Truly Non-blocking Writes

Luis Useche[2] Ricardo Koller[2] Raju Rangaswami[2]
Akshat Verma[1]

[1]IBM Research, India

[2]School of Computing and Information Sciences
College of Engineering and Computing

FLORIDA INTERNATIONAL UNIVERSITY

HotStorage Workshop, 2011

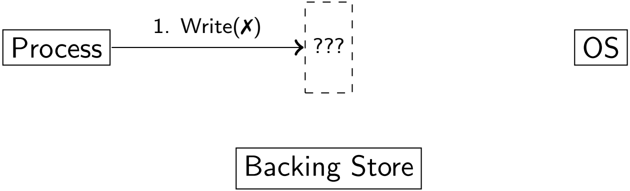
Introduction

- Memory access granularity is smaller than disk's

Introduction

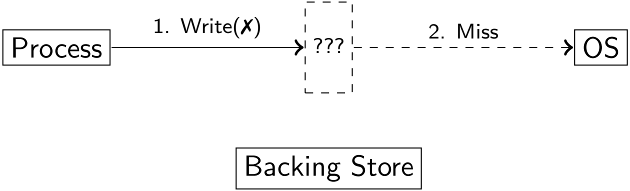
- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.

- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.



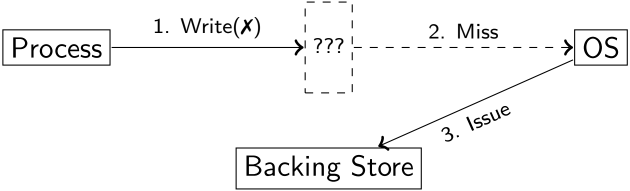
Introduction

- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.



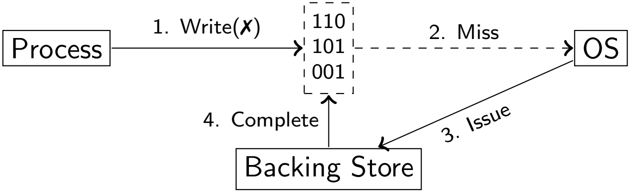
Introduction

- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.



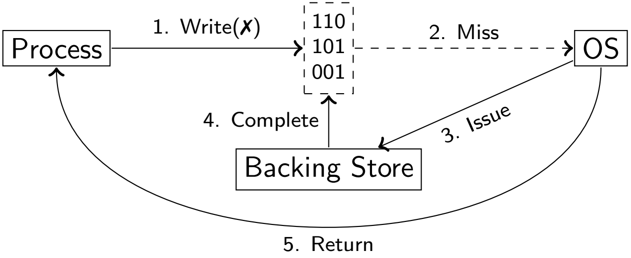
Introduction

- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.

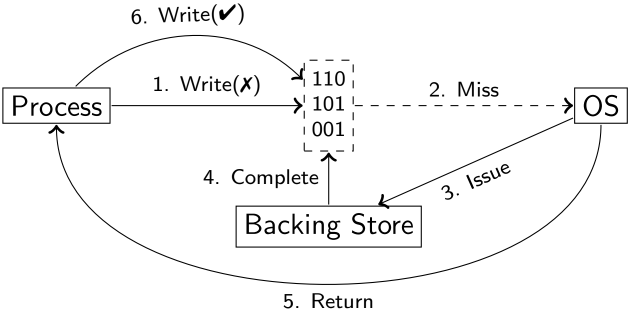


Introduction

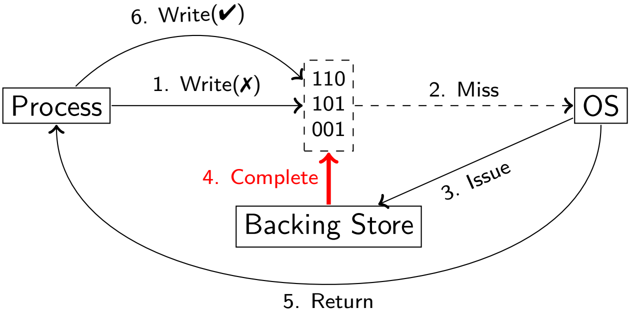
- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.



- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.

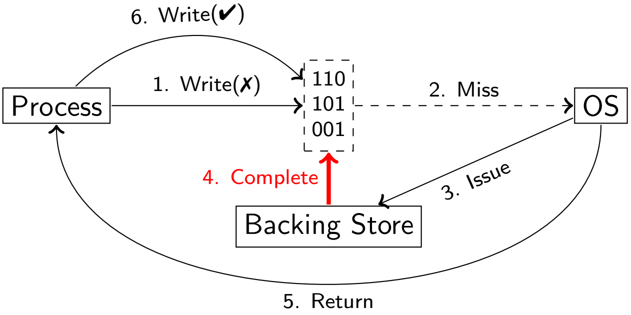


- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.

