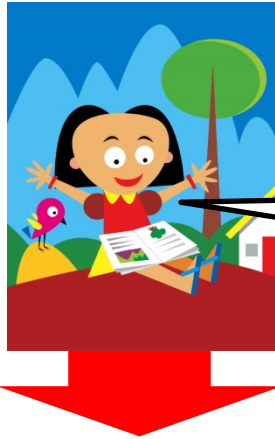


A Fast Window Join over Data Streams

Hideyuki Kawashima

Presented at
ICNC'11, FPL'11, MCSoc'12, SSDBM'13.

Data Stream Management System (DSMS)



How many packets are
arrived for port 80
in a minute ?

```
SELECT COUNT(*)  
FROM   eth0[TIME 1 MIN]  
WHERE  port = 80
```

20

Relational schema

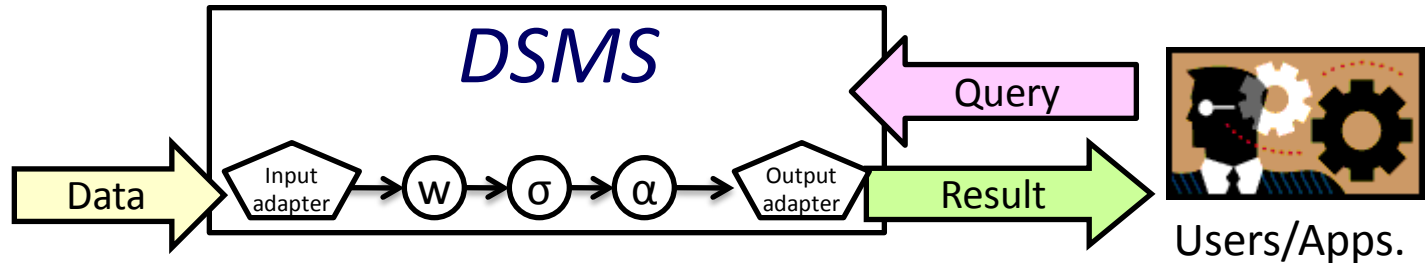
- Destination IP
- Source IP
- Destination Port
- Source Port
- Interface (e.g. eth0)
- Length
- Version (e.g. IPV4)
- Payload

Q₁

Relation
eth0

DSMS

```
SELECT COUNT(*)  
FROM   eth0[TIME 1 MIN]  
WHERE  port = 80
```



- SQL is translated to operator tree.
- On arrival of data, tree is evaluated.
- Operators are based on relational database
 - w (Window): Cutting off relations from a stream
 - σ (Selection): Filter
 - α (Aggregation): such as AVG, MIN, MAX

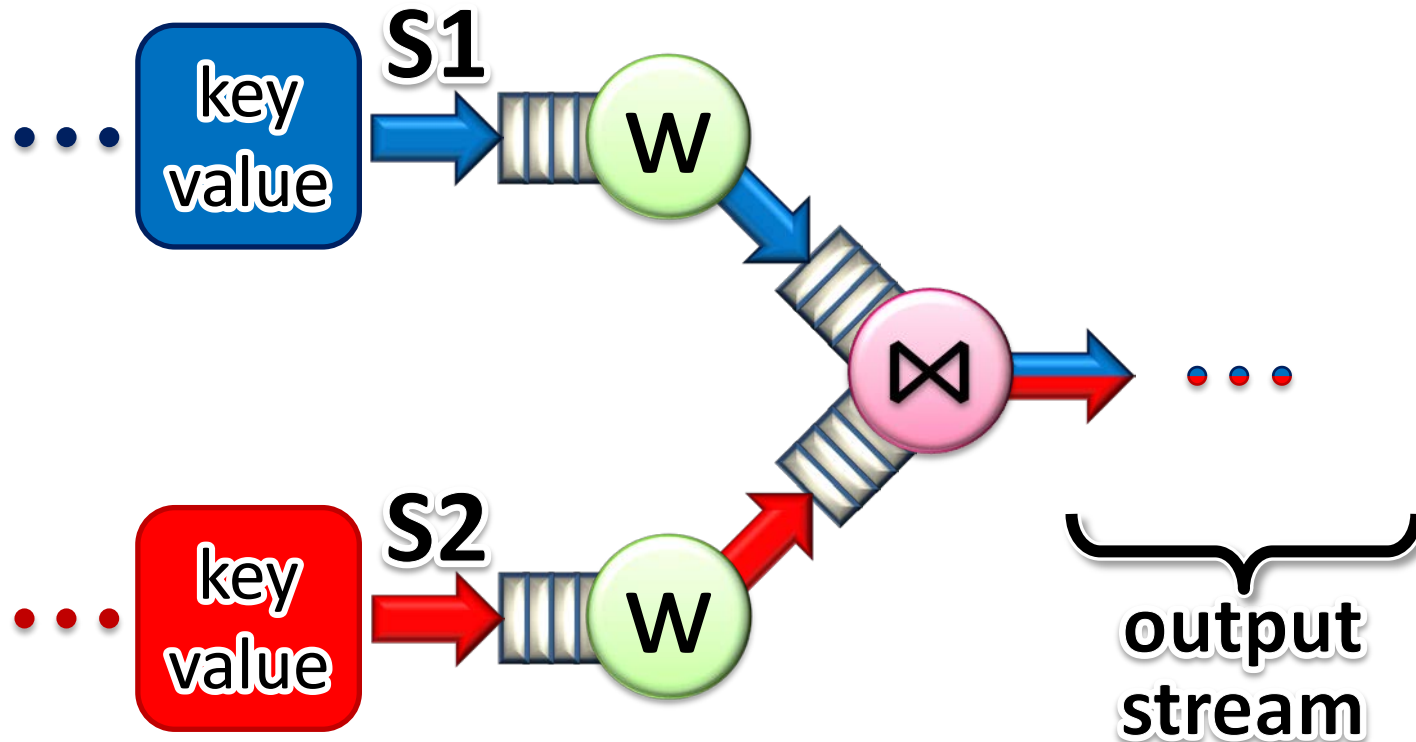
Window Join

SELECT *

FROM S1 [Rows 100], S2 [Rows 100]

