

Index

- Agile software, software development process, 83–84
- Amdahl's law, performance, and scalability testing, 97–99
- Application binary interface (ABI), application programming interfaces (APIs) contrasted, 45
- Application programming interface (API), 44–47
 - generally, 44–45
 - Google, 46–47
 - Java, 45–46
 - scalable performance, 2
 - Windows, 45
- Application programming interface (API) profiling, 249–338. *See also* Java application programming interfaces (APIs); Performance and scalability testing
 - defense lines, 252–253
 - defined, 251–252
 - execution stack, 253–254
 - overview, 249
 - perfBasic*, 254–260
 - generally, 254–255
 - logging format, 255–256
 - log parser, 256–258
 - performance maps, 258–260
 - summarization file, 260
- Application programming interface (API) profiling case study, 303–338
 - combo logs, 325–333
 - client side performance map, 325–327
 - server side performance map, 327–334
 - custom logs, 320–325
 - adapter, 320–321
 - performance map generation, 321–325
 - enabling, 304–312
 - apf.properties* file settings, 306–308
 - best coding practices, 311–312
 - generally, 304–305
 - mechanism of populating log entry, 305
 - non-Java programs, 312
 - parsing workflow, 308–310
 - source and target projects, 306
 - verification of profiling-enabled source code, 310–311

- Application programming interface (API)
 - profiling case study (*Continued*)
 - overview, 303–304
 - performance and scalability problems, 333–337
 - analysis, 336–337
 - baseline, 333–335
 - optimization, 335–336
 - standard logs, 313–320
 - data generation, 313–314
 - parsing, 314–316
 - performance map analysis, 319–320
 - performance map generation, 316–319
- Application programming interface (API)
 - profiling enablement, 263–280. *See also PerfBasic* (API profiling)
 - begin line, 272–274
 - comments processing, 271–272
 - enabling profiling, 267–270
 - end line, 275–276
 - files processing, 266–267
 - global parameters, 265
 - inner classes processing, 270–271
 - main logic, 266
 - main method processing, 276–277
 - overview, 263
 - return statements processing, 274–275
 - structure, 264–265
 - test program, 277–279
- Application programming interface (API)
 - profiling implementation, 281–301
 - adapter class, 300
 - analyzer class, 299–300
 - CallRecord* class, 294
 - driver program, 286–289
 - global parameters, 289–291
 - graphics tool:
 - dot* and *neato*, 281–284
 - ILOG, 284–285
 - resolution, 286
 - Link class, 293–294
 - logReader*, 291–292
 - logWriter*, 292
 - Node class, 292–293
 - overview, 286–287
 - parser class, 294–298
 - utility* class, 294
 - xmlProcessor* class, 298–299
- Application software, categorization, 54–55
- Aristotle, 503
- Array processing, optimization, and tuning
 - software application (queuing theory), 223–226
- Babbage, Charles, 3, 6
- Backus, John, 8
- Balanced queuing system, optimization, and tuning
 - software application (queuing theory), 240–244
- Batch jobs:
 - networked queuing systems, 170–171
 - scalability testing, 75–82
 - software performance testing, 86–95
- BEA Systems, 188–191
- Begin line, application programming interface (API) profiling
 - enablement, 272–274
- Benchmarking, performance benchmarking
 - testing, 74
- Bifurcating, 241
- BIOS (Basic Input/Output System),
 - systems software categorization, 53
- Bottlenecks:
 - central processing unit (CPU), *perfmon*, 119–121
 - disk I/O bottlenecks diagnosis, *perfmon*, 121–124
 - networked queuing systems, 170–171
 - Perfmon*, multithreading, 51–52
 - service-oriented architecture (SOA)
 - application (queuing theory), test results, 197–200
 - Task Manager, 125–128
 - Von Neuman, John, 8
- Business software, application software
 - categorization, 54
- Cache memory:
 - optimization and tuning software
 - application (queuing theory), 226–227
 - storage and hierarchy, 22–23
- Carlyle, Thomas, 249
- Central processing unit (CPU):
 - bottlenecks diagnosis, *perfmon*, system performance counters, 119–121
 - hyperthreading, 11–13
 - Perfmon*, multithreading, 48–52

