Темная сторона изолятов в Dart

Станислав Чернышев, к.т.н., доцент СПбГУАП и СПБГЭУ

Обо мне

- ***Более 10 лет работы в IT (ВПК, МО, проекты на заказ)***
- ***Основной вид деятельности – наука и преподавание***
- ***Ученики работают в различных компаниях (VK, Paragon Software, Astra, СПб ИАЦ, Радар ММС и т.д.)***
- ***Автор учебных книг***

Обо мне


Станислав Чернышев
ОСНОВЫ DART
2-е издание, переработанное и дополненное
Использование данного файла означает, что вы согласны с условиями лицензии, расположенной на следующей странице
2024


Станислав Чернышев
Основы Dart
Версия 3.4

Что такое изолят?

Способы создания изолята

run

```
Future<R> run<R>(
  FutureOr<R> computation(),
  {Sting? debugName}
)
```

Способы создания изолята

run

```
Future<R> run<R>(
  FutureOr<R> computation(),
  {Sting? debugName}
)
```

spawn

```
Future<Isolate> spawn<T>(
  void entryPoint(T message),
  T message,
  {bool paused = false,
  bool errorsAreFatal = true,
  SendPort? onExit,
  SendPort? onError,
  @Since("2.3") String? debugName}
)
```

Создание новой изоляционной группы

spawnUri

```
Future<Isolate> spawnUri(
 Uri uri,
 List<String> args,
 dynamic message,
 {bool paused = false,
 SendPort? onExit,
 SendPort? onError,
 bool errorsAreFatal = true,
 bool? checked,
 Map<String, String>? environment,
 bool automaticPackageResolution = false,
 @Since("2.3") String? debugName}
)
```

spawn vs spawnUri

Kandinsky 3.0

spawn vs spawnUri

Флаг компиляции	Результат	Описание
exe	Автономный исполняемый файл	**Автономный исполняемый файл**, компилируемый под целевую платформу и содержащий урезанную среду выполнения Dart.
aot-snapshot	Модуль AOT	Файл, компилируемый под целевую платформу, **без среды выполнения Dart.**
jit-snapshot	Модуль JIT	Файл, компилируемый под целевую платформу, с **промежуточным оптимизированным представлением исходного кода**, который был получен в ходе учебного запуска программы.
kernel	Kernel модуль	Файл, содержащий промежуточное представление исходного кода в виде абстрактного синтаксического дерева (Kernel AST) . **Может запускаться на любых платформах.**

exe и ***aot-snapshot*** не поддерживают dart:mirrors и dart:developer

spawn vs spawnUri

Isolate.spawnUri(Uri.parse('**new_group.aot**'), [], null)

spawn vs spawnUri

C:\code\dart\deep_isolate\bin>dart compile exe main_isolate.dart

C:\code\dart\deep_isolate\bin>dart compile aot-snapshot new_group.dart

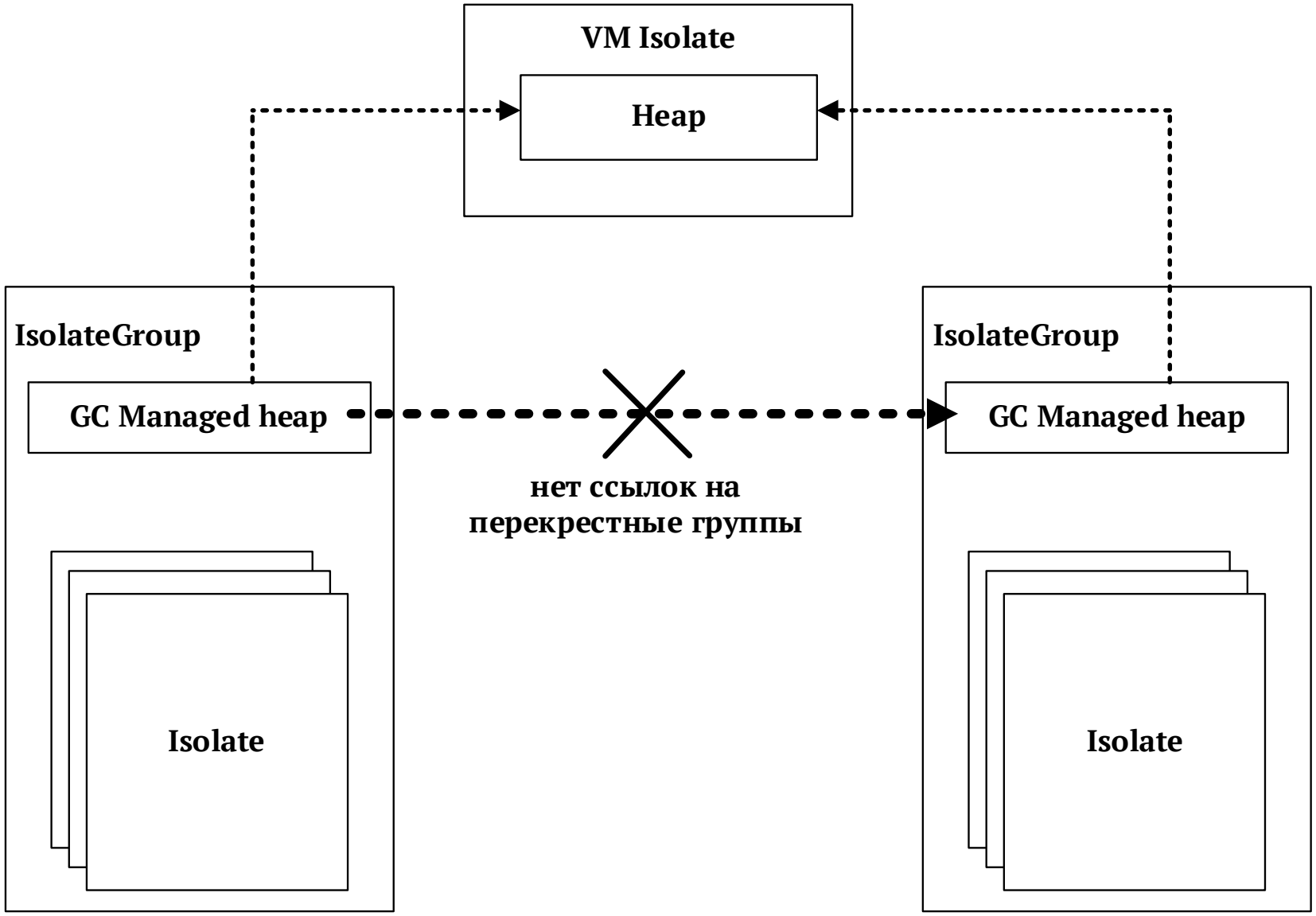
C:\code\dart\deep_isolate\bin>main_isolate.exe

