Évaluation des performances des SGBDOO : un modèle de simulation générique

Jérôme Darmont

Laboratoire d'Informatique (LIMOS)
Université Blaise Pascal – Clermont-Ferrand II
Complexe Scientifique des Cézeaux
63177 Aubière Cedex

Mél : jerome.darmont@libd2.univ-bpclermont.fr
☎ 04-73-40-77-68
Fax : 04-73-40-74-44

Résumé : Les performances des Systèmes de Gestion de Bases de Données Orientés Objets (SGBDOO) restent un problème d'actualité, à la fois pour les concepteurs et pour les utilisateurs. L'approche la plus répandue pour évaluer ces performances est l'expérimentation, qui consiste à mesurer directement les performances d'un système existant. Or, la simulation aléatoire à événements discrets présente divers avantages dans ce contexte (évaluation *a priori*, souplesse, faible coût…), mais reste très peu utilisée dans le domaine des bases de données orientées objet. L'objectif de cet article est de présenter un modèle de simulation générique, VOODB, permettant l'évaluation des performances de différents types de SGBDOO. Afin de valider cette approche, nous avons simulé le fonctionnement du SGBDOO O_2 et du gestionnaire d'objets persistants Texas. Les résultats de simulation obtenus ont été comparés avec les performances des systèmes réels, mesurés par expérimentation dans les mêmes conditions, grâce au banc d'essais OCB. Les deux séries de résultats sont apparues cohérentes.

Mots clés : Bases de données orientées objet, évaluation de performance, expérimentation, simulation aléatoire à événements discrets, bancs d'essais

---

1. Introduction

Quelque temps après l'émergence des premiers Systèmes de Gestion de Bases de Données Orientés Objet (SGBDOO) est apparu le besoin d'outils de mesure de performance pour les gestionnaires d'objets persistants, besoin exprimé par des appels répétés lors de conférences [ATKI90]. L'idée sous-jacente à ces appels était que l'évaluation de performance est un composant essentiel dans le développement de gestionnaires d'objets persistants bien conçus et efficaces. De telles mesures sont indispensables, d'une part, pour valider ou réfuter les suppositions des concepteurs quant au comportement réel des SGBDOO et de la charge de travail qu'ils supportent et, d'autre part, pour disposer d'éléments de comparaison en ce qui concerne l'efficacité réelle de différentes technologies. Il existe donc deux raisons bien distinctes pour mesurer les performances d'un SGBDOO :

- établir quel est le comportement d'un produit afin qu'il puisse être comparé à d'autres par des utilisateurs potentiels ;

- mieux comprendre les mécanismes qui régissent le système afin de pouvoir éventuellement améliorer sa conception.

Deux approches possibles pour évaluer les performances d'un système donné sont l'expérimentation et la simulation aléatoire à événements discrets.

L'expérimentation (mesures de performance sur un système existant) est l'approche la plus naturelle et *a priori* la plus simple à mettre en œuvre. Cependant, il est nécessaire d'acquérir le système à étudier, de l'installer et d'y implanter une base de données significative de son exploitation future. L'expérimentation elle-même peut être longue à mettre en place et les coûts globaux d'investissement et d'exploitation peuvent s'avérer élevés.

Dans certains domaines (systèmes, réseaux, bases de données distribuées…), la simulation aléatoire à événements discrets est depuis longtemps utilisée en complément ou en remplacement de l'expérimentation. Elle ne nécessite pas l'installation (ni l'acquisition) du système étudié et peut même être employée pour étudier un système encore en cours de développement (évaluation de performance *a priori*). L'exécution d'un programme de simulation est en général plus rapide qu'une expérimentation. Les coûts d'investissement et d'exploitation sont faibles. Cependant, cette approche implique la conception d'un modèle du système étudié. La fiabilité des résultats obtenus dépend directement de la qualité et de la validité de ce modèle. La difficulté principale est donc d'élaborer et de valider le modèle de simulation, ce qui est généralement effectué en accord avec une méthodologie de modélisation.

La simulation aléatoire est peu utilisée dans le domaine des bases de données orientées objet. L'objectif de cet article est d'une part de montrer que cette approche reste une alternative valable à l'expérimentation pour évaluer les performances des SGBDOO (notamment *a priori*) et, d'autre part, de proposer un modèle de simulation très générique (appelé VOODB) qui soit réutilisable pour diverses études de performance. Ce modèle a été obtenu par l'application d'une méthodologie de modélisation. Il a été utilisé afin de simuler le fonctionnement des systèmes O_2 [DEUX91] et Texas [SING92], qui sont très différents l'un de l'autre. Les résultats de simulation ont été comparés aux résultats obtenus par expérimentation sur O_2 et Texas à l'aide du banc d'essais OCB [DARM98], qui est un banc d'essais générique qui peut être paramétré pour modéliser des types de bases de données et d'applications très variés. Les deux séries de résultats sont apparues cohérentes.

La suite de cet article est organisée de la façon suivante. La Section 2 établit un état de l'art concernant les bancs d'essais et les modèles de simulation pour SGBDOO. Les Sections 3 et 4 sont consacrées aux outils d'évaluation de performance que nous avons utilisés : le banc d'essais OCB et le modèle de simulation VOODB. La Section 5 présente les expériences que nous avons menées. Nous concluons finalement cet article et présentons quelques perspectives de recherche dans la Section 6.

2. État de l'art sur l'évaluation des performances des SGBDOO

Nous présentons dans cette section un bref état de l'art sur les deux principales approches qui peuvent être envisagées pour évaluer les performances des SGBDOO : l'expérimentation par banc d'essais et la simulation aléatoire à événements discrets. Nous discutons des travaux existants dans chaque domaine.

2.1. Bancs d'essais orientés objet

L'expérimentation par un banc d'essais consiste à effectuer une série de tests sur un SGBD existant, dans le but d'estimer ses performances dans des conditions données. Les bancs d'essais sont généralement utilisés pour comparer les performances globales des SGBD, mais aussi pour illustrer les avantages d'un système ou d'un autre dans une situation donnée, ou pour déterminer une configuration matérielle optimale (taille du cache mémoire, nombre de disques, etc.) pour un SGBD et/ou une application donnés. Typiquement, un banc d'essais est composé de deux éléments principaux :

- une base de données (schéma conceptuel et méthode de génération) ;
- une charge, c'est-à-dire un ensemble d'opérations à effectuer sur la base de données (par exemple, différentes sortes de requêtes) et un protocole détaillant de déroulement de l'exécution de ces opérations.

Dans le monde des SGBD relationnels, le *Transaction Processing Council* (TPC) joue un rôle prépondérant en matière de bancs d'essais. Cet institut a pour mission de définir des bancs d'essais standards, de vérifier leur application correcte par les compagnies qui souhaitent voir leur produit testé et de publier régulièrement les résultats de ces tests de performance. En revanche, il n'existe pas de banc d'essais standard pour les SGBDOO, même si les plus souvent cités et utilisés, OO1 [CATT91], HyperModel [ANDE90] et OO7 [CARE93], font office de standard de fait. Justitia [SCHR94] est également intéressant pour son approche multi-utilisateurs opérationnelle.

OO1 (*Object Operations 1*), aussi appelé « the Cattell benchmark », est né au début des années 90, alors qu'il n'existait pas de banc d'essais approprié pour évaluer les performances des applications d'ingénierie (CAO, PAO, AGL…). C'est un banc d'essais simple et facile à implémenter (schéma constitué de deux classes ; trois types d'opérations). Il a été employé pour évaluer les performances de nombreux SGBD relationnels et orientés objet.

Le banc d'essais HyperModel (aussi appelé banc d'essais Tektronix dans la littérature) possède à la fois un schéma plus riche (basé sur un modèle hypertexte étendu) et une gamme plus importante d'opérations (une vingtaine) qu'OO1. Cela le rend potentiellement plus pertinent pour mesurer les performances des applications d'ingénierie utilisant des bases de données orientées objet, mais la notion d'objet complexe reste encore peu exploitée dans HyperModel.

OO7 est plus récent qu'OO1 et HyperModel et, par conséquent, il s'appuie sur ces derniers pour proposer un banc d'essais plus complet et pour simuler l'exécution de transactions variées sur une base de données diversifiée. Il a été conçu pour évaluer les performances des techniques et des algorithmes d'implantation des SGBDOO d'une manière plus générique que ses aînés et pour remédier à leurs carences, notamment en terme de complexité des objets manipulés et d'accès associatifs à ces objets. Le schéma d'OO7 comprend une dizaine de classes organisées en hiérarchies complexes. La charge d'OO7 est constituée de quinze types d'opérations.

Pour terminer, Justitia s'appuie sur la constatation que les bancs d'essais antérieurs ne présentent pas de fonctionnalité multi-utilisateurs satisfaisante (ce qu'il est important de tester dans un contexte client-serveur) et ne savent pas tester les capacités d'un SGBDOO à réorganiser sa base de données. Il prend donc en

compte de façon implicite des utilisateurs multiples. Le schéma de la base d'objets de Justitia et sa gamme d'opérations sont néanmoins plus limités que ceux d'HyperModel ou OO7.

Les bancs d'essais existants ont été conçus pour évaluer les performances d'applications d'ingénierie. Même s'ils restent assez généraux dans l'esprit, ils manquent de généricité et s'adaptent moins bien à d'autres domaines comme les finances, les télécommunications et le multimédia [TIWA95]. C'est pourquoi, parallèlement, ont été développés plusieurs bancs d'essais dédiés à l'étude de domaines particuliers (regroupement d'objets, bases de données actives…).

