| Name | Short Description | Time Complexity | Detailed Explanation | Benchmark Description |
|---|---|---|---|---|
| 2mm | 2 Matrix Multiplications | O(N^3) | Performs two matrix multiplications and then adds the result of the second multiplication to the result of the first. | 2mm 2 Matrix Multiplications (E=A.B; F=E.C; G=F+D) |
| 3mm | 3 Matrix Multiplications | O(N^3) | Performs three matrix multiplications in a specific order. | 3mm 3 Matrix Multiplications (E=A.B; F=C.D; G=E.F) |
| adi | Alternating Direction Implicit solver | O(N^3) | Solves a linear system of equations using the Alternating Direction Implicit (ADI) method. | adi Alternating Direction Implicit solver |
| atax | Matrix Transpose and Vector Multiplication | O(N^2) | Calculates the product of a matrix and a vector, where the matrix is first transposed. | atax Matrix Transpose and Vector Multiplication |
| bicg | BiCG Sub Kernel of BiCGStab Linear Solver | O(N^2) | Performs a kernel of the BiConjugate Gradient Stabilized (BiCGStab) linear solver. | bicg BiCG Sub Kernel of BiCGStab Linear Solver |
| cholesky | Cholesky Decomposition | O(N^3) | Decomposes a symmetric, positive-definite matrix into a product of a lower triangular matrix and its transpose. | cholesky Cholesky Decomposition |
| correlation | Correlation Computation | O(N^3) | Calculates the correlation coefficient between pairs of vectors. | correlation Correlation Computation |
| covariance | Covariance Computation | O(N^3) | Calculates the covariance matrix for a set of vectors. | covariance Covariance Computation |
| deriche | Edge detection filter | O(N^2) | Applies an edge detection filter to an image. | deriche Edge detection filter |
| doitgen | Multi-resolution Analysis Kernel | O(N^3) | Performs a multi-resolution analysis using a specific kernel. | doitgen Multi-resolution analysis kernel (MADNESS) |
| durbin | Toeplitz system solver | O(N^2) | Solves a Toeplitz system of equations using the Durbin recursion algorithm. | durbin Toeplitz system solver |
| fdtd-2d | 2-D Finite Different Time Domain Kernel | O(N^3) | Simulates a 2D electromagnetic wave using the finite difference time domain method. | fdtd-2d 2-D Finite Different Time Domain Kernel |
| floyd-warshall | All Pairs Shortest Path | O(N^3) | Given a graph with N vertices and edge weights, find the shortest path between all pairs of vertices. This algorithm uses dynamic programming to iteratively update a matrix of shortest distances between pairs of vertices. The time complexity is O(N^3) due to the three nested loops used for the dynamic programming updates. | |
| gemm | Matrix-Multiply | O(N^3) | Computes the product of two matrices. | gemm Matrix-multiply C=alpha.A.B+beta.C |
| gemver | Vector Multiplication and Matrix Addition | O(N^2) | Calculates the product of a matrix and a vector, then adds the resulting vector to another vector. | gemver Vector Multiplication and Matrix Addition |
| gesummv | Scalar, Vector and Matrix Multiplication | O(N^2) | Performs a combination of scalar, vector and matrix multiplication. | gesummv Scalar, Vector and Matrix Multiplication |
| gramschmidt | Gram-Schmidt Decomposition | O(N^3) | Calculates the Gram-Schmidt orthogonalization of a set of vectors. | gramschmidt Gram-Schmidt decomposition |
| head-3d | Heat Equation Over 3D Data Domain | O(N^3) | Simulates the heat equation over a 3D domain using the finite difference method. | head-3d Heat equation over 3D data domain |
| jacobi-1D | 1-D Jacobi Stencil Computation | O(N^2) | Calculates the Jacobi stencil for a 1D grid of points. | jacobi-1D 1-D Jacobi stencil computation |
| jacobi-2D | 2-D Jacobi Stencil Computation | O(N^2) | Calculates the Jacobi stencil for a 2D grid of points. | jacobi-2D 2-D Jacobi stencil computation |
| lu | LU Decomposition | O(N^3) | Decomposes a square matrix into a lower triangular matrix and an upper triangular matrix. | lu LU decomposition |

| | | | | |
|---|---|---|---|---|
| ludcmp | LU Decomposition Followed by Forward and Back Substitution | O(N^3) | Decomposes a matrix into a lower triangular matrix and an upper triangular matrix, then solves a system of linear equations using forward and back substitution. | ludcmp LU decomposition followed by FS |
| mvt | Matrix Vector Product and Transpose | O(N^2) | Given a matrix A of size N x N and two vectors x and y of size N, this benchmark computes y = Ax and x = A^Ty. This involves N^2 operations. | mvt Matrix Vector Product and Transpose |
| nussinov | Dynamic programming algorithm for sequence alignment | O(N^3) | This benchmark solves the RNA secondary structure prediction problem using the Nussinov dynamic programming algorithm. Given a sequence of RNA nucleotides, the algorithm finds the optimal pairing of nucleotides that maximizes the number of base pairs. The algorithm has a time complexity of O(N^3) where N is the length of the sequence. | nussinov Dynamic programming algorithm for sequence alignment |
| seidel | 2-D Seidel stencil computation | O(N^2) | Given a 2-D grid of size N x N and a stencil, this benchmark computes a new grid by applying the stencil iteratively until convergence. The stencil is applied by computing the weighted sum of neighboring grid values. The computation for each grid element takes O(1) time and the algorithm requires O(N^2) iterations, giving a total time complexity of O(N^2). | seidel 2-D Seidel stencil computation |
| symm | Symmetric matrix-multiply | O(N^3) | Given two symmetric matrices A and B of size N x N, this benchmark computes the product C = AB + BA using a symmetric matrix-multiply algorithm. The algorithm requires O(N^3) operations. | symm Symmetric matrix-multiply |
| syr2k | Symmetric rank-2k operations | O(N^3) | Given two matrices A and B of size N x K, this benchmark computes the product C = alphaAA^T + betaBB^T using a symmetric rank-2k algorithm. The algorithm requires O(N^2*K) operations. | syr2k Symmetric rank-2k operations |
| syrk | Symmetric rank-k operations | O(N^3) | Given a matrix A of size N x K, this benchmark computes the product C = alphaA^TA + betaC using a symmetric rank-k algorithm. The algorithm requires O(N^2K + N*K^2) operations. | syrk Symmetric rank-k operations |
| trisolv | Triangular solver | O(N^2) | Given a triangular matrix A and a vector b of size N, this benchmark solves the system Ax = b for x using a forward or backward substitution algorithm. The algorithm requires O(N^2) operations. | trisolv Triangular solver |
| trmm | Triangular matrix-multiply | O(N^3) | Given a triangular matrix A and a matrix B of size N x K, this benchmark computes the product C = AB or C = BA using a triangular matrix-multiply algorithm. The algorithm requires O(N^3) operations. | trmm Triangular matrix-multiply |