WHERE S1.key = S2.key

CQL
Query





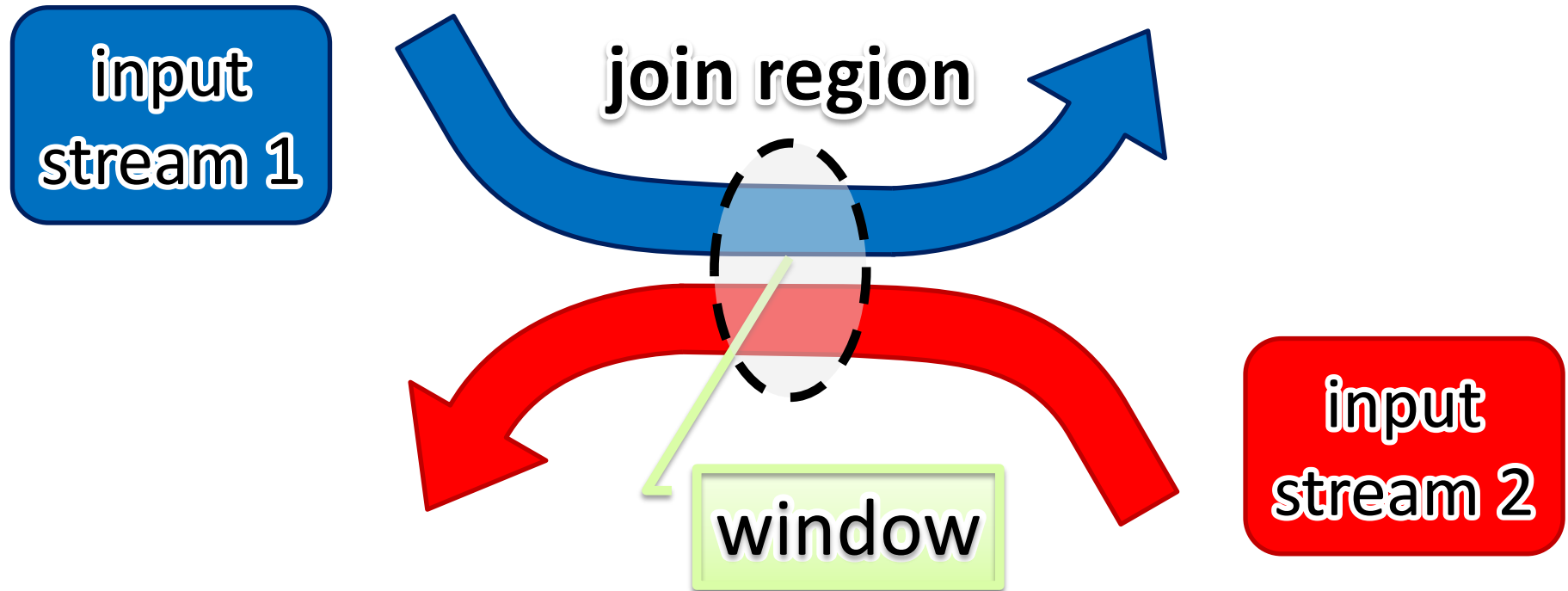
HANDSHAKE JOIN

J. Teubner and R. Müller, SIGMOD'11

Handshake Join

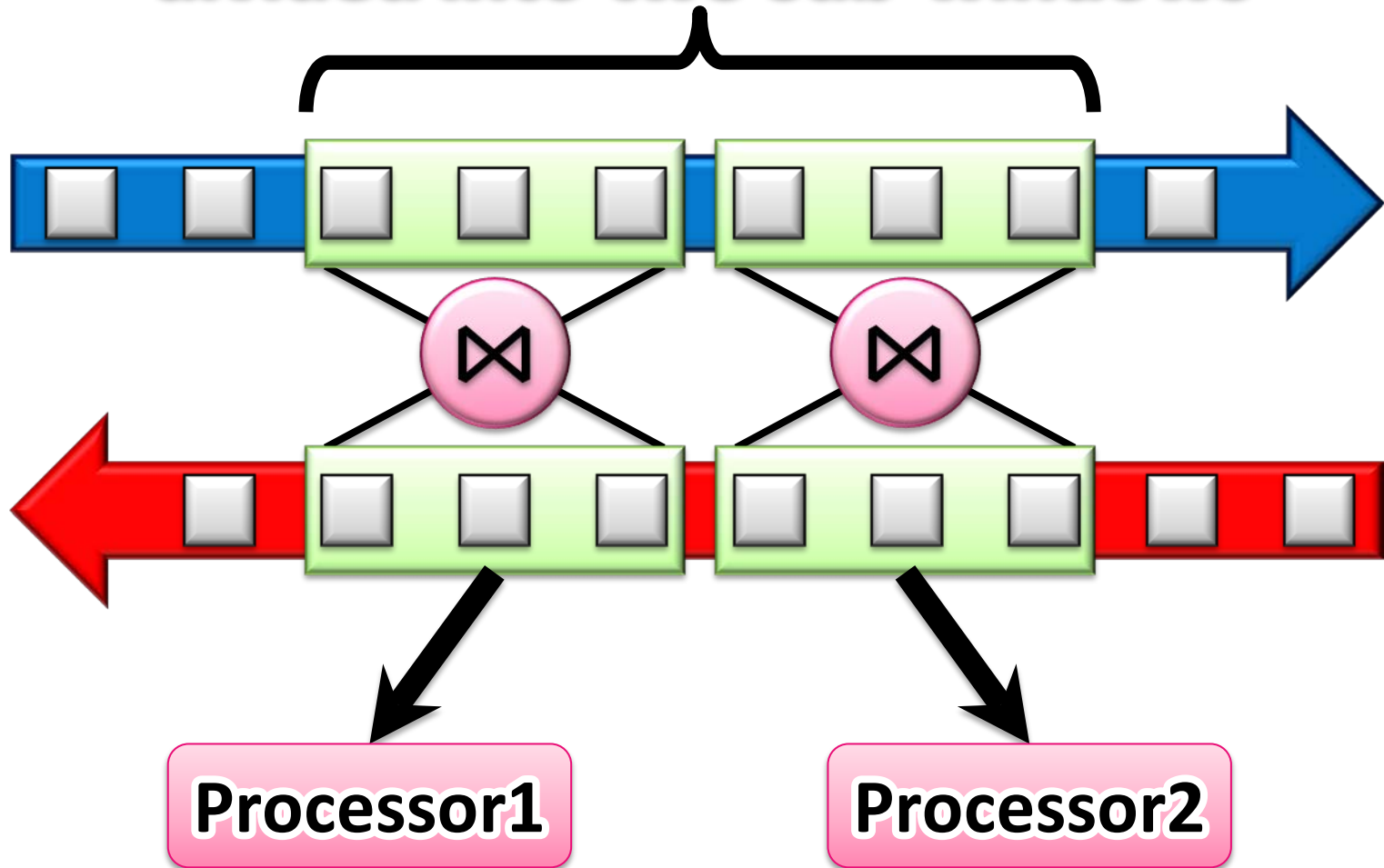
basic idea Flow in opposite direction

advantage Highly parallel evaluation



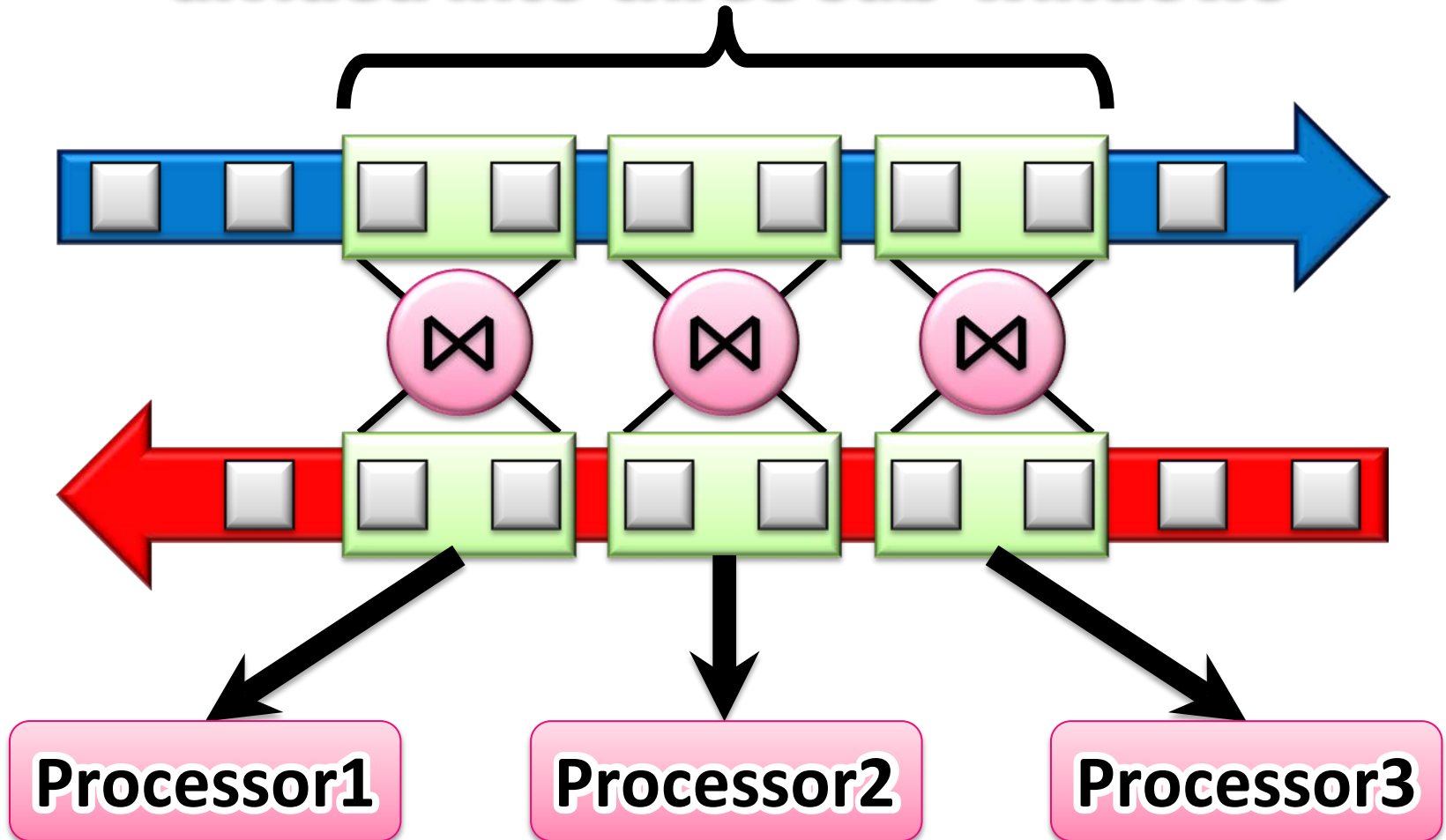
