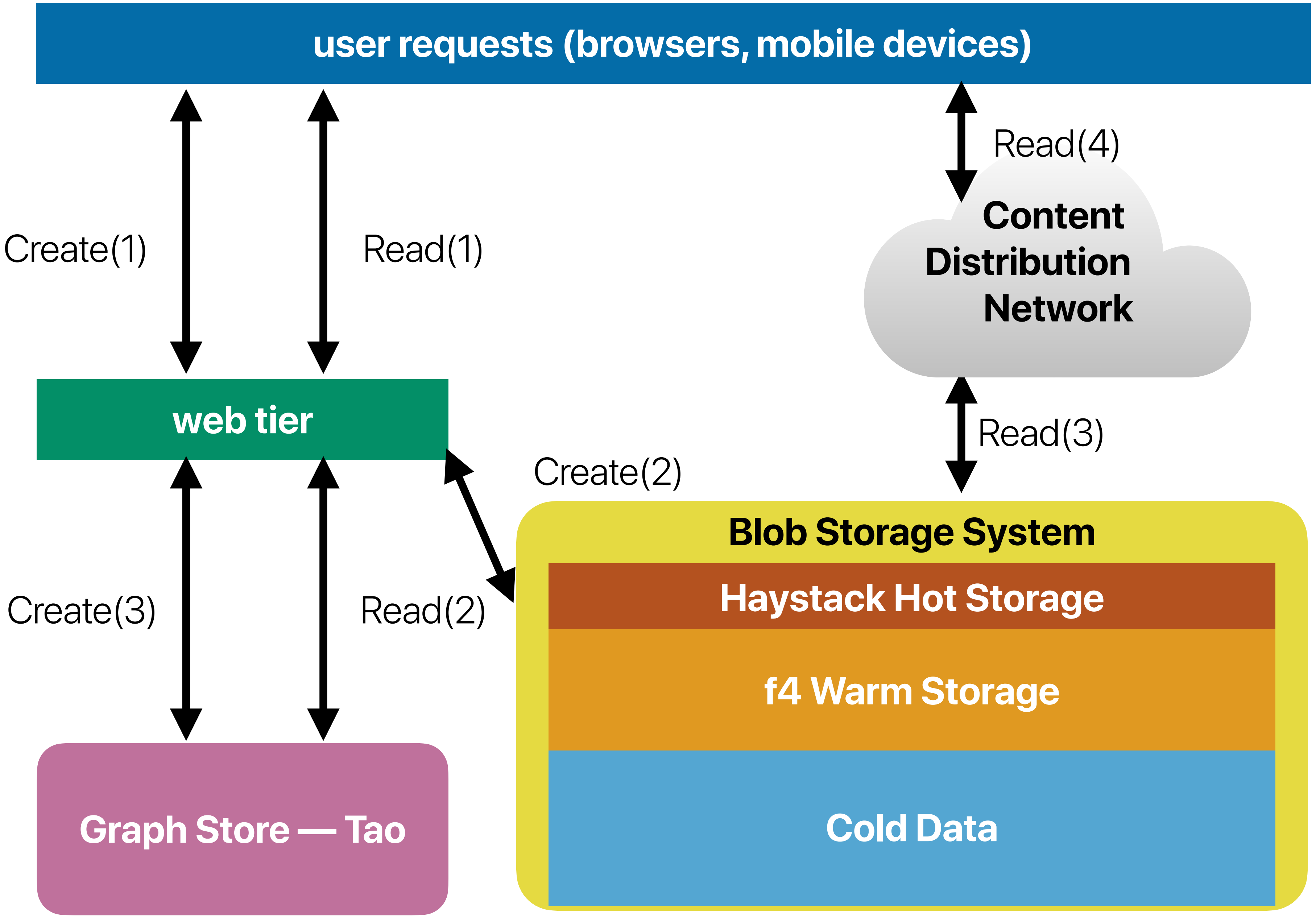
Google Search & Virtual Machines

Hung-Wei Tseng

Recap: GFS v.s. WAS v.s. Facebook

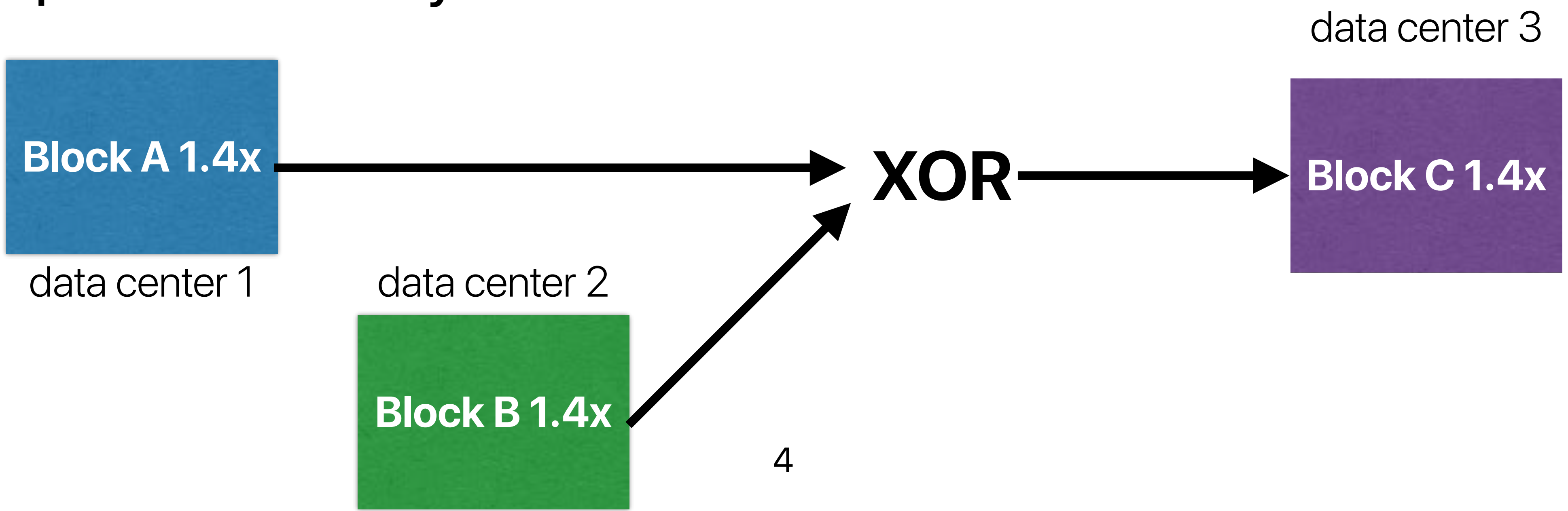
	GFS (OSDI 2003)	Facebook Haystack (OSDI 2010)	WAS (SOSP 2011)
File organizations	file chunk block	volume needle	stream extent record
System architecture	master chunkserver	directory haystack store	stream manager extent nodes
Data updates			append only updates
Consistency models	relaxed consistency		strong consistency
Data formats	files	photo/needle	multiple types of objects
Replications	intra-cluster replication	RAID-6 & geo-replication	geo-replication
Usage of nodes	chunk server can perform both		separate computation and storage

Recap: Facebook storage architecture



Recap: Storage efficiency

- Reed-Solomon erasure coding
  - Strips: 10GB data + 4GB parity — 1.4x space efficiency
  - One volume contains 10 strips
- XOR Geo-replication
  - Use XOR to reduce overhead further (e.g., Azure makes full copies)
  - Block A in DC1 + block B in DC2 -> parity block P in DC3
  - Any two blocks can be used to generate the third
  - 1.5x space efficiency
- 1.4*1.5 = 2.1x space efficiency in total

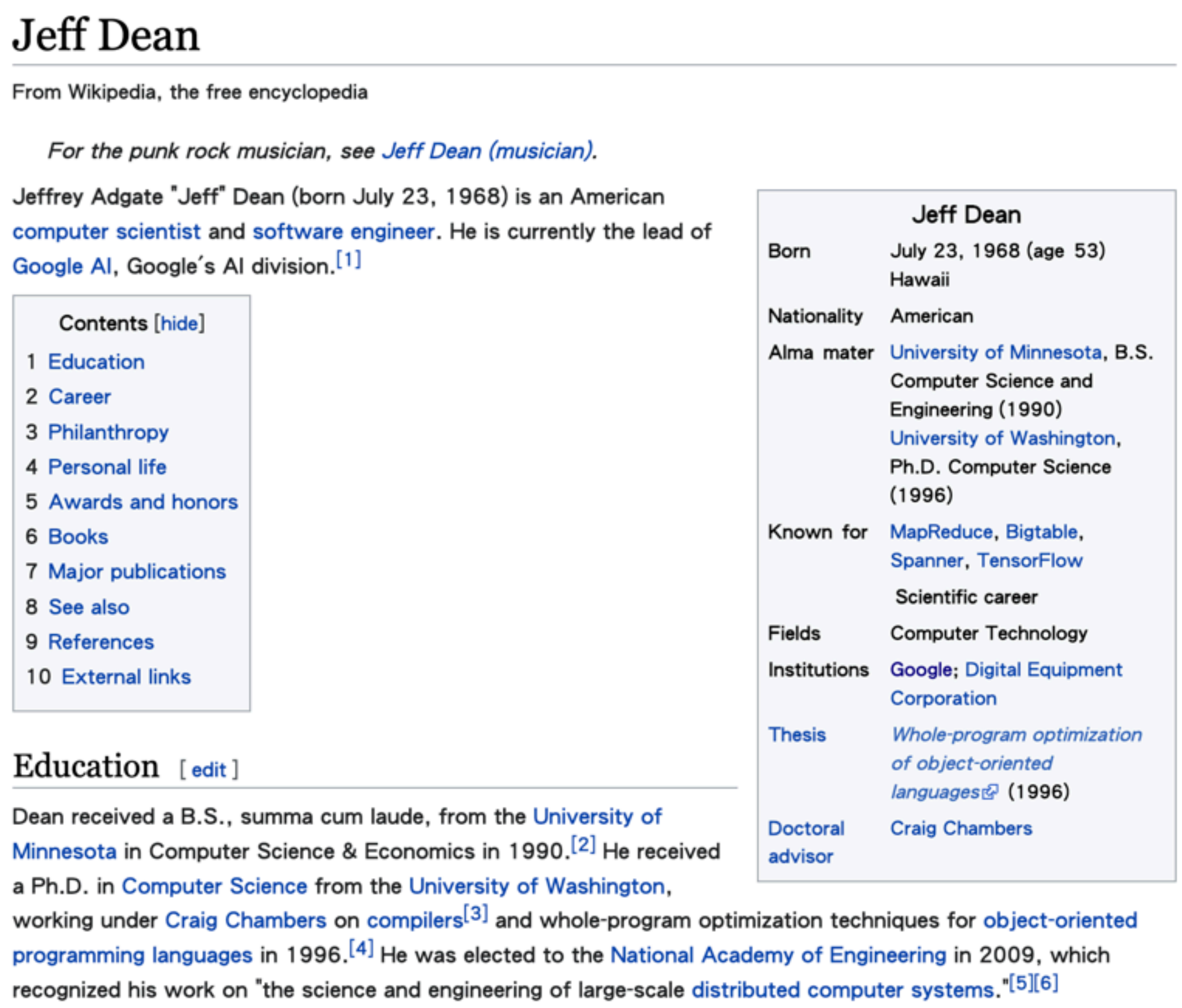


Outline

- Google Search
- Virtual machines

Web search fo
cluste

Luiz Andre Bar



Jeff Dean

From Wikipedia, the free encyclopedia

For the punk rock musician, see Jeff Dean (musician).