Ограничения на использование SendPort

`spawn` **vs** **`spawnUri`**

Kandinsky 3.0

Условное представление Dart VM



Свойство hashCode класса Object

```dart
class Cat {
  final name;
  const Cat(this.name);
}

void main(List<String> arguments) async {
  var cat = Cat('Max');
  print('Cat hashCode: ${cat.hashCode}');

  var value = 10;
  print('Value hashCode: ${value.hashCode}');
}
// Cat hashCode: 206932202
// Value hashCode: 116010
```

Свойство hashCode класса Object

```dart
void main(List<String> arguments) async {
  var cat = Cat('Max');
  print('${cat.hashCode}'); // 344563804
  print('${identityHashCode(cat)}'); // 1013285941

  var value = 10;
  print('${value.hashCode}'); // 116010
  print('${identityHashCode(value)}'); // 116010
}
```

Как объекту присваивается hashCode?

Как объекту присваивается hashCode?

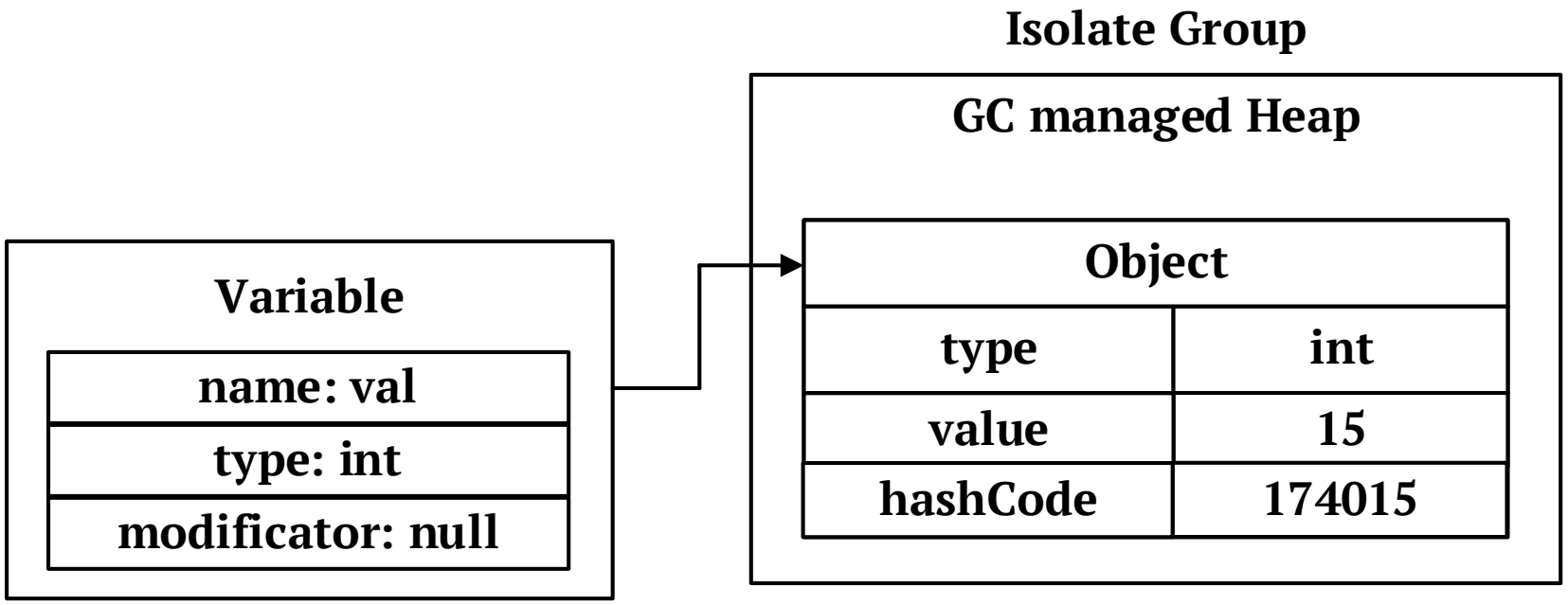
```
// sdk/runtime/vm/object_graph.cc
uint32_t HeapSnapshotWriter::GetHashHelper(Thread* thread, ObjectPtr obj) {
  uint32_t hash;
#if defined(HASH_IN_OBJECT_HEADER)
  hash = Object::GetCachedHash(obj);
  if (hash == 0) {
    ASSERT(
       !thread->heap()->old_space()->IsObjectFromImagePages(obj)
    );
    hash = GenerateHash(thread->random());
    Object::SetCachedHashIfNotSet(obj, hash);
  }
#else
  Heap* heap = thread->heap();
  hash = heap->GetHash(obj);
  if (hash == 0) {
    ASSERT(!heap->old_space()->IsObjectFromImagePages(obj));
    hash = GenerateHash(thread->random());
    heap->SetHashIfNotSet(obj, hash);
  }
#endif
  return hash;
}
```

