# Bases de données documentaires et distribuées, http://b3d.bdpedia.fr

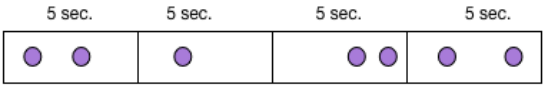
## Fenetrage avec Flink

Philippe Rigaux

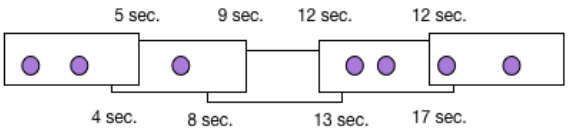
le cnam

# Le fenêtrage avec Flink

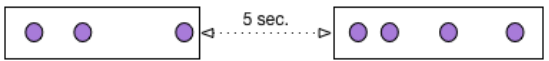
Flink permet de découper les flux en **fenêtres**. Trois types de fenêtres possibles.




**Fenêtres fixes (TumblingWindow)**




**Fenêtres glissantes (SlidingWindow)**




**Fenêtres de session (SessionWindow)**

# Opérations sur les fenêtres

Attention à bien évaluer les ressources nécessaires !

- application incrémentale d'une fonction *Reduce()*
- application incrémentale d'une fonction *Fold()*
- enfin, toute fonction s'appliquant à l'ensemble des éléments de la fenêtre, et qui ne peut pas s'appliquer incrêmentalement.

La troisième est potentiellement consomatrice de mémoire.

# Un premier exemple de fenêtre

Au préalable, lancer notre générateur de flux sur le port 9000.

```
import org.apache.flink.streaming.api.windowing.assigners._;

val stream = senv.socketTextStream("localhost", 9000, '\n')
val w = stream.map ( { x => Tuple1(x.toInt) } )
              .windowAll(TumblingProcessingTimeWindows.of(Time.seconds(5)))
               .fold("Liste: ") { (acc, v) => acc + " | " + v }
               .print()
senv.execute(" ")
```

On cumule les entiers de la fenêtre, toutes les 5 secondes.

# Avec fenêtre glissante

La fenêtre couvre 10 secondes, et est évaluée toutes les 5 secondes.

```
import org.apache.flink.streaming.api.windowing.assigners._;

val stream = senv.socketTextStream("localhost", 1234, '\n')
val w = stream.map ( { x => Tuple1(x.toInt) } )
               .windowAll(SlidingProcessingTimeWindows.of(Time.seconds(10),
               .fold("Liste: ") { (acc, v) => acc + " | " + v }
               .print()
senv.execute(" ")
```

Le `windowAll()` indique qu'il n'y a pas de partitionnement.

# Dernier exemple : avec partitionnement

L'exemple suivant partitionne le flux d'entiers en 2 : les pairs et les impairs.

```
import org.apache.flink.streaming.api.windowing.assigners._;

val stream = senv.socketTextStream("localhost", 9000, '\n')
case class MonEntier (classe: Int, valeur: Int)
val w = stream.map ( { x => x.toInt } ).map({ x => MonEntier (x % 2, x)} )
              .keyBy("classe")
              .window(TumblingProcessingTimeWindows.of(Time.seconds(5)))
              .fold("Liste: ") { (acc, v) => acc + " | " + v.valeur }
              .print()
senv.execute(" ")
```

Le niveau de parallélisme autorisé par ce fenêtrage n'est que de deux.