Jeffrey Adgate "Jeff" Dean (born July 23, 1968) is an American computer scientist and software engineer. He is currently the lead of Google AI, Google's AI division.[1]

Jeff Dean	
Born	July 23, 1968 (age 53) Hawaii
Nationality	American
Alma mater	University of Minnesota, B.S. Computer Science and Engineering (1990) University of Washington, Ph.D. Computer Science (1996)
Known for	MapReduce, Bigtable, Spanner, TensorFlow
	Scientific career
Fields	Computer Technology
Institutions	Google; Digital Equipment Corporation
Thesis	*Whole-program optimization of object-oriented languages* (1996)
Doctoral advisor	Craig Chambers

Contents [hide]

Education [edit]

Dean received a B.S., summa cum laude, from the University of Minnesota in Computer Science & Economics in 1990.[2] He received a Ph.D. in Computer Science from the University of Washington, working under Craig Chambers on compilers[3] and whole-program optimization techniques for object-oriented programming languages in 1996.[4] He was elected to the National Academy of Engineering in 2009, which recognized his work on "the science and engineering of large-scale distributed computer systems."[5][6]

https://www.pollev.com/hungweitseng close in 1:00

Google search architecture

- How many of the following fulfill the design agenda of the Google search architecture described in this paper?
  ① Reduce the hardware cost by using commodity-class and unreliable PCs
  ② Use RAID to provide efficiency and reliability
  ③ Use replication for better request throughput and availability
  ④ Optimize for the peak performance
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Google Search

A

B

C

D

E

Total Results: 0

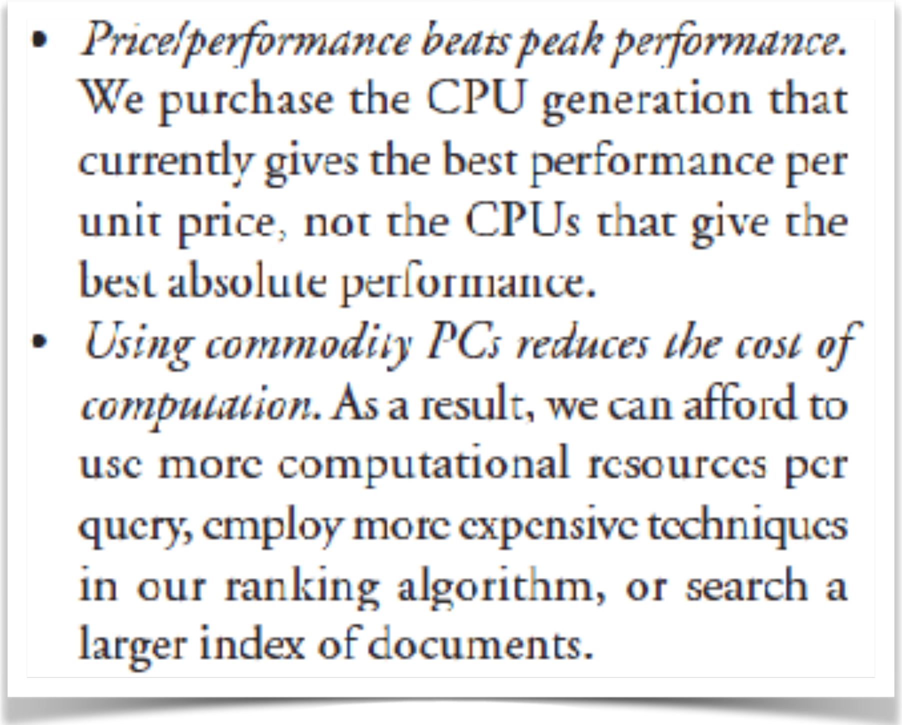
Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

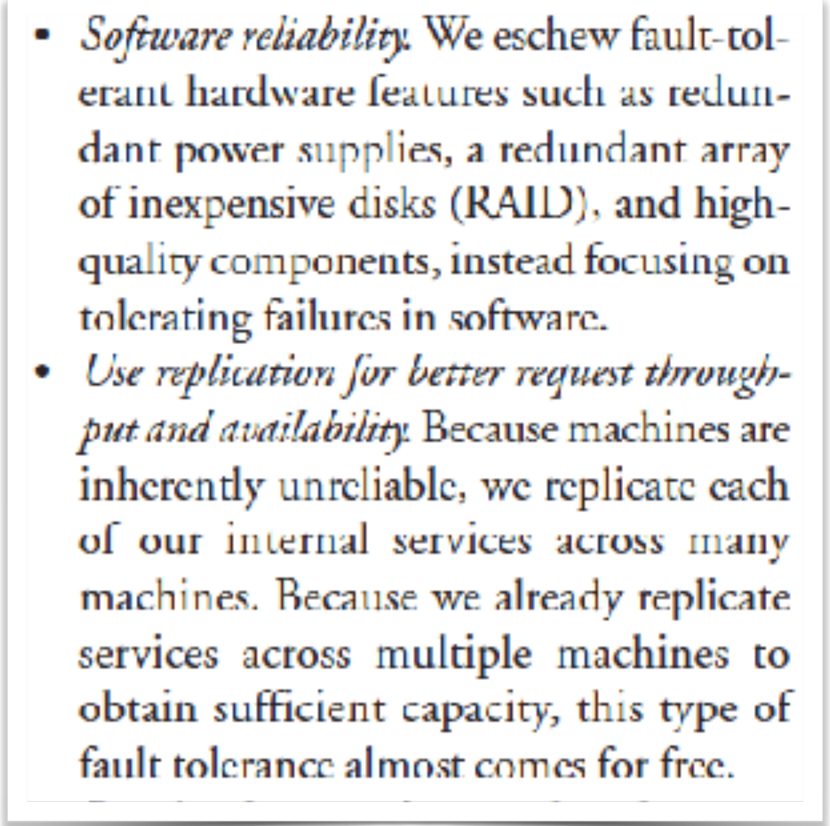
Google search architecture

- How many of the following fulfill the design agenda of the Google search architecture described in this paper?
  - [✓ ①] Reduce the hardware cost by using commodity-class and unreliable PCs
  - ② Use RAID to provide efficiency and reliability
  - [✓ ③] Use replication for better request throughput and availability — replica, replica, replica
  - ④ Optimize for the peak performance — for performance per dollar
  - A. 0
  - B. 1
  - [✓ C. 2]
  - D. 3
  - E. 4

Also reliability and fault-tolerance



- *Price/performance beats peak performance.* We purchase the CPU generation that currently gives the best performance per unit price, not the CPUs that give the best absolute performance.
- *Using commodity PCs reduces the cost of computation.* As a result, we can afford to use more computational resources per query, employ more expensive techniques in our ranking algorithm, or search a larger index of documents.



- *Software reliability.* We eschew fault-tolerant hardware features such as redundant power supplies, a redundant array of inexpensive disks (RAID), and high-quality components, instead focusing on tolerating failures in software.
- *Use replication for better request throughput and availability.* Because machines are inherently unreliable, we replicate each of our internal services across many machines. Because we already replicate services across multiple machines to obtain sufficient capacity, this type of fault tolerance almost comes for free.

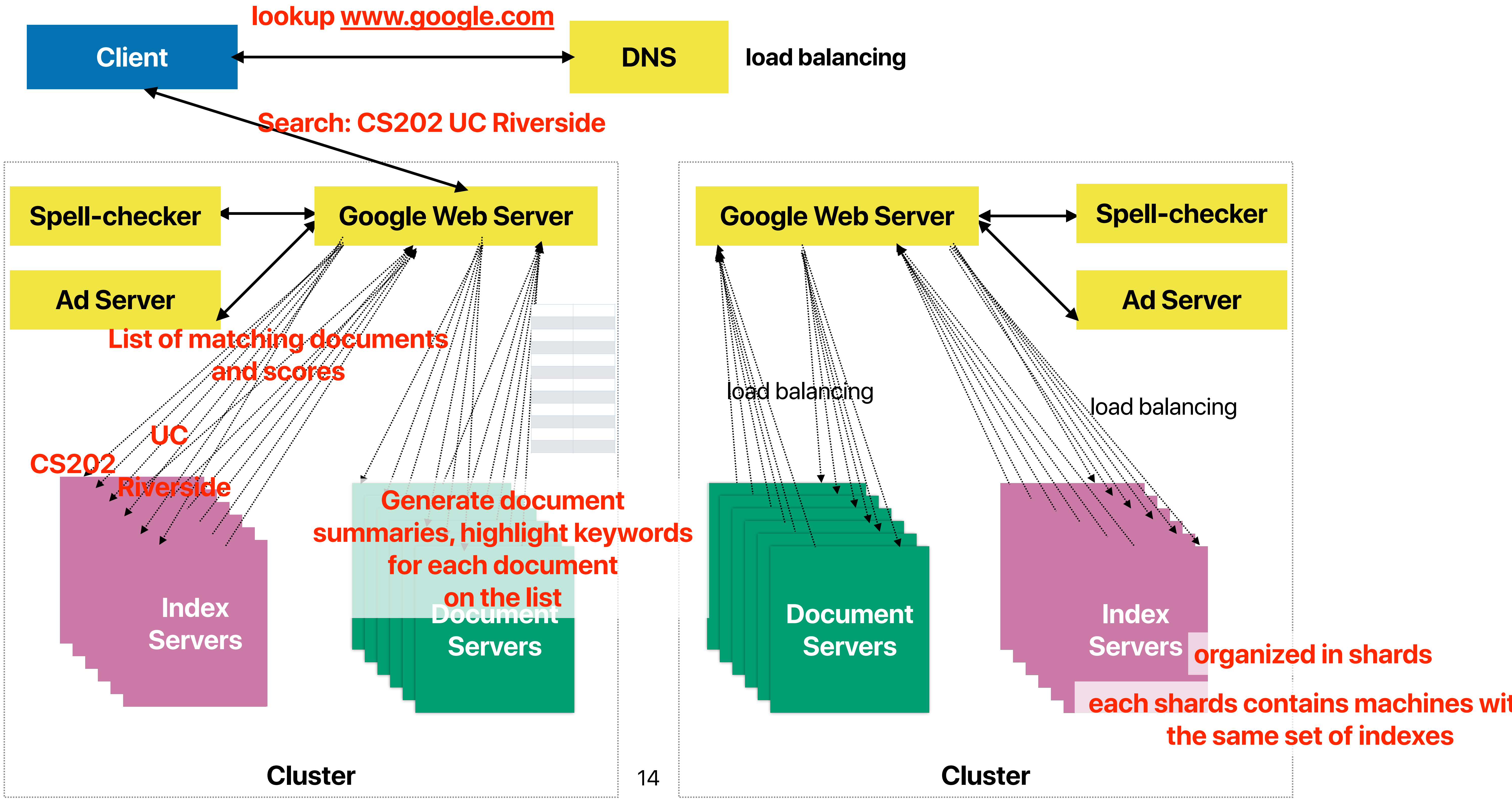
Why Google Search Architecture?

