Apache モジュール mod_unique_id

説明:	それぞれのリクエストに対する一意な識別子の入った環境変数を 提供する
ステータス:	Extension
モジュール識別子:	unique_id_module
ソースファイル:	mod_unique_id.c

概要

このモジュールは非常に制限された条件下で、 それぞれのリクエストに「すべて」のリクエストに対して 一意に決まることが保証されている魔法のトークンを提供します。 この一意な識別子は、適切に設定されたクラスタでは複数の マシンの間でさえも一意になります。それぞれのリクエストに対して環境変数 UNIQUE_ID に識別子が設定されます。 一意な識別子が便利な理由はいろいろありますが、 このドキュメントの目的からは外れるため、ここでは説明しません。

トピック

ディレクティブ

このモジュールにディレクティブはありません。

理論

まずはじめに、Apache サーバが Unix マシンでどのように動作をするかを簡単に説明します。 この機能は現時点では Windows NT ではサポートされていません。 Unix マシンでは Apache はいくつかの子プロセスを作成し、 その子プロセスが一つずつリクエストを処理します。それぞれの子プロセスは、 生存期間中に複数のリクエストを扱うことができます。この議論では子プロセス間では一切データを共有しないことにします。 以後、この子プロセスのことを httpd プロセスと呼びます。

あなたのウェブサイトにはあなたが管理するいくつかのマシンがあるとします。 それらをまとめてクラスタと呼ぶことにします。それぞれのマシンは複数の Apache を実行することもできます。 これらすべてをまとめたものが「宇宙」であると考えられます。 いくつかの仮定の下で、クラスタのマシン間がたくさん通信をすることなく、 この宇宙の中でそれぞれのリクエストに一意な識別子を生成できることを示します。

クラスタにあるマシンは以下の要求を見たさなければなりません。(マシンが一つだけだとしても、NTP で時計を合わせる方が良いです。)

- NTP や他のネットワーク上で時間を合わせるプロトコルによって 各マシンの時間の同期が取られていること。
- モジュールがホスト名を引いて違う IP アドレスを受け取ることができるように、 クラスタのそれぞれのマシンのホスト名が違うこと。

オペレーティングシステムにおいては、pid (プロセス ID) が 32 ビットの範囲内であることを仮定します。オペレーティングシステムの pid が 32 ビットを超える場合は、簡単な

修正ではありますが、 コードを変更する必要があります。

これらの仮定が満たされていると、ある時点において、 クラスタ内のどのマシンのどの httpd プロセスでも、一意に同定することができます。これはマシンの IP アドレスと httpd プロセスの pid で十分に行なうことができます。 ですから、リクエストに一意な識別子を生成するためには、 時刻を区別する必要があるだけです。

時刻を区別するために、Unix のタイムスタンプ (UTC の 1970 年 1 月 1 日からの秒数) と、16 ビットのカウンタを使います。 タイムスタンプの粒度は一秒ですので、一秒間の 65536 までの値を表現するためにカウンタを使用します。四つの値 (ip_addr, pid, time_stamp, counter) で各 httpd プロセスで一秒の間に 65536 リクエストを数えあげることができます。 時間が経つと pid が再利用されるという問題がありますが、 この問題を解決するためにカウンタが使用されます。

httpd の子プロセスが作成されると、カウンタは (その時点のマイクロ秒 ÷ 10) modulo 65536 で初期化されます (この式はいくつかのシステムにある、マイクロ秒の タイマの下位ビットが異なるという問題を解決するために選ばれました)。 一意な識別子が生成されたとき、使用されるタイムスタンプは ウェブサーバにリクエストが到着した時刻になります。 カウンタは識別子が生成されるたびに増加します (あふれた場合は 0 に戻ります)。

カーネルはプロセスをフォークすると、それぞれのプロセスのために pid を生成します。pid は繰り返されることが許可されています (pid の値は多くの Unix では 16 ビットですが、新しいシステムでは 32 ビットに拡張されています)。 ですから、ある程度の時間が経過すると同じ pid が再び使用されます。 しかし、一秒内に再使用されなければ、 四つの値の一意性は保たれます。つまり、我々はシステムが一秒間 に 65536 個のプロセスを起動しないと仮定しています (いくつかの Unix では 32768 プロセスですが、それですらほとんどあり得ないでしょう)。

