6.033 Spring 2019

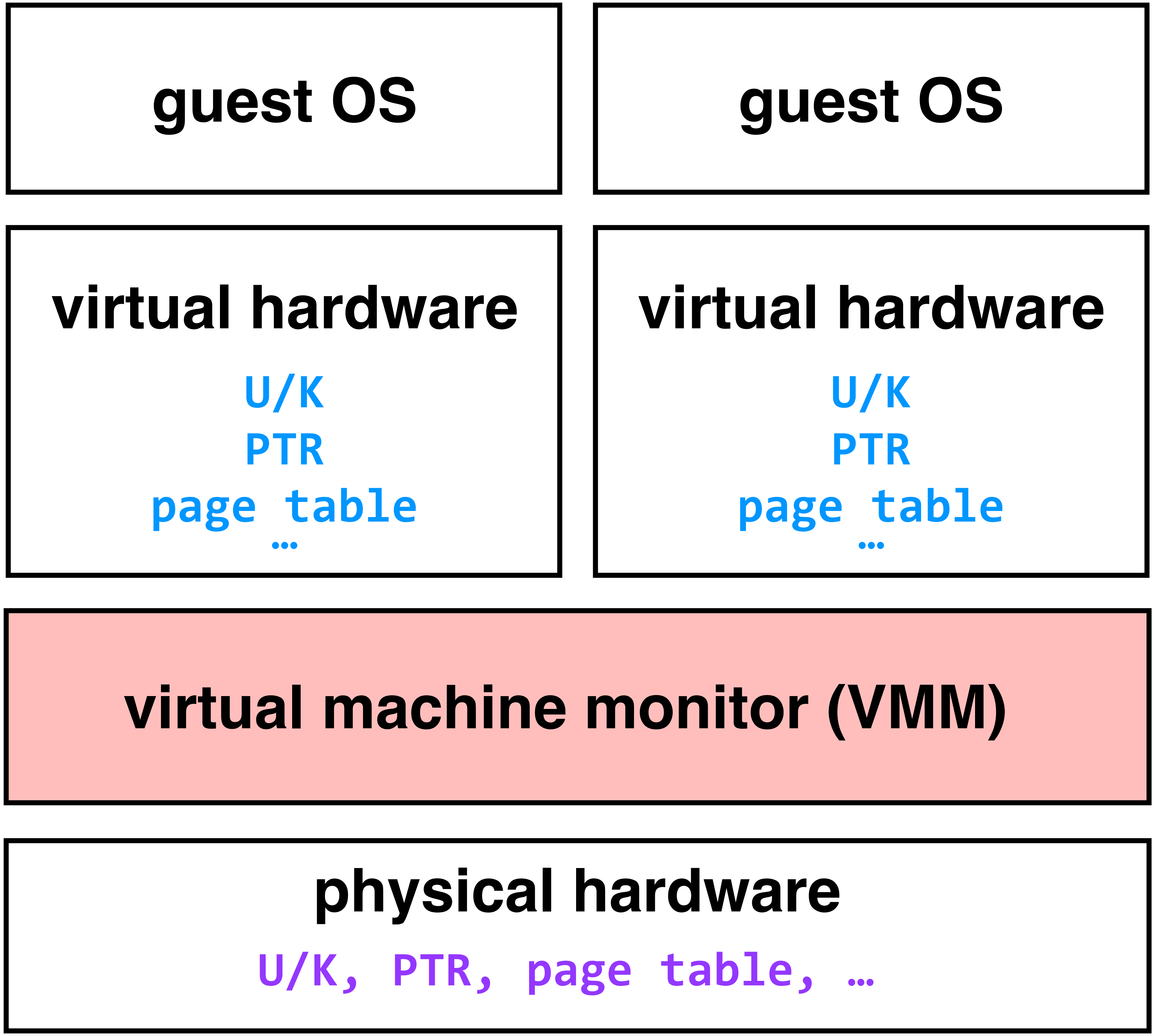
Lecture #7

- **Approaching Performance Problems**
- **General Performance-improvement Techniques**

operating systems enforce modularity on a single machine using **virtualization**

in order to enforce modularity + build an effective operating system

1. programs shouldn't be able to refer to (and corrupt) each others' **memory** → **virtual memory**
2. programs should be able to **communicate** → **bounded buffers** (virtualize communication links)
3. programs should be able to **share a CPU** without one program halting the progress of the others → **threads** (virtualize processors)



