Université de Montréal

Évaluation, par simulation, des performances de serveurs de commerce électronique. Étude de cas : le serveur GNP.

Par

Slimane Bah

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M. Sc)
en informatique

Août, 2003

Université de Montréal

Faculté des études supérieures

Ce mémoire est intitulé :

Évaluation, par simulation, des performances de serveurs de commerce électronique.
Étude de cas : le serveur GNP.

présenté par :

Slimane Bah

a été évalué par un jury composé des personnes suivantes :

Houari A. Sahraoui
Président-rapporteur

Rachida Dssouli
Directrice de recherche

Ferhat Khendek
Membre du jury

Mémoire accepté le :
12 novembre 2003

RÉSUMÉ

Ce travail s'intéresse à l'étude et l'évaluation des performances d'une plate-forme de commerce électronique. En effet, il s'agit du serveur GNP, une plate-forme générique de négociation. L'étude s'inscrit dans le cadre d'un projet conjoint entre le CIRANO et Bell Télécoms à travers les laboratoires universitaires de Bell. Le projet GNP est initié pour donner une autre dimension au commerce interentreprises.

La mise en marché d'une telle plate forme nécessite une certaine vigilance car le domaine du e-commerce est un domaine très sensible aux imperfections et où la concurrence est accrue. Donc, dans le but de réussir sa commercialisation, des tests bien définis doivent être effectués. En particulier, une évaluation de performance s'impose. L'importance des tests de performance vient du fait qu'ils évaluent le système du point de vue utilisateur et non du point de vue fonctionnalités. L'utilisateur est un facteur sensible et déterminant dans les applications de commerce électronique.

Toutefois, avant d'effectuer de tels tests une compréhension de l'environnement, de ses caractéristiques et une modélisation de la charge de travail constituent la base de toute évaluation de performance. Nous avons étudié les spécificités de l'environnement commerce électronique et de son trafic puis nous avons mis en place un modèle de charge que nous avons appliqué au système étudié afin d'évaluer son temps de réponse. Nous avons remarqué en particulier l'influence du comportement des clients sur la plate-forme. Nous présentons dans ce travail la démarche suivie et l'analyse des résultats obtenus.

Mots Clés : évaluation de performance, commerce électronique, charge de travail, GNP, modèle de trafic, tests de performance, temps de réponse.

Abstract

The performance evaluation is an important aspect before or during the use of any system. In particular, when it's a question of an e-commerce platform, the challenge is great. Indeed, the e-business domain is very sensitive to the imperfections of the system and where competition is keen. Our work is interested in the study and the performance evaluation of a generic negotiation platform called GNP. The GNP project was initiated to give another dimension to the business between companies.

The marketing of this server requires a certain attentiveness and well defined tests should be made especially performance testing. The importance of such tests comes because they estimate the system from the user point of view and not from functionalities point of view. The user is a sensitive and determining factor in the e-commerce applications.

However, before making tests, an understanding of the environment, its characteristics and modelling the workload is essential task for any performance evaluation. We studied the particularities of the e-commerce environment and of its traffic then we set up a workload model which we applied to the system to estimate its response time. We noticed in particular the influence of the behaviour of the customers on the platform. We present in this work the followed methodology and the obtained results

Key words : performance evaluation, e-commerce, workload, GNP, traffic model, performance testing, response time.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABREVIATIONS ET DES SYMBOLES

- **API :** Application Programming Interface.
- **CPU :** Central Process Unit. Sert en général à désigner le processeur.
- **CSET :** Chip Secure Electronique Transaction.
- **DTD :** Document Type Definition. Document contenant les règles que doivent respecter les attributs et les éléments d'un document XML pour que celui-ci soit valide.
- **EDI :** Electronic Data Interchange.
- **EJB :** Entreprise JavaBeans.
- **GAMME :** Génération de Multiples Marchés Electroniques. Projet de recherche scientifique et expérimental sur les marchés électroniques. Projet du CIRANO.
- **GEE :** Generic Experimentation Engine. Plate forme orientée jeux et permettant l'étude du comportement d'un joueur devant différentes situations. Projet du CIRANO.
- **GNP :** Generic Negociation Plateform. Version élargie de GEE, c'est un serveur permettant l'utilisation de plusieurs modèles de négociations. Projet du CIRANO.
- **HTML :** HyperText Markup Language.
- **HTTP:** HyperText Transfer Protocol.
- **JMS :** Java Message Service.
- **JSP :** JavaServer Pages.

- **LAN :** Local Area Network. Désigne les réseaux locaux.
- **NTK :** Negociation ToolKit. Une API de GNP facilitant la manipulation de documents
- **RSA :** Algorithme de chiffrement à clé publique mis au point par Rivest, Shamir et Adleman.
- **SDK :** Software Development Kit.
- **SLA :** Service Level Agreement. Accord sur la qualité de service.
- **SNE :** Serveur de Négociation Electronique. Premier projet réalisé dans le cadre de TEM. Projet du CIRANO.
- **SSL :** Secure Sockets Layer.
- **TCP :** Transmission Control Protocol.
- **TEM :** Towards Electronic Marketplaces. Projet dont l'objectif est de construire et d'expérimenter un ensemble de services avancés articulés autour d'une place de marché ouverte. Projet du CIRANO.
- **URL :** Uniform Ressource Locator.
- **XML :** eXtensible Markup Language.

REMERCIEMENTS

A l'issue de ce travail je tiens à exprimer ma profonde reconnaissance à ma directrice de recherche **Dr. Rachida DSSOULI**, professeur à l'Université Concordia, de m'avoir orienté vers un sujet si intéressant et pour les efforts et soutien qu'elle n'a cessé de prodiguer. Ma profonde gratitude va aussi à **M. Robert Gérin-Lajoie**, directeur exécutif groupe commerce électronique au CIRANO, pour m'avoir ouvert les portes de cette institution, pour son aide et son suivi.

Mes remerciements vont aussi à M. Charles Helou, pour son aide et sa collaboration afin de réussir ce travail. Je remercie également M. Mohamed Vall Ould Mohamed Salem pour ses conseils et le temps qu'il a bien voulu m'accorder pour mieux cerner mon sujet.

Merci aux membres du jury pour le temps consacré à l'évaluation de mon travail.

Ce travail n'aurait sûrement pas vu le jour sans la contribution de l'Université de Montréal, les Laboratoires Universitaires de BELL (LUB) et du Centre Interuniversitaire de Recherche en Analyse des Organisations (CIRANO). A tous je dis merci beaucoup.

INTRODUCTION

Les technologies de l'information et les réseaux sont les clés de la société électronique moderne. Ces éléments ont changé la façon dont les personnes et les entreprises interagissent et coopèrent. Les réseaux en particuliers, ont connu un grand essor depuis leur apparition. Ils continuent à être le moyen le plus utilisé pour l'échange de tout type de données. De plus ils offrent différents services et applications. L'apparition d'applications gourmande en ressources, entre autre, la Vidéo sur demande, le télé-enseignement, la visioconférence et surtout le commerce électronique affectent beaucoup certains paramètres directement perceptibles par l'utilisateur final.

Les réseaux, Internet en particulier, ont contribué à la vulgarisation et à la diffusion du commerce électronique à grande échelle. Celui-ci existe depuis quelques années déjà, mais sa croissance n'a jamais semblé aussi assurée qu'aujourd'hui. Il parait que de plus en plus d'entreprises veulent profiter des avantages et des bénéfices futurs annoncés. Ils veulent tous participer à cette ruée vers l'or virtuelle.

La nouvelle économie prend plusieurs formes. Les enchères virtuelles sont une variante très importante du commerce électronique. En effet, plusieurs gouvernements choisissent ce mode pour procéder à l'adjudication de contrats de tout type. Cette expansion a donné lieu à des endroits virtuels de rencontres entre vendeurs et acheteurs. Ces lieux sont communément appelés places de marché virtuelles. Celles-ci sont prises en charge entièrement par des serveurs qui assurent les fonctionnalités de contrôle et de suivi des négociations. Donc, une infrastructure logiciels et matérielles adéquate est nécessaire.

Devant ce développement continuel du commerce électronique, le problème de la performance se pose. Ainsi le temps de chargement d'un document ou le temps mis par une transaction commerciale peut causer la non satisfaction de l'utilisateur et avoir des

conséquences désastreuses sur le chiffre d'affaire ou sur l'image de marque de l'entreprise.

En effet, il est très important d'assurer une bonne qualité de service aux utilisateurs dont le nombre augmente de jour en jour et deviennent de pus en plus exigent. Il suffit de cliquer sur un lien pour que le client se connecte sur un serveur plus performant offrant les mêmes services. Donc il est indispensable de doubler d'efforts pour tester la performance des serveurs. Les serveurs de commerce électronique sont loin de faire l'exception. En fait, ils sont amenés à supporter des charges de travail énorme et chaotique. D'ou l'incessant besoin de tester à l'avance leurs performances afin de pouvoir détecter tout dysfonctionnement et prévoir leurs comportements sous pression.

Dans le cadre du projet TEM (Towards Electronic Marketplaces), les laboratoires de BELL en partenariat avec le Centre Interuniversitaire de Recherche et d'ANalyse des Organisations (CIRANO) ont développé une plate forme générique de négociations GNP (Generic Negociation Platform). Elle est conçue de telle sorte à prendre en charge plusieurs types de négociations, en particuliers les enchères ouvertes. Les tests de performance de cette plate forme revêtent une importance particulière. Le Département d'Informatique et de Recherche Opérationnelle (DIRO) de l'Université de Montréal a contribué dans ces tests.

L'évaluation des performances d'un système passe tout d'abord par la caractérisation de sa charge de travail. En effet, il est primordial de comprendre et d'étudier le trafic traversant le système. Chaque plate-forme a ses particularités qui dépendent du type de requête qu'elle reçoit, l'intensité du trafic, la nature du trafic et bien sur de l'utilisateur. La charge d'une plate-forme de commerce électronique est étroitement liée au comportement et aux interactions des clients avec le système. Surtout lorsqu'il s'agit de négociations.

Notre apport pour le projet TEM se résume, premièrement, dans l'étude du trafic e-commerce. Nous avons élaboré un modèle de charge, pour GNP, basé sur des modèles stochastiques analytiques. Ce modèle, que nous avons pris le soin de concevoir, prend en considération les particularités d'un trafic Web et donc de commerce électronique. Il nous a servi, dans un deuxième temps, pour simuler les utilisateurs en générant des

requêtes vers le serveur GNP. Nous nous sommes intéressé spécifiquement à l'attitude des utilisateurs. Pour ce faire nous avons étudié la notion de « think time ». d'autre part afin d'évaluer, pratiquement, les performances du serveur GNP, nous avons mis en œuvre des scénarii de tests selon des objectifs bien précis, récupéré et analysé les résultats obtenus. Ce qui nous a permis d'analyser le lien entre le temps de réponse et le comportement des clients de la plate-forme. Signalons d'emblée que la version de GNP que nous avons testée est la version 1 qui a laissé place à une nouvelle version améliorée de GNP et plus performante.

Nous souhaitons que ce rapport traduise de façon fidèle le travail effectué. Pour ce faire, nous l'avons organisé en cinq chapitres. Dans le premier chapitre nous situons le projet GNP dans son contexte général tout en soulevant la problématique de performance. Le deuxième chapitre est consacré au processus d'évaluation des performances des systèmes, aux concepts de base nécessaires pour comprendre et réussir les tests de performances. Il présente les différentes techniques de tests. Au troisième chapitre nous avons présenté la notion de charge de travail et sa caractérisation. Nous avons aussi mis l'accent sur les spécificités du trafic Web et commerce électronique. L'introduction du commerce électronique, de la performance des serveurs de e-commerce ainsi que l'environnement GNP a fait l'objet du quatrième chapitre. Le dernier chapitre mettra en valeur la charge de test utilisée, la démarche suivie pour la mise au point des scénarii de tests. Il présente aussi les résultats obtenus et analysés.

Chapitre 1

CONTEXTE GÉNÉRAL DU PROJET

L'étude que nous avons réalisée fait partie d'un projet global initié par les Laboratoires Universitaires de Bell (LUB) et le Centre Interuniversitaire de Recherche en Analyse des Organisations (CIRANO). Dans ce chapitre nous nous consacrons à présenter le dit projet et son évolution.

Dans la première section nous donnerons un aperçu sur le sujet. Une description des travaux de recherche déjà réalisés sera faite dans la deuxième section. Ensuite, nous exposerons la problématique que nous nous proposons d'étudier.

1.1. Présentation du sujet

Le commerce électronique a connu une grande expansion grâce à plusieurs facteurs facilitant son utilisation. Cette évolution rapide a engendré des conséquences économiques intéressantes en terme de bénéfice. Ce qui pousse concepteurs, développeurs et fabricants d'améliorer leurs services et ainsi attirer plus de clients. Pour ce faire, les technologies de l'information restent la clé pour atteindre un tel but.

Les technologies de l'Internet ont et auront un impact important sur la structure de nos économies modernes. Ces technologies permettent non seulement d'informatiser les processus d'affaires existants, mais aussi de transformer les organisations et les échanges pour les rendre plus efficaces. Le CIRANO (Centre Interuniversitaire de Recherche et d'Analyse des Organisations) s'est donné le mandat d'explorer ce second volet. Depuis plus de cinq années, une équipe de recherche du CIRANO, en

collaboration avec les Laboratoires Universitaires de Bell, s'est intéressée au design et la mise en place de marchés et de mécanismes d'enchères sur l'Inforoute.

Dans cette perspective, un premier projet a vu le jour sous le nom de GAMME. Celui-ci a permis la création d'un Serveur de Négociation Electronique (SNE) utilisé dans le cadre d'un projet-pilote pour la vente de copeaux de bois. Le projet TEM (Towards Electronic Marketplaces), en cours, prend la relève. Son objectif est de construire et d'expérimenter un ensemble de services avancés articulés autour d'une place de marché ouverte et susceptibles d'être greffés au SNE [21].

Le projet TEM passe par deux phases essentielles. La première phase a donné lieu à un engin d'expérimentation générique GEE (Generic Experimentation Engine). Ce dernier a connu une grande amélioration et évolution dans la deuxième phase du projet pour donner naissance à GNP (Generic Negociation Platform) [21] et [1].

GEE est basé sur une plate forme spécialement conçue pour expérimenter différents types de jeux et d'étudier le comportement d'un joueur. En outre, il est considéré comme un engin flexible permettant l'implantation de plusieurs types de jeux. En se basant sur ces caractéristiques, la plate forme GNP a été mise au point. C'est une version améliorée de GEE permettant l'utilisation de plusieurs modèles de négociations et selon divers types de marché [21].

Le principal défi relevé par le projet TEM est de reformuler la manière de prendre une décision et de faire fonctionner un marché. Le projet TEM a donc pour but de fournir des plates formes capables de supporter une large gamme de règles et d'algorithmes de négociations [1].

Toutefois, garantir une qualité de service acceptable pour ces plates formes, passe tout d'abord par assurer une performance satisfaisante. Nous nous intéresserons plus exactement, dans notre étude, à l'évaluation des performances des serveurs de commerce électronique et en particulier GNP.

1.2. Etat de l'art

La performance des applications et serveurs Web a fait l'objet d'une multitude de travaux de recherches. La majorité du travail effectué s'est consacré à la caractérisation

[1] http://www.cirano.qc.ca/eree

de la charge de travail « *Workload* » Web et au développement d'outils adéquats pour générer un trafic représentatif.

Diverses techniques et méthodes sont utilisées et des outils de génération de trafic comme SPECweb99, SURGE et WAGON ont vu le jour. Nous décrirons sommairement les trois benchmarks les plus cité en littérature [15], [17].

- **SPECweb99**[1] **:** SPECweb99 est un benchmark utilisé pour l'évaluation des performances des serveurs HTTP. Ce benchmark a été développé par *Standard Performance Evaluation Corporation* (SPEC). Il est conçu pour mesurer la capacité d'un système à agir tel qu'un serveur Web manipulant aussi bien des pages Web statiques que dynamiques. SPECweb99 utilise une charge de travail qui est plus appropriée aux modèles d'utilisation Web actuels à savoir un contenu dynamique et des connexions persistantes.
- **SURGE :** *Scalable URL Reference Generator* est un outil de génération de charge de travail dont le but est d'imiter fidèlement un flux de requêtes HTTP. Il génère des références correspondant aux mesures empiriques de différents paramètres. Entre autre, la distribution de la taille des fichiers du serveur, la distribution de la taille de la requête et les périodes d'inactivités des utilisateurs. Afin de générer un tel trafic, SURGE est basé sur une approche analytique intégrant plusieurs méthodes.
- **WAGON :** *Web trAffic GeneratOr beNchmark* est aussi un logiciel de benchmark de serveur implémentant un nouveau modèle de trafic Web. La particularité de ce modèle est qu'il considère le comportement typique des utilisateurs lors d'une navigation hypertexte sur un serveur Web. En outre, il est formulé comme un processus stochastique ponctuel qui décrit quand un utilisateur arrive sur le serveur et comment il navigue sur celui-ci. WAGON intègre ce modèle agissant au niveau session et donc il a une correspondance directe avec le comportement hypertexte de la navigation des utilisateurs.

Les benchmarks conçus sont utilisés pour évaluer différents aspects des serveurs Web en se basant sur des modèles, bien étudiés, de trafic. Le but essentiel des recherches faites

[1] http://www.specbench.org/web99/

est de générer un trafic se rapprochant le plus possible de la charge de travail réelle. Ainsi cela permettra de simuler le « *Workload* » et capter les paramètres de performance.

Toutefois, la grande majorité des recherches se focalise sur les serveurs Web. Les serveurs de commerce électronique sont rarement visés et donc leurs caractéristiques ne sont pas prises en compte lors de la génération du trafic.

En particulier le serveur que nous nous proposons d'étudier et qui est toujours en phase de développement n'a connu que peu de recherche. Notre tâche sera d'élaborer un modèle de charge de travail pour le serveur et réaliser des tests dynamiques se basant sur des modèles analytiques.

1.3. Problématique

La technologie du commerce électronique est influencée, en général, par deux facteurs importants. D'une part, les applications du e-commerce ont connu et continuent de connaître une grande évolution technologique. Ce qui implique que le temps de leur mise en marché devient un facteur déterminant vu la concurrence existante. D'autre part, le nombre de personne qui procède au magasinage par Internet est en croissance vertigineuse. Par conséquent, la consommation et la demande de la bande passante ont exponentiellement augmenté poussant les réseaux, les noeuds et les serveurs aux limites de leur capacité.

Cette réalité pousse constructeurs et développeurs à bien tester leurs produits avant leur commercialisation. Les tests de performance trouvent leur importance dans le fait qu'ils s'attaquent à l'aspect efficacité du système. La performance est une notion complexe qui dépend de plusieurs paramètres. En effet, tout utilisateur d'un réseau d'ordinateurs est concerné par sa performance. Ainsi, différentes perceptions peuvent être analysées. Du point de vue utilisateur final la performance se mesure, entre autre, par le temps d'exécution des programmes, le temps de réponse et le pourcentage de connexions refusées. Pour un administrateur, la performance est exprimée, essentiellement, en terme de taux de service (nombre de requêtes que le système peut servir par unité de temps, trames perdues…etc.). Donc, avant de se lancer dans la conception des piles de tests, il faut définir d'une manière claire les objectifs escomptés et bien étudier le sujet à tester.

Les tests de performance ont pour objectif de mesurer et d'étudier le comportement du système sous différentes charges. Plusieurs paramètres peuvent être évalués. Citons à titre d'exemple le temps de réponse, le débit, le degré de disponibilité, l'utilisation des ressources, les cycles CPU, l'espace disque, les opérations d'accès au disque et l'utilisation de la mémoire.

L'évaluation de la performance passe en premier lieu par la bonne définition de la charge de travail. Diverses techniques peuvent être appliquées selon la nature du trafic, la disponibilité des données et la disponibilité du système. Le défi donc, réside dans le fait de bien modéliser le trafic et puis utiliser ce modèle afin de réaliser les tests de performance désirés.

1.4. Conclusion

Nous avons donné un aperçu sur le projet global TEM qui est caractérisé par les deux plates-formes génériques GEE et GNP. Nous avons aussi mis en évidence le problème de performance. Celui-ci est particulièrement dangereux pour les applications de types commerce électronique. En effet, dans un domaine où la concurrence est de taille, la qualité de service devient un enjeu. Les tests de performance apportent un appui considérable pour la localisation des défectuosités et la prévision du comportement des serveurs.

C'est dans ce cadre là que notre travail s'inscrit. Nous tenterons de définir la nature du trafic destiné à un serveur de commerce électronique. En se basant sur les travaux de recherche déjà réalisés, nous élaboreront un benchmark et nous l'utiliserons afin de mesurer le temps de réponse du serveur GNP.

Dans le reste du présent rapport, notre intérêt portera sur la caractérisation du « *Workload* » et son application pour réaliser des tests de performance permettant d'apprécier le temps de réponse. Mais avant d'expliciter l'approche suivie, il est nécessaire de présenter quelques notions importantes. D'emblée il semble important de bien comprendre les trois notions fondamentales pour ce projet à savoir les tests de performance, la charge de trafic et le commerce électronique. Nous allons tout d'abord présenter les principes d'une évaluation de performance ainsi que les différentes et techniques suivies.

Chapitre 2

EVALUATION DE LA PERFORMANCE DES SYSTÈMES

L'évaluation des performances d'un système est un processus compliqué. Ce processus implique multiples analyses et décisions. Par conséquence et contrairement à ce qu'on peut avoir comme première impression, l'évaluation de la performance d'un système ne peut pas être immédiate. En effet, chaque étude demande une bonne connaissance du système à tester et un choix soigneux des méthodes et des techniques à utiliser. L'art des tests de performance réside dans la façon de déterminer le vrai problème et d'appliquer les bonnes méthodes tout en respectant un délai bien précis. Ce délai devient de plus en plus court et de plus en plus critique dés qu'il s'agit de systèmes de commerce électronique. Donc pour une bonne estimation des performances, différents modèles sont à mettre en œuvre.

Dans ce chapitre nous allons tout d'abord donner un aperçu global sur l'évaluation des performances. Nous enchaînerons ensuite par montrer l'importance d'une telle évaluation. Dans une troisième section, nous mettrons l'accent sur l'approche générale suivie pour mettre cette activité en pratique. Le dernier point de ce chapitre sera la description des trois plus importantes techniques utilisées pour l'évaluation des performances.

2.1. Préliminaires

La performance est une notion qui intéresse tous les acteurs dans un système informatique. En effet, le but de tout usager, administrateur ou concepteur de système est d'obtenir ou fournir une performance élevée avec un moindre coût. Elle est requise à toutes les étapes du cycle de vie d'un système informatique. Elle permet de comparer et faire un choix entre deux ou plusieurs conceptions ou systèmes. Par ailleurs, évaluer la performance d'un système existant donne une idée pertinente sur son degré d'efficacité et sur les améliorations éventuelles. En particulier il est important dans une étude de performance de faire le bon choix des mesures de performance, de l'environnement de mesure et de la technique. Pour atteindre cet objectif, une bonne compréhension de la performance est nécessaire.

Le problème fondamental d'une analyse de performance d'un système informatique est de définir la performance ainsi que les critères de son évaluation. En fait, il est à signaler que la performance est une caractéristique qualitative et fortement subjective. En effet, elle est étroitement liée aux besoins des personnes concernées par le système. Ainsi, la perception de la performance change des concepteurs aux utilisateurs.

La performance peut être définit comme étant l'efficacité avec laquelle les ressources du système informatique sont utilisées pour atteindre les objectifs du système [7]. Une deuxième définition vient compléter la première : la performance est la précision avec laquelle un système coïncide avec les espérances des personnes impliquées dans le système [6].

D'emblée il faut faire la différence entre l'évaluation d'un système et l'évaluation de sa performance. Cette dernière n'est qu'un aspect sous lequel un système peut être étudié et évalué. Toutefois c'est un aspect déterminant et très important pour la survie de tout système informatique. Effectivement, l'évolution rapide des technologies de l'information aujourd'hui et la courses vers la relève des défis les plus compliqués entre industriels et chercheurs ouvre les portes grandes ouvertes à la concurrence. Une réussite ne peut donc se prononcer que grâce à une qualité de service et une grande performance du système.

2.2. Motivation

Le Web a connu un grand essor depuis son apparition. Il continue à être le service le plus utilisé par les internautes dont le nombre croît à une vitesse vertigineuse. Cette croissance est accompagnée par celles des sites Web qui se multiplient de jour en jour. Devant cette demande intense des services offerts par Internet, le problème de la performance se pose. Ainsi le temps de chargement d'un document ou le temps mis par une transaction commerciale peut avoir des conséquences désastreuses sur le chiffre d'affaire et/ou sur l'image de marque de l'entreprise. En outre, la concurrence accrue dans ce domaine et la complexité des exigences des clients surtout en terme de qualité de service et de performance, ne laissent pas le choix aux différents acteurs.

Par ailleurs, les enjeux d'Internet nécessitent des décisions pertinentes. En effet, il a été rapporté que les sites de commerce électronique ont généré 132 billions de dollars en 2000. Le double de ce qui a été annoncé en 1998 [1]. Bien que la puissance des serveurs hébergeant les sites de commerce électronique a augmenté, ces sites étaient incapable d'améliorer le niveau de service offert aux internautes. En effet, un rapport cite que les sites de e-commerce ont perdu environ 420 millions de dollars des revenues en 1999. Cette perte est due à la lenteur du traitement des transactions [2]. Ainsi il est très souhaitable et nécessaire de se focaliser sur la performance des serveurs utilisés dans l'environnement du commerce électronique. Dans le but de garantir une certaine performance du système, une étude du dit système doit être minutieusement menée.

2.3. L'approche d'une évaluation des performances

La performance est un critère clé pour la conception, le développement et l'utilisation d'un système informatique. En réalité, le but des ingénieurs systèmes, des analystes et des utilisateurs est d'avoir un haut niveau de performance pour un coût donné.

2.3.1 Concept de base

Nombreuses sont les organisations qui dépensent beaucoup d'argent pour installer des architectures système, pour la maintenance et pour la mise à jour de leur environnement. Cependant dans la majorité des cas, la performance ou la capacité globale du système

[1] http://www.activmediaresearch.com/real_numbers_2000.html
[2] http://www.zdnet.com/ecommerce/stories/main/0,10475,2636088,00.html

n'est pas connue. De surcroît, la performance adéquate pour leur besoin est mal comprise. Pour définir la performance adéquate qui constitue l'objectif à atteindre, nous présentons la figure suivante [4] :

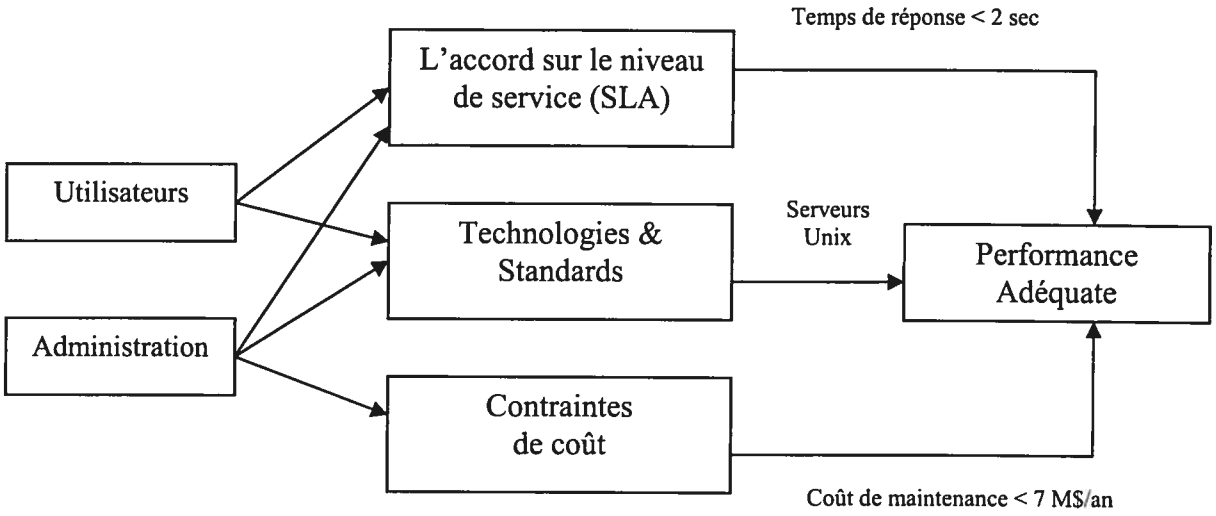

Figure 2.1 : Définition de la performance adéquate

Le schéma montre que la performance adéquate est le point de rencontre de trois éléments.