何らかの理由で、同じ時刻が繰り返されたとしましょう。 つまり、システムの時計が狂っていて、もう一度過去の時刻になってしまった (もしくは進みすぎていたときに、 正しい時刻に戻したために再び将来の時刻になってしまった) とします。 この場合、pid とタイムスタンプが再使用されることが簡単に示されます。 カウンタ初期化用の関数は、この問題の回避を手助けしようと選択されています。 本当はカウンタの初期化をするためにランダムな数字を使いたいのですが、 ほとんどのシステムでは簡単に使用できる数は無いことに注意してください (すなわち、rand ()は使えません。rand () には seed を与える必要があり、seed には時刻を使えません。一秒単位では、 その時刻はすでに繰り返されているからです)。 これは、完璧な対策ではありません。

この対策はどのくらい効果があるでしょうか? ここでは、マシン群の中の一つは最大で一秒に 500 リクエストを扱うと仮定します (これを書いている時点では妥当な上限です。 通常システムがすることは静的なファイルを取りだすだけではありませんから)。 それを行なうために、そのマシンは並行して来るクライアントの数に 応じた数の子プロセスを要求します。 しかしながら、悲観的に考えて、一つの子プロセスが一秒に 500 リクエストを扱えるとします。そうすると、(一秒の精度において) 時刻が同じ時を繰り返すと、この子プロセスがカウンタの値を再び使い、 一意性が壊れる可能性が 1.5% あります。 これは非常に悲観的な例で、実世界の値では、ほとんど起こりそうにありません。 それでもこれが起こる可能性のあるようなシステムなら、 (プログラムコードを編集して) カウンタを 32 ビットにするのが良いでしょう。

サマータイムにより時計が「戻される」ことを気にしている人が いるかもしれません。ここで使用される時間は UTC であり、 それは「常に」進むのでここでは問題になりません。

x86 上の Unix はこの条件を満たすために適切な設定が必要かもしれないことに 注意してください。マザーボードの時計は UTC になっていて、 他の時間はそこから適切に補正されることを仮定できるように 設定されなければなりません。そのような場合でさえ、NTP を使っているならばリブート後にすぐ正しい UTC の時間になるでしょう。

UNIQUE_ID 環境変数は 112 ビット (32 ビット IP アドレス、32 ビット pid, 32 ビットタイムスタンプ、16 ビットカウンタの四つの組) をアルファベット [A-Za-z0-9@-] を用いて MIME の base64 符号化と同様の方法により符号化し、19 の文字を生成することにより作成されます。MIME の base64 のアルファベットは実際は [A-Za-z0-9+/] ですが、 + と / とは URL では特別な符号化が必要なので、あまり望ましくありません。 全ての値はネットワークバイトオーダで符号化されますので、 符号は違ったバイトオーダのアーキテクチャ間で比較可能です。 実際の符号化の順番は: タイムスタンプ、IP アドレス、pid, カウンタです。この順には目的がありますが、 アプリケーションは符号を解析するべきではないことを強調しておきます。 アプリケーションは符号化された UNIQUE_ID 全体を透過的なトークンとして扱うべきです。 UNIQUE_ID は他の UNIQUE_ID との等価性を調べるためだけにのみ使用できます。

この順番は将来、既存の UNIQUE_ID のデータベースとの衝突を心配することなく符号を変更することが 可能になるように選択しています。 新しい符号はタイムスタンプを最初の要素として残すのが望ましく、 それ以外は同じアルファベットとビット長を使うことができます。 タイムスタンプは本質的に増加系列ですので、 クラスタの全てのマシンがリクエストとサーバ機能を停止して、 古い符号化方式を使用するのをやめるフラグ秒があれば十分です。 その後は、リクエストを再開し、 新しい符号を発行することができるようになります。

我々はこれが、 この問題に対する比較的移植性の高い解決法だと考えています。 Windows NT のようなマルチスレッドのシステムに拡張することができますし、 将来必要になればさらに増やすこともできます。 ID は必要に応じて長くすることができますので、生成された ID は実質上、無限に有効です。また、クラスタのマシン間の通信も事実上必要なく (NTP による同期のみが必要で、これはオーバヘッドはあまりありません)、httpd プロセス間の通信も必要ありません (通信はカーネルにより割り当てられた pid の値により暗黙の内に行なわています)。 さらに限られた状況下では、ID はさらに短くすることができますが、より多くの情報を仮定する必要がでてきます (例えば、32 ビット IP アドレスはどのサイトにおいても過剰な情報ですが、 それの代わりになる移植性のあるものはありません)。