Optimizing Storage and Memory Systems for Energy and Performance

Luis Useche

Dissertation Defense, July 2012

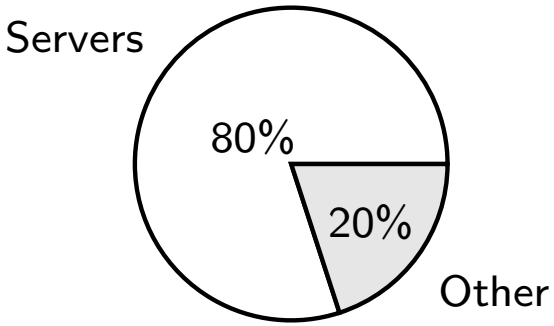
The Energy Problem

The U.S. Government spent $450 million in 2007 for data-center energy

The Energy Problem

The U.S. Government spent $450 million in 2007 for data-center energy

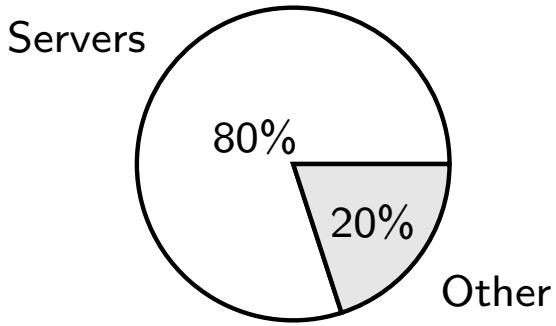
Datacenters Power Breakdown



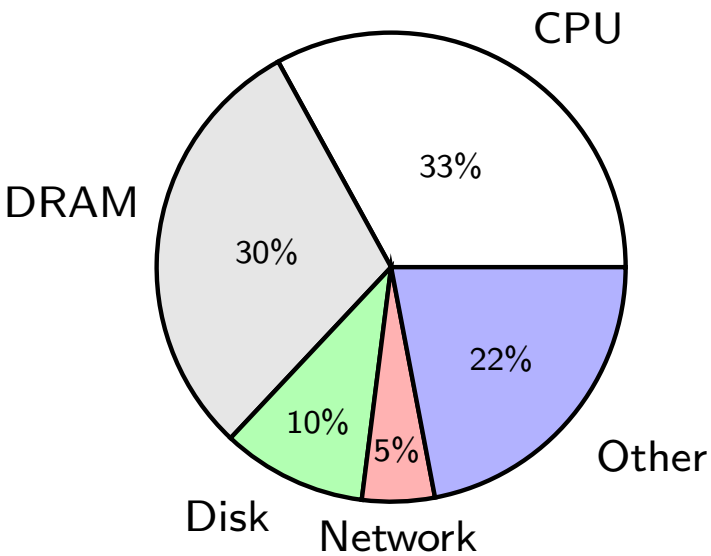
The Energy Problem

The U.S. Government spent $450 million in 2007 for data-center energy

Datacenters Power Breakdown



Server Power Breakdown

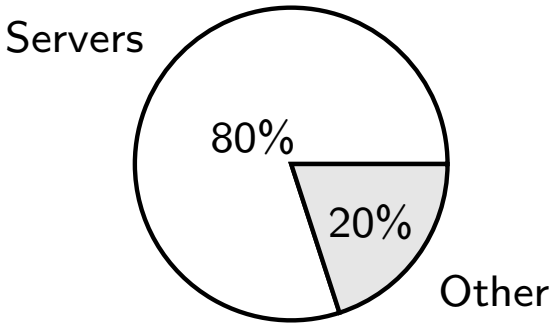


The Energy Problem

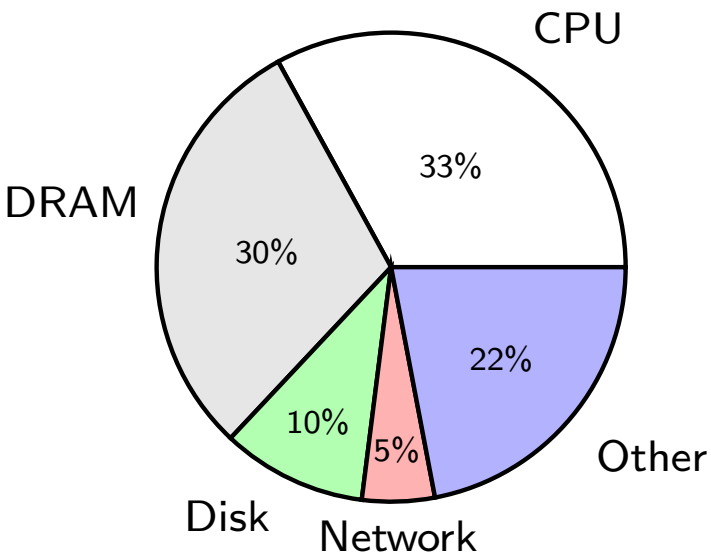
The U.S. Government spent $450 million in 2007 for data-center energy

- Disks and Memory account for 32% of data centers energy cost
- Or $144 Million
- Worse with increasing DRAM and storage needs

Datacenters Power Breakdown



Server Power Breakdown

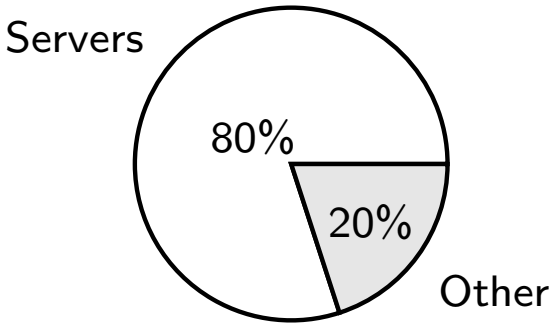


The Energy Problem

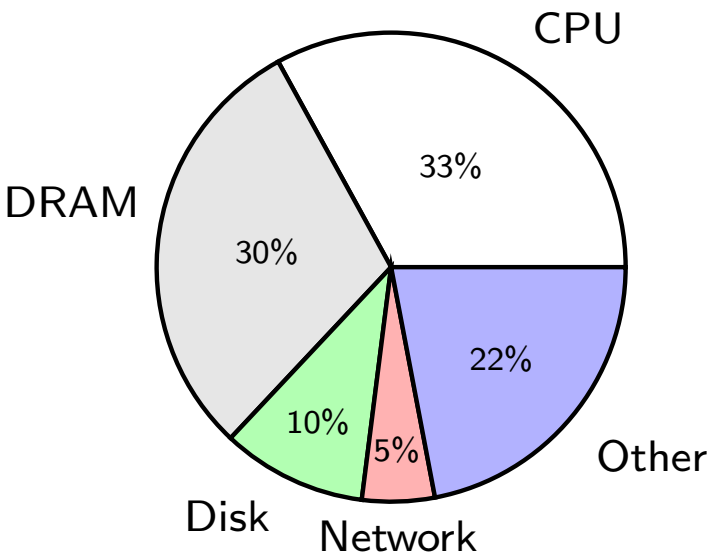
The U.S. Government spent $450 million in 2007 for data-center energy

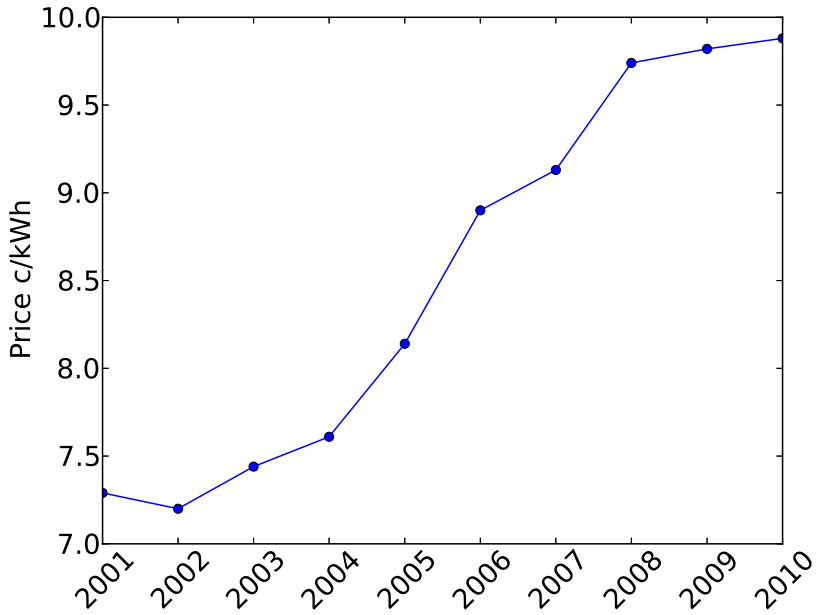
- Disks and Memory account for 32% of data centers energy cost
- Or $144 Million
- Worse with increasing DRAM and storage needs

Datacenters Power Breakdown



Server Power Breakdown

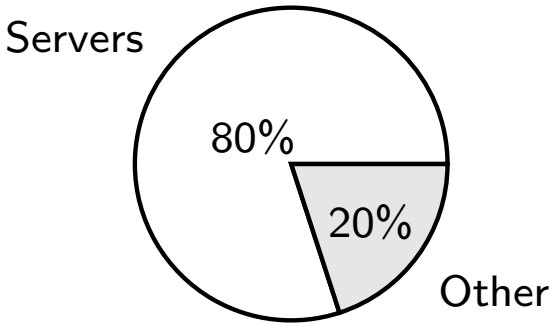




The Energy Problem

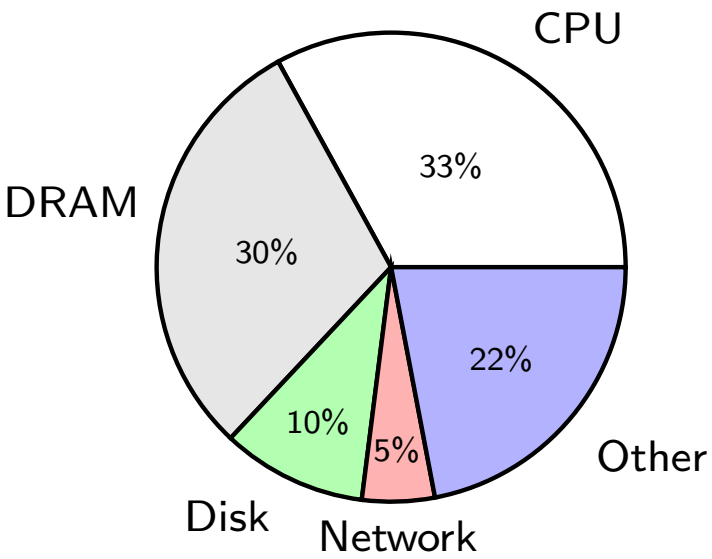
The U.S. Government spent $450 million in 2007 for data-center energy

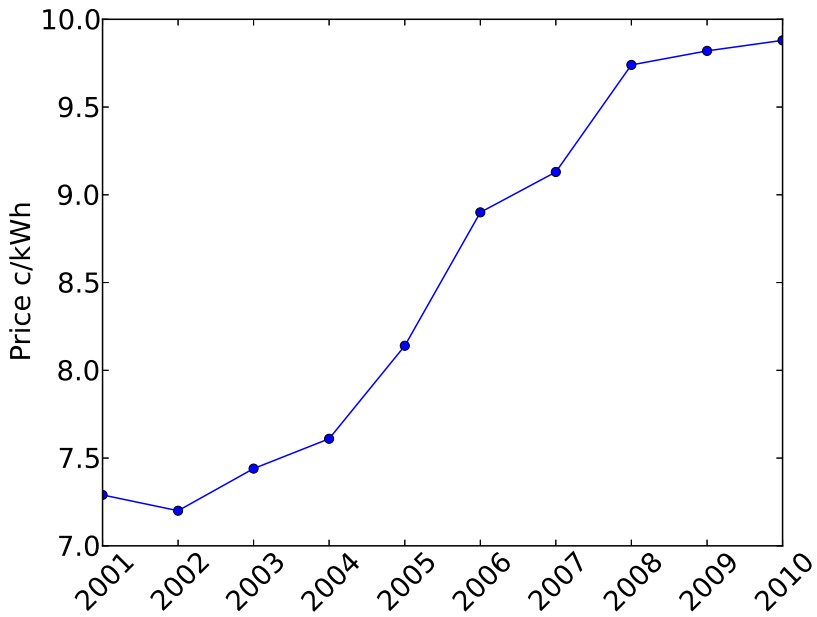
Datacenters Power Breakdown



- **Disks and Memory** account for 32% of data centers energy cost
- Or $144 Million
- Worse with increasing DRAM and storage needs

Server Power Breakdown





Benefits and Challenges

- Reduce data-centers operation cost
- Longer lasting battery on mobile devices

Benefits and Challenges

- Reduce data-centers operation cost
- Longer lasting battery on mobile devices

Challenge

Saving power usually trades-off performance for energy efficiency

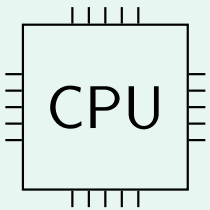
Benefits and Challenges

- Reduce data-centers operation cost
- Longer lasting battery on mobile devices

Challenge

Saving power usually trades-off performance for energy efficiency

Example



- Multiple power levels
- Lower power levels $\Rightarrow$ Lower performance

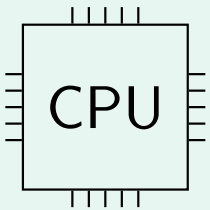
Benefits and Challenges

- Reduce data-centers operation cost
- Longer lasting battery on mobile devices

Challenge

Saving power usually trades-off performance for energy efficiency

Example

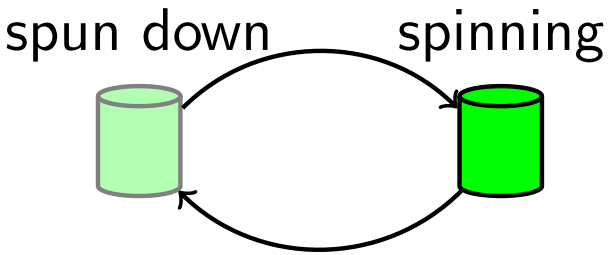


- Multiple power levels
- Lower power levels $\Rightarrow$ Lower performance

Example

- It continuously refreshes itself even when idle
- Exchanging DRAM for flash would increase paging activity

Disks are not energy proportional



- Two power consumption levels
- Can only save power when not rotating (spun down)
- Accesses while spun-down incur in delays of seconds
- Constant power consumption while spinning

Solutions

Disk

1. Spin-down the disk when not in use
   - ✓ Requests incur delay waiting for the disk to spin-up
   - ✓ Workloads do not allow long idle times

Solutions

Disk

1. Spin-down the disk when not in use
   - ✓ Requests incur delay waiting for the disk to spin-up
   - ✓ Workloads do not allow long idle times
2. Aggressive power-aware DRAM caching
   - ✓ Memory may not be big enough to hold the working set of a long time
   - ✓ Memory is not persistent with potential data loss

Solutions

Disk

1. Spin-down the disk when not in use
   - ✓ Requests incur delay waiting for the disk to spin-up
   - ✓ Workloads do not allow long idle times
2. Aggressive power-aware DRAM caching
   - ✓ Memory may not be big enough to hold the working set of a long time
   - ✓ Memory is not persistent with potential data loss
3. External Caching

Solutions

Disk

1. Spin-down the disk when not in use
   - ✓ Requests incur delay waiting for the disk to spin-up
   - ✓ Workloads do not allow long idle times
2. Aggressive power-aware DRAM caching
   - ✓ Memory may not be big enough to hold the working set of a long time
   - ✓ Memory is not persistent with potential data loss
3. External Caching
4. On multi-disk systems, duplicate data and use fewer disks

Solutions

Disk

1. Spin-down the disk when not in use
   - ✓ Requests incur delay waiting for the disk to spin-up
   - ✓ Workloads do not allow long idle times
2. Aggressive power-aware DRAM caching
   - ✓ Memory may not be big enough to hold the working set of a long time
   - ✓ Memory is not persistent with potential data loss
3. External Caching
4. On multi-disk systems, duplicate data and use fewer disks

Memory

1. Increase locality in DRAM
2. Use multi-power DRAM

Solutions

Disk

1. Spin-down the disk when not in use **[Helmbold et al. 00]**
   - ✓ Requests incur delay waiting for the disk to spin-up
   - ✓ Workloads do not allow long idle times
2. Aggressive power-aware DRAM caching **[Axboe 04]**
   - ✓ Memory may not be big enough to hold the working set of a long time
   - ✓ Memory is not persistent with potential data loss
3. External Caching **[Marsh et al. 94]** $\Leftarrow$ **Simulation**
4. On multi-disk systems, duplicate data and use fewer disks **[Pinheiro et al. 04]** $\Leftarrow$ **No proportionality**

Memory

1. Increase locality in DRAM **[Sudan et al. 10]** $\Leftarrow$ **Idle Power**
2. Use multi-power DRAM **[Deng et al. 11]** $\Leftarrow$ **Idle Power**

Thesis Statement

In this dissertation we aim to reduce the energy consumption of computer systems by pursuing three complementary research directions:

Thesis Statement

In this dissertation we aim to reduce the energy consumption of computer systems by pursuing three complementary research directions:

- designing a multi-disk energy proportional storage system using commodity devices (SRCMap),

Thesis Statement

In this dissertation we aim to reduce the energy consumption of computer systems by pursuing three complementary research directions:

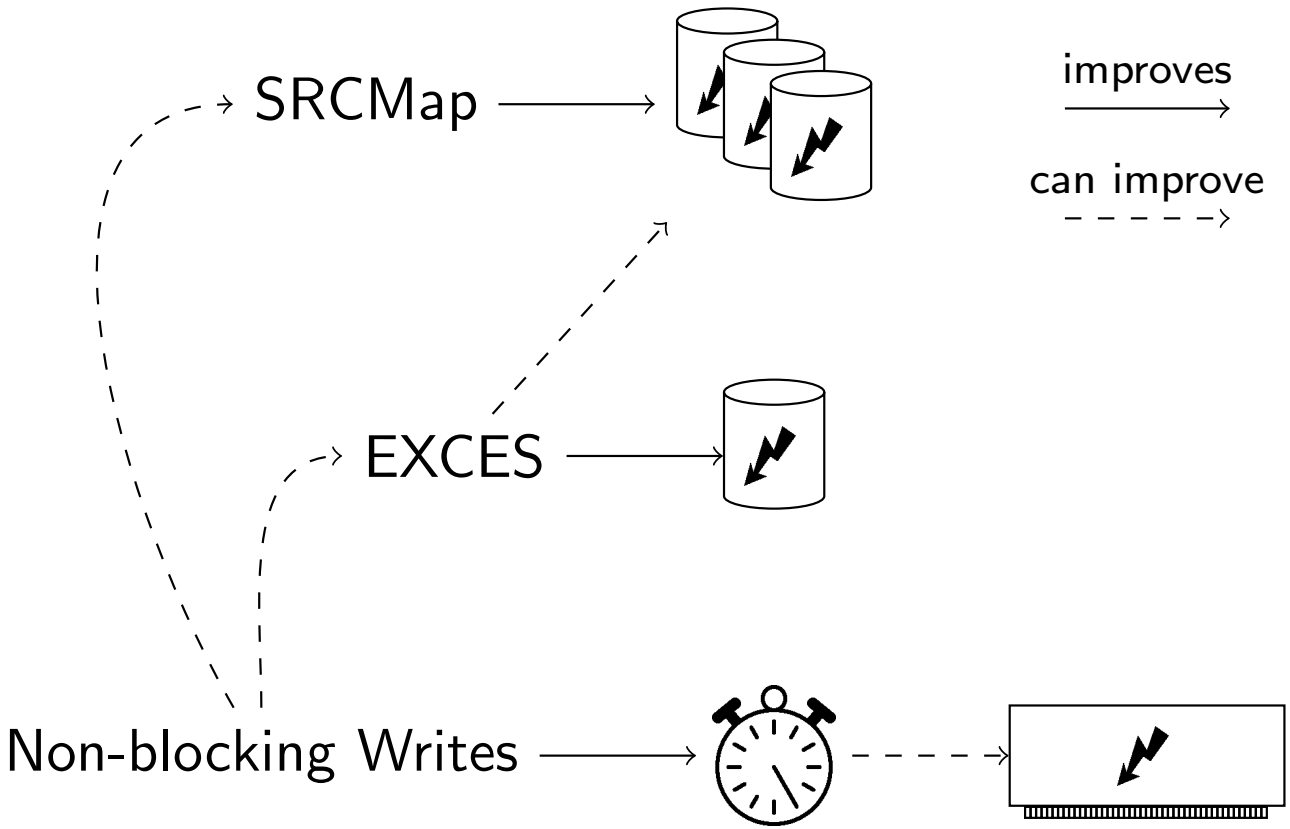
- designing a multi-disk energy proportional storage system using commodity devices (SRCMap),
- designing and implementing energy-efficient storage systems using flash devices as an external cache to reduce energy consumption in disks (EXCES), and

Thesis Statement

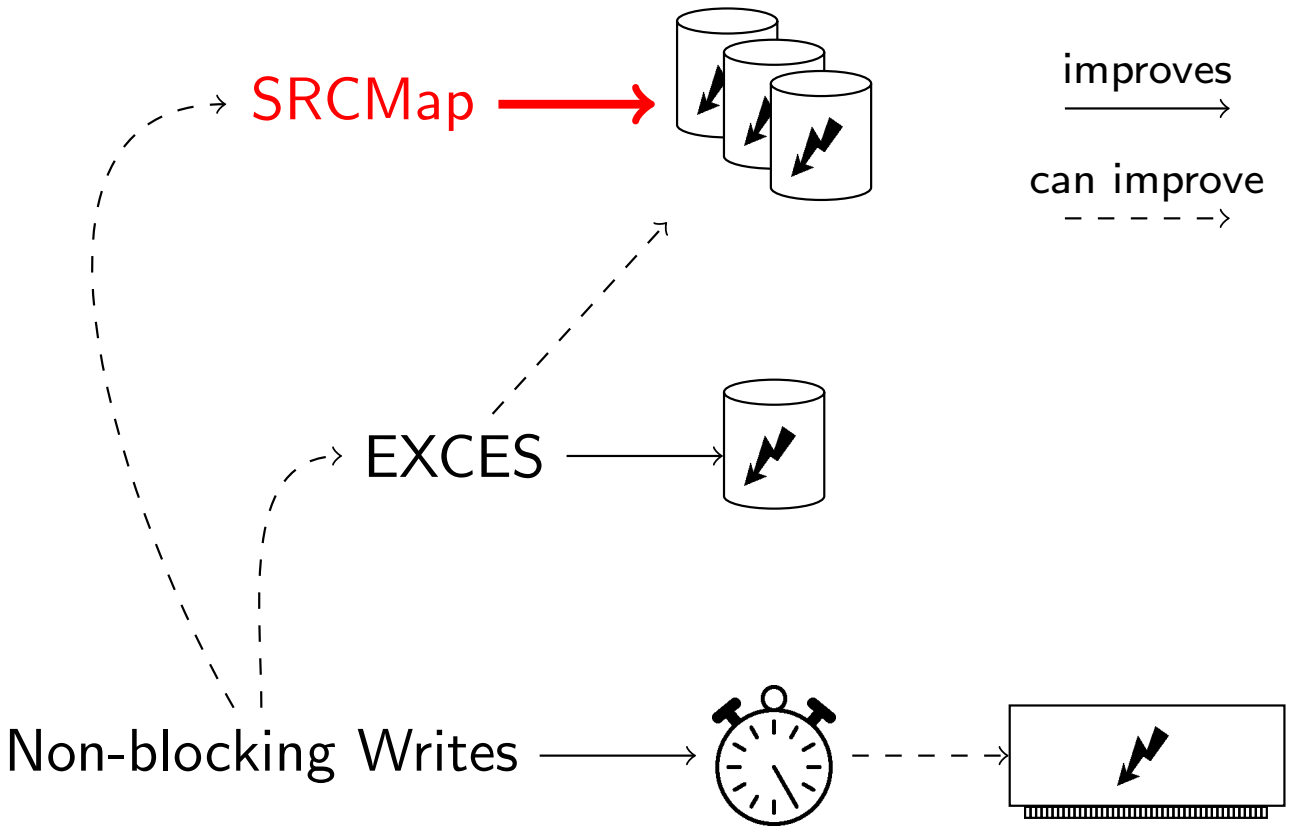
In this dissertation we aim to reduce the energy consumption of computer systems by pursuing three complementary research directions:

- designing a multi-disk energy proportional storage system using commodity devices (SRCMap),
- designing and implementing energy-efficient storage systems using flash devices as an external cache to reduce energy consumption in disks (EXCES), and
- designing and implementing a new mechanism to eliminate blocking read page operations due to page writes that could, otherwise, disturb disks in their low-power state and slowdown applications (Non-blocking Writes).

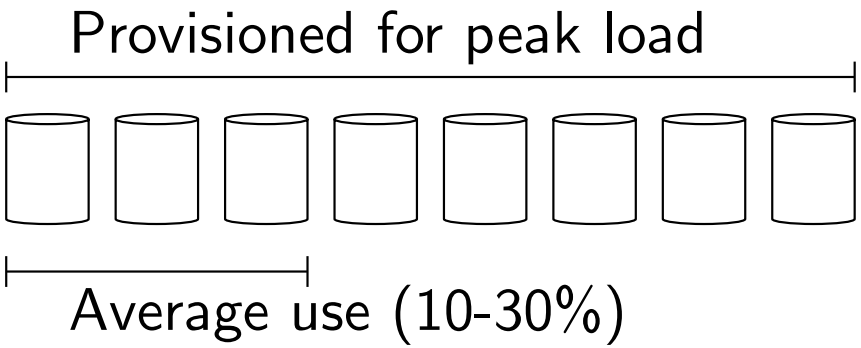
Contributions



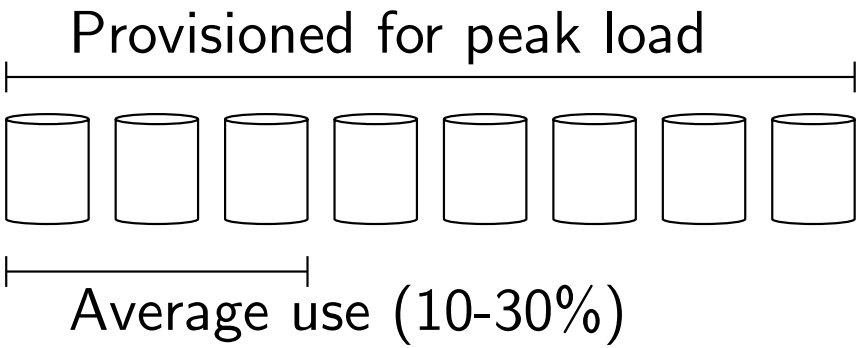
Contributions



Energy Proportional Multi-disk Systems

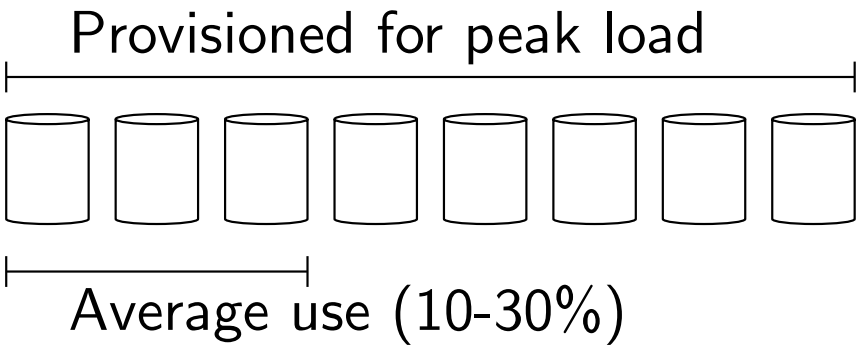


Energy Proportional Multi-disk Systems



- Consolidation is a well known technique for energy proportionality in virtualized servers

Energy Proportional Multi-disk Systems



- Consolidation is a well known technique for energy proportionality in virtualized servers

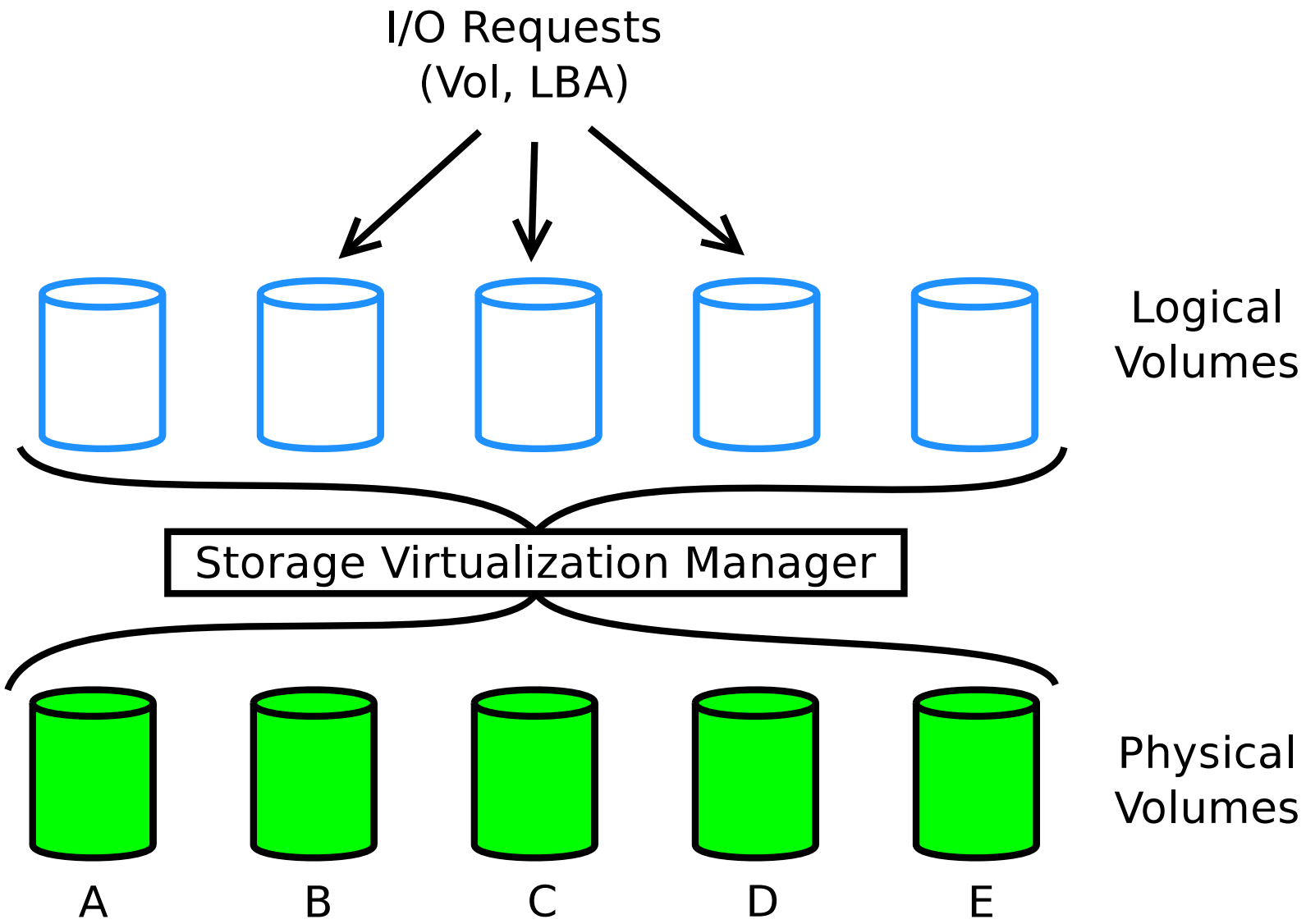
Can we use a storage virtualization layer to design a practical energy proportional storage system?

