Оптимизация Generics, которая не работает

Andrey.G.Mikhaylov@Kaspersky.com

kaspersky

Случай с гитхаба

```
class BaseClass<T>
{
    private List<T> _list = new List<T>();

    public BaseClass()
    {
        Enumerable.Empty<T>();
    }

    public void Run()
    {
        for (var i = 0; i < 8000000; i++)
        {
            if (_list.Any())
            {
                return;
            }
        }
    }
}
```

https://github.com/dotnet/runtime/issues/3877

```
class BaseClass<T>
{
    private List<T> _list = new List<T>();

    public BaseClass()
    {
        Enumerable.Empty<T>();
    }

    public void Run()
    {
        for (var i = 0; i < 8000000; i++)
        {
            if (_list.Any())
            {
                return;
            }
        }
    }
}
```

```
class DerivedClass : BaseClass<object>
{
}
```

https://github.com/dotnet/runtime/issues/3877

```
new BaseClass<object>().Run()
```

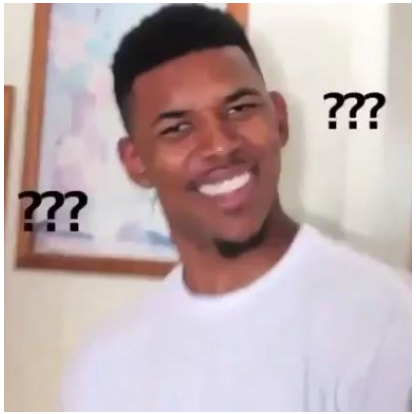
```
new DerivedClass().Run()
```

https://alexandrnikitin.github.io/blog/dotnet-generics-under-the-hood/


```
new BaseClass<object>().Run()
```


```
new DerivedClass().Run()
```

https://alexandrnikitin.github.io/blog/dotnet-generics-under-the-hood/



1x

3x

```
new BaseClass<object>().Run()
```

⏱️

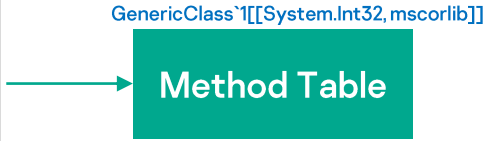
```
new DerivedClass().Run()
```

https://alexandrnikitin.github.io/blog/dotnet-generics-under-the-hood/

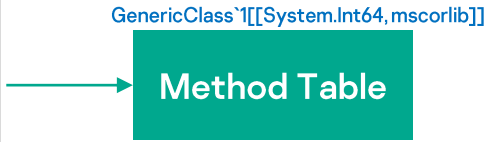
Как код generics хранится в памяти

```
class GenericClass<T>
{
    private T _value;

    public T GetValue()
    {
        return _value;
    }
}
```

```
class GenericClass<int>
{
    private int _value;

    public int GetValue()
    {
        return _value;
    }
}
```



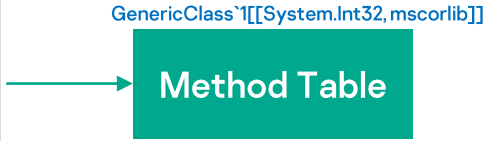

```
class GenericClass<long>
{
    private long _value;

    public long GetValue()
    {
        return _value;
    }
}
```



```
class GenericClass<int>
{
    private int _value;

    public int GetValue()
    {
        return _value;
    }
}
```

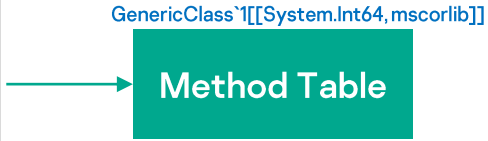

```
class GenericClass<long>
{
    private long _value;

    public long GetValue()
    {
        return _value;
    }
}
```

```
class GenericClass<int>
{
    private int _value;

    public int GetValue()
    {
        return _value;
    }
}
```

