

Energy-aware Memory Management through Database Buffer Control

Chang S. Bae, Tayeb Jamel

Northwestern Univ.

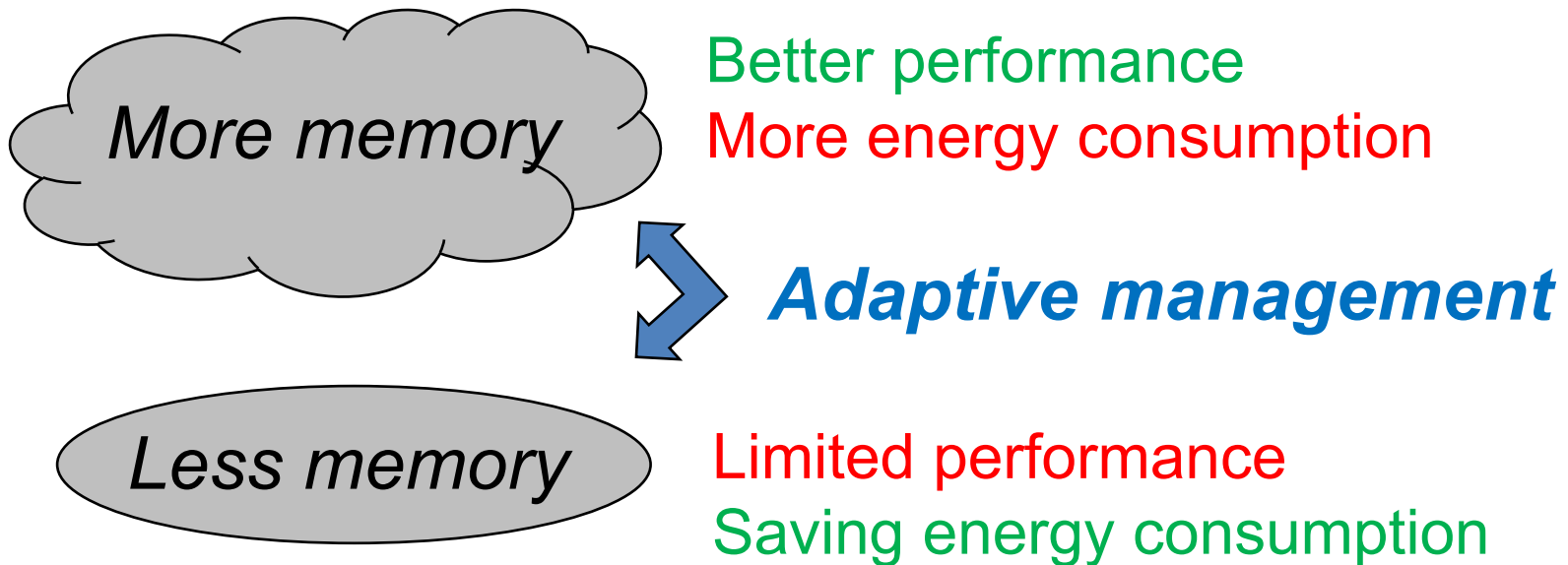
Intel Corporation

Presented by Chang S. Bae



Goal and motivation

- Energy-aware memory management in software
 - In idle time, unload memory to save energy
 - In peak time, load memory to get performance



Contributions of this work

- Energy-aware memory management in software
 - In idle time, unload memory to save energy
 - In peak time, load memory to get performance
- Application-level approach
 - Dynamically adaptive buffer management in data base engine
- Real measurements
- Design and implement in a system

Outline

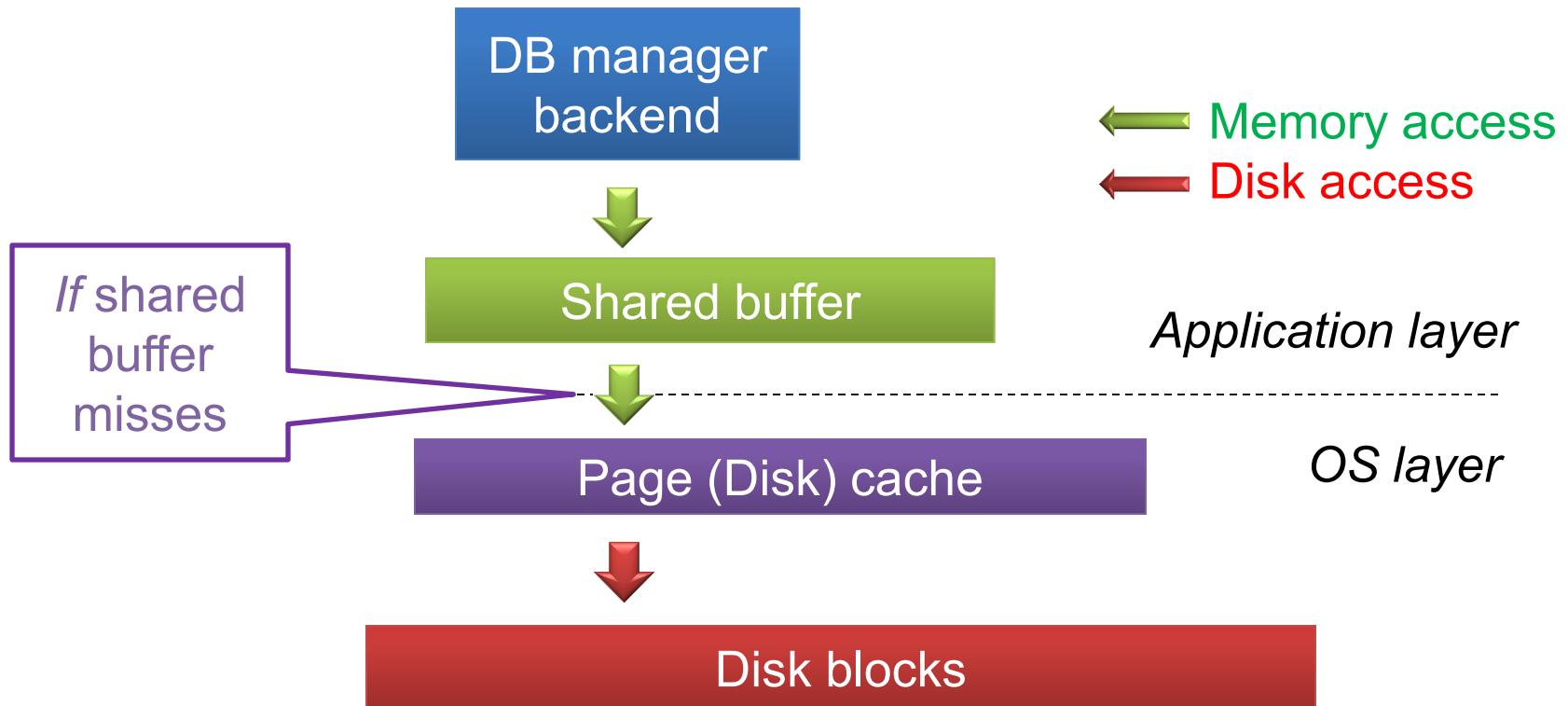
- Introduction
- Background and metrics
 - Application driven approach
 - Memory usage in database engine
 - Metrics
- Database buffer pool control
- Evaluation
- Conclusion

Application driven energy savings

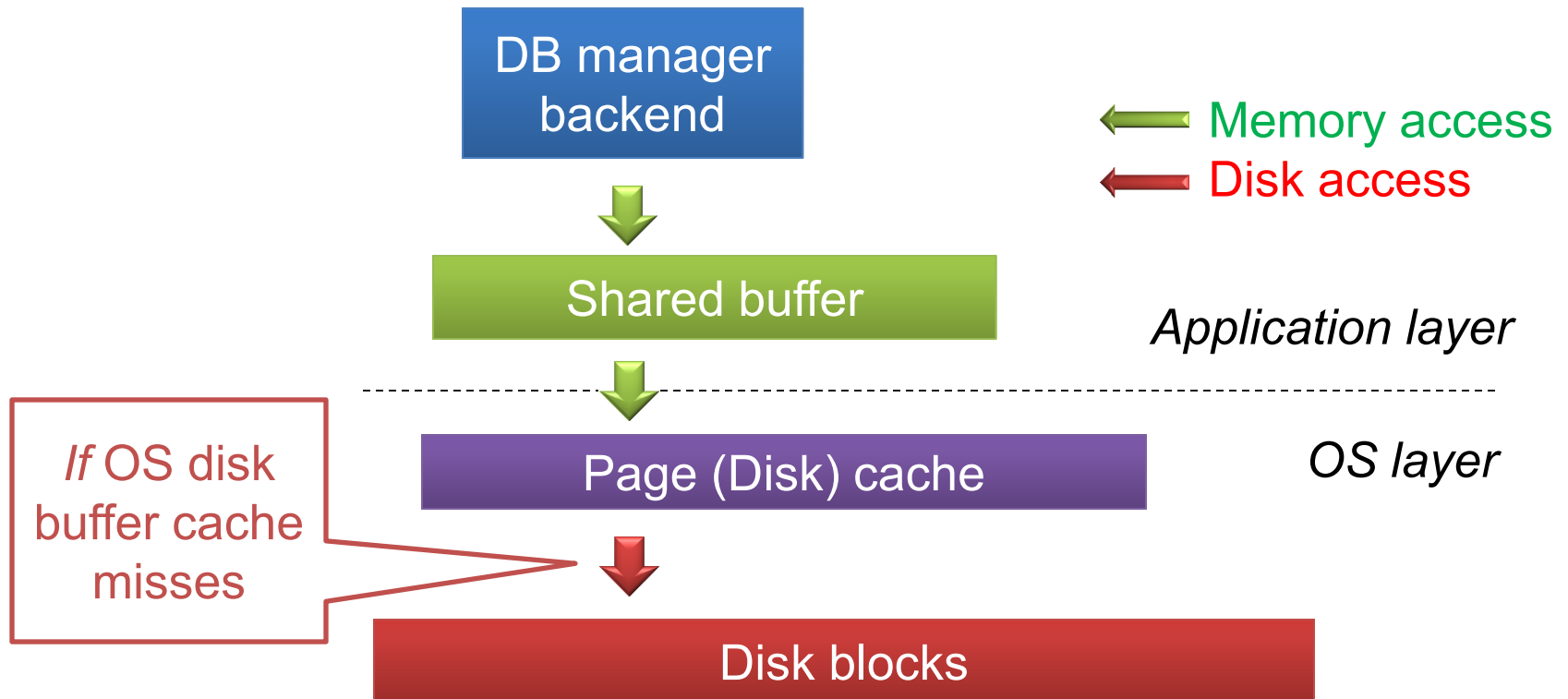
- Software over hardware
 - Hardware approach takes more delay to do measurement and inference
- Application-level over OS-level
 - Application itself directly and best knows its behavior

