C++ Transactional Memory
言語拡張の紹介

2012/7/28

Boost.勉強会 #10 東京

はじめに

誰？

twitter @yohhoy / hatena id:yohhoy

何を？

C++11の”次”規格へ提案されている新機能の紹介

“ちょっと(?)未来”のお話です

どうして？

Google先生に聞いても情報がほとんど無い...

→ 勉強会 駆動 教えて君

情報源

”**Draft Specification of Transactional Language Constructs for C++**”, Version 1.1, Feb. 3, 2012

https://sites.google.com/site/tmforcplusplus/

一部翻訳→ http://d.hatena.ne.jp/yohhoy/20120413/

N3341 **Transactional Language Constructs for C++**

http://www.open-std.org/jtc1/sc22/WG21/docs/papers/2012/

→ いきなり次世代標準規格C++1y入りは目指さず、
まずはTR(Technical Report)として提案される

もくじ

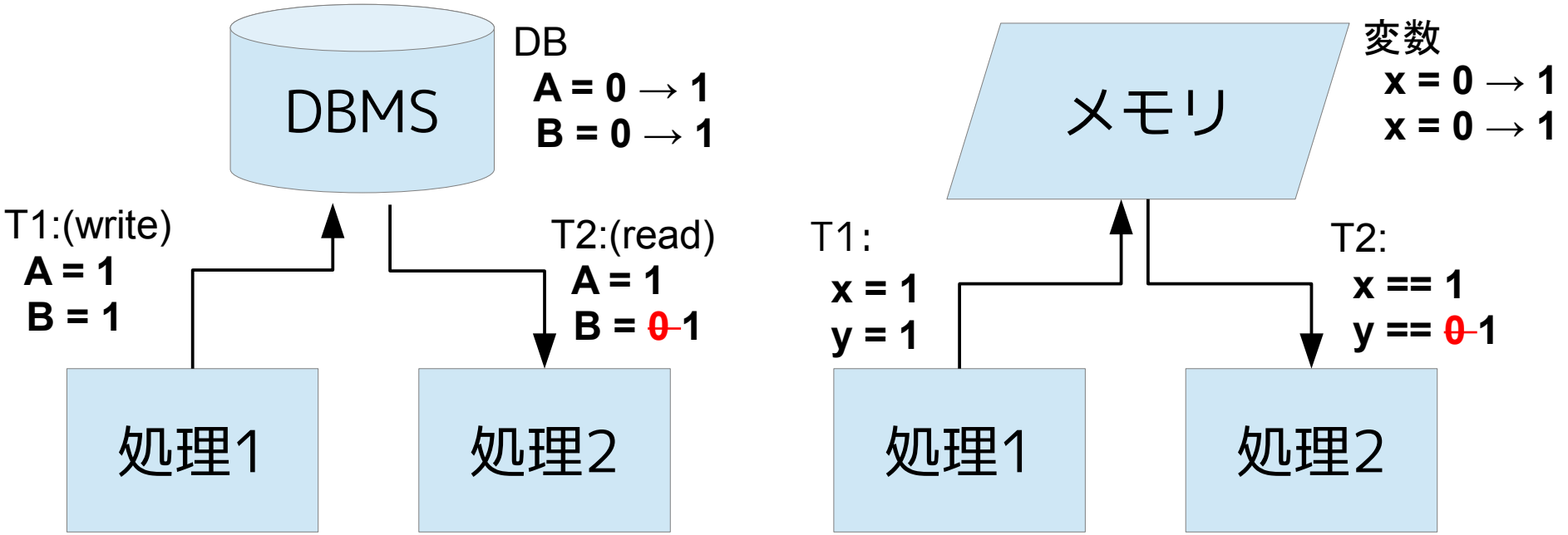
- Transactional Memoryの概略
- データベースシステムとの対比, 導入の目的
- ACID特性, Composability

