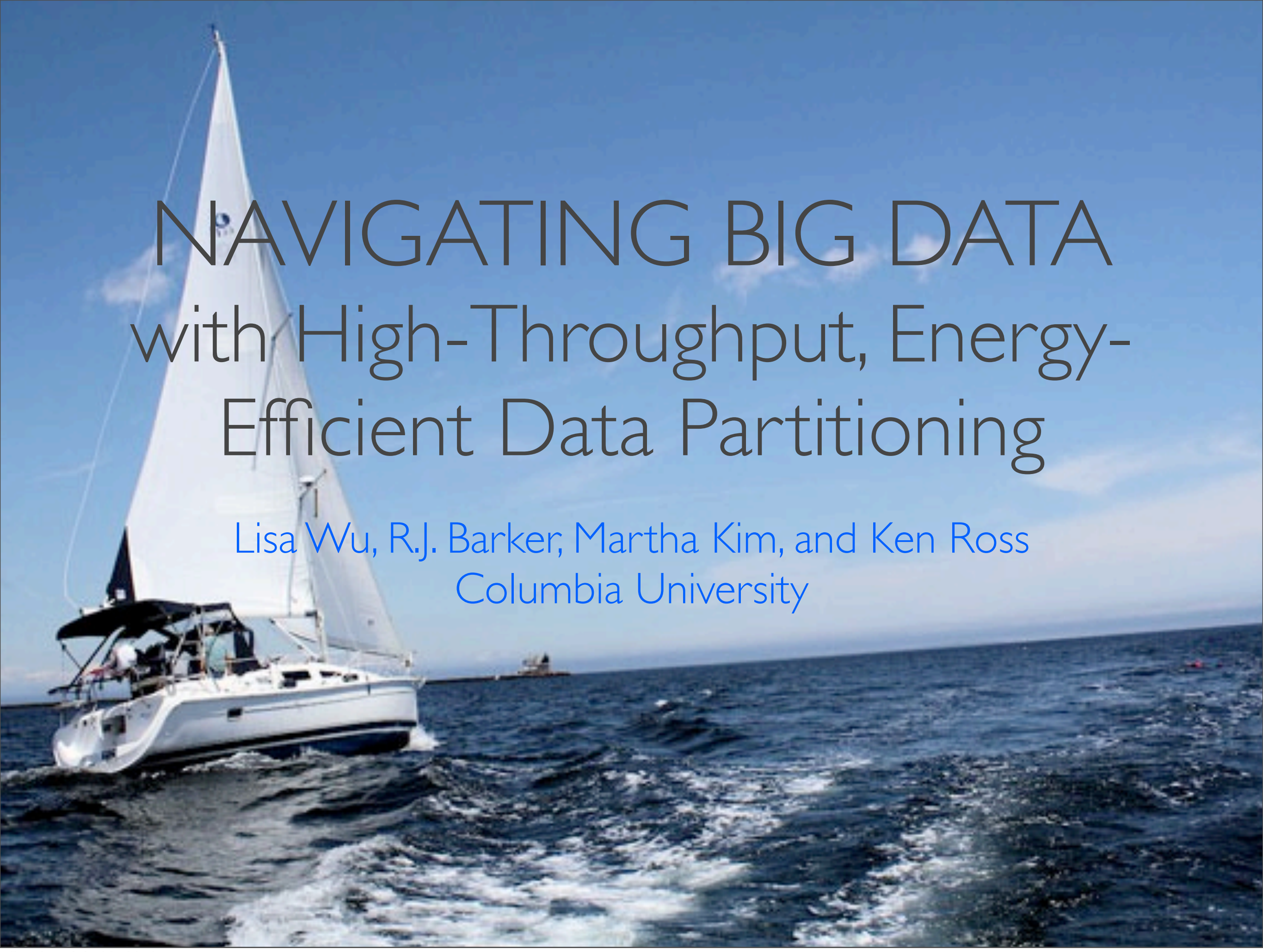


# NAVIGATING BIG DATA with High-Throughput, Energy-Efficient Data Partitioning

Lisa Wu, R.J. Barker, Martha Kim, and Ken Ross
Columbia University

# BIG DATA is here

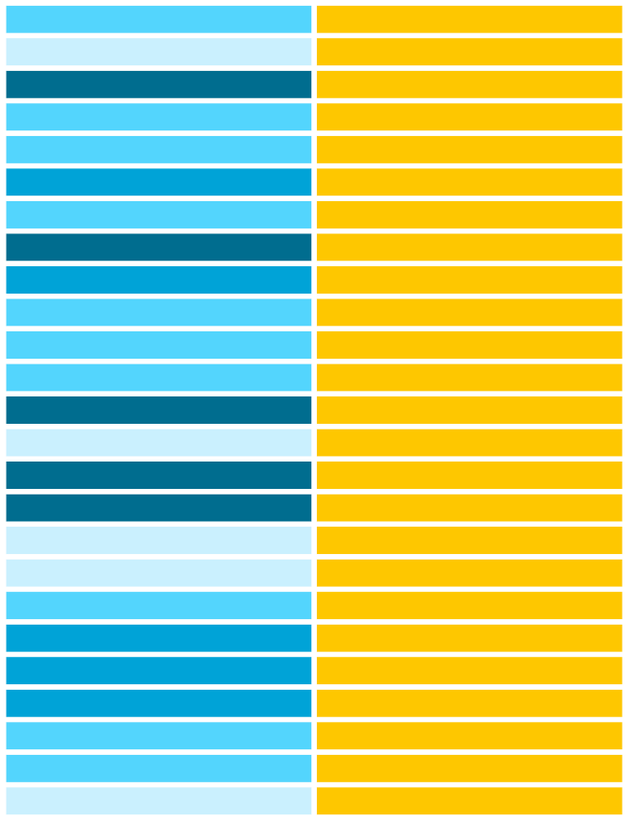
Sources:
IDC Worldwide Big Data Technology and Services 2012-2015 Forecast, #233485, March 2012
The Next Web, DAZEINFO
NetApp

# Do sunblock sales correlate with weather?

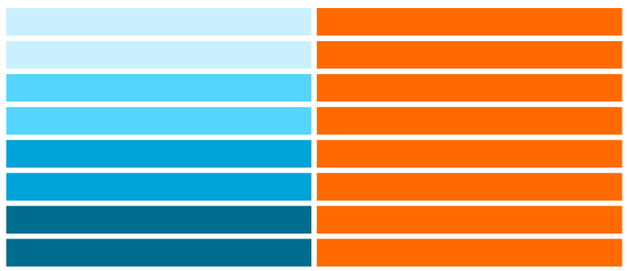
# JOINs = Cross Reference

# JOINs = Cross Reference

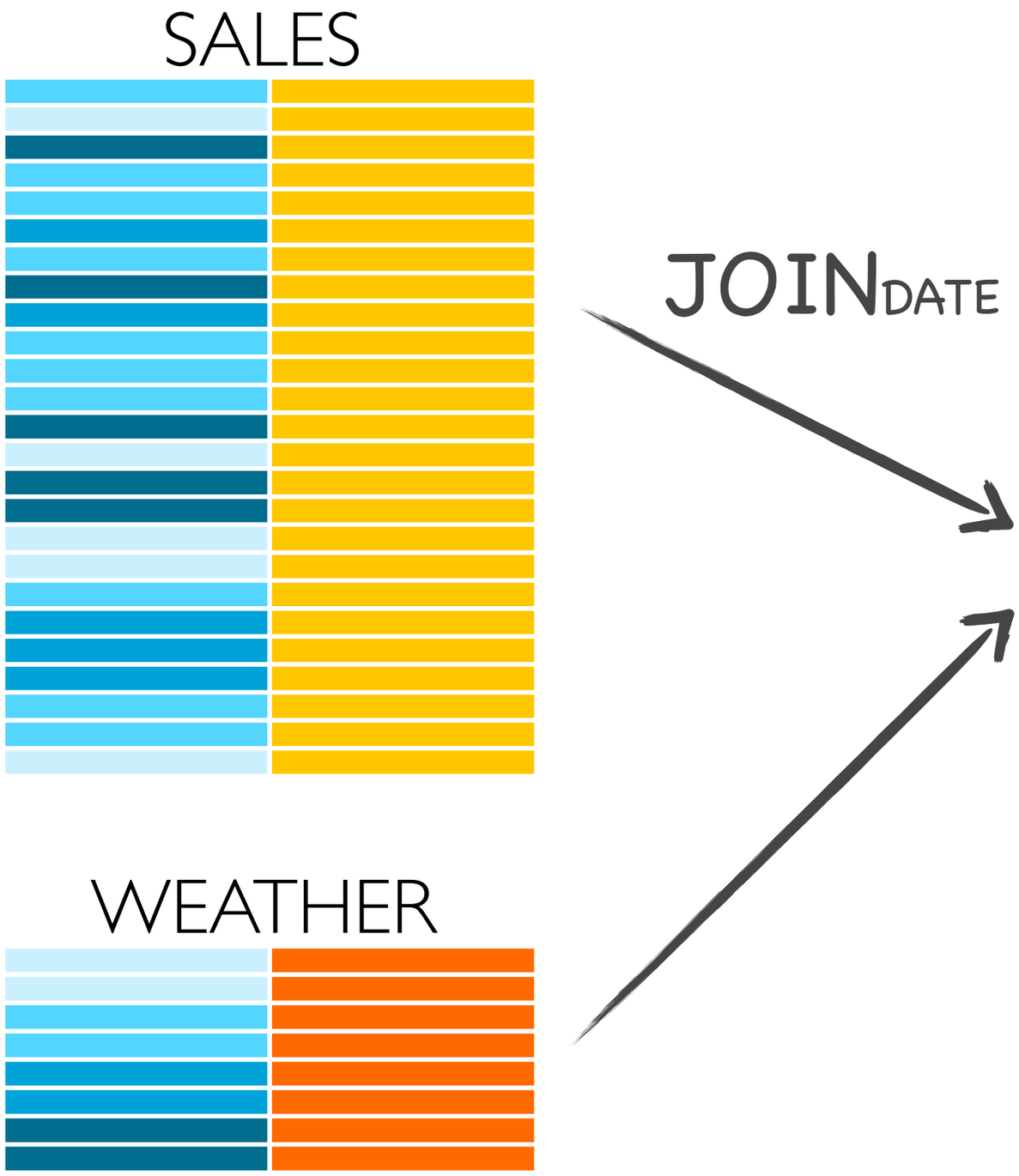
SALES



WEATHER

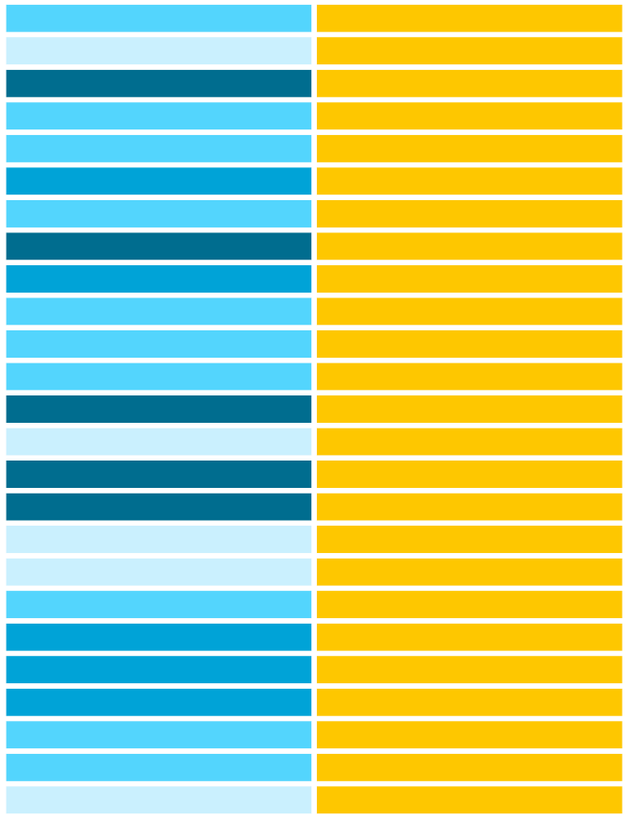


# JOINs = Cross Reference



# JOINs = Cross Reference

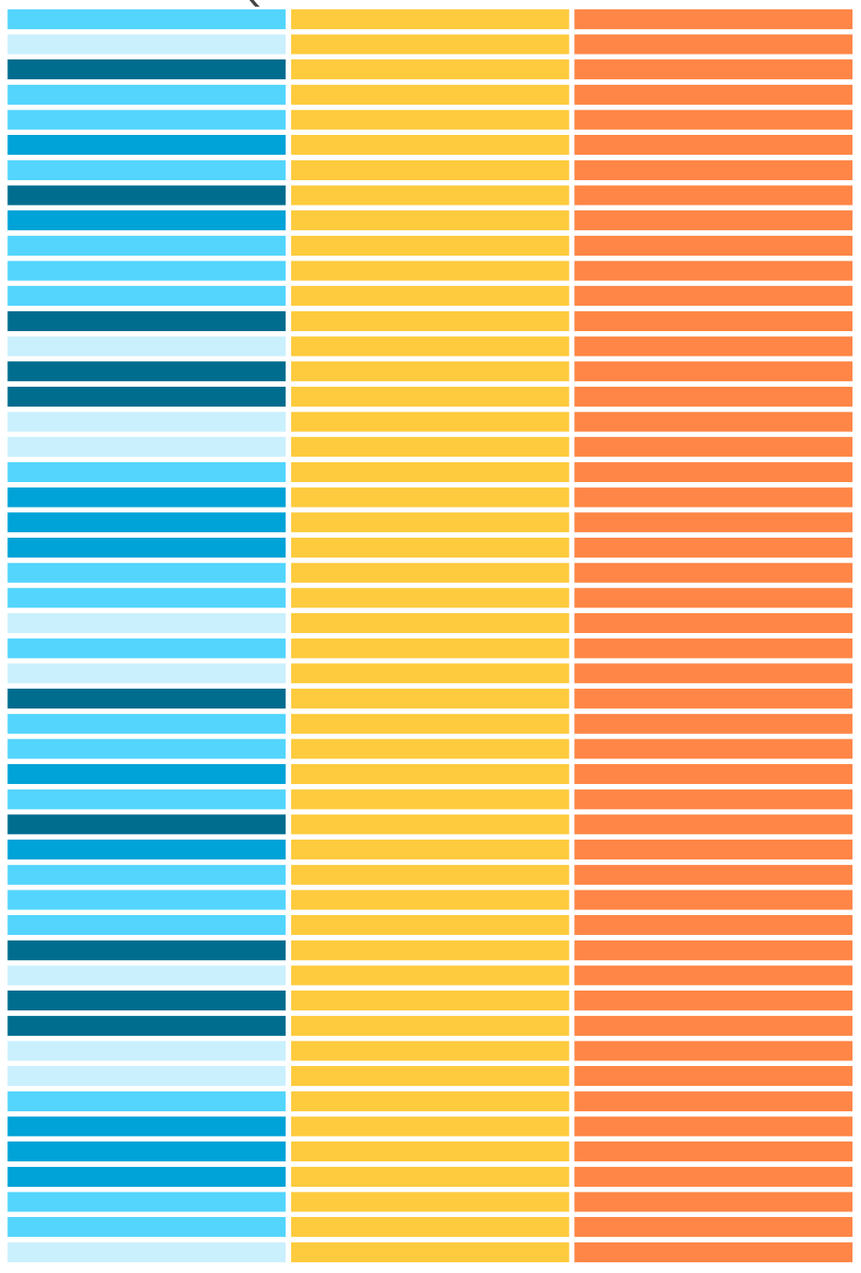
SALES



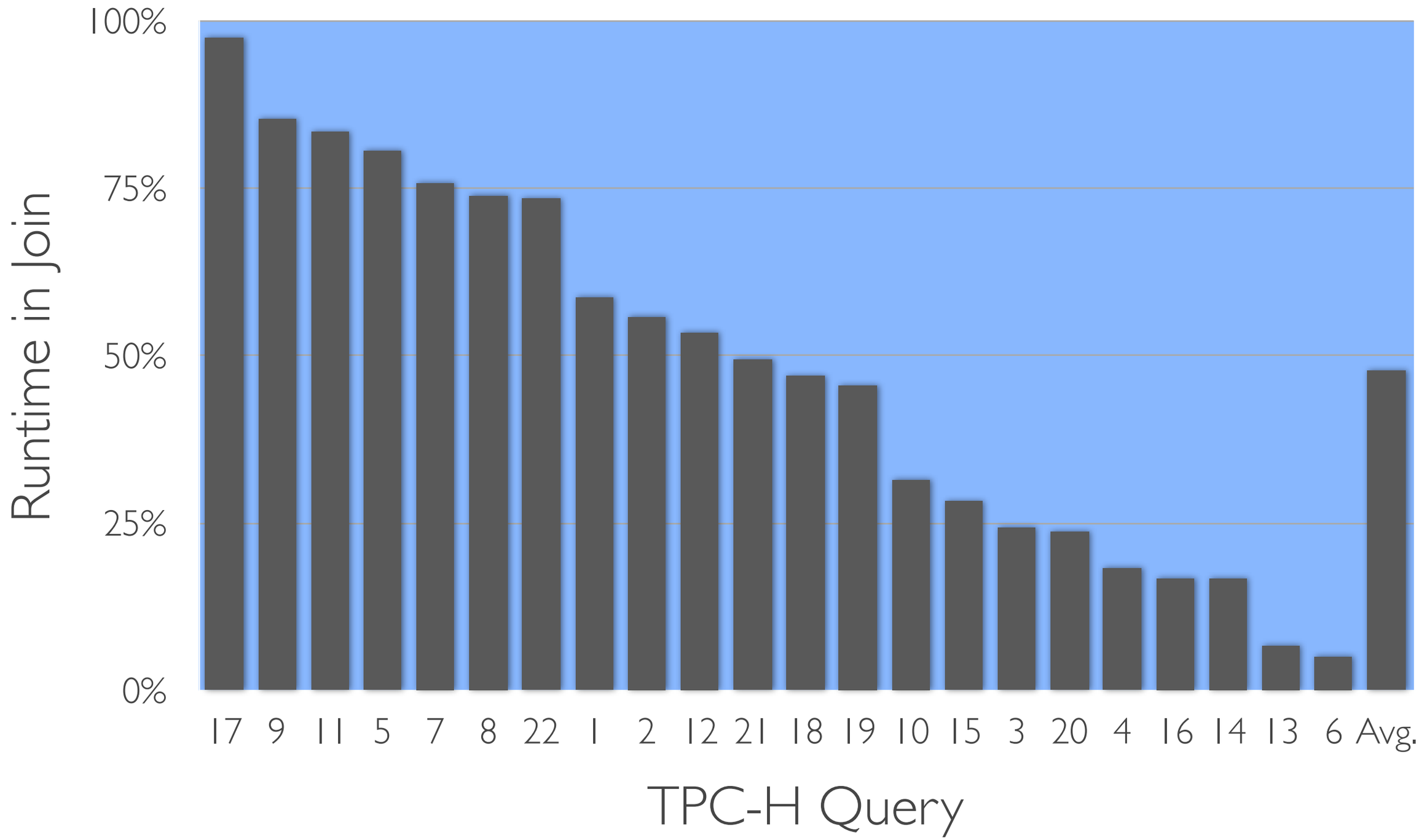
WEATHER

JOIN$_{DATE}$

JOIN$_{DATE}$(SALES, WEATHER)

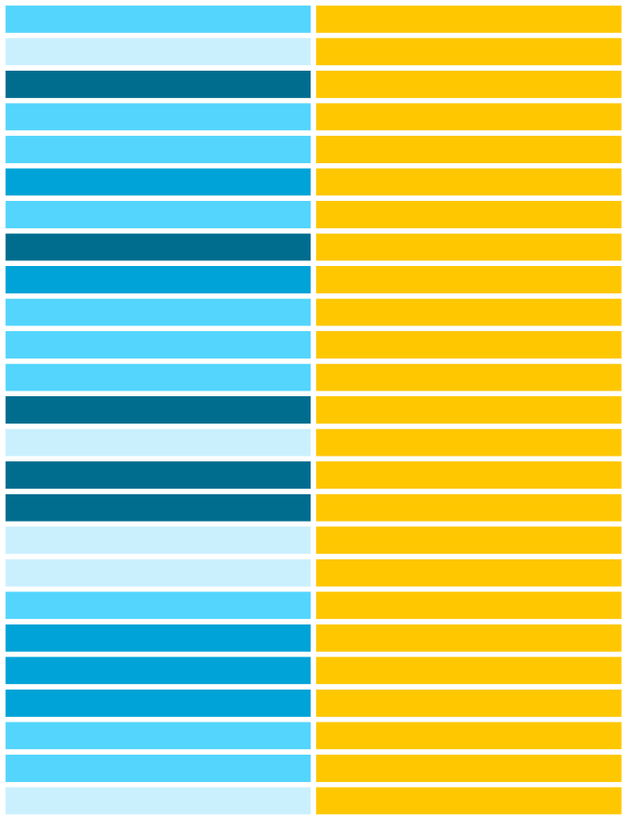


# JOINs = 47% TPC-H Execution Time

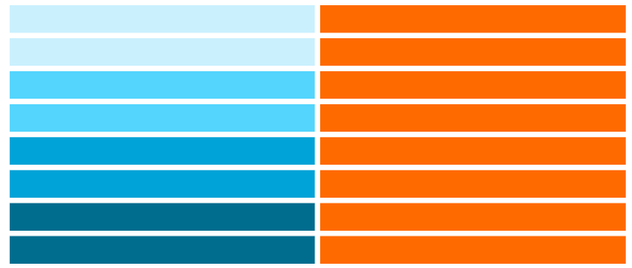


# Naïve JOINs of BIG DATA Thrash the Cache

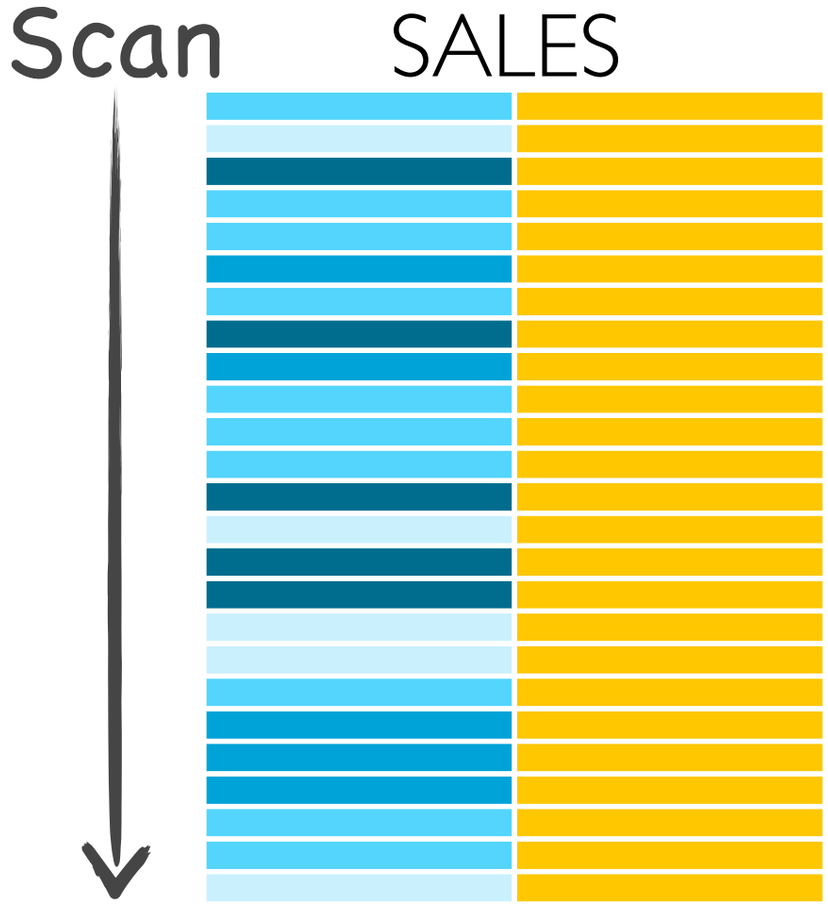
SALES

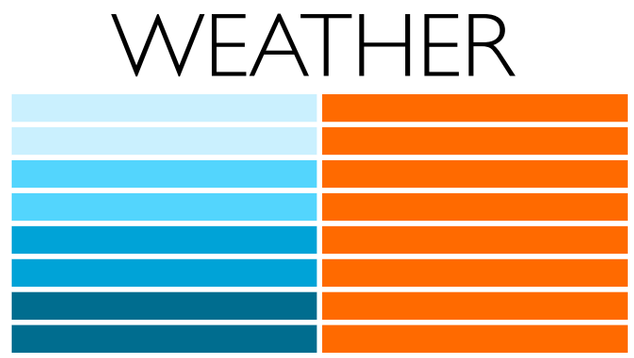


WEATHER

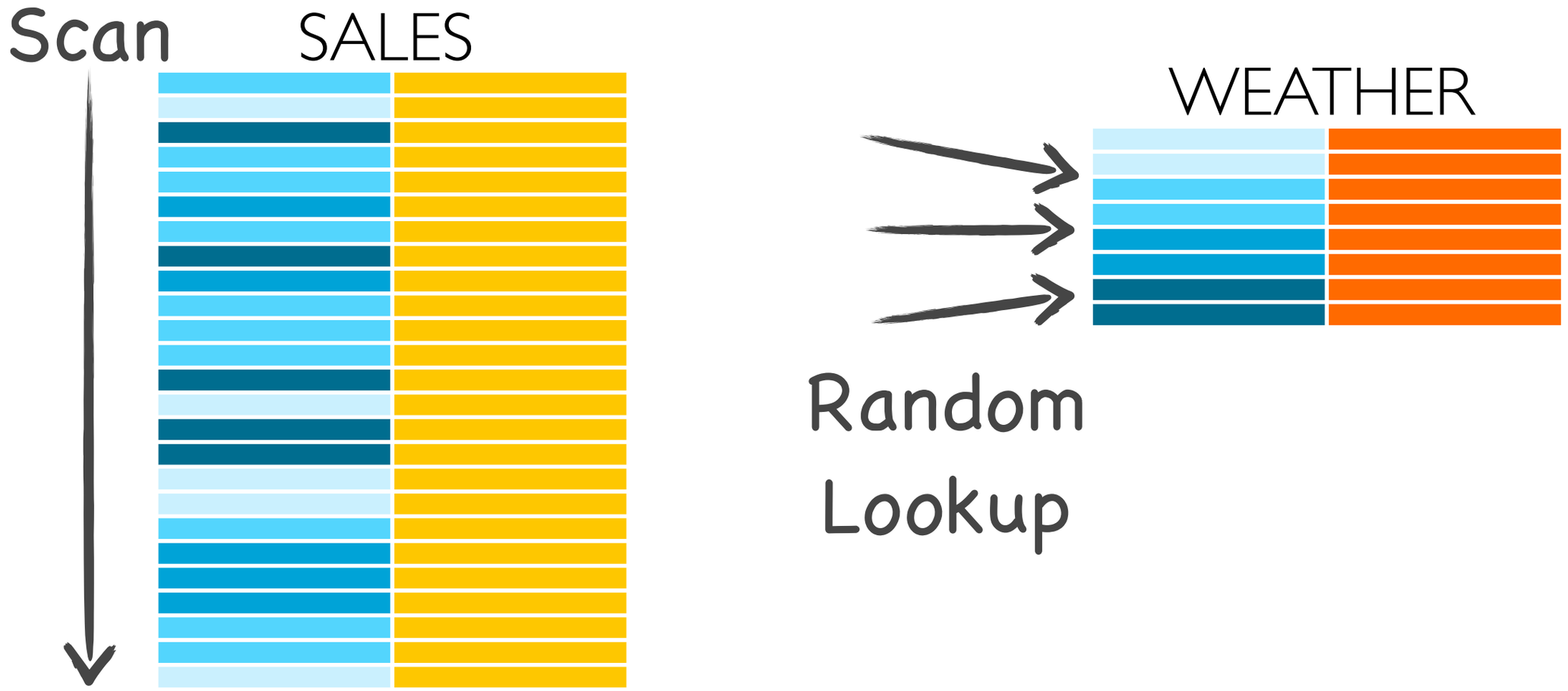


# Naïve JOINs of BIG DATA Thrash the Cache

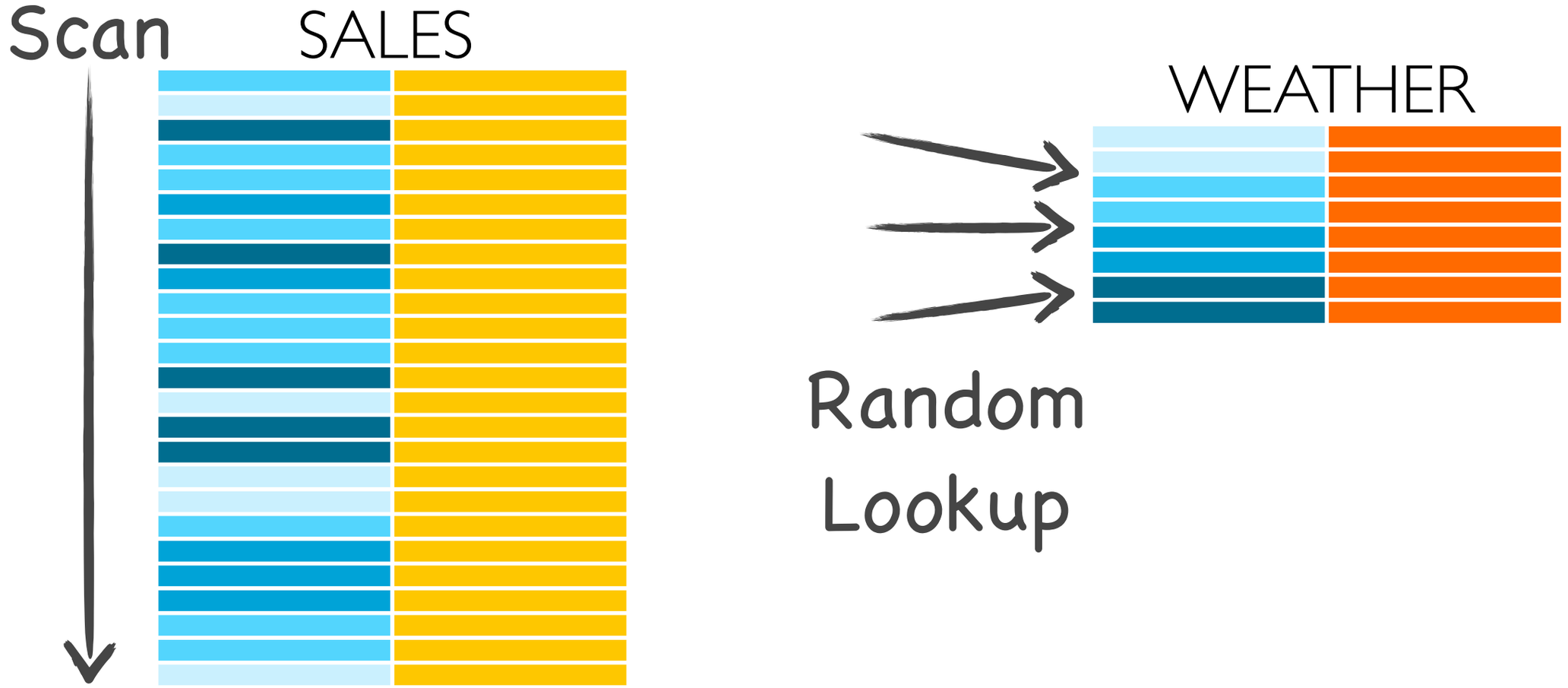


# Naïve JOINs of BIG DATA Thrash the Cache



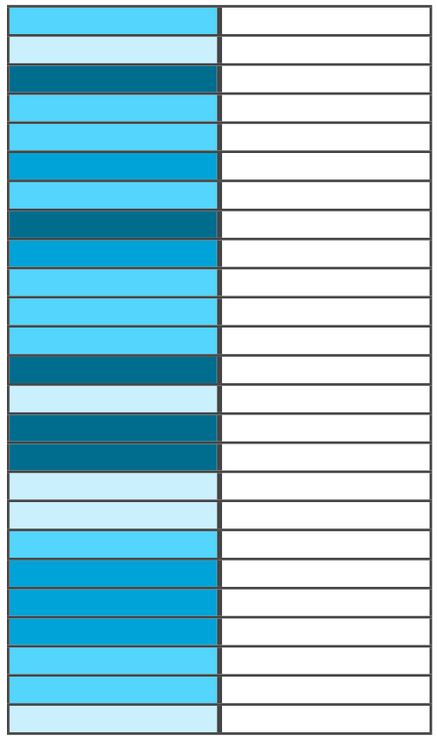
# Naïve JOINs of BIG DATA Thrash the Cache