Handshake Join | Parallelization

divided into two sub-windows



Handshake Join | Parallelization

divided into three sub-windows

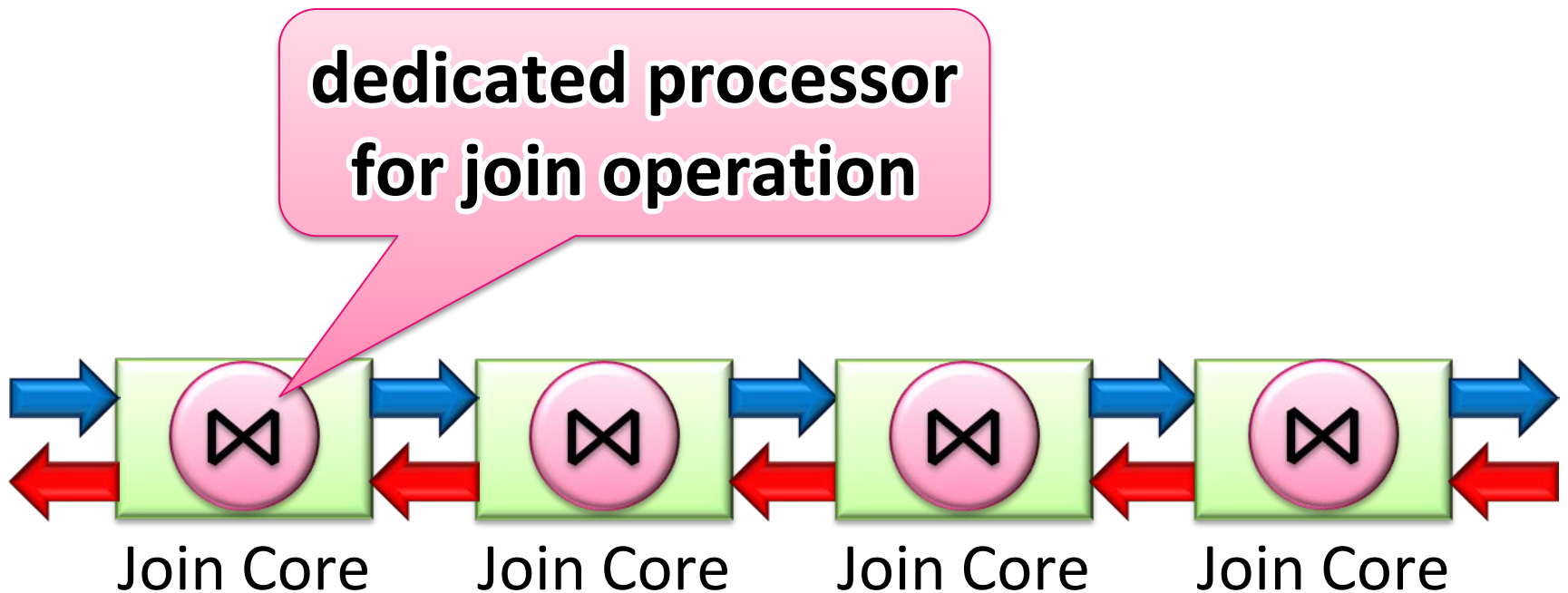




Naïve IMPLEMENTATION

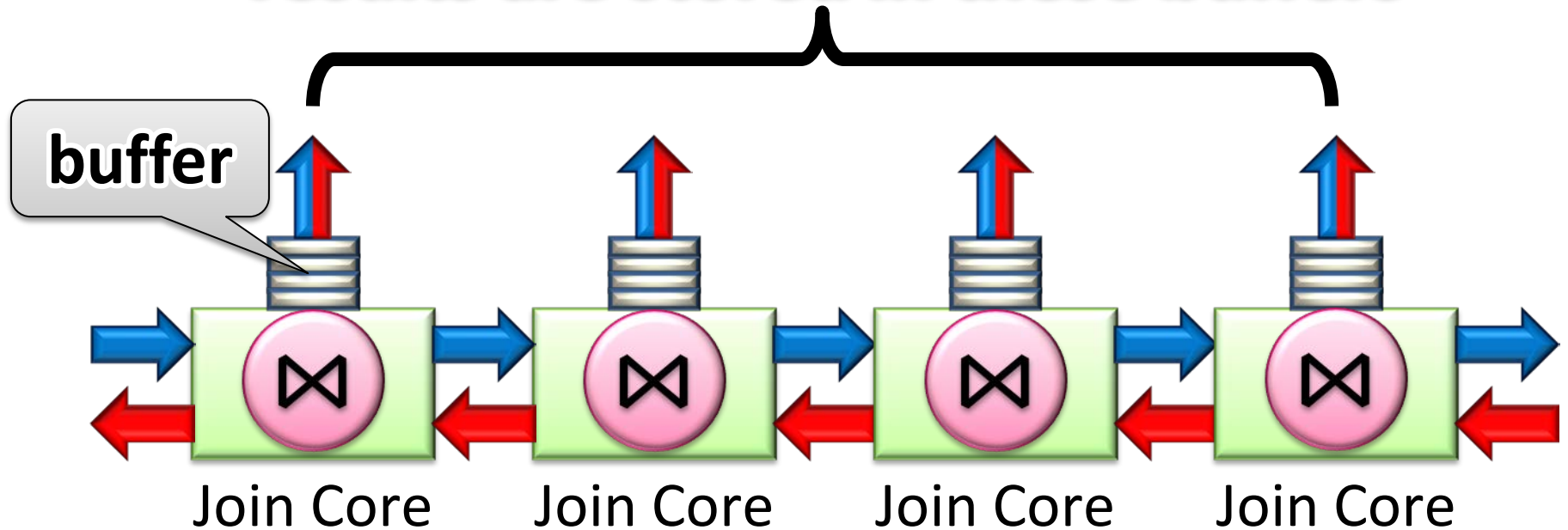
HANDSHAKE JOIN、J. Teubner and R. Müller, SIGMOD'11