Как объекту присваивается hashCode?

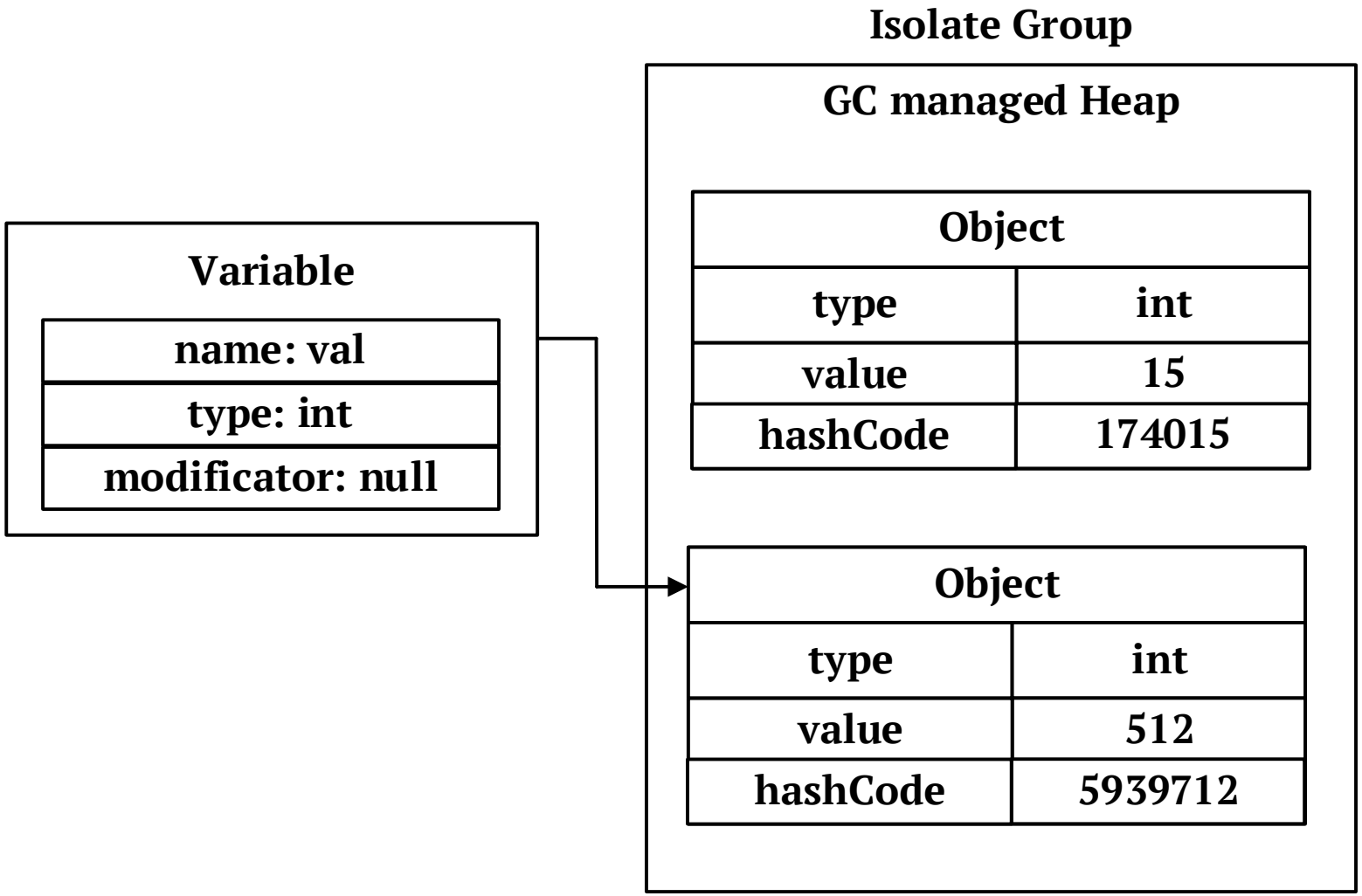
```
// sdk/runtime/vm/object_graph.cc
static uint32_t GenerateHash(Random* random) {
  uint32_t hash;
  do {
    hash = random->NextUInt32();
  } while (hash == 0 || (kSmiBits < 32 && !Smi::IsValid(hash)));
  return hash;
}
```

Объявление переменной или имени?

