Communication with Double Buffering

Issaku Kanamori (RIKEN)

December 12, 2019 at R-CCS
Fugaku QCD Coding workshop

RIKEN R-CCS RIKEN
Center for
Computational Science

Outline

1. Introduction
2. Algorithm with Double Buffering
3. Benchmark (A64FX 1 node)
4. Conclusions

Outline

1. Introduction
2. Algorithm with Double Buffering
3. Benchmark (A64FX 1 node)
4. Conclusions

Acknowledgments

this talk is based on discussion with the codesign team for LQCD

Introduction

Performance Bottle Neck on Lattice QCD

- the most time consuming: mult of D in the solver
- memory bandwidth
- communication bandwidth

  - neighboring communication in D: need to wait for boundary data comes
  - overlapping communication and computation: as computation becomes faster, it becomes more difficult to hide communication

double buffering algorithm may reduce the comm. overhead
(implementation: RDMA through the uTofu interface)

Algorithm with Double Buffering

Neighboring Communication

process 1 (send)

send buffer

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

Neighboring Communication

process 1 (send)

send buffer

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

process 2 (recv.)

recv. buffer

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

Neighboring Communication

process 1 (send)

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

send buffer

P

process 2 (recv.)

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

recv. buffer

R

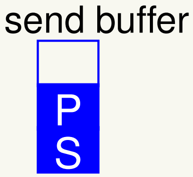
send buf. P : Packed P : Sending

recv. buf. R : Receiving, R : Receiving done, U : being Used

Neighboring Communication

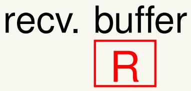
process 1 (send)

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```



process 2 (recv.)

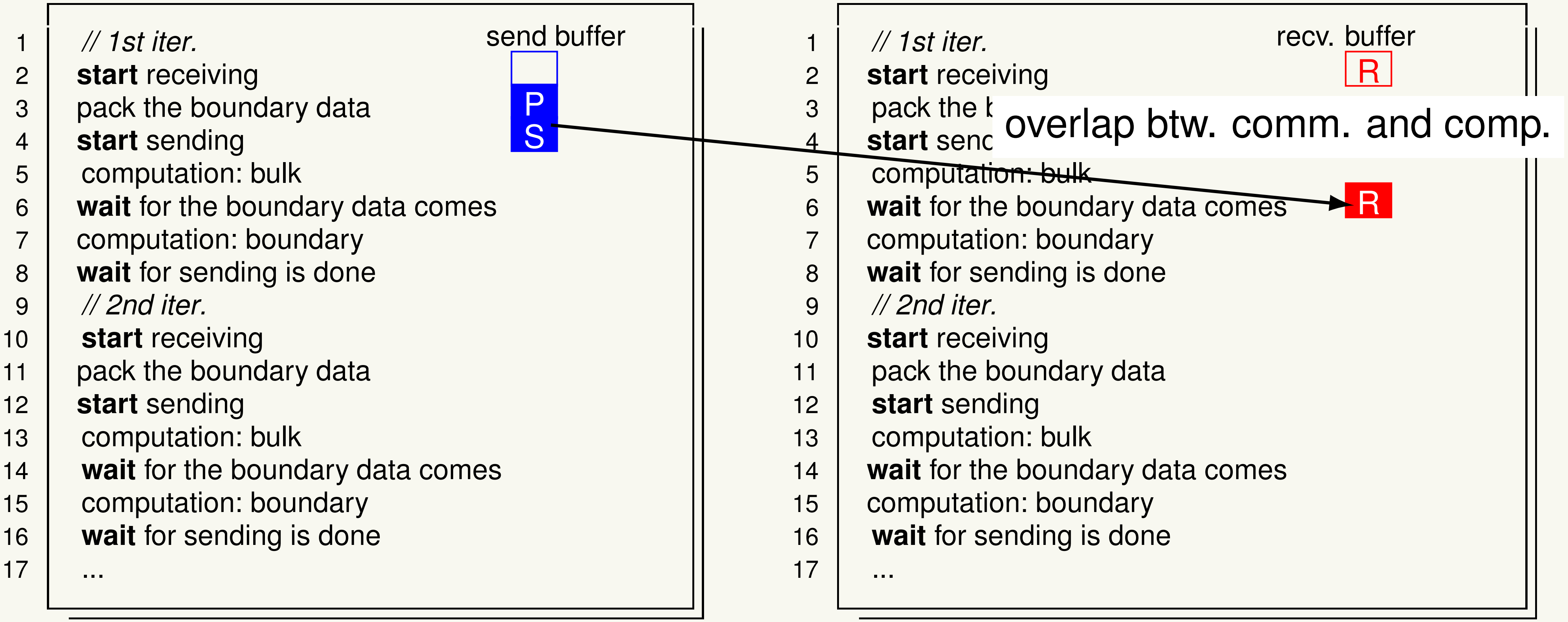
```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```



send buf. P : Packed P : Sending

recv. buf. R : Receiving, R : Receiving done, U : being Used

Neighboring Communication



process 1 (send)

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

process 2 (recv.)

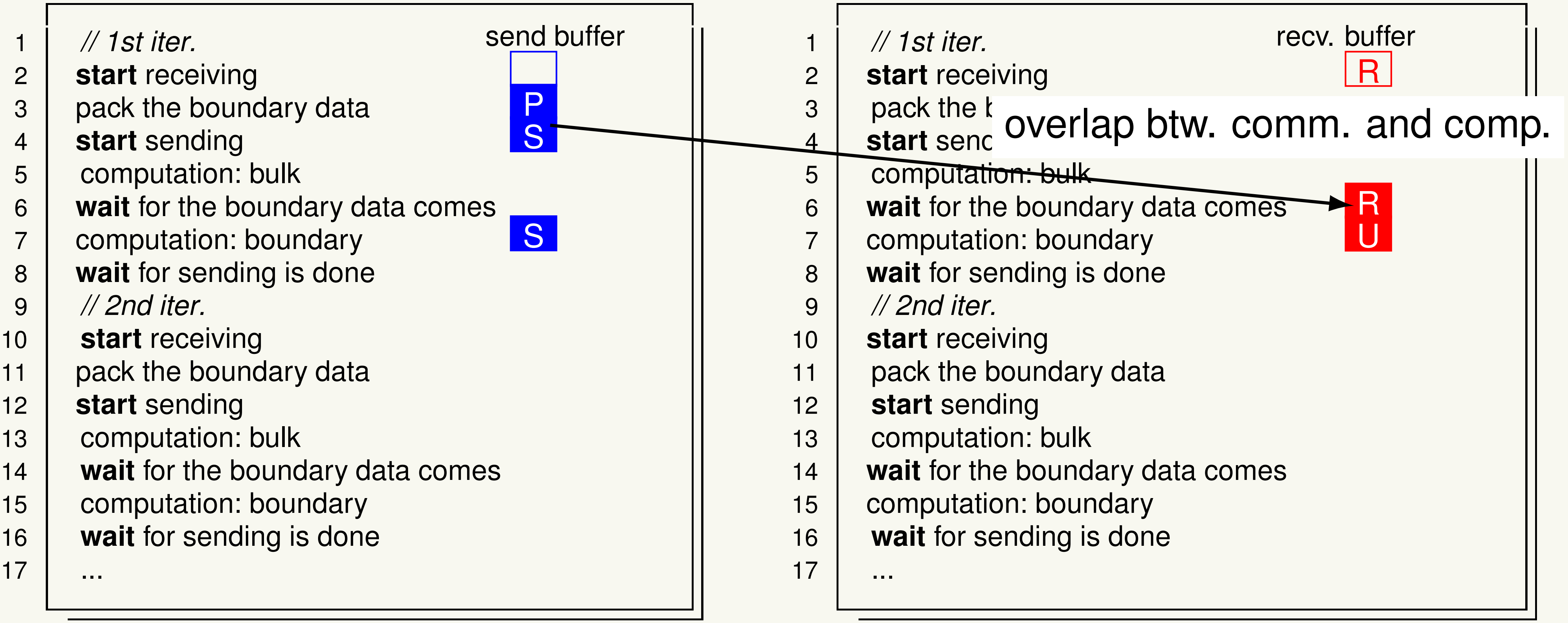
```
// 1st iter.
start receiving
pack the b
start send
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

overlap btw. comm. and comp.

send buf. **P** : Packed **P** : Sending

recv. buf. R : Receiving, **R** : Receiving done, **U** : being Used

Neighboring Communication



process 1 (send)

```
// 1st iter.
start receiving
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
 // 2nd iter.
 start receiving
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
 wait for sending is done
 ...
```

process 2 (recv.)

```
// 1st iter.
start receiving
 pack the b
start senc
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
 // 2nd iter.
start receiving
 pack the boundary data
 start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 ...
```

overlap btw. comm. and comp.

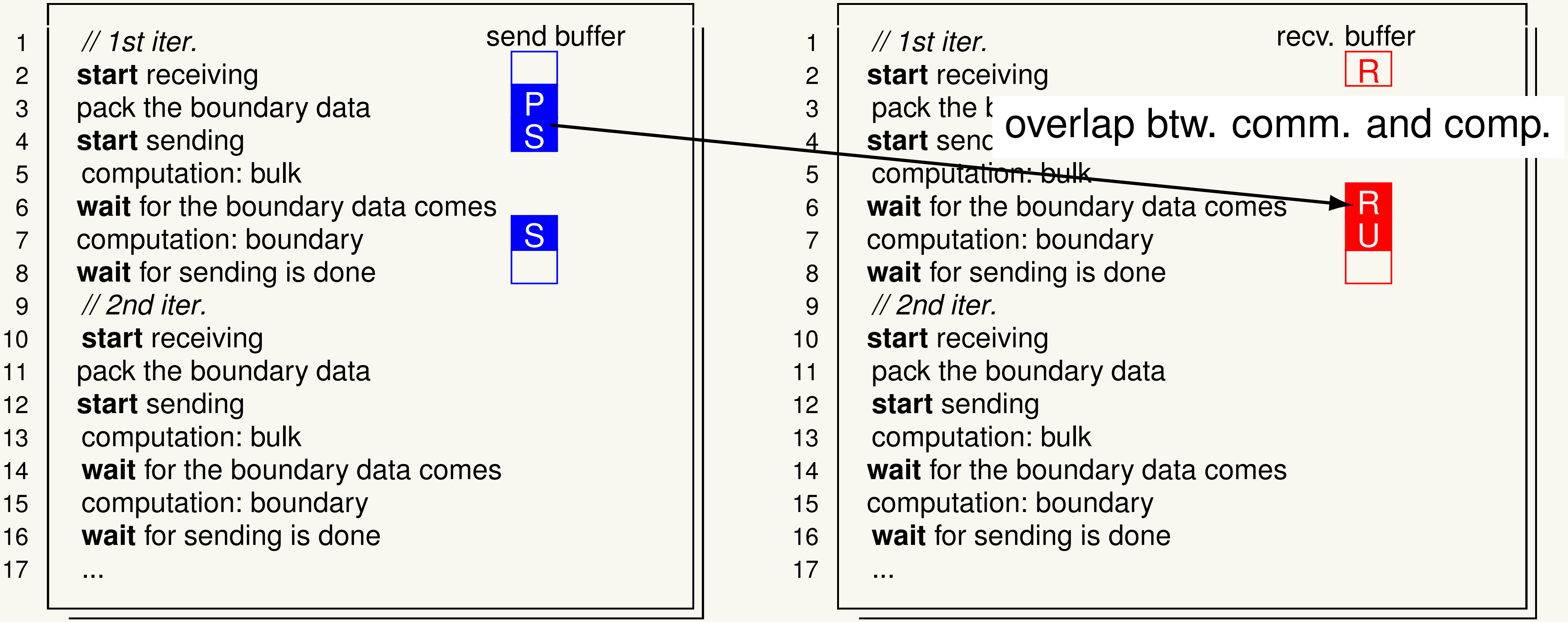
send buf. P : Packed P : Sending

recv. buf. R : Receiving, R : Receiving done, U : being Used

Neighboring Communication

process 1 (send)

process 2 (recv.)



