PAI Benchmark

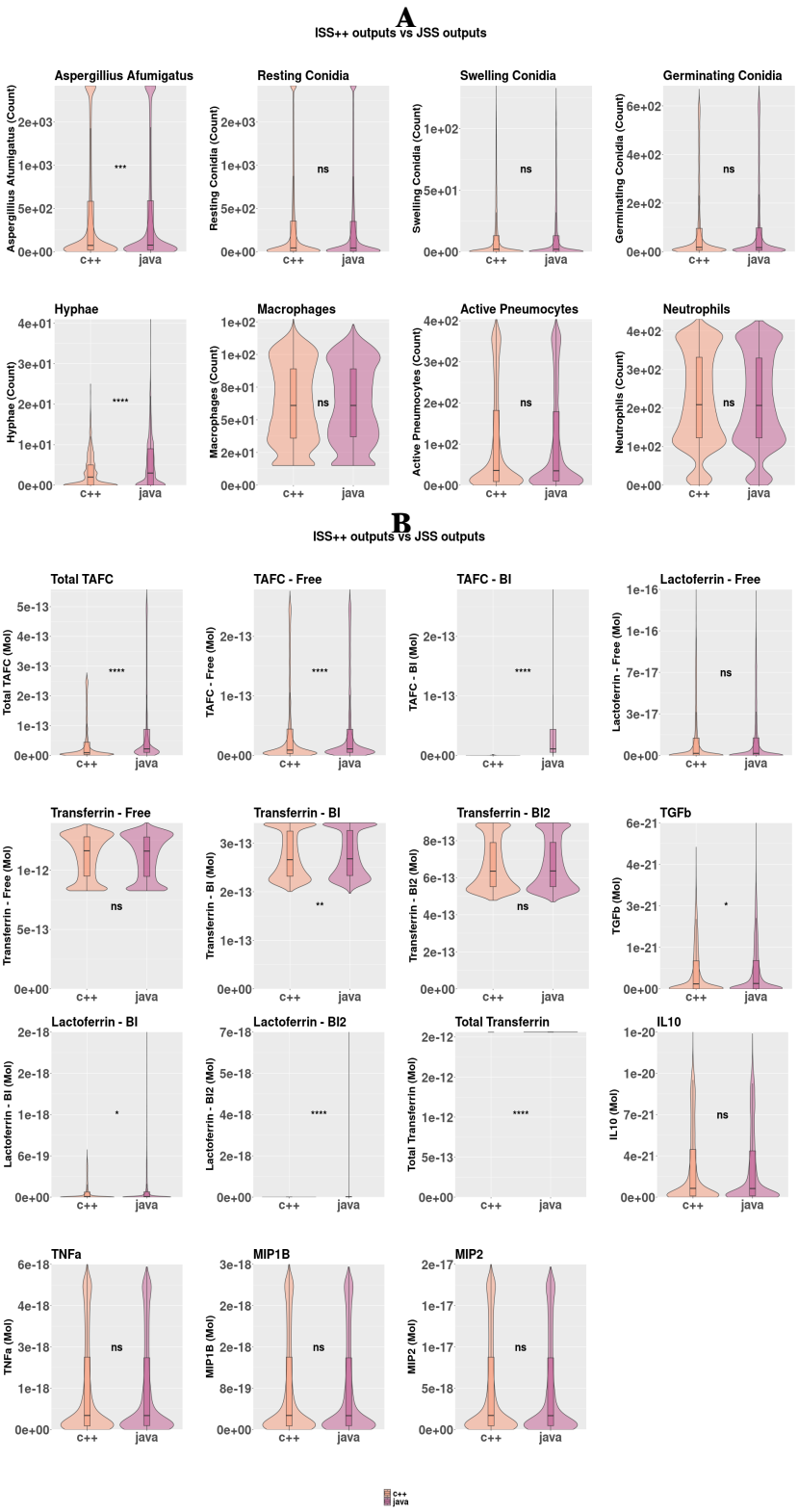

Figure S1: Violin plots comparing outputs of the C++ ABM (Orange) and Java ABM (Pink), with statistical significance of median differences assessed by the Wilcoxon rank-sum test (* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$; **** $p < 0.0001$). The ABM includes 8 agent and 15 molecule outputs. Simulations were initialized with 10×10×10 voxels, 640 pneumocytes, 1920 conidia, 15 macrophages, 209 maximum macrophages, and 522 maximum neutrophils. Each simulated hour corresponds to 30 iterations, that is, the 2160 iterations of each simulation is equivalent to the first 72 hours post infection. Except for the difference in implementation language (C++ and Java), stochastic factors, all simulations were supposed to be identical. Each simulation was run 100 times on a Windows 10 Education 64-Bit Dell laptop, Intel® Core™ Ultra 7 155U (14CPUs), ~2.1GHz, 16384MB RAM, using the MinGW toolchain with g++ 12.4.0 using level three optimization (PAI++), and Java 23.0.1 (jPAI). Each plot aggregates data from all 2160 iterations across the 100 simulations (a total of 216,000 observations). **A -** Agent cell outputs. **B -** Molecular outputs. Some of the model's molecules have multiple forms (multiple outputs). These are representing the free, bound to iron (BI), and bound to two iron ions (BI2) forms.

