Modélisation, Évaluation de Performances et Dimensionnement du World Wide Web

Nicolas Niclausse

Thèse de Doctorat en Sciences
École Doctorale SPI (Discipline : Informatique)
Université de Nice – Sophia Antipolis

Email: Nicolas.Niclausse@sophia.inria.fr

Vendredi 25 Juin 1999

PLAN

- Introduction
- Modélisation du trafic Web
- Evaluation de performances du WWW
- Caches Web
- Conclusion et perspectives

Le WWW

➢ **Le World Wide Web**

- ➞ Créé au CERN en 1990 par Tim Berners-Lee
- ➞ Protocole Client/Serveur HTTP
- ➞ Multi-plateformes

  NeXT, UNIX, Windows, MacOS, etc.
- ➞ Serveurs HTTP forment une toile sur Internet
- ➞ Documents hypertextes HTML

  objets inclus (images, sons, CSS, etc.)

⇨ **Trafic HTTP majoritaire sur Internet depuis 1995**

Problématique

➢ Croissance exponentielle du trafic

– Nombre de pages:
120 M (Juin 1997), 200 M (Nov. 1997), 275 M (Mars 1998)
Croissance: = x2 en moins de 9 mois
= 19,4 million pages / mois
= 16% de nouvelles pages par mois
= 7,5 nouvelles pages par secondes

– Nombre de sites:
1 570 000 serveurs Web (1997), 2 851 100 serveurs Web (1998)
Croissance: = 106000 serveurs par mois
= 6,7% serveur de plus / mois

Problématique *(suite)*

➢ **Besoin de qualité de service**

- ➞ Temps de réponse trop grand ⇒ le client quitte le site
- ➞ Cause: Serveur et/ou réseau

➞ **Comment assurer cette QoS ?**

- ➞ Dimensionnement du réseau et des serveurs
- ➞ Améliorer protocoles et serveurs

➞ **Différentes approches possibles:**

- ➞ Théorique (files d'attente)
- ➞ Simulations
- ➪ Expérimentations

➞ **Besoin d'un modèle du trafic**

Problématique *(suite)*

➢ **Grande variabilité du trafic**

→ Rafales, Corrélations (LRD [Willinger et al. 94]).

→ Distributions sous-exponentielles (fichiers, requêtes) [Crovella et al. 96]

$$\forall \varepsilon > 0, \lim_{x \to \infty} e^{\varepsilon x} \mathbb{P}[X > x] = \infty$$

Pareto, LogNormale, Weibull, Inverse Gaussienne, etc.

➢ **Plusieurs composantes:**

→ Réseau: HTTP au dessus de TCP

→ Client: parsing HTML, affichage

→ Serveur: Ordonnancement processus, E/S, scripts

État de l'Art

- **Caractérisation statistique**
  - Taille des fichiers : queue lourde (Pareto, ...)
  - Trafic HTTP = Corrélations (LRD ?)
  - Popularité = Zipf (*i.e.* très concentrée sur quelques documents: 50% des clics sur 1% des serveurs)
- **Modélisation et analyse de performances**
  - Problèmes théoriques difficiles et nouveaux ⇒ peu de résultats
  - Recours aux benchmarks pour évaluer les serveurs réels:
  - SPECWeb96, WebStone = modèles simplistes (taille des fichiers, trafic périodique)

État de l'Art *(suite)*

- **SURGE [Crovella et Barford 98]**
  - → Distributions analytiques, analyse de fichiers de logs

    Taille fichiers, requêtes, popularité, inter-requêtes actives/inactives
  - → Pas de notion de client, ni hypertexte.
  - → Inter-requêtes = dépendant du protocole.
  - → Résultats sur réseau Ethernet seulement
- **Client [Pitkow et al. 98]**
  - → Nombre de clics = Inverse Gaussienne
  - → Contenu du serveur = Graphe pondéré

Modèle

- **Objectifs**
  - Modèle indépendant du protocole
  - Prendre en compte la structure hypertexte
- **Trafic HTTP**
  - Requêtes utilisateurs (clic)
  - Requêtes automatiques (images, etc.)

