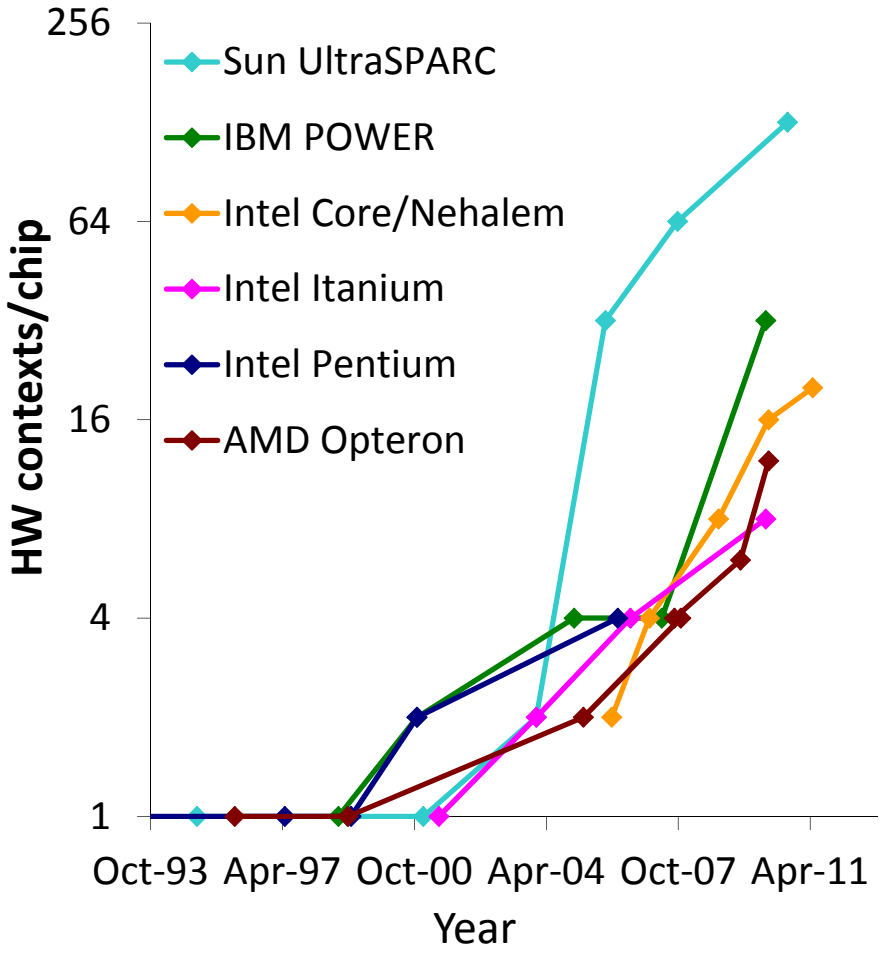
PLP: Page Latch-free Shared-everything OLTP

Ippokratis Pandis† **Pınar Tözün**‡ *Ryan Johnson*❖
Anastasia Ailamaki‡

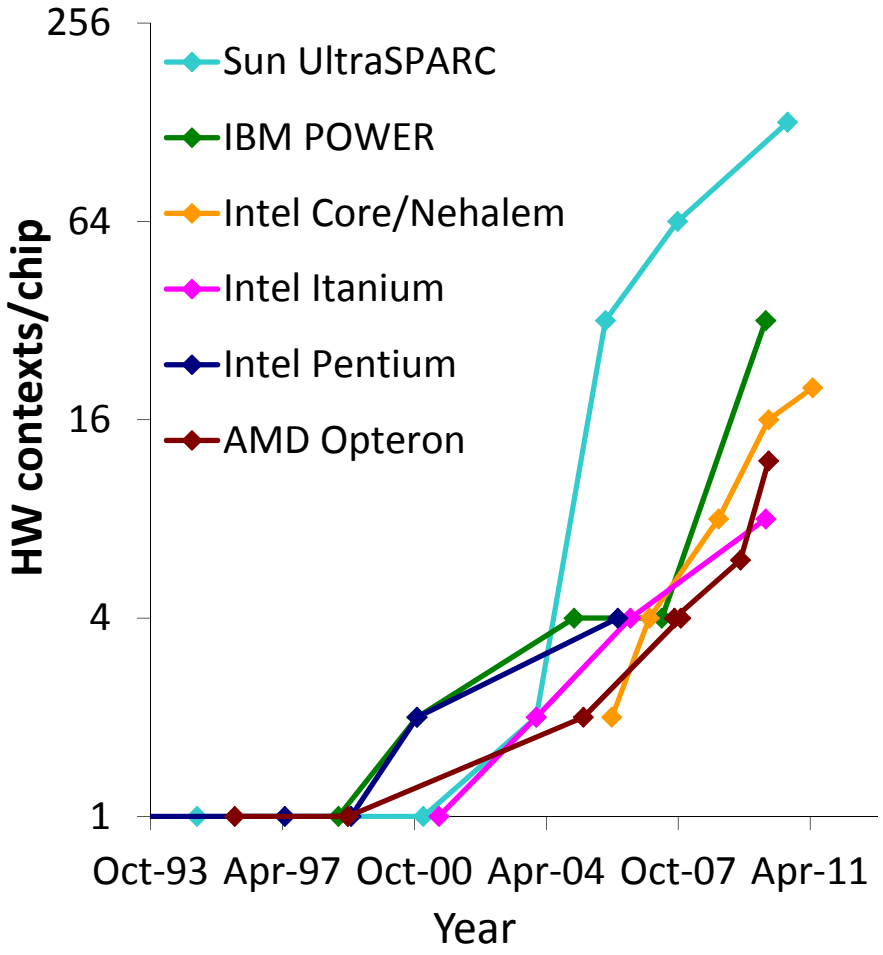
†IBM Almaden Research Center
‡École Polytechnique Fédérale de Lausanne
❖University of Toronto