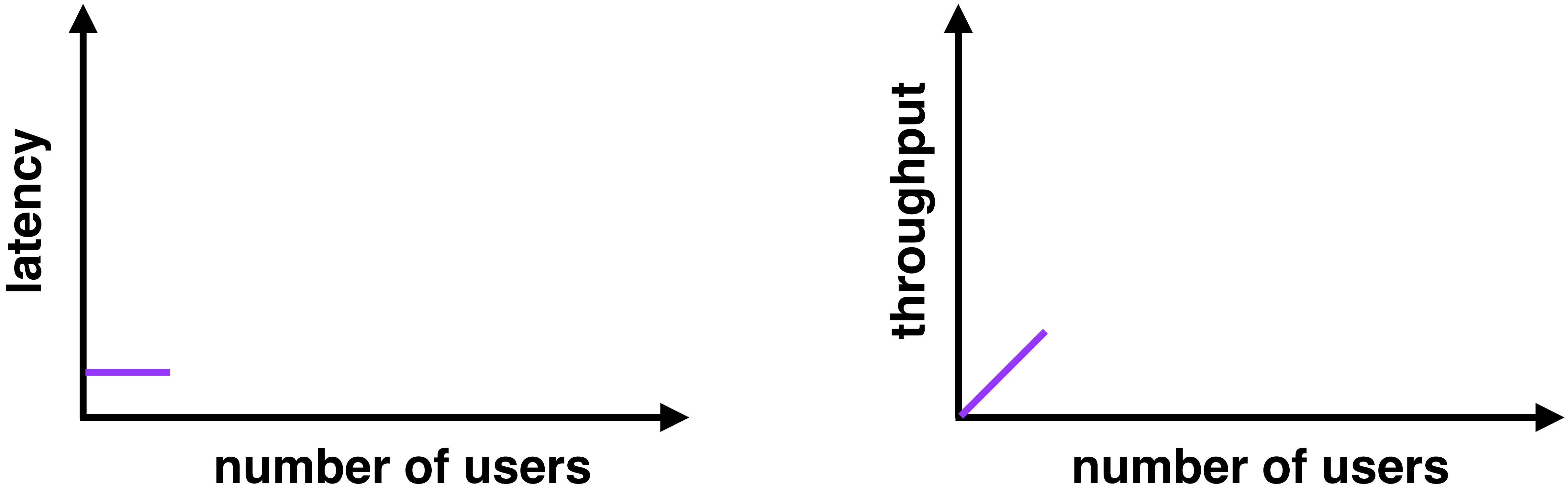
virtual machines: enforce modularity between multiple OSes running on the same physical machine

how do we get systems (operating or otherwise) to not just work, but to work well?