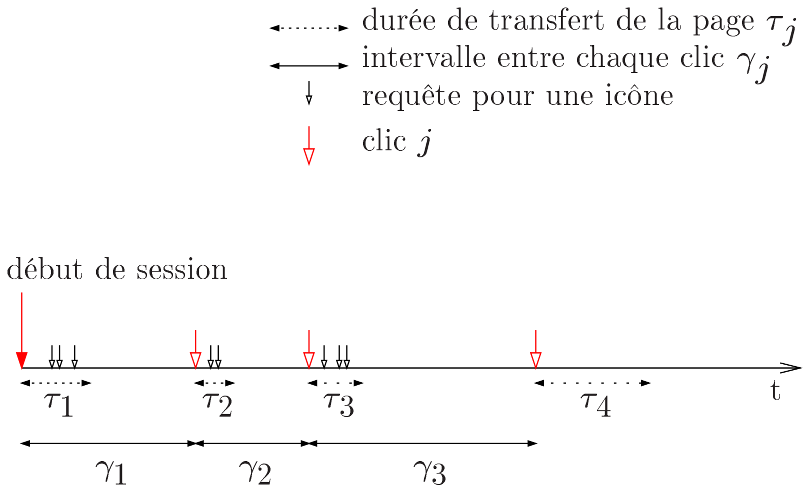
    dépend du protocole

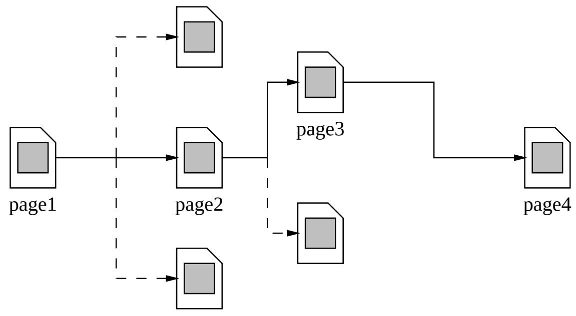
Modèle *(suite)*

➢ Session d'un client

→ Nombre de clics

→ Temps de lecture entre clics





Modèle *(suite)*

- **Serveur (architecture hypertexte) = graphe pondéré**
  - documents Web (URL) = noeud
  - liens HTML = arcs
  - probabilité de routage (popularité)
- **Processus Ponctuel Marqué**
  - Date d'Arrivée = Début de Session
  - Marques:

    Nombre de clics

    Temps de lecture entre chaque clic

    Adresse (URL) de chaque clic (ou taille de fichier pour simulateur)

Analyse statistique

- **Analyse de fichiers logs**
  - → Identifications des sessions
  - → Manque d'informations $\Rightarrow$ heuristiques
- **Méthodologie: identification des distributions**
  - → Test d'Ajustement (Anderson-Darling, Kolmogorov-Smirnov)
  - → Métriques (λ^2, différence entre fonction de répartition)