- *L'accord sur le niveau de service (Service Level Agreement SLA)* : Il est spécifique pour chaque organisation et est déterminé par les utilisateurs et l'administrateur. Il indique les contraintes sur les paramètres de performance pour garantir un certain niveau de satisfaction.
- *Technologies & Standards* : Atteindre le niveau de service souhaité (SLA) peut se faire grâce à différents moyens et technologies. Le choix des standards (réseaux, logiciels ou autres) peut influencer sur la qualité du service obtenue.
- *Les contraintes de coût* : Définir la performance adéquate pour respecter un SLA peut s'avérer facile si le problème des ressources financières ne se pose pas. En fait, les contraintes budgétaires limite l'espace de solutions.

Nous pouvons, ainsi, dire qu'un système possède une performance adéquate si l'accord sur le niveau de service (SLA) est, à tout moment, atteint pour des technologies et

standards particuliers tout en respectant les contraintes budgétaires. Le concept ainsi défini nous pouvons présenter l'approche systématique pour une étude de performance.

2.3.2 Les étapes d'évaluation des performances

Une bonne évaluation de performance ne peut être produite automatiquement. Chaque évaluation requiert une connaissance approfondie du système étudié et un choix minutieux de la méthodologie, du modèle du trafic et des outils.

Effectivement, la plupart des problèmes de performance sont uniques. Les métriques, la charge de travail, et les techniques utilisées pour un cas ne peuvent être repris pour un autre problème. Toutefois, des étapes communes, à tous les processus d'évaluation de performance, ont été élaborées [4] et [19]. Les dites étapes sont énumérées ci-dessous :

- *Spécifier les objectifs et définir le système* : la toute première étape dans n'importe quel processus d'évaluation de performance est de délimiter le système en définissant ses frontières et ensuite, déterminer les objectifs de l'étude. Cette activité revêt une importance particulière puisqu'elle affecte le choix des métriques et du modèle de la charge de travail.
- *Lister les services et les sorties* : chaque système offre un certain nombre de services. Un réseau permet par exemple l'envoi de paquets vers une destination précise, un serveur de commerce électronique permet, entre autre, de faire des transactions commerciales. En plus, pour chaque service offert il existe un ensemble de sorties possibles. Celles-ci peuvent être désirées ou non. L'inventaire des services et des résultats éventuels, constitue la seconde étape influente sur les métriques et la charge de travail.
- *Choisir les métriques* : Dans cette étape s'effectue le choix des critères de performances. Ces critères sont appelés métriques et sont divers et multiples. Ainsi, il est essentiel de choisir un sous ensemble représentatif. Les métriques sont généralement liées à la vitesse (débit, délai), précision (taux d'erreur) et la disponibilité du service. Une façon pour choisir les bonnes métriques est de lister les services offerts par le système et leur sorties. Celles-ci sont classées en trois catégories. Le schéma de la figure 2.2 montre la relation entre les services, les sorties et les métriques

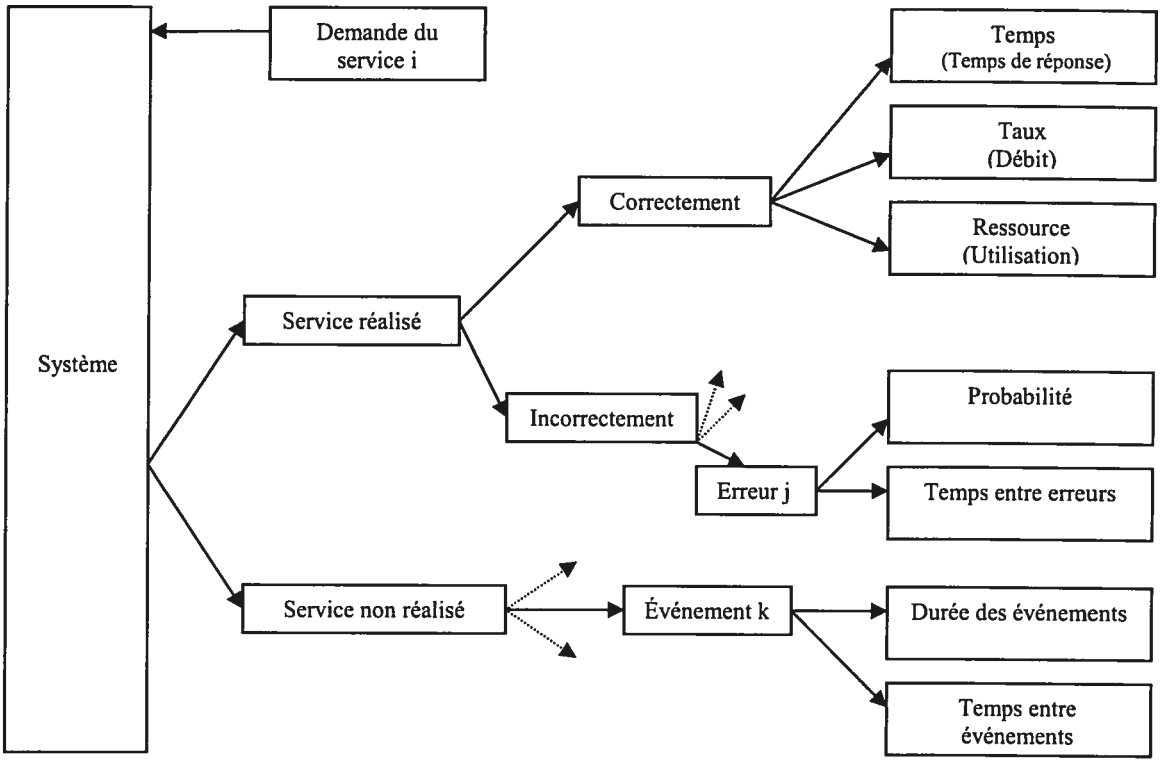

Figure 2.2 : La relation entre le choix des métriques, les services et les sorties

Pour une meilleure appréciation des métriques nous donnons les définitions suivantes :

- ✓ **Le temps de réponse** est défini comme étant l'intervalle de temps entre le lancement d'une requête et la réception d'une réponse par le système.
- ✓ **Le débit** est le taux de requêtes pouvant être servies par le système par unité de temps. Pour les traitements séquentiels le débit est mesuré par le nombre de travaux réalisés par seconde. Dans le cas des systèmes interactifs, le débit est mesuré par le nombre de requêtes par seconde. Pour les processeurs, le débit est exprimé par le nombre de millions d'instructions par seconde et concernant les réseaux il s'exprime en nombre de bits transmis par seconde.

- ✓ **L'utilisation** d'une ressource désigne la fraction de temps que prend une ressource pour servir les requêtes. Elle est donnée par le rapport entre le temps d'occupation de la ressource et la durée du test.
- ✓ **Disponibilité d'un système** est le temps durant lequel un système est disponible pour servir les requêtes.
- ✓ **La fiabilité** est donnée par la probabilité qu'une erreur se produise.

Les métriques ainsi comprises, définies et choisies la prochaine étape dans le processus d'évaluation de la performance peut être atteinte.

- *Lister les paramètres* : Dans le but de bien cerner le problème, une liste de paramètres qui affectent la performance doit être établie. Deux catégorie sont identifiées : les paramètres système et les paramètres liés au trafic. Les premiers sont, en général, peu variables d'une installation à l'autre. Ils concernent les paramètres matériels et logiciels. Nous pouvons citer comme exemple la vitesse du processeur, le débit du réseau et le système d'exploitation. Quant aux paramètres de la charge de travail ils sont caractéristiques des exigences des utilisateurs et donc variables. Nous donnons l'exemple du temps entre deux requêtes successives et le temps de service.
- *Choisir les facteurs à étudier* : Comme nous l'avons annoncé dans l'étape précédente, certains paramètres varient et d'autres non. Les paramètres qui changent durant l'évaluation sont appelés *facteurs* et leurs valeurs sont dites *niveaux*. Nous nous intéresserons à modéliser les facteurs et fixer tout les autres paramètres. En effet, les facteurs ont la caractéristique d'avoir un grand impact sur la performance du système. Durant le choix des facteurs, il est important de prendre en considérations les contraintes technologiques, temporelles et autres. Ainsi, les chances de trouver des solutions raisonnables et implantables, seront grandes.
- *Choisir la technique d'évaluation* : Une importance particulière est accordée à cette étape. Elle influence tout le processus d'évaluation. Il existe trois techniques d'évaluation : la mesure du système réel, la simulation et les

méthodes analytiques. Nous les aborderons en détails dans la prochaine section. Le choix de la bonne technique dépend du temps et des ressources disponibles ainsi que du niveau de précision souhaité.

- *Déterminer la charge de travail* : la charge de travail ou « *Workload* » sera étudiée dans le prochain chapitre vu son intérêt. Il consiste en l'ensemble des services demandés au système et peut être exprimé de différentes façons. En effet, le modèle de la charge de travail est étroitement lié à la technique d'évaluation choisie. Cependant, quelque soit la manière de l'exprimer, il doit représenter l'utilisation réelle du système. Autrement, les résultats de l'évaluation de performance seront insensés. D'où le poids de la caractérisation de la charge de travail dans un tel processus.
- *Concevoir les expériences* : une fois à ce stade, nous avons besoin de décider d'une séquence d'expériences offrant un maximum d'information avec un minimum d'effort. En particulier deux phases sont recommandées. Dans la première phase le nombre de facteurs serait élevé avec un nombre réduit de niveaux. Le but étant de déterminer l'effet des divers facteurs. Une seconde phase serait de réduire le nombre de facteurs et augmenter le nombre de niveaux pour les facteurs ayant un impacte significatif.
- *Analyser et interpréter les données* : les outils de mesure collectent d'immenses informations. Pour que ces informations soient exploitables, elles doivent être analysées et transformées en une information synthétique et significative. Ensuite, les données doivent être interprétées en tenant compte de leur variabilité puisque les résultats sont des quantités aléatoires.
- *Présenter les résultats* : La dernière étape dans le processus d'évaluation est la présentation des résultats analysés et interprétés. Afin de communiquer ces résultats et pouvoir en tirer profits, il est essentiel de les présenter dans une manière compréhensible, simple et globale.

Ainsi nous avons présenté les étapes constituant la démarche systématique pour évaluer la performance d'un système informatique. D'une manière plus compacte le schéma de la figure 2.3 résume le processus de mesure des performances :

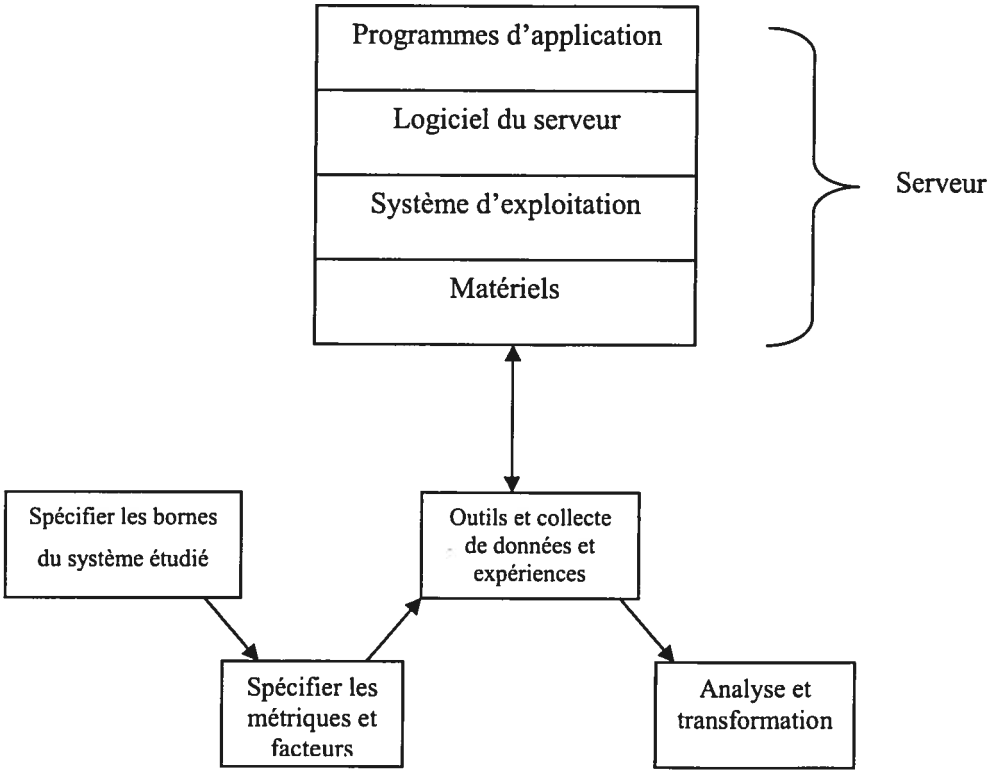

Figure 2.3 : Processus de mesure de la performance

L'approche étant bien définie, nous revenons sur deux étapes importantes à savoir le choix des techniques d'évaluation et la caractérisation de la charge de travail. Dans la section suivante nous présentons les différentes techniques.

2.4. Les techniques d'évaluation des performances

Etudier les performances d'un système peut s'effectuer sous différents angles. En fait, il existe trois techniques pour évaluer les performances à savoir, la mesure du système réel, la simulation et le modèle analytique. Le choix d'une telle ou telle technique est conduit par plusieurs paramètres. Le schéma ci-dessous présente les trois manières d'évaluation.

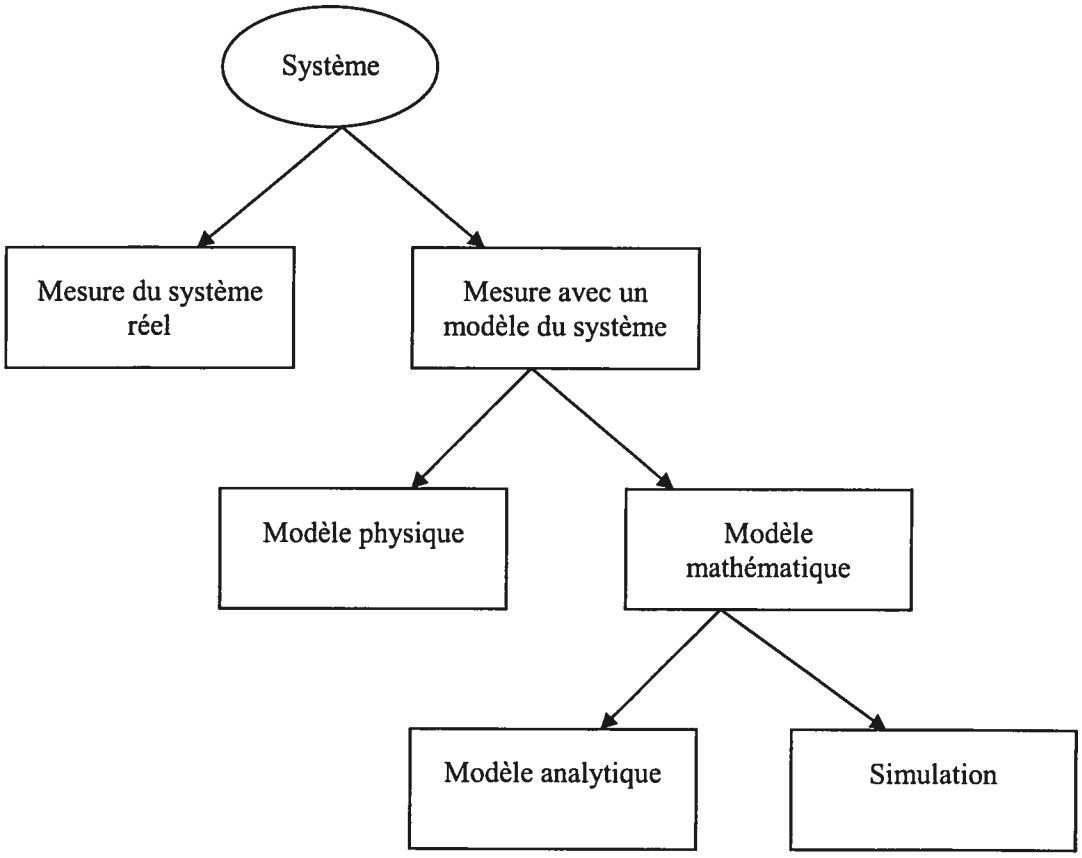

Figure 2.4 : Les différentes manières d'étudier un système

Le choix de la méthode peut s'avérer délicat. Il existe deux grandes catégories de techniques : les techniques empiriques basées sur la mesure du système réel et les techniques de modélisation. Ceux-ci regroupent deux techniques la simulation et les techniques analytiques. Nous expliciterons les trois techniques pour une meilleure compréhension de leur rôle [1], [19] et [1].

2.4.1 La mesure du système réel

Cette technique est dite empirique car elle n'utilise pas de modèle. Cependant, elle repose sur l'observation directe du système étudié ou d'un système similaire. Par conséquent, elle ne peut être utilisée que lorsque le système existe déjà ou son prototype. Lors de la planification d'une mesure, il est primordial de tenir compte de certains

[1] http://www.frontrunnercpc.com/info/infomain-fs.htm

aspects tels que l'objectif de la mesure, le choix de la charge de travail et son implémentation, les données à collecter, les outils utilisés pour cette fin et comment valider les résultats.

A première vue, la mesure et l'analyse des performances basées sur un système réel semblent attrayantes. Toutefois, elles cachent plusieurs limitations. En effet, la charge de travail appliquée à un système déployé varie de jour en jour sinon de minute en minute, ainsi il est quasi impossible de réutiliser une certaine charge de travail pour tester une autre configuration du système. Une solution serait d'utiliser des charges de travail synthétiques qui ont la caractéristique d'être répétables. De surcroît, pour mener à bien les mesures il faut disposer d'outils adéquats.

Il existe en général deux façons de mesurer un système informatique à savoir la mesure par détection d'événements et la mesure par échantillonnage. La manière de procéder aux mesures dépend largement du type de l'analyse désirée et du niveau auquel l'analyse est effectuée.

- *La mesure par détection d'événements* : par événement on désigne un changement dans l'état du système. Donc, une façon de mesurer une activité du système est de capter tous les événements relatifs à cette activité. Les données seront collectées et enregistrées par un moniteur. Le principe de cette méthode est d'insérer des bouts de code à des emplacements bien précis du système d'exploitation. Lorsqu'un événement se produit, le bout de code se charge de le transmettre au moniteur ainsi que toutes les informations utiles. Cette technique a l'avantage de fournir plus d'information que les autres. Toutefois, elle ne peut être utilisée sans inclure ses outils dans le système d'exploitation.

- *La mesure par échantillonnage* : cette technique est souvent préférée à la précédente. Elle consiste à interrompre le système à intervalle régulier afin de détecter l'état de certains de ses composants. Si le nombre d'échantillons est assez grand, ce type de mesure est très valable. En outre, elle a l'avantage de fournir moins de données que la mesure par événements et rend, par la suite, l'analyse facile et simplifiée. Aussi, elle perturbe moins le système et donc elle affecte pas

la performance du système. La seule difficulté qu'elle présente est le choix d'un bon échantillon.

2.4.2 La simulation

La simulation est définie comme l'utilisation de modèles numériques et logiques d'un système, d'un concept ou d'une opération pour en examiner son éventuel comportement dans le temps [14].

Conceptuellement la simulation modélise un système réel sous forme d'un programme spécialisé. Elle permet une modélisation à n'importe quel niveau de détails et support la collecte de toute métrique de performance, définissable et programmable.

La simulation est un bon outil pour évaluer des systèmes complexes ou inexistants. Nous pouvons alors, prendre des mesures réelles à l'aide d'outils de simulation et évaluer le comportement du système sous certaines circonstances. La simulation permettra la détection des anomalies de conception avant la construction du système.

Les modèles de simulation peuvent être vus sous trois angles différents, à savoir la simulation continue et discrète, la simulation statique et dynamique et enfin la simulation déterministe et stochastique.

La simulation continue est rarement utilisée pour analyser les performances des systèmes informatiques. Elle concerne la modélisation, sur un axe temporel, d'un système via une représentation ou les variables d'état changent constamment avec le temps. Quant à la simulation discrète, elle est composée de deux catégories : simulation dirigée par événements et simulation basée sur les cycles.

- *La simulation dirigée par événements* : elle modélise l'activité d'un système comme une série d'événements asynchrones qui se produisent à intervalle irrégulier. Ce type de simulation permet de modéliser une grande variété de systèmes.

- *La simulation basée sur les cycles* : dans ce type de simulation tous les changements observés dans l'état du système sont synchronisés avec une horloge unique. La simulation est essentiellement, une machine à états finis qui change

d'état à chaque coup d'horloge. Elle sert souvent à modéliser le noyau des processeurs.

La simulation est dite statique quand le modèle représente le système à un moment donné où quand il décrit un système ou le temps ne joue aucun rôle. La simulation dynamique est une modélisation d'un système évoluant avec le temps.

Lorsque le modèle de simulation ne contient aucune composante probabiliste, il est dit déterministe. Dans ce type de simulation, les sorties sont bien définies quand l'ensemble des entrées et l'ensemble des relations entrées/sorties sont déterminés. Toutefois, presque tous les systèmes réels possèdent des entrées aléatoires et donc nécessitent une simulation stochastique. Notons que les résultats d'un tel modèle sont, eux même, aléatoires et doivent être considérés comme une estimation des caractéristiques réelles du modèle.

Une combinaison entre les trois dimensions peut être effectuée pour aboutir à un bon simulateur. En effet, la majorité des modèles de simulation pour l'analyse des performances des systèmes informatiques sont discrets, dynamique et stochastiques.

La simulation possède cependant certains inconvénients. La conception et l'implantation d'un simulateur sont des tâches complexes. La simulation, elle-même, requiert une certaine expertise de la part de l'évaluateur et beaucoup de temps d'exécution. Cependant, une fois développé, le simulateur peut être utilisé de façon répétitive, utile et économique.

2.4.3 Les modèles analytiques

Les modèles analytiques d'un système sont un ensemble d'équations mathématiques dont des variables indépendantes (les entrées), produisent un ensemble unique de variables dépendantes (les sorties) [14].

Les modèles de performance analytiques sont des formulations mathématiques qui décrivent les aspects les plus pertinents d'un système informatique. En effet, ils se basent sur les méthodes d'analyse mathématique et ainsi ils sont la transcription mathématique du système. Les modèles analytiques constituent un bon outil pour

évaluer rapidement la performance d'un nouveau système ou d'un système qui vient d'être modifié.

Toutefois, les systèmes d'aujourd'hui sont tellement complexes qu'il s'avère difficile de les traiter à l'aide d'une approche si simple que la modélisation analytique. Cependant, cette approche a prouvé son efficacité dans l'analyse des systèmes en ligne et orienté transaction. Ceux-ci sont les plus difficiles à étudier par une simulation ou autre méthode d'analyse [1].

Par ailleurs, la modélisation des systèmes par des méthodes analytiques repose surtout sur la branche de mathématique appelée la théorie des files d'attente. Les chaînes de Markov sont aussi très utilisées. Donc, pour profiter de ces modèles, une bonne connaissance mathématique est nécessaire [23] et [18].

Un modèle de file d'attente est la représentation du système en un ensemble de centres de services traitants les demandes des utilisateurs. Un centre de service peut être n'importe quelle ressource active du système, un processeur par exemple. Les utilisateurs sont les tâches, les transactions ou les arrivées au centre de service. Afin d'analyser un système de files d'attente, nous avons besoins d'un certain nombre d'information. En effet, l'étude doit se faire en se basant sur la description du processus d'arrivée, le mécanisme de service et les caractéristiques de la file d'attente.

Un autre modèle analytique assez utilisé est la chaîne de Markov. Celle-ci est un automate à état fini avec une probabilité pour chaque transition. C'est-à-dire la probabilité que l'état suivant soit S_2 alors que l'état actuel est S_1. Cette probabilité est supposée indépendante du temps. En fait, les chaînes de Markov sont basées sur le processus de Markov qui est un processus dont l'état futur ne dépend de son passé qu'à travers son état à l'instant courant. Autrement dit, le présent étant connu, le futur est indépendant du passé [18].

Les trois techniques d'évaluation de performance ainsi présentées, le choix entre les différentes techniques pose souvent des problèmes. Nous présentons dans la section suivante une comparaison entre les différentes techniques d'évaluation.

[1] http://www.dmi.usherb.ca/personnel/prof_info/gabriel.girard/cours-ift628/notesdecours/performance/node1.html

2.4.4 Comparaison des techniques

Les trois techniques d'évaluation des performances sont la mesure du système réel, la simulation et la modélisation analytique. Il y a plusieurs considérations qui aident dans le choix de la technique à utiliser [19]. Les critères de sélection sont résumés dans le tableau 2.1.

Le plus important des paramètres à prendre en compte est l'étape du cycle de vie ou se trouve le système à étudier. Concernant la mesure, par exemple, elle ne peut être faite que si le système ou son prototype existent. Le temps disponible pour effectuer l'évaluation est un deuxième critère à considérer. La technique la plus rapide est sans doute la technique analytique.

En outre, la disponibilité des outils d'analyse de performance devrait être tenu en compte lors du choix de la technique. Les outils peuvent être les habiletés de modélisation, les langages de simulation ou les instruments de mesure. Par ailleurs, le niveau de précision désiré est un autre point important. Effectivement, la modélisation nécessite souvent des simplifications et hypothèses qu'il faut bien gérer. La simulation peut fournir beaucoup de détails avec le moins d'hypothèses possible.

En général le but d'une évaluation de performance est soit de comparer différentes alternatives ou trouver la valeur optimale des paramètres. Les méthodes analytiques donnent de bons résultats concernant l'effet de divers paramètres et leur interaction. Un dernier point reste à considérer dans le choix de la technique à savoir le coût. Notons que la mesure est une technique très coûteuse.

Techniques / Critères	Modélisation analytique	Simulation	Mesure
Etape dans le cycle de vie	Peu importe	Peu importe	Prototype
Temps demandé	Petit	Moyen	Variable
Outils	Analystes	Langages de programmation	Instruments
Précision	Faible	Modérée	Variables
Compromis de l'évaluation	Facile	Modéré	Difficile
Coût	Faible	Moyen	Elevé

Tableau 2.1 : Comparaison des techniques d'évaluation des performances

D'après le tableau 2.1, il est évident que la simulation est de loin la technique la plus attrayante toutefois la mise en place d'un simulateur efficace est une tache complexe et assez onéreuse. Le choix d'une technique ou d'une autre doit normalement être guidé par la disponibilité des outils, le but et l'importance de l'étude.

En outre, il est à signaler que la liste du tableau 2.1 n'est pas exhaustive et d'autres critères peuvent être rajoutés au tableau afin de faire le bon choix. Par ailleurs, il est utile, des fois, de combiner deux ou plusieurs techniques simultanément. Cette façon de faire permet de regrouper les avantages de plusieurs techniques et donc rendre l'étude de performance plus précise et plus fiable. Par exemple, nous pouvons utiliser la simulation et la modélisation analytique pour vérifier et valider les résultats de chacune des techniques.

2.5. Conclusions

Dans ce chapitre, nous avons présenté le problème de performance et son importance. Nous avons mis l'accent sur son rôle durant la vie d'un système. Nous avons ensuite décrit l'approche systémique pour mener une étude de performance. Sachant que les

systèmes diffèrent mais la méthodologie reste pratiquement la même pour toute évaluation de performance correcte.

Une étape importante dans l'étude de la performance des systèmes et le choix de la technique d'évaluation. Nous nous sommes focalisés sur la description des différentes techniques à savoir la mesure, la simulation et la modélisation analytique. Pour chaque technique nous avons présenté les principes de base puis une comparaison des dites méthodologies a été donné. Les critères de choix pouvant être variables et nombreux, nous avons exposé les plus pertinents.

Dans le chapitre suivant nous nous consacrerons à un autre point aussi important et déterminant que le choix de la technique : la caractérisation de la charge de travail. Nous présenterons les étapes de caractérisation et la modélisation de la charge de travail. Nous introduirons par la même occasion les processus stochastiques souvent utilisés pour modéliser une telle charge.

Chapitre 3

CARACTÉRISATION ET MODELISATION DU TRAFIC

La croissance du nombre d'utilisateurs et des services Web qui demandent beaucoup de ressources tel que le commerce électronique, pose des problèmes de performance. En fait, non seulement il est nécessaire de fournir un service efficace en terme de fiabilité et de rapidité, mais aussi il est important d'anticiper l'évolution du trafic afin de maintenir une bonne qualité de service.

La performance d'un système distribué composé de plusieurs clients, des serveurs et de réseaux, dépend étroitement des caractéristiques de sa charge. Ainsi, une bonne étude d'évaluation de performance passe tout d'abord par la caractérisation de la charge de travail. Donc, il est primordial de comprendre les propriétés statistiques du trafic traversant le système. Par la suite, nous pourrons étudier et développer un modèle de charge de travail approprié pour une évaluation de performance.

