

Username: Pralay Patoria **Book:** Pro .NET Performance. No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copy right laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Index

A

- [Aggregation](#)
- [Algorithm optimization](#)
 - [approximation](#)
 - [c-approximation algorithm](#)
 - [maximum cut](#)
 - [traveling salesman](#)
 - [gap compression](#)
 - [index compression](#)
 - [introduction](#)
 - [memoization and programming](#)
 - [all-pairs-shortest-paths](#)
 - [calculation](#)
 - [dynamic programming](#)
 - [edit distance](#)
 - [Fibonacci numbers](#)
 - [recursion](#)
 - [memory and string dictionary](#)
 - [probabilistic algorithms](#)
 - [fermat primality test](#)
 - [maximum cut](#)
 - [taxonomic complexity](#)
 - [Big-Oh notation](#)
 - [complexity classes and TM](#)
 - [master theorem](#)
 - [variable length encoding](#)
- [Allocation profilers](#)
 - [CLR profiler](#)
 - [allocation graph button](#)
 - [heap dumps](#)
 - [histogram buttons](#)
 - [main report view](#)
 - [objects](#)
 - [time line button](#)
 - [Visual view](#)
 - [memory-related statistics](#)
 - [Visual Studio profiler](#)
 - [allocation view](#)
 - [function details view](#)
 - [JackCompiler.exe](#)
 - [object lifetime view](#)
 - [summary view](#)
- [All-pairs-shortest-paths](#)
 - [Floyd-Warshall algorithm](#)
 - [observation](#)
 - [real-life scenarios](#)
 - [shortest path \(SP\)](#)
- [ANTS memory profiler](#)
 - [comparison](#)
 - [directories](#)
 - [individual string](#)
 - [strings](#)
- [ASP.NET applications](#)
 - [features](#)
 - [mechanisms](#)
 - [overview](#)
 - [per-compiling](#)
 - [pitfalls](#)
 - [process model](#)
 - [server affinities](#)
 - [server-side output cache](#)
 - [tracing and debugging](#)
 - [view state](#)
- [Asynchronous sockets](#)

B

- Base class library (BCL)
- Big-Oh notation algorithm
- Built-in Windows tools
 - counter categories
- Event Tracing for Windows (ETW)
 - custom ETW providers
 - kernel providers
 - partial list
 - PerfMonitor
 - PerfView tool
 - sampling
 - Windows Performance Toolkit
- performance counters
 - components
 - configuring performance
 - logs and alerts
 - monitoring memory
 - points
 - steps
 - tools

C

- C++ AMP
 - algorithm
 - array_view class
 - framework
 - GPU-based programs
 - I/O devices
 - lambda function's
 - parallel_foreach
- c-approximation algorithm
- C# 5 async methods
 - async and await
 - asynchronous version
 - continuation
 - dispose method
 - exception handling
 - forecasts variable
 - language features
 - TaskScheduler API
 - WinRT (Windows Runtime) APIs
- C++/CLI language extensions
 - advantages
 - .cpp file
 - DoWork method
 - GetName method
 - GlobalAlloc memory
 - IL code vs. native code
 - marshal_as helper library
 - marshal_context object
 - Windows 8 WinRT
- Chatty protocols
- CLR allocation profiler
 - allocation graph button
 - heap dumps
 - histogram buttons
 - main report view
 - objects
 - time line button
 - Visual view
- CLR generics
 - C++ templates
 - limitations
 - turing-complete template resolution mechanism
 - type-safe template class
- java
 - code conversion
 - Object array
 - type erasure
- Code generation
 - dynamic lightweight code
 - LCG
- Collections
 - built-in arrays
 - cache considerations
 - cache-core-memory relationships
 - cache hit
 - cache line
 - cache miss ratio

- code generation
- hitting the memory wall
- large main memories
- Level 1 cache for data
- Level 2 cache for data
- Level 3 cache for data
- Level 1 cache for program instructions
- matrix multiplication
- concurrent collections in .NET Framework
 - AddOrUpdate method
 - ConcurrentBagT
 - ConcurrentDictionaryK,V
 - ConcurrentQueueT
 - ConcurrentStackT
- custom
 - disjoint-set/union-find data structure
 - one-shot collections
 - skip list data structure
- IEnumerableT
- .NET framework
 - additional requirements
 - amortized $O(1)$ cost collections
 - DictionaryK,V
 - HashSetT
 - LinkedListT
 - ListT
 - SortedDictionaryK,V
 - SortedListK,V
 - SortedSetT
 - StackT
 - storage requirements
 - strings
 - ubiquitous space-time tradeoff
- COM Callable Wrapper (RCW)
- COM interoperability. *See also* C++/CLI language extensions
- apartments
 - ASP.NET
 - cross-thread parameter
 - STA, MTA and NTA
 - threads and objects
 - types
- CCW and RCW
- exceptions
- lifetime management
- managed client
- NoPIA
- practices
- TLB import and code access security
- unmanaged client
- wrappers

