industrielles est un problème complexe. En effet, il s'agit de stocker un grand nombre de données – sur la durée de vie des installations, soit plusieurs décennies – tout en supportant la charge des insertions temps réel et des requêtes d'extraction et d'analyse soumises au SGBD.

Au sein de ces centrales, le système d'information est constitué dans un premier temps d'un ensemble de dispositifs informatiques utilisés en temps réel pour l'exploitation et la surveillance des installations : des systèmes de contrôle-commande prennent en charge des actions automatiques sur le procédé, tandis

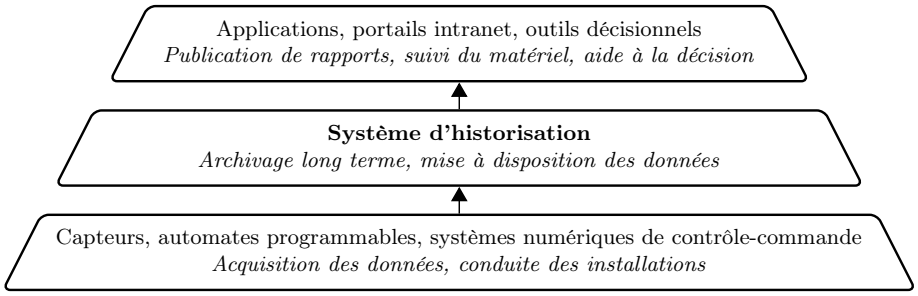

Figure 1. Position des systèmes d'historisation dans le système d'information de production

que d'autres systèmes acquièrent et traitent les données de production pour permettre leur visualisation et assister les opérateurs dans leurs activités de contrôle, de suivi des performances ou d'optimisation de la conduite. À ce niveau, les données sont principalement issues des capteurs ou des rapports d'état du contrôle-commande, mais également de saisies manuelles et d'applications métiers spécifiques pour certains traitements.

L'accès direct à ces données est délicat à cause des impératifs de sûreté : toute intervention sur les données doit se faire sans impacter le fonctionnement de l'existant. À EDF, cet accès est fourni par ce qu'il est convenu d'appeler un « système d'historisation », chargé d'archiver les données de production. Ces données sont alors mises à disposition de diverses applications sans contraintes temps réel (suivi de matériel, outils décisionnels, etc.) et d'acteurs extérieurs à la centrale (ingénierie, R&D, etc.). La figure 1 reprend l'organisation du système d'informations des centrales d'EDF, et présente le positionnement des systèmes d'historisation comme intermédiaires fondamentaux pour l'accès aux données de production.

À l'échelle d'une centrale, des milliers de séries temporelles sont ainsi archivées par le système d'historisation. Chaque série, désignée par un identifiant de « repère », est une séquence de mesures horodatées, généralement représentées par un ensemble de paires date-valeur – où la valeur peut posséder plusieurs dimensions, typiquement la mesure et une métadonnée sur sa qualité. Les périodes d'échantillonnage varient selon les repères, allant de 200 ms – pour la vitesse de rotation de la turbine par exemple – à quelques minutes. Certains repères correspondent à des variables de type événement, dont l'acquisition se produit lors d'un changement d'état – l'ouverture ou la fermeture d'une vanne par exemple.

Pour manipuler des données structurées, les systèmes de gestion de bases de données relationnels (SGBDR) sont la solution privilégiée depuis une quarantaine d'années. Cependant, ceux-ci sont peu utilisés dans ce contexte industriel très spécifique soumis à des contraintes de performances en insertion. Depuis les années 90, EDF a fait le choix d'une catégorie de SGBD dédiée à ce cas d'ap-

plication : les progiciels d'historisation. Ces systèmes ont continué à évoluer en parallèle des SGBDR, en se spécialisant pour ce segment du marché.

