



C++11:スレッド・ライブラリひとめぐり

C++11:スレッド・ライブラリひとめぐり【補足編:2】

C++

印刷用を表示

ツイート 14 2 33 G+

επιοσημη [著] 2018/02/15 14:00

ダウンロード↓ サンプルファイル (53.3 KB)

マルチスレッド・アプリケーションでは複数のスレッドが互いに干渉することなくフルスピードでぶん回ってくれるのが最もパフォーマンス高くて理想的。ところが実際にはそうも言ってられません、他のスレッドから邪魔されないよう一時的に他スレッドに待ってもらわなければならないことがしばしば起こります。前回に引き続きC++スレッドサポートライブラリによるスレッドの排他のおはなし。

目次

- ・ 前回:『C++11:スレッド・ライブラリひとめぐり【補足編:1】』
- ### mutex:スレッド同士が邪魔しない／させないからくり

とっても簡単な関数: add()を用意しました。

list01

```
void add(long* address, long value) {  
    *address += value;  
}
```

どうということはない、addressが指すlong値にvalueを加えるだけの簡単なお仕事です。4つのスレッドがこの関数呼んで1つのlong値にそれぞれ何度も+1、-1、+2、-2します。

list02

```
#include <iostream>  
#include <thread>  
#include <functional>  
#include <chrono>  
#include <mutex>
```

// data-race が発生するハダカの足し算

メンバーメニュー

オプション



Special Contents PR

コーディングもテストも持ち回りな新プログラミング手法「モブプログラミング」とは何か——本場 Hunter社に学ぶ

厳選!キャリアインタビュー PR

フロントエンドの責任者は沖縄在住? ストレスフリーな環境で東京と全く変わらないチャレンジが可能なユミルリンク

次々と移り変わる技術のトレンドをつかみ、エンジニアとデザイナーの橋渡し役を務める——フロントエンドエンジニアの面白さとやりがいとは?

人気ランキング

- 今日 月間
- 1 [情シスの仕事こそ、クリエイティブでおもしろい! 12000人以上が利用するヤフーの社内システムづくり【デブサミ2018】](#)
 - 2 [大規模解析サービスを支える監視サービスと監視構成のポイント](#)
 - 3 [業務システムでも最新技術を使いたい! ITILの壁を乗り越えるため、機能の一部をマイクロサービス化してクラウドに移行【デブサミ2018】](#)

```

void add(long* address, long value) {
    *address += value;
}

void run(std::function<void(long*, long)> fun, long* address) {
    using namespace std;
    using namespace std::chrono;

    auto task = [=](int n, long v) { while ( n-- ) fun(address, v); };
    const int N = 100000;
    thread threads[4];
    auto start = high_resolution_clock::now();

    // スレッドを4本起こして
    threads[0] = thread(task, N, 1L);
    threads[1] = thread(task, N, -1L);
    threads[2] = thread(task, N, 2L);
    threads[3] = thread(task, N, -2L);

    // 全スレッド終了を待つ
    for (thread& thr : threads) thr.join();
    auto stop = high_resolution_clock::now();

    cout << duration_cast<milliseconds>(stop - start).count() << "[ms]" << flush;

}

int main() {
    using namespace std;

    long count = 0L;

    cout << "data-race:          ";
    count = 0;
    run(add, &count);
    cout << ", count = " << count << endl;
}

```

4つのスレッドが何度も行う加算と減算は相殺されて最終的には初期値である0に戻るかというところにあらず、

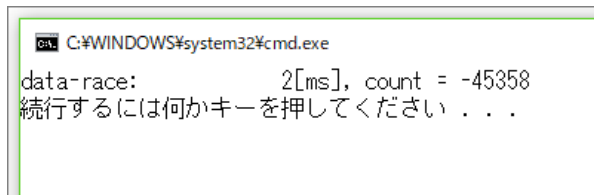


fig01

add()内の処理: *address += valueは、「*addressを読み／valueを加えて／*addressに書く」の3ステップを行います。この3ステップ中に他のスレッドが割り込むことで辻褄が合わなくなるんです。例えば*addressが100のとき、+1するスレッドAと-1するスレッドBがほとんど同時にadd()したとしましょう。両者は*addressを読み、Aは100+1→101、Bは100-1→99を計算します。続いて双方が*addressに書き込むと、どちらが後になるかによって*addressは101か99のいずれかとなり、いずれにせよ期待する100にはなりません。