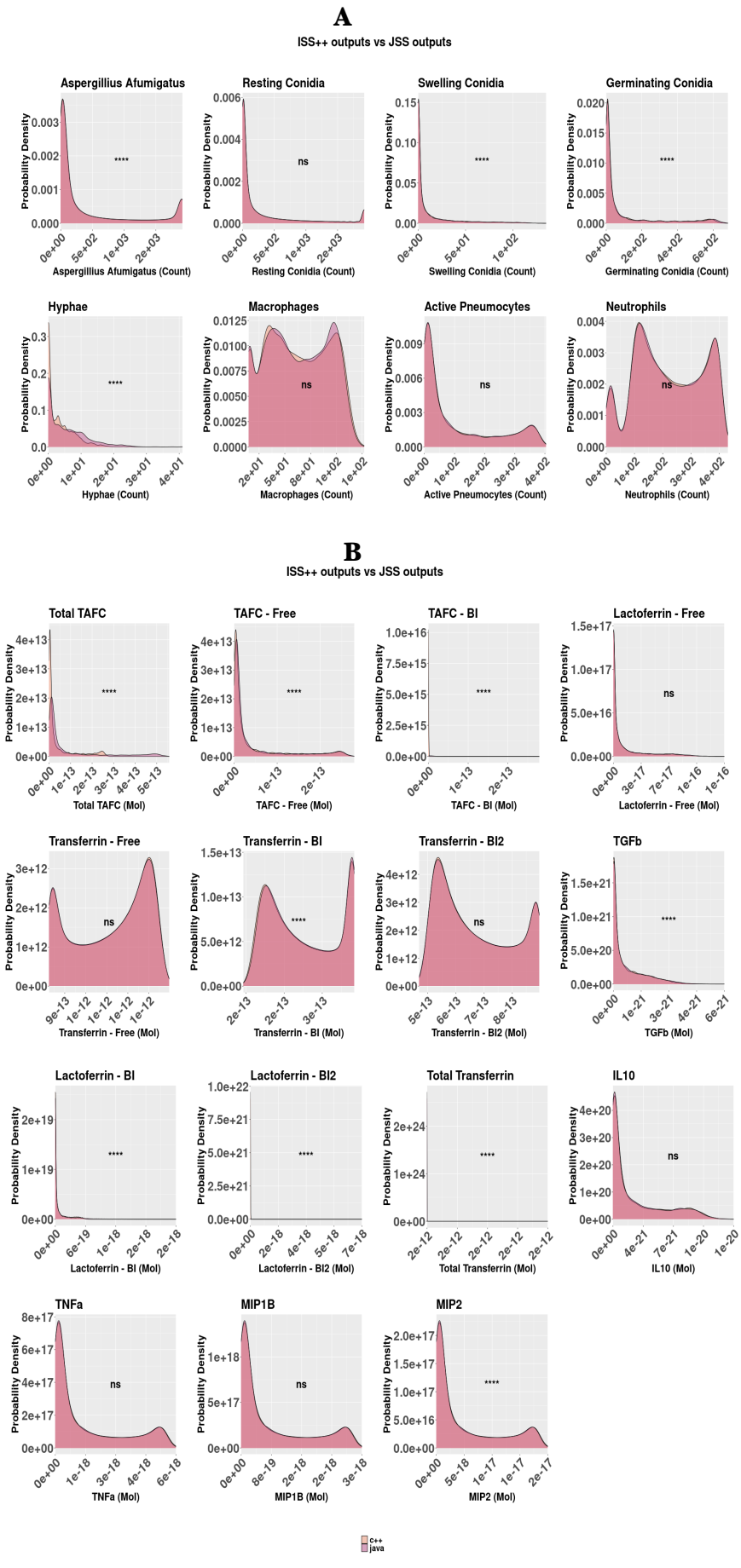

Figure S2: Density plots comparing outputs of the C++ ABM (Orange) and Java ABM (Pink), with statistical significance of distribution differences assessed by the Cramér test (* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$; **** $p < 0.0001$). The ABM includes 8 agent and 15 molecule outputs. Simulations were initialized with 10×10×10 voxels, 640 pneumocytes, 1920 conidia, 15 macrophages, 209 maximum macrophages, and 522 maximum neutrophils. Each simulated hour corresponds to 30 iterations, that is, the 2160 iterations of each simulation is equivalent to the first 72 hours post infection. Except for the difference in implementation language (C++ and Java), stochastic factors, all simulations were supposed to be identical. Each simulation was run 100 times on a Windows 10 Education 64-Bit Dell laptop, Intel® Core™ Ultra 7 155U (14CPUs), ~2.1GHz, 16384MB RAM, using the MinGW toolchain with g++ 12.4.0 using level three optimization (PAI++), and Java 23.0.1 (jPAI). Each plot aggregates data from all 2160 iterations across the 100 simulations (a total of 216,000 observations). **A -** Agent cell outputs. **B -** Molecular outputs. Some of the model's molecules have multiple forms (multiple outputs). These are representing the free, bound to iron (BI), and bound to two iron ions (BI2) forms.