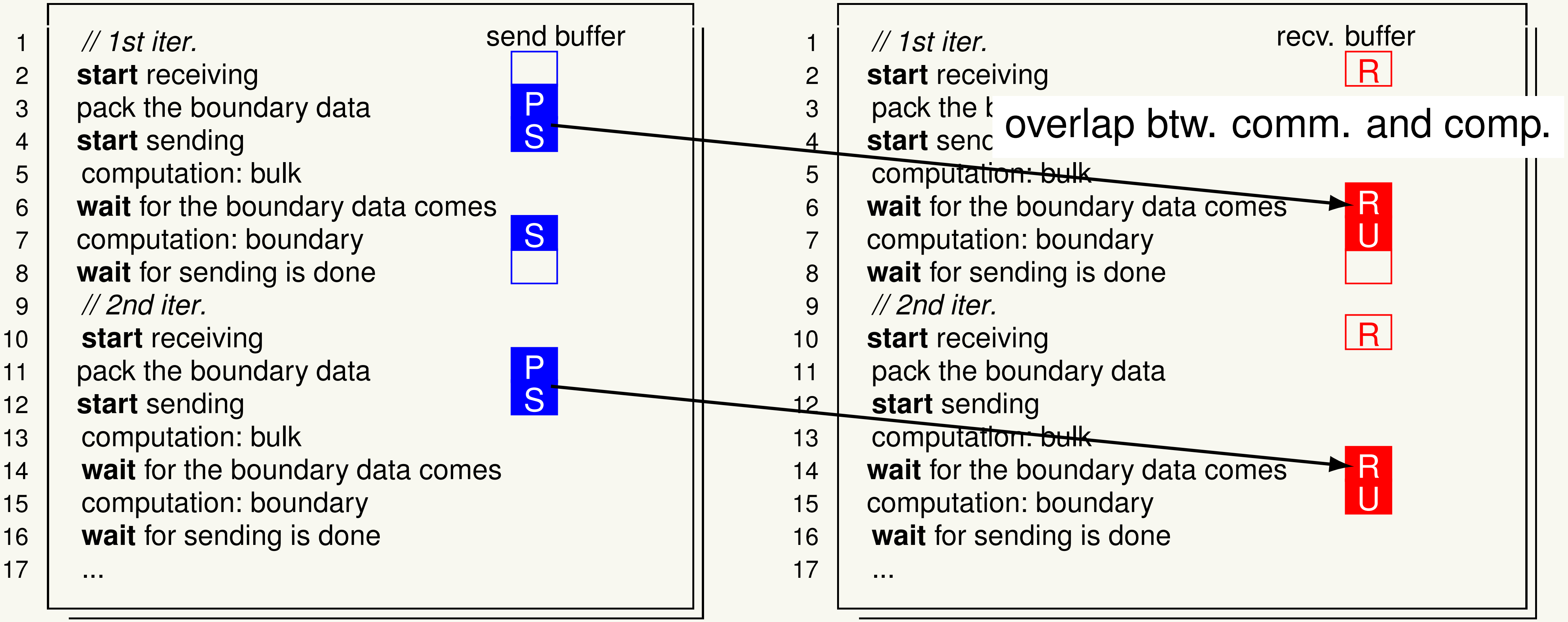
```
// 1st iter.
start receiving
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
 // 2nd iter.
 start receiving
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
 wait for sending is done
 ...
```

```
// 1st iter.
start receiving
 pack the b
start senc
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
 // 2nd iter.
start receiving
 pack the boundary data
 start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 ...
```

send buf. P : Packed P : Sending

recv. buf. R : Receiving, R : Receiving done, U : being Used

Neighboring Communication



process 1 (send)

```
 // 1st iter.
start receiving
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
 // 2nd iter.
 start receiving
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
 wait for sending is done
 ...
```

process 2 (recv.)

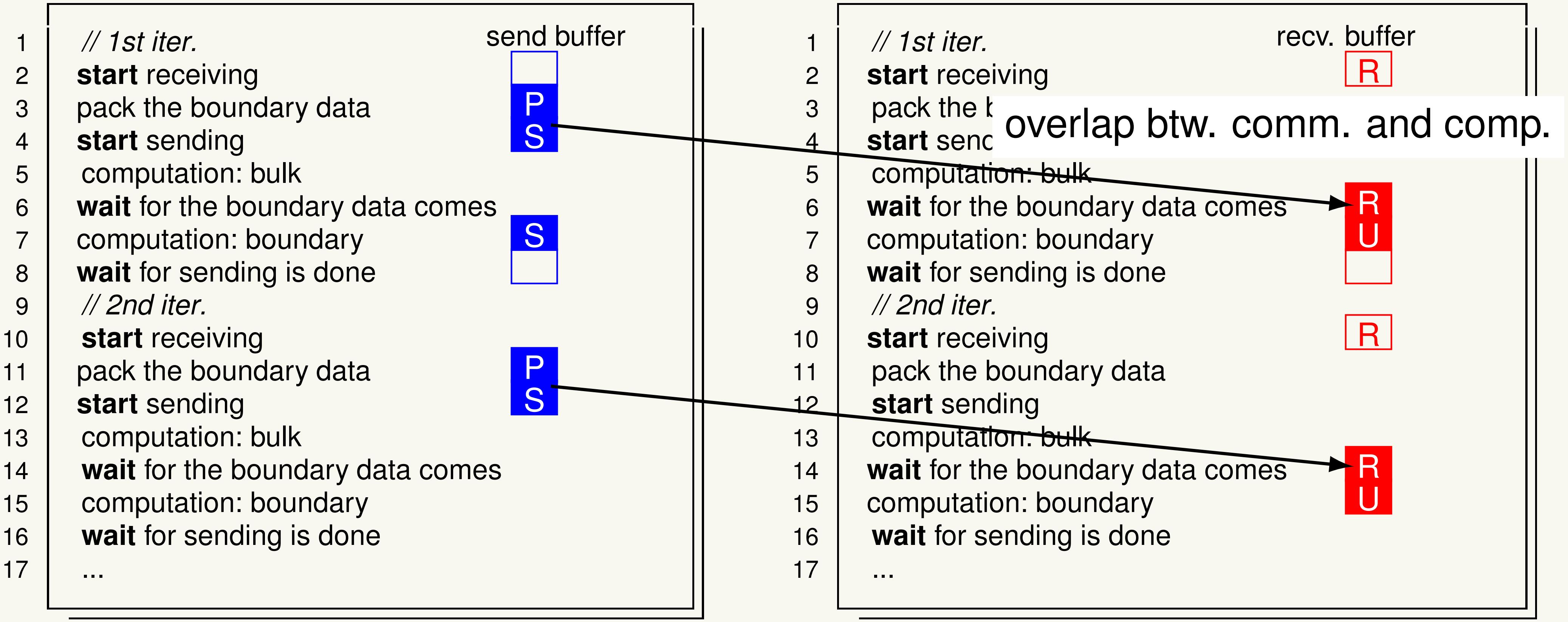
```
// 1st iter.
start receiving
 pack the b
start send
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
 pack the boundary data
 start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 ...
```

overlap btw. comm. and comp.

send buf. P : Packed P : Sending

recv. buf. R : Receiving, R : Receiving done, U : being Used

Neighboring Communication



process 1 (send)

```
// 1st iter.
start receiving
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

process 2 (recv.)

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

overlap btw. comm. and comp.

send buf. P : Packed P : Sending

recv. buf. R : Receiving, R : Receiving done, U : being Used

Neighboring Communication, cont'd

If the receiving process delays....

send buffer

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

recv. buffer

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk

wait for the boundary data comes

computation: boundary


wait for sending is done


// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

Neighboring Communication, cont'd

If the receiving process delays....

send buffer

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

recv. buffer

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk

wait for the boundary data comes

computation: boundary


wait for sending is done


// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

Neighboring Communication, cont’d

If the receiving process delays....

send buffer

P

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

recv. buffer

R

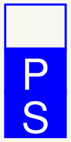
```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk

wait for the boundary data comes

computation: boundary


wait for sending is done


// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

Neighboring Communication, cont'd

If the receiving process delays....

