

Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| V | Vorwort | 11 |
| V.1 | Zielgruppe | 11 |
| V.2 | Über dieses Buch | 12 |
| V.3 | Intel®-Produkte zur Softwareentwicklung | 13 |
| V.4 | Danksagung | 13 |
| 1 | Einführung in die Multicore-Architektur | 15 |
| 1.1 | Gründe für Gleichzeitigkeit auf Softwareebene | 16 |
| 1.2 | Parallele Rechnerplattformen | 18 |
| | Parallelverarbeitung in Mikroprozessoren | 19 |
| | Unterschiede zwischen Multicore-Architektur und Hyperthreading-Technologie | 22 |
| | Multithreading auf Singlecore- und Multicore-Plattformen | 23 |
| 1.3 | Was ist Performance? | 25 |
| | Das Amdahlsche Gesetz | 25 |
| | Gustafsons Gesetz | 28 |
| 1.4 | Zusammenfassung | 30 |
| 2 | Systemüberblick: Threading | 31 |
| 2.1 | Threads definieren | 31 |
| 2.2 | Threads aus der Sicht des Systems | 32 |
| | Threading oberhalb der Betriebssystemebene | 32 |
| | Threads innerhalb des Betriebssystems | 35 |
| | Threads innerhalb der Hardware | 38 |
| 2.3 | Was passiert, wenn ein Thread erzeugt wird? | 39 |
| 2.4 | Modelle der Anwendungsprogrammierung und Threading | 41 |
| 2.5 | Virtuelle Umgebung: Virtuelle Maschinen und Plattformen | 41 |
| | Laufzeitvirtualisierung | 41 |
| | Systemvirtualisierung | 41 |
| 2.6 | Zusammenfassung | 44 |
| 3 | Die grundlegenden Konzepte der parallelen Programmierung | 45 |
| 3.1 | Design für Threads | 45 |
| | Aufgabenzerlegung | 46 |
| | Datenzerlegung | 46 |
| | Datenflusszerlegung | 47 |
| | Bedeutung der verschiedenen Zerlegungsarten | 48 |

| | | |
|----------|--|-----------|
| 3.2 | Herausforderungen | 49 |
| 3.3 | Patterns der parallelen Programmierung | 49 |
| 3.4 | Ein Fallbeispiel: Fehlerverteilung | 51 |
| | Analyse des Fehlerverteilungsalgorithmus | 54 |
| | Ein alternativer Ansatz: Parallele Fehlerverteilung | 54 |
| | Andere Alternativen | 56 |
| 3.5 | Zusammenfassung | 57 |
| 4 | Threading und parallele Programmierkonstrukte | 59 |
| 4.1 | Synchronisierung | 59 |
| 4.2 | Kritische Abschnitte | 61 |
| 4.3 | Deadlocks | 62 |
| 4.4 | Synchronisierungsprimitive | 63 |
| | Semaphore | 63 |
| | Sperren | 66 |
| | Bedingungsvariablen | 68 |
| 4.5 | Nachrichten | 70 |
| 4.6 | Ablaufsteuerungsbasierte Konzepte | 72 |
| | Fence | 72 |
| | Barrier | 73 |
| | Implementierungsabhängige Threading Features | 74 |
| 4.7 | Zusammenfassung | 75 |
| 5 | Threading-APIs | 77 |
| 5.1 | Threading-APIs für Microsoft Windows | 77 |
| | Win32/MFC Thread-APIs | 77 |
| 5.2 | Threading APIs für das .NET Framework von Microsoft | 103 |
| | Threads erzeugen | 103 |
| | Threads verwalten | 105 |
| | Thread Pools | 108 |
| | Thread-Synchronisierung | 112 |
| 5.3 | POSIX-Threads | 114 |
| | Threads erzeugen | 115 |
| | Threads verwalten | 116 |
| | Thread-Synchronisierung | 117 |
| | Signalisieren | 118 |
| | Kompilieren und Verlinken | 125 |
| 5.4 | Zusammenfassung | 125 |

| | | |
|----------|---|------------|
| 6 | OpenMP: Eine portierbare Lösung für Threading | 127 |
| 6.1 | Aufgaben beim Ausführen einer Schleife in Threads | 128 |
| | Iterationsübergreifende Abhängigkeit | 129 |
| | Data Race Conditions | 131 |
| | Gemeinsame und private Daten verwalten | 132 |
| | Scheduling und Partitionierung von Schleifen | 133 |
| | Effektive Verwendung der reduction-Klauseln | 137 |
| 6.2 | Threading Overhead minimieren | 139 |
| | Worksharing-Konstrukte | 140 |
| 6.3 | Performanceorientierte Programmierung | 141 |
| | Barrier und Nowait | 141 |
| | Singlethread- und Multithread-Ausführung miteinander verzahnen | 143 |
| | Daten kopieren | 144 |
| | Aktualisierungen von gemeinsamen Variablen schützen | 146 |
| | Taskqueuing – die OpenMP-Erweiterung von Intel | 148 |
| 6.4 | OpenMP-Bibliotheksfunktionen | 150 |
| 6.5 | OpenMP-Umgebungsvariablen | 151 |
| 6.6 | Kompilierung | 151 |
| 6.7 | Debugging | 153 |
| 6.8 | Performance | 155 |
| 6.9 | Zusammenfassung | 156 |
| 7 | Lösungen zu den häufigsten Problemen der parallelen Programmierung | 157 |
| 7.1 | Zu viele Threads | 157 |
| 7.2 | Data Races, Deadlocks und Livelocks | 160 |
| | Deadlocks | 162 |
| 7.3 | Heavily Contended Locks | 165 |
| | Prioritätsinversion | 166 |
| | Lösungen für Heavily Contended Locks | 167 |
| 7.4 | Nicht-blockierende Algorithmen | 169 |
| | Das ABA-Problem | 171 |
| | Cache-Zeilen-Pingpong | 173 |
| | Probleme mit der Speicherrückforderung | 173 |
| | Empfehlungen | 174 |
| 7.5 | Threadsichere Funktionen und Bibliotheken | 174 |
| 7.6 | Speicherprobleme | 176 |
| | Bandbreite | 176 |
| | Im Cache arbeiten | 177 |
| | Speicherkonflikte | 179 |

