<<C++应用程序性能优化>>

图书基本信息

书名：<<C++应用程序性能优化>>

13位ISBN编号：9787121106330

10位ISBN编号：7121106337

出版时间：2010-6

出版时间：冯宏华、徐莹、程远、 等 电子工业出版社 (2010-06出版)

作者：冯宏华 等 著

页数：308

前言

在计算机发展的早期阶段，硬件资源相对而言是非常昂贵的。
不论是CPU运行时间，还是内存容量，都给编程人员设置了很大的限制。
因此，当时程序对运行性能和内存空间占用的要求是非常严格的。
很多开发人员为了减少1%的CPU运行时间，为减少几十个，甚至几个字节而孜孜努力。
随着计算机技术的快速发展，硬件资源变得相对便宜。
因此有的观点认为在开发软件时，软件的性能优化将不再重要，硬件将解决性能问题。
但事实上，这种观点是相对片面的。
的确，硬件的发展解决了部分软件的性能问题。
但随着硬件计算能力的提高，人们对软件功能的要求也越来越高。
当今的软件功能越来越复杂，给用户的界面和操作体验也越来越智能和友好，这些需求带来的软件性能上的要求是硬件不能完全解决的。
很多实际的项目证明，如果在开发软件时不重视性能的优化，最终虽然实现了功能上的要求，但软件的运行效率低下，也不能给用户带来很好的效益。
因此，软件的性能优化是计算机软件开发过程中需要一直关注的重要因素。

<<C++应用程序性能优化>>

内容概要

《C++应用程序性能优化（第2版）》主要针对的是C++程序的性能优化，深入介绍C++程序性能优化的方法和实例。
全书由5篇组成，第1，2篇介绍C++语言的对象模型，该篇是优化C++程序的基础；第3篇主要针对如何优化C++程序的内存使用；第4篇介绍如何优化程序的启动性能；第5篇介绍了三类性能优化工具，即内存分析工具、性能分析工具和I/O检测工具，它们是测量程序性能的利器。
《C++应用程序性能优化（第2版）》适用于有一定C++程序开发经验的开发人员，也可以作为高校相关专业师生的参考书。

作者简介

冯宏华，清华大学计算机科学与技术系硕士。
IBM中国开发中心高级软件工程师。
2003年12月加入IBM中国开发中心，主要从事IBM产品的开发、性能优化等工作。
兴趣包括C/C++应用程序性能调优，Windows应用程序开发，Web应用程序开发等。
徐莹，山东大学计算机科学与技术系硕士。
2003年4月加入IBM中国开发中心，现任IBM中国开发中心开发经理，一直从事IBM软件产品在多个操作系统平台上的开发工作。
曾参与IBM产品在Windows和Linux平台上的性能优化工作，对C/C++编程语言和跨平台的大型软件系统的开发有较丰富的经验。
程远，北京大学计算机科学与技术系硕士。
IBM中国开发中心高级软件工程师。
2003年加入IBM中国开发中心，主要从事IBM Productivity Tools产品的开发、性能优化等工作。
兴趣包括C／C++编程语言，软件性能工程，Windows／Linux平台性能测试优化工具等。
汪磊，北京航空航天大学计算机科学与技术系硕士，目前是IBM中国软件开发中心高级软件工程师。
从2002年12月加入IBM中国开发中心至今一直从事旨在提高企业生产效率的应用软件开发。
兴趣包括C／C++应用程序的性能调优，Java应用程序的性能调优。