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

send buffer

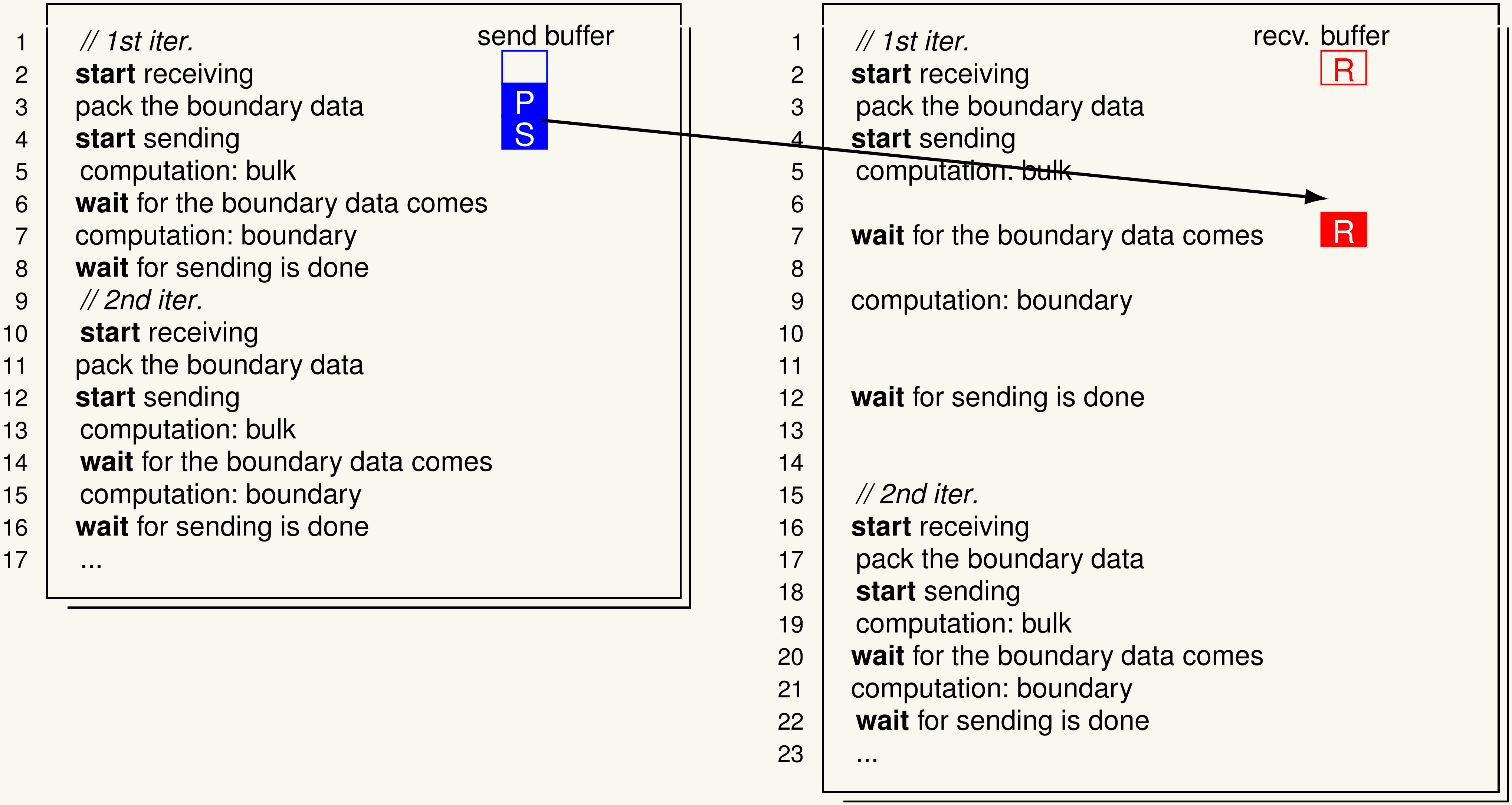


```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk

wait for the boundary data comes

computation: boundary


wait for sending is done


// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

recv. buffer

Neighboring Communication, cont’d

If the receiving process delays....

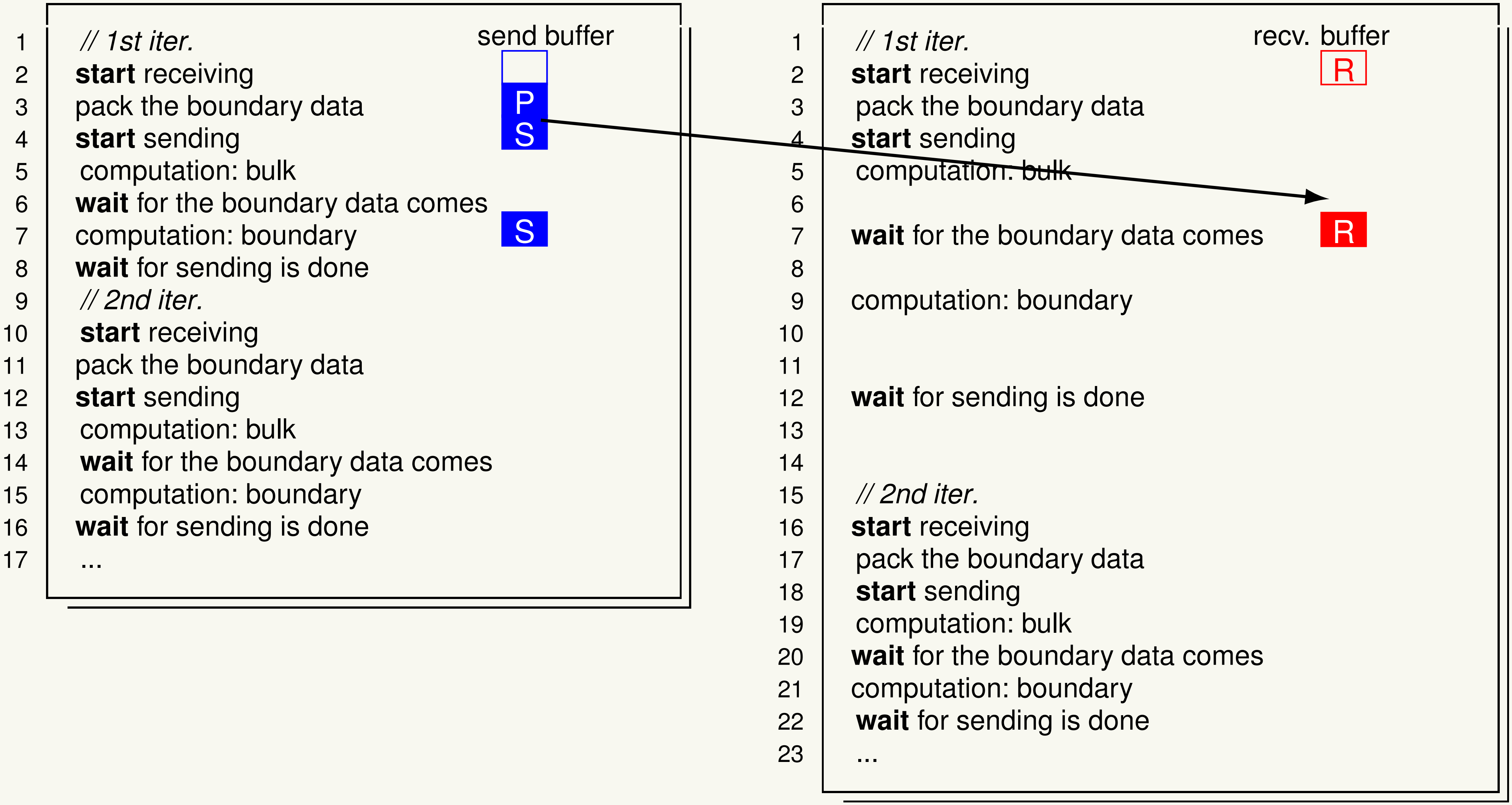


```
// 1st iter.
start receiving
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
 start receiving
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
wait for sending is done
 ...
```

```
// 1st iter.
start receiving
 pack the boundary data
start sending
 computation: bulk

wait for the boundary data comes

computation: boundary


wait for sending is done


// 2nd iter.
start receiving
 pack the boundary data
 start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 ...
```

Neighboring Communication, cont’d

If the receiving process delays....

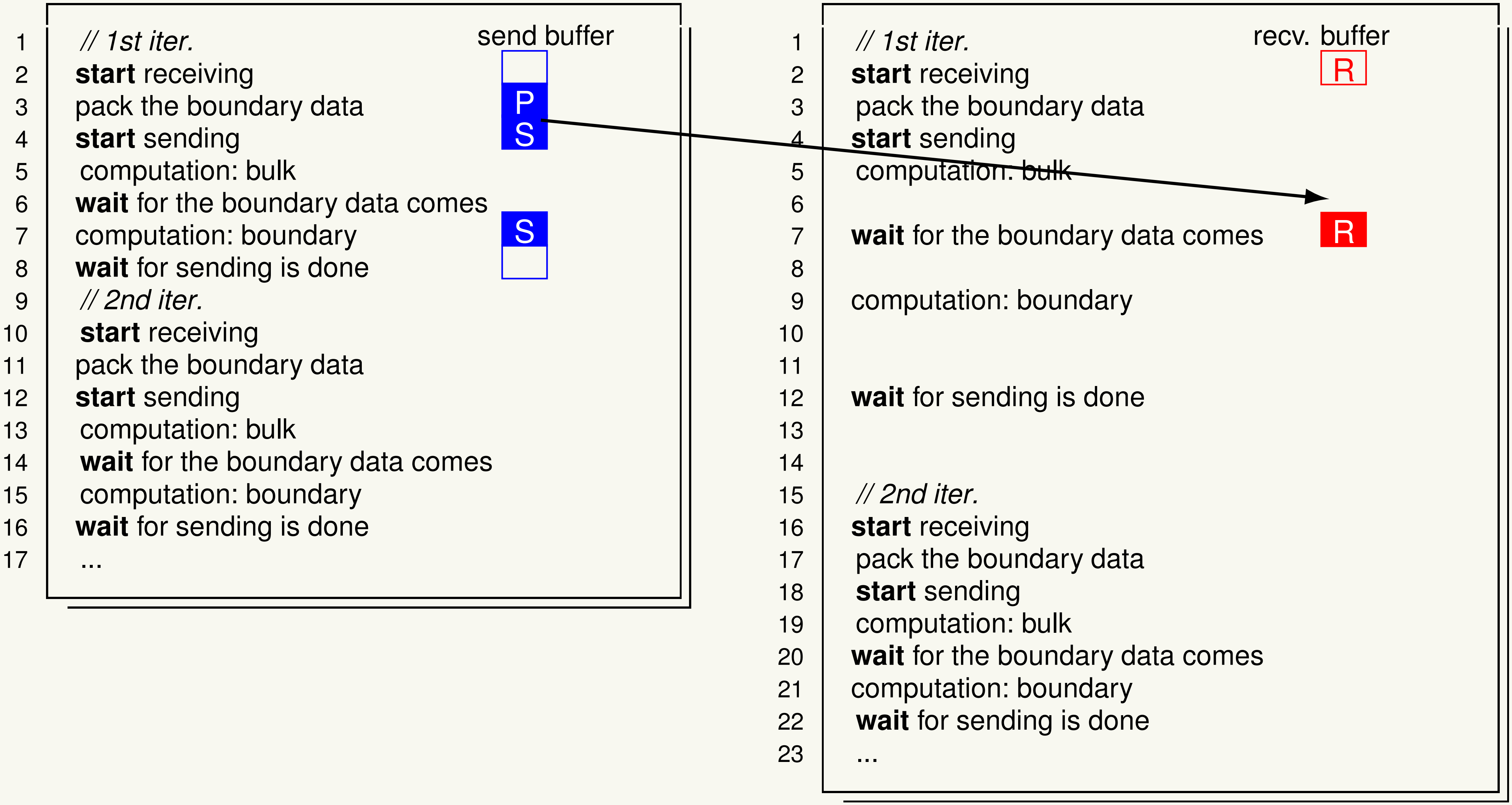


```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk

wait for the boundary data comes

computation: boundary


wait for sending is done


// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

Neighboring Communication, cont’d

If the receiving process delays....

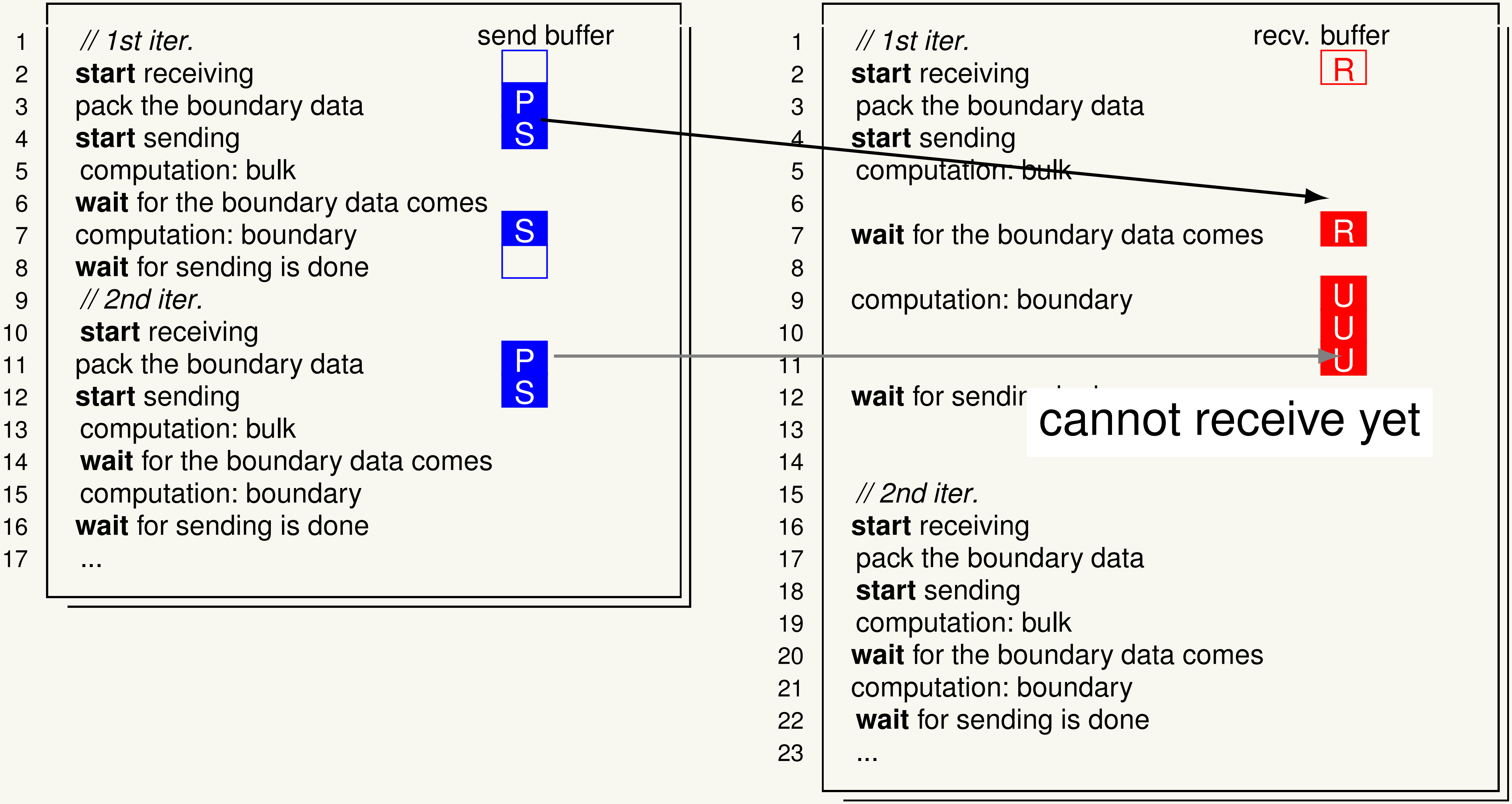


```
// 1st iter.
start receiving
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
 start receiving
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
wait for sending is done
 ...
```

```
// 1st iter.
start receiving
 pack the boundary data
start sending
 computation: bulk

wait for the boundary data comes

computation: boundary


wait for sending is done


// 2nd iter.
start receiving
 pack the boundary data
 start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 ...
```

Neighboring Communication, cont’d

If the receiving process delays....

