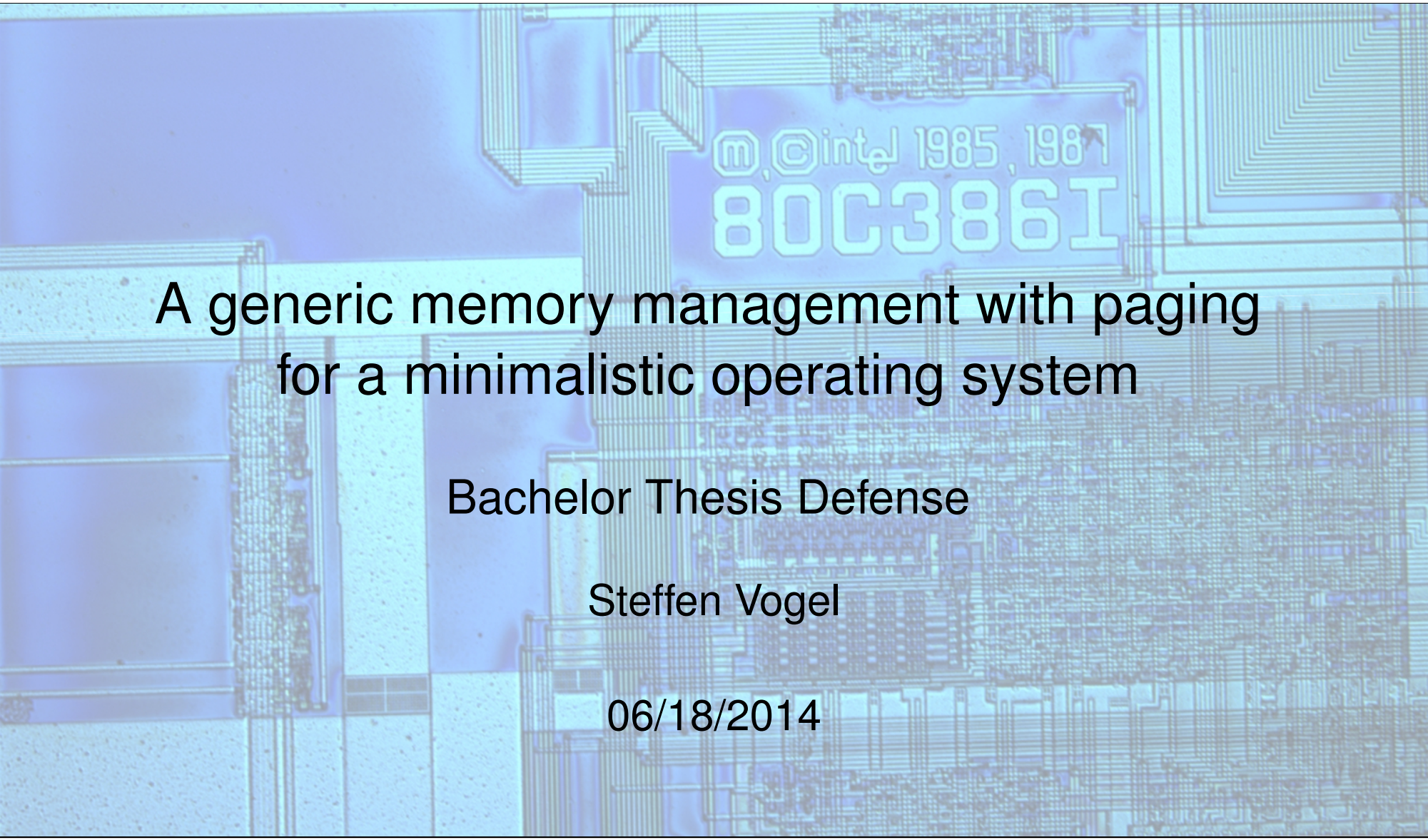


A generic memory management with paging for a minimalistic operating system

Bachelor Thesis Defense

Steffen Vogel

06/18/2014

E.ON Energy Research Center

RWTH AACHEN UNIVERSITY

Motivation

- Add support for 64 bit userspace in MetalSVM
  - 32 bit only systems become rare
  - ... also for embedded systems
  - Intel SCC $\Rightarrow$ Intel Xeon Phi
- Find the most minimalistic way to implement paging
- Try something different to known Unixes / Linuxes / BSDs
- Explore the limits of the x86 memory architecture[1]

[1] The Intel x86 MMU is turing complete [1]. ☺

Agenda

- Objective
- MetalSVM
- Paging
  - Self-mapped Page Tables
- Conclusion
  - Skipped parts
- Outlook

Objective

- Full paging support of 64 bit userspace processes
- Improve code quality and perspicuity
- Unified implementation for 32 and 64 bit

Focus on virtual memory. Not:

- Demand-Paging, Swapping, Segmentation
- Copy-on-Write
- NUMA[2] optimization
- Physical memory allocation

[2] Non-Uniform Memory Access.

Objective

- Full paging support of 64 bit userspace processes
- Improve code quality and perspicuity
- Unified implementation for 32 and 64 bit

Focus on virtual memory. Not:

- Demand-Paging, Swapping, Segmentation
- Copy-on-Write
- NUMA[2] optimization
- Physical memory allocation

[2] Non-Uniform Memory Access.

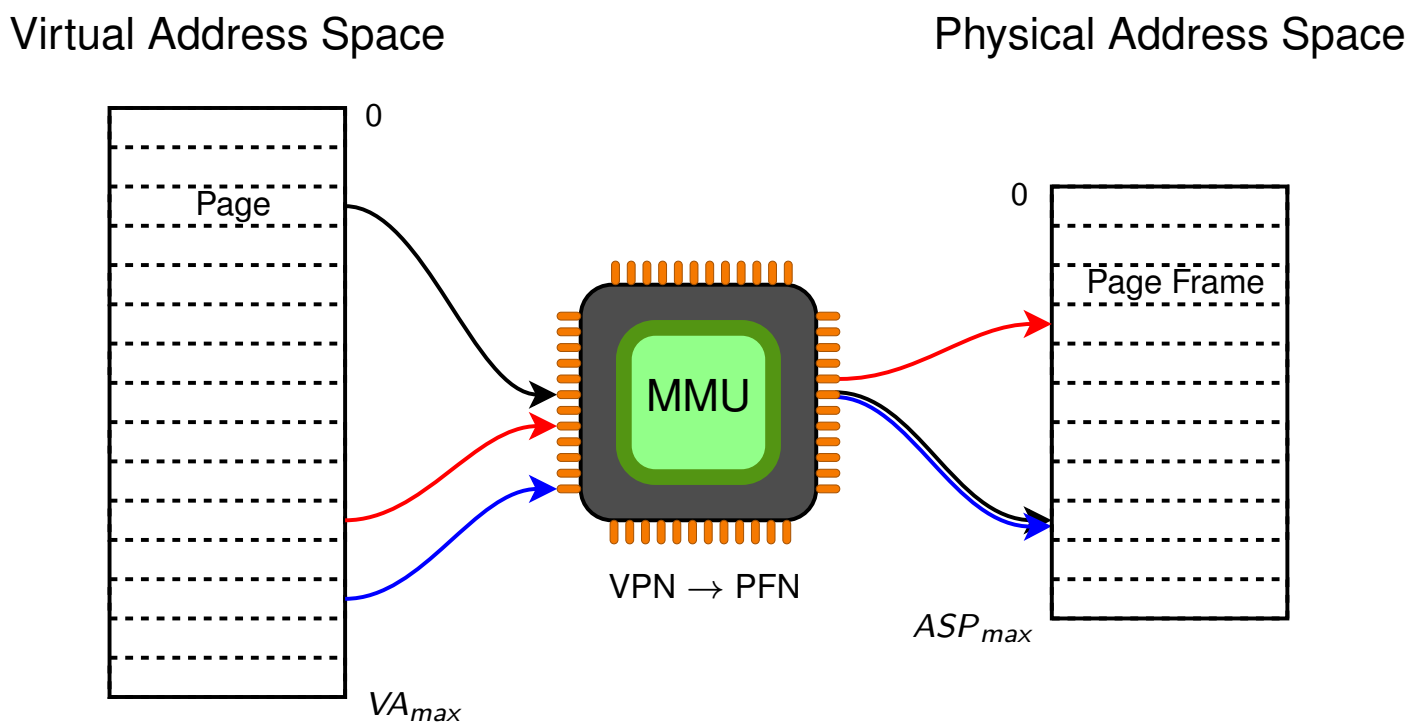
MetalSVM

- Minimal research operating system[3]
- Developed at the former LfBS [5]
- Targeted at Intel's x86 architecture (IA-32)
  - Supports 64 bit extensions (Intel 64, IA-32e)
  - Simple ARM port available
  - Hypervisor for Intel's Single-chip Cloud Computer (SCC)
- Spin-off of eduOS
  - Kernel used for education at the RWTH[4]
  - "Operating Systems" lectured at the ACS

[3] http://www.metalsvm.org/.

[4] http://www.github.com/RWTH-OS/eduOS.

Paging Basics

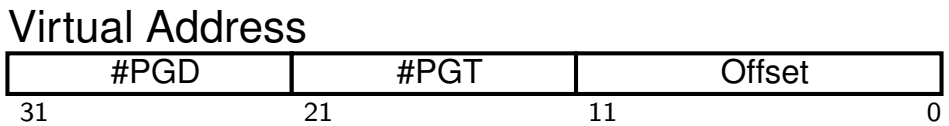


Intel's x86 Architecture

- Address translation by page tables
  - Hierarchical lookup tables compose a *search tree*
  - Reduced size in contrary to flat table
  - Tables reside in the main memory
- With up to four levels of indirection:
  - **PML4** Page Map Level 4
  - **PDPT** Page Directory Pointer Table
  - **PGD** Page Directory
  - **PGT** Page Table
- Translation Lookaside Buffer (TLB)

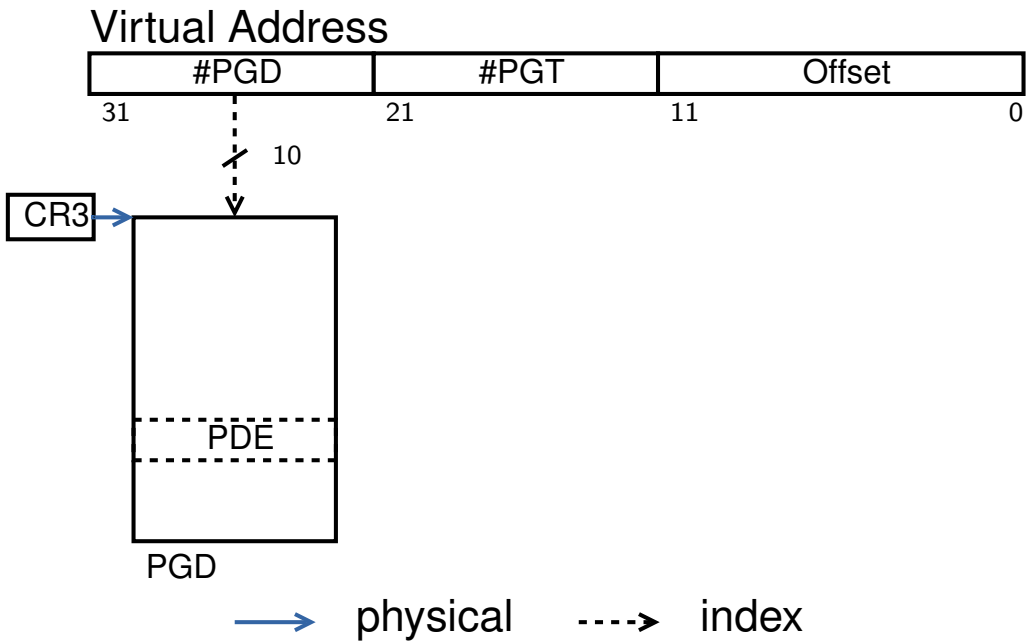
Page Table Walk: 32 bit

There are two levels of tables in legacy 32 bit [4]:



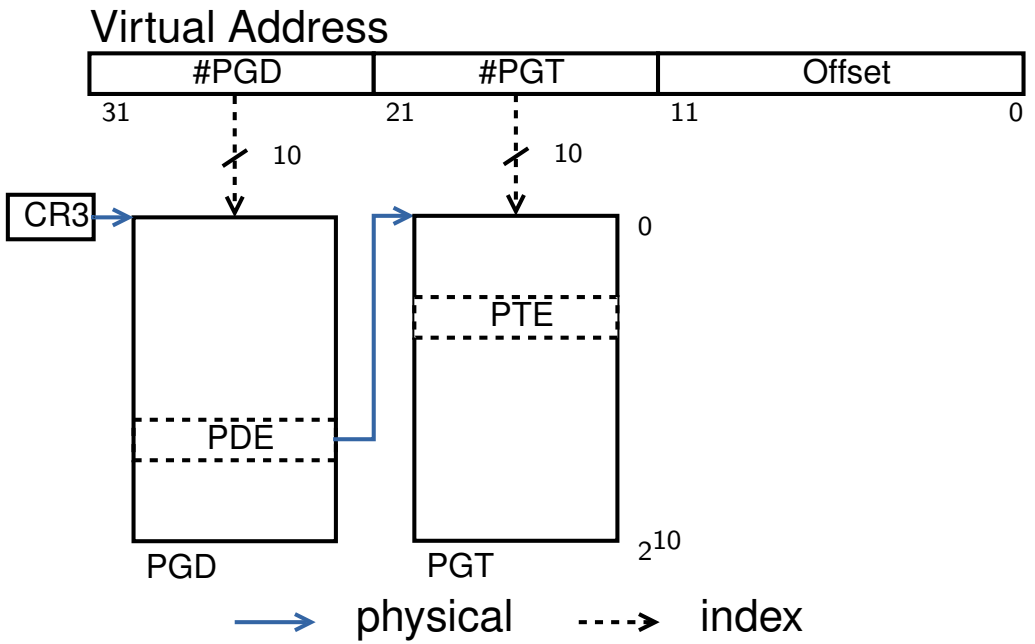
Page Table Walk: 32 bit

There are two levels of tables in legacy 32 bit [4]:



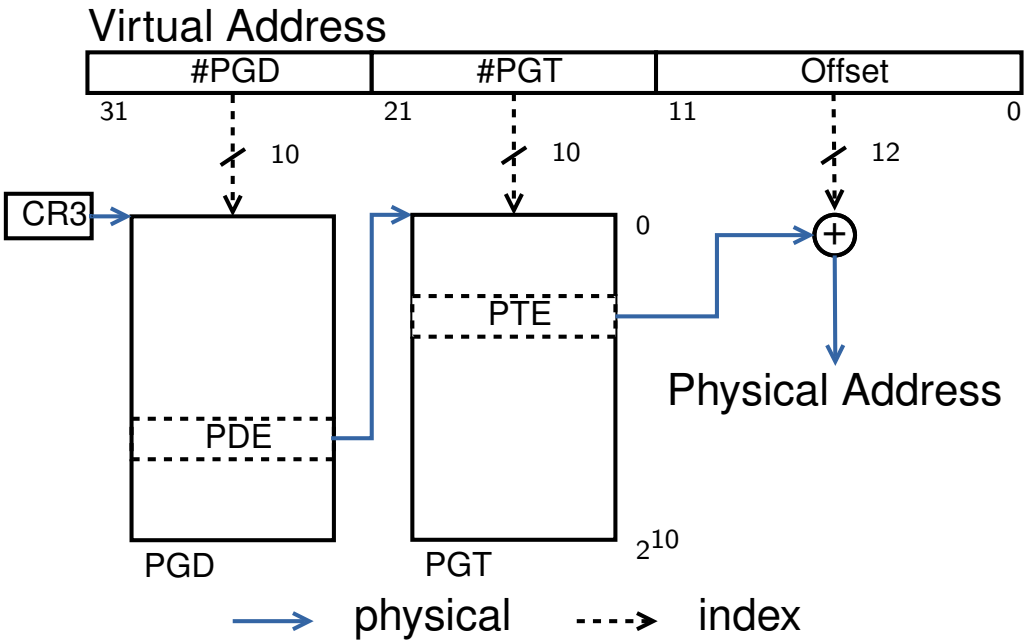
Page Table Walk: 32 bit

There are two levels of tables in legacy 32 bit [4]:



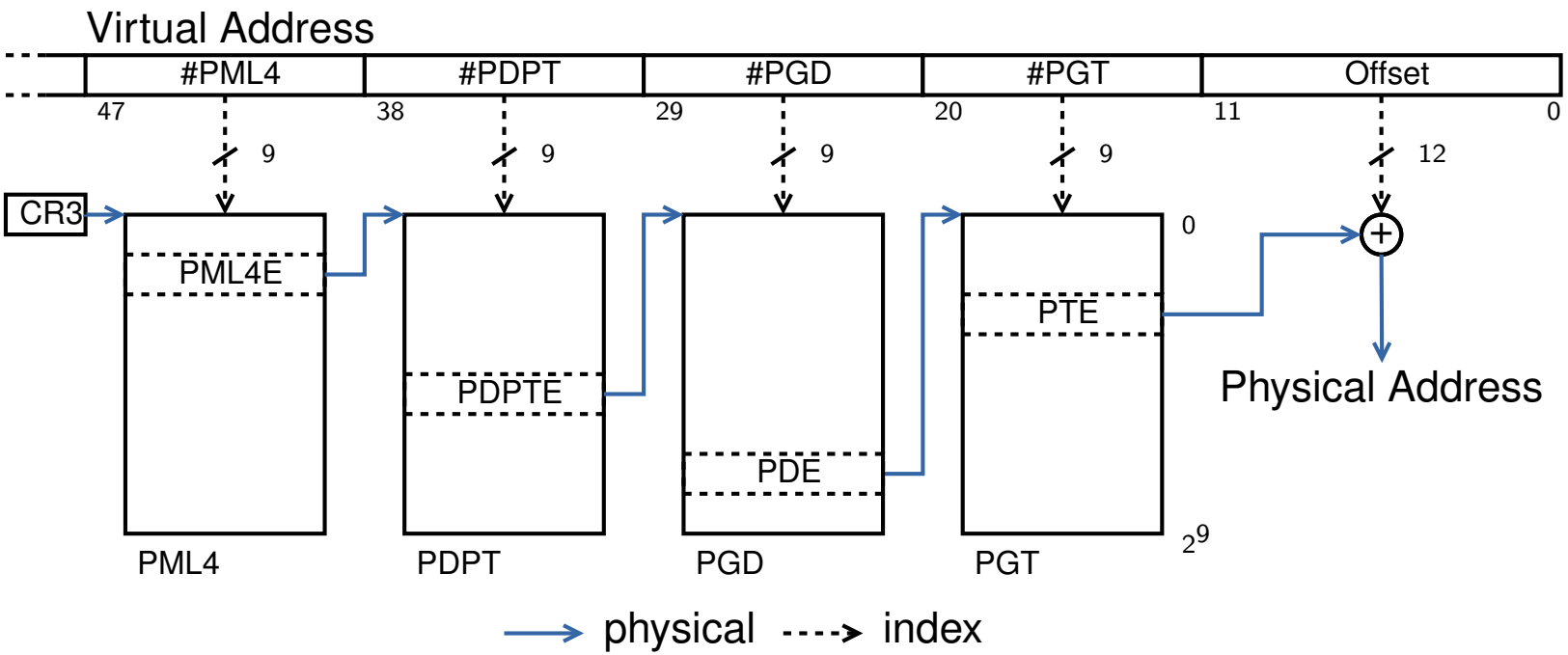
Page Table Walk: 32 bit

There are two levels of tables in legacy 32 bit [4]:



Multi Level Paging: 64 bit

Larger address space requires more levels of indirection:



Operations on Tables

Address translations are performed by hardware (MMU).
OS only needs the ability to modify the tables:

- Map page frames — `map_region()`
- Un-map page frames — `unmap_region()`
- Copy a whole page map tree — `copy_page_map()`
- Delete a whole page map tree — `drop_page_map()`
- Change properties of a mapping — `set_page_flags()`

Operations on Tables

Address translations are performed by hardware (MMU).
OS only needs the ability to modify the tables:

- Map page frames — `map_region()`
- Un-map page frames — `unmap_region()`
- Copy a whole page map tree — `copy_page_map()`
- Delete a whole page map tree — `drop_page_map()`
- Change properties of a mapping — `set_page_flags()`

Operations on Tables

Address translations are performed by hardware (MMU).
OS only needs the ability to modify the tables:

- Map page frames — `map_region()`
- Un-map page frames — `unmap_region()`
- Copy a whole page map tree — `copy_page_map()`
- Delete a whole page map tree — `drop_page_map()`
- Change properties of a mapping — `set_page_flags()`

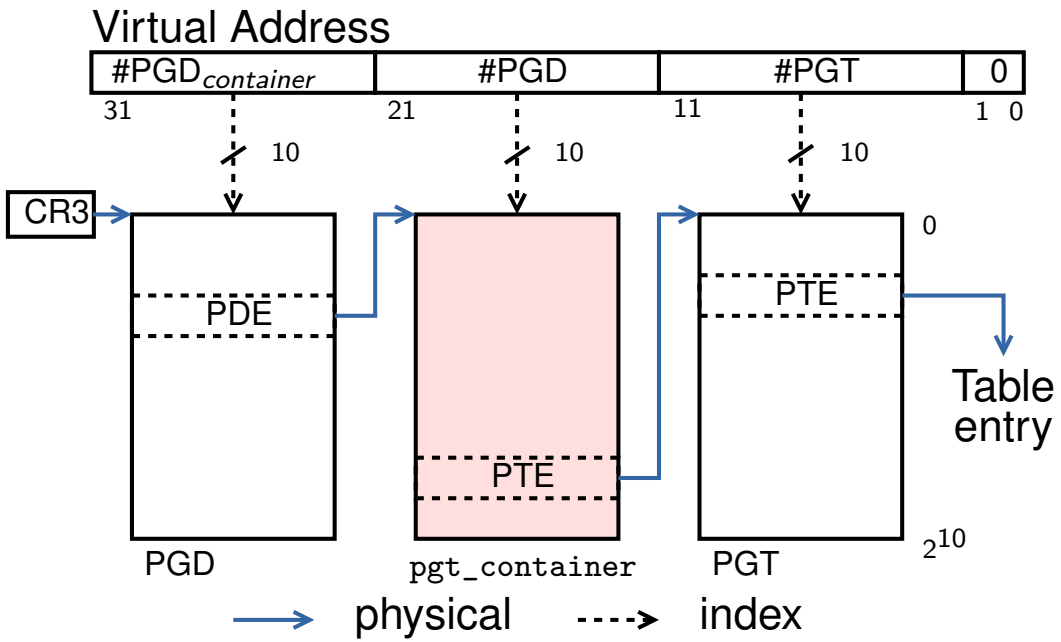
Problem

- Page tables are referenced by physical addresses
  - Everything else would incur a endless recursion
- OS can only access virtual addresses directly[5]
- We need to map the tables into the virtual address space!

[5] At least in 64 bit mode.

Previous Approach

MetalSVM used to have an additional table for this purpose.



Drawbacks

- Additional space required for containers
  - Multiple containers per process
  - Multiple containers per paging level
- Managable for 32 bit $\Rightarrow$ becomes tricky for 64 bit
  - Tables and containers have to be kept in sync
  - Free/Allocate space of containers *and* tables

Avoid too much Containers

Self-mapped Page Tables

SOLUTION:

Use *self-references* to reuse the root-table as a container.

- No containers required
  - No waste of memory due to overhead
  - Only virtual address space is occupied
- *All page tables* of the *current* process are mapped in the Virtual Address Space

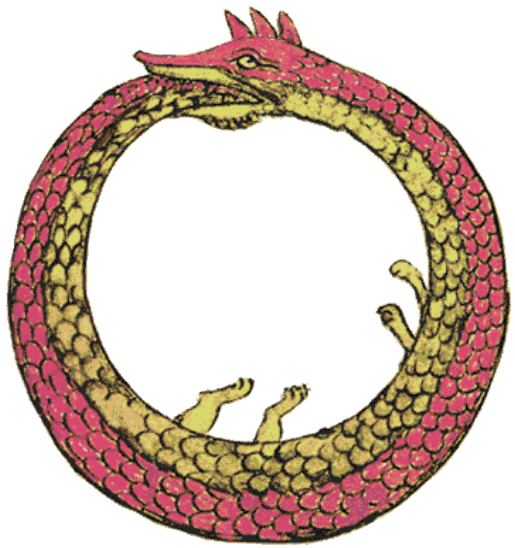


Figure: The ouroboros.

Self-mapped Page Table Walk

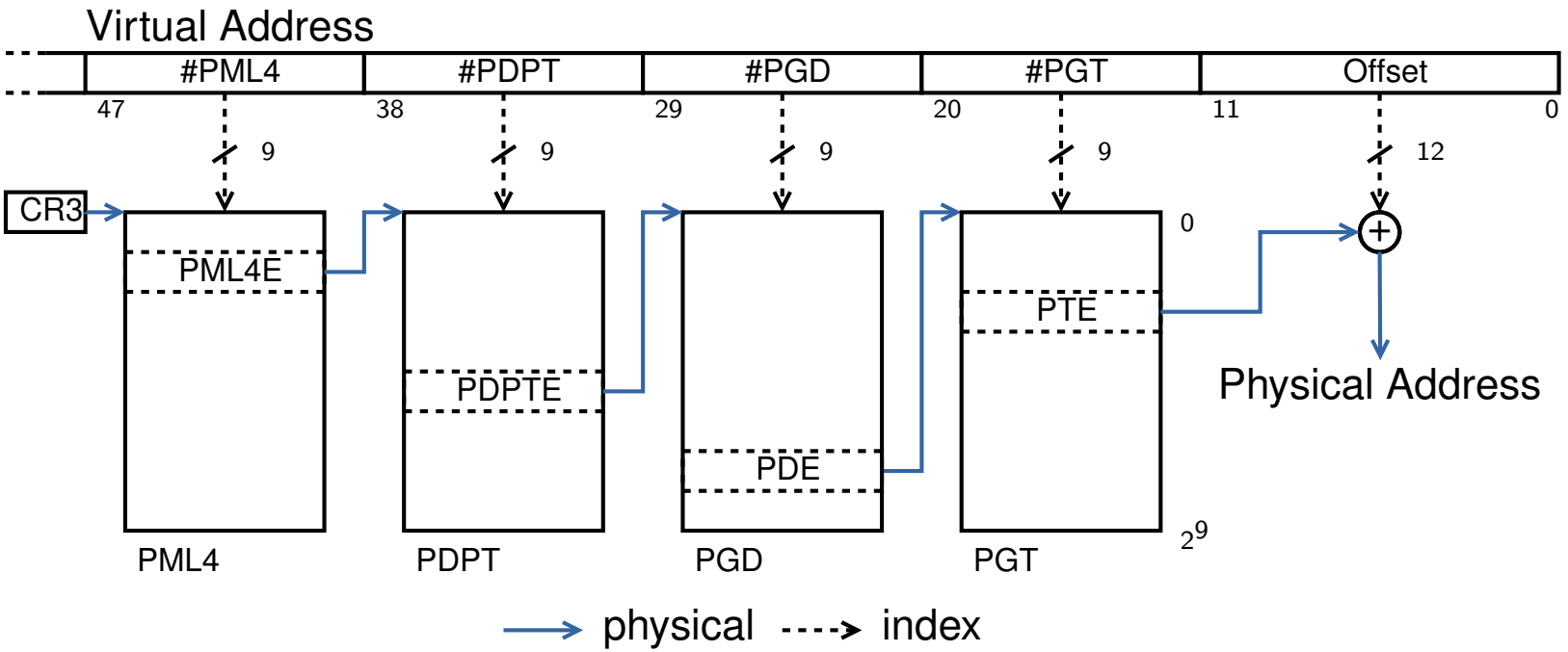


Figure: Paging in Longmode.

Self-mapped Page Table Walk

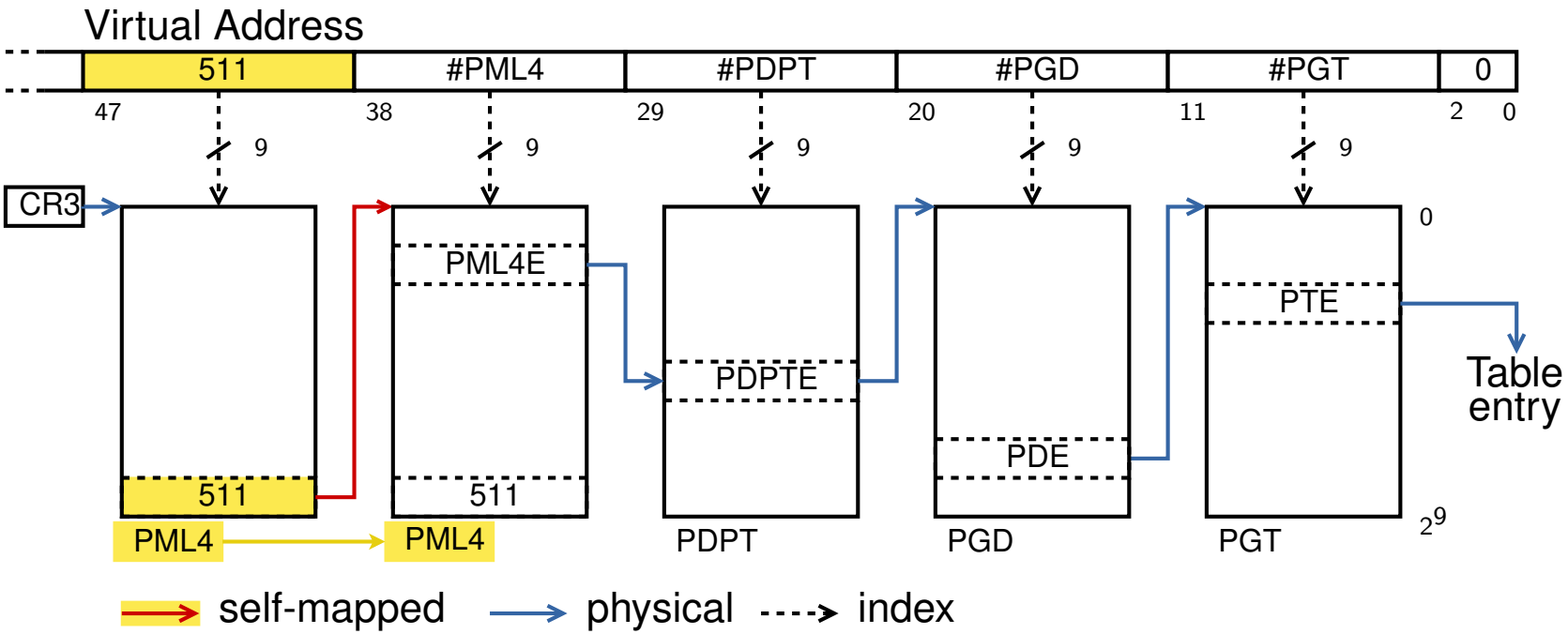


Figure: Self-mapped Page Tables (PGTs).

Self-mapped Page Table Walk

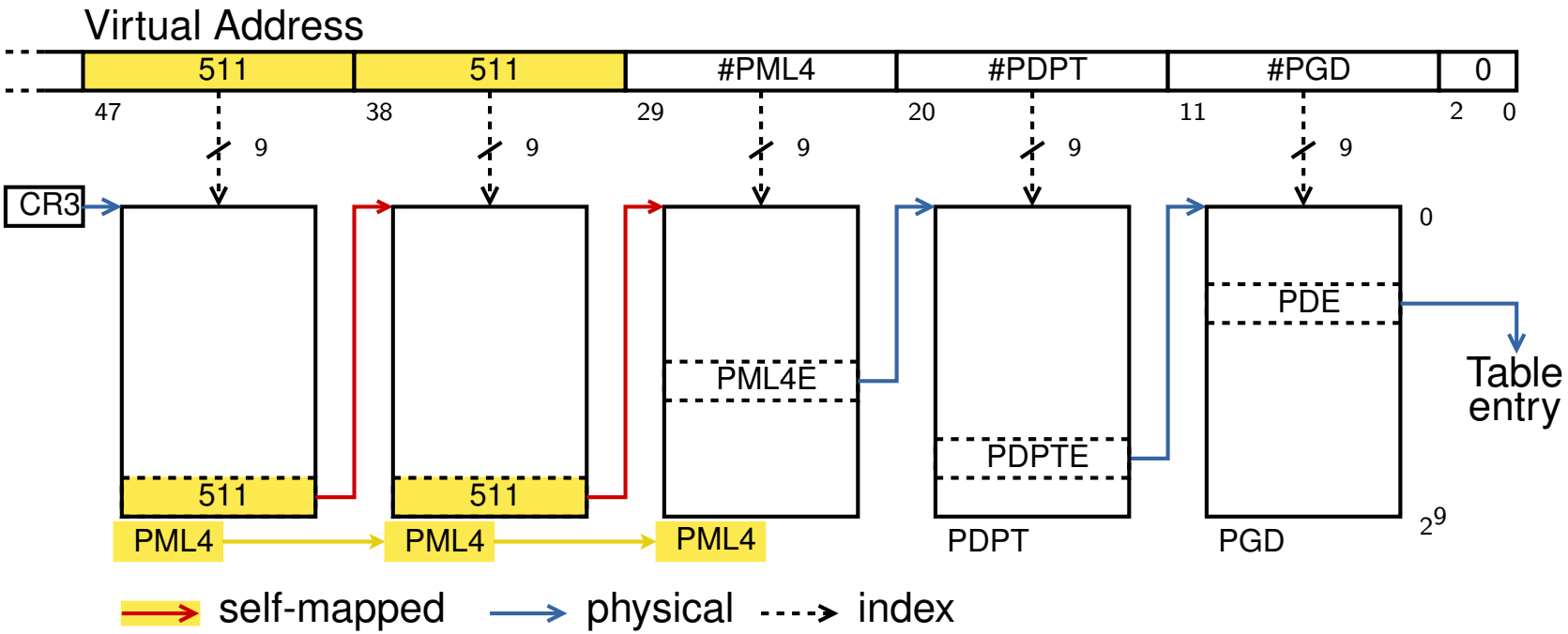


Figure: Self-mapped Page Directorys (PGDs).

Self-mapped Page Table Walk

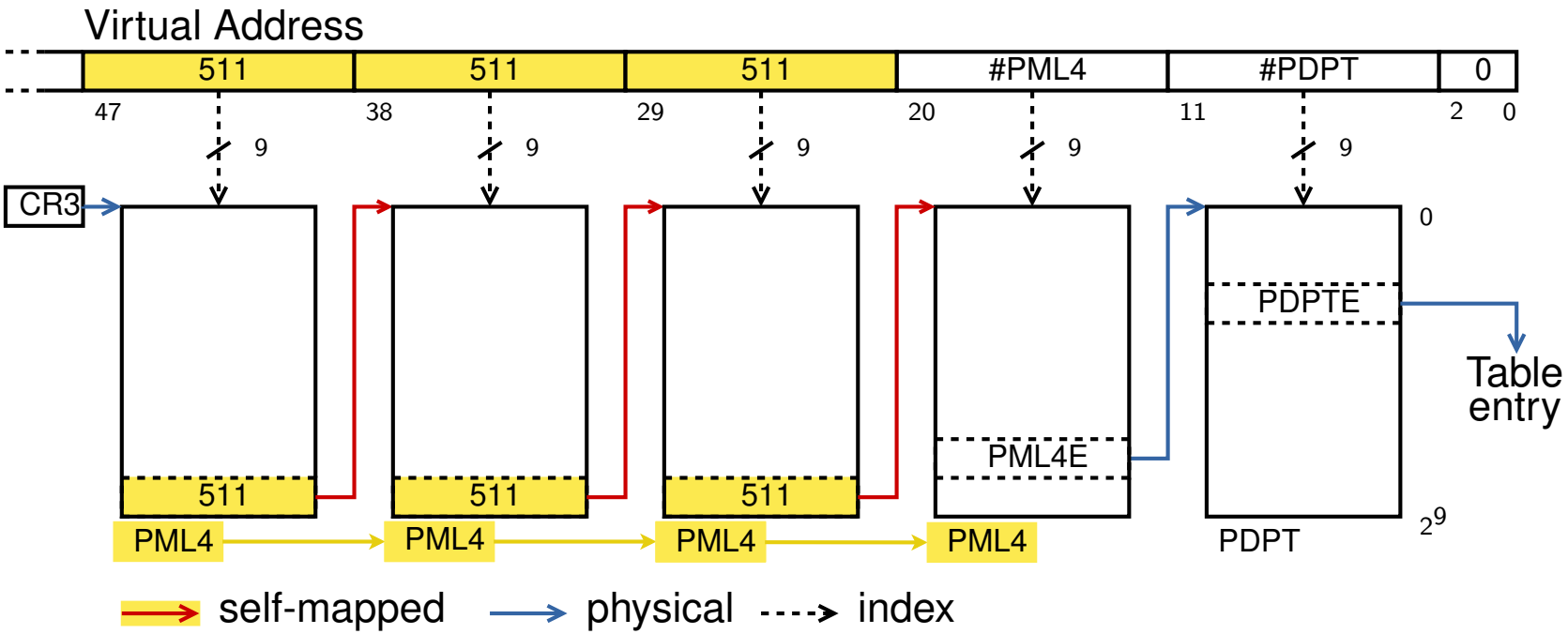


Figure: Self-mapped Page Directory Pointer Tables (PDPTs).

