Сборка мусора в .NET

Андрей Акиньшин

Барнаульское сообщество .NET разработчиков

bug.ineta.ru
www.facebook.com/groups/dotnetbarnaul/

Управляемая куча

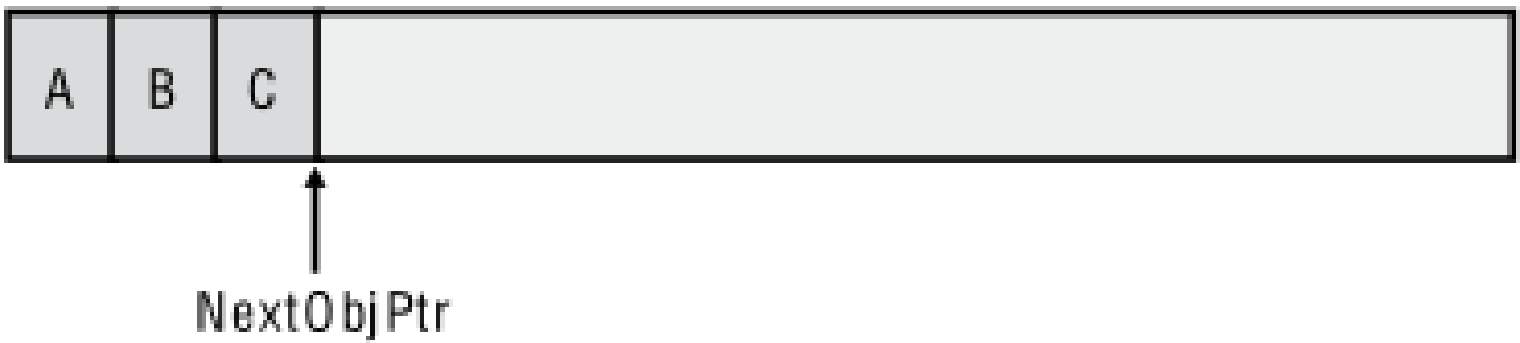
Плюсы:

- Автоматическое управление памятью

Минусы:

- Автоматическое управление памятью

Выделение памяти



Начинаем собирать мусор

- Поколение 0 заполнено
- Вызван метод *GC.Collect*
- Свободной памяти осталось мало
- Выгрузка домена приложения
- Завершение работы CLR

Алгоритм

- Фаза маркировки
- Фаза сжатия
- Финализация

Маркировка

Корни приложения:

- Статические поля
- Параметры методов
- Локальные переменные
- Регистры процессора
- GCHandle
- Очередь финализации

Поколения

- Чем младше объект, тем короче его время жизни
- Чем старше объект, тем длиннее его время жизни
- Убрать часть кучи быстрее, чем всю кучу
- Тяжело перемещать что-то большое

Поколения

- Поколение 0 ($\sim$ *256 KB*)
- Поколение 1 ($\sim$ *2 MB*)
- Поколение 2 ($\sim$ *10 MB*)

GC.MaxGeneration == 2 (но всегда ли?)

Куча больших объектов (LOH)