Baseline Implementation

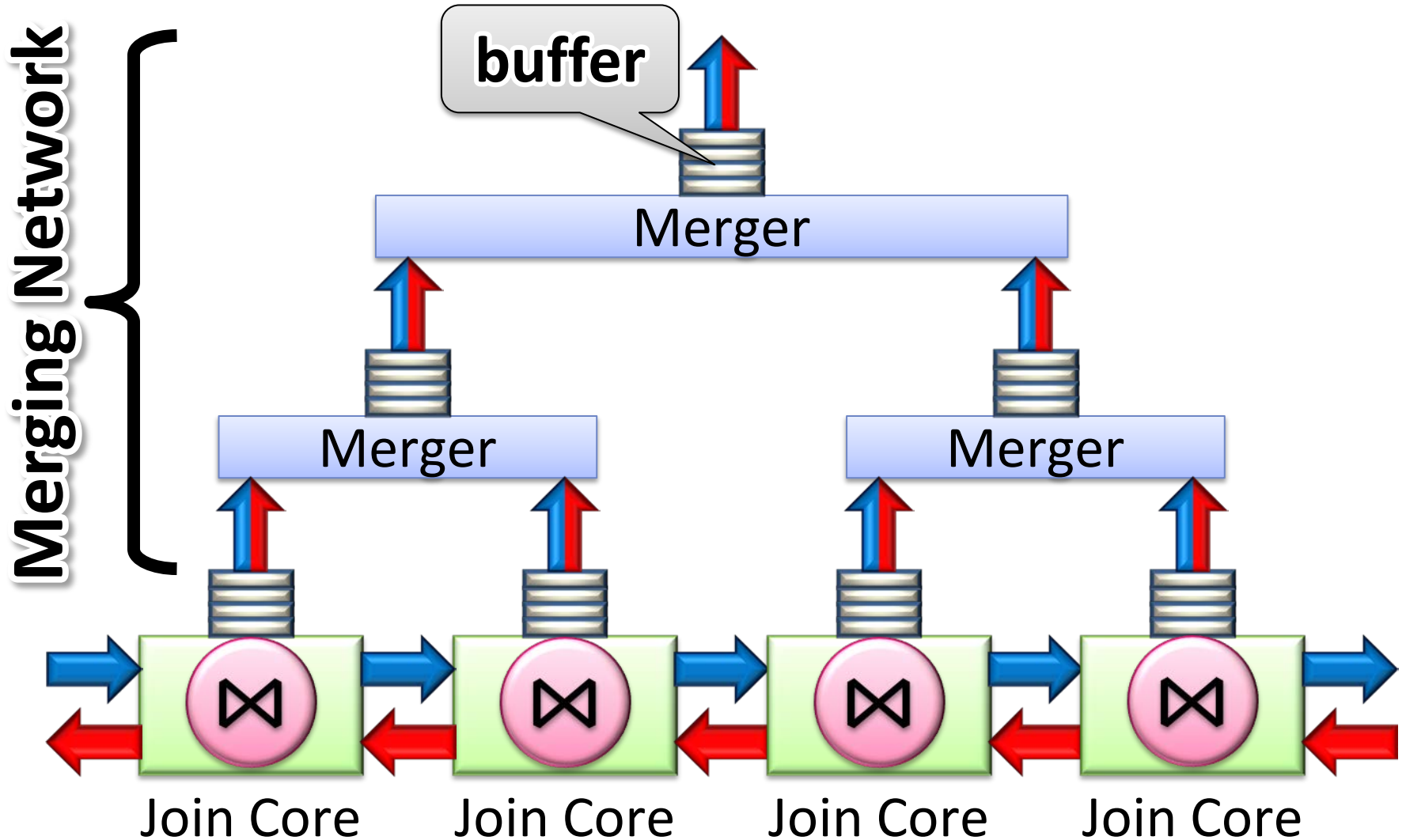


Baseline Implementation

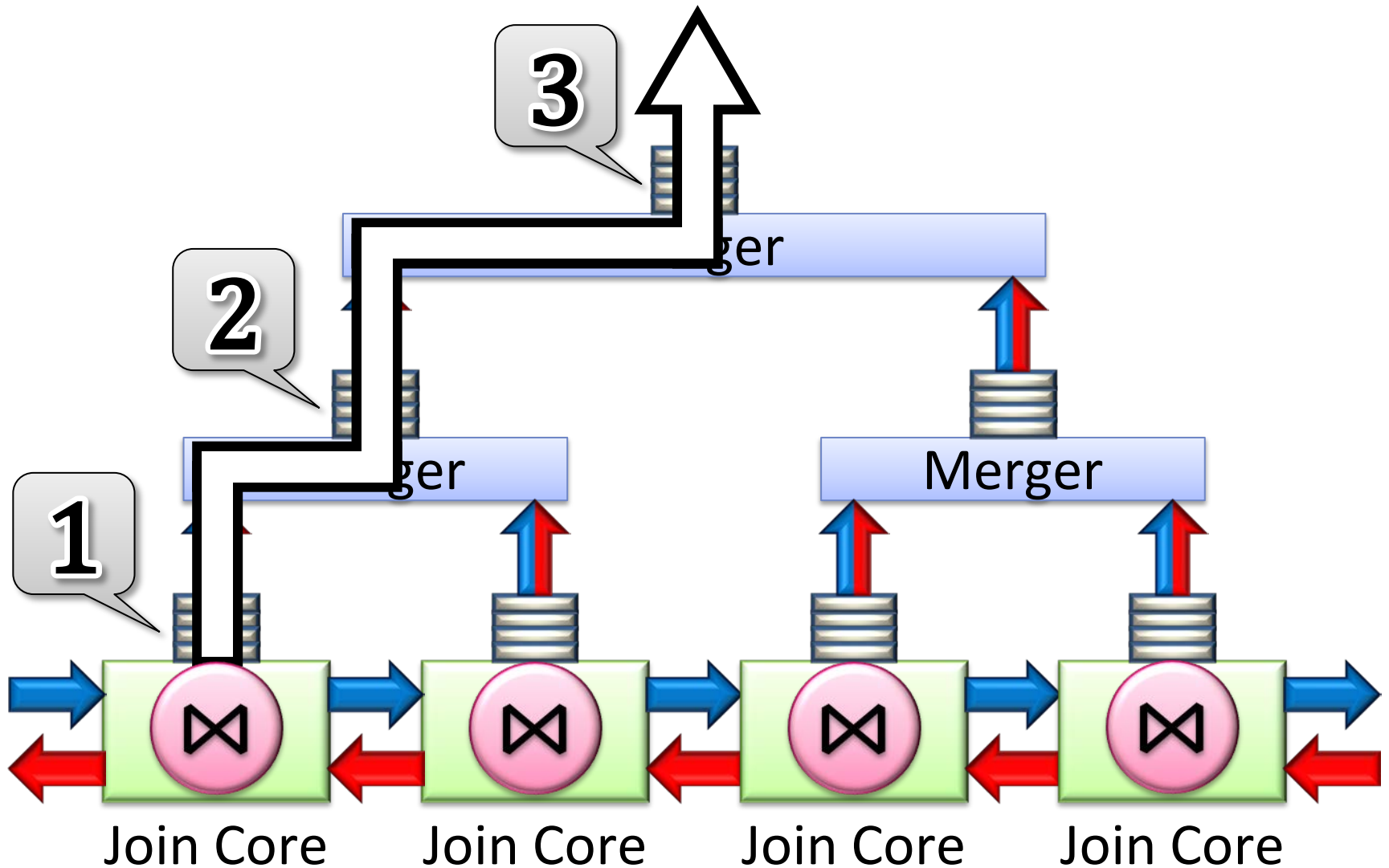
results are stored in these buffers



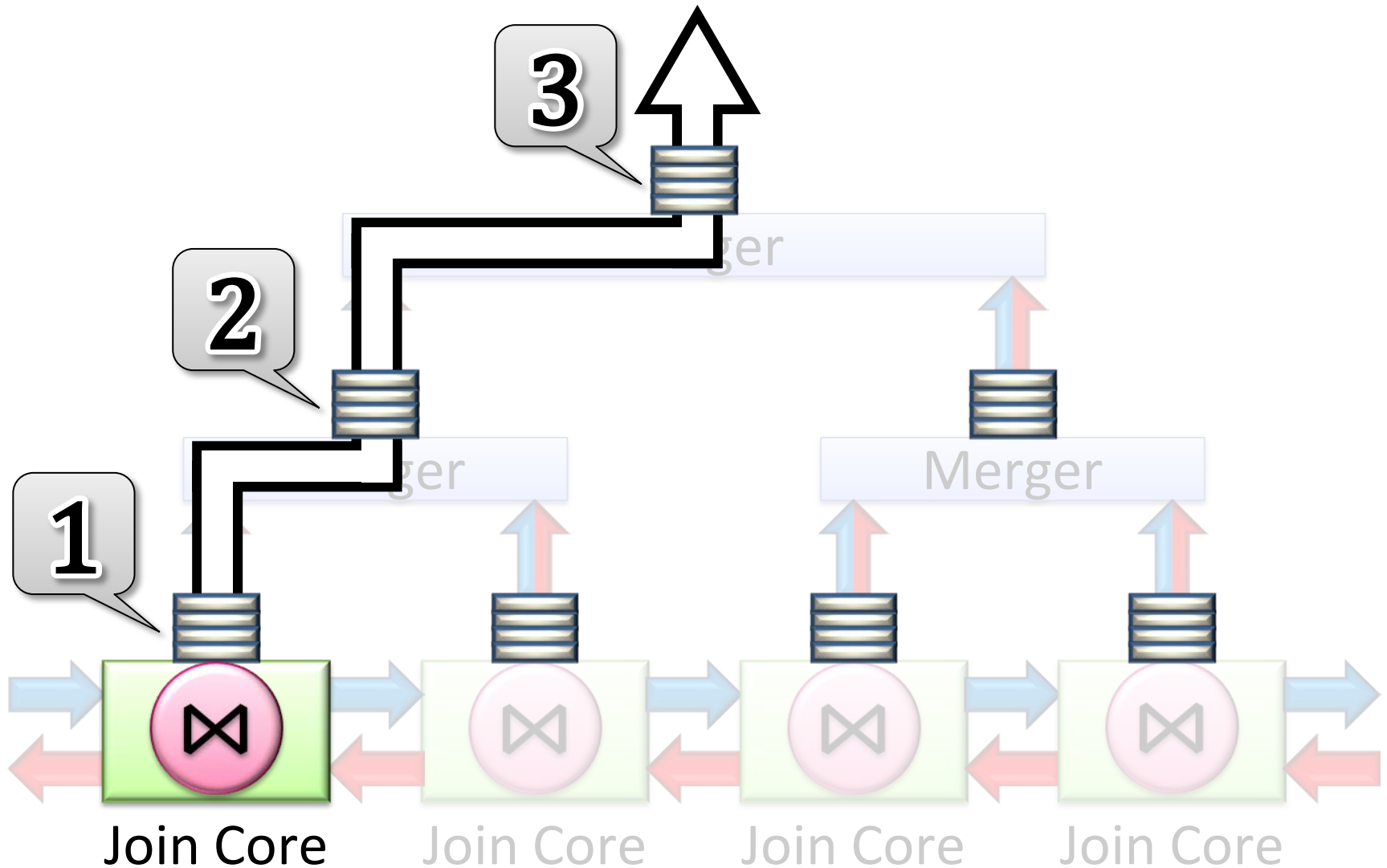
Baseline Implementation



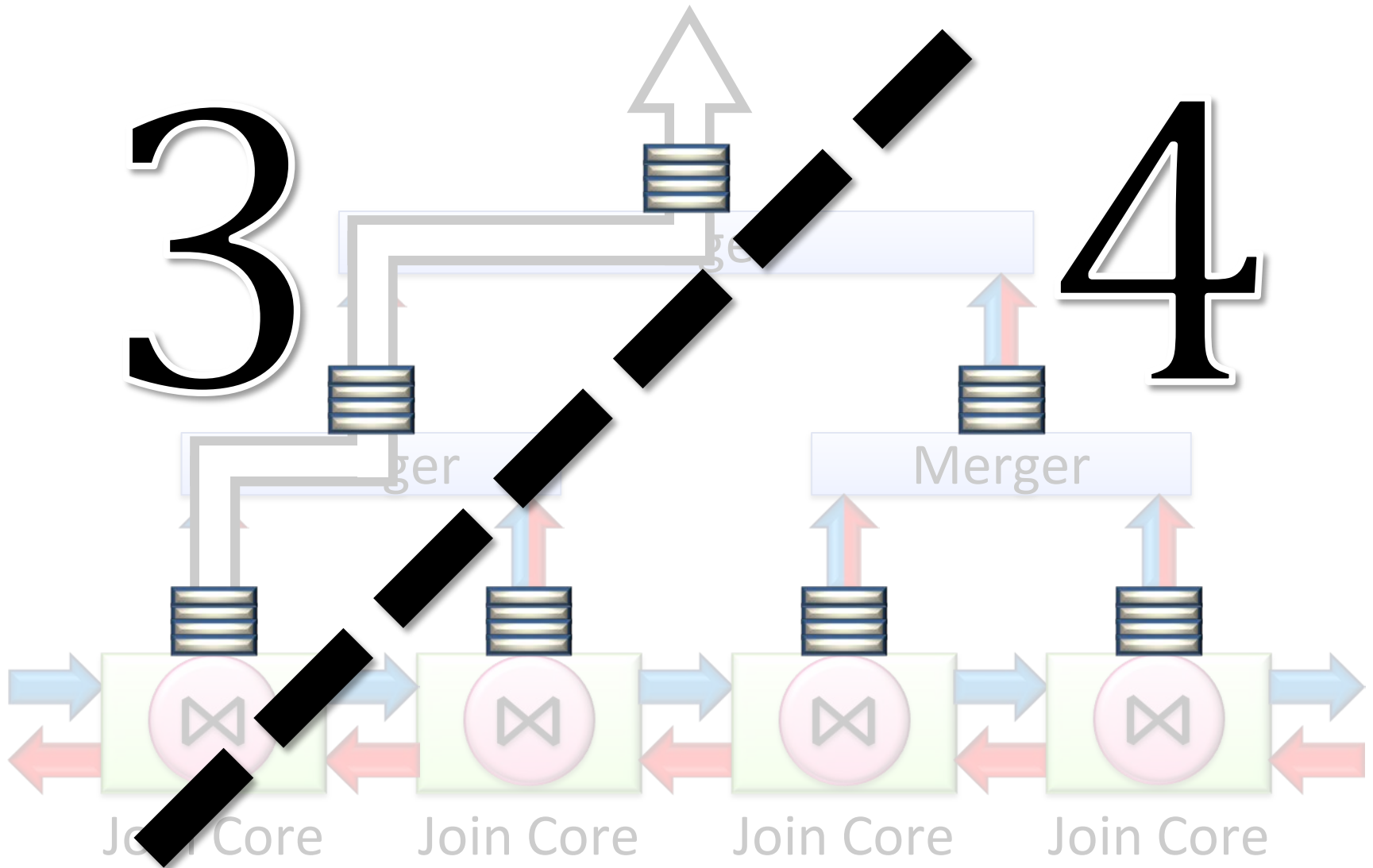