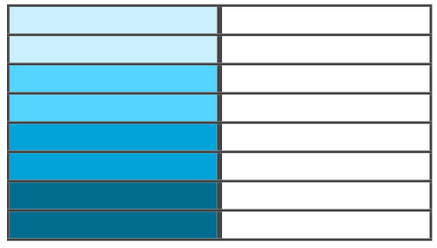

Small table doesn't fit into cache → lookups thrash

# Partitioned JOIN

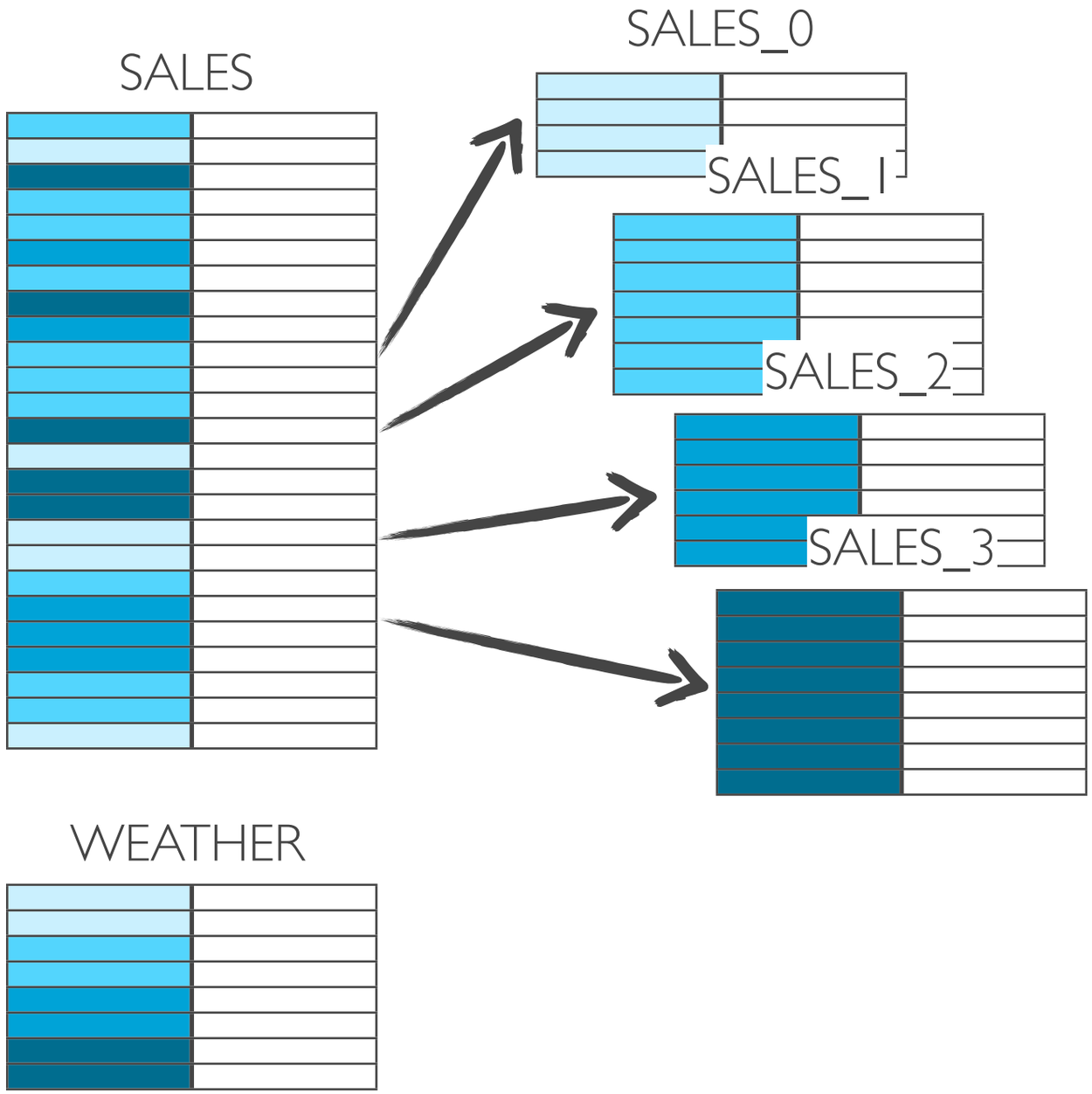
SALES



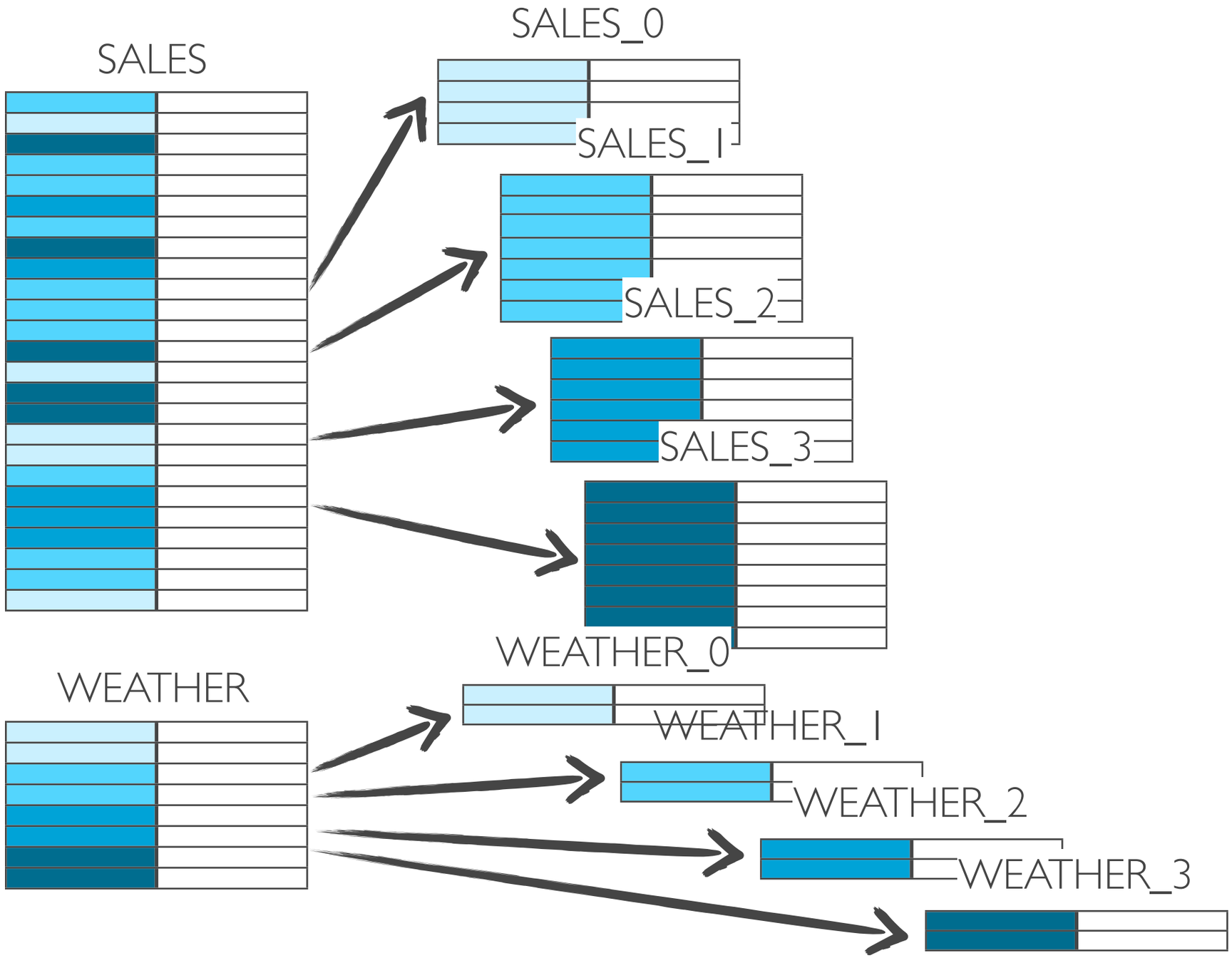
WEATHER



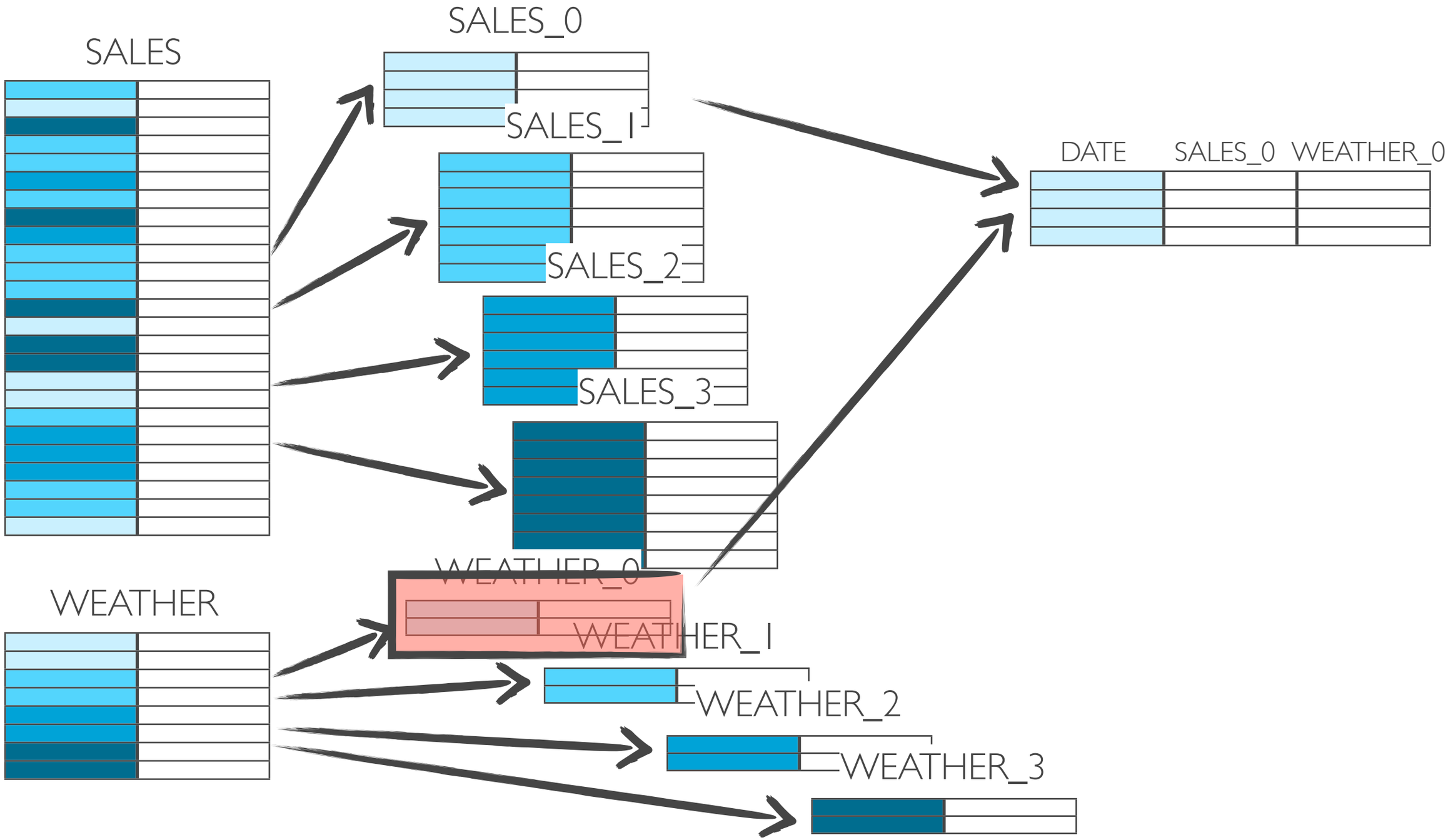
# Partitioned JOIN



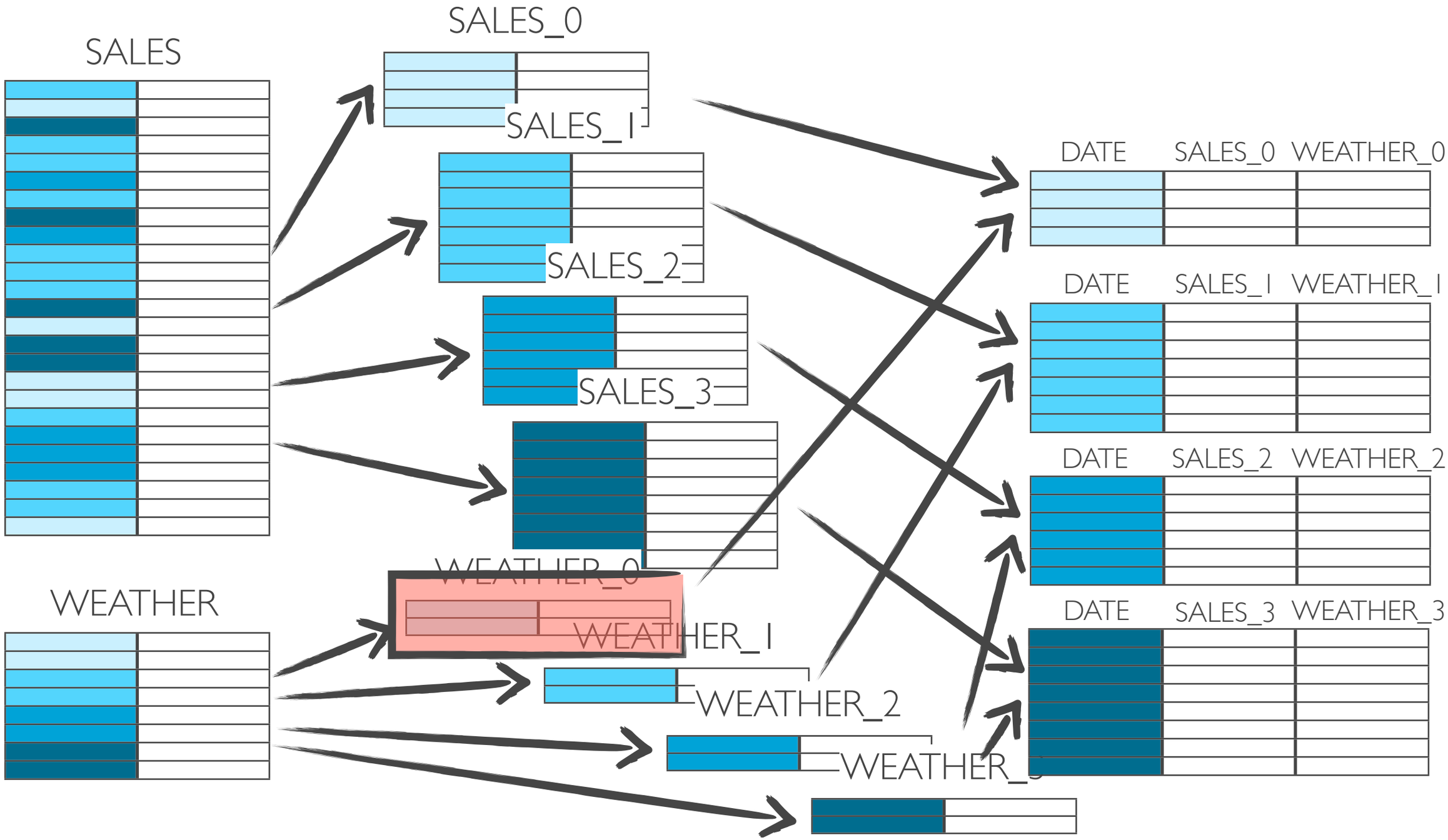
# Partitioned JOIN



# Partitioned JOIN



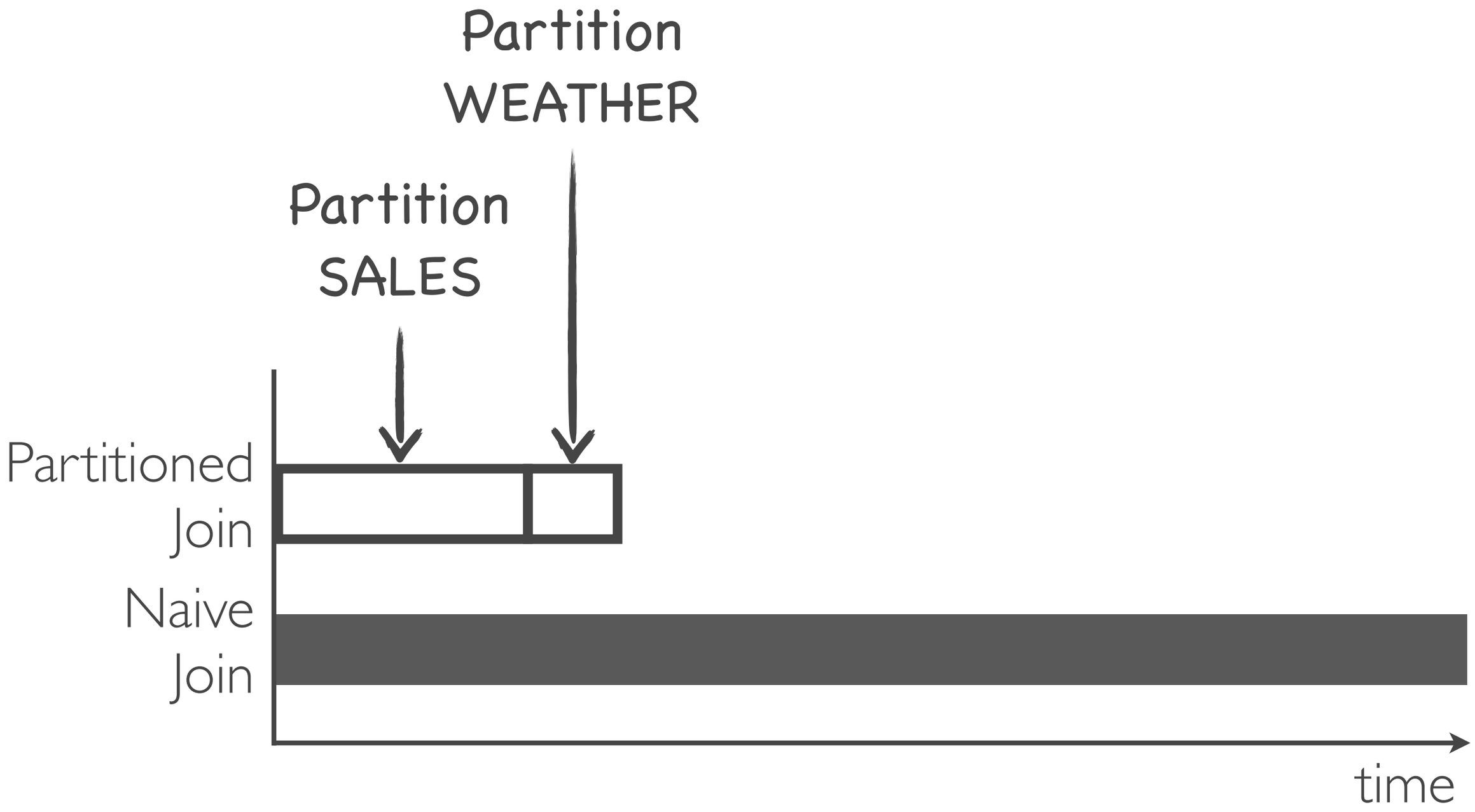
# Partitioned JOIN



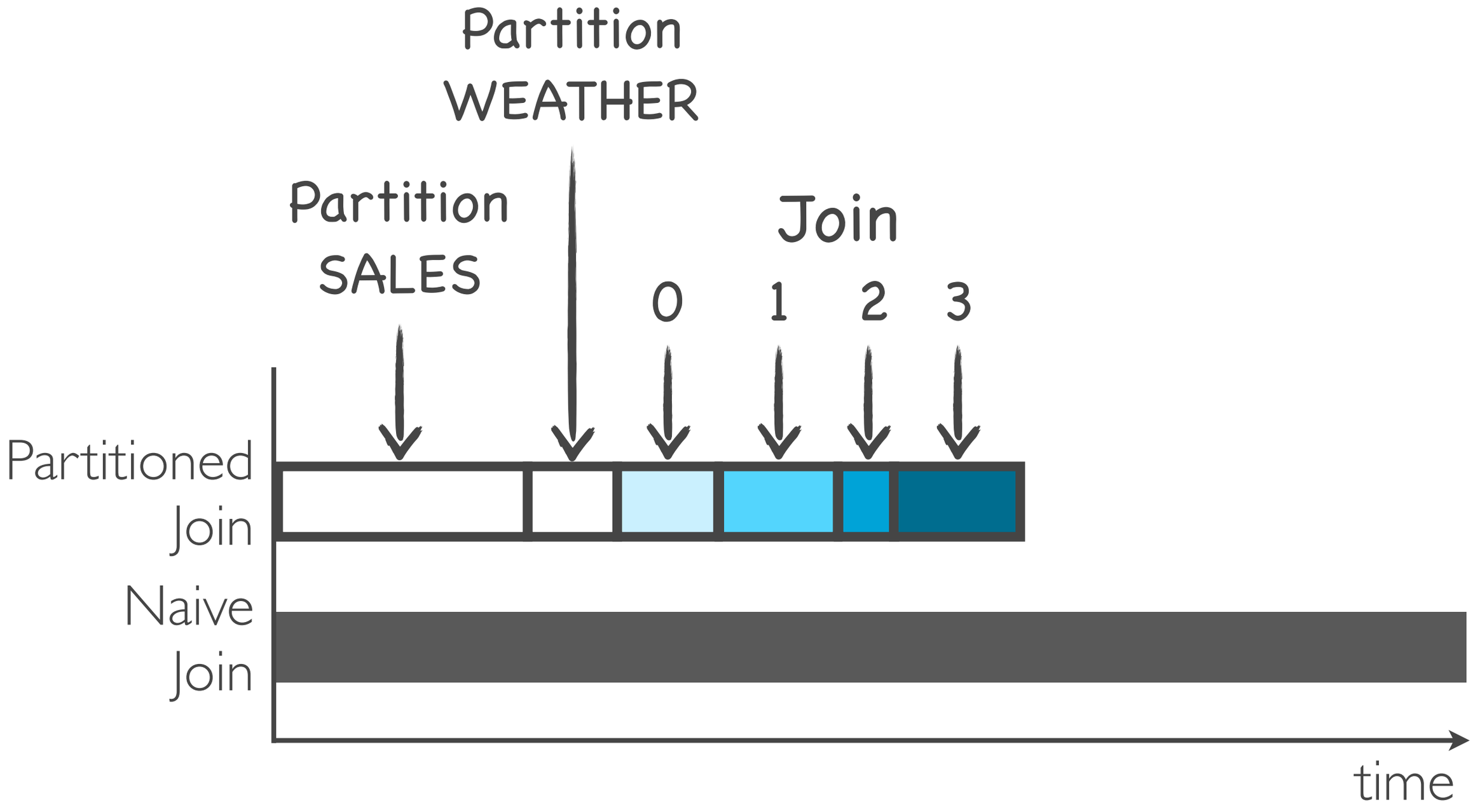
# JOIN Runtime

# JOIN Runtime

# JOIN Runtime

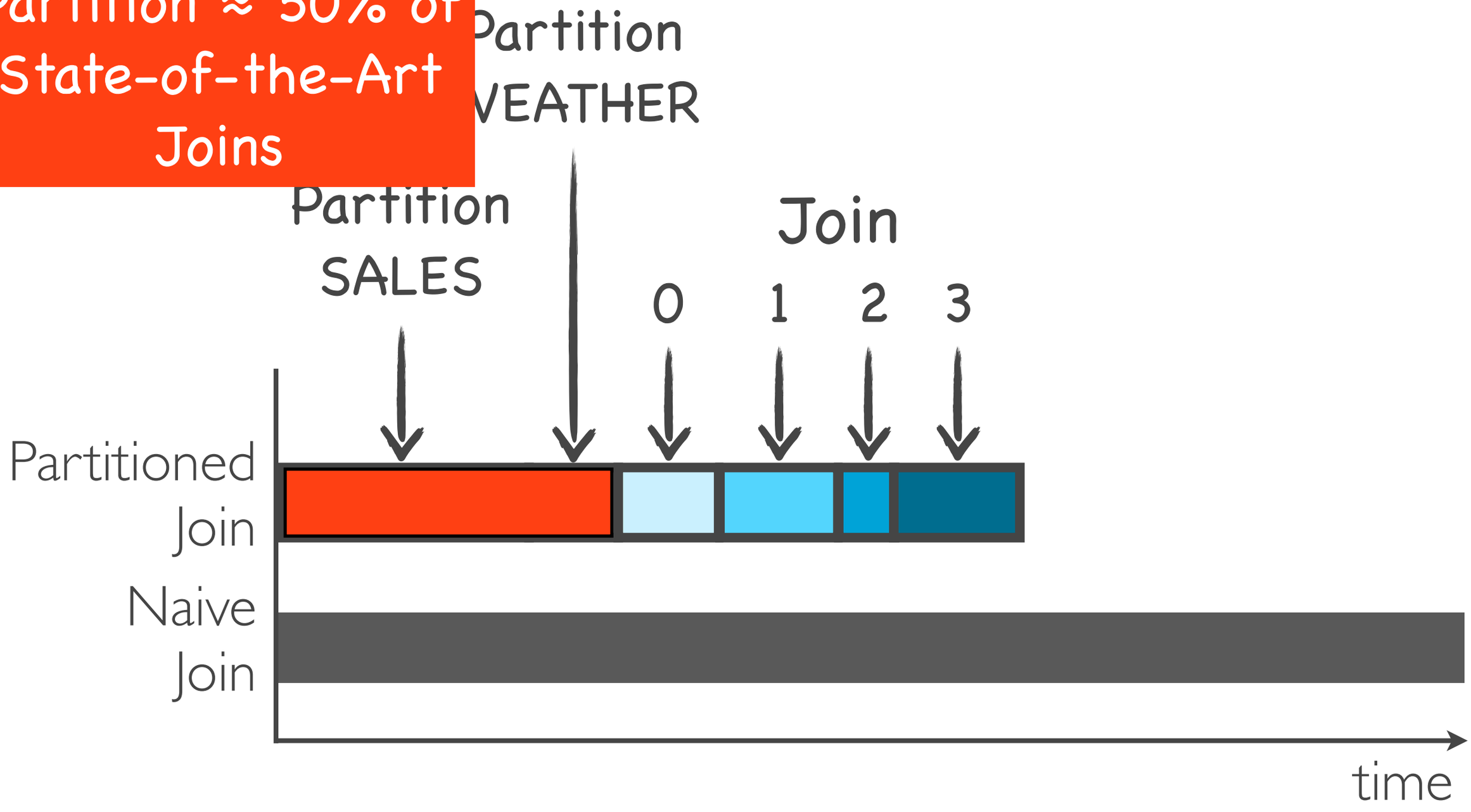


# JOIN Runtime



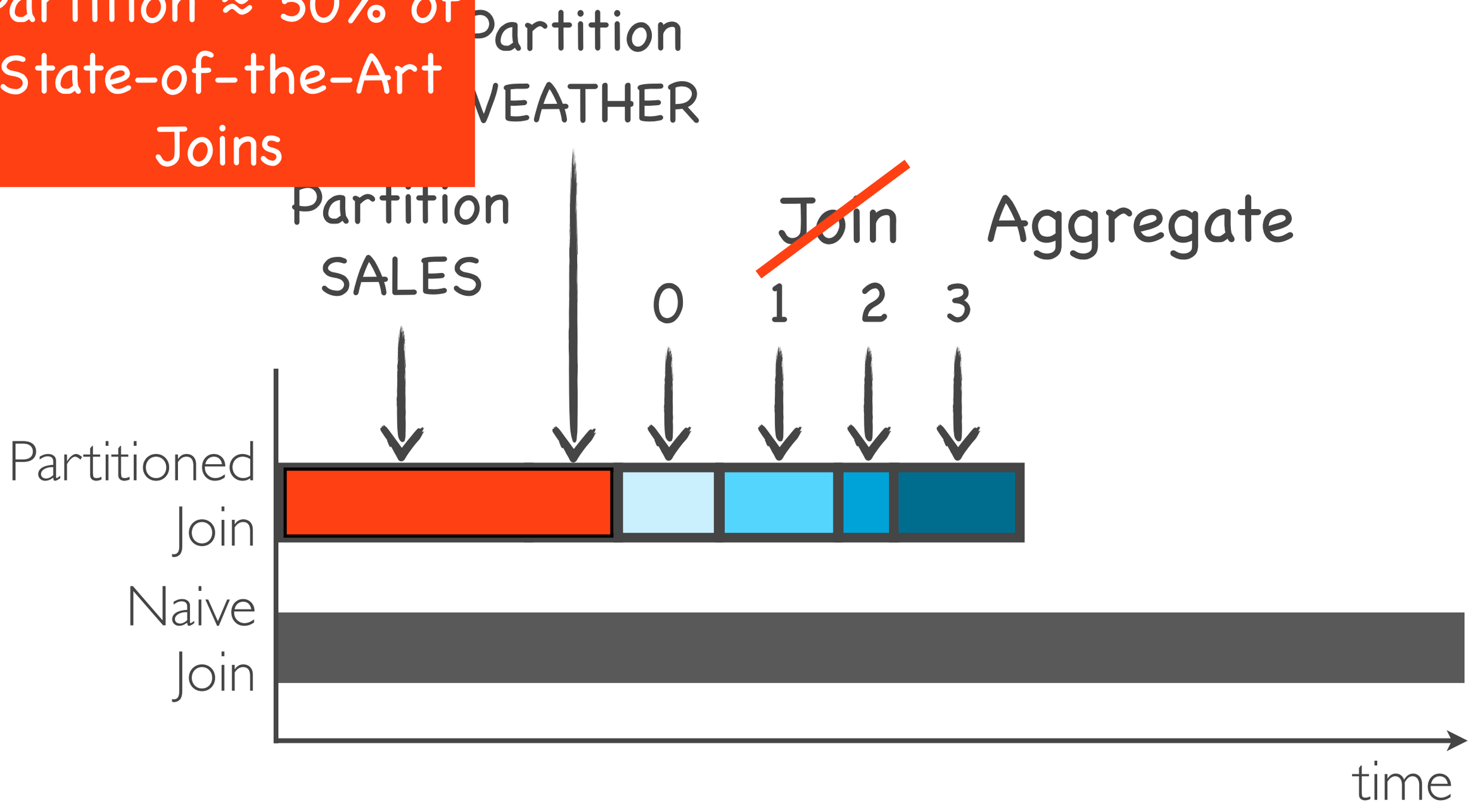
# JOIN Runtime

Partition ≈ 50% of State-of-the-Art Joins



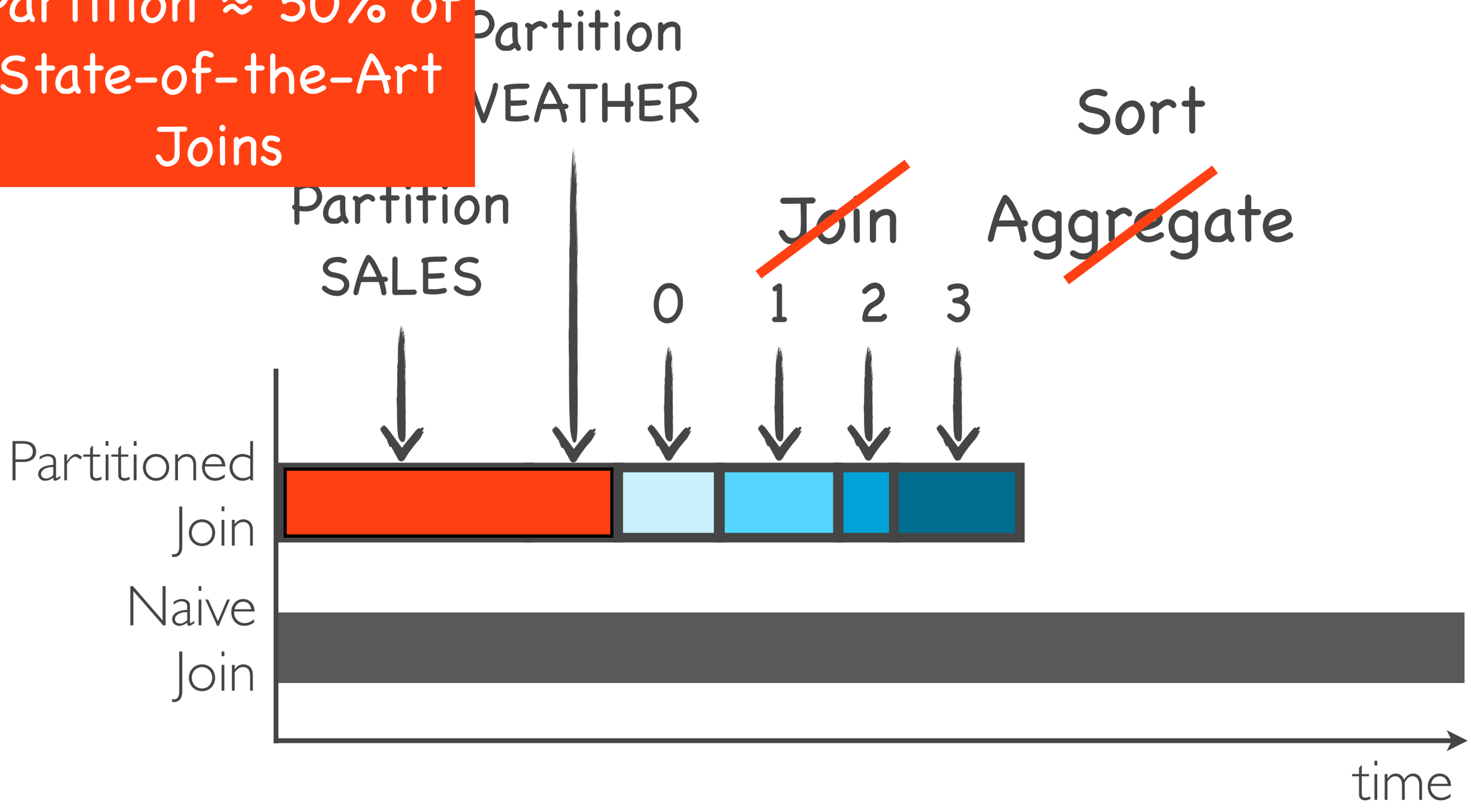
# JOIN Runtime

Partition ≈ 50% of State-of-the-Art Joins



# JOIN Runtime

Partition ≈ 50% of State-of-the-Art Joins



# Data Partitioning is...

# Data Partitioning is...

- broadly applicable in the database domain
  - partitioned-operations reduce I/O cost, increase parallel processing, and reduce shipping costs

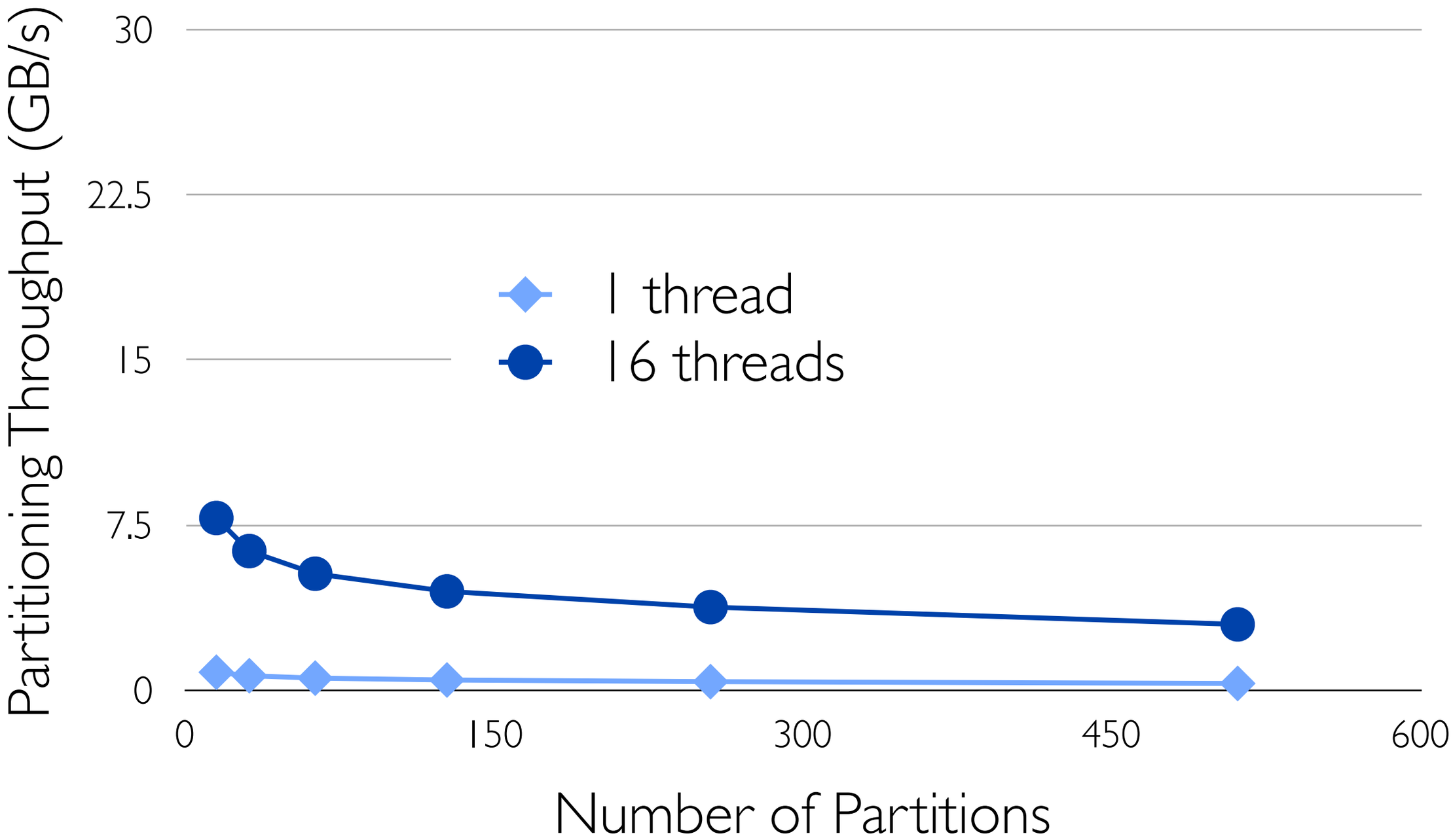
# Data Partitioning is...

- broadly applicable in the database domain
  - partitioned-operations reduce I/O cost, increase parallel processing, and reduce shipping costs
- widely used by commercial databases
  - Oracle 11g, IBM DB2, Microsoft SQL Server

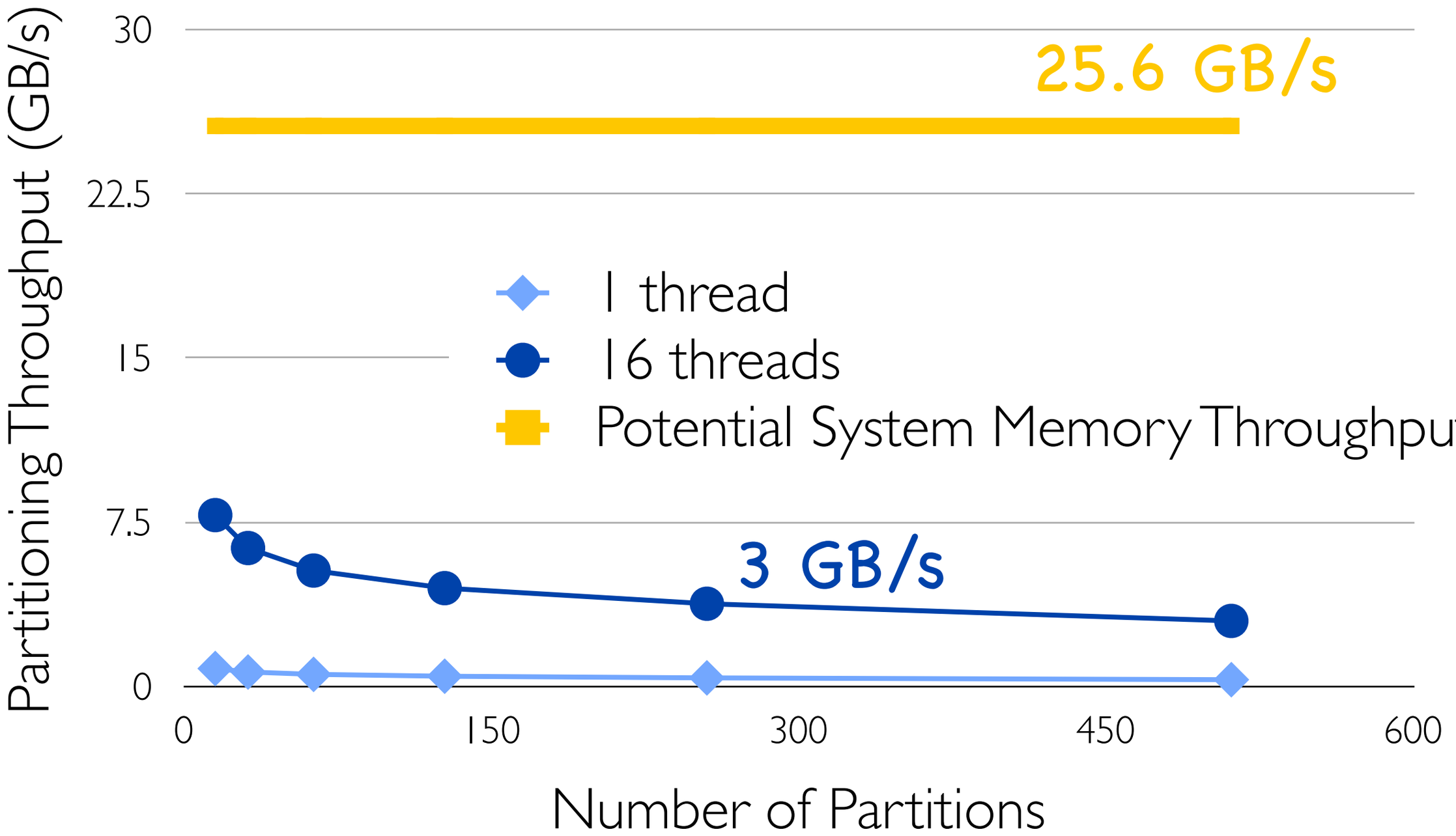
# Data Partitioning is...

- broadly applicable in the database domain
  - partitioned-operations reduce I/O cost, increase parallel processing, and reduce shipping costs
- widely used by commercial databases
  - Oracle 11g, IBM DB2, Microsoft SQL Server
- applicable in the non-database domain
  - divide-and-conquer, map-reduce

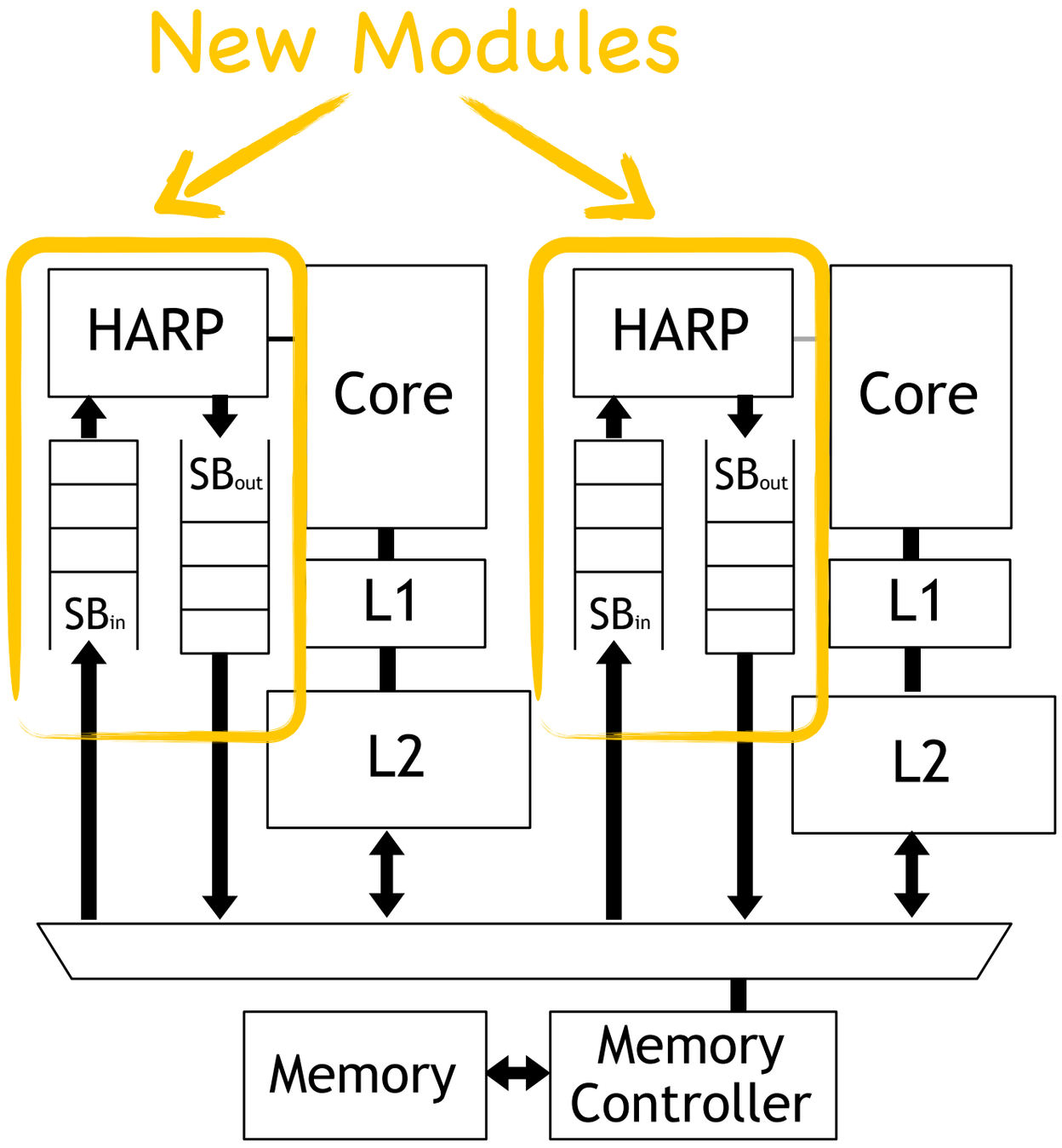
# SW Partitioning Performance



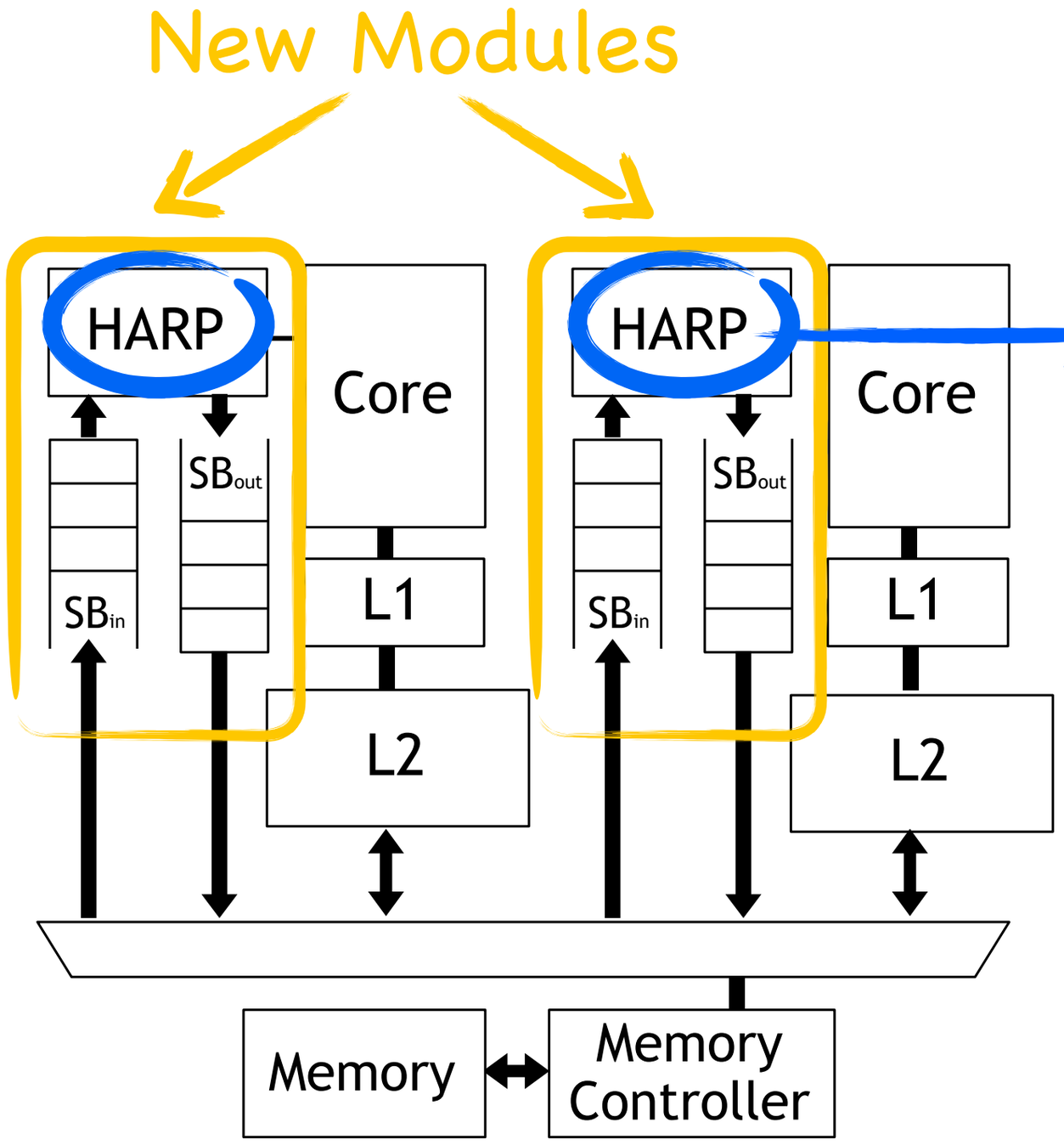
# SW Partitioning Performance



# Research Overview

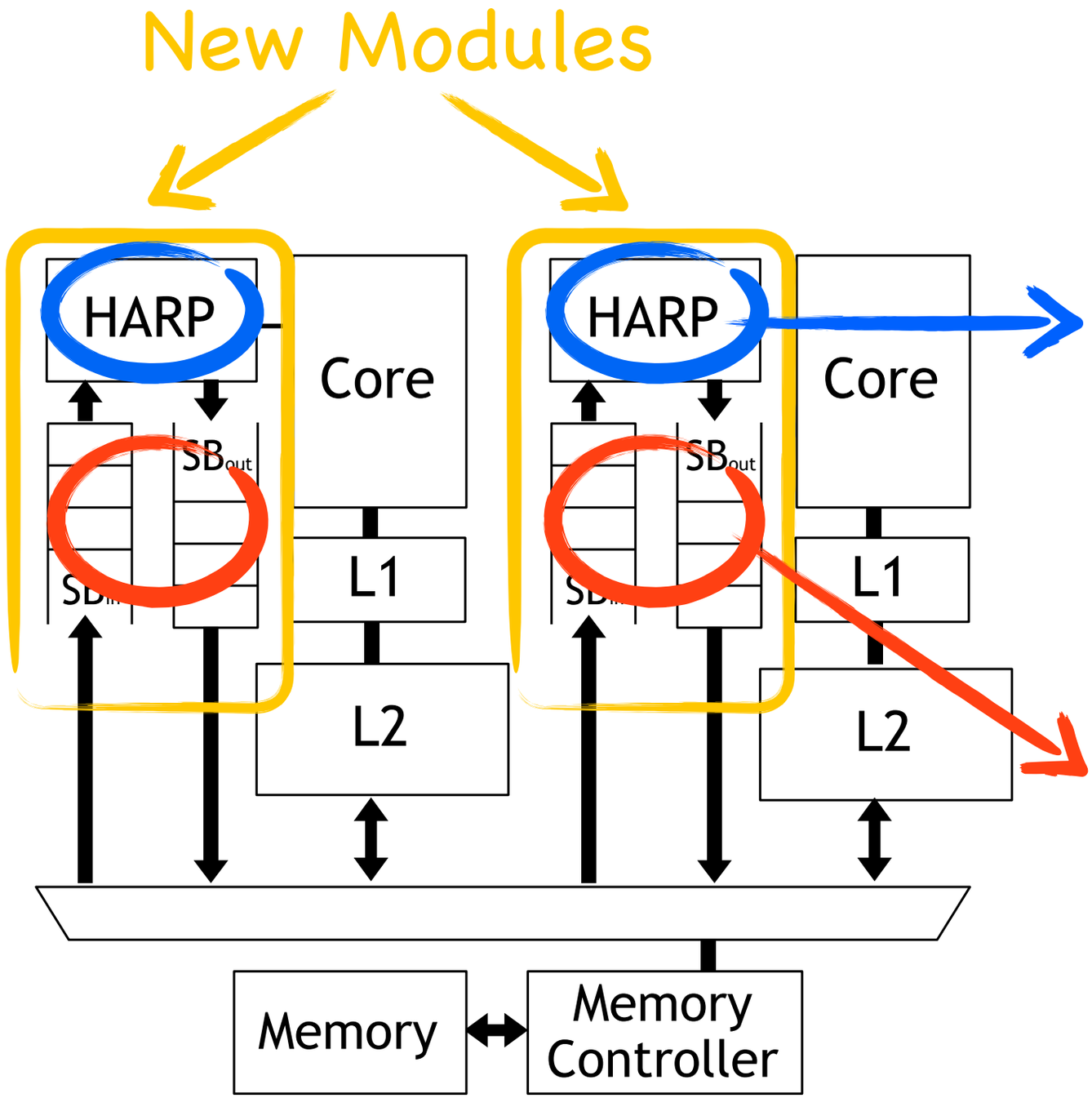


# Research Overview




Hardware accelerated range partitioner (HARP): 7.8X more performance @ 7.5X less energy

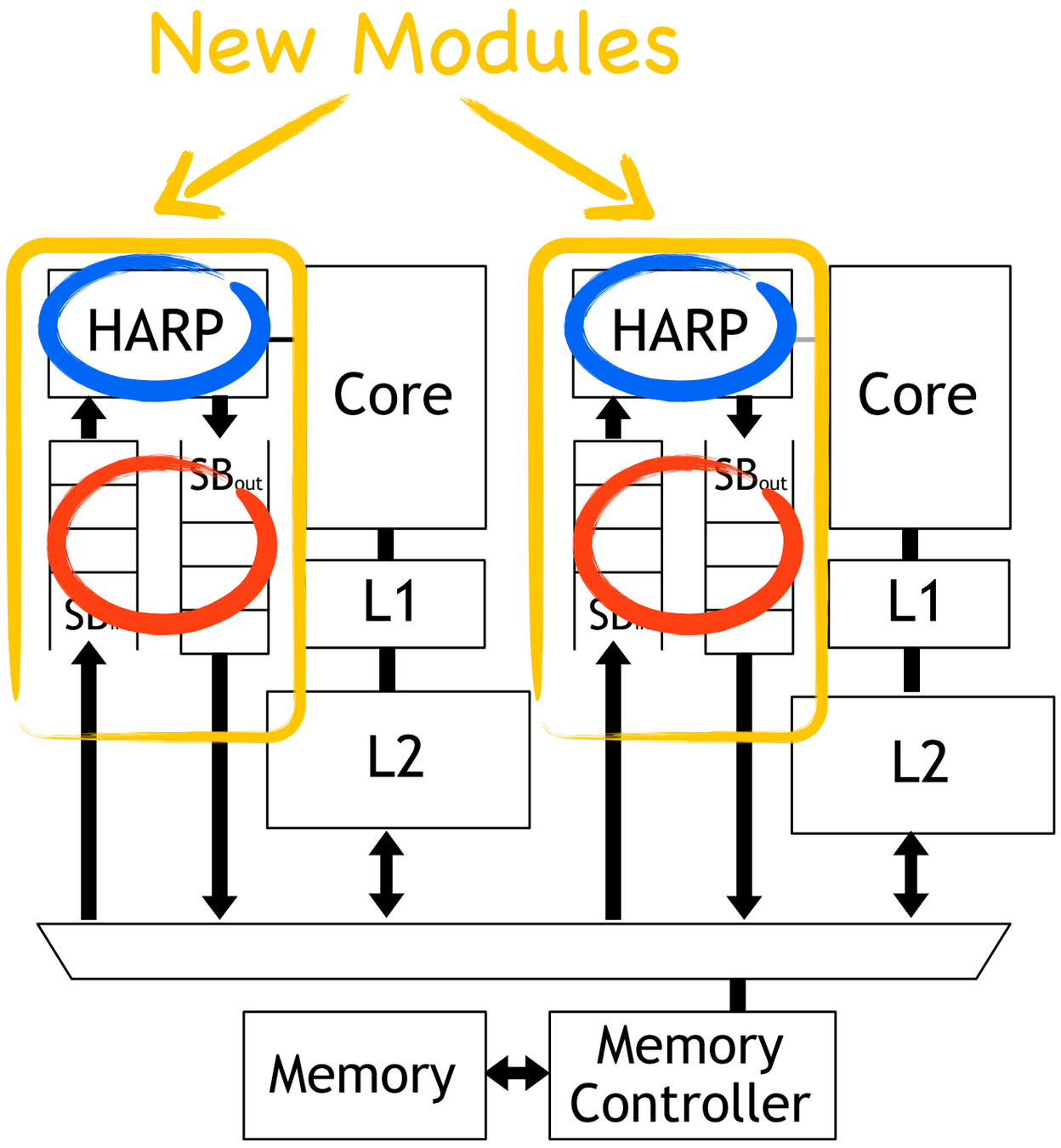
# Research Overview




Hardware accelerated range partitioner (HARP): 7.8X more performance @ 7.5X less energy