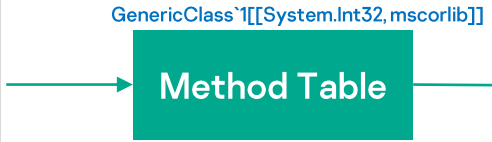


Машинный код..?

```
class GenericClass<long>
{
    private long _value;

    public long GetValue()
    {
        return _value;
    }
}
```



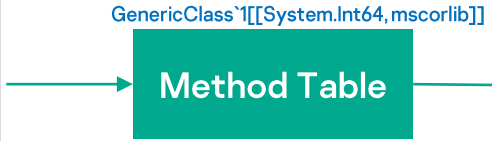
```
class GenericClass<int>
{
    private int _value;

    public int GetValue()
    {
        return _value;
    }
}
```



Машинный код (псевдокод)

```
int GetValue()
{
    return _value;
}
```

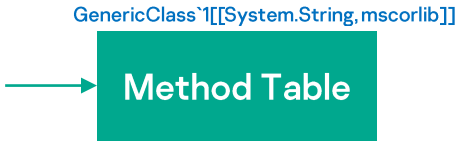
```
class GenericClass<long>
{
    private long _value;

    public long GetValue()
    {
        return _value;
    }
}
```



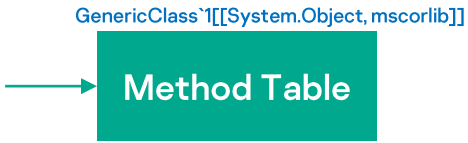
Машинный код (псевдокод)

```
long GetValue()
{
    return _value;
}
```

```
class GenericClass<string>
{
    private string _value;

    public string GetValue()
    {
        return _value;
    }
}
```

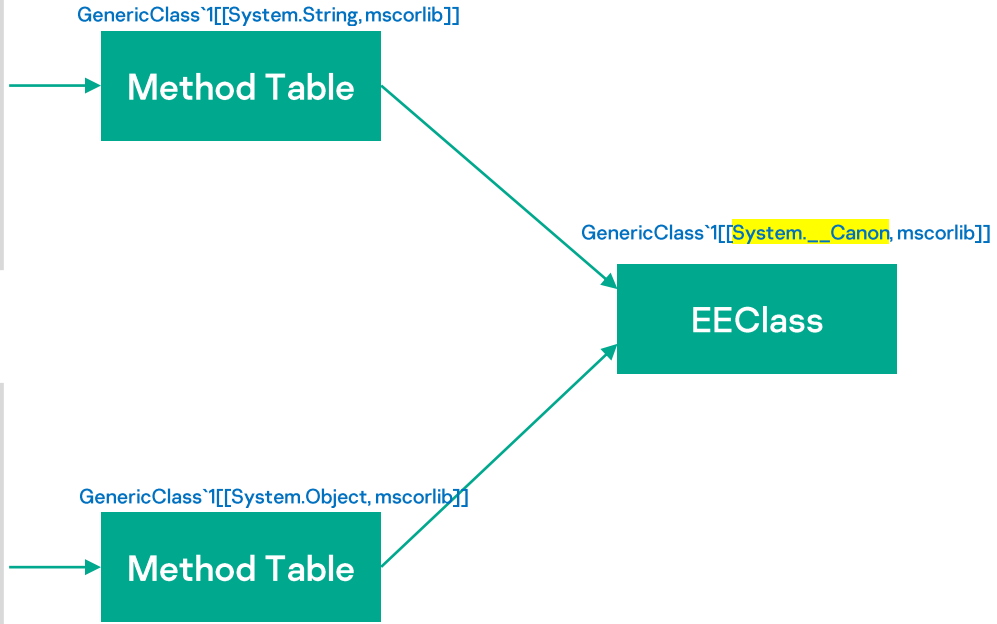



```
class GenericClass<object>
{
    private object _value;

    public object GetValue()
    {
        return _value;
    }
}
```

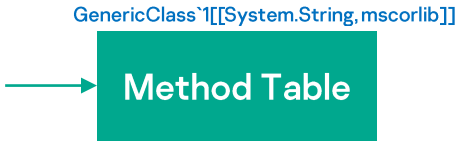


```
class GenericClass<string>
{
    private string _value;

    public string GetValue()
    {
        return _value;
    }
}
```

```
class GenericClass<object>
{
    private object _value;

    public object GetValue()
    {
        return _value;
    }
}
```