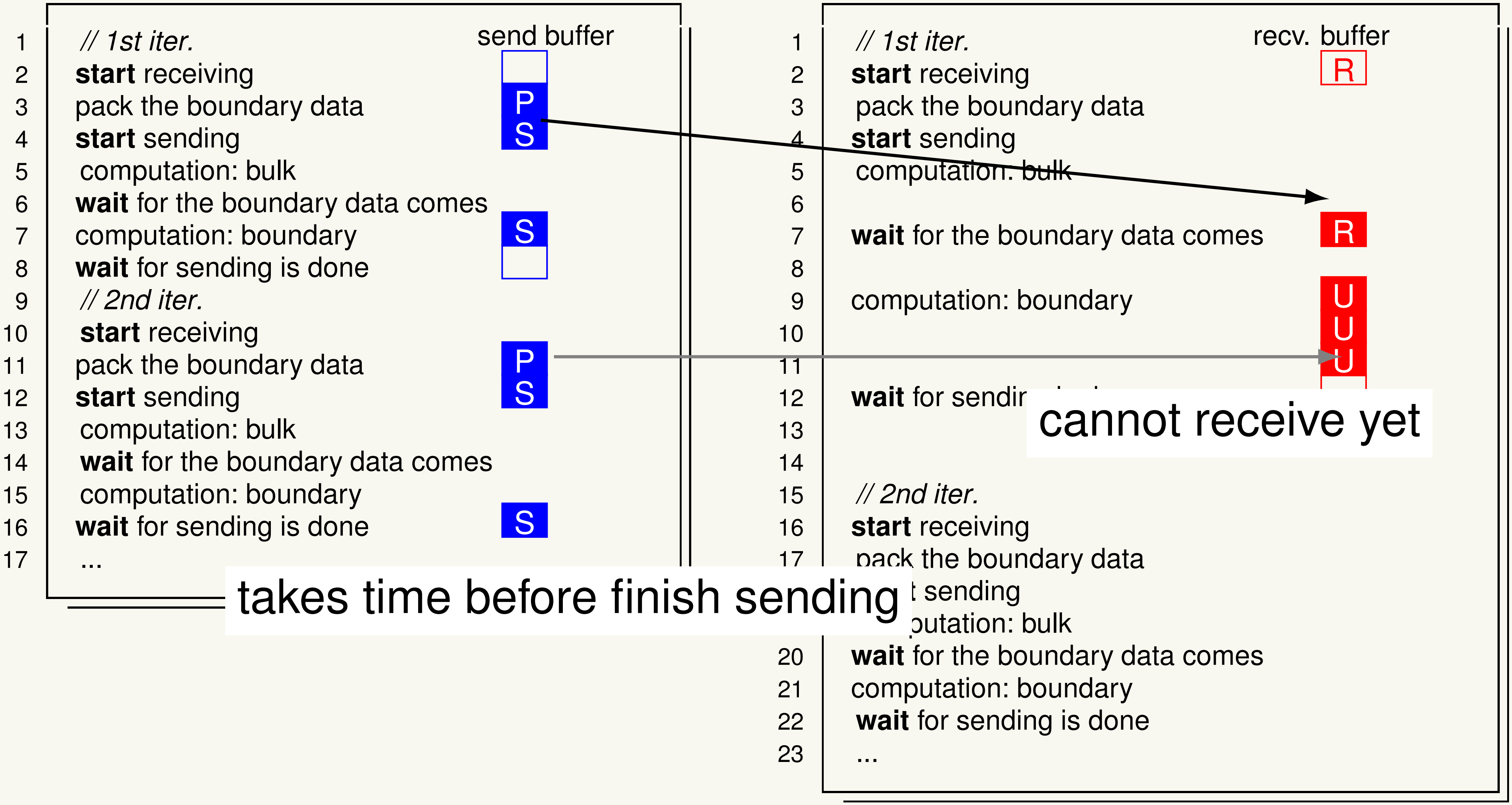


```
// 1st iter.
start receiving
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
 // 2nd iter.
 start receiving
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
wait for sending is done
 ...
```

```
// 1st iter.
start receiving
 pack the boundary data
start sending
 computation: bulk

wait for the boundary data comes

computation: boundary


wait for sendir


// 2nd iter.
start receiving
 pack the boundary data
 start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 ...
```

Neighboring Communication, cont’d

If the receiving process delays....



```
// 1st iter.
start receiving
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
 start receiving
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
wait for sending is done
 ...
```

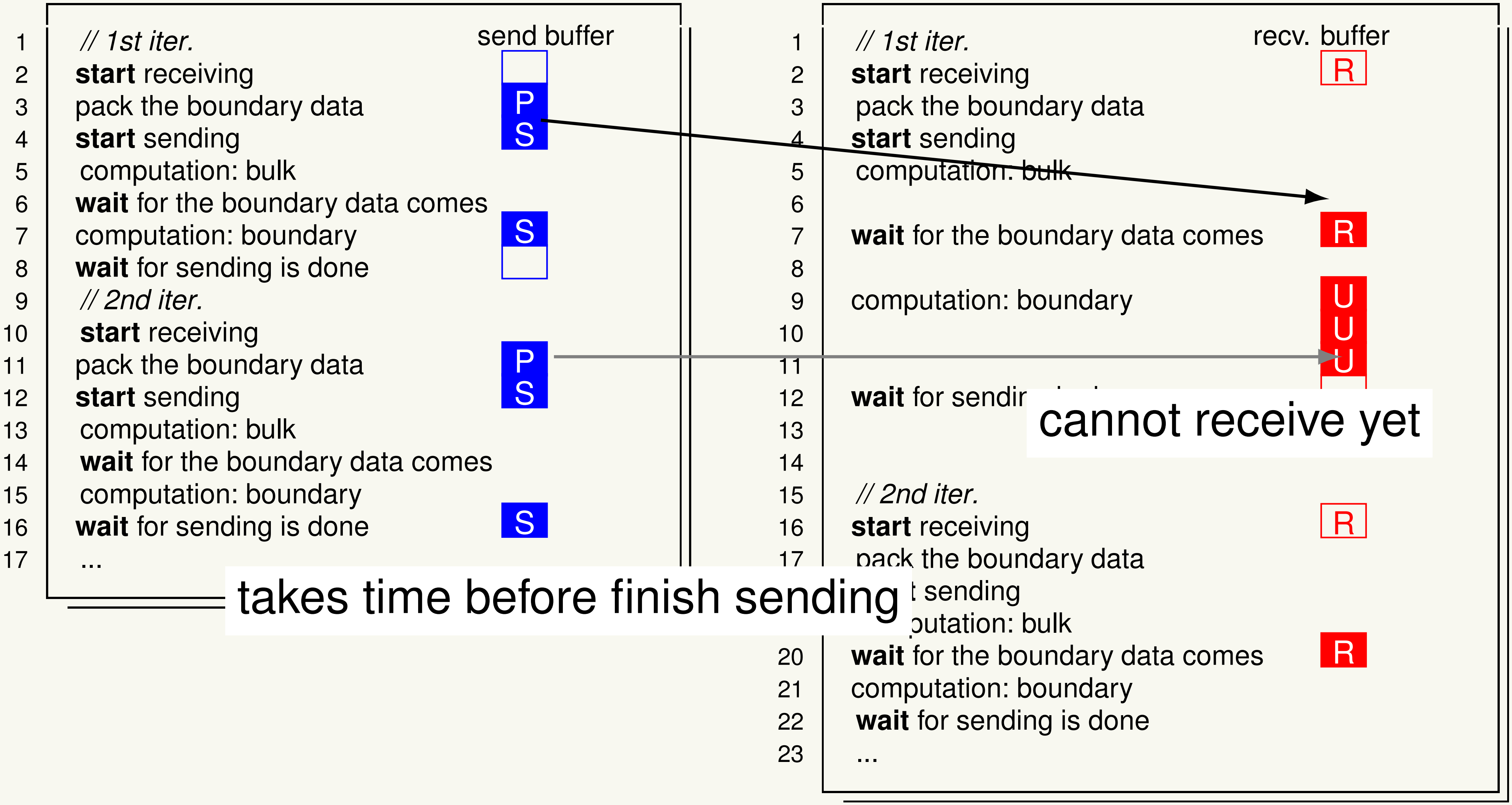
```
// 1st iter.
start receiving
 pack the boundary data
start sending
 computation: bulk

wait for the boundary data comes

computation: boundary


wait for sendir



// 2nd iter.
start receiving
 pack the boundary data
t sending
putation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 ...
```

cannot receive yet

takes time before finish sending

Neighboring Communication, cont'd

If the receiving process delays....



```
// 1st iter.
start receiving
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
 // 2nd iter.
 start receiving
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
wait for sending is done
 ...
```

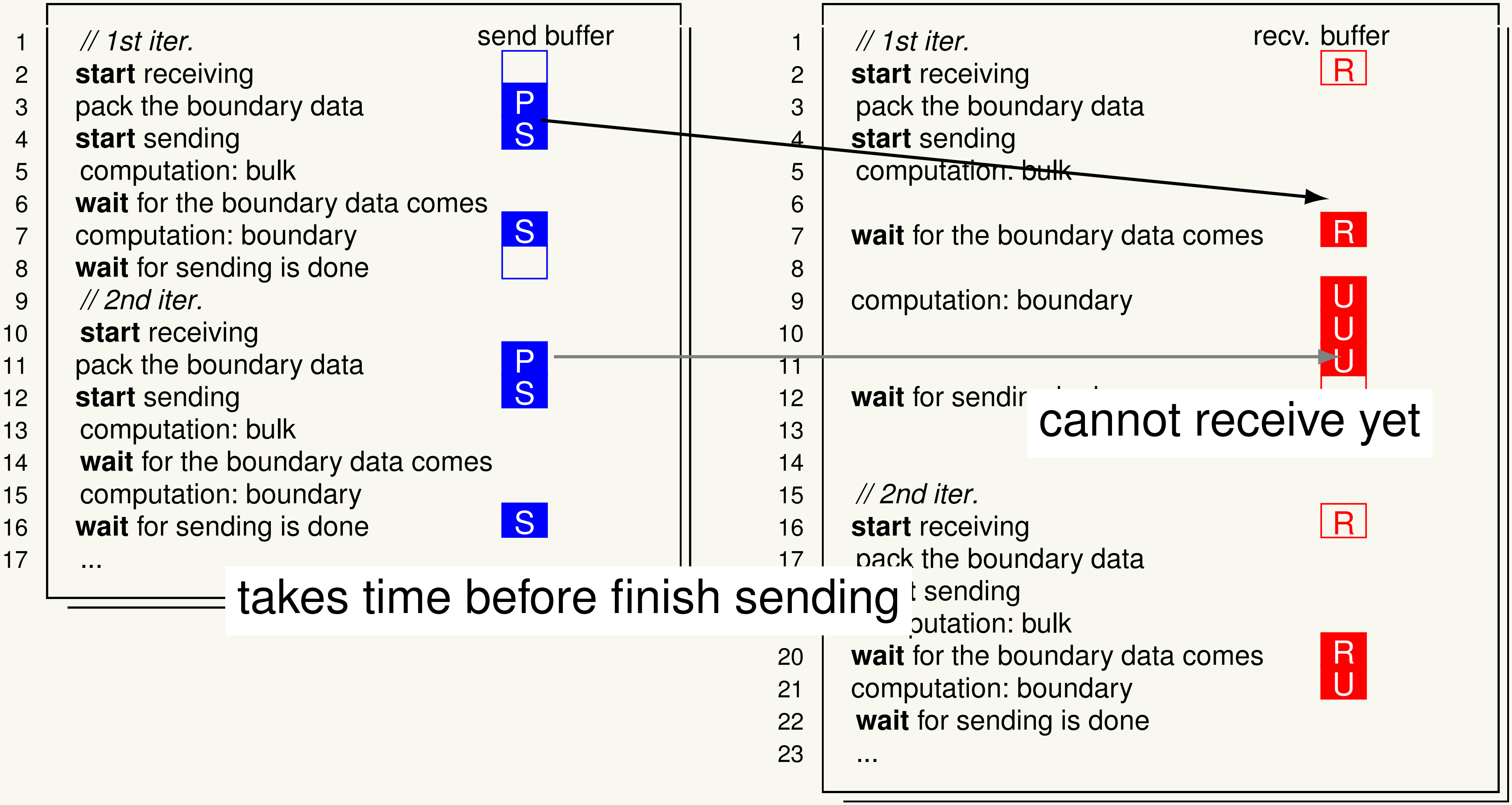
```
// 1st iter.
start receiving
 pack the boundary data
start sending
 computation: bulk

wait for the boundary data comes

computation: boundary


wait for sendir


 // 2nd iter.
start receiving
 pack the boundary data
t sending
putation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 ...
```

cannot receive yet

takes time before finish sending

Neighboring Communication, cont’d

If the receiving process delays....



