

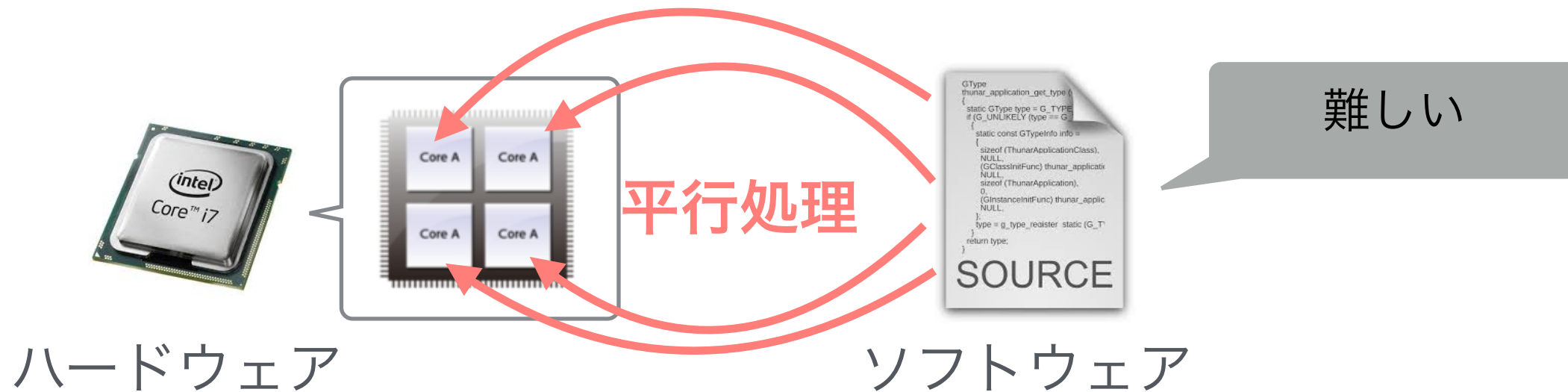
Learning from Mistakes — A Comprehensive Study on Real World Concurrency Bug Characteristics

Shan Lu Soyeon Park Eunsoo Seo Yuanyuan Zhou

Department of Computer Science,
University of Illinois at Urbana Champaign, Urbana, IL 61801
{shanlu,soyeon,eseo2,yyzhou}@uiuc.edu

背景

- マルチコア普及と共に平行処理プログラミングも普及



- 先行研究
 - バグ検出
 - ▶ データ競合バグ, **デッドロック**がメイン
 - テスト
 - ▶ スレッドごとの処理を時間差でテスト
 - ▶ 同時に実行できるかはわからない
- 平行処理で起きたバグそのものの特徴を分析した研究はない

