

.NET性能优化



[.NET性能优化 下载链接1](#)

著者:[美]萨沙·戈德斯汀

出版者:人民邮电出版社

出版时间:2018-8

装帧:平装

isbn:9787115485861

本书详细解释了影响应用程序性能的Windows、CLR和物理硬件的内部结构，并为读者提供了衡量代码如何独立于外部因素执行操作的工具。书中提供了大量的C#代码示例和技巧，将帮助读者最大限度地提高算法和应用程序的性能，提高个人竞争优势，使用更低的成本获取更多的用户。

本书共11章，第1章和第2章关注性能的度量指标及性能评测；第3章和第4章则深入CLR内部，专注于类型与CLR垃圾回收的内部实现；第5~8章及第11章讨论.NET框架中的几个特定的方面，以及CLR提供的几种可用来进行性能优化的手段；第9章对复杂度理论和算法进行了简单的尝试；第10章则包含了一些独立话题，包括启动时间优化、异常及

.NET反射等。

本书适合已经拥有一定C#语言和.NET框架的编程基础，对相关概念较为熟悉的中高级程序员阅读学习。

本书可以帮助你充分挖掘算法和应用的潜力，避免常见陷阱，优化程序性能，发挥竞争优势，降低运行成本，提高用户满意度。

本书通过介绍大量的分析器和度量工具来指导读者进行性能度量，并讲解了操作系统和CLR是如何以意想不到的方式来影响程序性能的，同时还通过可工作的示例和真实案例来演示性能改进。

通过阅读本书，读者可以学到：

- * 找到并定位瓶颈，以获得zui大的性能效果；
- * 使用垃圾回收器高效管理内存；
- * 更深入地理解底层操作系统及其性能特点，从而更高效地编程；
- * 通过并行化、缓存、微优化和许多其他的技术来提升应用程序的性能。

本书包含大量C#代码示例和技巧，可以帮助读者充分利用程序中每一处可能的性能提升，如降低内存占用、一致化CPU使用，减少网络和磁盘的I/O操作等。本书将会改变你对.NET程序开发的思考方式。

作者介绍:

Sasha Goldshtein(萨沙·戈德斯汀) 是微软公司Visual C#方向的MVP，也是SELA Group的首席技术官 (CTO)。Sasha领导了SELA技术中心的性能与排错团队，并且在多个领域提供咨询服务，包括生产环境调试、应用程序性能排错及分布式架构。Sasha的经验主要集中在C#与C应用程序开发，以及高可伸缩性和高性能系统架构等方面。他经常在微软公司的相关会议上发表演讲，并举办了如“.NET性能”“.NET调试”“深入Windows”等多项培训课程。

Dima Zurbalev(迪马·祖巴列夫) 是SELA Group性能与调试团队紧急响应组的高级咨询师。Dima在性能优化和排错上帮助客户完成了许多几乎不可能完成的任务，引导他们深入理解CLR及Windows的内部细节。他的大部分开发经验围绕.NET与C基础项目进行，同时，他也在为CodePlex上的多个项目贡献代码。

Ido Flatow(伊多·弗莱托) 是微软公司Connected系统方向的MVP，也是SELA团队的高级架构师。他拥有超过15年的行业经验，目前是SELA的Windows Azure及Web领域的专家之一，专长为WCF、ASP.NET、Silverlight及IIS等技术。他是一名微软认证培训师 (Microsoft Certified Trainer, MCT)，也是微软官方WCF 4.0课程 (10263A) 的合作者。他同样也经常微软公司的相关会议上发表演讲。

目录: 第 1章 性能指标 1

- 1.1 性能目标 1
- 1.2 性能指标 3
- 1.3 小结 4
- 第2章 性能度量 5
 - 2.1 性能度量方式 5
 - 2.2 Windows内置工具 5
 - 2.2.1 性能计数器 6
 - 2.2.2 Windows事件追踪 10
 - 2.3 时间分析器 20
 - 2.3.1 Visual Studio采样分析器 20
 - 2.3.2 Visual Studio检测分析器 24
 - 2.3.3 时间分析器的gao级用法 25
 - 2.4 内存分配分析器 27
 - 2.4.1 Visual Studio内存分配分析器 27
 - 2.4.2 CLR分析器 29
 - 2.5 内存分析器 34
 - 2.5.1 ANTS Memory Profiler 34
 - 2.5.2 SciTech .NET Memory Profiler 36
 - 2.6 其他分析器 38
 - 2.6.1 数据库和数据访问分析工具 38
 - 2.6.2 并发分析工具 38
 - 2.6.3 I/O分析工具 40
 - 2.7 微基准测试 41
 - 2.7.1 设计不佳的微基准测试示例 41
 - 2.7.2 微基准测试指南 44
 - 2.8 小结 45
- 第3章 类型揭秘 47
 - 3.1 示例 47
 - 3.2 引用类型和值类型在语义上的区别 48
 - 3.3 存储、分配和销毁 48
 - 3.4 引用类型揭秘 50
 - 3.4.1 方法表 51
 - 3.4.2 调用引用类型实例的方法 55
 - 3.4.3 非虚方法的分发 56
 - 3.4.4 静态方法和接口方法的分发 58
 - 3.4.5 同步块索引和lock关键字 59
 - 3.5 值类型揭秘 63
 - 3.6 值类型的虚方法 65
 - 3.7 装箱 65
 - 3.7.1 避免在调用值类型的Equals方法时产生装箱 67
 - 3.7.2 GetHashCode方法 70
 - 3.8 使用值类型的zui佳实践 72
 - 3.9 小结 72
- 第4章 垃圾回收 73
 - 4.1 为什么需要垃圾回收 73
 - 4.1.1 空闲列表管理 73

