Java compilation results

Contents

8x8

JMH version: 1.35

VM version: JDK 20.0.1, Java HotSpot(TM) 64-Bit Server VM, 20.0.1+9-29

VM invoker: C:\Program Files\Java\jdk-20\bin\java.exe

VM options: -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\lib\idea_rt.jar=49205:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8

Blackhole mode: compiler (auto-detected, use -Djmh.blackhole.autoDetect=false to disable)

Warmup: <none>

Measurement: 5 iterations, 10 s each

Timeout: 10 min per iteration

Threads: 1 thread, will synchronize iterations

Benchmark mode: Average time, time/op

Benchmark: org.example.MatrixMultiplicationBenchmarking.multiplication

Run progress: 0,00% complete, ETA 00:04:10

Fork: 1 of 5

Iteration   1: 0,004 ms/op

Iteration   2: 0,003 ms/op

Iteration   3: 0,003 ms/op

Iteration   4: 0,003 ms/op

Iteration   5: Average Memory Usage: 6.2688930602542206 bytes

Average CPU Time Used: 7.444429513432486E-4 ms

0,003 ms/op

Run progress: 20,00% complete, ETA 00:03:24

Fork: 2 of 5

Iteration   1: 0,004 ms/op

Iteration   2: 0,003 ms/op

Iteration   3: 0,003 ms/op

Iteration   4: 0,003 ms/op

Iteration   5: Average Memory Usage: 1.4951928775254448 bytes

Average CPU Time Used: 8.61876418681904E-4 ms

0,003 ms/op

Run progress: 40,00% complete, ETA 00:02:33

Fork: 3 of 5

Iteration 1: 0,004 ms/op

Iteration 2: 0,003 ms/op

Iteration 3: 0,003 ms/op

Iteration 4: 0,003 ms/op

Iteration 5: Average Memory Usage: 3.500351955723359 bytes

Average CPU Time Used: 0.0010266858397073887 ms

0,003 ms/op

Run progress: 60,00% complete, ETA 00:01:42

Fork: 4 of 5

Iteration 1: 0,004 ms/op

Iteration 2: 0,003 ms/op

Iteration 3: 0,003 ms/op

Iteration 4: 0,003 ms/op

Iteration 5: Average Memory Usage: 6.586391485432776 bytes

Average CPU Time Used: 9.913007998409446E-4 ms

0,003 ms/op

Run progress: 80,00% complete, ETA 00:00:51

Fork: 5 of 5

Iteration 1: 0,004 ms/op

Iteration 2: 0,003 ms/op

Iteration 3: 0,003 ms/op

Iteration 4: 0,003 ms/op

Iteration 5: Average Memory Usage: 4.445895714723919 bytes

Average CPU Time Used: 9.665460061304766E-4 ms

0,003 ms/op

Result "org.example.MatrixMultiplicationBenchmarking.multiplication":

0,003 ±(99.9%) 0,001 ms/op [Average]

(min, avg, max) = (0,003, 0,003, 0,004), stdev = 0,001

CI (99.9%): [0,003, 0,004] (assumes normal distribution)

Run complete. Total time: 00:04:15

REMEMBER: The numbers below are just data. To gain reusable insights, you need to follow up on why the numbers are the way they are. Use profilers (see -prof, -lprof), design factorial experiments, perform baseline and negative tests that provide experimental control, make sure the benchmarking environment is safe on JVM/OS/HW level, ask for reviews from the domain experts. Do not assume the numbers tell you what you want them to tell.

NOTE: Current JVM experimentally supports Compiler Blackholes, and they are in use. Please exercise extra caution when trusting the results, look into the generated code to check the benchmark still works, and factor in a small probability of new VM bugs. Additionally, while comparisons between different JVMs are already problematic, the performance difference caused by different Blackhole modes can be very significant. Please make sure you use the consistent Blackhole mode for comparisons.