Ces progiciels d'historisation sont des solutions propriétaires avec des coûts de licence de l'ordre de plusieurs dizaines de milliers d'euros, et dont le fonctionnement interne n'est pas dévoilé. D'un autre coté, de nombreuses solutions open-source existent pour manipuler de telles données. Les SGBDR en font partie, mais également certains SGBD « NoSQL » dont l'interface d'accès simplifiée peut être adaptée au contexte. Cependant, leurs différences avec les progiciels d'historisation restent mal connues, tant au niveau des fonctionnalités que des performances : alors que de nombreux benchmarks permettent d'évaluer les SGBD dans divers environnements d'utilisation, il n'en existe à notre connaissance aucun à ce jour pour l'historisation de données industrielles.

Dans ce contexte, notre contribution vise à donner des éléments de réponse sur le positionnement des progiciels d'historisation parmi les SGBD. Pour cela, nous présentons dans un premier temps les progiciels d'historisation, en analysant les différences avec les SGBDR, les systèmes de gestion de flux de données (*Data Stream Management System* ou DSMS) et les SGBD NoSQL. Nous proposons ensuite un benchmark, inspiré du fonctionnement de l'historisation des données des centrales nucléaires, et générant des requêtes d'insertion, de mise à jour, de récupération et d'analyse des données. Des résultats expérimentaux sont présentés pour le progiciel d'historisation InfoPlus.21 [2], le SGBDR MySQL [5], et le SGBD NoSQL Berkeley DB [4].

2 Progiciels d'historisation de données

Les progiciels d'historisation de données – comme InfoPlus.21 [2], PI [6] ou Wonderware Historian [3] – sont des progiciels propriétaires conçus pour archiver et interroger les données issues de l'automatique industrielle. Leurs fonctionnalités de base sont toutefois proches de celles des systèmes de gestion de base de données (SGBD), mais spécialisées dans l'historisation de séries temporelles.

Ces progiciels d'historisation de données – ou *data historians* – ont pour but de collecter en temps réel les données en provenance du procédé industriel et de les restituer à la demande. Pour cela, ils sont capables de gérer les données de plusieurs milliers de capteurs : leur historique, pouvant porter sur des décennies de fonctionnement, est alimenté par un flux constant de données en insertion, tout en effectuant des calculs et en répondant aux besoins des utilisateurs.

Les progiciels d'historisation supportent des rythmes d'insertion élevés, et peuvent ainsi traiter plusieurs centaines de milliers d'évènements – soit quelques mégaoctets de données – à la seconde. Ces performances sont permises par une conception spécifique des buffers d'insertion, qui conservent les valeurs récentes en mémoire volatile pour les écrire ensuite sur disque, triées chronologiquement. La fenêtre temporelle associée à ce buffer impose des contraintes fortes sur l'horodatage des données. Ces contraintes peuvent ainsi porter sur un intervalle de validité, par exemple uniquement des dates passées ou appartenant à une fenêtre temporelle donnée, ou encore sur la séquentialité des données, ie. les insertions

doivent être effectuées en respectant leur ordre chronologique. Le non-respect de ces contraintes peut entraîner une diminution notable des performances, voire l'impossibilité d'insérer les données.

Le modèle de données utilisé par les progiciels d'historisation est un *modèle hiérarchique*, permettant de représenter les données suivant la structuration physique des installations et faciliter ainsi la consultation de données similaires groupées par sous-systèmes. Dans le domaine de la production d'électricité par exemple, les repères peuvent être regroupés dans un premier temps par site d'exploitation, puis par unité de production, et enfin par système élémentaire (e.g. le système de production d'eau déminéralisée).

Les progiciels d'historisation proposent également une vision relationnelle des données archivées, accessible à l'aide d'une interface SQL. Les progiciels d'historisation peuvent cependant ne pas se conformer à la norme SQL dans son ensemble. De plus, les progiciels d'historisation ne sont pas conçus pour gérer des bases de données relationnelles dans le cas général, certains types de données n'étant pas supportés (image, texte, BLOB, etc.).

