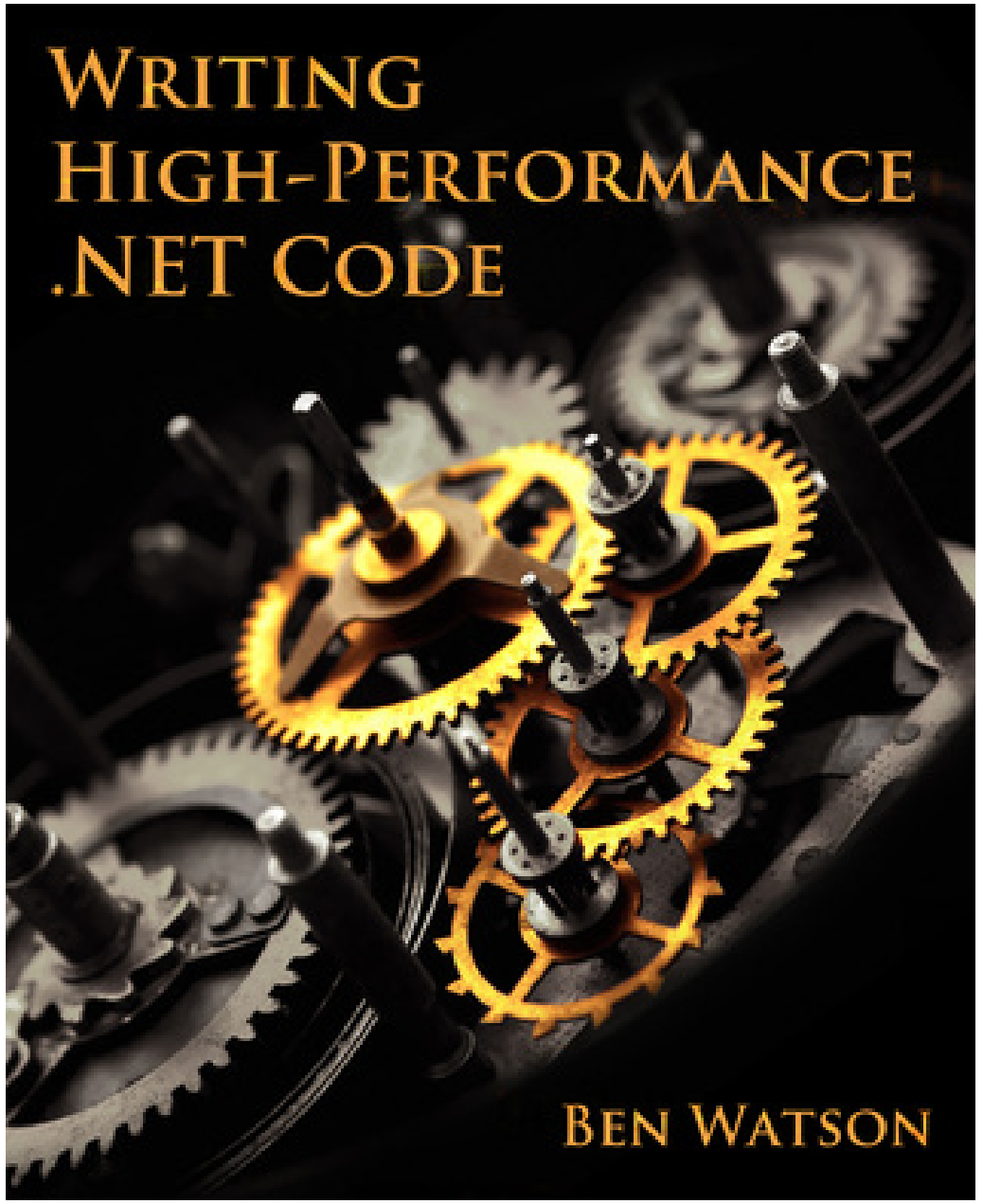
.NET Garbage Collection

Fundamentals of .NET Garbage Colleciton

Writing High-Perforamnce .NET Code

- by Ben Watson
- www.writinghighperf.net

> *In .NET, you need to think of memory performance at least as much as CPU performance. It is so fundamental to smooth .NET operation, that the most significant chunk of this book's content is dedicated to just this topic.* - Ben Watson, Writing High-Perforamnce .NET Code



Heaps

Heap = Process memory reserved for allocation of data.

A mangaged process has 2 kinds of heaps:

- **Native Heap**: used by Windows & CLR for unmanaged memory: Windows API, OS data structures, the CLR itself
- **Managed Heap** (*GC Heap*): used by the CLR to allocation Objects subjected to garbage collection

The Managed Heap

The Managed Heap is devided in 2 sections.

- **Small Object Heap (SOH)** for objects < 85000 bytes which are most of all objects
- **Large Object heap (LOH)** for objects >= 85000 bytes mostly arrays and strings

Size of an empty object: 16 bytes (32bit), 32 bytes (64bit).
Actually its 12 bytes and 24 bytes but the memory manager can't handle that very well.

The Small Object Heap

The Small Object Heap is divided in 3 generations.

