PMAA 2014
CP 3.3 Software
July 3rd, Lugano

The FlexiBLAS Library for Easy Switching of BLAS Implementations in Scientific Computing

Martin Köhler and Jens Saak

Computational Methods in Systems and Control Theory
Max Planck Institute for Dynamics of Complex Technical Systems

MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

http://www.mpi-magdeburg.mpg.de/projects/flexiblas

What is BLAS?

BLAS routine organization

Basic Linear Algebra Subprograms (BLAS) –

standard building blocks for performing vector and matrix operations.

What is BLAS?

BLAS routine organization

Basic Linear Algebra Subprograms (BLAS) –

standard building blocks for performing vector and matrix operations.

Let α, β be scalars, x, y be vectors, A, B, C be matrices.

level	included operations	data	flops
1	αx, $\alpha x + y$, $x^* y$, $\|\|x\|\|_2$, $\|\|x\|\|_1$, $\|\|x\|\|_\infty$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
2	$\alpha Ax + \beta y$, $\alpha A^* x + \beta y$, $A + \alpha xy^*$, $A + \alpha xx^*$, $A + \alpha xy^* + \beta yx^*$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
3	$\alpha AB + \beta C$, $\alpha AB^* + \beta C$, $\alpha A^* B^* + \beta C$, $\alpha AA^* + \beta C$, $\alpha A^* A + \beta C$ rank k updates $\alpha A^* B + \beta C$, $\alpha B^* A + \beta C$ rank $2k$ updates	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$

What is BLAS?

BLAS routine organization

Level 3 BLAS especially attractive for communication avoidance and parallelism.

Let α, β be scalars, x, y be vectors, A, B, C be matrices.

level	included operations	data	flops
1	αx, $\alpha x + y$, $x^* y$, $\|\|x\|\|_2$, $\|\|x\|\|_1$, $\|\|x\|\|_\infty$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
2	$\alpha Ax + \beta y$, $\alpha A^* x + \beta y$, $A + \alpha xy^*$, $A + \alpha xx^*$, $A + \alpha xy^* + \beta yx^*$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
3	$\alpha AB + \beta C$, $\alpha AB^* + \beta C$, $\alpha A^* B^* + \beta C$, $\alpha AA^* + \beta C$, $\alpha A^* A + \beta C$ rank k updates $\alpha A^* B + \beta C$, $\alpha B^* A + \beta C$ rank $2k$ updates	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$

What is BLAS?

Some important BLAS implementations

Open Source

- NetLib BLAS: `http://www.netlib.org/blas/`
  (The reference implementation)
- OpenBLAS: `http://www.openblas.net/`
  (uses assembler level optimization and threading)
- Automatically Tuned Linear Algebra Software (ATLAS):
  `http://math-atlas.sourceforge.net/`
  (provides automatic tuning for specific processors and threading)

Hardware Vendor Implementations

- Intel® Math kernel library (MKL):
  `http://software.intel.com/en-us/intel-mkl/`
  (the fastest implementation on ccNUMA machines; provides hardware optimization and threading)
- AMD Core Math Library (ACML): `http://developer.amd.com/tools/cpu-development/amd-core-math-library-acml/`
  (An ATLAS version tuned by AMD?)
- Apple Accelerate: (the same from Apple ?)

Why do we need FlexiBLAS?

Linker Problems

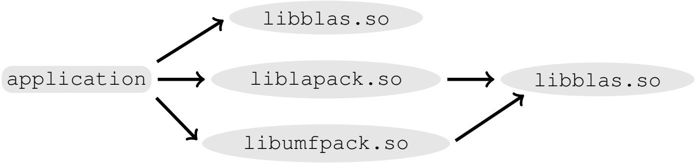

Figure: A sample application using BLAS

Why do we need FlexiBLAS?

Linker Problems

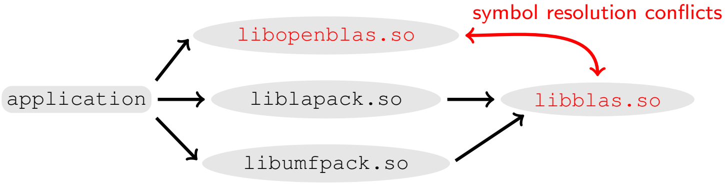

Figure: . . . after linking with a different BLAS-implementation

Why do we need FlexiBLAS?