Applied to database buffer pool management

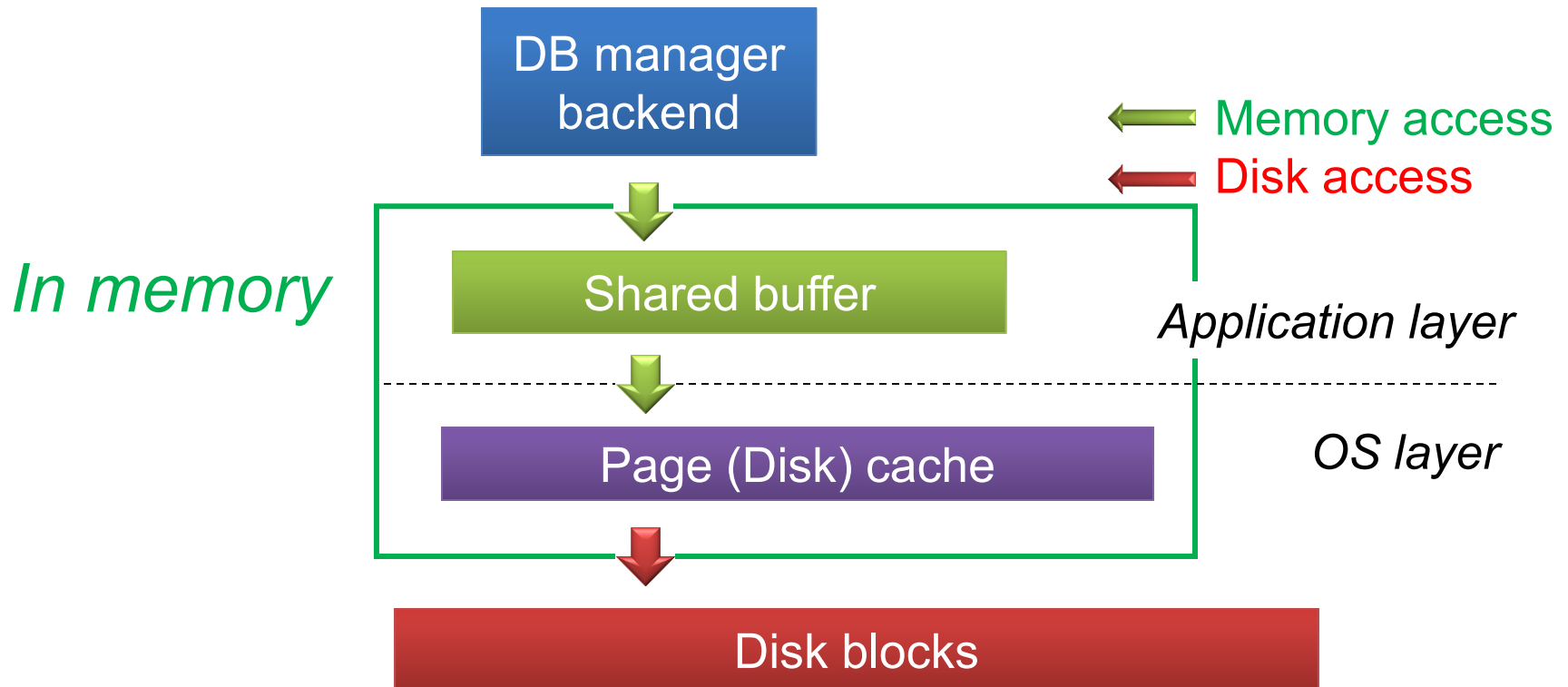
Database manager uses memory as buffer pool



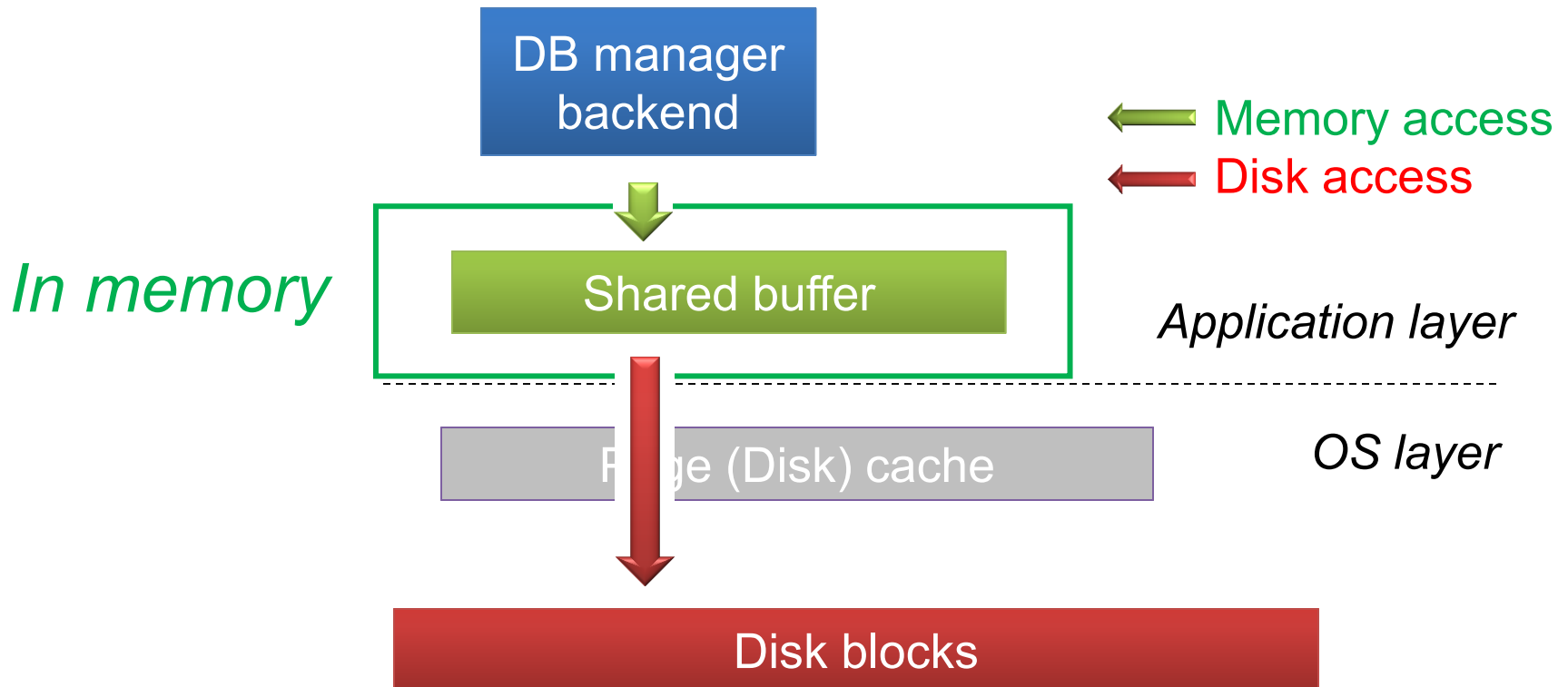
Database manager uses memory as buffer pool