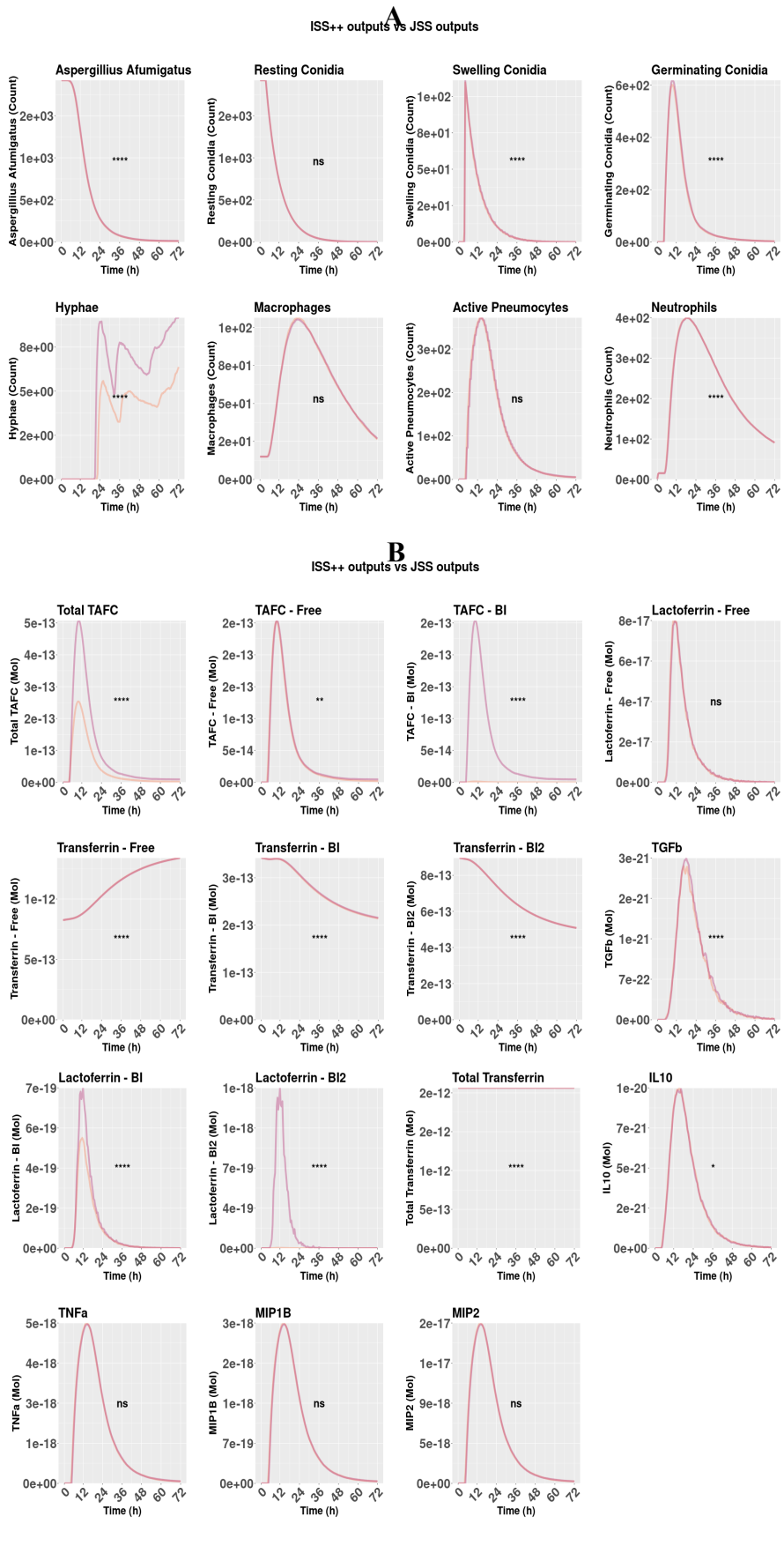

Figure S3: Time series plots comparing outputs of the C++ ABM (Orange) and Java ABM (Pink). Statistical significance of mean differences was assessed with Paired t-test. (* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$; **** $p < 0.0001$). The ABM tracks 8 agent and 15 molecule outputs. Simulations were initialized with 10×10×10 voxels, 640 pneumocytes, 1920 conidia, 15 macrophages, 209 maximum macrophages, and 522 maximum neutrophils. Each simulated hour corresponds to 30 iterations, that is, the 2160 iterations of each simulation is equivalent to the first 72 hours post infection. Except for the difference in implementation language (C++ and Java), stochastic factors, all simulations were supposed to be identical. Each simulation was run 100 times on a Windows 10 Education 64-Bit Dell laptop, Intel® Core™ Ultra 7 155U (14CPUs), ~2.1GHz, 16384MB RAM, using the MinGW toolchain with g++ 12.4.0 using level three optimization (PAI++), and Java 23.0.1 for the (jPAI). Each plot shows the average of 100 simulations. **A -** Agent cell outputs. **B -** Molecular outputs. Some of the model's molecules have multiple forms (multiple outputs). These are representing the free, bound to iron (BI), and bound to two iron ions (BI2) forms.