Dans ce chapitre nous nous intéresserons à la charge de travail. Tout d'abord, nous définirons des notions importantes concernant celle-ci. Ensuite, nous présenterons quelques généralités sur la charge de travail puis nous exposerons une classification des différents modèles de charges de travail destinées aux tests. Par la suite, nous énumérerons les étapes nécessaires et typiques pour la caractérisation de la charge de travail ainsi que les techniques utilisées. Enfin, Nous terminerons par expliciter les

spécifications du trafic que nous avons étudié et qui sont des propriétés générales pour tous les trafics Web.

3.1. Terminologie

Avant d'aller plus loin dans le sujet, nous jugeons utile de définir quelques termes utilisés dans la suite [4].

- **Caractérisation de la charge de travail :** dans le but de tester plusieurs alternatives sous les mêmes conditions, la charge de travail doit être répétable. Puisque l'environnement réel de l'utilisateur est généralement non répétable, il est nécessaire d'étudier cet environnement, observer les caractéristiques principales et développer un modèle de charge de travail à utiliser répétitivement. Ce processus est appelé caractérisation de la charge de travail.

- **Composante de base :** les données résultantes de la mesure de la charge sont constituées des services demandés ou des demandes de ressources. Ceux-ci sont initiés par un nombre d'utilisateurs du système. Le terme *utilisateur* représente l'entité qui envoie les requêtes de service vers le système testé. Dans la littérature de la caractérisation de la charge de travail le terme *composante de base* est utilisé à la place d'*utilisateur*.

- **Paramètres de la charge de travail :** les quantités mesurées, demandes de service ou demandes de ressource, employées pour modéliser ou caractériser la charge de travail sont dites paramètres de la charge de travail. Les types de transactions, les instructions, la taille des paquets ou le nombre d'utilisateur sont des exemples de paramètres.

3.2. Généralités

La charge de travail ou « Workload » d'un système peut être définit comme l'ensemble de toutes les entrées que le système reçoit de son environnement et ce durant une période de temps déterminée. Par exemple si le système étudié est un serveur de base de données, alors sa charge de travail consiste en toutes les transactions traitées durant un temps d'observation. En fait, La performance est la réaction du système à une charge de

travail spécifique. Il est donc essentiel d'utiliser le bon « Workload » lors de l'évaluation du système.

Le processus de caractérisation du « Workload » décrit, avec précision, la charge de travail globale du système du point de vu composantes. Chaque composante fait l'objet d'une partition en composantes de base (figure 3.1). Les composantes de base sont caractérisées par les deux paramètres : *Intensité de la charge de travail* et *Demande de service* pour chaque ressource [4] et [14].

L'intensité de la charge de travail donne une mesure de la charge subite par le système en terme d'unités de travail arrivant au système (nombre de requête/sec, nombre de cliques/jour). Quand à la demande de service elle spécifie le total des temps de service de chaque composante de base. Le temps de service est le temps nécessaire à une ressource pour traiter une donnée (le temps d'attente étant exclu).

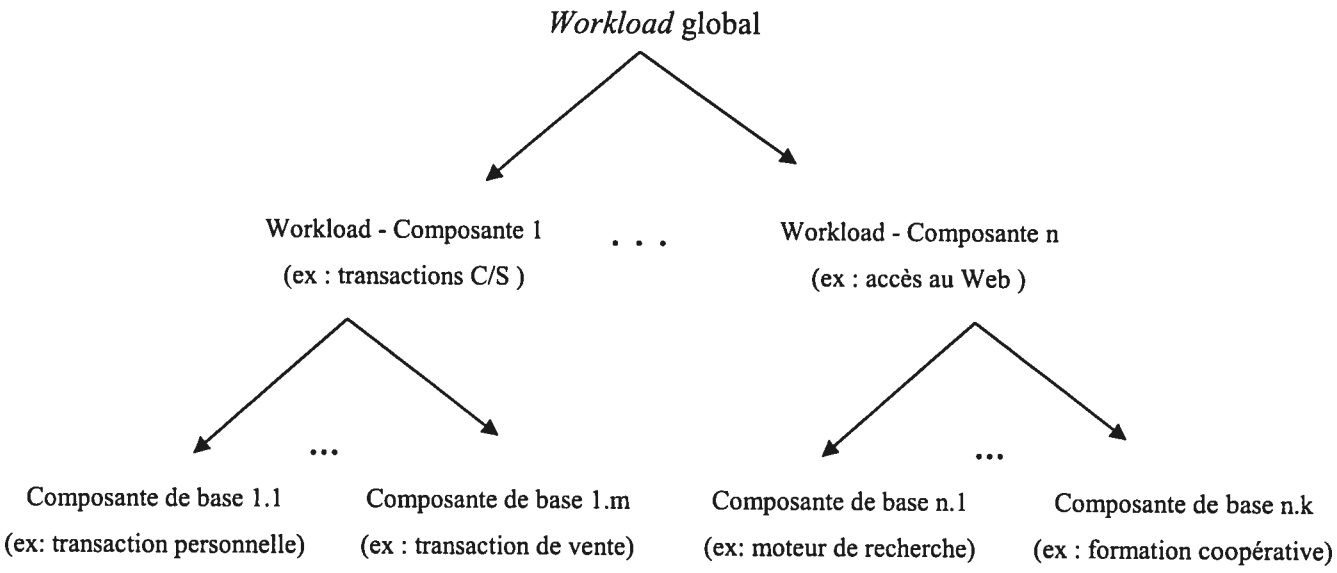

Figure 3.1 : processus de caractérisation de la charge de travail

La description quantitative de la charge du travail est dite caractérisation de la charge de travail. Elle est exprimée en fonction des paramètres susceptibles d'influencer le comportement du système et qui servent à produire la charge de travail. La caractérisation de la charge est indispensable et fondamentale dans toutes les études

d'évaluation de performance. Elle est vitale et sert de point d'entrée pour la conception de modèles de charge de travail.

Il est certainement difficile de gérer un « Workload » réel vu le grand nombre d'éléments qu'il comporte. Cependant, et dans le but de solutionner des problèmes pratiques, on est amené à synthétiser l'information servant à la description de la charge de travail. En d'autres termes, nous avons besoin d'élaborer un modèle qui ne prend en compte que les caractéristiques les plus pertinentes.

Un modèle de charge de travail sert à orienter la charge de travail d'un système réel vers une image synthétique, utilisable et réaliste. Le modèle peut aussi être considéré comme entrée d'un autre modèle du système évalué. Les objectifs derrière l'utilisation de modèles de charge de travail sont [1]:

1. Disposer d'une représentation de la charge de travail pour une évaluation comparative de la performance de différents systèmes.
2. Disposer d'un environnement contrôlé et reproductible pour des études expérimentales optimisées de performance.
3. Réduire la quantité de données à analyser.
4. Présenter la charge de travail du système sous une forme requise par un autre modèle du système.

La charge de travail est définit par rapport à un certains nombre de caractéristiques et de paramètres. Le choix de ceux-ci est lié au but de l'étude. En réalité, un système qui est soigneusement réglé pour une charge de travail spécifique peut ne pas atteindre les objectifs de performances. Cet échec est surtout engendré par le fait que le système réel possède des caractéristiques différentes du modèle. Ainsi, la flexibilité et la contrôlabilité des caractéristiques de la charge de travail s'avèrent des propriétés intéressantes dans un modèle.

Les modèles de la charge peuvent être effectués à différents niveaux. En particulier, au niveau physique, virtuel et fonctionnel [1]:

- Niveau physique *:* le modèle est basé sur la mesure de la consommation de ressources matérielles ou logicielles du système. Il est surtout utile lorsque

[1] http://www.dmi.usherb.ca/personnel/prof_info/gabriel.girard/cours-ift628/notesdecours/performance/node1.html

- l'objectif de l'étude est de mettre en œuvre un nouveau système ou de faire de la planification de capacité.
- Niveau virtuel : Pour ce modèle, la consommation est lié aux ressources virtuelles. Ainsi, les composantes de base sont caractérisées, entre autre, par le nombre d'accès à un fichier, la mémoire virtuelle utilisée et le type de commande interactive. Ce modèle est utilisé en particulier pour comparer les systèmes.
- Niveau fonctionnel : il est caractérisé par les applications contenues dans la charge de travail. Celle-ci peut intégrer, par exemple, des fonctions comme un programme de tri ou une compilation. Le modèle sert aussi bien pour une planification de capacité que pour la conception de nouveaux systèmes.

Il existe plusieurs types de modèles de charge de travail qui répondent aux divers besoins et particularités de certains environnements. Nous présenterons dans les sections qui suivent les grandes lignes de ces modèles.

3.3. Classification des modèles de la charge de test

La charge de travail utilisée pour des fins de mesure ou dans une étude de performance est dite charge de test. Celle-ci peut être considérée comme modèle de la charge de travail. Il est vrai que dans certains cas, la charge de test coïncide avec la charge réelle et naturelle que le système reçoit. Toutefois, il est extrêmement difficile de reproduire cette même charge réelle même en essayant d'assurer les mêmes conditions. Afin de comparer rigoureusement deux résultas, la reproductibilité d'une charge de test est primordiale.

Il existe plusieurs façons pour implanter un modèle de charge de test. Ainsi, nous trouvons une classification des modèles de charge de test selon la méthode suivie dans l'implémentation. En effet, trois classes sont définit à savoir les charges de tests réelles, les charges de tests synthétiques et les charges de tests artificielles [14], [23] et [1].

3.3.1 Les charges de tests réelles

La charge de test réelle peut être considérée comme un modèle. Pratiquement, il est, sans doute, le modèle le plus représentatif et le plus coûteux aussi. La charge de test réelle est

[1] http://www.dmi.usherb.ca/personnel/prof_info/gabriel.girard/cours-ift628/notesdecours/performance/node1.html

composée de tous les programmes et données naturels que le système manipule pendant une session de mesure. Elle est générée par la communauté des utilisateurs dans l'environnement original de production. La trace de la charge de travail est une autre forme de charge de test réelle. Elle consiste en une séquence chronologique de données représentant des événements particuliers survenus durant une session de test. Il est à noter que la trace est surtout utilisée pour les modèles de simulation.

La charge de test réelle, en général, ne peut être reproduite sous sa composition exacte. Cependant, si les propriétés statistiques de la charge de travail du système ne changent pas avec le temps, la charge de test est statistiquement reproductible. En outre, avec un tel modèle, le seul choix à faire est celui de la durée de mesure.

En fait, durant une évaluation du système orientée par une charge de test réelle, les intervalles de mesure doivent être suffisamment long. De plus, il est nécessaire d'analyser un grand nombre de données pour s'assurer de l'exactitude des statistiques. L'intervalle de mesure peut varier de quelques heures à quelques semaines lorsque la charge de travail est sujette à des changements significatifs durant cette période.

La charge de travail d'un système peut rester stationnaire pour une longue période de temps. Toutefois, ses caractéristiques changent lentement selon la tendance des utilisateurs, l'ajout ou la suppression d'applications. Les utilisateurs changent en fonction du système et tout changement dans leurs habitudes, entraîne des changements au niveau des caractéristiques de la charge de travail. Ainsi, à long terme la charge de test réelle n'est pas reproductible. En outre, le contrôle des entrées du système est quasi impossible. Donc, il est particulièrement difficile de vérifier l'effet de différentes caractéristiques de la charge sur la performance du système.

En pratique, les raisons qui empêchent l'utilisation, à grande échelle, de la charge de test réelle peuvent être énumérées comme telles [1] :

- La non flexibilité, puisque nous ne pouvons agir sur les programmes et leur consommation en ressources.
- Nécessité de garder les fichiers originaux pour une ré-exécution ultérieure. Ce qui peut entraîner des problèmes d'interférences et des pénalités économiques.

[1] http://www.dmi.usherb.ca/personnel/prof_info/gabriel.girard/cours-ift628/notesdecours/performance/node1.html

- La confidentialité de certains programmes et données. Ce qui empêchera toute manipulation concernant ces programmes et données.

D'autres types de charges de tests sont disponibles pour remédier à ces inconvénients comme les charges de tests synthétiques.

3.3.2 Les charges de tests synthétiques

Une charge de test synthétique a des caractéristiques semblables à celles d'une charge de travail réelle. En fait, elle est la représentation ou le modèle de la charge réelle. De plus, elle peut être utilisée répétitivement et d'une manière contrôlée. L'utilisation des charges de tests synthétiques trouve sa justification dans le fait qu'elles ne requièrent guère de fichiers de données du système réel. En réalité, ces derniers, peuvent être énorme de taille et contenir des données sensibles. De plus, non seulement modifier la charge synthétique s'avère une tâche simple mais aussi n'affecte pas les opérations.

La charge de test synthétique est composée soit d'un sous ensemble de composants de base de la charge réelle, soit un mélange de ces dits composants et des composants construits dans le but d'une certaine étude. Le premier type de charges synthétiques est appelé banc d'essais ou benchmark tandis que le second type est nommé les charges synthétiques hybrides.

Les principaux avantages des bancs d'essais se résument dans leur simplicité et leur application à une large gamme de systèmes. En effet, ils sont facile à concevoir et sont utiles pour des environnements complexes de multiprogrammation, de temps partagé, de multitraitement, de base de données et de temps réel. Effectivement, les bancs d'essais s'exécutent sur des machines réelles et dans des conditions aussi réelles ce qui leur donne tant de flexibilité. De surcroît, ils servent à évaluer aussi bien le logiciel que le matériel et peuvent comparer les performances d'un système avant et après certains changements.

Lorsque l'évaluation concerne un système interactif, un modèle de charge interactif doit être mis en place. La charge de test synthétique interactive est implantée à partir de scénarios. Un scénario est une description des fonctionnalités exécutée par une série de commande de l'usager. Ce dit scénario est explicité à l'aide d'un langage indépendant

du système dont la longueur doit être raisonnable puisqu'elle influence le coût du banc d'essais. L'exécution du scénario se fait par le biais d'un script. Celui-ci, n'est autre qu'une suite de commandes exécutables envoyées au système. Les scripts extraits de la charge réelle forment la charge de test synthétique naturelle (banc d'essais) pour un système interactif.

Toutefois, il n'est pas toujours possible de construire une charge de test basé sur des programmes réels. Effectivement, en tenant compte des considérations de sécurité ou du fait que les tâches réelles engendrent des changements dans le système, il est difficile d'appliquer une charge de test utilisant des programmes réels.

Les programmes réels sont alors remplacés par des programmes synthétiques ou des programmes noyaux. Un programme synthétique simule l'utilisation des ressources du système selon les spécifications et les caractéristiques du modèle de la charge de travail. Cependant, il ne fait aucun travail utile. Il existe deux façons pour construire un programme synthétique : soit du point de vue des demandes de ressources soit celui des demandes de services. La seconde manière de faire rend le programme synthétique indépendant de la configuration du système et du système d'exploitation. Ainsi, il peut être utilisé pour comparer deux systèmes différents.

Tandis que la structure « demande de ressources » pour les programmes synthétiques, peut être utilisée comme entrée d'un simulateur. Dans certains cas, la structure de demande doit être convertie en des programmes exécutables appelés tâches synthétiques. Celles-ci simulent les exigences de la CPU, des entrées/sorties et de la mémoire requise. Les caractéristiques d'une tâche synthétique, peuvent être modifiées grâce à ses paramètres de contrôle.

Les programmes noyaux sont caractérisés par une consommation de ressources connue et non modifiable. Leur champ d'application est moins large que celui des programmes synthétiques. Ainsi les modèles de charge contenant un mélange de programmes de la charge réelle et de programmes synthétiques ou noyaux sont dit hybrides synthétiques.

3.3.3 Les charges de tests artificielles

Elles n'utilisent aucune composante de base de la charge réelle. Par contre, ces modèles sont construits à base de programmes à objectif précis et de paramètres descriptifs. Il est possible de diviser les charges de tests artificielles en deux catégories : *modèles exécutables* et *modèles non exécutables*.

Les modèles non exécutables sont décrits par un ensemble de paramètres qui reflètent la même utilisation réelle des ressources. En outre, ils servent comme entrée d'un simulateur ou d'un modèle analytique. Tandis que les modèles artificiels exécutables consistent en une suite de programmes développés spécialement pour expérimenter les aspects du système. Il existe une multitude de charge de test artificielle exécutable. Nous présentons ci-dessous les plus importantes [14], [23] et [1].

- Le mélange d'instructions : il est basé sur l'analyse de la fréquence d'exécution des différentes instructions. C'est une spécification d'instructions couplées avec leur fréquence. En pratique, un programme ayant la même fréquence d'exécution qu'un ensemble de programme est considéré comme modèle. Par ailleurs, le mélange d'instruction est étroitement lié à l'utilisation de la CPU. En effet, ce modèle sert essentiellement à la comparaison et à la conception de nouveaux processeurs. Le principal inconvénient du mélange d'instructions et qu'il ne permet pas d'évaluer le logiciel.

- Les traces : la trace est un enregistrement d'événements présélectionnés préservant l'ordre dans lequel ces événements se sont produits. Un événement peut être un changement d'état du système, un début de tâche ou une exécution d'une instruction. Ce modèle de charge de test est utilisé pour orienter les modèles de simulation surtout quand l'ordre est important dans l'analyse. En outre, il est possible de considérer une trace issue d'une charge réelle comme d'une charge représentative. Toutefois, pour certains objectifs, la trace reste insuffisante et non significative. Par suite, elle doit être manipulé avec précaution.

[1] http://www.dmi.usherb.ca/personnel/prof_info/gabriel.girard/cours-ift628/notesdecours/performance/node1.html

- Les pilotes interactifs : la charge de travail d'un système interactif est largement influencée par le comportement des utilisateurs. Les modèles de charge pour ce type de système doivent prendre en considération aussi bien les utilisateurs que les programmes. Un pilote interactif n'est pas, en soi, un modèle de charge mais un générateur de charge de test. Il consiste en un modèle de l'interface utilisateur et un modèle des exigences des traitements utilisateurs. Il intercepte les messages dirigés vers l'usager et simule l'action appropriée. Deux variantes de pilotes interactifs sont utilisées : *les pilotes internes* lorsque le programme est exécuté sur le système testé et *les pilotes externes* lorsqu'il est connecté au système mesuré mais s'exécute sur un composant séparé.

De surcroît, les programmes synthétiques et les programmes noyaux, déjà présentés, sont aussi considérés comme des charges de tests artificielles.

Il est à signaler que la conception et l'implantation de modèles complexes de charges de tests peuvent s'avérer très coûteuses. Dans certaines études, une mesure simple concernant l'importance de la charge peut suffire. Nous résumons les différentes charges de tests dans la figure ci-dessous qui présente leur classification.

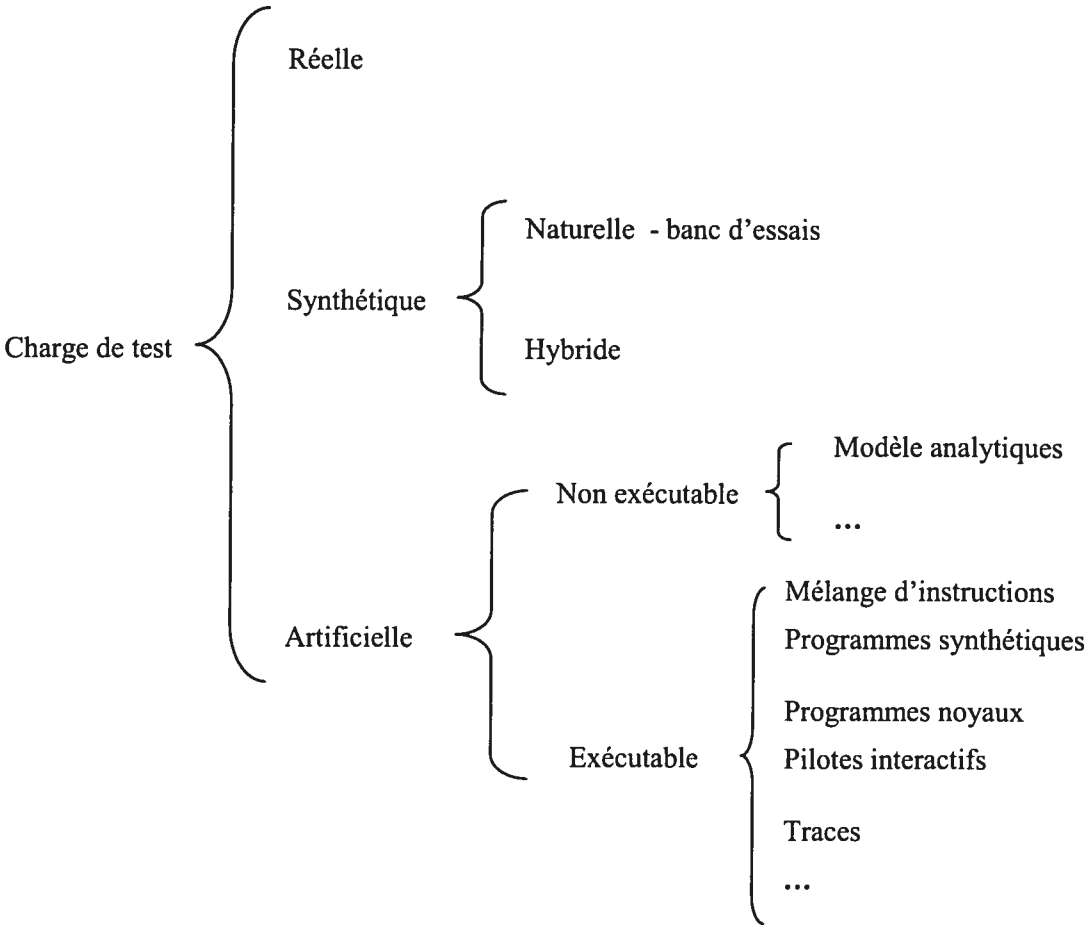

Figure 3.2 : Classification des charges de travail de test

Une fois les différents types de charges de tests présentés, il reste à exposer les étapes suivies pour construire un modèle de charge de travail ainsi que les diverses techniques.

3.4. La caractérisation de la charge de travail

Le trafic Internet est en perpétuelle croissance. Les charges générées par les multiples applications Web posent des problèmes de performance. Caractériser la charge de travail constitue la clé de toute étude d'évaluation de performance. Par ailleurs, comprendre les caractéristiques de la charge de travail est extrêmement important pour la conception des systèmes informatiques. Pour ce faire, nous disposons d'une méthodologie et de techniques de caractérisation.

3.4.1 Les étapes de caractérisation

Quelque soit la nature de la charge de travail, une méthodologie commune sert à comprendre et à mettre en exergue les caractéristiques de cette charge de travail. L'approche présentée se focalise sur une caractérisation de charge orientée ressources. Elle propose les étapes suivantes [4] :

1. Choix du point de vue : la première chose à faire avant de commencer l'étude de la charge de travail est de définir clairement le point de vue de l'analyse. En effet, la charge de travail peut être considérée sous différents angles, entre autre, vue du côté serveur, vue du côté client ou vue du coté réseau. Le processus de caractérisation de la charge n'est valide que lorsqu'on établit l'aspect selon lequel la charge de travail va être caractérisé. Cette étape est en quelque sorte l'étape d'étude des besoins.

2. Identification des composantes de base : dans cette étape, nous identifions les composantes de base de la charge de travail. Le choix de ces composantes dépend étroitement de la nature du système et du but de la caractérisation. Les transactions et les requêtes constituent les composantes les plus utilisées. A la suite de cette étape, nous devrons être en mesure d'affirmer que la charge étudiée repose essentiellement sur telle et telle composante.

3. Choix des paramètres de caractérisation : chaque composante, préalablement identifiée, peut être représentée par un certains nombre de paramètres. A ce niveau, se fait le choix des paramètres caractérisants chaque composante de base de la charge étudiée. En fait, la nature des paramètres est dictée par l'information, concernant les entrées, requise par les modèles analytiques. En outre, il est possible de représenter chaque composante par deux types de paramètres. Le premier type concerne l'*intensité de la charge de travail* (taux d'arrivée, nombre de clients, temps de réflexion ou « think time ») alors que le second fait référence aux *demandes de service* (l'utilisation de CPU, le temps des entrées/sorties).

4. Collecte de données : cette étape nous permet d'assigner des valeurs aux paramètres des composantes du modèle. Elle inclut les tâches suivantes :

- Identification de la session de mesure la plus appropriée pour bâtir les analyses. En effet, l'observation du système, de la charge et des indexes de performance pendant une durée dépendante de la nature du système, permet de choisir la bonne session de mesure.
- Pilotage et mesure des activités du système durant la session de mesure choisie. Il existe plusieurs outils de mesure donc une bonne sélection des outils et moniteurs s'impose.
- Affectation de valeurs aux paramètres de caractérisation des composantes de base et ce en se basant sur les données collectées.

5. Classification de la charge de travail : la charge de travail dans le monde réel est un ensemble de composantes hétérogènes. En effet, c'est un amalgame d'applications de différents types. Ainsi représenter la charge par une seule classe est un manque de rigueur et de précision pour le modèle. L'objectif de cette étape est de grouper les composantes similaires. La similitude des composantes peut être évalué sous maints aspects à savoir la consommation de ressources, l'appartenance à une application, la position géographique (locale ou distant), le mode de traitement (interactif, transactionnel ou batch) ou la fonctionnalité entretenue.
6. Calcul des paramètres de la classe : comme nous l'avons vu dans la quatrième étape, chaque composant est caractérisé par ses paramètres. Suite à la répartition de la charge de travail en classes, le problème qui se pose est comment calculer les valeurs des paramètres représentatifs d'une classe de composants ? différentes techniques répondent à la question. Ces techniques seront présentées dans la section suivante.

3.4.2 Les techniques de caractérisation

Nous disposons d'une grande variété de caractéristiques des demandes de service ou de ressource. Toutefois, en pratique et pour des raisons à la fois de simplification et de pertinence, seules les caractéristiques ayant un impact significatif sur la performance

sont retenues. Elles représenteront la charge de travail. Les techniques énumérées plus bas, ont largement servi afin de caractériser la charge de travail.

- La moyenne : la façon la plus simple pour qualifier les paramètres d'une classe de composants de la charge de travail, est de calculer la moyenne de toutes les valeurs de paramètres observées. Cependant, dans certains cas la moyenne est inappropriée et reste inexpressive. Par suite, la médiane, le mode, la moyenne géométrique ou harmonique sont fortement recommandés. Plus de détails sur ces valeurs et leur utilisation sont présentés dans [19].

- Indication de la dispersion : la moyenne est insuffisante lorsque les valeurs des données sont très variables. La *variance* spécifie la variabilité des données et leur disposition autour de la moyenne. Toutefois, la racine carrée de la variance, appelée *écart type*, est souvent plus significative car elle est exprimée en même unité que de la moyenne. Le rapport de l'écart type à la moyenne est dit *covariance*. Une covariance égale à zéro implique une variance nulle et indique par la même occasion que le paramètre mesuré est constant. Une covariance élevée veut nécessairement dire que la moyenne est insuffisante. En fait, si la covariance est élevée il serait plus judicieux soit de diviser les utilisateurs en classes homogènes soit de considérer l'histogramme complet. L'histogramme est l'objet du prochain point [19].

- Histogramme à monoparamètre : un histogramme présente les fréquences relatives des différentes valeurs d'un paramètre. Pour des paramètres à valeur continue, il est nécessaire de diviser l'intervalle de valeurs en plusieurs sous intervalles appelés cellules. Puis dénombrer les observations qui appartiennent à chaque cellule. Ces données sont alors utilisées dans des modèles de mesure ou de simulation afin de générer une charge de test. En outre, de tels histogrammes sont utiles lors de la détermination ou la vérification de la fonction de distribution dans les modèles analytiques. Le problème principal des histogrammes à un seul paramètre est le fait qu'ils ne tiennent pas en compte de la corrélation entre paramètres. En faisant appel aux histogrammes multiparamètres, ce problème est résolu [19].

- Histogramme multiparamètres : lorsqu'il existe une certaine corrélation entre les paramètres, l'histogramme multiparamètres s'avère intéressant. Il est présenté sous

forme d'un nuage de points mettant en évidence la relation entre les données. Notons toutefois, qu'il est difficile de tracer des histogrammes multiparamètres dès que le nombre de paramètres dépasse deux. C'est pour cette raison qu'ils sont rarement utilisés [19].

- Analyse en composantes principales : une méthode largement adoptée pour classifier les composants de la charge de travail est la somme pondérée des valeurs de ses paramètres. En fait, à chaque paramètre est attribué un poids et la somme est employée pour distinguer les composantes. Par exemple, des composants faibles en demande, moyen ou de demande élevée. Le choix du poids n'est pas arbitraire, il se fait grâce à l'analyse en composantes principales. Cette analyse est une procédure mathématique qui transforme un ensemble de paramètres, corrélés éventuellement, en un ensemble réduit de paramètre non corrélés. Ceux-ci sont appelés les *facteurs principaux* [19].