- C++TM言語拡張は何を提供する／しないのか？
- 既存のロック, atomic変数との関係

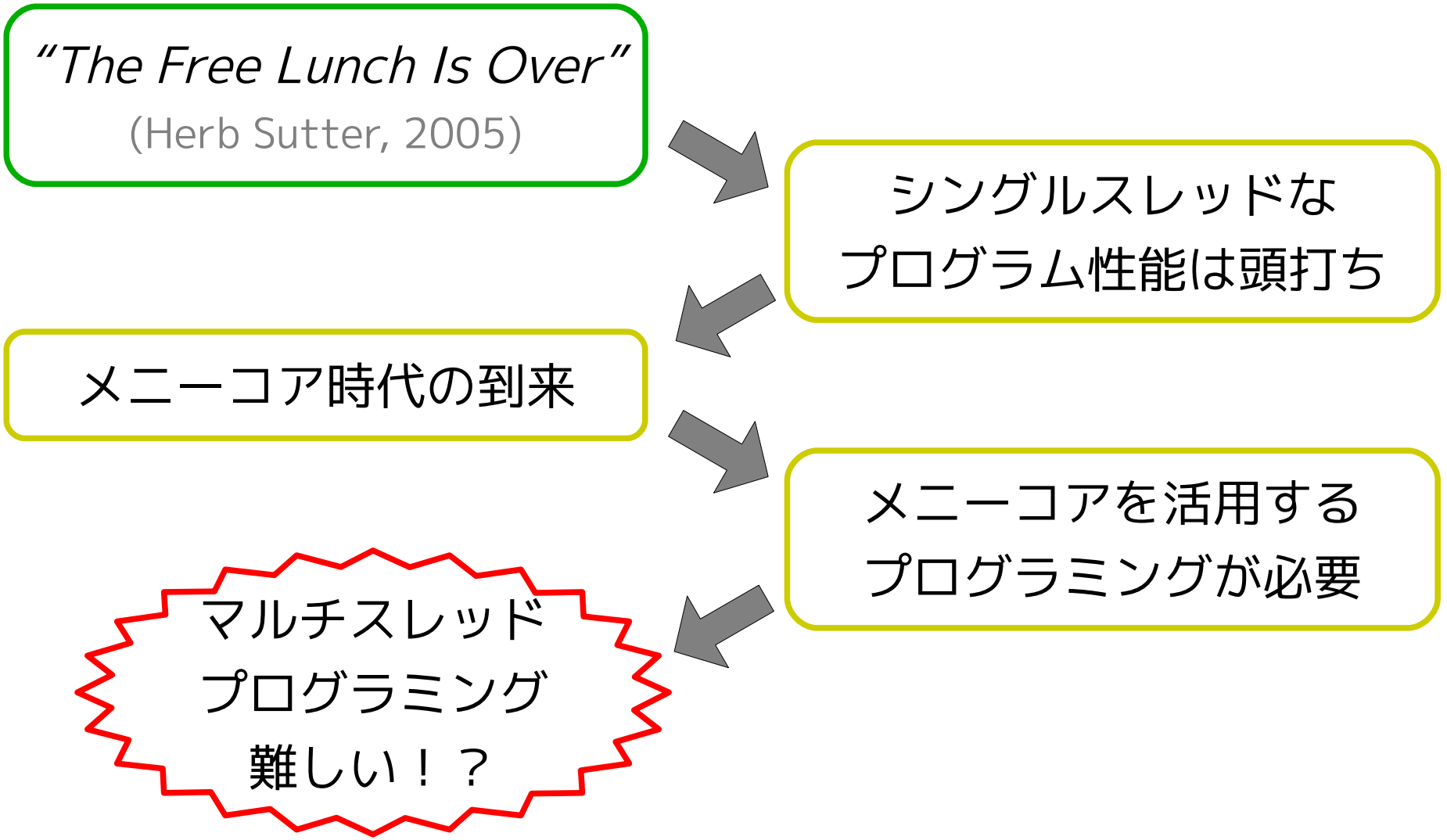
- C++TM言語拡張の新キーワード
- __transaction_relaxed, __transaction_atomic
- __transaction_cancel, C++例外との関係