Self-mapped Page Table Walk

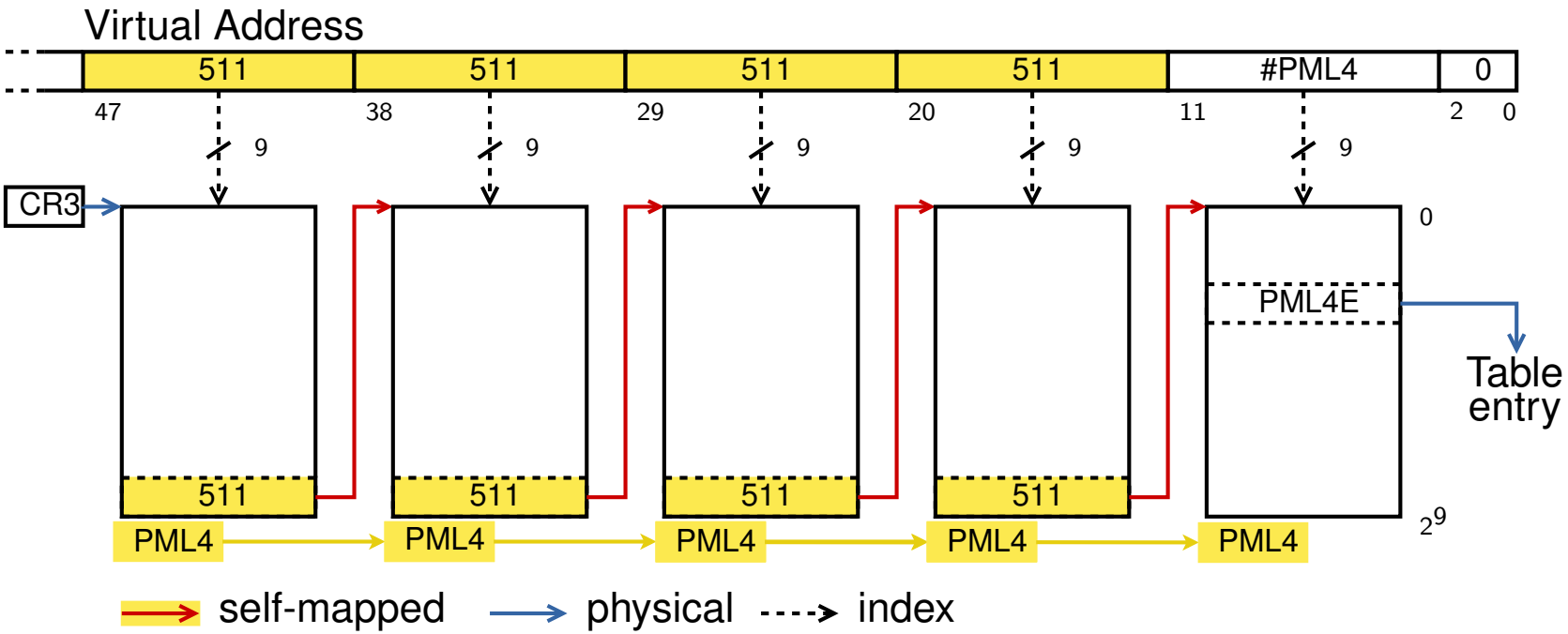


Figure: Self-mapped Page Map Level 4 (PML4).

Table Base Addresses

All page tables (including PGD .. PML4) are accessible by using the following addresses:

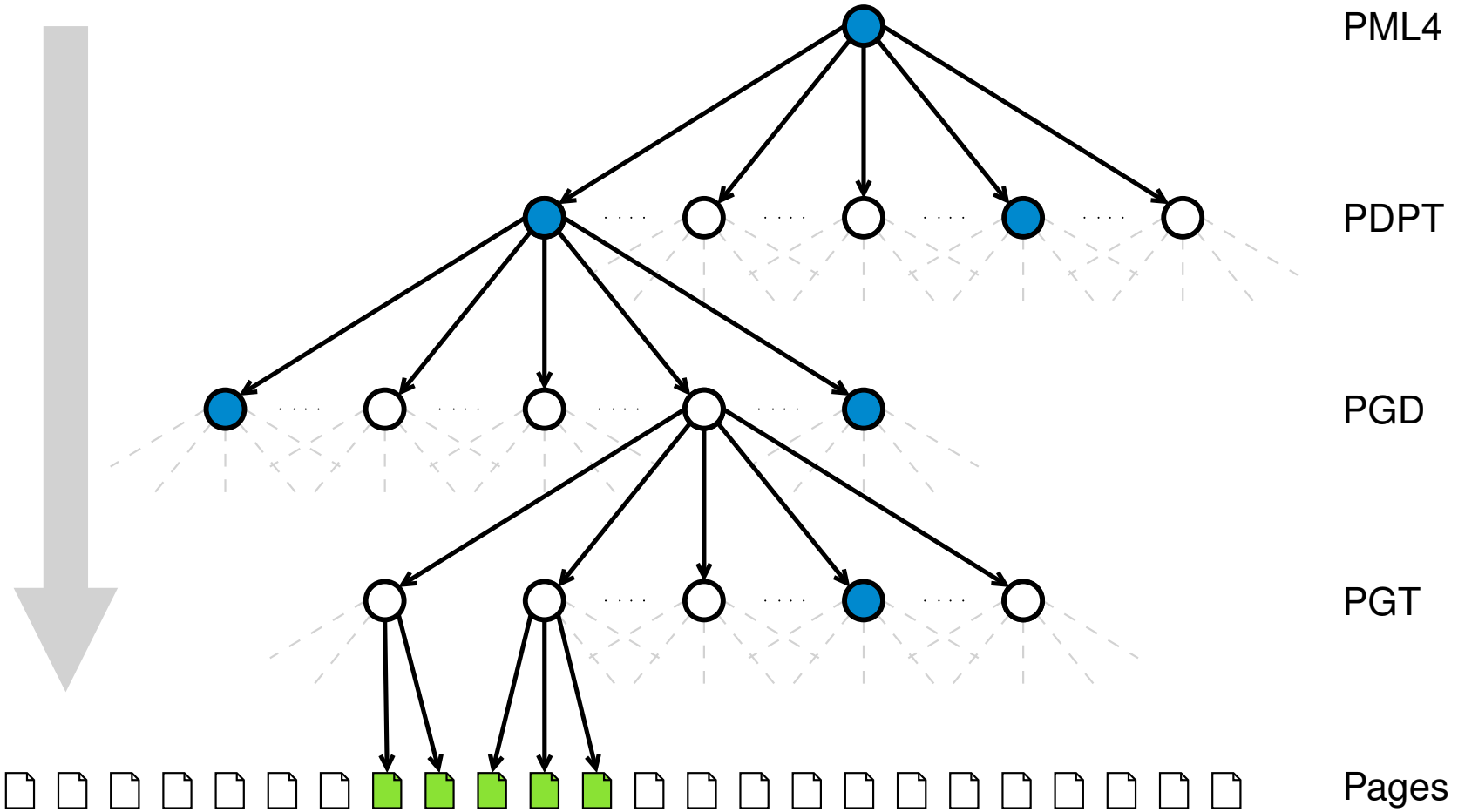
Table	Address
PGT	0xFFFFFF8000000000
PGD	0xFFFFFFFFC0000000
PDPT	0xFFFFFFFFFFE00000
PML4	0xFFFFFFFFFFFFF000

This example the last (512th) entry for self-referencing.
All other entries could also be used.

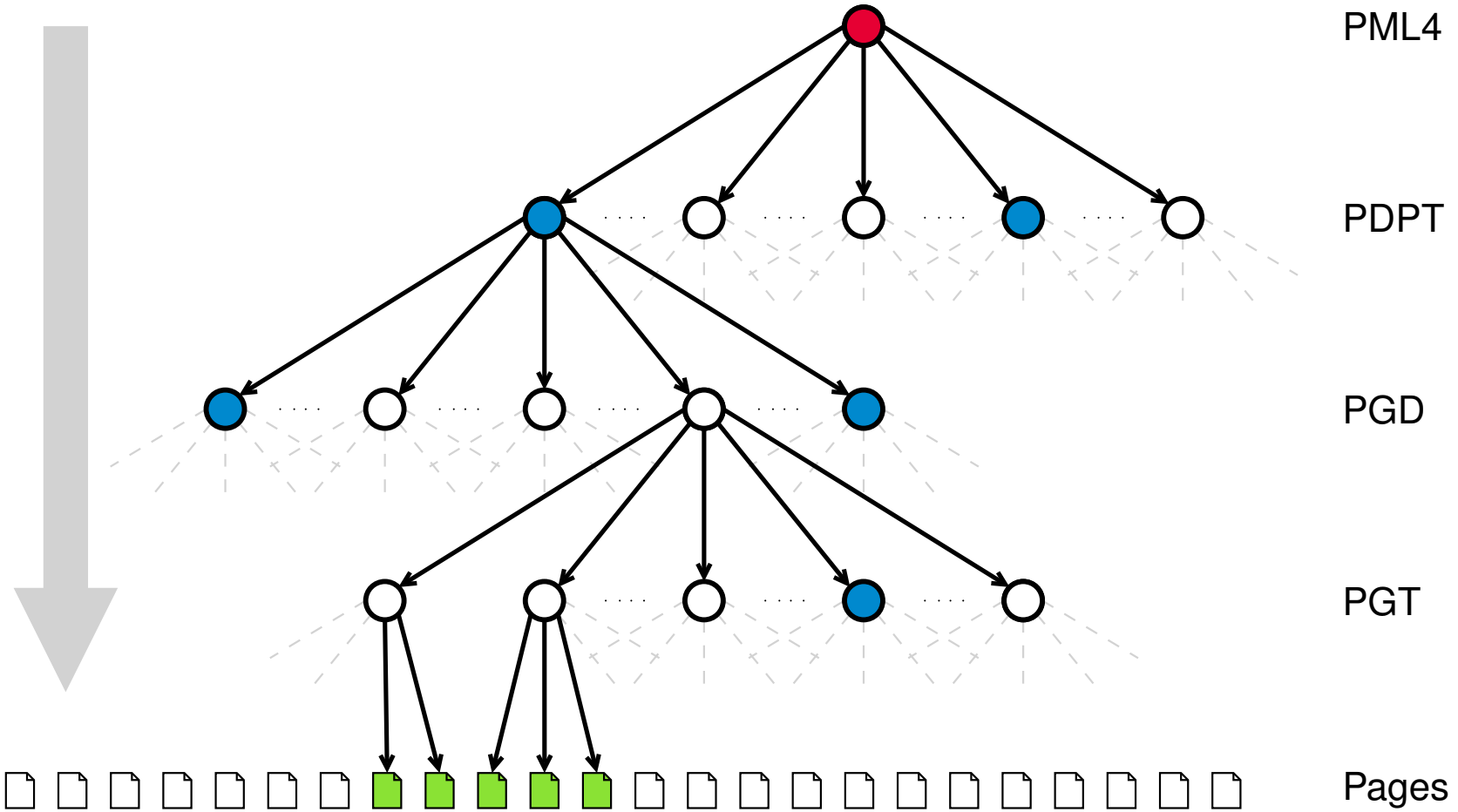
Traversal: Top-Down

- Multi-level page tables constitute a search tree
  - Root: PML4 table
  - Leaves: PGT's
- Using known tree traversals (pre, post, in-order)
  - Start at the root node
  - Descend to the PGT's
- Using recursive function invocations per table/node
- Different operations require diffrent traversals

