

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;
            }
        }
    }
}
```

```
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>
{
}
```

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



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

1x

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

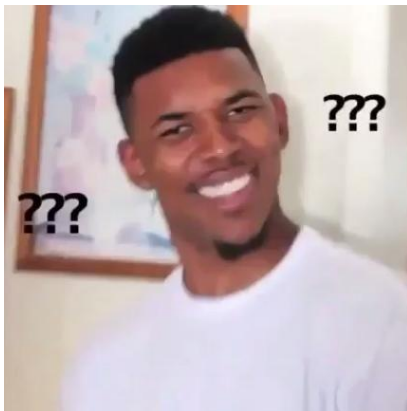


3x

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

1x

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



3x

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

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;
    }
}
```

GenericClass`1[[System.Int32,mscorlib]]

Method Table

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

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

GenericClass`1[[System.Int64,mscorlib]]

Method Table

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

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

GenericClass`1[[System.Int32,mscorlib]]

Method Table

GenericClass`1[[System.Int32,mscorlib]]

EEClass

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

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

GenericClass`1[[System.Int64,mscorlib]]

Method Table

GenericClass`1[[System.Int64,mscorlib]]

EEClass

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

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

GenericClass`1[[System.Int32,mscorlib]]

Method Table

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

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

GenericClass`1[[System.Int64,mscorlib]]

Method Table

Машинный код..?

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

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

GenericClass`1[[System.Int32,mscorlib]]

Method Table

Машинный код (псевдокод)

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

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

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

GenericClass`1[[System.Int64,mscorlib]]

Method Table

Машинный код (псевдокод)

```
long 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

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

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

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

Method Table

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

EEClass

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

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

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

Method Table



```
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..?


```
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

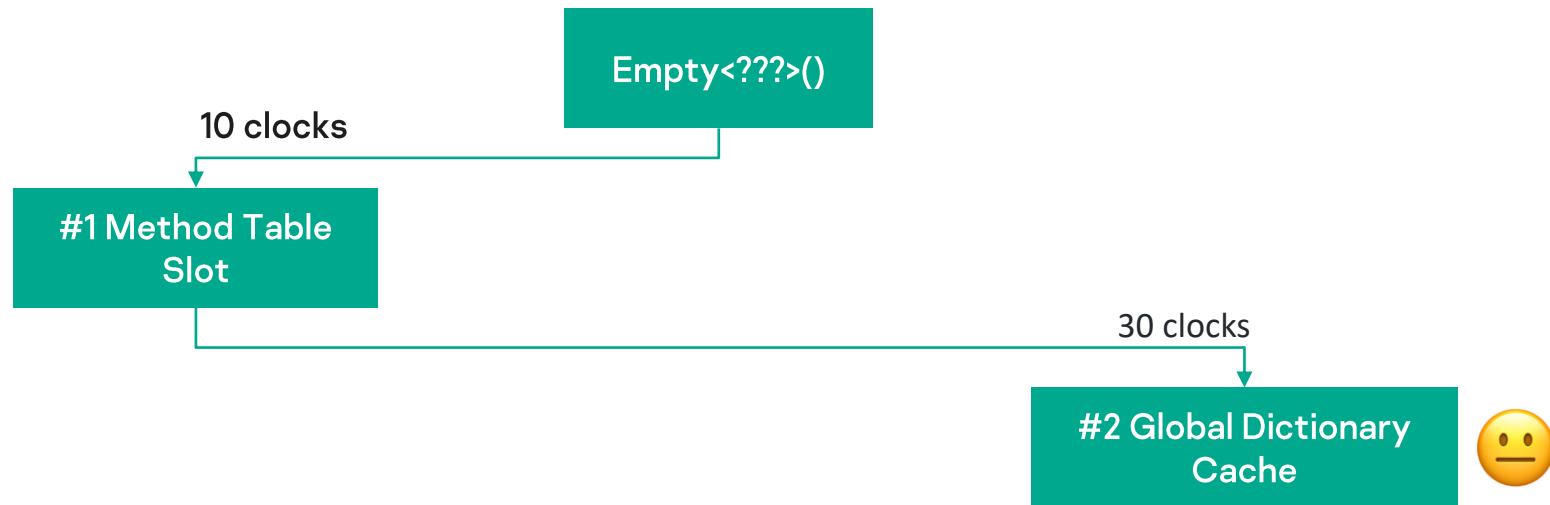

