# Как найти дедлок, не попав в него

Никита Коваль

nkoval@devexperts.com

DEVEXPERTS

# Жизненный пример

Slide 2/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

```java
interface Con { // Connection
  void onRegister(ConnectionManager cm);
  void close();
}

class ConnectionManager {
  Set<Con> cons = new HashSet<>();

  synchronized void register(Con c) {
    cons.add(c);
    con.onRegister(this);
  }

  synchronized void unregister(Con c) {
    cons.remove(c);
  }

  ...
```

```java
  // ... ConnectionManager

  synchronized int getConCount() {
    return cons.size();
  }

  synchronized void closeAll() {
    for (Con c: cons)
      c.close();
  }

}
```

Slide 3/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

```java
class ConnectionImpl implements Con {
  ConnectionManager cm;

  @Override
  public synchronized void onRegister(ConnectionManager cm) {
    this.cm = cm;
  }

  @Override
  public synchronized void close() {
    if (cm != null) {
      cm.unregister(this);
      cm = null;
    }
  }
}
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

Запускаем функциональный тест…

Все работает!

Запускаем стресс тест…

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

```java
public void test() throws Exception {
  ConnectionManager cm = new ConnectionManager();
  new Thread(() -> {
    while (true) {
      Connection con = new ConnectionImpl();
      cm.register(con);
      con.close(); // lock con -> lock cm
    }
  }).start();
  for (int i = 1; i < 100_000; i++) {
    System.out.println("Iteration #" + i);
    cm.closeAll(); // lock cm -> lock con
  }
}
```

Slide 6/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

```
Java stack information for the threads listed above:
===================================================
"Thread-0":
  at a.ConnectionManager.unregister(Sample.java:20)
  - waiting to lock <0x0000000795b8ee60> (a a.ConnectionManager)
  at a.ConnectionImpl.close(Sample.java:45)
  - locked <0x0000000795d72fd8> (a a.ConnectionImpl)
  at a.StressTest.lambda$test$0(StressTest.java:16)
  at a.StressTest$$Lambda$1/1297685781.run(Unknown Source)
  at java.lang.Thread.run(Thread.java:745)
"main":
  at a.ConnectionImpl.close(Sample.java:44)
  - waiting to lock <0x0000000795d72fd8> (a a.ConnectionImpl)
  at a.ConnectionManager.closeAll(Sample.java:29)
  - locked <0x0000000795b8ee60> (a a.ConnectionManager)
  at a.StressTest.test(StressTest.java:21) <27 internal calls>
Found 1 deadlock.
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

ConnectionManager cm
Connection con

| | |
|---|---|
| 1 MONITORENTER con | |
| 2 con.close() | |
| | 3 MONITORENTER cm |
| | 4 cm.closeAll() |
| 5 MONITORENTER cm | |
| 6 cm.unregister(con) | |
| | 7 MONITORENTER con |
| | 8 con.close() |

Slide 8/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

ConnectionManager cm
Connection con

| | |
|---|---|
| 1 **MONITORENTER con** | |
| 2 con.close() | |
| | 3 MONITORENTER cm |
| | 4 cm.closeAll() |
| 5 MONITORENTER cm | |
| 6 cm.unregister(con) | |
| | 7 MONITORENTER con |
| | 8 con.close() |

Slide 9/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

ConnectionManager cm
Connection con

| | |
|---|---|
| 1 MONITORENTER con | |
| 2 con.close() | |
| | 3 MONITORENTER cm |
| | 4 cm.closeAll() |
| 5 MONITORENTER cm | |
| 6 cm.unregister(con) | |
| | 7 MONITORENTER con |
| | 8 con.close() |

Slide 10/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

ConnectionManager cm
Connection con

| | |
|---|---|
| 1 MONITORENTER con | |
| 2 con.close() | |
| | 3 **MONITORENTER cm** |
| | 4 cm.closeAll() |
| 5 MONITORENTER cm | |
| 6 cm.unregister(con) | |
| | 7 MONITORENTER con |
| | 8 con.close() |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

ConnectionManager cm
Connection con

| | |
|---|---|
| 1 MONITORENTER con | |
| 2 con.close() | |
| | 3 MONITORENTER cm |
| | 4 **cm.closeAll()** |
| 5 MONITORENTER cm | |
| 6 cm.unregister(con) | |
| | 7 MONITORENTER con |
| | 8 con.close() |

