SERIAL VS PARALLEL EXECUTION AND DOWNLOAD

Introduction 🎯

Due to my personal smartphone not being an Android, I tested my SuperDuperDownload application inside the Android Studio emulator. In particular, I virtualized a Google Pixel 3A on my Apple Macbook Air M1, with 8 core (4 performance cores and 4 efficiency cores), a 7-core GPU, a 16-

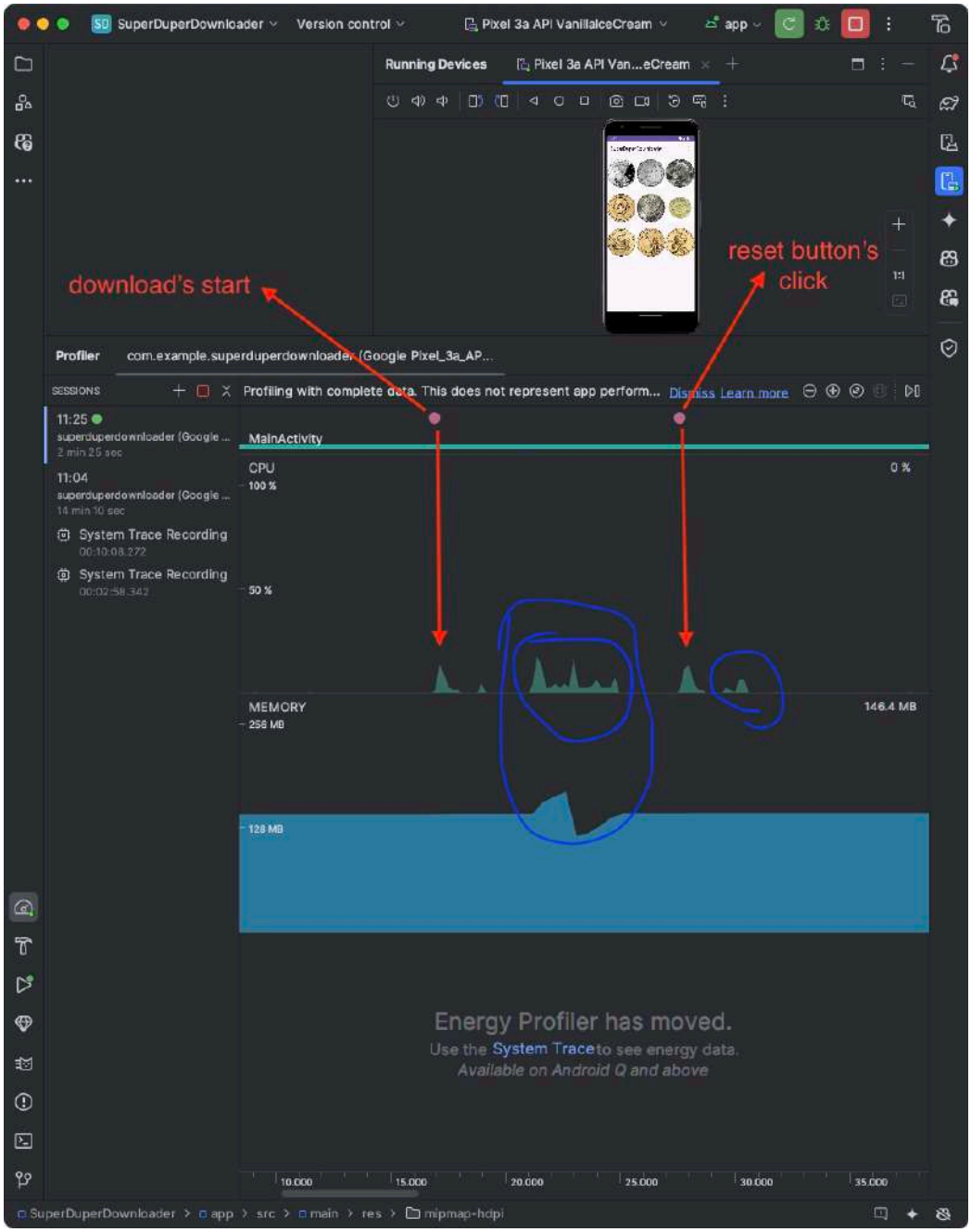
core neural engine and 8GB of unified memory (RAM, etc). I encountered some technical difficulties, e.g. crashes due to memory management or performance worsening, but, in the end, the project was completely doable.
Also, it is worth mentioning that I changed slightly the app behavior: instead of having already all coin images and rotate them when each download is done, I have left the table layout (initially) empty, ready to be filled with the coin pictures one by one, once downloaded.

It is crucial to well organize the thread's tasks, otherwise the application could not work as expected, or worse: crash. Secondly, I've noticed, as expected, that the parallel execution of the nines download is faster than the sequential one; although it is not a big difference (I tested the downloads with an optical fiber connection, so the download was relatively fas, regardless of the execution's type).

