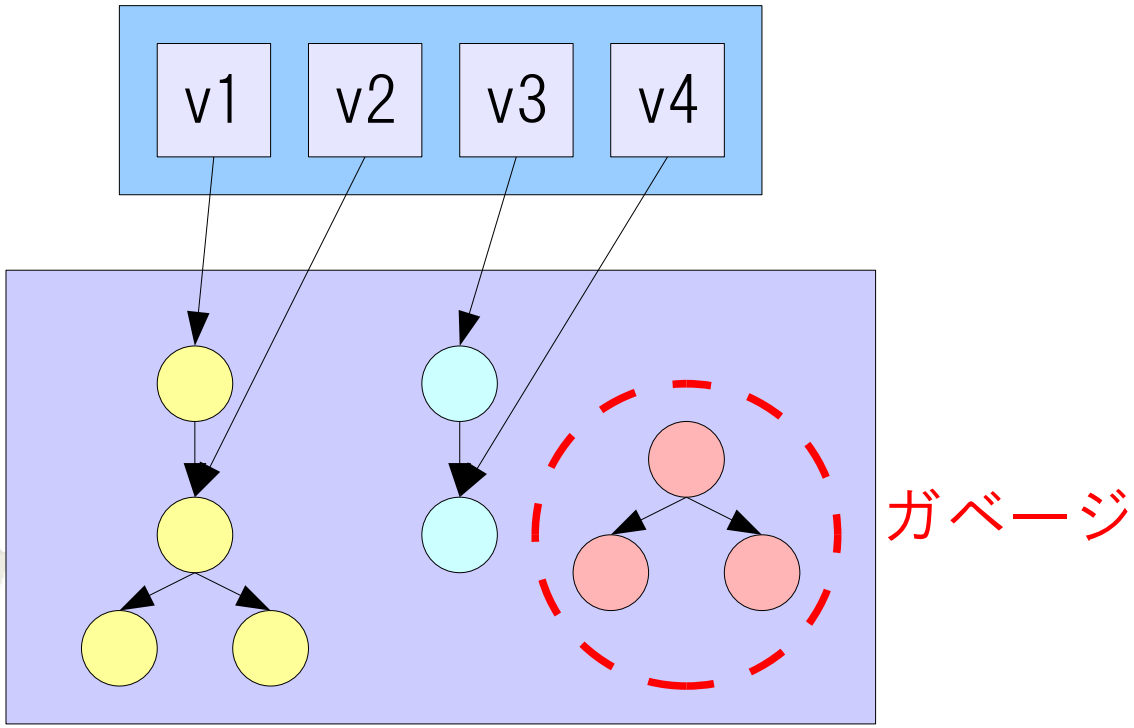
マルチスレッド環境下の参照カウントGCの為のカウンタ更新スレッド数の静的解析方法