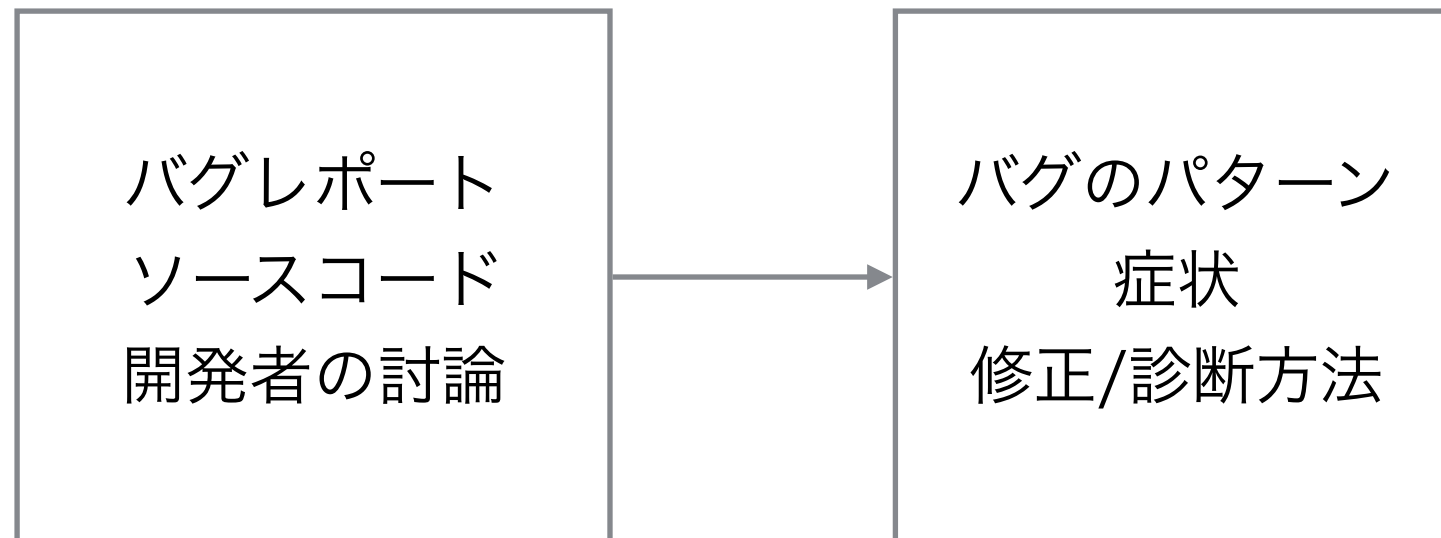
目的とアプローチ

- ・ 目的

- 平行処理プログラミングで発生するバグの特徴を理解する
 - ▶ バグのパターン, 症状, 修正方法

- ・ アプローチ

- 目視で検証



考えられる貢献

バグ検出

開発者が修正するために必要な情報
の調査

テスト

代表的なバグを選択的にテスト

プログラミング言語設計

有益なガイドラインを提供

データセット

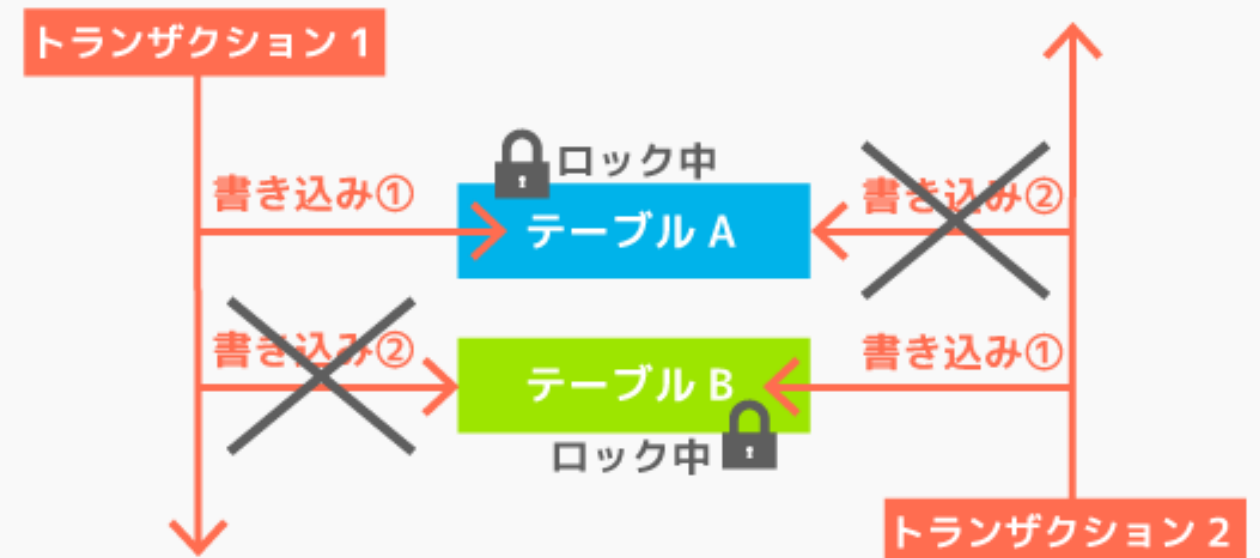
- 平行処理に関する105個のバグを取得

Application	Description	# of Bug Samples	
		Non-Deadlock	Deadlock
MySQL	Database Server	14	9
Apache	Web Server	13	4
Mozilla	Browser Suite	41	16
OpenOffice	Office Suite	6	2
Total		74	31