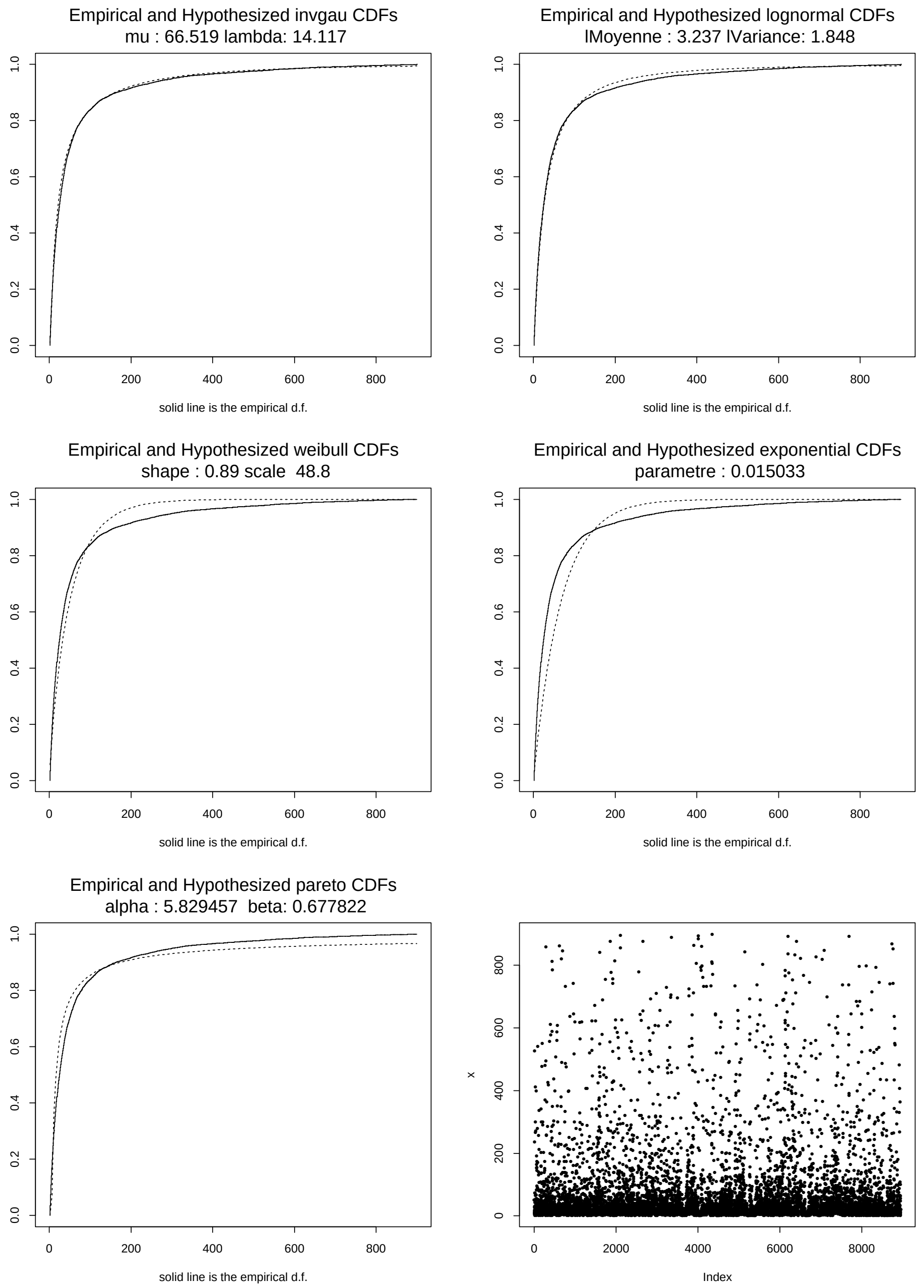


FIG. 1 – Durée entre clics

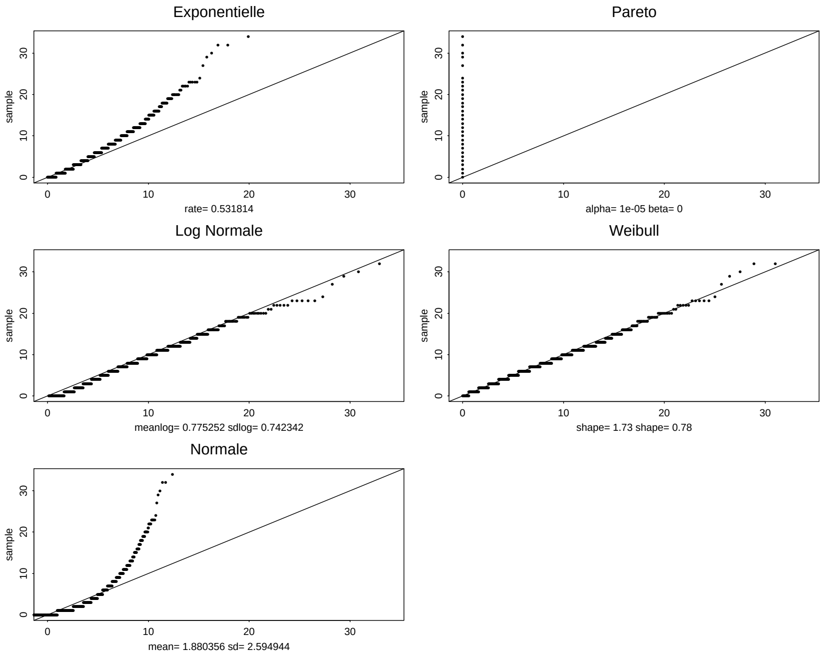
Analyse statistique *(suite)*

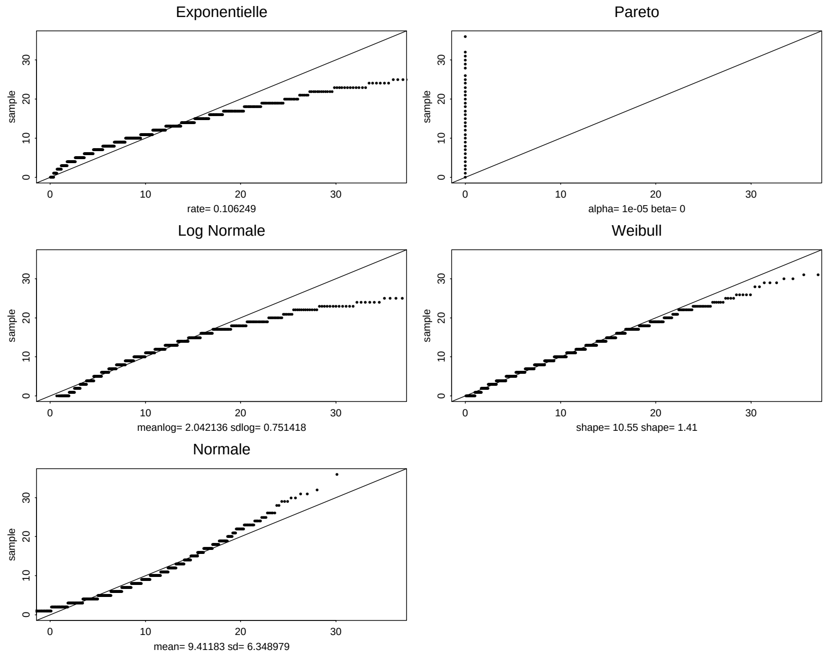
- **Processus d'arrivées :**
  - Requêtes HTTP: Corrélées, loi Weibull
  - Arrivées de Sessions: Processus de Poisson
- **Clients**
  - Nombre de clics: loi sous-exponentielle *Inverse Gaussienne, Pareto*
  - Durée entre clics: loi sous-exponentielle *Inverse Gaussienne, Lognormale*
- **Remarques**
  - Test d'ajustements souvent négatifs : autres approches ? (mélanges, ...)

Validation

- **Trafic HTTP**
  - → Corrélations (H >0.5)
  - → Loi Marginale = loi Weibull
- **Popularité**
  - → Calcul de popularité à partir des fichiers de logs perfectible
- **Localité temporelle**
  - → Distribution identique aux données réelles (Lognormale)

14 Modélisation du trafic Web





WAGON

➢ Approche Expérimentale

- Générateur de trafic HTTP: WAGON
