

Hands-on: Matrix-matrix product: Well-tuned DGEMM vs. others

- Choose either C/C++ (`c`) or Fortran (`fortran`) samples. Both of them are fine, as well.

C/C++

How to compile and how to execute

1. Run a script of creating working space

- See `objst` in `create_project.sh`.
 - It indicates a list of the settings of compile options.
 - Before running `create_project.sh`, please modify `[resource_group]` to the adequate resource group in the `task.sh`.
- Running `create_project.sh`, the executable file (`run.x`) is generated in `Obj_***` directory.
 - This directory is also automatically generated.
 - When you desire to use the own compiler, set `MAKE_DIR` variable in `create_project.sh` from `config` to `config.own`.
- Example:

```
$ cd c
$ bash create_project.sh
$ ls
Obj_fj-ssl2  Obj_fj-ssl2so
```

2. Run program

- A job script to execute the program is located in `Obj_***/results`.
- You can run the program either:

```
## To run as a batch job
$ cd Obj_fj-ssl2/results
$ pjsub task.sh
## Or, to run in an interactive job
$ cd Obj_fj-ssl2/results
$ bash task.sh
```

- Each of the samples in the Exercises will be completed within 2 minutes.
 - For safety, we set the elapsed time of the job script as 4 minutes.
 - One job script contains the execution with different kinds of matrix dimension. We treat only square matrices.
 - On the other hand, in `c.fapp` we fix a single specific matrix dimension.

Exercises A

- E1: Examine difference between a direct use of DGEMM and other implementations of matrix-matrix product in `main.c`.
- E2: Compare the performance (GFlop/s) of DGEMM to that in the others when changing matrix dimension, in `Obj_fj-ssl2/`.
 - Also, compare those to the peak value of GFlop/s of the single core in A64FX.

Exercises B (advanced)

- E3: Try to use the dynamic library of Fujitsu SSL2, instead of the static one. Check `config/Makefile.fj-ssl2so`.
- E4: In `c.fapp`, perform the basic CPUPA analysis using `fapp`. Using the CPUPA reports, compare `dgemm` to `dgemv` from the viewpoints of use of cache.
 - On the basic CPUPA, `fapp`-measurements are performed five times (i.e., `-Hevent=pa1`, `-Hevent=pa2`, `-Hevent=pa3`, `-Hevent=pa4`, and `-Hevent=pa5`).
 - Run the job with `task-Bpa.sh`. After the job completion, perform analyses using `perf.sh`. Using the resultant CSV files and `cpu_pa_report.xlsm`, you can obtain the CPUPA report.
 - Indeed, you can find that there are various kinds of difference between `dgemm` and `dgemv`, other than cache.

Fortran

How to compile and how to execute

1. Run a script of creating working space

- See `objst` in `create_project.sh`.
 - It indicates a list of the settings of compile options.
 - Before running `create_project.sh`, please modify `[resource_group]` to the adequate resource group in the `task.sh`.
- Running `create_project.sh`, the executable file (`run.x`) is generated in `Obj_***` directory.
 - This directory is also automatically generated.
 - When you desire to use the own compiler, set `MAKE_DIR` variable in `create_project.sh` from `config` to `config.own`.
- Example:

```
$ cd fortran
$ bash create_project.sh
$ ls
Obj_fj-ssl2  Obj_fj-ssl2.fast  Obj_fj-ssl2so
```

2. Run program

- A job script to execute the program is located in `Obj_***/results`.
- You can run the program either:

```
## To run as a batch job
$ cd Obj_fj-ssl2/results
$ pjsub task.sh
## Or, to run in an interactive job
$ cd Obj_fj-ssl2/results
$ bash task.sh
```

- Each of the samples in the Exercises will be completed within 2 minutes.
 - For safety, we set the elapsed time of the job script as 4 minutes.
 - One job script contains the execution with different kinds of matrix dimension. We treat only square matrices.
 - On the other hand, in `fortran.fapp` we fix a single specific matrix dimension.

Exercises A

- E1: Examine difference between a direct use of DGEMM and other implementations of matrix-matrix product in `main.F90`.
- E2: Compare the performance (GFlop/s) of DGEMM to that in the others when changing matrix dimension, in `Obj_fj-ssl2/`. Also, compare those to the peak value of GFlop/s of the single core in A64FX.
- E3: Try the case of `fortran/Obj_fj-ssl2.fast`, in which `-Kfast` option is used. Are there any differences from the case of `Obj_fj-ssl2`?

Exercises B (advanced)

- E4: Try to use the dynamic library of Fujitsu SSL2, instead of the static one. Check `config/Makefile.fj-ssl2so`.
- E5: In `fortran.fapp`, perform the basic CPUPA analysis using `fapp`. Using the CPUPA reports, compare `dgemm` to `dgemv` from the viewpoints of use of cache.
 - On the basic CPUPA, `fapp`-measurements are performed five times (i.e., `-Hevent=pa1`, `-Hevent=pa2`, `-Hevent=pa3`, `-Hevent=pa4`, and `-Hevent=pa5`).
 - Run the job with `task-Bpa.sh`. After the job completion, perform analyses using `perf.sh`. Using the resultant CSV files and `cpu_pa_report.xlsm`, you can obtain the CPUPA report.
 - Indeed, you can find that there are various kinds of difference between `dgemm` and `dgemv`, other than cache.
- E6: Consider the case of using Fortran `matmul` function as a matrix-matrix-product routine.
 - See `fortran/config/Makefile.fj-ssl2.matmul`. We have two cases:
 - `matmul` with `-Nalloc_assign -O3` option. The job execution time might become longer. (~10 minutes)
 - `matmul` with `-Kfast`, but not specifying `-Nalloc_assign`.