# JAVA程序性能优化

### 第1章 程序优化的目标

1.1 程序优劣的标准

1.2 可读性、可扩展性、性能的平衡

1.3 时间与空间的平衡

1.3.1 以空间换取时间

1.3.2 以时间换取空间

1.4 低成本与高性能

1.4.1 摩尔定律

1.4.2 PCServer集群替代小型机

1.5 性能优化的一般策略与方法

1.5.1 设计优化

1.5.2 代码优化

1.5.3 配置优化

1.5.4 外部条件的优化

### 第2章 了解JAVA虚拟机

2.1 JAVA虚拟机概述

2.2 运行时数据区

2.2.1 程序计数器

2.2.2 虚拟机栈

2.2.3 方法区

2.2.4 本地方法栈

2.2.5 常量池

2.2.6 堆

2.3 垃圾收集机制

2.3.1 垃圾收集简介

2.3.2 对象的生命周期

2.3.3 强引用、软引用、弱引用、幽灵引用

2.3.4 引用计数算法

2.3.5 根搜索算法

2.3.6 分代收集算法

2.4 虚拟机的选择

2.4.1 32位 or 64位

2.4.2 Client模式 or Server模式

### 第3章 寻找程序瓶颈

3.1 不要推测，要测量

3.1.1 JAVA程序常用性能指标

3.1.2 性能指标的一般检测方法

3.2 JConsole的使用

3.2.1 设置程序运行时可以被JConsolse连接分析

3.2.2 应用程序概述

3.2.3 内存

3.2.4 线程

3.2.5 类

3.2.6 VM摘要

3.2.7 MBean

3.3 开源工具VisualVM

3.3.1 VisualVM的安装与连接

3.3.2 安装VisualVM插件

3.3.3 监控应用程序概况

3.3.3 堆转储快照

3.3.4 性能分析

3.3.5 BTrace动态日志跟踪

3.3.6 OQL支持

3.4 内存泄漏分析

3.4.1 垃圾收集报告

3.4.2 垃圾收集报告分析

3.4.3 常见内存泄漏陷阱

### 第4章 选择合适的算法与数据结构

4.1 循环、递归与递推

4.2 问题分治

4.3 以映射表替代复杂算法

4.4 冗余数据设计

4.5 异常处理的滥用

4.6 String、StringBuffer与StringBuilder

4.7 ArrayList

4.8 Linkedlist

4.9 HashMap

### 第5章 缓存与对象池

5.1 内存和磁盘

5.1.1 计算机的多级存储

5.1.2 内存和磁盘的速度差异

5.2 操作系统缓存机制

5.2.1 虚拟内存

5.2.2 文件系统

5.2.3 Linux的页面缓存

5.3 缓存实现

5.3.1 如何降低磁盘IO负载

5.3.2 提高缓存命中率

5.3.3 分布式缓存

5.3.4 JAVA访问Redis

5.4 对象池

### 第6章 多线程

6.1 为什么用多线程

6.1.1 现代处理器的多核架构

6.1.2 提高CPU利用率

6.1.3 多进程与多线程

6.1.4 多线程的优势

6.2 JAVA内存模型

6.2.1 主内存与工作内存

6.2.2 原子性、可见性与有序性

6.2.3 volatile关键字

6.2.4 long和double的非原子性协定

6.2.5 先行发生原则

6.3 线程的同步协作

6.3.1 synchronized

6.3.2 wait、notify与notifyAll

6.3.3 Condition

6.3.4 volatile

6.3.5 ReentrantLock

6.3.6 ReadWriteLock

6.3.7 atomic包

6.3.8 CountDownLatch