Table 3. Our application set and bug set

- デッドロックかどうかで分類
 - 修正方法が変わってくる
 - 先行研究はデッドロック検出がメイン
 - ↑ 処理の順番に焦点をあてたいから？

デッドロック



それぞれがロック中のテーブルにアクセスしようとして両すくみの状態となってしまう、処理が停止する

分類方法

- ・ 3つの側面に注目
 - バグのパターン（特徴）
 - Manifest
 - バグの修正方法

Definitions Related to Bug Pattern Study			
Dimension	Category	Description	Abbr.
Bug Pattern*	Atomicity Violation	The desired serializability among multiple memory accesses is violated. (i.e. a code region is intended to be atomic, but the atomicity is not enforced during execution.)	Atomicity
	Order Violation	The desired order between two (groups of) memory accesses is flipped. (i.e. A should always be executed before B , but the order is not enforced during execution.)	Order
	Other	Concurrency bugs other than the atomicity violation and order violation.	Other
Definitions Related to Bug Manifestation Study			
Dimension	Term	Definition	
Bug Manifestation	Manifestation Condition	A specific execution order among a smallest set (S) of memory accesses. As long as that order is enforced, no matter how, the bug is guaranteed to manifest.	
	# of threads involved	The number of distinct threads that are included in S .	
	# of variable involved	The number of distinct variables that are included in S .	
	# of accesses involved	The number of accesses that are included in S .	
Definitions Related to Bug Fix Study			
Dimension	Category	Description	Abbr.
Non-deadlock Fix Strategy	Condition Check	(1) While-flag; or (2) optimistic concurrency with consistency check.	COND
	Code Switch	Switch the order of certain statements in the source code.	Switch
	Design Change	Change the design of data structures or algorithms.	Design
	Lock Strategy	(1) Add/change locks; or (2) adjust the region of critical sections.	Lock
	Other	Strategies other than the above ones.	Other
Deadlock Fix Strategy	Give up resource	Not acquiring a resource (lock, etc.) for certain code region.	GiveUp
	Split Resource	Split a big resource to smaller pieces to avoid competition.	Split
	Change acquisition order	Switch the acquisition order among several resources.	AcqOrder
	Other	Strategies other than the above ones.	Other
Concerns in Transactional Memory	Very long code	A code region is too long to be put into a transaction.	Long
	Rollback Problem	Some I/O and system calls are hard to roll back.	Rollback
	Code Nature	Source code with certain design is hard to turn to transaction.	Nature

Table 2. Our characteristic categories and definitions. (*: The bug pattern category is determined by the root cause of a concurrency bug, i.e. what type of programmers' synchronization intention is violated, *regardless of possible bug fix strategies.*)

分類方法：バグのパターン（特徴）

Definitions Related to Bug Pattern Study			
Dimension	Category	Description	Abbr.
Bug Pattern*	Atomicity Violation	The desired serializability among multiple memory accesses is violated. (i.e. a code region is intended to be atomic, but the atomicity is not enforced during execution.)	Atomicity
	Order Violation	The desired order between two (groups of) memory accesses is flipped. (i.e. <i>A</i> should always be executed before <i>B</i> , but the order is not enforced during execution.)	Order
	Other	Concurrency bugs other than the atomicity violation and order violation.	Other