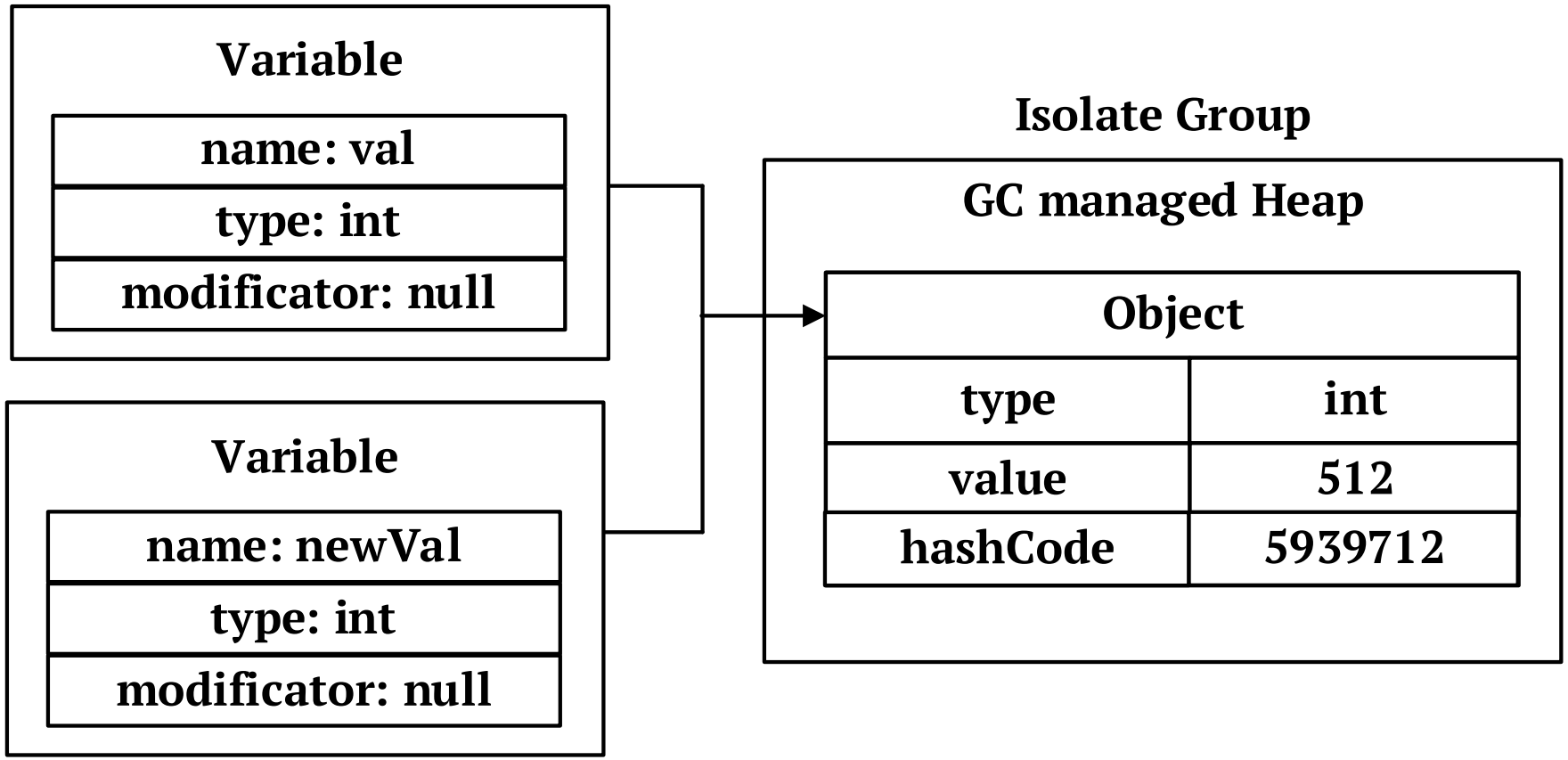
```
var val = 15;
```


Isolate Group
GC managed Heap
Variable
name: val
type: int
modificator: null
Object
type
int
value
15
hashCode
174015

```
val = 512;
```


Isolate Group
GC managed Heap
Variable
name: val
type: int
modificator: null
Object
type
int
value
15
hashCode
174015
Object
type
int
value
512
hashCode
5939712

