Laboratoire 1 : Entrées-Sorties avec Tampons

Introduction

Le but de ce laboratoire est de travailler un peu avec l'écriture de fichier et de se familiariser avec les entrée-sortie des fichiers. Il permet également de voir les différences entre un IO bufférisé ou non.

Code

Afin d'extraire les statistiques qui étaient intéressantes de notre programme, nous avons légèrement modifier le code en y ajoutant des lignes permettant d'écrire dans un fichier .csv afin de pouvoir les traiter facilement dans Excel. Pour se faire, nous somme passer par plusieurs objets :

File("Output.csv") → FileOutputStream(File) → BufferedOutputStream(FileOutputStream) →

PrintStream(BufferedOutputStream)

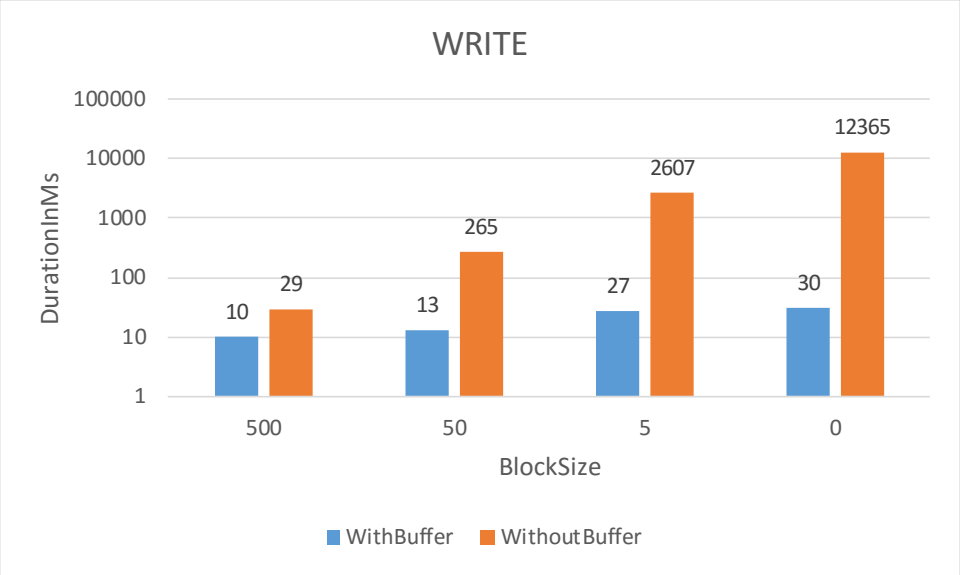
Nous utilisons donc un PrintStream qui nous permet d'utiliser les méthodes sur cet objet comme un simple println(), ce qui facilite grandement la tâche.
L'extraction des données se fait donc directement dans le code avec des appels à un objet PrintStream au endroit convenable. J'ai opté pour la méthode simple, c'est-à-dire directement mettre dans le main les écriture du fichier en modifiant légèrement les méthodes de test afin qu'elles retournent le temps. Il aurait été possible d'également modifié les méthodes de test afin qu'elles sortent directement les résultats dans un fichier mais je ne l'ai pas fait par soucis de simplicité de lecture du code.

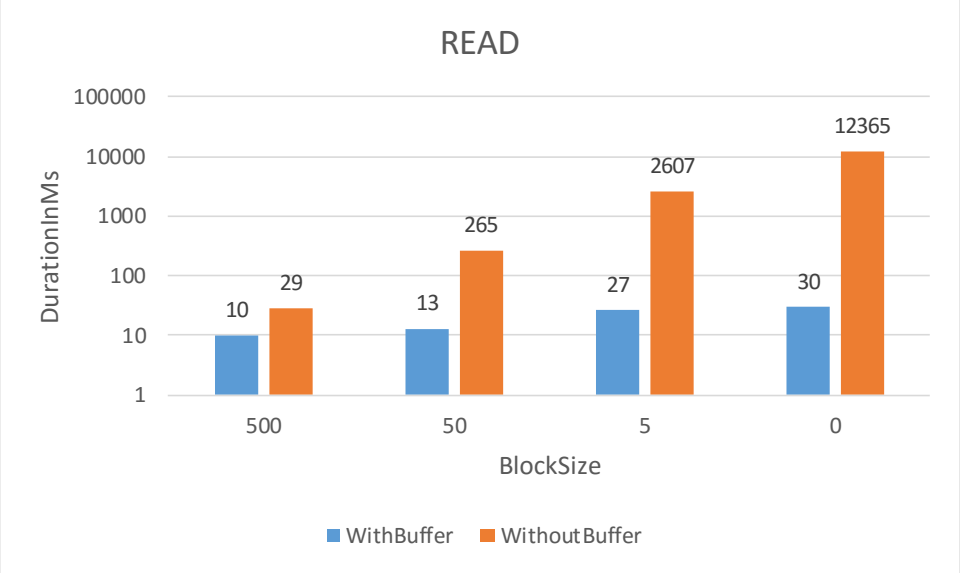
Résultats

Les tests sont effectués en mode WRITE et READ sur des IO Bufférisé ou non. Cela nous donne les données suivantes.

operation	strategy	blockSize	fileSizeInBytes	durationInMs
WRITE	BlockByBlockWithBufferedStream	500	10485760	26
	BlockByBlockWithBufferedStream	50	10485760	34
	BlockByBlockWithBufferedStream	5	10485760	68
	ByteByByteWithBufferedStream	0	10485760	180
	BlockByBlockWithoutBufferedStream	500	10485760	61
	BlockByBlockWithoutBufferedStream	50	10485760	591
	BlockByBlockWithoutBufferedStream	5	10485760	3497
	ByteByByteWithoutBufferedStream	0	10485760	16380
READ	BlockByBlockWithBufferedStream	500	10485760	10
	BlockByBlockWithBufferedStream	50	10485760	13
	BlockByBlockWithBufferedStream	5	10485760	27
	ByteByByteWithBufferedStream	0	10485760	30
	BlockByBlockWithoutBufferedStream	500	10485760	29
	BlockByBlockWithoutBufferedStream	50	10485760	265
	BlockByBlockWithoutBufferedStream	5	10485760	2607
	ByteByByteWithoutBufferedStream	0	10485760	12365

Cela peut être traduit en 2 graphes (WRITE et READ) de la manière suivante :





Interprétation

On voit clairement que l'IO bufférisé prends moins de temps à s'effectuer que celui sans buffer. Cela est dû au fait que si on ne bufférise pas l'IO, afin de lire un fichier de la même taille, le programme doit faire bien plus d'appels système afin de lire tous les caractères du fichier ce qui est long et couteux pour un ordinateur.