・ 3つ

- Atomicity

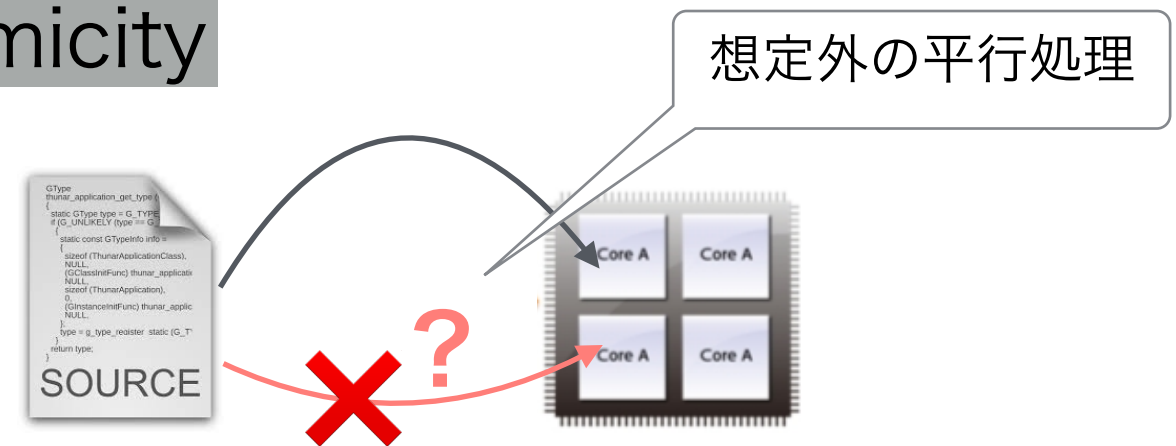
- 単独処理のつもりで書いたけど、平行処理している

- Order：処理の順番

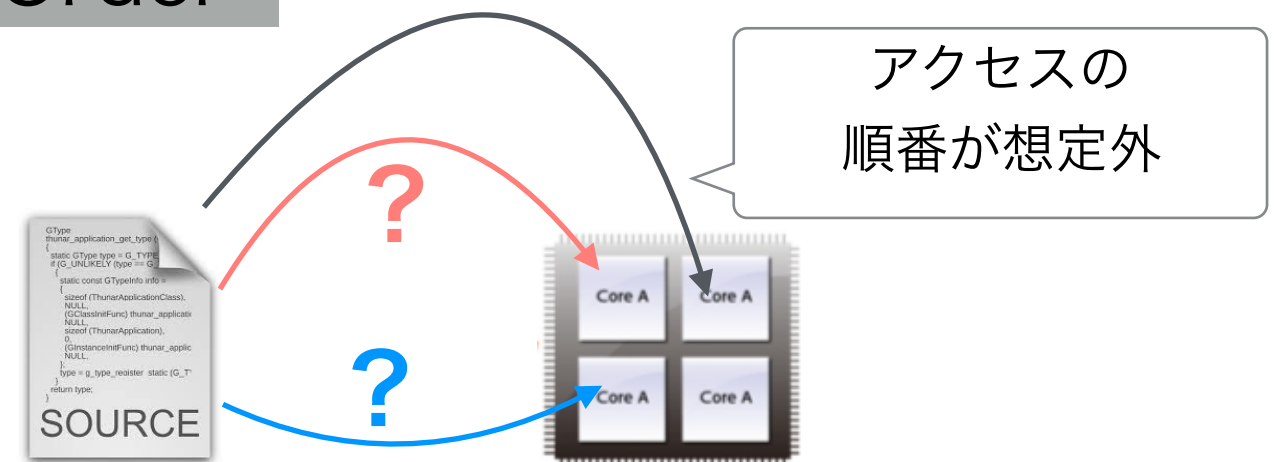
- 処理の順番が間違っている

- その他

Atomicity



Order



分類方法：Manifest

Definitions Related to Bug Manifestation Study		
Dimension	Term	Definition
Bug Mani- festation	Manifestation Condition	A specific execution order among a smallest set (S) of memory accesses. As long as that order is enforced, no matter how, the bug is guaranteed to manifest.
	# of threads involved	The number of distinct threads that are included in S .
	# of variable involved	The number of distinct variables that are included in S .
	# of accesses involved	The number of accesses that are included in S .