Benchmark	Mode	Cnt	Score	Error	Units
MatrixMultiplicationBenchmarking.multiplication	avgt	25	0,003	± 0,001	ms/op

Process finished with exit code 0

32x32

JMH version: 1.35

VM version: JDK 20.0.1, Java HotSpot(TM) 64-Bit Server VM, 20.0.1+9-29

VM invoker: C:\Program Files\Java\jdk-20\bin\java.exe

VM options: -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\lib\idea_rt.jar=49308:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8

Blackhole mode: compiler (auto-detected, use -Djmh.blackhole.autoDetect=false to disable)

Warmup: <none>

Measurement: 5 iterations, 10 s each

Timeout: 10 min per iteration

Threads: 1 thread, will synchronize iterations

Benchmark mode: Average time, time/op

Benchmark: org.example.MatrixMultiplicationBenchmarking.multiplication

Run progress: 0,00% complete, ETA 00:04:10

Fork: 1 of 5

Iteration 1: 0,028 ms/op

Iteration 2: 0,025 ms/op

Iteration 3: 0,024 ms/op

Iteration 4: 0,024 ms/op

Iteration 5: Average Memory Usage: 22.34158724969513 bytes

Average CPU Time Used: 0.013411222847768094 ms

0,024 ms/op

Run progress: 20,00% complete, ETA 00:03:24

Fork: 2 of 5

Iteration 1: 0,031 ms/op

Iteration 2: 0,026 ms/op

Iteration 3: 0,024 ms/op

Iteration 4: 0,024 ms/op

Iteration 5: Average Memory Usage: 19.667894432687778 bytes

Average CPU Time Used: 0.012640642962761878 ms

0,026 ms/op

Run progress: 40,00% complete, ETA 00:02:33

Fork: 3 of 5

Iteration 1: 0,032 ms/op

Iteration 2: 0,026 ms/op

Iteration 3: 0,024 ms/op

Iteration 4: 0,024 ms/op

Iteration 5: Average Memory Usage: 21.226168720456563 bytes

Average CPU Time Used: 0.00954884684887033 ms

0,024 ms/op

Run progress: 60,00% complete, ETA 00:01:42

Fork: 4 of 5

Iteration 1: 0,033 ms/op

Iteration 2: 0,025 ms/op

Iteration 3: 0,024 ms/op

Iteration 4: 0,023 ms/op

Iteration 5: Average Memory Usage: 59.35557555117983 bytes

Average CPU Time Used: 0.0117589738209755 ms

0,023 ms/op

Run progress: 80,00% complete, ETA 00:00:51

Fork: 5 of 5

Iteration 1: 0,031 ms/op

Iteration 2: 0,025 ms/op

Iteration 3: 0,024 ms/op

Iteration 4: 0,024 ms/op

Iteration 5: Average Memory Usage: 44.12990079562545 bytes

Average CPU Time Used: 0.011262224992914694 ms

0,023 ms/op

Result "org.example.MatrixMultiplicationBenchmarking.multiplication":

0,026 ±(99.9%) 0,002 ms/op [Average]

(min, avg, max) = (0,023, 0,026, 0,033), stdev = 0,003

CI (99.9%): [0,023, 0,028] (assumes normal distribution)

Run complete. Total time: 00:04:15

REMEMBER: The numbers below are just data. To gain reusable insights, you need to follow up on why the numbers are the way they are. Use profilers (see -prof, -lprof), design factorial experiments, perform baseline and negative tests that provide experimental control, make sure the benchmarking environment is safe on JVM/OS/HW level, ask for reviews from the domain experts. Do not assume the numbers tell you what you want them to tell.

NOTE: Current JVM experimentally supports Compiler Blackholes, and they are in use. Please exercise extra caution when trusting the results, look into the generated code to check the benchmark still works, and factor in a small probability of new VM bugs. Additionally, while comparisons between different JVMs are already problematic, the performance difference caused by different Blackhole modes can be very significant. Please make sure you use the consistent Blackhole mode for comparisons.