  Web trAffic GeneratOr and beNchmark
- Développé en collaboration avec Véronique Marquion, Frédérique Ah Yave et César Jalpa-Villanueva, sous la direction de Zhen Liu.
- Trafic = P.Ponctuel marqué, distributions et paramètres configurables.
- Utilise contenu d'un serveur réel
- Simulation de délai et limitation de débit.
  modem, satellite, WAN, etc.

Environnement d'expérimentation

- **Stations UNIX, Ethernet 10Mb/s**

  Linux, Solaris

- **2 serveurs (contenu) avec caractéristiques différentes**

  (taille des documents, nombre d'images/page HTML)

- **Paramètres clients**

Variable	Loi	Moyenne
Arrivée de sessions	P.Poisson	$(0.0005 \leq \lambda \leq 0.02)$
Nombre de clics	Inv. Gaussienne $(\mu = 5; \lambda = 3)$	5
Durée inter-clic	LogNormale $(m = 3; \sigma = 1.1)$	36.8s

Comparaison de HTTP/1.0 et HTTP/1.1

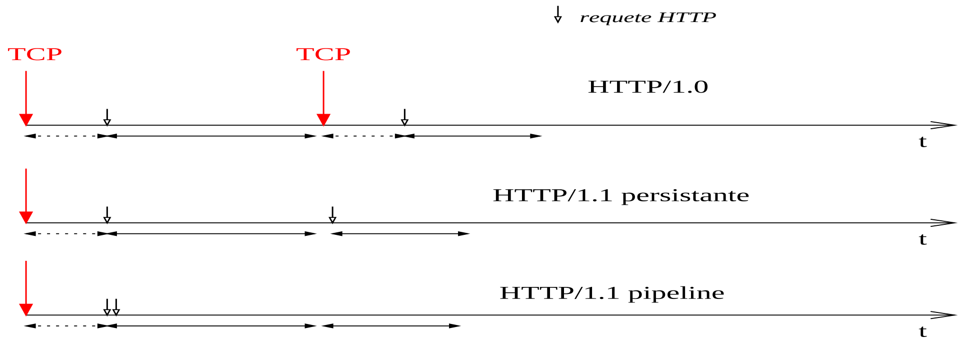
➢ Interactions TCP / HTTP

➩ HTTP/1.0 défini indépendamment de TCP ⇒ peu efficace:

