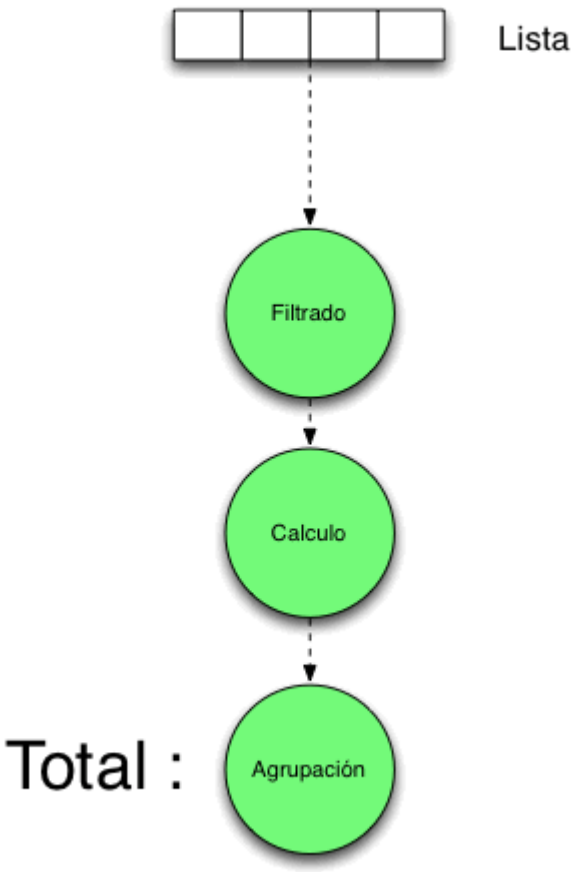
Poco a poco vamos conociendo como utilizar las expresiones lambda y sacarles partido. Hay en algunas ocasiones que nos pueden resultar especialmente útiles. Por ejemplo nos podemos encontrar con listas de elementos que tienen un tamaño muy grande y tenemos la necesidad de procesarlas realizando varias operaciones con expresiones lambda. Operaciones que pueden suponer desde filtrados, pasando por cálculos hasta agrupaciones. Veamos un ejemplo sencillo:

```
package com.arquitecturajava.ejemplo2;

import java.util.Arrays;
import java.util.List;

public class Principal {

public static void main(String[] args) {

List lista= Arrays.asList(100,200,300,100,500);

double total=lista.stream()
.filter(elemento->(elemento>200))
.mapToDouble(elemento->elemento*1.21).sum();

System.out.println(total);
}

}
```

En este caso creamos una lista de 5 números (en un caso real serian miles) y filtramos aquellos que sean mayores de 200. Una vez filtrados realizamos el cálculo de sumarles el