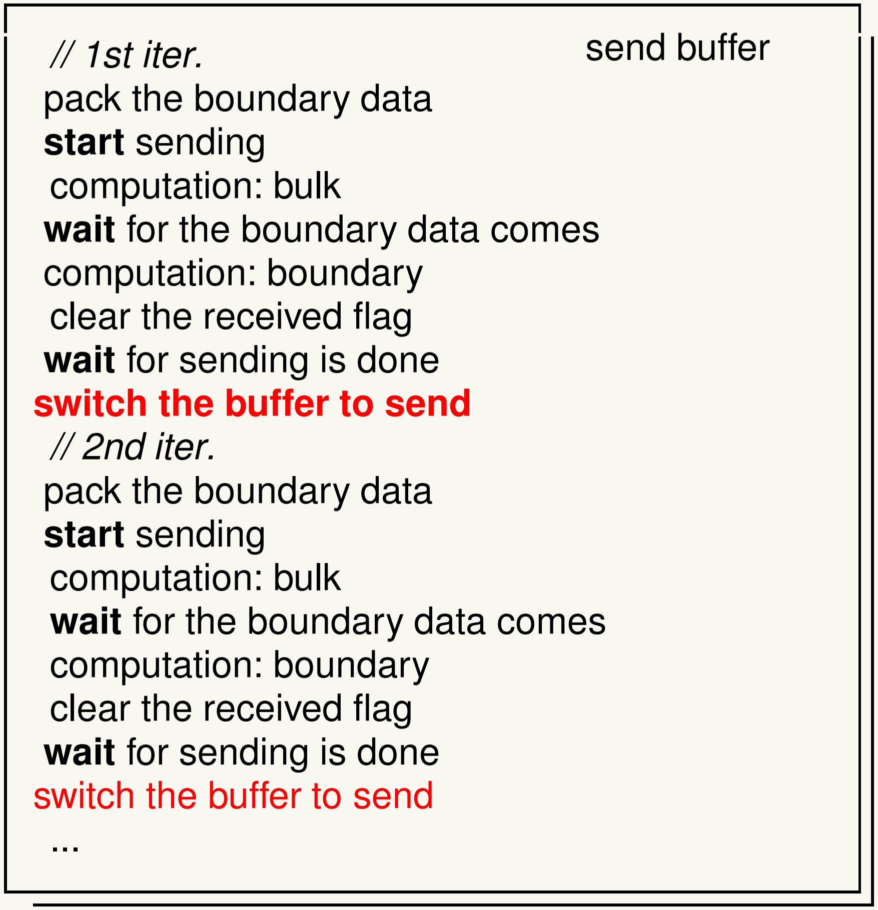
```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
// 2nd iter.
start receiving
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

```
// 1st iter.
start receiving
pack the boundary data
start sending
computation: bulk

wait for the boundary data comes

computation: boundary


wait for sendir


// 2nd iter.
start receiving
pack the boundary data
t sending
putation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
...
```

cannot receive yet

takes time before finish sending

Neighboring Communication: Double Buffering (RDMA)



```
 // 1st iter.
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 // 2nd iter.
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 ...
```

Neighboring Communication: Double Buffering (RDMA)

send buffer

```
  // 1st iter.
 pack the boundary data
 start sending
  computation: bulk
 wait for the boundary data comes
 computation: boundary
  clear the received flag
 wait for sending is done
switch the buffer to send
  // 2nd iter.
 pack the boundary data
 start sending
  computation: bulk
  wait for the boundary data comes
  computation: boundary
  clear the received flag
 wait for sending is done
switch the buffer to send
  ...
```

recv. buffers

```
  // 1st iter.
 pack the boundary data

 start sending
 computation: bulk


 wait for the boundary data comes
 computation: boundary



 clear the received flag
 wait for sending is done


switch the buffer to send
  // 2nd iter.
  pack the boundary data
  start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
  wait for sending is done
  switch the buffer to send
  ...
```

Neighboring Communication: Double Buffering (RDMA)

send buffer

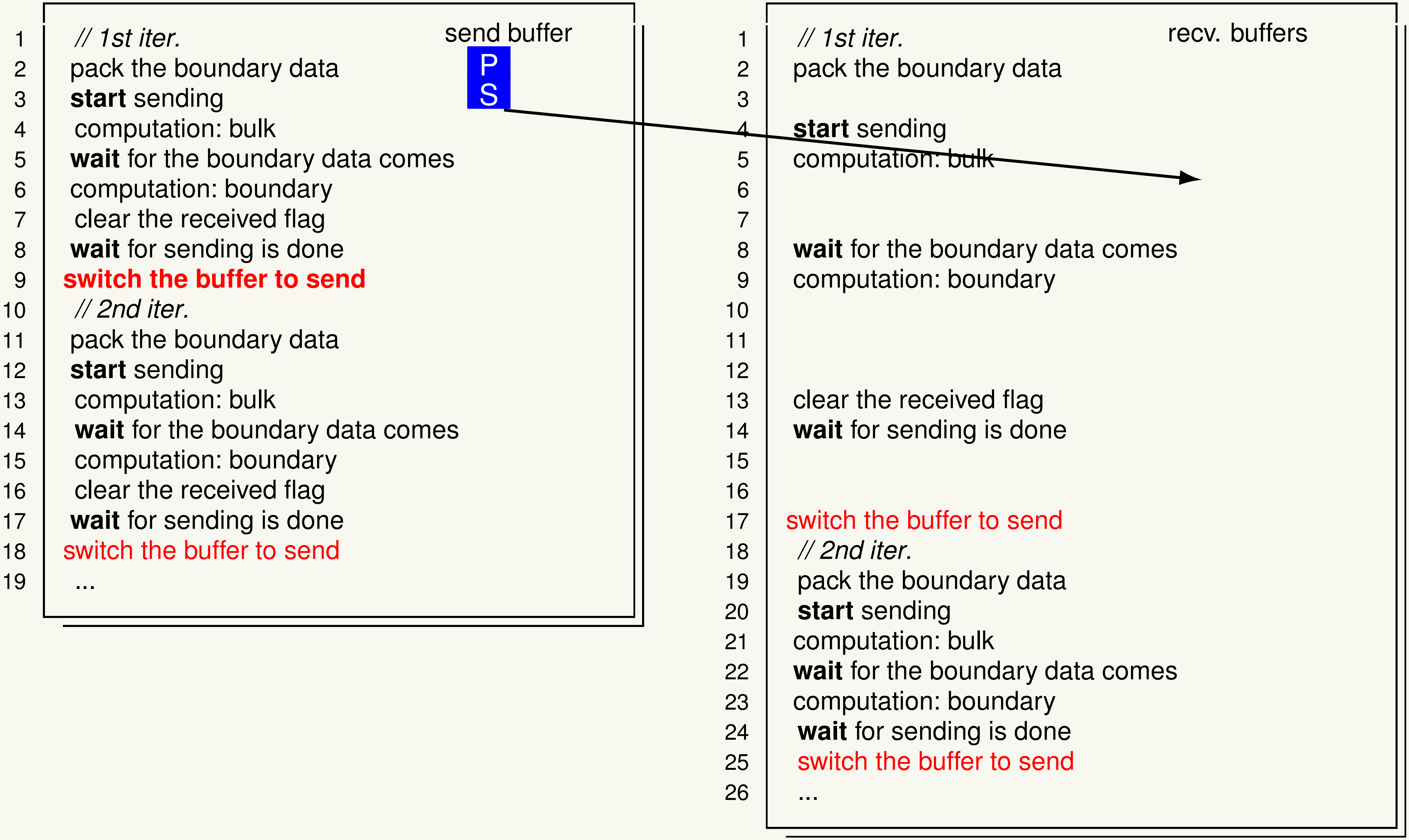
P

```
// 1st iter.
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
clear the received flag
wait for sending is done
switch the buffer to send
// 2nd iter.
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
clear the received flag
wait for sending is done
switch the buffer to send
...
```

recv. buffers

```
// 1st iter.
pack the boundary data

start sending
computation: bulk


wait for the boundary data comes
computation: boundary



clear the received flag
wait for sending is done


switch the buffer to send
// 2nd iter.
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
switch the buffer to send
...
```

Neighboring Communication: Double Buffering (RDMA)

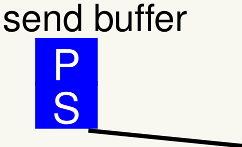


```
// 1st iter.
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 // 2nd iter.
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 ...
```

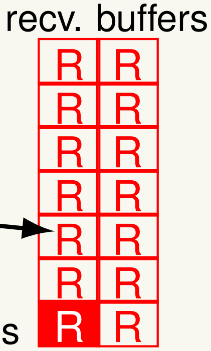
```
// 1st iter.
pack the boundary data