- Les modèles de Markov : dans certains cas, il est important non seulement d'avoir le nombre de demande de service mais aussi leur ordre chronologique. En effet, dans bien de situations la prochaine requête est déterminée par des requêtes précédentes. Ainsi, quand l'état futur du système dépend seulement de son état courant, le système suit le modèle de Markov. De tels modèles connaissent un grand succès dans l'analyse des files d'attente. En général, le modèle est décrit à l'aide d'une matrice de transition donnant les probabilités pour chaque état futur possible sachant que le système est à l'état courant. Les probabilités de transitions donnent une image plus fidèle sur l'ordre des requêtes ainsi sur l'exécution du la charge [18].

- Regroupement : pratiquement la charge de travail mesurée consiste en un grand nombre de composantes. Pour des besoins d'analyse, il est plus simple de classifier ces composantes. Il est aussi préférable que les nouvelles classes regroupent un petit nombre de composantes présentant des similarités. Ainsi, un seul membre de chaque groupe représentera sa classe lors d'une éventuelle étude sur l'effet d'une certaine charge sur la classe de composantes. Pour bien définir les classes, la littérature propose plusieurs métriques de distance [14]. En outre, la variance intergroupe doit être grande alors que la variance intragroupe petite. Deux catégories de techniques sont utilisées : hiérarchique et non hiérarchique [19].

Comprendre et caractériser la charge de travail répond aux questions clés de toute étude de performance. Cependant, l'étude du trafic ne serait pas complète sans présenter ses propriétés. En effet, récemment des études ont prouvé que le trafic LAN, Web et particulièrement le trafic des applications de commerce électronique montrent des propriétés jusque là non considérées par les études de recherches.

3.5. Les propriétés du trafic

Les applications réseaux en général et Internet en particulier ne cessent de se développer. Ainsi à l'avènement des communications à large bandes caractérisées par un trafic hétérogène, les hypothèses communément adoptées concernant les modèles traditionnels de trafic sont remises en cause. En fait, lors de la modélisation d'une communication réseau, le modèle de poisson était fort présent puisqu'il présentait des propriétés théoriques attractives.

Toutefois, plusieurs études [11], [25] et [27] ont montré que le trafic aussi bien LAN que Web a des propriétés non considérées par les modèles traditionnels, à savoir l'autosimilarité et la non-stationnarité.

3.5.1 Trafic auto-similaire

Ce n'est que récemment que les chercheurs ont mis à l'évidence l'aspect d'auto-similarité des trafics réseau. Cette découverte a complètement remis en cause la façon suivie lors de la modélisation des charges de réseaux. En effet, les processus et distributions habituellement utilisées telle que la distribution de poisson, ne permettent pas de modéliser un trafic auto-similaire. Par suite, la charge de test ne reflète pas une charge de travail réelle.

Intuitivement, les phénomènes auto-similaires présentent des similarités structurelles à travers toutes les échelles de temps, sinon sur une large tranche de l'échelle temporelle. Concernant le trafic réseau, l'auto-similarité se manifeste dans l'absence d'une longueur naturelle de rafales. En fait, à chaque niveau de l'échelle de temps, allant de quelques millisecondes à des heures, la transmission en rafales consiste en des sous périodes de transmissions en rafales. Celles-ci sont séparées par des périodes ou la transmission connaît moins de rafales.

L'auto-similarité est une propriété des objets fractals dont l'apparence est invariante quelque soit l'échelle d'observation. Donc un trafic auto-similaire présente des transmissions par rafales sur un éventail d'échelle de temps. Un tel trafic est, par conséquent, caractérisé par une forte corrélation entre ses valeurs. En effet, les valeurs observées à tout moment sont liées à toutes les valeurs futures. Ces constats ont de lourdes conséquences sur la performance lorsqu'ils ne sont pas pris en considération par le modèle.

Effectivement, durant les périodes de congestion du réseau, la congestion devient persistante et peut entraîner de grandes pertes. En outre, pour un trafic auto-similaire, les périodes de congestion ne sont pas prévisibles contrairement aux modèles classiques. Ces implications, défient la performance des systèmes. Ainsi les modèles de charge de travail doivent prendre en considération la notion d'autosimilarité.

Les modèles de processus auto-similaire ont l'avantage d'exprimer le degré d'autosimilarité d'une série à l'aide d'un seul paramètre. Ce dernier est appelé paramètre de *Hurst* noté **H**, il est compris entre 1/2 et 1 strictement. Plus **H** tend vers 1 plus l'autosimilarité est forte. Toutes les méthodes de test de l'autosimilarité des modèles se basent sur l'évaluation du paramètre de *Hurst* en le comparant à la valeur 1/2.

Des études [9], [11], [13], [24] et [27] ont montré que dans certains cas, l'autosimilarité du trafic s'explique parfaitement par les caractéristiques du système de fichier et par comportement de l'utilisateur. Ce dernier, est caractérisé à la fois par des périodes d'intense activité et de longue périodes d'inactivité. En pratique, un trafic auto-similaire peut être généré en combinant un grand nombre de sources ON/OFF. Autrement dit, des sources ayant des périodes d'activité et d'inactivité. De plus, la longueur de ces périodes doit être caractérisées par des distributions à queue lourde.

Une distribution est dite à queue lourde si sa forme est asymptotiquement hyperbolique. Statistiquement, cela veut dire que les observations ayant des valeurs très grandes, ont une probabilité non négligeable. Ce type de distribution attire de plus en plus les concepteurs de systèmes pour des objectifs de simulation. Malheureusement, elles présentent des propriétés très différentes des distributions usuelles. De telles propriétés rendent la stabilité d'une simulation un objectif difficile à atteindre.

3.5.2 Trafic non stationnaire

Plusieurs travaux d'analyse du trafic Web supposent qu'il est approximativement stationnaire. Toutefois, des études récentes [10] portant sur l'analyse du trafic de commerce électronique, ont montré que celui-ci est non-stationnaire même à de petits intervalles de temps.

N'importe quelle analyse de la charge basée sur la notion de « jour typique » peut être complètement fausse, puisqu'il n'y a pas de « jour typique ». En effet, sans la prise en compte de la non-stationnarité, un générateur artificiel de charge ne pourra produire un trafic semblable au trafic réel.

La non-stationnarité devient parmi les caractéristiques fondamentales du trafic Internet. Elle affecte considérablement plusieurs variables mesurées sur une ligne du réseau. Par suite, un modèle de charge de travail qui néglige de telles variations, sera moins proche de la charge réelle. La non-stationnarité a été établie pour le trafic Web au niveau TCP [8] et au niveau HTTP [10].

Nous pouvons modéliser un trafic non-stationnaire suivants deux saveurs : paramétrique et non paramétrique. Les modèles paramétriques sont faciles à caractériser puisqu'ils disposent d'un ensemble connu de paramètres. Cependant, ils sont moins utilisés en pratique. Quant aux modèles non paramétriques, ils sont les plus répandus pour la modélisation du trafic Internet et commerce électronique en particulier. Ils sont par contre, beaucoup plus difficiles à caractériser.

En pratique, certaines hypothèses doivent être considérées afin de simplifier le processus de caractérisation de la non-stationnarité. En effet, le trafic est supposé stationnaire sauf pour des intervalles plus larges que les intervalles durant lesquels les mesures sont faites. Autrement dit, ceci nous permet de décider à quelle échelle nous devons nous soucier de la non-stationnarité. Cette façon de faire permet de mieux représenter les propriétés de la charge de travail que lorsque la non-stationnarité est totalement ignorée. Par conséquent, l'évaluation de la performance sera plus pertinente.

3.6. Conclusion

Tout au long de ce chapitre nous avons mis l'accent sur le point clé de toute étude de performance. Effectivement, comprendre et caractériser la charge de travail est l'une des étapes primordiales dans le processus d'évaluation des performances des systèmes. Plus la charge de test est réaliste plus les résultats de l'analyse sont convaincants.

Ainsi nous nous sommes attardé sur la caractérisation et la modélisation de la charge de travail. Tout d'abord nous avons présenté les différents modèles de charge de travail. Ensuite nous, avons énuméré les étapes communes pour caractériser cette charge de travail ainsi que les techniques de caractérisation. Enfin, nous avons présenté deux propriétés du trafic Internet en général et du commerce électronique en particulier à savoir l'autosimilarité et la non-stationnarité. Ces propriétés ont été récemment démontrées et remettent en question les modèles traditionnels tel que le modèle de poisson.

Dans le chapitre suivant nous présenterons une vue générale sur les applications de type commerce électronique. Celles-ci feront l'objet de notre étude de performance. Nous donnerons dans un premier temps une description des applications e-commerce. Puis nous aborderons le problème de la performance des serveurs de commerce électronique pour enfin conclure avec la présentation du serveur GNP qui fera l'objet de notre étude de cas.

Chapitre 4

LES APPLICATIONS DE COMMERCE ELECTRONIQUE

Aucun événement n'a affecté la manière de pratiquer du commerce comme l'a fait l'énorme expansion de l'Internet. Accessible de par le monde, il est devenu un élément essentiel de notre vie quotidienne. De plus il a introduit de nouvelles façons et méthodes pour développer le commerce à savoir le commerce électronique. Avant tout, les transactions de e-commerce font partie du trafic Web. Cependant, elles ont des particularités que nous éluciderons dans ce qui suit.

Dans la suite de ce chapitre nous allons exposer les principes de bases du e-commerce. Un deuxième volet de ce chapitre sera consacré aux aspects de performance concernant les serveurs de commerce électronique. La présentation du serveur GNP, de son environnement et son architecture fera l'objet du dernier point. GNP constitue la plate forme qui a servi aux différents tests que nous avons réalisés.

4.1. Description

Le commerce électronique est l'automatisation de transactions commerciales utilisant des ordinateurs et des technologies de communication. Techniquement, c'est une activité commerciale qui s'effectue sur des réseaux reliant des dispositifs électroniques (principalement des ordinateurs). Fondamentalement, il désigne un moyen peu coûteux de relier des ordinateurs pour effectuer des tâches qui exigent depuis toujours beaucoup

de temps et d'argent de la part des entreprises. Il s'agit par exemple de la vente de produits, de la facturation, du contrôle des inventaires et de la communication avec les fournisseurs et les clients.

Le commerce électronique n'est pas en soi un phénomène nouveau. En effet, les échanges électroniques, notamment grâce aux échanges informatisés de données (EDI), se sont largement développés au cours des dix dernières années entre entreprises et entre entreprises et administrations. Toutefois, il était resté jusque là largement réservé aux utilisations interentreprises. Le développement très rapide de l'Internet bouleverse ces perspectives surtout en ce qui concerne les ventes aux consommateurs. En effet, le commerce électronique permet d'offrir un service de qualité aux clients à des prix compétitifs en réduisant le risque d'erreurs et le temps de traitement des procédures. Les entrepreneurs sont maintenant capables de fonder leurs propres commerces plus aisément et avec un petit fond d'investissement. Toutefois, certains problèmes doivent être solutionnés avant de démarrer toute initiative dans ce domaine. Il est crucial de prêter un intérêt particulier à la sécurité, au choix du mode de payement, aux aspects juridiques et fiscaux et à l'évaluation continuelle de la performance du système.

En fait, les facteurs importants qui interviennent dans la réussite d'un système de commerce électronique peuvent être classés en trois classes : stratégique, technique et fonctionnelles. Dans la première classe nous retrouvons la stratégie de gain, la politique de marketing et l'établissement d'un environnement ergonomique pour les clients. La classe technique s'intéresse à la qualité de service tel que le temps de réponse, le débit des transactions, la qualité du son et de l'image, la fiabilité et bien d'autre. La dernière classe se focalise sur les aspects fonctionnels du système à savoir la procédure d'achat, de vente, d'affichage, de recherche d'informations, de négociation, de livraison…etc.

Nous pouvons illustrer ces différentes fonctionnalités, les divers acteurs qui rentrent en jeu et leur interactivité, d'une manière simplifiée, grâce au schéma suivant :

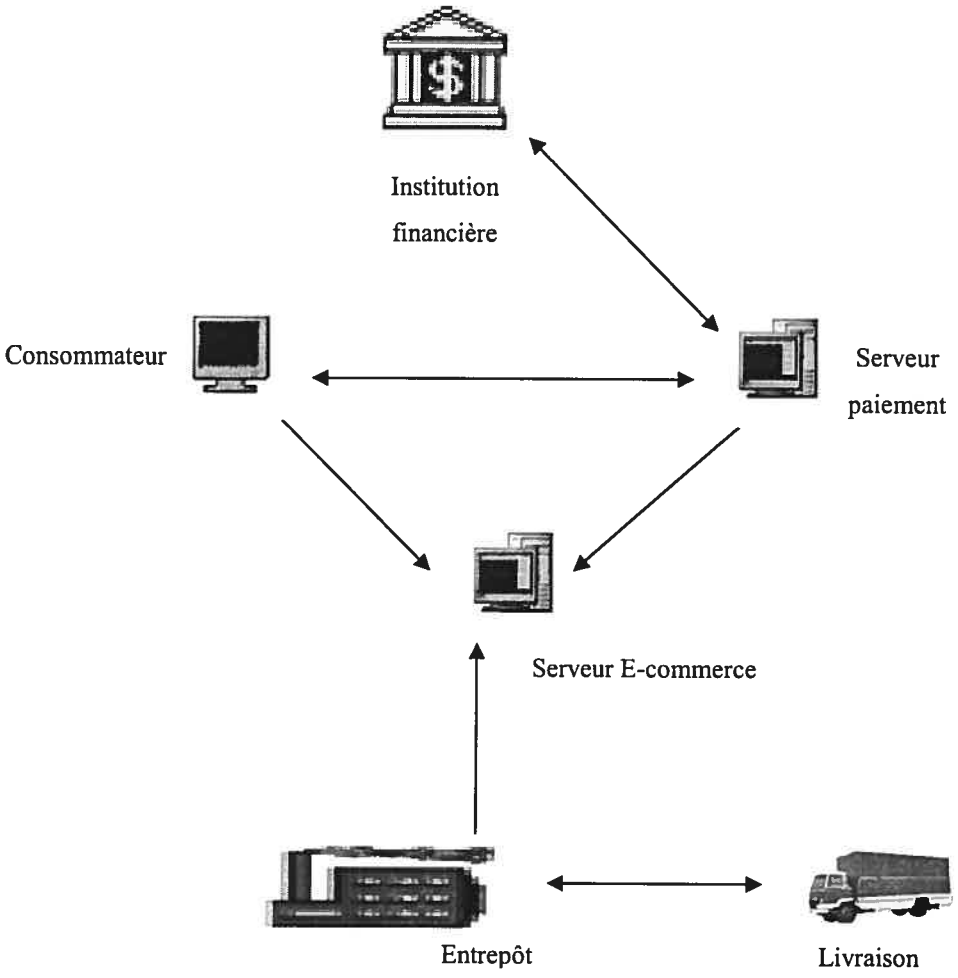

Figure 4.1 : Schéma global d'une plate forme de commerce électronique

En effet, cette illustration présente les diverses fonctions que doit intégrer une plate forme de commerce électronique. A commencer par la consultation sur le fureteur jusqu'à la livraison. Cependant, il est à signaler que le e-commerce possède quatre niveaux de complexité et est divisé en quatre catégories qui font la différence entre les multiples applications disponible sur le Web.

4.1.1 Les niveaux du commerce électronique

Les applications de commerce électronique sont passées par plusieurs étapes et ont connu une grande évolution. Cette progression a donné lieu à plusieurs niveaux de complexité. Pratiquement nous comptons quatre niveaux mais trois seulement sont en mesure d'être exploités.

1. **Le catalogue en ligne** : Page Web statique ou dynamique dont le rôle s'arrête à la description des produits à vendre. En effet, C'est un catalogue plus ou moins complet de produits à vendre. Il peut aller d'une simple page HTML par produit à une base de données interfacées avec le Web. Cette interface permettra au client potentiel de rechercher un produit par famille, par référence ou par tout autre critère. Dans les deux cas, la vente se terminera par un moyen conventionnel.

2. **La commande en ligne** : C'est un catalogue en ligne qui offre la possibilité d'envoi de commandes directement sur le Web. Ainsi quand le client consulte le catalogue, il peut à tout moment ajouter ou retirer des articles dans une sorte de caddie virtuel, sa sélection terminée un bon de commande s'affiche avec les articles choisis. Le client indique alors ses coordonnées afin que le commerçant valide sa commande et termine la procédure d'achat.

3. **Le paiement en clair** : Non seulement le client choisit son produit et le commande sur Internet, mais en plus, il le paye sur le réseau. Certes, le client peut effectuer des achats en payant directement via le réseau, cependant son identité n'est pas protégée. Le numéro de la carte de crédit se transmet en clair. Ce mode n'est plus utilisé.

4. **Le paiement sécurisé** : C'est une amélioration du paiement en clair dans la mesure ou les informations relatives au client sont cryptées avant leur acheminement par le réseau. En effet, Il garantit d'une manière ou d'une autre, que les informations sensibles sur le compte de l'acheteur ne circuleront jamais en clair sur le réseau. Soit l'échange est crypté entre le navigateur et le serveur Web. Soit on opte pour des solutions de paiement externes au Web qui dispose de son propre mécanisme de sécurité. Plusieurs protocoles de sécurité ont vu le jour tels que SSL (Secure Sockets Layer) et CSET (Chip Secure Electronique Transaction).

Dans chaque niveau cité plus haut, on peut distinguer quatre grandes catégories d'applications de commerce électronique qui feront l'objet du prochain paragraphe.

4.1.2 Les types de commerce électronique

La nature des acteurs qui rentrent en jeu lors des transactions de commerce électronique définit quatre grandes catégories.

1. **Commerce à Consommateur (B2C Business to consumer)** : Tous les commerces de détails qui ont un site transactionnel entrent dans cette catégorie. Le B2C est sans doute l'aspect le plus visible du commerce électronique. Le consommateur peut acheter directement sur Internet des biens et services. Ce type de commerce prend de plus en plus d'ampleur et le nombre de consommateurs grimpe de jour en jour.
2. **Consommateur à Consommateur (C2C Consumer to Consumer)** : Commerce électronique entre un consommateur et un autre consommateur. Le site pur de C2C serait le site du consommateur lui-même qui indique sur son site qu'il désire vendre un ou des produits et services sans en faire le commerce. Ces sites sont rares. Pour aider les transactions entre consommateurs, il existe des sites "Marchés aux puces virtuels" que l'on désigne aussi comme sites de petites annonces classées virtuelles. Pour la vente aux enchères en ligne, on les nomme alors *encanteurs*.
3. **Commerce à Administration (B2G Business to Governement)** : Désigne les transactions entre une entreprise et une administration. La transmission d'une déclaration de revenu vers un ministère en est un exemple. Il est surtout utilisé dans les appels d'offres mais reste en deçà des espérances. De plus en plus nous allons entendre parler de ce type de commerce.
4. **Commerce à Commerce (B2B Business to Business)** : C'est le commerce interentreprises. Grâce à Internet, le secteur du B2B est en train de vivre sa révolution. S'agissant du volume d'affaires qu'il représente, et même si les chiffres prévisionnels sont à manier avec précaution, les estimations à l'échelle mondiale, à l'horizon 2003, font état de 1 200 à 10 000 milliards de dollars. De plus c'est dans la pratique courante des échanges interentreprises que les vrais changements se produisent. Les entreprises voient aujourd'hui apparaître une nouvelle façon d'organiser leurs achats ou leurs ventes, en s'appuyant sur les places de marché.

Parmi les types de commerce électronique énumérés plus haut, deux catégories sont les plus répandues à savoir le B2B et le B2C. Une fois les différents types et catégories de commerce électronique présentés, nous nous intéresserons dans la section prochaine à la performance de ses applications. En particulier, nous aborderons l'environnement e-commerce afin de le comprendre et de pouvoir caractériser, au mieux, sa charge.

4.2. La performance des serveurs de commerce électronique

Le commerce électronique est entrain de mettre une lourde pression sur les organismes de technologie de l'information. L'objectif étant d'estimer le plus exactement possible les besoins du système.

Selon un rapport publié par Forrester, la croissance du commerce électronique dans le monde atteindra 3797,7 milliards de dollars US en 2003. Elle doublera pratiquement en une année pour atteindre le chiffre de 6789,9 milliards de dollars US. Les enjeux du e-commerce sont énormes et des sommes pharamineuses sont en jeu, ce qui implique une concurrence accrue. Le futur des entreprises oeuvrant dans le commerce électronique dépend de l'efficacité de leur infrastructure.

Les applications de e-commerce ont tendance à saturer les serveurs et les ressources réseau rapidement et entraînent, par conséquent, les entreprises dans une boucle infinie de mises à jour et de migrations. En fait, ces applications sont des applications distribuées complexes combinant des fonctionnalités de diverses composantes.

La conception, la configuration et la gestion de la performance d'une telle infrastructure requièrent une bonne compréhension de la performance de toutes les composantes ainsi qu'une bonne caractérisation de la charge de travail. L'évaluation de la performance et la planification de capacité peuvent se faire via quatre méthodes [9] et [24]:

• **La prédiction statistique :** elle consiste en la collecte des données de performance telles que le CPU, la mémoire, l'utilisation du disque et du réseau et la charge de travail. En se basant sur cet historique les tendances futures peuvent être prédites. Cette méthode est acceptable en terme d'exactitude, cependant il est normal de se trouver face à des variations inattendues. Ainsi, il serait plus judicieux de ne pas se baser uniquement sur cette méthode.

• **La simulation d'utilisateur :** elle repose sur la construction de modèle de charge pour le système de e-commerce. L'application de différentes charges simulant des utilisateurs permet de déterminer le comportement du système. Le défi essentiel est de mettre en place un modèle efficace. Certes, cette méthode donne une vue fiable sur la performance sous de multiples charges, mais elle peut être sans grande utilité lorsque le modèle est mal conçu.

• **Le profil :** ce n'est autre que l'utilisation des routines de benchmark standards déjà existantes. Le résultat permet une évaluation de la performance du e-commerce au niveau matériel et non au niveau application.

• **Les charges de travail synthétiques :** elles permettent le teste des application de commerce électronique ainsi que la simulation des activités des utilisateurs.

Toutefois, avant de pouvoir tester ou évaluer la performance de telles applications, une bonne compréhension de l'environnement e-commerce s'impose.

4.2.1 L'architecture e-commerce

L'architecture d'une organisation de commerce électronique dépend légèrement du type du e-commerce supporté. Cependant, le schéma ci-dessous présente une architecture générique pour un site de commerce électronique.

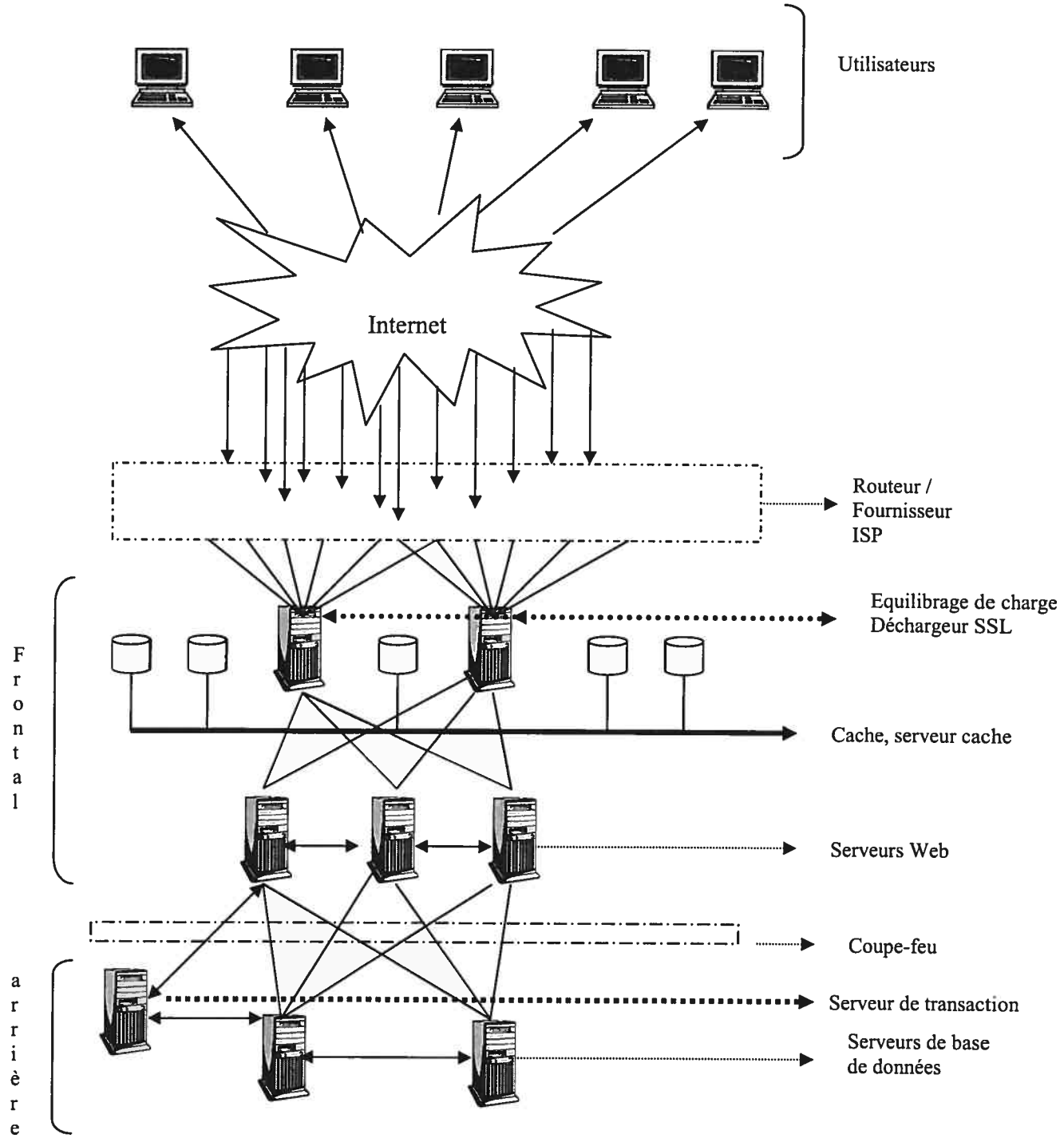

Figure 4.2 : architecture générique d'un site de commerce électronique

L'architecture présentée regroupe des composantes importantes dans la gestion des sites et d'applications de commerce électronique. En effet, nous pouvons distinguer :

- Les serveurs d'équilibrage de charge : ils utilisent des moteurs à base de règles pour correspondre un serveur Web particulier à un utilisateur. Ils décident donc, quel serveur prendra en charge quel utilisateur. Souvent l'utilisation de clusters de serveur Web s'avère avantageuse. Cette technique permet de grouper des serveurs indépendants pour répondre aux requêtes comme le ferai une seule unité ce qui améliore la performance du système.

- Les serveurs cache : les sites de e-commerce rencontrent quotidiennement des milliers de demandes pour la même information, au même moment et du même emplacement. les serveurs cache emmagasine et garde en mémoire les pages Web fréquemment appelées localement. Une conséquence directe est l'amendement du temps de réponse et par la suite le trafic Web.

- Les serveurs Web : ces serveurs abritent les applications de commerce électronique. En fait, ils traitent les pages Web en réponse aux requêtes des fureteurs.

- Le coupe-feu : il vérifie les requêtes avant leur acheminement en provenance ou vers les serveurs Web. Le coupe-feu est un souci majeur pour la sécurité et la performance du commerce électronique. Il doit assurer non seulement une bonne protection mais aussi un débit acceptable afin d'éviter une congestion au niveau du pare-feu et menacer la performance du système.

- Le serveur de transaction : la gestion des transactions commerciales d'un site de e-commerce se fait à l'aide d'un serveur de transaction. Ils sont responsables du maintien d'une performance élevée, de la disponibilité du service et de l'intégrité des données.

- Les serveurs de base de données : ils sont devenus une partie intégrante de l'infrastructure du commerce électronique. La majorité des données et informations des applications e-commerce résident dans ces serveurs. La disponibilité des données ainsi que l'aspect dynamique du contenu de la base sont des points principaux lors de la conception des serveurs de base de données.

Comme nous pouvons le constater l'infrastructure du commerce électronique peut être vue sous deux angles différents. En effet, elle est composée de deux types de serveurs à savoir les serveurs frontaux et les serveurs arrière. Nous détaillerons les composantes de chacune des parties dans la section suivante.

