TP Évaluation de performances

Sebastien Gougeaud

2021/2022

Quelques informations avant de démarrer ce TP :
- le langage de programmation utilisé pour les deux exercices est laissé à votre choix ;
- l'exercice 2, portant sur la génération de trace de requêtes, sera à rendre par mail à l'adresse suivante : `pro.seb.gougeaud@gmail.com` ;
- le rendu du mail consistera en une archive (zip, tar, tar.gz) contenant le code source, le matériel de compilation (si besoin) ainsi que l'exemple de fichier de sortie demandé à la question 5 ;
- le rendu devra être fait à la fin de la séance ;
- les rendus seront comparés les uns aux autres, et des points seront enlevés s'il y a une correspondance suspecte entre eux.

1 Exploration de trace de requêtes

Objectif : Soit un ensemble de fichiers contenant une trace de requêtes, vous allez déterminer à quelle trace correspond un profil donné.
Une requête est constituée de cinq champs décrits dans le tableau ci-dessous :

CHAMP	DESCRIPTION
estampille	identifiant unique de la requête
date de début	date à laquelle la requête est lancée
type	lecture (0) ou écriture (1)
adresse	premier octet ciblé par la requête
taille	nombre d'octets ciblés par la requête

Trois fichiers de trace de 25'000 requêtes vous sont fournis, traces qui ont été générées avec des paramètres d'entrée différents. Vous devez alors répondre aux questions suivantes en les justifiant. Pour ceci, vous allez devoir écrire des programmes ou des scripts vous permettant d'extraire des traces les informations liées à la question et de générer des graphes mettant en valeur ces informations.

1- Quelle trace génère le plus d'écritures ? Laquelle génère le plus d'octets écrits ?

2- Quelle trace cible le plus grand espace d'adressage ?

3- Quelle trace génère des requêtes de taille constante ? Quelle trace génère des requêtes avec des tailles générées aléatoirement en suivant une loi normale ?

4- Quelle trace possède le plus long stream de lecture ?

5- Quelle trace génère le travail le plus long (en temps) ?

2 Génération de trace de requêtes

Objectif : Soit un ensemble de paramètres donnés en entrée de programme, vous allez concevoir un programme qui va générer une trace de requêtes.
L'implémentation du programme peut être décomposée de la manière suivante :

1- Création de la structure de données représentant une requête : pour ceci, se référer aux champs décrits dans l'exercice 1.

2- Écriture des fonctions de génération des champs de requête. Le tableau ci-dessous vous donne pour chacun d'entre eux les générations à implémenter :

CHAMP	GÉNÉRATION
estampille	séquentielle, de 0 à n
date de début	séquentielle, à partir de t_0 avec un pas t
	aléatoire, $t_i = t_{i-1} + t$ avec t loi uniforme entre $[t_{min}, t_{max}]$
type	aléatoire, probabilité p d'être une écriture
	stream, constitué de n_r lectures puis n_w écritures, puis n_r lectures, etc.
adresse	séquentielle, à partir de a avec un pas égal à la taille s
	aléatoire, loi uniforme sur un espace donné
	aléatoire, avec espace de taille h, probabilité p_h et espace de taille c, probabilité $1 - p_h$
taille	constante, de taille s
	aléatoire, loi uniforme entre $[s_{min}, s_{max}]$
	aléatoire, loi normale en fonction de la moyenne s_{moy} et la variance var

3- Gestion de l'entrée des paramètres de génération. Vous avez plusieurs possibilités : lecture d'un fichier de configuration, extraction des arguments de la ligne de commande ou lecture de l'entrée standard. Évitez l'écriture des paramètres dans le fichier source, cela force à re-compiler le programme à chaque fois que l'on modifie les paramètres.

4- Écriture de la fonction principale du programme, qui récupère les paramètres, génère N requêtes, puis les écrit dans un fichier de sortie `output_file`. N et `output_file` sont également des paramètres de configuration.

5- Exécutez votre programme pour obtenir une trace de 10'000 requêtes, avec les paramètres suivants : date de début séquentielle à partir de 0 avec un pas d'1 seconde ; type aléatoire, 70% de lecture ; adresse aléatoire avec 20% de chance de tomber entre 0 et 800 Mo et 80% de chance de tomber entre 800 Mo et 1 Go ; taille aléatoire avec une loi uniforme entre 4 ko et 1 Mo.