Linker Problems: Existing Solutions

- `LD_LIBRARY_PATH` / `LD_PRELOAD`
  only applicable for single file implementations
  (i.e. NOT Intel® MKL, or ATLAS)
- static libraries
  drastically increased binary sizes, often complicated linking
- `update-alternatives` (Debian/Ubuntu/Suse)
  requires super-user privileges and has similar restrictions as `LD_LIBRARY_PATH` / `LD_PRELOAD`
- `eselect` / `pkg-config` (Gentoo)
  requires super-user privileges and switches at build-time only

Why do we need FlexiBLAS?

Compatibility Issues

`gfortran` vs `f2c/intel` interface style

- **different calling sequences:**
  `f2c` and `intel` return complex numbers as additional function parameters.
- **affected routines:** `zdotc`, `zdotu`, `cdotc`, `cdotu` (level 1)

Why do we need FlexiBLAS?

Compatibility Issues

gfortran vs f2c/intel interface style

- **different calling sequences:**
  f2c and intel return complex numbers as additional function parameters.
- **affected routines:** zdotc, zdotu, cdotc, cdotu (level 1)

auxiliary routine treatment

Routines sc/dzabs1 are missing in ATLAS and derived implementations, such as Apple Accelerate / AMD ACML.

Why do we need FlexiBLAS?

Compatibility Issues

`gfortran` vs `f2c/intel` interface style

- **different calling sequences:**
  `f2c` and `intel` return complex numbers as additional function parameters.
- **affected routines:** `zdotc`, `zdotu`, `cdotc`, `cdotu` (level 1)

auxiliary routine treatment

Routines `sc/dzabs1` are missing in ATLAS and derived implementations, such as Apple Accelerate / AMD ACML.

dependency detection problems

Correct/reliable detection of alternative BLAS implementations not guaranteed for many software packages.

Why do we need FlexiBLAS?

Profiling

- Profiling usually requires additional compiler settings
- Profiler data requires additional (sometimes confusing) tools for evaluation
- Profilers often induce considerable overhead influencing the runtime behavior of the profiled application
- Profiling needs to be active for entire applications

Why do we need FlexiBLAS?

Profiling

- Profiling usually requires additional compiler settings
- Profiler data requires additional (sometimes confusing) tools for evaluation
- Profilers often induce considerable overhead influencing the runtime behavior of the profiled application
- Profiling needs to be active for entire applications

Often only execution times and numbers of calls of single routines are of interest.

How does it work?

General Approach (Idea)

Long Story Short

We employ a plugin-like framework on top of the POSIX features for dynamic loading of shared libraries at runtime.

Similar Approach

`liftracc` Project:
T. Beisel, M. Niekamp, C. Plessl;
Paderborn Center for Parallel Computing; 2010
`http://github.com/pc2/liftracc`

How does it work?

General Approach (Idea)

Long Story Short

We employ a plugin-like framework on top of the POSIX features for dynamic loading of shared libraries at runtime.

POSIX.1 2001 `dl*`-family

`dlopen`	add a shared library and its dynamic dependencies to the current address space.
`dlsym`	search for symbols in the current address space beginning in the handle retrieved by `dlopen`.
`dlclose`	close a previously opened shared library if no other references to the library exist.
`dlerror`	provide human readable error messages.

How does it work?

General Approach (Issues)

`dlopen` based issues to solve

1. `dlopen` only integrates selected parts of the library: Each required BLAS call needs to be initialized separately.
2. Dynamically (runtime) loaded symbols can not be resolved while linking a program.
3. `dlopen` only loads a single file: Multi-file implementations require additional treatment.

How does it work?

Initialization

`__attribute__((constructor))`

- Automatically executes before the program starts.
- Reads configuration.
- Explicitly resolves all BLAS-routines to make sure they get loaded by `dlopen`.
- Initializes profiling data if desired.

How does it work?

Initialization

`__attribute__((constructor))`

- Automatically executes before the program starts.
- Reads configuration.
- Explicitly resolves all BLAS-routines to make sure they get loaded by `dlopen`.
- Initializes profiling data if desired.

`__attribute__((destructor))`

- Automatically executes after the main program exits.
- Cleanly closes the loaded shared library.
- Potentially prints profiling results.

How does it work?

Wrapper Functions

Goal

Provide a 100% Netlib-BLAS compatible API and ABI for use in user applications.

How does it work?

Wrapper Functions

Goal

Provide a 100% Netlib-BLAS compatible API and ABI for use in user applications.