4.2.2 L'environnement e-commerce

Nous avons déjà présenté les divers types de commerce électronique. La configuration des serveurs ne diffère pas trop d'un type à l'autre surtout quand il s'agit du B2B et du B2C les deux types les plus utilisés. En réalité, quelque soit la nature du serveur de commerce électronique, deux parties essentielles le compose : le réseau frontal et arrière.

Typiquement, les serveurs frontaux comprennent le serveur Web, l'application serveur, le serveur d'équilibrage de charge et le déchargeur SSL (Secure Socket Layer). Les serveurs Web frontaux sont les seuls autorisés à accéder à la base de données arrière et éventuellement les services d'applications. Les serveurs d'application sont responsables des services de logique d'affaires. En outre, ils constituent la partie la plus chargée dans un environnement B2C. Cette charge est essentiellement due au fort trafic généré par les requêtes dynamiques et sécurisées reçues par le serveur. L'emplacement des serveurs d'application a des conséquences sur l'adaptabilité, la disponibilité et la sécurité de l'environnement. Pour la majorité des environnements B2B le serveur d'application est un serveur d'application dédié.

L'énorme charge que reçoivent les serveurs de e-commerce ainsi que les besoins de disponibilité nécessitent un réseau de serveurs frontaux. Ce qui améliore fondamentalement l'adaptabilité et la tolérance aux fautes des serveurs face aux rafales. Les équilibreurs de charge oeuvrent aussi dans ce sens. En fait, ils distribuent les requêtes utilisateurs à un groupe de serveurs qui apparaît de l'extérieur comme un seul serveur virtuel. De plus, il a pour rôle d'affecter le trafic utilisateur au serveur le plus disponible ou le plus apte à répondre. Ils se basent en général, sur des mécanismes sophistiqués pour rechercher le serveur avec le moins de connexions, le moins chargé ou ayant le meilleur temps de réponse.

SSL est un protocole d'authentification pour les utilisateurs développé par Netscape et se base sur le protocole de cryptage RSA. Presque tous les sites Web de commerce

orientés transactions nécessitant une carte de crédit ou des informations personnelles utilise SSL. Un déchargeur SSL décrypte toutes les requêtes http arrivant au serveur. Il est à signaler que le lien entre la partie frontale et arrière de l'environnement est totalement sécurisé. Ainsi les transactions sécurisées sont décodées au niveau frontal avant d'être acheminées vers les serveurs arrière.

Les dits serveurs arrière, essentiellement composés de serveurs de base de données et de pare-feu. Celui-ci empêchera les accès non autorisés par les clients et ainsi protégera les données sensibles. En outre, les pare-feu offrent un service de sécurité par le biais du contrôle de connexions.

Les serveurs de base de données résident dans l'arrière de l'environnement et hébergent aussi bien les données relatives aux transactions de commerce électronique que les informations sensibles du client. Celui-ci ne se connecte jamais directement à ces serveurs. Effectivement, les serveurs Web frontaux initient des connexions aux serveurs de base de données lorsqu'un client évoque certaines fonctionnalités tels que l'ouverture d'une session, la vérification de l'inventaire ou l'envoi d'une commande. La disponibilité des données est un point important à tenir en compte.

De plus, afin de mieux répondre aux exigences de performance des sites de commerce électronique une étude du trafic s'avère primordiale. Dans la section suivante nous essayerons de caractériser la charge de travail typique de e-commerce.

4.2.3 La charge de travail du commerce électronique

Nous avons déjà présenté, dans le chapitre trois, les aspects relatifs à la modélisation et la caractérisation de la charge de travail pour un trafic Web. La plupart des idées avancées restent valables pour le trafic e-commerce puisqu'il est bien un trafic Web avant tout. Cependant, il est très important de prendre en considération les particularités de la charge de travail des applications de commerce électronique. L'évaluation et l'analyse de la performance n'en seront que plus pertinentes et plus fiables.

Pour un serveur Web, les cliques successives d'une URL (Uniform Ressource Locator) peuvent être traités indépendamment et considérés comme des visites aléatoires au serveur. Quant au serveur de e-commerce, il garde trace des informations sur chaque client et ce en implémentant un gestionnaire de session. En outre, les consommateurs sur

un site transactionnel, suivent une séquence typique d'URLs le conduisant vers l'achat du produit.

Les serveurs de e-commerce sont plus complexes que les serveurs Web traditionnels. Les différences majeures entre la charge de travail Web et celle du commerce électronique sont [24] :

1. La présence d'un niveau élevé d'activité de traitement en ligne de transactions. Elle est due à la croissance des transactions de base de données engendrées par chaque requête utilisateur. Pour des raisons de sécurité, toutes les données sont présentes dans les serveurs arrière protégés par un coupe-feu sécurisé.

2. L'activité de la base de données est caractérisée par le fait qu'une grande proportion des requêtes arrive via un mode sécurisé. La sécurité est moins pesante pour le trafic B2C que pour le trafic B2B. Ce qui est compréhensible vu les contraintes imposées pour les transactions commerce à commerce. L'accroissement de la quantité d'échange sécurisé implique des traitements lourds au niveau du serveur frontal. Les déchargeurs SSL réduisent la charge du système. Le temps de traitement pour ce processus s'ajoute au temps de réponse perceptible par l'utilisateur final. Par ailleurs, le temps pris par les transactions commerciales proprement dites augmente l'inconstance du temps de réponse.

3. Le taux de requêtes dynamiques, qui demandent un certain nombre de traitement, est très élevé. En fait, dans les environnements de commerce électronique toutes les transactions sont traitées comme des requêtes dynamiques.

Ces différences mettent plus de pression et plus de contraintes pour atteindre un bon niveau de performance. En effet, il s'avère que les sites de e-commerce connaissent plus de rafales que les serveurs Web normaux. La cause principale est accordée à la nature de la charge de travail du commerce électronique. Hormis les propriétés précédemment établies pour le trafic Web telles que l'auto-similarité et la non-stationnarité, le trafic commercial transactionnel a ses propres particularités.

Plusieurs études portant sur le trafic Web et LAN ont attribué l'autosimilarité à l'agrégation de processus à forte dépendance avec des périodes d'activité et de non activité. Dans le monde e-commerce, les temps de réponses suivent, naturellement, des

distributions à queue lourde. Ce constat est vérifié malgré le fait que la taille des fichiers de réponse et de requête demeurent presque constante. En outre, ce constat contredit les observations démontrées dans un environnement Web ou la distribution à queue lourde de la taille des fichiers est considérée à l'origine de la nature de la distribution du temps de réponse.

Pour une architecture de commerce électronique, la taille de fichier ne suit pas une distribution à queue lourde. Ce type de distribution trouve sa justification dans le Web puisque les fichiers transférés incluent images, animations et vidéo. L'utilisation de tels fichiers dans le e-commerce reste limitée pour un pur souci d'amélioration de la performance.

Cependant, le temps de réponse suit une distribution à queue lourde aussi bien pour les sites B2B que pour les sites B2C. Ce qui implique que le temps de réponse perçu par l'utilisateur final peut augmenter par des ordres de grandeurs importants sous certaines conditions de charge. Toutefois, il est impératif, vu l'aspect critique des applications e-commerce, de garder le temps de réponse en dessous des limites normales même avec des charges élevées.

Puisque la taille des fichiers, dans un environnement de e-commerce, ne suit pas une distribution à queue lourde, nous pouvons supposer sans risque que le temps de transfert n'affecte pas le temps de réponse. Ainsi, le temps de traitement est l'acteur principal qui affecte le temps de réponse. En effet, le traitement des transactions en ligne, les vérifications de sécurité et les traitements des serveurs de base de données consomment énormément de temps.

Par conséquent, les variations du temps de réponse, dans un contexte de commerce électronique, sont attribuées aux variations du temps de réponse observées dans les serveurs arrières. Ce qui constitue la grande particularité des applications du e-commerce.

Nous essayerons d'appliquer notre connaissance des ces faits au système étudié afin de fournir une image claire et fidèle de sa performance. Nous présenterons dans la section suivante la plate forme testée.

4.3. Etude de cas : le serveur GNP

Le progrès rapide et important des technologies de l'information a énormément changé la structure de l'économie. Ceci se voit clairement à travers le commerce électronique et la manière dont les négociations se déroulent tout au long d'une enchère. Certes, la création d'un marché électronique efficace suscite plusieurs défis. L'infrastructure de négociation doit être capable de supporter une assez large gamme de règles et d'algorithmes de négociations. C'est dans cette perspective que GEE (Generic Experimentation Engine) a été conçue et créée pour donner naissance ensuite à une plate-forme évoluée de négociation GNP (Generic Negociation Platform). Celle-ci a été mise en place pour s'insérer dans une place de marché virtuelle.

4.3.1 Les places de marchés virtuelles

En l'absence d'un marché défini et efficace, la recherche d'occasions d'affaires pour combler un besoin pressant ou pour écouler un stock est potentiellement coûteuse, en termes de temps et d'argent. D'où l'intérêt pour les entreprises d'avoir accès à une place de marché où elles peuvent en tout temps vendre ou acheter.

Le concept de place de marché consiste en la création d'un nouveau lieu et d'une nouvelle façon de faire du négoce, entre acheteurs et vendeurs. Construire une place de marché a pour conséquence l'optimisation de l'acte d'achat et celui de vente. Par extension, la place de marché devient un lieu d'échange d'informations, de services et de biens qui enrichit la relation client/fournisseur. Généralement, la place de marché est un espace virtuel de commerce B2B. Son objectif est d'agréger offres et demandes, client et vendeurs. La place de marché est souvent dédiée à un secteur d'activité particulier : pétrole, métallurgie, automobile...etc.

La place de marché offre également l'avantage de diversifier les modes de transactions commerciales. Il n'est pas rare que la place de marché offre plusieurs forme d'échange : vente classique (soumises aux conditions générales de ventes et d'achats), adjudication ou vente par lots. Les entreprises dotées d'un important carnet d'adresses décident de créer leur propre place de marché et invitent leurs fournisseurs à y participer, lesquels invitent leurs propres fournisseurs ou leurs acheteurs.

Qu'elle soit traditionnelle ou virtuelle, toute place de marché est constituée de trois grandes catégories d'acteurs : les acheteurs, les vendeurs et les opérateurs. Nous appellerons opérateur de marché une personne ou une entreprise qui gère l'intendance de la place de marché. Ainsi, l'opérateur d'une place de marché virtuelle est l'administrateur du système informatique qui gère une variété de tâches relatives à l'organisation du marché : inscription des clients, contrôle d'accès et organisation.

La figure 4.3 illustre l'architecture d'une place de marché électronique. Il est bien clair que les différents acteurs communiquent via plusieurs technologies et standards. Ce schéma met aussi en exergue les points essentiels qu'une place de marché doit traiter à savoir la sécurité, les règles économique du marché et les outils nécessaires pour une navigation agréable pour le client [21].

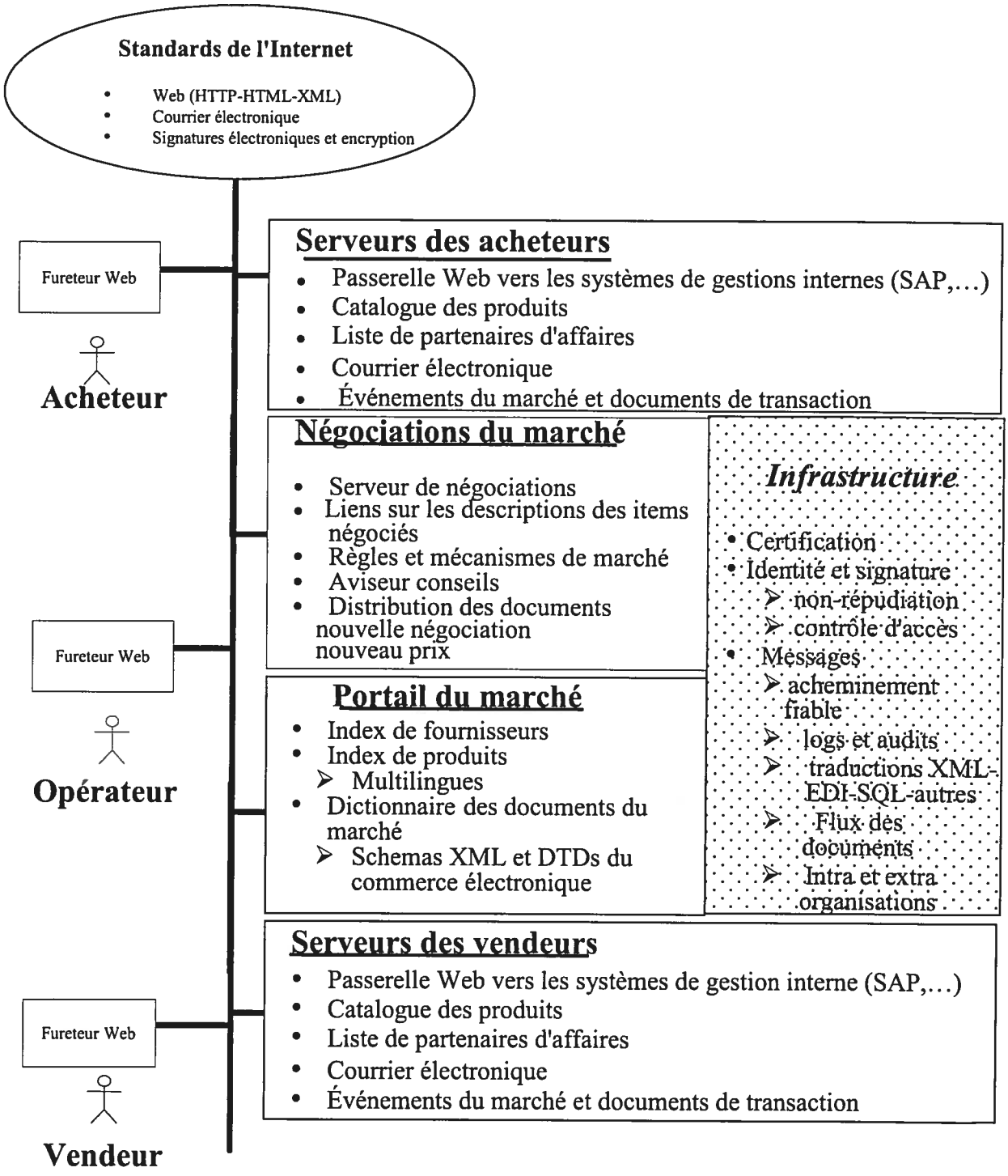

Figure 4.3 : architecture d'une place de marché électronique

Il faut bien comprendre qu'une place de marché est avant tout une plate-forme technologique qui s'engage vis-à-vis des acheteurs et des fournisseurs. Typiquement, c'est un serveur informatique auquel peuvent accéder, via un réseau (Internet), acheteurs et vendeurs. Il faut noter que les entreprises qui font du e-commerce sont mieux préparées à l'intégration des places de marché, parce que leur système d'information a été conçu pour supporter des échanges avec l'extérieur.

Par ailleurs, la conception d'un mécanisme de marché électronique peut se faire à plusieurs niveaux de complexité technologique. Pratiquement, plus les implications électroniques sont fortes au sein du processus de négociation, plus son niveau de complexité est élevé. Ces différents niveaux de complexité vont du simple affichage électronique aux marchés véritablement optimisés, en passant par les soumissions informatiques et les enchères électroniques. Ces niveaux de complexité peuvent être interprétés comme quatre « couches de services », chaque couche étant une évolution de la couche précédente. Le choix du niveau de complexité pour une industrie donnée dépend de l'analyse que l'on fait des besoins de l'entreprise ou de l'industrie dans son ensemble.

Le système que nous avons étudié constitue une place de marché virtuelle traitant plusieurs types de négociations. Ainsi, GNP constitue une plate-forme de commerce électronique spécialement dédiée aux échanges entreprise à entreprise.

4.3.2 L'architecture de GNP

L'architecture de GNP est basée sur un système d'applications Web distribuées utilisant les servlets, les pages JSP et les scripts écrit en JPython. Hormis les technologies supportées par GNP, celui-ci doit faire face à d'autres problèmes relatifs aux exigences d'une plate-forme électronique de négociation. A savoir la sécurité, l'évolution, la fiabilité et la persistance. Pour ce faire GNP va être exécuté en tant que composante réutilisable dans un serveur EJB (Entreprise JavaBeans). Ce dernier interagit avec une base de données relationnelle.

Les EJB sont des composants logiciels réutilisables, c'est-à-dire des bouts de logiciels indépendants pouvant être assemblés pour construire une application à base d'objets métiers distribués dans un environnement Java. L'EJB typique est constitué de méthodes

qui encapsulent la logique liée à un métier. Ils ne possèdent pas d'interface graphique et sont destinés à être placés sur un serveur.

Un client n'accède pas directement à un EJB. En fait, les spécifications EJB décrivent le conteneur : fournisseur de services aux Entreprise JavaBeans. Les conteneurs EJB sont des interfaces entre les EJB et le monde extérieur. Ils invoquent leurs méthodes, gèrent les aspects techniques tels que la sécurité, les transactions, la concurrence et tout autre service. Un conteneur est responsable de la mise à disposition des classes EJB aux clients [21] et [1].

La plate-forme générique de négociation a une architecture multi-tiers avec 5 tiers :

- **Les participants** : acheteurs, vendeurs et régisseurs de marché rejoignent les serveurs GNP avec leurs fureteurs, soient par des pages HTML, soient par des composants Java (des Applets) activés à l'intérieur du fureteur ou directement dans des agents Java ou JPython.
- **Le portail HTTP de contenu** : Utilisant des Servlets Java et des pages JSP (Java Server Pages). GNP utilise le serveur http BEA-Weblogic.
- **Le serveur de requêtes GNP** : Utilisant les Enterprises JavaBeans (EJB) pour les sessions et les entités.
- **Le serveur des encanteurs** : Serveur des EJB activé par des messages asynchrones « Java Message Service » (JMS). GNP utilise le serveur BEA-Weblogic.
- **Le serveur de données SQL** : GNP est compatible avec les deux systèmes de gestion de base de données PostgreSQL et Oracle8i. La version actuelle GNP utilise PostgresSQL.

[1] http://www.cirano.qc.ca/eree

L'architecture globale du système est donnée dans le schéma ci-dessous :

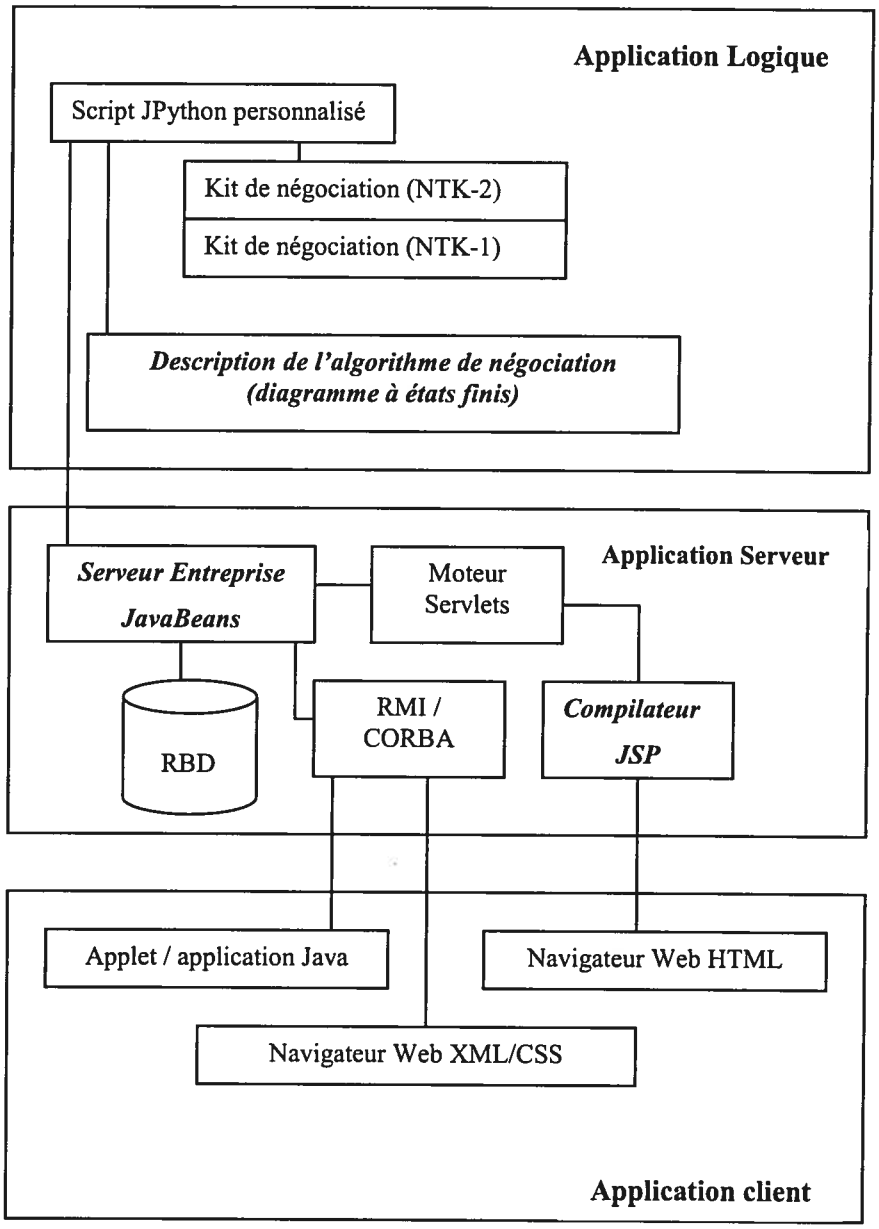

Figure 4.4 : architecture de la plate-forme GNP

Du point de vue GNP une négociation est vue comme un document. Dans le but de faciliter la manipulation des documents internes, GNP est muni d'une d'API

(Application Programming Interface) à deux niveaux appelée Negotiation ToolKit (NTK).

Le premier niveau (NTK-1) se charge des fonctions de gestion de stockage (chargement, recherche et enregistrement des informations concernant la négociation). Tandis que le niveau deux (NTK-2) offre les fonctions et les objets fréquemment utilisés par le concepteur de la négociation tels que l'objet mise et l'objet négociation.

La nature du trafic qui traverse la plate forme GNP nécessite un ensemble de mécanismes et de propriétés que le système doit vérifier. Nous allons, dans le paragraphe suivant donner les spécificités de GNP.

4.3.3 La plate-forme GNP

Comprendre le fonctionnement et les principes de GNP est une étape primordiale pour aborder efficacement l'analyse de sa performance. La plate-forme de négociation générique GNP est conçue pour servir de base aux nouvelles règles économiques nécessaire aux systèmes de négociation industrie à industrie. Elle est construite autour des concepts suivants [21] :

• Les enchères directes, renversées et des deux côtés : GNP prend en charge les trois catégories d'enchères. Les enchères directes constituent la forme la plus connue d'une négociation, là ou le gagnant est celui qui mise le plus. Tandis que les enchères renversées ont pour but de chercher celui qui offre le plus bas prix. Ce type d'enchères est surtout utilisé pour les appels d'offres. Quant à l'enchère des deux côtés elle fait participer des acteurs se trouvant des deux faces de la négociation. En effet, les participants, vendeurs et acheteurs, soumettent leur mise en même temps. Ensuite un mécanisme de jumelage entre les offres et les demandes est exécuté.

• Des négociations conduites avec des phases et des rondes : lorsque les règles économiques changent durant une négociation, celle-ci s'effectue en plusieurs phases. Durant une phase les règles sont invariantes. Une phase est composée d'un certain nombre de rondes. Pratiquement, une ronde est la succession de trois événements : l'envoi de l'annonce par l'encanteur, une ou plusieurs mises des participants et finalement l'évaluation des mises par l'encanteur.

• Plusieurs types d'enchères possibles : en particulier, les enchères anglaises, hollande, les enchères cadencées et les enchères synchronisées. Le facteur temps pour chacune des enchères peut être ajusté en fonction de la demande du marché.

• Un marché avec des notes, score et ordonnancement complexe : il arrive que la qualité des produits et des entreprises engagés dans une négociation varie. Dans ce cas, le système GNP intègre le facteur qualité dans le processus d'évaluation des mises. De plus, GNP utilise une fonction de score paramétrée pour ordonnancer les ordres.

• Les enchères multi-unitaires : elles permettent la négociation du prix et de la quantité de plusieurs unités d'un même produit. Ce type d'enchères à la particularité de fragmenter les unités en divers niveaux de prix et de quantité.

• Les enchères multi-produits et enchères synchronisée : la négociation simultanée de plusieurs produits reliés, en général complémentaires, s'inscrit dans les enchères multi-produits. L'annonceur accepte des mises partielles et espère qu'une combinaison de ces mises couvrira sa demande. La synchronisation de la négociation est assurée par GNP.

• Les enchères multi-attributs : ce n'est autre qu'une variante des enchères précédentes. La seule différence est l'ajout d'attributs constituant un pacte à confirmer entre participants tel que le respect de la date de livraison.

• Les ordres simples et les ordres complexes : Les ordres et les mises avec seulement un prix et une quantité sont la base de toute négociation. Des ordres plus complexes sont souvent requis dans une négociation d'affaire. En effet, un ordre avec le prix courant et le prix de réserve (prix limite), un ordre avec un vecteur de prix-quantité exprimant une fonction d'offre ou de demande ou un ordre sur plusieurs objets combinés dans une négociation combinée, sont souvent très utiles.

Par ailleurs, La plate forme de négociation générique (GNP) est basée sur la manipulation des documents XML (eXtensible Markup Language) à l'aide de scripts JPython. La plate forme GNP est dite orientée documents du moment ou chaque négociation est représentée par un document et se traduit par la génération et la manipulation de plusieurs documents ou fichiers XML. En effet, la négociation met

différents documents XML en jeu : *Announce*, *Rules*, *ProductReference*, *Negociation*, *Quote*, *Order*, *Response* et *Adjudication.*

Le système GNP est générique. Il permet une implantation rapide et flexible de différents types de négociations. Pour cet objectif, les aspects informatiques et économiques ont été séparés de manière à rendre autonome l'implantation de nouveaux algorithmes économiques. Cette séparation est illustrée dans la figure qui suit.

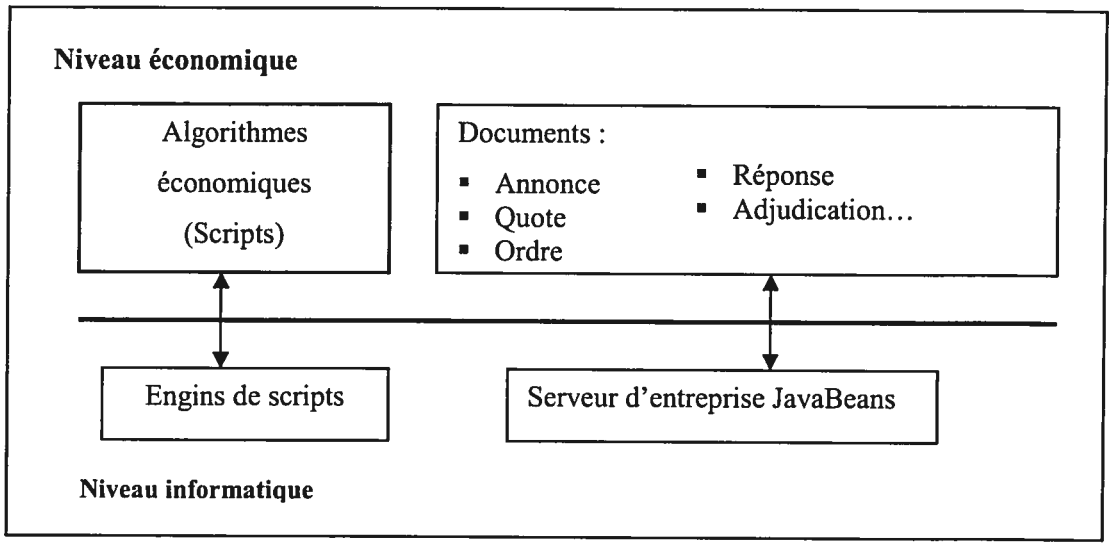

Figure 4.5: Séparation du niveau informatique et économique dans GNP

D'autre part, étant donné que GNP est désigné spécialement pour traiter automatiquement des négociations, il est muni des caractéristiques suivantes :

- **Fiabilité transactionnelle** : Après la réception par un participant de son accusé de réception, le système garanti que le serveur traitera éventuellement sa commande. Cependant cela peut prendre un certain temps si le système est surchargé ou en panne.
- **La performance** : Un temps réponse court, une latence réduite et un débit raisonnable lors de la réception des ordres des participants sont essentiels.
- **La sécurité** : Elle doit couvrir les quatre aspects classiques à savoir l'identification, la confidentialité, le contrôle d'accès et l'intégrité.

