

- Abyss Web servers, 100–108
- Ademaj, A., 55
- Aidemark, J., 55
- Akkerman, W., 232
- Albinet, A., 228, 287, 300
- Anderson, D., 142
- Anomaly-based intrusion detectors
 - algorithms, 152
 - assessing results, 156–157
 - background data, 149–153
 - composition of test data, 149–153
 - composition of training data, 146–149
 - evaluating dependability, 141–159
 - evaluation procedure, 142–143
 - impact of test data composition on effectiveness, 149–153
 - impact of training data composition on effectiveness, 146–149
 - Markov-based, 146, 147–148, 149, 151, 152
 - methodology for injecting signal into background data, 153–156
 - overview, 141–143
 - signal characteristics, 152–153
 - stride, 146, 147–148, 151, 152–153, 157
- Apache Web servers, 100–108
- Application layer, 37–38
- Arlat, J., 55, 91, 207, 228, 256, 285, 287, 288, 291, 311, 314
- AutoMark benchmark, 112
- Automatic system recovery (ASR), 36
- Automotive control systems, dependability
 - benchmarking, 111–139. *See also* electronic control units (ECUs)
 - benchmark properties, 136–138
 - COTS-based, 111
 - dependability measures, 117–118
 - electronic control units for, 111–112
 - engine model characteristics, 114–116
 - execution profile, 118–127, 134
 - experimental setup, 122–123, 129
 - how four-stroke engines work, 113–114
 - overview, 111–116, 116–117
 - procedure, 121–127
 - prototype implementation, 127–136
- Autonomic computing benchmark, 1–20
 - challenges and pitfalls, 15–19
 - characteristics, 2–3
 - compared with DBench-OLTP dependability benchmark, 6
 - cost issues, 3
 - disturbance overview, 7–8
 - methodology overview, 4–7
 - metrics overview, 2, 8–12
 - potential uses, 15
 - requirements overview, 2–4
 - results comparison, 14–15, 16
 - sample results, 10–15
 - system administrative considerations, 3
 - workload, 6, 12–15, 118–119, 134
- Autonomic Computing Maturity Model, 8, 18
- Availability benchmarking
 - attributes, 24–25, 36
 - “five-nines” probability, 23
 - requirements, defined, 23
 - in WEB-DB context, 95
- Avizienis, A., 220, 285
- Baggio, J., 346
- Ballista project
 - CRASH scale, 206–207, 313
 - early results, 204–205
 - fault dimensionality, 212
 - fine-grain data structures, 212–213
 - history, 204–206
 - interface robustness testing, 201–223
 - lessons learned, 215, 217–222

- Ballista project (*continued*)
 - overview, 201–202, 313
 - publications, 204
 - quantifying test results, 214–215
 - scalability, 205, 206, 221
 - system killer robustness vulnerabilities, 215, 216–217
 - system state initialization, 213–214
 - test case construction, 208–210
 - test concurrency, 211
 - test repeatability, 210–211
- BAR system, 165, 167
- Bartlett, W., 346
- Barton, J., 202, 203, 314
- Baseline performance measures
 - DBench-OLTP dependability benchmark, 70, 72, 81–82, 83, 84, 85, 86
 - WEB-DB dependability benchmark, 94, 95, 101–102
- Benchmark controllers (BCs), 231–234
- Benchmark management system (BMS)
 - defined, 68, 71, 94
 - as element in DeBERT setup, 257
 - for OLTP systems, 68, 71
 - for Web servers, 93, 94, 101–103
- Benchmark targets (BTs)
 - for automotive control systems, 116, 126–127, 134
 - compared with systems under benchmark, 65, 94, 103–104, 116
 - for DeBERT benchmark, 257, 271–272
 - defined, 67, 116
 - for diesel ECU experiment, 128, 134
 - for OLTP systems, 65, 67
 - operating systems as, 228–229, 288
 - real-time applications kernel as, 256, 257, 271
 - for WEB-DB systems, 93, 94, 103–104
- Benchmarking. *See also* dependability benchmarking
 - autonomic computing, 1–20
 - availability analysis, 23, 24–25, 36
 - Camp definition, 55
 - IBM's requirements, 2–4
 - measuring performance compared with availability, 35
 - methodology for assessing operating systems, 314–320
 - overview, xiii–xvii
 - for partially autonomic systems, 16–17
 - performance benchmarks compared with dependability benchmarks, xiii, xiv, xvi, xvii, 91
 - reliability analysis, 24, 25–26
 - for robustness, 26–28
 - role of metrics, 2
 - self-healing, 18–19
 - Service Complexity Benchmark, 28–31, 32, 35
 - serviceability analysis, 24, 30–31