Database manager uses memory as buffer pool



Managing Buffer pool with page caching disabled

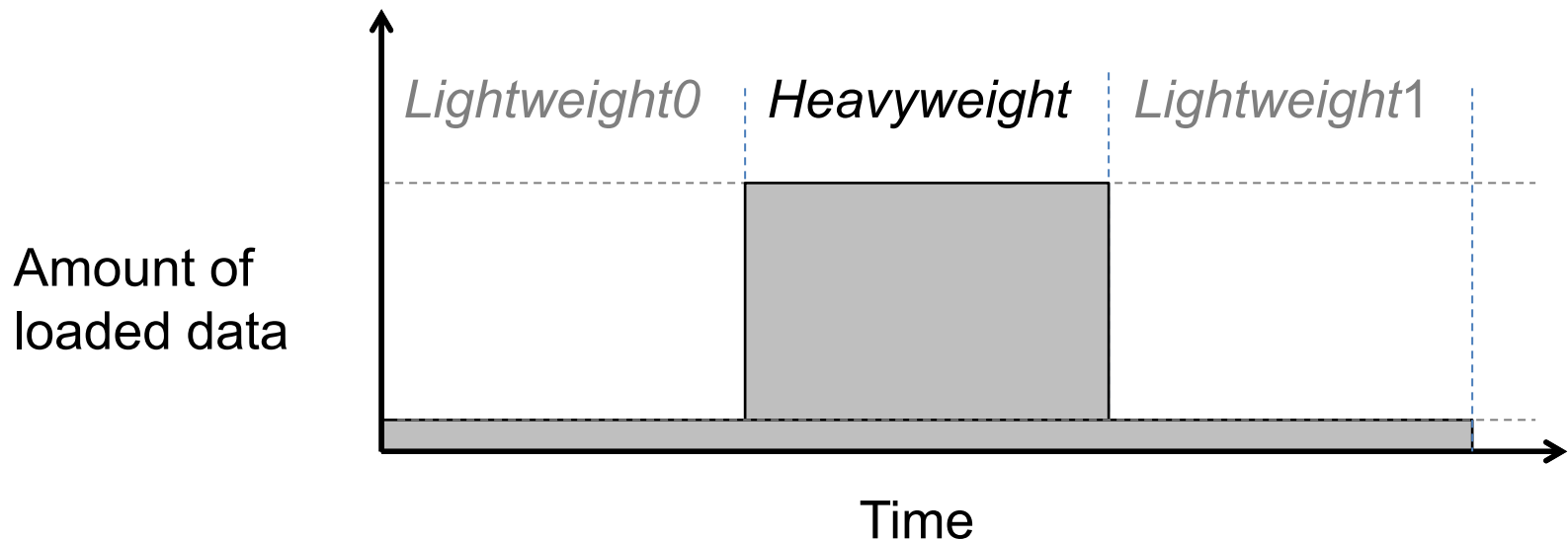


Monitoring buffer pool stats

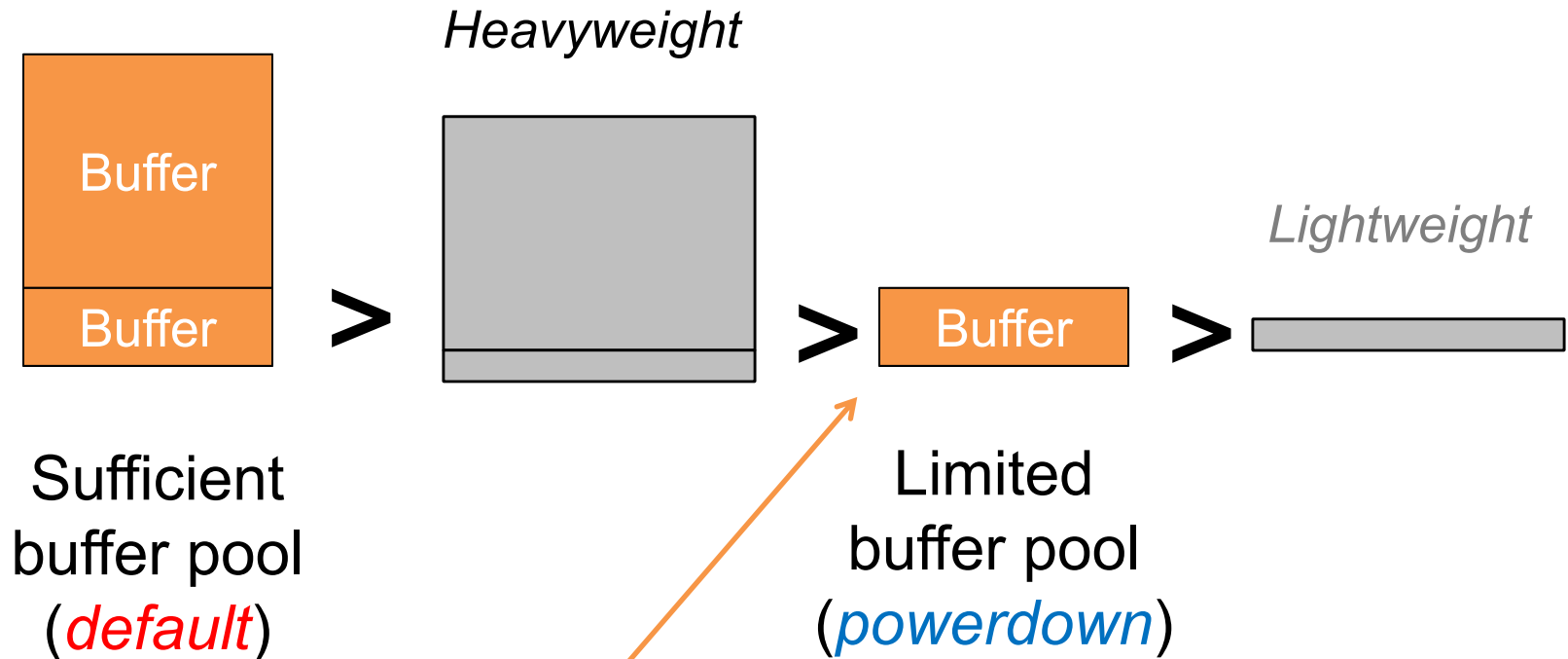
- **Buffer hit rate (*Bhit*)**
 - When to *expand* buffer pool
- **Buffer utilization (*Butil*)**
 - Recently used buffer size
 - When to *shrink* buffer pool

Scenario for workloads

- Online transactional processing (OLTP)
- Amount of loaded database varies
 - Different number of warehouse in 3-phases
 - Preserve TPC-C compliancy

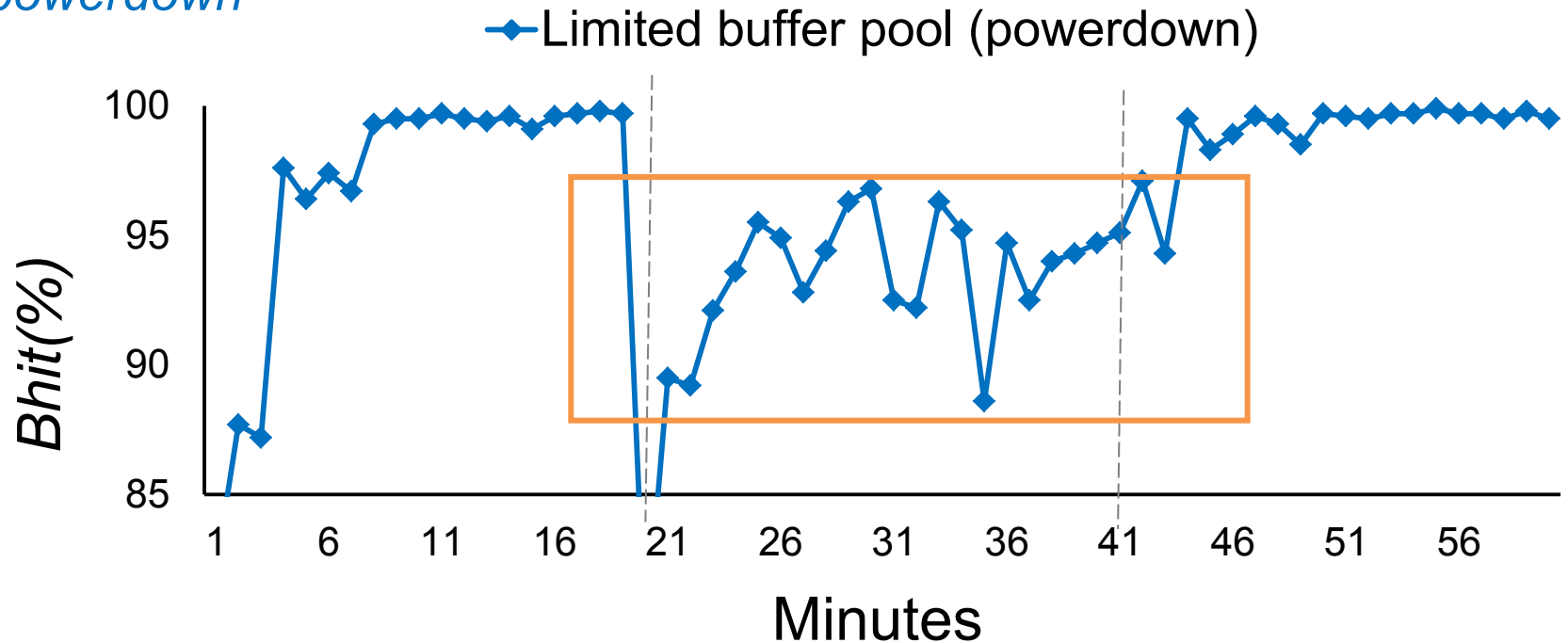
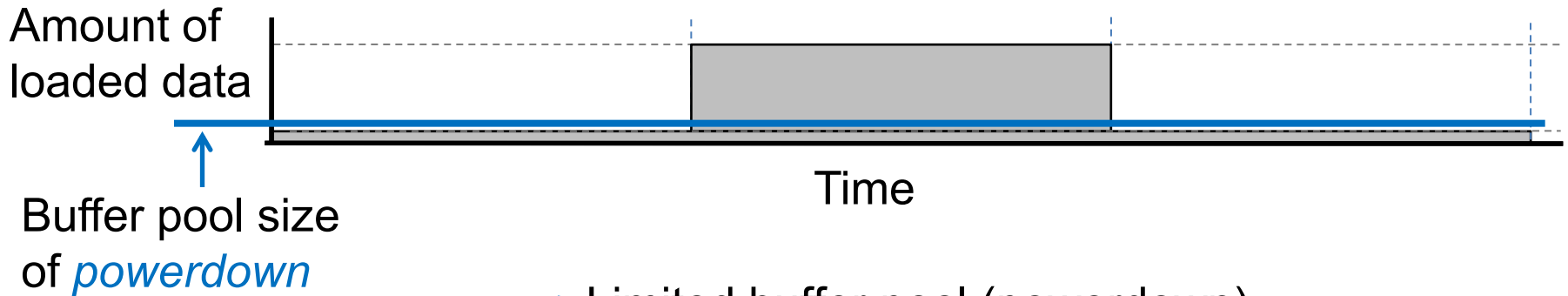


Buffer pool setup for database load

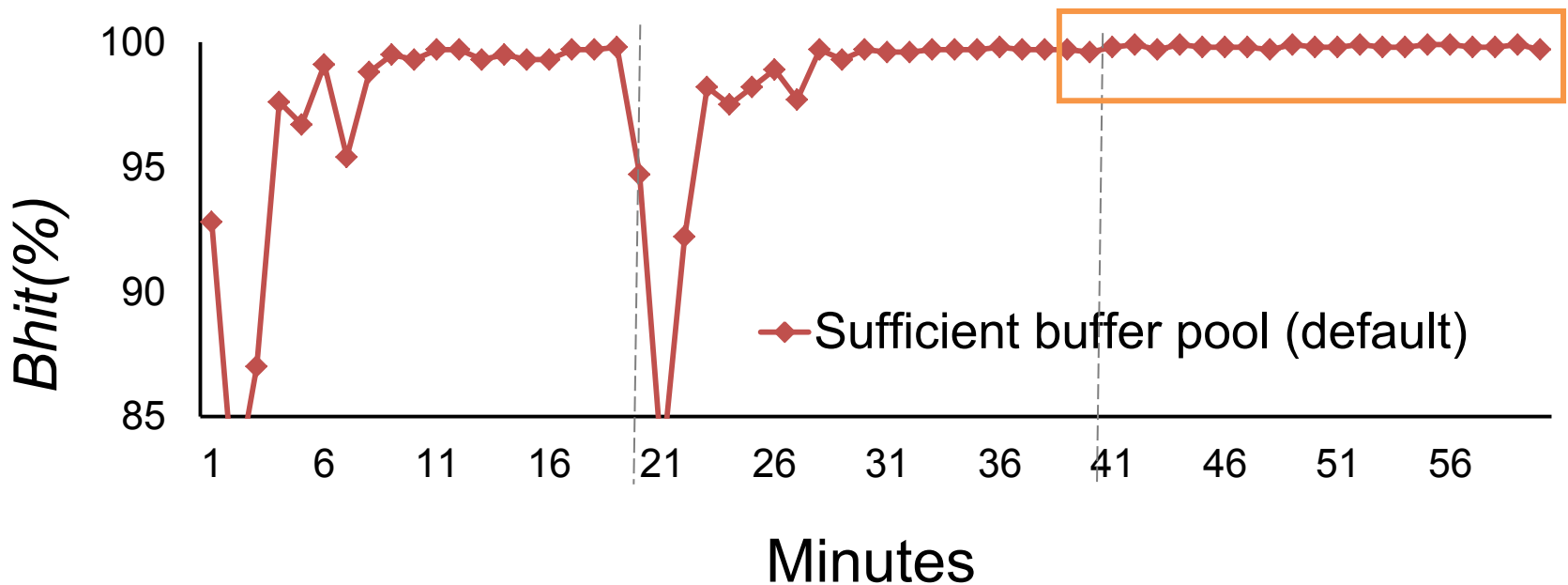
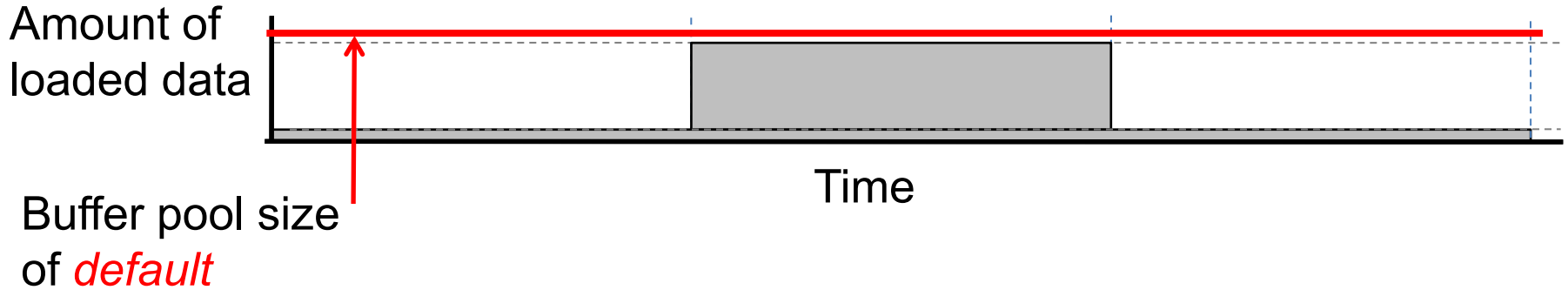