Slide 12/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

ConnectionManager cm
Connection con

| | |
|---|---|
| 1 MONITORENTER con | |
| 2 con.close() | |
| | 3 MONITORENTER cm |
| | 4 cm.closeAll() |
| 5 **MONITORENTER cm** | |
| 6 cm.unregister(con) | |
| | 7 **MONITORENTER con** |
| | 8 con.close() |

Slide 13/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# И ПРАВДА ДЕДЛОК!

```
1   MONITOR
2   con.clos
5   MONITORENTER cm
6   cm.unregister(con)
```

```
    ITORENTER
    cm.c seAll
7   MONITORENTER
8   con.close()
```

Slide 14/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

26
03

# Ещё пример дедлока

```
void transfer(Account from,
              Account to,
              int amount) {
  from.lock()
  to.lock()
  // Do transfer
  unlock(from, to)
}
```

| transfer(a, b, …): | transfer(b, a, …): |
|---|---|
| 1 a.lock() | |
| | 2 b.lock() |
| | 3 a.lock() |
| 4 b.lock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Ещё пример дедлока

```
void transfer(Accoun
              Acco
              int

  from.lock()
  to.lock()
  // Do transfer
  unlock(from, to)
}
```

**И ТУТ ДЕДЛОК!**

| transfer(a, b, …): | transfer(b, a, …): |
|---|---|
| 1 a.lock() | |
| | 2 b.lock() |
| | 3 a.lock() |
| 4 b.lock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Подходы к анализу

- Статический анализ
- Model Checking
- Динамический анализ

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Статический анализ

- Анализирует программу не выполняя ее
- Может гарантировать отсутствие взаимных блокировок
- Обычно много false positives

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Model checking

- Анализирует **модель** программы
- Может гарантировать отсутствие взаимных блокировок
- Нет false positives
- Сложно построить модель
- Требует много вычислительных ресурсов

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Динамический анализ

- Зависит от конкретного исполнения
- Мало false positives
- Применим к большим приложениям

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Динамический анализ: техники

- Анализ трасс исполнения (off-line)
  - `JCarder`, `MulticoreSDK`
- **Обнаружение в момент возникновения (on-line)**
  - Доступна вся контекстная информация
    - Stacktrace, внутреннее состояние
  - `VisualThreads`