- Storage virtualization I/O indirection useful for consolidation.

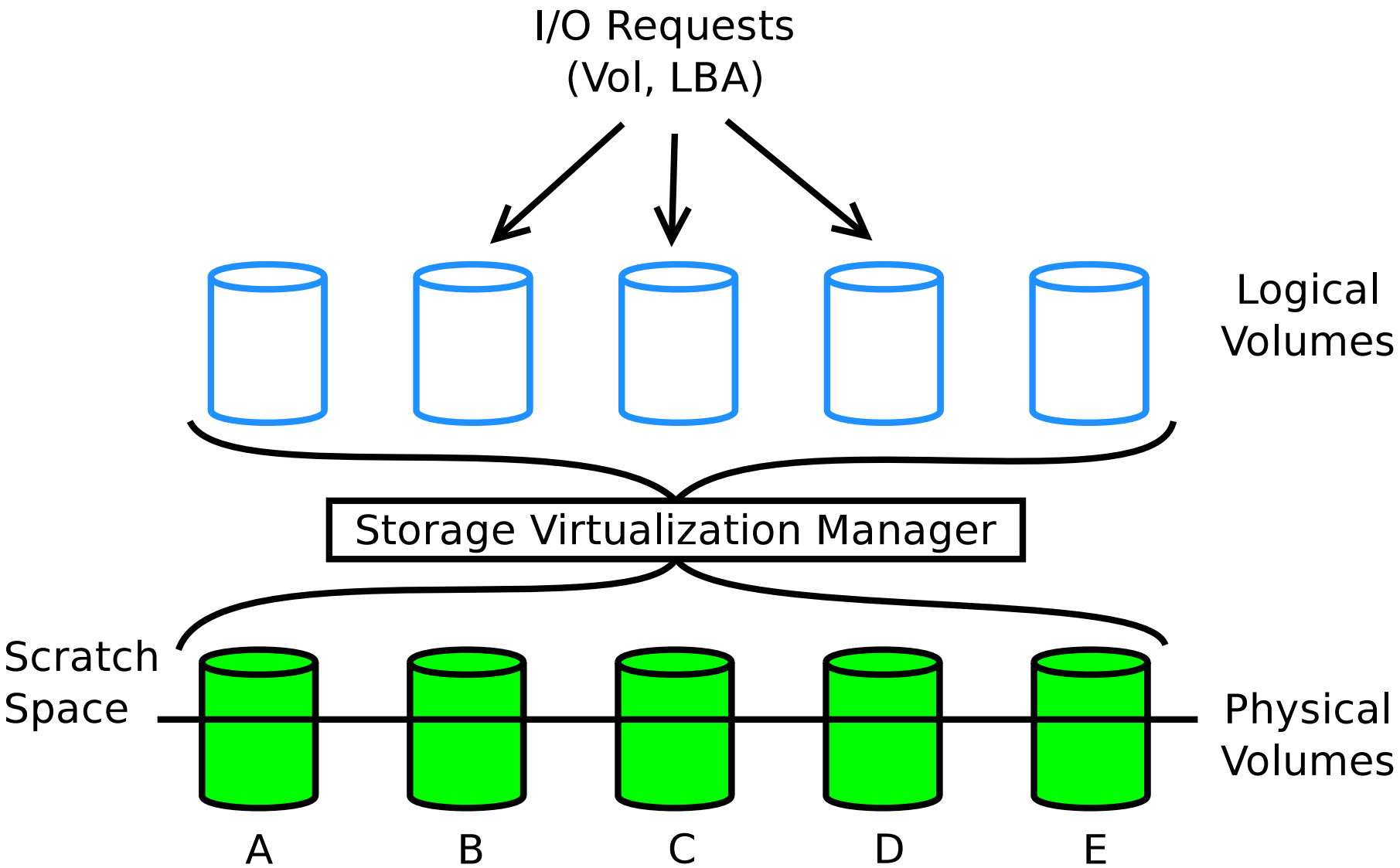
Challenge

Moving logical volumes from one device to another is prohibitively expensive.

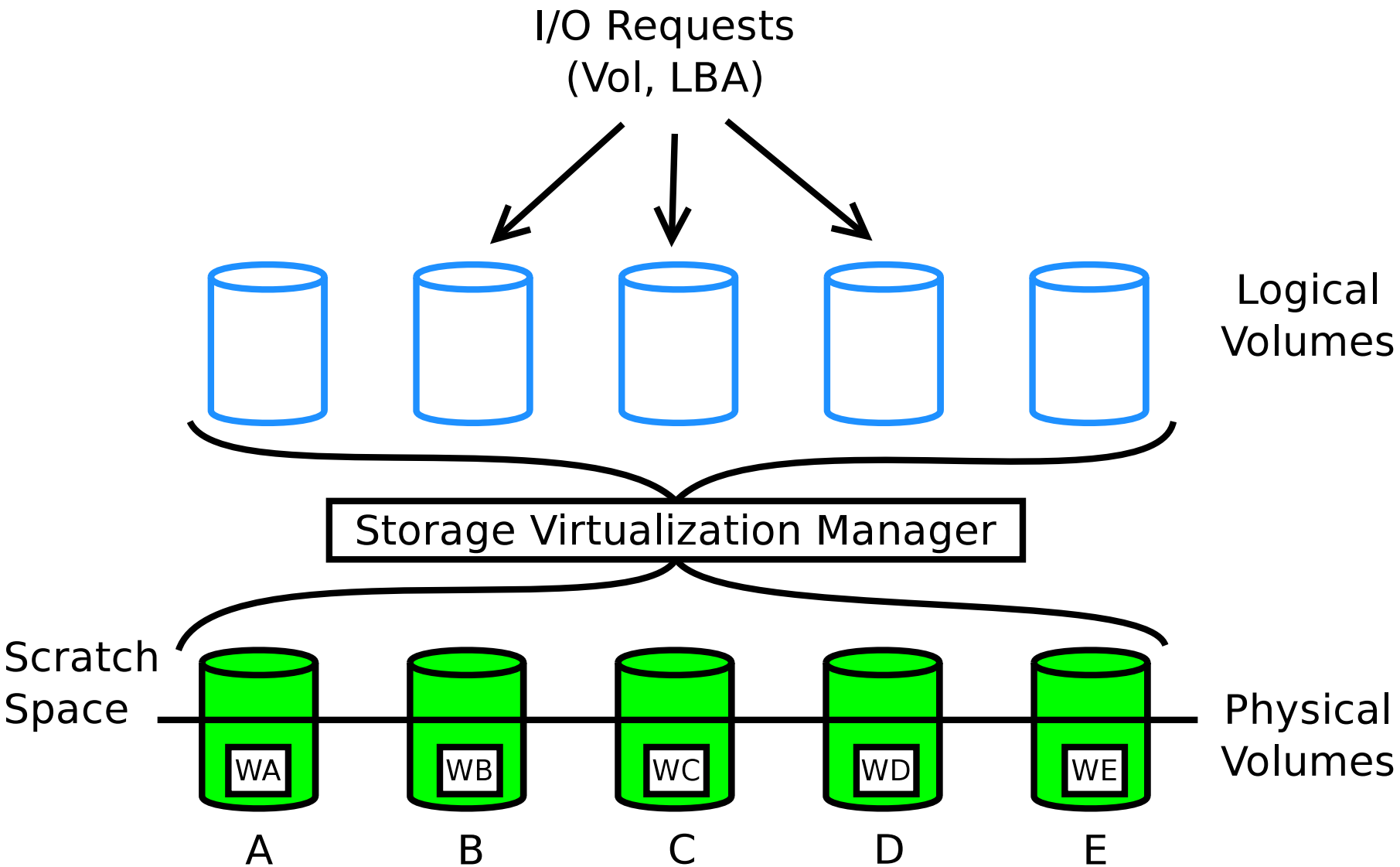
SRCMap: Overview



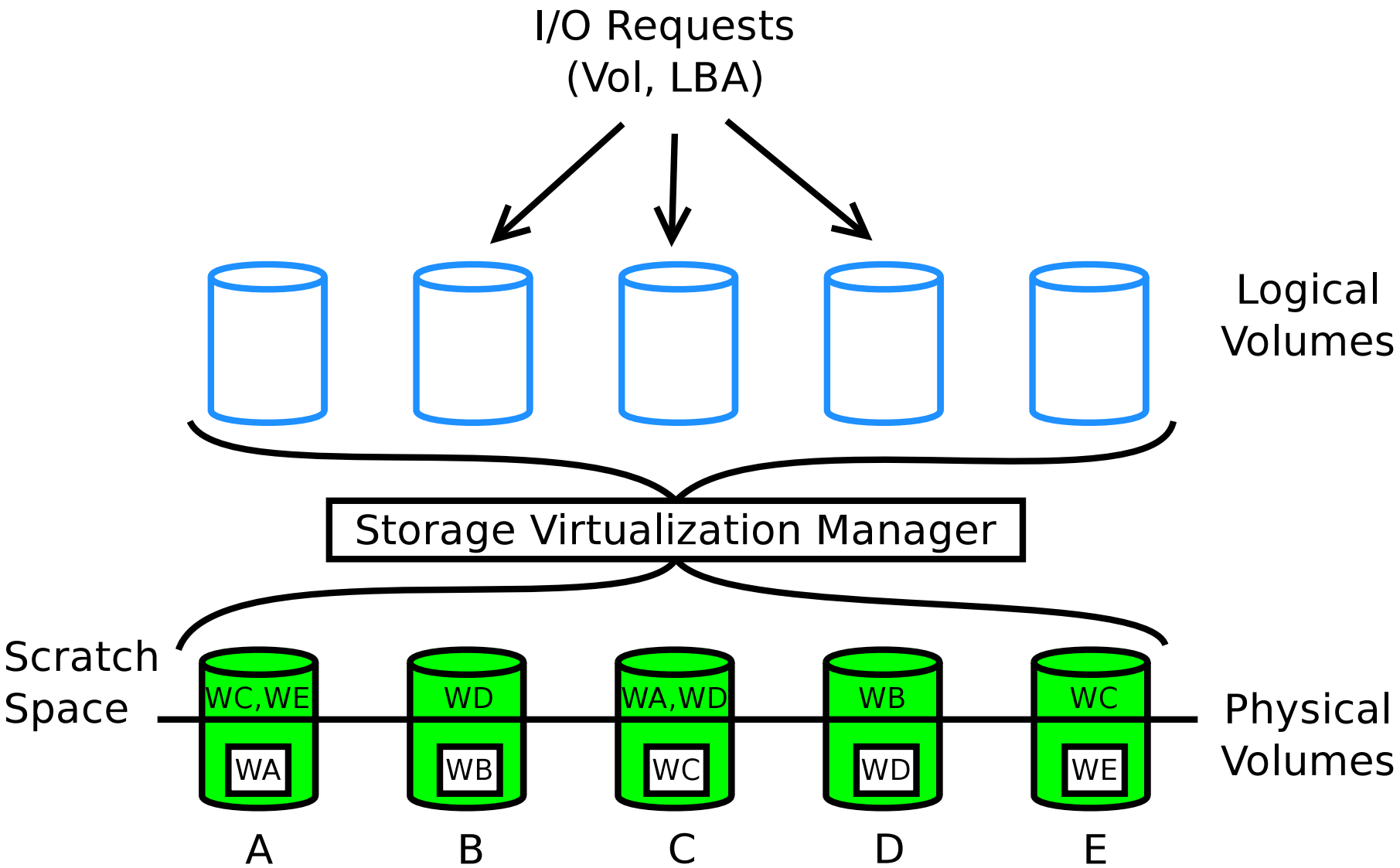
SRCMap: Overview



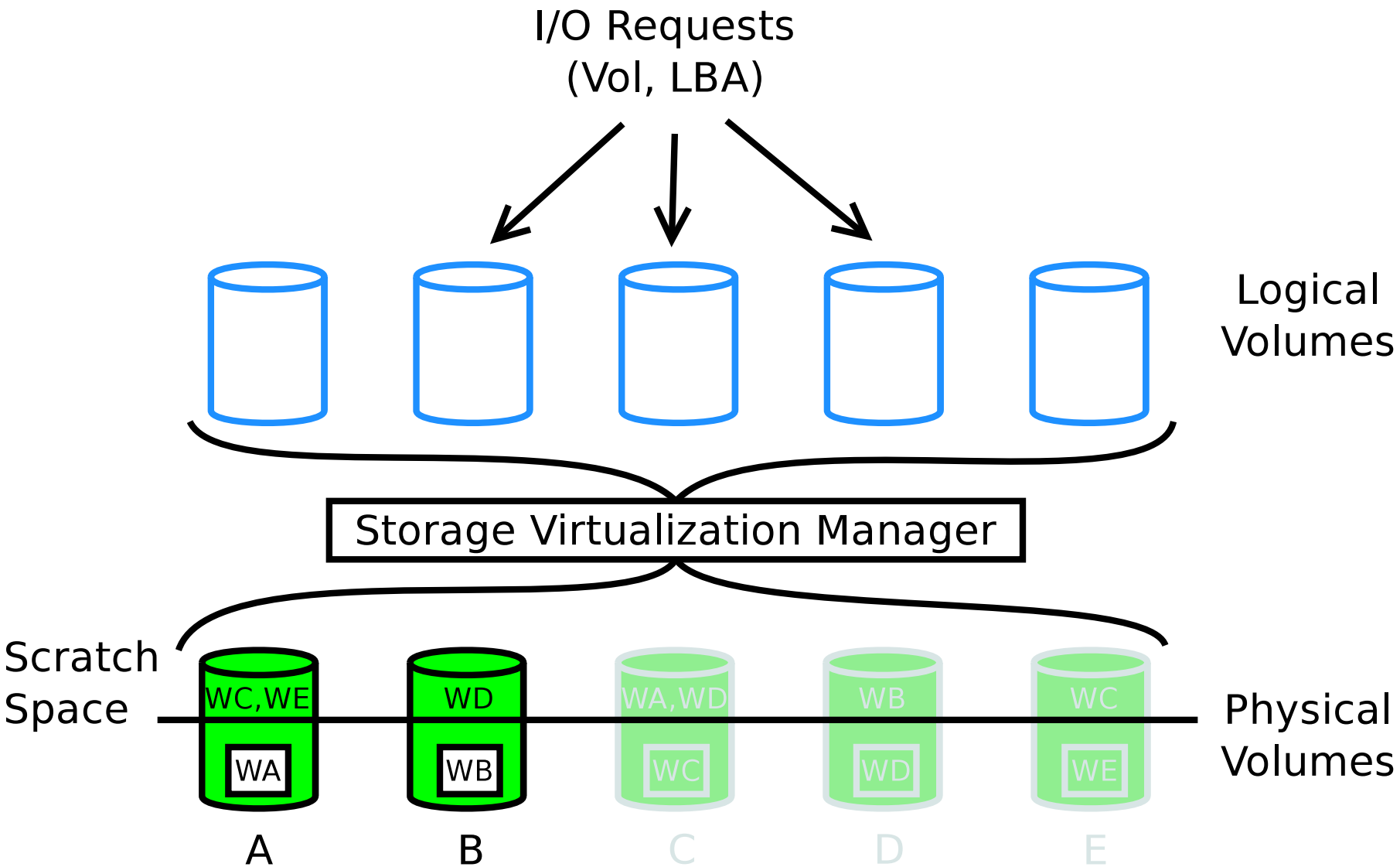
SRCMap: Overview



SRCMap: Overview



SRCMap: Overview



SRCMap: Summary

- Achieves one energy level per-disk, low space overhead, and workload shift adaptation

SRCMap: Summary

- Achieves one energy level per-disk, low space overhead, and workload shift adaptation
- Implemented and evaluated a prototype with real workload block traces
  - ✓ 35-59% of power savings
  - ✓ At least 50% of disks powered-off in an 8-disk server environment
  - ✓ Only 0.003% of I/Os had spin-up delay
  - ✗ Synchronization I/Os can affect performance

SRCMap: Summary

- Achieves one energy level per-disk, low space overhead, and workload shift adaptation
- Implemented and evaluated a prototype with real workload block traces
  - ✓ 35-59% of power savings
  - ✓ At least 50% of disks powered-off in an 8-disk server environment
  - ✓ Only 0.003% of I/Os had spin-up delay
  - ✗ Synchronization I/Os can affect performance

Impact

- Power savings with little initial manual tuning required
- Could achieve even better results in data-centers

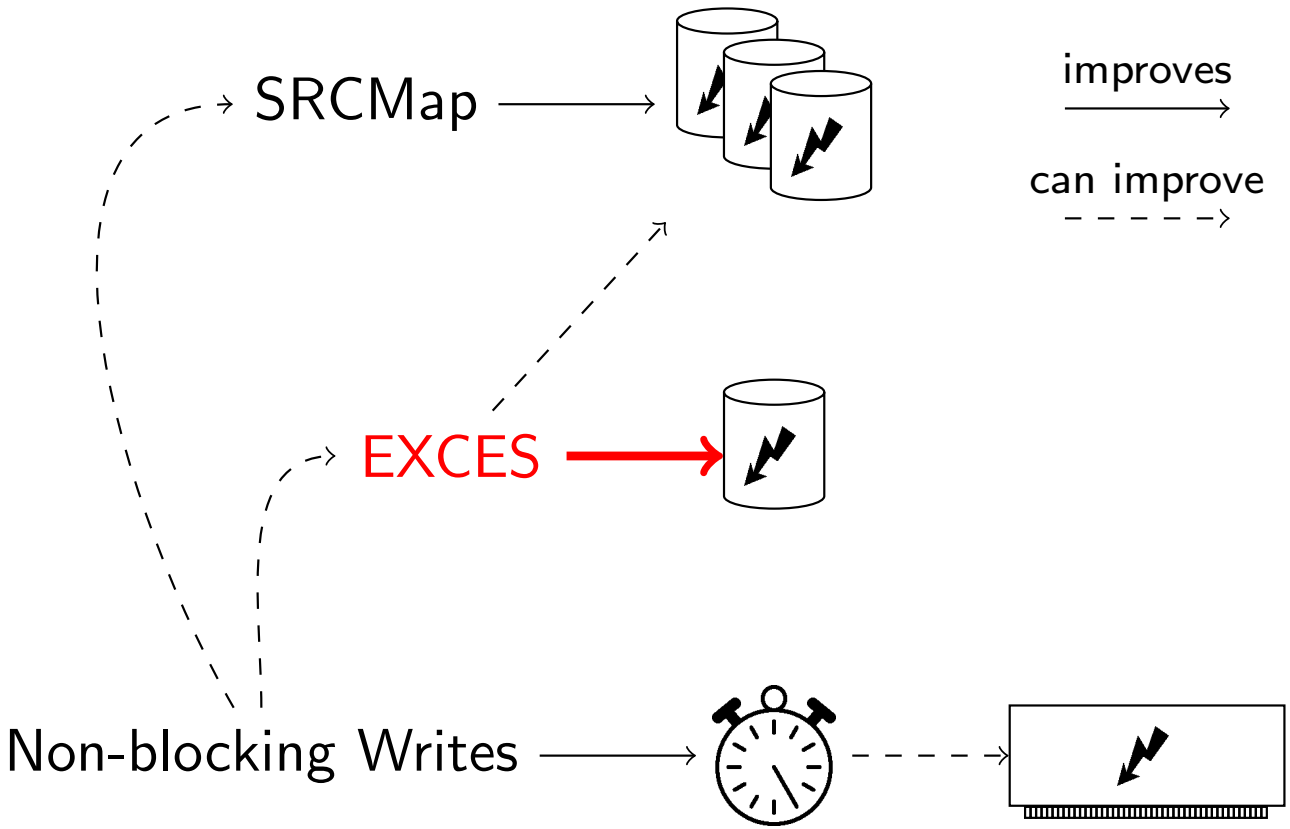
SRCMap: Summary

- Achieves one energy level per-disk, low space overhead, and workload shift adaptation
- Implemented and evaluated a prototype with real workload block traces
  - ✓ 35-59% of power savings
  - ✓ At least 50% of disks powered-off in an 8-disk server environment
  - ✓ Only 0.003% of I/Os had spin-up delay
  - ✗ Synchronization I/Os can affect performance

Impact

- Power savings with little initial manual tuning required
- Could achieve even better results in data-centers

Contributions



EXCES: External Caching for Energy-efficient Storage

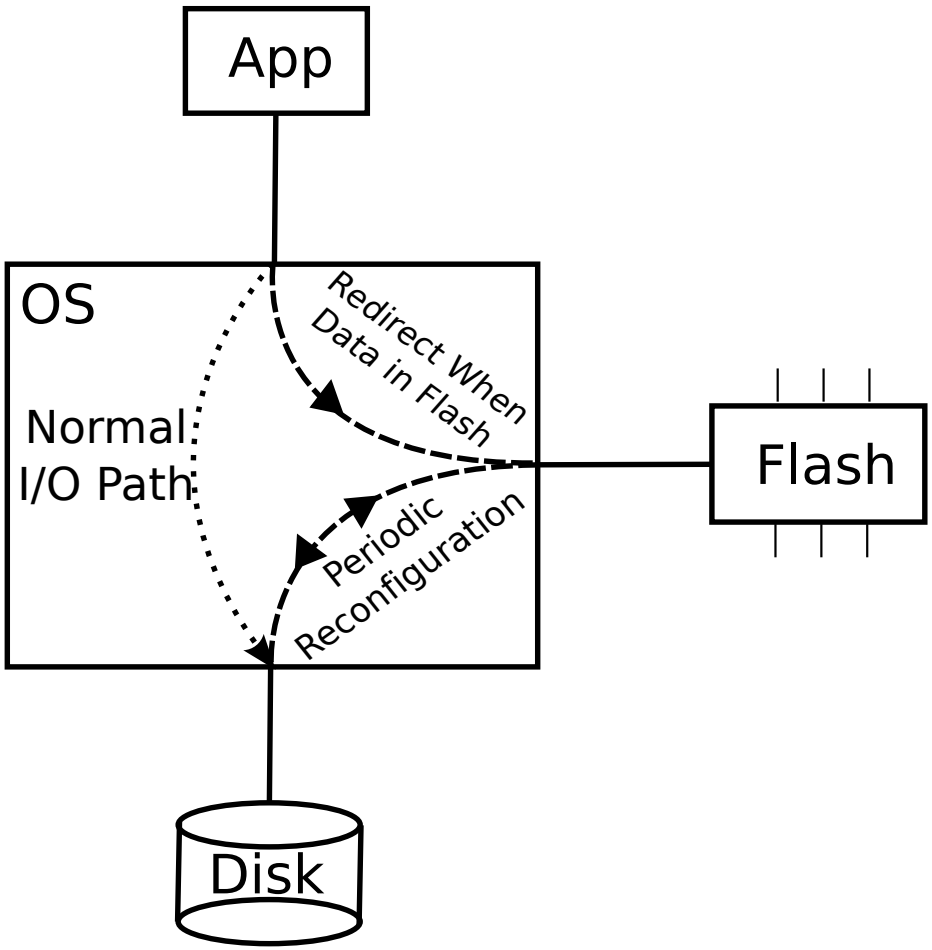
Goals

- Increase disk inactivity by: caching, prefetching, and write buffering in flash device
- Make effective use of external flash caching by adapting to workload changes
- Ensure block-level consistency in all system states

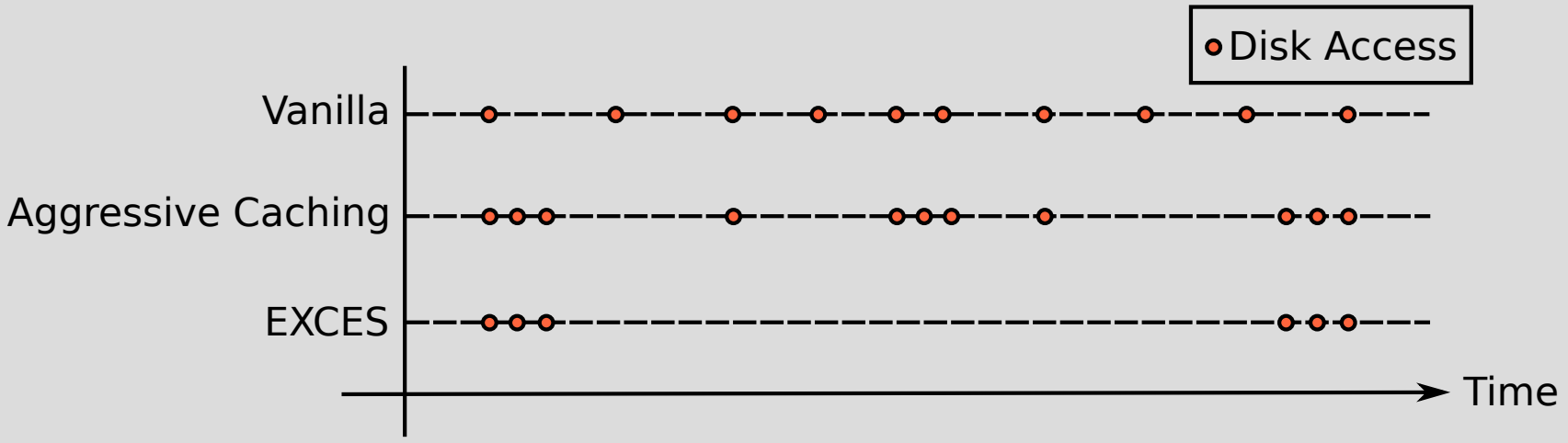
EXCES: External Caching for Energy-efficient Storage

Goals

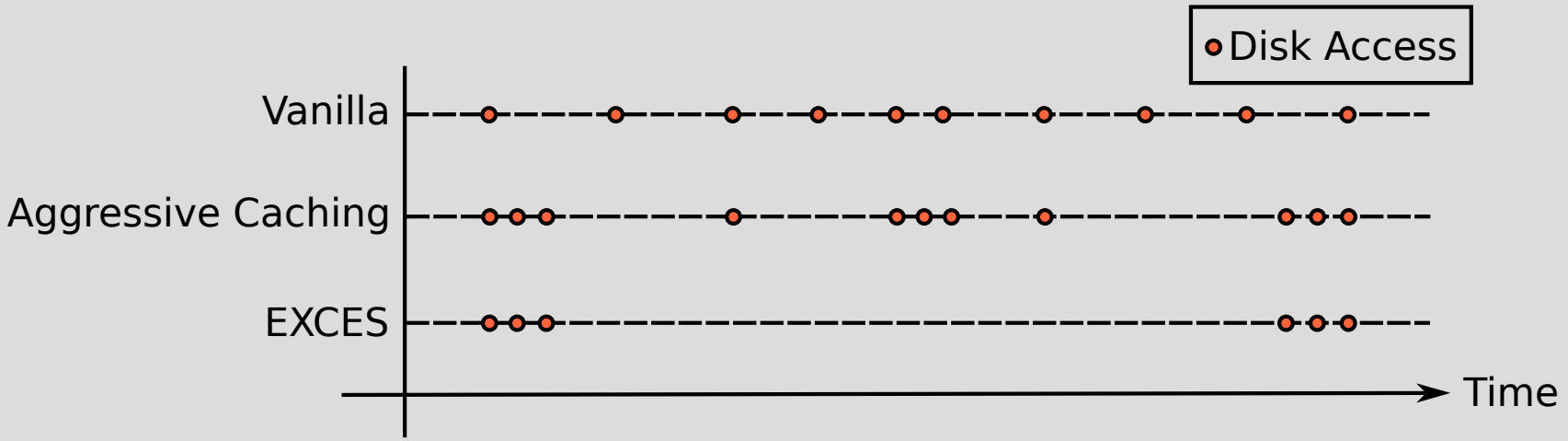
- Increase disk inactivity by: caching, prefetching, and write buffering in flash device
- Make effective use of external flash caching by adapting to workload changes
- Ensure block-level consistency in all system states



Disk Activity Example



Disk Activity Example



- First implementation of energy efficient external caching
- We designed and implemented EXCES for Linux
- We evaluated EXCES in a laptop using:
  - ✓ USB and CF external device
  - ✓ Desktop, developer, and server workloads

EXCES: Summary

- ▶ Overall savings of 2-14% across desktop, developer, and server workloads
- ✗ External caching can have a substantial impact in performance (up to 14x)

EXCES: Summary

- ▶ Overall savings of 2-14% across desktop, developer, and server workloads
- ✗ External caching can have a substantial impact in performance (up to 14x)

Impact

- ▶ Better energy efficiency in mobile devices by simply plugging an external device

So far...