Par ailleurs, l'adaptation des bancs d'essais OO1, HyperModel ou OO7 à un contexte donné (par exemple, l'étude du regroupement d'objets) nécessite la conception d'un banc d'essais dérivé permettant de prendre en compte des éléments spécifiques. Il n'est pas possible de simplement paramétrer ces bancs d'essais pour les adapter à de tels besoins.

2.2. Modèles de simulation de SGBDOO

L'utilisation de la simulation aléatoire à événements discrets est peu répandue dans le domaine des bases de données objet. La principale difficulté consiste à concevoir un « bon » modèle fonctionnel du système étudié. Un tel modèle doit être représentatif des performances à évaluer, avec le degré de précision requis. Déduire les caractéristiques significatives d'un système et les traduire dans un langage de simulation reste souvent une affaire de spécialistes.

La littérature traitant de modèles de simulation de SGBDOO n'est pas très abondante. En général, le système étudié est disponible et l'approche par expérimentation est naturellement préférée à l'approche par simulation. Néanmoins, certains concepteurs de techniques d'optimisation ont recours à la simulation aléatoire à événements discrets afin de valider leurs propositions. Par exemple, un modèle dédié écrit en PAWS a été proposé par [CHAN89] pour valider une technique de regroupement d'objets et une technique de gestion de cache mémoire associée, dans un contexte CAO. L'objectif était de déterminer comment différentes méthodes d'optimisation influencent les performances lorsque les caractéristiques de l'application qui accède aux données varient, ainsi que les relations entre le groupement des objets et des paramètres comme le taux de lecture/écriture. La simulation aléatoire à événements discrets a également été utilisée par [GAY97] pour comparer l'efficacité de stratégies de regroupement pour SGBDOO. Les modèles proposés étaient codés en SLAM II.

D'autres approches de simulation, qui ne relèvent pas de la simulation aléatoire à événements discrets, présentent des caractéristiques intéressantes. Par exemple, les travaux de [CHEN91] concernent l'efficacité de différents schémas de regroupement d'objets lorsque des paramètres comme le taux de lecture/écriture varient. Les auteurs se sont plus particulièrement centrés sur la modélisation des disques durs. Le logiciel CLAB (*Clustering Laboratory*) [TSAN92] a, lui, été conçu pour comparer des algorithmes de partitionnement de graphes appliqués au regroupement d'objets sur disque. Il est constitué d'un ensemble d'outils Unix programmés en C++ et qui peuvent être assemblés selon différentes configurations. D'autres travaux encore, qui relèvent des domaines des bases de données réparties et parallèles, peuvent s'avérer utiles, comme les modèles de charge de [HE93, BATE95].

Ces différentes études amènent les observations suivantes. Premièrement, la plupart des modèles de simulation existants sont dédiés à un seul type de problème (le groupement d'objets, par exemple). De plus, ils n'exploitent à chaque fois qu'un seul type de SGBDOO, dont les caractéristiques ne sont pas toujours clairement spécifiées, alors que diverses architectures influençant les performances sont possibles (serveur d'objets, serveur de pages, etc.). Par ailleurs, la précision des spécifications de ces modèles varie beaucoup, si bien qu'il n'est pas toujours aisé de les reproduire d'après les publications disponibles. Finalement, aucun de ces modèles n'a fait l'objet, à notre connaissance, d'un processus de validation. La fiabilité des résultats de simulations obtenus n'est donc pas garantie.

3. Le banc d'essais OCB

Nous avons dans un premier temps proposé un banc d'essais appelé OCB (*the Object Clustering Benchmark*) afin d'estimer spécifiquement l'impact de différentes politiques de regroupement d'objets sur les performances globales des SGBDOO. Ce banc d'essais trouve sa motivation dans les lacunes présentées par les bancs d'essais usuels dans ce domaine. Ces bancs d'essais adoptent en effet des schémas de base de données relativement simples, que ne peuvent exploiter pleinement la plupart des algorithmes de groupement d'objets, et leur charge n'est pas adaptée car elle prend en compte des opérations (telles que des accès aléatoires) qui ne peuvent bénéficier d'aucun regroupement. De plus, leur gamme de paramètres est réduite et ne permet pas de modéliser finement des applications spécifiques.

La version d'OCB présentée dans cet article est une version améliorée de celle présentée dans [DARM98] : le banc d'essais a été raffiné au niveau du schéma de la base de données et sa charge a été largement étendue afin d'obtenir un outil pleinement générique. Les points forts de ce banc d'essais sont :

- sa généricité : il est notamment possible de reproduire avec OCB le comportement des bancs d'essais existants [DARM99] ;
- son adaptabilité : OCB peut être paramétré pour modéliser des types de bases de données et d'applications très variés.

Le code d'OCB (actuellement en C++) est par ailleurs disponible gratuitement, sur demande à l'auteur.

3.1. Base de données d'OCB

La base de données d'OCB est à la fois riche, facile à générer et aisément paramétrable (Figure 1). Elle est constituée de *NC* classes, toutes dérivées de la métaclasse *CLASS*. Une classe est définie par deux paramètres : *MAXNREF*, le nombre maximum de références possibles pour cette classe et *BASESIZE*, la taille de base de la classe (qui est un incrément utilisé pour calculer la taille de ses instances au cours de la génération des graphes d'héritage). Comme différentes références peuvent pointer sur la même classe, les relations 0-N, 1-N et M-N sont implicitement modélisées. Chacune de ces références (*CRef*) possède un type mémorisé dans l'attribut *TRef*. Il existe *NREFT* types de références différents. Un type de référence peut être, par exemple, une sorte d'héritage, une relation d'agrégation ou d'association, etc. Finalement, chaque classe est dotée d'un itérateur qui permet d'accéder à toutes ses instances.

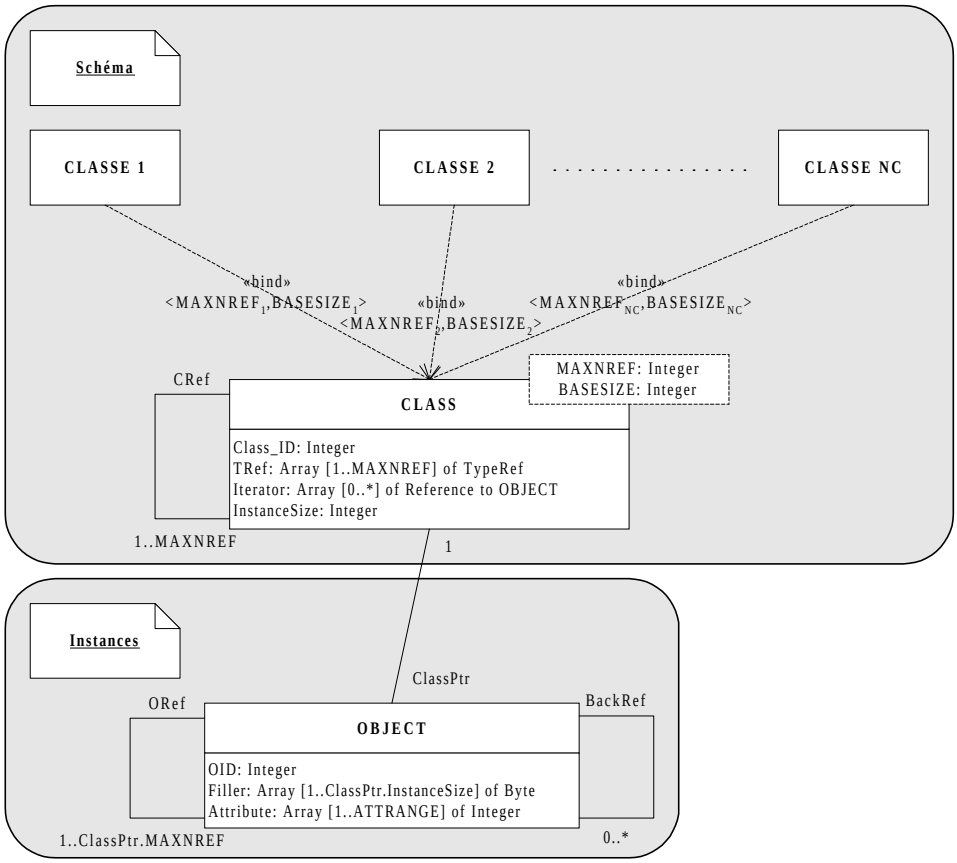

Figure 1 : Schéma de la base de données d'OCB

Chaque objet (instance de la classe *OBJECT*) possède *ATTRANGE* attributs entiers accessibles et modifiables par les transactions du banc d'essais. Une chaîne de caractères de longueur *InstanceSize* simule la taille réelle de l'objet. Chaque objet pointe (*via* les références *ORef*) sur au plus *MAXNREF* objets choisis dans l'itérateur de la classe référencée par la classe mère de l'objet en question, *via* la relation *CRef* correspondante. Pour chaque référence directe *ORef* d'un objet o_i vers un objet o_j, il existe de plus une référence inverse *BackRef* de o_j vers o_i.

Les paramètres qui définissent la base d'objets d'OCB sont récapitulés dans la Table 1.