Output Data Flow



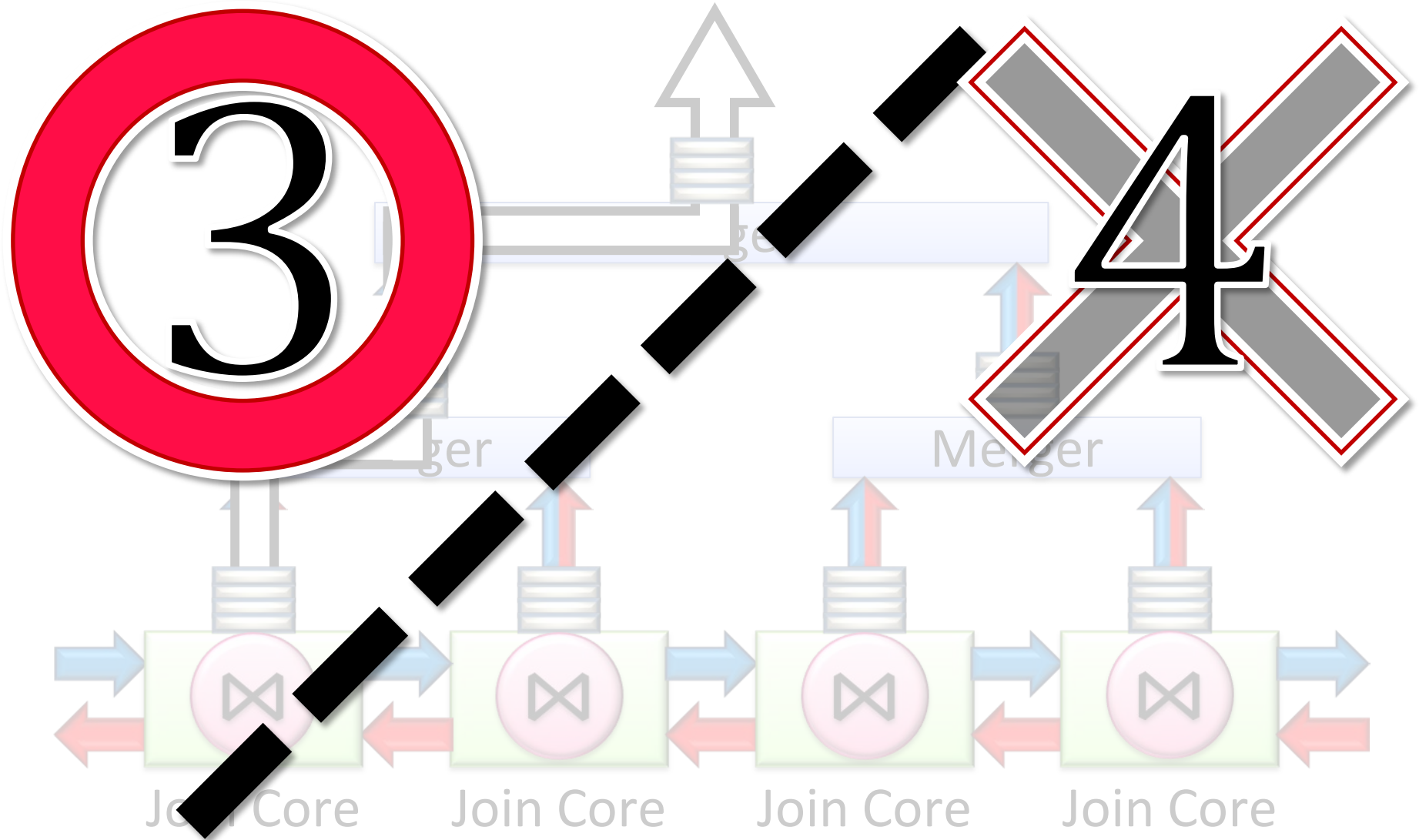
Issue | Inefficient Buffer Utilization



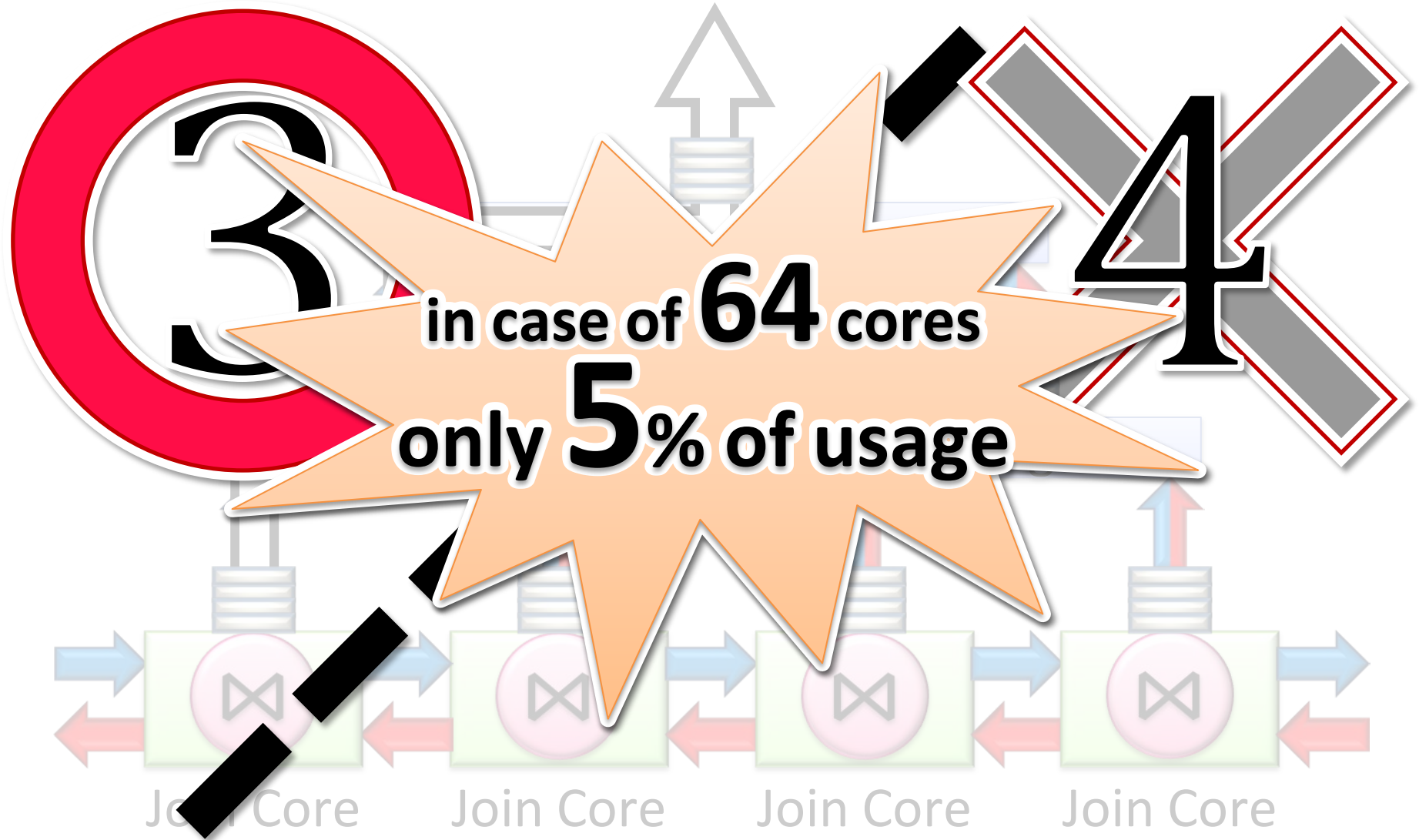
Issue | Inefficient Buffer Utilization



Issue | Inefficient Buffer Utilization



Issue | Inefficient Buffer Utilization