- Both SRCMap and EXCES are cache systems
- Their effectiveness depends on the number of cache hits
- Caching is performed at page granularity

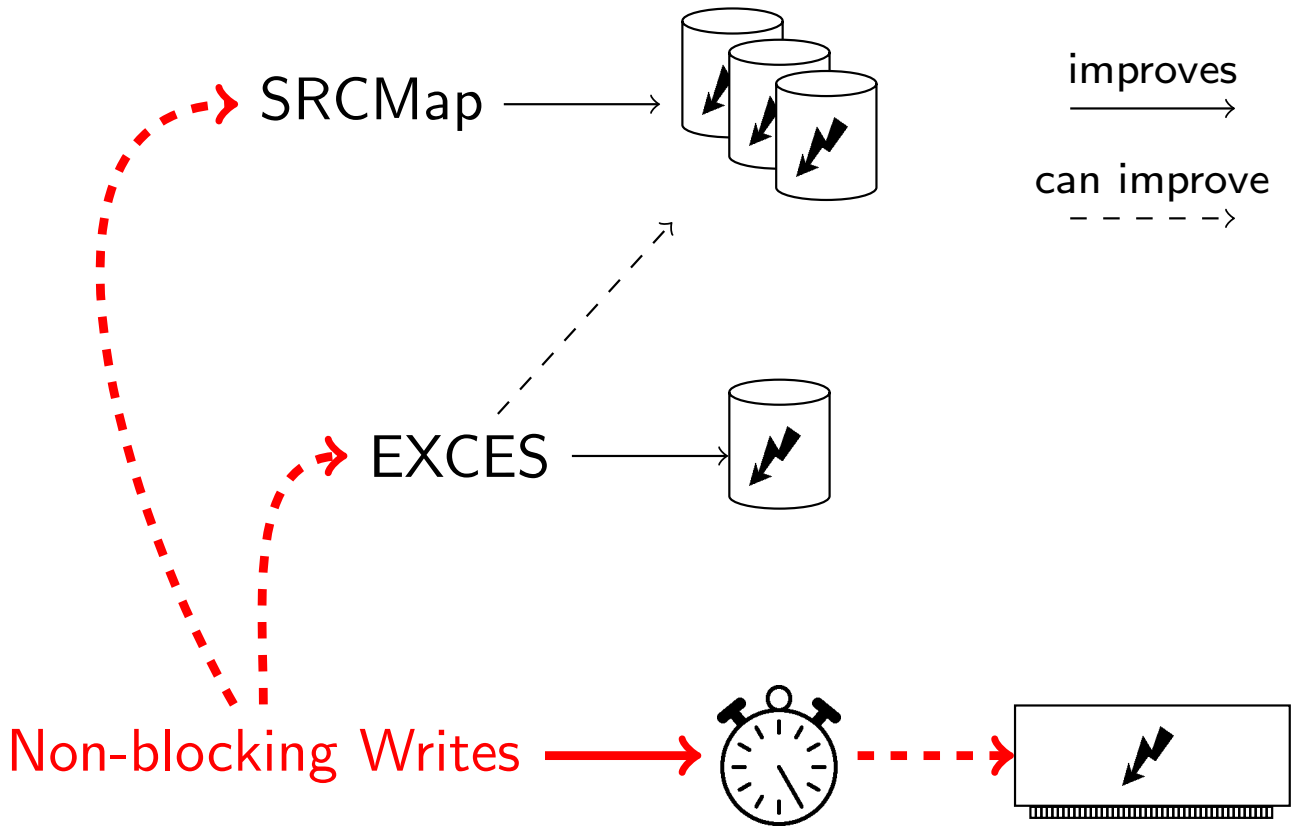
So far...

- Both SRCMap and EXCES are cache systems
- Their effectiveness depends on the number of cache hits
- Caching is performed at page granularity

Handling Misses

- Reads to non-cached pages still require disk attention
- Full-page writes can be buffered in cache
- Sub-page writes require disk attention

Contributions



I/O Traffic Control with Non-blocking Writes

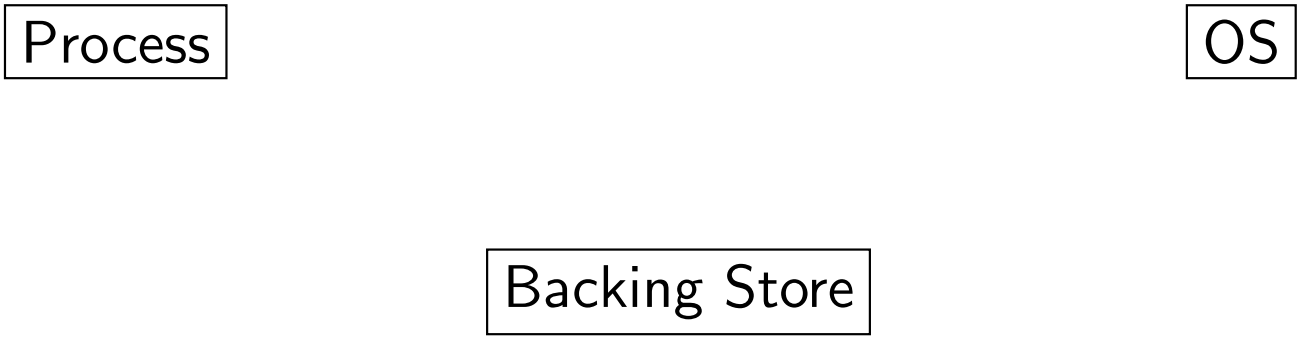
- Initially focused on OS caches: FS and VM

I/O Traffic Control with Non-blocking Writes

- Initially focused on OS caches: FS and VM
- Memory access granularity is smaller than disk's

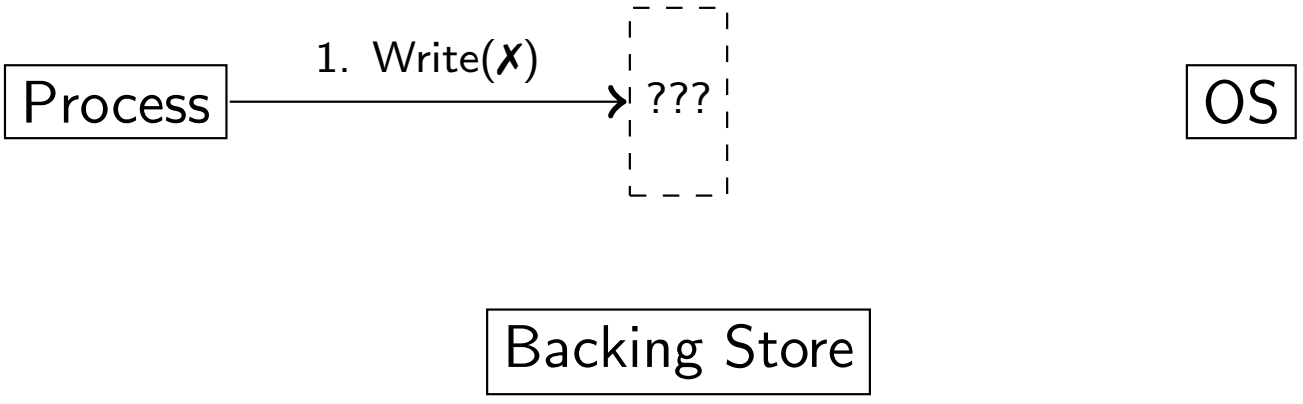
I/O Traffic Control with Non-blocking Writes

- Initially focused on OS caches: FS and VM
- Memory access granularity is smaller than disk's $\Rightarrow$ Writes to an out-of-core page require a full page fetch.



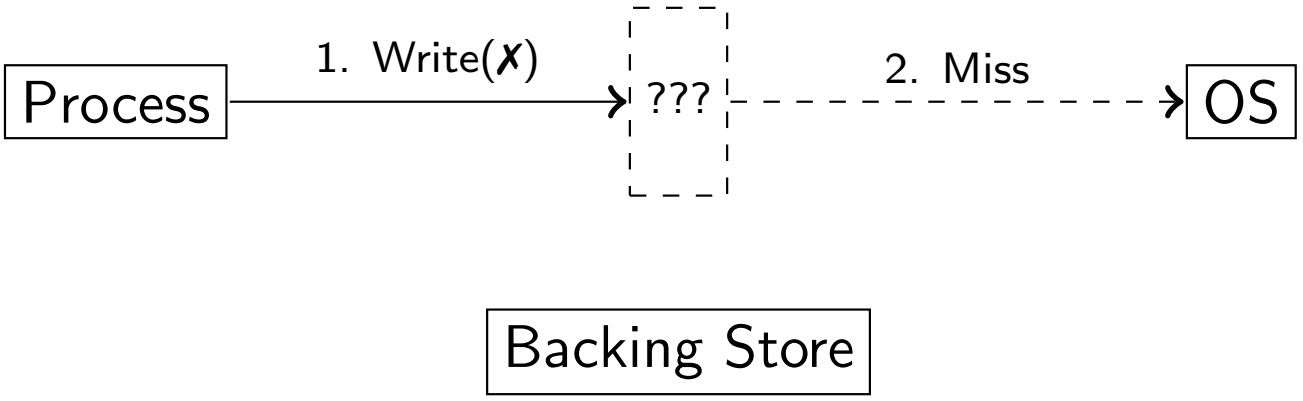
I/O Traffic Control with Non-blocking Writes

- Initially focused on OS caches: FS and VM
- Memory access granularity is smaller than disk's $\Rightarrow$ Writes to an out-of-core page require a full page fetch.



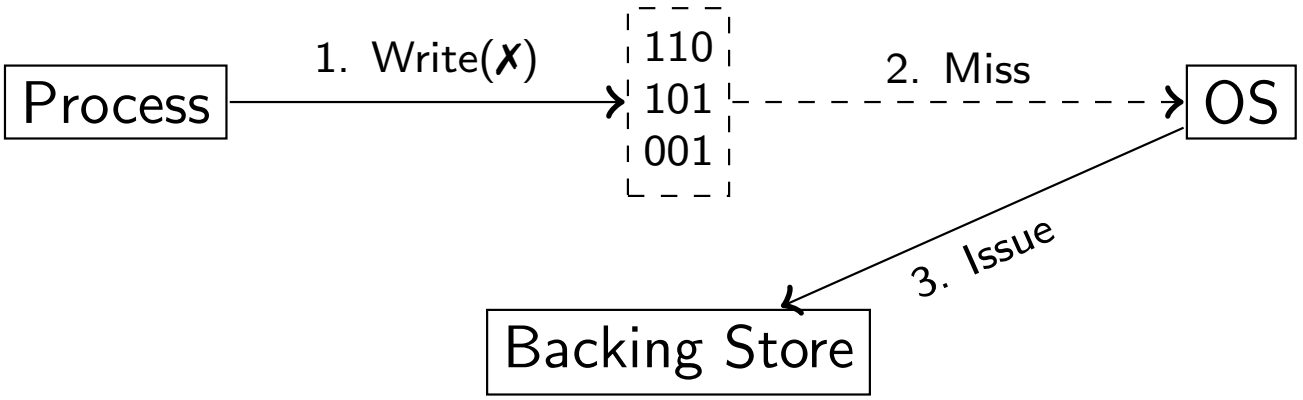
I/O Traffic Control with Non-blocking Writes

- Initially focused on OS caches: FS and VM
- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.



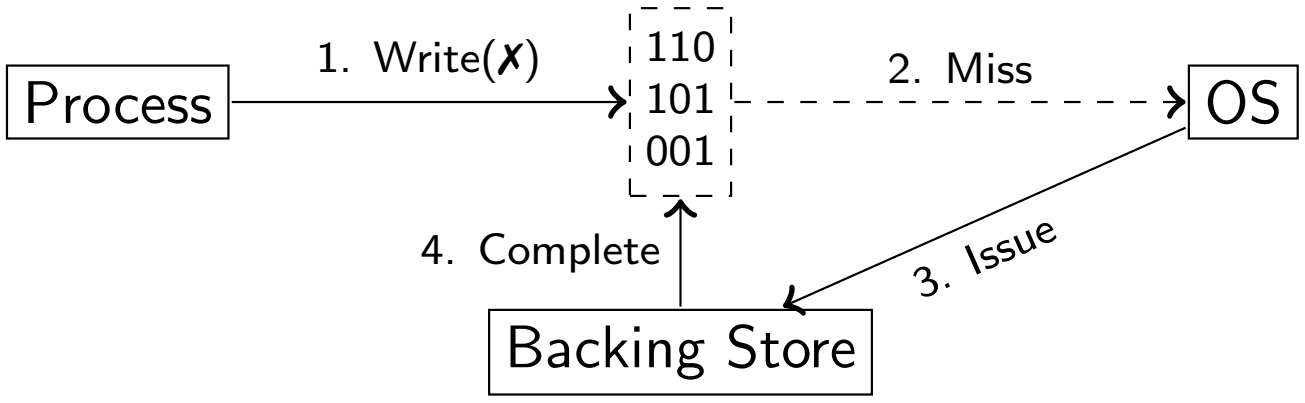
I/O Traffic Control with Non-blocking Writes

- Initially focused on OS caches: FS and VM
- Memory access granularity is smaller than disk's $\Rightarrow$ Writes to an out-of-core page require a full page fetch.



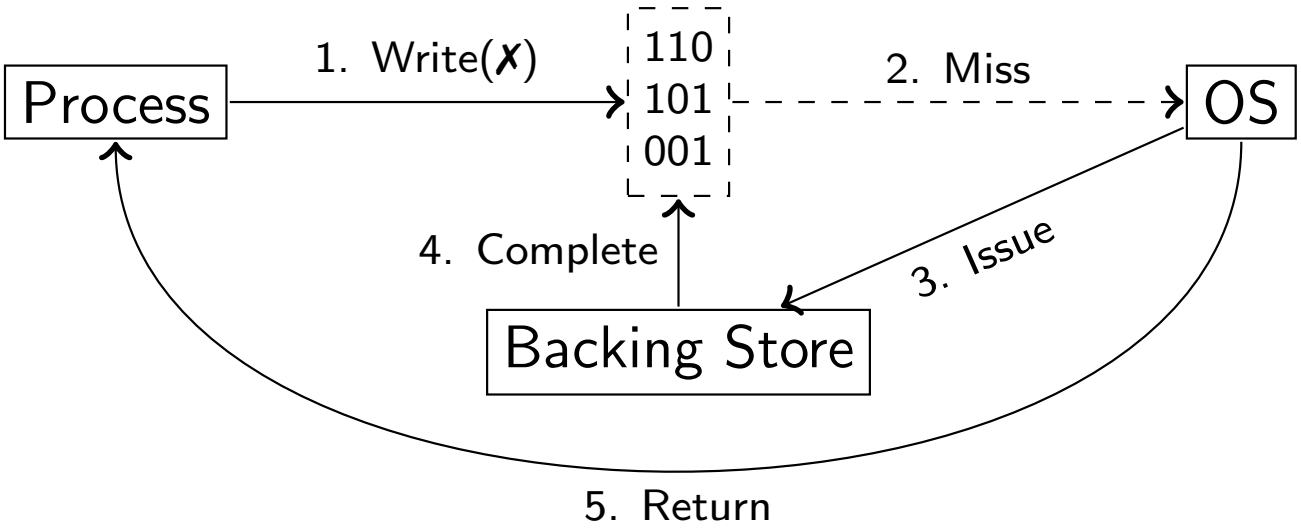
I/O Traffic Control with Non-blocking Writes

- Initially focused on OS caches: FS and VM
- Memory access granularity is smaller than disk's $\Rightarrow$ Writes to an out-of-core page require a full page fetch.



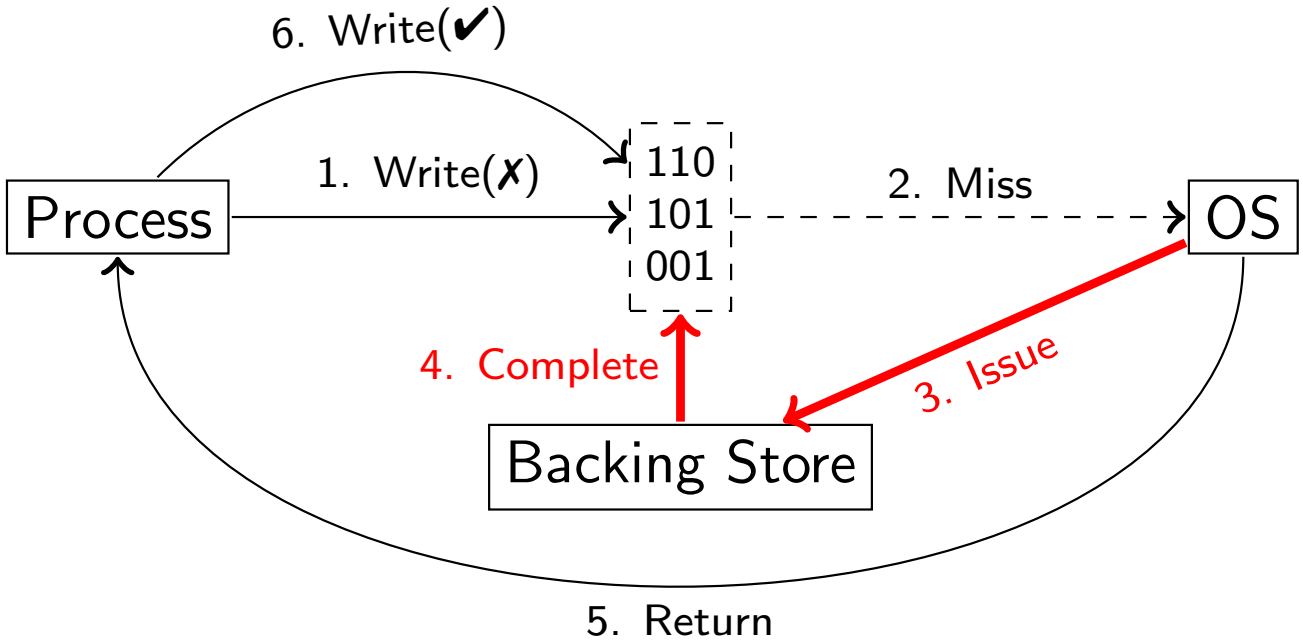
I/O Traffic Control with Non-blocking Writes

- Initially focused on OS caches: FS and VM
- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.



I/O Traffic Control with Non-blocking Writes

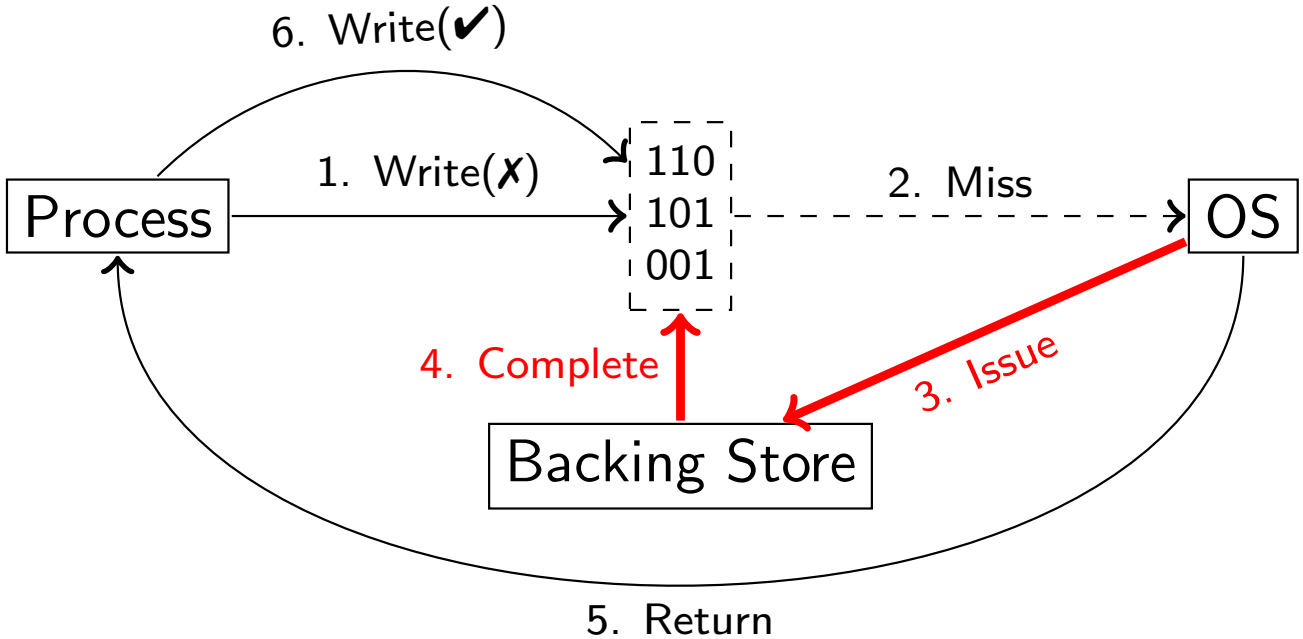
- Initially focused on OS caches: FS and VM
- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.



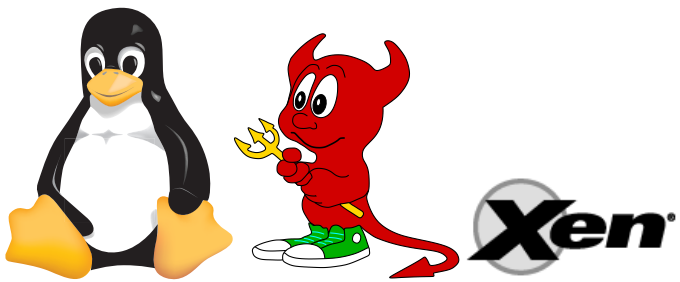
For writes: why request and wait for data that the application doesn't need?

I/O Traffic Control with Non-blocking Writes

- Initially focused on OS caches: FS and VM
- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.

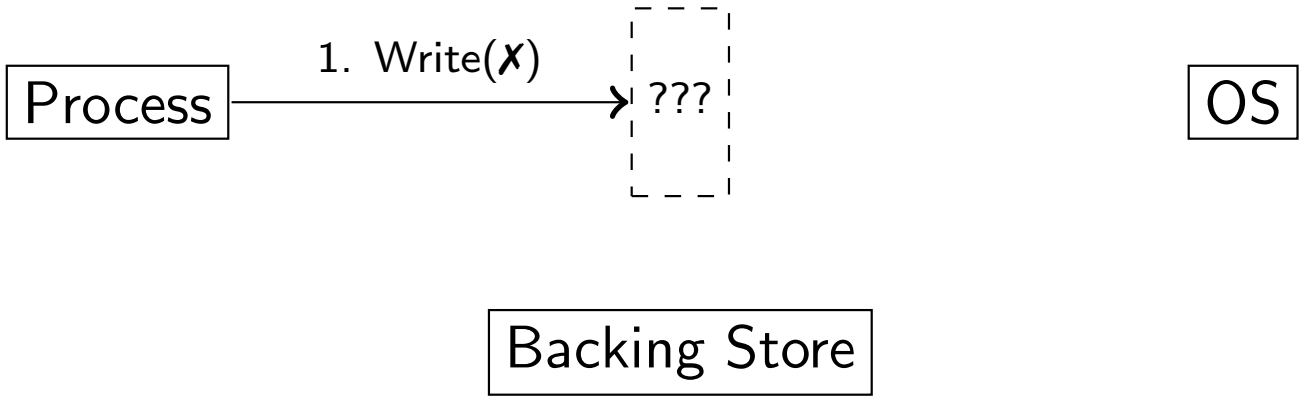


For writes: why request and wait for data that the application doesn't need?