 - user-relevant software reliability, 185–199
- Berrojo, L., 55
- Bezier, B., 202
- BFT systems. *See* Byzantine-fault-tolerant (BFT) systems
- Bhat, M., 346
- BMS. *See* benchmark management system (BMS)
- Bohrbugs, 192
- Breakpoints, 299, 316, 317
- Brodley, C.E., 149, 152, 154
- Brown, A., 6, 15, 18, 36, 68, 69, 85, 92, 100, 186, 256, 286
- Brown, A.B., 15
- Brubaker, D., 232
- Buchacker, K., 92, 314
- Byzantine-fault-tolerant (BFT) systems
 - BAR system, 165, 167
 - Castro-Liskov BFT, 164, 166
 - defined, 164–165
 - experimental results using Vajra framework, 173–176
 - Fleet system, 165, 167–168
 - Immune system, 165, 166–167
 - ITDOS system, 165, 167
 - SecureRing system, 165, 166–167
 - survey list, 165
- Camp, R.C., 55
- Cannon, E.H., 346
- Carreira, J., 55, 203, 286, 314
- Carrette, G., 203, 313
- Castelli, R.E., 190
- Castro-Liskov BFT (Byzantine fault tolerance), 163, 164, 166
- Check, M.A., 346
- Chen, C.L., 347
- Chen, P.M., 314
- Chevochet, P., 256
- Chillarege, R., 69, 77, 85, 96, 97, 100, 314
- Chou, A., 228, 286, 290, 314
- Christmannson, J., 69, 77, 96, 97, 100
- Cisco Systems, Inc., 142
- Cluster layer, 37, 46
- Cluster nodes, 45, 46, 47, 48–49

- Clusters. *See* SRB-X (System Recovery Benchmark for Clusters)
- Code injection, 316, 317, 318, 327–328, 331–332
- Complexity. *See* Service Complexity Benchmark A (SCB-A)
- Constantinescu, C., 55, 56, 57
- COTS (commercial-off-the-shelf)
 as automotive component option, 111
 benchmarking of components for space applications, 255–281
 as computer option, 55, 56
 Open Source alternative, 285–286
 pros and cons, 285
 software robustness vulnerabilities, 205, 206, 211, 217–218
- CPV (protocol violations), 57
- CRASH scale, 206–207, 230, 293
- Crashme program, 203, 313
- CRC (cyclical redundancy codes), 56
- Cristian, F., 203
- Crouzet, Y., 228
- Czeck, E., 202
- Das, D., 347
- Data breakpoints, 316, 317
- Data injection, 316, 318, 328–329, 332
- Database management systems (DBMS). *See also* on-line transaction processing (OLTP)
 comparing configurations using DBench-OLTP benchmark, 81–84
 in DBench-OLTP benchmarking examples, 79
 faultload based on operator faults, 76–77
 faultload based on software faults, 78–79
 as key components of OLTP, 63, 64
- DBench (Dependability Benchmarking Project), xvii, 6, 24, 65, 178, 186, 191–192, 201, 222, 228, 256, 286, 288
- DBench-OLTP dependability benchmark
 benchmark management system, 71
 compared with autonomic computing benchmark, 6
 compared with TPC-C, 64–65, 70, 71, 72, 73–74
 comparing different transactional systems, 80–85
 defined, 6, 64
 dependability measures, 71
 examples, 79–87
 experimental setup, 71–72
 faultload, 71, 74–79
 measures for, 70–71
 nonintrusiveness, 87
 overview, 64–65, 70
 performance measures, 70, 71, 72, 81–82, 83, 84, 85, 86
 portability, 86
 procedure and rules, 72–73
 properties, 85–87
 repeatability, 86
 representativeness, 85–86
 run phases, 72
 scalability, 86–87
 simplicity, 87
 system under benchmarking, 71–72, 79–80
 workload, 73–74
- Debar, H., 142, 152
- DeBERT benchmark
 conduct, procedures, and rules, 257–260
 divergence metric, 260–262
 execution steps, 259–260
 fault model, 265–268
 faultload, 274–275
 general features, 258–259
 metrics and measurement techniques, 260–262
 objective, 256
 properties, 277–278
 RTEMS real-time kernel, 271–274, 275, 276
 setup elements, 257–258
 specification, 256–268
 system under benchmarking, 257
 time measurement, 279–281
 trial, 268–278
 validation, 277–278
 workload, 262–265, 272
- Denning, D., 152
- Dependability attributes, 65
- Dependability benchmarking
 anomaly-based intrusion detectors, 141–159
 Apache vs. Abyss Web server example, 100–108
 automotive control systems, 111–139
 compared with performance benchmarking, xiii, xiv, xvi, xvii, 91
 DeBERT benchmark specification, 256–268
 defined, 92
 and device drivers, 286–307
 experimental setup component, 92
 faultload component, 92
 impact of technology, 56
 main components, 92
 measures component, 92
 operating system specifications, 228–234

- Dependability benchmarking (*continued*)
