CS542200 Parallel Programming

Homework 5: Observe the behavior of UCX yourself

Due: Sun. Dec 25 23:59, 2024

1 GOAL

Develop observability code within the UCX framework to gain a comprehensive understanding of its architecture and enhance our knowledge in modern parallel computing frameworks, with the following objectives:

1. **Comprehend the UCX Framework**: Delve into the UCX framework to understand the roles and interconnections of its subcomponents, thereby gaining a deeper insight into its overall functionality.

2. **Examine TLS Impact**: Investigate how Transport Layer Security (TLS) is influenced by the interactions and dependencies between UCP, UCS, and UCT within the UCX framework.

3. **Learn Program Structure**: Acquire knowledge about the program structure of modern parallel computing frameworks, focusing on how they are designed, implemented, and optimized for performance.

2 REQUIREMENTS

- In this assignment, you are required to modify and recompile the UCX system (the underlying communication protocol for MPI) to add the following two features:
  - Display the UCX transport protocols currently configured in the system.
    - ◆ You are required to insert code at // TODO: PP-HW (src/ucs/config/parser.c) to ensure your program is compatible with the UCX framework.
  - Output the UCX transport protocols actively utilized by the program.

3 RUN YOUR PROGRESS

1. Recompile UCX and install it in the `$HOME/ucx-pp` directory, ensuring that a lib folder is present that contains the required dynamic link files. (libucp.so.0, libuct.so.0, libucm.so.0, libucs.so.0).

Note that every time you modify the UCX code, you need to re-execute the last line of commands to recompile.

```
git clone -b pp2024 https://github.com/NTHU-LSALAB/UCX-lsalab
cd UCX-lsalab
mkdir build && cd build
../configure --prefix=$HOME/ucx-pp/ --with-go=no
srun -n 1 -c 12 make -j12 install
```

2. We provide a custom command `mpiucx`, designed to replace `mpirun`, which will utilize your self-compiled UCX as the underlying communication framework for running MPI parallel programs. You can find the test cases in UCX-lsalab/test/

```
module load openmpi/ucx-pp
mpiucx -x UCX_LOG_LEVEL=info -np 2 ./mpi_hello.out
```

3. Alter the internal code of UCX to enable printing of:
   ○ The UCX_TLS information is currently specified by the system.
   ○ The final TLS transport method was selected by UCX.

4. You may need to trace the files within src/ucp, src/uct, and src/ucs directories and modify at least the following two files to complete the homework.
   ○ ucp/core/ucp_worker.c
      ■ Invoke ucp_config_print to print UCX_TLS.
      ■ Print Line 2 by identifying a key variable.
   ○ ucs/config/parser.c
      ■ Implement the ucs_config_parser_print_opts function to be invoked by ucp_config_print here.

4 RESULT

For each transport, the output comprises 2 lines.
- Line 1: The information from UCX_TLS needs to be exactly the same.
- Line 2: The information should include only the key strings of transport protocols selected by UCX. The example must contain strings.

  cfg#0 tag(sysv/memory cma/memory)

Sample input & output
- For the openmpi/ucx-pp configuration, the default transport protocol is set to utilize ud_verbs.

```
[willian@apollo31 mpi]$ mpiucx -np 1 ./mpi_hello.out
UCX_TLS=ud_verbs
0x56544467c8f0 self cfg#0 tag(ud_verbs/ibp3s0:1)
Hello world from processor apollo31, rank 0 out of 1 processors
```

- We have enabled UCX to leverage all available transport protocols by setting up UCX_TLS.

```
[willian@apollo31 mpi]$ mpiucx -n 2 -x UCX_TLS=all ./send_recv.out
UCX_TLS=all
0x5580604e1a90 self cfg#0 tag(self/memory cma/memory)
UCX_TLS=all
0x559a90bb7a00 self cfg#0 tag(self/memory cma/memory)
UCX_TLS=all
0x5580604e1a90 intra-node cfg#1 tag(sysv/memory cma/memory)
UCX_TLS=all
0x559a90bb7a00 intra-node cfg#1 tag(sysv/memory cma/memory)
Process 0 sent message 'Hello from rank 0' to process 1
Process 1 received message 'Hello from rank 0' from process 0
```

5 REPORT

Here is the report template https://hackmd.io/@William-Mou/pp-hw5-report
Please download the markdown and submit the .md file.

6 GRADING

1. **[30%] Correctness**
   - The pass rate of test data for ucx-judge.
2. **[20%] Demo**
   - A demo session will be held remotely. You'll be asked questions about the homework.
3. **[50%] Report**
   - Grading is based on your evaluation, discussion and writing. If you want to get more points, design or conduct more experiments to analyze your implementation.

7 SUBMISSION

Upload the files below to eeclass. (**DO NOT COMPRESS THEM**)

- hw5.diff
    - You can obtain this file by
    - **git add -A && git commit -m "Observed the behavior of UCX."**
    - **git diff --color remotes/origin/pp2024 > hw5.diff**
- hw5_{student_ID}.md or hw5_{student_ID}.pdf
    - If possible, please submit it in Markdown and provide a HackMD link or PDF file as a backup.
    - Please confirm that your image is displayed correctly.
- **KEEP $HOME/ucx-pp/ CORRECTLY**

8 HINTS

- Type ucx-judge on **Apollo** to run the test cases.
- Scoreboard:
    - https://apollo.cs.nthu.edu.tw/pp24/scoreboard/ucx/
- You can use:
    - ucx_info -d to show all available TLS in your cluster.
    - -x UCX_LOG_LEVEL=info in mpirun to obtain debug information during UCX runtime
    - -x UCX_TLS=<TLS> to specify the permitted TLS for your program.
- Refer to slide **p37** of the presentation to determine which component is accountable for the UCX_TLS configuration in the system.
- Consult slides **p41** to **p42** of the presentation for insights into UCP's architecture, which will aid in navigating your code to pinpoint the transport protocols actually in use.
- If you encounter any problems with the homework specification, judge scripts, example source code, or test cases, please contact the TA at pp@lsalab.cs.nthu.edu.tw or eeclass.

- You are allowed to discuss and exchange ideas with others, but you are required to write the code yourself. If we find you cheating, you'll receive 0 points.