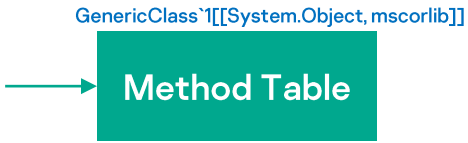


```
class GenericClass<string>
{
    private string _value;

    public string GetValue()
    {
        return _value;
    }
}
```



Машинный код..?

```
class GenericClass<object>
{
    private object _value;

    public object GetValue()
    {
        return _value;
    }
}
```



```
class GenericClass<string>
{
    private string _value;

    public string GetValue()
    {
        return _value;
    }
}
```

GenericClass`1[[System.String, mscorlib]]

Method Table

```
class GenericClass<object>
{
    private object _value;

    public object GetValue()
    {
        return _value;
    }
}
```

GenericClass`1[[System.Object, mscorlib]]

Method Table

Машинный код (псевдокод)

```
__Canon GetValue()
{
        return _value;
}
```

Какие проблемы у общего кода

```
class GenericClass<T>
{
    void Foo()
    {

    }
}
```

```
class GenericClass<T>
{
    void Foo()
    {
        Enumerable.Empty<T>();
    }
}
```

```
class GenericClass<T>
{
    void Foo()
    {
        Enumerable.Empty<T>();
    }
}
```


Машинный код (псевдокод)

```
class GenericClass<__Canon>
{
    void Foo()
    {
        Enumerable.Empty<__Canon>();
    }
}
```

```
class GenericClass<T>
{
    void Foo()
    {
        Enumerable.Empty<T>();
    }
}
```

Машинный код (псевдокод)

```
class GenericClass<__Canon>
{
    void Foo()
    {
        Enumerable.Empty<__Canon>();
    }
}
```

Какой машинный код позвать для Empty<>()?

```
class GenericClass<T>
{
    void Foo()
    {
        Enumerable.Empty<T>();
    }
}
```

Машинный код (псевдокод)

```
class GenericClass<__Canon>
{
    void Foo()
    {
        Enumerable.Empty<__Canon>();
    }
}
```

Какой машинный код позвать для Empty<>()?

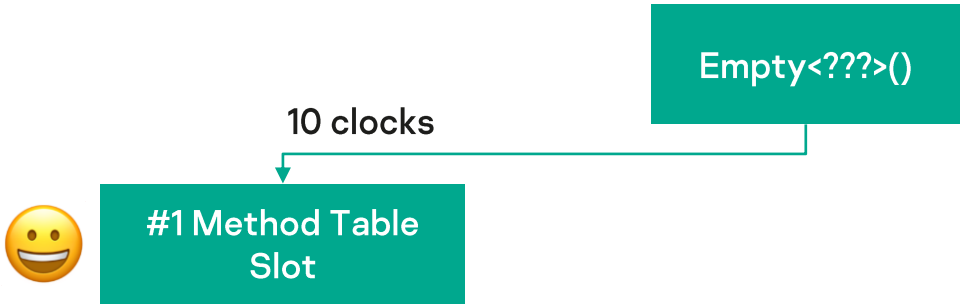
Empty<string>()

Empty<int>()