- ・ 以下の条件と平行処理に関するバグとの関係
 - スレッド数
 - 変数の数
 - アクセス数

結果：バグのパターン

※前提条件（74件）は、非デッドロックバグ

Application	Total	Atomicity	Order	Other
MySQL	14	12	1	1
Apache	13	7	6	0
Mozilla	41	29	15	0
OpenOffice	6	3	2	1
Overall	74	51	24	2

合計72

知見：ほとんど（72/74件）がAtomicityとOrderに分類できる

示唆：まず、これらのパターンに焦点を当てる

知見：現状のツールで検出できないOrderバグ（24/74件）

示唆：新しい検出技術が必要（現在はデッドロックのバグ検出がメイン）

結果：バグ例, Atomicity

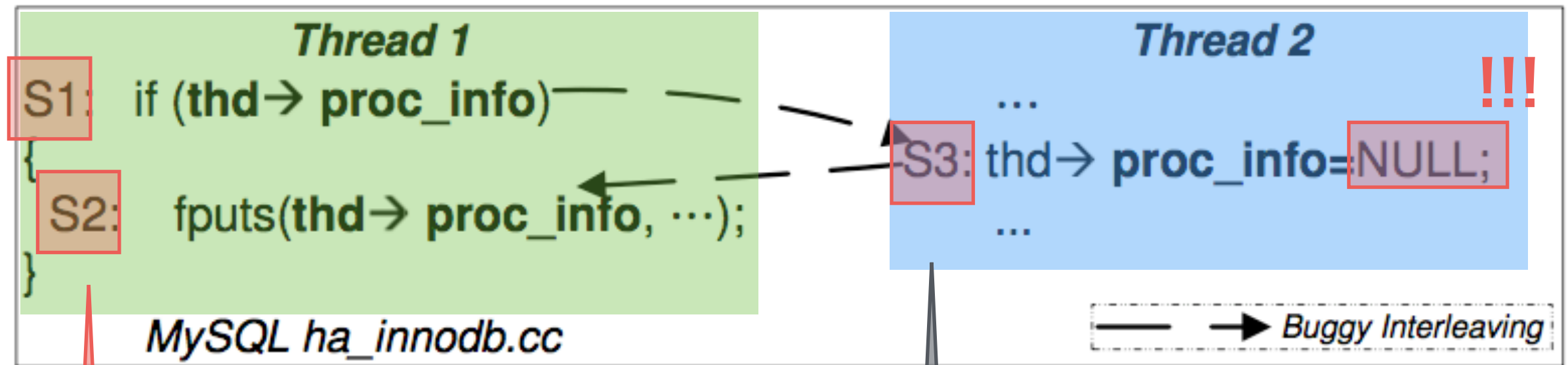


Figure 1. An atomicity violation bug from MySQL.

Thread 1では、
S1が値（非NULL）を読み込むと
S2も同じ値を読みこむと思う

S2に値が読みこむ前に
S3が値を変えてしまう

開発者が予期せぬ処理が入ってた
(ある意味, 順序の問題 (Order) ともいえる)

結果：Manifest, スレッド数とバグの関係

Definitions Related to Bug Manifestation Study		
Dimension	Term	Definition
Bug Mani- festation	Manifestation Condition	A specific execution order among a smallest set (S) of memory accesses. As long as that order is enforced, no matter how, the bug is guaranteed to manifest.
	# of threads involved	The number of distinct threads that are included in S .
	# of variable involved	The number of distinct variables that are included in S .
	# of accesses involved	The number of accesses that are included in S .