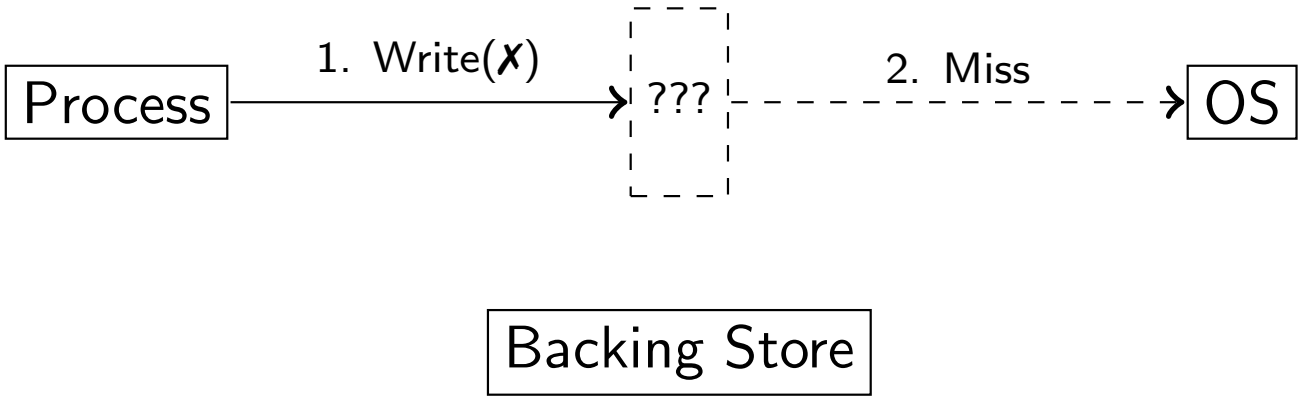


Non-blocking Writes: Basic Approach

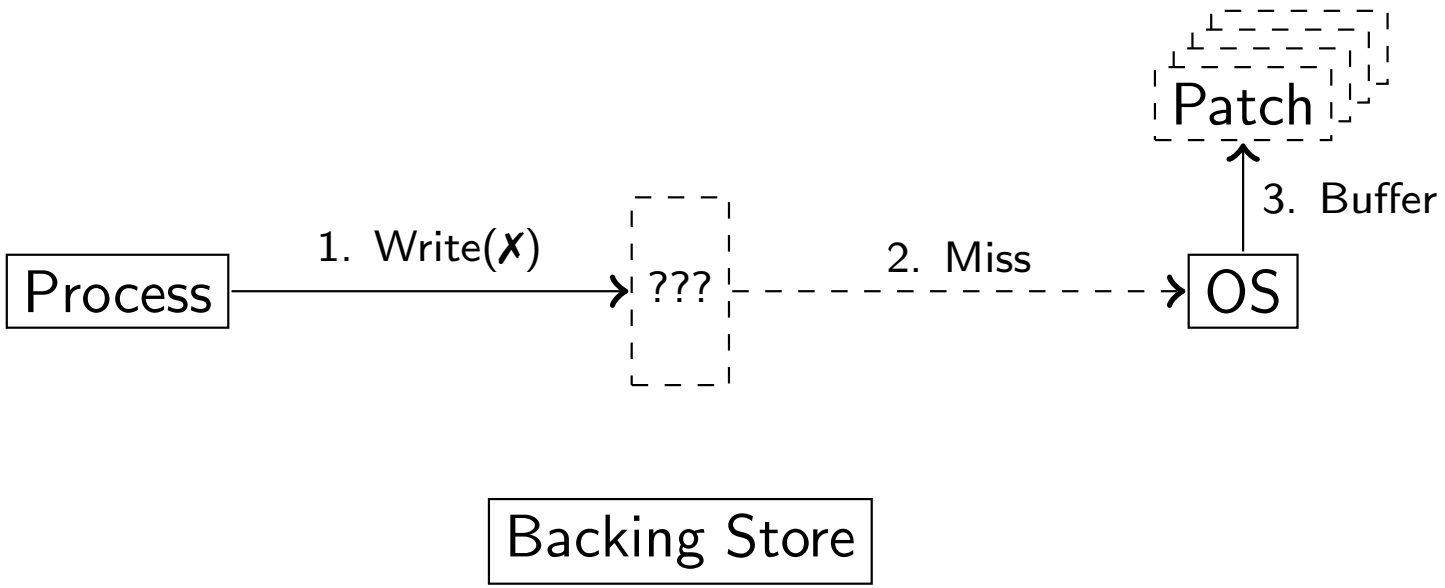
Non-blocking Writes: Basic Approach



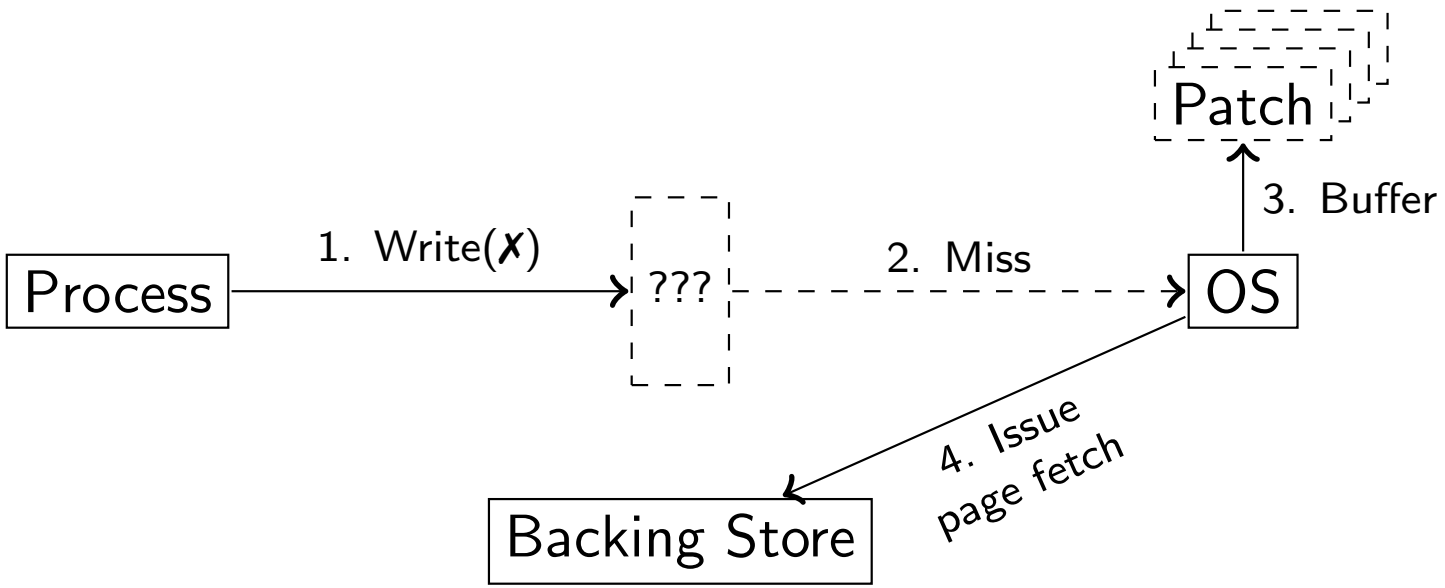
Non-blocking Writes: Basic Approach



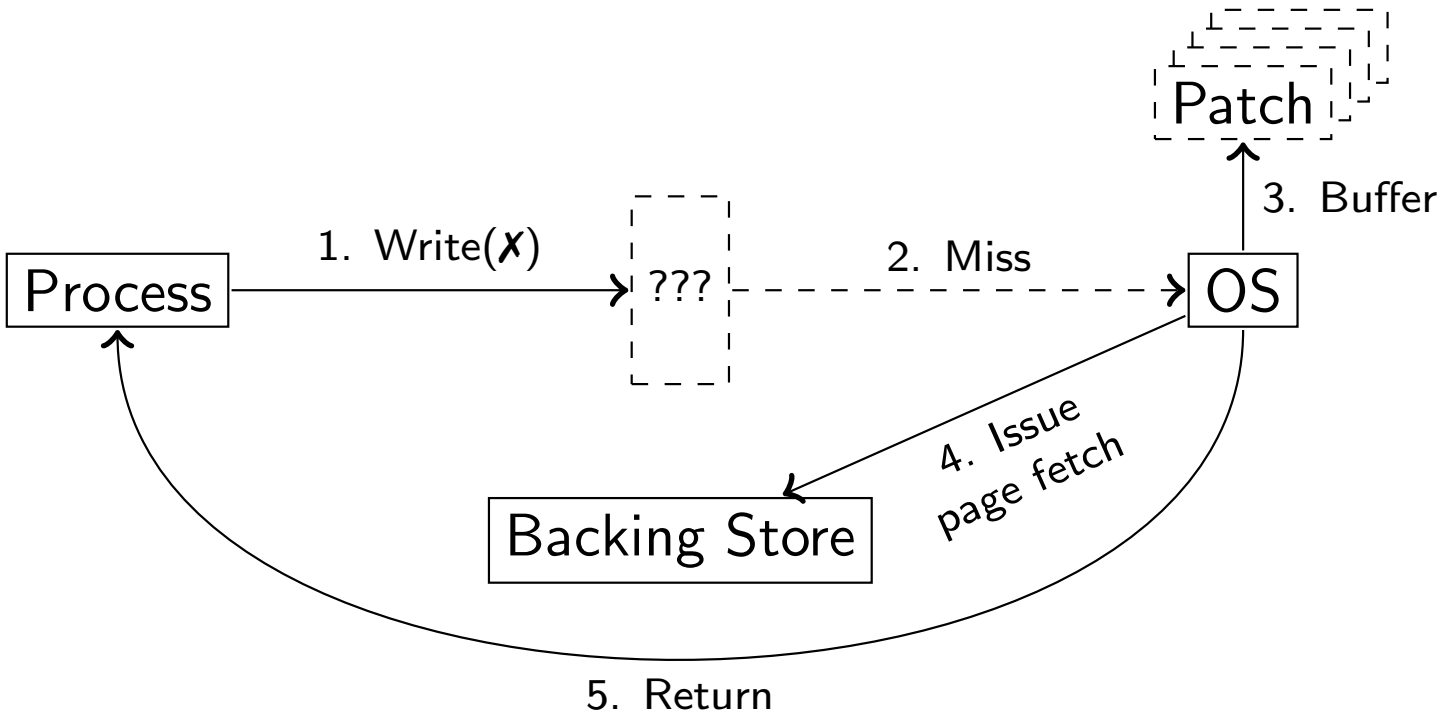
Non-blocking Writes: Basic Approach



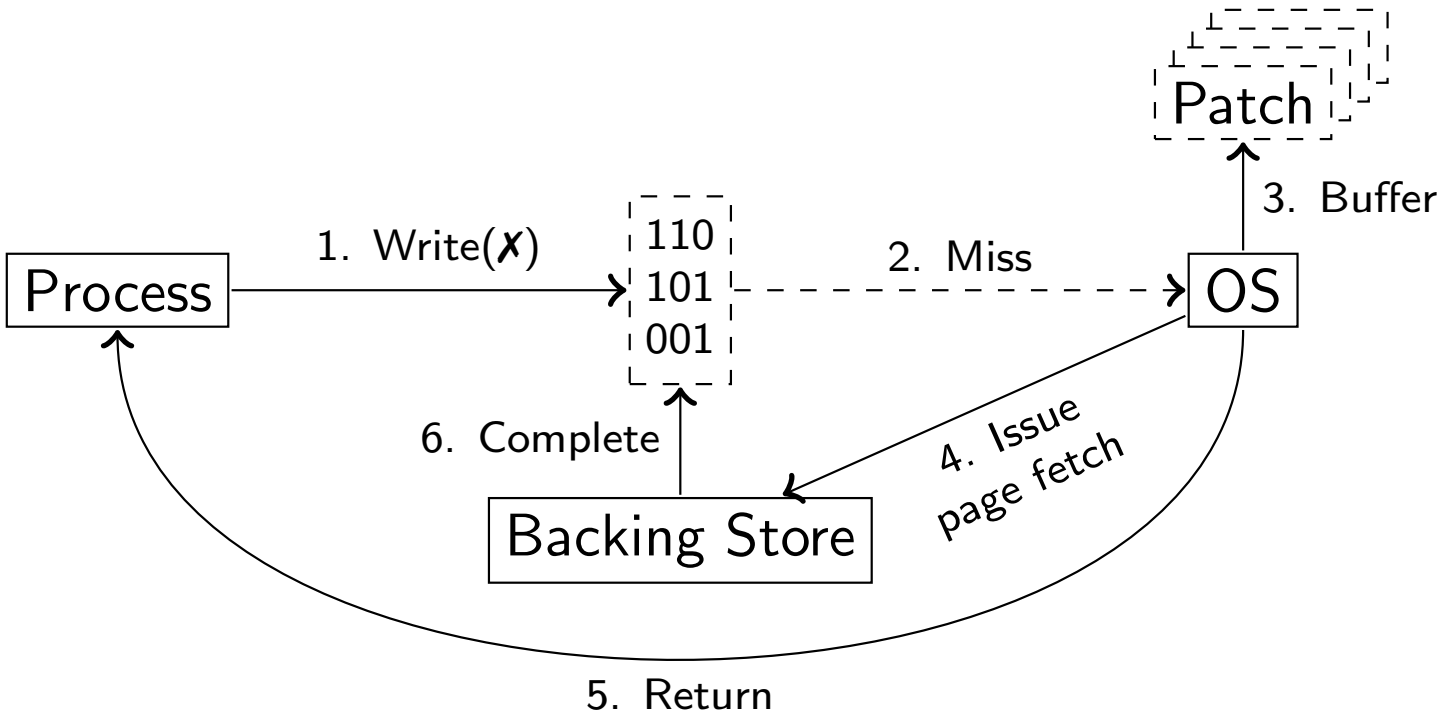
Non-blocking Writes: Basic Approach



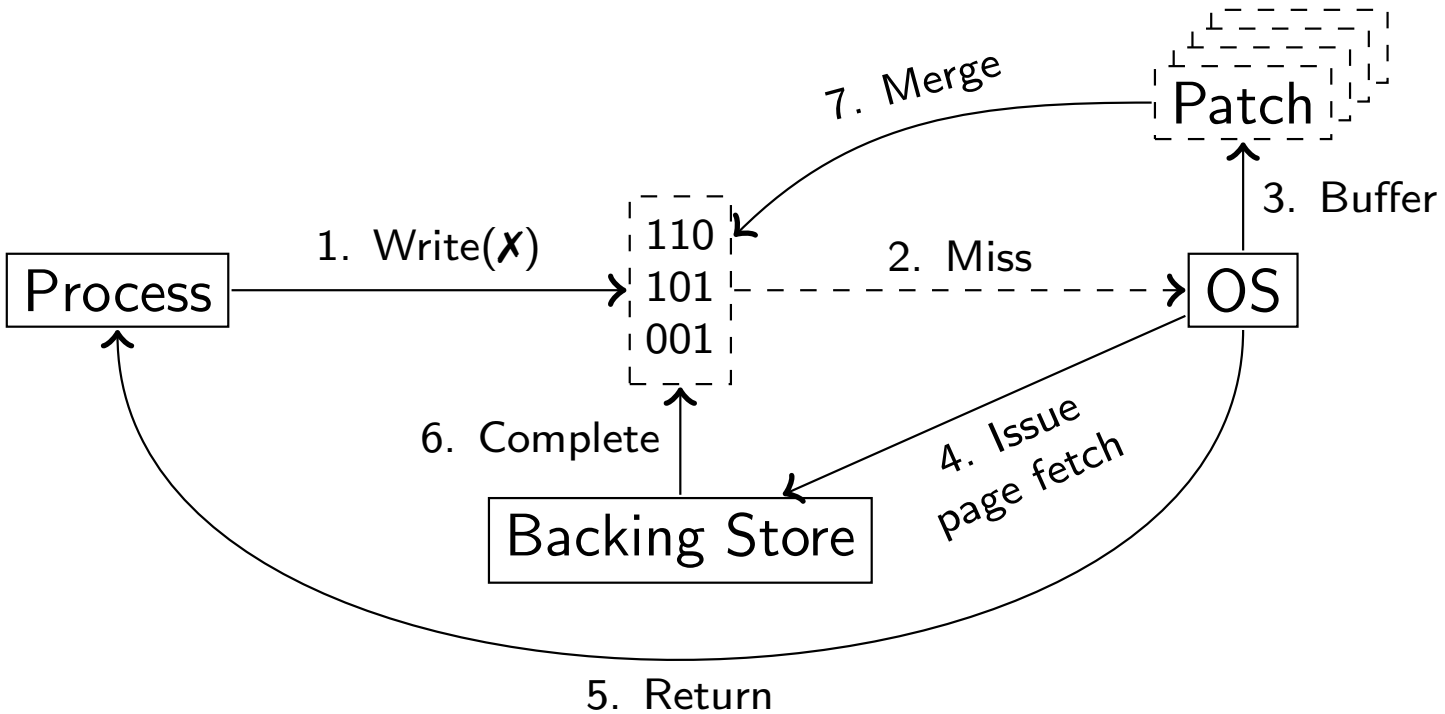
Non-blocking Writes: Basic Approach



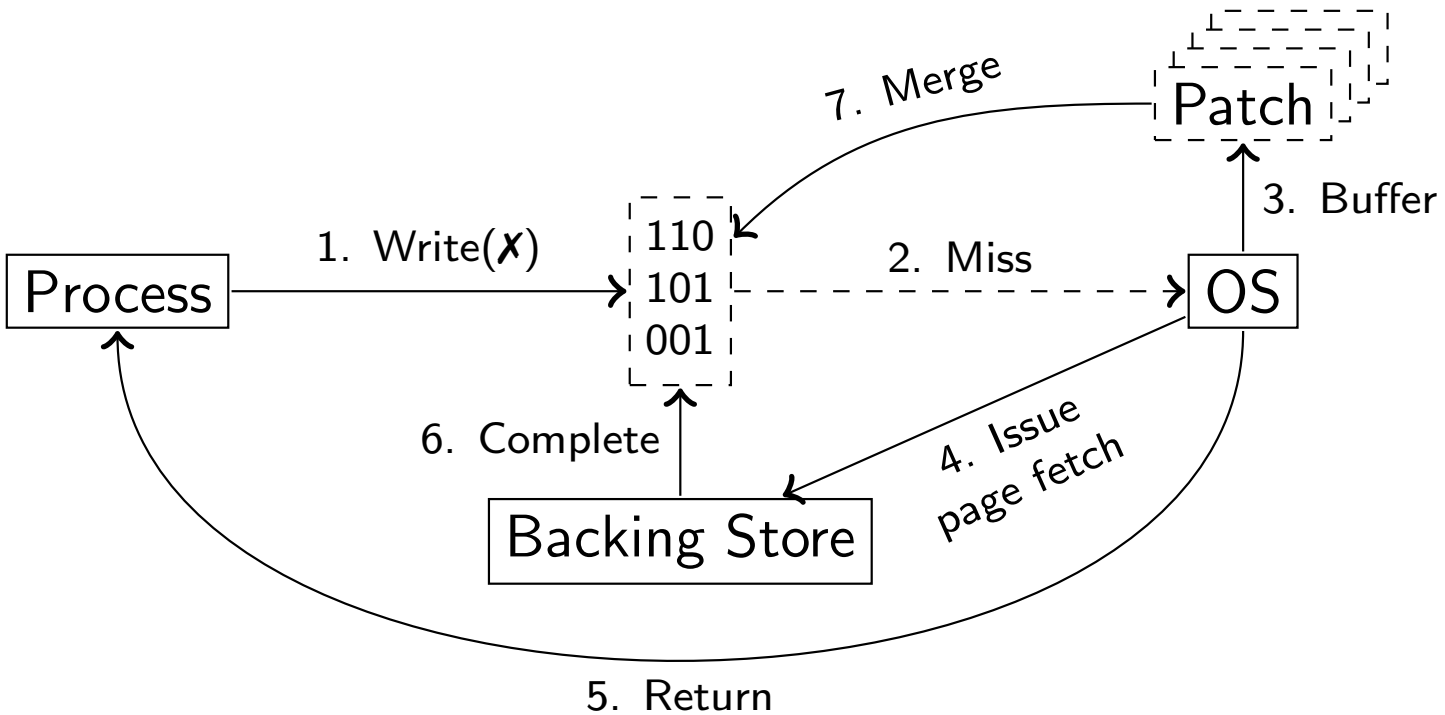
Non-blocking Writes: Basic Approach



Non-blocking Writes: Basic Approach



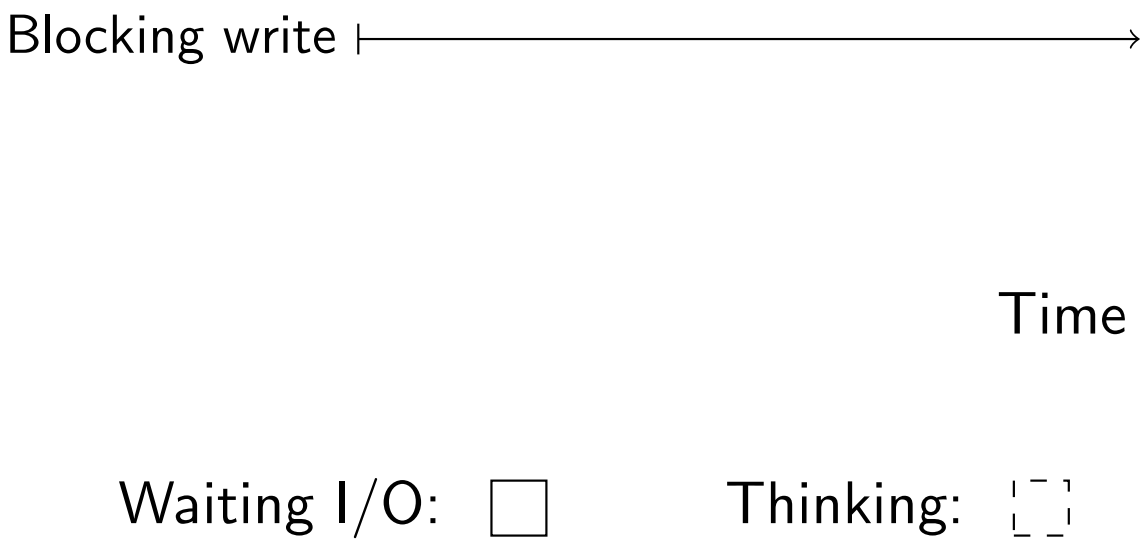
Non-blocking Writes: Basic Approach



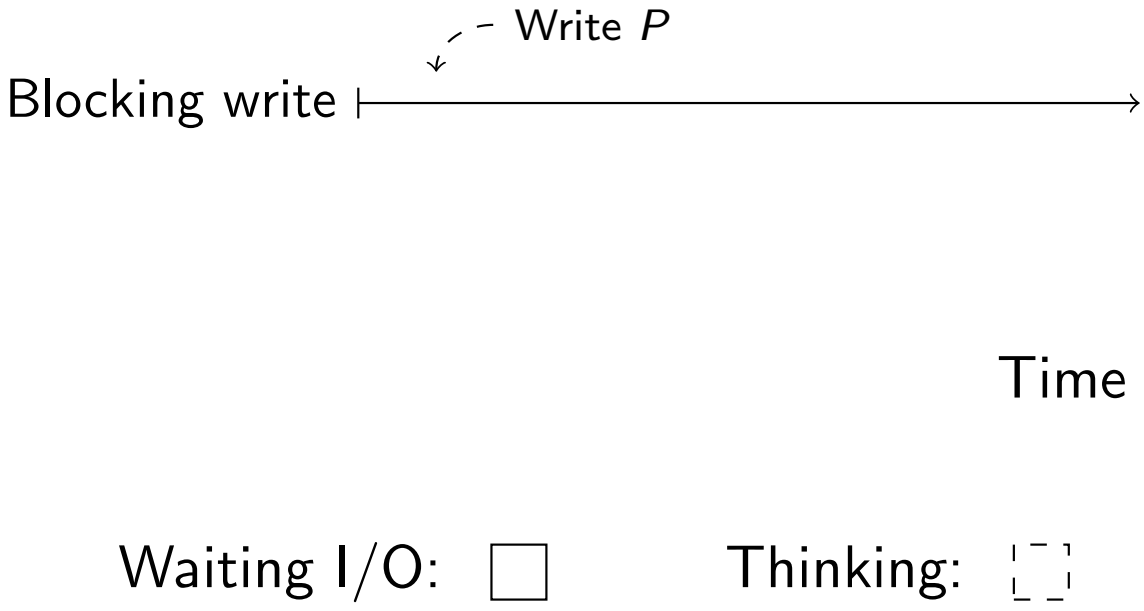
Benefits

1. Reduced access latency to out-of-core pages
2. Increased backing store bandwidth usage

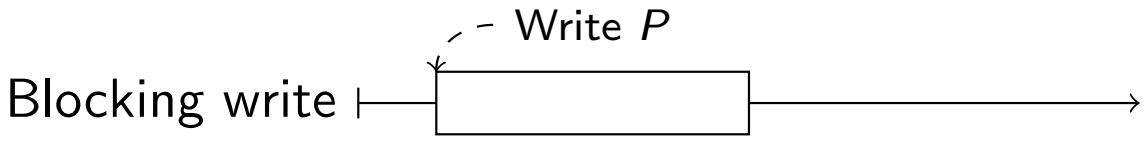
NBW: Page Fetch Asynchrony



NBW: Page Fetch Asynchrony



NBW: Page Fetch Asynchrony

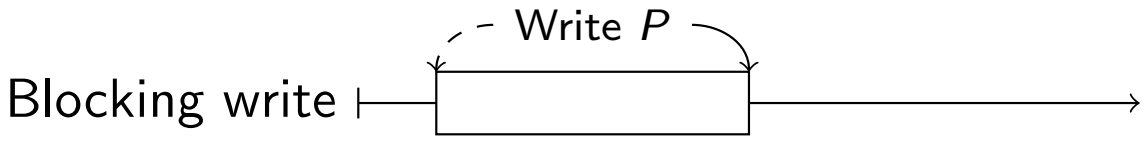


Time

Waiting I/O: 

Thinking: 

NBW: Page Fetch Asynchrony

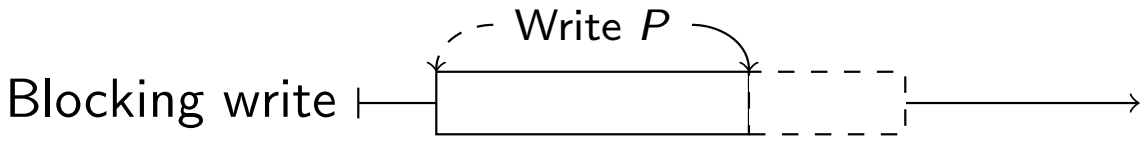


Time

Waiting I/O: 

Thinking: 

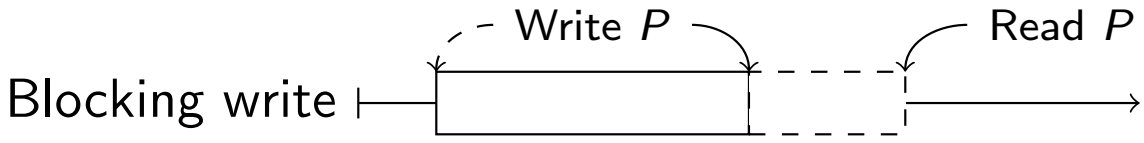
NBW: Page Fetch Asynchrony



Time

Waiting I/O: ☐ Thinking: ⬚

NBW: Page Fetch Asynchrony

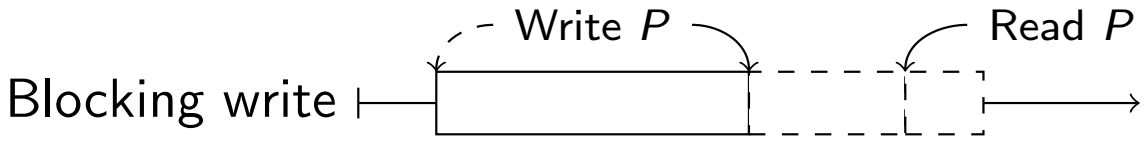


Time

Waiting I/O: 

Thinking: 