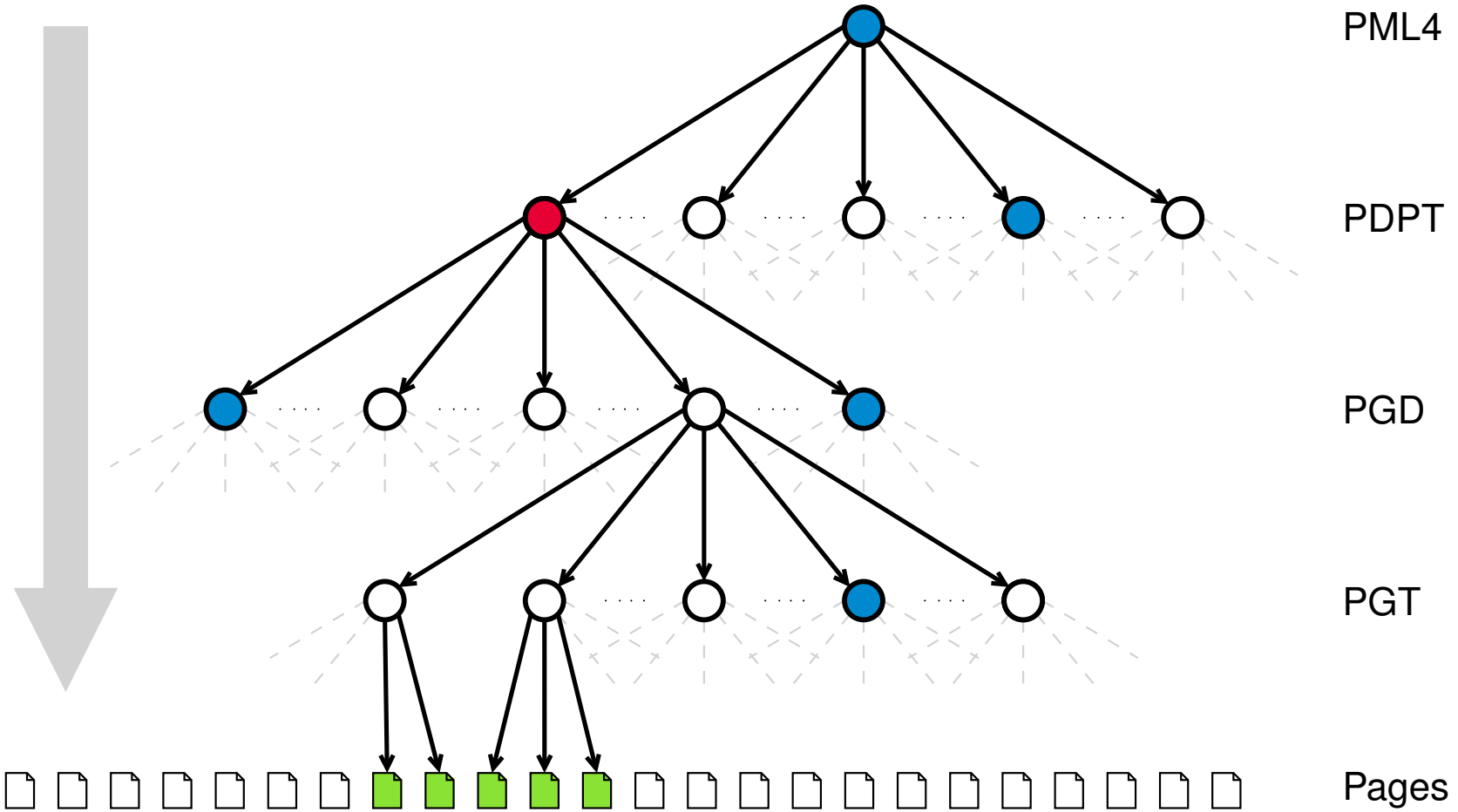
Tree Traversals



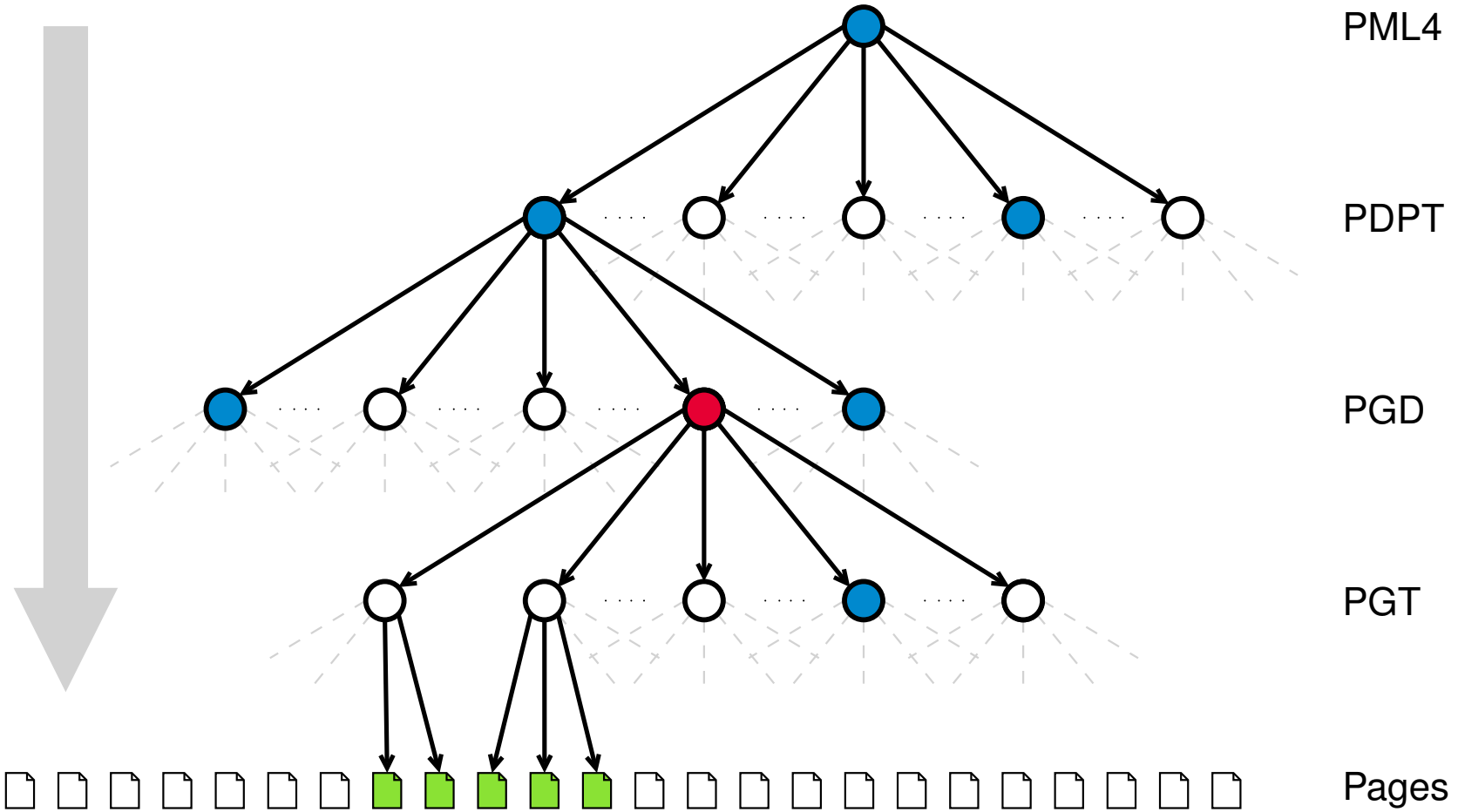
Tree Traversals



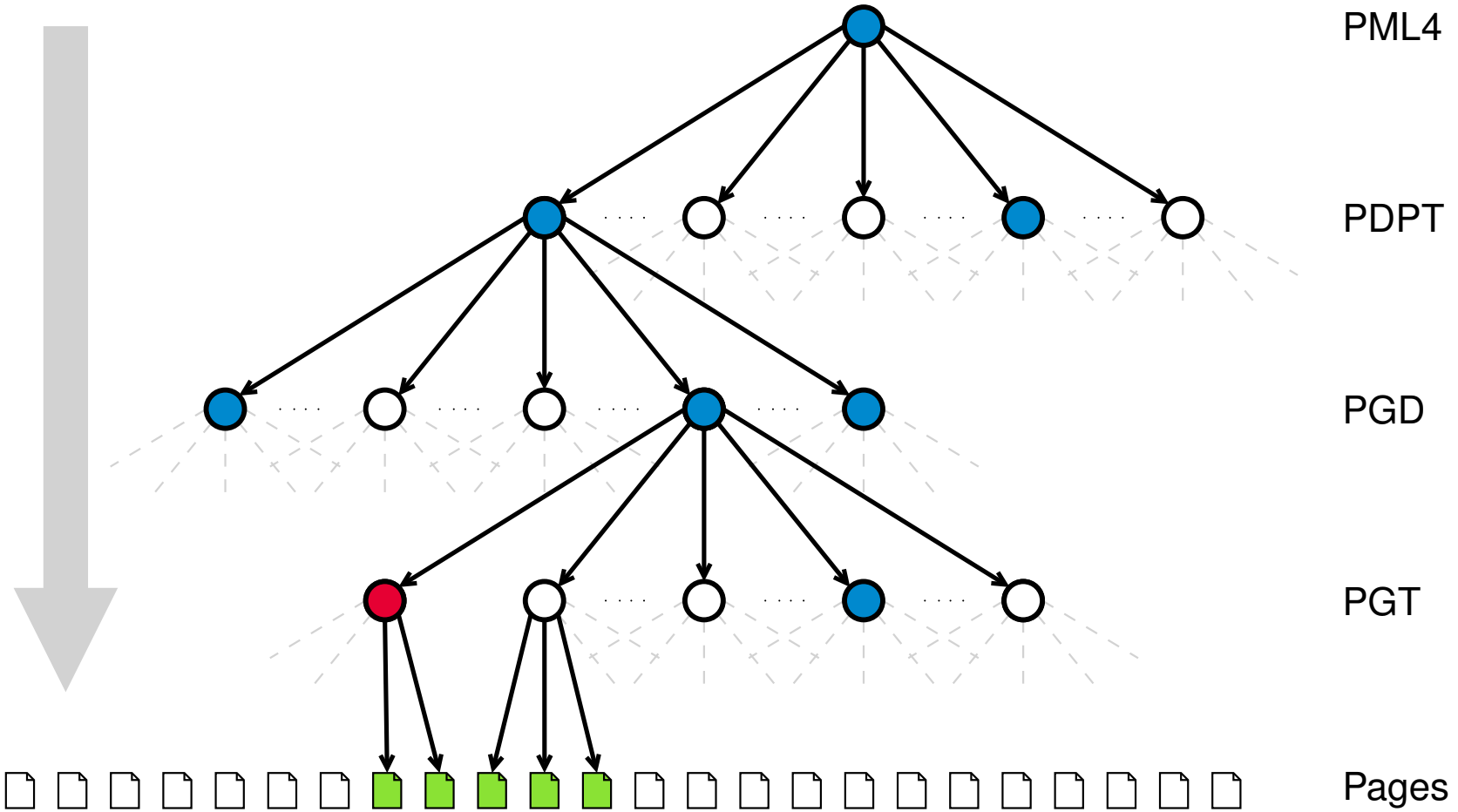
Tree Traversals



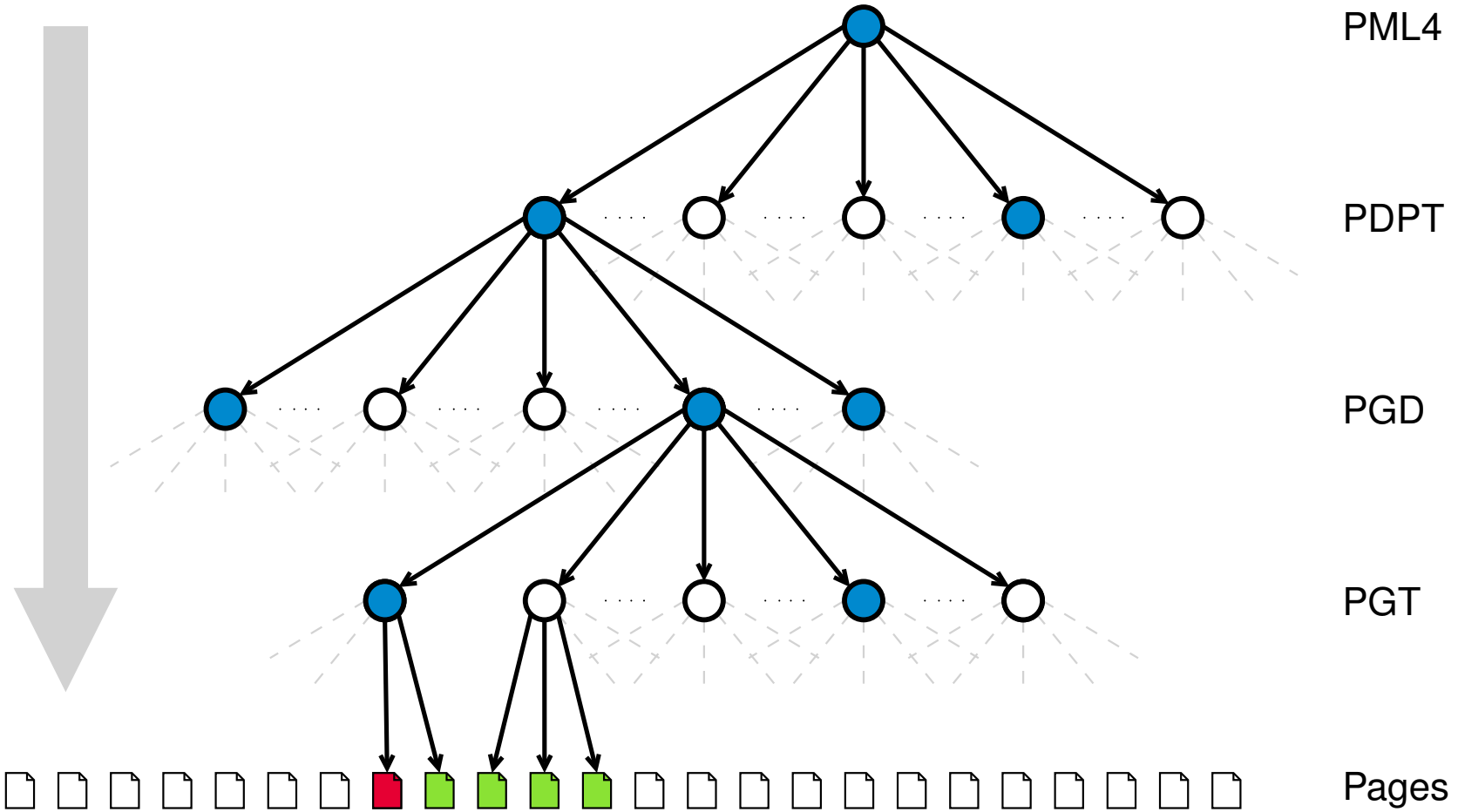
Tree Traversals



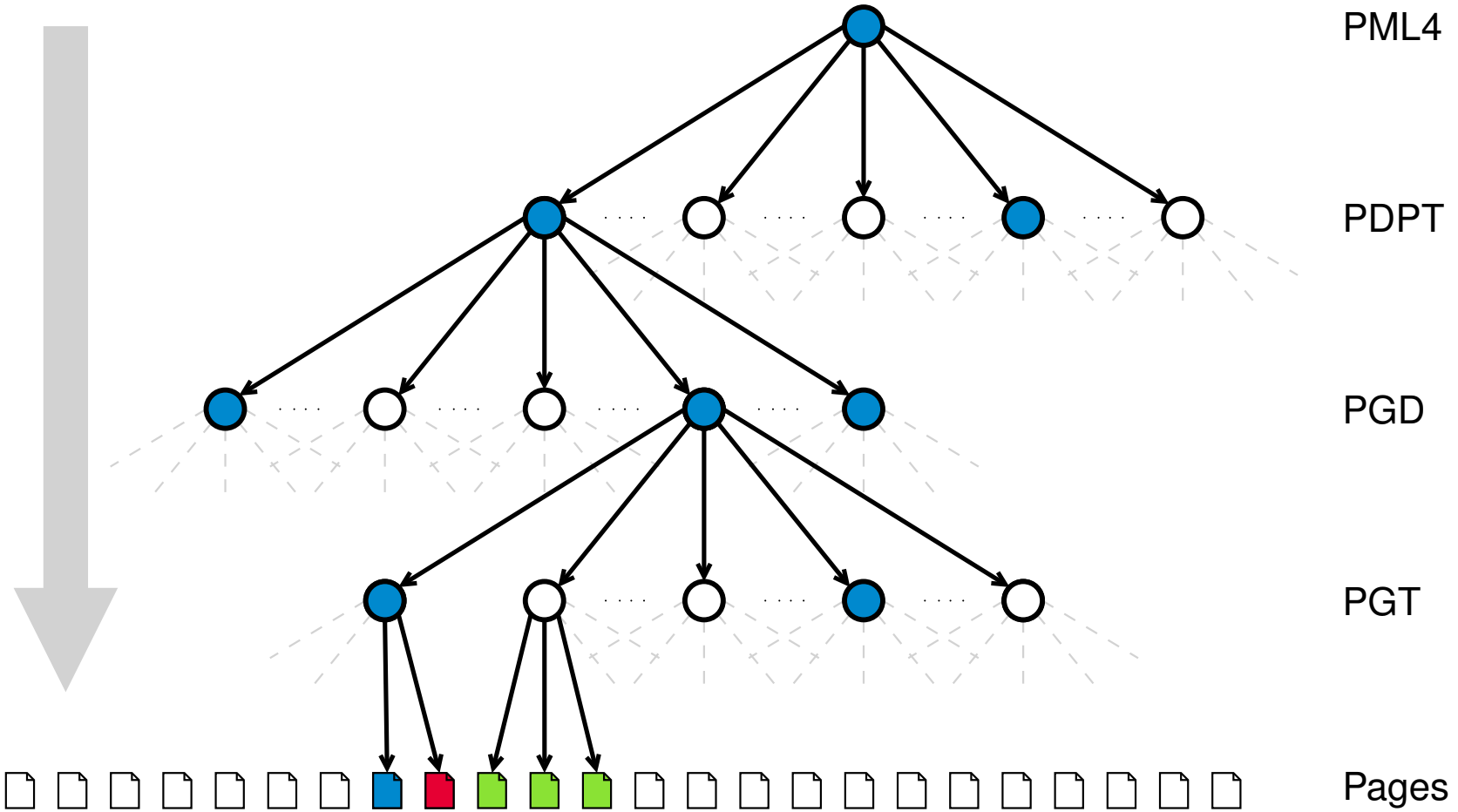
Tree Traversals



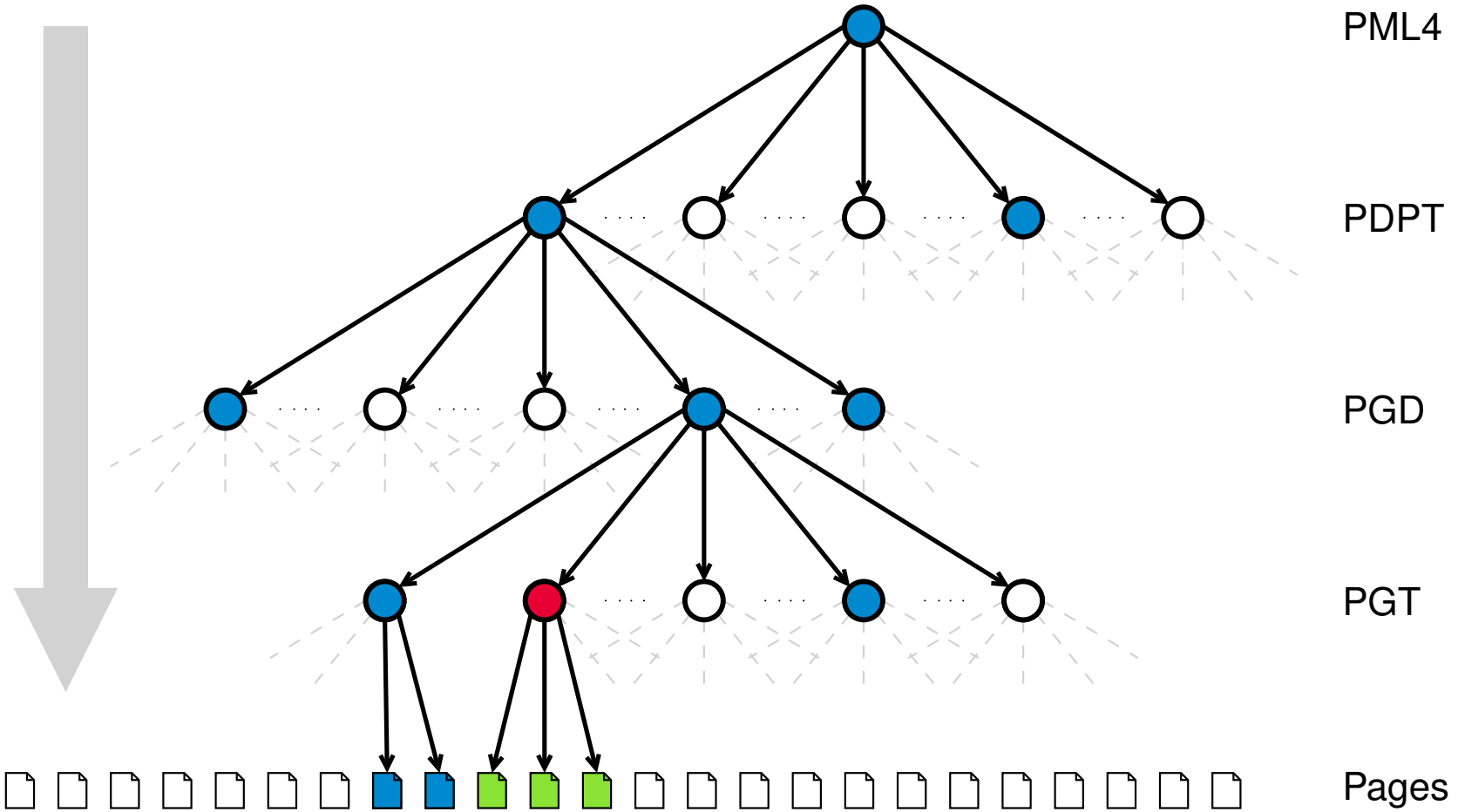
Tree Traversals



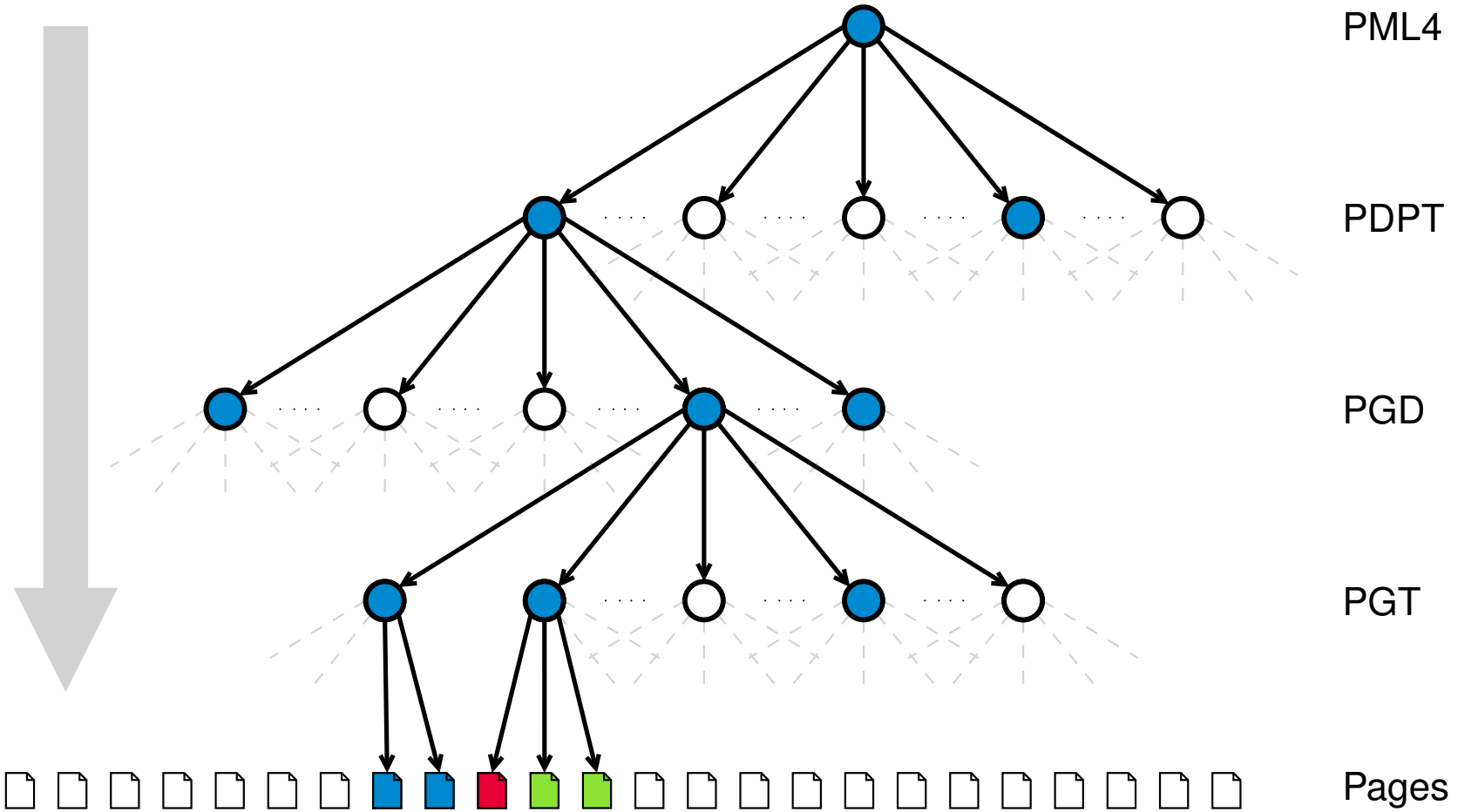
Tree Traversals



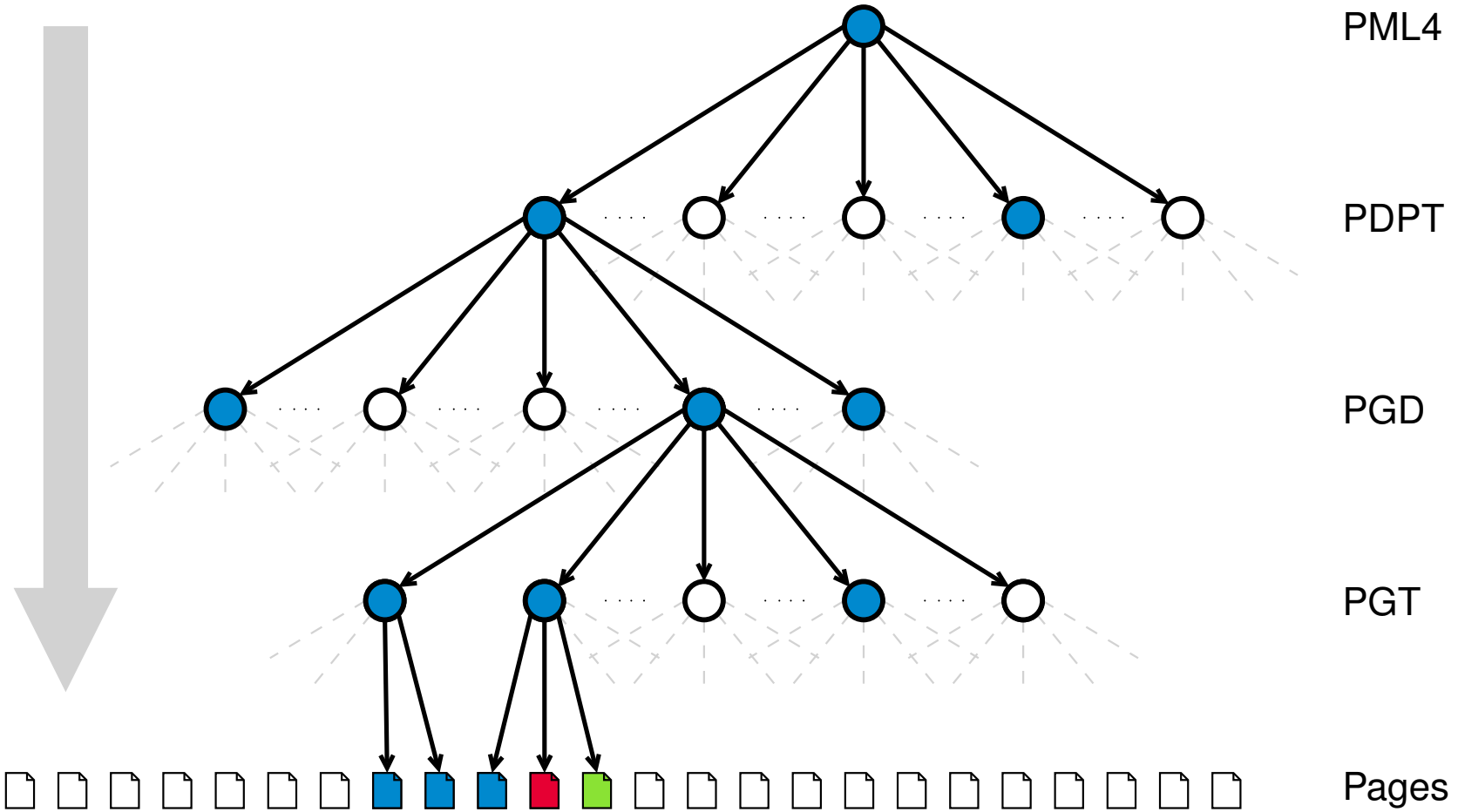
Tree Traversals