start sending
computation: bulk


wait for the boundary data comes
computation: boundary



clear the received flag
wait for sending is done


switch the buffer to send
 // 2nd iter.
 pack the boundary data
 start sending
computation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 switch the buffer to send
 ...
```

Neighboring Communication: Double Buffering (RDMA)

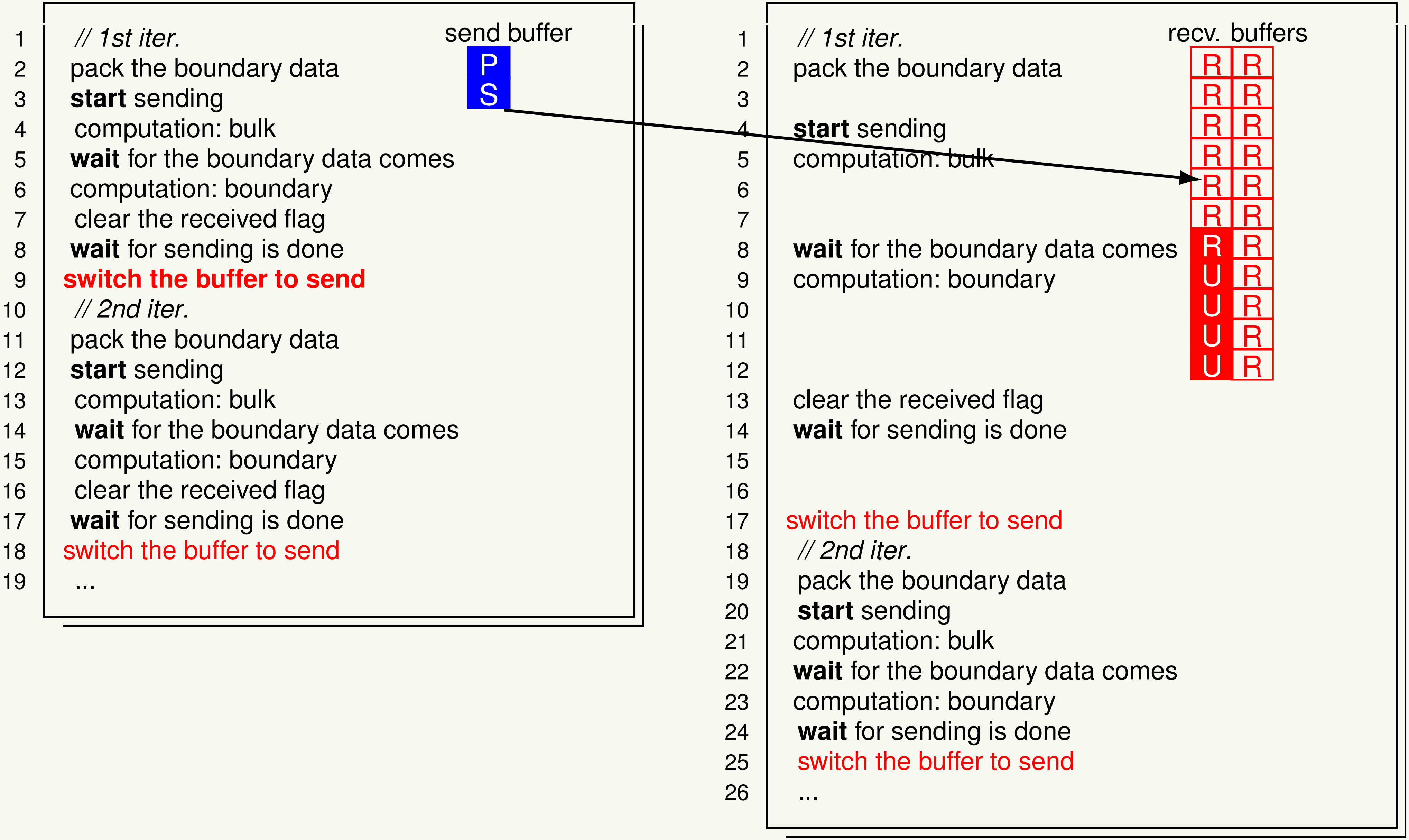
```
// 1st iter.
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
clear the received flag
wait for sending is done
switch the buffer to send
// 2nd iter.
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
clear the received flag
wait for sending is done
switch the buffer to send
...
```



```
// 1st iter.
pack the boundary data

start sending
computation: bulk


wait for the boundary data comes
computation: boundary



clear the received flag
wait for sending is done


switch the buffer to send
// 2nd iter.
pack the boundary data
start sending
computation: bulk
wait for the boundary data comes
computation: boundary
wait for sending is done
switch the buffer to send
...
```



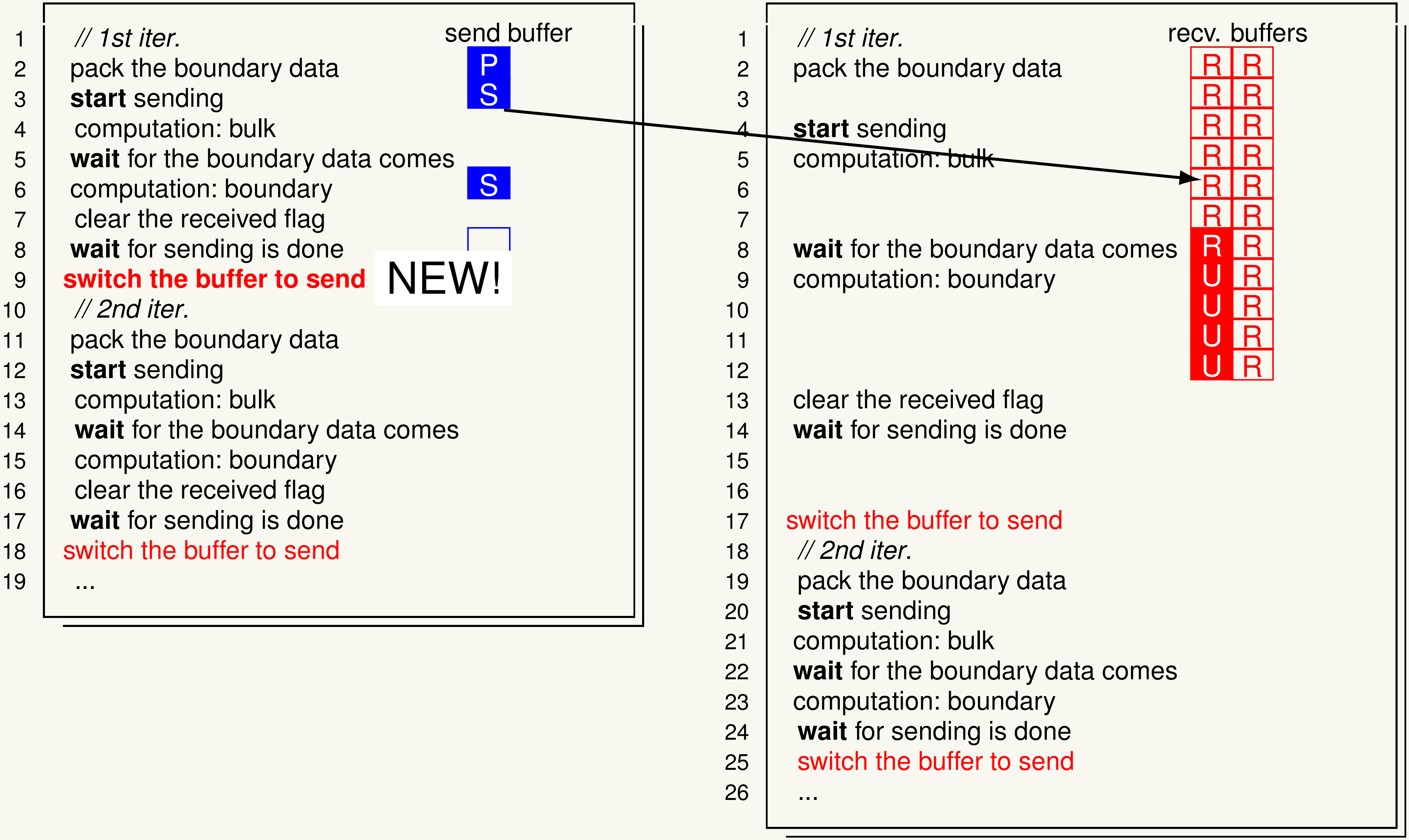
Neighboring Communication: Double Buffering (RDMA)



```
// 1st iter.
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 // 2nd iter.
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 ...
```

```
// 1st iter.
pack the boundary data

start sending
computation: bulk


wait for the boundary data comes
computation: boundary



clear the received flag
wait for sending is done


switch the buffer to send
 // 2nd iter.
 pack the boundary data
 start sending
computation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 switch the buffer to send
 ...
```

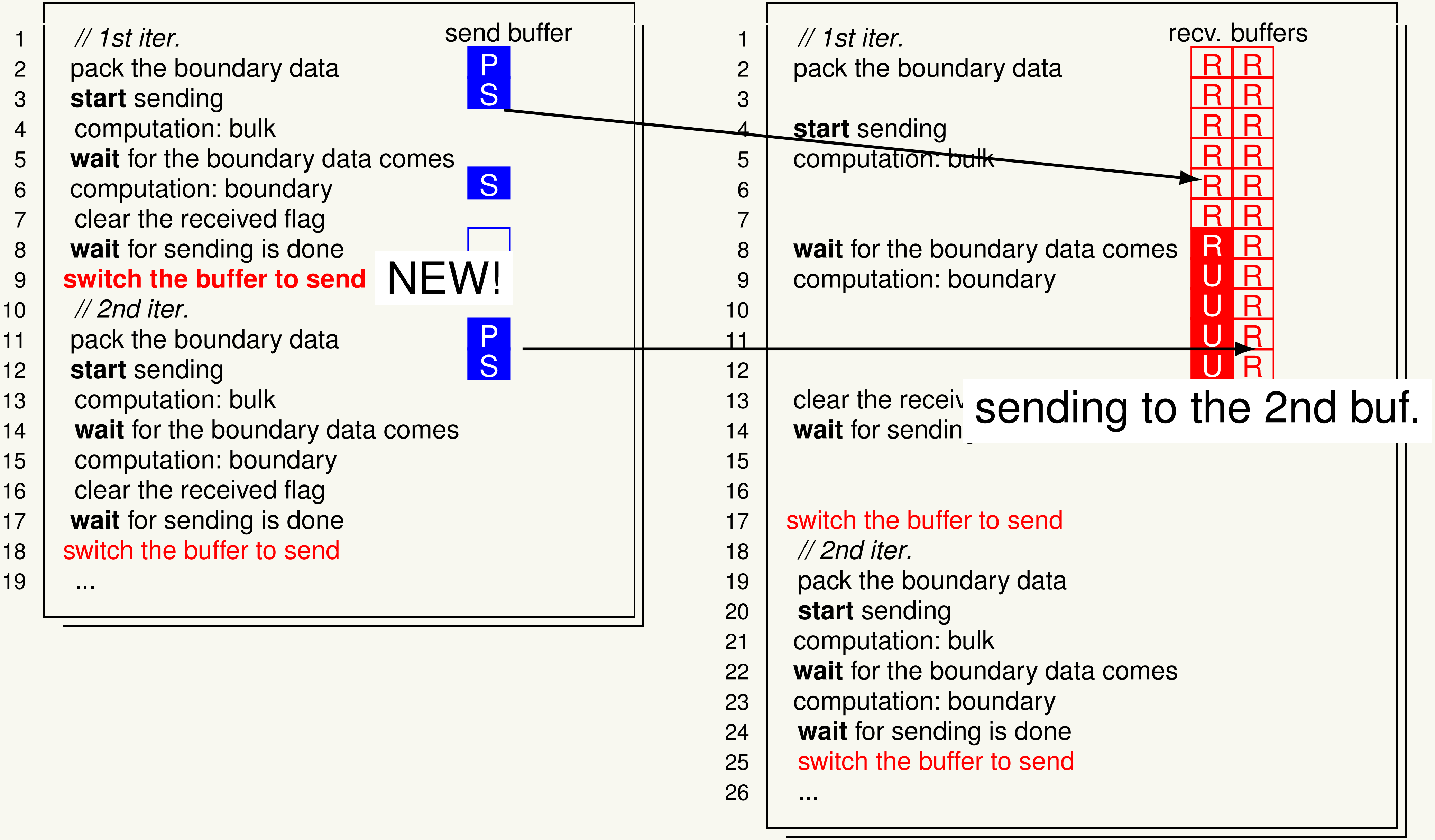
Neighboring Communication: Double Buffering (RDMA)



```
// 1st iter.
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 // 2nd iter.
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 ...
```