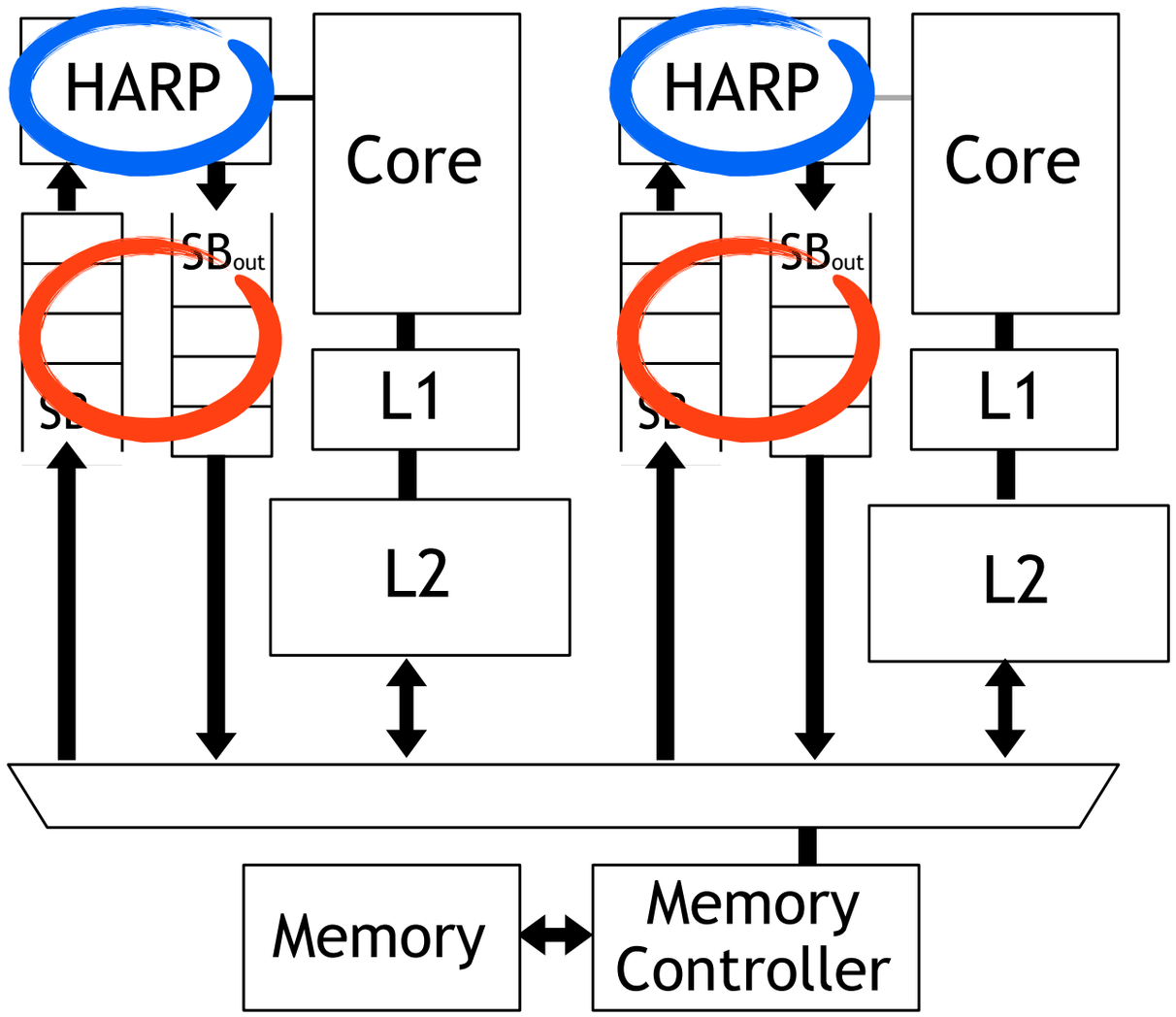
Streaming framework: Can keep up with the throughput of HARP

# Remainder of the Talk

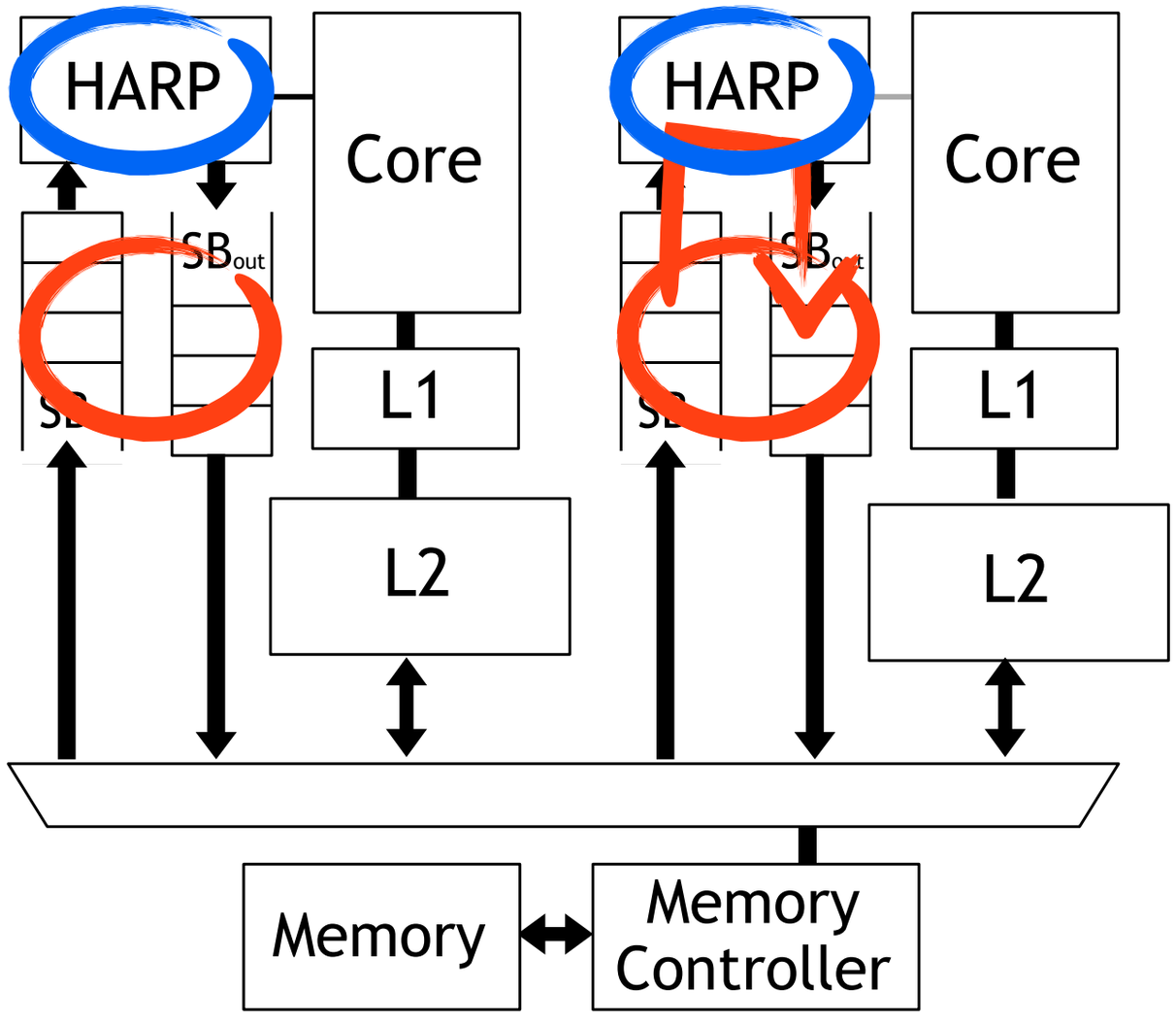



- Brief System Overview
- HARP UArch
- Streaming Framework UArch
- HARP and Streaming Framework Evaluation
- Discussion