| | | |
|-----------|---|------------|
| 7.7 | Cachebezogene Probleme | 181 |
| | False Sharing | 182 |
| | Speicherkonsistenz | 184 |
| | Aktuelle IA-32-Architektur | 185 |
| | Itanium®-Architektur | 187 |
| | Höhere Sprachen | 190 |
| 7.8 | Pipeline Stalls auf IA-32-Prozessoren vermeiden | 191 |
| 7.9 | Datenorganisation für High Performance | 191 |
| 7.10 | Zusammenfassung | 193 |
| 8 | Debugging-Techniken für Multithread-Code | 195 |
| 8.1 | Allgemeine Debugging-Techniken | 195 |
| | Debug-freundliches Design | 195 |
| | Anwendungen erweitern – mit Trace Buffers | 198 |
| 8.2 | Multithread-Anwendungen unter Windows debuggen | 202 |
| | Das Threads-Fenster | 202 |
| | Ablaufverfolgungspunkte (Tracepoints) | 203 |
| | Haltepunktfilter | 204 |
| | Thread-Namen festlegen | 204 |
| | Ein Beispiel | 205 |
| 8.3 | Multithread Debugging mit GDB | 209 |
| | Benachrichtigung bei Thread-Erzeugung | 209 |
| | Auflistung aller Threads in der Anwendung | 209 |
| | Threadspezifische Haltepunkte setzen | 210 |
| | Zwischen Threads wechseln | 211 |
| | Einen Befehl auf eine Gruppe von Threads anwenden | 211 |
| 8.4 | Zusammenfassung | 212 |
| 9 | Grundlagen der Singlecore-Prozessoren | 213 |
| 9.1 | Grundlagen der Prozessorarchitektur | 213 |
| 9.2 | Superskalare und EPIC-Architektur im Vergleich | 219 |
| 9.3 | Zusammenfassung | 221 |
| 10 | Threading auf Intel®- Multicore-Prozessoren | 223 |
| 10.1 | Hardware Threading | 223 |
| | Threading bei Intel | 227 |
| 10.2 | Hyperthreading-Technologie | 227 |
| | Der Unterschied zwischen Multiprozessor- und Hyperthreading-Technologie | 229 |
| | Architektur der HT-Technologie | 229 |
| 10.3 | Multicore-Prozessoren | 231 |
| | Architekturdetails | 231 |
| | Vergleich von Multiprozessoren und Multicore-Prozessoren | 234 |
| | Multicore für Itanium®-Architektur | 235 |

| | | |
|-----------|---|------------|
| 10.4 | Interaktion zwischen mehreren Prozessoren | 238 |
| | Interprozessorkommunikation und Multithread-Programmierung | 238 |
| 10.5 | Energieverbrauch | 240 |
| | Energiemessverfahren | 241 |
| | Stromverbrauch senken | 242 |
| | Was kommt nach der Multicore-Prozessorarchitektur? | 243 |
| 10.6 | Zusammenfassung | 245 |
| 11 | Intel®-Produkte für die Softwareentwicklung | 247 |
| 11.1 | Überblick | 247 |
| | Untersuchen | 247 |
| | Erstellen/Ausdrücken | 248 |
| | Debuggen | 248 |
| | Tuning | 249 |
| 11.2 | Der Intel® Thread Checker | 249 |
| | Funktionsweise | 250 |
| | Hinweise für die Nutzung | 251 |
| | Intel® Thread Checker im Zusammenspiel mit OpenMP | 252 |
| 11.3 | Intel-Compiler | 253 |
| | OpenMP | 253 |
| | Softwarebasierte spekulative Vorausberechnung | 257 |
| | Compileroptimierung und Cache-Optimierung | 257 |
| 11.4 | Intel® Debugger | 258 |
| 11.5 | Intel-Bibliotheken | 259 |
| | Intel® Math Kernel Library | 259 |
| | Intel® Integrated Performance Primitives | 260 |
| | Parallele Programme und die Probleme beim Einsatz paralleler Bibliotheken | 260 |
| | Die Zukunft | 261 |
| | Intel®-Threading-Bausteine | 261 |
| 11.6 | Intel® VTune™ Performance Analyzer | 261 |
| | Hotspots finden | 262 |
| | Mit Call Graph einen Threading-Punkt finden | 263 |
| | Lastenausgleich überprüfen | 264 |
| 11.7 | Intel® Thread Profiler | 265 |
| 11.8 | MPI-Programmierung | 266 |
| | Intels MPI-Unterstützung | 266 |
| 11.9 | Zusammenfassung | 270 |
| A | Glossar | 271 |
| | Stichwortverzeichnis | 283 |