```
// 1st iter.
pack the boundary data

start sending
computation: bulk


wait for the boundary data comes
computation: boundary



clear the received flag
wait for sending is done


switch the buffer to send
 // 2nd iter.
 pack the boundary data
 start sending
computation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 switch the buffer to send
 ...
```

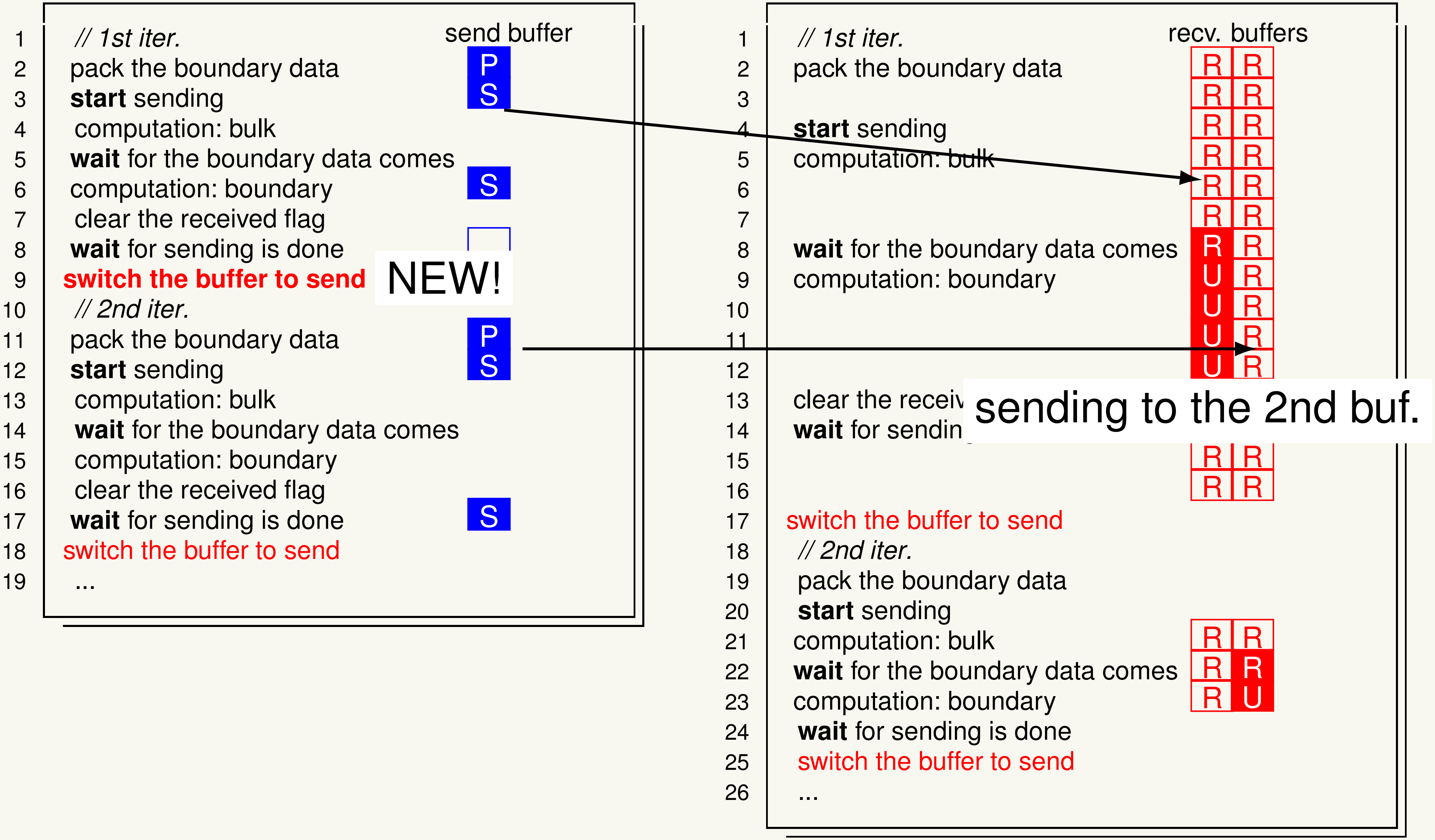
Neighboring Communication: Double Buffering (RDMA)



```
 // 1st iter.
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 // 2nd iter.
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 ...
```

```
 // 1st iter.
pack the boundary data

start sending
computation: bulk


wait for the boundary data comes
computation: boundary



clear the receiv
wait for sendin


switch the buffer to send
 // 2nd iter.
 pack the boundary data
 start sending
computation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 switch the buffer to send
 ...
```

Neighboring Communication: Double Buffering (RDMA)



```
// 1st iter.
pack the boundary data
start sending
 computation: bulk
wait for the boundary data comes
computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 // 2nd iter.
pack the boundary data
start sending
 computation: bulk
 wait for the boundary data comes
 computation: boundary
 clear the received flag
wait for sending is done
switch the buffer to send
 ...
```

NEW!