千葉大学大学院融合科学研究科情報科学専攻
有馬 大介， 今泉 貴史

背景

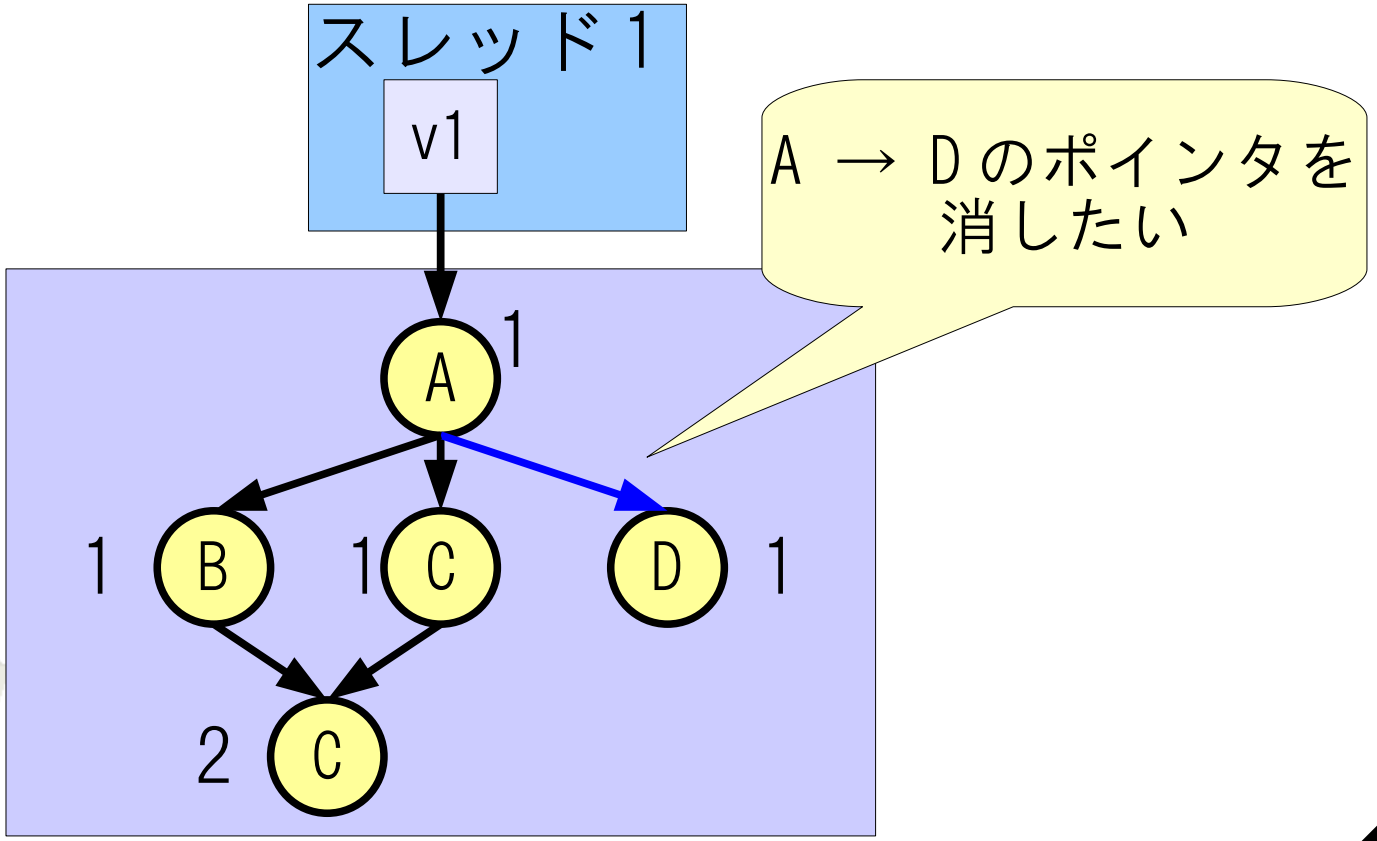
ガベージコレクション（GC）

- 不必要なメモリ領域を自動的に解放
  - 不要なメモリ領域　：　ガベージ
- アクセスできないデータをガベージと判定



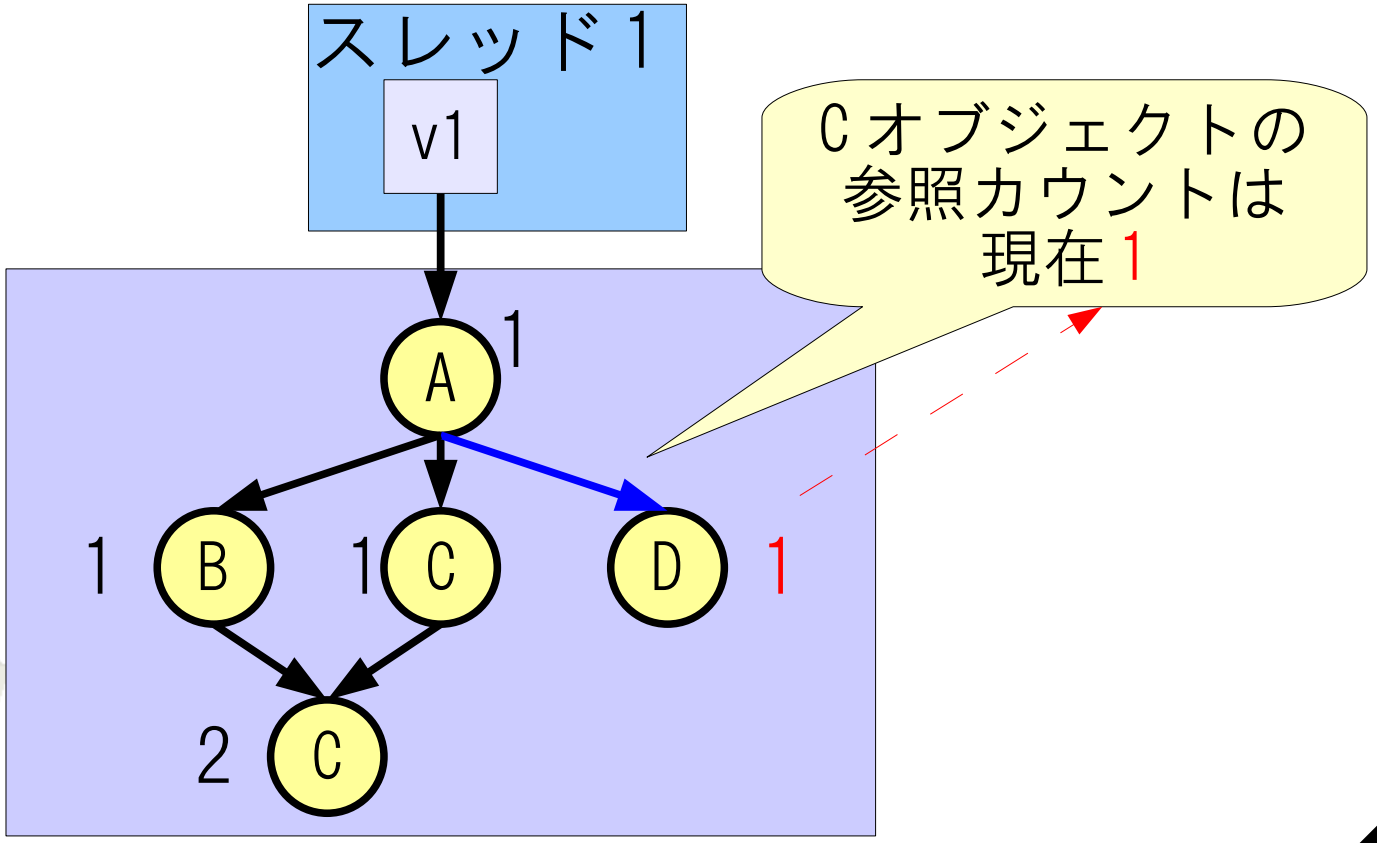
参照カウントGC

- オブジェクトの被参照数をカウント
  - ポインタ書き換え時にカウンタ更新
- カウント0 ： ガベージ



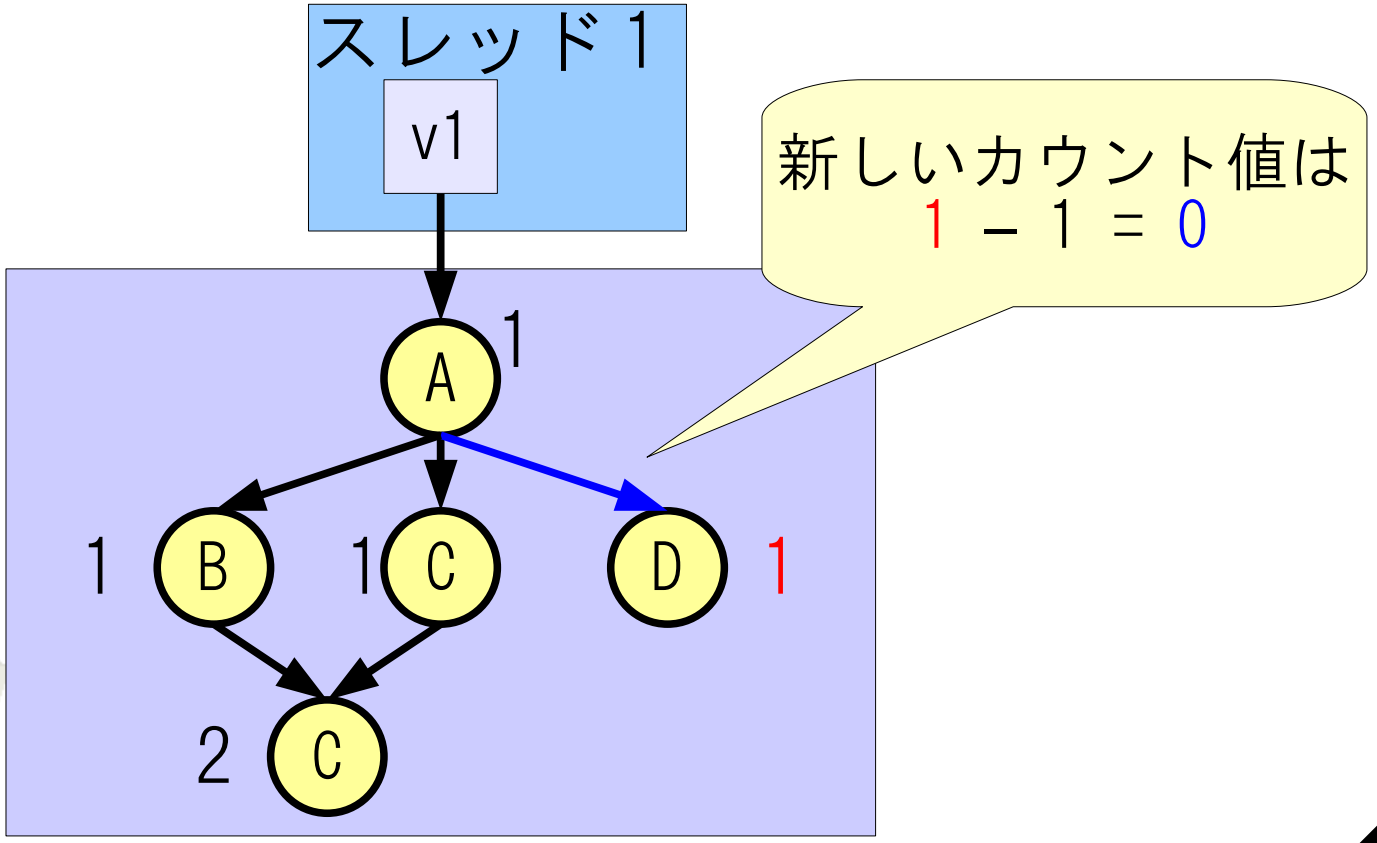
参照カウントGC

- オブジェクトの被参照数をカウント
  - ポインタ書き換え時にカウンタ更新
- カウント0 ： ガベージ



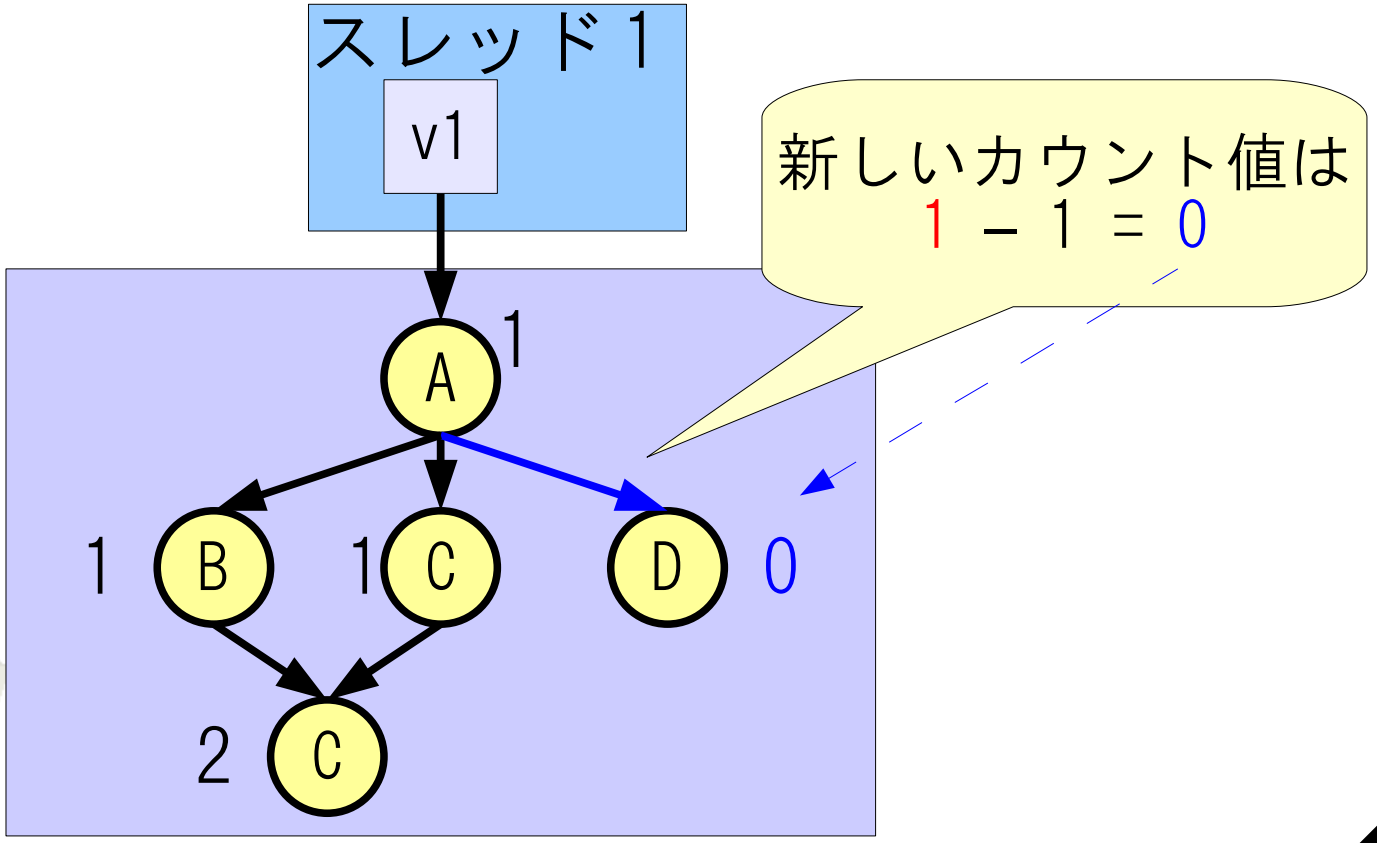
参照カウントGC

- オブジェクトの被参照数をカウント
  - ポインタ書き換え時にカウンタ更新
- カウント0 ： ガベージ



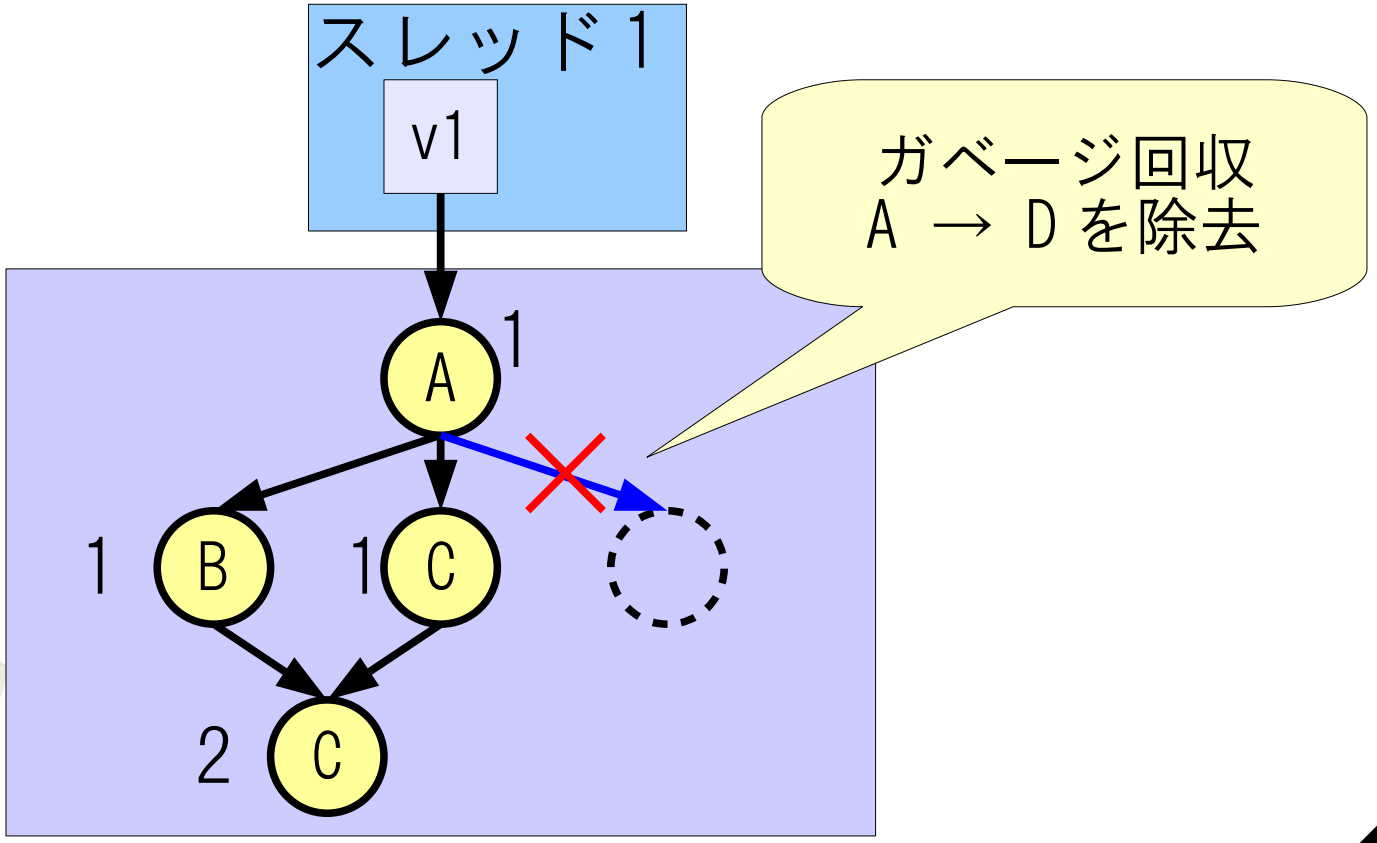
参照カウントGC

- オブジェクトの被参照数をカウント
  - ポインタ書き換え時にカウンタ更新
- カウント0 ： ガベージ



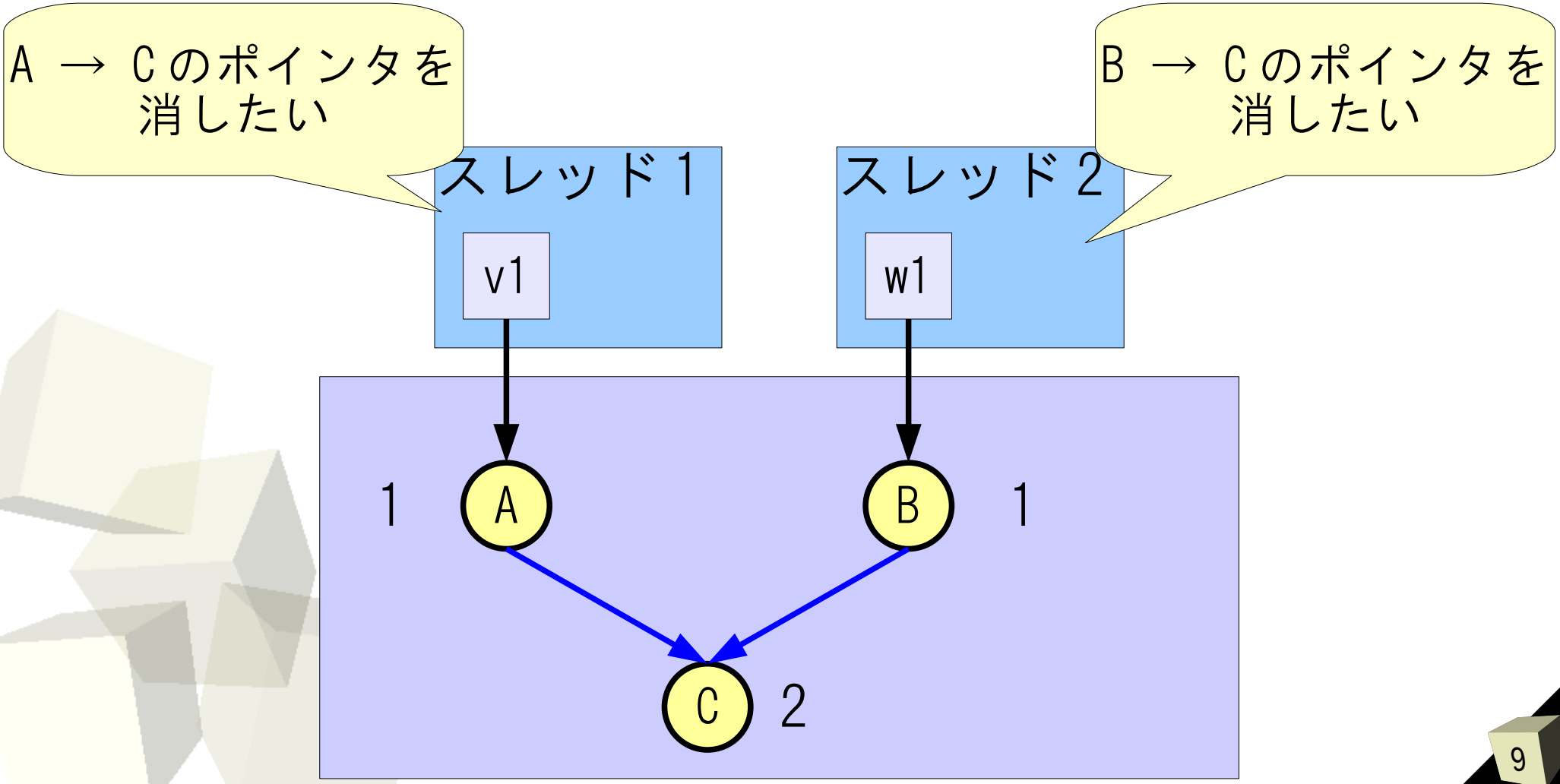
参照カウントGC

- オブジェクトの被参照数をカウント