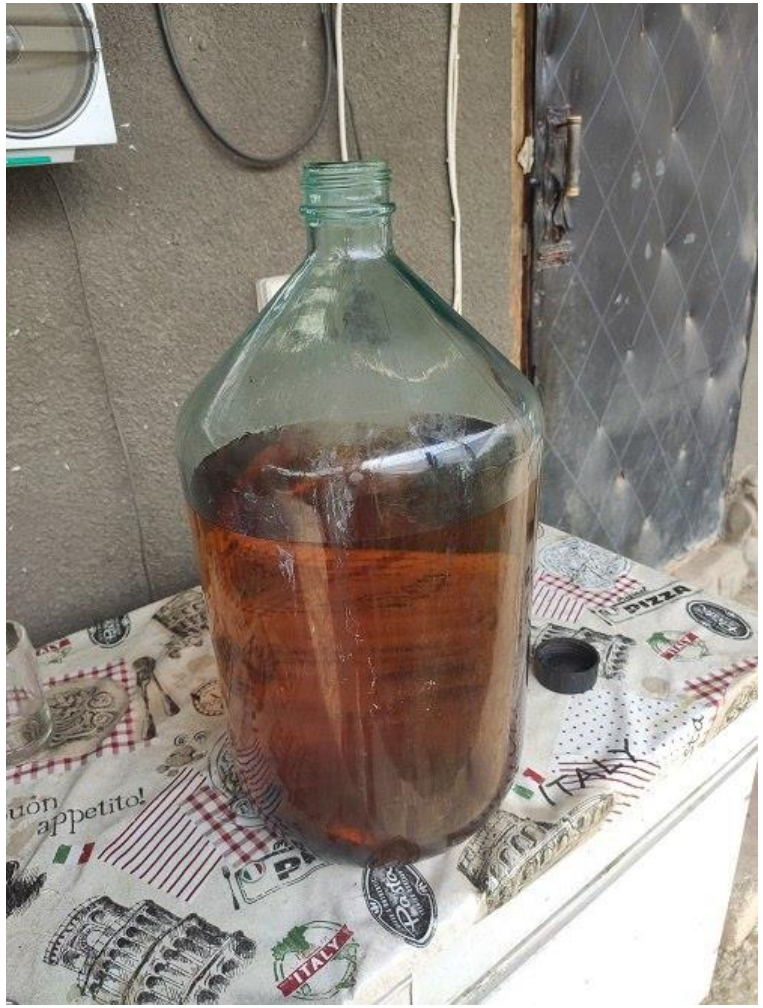
```
var newVal = val;
```



```dart
import 'dart:convert';

void main(List<String> arguments) async {
  var val = 'Hello';
  var newVal = 'Hello';
  print('${val.hashCode}'); // 828172268
  print('${newVal.hashCode}'); // 828172268
  print('${identical(val, newVal)}'); // true

  var list = [1, 2, 3];
  var newList = [1, 2, 3];
  print('${list.hashCode}'); // 664960072
  print('${newList.hashCode}'); // 464510092
  print('${identical(list, newList)}'); // false

  var value = jsonDecode('{"a": 1, "b": 1}');
  print(identical(value['a'], value['b'])); // true
  print(identityHashCode(value['a'])); // 11601
  print(identityHashCode(value['b'])); // 11601
}
```

```dart
class Cat {
  final name;
  const Cat(this.name);
}

void main(List<String> arguments) async {
  var cat = const Cat('Max');
  var newCat = const Cat('Max');
  print('${cat.hashCode}'); // 940337853
  print('${newCat.hashCode}'); // 940337853
  print('${identical(cat, newCat)}'); // true

  var list = const [1, 2, 3];
  var newList = const [1, 2, 3];
  print('${list.hashCode}'); // 286116521
  print('${newList.hashCode}'); // 286116521
  print('${identical(list, newList)}'); // true
}
```

```dart
void main(List<String> arguments) {
  var str1 = 'Mobius';
  var str2 = 'Mobius';
  print(identical(str1, str2)); // true

  var str3 = 'Mob' + 'ius';
  print(identical(str1, str3)); // false

  var value = jsonDecode('{"a": "f", "b": "f"}');
  print(identical(value['a'], value['b'])); // true

  value = jsonDecode('{"a": "wtf", "b": "wtf"}');
  print(identical(value['a'], value['b'])); // false
  print(identityHashCode(value['a']) == identityHashCode(value['b'])); // true
}
```


uon appetito!
PIZZA
ITALY

Жизненный цикл объекта

1-поколение

2-поколение


Achievement unlocked
Превозмогун

Алгоритм сборки мусора в 1-м поколении

Алгоритм сборки мусора во 2-м поколении

1. живые объекты помечаются меткой;

Алгоритм сборки мусора во 2-м поколении

1. живые объекты помечаются меткой;
2. прочесывание памяти для удаления мертвых объектов;

Алгоритм сборки мусора во 2-м поколении

1. живые объекты помечаются меткой;
2. прочесывание памяти для удаления мертвых объектов;
3. уплотнение объектов в памяти, что позволяет уменьшить ее фрагментацию.