How to Improve Performance in Two Easy Steps

1. **measure** the system to find the bottleneck

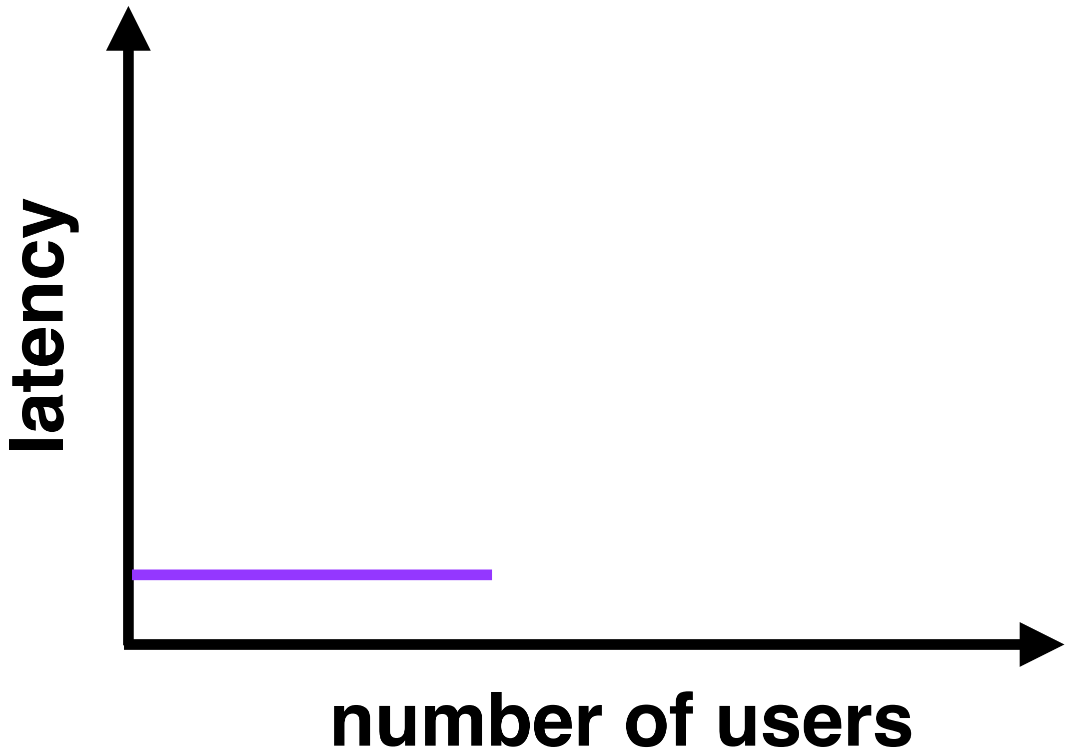
2. **relax** the bottleneck

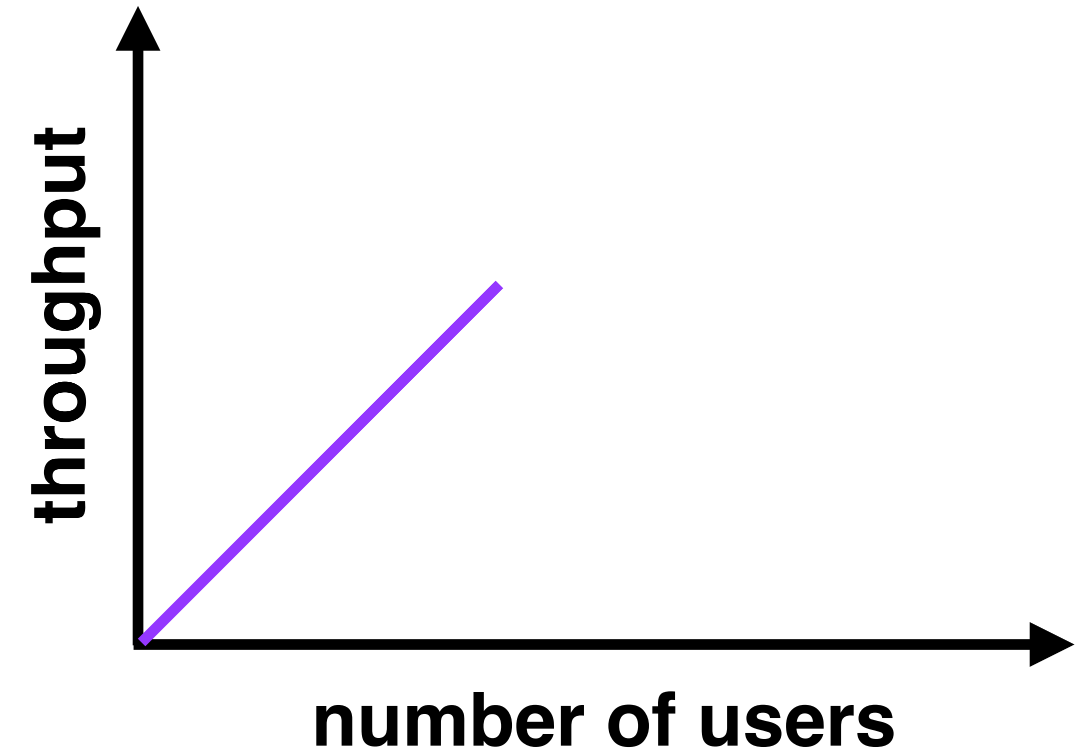


<u>few users</u>

low latency

low throughput (few users = few requests)