DATA-INTENSIVE APPLICATIONS AND SYSTEMS

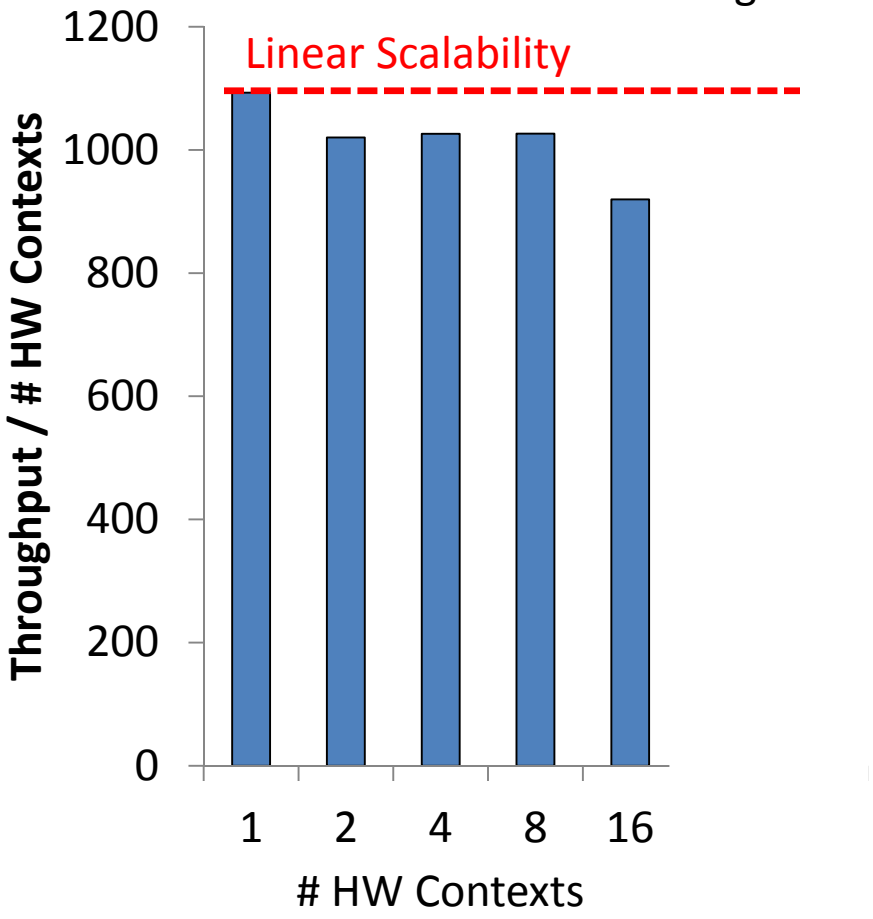
OLTP on Modern Hardware



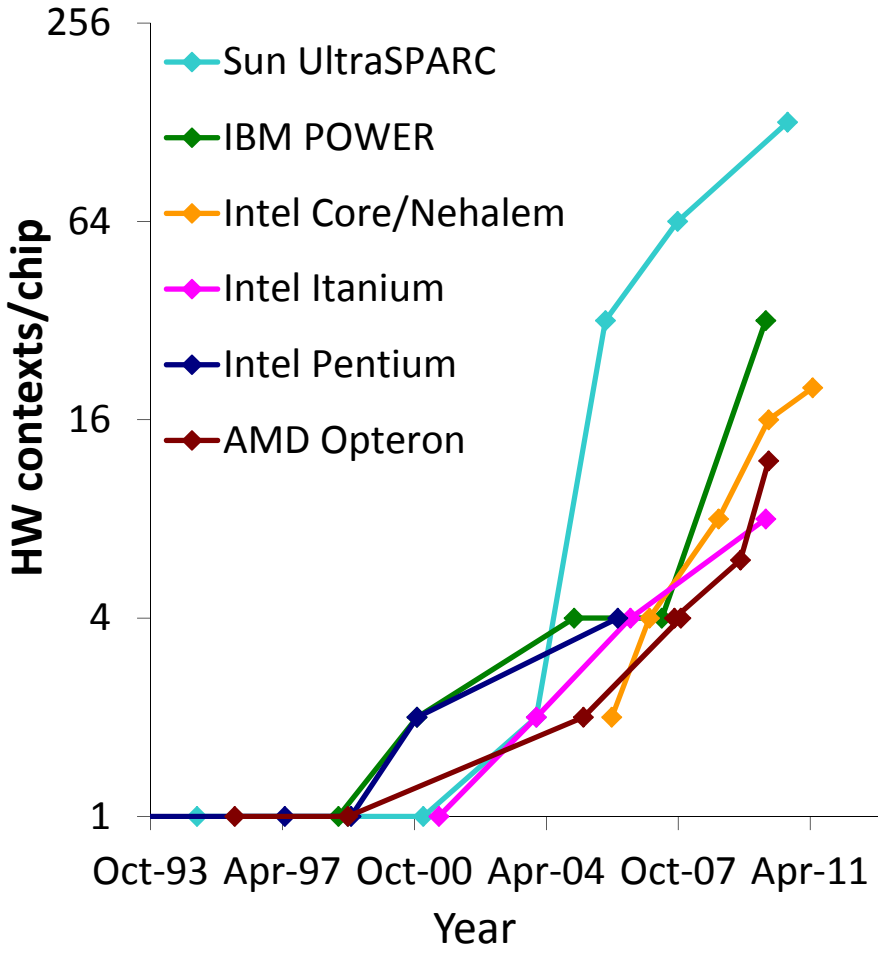
OLTP on Modern Hardware

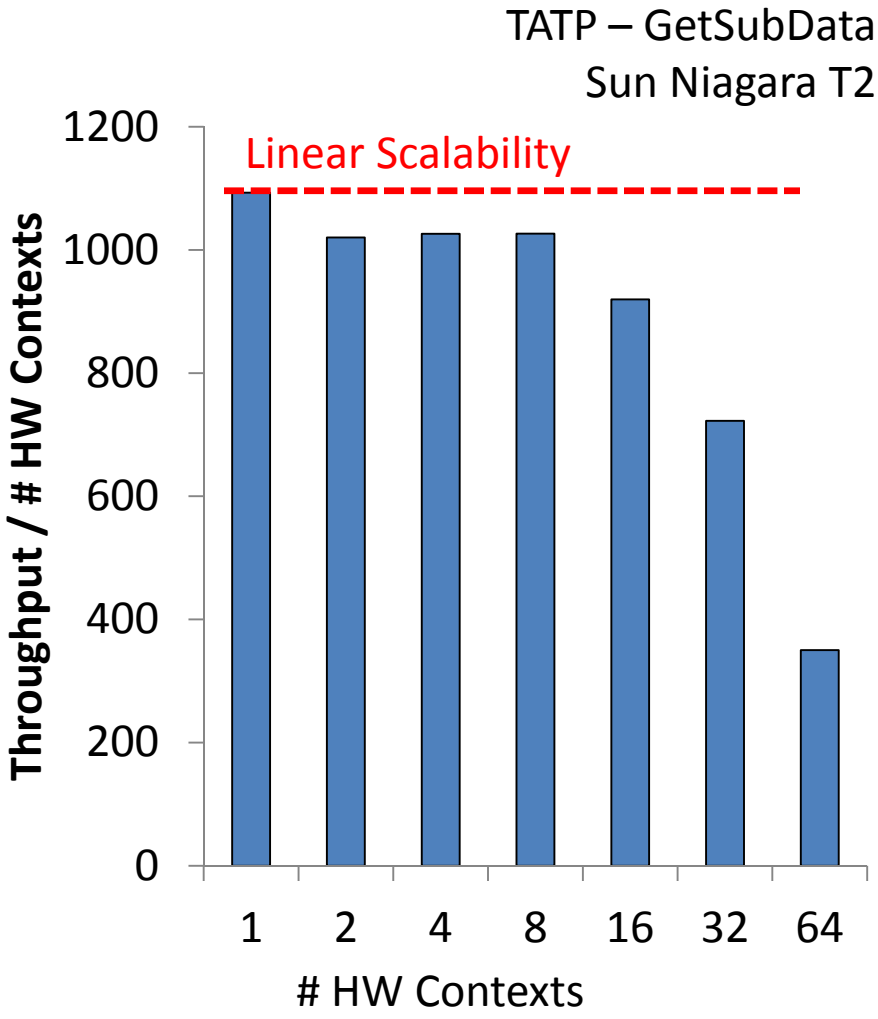


TATP – GetSubData
Sun Niagara T2

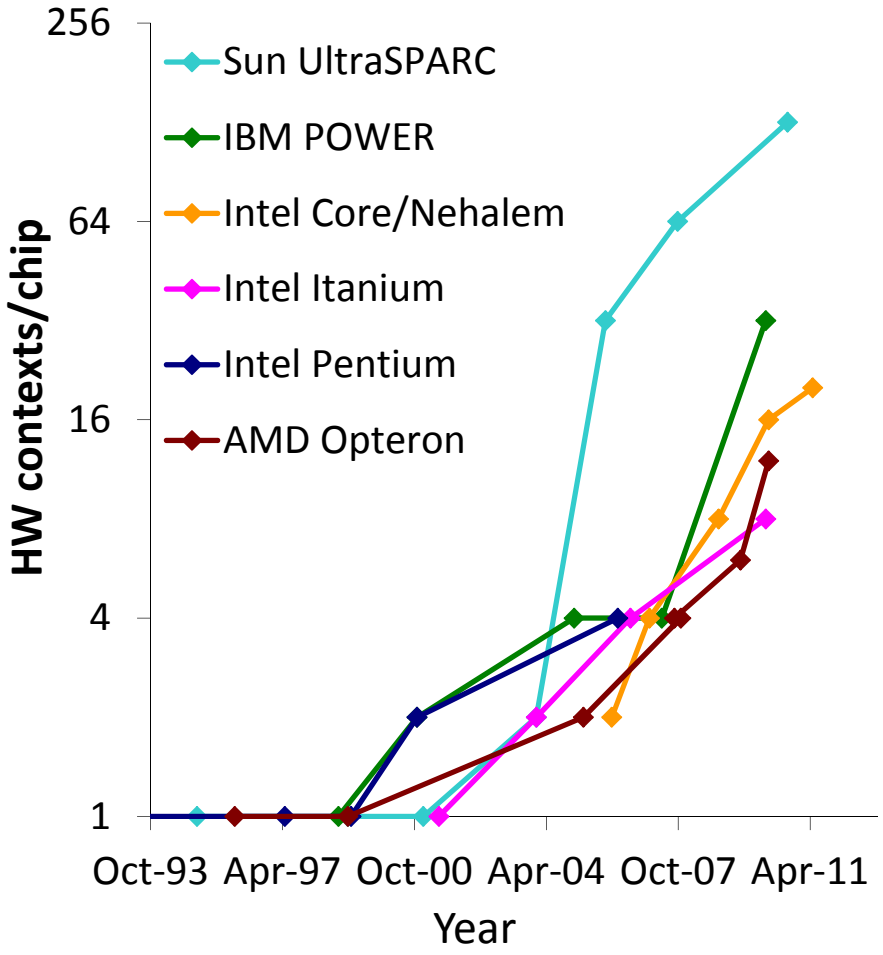


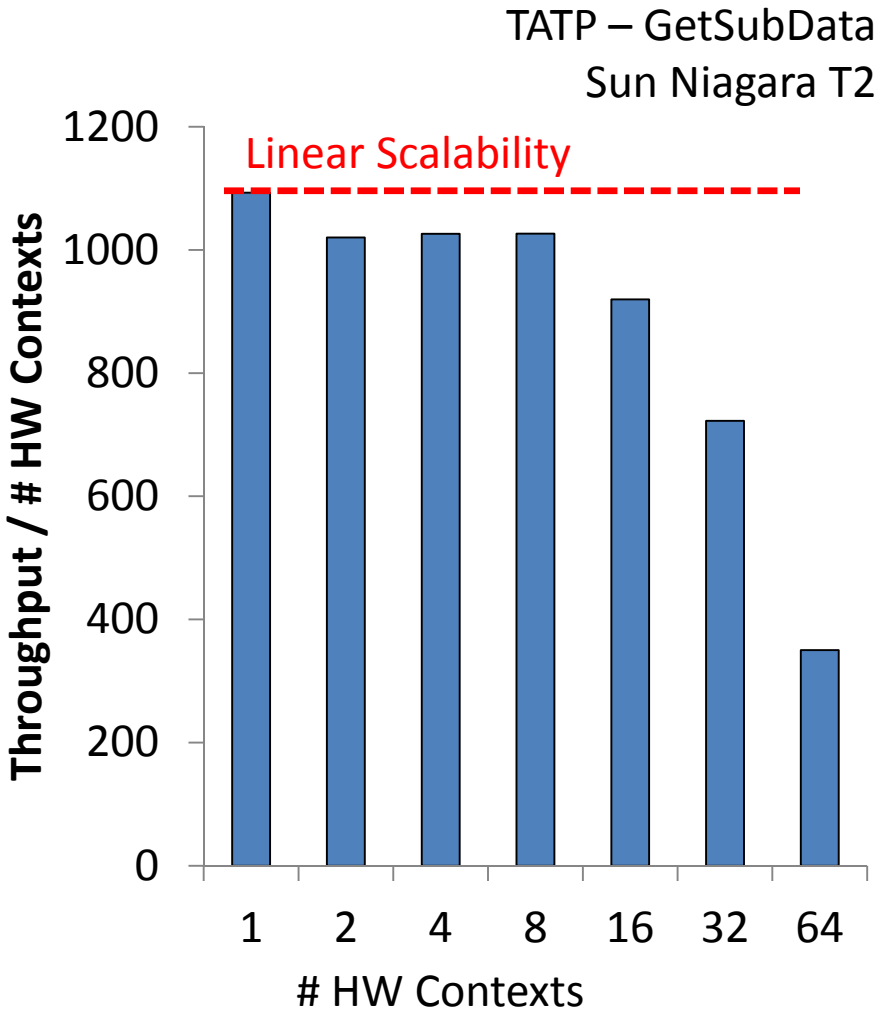
OLTP on Modern Hardware





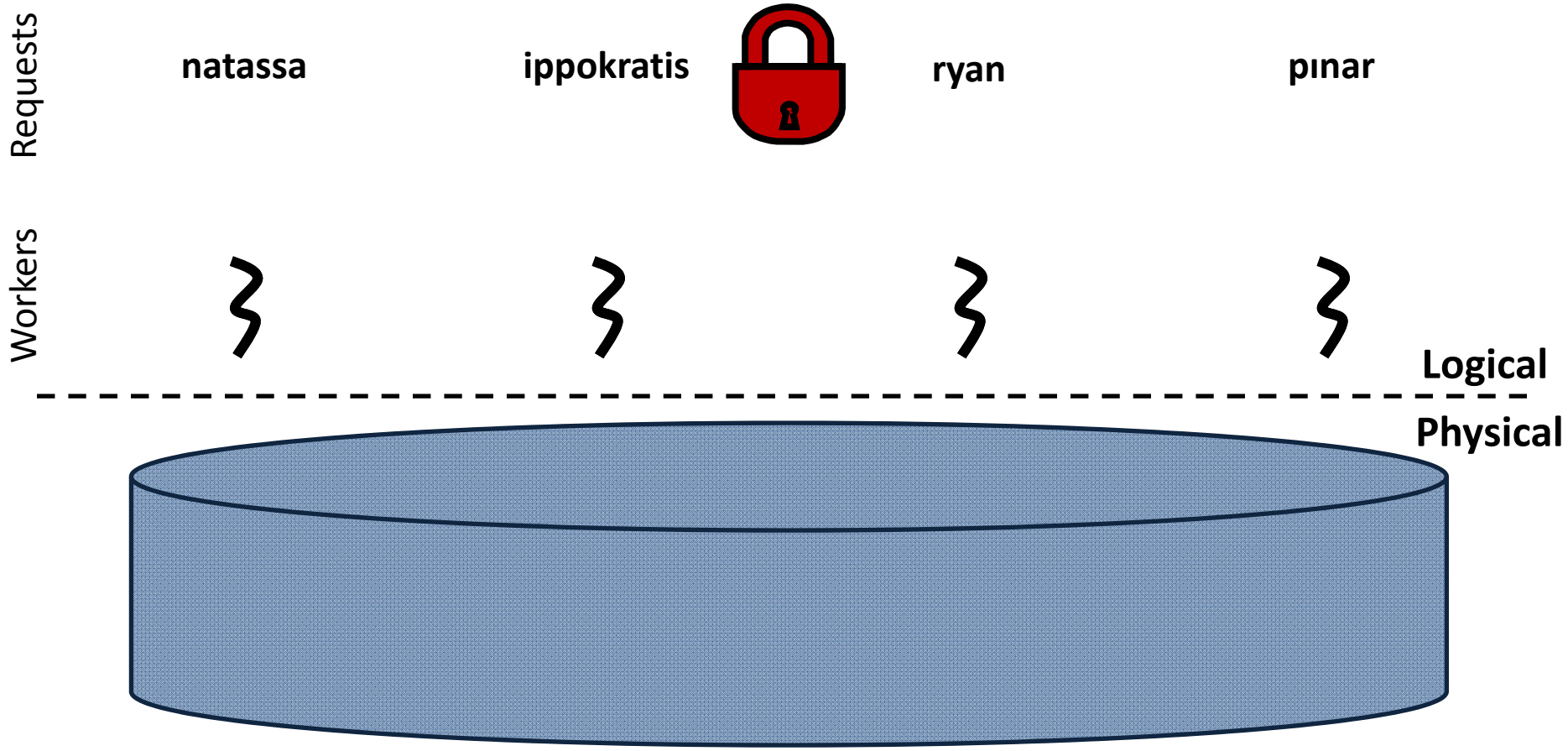
OLTP on Modern Hardware



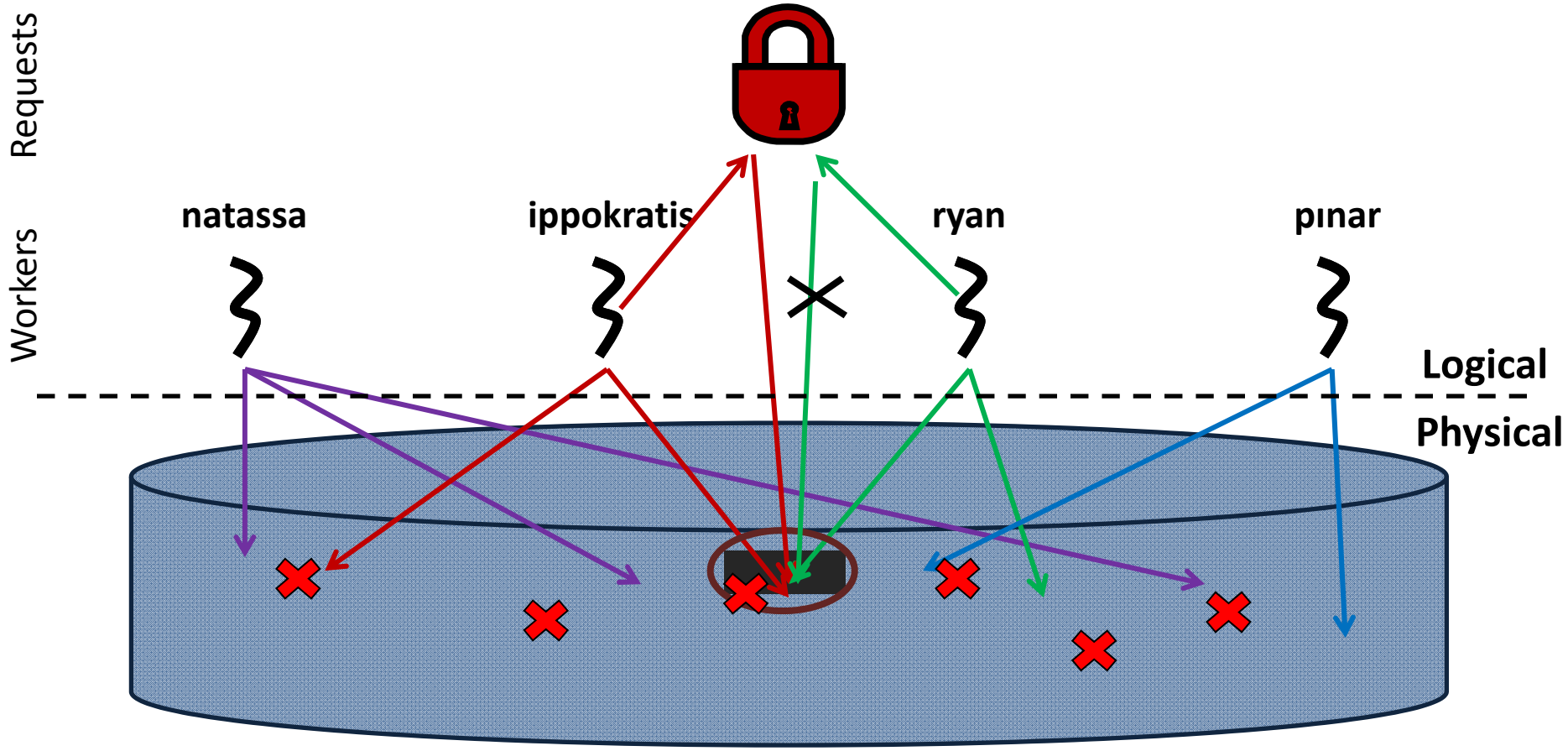


More HW Contexts != Higher Throughput

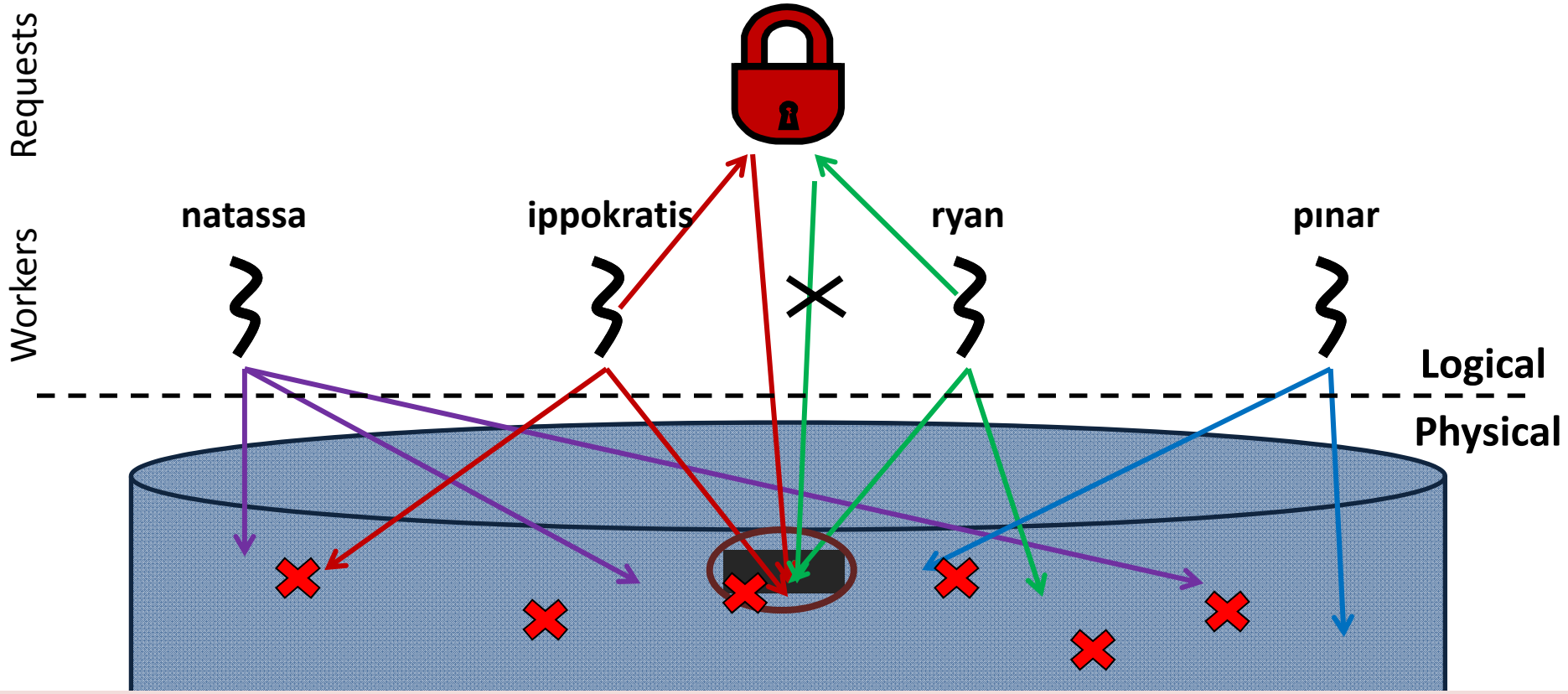
Shared-Everything



Shared-Everything

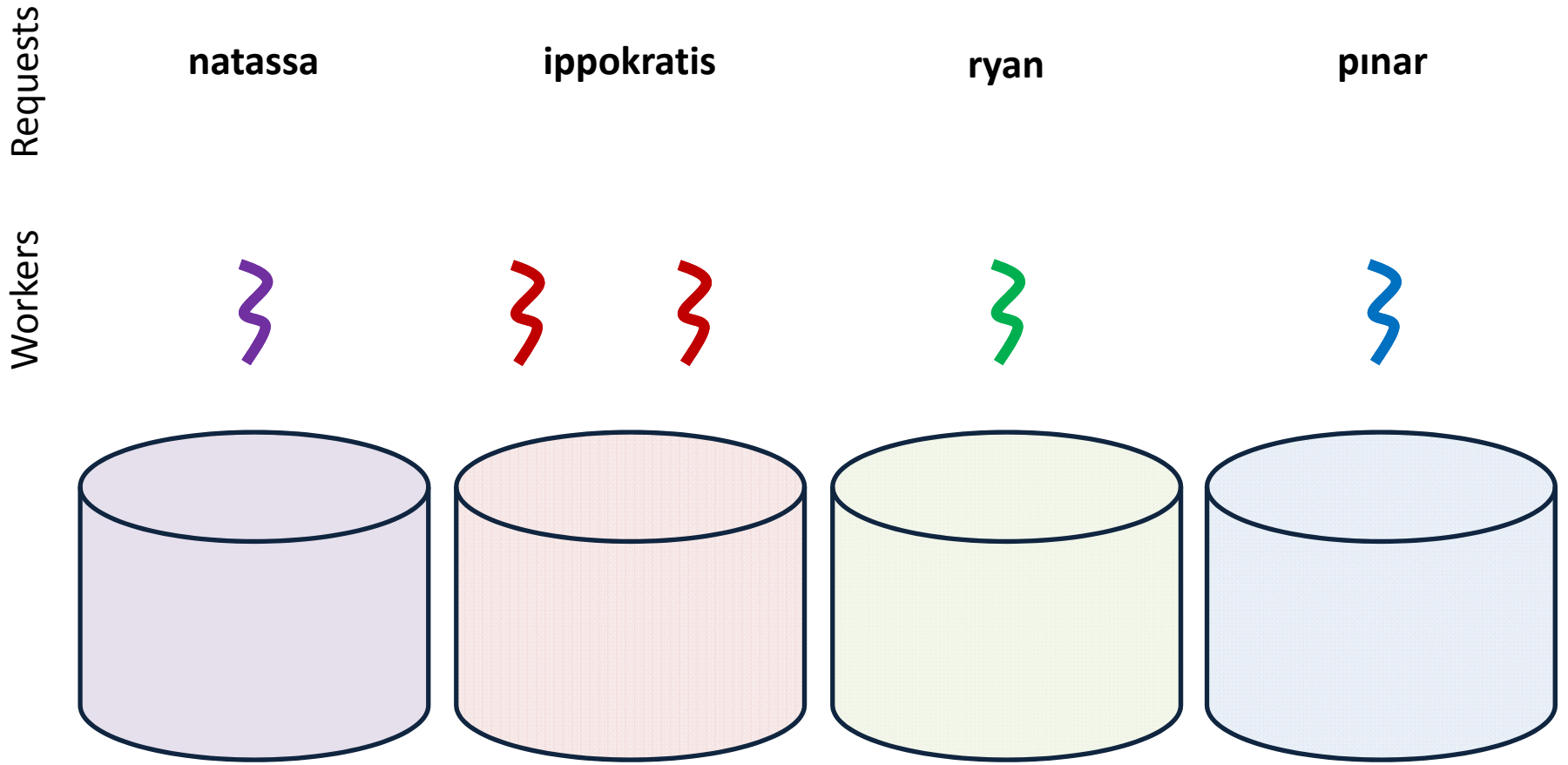


Shared-Everything



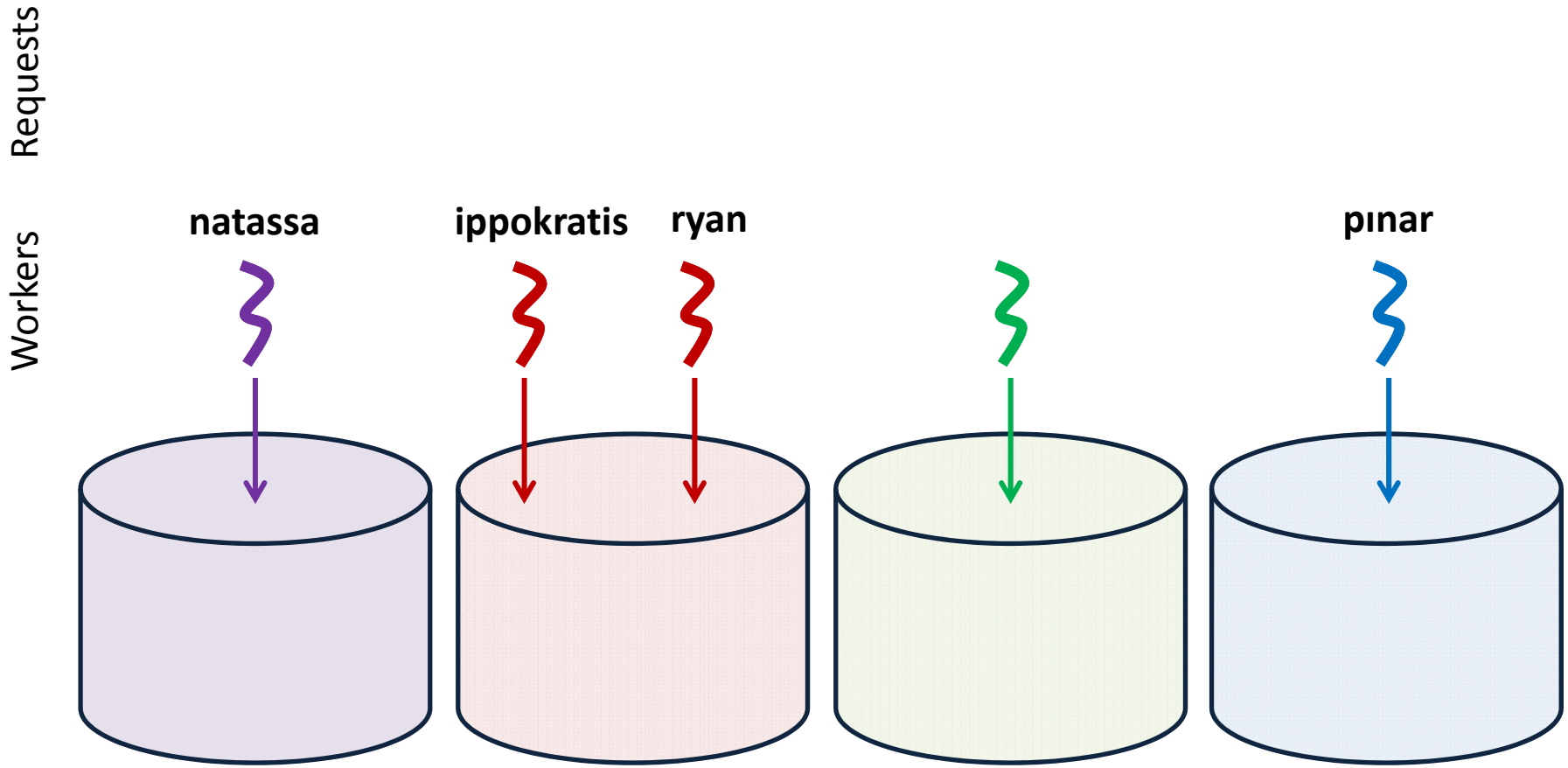
- Contention on shared data objects

Shared-Nothing – Physically Partitioned



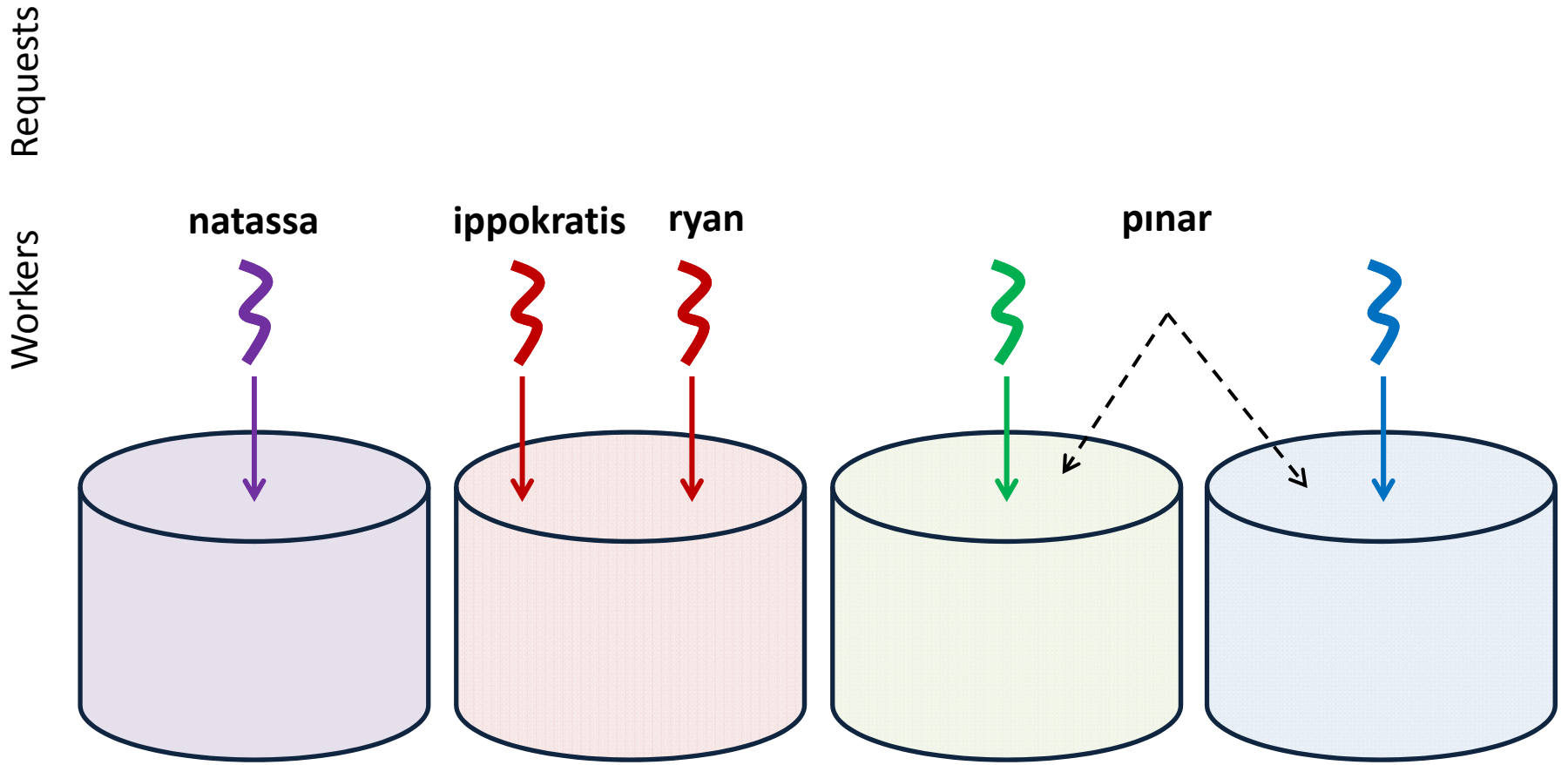
Shared-Nothing – Physically Partitioned

✓ Explicit contention control