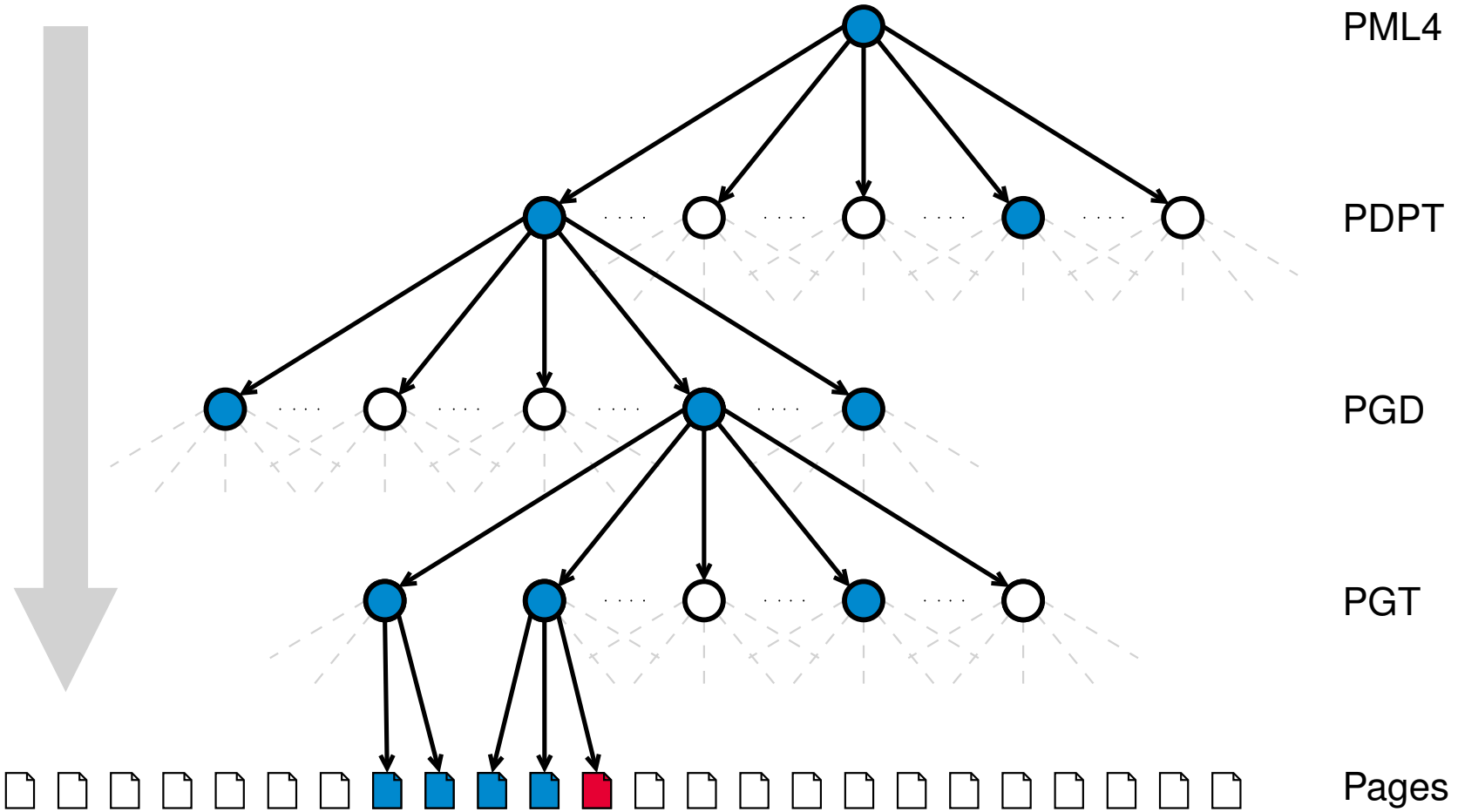
Tree Traversals



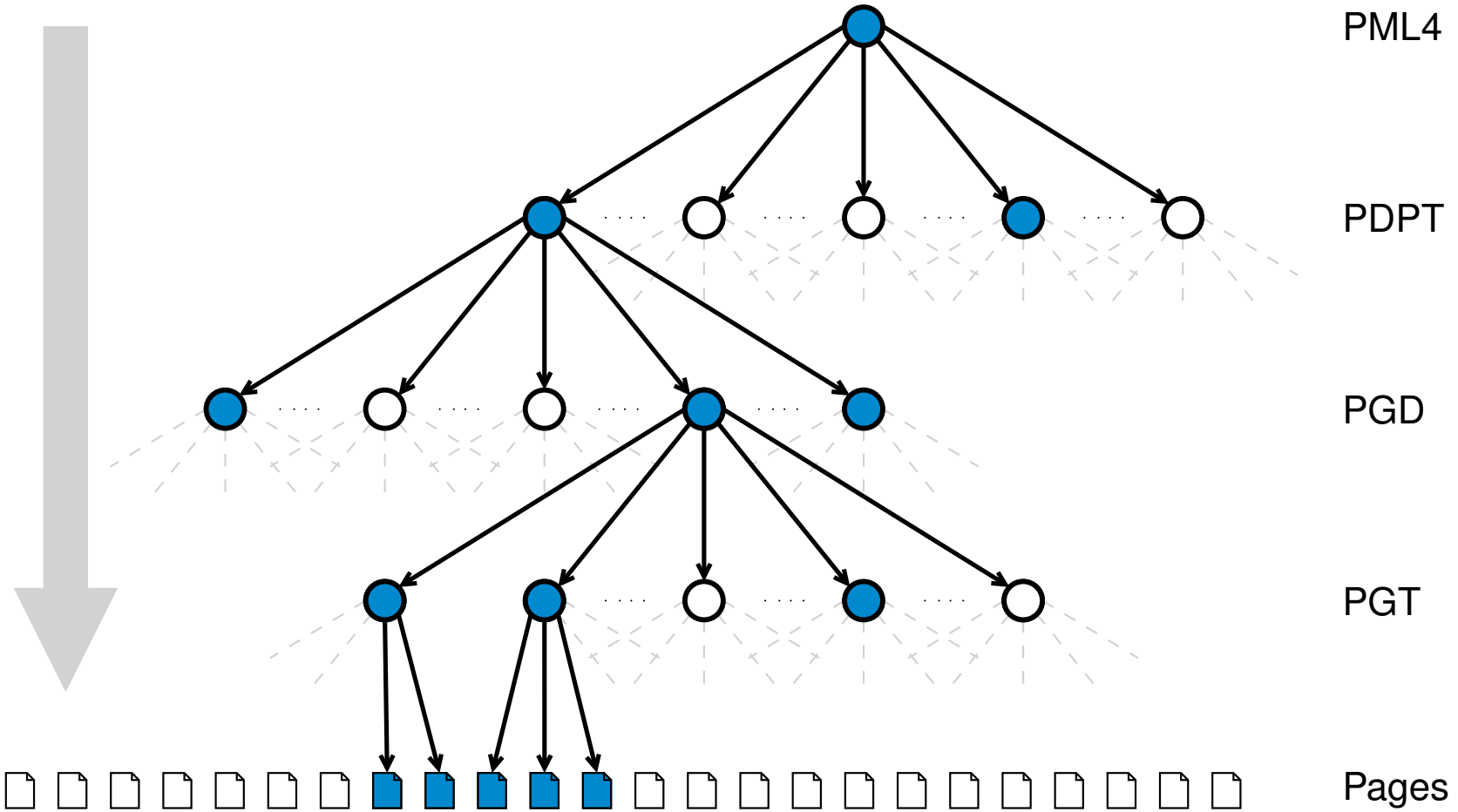
Tree Traversals



Tree Traversals



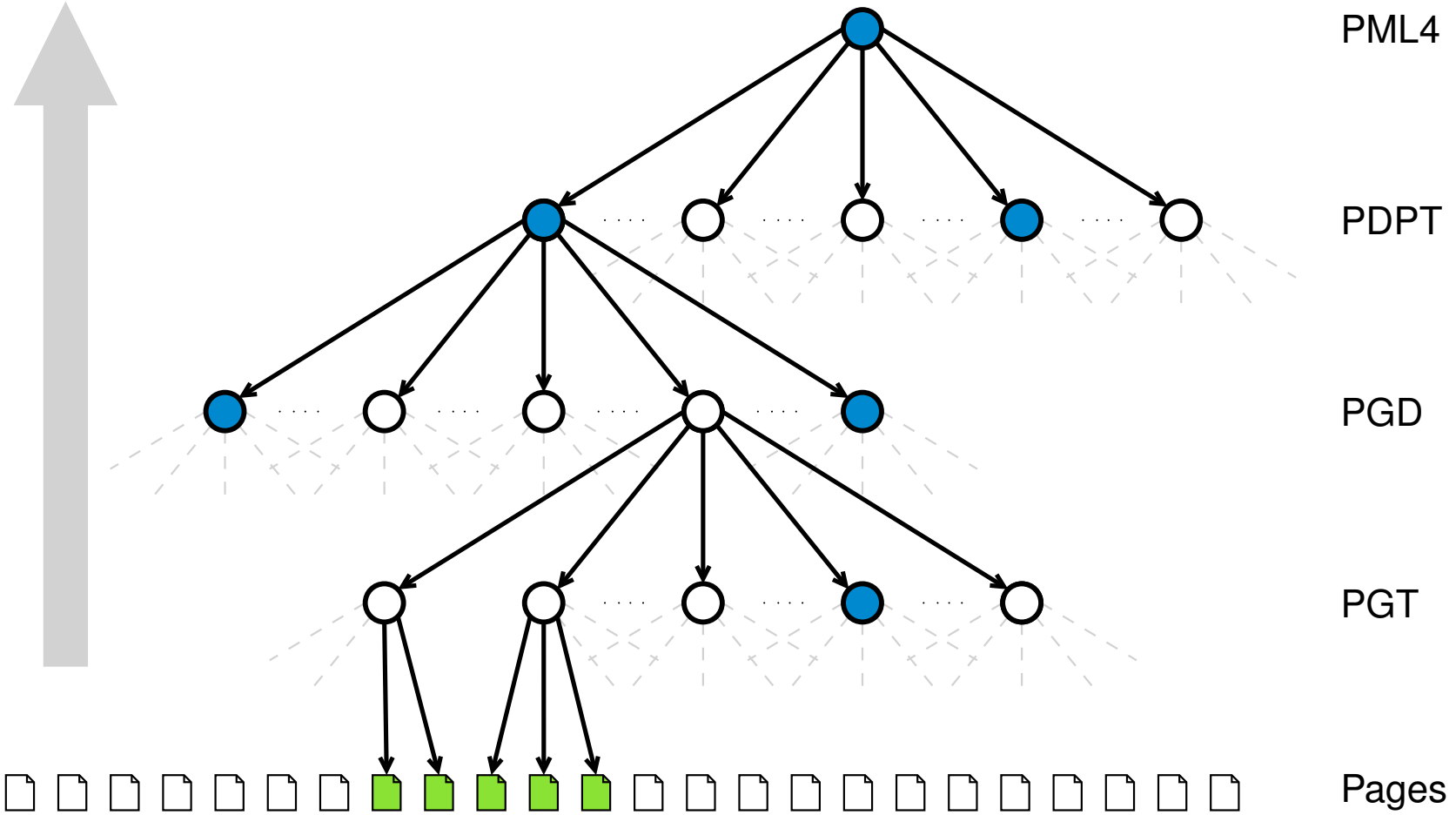
Tree Traversals



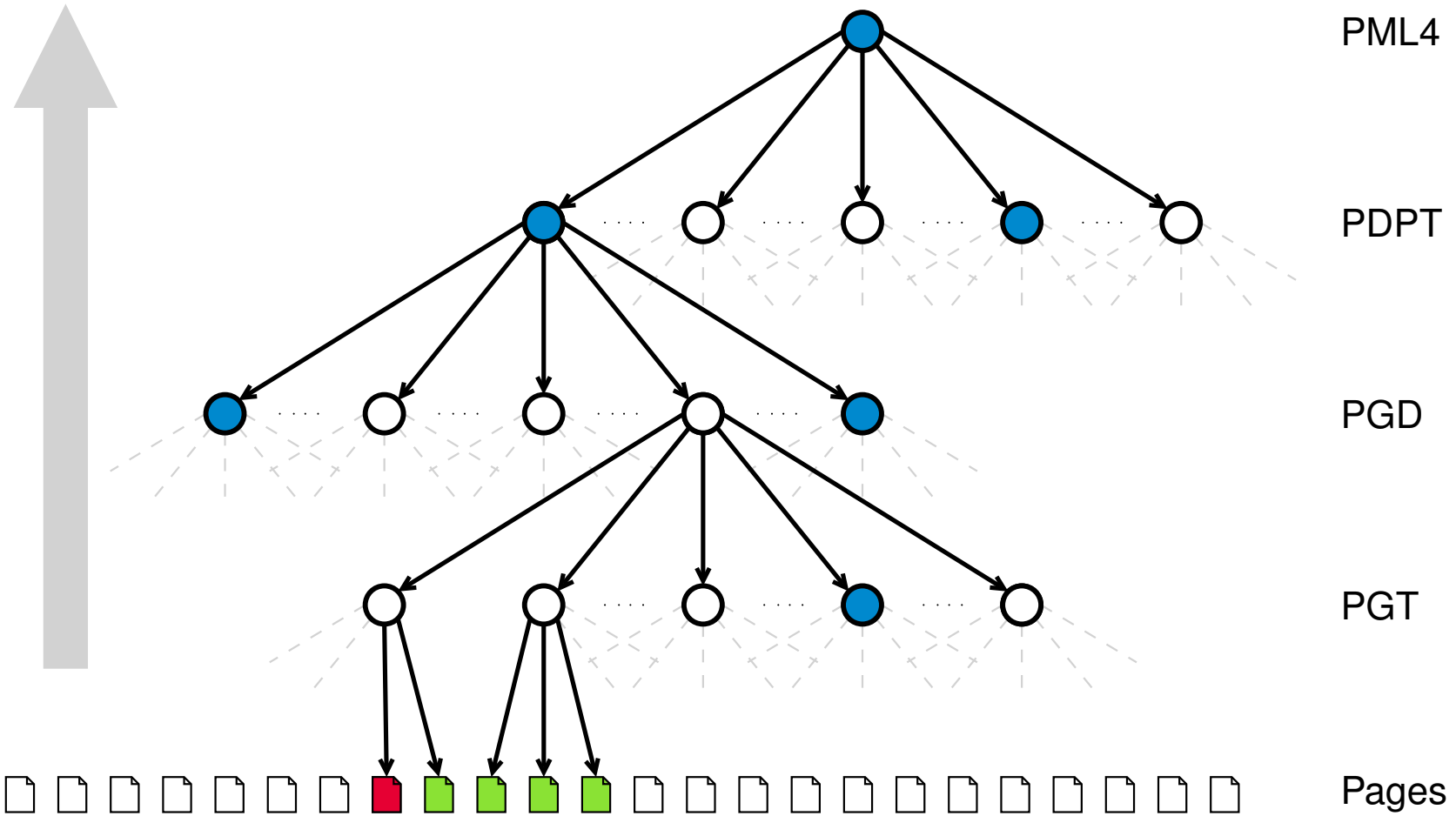
Traversal: Bottom-Up

- Start in the lowest-order table (PGT)
- Update superior table
  - Updates on missing tables will cause a page-fault
- Use page-fault handler to create tables on-the-fly
  - Nested page-faults might occur