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# АЛГОРИТМ

DEVEXPERTS

# Иерархия блокировок

Любые две блокировки берутся всегда в одном порядке внутри одного потока

```
void transfer(Account from, Account to, int amount) {
  // Order lock acquisitions by ids
  if (from.id < to.id)
     lock(from, to)
   else
     lock(to, from)
   // Do transfer
  unlock(from, to)
}
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Потенциальная взаимная блокировка

- Иерархия блокировок – единственный на практике способ борьбы с дедлоками
- *Потенциальный дедлок* – нарушение этой иерархии

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Граф блокировок

- Ориентированный граф
- Вершина $\leftrightarrow$ блокировка
- Ребро $(\boldsymbol{a}, \boldsymbol{b})$ означает, что $\boldsymbol{a} \rightarrow \boldsymbol{b}$
- <u>Цикл = потенциальный дедлок</u>

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Граф блокировок: пример

| Thread 1 | Thread 2 |
|---|---|
| 1 a.lock() | |
| 2 b.lock() | |
| 3 unlock(a, b) | |
| | 4 b.lock() |
| | 5 a.lock() |
| | 6 unlock(a, b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Граф блокировок: пример

| Thread 1 | Thread 2 |
|---|---|
| 1 a.lock() | |
| 2 b.lock() | |
| 3 unlock(a, b) | |
| | 4 b.lock() |
| | 5 a.lock() |
| | 6 unlock(a, b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Граф блокировок: пример

| Thread 1 | Thread 2 |
|---|---|
| 1 a.lock() | |
| 2 **b.lock()** | |
| 3 unlock(a, b) | |
| | 4 b.lock() |
| | 5 a.lock() |
| | 6 unlock(a, b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Граф блокировок: пример

| Thread 1 | Thread 2 |
|---|---|
| 1 a.lock() | |
| 2 b.lock() | |
| 3 **unlock(a, b)** | |
| | 4 b.lock() |
| | 5 a.lock() |
| | 6 unlock(a, b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Граф блокировок: пример

| Thread 1 | Thread 2 |
|---|---|
| 1 a.lock() | |
| 2 b.lock() | |
| 3 unlock(a, b) | |
| | 4 **b.lock()** |
| | 5 a.lock() |
| | 6 unlock(a, b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Граф блокировок: пример

| Thread 1 | Thread 2 |
|---|---|
| 1 a.lock() | |
| 2 b.lock() | |
| 3 unlock(a, b) | |
| | 4 b.lock() |
| | 5 **a.lock()** |
| | 6 unlock(a, b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Принцип минимизации: проблема

- Два цикла показывают на одну ошибку

| Thread 1 | Thread 2 |
|---|---|
| 1 a.lock() | |
| 2 c.lock() | |
| 3 b.lock() | |
| 4 unlock(a,b,c) | |
| | 5 b.lock() |
| | 6 a.lock() |
| | 7 unlock(a,b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Принцип минимизации: проблема (1)

- Два цикла показывают на одну ошибку

| Thread 1 | Thread 2 |
|---|---|
| 1 a.lock() | |
| 2 c.lock() | |
| 3 b.lock() | |
| 4 unlock(a,b,c) | |
| | 5 b.lock() |
| | 6 a.lock() |
| | 7 unlock(a,b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Принцип минимизации: проблема (2)

- Два цикла показывают на одну ошибку

| Thread 1 | Thread 2 |
|---|---|
| 1 a.lock() | |
| 2 **c.lock()** | |
| 3 b.lock() | |
| 4 unlock(a,b,c) | |
| | 5 b.lock() |
| | 6 a.lock() |
| | 7 unlock(a,b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Принцип минимизации: проблема (3)

- Два цикла показывают на одну ошибку

| | Thread 1 | Thread 2 |
|---|---|---|
| 1 | a.lock() | |
| 2 | c.lock() | |
| 3 | **b.lock()** | |
| 4 | unlock(a,b,c) | |
| 5 | | b.lock() |
| 6 | | a.lock() |
| 7 | | unlock(a,b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Принцип минимизации: проблема (4)

- Два цикла показывают на одну ошибку

| | Thread 1 | | Thread 2 |
|---|---|---|---|
| 1 | a.lock() | | |
| 2 | c.lock() | | |
| 3 | b.lock() | | |
| 4 | **unlock(a,b,c)** | | |
| | | 5 | b.lock() |
| | | 6 | a.lock() |
| | | 7 | unlock(a,b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Принцип минимизации: проблема (5)

- Два цикла показывают на одну ошибку

| Thread 1 | Thread 2 |
|---|---|
| 1 a.lock() | |
| 2 c.lock() | |
| 3 b.lock() | |
| 4 unlock(a,b,c) | |
| | 5 **b.lock()** |
| | 6 a.lock() |
| | 7 unlock(a,b) |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Принцип минимизации: проблема (6)

- Два цикла показывают на одну ошибку

| Thread 1 | Thread 2 |
|---|---|
| 1 a.lock() | |
| 2 c.lock() | |
| 3 b.lock() | |
