PDC Summer School 2015

Examples of CUDA Libraries

2015-08-21

Michael Schliephake

KTH – CSC – HPCViz

Thrust

- Parallel algorithms and data structures (STL-like), (https://developer.nvidia.com/Thrust)
- Also useful for applications using multi-core processors (OpenMP, TBB)

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/generate.h>
#include <thrust/sort.h>
#include <thrust/copy.h>
#include <cstdlib>

int main(void)
{
    // generate 32M random numbers on the host
    thrust::host_vector<int> h_vec(32 << 20);
    thrust::generate(h_vec.begin(), h_vec.end(), rand);

    // transfer data to the device
    thrust::device_vector<int> d_vec = h_vec;

    // sort data on the device (846M keys per second on GeForce GTX 480)
    thrust::sort(d_vec.begin(), d_vec.end());

    // transfer data back to host
    thrust::copy(d_vec.begin(), d_vec.end(), h_vec.begin());

    return 0;
}
```

BLAS - Overview

- Basic Linear Algebra Subprograms
  - Computational basic operations for numerical linear algebra
  - Starting point for high level routines
  - Default implementation in Netlib
    http://netlib.org/blas/
  - Optimized implementations by processor or compiler manufacturers
    - Intel (MKL library)
    - AMD (ACML library)
    - …

BLAS – Levels of Operations

- Level 1: scalar and vector operations
  - $y \leftarrow \alpha x + y, \alpha \leftarrow x^T y, \alpha \leftarrow \|x\|_2$
- Level 2: matrix-vector operations
  - $y \leftarrow \alpha A x + \beta y, x \leftarrow T^{-1} x$
- Level 3: matrix-matrix operations
  - $C \leftarrow \alpha A B + \beta C, X \leftarrow \alpha T^{-1} X$

BLAS – Surface-to-Volume effect

BLAS level	# Memory accesses	# Floating point operations
Level 1	O(n)	O(n)
Level 2	$O(n^2)$	$O(n^2)$
Level 3	$O(n^2)$	$O(n^3)$

BLAS - Usage

FUNCTION	xDOTU (N, X, INCX, Y, INCY)	$dot \leftarrow x^T y$
FUNCTION	xDOTC (N, X, INCX, Y, INCY)	$dot \leftarrow x^H y$
FUNCTION	xxDOT (N, X, INCX, Y, INCY)	$dot \leftarrow \alpha + x^T y$
FUNCTION	xNRM2 (N, X, INCX)	$nrm2 \leftarrow \|\|x\|\|_2$
FUNCTION	xASUM (N, X, INCX)	$asum \leftarrow \|\|re(x)\|\|_1 + \|\|im(x)$
FUNCTION	IxAMAX(N, X, INCX)	$amax \leftarrow 1^{st} k \ni \|re(x_k)\| + \|$
		$= max(\|re(x_i$

Level 2 BLAS

	options	dim	b-width	scalar	matrix	vector	scalar	vector	
xGEMV (	TRANS,	M, N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x +$
xGBMV (	TRANS,	M, N,	KL, KU,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x +$
xHEMV (	UPLO,	N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$
xHBMV (	UPLO,	N,	K,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$
xHPMV (	UPLO,	N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$
xSYMV (	UPLO,	N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$
xSBMV (	UPLO,	N,	K,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$
xSPMV (	UPLO,	N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$
xTRMV (	UPLO, TRANS, DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H$
xTBMV (	UPLO, TRANS, DIAG,	N,	K,		A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H$
xTPMV (	UPLO, TRANS, DIAG,	N,							$\leftarrow A^T x, x \leftarrow A^H$
xTRSV (	UPLO, TRANS, DIAG,	N,							$x, x \leftarrow A^{-T} x, x \leftarrow$
xTBSV (	UPLO, TRANS, DIAG,	N,	K,						$x, x \leftarrow A^{-T} x, x \leftarrow$
xTPSV (	UPLO, TRANS, DIAG,	N,							$x, x \leftarrow A^{-T} x, x \leftarrow$
	options	dim	scal						
xGER (		M, N,	ALPH						$^T + A, A - m \times n$

Level 2 and Level 3 BLAS

Matrix types:

GE - GEneral	GB - General Band	
SY - SYmmetric	SB - Sym. Band	SP - Sum. Packed
HE - HErmitian	HB - Herm. Band	HP - Herm. Packed
TR - TRiangular	TB - Triang. Band	TP - Triang. Packed

BLAS - Usage

- Quick reference:
  http://www.netlib.org/blas/blasqr.pdf
- Prefixes
  - 'S', 'D', 'C', 'Z'
  - Extended precisions: 'ES', 'ED', 'EC', 'EZ'
- Matrix types
  - 'GE', 'GB', 'SY', 'SB', 'SP'
  - 'HE', 'HB', 'HP', 'TR', 'TB', 'TP'
- Options
  - TRANx, UPLO, DIAG, SIDE

BLAS - History

- Developed over several years:
  - C L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh: Basic Linear Algebra Subprograms for FORTRAN usage.
    ACM Trans. Math. Software, 5:308-323, 1979.
  - J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson: An extended set of FORTRAN Basic Linear Algebra Subprograms.
    ACM Trans. Math. Software, 14:1-32, 399, 1988. (Algorithm 656).
  - J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling: A set of Level 3 Basic Linear Algebra Subprograms.
    ACM Trans. Math. Software, 16:1-28, 1990. (Algorithm 679).
  - L. S. Blackford, J. Demmel, J. Dongarra, I. S. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington & R. C. Whaley: An Updated Set of Basic Linear Algebra Subprograms (BLAS).
    ACM Trans. Math. Software, 28:135-151, 2002.

cuBLAS

- BLAS on top of CUDA
- No interaction with driver required
- Usage model for cuBLAS API
  - Create vectors and matrices in GPU
  - Fill objects with data
  - Call CUBLAS functions
  - Download result
- When using CUBLASXT XT, user manages only host memory

LAPACK - Overview

- Linear Algebra PACKage
  - Linear systems
  - Linear least square problems
  - Eigenvalue and singular value problems
  - Matrix factorizations
  - Condition and error estimates
- Implementation based on BLAS and block-partitioned algorithms
  - Portability
  - Performance
  - Also multi-threaded implementations available

LAPACK - Overview

- Structure
  - Driver routines
    - Solve a complete problem using computational routines
  - Computational routines
    - Solve a single computational task
    - Use when driver not appropriate
  - Auxiliary routines
    - Some extensions to BLAS
    - Common low-level operations
    - Block operations
- Matrix types
  - Dense, banded
  - Numbers real or complex
  - Precision single or double

MAGMA - Overview

- LAPACK-like, new generation for heterogenous systems
- Combines different approaches to deal with massive parallelism and CPU-GPU communication
- Designed in way that allows easy portation of LAPACK-based codes
- Some important BLAS routines implemented by MAGMA itself
- http://icl.cs.utk.edu/magma/index.html

MAGMA - Overview

- Usage model
  - CPU interface: interface for input data and results on CPU's memory
  - GPU interface: interface for input data and results on GPU's memory
- Available on the web

  `http://icl.cs.utk.edu/magma/software/`

LAPACK - Documentation

- Availability
  - http://www.netlib.org/lapack/
- Manufacturer-provided
  - For example MKL, AMCL, NAG
- Users' Guide
  - http://www.netlib.org/lapack/lug/
- Quick Reference
  - http://www.netlib.org/lapack/lapackqref.ps

Library-based CG algoritm