Benchmark	Mode	Cnt	Score	Error	Units
MatrixMultiplicationBenchmarking.multiplication	avgt	25	0,026	± 0,002	ms/op

Process finished with exit code 0

64x64

JMH version: 1.35

VM version: JDK 20.0.1, Java HotSpot(TM) 64-Bit Server VM, 20.0.1+9-29

VM invoker: C:\Program Files\Java\jdk-20\bin\java.exe

VM options: -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\lib\idea_rt.jar=49424:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8

Blackhole mode: compiler (auto-detected, use -Djmh.blackhole.autoDetect=false to disable)

Warmup: <none>

Measurement: 5 iterations, 10 s each

Timeout: 10 min per iteration

Threads: 1 thread, will synchronize iterations

Benchmark mode: Average time, time/op

Benchmark: org.example.MatrixMultiplicationBenchmarking.multiplication

Run progress: 0,00% complete, ETA 00:04:10

Fork: 1 of 5

Iteration   1: 0,205 ms/op

Iteration   2: 0,224 ms/op

Iteration   3: 0,200 ms/op

Iteration   4: 0,171 ms/op

Iteration   5: Average Memory Usage: 516.1711807569536 bytes

Average CPU Time Used: 0.0990692689747362 ms

0,172 ms/op

Run progress: 20,00% complete, ETA 00:03:24

Fork: 2 of 5

Iteration   1: 0,223 ms/op

Iteration   2: 0,215 ms/op

Iteration   3: 0,198 ms/op

Iteration   4: 0,170 ms/op

Iteration   5: Average Memory Usage: 245.29120638718845 bytes

Average CPU Time Used: 0.10333104887577585 ms

0,174 ms/op

Run progress: 40,00% complete, ETA 00:02:33

Fork: 3 of 5

Iteration 1: 0,219 ms/op

Iteration 2: 0,215 ms/op

Iteration 3: 0,210 ms/op

Iteration 4: 0,176 ms/op

Iteration 5: Average Memory Usage: 299.87805961756993 bytes

Average CPU Time Used: 0.10988868398598621 ms

0,174 ms/op

Run progress: 60,00% complete, ETA 00:01:42

Fork: 4 of 5

Iteration 1: 0,190 ms/op

Iteration 2: 0,230 ms/op

Iteration 3: 0,193 ms/op

Iteration 4: 0,180 ms/op

Iteration 5: Average Memory Usage: 350.7022546060807 bytes

Average CPU Time Used: 0.07349274836558425 ms

0,180 ms/op

Run progress: 80,00% complete, ETA 00:00:51

Fork: 5 of 5

Iteration 1: 0,196 ms/op

Iteration 2: 0,236 ms/op

Iteration 3: 0,222 ms/op

Iteration 4: 0,183 ms/op

Iteration 5: Average Memory Usage: 217.50434479246448 bytes

Average CPU Time Used: 0.07826641377180918 ms

0,178 ms/op

Result "org.example.MatrixMultiplicationBenchmarking.multiplication":

0,197 ±(99.9%) 0,016 ms/op [Average]

(min, avg, max) = (0,170, 0,197, 0,236), stdev = 0,021

CI (99.9%): [0,182, 0,213] (assumes normal distribution)

Run complete. Total time: 00:04:15

REMEMBER: The numbers below are just data. To gain reusable insights, you need to follow up on why the numbers are the way they are. Use profilers (see -prof, -lprof), design factorial experiments, perform baseline and negative tests that provide experimental control, make sure the benchmarking environment is safe on JVM/OS/HW level, ask for reviews from the domain experts. Do not assume the numbers tell you what you want them to tell.

NOTE: Current JVM experimentally supports Compiler Blackholes, and they are in use. Please exercise extra caution when trusting the results, look into the generated code to check the benchmark still works, and factor in a small probability of new VM bugs. Additionally, while comparisons between different JVMs are already problematic, the performance difference caused by different Blackhole modes can be very significant. Please make sure you use the consistent Blackhole mode for comparisons.

