Running scripts from CLI in PHP frameworks & Parallel Processing

Parallel Processing is something that we all might be aware of, but how and where to use it more effectively is the bigger question. I also never thought of that but while working on my current project which is in Kohana PHP framework I faced couple of challenges where my goal was to create 6 interior images from one product image in 3 different formats. It involved the following steps:

- First I had to take the product image, re-size and merge with a shadow background.
- Then take the merged image and again merge on the wall of a interior house image which also involved lot of calculation for determining on where to place the image which I am not going to explain here.
- And finally since I had 6 different type of interior images where I had to place the shadow merged image and create three different sized versions of each final interior image such as zoomed, original and thumbnail, so if you calculate then with all these image processing I had to create 18 images out of one image.

Now coming back to my first problem again, the code is written in such a way as to use framework's base classes and their methods. Since most of the frameworks are setup in such a way if we would like to use their classes or methods then we have to run them from the browser so it can include all the required class files and redirect the user to the controllers and its method/ actions and work properly.

But in my case, since I had about 3000 images stored for which I had to generate the above mentioned interior images,it became a total of about 54000 images to be processed by the script. This is really a lot and if we had to run from browser then we always run the risk of script timeout or browser becoming non-responsive!! So while looking on web I found that we can run these kind of scripts from command line which can use the framework's structure to include the class files and its methods.

In Kohana it can be done like:

```
$ cd <to root location of your codebase>
$ sudo php index.php <controller name/method name if required(or default to index)>
//To avoid permission issues I used sudo which will run the script as a root user
```

***Note: All the examples shown are based on Linux environment but if you want to try with Windows or some other environment then you have to modify the program paths accordingly.**

So here my main point is, as we have option for running script from CLI for Kohana we should definitely have for other type of PHP frameworks such as cakePHP, zend, Symphony etc.. which we can find and try as needed.

Now the next problem was with creating such a huge number of images with one process was taking forever, so referring to this post I tried to split this single process in multiple parallel processes where each process will be assigned a specific number of images to process and that way all the processes can be started at a time similar to a "**fake fork"**.

I modified the script a little bit to look for any passed argument/parameter first. For the first time when I ran the script I didn't want to pass any argument so all the parallel processes can be started/initiated by this script so modified in such a way if there is no parameters passed then it will count the number of images to be processed and if I want to assign 500 images to each process then it will start 6 processes (as in my case I have 3000 images) by sending the process

number as an argument to each newly started process to help in querying the db for finding the appropriate number and set of images to be processed.
Here is the code snippet which might help understanding the above process better:

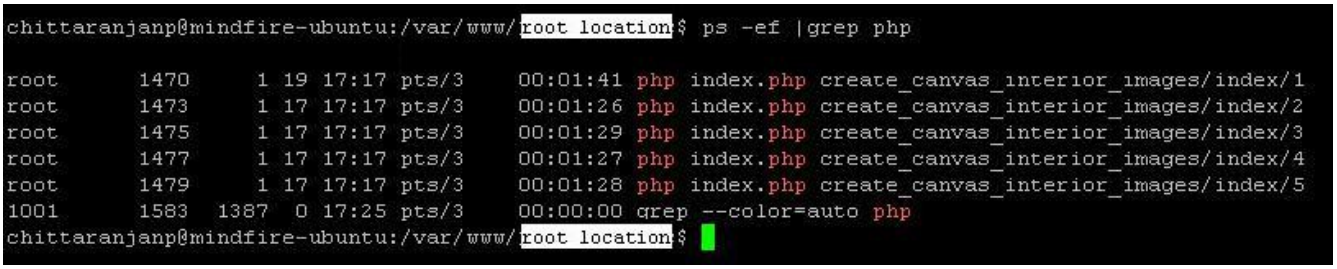
```
$productIds     = [Some DB query to get all the product Ids that we want to process];
$processLimit   = 500; //number of images or records to process per instance of script
$totalProcesses = round(count($productIds) / $processLimit);

for ($i = 1; $i <= $totalProcesses; $i++) {
        system('php index.php create_canvas_interior_images/index/'.$i.' >/dev/null >&- >/dev/null &');
        echo "Image creation process # $i has started successfully\n\n";
}
```

As in the above code you can see I am getting the number ***productIds*** and based on the ***processLimit*** I am finding how many number of processes need to be initiated and looping through the total number of processes and initiating the new processed by sending the process number such as 1, 2, 3... as an parameter to started processes

Here is the command that I tried from command line as shown in the image below with the output saying how many numbers of processes initiated as echoed in the script on the CLI:

and to check if the processes are initiated properly or not I ran the following Linux command as shown in the image with output:



This way based on the passed argument/parameter I fetched that particular set of records from db and all 6 processes started creating their assigned images all at a time in a parallel manner and at the end I am writing the output for success or failure of each image processing to DB with the process number so I can track them later.

This approach helped me finding a solution to achieve my goal and making the processing faster for such a huge number of images and was appreciated by my client so thought to share with you all so it can be useful in any similar scenario/situation.

But note that if I had a very large number of images then I would think before doing this as it

may start several parallel processes which might overload server and bring it down. So we need to think about what number of processes will still work and achieve best output.