- The demand of performing search queries efficiently
  - Each query reads hundreds of MBs of data
  - Support the peak traffic would require expensive supercomputers or high-end servers
- We need a **cost-effective** approach to address this demand
  - Google search is compare against "AltaVista" search engine that uses DEC's high-performance alpha-based multiprocessor systems
  - AltaVista is later acquired by Yahoo! and you know the later story…

What Google proposes?

- Using commodity-class / unreliable PCs
- Provide reliability in software rather in hardware
- Target the best aggregate request throughput, not peak server response time

Google query-serving architecture



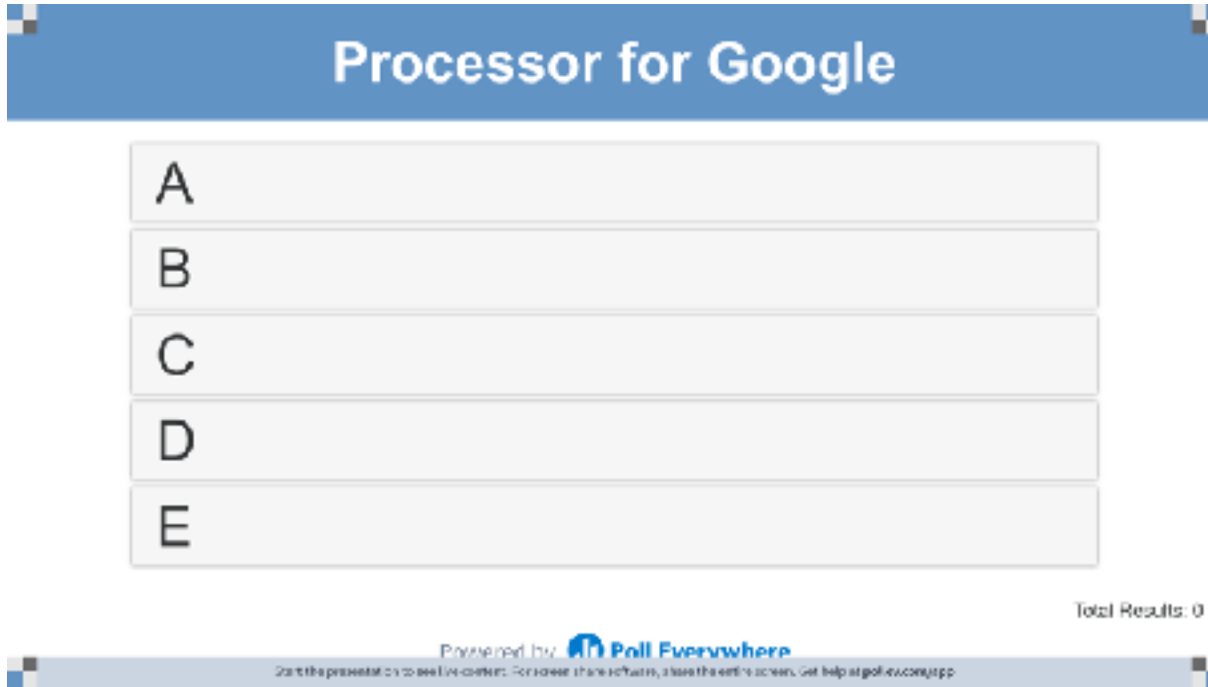
Replication is the key

- Scalability: simply add more replicas, the service capacity can improve
- Availability: even though one machine fails, another replica to take over

https://www.pollev.com/hungweitseng close in 1:00

What kind of processors Google search needs

- If we are designing a processor just for Google search or similar type of applications, how many of the following targets/features would fulfill the demand?
  ① Can execute many instructions from the same process/thread simultaneously
  ② Can execute many processes/threads simultaneously
  ③ Can predict branch outcome accurately
  ④ Have very large cache capacity
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4



Processor for Google

A

B

C

D

E

Total Results: 0

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

What kind of processors Google search needs

- If we are designing a processor just for Google search or similar type of applications, how many of the following targets/features would fulfill the demand?
  - ① Can execute many instructions from the same process/thread simultaneously
  - ✓ ② Can execute many processes/threads simultaneously
  - ✓ ③ Can predict branch outcome accurately
  - ④ Have very large cache capacity

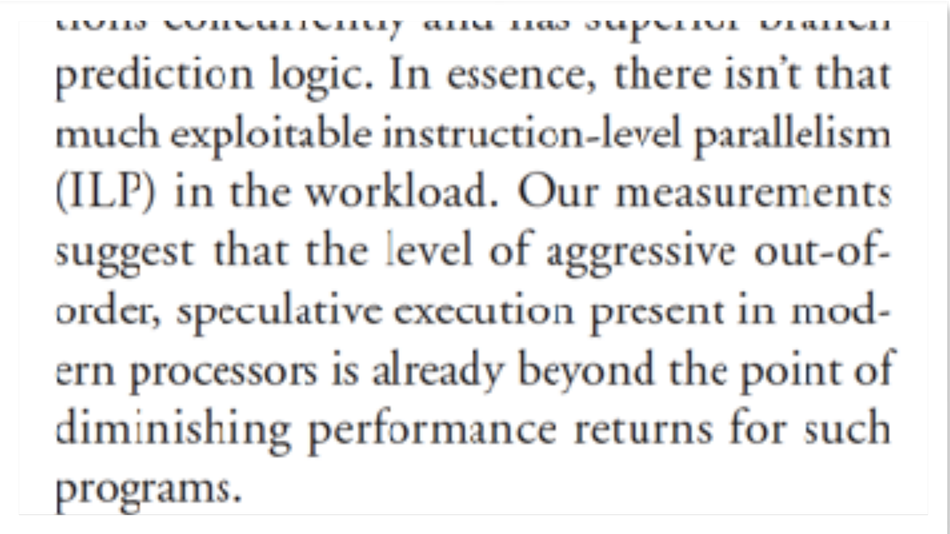
  A. 0

  B. 1

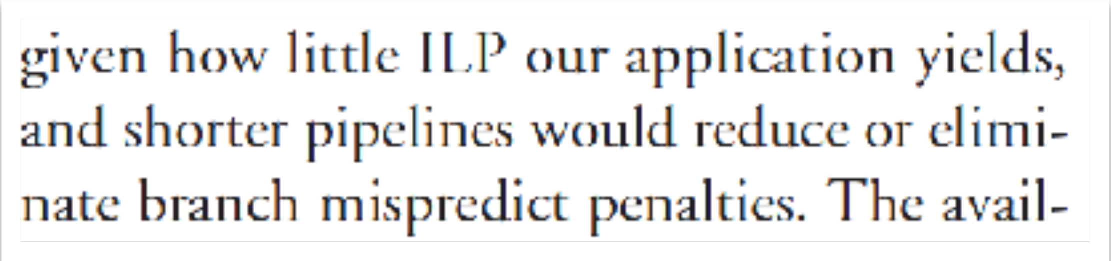
  [✓ C. 2]

  D. 3

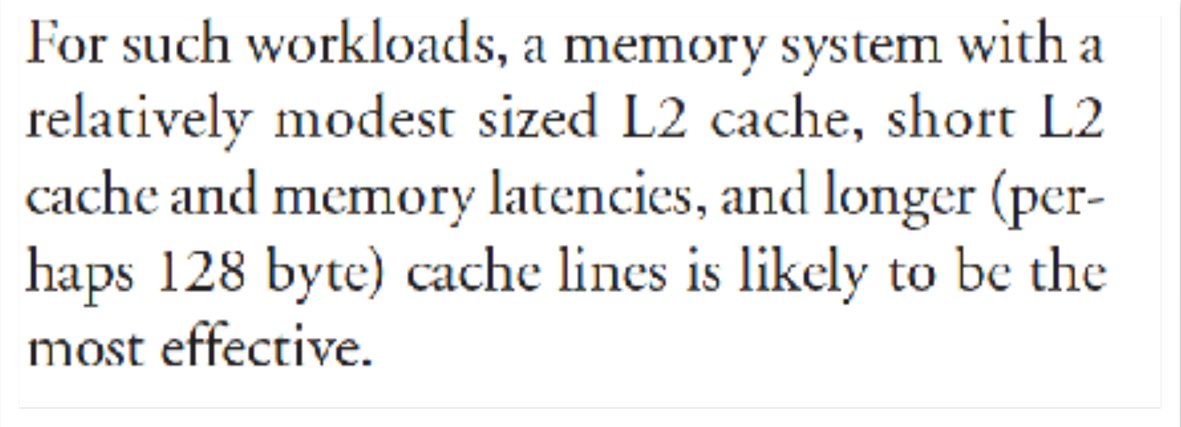
  E. 4



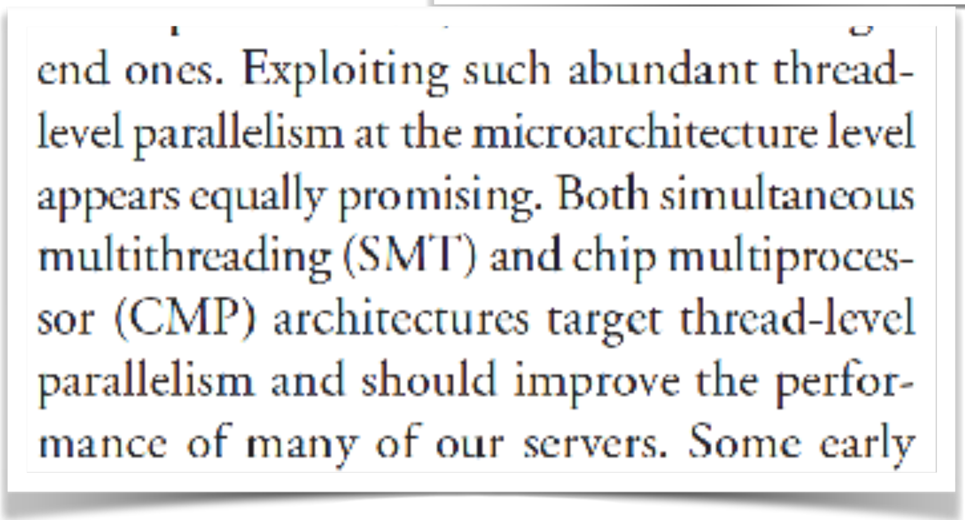
tions concurrently and has superior branch prediction logic. In essence, there isn't that much exploitable instruction-level parallelism (ILP) in the workload. Our measurements suggest that the level of aggressive out-of-order, speculative execution present in modern processors is already beyond the point of diminishing performance returns for such programs.



given how little ILP our application yields, and shorter pipelines would reduce or eliminate branch mispredict penalties. The avail-



For such workloads, a memory system with a relatively modest sized L2 cache, short L2 cache and memory latencies, and longer (perhaps 128 byte) cache lines is likely to be the most effective.