- **La rapidité** : La plate forme doit être capable de réagir et répondre rapidement aux différentes situations.
- **La traçabilité** : GNP doit pouvoir justifier ses actions, à tout moment, aux utilisateurs. Ce qui lui permettra de gagner leur confiance.

Ainsi, nous avons présenté l'environnement qui nous a servi pour faire nos tests. Nous avons en particulier exploré les enchères ouvertes de GNP.

4.4. Conclusion

Dans ce chapitre il a été question de faire le tour de la plate-forme GNP. Celle-ci étant considérée comme une partie de la place de marché virtuelle, il a fallu, avant tout, introduire les applications de e-commerce et la particularité de leur trafic. Puis nous avons donné un aperçu sur les places de marché virtuelles. Ensuite, nous avons exposé l'architecture de GNP basée sur les EJB et les documents XML. Puis toute en insistant sur la séparation entre le niveau économique et informatique qui constitue l'un des points fort de l'architecture de GNP, nous avons présenté ses différentes caractéristiques. Ainsi, les négociations sur GNP ne sont autres qu'une manipulation de documents. Notons que les plus importants sont *Announce*, *Order* et *Rules*. D'autre part, GNP s'annonce très optimiste quant à la diversité et la complexité des négociations qu'il prend en charge. Cependant, le problème de sa performance se pose.

Malgré la rareté des articles publiés traitant les approches utilisées pour tester la performance des systèmes de commerce électronique, ce sujet reste extrêmement important pour la majorité des projets de l'industrie. Assez souvent, les problèmes soulevés suite à la commercialisation d'un serveur sont liés plutôt à une dégradation de sa performance qu'à un mal fonctionnement ou une défaillance de ce dernier.

Donc, après avoir bien compris le fonctionnement et la structure de GNP, nous allons dans le chapitre suivant présenter le travail réalisé. Nous définirons nos objectifs, présenteront le modèle utilisé, construirons les scénarios de tests puis nous exposerons et analyserons les résultats obtenus.

Chapitre 5

MODÉLISATION ET REALISATION

Nous avons jusqu'à présent introduit les éléments nécessaires qui nous permettrons de concevoir les bonnes séquences de tests. En effet, l'étude de la plate forme GNP, la charge de travail, les principes des tests de performance ainsi que les application de commerce électronique nous ont beaucoup aidé dans l'étape de la mise en œuvre des scénarios de test.

Cependant, avant de réaliser ces tests et récolter les résultats, nous avons passé par deux étapes essentielles. Tout d'abord il a fallu définir clairement nos objectifs qui ont guidé notre étude. La seconde étape concernait la modélisation de la charge de travail à considérer pour nos tests.

Dans ce chapitre nous allons exposer la démarche suivie expliquant ainsi les métriques, les objectifs et les scénarios considérés. Ces derniers nous ont permit de mettre en place un Benchmark selon une charge de travail bien précise. Ensuite nous présenterons les résultats obtenus que nous essayerons d'analyser et d'en tirer des conclusions quant à la performance du serveur sous étude à savoir le GNP.

[1] http://www.cirano.qc.ca/eree

5.1. Description de l'approche

Le problème de test de performance est tellement vaste et compliqué qu'il faut orienter l'étude vers un sens bien précis. En effet, il peut être abordé sous différents angles selon les objectifs.

Par ailleurs, comme nous l'avons mentionné dans les chapitres précédents, l'évaluation ou l'étude des performances d'un système suit des étapes systématiques. Certes, les techniques de test de performance changent selon le système mais une approche commune reste valable pour toute étude d'évaluation de performance.

Notre étude n'a pas échappé à la règle. En effet, durant notre évaluation de GNP nous avons suivi l'approche décrite dans le deuxième chapitre. Le schéma ci bas illustre cette démarche.

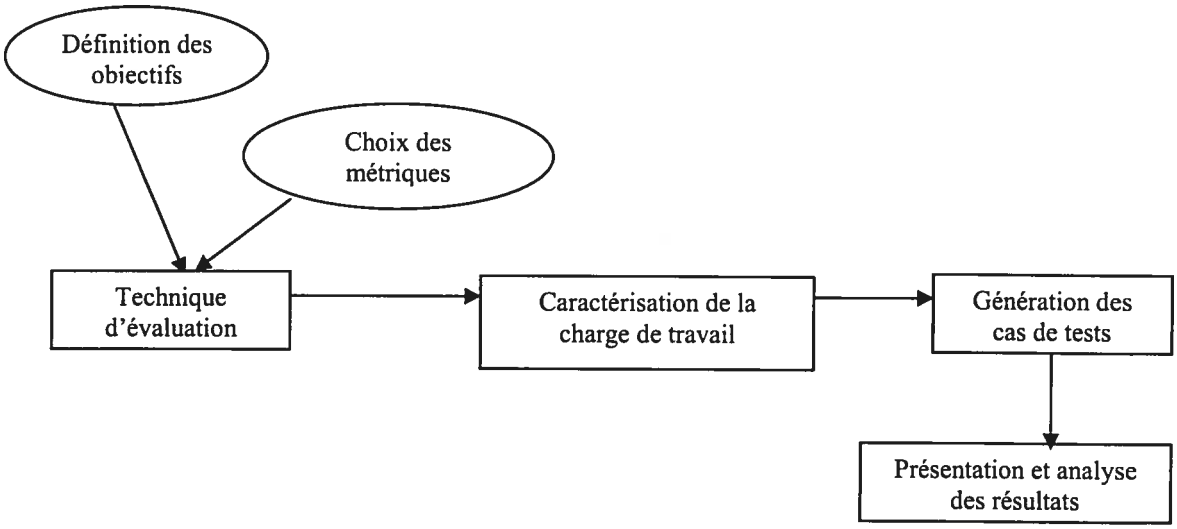

Figure 5.1 : approche suivi pour l'évaluation de la performance de GNP

Nous présenterons dans la suite les deux étapes les plus importantes à savoir la définition des objectifs et la modélisation de charge de travail.

5.1.1 Les objectifs de l'étude

La performance peut être abordée de plusieurs façons. Il pourrait s'agir de la prédiction de la performance, de la planification de capacité, de l'évaluation de certaines métriques d'un système existant ou bien de l'étude de la performance d'un système en phase de conception.

Le système que nous avons étudié existe bien mais il est toujours sous tests et connais plusieurs modifications et améliorations pour le rendre de plus en plus performant. C'est dans ce cadre que s'inscrit notre étude.

Des études précédentes ont essayé d'étudier la performance du serveur GNP [2]. Les tests exécutés étaient statiques ce qui ne reflétait guère la réalité. La charge de travail naturelle que subit le serveur est, en effet, loin d'être représentée par des mesures statiques.

Notre principal objectif sera de mettre en place un modèle de charge de travail représentant assez fidèlement le trafic qui traverse le serveur. En l'absence d'historique pour le serveur, notre caractérisation se basera sur des études similaires qui ont pris pour cible les serveurs et les charges de travail de commerce électronique. Nous avons opté pour une simulation basée sur des modèles analytiques pour mettre en œuvre notre modèle et exécuté nos tests. Ainsi, l'aspect dynamique de ces derniers les rendra plus réalistes.

En outre, nous utiliserons le modèle de charge de travail établi afin de mesurer le temps de réponse du serveur et étudier l'influence du comportement de l'utilisateur sur cette métrique. En fait, concernant les systèmes de commerce électronique, l'utilisateur joue un rôle important puisque son interactivité avec le système est très déterminante. Sans omettre que le succès d'une plate forme de commerce électronique repose largement sur la satisfaction de ses clients.

Dans ce contexte là, notre étude vient apporter sa contribution dans l'évaluation globale de la performance de la plate-forme générique de négociation GNP.

5.1.2 La modélisation du trafic

Notre choix de modélisation s'est porté vers la simulation utilisant les modèles analytiques. Effectivement, nous avons conçu un certain nombre de scripts qui simulent les requêtes utilisateurs et génèrent une charge pour le serveur. Par suite, notre modèle s'appuie sur les lois et distributions statistiques pour reproduire les processus stochastiques du trafic.

Le trafic Web en général et des systèmes de commerce électronique présente deux caractéristiques principales à savoir l'autosimilarité et la non-stationnarité. Pour générer un trafic proche de la charge réel et donc présentant ces deux particularités, nous avons utilisé des distributions statistiques dites à queue lourde. En outre, pour assurer autosimilarité et comme décrit dans la littérature, il existe différentes méthodes, entres autres, le bruit gaussien fractionnel et l'utilisation de la théorie des files d'attentes ($M/G/\infty$). Dans notre modèle, il a été question de la méthode de superposition de processus active/inactive (ON/OFF) avec des distributions à queue lourde.

a) Préliminaires

La prise en compte du temps de réflexion de l'utilisateur ou « think time » constitue le processus ON/OFF considéré. En effet, il est caractérisé par ses deux états actif lorsque l'utilisateur envoi des requêtes et inactif lorsque ce dernier est en mode passif. La définition du « think time » est présentée dans la figure 5.2 qui illustre aussi la différence entre le « think time », le temps de réaction du serveur et le temps de réponse.

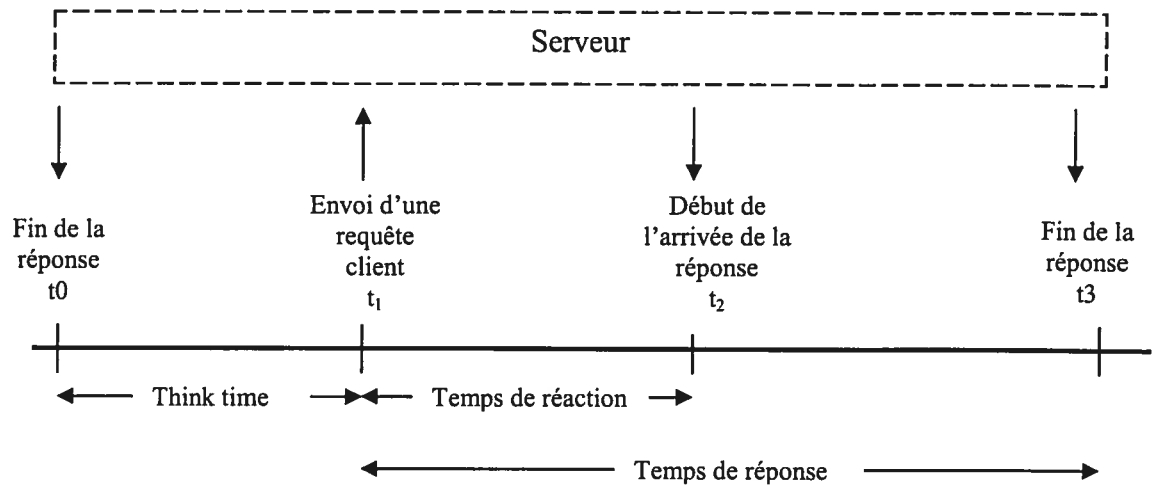

Figure 5.2 : Différence en « think time », temps de réaction et temps de réponse

D'autre part, le processus doit suivre une distribution dite *à queue lourde* (heavy tailed distribution). La catégorie de distributions utilisée pour modéliser les caractéristiques d'un processus extrêmement variable est la catégorie des distributions *à queue lourde* ou *à loi de puissance*. Mathématiquement, une fonction de répartition F(x) d'une variable aléatoire X est *à queue lourde* si : $\mathbf{1 - F(x) = \overline{F}(x) \sim cx^{-\alpha}}$ avec **c** une constante et α le paramètre de la distribution. A(x) ~ B(x) signifie que la limite quand $x \to \infty$ de $\frac{A(x)}{B(x)}$ est égale à 1.

Cela signifie qu'indépendamment de la distribution pour les petites valeurs de la variable aléatoire, si la forme asymptotique de la distribution est hyperbolique alors elle est à queue lourde. En pratique, les variables aléatoires qui suivent ce type de distributions, ont la particularité d'avoir plusieurs observations de petites valeurs et seulement quelques observations à grandes valeurs. Toutefois, malgré que la majorité des valeurs soit faibles, la contribution des grandes valeurs dans la moyenne ou la variance est très importante.

b) Modélisation

Nous avons basé notre étude sur les modèles analytiques. Ils sont souvent exprimés sous forme d'équations mathématiques. Notons que certaines hypothèses s'avèrent indispensables pour faire face à la complexité du monde réel. Plusieurs processus

stochastiques étaient à modéliser : l'arrivée des utilisateurs dans une négociation, le choix de la négociation, le temps de réflexion ou « think time » et le choix de la requête à exécuter.

Les deux plus importants processus étaient le « think time » qui reflète le comportement de l'utilisateur et la connexion au serveur ou le temps d'inter arrivée. Celui-ci a été modélisé par une loi exponentielle. En fait, pour notre étude nous avons supposé que plus on approche de la fin de la négociation plus le nombre d'utilisateurs augmente. Ce qui parait logique puisque c'est la fin d'une négociation qui importe le plus. En outre, le modèle exponentiel est utilisé pour représenter le temps passé avant l'arrivé d'un événement. Un événement peut être une connexion au serveur.

La fonction de densité de la loi exponentielle est donnée par :

$$\begin{cases} f(x) = \lambda e^{-\lambda x} & \textbf{pour} \quad x \geq 0 \\ f(x) = 0 & \textbf{sinon} \end{cases}$$

Le temps moyen entre deux arrivées est égal à $1/\lambda$. La moyenne est obtenu par le calcule de l'espérance $E(x)$ de la loi et ce comme suit :

$$E(x) = \int_{-\infty}^{+\infty} x f(x) dx$$

Pour notre simulation, nous avons pris comme moyenne 20 secondes, ce qui veut dire qu'en moyenne les utilisateurs se connectent toutes les vingts secondes. Afin de simuler la loi nous avons considéré l'inverse de la fonction de répartition et un générateur de nombre aléatoire entre 0 et 1. La fonction de répartition $F(x)$ se calcule par intégration de la fonction de densité.

$$F(x) = \int_{-\infty}^{x} f(u) du$$

Ainsi $\mathbf{F(x) = 1 - e^{-\lambda x}}$ et donc $\mathbf{F^{-1}(x) = -} \frac{1}{\lambda} \mathbf{\ln u}$

Tel que : $\mathbf{F^{-1}}$ est la fonction inverse de F, **u** est une variable aléatoire uniforme dans l'intervalle [0,1] et **ln** est le logarithme népérien.

D'autre part, pour la modélisation du « think time » nous avons opté pour une distribution à queue lourde afin d'assurer autosimilarité de la charge de test. Certaines études [17], [11], [22] et [16] ont utilisé la distribution de Pareto pour modéliser le temps de réflexion des usagers. Cette distribution met l'accent sur le fait que les événements à faible probabilité influent beaucoup plus que les événements à grande probabilité d'occurrence. Nous croyons profondément que des « think time » très petits, même peu probable, affectent considérablement le temps de réponse et ainsi la performance du serveur. La loi de Pareto est donnée par la fonction de densité suivante :

$$\begin{cases} \mathbf{f(x) = \alpha} \dfrac{k^{\alpha}}{x^{\alpha+1}} & \textbf{pour } x > k \\ \mathbf{f(x) = 0} & \textbf{sinon} \end{cases}$$

Avec **k** est la valeur minimale que peut prendre **x** et α est le paramètre de la loi qui mesure le biais.

Comme nous pouvons le remarquer la loi est bornée seulement à gauche. Etant donné que la convergence de la loi de Pareto est difficile dans ce cas, nous avons utilisé le « Pareto borné » qui rajoute une borne supérieure $\mathbf{X_{max}}$. Ainsi, la fonction de distribution devient :

$$\begin{cases} \mathbf{f(x) =} \dfrac{\alpha X_{\max}{}^{\alpha} k^{\alpha}}{x^{\alpha+1}(X_{\max}{}^{\alpha} - k^{\alpha})} & \textbf{pour } k < x < \mathbf{X_{max}} \\ \mathbf{f(x) = 0} & \textbf{sinon} \end{cases}$$

Par un calcule d'intégrale nous trouvons l'espérance de la loi **E(x),** la fonction de répartition **F(x)** et son inverse $\mathbf{F^{-1}(x)}$ servant pour des fins de simulation, comme suit :

- **La moyenne :**

$$\mathbf{E(x)} = \frac{\alpha X_{\max}\left[1-\left(\frac{X_{\max}}{k}\right)^{\alpha-1}\right]}{(\alpha-1)\left[1-\left(\frac{X_{\max}}{k}\right)^{\alpha}\right]}$$

- **La fonction de répartition :**

$$\mathbf{F(x)} = \frac{X_{\max} - \frac{X_{\max}{}^{\alpha} \times k^{\alpha}}{x^{\alpha}}}{X_{\max}{}^{\alpha} - k^{\alpha}}$$

- **La fonction inverse de la fonction de répartition :**

$$\mathbf{F^{-1}(x)} = \frac{k}{\left[1-x\left(1-\left(\frac{k}{X_{\max}}\right)^{\alpha}\right)\right]^{\frac{1}{\alpha}}}$$

Pour simuler la loi il suffit de générer une variable aléatoire uniforme entre 0 et 1 puis calculer son image à l'aide de la fonction inverse de la fonction de répartition.

Ces deux distributions représentent le cœur de notre modèle, nous avons utilisé une loi uniforme avec des probabilités de 0,5 pour le choix des négociations. En fait, lorsqu'un utilisateur se connecte sur le serveur, il a le choix entre deux négociations. En outre, chaque utilisateur envoi un certain nombre de requêtes. En effet, le serveur GNP offre la possibilité d'envoyer plusieurs types de requêtes.

Ainsi, notre modèle permet de prendre en considération plusieurs processus stochastiques qui interviennent dans la charge réelle. Il tient compte également des particularités du trafic commerce électronique.

5.2. Scénarios et tests

La plate-forme générique de négociation nous a servi d'outil d'expérimentation. En effet, nous avons employé notre modèle de charge pour simuler le comportement des utilisateurs de la plate-forme sur une enchère ouverte. Des enchères sont *ouvertes* quand les mises envoyées par les négociateurs sont connues de tous les autres participants qui peuvent réagir en conséquence. Elles sont en fait, la version informatique des enchères classiques dites à la criée.

Dans cette section nous présenterons l'objectif de nos tests qui repose sur l'importance du temps de réponse et du comportement de l'utilisateur vis-à-vis d'une plate-forme de commerce électronique. Puis nous décrirons l'environnement ou se sont déroulé nos expériences. Ensuite, il s'agira d'expliciter les différents scénarios considérés et enfin nous donnerons l'architecture des tests exécutés. La section suivante sera consacrée à la présentation des résultats et leur analyse.

5.2.1 L'objectif des tests

Nous avons orienté nos tests vers un objectif bien précis. L'étude du temps de réponse du serveur tenant compte du comportement des clients. Des résultats concernant le temps de réponse du serveur ont été récolté et analysé.

Le temps de réponse est l'une des plus importantes métriques de performance. Spécialement dans la conception et l'analyse de tout serveur ou système. Un trafic très dense implique une saturation des files d'attentes du serveur. Par conséquent, des temps de réponse énorme sont observés.

Une plate forme de e-commerce subit une grande masse de trafic et doit supporter un grand nombre d'utilisateurs. Ceux-ci jouent un rôle déterminant dans la réussite d'un marché électronique. Par conséquent leurs exigences doivent être soigneusement prises en considération. En réalité, le serveur de commerce électronique doit suivre les fluctuations rapides d'un marché où les décisions des clients et fournisseurs doivent être traitées dans des délais raisonnables. L'importance du temps de réponse comme mesure est due au fait que sa valeur est fortement reliée au coût d'utilisation du système.

Il s'agira donc pour nous d'évaluer le temps de réponse du serveur d'une part et d'analyser le comportement des clients. Ces derniers influencent la plate-forme puisqu'ils sont la source des principales requêtes envoyées au serveur. Donc la performance du système est liée, dans un certain sens, à la conduite des usagers. Celle-ci a été modélisée via la notion de « think time » qui représente la réaction des utilisateurs.

5.2.2 L'environnement des tests

Il est important de décrire l'environnement et les conditions des tests pour que le processus d'analyse et d'interprétation des résultats soit à la fois précis et pertinent. Signalons que les tests sont exécutés à partir du réseau local des laboratoires universitaires de Bell (LUB) à l'Université de Montréal. Le serveur est distant sur le réseau du LUB au Centre Interuniversitaire de Recherche en Analyse des Organisations (CIRANO).

Les scripts de test sont écrits en Jython (Java Python). C'est l'implémentation Java du langage Pyhton ce qui le rend un langage pur Java à cent pour cent et profite donc des points fort de Java. Ses principales caractéristiques, en plus de celles de Java, sont :

- Une syntaxe très simple. Combinée à des types de données évolués (listes, dictionnaires...etc.) elle conduit à des programmes à la fois très compacts et très lisibles.
- Jython est dynamique (l'interpréteur peut évaluer des chaînes de caractères représentant des expressions ou des instructions Jython), orthogonal (un petit nombre de concepts suffit à engendrer des constructions très riches), réflectif (il supporte la *métaprogrammation*, par exemple la capacité pour un objet de se rajouter ou de

s'enlever des attributs ou des méthodes, ou même de changer de classe en cours d'exécution) et introspectif (un grand nombre d'outils de développement, comme le *debugger* ou le *profiler*, sont implantés en Python lui-même).

- Comme *Scheme* ou *SmallTalk*, Jython est dynamiquement typé. Tout objet manipulable par le programmeur possède un type bien définit à l'exécution. Il n'a pas besoin d'être déclaré à l'avance.

Les caractéristiques de l'environnement de test sont :

- **Réseau** : Le réseau local du Laboratoire Universitaire de Bell est un Fast-Ethernet 100 base TX. Les machines sont connectées à l'aide des nouveaux standards de fils UTP (Unshielded Twisted Pair) catégorie 5 (100 ohms) tous prêts pour le Gigabit Ethernet. Les tests s'effectuent à distance et la connexion entre serveur et client se fait via le réseau public Internet.
- **Machines** : Nous utilisons une machine cliente pour lancer les scripts de tests. Les caractéristiques de la machine client et serveur sont décrites dans le tableau 5.1.
- **Outils** : Pour nos tests nous utiliserons la version 2.0 de Jython (Jython2.0) et Java (TM) 2 SDK (Software Development Kit), Standard Edition, version 1.3.1.

GNP est exécuté avec le serveur Weblogic et le SGBD PostgresSQL. Il s'agit de la version 6.1 de BEA-Weblogic (Weblogic6.1). Quant à GNP c'est la version 1.1 que nous étudions à savoir GNP1.1. Cette version a connu des changements et des améliorations pour donner naissance à une deuxième version. Celle-ci n'a pas fait l'objet de notre étude.

Machines / Caractéristiques	Machine Serveur	Machine Client
Processeur	Pentium III	Pentium III
Vitesse du Processeur	2 x 600 MHz	733 MHz
Mémoire cache	256 Kb	256 Kb
RAM	513 Mb	256 Mb
Système d'exploitation	Linux (Red Hat 7.1)	Linux (Red Hat 7.0)

Table 5.1 : Description des machines utilisées dans les tests

5.2.3 Les scénarios de test

Les fonctionnalités offertes par la plate-forme GNP sont assurées grâce à un grand nombre de modules, de classes et de méthodes Java. Il nous est impossible de tester toutes les fonctions de GNP. Par contre, suite à l'étude des différents documents utilisés par GNP, nous avons pu dégager un ensemble pertinent de transactions. Les requêtes fréquemment utilisées par les utilisateurs de GNP sont :

- **submitAnnounce()** : Permet d'initialiser une annonce et de créer une négociation. Elle retourne un document contenant le résultat de l'annonce.
- **submitOrder()** : C'est la méthode par laquelle une mise est soumise. La mise est relative à un produit. Le résultat retourné par cette méthode est le numéro de l'ordre qui a été crée.
- **getOrder()** : Permet d'afficher les informations relatives à un ordre déjà effectué par un participant.
- **getOrdersForProductReference()** : Permet d'afficher une énumération des mises destinées à un produit.

- **getresponsesDocumentForOrder()** : Retourne régulièrement le statut d'une négociation.
- **deleteOrder()** : Donne la possibilité à l'administrateur de supprimer un ordre.

Notons que parmi les six fonctions énumérées plus haut, nous avons utilisé les trois plus importantes et qui sont utilisées par les usagers. Les autres sont soit très rarement évoqués soit réservés à l'opérateur de la place de marché. Ainsi, nous avons utilisé *submitAnnounce()* pour la création des négociations et l'initialisation de l'enchère. *submitOrder()* et *getOrder()*. En effet, puisque notre étude s'intéresse aux réactions des utilisateurs alors il a été judicieux de faire appel à des requêtes reflétant l'interaction des participants avec le système. Durant tous les tests nous avons pondéré *submitOrder()* avec une probabilité de 0,8 tandis que *getOrder()* a reçu une probabilité de 0,2.

Nous avons déroulé les tests en deux phases : une phase « hors ligne » et qui consistait à la simulation des deux distributions. Il s'agit de la loi exponentielle pour de temps d'inter arrivée ou de connexion des utilisateurs et la distribution de Pareto pour le temps de réflexion ou « think time ». Puis une phase « en ligne » qui utilise les résultats de la première phase pour lancer des requêtes vers le serveur GNP. Concernant la phase « hors ligne », nous avons simulé les distributions et généré des temps d'inter-arrivé et des « think time » avec un intervalle de confiance de 95%.

Nos tests comprennent essentiellement deux catégories de scénarios. Dans chaque scénario nous avons exécuté un ensemble de scripts. Ceux-ci génère une charge de test et récupère le temps de réponse.

Dans la première catégorie il a été question de garder la moyenne du « think time » constante. La génération de ces valeurs se fait par la simulation de la loi de Pareto comme expliqué précédemment. L'invariance de la moyenne a été retenue avec un intervalle de confiance de 95%. Pour ce faire, il a fallu changer le paramètre k et α de la distribution de Pareto à chaque exécution d'un nouveau scénario. La borne supérieure du Pareto bornée X_{max} a été fixée à 10 min pour toute l'étude. Ce qui est très logique.

Par ailleurs, dans la seconde catégorie, il s'agissait de fixer le paramètre k à une valeur raisonnable. Rappelons que ce dernier désigne la valeur minimale que peut prendre la

variable aléatoire. Dans notre cas nous l'avons maintenu à 0,5 seconde. Puis en variant le α nous avons récupérer les temps de réponse. Plus le α croît plus la moyenne du « think time » baisse. Ce qui veut dire plus d'intensité pour le serveur.

Il est à signaler que pour chaque scénario nous avons trois cas possibles selon les résultats obtenus dans les différentes étapes qui le constituent :

1. Succès : l'étape s'est terminée avec succès et tous les résultats observés sont raisonnables.
2. Succès Partiel : l'étape s'est bien achevée cependant des résultats paraissent incompatibles avec l'ensemble des résultats ou il y a eu des erreurs lors de l'exécution.
3. Echec : l'étape n'a pas pu être complété.

Un scénario n'est jugé réussit que si toutes ses étapes ont été franchies avec succès.