# HARP System Architecture

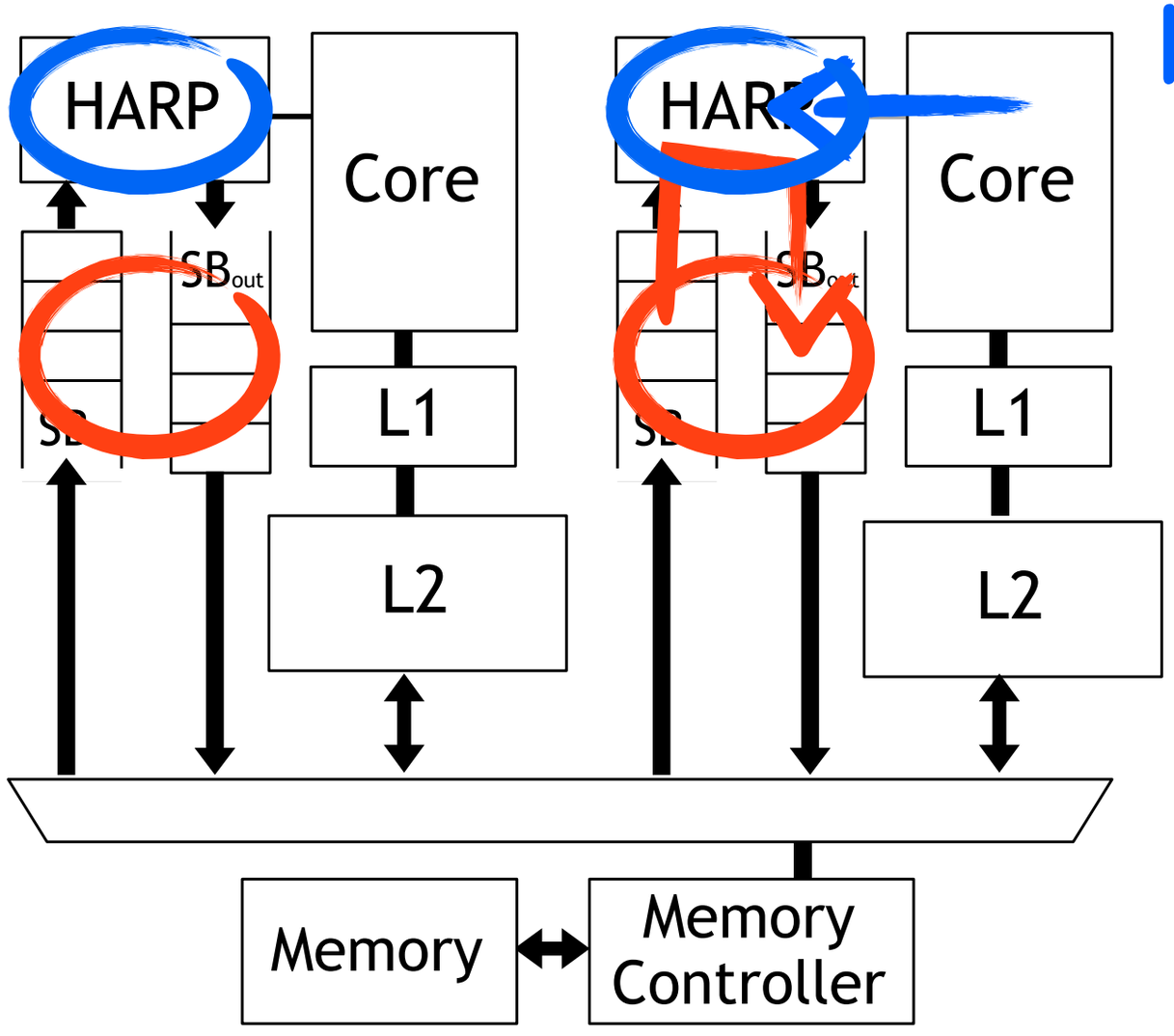


# HARP System Architecture




HARP communicates to/from memory through stream buffers: SBin, SBout

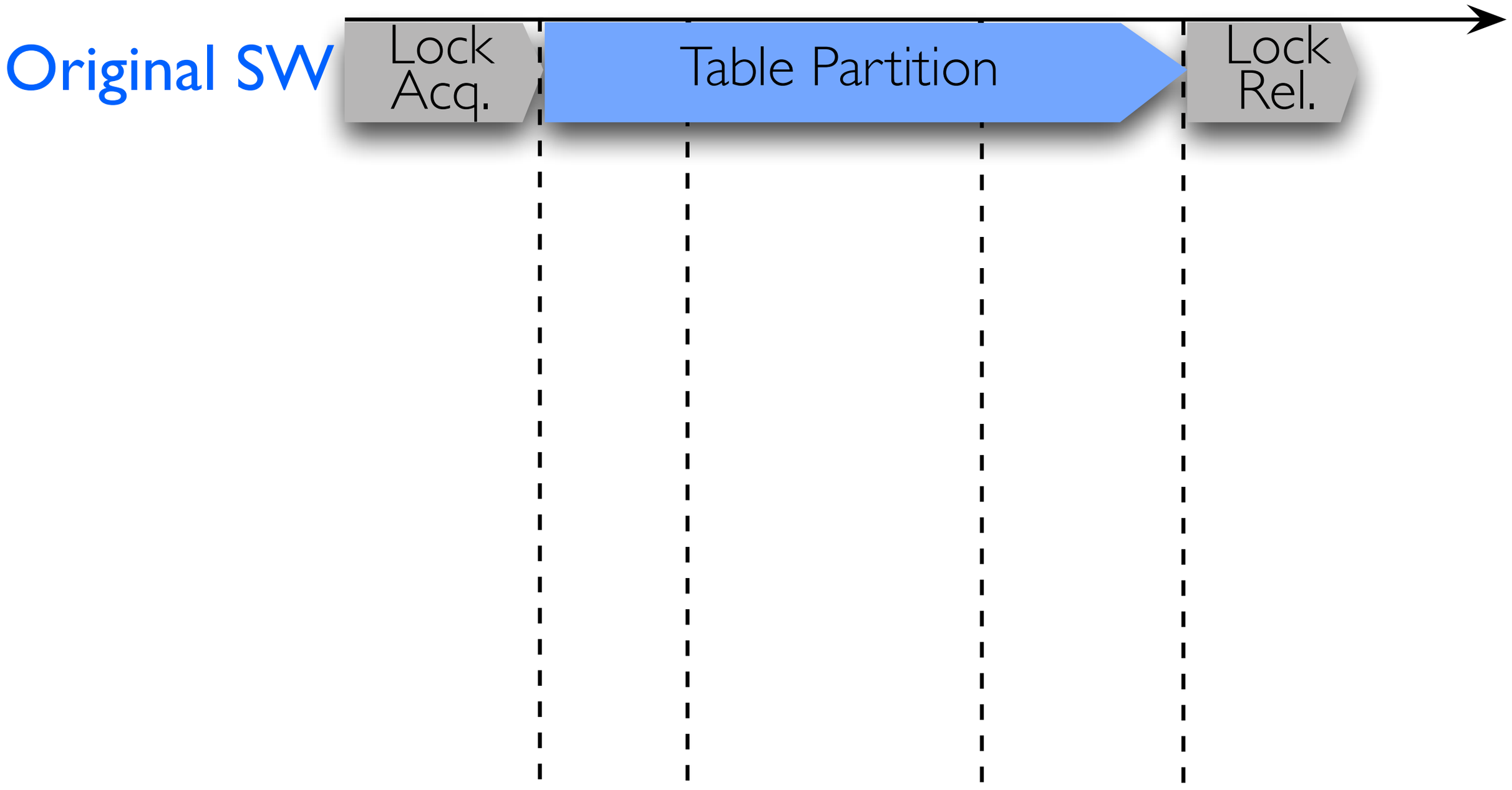
# HARP System Architecture



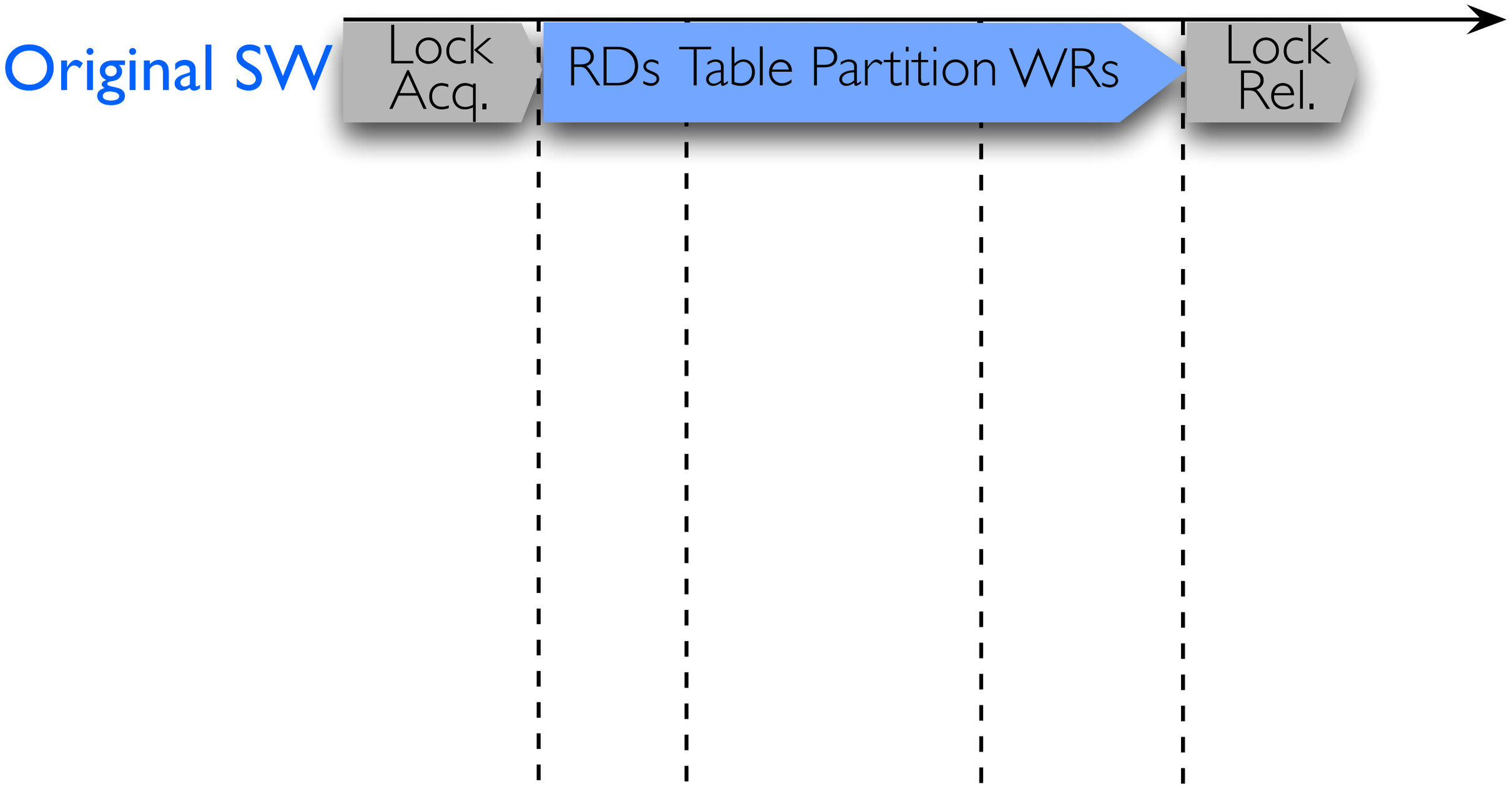

HW Partitioning with SW configuration

HARP communicates to/from memory through stream buffers: SBin, SBout

# Programming Model



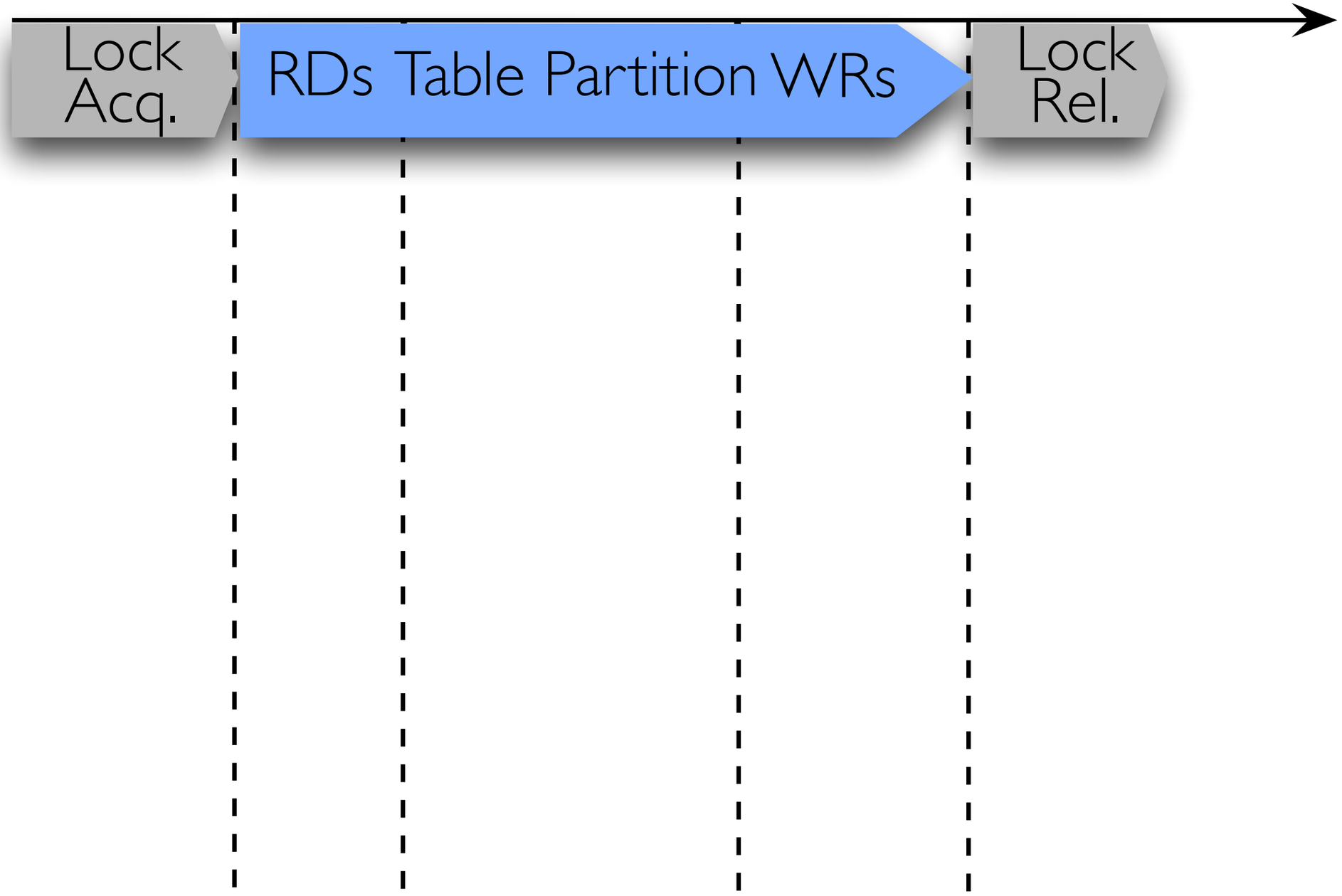
# Programming Model



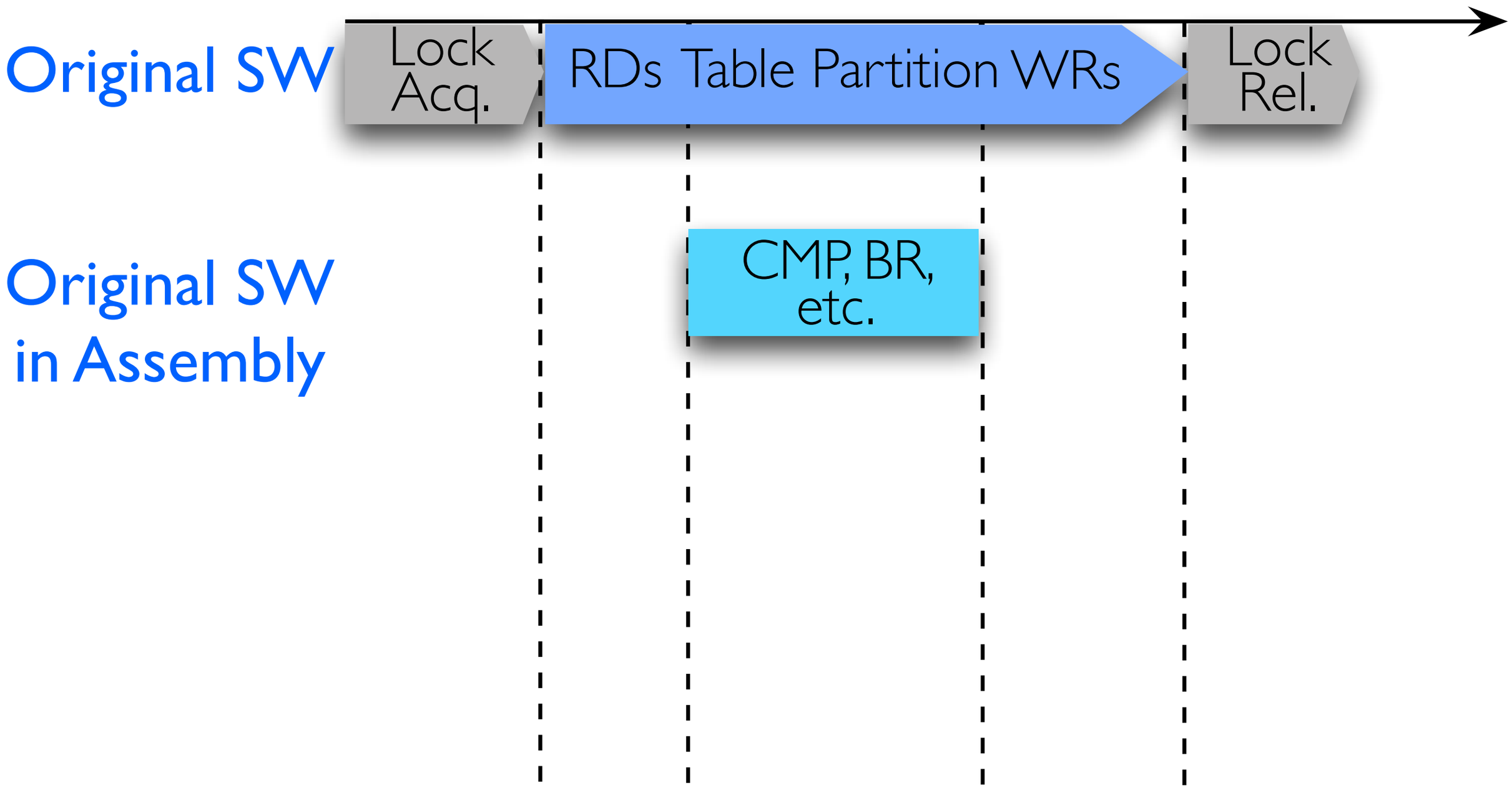
# Programming Model

Original SW

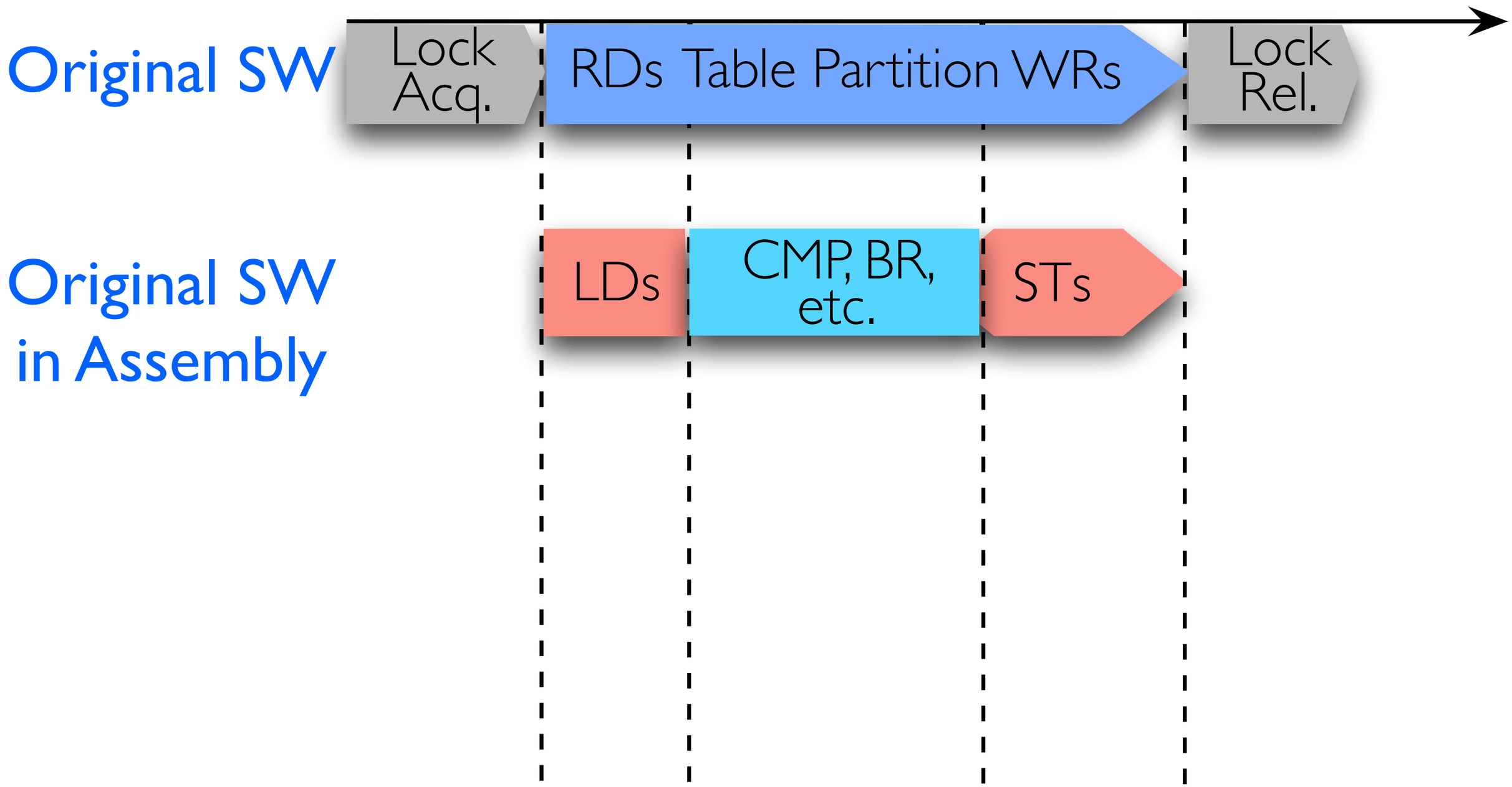
Original SW in Assembly



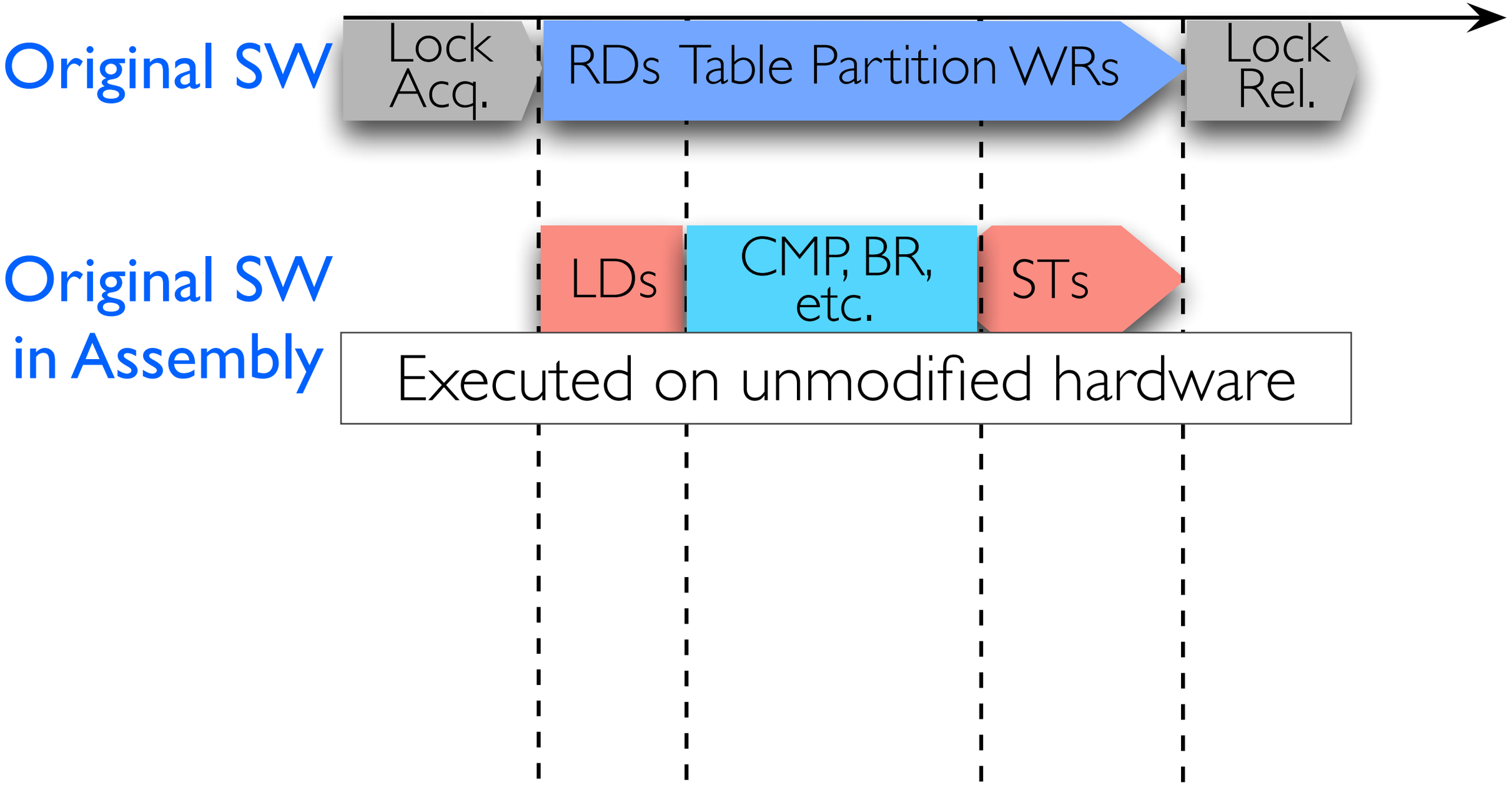
# Programming Model



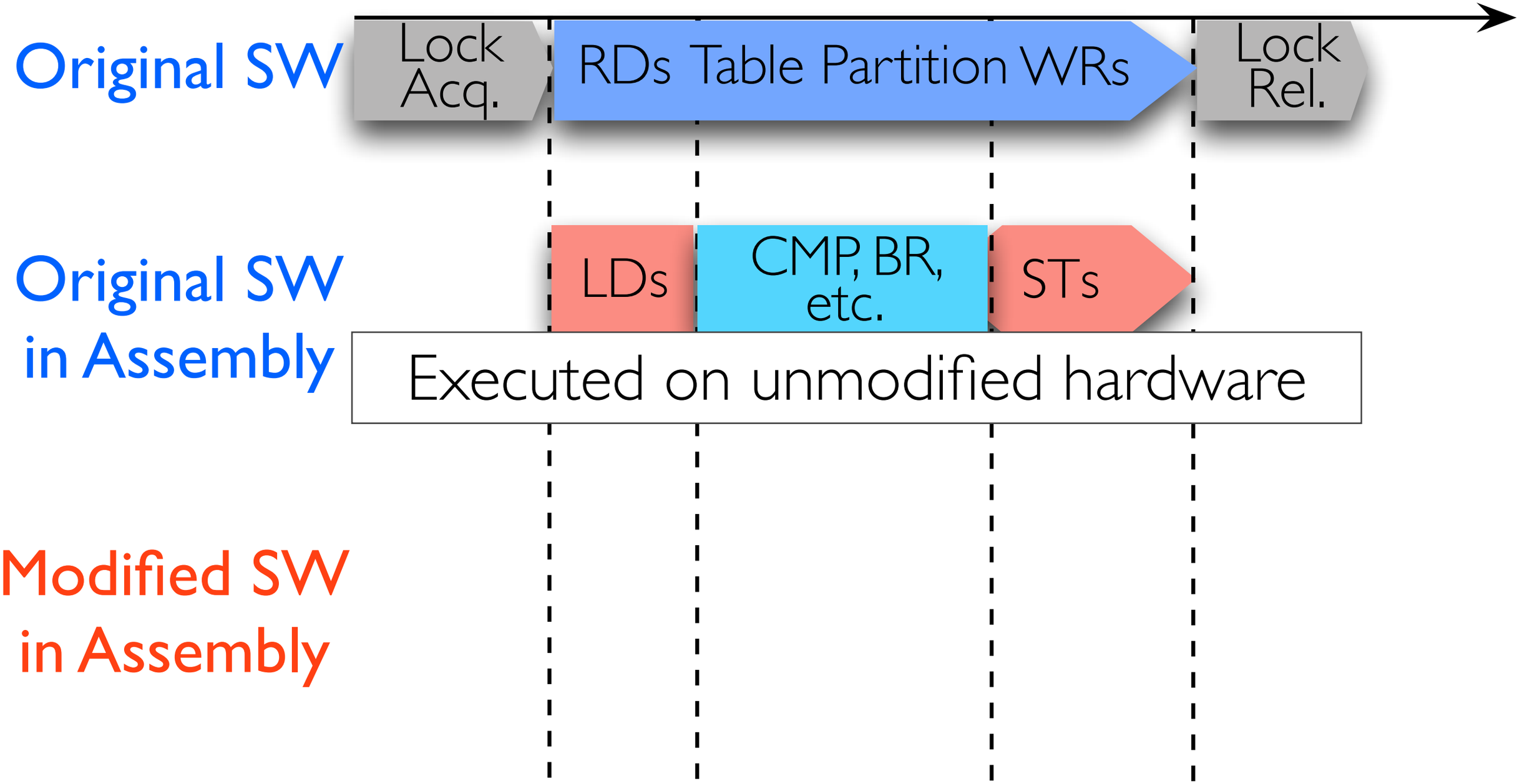
# Programming Model



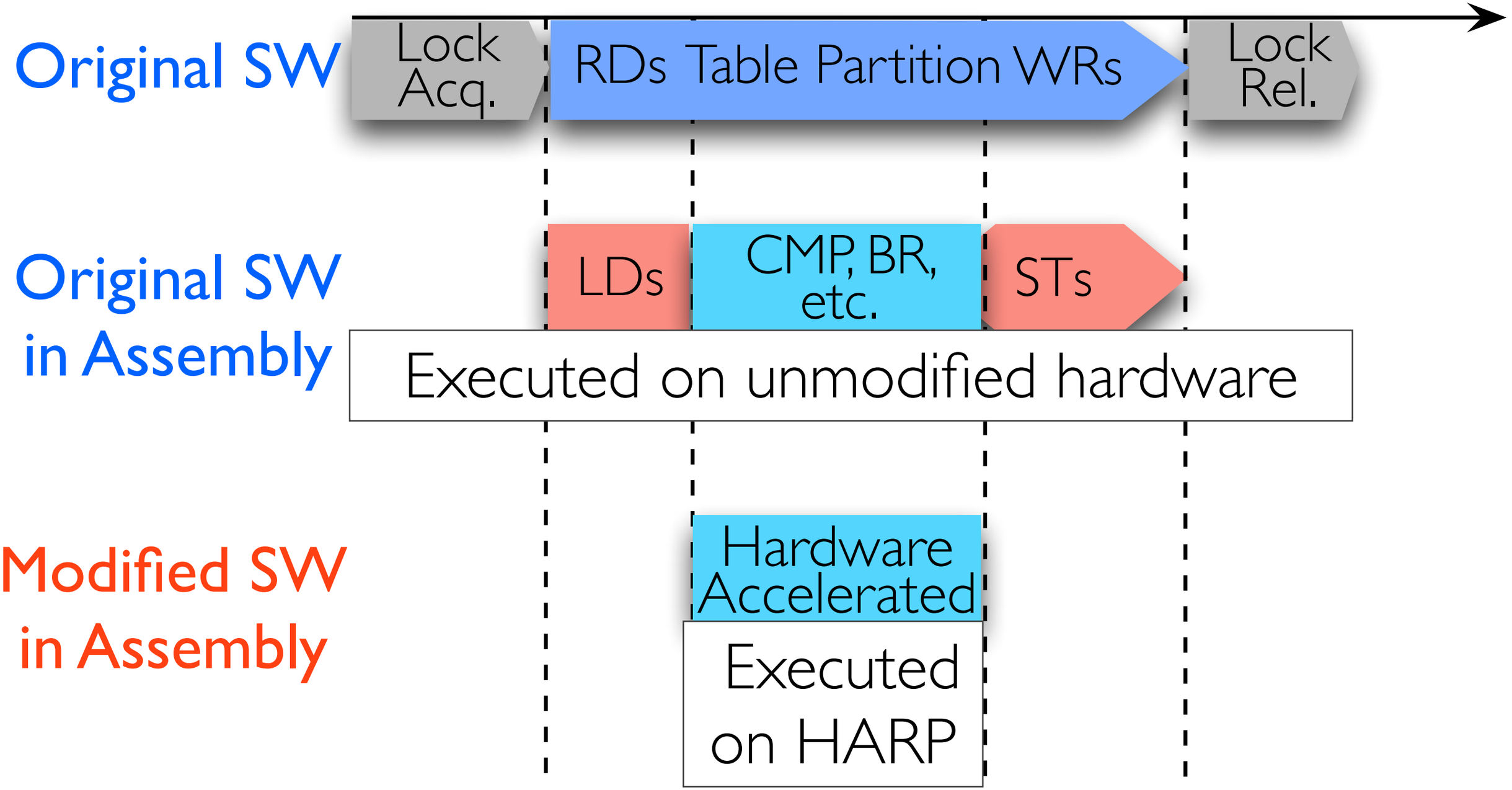
# Programming Model



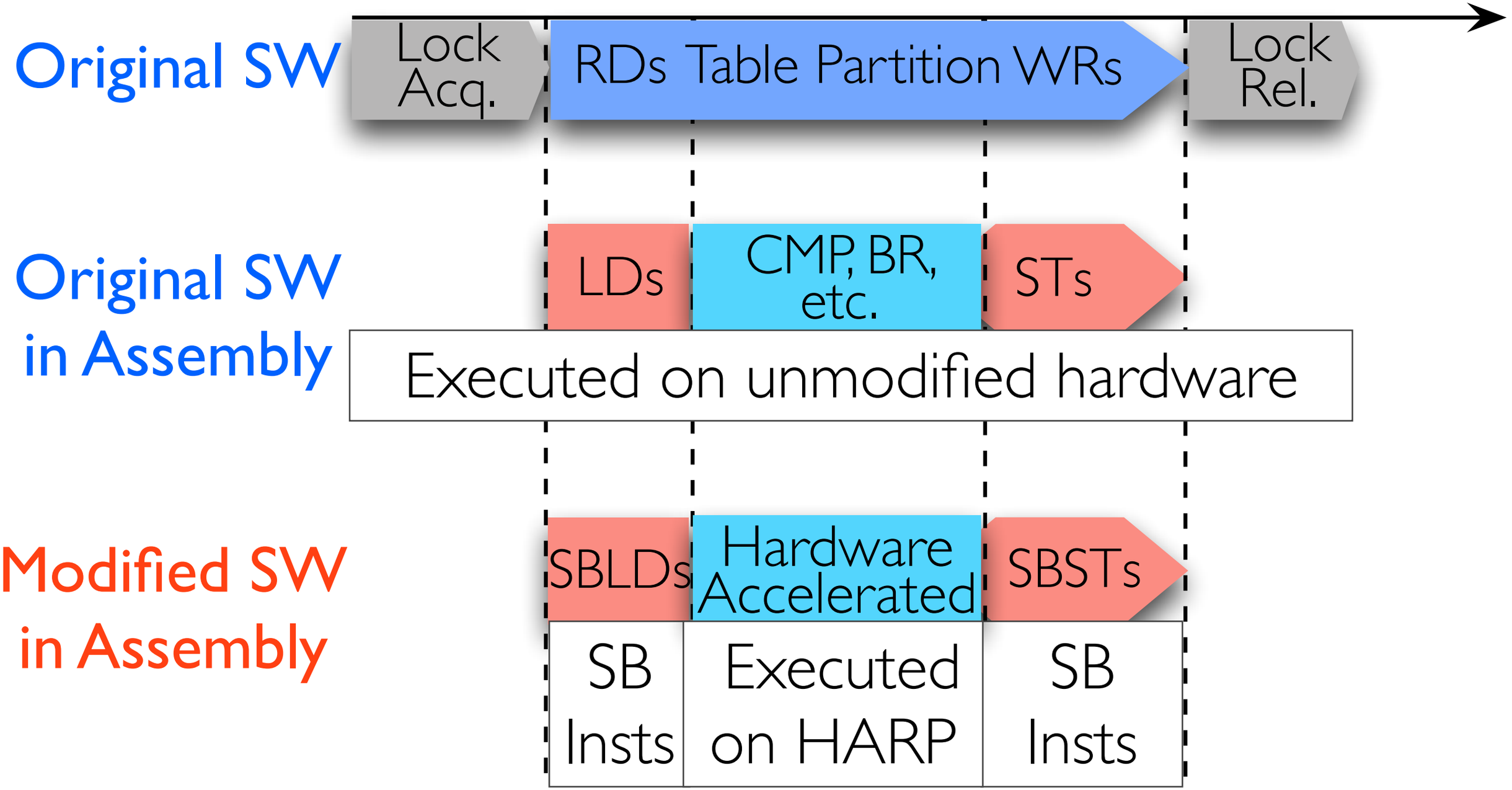
# Programming Model



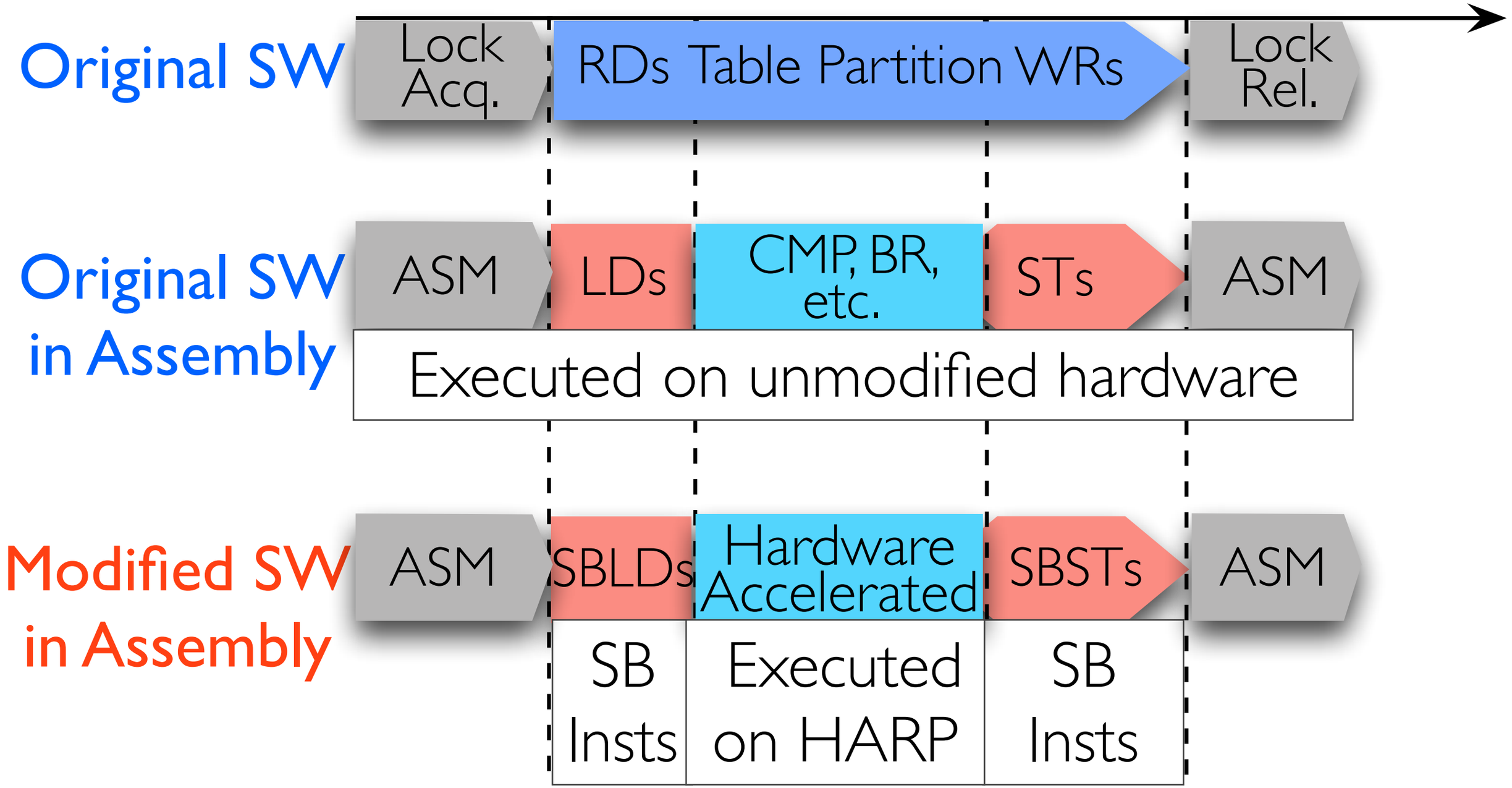
# Programming Model



# Programming Model



# Programming Model



# Range Partition

| X | Y | Z |
|---|---|---|
| 15 | | |
| 8 | | |
| 27 | | |
| 20 | | |
| 52 | | |
| 29 | | |
| 16 | | |
| 31 | | |

# Range Partition

key

| X | Y | Z |
|---|---|---|
| 15 | | |
| 8 | | |
| 27 | | |
| 20 | | |
| 52 | | |
| 29 | | |
| 16 | | |
| 31 | | |

# Range Partition

key

| X | Y | Z |
|---|---|---|
| 15 | | |
| 8 | | |
| 27 | | |
| 20 | | |
| 52 | | |
| 29 | | |
| 16 | | |
| 31 | | |

10

20

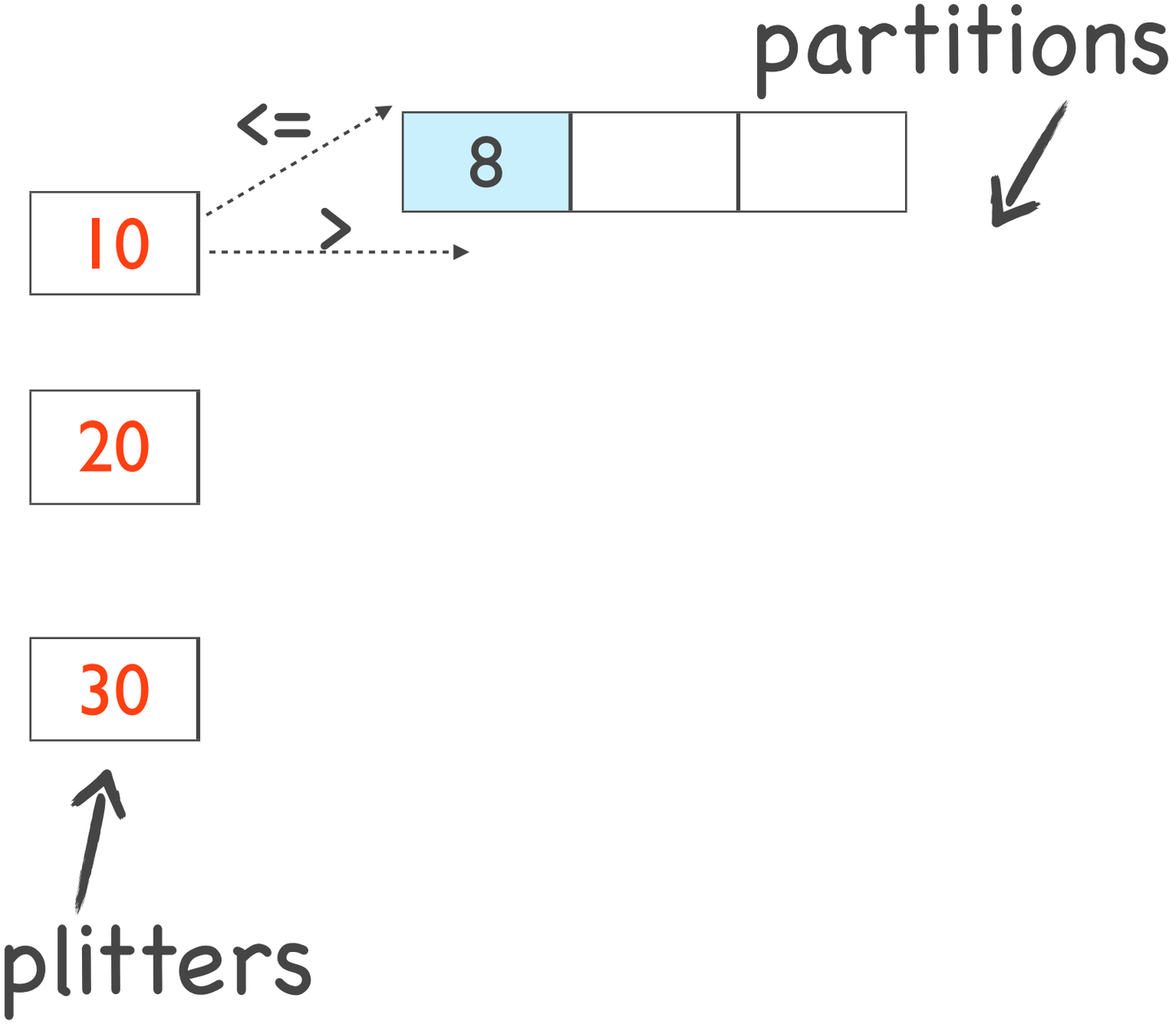
30

splitters

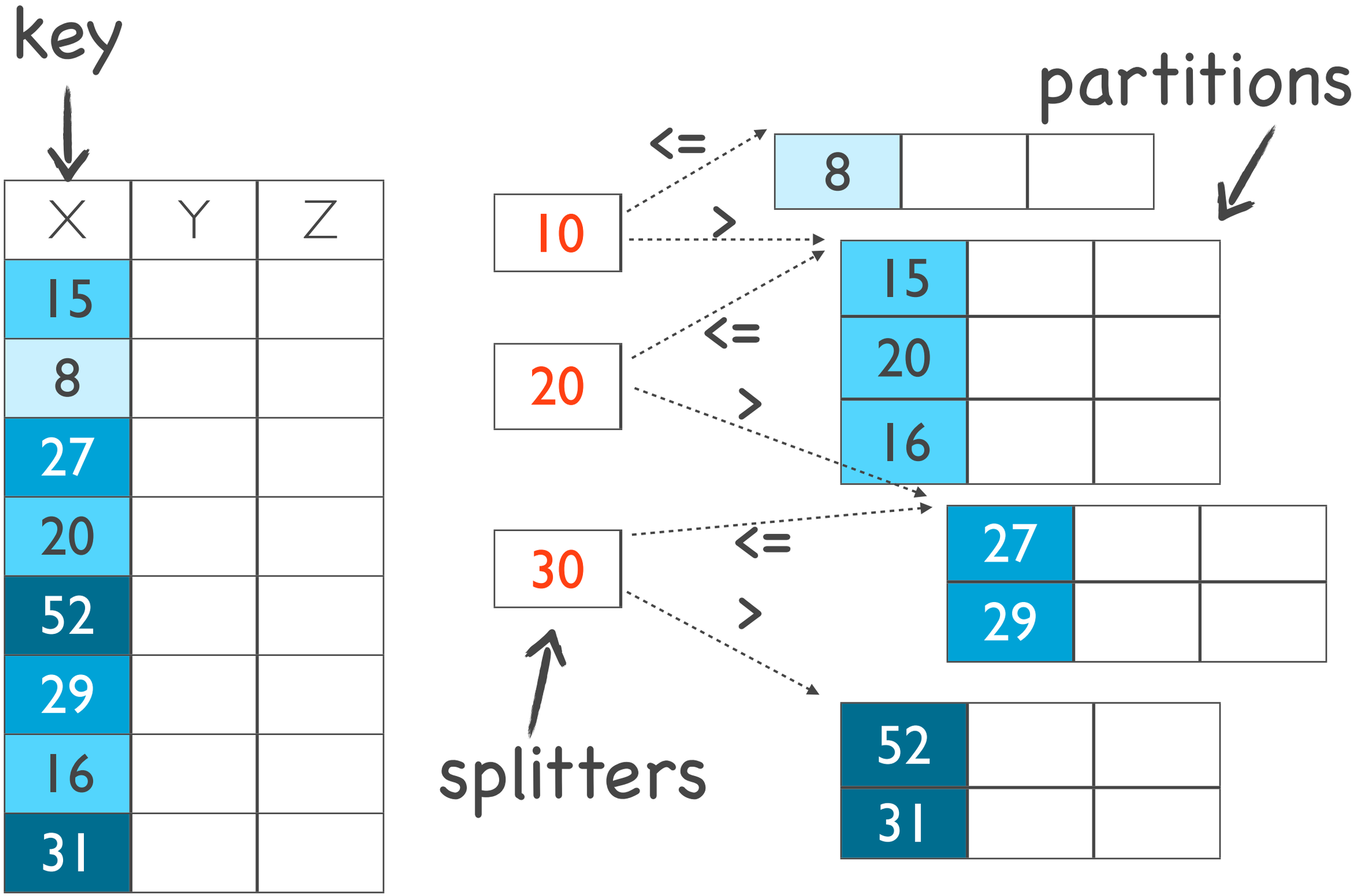
# Range Partition

key

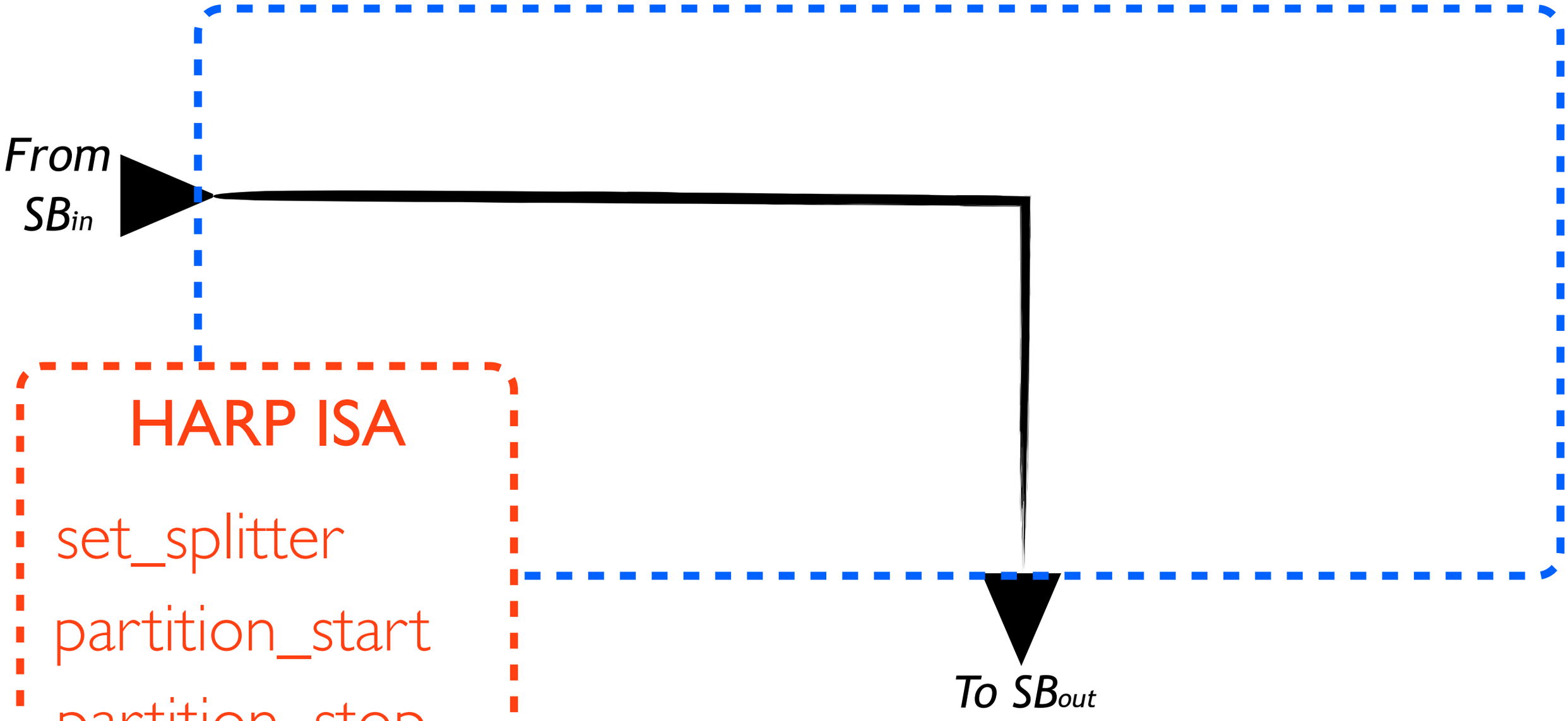
| X | Y | Z |
|---|---|---|
| 15 | | |
| 8 | | |
| 27 | | |
| 20 | | |
| 52 | | |
| 29 | | |
| 16 | | |
| 31 | | |



