Issue: for the old web pages, when one client log in and choose the part (pollen/seed) he is going to compute. He is redirected to the “driver.php” page, the page executes several time-consuming programs (java –jar \*.jar et. al) sequentially using the blocking php function “exec”. Thus the client cannot return to the “choose.php” page to pick up additional computations on other positions in the map. Furthermore, other clients cannot access the driver.php either.

Solutions: several strategies are used to solve this problem. The strategy adopted is listed as the first solution below. The basic model consists of 3 entities: the front page “driver.php”, the database table and the daemon process “php.exe” executing “daemon.php”. The strategies differ only in how the daemon process is launched.



Solution 1. Driver.php is modified to execute only one task: when a client logs into the system. It wraps all the parameters input by the client into a record and inserts this record into the table. On one hand, all the input from the users form a queue as a table in the database, thus the background daemon process can process the record one by one. On the other hand, it increases the responsiveness of the web site. The time needed is just to insert a record into the table. The page needn’t wait for the completion of the task to return. Thus the page can return back and the client can input another set of parameters if he would like. At the same time, the other clients can log into the website as well. So the blocking problem is solved. The code is conventional, just some php SQL insert commands.

The daemon is running as a background process by the task scheduler of the windows system; it is scheduled to restart every one hour in case the daemon process is killed for unexpected reasons. Only one instance of this php script is allowed. After a record is processed, it is deleted from the table, and the following record is retrieved. The configuration steps are as follows for Windows 2008 server:

1. Control panel->administrative tools->task scheduler
2. Create Task
3. In the “General” tab
* check “run whether user is logged on or not”
* check “hidden”,
1. In the “Triggers” tab,
* select “on a schedule” in “begin the task”
* check “one time” in “ settings”
* check “repeat task every” 1 hour for a duration of indefinitely
* Check “enabled”;
1. In the “action” tab,
* Select “start a program”
* In “program/script” set “C:\php\php.exe”
* In “Add arguments”, set “daemon.php >> daemon\_errors.log”, log file is optional
* In “start in” set the folder where “daemon.php” resides.
1. In the “settings” tab:
* Check “allow task to be run on demand”
* Check “run task as soon as possible after a scheduled start is missed”
* At the bottom choose “Do not start a new instance”.

Solution 2:

Another strategy is tried in which the daemon is launched by user rather than being scheduled by task scheduler. This strategy didn’t succeed on windows, but might work on Linux with Apache. The general idea is to replace the blocking function “exec” with nonblocking function “pclose(popen(“start /b php daemon.php”)”. But it always dies after running for an amount of time, I still don’t figure out what is happening here.