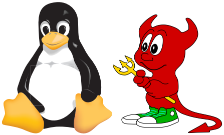


For writes: why wait for data that the application doesn't need?

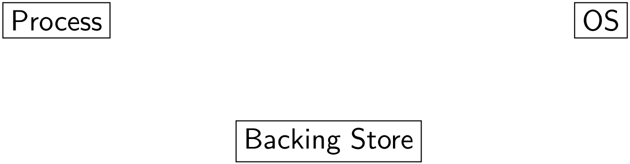
- Memory access granularity is smaller than disk's ⇒ Writes to an out-of-core page require a full page fetch.



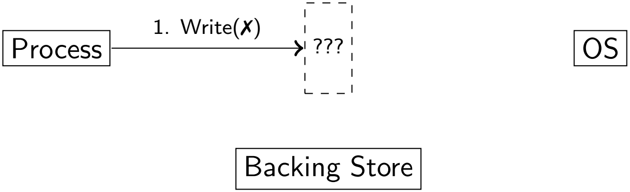
For writes: why wait for data that the application doesn't need?



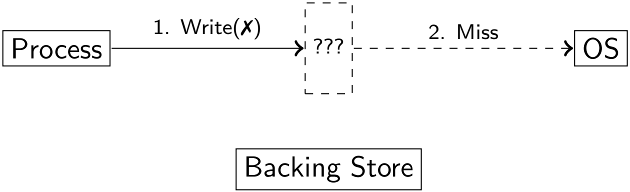
Non-blocking Writes: Basic Approach



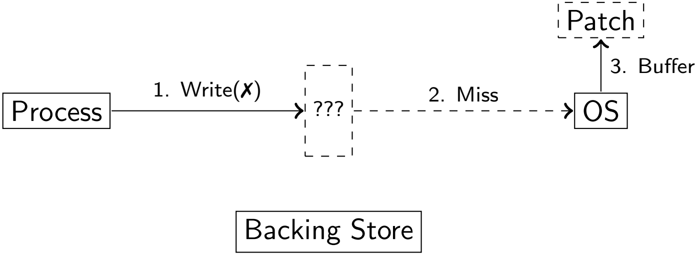
Non-blocking Writes: Basic Approach



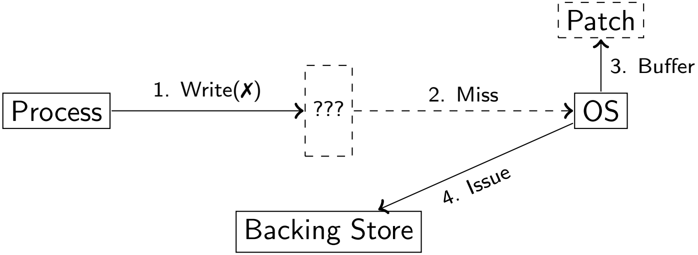
Non-blocking Writes: Basic Approach



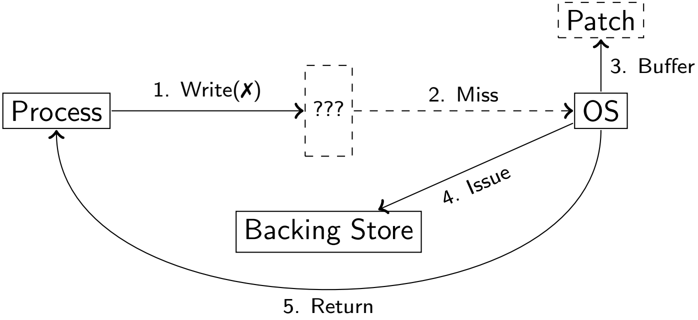
Non-blocking Writes: Basic Approach



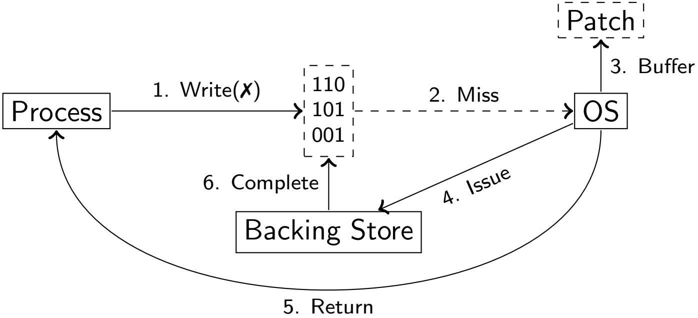
Non-blocking Writes: Basic Approach



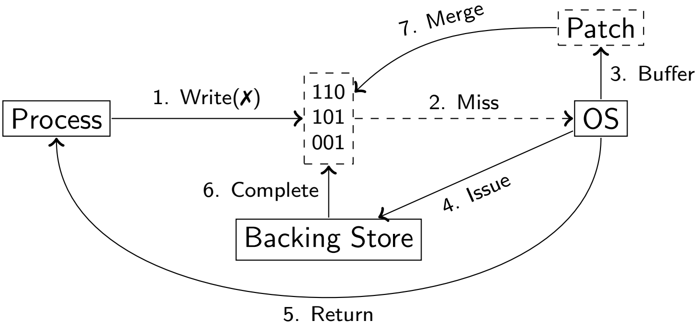
Non-blocking Writes: Basic Approach



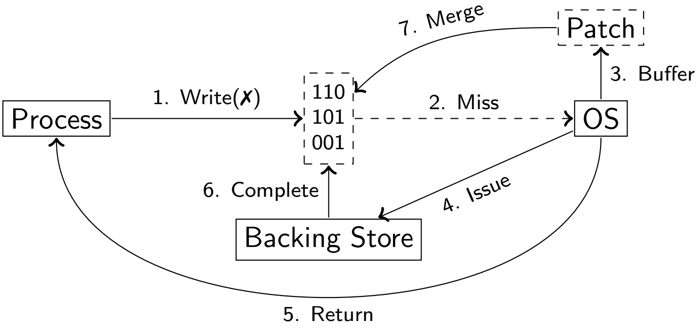
Non-blocking Writes: Basic Approach



Non-blocking Writes: Basic Approach



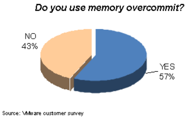
Non-blocking Writes: Basic Approach



Benefits

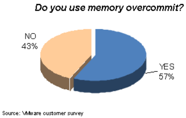
1. Application execution time reduction
2. Increased backing store bandwidth usage

Motivation → Higher Fault Rates

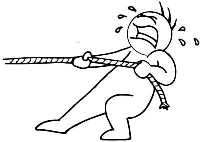


Memory over-committed in virtualized environments

Motivation → Higher Fault Rates

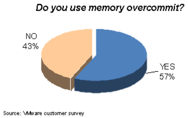


Memory over-committed in virtualized environments

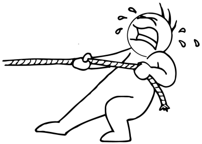