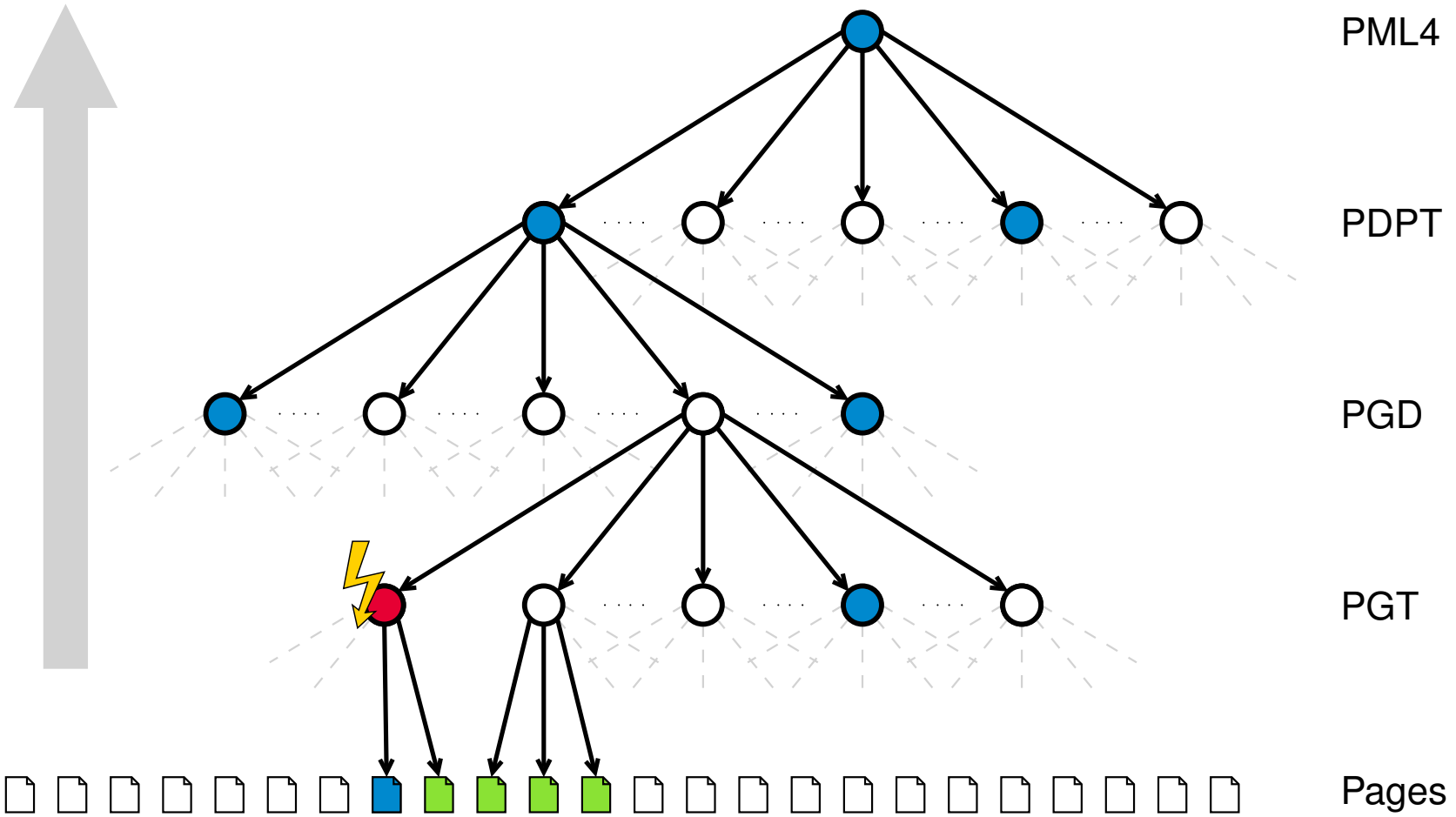
Pagefaults



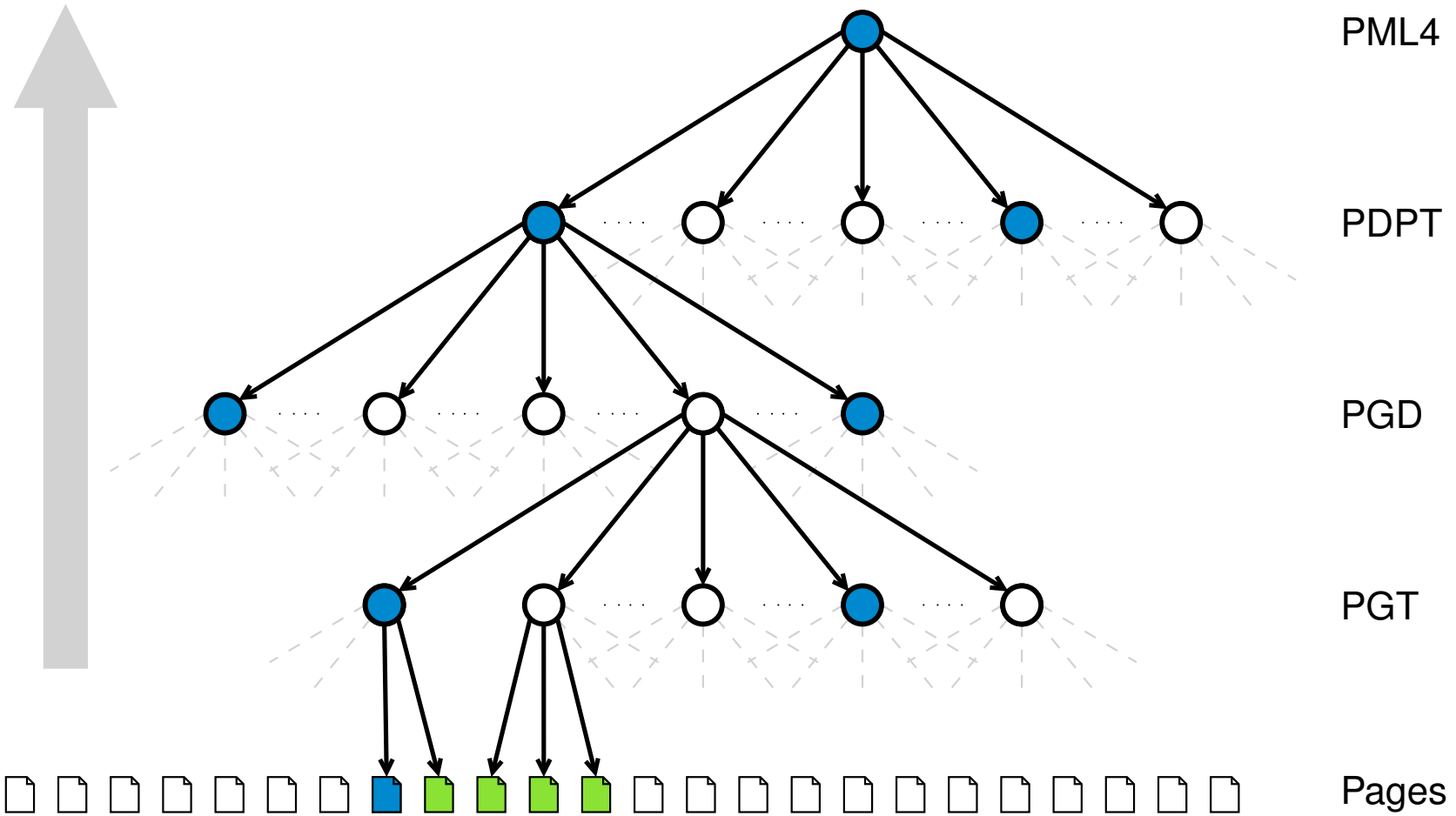
Pagefaults



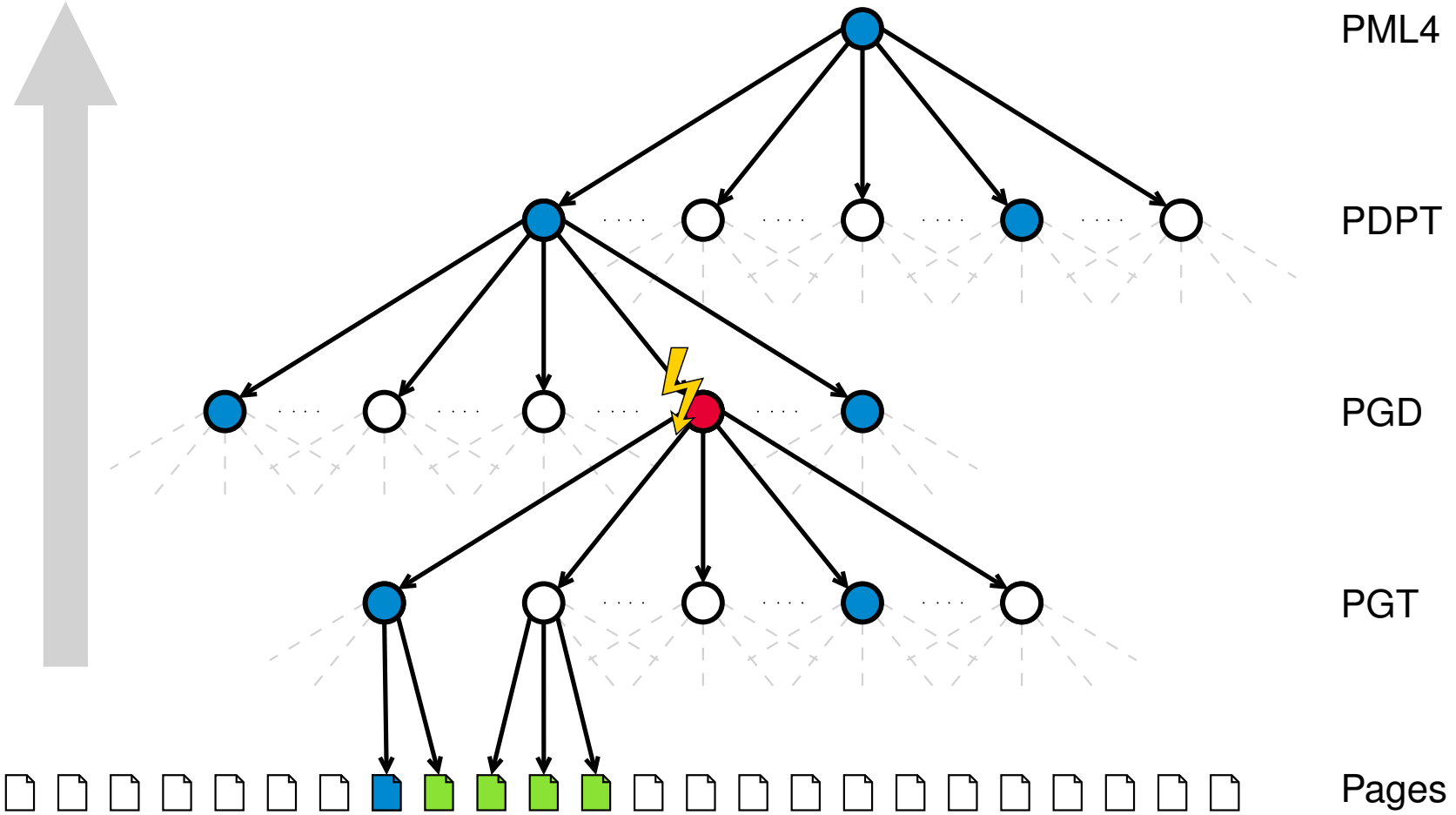
Pagefaults



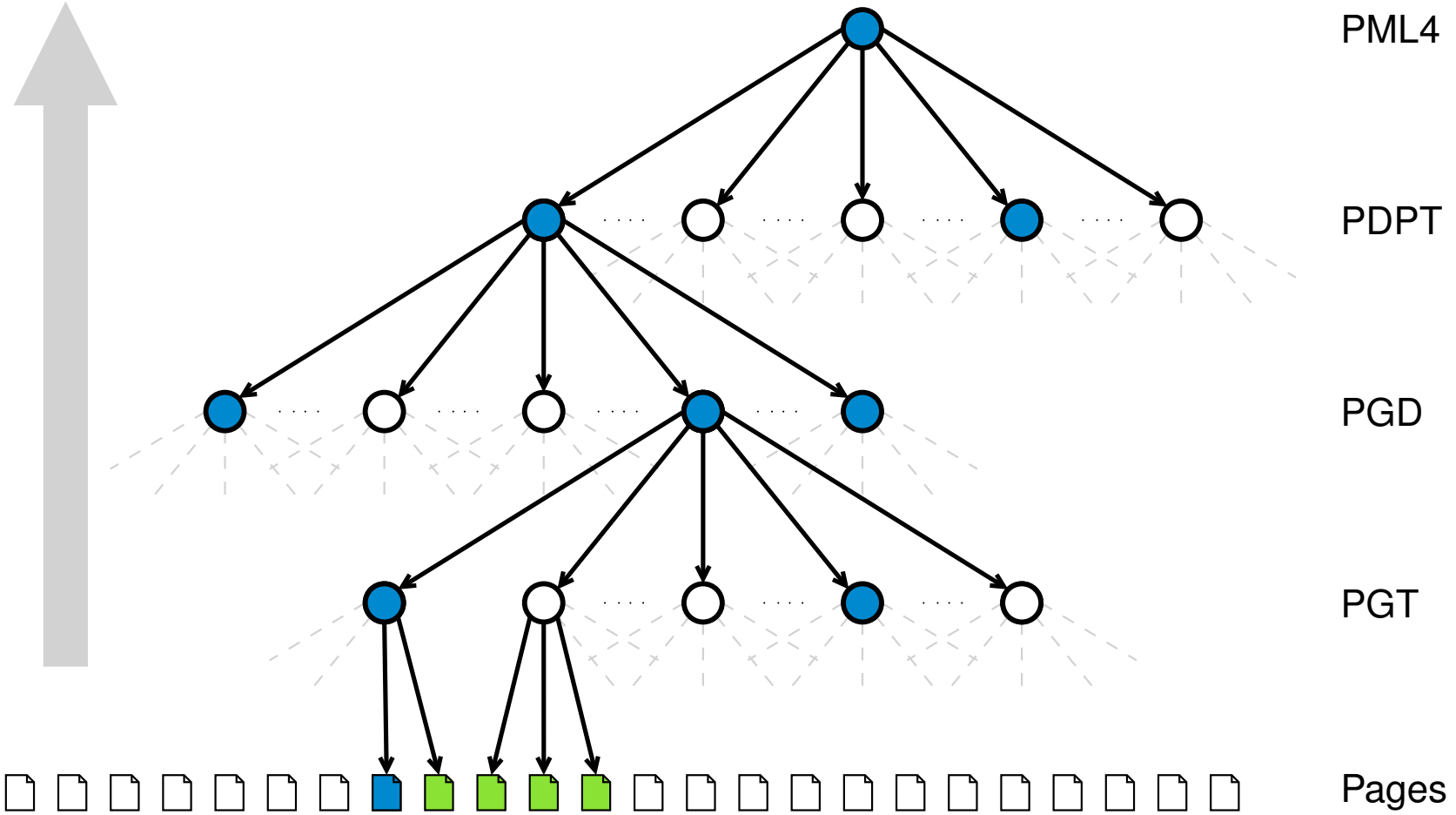
Pagefaults



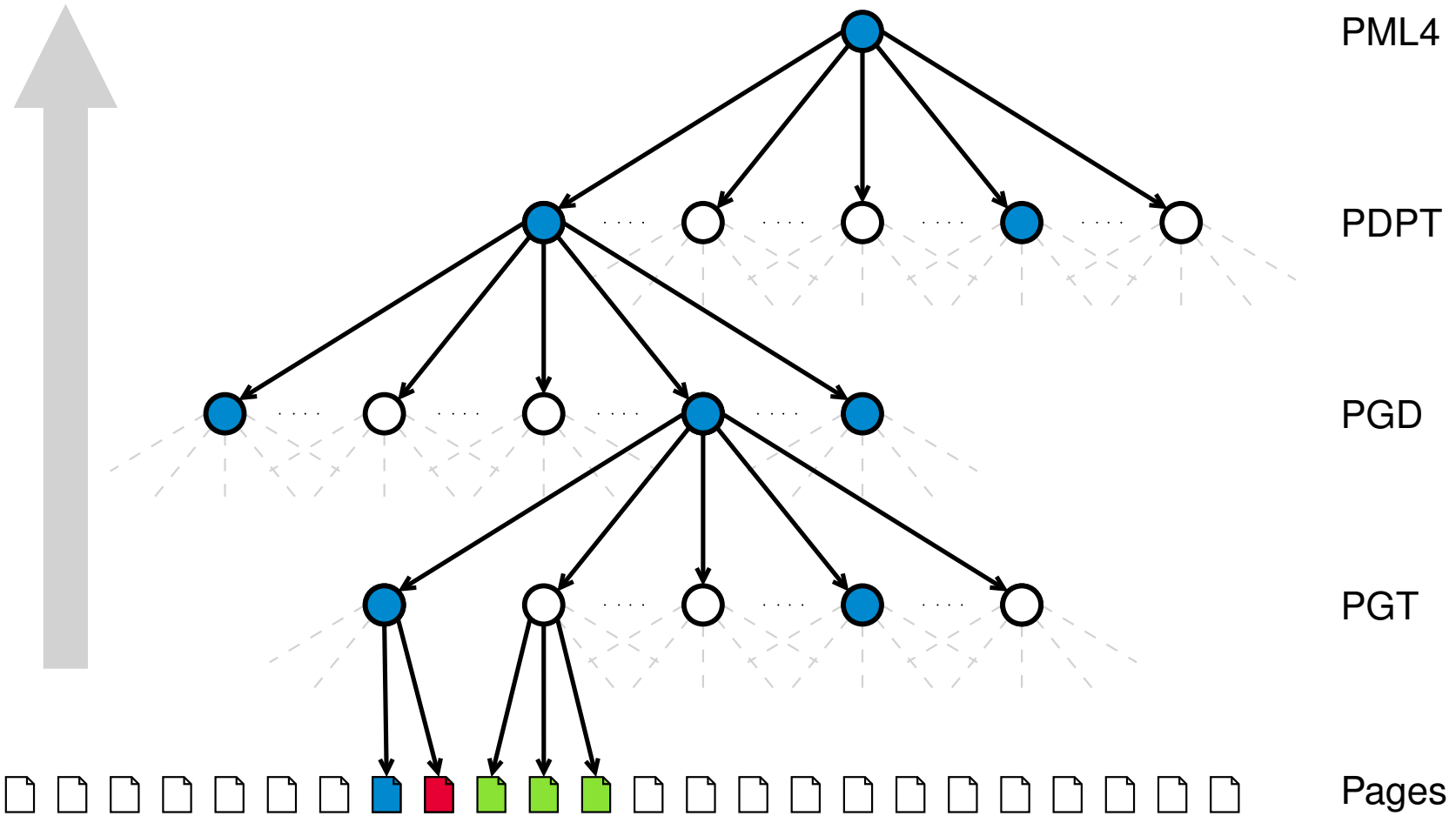
Pagefaults