 - overview, xiii–xvii, 92–93
 - procedures and rules component, 92
 - role of environmental test tools, 55–60
 - Web servers (WEB-DB), 91–108
 - workload component, 92
- Detection interval, 5
- Detection time, 72, 73
- Detours interception tool, 232, 250
- DeVale, J., 91, 204, 219, 220, 222, 228, 285, 288, 293, 294, 311, 313
- Device drivers
 - analyzing impact of faults, 286–287
 - faulty, dependability benchmarking, 286–307
 - as operating system issue, 286
- D’haeseleer, P., 147
- Dingman, C., 36, 202, 205
- Disk management layer, 38
- Disruption load, 193–194
- Distributed fault injection, 164, 168
- Disturbances. *See also* faults
 - categories, 7–8
 - defined, 2
 - design challenges, 16
 - detection of restart failure, 8, 10
 - vs. faults, 2
 - load resolution, 8, 10
 - loss of data, 8, 10
 - metrics, 8–12
 - resource contention, 8, 9
 - sample Autonomic Computing Benchmark results, 12–15
 - unexpected shutdown, 7
- Dodd, P.E., 341
- Donohoe, P., 112
- Driver programming interface (DPI)
 - as aspect of operating system kernel, 288–289
 - comparative study, 289–290
 - faulty driver observation level, 293–294
 - hardware drivers, 290
 - overview, 289
 - software drivers, 290
 - system call categories, 290–291
- Drivers. *See* device drivers
- Driving cycles, 119, 120, 134–135
- Dtrace (Solaris Dynamic Tracing Guide), 43
- Durães, J., 55, 68, 69, 77, 92, 96, 97, 98, 100, 108, 215, 228, 286, 288, 295
- Duran, J., 206
- ECCs (error-correcting codes), 28, 56, 318
- Eckhardt, D.E., 220
- Edmonds, L.D., 341
- Edwards, D., 286, 287, 288, 314
- EEMBC (EDN Embedded Microprocessor Benchmark Consortium), 112
- Egan, J.P., 156
- Electronic control units (ECUs)
 - benchmark procedure, 121–127
 - comparing SoC-based diesel engines, 127–136
 - deducing dependability measures from experimental measurements, 132
 - diesel engine functional specification, 128
 - execution profile, 118–121, 134
 - experimental setup, illustrated, 129
 - model characteristics, 114–116
 - prototype description, 128–129
 - prototype experiment configurations, 132–135
 - prototype faultload specification tool, 130–131
 - prototype procedure implementation, 131–132
 - prototype workload implementation, 129–130
 - role in automotive control systems, 111–112
 - role of engine models, 119
- Electrostatic discharge (ESD) operating test, 60
- Elling, Richard, 191
- Embedded Microprocessor Benchmark Consortium (EEMBC), 112
- Emulab test bed, 173
- Engines, automotive, how they work, 113–114. *See also* automotive control systems, dependability benchmarking
- Environmental test tools, 55–60
- Error injection, 314, 315–321, 325, 330, 332, 333
- Error-correcting codes (ECCs), 28, 56, 318
- European Project on Dependability Benchmarking, 228
- Fabre, J.-C., 79
- Failures in time (FIT), 24, 25, 29
- Fault injection
 - anomaly detector test data methodology, 153–156
 - code injection, 316, 317, 318, 327–328, 331–332
 - data injection, 316, 318, 328–329, 332
 - directly into kernel space, 314
 - distributed, 164, 168

- IBM autonomic computing benchmark
 - methodology, 1–2, 4–7
- injection slots in autonomic computing
 - benchmark test phase, 4–7
- injection slots in DBench-OLTP runs, 72–73
- injection slots in WEB-DB runs, 98–100
- physical (PFI), 57
- silent data corruption and, 57–58
- simulated (SFI), 57
- stack injection, 316, 317, 322–325, 327, 330
- system register injection, 316, 318, 320, 325–327, 330–331
- in Vajra framework, 168, 170–172, 177, 178
- Vajra results, 174–175
- value of employing, 335–336
- Fault injection target (FIT), 78, 97
- Fault representativeness, 69. *See also* representativeness
- Fault Robustness Benchmark (FRB-A), 26–28, 29, 32, 35
- Faultload
 - Apache vs. Abyss comparison, 105–107
 - automatic control system benchmarking, 119–121, 130–131, 134
 - DBench-OLTP benchmarking, 68–69, 71, 74–79
 - DeBERT benchmarking, 265–268, 274
 - drivers and kernel benchmarking, 292, 293, 298
 - operating system benchmarking, 230–231, 247–249
 - WEB-DB benchmarking, 92, 93, 95, 96–99, 106, 107
- Faults
 - Byzantine, defined, 164–165
 - classes, 68–69
