گزارشی از برنامه‌نویسی موازی در پی‌اچ‌پی

نویسنده : علا عالم فلکی (ala.falaki@gmail.com)

امروزه با توجه به افزایش روز افزون ظرفیت سخت‌افزارهای کامپیوتر و به وجود آمدن پردازنده‌های چند هسته‌ای و کامپیوترهای سرور با چند پردازنده مجزا، نوشتن برنامه‌هایی برای استفاده از این ظرفیت به عهده ما می‌باشد. با این حال به دلیل نبود قابلیت برنامه‌نویسی موازی در هسته پی‌اچ‌پی شیوه نوشتن برنامه‌های موازی در این زبان به نوعی چالش برانگیز است. با این وجود می‌توان با استفاده از کتابخانه‌های نوشته شده قابلیت برنامه‌نویسی موازی را در پی‌اچ‌پی شبیه‌سازی نمود. در این مقاله ابتدا مفهوم کلی برنامه نویسی‌موازی و دلایل استفاده از آن بیان شده و سپس شیوه‌های متفاوت شبیه‌سازی آن بررسی می‌شود.

کلمات کلیدی: برنامه‌نویسی موازی، چندنخی، چندفرآیندی، توزیع فرآیندها

۱. مقدمه:

با قدرتمند شدن سرورها و موجود بودن پردازنده‌های چند هسته‌ای و یا نصب چندین پردازنده بر روی سرورها یکی از مهم‌ترین وظایف توسعه‌دهندگان نرم‌افزار استفاده از ظرفیت کامل سخت‌افزارهای سیستم است تا بتوانند کارایی نرم‌افزار خود را حداکثر کنند، حداکثرکردن کارایی در این مقاله به معنی حداقل کردن زمان اجرای برنامه می‌باشد. لازمه رسیدن به این هدف استفاده از تکنیک‌های برنامه‌نویسی موازی است. به این معنی که بخش‌هایی از برنامه که قابلیت اجرای همزمان دارند را در فرآیندهایی به صورت همزمان اجرا نماییم، برای مثال اگر فرآیند ثبت‌نام در یک سایت شامل ارسال پست‌الکترونیکی تایید عضویت و همچنین ارسال پیامک خوش‌آمدگویی باشد، به دلیل وابسته نبودن هر مرحله از عملیات به مرحله قبل، می‌توان این مراحل را به طور همزمان انجام داد. و یا اگر برای ارسال تعداد بالای پست‌الکترونیکی بخواهیم از برنامه‌نویسی موازی استفاده نماییم، می‌توانیم به جای ارسال یک به یک پیام‌ها از طریق یک فرآیند ترتیبی، این فرآیند را به طور موازی انجام دهیم تا زمان کمتری صرف شود. (توجه داشته باشید که بعضی از الگوریتم‌ها یا برنامه‌ها ماهیت ترتیبی دارند و برای ادامه فعالیت، جواب هر مرحله وابسته به جواب مرحله قبل می‌باشد. برای درک بهتر قضیه مثال ثبت‌نام را در نظر بگیرید؛ در این مثال ارسال پست‌الکترونیکی وابسته به موفق بودن ثبت‌نام و ثبت موفق اطلاعات در پایگاه‌داده است، پس این مراحل را نمی‌توان به صورت موازی برنامه‌نویسی نمود.)

وب‌سرور آپاچی که وظیفه ارسال درخواست کاربران به پی‌اچ‌پی را دارد (به نوعی اجرا کننده پی‌اچ‌پی به حساب می‌آید) خود از قابلیت‌های چندنخی[۱] و چندفرآیندی[۲]پشتیبانی می‌کند[۱]. پی‌اچ‌پی نیز بعد از آپدیت رسمی ۵.۲ از قابلیت چندنخی پشتیبانی می‌کند. (اگر این قابلیت در پی‌اچ‌پی پیشبینی نمیشد همواره مشکلاتی در زمان دسترسی به داده‌های مشترک به وجود می‌آمد، برای مثال بن‌بست[۳]، ضعیت رقابتی[۴] و...)