```
Empty<??>()
```



#1 Method Table
Slot

10 clocks

Empty<??>()



Empty<??>()

10 clocks

#1 Method Table
Slot

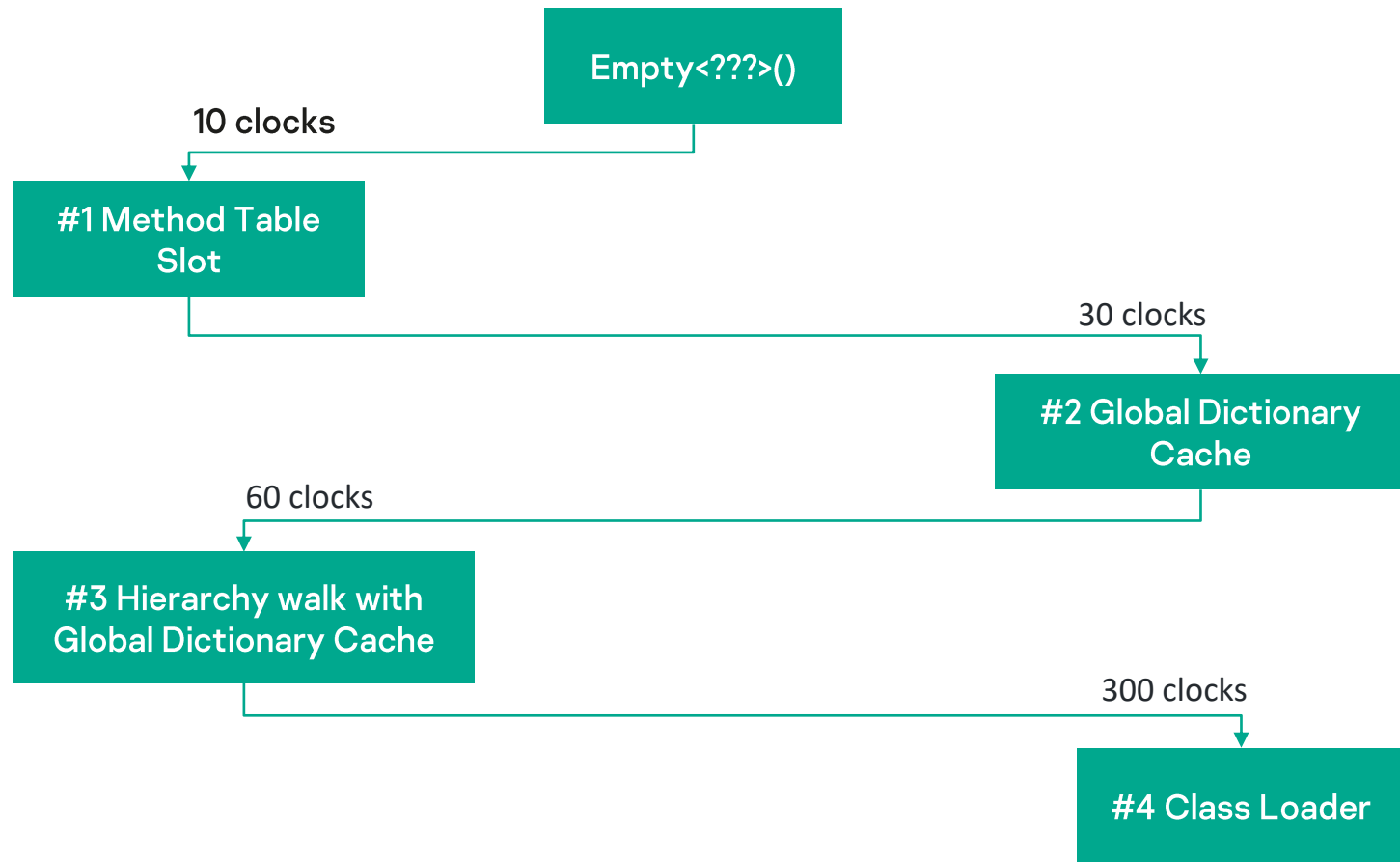
30 clocks

#2 Global Dictionary
Cache

60 clocks

#3 Hierarchy walk with
Global Dictionary Cache





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()
{
}
```



Method Table

...