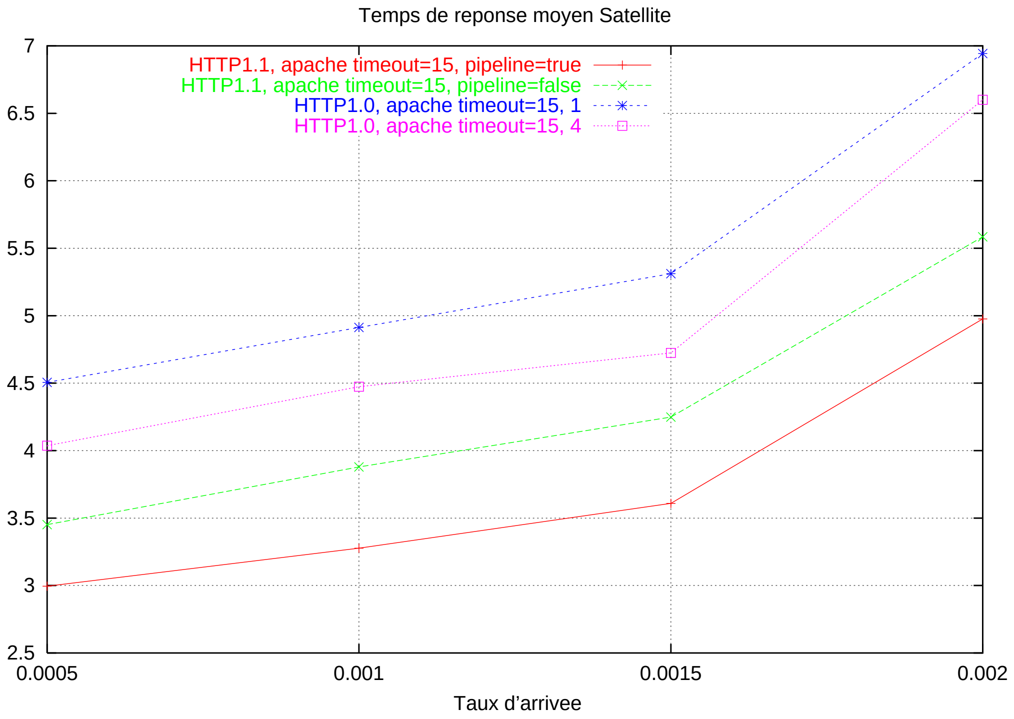
➞ 1 requête HTTP = 1 nouvelle connexion TCP

➞ Taille petite (50% < 3Ko, AOL) ⇒ 80% des paquets transmis en slow-start

➩ HTTP/1.1: connexions persistantes + pipeline



Comparaison de HTTP/1.0 et HTTP/1.1 *(suite)*

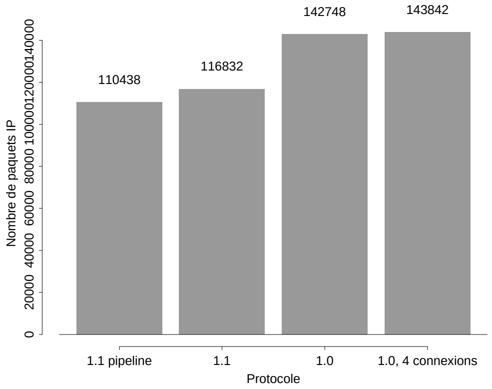


Comparaison de HTTP/1.0 et HTTP/1.1 *(suite)*

➢ Performances

→ Temps de réponse moyen HTTP/1.1 $\prec$ HTTP/1.0 avec 4 connexions //

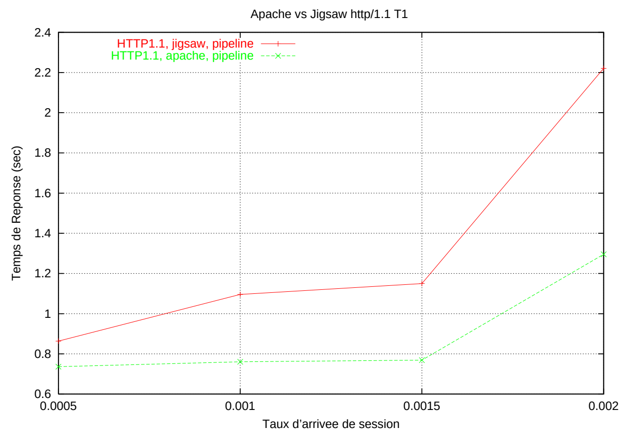
→ Nombre de paquets IP inférieur en 1.1



→ Pipeline efficace (non implémenté dans IE ou Netscape)

Comparaison de Apache et Jigsaw

- ➩ Apache = 55% des serveurs Web
- ➩ Jigsaw = implémentation du W3C (Java)
- ➢ **Performances**
  - → Apache plus rapide (C plus rapide que Java)



