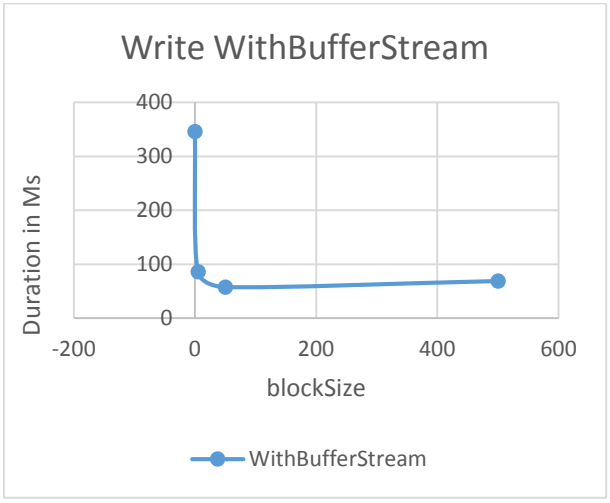
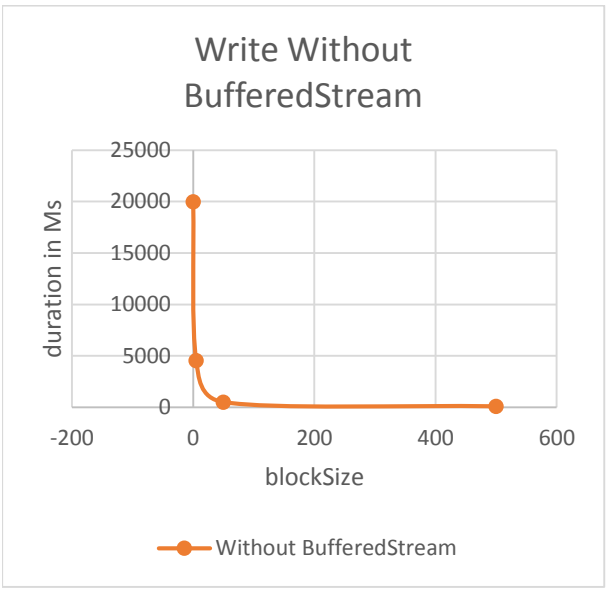
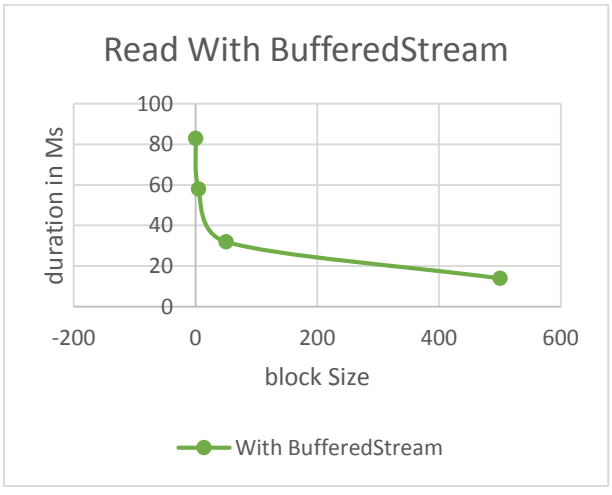
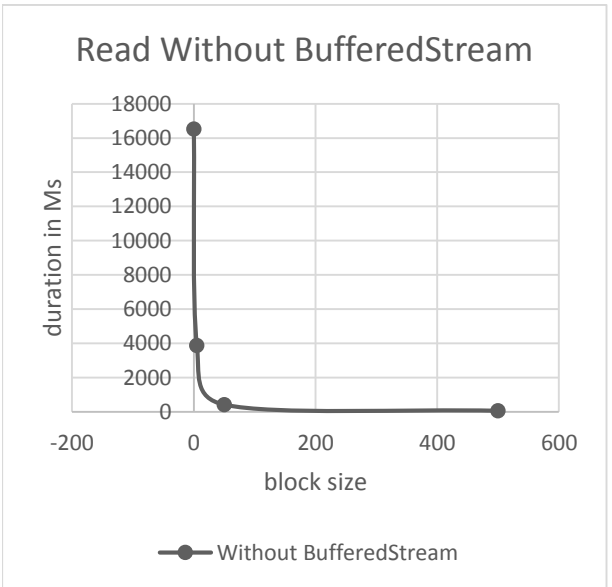
Dans ce texte il est question d'analyser les performances obtenu en utilisant des flux « bufférisés » ou pas. Pour cela des expériences d'écritures et de lectures ont été effectué sur des jeux de données générés aléatoirement et les temps d'exécutions ont été notés.

1) Pour ma par les expériences ont été effectués sur :
   - Machine Windows 7 Professionnel.
   - Système d'exploitation 64 bits
   - RAM 4,00 Go
   - Intel(R) Core(TM) i3-3110M
   - CPU 2.40GHz
   -
2) En fonction du type d'opération effectué, du fait que le flux utilisé sois bufférisé ou pas (la stratégie employé) , la taille des blocs de données et la taille du fichier en Byte, nous avons obtenu des mesure de temps. Pour les quelque expérience qui ont été fait, le résumé des donnée relevé est présenté ci-dessous.









3) Au travers des différents plots que je plus haut il est indéniable que nous obtenons de meilleurs performance et terme de gain de temps avec des flux bufférisé que sans. Pour les mêmes jeux de données les temps d'exécutions sont largement différents.
Par exemple l'écriture byte à byte avec tampon prend 346 ms tandis que l'écriture byte à byte sans tampons prend 19974 ms.
Si on considère plutôt l'écriture ou la lecture des blocs de donné avec tampon on voit pour des blocs de 50 bytes par exemple en écriture le temps nécessaire est de 58 ms et en lecture le temps nécessaire est 32 ms hors l'écriture et la lecture sans usage de tampon me donne des temps de 495 ms et 408 ms respectivement.

4) Sur conseil du professeur j'ai dû créer 3 interfaces et 3 classes pour attendre le but du laboratoire. Les interfaces sont pour des buts d'éventuel généricités. La classe MyData pour stocker les données relevant, la classe FileRecorder pour enregistrer les données dans un fichier et la classe CsvSerializer pour formater les données avant de les stocker.

```
public class CsvSerializer implements ISerializer{

  public CsvSerializer() {}

  public void serialize(IData data, PrintStream printStream){

    //formatting the Data according to the csv format
    String csv = ((MyData)data).getOperation() + "," +
                 ((MyData)data).getStrategy() + "," +
                 ((MyData)data).getBlockSize() + "," +
                 ((MyData)data).getFileSizeInBytes() + "," +
                 ((MyData)data).getDurationInMs() ;

    //writing to the csv file through a PrinStream connection to the file
    printStream.println(csv);
  }
```

```
public class BufferedIOBenchmark {

  static final Logger LOG = Logger.getLogger(BufferedIOBenchmark.class.getName());
  private long timeTaken;
  private IData mydata;
  private ISerializer serializer = new CsvSerializer();
  private IRecorder recorder = new FileRecorder("result.csv", serializer);

  public BufferedIOBenchmark() {
     ((FileRecorder) recorder).init();
  }
```

```
  }
  LOG.log(Level.INFO, "  > Done in {0} ms.", timeTaken = Timer.takeTime());
  IData mydata = new MyData("WRITE", ioStrategy.toString(), blockSize, numberOfBytesToWrite, timeTaken);
  ((FileRecorder) recorder).record(mydata);

}
```