- 4.1.2 引用计数垃圾回收 74
- 4.2 追踪垃圾回收 75
 - 4.2.1 标记阶段 76
 - 4.2.2 清理与压缩阶段 80
 - 4.2.3 固定 82
- 4.3 垃圾回收器的特征 83
 - 4.3.1 垃圾回收时暂停线程 83
 - 4.3.2 在垃圾回收时挂起线程 83
 - 4.3.3 工作站垃圾回收 85
 - 4.3.4 服务器垃圾回收 86
 - 4.3.5 切换垃圾回收特征 87
- 4.4 代 89
 - 4.4.1 “代”模型的假设 89
 - 4.4.2 .NET中“代”的实现 90
 - 4.4.3 大对象堆 93
 - 4.4.4 跨代引用 94
 - 4.4.5 后台垃圾回收 96
- 4.5 垃圾回收段和虚拟内存 97
- 4.6 终结化 100
 - 4.6.1 手动确定性终结化 100
 - 4.6.2 自动的非确定性终结化 100
 - 4.6.3 非确定性终结的缺点 102
 - 4.6.4 Dispose模式 104
- 4.7 弱引用 106
- 4.8 使用垃圾回收器 108
 - 4.8.1 System.GC类 108
 - 4.8.2 使用CLR宿主与垃圾回收器进行交互 111
 - 4.8.3 垃圾回收触发器 111
- 4.9 垃圾回收性能zui佳实践 112
 - 4.9.1 “代”模型 112
 - 4.9.2 固定 113
 - 4.9.3 终结化 114
 - 4.9.4 其他建议与zui佳实践 114
- 4.10 小结 117
- 第5章 集合和泛型 119
 - 5.1 泛型 119
 - 5.1.1 .NET泛型 121
 - 5.1.2 泛型约束 122
 - 5.1.3 CLR泛型的实现 125
 - 5.2 集合 131
 - 5.2.1 并发集合 132
 - 5.2.2 缓存 133
 - 5.3 自定义集合 137
 - 5.3.1 分离集（并查集） 137
 - 5.3.2 跳跃表 138
 - 5.3.3 一次性集合 139
 - 5.4 小结 141
- 第6章 并发和并行 142
 - 6.1 挑战与所得 142
 - 6.2 从线程到线程池，再到任务 143
 - 6.2.1 任务并行 148
 - 6.2.2 数据并行 153
 - 6.2.3 C# 5异步方法 156

- 6.2.4 TPL中的gao级模式 159
- 6.3 同步 160
 - 6.3.1 无锁代码 161
 - 6.3.2 Windows同步机制 165
 - 6.3.3 缓存 167
- 6.4 通用的GPU计算 168
 - 6.4.1 C++ AMP简介 169
 - 6.4.2 矩阵相乘 171
 - 6.4.3 多体仿真 171
 - 6.4.4 tile和共享内存 172
- 6.5 小结 175
- 第7章 网络、I/O和序列化 176
 - 7.1 I/O基本概念 176
 - 7.1.1 同步与异步I/O 176
 - 7.1.2 I/O完成端口 177
 - 7.1.3 .NET线程池 181
 - 7.1.4 内存复制 181
 - 7.2 分散-聚集I/O 182
 - 7.3 文件I/O 182
 - 7.3.1 缓存提示 183
 - 7.3.2 非缓存I/O 183
 - 7.4 网络I/O 184
 - 7.4.1 网络协议 184
 - 7.4.2 网络套接字 185
 - 7.5 数据序列化与反序列化 186
 - 7.5.1 序列化基准测试 187
 - 7.5.2 数据集 (DataSet) 序列化 189
 - 7.6 Windows通信基础类库 189
 - 7.6.1 限流 189
 - 7.6.2 处理模型 190
 - 7.6.3 缓存 191
 - 7.6.4 异步WCF客户端与服务器 191
 - 7.6.5 绑定 192
 - 7.7 小结 193
- 第8章 不安全的代码以及互操作 194
 - 8.1 不安全的代码 194
 - 8.1.1 对象固定与垃圾回收句柄 195
 - 8.1.2 生存期管理 196
 - 8.1.3 分配非托管内存 196
 - 8.1.4 内存池 197
 - 8.2 平台调用 198
 - 8.2.1 P/Invoke.net与P/Invoke Interop Assistant软件 199
 - 8.2.2 绑定 200
 - 8.2.3 列集器存根程序 201
 - 8.2.4 原生同构类型 204
 - 8.2.5 列集方向、值类型和引用类型的列集 205
 - 8.2.6 代码访问安全性 206
 - 8.3 COM互操作性 206
 - 8.3.1 生存期管理 207