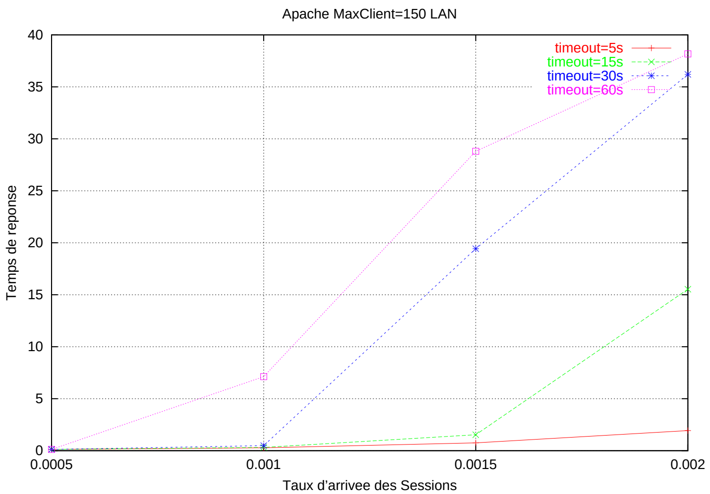
  - → Jigsaw utilise beaucoup plus de CPU

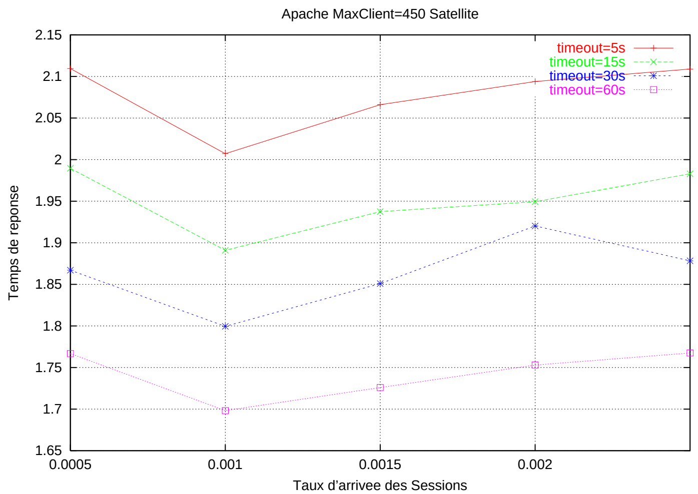
Paramétrage de Apache et Jigsaw

➢ Gestion des connexions persistantes

→ Apache : peu adaptative (temps de réponse x10 si mal configuré)

→ Jigsaw : adaptative, mais beaucoup de paramètres





➢ **Utile pour mettre au point serveur:**

Détection d'un bug dans Jigsaw

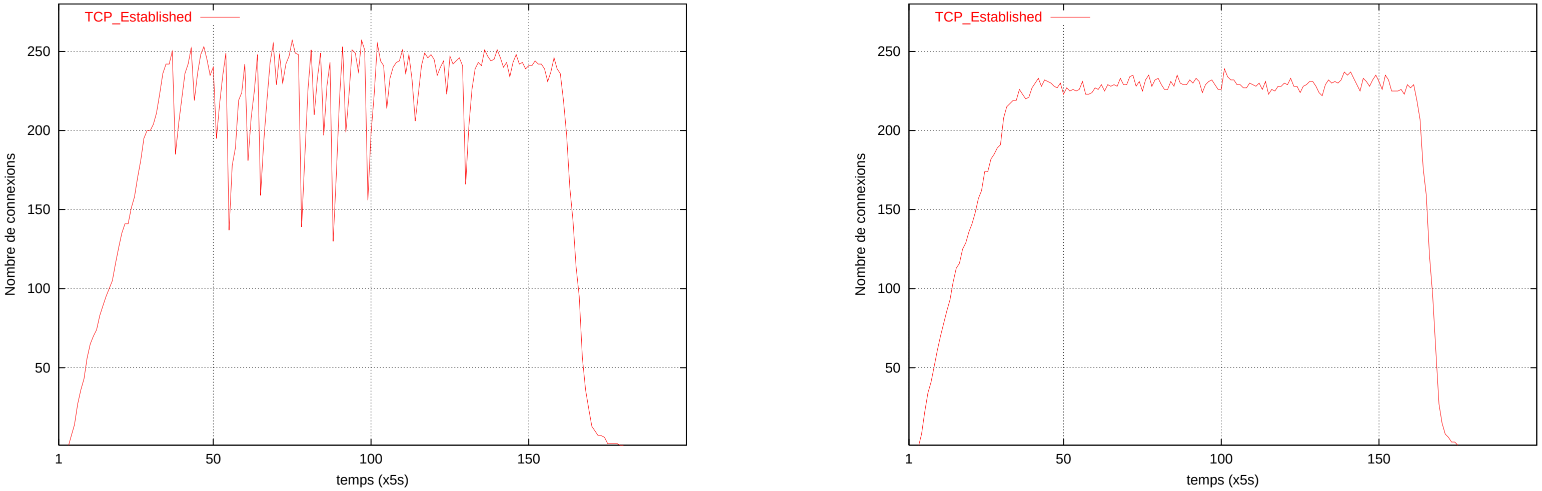
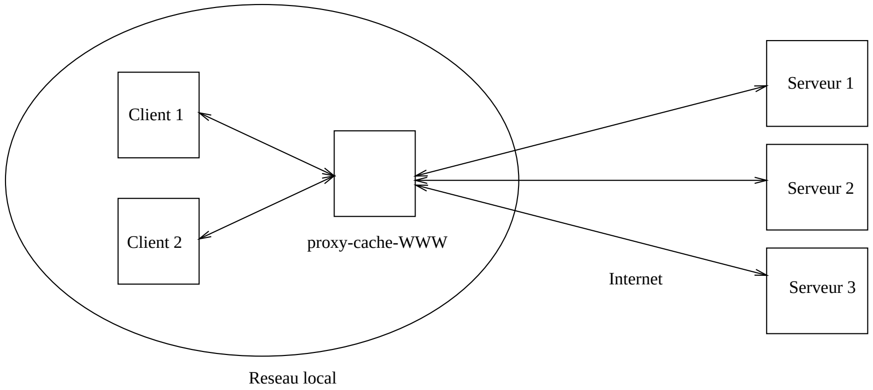