| 4 unlock(a,b,c) | |
| | 5 b.lock() |
| | 6 **a.lock()** |
| | 7 unlock(a,b) |

Slide 39/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Принцип минимизации

- Более короткий цикл полезнее
- Сообщаем о самом коротком

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Скелет алгоритма

- Анализ операций взятия и отпускания блокировки
- Анализ взятия блокировки:
  - Добавить блокировку в  множество взятых текущим потоком блокировок
  - Добавить рёбра в граф блокировок
  - Сообщить о новых циклах
- Анализ отпускания блокировки:
  - Удалить блокировку из множества

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Анализ взятия и отпускания блокировки

| Thread 1 | Thread 2 |
|---|---|
| | 1 b.lock() |
| | 2 a.lock() |
| | 3 a.unlock() |
| 4 a.lock() | |
| 5 c.lock() | |
| | 6 b.unlock() |
| 7 b.lock() | |
| 8 a.unlock() | |
| 9 c.unlock() | |
| 10 b.unlock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Анализ взятия и отпускания блокировки (1)

| Thread 1 | Thread 2 |
|---|---|
| | 1 **b.lock()** |
| | 2 a.lock() |
| | 3 a.unlock() |
| 4 a.lock() | |
| 5 c.lock() | |
| | 6 b.unlock() |
| 7 b.lock() | |
| 8 a.unlock() | |
| 9 c.unlock() | |
| 10 b.unlock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Анализ взятия и отпускания блокировки (2)

| | Thread 1 | Thread 2 |
|---|---|---|
| 1 | | b.lock() |
| 2 | | **a.lock()** |
| 3 | | a.unlock() |
| 4 | a.lock() | |
| 5 | c.lock() | |
| 6 | | b.unlock() |
| 7 | b.lock() | |
| 8 | a.unlock() | |
| 9 | c.unlock() | |
| 10 | b.unlock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Анализ взятия и отпускания блокировки (3)

| Thread 1 | Thread 2 |
|---|---|
| | 1 b.lock() |
| | 2 a.lock() |
| | 3 **a.unlock()** |
| 4 a.lock() | |
| 5 c.lock() | |
| | 6 b.unlock() |
| 7 b.lock() | |
| 8 a.unlock() | |
| 9 c.unlock() | |
| 10 b.unlock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Анализ взятия и отпускания блокировки (4)

| | Thread 1 | Thread 2 |
|---|---|---|
| 1 | | b.lock() |
| 2 | | a.lock() |
| 3 | | a.unlock() |
| 4 | a.lock() | |
| 5 | c.lock() | |
| 6 | | b.unlock() |
| 7 | b.lock() | |
| 8 | a.unlock() | |
| 9 | c.unlock() | |
| 10 | b.unlock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Анализ взятия и отпускания блокировки (5)

| | Thread 1 | Thread 2 |
|---|---|---|
| 1 | | b.lock() |
| 2 | | a.lock() |
| 3 | | a.unlock() |
| 4 | a.lock() | |
| 5 | **c.lock()** | |
| 6 | | b.unlock() |
| 7 | b.lock() | |
| 8 | a.unlock() | |
| 9 | c.unlock() | |
| 10 | b.unlock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Анализ взятия и отпускания блокировки (6)

| Thread 1 | Thread 2 |
|---|---|
| | 1 b.lock() |
| | 2 a.lock() |
| | 3 a.unlock() |
| 4 a.lock() | |
| 5 c.lock() | |
| | 6 **b.unlock()** |
| 7 b.lock() | |
| 8 a.unlock() | |
| 9 c.unlock() | |
| 10 b.unlock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Анализ взятия и отпускания блокировки (7)

| Thread 1 | Thread 2 |
|---|---|
| | 1 b.lock() |
| | 2 a.lock() |
| | 3 a.unlock() |
| 4 a.lock() | |
| 5 c.lock() | |
| | 6 b.unlock() |
| 7 **b.lock()** | |
| 8 a.unlock() | |
| 9 c.unlock() | |
| 10 b.unlock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Анализ взятия и отпускания блокировки (8)

| Thread 1 | Thread 2 |
|---|---|
| | 1 b.lock() |
| | 2 a.lock() |
| | 3 a.unlock() |
| 4 a.lock() | |
| 5 c.lock() | |
| | 6 b.unlock() |
| 7 b.lock() | |
| 8 **a.unlock()** | |
| 9 c.unlock() | |
| 10 b.unlock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Анализ взятия и отпускания блокировки (9)

| Thread 1 | Thread 2 |
|---|---|
| | 1 b.lock() |
| | 2 a.lock() |
| | 3 a.unlock() |
| 4 a.lock() | |
| 5 c.lock() | |
| | 6 b.unlock() |
| 7 b.lock() | |
| 8 a.unlock() | |
| 9 **c.unlock()** | |
| 10 b.unlock() | |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Анализ взятия и отпускания блокировки (10)

| Thread 1 | Thread 2 |
|---|---|
| | 1 b.lock() |
| | 2 a.lock() |
| | 3 a.unlock() |
| 4 a.lock() | |
| 5 c.lock() | |
| | 6 b.unlock() |
| 7 b.lock() | |
| 8 a.unlock() | |
| 9 c.unlock() | |
| 10 **b.unlock()** | |

Slide 52/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Множество взятых блокировок

- Обычно блокировка используется так:

```
synchronized(o) {
  // Do something
}

synchronized void f() {
  // Do something
}
```

```
l.lock()
try {
  // Do something
} finally {
  l.unlock()
}
```

- Блокировки берутся и отпускаются в LIFO (стек) порядке

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Множество взятых блокировок

- Стек с возможностью удаления из середины
- Почти всегда работает за $O(1)$
- В худшем случае за $O(B)$
  - $B$ – количество блокировок, которые держит поток

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Топологический порядок

- Задается ребрами ориентированного графа
- Нет топологического порядка ⇔ есть цикл

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Топологический порядок

- Задается ребрами ориентированного графа
- Нет топологического порядка ⇔ есть цикл

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Топологический порядок

- Инкрементальная поддержка топологического порядка
  - MNR алгоритм, предложенный Marchetti-Spaccamela и др.
- Поддерживается только ацикличная часть графа блокировок
- Ребро $(\boldsymbol{a}, \boldsymbol{b})$ приводит к нарушению топ. порядка $\Rightarrow$ кратчайший путь $\boldsymbol{b} \rightsquigarrow \boldsymbol{a}$ обозначает искомый цикл

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Сложность алгоритма

- Анализ операции взятия блокировки
  - Если не образуется новых циклов
    - Обычно это так
    - $O(B + |V| + |E|)$
  - Если новые циклы найдены
    - $O\big(B \cdot (|V| + |E|)\big)$
- Анализ операции отпускания блокировки
  - Удаление блокировки из множества блокировок
    - Обычно за $O(1)$
    - В худшем случае за $O(B)$

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Ограничения: минимизация цикла

```
void transfer(Account from,
              Account to,
              int amount) {
  from.lock()
  to.lock()
  // Do transfer
  unlock(from, to)
}
```

```
1  transfer(a, c, …)
2  transfer(c, b, …)
3  transfer(b, a, …)
4  transfer(a, b, …)
```

Slide 59/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Ограничения: минимизация цикла (1)

```
void transfer(Account from,
              Account to,
              int amount) {
  from.lock()
  to.lock()
  // Do transfer
  unlock(from, to)
}
```

```
1  transfer(a, c, …)
2  transfer(c, b, …)
3  transfer(b, a, …)
4  transfer(a, b, …)
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Ограничения: минимизация цикла (2)

```
void transfer(Account from,
              Account to,
              int amount) {
  from.lock()
  to.lock()
  // Do transfer
  unlock(from, to)
}
```

```
1  transfer(a, c, …)
2  transfer(c, b, …)
3  transfer(b, a, …)
4  transfer(a, b, …)
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Ограничения: минимизация цикла (3)

```
void transfer(Account from,
              Account to,
              int amount) {
  from.lock()
  to.lock()
  // Do transfer
  unlock(from, to)
}
```

```
1  transfer(a, c, …)
2  transfer(c, b, …)
3  transfer(b, a, …)
4  transfer(a, b, …)
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Ограничения: минимизация цикла (4)

```
void transfer(Account from,
              Account to,
              int amount) {
  from.lock()
  to.lock()
  // Do transfer
  unlock(from, to)
}
```

```
1  transfer(a, c, …)
2  transfer(c, b, …)
3  transfer(b, a, …)
4  transfer(a, b, …)
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Ограничения: минимизация цикла

- Добавление ребра $(a, b)$ образовало цикл
- Минимизация $\langle b, \ldots, a \rangle \Leftrightarrow$ минимизация $b \rightsquigarrow a$
  - Можно реализовать с помощью BFS
- Обычно длина цикла 2 или 3
- Текущая реализация не делает минимизацию

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Ограничения: независимые циклы

- Независимые логически циклы:
  - $\langle a, c, b \rangle$
  - $\langle a, b \rangle$

- Независимый цикл не будет обнаружен вследствие принципа минимизации
- Обычно один полезный цикл лучше ☺

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Ограничения: single threaded цикл

```
Lock a, b
---------------
1   a.lock()
2   b.lock()
3   unlock(a, b)

4   b.lock()
5   a.lock()
6   unlock(a, b)
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Ограничения: guarded цикл

| Thread 1 | Thread 2 |
|---|---|
| 1 `g.lock()` | |
| 2 `a.lock()` | |
| 3 `b.lock()` | |
| 4 `unlock(a, b)` | |
| 5 `g.unlock()` | |
| | 6 `g.lock()` |
| | 7 `b.lock()` |
| | 8 `a.lock()` |
| | 9 `unlock(a, b)` |
| | 10 `g.unlock()` |