In the following sections, I will describe both parallel and sequential execution of the app and the two relative cases, completing the nine downloads.
In the end, I will draw my conclusions.

Sequential Scenario 🧬

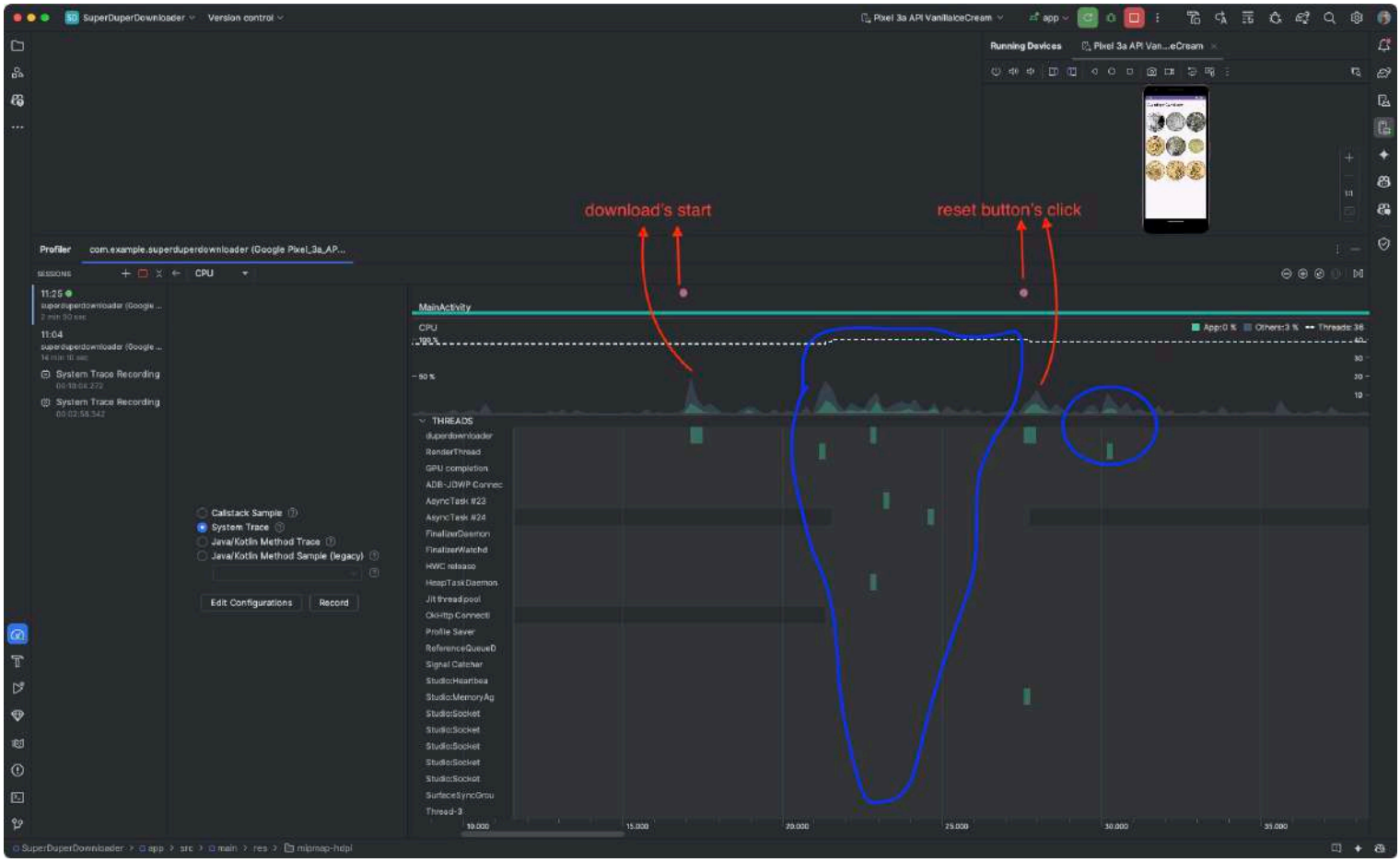
The sequential execution, in general, is almost always slower. On the other hand it is simpler to manage and less prone to errors. Let us see the resulting graphs of the application.



As we can see, there are numerous peaks of CPU usage. In detail:

- We have one as soon as we click on the “Serial Download Images” menù item and the download starts;
- The middle blue circle highlights the various peaks for each image download;
- The outer blue circle highlights also the memory usage and its changes;
- The red arrows on the right highlight the event of when the item “Reset Images” is clicked;
- Lastly, the rightmost blue circle highlights the actual action of resetting the images’s table layout.