Общая память изоляционных групп

```dart
void main(List<String> arguments) async {
  await Future.wait([
    Future.delayed(Duration(seconds: 2), () {
      var a = const ['22'];
      var b = 44;
      var cat = const Cat('Max');
      var c = ['45'];
      print('Main cat: ${identityHashCode(cat)}'); // 693067717
      print('Main a: ${identityHashCode(a)}'); // 405082921
      print('Main b: ${identityHashCode(b)}'); // 510444
      print('Main c: ${identityHashCode(c)}'); // 959463167
    }),
    Isolate.run(() => isolateFoo()),
  ]);
}

void isolateFoo() async {
  var cat = const Cat('Max');
  final a = const ['22'];
  var b = 44;
  var c = ['45'];
  print('Isolate cat: ${identityHashCode(cat)}'); // 693067717
  print('Isolate a: ${identityHashCode(a)}'); // 405082921
  print('Isolate b: ${identityHashCode(b)}'); // 510444
  print('Isolate c: ${identityHashCode(c)}'); // 243495788

  b += 3;
  print('Isolate b: ${identityHashCode(b)}'); // 545247
  await Future.delayed(Duration(seconds: 2));
}
```

```dart
void main(List<String> arguments) async {
  await Future.wait([
    Future.delayed(Duration(seconds: 2), () {
      var a = const ['22'];
      var cat = const Cat('Max');
      var b = ['45'];
      var c = 44;
      var d = 'hi!';
      var e = 23.1;
      print('Main cat: ${identityHashCode(cat)}'); // 640354238
      print('Main a: ${identityHashCode(a)}'); // 896670331
      print('Main b: ${identityHashCode(b)}'); // 913878150
      print('Main c: ${identityHashCode(c)}'); // 510444
      print('Main d: ${identityHashCode(d)}'); // 1009204041
      print('Main e: ${identityHashCode(e)}'); // 15509272291868675
    }),
    Isolate.spawnUri(Uri.parse('new_group.dart'), [], null),
  ]);
}
```

```dart
void main() {
  var a = const ['22'];
  var cat = const Cat('Max');
  var b = ['45'];
  var c = 44;
  var d = 'hi!';
  var e = 23.1;
  print('IG cat: ${identityHashCode(cat)}'); // 478627355
  print('IG a : ${identityHashCode(a)}'); // 174622686
  print('IG b : ${identityHashCode(b)}'); // 167233800
  print('IG c : ${identityHashCode(c)}'); // 510444
  print('IG d: ${identityHashCode(d)}'); // 1009204041
  print('IG e: ${identityHashCode(e)}'); // 15509272291868675
}
```

Разница в передаче обычных и константных объектов между изолятами группы

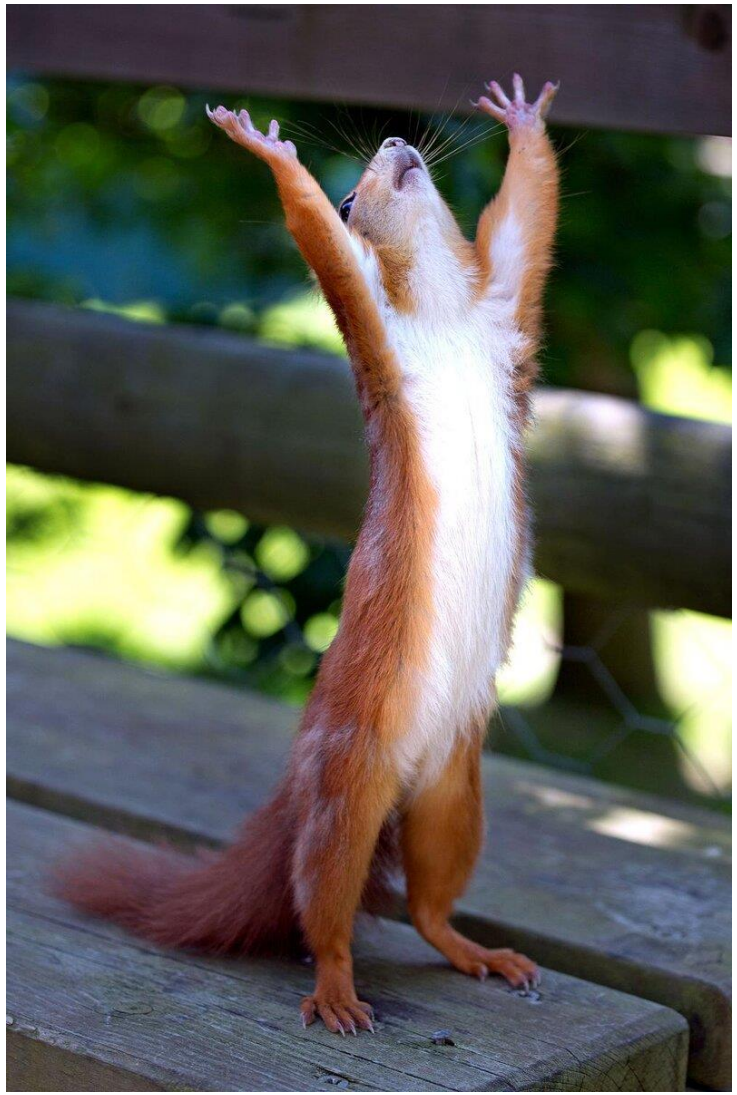
```dart
void main(List<String> arguments) async {
  var cat1 = const Cat('Max');
  print('Main cat1: ${identityHashCode(cat1)}');

  var cat2 = Cat('Max');
  print('Main cat2: ${identityHashCode(cat2)}');

  await Future.wait([
    Isolate.run(() => isolateFoo(cat1, 1)),
    Isolate.run(() => isolateFoo(cat2, 2)),
  ]);
  await Future.delayed(Duration(seconds: 3));
}

void isolateFoo(Cat cat, int number) async {
  print('Isolate cat$number: ${identityHashCode(cat)}');
  await Future.delayed(Duration(seconds: 1));
}
// Main cat1: 528416660 !!!
// Main cat2: 9390701
// Isolate cat1: 528416660 !!!
// Isolate cat2: 683512425
```