<<C++应用程序性能优化>>

书籍目录

第1篇 应用程序性能优化概述 第1章 应用程序性能优化概述 2 1.1 应用程序性能的定义 2 1.2 性能基准 3 1.2.1 基准负载 3 1.2.2 基准用例 4 1.2.3 性能基准的运行 5 1.2.4 性能基准结果 6 1.3 性能分析方法概述 7 1.4 性能优化方法概述 9 1.5 本章小结 10 第2篇 C++程序优化基础 第2章 C++对象模型 12 2.1 基本概念 12 2.1.1 程序使用内存区 12 2.1.2 全局/静态存储区及常量数据区 15 2.1.3 堆和栈 16 2.1.4 C++中的对象 18 2.2 对象的生命周期 18 2.3 C++对象的内存布局 23 2.3.1 简单对象 23 2.3.2 单继承 26 2.3.3 多继承 29 2.4 构造与析构 38 2.5 本章小结 40 第3章 C++语言特性中的性能分析 41 3.1 构造函数与析构函数 42 3.2 继承与虚拟函数 53 3.3 临时对象 61 3.4 内联函数 75 3.5 本章小结 83 第4章 常用数据结构的性能分析 84 4.1 常用数据结构性能分析 84 4.1.1 遍历 89 4.1.2 插入 91 4.1.3 删除 94 4.1.4 排序 96 4.1.5 查找 100 4.2 动态数组的实现及分析 102 4.2.1 动态数组简介 102 4.2.2 动态数组实践及分析 104 4.3 本章小结 110 第3篇 内存使用优化 第5章 操作系统的内存管理 112 5.1 Windows内存管理 112 5.1.1 使用虚拟内存 113 5.1.2 访问虚拟内存时的处理流程 115 5.1.3 虚拟地址到物理地址的映射 117 5.1.4 虚拟内存空间使用状态记录 120 5.1.5 进程工作集 121 5.1.6 Win32内存相关API 123 5.2 Linux内存管理机制 132 5.2.1 进程的内存布局 133 5.2.2 物理内存管理 135 5.2.3 虚拟内存管理 136 5.2.4 虚拟地址映射为物理地址 137 5.3 本章小结 138 第6章 动态内存管理 139 6.1 operator new/delete 139 6.2 自定义全局operator new/delete 144 6.3 自定义类operator new/delete 148 6.4 避免内存泄漏 151 6.5 智能指针 156 6.6 本章小结 166 第7章 内存池 167 7.1 自定义内存池性能优化的原理 167 7.1.1 默认内存管理函数的不足 167 7.1.2 内存池的定义和分类 168 7.1.3 内存池工作原理示例 168 7.2 一个内存池的实现实例 170 7.2.1 内部构造 170 7.2.2 总体机制 171 7.2.3 细节剖析 174 7.2.4 使用方法 183 7.2.5 性能比较 184 7.3 本章小结 184 第4篇 应用程序启动性能优化 第8章 动态链接与动态库 186 8.1 链接技术的发展 186 8.1.1 编译、链接和加载 187 8.1.2 静态链接与静态链接库 189 8.1.3 动态链接与动态库 195 8.2 Windows DLL，Dynamic Linked Library 196 8.2.1 DLL基础 196 8.2.2 DLL如何工作 200 8.2.3 关于DLL的杂项 208 8.3 Linux DSO 209 8.3.1 DSO与ELF 209 8.3.2 DSO如何工作 217 8.3.3 构建与使用DSO 223 8.4 本章小结 233 第9章 程序启动过程 234 9.1 Win32程序启动过程 234 9.2 Linux程序启动过程 238 9.3 影响程序启动性能的因素 239 9.3.1 源代码因素 240 9.3.2 动态链接库因素 241 9.3.3 配置文件/资源文件因素 247 9.3.4 其他因素 248 9.4 本章小结 250 第10章 程序启动性能优化 251 10.1 优化程序启动性能的步骤 251 10.2 测试程序启动性能的方法 252 10.3 优化可执行文件和库文件 255 10.3.1 减少动态链接库的数量 255 10.3.2 减小动态链接库尺寸 257 10.3.3 优化可执行文件和库文件中的代码布局 257 10.4 优化源代码 259 10.4.1 优化启动时读取的配置文件及帮助文件 259 10.4.2 预读频繁访问的文件 260 10.4.3 清除产生exception的代码 261 10.4.4 PreLoad 262 10.4.5 延迟初始化 262 10.4.6 多线程化启动 263 10.5 本章小结 264 第5篇 性能工具 第11章 内存分析工具IBM Rational Purify 266 11.1 Rational Purify工作原理 266 11.2 使用Rational Purify来发现内存泄漏 269 11.2.1 内存泄漏及其对应用程序性能的影响 269 11.2.2 用PerfMon来发现Windows系统中有严重后果的内存泄漏 270 11.2.3 用Rational Purify来定位内存泄漏 273 11.2.4 典型的内存泄漏错误 274 11.3 Rational Purify使用指南 277 11.4 Rational Purify实例分析 283 11.5 本章小结 287 第12章 性能分析工具IBM Rational Quantify 288 12.1 Rational Quantify工作原理 289 12.2 Rational Quantify使用指南 290 12.3 Rational Quantify实例分析 293 12.4 本章小结 297 第13章 实时IO监测工具FileMon 298 13.1 FileMon的工作原理 298 13.2 FileMon使用指南 301 13.3 使用FileMon解决问题 303 13.4 本章小结 306 参考文献 307