  - ポインタ書き換え時にカウンタ更新
- カウント0 ： ガベージ



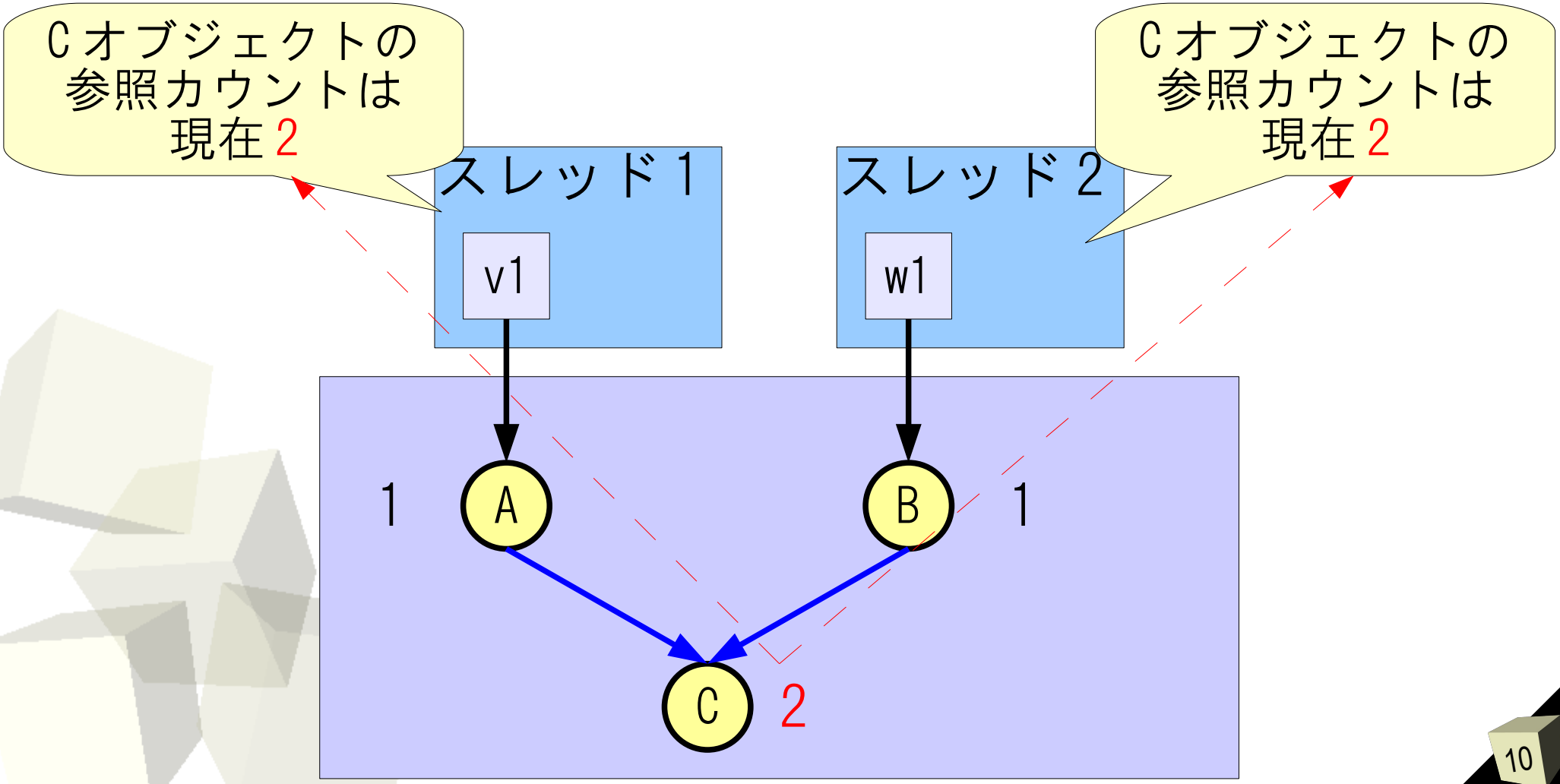
マルチスレッド環境下の参照カウントGC

- 複数のスレッドがオブジェクト操作
  - 参照カウンタ操作が競合する可能性



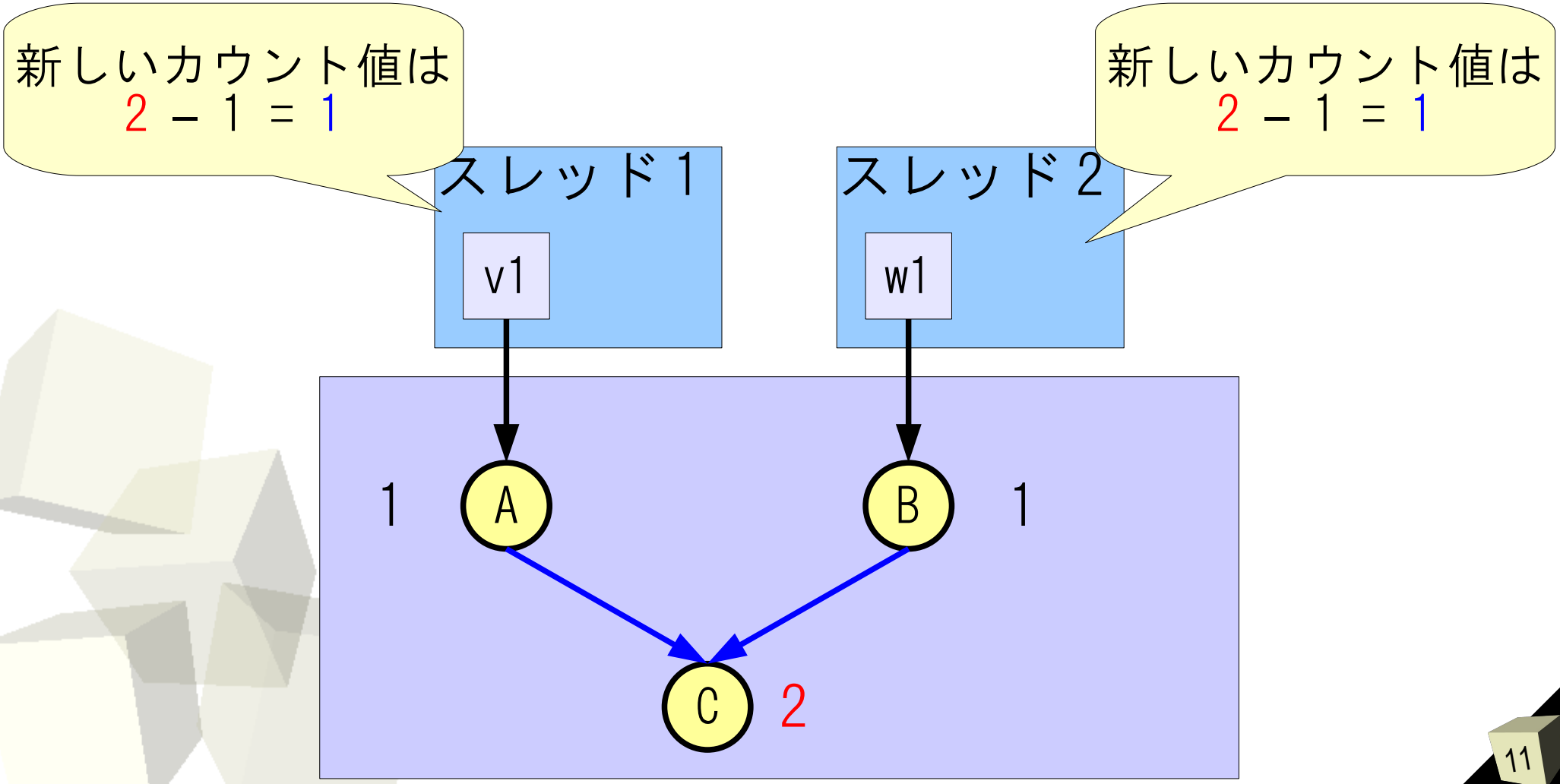
マルチスレッド環境下の参照カウントGC

- 複数のスレッドがオブジェクト操作
  - 参照カウンタ操作が競合する可能性



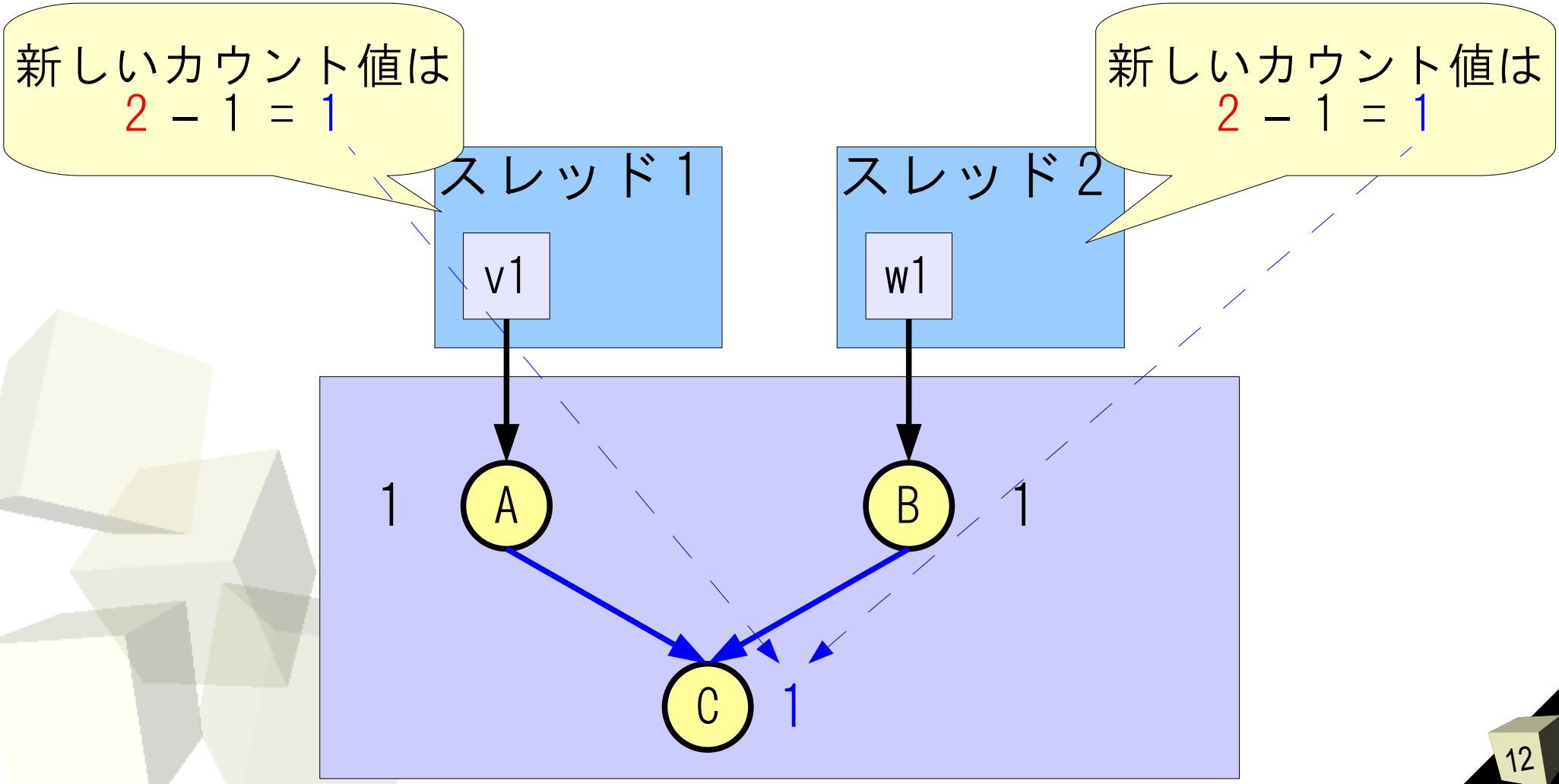
マルチスレッド環境下の参照カウントGC

- 複数のスレッドがオブジェクト操作
  - 参照カウンタ操作が競合する可能性



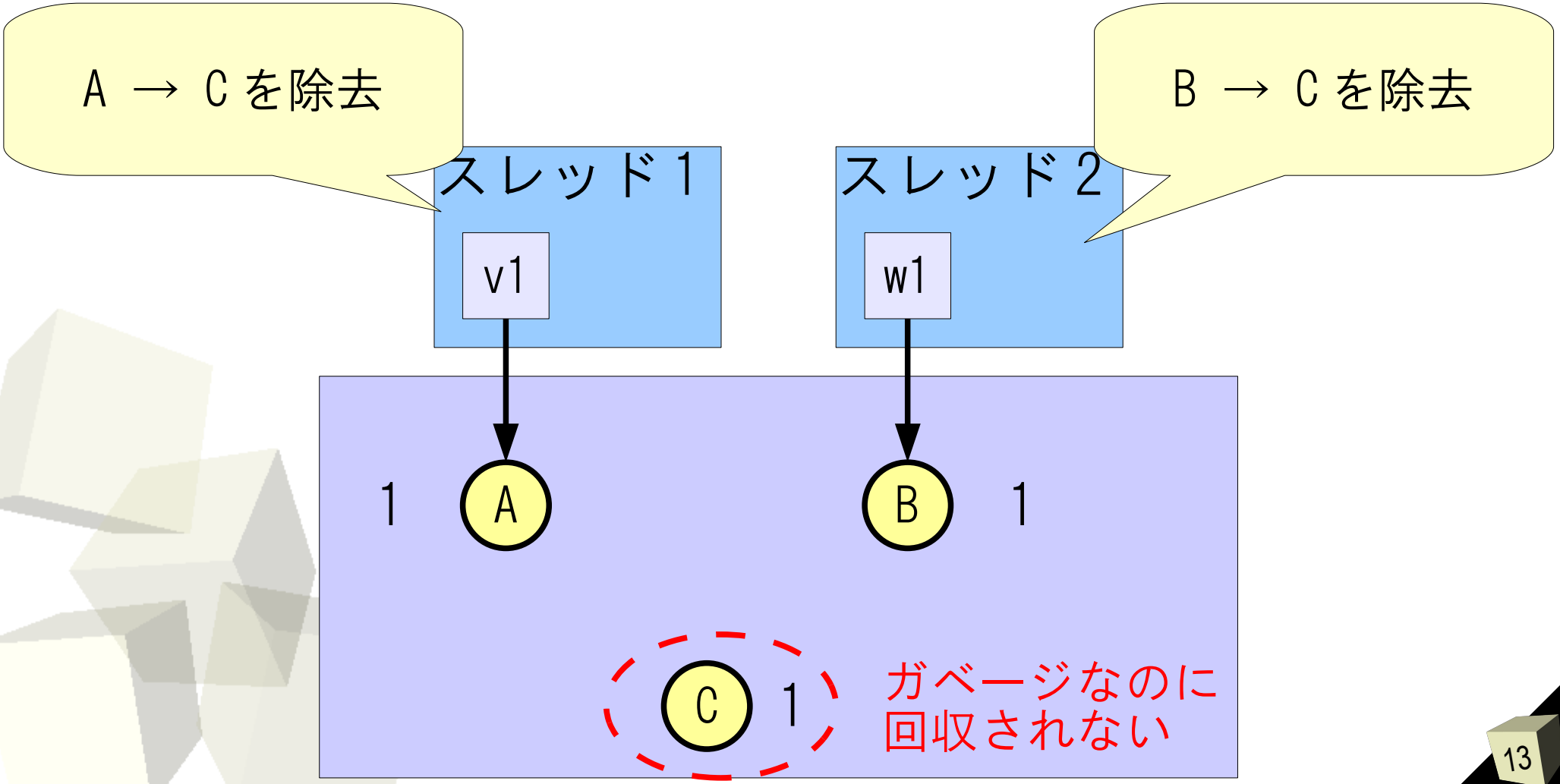
マルチスレッド環境下の参照カウントGC

- 複数のスレッドがオブジェクト操作
  - 参照カウンタ操作が競合する可能性



マルチスレッド環境下の参照カウントGC

- 複数のスレッドがオブジェクト操作
  - 参照カウンタ操作が競合する可能性



マルチスレッド環境下の参照カウントGC

- 競合を避けるために
  - カウンタ更新にアトミック操作を利用
  - しかしアトミック操作は高コスト
- 競合しないオブジェクトもあり
  - アトミック操作を用いるのは無駄
  - できればアトミック操作を使わず、カウンタ更新したい