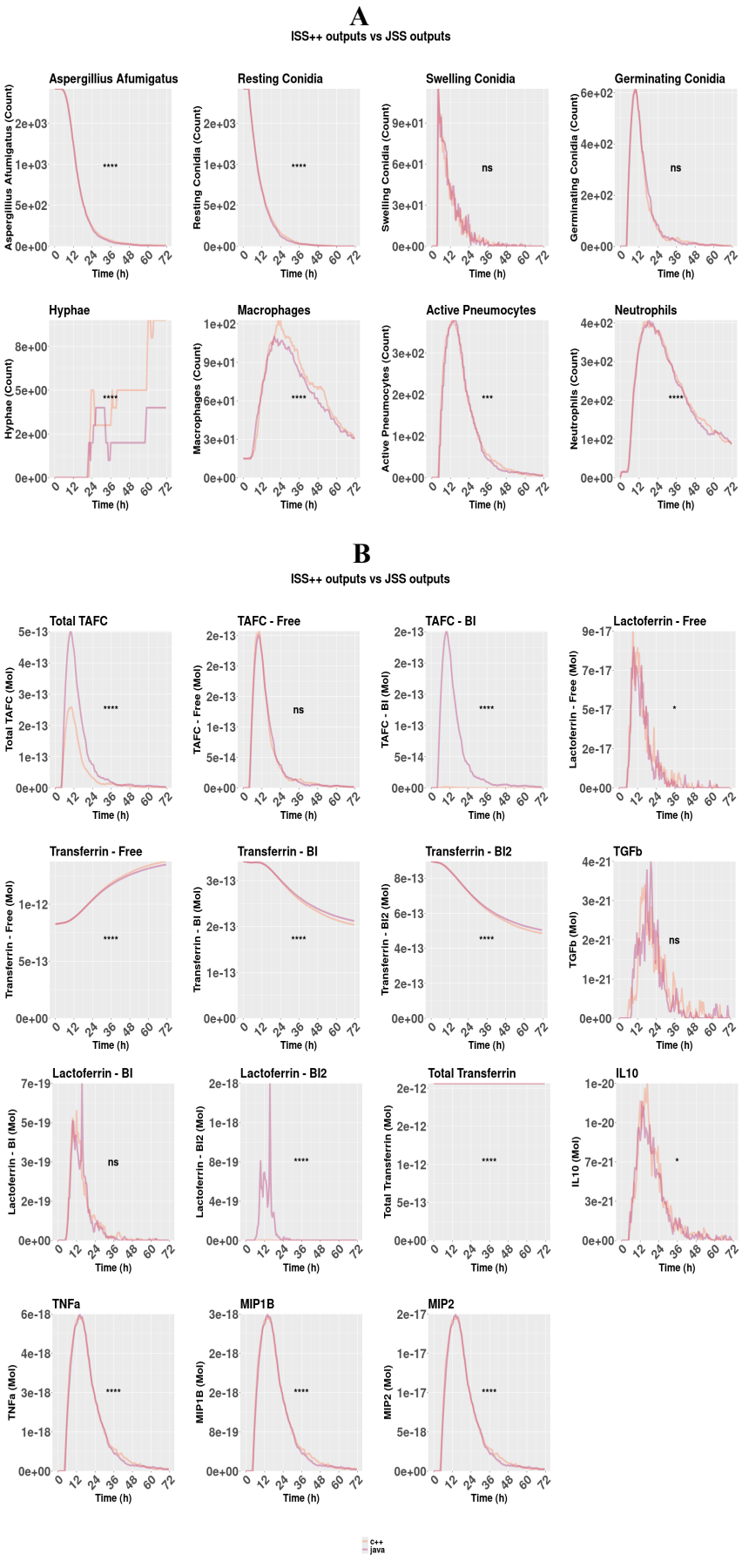

Figure S4: Time series plots comparing outputs of the C++ ABM (Orange) and Java ABM (Pink). Statistical significance of mean difference was assessed with the Paired t-test. (* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$; **** $p < 0.0001$). The ABM tracks 8 agent and 15 molecule outputs. Simulations were initialized with 10×10×10 voxels, 640 pneumocytes, 1920 conidia, 15 macrophages, 209 maximum macrophages, and 522 maximum neutrophils. Each simulated hour corresponds to 30 iterations, that is, the 2160 iterations of each simulation is equivalent to the first 72 hours post infection. Except for the difference in implementation language (C++ and Java), stochastic factors, all simulations were supposed to be identical. Each simulation was run 100 times on a Windows 10 Education 64-Bit Dell laptop, Intel® Core™ Ultra 7 155U (14CPUs), ~2.1GHz, 16384MB RAM, using the MinGW toolchain with g++ 12.4.0 using level three optimization (PAI++), and Java 23.0.1 for the (jPAI). Each plot depicts the results of a single simulation run over. **A -** Agent cell outputs. **B -** Molecular outputs. Some of the model's molecules have multiple forms (multiple outputs). These are representing the free, bound to iron (BI), and bound to two iron ions (BI2) forms.