En plus du SQL, les progiciels d'historisation fournissent également une interface dédiée pour les insertions, mises à jour et récupération des données. Les insertions sont fonctionnellement comparables aux requêtes SQL "INSERT", en évitant l'analyse syntaxique et les conversions de types. L'interface de récupération des données cependant diffère significativement du SQL. Les extractions peuvent être définies avec des conditions de filtrage (typiquement avec des seuils de valeur ou des vérifications du champ qualité), du ré-échantillonnage ou des calculs d'agrégats sur des intervalles temporels – par exemple pour calculer la moyenne horaire. Bien que les conditions de filtrage et la définition d'intervalles soient traduisibles simplement en SQL, l'interpolation de valeurs (avec divers algorithmes d'interpolation : par pallier, linéaire, etc.) peut être fastidieuse à définir, tant en SQL qu'avec des interfaces NoSQL usuelles, en particulier lorsque plusieurs séries temporelles – possédant leur propre période d'échantillonnage – sont concernées, comme pour le produit de deux séries par exemple.

Néanmoins, les entrepôts de paires clé-valeur ordonnées fournissent des méthodes d'accès NoSQL proches, comme les curseurs de Berkeley DB [4]. Ces curseurs peuvent être positionnés sur une valeur de clé, et être incrémentés ou décrémentés suivant l'ordre des clés – pour récupérer les valeurs consécutives d'une série temporelle dans ce contexte. Malgré tout, l'interface des progiciels d'historisation est spécialisée, et donc combine de nombreux algorithmes et traitements usuels adaptés aux besoins industriels, en plus de l'extraction de données brutes.

Concernant la politique tarifaire, les progiciels d'historisation sont proposés avec des licences propriétaires. Il est difficile d'estimer le coût de ces licences car leur prix est négocié pour chaque client. L'ordre de grandeur de ce coût est de plusieurs dizaine de milliers d'euros par serveur, avec des tarifs dégressifs en fonction du nombre de repères – allant de 10 à 0.5 euros par repère. Les progiciels

d'historisation proposent une offre de support et de maintenance, dont le coût annuel correspond à un pourcentage du prix de la licence.

Synthèse Les progiciels d'historisation sont des produits conçus et vendus pour un usage industriel spécifique. Les autres SGBD peuvent avoir des cadres d'application plus variés, pour un coût potentiellement moins important, mais n'incluent généralement pas la plupart des fonctionnalités métier que possèdent les progiciels d'historisation. Ces systèmes ne supportent typiquement pas les protocoles de communication industriels, ni les algorithmes de compression avec perte spécifiques aux séries temporelles, l'interpolation ou le ré-échantillonnage. Dans l'ensemble, les progiciels d'historisation peuvent être caractérisés par :

- une structure de schéma hiérarchique simple, basée sur les repères,
- une architecture centralisée,
- une conception optimisée pour l'archivage long-terme d'un grand volume de données ordonnées chronologiquement, où la date joue un rôle fondamental,
- des algorithmes de compression adaptés,
- une "interface NoSQL" avec filtrage, ré-échantillonnage et calcul d'agrégats,
- une interface SQL,
- pas de transactions,
- uniquement des données de capteur (pas d'image, de blob, etc.),
- des applications intégrées spécialisées pour les données industrielles,
- une politique tarifaire basée sur le nombre de repères.

En quelque sorte, les progiciels d'historisation sont des SGBD non relationnels – donc « NoSQL » – qui ont su s'imposer sur un marché de niche. Pour autant, ils ne correspondent à aucune des catégories de SGBD existantes.

3 Benchmark adapté à l'historisation de données

Une évaluation basée sur des critères fonctionnels est importante, mais ne permet pas d'avoir une idée quantitative sur les capacités de traitement de chaque système. Par ailleurs, les éditeurs des progiciels d'historisation ne publient pas les capacités de traitement de leurs solutions. L'utilisation d'un benchmark est donc le seul moyen d'évaluer les différences de performances entre ces systèmes. Cependant, cette comparaison ne s'avère pas si facile, les fonctionnalités, les interfaces et le modèle de données sous-jacents étant différents.

