AMD **Accelerated** Parallel Processing TECHNOLOGY

SAMPLE

SPMV

1 Overview

1.1 Location

`$<APPSDKSamplesInstallPath>\samples\C++Amp\`

1.2 How to Run

See the *Getting Started* guide for how to build samples. You first must compile the sample.

Use the command line to change to the directory where the executable is located. The default executables are placed in `$<APPSDKSamplesInstallPath>\samples\C++Amp\bin\x86\` for 32-bit builds, and `$<APPSDKSamplesInstallPath>\samples\C++Amp\bin\x86_64\` for 64-bit builds.

Type the following command(s).

1. `SPMV`

   This runs the program with the default options `-l 16384`

2. `SPMV -h`

   This prints the help file.

Ensure Microsoft® Visual Studio® 2012 or higher is installed.

1.3 Command Line Options

Table 1 lists, and briefly describes, the command line options.

Table 1 Command Line Options

Short Form	Long Form	Description
`-h`	`--help`	Show all command options and their respective meaning.
`-q`	`--quiet`	Quiet mode. Suppresses text output.
`-e`	`--verify`	Verify results against reference implementation.
`-t`	`--timing`	Print timing-related statistics.
`-v`	`--version`	AMD APP SDK version string.
`-d`	`--deviceId`	Select deviceId to be used (0 to N-1, where N is the number of available devices).
`-i`	`--iterations`	Number of times to repeat each algorithm.
`-V`	`--array_view`	Use array_view instead of array.
`-l`	`--length`	Length of row/column of square matrix.
`-p`	`--ellpackr`	Use Ellpackr format instead of Compressed Sparse Row (CSR) format.

2 Introduction

This sample computes large sparse matrix-vector multiplication on the GPU in two different storage formats: CSR or ELLPACKR.

3 Implementation Details

Sparse matrix usually refers to a class of matrices, the majority of their position are zeros. For Sparse matrix, in order to save storage space, we usually save only those non-zero elements, and record their position, we do not record the position of the elements that are zeros.

The CSR (Compressed Sparse Row) format is shown in Figure 1. The VAL array stores all the non-zero elements of the matrix. The J array has the same number of elements as the VAL array, and it stores the column index of the corresponding element in VAL. The I array stores the index in array J where a new row starts.

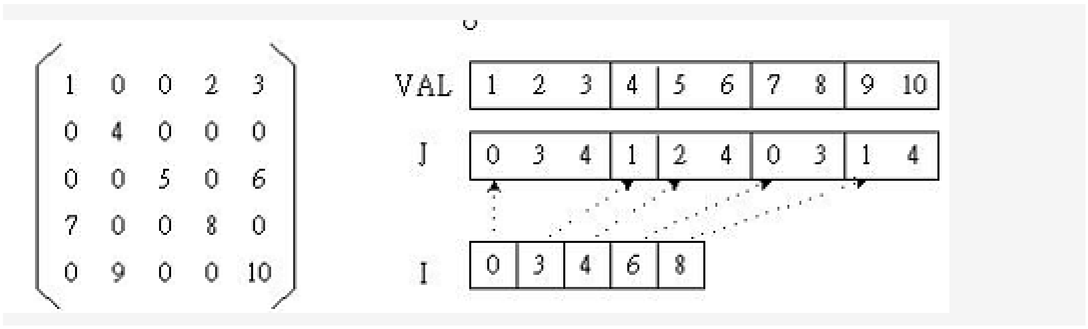

Figure 1 CSR Format

We can assign the computation of each row to a thread, so the total number of thread is equal to the number of rows.

The SPMV algorithm of CSR format is composed of:

1. Compute the matrix row our thread is responsible for.
2. Compute start/end indices of our matrix row in the sparse value array.
3. Build a partial sum for the row, striding by cCSRNThreadsPerRow.

```
for(int i = colStart + localIdxInRow; i < colEnd; i += cCSRNThreadsPerRow)
{
    thrPartialSum += csrVals_view[i] * inVector_view[csrCols_view[i]];
}
```

4. Reduce partial sums across the tile. The code implementation is as follow:

```
if(localIdxInRow < 16) partialSums[idx.local[0]] += partialSums[idx.local[0] + 16];
idx.barrier.wait_with_tile_static_memory_fence();
if(localIdxInRow < 8) partialSums[idx.local[0]] += partialSums[idx.local[0] +  8];
idx.barrier.wait_with_tile_static_memory_fence();
if(localIdxInRow < 4) partialSums[idx.local[0]] += partialSums[idx.local[0] +  4];
idx.barrier.wait_with_tile_static_memory_fence();
if(localIdxInRow < 2) partialSums[idx.local[0]] += partialSums[idx.local[0] +  2];
idx.barrier.wait_with_tile_static_memory_fence();
if(localIdxInRow == 0)
{
    outVector[rowIdx] = partialSums[idx.local[0]] + partialSums[idx.local[0] + 1];
}
```

ELLPACKR (Ellpack-Itpack) format uses a value matrix, V, that stores all non-zero values in a matrix, and each row is zero-padded to the same size as the row with the maximum number of non-zero elements. The row index in the V matrix is the same as in the original sparse matrix. The column index of each value is stored in the matrix J. The significant computation is:

```
for(int i = 0; i < rowLength; ++i)
{
     const int offset = i * denseEdge + idx[0];
     sum += ellpackrVals[offset] * inVector[ellpackrCols[offset]];
}
```

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For AMD Accelerated Parallel Processing:

URL:	**developer.amd.com/appsdk**
Developing:	**developer.amd.com/**
Forum:	**developer.amd.com/openclforum**