More process running with multi-core and virtualized environments

Motivation → Higher Fault Rates



Memory over-committed in virtualized environments



More process running with multi-core and virtualized environments

Memory hierarchy moving towards a more active and faster backing store

Motivation $\rightarrow$ % Non-blocking faults

- We calculate the % of faults that can benefit in all our workloads

Motivation → % Non-blocking faults

- We calculate the % of faults that can benefit in all our workloads:

  Image Processing Rendering of SVG images
  Developer Unit and performance testing
  Server Application, database, and mail server

Motivation → % Non-blocking faults

- We calculate the % of faults that can benefit in all our workloads:

  Image Processing: Rendering of SVG images

  Developer: Unit and performance testing

  Server: Application, database, and mail server
- Simulator with full-system memory traces.
- RAM set to 50% of app footprint

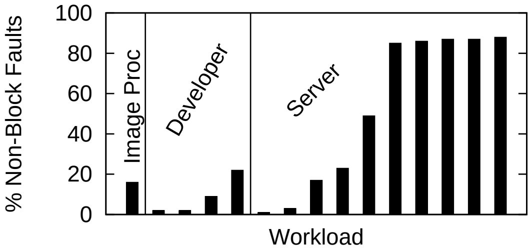
Motivation → % Non-blocking faults

- We calculate the % of faults that can benefit in all our workloads:

  Image Processing — Rendering of SVG images

  Developer — Unit and performance testing

  Server — Application, database, and mail server
- Simulator with full-system memory traces.
- RAM set to 50% of app footprint
- Up to 80% of page faults benefit



Related Work

Alternatives to non-blocking writes:

Perfect DRAM Provision

Unpredictable or unbounded.

Prefetching

Can incur false positives and false negatives.