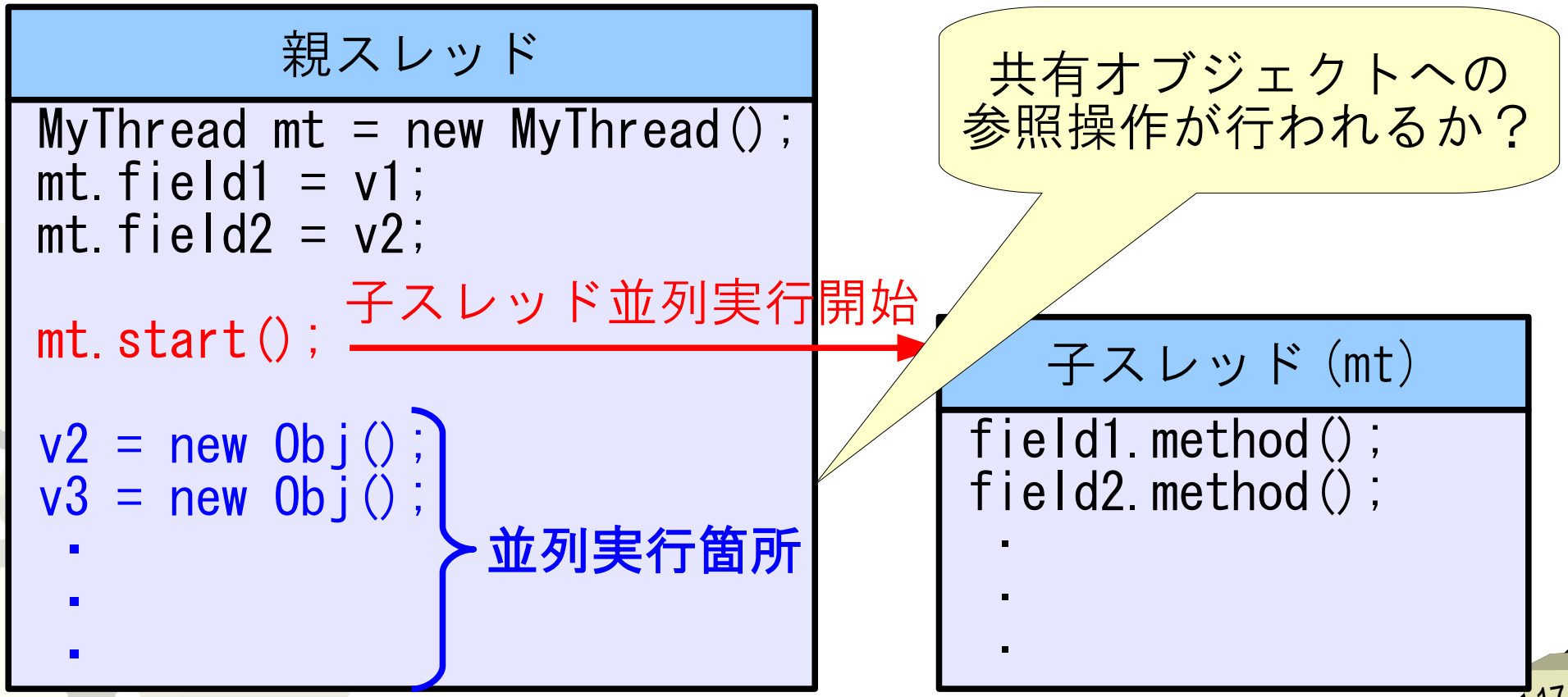
提案：カウンタ更新スレッド数の静的解析

- 静的解析によってオブジェクトを区別
  - カウンタ更新が競合する可能性があるもの
  - 競合しないもの
- コンパイル時、参照カウンタ更新にアトミック操作が必要無い箇所を検出
  - 無駄なアトミック操作を省き、
    プログラム処理を高速化

解析方法の概要

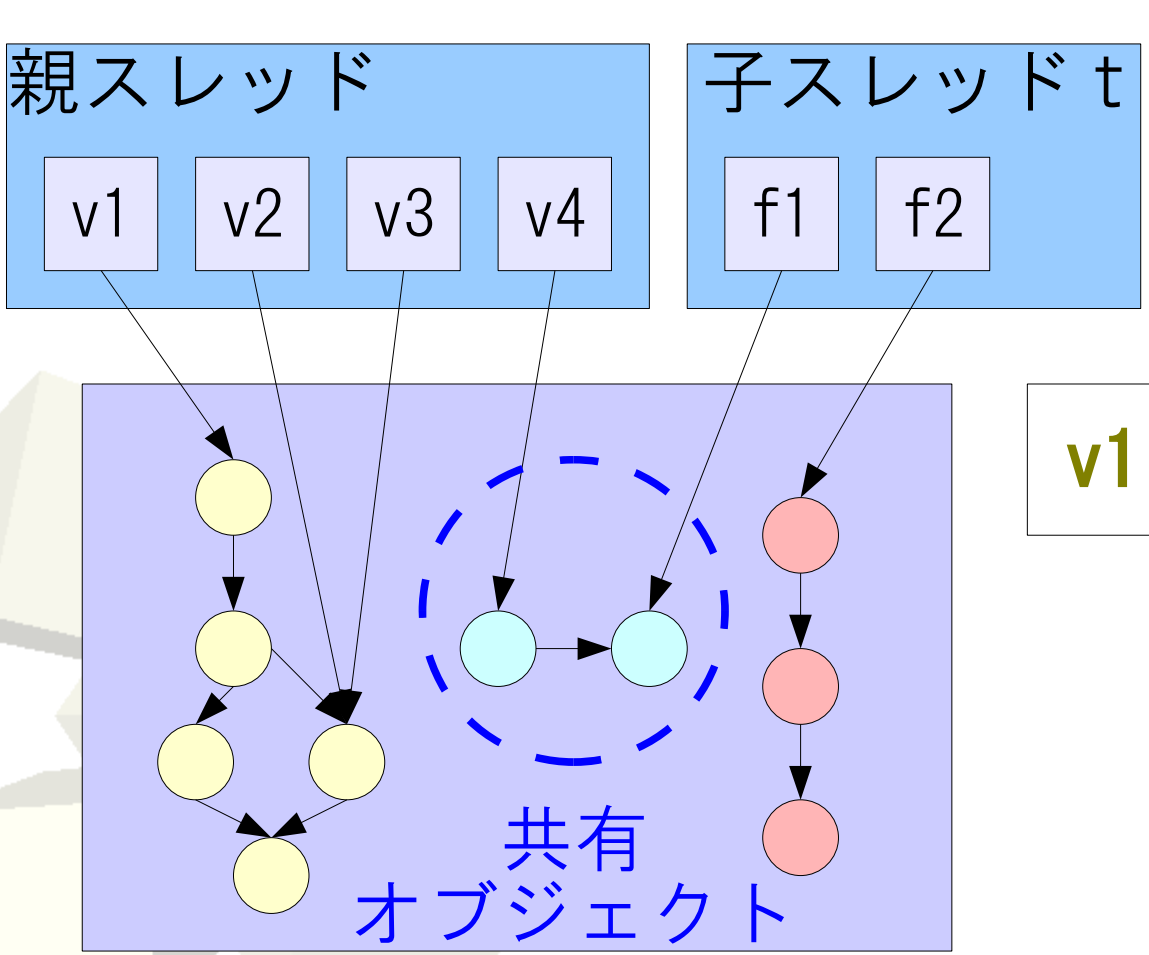
解析方法の概要

- 子スレッドとの並列実行箇所に注目
  - 共有オブジェクトへの参照を操作するかを確認



解析方法の概要

■親・子スレッドで共有されている
オブジェクト群に注目

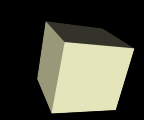


並列実行箇所で
使用されるか？

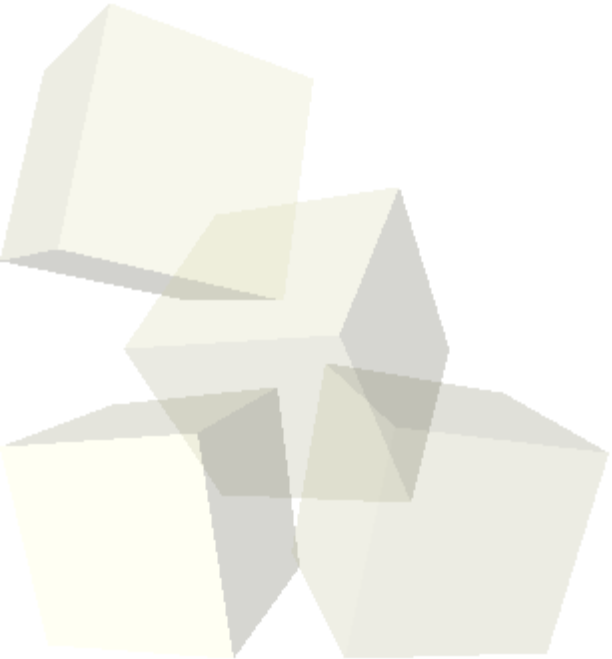
v1 v2 v3	v4 t.f1	t.f2
単一	単一？ 複数？	単一

解析方法の概要

- 制御フローグラフの作成
  - プログラム処理の流れを調べる
- 別名解析
  - 同じオブジェクト群を指す変数をグループ分け
- カウンタ更新スレッド数の解析
  - 共有オブジェクトを指すグループが並列実行箇所で使われるか？