5.2.3 l'architecture des tests

Nous avons mis en œuvre un ensemble de scripts qui correspondent aux scénarios et fonctions nécessaires à leur exécution. Ainsi nous avons créé notre propre Benchmark qui servira à générer une charge de travail et récupérer les résultats concernant les performances du serveur. L'approche globale suivie pour les tests est montrée sur la figure suivante :

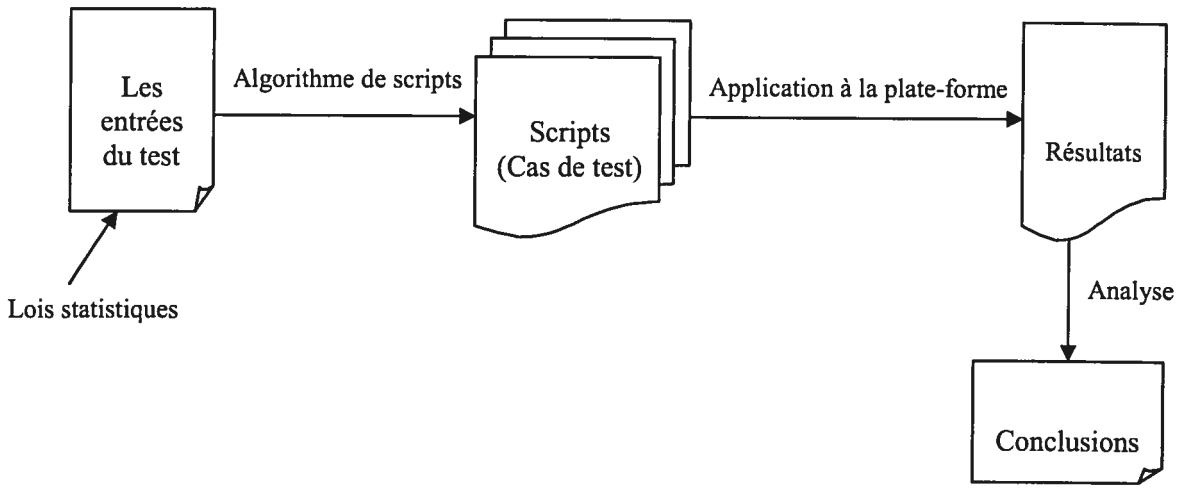

Figure 5.3 l'approche générale de test

Les scripts utilisés se basent sur le mécanisme du multi-threading que le langage Jython offre. Effectivement, un ensemble de thread sont créés et lancés. Ceux-ci ont pour but d'envoyer parallèlement des requêtes au serveur. Quand le thread exécute la transaction il calcul le temps de réponse. Une moyenne des temps de réponse est calculée à la fin du script. L'exécution des scripts est illustrée dans le schéma ci bas :

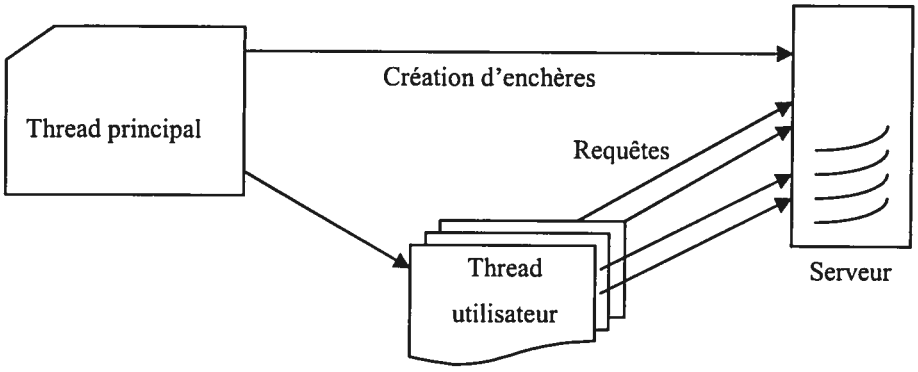

Figure 5.4 : vue d'ensemble sur l'exécution des scénarios

Pratiquement, le processus principal créée une enchère. Des utilisateurs se connectent sur le serveur avec une moyenne de 20 secondes suivant une loi exponentielle. Puis ils choisissent l'une des enchères pour participer aux négociations. Nous avons affecté une probabilité de 0,5 à chaque enchère. Par la suite, les utilisateurs envoient leur requêtes après un moment de réflexion suivant la loi de Pareto. Les requêtes envoyées sont *submitOrder()* et *getOrder()*. Elles suivent une loi binomiale avec une probabilité de 0,8 pour la première et 0,2 pour la seconde. Des exemples de scripts sont donnés en annexe.

Comme nous l'avons mentionné certaines simplifications et hypothèses étaient nécessaires pour l'accomplissement de notre tâche. En pratique, le nombre d'utilisateurs **U** et de requêtes exécutées **R** est déterminé par le nombre de « think time » **T** et de temps d'inter arrivée **I** généré par les distributions. Les relations qui lient les différentes valeurs sont les suivantes : $\mathbf{U = E + I}$ avec **E** est le nombre d'enchères puis $\mathbf{R = T + U}$.

Les deux principaux algorithmes sont présentés dans la suite :

1- Initialisation
2- Création des négociations
3- Exécution du processus principal
- *lire une valeur du temps d'inter arrivée*
- *mettre le processus en veille durant ce temps*
- *connecter un utilisateur à la négociation*
- *lancer le processus « utilisateur »*
- *répéter 3 jusqu'à épuisement des temps inter arrivée*

4- Si tous les processus ont fini aller à 5 sinon attendre
5- Produire le rapport de résultats

Figure 5.5.a L'algorithme principal

De j = 1 à Nombre de requêtes par utilisateur :
- *Choisir un type de requête*
- *Envoyer une requête au serveur*
- *Lire une valeur de « think time »*
- *Mettre le processus en attente pendant ce temps là*
- *i = i +1*

Figure 5.5.b L'algorithme du processus utilisateur

L'architecture du modèle des scénarios est présentée dans l'illustration qui suit :

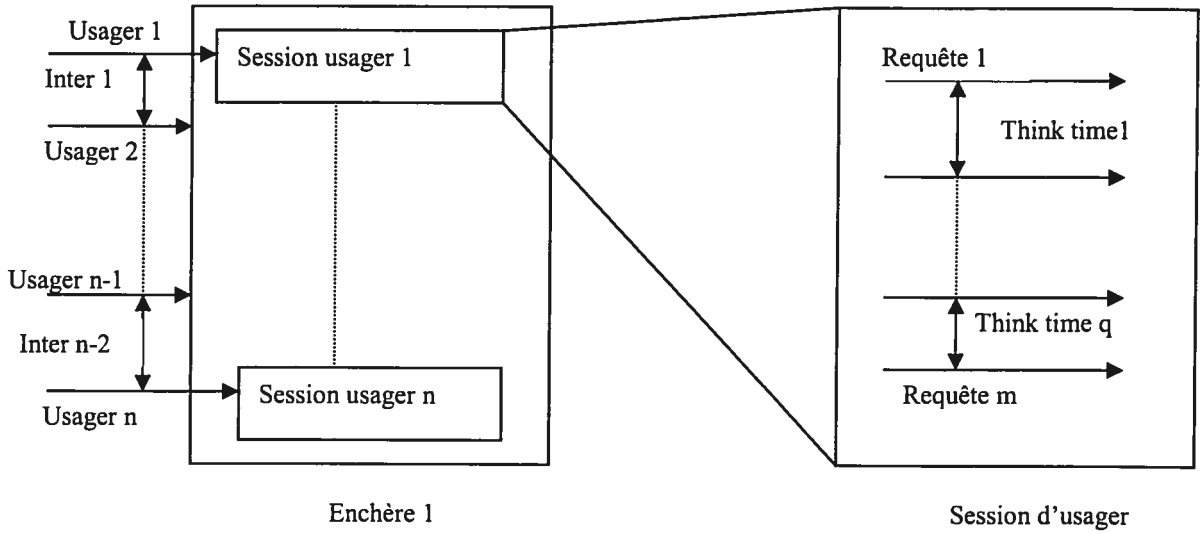

Figure 5.6 Architecture du modèle des scénarios

C'est ainsi que la présentation de notre modèle s'achève pour laisser place à une étape aussi cruciale qui n'est autre que la présentation des résultats et leur analyse.

5.3. Résultats et analyse

L'analyse des données a une importance particulière car il ne suffit pas de récolter l'information mais il est indispensable de bien savoir l'interpréter. Par conséquent, cette étape constitue un point critique dans cette étude. L'analyse de résultats est un outil d'aide à la décision. Des résultats présentés d'une manière où les conclusions sont difficiles à ressortir, ne servent à rien.

Nous allons présenter nos résultats sous forme de tableaux et de graphiques qui les rendront exploitables. Nous nous investirons à analyser et interprété ces résultats. L'objectif essentiel est de donner une vue sur le comportement du serveur GNP et d'expérimenter le modèle de charge que nous avons établi.

5.3.1 Première série de résultats

Nous rappelons que le premier scénario consiste à garder la moyenne du temps de réflexion des utilisateurs invariable. Les deux paramètres de la distribution de Pareto changent donc d'un cas de test à l'autre. Nous comptons neuf cas de test dans ce scénario. Les résultats obtenus sont représentés dans le tableau suivant :

Scénarios	α	k	La moyenne du "think time" (sec)	Temps de réponse moyen du serveur (sec)	Variance	Écart type
1	1.25	0.5	2.075	0.040	0.019	0.140
2	1.35	0.6	2.117	0.089	0.035	0.189
3	1.45	0.69	2.108	0.054	0.009	0.095
4	1.55	0.77	2.114	0.056	0.008	0.089
5	1.65	0.84	2.102	0.065	0.019	0.139
6	1.75	0.91	2.107	0.062	0.012	0.110
7	1.85	0.96	2.080	0.062	0.019	0.137
8	1.95	1.0	2.047	0.088	0.122	0.350
9	3.0	1.37	2.00	0.062	0.007	0.088

Table 5.2 Résultats de la première catégorie de scénarios

Le tableau ci haut résume les résultats recueillis. Il présente le temps de réponse moyen pour chaque scénario. Suite à la manipulation des données brutes que les scripts ont enregistrées concernant le temps de réponse, il s'avère plus intéressant d'exposer les résultats différemment. En effet, afin de mieux apprécier les résultats de la table 5.2 et permettre leur analyse, nous avons transformé les dits résultats en pourcentage. L'ensemble des graphiques 5.7.x présente le pourcentage des requêtes ayant un certain temps de réponse pour chaque scénario. Par exemple, dans le graphique 5.7.a le pourcentage des requêtes ayant un temps de réponse égale à 0,231 secondes est de 2,07% de l'ensemble des requêtes émises. L'analyse des résultats obtenus sera faite dans la section 5.3.3.

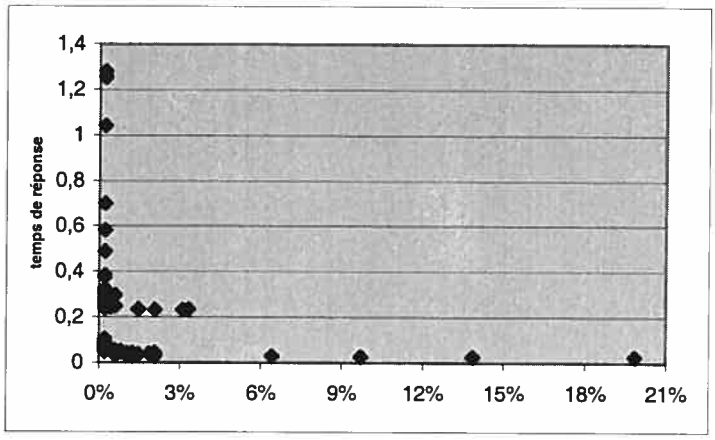

Figure 5.7.a Pourcentage des requêtes pour le scénario 1

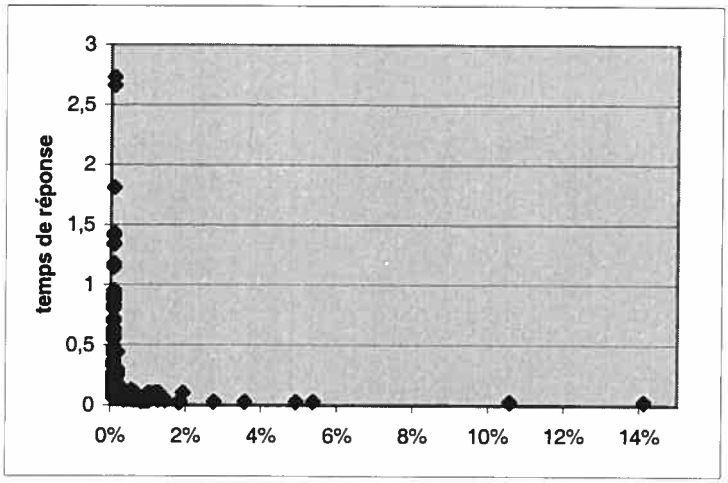

Figure 5.7.b Pourcentage des requêtes pour le scénario 2

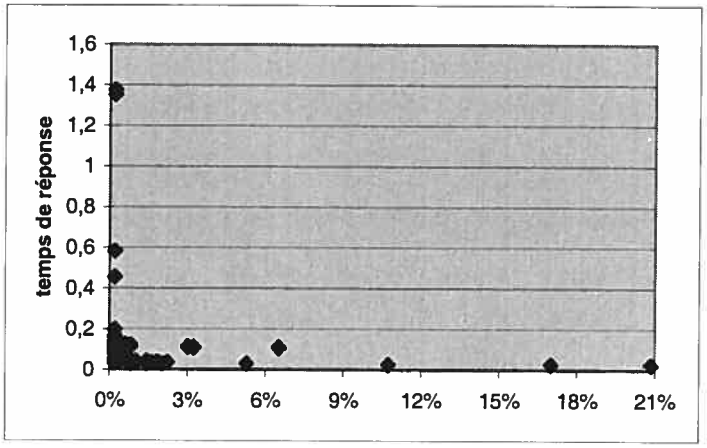

Figure 5.7.c Pourcentage des requêtes pour le scénario 3

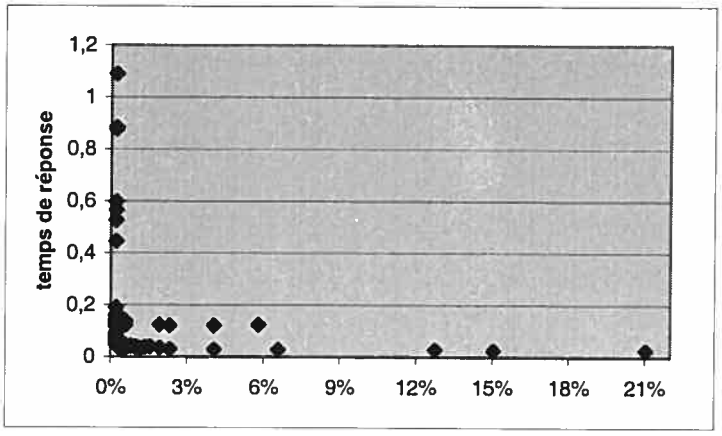

Figure 5.7.d Pourcentage des requêtes pour le scénario 4

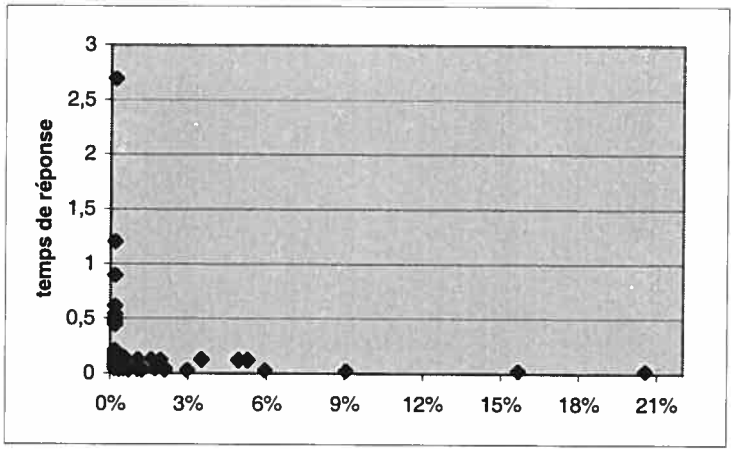

Figure 5.7.e Pourcentage des requêtes pour le scénario 5

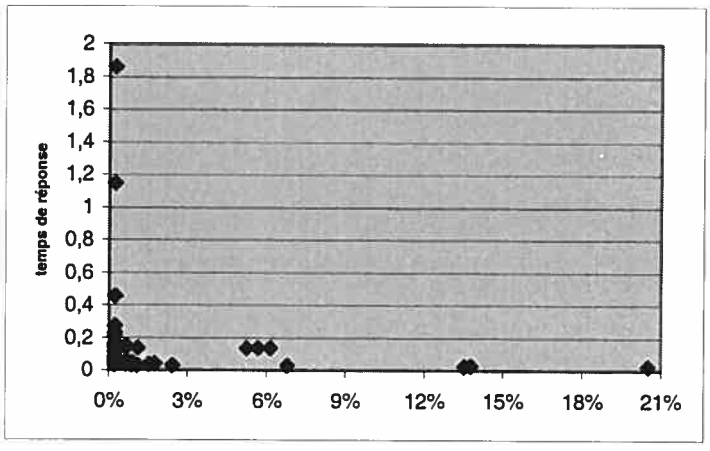

Figure 5.7.f Pourcentage des requêtes pour le scénario 6

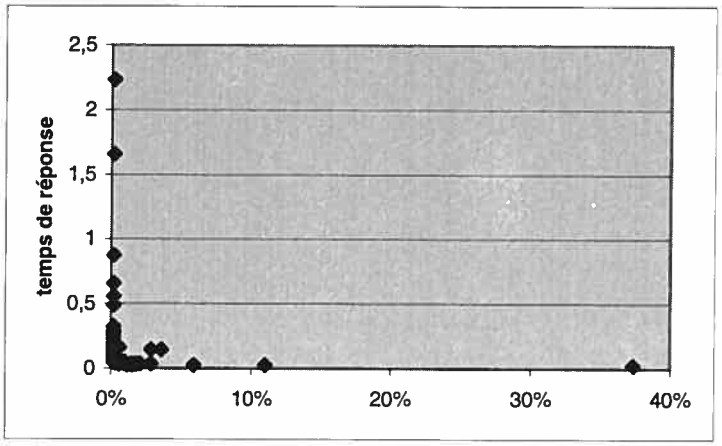

Figure 5.7.g Pourcentage des requêtes pour le scénario 7

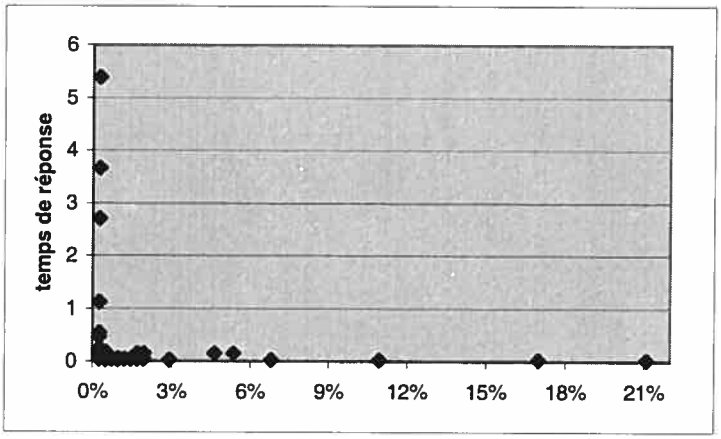

Figure 5.7.h Pourcentage des requêtes pour le scénario 8

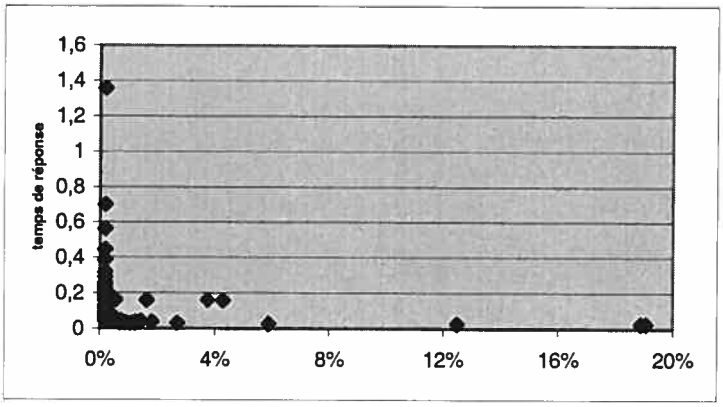

Figure 5.7.i Pourcentage des requêtes pour le scénario 9

5.3.2 Deuxième série de résultats

Dans cette catégorie de scénarios le paramètre k de la distribution de Pareto est maintenu à la valeur 0,5 seconde. Tandis que le paramètre α variait entraînant ainsi la variation de la moyenne du « think time ». En fait, plus le α est grand plus la moyenne du « think time » est petite et donc plus est grande la pression sur le serveur. Nous disposons de dix cas de test pour cette série de scénarios. La table suivante résume les résultats obtenus :

Scénarios	α	« think time » moyen (sec)	Temps de réponse moyen du serveur (sec)	Variance	Ecart type
10	1.35	1.767	0.083	0.071	0.267
11	1.45	1.544	0.081	0.010	0.101
12	1.55	1.380	0.081	0.008	0.0.94
13	1.65	1.256	0.082	0.012	0.112
14	1.75	1.160	0.099	0.041	0.204
15	1.85	1.085	0.078	0.008	0.094
16	1.95	1.025	0.092	0.024	0.157
17	2.20	0.916	0.105	0.024	0.157
18	2.50	0.833	0.090	0.016	0.129
19	3.0	0.70	0.087	0.015	0.124

Table 5.3 Résultats de la seconde catégorie de scénarios

Nous présentons les graphiques des pourcentages telles que décrit auparavant. L'analyse suivra dans la section suivante.

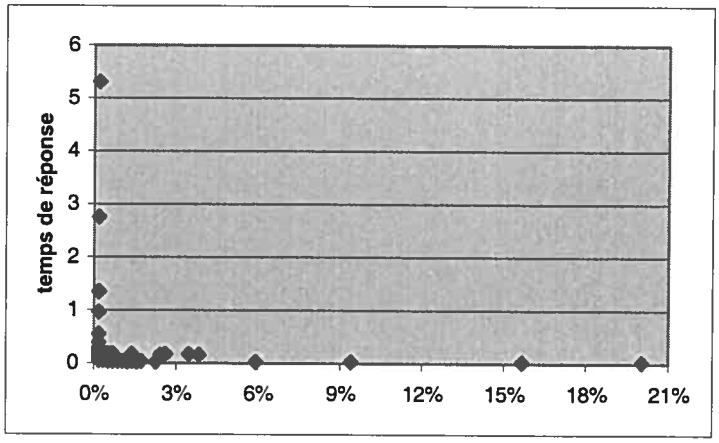

Figure 5.8.a Pourcentage des requêtes pour le scénario 10

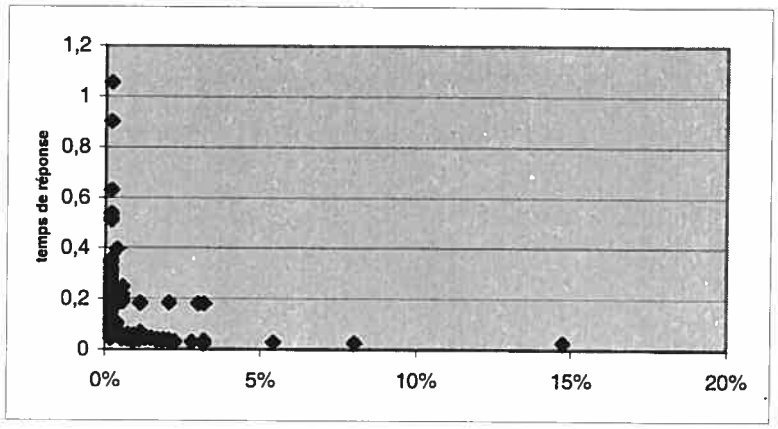

Figure 5.8.b Pourcentage des requêtes pour le scénario 11

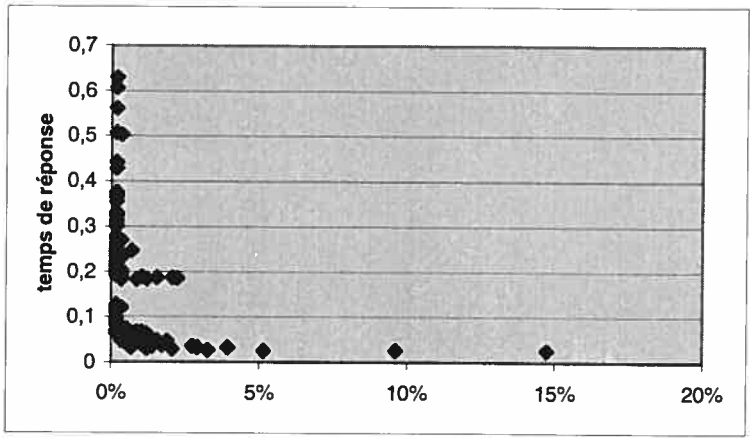

Figure 5.8.c Pourcentage des requêtes pour le scénario 12

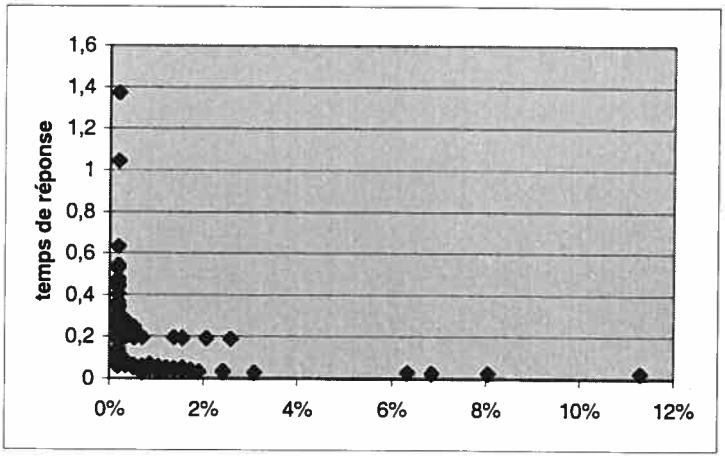

Figure 5.8.d Pourcentage des requêtes pour le scénario 13

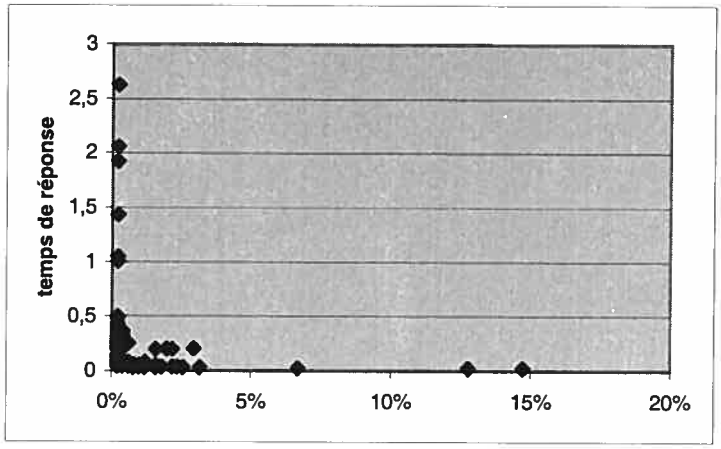

Figure 5.8.e Pourcentage des requêtes pour le scénario 14

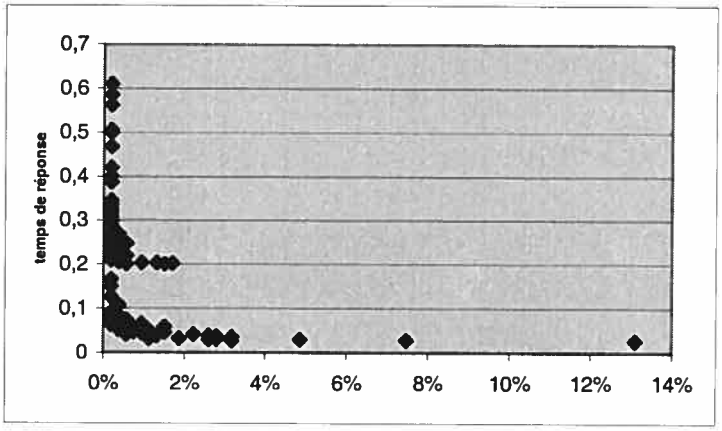

Figure 5.8.f Pourcentage des requêtes pour le scénario 15

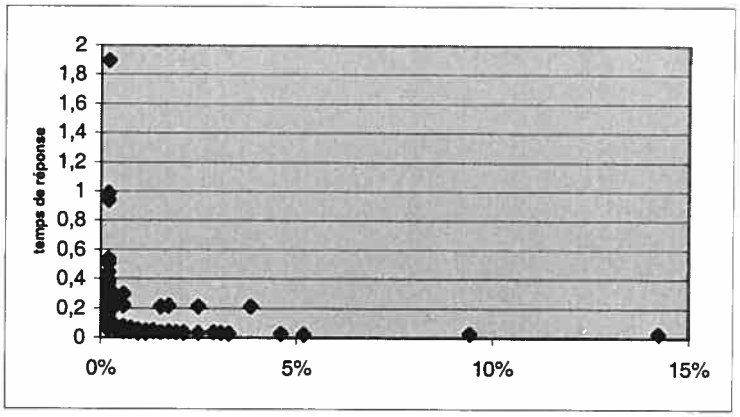

Figure 5.8.g Pourcentage des requêtes pour le scénario 16

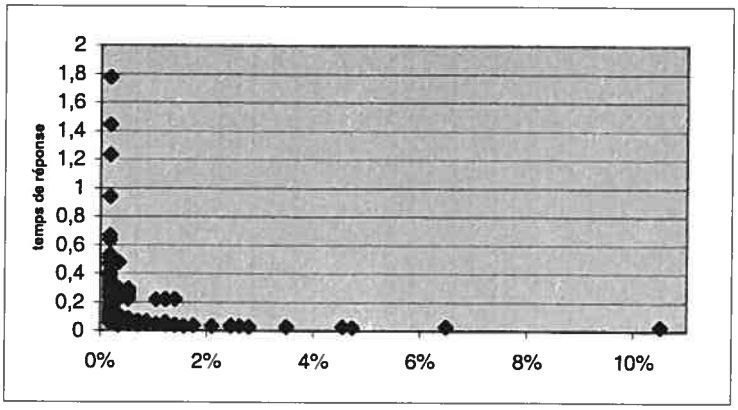

Figure 5.8.h Pourcentage des requêtes pour le scénario 17

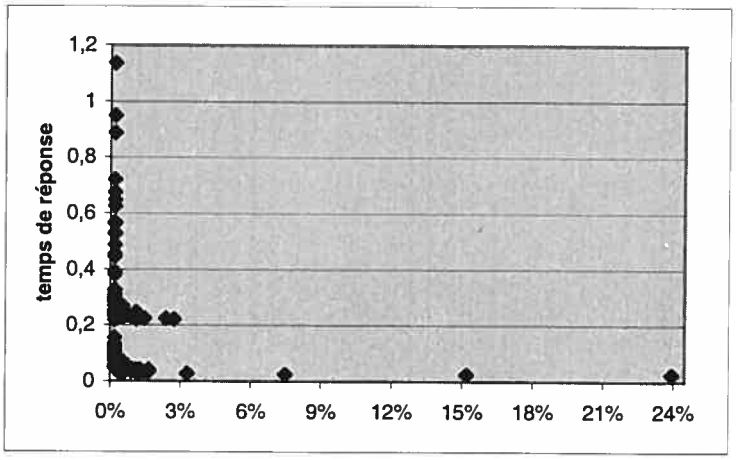

Figure 5.8.i Pourcentage des requêtes pour le scénario 18

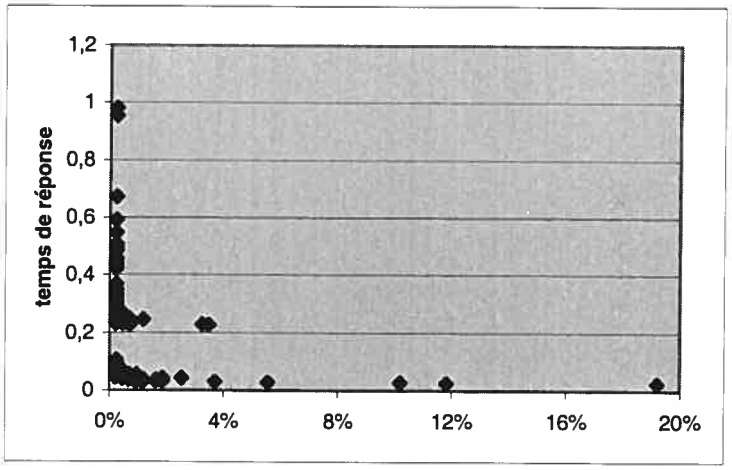

Figure 5.8.j Pourcentage des requêtes pour le scénario 19

Nous nous proposons dans la section suivante d'analyser les différents graphes et essayer d'interpréter les résultats présentés.