Paramètre	Libellé	Valeur par défaut
NC	Nombre de classes dans la base de données	50
MAXNREF (i)	Nombre maximum de références, par classe	10
BASESIZE (i)	Taille de base des instances, par classe	50 octets
NO	Nombre total d'objets	20000
NREFT	Nombre de types de références (héritage, agrégation, etc.)	4
ATTRANGE	Nombre d'attributs entiers dans un objet	1
CLOCREF	Localité de référence au niveau des classes	NC
OLOCREF	Localité de référence au niveau des instances	NO
MAXRETRY	Nombre maximum d'essais lors de la connexion des objets	3
DIST1	Loi de distribution aléatoire des types de références	Uniforme
DIST2	Loi de distribution aléatoire des références de classes	Uniforme
DIST3	Loi de distribution aléatoire des objets dans les classes	Uniforme
DIST4	Loi de distribution aléatoire des références d'objets	Uniforme

Table 1 : Paramètres définissant la base de données d'OCB

3.2. Charge d'OCB

Les quatre grands types de transactions utilisées pour constituer la charge d'OCB sont les suivantes. Les paramètres qui les définissent sont récapitulés dans la Table 2.

- *Accès aléatoire :* Accès à *NRND* objets sélectionnés au hasard.
- *Balayage séquentiel :* Lecture des instances d'une classe sélectionnée au hasard (*Balayage simple*). Une *Recherche par valeur* effectue en sus un test sur la valeur de *NTEST* attributs, pour chaque instance accédée.
- *Parcours :* Les opérations de parcours sont divisées en deux types : les *Accès associatifs* (ou *Accès ensemblistes*) et les *Accès navigationnels*, qui correspondent à des parcours en largeur d'abord et des parcours en profondeur d'abord, respectivement. Les accès navigationnels sont subdivisés en trois catégories : les *Parcours simples* (en profondeur d'abord), les *Parcours hiérarchiques* qui suivent toujours le même type de référence et les *Parcours stochastiques* dans lesquels la prochaine référence à suivre est déterminée aléatoirement. Chaque type de parcours démarre à partir d'un objet racine sélectionné au hasard et procède jusqu'à une profondeur prédéterminée. Tous ces parcours peuvent être inversés pour suivre les références inverses des objets.
- *Mise à jour :* Les opérations de mise à jour sont également subdivisées en deux catégories. Les *Évolutions de schéma* gèrent l'insertion et la suppression des objets *CLASS*. Une classe à effacer est sélectionnée au hasard. Les *Évolutions de la base* administrent l'insertion et la suppression des objets *OBJECT*. Un objet à effacer est sélectionné au hasard. Finalement, les *Mises à jour d'attribut* permettent des changements de valeur pour les attributs des objets. Ces objets sont soit choisis aléatoirement (*Mise à jour aléatoire* de *NUPDT* objets sélectionnés au hasard), soit sont les instances d'une classe sélectionnée au hasard (*Mise à jour séquentielle*).

Paramètre(s)	Libellé(s)	Valeur(s) par défaut
NRND	Nombre d'objets accédés par les Accès aléatoires	50
NTEST	Nombre d'attributs testés dans les Recherches par valeur	1
SETDEPTH, SIMDEPTH, HIEDEPTH, STODEPTH	*Profondeur :* Accès ensemblistes, Parcours simples, Parcours hiérarchiques, Parcours stochastiques	3, 3, 5, 50
NUPDT	Nombre d'objets mis à jour dans les Mises à jour aléatoires	50
DIST5, DIST6, DIST7, DIST8, DIST9, DISTA, DISTB	*Loi de distribution aléatoire :* objets des Accès aléatoires, classes des Balayages séquentiels, objets racines de Parcours, classes des Évolutions de schéma, objets des Évolutions de la base, objets des Mises à jour aléatoires, classes des Mises à jour séquentielles	Uniforme
PRND, PSCAN, PRANGE, PSET, PSIMPLE, PHIER, PSTOCH, PCINSERT, PCDEL, POINSERT, PODEL, PRNDUP, PSEQUP	*Probabilité d'occurrence :* Accès aléatoire, Balayage simple, Recherche par valeur, Accès associatif, Parcours simple, Parcours hiérarchique, Parcours stochastique, Insertion de classe, Suppression de classe, Insertion d'objet, Suppression d'objet, Mise à jour aléatoire, Mise à jour séquentielle ($\sum P^* = 1$)	0.1, 0.05, 0.05, 0.2, 0.2, 0.2, 0.1, 0.005 0.005, 0.02, 0.02 0.025, 0.025
COLDN	Nombre de transactions exécutées à froid	1000
HOTN	Nombre de transactions exécutées à chaud	10000
THINK	Temps de latence moyen entre deux transactions	0
CLIENTN	Nombre de clients (utilisateurs de la base)	1
RSEED	Germe du générateur aléatoire	Germe par défaut

Table 2 : Paramètres définissant la charge d'OCB

4. Le modèle de simulation VOODB

Partant des limitations constatées des modèles de simulation de SGBDOO dédiés de la littérature, le modèle de simulation VOODB (*Virtual Object-Oriented Database*) a été conçu de façon à être plus générique. D'autre part, il a été obtenu par l'application d'une méthodologie de modélisation dont l'objectif est de produire des modèles génériques de simulation fiables et réutilisables. VOODB intègre finalement le banc d'essais OCB, ce qui permet de bénéficier d'un modèle de charge également générique.

4.1. Méthodologie de modélisation

Les SGBDOO sont des systèmes complexes, si bien que modéliser leur fonctionnement en vue d'étudier leurs performances peut s'avérer être également une tâche complexe. C'est pourquoi une méthodologie de modélisation a été utilisée pour produire VOODB. C'est une extension générique de l'approche classique de modélisation, dont les principes ont été définis depuis longtemps par divers auteurs [SARG79, NANC81, BALC92, KELL97] et qui consiste à formaliser la connaissance informelle concernant un système au sein d'un *modèle de connaissance* (incluant un *modèle de charge*) dédié à ce système. Ceci permet de faire abstraction des contraintes liées à l'environnement de simulation employé et facilite la communication entre les experts du système et les experts en modélisation. C'est seulement dans un second temps que le modèle de connaissance est traduit en un programme de simulation.

L'approche que nous préconisons étend le champ d'étude à toute une classe de systèmes (Figure 2). Le modèle de connaissance doit donc être paramétrable et modulaire. Nous proposons également la caractérisation séparée du modèle de charge, ce qui permet la réutilisation de modèles de charges existants dans des bancs d'essais comme OO1, OO7 ou OCB. Le programme de simulation générique (*modèle d'évaluation*) est obtenu de manière systématique. Son architecture modulaire est le résultat des deux modèles sur lesquels il est basé. Le programme de simulation final pour une étude de cas donnée est obtenu par instanciation du modèle d'évaluation générique. Cette approche garantit une bonne réutilisabilité et permet des gains de temps importants lors de la phase d'analyse du système.

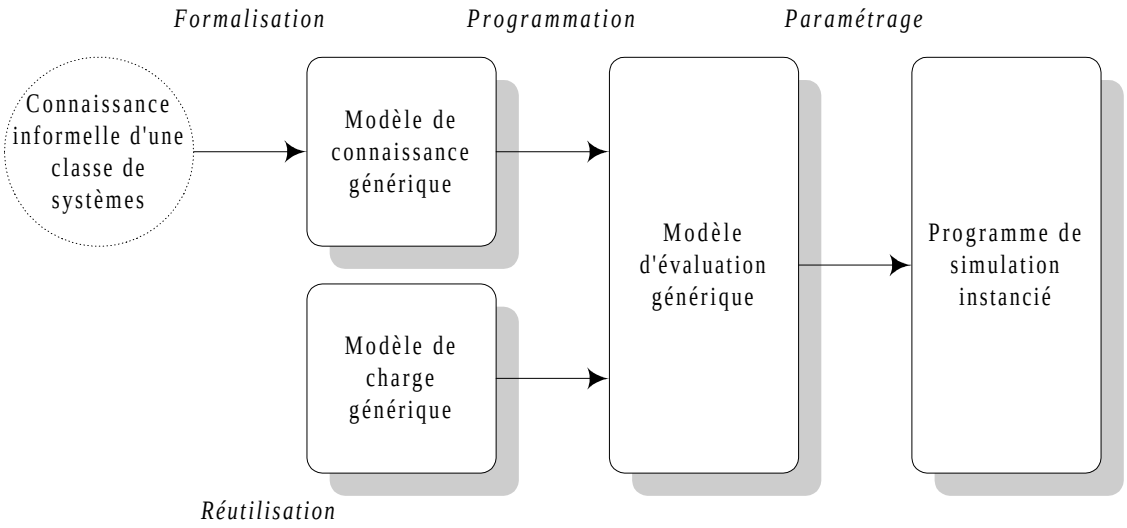

Figure 2 : Méthodologie de modélisation

4.2. Modèle de connaissance de VOODB

Le modèle de connaissance spécifie le fonctionnement de VOODB. Il décrit l'exécution des transactions au sein d'un SGBDOO (Figure 3).