Opportunity to save energy
Possible to degrade performance

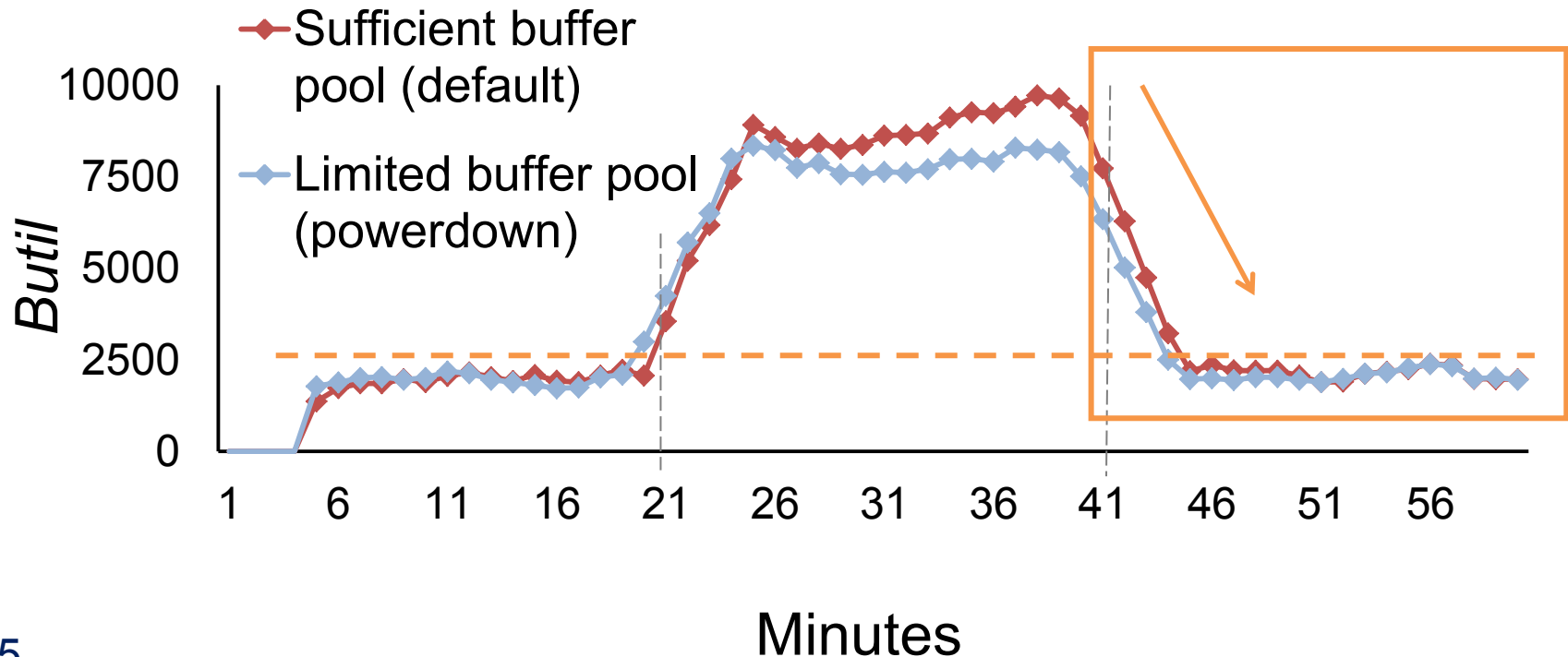
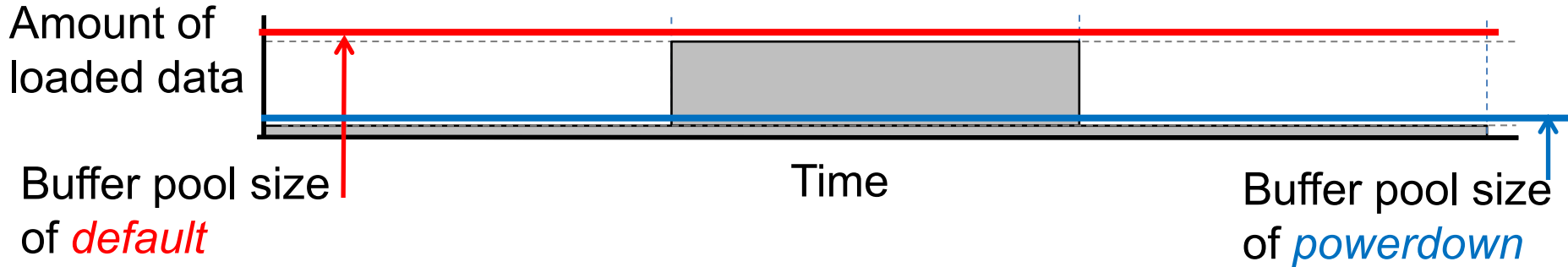
Bhit: possible to infer *when to expand* buffer pool



Bhit: hard to infer *when to shrink* buffer pool



Butil: possible to infer when to shrink buffer pool



Outline

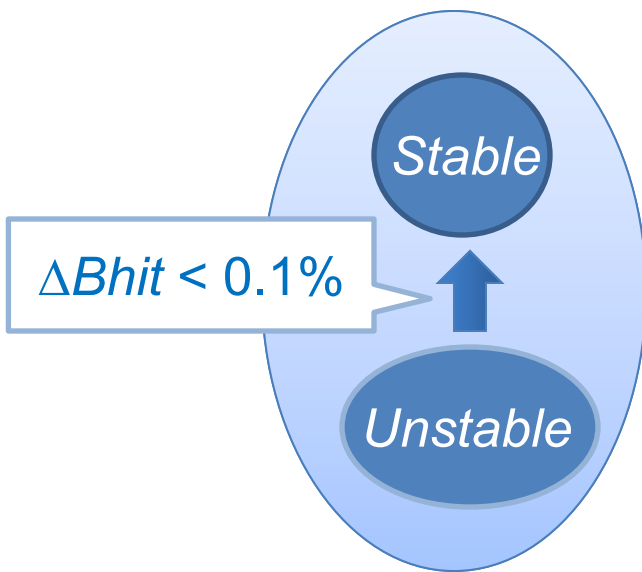
- Introduction
- Background and metrics
- Database buffer pool control
 - Threshold-based heuristics
 - State transitions
- Evaluation
- Conclusion

Threshold-based heuristics

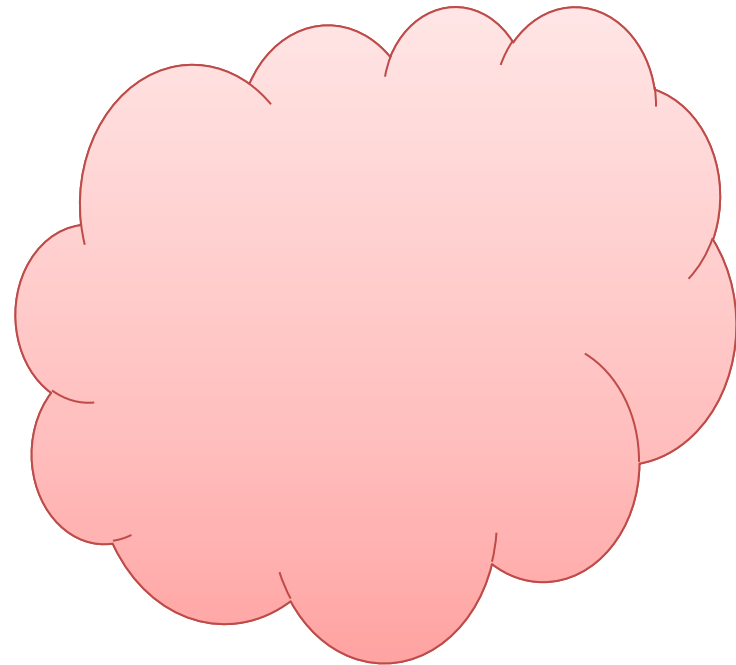
- High-level states
 - *Expansion*: increased buffer pool size
 - *Shrink*: reduced buffer pool size
 - Assuming two distinct domains
- Threshold value triggers state transition
 - *Threshold for Bhit* triggers expansion
 - *Threshold for Butil* triggers shrink