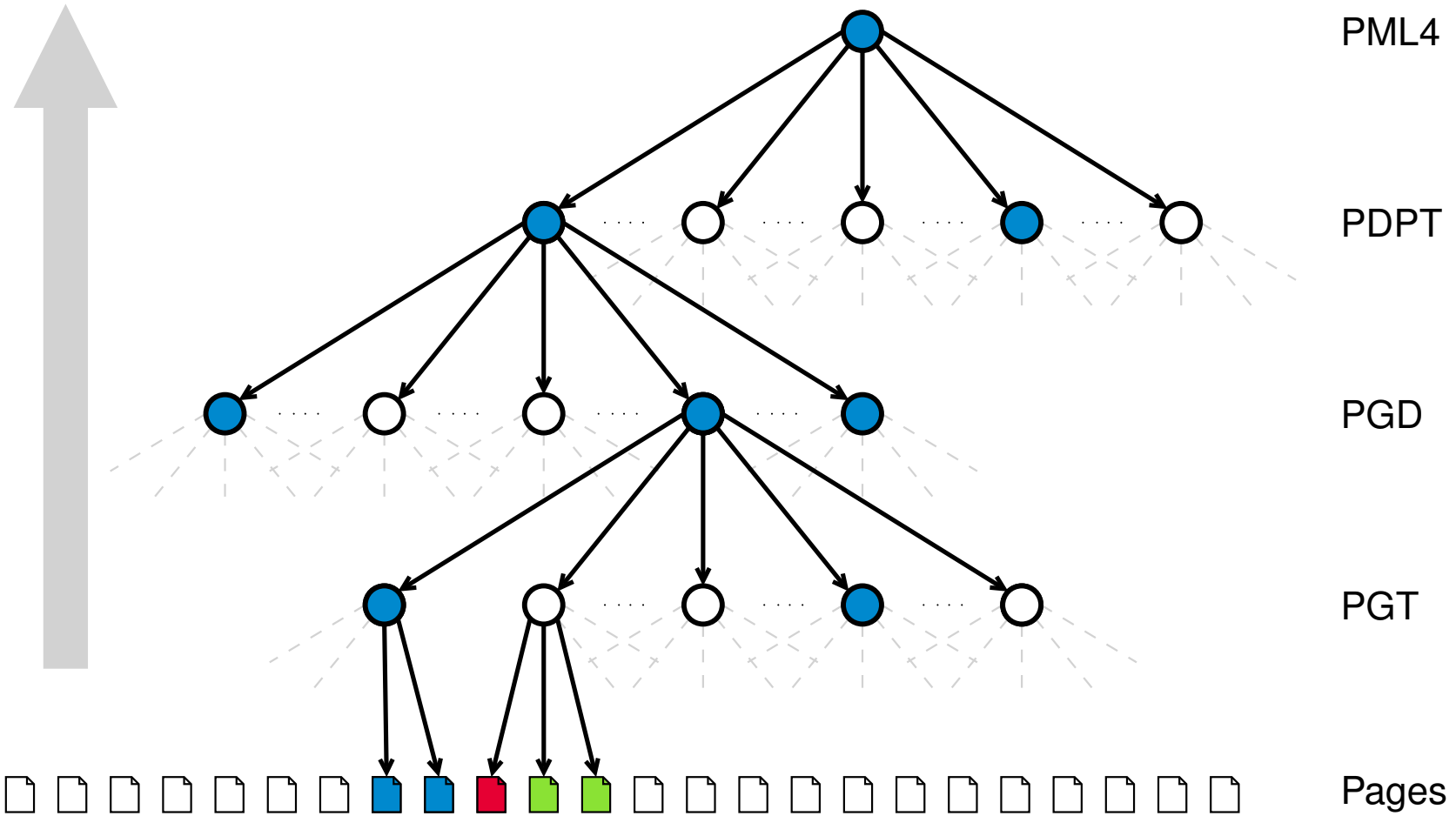
Pagefaults



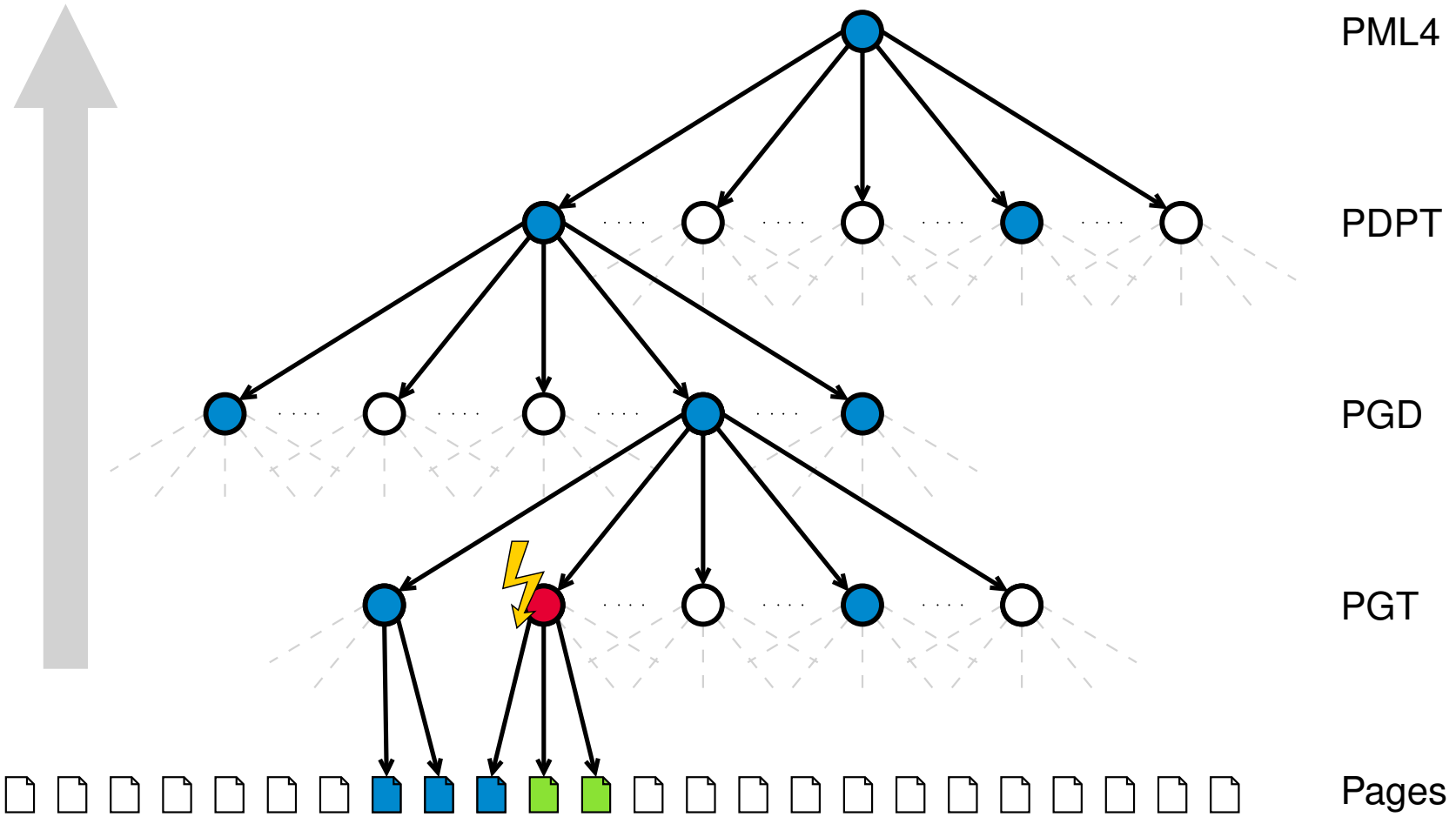
Pagefaults



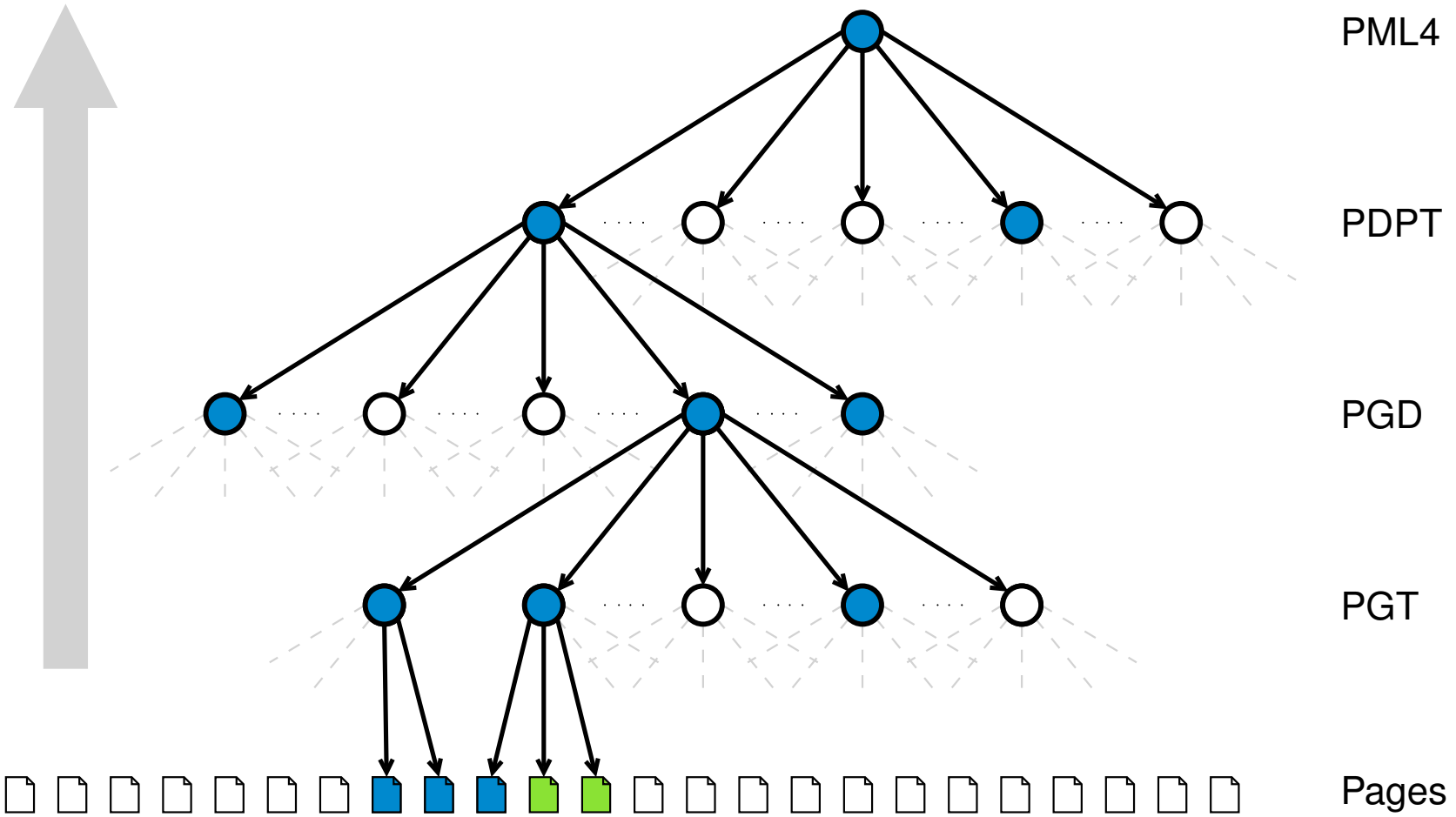
Pagefaults



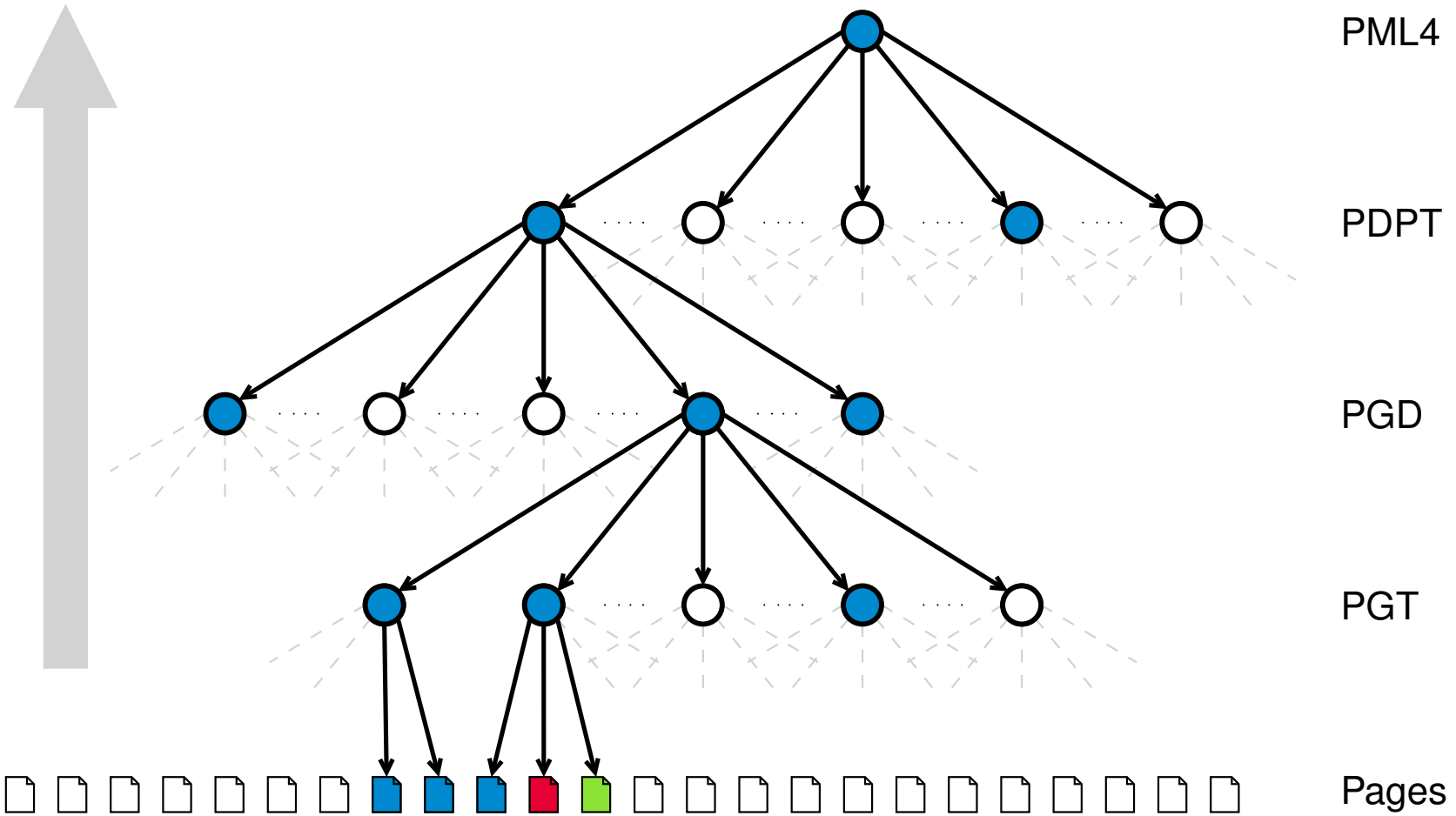
Pagefaults



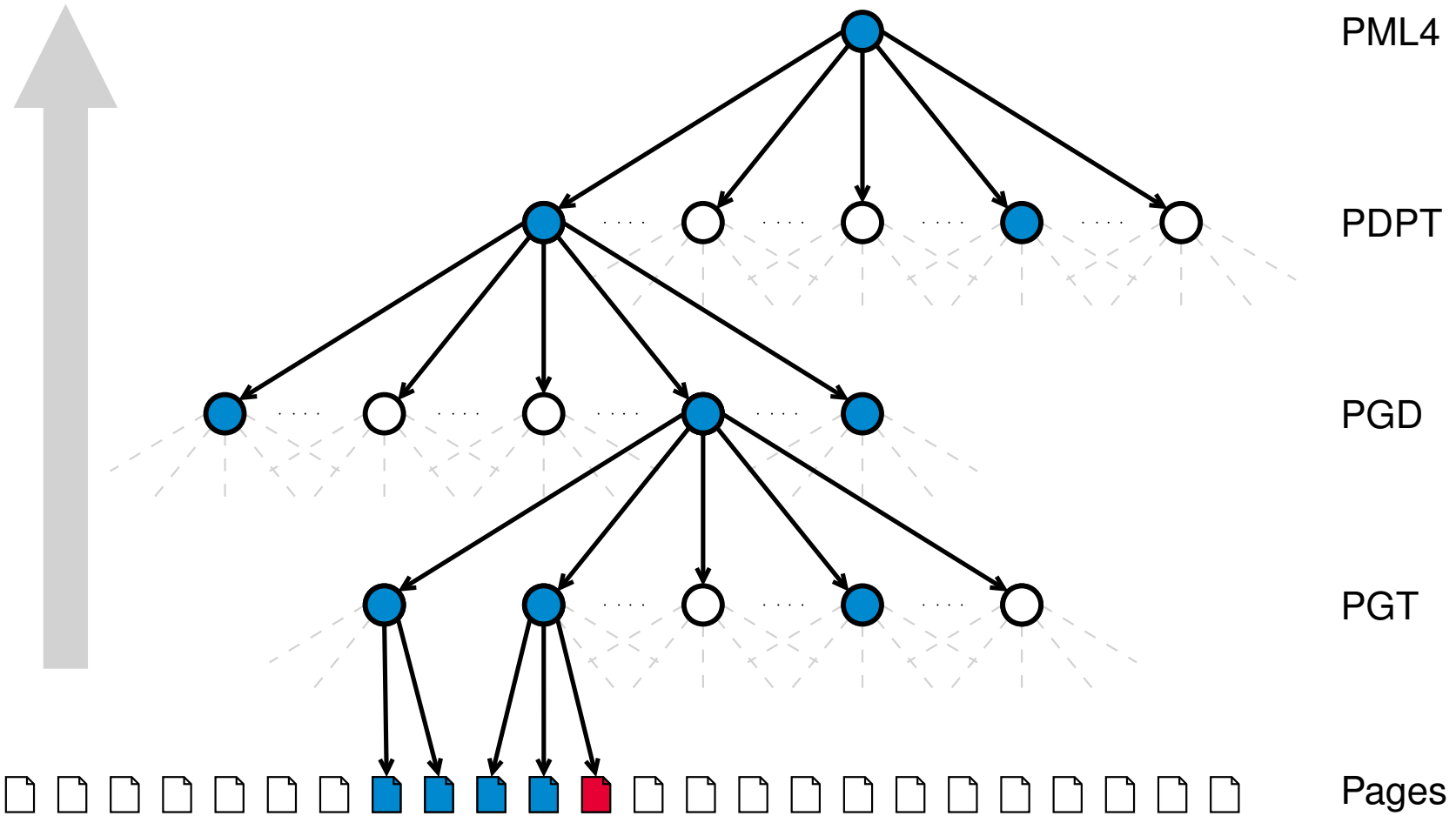
Pagefaults



Pagefaults



Pagefaults



Who else?