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# РЕАЛИЗАЦИЯ

DEVEXPERTS

# Dl-Check

- https://github.com/Devexperts/dlcheck
- Удобен для тестирования
  - Кидает исключение при `-Ddlcheck.fail=true`
  - Можно настроить своё поведение
- Реализован в виде Java агента
  - `java -javaagent:dlcheck.jar -jar your_app.jar`
  - Модифицирует байт-код с помощью ASM

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Трансформация байт-кода

Анализ операций взятия и отпускания блокировки:

- Первое и последнее действие `synchronized` метода
- `MONITORENTER` и `MONITOREXIT` инструкции
- `java.util.concurrent.locks.Lock`
  - Метод `lock`
  - Успешное выполнение метода `tryLock`
  - Метод `unlock`

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Synchronized метод: на пальцах

- Анализ взятия блокировки
- Анализ отпускания блокировки

```
synchronized void f() {
  // Do something
}
```

```
synchronized void f() {
  onLockAcquire(this,
                2 /*location id*/);
  try {
    // Do something
  } finally {
    onLockRelease(this);
  }
}
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Synchronized мето

**А если… исключение?**

- Анализ взятия блокировки
- Анализ отпуск блокировки

```
synchronize{
  // Do som
}
```

```
g() {
onLockAcquire(this,
                2 /*location id*/);
try {
  // Do something
} finally {
  onLockRelease(this);
}
}
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Synchronized метод: исключение

НЕПРАВИЛЬНО

```
synchronized void f() {
  onLockAcquire(this, 2);
  try {
    // Do something
  } finally {
    onLockRelease(this);
  }
}
```

ПРАВИЛЬНО

```
synchronized void f() {
  try {
    onLockAcquire(this, 2);
    // Do something
  } finally {
    onLockRelease(this);
  }
}
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Synchronized метод: байт-код

- Анализ взятия блокировки
- Анализ отпускания блокировки

```
synchronized void g() {
  try {
    onLockAcquire(this, 2);
    // Do something
  } finally {
    onLockRelease(this);
  }
}
```

```
synchronized void g();
Code:
   0: aload_0 // this
   1: iconst_2 // location id
   2: invokestatic #22 // onLockAcquire
      // Do something
   5: aload_0 // this
   6: invokestatic #26 // onLockRelease
   9: return
  10: aload_0 // this
  11: invokestatic #26 // onLockRelease
  14: athrow
Exception table:
  from   to   target   type
     0   9      10     any
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Synchronized блок

```
void f() {
  synchronized (o) {
    // Do something
  }
}
```

```
void f() {
  MONITORENTER o
  try {
    // Do something
  } finally {
    MONITOREXIT o
  }
}
```

- Анализ взятия блокировки
- Анализ отпускания блокировки

```
void f() {
  MONITORENTER o
  try {
    onLockAcquire(o, 42);
    // Do something
  } finally {
    MONITOREXIT o
    onLockRelease(o);
  }
}
```

Slide 75/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Synchronized блок

```
void f() {
   synchronized (o) {
      // Do something
   }
}

void f() {
   MONITORENTER
   {
      o
   }
   MONITO
   }
}
```

Не все так просто!

```
void f() {
   MONITORENTER o
   try {
      onLockAcquire(o, 42);
      // Do something
   } finally {
      MONITOREXIT o
      onLockRelease(o);
   }
}
```

Slide 76/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Synchronized блок: за кадром

- Анализ взятия блокировки
- Анализ отпускания блокировки

```
void f() {
  synchronized (o) {
    // Do something
  }
}
```

```
Exception table:
  from   to  target type
     7    9      12  any
    12   15      12  any
```

```
void f();
Code:
   0: aload_0 // this
   1: getfield  #2 // monitor
   4: dup
   5: astore_1 // store monitor
   6: monitorenter
      // Do something
   7: aload_1 // monitor
   8: monitorexit
   9: goto  17 // skip finally
  12: astore_2 // exception
  13: aload_1 // monitor
  14: monitorexit
  15: aload_2 // exception
  16: athrow
  17: return
```

try

catch

Slide 77/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Synchronized блок: исключения

НЕПРАВИЛЬНО

```
void f() {
  MONITORENTER o
  try {
    onLockAcquire(o, 42);
    // Do something
  } finally {
    MONITOREXIT o
    onLockRelease(o);
  }
}
```