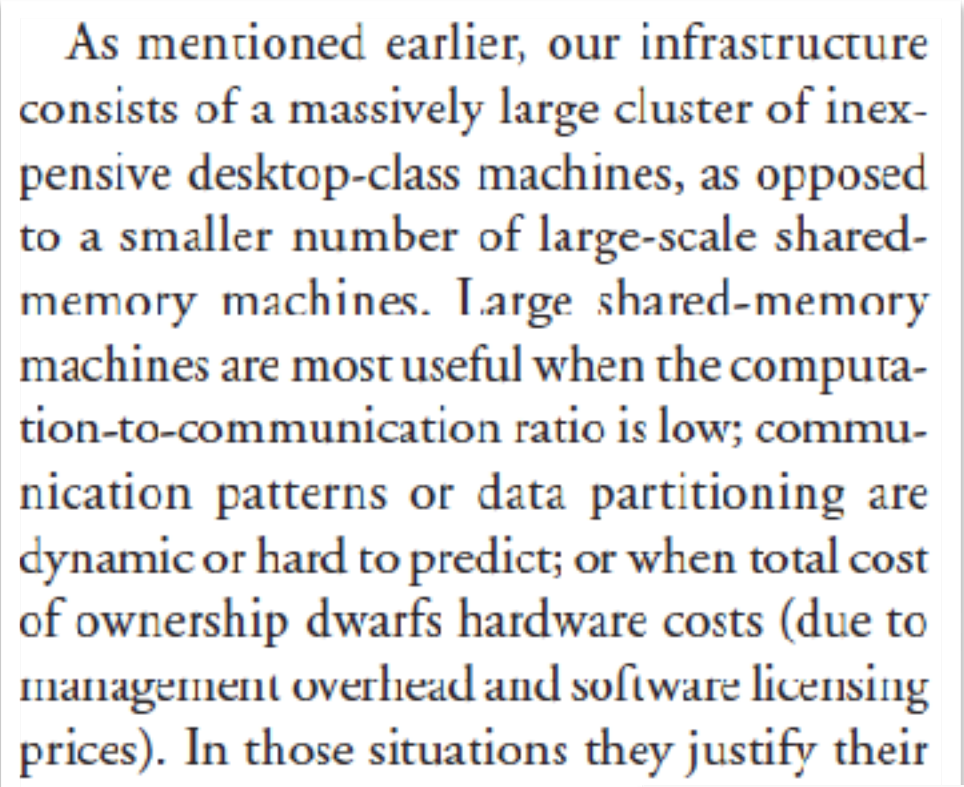


end ones. Exploiting such abundant thread-level parallelism at the microarchitecture level appears equally promising. Both simultaneous multithreading (SMT) and chip multiprocessor (CMP) architectures target thread-level parallelism and should improve the performance of many of our servers. Some early

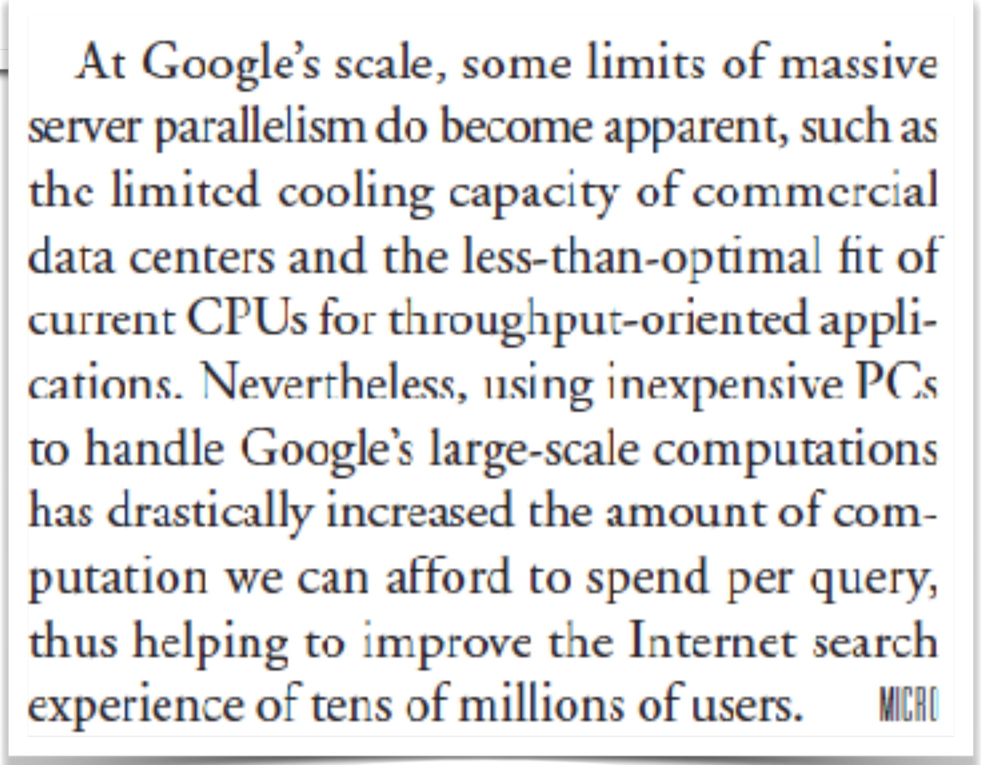
Hardware

- Processor
  - Index search has little ILPs — doesn't need complex cores
  - Index search can be highly parallelized — processors with thread-level parallelism would be a good fit (e.g. Simultaneous Multithreading, SMT and Chip Multicprocessor, CMP)
  - Branch predictor matters
- Memory: Good spatial locality. Moderate cache size will suffice
- Storage: No SCSI, No RAID — not worth it
- Power: is an issue, but only $1,500/mo operating bill vs $7,700 capital expense

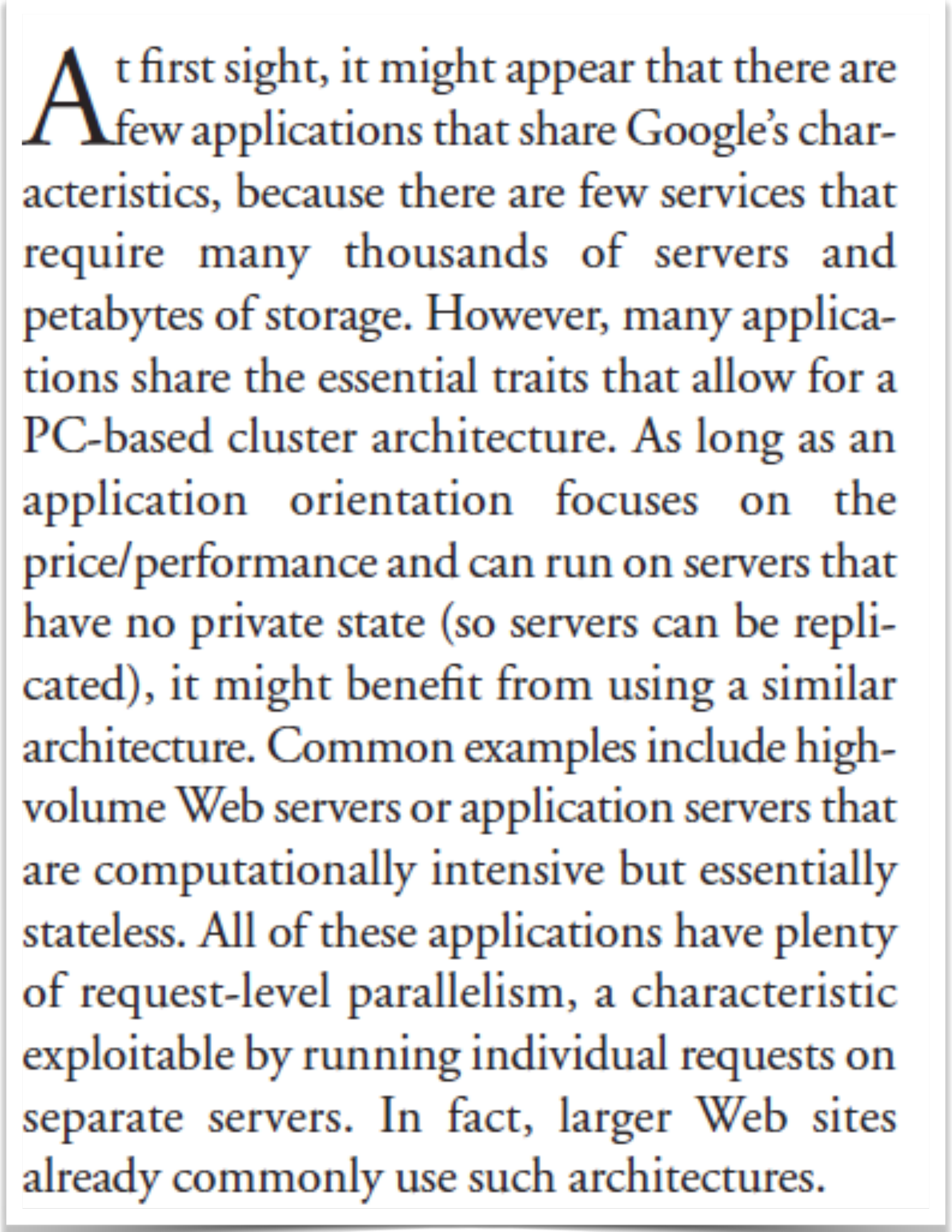
Will their architecture work for other things?



As mentioned earlier, our infrastructure consists of a massively large cluster of inexpensive desktop-class machines, as opposed to a smaller number of large-scale shared-memory machines. Large shared-memory machines are most useful when the computation-to-communication ratio is low; communication patterns or data partitioning are dynamic or hard to predict; or when total cost of ownership dwarfs hardware costs (due to management overhead and software licensing prices). In those situations they justify their high price tags.



At Google's scale, some limits of massive server parallelism do become apparent, such as the limited cooling capacity of commercial data centers and the less-than-optimal fit of current CPUs for throughput-oriented applications. Nevertheless, using inexpensive PCs to handle Google's large-scale computations has drastically increased the amount of computation we can afford to spend per query, thus helping to improve the Internet search experience of tens of millions of users.