For confirmation, let us see the graph of the energy usage profiler.

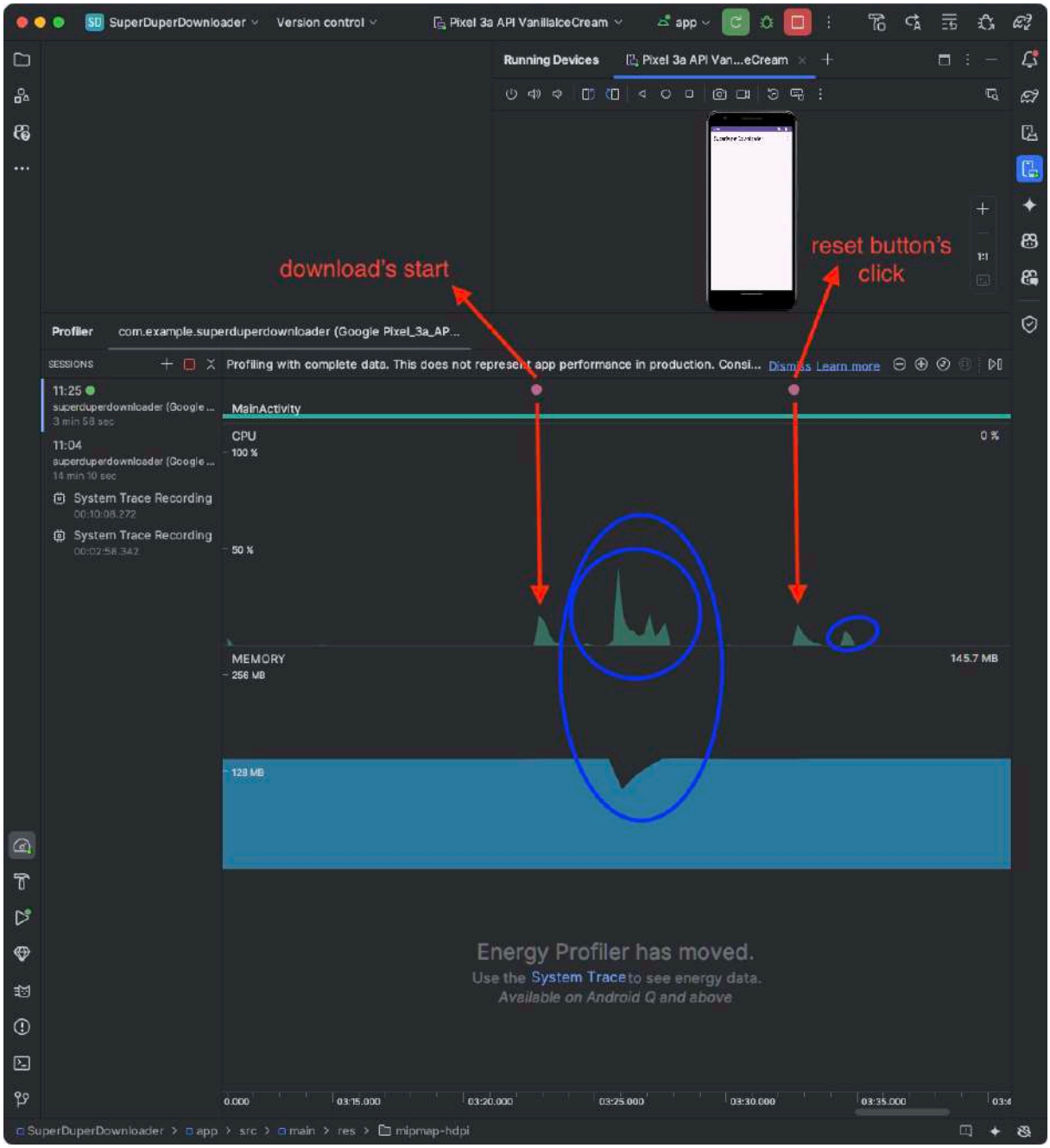


Indeed, the energy consumption's peaks match the CPU usage peaks and the relatives threads. Now, let us move to the parallel case, whom, as we will see will take less time to be finished.

Parallel Scenario 🛤️

The parallel execution, in general, is almost always better and faster, at the price of highest attention due to the threads' managing. On the other hand, the choice fall on the developer(s), according to the app needs.

Let us see the resulting graphs of the application.

