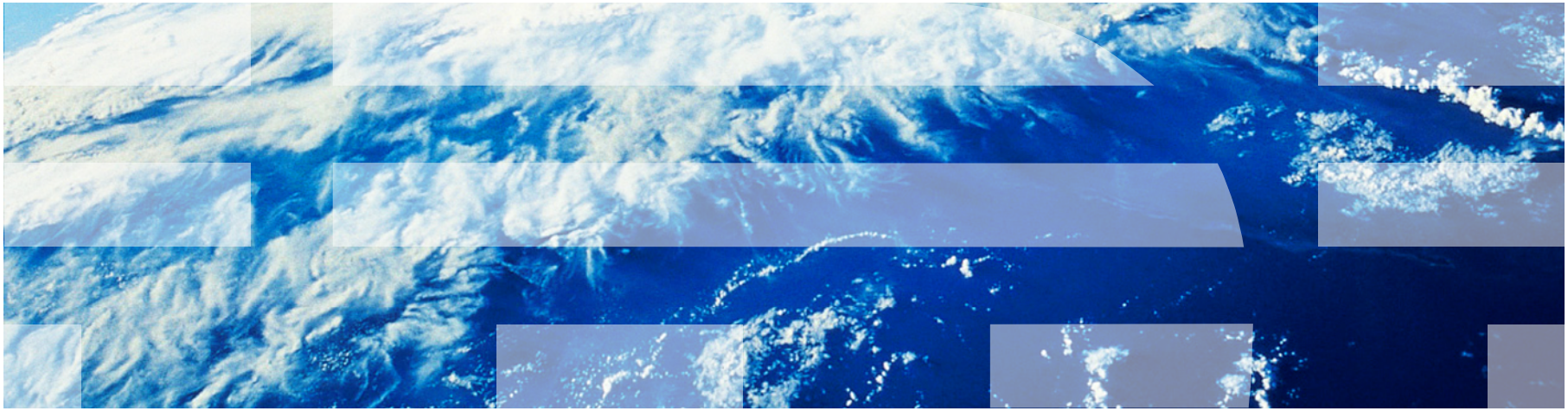
IBM

Using Program Closures to Make an API Thread Safe

Eitan Farchi, Itai Segall, Joao M. Lourenco, Digo Sousa, PADTAD, July 2012



High Level Data Races

- Identifies shared variables transactions intended to be atomic that are not atomic
- Introduced by Artho, et al in 2003
- Definition
  - A view V of thread t is a set of shared variables accessed by t under a lock. The set of all views of thread t is denoted by V(t)
  - A maximal view of a thread t, Vmax is a view not contained in any other view of t
  - Thread t1 and t2 are said to have a High Level Data Race if the intersection of a maximal view Vmax of t1 with the views of t2, V(t2), does not form a chain
    - It forms a chain if every two sets in the intersection are contained in each other
    - In other words, the order of the intersection set under inclusion is linear
    - {{x, y}, {x } } is a chain while {{x}, {y}} is not

High Level Data Races – an Example

- Example
  - V(t1) ={ {x, y, z} }, V(t2) = { {x }, {y, z} }
  - Thus {x, y, z} is a maximal view of t1
  - The intersection of {x, y, z} and V(t2) is V(t2) an is not a chain

- Intuition
  - If one thread accesses a set of shared variables under a lock and the other is not
    - One thread is treating that access as a transaction while the other is not, why?

High Level Data Races with Dependencies

- Aim is to reduce false alarms
- Definition
  - We say that two views of thread t, V and W are dependent if
    - There is some execution in which they execute one after the other and
    - Their intersection is not empty
    - This defines a directed graph D
    - We say that thread r has a high level data race (with dependencies) with thread t if the intersection of a maximal view Vmax of r with a maximal length path in D does not form a chain
- Other definitions of are possible. For example,
  - Two views V and W are dependent if
    - There is some execution in which they execute one after the other and
    - There is a data dependency between them (possibly through and auxiliary variable)
  - Future work - same definitions could be developed regardless of the particular dependency
- For practical usage we focus on maximal length paths in D that does not include repeating views

High Level Data Races with Dependencies – an Example

- Thread t1 is executing –

If (x > 0) lock() ;x--; y++; unlock() else
 lock(); x++; y++ unlock()
If (y > 0) lock() y--; z++ unlock()

- Thread t2 is executing -

lock(); x++; y++; z++ unlock();

- V = {x, y, z} is a maximal view of thread t2, and ({x, y}, {y, z}) is a maximal path of thread t1, intersecting the two results in {{x, y}, {y, z}} and does not form a chain
  - We consider the maximal path ({x, y}, {y, z}) as the views {x, y} and {y, z} are dependent
    - Their intersection is not empty

The Closure of a Concurrent Program P

- Aim – identify dormant high level data races that may be introduced due to
  - Maintenance of the program in general
  - Making an API thread safe
- Definition
  - For a given thread t, and a maximal path in D, (V1,…, Vk) that does not have repeating views, create a new thread r associated with the signal view obtained from the union of V1,…, Vk
  - The closure of P, clos(P), is obtained by performing the previous operation on every such maximal path
- Each added thread has one view so the intersection of newly added threads does not form high level data races
- Each high level data race in P is also a high level data race in clos(P). If P = clos(P), we say that the program is closed
  - Clos(clos(P)) = clos(P) as thread in clos(P)-P have single view and intersecting them will always create a chain

The Closure of a Concurrent Program P – an Example

- Program P with one thread, t1,

```
lock() x = y; unlock();
If(cond_1) lock(); z = x; unlock() else
 lock() w = y; unlock()
lock() r = t; unlock();
```

- D includes two maximal paths to consider ({x, y}, {z, x}) and ({x, y}, {w, y}) resulting in adding two threads –
  - T2 with single view {x, y, z}
  - T3 with single view {x, y, w}
- Clos(P) has two high two data races
  - They highlight potential future high level data races in P

Using the Closure Operation to Make an API Thread Safe

- An API (Application Programming Interface) is a set of interfaces exposed to the user
- Each interface may spawn additional threads while it executes
- Denote each API interface by Pi. Denote the concurrent program obtained from executing P1,…Pk in parallel by P
- The task of making an API thread safe consists of
  - Removing (low level) data races from the concurrent program P by adding appropriate locks
  - Identifying and removing any deadlock that resulted from the first step
  - Analyzing high level data races in P
- Instead of analyzing high level data races in P we analyze high level data races in clos(Pi) for each exposed application interface Pi

Using the Closure Operation to Make an API Thread Safe (Continued)

- Only analyzing the high level data races of clos(Pi) is justified as followed
  - A high level data race in P is a result of a maximal view in some thread, Vmax, intersecting with views V and W of another thread, resulting in T and S not contained in each other
  - That means that V and W are not contained in each other
  - Which means that either V and W are along a maximal path in some D and were flagged by the closure operation or
    - They have no control and/or data intersection dependency between them
- Thus, its enough to inspect the high level data races in clos(Pi) for each Pi in P

Conclusion and Future Work

- The closure operation clos(P) of a concurrent program P was introduced
  - Aim - identify dormant high level data races that may be introduced due to
    - Maintenance of the program in general
    - Making an API thread safe
- The closure operation was used for making an API thread safe
  - Further research is needed to determine the applicability of the closure operation to the general setting of concurrent program maintenance
- High level data races with dependencies can be defined in a variety of ways
  - Future research may set a general framework for the definition of high level data races with dependencies in which the tradeoff between the various possible defintions can be highlighted