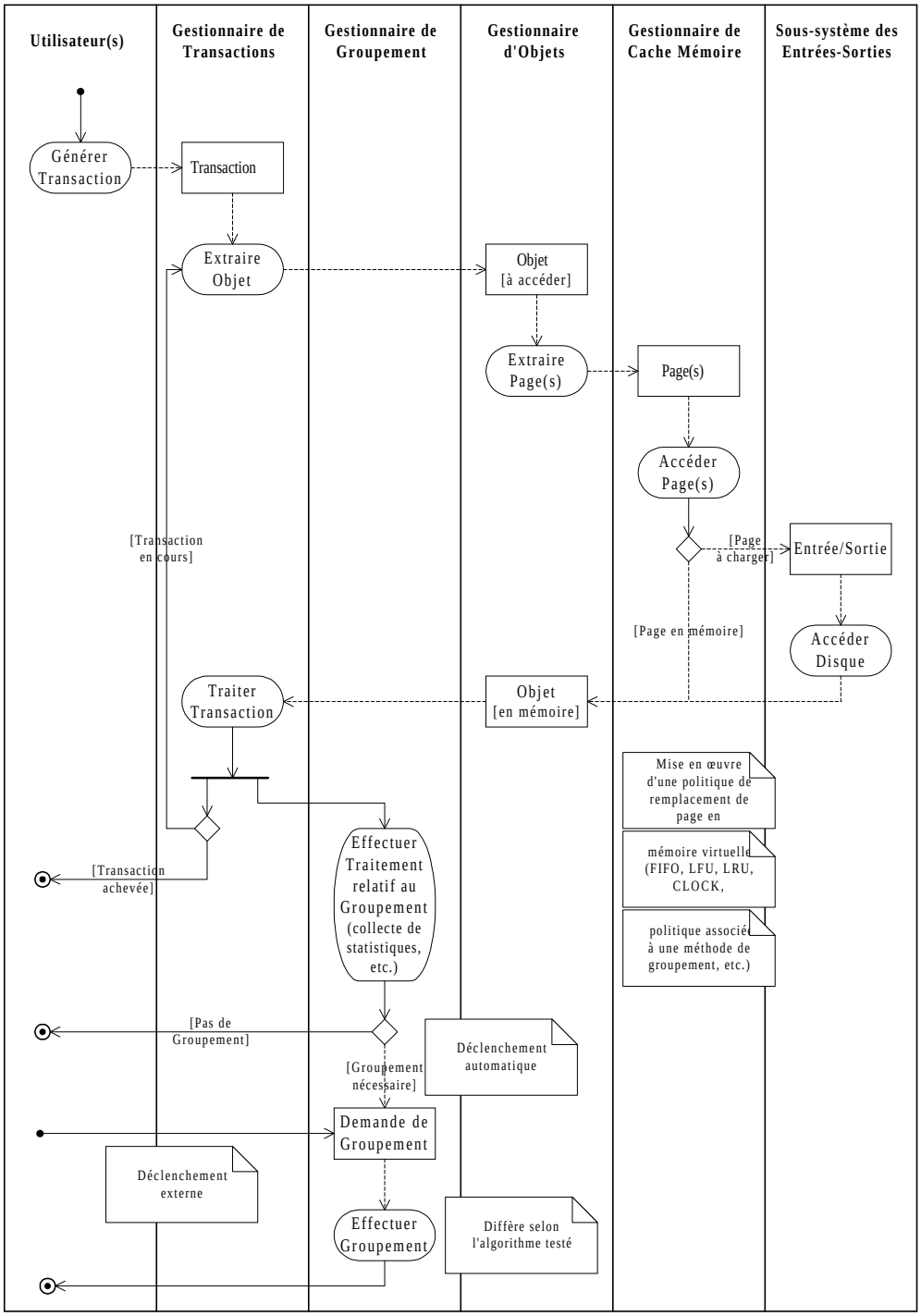

Figure 3 : Modèle de connaissance de VOODB

Ces transactions (les *parcours* du banc d'essais OCB) sont générées par les *Utilisateurs*, qui les soumettent après un temps de latence prédéfini au *Gestionnaire de Transactions*. Ce dernier détermine les objets qui doivent être chargés en mémoire pour traiter la transaction courante et effectue les opérations nécessaires à ce traitement. Chacun des objets nécessaire à une transaction est requis par le *Gestionnaire de Transactions* auprès du *Gestionnaire d'Objets*, qui recherche sur quelle page disque est stocké l'objet en question. La page

est ensuite requise par le *Gestionnaire d'Objets* auprès du *Gestionnaire de Cache Mémoire*, qui vérifie si la page est présente en mémoire vive. Si elle ne s'y trouve pas, une requête de chargement de page est transmise par le *Gestionnaire de Cache Mémoire* au *Sous-système des Entrées-Sorties* qui est chargé de gérer les accès disque physiques. Le *Gestionnaire de Cache Mémoire* est également chargé de mettre en œuvre une stratégie de remplacement des pages dans le cache. Lorsqu'une opération unitaire sur un objet est terminée, le *Gestionnaire de Groupement* peut, si nécessaire, mettre à jour des statistiques d'utilisation de la base de données. Une analyse de ces statistiques peut déclencher un regroupement des objets, qui est alors également exécuté par le *Gestionnaire de Groupement*.

Une telle réorganisation de la base de données peut également être déclenchée de façon externe par les Utilisateurs. Les seuls traitements qui diffèrent lorsque deux algorithmes de groupement différents sont testés sont ceux effectués par le *Gestionnaire de Groupement*. Les autres traitements restent les mêmes quel que soit le modèle d'évaluation.

Le modèle de connaissance est hiérarchique. Chacune de ses activités (⬭) peut être détaillée, comme cela est illustré dans la Figure 4 pour la règle de fonctionnement « Accéder Disque ». Il est spécialement adapté à l'étude de politiques de groupement d'objets, dont le comportement est difficile à prévoir *a priori*.

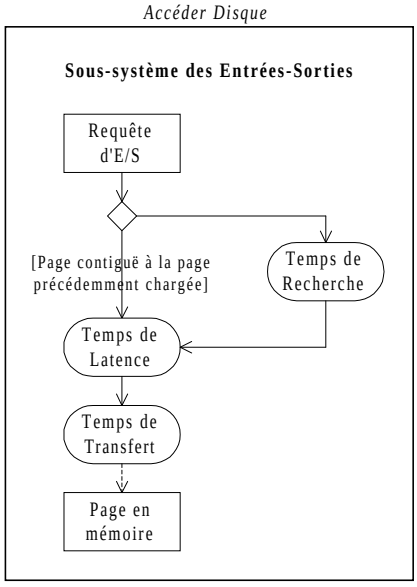

Figure 4 : Détail de la règle de fonctionnement « Accéder Disque »

Il prend également en compte différentes stratégies de remplacement de pages dans le cache mémoire. Cependant, d'autres techniques d'optimisation peuvent également y être inclues. Il suffit pour cela d'ajouter des fonctionnalités au modèle, sous la forme de nouvelles « lignes d'eau » (*swimlanes*).

Par ailleurs, les ressources physiques qui apparaissent en tant que lignes d'eau dans ce modèle de connaissance peuvent être qualifiées de *ressources actives*, puisqu'elles effectuent des traitements. Néanmoins, le système comprend également des *ressources passives*, qui n'effectuent pas de traitement direct, mais sont utilisées par les ressources actives. Ces ressources passives n'apparaissent pas dans la Figure 3, mais doivent cependant être listées de façon exhaustive (Table 3).

Ressource passive
Processeur et mémoire primaire dans une architecture centralisée ou *processeur et mémoire primaire du serveur* dans une architecture client-serveur
Processeur et mémoire primaire des clients dans une architecture client-serveur
Contrôleur de disque et mémoire secondaire du serveur
Base de données : son accès concurrent est géré par un gestionnaire d'ordonnancement, qui applique une politique d'ordonnancement des transactions en fonction du niveau de multiprogrammation.

Table 3 : Ressources passives de VOODB

4.3. Modèle d'évaluation de VOODB

Le modèle d'évaluation générique est la traduction du modèle de connaissance dans un formalisme mathématique ou dans un langage de programmation. Cette traduction s'effectue de façon quasi-automatique. Dans cette étude, les modèles d'évaluation sont des modèles de simulation aléatoire à événements discrets, qui sont bien adaptés à l'étude du comportement transitoire d'un SGBD. Les programmes de simulation adaptés à une étude donnée sont obtenus par paramétrage du modèle d'évaluation générique. L'architecture de notre modèle d'évaluation générique est présentée dans la Figure 5.