Benchmark	Mode	Cnt	Score	Error	Units
MatrixMultiplicationBenchmarking.multiplication	avgt	25	0,197	± 0,016	ms/op

Process finished with exit code 0

128x128

JMH version: 1.35

VM version: JDK 20.0.1, Java HotSpot(TM) 64-Bit Server VM, 20.0.1+9-29

VM invoker: C:\Program Files\Java\jdk-20\bin\java.exe

VM options: -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\lib\idea_rt.jar=64220:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8

Blackhole mode: compiler (auto-detected, use -Djmh.blackhole.autoDetect=false to disable)

Warmup: <none>

Measurement: 5 iterations, 10 s each

Timeout: 10 min per iteration

Threads: 1 thread, will synchronize iterations

Benchmark mode: Average time, time/op

Benchmark: org.example.MatrixMultiplicationBenchmarking.multiplication

Run progress: 0,00% complete, ETA 00:04:10

Fork: 1 of 5

Iteration 1: 2,051 ms/op

Iteration 2: 1,649 ms/op

Iteration 3: 2,625 ms/op

Iteration 4: 1,499 ms/op

Iteration 5: Average Memory Usage: 2220.970913174195 bytes

Average CPU Time Used: 0.539681964719093 ms

1,627 ms/op

Run progress: 20,00% complete, ETA 00:03:24

Fork: 2 of 5

Iteration 1: 2,093 ms/op

Iteration 2: 1,569 ms/op

Iteration 3: 1,566 ms/op

Iteration 4: 1,549 ms/op

Iteration 5: Average Memory Usage: -11.297233512521306 bytes

Average CPU Time Used: 0.1812311524845942 ms

1,540 ms/op

Run progress: 40,00% complete, ETA 00:02:33

Fork: 3 of 5

Iteration 1: 1,981 ms/op

Iteration 2: 1,578 ms/op

Iteration 3: 1,578 ms/op

Iteration 4: 1,570 ms/op

Iteration 5: Average Memory Usage: -208.23182698303305 bytes

Average CPU Time Used: 0.36887532407863216 ms

1,574 ms/op

Run progress: 60,00% complete, ETA 00:01:41

Fork: 4 of 5

Iteration 1: 2,157 ms/op

Iteration 2: 1,701 ms/op

Iteration 3: 1,583 ms/op

Iteration 4: 1,645 ms/op

Iteration 5: Average Memory Usage: 3672.1073862463 bytes

Average CPU Time Used: 0.38049838232257177 ms

1,635 ms/op

Run progress: 80,00% complete, ETA 00:00:50

Fork: 5 of 5

Iteration 1: 2,101 ms/op

Iteration 2: 1,670 ms/op

Iteration 3: 1,582 ms/op

Iteration 4: 1,563 ms/op

Iteration 5: Average Memory Usage: 1745.0142698855052 bytes

Average CPU Time Used: 0.4964241345734144 ms

1,592 ms/op

Result "org.example.MatrixMultiplicationBenchmarking.multiplication":

1,731 ±(99.9%) 0,206 ms/op [Average]

(min, avg, max) = (1,499, 1,731, 2,625), stdev = 0,275

CI (99.9%): [1,525, 1,937] (assumes normal distribution)

Run complete. Total time: 00:04:14

REMEMBER: The numbers below are just data. To gain reusable insights, you need to follow up on why the numbers are the way they are. Use profilers (see -prof, -lprof), design factorial experiments, perform baseline and negative tests that provide experimental control, make sure the benchmarking environment is safe on JVM/OS/HW level, ask for reviews from the domain experts. Do not assume the numbers tell you what you want them to tell.

NOTE: Current JVM experimentally supports Compiler Blackholes, and they are in use. Please exercise extra caution when trusting the results, look into the generated code to check the benchmark still works, and factor in a small probability of new VM bugs. Additionally, while comparisons between different JVMs are already problematic, the performance difference caused by different Blackhole modes can be very significant. Please make sure you use the consistent Blackhole mode for comparisons.

