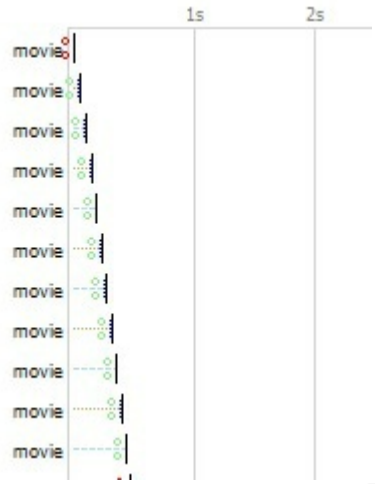
Requêtes synchrones VS requêtes asynchrone

I. Requêtes Synchrones

Exemple simpliste d'une requête interrogeant le serveur end point d'allociné

```
WebRequest request = WebRequest.Create("http://api.allocine.fr/rest/v3/movie?partner=Q
WebResponse response = request.GetResponse();
Stream dataStream = response.GetResponseStream();
StreamReader reader = new StreamReader(dataStream);
string responseFromServer = reader.ReadToEnd();
Console.WriteLine(responseFromServer);
reader.Close();
response.Close();
```

Si l'on exécute ces requêtes de manière répétitive en simultané en sondant les trames avec le programme avec `fiddler2`, on obtient ceci :



Timeline de chaque requête capturé sur `Fiddler2`

Les requêtes s'exécutent bien les unes après les autres. C'est à dire que le temps d'exécution du programme est égal à la somme du temps de réponse de chaque requêtes.

II. Requêtes Asynchrones

Exemple avec la classe `WebRequestAsync` permettant d'exécuter une requête de manière asynchrone

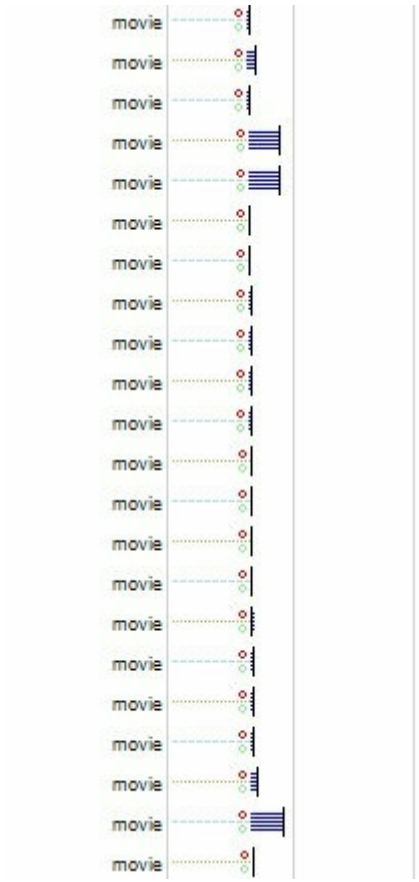
```
[Author("Julien LELEU", version = 1.0)]
public class WebRequestAsync
{
  private HttpWebRequest webRequest;

  public WebRequestAsync(string url)
  {
    webRequest = (HttpWebRequest)WebRequest.Create(url);
  }

  public void StartWebRequest()
  {
    try
    {
      webRequest.BeginGetResponse(
        new AsyncCallback(DisplayWebResponse), webRequest
        );
    }
    catch(Exception e)
    {
      Console.WriteLine(e);
    }
  }

  private void DisplayWebResponse(IAsyncResult result)
  {
    Stream dataStream = null;
    StreamReader reader = null;
    HttpWebResponse response = null;
    try
    {
      response = (
        result.AsyncState as HttpWebRequest
      ).EndGetResponse(result) as HttpWebResponse;
      dataStream = response.GetResponseStream();
      reader = new StreamReader(dataStream);
      string responseFromServer = reader.ReadToEnd();
      Console.WriteLine(m.Movie.Title);
    }
    catch (Exception e)
    {
      Console.WriteLine(e);
    }
    finally
    {
      reader.Close();
      response.Close();
    }
  }
}
```

```
static void Main(string[] args)
{
  for (int i = 0; i < 100; i++)
  {
    new WebRequestAsync("
    http://api.allocine.fr/rest/v3/movie?partner=QXJjaGltZWQ&code=61282&format=json
    ").StartWebRequest();
  }
    Console.ReadKey();
}
```



Timeline de chaque requête capturé sur `Fiddler2`

On observe donc bien que chaque requête est traitée en parallèle. Lorsqu'une requête a un temps de réponse plus élevé que les autres, elle n'impacte pas leur date de lancement et par conséquent le temps d'exécution du programme.