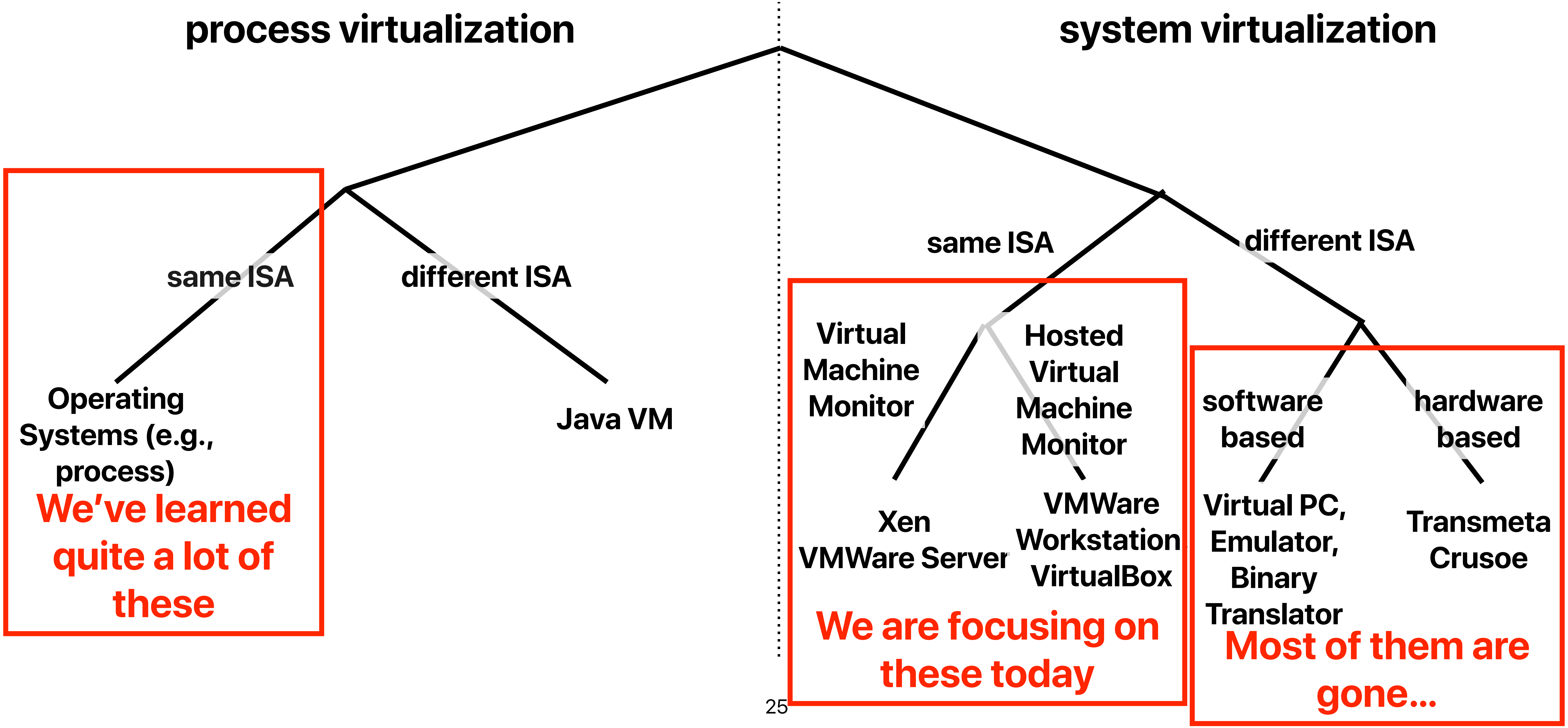
At first sight, it might appear that there are few applications that share Google's characteristics, because there are few services that require many thousands of servers and petabytes of storage. However, many applications share the essential traits that allow for a PC-based cluster architecture. As long as an application orientation focuses on the price/performance and can run on servers that have no private state (so servers can be replicated), it might benefit from using a similar architecture. Common examples include high-volume Web servers or application servers that are computationally intensive but essentially stateless. All of these applications have plenty of request-level parallelism, a characteristic exploitable by running individual requests on separate servers. In fact, larger Web sites already commonly use such architectures.

Metrics we care about data center design

- Costs — machine architecture, distributed system architecture, replication strategies
- Power — machine architecture
- Energy — machine architecture
- Space-efficiency — erasure coding, replication, distributed
- Throughput — replication, distributed
- Reliability — replication

Virtual Machines

Taxonomy of virtualization



process virtualization

same ISA

Operating Systems (e.g., process)

We've learned quite a lot of these

different ISA

Java VM

system virtualization

same ISA

Virtual Machine Monitor

Xen
VMWare Server

Hosted Virtual Machine Monitor

VMWare Workstation
VirtualBox

We are focusing on these today

different ISA

software based

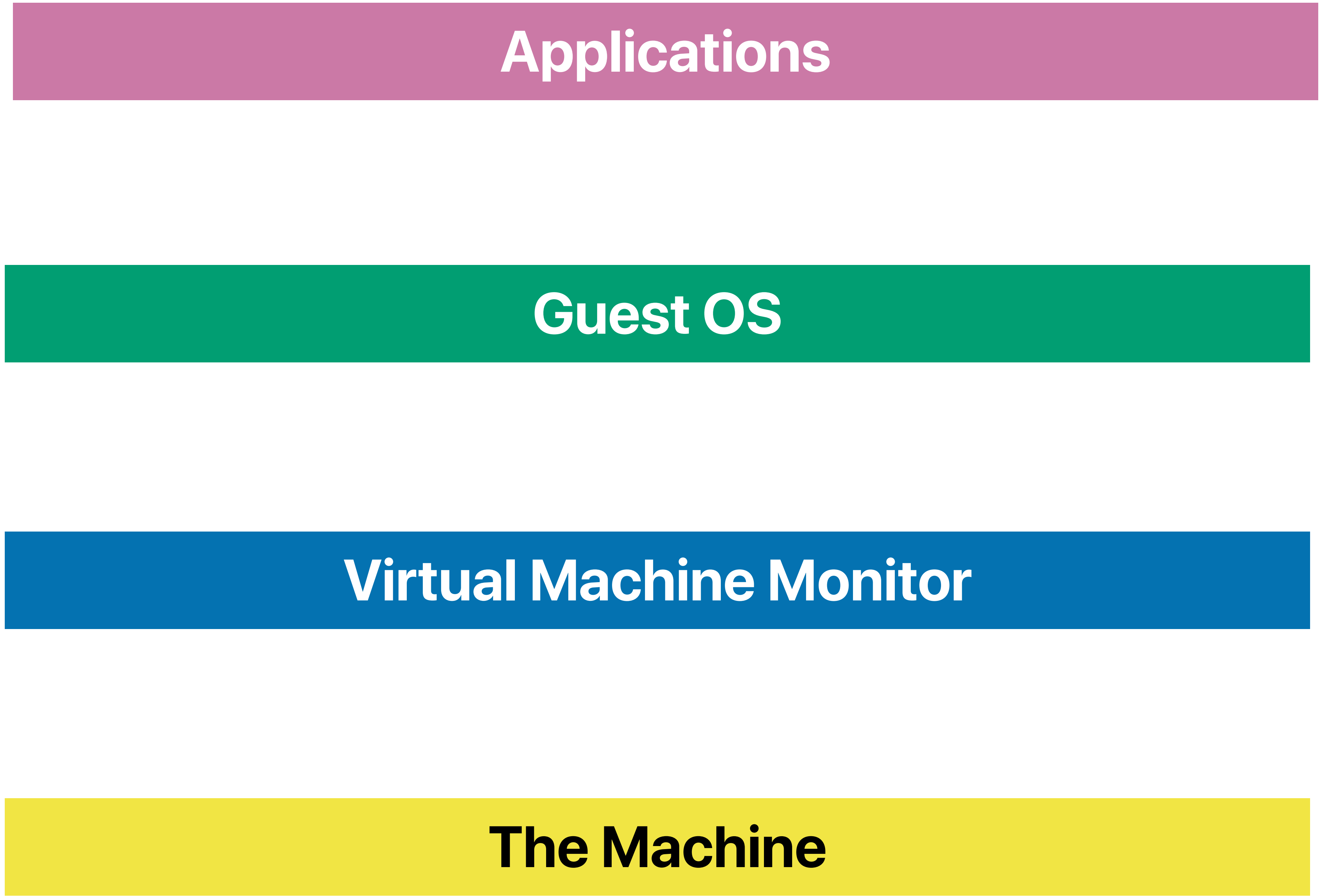
Virtual PC, Emulator, Binary Translator

hardware based

Transmeta Crusoe

Most of them are gone...

