
	<p style="text-align: center;"><b>Hochleistungsrechner</b> Praktikum Prof. Dr.-Ing. W. Rehm</p> <hr/> <p style="text-align: center;"><b>CUDA: BLAS</b> Bearbeiter: René Oertel Letzte Änderung: René Oertel, 28.05.2014</p>	
---	---	---

## 1 Ziel

Hardwarespezifische Softwareoptimierungen sind aufwendig und erfordern ein tiefgründiges Verständnis der zugrunde liegenden Architektur. Außerdem ist die Fokussierung auf eine bestimmte Hardware für die Verwendung der Software bei einer breiten Nutzerbasis nicht geeignet. Aus diesem Grund werden in Anwendungen häufig Anwendungskernels bzw. Algorithmen nur mit den Standardkonstrukten der Hochsprachen in einer simplen Form umgesetzt. Der Compiler steht damit in der Verantwortung, je nach Hardware eine andere Optimierung vorzunehmen, sofern die Anwendung z. B. für verschiedene Prozessorarchitekturen übersetzt werden kann oder wird. Dies führt meist dazu, dass vorhandene (Hardware-)Ressourcen oder Konzepte zur Steigerung der Ausführungsgeschwindigkeit ungenutzt bleiben. Aus diesem Grund soll in dieser praktischen Übung die Verwendung von optimierten Bibliotheken im Vordergrund stehen. Diese werden von einer Vielzahl von versierten Hardware- und Software-Entwicklern über einen großen Zeitraum fortlaufend optimiert und weiterentwickelt, wodurch dieser Aufwand dem Anwendungsprogrammierer abgenommen wird. Im Vergleich mit einer Open-Source-Implementierung von BLAS (OpenBLAS) sollen im konkreten Fall zwei verschiedene BLAS-Implementierungen von NVIDIA genutzt werden.

## 2 Allgemeines

Das grundlegende Verständnis zu BLAS (Basic Linear Algebra Subprograms) sollte bereits aus einer anderen Übung bekannt sein bzw. kann aus verschiedenen Quellen, z. B. [6] bezogen werden. Darüber hinaus sind grundlegende Kenntnisse zu NVIDIA CUDA C notwendig, die z. B. auch aus den Referenzdokumentationen [1, 2, 3] bzw. aus den Beispielen des GPU Computing SDK<sup>1</sup> von NVIDIA abgeleitet werden können.

### 2.1 OpenBLAS

OpenBLAS ist eine Weiterentwicklung von GotoBLAS2, das keine handgeschriebenen Optimierungen für neuere Prozessoren erhalten hat, nachdem die Entwicklung eingestellt wurde. GotoBLAS2<sup>2</sup> wurde ursprünglich von Kazushige Goto am TACC (Texas Advanced Computing Center) entwickelt. OpenBLAS beinhaltet auch Optimierungen für aktuelle Prozessoren, z. B. für die der Intel „Haswell“ Mikroarchitektur. Es wird am Institut für Software der Chinesischen Akademie der Wissenschaften entwickelt und findet momentan weite Verbreitung bei verschiedenen Linux-Distributionen. Es ersetzt damit die Referenzimplementierung von Netlib<sup>3</sup>. Kommerzielle Implementierungen von BLAS sind z. B. die Intel Math Kernel Library (MKL) oder AMD Core Math Library (ACML).

#### Installation

Der Quellcode von OpenBLAS steht bei GitHub zum Download bereit und sollte spezifisch auf dem jeweiligen System übersetzt und installiert werden, auf dem es genutzt werden soll. Generische Installationen der Linux-Distributionen sind unter Umständen nicht geeignet.

Der Quellcode kann mit den folgenden Befehlen heruntergeladen und übersetzt werden:

```
git clone git://github.com/xianyi/OpenBLAS
cd OpenBLAS
make
```

<sup>1</sup><https://developer.nvidia.com/gpu-computing-sdk>

<sup>2</sup><https://www.tacc.utexas.edu/tacc-projects/gotoblas2>

<sup>3</sup><http://www.netlib.org/blas/>

Nach Abschluss des Übersetzungsvorgangs sollten `libopenblas_<arch>-<vers>.{so,a}` Bibliotheken im OpenBLAS-Verzeichnis existieren. Die genaue Bezeichnung hängt demzufolge von der Architektur und der Versionsnummer von OpenBLAS ab und diese wird am Ende des `make`-Befehls ausgegeben.

## Verwendung

Für die Verwendung muss die Anwendung gegen die OpenBLAS-Bibliothek gelinkt werden. Mit den folgenden Linker-Parametern kann dies beispielhaft erfolgen:

---

```
-L<OpenBLAS-Verzeichnis> -lopenblas
```

---

## 2.2 NVBLAS

Die NVIDIA NVBLAS-Bibliothek [4] ist eine Multi-GPU-Implementierung für BLAS, die als Ersatz herkömmlicher BLAS-Implementierungen ohne Änderung des Quellcodes dienen kann. Sie ist seit NVIDIA CUDA 6.0 verfügbar und setzt auf die cuBLAS-Bibliothek (siehe Abschnitt 2.3) auf. Die Verwaltung von GPU-Ressourcen erfolgt implizit durch die Bibliothek, weshalb auf Basis von Heuristiken nur bestimmte Problemgrößen auf die GPU ausgelagert werden. Das Verhalten der Bibliothek lässt sich durch Umgebungsvariablen bzw. durch eine Konfigurationsdatei beeinflussen.