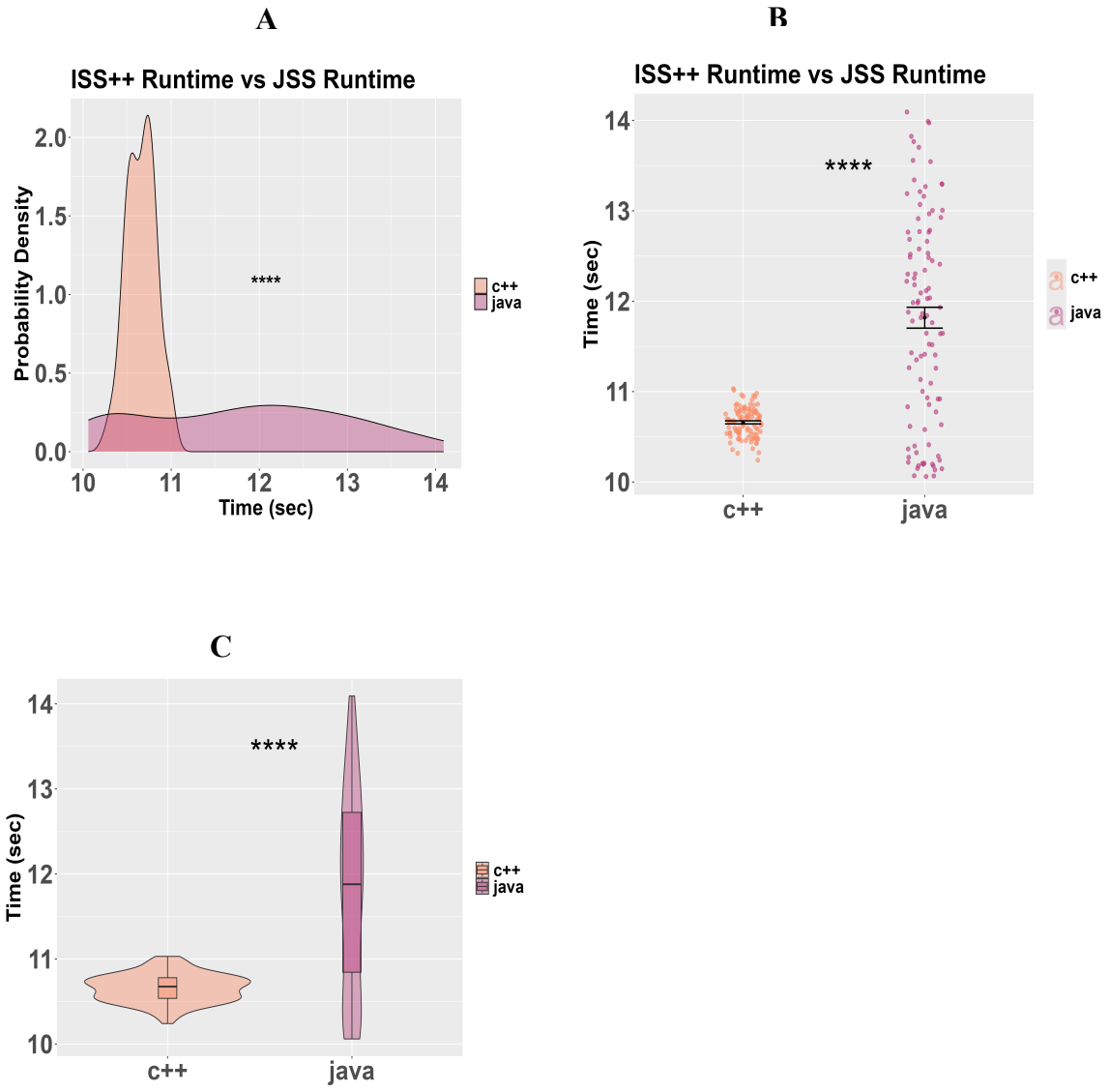

Figure S5: Runtime comparison between the C++ (Orange) and Java (Pink) implementations of the ABM. All simulations were initialized with 10×10×10 voxels, 640 pneumocytes, 1920 conidia, 15 macrophages, 209 maximum macrophages, and 522 maximum neutrophils. Each simulation runs for 2160 iterations—equivalent to 72 hours post-infection (30 iterations per hour). Simulations were run on a Windows 10 Education 64-Bit Dell laptop, Intel® Core™ Ultra 7 155U (14CPUs), ~2.1GHz, 16384MB RAM. The C++ version was compiled with g++ 12.4.0 using the MinGW toolchain with level three optimization; the Java version used Java 23.0.1. Each simulator was executed 100 times on a Windows 10 Dell laptop. **A -** Density Plots: Show the estimated probability density of runtime (in seconds) across the 100 simulation runs for each implementation. Differences in runtime distributions were tested using the Cramér. **B -** Jitter Plots: Scatterplots displaying individual runtime values (in seconds) for each of the 100 simulation runs. Median differences between simulators were assessed using the Wilcoxon rank-sum test. **C -** Violin Plots: Depict the distribution and spread of runtime values across the 100 runs for each implementation. Median comparisons were performed using the Wilcoxon rank-sum test. (* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$; **** $p < 0.0001$).