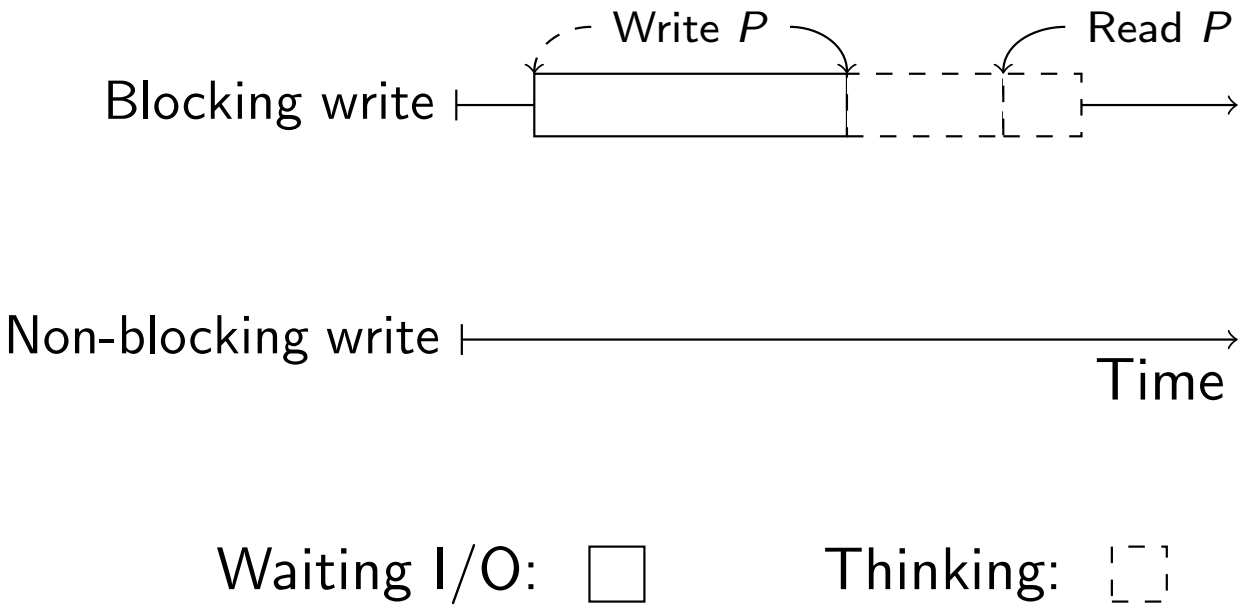
NBW: Page Fetch Asynchrony



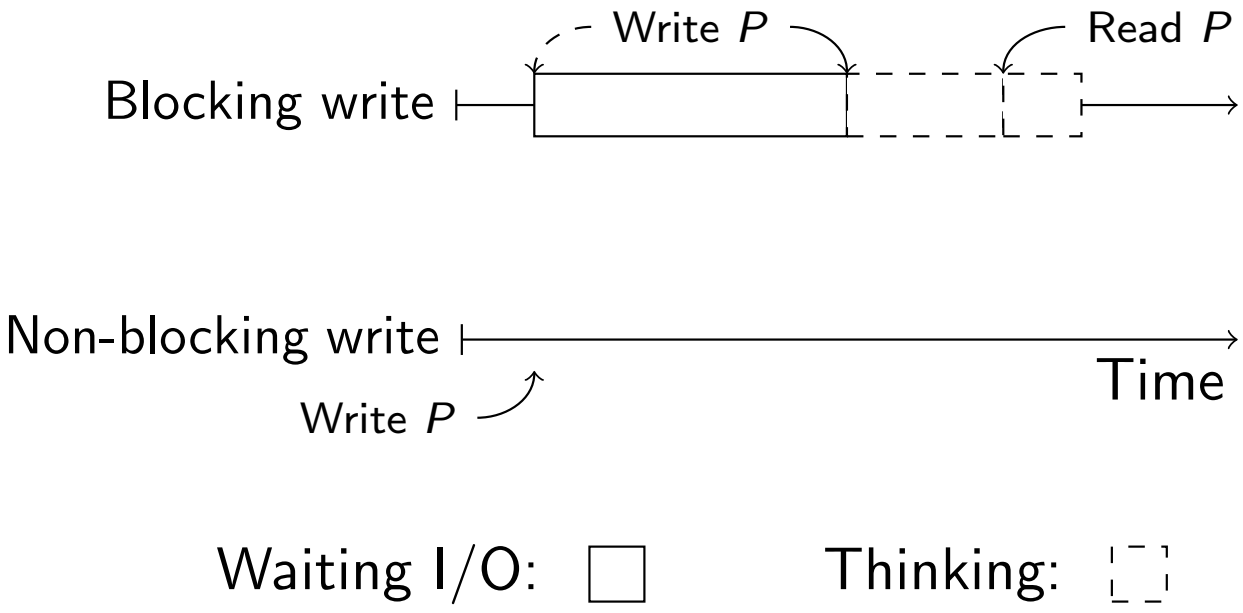
Time

Waiting I/O: ☐ Thinking: ⬚

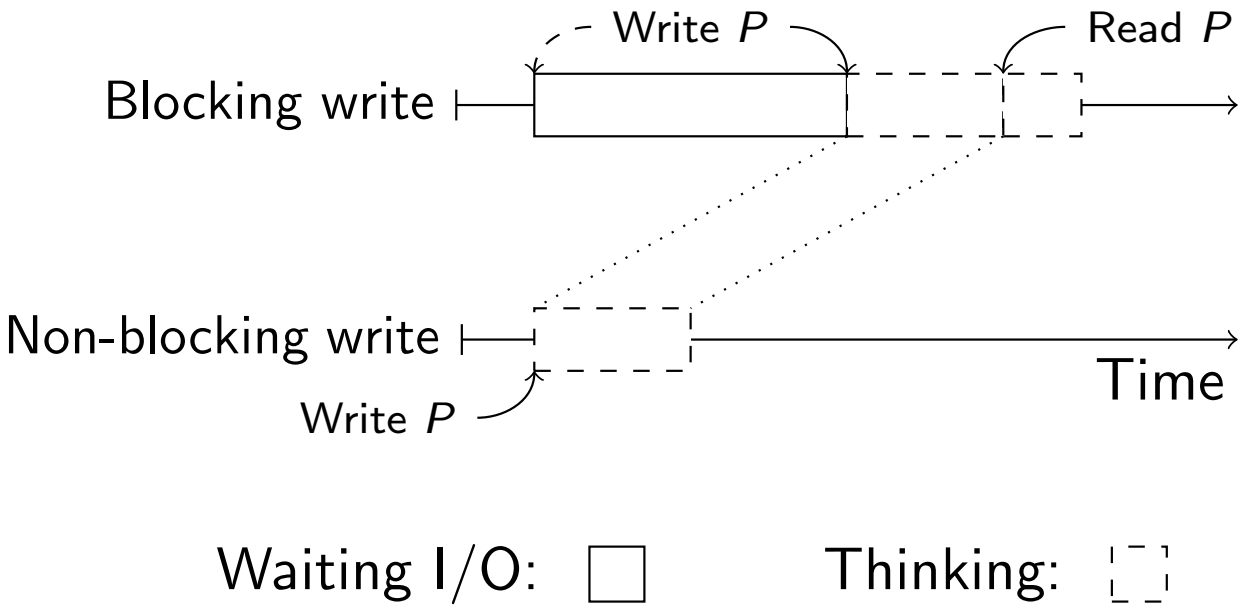
NBW: Page Fetch Asynchrony



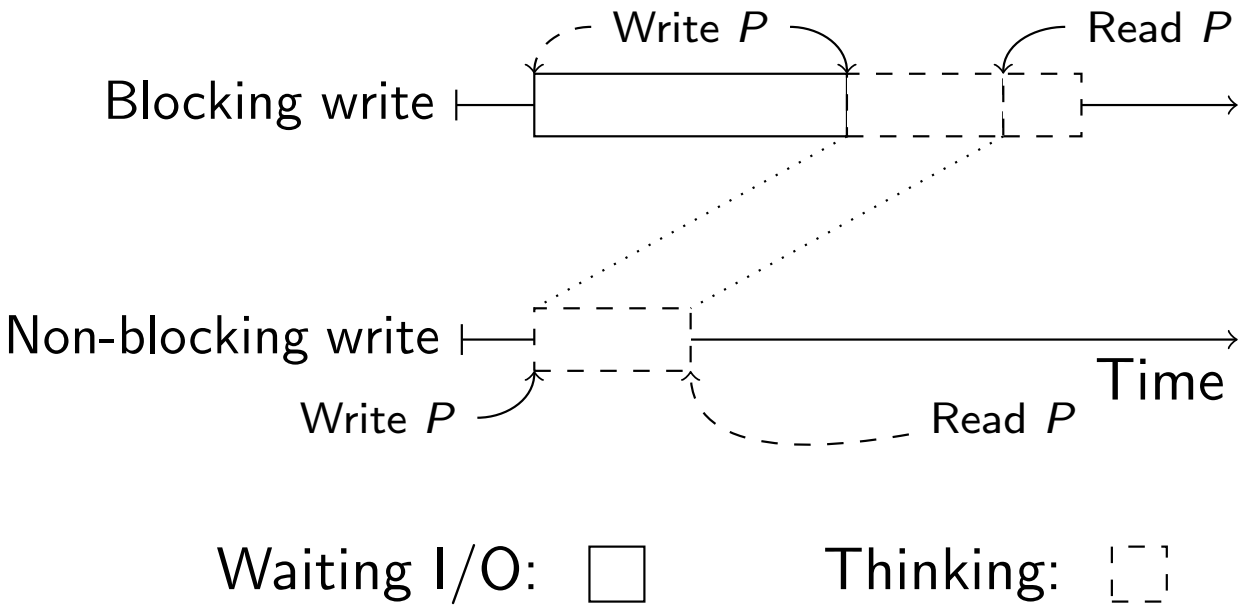
NBW: Page Fetch Asynchrony



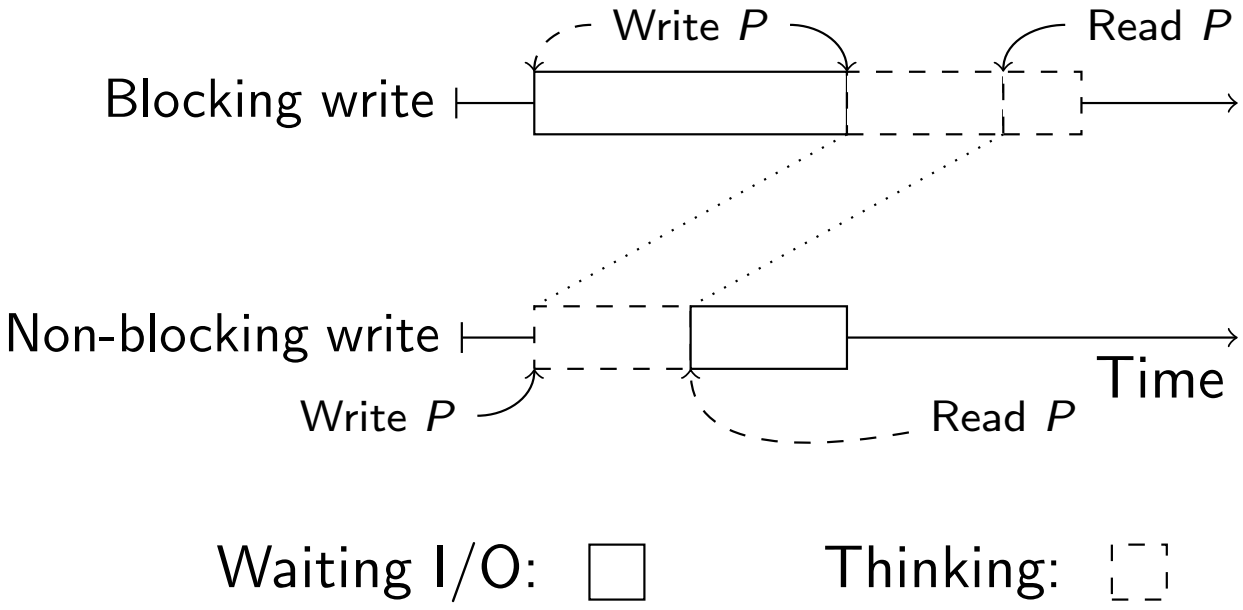
NBW: Page Fetch Asynchrony



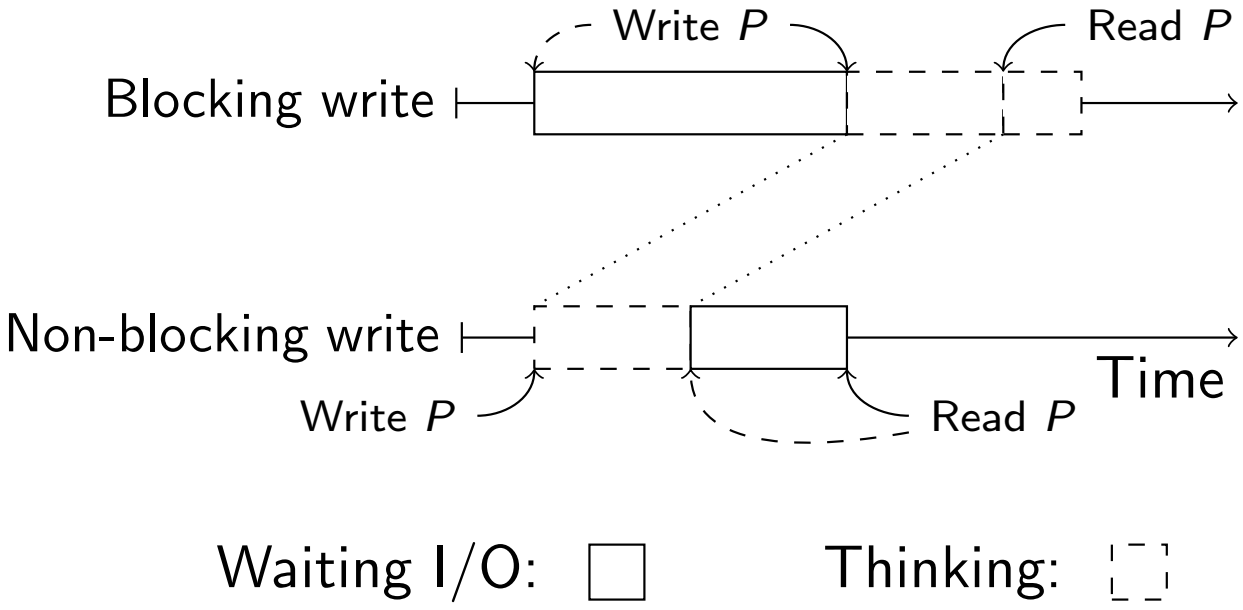
NBW: Page Fetch Asynchrony



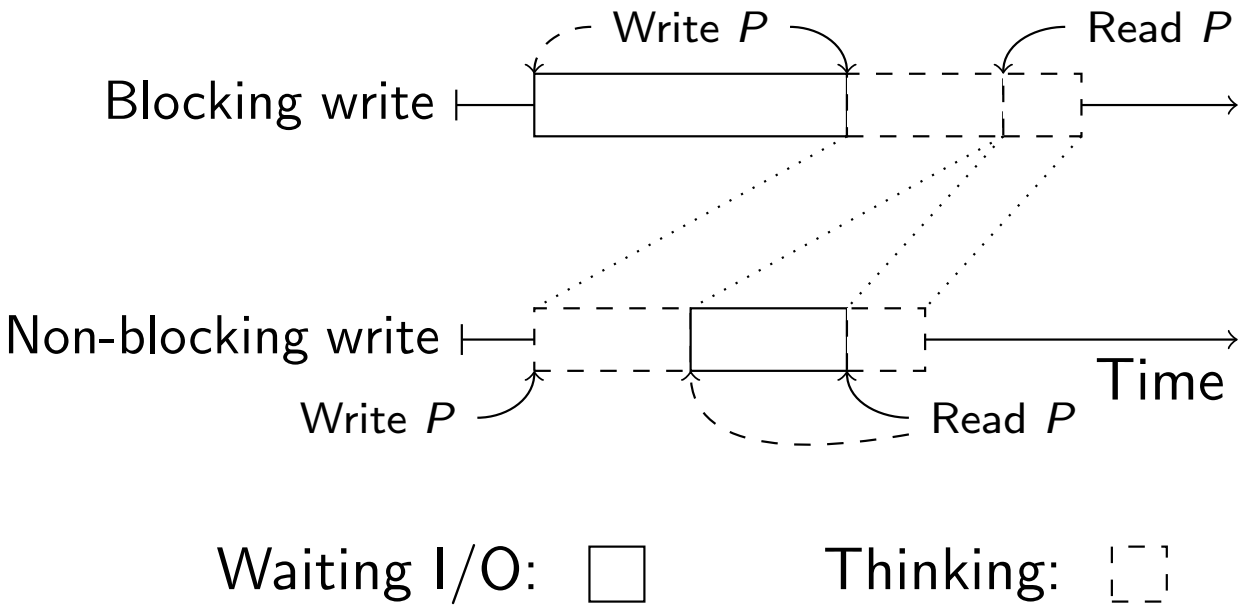
NBW: Page Fetch Asynchrony



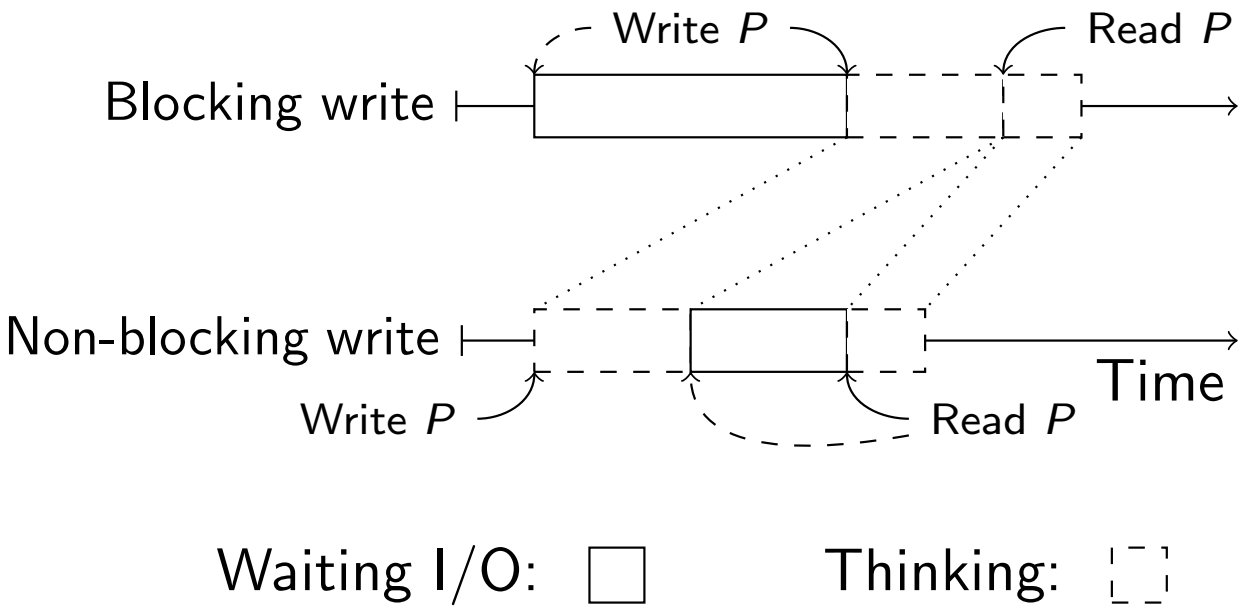
NBW: Page Fetch Asynchrony



NBW: Page Fetch Asynchrony

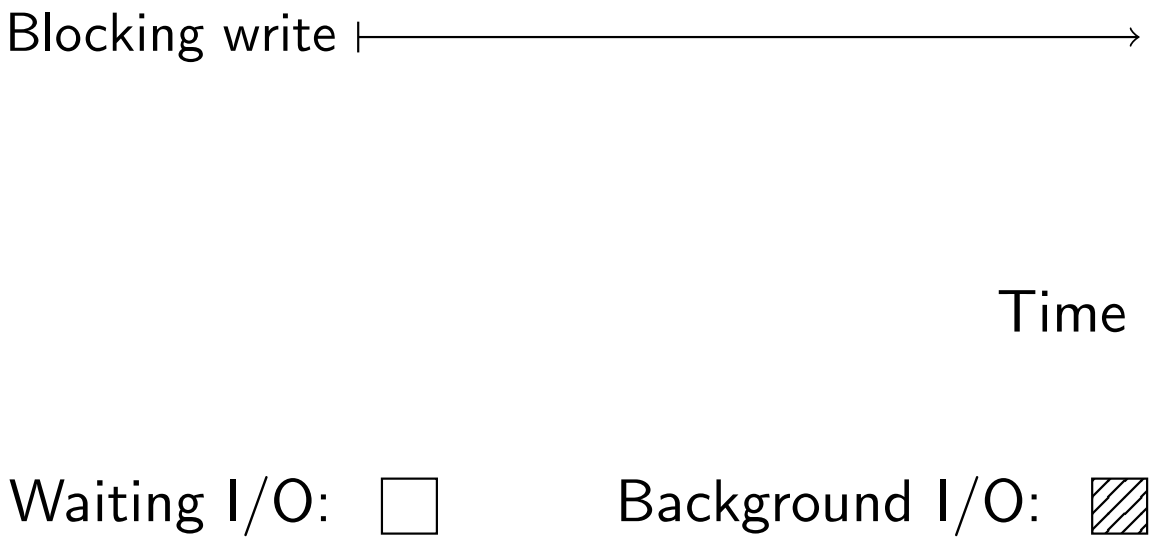


NBW: Page Fetch Asynchrony

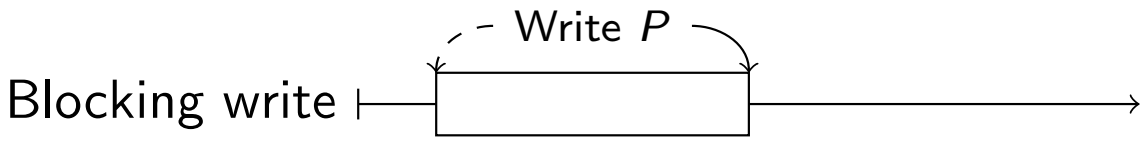


- Immediate return from write
- Overlaping of I/O and computation

NBW: Page Fetch Parallelism



NBW: Page Fetch Parallelism

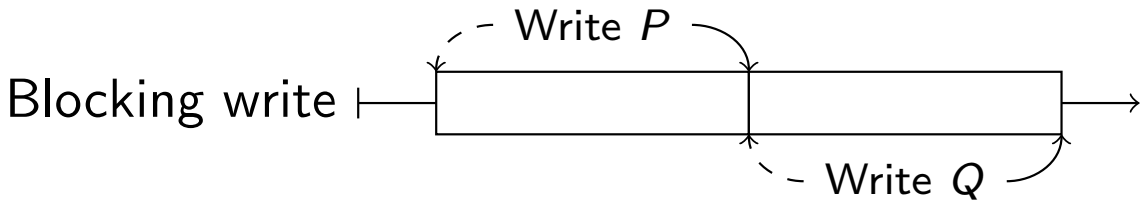


Time

Waiting I/O: 

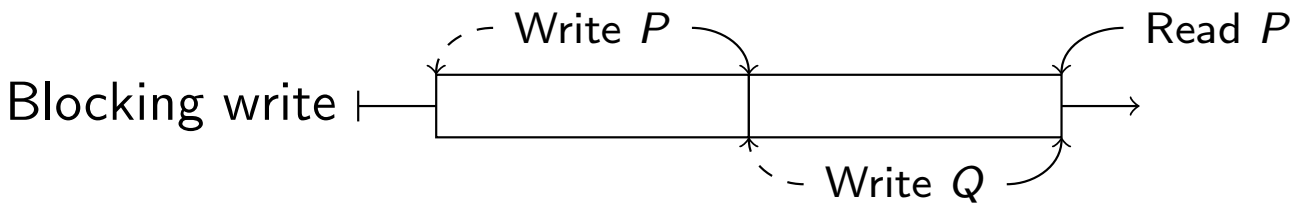
Background I/O: 