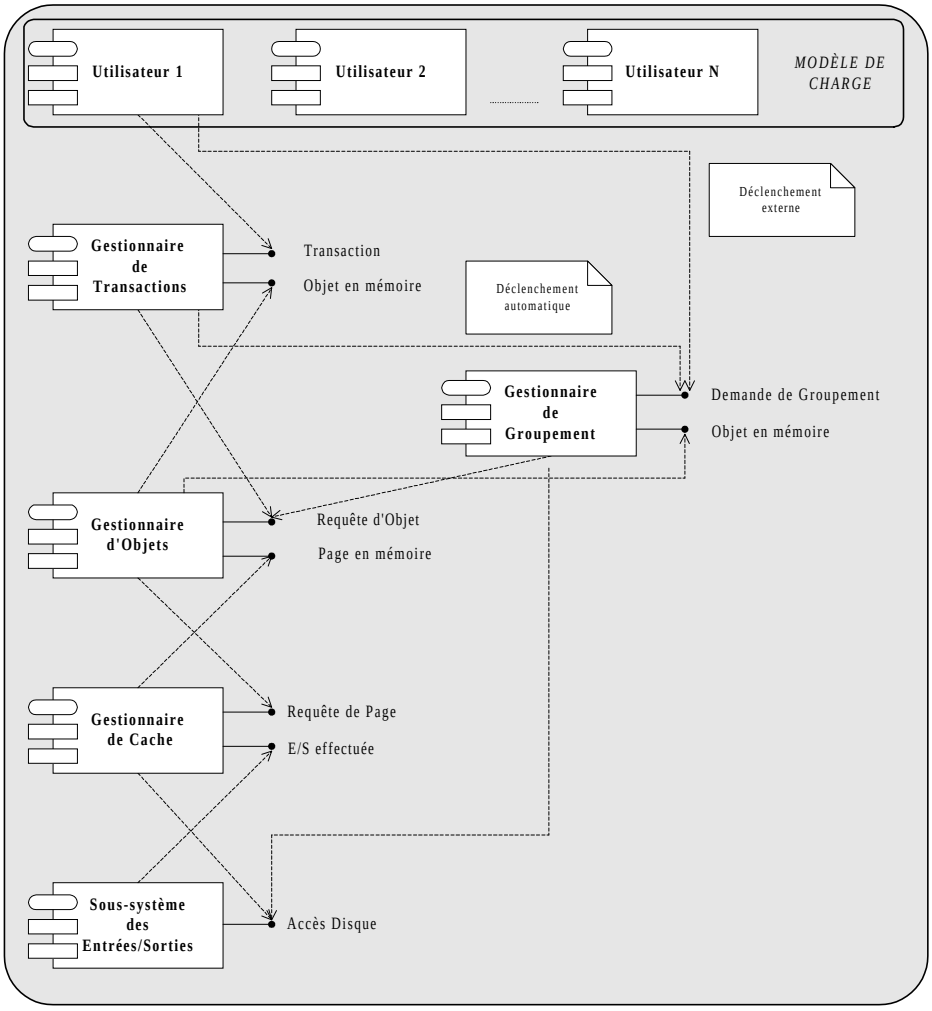

Figure 5 : Architecture du modèle d'évaluation

La traduction du modèle de connaissance est effectuée comme suit :

- chaque ressource active (ligne d'eau) devient un composant du programme de simulation (c'est-à-dire une classe) ;

- chaque objet (▭) devient une interface de ces composants (il est utilisé en tant que paramètre des messages qui sont passés entre deux classes) ;
- chaque activité (⬭) devient une méthode d'un composant.

Les ressources passives n'apparaissent toujours pas de façon explicite dans ce diagramme, bien qu'elles soient évidemment présentes dans le modèle d'évaluation. Ces ressources passives sont des classes qui possèdent deux méthodes principales : la réservation et la libération de la ressource.

Le modèle de charge est intégré au modèle d'évaluation au niveau des utilisateurs. Il représente le comportement des utilisateurs.

Notre modèle générique permet de simuler le fonctionnement de divers types de SGBDOO. Il est notamment adapté aux différentes configurations des architectures client-serveur, qui sont dorénavant standards dans les SGBDOO.

5. Mise en œuvre, expériences et résultats

Différents environnements de simulation sont disponibles actuellement. Ils sont en général fournis avec un simulateur donné. Nous avions dans un premier temps sélectionné le logiciel de simulation QNAP2 [SIMU95] pour implanter VOODB.

Cependant, l'exécution des modèles s'est avérée trop lente pour que nous puissions envisager une campagne de simulation intensive. Ceci est principalement dû au fait que le langage de QNAP2 est interprété. Pour pallier ce problème, nous avons conçu puis validé un environnement de simulation dénommé DESP-C++ [DARM99].

Cependant, le modèle de simulation lui-même doit également être validé, sans quoi nous n'avons aucune garantie que les résultats de simulation sont cohérents avec la réalité. Afin de vérifier que ce modèle est effectivement fiable, nous avons simulé le comportement de deux systèmes gérant la persistance des objets : O_2 [DEUX91] et Texas [SING92]. Nous avons comparé ces résultats à ceux fournis par expérimentation sur les systèmes réels. Pour cela, nous avons utilisé le banc d'essais OCB, qui est utilisable aussi bien pour l'expérimentation qu'en tant que modèle de charge de VOODB. Le choix de Texas provient du fait que nous nous sommes par ailleurs intéressés à l'évaluation des performances de la technique de regroupement d'objets DSTC [BULL96], qui est implantée dans Texas.

Notre but est de mettre en parallèle les deux approches (expérimentation et simulation) afin de valider le modèle de simulation VOODB et non de comparer les performances de Texas et d'O_2. Les tests présentés dans cette section n'ont en effet pas été effectués dans les mêmes conditions (matériels différents, notamment). D'autre part, Texas et O_2 sont des systèmes très dissemblables dans leur philosophie et leurs fonctionnalités. O_2 est un SGBDOO complet assurant l'intégrité des données et leur accès concurrent et sécurisé alors que Texas se positionne uniquement en tant qu'outil de persistance efficace pour le langage C++. L'intérêt de comparer de façon détaillée ces deux systèmes est donc limité.

5.1. Expériences réalisées

5.1.1. Variation de la taille de la base

Nous avons en premier lieu fait varier la taille de la base d'objets (nombre de classes et nombre d'instances dans la base) et mesuré les performances des systèmes étudiés (temps de réponse moyen par transaction et nombre d'entrées-sorties nécessaires à l'exécution de ces transactions). Dans cette série d'expériences, le nombre de classes dans le schéma *NC* est fixé à 20 ou à 50 et le nombre d'instances dans la base *NO* évolue entre 500 et 20000.

5.1.2. Variation de la taille du cache ou de la mémoire primaire

Dans un second temps, nous avons fait varier la taille de la mémoire cache du serveur (dans le cas d'O_2) ou de la mémoire vive disponible (dans le cas de Texas), afin de constater les effets sur les performances du système (temps de réponse moyen et nombre moyen d'entrées-sorties). L'objectif est également de simuler la réaction du système lorsque le rapport (taille mémoire / taille base d'objets) diminue. Comme nous répliquons plusieurs fois nos expériences afin d'obtenir des résultats significatifs, générer des bases trop grosses de façon répétitive aurait eu un coût prohibitif. Par contre, il est facile de jouer sur la taille du cache ou de la mémoire vive.

Sous O_2, la taille du cache du serveur est spécifiée grâce à des variables d'environnement. Notre version de Texas quant à elle est implantée sous Linux. Il est possible d'indiquer au démarrage du système d'exploitation la taille de la mémoire vive disponible. La taille du cache ou de la mémoire a varié dans ces expériences de 8 Mo à 64 Mo. La taille de la base d'objets était, elle, fixe (50 classes, 20000 instances).

5.2. Conditions expérimentales

5.2.1. Conditions d'expérimentation sur les systèmes réels

Le serveur O_2 que nous utilisons (version 5.0) est installé sur une station de travail biprocesseur IBM RISC 6000 43P240. Chaque processeur est un Power PC 604e 166. La station dispose d'une mémoire centrale de 1 Go (RAM ECC). Le système d'exploitation est AIX version 4. Le cache du serveur O_2 est par défaut configuré à une taille de 16 Mo.

Le prototype Texas que nous utilisons (version 0.5) est installé sur un PC Pentium-II 266 disposant de 64 Mo de mémoire SDRAM. Le système d'exploitation est Linux version 2.0.30. La taille de la partition de *swap* est fixée à 64 Mo.

5.2.2. Conditions de simulation

Nos modèles de simulation fonctionnent sur un PC Pentium-II 266 sous Windows 95 (OSR2) disposant de 64 Mo de mémoire SDRAM. Ils ont été compilés avec le compilateur GNU C++ version 2.7.2.1.

Paramètres de simulation

Afin de simuler le comportement de différents systèmes VOODB doit été paramétré de façon appropriée. Le paramétrage pour les systèmes O_2 et Texas est indiqué en Table 4. Les valeurs de la plupart de ces paramètres ont été déduites des spécifications des systèmes étudiés : classe du système, taille des pages disque, politiques de remplacement de pages dans le cache, de préchargement et de groupement d'objets, placement initial des objets, niveau de multiprogrammation, nombre d'utilisateurs. En revanche, de telles spécifications ne sont pas toujours disponibles pour les autres paramètres. Leurs valeurs ont donc été obtenues par mesures.

Par exemple, nous disposions d'indications sur les performances du disque dur de l'IBM sur lequel est installé O_2. En revanche, nous avons dû utiliser le logiciel *Coretest* [CORE88] pour mesurer celles du disque du PC où est installé Texas. De même, si la taille du cache mémoire d'O_2 est parfaitement connue (c'est un paramètre système), Texas utilise lui la mémoire virtuelle d'Unix. Nous avons donc procédé par essai-erreur pour trouver la bonne valeur de ce paramètre, pour chaque configuration mémoire utilisée, avant d'effectuer nos tests.