Virtual machine architecture



Applications

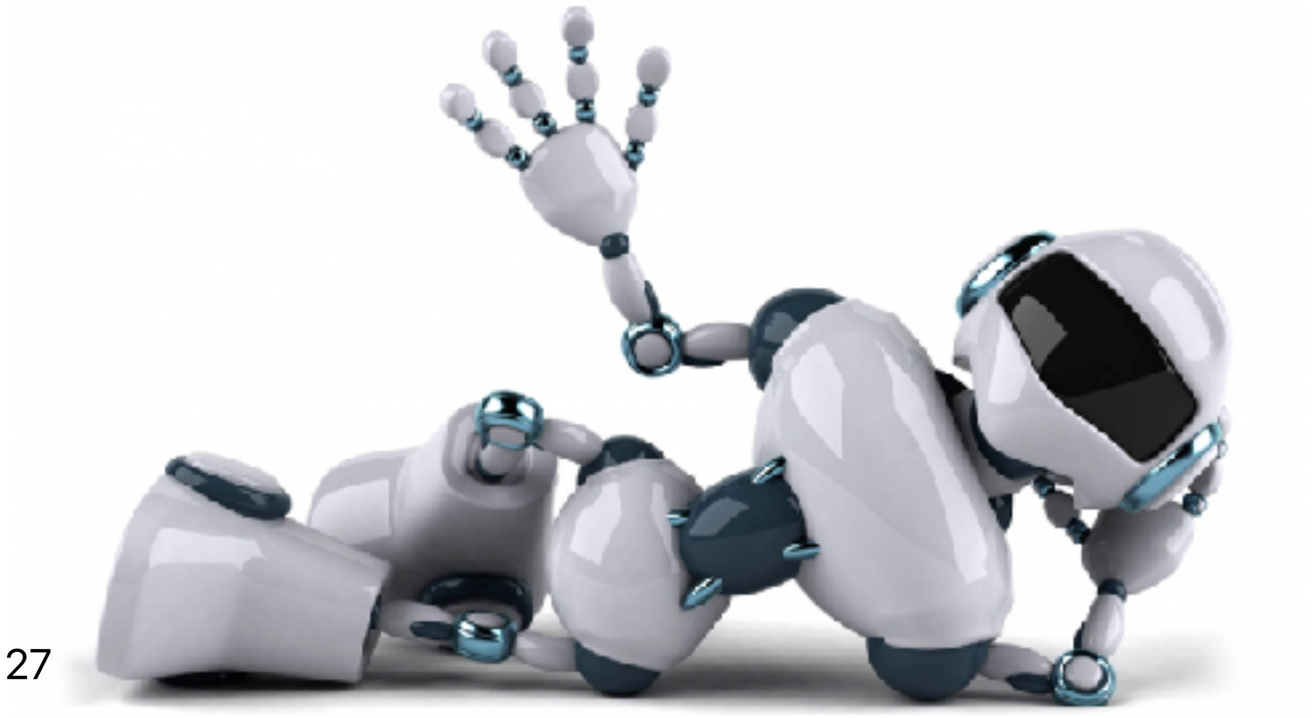
Guest OS

Virtual Machine Monitor

The Machine

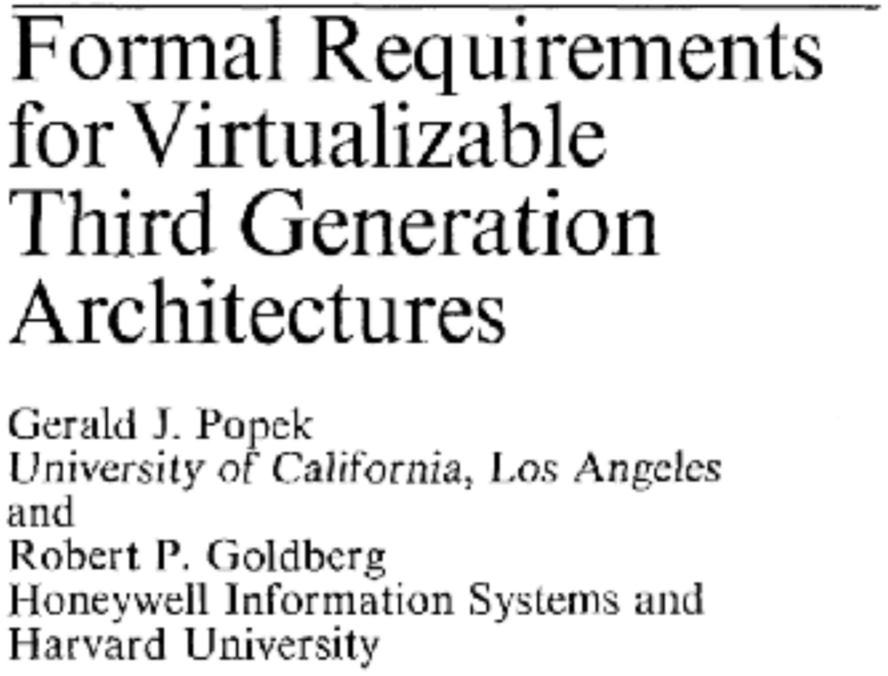
Three Laws of Robotics

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey orders given it by human beings except where such orders would conflict with the First Law.
- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.



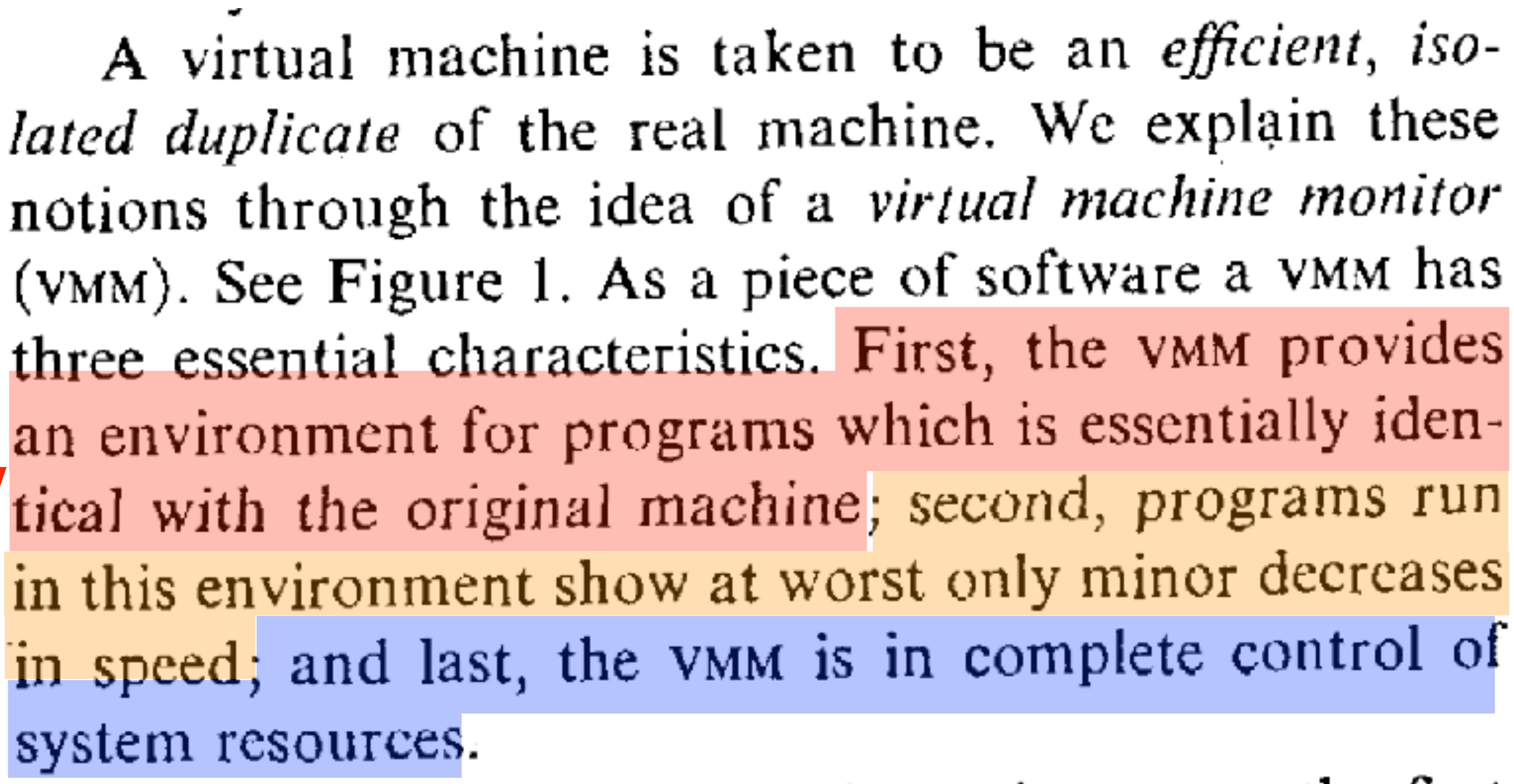
https://s3-ap-southeast-1.amazonaws.com/cloud-skcript/wp-content/uploads/2014/05/25090337/robots.jpg

Back to 1974...



Formal Requirements for Virtualizable Third Generation Architectures

Gerald J. Popek
University of California, Los Angeles
and
Robert P. Goldberg
Honeywell Information Systems and
Harvard University



A virtual machine is taken to be an *efficient, isolated duplicate* of the real machine. We explain these notions through the idea of a *virtual machine monitor* (VMM). See Figure 1. As a piece of software a VMM has three essential characteristics. First, the VMM provides an environment for programs which is essentially identical with the original machine; second, programs run in this environment show at worst only minor decreases in speed; and last, the VMM is in complete control of system resources.

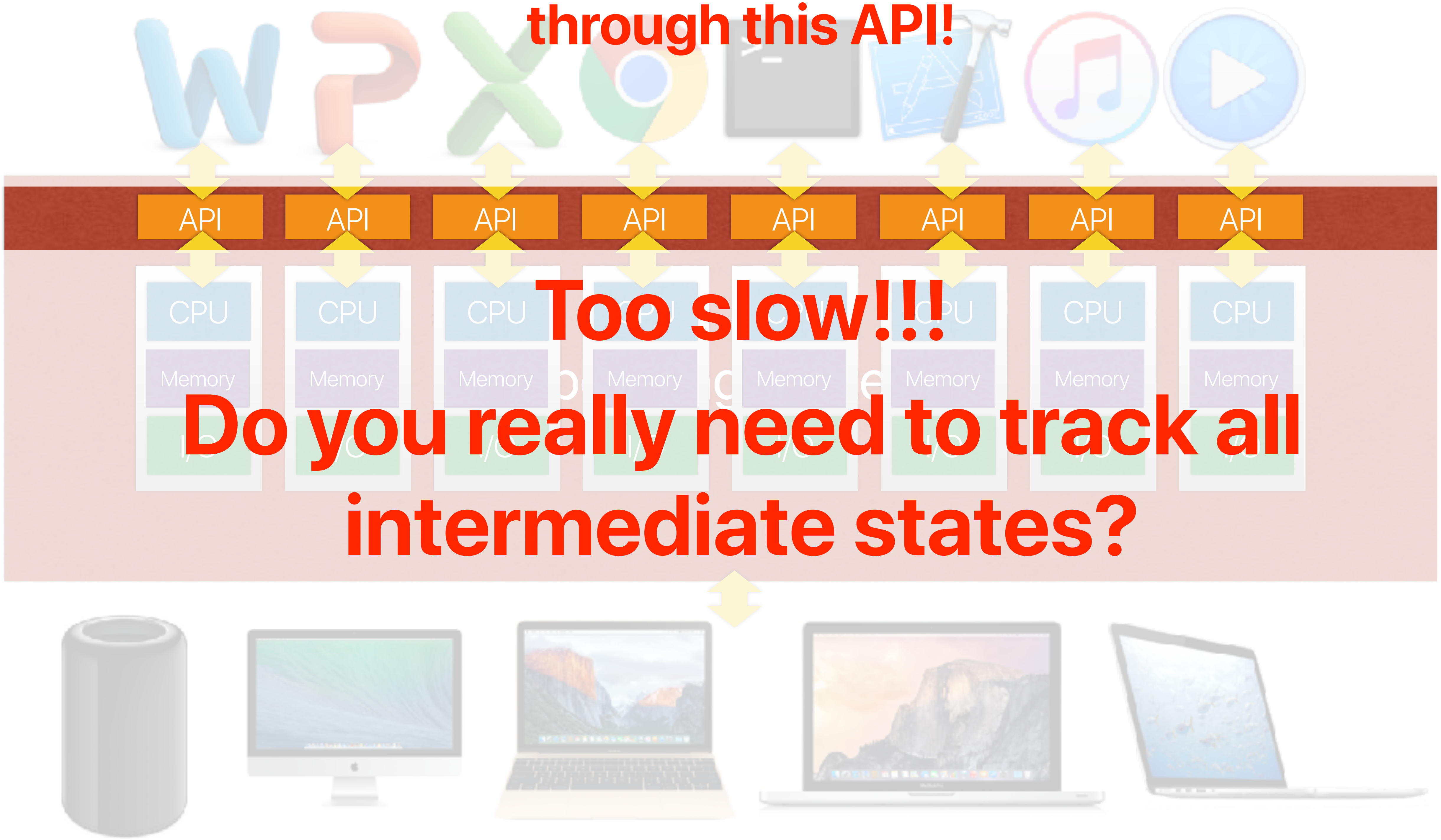
Fidelity

Performance

Safety and isolation

Recap: virtualization

However, we don't want everything to pass through this API!