 - defined, 2
 - vs. disturbances, 2
 - hardware, 68, 76–77
 - operator, 68, 74–76
 - portability, 69
 - software, 68, 77–79
- Fernsler, K., 204, 214
- FERRARI system, 203
- FIAT system, 203, 314
- Field replaceable units (FRUs), 28–31
- FINE system, 314
- FIT (failures in time), 24, 25, 29
- FIT (fault injection target), 78, 97
- Fleet system, 165, 167–168
- Folkesson, P., 55
- Forrest, S., 142, 147, 149, 152, 153
- Four corners tests, 56, 57–60
- Fox, K.L., 152
- Fraser, T., 287
- FRB-A (Fault Robustness Benchmark), 26–28, 29, 32, 35
- FTAPE system, 203
- Fuchs, M., 112
- Fuzz project, 203, 313
- Ganapathi, Archana, 198
- Gao, W., 347
- Garzia, Mario R., 186, 197
- Gehani, N., 203
- Gehrke, J., 66, 68
- Ghosh, A.K., 142, 158
- Gil, P., 119–120, 138
- Godfrey, M.W., 286, 300
- Goodenough, J., 203
- Gracia, J., 116, 120, 138
- Gray, J., xiii, 68, 85, 91, 100, 192
- Ground truth
 - defined, 143
 - establishing, 144–145
 - overview, 143–144
 - support for evaluation goals, 145–146
- G-SWFIT (Generic Software Fault Injection Technique), 77, 79, 97, 100, 108
- Gu, W., 286, 288, 301, 314
- Guenzer, C.S., 341
- Gulati, K., 346
- Hardware faults
 - in DBench-OLTP benchmarking, 68, 76–77
 - in WEB-DB benchmarking, 92, 96, 98, 100
- Hardware layer, 38, 288, 289, 290
- Hareland, S., 56, 342
- Heffernan, D., 111
- Hellerstein, J., 18
- Heywood, J., 112
- Hill, I., 203
- Hofmeyr, S.A., 147, 157
- Horn, P., 1
- Howard, J.W., 341
- Hunt, G., 232
- IBM Autonomic Computing Benchmark development, 1–20
- ICs, impact of radiation on reliability, 341–347
- IEC (International Electrotechnical Commission), 60
- IEEE-ISTO 5001 standard, 127, 129. *See also* Nexus interface
- IFI (individual fault injectors), Vajra, 168, 171, 172, 174

- IIS (Internet Information Services), 190
- Ilgun, K., 142
- Immune system, 163, 165, 166–167
- Individual fault injectors (IFI), Vajra, 168, 171, 172, 174
- Injection interval, defined, 5
- Injection slots
 - in autonomic computing benchmark test phase, 4–7
 - in DBench-OLTP runs, 72–73
 - in WEB-DB runs, 98–100
- Injection time, 72, 73
- Instruction breakpoints, 316, 317
- Integrated circuits, impact of radiation on reliability, 341–347
- Intel, 80, 232, 298, 313, 318, 321, 325, 342, 345, 346
- Interceptor, in Vajra framework, 168–170
- Interface robustness testing, 201–223
- Internal combustion engines. *See* automotive control systems, dependability benchmarking
- International Electrotechnical Commission (IEC), 60
- International Technology Roadmap for Semiconductors (ITRS), 56, 342
- Internet Information Services (IIS), 190
- Internet Security Systems, Inc., 142
- Intrusion detection. *See* anomaly-based intrusion detectors
- Intrusion tolerant distributed object systems (ITDOS), 163, 165, 167
- Itanium microprocessors, 342, 344, 345, 346
- ITDOS (intrusion tolerant distributed object systems), 163, 165, 167
- ITRS (International Technology Roadmap for Semiconductors), 56, 342
- Iyer, R.K., 68, 85, 314, 317
- J2EE (Java 2 Enterprise Edition), 2, 4, 6–7, 12
- Jalote, Pankaj, 192
- Jarboui, T., 247, 288
- Java. *See* J2EE (Java 2 Enterprise Edition)
- Javitz, H., 152
- JEDEC standard, 343, 344, 345
- Jenn, E., 91
- Jha, S., 142, 146
- Johansson, A., 287
- JVM (Java Virtual Machine)
 - dependability benchmark implementation and results, 236–238
 - effort required for benchmark implementation, 250–251
 - refining benchmark temporal measures, 241–244, 245
 - Windows-Linux comparison with PostMark, 244–245, 246
- Kalakech, A., 92, 193, 228, 230, 234, 237, 247, 248, 288, 289
- Kanawati, G., 203
- Kanoun, K., xvii, 92, 228, 286, 288, 298
- Kao, W., 314
- Karapetian, A.V., 342, 346
- Karlsson, J., 55
- Katayama, Y., 347
- Katcher, J., 230
- Keep interval, 6
- Keep time, 72, 73
- Killourhy, K.S., 143
- Kim, S., 55
- Knorr, Eric, 185
- Koopman, P., 36, 79, 91, 92, 100, 202, 204, 206, 207, 212, 220, 228, 231, 256, 285, 288, 293, 294, 311, 313
- Korn, D., 212
- Kropp, N., 202, 204, 208, 313
- Kuhn, D.R., 76
- L & B (Lane and Brodley) detectors, 154–156
- Labrosse, J.J., 134
- Lane, T., 149, 152, 154
- Lanford, W.A., 341