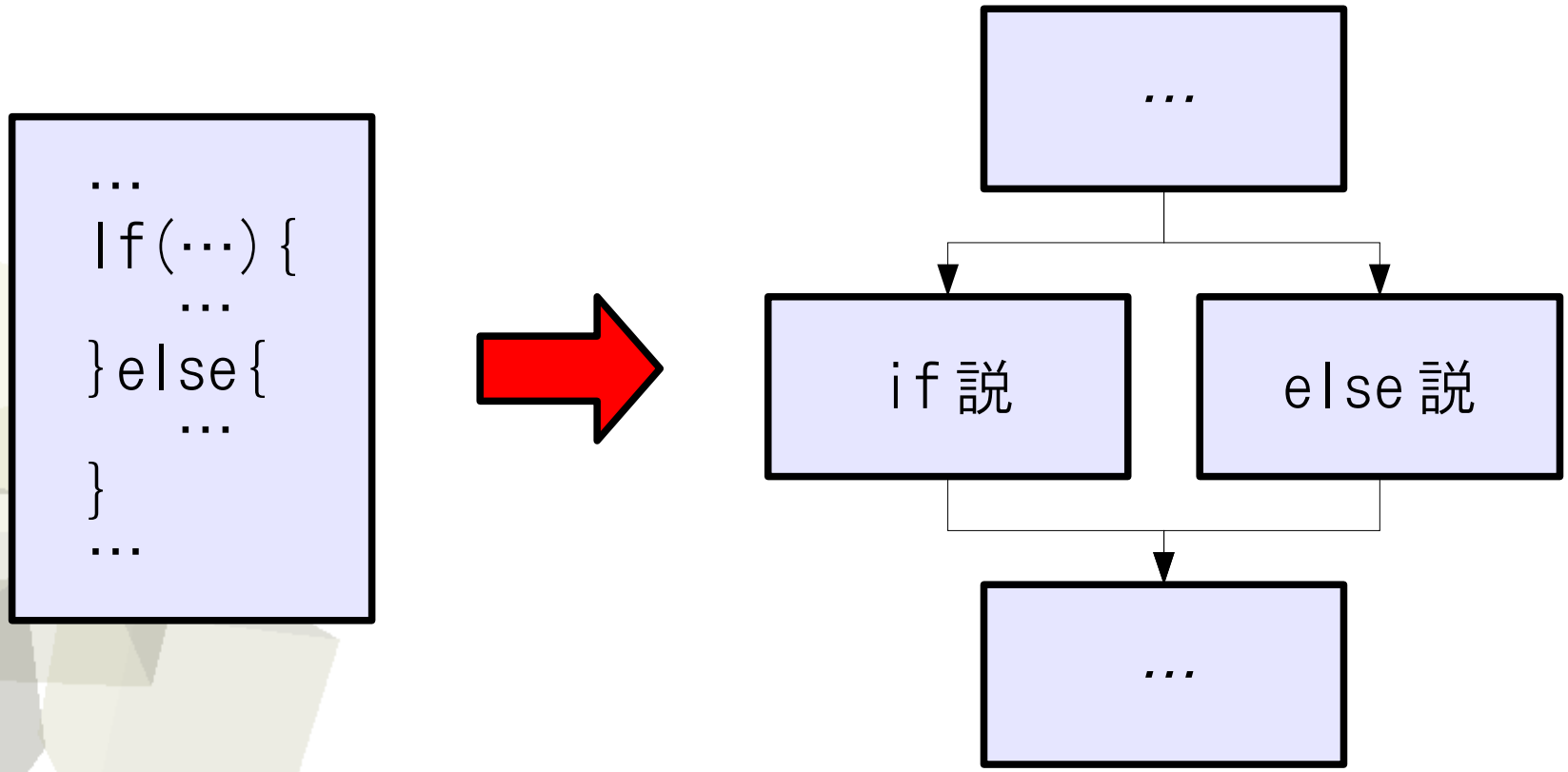


制御フローグラフの作成



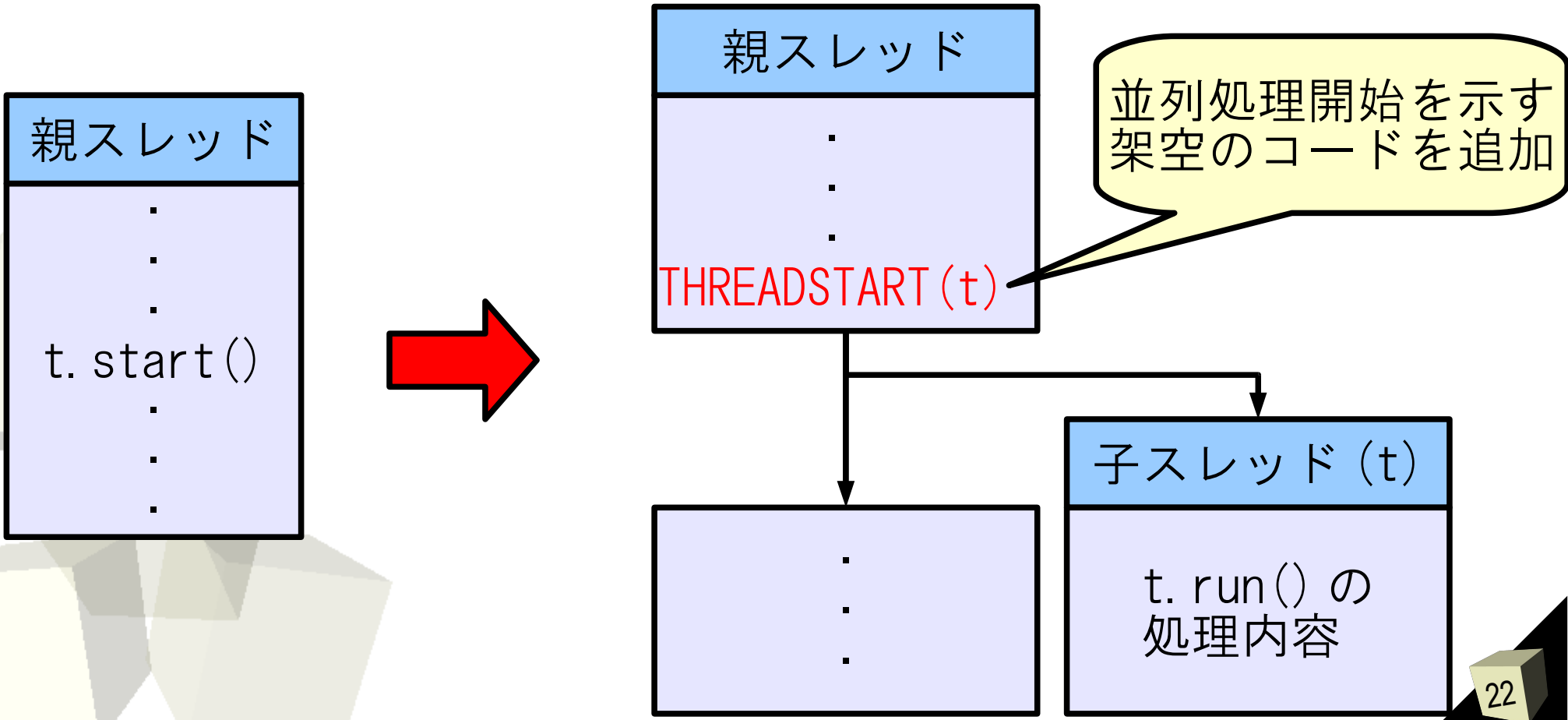
制御フローグラフの作成

- スレッドの並列実行開始地点を調べる
- 変数のグループ分けを行う際にも利用
  - プログラム処理の分岐等を考慮



スレッドの並列実行開始地点

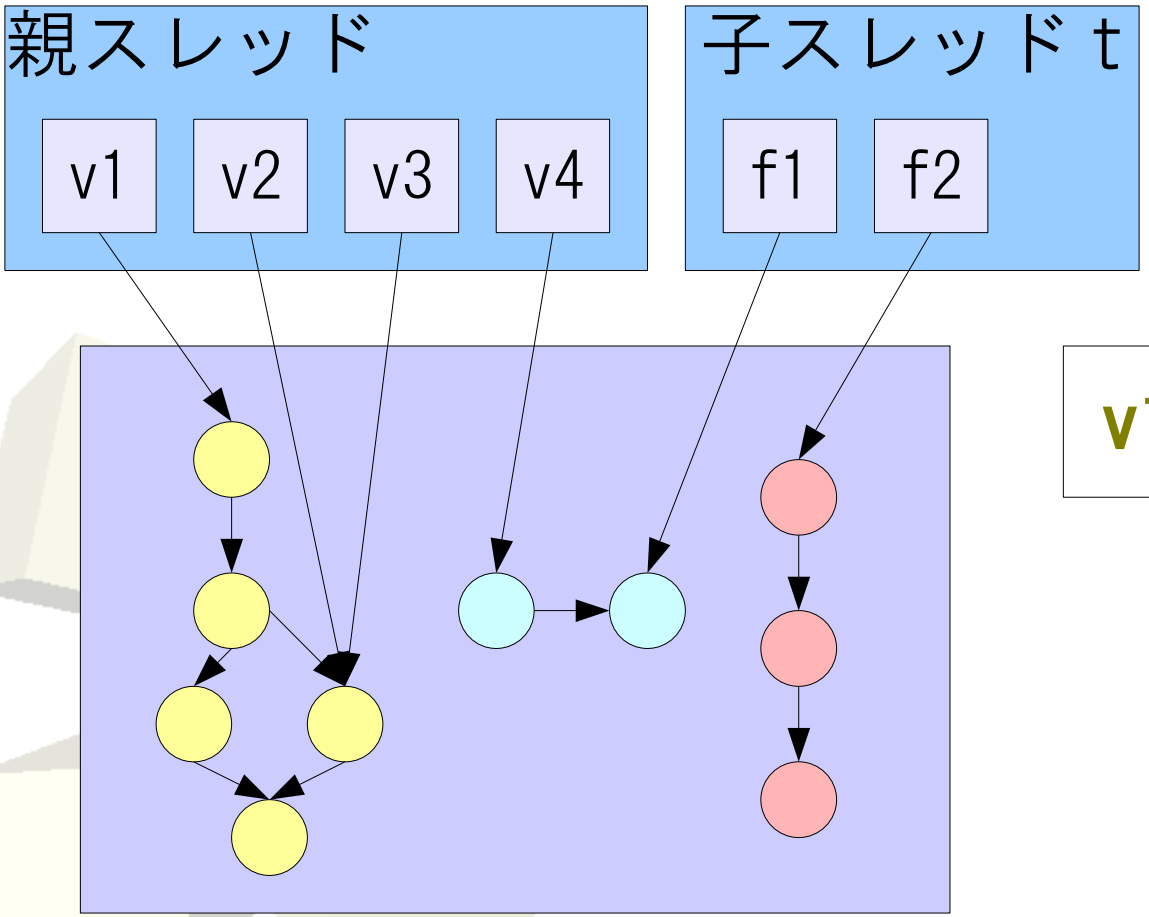
- 親スレッドの処理内容ブロックと
  子スレッドの処理内容ブロックを分離



别名解析

変数の別名解析

- 同一のオブジェクト群を指す変数をグループ分け



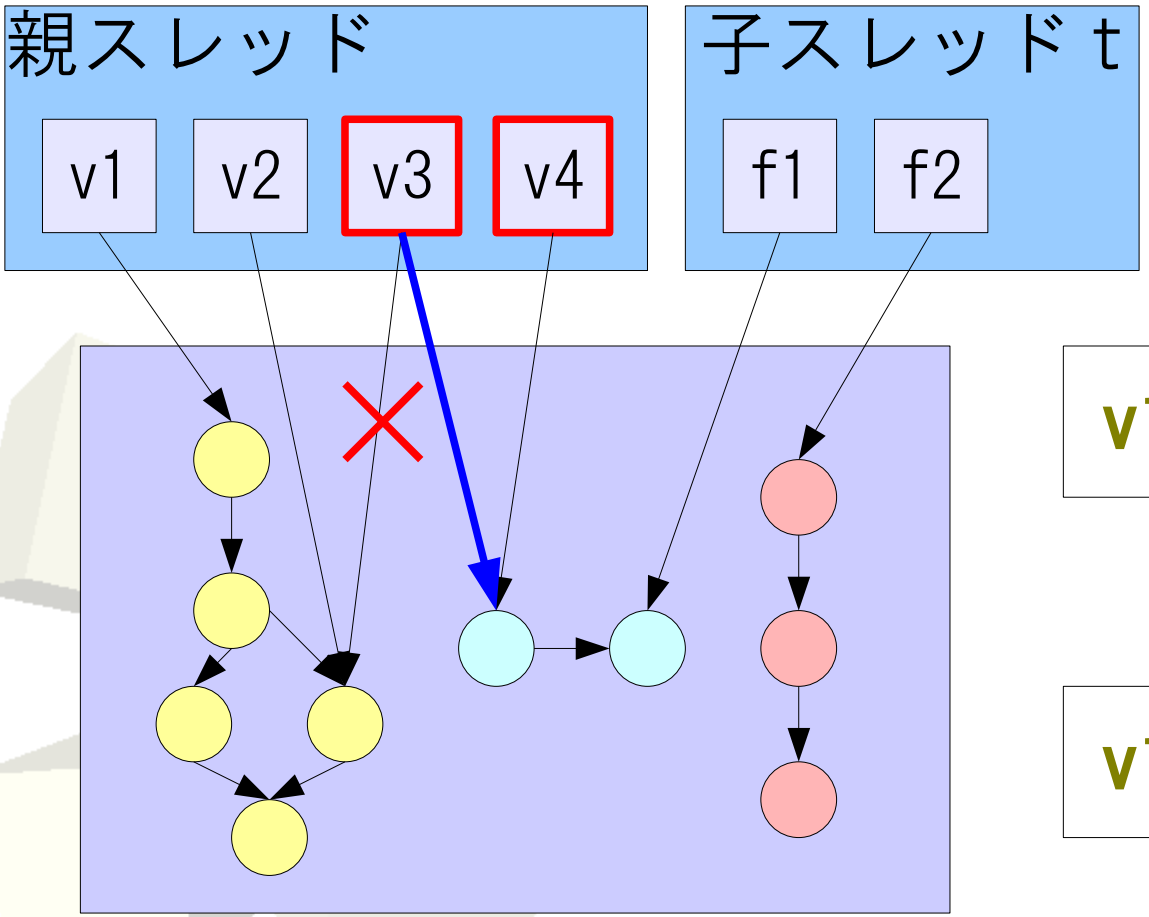
v1 v2 v3	v4 t.f1	t.f2

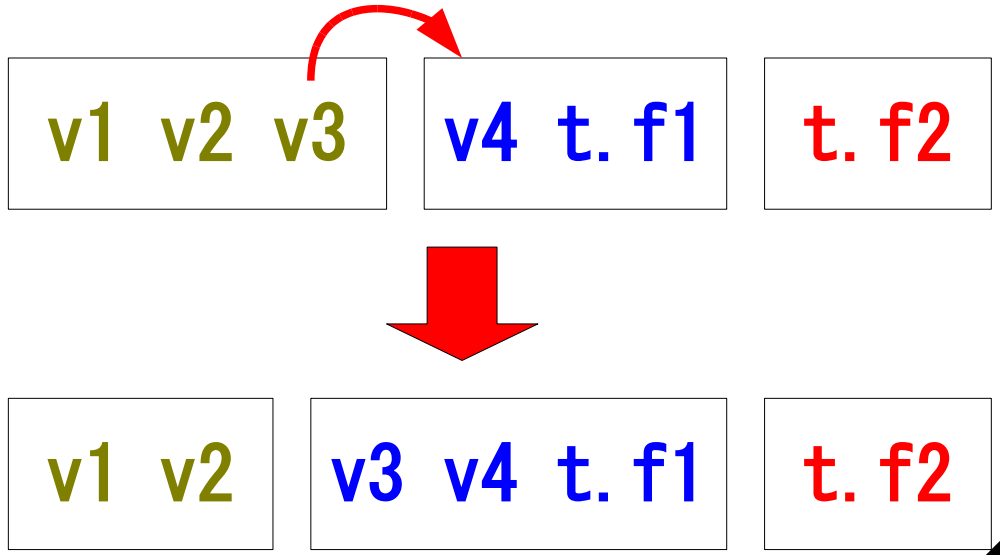
変数の別名解析

- 各文で成り立つ別名関係を算出
- 算出に用いるもの
  - 一つ前の文における別名関係
  - 代入文

例：代入文 v3 = v4 による別名関係の変化

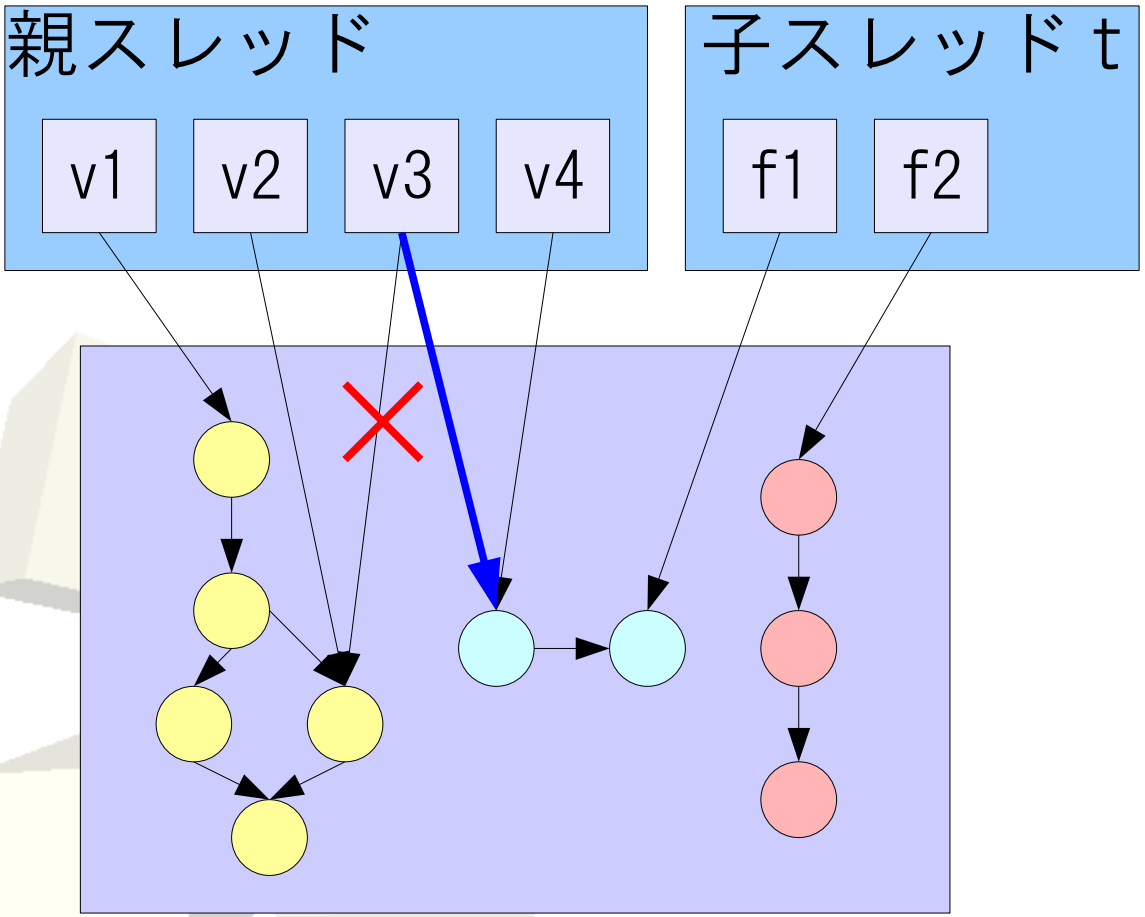
- v3 と v4 は同じオブジェクト群を指す
  - v3 を v4 のグループに移動