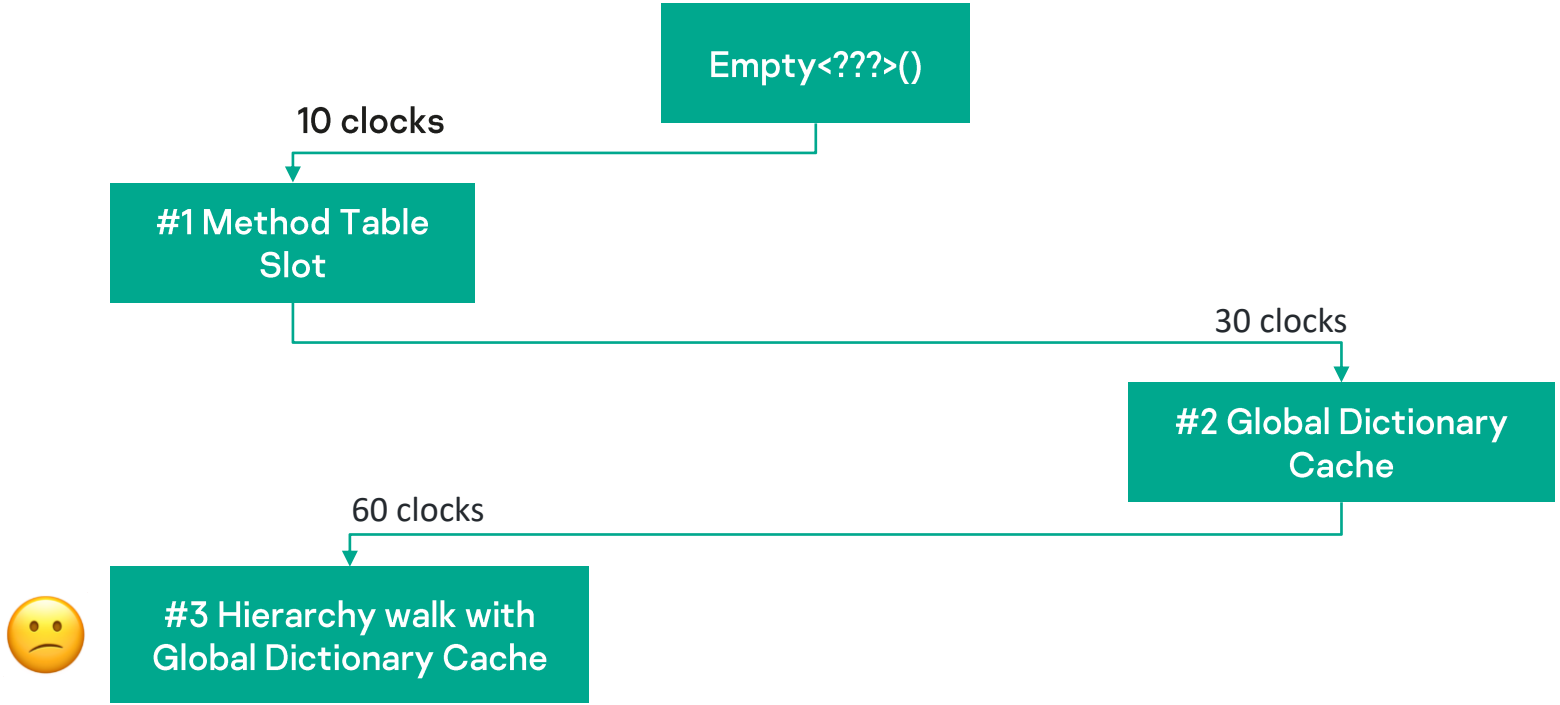
...