Compare-And-Swap (CAS)

Complexity classes. *See* Turing machine (TM)

Concurrency and parallelism. *See also* Synchronization

- advanced patterns, TPL
- advantage
- C# 5 async methods
 - async and await
 - asynchronous version
 - continuation
 - Dispose method
 - exception handling
 - forecasts variable
 - language features
 - TaskScheduler API
 - WinRT (Windows Runtime) APIs
- data parallelism
 - for and foreach loops
 - parallel LINQ
- frameworks
- GPU computing
 - C++ AMP
 - matrix multiplication
 - N-Body simulation
 - parallelism
 - tiles and shared memory
- harnessing parallelism
- introduction
- I/O operations
- Sutter, Herb
- task parallelism
 - class

- decomposition
- exceptions and cancellation
- Parallel.Invoke
- partition method
- QuickSort algorithm
- recursive algorithms
- TaskRun method
- thread
 - CPU
 - Intel i7 system
 - prime numbers
 - questions and answers
 - thread pools
- Concurrency profilers
 - contention
 - visualization
- Constrained execution region (CER)
- Content Delivery Networks (CDNs)
- C++ templates
 - limitations
 - turing-complete template resolution mechanism
 - type-safe template class

D

- Data access layer (DAL)
- Database and data access profilers
- Data-level parallelism. See Instruction-level parallelism (ILP)
- Data parallelism
 - for and foreach loops
 - Parallel.For
 - Parallel.ForEach
 - ParallelLoopState class
 - Stop method
 - testing loop
 - parallel LINQ (PLINQ)
 - advantages
 - AsParallel extension method
 - loops
 - queries
 - thread-local buffers
- Data serialization and deserialization
 - DataSet serialization
 - serializer benchmarks
 - comparison
 - operations/sec
 - serializers
- Denial-of-service (DOS)
- Disjoint-set/union-find data structure
- Domain Name Service (DNS) protocol
- Dynamic lightweight code generation
 - C# code
 - headers
 - __makeref and __refvalue
 - Marshal.PtrToStructure method
 - ReadPointer method
 - source code
 - structure
 - TcpHeader
 - third-party frameworks
 - TypedReference

E

- Event Tracing for Windows (ETW)
 - custom ETW providers
 - kernel providers
 - partial list
 - PerfMonitor
 - advantages
 - bottom up report
 - CPU, GC and JIT reports
 - GC analysis report
 - HTML files
 - JIT analysis report
 - runAnalyze command
 - source code folder
 - top down report

- TraceEvent
- PerfView tool
 - features
 - file view
 - MemoryLeak.exe
 - reference chain
 - sampling
- Windows Performance Toolkit (WPT)
 - call stack aggregation
 - detailed stack frames
 - graph
 - I/O operations
 - raw report
 - sampling profilers
 - SDK installation directory
 - XPerf.exe
- Exceptions

F

- Fermat's little theorem
- FIFO (first-in-first-out) queue
- Finalization, GC
 - automatic non-deterministics
 - bullet-proof feature
 - dispose pattern
 - event
 - experiment
 - manual deterministics
 - memoryleak
 - practices
 - queue
 - reference types
 - resources
 - resurrection
 - thread
 - unmanaged resources
- Flavors, GC
 - CER
 - disadvantages
 - GC flavors and sub-flavors
 - scenarios
 - server GC
 - threads
 - mark phase
 - suspension
 - sweep phase
 - types
 - workstation GC
 - concurrent GC
 - non-concurrent GC
 - types
- Floating-point addition (FADD)
- Floyd-Warshall algorithm

G

- Garbage collection (GC). *See also* Tracing garbage collection
 - allocation cost
 - coordination concepts
 - deallocation cost
 - definition
 - design goals
 - finalization
 - automatic mechanism
 - bullet-proof feature
 - dispose pattern
 - manual deterministics
 - practices
 - resources
 - unmanaged resources
 - flavors
 - application threads
 - GC flavors and sub-flavors
 - scenarios
 - server GC
 - workstation GC
 - free list management