- **Gen 0** the 1st genertaion
- **Gen 1** the 2nd generation (serves as cache for Gen 2)
- **Gen 2** the 3rd and last generation

Once in Gen 2, objects stay here till they *die*.

Objects are *alive* as long as they are referenced by a **GC root**, e.g. static variables, managed threads or pinned handles.

Segments

Segments = Fixed sized parts of memory owned by SOH or LOH.

- LOH & Gen 2 can span multiple segements
- Gen 0 & Gen 1 always reside in the same segment

Allocation

The CLR tries to allocate new objects 3 times:

Fast Allocation

1. Try to allocate at the end of Gen 0

Slow Allocation

2. Try to allocate anywhere within Gen 0
3. Try to expand Gen 0 and allocate at the new end

If expansion exceeds segement size a *Garbage Collection* is triggered!

Garbage Collection

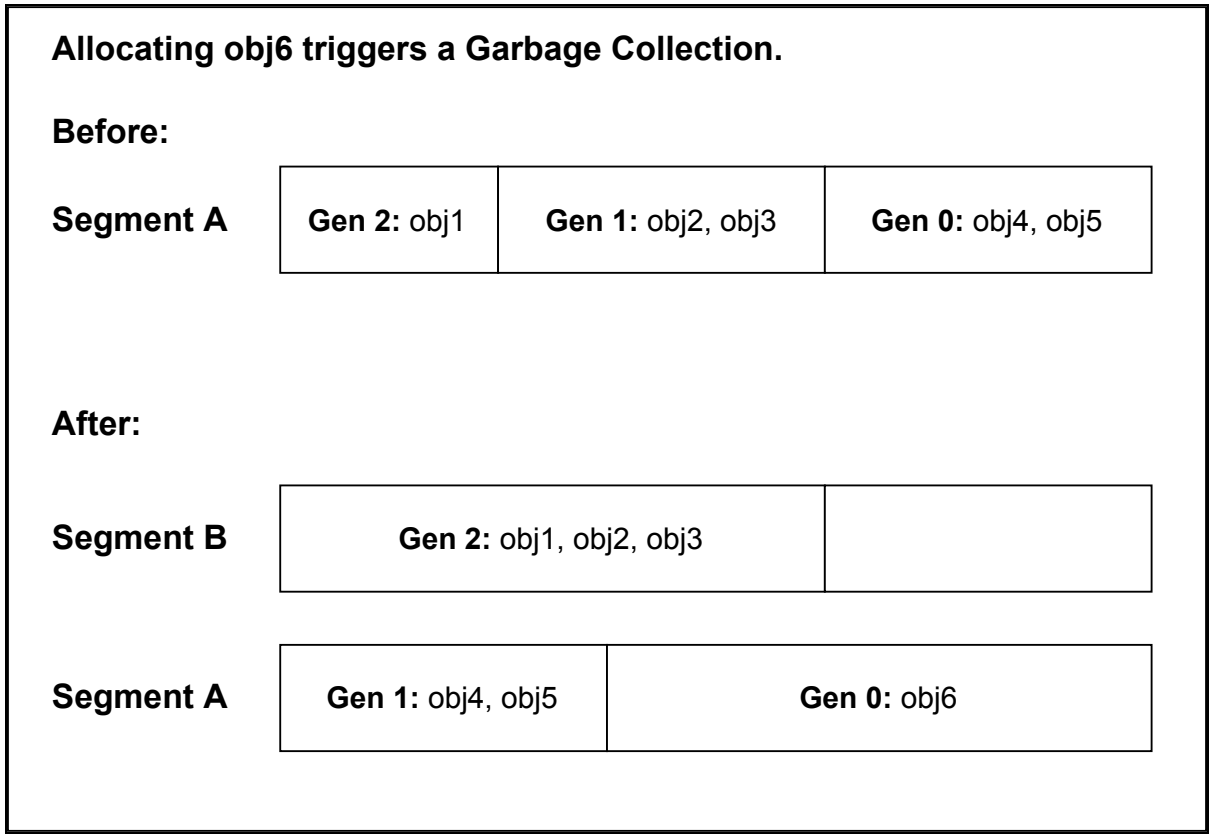
Garbage Collection consists of 4 phases:

1. **Suspension**: Suspend all managed threads
2. **Mark**: Mark all objects reachable from any GC root as *seen* (alive)
3. **Compact**: Relocate and promote seen objects to reduce memory fragmentation
4. **Resume**: Resume all suspended threads

Garbage Collection happens only for the triggering generation and lower generations. Gen 2 colleciton also collects Gen 1 and Gen 0. Gen 1 collection also collects Gen 0.

All non-*seen* objects die of here!

Example



Note: We just moved 5 objects around to allocate 1 object.

The Most Important Rule

> *[...] the garbage collector was explicitly designed with this idea in mind:* ***Collect objects in Gen 0 or not at all.*** - Ben Watson, Writing High-Perforamnce .NET Code

- Garbage collection for Gen 1 and Gen 2 are expensive

Time spend on garbage collection is time not spend on the rest of your program!