Introduction

Complexity VS. Simplicity

Yubin Xia

What is system? Why should I learn how to *think-in-system*?

SYSTEM: WHAT & WHY

What does "System" Mean?

- 1: Way to use hardware
  - Knowledge of OS, hardware, compiler, runtime, etc.
  - E.g., page table, TLB, file system
- 2: Way to think systematically
  - A way of analyzing and solving problems
  - E.g., system design principles

Why System?

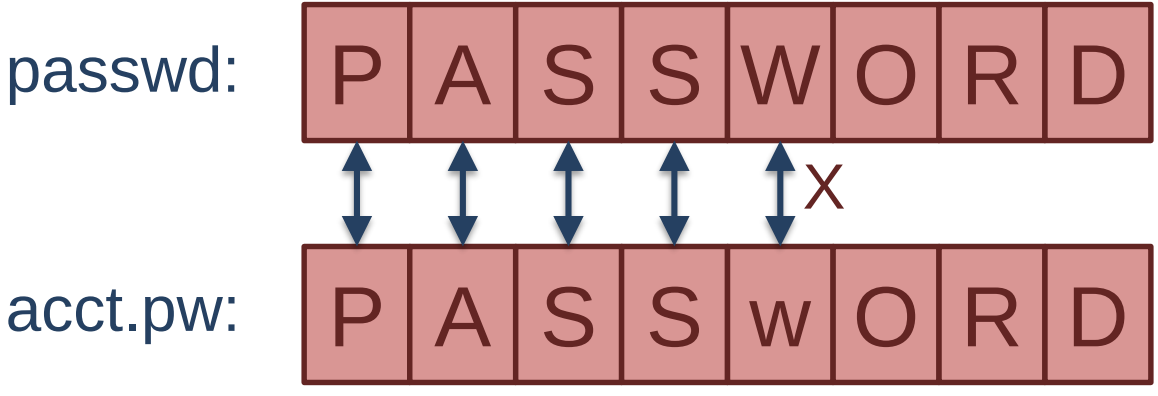
For fun and profit

Reason-1: For Fun (Finding Problems)

- Know a system by playing with it
  - The more you know about the system, the more likely to find interesting *features* (or, *bugs* with another name)

An Example: Guessing Password

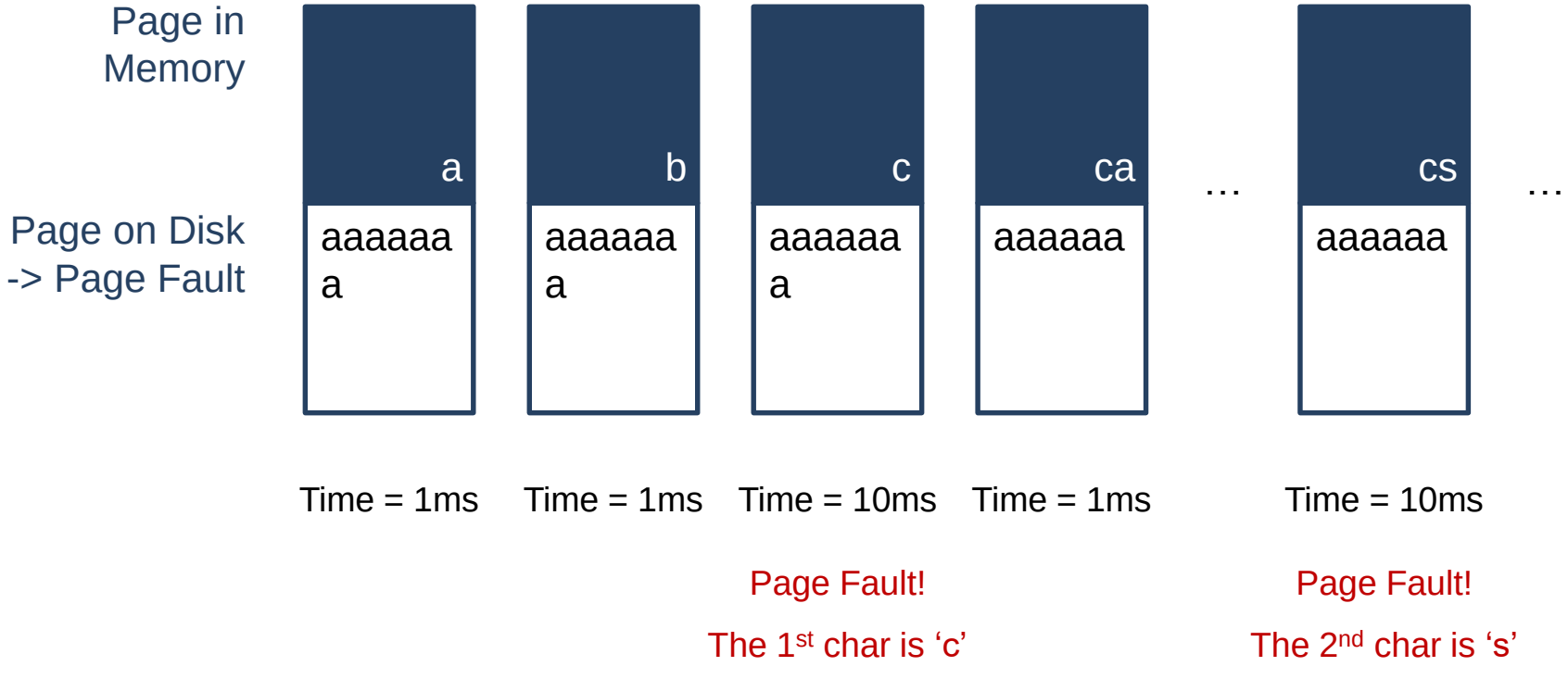
```
checkpw (user, passwd):
      acct = accounts[user]
      for i in range(0, len(acct.pw)):
          if acct.pw[i] ≠ passwd[i]:
            return False
      return True
```