- performance and scalability testing
 - factors, 100–103
 - von Neumann machine, 7–8, 18
- Chip, defined, 20
- Chipset, defined, 20
- Chipsets, Intel Core microarchitecture, 20–21
- C language, 42
- C++ language, 42, 253–254
- Client/server architecture, enterprise software, 57–59
- Closed model ($M/M/m/N/N$), networked queuing systems, 162–166
- COBOL, 42
- Componentry, enterprise software, 61
- Computer technology. *See also* Intel Core microarchitecture; Intel machine
 - advances in, 5–6, 18
 - Intel Core microarchitecture, 13–17
 - Intel machine, 9–17
 - sizing hardware, 35–37
 - Sun machine, 17–18
 - Turing machine, 6–7
 - von Neumann machine, 7–8, 18
 - Zuse machine, 8
- Continuous distribution and distribution density function, 145
- Continuous random variable, 145
- Covering index, optimization, and tuning software application (queuing theory), 228–229
- Cursor-sharing, service demand reduction, optimization and tuning software application (queuing theory), 229–231
- Database-centric application software categorization, 55
- Database deadlocks, performance, and scalability testing factors, 110
- Database double buffering, optimization, and tuning software application (queuing theory), 235–240
- Database statistics, performance, and scalability testing factors, 107–108
- Data latency reduction, optimization, and tuning software application (queuing theory), 232–233
- Da Vinci, Leonardo, 135
- Defense lines, application programming interface (API) profiling, 252–253
- Deterministic process, probability theory, 146
- Device drivers, systems software categorization, 53–54
- Discrete distribution and probability distribution series, 144
- Discrete random variable, probability theory, 144
- Disk I/O bottlenecks diagnosis, *perfmon*, system performance counters, 121–124
- Distributed mass function, probability theory, 144–145
- Distribution density function:
 - continuous distribution, 145
 - probability theory, 144–145
- Distribution functions, probability theory, 143–145
- Edison, Thomas Alva, 263
- Electronic Discrete Variable Automatic Computer (EDVAC, Von Neumann machine), 7–8
- End line, application programming interface (API) profiling enablement, 275–276
- Enterprise software, 55–63
 - client/server architecture, 58
 - componentry, 61
 - defined, 55–57
 - monolithic architecture, 57
 - multithreading, 47
 - N-tier architecture, 60
 - service-oriented architecture, 61
 - three-tier architecture, 59
- Entertainment software, application software categorization, 53
- Exponential distribution function:
 - memoryless property of, 361–363
 - probability theory applications, 146–151
- Extraneous logic elimination, service demand reduction, optimization, and tuning software application (queuing theory), 231–232
- Extreme programming, software development process, 84–86

- Feedback, networked queuing systems, 159
- Finite response time, networked queuing systems, 166–168
- FORTTRAN, 42
- Galileo Galilei, 135
- Gates, Bill, 42
- Genealogy, networked queuing systems, 171–172
- General process, probability theory, 146
- Gerrold, David, 361
- Google application programming interfaces (APIs), 46–47
- Hard disks, memory and storage hierarchy, 23
- Hardware, performance and scalability testing factors, 100–103
- Hardware platform, 5–37. *See also* Intel Core microarchitecture; Intel machine
 - Intel Core microarchitecture, 13–17
 - Intel machine, 9–17
 - scalable performance, 3
 - sizing hardware, 35–36
 - Sun machine, 17–18
 - Turing machine, 6–7
 - von Neumann machine, 7–8, 18
 - Zuse machine, 8
- Hardware principle, software performance data principles, 129–130
- Hierarchy, memory, and storage, 22–23
- Hyperthreading, Intel machine, 9–13
- Institute for Advanced Studies (IAS, Princeton University), 18
- Intel Core microarchitecture, 13–17
 - chipsets, 20–21
 - motherboards, 19–20
 - networking, 27–28
 - processors, 18–19
 - RAIDs, 23–27
 - storage, 21–23
 - Turing model, 30–35
- Intel machine, 9–17
 - historical perspective, 9
 - hyperthreading, 9–13
 - multicore architecture, 13–16
 - system monitoring tools, 17
- Internal application software categorization, 55
- Java application programming interfaces (APIs), 45–46, 253–254. *See also* Application programming interface (API) profiling
- Java virtual machine (JVM), 43–44
- Jackson’s Theorem, 156
- Kendall notation:
 - networked queuing systems, 152
- Languages, software stack, 42–44, 253
- Leonardo da Vinci, 135
- Licensing, performance, and scalability testing factors, 110
- Linux platforms, system performance counters, 128–129
- Little’s law, networked queuing systems, 154–155
- Logging format, *perfBasic*, API profiling, 255–256
- Log parser, API profiling, 256–258
- Main memory, memory, and storage hierarchy, 23
- Markov process:
 - memoryless property, 344
 - probability theory applications, 144–146
- Mathematical symbols, queuing theory, 142
- Media software, application software categorization, 54
- Medical records (MedRec) application:
 - queuing theory, 188–191
 - test results, 191–198
- Memory, storage, and hierarchy, 22–23
- Memory leaks:
 - Turing model, 30–35
 - perfmon*, system performance counters, 118–119
- Memoryless property, of exponential distribution, 361
- Method begin line, application programming interface (API) profiling enablement, 272–274
- Method end line, application programming interface (API) profiling enablement, 275–276