Shrink: waiting for stabilized

- Transit to *Stable*, if warmed up against cold misses



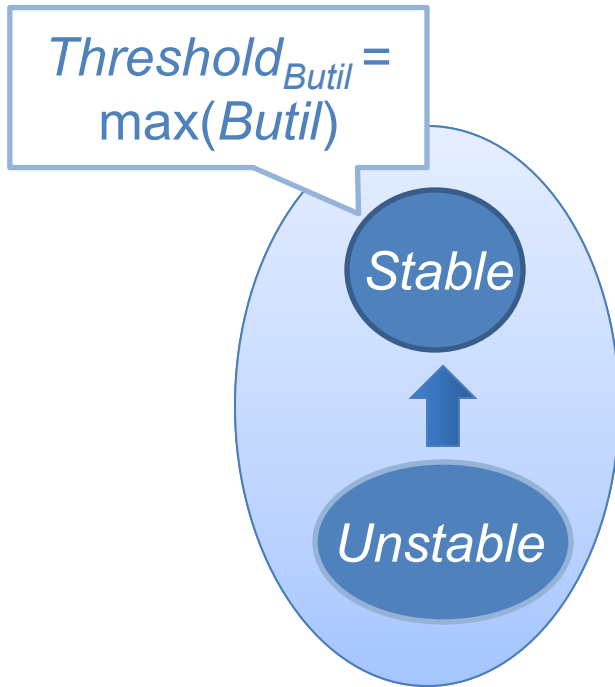
Shrink



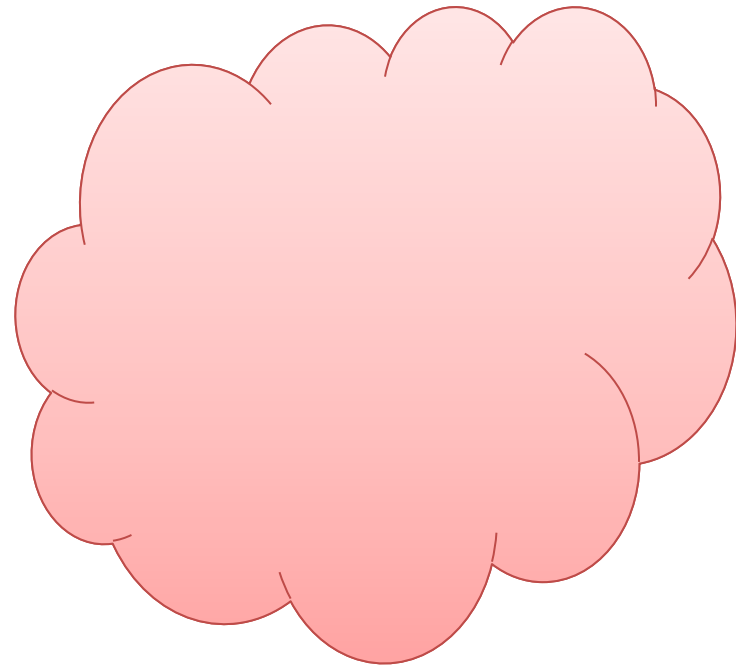
Expansion

Shrink: check if expansion needed

- Monitor B_{hit} and compare $Threshold_{B_{hit}}$
- Monitor B_{util} to keep maximum as $Threshold_{B_{util}}$



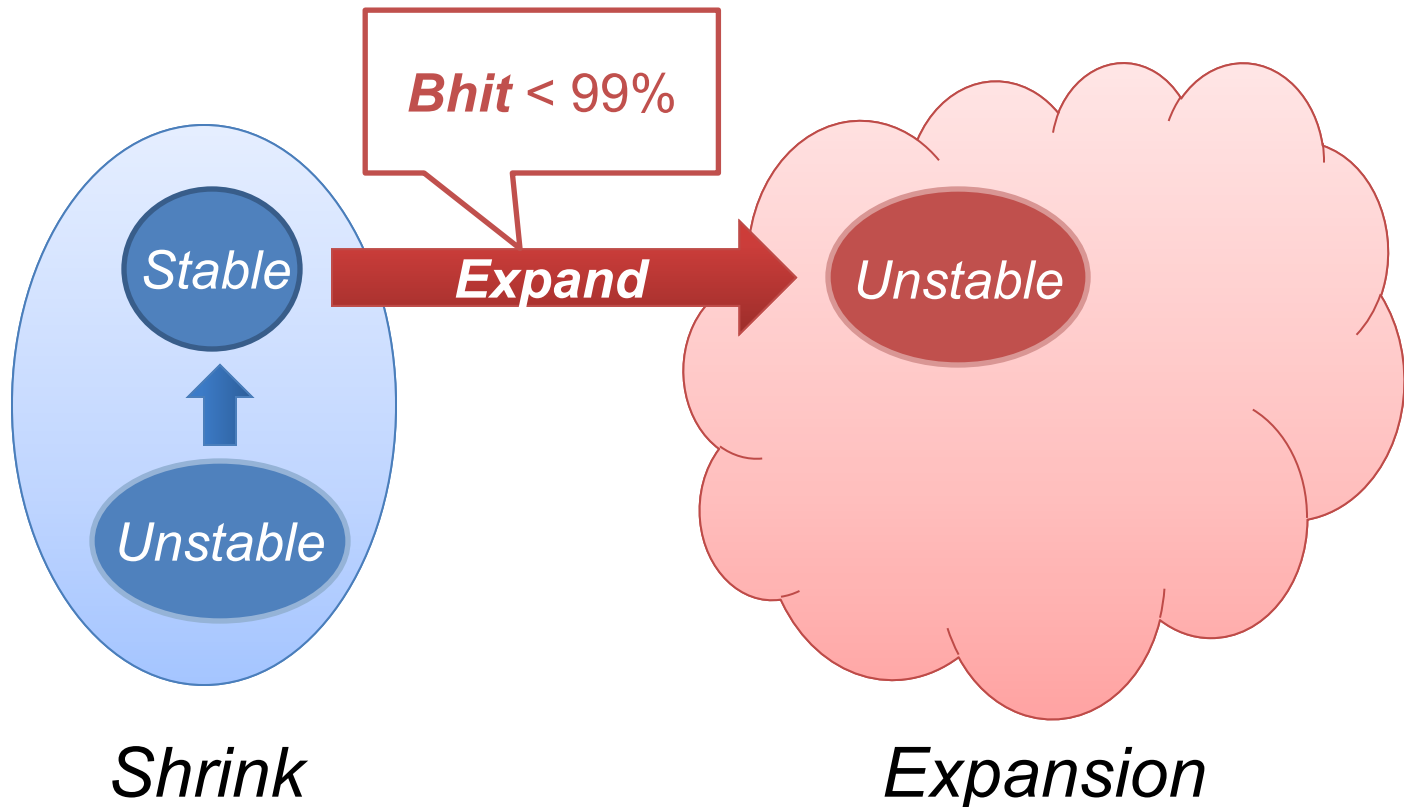
Shrink



Expansion

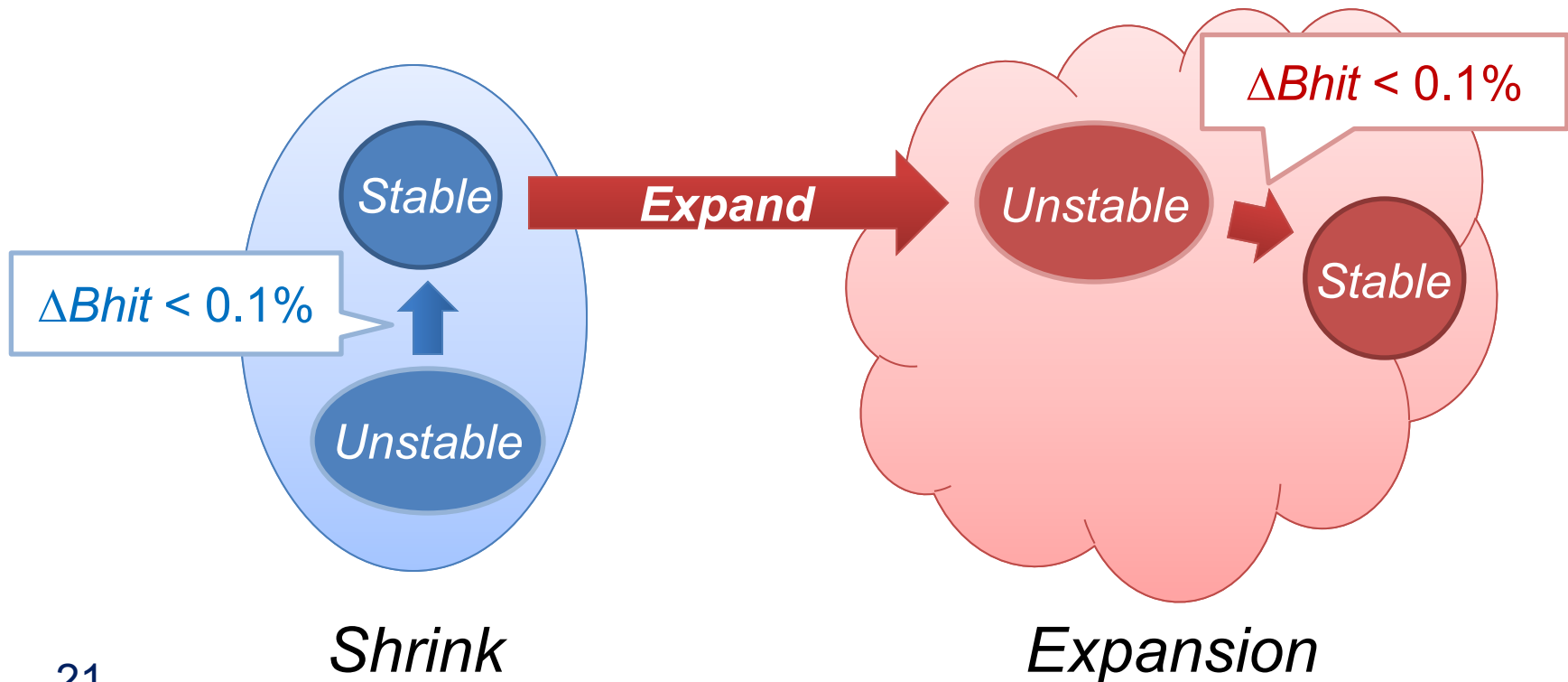
Buffer pool expansion

- Expanding buffer pool if compromised performance
 - Fixed $Threshold_{Bhit}$ to 99% for the best performance



