1. Record Nr. UNICASUBO2761253

Titolo	The practice of cultural studies / Richard Johnson, Deborah Chambers, Parvati Raghuram and Estella Tincknell
Pubbl/distr/stampa	London ; Thousand Oaks ; New Delhi, : Sage, 2004
ISBN	0761960996 0761961003
Descrizione fisica	XI, 300 p. : ill. ; 25 cm.
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia

2. Record Nr. UNINA9910967869403321

Autore	Hirt Marcus
Titolo	Oracle JRockit : the definitive guide : develop and manage robust Java applications with Oracle's high-performance Java Virtual Machine / / Marcus Hirt, Marcus Lagergren
Pubbl/distr/stampa	Birmingham, U.K., : Packt Enterprise, 2010
ISBN	9786612624094 9781282624092 1282624091 9781847198075 1847198074
Edizione	[1st edition]
Descrizione fisica	1 online resource (588 p.)
Altri autori (Persone)	LagergrenMarcus
Disciplina	005.1 005.133
Soggetti	Application software - Development Java (Computer program language)
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Includes index.
Nota di bibliografia	Includes bibliographical references (p. [493]-501) and index.

Nota di contenuto

Cover; Copyright; Credits; Foreword; About the Authors; About the Reviewers; Table of Contents; Preface; Chapter 1: Getting Started; Obtaining the JRockit JVM; Migrating to JRockit; Command-line options; System properties; Standardized options; Non-standard options; Changes in behavior; A note on JRockit versioning; Getting help; Summary; Chapter 2: Adaptive Code Generation; Platform independence; The Java Virtual Machine; Stack machine; Bytecode format; Operations and operands; The constant pool; Code generation strategies; Pure bytecode interpretation; Static compilation
Total JIT compilationMixed mode interpretation; Adaptive code generation; Determining ""hotness""; Invocation counters; Software-based thread sampling; Hardware-based sampling; Optimizing a changing program; Inside the JIT compiler; Working with bytecode; Bytecode obfuscation; Bytecode ""optimizers""; Abstract syntax trees; Where to optimize; The JRockit code pipeline; Why JRockit has no bytecode interpreter; Bootstrapping; Runtime code generation; Trampolines; Code generation requests; Optimization requests; On-stack replacement; Bookkeeping; A walkthrough of method generation in JRockit
The JRockit IR formatJIT compilation; Generating optimized code; Controlling code generation in JRockit; Command-line flags and directive files; Command-line flags; Directive files; Summary; Chapter 3: Adaptive Memory Management; The concept of automatic memory management; Adaptive memory management; Advantages of automatic memory management; Disadvantages of automatic memory management; Fundamental heap management; Allocating and releasing objects; Fragmentation and compaction; Garbage collection algorithms; Reference counting; Tracing techniques; Mark and sweep; Stop and copy
Stopping the worldConservative versus exact collectors; Livemaps; Generational garbage collection; Multi generation nurseries; Write barriers; Throughput versus low latency; Optimizing for throughput; Optimizing for low latency; Garbage collection in JRockit; Old collections; Nursery collections; Permanent generations; Compaction; Speeding it up and making it scale; Thread local allocation; Larger heaps; 32-Bits and the 4-GB Barrier; The 64-bit world; Cache friendliness; Prefetching; Data placement; NUMA; Large pages; Adaptability; Near-real-time garbage collection; Hard and soft real-time
JRockit Real TimeDoes the soft real-time approach work?; How does it work?; The Java memory API; Finalizers; References; Weak references; Soft references; Phantom references; Differences in JVM behavior; Pitfalls and false optimizations; Java is not C++; Controlling JRockit memory management; Basic switches; Outputting GC data; Set initial and maximum heap size; Controlling what to optimize for; Specifying a garbage collection strategy; Compressed references; Advanced switches; Summary; Chapter 4: Threads and Synchronization; Fundamental concepts; Hard to debug; Difficult to optimize
Latency analysis

Sommario/riassunto

Develop and manage robust Java applications with Oracle's high-performance JRockit Java Virtual Machine with this book and eBook