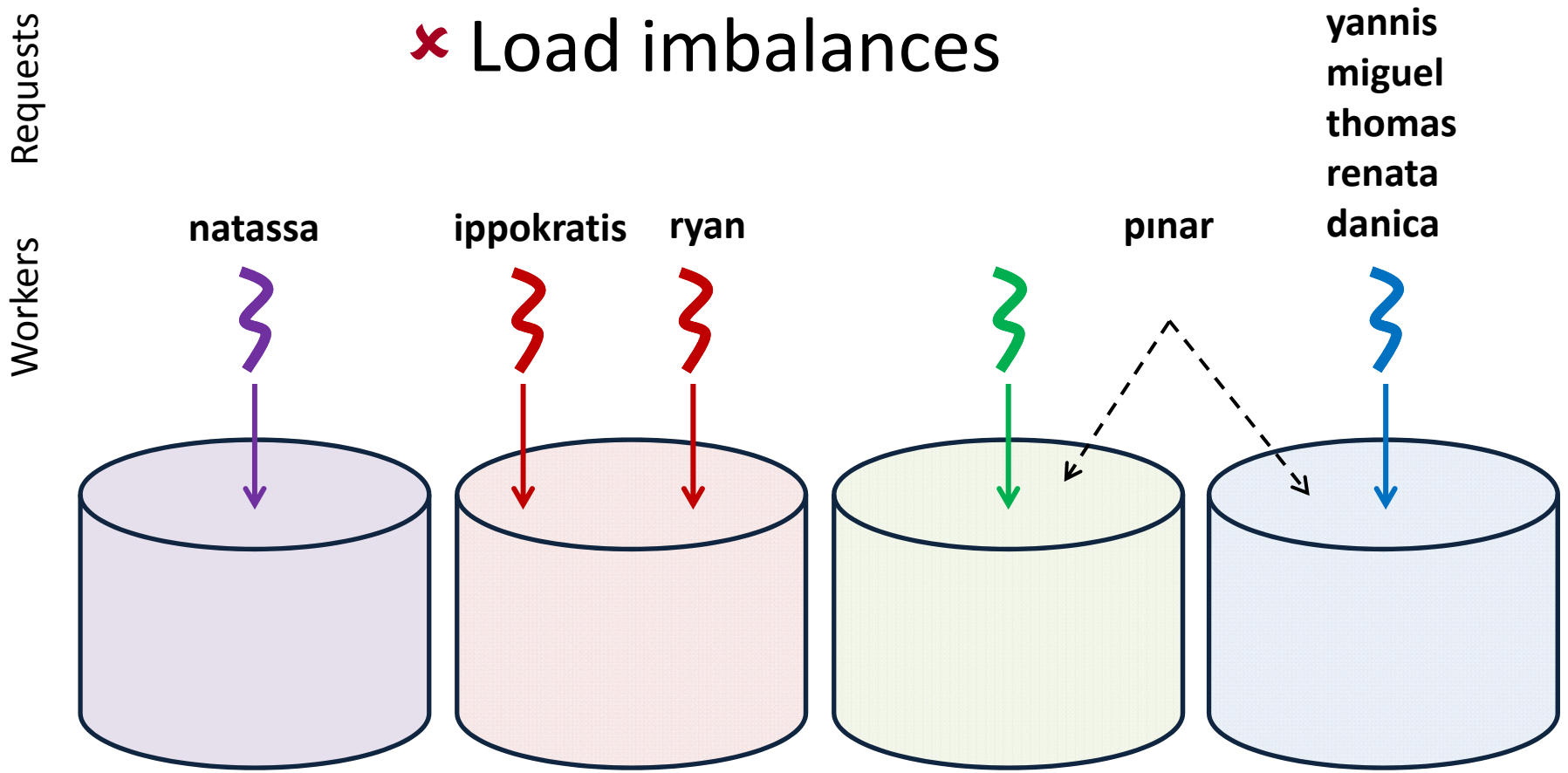
Shared-Nothing – Physically Partitioned

- ✓ Explicit contention control
- ✗ Distributed transactions



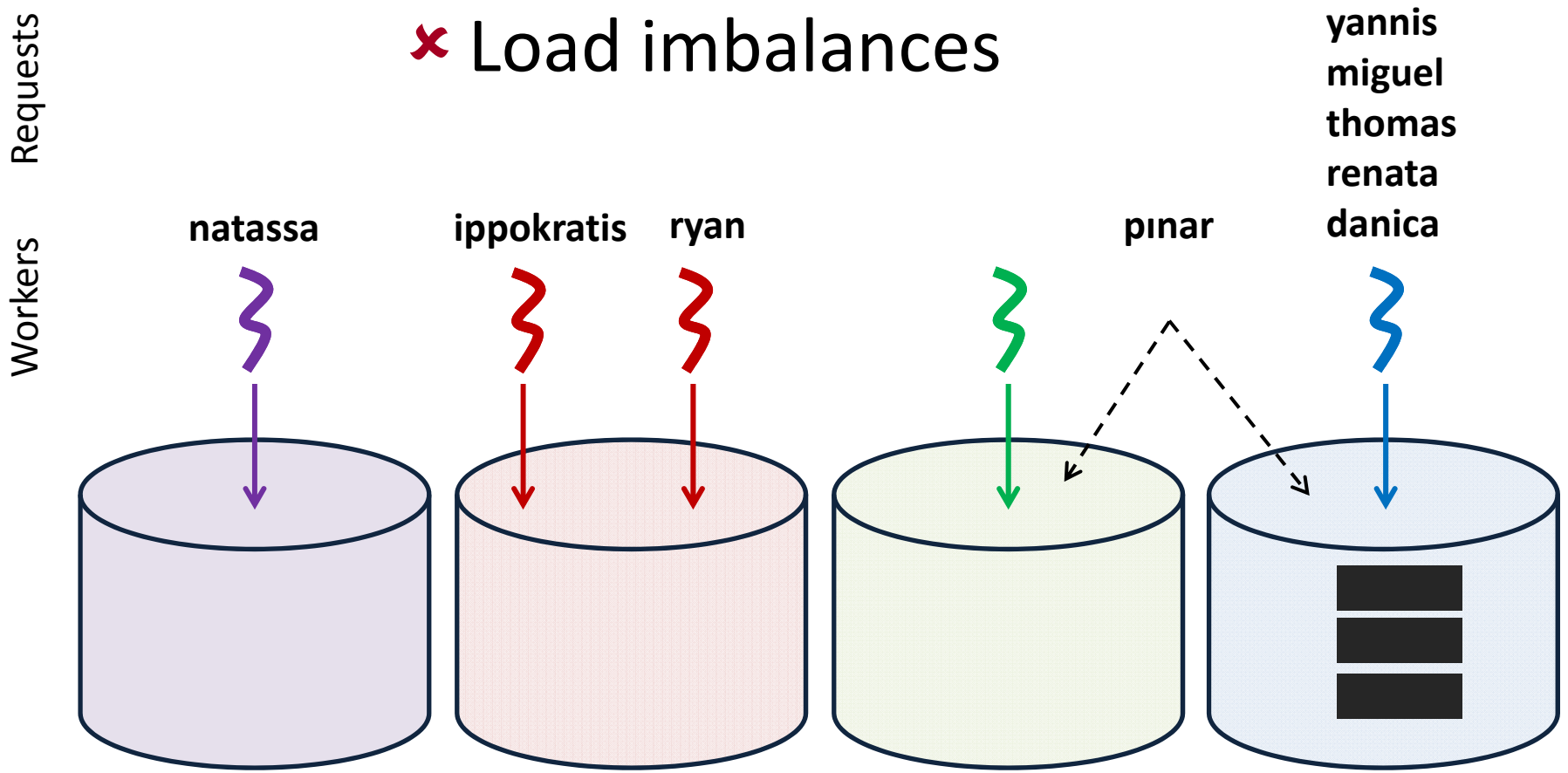
Shared-Nothing – Physically Partitioned

- ✓ Explicit contention control
- ✗ Distributed transactions
- ✗ Load imbalances



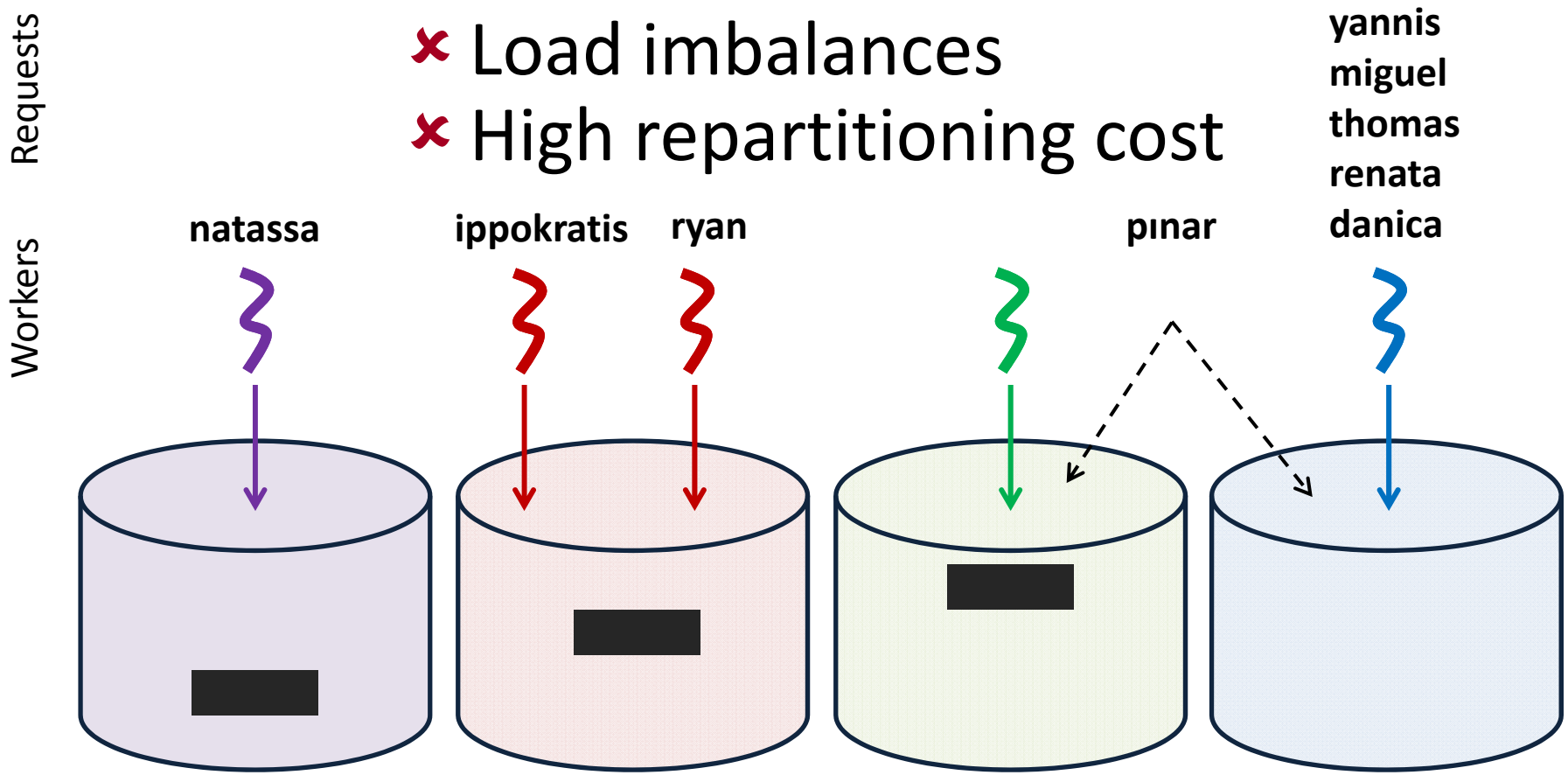
Shared-Nothing – Physically Partitioned

- ✓ Explicit contention control
- ✗ Distributed transactions
- ✗ Load imbalances



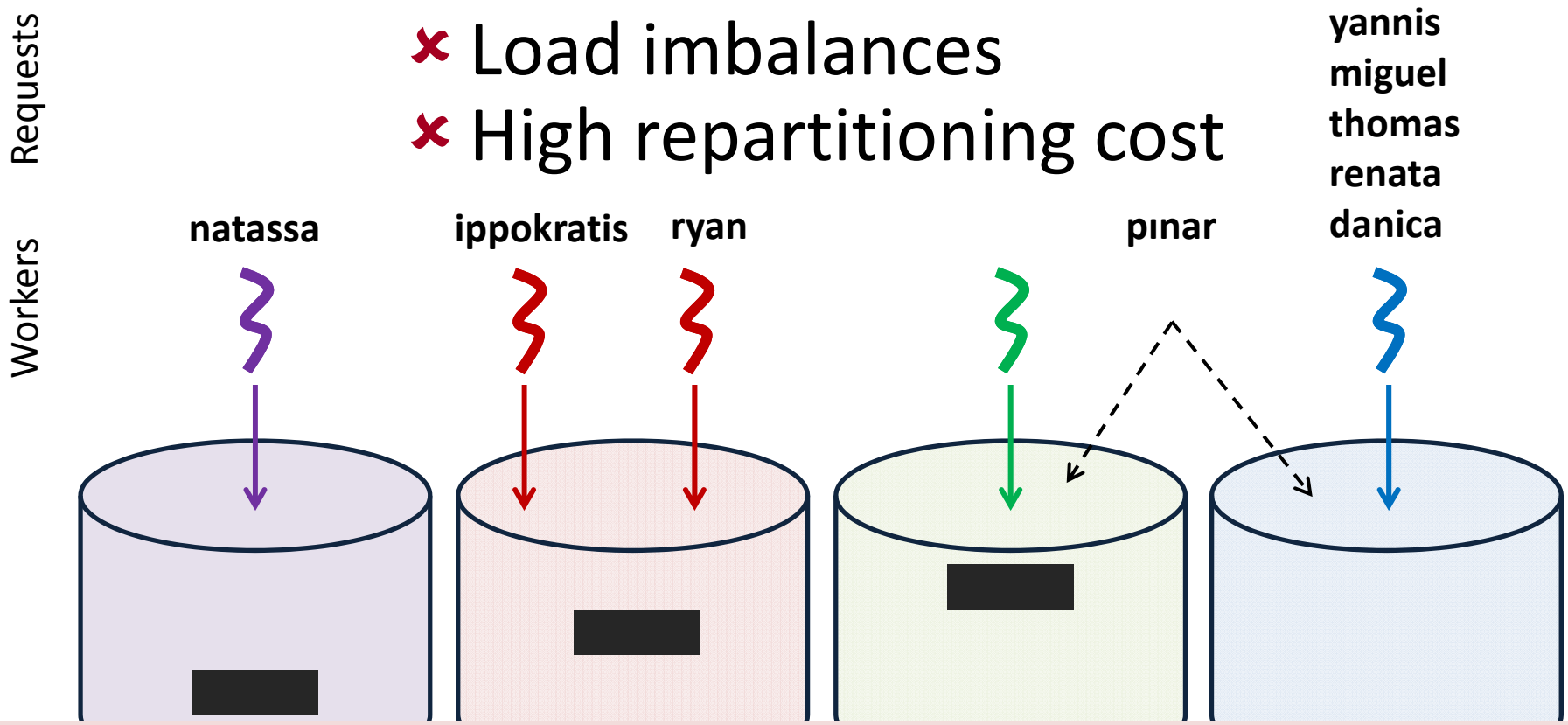
Shared-Nothing – Physically Partitioned

- ✓ Explicit contention control
- ✗ Distributed transactions
- ✗ Load imbalances
- ✗ High repartitioning cost



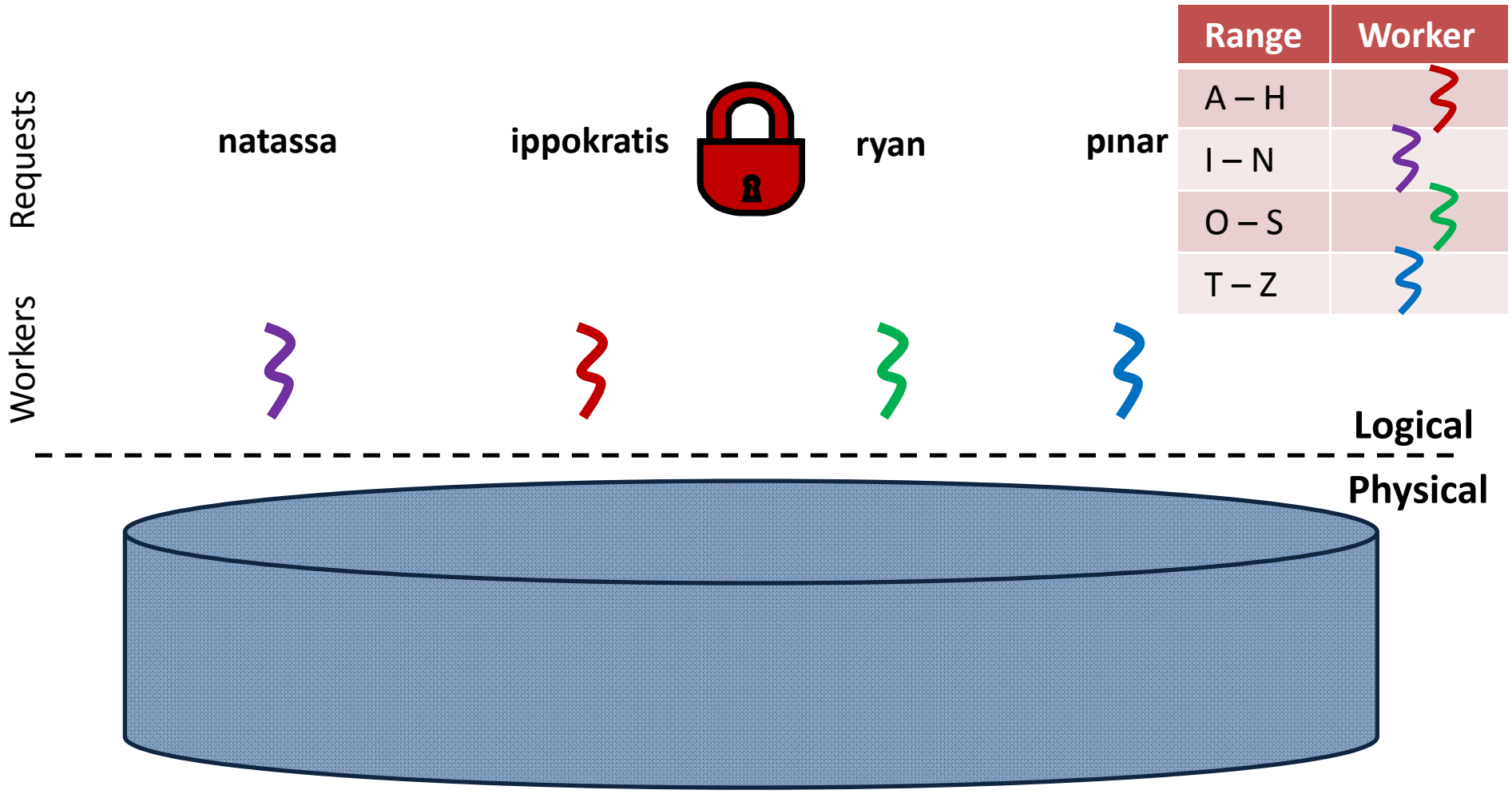
Shared-Nothing – Physically Partitioned

- ✓ Explicit contention control
- ✗ Distributed transactions
- ✗ Load imbalances
- ✗ High repartitioning cost