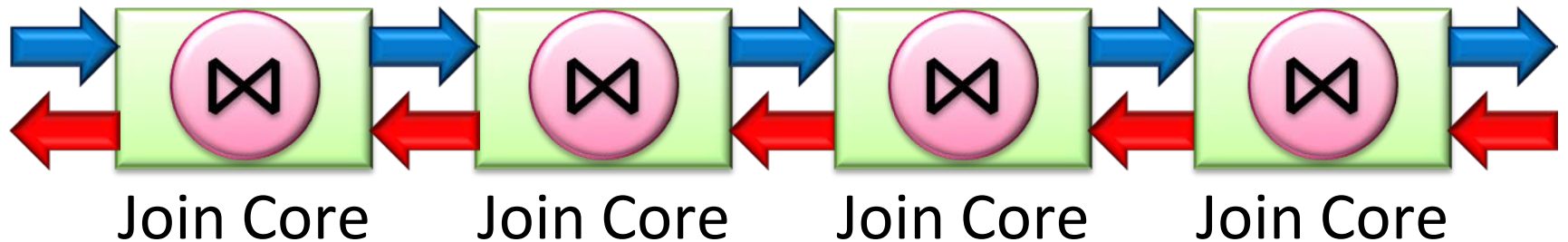




PROPOSED
IMPLEMENTATION

Adaptive Merging Network

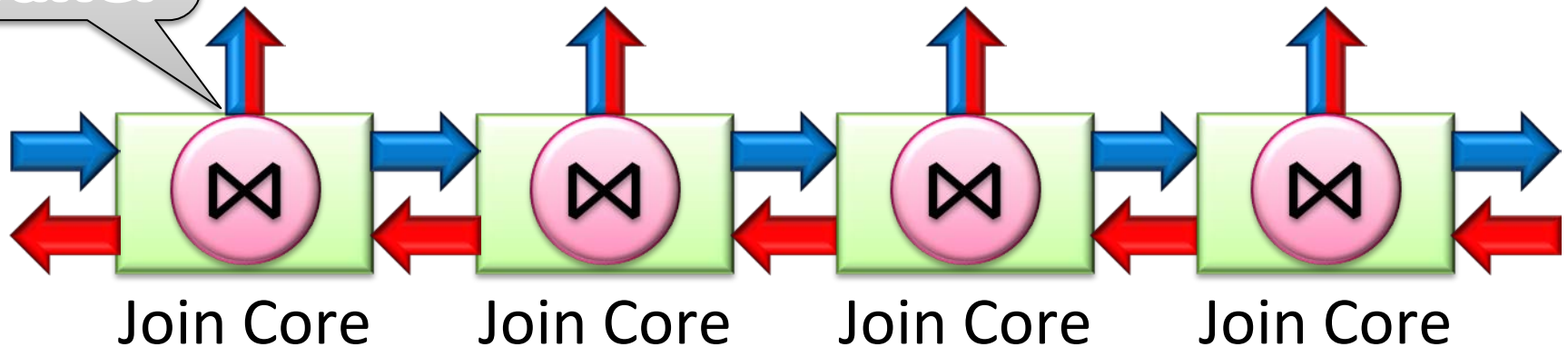
Proposed Implementation



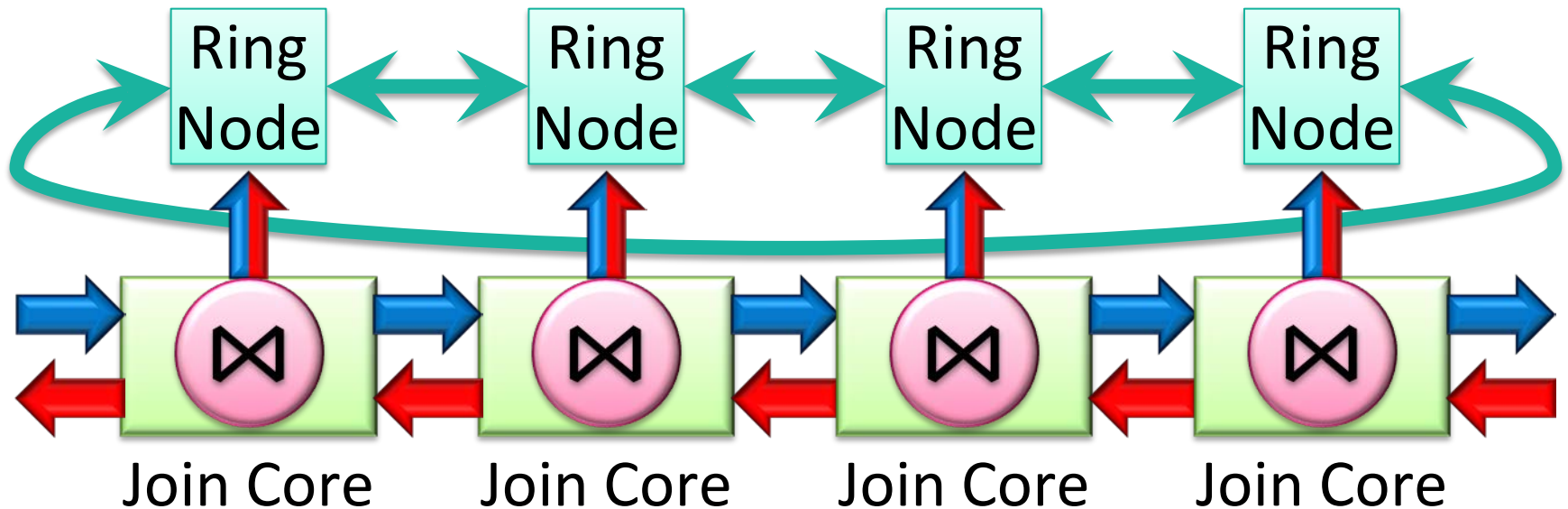
Proposed Implementation

Here, no buffering is required!