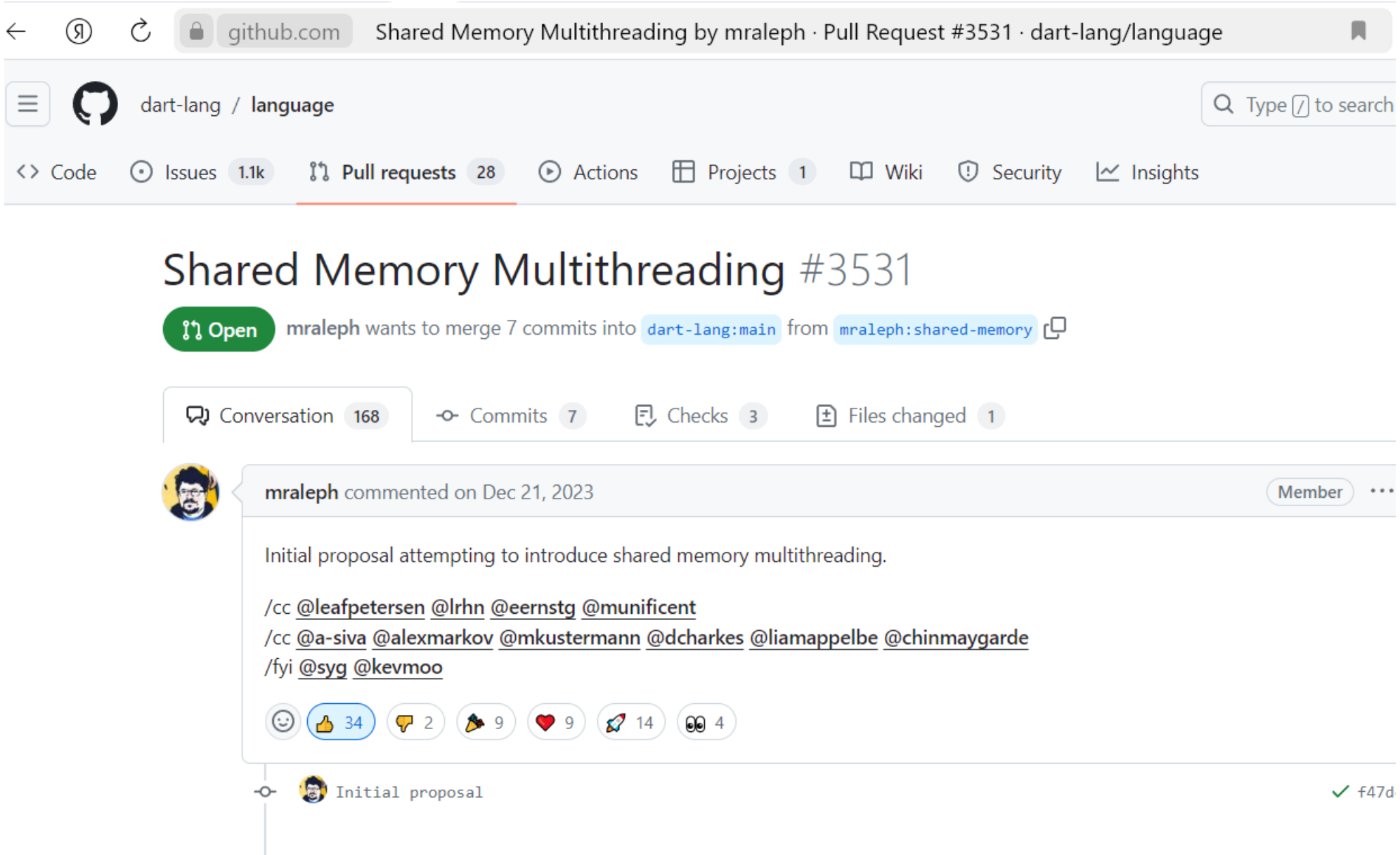
```
C:\code\dart\deep_isolate>bin\deep_isolate.exe 1000000 1
0:00:01.621376 count: 0

C:\code\dart\deep_isolate>bin\deep_isolate.exe 1000000 0
0:00:01.446942 count: 0
```

