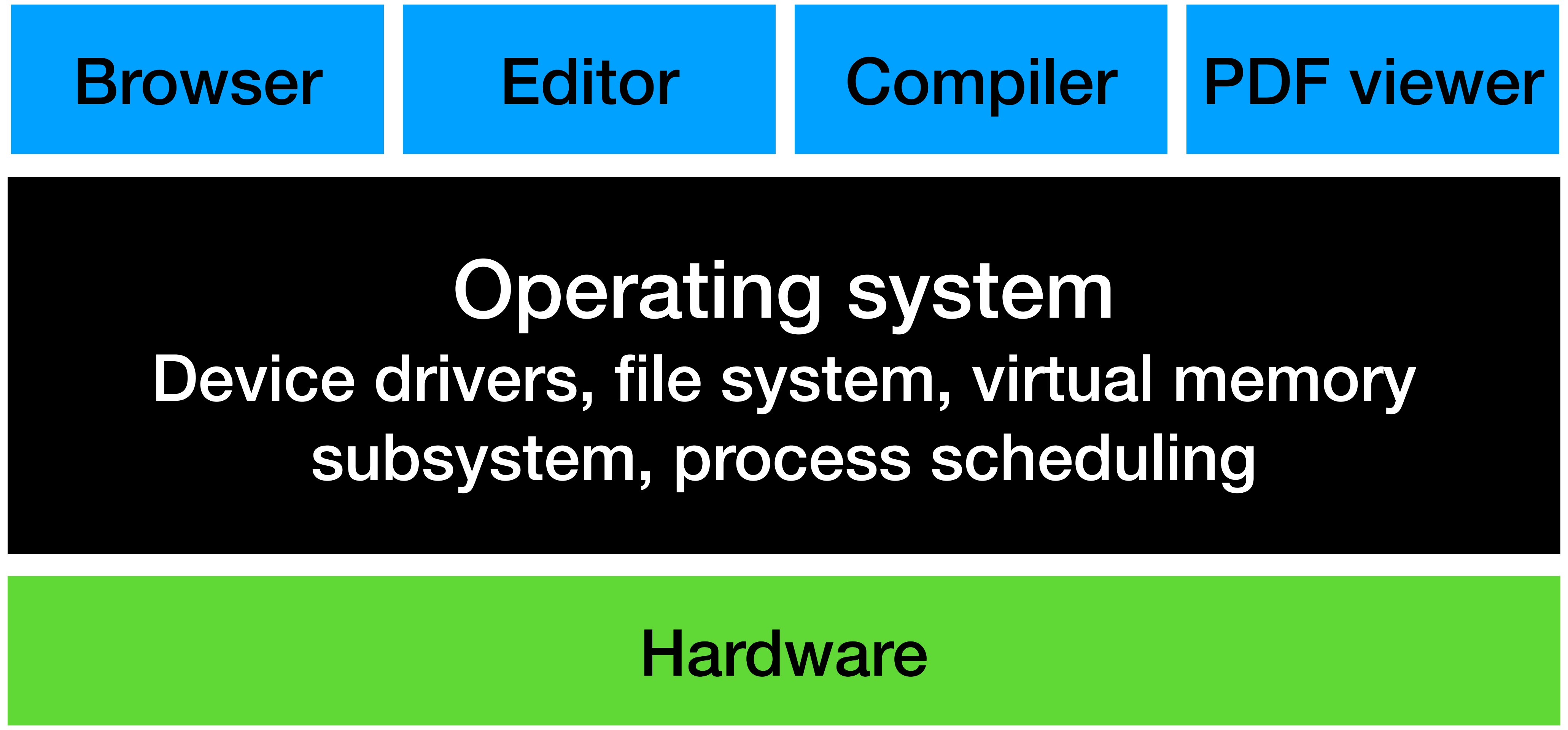
Alternate OS organization

Abhilash Jindal

Monolithic kernels

Example: xv6, Linux, Windows, etc

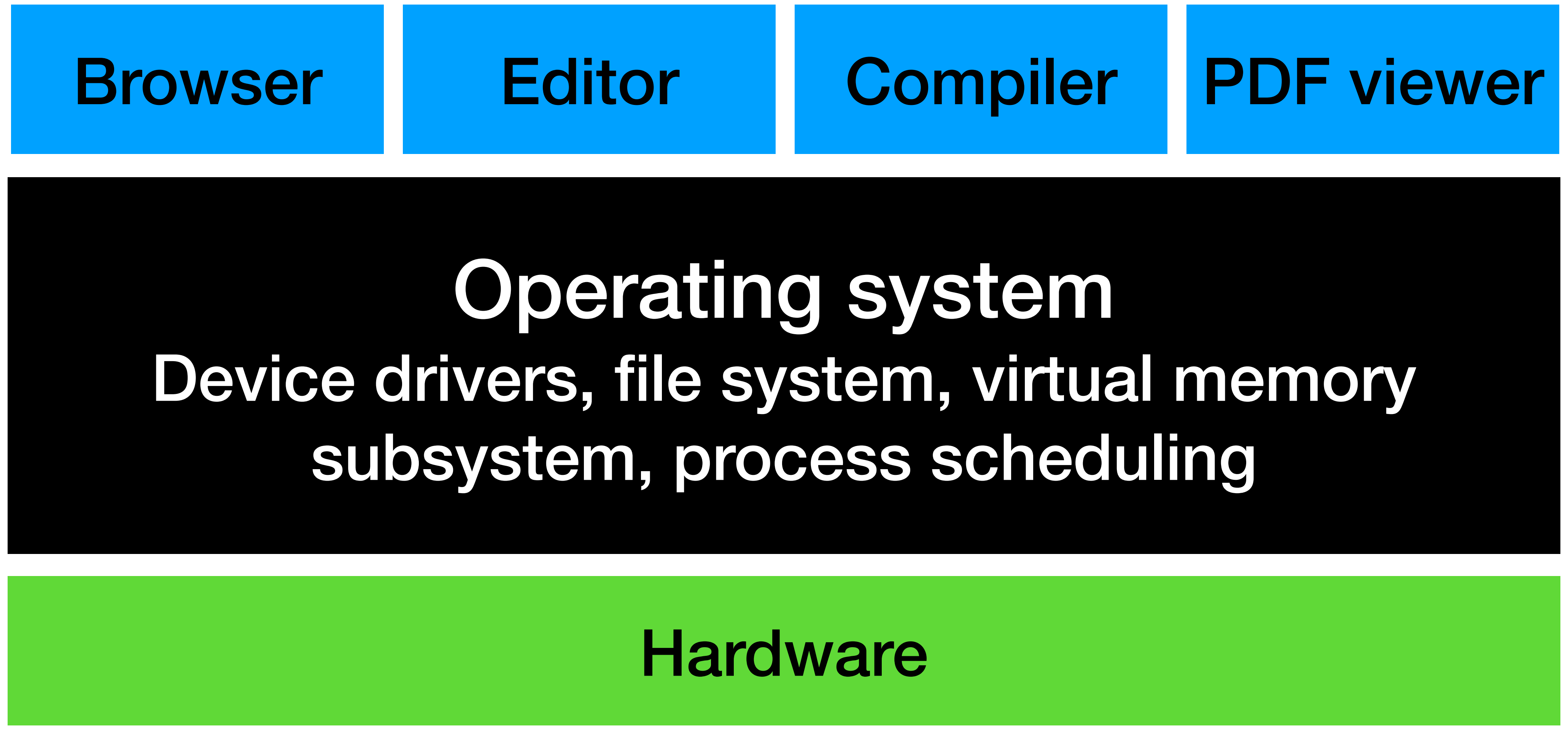
- A lot of components are running in privileged mode (ring 0 on x86) in the kernel's address space
- Large trusted computing base (TCB): A bug/security hole in one component can crash the system/ compromise system's security