**NO
buffer**

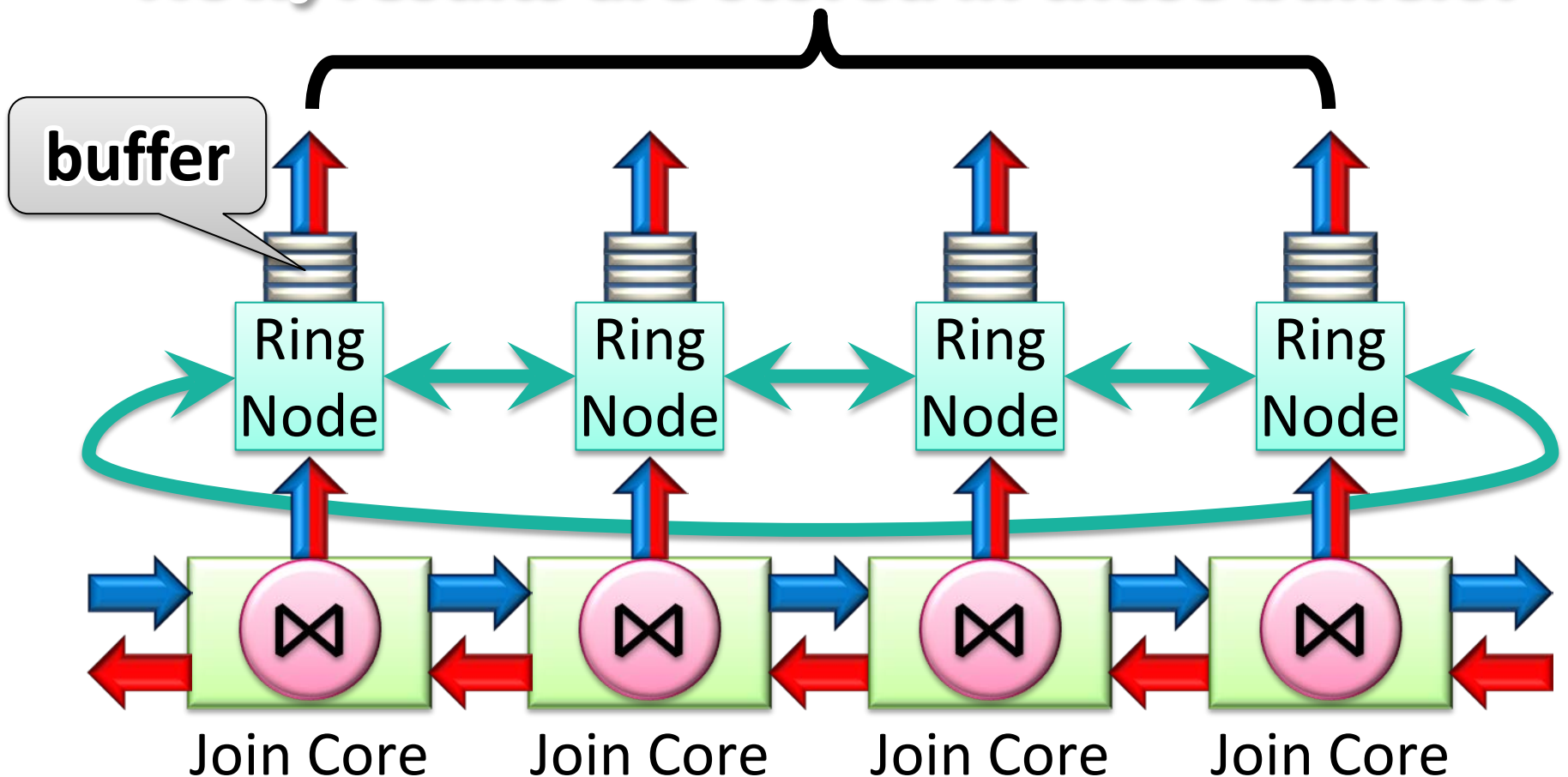


Proposed Implementation

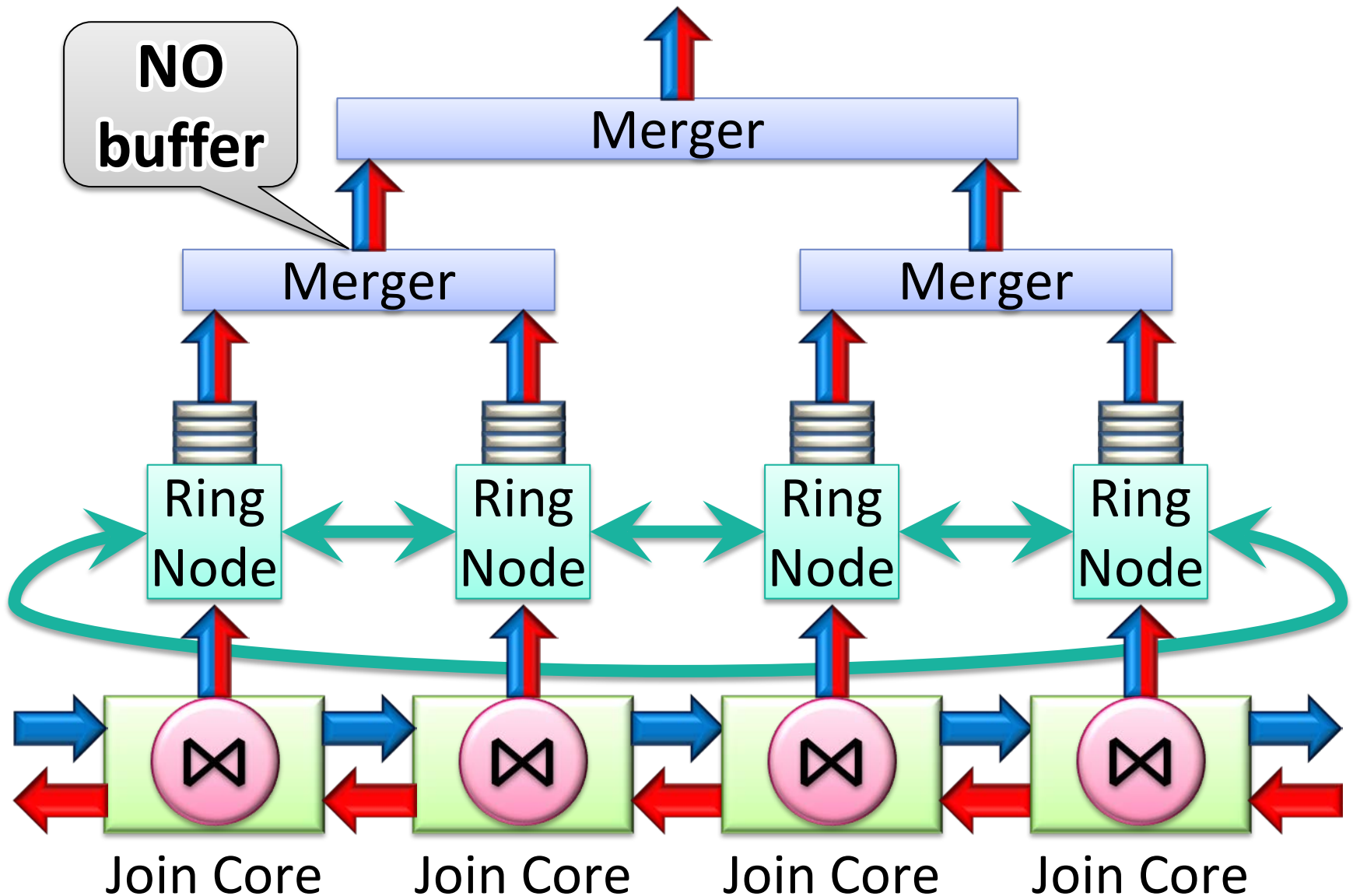


Proposed Implementation

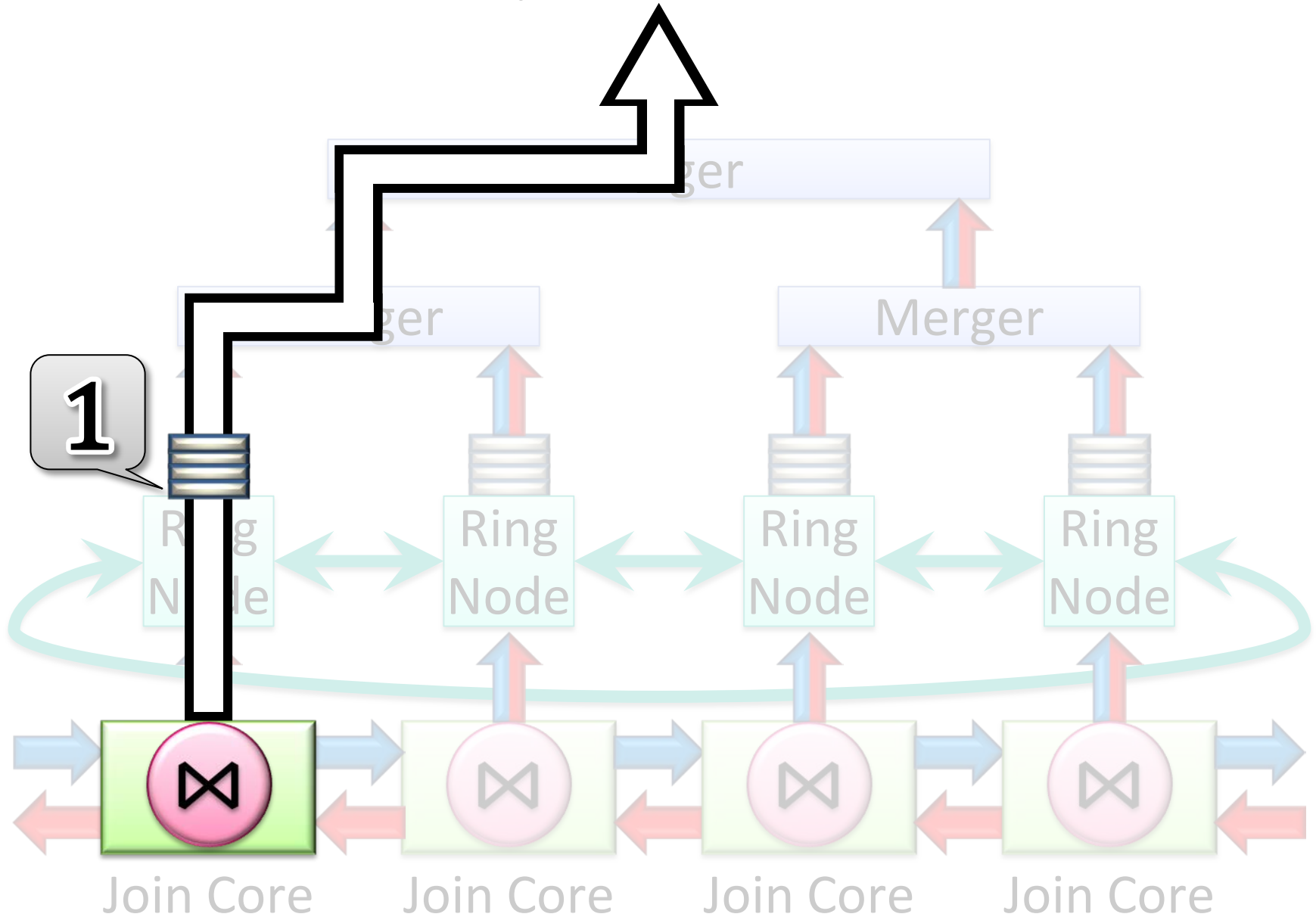
Now, results are stored in these buffers!