章节摘录

插图：当通过VirtualAlloc申请一块虚拟内存时，虚拟内存管理器是如何知道哪些内存块是自由的，可以用来满足此次内存请求呢？
即Win32虚拟内存如何维护和记录每一个进程的4GB虚拟内存地址空间的使用状态，如各个区域的状态、大小及起始地址呢？
上一节中，读者也许会认为可以通过遍历页目录和页表中的项值来收集虚拟内存空间的使用状态，但这样做首先有效率问题，因为每次申请内存都需要做一次搜索。
但这个方法不仅仅是因为效率有问题，而且还是行不通的，对预留的页来说，虚拟内存管理器并没有为之分配物理存储。
所以也就不会为其填写页表项，这时遍历页表无法分辨某块虚拟内存是自由还是预留的。
另外即使对提交页来说，遍历页表也无法得到完整的信息，正如5.1.1节中提到的Win32在虚拟内存管理时用到的主要策略demand-paging，即Win32虚拟内存管理器在程序没有实际访问某块内存前，总是假定这块内存不会被访问到，因此不会为这块内存做过多处理，包括不会为其分配真正的物理内存空间，甚至页表，即进程中用来完成虚拟地址到物理地址映射的页表的存储空间也是随需分配的。
Win32虚拟内存管理器使用另外一个数据结构来记录和维护每个进程的4 GB虚拟地址空间的使用及状态信息，这就是虚拟地址描述符树（Virtual Address Descriptor，VAD）。
每一个进程都有一个自己的’VAD集合，这个集合中的VAD被组织成一个自平衡二叉树，以提高查找的效率。
另外只有预留或者提交的内存块才会有VAD，自由的内存块没有VAD（因此不在VAD树结构中的虚拟地址块就是自由的）。
’VAD的组织如图5.5所示。
（1）当程序申请一块新内存时，虚拟内存管理器只需访问VAD树。
找到两个相邻VAD，只要小的VAD的上限与大的VAD的下限之间的差值满足所申请的内存块的大小需求，即可使用二者之间的虚拟内存。
（2）当第一次访问提交的内存时，虚拟内存管理器根据上一节描述的流程。
即总是假定该数据页已在物理内存中，并进行虚拟地址到物理地址的转换。
当找到相应的页目录项后发现该页目录项并没有指向一个合法的页表，它就会查找该进程的VAD树。

编辑推荐

“ 工欲善其事必先利其器 ”，《C++应用程序性能优化(第2版)》针对C++程序性能优化问题，深入剖析程序的内在本质(编程设计因素)和外在形式(系统结构因素)，展示应用程序优化的方法和实例。
第1篇 性能优化全局性介绍。
概述性能优化工作的基本概念、流程和方法论。
第2篇 C++程序优化基础。
介绍C++语言的对象模型、与性能有关的语言特性及数据结构的性能。
第3篇 C++程序内存使用的优化。
介绍如何在特定平台下进行内存优化，并深入介绍C++语言管理动态内存的机制和方法，以及内存池的实现。
第4篇 程序启动性能优化。
介绍动态库的基本知识，以及程序启动性能优化的具体方法。
介绍几类性能测量和分析的利器。
如内存分析工具、性能分第5篇 介绍几类性能测量和分析的利器。
如内存分析工具、性能分析工具、I／检测工具等。
系统介绍性能优化的工作流程，方法论，C++性能特性，分析工具等，并配合大量最佳实践及代码实例。

<<C++应用程序性能优化>>

版权说明