Pour cela, nous nous concentrons sur des opérations (requêtes) simples sur un schéma de base de données générique. Nous proposons donc un benchmark et l'utilisons pour évaluer un progiciel d'historisation, un entrepôt de paires clé-valeur ordonnées et un SGBDR.

Alors que de nombreux benchmarks existent pour les systèmes de gestion de base de données relationnels, comme TPC-C ou TPC-H [7,8], il n'en existe, à notre connaissance, pas pour les progiciels d'historisation. L'idée de comparer ces systèmes à l'aide d'un benchmark existant – adapté aux SGBDR – est donc naturelle. Cependant, il ne nous a pas semblé possible de mettre en oeuvre

un benchmark du Transaction Processing Performance Council (TPC) pour les raisons suivantes :

- Les progiciels d'historisation ne respectent pas les contraintes ACID et ne permettent pas les transactions.
- L'insertion au fil de l'eau est une opération primordiale pour les systèmes d'historisation, ce qui exclut les benchmarks insérant les données par groupements, comme TPC-H.
- Les progiciels d'historisation sont conçus pour traiter des séries temporelles. Il est nécessaire que le benchmark manipule ce type de données pour que les résultats soient significatifs.

Les benchmarks pour les DSMS, comme *Linear Road* [1] auraient aussi pu être envisagés ; mais les progiciels d'historisation ne supportant pas les requêtes continues, leurs mises en œuvre auraient été impossibles.

Pour comparer les performances des progiciels d'historisation avec d'autres SGBD, nous définissons un benchmark basé sur un scénario reprenant le fonctionnement de l'historisation des données des centrales nucléaires d'EDF. Dans ce contexte, les données sont issues de capteurs répartis sur le site d'exploitation et agrégées par un démon servant d'interface avec le système d'historisation. Pour les insertions, ce benchmark simule le fonctionnement de ce démon et génère pseudo-aléatoirement les données à insérer. Ces données sont alors accessibles par des applications ou utilisateurs distants, qui peuvent envoyer des requêtes pour mettre à jour, récupérer ou analyser ces données. Après la phase d'insertion, ce benchmark génère un ensemble simple mais représentatif de ce type de requêtes.

3.1 Modèle de données

Ce benchmark manipule les données selon un schéma minimal, centré sur les données de séries temporelles. Pour chaque type de variable – analogique ou booléen – une table de description est définie (resp. `ana` et `bool`). Les mesures sont stockées dans des tables différentes (resp. `anavalues` et `boolvalues`). La figure 2 présente le schéma logique utilisé pour ce benchmark.

Chaque repère est en particulier associé à un identifiant (`AnaId` ou `BoolId`). Pour les données analogiques, la table de description contient également une période d'échantillonnage théorique (`Interval`) et deux seuils délimitant les valeurs critiques basses (`Threshold1`) ou hautes (`Threshold2`).

Les séries temporelles sont stockées dans les tables `anavalues` et `boolvalues`, qui contiennent l'identifiant (`AnaId` ou `BoolId`) – en tant que clé étrangère –, la date de la mesure avec une précision à la milliseconde (`Date`), la valeur (`Value`) et un tableau de huit bits pour les méta données (`Quality`).

Pour que ce benchmark soit compatible avec les modèles de données hiérarchiques utilisés par les progiciels d'historisation, le modèle relationnel défini précédemment ne peut pas être imposé. Dans la figure 3, nous proposons un modèle hiérarchique équivalent, permettant de représenter les mêmes données et d'exécuter des requêtes fonctionnellement équivalentes.