Inflexibility of monolithic kernels

Example: xv6, Linux, Windows, etc

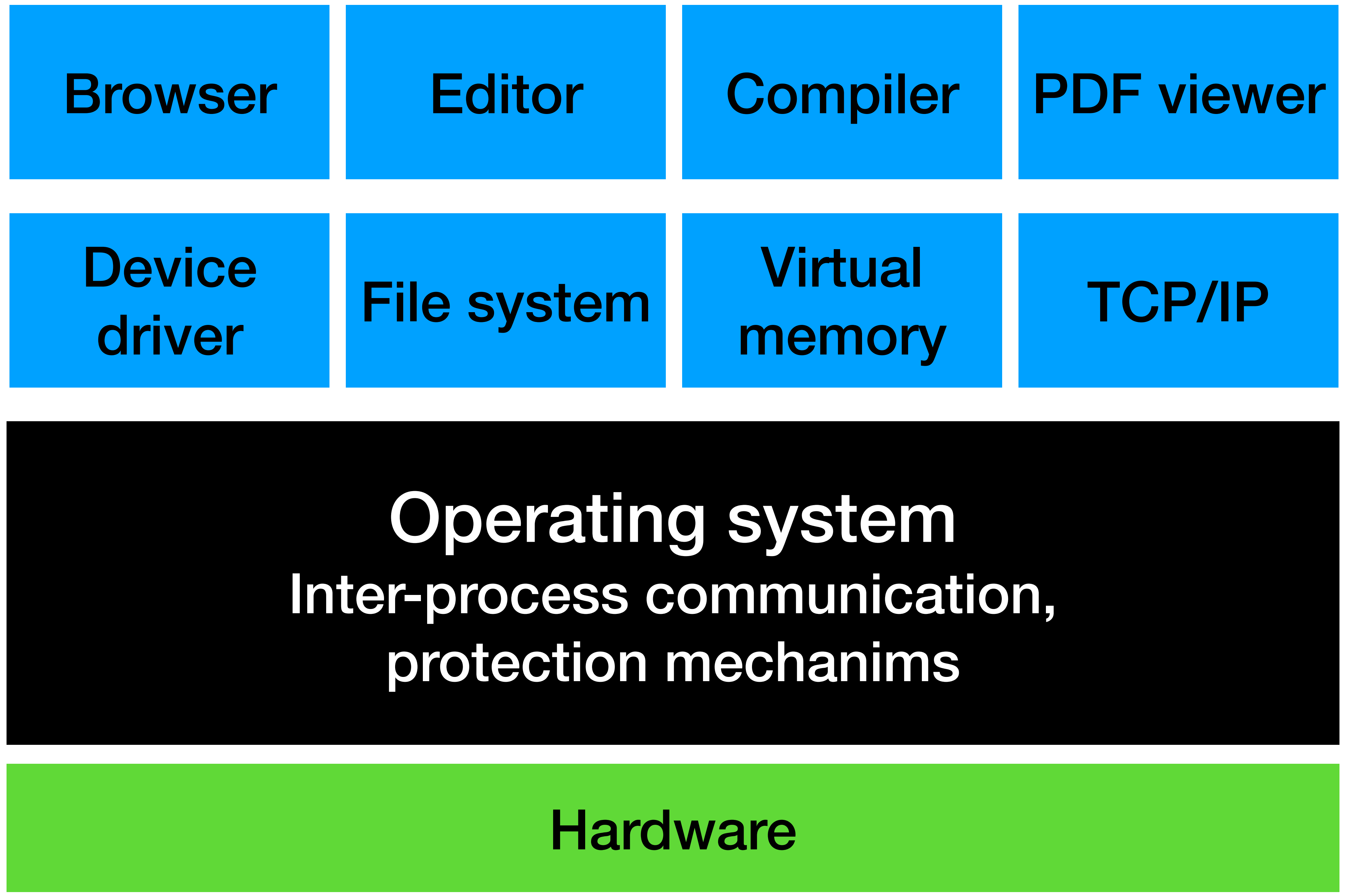
- OS components provide “higher-level abstractions” that optimize for the common case
- Example: LRU page replacement, file system interface
- “Uncommon applications” might suffer in performance



Microkernels

Example: L4, seL4

- OS becomes thinner with safe IPC facilities. “Servers” run in the user space
- Example: Using x86’s IO privilege level and IO permission bitmap to give device drivers direct access to particular devices
- More safety: Bug/security hole in one server does not affect whole system



Microkernel: flexibility

- Example: applications can pick their own “pager”.
- Different pagers can implement different page replacement strategies: FIFO, LRU, etc