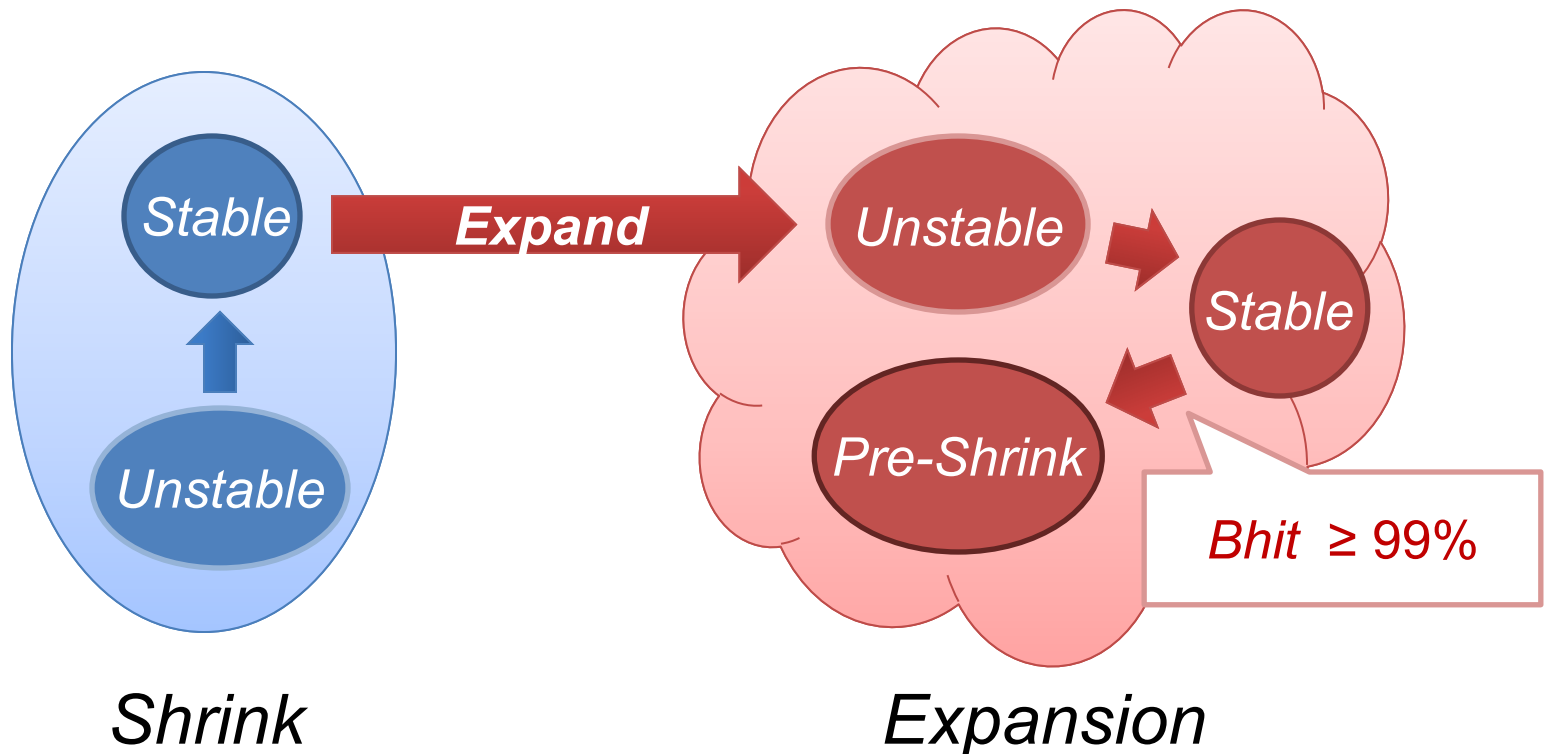
Expansion: waiting for being stabilized

- Transit to *Stable* if warmed up against cold misses



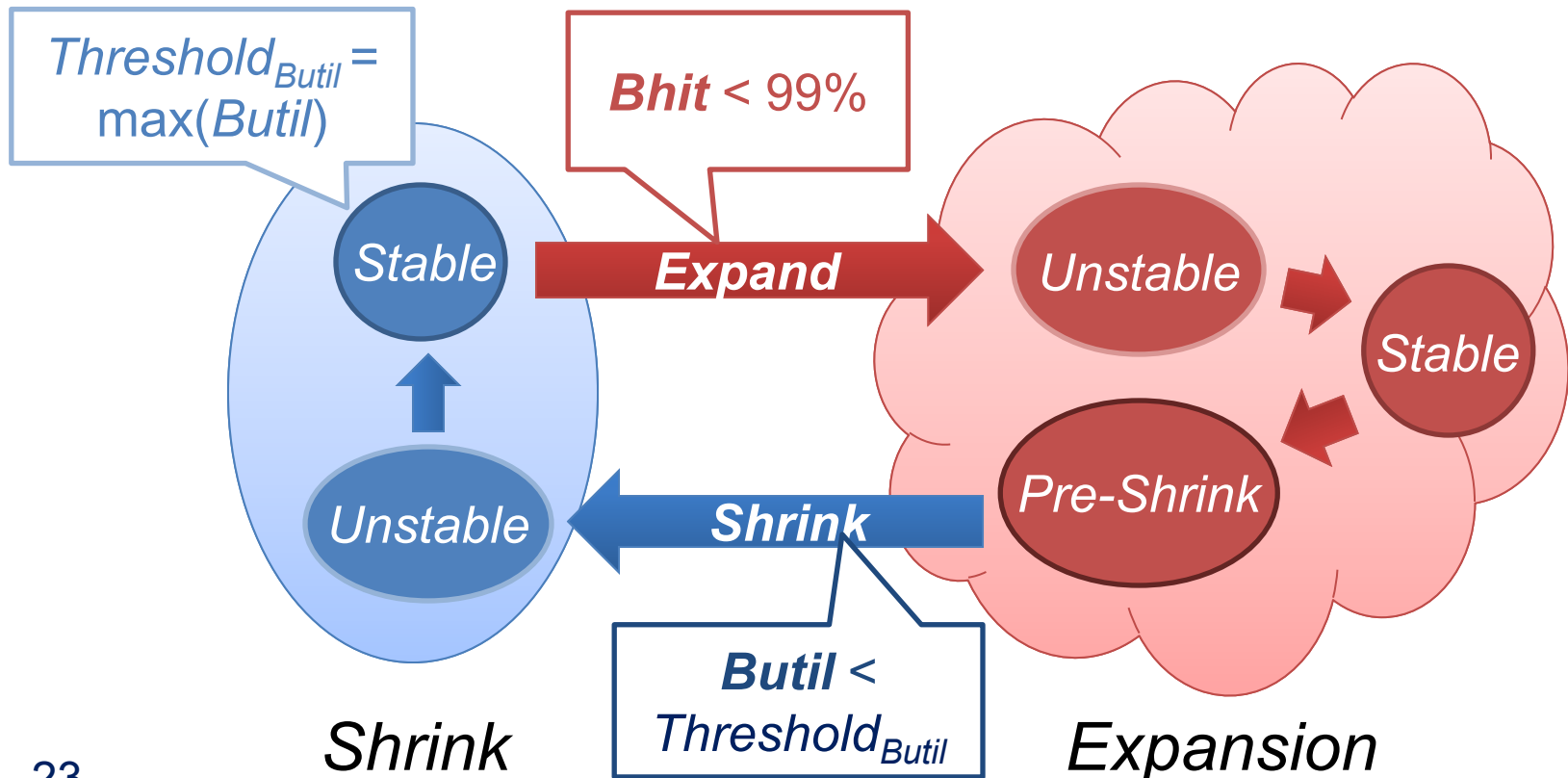
Expansion: check if shrink needed

- Move to *Pre-Shrink* when buffer pool looks sufficient
- *Pre-Shrink*: monitor B_{util} and compare $Threshold_{B_{util}}$



Buffer pool shrink

- Shrink buffer pool if B_{util} is below $Threshold_{B_{util}}$



Outline

- Introduction
- Background and metrics
- Database buffer pool control
- **Evaluation**
 - Software and hardware setup
 - Experimental results
- **Conclusion**

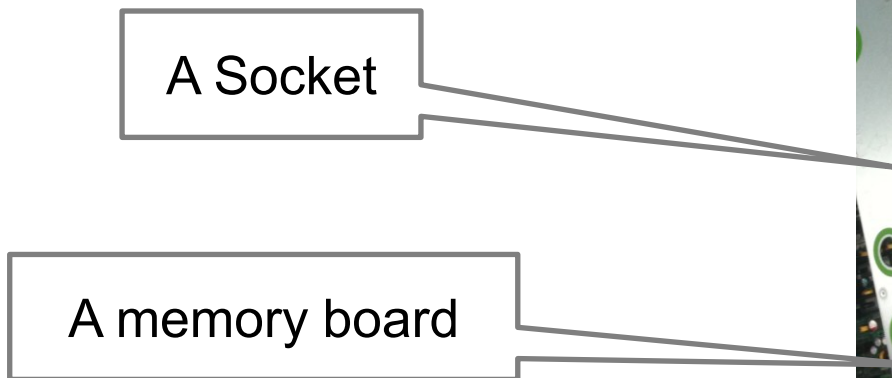
Software setup

- OLTP – TPCC UVa v.1.2.3
 - Modified to load different size of data in 3-phases
- DB manager – PostgreSQL v.8.4.6
 - Resize buffer pool during a runtime
- OS – RedHat Enterprise Linux v.5.5
 - Modified Kernel v.2.6.28.8
 - Consecutive virtual address linear to physical address in each socket