実際の別名関係

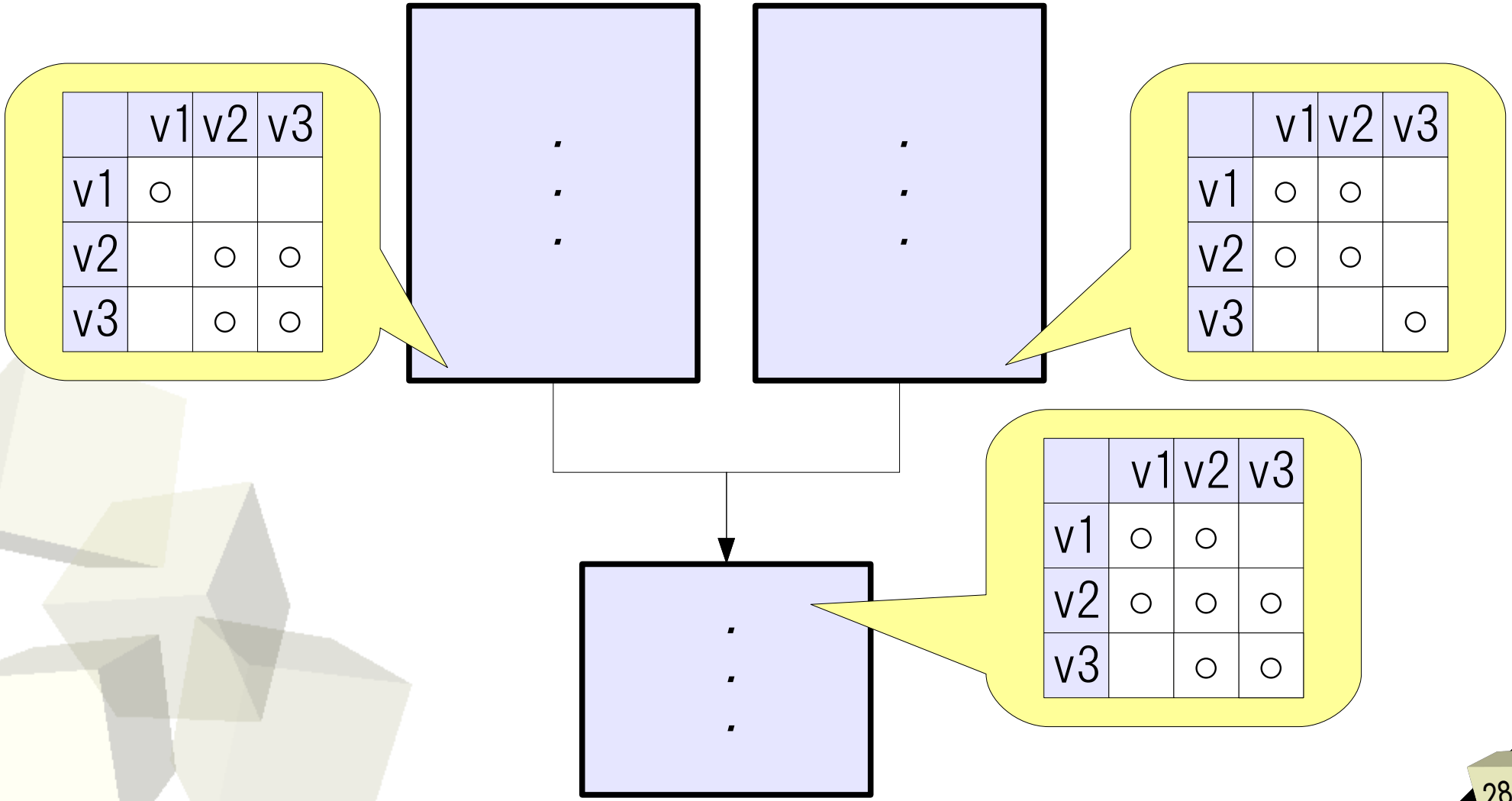
- 表形式で別名関係を管理

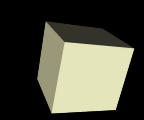


	v1	v2	v3	v4	t. f1	t. f2
v1	○	○				
v2	○	○				
v3			○	○	○	
v4			○	○	○	
t. f1			○	○	○	
t. f2						○

