Výpočetní jádra

SciMark provádí následující úlohy:

- **Fast Fourier Transform (FFT)** performs a one-dimensional forward transform of 4K complex numbers. This kernel exercises complex arithmetic, shuffling, non-constant memory references and trigonometric functions. The first section performs the bit-reversal portion (no flops) and the second performs the actual Nlog(N) computational steps.

  The data size for the LARGE version of the benchmark is $2^20 = 1048576$ complex numbers.

- **Jacobi Successive Over-relaxation (SOR)** on a 100x100 grid exercises typical access patterns in finite difference applications, for example, solving Laplace's equation in 2D with Drichlet boundary conditions. The algorithm exercises basic "grid averaging" memory patterns, where each $A(i,j)$ is assigned an average weighting of its four nearest neighbors.

  The inner loops of the kernel look like
  
  ```c
  for (int i=1; i < Mm1; i++)
  {
    double[] Gi = G[i];
    double[] Gim1 = G[i-1];
    double[] Gip1 = G[i+1];
    for (int j=1; j < Nm1; j++)
      Gi[j] = omega_over_four * (Gim1[j] + Gip1[j] + Gi[j-1] + Gi[j+1]) + one_minus_omega * Gi[j];
  }
  ```

  Note that we do some hand-optimizing by aliasing the rows of $G[][]$ to streamline the array accesses in the update expression.

  The data size for the LARGE version of the benchmark uses a 1,000x1,000 grid.

- **Monte Carlo integration** approximates the value of Pi by computing the integral of the quarter circle $y = \sqrt{1 - x^2}$ on $[0,1]$. It chooses random points with the unit square and compute the ratio of those within the circle. The algorithm exercises random-number generators, synchronized function calls, and function inlining.

  (Note that this kernel uses only scalars, hence the LARGE version of the benchmark is identical.)

- **Sparse matrix multiply** uses an unstructured sparse matrix stored in compressed-row format with a prescribed sparsity structure. This kernel exercises indirection addressing and non-regular memory references. A 1,000 x 1,000 sparse matrix with 5,000 nonzeros is used, with the following storage pattern:

  ```
  * * * * *  *
  * * * * *  *
  * * * * *  *
  * * * * *  *
  * * * * *  *
  ```
That is, each row has approximately 5 nonzeros, evenly spaced between the first column and the diagonal.

The data size for the LARGE version of the benchmark uses a 100,000 x 100,000 matrix with 1,000,000 nonzeros.

- **dense LU matrix factorization** Computes the LU factorization of a dense 100x100 matrix using partial pivoting. Exercises linear algebra kernels (BLAS) and dense matrix operations. The algorithm is the right-looking version of LU with rank-1 updates.

The data size for the LARGE version of the benchmark uses a 1,000 x 1,000 matrix.