Создание изолята из другого изолята

```dart
import 'dart:isolate';
import 'dart:async';

void main() async {
  print('Main isolate started');
  await Future.wait([
    Isolate.run(firstIsolate),
    Future.delayed(Duration(seconds: 1)),
  ]);
}

Future<void> firstIsolate() async {
  print('First isolate started');
  await Future.wait([
    Isolate.run(secondIsolate),
    Future.delayed(Duration(seconds: 1)),
  ]);
}

Future<void> secondIsolate() async {
  print('Second isolate started');
}
// Main isolate started
// First isolate started
// Second isolate started
```

Будущее изолятов




github.com
Shared Memory Multithreading by mraleph · Pull Request #3531 · dart-lang/language
dart-lang / language
Type / to search
Code
Issues 1.1k
Pull requests 28
Actions
Projects 1
Wiki
Security
Insights
Shared Memory Multithreading #3531
Open
mraleph wants to merge 7 commits into dart-lang:main from mraleph:shared-memory
Conversation 168
Commits 7
Checks 3
Files changed 1
mraleph commented on Dec 21, 2023
Member
Initial proposal attempting to introduce shared memory multithreading.
/cc @leafpetersen @lrhn @eernstg @munificent
/cc @a-siva @alexmarkov @mkustermann @dcharkes @liamappelbe @chinmaygarde
/fyi @syg @kevmoo
34
2
9
9
14
4
Initial proposal
f47d

Выводы

- Объекты простых общих типов данных представлены одним экземпляром;

Выводы

- Объекты простых общих типов данных представлены одним экземпляром;
- Константа времени компиляции представлена одним объектом только в рамках изоляционной группы;

Выводы

- Объекты простых общих типов данных представлены одним экземпляром;
- Константа времени компиляции представлена одним объектом только в рамках изоляционной группы;
- Передача константы времени компиляции и объектов простых общих типов данных между изолятами группы требует меньше затрат;

Выводы

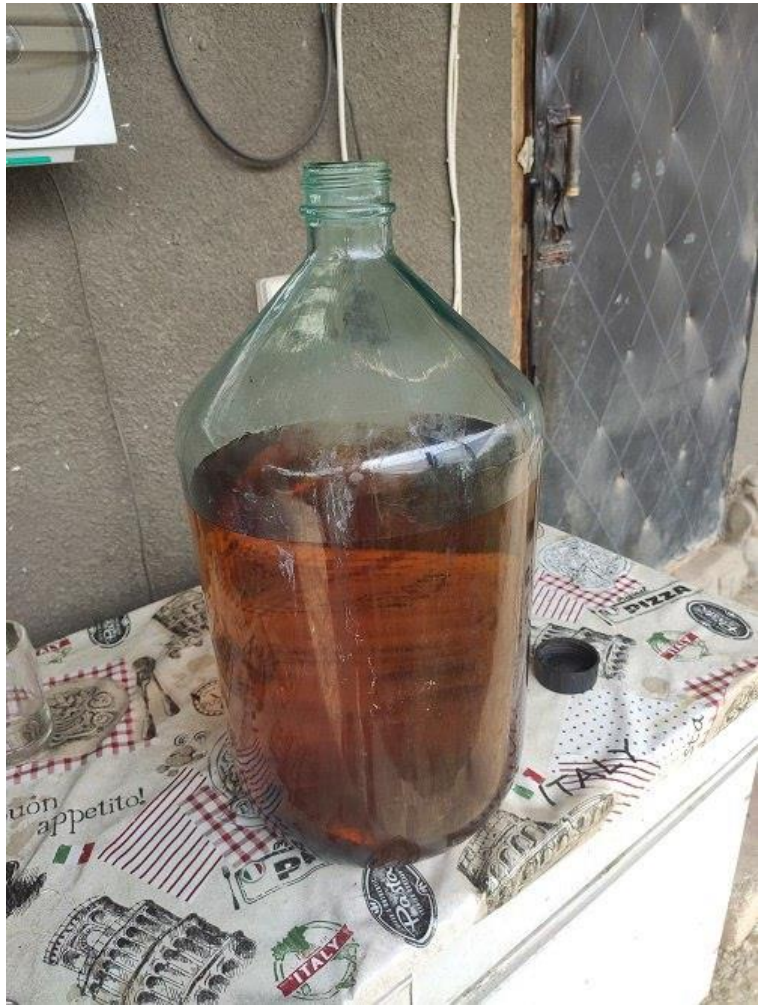
- Объекты простых общих типов данных представлены одним экземпляром;
- Константа времени компиляции представлена одним объектом только в рамках изоляционной группы;
- Передача константы времени компиляции и объектов простых общих типов данных между изолятами группы требует меньше затрат;
- От кого пошло, что изолят нельзя запускать из другого изолята?

Выводы

- Объекты простых общих типов данных представлены одним экземпляром;
- Константа времени компиляции представлена одним объектом только в рамках изоляционной группы;
- Передача константы времени компиляции и объектов простых общих типов данных между изолятами группы требует меньше затрат;
- От кого пошло, что изолят нельзя запускать из другого изолята?
- Изоляты – это не страшно!

Вопросы?

Станислав Чернышев, к.т.н., доцент СПбГУАП и СПБГЭУ


uon appetito!
PIZZA
ITALY