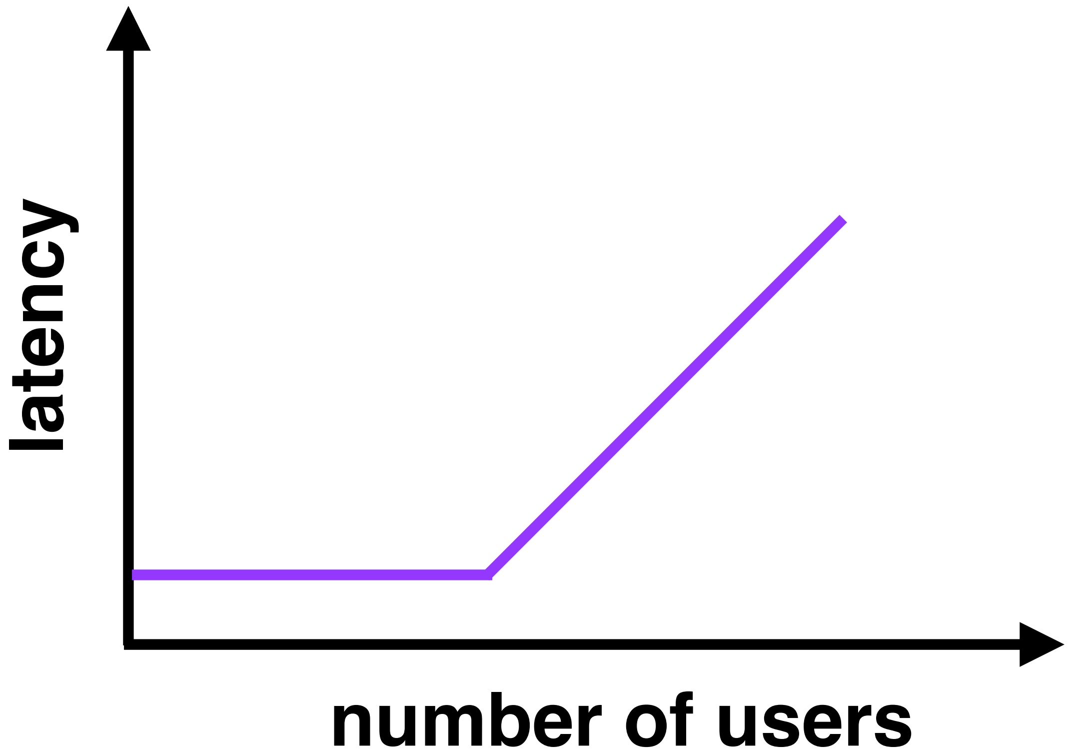


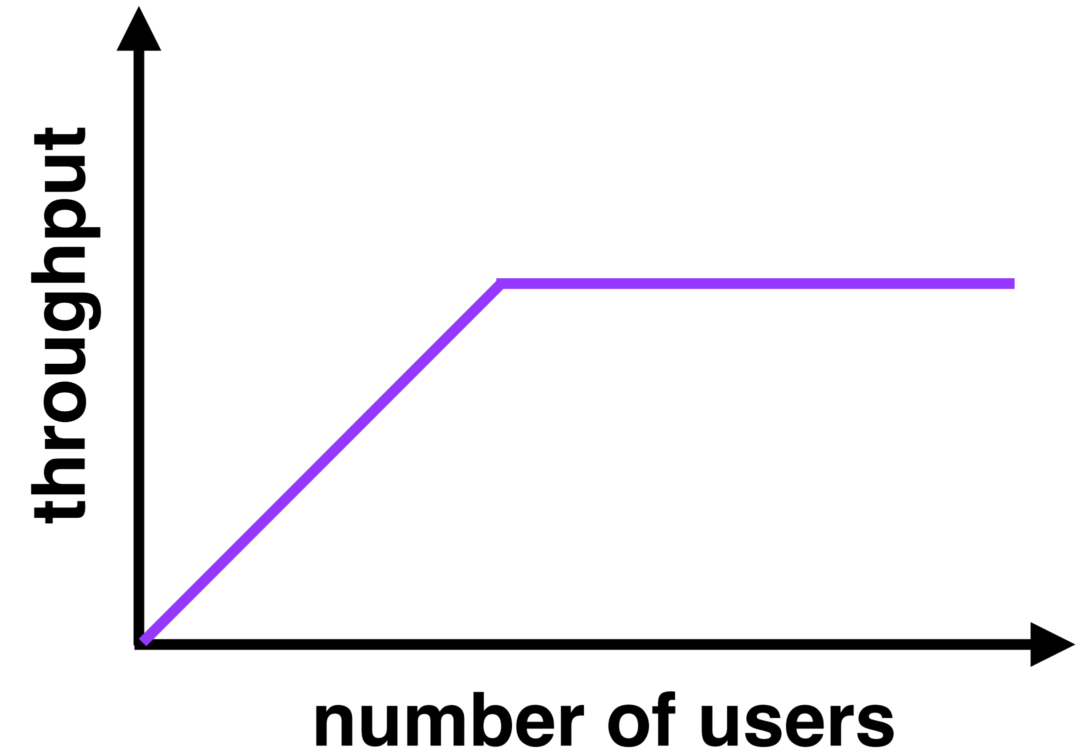


<u>moderate users</u>

low latency (new users consume previously idle resources)

high throughput (more users = more requests)





<u>**many users**</u>

high latency (requests queue up)

throughput plateaus (can’t serve requests any faster)