با توجه به پیشبینی این امکانات متاسفانه امکان برنامه‌نویسی موازی در هسته پی‌اچ‌پی پیشبینی نشده است، دلیل نبود امکان برنامه‌نویسی موازی در این زبان این است که پی‌اچ‌پی یک زبان اسکریپتی بوده و توسط مفسر[۵] (به جای

۱ Multi-threading
۲ Multi-Processing
۳ Death-Lock
۴ Race Condition
۵ interpreter

کامپایلر) در محیط وب‌سرویس‌ها (برای مثال آپاچی) اجرا می‌شود. در ابتدا این زبان برای تولید تگ‌های HTML استفاده می‌شد اما با گذشت زمان و گسترش این زبان و افزایش محبوبیت آن، انتظارات از این زبان بیشتر شد و در نتیجه قابلیت‌های این زبان گسترش یافت.

برای دستیابی به هدف اجرای همزمان بخش‌های مختلف برنامه، روش‌های مختلفی پیشنهاد می‌شود که هرکدام از این روش‌ها مزایا و معایب خود را دارد. هدف از نگارش این مقاله آموزش این روش‌ها نبوده و هدف تنها معرفی این روش‌ها است.

در بخش‌های بعدی این مقاله با روش‌های **چندنخی**، **چندفرآیندی** و **توزیع فرآیندها**[۶] آشنا می‌شویم.

۲. چندنخی:

یک نخ به مجموعه‌ای از دستورالعمل‌ها[۷] گفته می‌شود که در دل یک فرآیند اجرا می‌شوند و دارای اشاره‌گر دستور[۸]، ثبات[۹] و پشته [۱۰] مختص به خود می‌باشند . فضای آدرس[۱۱] نخ‌های موجود در یک فرآیند یکسان هستند، به همین دلیل نخ‌ها به راحتی می‌توانند از داده‌های مشترک استفاده نمایند اما این یکسان بودن فضای آدرس، در صورت بروز خطا در یک نخ، باعث مختل شدن ادامه کل فرآیند [۱۲] (سایر نخ‌های موجود در آن فرآیند) می‌شود.

به نرم‌افزاری که توسط بیش از یک نخ اجرا شود، نرم‌افزار چندنخی گفته می‌شود. مزیت استفاده از روش چندنخی این هست که برای موازی‌سازی دستورات نیاز به اجرای چند کپی[۱۳] در حال اجرا (چند فرآیند برای اجرای یک برنامه) از برنامه نیست . عیب این روش در آزمایشی بودن این کتابخانه هست. (درحال حاضر نسخه پایدار آن عرضه نشده است.)

برای استفاده از قابلیت چندنخی در پی‌اچ‌پی باید از نسخه‌ای که نخ امنی[۱۴] در آن فعال است، استفاده کرد.

نسخه‌ی نخ‌امن نسخه‌ای است که که مشکلات احتمالی دسترسی به داده‌های مشترک را حل کرده است.

همانطور که گفته شد ابزار چند نخی در پی‌اچ‌پی پیشبینی نشده است. برای اضافه کردن این قابلیت می‌توان از کتابخانه‌ای به اسم pthreads استفاده کرد.

کتابخانه Pthreads شامل تمام ابزار مورد نیاز برای یک برنامه‌نویس است تا بتواند فرآیند چندنخی را پیاده‌سازی نماید. از قابلیت‌های این کتابخانه می‌توان به پشتیبانی کامل آن از شی‌گرایی[۱۵] اشاره کرد . در زیر مثالی از نوشتن برنامه چندنخی می‌بینید .[۲]

۶ Distributed Parallel Processing
۷ Instructions
۸ Instruction Pointer
۹ Register
۱۰ Stack
۱۱ Address Space
۱۲ Process
۱۳ Instance
۱۴ Z.T.S (zend thread safe)
۱۵ Object Oriented

```
<?php
class ChildThread extends Thread {
        public $data;
        public function run() {
          /* Do some work */
          $this->data = 'result';
        }
}
$thread = new ChildThread();
if ($thread->start()) {
        // Do some work here, while already doing other work in the child thread.
        // wait until thread is finished
        $thread->join();
        // we can now even access $thread->data
}
```