Paramètre	Code	Valeur pour O_2	Valeur pour Texas
Classe du système	SYSCLASS	Serveur de pages	Centralisé
Débit réseau	NETTHRU	$+\infty$*	N/A
Taille d'une page disque	PGSIZE	4096 octets	4096 octets
Taille du cache de pages	BUFFSIZE	3840 pages	3275 pages
Politique de remplacement de pages dans le cache	PGREP	LRU	LRU
Politique de préchargement des objets	PREFETCH	Aucune	Aucune
Politique de groupement d'objets	CLUSTP	Aucune	DSTC
Placement initial des objets	INITPL	Séquentiel optimisé	Séquentiel optimisé
Temps de recherche sur disque	DISKSEA	6,3 ms	7,4 ms
Temps de latence disque	DISKLAT	2,99 ms	4,3 ms
Temps de transfert disque	DISKTRA	0,7 ms	0,5 ms
Niveau de multiprogrammation	MULTILVL	10	1
Temps d'acquisition des verrous	GETLOCK	0,5 ms	0
Temps de restitution des verrous	RELLOCK	0,5 ms	0
Nombre d'utilisateurs	NUSERS	1	1

Table 4 : Paramètres définissant les systèmes O_2 et Texas au sein de VOODB

Précision des résultats

Nos résultats de simulation sont obtenus avec des intervalles de confiance à c = 95 %. Pour les déterminer, nous avons utilisé la méthode exposée dans [BANK96]. Pour un échantillon donné, la moyenne $\overline{X}$ et l'écart-type σ sont calculés. La largeur d du demi-intervalle de confiance est donnée par la formule $d = t_{n-1,1-\alpha/2} \cdot \sigma / \sqrt{n}$, où la valeur de t est fixée par la t-distribution de Student, n est le nombre de réplications et $\alpha = 1 - c$. L'intervalle de confiance $[\overline{X} - d, \overline{X} + d]$ contient la valeur de la moyenne avec une probabilité c = 0,95.

Comme nous souhaitons obtenir des valeurs qui ne s'écartent pas de plus de 5 % de la valeur de la moyenne calculée avec une confiance de 95 %, nous avons d'abord effectué une étude pilote avec $n = 10$, puis nous avons calculé le nombre n^* d'itérations supplémentaires à effectuer pour satisfaire la condition à

* Dans notre configuration matérielle, le serveur et le client O_2 se trouvent sur la même machine physique.

l'aide de la formule $n^* = n \cdot \left(d/d^*\right)^2$, où d est le demi-intervalle de confiance de l'étude pilote et d^* le demi-intervalle de confiance pour toutes les réplications (le demi-intervalle désiré).

Nos résultats de simulation ont montré que la précision requise était obtenue pour tous nos critères de performance lorsque $n + n^* \geq 100$, avec une large marge de sécurité. Nous avons donc effectué 100 réplications dans toutes nos expériences. Afin de préserver la clarté des résultats dans les graphes qui suivent, nous n'avons pas fait figurer les intervalles de confiance encadrant les valeurs moyennes obtenues. Ils sont néanmoins calculés en standard par DESP-C++.

5.3. Résultats concernant O_2

5.3.1. Variation de la taille de la base

La Figure 6 et la Figure 7 présentent le nombre d'entrées-sorties globalement nécessaires à l'exécution des transactions et le temps de réponse moyen par transaction, en fonction du nombre d'instances dans la base. L'évolution de ces deux critères de performance est similaire, ce qui est normal, étant donné que le gros des traitements effectués par le système lors de l'exécution du banc d'essais OCB consiste à charger des objets depuis le disque. Nous pouvons également constater que les résultats de simulation et les résultats mesurés sur le système réel diffèrent légèrement en valeur absolue mais affichent clairement la même tendance. Le comportement du modèle de simulation VOODB est donc bien conforme à la réalité.

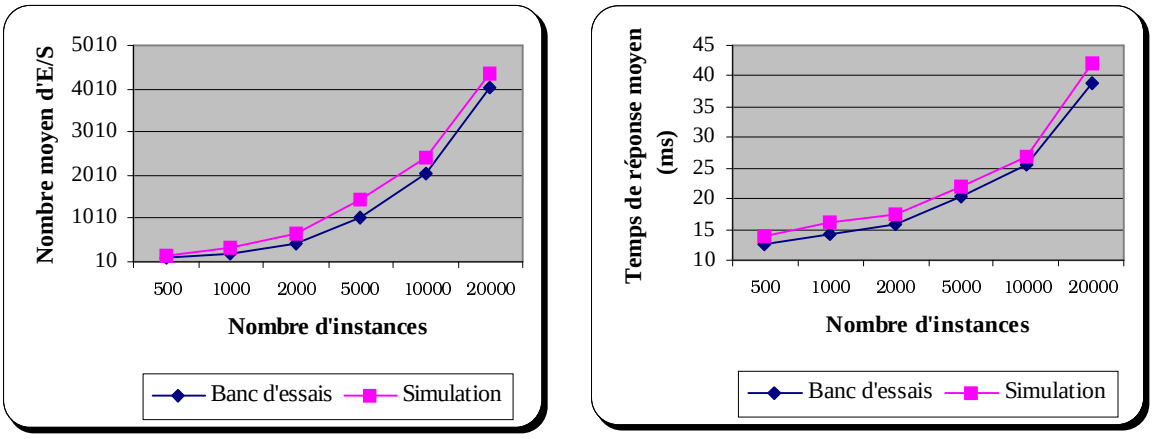

Figure 6 : Nombre d'entrées-sorties et temps de réponse moyens en fonction du nombre d'instances – O_2, 20 classes

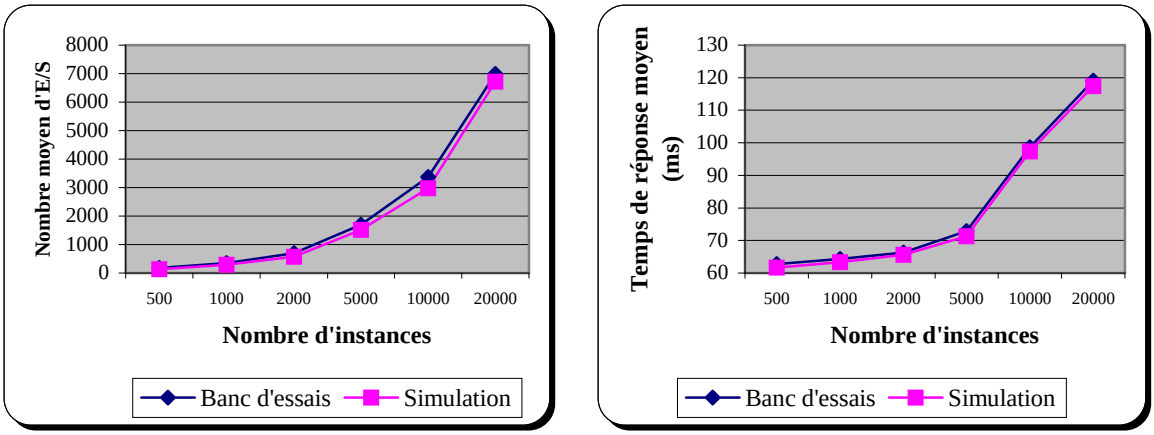

Figure 7 : Nombre d'entrées-sorties et temps de réponse moyens en fonction du nombre d'instances – O_2, 50 classes

5.3.2. Variation de la taille du cache

Les résultats obtenus pour cette expérience en termes de nombre moyen d'entrées-sorties et de temps de réponse moyen sont présentés dans la Figure 8. Celle-ci montre que les performances d'O_2 se dégradent de façon significative lorsque la taille de la base d'objets (28 Mo en moyenne) devient supérieure à la taille du cache. Cette dégradation des performances est linéaire. Cette figure montre par ailleurs que les performances d'O_2 peuvent à nouveau être retrouvées à l'aide de notre modèle de simulation VOODB.

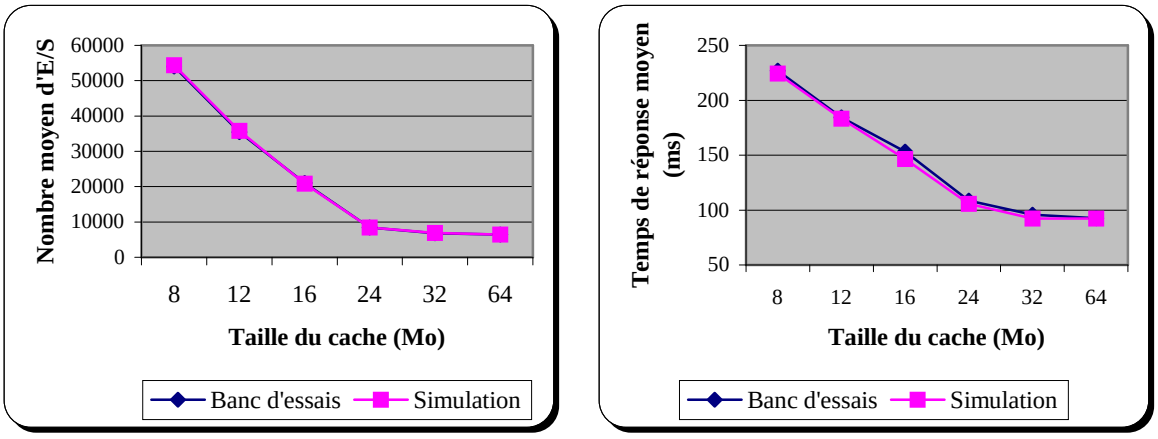

Figure 8 : Nombre d'entrées-sorties et temps de réponse moyens en fonction de la taille du cache (O_2)

5.4. Résultats concernant Texas

5.4.1. Variation de la taille de la base

La Figure 9 et la Figure 10 présentent le nombre d'entrées-sorties globalement nécessaires à l'exécution des transactions et le temps de réponse moyen par transaction, en fonction du nombre d'instances dans la base. Comme dans le cas d'O_2, la corrélation entre ces deux critères de performance apparaît évidente.