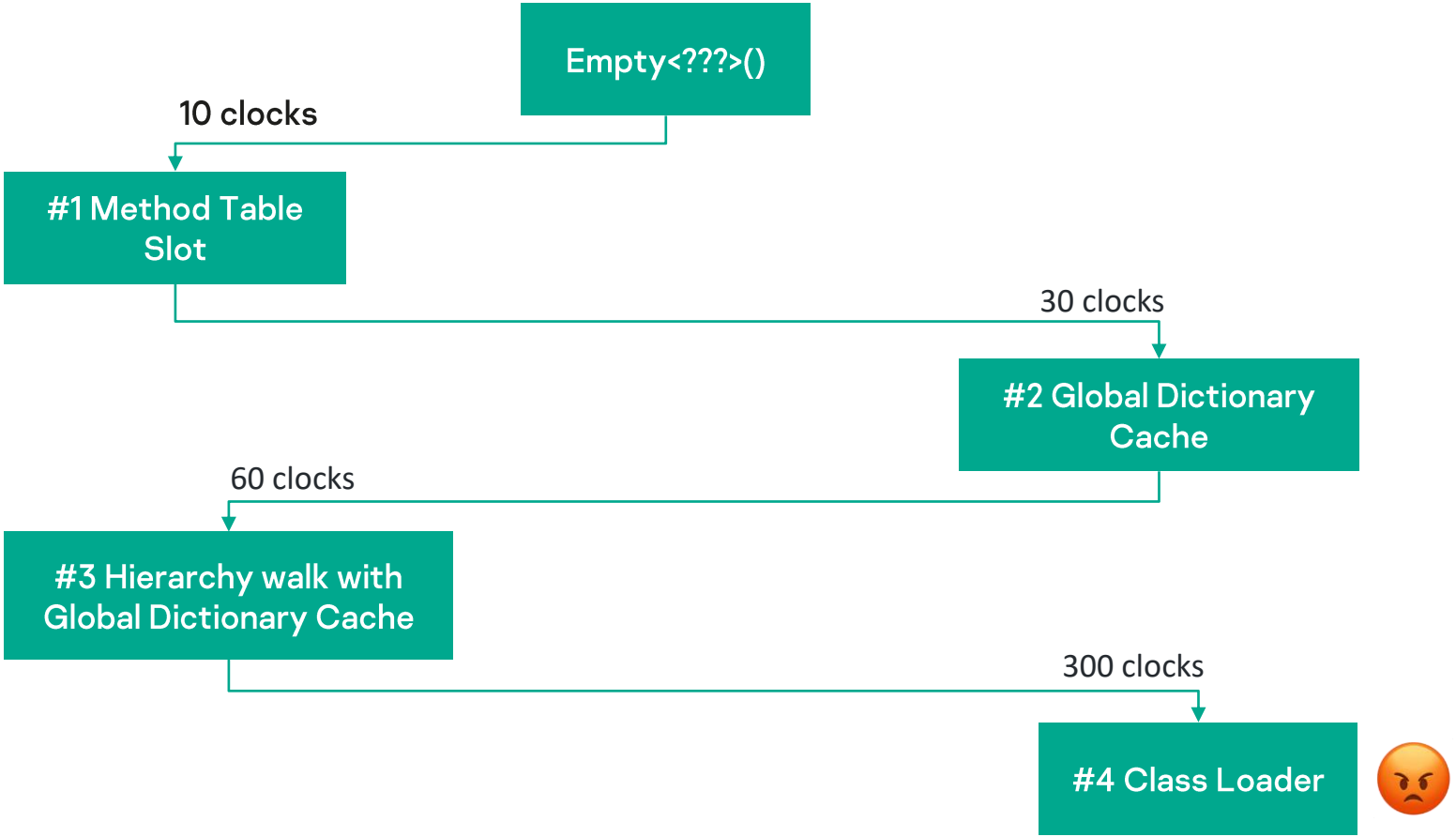
Как рантайм ищет generic метод

Empty<???>()


Empty<???>()
10 clocks
#1 Method Table Slot


Empty<???>()
10 clocks
#1 Method Table Slot
30 clocks
#2 Global Dictionary Cache
😐


Empty<???>()
10 clocks
#1 Method Table Slot
30 clocks
#2 Global Dictionary Cache
60 clocks
#3 Hierarchy walk with Global Dictionary Cache


Empty<???>()
10 clocks
#1 Method Table Slot
30 clocks
#2 Global Dictionary Cache
60 clocks
#3 Hierarchy walk with Global Dictionary Cache
300 clocks
#4 Class Loader

Method Table

...

Slot 1

Runtime

Method Table

...