How to Improve Performance in Two Easy Steps

1. **measure** the system to find the bottleneck

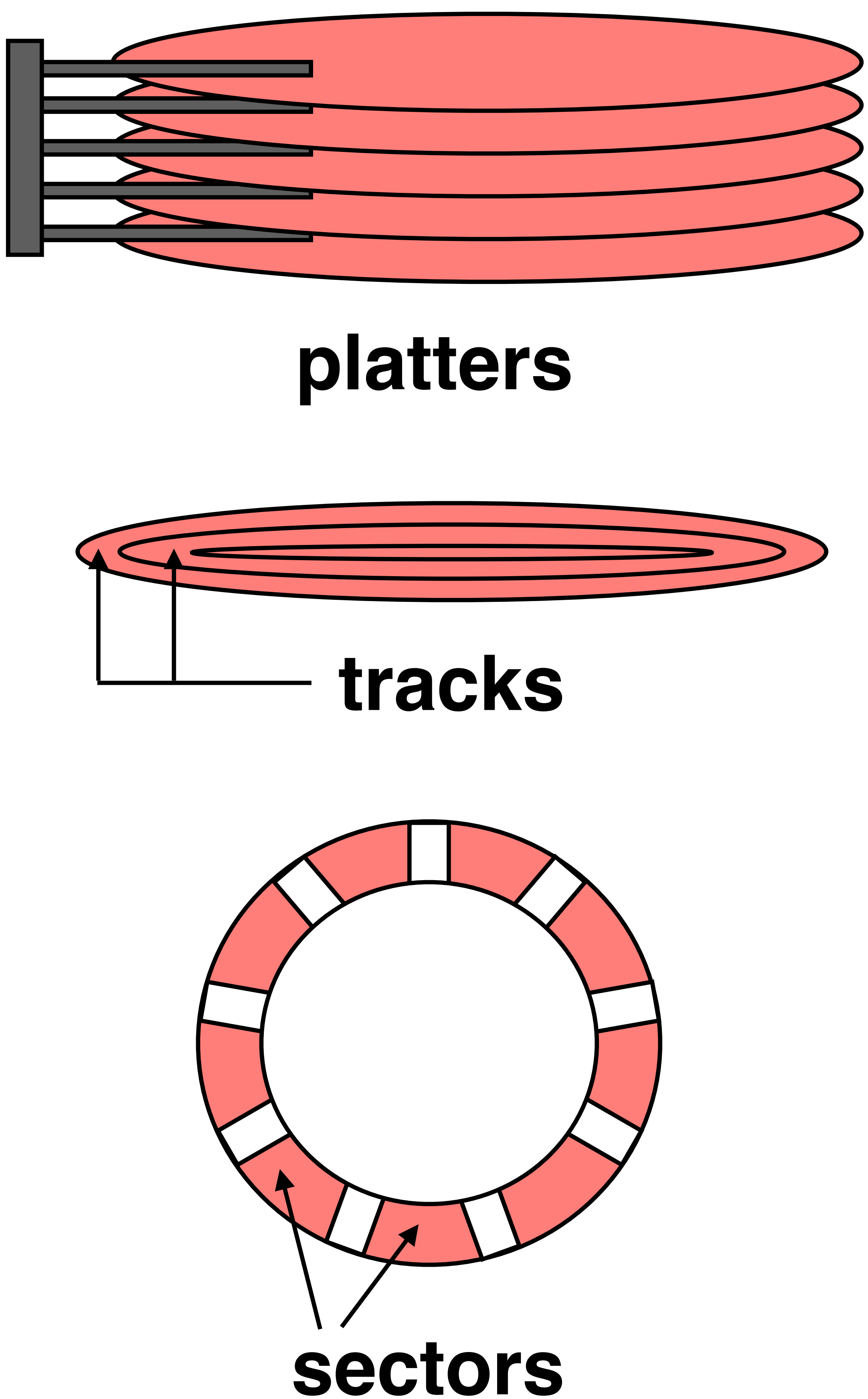
2. **relax** the bottleneck

How to Improve Performance in Two Easy Steps

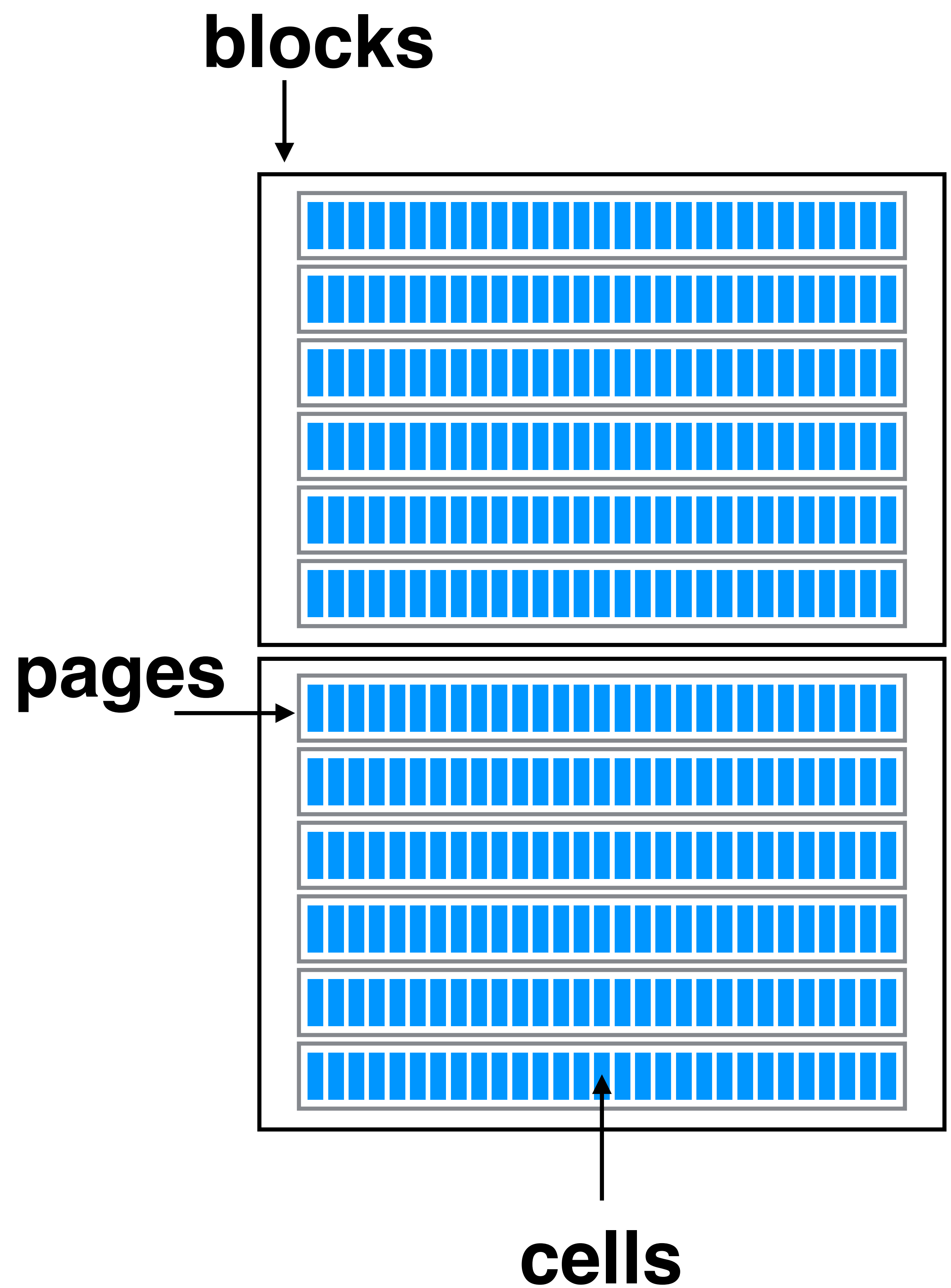
1. **measure** the system, and compare it to our system **model**, to find the bottleneck

2. **relax** the bottleneck

HDDs



SSDs



example disk specs (Hitachi 7K400)

capacity: 400GB
number of platters: 5
number of heads: 10
number of sectors per track: 567-1170
number of bytes per sector: 512
time for one revolution: 8.3ms
average read seek time: 8.2ms
average write seek time: 9.2ms

How to Improve Performance in Two Easy Steps

1. **measure the system to find the bottleneck**

2. **relax the bottleneck**
    - batch requests
    - cache data
    - exploit concurrency
    - exploit parallelism

- **Approaching Performance Problems**
  We approach performance problems in systems by **measuring** and **modeling** our system to find the bottleneck, and then **relaxing** (fixing) the bottleneck

- **Performance-improvement Techniques**
  Four common techniques to improve performance: **batching**, **caching**, **concurrency**, and **parallelism**. To be effective, all of these techniques require an understanding of how the underlying system works and is used