FIG. 2 – connexions TCP ouvertes

Introduction



- Réseau lent (WAN)
- Serveurs chargés
- Petits nombre de documents très populaires
- **Cache près des clients**

Politiques de remplacement

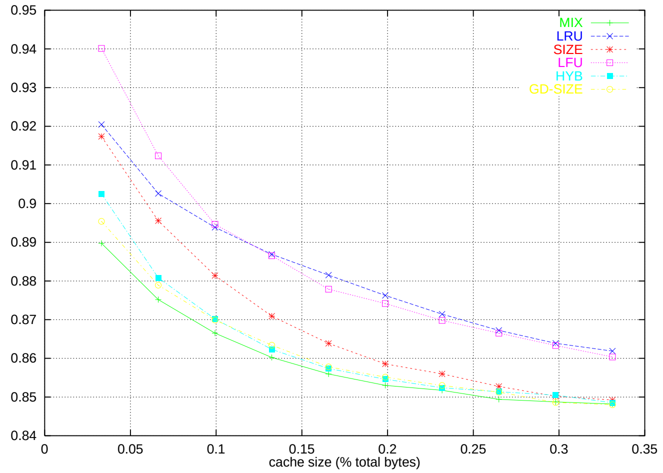
- Cache devient plein $\Rightarrow$ quel document remplacer ?
- Politique de remplacement: *LRU, LFU*
- Particularités du Web:
  - → Tailles des fichiers (queue lourde), temps d'accès variables.

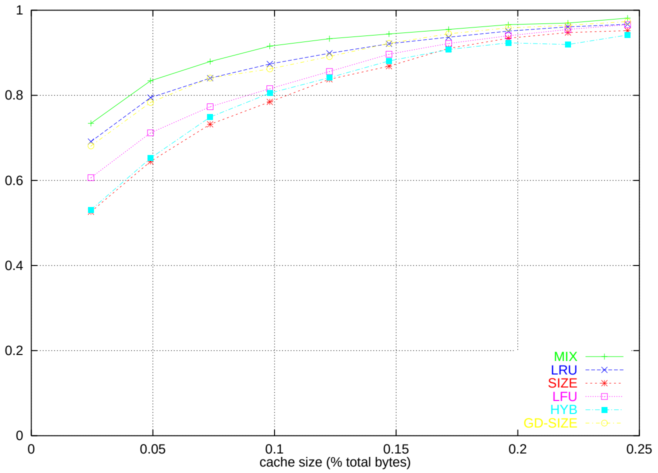
Cache Client (proxy)

- **Objectif:** Diminuer temps de réponse utilisateur
- **Idée:** Prendre en compte plusieurs critères, dont le temps de chargement du document.
- **Méthode:** Simulation à partir de traces réelles
- **Politique MIX**