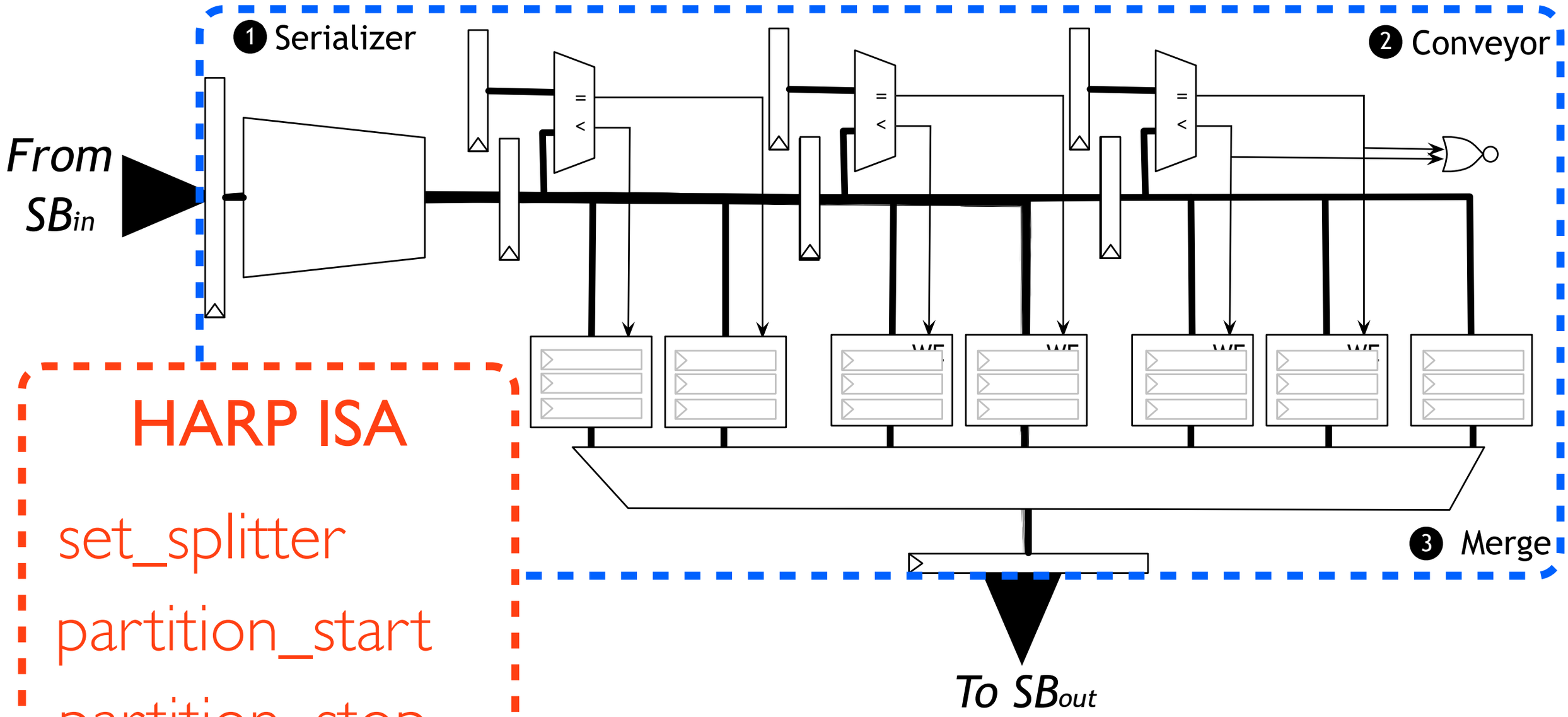
# Range Partition



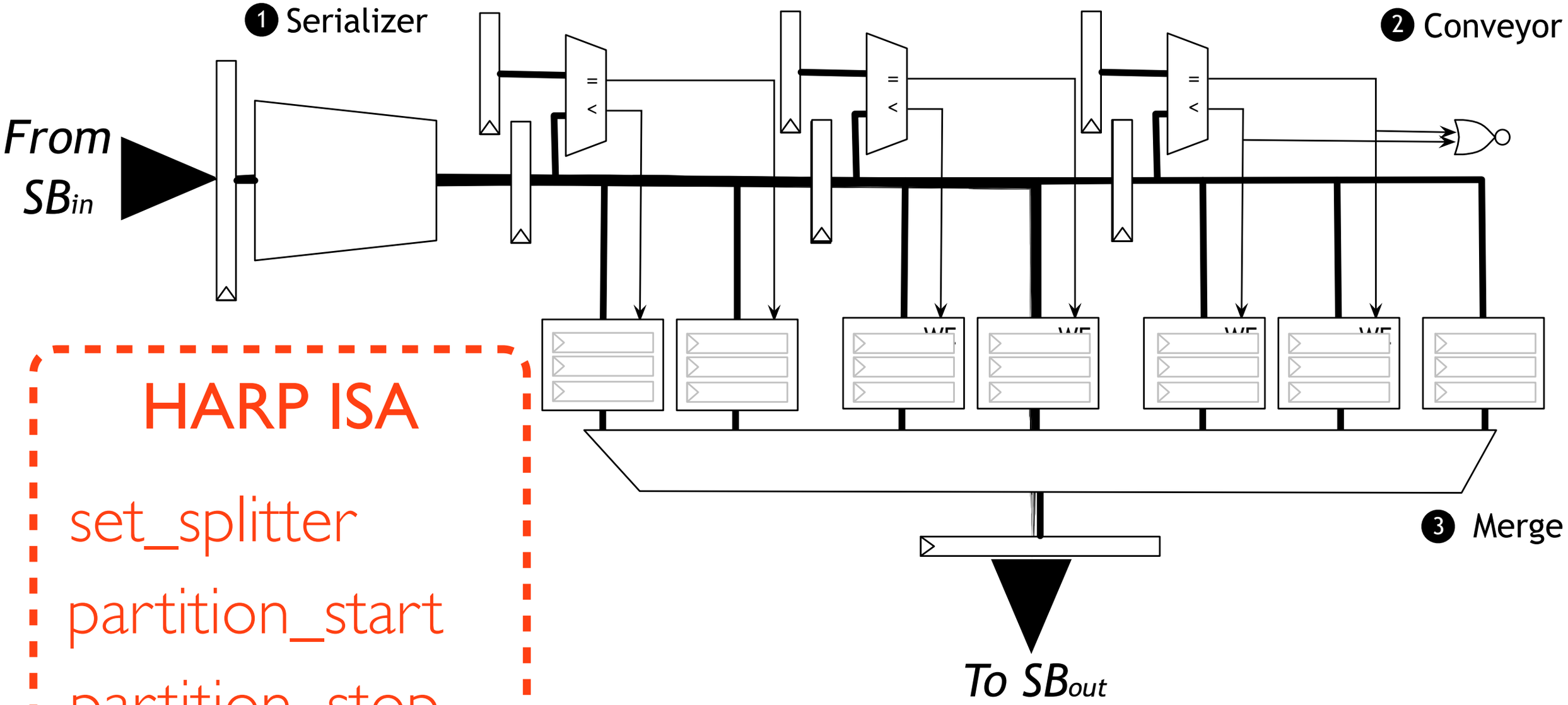
# HARP Microarchitecture



# HARP Microarchitecture



# Step 1: HARP Configuration



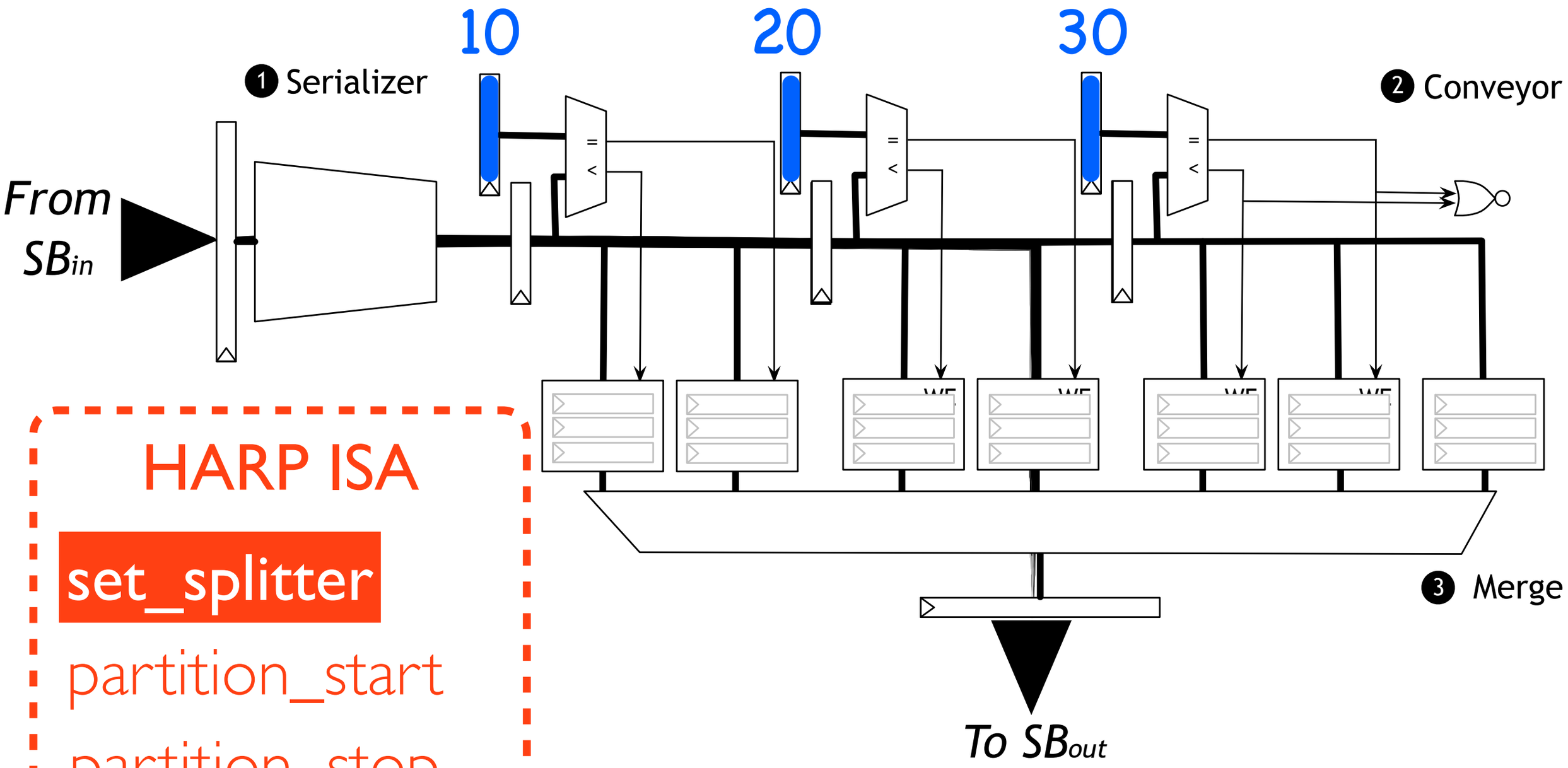

**HARP ISA**

set_splitter

partition_start

partition_stop

# Step 1: HARP Configuration



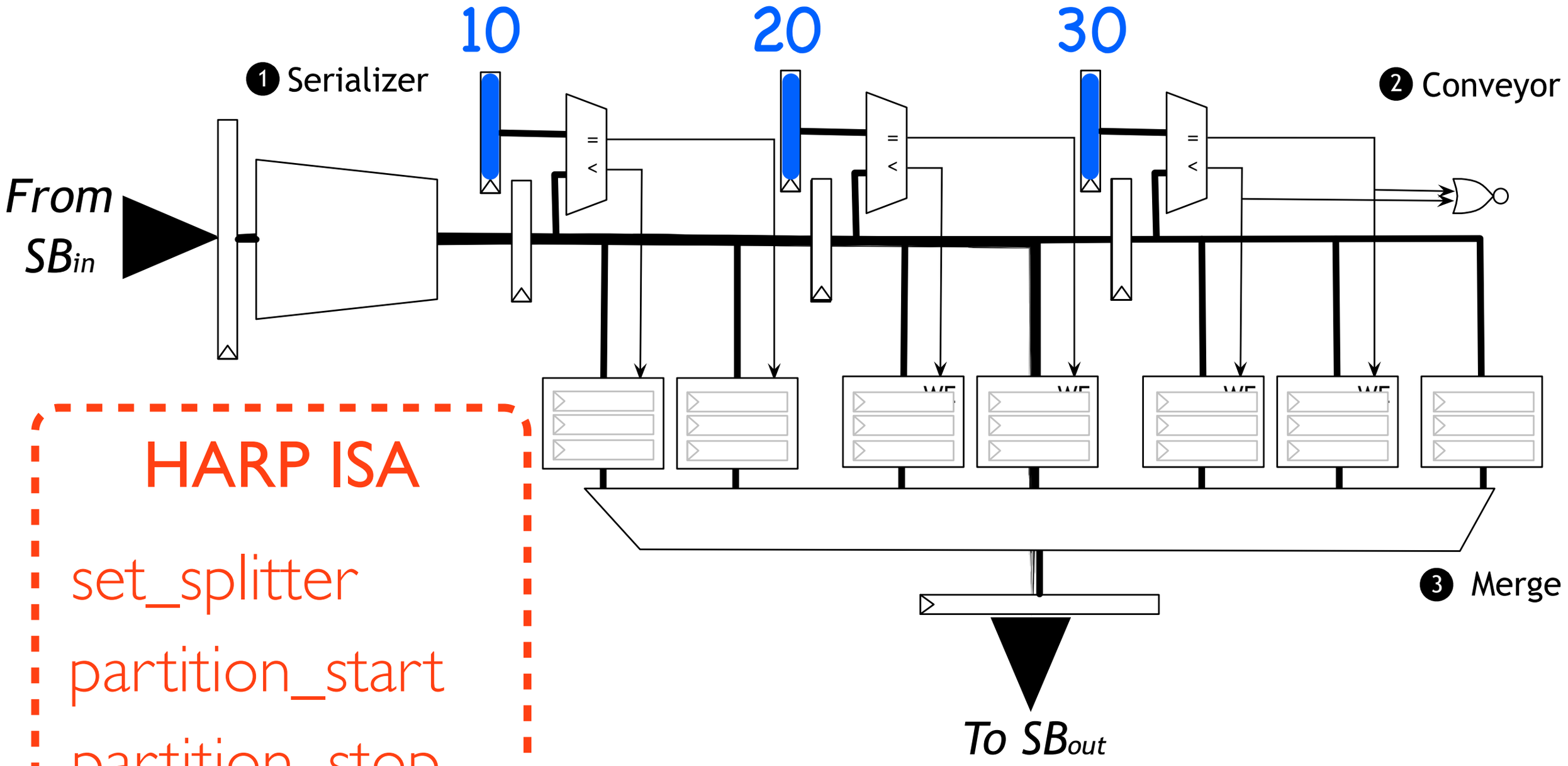

HARP ISA

**set_splitter**

partition_start

partition_stop

# Step 2: Signal HARP to Start Processing



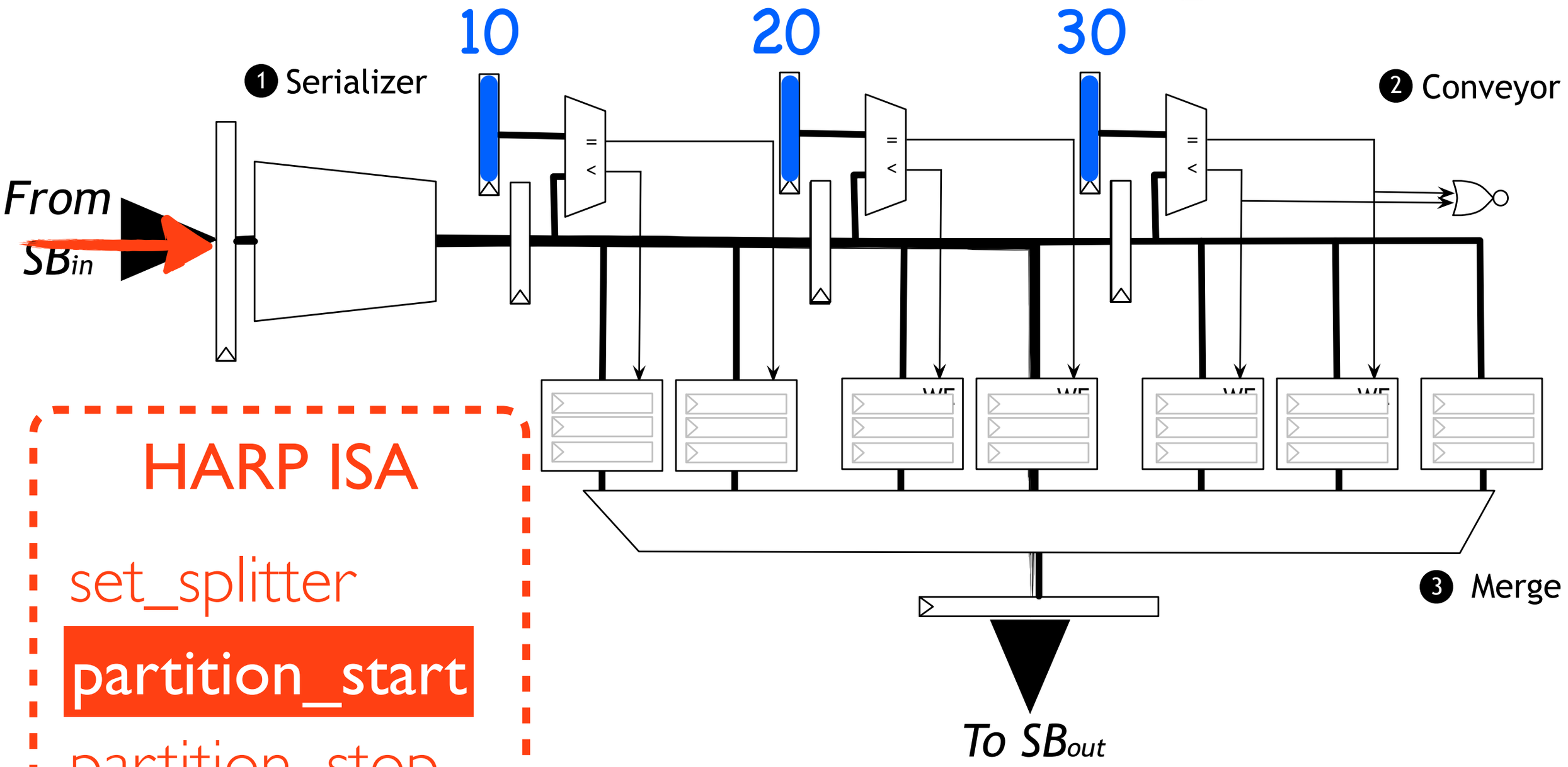

**HARP ISA**

set_splitter

partition_start

partition_stop

# Step 2: Signal HARP to Start Processing



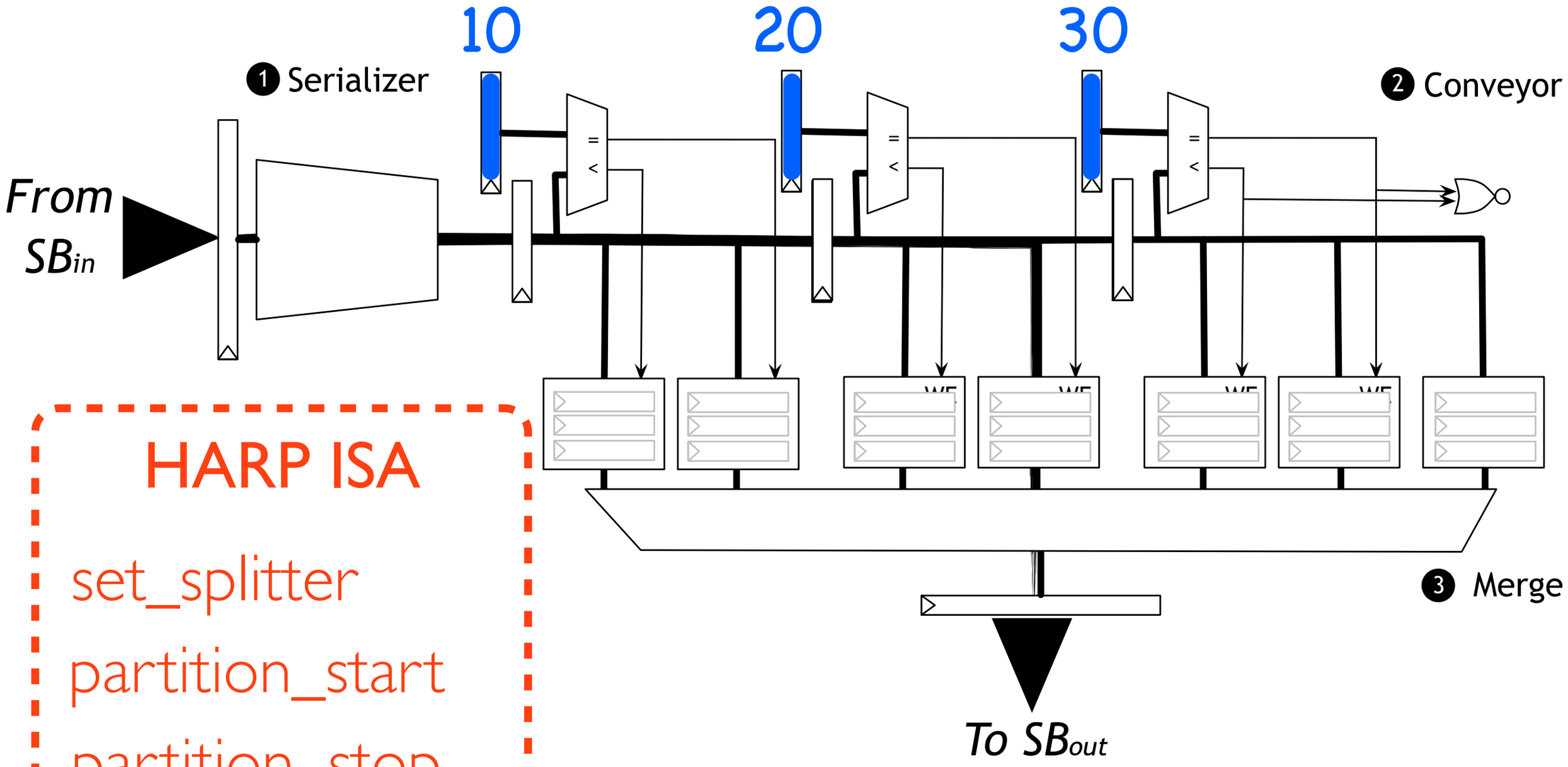

HARP ISA

set_splitter

**partition_start**

partition_stop

# Step 3: Serialize SBin Cachelines into Records



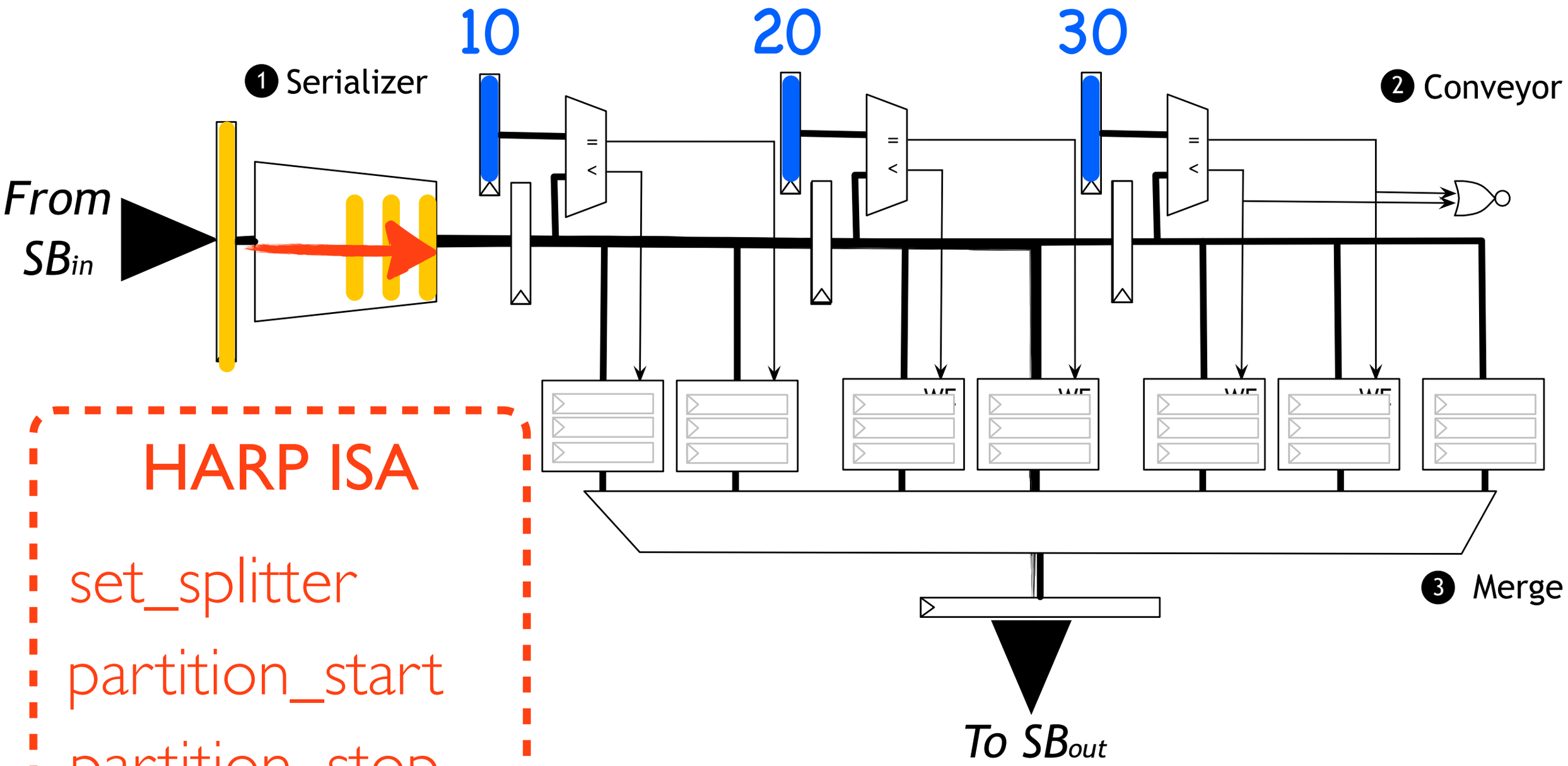

**HARP ISA**

set_splitter

partition_start

partition_stop

# Step 3: Serialize SBin Cachelines into Records



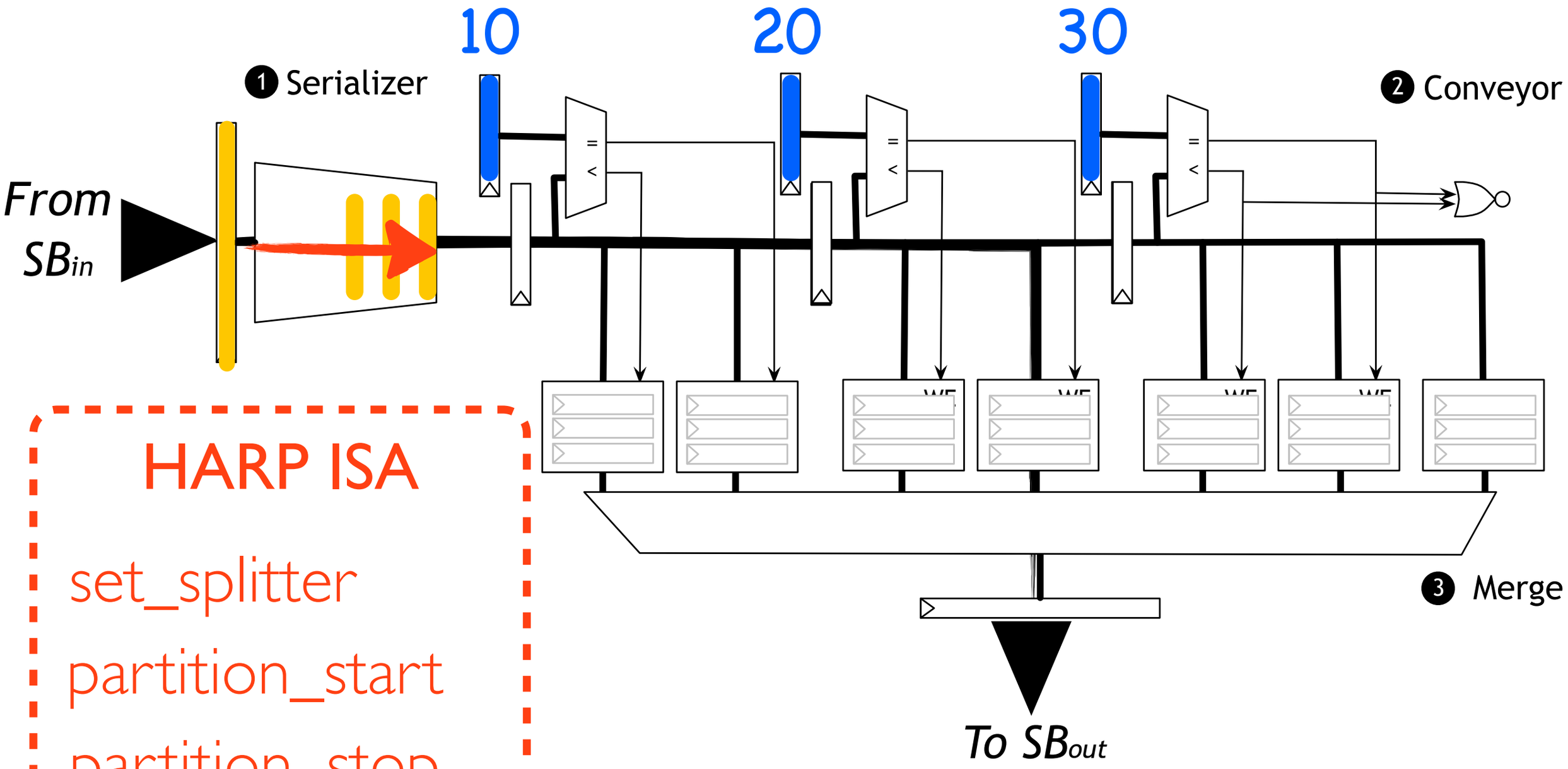

HARP ISA

set_splitter

partition_start

partition_stop

# Step 4: Comparator Conveyor



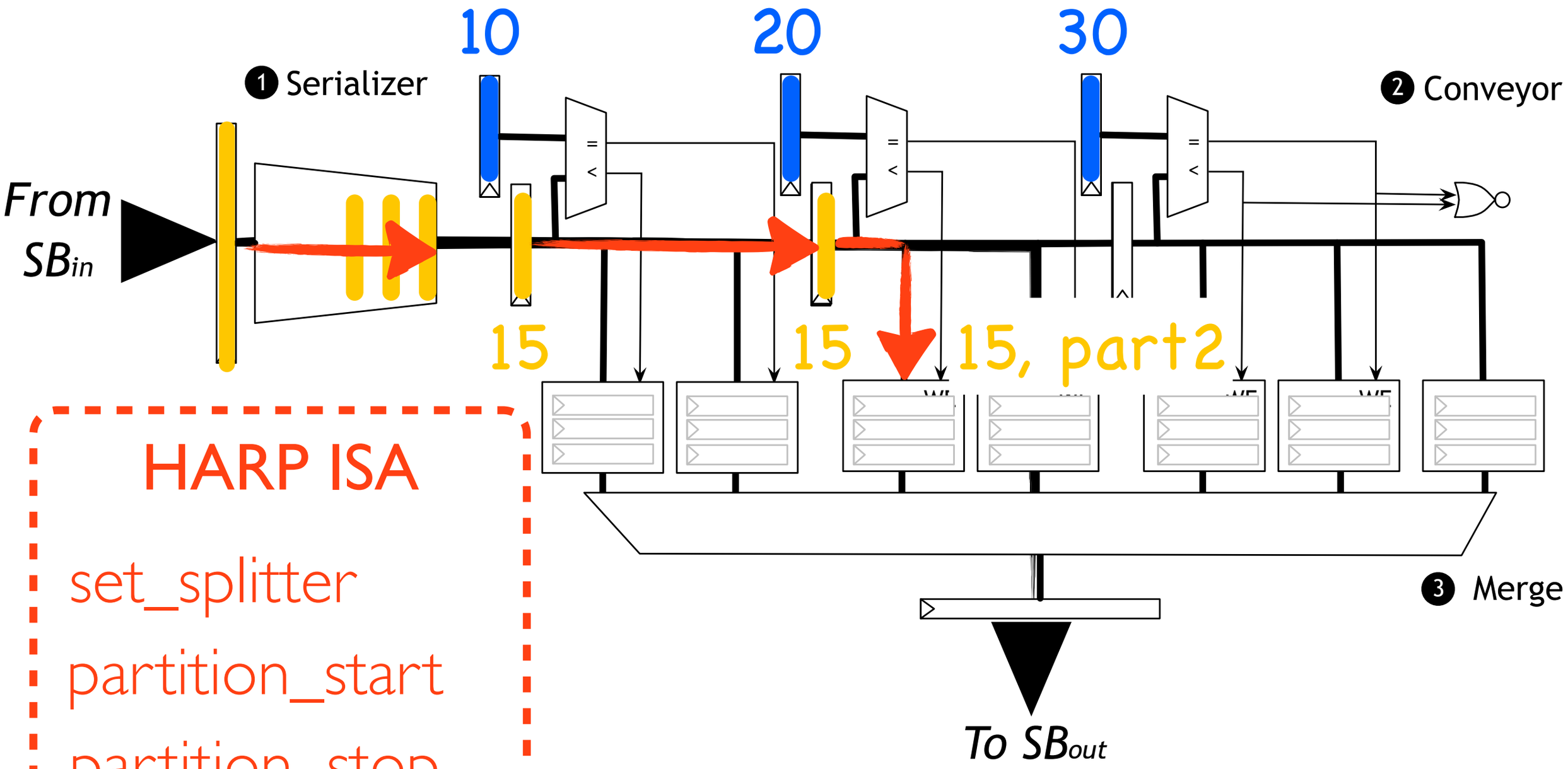

HARP ISA

set_splitter

partition_start

partition_stop

# Step 4: Comparator Conveyor



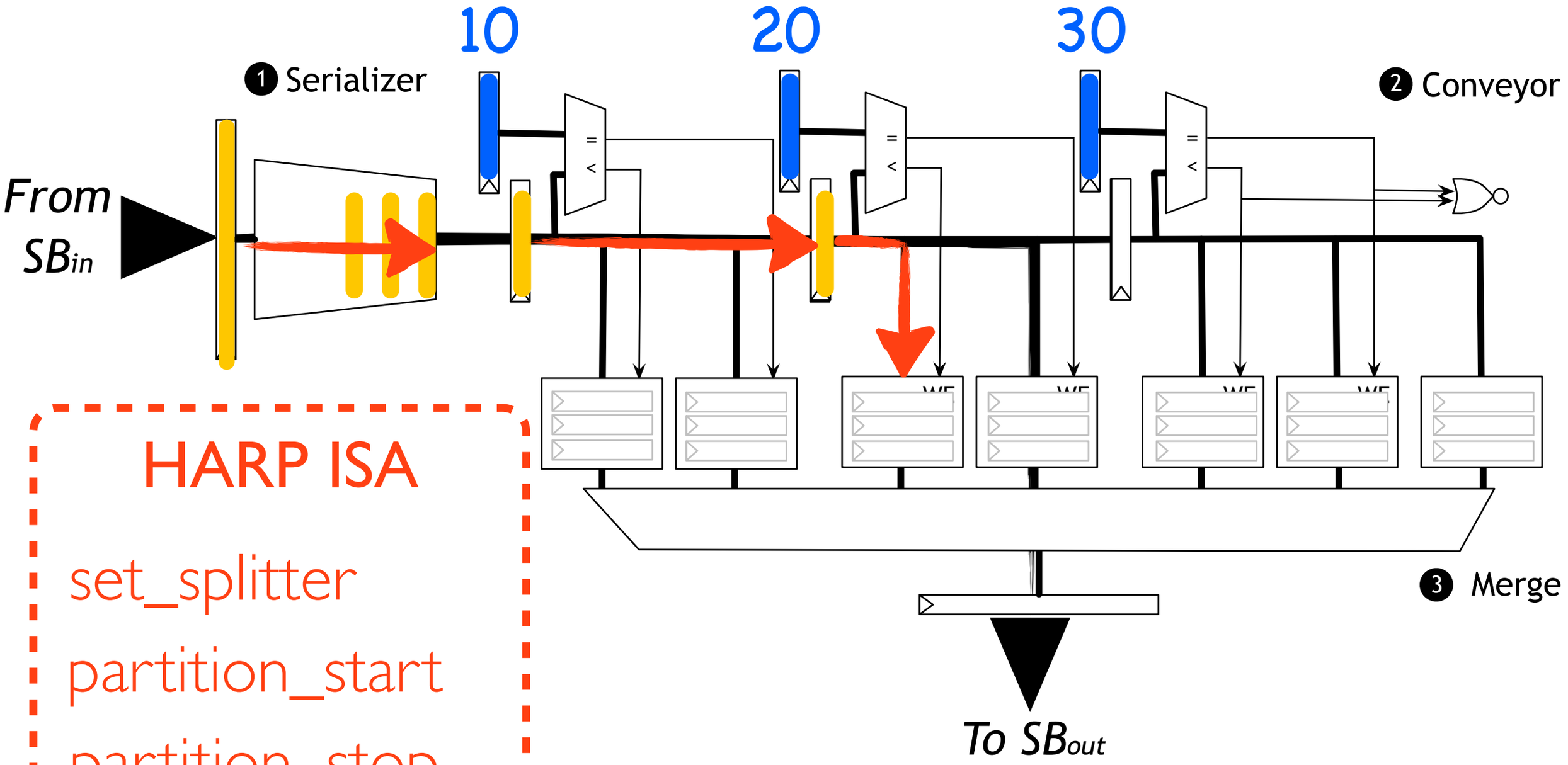

HARP ISA

set_splitter

partition_start

partition_stop

# Step 5: Merge Output Records to SBout



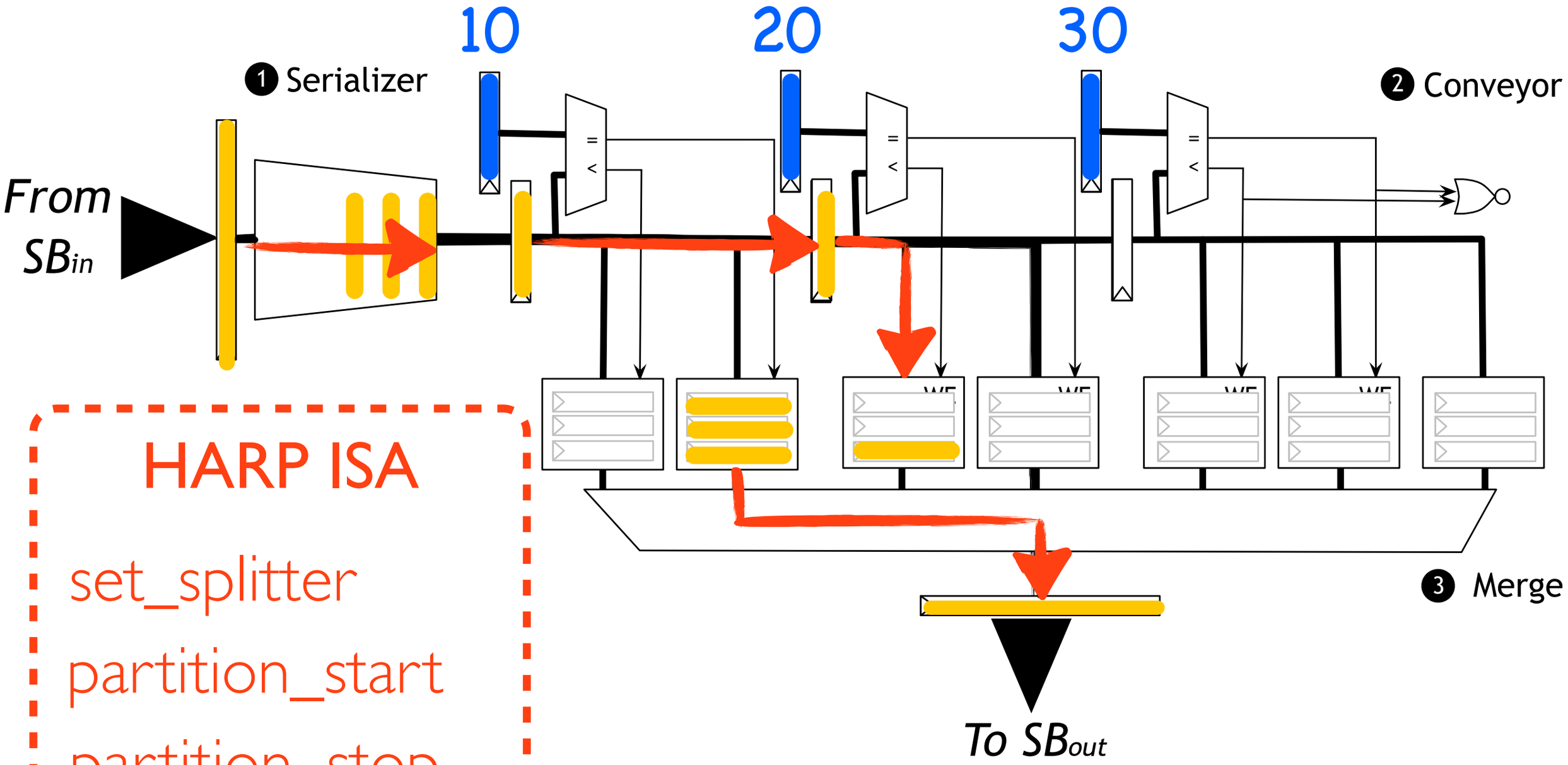

HARP ISA

set_splitter

partition_start

partition_stop

# Step 5: Merge Output Records to SBout



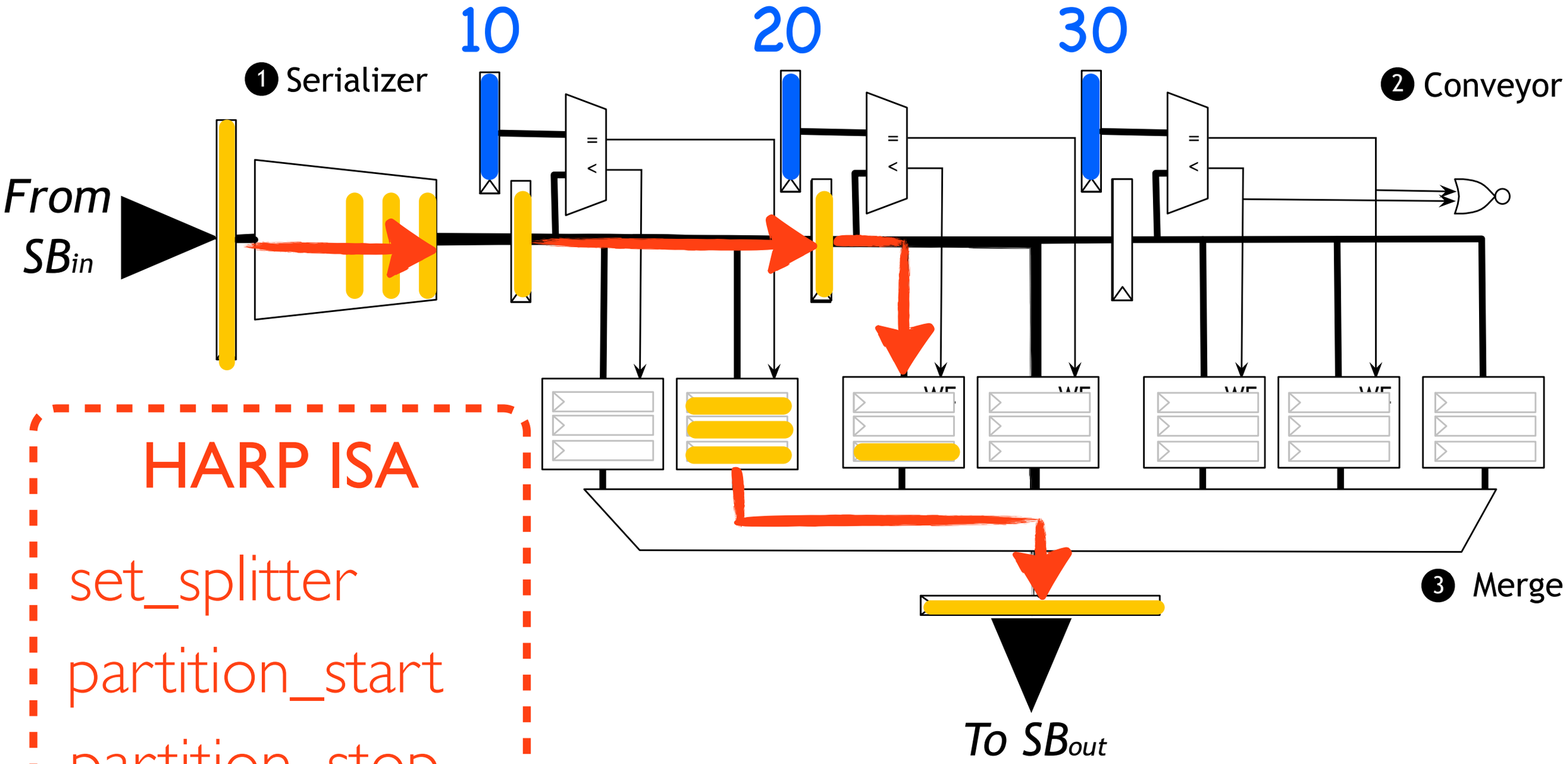

HARP ISA

set_splitter

partition_start

partition_stop

# Step 6: Drain In-Flight Records and Signal HARP to Stop Processing



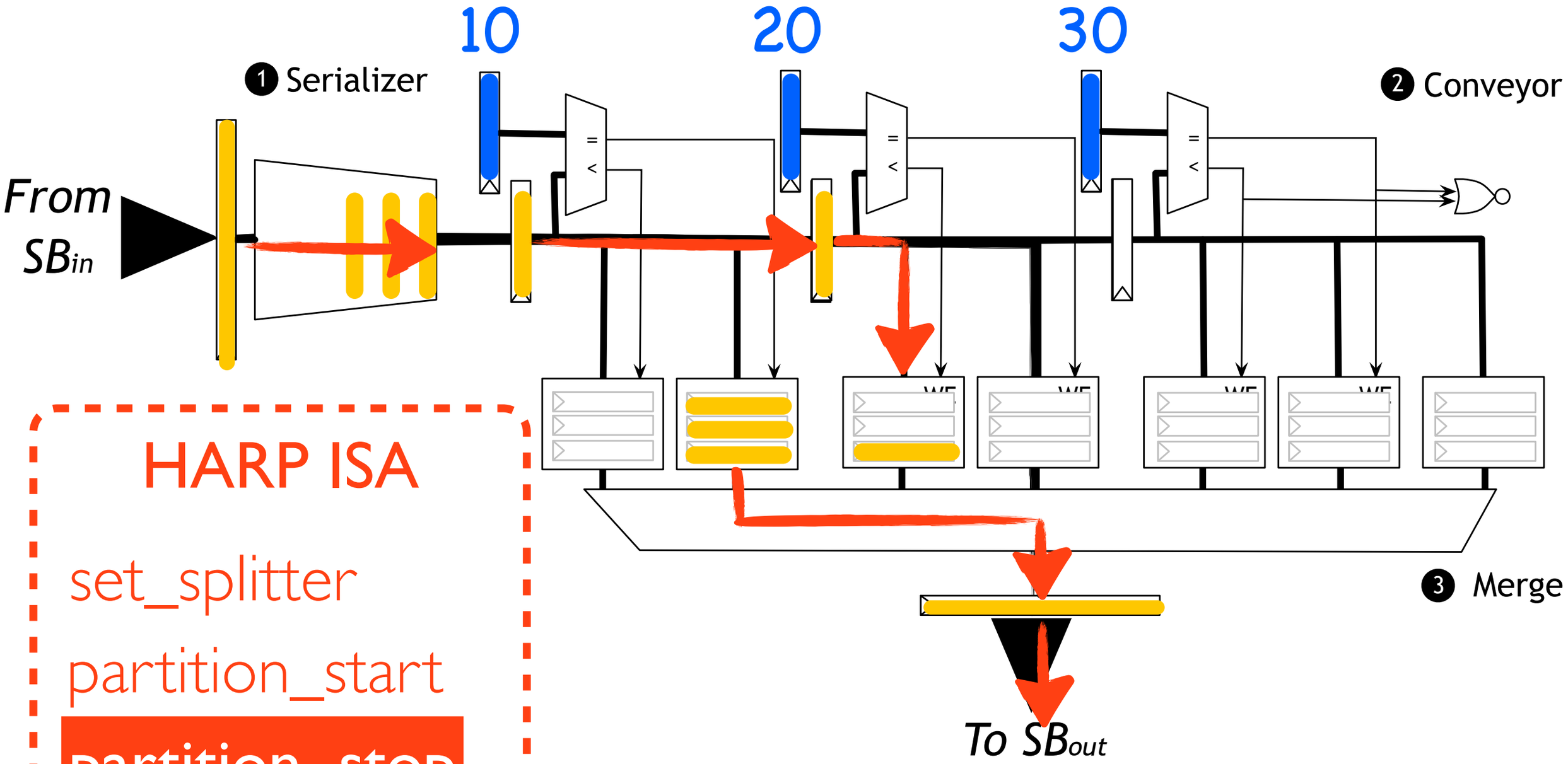

HARP ISA

set_splitter

partition_start

partition_stop

# Step 6: Drain In-Flight Records and Signal HARP to Stop Processing



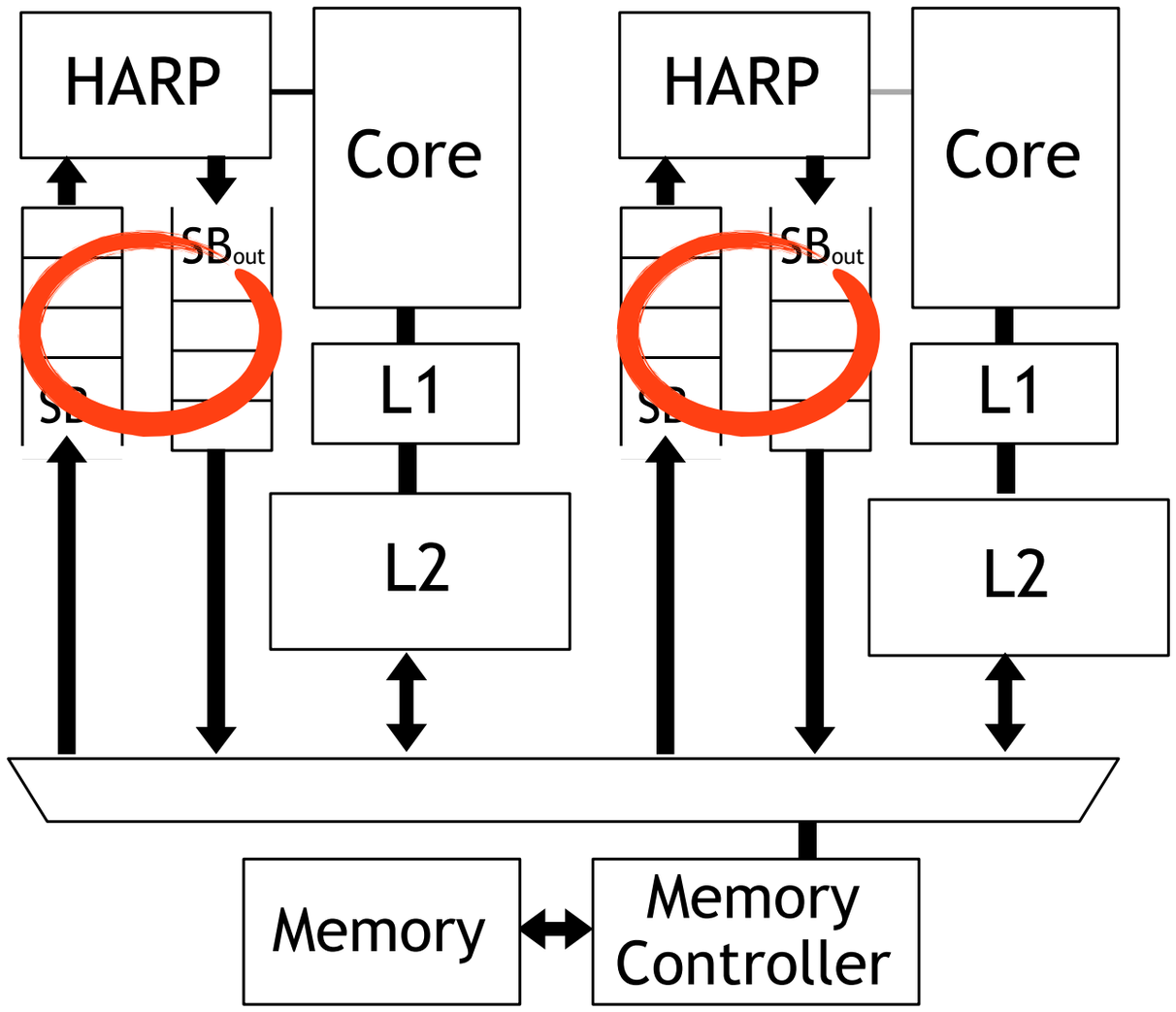

HARP ISA

set_splitter

partition_start

**partition_stop**

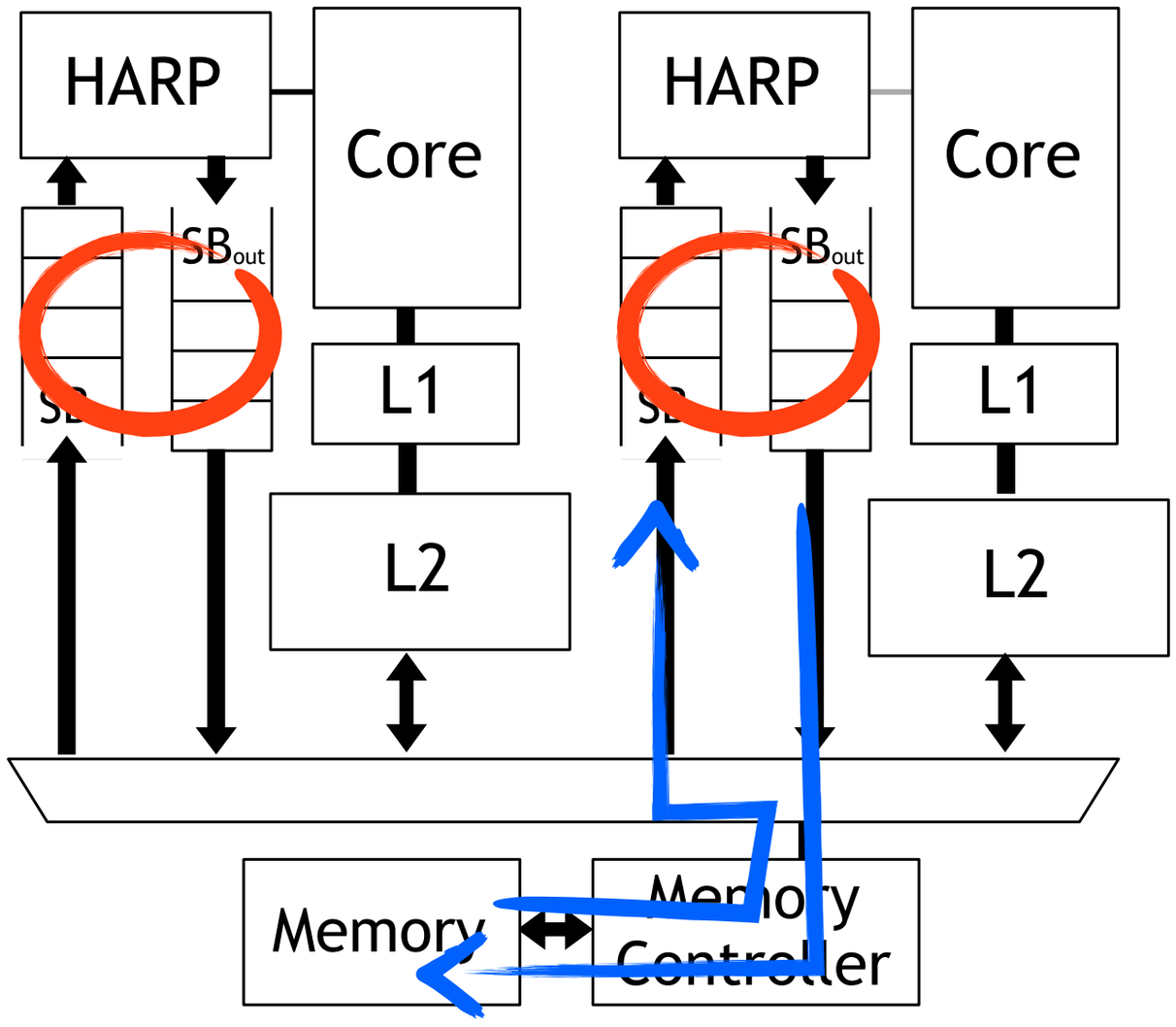
# Streaming Framework Architecture




## Inspired by Jouppi's work

Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In ISCA, 1990.

# Streaming Framework Architecture




Software-controlled data streaming in/out

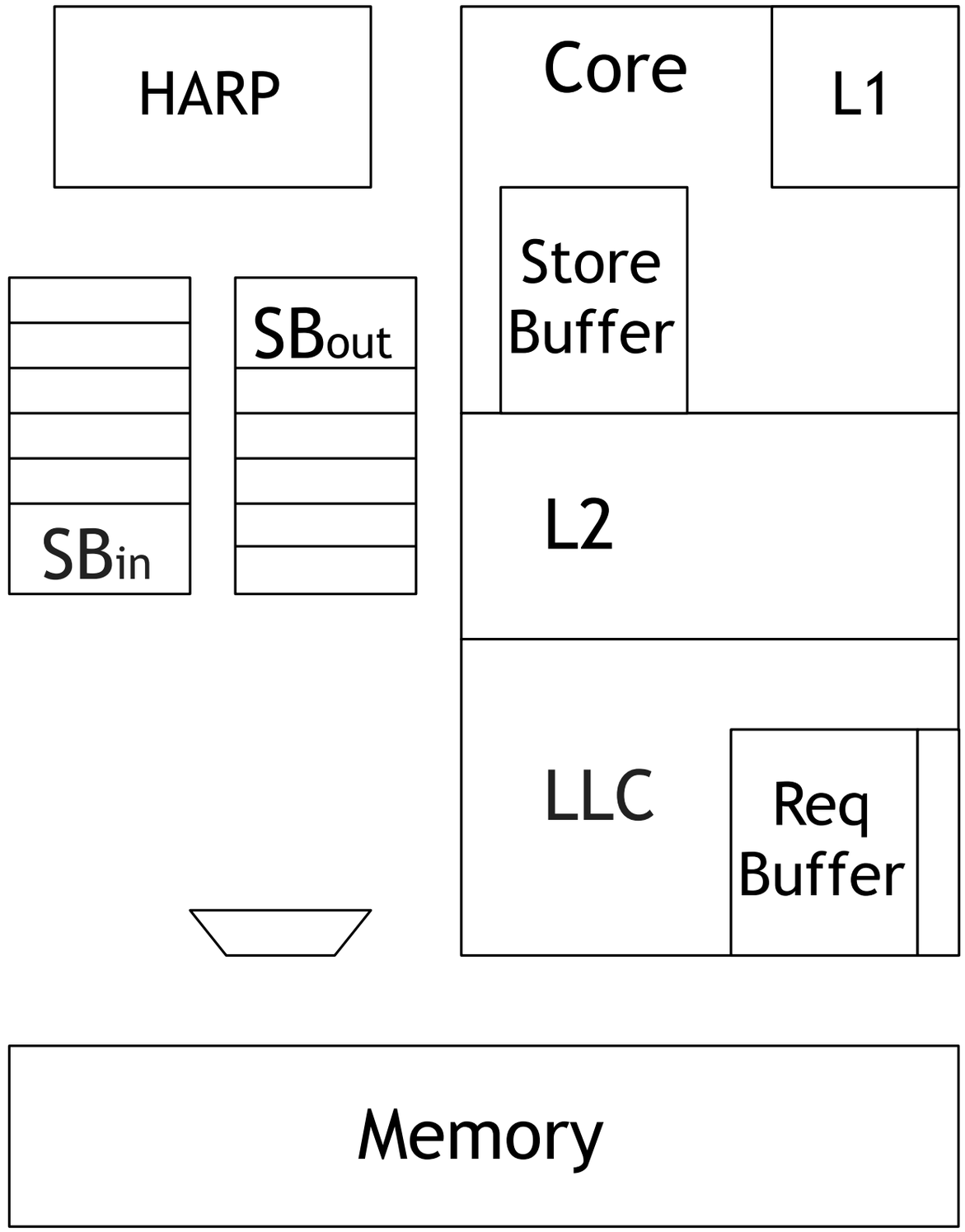
## Inspired by Jouppi's work

Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In ISCA, 1990.

