Julien Leroy RES

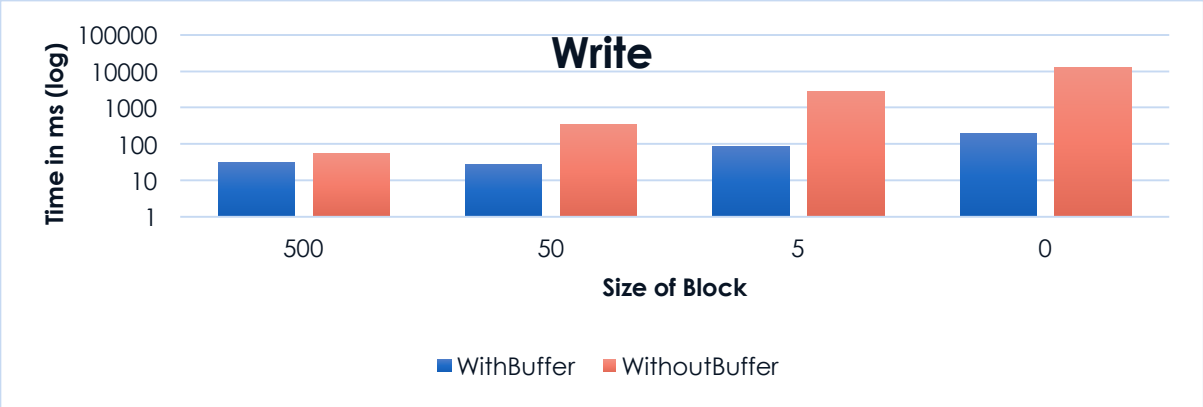
https://github.com/limayankee/RES_IO

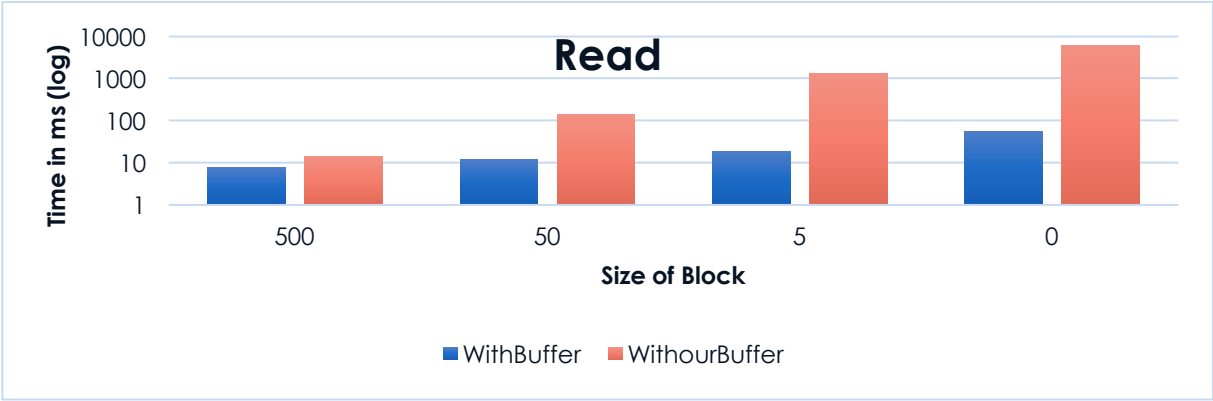
Condition de l'expérience

Ces donnes ont été générées sur un MacBook PRO OS 10.11, 4 cœur 2.8 GHz Intel Core i7, disque SSD PCI HFS+ journalisé

Présentation des données

operation	strategy	blockSize	fileSizeInByte	durationInMs
WRITE	BlockByBlockWithBufferedStre	500	10485760	32
WRITE	BlockByBlockWithBufferedStre	50	10485760	29
WRITE	BlockByBlockWithBufferedStre	5	10485760	87
WRITE	ByteByByteWithBufferedStrea	0	10485760	197
WRITE	BlockByBlockWithoutBuffered	500	10485760	57
WRITE	BlockByBlockWithoutBuffered	50	10485760	351
WRITE	BlockByBlockWithoutBuffered	5	10485760	2883
WRITE	ByteByByteWithoutBufferedSt	0	10485760	12582
READ	BlockByBlockWithBufferedStre	500	10485760	8
READ	BlockByBlockWithBufferedStre	50	10485760	12
READ	BlockByBlockWithBufferedStre	5	10485760	19
READ	ByteByByteWithBufferedStrea	0	10485760	56
READ	BlockByBlockWithoutBuffered	500	10485760	15
READ	BlockByBlockWithoutBuffered	50	10485760	139
READ	BlockByBlockWithoutBuffered	5	10485760	1310
READ	ByteByByteWithoutBufferedSt	0	10485760	6221





Analyse des données

Pour l'écriture et la lecture comme nous pouvons le voir sur le graphique si dessus, avec une taille de bloc de 500 octets il y a peu de différance avec ou sans buffer. Ceci c'explique par le fait que les blocs de 500 font presque la taille des blocs du disque. Le driver de disque a moins de travail à effectuer.

Par contre quand on s'écarte de la taille du bloc, la différance entre l'accès avec buffer et sans buffer devient de plus en plus importante, l'échelle du graphique étant logarithmique, il faut regarder les graduations. Pour l'écriture byte à byte avec buffer nous sommes a 197ms et sans buffer nous sommes a 12582, soit ~60 fois plus. Et pour la lecture c'est encore pire nous sommes à 56 ms en buffer et à 6221, soit ~110 fois plus.

Tout ceci s'explique par le fait que les accès sans buffer font accès au disque pour chaque byte, alors que l'accès avec buffer accède à des blocs dus disque et si les bytes chercher sont dans le même blocs, il n'y a pas besoin de le recharger.

Méthode de capture des données

Pour la l'écriture dans le fichier csv, j'ai créer une classe 'CsvFile'. Dans le constructeur, j'ouvre un fichier en sortie avec 'OutputStreamWriter' et 'FileWriter'. J'ai mis à disposition une méthode 'write' qui écrit dans le fichier et une méthode 'close' qui ferme le fichier. Dans la class 'BufferedIOBenchmark' j'ai déclaré un objet 'CsvFile' nommé 'csv' que j'initialise au début du 'main'. Ensuite à chaque écriture dans le log j'écris aussi dans le fichier csv avec la méthode 'write'. Le temps a été place dans une variable 'long' afin d'avoir la même valeur dans le terminal et dans le fichier csv.