PHPのsessionとflockとか。

坂本昌彦
id:msakamoto-sf
2008-02-28
第31回PHP勉強会発表資料
sakamoto-gsyc-3s@glamenv-septzen.net

誰？

81世代。
PHP, Javaプログラマー
株式会社エヌ・エス・ディ所属
事務所は立川だけど新宿中心に
ぐるぐる出向したりしてます。

お仕事：Javaでソケット通信とか。
PHP：趣味。会社としては殆どやってない。

アジェンダ

だらだら喋ることになりそうなのであまりあてにできませんが、大まかに次の三つ。

1. 「flock() = アドバイザリ・ロック」って何？
2. PHPのsession機構ってflock()使ってる？
3. 自前でセッション保存関数を用意する場合の注意事項って？

それでは始まります。

0. そもそもの始まり。

Xhwlayという自作ライブラリ

「イベント駆動指向ステートフルページフロー実行エンジン」とかいう長ったらしい名前ですが、この中でセッション情報的なデータをファイルに保存する処理をある日の事実装してました。

「PHPって楽だな～」

・・・と、
file()関数やら
file_{put|get}_contents関数やら
で実装してました。

・・・が。

ふと、

「これって同時アクセスされた時どうなるんだ？」

という疑念が・・・。

CGI時代はアクセスカウンタ

要するに、PerlCGI時代全盛期は

「アクセスカウンタのデータファイルの読み書き」

でさんざん論議された、枯れた内容だった訳です。

最初のコード(イメージ)

```
<?php
$data = file_get_contents($filename);
$cnt = (integer)trim($data);
$cnt++;
file_put_contents($filename, $data);
...
```

・・・駄目駄目です。

「CGIやDBのロックと同時実行制御」
http://jn.swee.to/cano/lock/index.shtml
「CGIのファイルロック問題」
http://blog.mikage.to/mika/2005/07/cgi_0916.html
「ファイルロック(排他処理)について」
http://tech.bayashi.net/pdmemo/filelock.html

↑ ↑を読んで出直しなさい、自分。

PHPでのflock()

http://testwiki.仮.jp/index.php?PHP

↑の
「PHP/ファイルロック」シリーズ
が詳しい。
「PHP/ファイルロック/設計」ページに包括的なまとめが載っている。

少し簡単にまとめると

flock($fp, $mode)

$fp : fopen()されたファイルポインタ
$mode : →
LOCK_EX – 排他ロック取得(Write)
LOCK_SH – 共用ロック取得(Read)
LOCK_UN – 取得したロックの解放
LOCK_NB – ロック取得までwaitしない(LOCK_NBは
Windows非対応)

修正すると･･･

```
<?php
// (エラー処理は省略してます)

$fp = fopen($filename, 'a+b'); // 'a' or 'r' で、ファイルを弄らずにオープン
flock($fp, LOCK_EX); // すぐに排他ロック
// ロック「後」に、ファイルポインタの操作とデータのreadをします。
fseek($fp, 0, SEEK_SET);
$data = '';
while (!feof($fp)) {
    $_buf = fread($fp, 8192);
    $data .= $_buf;
}
...(データ処理)...

// データの保存
fseek($fp, 0, SEEK_SET);
ftruncate($fp, 0);
fwrite($fp, $_data, strlen($_data));

// flock($fp, LOCK_UN)を呼ばずに直接fclose()することで、バッファフラッシュ
// とLOCK_UNを暗黙的に行います。
fclose($fp);
```

で、そもそも

flock()とか
「アドバイザリ・ロック」
って何？

flock()

元々はBSDというUnixOSで、「advisory lock」を実装する為に用意された、C言語の関数(システムコール)です。
″man 2 flock″

「アドバイザリ・ロック」？

「adivisory lock」
「推奨ロック」「問い合わせ型ロック(BSDマニュアル)」と訳されている場合も。

つまり、異なるプロセスでも同じロック関数を使うことでロックを実現できる。
逆に言うと、同じ関数を呼ばないとロックできない。
flock()したやつはvimで編集できたりしちゃう。
←→「強制ロック(mandatory lock)」
open(2)のレベルでロックがかかる。

fcntl()というのは？

これもadvisory lock取得の為のシステムコール。

″man 2 fcntl″

「(・ω・)∩
何が何だか分かりません。」

歴史

「先生、すごい・・・
ややこしいです。」

※自分なりに調べてみたんですが、間違っている
点あったら指摘して頂けると助かります。

最初は4.2BSDのflock(2)