```
Letting initial guess be x0, compute r <- f - Ax0
for i <- 1; 2;... until convergence do
Solve Mz <- r   // Sparse lower and
                // upper triangular solves
rhoi <-  r^T z
if i == 1 then
   p <- z
else
 beta <- rhoi/rho_i-11
   p <- z +beta*p
end if
Compute q <- Ap // Sparse matrix-vecton
                // multiplication
alpha <- rhoi/p^Tq
  x <- x + alpha*p
  r <- r - alpha*q
end for
```

[Iterative Methods Using CUSPARSE and CUBLAS. NVIDIA Whitepaper June 2011.]

CG implementation I

```
//2: repeat until convergence (based on max. it. and relative residual)
for (i=0; i<maxit; i++){
    //3: Solve M z = r (sparse lower and upper triangular solves)
    cusparseDcsrsv_solve(handle, CUSPARSE_OPERATION_TRANSPOSE,
                         n, 1.0, descrpR, valR, csrRowPtrR, csrColIndR,
                         inforRt, r, t);
    cusparseDcsrsv_solve(handle, CUSPARSE_OPERATION_NON_TRANSPOSE,
                         n, 1.0, descrpR, valR, csrRowPtrR, csrColIndR,
                         inforR, t, z);

    //4: \rho = r^{T} z
    rhop= rho;
    rho = cublasDdot(n, r, 1, z, 1);
    if (i == 0){
        //6: p = z
        cublasDcopy(n, z, 1, p, 1);
    }
    else{
        //8: \beta = rho_{i} / \rho_{i-1}
        beta= rho/rhop;
        //9: p = z + \beta p
        cublasDaxpy(n, beta, p, 1, z, 1);
        cublasDcopy(n, z, 1, p, 1);
    }
```

CG implementation II

```
    //11: Compute q = A p (sparse matrix-vector multiplication)
    cusparseDcsrmv(handle, CUSPARSE_OPERATION_NON_TRANSPOSE, n, n, 1.0,
                   descrA, valA, csrRowPtrA, csrColIndA, p, 0.0, q);

    //12: \alpha = \rho_{i} / (p^{T} q)
    temp = cublasDdot(n, p, 1, q, 1);
    alpha= rho/temp;
    //13: x = x + \alpha p
    cublasDaxpy(n, alpha, p, 1, x, 1);
    //14: r = r - \alpha q
    cublasDaxpy(n,-alpha, q, 1, r, 1);

    //check for convergence
    nrmr = cublasDnrm2(n, r, 1);
    if (nrmr/nrmr0 < tol){
        break;
    }
}
```