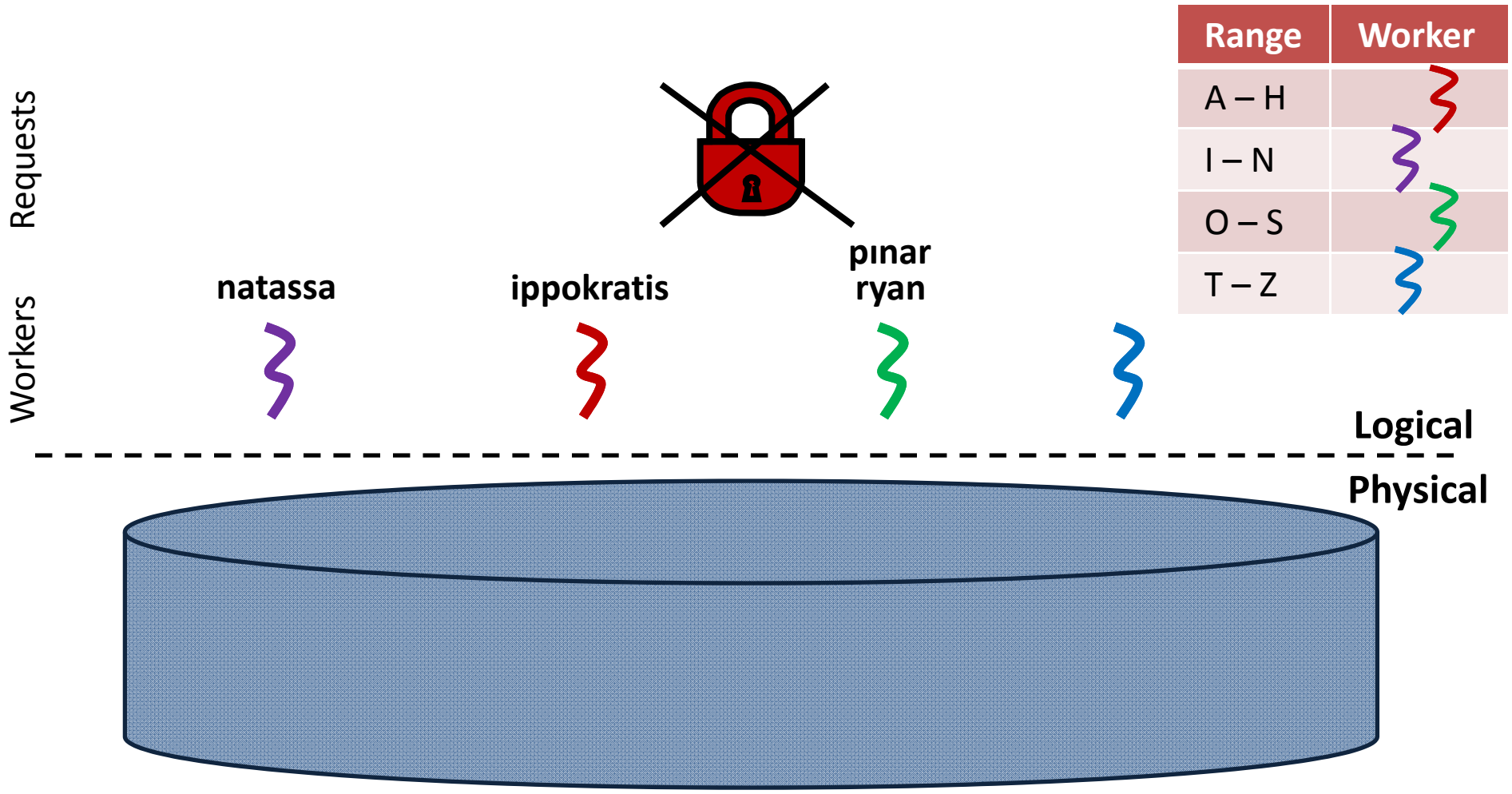


Great for some workloads, not all

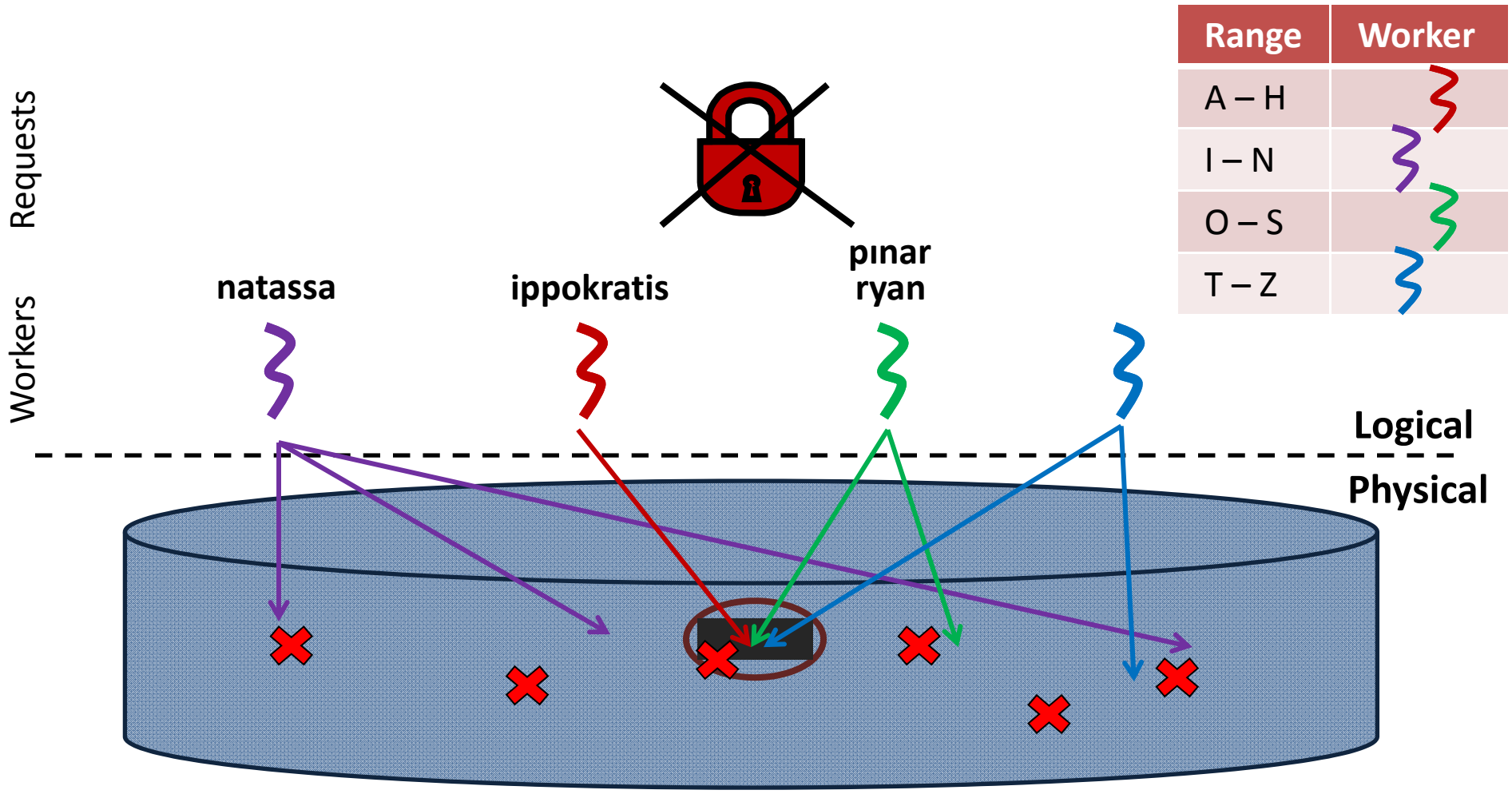
Logically Partitioned



Logically Partitioned

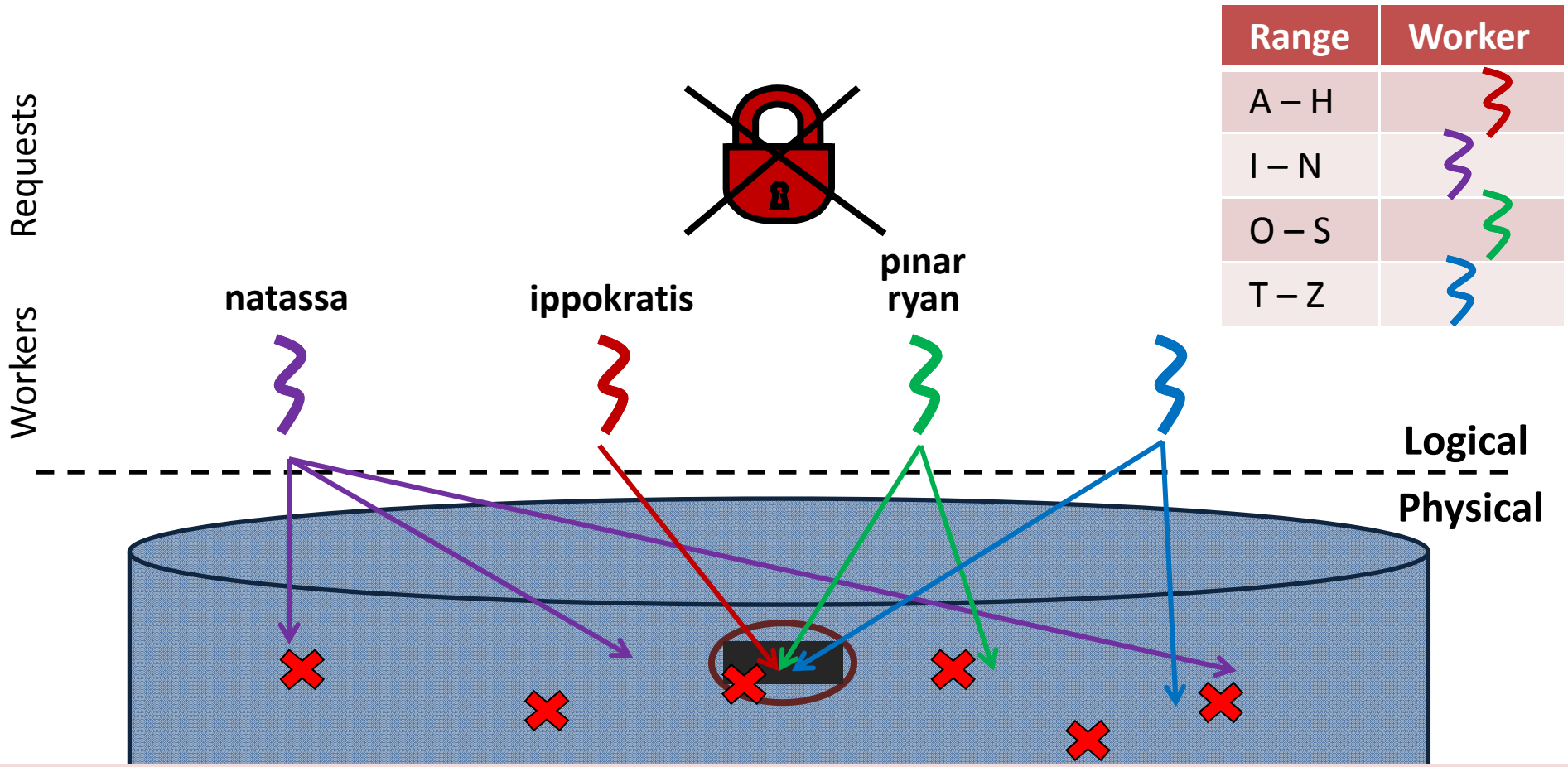


Logically Partitioned



Range	Worker
A – H	
I – N	
O – S	
T – Z	

Logically Partitioned

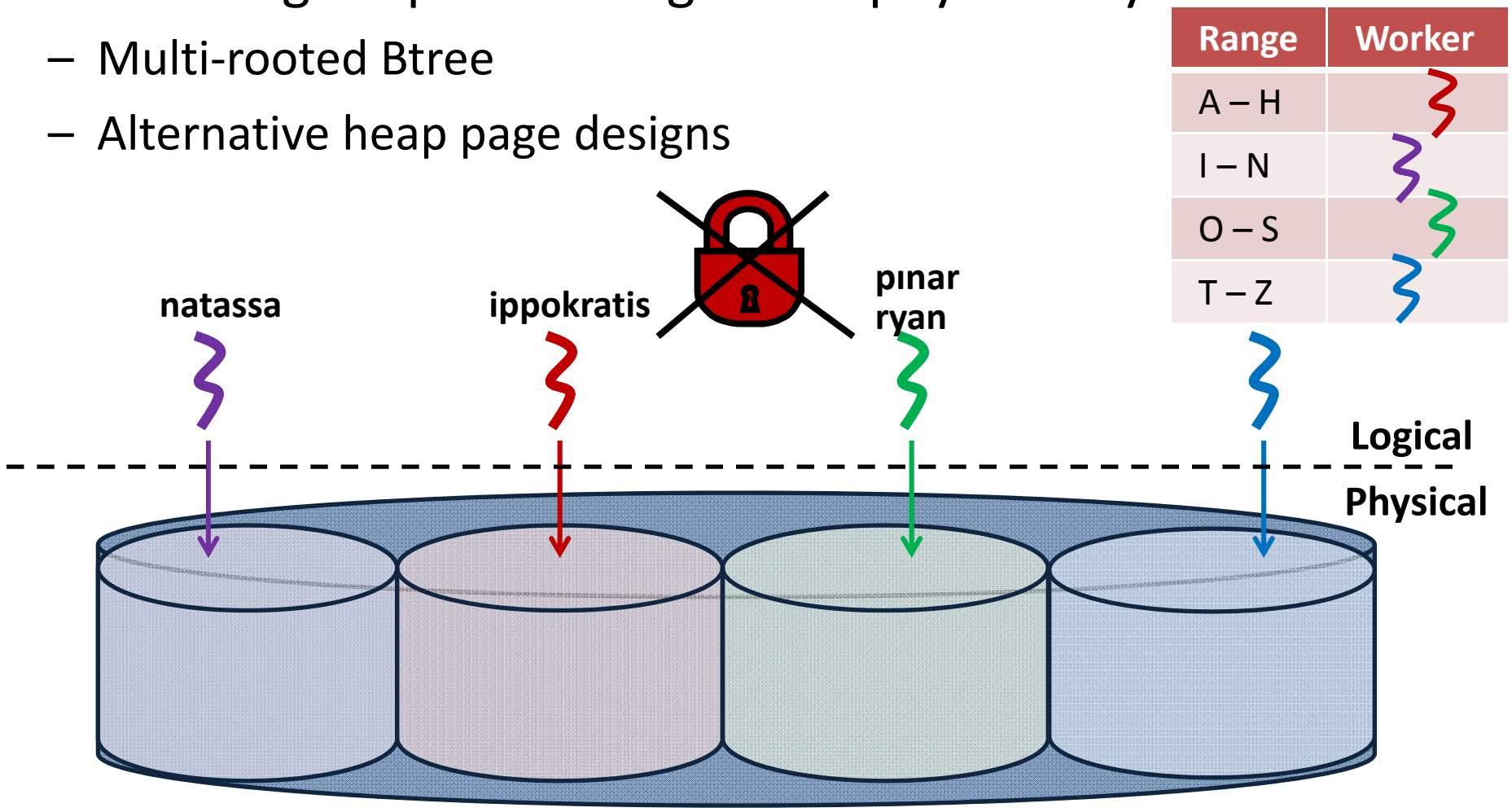


Range	Worker
A – H	
I – N	
O – S	
T – Z	

Contention at the **physical** layer

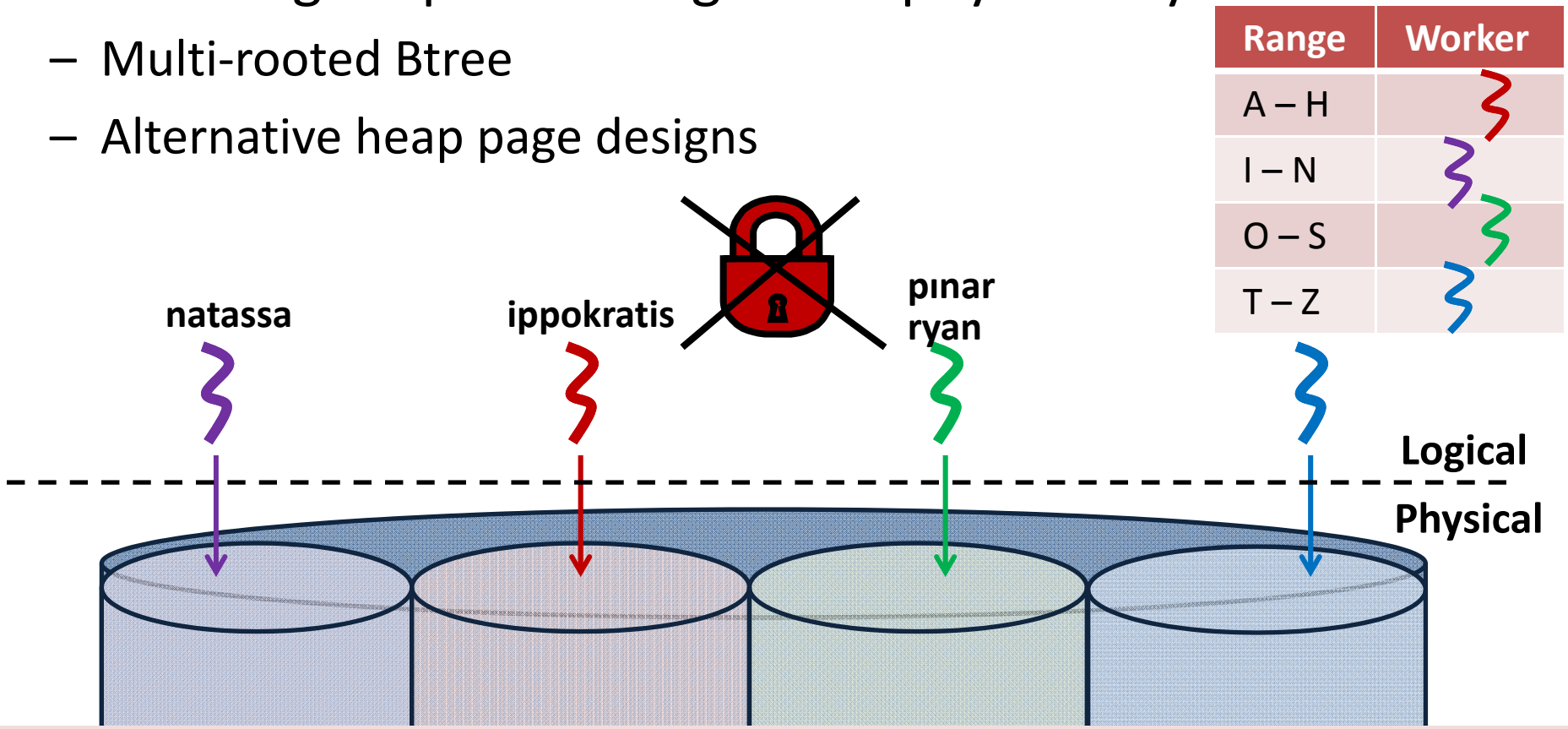
Physiological Partitioning

- Extends logical partitioning at the physical layer
  - Multi-rooted Btree
  - Alternative heap page designs



Physiological Partitioning

- Extends logical partitioning at the physical layer
  - Multi-rooted Btree
  - Alternative heap page designs



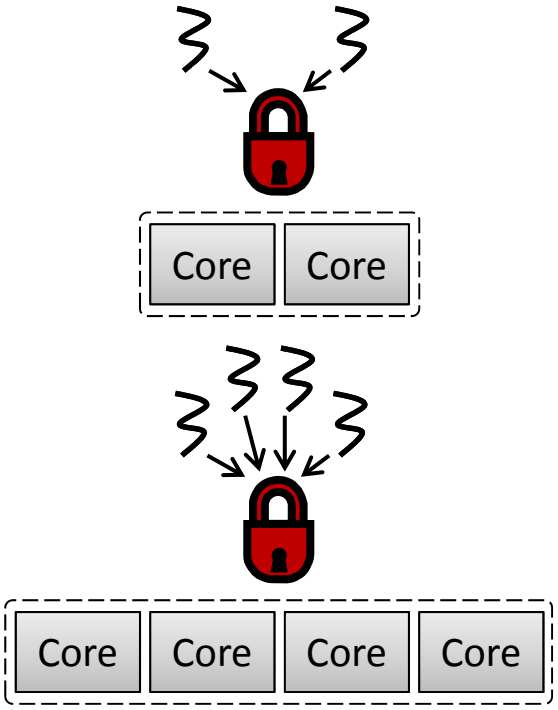
- ✓ Contention eliminated at both logical & physical layers
- ✓ Fast repartitioning

Outline

- Introduction
- **Types of Critical Sections**
- Physiological Partitioning (PLP)
- Results
- Conclusion