最初はflock(2)によるアドバイザリ・ロッで、ファイルの全範囲をロックする機構を提供していた・・・らしい。

全範囲にロックがかかるけど、次のような長所も:
・最後にファイルがcloseされる時にファイルロックが解放される。
・書き込み権限を持っていなくても排他ロック可能

POSIX 1.x で採用されたfcntl()

→ファイルの一部分をロックできる機能があった。

元々System V Release 3 においてfcntl()システムコールで実装されたもの。lockf()はfcntl()のラッパーとして同システムで提供された。

4.4BSDでPOSIX準拠

POSIX : Portable Operating System Interface(Wikipedia読め)

4.4BSDの開発でPOSIX準拠し、ファイルの一部ロックの実装のためfcntl(2)の実装に乗り出したが・・・

POSIXのfcntl(2)、何か
イケテナイお。

「複数のプロセスから参照されているファイル記述子に対して、どこか一つでもcloseシステムコールを呼ぶと全部ロックが解放される。」

「排他ロックを得るためにはファイルを書き込みモードでオープンしなければならない。ファイルに対する書き込み権限を有していないと、排他ロックが得られない。」

4.4BSDが採った二枚舌

「fcntl(2)についてはPOSIX準拠にするお。」
「flock(2)については4.2BSDの遣り方(長所)を残すお。」
更に・・・
「fcntl(2)はプロセスIDでロックを見分けるお。」
「flock(2)はi-node番号でロックを見分けるお。」
(※ファイル記述子ではなくて、i-node番号で見分けてるようです。)

flock(2)はUNIX標準では無いという事実

・POSIXで定義されているのはfcntl(2)の方・・・らしい。

・なので、PHPにはHAVE_FLOCKというコンパイル時のdefine値があり、flock(2)が無いシステムではこれを外してコンパイルすることで、fcntl(2)を使うようになる。

・Perlも、flock(2)が使えない場合は内部でエミュレートしていたりする。

対応状況

SUS(*1) : **fcntl()のみ。**lockf()はオプション。
FreeBSD 5.2.1 : fcntl(), lockf(), flock()全部O.K.
Linux 2.4.22 : 同上
MacOSX 10.3 : 同上
Solaris 9 : 同上
HP-UNIX(バージョン不明) : 同上
(*1 : Sigle Unix Specification : POSIXの後に出てきた共通仕様)
※↑の内強制ロック(MandatoryLock)を提供しているのはLinuxとSolarisのみ。

なーんだ、全部サポートしてんじゃん？

細部が違う。

（例：Linux & BSD とHPの違い）

fork()したときのロックの引き継ぎに違いがある。

Linux, BSDはflock()で取得したロックをfork()したプロセスでも引き継げるが、HP-UNIXの場合は引き継げない。
JMAN、BSDのマニュアル、および以下のURL参照

http://docs.hp.com/ja/B2355-60129/flock.2.html

HP社からの最後の駄目だし

「flock() は、どの UNIX 標準の一部でもありません。
したがって、プラットフォーム間で移植性のあるアプリケーションを開発する場合には、flock() ではなく fcntl() ファイルロックインタフェースを使用してください。」

http://docs.hp.com/ja/B2355-60129/flock.2.html

まあ、PHP使ってる限りは

あんまり細かい差異は意識しなくて良いかも。

そもそもfork()なんてPHPレベルじゃ一般的なWebアプリでは使わないし・・・。

(´ー`)ﾌｩｰ. . .

ふと。

PHPのセッションのデフォルトの保存機構（ファイル保存）も似たような条件。

ということでようやく

2.