Slot 1

Slot 2

Slot 3

Slot 4

Slot 5

Slot 6

Slot 7

```
void Method1()
{
}

void Method2()
{
}

void Method3()
{
}
```

```
class BaseClass<T>
{
    private List<T> _list = new List<T>();

    public BaseClass()
    {
        Enumerable.Empty<T>();
    }

    public void Run()
    {
        for (var i = 0; i < 8000000; i++)
        {
            if (_list.Any())
            {
                return;
            }
        }
    }
}
```

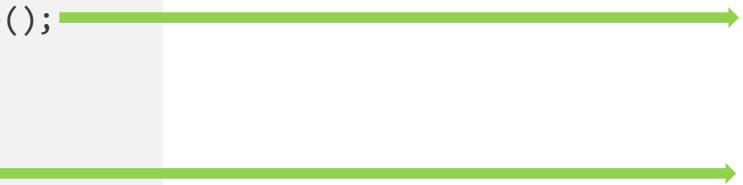
```
class BaseClass<T>
{
    private List<T> _list = new List<T>();

    public BaseClass()
    {
        Enumerable.Empty<T>();
    }

    public void Run()
    {
        for (var i = 0; i < 8000000; i++)
        {
            if (_list.Any())
            {
                return;
            }
        }
    }
}
```