- LANSCE (Los Alamos Neutron Science Center), 342, 343–345
- Laplante, P.A., 256
- Laprie, J.-C., 65
- Latency, 318, 320, 332, 333, 336
- Layering, system, 37–38
- Leaphart, E.G., 112
- Lee, I., 68, 85, 314
- Lee, W., 147
- Leen, G., 111
- Leveson, N.G., 112
- Levidow, B., 286
- Liden, P., 342
- Lightstone, S., 2, 93
- Lindholm, T., 230, 236
- Linpack benchmark, 56, 57, 58, 60, 345
- Linux operating system
 - benchmark execution sequence, 231–234
 - benchmarking methodology, 314–320
 - comparing kernel behavior on different processors, 313
 - crash latency analysis, 330–332
 - crash severity analysis, 332–335

- device driver overview, 286
- JVM dependability benchmark implementation and results, 236–238
- parameter corruption techniques, 230–231
- PostMark dependability benchmark implementation and results, 234–236
- PostMark-JVM comparison with Windows, 244–245, 246
- refining benchmark temporal measures, 239–244
- role of Strace interception tool, 232, 250
- Littlewood, Bev, 186, 220
- Loki fault injector, 177
- Los Alamos Neutron Science Center (LAN-SCE), 342, 343–345

- Maamar, A., 347
- Madeira, H., 36, 55, 65, 68, 69, 70, 71, 74, 77, 79, 86, 92, 96, 97, 98, 100, 108, 192, 195, 202, 228, 285, 286, 288, 295, 311
- MAFALDA system, 314
- Maintenance Robustness Benchmark (MRB-A), 26–28, 29, 32, 35, 191
- MALFADA-RT system, 203
- Marick, B., 209
- Markov-based detectors, 146, 147–148, 149, 151, 152
- Marsden, E., 285
- Martinez, R.J., 55
- Master fault injector (MFI), Vajra, 168, 171–172, 174
- Matassa, L., 286, 287, 288, 314
- Maturity index, 2, 3, 8, 11, 12, 13, 14–15, 19
- Mauro, James, 24, 36, 40
- Maxion, R.A., 147, 153, 156
- May, T.C., 341
- McCluskey, E.J., 55
- McGrath, R., 232
- Mé, L., 152
- Measures
 - automotive control system dependability, 117–118
 - for DBench-OLTP dependability benchmark, 70–71
 - as dependability benchmark component, 92
 - drivers and kernel dependability issues, 293–297
 - meaningful, as goal of Vajra framework, 164
 - operating system issues, 229–230, 311, 312–313
 - for quality of service, 17–18
 - for quantifying operating system dependability, 312–313
 - scoring, 17–19
 - for user-focused software reliability, 195–196
 - for WEB-DB dependability benchmark, 95–96
- Mehdi, N., 146
- Metrics. *See* maturity index; measures; throughput index
- MFI (master fault injector), Vajra, 168, 171–172, 174
- Michaels, Daniel, 186
- Microprocessors
 - hadron effect overview, 341–342
 - impact of radiation on reliability, 341–347
 - LANSCE experimental results, 345–346
 - manufacturing scaling trends, 342–343
 - neutron SER measurements, 342, 343–345
- Microsoft, 186, 188, 196–198. *See also* Windows operating systems
- Miller, B., 36, 203, 313
- Miller, G., 114
- Mitra, S., 347
- Moreira, F., 256
- Morioka, S., 347
- Motorola Power PC, 313, 318, 321, 325
- MRB-A (Maintenance Robustness Benchmark), 26–28, 29, 32, 35, 191
- Mukherjee, A., 36, 205, 228, 286
- Mukherjee, S.S., 347
- Murphy, B., 286
- Musa, J., 187, 193, 198, 214
- MuTs (modules under test), 201–202, 207, 209–210

- Neutrons
 - LANSCE experimental results, 345–346
 - microprocessor SER measurements, 342, 343–345
 - overview of impact on IC reliability, 341–342
 - SER characterization, 343–345
- Nexus interface, 129, 131–132, 136, 137
- Ng, W., 314
- Noda, K., 346
- Nodes, cluster, 45, 46, 47, 48–49
- Nonintrusiveness
 - as automotive control system benchmark property, 137
 - as DeBERT benchmark property, 278
 - as OLTP system benchmark property, 70, 87
- Normand, E., 343

- Off-the-shelf (OTS) components. *See* COTS (commercial-off-the-shelf)
- OMG (Object Management Group), 203
- On-line transaction processing (OLTP) systems, dependability benchmarking, 63–87. *See also* DBench-OLTP dependability benchmark; TPC-C (Transaction Processing Performance Council Benchmark C)
 - application environments, 64–65
 - benchmark management system, 71
 - benchmark target, 65
 - compared with autonomic computing benchmark, 6
 - compared with TPC-C, 64–65, 70, 71, 72, 73–74
 - comparing different transactional systems, 80–85
 - defined, 6, 64