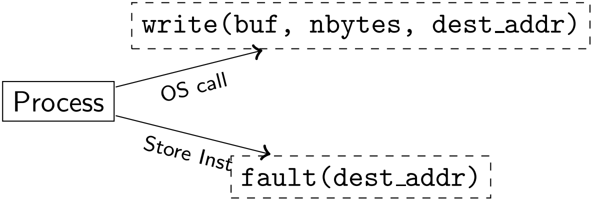
Asynchronous System Calls

1. Do not work with memory mapped pages
2. Written data *not* immediately available for reading

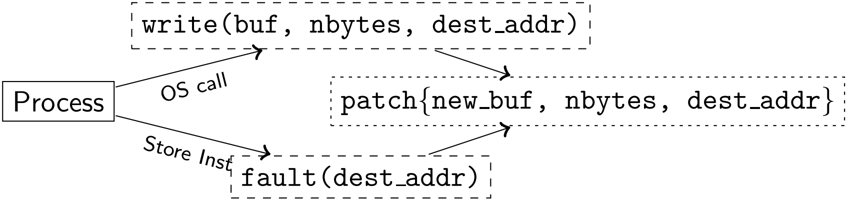
Solution Challenges

Process

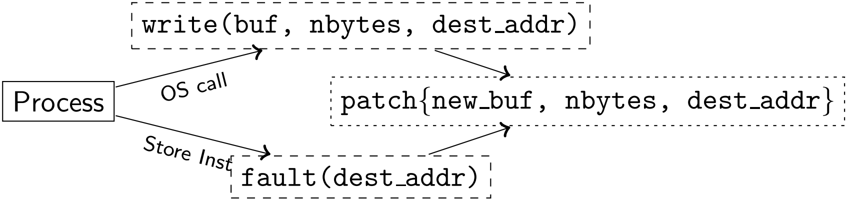
Solution Challenges



Solution Challenges



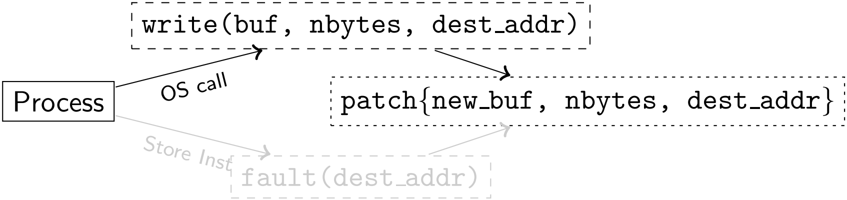
Solution Challenges



Information Per Non-blocking Write

Information
Write Offset
Data Written
Size of Data

Solution Challenges



Information Per Non-blocking Write

Information	Supervised `write()`
Write Offset	✔
Data Written	✔
Size of Data	✔

Solution Challenges

Information Per Non-blocking Write

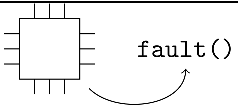
Information	Supervised write()	Unsupervised Fault
Write Offset	✔	✔
Data Written	✔	✗
Size of Data	✔	✗

Solution Challenges

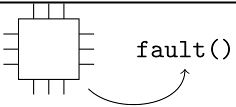
Information Per Non-blocking Write

Information	Supervised `write()`	Unsupervised Fault
Write Offset	✔	✔
Data Written	✔	✗
Size of Data	✔	✗

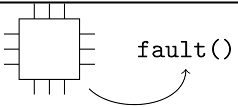
Handling Unsupervised Writes

Approach	Description	Fast	All Arch?	Low Mem?
Full Feature Hardware		✔	✗	✔

Handling Unsupervised Writes

Approach	Description	Fast	All Arch?	Low Mem?
Full Feature Hardware		✔	✗	✔
Opcode Disassembly	4 bytes offset `sw $t1, 0xff` data	✔	✗	✔

Handling Unsupervised Writes

Approach	Description	Fast	All Arch?	Low Mem?
Full Feature Hardware		✔	✗	✔
Opcode Disassembly	4 bytes offset `sw $t1, 0xff` data	✔	✗	✔
Page Diff-Merge	Disk Page or 0-buffer and 1-buffer Updated Page	✗	✔	✗

Quantifying Benefits

1. Fraction of non-blocking write faults ✔
2. Outstanding write faults (over time)
3. Savings in execution time (new!)

Virtual Memory Simulator

Input RAM size & Full System Memory Traces

Output Performance statistics

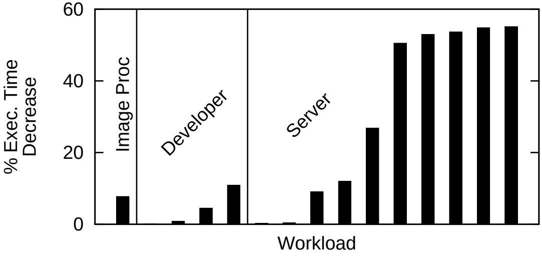
- Memory size set to 50% of workloads footprint
- Creating patches is not required