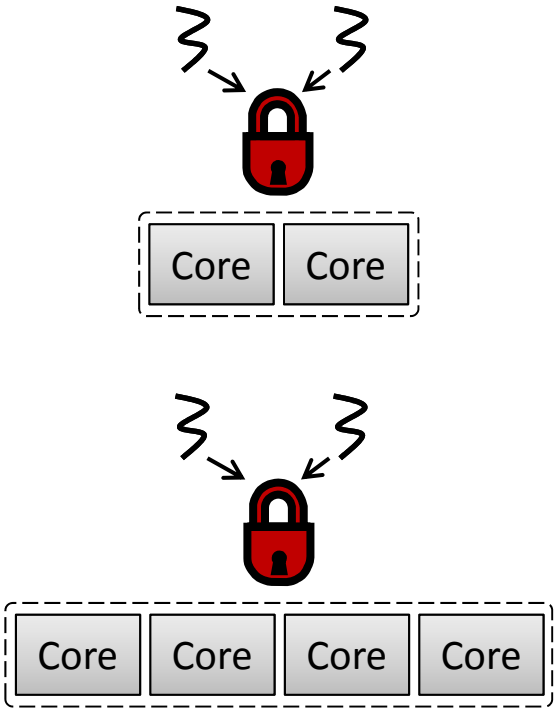
Critical Sections

Unscalable



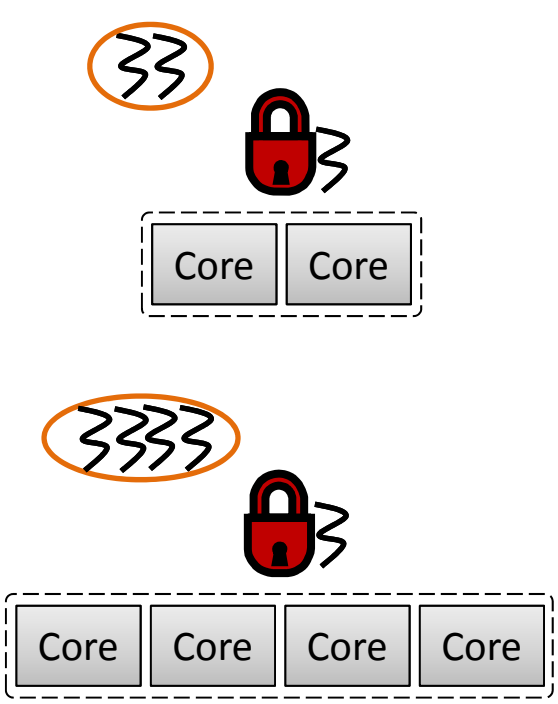
Locking, Latching

Fixed



Point-to-point communication

Composable



Logging

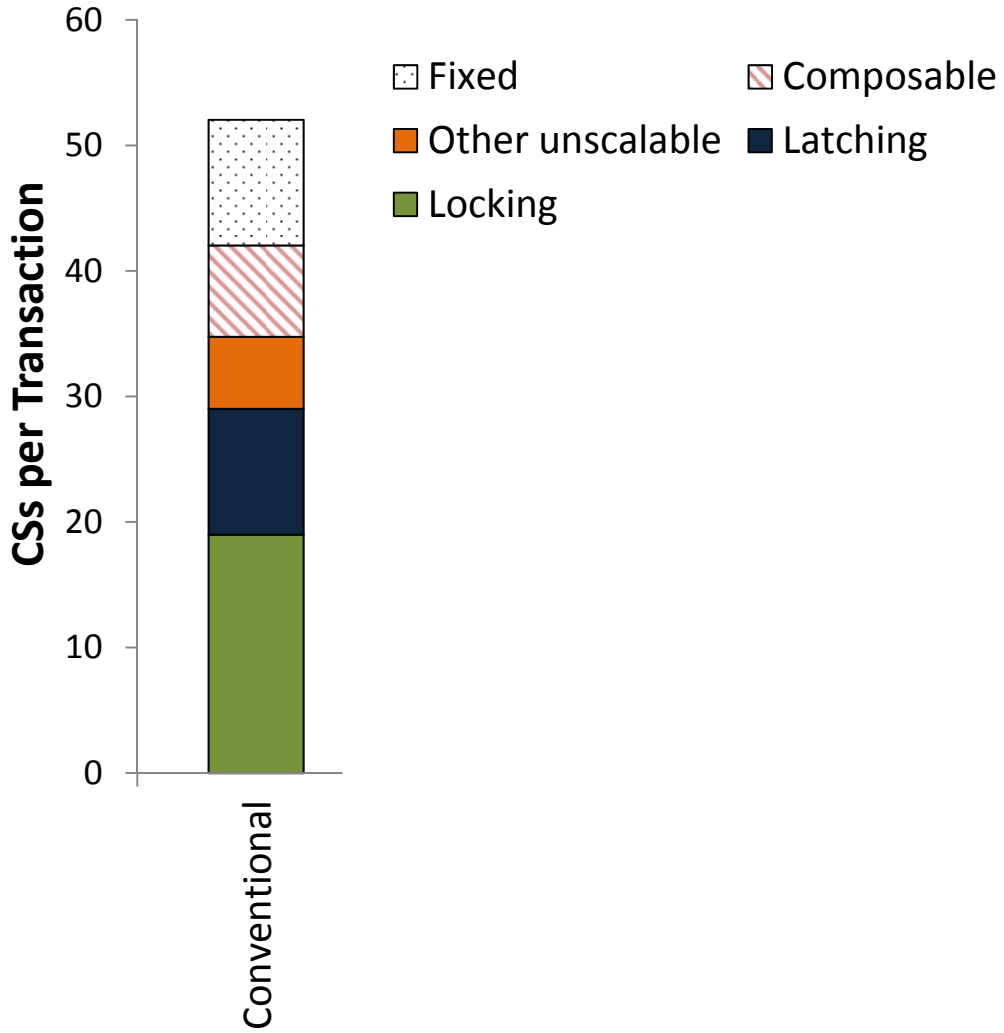
Unscalable → Fixed / Composable

Breakdown of the Critical Sections

…and its impact on performance

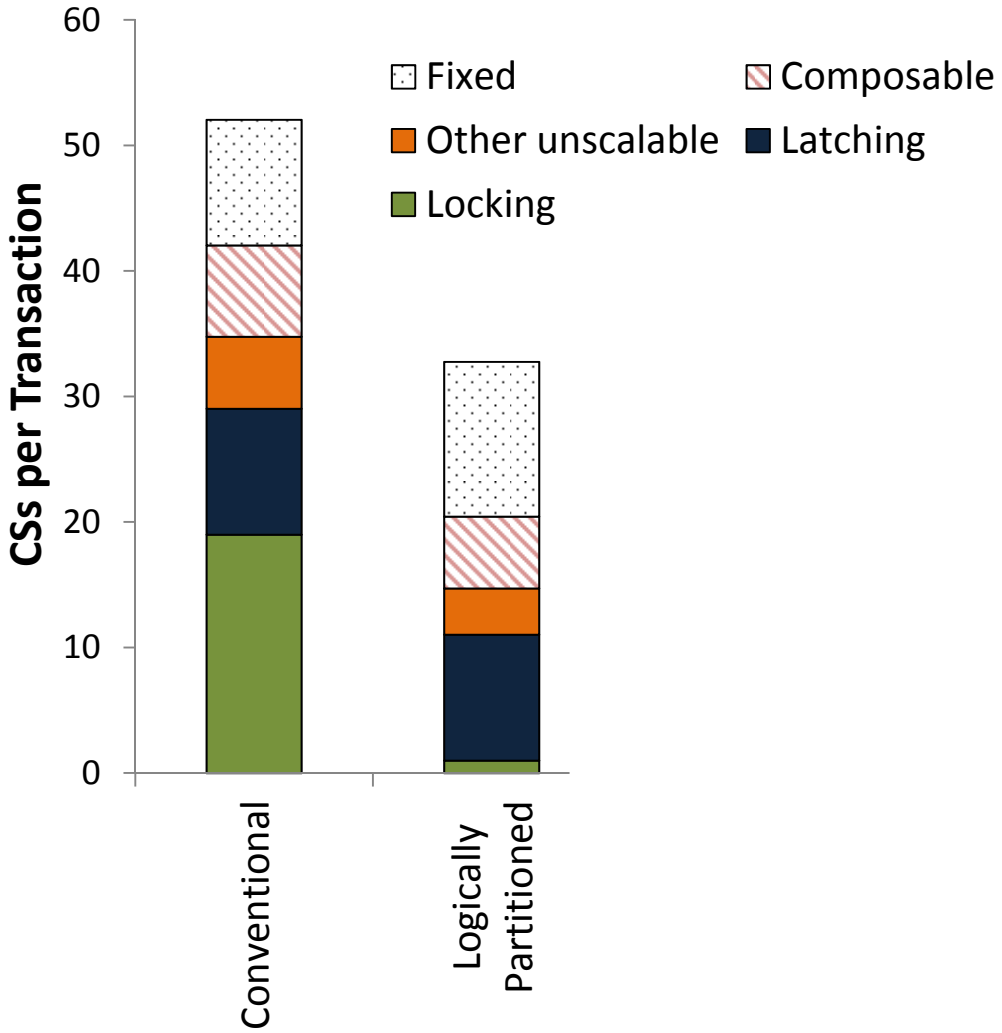
Probe one customer, update balance

4 socket Quad AMD



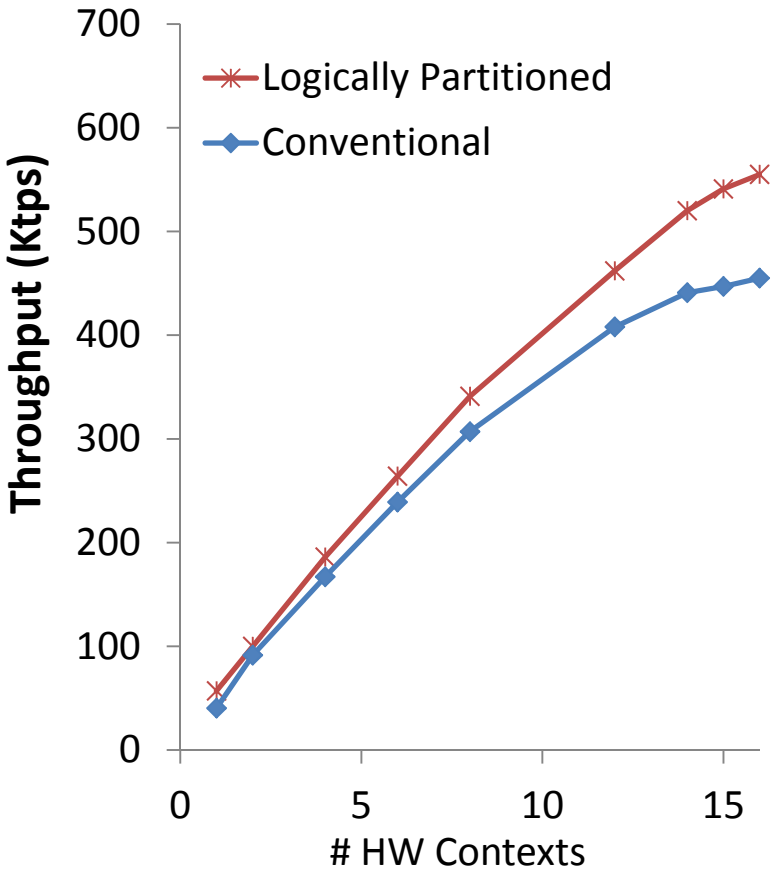
Breakdown of the Critical Sections

…and its impact on performance



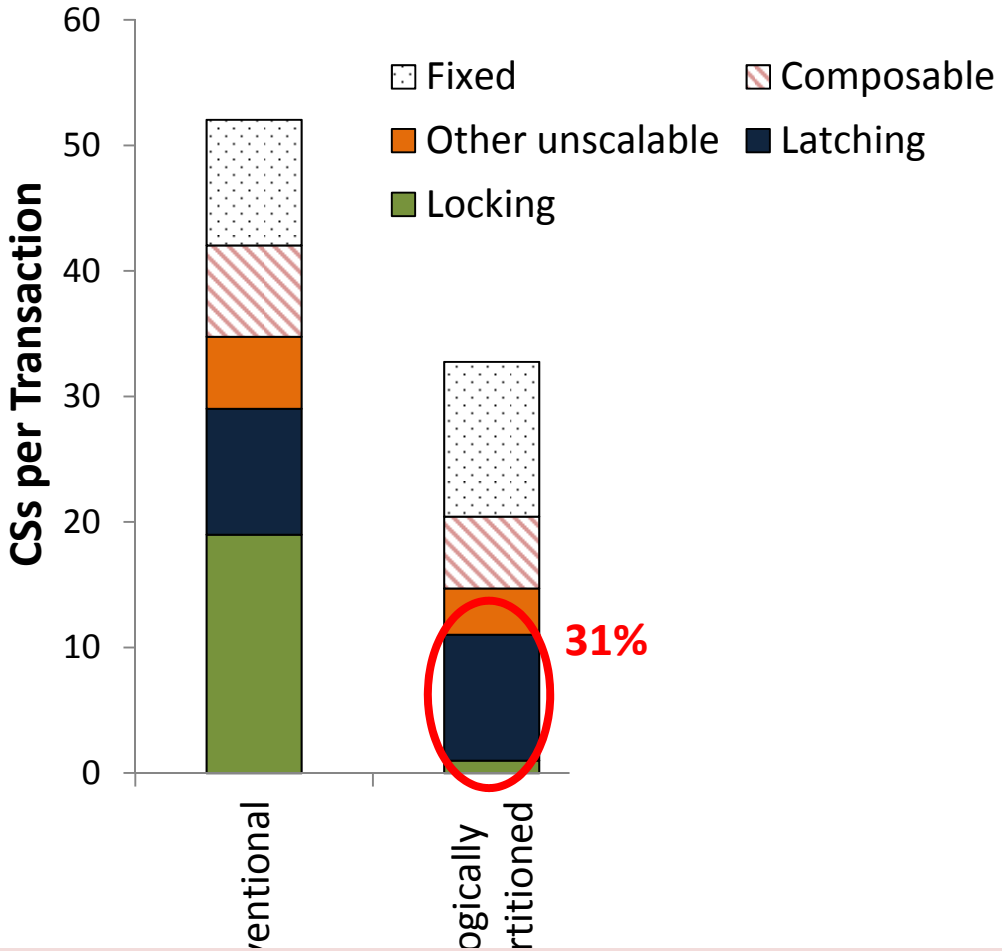
Probe one customer, update balance

4 socket Quad AMD



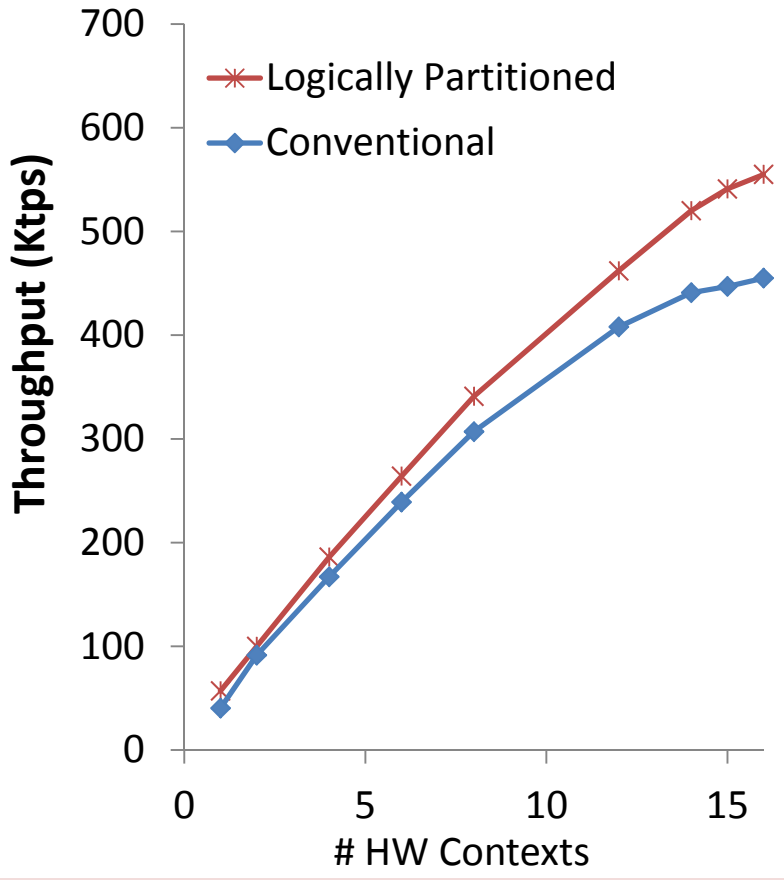
Breakdown of the Critical Sections

…and its impact on performance



Probe one customer, update balance

4 socket Quad AMD



Latching related CSs remain with logical-partitioning

Outline

- Introduction
- Types of Critical Sections
- **Physiological Partitioning (PLP)**
- Results
- Conclusion