NBW: Page Fetch Parallelism



Waiting I/O: ☐ Background I/O: ▨

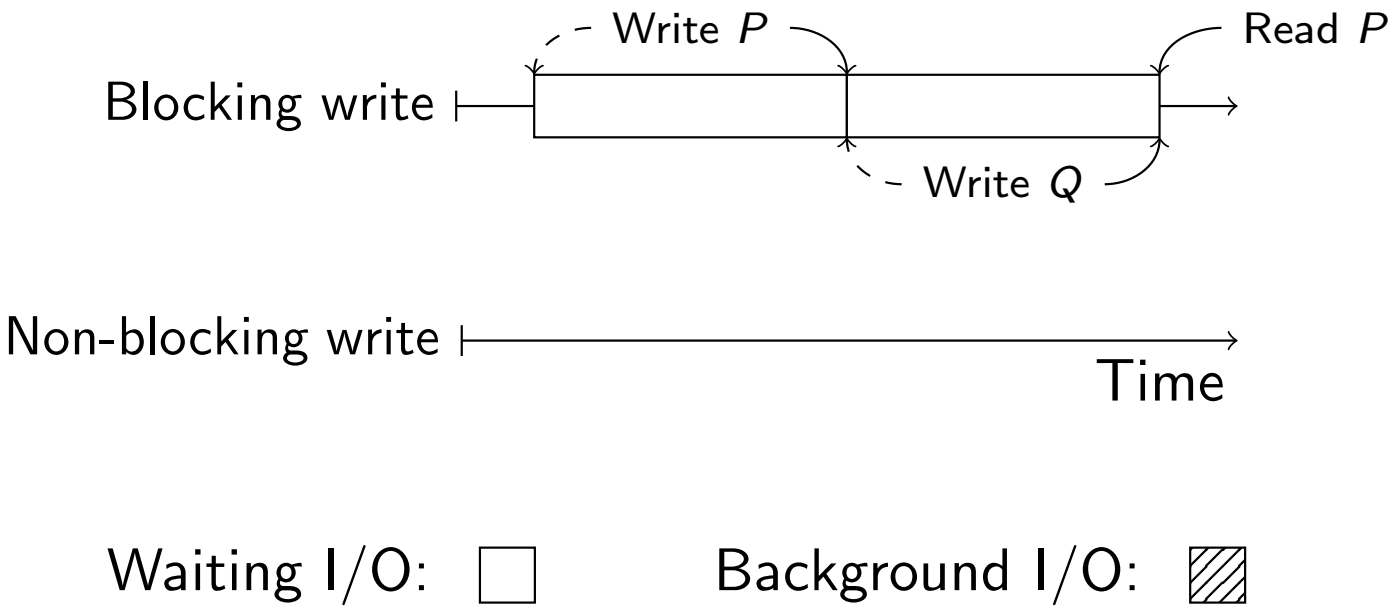
NBW: Page Fetch Parallelism



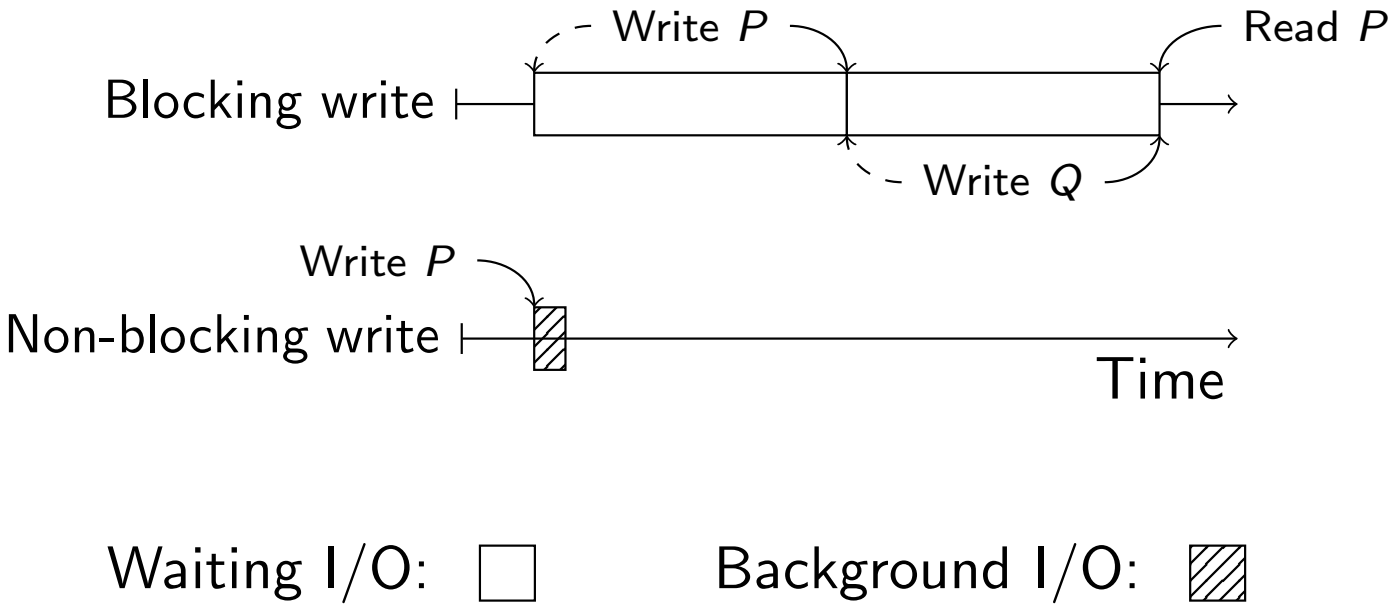
Time

Waiting I/O: ☐ Background I/O: ▨

NBW: Page Fetch Parallelism

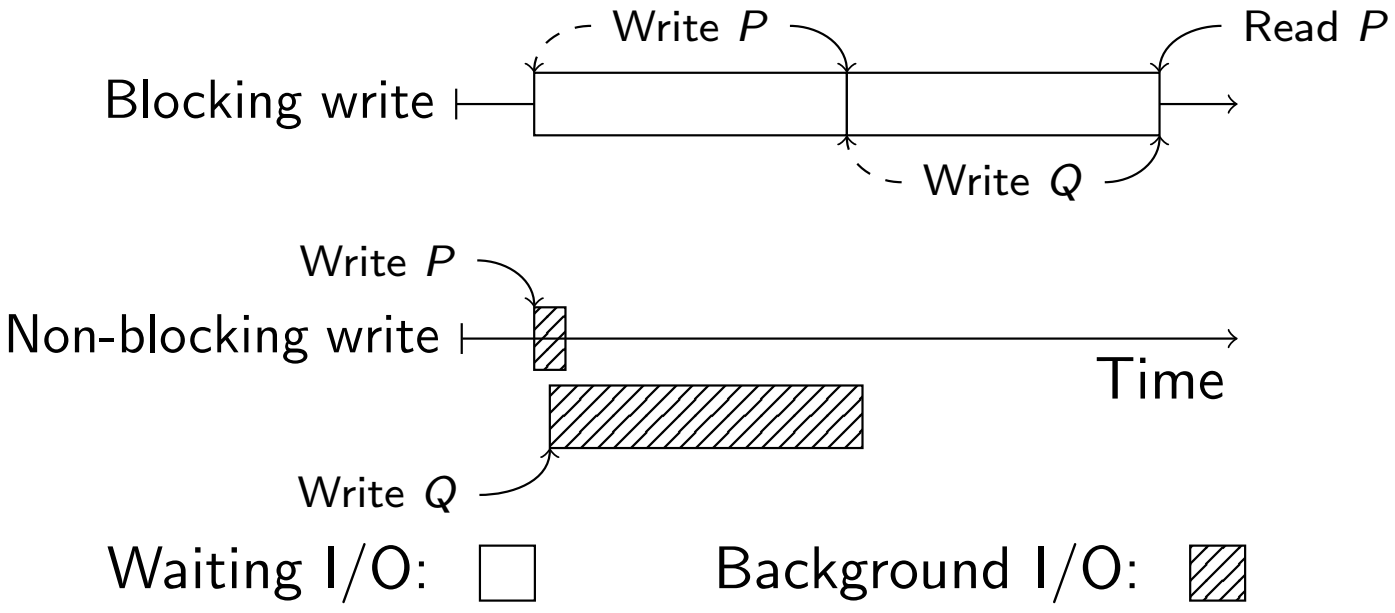


NBW: Page Fetch Parallelism

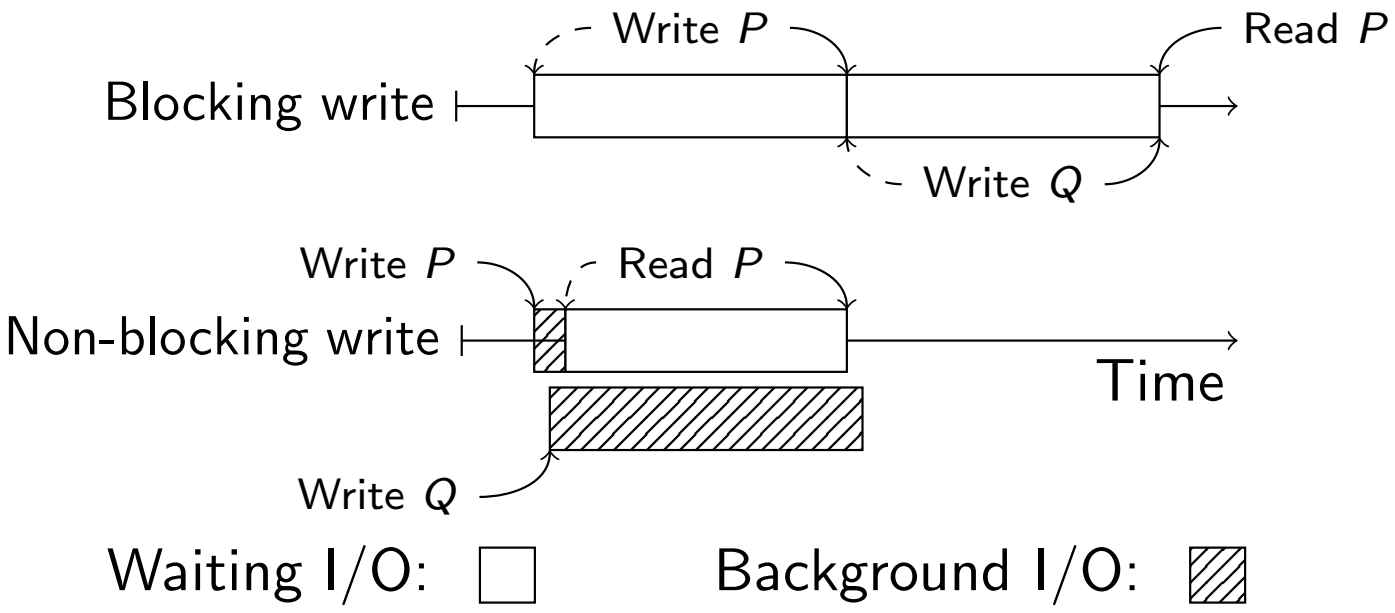


Waiting I/O: ☐ Background I/O: ▨

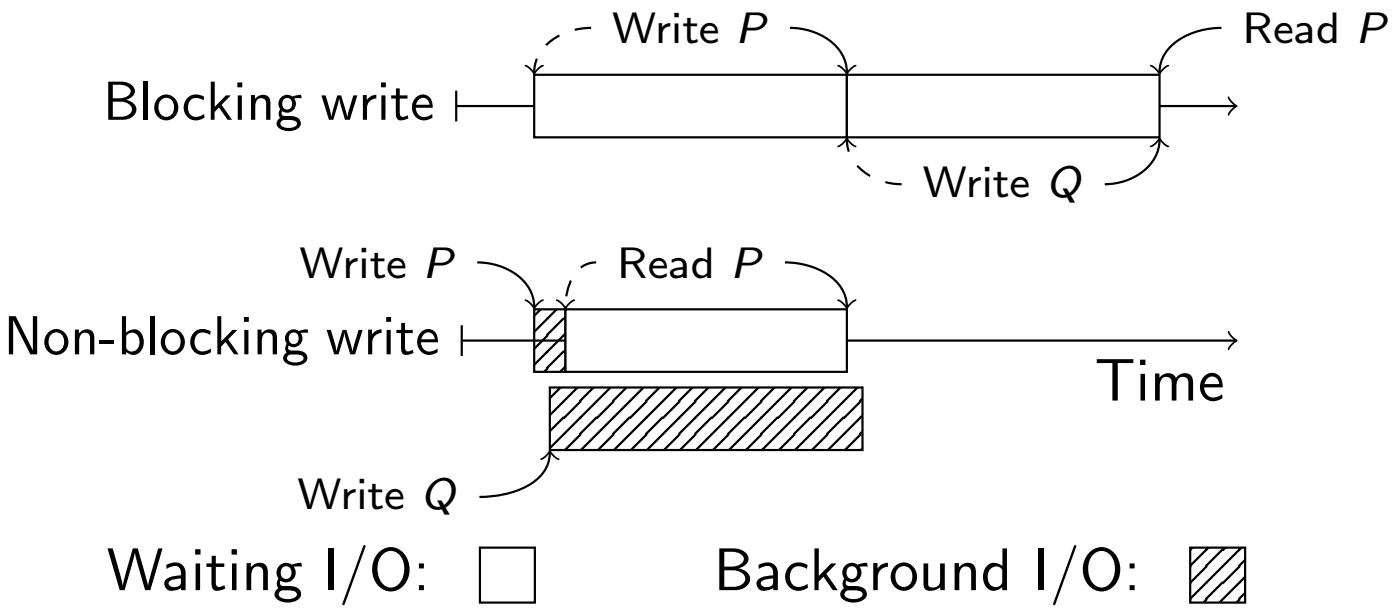
NBW: Page Fetch Parallelism



NBW: Page Fetch Parallelism



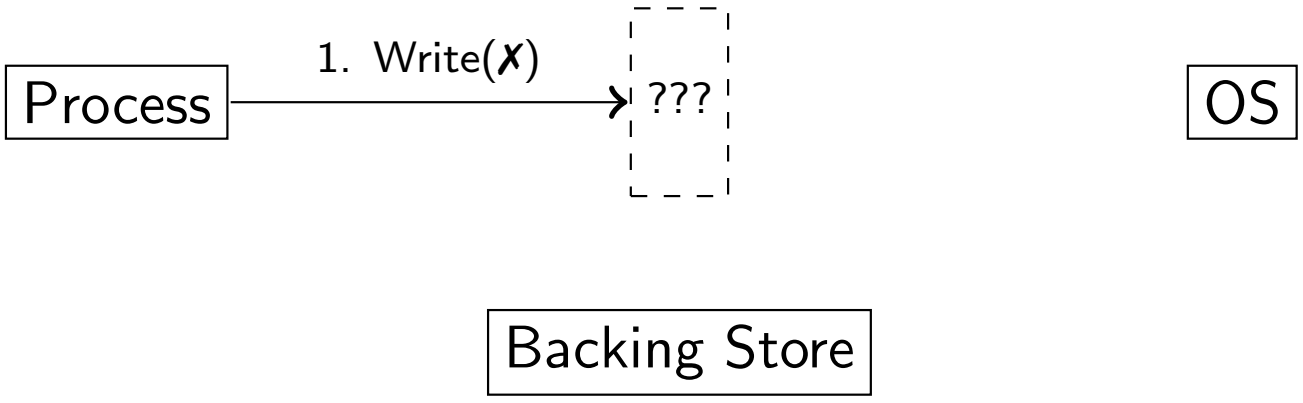
NBW: Page Fetch Parallelism



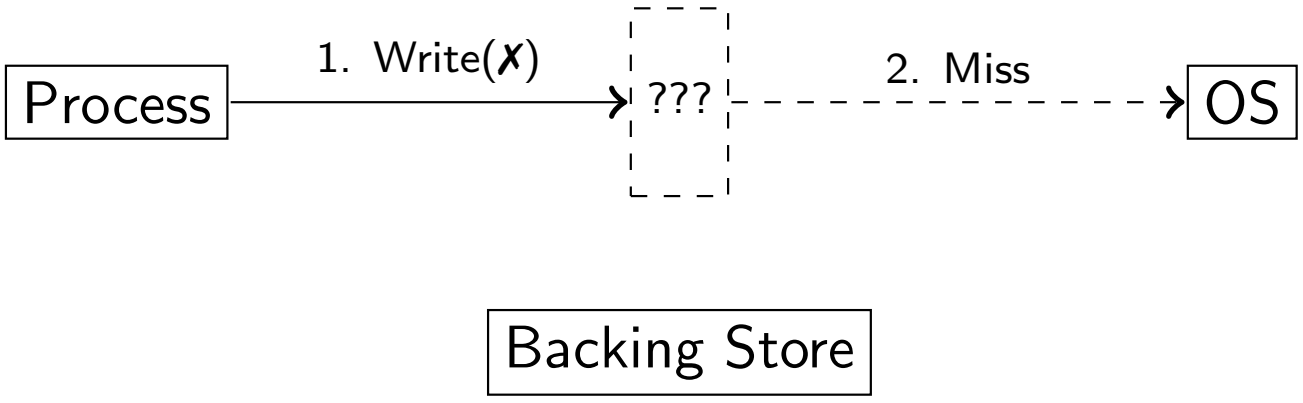
- Increases I/O parallelism
- Increases I/O scheduler efficiency