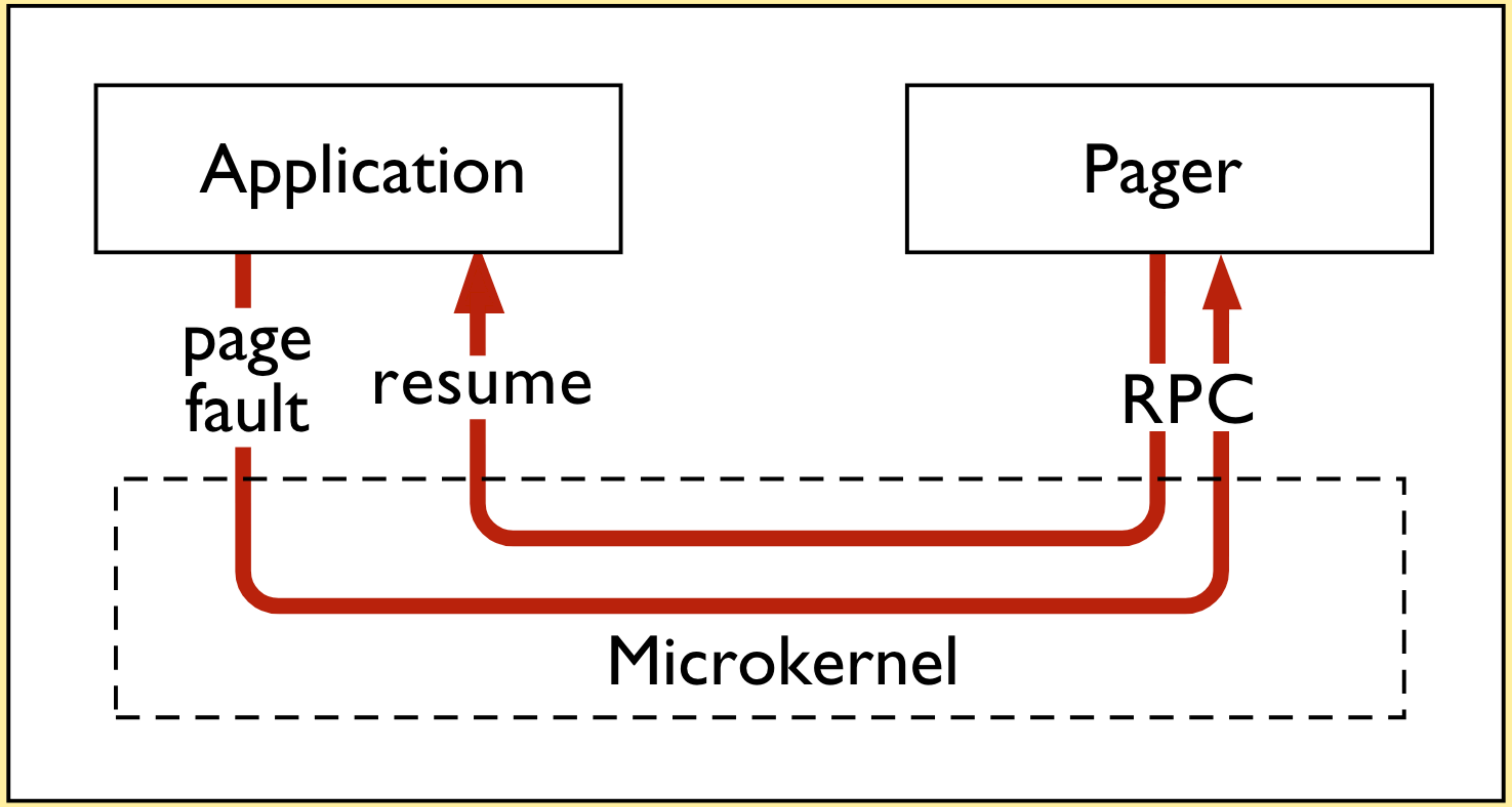
