SCC Reproducibility Challenge: Symmetric Block-Cyclic Distribution: Fewer Communications Leads to Faster Dense Cholesky Factorization

Lionel **Eyraud-Dubois**, Mathieu **Faverge**, Julien **Langou**,
Florent **Pruvost**, Mathieu **Vérité**

LaBRI, Inria Center at the University of Bordeaux
University of Colorado, Denver

Main messages of the paper

2DBC is not optimal for symmetric operations

- **Symmetric Block Cyclic** distribution induces fewer communications

Fewer communications results in lower execution time

- Shown experimentally with Chameleon, on 15 to 36 nodes with 36 cores each

Instructions of the challenge

Reproduce the main findings of the paper

1. Cholesky factorization with SBC performs fewer communications than 2DBC
2. This results in faster execution

Objectives: observe (1), check (2) on your own hardware

Scalability issue

- Most teams have 3 nodes, whereas SBC starts to show gain for $P \geq 6$.
- To observe (1), you may have to set up several MPI processes per node (one per GPU?)
- You should try and report on several configurations (number of MPI processes per node)
- Because of different hardware, block size might need to be tuned differently

Technical details

Software environment

- Experiments in the paper use Chameleon, StarPU, OpenMPI, Intel MKL.
- Using another Cholesky implementation is accepted, if it can use an SBC distribution
- Latest Chameleon/StarPU has support for NVIDIA and AMD GPUs
- Development version of Chameleon has support for arbitrary custom data distribution (see `https://solverstack.gitlabpages.inria.fr/chameleon/#doc-using-custom-distributions`)

Installing the software

- Git repository: `https://gitlab.inria.fr/solverstack/chameleon.git`
- Can be compiled with CMake
- Available in Debian, Brew (release) and Guix, Spack (can install the development version)
- Complete documentation at `https://solverstack.gitlabpages.inria.fr/chameleon/#doc-install`

What is Chameleon?

Quick presentation

- Chameleon is a C library which provides routines for high performance linear algebra
- Features:
  - LU, Cholesky, QR and LQ factorizations
  - Real and complex arithmetic in both single and double precision
  - Support for Linux and Mac OS/X (tested on AMD Zen(s), Intel x86-64 and IBM Power)
  - Generic interface with StarPU, PaRSEC, QUARK, OpenMP runtime systems
  - (with StarPU) exploit multiple GPUs through cuBLAS/hipBLAS
  - (with StarPU and MPI) exploit clusters of nodes with distributed memory
- The `testing/` directory contains an executable for each operation (see `https://solverstack.gitlabpages.inria.fr/chameleon/#doc-using-executables`)

Typical result

```
$ salloc --nodes=5 --time=01:00:00 --constraint p100 --exclusive --ntasks-per-node=2 --threads-per-core=1
$ export STARPU_MPI_STATS=1

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 -P 5 --nowarmup --mtxfmt=1
[starpu_comm_stats][0] TOTAL:    20000000000.000000 B    19073.486328 MB     458.978424 B/s     0.000372
0;dpotrf;17;0;5;2;1;500;121;100000;100000;1804289383;0.000000e+00;4.253680e+01;7.836470e+03

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 --custom=sbc.txt --nowarmup --mtxf
[starpu_comm_stats][0] TOTAL:    12060000256.000000 B    11501.312500 MB     283.743073 B/s     0.000230
0;dpotrf;17;0;5;1;1;500;121;100000;100000;1804289383;0.000000e+00;4.153677e+01;8.025138e+03
```

Typical result

Allocate 5 nodes with SLURM

2 MPI processes per node

```
$ salloc --nodes=5 --time=01:00:00 --constraint p100 --exclusive --ntasks-per-node=2 --threads-per-core=1
$ export STARPU_MPI_STATS=1

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 -P 5 --nowarmup --mtxfmt=1
[starpu_comm_stats][0] TOTAL:    20000000000.000000 B    19073.486328 MB     458.978424 B/s     0.000372
0;dpotrf;17;0;5;2;1;500;121;100000;100000;1804289383;0.000000e+00;4.253680e+01;7.836470e+03

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 --custom=sbc.txt --nowarmup --mtxf
[starpu_comm_stats][0] TOTAL:    12060000256.000000 B    11501.312500 MB     283.743073 B/s     0.000230
0;dpotrf;17;0;5;1;1;500;121;100000;100000;1804289383;0.000000e+00;4.153677e+01;8.025138e+03
```

Typical result

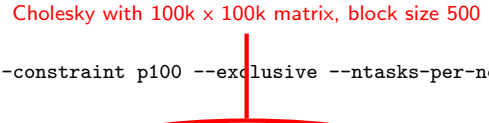
Print stats about amount of communication

```
$ salloc --nodes=5 --time=01:00:00 --constraint p100 --exclusive --ntasks-per-node=2 --threads-per-core=1
$ export STARPU_MPI_STATS=1

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 -P 5 --nowarmup --mtxfmt=1
[starpu_comm_stats][0] TOTAL:    20000000000.000000 B    19073.486328 MB     458.978424 B/s     0.000372
0;dpotrf;17;0;5;2;1;500;121;100000;100000;1804289383;0.000000e+00;4.253680e+01;7.836470e+03

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 --custom=sbc.txt --nowarmup --mtxf
[starpu_comm_stats][0] TOTAL:    12060000256.000000 B    11501.312500 MB     283.743073 B/s     0.000230
0;dpotrf;17;0;5;1;1;500;121;100000;100000;1804289383;0.000000e+00;4.153677e+01;8.025138e+03
```