Delaying Page Fetches

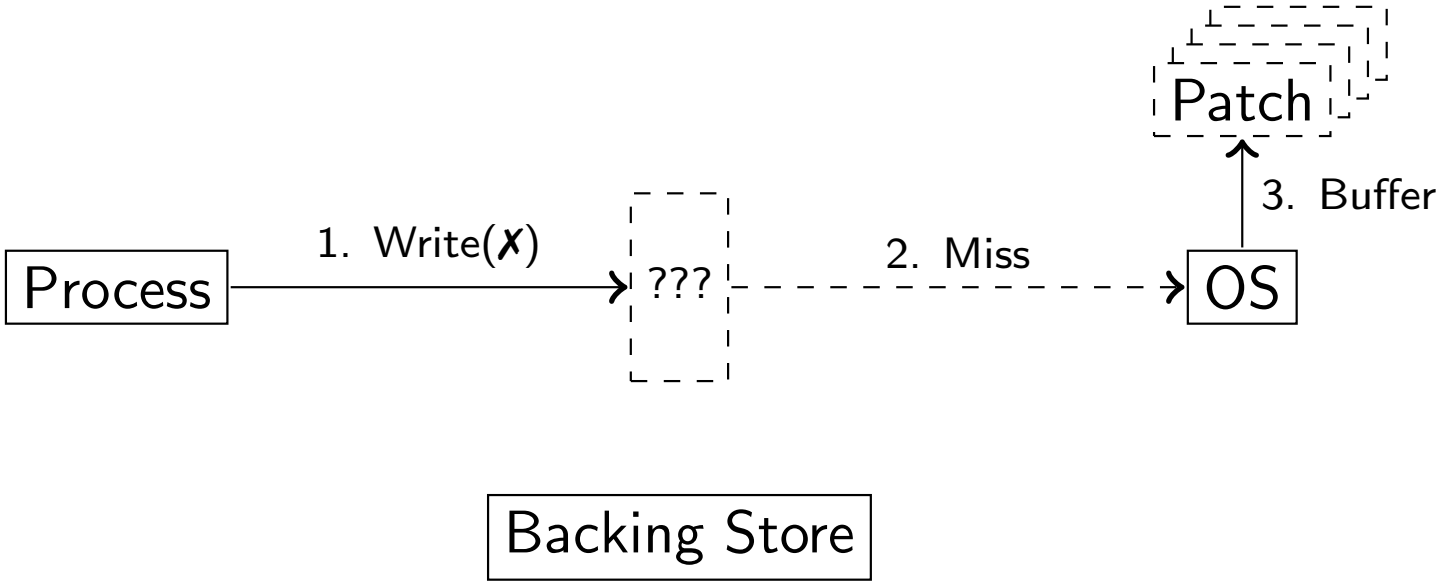
Delaying Page Fetches



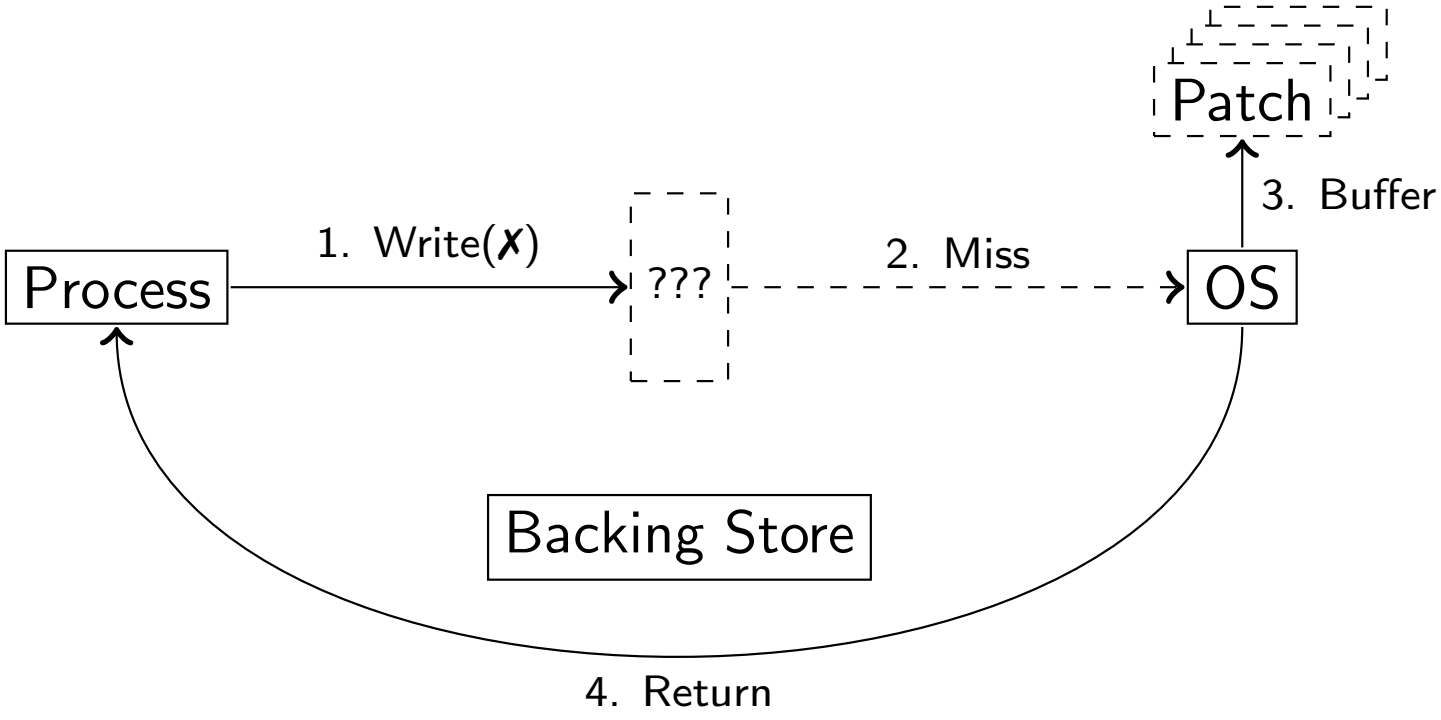
Delaying Page Fetches



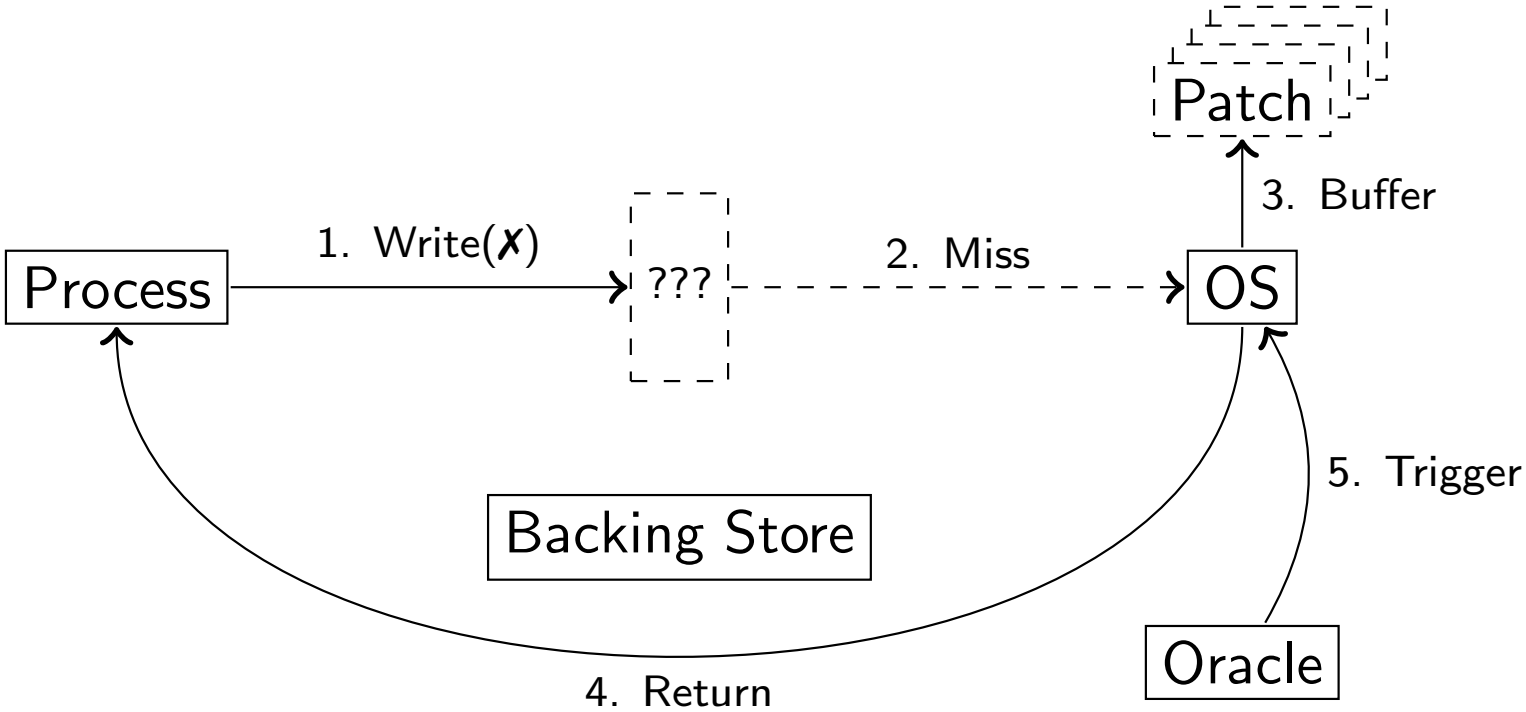
Delaying Page Fetches



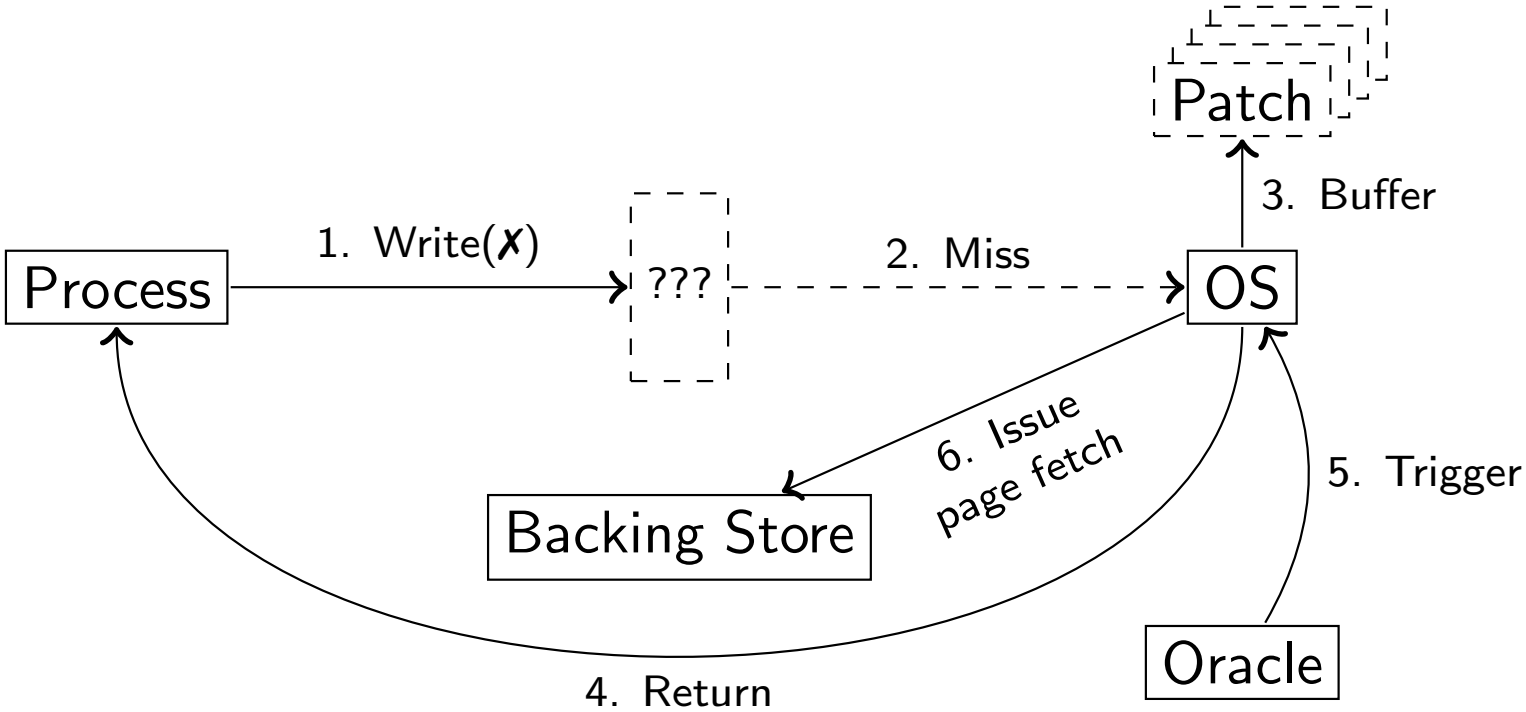
Delaying Page Fetches



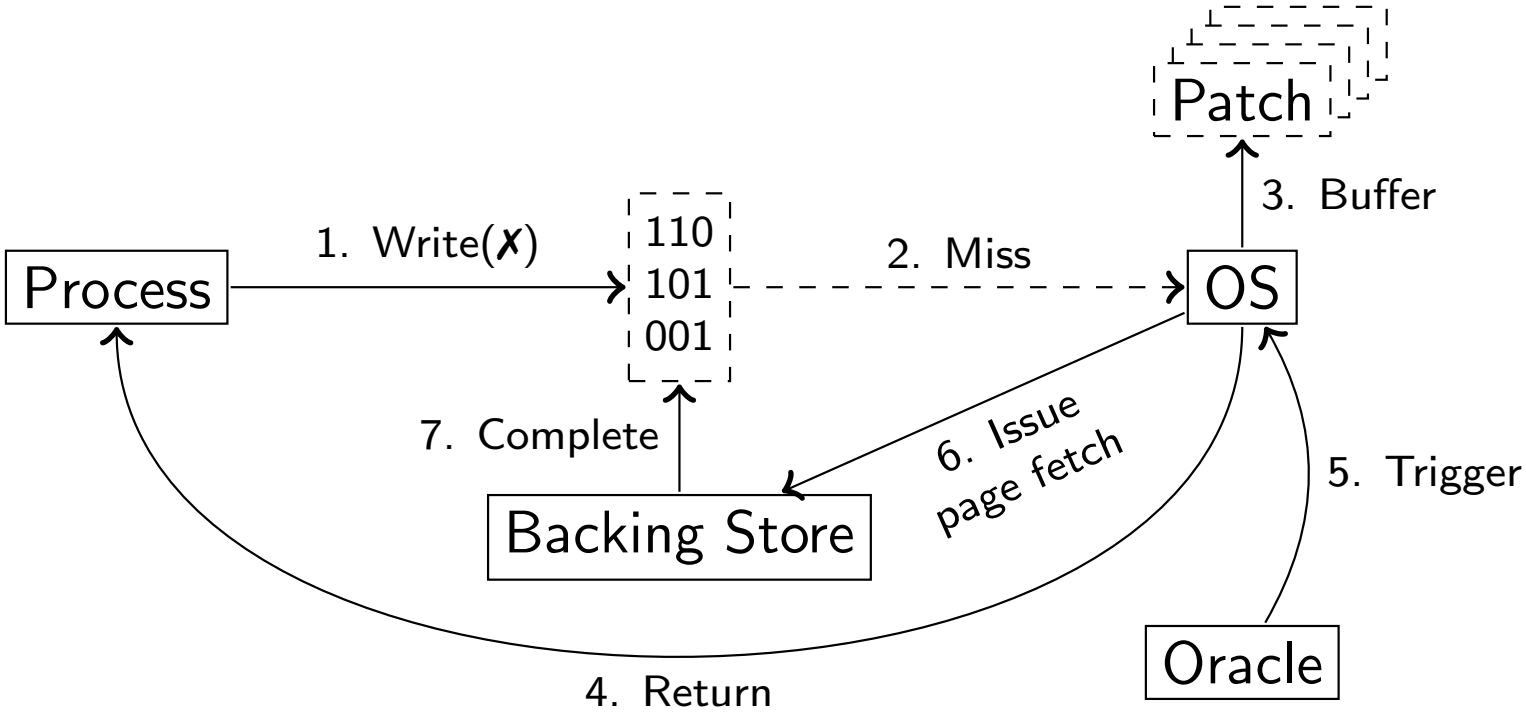
Delaying Page Fetches



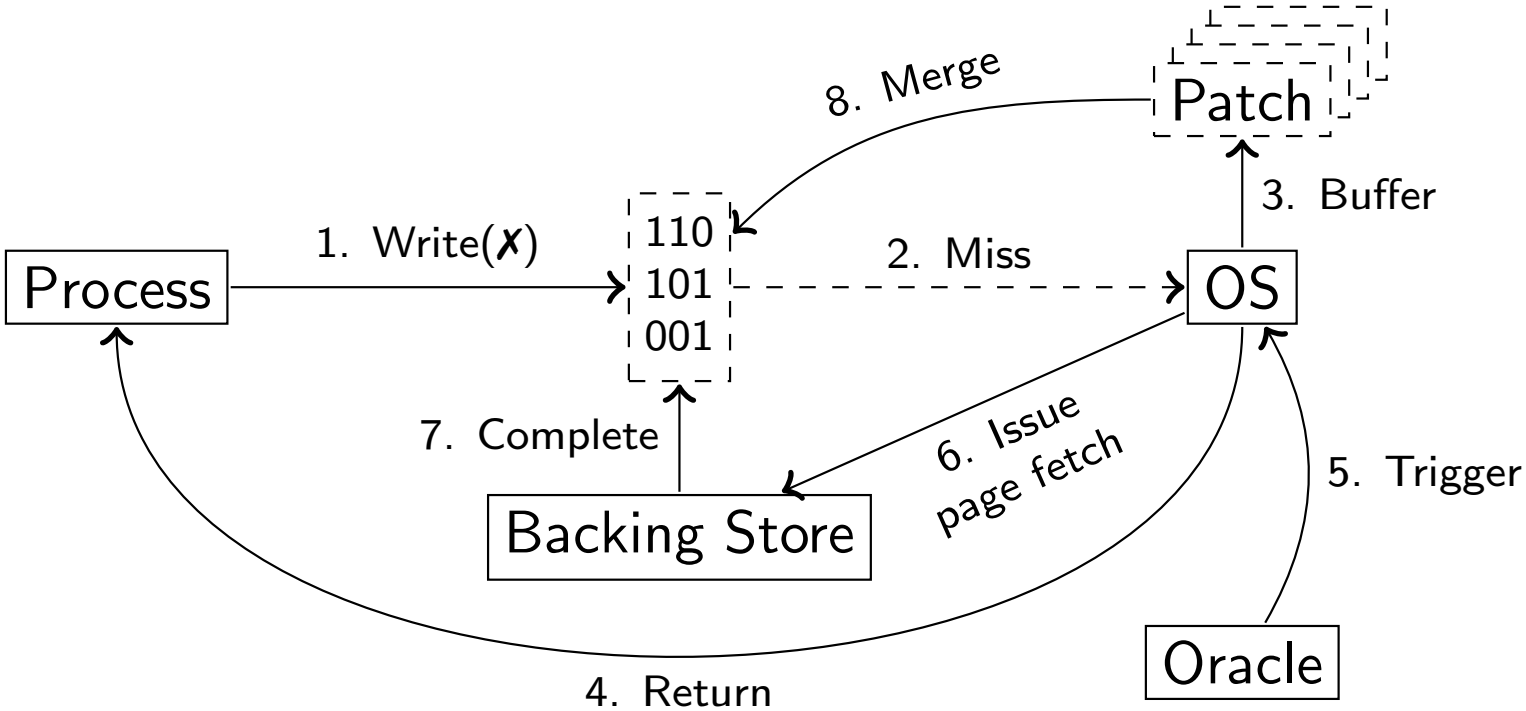
Delaying Page Fetches



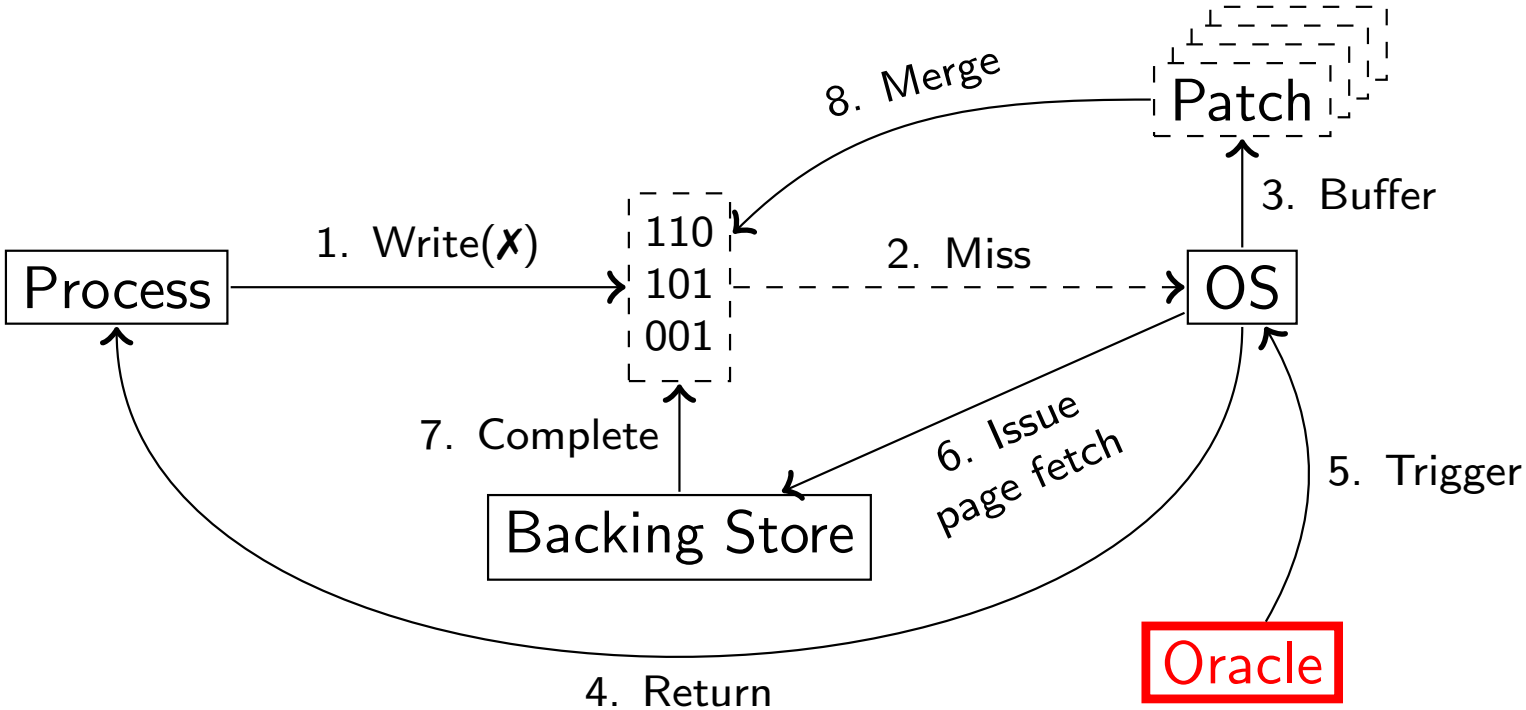
Delaying Page Fetches



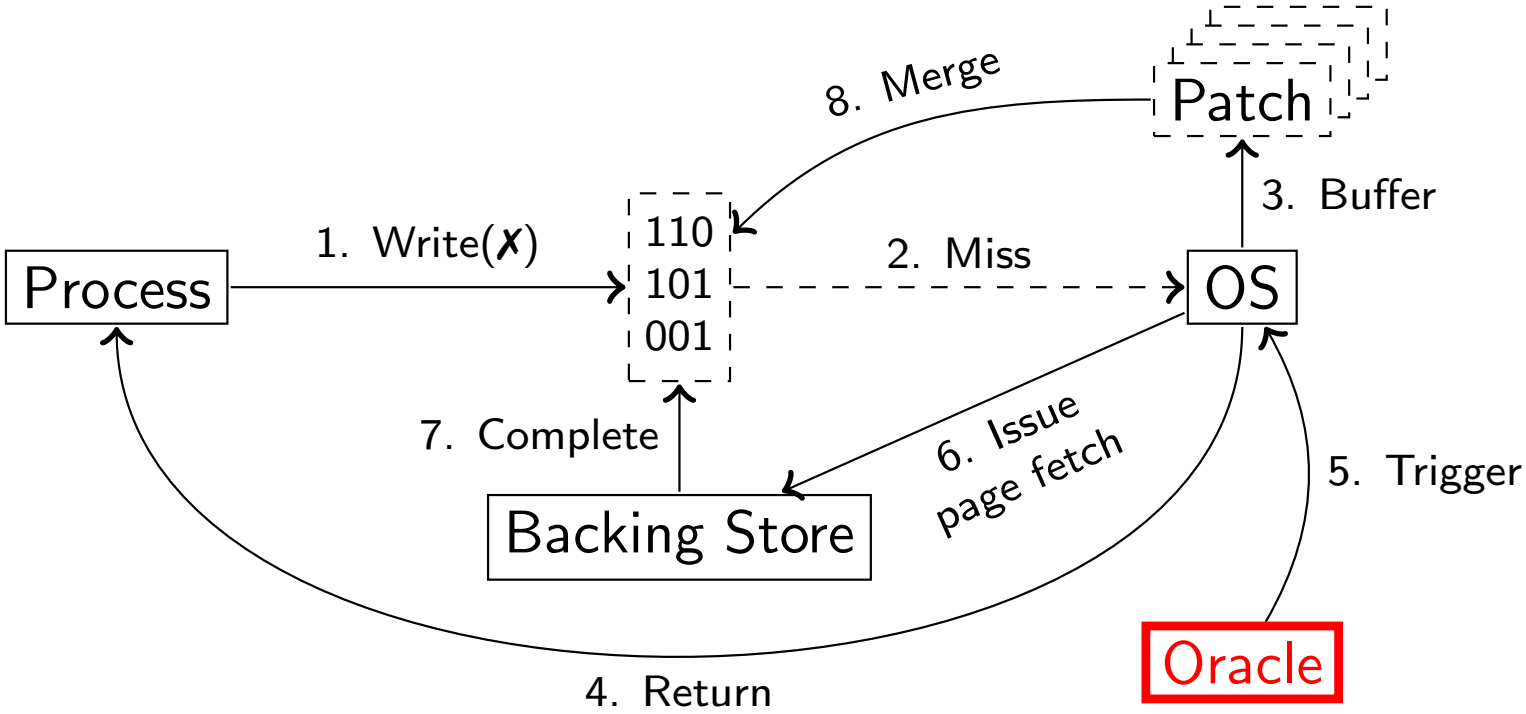
Delaying Page Fetches



Delaying Page Fetches



Delaying Page Fetches



Oracle Uses

1. Save power by delaying fetches until disks spin-up (EXCES & SRCMap)
2. Help systems under resource pressure: Disk or Memory
3. Reduce memory usage by delaying page allocation

Patch Optimizations

Read

We can read data from patches

- Making reads to out-of-core pages also asynchronous
- Delay fetch further if required

Merge

Merge a new patch with an old one and build a new bigger patch

- Reduce meta-data

Re-use

If re-writing a patch, re-use the old one

- Reduce meta-data
- Reduce patch queue size

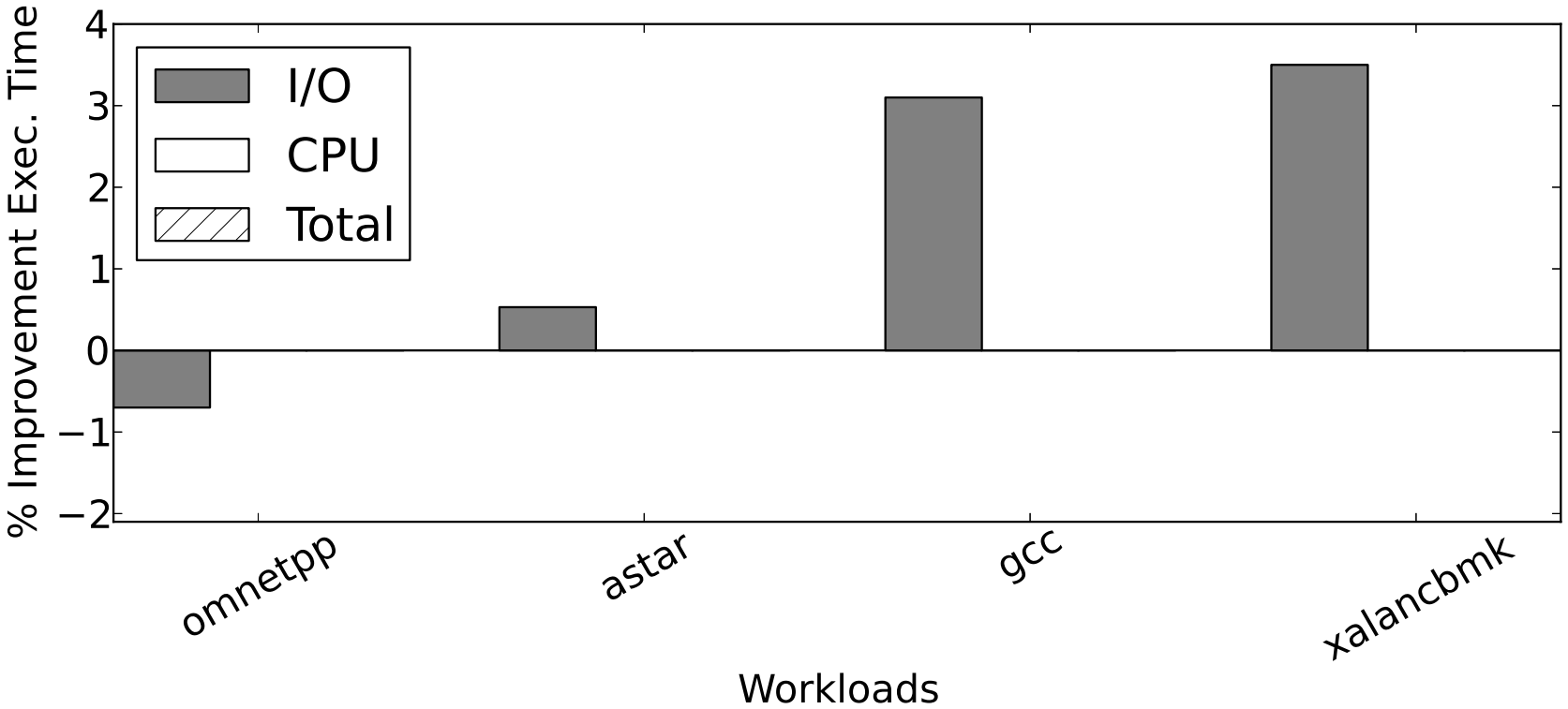
NBW Evaluation

- We implemented NBW for Linux for x86
- All patch optimizations implemented
- We only evaluate *asynchronous* mode for performance
- We use 10 workloads:
  - ✓ 4 SPEC CPU2006 (single-threaded)
  - ✓ 4 DaCapo (multi-threaded)
  - ✓ SPEC SFS file server
  - ✓ SPEC Power application server

NBW Evaluation

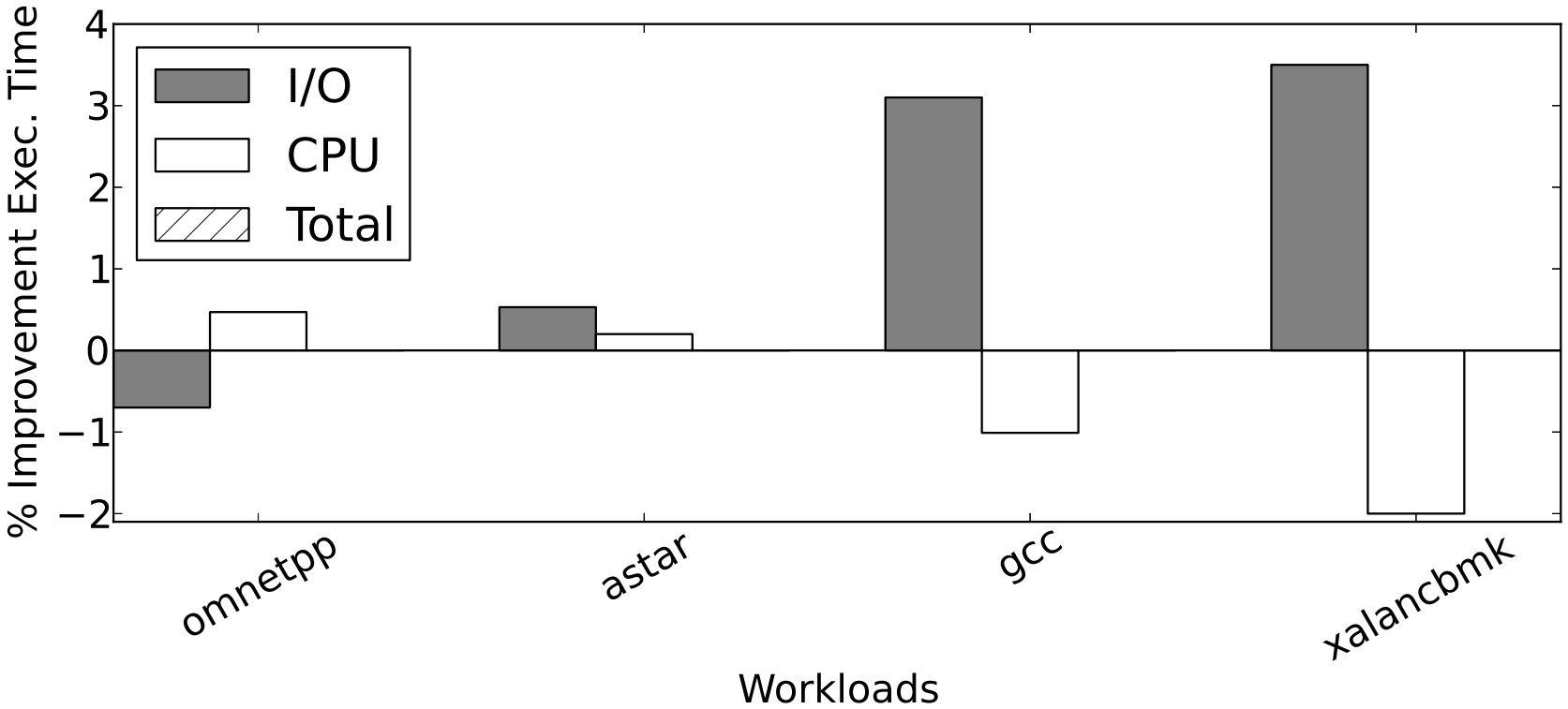
- We implemented NBW for Linux for x86
- All patch optimizations implemented
- We only evaluate *asynchronous* mode for performance
- We use 10 workloads:
  - ✓ 4 SPEC CPU2006 (single-threaded)
  - ✓ 4 DaCapo (multi-threaded)
  - ✓ SPEC SFS file server
  - ✓ SPEC Power application server

NBW Evaluation: Single Thread



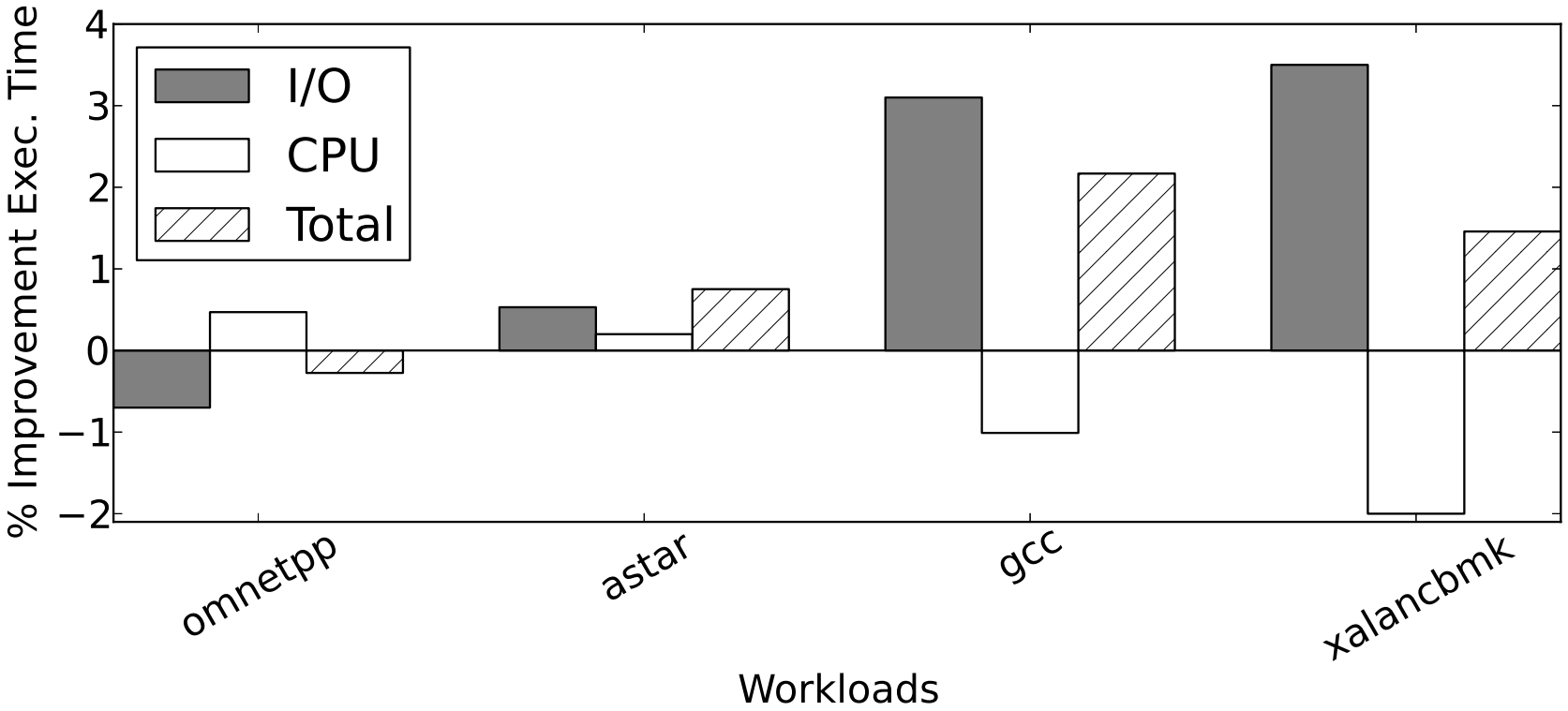
- Average of 5 runs (<2% COV)

NBW Evaluation: Single Thread



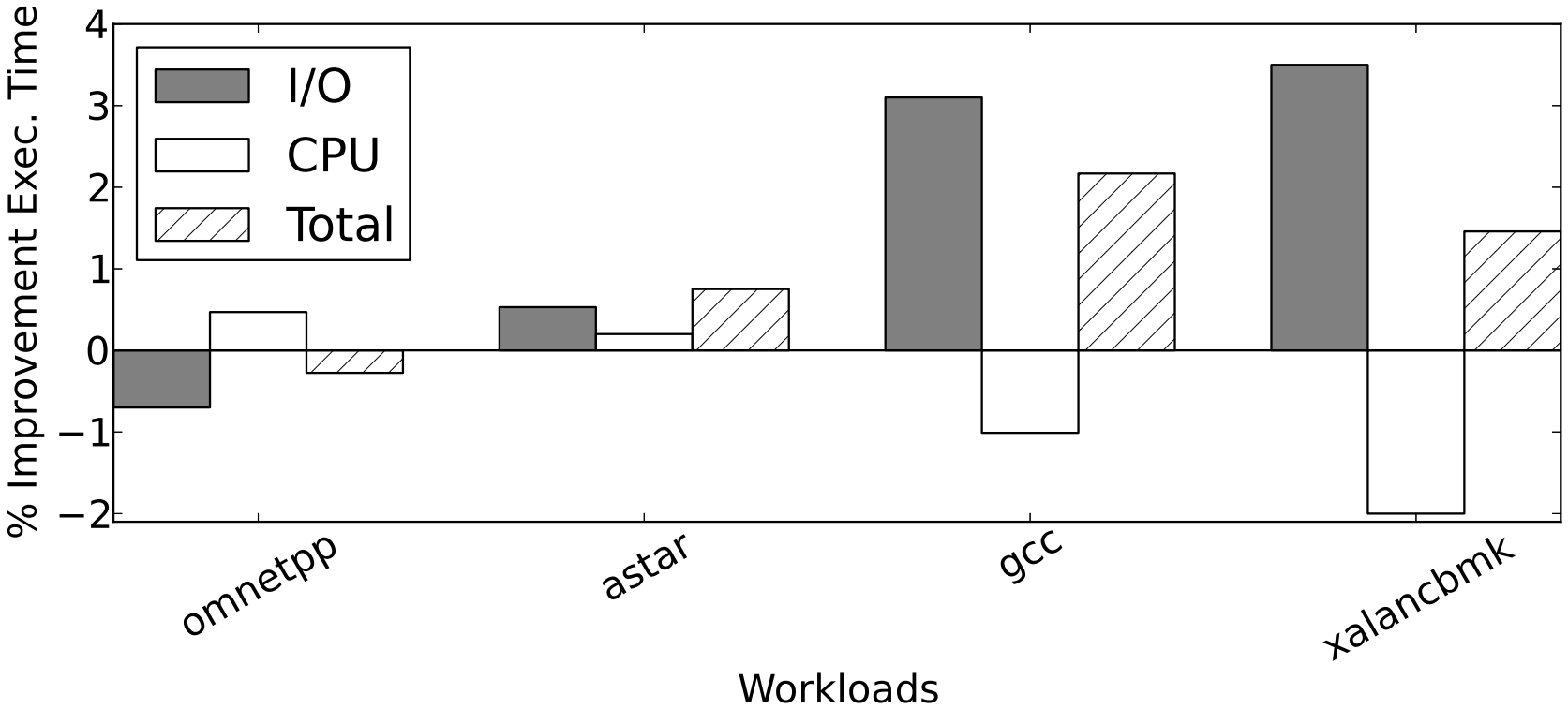
- Average of 5 runs (<2% COV)

NBW Evaluation: Single Thread



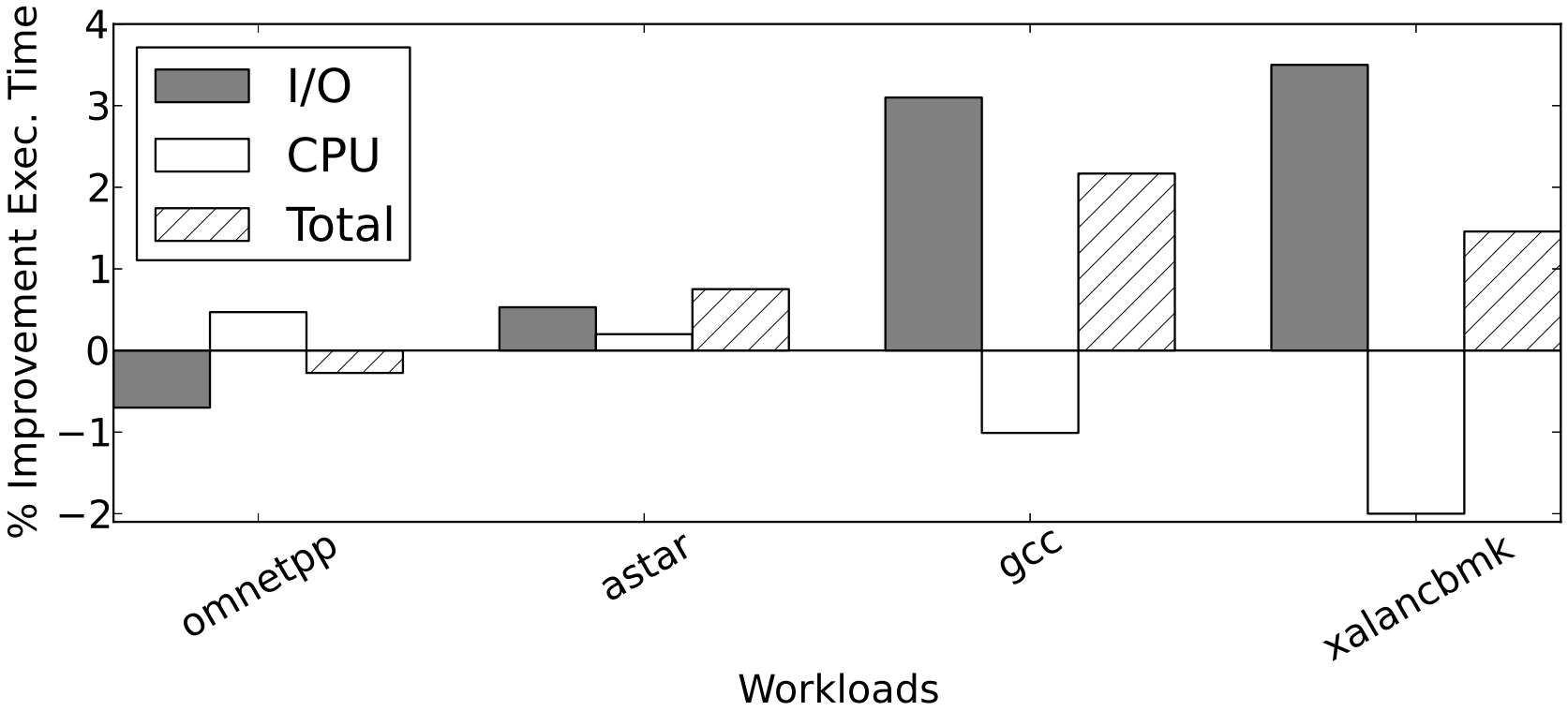
- Average of 5 runs (<2% COV)