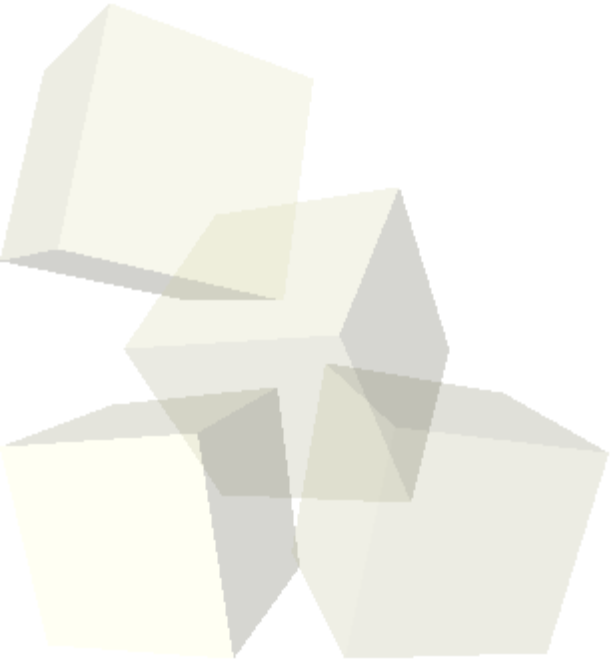
実際の別名関係

- ブロックの合流点
  - 合流前のブロックの情報を統合



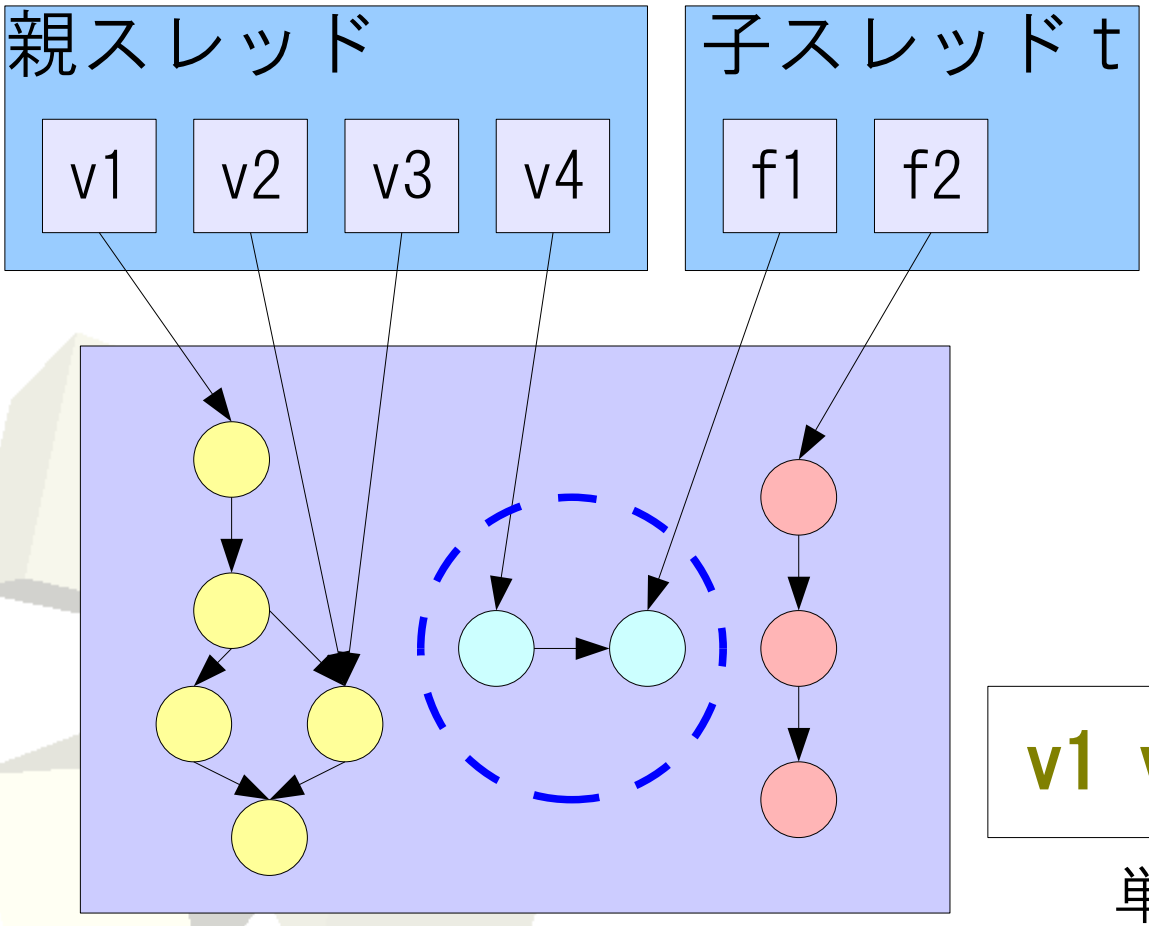


カウンタ操作スレッド数解析



カウンタ操作スレッド数解析

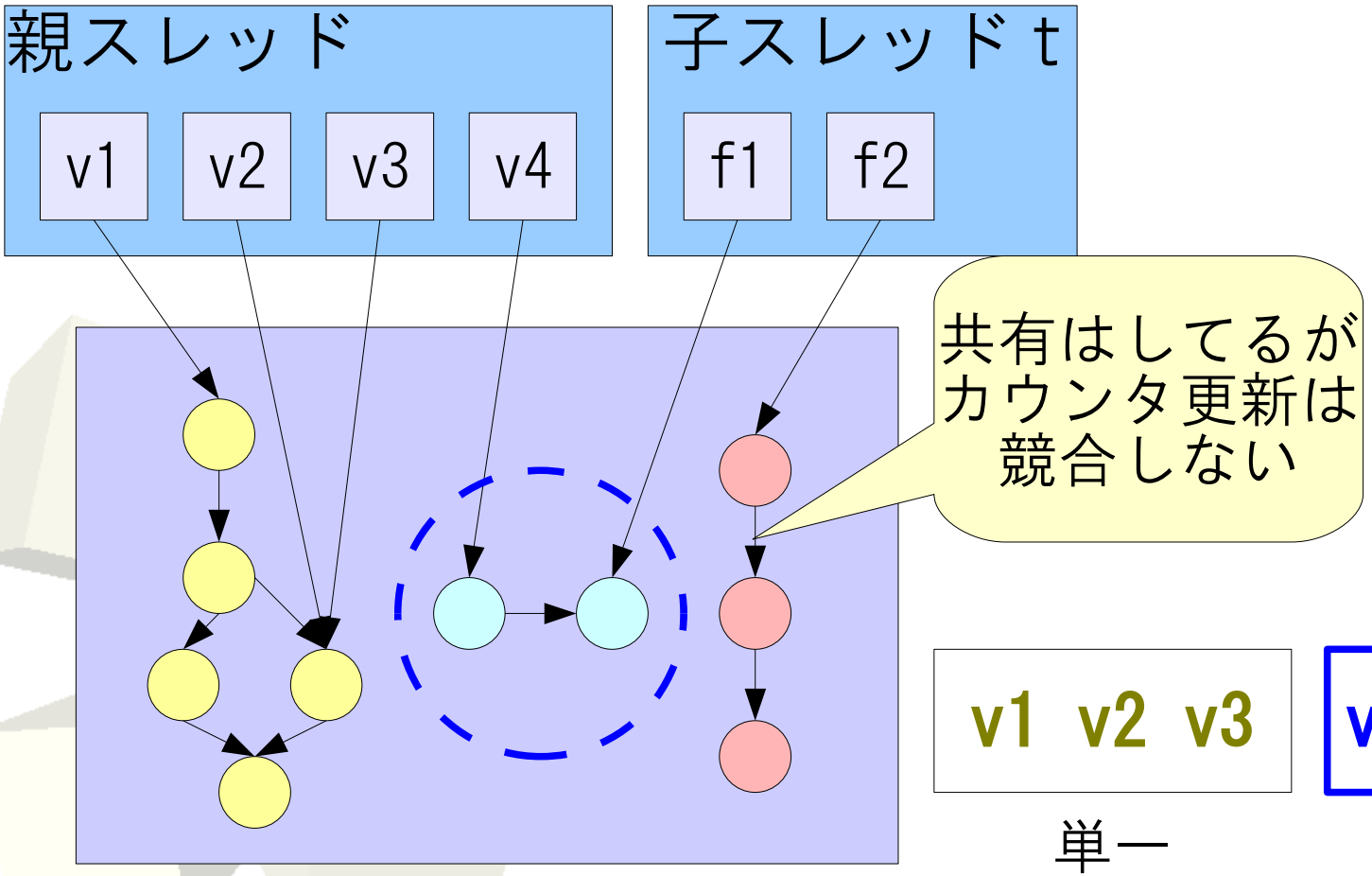
- カウンタ操作するスレッドが単一か複数か
  - グループ毎に算出



v1 v2 v3	v4 t.f1	t.f2
単一	?	単一

カウンタ操作スレッド数解析

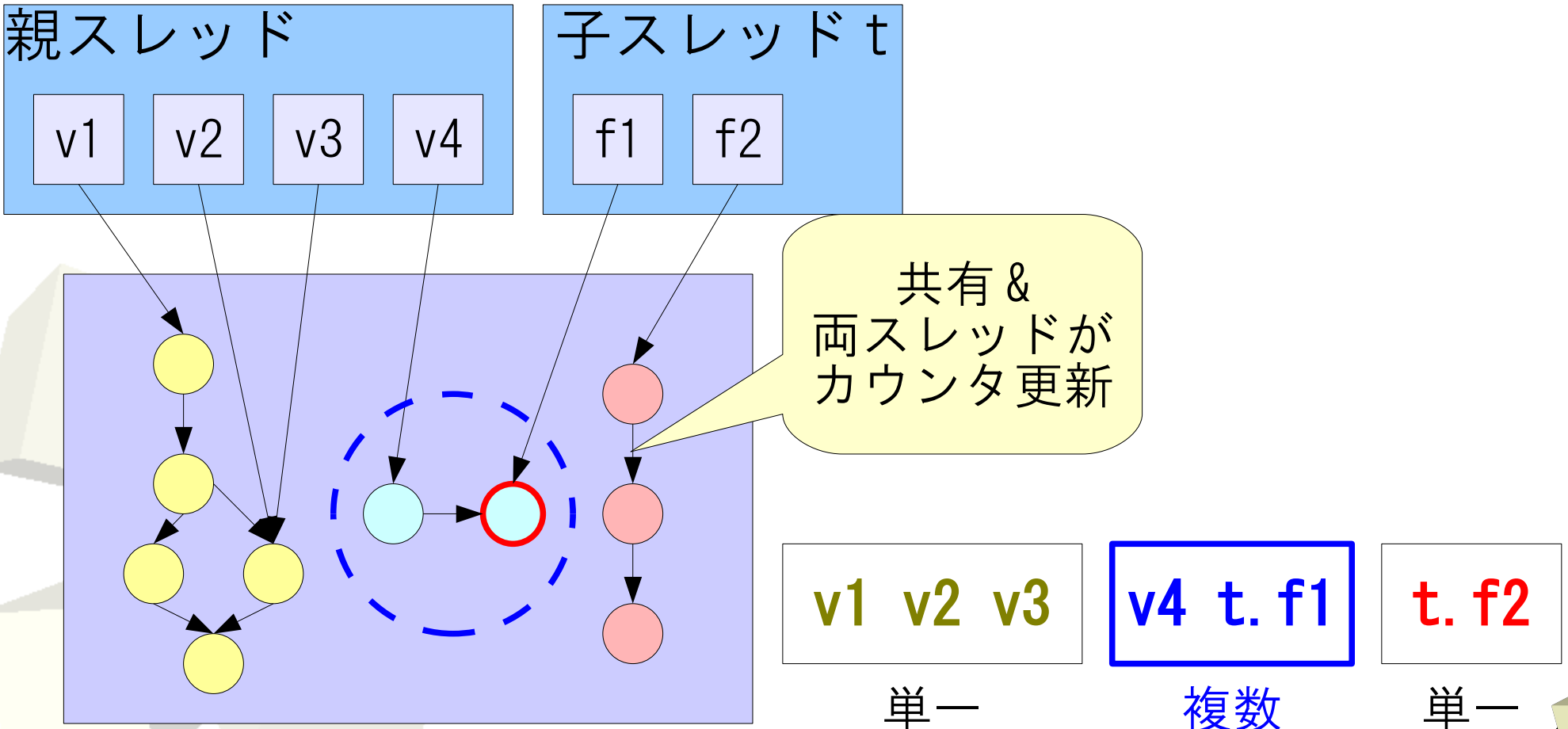
- 並列実行中、t.f1 が指すオブジェクト群の参照カウンタを更新しない場合



v1 v2 v3	v4 t.f1	t.f2
単一	単一	単一

カウンタ操作スレッド数解析

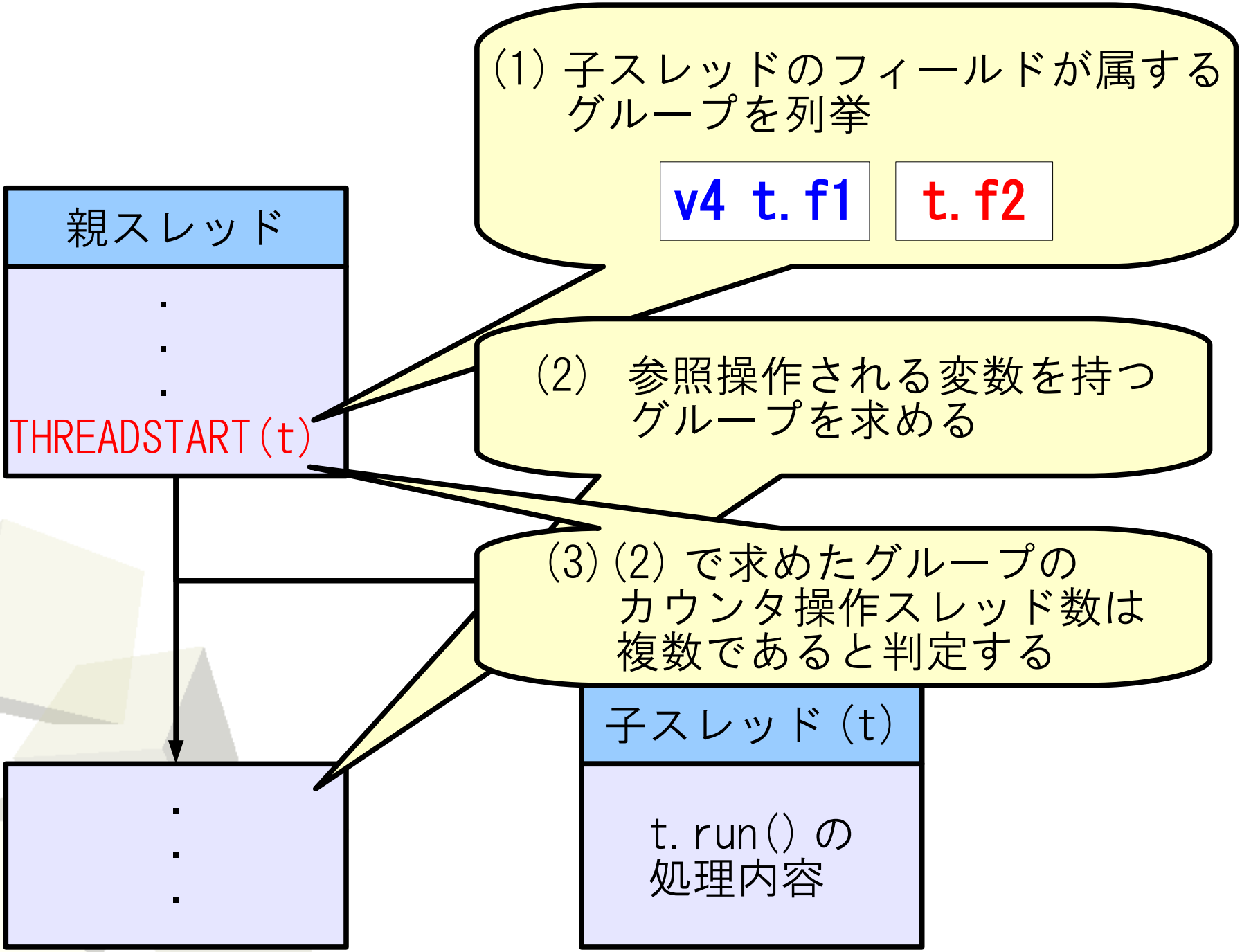
- 並列実行中、t.f1 が指すオブジェクト群の参照カウンタを更新する場合



カウンタ操作スレッド数解析

- 並列実行箇所から算出
  - 共有オブジェクトを指しているグループが使われているかを確認
- 代入文から算出
  - グループの入脱退

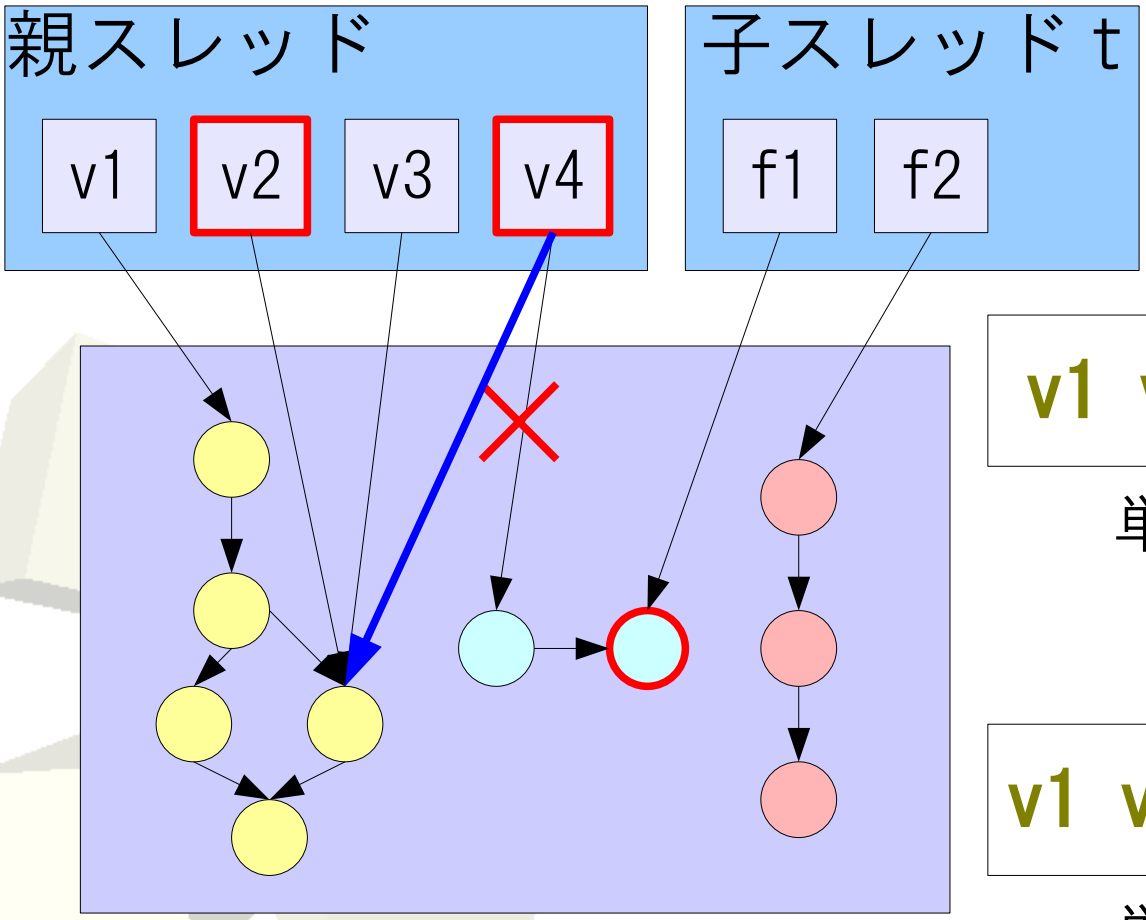
並列実行箇所による操作スレッド数の算出

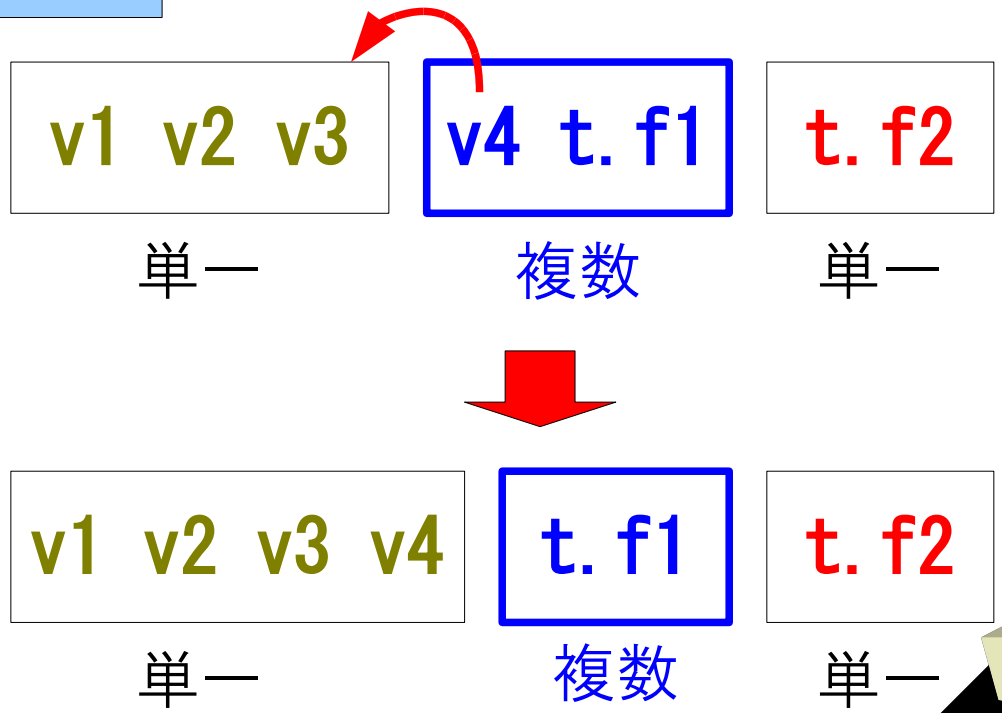


代入文による操作スレッド数の算出

■例　：　v4 ＝ v2

・v4 は v2 と同じオブジェクト群を指す

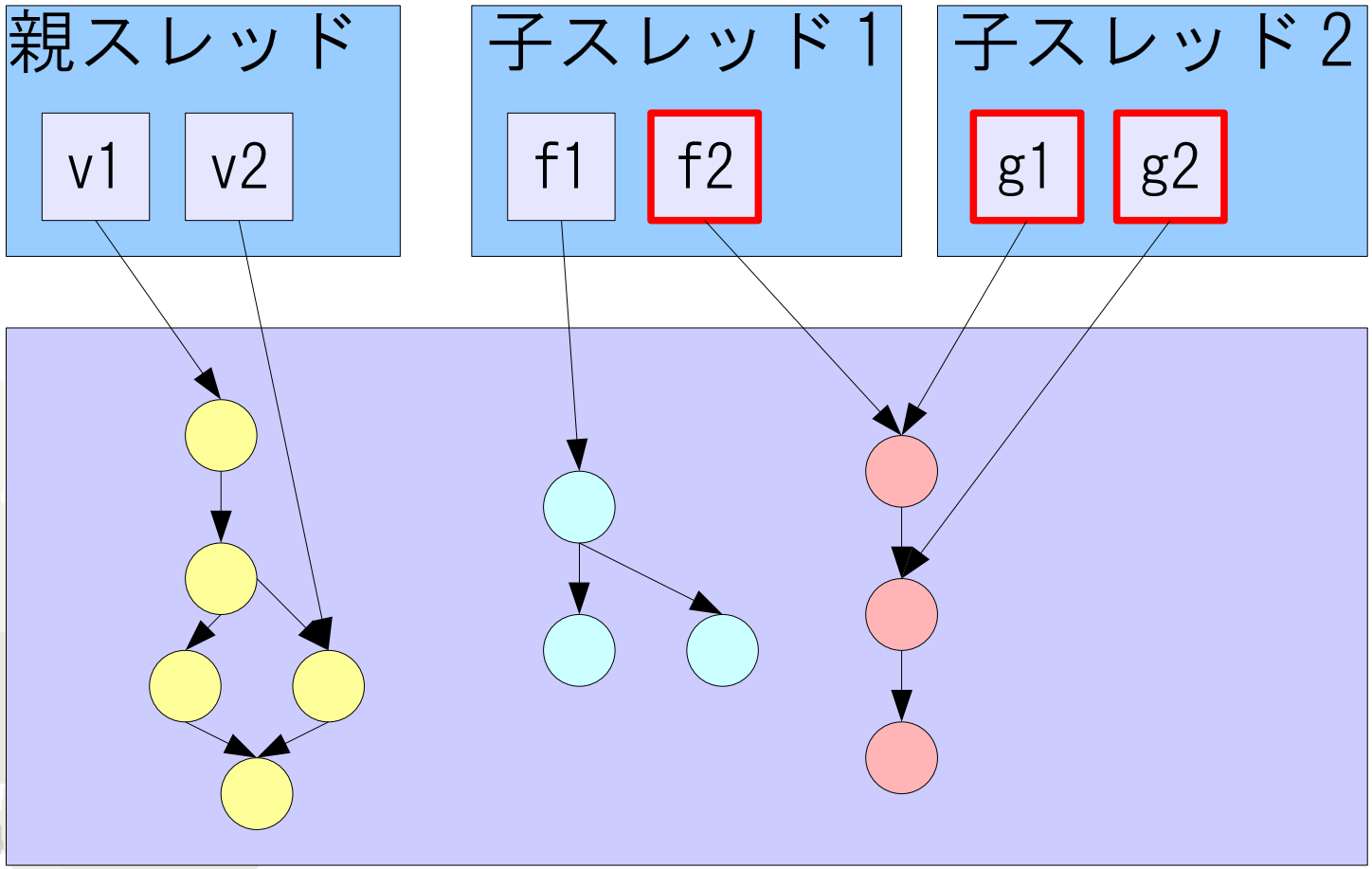




解析結果

子スレッドが複数いても対応可能