$$\frac{lat^{0.1} * nref}{tref * size}$$

Cache Client (proxy) *(suite)*





➢ **Résultats:**

→ Temps de réponse $\leq$ aux autres politiques

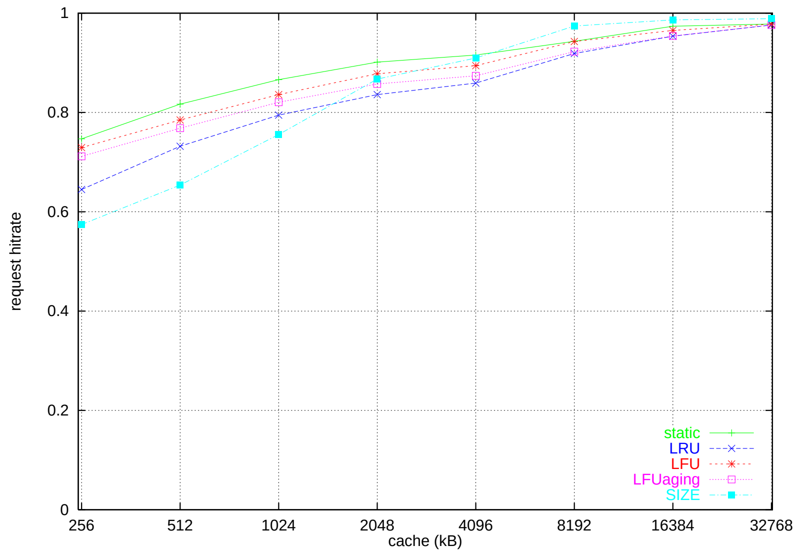
→ Vrai aussi pour taux de réussite (requête et octet)

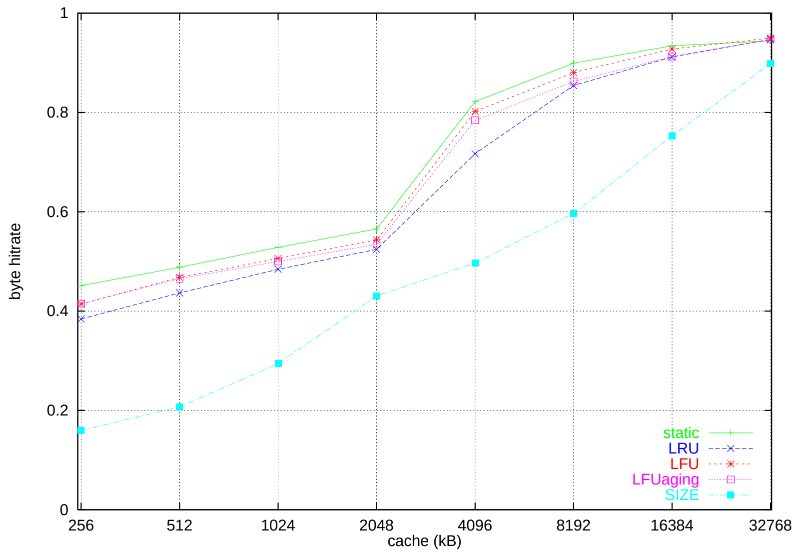
➢ **Problème:** + couteux (mémoire, CPU), amélioration peu importantes

Cache serveur

- **Objectif: diminuer temps de traitement (accès disque)**
  - Cache en mémoire vive (RAM)
  - LRU peu efficace
- **Idée:** Cache statique *(garder en mémoire documents les plus accédés sur période précédente)*
  - Basé sur analyse de fichiers de logs (jour précédent par ex.)
  - Maximiser taux de réussite (octet transmis à partir du cache) et coût CPU.
- **Preuve d'optimalité** (sous certaines hypothèses stochastiques)
- **Simulations:** Bonnes performances pour cache petit
- **Défaut et amélioration :** Peu réactif, $\Rightarrow$ cache statique + dynamique

Cache serveur *(suite)*





Conclusion

- ➞ Politique de remplacement de caches
- ➢ Modèle générique
- ➢ Approche expérimentale: WAGON
- ➞ Implémentation de HTTP/1.1 + pipeline souhaitable
- ➞ Gestion des connexions persistantes importante

- ➞ « Web Traffic Modeling and Performance Comparison Between HTTP1.0 and HTTP1.1 », *Performance'99 (à paraître)* [Z. Liu, N. Niclausse et C. Jalpa-Villanueva]
- ➞ « Static Caching of Web Servers », *Multimedia Computing And Networking 1998* [Z. Liu, P. Nain, N. Niclausse et D. Towsley]
- ➞ « A new efficient caching policy for the World Wide Web», *Workshop on Internet Server Performance (WISP'98)* [Z. Liu, N. Niclausse et P. Nain]

Perspectives

- HTTP/NG
- Amélioration analyse statistique
- Influence des distributions, robustesse ?
- Analyse théorique (files d'attente)
- Rajouter dans le modèle documents inclus
- Evaluation des proxy-caches
- Expérimentation sur réseau dédié
- Nouveaux types d'utilisations (XML, ...)