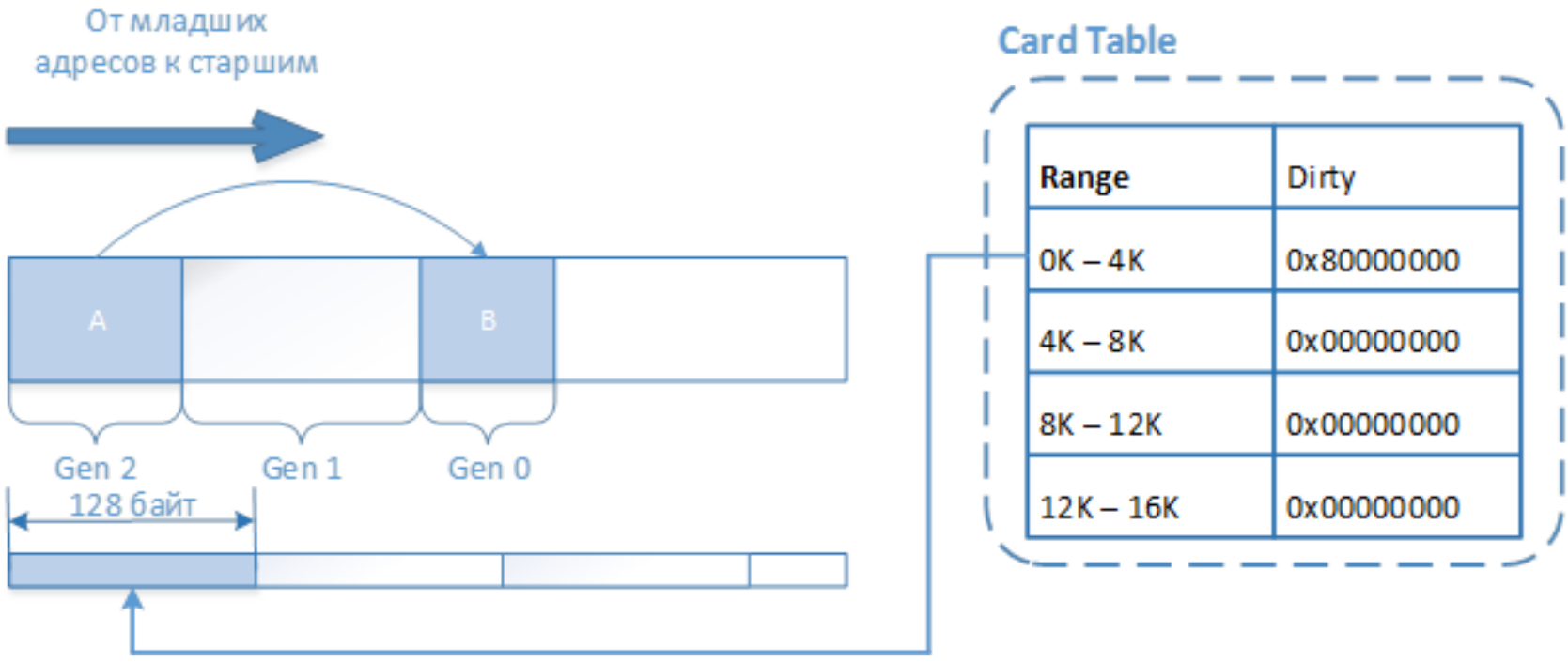
- Объекты размером от 85000 байт
- Объекты не перемещаются
- Хорошая практика: пулинг больших объектов

No more memory fragmentation in .NET 4.5.1:

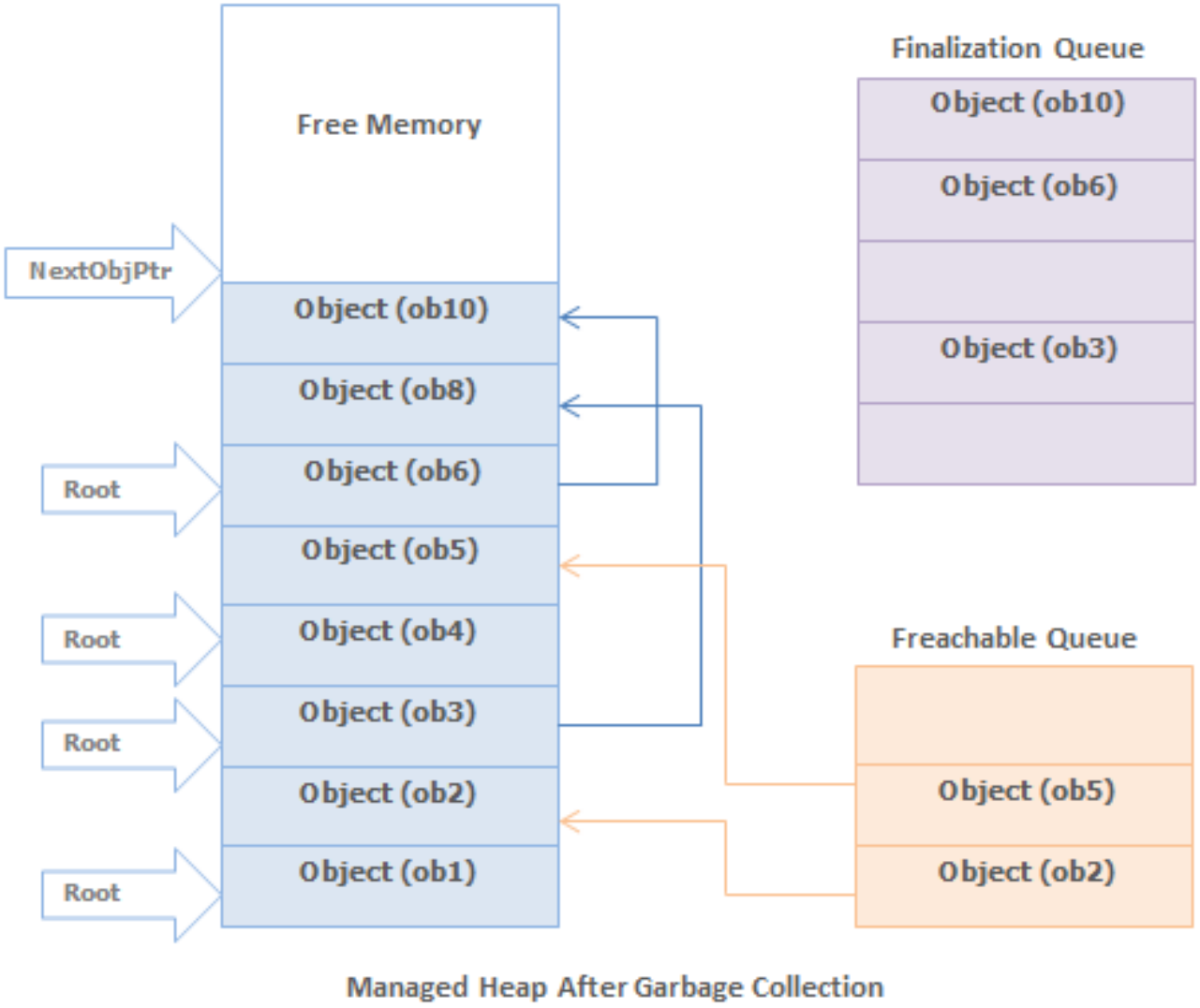
```
GCSettings.LargeObjectHeapCompactionMode =
    GCLargeObjectHeapCompactionMode.CompactOnce;
GC.Collect(); // This will cause the LOH to be compacted (once).
```