- f2, g1, g2 が並列実行箇所で用いられる場合、競合可能性ありと判定



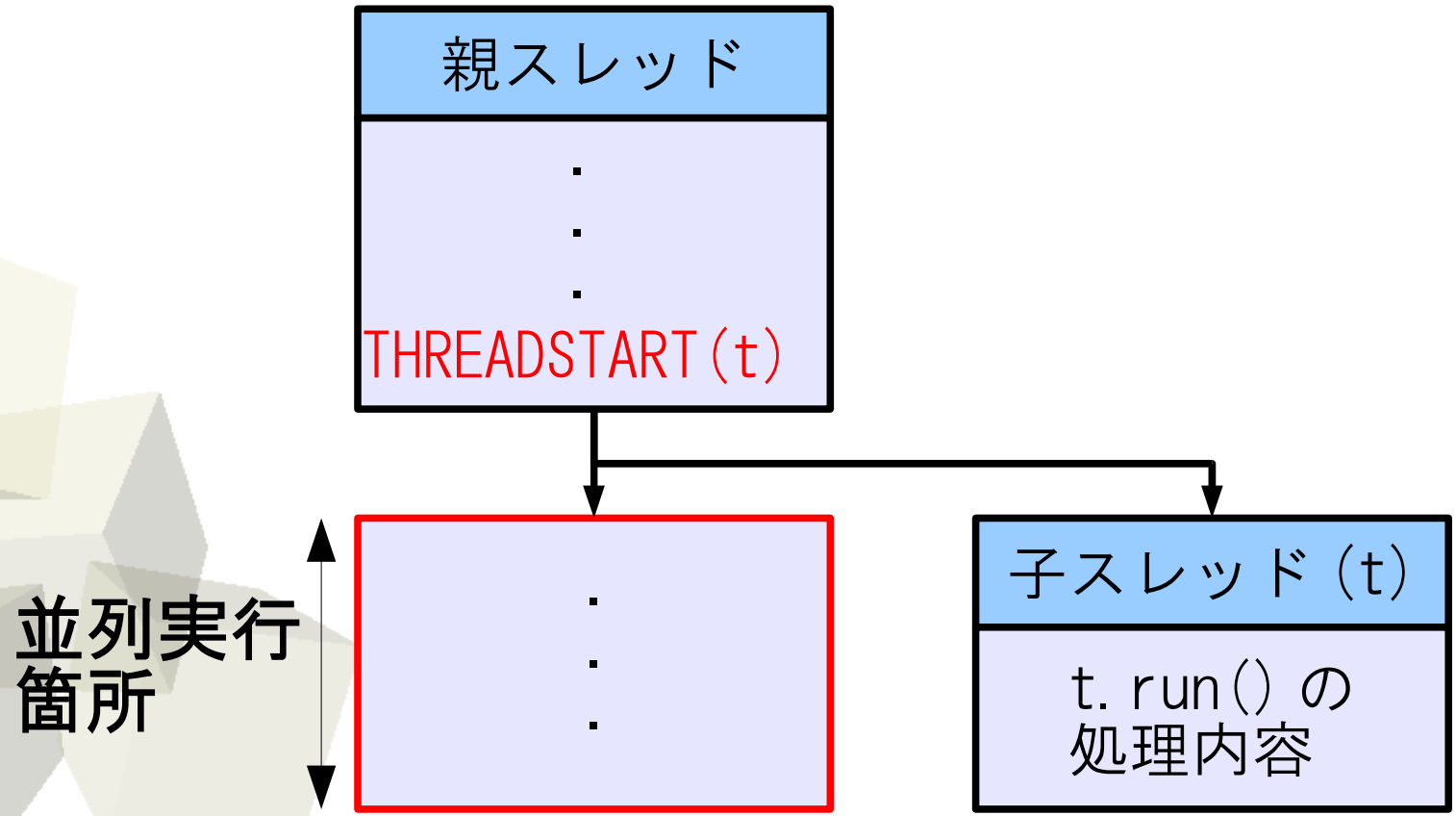
考察

カウンタ操作スレッド数解析の解析精度

- 今回の解析で注目した点
  - 親スレッドの並列実行箇所で子スレッドと同じグループの変数が使われるか？
- 別名解析の結果に依存
  - プログラムによって異なる
- 並列実行箇所の長さに依存
  - 子スレッドのフィールドと同一グループの変数が使われる可能性に関係

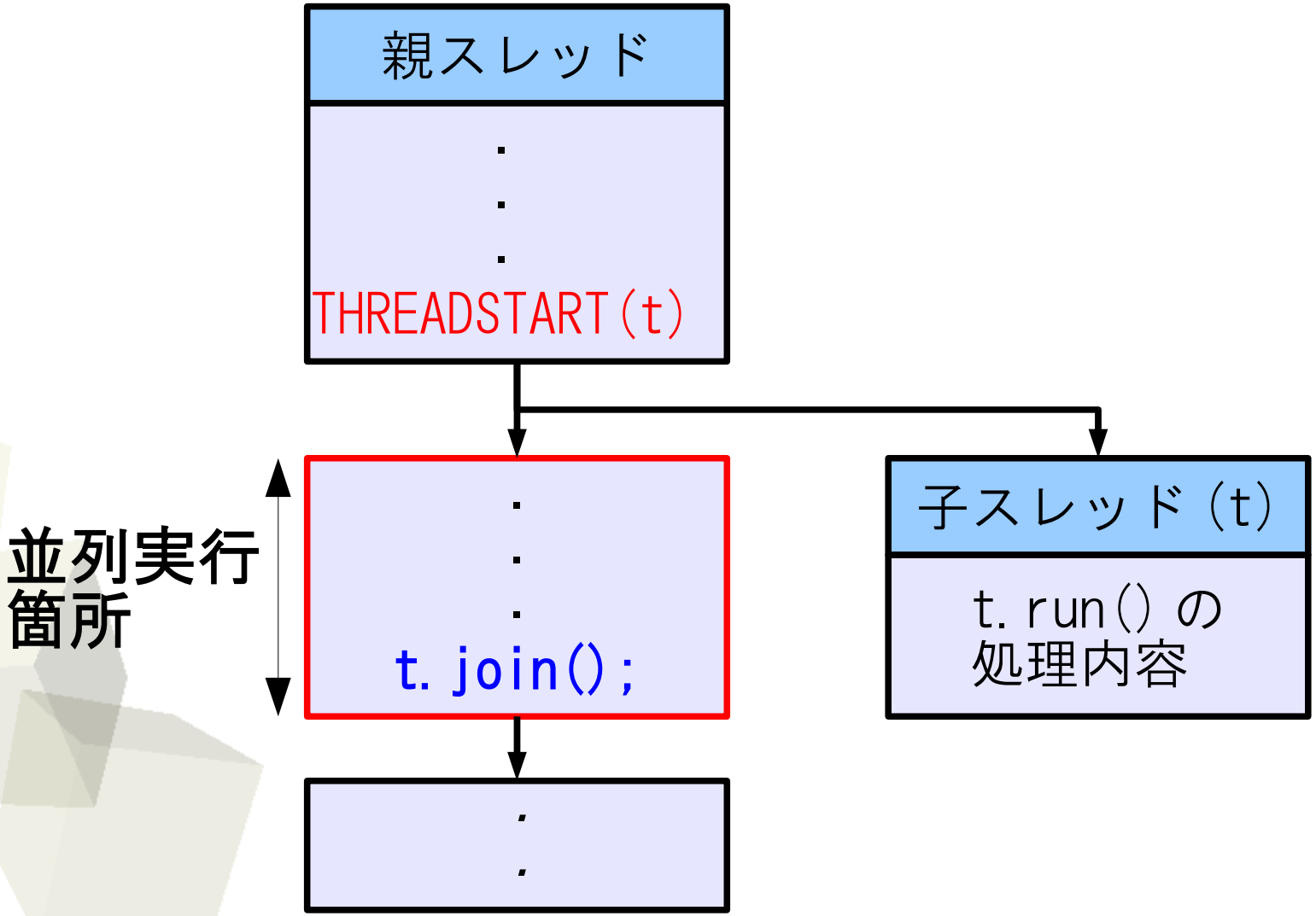
並列実行箇所の決定

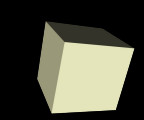
- 子スレッド実行開始以降、
  全てを並列実行箇所と見なしている
  - 中には並列実行しない箇所も



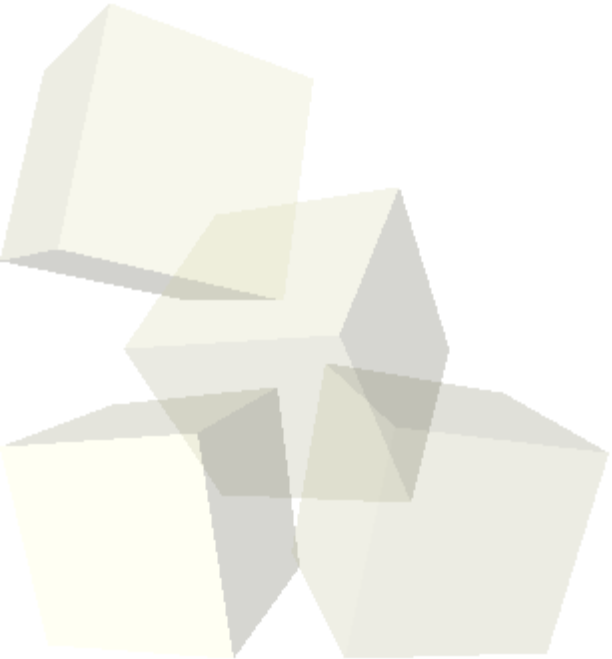
より正確な並列実行箇所の決定

- 子スレッドを制御する文に注目
  - joinメソッド ： スレッドの停止を待つ





終わりに



まとめ

- カウンタ更新にアトミック操作が必要なオブジェクトの静的解析方法の提案
  - 制御フローグラフの作成と並列実行箇所の検出
  - 同じオブジェクト群を指している変数をグループにまとめる
  - 並列実行箇所で複数スレッドから参照操作されるグループを求める

今後の課題

- 解析精度の向上
  - より正確な並列実行箇所の決定
    - →子スレッドの処理完了待ち
    - →子スレッドの処理停止・再開
- 実言語上への実装
  - 解析結果を元にし、適切なカウンタ更新を行う参照カウントGCを搭載したプログラミング言語処理系

ご清聴ありがとうございました