- Microsoft Windows. *See* Windows
- Middleware software, categorization, 55
- $M/M/m/N/N$ model (closed), networked queuing systems, 162–166
- Monolithic architecture, enterprise software, 57
- Moore, Gordon E., 5–6
- Moore’s law, 5–6
- Motherboards, Intel Core microarchitecture, 19–20
- MPLS, 233
- Multicore architecture, Intel machine, 13–16
- Multiple parallel queues, single-queue multiple servers versus, 160–162
- Multithreading, software platform, 47–53
- Networked queuing systems, 153–172
 - feedback, 156
 - finite response time, 166–169
 - genealogy, 171–172
 - generally, 153
 - Little’s law, 154
 - $M/M/m/N/N$ model (closed), 162–166
 - multiple parallel queues versus single-queue multiple servers, 160–162
 - open model ($M/M/I$), 155–159
 - open model validity, 169
 - system bottlenecks, 170
 - utilization, service time, and response time (Triad II), 159
- Networking, Intel Core microarchitecture, 27–28
- Newton, Isaac, 303
- Noninteractive batch jobs, performance metrics, 1
- N-tier architecture, enterprise software, 60–61
- Online transaction processing (OLTP):
 - hyperthreading, 9
 - performance metrics, 87
 - queuing systems, 153, 158–159
 - (*See also* Queuing theory)
 - scalability testing, 75–82
 - software performance testing, 86–95
- Open model ($M/M/I$):
 - networked queuing systems, 155–159
 - validity of, 169–170
- Operating system (OS):
 - performance and scalability testing factors, 103–107
 - systems software categorization, 55
- Optimization and tuning software application (queuing theory), 205–245
 - balanced queuing system, 240–244
 - overview, 205–207
 - performance and scalability, 208–220
 - application, 215–220
 - factor isolation, 208–215
 - problem characterization, 207–208
 - techniques, 220–240
 - array processing, 223–226
 - caching, 226–228
 - database double buffering, 235–240
 - data latency reduction, 232–233
 - generally, 220
 - service demand reduction, 228–231
 - covering index, 228
 - cursor-sharing, 229–231
 - extraneous logic elimination, 231–232
 - wait events and service demands, 221–223
- Optimization testing, performance optimization and tuning testing, 70–74
- PerfBasic* (API profiling), 254–260.
 - See also* Application programming interface (API) profiling case study; Application programming interface (API) profiling enablement; Application programming interface (API) profiling implementation
 - generally, 254–255
 - logging format, 255
 - log parser, 256–258
 - performance maps, 258–260
 - summarization file, 260
- Perfmon*:
 - CPU bottlenecks diagnosis, system performance counters, 111–128
 - disk I/O bottlenecks diagnosis, system performance counters, 121–125
 - memory leak diagnosis, system performance counters, 118–119
 - multithreading, 47–53