Quantifying Benefits → Metric

- How to measure the additional parallelism?
- **Outstanding Write Faults (OWF):** # of parallel write faults at any time
  - ✓ OWF $\leq$ OIO
  - ✓ OWF $\leq$ 1 for single threaded applications
  - ✓ OWF $\geq$ 0 when using non-blocking writes
- We need the variations over time as well
- **E[**OWF**]**: time-weighted average OWF

Quantifying Benefits → Time Reduction

- These results are not in the paper
- Execution time = Trace time + Synchronous read time
- Write time of dirty page on evictions ignored
- Rough estimate: error proportional to the number of dirty pages evicted



Conclusions and Future Work

- We presented non-blocking writes: a technique to eliminate read-before-writes
  - ✓ Reduced execution time
  - ✓ Increased device usage
- We estimate a reduction times of 0.1-54%
- In the future, we are planning to implement non-blocking writes to better study its implications
  - ✓ What workloads benefit from Non-blocking writes?

Questions?

Virtual Memory Simulator

Input: RAM size & Mem Traces

Output: Per Entry: Timestamp and event (hit, miss, evict);
Global: Performance stats.

- Writes to out-of-core pages considered non-blocking
- Non-blocking status revoked when:
  1. The page is read before I/O completion
  2. The page is evicted before I/O completion

Quantifying Benefits → Full System Memory Traces

Modified x86 software-MMU QEMU to log all memory accesses:

- *Instruction count, CR3, virtual/physical address, access-mode, page privileges.*

Workloads

Type	#	Footprint Avg/Std (MB)
Server	10	294/158
Developer	4	269/183
Image	1	149/0

Solution Approaches → Page Diff-Merge

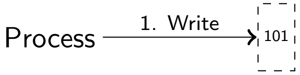
1. Write in two pages: 0-page and 1-page.
2. Merge with *and* and *or*.

Process

Backing Store

Solution Approaches → Page Diff-Merge

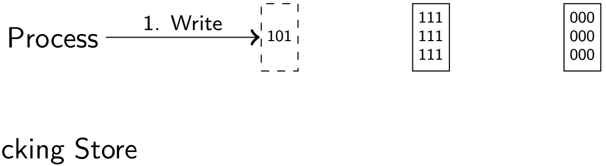
1. Write in two pages: 0-page and 1-page.
2. Merge with *and* and *or*.



Backing Store

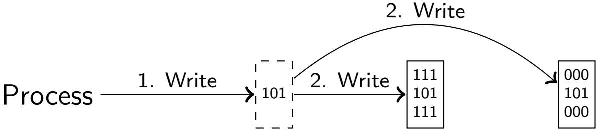
Solution Approaches → Page Diff-Merge

1. Write in two pages: 0-page and 1-page.
2. Merge with *and* and *or*.



Solution Approaches → Page Diff-Merge

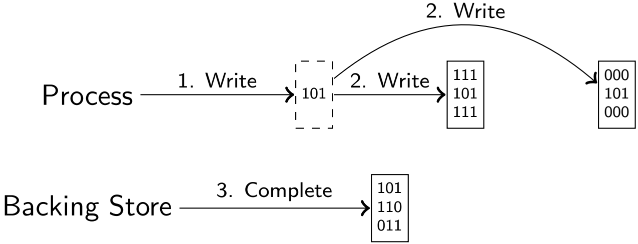
1. Write in two pages: 0-page and 1-page.
2. Merge with *and* and *or*.



Backing Store

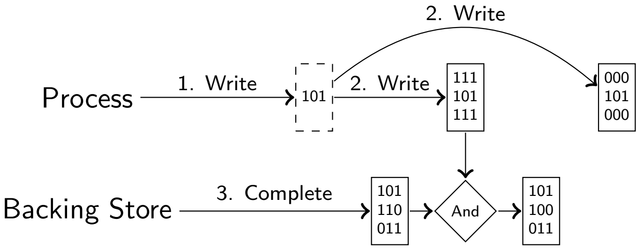
Solution Approaches → Page Diff-Merge

1. Write in two pages: 0-page and 1-page.
2. Merge with *and* and *or*.



Solution Approaches → Page Diff-Merge

1. Write in two pages: 0-page and 1-page.
2. Merge with *and* and *or*.



Solution Approaches → Page Diff-Merge

1. Write in two pages: 0-page and 1-page.
2. Merge with *and* and *or*.