```
// 1st iter.
pack the boundary data

start sending
computation: bulk


wait for the boundary data comes
computation: boundary



clear the receiv
wait for sendin


switch the buffer to send
 // 2nd iter.
 pack the boundary data
 start sending
computation: bulk
wait for the boundary data comes
computation: boundary
 wait for sending is done
 switch the buffer to send
 ...
```

sending to the 2nd buf.

Remote Direct Memory Access (RDMA)

we use “put” in sending

it directly writes to the memory on the remote process

- send: put (directly memory on the remote process)

  boundary data + watchdog flag

- Wait (recv.): check the flag is updated

  after the boundary computation, the flag is reset

Remote Direct Memory Access (RDMA)

we use “put” in sending
it directly writes to the memory on the remote process

- send: put (directly memory on the remote process)
  boundary data + watchdog flag
- Wait (recv.): check the flag is updated
  after the boundary computation, the flag is reset

NOTE 1: 2 buffers are enough: “sending proc.” also receives data from “receiving proc.”⇒automatic synchronization

Remote Direct Memory Access (RDMA)

we use “put” in sending

it directly writes to the memory on the remote process

- send: put (directly memory on the remote process)

  boundary data + watchdog flag
- Wait (recv.): check the flag is updated

  after the boundary computation, the flag is reset

NOTE 1: 2 buffers are enough: “sending proc.” also receives data from “receiving proc.”⇒automatic synchronization

NOTE 2: one can alternatively use MPI (persistent) communication to implement double buffering.

Benchmark

Test detail: Jacobi method for 2-dim system

base: `http://theo.phys.sci.hiroshima-u.ac.jp/~ishikawa/APL9WG/stencil_double_buffering_mpi-1.0.tar.gz`

target system: $Mx = b$ with

$$(Mx)(i,j) =$$
$$\underbrace{(4+m^2)x(i,j)}_{\equiv Dx}\underbrace{-x(i+1,j) - x(i-1,j) - x(i,j+1) - x(i,j-1)}_{\equiv Hx}$$

$$\xrightarrow{\text{cont. limit}} (m^2 - \partial^2)x$$

Jacobi method

$$x^{(k)} \to x^{(k+1)} = D^{-1}(b - Hx^{(k)})$$

Only the hopping H contains the communication

- fixed number of iterations: 10
- local lattice size: 60×60
- communication buffer: needed size + dummy (+ flag)

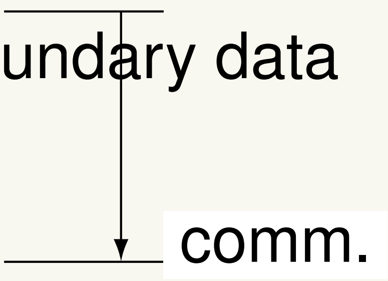
$$s = \frac{\text{needed} + \text{dummy}}{\text{needed}}, 1 \le s \le 8192$$

Hopping (Mult of H)

1. packing the boundary data
2. start sending/receiving the boundary data
3. calculate: internal area
4. wait for receiving
5. calculate: boundary area
6. wait for sending finished

Hopping (Mult of H)

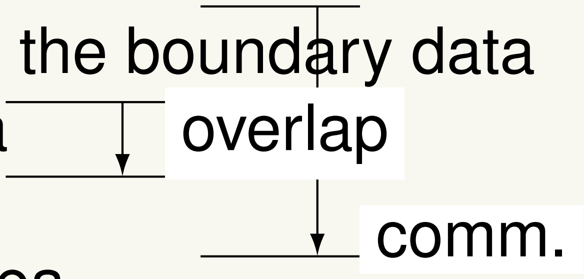
1. packing the boundary data
2. start sending/receiving the boundary data
3. calculate: internal area
4. wait for receiving
5. calculate: boundary area
6. wait for sending finished



comm.

Hopping (Mult of H)

1. packing the boundary data
2. start sending/receiving the boundary data
3. calculate: internal area
4. wait for receiving
5. calculate: boundary area
6. wait for sending finished

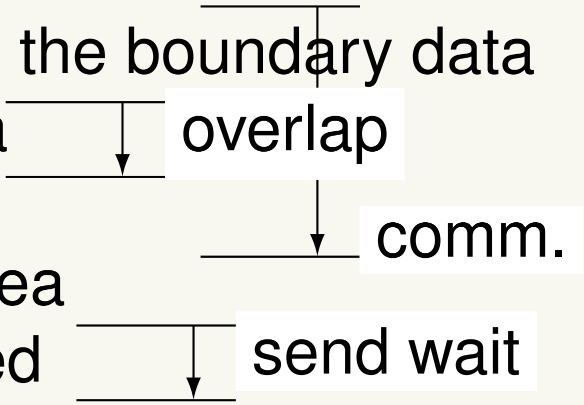


overlap

comm.

Hopping (Mult of H)

1. packing the boundary data
2. start sending/receiving the boundary data
3. calculate: internal area
4. wait for receiving
5. calculate: boundary area
6. wait for sending finished



Hopping (Mult of H)

1. packing the boundary data
2. start sending/receiving the boundary data
3. calculate: internal area
4. wait for receiving
5. calculate: boundary area
6. wait for sending finished

overlap

comm.

send wait

non-overlap = comm. $-$ overlap
= start sending/receiving + wait for receiving

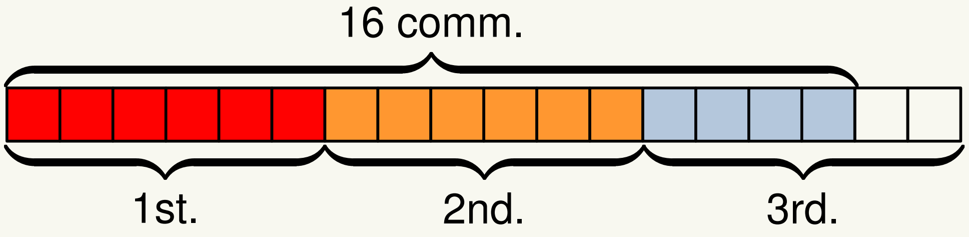
Estimated Bandwidth and uTofu Interface

uTofu

- Low level interface to use Tofu Interconnect
- It allows to specify Tofu Network Interface (TNI) to use
  tuning with the optimal TNI assignment for QCD
- 6 TNI/node, 6.8GB/s for each TNI

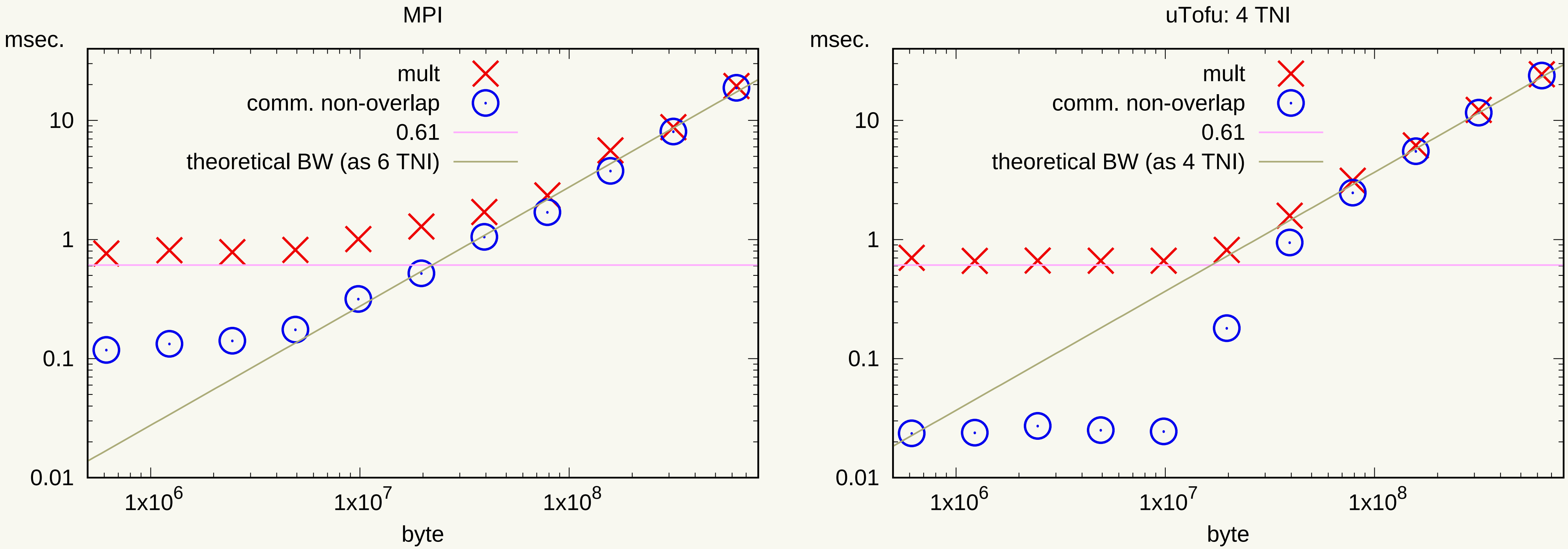
Bandwidth estimate

- 1 node with 4 MPI proc.
  $\Rightarrow$(4 directions) × (4 ranks) = 16 comm.
  (each comm. has the same size)
- Using 4 TNI: each TNI is used 4 times
  $6.8 \times 4 = 27.2$ GB/s
- Using 6 TNI: each TNI is used 2 or 3 times
  $6.8 \times 6 \times \frac{16}{18} = 36.3$ GB/s



Performance: mult of H on A64fx 1 node

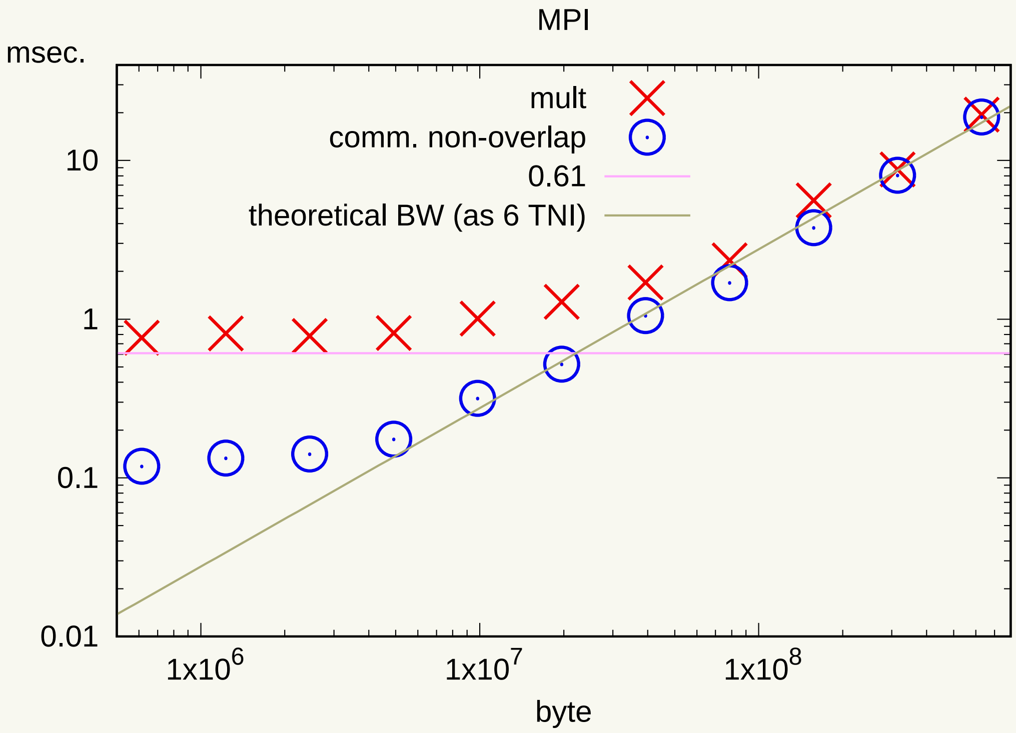
NOTE: result on the evaluation environment, it does not guarantee the performance on the actual Fugaku

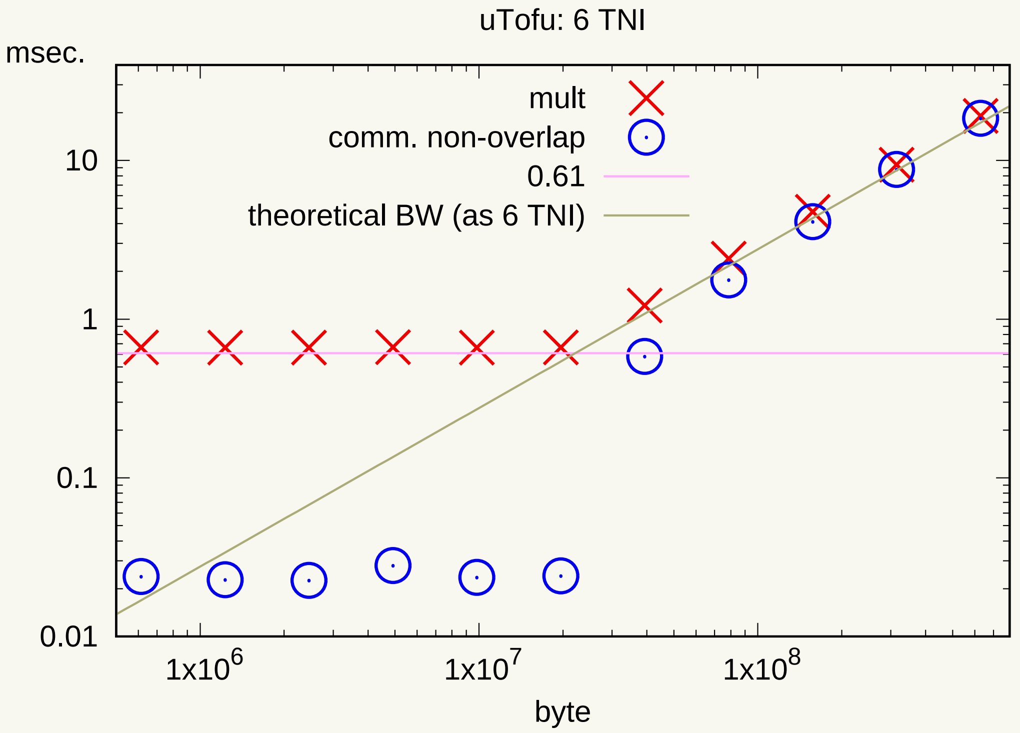


- comm. overlap $\simeq$ 0.61 msec. (replaced with a line in the plots)
- both show good scaling for large communication data size
- uTofu interface has a smaller overhead
- TNI for uTofu — 0:$+x$, 1:$-x$, 2:$+y$, 3:$-y$

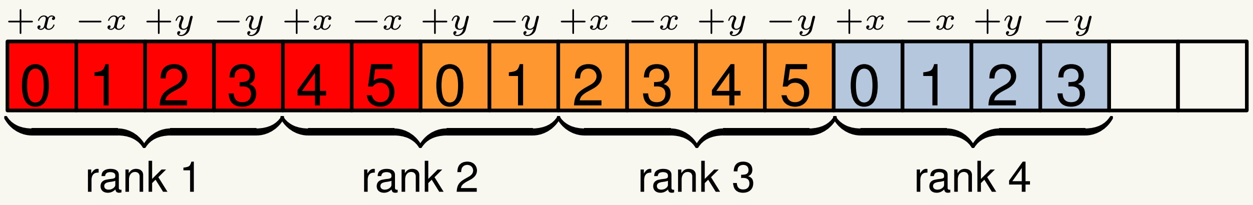
Performance: mult of H on A64fx 1 node

NOTE: result on the evaluation environment, it does not guarantee the performance on the actual Fugaku



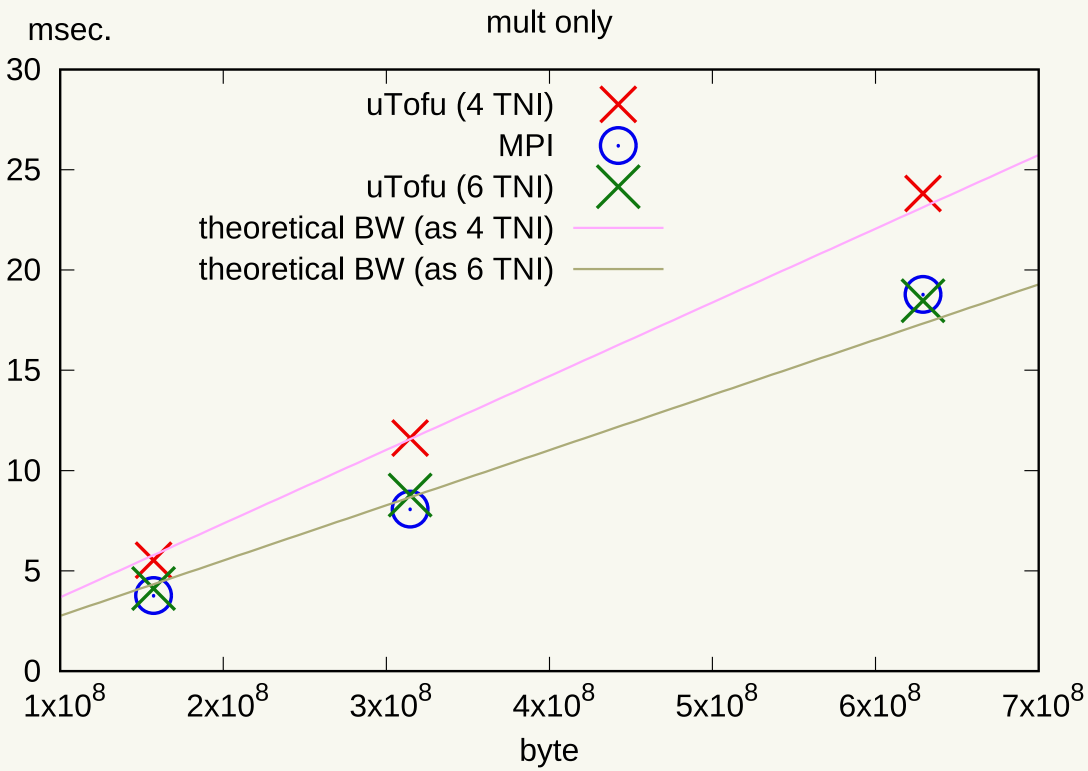


- comm. overlap $\simeq$ 0.61 msec. (replaced with a line in the plots)
- both show good scaling for large communication data size
- uTofu interface has a smaller overhead
- TNI for uTofu


$+x$ $-x$ $+y$ $-y$ $+x$ $-x$ $+y$ $-y$ $+x$ $-x$ $+y$ $-y$ $+x$ $-x$ $+y$ $-y$
0 1 2 3 4 5 0 1 2 3 4 5 0 1 2 3
rank 1
rank 2
rank 3
rank 4

Performance: mult of H on A64fx 1 node, cont'd

NOTE: result on the evaluation environment, it does not guarantee the performance on the actual Fugaku



large comm. size
$\Rightarrow$time for mult
$\simeq$ time for comm.

saturation of the network bandwidth

- MPI: 32.4 GB/s
- uTofu (4 TNI): 25.8 GB/s — cf. $6.8 \times 4 = 27.2$ GB/s
- uTofu (6 TNI): 33.0 GB/s — cf. $6.8 \times 16/18 = 36.3$ GB/s

Conclusions

Conclusions

to accelerate neighboring communication, we have implemented double buffering algorithm

test with a simple 2-dim system

- using uTofu interface seems promising

Future: to do (or on going) for Fugaku

- implement double buffering+uTofu to QCD code

  qws: Nakamura-san's talk
- multi nodes, proper TNI settings,...
- official predicted performance for LQCD (vs. K-computer): x25+

  "+" will be how much????