5.3.3 Analyse des résultats

L'étape la plus importante dans une étude est bien sûr l'analyse des résultats. En effet, toute l'étude repose sur la pertinence de l'interprétation de la masse de données collectée. Un analyste doit aider à prendre des décisions pour améliorer le rendement. Dans notre étude il s'agissait d'un aspect particulier de la performance de la plate-forme générique de négociation GNP.

Dans un premier temps nous avions tracé des courbes représentants l'évolution du temps de réponse moyen dans le temps et en fonction du « think time ». Les graphiques caractéristiques ne donnaient aucune information utile. Le calcule de la moyenne, la variance et l'écart type tel que présenté sur les tables 5.2 et 5.3 ne révèlent pas le secret des données. En fait, ces valeurs sont proches et s'avère non pertinentes dans notre cas.

Il était donc plus judicieux de faire appel à une autre technique dont les graphes sont donnés plus haut. Il s'agit de considérer les résultats brut de temps de réponse et de calculer le pourcentage de l'occurrence de chacun. Notons que chaque temps de réponse représente une requête. Cette façon de faire nous a permis de tirer quelques conclusions.

Effectivement, nous pouvons clairement remarquer qu'à l'intérieur d'une série de scénarios la dispersion des points sur le graphe change peu par rapport à l'autre série. Particulièrement, les points se concentrent sur deux parties du graphique. Si nous essayons de tracer la limite entre les deux zones nous allons nous rendre compte qu'il s'agit du point représentant le temps de réponse 0,2 seconde. Ce point représente un seuil pour le temps de réponse.

Dans la première série le nombre de points se situant en haut du seuil est beaucoup plus faible que celui de la deuxième série. Autrement dit, le pourcentage des temps de réponse dépassant 0,2 seconde est plus important dans la série 1 que dans la seconde série. A titre d'exemple le tableau ci-dessous montre cette différence pour certains scénarios.

Scénarios	Pourcentage des temps de réponse < 0.2
3 (1ere catégorie)	99.19 %
6 (1ere catégorie)	98.69 %
9 (1ere catégorie)	97.15 %
11(2eme catégorie)	*96.86 %*
13(2eme catégorie)	*89.19 %*
18(2eme catégorie)	*69.35 %*

Table 5.4 Pourcentage des requêtes avec un temps de réponse inférieur à 0,2

Il est clair donc que 0,2 seconde représente un seuil pour notre étude. La différence entre les deux catégories est bien établie. Rappelons, cependant, que dans la première série la moyenne du « think time » était constante avec un intervalle de confiance de 95%. Alors que dans la seconde série la moyenne était sujette à des variations. Le « think time » représente le comportement de l'utilisateur. Il reflète sa réaction et ses interactions avec le serveur. Nous pouvons donc dire que le serveur de commerce électronique sera autant plus chargé lorsqu'il fait face à un certain type de clients. L'administrateur du serveur peut s'attendre à des temps de réponse dépassant 0,2 seconde par requête. Ce seuil peut grimper en fonction du nombre d'utilisateurs, de requêtes et aussi du type d'usagers. En effet, des usagers moins patients seront caractérisés par un « think time » faible et donc le serveur aura plus de requêtes à satisfaire.

Nous pouvons donc caractériser les utilisateurs et modéliser leur comportement grâce à la notion de « think time ». Par suite, le modèle de charge sera beaucoup plus réaliste. L'importance de l'attitude des clients est évidente, surtout pour une plate forme de commerce électronique et particulièrement de négociation.

La distribution de Pareto nous a servi à modéliser les réactions des utilisateurs. Durant l'étude, nous avons varié le paramètre α et observé son influence. Donc une estimation de ce paramètre n'était pas nécessaire. Toutefois, nous pouvons exploiter les expériences que nous avons menées pour faire un choix de valeurs pour le paramètre α de la distribution.

En fait, nous pouvons établir une relation entre l'attitude des utilisateurs et la valeur de α. Ainsi lors de la modélisation de la charge de travail pour le serveur, nous pouvons classifier les usagers en catégories en fonction de leur réaction envers le serveur et pour chaque catégorie utiliser la valeur convenable dans le modèle. Par exemple, des usagers qui misent très souvent et donc génèrent un grand trafic seront plutôt modélisé avec un α entre 2,2 et 3.

En conclusion, nos tests nous ont permis d'établir l'influence du comportement des usagers sur une plate-forme de négociation en particuliers sur le temps de réponse. Nous avons, par ailleurs, mis en place un modèle de charge de travail qui prend en considération les particularités du trafic Web et commerce électronique. Enfin, l'étude nous a servi pour estimer sommairement le paramètre de biais de la distribution de Pareto qui modélise le comportement utilisateur et plus directement le « think time ».

5.4. Conclusion

Dans ce chapitre qui constitue le cœur de notre projet, nous avons décrit le modèle de charge que nous avons utilisé dans les tests. Pour ce faire nous avions préalablement défini les objectifs de l'étude. Le modèle établi répond aux caractéristiques du trafic telles que autosimilarité et la non-stationnarité.

Puis nous avons présenté l'approche suivie dans la mise en œuvre des scénarios de tests. L'architecture des tests a été exposée ainsi que la description générale des scripts exécutés. Ces derniers nous ont permis de récolter des données concernant le temps de réponse.

Ces résultats nous ont permis de constituer une idée sur le serveur suivant la métrique étudiée à savoir le temps de réponse. Il est à signaler que durant cette étude le réseau ainsi que les caractéristiques de l'environnement n'ont pas été prises en compte. Nous nous sommes surtout focalisé sur le serveur lui-même afin d'évaluer sa propre performance en faisant abstraction de l'influence des autres facteurs.

Pratiquement, nous avons établi un seuil pour le temps de réponse et nous avons mis en évidence l'influence du comportement des utilisateurs sur la performance de la plate-forme générique de négociation.

Par ailleurs, la version que nous avons testée est toujours sous développement. C'est une version non stable et qui évolue de jour en jour. Ainsi, les résultats obtenus ne traduisent pas d'une manière fidèle le serveur commercialisable mais constituent plutôt une référence pour le suivi de ses performances et pour les futures améliorations de GNP. Par ailleurs, il est à noter que nous n'avons pas pu terminer tous les tests prévus puisque le serveur n'était plus disponible.

CONCLUSION

Il est évident que les applications et solutions de commerce électronique sont devenues d'une importance vitale pour la nouvelle économie. Malgré le succès dont celle-ci a fait preuve, les experts estiment qu'elle n'est qu'à ses débuts. Pour participer à l'évolution de la Net économie, BELL télécoms Canada et le CIRANO ont mis au point une plate forme générique de négociation (GNP). Toutefois, sachant que la concurrence entre développeurs et constructeurs ne se fait pas attendre dans le monde du commerce électronique, la commercialisation d'une telle structure doit se faire avec précaution. Surtout quand il s'agit de performance. C'est dans ce sens que le test de GNP a été le centre d'intérêt.

L'évaluation des performances des systèmes partagés et des applications de commerce électronique est devenue de plus en plus nécessaire. Cette nécessité se justifie par les prévisions actuelles concernant le nombre de clients potentiels sur la toile. Ce qui implique une grande charge pour les serveurs et infrastructures informatiques de commerce électronique.

Notre projet portait donc sur les tests de performance d'une version de la plate-forme GNP. En particulier, il a été question d'évaluer l'influence du temps de réponse et le comportement des usager sur serveur à l'aide d'une charge de test bien définie. La première tâche à effectuer était de comprendre et de cerner les différents aspects du sujet. D'abord, il fallait situer le projet dans son contexte général et analyser les différentes étapes de son évolution. Ensuite, nous devions étudier les techniques d'évaluation de performance et comprendre la démarche à suivre. Etant donné que notre première tâche était d'élaborer un modèle pour le trafic, l'étude de la caractérisation de la charge de travail constituait une étape important dans notre étude.

Construire des scénarios de tests pour GNP ne serait pas possible sans avoir examiner de prés les applications de commerce électronique et l'environnement GNP. A la lumière de l'étude de tout l'environnement, nous avons défini les objectifs de tests en particulier le temps de réponse. Ce paramètre est très important dans le processus d'évaluation de performance d'un serveur de commerce électronique. Il permet de donner une appréciation du comportement du serveur vu par l'utilisateur final.

La définition des objectifs nous a permis de mettre en œuvre les scénarios et scriptes de tests. Leur exécution qui devait se faire pendant une longue période de test a été une contrainte pour nous. Toutefois, nous avons pu récolter des résultats intéressants servant dans l'analyse des performances de GNP.

La course contre la performance est loin d'être terminée. En effet, élargir les paramètres de performance, étudiés au cours de ce travail, susciterait un intérêt particulier chez toute l'équipe de développement de GNP. Nous pensons, entre autres, à l'évaluation de la mémoire cache et de la saturation du CPU. Un autre volet tout aussi important est le perfectionnement du modèle de la charge de travail qui constitue le piler de toute analyse de performance. D'avantage de phénomènes doivent être inclus dans le modèle, le choix des distributions doit être fait avec soin afin de correspondre au trafic réel.

BIBLIOGRAPHIES

- **[1]** Averill M. Law, W. David Kelton, *Simulation modeling and analysis*, troisième édition, Boston : McGraw-Hill, 2000.
- **[2]** Charles Hélou, *Évaluation de la performance d'une application de type commerce électronique*, mémoire de maîtrise, Université de Montréal 2001.
- **[3]** D. Krishnamurthy, J. Rolia, M. Litoiu, *Performance Evaluation and Stress Testing for E-Commerce Systems*, Electronic Commerce Technologies Trends: Challenges and Opportunities, Editors W. Kou and Y. Yesha, Midrange Computing, Mars 2000.
- **[4]** Daniel A. Menascé, Virgilio A.F. Almeida, *Capacity planning for Web performance : metrics, models, and methods*, Upper Saddle River, NJ : Prentice Hall, 1998.
- **[5]** Diwakar Krishnamurthy, Jerome Rolia, Predicting the QoS of an Electronic Commerce Server : Those Mean Percentiles, *Workshop on Internet Server Performance*, Madison, Wisconsin USA , Juin 1998.
- **[6]** Doherty, W.J. Scheduling TSS/360 for responsiveness, AFIPS Proc. FJCC, 1970.
- **[7]** Graham,R.M, *Performance prediction advanced course on software engineering*, Springer-Verlag, 1973.
- **[8]** Jin Cao, William S. Cleveland, Dong Lin, Don X. Sun, On the Nonstationarity of Internet Traffic, *ACM SIGMETRICS*, 2001.
- **[9]** K Kant, V. Tewari, and R. Iyer, Geist : A generator of e-commerce and internet server traffic, *ISPASS 2001*, Novembre 2001.
- **[10]** Krishna Kant, M. Venkatachalam, Characterizing Non-stationarity in the presence of long-range dependence, *SPECTS 2002*, San Deigo, CA, July 2002.

- **[11]** Mark E. Crovella, Azer Bestavros, Self-Similarity in World Wide Web Traffic Evidence and Possible Causes, *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, May 1996.
- **[12]** Mark E. Crovella, Lester Lipsky, Long-lasting transient conditions in simulations with heavy-tailed wrkloads, *Winter Simulation Conference*, 1997.
- **[13]** Martin F. Arlitt, Carey L. Williamson, Web Server Workload Characterization: The Search for Invariants (Extended Version), *the International conference on measurement and modeling of computer systems*, Mai 1996.
- **[14]** Michael F. Morris, Paul F. Roth, *Computer performance evaluation : tools and techniques for effective analysis*, New York : Van Nostrand Reinhold, 1982.
- **[15]** Nicolas Niclausse, Cesar Jalpa-Villanueva, Sylvain Barbier, Traffic Model and Performance Evaluation of Web Servers, Rapport de recherche de l'INRIA, RR 3840, décembre 1999.
- **[16]** Paul Barford, Azer Bestavros, Adam Bradley, Mark Crovella, Changes in Web Client Access Patterns Characteristics and Caching Implications, *World Wide Web Special Issue on Characterization and performance Evaluation* 1999.
- **[17]** Paul Barford, Mark Crovella, Generating Representative Web Workloads for Network and Server Performance Evaluation, *ACM SIGMETRICS*, 1998.
- **[18]** Peter G. Harrison, Naresh M. Patel, *Performance modelling in communication networks and computer architectures*, Wokingham Grande Bretagne : Addison-Wesley, 1993.
- **[19]** Raj Jain, *the art of computer systems performance analysis : techniques for experimental design, measurement, simulation and modeling*, New York : Wiley, 1991.
- **[20]** Reed, W. J, *The Pareto, Zipf and other power laws,* Economics Letters – 2001.
- **[21]** Robert Gérin-Lajoie, Vincent Trussart, Sophie Lamoureux, Isabelle Therrien, *GNP V1.0 – Architecture Document*, CIRANO, July 2000, v0.2 D.
- **[22]** Shuang Deng, Empirical model of WWW document arrivals at access link, *IEEE International Conference on Communication*, Juin 1996.
- **[23]** Svobodova Liba, *computer performance measurement and evaluation methods: analysis and applications*, New York : Elsevier, 1976.

- **[24]** U. Vallamsetty, K. Kant, and P. Mohapatra, Characterization of E-Commerce Traffic, *International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems*, 2002.
- **[25]** Vern Paxson, Sally Floyd, Wide-Area Traffic : The Failure of Poisson Modeling, *ACM SIGCOMM'94* London, Août 1994
- **[26]** Wei Xie, H. Sun, Y. Cao, K.S. Trivedi, *Modeling of online service availability perceived by web users*, Rapport technique 2002, Center for Advanced Computing and Communication (CACC), Duke university, Durham 2002.
- **[27]** Will E. Leland, Murad S. Taqqu, Walter Willinger, Daniel V. Wilson, On the Self-Similar Nature of Ethernet Traffic (extended version), *IEEE/ACM Transactions on Networking*, 1994.

ANNEXE A

SCRIPT DE SIMULATION DE LA LOI PARETO BORNÉE

```
from string import *
from sys import *
import time
import random
global moyenne_reelle
alpha = 3.0
k=0.5
max=600

def calcul_moyenne_reelle():
        q1 = 1- ( pow(max/k,alpha-1))
        q2= 1- ( pow(max/k,alpha))
        q3 = q1/q2
        q4 = (alpha * max) / (alpha - 1 )
        return q3*q4

def Paretostream(a, ka, m):
        x1 = 1 - pow((ka/m),a)
        x2 = x1 * random.random()
        res = ka / ( pow((1-x2),1/a))
        return res

moyenne_reelle = calcul_moyenne_reelle()
compt = int(0)
```

```
fichier_thinktime = raw_input("donner le nom du fichier resultat\n")

f = open(fichier_thinktime, 'w')
f.write('LES THINK TIME SUIVANT LA LOI PARETO BORNE : alpha =' + `alpha` + ' et k =' + `k` + '\n\n\n')
think = Paretostream(alpha,k,max)
compt = 1
moyenne_th = think
somme = think
while (moyenne_th < moyenne_reelle - 0.05) or (moyenne_th > moyenne_reelle + 0.05):
        f.write(`think` + '\n')
        think = Paretostream(alpha,k,max)
        compt = compt + 1
        somme = somme + think
        moyenne_th = somme / compt
f.write('\n\n la moyenne theorique : ' + `moyenne_th` + '\n' + 'la moyenne reelle : ' + `moyenne_reelle`)
f.write('\n nombre de think time genere : ' + `compt` - 1)
f.close()
```

ANNEXE B

SCRIPT PRINCIPAL

```
#
# AUTHOR : SLIMANE BAH
#
#  SCERAIO 2 :
#
# THIS SCRIPT CREATES A WORKLOAD USING THINK TIME (PARETO) AND THE USER'S # ARRIVAL (EXPONENTIEL).
#
#***************** IMPORTS *************************
from javax.naming import *
from cirano.gnp.beans import NegotiatorSessionHome
from cirano.gnp.beans import AnnouncerSessionHome
from cirano.gnp import ContextUtil
from cirano.gnp import DOMUtil
from javax.ejb import *
from java.util import *
from java.lang import System, Exception
from string import *
from sys import *
import thread
import string
import time
import java.lang.Thread
import random
import globals
```

```
#******************** VARIABLES GLOBALES ****************
global list
global nb_requetes
global mean_interarrival
global arrivals
global max_users
global thinktime
global mean_thinktime
global fichierT
global fichierI
global max_req
#******************** USEFULL FUNCTION **********************
def ajouter(x,y):
    return x+y

def lire_think_time(fichier):
    return float(fichier.readline())

def lire_inter(fichier):
    return float(fichier.readline())

def type_requette():
    u = random.random()
    if u < 0.2 :
        return 1
    if u>=0.2 :
        return 0

def choix_nego():
    u = random.random()
    if u<=0.5 :
        return 1
    if u>0.5 :
        return 0
```

```
def rempli_list_think():
    val= float(globals.fichiert.readline())
    while val != -1.0:
        globals.list_think.append(val)
        val = float(globals.fichiert.readline())

def rempli_list_inter():
    val= float(globals.fichieri.readline())
    while val != -1:
        globals.list_inter.append(val)
        val = float(globals.fichieri.readline())
#************************ THREAD **************
def arrivee_users(indice, intertime, serveur):
# indice commence a zero
    while globals.index_inter <= globals.nb_inter :
        time.sleep(intertime)
        print("user : u" +`indice`+ '\n')
        user = 'u'+`indice`
        pwd = 'u'+`indice`
        globals.list_users.append([user,pwd])
        ctx =  ContextUtil.getNamingContext(serveur, user, pwd)
        if indice<18 :
            nbr_req = 46
        if indice>=18 and indice<=23 :
            nbr_req = 45
        if choix_nego()==1:
            globals.arg = globals.arg1
        if choix_nego()==0 :
            globals.arg = globals.arg2
        argum = indice, globals.arg, nbr_req
        thread.start_new_thread(users_requette,argum)
        if globals.index_inter != globals.nb_inter :
            intertime=globals.list_inter[globals.index_inter]
```

```
            globals.index_inter = globals.index_inter + 1
            indice = indice + 1
#-------------------------------------------------------------
def users_requette(indice, args, nbr_req):
    for j in range(nbr_req):
        if j!= nbr_req-1:
            type_req=type_requette()
            thread.start_new_thread(globals.function[type_req],args)
            waiting = globals.list_think[globals.index_think]
            globals.index_think = globals.index_think + 1
            print ('think = ' + `waiting`+ '\n')
            time.sleep(waiting)
        if j==nbr_req-1:
            type_req=type_requette()
            thread.start_new_thread(globals.function[type_req],args)
#******************* REQUESTS ***********
def envoi_annonce(announcerhome, annfich):
    try:
      announcer = announcerhome.create()
      fdann = open(annfich)
      timebefore = float(System.currentTimeMillis())
      doc = announcer.submitAnnounce(fdann.read())
      diff = float(System.currentTimeMillis() - timebefore)/1000
      print ('le temps de reponse de submitannounce est : ' + `diff`)
      list.append(diff)
      fdann.close()
      globals.compteur = globals.compteur + 1
    except Exception, e:
      print ("EXCEPTION DECLENCHE : ")
      print e
      diff = 0.0
      list.append(diff)
      globals.compteur = globals.compteur + 1
      print ('announce failed : diff = ' + `diff`)
```

```
#-------------------------------------------------------------
def envoi_order(negotiatorhome, prodrefGPK, negoGPK, quoteGPK,ordfich):
      try:
        negosession = negotiatorhome.create()
        fdord=open(ordfich)
        timebefore = float(System.currentTimeMillis())
        negosession.submitOrder(prodrefGPK, negoGPK, quoteGPK, fdord.read())
        diff = float(System.currentTimeMillis() - timebefore)/1000
        print('temps de reponse de submitorder : ' + `diff`)
        list.append(diff)
        fdord.close()
        globals.compteur = globals.compteur + 1
      except Exception, e:
        print ("EXCEPTION DECLENCHE : ")
        print e
        diff=0.0
        list.append(diff)
        globals.compteur = globals.compteur + 1
        print ('order failed: diff= ' + `diff`)
#-------------------------------------------------------------
def lire_order(negotiatorhome, prodrefGPK, negoGPK, quoteGPK, ordfich):
      try:
        negotiator = negotiatorhome.create()
        fdord=open(ordfich)
        negotiator.submitOrder(prodrefGPK, negoGPK, quoteGPK, fdord.read())
        enum = negotiator.getOrdersForProductReference(prodrefGPK)
        for elem in enum:
                ordGPK = elem.getGPK()
                timebefore = float(System.currentTimeMillis())
                document = negotiator.getOrder(ordGPK)
                diff = float(System.currentTimeMillis() - timebefore)/1000
                print('temps de reponse de getorder : ' + `diff`)
                list.append(diff)
```

```
                break
            fdord.close()
            globals.compteur = globals.compteur + 1
        except  Exception, e:
            print ("EXCEPTION DECLENCHE :")
            print e
            diff=0.0
            list.append(diff)
            globals.compteur = globals.compteur + 1
            print ('bringorder failed: diff= ' + `diff`)
#********************** FIN DES FONCTIONS**********************
        #******************** RAPPORT ET RESULTATS ***************
def report() :
    fichier_resultat = raw_input("donner le nom du fichier resultat\n")
    fdout=open(fichier_resultat, 'w')
    fdout.write('Scenario 2  : aplha = 1.35 ; k = 0.6 ; m = 20' + '\n////////////////////////////////\n')
    #k=int(0)
    #for k in range(2):
    total=float(0.0)
    #fdout.write('method: ' + `function[k]` + '\n')
    fdout.write("temps de rep : \n")
    for j in range(len(list)) :
        fdout.write(`list[j]` + '\n')
        total = total + list[j]
        fdout.write('\n')
    fdout.write('total= ' + `total` + '\n')
    moyenne = float(total /len(list))
    fdout.write('Moyenne = ' + `moyenne` + "\n")
    fdout.write('-------------------------------------\n')
    fdout.write ('*********************************************\n')
    fdout.close()
##****************** MAIN PROGRAM ***********
        #************ INIT ************
globals.compteur = int(0)
```

```
globals.nb_inter = 22
globals.index_think = 0
globals.index_inter = 0
fichierT = open('think2','r')
fichierI = open('inter2','r')
globals.function = [envoi_order, lire_order]
globals.fichiert = fichierT
globals.fichieri = fichierI
globals.list_users = []
list=[]
max_req = 1098
globals.list_think = []
globals.list_inter = []
rempli_list_think()
rempli_list_inter()
server = "t3://merlin.lub.umontreal.ca:7001"
file_ann="/home/skalante/Projets/gnp/tests/announce.xml"
file_ord ="/home/skalante/Projets/gnp/tests/order.xml"
#*************** FIRST NEGOCIATION *********************
ctx = ContextUtil.getNamingContext(server,'announcer','announcer')
print ("connexion 1 reussie au serveur .....")
fdann = open("/home/skalante/Projets/gnp/tests/announce.xml")
announcerhome = ctx.lookup("beans.AnnouncerSessionHome")
announcer = announcerhome.create()
doc = announcer.submitAnnounce(fdann.read())
negoGPK = long( DOMUtil.get(doc, "/gnpDocument/announceResult@negotiationGPK") )
prodrefGPK = long(DOMUtil.get(doc,
"/gnpDocument/announceResult/productReference@GPK") )
quoteGPK = long(DOMUtil.get(doc, "/gnpDocument/announceResult@quoteGPK") )
negotiatorhome = ctx.lookup("beans.NegotiatorSessionHome")
fdann.close()
# ********************* SECOND NEGOCIATION ******************
ctx1 = ContextUtil.getNamingContext(server,'announcer','announcer')
print ("connexion 2 reussie au serveur .....")
```

```
fdann1 = open("/home/skalante/Projets/gnp/tests/announce.xml")
announcerhome1 = ctx1.lookup("beans.AnnouncerSessionHome")
announcer1 = announcerhome1.create()
doc1 = announcer1.submitAnnounce(fdann1.read())
negoGPK1 = long( DOMUtil.get(doc1, "/gnpDocument/announceResult@negotiationGPK") )
prodrefGPK1 = long(DOMUtil.get(doc1,
"/gnpDocument/announceResult/productReference@GPK") )
quoteGPK1 = long(DOMUtil.get(doc1, "/gnpDocument/announceResult@quoteGPK") )
negotiatorhome1 = ctx1.lookup("beans.NegotiatorSessionHome")
fdann1.close()
# ************************************************************
globals.arg1 =  negotiatorhome, prodrefGPK, negoGPK, quoteGPK, file_ord
globals.arg2 = negotiatorhome1, prodrefGPK1, negoGPK1, quoteGPK1, file_ord
#************EXECUTION OF THREADS
print ("traitement en cours.....")
i = int(0)

        # ********* 1st USER for the 1st NEGO ***********
user='u' + `i`
print("user : u" +`i`+ '\n')
pwd='u' + `i`
ctx = ContextUtil.getNamingContext(server, user, pwd)
nbr_req = 46
args = globals.arg1
argum = i, args, nbr_req
thread.start_new_thread(users_requette,argum)
i=i+1
        #*************** 1st USER for the 2nd NEGO ***********
user='u' + `i`
print("user : u" +`i`+ '\n')
pwd='u' + `i`
ctx = ContextUtil.getNamingContext(server, user, pwd)
nbr_req = 46
args = globals.arg2
```

```
argum = i, args, nbr_req
thread.start_new_thread(users_requette,argum)
i=i+1
#****************************************
interarrival = globals.list_inter[globals.index_inter]
globals.index_inter = globals.index_inter + 1
argum1 = i, interarrival, server
thread.start_new_thread(arrivee_users,argum1)
#************************* Waiting the end of threads
j = int(0)
while globals.compteur < max_req :
        print("wait...\n")
        print `globals.compteur`
        time.sleep(1)
fichierT.close()
fichierI.close()
print('longueur ' + `len(list)`) # verification
report()
print "fin du script bye\n----------\n "
```