Hardware setup

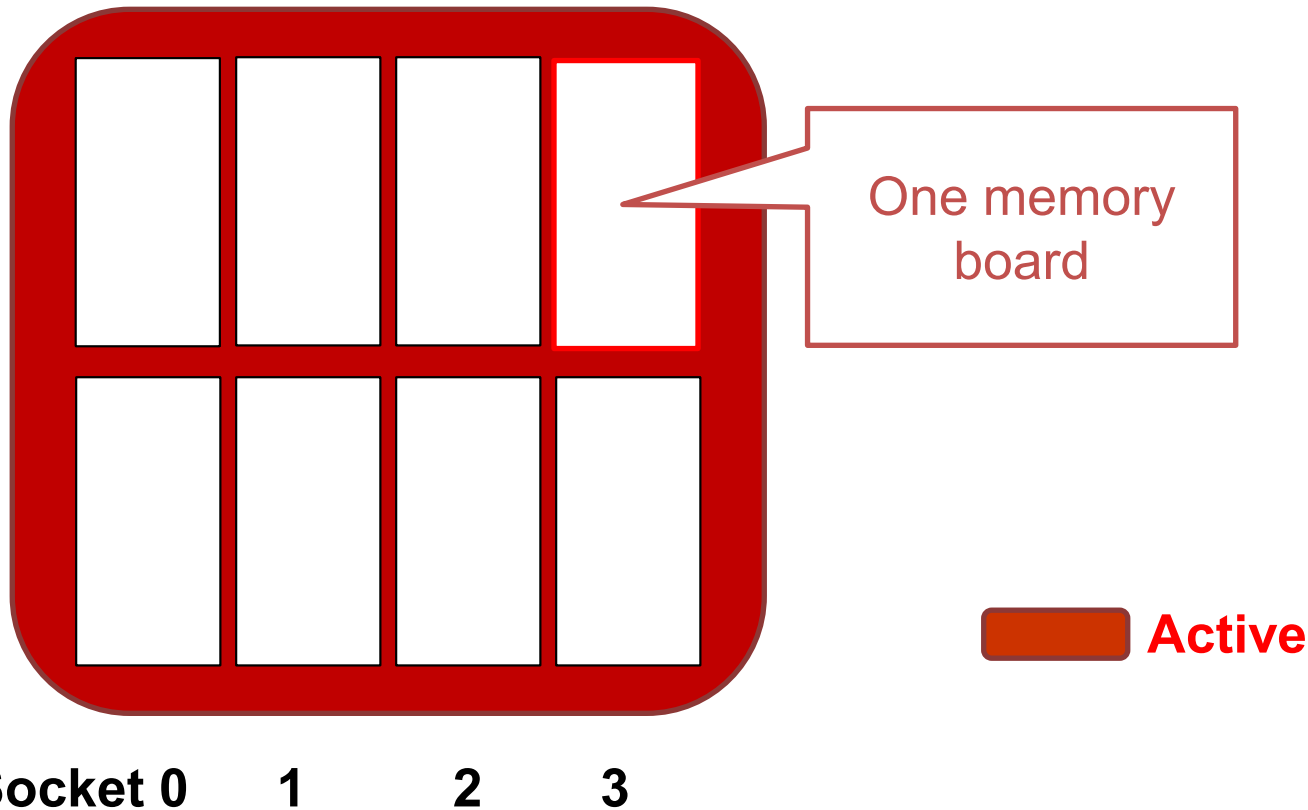
- CPU – Intel Xeon x7560 2.26GHz
 - 64 cores (*16 cores x 4 sockets*)
- DRAM –DDR3 1066MHz
 - 128 GB (*16GB x 2 memory-boards x 4 sockets*)