Slot 1

Slot 2

Slot 3

```
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;
            }
        }
    }
}
```

#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;
            }
        }
    }
}
```

#1 Method Table

Slot 1

Slot 2

#2 Global Dictionary Cache

```
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



```
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

▼ 5 ■■■■■ src/vm/jithelpers.cpp

☐ Viewed ...

ix	@@ -3742,7 +3742,8 @@ JIT_GenericHandleWorker(
3742	3742 {
3743	3743 // Add the denormalized key for faster lookup next time. This is not a critical entry - no need
3744	3744 // to specify appdomain affinity.
3745	- 3745 AddToGenericHandleCache(&key, res);
3745	+ 3745 JitGenericHandleCacheKey denormKey((CORINFO_CLASS_HANDLE)pMT, NULL, signature);
3746	+ 3746 AddToGenericHandleCache(&denormKey, res);
3746	3747 return (CORINFO_GENERIC_HANDLE) (DictionaryEntry) res;
3747	3748 }
3748	3749 }
3749	3750 }
3750	3751 DictionaryEntry * pSlot;
3751	3752 CORINFO_GENERIC_HANDLE result = (CORINFO_GENERIC_HANDLE)Dictionary::PopulateEntry(pMD, pDeclaringMT, signature, FALSE, &pSlot);
3752	3753
3753	3754
3754	3755 if (pSlot == NULL)
3755	3756 {
3756	3757 // If we've overflowed the dictionary write the result to the cache.
3757	3758 BaseDomain *pDictDomain = NULL;
3758	3759
3759	3760 if (pMT != NULL)
3760	3761 {
3761	3762 pDictDomain = pDeclaringMT->GetDomain();
3762	3763 }
3763	3764 else
3764	3765 {
3765	3766 pDictDomain = pMD->GetDomain();
3766	3767 }
3767	3768
3769	+ 3769 // Add the normalized key (pDeclaringMT) here so that future lookups of any
3770	+ 3770 // inherited types are faster next time rather than just just for this specific pMT.
3768	3771 JitGenericHandleCacheKey key((CORINFO_CLASS_HANDLE)pDeclaringMT, (CORINFO_METHOD_HANDLE)pMD, signature, pDictDomain);
3769	3772 AddToGenericHandleCache(&key, (HashDatum)result);
3770	3773 }

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;
            }
        }
    }
}
```



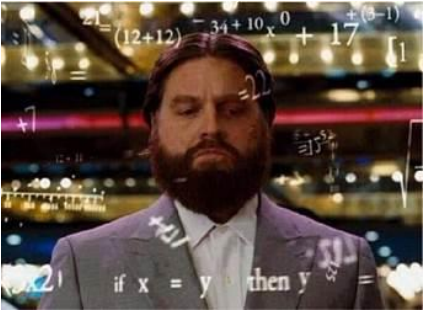
**#2 Global
Dictionary
Cache**


```
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()
    {
    }
}
```

#1 Method Table

Slot 1

Slot 2

Slot 3

Slot 4

```
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

1x

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



#3 Hierarchy walk with Global Dictionary Cache

3x

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

#2 Global Dictionary Cache

1x

#1 Method Table

0.3x

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



#3 Hierarchy walk with Global Dictionary Cache

3x

#1 Method Table

0.3x

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

1x

0.3x



3x

0.3x

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

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

Выводы

47

- 1/ Код методов **Generic** классов шарится, если параметр класса **Reference Type**
- 3/ Скорость вызова зависит от ряда оптимизаций применяемых рантаймом: **method table**, **global cache**, **hierarchy cache**, **class loader**.

- 2/ При вызове метода содержащего **Generic** вызов рантайму приходится искать для него машинный код.
- 4/ Можно подсказать рантайму оптимальный путь выполнения добавив фейковые методы



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