## Verwendung

Die Verwendung von NVBLAS erfordert analog zur Nutzung anderer CUDA-Bibliotheken eine korrekte Systemkonfiguration bzgl. Bibliothekspfade. Im Linux-Umfeld werden diese durch `LIBRARY_PATH` (Linker) bzw. `LD_LIBRARY_PATH` (Loader) bestimmt. Sind diese korrekt gesetzt (etwa durch Laden eines „Module“), dann ist folgender Linker-Parameter zu verwenden:

---

```
-lnvblas
```

---

Des Weiteren muss eine `nvblas.conf`-Datei mit grundlegenden Parametern existieren, also z. B.:

---

```
NVBLAS_LOGFILE nvblas.log  
NVBLAS_CPU_BLAS_LIB libopenblas.so
```

---

## 2.3 cuBLAS

Bei NVIDIA cuBLAS [5] handelt es sich um eine optimierte BLAS-Bibliothek, die direkt auf die NVIDIA CUDA C Funktionalität aufsetzt. Dies bedeutet für die Verwendung in einer herkömmlichen Anwendung, dass diese um CUDA-Funktionen erweitert werden muss. Insbesondere setzt cuBLAS eine Speicherverwaltung mithilfe der CUDA Runtime voraus. Konkret bedeutet dies, dass die Matrizen explizit zwischen Host- und Device-Memory ausgetauscht werden müssen. Die cuBLAS-Bibliothek wird vom CUDA Toolkit mitgeliefert und steht ohne weitere Installation bereit.

## Verwendung

Die Verwendung von cuBLAS erfolgt normalerweise in drei Schritten:

1. cuBLAS-Handle anlegen:

---

```
cublasStatus_t  
cublasCreate(cublasHandle_t *handle)
```

---

2. cuBLAS-Funktion aufrufen, z. B. `dgemm`:

---

```
cublasStatus_t
cublasDgemm(cublasHandle_t handle,
cublasOperation_t transa, cublasOperation_t transb,
int m, int n, int k,
const double *alpha,
const double *A, int lda,
const double *B, int ldb,
const double *beta,
double *C, int ldc)
```

---

3. cuBLAS-Handle löschen:

---

```
cublasStatus_t
cublasDestroy(cublasHandle_t handle)
```

---

Für eine detaillierte Beschreibung der Parameter von `cublasDgemm` sei auf das GPU Computing SDK von NVIDIA bzw. auf die cuBLAS-Dokumentation [5] verwiesen.

### 3 Kontrollfragen

1. Welche Vor- und Nachteile bieten herstellerspezifische BLAS-Implementierungen?
2. Welche Alternativen existieren neben OpenBLAS?
3. Welche Voraussetzungen benötigt NVBLAS für die Nutzung, wenn alle BLAS-Routinen verwendet werden sollen?

## 4 Durchführung

### 4.1 Aufgabe 1

Implementieren Sie ein Programm, welches die (OpenBLAS) **BLAS SGEMM**- und **DGEMM**-Routinen vermisst! Notieren Sie für verschiedene Problem- bzw. Matrizengrößen die erreichten GFLOPS. Verwenden Sie als Grundlage ggf. die Vorgabedateien.

### 4.2 Aufgabe 2

Übersetzen bzw. linken Sie das Programm aus Aufgabe 1 unter Verwendung von **nvBLAS**! Führen Sie Messungen mit den gleichen Problemgrößen durch und notieren Sie auch dafür die Ergebnisse.

### 4.3 Aufgabe 3

Modifizieren Sie das Programm für die Nutzung von **cuBLAS**! Führen Sie analog zu den vorherigen Aufgaben Messungen durch und notieren Sie die Ergebnisse.

## 5 Auswertung

1. Stellen Sie die Messergebnisse aus den Aufgaben 1–3 grafisch, z. B. mit `gnuplot`, dar!
2. Welche Messwerte haben Sie warum erwartet und welche haben Sie tatsächlich gemessen?
3. Welche Vor- und Nachteile sehen Sie bei der Verwendung von cuBLAS gegenüber NVBLAS?

## Literatur und wichtige Links

- [1] NVIDIA Corporation: NVIDIA CUDA C Programming Guide  
[http://docs.nvidia.com/cuda/pdf/CUDA\\_C\\_Programming\\_Guide.pdf](http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf)  
<http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>
- [2] NVIDIA Corporation: NVIDIA CUDA C Best Practices Guide  
[http://docs.nvidia.com/cuda/pdf/CUDA\\_C\\_Best\\_Practices\\_Guide.pdf](http://docs.nvidia.com/cuda/pdf/CUDA_C_Best_Practices_Guide.pdf)  
<http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html>
- [3] NVIDIA Corporation: NVIDIA CUDA Runtime API  
[http://docs.nvidia.com/cuda/pdf/CUDA\\_Runtime\\_API.pdf](http://docs.nvidia.com/cuda/pdf/CUDA_Runtime_API.pdf)  
<http://docs.nvidia.com/cuda/cuda-runtime-api/index.html>
- [4] NVIDIA Corporation: NVBLAS User Guide  
[http://docs.nvidia.com/cuda/pdf/NVBLAS\\_Library.pdf](http://docs.nvidia.com/cuda/pdf/NVBLAS_Library.pdf)  
<http://docs.nvidia.com/cuda/nvblas/index.html>
- [5] NVIDIA Corporation: cuBLAS User Guide  
[http://docs.nvidia.com/cuda/pdf/CUBLAS\\_Library.pdf](http://docs.nvidia.com/cuda/pdf/CUBLAS_Library.pdf)  
<http://docs.nvidia.com/cuda/cublas/index.html>
- [6] C. L. Lawson, R. J. Hanson, D. Kincaid und F. T. Krogh: Basic Linear Algebra Subprograms for FORTRAN usage, ACM Transactions on Mathematical Software (TOMS), Volume 5 Issue 3, Sept. 1979, Seiten 308–323  
<http://dx.doi.org/10.1145/355841.355847>

28. Mai 2014