 - dependability measures, 71
 - examples, 79–87
 - experimental setup, 67–68, 71–72
 - faultload, 68–69, 71, 74–79
 - metrics, 66–67, 70–71
 - nonintrusiveness, 70, 87
 - overview, 64–65, 70
 - performance measures, 70, 71, 72, 81–82, 83, 84, 85, 86
 - portability, 69, 86
 - procedures and rules, 68, 72–73
 - properties, 69–70, 85–87
 - repeatability, 69, 86
 - representativeness, 69, 85–86
 - run phases, 72
 - scalability, 69, 86–87
 - simplicity, 70, 87
 - specifying benchmark components, 66–69
 - system under benchmark, 65, 67, 71–72, 79–80
 - system under test, 65
 - typical environment, 65
 - workload, 68, 73–74
- Open Source, as alternative to COTS components, 285–286
- Operating system layer, 38
- Operating systems. *See also* Linux operating system; Windows operating systems for avionics and space environments, 255–281
 - benchmarking against faults, 311–336
 - characterization studies, 313–314
 - comparing configurations using DBench-OLTP benchmark, 82
 - comparing configurations using WEB-DB benchmark, 100–108
 - creating dependability benchmark, 311–336
 - dependability benchmark execution profile, 230–231
 - dependability benchmark implementation, 231–234
 - dependability benchmark properties, 247–250
 - dependability benchmark specifications, 228–234
 - device driver issues, 285–307
 - failure behavior studies, 313–314
 - faultload, 230–231, 247–249
 - interactions between kernel and its environment, 287–289
 - metrics for quantifying dependability, 312–313
 - parameter corruption, 230–231, 247–248
 - robustness overview, 227–228
 - workload, 230, 247
- Operational faults, 96, 98, 99, 105, 106
- Operator faults, 68, 69, 73, 74–76, 79–87
- Orchestra fault injector, 178
- Ostrand, T.J., 207
- Ouais, I., 289
- Palau, J.M., 341
- Pan, J., 204, 212
- PASIS protocol, 179, 180
- Passino, K.M., 112
- Pasztor, Andy, 186
- Patterson, D., 36, 68, 69, 85, 100, 198, 286
- Pentium 4 (Intel), 313, 318, 321, 325
- Percentage of failure rate (PFR), 25–26, 27, 29, 30, 31
- Performance benchmarks. *See* SPECjAppServer2004 performance benchmark; SPECWeb99 performance benchmark; Standard Performance Evaluation Corporation (SPEC) benchmarks; TPC-C (Transaction Processing Performance Council Benchmark C)
- Phalanx system, 163
- Pientka, B., 211
- Portability
 - as automotive control system benchmark property, 136
 - as DeBERT benchmark property, 277–278
 - as OLTP system benchmark property, 69, 86
 - as operating system benchmark property, 249–250
 - as WEB-DB benchmark property, 100

- POSIX operating system, 203, 204, 205, 206, 207–208, 213, 220, 228, 231, 232
- PostMark
 dependability benchmark implementation and results, 234–236
 effort required for benchmark implementation, 250–251
 refining benchmark temporal measures, 239–241, 242
 Windows-Linux comparison with JVM, 244–245, 246
- Power PC (G4), 313, 318, 321, 325
- Practical BFT system. *See* Castro-Liskov BFT (Byzantine fault tolerance)
- Pramanick, Ira, 24
- Puaut, I., 256
- Pulkrabek, W.W., 114
- QNX operating system, 205, 216, 218
- Quach, N., 347
- Query/Update (Q/U) protocol, 179–180
- Radiation, impact on integrated circuit reliability, 341–347
- RAID (redundant array of independent disks), 19, 38
- Ramakrishnan, R., 66, 68
- Rashid, F., 347
- Recovery, 5–6, 24, 36, 72, 73, 336
- Redlin, C., 15
- Redundant array of independent disks (RAID), 19, 38
- Register injection, 316, 318, 320, 325–327, 330–331
- Reinhardt, S.K., 347
- Reliability
 benchmarking for software, 185–199
 as dependability attribute, 65
 impact of radiation on ICs, 341–347
- Repeatability
 as automotive control system benchmark property, 137–138
 as Ballista project test issue, 210–211
 as DeBERT benchmark property, 277
 as OLTP system benchmark property, 69, 86
 as operating system benchmark property, 249
 as WEB-DB benchmark property, 100
- Representativeness
 as automotive control system benchmark property, 138
 as DeBERT benchmark property, 277
 as OLTP system benchmark property, 69, 85–86
 as operating system benchmark property, 247–248
 as WEB-DB benchmark property, 100