Benchmark	Mode	Cnt	Score	Error	Units
MatrixMultiplicationBenchmarking.multiplication	avgt	25	1,731	± 0,206	ms/op

Process finished with exit code 0

256x256

JMH version: 1.35

VM version: JDK 20.0.1, Java HotSpot(TM) 64-Bit Server VM, 20.0.1+9-29

VM invoker: C:\Program Files\Java\jdk-20\bin\java.exe

VM options: -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\lib\idea_rt.jar=64573:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8

Blackhole mode: compiler (auto-detected, use -Djmh.blackhole.autoDetect=false to disable)

Warmup: <none>

Measurement: 5 iterations, 10 s each

Timeout: 10 min per iteration

Threads: 1 thread, will synchronize iterations

Benchmark mode: Average time, time/op

Benchmark: org.example.MatrixMultiplicationBenchmarking.multiplication

Run progress: 0,00% complete, ETA 00:04:10

Fork: 1 of 5

Iteration   1: 20,642 ms/op

Iteration   2: 21,149 ms/op

Iteration   3: 19,879 ms/op

Iteration   4: 19,276 ms/op

Iteration   5: Average Memory Usage: 30061.599681655392 bytes

Average CPU Time Used: 7.124154397134898 ms

18,900 ms/op

Run progress: 20,00% complete, ETA 00:03:25

Fork: 2 of 5

Iteration   1: 20,619 ms/op

Iteration   2: 19,876 ms/op

Iteration   3: 19,946 ms/op

Iteration   4: 20,549 ms/op

Iteration   5: Average Memory Usage: 18627.57543716958 bytes

Average CPU Time Used: 6.257015046766979 ms

20,988 ms/op

Run progress: 40,00% complete, ETA 00:02:33

Fork: 3 of 5

Iteration 1: 21,441 ms/op

Iteration 2: 20,577 ms/op

Iteration 3: 20,537 ms/op

Iteration 4: 20,418 ms/op

Iteration 5: Average Memory Usage: 10336.066061106523 bytes

Average CPU Time Used: 6.718827415359208 ms

20,470 ms/op

Run progress: 60,00% complete, ETA 00:01:42

Fork: 4 of 5

Iteration 1: 22,046 ms/op

Iteration 2: 21,327 ms/op

Iteration 3: 20,597 ms/op

Iteration 4: 20,436 ms/op

Iteration 5: Average Memory Usage: -1250.0962025316455 bytes

Average CPU Time Used: 7.452742616033755 ms

21,306 ms/op

Run progress: 80,00% complete, ETA 00:00:51

Fork: 5 of 5

Iteration 1: 21,404 ms/op

Iteration 2: 20,737 ms/op

Iteration 3: 20,837 ms/op

Iteration 4: 20,860 ms/op

Iteration 5: Average Memory Usage: 11526.287128712871 bytes

Average CPU Time Used: 5.919554455445544 ms

19,560 ms/op

Result "org.example.MatrixMultiplicationBenchmarking.multiplication":

20,575 ±(99.9%) 0,537 ms/op [Average]

(min, avg, max) = (18,900, 20,575, 22,046), stdev = 0,717

CI (99.9%): [20,038, 21,112] (assumes normal distribution)

Run complete. Total time: 00:04:15

REMEMBER: The numbers below are just data. To gain reusable insights, you need to follow up on why the numbers are the way they are. Use profilers (see -prof, -lprof), design factorial experiments, perform baseline and negative tests that provide experimental control, make sure the benchmarking environment is safe on JVM/OS/HW level, ask for reviews from the domain experts. Do not assume the numbers tell you what you want them to tell.

NOTE: Current JVM experimentally supports Compiler Blackholes, and they are in use. Please exercise extra caution when trusting the results, look into the generated code to check the benchmark still works, and factor in a small probability of new VM bugs. Additionally, while comparisons between different JVMs are already problematic, the performance difference caused by different Blackhole modes can be very significant. Please make sure you use the consistent Blackhole mode for comparisons.

Benchmark	Mode	Cnt	Score	Error	Units
MatrixMultiplicationBenchmarking.multiplication	avgt	25	20,575	± 0,537	ms/op

Process finished with exit code 0

512x512

JMH version: 1.35

VM version: JDK 20.0.1, Java HotSpot(TM) 64-Bit Server VM, 20.0.1+9-29

VM invoker: C:\Program Files\Java\jdk-20\bin\java.exe

VM options: -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\lib\idea_rt.jar=64743:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8

Blackhole mode: compiler (auto-detected, use -Djmh.blackhole.autoDetect=false to disable)