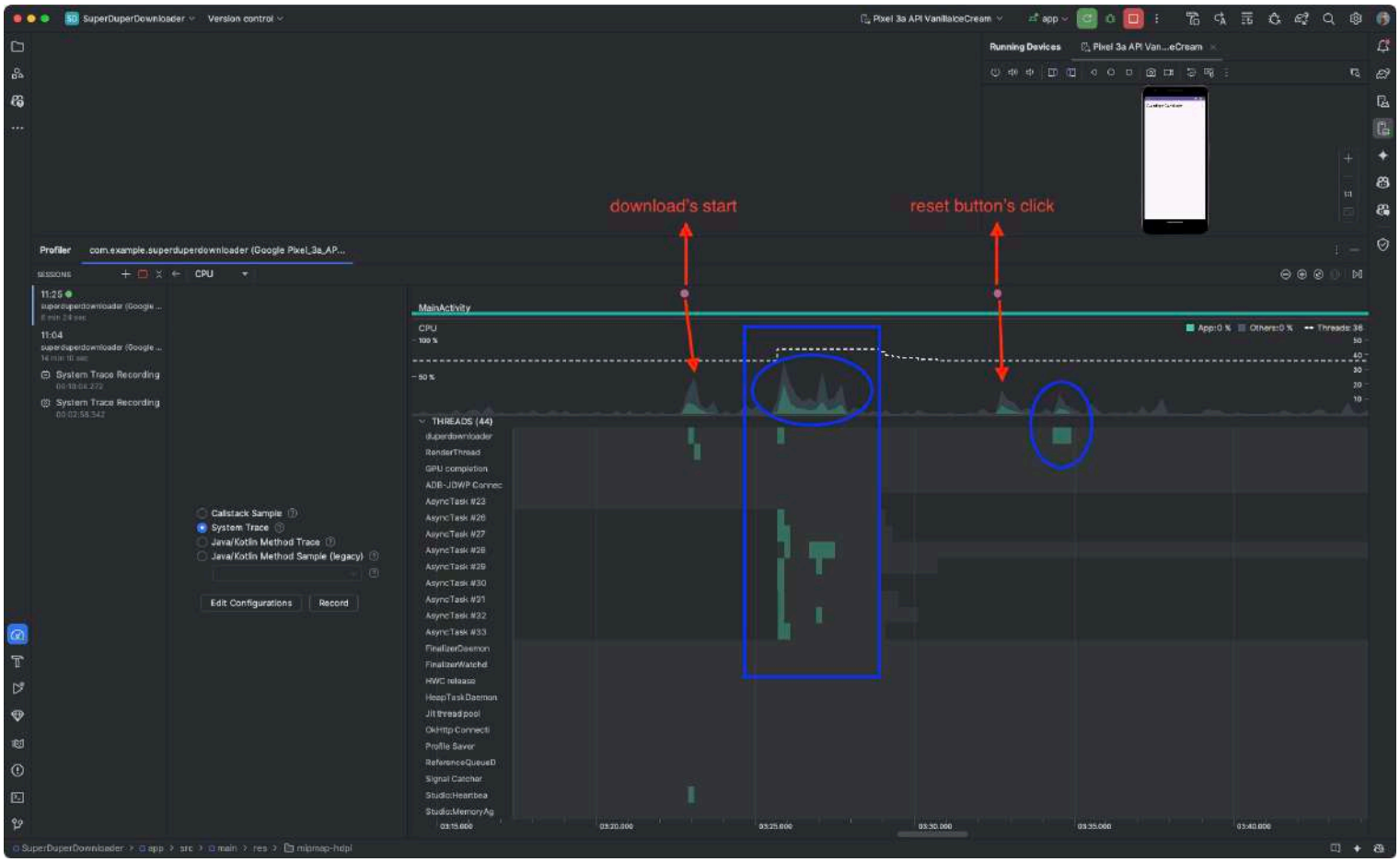


Its obvious that the execution time (circled in blue) is shorter. Let us analyse the graph:

- We have the same initial peak, highlighted by the leftmost red arrow, of the sequential run as soon as we click on the "Parallel Download Images" menù item and the download starts;
- The middle blue circle highlights a slightly bigger peak at the start of the download, with a flattened green graph after;
    - **Most importantly, the overall execution is visibly shorter**;

- The outer blue circle highlights also the memory usage and its changes;
  - **Overall, the memory is used for far less time**;
- The red arrows on the right highlight the event of when the item “Reset Images” is clicked;
- Lastly, the rightmost blue circle highlights the actual action of resetting the images’s table layout.

For confirmation, let us see the graph of the energy usage profiler.



Indeed, the energy consumption’s peaks match the CPU usage peaks and the relatives threads.
In this particular scenario of this particular app, the parallel execution (and then download) is clearly the better choice.

Conclusions END

It is evident that the SuperDuperDownloader app performs better using a parallel approach, whom leads to a significant reduction in processing time, but at the cost of more CPU usage in a short window of time. Moreover, the download time is clearly shorter using a parallel approach.
Although, the latter requires a much more careful management of resources to avoid overloading or crashing the system. In the end, as mentioned before, the choice between the two is the developer’s and it depends on the particular case, on a particular scenario, etc.