Not so many...

- Hobby OSes[6]
- Microsoft's NT kernel [3]?
- Mentioned by the Alpha Architecture reference manual [2]

Why is it not widely used?

- Not documented in *official* Intel and AMD manuals
- Not as portable as expected
  - Only some of the Linux's architectures support it

[6] `http://wiki.osdev.org/Page_Tables`.

Conclusion

- Rewrite of paging subsystem with more features:
  - Complete support of 64 bit user space applications
  - Release of unused page tables
  - Partial support for huge pages[7]
  - Execute-disable flag[8]
- Additional operations:
  - Print page dump
  - Collect statistics (accessed / dirty)
  - ... and more can easily be implemented

[7] Larger page sizes by truncating the table walk.

[8] Disable code execution in certain memory regions.

Conclusion

- Rewrite of paging subsystem with more features:
  - Complete support of 64 bit user space applications
  - Release of unused page tables
  - Partial support for huge pages[7]
  - Execute-disable flag[8]
- Additional operations:
  - Print page dump
  - Collect statistics (accessed / dirty)
  - ... and more can easily be implemented

[7] Larger page sizes by truncating the table walk.

[8] Disable code execution in certain memory regions.

Conclusion

- Rewrite of paging subsystem with more features:
  - Complete support of 64 bit user space applications
  - Release of unused page tables
  - Partial support for huge pages[7]
  - Execute-disable flag[8]
- Additional operations:
  - Print page dump
  - Collect statistics (accessed / dirty)
  - ... and more can easily be implemented

[7] Larger page sizes by truncating the table walk.

[8] Disable code execution in certain memory regions.

Conclusion

- Self-mapped approach isn't as generic as expected
- Comparison to Linux is difficult:
  - Different `malloc()` strategies (glibc vs newlib)
  - MetalSVM has larger overhead for rising a page fault
    - Linux is fast for mapping single pages
      but slower for mapping large regions.
- Smaller and unified code base: easier maintainability
- Less macro hacking: improved readability of code

Conclusion

- Self-mapped approach isn't as generic as expected
- Comparison to Linux is difficult:
  - Different `malloc()` strategies (glibc vs newlib)
  - MetalSVM has larger overhead for rising a page fault
    - Linux is fast for mapping single pages
      but slower for mapping large regions.
- Smaller and unified code base: easier maintainability
- Less macro hacking: improved readability of code

What else?

- Virtual Memory Area (VMA) in Kernel Space
  - Previously: find free region by walking through the tables
    - Slow, architecture dependend
    - Demand-Paging, Swapping?
- Dynamic heap allocator in Kernel Space (Buddy System)
  - Previously: allocate with page size granularity
    - Waste of memory
    - Slow (un-)mapping

What else?

- Automation of test cycles
  - Tests on real hardware and SMP
  - Shorter turnaround times between test cycles
  - UART, PXE ...
- Benchmarks
  - Performance Monitoring Counter (PMC)
  - Membench
  - Translation Lookaside Buffer / Cache misses

Outlook

- Complete 32 bit version in MetalSVM
- Port concept to eduOS[9]
- Evaluate portability to other architectures
  - 64 bit ARM?
  - Sparc?
  - Alpha!
- Still room for performance improvements

[9] Work in progress.

Thank you for your kind attention!

Steffen Vogel – post@steffenvogel.de

Institute for Automation of Complex Power Systems
(Started at the Chair for Operating Systems)

E.ON Energy Research Center, RWTH Aachen University
Mathieustraße 10
52074 Aachen

www.eonerc.rwth-aachen.de
www.lfbs.rwth-aachen.de

Find code, slides and thesis at:
www.steffenvogel.de/2014/06/13/bachelor

Selected References

[1] J. Bangert, S. Bratus, and R. Shapiro.
Shmoocon talk: Page fault liberation army.
Trust Lab, Dartmouth College, Februar 2013.

[2] Compaq Computer Corporation, Houston, TX, USA.
Alpha Architecture Reference Manual, 4 edition, Januar 2002.

[3] Dave Probert.
Windows Kernel Architecture Internals.
Microsoft, MSRA/UR Workshop - Beijing, China, April 2010.

[4] Intel Corporation, Santa Klara, CA, USA.
Intel 64 and IA-32 Architectures Software Developer's Manual, Volumes 3A, 3B & 3C: System Programming Guide, Februar 2014.

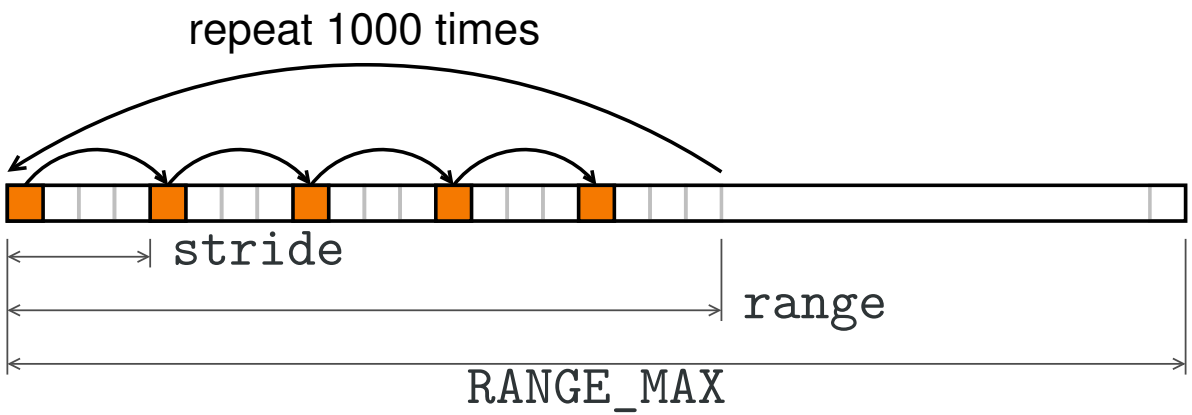
[5] P. Reble, J. Galowicz, S. Lankes, and T. Bemmerl.
Efficient implementation of the bare-metal hypervisor MetalSVM for the SCC.
In *Proceedings of the 6th Many-core Applications Research Community (MARC) Symposium*, pages 59–65, Juli 2012.

Acronyms

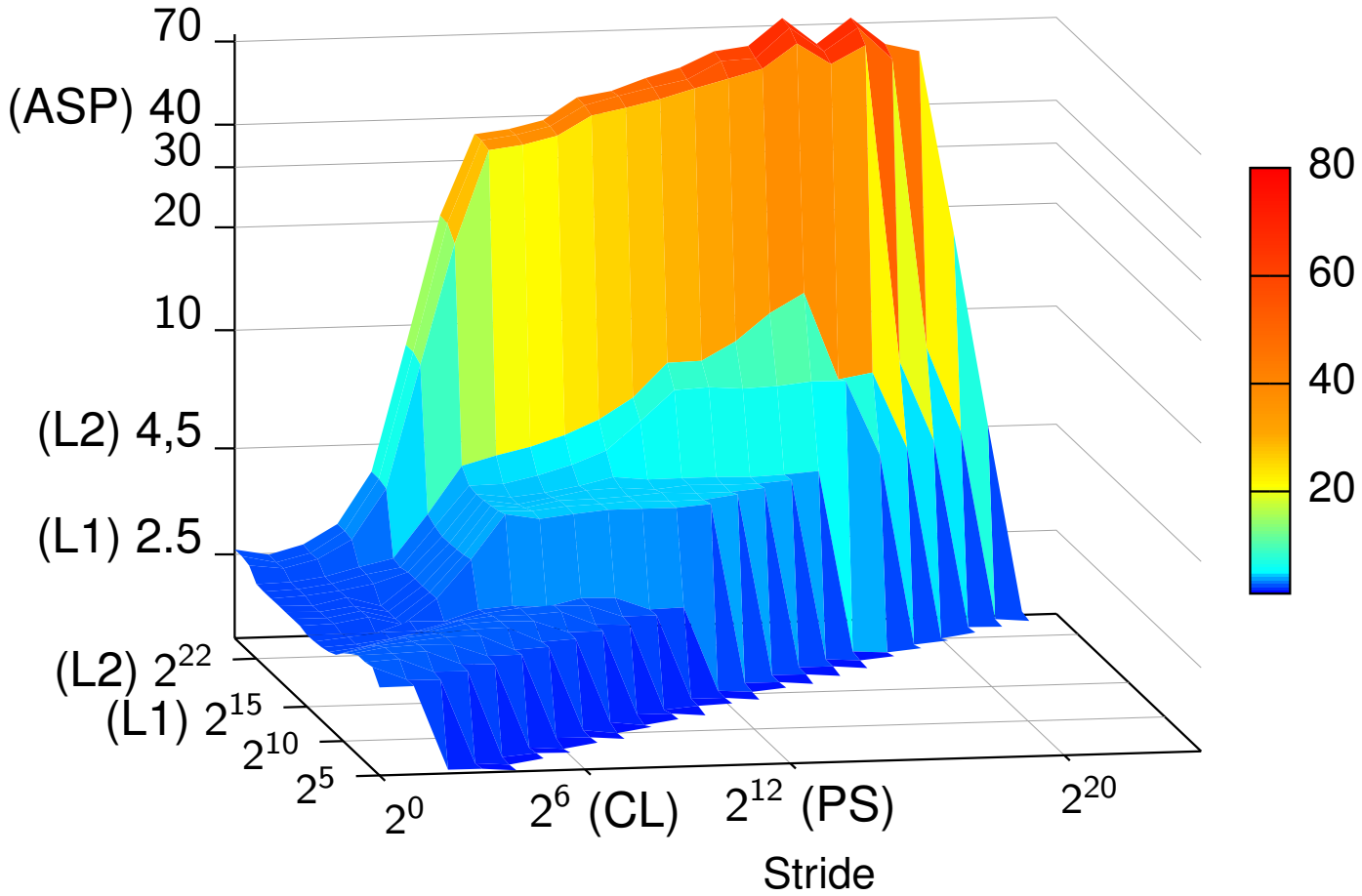
ASP	Main Memory
CL	Cache Line
L1	Level 1 Cache
L2	Level 2 Cache
MMU	Memory Management Unit
PFN	Page Frame Number
PMC	Performance Monitoring Counter
TLB	Translation Lookaside Buffer
VA	Virtual Address Space
PA	Physical Address Space
PS	Page Size
VMA	Virtual Memory Area
VPN	Virtual Page Number
PML4	Page Map Level 4
PML4E	Page Map Level 4 Entry
PDPT	Page Directory Pointer Table
PDPTE	Page Directory Pointer Table Entry
PGD	Page Directory
PDE	Page Directory Entry
PGT	Page Table
PTE	Page Table Entry
PF	Page Frame
CR3	Control Register 3

Benchmarks

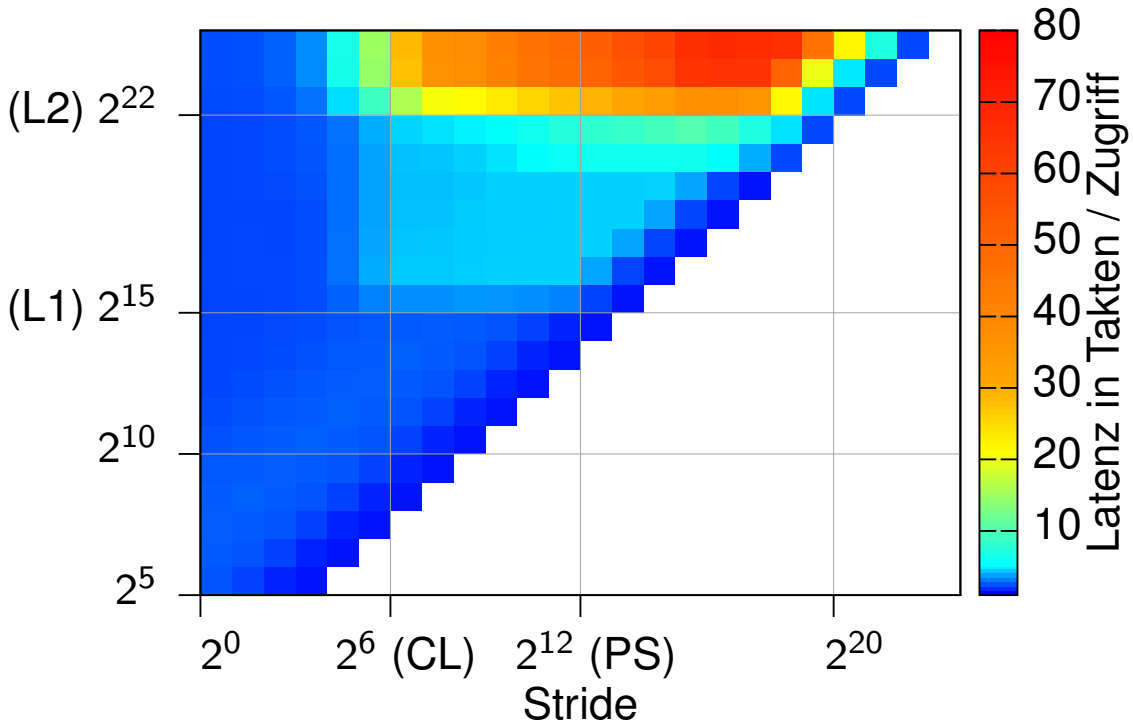
- Performance Monitoring Counter (PMC)
- Membench
  - Walk through memory by varying range and stride
  - Measure cost in terms CPU cycles and cache / TLB miss ratios
  - Infer cache and TLB sizes from results



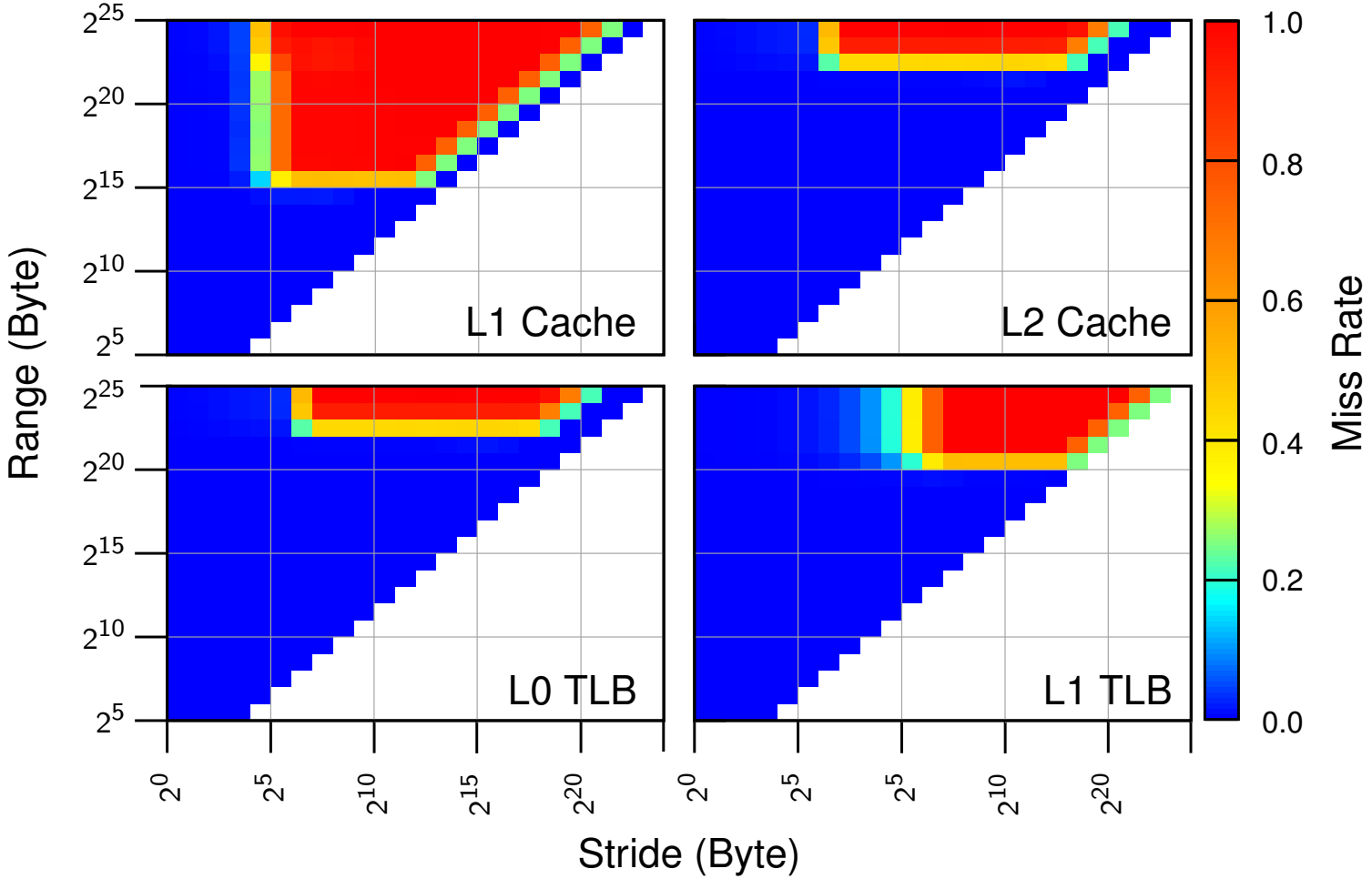
Membench Results



Access Latency



Cache / TLB misses



Mapping Cost