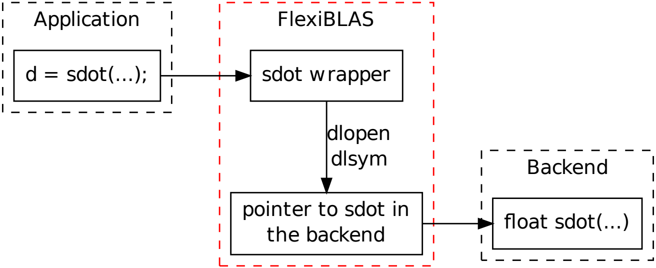

Figure: Calling `sdot` from an application via FlexiBLAS.

How does it work?

Profiling

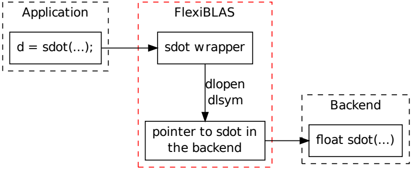

Figure: Calling `sdot` from an application via FlexiBLAS.

Basic Profiling

- Use `__attribute__((constructor))` to initialize global counters and timer variables for each BLAS-routine.
- Increase counters and timers inside the wrapper functions.
- Use `__attribute__((destructor))` for evaluation of the global variables and printing of statistics.

How does it work?

Multi-file-BLAS treatment

Remaining Question

How do we treat BLAS libraries consisting of multiple files (e.g. MKL and some versions of ATLAS), when the `dl*`-family can only use single file shared object libraries?

How does it work?

Multi-file-BLAS treatment

Remaining Question

How do we treat BLAS libraries consisting of multiple files (e.g. MKL and some versions of ATLAS), when the `dl*`-family can only use single file shared object libraries?

Simple Trick

Place an additional surrogate library between FlexiBLAS and, e.g., MKL that references all necessary symbols in MKL and behaves like a netlib-BLAS interface from the view of the dynamic linker.

How is it used?

We provide a tool that closely follows Gentoo's `eselect` syntax.

To check for backends, do

```
flexiblas list
```

To select the active backend, use

```
flexiblas set BLAS_BACKEND_NAME
```

How is it used?

We provide a tool that closely follows Gentoo’s `eselect` syntax.

To check for backends, do

```
  flexiblas list
```

To select the active backend, use

```
  flexiblas set BLAS_BACKEND_NAME
```

Alternatively we use an environment variable as in:

```
export FLEXIBLAS=/usr/lib/libopenblas.so
```

or

```
export FLEXIBLAS=ATLAS
```

How is it used?

We provide a tool that closely follows Gentoo's `eselect` syntax.

To check for backends, do

```
  flexiblas list
```

To select the active backend, use

```
  flexiblas set BLAS_BACKEND_NAME
```

Alternatively we use an environment variable as in:

```
export FLEXIBLAS=/usr/lib/libopenblas.so
```

or

```
export FLEXIBLAS=ATLAS
```

Both rely on configuration files generated automatically in `/etc/flexiblasrc` and `~/.flexiblasrc`

What are the plans for the future of FlexiBLAS?

MS Windows

- NOT POSIX
- replacement for the `dl*`-family identified and first tests look very promising.

What are the plans for the future of FlexiBLAS?

MS Windows

- NOT POSIX
- replacement for the `dl*`-family identified and first tests look very promising.

switching BLAS during process execution

- attractive when the BLAS implementation does not detect itself whether it is used in a threaded section of a program.
- requires additional functions in the API.

What are the plans for the future of FlexiBLAS?

MS Windows

- NOT POSIX
- replacement for the `dl*`-family identified and first tests look very promising.

switching BLAS during process execution

- attractive when the BLAS implementation does not detect itself whether it is used in a threaded section of a program.
- requires additional functions in the API.

Details

M. Köhler and J. Saak, *FlexiBLAS - A flexible BLAS library with runtime exchangeable backends*, Tech. Rep. 284, LAPACK Working Note, Jan. 2014.

What are the plans for the future of FlexiBLAS?

MS Windows

- NOT POSIX
- replacement for the `dl*`-family identified and first tests look very promising.

Thank you very much for your attention!

for the software package visit:

`http://www.mpi-magdeburg.mpg.de/projects/flexiblas`

switching BLAS during process execution

- attractive when the BLAS implementation does not detect itself whether it is used in a threaded section of a program
- requires additional functions in the API.

Details

M. Köhler and J. Saak, *FlexiBLAS - A flexible BLAS library with runtime exchangeable backends*, Tech. Rep. 284, LAPACK Working Note, Jan. 2014.