Nous pouvons de plus constater que, comme dans le cas d'O_2, les résultats de simulation et ceux mesurés sur le système réel diffèrent légèrement en valeur absolue, mais affichent la même tendance.

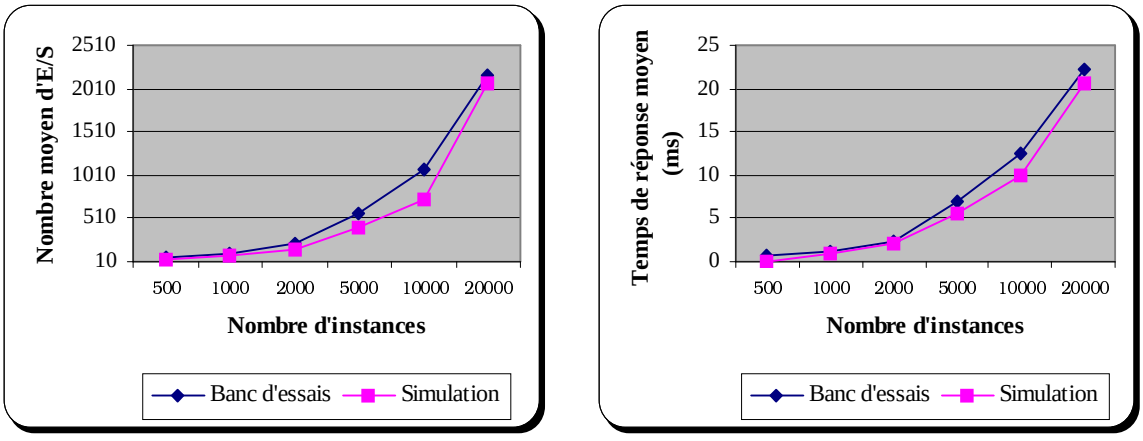

Figure 9 : Nombre d'entrées-sorties et temps de réponse moyens en fonction du nombre d'instances – Texas, 20 classes

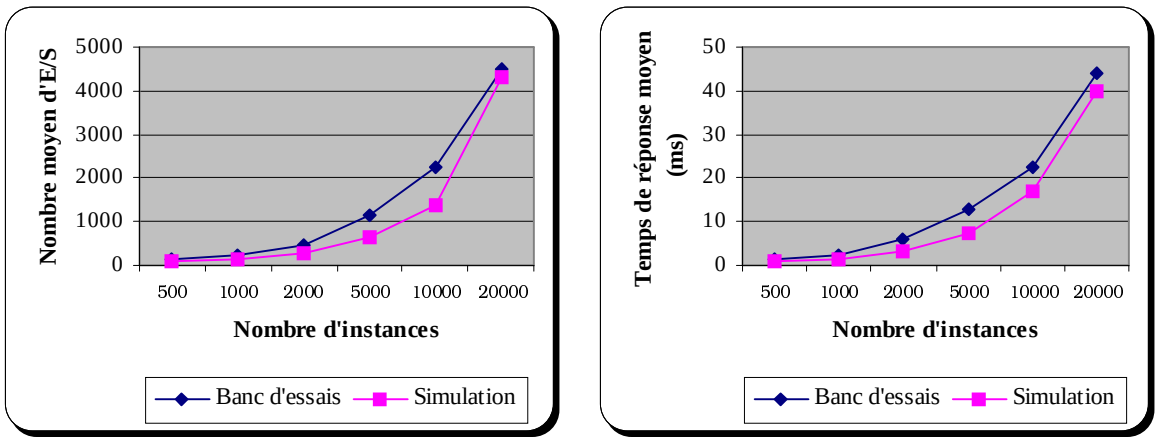

Figure 10 : Nombre d'entrées-sorties et temps de réponse moyens en fonction du nombre d'instances – Texas, 50 classes

5.4.2. Variation de la taille de la mémoire primaire

Comme Texas fait appel aux mécanismes de mémoire virtuelle du système d'exploitation, nous avons étudié les effets d'une réduction de la taille de la mémoire centrale disponible sous Linux. Les résultats obtenus en termes de nombre moyen d'entrées-sorties et de temps de réponse moyen sont présentés respectivement dans la Figure 11.

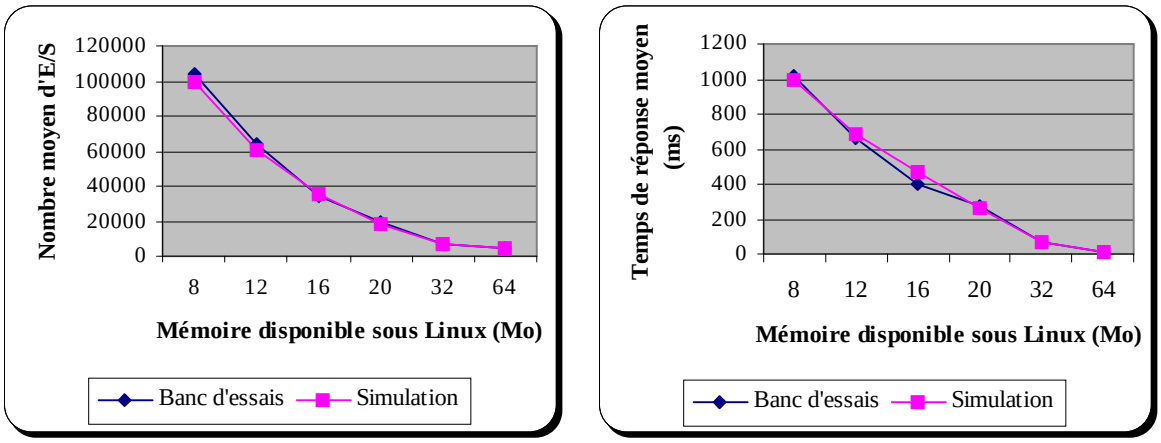

Figure 11 : Nombre d'entrées-sorties et temps de réponse moyens en fonction de la taille de la mémoire (Texas)

Elle montre que les performances de Texas se dégradent rapidement lorsque la taille de la mémoire centrale devient inférieure à celle de la base de données (21 Mo en moyenne). Cette dégradation importante des performances est due à la politique de chargement des objets de Texas, qui provoque la réservation en mémoire centrale de nombreuses pages avant qu'elles soient effectivement chargées. Cette politique génère un *swap* d'autant plus important que la taille de la mémoire centrale est réduite. Les résultats de simulation fournis par VOODB sont par ailleurs toujours conformes à la réalité.

5.5. Étude de sensibilité

En amont de ces expériences sur Texas et O_2, nous avons mené une étude de sensibilité pour déterminer l'impact réel d'une variation des différents paramètres de VOODB (à l'exception de ceux qui définissent la charge et qui dépendent du modèle de charge utilisé).

En effet, un modèle de simulation doit avoir une bonne sensibilité. Nous l'avons vérifié en faisant varier les paramètres qui définissent le *Sous-système des Entrées-Sorties* (Figure 12), ainsi que ceux qui concernent la configuration client-serveur du système (Figure 13). Les autres paramètres de VOODB ont été fixés à leur valeur par défaut.

La Figure 12 confirme l'hypothèse que le *Sous-système des Entrées-Sorties* participe largement à la sensibilité du système, mais la Figure 13 indique que les temps de transfert réseau sont également cruciaux. Leur influence est plus importante pour les architectures client-serveur qui sollicitent davantage le réseau, comme les architectures en serveur de pages.

Notre étude a également porté sur l'effet d'autres paramètres, comme la taille du cache disque et la politique de remplacement des pages dans le cache ou le temps nécessaire au contrôle de concurrence. Il est apparu que la sensibilité aux variations de ces paramètres était également bonne.

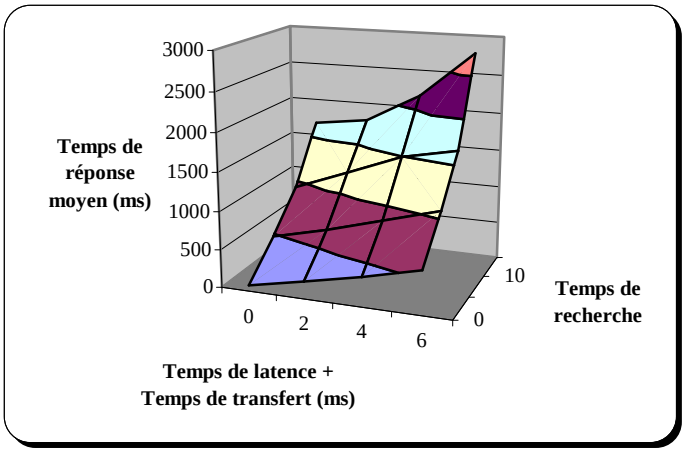

Figure 12 : Influence des performances du disque

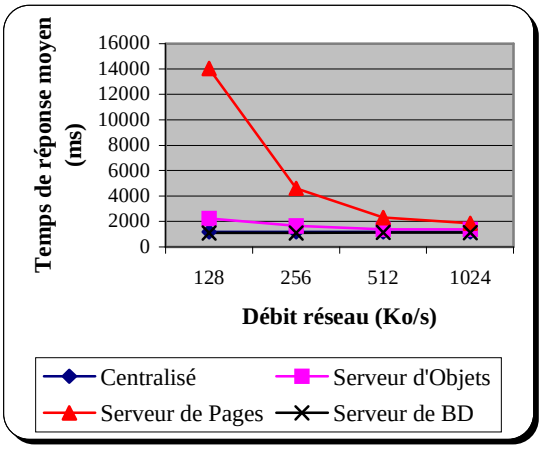

Figure 13 : Influence de la classe de système et du débit réseau

6. Conclusion

L'approche par simulation constitue une bonne alternative à l'expérimentation pour évaluer les performances des systèmes informatiques, comme l'ont montré de nombreux travaux menés depuis plus de vingt ans. Cependant, cette approche a été négligée au cours de ces dernières années en ce qui concerne les SGBDOO. Nous avons cherché à réhabiliter la simulation dans ce domaine, mais aussi à la mettre en œuvre dans une optique plus ambitieuse, en proposant un outil d'évaluation des performances générique et réutilisable. Cet outil est constitué du modèle de simulation VOODB et du banc d'essais OCB, qui est employé en tant que modèle de charge dans VOODB.