- 8.3.2 单元列集 208
- 8.3.3 TLB导入与代码访问
安全性 209
- 8.3.4 无主互操作程序集
(NoPIA) 209
- 8.3.5 异常 210
- 8.4 C++/CLI语言扩展 211
 - 8.4.1 marshal_as辅助库 213
 - 8.4.2 IL代码与原生代码 214
- 8.5 Windows 8 WinRT互操作 214
- 8.6 互操作的zui佳实践 215
- 8.7 小结 215
- 第9章 算法优化 216
 - 9.1 复杂度的维度 216
 - 9.1.1 大O复杂度 216
 - 9.1.2 主定理 217
 - 9.1.3 图灵机与复杂度分类 218
 - 9.1.4 停机问题 219
 - 9.1.5 NP完全问题 221
 - 9.1.6 记忆与动态规划 221
 - 9.1.7 编辑距离 222
 - 9.1.8 每对顶点间的zui短路径 224
 - 9.2 近似算法 226
 - 9.2.1 旅行商问题 226
 - 9.2.2 zui大割 227
 - 9.3 概率算法 227
 - 9.3.1 概率zui大割 227
 - 9.3.2 费马质数测试 228
 - 9.4 索引与压缩 228
 - 9.4.1 变量的长度编码 228
 - 9.4.2 压缩索引 229
 - 9.5 小结 230
- 第 10章 性能模式 232
 - 10.1 JIT编译器优化 232
 - 10.1.1 标准的优化方法 232
 - 10.1.2 方法内联 233
 - 10.1.3 消除边界检查 234
 - 10.1.4 尾调用 236
 - 10.1.5 启动性能 238
 - 10.1.6 使用NGen进行JIT预
编译 239
 - 10.1.7 多核后台JIT编译 241
 - 10.2 关于启动性能的其他技巧 243
 - 10.2.1 将强命名程序集置于
GAC中 243
 - 10.2.2 防止本机镜像发生地址
重排 243
 - 10.2.3 减少程序集数目 244
 - 10.3 处理器相关的优化 245
 - 10.3.1 单指令多数数据流
(SIMD) 245
 - 10.3.2 指令级别并行 247
 - 10.4 异常 250
 - 10.5 反射 250

- 10.6 代码生成 251
 - 10.6.1 直接用源代码生成代码 251
 - 10.6.2 用动态轻量级代码生成技术 (LCG) 生成代码 253
- 10.7 小结 257
- 第 11 章 Web应用性能 258
 - 11.1 测试Web应用的性能 258
 - 11.1.1 Visual Studio Web性能测试和压力测试 259
 - 11.1.2 HTTP监控工具 260
 - 11.1.3 分析工具 260
 - 11.2 提高Web服务器的性能 261
 - 11.2.1 缓存公用对象 261
 - 11.2.2 使用异步页面、模块和控制器 262
 - 11.2.3 创建异步页面 263
 - 11.2.4 创建异步控制器 265
 - 11.3 ASP.NET环境调优 265
 - 11.3.1 关闭ASP.NET跟踪和调试 266
 - 11.3.2 关闭视图状态 267
 - 11.3.3 服务端输出缓存 268
 - 11.3.4 对ASP.NET应用程序进行预编译 269
 - 11.3.5 ASP.NET进程模型调优 270
 - 11.4 配置IIS 271
 - 11.4.1 输出缓存 271
 - 11.4.2 应用程序池配置 273
 - 11.5 网络优化 274
 - 11.5.1 使用HTTP缓存头 274
 - 11.5.2 启用IIS压缩 277
 - 11.5.3 精简与合并 279
 - 11.5.4 使用内容发布网络 (CDN) 280
 - 11.6 对ASP.NET应用程序进行扩容 (scaling) 281
 - 11.6.1 向外扩容 281
 - 11.6.2 ASP.NET扩容机制 282
 - 11.6.3 向外扩容的隐患 282
 - 11.7 小结 283

• • • • • [\(收起\)](#)

[.NET性能优化_下载链接1](#)

标签

.NET

计算机

性能优化

C

#技术

paperbook

m

Performance

评论

好看..

[.NET性能优化_下载链接1](#)

书评

[.NET性能优化_下载链接1](#)