Write barrier method

Card table: храним по одному биту для каждого 128-байтного диапазона



Финализация



Managed Heap After Garbage Collection

Паттерн Disposable

```
// Design pattern for a base class.
public class Disposable : IDisposable
{
  private bool disposed = false;

  //Implement IDisposable.
  public void Dispose()
  {
    Dispose(true);
    GC.SuppressFinalize(this);
  }

  protected virtual void Dispose(bool disposing)
  {
    if (!disposed)
    {
      if (disposing)
      {
        // Free other state (managed objects).
      }
      // Free your own state (unmanaged objects).
      // Set large fields to null.
      disposed = true;
    }
  }

  // Use C# destructor syntax for finalization code.
  ~Disposable()
  {
    Dispose(false); // Simply call Dispose(false).
  }
}
```

Типичная ошибка

```
public static class Global
{
  public static event EventHandler Update;
}

public class Foo
{
  public Foo()
  {
    Global.Update += OnUpdate;
  }

  ~Foo()
  {
    Global.Update -= OnUpdate;
  }

  private void OnUpdate(object sender, EventArgs e)
  {
    // Some logic
  }
}
```

Слабые ссылки

```
var reference = new WeakReference(new object());

Console.WriteLine(reference.Target == null); // False

GC.Collect();
Thread.Sleep(1000);

Console.WriteLine(reference.Target == null); // True
```

Объекты с гарантированной финализацией

Свойства *CriticalFinalizerObject* и его наследников:

- Ранняя JIT-компиляция
- Поздний вызов финализаторов
- Гарантированный вызов финализаторов

Режимы работы GC

- Рабочая станция / сервер
- Однопоточный / фоновый (конкурентный)

GCLatencyMode

- *Batch* — Отключает фоновую сборку
- *Interactive* — Включает фоновую сборку
- *LowLatency* — Сборка поколения 2 выполняется только при GC.Collect() или малом количестве свободной памяти
- *SustainedLowLatency* — Продолжительные периоды нежелательной сборки мусора

Ручное управление

- Ручной вызов сборки мусора GC.Collect() GCCollectionMode $\in$ { Default, Forced, Optimized}
- *AddMemoryPressure()* и *RemoveMemoryPressure()*
- *class MemoryFallPoint*