در مثال بالا وظیفه مورد نظر را بین دو نخ تقسیم می‌کنیم(نخ اصلی برنامه+نخ ایجاد شده توسط کلاس ChildThread) و در نهایت بعد از اتمام کار نخ‌ها، جواب نهایی را در نخ اول پردازش می‌کنیم. اگر در زمان اجرای برنامه نیاز به برقراری ارتباط(انتقال داده) میان نخ‌های مختلف باشد، چندنخی انتخاب بهتری نسبت به چندفرآیندی خواهد بود.(مثال‌های بیشتر در [۳])

۳. چندفرآیندی:

فرآیند یک نمونه از برنامه در حال اجرا می‌باشد که شامل کدهای برنامه و فعالیت‌های کنونی آن است و با توجه به نوع سیستم عامل می‌تواند از چند نخ تشکیل شده باشد.[۴] در پی‌اچ‌پی فرآیند اصلی توانایی تولید فرآیندهای فرزند را دارد، به نحوی که هر فرآیند تولید شده به طور کاملا مستقل از فرآیند اصلی اجرا می‌شود و این استقلال باعث می‌شود ارتباط(انتقال داده) میان فرآیندهای در حال اجرا بسیار سخت باشد.

به نرم‌افزاری که توسط چند فرآیند مجزا اجرا می‌شود برنامه چندفرآیندی گفته می‌شود. مزیت این روش موجود بودن توابع آماده در هسته اصلی پی‌اچ‌پی برای پیاده‌سازی این روش است و عیب آن هزینه بر بودن تولید فرآیندهای جدید و تعویض کردن فرآیندهای در حال اجرا[۱۶] توسط سیستم‌عامل است. روش‌های مختلفی برای تولید فرآیندهای جدید در پی‌اچ‌پی پیشبینی شده است، در ادامه به بررسی برخی از این روش‌ها میپردازیم.

۳-۱. FORK :

با استفاده از تابع pcntl_fork میتوانیم فرآیندهای فرزندی ایجاد نماییم که تنها تفاوت آنها با فرآیند اصلی در شماره فرآیند (PID) و شماره فرآیند والد(PPID) (برای مشخص کردن فرآیندی که آغازگر آن بوده) است.[۵] برای درک بهتر به مثال زیر توجه کنید:[۶]

```php
<?php
$workload = "some work load";
$processId = pcntlfork();
if ($processId < 0){
        die('Fork failed!');
} else if ($processId == 0) {
        // child starts working here
        trim($workload);
} else {
        // do other stuff in parent
        // and parent waits for child
        pcntlwait($status);
}
```

در این مثال یک فرآیند فرزند تولید میکنیم و بعد از آن می‌توانیم ادامه کار را در فرآیند والد انجام داده و منتظر پاسخ از فرآیند فرزند باشیم. برای مشاهده مثال‌های بیشتر می‌توانید به [۵] مراجعه کنید.

۳-۲. Execute Command:

با استفاده از ۳ تابع زیر می‌توانیم دستورات پی‌اچ‌پی را به وسیله فرآیندهای مجزا اجرا نماییم.

- Exec : تنها آخرین خط نتیجه حاصل از فرآیند جدید را برمیگرداند.
- exec_shell : تمام نتایج حاصل از فرآیند جدید را برمیگرداند.(در قالب نوع داده‌ای رشته[۱۷])
- passthru : تمام نتایج خام حاصل از فرآیند جدید را برمیگرداند.(برای مثال در قالب نوع داده‌ای باینری[۱۸])

توجه داشته باشید اجرای دستورات پی‌اچ‌پی توسط این توابع، عملیاتی پر هزینه است. به خصوص در زمان بالا رفتن تعداد فرآیندها بهره‌وری سرور به طرز محسوسی کاهش پیدا خواهد کرد. (ایجاد فرآیندها و تعویض فرآیندهای در حال اجرا توسط سیستم‌عامل هزینه زیادی دارد.)