Typical result

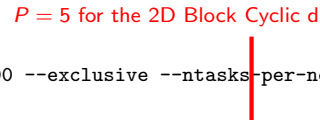



```
$ salloc --nodes=5 --time=01:00:00 --constraint p100 --exclusive --ntasks-per-node=2 --threads-per-core=1
$ export STARPU_MPI_STATS=1

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 -P 5 --nowarmup --mtxfmt=1
[starpu_comm_stats][0] TOTAL:    20000000000.000000 B    19073.486328 MB     458.978424 B/s     0.000372
0;dpotrf;17;0;5;2;1;500;121;100000;100000;1804289383;0.000000e+00;4.253680e+01;7.836470e+03

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 --custom=sbc.txt --nowarmup --mtxf
[starpu_comm_stats][0] TOTAL:    12060000256.000000 B    11501.312500 MB     283.743073 B/s     0.000230
0;dpotrf;17;0;5;1;1;500;121;100000;100000;1804289383;0.000000e+00;4.153677e+01;8.025138e+03
```

Typical result




```
$ salloc --nodes=5 --time=01:00:00 --constraint p100 --exclusive --ntasks-per-node=2 --threads-per-core=1
$ export STARPU_MPI_STATS=1

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 -P 5 --nowarmup --mtxfmt=1
[starpu_comm_stats][0] TOTAL:    20000000000.000000 B    19073.486328 MB      458.978424 B/s      0.000372
0;dpotrf;17;0;5;2;1;500;121;100000;100000;1804289383;0.000000e+00;4.253680e+01;7.836470e+03

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 --custom=sbc.txt --nowarmup --mtxf
[starpu_comm_stats][0] TOTAL:    12060000256.000000 B    11501.312500 MB      283.743073 B/s      0.000230
0;dpotrf;17;0;5;1;1;500;121;100000;100000;1804289383;0.000000e+00;4.153677e+01;8.025138e+03
```

Typical result


```
$ salloc --nodes=5 --time=01:00:00 --constraint p100 --exclusive --ntasks-per-node=2 --threads-per-core=1
$ export STARPU_MPI_STATS=1

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 -P 5 --nowarmup --mtxfmt=1
[starpu_comm_stats][0] TOTAL:    20000000000.000000 B    19073.486328 MB      458.978424 B/s      0.000372
0;dpotrf;17;0;5;2;1;500;121;100000;100000;1804289383;0.000000e+00;4.253680e+01;7.836470e+03

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 --custom=sbc.txt --nowarmup --mtxf
[starpu_comm_stats][0] TOTAL:    12060000256.000000 B    11501.312500 MB      283.743073 B/s      0.000230
0;dpotrf;17;0;5;1;1;500;121;100000;100000;1804289383;0.000000e+00;4.153677e+01;8.025138e+03
```

Typical result

```
$ salloc --nodes=5 --time=01:00:00 --constraint p100 --exclusive --ntasks-per-node=2 --threads-per-core=1
$ export STARPU_MPI_STATS=1

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 -P 5 --nowarmup --mtxfmt=1
[starpu_comm_stats][0] TOTAL:    20000000000.000000 B    19073.486328 MB     458.978424 B/s     0.000372
0;dpotrf;17;0;5;2;1;500;121;100000;100000;1804289383;0.000000e+00;4.253680e+01;7.836470e+03

$ mpiexec --map-by socket chameleon_dtesting -o potrf -n 100000 -b 500 --custom=sbc.txt --nowarmup --mtxf
[starpu_comm_stats][0] TOTAL:    12060000256.000000 B    11501.312500 MB     283.743073 B/s     0.000230
0;dpotrf;17;0;5;1;1;500;121;100000;100000;1804289383;0.000000e+00;4.153677e+01;8.025138e+03
```

Custom SBC distribution

Fewer communications

Slightly lower time

Rubric

Baseline expectations

- Install & experiment
- Observe communication reduction
- Observe & quantify the speedup relative to your hardware

Above & Beyond

- Have fun, be creative, do science!
- Do not attempt to do everything: doing simple and relevant experiments correctly is best

Report structure (4 to 6 pages)

- Present your architecture
- Describe your experiments (succesful or failed)
- Provide results and interpretations

Instructions of the challenge – extensions

Possible extensions

1. Analyze the effect of different process placement to fit the SBC distribution
2. Measure energy usage to see if fewer communications result in lower energy
3. Compare several Cholesky implementations
4. Compare measurements of amount of communications versus what the theory predicts (for both 2DBC and SBC)
5. Beyond Cholesky factorization: observe (1) and (2) for POSV (factorize+solve) and/or POTRI
6. Explore 2.5D algorithms (requires more MPI processes)
7. Explore other distributions (`https://inria.hal.science/hal-04093162`, `https://inria.hal.science/hal-04013708`)
8. Probably many other possibilities, be creative and curious!

Questions?

Contact us if you need more information

- Ask questions on the Discourse server
  `https://discourse.studentclustercompetition.us/`
- The sooner, the better, we might not be available immediately
- Our timezones: mostly French time (Paris), J. Langou is in Denver