Transactional Memory(TM)？

発祥はデータベースな世界の”**トランザクション**”から。
同時操作から“**メモリ**”というリソースの一貫性を保護。



TM導入の目的(1)


“The Free Lunch Is Over”
(Herb Sutter, 2005)
シングルスレッドな
プログラム性能は頭打ち
メニーコア時代の到来
メニーコアを活用する
プログラミングが必要
マルチスレッド
プログラミング
難しい！？

TM導入の目的(2)

Transactional Memoryとは

Programming Abstraction

- **並行処理**の記述を**容易**にする**プログラミング抽象**
- プログラマは一連の処理に対し”トランザクション”と宣言する

TMは特段新しいものではなく、既にいくつか実装系が存在

- H/W: IBM BlueGene/Q, (Intel Haswell TSX), (AMD ASF ?)
- S/W: Haskell, Clojure, 各ベンダC++STM, TBoost.STM

C++TM言語拡張の仕様へフィードバック

Intel's TSX = Transactional Synchronization Extensions

AMD's ASF = Advanced Synchronization Facility

TMとACID特性

“トランザクション”システムを名乗るからには...

Atomicity（原子性）

　トランザクション内の処理は**「全て処理される」か「全く処理されない」のいずれか。**（処理途中のキャンセルも含む）

Consistency（一貫性）

　トランザクション処理前後で**メモリ内容に一貫性**がある。
（例：リンクリストのノードがポインタで正しくつながる）

Isolation（分離性）

　トランザクション処理中の**中間状態は他コードから見えない。**

Durability（永続性）

　→ “メモリ”なので電源きったらパア

Composability

Transactional Memoryがもたらす大きなメリット

　トランザクションは安全に”**入れ子**”にできます。

　幾つかの小さなトランザクションを束ねて、1つの大きなトランザクションを構成することが可能です。

デッドロックの恐怖から解放！

C++テンプレート駆使したGenericコードとも相性良い

Photo: espressoed/flickr

もくじ

- Transactional Memoryの概略
- データベースシステムとの対比, 導入の目的
- ACID特性, Composability
- C++TM言語拡張は何を提供する／しないのか？
- 既存のロック, atomic変数との関係
- C++TM言語拡張の新キーワード
- __transaction_relaxed, __transaction_atomic
- __transaction_cancel, C++例外との関係

Transactional Language Constructs for C++ が提供するもの(1)

C++TM言語拡張の仕様は...

トランザクションを**表現**する構文と、トランザクションがどのように**振舞う**かのセマンティクスを定義する。

たったの35page！(1310page@C++11)

- 新しいキーワードを**3つ**追加
- 新しい属性を**5つ**追加
  - C++11で追加された”Generaized Attibuttes”構文を使います。
- C++メモリモデルを**拡張**

g++もClangも未実装

Transactional Language Constructs for C++ が提供するもの(2)

コードの雰囲気

キーワード

- __transaction_relaxed
- **__transaction_atomic**
- **__transaction_cancel**

属性

- outer
- transaction_callable
- **transaction_safe**
- transaction_unsafe
- transaction_may_cancel_outer

```
[[transaction_safe]]
bool is_valid(Node *);

__transaction_atomic {
  node->next = curr_node;
  curr_node->prev = node;

  if ( !is_valid(node) )
    __transaction_cancel;
}
```

Transactional Language Constructs for C++ が提供しないもの

C++TM言語拡張の仕様は...

TMをどのように実現するか(実装方法)を**定義しない。**

- Software TM／Hardware TM, Hybrid TM
- ワード単位／オブジェクト単位, 単一グローバルロック
- 楽観的／悲観的な同期戦略

etc.