「PHPのソースコード、大丈夫お？」

「ソースコード調べるお～！！」

長いので
省略

結論

「PHP4.4.8, PHP5.2.5 とも、ちゃんとflock()使ってるお～！」

「PHPのデフォルトのファイル保存でsession使っている限りは、安心して大丈夫だお～！」

疑ってすみません。

３．注意パターン

DBなどにセッションデータを保存するよう自前で関数を定義し、
session_set_save_handler()
で設定していた場合。

チェックポイント

・ロックは掛けているか？
→DB使用ならトランザクションを使用しているか、分離レベル(Isolation Level)は適切か。

・ignore_user_abort()の使用は適切か？
→DB使用時でignore_user_abort()を呼んでいない時、クライアントが接続断してPHPスクリプトが終了しても、トランザクションは適切にロールバックされているか？

チェックポイント（続き）

・DBを使用している場合

セッション保存用のトランザクションと、ビジネスロジック実行用のトランザクションは分離しているか？
（接続を二つ持つ、トランザクションがネストできるDBMSを使用している、など）

サンプル

「先生、時間が
ありません！！」

それでもサンプル

1．session_file_acid.php

↓

「ちゃんと排他されてるお～～！！」

MyISAMでサンプル

２．session_mysql_myisam.php

↓

「なんかおかしいお～～！！
ロックがかかってないお～～～！！」

トランザクションが使えるInnoDBで。

３．session_mysql_innodb_0.php

↓

「ロックがかかってないお～～！！？？」

原因

１．SELECTがFOR UPDATEになっていないため、UPDATE前の値を他のDB接続から読み取れてしまう。

２．さらに、PHPがセッションデータを復元するときのselectSQLと、セッションデータを保存するときのreplace(updateでも症状同じ)間にsleepを入れてしまっているため、リクエストが早く終わる方が、先にトランザクションをcommitしてしまう。

InnoDBの分離レベル

- ・http://www.hi-ho.ne.jp/~illusia/nif/mysql_4th_stage.htm
- http://dev.mysql.com/doc/refman/5.1/ja/innodb-transaction-isolation.html
- http://dev.mysql.com/doc/refman/5.1/ja/innodb-locking-reads.html
- →デフォルトはREPEATABLE READ
- →一番厳しいのがSERIALIZABLE

SERIALIZABLEにしてみるお！

4. session_mysql_innodb_1.php

↓

「ロックはかかってるっぽいけど・・・
データの読み込みが共有されてるっぽいお～～」

原因

- SERIALIZABLEでは、単純なSELECTステートメントを″ … LOCK IN SHARE MODE″として処理しているため、データ自体は読み取れてしまう。
- →update前のレコードを読み取らせたくない＝select自体でロックしてしまいたい、場合は″SELECT … FOR UPDATE″を使いましょう。

SERIALIZABLE + SELECT FOR UPDATE

5. session_mysql_innodb_2.php

↓

「完璧だお！」

それにしても

ここまで細かくする必要性はあるのか？
そもそもセッションに、衝突が発生したら困るような
データを入れるような処理がある方が問題か
も・・・。

まとめ

「先生、すごい・・・
奥深いです。」

参考資料

・http://www.hi-ho.ne.jp/~illusia/nif/mysql_4th_stage.htm
・http://dev.mysql.com/doc/refman/5.1/ja/innodb-transaction-model.html
・http://dev.mysql.com/doc/refman/5.1/ja/innodb-transaction-isolation.html
・http://dev.mysql.com/doc/refman/5.1/ja/innodb-locking-reads.html
・http://dev.mysql.com/doc/refman/5.0/en/innodb-transaction-model.html
・http://dev.mysql.com/doc/refman/5.0/en/innodb-transaction-isolation.html
・http://dev.mysql.com/doc/refman/5.0/en/innodb-locking-reads.html

・「4.4BSDの設計と実装」
・「Advanced Programming in the Unix Environment」

質疑応答時間の突っ込み

「mkdir型のロックもあるよね。
そっちの方がtestとsetがATOMICにできる。」

「flock()はNFSには使えないから・・・。
= セッションデータの保存先をNFSで共有すると
flock()の意味が無くなるので注意。」