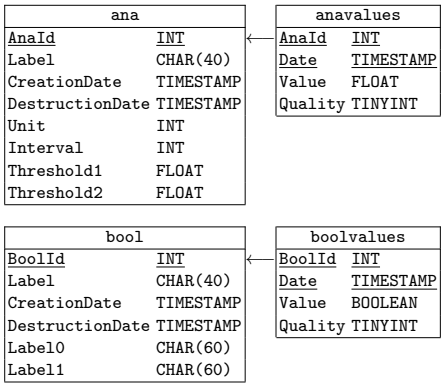

Figure 2. Schéma logique de la base de données

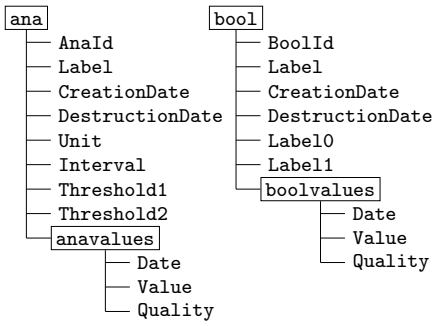

Figure 3. Schéma logique équivalent pour le modèle hiérarchique

3.2 Requêtes

Ce benchmark définit douze requêtes représentatives de l'usage d'EDF pour évaluer les performances de chaque système, et identifier les optimisations spécifiques à certains types de requêtes. Pour conserver une définition simple et faciliter l'analyse des performances, les interactions entre les requêtes ne sont pas prises en comptes : les requêtes sont exécutées une par une dans un ordre établi. En particulier, l'évaluation des performances malgré une charge continue en insertion n'est pas considérée, même si cela correspond à une situation plus réaliste. De même, les traitements spécifiques aux séries temporelles proposés par les progiciels d'historisation ne font pas partie des requêtes exécutées par le benchmark, car leur équivalent en SQL standard peut s'avérer compliqué à définir (par exemple pour calculer une moyenne pondérée par les intervalles temporels – variables – des mesures). Pour chaque requête, les équivalents NoSQL doivent permettre d'obtenir les mêmes résultats.

Insertion des données
R0.1 Insertion de valeur analogique.
Paramètres `ID`, `DATE`, `VAL` et `QUALITY`.

```
INSERT INTO anavalues VALUES
  ([ID],[DATE],[VAL],[QUALITY]);
```

R0.2 Insertion de valeur booléenne.

Modification des données La mise à jour de données est une opération rare pour les systèmes d'historisation. Le benchmark considère cependant l'impact des mises à jour sur les performances.

R1.1 Mise à jour d'une valeur analogique et de son champ qualité.
Paramètres `VAL`, `ID` et `DATE`.

```
UPDATE anavalues
SET Value = [VAL], Quality = (Quality | 128)
WHERE AnaId = [ID] AND Date = [DATE];
```

R1.2 Mise à jour d'une valeur booléenne et de son champ qualité.

Extraction de données brutes

R2.1 Valeurs analogiques brutes.
Paramètres `ID`, `START` et `END`.

```
SELECT * FROM anavalues
WHERE AnaId = [ID] AND Date BETWEEN [START] AND [END]
ORDER BY Date ASC;
```

R2.2 Valeurs booléennes brutes.

Calcul d'agrégats

R3.1 Dénombrement de données analogiques.
Paramètres `ID`, `START` et `END`.

```
SELECT count(*) FROM anavalues
WHERE AnaId = [ID] AND Date BETWEEN [START] AND [END]
```

R3.2 Dénombrement de données booléennes.

R4 Somme de valeurs analogiques.

R5 Moyenne de valeurs analogiques.

R6 Minimum et Maximum de valeurs analogiques.

Filtrage sur la valeur

R7 Dépassement de seuil critique.
Paramètres `ID`, `START` et `END`.

```
SELECT Date, Value FROM ana, anavalues
WHERE ana.AnaId = anavalues.AnaId AND ana.AnaId = [ID]
  AND Date BETWEEN [START] AND [END]
  AND Value > ana.Threshold2;
```

R8 Dépassement de valeur.