۳-۳. Piping:

مشکل اصلی دو روش گفته شده، عدم امکان برقراری ارتباط[۱۹] (انتقال داده) میان دو فرآیند در زمان اجرا است. برای رفع این مشکل روشی به اسمpiping معرفی شده است که با استفاده از ورودی /خروجی استاندارد(STDIN/STDOUT) و تابع proc_open (که یک فرآیند جدید ایجاد کرده و آی‌دی[۲۰] آن را برمیگرداند) این ارتباط را برقرار می‌کند . در زیر مثالی از این روش میبینید.[۶]

۱۷ String data type
۱۸ Binary data type
۱۹ Communicate
۲۰ Handle

```
// pipeclient.php, uses procopen() function to create the
// worker process, and send instructions to it
// 0 => stdin for worker    1 => stdout for worker

$descriptorspec = array( 0 => array("pipe", "r"),  1 => array("pipe", "w")  );

$worker = procopen("php pipeworker.php", $descriptorspec, $pipes);
if ($worker) {
        fwrite($pipes[0], "hello");
        while (!feof($pipes[1])) {
                echo fgets($pipes[1]). "\n";
        }
proc_close($worker);
}

// pipe_worker.php, all it does is to read instructions
// from STDIN, and write response to STDOUT
$line = fread(STDIN,4096);
fwrite(STDOUT, "$line world");
```

۴. توزیع فرآیندها:

روش‌هایی که پیش از این گفته شد برای وب‌سایت‌هایی با یک سرور مناسب می‌باشند، برای افزایش بهره‌وری این سایت‌ها می‌توان تعداد CPUهای سرور را افزایش داد. با این حال بعد از حجیم شدن بیش از حد سایت، روش‌های گفته شده جوابگو نخواهند بود، پس نیاز به بیش از یک سرور خواهیم داشت، و در این شرایط برای افزایش بهره‌وری می‌توان از توزیع فرآیندها بین چند سرور استفاده کرد.

برای برقراری ارتباط بین فرآیندها می‌توان از برنامه‌نویسی سوکت[۲۱] استفاده کرد، در عین ساده بودن برنامه‌نویسی سوکت باید توجه داشت که شرایط خاص مورد توجه قرار بگیرند. برای مثال در صورت قطع شدن ارتباط میان دو سرور چه اتفاقی خواهد افتاد ؟ اگر فرآیندی که وظیفه پردازش را دارد در حین پردازش از دسترس خارج شود چه اتفاقی می‌افتد؟ اگر فرآیندی که وظیفه پردازش را دارد موجود نباشد چه اتفاقی می‌افتد؟ و شرایط مشابه زیادی که باید در زمان نوشتن کد به آنها توجه نماییم، در این شرایط می‌توان از کتابخانه‌های[۲۲] موجود استفاده کرد . (برای مثال Gearman و ZeroMQ)

ماهیت کتابخانه‌های ذکر شده از بین بردن پیچیدگی‌های برنامه‌نویسی سوکت می‌باشد، به همین دلیل تکنیک‌های برنامه‌نویسی این کتابخانه‌ها روان و ساده هست. تنها فرآیند نصب این کتابخانه‌ها اندکی دشوار است، با این حال ارزش نصب و استفاده را دارند. (برای مشاهده مثال‌هایی از کتابخانه ZeroMQ به [۷] و مثال‌هایی از کتابخانه Gearman به [۸] مراجعه نمایید.)

۲۱ Socket Programming

۲۲ Libraries

۵. منابع:

[۱] http://httpd.apache.org/docs/2.2/mod/

[۲] http://www.mullie.eu/parallel-processing-multi-tasking-php/

[۳] https://github.com/krakjoe/pthreads/tree/master/examples

[۴] http://en.wikipedia.org/wiki/Process_%28computing%29

[۵] http://php.net/manual/en/function.pcntl-fork.php

[۶] http://code.hootsuite.com/parallel-processing-task-distribution-with-php/

[۷] https://github.com/imatix/zguide/tree/master/examples/PHP

[۸] http://gearman.org/examples/