並行性無視でTMを実現

プログラマは「**何を保護するか(what)**」を記述し、
「どのように保護するか(how)」は処理系任せ。

Lock, atomic, TM

ロック（ミューテックス）

古典。構造的でないデータやGenericなコードでは使いにくい。複数ロックで容易にデッドロック。

atomic変数

Lock-Free実装などで利用。Primitiveなため軽量かつ高速。メモリモデルの理解なしに迂闊に手を出すと...死。

Transactional Memory

原理的にデッドロックしない。抽象度の高い記述ができ開発効率向上*。実行時動作も十分に高速...？

(*) http://justingottschlich.com/wp-content/uploads/2012/05/2012.SG1_.tm_final.pdf

Lock, atomic, TM

ロック（ミューテックス）

古典。構造的でないデータやGenericなコードは使いにくい。複数ロックで容易にデッドロック。

atomic変数

Lock-Free実装などで利用。Primitiveな操作は軽量かつ高速。メモリモデルの理解なしに迂闊に手を出すと…死。

Transactional Memory

原理的にデッドロックしない。抽象度の高い記述ができ開発効率向上*。十分に高速…？

相補的に共存可能

魔法の言葉
「処理系依存」

動作速度はコンパイラやライブラリ実装による

もくじ

- Transactional Memoryの概略
- データベースシステムとの対比, 導入の目的
- ACID特性, Composability

- C++TM言語拡張は何を提供する／しないのか？
- 既存のロック, atomic変数との関係

- C++TM言語拡張の新キーワード
- __transaction_relaxed, __transaction_atomic
- __transaction_cancel, C++例外との関係

relaxed vs. atomic

2種類のトランザクションを提供

__transaction_relaxed

弱い分離性(Weak Isolation)。**他のトランザクション処理と分離**して実行される。一方非トランザクション処理とは干渉。

__transaction_atomic

強い分離性(Strong Isolation)。**他の全てのメモリアクセス操作と分離**して実行される。

Photo: timsackton/flickr

relaxed, atomicとcancel

relaxed／atomicトランザクション内でできること

__transaction_relaxed

　トランザクション内で**任意の処理を実行可能**。ファイルI/Oといった非可逆な操作もOK。キャンセル不可。

__transaction_atomic

　トランザクション内の処理は**”安全”な操作に限定される**。**__transaction_cancel**により処理途中で**キャンセル可能**（途中までの変更をロールバック）。

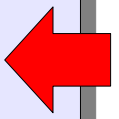
__transaction_atomicとC++例外

トランザクション内からの例外送出はコミット操作になる。

キャンセル例外送出の新構文。
ただし例外は整数型に限定。
（std::exception等はNG）

```
int x = 0, y = 0;
__transaction_atomic {
  x = 42;
  throw -1;
  y = 100;
}
// x = 42 && y = 0
```

```
__transaction_atomic {
  x = 42;
  __transaction_cancel throw -1;
}
// x = 0
```



ほか

説明していないもの

- 式／関数に対するトランザクション指定
- outer属性による最外(outermost)トランザクション指定
- transaction_may_cancel_outer属性, cancel文のレキシカルスコープ制約
- transaction_unsafe/safe属性の明示と推論
- トランザクションへのnoexcept指定
- メモリモデルの拡張（happens-beforeがどーたら）

etc.

概略まとめ→ http://d.hatena.ne.jp/yohhoy/20120414/

まとめ

C++ Transactional Memory言語拡張

- C++11の次のTR／標準規格を目指した提案
- 並行処理の記述を容易にするプログラミング抽象
- トランザクションを表現する新しい構文を追加
- トランザクションの動作を定義（実装方法はスコープ外）
- 既存のロックやatomic変数と併用可能

で、いつから使えんの？

今日から。

GCC 4.7から実験的サポートが始まっています。

http://gcc.gnu.org/wiki/TransactionalMemory

対応状況調査→ http://d.hatena.ne.jp/yohhoy/20120603/

Try **“g++ -std=c++11 -fgnu-tm”**, and enjoy it.

→ そして~~教えて~~情報共有ください（当初目的）

Photo: gwire/flickr