Calcul d'agrégats avec filtrage sur plusieurs séries

R9 Repère avec valeurs anormales.
Récupère le repère dont les valeurs ne sont, le plus souvent, pas comprises entre ses deux seuils critiques entre deux dates.
Paramètres `START` et `END`.

```
SELECT Label, count(*) as count FROM ana, anavalues
WHERE ana.AnaId = anavalues.AnaId
  AND Date BETWEEN [START] AND [END]
  AND (Value > Threshold2 OR Value < Threshold1)
GROUP BY ana.AnaId ORDER BY count DESC LIMIT 1;
```

Opérations sur les dates

R10 Vérification de la période d'échantillonnage.
Récupère les repères dont la période d'échantillonnage ne respecte pas la valeur `Interval` donnée dans la table `ana`.
Paramètres `START` et `END`.

```
SELECT values.AnaId, count(*) as count FROM ana,
( SELECT D1.AnaId, D1.Date,
         min(D2.Date-D1.Date) as Interval
  FROM anavalues D1, anavalues D2
  WHERE D2.Date > D1.Date AND D1.AnaId = D2.AnaId
    AND D1.Date BETWEEN [START] AND [END]
  GROUP BY D1.AnaId, D1.Date
) as values
WHERE values.AnaId = ana.AnaId
  AND values.Interval > ana.Interval
GROUP BY values.AnaId ORDER BY count DESC LIMIT 1;
```

Extraction de valeurs courantes Ces deux requêtes ne possédant pas de paramètre, elles ne sont exécutées qu'une seule fois pour éviter d'utiliser le cache sur les requêtes – stocker leur résultats pour ne pas avoir à les réévaluer. Elles permettent de récupérer les valeurs les plus récentes pour chaque repère de la base.

R11.1 Valeurs analogiques courantes.

```
SELECT AnaId, Value FROM anavalues,
WHERE (AnaId, Date) IN
( SELECT AnaId, max(Date) FROM anavalues
  GROUP BY AnaId )
ORDER BY AnaId;
```

R11.2 Valeurs booléennes courantes.

4 Expérimentation du benchmark

Ce benchmark a été exécuté avec le progiciel d'historisation InfoPlus.21, le SGBDR MySQL et le SGBD NoSQL Berkeley DB. Les progiciels d'historisation sont des solutions propriétaires avec des conceptions distinctes et donc des

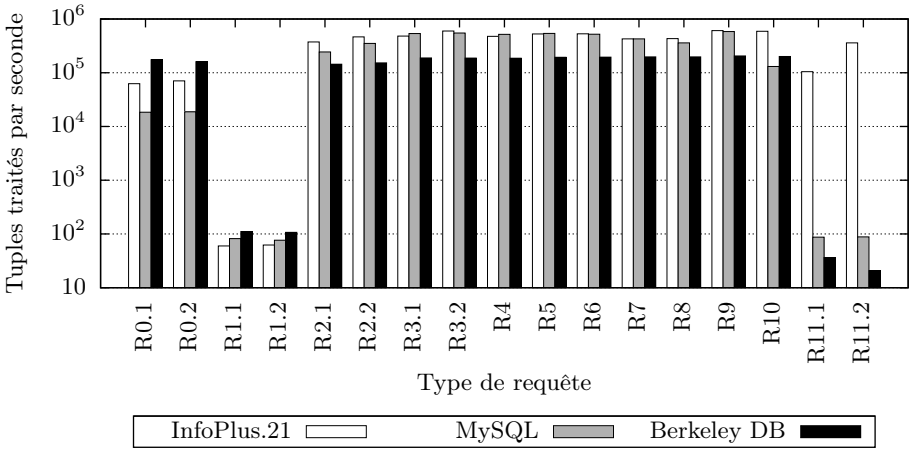

Figure 4. Capacités de traitement

performances différentes. Nous avons choisi l'un des plus répandus, InfoPlus.21, utilisé à EDF dans le domaine nucléaire. Nous avons retenu MySQL pour sa facilité d'utilisation et sa pérennité avec une communauté d'utilisateurs importante, essentiels pour un usage industriel. Enfin, nous avons choisi l'entrepôt de paires clé-valeur ordonnées Berkeley DB pour nos expérimentations. Cette catégorie de système NoSQL est particulièrement adaptée aux requêtes usuelles basées sur des intervalles de clés (*range queries*). MySQL (avec le moteur de stockage InnoDB) comme Berkeley DB ont été optimisés pour ce contexte d'utilisation, notamment en désactivant la gestion de transactions.