- generational model
 - assumptions
 - background GC
 - large object heap
 - .NET implementation
 - practices
 - references
- interaction
 - CLR hosting
 - System.GC class
 - triggers
- management cost
- memory leak
- memory management
- optimizations
- practices
 - finalization
 - generational model
 - object graphs
 - paging and unmanaged memory
 - pinning
 - pooling objects
 - static code analysis (FxCop)
 - value types
- reference-counting garbage collection
- segments and virtual memory
 - approaches
 - CLR handles
 - COM objects and unmanaged code
 - ephemeral segment
 - fragmentation
 - managed code
 - memory space
 - segment/VM hoarding
 - VMMMap
- weak references
 - application code
 - event
 - GC handle
 - long weak references
 - short weak references
 - strong reference
 - various scenarios
- GC segments and virtual memory
 - approaches
 - CLR handles
 - COM objects and unmanaged code
 - ephemeral segment
 - fragmentation
 - managed code
 - memory space
 - segment/VM hoarding
 - VMMMap
- Generational model, GC
 - assumptions
 - background GC
 - large object heap
 - mid-life crisis
 - .NET implementation
 - large object/generation
 - old object/generation
 - pinned objects
 - types
 - youngest objects/generation
 - object life expectancies
 - practices
 - references
 - card table
 - JIT compiler
 - mark phase
 - micro-optimization
 - non-null type
 - statements
- Generic programming
 - ArrayList
 - ArrayList with boxed Point2D
 - binary search, sorted array
 - CLR generics
 - C++ templates
 - java
 - constraints

- closed generic type
- ComparableT
- IMathT interface
- interface
- interface vs. IEquatableT constraints
- Java
- ListT
- open generic type
- generic code conversion
- generics internals
- ListT
- non-boxing
- System.Object method
- type safety
- Generic type. *See* Generic programming
- Global assembly cache (GAC)
- GPU computing. *See also* C++ AMP
 - matrix multiplication
 - N-Body simulation
 - parallelism
 - tiles and shared memory
 - element
 - matrix multiplication algorithm
 - multiple threads
 - .NET
 - parallel_for_each
 - speed increase
 - tile_barrier objects
 - tile_static storage



H

- HTTP monitoring tools



I

- IL code vs. native code
- Image packers
- Index compression
- Instruction-level parallelism (ILP)
 - approaches
 - combined data
 - data chunks
 - loop functions
 - managed vs. unmanaged code
 - superscalar execution
 - trivial implementation
- Internet Information Services (IIS)
 - output cache
 - kernel-mode
 - user-mode cache
 - pool configuration
 - idle timeouts
 - processor affinity
 - recycling
 - web garden
 - types
- Interoperability. *See* COM interoperability
- I/O completion port (IOCP)
 - CreateIoCompletionPort
 - GetQueuedCompletionStatus
 - overlapped structure
 - Pack method
 - structure and operation
 - TestIOCP method
 - thread handles
- I/O concepts
 - copying memory
 - data buffer
 - unmanaged memory
 - exposing part
 - file I/O
 - cache hinting
 - unbuffered I/O
 - I/O completion port (IOCP)
 - CreateIoCompletionPort
 - GetQueuedCompletionStatus
 - overlapped structure

- Pack method
 - structure and operation
- TestIOCP method
 - thread handles
- .NET Thread Pool
 - overlapping
- performance gains
 - scatter-gather
 - synchronous and asynchronous
- I/O Profilers

J, K

- Java generics
 - code conversion
 - Object array
 - type erasure
- JIT compiler
 - common subexpression reduction
 - constant folding
 - .ini file
 - method inlining
 - range check elimination
 - array
 - loop function
 - standard optimizations
 - tail calling
 - GCD method
 - heuristics
 - integers
 - method
 - online details
- Just-In-Time Compiler (JIT)

L

- Large object heap (LOH)
- LIFO (last-in-first-out) queue
- Lightweight Code Generation (LCG)
- LINQ (Language INtegrated Query). See Parallel LINQ (PLINQ)

M

- Managed profile guided optimization (MPGO)
- Managed vs. unmanaged code
- Master theorem
- Memoization and programming
 - all-pairs-shortest-paths
 - Floyd-Warshall algorithm
 - observation
 - real-life scenarios
 - shortest path (SP)
 - calculation
 - dynamic programming
 - edit distance
 - Fibonacci numbers
 - recursion
- Memory profiler
 - ANTS
 - comparison
 - directories
 - individual string
 - strings
 - SciTech .NET
 - dump files
 - FileInformation instances
 - memory snapshots
 - post mortem
- Method inlining
- Metrics See Performance metrics
- Microbenchmark
 - code
 - environment
 - guidelines
 - poor design
 - conclusion