Warmup: <none>

Measurement: 5 iterations, 10 s each

Timeout: 10 min per iteration

Threads: 1 thread, will synchronize iterations

Benchmark mode: Average time, time/op

Benchmark: org.example.MatrixMultiplicationBenchmarking.multiplication

Run progress: 0,00% complete, ETA 00:04:10

Fork: 1 of 5

Iteration   1: 205,821 ms/op

Iteration   2: 186,866 ms/op

Iteration   3: 187,459 ms/op

Iteration   4: 185,738 ms/op

Iteration   5: Average Memory Usage: 242571.33333333334 bytes

Average CPU Time Used: 64.68939393939394 ms

189,848 ms/op

Run progress: 20,00% complete, ETA 00:03:25

Fork: 2 of 5

Iteration   1: 191,781 ms/op

Iteration   2: 179,452 ms/op

Iteration   3: 182,256 ms/op

Iteration   4: 183,544 ms/op

Iteration   5: Average Memory Usage: 314917.63503649633 bytes

Average CPU Time Used: 69.44890510948905 ms

183,178 ms/op

Run progress: 40,00% complete, ETA 00:02:34

Fork: 3 of 5

Iteration 1: 198,320 ms/op

Iteration 2: 183,278 ms/op

Iteration 3: 187,957 ms/op

Iteration 4: 188,406 ms/op

Iteration 5: Average Memory Usage: 291849.5464684015 bytes

Average CPU Time Used: 65.10408921933086 ms

184,930 ms/op

Run progress: 60,00% complete, ETA 00:01:42

Fork: 4 of 5

Iteration 1: 204,665 ms/op

Iteration 2: 209,307 ms/op

Iteration 3: 201,105 ms/op

Iteration 4: 196,116 ms/op

Iteration 5: Average Memory Usage: 170011.93700787402 bytes

Average CPU Time Used: 61.85826771653543 ms

182,954 ms/op

Run progress: 80,00% complete, ETA 00:00:51

Fork: 5 of 5

Iteration 1: 214,771 ms/op

Iteration 2: 174,515 ms/op

Iteration 3: 173,066 ms/op

Iteration 4: 177,237 ms/op

Iteration 5: Average Memory Usage: 314737.6642335766 bytes

Average CPU Time Used: 80.04744525547446 ms

185,543 ms/op

Result "org.example.MatrixMultiplicationBenchmarking.multiplication":

189,525 ±(99.9%) 8,105 ms/op [Average]

(min, avg, max) = (173,066, 189,525, 214,771), stdev = 10,820

CI (99.9%): [181,420, 197,629] (assumes normal distribution)

Run complete. Total time: 00:04:17

REMEMBER: The numbers below are just data. To gain reusable insights, you need to follow up on why the numbers are the way they are. Use profilers (see -prof, -lprof), design factorial experiments, perform baseline and negative tests that provide experimental control, make sure the benchmarking environment is safe on JVM/OS/HW level, ask for reviews from the domain experts. Do not assume the numbers tell you what you want them to tell.

NOTE: Current JVM experimentally supports Compiler Blackholes, and they are in use. Please exercise extra caution when trusting the results, look into the generated code to check the benchmark still works, and factor in a small probability of new VM bugs. Additionally, while comparisons between different JVMs are already problematic, the performance difference caused by different Blackhole modes can be very significant. Please make sure you use the consistent Blackhole mode for comparisons.

Benchmark	Mode	Cnt	Score	Error	Units
MatrixMultiplicationBenchmarking.multiplication	avgt	25	189,525	± 8,105	ms/op

Process finished with exit code 0

1024x1024

JMH version: 1.35

VM version: JDK 20.0.1, Java HotSpot(TM) 64-Bit Server VM, 20.0.1+9-29

VM invoker: C:\Program Files\Java\jdk-20\bin\java.exe

VM options: -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1\lib\idea_rt.jar=57021:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1\bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8

Blackhole mode: compiler (auto-detected, use -Djmh.blackhole.autoDetect=false to disable)