Nous avons montré dans la Section 5 que VOODB permet de retrouver les mesures de performance effectuées directement sur deux systèmes très différents : O_2 et Texas. De plus, VOODB peut aussi être employé pour évaluer l'efficacité de méthodes d'optimisation, comme l'ont montré d'autres résultats que nous avons obtenus lors de l'évaluation de l'impact de la méthode de regroupement DSTC sur les performances globales du gestionnaire d'objets persistants Texas [DARM99]. Ces diverses expériences illustrent la flexibilité et la généricité de VOODB. Nous pensons que VOODB, paramétré de façon adéquate, est un outil apte à effectuer des campagnes intensives d'évaluation pour différents types de systèmes.

La qualité d'un modèle de simulation dépend largement du processus de conception utilisé. La méthodologie de modélisation que nous avons mise en œuvre permet, de manière systématique, d'analyser un système, de spécifier des modèles fiables pour ce système et d'envisager une automatisation de leur traduction dans des langages de programmation ou de simulation. Elle autorise une bonne réutilisabilité des modèles ainsi obtenus.

Par ailleurs, pour tester le comportement du système étudié sous ses différents aspects, il faut disposer d'un modèle de charge paramétrable. L'intérêt du banc d'essais OCB en tant que modèle de charge réside en premier lieu dans son adaptabilité. Plusieurs sortes de bases d'objets très différentes peuvent être modélisées avec OCB, de même que différents types d'applications utilisant ces bases de données. Il s'agit là d'une fonctionnalité importante, dans la mesure où les applications bases de données orientée objet conduisent à une grande variété d'accès. D'autre part, les aspects stochastiques présents dans OCB permettent de prendre en compte des comportements rencontrés dans des situations réelles.

Il est important de signaler que le banc d'essais OCB peut également être utilisé dans une approche classique par expérimentation.

Les perspectives ouvertes par cette étude concernent principalement l'évaluation des performances de divers SGBDOO existants ou futurs, dans le but de comparer différents systèmes ou diverses solutions techniques au sein de ces systèmes (notamment des techniques d'optimisation comme le regroupement d'objets).

Le modèle de simulation VOODB est par ailleurs susceptible d'évoluer. Sa modularité permet une reconfiguration aisée par adjonction ou remplacement de composants. Par exemple, il pourrait être étendu par incorporation de composants susceptibles de prendre en compte différentes politiques de préchargement des objets, de contrôle de concurrence ou d'optimisation des requêtes. VOODB pourrait également proposer en standard des modèles de charge autres qu'OCB ou intégrer des modèles de dysfonctionnements pour tester la robustesse du système face aux aléas.

Références bibliographiques

[ANDE90] T.L. Anderson, A.J. Berre, M. Mallison, H.H. Porter III, B. Scheider, "The HyperModel Benchmark", *International Conference on Extending Database Technology (EDBT '90)*, Venice, Italy, March 1990, pp. 317-331

[ATKI90] M.P. Atkinson, D. Maier, "Perspectives on persistent object systems", *4th International Workshop on Persistent Object Systems* (Concluding remarks on Workshop), Martha's Vineyard, USA, September 1990, pp. 425-426

[BALC92] O. Balci, R.E. Nance, "The simulation model development environment: an overview", *1992 Winter Simulation Conference*, 1992, pp. 726-736

[BANK96] J. Banks, "Output Analysis Capabilities of Simulation Software", *Simulation*, Vol. 66, No. 1, January 1996, pp. 23-30

[BATE95] C. Bates, I. Jelly, I. Lalis, P. Menhart, "Simulating transaction processing in parallel database systems", *7th European Simulation Symposium (ESS '95)*, Erlanger-Nuremberg, Germany, October 1995, pp. 193-197

[BULL96] F. Bullat, M. Schneider, "Dynamic Clustering in Object Database Exploiting Effective Use of Relationships Between Objects", *ECOOP '96*, Linz, Austria, July 1996; *LNCS* Vol. 1098, pp. 344-365

[CARE93] M.J. Carey, D.J. Dewitt, J.F. Naughton, "The OO7 Benchmark", *ACM SIGMOD International Conference on Management of Data*, Washington DC, May 1993, pp. 12-21

[CATT91] R.G.G. Cattell, "An Engineering Database Benchmark", *The Benchmark Handbook for Database Transaction Processing Systems*, Morgan Kaufmann, 1991, pp. 247-281

[CHAN89] E.E. Chang, R.H. Katz, "Exploiting Inheritance and Structure Semantics for Effective Clustering and Buffering in an Object-Oriented DBMS", *ACM SIGMOD International Conference on Management of Data*, Portland, Oregon, June 1989, pp. 348-357

[CHEN91] J.R. Cheng, A.R. Hurson, "Effective clustering of complex objects in object-oriented databases", *ACM SIGMOD International Conference on Management of Data*, Denver, Colorado, May 1991, pp. 22-31

[CORE88] CORE International, *CORE Disk Performance Test Program Version 2.8 User Manual*, 1988

[DARM98] J. Darmont, B. Petit, M. Schneider, "OCB: A Generic Benchmark to Evaluate the Performances of Object-Oriented Database Systems", *6th International Conference on Extending Database Technology (EDBT '98)*, Valencia, Spain, March 1998; *LNCS* Vol. 1377, pp. 326-340

[DARM99] J. Darmont, *Étude des performances de méthodes de groupement dynamiques dans les bases de données orientées objet*, Thèse de Doctorat, Université Blaise Pascal – Clermont-Ferrand II, janvier 1999

[DEUX91] O. Deux et al., "The O_2 System", *Communications of the ACM*, Vol. 34, No. 10, October 1991, pp. 34-48

[GAY97] J.-Y. Gay, L. Gruenwald, "A Clustering Technique for Object Oriented Databases", *8th International Conference on Database and Expert Systems Applications (DEXA '97)*, Toulouse, France, September 1997; *LNCS* Vol. 1308 (Springer), pp. 81-90

[GEPP94] A. Geppert, S. Gatziu, K.R. Dittrich, *Performance evaluation of an active database management system: OO7 meets the BEAST*, Technical Report No. IFI-94-18, Computer Science Department, University of Zurich, Switzerland, November 1994

[HE93] M. He, A.R. Hurson, L.L. Miller, D. Sheth, "An Efficient Storage Protocol for Distributed Object-Oriented Databases", *IEEE Parallel and Distributed Processing*, 1993, pp. 606-610

[JUN97] W. Jun, L. Gruenwald, "Experiences with the OO7 benchmark for concurrency control technique performance evaluation", *OOPSLA '97 Workshop on Experiences Using Object Data Management in the Real-World* (Position paper), Atlanta, Georgia, October 1997

[KELL97] P. Kellert, N. Tchernev, C. Force, "Object-oriented methodology for FMS modelling and simulation", *International Journal Computer Integrated Manufacturing*, Vol. 10, No. 6, 1997, pp. 405-434

[NANC81] R.E. Nance, "Model representation in discrete event simulation: the conical methodology", *Technical Report* CS-81003-R, Department of Computer Science, Virginia Tech, Blacksburg, Va., 1981

[SCHR94] H. Schreiber, "JUSTITIA: a generic benchmark for the OODBMS selection", *4th International Conference on Data and Knowledge Systems in Manufacturing and Engineering*, Shatin, Hong Kong, May 1994, pp. 324-331

[SIMU95] Simulog, *QNAP2 Reference Manual*, 1995

[SARG79] R.G. Sargent, "Validation of simulation models", *1979 Winter Simulation Conference*, San Diego, 1979, pp. 497-503

[SING92] V. Singhal, S.V. Kakkad, P.R. Wilson, "Texas: An Efficient, Portable Persistent Store", *5th International Workshop on Persistent Object Systems*, San Miniato, Italy, 1992

[TIWA95] A. Tiwary, V.R. Narasayya, H.M. Levy, "Evaluation of OO7 as a system and an application benchmark", *OOPSLA '95 Workshop on Object Database Behavior, Benchmarks and Performance*, Austin, Texas, October 1995

[TSAN92] M.M. Tsangaris, J.F. Naughton, "On the Performance of Object Clustering Techniques", *ACM SIGMOD International Conference on Management of Data*, San Diego, California, June 1992, pp. 144-153