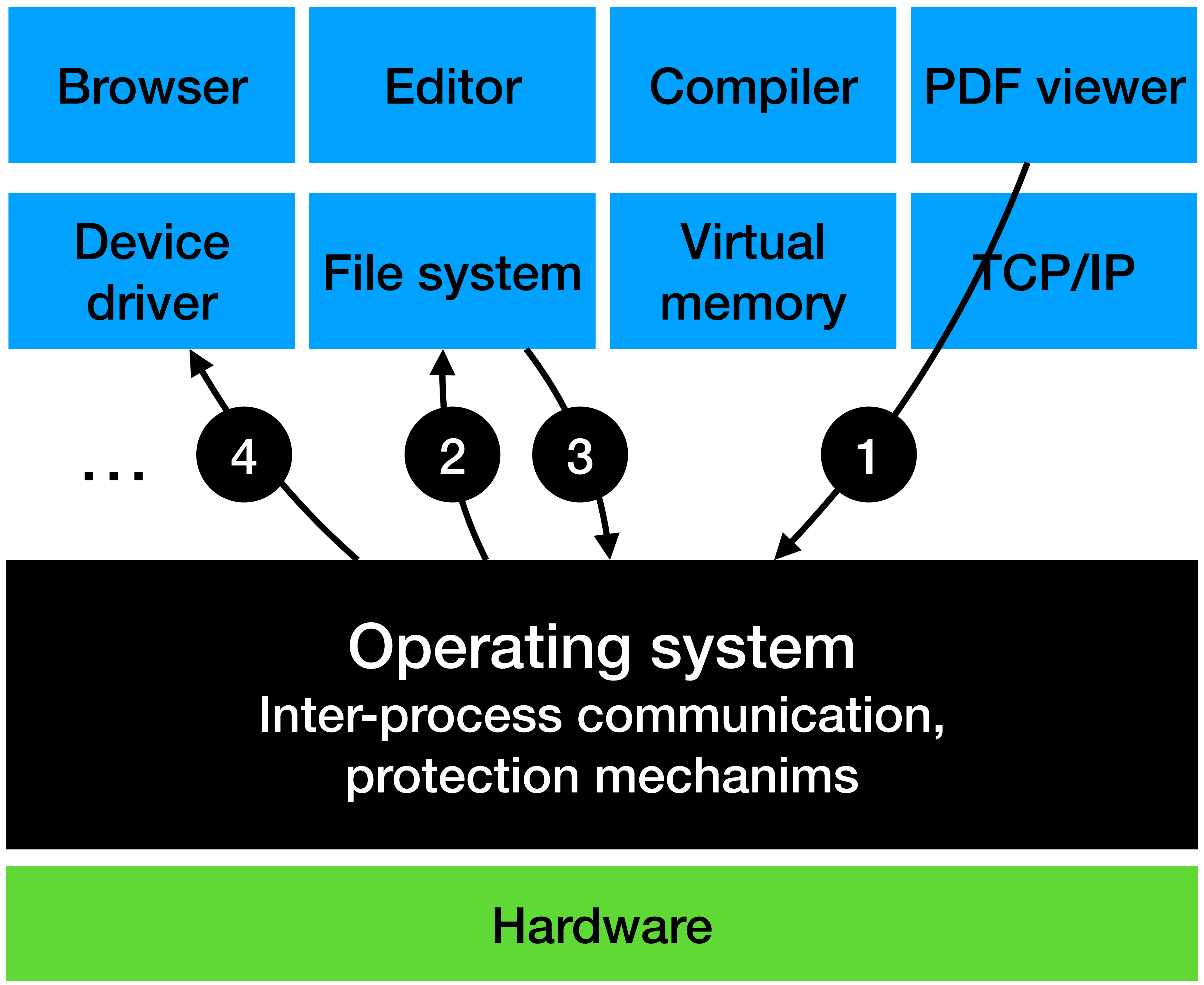