- 4 [デプロイ自動化をマルチクラウドで! CDツール「Spinnaker」をAWS上で検証してみた【デブサミ2018】](#)
- 5 [超簡単! WPFなどの.NETのアプリからUWPのAPIを使う ～日本語の読み仮名を取得するAPIを題材に](#)
- 6 [デブサミ2018、講演関連資料まとめ](#)
- 7 [オブジェクト指向がわからない! そんなあなたの脳味噌をオブジェクト脳にする準備体操](#)
- 8 [サーバーレスアーキテクチャとは何か?～AWS LambdaとAPI Gatewayによる簡単なAPIの実装を試す](#)
- 9 [Pythonは今のうちに習得すべき?『スラスラわかるPython』著者・岩崎さんとPyCon JP理事の寺田さんが語る](#)
- 10 [650円で買えるマイコンボード「Raspberry Pi Zero」でIoTをはじめよう! ～環境構築とLチカのレシピ](#)

新着

[記事](#)
[ニュース](#)

業務システムでも最新技術を使いたい! ITILの壁を乗り越えるため、機能の一部をマイクロサービス化してクラウドに移行【デブサミ2018】

大規模解析サービスを支える監視サービスと監視構成のポイント

デプロイ自動化をマルチクラウドで! CDツール「Spinnaker」をAWS上で検証してみた【デブサミ2018】

超簡単! WPFなどの.NETのアプリからUWPのAPIを使う～日本語の読み仮名を取得するAPIを題材に

情シスの仕事こそ、クリエイティブでおもしろい! 12000人以上が利用するヤフーの社内システムづくり【デブサミ2018】

[新着記事一覧を見る](#)

Pick Up Links

- Android/iOS固有の機能を使ったアプリもFlash Builderで!
- アプリケーション開発の生産性と品質の向上を支援する
- エンジニア必見! 翔泳社デジタルファースト新刊のお知らせ
- 価値を生む開発に集中しつづける現場インタビュー〜クラウド最前線
- 広告出稿、イベント出展についてのお問い合わせはこちら
- 最新テクノロジーが導くデジタルネイティブなアプリ開発

add()内で行われる「読んで／計算して／書く」一連の処理はその途中で割り込まれてはならないatomic(不可分)な処理なのです。atomic性が保証されないためにデータの辻褄が合わなくなる現象はdata race(データ競合)と呼ばれています。

atomicでなくてはならない一連の処理の実行を複数のスレッドに行わせないカラクリがmutex。

ヘッダ<mutex>に定義されたstd::mutexの主要メンバ関数は、

- lock():ロックを取得する
- try_lock():ロックの取得を試みる
- unlock():ロックを解放する(手放す)

の3つ。ロックは実行権／使用権を手に入れる鍵であり、この鍵を取得できるスレッドはただ1つです。lock()によって使用中となったmutexを他のスレッドがlock()すると、そのスレッドはunlock()によって解放されるまで待ち状態となりlock()から戻ってきません。複数のスレッドがロックを待っている状態でロックが解放されると、いずれか1つのスレッドのみがロックを取得しlock()から抜けてきます。なので、

list03

```
std::mutex mtx;

void mutex_add(long* address, long value) {
    mtx.lock();
    *address += value;
    mtx.unlock();
}
```

としておけば複数のスレッドがmutex_add()してもmtx.lock()とmtx.unlock()に挟まれた処理を行えるスレッドは1つだけ(ほかのスレッドは待たされる)となりatomic性が保証されるってスポンサーです。

mutexでガードしていないハダカのadd()と比べてみました。

list04

```
// std::mutexでガードした足し算
std::mutex mtx;

void mutex_add(long* address, long value) {
    mtx.lock();
    *address += value;
    mtx.unlock();
}
```

.....

```
int main() {
    using namespace std;

    long count = 0L;

    cout << "data-race:          ";
    count = 0;
    run(add, &count);
    cout << ", count = " << count << endl;

    cout << "std::mutex:          ";
    count = 0;
```

**Yahoo! JAPANの
技術リード
エキスパート対談**

国内最大級メディアを支える
技術リードが探る、開発現場
の課題解決[PR]