スレッド数とバグの関係

Non-deadlock concurrency bugs					
Application	Total	Env.	>2 threads	2 threads	1 thread
MySQL	14	1	1	12	0
Apache	13	0	0	13	0
Mozilla	41	1	0	40	0
OpenOffice	6	0	0	6	0
Overall	74	2	1	71	0

Deadlock concurrency bugs					
Application	Total	Env.	>2 threads	2 threads	1 thread
MySQL	9	0	0	5	4
Apache	4	0	0	4	0
Mozilla	16	0	1	14	1
OpenOffice	2	0	0	0	2
Overall	31	0	1	23	7

Table 5. The number of threads (or environments) involved in concurrency bugs.

知見：ほとんど（101/105件）がスレッド数2以上。さらに、**スレッド数2が多い**
 示唆：ペアワイズテスト（ペアのスレッド毎）でテストすればいい

2スレッドが多い
 スレッドが大きいほどバグがしやすいわけではないというわけではない
関与しているのが2スレッドの場合が多い

知見：デッドロックバグのうちのいくつか（7/31件）は1スレッドだけ
 示唆：検出・回避が簡単

結果：Manifest, 変数の数とバグの関係

Definitions Related to Bug Manifestation Study		
Dimension	Term	Definition
Bug Mani- festation	Manifestation	A specific execution order among a smallest set (S) of memory accesses.
	Condition	As long as that order is enforced, no matter how, the bug is guaranteed to manifest.
	# of threads involved	The number of distinct threads that are included in S .
	# of variable involved	The number of distinct variables that are included in S .
	# of accesses involved	The number of accesses that are included in S .

変数の数とバグの関係

Non-deadlock concurrency bugs			
Application	Total	>1 variables	1 variable
MySQL	14	6	8
Apache	13	4	9
Mozilla	41	15	26
OpenOffice	6	0	6
Overall	74	25	49

Deadlock concurrency bugs				
Application	Total	>2 resources	2 resources	1 resource
MySQL	9	0	5	4
Apache	4	0	4	0
Mozilla	16	1	14	1
OpenOffice	2	0	0	2
Overall	31	1	23	7

Table 6. The number of variables (resources) involved in concurrency bugs.

知見：非デッドロックのバグの66%（49/74件）は1つの変数を含む
示唆：1つの変数への同時アクセスに着目

知見：非デッドロックのバグの34%（無視できない数）は複数の変数を含む
示唆：新しいバグ検出ツールが必要

既存のツールは単一変数同時実行バグのみに焦点を当てている

知見：97%（30/31件）は多くとも2つのリソースを含む
示唆：2つのリソースの取得と解放の順序をペア毎にテストすればいい

結果：平行プログラミングのバグを修正するには？

- ・ TM（トランザクションメモリ）が有効か調べた
 - 有効だと言われている
- ・ 今回取得したバグでは20件だけ有効だった

Application	Total	Can Help	TM might help(concerns:)			Little Help
			Long	Rollback	Nature	
MySQL	23	7	0	14	0	2
Apache	17	7	0	3	1	6
Mozilla	57	25	8	9	5	10
OpenOffice	8	2	0	4	0	2
Overall	105	41	8	30	6	20

Table 10. Can TM help avoid concurrency bugs?

- ・ もっといいTM作るには、今回の調査の知見使えると思う
 - 非デッドロックの順番を考慮する

まとめ

- ・ 平行処理プログラミングが普及している
- ・ 非デッドロックバグのうち、処理順序を考慮してバグの特徴を調査した
- ・ 非デッドロックバグ案外多い（74/105件），ちゃんと分析したら焦点を当てるべき対象，対処法わかりそう
- ・ 現存するツールでは物足りない，新しいツールを作る必要性

所感

- ・ やりたいことが明瞭でわかりやすい
- ・ 読みやすい
 - あんまりみたことのない構成（分野の問題？）
 - ▶ 1.1 「Motivation」, 1.2 「Contributions」
- ・ 105件でいいんや
- ・ ものすごく参考になった
 - ・ 構成, 研究の仕方, 参考文献の多さ. . .