

Saint Petersburg National Research University of Information Technologies, Mechanics and
Optics (ITMO University)
Faculty of Informational Technologies and Programming

REPORT

about laboratory work № 1

« Point-to-Point Communications in MPI»

Student

Nerinovsky Arseny
(Surname, initials)

M41331
Group

Saint-Petersburg, 2018

Report

June 29, 2019

1

MPI_SEND, MPI_Send, MPI_Ssend, MPI_Rsend) .

2

, .

3

Open MPI - , , . , Open MPI , , MPI. Open MPI , .
MPI " ." () (). " " .

- MPI_Send MPI_Send , . (,).
- MPI_Bsend May buffer; , . MPI. .
- MPI_Ssend ,
- MPI_Rsend , . .
- MPI_Isend . . , , , (. MPI_Request_free). , , I immediate, MPI_Isend . . .

return, send . , -Mpi_ibsend -MPI_Ssend . , . -MPI_Irsend MPI_Rsend,

, "" , . , MPI .

4

[1]: %cat hello.c

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

#include <unistd.h>
#include <stdbool.h>

#ifdef MSG_LEN
```

```

# define MSG_LEN 32
#endif

#ifdef SEND_FN
# define SEND_FN MPI_Send
#endif

#if !defined(SYNC) && !defined(SEND_RECV) && !defined(ASYNC)
# define SYNC
#endif

void rand_str(char *str, size_t len)
{
    for(size_t i = 0; i < len - 1; ++i) {
        str[i] = rand() % 26 + 64;
    }
    str[len] = 0;
}

int main(int argc, char **argv)
{
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    MPI_Request req;
    MPI_Status status;
    bool wait = false;
    srand(rank+10);

    char buf[MSG_LEN], rbuf[MSG_LEN];

#ifdef SYNC
    printf("SYNC\n");
#endif

#ifdef SEND_RECV
    printf("SEND_RECV\n");
#endif

#ifdef ASYNC
    printf("ASYNC\n");
#endif
}

```

```

        for(size_t i = 0; i < 10; ++i) {

#ifdef SYNC
            if( (i + rank) % 2 == 0 ) {
                MPI_Recv(buf, MSG_LEN, MPI_CHAR, !rank, 0,
MPI_COMM_WORLD,MPI_STATUS_IGNORE);
                printf("%sRECV(%d) : %s\n",rank*44, " ", rank, buf);
            } else {
                rand_str(buf, MSG_LEN);
                printf("%sSEND(%d) : %s\n", rank*44, " ", rank, buf);
                SEND_FN(buf, MSG_LEN, MPI_CHAR, !rank, 0,
MPI_COMM_WORLD);
            }
        }

#ifdef SEND_RECV
            rand_str(buf, MSG_LEN);
            printf("%sSEND(%d) : %s\n", rank*44, " ", rank, buf);
            MPI_Sendrecv(buf, MSG_LEN, MPI_CHAR, !rank, 0,
                                rbuf, MSG_LEN, MPI_CHAR, !rank,
0,
                                MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
            printf("%sRECV(%d) : %s\n",rank*44, " ", rank, rbuf);
        }

#ifdef ASYNC
            if( (i + rank) % 2 == 0 ) {
                if(wait) {
                    MPI_Wait(&req, &status);
                    MPI_Irecv(buf, MSG_LEN, MPI_CHAR, !rank, 0,
MPI_COMM_WORLD, &req);
                    wait = true;
                    printf("%sRECV(%d) : %s\n",rank*44, " ", rank,
buf);
                }
            } else {
                rand_str(buf, MSG_LEN);
                printf("%sSEND(%d) : %s\n", rank*44, " ", rank, buf);
                if(wait) MPI_Wait(&req, &status);
                MPI_Isend(buf, MSG_LEN, MPI_CHAR, !rank, 0,
MPI_COMM_WORLD, &req);
                wait = true;
            }
        }

        // sleep(rand() % 5);

```

```

    }

    MPI_Finalize();
}

```

5

```

[2]: import subprocess
import os

def compile(*defs, **defskw):
    args = [f"-D{k}" for k in defs] + [f"-D{k}={v}" for k, v in defskw.items()]
    _cmd = 'mpicc -o hello hello.c'.split() + args
    # print(' '.join(_cmd))
    cmd = subprocess.run(_cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

    if(cmd.stdout): print('cmd.stdout', cmd.stdout)
    if(cmd.stderr): print('cmd.stderr', cmd.stderr)

def run(env=None):
    cmd = subprocess.run('mpiexec -np 2 ./hello'.split(), stdout=subprocess.
    ↳PIPE, stderr=subprocess.PIPE, env=env)
    if(cmd.stderr): print('cmd.stderr', cmd.stderr)

```

5.1

```

[3]: for i in range(8):
    compile(MSG_LEN=10**i)
    print(f"Using message length {10**i}", end="\n\t")
    %timeit run()
    print()

```

Using message length 1

11.1 ms ± 218 μ s per loop (mean ± std. dev. of 7 runs, 100 loops each)

Using message length 10

11.1 ms ± 222 μ s per loop (mean ± std. dev. of 7 runs, 100 loops each)

Using message length 100

11.3 ms ± 249 μ s per loop (mean ± std. dev. of 7 runs, 100 loops each)

Using message length 1000

11.1 ms ± 343 μ s per loop (mean ± std. dev. of 7 runs, 100 loops each)

Using message length 10000

13.3 ms ± 481 μ s per loop (mean ± std. dev. of 7 runs, 100 loops each)

Using message length 100000

27.6 ms ± 1.6 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

Using message length 1000000

166 ms ± 6.05 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

Using message length 10000000

5.46 ms ± 260 μ s per loop (mean ± std. dev. of 7 runs, 100 loops each)

5.2

send .

```
[4]: for snd in 'MPI_Rsend MPI_Ssend MPI_Send'.split():

    compile(MSG_LEN=10**8, SEND_FN=snd)

    print(f"Using {snd} as send function", end="\n\t")
    %timeit run()
    print()
```

Using MPI_Rsend as send function

5.37 ms ± 232 μ s per loop (mean ± std. dev. of 7 runs, 100 loops each)

Using MPI_Ssend as send function

5.72 ms ± 378 μ s per loop (mean ± std. dev. of 7 runs, 100 loops each)

Using MPI_Send as send function

5.66 ms ± 204 μ s per loop (mean ± std. dev. of 7 runs, 100 loops each)

5.3 SYNC vs ASYNC vs SENDRECV

recv.

```
[5]: for snd in 'SYNC ASYNC SEND_RECV'.split():

    compile(snd, MSG_LEN=10**8)

    print(f"Using {snd}", end="\n\t")
    %timeit run()
    print()
```

Using SYNC

5.94 ms \pm 154 μ s per loop (mean \pm std. dev. of 7 runs, 100 loops each)

Using ASYNC

5.77 ms \pm 90.2 μ s per loop (mean \pm std. dev. of 7 runs, 100 loops each)

Using SEND_RECV

5.91 ms \pm 165 μ s per loop (mean \pm std. dev. of 7 runs, 100 loops each)

6

,, , MPI_Ssend , MPI_Send , MPI_Ssend . MPI_Send MPI . MPI_Bsend ,
MPI_Isend, MPI_Bsend . MPI_Send, MPI_Issend ., , MPI.