- framework
- as and is keyword
- operations
- virtual modifier

- Minimal spanning tree (MST)
- Multi-core background JIT compilation
- Multi-threaded apartment (MTA)
- Mutator threads. *See* Threads

N

- Nagle's algorithm
- Native Image Generator (NGen.exe). *See* Pre-JIT compilation
- N-Body simulation
- .NET. *See* Garbage collection (GC)
- .NET Thread Pool
- Networking
 - network protocols
 - chatty protocol session
 - encoding and redundancy
 - message chunking
 - pipelining
 - streaming
 - Server applications
 - sockets
 - asynchronous
 - buffer
 - Nagle's algorithm
 - registered I/O
- Neutral-threaded apartment (NTA)
- NGEN

O

- Optimization
 - CDNs
 - HTTP cache headers
 - dynamic content
 - static content
 - IIS components
 - client applications
 - configuring compression
 - dynamic compression
 - static compression
 - minification and bundling
 - bundles folder
 - CSS files
 - meaning
 - problems
 - script
 - use of

P

- Parallelism. *See* Concurrency and parallelism
- Parallel LINQ (PLINQ)
 - advantages
 - AsParallel extension method
 - loops
 - queries
 - thread-local buffers
- PerfMonitor
 - advantages
 - bottom up report
 - CPU, GC and JIT reports
 - GC analysis report
 - HTML files
 - JIT analysis report
 - runAnalyze command
 - source code folder
 - top down report
 - TraceEvent
- Performance measurement
 - allocation profilers
 - approaches
 - black-box testing

- CLR profiler
 - memory-related statistics
- Visual Studio profiler
- built-in Windows tools
 - Event Tracing for Windows (ETW)
 - performance counters
- concurrency profilers
- database and data access profilers
- I/O profilers
- memory profiler
 - ANTS
 - SciTech .NET
- microbenchmark
 - approaches
 - guidelines
 - poor design
- MSDN features
- time profilers
 - customization
 - guidelines
 - instrumentation profiling
 - profiling modes
 - sampling mode
 - Visual Studio sampling profiler
- tools
- white-box testing
- Performance metrics
 - application types
 - architecture
 - goals
 - application types
 - environment
 - statements
 - well-defined
 - partial list
 - process
 - software development lifecycle
- Performance patterns
 - code generation
 - dynamic lightweight code
 - source code
 - third-party frameworks
 - exceptions
- JIT compiler
 - .ini file
 - method inlining
 - range check elimination
 - standard optimizations
 - tail calling
 - processor-specific optimization
 - features
 - ILP
 - SIMD
- reflection
- startup
 - GAC
 - image packers
 - MPGO
 - multi-core JIT compilation
 - native image
 - pre-JIT compilation
 - reduce assemblies
 - require rebasing
 - types
- Pinning operation
 - approaches
 - disadvantages
 - fragmentation
 - GC handle
 - managed and unmanaged code
 - .NET memory management model
 - practices
- P/Invoke
 - binding
 - blittable types
 - code access security
 - C-style functions
 - DllImport attribute
 - FindFirstFile method
 - IntPtr
 - marshaller stubs

- direction, value and reference types
- HRESULT returns
- IL Stub Diagnostics
- managed code execution
- memory allocation
- ML and assembly code
- native code
- steps
- unmanaged code execution
- unmarshaling
- PInvoke.net and interop assistant
- WIN32_FIND_DATA struct
- Pipelining protocols
- Platform invoke. *See* P/Invoke
- Pre-JIT compilation
 - advantages
 - CLR
 - NGen.exe tool
 - shared-system scenarios
- Primary Interop Assemblies (PIA)
- Probabilistic algorithms
 - fermat primality test
 - maximum cut

Q

- QuickSort algorithm

R

- Recursive algorithms
- Reduction. *See* Aggregation
- Reference-counting garbage collection
- Reflection
- Registered I/O (RIO)
- Runtime Callable Wrapper (RCW)

S

- SciTech .NET memory profiler
 - dump files
 - FileInformation instances
 - memory snapshots
 - post mortem
- Serialization. *See* Data serialization and deserialization
- Server, Web application
 - asynchronous
 - controller classes
 - I/O operation
 - page
 - threads
 - cache objects
- Single instruction multiple data (SIMD)
 - AVX processor
 - data-level parallelism
 - FADD
 - integer value
 - JIT compiler
 - machine code
 - processor instructions
 - SSE instructions
- Single-threaded apartment (STA)
- Skip list data structure
- Socket buffer
- Static code analysis (FxCop)
- Streaming SIMD extensions (SSE)
- Sutter, Herb
- Synchronization
 - agent-based languages
 - cache considerations
 - invalidation/collision
 - localSums array
 - sequential method
 - sum variable
 - directions
 - execution history