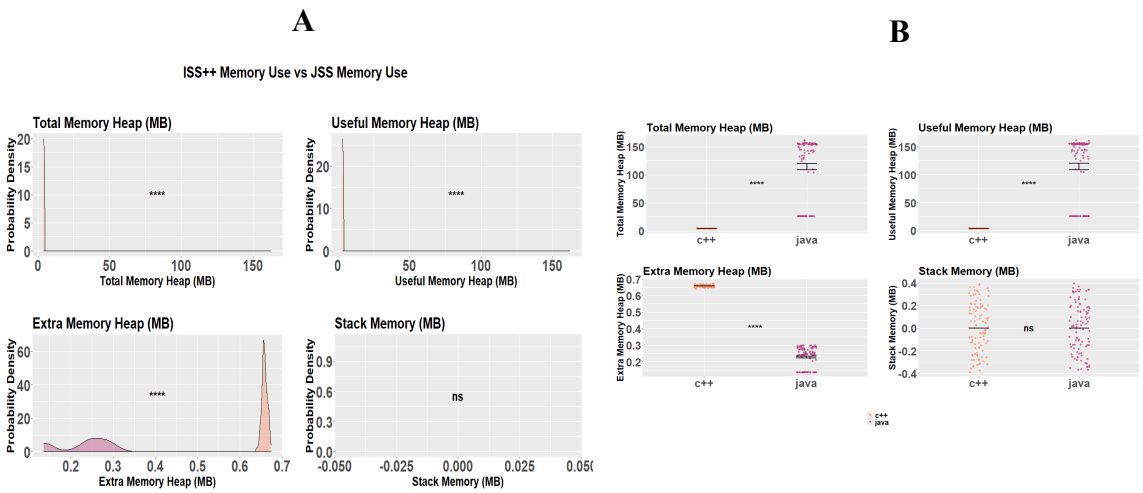

Figure S6: Comparison of memory usage between the C++ (Orange) and Java (Pink) implementations of the ABM. Simulations were initialized with 10×10×10 voxels, 640 pneumocytes, 1920 conidia, 15 macrophages, 209 maximum macrophages, and 522 maximum neutrophils. Each simulation runs for 2160 iterations—equivalent to 72 hours post-infection (30 iterations per hour). Simulations were run on a Windows 10 Education 64-Bit Dell laptop, Intel® Core™ Ultra 7 155U (14CPUs), ~2.1GHz, 16384MB RAM. The C++ version was compiled with g++ 12.4.0 using the MinGW toolchain with level three optimization; the Java version used Java 23.0.1. Memory usage for both simulators was measured using Valgrind, which recorded four types of memory metrics: total memory use, useful memory heap, extra memory heap, and stack memory. **A -** Density Plots: Show the estimated probability density of each memory value across the 100 simulation runs for each implementation. Differences in each memory metric distribution were tested using the Cramér test. **B.** Jitter Plots: displaying the maximum memory used (in MB) for each of the 100 simulation runs. Median differences between simulators were evaluated using the Wilcoxon rank-sum test. (* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$; **** $p < 0.0001$).

