spmvELLRT: A library for efficient sparse matrix vector product on GPUs

F. Vazquez, G. Ortega, J.J. Fernandez, E.M. Garzon
Dpt Computer Architecture. Almeria University
Cra Sacramento s/n Almeria 04120 Spain.
Contact: f.vazquez@ual.es ; gmartin@ual.es

Description:

Sparse matrices are involved in linear systems, eigensystems and partial differential equations from a wide spectrum of scientific and engineering disciplines. Hence, sparse matrix vector product (SpMV) is considered as key operation in engineering and scientific computing. For these applications the optimization of the sparse matrix vector product (SpMV) is very relevant. However, the irregular computation involved in SpMV prevents the optimum exploitation of computational architectures when the sparse matrices are very large. Graphics Processing Units (GPUs) have recently emerged as platforms that yield outstanding acceleration factors. SpMV implementations for GPUs have already appeared on the scene. Recently a new, efficient format for SpMV for GPUs has successfully been introduced and evaluated (ELLRT). This format is based on the format ELLPACKR, which also derives from the well known ELLPACK and allows storage of the sparse matrix in a regular manner. In general, the new format turns out to outperform other formats previously used in scientific computing.

The library *spmvELLRT* implements efficient sparse matrix vector product on GPUs based on the new approach described in:

> Improving the performance of the sparse matrix vector product with GPUs.
> F. Vazquez, G. Ortega, J.J. Fernandez, E.M. Garzon
> Procs. 10th IEEE Intl. Conf. Computer and Information Technology (CIT 2010),
> pp: 1146-1151, 2010.

Please, cite this article if you use *spmvELLRT* in your work.

Contents of the package:

The package spmvELLRT.zip contains the following files needed for you to take full advantage of our efficient approach for sparse matrix vector product:

> spmvELLRT.h: Header file with the prototype of the function provided for public use.
> libspmvELLRT.a: Library to be linked statically with your program (linux, 64 bits)
>
> spmvELLRT.pdf: This documentation.

In addition, the following files are also provided as illustrative examples:

> dw2048.mtx: Example Matrix (in matrix market format,
> see http://math.nist.gov/MatrixMarket/)
> spmvELLRT.cu: Example program that illustrates the use of the library.

Description of the function for efficient sparse matrix vector product

The file *spmvELLRT.h* contains the prototype of the function **spmv_ELLRT(...)** and the library *libspmvELLRT.a* contains the binary code:

```
void   spmv_ELLRT(  float *d_BA, float *d_v, float *d_u, unsigned int *d_BIndex,
                    unsigned int *d_BCol, int nrows, int alignment,
                    int blocksize, int threads, int texture)
```

where:

d_BA: Input Matrix in format ELLRT
d_v: Input Vector
d_u: Output Result
d_BIndex: Index columns of the matrix in format ELLRT
d_BCol: Length of each row of the input matrix (i.e. No. Non-zeroes per row)
nrows: No. rows
alignment: Length (in bytes) of each row in the input matrix. It has to be multiple of 64 bytes to make the matrix stored in aligned segments. (64 bytes = 4 bytes/float * 16 threads half-warp). The calculation of this value is simple and can be seen in the example program provided in the package, alignment = (ceil(nrows*threads/64))*64
blocksize: Size of the block, in threads.
threads: Parameter T of the algorithm (see the publication). No. Threads per row.
texture: 0/1 . Use of the texture cache (1) or not (0).

Example of a main program:

The program *spmvELLRT.cu* is an example that illustrates the use of the function provided in the library. The usage is:

spmvELLRT matrix blocksize threads device texture

where:

matrix: File containing the input matrix in matrix market format,
see http://math.nist.gov/MatrixMarket/
blocksize: Size of the block, in threads.
threads: Parameter T of the algorithm (see the publication). No. Threads per row.
device: ID of the GPU to use –of all GPUs available-
texture: Use of the texture cache (1) or not (0).

Example: spmvELLRT cant.mtx 256 1 0 1

On console, the program outputs:

```
Sym: 1. Pattern: 0. Nrows: 62451. Ncols: 62451. Entries: 2034917

***
Algorithm: ELLRT. T = 1. BS = 256. Alignment = 62451
Memory: 37.8866 MB. Matrix name: cant.mtx. NonZeros: 4007384 62451x62451
***
```