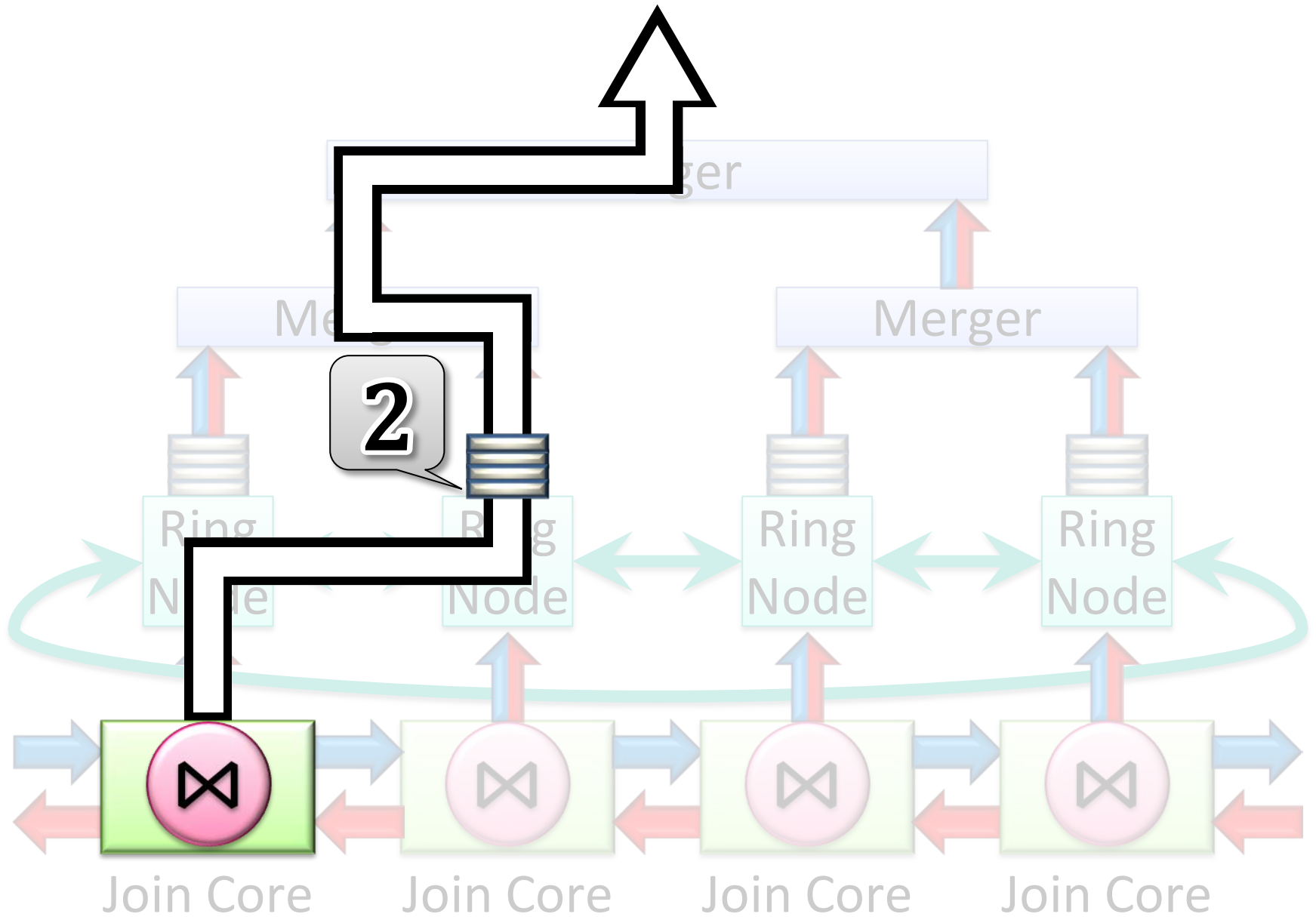


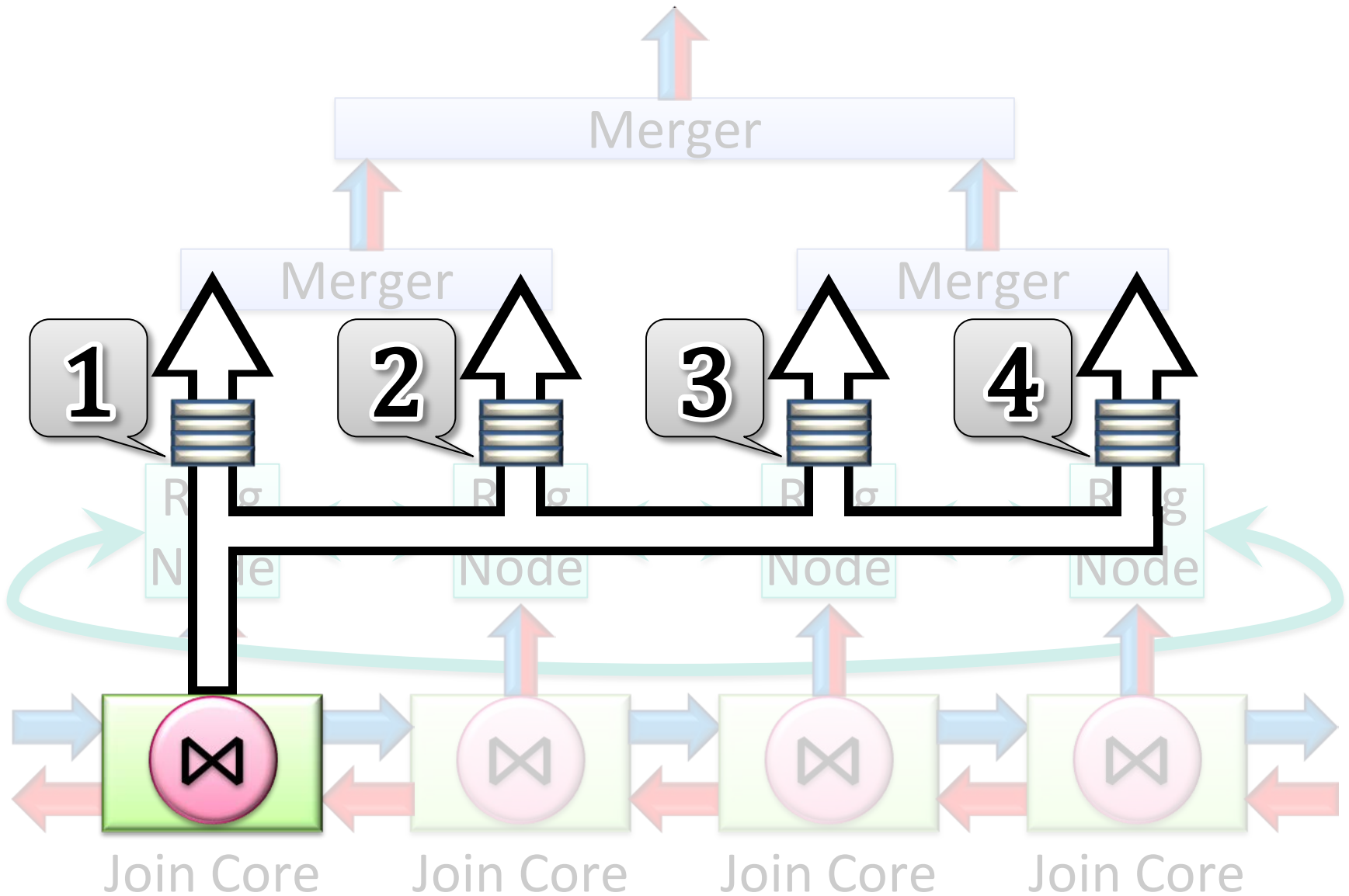
Proposed Implementation



Output Data Flow

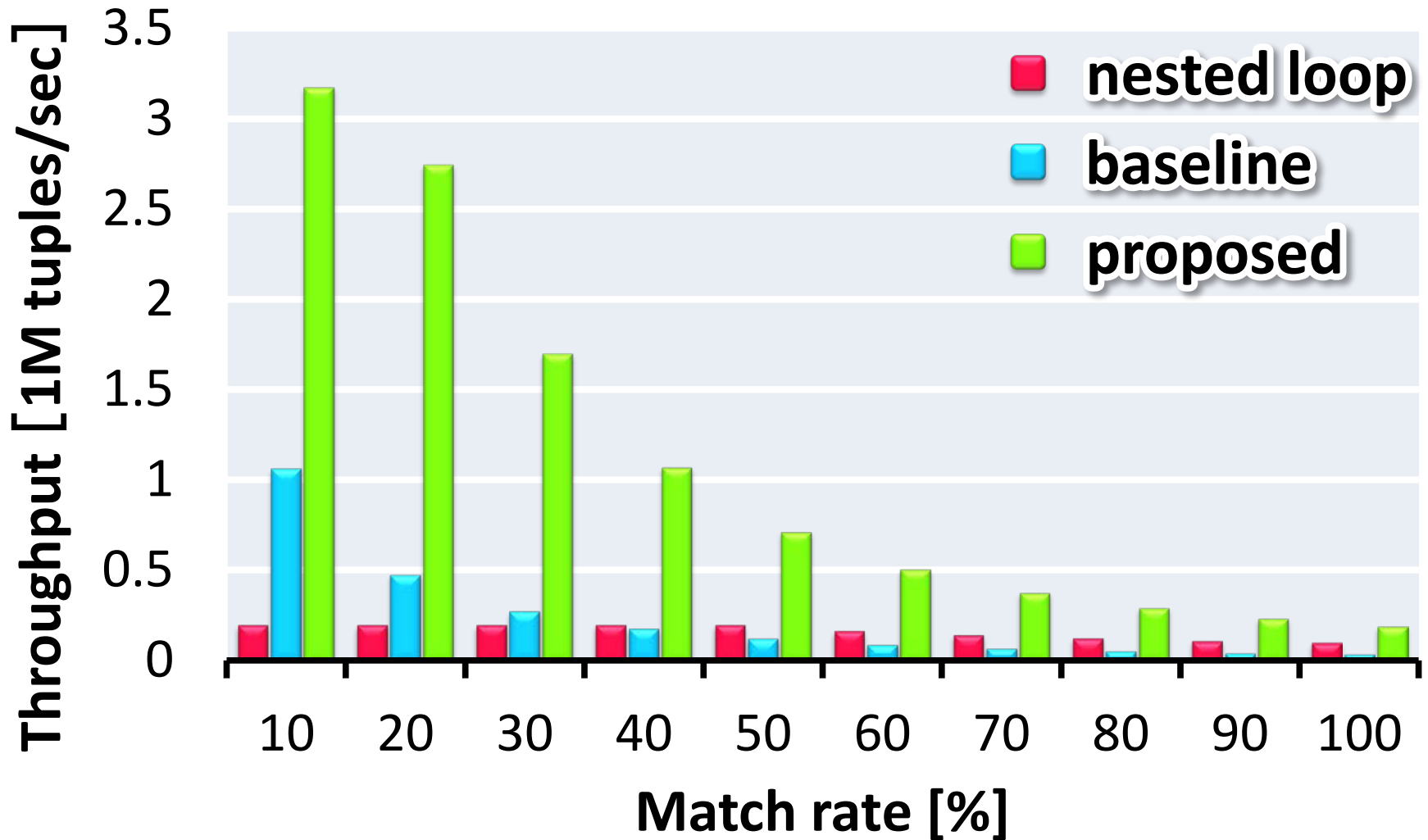








Performance | Proposed (64 join cores)



Basic: 6.7 millions of tuples per second

Proposal: 14.6 millions of tuples per second

- UDP-RX
- Window join (64-cores)

Performance
Monitor

DSMS

Demo @ SSDBM'13

Publications

- Yasin Oge, Takefumi Miyoshi, Hideyuki Kawashima, Tsutomu Yoshinaga: “A fast handshake join implementation on FPGA with adaptive merging network”. **SSDBM** 2013
- Yasin OGE, Takefumi MIYOSHI, Hideyuki KAWASHIMA, Tsutomu YOSHINAGA: "Design and Implementation of a Handshake Join Architecture on FPGA", **IEICE TRANSACTIONS(Journal)** Vol.E95-D No.12 pp.2919-2927
- Yasin Oge, Takefumi MIYOSHI, Hideyuki Kawashima, Tsutomu Yoshinaga, "Design and Implementation of a Merging Network Architecture for Handshake Join Operator on FPGA", **IEEE MCSOC**, pp.84-91, Sep. 2012.

Cited by ISCA'13 Paper
- Yasin Oge, Takefumi Miyoshi, Hideyuki Kawashima, and Tsutomu Yoshinaga, "An Implementation of Handshake Join on FPGA", **ICNC**, pp.95-104, Dec. 2011. 
- Takefumi Miyoshi, Hideyuki Kawashima, Yuta Terada and Tsutomu Yoshinaga, “A Coarse Grain Reconfigurable Processor Architecture for Stream Processing Engine”, **FPL**, Sep. 5-7, 2011

← Top tier conf. on FPGA