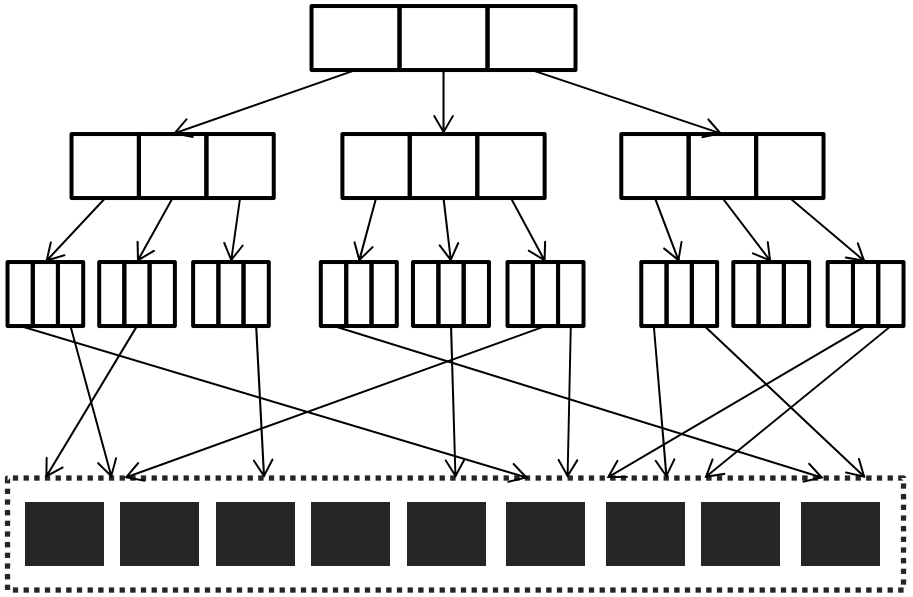
Physical Conflicts

Range	Worker
A – M	
N – Z	

Logical

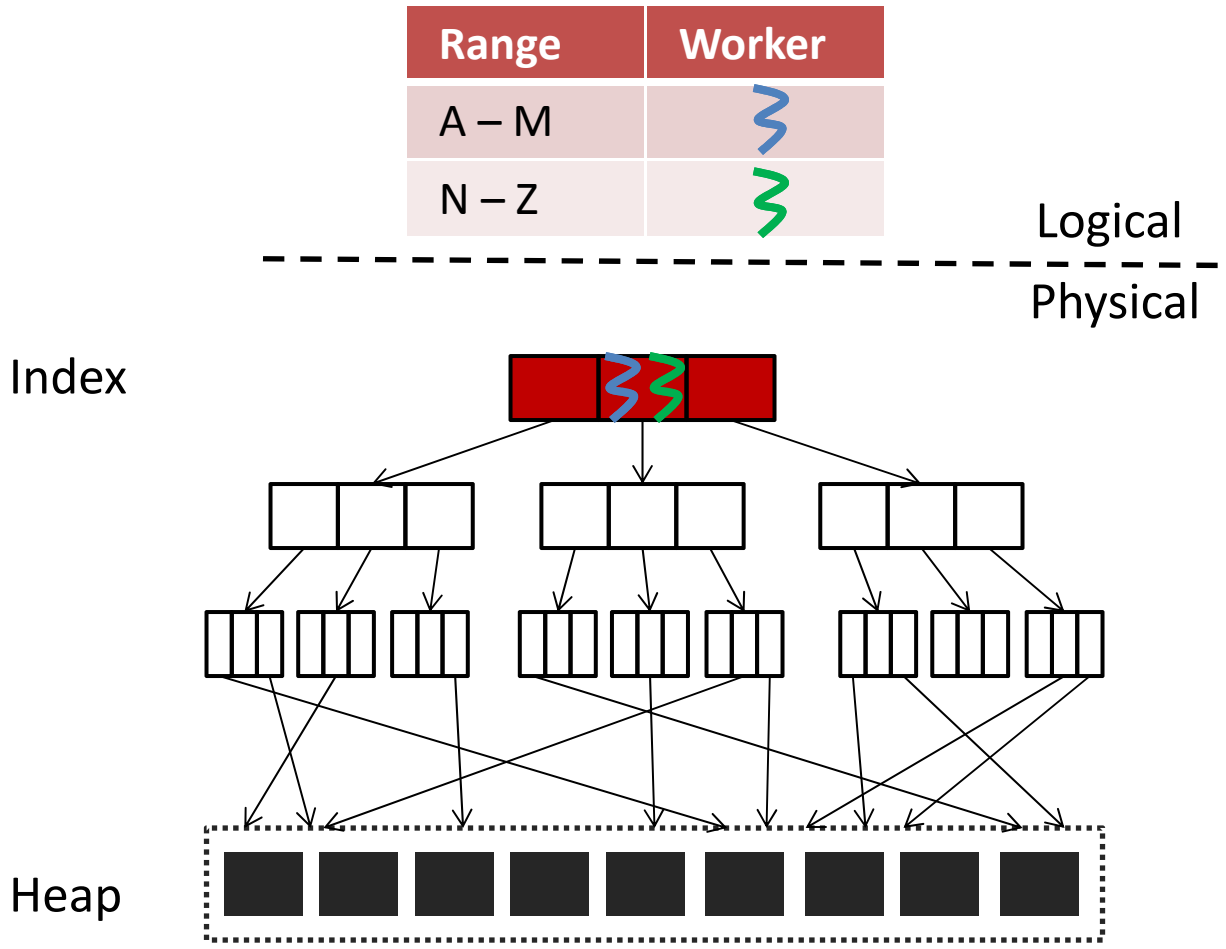
Physical

Index



Heap

Physical Conflicts



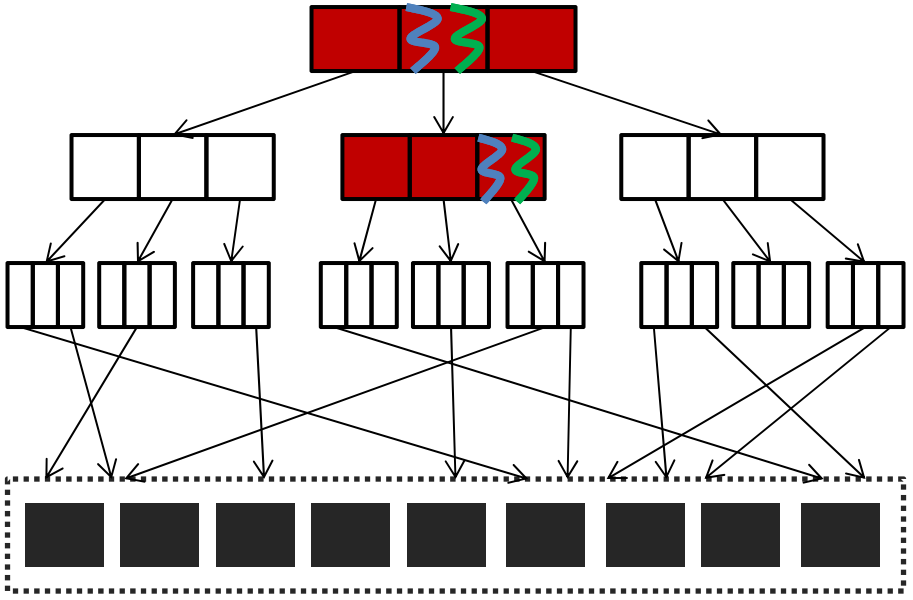
Physical Conflicts

Range	Worker
A – M	
N – Z	

Logical

Physical

Index



Heap

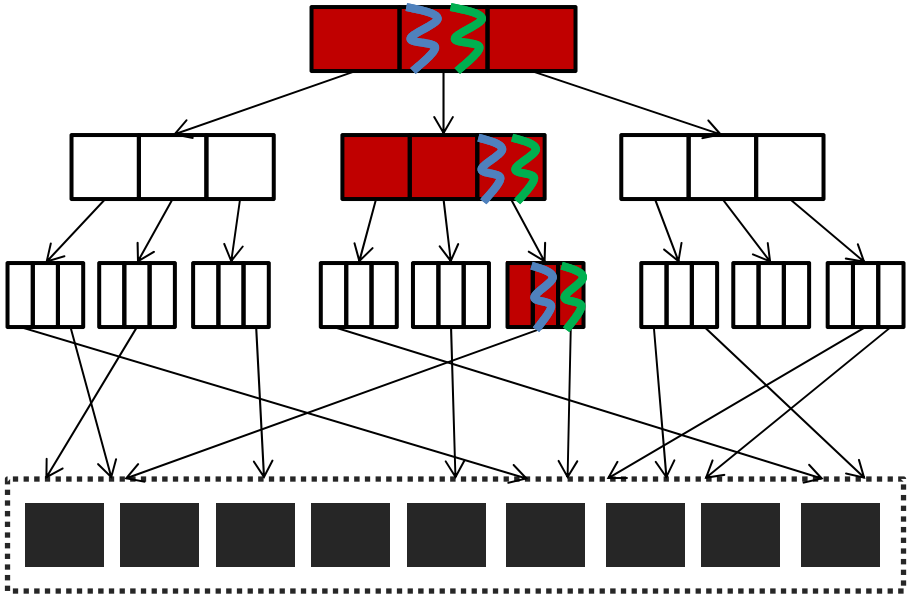
Physical Conflicts

Range	Worker
A – M	
N – Z	

Logical

Physical

Index



Heap

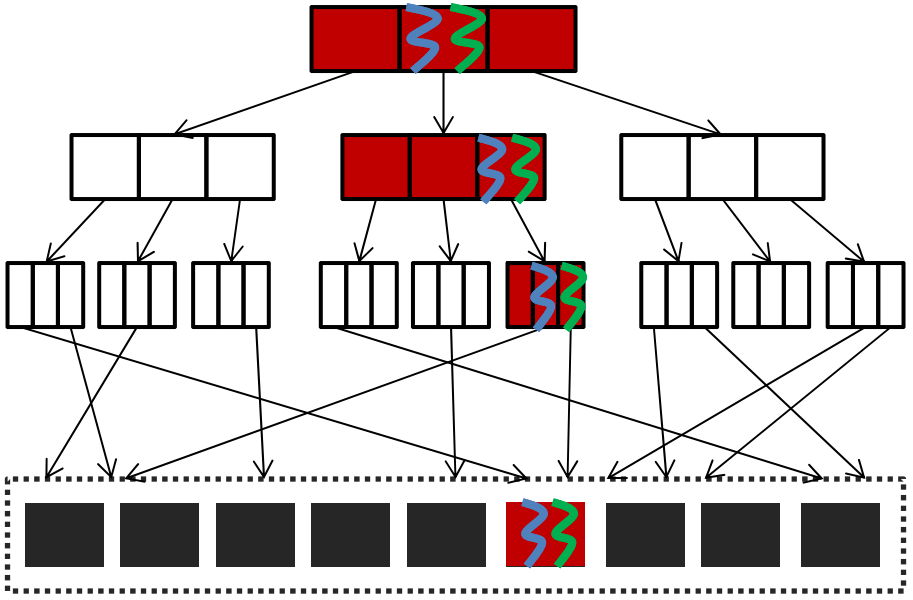
Physical Conflicts

Range	Worker
A – M	
N – Z	

Logical

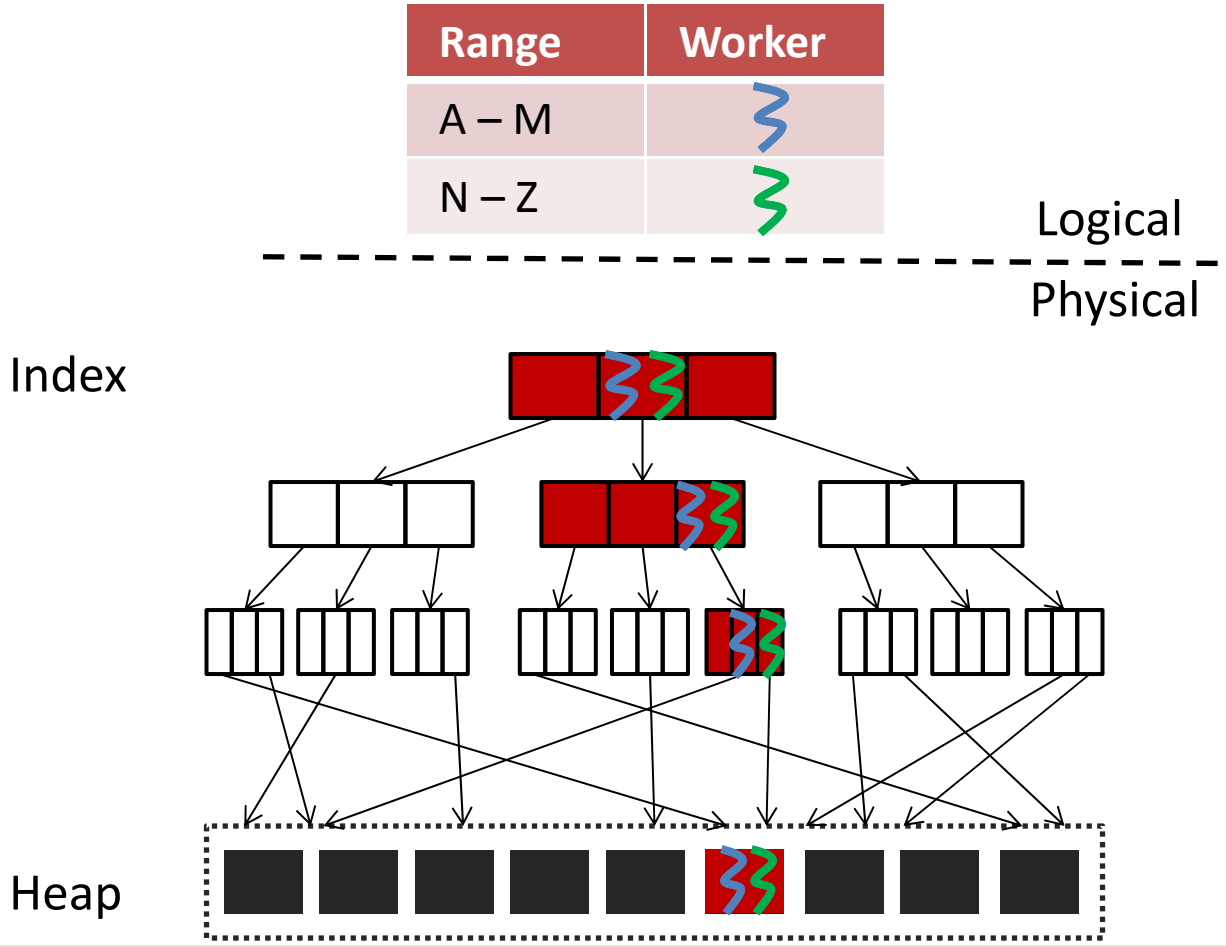
Physical

Index



Heap

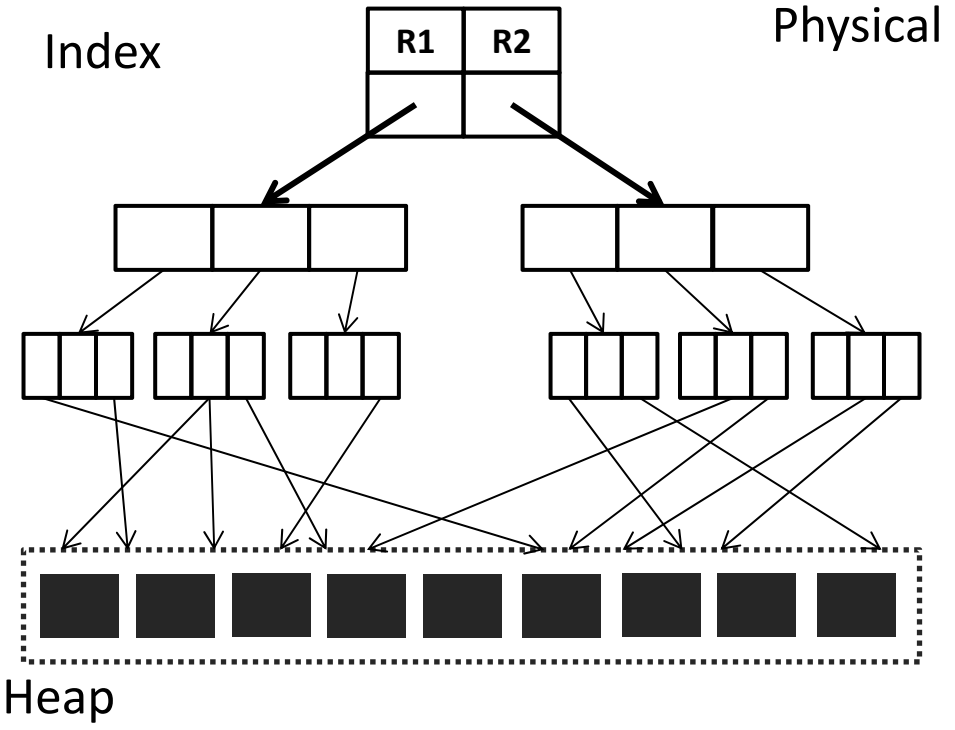
Physical Conflicts



Conflicts on both index & heap pages

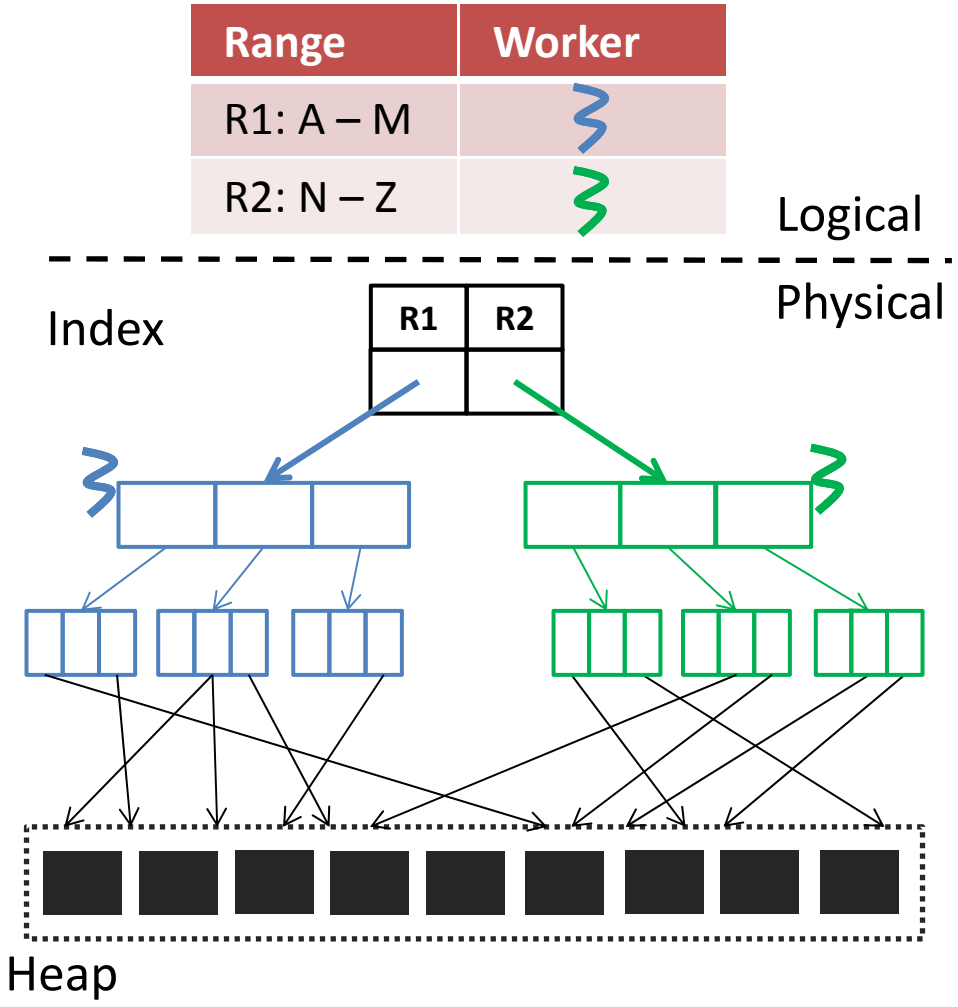
Physiological Partitioning (PLP)

Range	Worker
R1: A – M	
R2: N – Z	



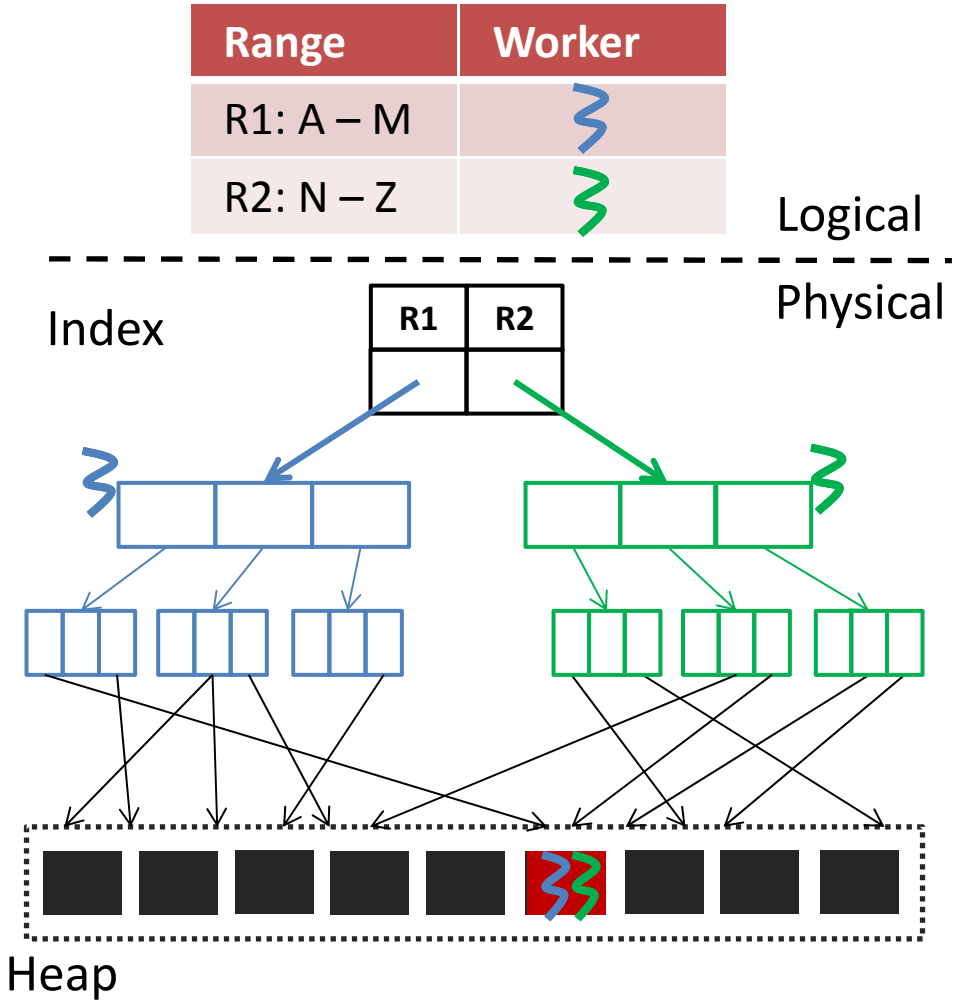
- Multi-rooted Btree
  - Routing table is the root

Physiological Partitioning (PLP)



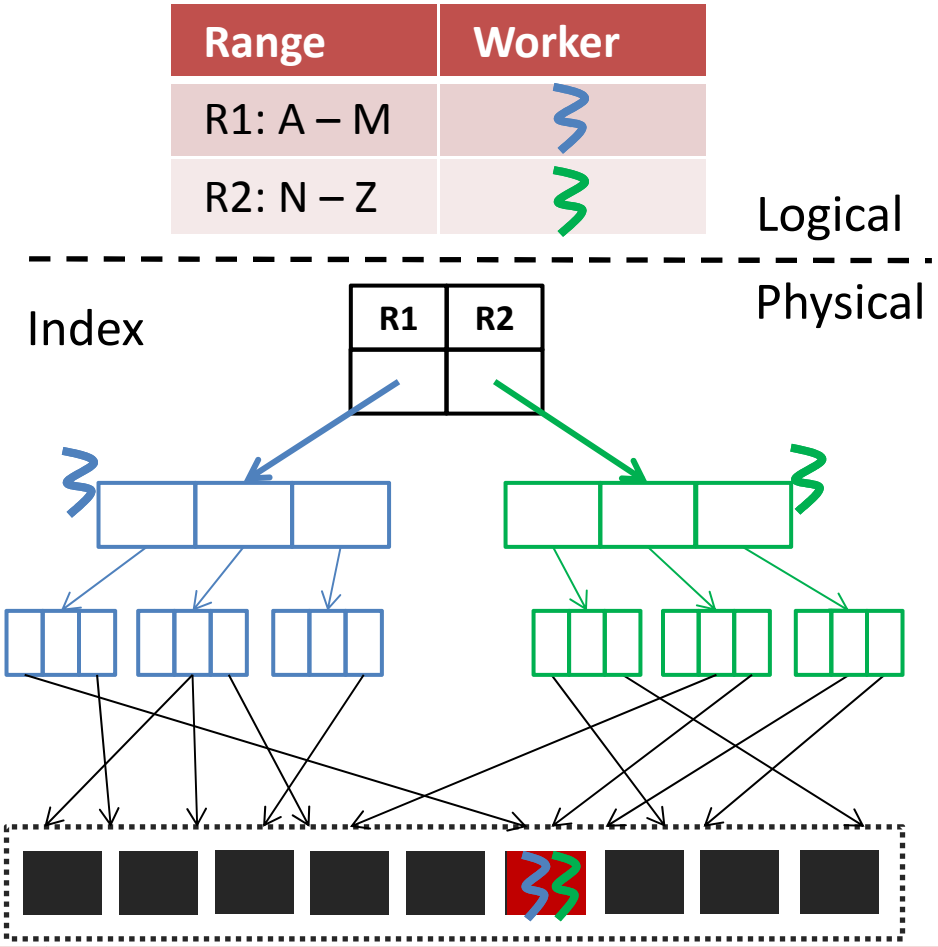
- Multi-rooted Btree
  - Routing table is the root
- ✓ Both logical & physical partitioning
- ✓ Reduces contention on index root
- ✓ Parallel structure modification operations
- ✓ Fast index repartitioning

Physiological Partitioning (PLP)



- Multi-rooted Btree
  - Routing table is the root
- ✓ Both logical & physical partitioning
- ✓ Reduces contention on index root
- ✓ Parallel structure modification operations
- ✓ Fast index repartitioning

Physiological Partitioning (PLP)



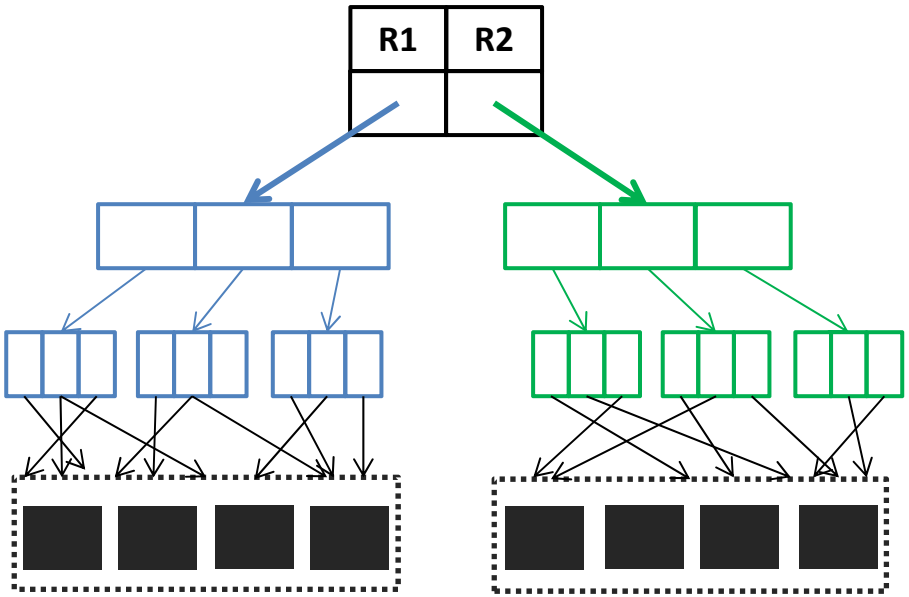
- Multi-rooted Btree
  - Routing table is the root
- ✓ Both logical & physical partitioning
- ✓ Reduces contention on index root
- ✓ Parallel structure modification operations
- ✓ Fast index repartitioning