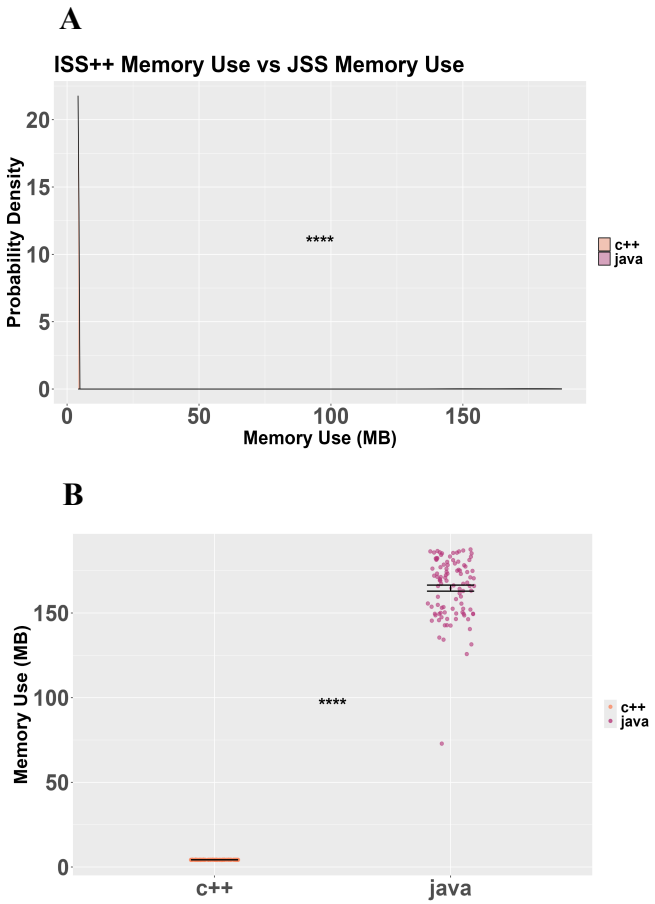

Figure S7: Comparison of memory usage between the C++ (Orange) and Java (Pink) implementations of the ABM. Simulations were initialized with identical parameters: 10×10×10 voxels, 640 pneumocytes, 1920 conidia, 15 macrophages, 209 maximum macrophages, and 522 maximum neutrophils. Each simulation ran for 2160 iterations, representing 72 hours post-infection (30 iterations per simulated hour). Simulations were executed on a Windows Education 64-Bit Dell laptop, Intel® Core™ Ultra 7 155U (14CPUs), ~2.1GHz, 16384MB RAM. The C++ implementation used the MinGW toolchain with the g++ 12.4.0 compiler using level three optimization; the Java version used Java 23.0.1. Memory usage for the C++ simulations was recorded using the Valgrind tool, while JProfiler (potentially more adequate for Java) was used for the Java simulations. **A -** Density Plots: Show the estimated probability density of the Memory usage (in megabytes, MB) across the 100 simulation runs for each implementation. Differences in each memory metric distribution were tested using the Cramér test. **B.** Jitter Plots: displaying the maximum memory used (in MB) for each of the 100 simulation runs. Median differences between simulators were evaluated using the Wilcoxon rank-sum test. (* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$; **** $p < 0.0001$).