Figure 1. Page fault processing

Towards real microkernels (1996)

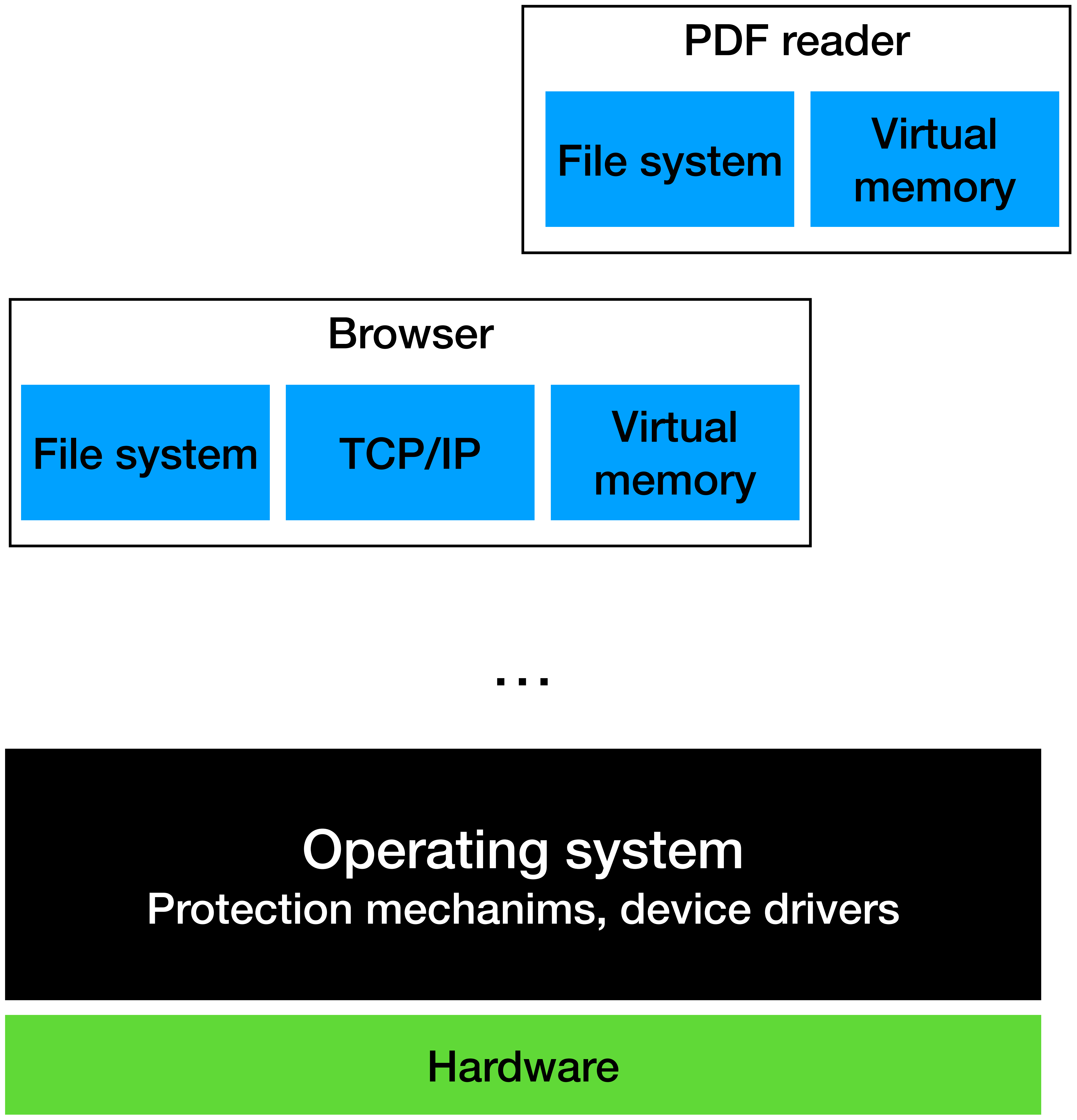
Challenge: IPC performance

- Passing data between apps and servers need to cross rings many times. Example: PDF viewer opens a file
- L4: OS allows apps and servers to share address space pages and communicate through them
- Still slower than monolithic kernels
- Used in safety critical systems like helicopters but not in regular desktops



Exokernel

- Applications are “library operating systems” (LibOS): they themselves package custom OS components. No IPC overhead!
- Exokernel OS exposes lowest-level abstractions: physical memory, disk blocks, TLB’s address context ID, packet filters
- Separate protection from management: applications understand what is the best way to manage hardware



Exokernel example

Context switching

- At every system call/interrupt, xv6 saves *all registers* on kernel stack
- Exokernel OS makes “switch out/switch in” upcalls into process to let it save/restore its own registers to/from wherever it want. Process can skip registers
- If process does not yield after switch out upcall, OS might kill the process at next timer interrupt (abort protocol)