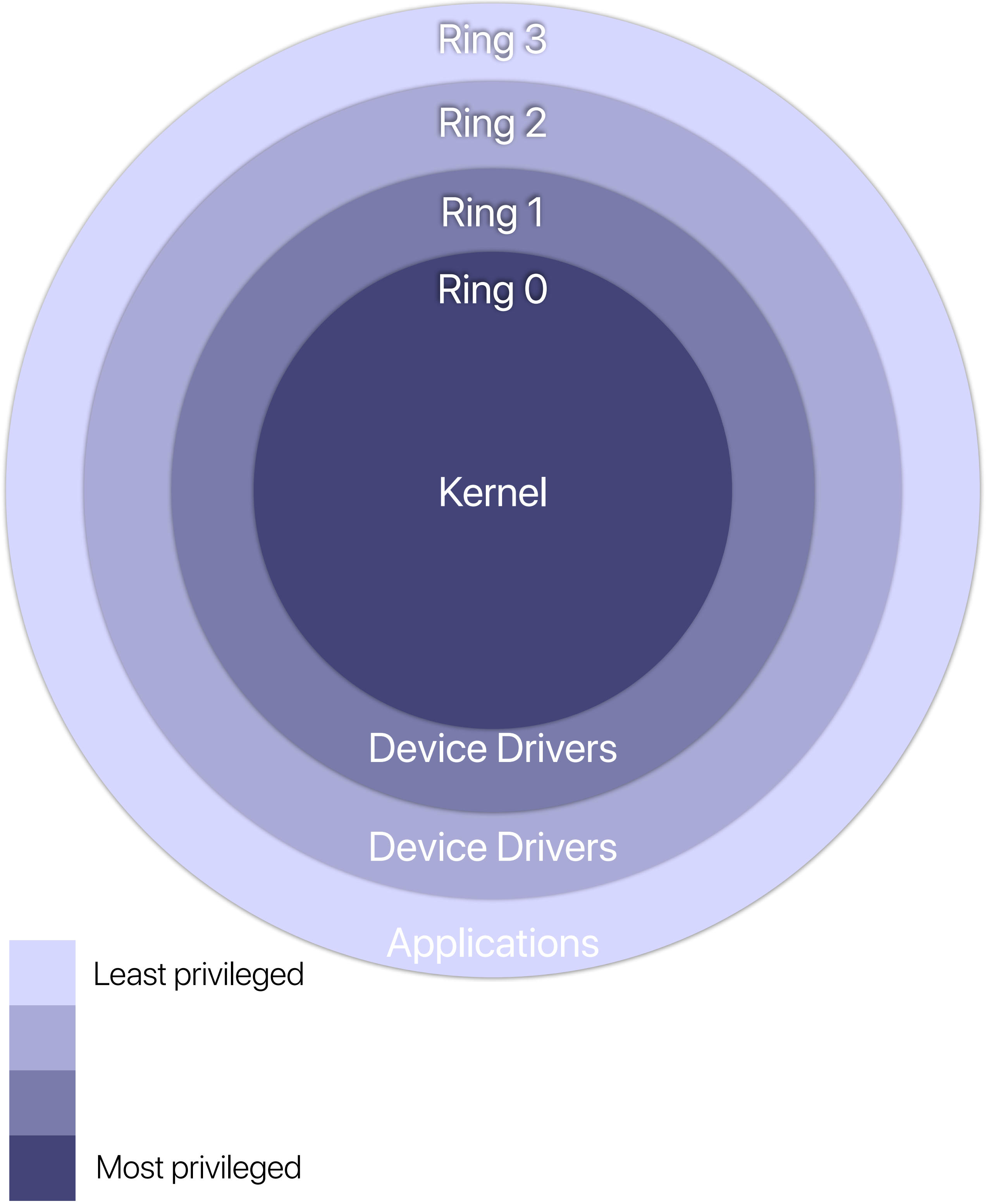


Too slow!!!

Do you really need to track all intermediate states?

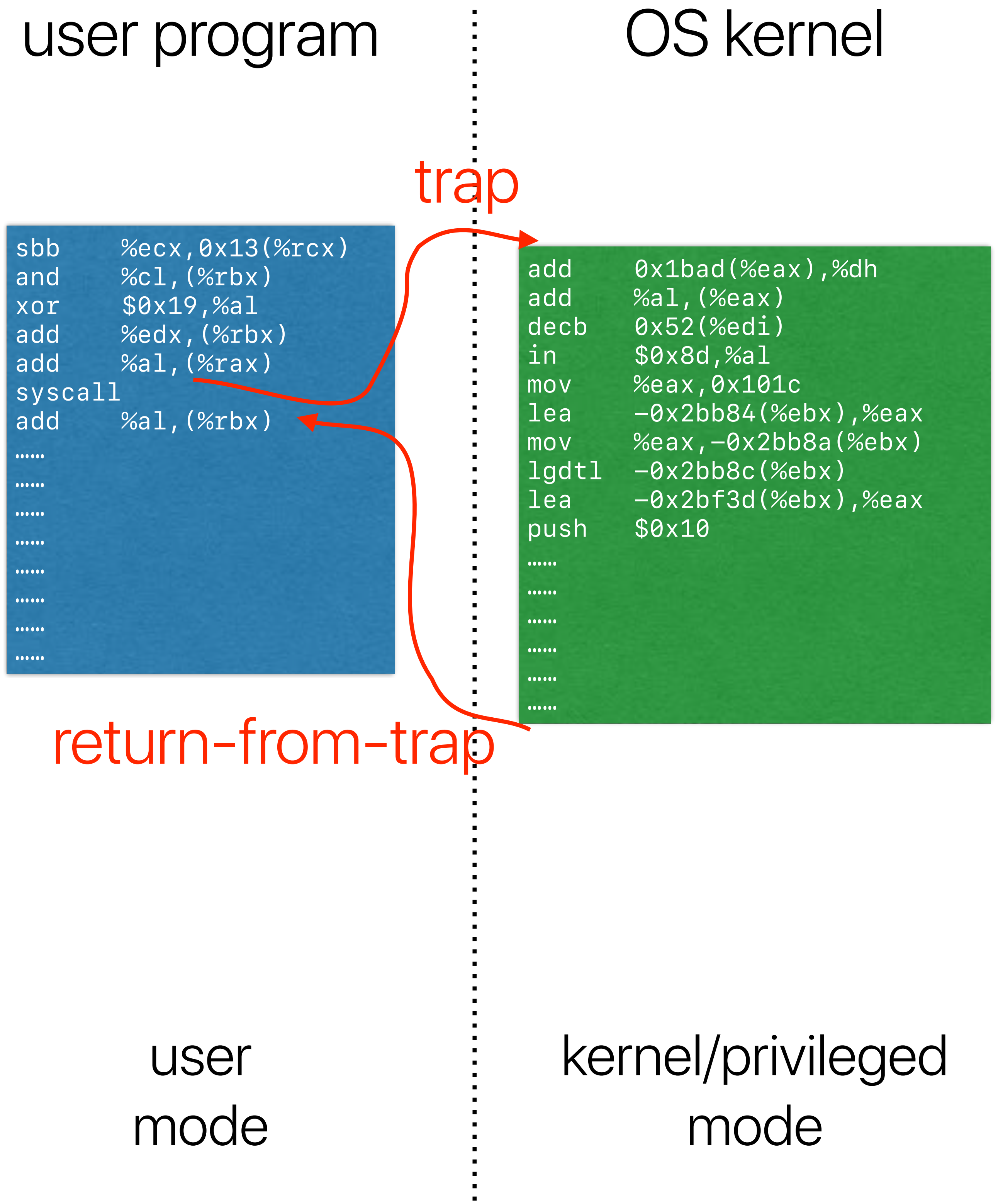
Recap: privileged instructions

- The processor provides **normal** instructions and **privileged** instructions
    - Normal instructions: ADD, SUB, MUL, and etc …
    - Privileged instructions: HLT, CLTS, LIDT, LMSW, SIDT, ARPL, and etc…
- The processor provides different modes
    - User processes can use normal instructions
    - Privileged instruction can only be used if the processor is in proper mode



Recap: How applications can use privileged operations?

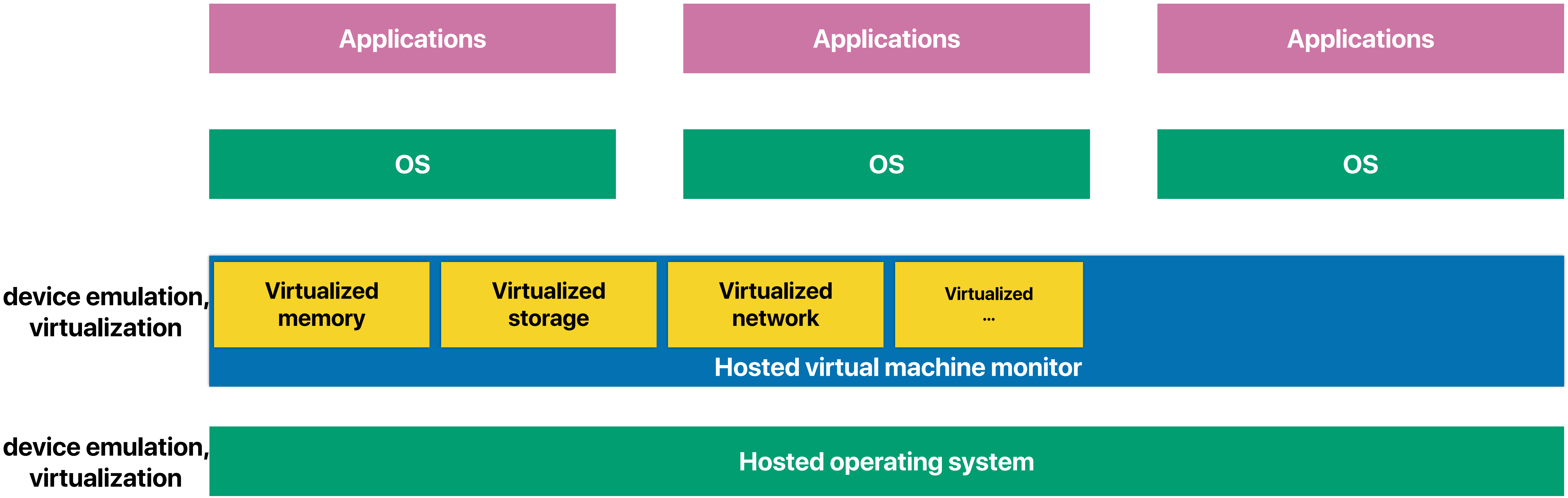
- Through the API: **System calls**
- Implemented in “trap” instructions
  - Raise an exception in the processor
  - The processor saves the exception PC and jumps to the corresponding exception handler in the OS kernel



```
sbb     %ecx,0x13(%rcx)
and     %cl,(%rbx)
xor     $0x19,%al
add     %edx,(%rbx)
add     %al,(%rax)
syscall
add     %al,(%rbx)
……
……
……
……
……
……
……
……
```

```
add     0x1bad(%eax),%dh
add     %al,(%eax)
decb    0x52(%edi)
in      $0x8d,%al
mov     %eax,0x101c
lea     -0x2bb84(%ebx),%eax
mov     %eax,-0x2bb8a(%ebx)
lgdtl   -0x2bb8c(%ebx)
lea     -0x2bf3d(%ebx),%eax
push    $0x10
……
……
……
……
……
……
```