✓ No need to latch index pages

✗ Still need to latch heap pages

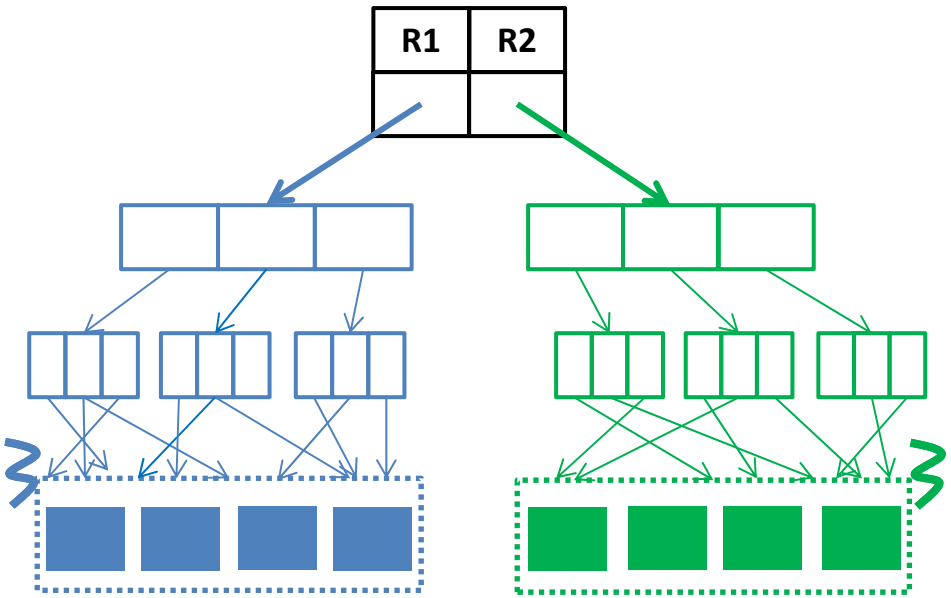
Heap Pages : Alternatives

PLP-Partition



Heap Pages : Alternatives

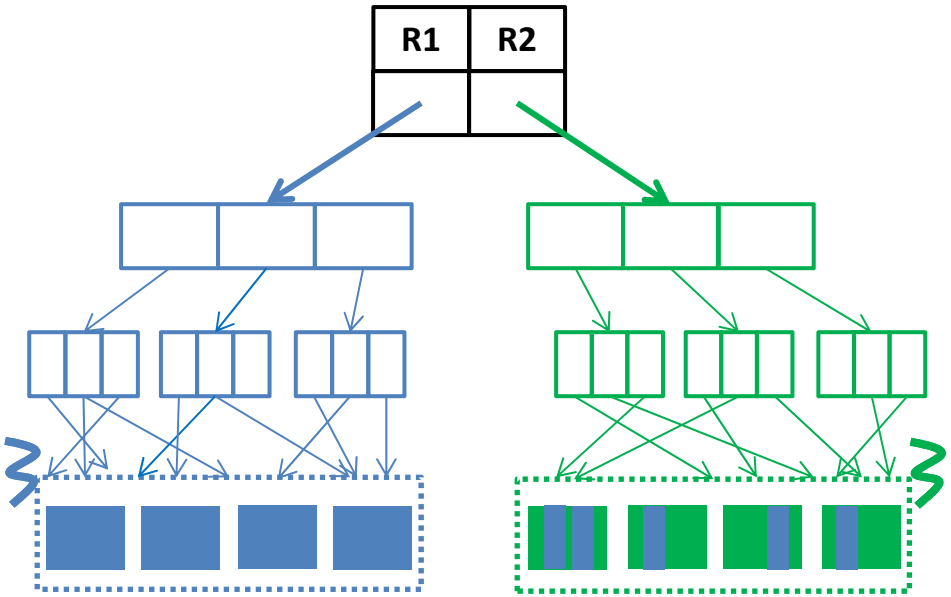
PLP-Partition



- Two-step record inserts
- Repartitioning worst-case: Scan entire partition

Heap Pages : Alternatives

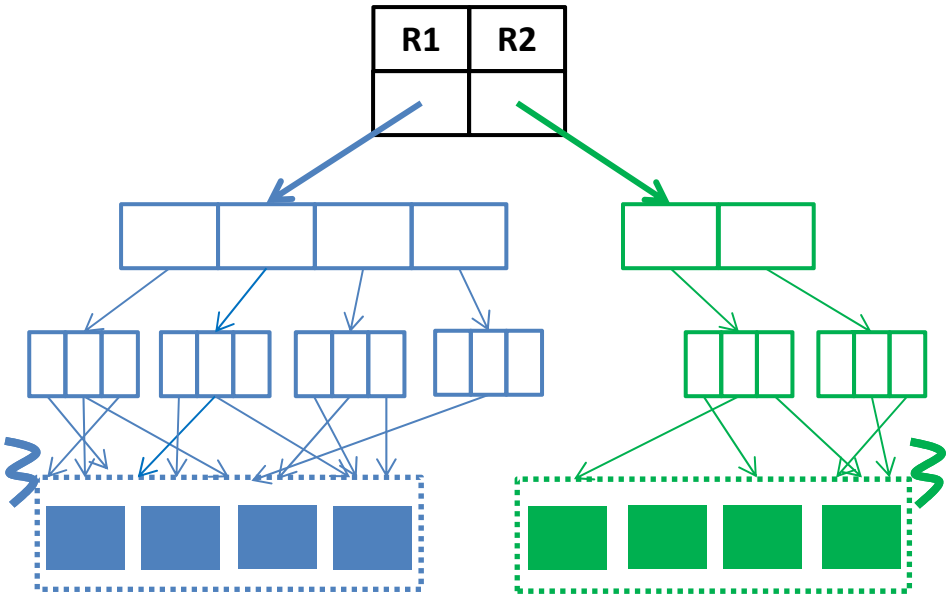
PLP-Partition



- Two-step record inserts
- Repartitioning worst-case: Scan entire partition

Heap Pages : Alternatives

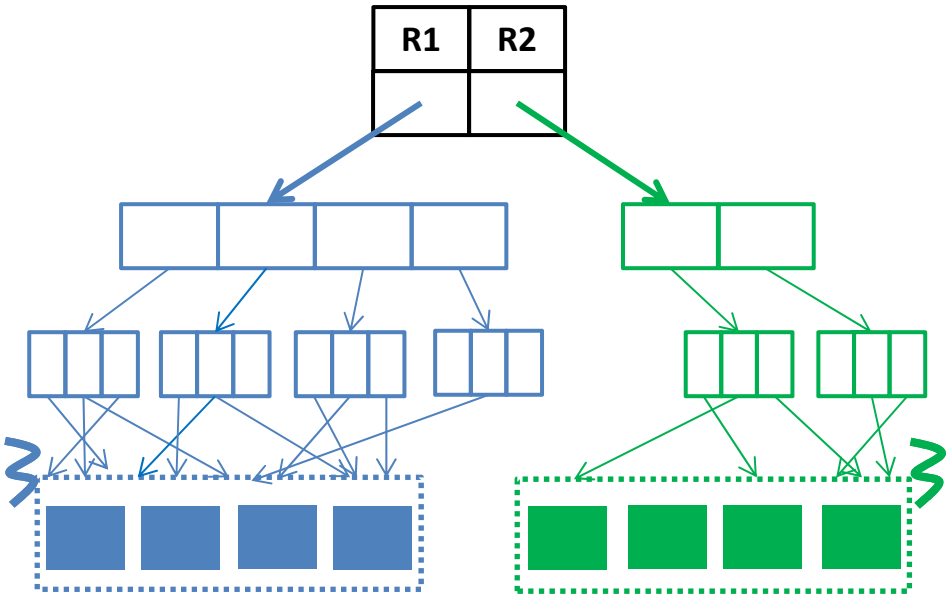
PLP-Partition



- Two-step record inserts
- Repartitioning worst-case: Scan entire partition

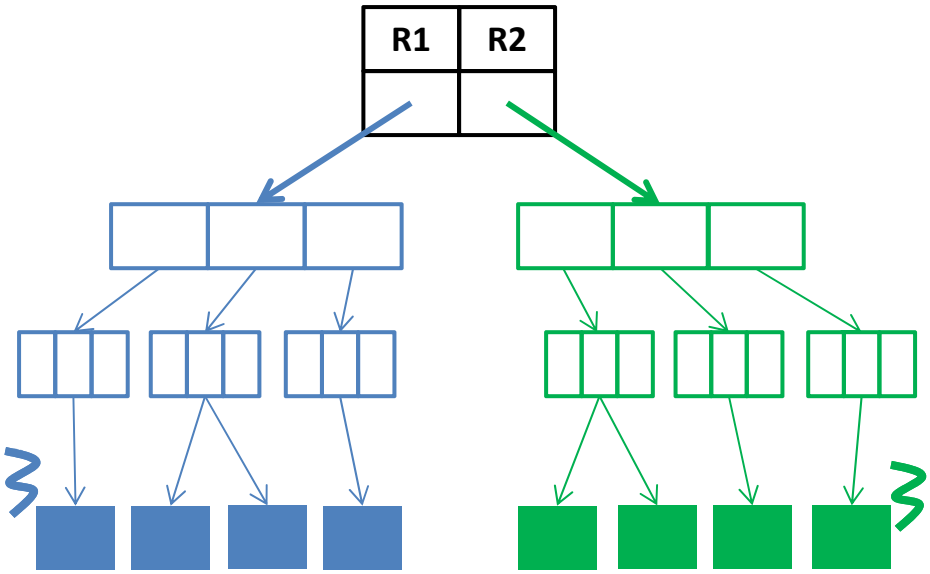
Heap Pages : Alternatives

PLP-Partition



- ✘ Two-step record inserts
- ✘ Repartitioning worst-case: Scan entire partition

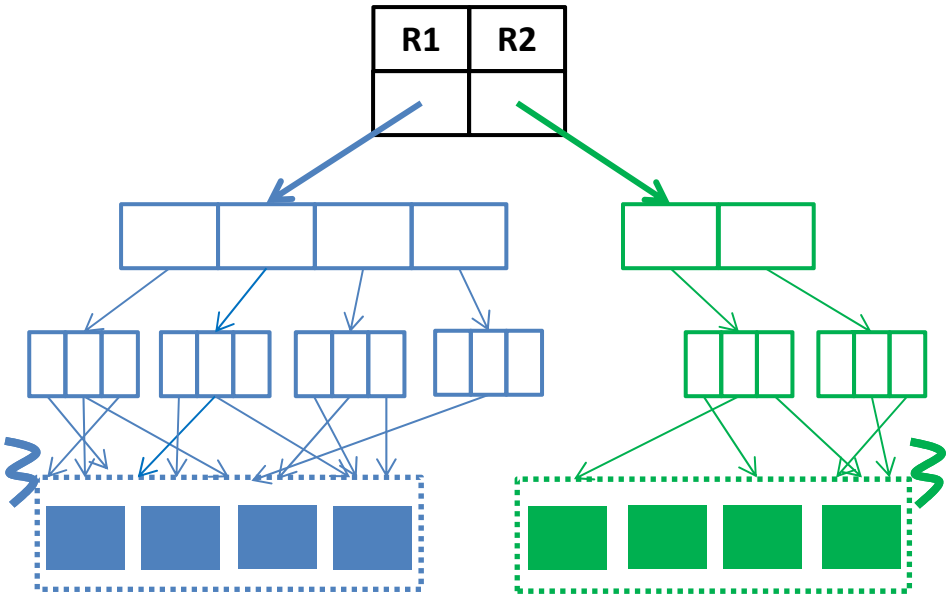
PLP-Leaf



- ✘ Two-step record inserts
- ✘ Fragmentation
- ✓ Repartitioning worst-case: Scan few pages

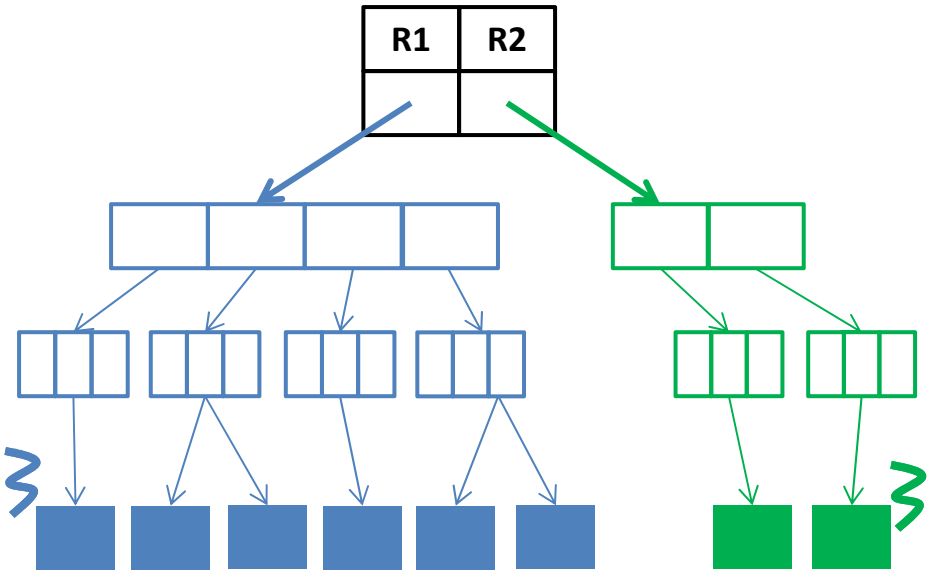
Heap Pages : Alternatives

PLP-Partition



- ✘ Two-step record inserts
- ✘ Repartitioning worst-case: Scan entire partition

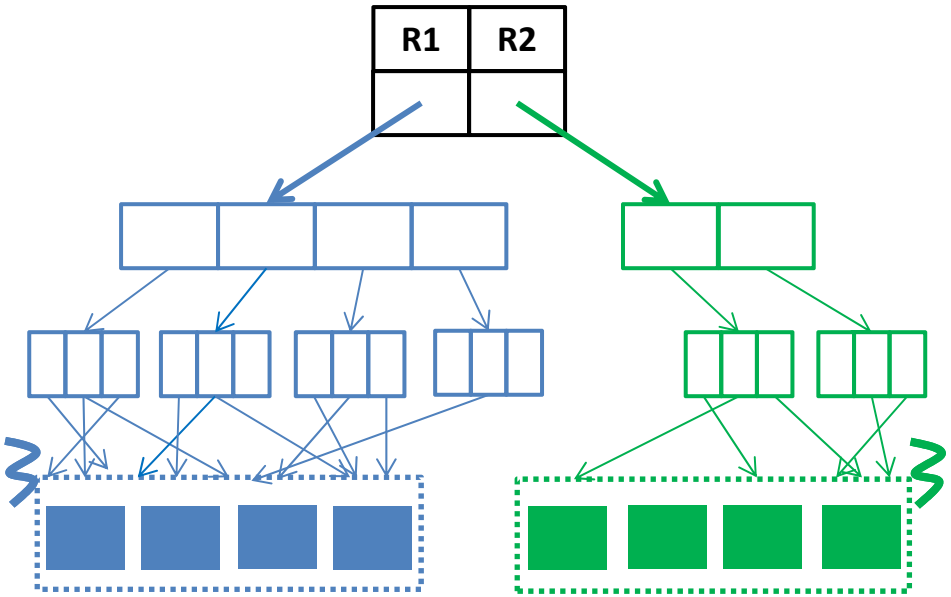
PLP-Leaf



- ✘ Two-step record inserts
- ✘ Fragmentation
- ✓ Repartitioning worst-case: Scan few pages

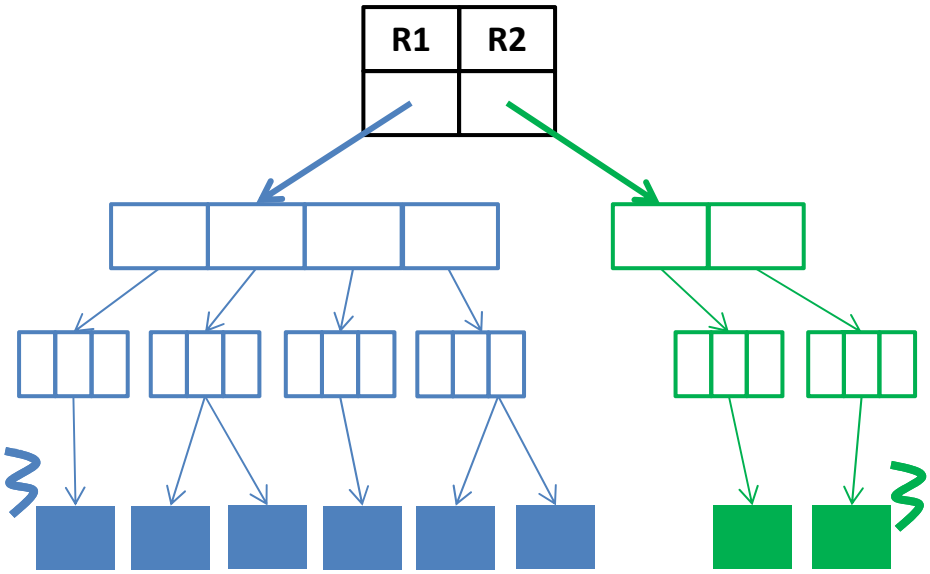
Heap Pages : Alternatives

PLP-Partition



- ✘ Two-step record inserts
- ✘ Repartitioning worst-case: Scan entire partition

PLP-Leaf



- ✘ Two-step record inserts
- ✘ Fragmentation
- ✓ Repartitioning worst-case: Scan few pages

✓ Latch free OLTP

Outline

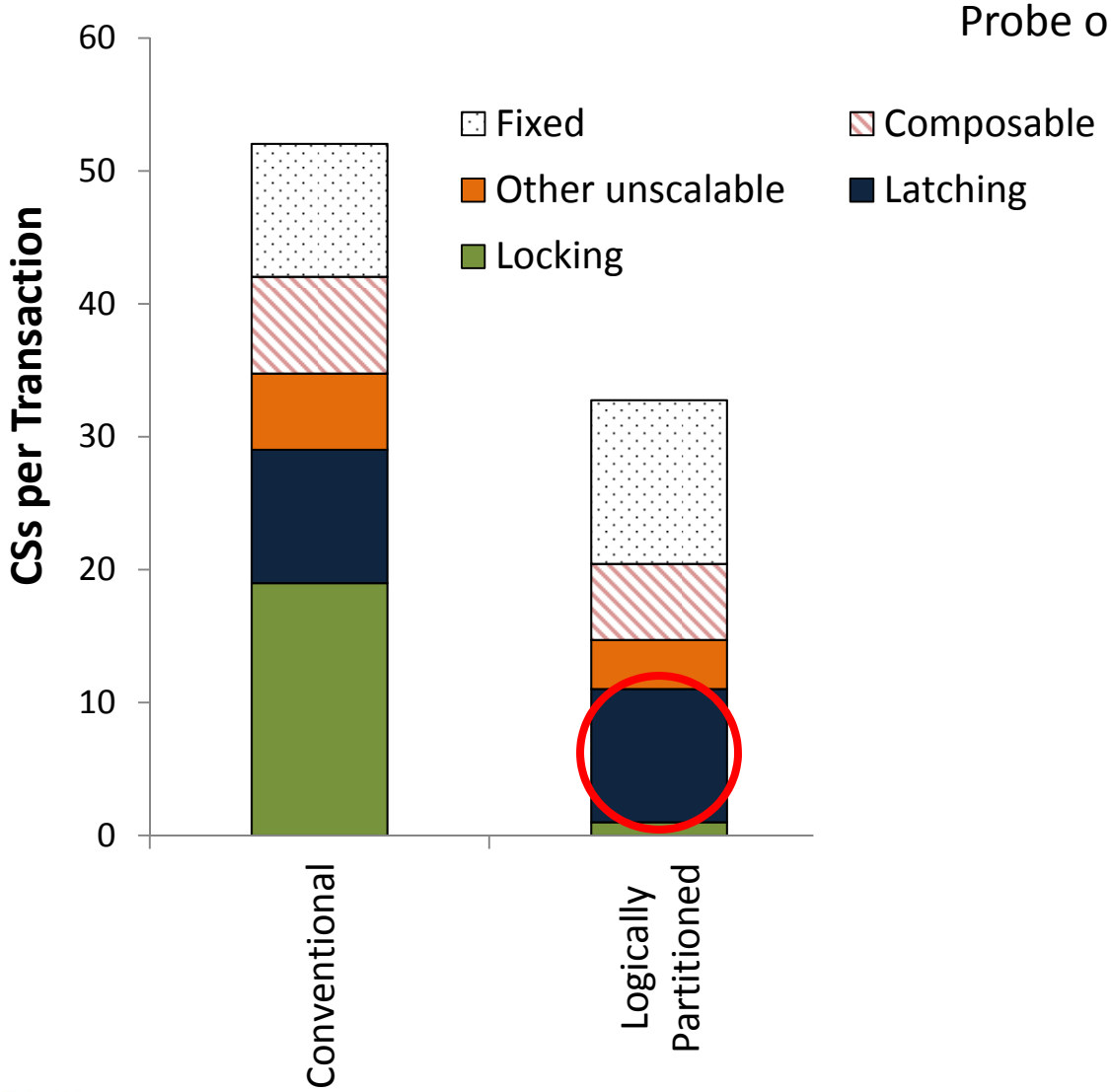
- Introduction
- Types of Critical Sections
- Physiological Partitioning (PLP)
- **Results**
- Conclusion

Setup

- All prototypes built on top of Shore-MT
  - State-of-the-art open-source DBMS
- Machines used
  - Sun Niagara T2, 64 HW ctxs
    - In order, 1.4GHz, 64GB RAM
  - 4 socket quad-core AMD Opteron, 16 HW ctxs
    - OoO, 2.4GHz, 64GB RAM
- #Partitions = #HW contexts available