Nehalem EX
Platform



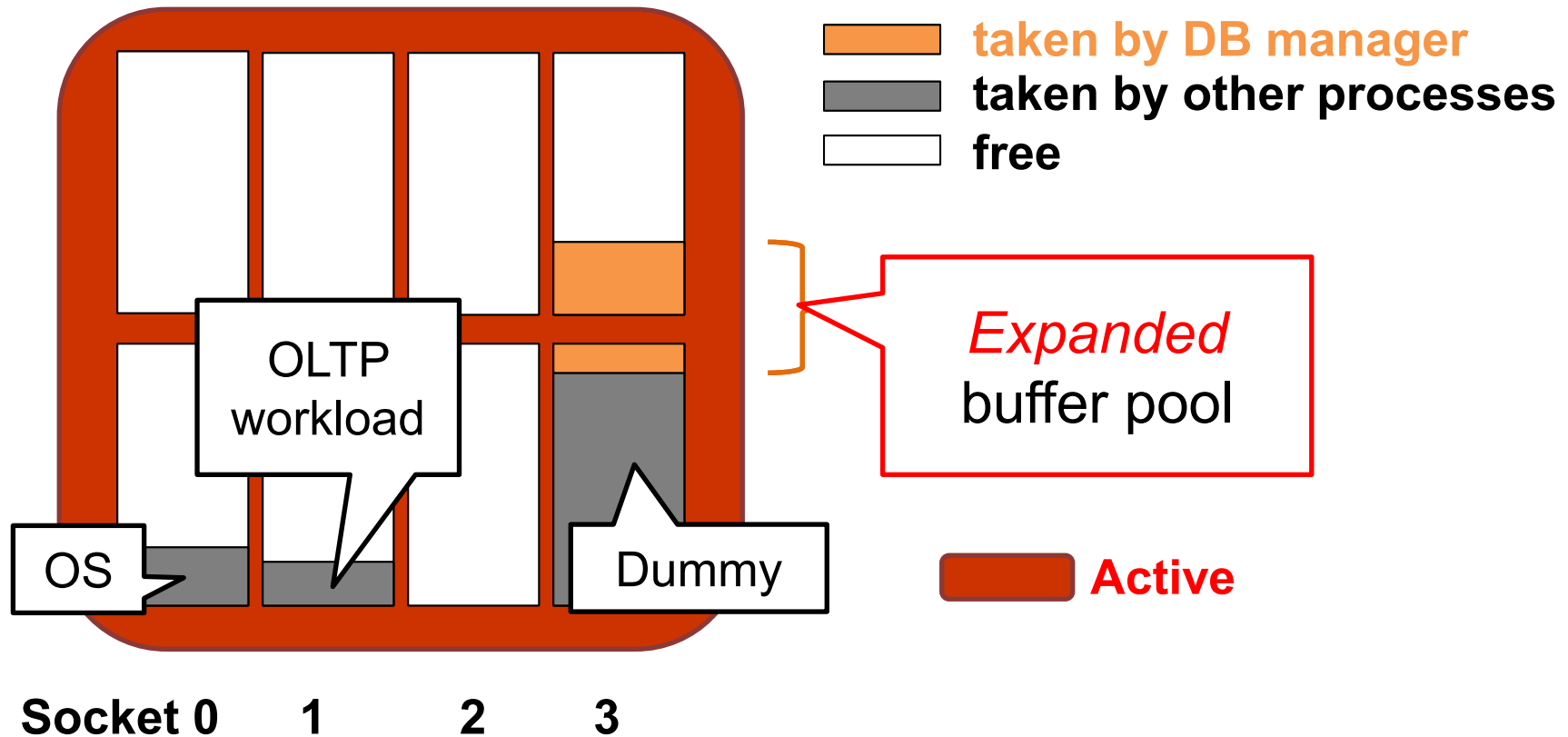
default configuration

- No memory power control



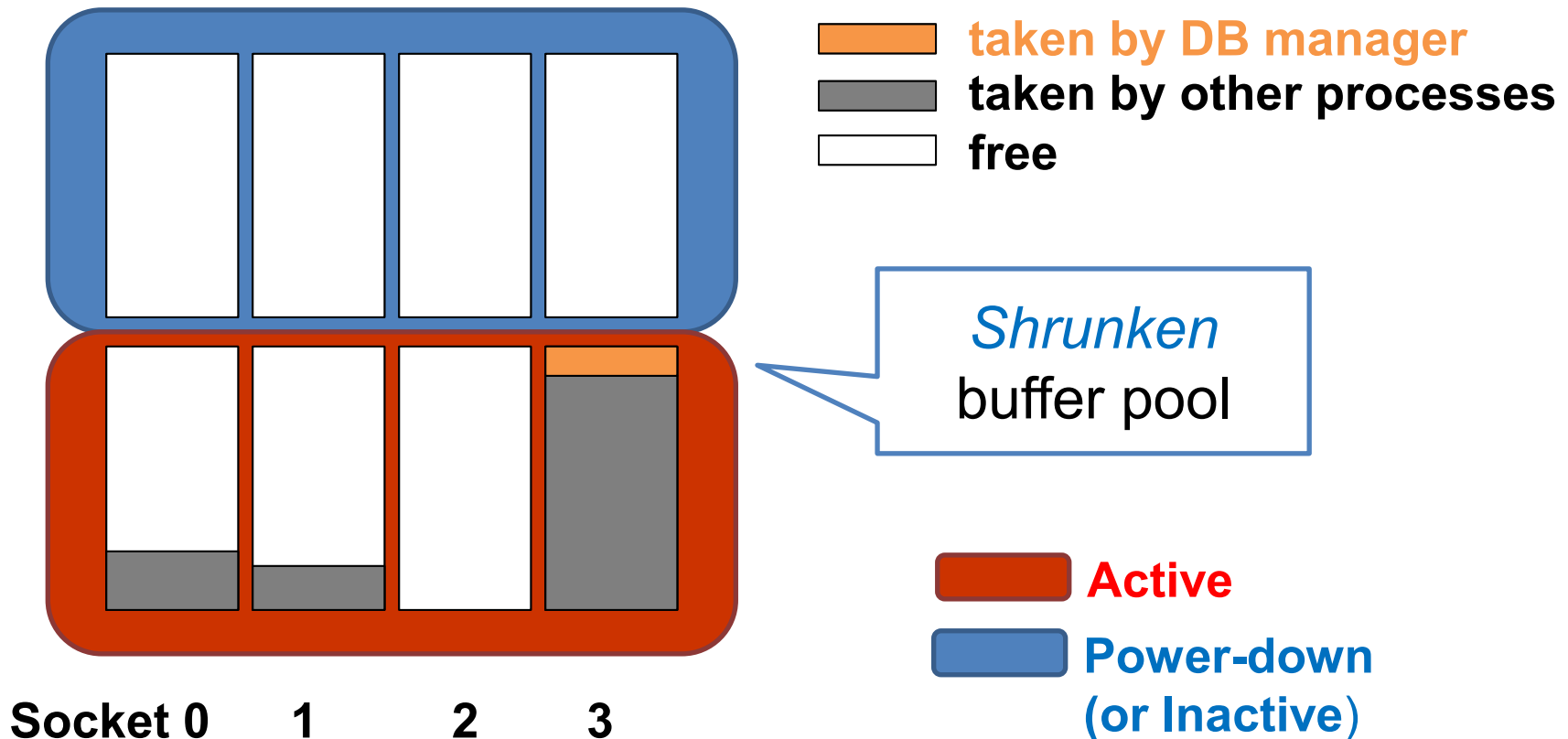
default configuration

- No memory power control

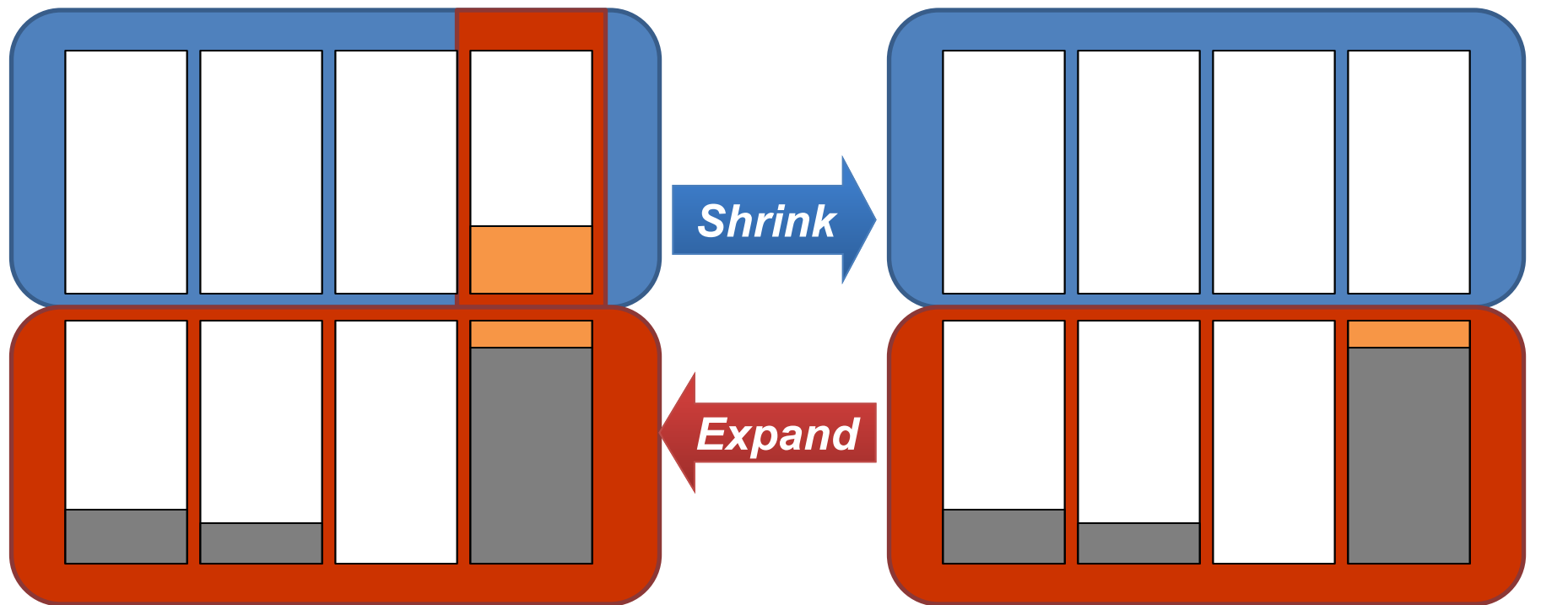


powerdown configuration

- Power-down secondary memory boards



adaptive configuration



Shrink

Expand

Socket 0 1 2 3

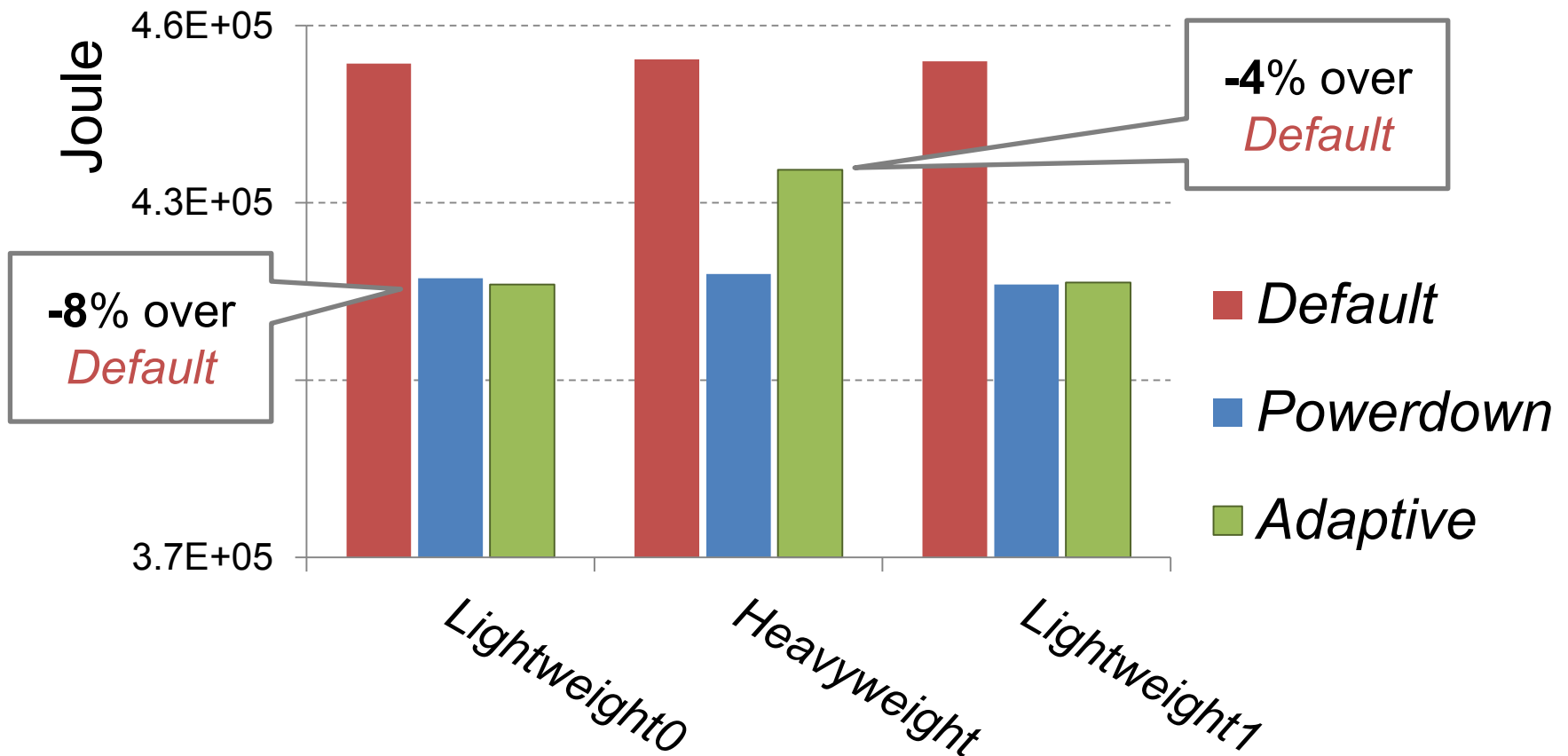
Socket 0 1 2 3

Active
**Power-down
(or Inactive)**

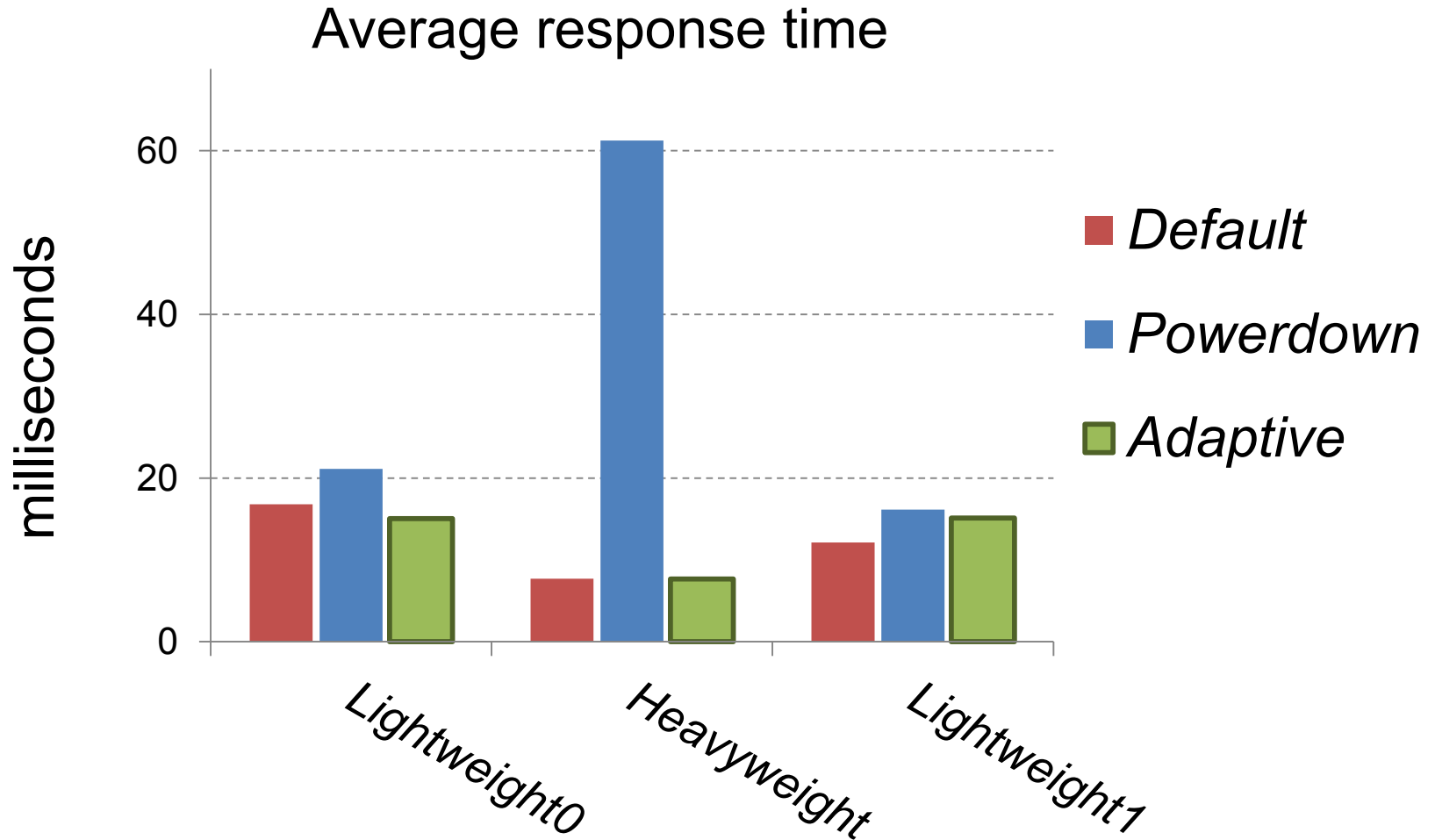
taken by DB manager
taken by other processes
free

Opportunity of energy savings

Energy consumption



Adaptively retained performance



Conclusion

- Opportunity to save energy in DRAM
 - Increasing energy consumption in memory resource
 - Unused memory in idle time
- Energy-aware database buffer pool management
 - Application driven approach
 - Threshold based heuristics
 - Energy saving without degraded performance
 - Extensible for wide applicability

Questions?

- Questions and Answers
- Author contact information

Chang.Bae@eecs.northwestern.edu, <http://www.changbae.org>

Jamel.Tayeb@intel.com

- Project website

Intel Energy Checker SDK

<http://software.intel.com/en-us/whatif>



E ECS