ПРАВИЛЬНО

```
void f() {
  MONITORENTER o
  try {
    onLockAcquire(o, 42);
    // Do something
  } finally {
    MONITOREXIT o
    try {
      onLockRelease(o);
    catch (Throwable t) {}
  }
}
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Synchronized блок: а вдруг не блок

```
void f() {
  MONITORENTER o
  try {
     // Do something
  } finally {
    MONITOREXIT o
  }
}
```

```
void f() {
  MONITORENTER o
  try {
    onLockAcquire(o, 42);
  } catch (Throwable t) {
    MONITOREXIT o
    onLockRelease(o);
    throw t;
  }
  try {
     // Do something
    ...
```

Наш try-catch

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# j.u.c.l.Lock: как используется

```
void f() {
  l.lock();
  try {
    // Do something
  } finally {
    l.unlock();
  }
}
```

```
void g() {
  l.lock();
  // Do something
  l.unlock();
}
```

```
void h() {
  b = l.tryLock();
  // Do something
  l.unlock();
}
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# j.u.c.l.Lock: как используе

```
void f() {
   l.lock();
   try {
      // Do s
   } finally
      l.unloc
   }
}
```

```
void g() {
   l.loc  ();
   // Do something
   l.unlock();
}
```

```
   l.tryLock();
   // Do something
   l.unlock();
}
```

Это же прям как непарные **MONITORENTER** и **MONITOREXIT**!

Slide 82/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# j.u.c.l.Lock: как выглядит байт-код

```
void g() {
   l.lock();
   // Do something
   l.unlock();
}
```

```
void g();
Code:
   0: aload_0 // this
   1: getfield  #3 // Lock l
   4: invokeinterface #4, 1 // lock()
      // Do something
   9: aload_0 // this
  10: getfield  #3 // Lock l
  13: invokeinterface #5, 1 // unlock()
  18: return
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# j.u.c.l.Lock: tryLock

```
void h() {
  b = l.tryLock();
  // Do something
  l.unlock();
}
```

```
void h() {
  b = l.tryLock();
  if (b) {
    try {
      onLockAcquire(l, 2);
    } catch (Throwable t) {
      l.unlock();
      onLockRelease(l);
      throw t;
    }
  }
  // Do something
  l.unlock();
  onLockRelease(l);
}
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Использование ClassTransformer

Типичный код:

```
public final byte[] transform(..., classFileBuffer) {
  ClassReader cr = new ClassReader(classFileBuffer);
  ClassWriter cw = new ClassWriter(...);
  ClassVisitor classTransformer = new MyTransformer(cw, ...);
  cr.accept(classTransformer, EXPAND_FRAMES);
  return cw.toByteArray();
}
```

Попробуем запустить…

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Использование ClassTransformer

```
java.lang.LinkageError: loader (instance of
sun/misc/Launcher$AppClassLoader): attempted duplicate class
definition for name: "com/intellij/util/lang/UrlClassLoader"
  at java.lang.ClassLoader.defineClass1(Native Method)
  at java.lang.ClassLoader.defineClass(ClassLoader.java:763)
  at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
  at java.net.URLClassLoader.defineClass(URLClassLoader.java:467)
  at java.net.URLClassLoader.access$100(URLClassLoader.java:73)
  at java.net.URLClassLoader$1.run(URLClassLoader.java:368)
  at java.net.URLClassLoader$1.run(URLClassLoader.java:362)
  at java.security.AccessController.doPrivileged(Native Method)
  at java.net.URLClassLoader.findClass(URLClassLoader.java:361)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
  at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:331)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
  ...
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Использование ClassTransformer

```
java.lang.LinkageError: Bad (instance of
sun/misc/Launcher$                                  duplicate class
definition                                          ang/UrlClassLoader"
  at java.lang.                                     e Method)
  at java.lang.(                                    ssLoader.java:763)
  at java.security.                                 neClass(SecureClassLoader.java:142)
  at java.net.URLClassLoader                        eClass(URLClassLoader.java:467)
  at java.net.URLClassLoader.access$100(            RLClassL                    73)
  at java.net.URLClassLoader$1.run(URLClassLoader.
  at java.net.URLClassLoader$1.run(URLClassLoader.
  at java.security.AccessController.doPrivileg
  at java.net.URLClassLoader.findClass(URLClass                av
  at java.lang.ClassLoader.loadClass(ClassLoader                424
  at sun.misc.Launcher$AppClassLoader.loadCla                  Launcher
  at java.lang.ClassLoader.loadClass(ClassLoader.java:3
  ...
```

Ну, ASM...

Ну погоди!

Slide 87/106. Copyright © 2017. Devexperts LLC. All rights reserved.

# Проблемный ClassWriter