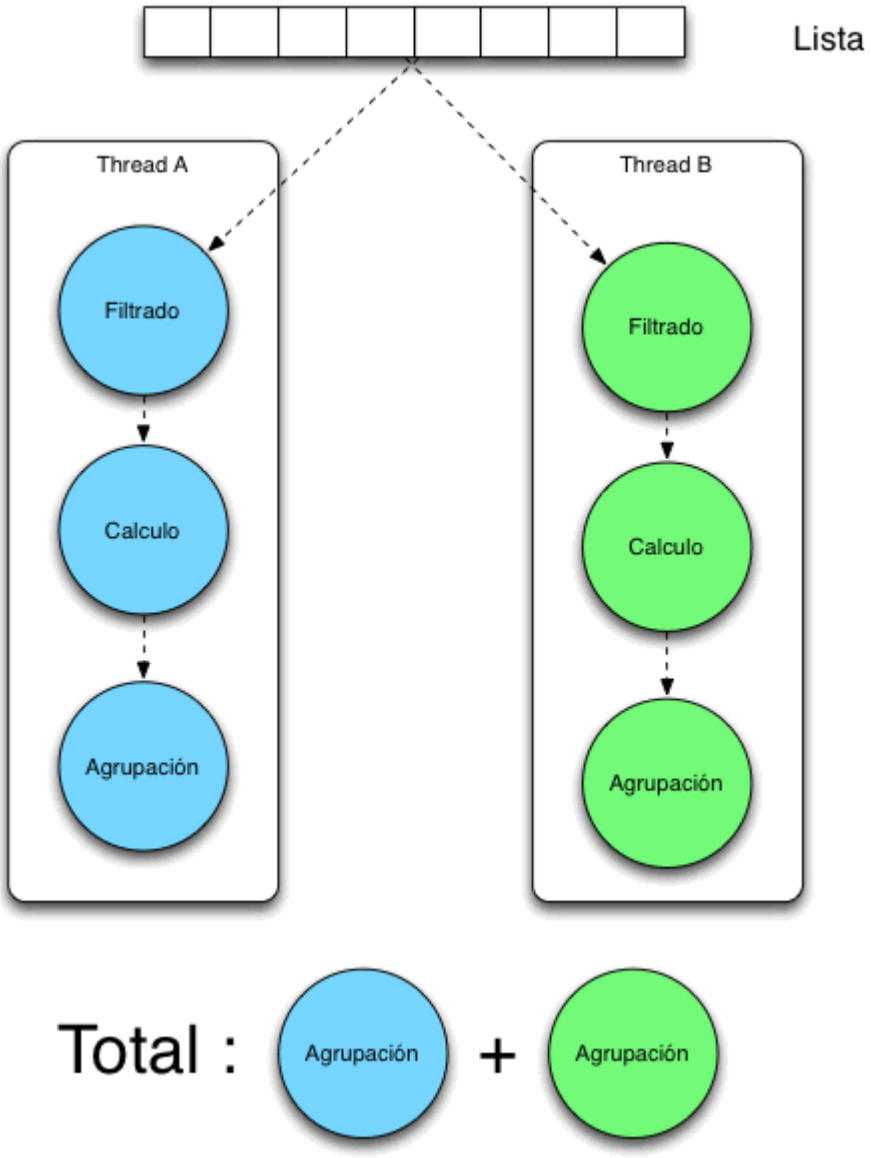
iva. Finalmente agrupamos los datos y los imprimimos por pantalla



.

Expresiones Lambda y Concurrencia

Sin embargo existen situaciones en la que la lista inicial puede tener un tamaño mucho más grande y es ahí nos vendría bien separar estas tareas de tal forma que puedan ser ejecutadas por Threads distintos. En nuestro caso no hay mucho problema porque la tarea de filtrado se puede dividir y que un Thread ordene una parte de la lista y el otro la otra parte. Con el resto de operaciones sucede parecido. Asi pues podemos reorganizar la tarea de la siguiente forma.



De esta forma si tenemos una maquina con varios cores la tarea puede terminar mucho antes ya que el trabajo esta siendo dividido. Normalmente programar este tipo de situaciones suele resultar complejo. Sin embargo a traves de expresiones lambda es trivial ya que estas soportan la capacidad de indicar que una tarea se puede realizar en paralelo.

```
package com.arquitecturajava.ejemplo2;

import java.util.Arrays;
import java.util.List;

public class Principal {

public static void main(String[] args) {

List lista= Arrays.asList(100,200,300,100,500);

double total=lista.stream()
.parallel()
.filter(elemento->(elemento>200))
.mapToDouble(elemento->elemento*1.21).sum();

System.out.println(total);
}

}
```

La tarea se ejecutará de la misma forma solo que en este caso Java realizará una partición del trabajo (hemos añadido el método parallel). No siempre ejecutar en paralelo es más rápido que ejecutar en serie. Todo depende del numero de elementos que tengamos que gestionar y de los cores que nuestro equipo tenga.

Otros Artículos relacionados: Expresiones Lambda ,Lambda ForEach, Streams