# Step 1: Issue sbload from Core

SB ISA

- sbload
- sbstore
- sbsave
- sbrestore



# Step 1: Issue sbload from Core

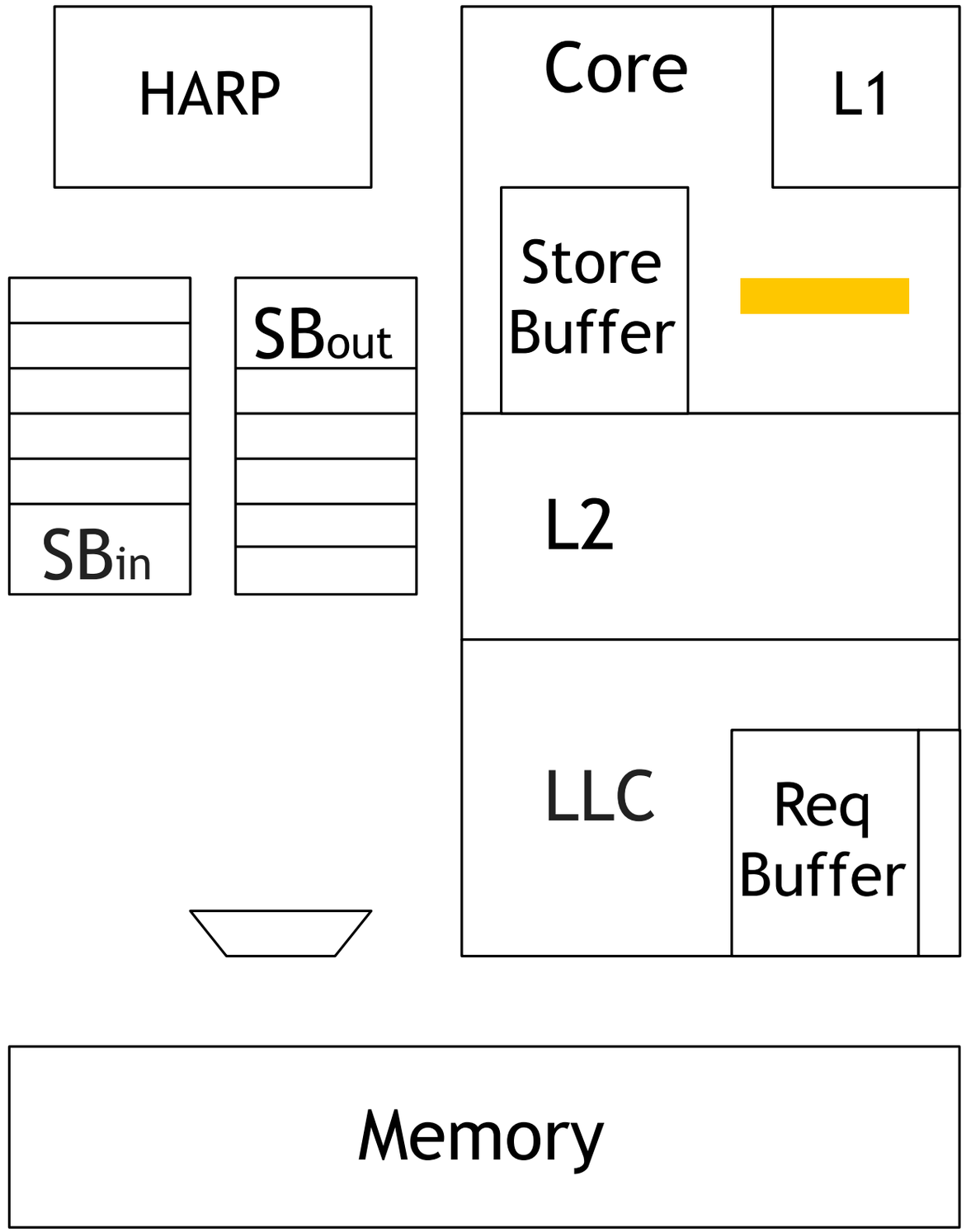
SB ISA

**sbload**

sbstore

sbsave

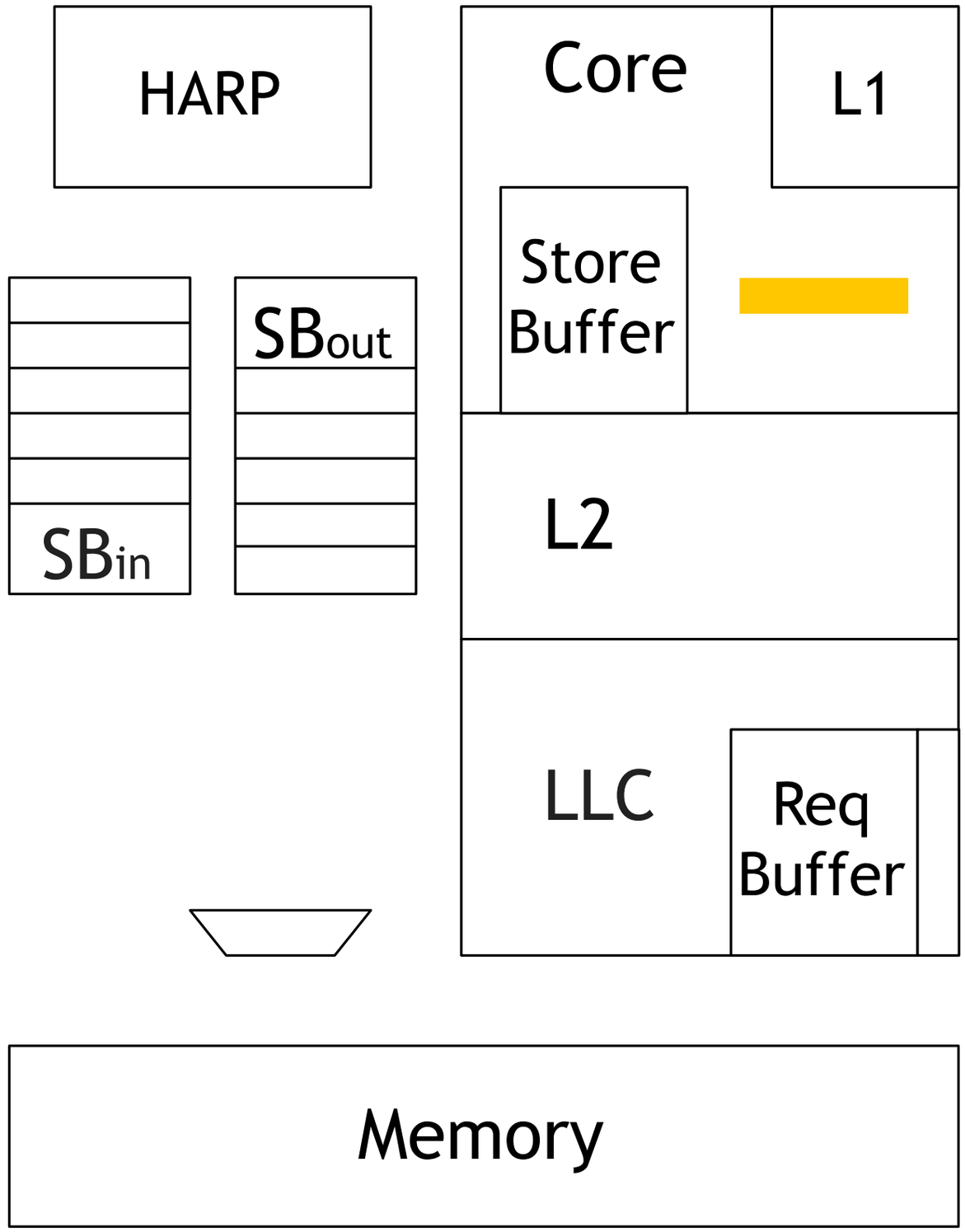
sbrestore



# Step 2: Send sbload from Req Buffer to Memory

**SB ISA**

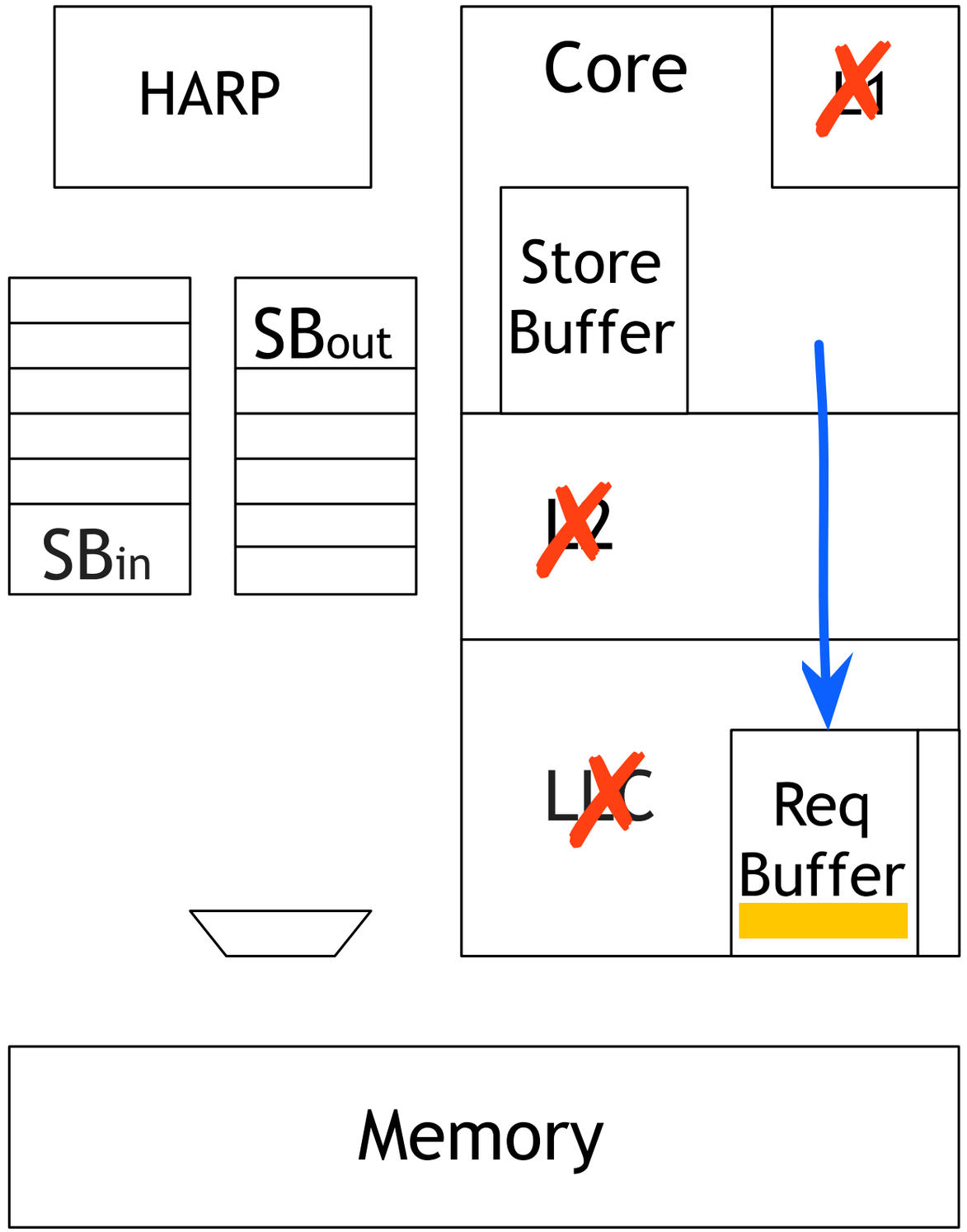
- sbload
- sbstore
- sbsave
- sbrestore



# Step 2: Send sbload from Req Buffer to Memory

SB ISA

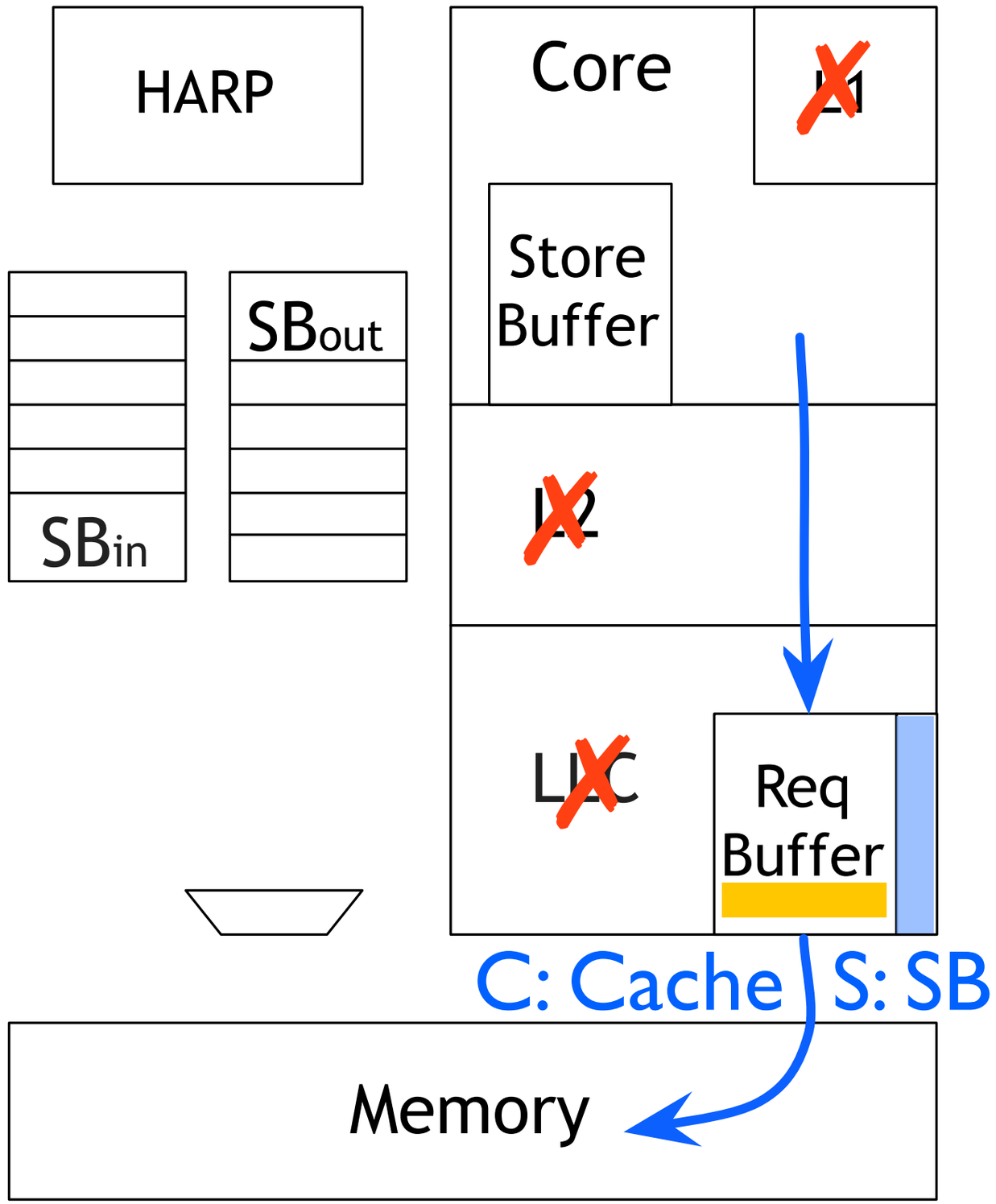
- sbload
- sbstore
- sbsave
- sbrestore



# Step 2: Send sbload from Req Buffer to Memory

SB ISA

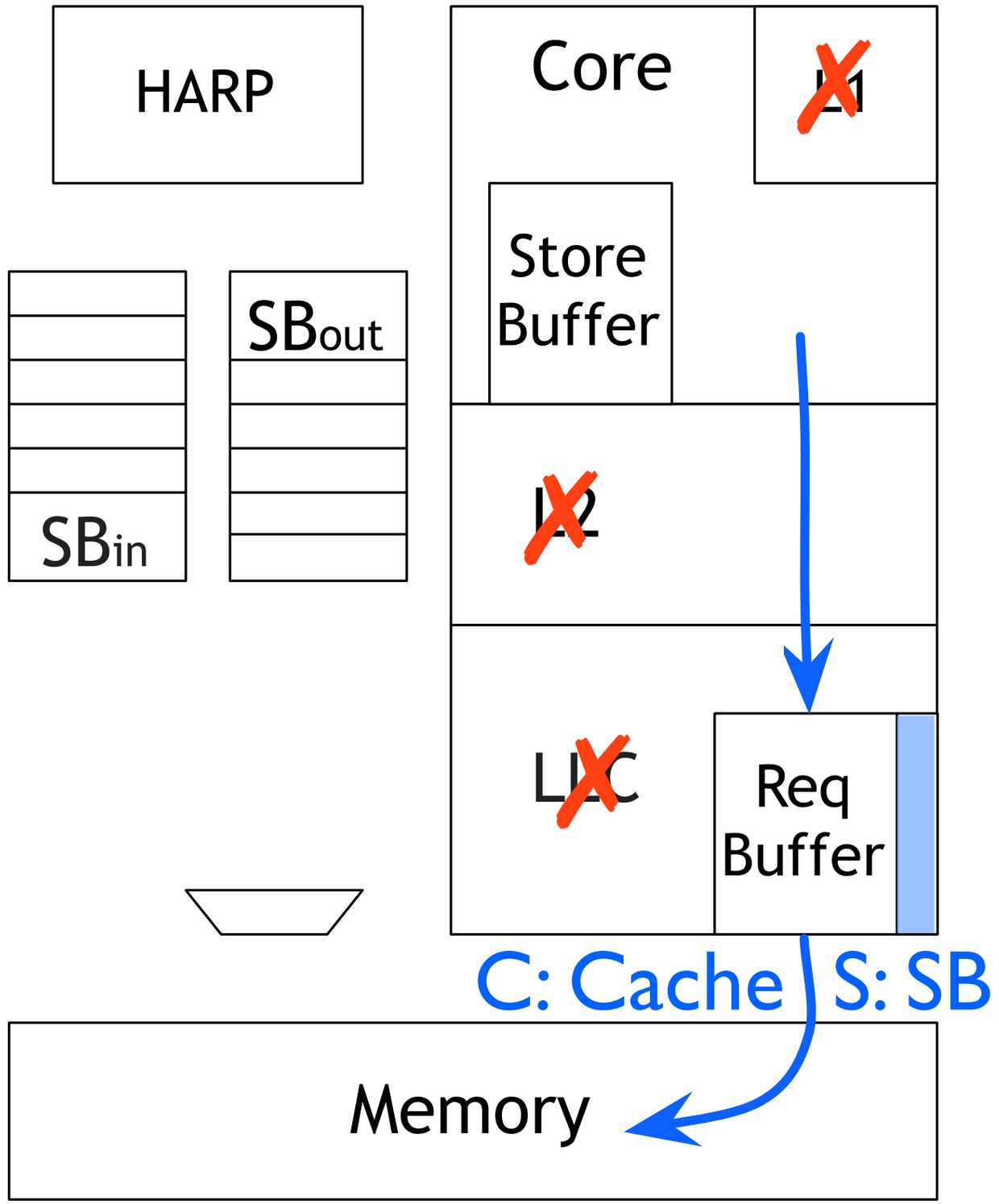
- sbload
- sbstore
- sbsave
- sbrestore



# Step 3: Data Return from Memory to SBin

SB ISA

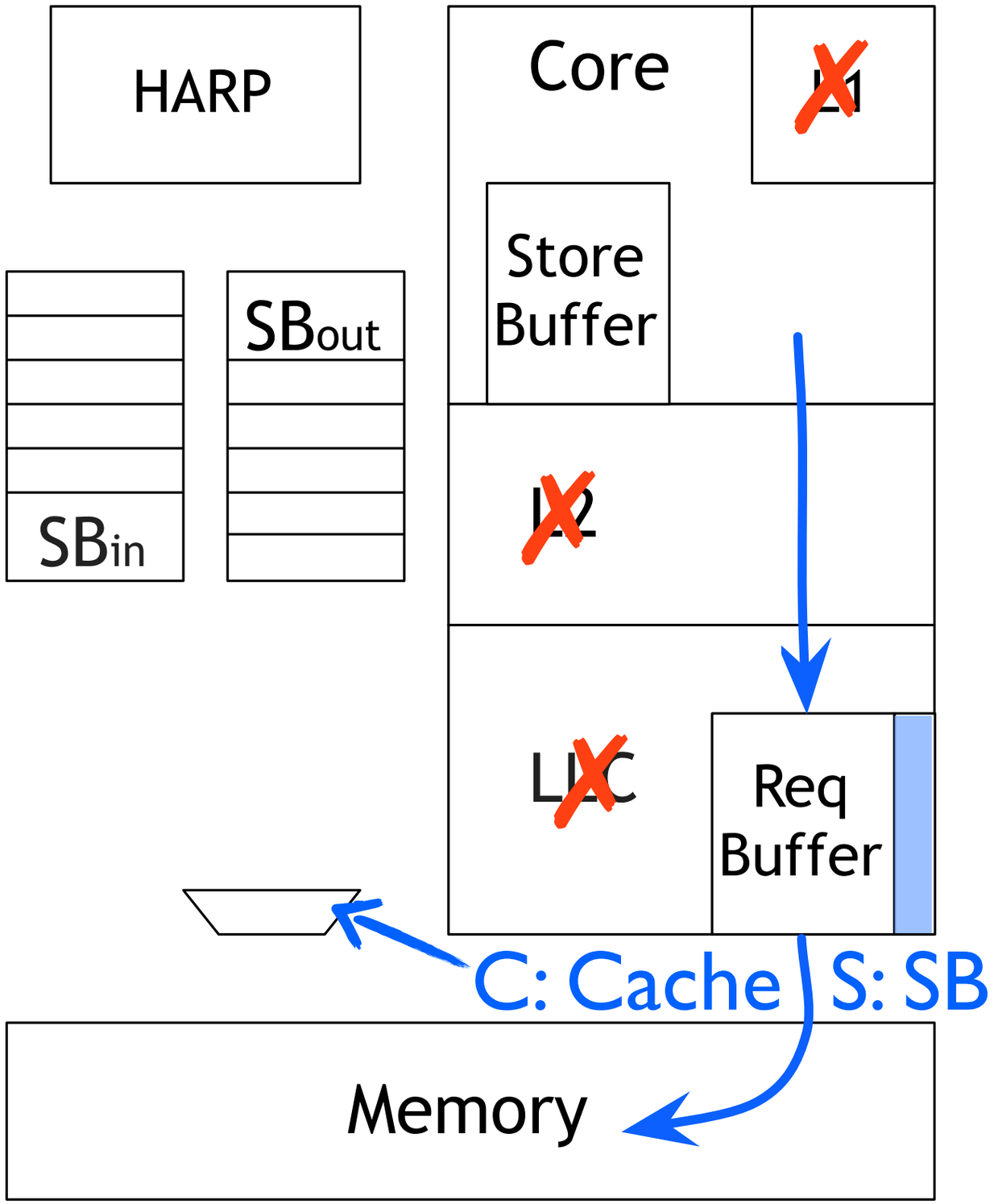
- sbload
- sbstore
- sbsave
- sbrestore



# Step 3: Data Return from Memory to SBin

SB ISA

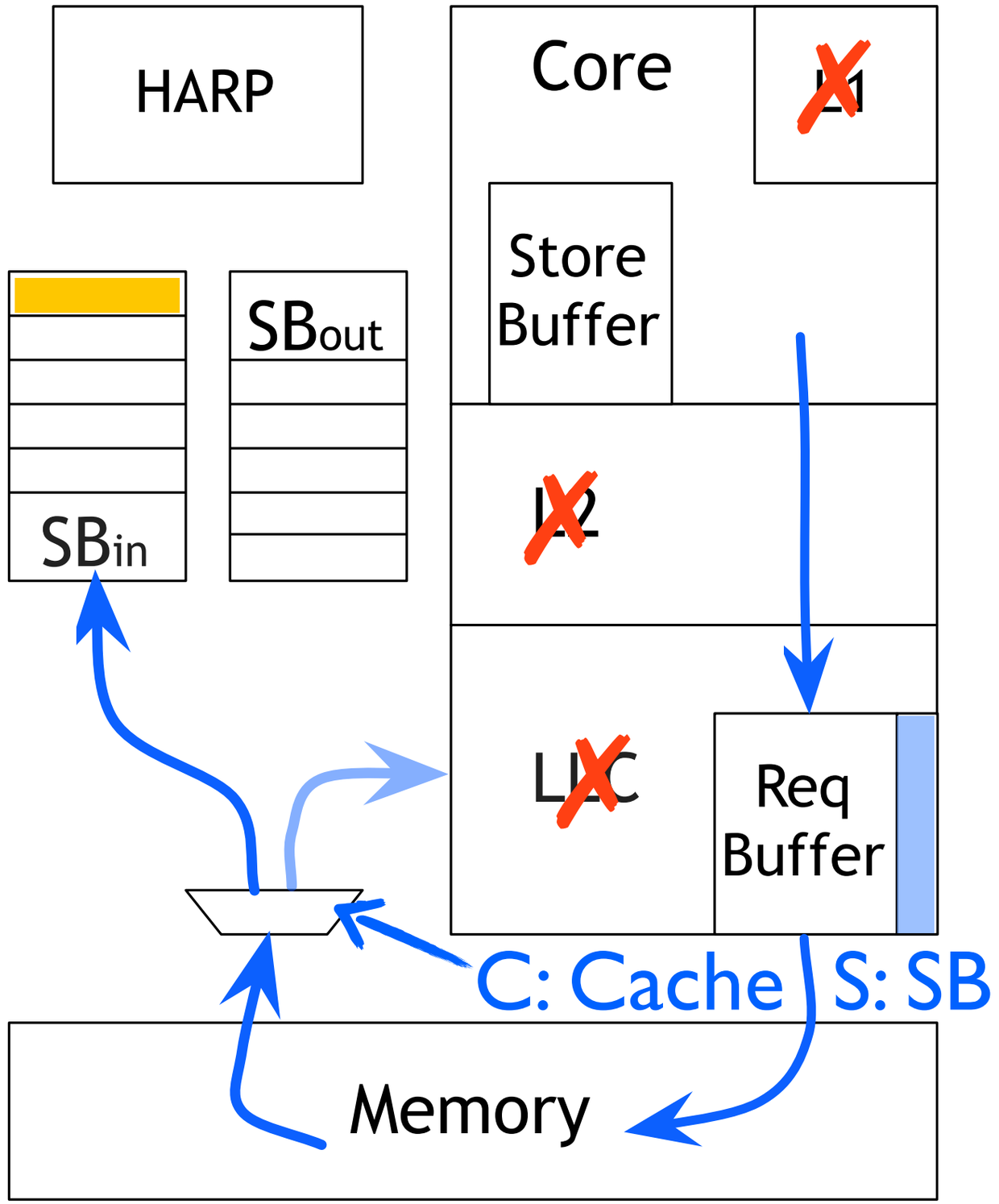
- sbload
- sbstore
- sbsave
- sbrestore



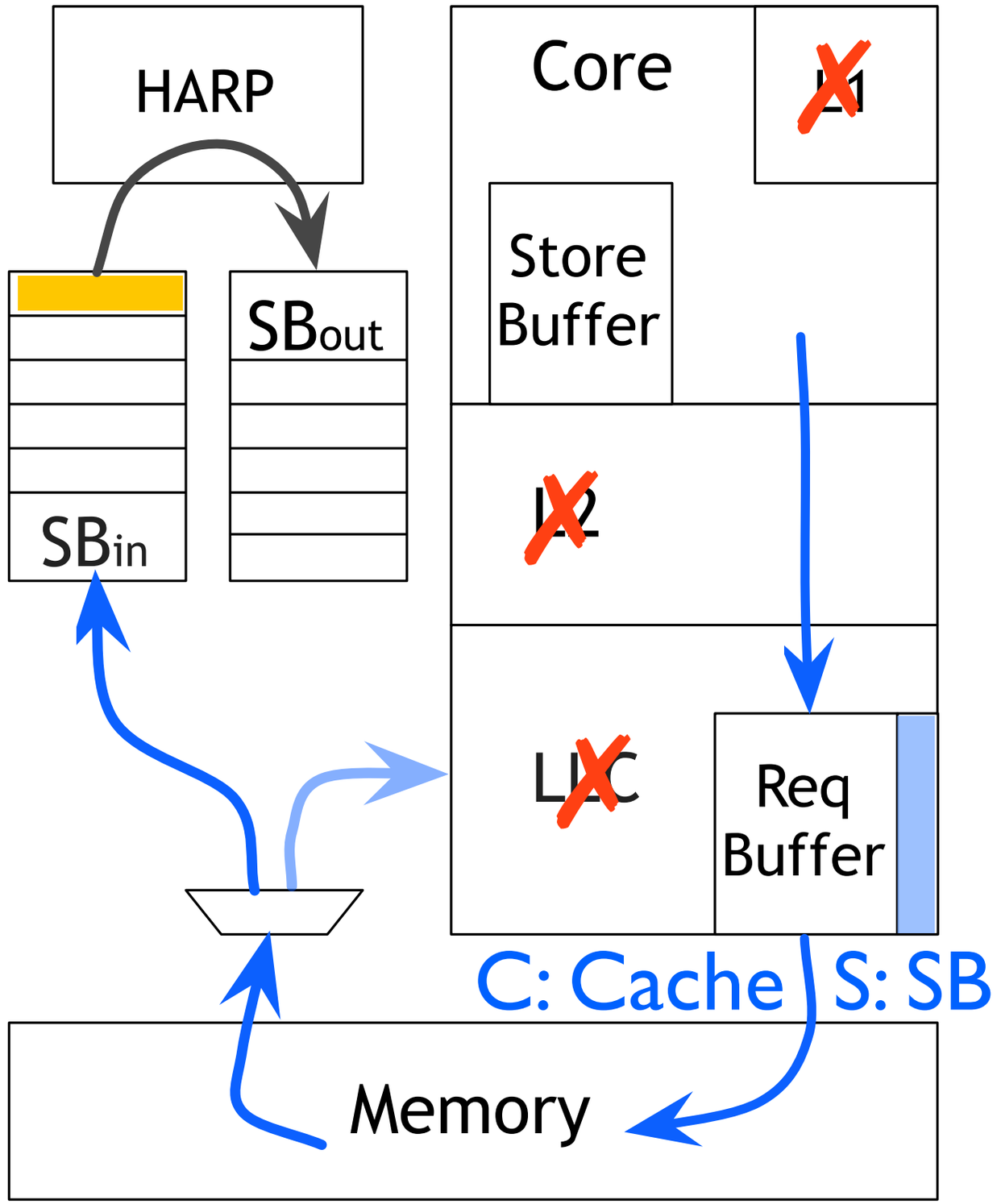
# Step 3: Data Return from Memory to SBin

SB ISA

- sbload
- sbstore
- sbsave
- sbrestore



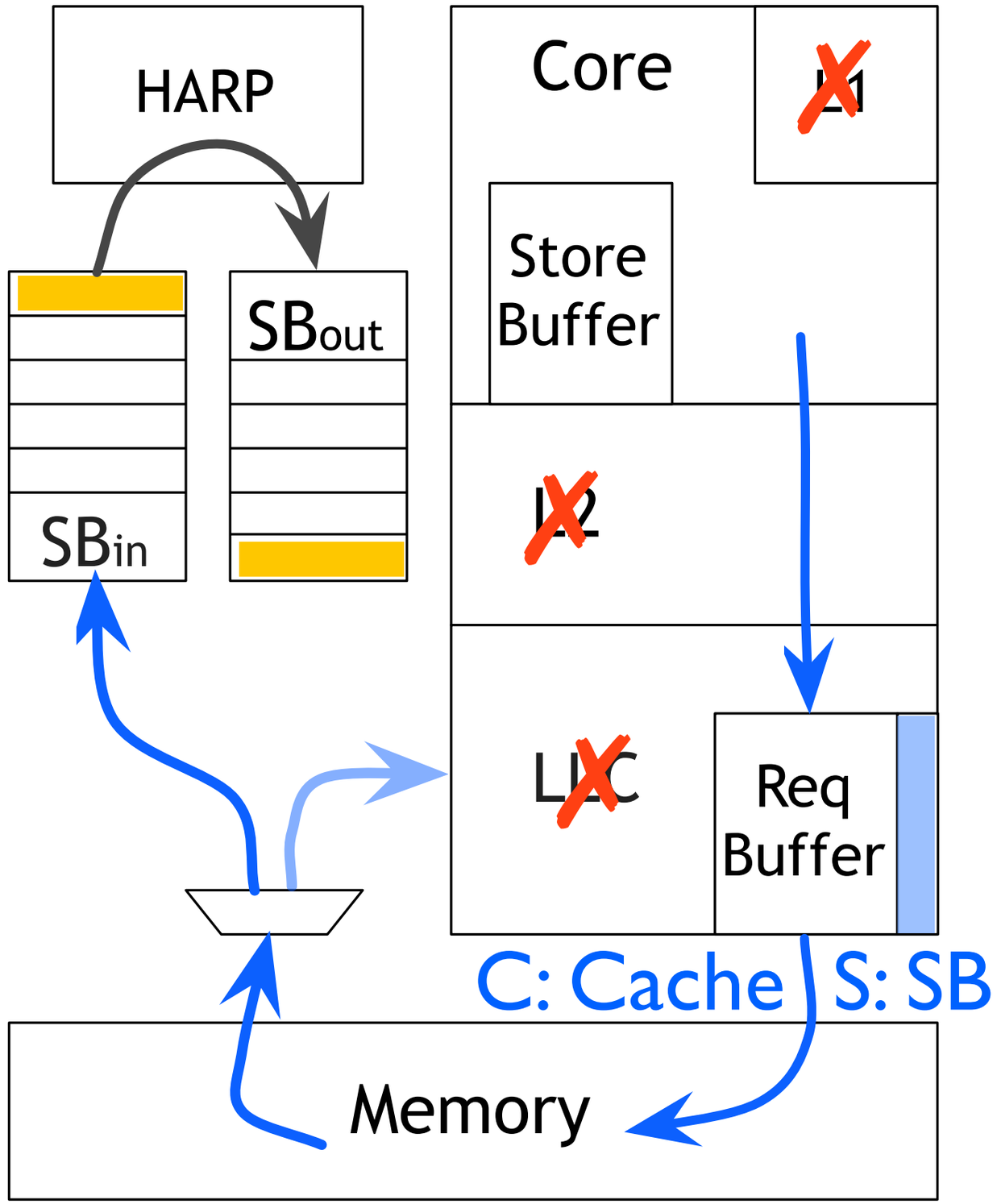
# Step 4: HARP Pulls Data from SBin and Pushes Data to SBout