UNIXコマンド辞典ショートカット

[ファイル操作](#)

[システム管理](#)

[ネットワーク管理](#)

[印刷処理](#)

[インストール](#)

[テキスト処理](#)

[ジョブ管理](#)

[デバイス処理](#)

[圧縮・解凍](#)

[UNIX基本講座](#)



@CodeZine

『CodeZine(コードジン)』の公式アカウントです。

@codezineさんをフォロー

```
run(mutex_add, &count);
cout << ", count = " << count << endl;
}
```

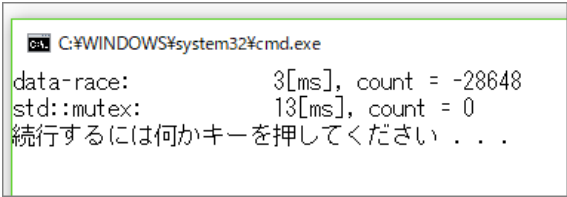


fig02

countはめでたく初期値0に戻っていますが、そのかわりちょっと遅くなっています。ロックの取得／解放には、そこそこの時間がかかるんですよ。

残るメンバ関数:try_lock()は、ロックが取得できるなら取得してtrue、取得できないならあきらめてfalseを返します。スレッドが待ち状態にはなりません。

次ページ》 スレッドの3状態

あなたにオススメ

2017/05/16

Visual C++ 2017で一足お先にfilesystem

2016/04/20

50年前に作られたメモリ管理アルゴリズム「Buddy memory allocation」

2015/07/09

数値演算アルゴリズムひとめぐり

2010/01/26

C++/CLI: とある文字列の相互変換 (コンバージョン)

¥4,298

コンテナ・ベース・オーケストレーション

Docker/Kubernetesで作るクラウド時代のシステム基盤 | 翔泳社の通販

¥3,240

Webサイトパフォーマンス実践入門 高速なWebページを作りたいあなたに | 翔泳社の通販

¥3,888

AWSによるサーバーレスアーキテクチャ | 翔泳社の通販

¥2,570

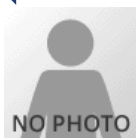
PHPしっかり入門教室 使える力が身につく、仕組みからわかる。 | 翔泳社の通販

PR [『マンガで分かるプログラミング用語辞典』プログラミング入門書の副教材としてぜひ](#)

PR [『C#で始めるテスト駆動開発入門』C#でのTDD実践方法をステップバイステップで紹介](#)



著者プロフィール



επιστημη (エピステマー)

C++に首まで浸かったプログラマー。Microsoft MVP, Visual C++ (2004.01~) だったり わんくま同盟でたまにセッションスピーカやったり 中国茶淹れてにわか茶人を気取ったり、あと Facebook とか。著書: - STL標準講座 (監修) -...

※プロフィールは、執筆時点、または直近の記事の寄稿時点での内容です
Article copyright © 2018 episteme, Shoeisha Co., Ltd.

バックナンバー

連載:C++11:スレッド・ライブラリひとめぐり

[C++11:スレッド・ライブラリひとめぐり【補足編:3】](#)

[C++11:スレッド・ライブラリひとめぐり【補足編:2】](#)

[C++11:スレッド・ライブラリひとめぐり【補足編:1】](#)

[ページトップへ](#)

CodeZineについて

[各種RSSを配信中](#)

プログラミングに役立つソースコードと解説記事が満載な開発者のための実装系Webマガジンです。
掲載記事、写真、イラストの無断転載を禁じます。
記載されているロゴ、システム名、製品名は各社及び商標権者の登録商標あるいは商標です。

SE
SHOEISHA



[ヘルプ](#)

[広告掲載のご案内](#)

[著作権・リンク](#)

[免責事項](#)

[会社概要](#)

[スタッフ募集!](#)

[メンバー情報管理](#)

[メールバックナンバー](#)

[マーケティング](#)

[エンタープライズ](#)

[IT人材](#)

[教育ICT](#)

[マネー・投資](#)

[ネット通販](#)

[イノベーション](#)

[ホワイトペーパー](#)

[プロジェクトマネジメント](#)

[書籍・ソフトを買う](#)

[電験3種対策講座](#)

[電験3種ネット](#)

[第二種電気工事士](#)

[メンバーメニュー](#) | [ログアウト](#)