- Reproducibility, defined, 249. *See also* repeatability
- Rimen, M., 317
- Robustness
 as availability attribute, 25, 36
 Ballista interface testing project, 201–223
 benchmarks for, 26–28
 defined, 202, 285
- RoCADE (robustness characterization against driver errors)
 experimental testbed, 297–300
 results and analyses, 300–305
- Rodriguez, M., 203, 256, 295, 296
- Rotenberg, E., 347
- RTEMS (real-time executive for multiprocessor systems), 271–274, 275, 276
- Ruiz, J.-C., 116
- Russel, G., 347
- Russinovich, M., 204
- Ryan, J., 152
- SaaS (Software as a Service), 185–186, 194–195
- Samson, J.R., 55
- Sato, H., 346
- Satoh, S., 342
- Scalability
 as automotive control system benchmark property, 137
 as Ballista project test issue, 205, 206, 221
 as DeBERT benchmark property, 278
 as OLTP system benchmark property, 69, 86–87
- SCB-A (Service Complexity Benchmark A), 28–31, 32, 35
- Schuette, M., 202
- Scott, D.T., 55
- SDC (silent data corruption), 56–60, 345
- SecureRing system, 165, 166–167
- SecureSpread system, 163
- Self-healing systems, 1, 17, 18–19
- Semiconductor devices. *See* microprocessors
- SER (soft error rate), 341, 342, 343–346
- Service Complexity Benchmark A (SCB-A), 28–31, 32, 35
- Services, cluster, 45, 46, 47–50
- Shannon, C.E., 147
- Shelton, C., 204, 216, 228, 230, 234

- Shetti, Nitin M., 192
- Shivakumar, P., 56, 342
- Shum, P., 6
- Shum, Peter, 186
- Sieh, V., 314
- Siewiorek, D., 36, 202, 228, 286
- Silent data corruption (SDC), 56–60, 345
- Simons, S., 347
- Simplicity
 - as OLTP dependability benchmark characteristic, 70, 87
 - as System Recovery Benchmark objective, 44, 50
- Single event upsets (SEUs), 121, 341, 343
- Single-bit errors, 56–57, 315, 317–318, 325, 327
- Slegel, T.H., 346
- Slegel, T.J., 346
- Slot intervals, 5–6
- Soft error rate (SER), 341, 342, 343–346
- Software. *See also* operating systems
 - error-return-code model, 203
 - handling exceptions, 203
 - interface robustness testing, 201–223
 - MuTs (modules under test), 201–202, 207, 209–210
 - reliability benchmarking, 185–199
 - signal-based model, 203
- Software as a Service (SaaS), 185–186, 194–195
- Software faults
 - in DBench-OLTP benchmarking, 73, 77–79
 - defined, 68
 - in WEB-DB benchmarking, 96–98, 105, 106, 107, 108
- Somani, A.K., 55, 76
- Soto, P., 152
- Space applications, dependability benchmarking of real-time components for, 255–281
- Spainhower, L., 346
- SPEC (Standard Performance Evaluation Corporation) benchmarks, xvi, 4, 23, 35, 65, 95, 102
- SPECjAppServer2004 performance benchmark, 4, 5, 6
- SPECWeb99 performance benchmark, 93, 94, 95, 98, 99, 100, 103, 108
- Spooner, J.T., 112
- SRB-A (System Recovery Benchmark for Operating Systems)
 - defining recovery, 40
 - defining systems, 38–40
 - factors affecting recovery, 41–43
 - measurements, 40–41
 - overview, 38
 - results metrics, 44–45
 - running, 43–44
 - workload issues, 43
- SRB-X (System Recovery Benchmark for Clusters)
 - client workload, 50
 - cluster configuration, 49
 - cluster definitions, 45–46
 - cluster layers, 37, 46
 - cluster outage modes, 46–47
 - defined, 45
 - reconfiguration outage triggers, 48–49
 - recovery measurement, 47–48
 - score calculation, 50–51
 - variables affecting recovery, 49–50
- Stack injection, 316, 317, 322–325, 327, 330
- Standard Performance Evaluation Corporation (SPEC) benchmarks, xvi, 4, 23, 35, 65, 95, 102
- Stathis, H., 56
- Steady-state time, 72, 73
- Stide detectors, 146, 147–148, 151, 152–153, 157
- Stott, D.T., 36, 315
- Strace interception tool, 232, 250
- Strigini, Lorenzo, 186, 220
- Sullivan, M., 85, 314
- Sun Microsystems, 24, 31, 43, 51–52, 92
- Suri, N., 287
- Survivable systems, 163, 164, 165, 179
- SUT (system under test), 5, 65
- Swift, M.M., 286, 305
- System Recovery Benchmarks (SRBs). *See also* SRB-A (System Recovery Benchmark for Operating Systems); SRB-X (System Recovery Benchmark for Clusters)
 - defined, 35
 - framework overview, 36–38
 - relationship to Fault and Maintenance Robustness Benchmarks (FRB-A and MRB-A), 35
 - relationship to Service Complexity Benchmark (SCB-A), 35
 - role of recovery attribute, 36