**SB ISA**

- sbload
- sbstore
- sbsave
- sbrestore

# Step 4: HARP Pulls Data from SBin and Pushes Data to SBout



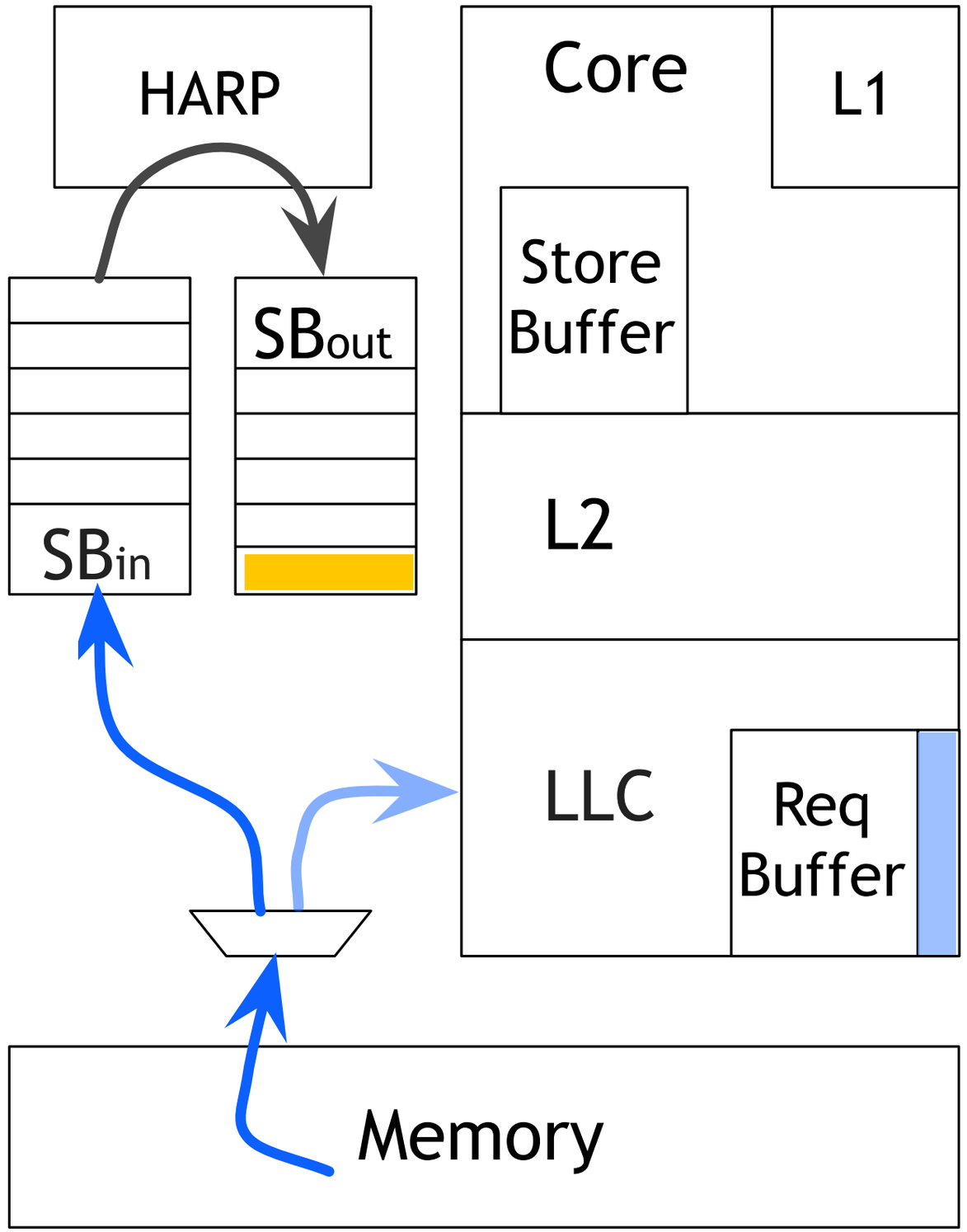

**SB ISA**

- sbload
- sbstore
- sbsave
- sbrestore

# Step 5: Issue sbstore from Core

SB ISA

- sbload
- sbstore
- sbsave
- sbrestore



# Step 5: Issue sbstore from Core

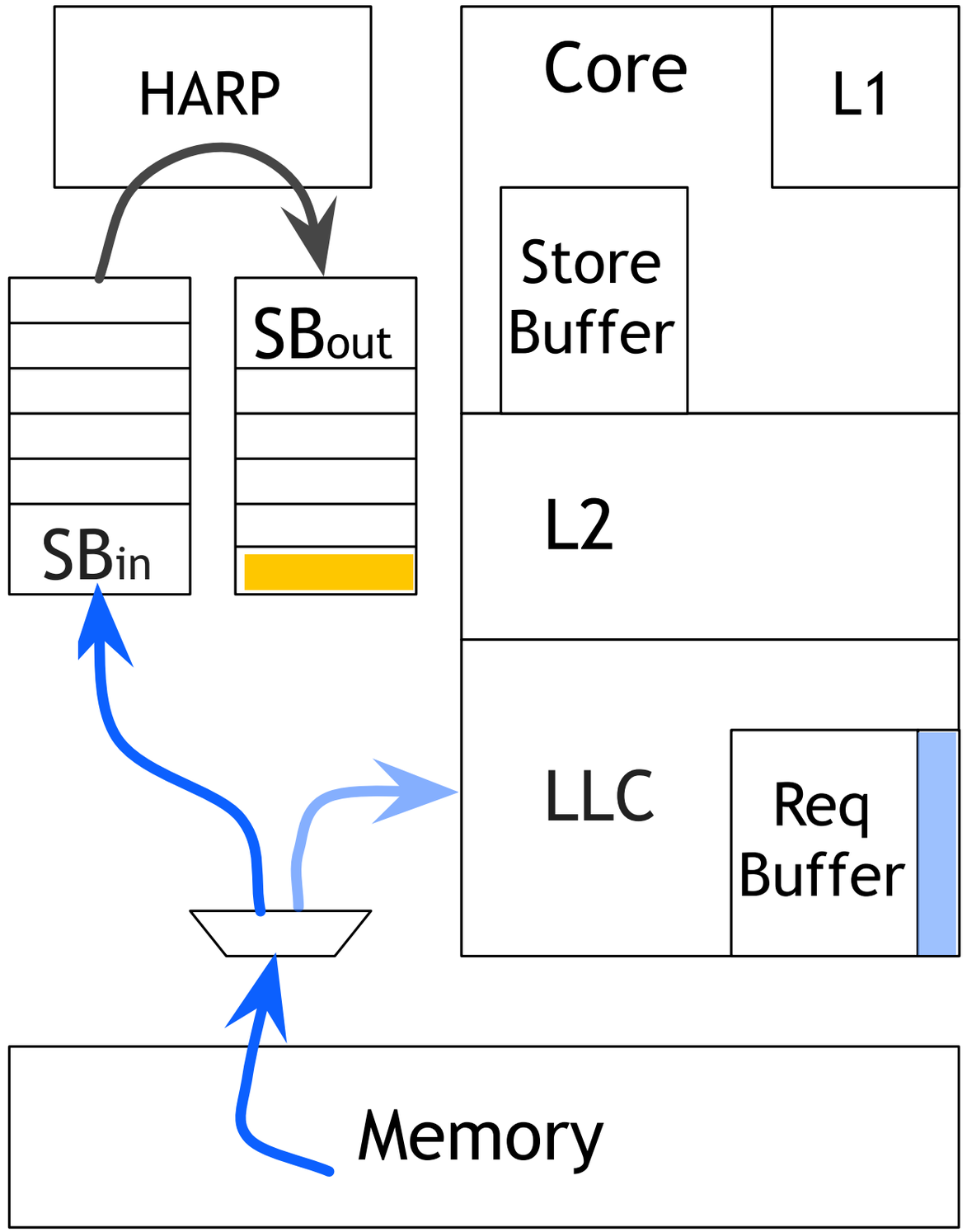
SB ISA

sbload

**sbstore**

sbsave

sbrestore



# Step 6: Data Copied from head of SBout to Store Buffer

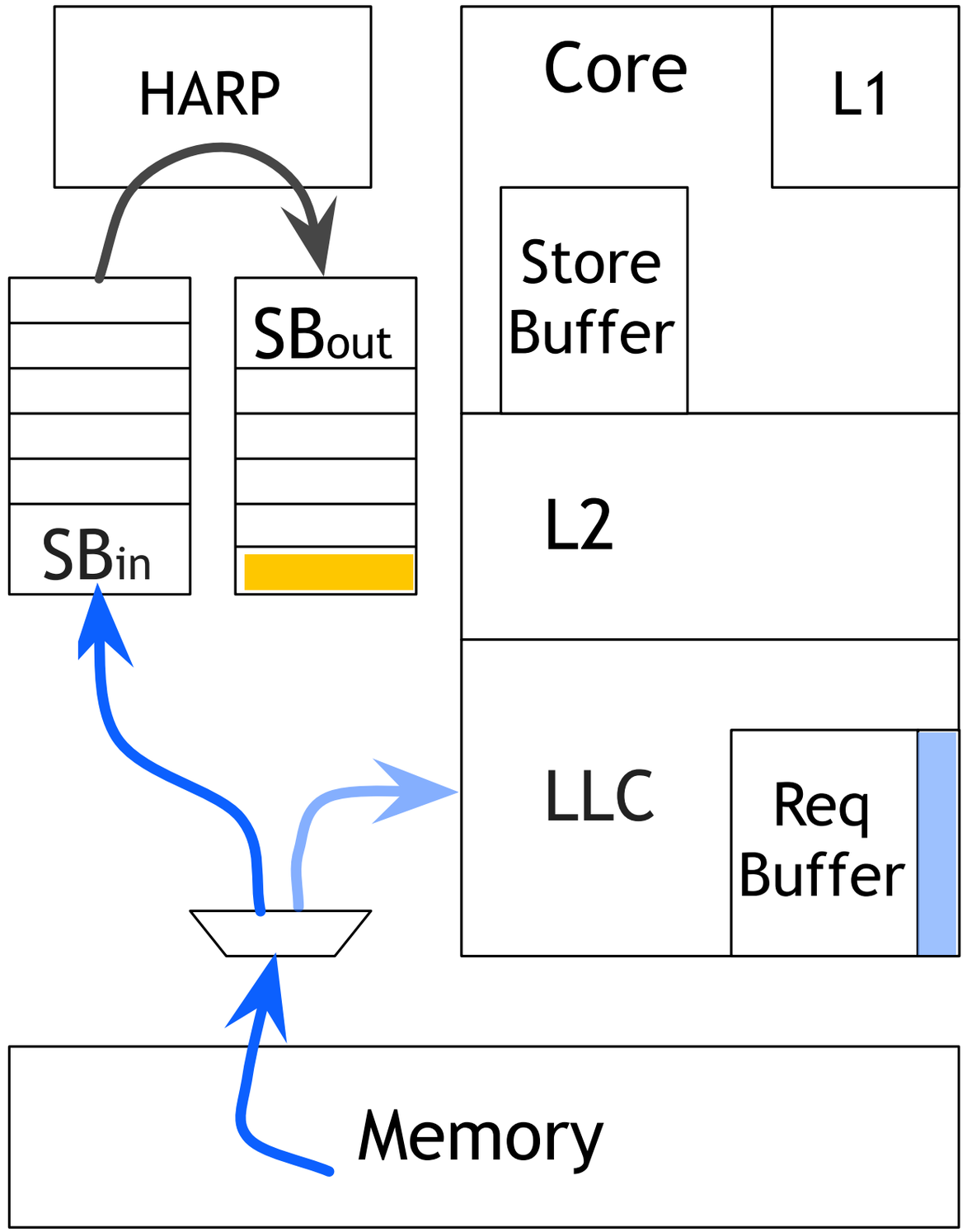
SB ISA

sbload

sbstore

sbsave

sbrestore



# Step 6: Data Copied from head of SBout to Store Buffer

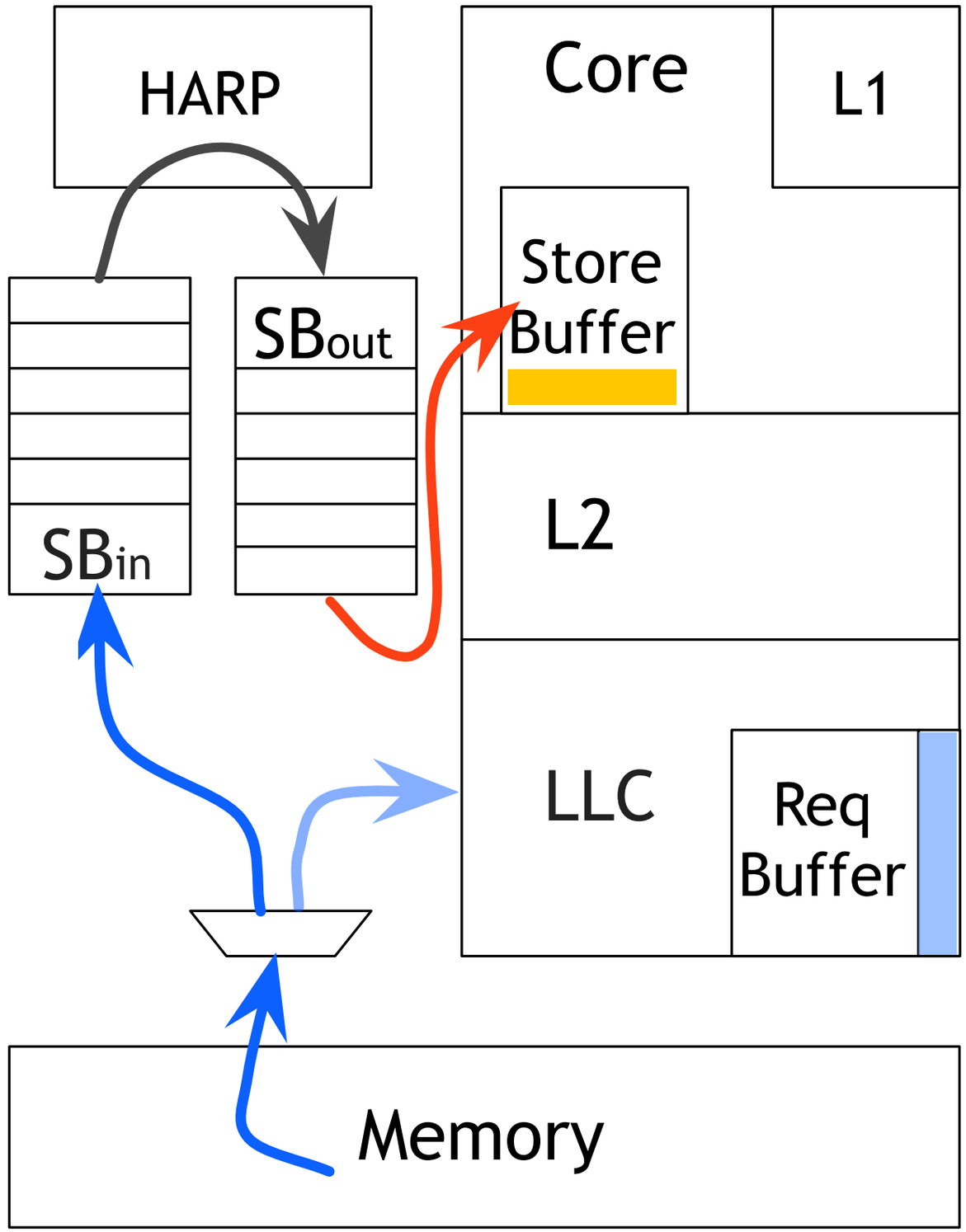
SB ISA

sbload

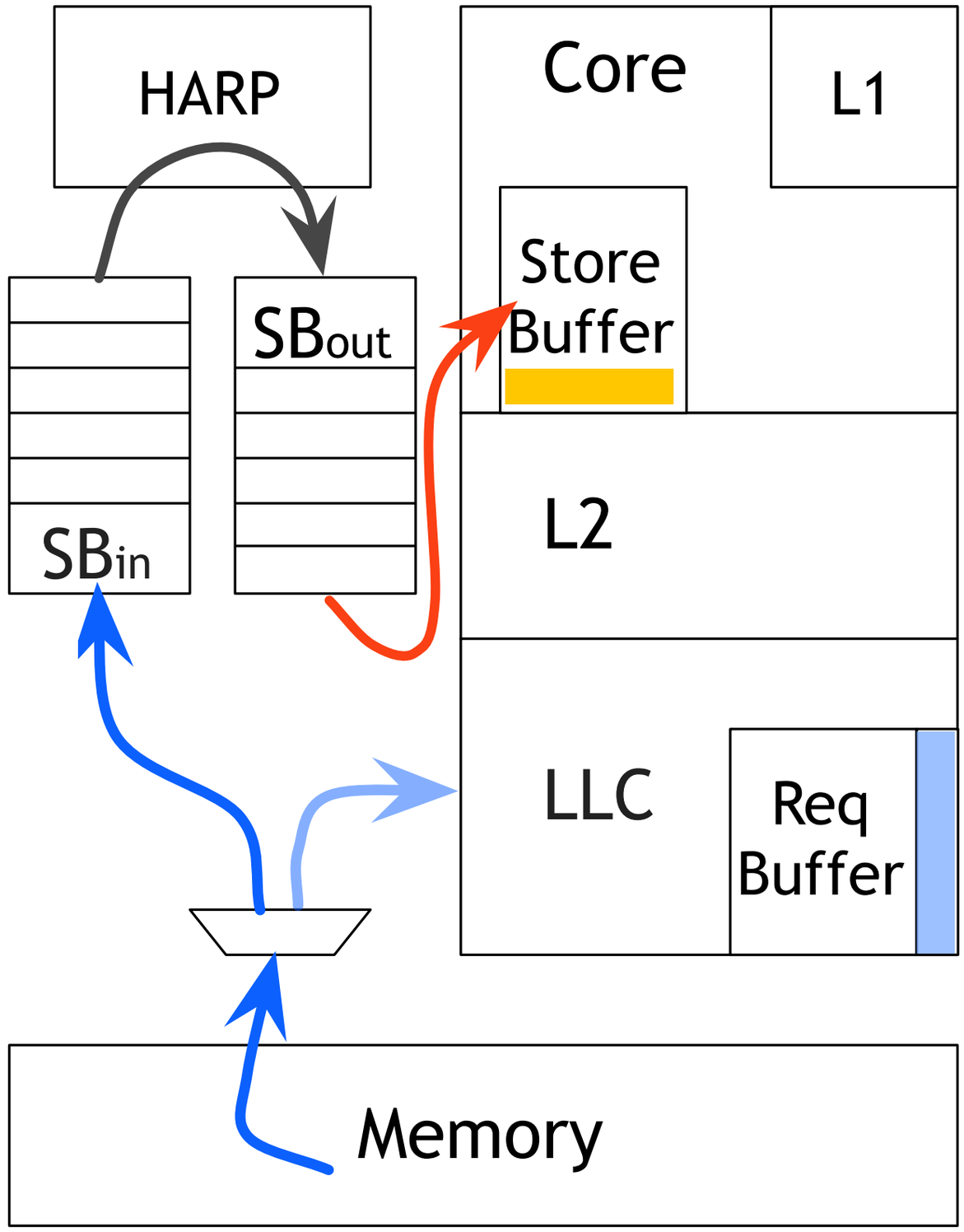
sbstore

sbsave

sbrestore



# Step 7: Data Written Back to Memory via Existing Store Datapath



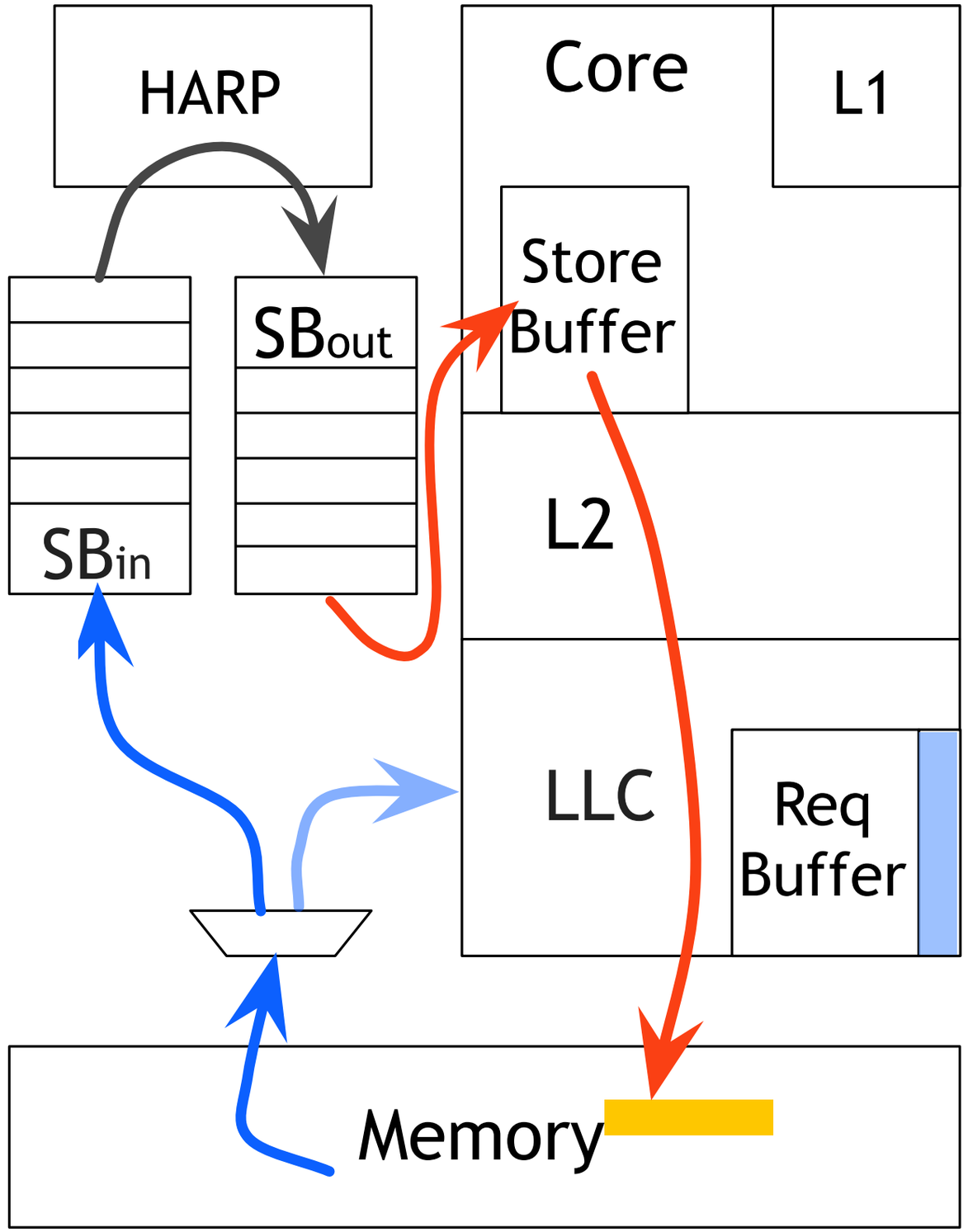

**SB ISA**

- sbload
- sbstore
- sbsave
- sbrestore

# Step 7: Data Written Back to Memory via Existing Store Datapath

**SB ISA**

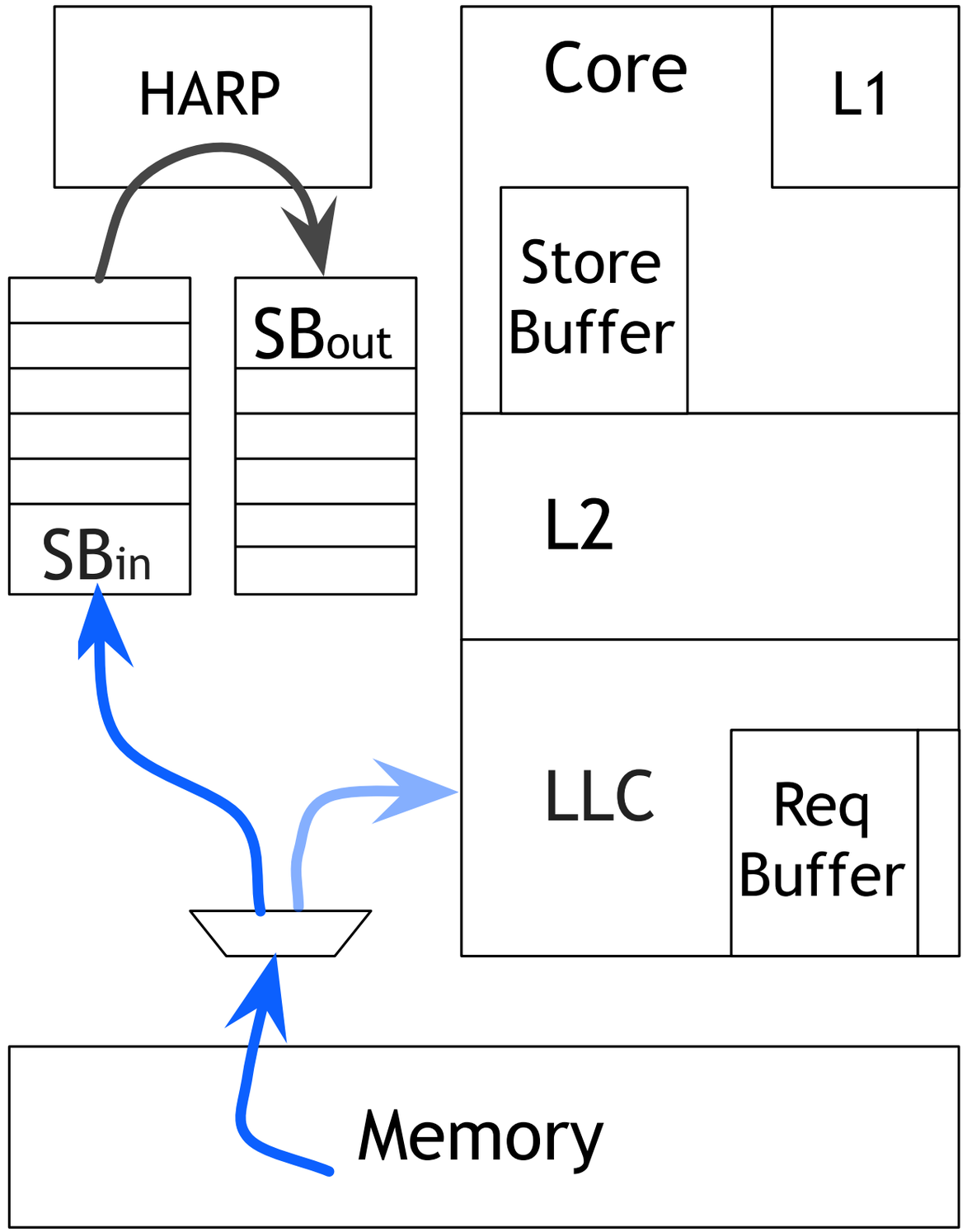
- sbload
- sbstore
- sbsave
- sbrestore



# Interrupts and Context Switches

**SB ISA**

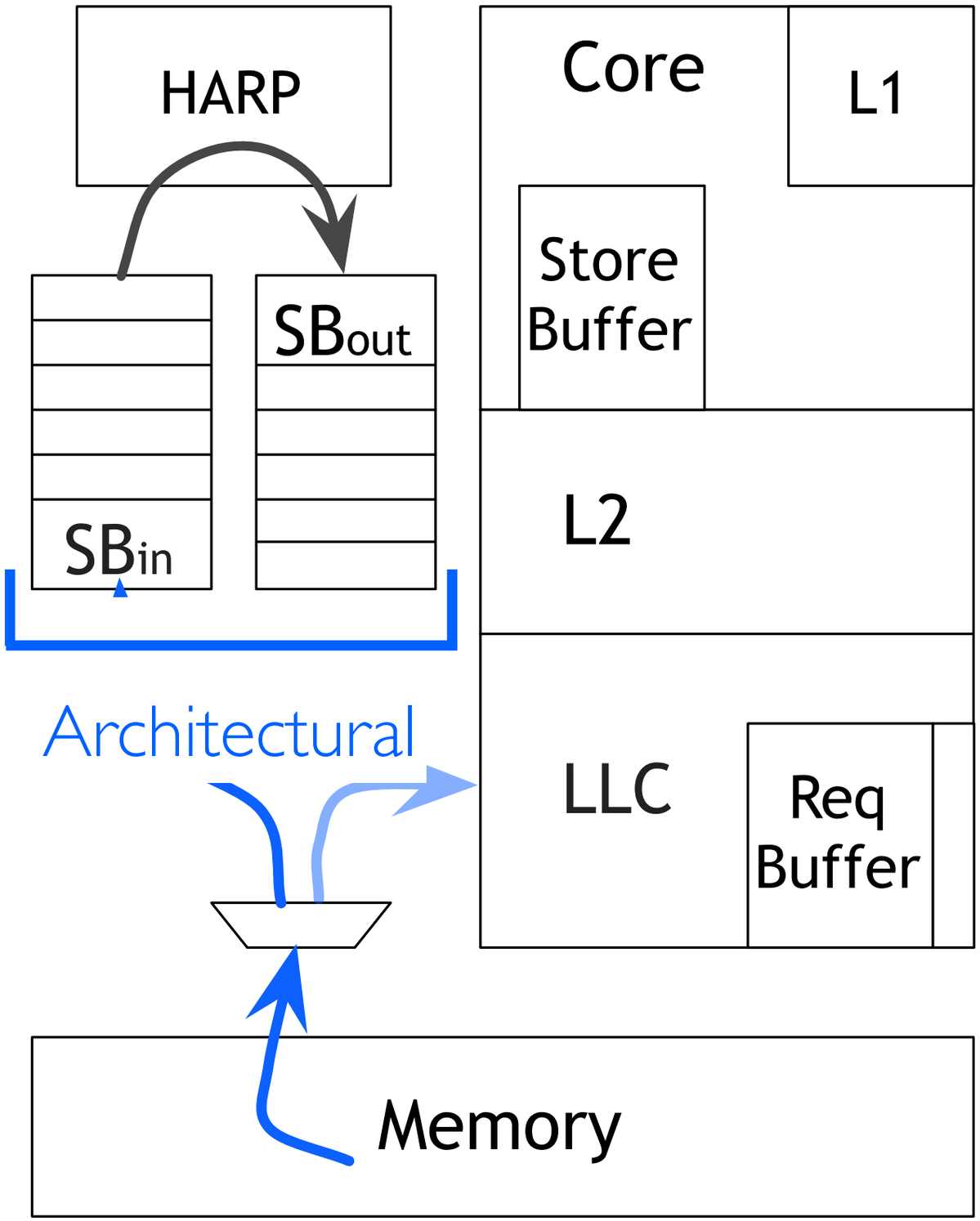
- sbload
- sbstore
- sbsave
- sbrestore



# Interrupts and Context Switches

**SB ISA**

- sbload
- sbstore
- sbsave
- sbrestore



# Interrupts and Context Switches

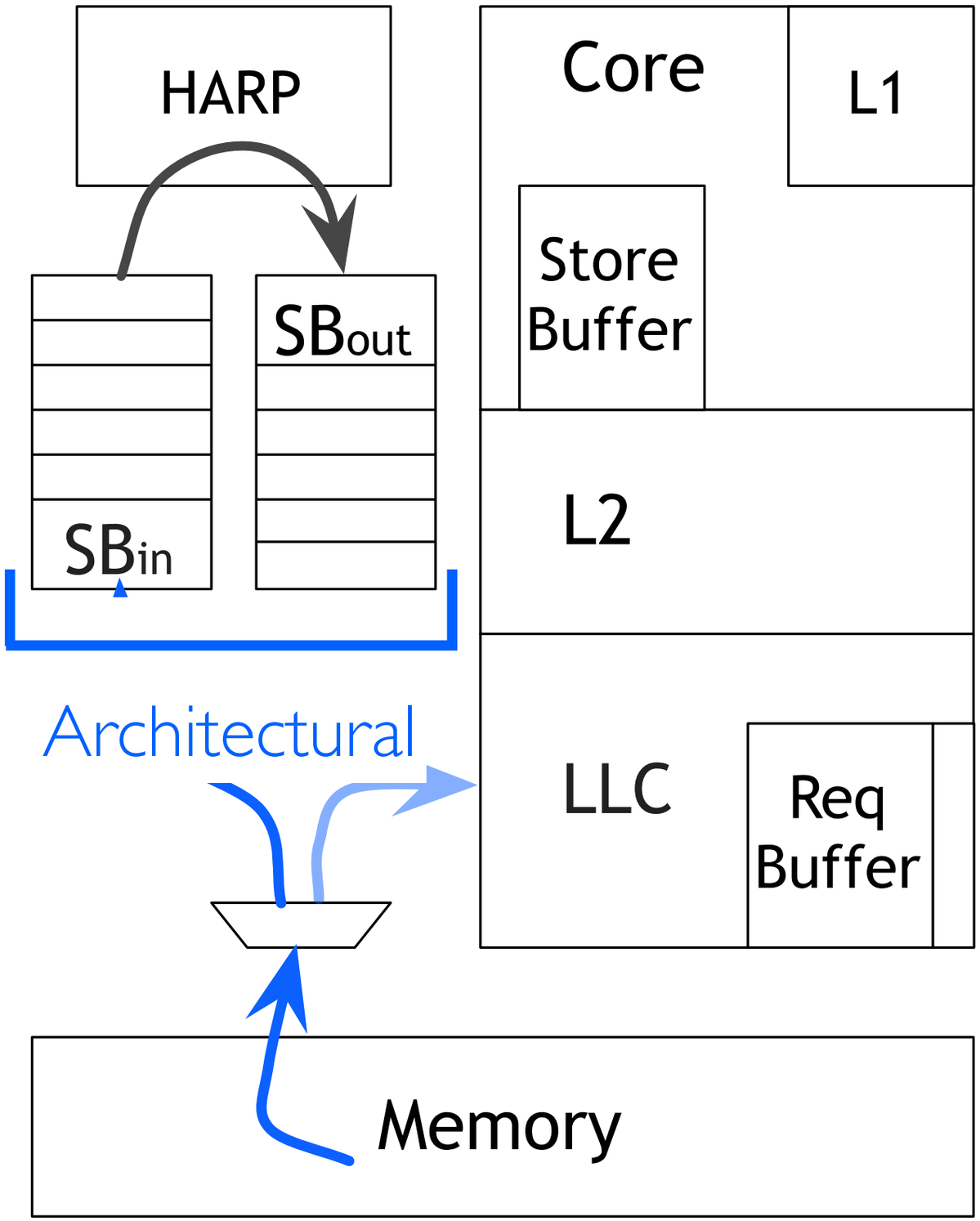
SB ISA

sbload

sbstore

**sbsave**

sbrestore



# Interrupts and Context Switches

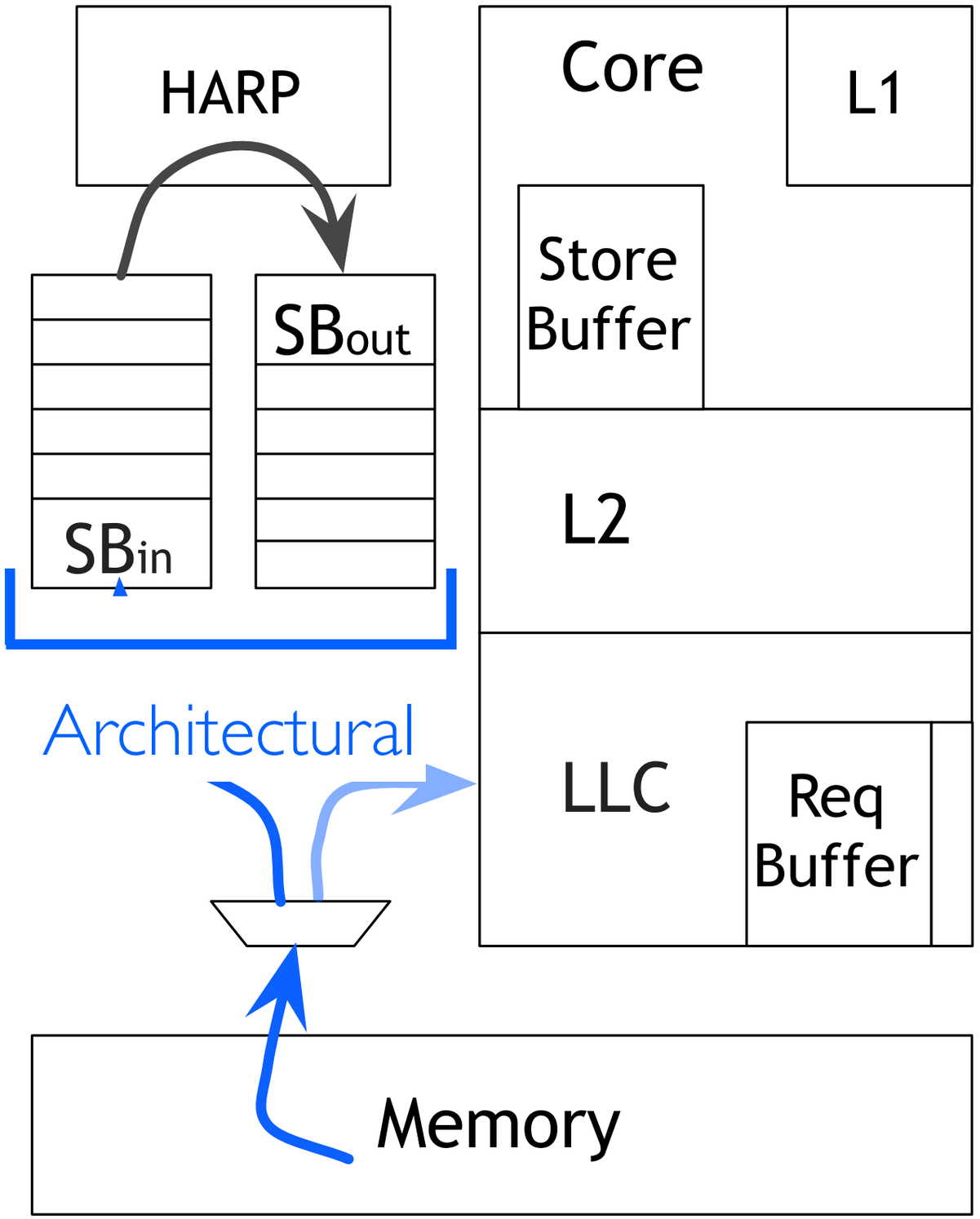
SB ISA

sbload

sbstore

**sbsave**

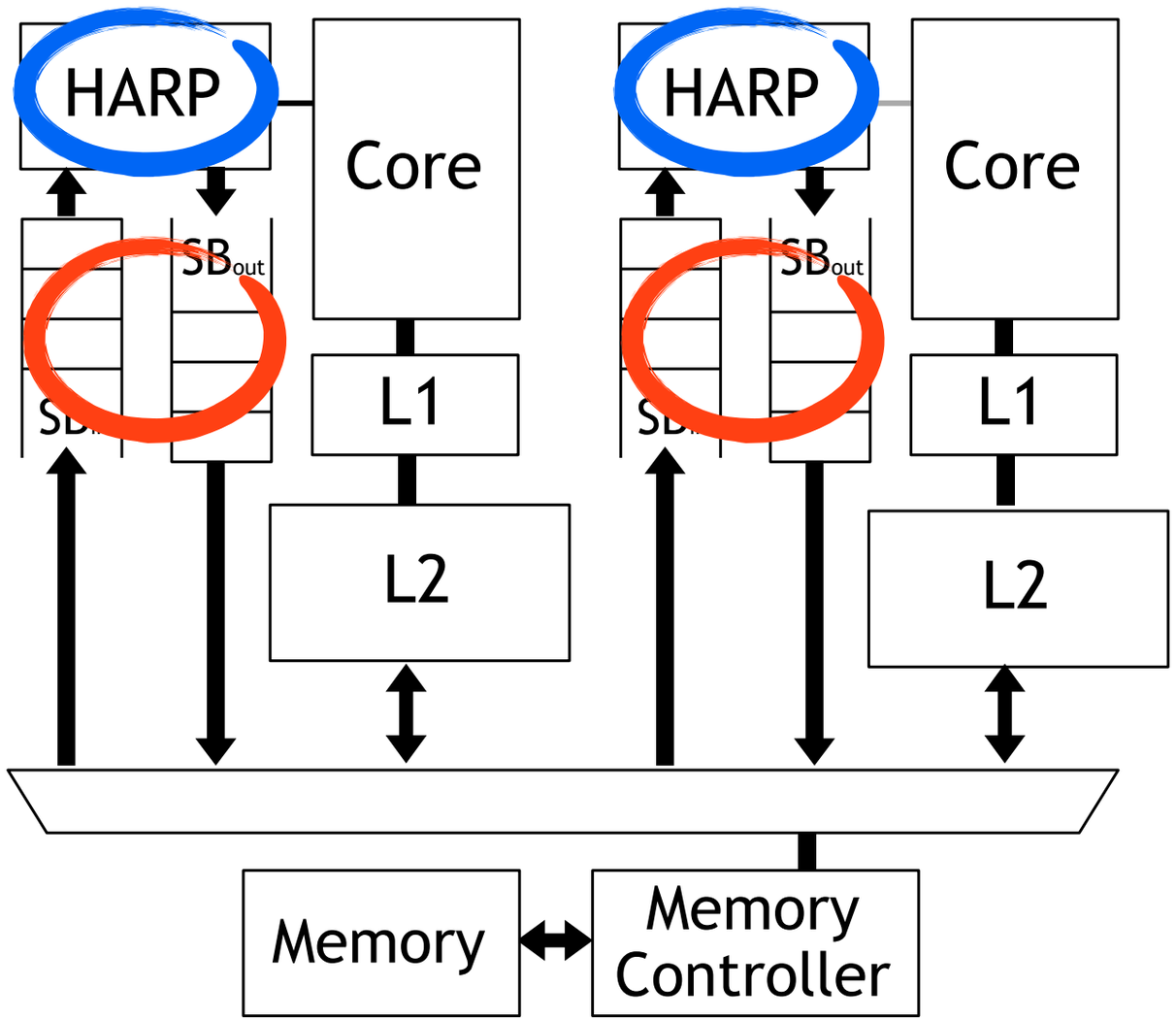
**sbrestore**



# Accelerator Integration Choice

- Tightly coupled and software controlled:
  - area/power savings
  - coherence
  - utilize hardware prefetchers
  - software-managed data layout
  - address-free domain for accelerators

# Remainder of the Talk




- ~~Brief System Overview~~
- ~~HARP UArch~~
- ~~Streaming Framework UArch~~
- HARP and Streaming Framework Evaluation
- Discussion and DSE

# Evaluation Methodology

# Evaluation Methodology

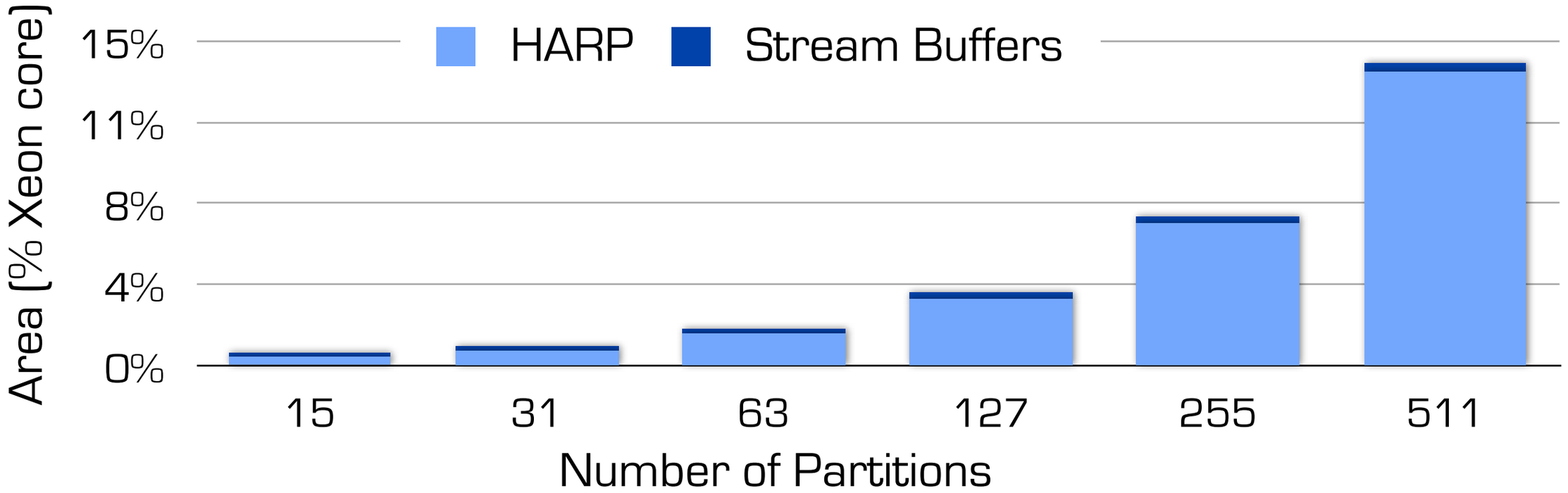
- HARP
  - Bluespec System Verilog implementation
  - Cycle-accurate simulation in BlueSim
  - Synthesis, P&R with Synopsys (32nm std cells)

# Evaluation Methodology

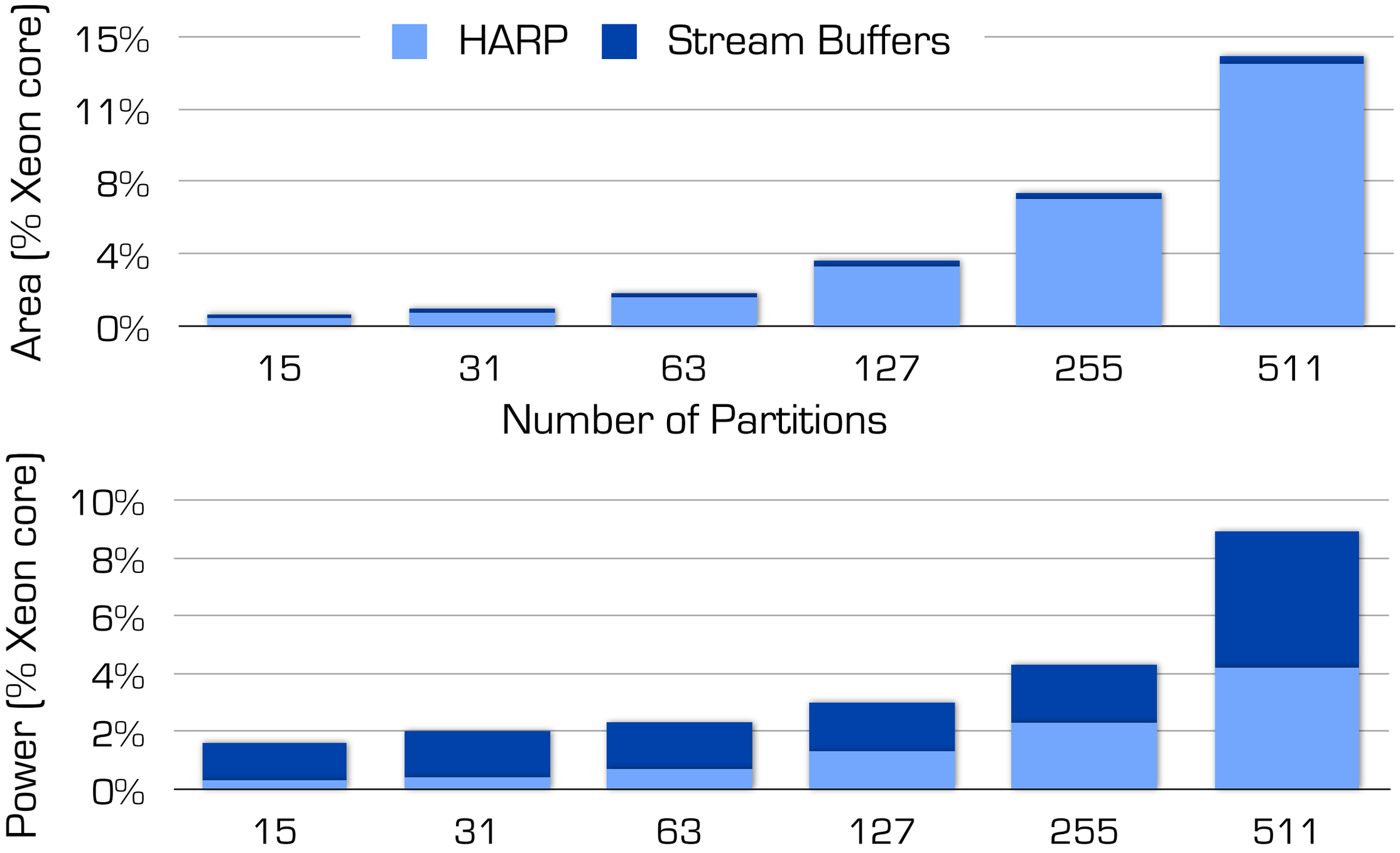
- HARP
  - Bluespec System Verilog implementation
  - Cycle-accurate simulation in BlueSim
  - Synthesis, P&R with Synopsys (32nm std cells)

- Streaming framework
  - 3 versions of 1GB table memcpy: c-lib, ASM (scalar), ASM(vector)
  - Conservative area/power estimates with CACTI

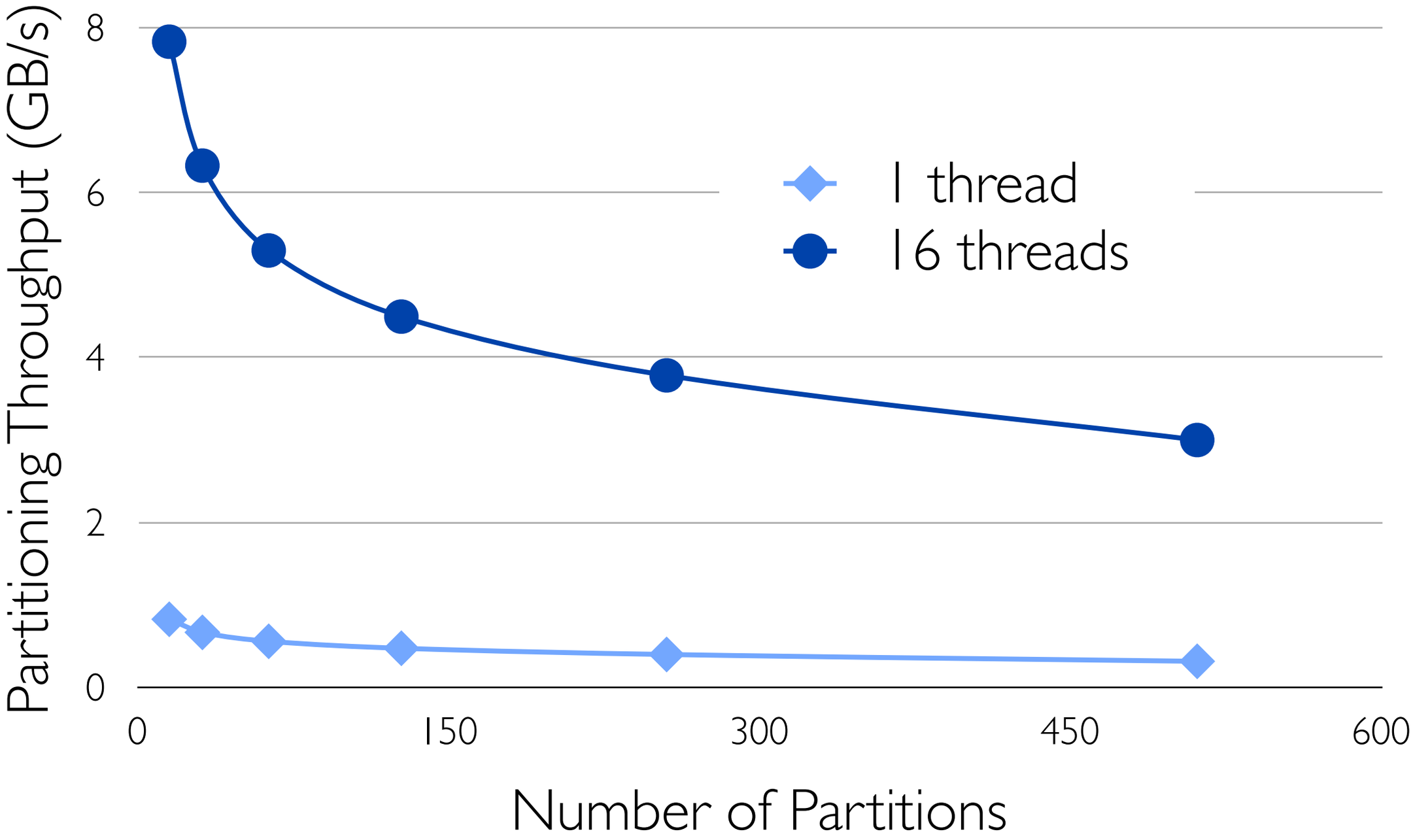
# Area and Power Overheads



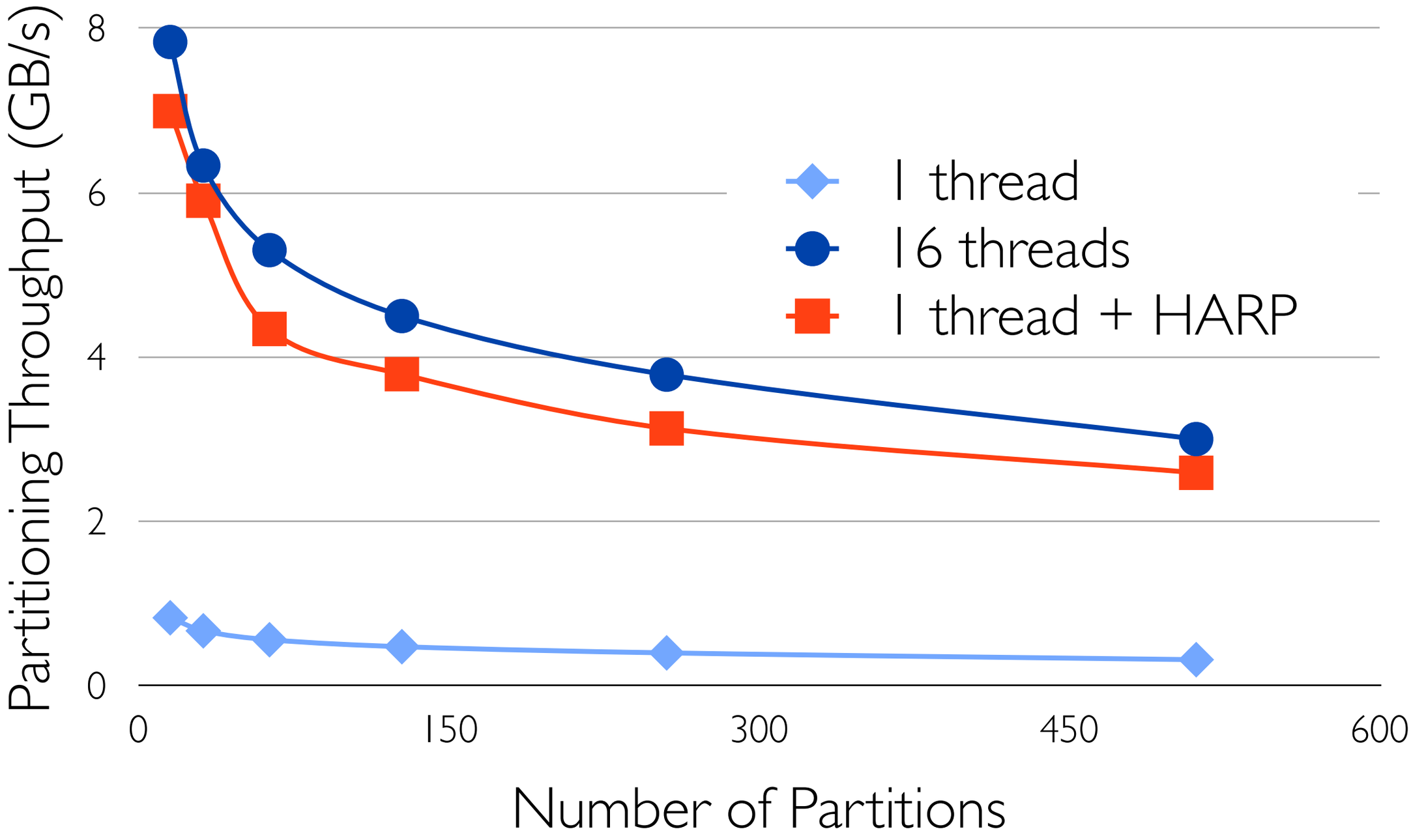
# Area and Power Overheads



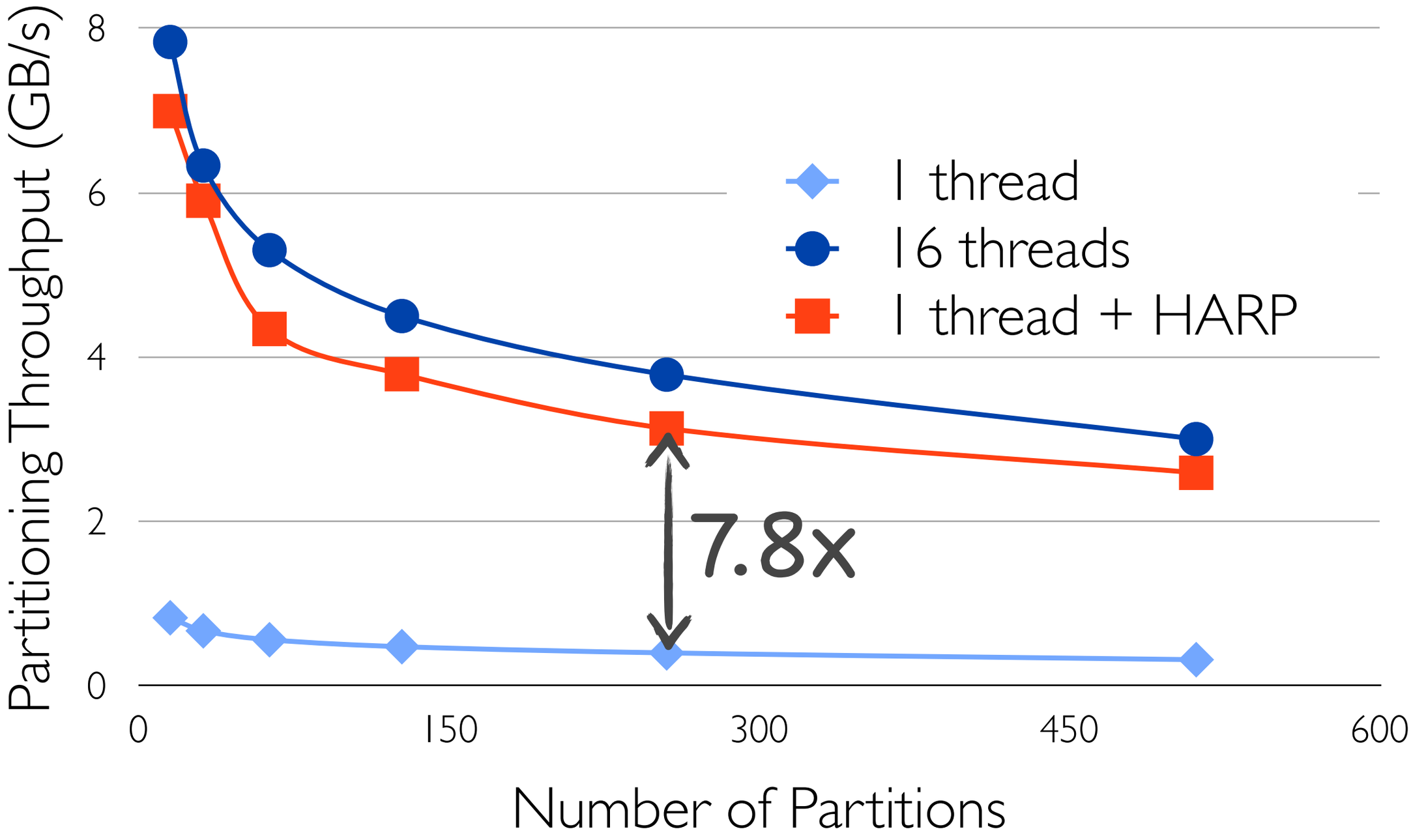
# SW Partitioning Performance



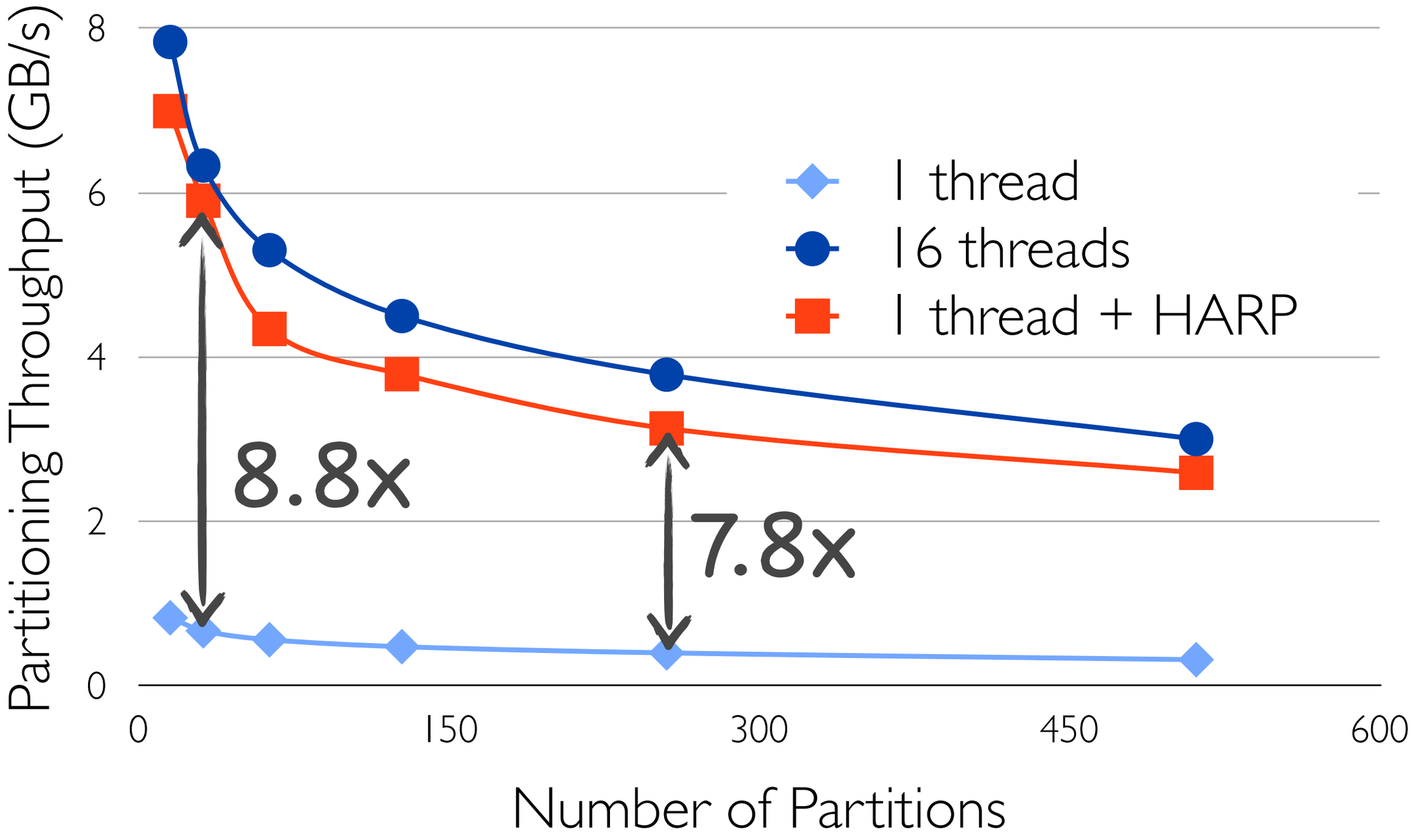
# Performance Evaluation



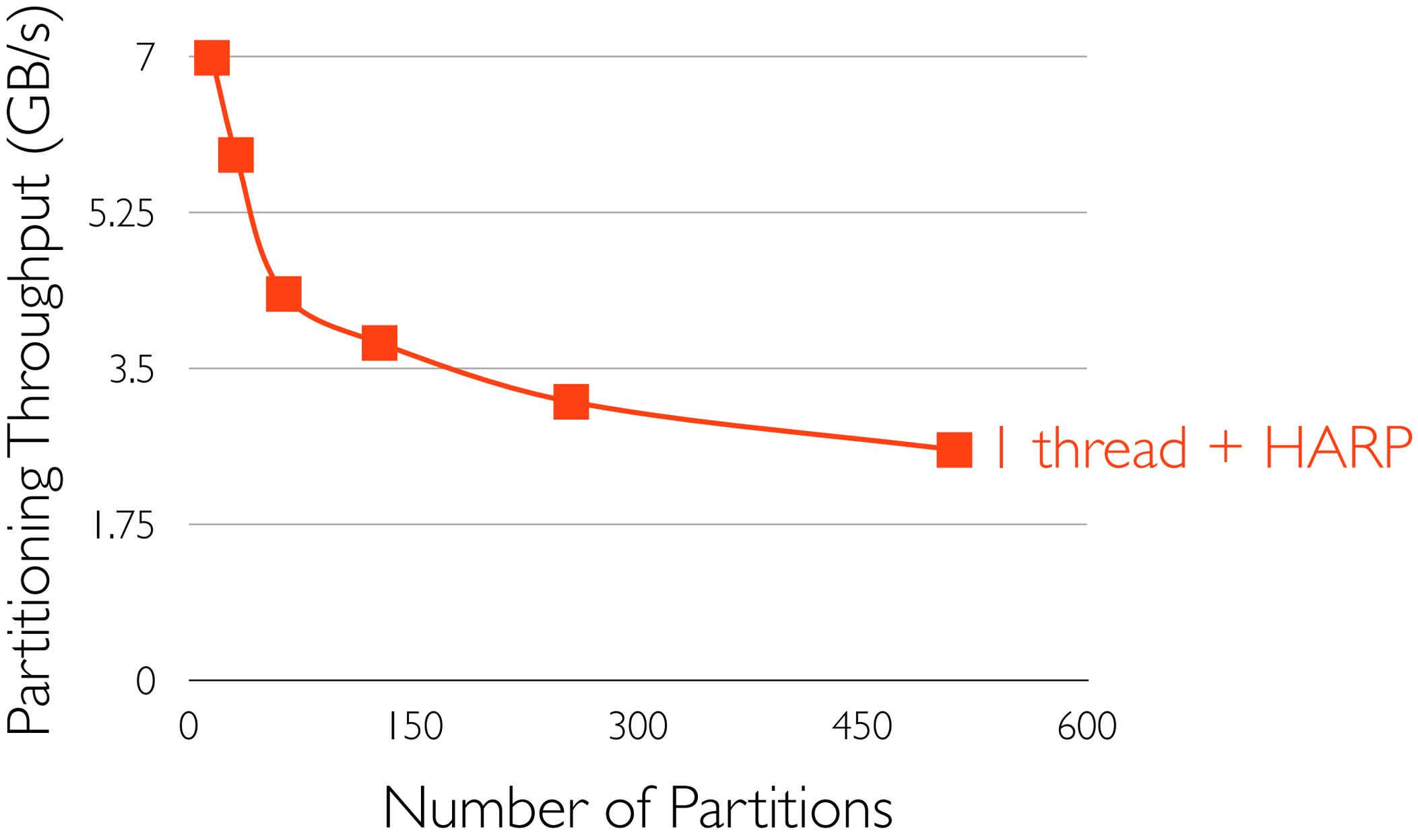
# Performance Evaluation



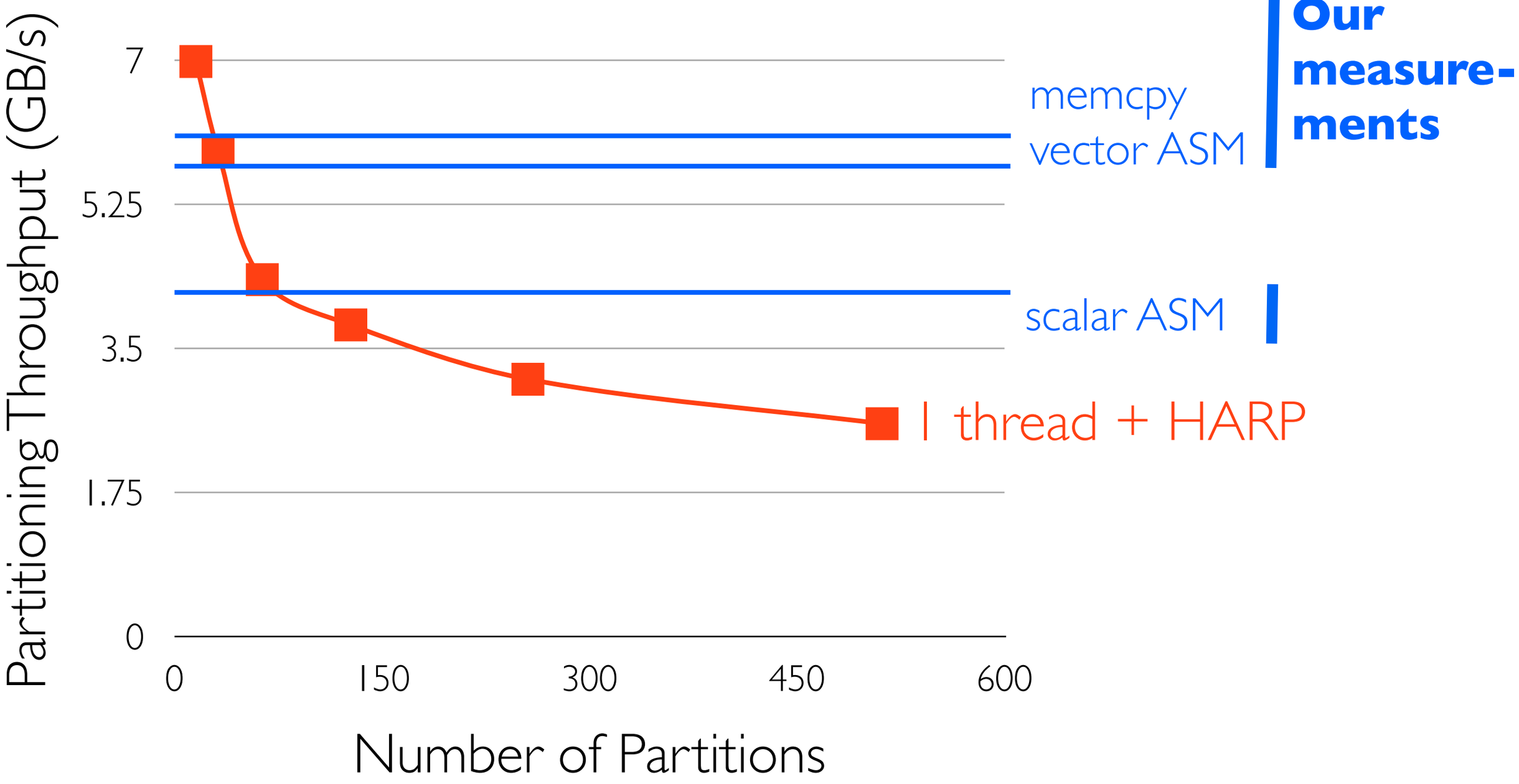
# Performance Evaluation



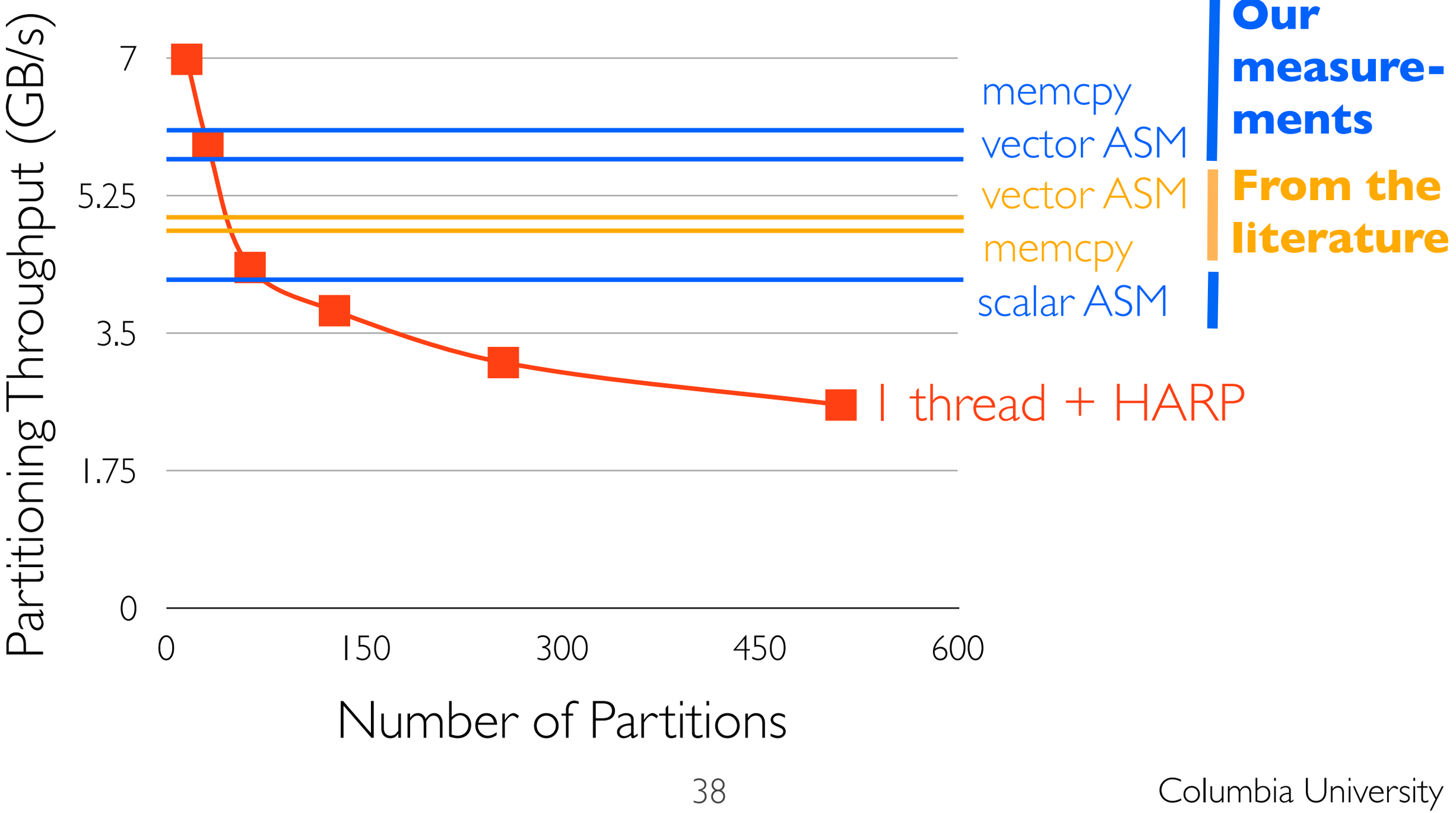
# Streaming Framework Provides Sufficient BW to Feed HARP?