```
protected String getCommonSuperClass(String type1,
                                     String type2) {
  Class<?> c1, c2;
  ClassLoader loader = getClass().getClassLoader();
  try {
    c1 = Class.forName(type1.replace('/', '.'), false, loader);
    c2 = Class.forName(type2.replace('/', '.'), false, loader);
  } catch (Exception e) {
    throw new RuntimeException(e);
  }
  if (c1.isAssignableFrom(c2))
    return type1;
  ...
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Напишем свой!

```
@Override
protected String getCommonSuperClass(String type1,
                                     String type2) {
  ClassInfo c = ciCache.getOrBuildClassInfo(type1, loader);
  ClassInfo d = ciCache.getOrBuildClassInfo(type2, loader);
  if (c.isAssignableFrom(d, ciCache, loader))
    return type1;
  if (d.isAssignableFrom(c, ciCache, loader))
    return type2;
  ...
```

`ClassInfo` доступен **ДО** трансформации

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Ура! Заработало!

# Фреймворк JAgent

- Проблема решена в JAgent
  - https://github.com/Devexperts/jagent
- Позволяет удобнее писать Java агенты

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Конфликт зависимостей

- Используем несколько агентов:

```
java -javaagent:agent1.jar -javaagent:agent2.jar
—jar my_app.jar
```

- Необходимо изолировать зависимости

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Способы изоляции

- Repackaging
  - `maven-shade-plugin` (или аналог)
  - Нужно знать транзитивные зависимости и их структуру
  - Иногда может не сработать (reflection)
- Отдельный ClassLoader
  - Не получится использовать для кода, вызов которого добавлен при трансформации

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# JAgent спешит на помощь

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# БЕНЧМАРКИ & ВЫВОДЫ

DEVEXPERTS

# Бенчмарки

- SpecJVM2008: Apache Derby (4 и 40 потоков)
- DaCapo: Apache Lucene
- DaCapo: Banking application
- …

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Производительность

Slide 97/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Производительность

Slide 98/106. Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Потребление памяти

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Жизненный пример

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

```java
public void testClose() {
  cm.register(con);
    // lock cm
    // lock con for con.onRegister()
  assertEquals(1, cm.getConCount());
  con.close();
    // lock con
    // lock cm for cm.unregister()
  assertEquals(0, cm.getConCount());
}
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

```
============================
!!! Potential deadlock !!!
============================
### Cycle in lock graph: ###
Lock ConnectionImpl@d70c109 was acquired at:
      a.ConnectionImpl.onRegister(Sample.java:39)
      a.ConnectionImpl.close(Sample.java:44)
Lock ConnectionManager@2be94b0f was acquired at:
      a.ConnectionManager.getConCount(Sample.java:24)
      a.ConnectionManager.register(Sample.java:15)
      a.ConnectionManager.unregister(Sample.java:20)
### Current lock stack: ###
Lock ConnectionImpl@d70c109 was acquired at:
      a.ConnectionImpl.onRegister(Sample.java:39)
      a.ConnectionImpl.close(Sample.java:44)
### Current stacktrace: ###
      a.ConnectionManager.unregister(Sample.java:20)
      a.ConnectionImpl.close(Sample.java:45)
      a.FunctionalityTest.testClose(FunctionalityTest.java:22)
      ...
```

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# И где всё это пригодилось?

- Внутренние проекты
- IntelliJ IDEA
- Kotlin
- …

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

# Итого

- Алгоритм для поиска потенциальных дедлоков on-line
- Реализован инструмент `Dl-Check`
  - https://github.com/Devexperts/dlcheck
- Ближайшие планы:
  - Контракты для описания правил взятия блокировок

 Copyright © 2017. Devexperts LLC. All rights reserved.

DEVEXPERTS

Спасибо за внимание!

НУ, ТЫ ЭТО…
ТЫ ЗАХОДИ, ЕСЛИ ЧЁ!