- System register injection, 316, 318, 320, 325–327, 330–331
- System under benchmarking (SUB)
 - in Apache vs. Abyss Web server example, 101, 102, 103–104

- in automotive control systems, 116, 122–123, 127
- compared with benchmark targets, 65, 94, 103–104, 116
- computer system software architecture, 288
- for DBench-OTLP dependability benchmark, 67, 71–72, 79–80
- for DeBERT benchmark, 257
- defined, 65, 67, 229, 288
- for WEB-DB dependability benchmark, 93–94
- System under test (SUT), 5, 65
- Taber, A., 343
- Tan, K.M.C., 147, 153, 156
- Telcordia, 25
- Temperature and voltage operating tests. *See* four corners tests
- Teng, H.S., 146
- Test tools, environmental, 55–60
- Thévenod-Fosse, P., 206
- Throughput index, 2, 3, 8, 12, 13–15, 19
- Tosaka, Y., 342
- TPC-C (Transaction Processing Performance Council Benchmark C)
 - defined, 63
 - dependability measures, 71
 - overview, 63–64
 - performance measures, 70–71
 - relationship to DBench-OLTP, 6, 63, 64–65, 70, 71, 72, 73–74, 76, 85
 - specifications, 5, 63–64, 70, 71, 72, 86
- Trade 6 reference workload example, 6, 12–15
- Transaction Processing Performance Council. *See* TPC-C (Transaction Processing Performance Council Benchmark C)
- Transactional systems. *See* on-line transaction processing (OLTP)
- Transparency, as goal of Vajra framework, 164
- Trivedi, K.S., 91
- Tsai, T., 36, 203, 286, 311, 312
- Tschaeche, O., 92
- Tu, Q., 286, 300
- Unconditional measures, 66–67
- Undetected errors, 56–57
- Unexpected shutdown faults, 7
- User-relevant software reliability, 185–199
- Vaidya, N.H., 76
- Vajra framework
 - architecture, 168–173
 - configuration, 172
 - fault injection, 170–172
 - future work, 178–181
 - goals, 164
 - individual fault injectors, 172
 - insights, 175–176
 - interception technique, 168–170
 - limitations, 172–173
 - master fault injector, 171–172
 - overhead, 174
 - overview, 164
 - results, 174–175
- Valdes, A., 152
- Vara, Vauhini, 186
- Vieira, M., 65, 68, 69, 70, 71, 74, 79, 86, 90, 92, 100, 285
- Vo, K.-P., 222
- Voas, J., 79
- Vollrath, J., 58
- VxWorks operating system, 211, 214, 217, 220–221, 279
- Wagner, D., 152
- Wagner, K.D., 55
- Warrender, C., 142, 147, 157
- Weaver, C.T., 347
- Weaver, H.T., 346
- Weaver, W., 147
- Web servers. *See also* WEB-DB dependability benchmark
 - Apache compared with Abyss, 100–108
 - dependability benchmarking, 91–108
 - measuring dependability, 93–100
 - overview, 91–93
- Web-based testing service. *See* Ballista project
- WEB-DB dependability benchmark
 - benchmark management system, 93–94
 - benchmark properties, 100
 - benchmark target, 93, 94, 103–104
 - defined, 92
 - dependability measures, 95–96
 - execution effort, 107–108
 - execution phases, 94–95
 - experimental setup, 93–94, 101–103
 - faultload, 92, 93, 95, 96–99, 106, 107
 - injection slot execution profile, 98–99
 - system under benchmarking, 93–94
 - workload, 95
- Weideman, N., 256
- Wender, S., 343
- WER (Windows Error Reporting), 197, 198
- Wichmann, B.A., 256
- Windows Error Reporting (WER), 197, 198

- Windows operating systems
 - benchmark execution sequence, 231–234
 - device driver overview, 286
 - JVM dependability benchmark implementation and results, 236–238
 - parameter corruption techniques, 230–231
 - PostMark dependability benchmark implementation and results, 234–236
 - PostMark-JVM comparison with Linux, 244–245, 246
 - refining benchmark temporal measures, 239–244
 - role of Detours interception tool, 232, 250
- Windows Server, 197–198
- Woods, M.H., 341
- Workload
 - automotive control system benchmarking, 118–119, 129–130, 134
 - DeBERT benchmarking, 262–265, 272
 - as dependability benchmark component, 92
 - drivers and kernel benchmarking, 292–293
 - DBench-OLTP benchmarking, 68, 73–74
 - operating system benchmarking, 230, 247
 - software reliability benchmarking, 193, 194
 - WEB-DB benchmarking, 95
- Xception tool, 203, 269–271, 276, 314
- Xiang, D., 147
- Xu, J., 314
- Yellin, F., 230, 236
- Yuste, P., 132, 135
- Zaatar, W., 289
- Zeng, C., 346
- Zhu, J., 24, 36, 69, 92, 93, 100, 256
- Ziegler, J.F., 341