Warmup: 2 iterations, 1 ms each

Measurement: 5 iterations, 10 s each

Timeout: 10 min per iteration

Threads: 1 thread, will synchronize iterations

Benchmark mode: Average time, time/op

Benchmark: org.example.MatrixMultiplicationBenchmarking.multiplication

Run progress: 0,00% complete, ETA 00:04:10

Fork: 1 of 5

Warmup Iteration   1: 6638,423 ms/op

Warmup Iteration   2: 6622,226 ms/op

Iteration   1: 4879,101 ms/op

Iteration   2: 4618,622 ms/op

Iteration   3: 4823,314 ms/op

Iteration   4: 4866,802 ms/op

Iteration   5: Average Memory Usage: 4226776.470588235 bytes

Average CPU Time Used: 2846.0588235294117 ms

4701,078 ms/op

Run progress: 20,00% complete, ETA 00:05:43

Fork: 2 of 5

Warmup Iteration   1: 6737,284 ms/op

Warmup Iteration   2: 7222,535 ms/op

Iteration   1: 5388,090 ms/op

Iteration   2: 5391,016 ms/op

Iteration   3: 5038,201 ms/op

Iteration   4: 5847,489 ms/op

Iteration   5: Average Memory Usage: 2404561.3333333335 bytes

Average CPU Time Used: 2584.3333333333335 ms

5357,585 ms/op

Run progress: 40,00% complete, ETA 00:03:52

Fork: 3 of 5

Warmup Iteration   1: 6701,454 ms/op

Warmup Iteration   2: 7184,065 ms/op

Iteration   1: 8083,978 ms/op

Iteration   2: 7733,547 ms/op

Iteration   3: 7972,670 ms/op

Iteration   4: 5553,719 ms/op

Iteration   5: Average Memory Usage: 2359441.3333333335 bytes

Average CPU Time Used: 2080.3333333333335 ms

5140,927 ms/op

Run progress: 60,00% complete, ETA 00:02:39

Fork: 4 of 5

Warmup Iteration   1: 7177,888 ms/op

Warmup Iteration   2: 6874,045 ms/op

Iteration   1: 5313,018 ms/op

Iteration   2: 5067,584 ms/op

Iteration   3: 5513,918 ms/op

Iteration   4: 5477,271 ms/op

Iteration   5: Average Memory Usage: 2374736.0 bytes

Average CPU Time Used: 2194.9166666666665 ms

5307,694 ms/op

Run progress: 80,00% complete, ETA 00:01:16

Fork: 5 of 5

Warmup Iteration   1: 7183,957 ms/op

Warmup Iteration   2: 7035,104 ms/op

Iteration 1: 5272,463 ms/op

Iteration 2: 5313,011 ms/op

Iteration 3: 5237,363 ms/op

Iteration 4: 5576,154 ms/op

Iteration 5: Average Memory Usage: 2371980.0 bytes

Average CPU Time Used: 3138.75 ms

5312,254 ms/op

Result "org.example.MatrixMultiplicationBenchmarking.multiplication":

5551,475 ±(99.9%) 706,348 ms/op [Average]

(min, avg, max) = (4618,622, 5551,475, 8083,978), stdev = 942,954

CI (99.9%): [4845,127, 6257,823] (assumes normal distribution)

Run complete. Total time: 00:06:15

REMEMBER: The numbers below are just data. To gain reusable insights, you need to follow up on why the numbers are the way they are. Use profilers (see -prof, -lprof), design factorial experiments, perform baseline and negative tests that provide experimental control, make sure the benchmarking environment is safe on JVM/OS/HW level, ask for reviews from the domain experts. Do not assume the numbers tell you what you want them to tell.

NOTE: Current JVM experimentally supports Compiler Blackholes, and they are in use. Please exercise extra caution when trusting the results, look into the generated code to check the benchmark still works, and factor in a small probability of new VM bugs. Additionally, while comparisons between different JVMs are already problematic, the performance difference caused by different Blackhole modes can be very significant. Please make sure you use the consistent Blackhole mode for comparisons.