- Performance:
 - scalability contrasted, 1–2 (*See also* Scalable performance)
 - software platform, 42
- Performance and scalability testing, 67–135. *See also* Application programming interface (API) profiling
- Amdahl's law, 97–99
- application programming interface (API) profiling case study, 333–337
- factors in, 99–111
 - database deadlocks, 110
 - database statistics, 107
 - generally, 99
 - hardware, 100
 - licensing, 110
 - operating system, 103
 - SQL server parameterization, 108
- optimization and tuning software
 - application (queuing theory), 205–240
 - factor isolation, 208
 - problem characterization, 207
- overview, 65–68
- performance benchmarking testing, 75
- performance optimization and tuning testing, 70–75
- performance regression testing, 68
- performance testing merits, 82
- QA testing, performance testing compared, 82
- scalability testing, 75
- software development process, 83–86
 - agile software, 83
 - extreme programming, 84
- software performance, 86–95
 - batch jobs, 92
 - generally, 86–95
 - online transaction processing (OLTP), 87
- software performance data principles, 129–131
- software performance measurements, stochastic nature of, 95
- system performance counters, 111–129
 - generally, 111–112
 - perfmon*:
 - CPU bottlenecks diagnosis, 119
 - disk I/O bottlenecks diagnosis, 121
 - memory leak diagnosis, 118
 - Task Manager, system bottlenecks diagnosis, 125
 - UNIX/Linux platforms, 128
 - Windows performance console, 112
- test types, 68
- Performance benchmarking testing, 75
- Performance maps (API profiling), 258, 316–327
 - client side, 325
 - server side, 327
- Performance optimization and tuning testing, 70
- Performance regression testing, 68
- Performance testing:
 - merits of, 82
 - QA testing compared, 82
- Platform principle, software performance data principles, 129
- Plato, 137, 303
- Poisson distribution:
 - memoryless property, 361–363
 - probability theory applications, 148
- Probability distribution series and discrete distribution, 144
- Probability theory, 142–145
 - continuous distribution and distribution density function, 144–145
 - discrete distribution and probability distribution series, 144
 - generally, 142–143
 - random variables and distribution functions, 143–144
- Probability theory applications, 145–152
 - exponential distribution function, 150
 - generally, 145–146
 - Kendall notation, 152
 - Markov process, 146
 - nodes versus systems, 152
 - Poisson distribution, 148
- Process, thread contrasted, 47
- Processors, Intel Core microarchitecture, 13–16
- Product engineering software, application software categorization, 54
- Programming languages, software stack, 42, 253

- QA testing, performance testing
 - compared, 82
- Quality principle, software performance
 - data principles, 131
- Quantitativeness, scalable performance, 6
- Queue length, queuing systems, 154
- Queuing nodes, queuing systems contrasted,
 - probability theory applications, 152
- Queuing systems, queuing nodes contrasted,
 - probability theory applications, 152
- Queuing theory, 135–177
 - concepts in, 139–141
 - mathematical symbols in, 142
 - medical records (MedRec) application, 188
 - networked queuing systems, 153
 - feedback, 159
 - finite response time, 166
 - genealogy, 171
 - generally, 145
 - Little's law, 154
 - $M/M/m/N/N$ model (closed), 162
 - multiple parallel queues versus single-queue multiple servers, 160
 - open model ($M/M/I$), 155
 - open model validity, 169
 - system bottlenecks, 170
 - utilization, service time, and response time (Triad II), 159
 - optimization and tuning software
 - application, 205–249
 - overview, 205–207
 - performance and scalability, 207–220
 - application, 215–220
 - factor isolation, 208–215
 - problem characterization, 207–208
 - optimization and tuning software
 - application techniques, 220–235
 - array processing, 223
 - balanced queuing system, 240
 - caching, 226
 - database double buffering, 235
 - data latency reduction, 232
 - generally, 220
 - service demand reduction, 228–232
 - covering index, 228
 - cursor-sharing, 229
 - extraneous logic elimination, 231
 - wait events and service demands, 221
 - overview, 139
 - performance and scalability testing, 137–138
 - perspective on, 135
 - probability theory, 143
 - continuous distribution and
 - distribution density function, 145
 - discrete distribution and probability
 - distribution series, 144
 - generally, 143
 - random variables and distribution
 - functions, 144
 - probability theory applications, 145–153
 - exponential distribution function, 150
 - generally, 145
 - Kendall notation, 152
 - Markov process, 146
 - nodes versus systems, 152
 - Poisson distribution, 148
 - service-oriented architecture (SOA)
 - application, 177–201
 - analytical model, 181
 - demand, 183
 - database server, 186
 - data storage, 187
 - generally, 183
 - network latency, 185
 - object creation, 184
 - XML SOAP serialization/deserialization, 184
 - XML Web service provider, 186
 - model/measurement comparisons, 198
 - overview, 177
 - test results, 191
 - validity, 200
 - XML Web services, 179
- RAIDS. *See* Redundant arrays of inexpensive disks (RAIDs)
- Random variables, probability theory, 143
- Reality principle, software performance
 - data principles, 130
- Redundant arrays of inexpensive disks (RAIDs):
 - Intel Core microarchitecture, 13–17
 - Perfmon*, multithreading, 47