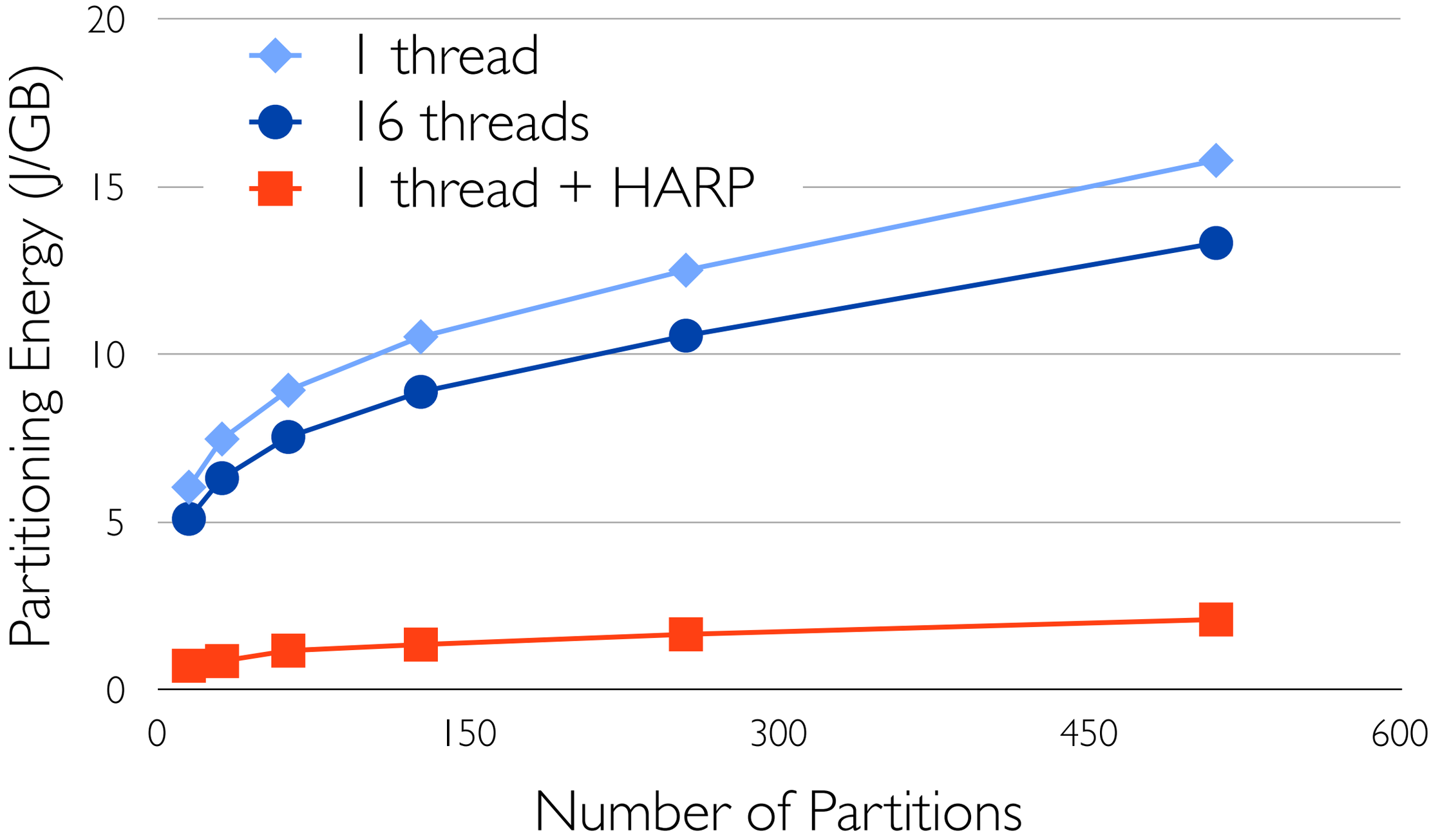
# Streaming Framework Provides Sufficient BW to Feed HARP?



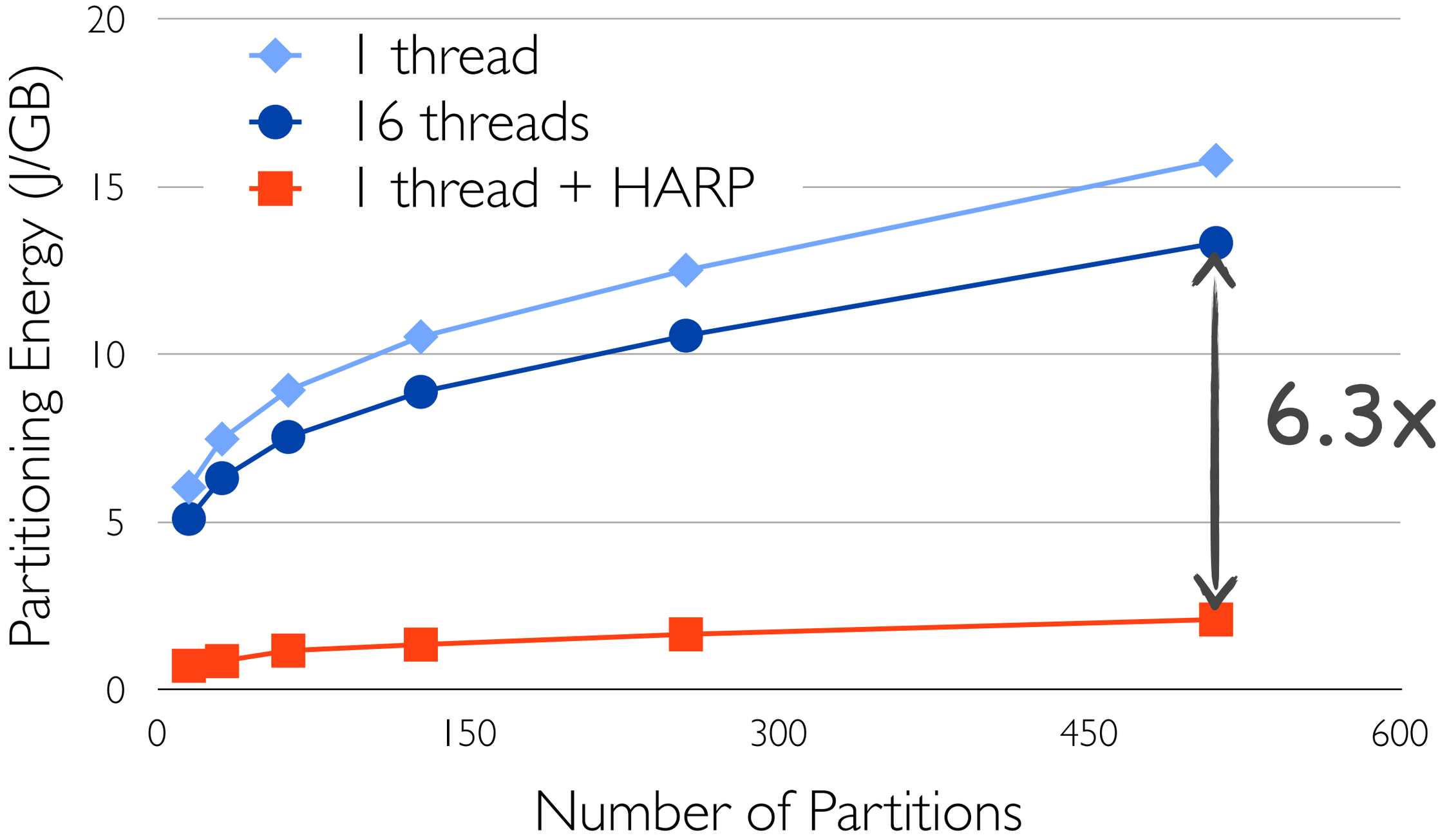
# Streaming Framework Provides Sufficient BW to Feed HARP?



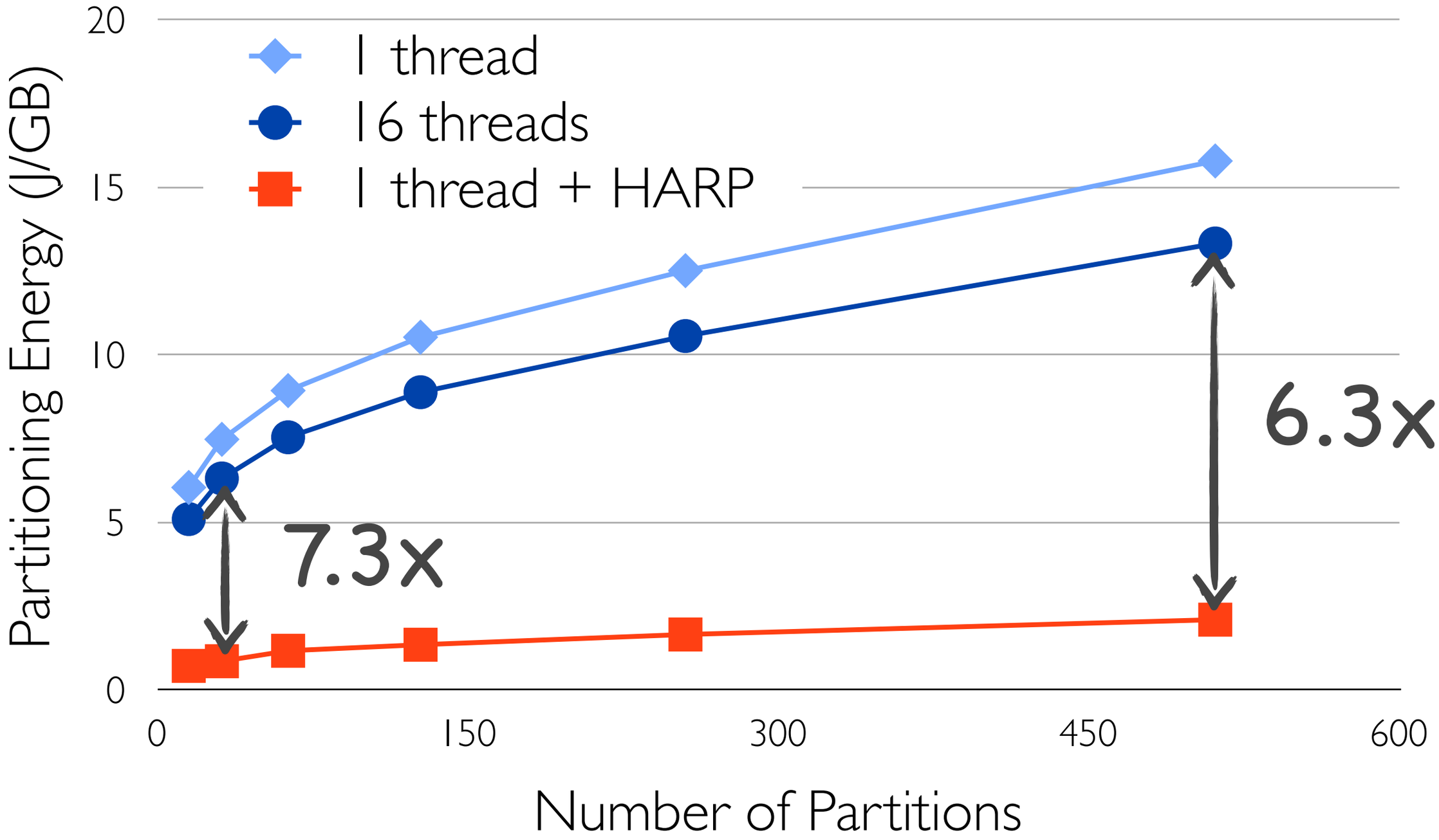
# HARP Energy vs. SW



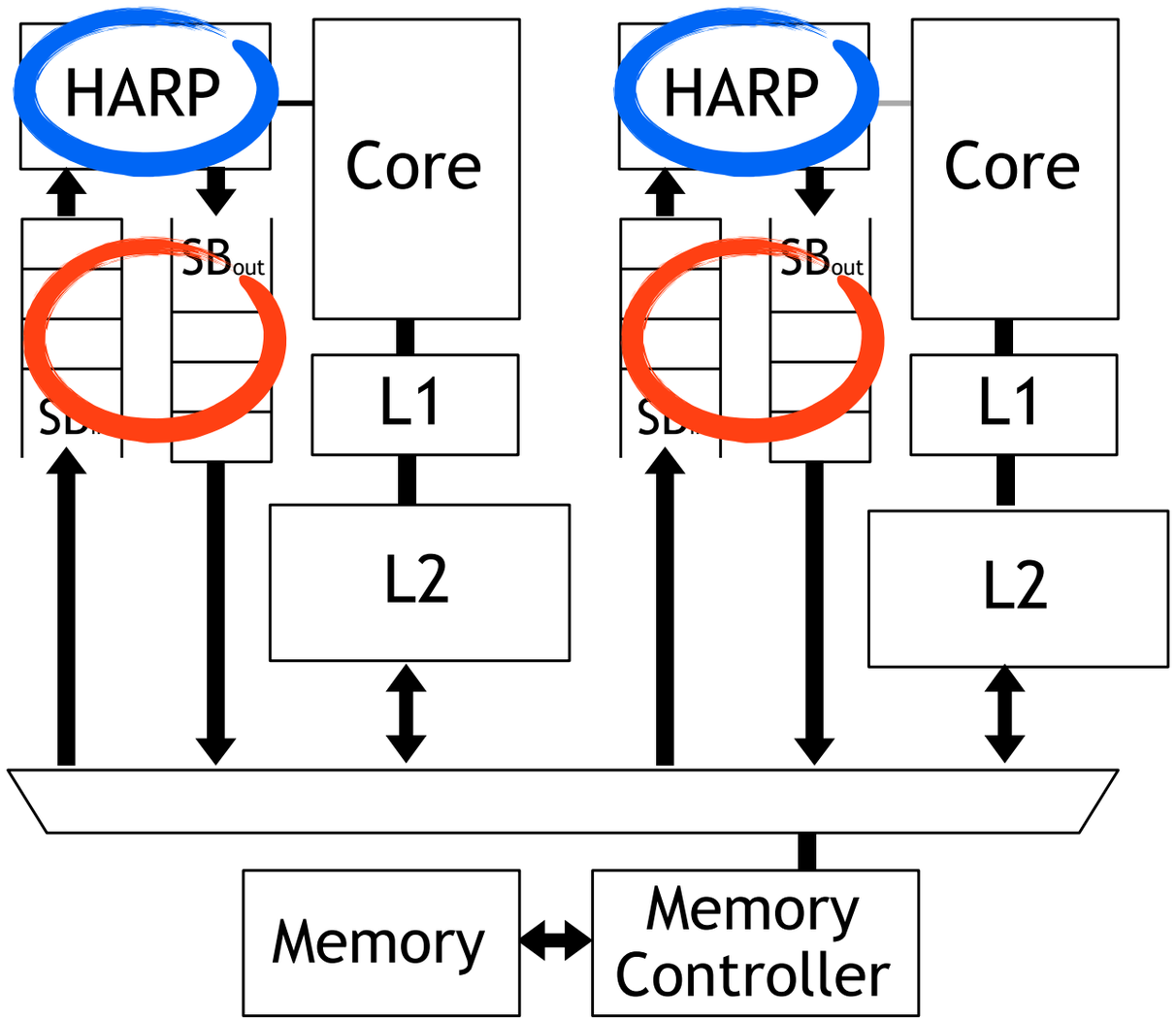
# HARP Energy vs. SW



# HARP Energy vs. SW



# Remainder of the Talk

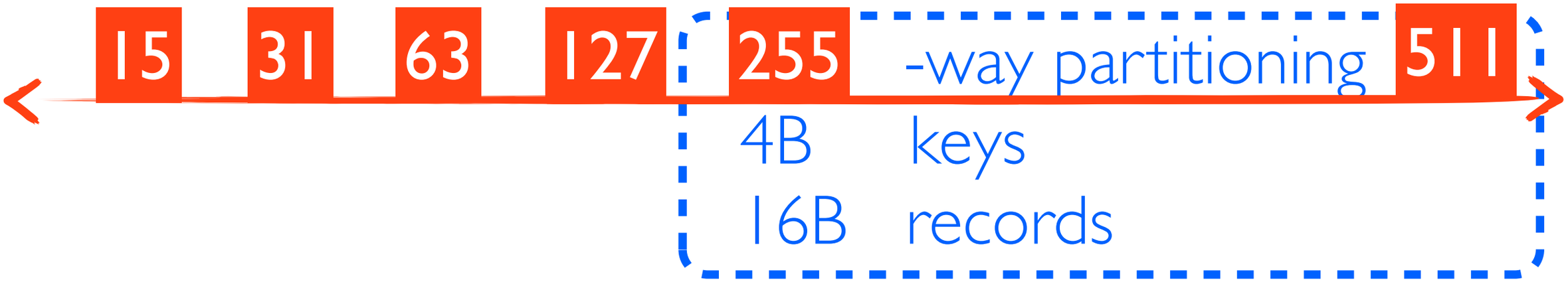



- ~~Brief System Overview~~
- ~~HARP UArch~~
- ~~Streaming Framework UArch~~
- ~~HARP and Streaming Framework Evaluation~~
- Discussion

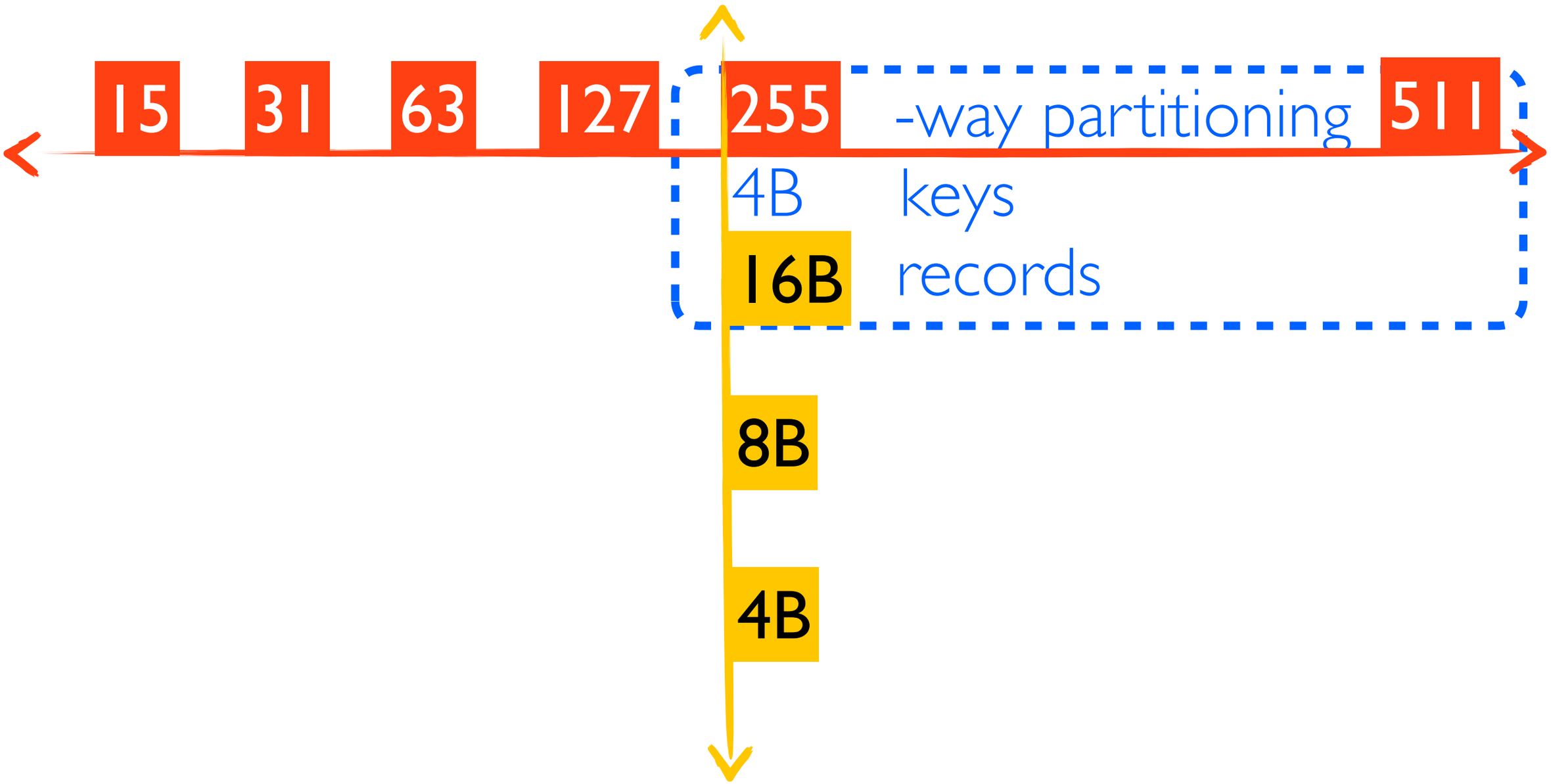
# Design Space Exploration (in the paper)

255 -way partitioning
4B keys
16B records

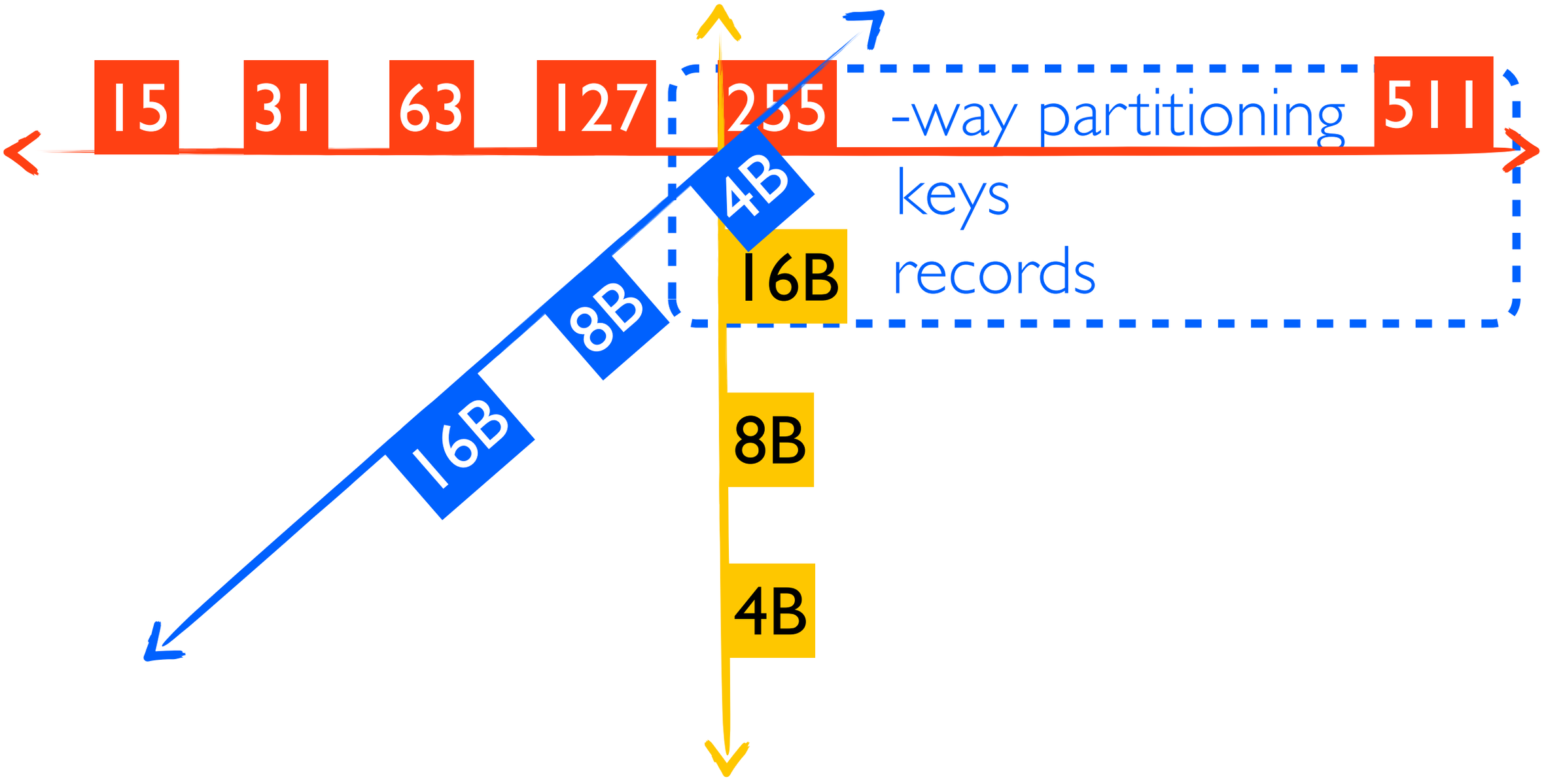
# Design Space Exploration (in the paper)



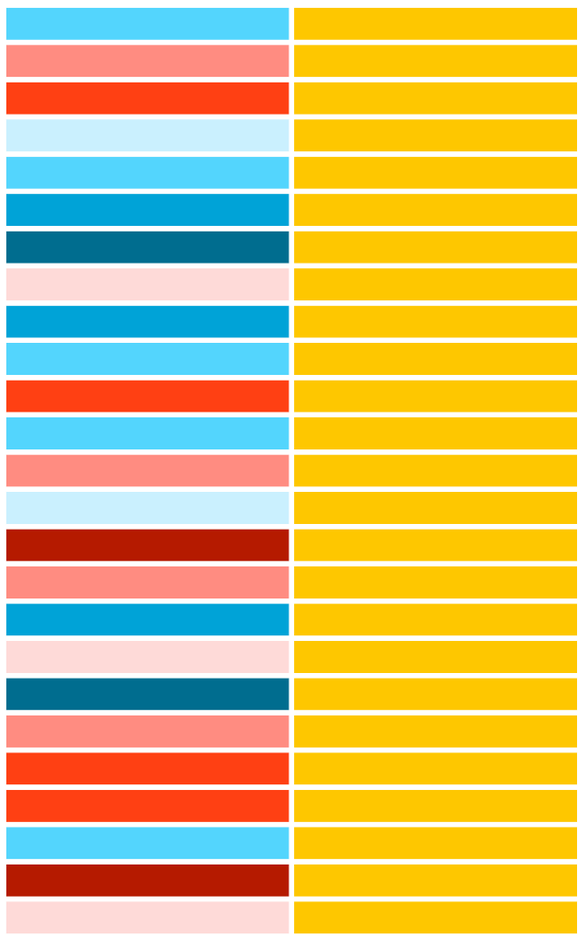
# Design Space Exploration (in the paper)



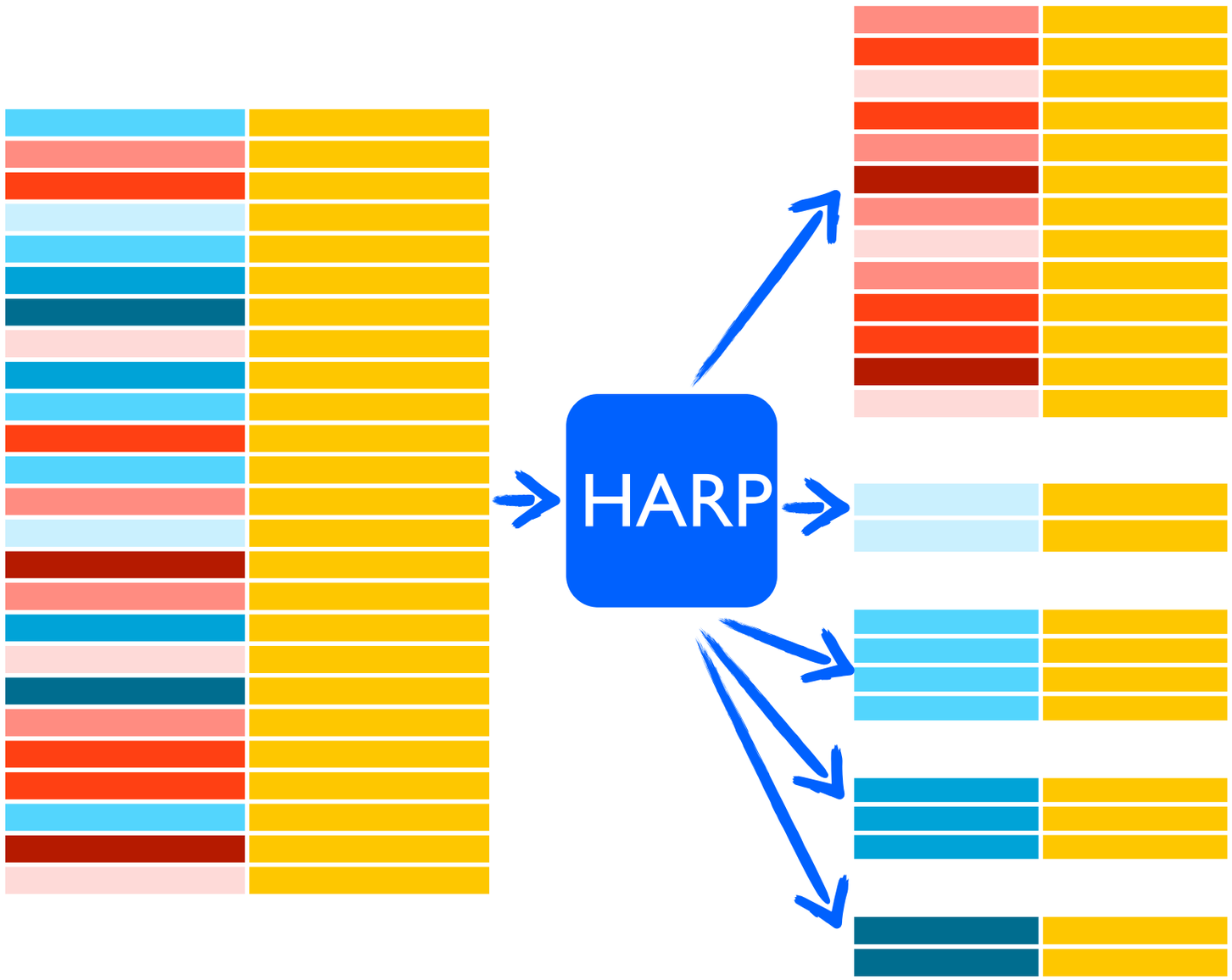
# Design Space Exploration (in the paper)



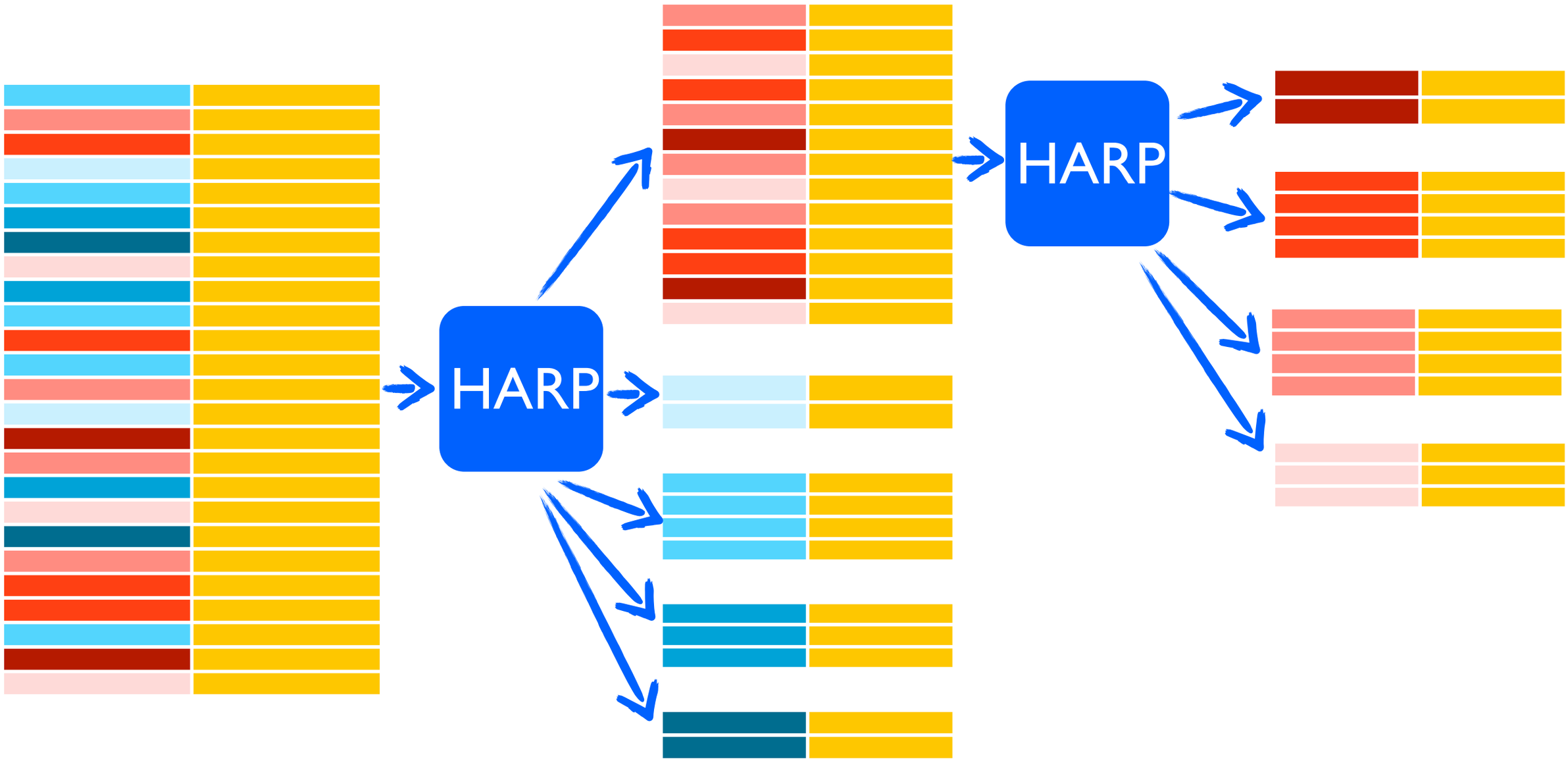
# Coping with Fixed Resources: Partitioning Factor > Partitioner Size



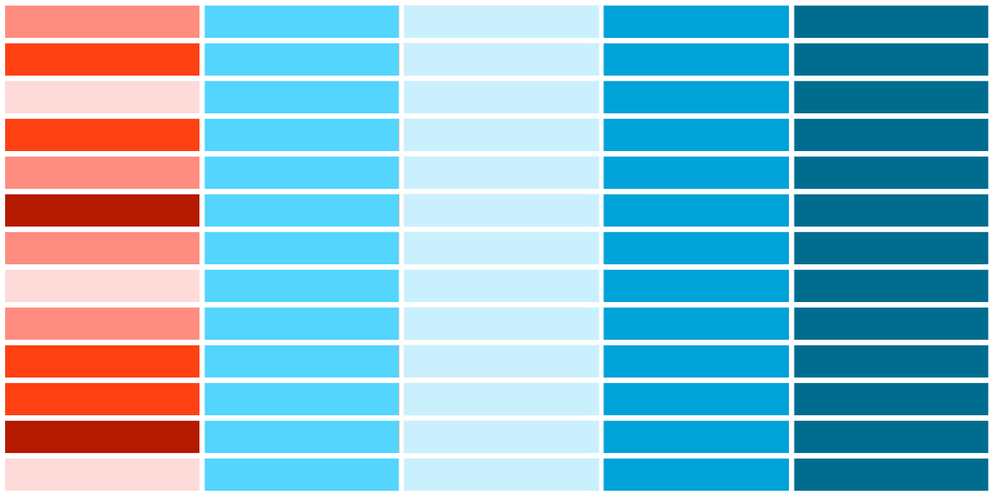
# Coping with Fixed Resources: Partitioning Factor > Partitioner Size



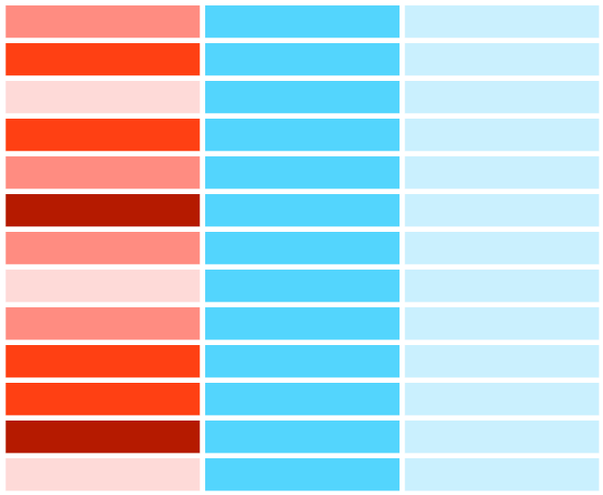
# Coping with Fixed Resources: Partitioning Factor > Partitioner Size

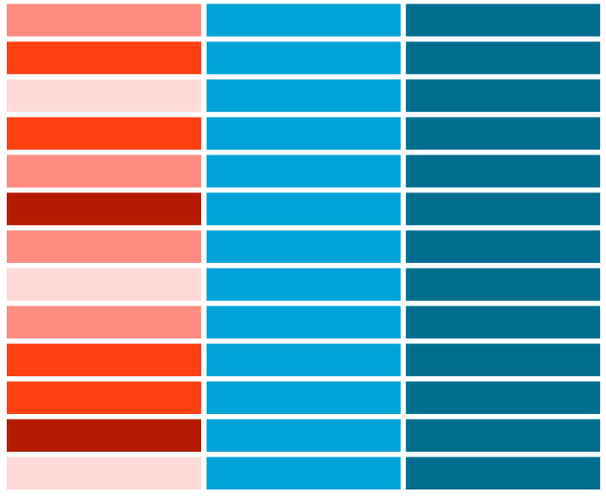


# Coping with Fixed Resources: Record Width > Record Size

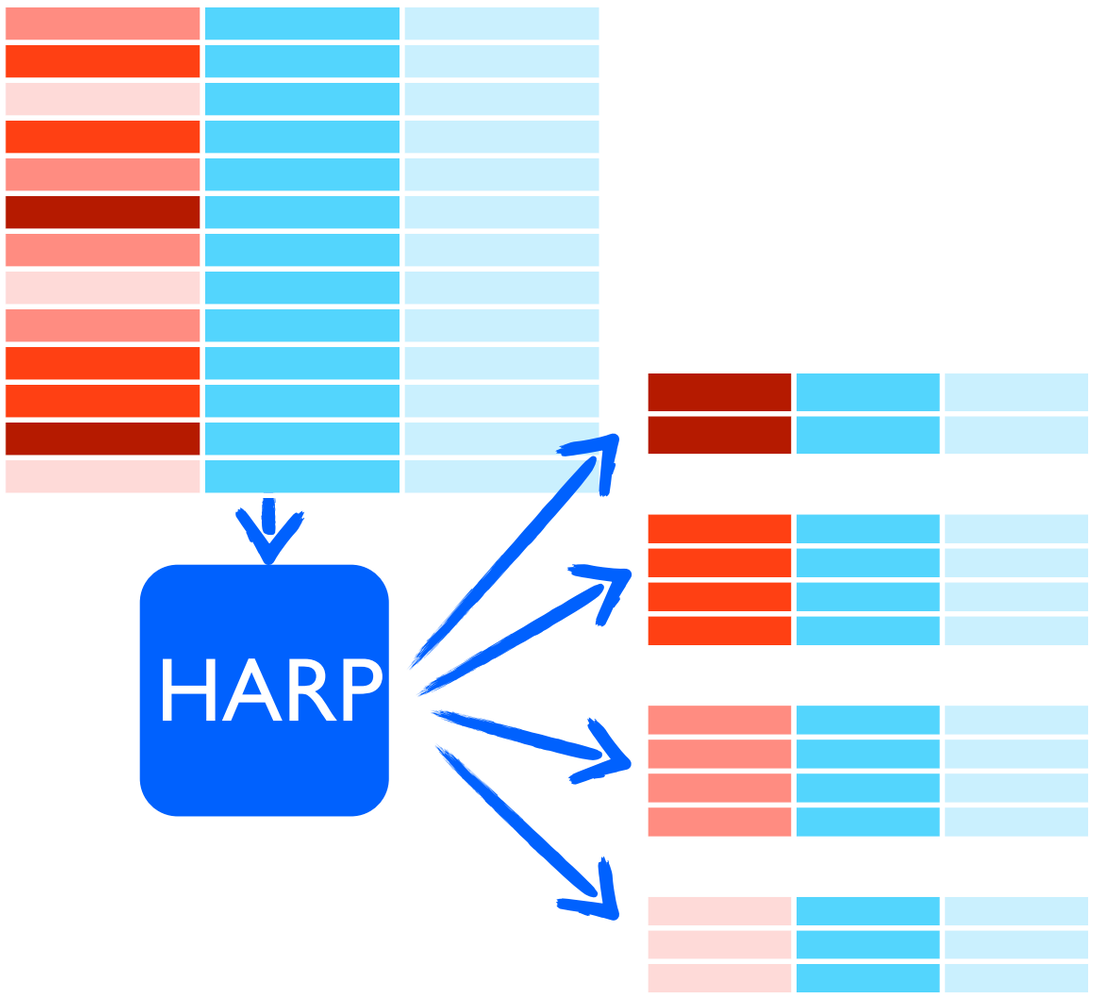


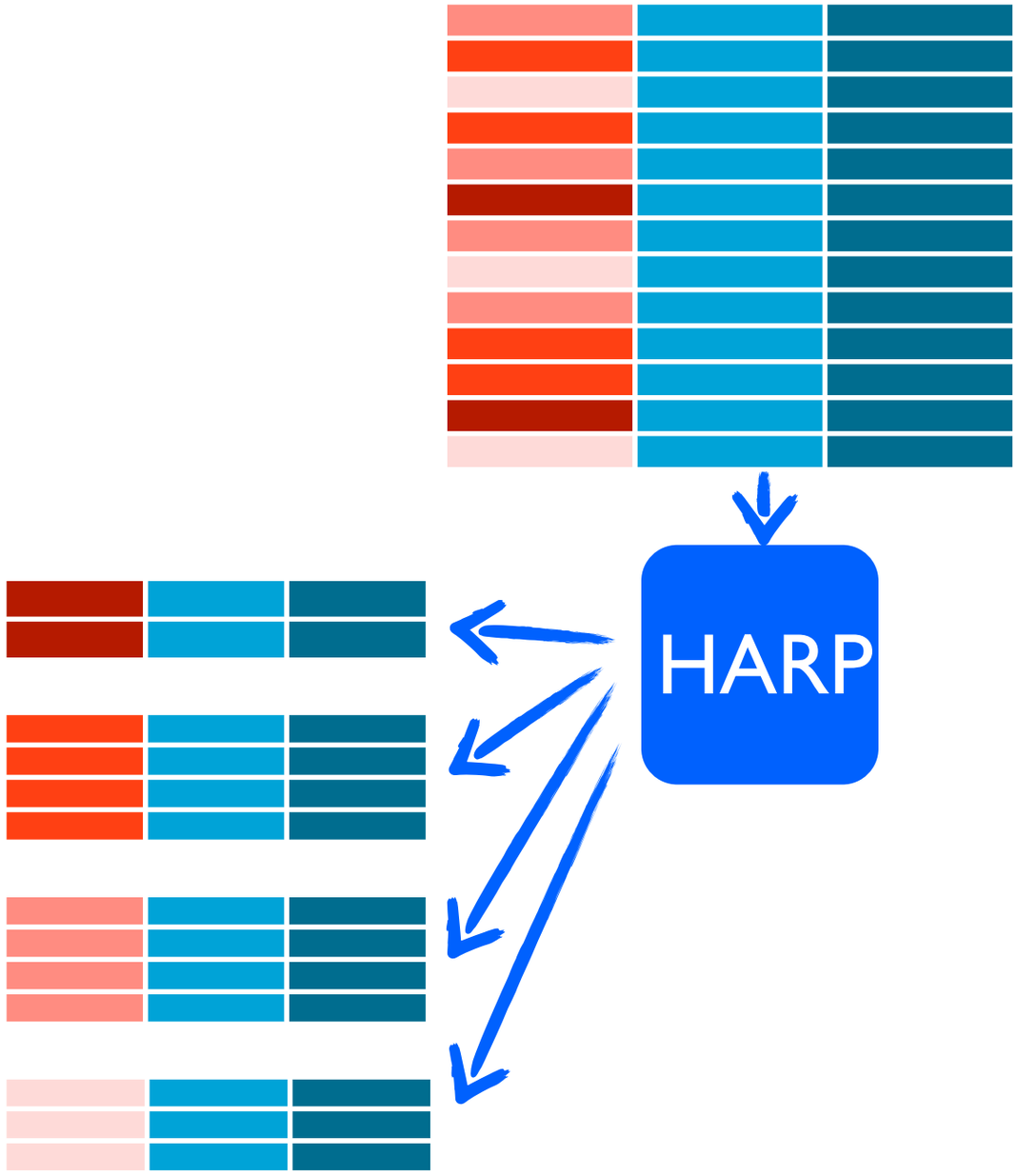
# Coping with Fixed Resources: Record Width > Record Size





# Coping with Fixed Resources: Record Width > Record Size



# Conclusion

- Data partitioning, which does not appear compute-heavy, can still benefit from acceleration
- Microarchitecture to pair streaming accelerator(s) to work closely with CPU
- Demonstrate how accelerators can rebalance system and improve memory bandwidth utilization, a scarce resource in big data analytics

# FIN