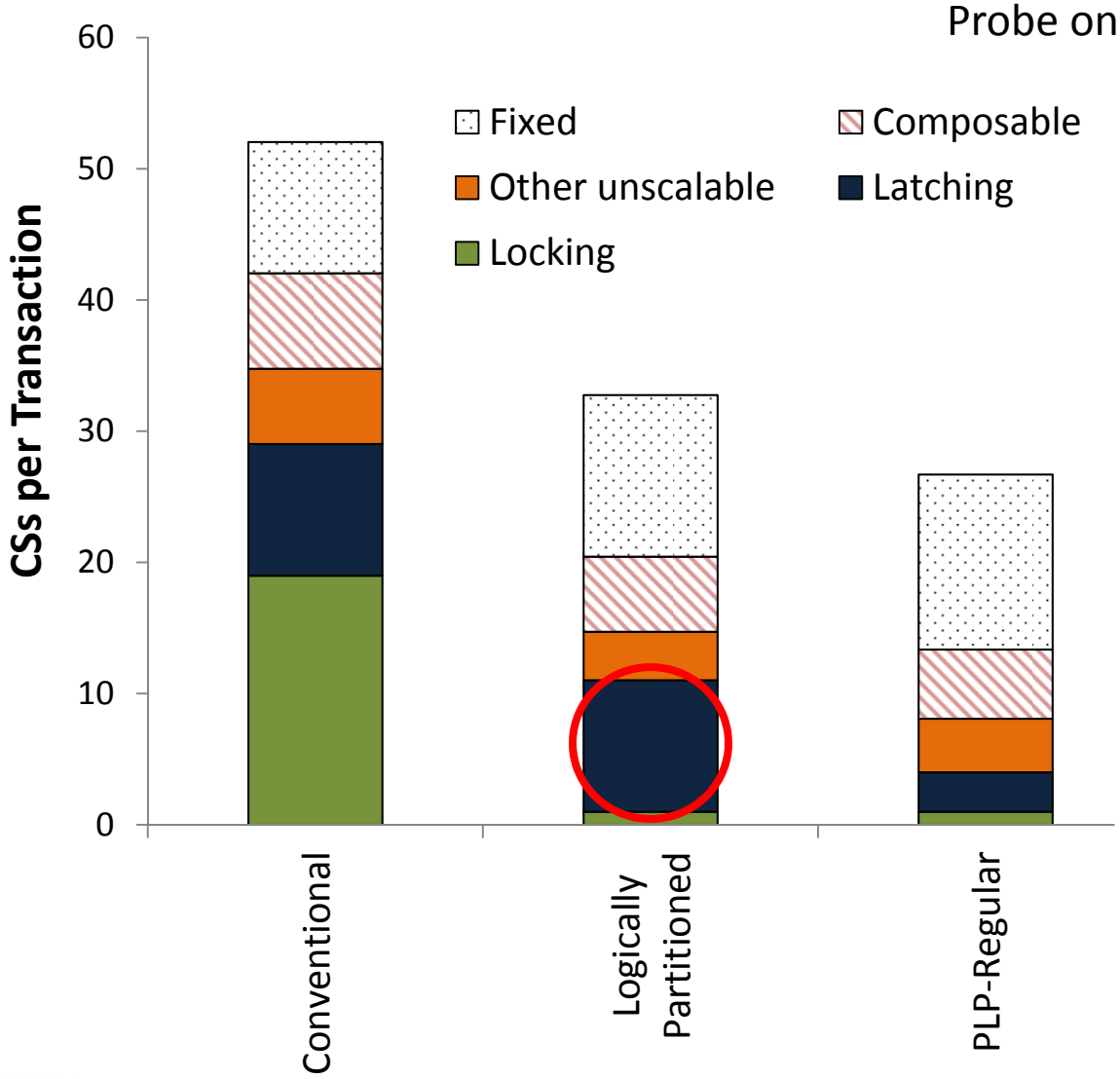
Breakdown of the Critical Sections

Probe one customer, update balance

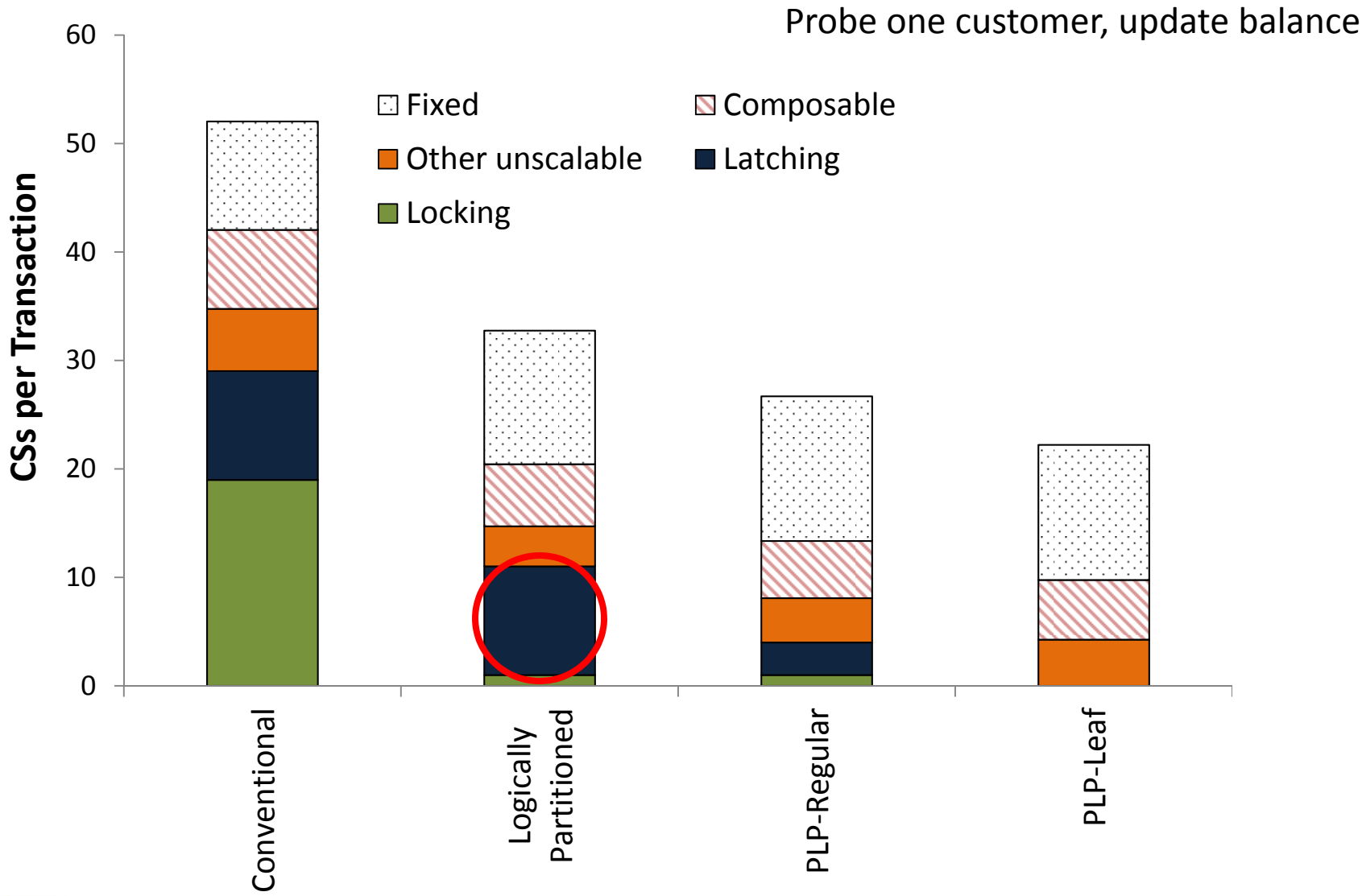


Breakdown of the Critical Sections

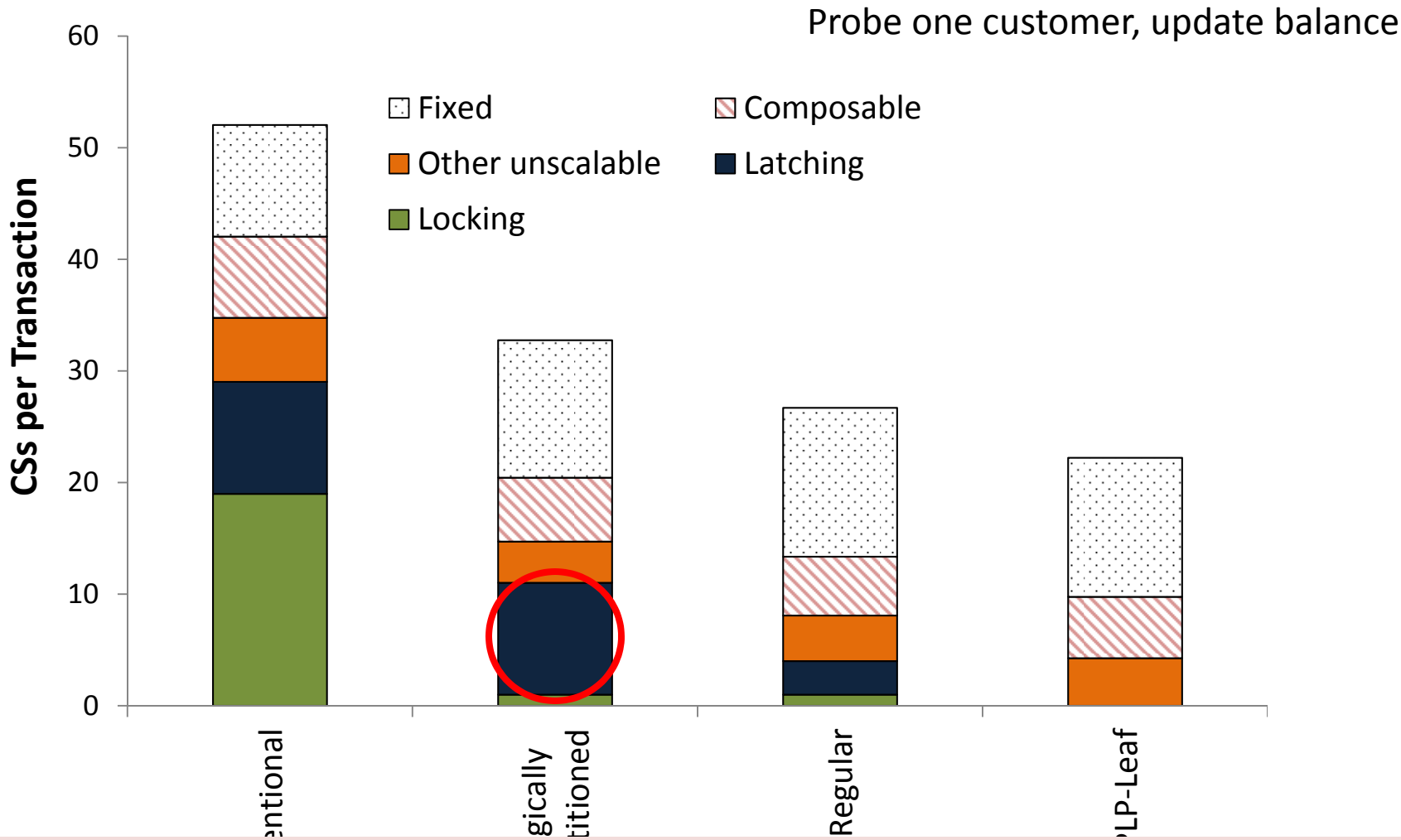
Probe one customer, update balance



Breakdown of the Critical Sections



Breakdown of the Critical Sections



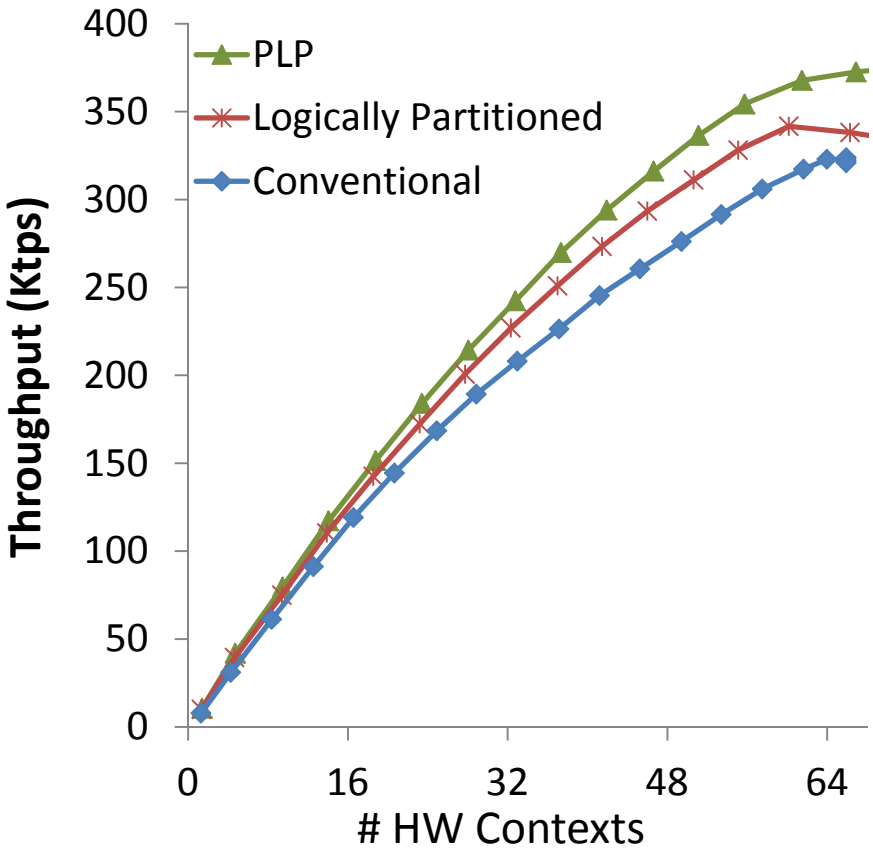
PLP eliminates majority of the unscalable CSs

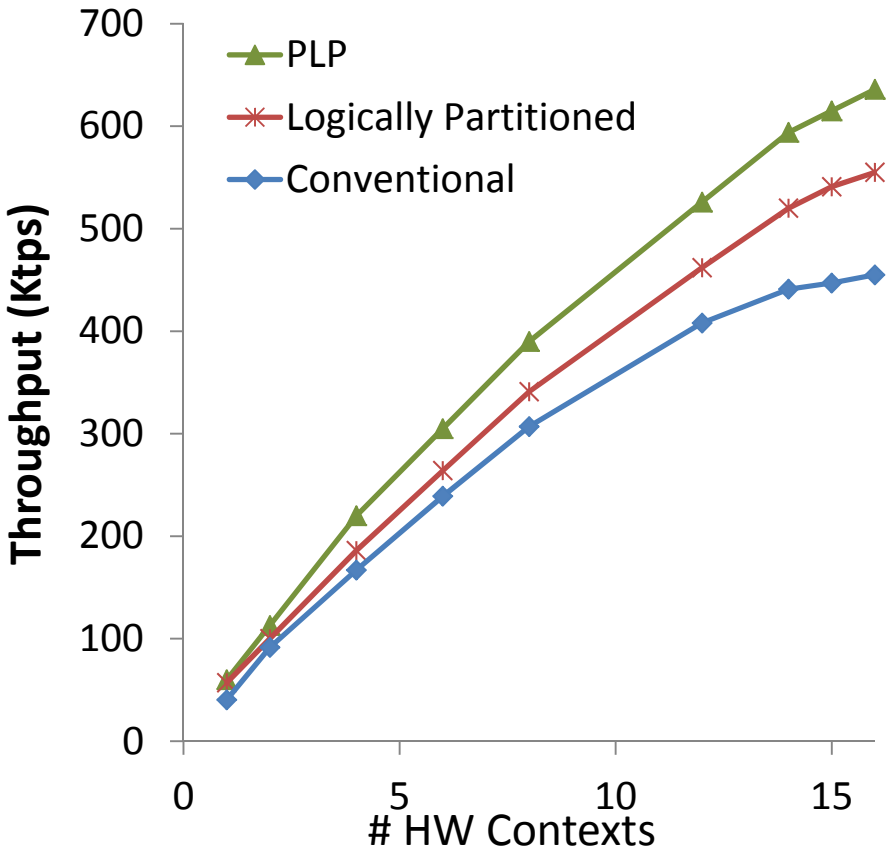
Performance on Multicores

Sun Niagara T2
64 HW ctxs, in order, 1.4GHz

TATP – GetSubData

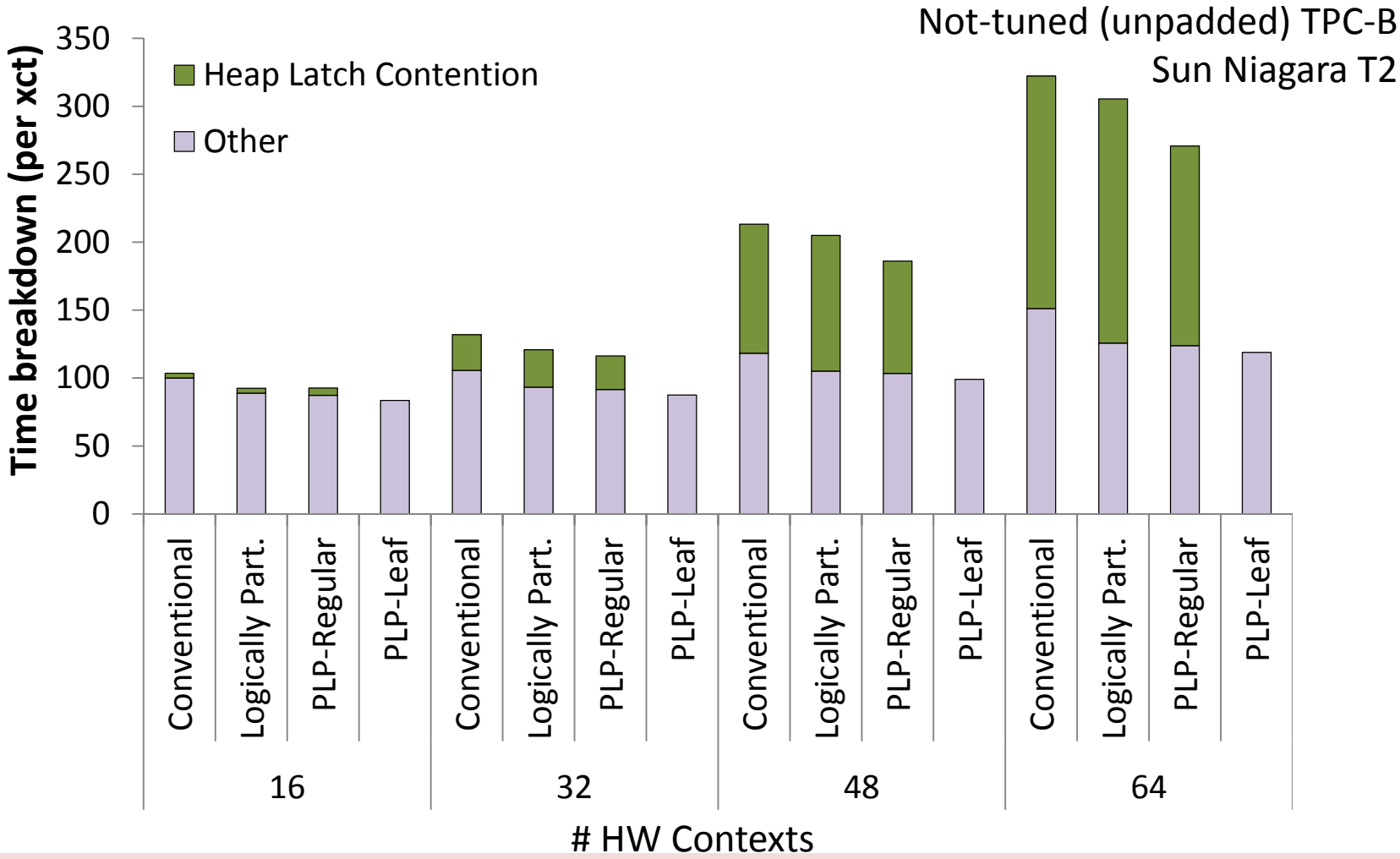
4 socket Quad AMD
16 HW ctxs, OoO, 2.8GHz





Benefits increase with faster hardware

Contention for heap pages



Avoids heap false sharing problems

Repartitioning Cost

Scenario: Splitting a partition to two (466MB)
1 primary and 1 secondary index
8KB pages, 100B records, 32B keys, 3-level B+tree

	Heap Records Moved	Primary & Secondary Index (Updates, Inserts, Deletes)
Shared-nothing		
PLP-Partition		
PLP-Leaf		

Repartitioning Cost

Scenario: Splitting a partition to two (466MB)
1 primary and 1 secondary index
8KB pages, 100B records, 32B keys, 3-level B+tree

	Heap Records Moved	Primary & Secondary Index (Updates, Inserts, Deletes)
Shared-nothing	233MB	2.4 million inserts 2.4 million deletes
PLP-Partition		
PLP-Leaf		

Repartitioning Cost

Scenario: Splitting a partition to two (466MB)
1 primary and 1 secondary index
8KB pages, 100B records, 32B keys, 3-level B+tree

	Heap Records Moved	Primary & Secondary Index (Updates, Inserts, Deletes)
Shared-nothing	233MB	2.4 million inserts 2.4 million deletes
PLP-Partition	233MB	2.4 million updates
PLP-Leaf		

Repartitioning Cost

Scenario: Splitting a partition to two (466MB)
1 primary and 1 secondary index
8KB pages, 100B records, 32B keys, 3-level B+tree

	Heap Records Moved	Primary & Secondary Index (Updates, Inserts, Deletes)
Shared-nothing	233MB	2.4 million inserts 2.4 million deletes
PLP-Partition	233MB	2.4 million updates
PLP-Leaf		

DATA-INTENSIVE APPLICATIONS AND SYSTEMS

Repartitioning Cost

Scenario: Splitting a partition to two (466MB)
1 primary and 1 secondary index
8KB pages, 100B records, 32B keys, 3-level B+tree

	Heap Records Moved	Primary & Secondary Index (Updates, Inserts, Deletes)
Shared-nothing	233MB	2.4 million inserts 2.4 million deletes
PLP-Partition	233MB	2.4 million updates
PLP-Leaf	8.3KB	85 updates

Repartitioning Cost

Scenario: Splitting a partition to two (466MB)
1 primary and 1 secondary index
8KB pages, 100B records, 32B keys, 3-level B+tree

	Heap Records Moved	Primary & Secondary Index (Updates, Inserts, Deletes)
Shared-nothing	233MB	2.4 million inserts 2.4 million deletes
PLP-Partition	233MB	2.4 million updates
PLP-Leaf	8.3KB	85 updates

PLP-Leaf: low repartitioning cost + latch-free

Conclusion

- Multicores expose the bottlenecks of DBMSs
- Understanding the critical sections is crucial
  - Identify the harmful ones and eliminate them
- Physiological partitioning
  - Apply the right partitioning at both logical & physical layers
  - Thread local locks & latch free data accesses
  - Eliminate majority of unscalable critical sections
  - Benefits of shared-nothing with easy repartitioning

Thank you!