#1 Method Table

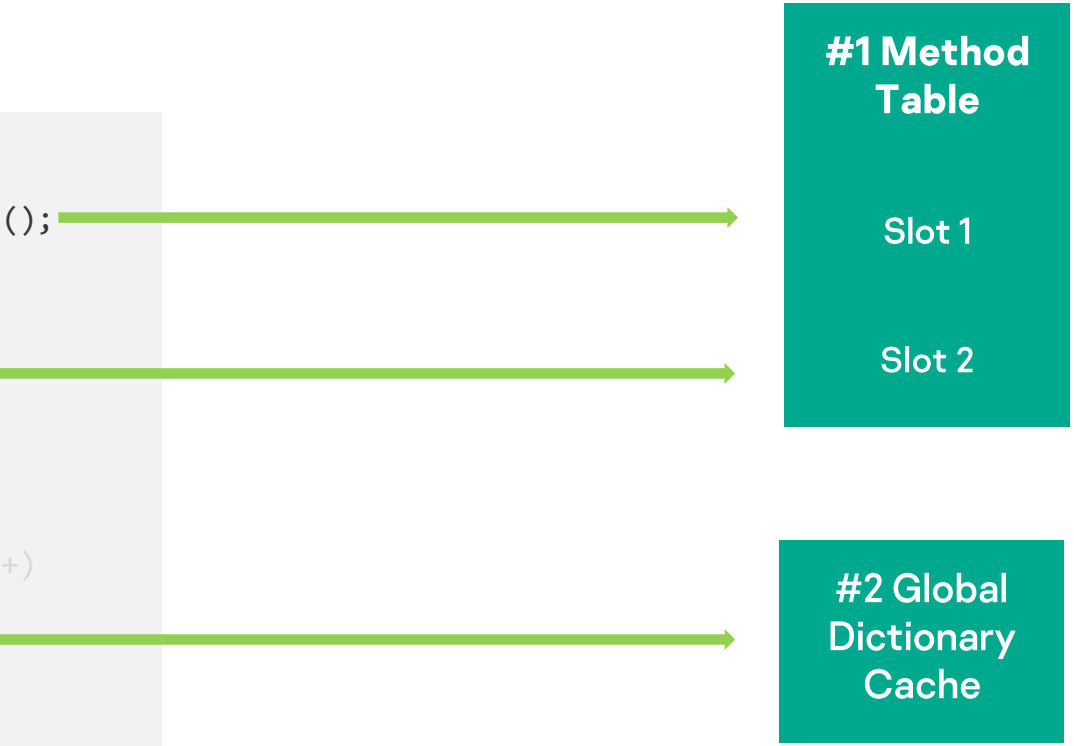
Slot 1

Slot 2

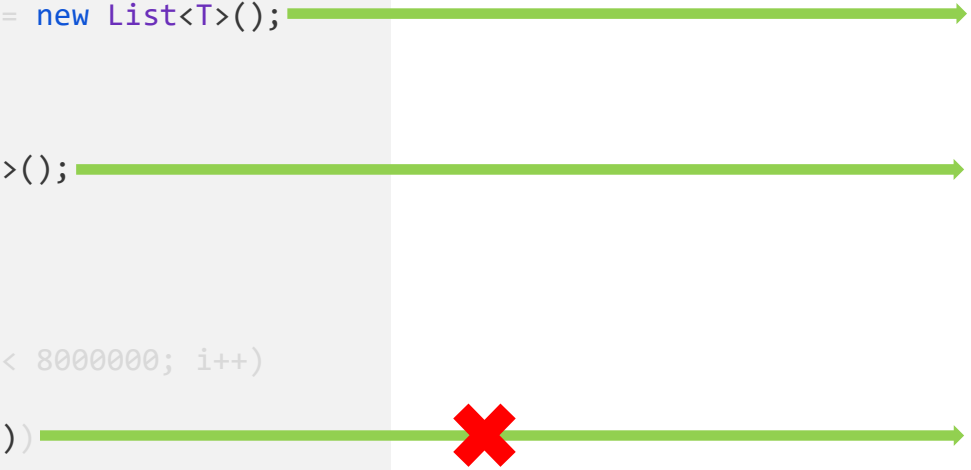
```
class BaseClass<T>
{
    private List<T> _list = new List<T>();

    public BaseClass()
    {
        Enumerable.Empty<T>();
    }

    public void Run()
    {
        for (var i = 0; i < 8000000; i++)
        {
            if (_list.Any())
            {
                return;
            }
        }
    }
}
```



```
class BaseClass<T>
{
    private List<T> _list = new List<T>();

    public BaseClass()
    {
        Enumerable.Empty<T>();
    }

    public void Run()
    {
        for (var i = 0; i < 8000000; i++)
        {
            if (_list.Any())
            {
                return;
            }
        }
    }
}
```



```
class BaseClass<T> DerviedClass
{
    private List<T> _list = new List<T>();

    public BaseClass()
    {
        Enumerable.Empty<T>();
    }

    public void Run()
    {
        for (var i = 0; i < 8000000; i++)
        {
            if (_list.Any())
            {
                return;
            }
        }
    }
}
```



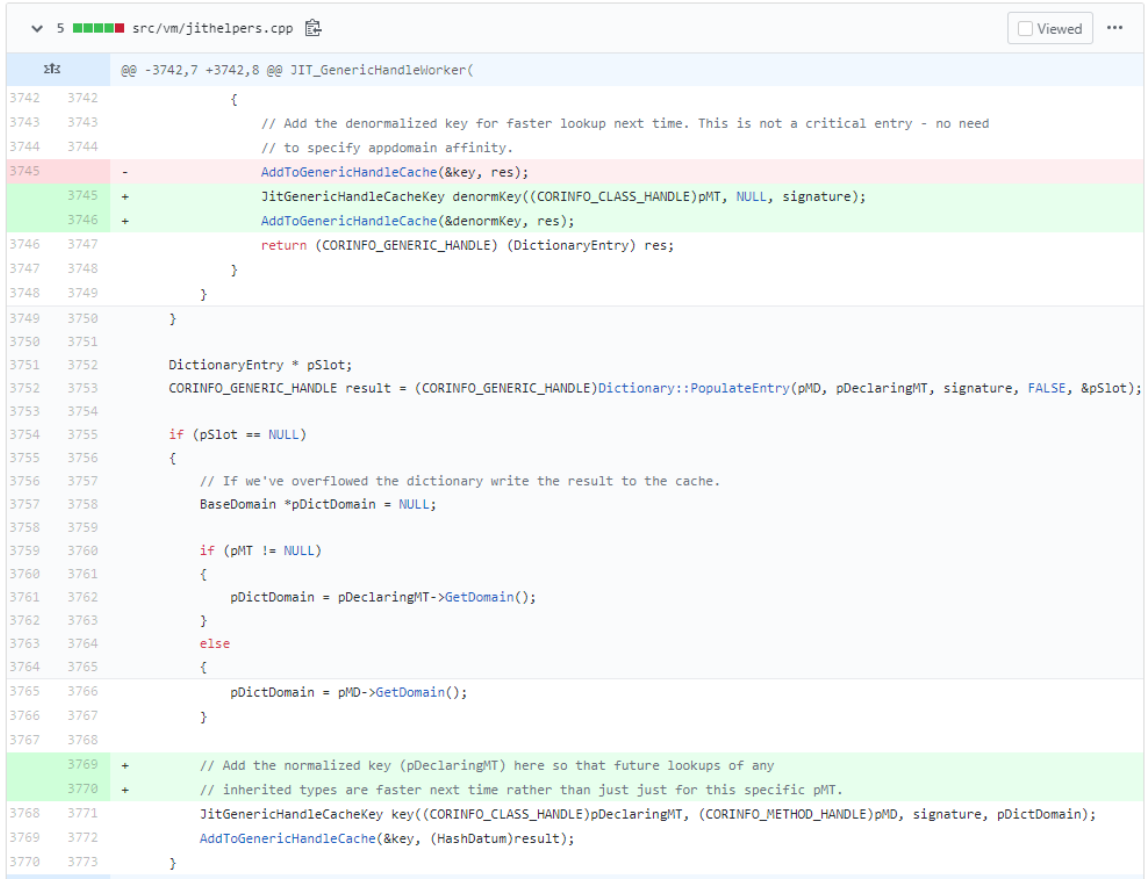
```
class BaseClass<T> DerviedClass
{
    private List<T> _list = new List<T>();

    public BaseClass()
    {
        Enumerable.Empty<T>();
    }

    public void Run()
    {
        for (var i = 0; i < 8000000; i++)
        {
            if (_list.Any())
            {
                return;
            }
        }
    }
}
```