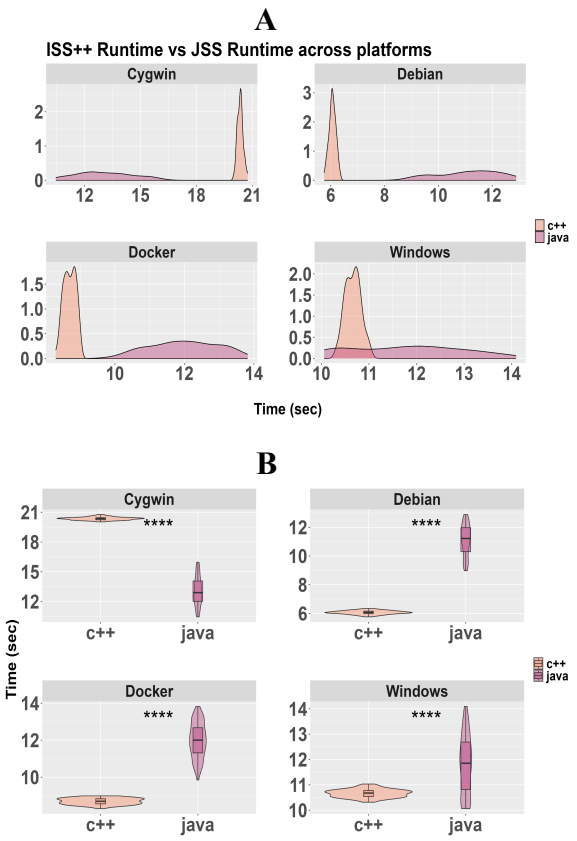

Figure S8: Runtime comparison between the C++ (Orange) and Java (Pink) implementations of the agent-based model (ABM) across four environments: Windows, Cygwin, Debian (via Windows Subsystem Linux – Maroon), and Docker with an Ubuntu 22.04 image. All those were run into run on Windows 10 Education 64-Bit Dell laptop, Intel® Core™ Ultra 7 155U (14CPUs), ~2.1GHz, 16384MB RAM. Each simulation was run 100 times with identical initial conditions: 10×10×10 voxels, 640 pneumocytes, 1920 conidia, 15 macrophages, 209 maximum macrophages, 522 maximum neutrophils, and 2160 iterations. Each simulated hour corresponds to 30 iterations, that is, the 2160 iterations of each simulation is equivalent to the first 72 hours post infection. The C++ simulator was compiled with g++ 14.2.0 using level three optimization for the MinGW, Cygwin and Docker. While for Debian we used g++ 12.2.0 with level three optimization. The Java version was executed using java 23.0.1 for the MinGW, Cygwin and Docker. While for Debian OpenJDK 17.014 was used. **A** –Density plots of runtime (in Seconds), displaying the likelihood distribution of runtimes for each platform. **B -** Violin plots comparing runtime distributions, with statistical significance assessing median differences via the Wilcoxon rank-sum test. (* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$; **** $p < 0.0001$).