Pour chaque système, le serveur de test possède un processeur Xeon Quad Core [3] E5405 2.0 GHz, 3 GB de RAM et trois disques durs de 73 GB 10K.

500 000 000 tuples de données sont insérés pour chaque type – analogique et booléen – ce qui correspond à 11.5 GB sans compression. 1 000 000 mises à jour sont ensuite effectuées, suivies de 1 000 requêtes R2 à R8, 100 requêtes R9 et R10, et 1 requête R11.1 et R11.2. Les paramètres des requêtes sont générés de manière à accéder, en moyenne, à 100 000 tuples pour les requêtes R2 à R8, et 10 000 000 tuples pour les requêtes R9 et R10.

4.1 Résultats

Les capacités de traitement sont données dans la figure 4, en indiquant le nombre de tuples traités à la seconde.

Pour l'analyse des résultats expérimentaux, il est possible de regrouper les requêtes présentant des performances similaires par catégories. On distingue quatre

3. Pour ces tests, seul un cœur est activé a cause du manque d'optimisation de notre progiciel d'historisation pour des insertions multi-threadées.

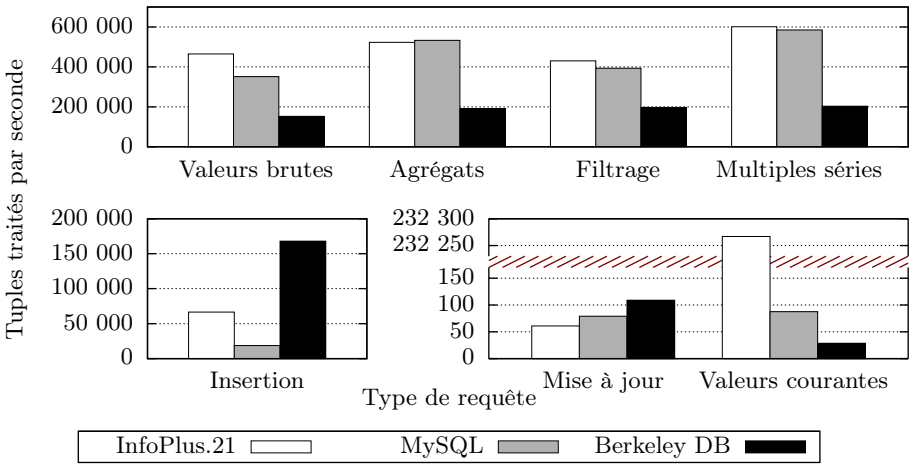

Figure 5. Capacités de traitement par catégorie

catégories de requêtes : les insertions (R0.1 et R0.2), les mises à jour (R1.1 et R1.2), les *range queries* (R2.1 à R10) et l'extraction des valeurs courantes (R11.1 et R11.2). Parmi les *range queries*, on identifie quatre types de requêtes : R2.1 et R2.2 pour les valeurs brutes, R3, R4, R5 et R6 pour les agrégats, R7 et R8 pour le filtrage, et R9 et R10 pour l'interrogation de multiples repères.

La figure 5 donne alors un aperçu des différences de performances en regroupant les requêtes similaires. Pour cette analyse, la requête R2.1 n'est pas prise en compte à cause des activités en arrière plan dues aux mises à jour, qui dégradent les performances de cette requête. La requête R10 est également ignorée parmi les résultats MySQL, à cause de mauvaises performances probablement dues à la définition des procédures stockées utilisées pour son traitement.