#1 Method Table

Slot 1

Slot 2

#3 Hierarchy walk with Global Dictionary Cache



https://github.com/dotnet/coreclr/pull/618/files

Как помочь рантайму в оптимизации

```
class BaseClass<T> DerviedClass
{
    private List<T> _list =

    public BaseClass()
    {

    }

    public void Run()
    {
        for (var i = 0; i < 8000000; i++)
        {
            if (_list.Any())
            {
                return;
            }
        }
    }
}
```

#2 Global Dictionary Cache


```
class BaseClass<T> DerviedClass
{
    private List<T> _list =

    public BaseClass()
    {

    }

    public void Run()
    {
        for (var i = 0; i < 8000000; i++)
        {
            if (_list.Any())
            {
                return;
            }
        }
    }
}
```

#2 Global Dictionary Cache

```
class BaseClass<T>
{
    ...

    public void Foo1()
    {
    }

    public void Foo2()
    {
    }
}
```

```
class BaseClass<T> DerviedClass
{
    private List<T> _list =

    public BaseClass()
    {

    }

    public void Run()
    {
        for (var i = 0; i < 8000000; i++)
        {
            if (_list.Any())
            {
                return;
            }
        }
    }
}
```

#1 Method Table

Slot 1

Slot 2

Slot 3

Slot 4

#2 Global Dictionary Cache


```
new BaseClass<object>().Run()
```

#3 Hierarchy walk with Global Dictionary Cache


```
new DerivedClass().Run()
```

#2 Global Dictionary Cache

#1 Method Table


```
new BaseClass<object>().Run()
```

#3 Hierarchy walk with Global Dictionary Cache

#1 Method Table


```
new DerivedClass().Run()
```

1x

0.3x

```
new BaseClass<object>().Run()
```

3x

0.3x

```
new DerivedClass().Run()
```

Выводы

1/ Код методов Generic классов шарится, если параметр класса Reference Type

2/ При вызове метода содержащего Generic вызов рантайму приходится искать для него машинный код.

3/ Скорость вызова зависит от ряда оптимизаций применяемых рантаймом: method table, global cache, hierarchy cache, class loader.

4/ Можно подсказать рантайму оптимальный путь выполнения добавив фейковые методы

Источники

Александр Никитин

https://alexandrnikitin.github.io

https://alexandrnikitin.github.io/blog/dotnet-generics-under-the-hood/

Крис МакКинси

https://github.com/cmckinsey

https://github.com/dotnet/coreclr/pull/618/files

Гитхаб обсуждение

https://github.com/dotnet/runtime/issues/3877

Ура! Мы живы

Андрей Михайлов

dotNET разработчик
Password Manager и Safe Kids

ТГ: @andrew_mikhaylov

kaspersky