NBW Evaluation: Single Thread



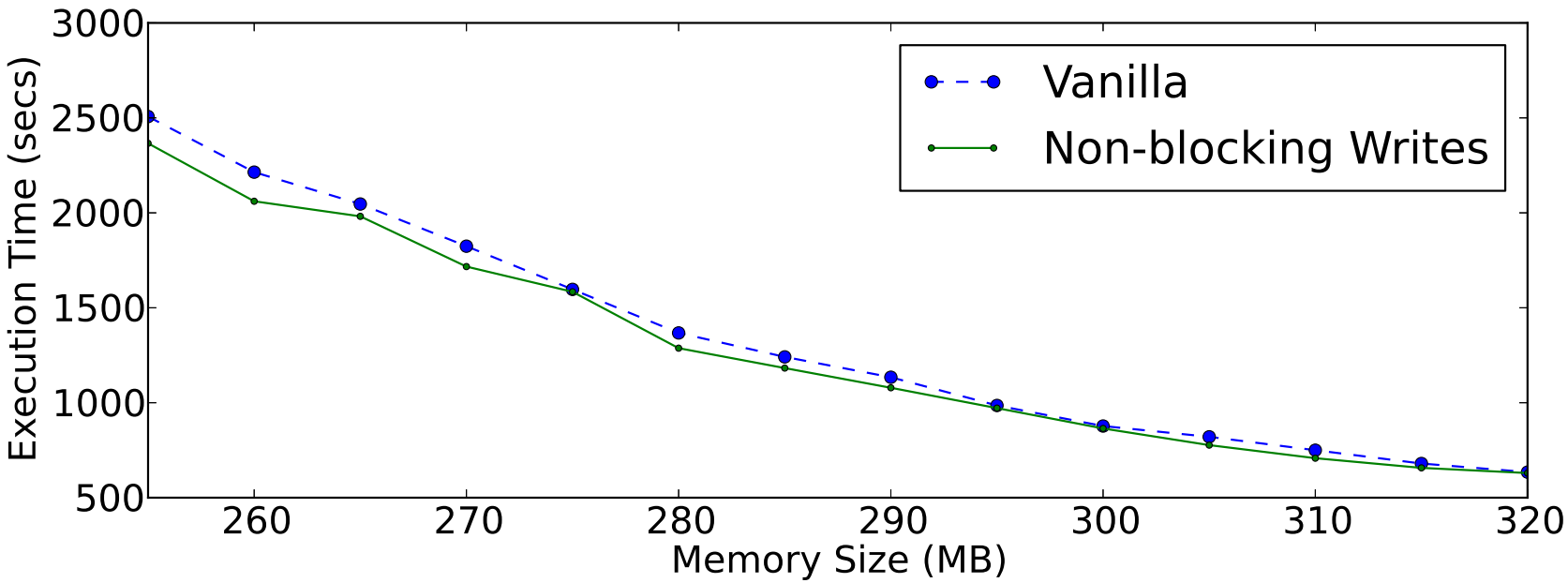
- Average of 5 runs (<2% COV)
- 2.1% gains, higher when memory is reduced
- Memory overhead usually $< 0.1\%$

NBW Evaluation: Single Thread



- Average of 5 runs (<2% COV)
- 2.1% gains, higher when memory is reduced
- Memory overhead usually $< 0.1\%$

NBW Evaluation: Constrained Memory



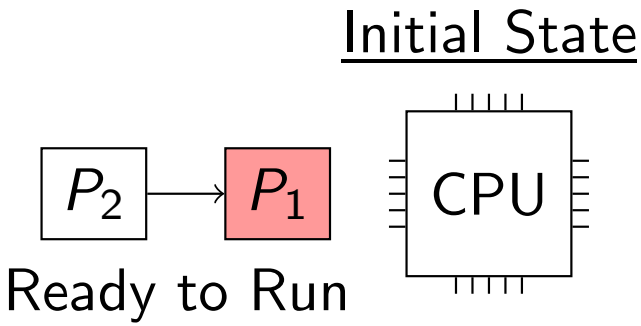
- 5 runs per point (<9.5% COV)
- Improvement increase with lower memory
- Up to 7% improvement (260MB)

NBW Scheduling Problem

Up to 8% degradation with multi-threaded apps

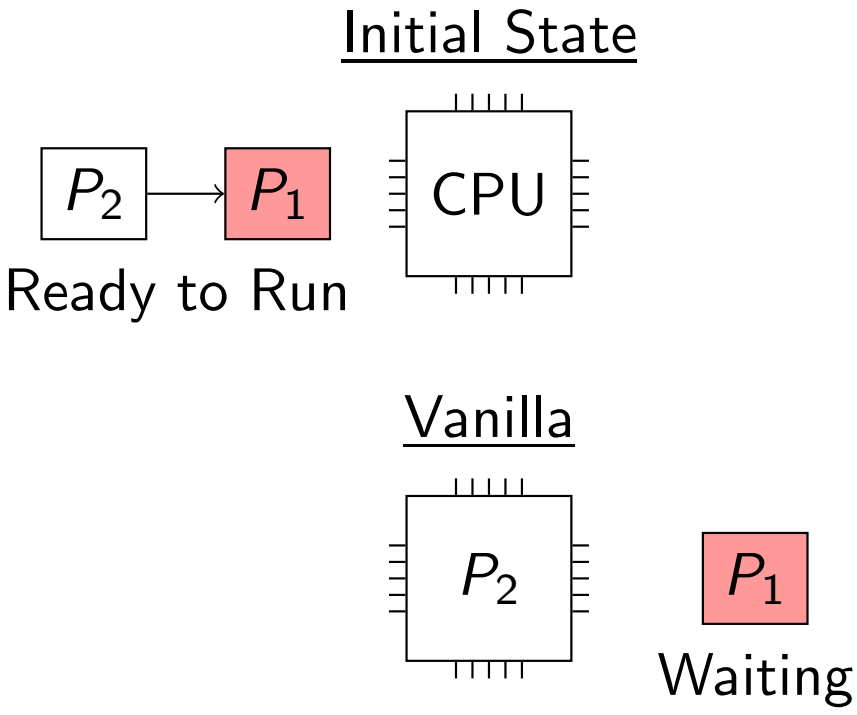
NBW Scheduling Problem

Up to 8% degradation with multi-threaded apps



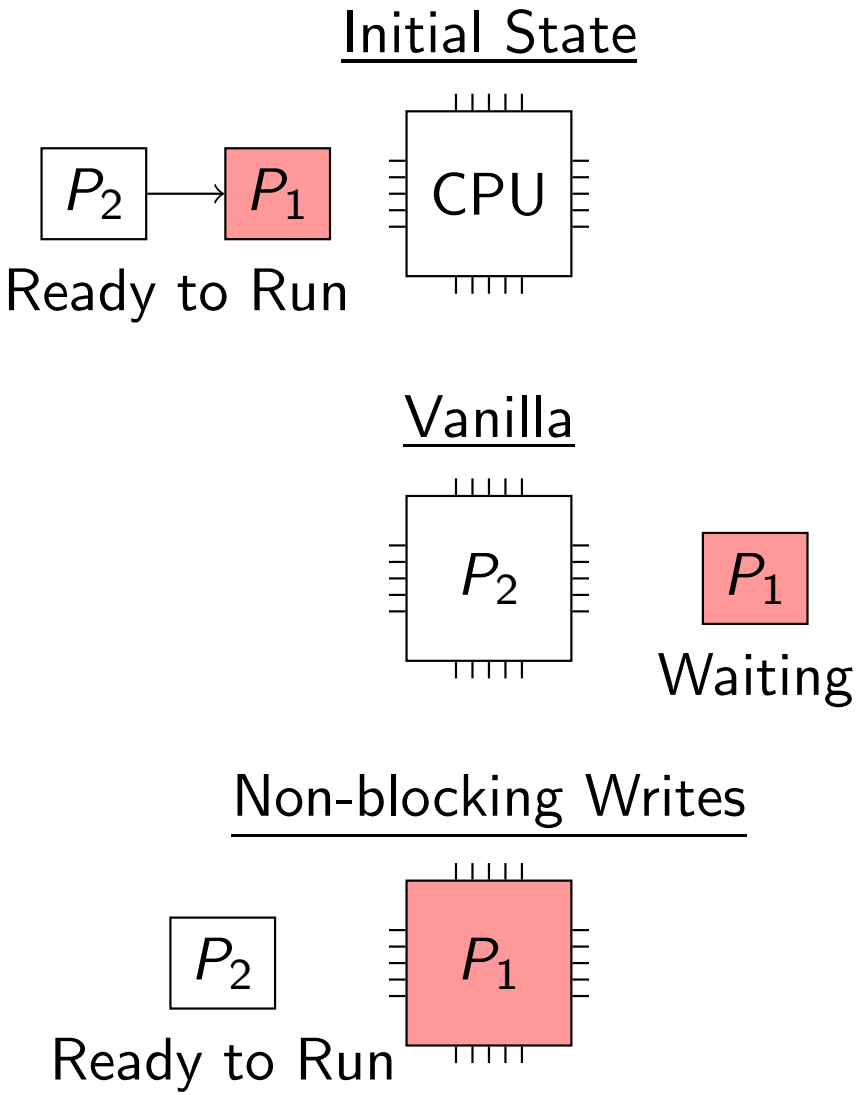
NBW Scheduling Problem

Up to 8% degradation with multi-threaded apps



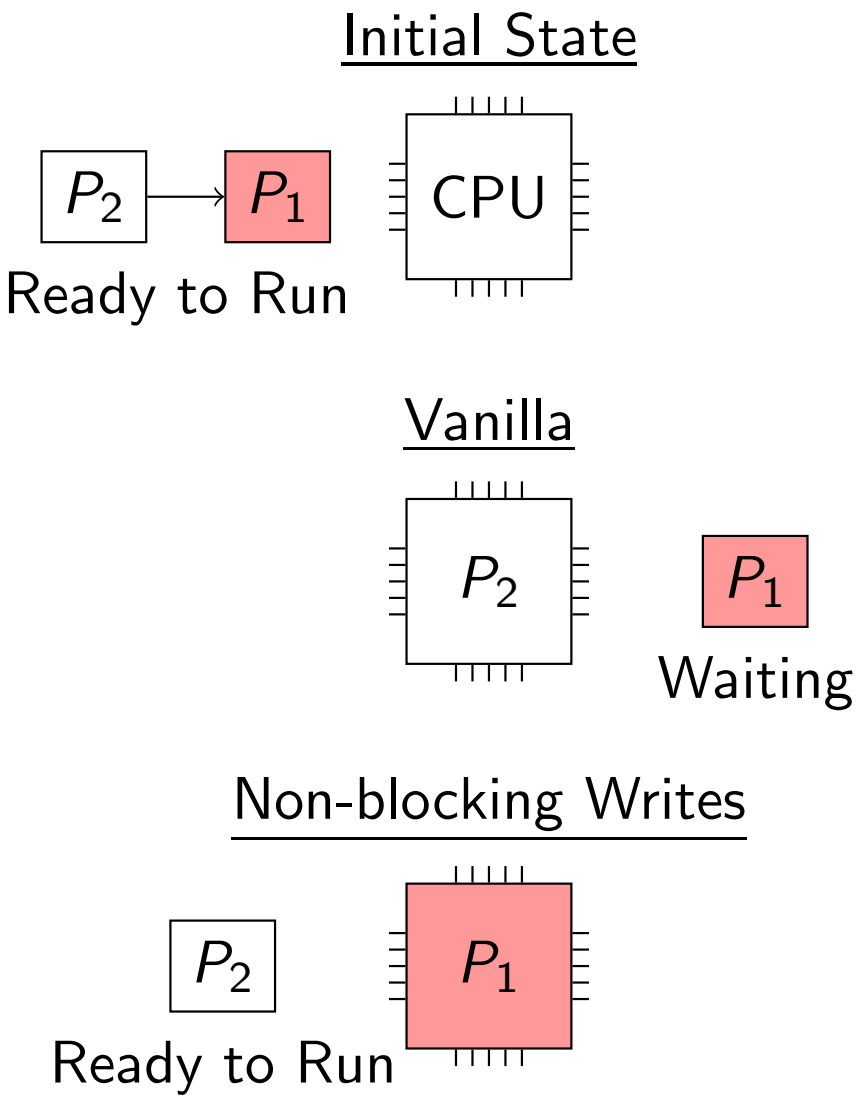
NBW Scheduling Problem

Up to 8% degradation with multi-threaded apps



NBW Scheduling Problem

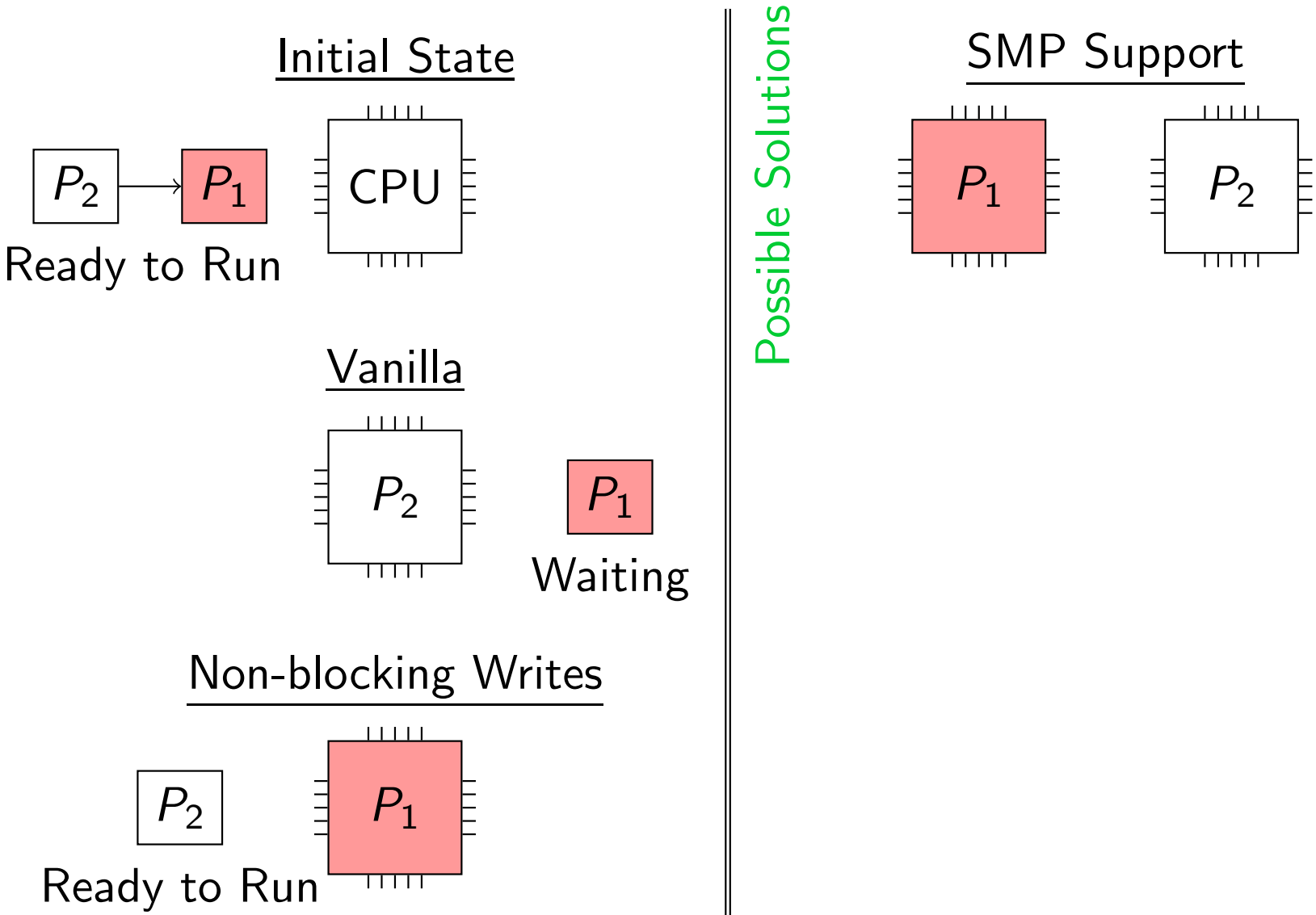
Up to 8% degradation with multi-threaded apps



Possible Solutions

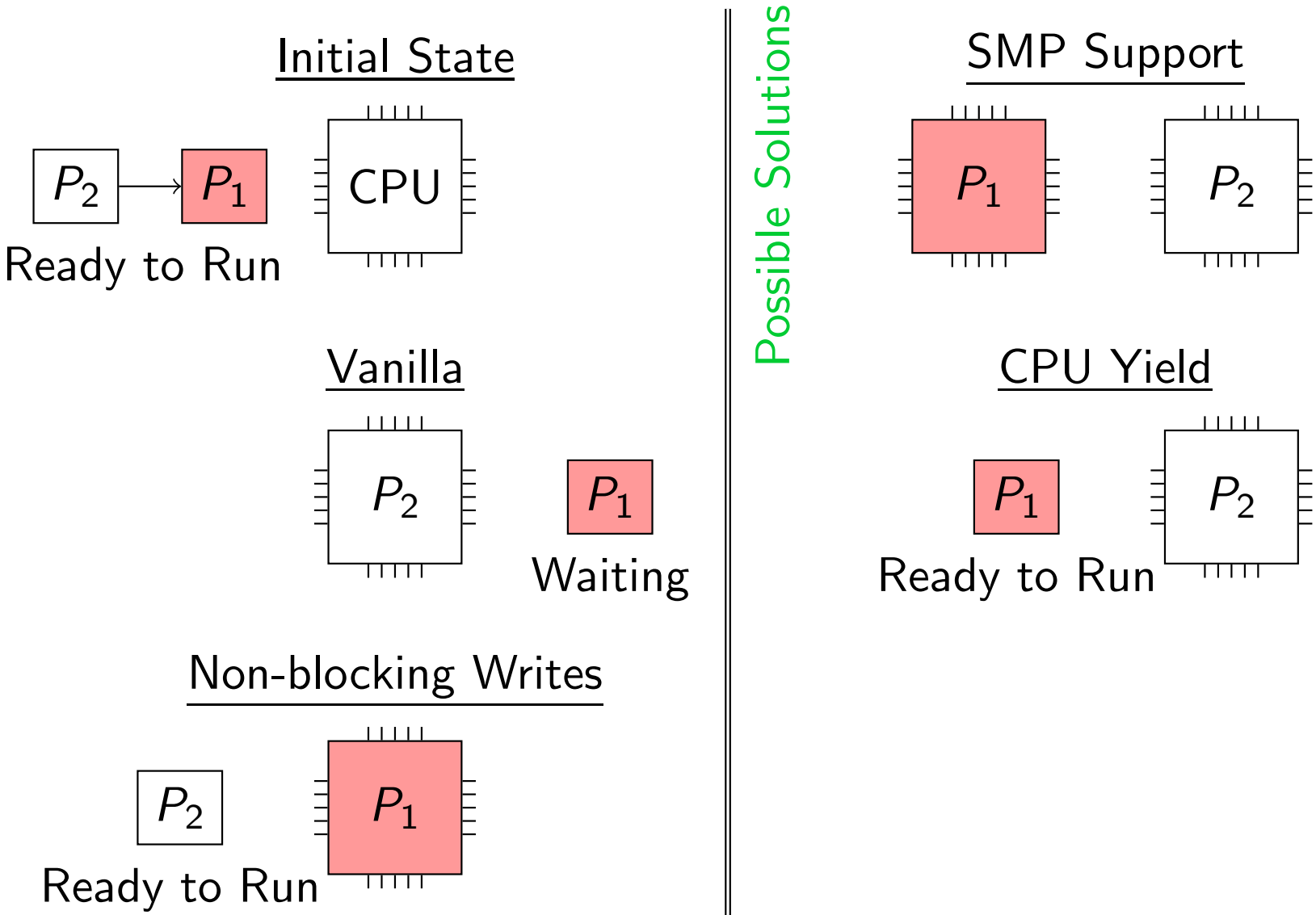
NBW Scheduling Problem

Up to 8% degradation with multi-threaded apps

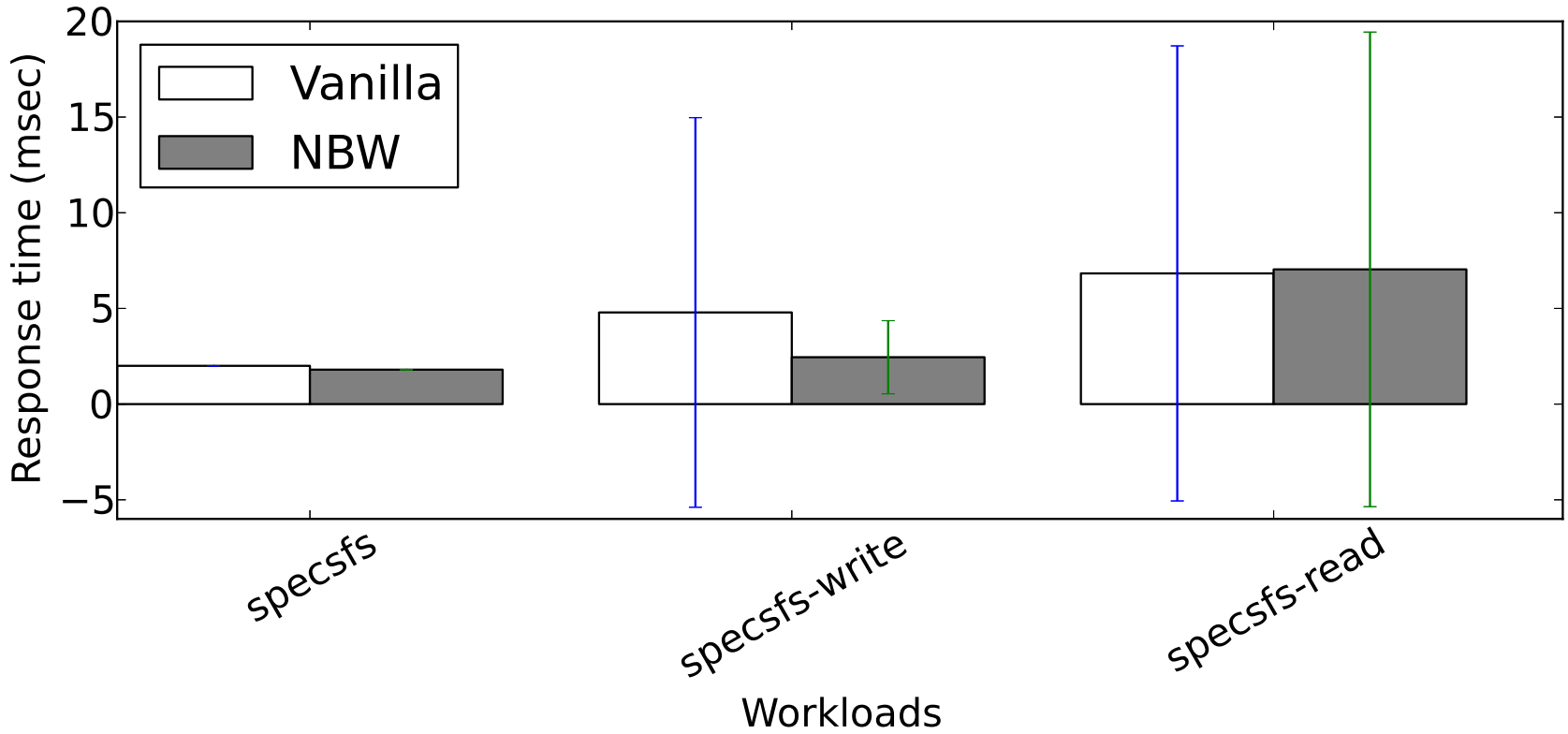


NBW Scheduling Problem

Up to 8% degradation with multi-threaded apps



NBW Evaluation: File System Response Time



- 50% write response time reduction in SPEC SFS
- Much lower variance ($10 \rightarrow 2$)
- Read latency remain unchanged

Non-blocking Writes: Summary

- Non-blocking writes allows asynchronous writes to pages not in memory
- Evaluation with server and developer workloads showed:
  - ✓ Up to 7% reduction in execution time for single-threaded applications
  - ✓ 50% reduction in writes response time for file system workloads
  - ✗ Degradation in performance of multi-threaded workloads

Non-blocking Writes: Summary

- Non-blocking writes allows asynchronous writes to pages not in memory
- Evaluation with server and developer workloads showed:
  - ✓ Up to 7% reduction in execution time for single-threaded applications
  - ✓ 50% reduction in writes response time for file system workloads
  - ✗ Degradation in performance of multi-threaded workloads

Impact

- Potential to increase idleness of disks in caching systems (SRCMap and EXCES)
- Decouples write response time from backing store performance
- Could improve viability of reduced-memory energy-efficient systems

Conclusions

- It is important to solve the energy problem in disks and memory
- Caching techniques can be used to save energy in disks:
  - ✓ 36%-59% energy savings in multi-disk systems
  - ✓ 2%–14% energy savings in single-disk systems
- We eliminate the page fetch-before-update with NBW:
  - ✓ Improves execution time in single-threaded applications
  - ✓ Write latency in file server workloads
  - ✓ Solutions proposed to sub-optimal scheduling problem in multi-threaded apps
  - ✓ Potential to improve power in EXCES and SRCMap
  - ✓ Potential to improve performance in resource constrained systems

Future Work

The Memory Challenge

- SSDs are becoming more popular $\Rightarrow$ Storage energy share will be smaller
- DRAM size continues to increase

Direction 1

Explore the effectiveness of reducing memory with NBW help

Direction 2

Use NBW to artificially reduce the page size

Publications (Presented Here)

Publication	Refs.
Design and Implementation of Non-blocking Writes Luis Useche, Ricardo Koller, Jesus Ramos, Raju Rangaswami, *Under Review*	–
Truly Non-blocking Writes Luis Useche, Ricardo Koller, Raju Rangaswami, Akshat Verma, *HotStorage 2011*	–
SRCMap: Energy Proportional Storage Using Dynamic Consolidation Akshat Verma, Ricardo Koller, Luis Useche, Raju Rangaswami, *FAST 2010*	23
EXCES: EXternal Caching in Energy Saving Storage Systems Luis Useche, Jorge Guerra, Medha Bhadkamkar, Mauricio Alarcon, and Raju Rangaswami *HPCA 2008*	22

Additional Publications

Publication	Refs.
BORG: Block-reORGanization for Self-optimizing Storage Systems Medha Bhadkamkar, Jorge Guerra, Luis Useche, Sam Burnett, Jason Liptak, Raju Rangaswami and, Vagelis Hristidis *FAST 2009*	53
The Case for Active Block Layer Extensions Jorge Guerra, Luis Useche, Medha Bhadkamkar, Ricardo Koller, and Raju Rangaswami *SPEED 2008*	3

Aknowledgement

Dissertation Committee

- Raju Rangaswami
- Giri Narasimhan
- Ajay Gulati
- Kaushik Dutta
- Ming Zhao

Collaborators

- Raju Rangaswami
- Ricardo Koller
- Jesus Ramos
- Jorge Guerra
- Akshat Verma