Comme on pouvait s'y attendre, les progiciels d'historisation gèrent les insertions efficacement par rapport aux SGBDR : InfoPlus.21 atteint 66 500 insertions par seconde (ips), soit environ 3.2 fois mieux qu'InnoDB et ses 20 500 ips. Toutefois, Berkeley DB atteint 168 000 ips, soit 2.5× mieux qu'InfoPlus.21. Ce résultat est à relativiser car Berkeley DB est utilisé en tant que librairie, sans mécanismes de communication entre processus, ce qui peut avoir un impact important sur les performances par rapport à MySQL ou InfoPlus.21.

L'extraction des valeurs courantes (R11.1 et R11.2) est le deuxième point fort prévisible des progiciels d'historisation, étant donnée leur conception particulière où les valeurs les plus récentes sont conservées en mémoire. Cette opération est plus rapide de plusieurs ordres de grandeur par rapport à MySQL (×1 850) ou Berkeley DB (×6 140).

Pour les autres requêtes d'analyse [4], MySQL et InfoPlus.21 présentent des performances très proches, avec au plus de 25% de différence entre ces deux

4. R2.1 et R10 exclues.

systèmes. Leurs capacités de traitement sont comprises entre 350 000 et 610 000 tuples par seconde selon les requêtes. Berkeley DB, de part la simplicité de son interface d'accès, présente des performances homogènes, mais sensiblement moins bonnes, autour de 190 000 tuples par seconde.

5 Conclusion

L'historisation de données industrielles est un contexte applicatif pour lequel les SGBDR sont peu utilisés, au profit de produits "de niche" spécialisés pour les traitements spécifiques des applications sous-jacentes. Les raisons de cette segmentation du marché sont historiques : elles se basent sur les contraintes de performances auxquelles les systèmes d'historisation sont soumis. Pourtant, même si ces progiciels d'historisation ont été conçus pour supporter une charge particulièrement importante en insertion de données, les résultats du benchmark que nous avons proposé mettent en évidence des performances du même ordre de grandeur pour les SGBDR et systèmes NoSQL : avec une configuration adaptée, MySQL et Berkeley DB pourraient supporter les charges identifiées à EDF et donc pourraient être compétitifs du point de vue "gestion de données".

Cependant, la comparaison fonctionnelle met en avant la force des progiciels d'historisation : ces fonctionnalités « métier » tendent à prendre le pas sur les performances. Les clients métiers et la facilité d'intégration (communication avec les autres systèmes, configuration originale adaptée au contexte, etc.) constituent une différence importante avec les autres catégories de SGBD. Néanmoins, pour s'affranchir du coût de licence ou pour intégrer le système d'historisation dans un environnement où les ressources matérielles sont limitées voire où un progiciel d'historisation n'est pas utilisable (système d'exploitation non supporté, etc.), un SGBD conventionnel pourrait être mis en œuvre dans ce contexte applicatif.

Références

1. A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. S. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. Linear Road : A Stream Data Management Benchmark. In *VLDB'04 : 30th International Conference on Very Large Data Bases*, pages 480–491, 2004.
2. Aspen Technology. *Database Developer's Manual*, 2007. Version 2006.5.
3. Invensys Systems. *Wonderware Historian 9.0 High-Performance Historian Database and Information Server*, 2007.
4. M. A. Olson, K. Bostic, and M. I. Seltzer. Berkeley DB. In *FREENIX'99 : 1999 USENIX Annual Technical Conference, FREENIX Track*, pages 183–191, 1999.
5. Oracle. *MySQL 5.5 Reference Manual*, 2011.
6. OSIsoft. *PI Server System Management Guide*, 2009. Version 3.4.380.
7. Transaction Processing Performance Council. *TPC Benchmark C Standard Specification*, 2007.
8. Transaction Processing Performance Council. *TPC Benchmark H Standard Specification*, 2008.