Benchmark	Mode	Cnt	Score	Error	Units
MatrixMultiplicationBenchmarking.multiplication	avgt	25	5551,475	± 706,348	ms/op

Process finished with exit code 0

2048x2048

JMH version: 1.35

VM version: JDK 20.0.1, Java HotSpot(TM) 64-Bit Server VM, 20.0.1+9-29

VM invoker: C:\Program Files\Java\jdk-20\bin\java.exe

VM options: -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\lib\idea_rt.jar=64941:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.2\bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8

Blackhole mode: compiler (auto-detected, use -Djmh.blackhole.autoDetect=false to disable)

Warmup: <none>

Measurement: 5 iterations, 10 s each

Timeout: 10 min per iteration

Threads: 1 thread, will synchronize iterations

Benchmark mode: Average time, time/op

Benchmark: org.example.MatrixMultiplicationBenchmarking.multiplication

Run progress: 0,00% complete, ETA 00:04:10

Fork: 1 of 5

Iteration 1: 57005,145 ms/op

Iteration 2: 56034,454 ms/op

Iteration 3: 56222,818 ms/op

Iteration 4: 55547,010 ms/op

Iteration 5: Average Memory Usage: 2.94227872E7 bytes

Average CPU Time Used: 16680.8 ms

54442,342 ms/op

Run progress: 20,00% complete, ETA 00:18:41

Fork: 2 of 5

Iteration 1: 54556,852 ms/op

Iteration 2: 54014,732 ms/op

Iteration 3: 54680,024 ms/op

Iteration 4: 53835,004 ms/op

Iteration 5: Average Memory Usage: 2.864332E7 bytes

Average CPU Time Used: 27971.6 ms

52916,654 ms/op

Run progress: 40,00% complete, ETA 00:13:47

Fork: 3 of 5

Iteration 1: 54452,882 ms/op

Iteration 2: 54648,860 ms/op

Iteration 3: 53117,157 ms/op

Iteration 4: 53133,259 ms/op

Iteration 5: Average Memory Usage: 2.9338304E7 bytes

Average CPU Time Used: 16393.2 ms

54146,974 ms/op

Run progress: 60,00% complete, ETA 00:09:08

Fork: 4 of 5

Iteration 1: 54793,403 ms/op

Iteration 2: 52497,401 ms/op

Iteration 3: 52994,949 ms/op

Iteration 4: 53854,852 ms/op

Iteration 5: Average Memory Usage: 2.92165248E7 bytes

Average CPU Time Used: 20412.0 ms

53044,003 ms/op

Run progress: 80,00% complete, ETA 00:04:32

Fork: 5 of 5

Iteration 1: 51517,668 ms/op

Iteration 2: 52139,407 ms/op

Iteration 3: 54578,600 ms/op

Iteration 4: 54253,637 ms/op

Iteration 5: Average Memory Usage: 2.9279288E7 bytes

Average CPU Time Used: 20974.6 ms

52746,634 ms/op

Result "org.example.MatrixMultiplicationBenchmarking.multiplication":

54046,989 ±(99.9%) 981,727 ms/op [Average]

(min, avg, max) = (51517,668, 54046,989, 57005,145), stdev = 1310,578

CI (99.9%): [53065,262, 55028,716] (assumes normal distribution)

Run complete. Total time: 00:22:37

REMEMBER: The numbers below are just data. To gain reusable insights, you need to follow up on why the numbers are the way they are. Use profilers (see -prof, -lprof), design factorial experiments, perform baseline and negative tests that provide experimental control, make sure the benchmarking environment is safe on JVM/OS/HW level, ask for reviews from the domain experts. Do not assume the numbers tell you what you want them to tell.

NOTE: Current JVM experimentally supports Compiler Blackholes, and they are in use. Please exercise extra caution when trusting the results, look into the generated code to check the benchmark still works, and factor in a small probability of new VM bugs. Additionally, while comparisons between different JVMs are already problematic, the performance difference caused by different Blackhole modes can be very significant. Please make sure you use the consistent Blackhole mode for comparisons.

Benchmark	Mode	Cnt	Score	Error	Units
MatrixMultiplicationBenchmarking.multiplication	avgt	25	54046,989	± 981,727	ms/op

Process finished with exit code 0