- lock-free code
 - C# method
 - Compare-And-Swap (CAS)
 - data structure
 - DoWithCAST
 - general version
 - incorrect result
 - in-stack queued spinlocks
 - Interlocked.CompareExchange
 - lost updates
 - memory models and volatile variables
 - multiplication method
 - operating system
 - processors
 - Push method
 - spinlock
 - suffers
 - Task Parallel Library
 - thread
 - TryPop method
- machine instructions
- message-passing processor
- transactional memory
- windows mechanisms
 - data structure
 - internal implementation
 - state semantics
 - user-mode programs
- System.GC class
 - control methods
 - enumeration
 - properties
 - scenarios
- diagnostic methods
- notifications

T

- Task parallelism
 - class
 - decomposition
 - exceptions and cancellation
 - binary tree lookup
 - canceled tasks
 - exception-handling paradigm
 - Parallel.Invoke
 - partition method
 - QuickSort algorithm
 - recursive algorithms
 - Task.Run method
- Threads
 - CPU
 - Intel i7 system
 - mark phase
 - pools
 - abstraction
 - approach
 - concurrency profiler report
 - FIFO and LIFO
 - priorities
 - prime numbers
 - questions and answers
 - suspension
 - sweep phase
 - task parallel library
 - types
- Time profilers
 - commercial tools
 - customization
 - guidelines
 - profiling modes
 - sampling mode
 - Visual Studio
 - instrumentation profiling
 - sampling profiler
- Tracing garbage collection
 - mark phase
 - eager collection
 - false positives and negatives
 - f-reachable queue

- GC handles
- local roots
- performance implications
- referenced objects
- SOS.DLL
- static roots
- .NET CLR
- pinning operation
 - approaches
 - disadvantages
 - fragmentation
 - GC handle
 - managed and unmanaged code
 - .NET memory management model
 - practices
- sweep and compact phase
 - GC model
 - linear objects
 - moving objects
 - next object pointer and GC heap
 - shaded objects
- Triggers, GC
- Turing machine (TM)
 - automata theory
 - halting problem
 - C# method
 - countable
 - DoesHaltOnInput
 - Hepler method
 - undecidable problems
 - iterations
 - NP-complete problems
 - characterization
 - P and NP
 - problems
 - $O(n^2)$ algorithms
 - P/NP-complete
 - polynomial time
 - state diagram

U

- Union-find data structure
- Unsafe code. *See also* C++/CLI language extensions
- C# project
- lifetime management
- memory pooling scheme
- pinning and GCHandles
- P/Invoke
 - binding
 - code access security
 - C-style functions
 - DllImport attribute
 - FindFirstFile method
 - IntPtr
 - marshaler stubs
 - PInvoke.net and interop assistant
 - WIN32_FIND_DATA struct
- unmanaged memory
- widespread technologies

V

- Variable length encoding
- Virtual memory. *See* GC segments and virtual memory
- Visual Studio
 - allocation profiler
 - allocation view
 - function details view
 - JackCompiler.exe
 - object lifetime view
 - Summary view
 - instrumentation profiling
 - _CAP
 - CPU-bound
 - functions view
 - instrumented method
 - .NET Reflector

- sampling profiler
 - accurate technique
 - Call Tree view
 - details view
 - exclusive samples
 - functions view
 - inclusive samples
 - intervals
 - partial list
 - profiler report
 - program running



W, X, Y, Z

- Weak references, GC
 - application code
 - event
 - GC handle
 - long weak references
 - short weak references
 - strong reference
 - various scenarios
- Web analyzing tools
- Web application performance
 - ASP.NET applications
 - mechanisms
 - overview
 - per-compiling
 - pitfalls
 - process model
 - scaling out
 - server affinities
 - server-side output cache
 - tracing and debugging
 - view state
 - features
 - HTTP monitoring tools
 - IIS
 - output cache mechanisms
 - pool configuration
 - optimization, 329–330, CDNs
 - compression IIS components
 - HTTP cache headers
 - minification and bundling
 - server
 - asynchronous pages, modules and controllers
 - cache objects
- Windows 8
- Windows Communication Foundation (WCF)
 - asynchronous operation
 - bindings
 - caching
 - process model
 - throttling
 - ASP.NET applications
 - configuration
 - parameters
 - ServiceThrottling object
- Windows Performance Toolkit (WPT)
 - call stack aggregation
 - detailed stack frames
 - graph
 - I/O operations
 - raw report
 - sampling profilers
 - SDK installation directory
 - XPerf.exe
- Windows Runtime (WinRT)