Время вызова сборщика мусора

```
using System;
using System.Threading;

public static class Program {
  public static void Main() {
    var t = new Timer(TimerCallback, null, 0, 2000);
    Console.ReadLine();
  }

  private static void TimerCallback(Object o) {
    Console.WriteLine("In TimerCallback: " + DateTime.Now);
    GC.Collect();
  }
}
```

Время вызова сборщика мусора

```
public class ImageWithCircle
{
  private const int Size = 10000;
  private readonly IplImage image;

  public ImageWithCircle()
  {
    image = Cv.CreateImage(new CvSize(Size, Size), BitDepth.U8, 3);
    DrawCircle();
  }

  ~ImageWithCircle()
  {
    Cv.ReleaseImage(image);
  }

  public void Save()
  {
    image.SaveImage("image.tif");
  }

  public void DrawCircle()
  {
    image.FloodFill(new CvPoint(Size / 2, Size / 2), CvColor.White);
    image.Circle(new CvPoint(Size / 2, Size / 2), Size / 4,
                 CvColor.Random(), 10);
  }
}
```

Время вызова сборщика мусора

```
public class ImageWithCircle
{
  private const int Size = 10000;
  private readonly IplImage image;

  public ImageWithCircle()
  {
    image = Cv.CreateImage(new CvSize(Size, Size), BitDepth.U8, 3);
    DrawCircle();
  }

  ~ImageWithCircle()
  {
    Cv.ReleaseImage(image);
  }

  public void Save()
  {
    image.SaveImage("image.tif");
    GC.KeepAlive(this);
  }

  public void DrawCircle()
  {
    image.FloodFill(new CvPoint(Size / 2, Size / 2), CvColor.White);
    image.Circle(new CvPoint(Size / 2, Size / 2), Size / 4,
                 CvColor.Random(), 10);
  }
}
```

Mono

- Boehm
- SGen

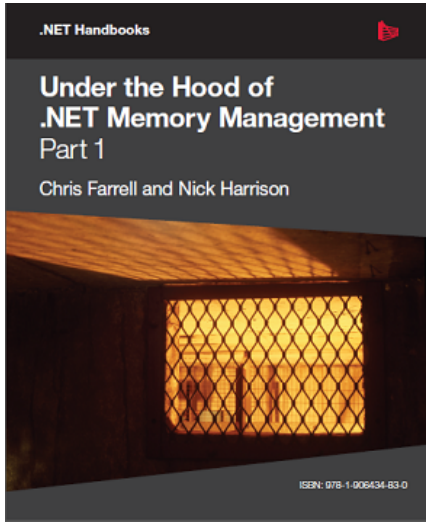
Диагностика

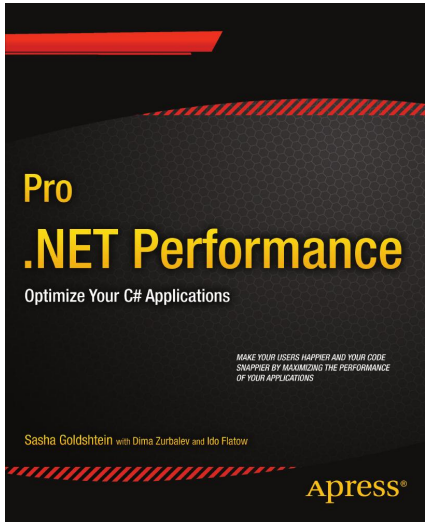
- MEMORY LEAKS: THE EASY WAY
  Not in this talk. Shell out $500 for a decent memory profiler.
  (c) Sasha Goldshtein

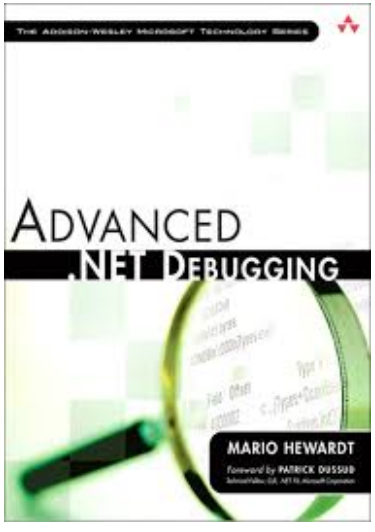
Диагностика

- MEMORY LEAKS: THE EASY WAY
  Not in this talk. Shell out $500 for a decent memory profiler.
  (c) Sasha Goldshtein
- sos.dll
- sosex.dll
- Dumps
- ...

Хорошие книжки







Спасибо за внимание!