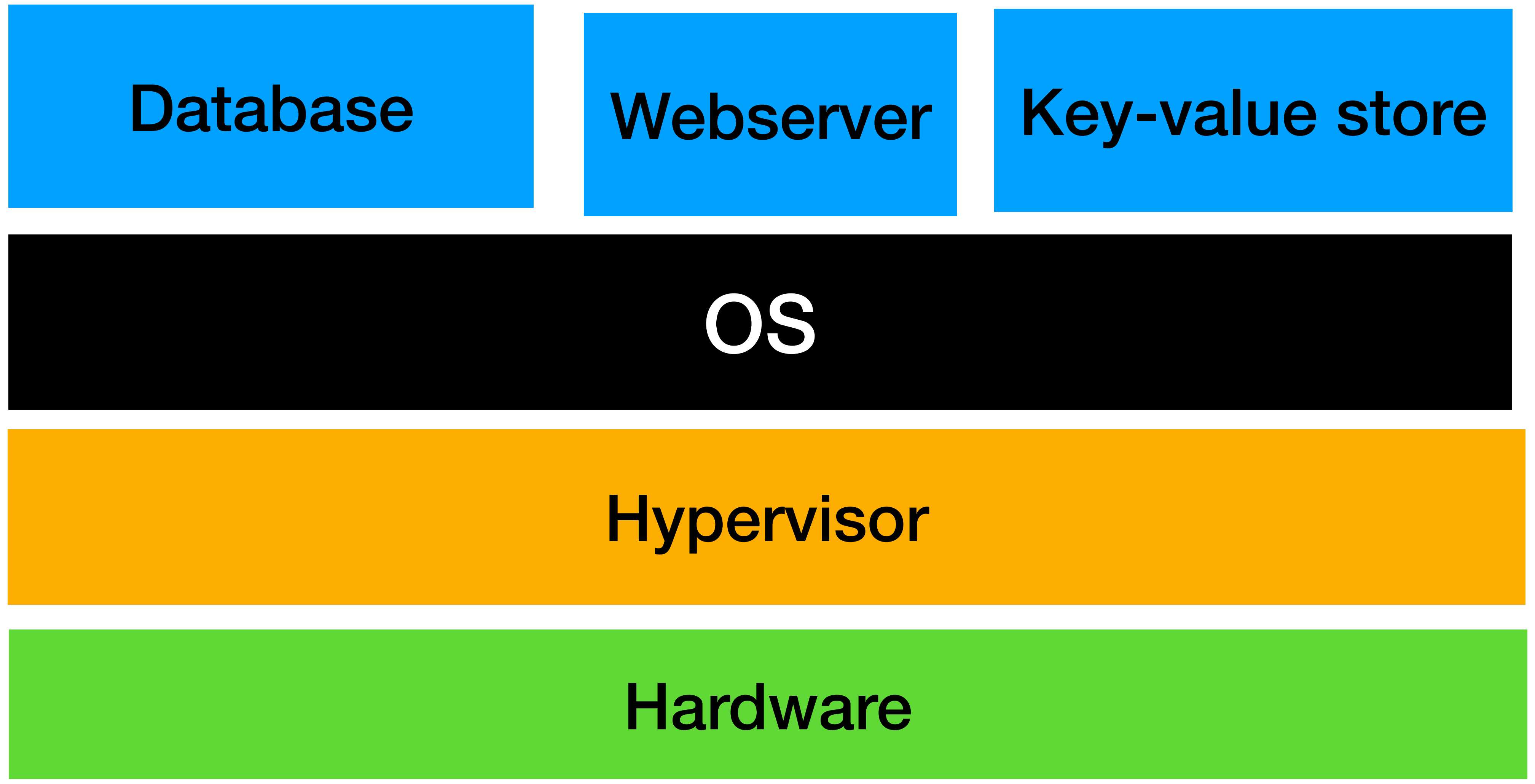
Exokernel example

Page replacement

- When Linux is running low on memory, swap daemon selects victim process, runs Clock algorithm to remove a victim’s page. LRU may not be optimal for the victim.
- Exokernel OS makes “PleaseReleasePage” upcall into victim process to let it decide the page to swap out.
- Page table pages are mapped into the address space of the process in a read-only manner
- Process does “DeAllocPage(physical address)” downcall (system call) to release the page
- If process does not release a page, OS might forcefully take a page away (abort protocol)