Hosted virtual machine



Applications

Applications

Applications

OS

OS

OS

device emulation, virtualization

Virtualized memory

Virtualized storage

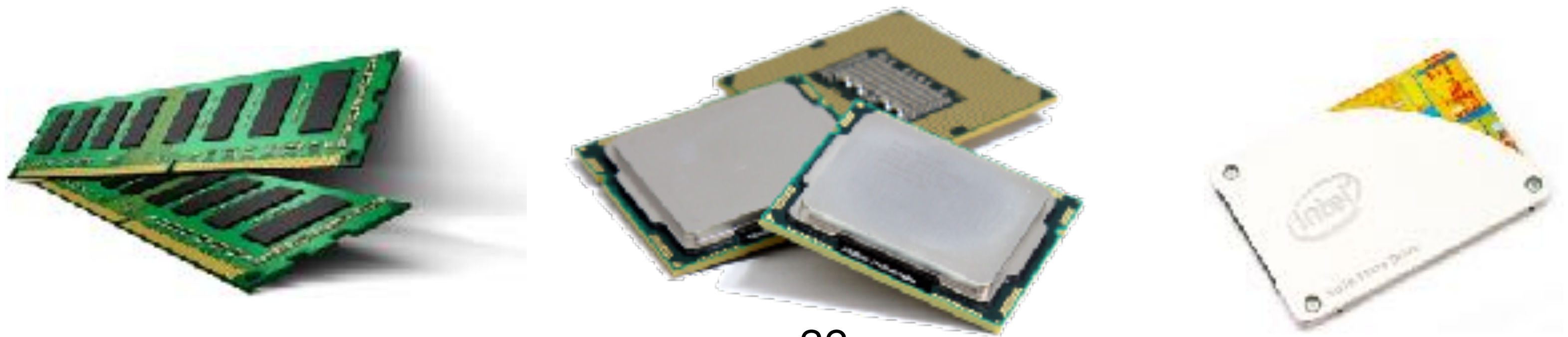
Virtualized network

Virtualized …

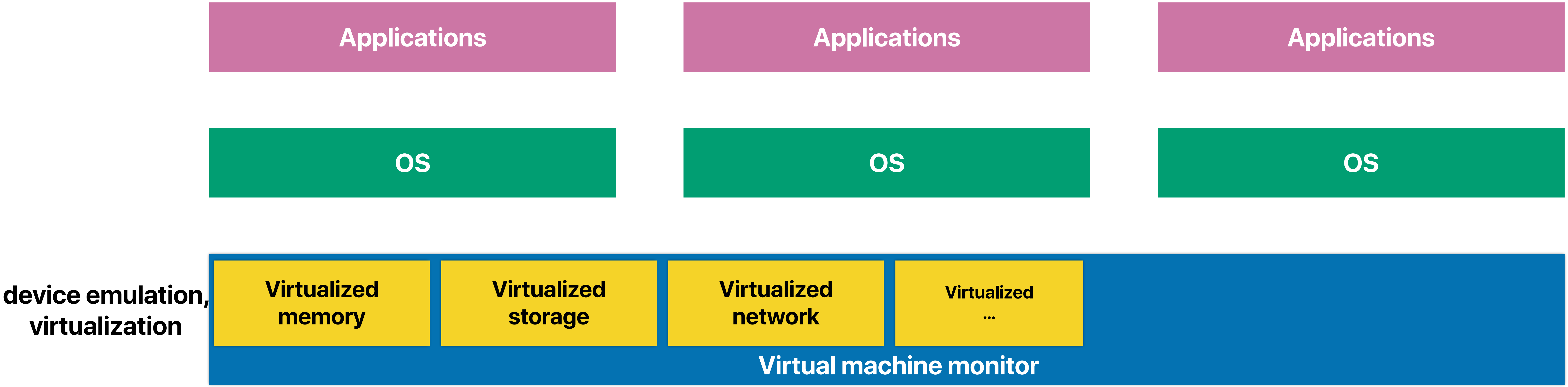
Hosted virtual machine monitor

device emulation, virtualization

Hosted operating system



Virtual machine monitors on bare machines



Applications

Applications

Applications

OS

OS

OS

device emulation, virtualization

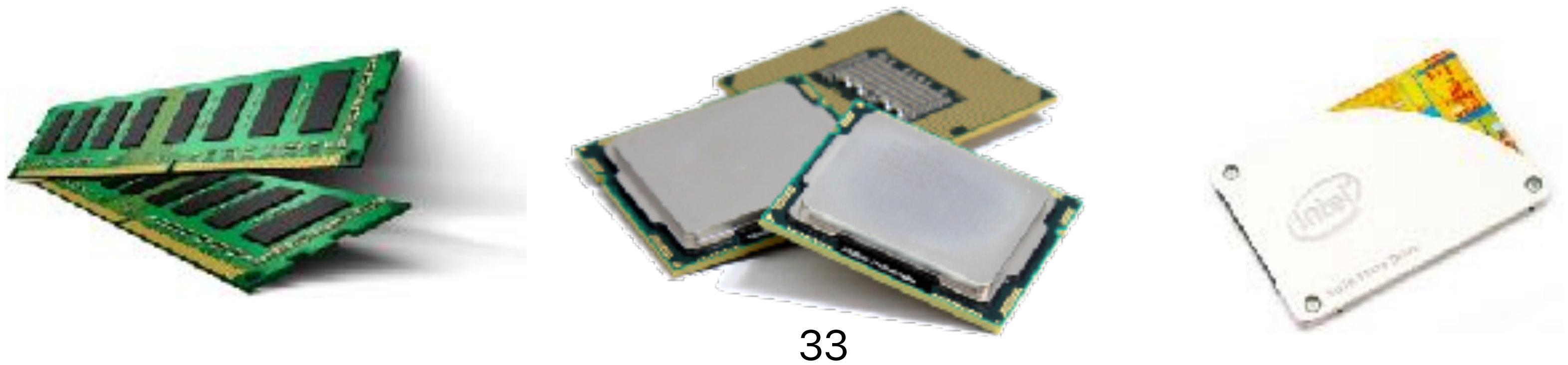
Virtualized memory

Virtualized storage

Virtualized network

Virtualized …

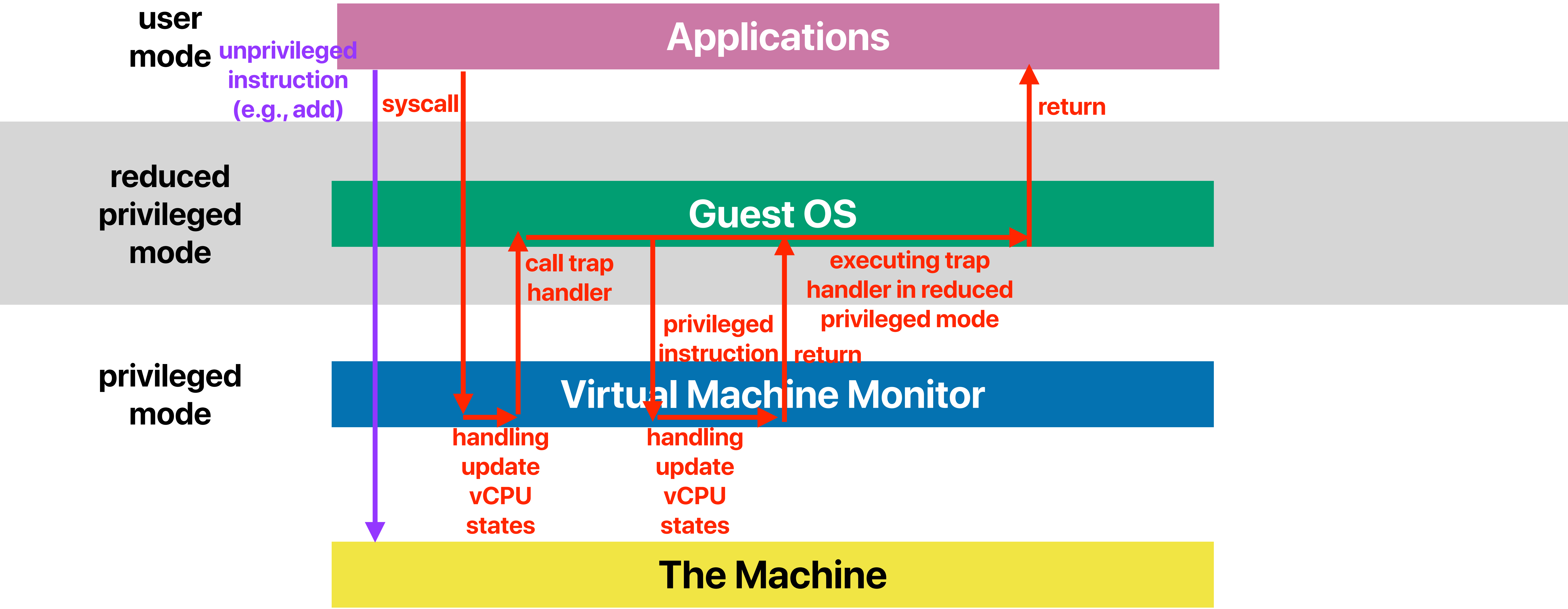
Virtual machine monitor



Three main ideas to classical VMs

- De-privileging
- Primary and shadow structures
- Tracing

CPU Virtualization: Trap-and-emulate



Announcement

- Group photo next lecture!
- Project revision
  - Allows you to revise your project with 30% of penalty on the unsatisfactory parts/test cases after the first-round of grading — say you got only 60% in the first-round, and you fixed everything before 3/11 — you can still get 60% +70%*40% = 88%
  - Please make an appointment with the TA through Google Calendar
- iEVAL — count as an extra, full-credit reading quiz, due 3/11
- Final — contains two parts (each account for 50%)
  - Part 1: 80 minute multiple choices/answers questions + two problem sets of comprehensive exam questions
  - Part 2: unlimited time between 3/11-3/17, open-ended questions

Computer
Science &
Engineering

202

つづく