- Registers, memory and storage
 - hierarchy, 22
- Regression tests, performance regression testing, 68
- Reliability principle, software performance data principles, 130
- Response time:
 - online transaction processing (OLTP), 1
 - queuing systems, 154
 - utilization, service time, and networked queuing systems, 155
- Scalability. *See also* Scalable performance
 - performance contrasted, 1–2
 - software platform, 41
- Scalability testing, 75
 - batch jobs, 76
 - generally, 75
 - online transaction processing (OLTP), 77–82
- Scalable performance, 1
 - factors in, 3–4
 - Intel Core microarchitecture, 13–17
 - chipset, 20–21
 - motherboard, 19–20
 - networking, 27–28
 - processors, 18–19
 - RAIDs, 24
 - storage, 22
 - Turing model, 6, 30
 - sizing hardware, 35
 - software platform, 41
- Service demand(s):
 - covering index, 228
 - cursor-sharing, 229
 - wait events, 221
- Service level agreement (SLA), Intel Core microarchitecture and Turing model, 35
- Service-oriented architecture (SOA), 177–201. *See also* XML Web services
 - analytical model, 181
 - demand, 183
 - database server, 186
 - data storage, 186
 - generally, 183–184
 - network latency, 185
 - object creation, 184
 - XML SOAP serialization/deserialization, 184
 - XML Web service provider, 186
 - enterprise software, 55–61
 - model/measurement comparisons, 198
 - overview, 177–179
 - test results, 191
 - validity, 200
 - XML Web services, 179
- Service time, utilization, response time, and networked queuing systems, 159
- Single-queue multiple servers, multiple parallel queues versus, 160
- Single-user application software
 - categorization, 54
- Sizing hardware, 35
- SOA. *See* Service-oriented architecture (SOA)
- Software development kit (SDK),
 - application programming interfaces (APIs), 44
- Software development process, 83
 - agile software, 83
 - extreme programming, 84
- Software performance data principles, 129
- Software performance measurements,
 - stochastic nature of, 95
- Software performance testing, 86
 - batch jobs, 86–95
 - generally, 86
 - online transaction processing (OLTP), 87–95
- Software platform, 42–63
 - application programming interfaces (APIs), 44–47
 - generally, 44
 - Google, 46
 - Java, 45
 - Windows, 46
 - categorization, 53
 - enterprise software, 55
 - client/server architecture, 58
 - componentry, 61
 - defined, 55
 - monolithic architecture, 57
 - N-tier architecture, 60
 - service-oriented architecture, 61
 - three-tier architecture, 59
 - multithreading, 47

- overview, 42
- scalable performance, 3
- software stack, 42, 253
- Software stack:
 - application programming interface (API)
 - profiling, 254
 - languages, 42
- SQL server parameterization, performance,
 - and scalability testing factors, 108
- SQL tuning, optimization, and tuning
 - software application, 228
- Storage:
 - data latency reduction, optimization, and
 - tuning software, 232
 - Intel Core microarchitecture, 13–17
- Summarization file, API profiling, 260
- Sun machine, 17–18
- System bottlenecks. *See also* Bottlenecks
 - networked queuing systems, 170
 - Task Manager, 125
- System monitoring tools, Intel machine, 17
- System performance counters, 111
 - generally, 111
 - perfmon*:
 - CPU bottlenecks diagnosis, 119
 - disk I/O bottlenecks diagnosis, 121
 - memory leak diagnosis, 118
 - Task Manager, system bottlenecks
 - diagnosis, 125
 - UNIX/Linux platforms, 128
 - Windows performance console, 112
- Systems software, categorization, 53
- Task Manager, system bottlenecks
 - diagnosis, 125
- Testing software, scalable performance, 3
- Thread, process contrasted, 47
- Three-tier architecture, enterprise
 - software, 59
- Throughput:
 - noninteractive batch jobs, 1
 - queuing systems, 152
- Tuning testing and performance
 - optimization testing, 70
- Turing, Alan, 6–7, 205
- Turing machine:
 - computer technology, 6–7
 - Intel Core microarchitecture, 13–17
- UNIX/Linux platforms, system
 - performance counters, 128
- Utilization, service time, response
 - time, and networked queuing
 - systems, 159
- VMWare™, 43
- Volume principle, software performance
 - data principles, 130
- Von Neumann, John, 7–8
- Von Neumann bottleneck, 8
- Von Neumann machine, computer
 - technology, 7–8, 18
- Wait events:
 - optimization and tuning software
 - application, 205–244
 - service demands, optimization, and
 - tuning software, 221
- Web application software categorization, 55
- Windows, application programming
 - interfaces (APIs), 45
- Windows performance console, system
 - performance counters, 111
- Wright, Frank Lloyd, 177
- XML Web services. *See also* Service-oriented
 - architecture (SOA)
 - API profiling implementation,
 - 287–300
 - medical records (MedRec)
 - application, 188
 - service-oriented architecture (SOA)
 - application (queuing theory),
 - 177–179
 - test results, 191
- Zuse, Konrad, 5, 8
- Zuse machine, computer technology, 8