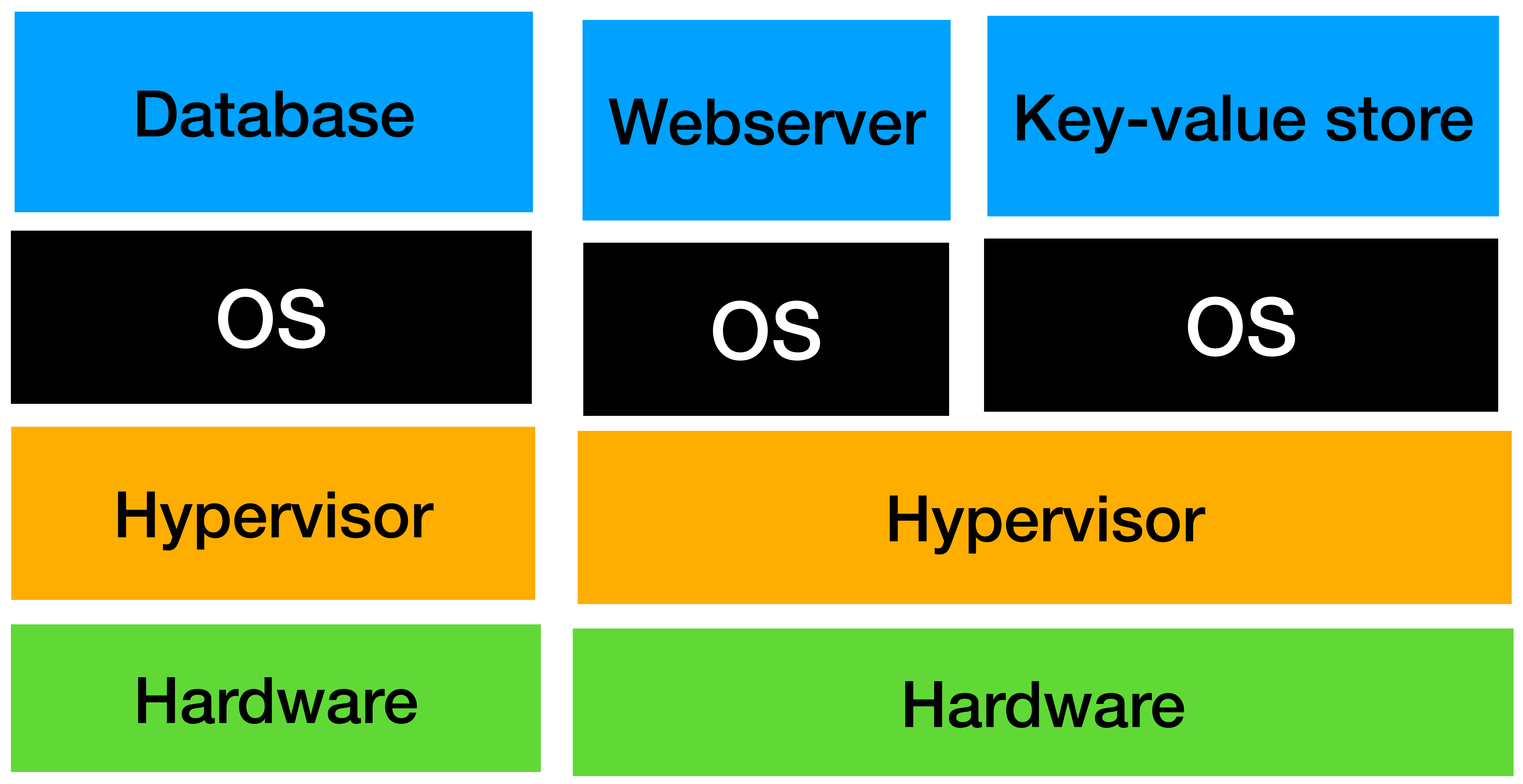
Monolithic applications in cloud

- Hypervisor multiplexes between OSes
- All applications are running within a single OS
- Application requirements are different!
    - It is ok to lose web server and key-value store data. Not ok for database.



Microservices

- In a cloud environment, customers deploy micro services



Unikernels

- Optimize the OS for the application that it will run!
- Application can pick its own networking stack, its own memory allocator
- There is no need for protection and paging since the OS is running just one process