Timing Attack: Guess One Character at a Time

- Guess a password
  - Allocate password 1 byte away from a page boundary
  - Page-fault means first char is OK
  - Example of cross-layer interactions
- Also known as “**Timing Attack**”

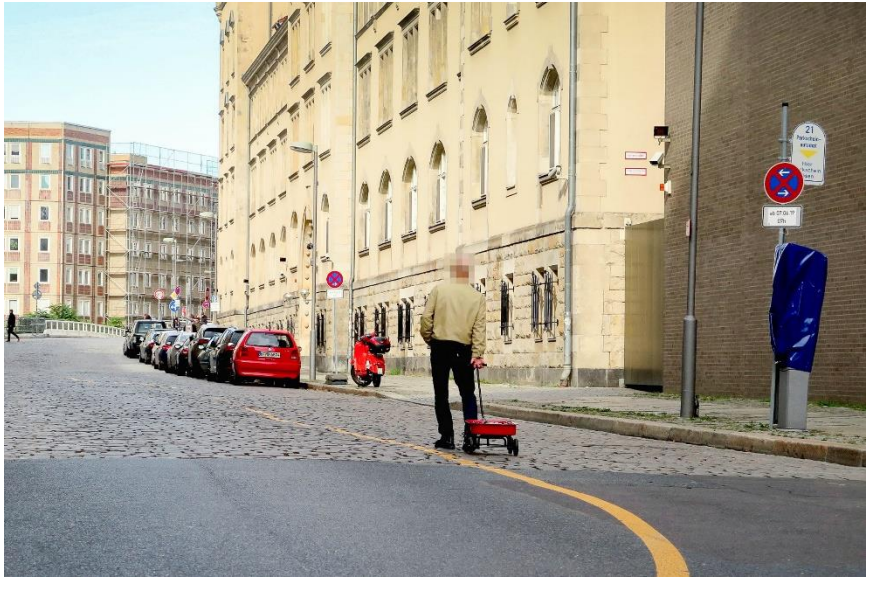
Timing Attack: Guess One Character at a Time

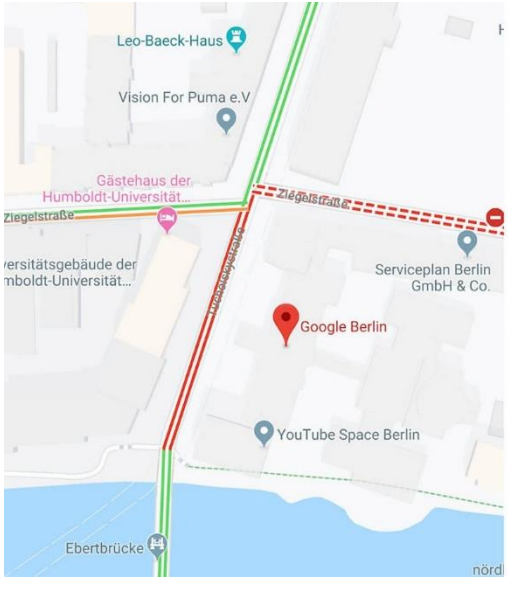


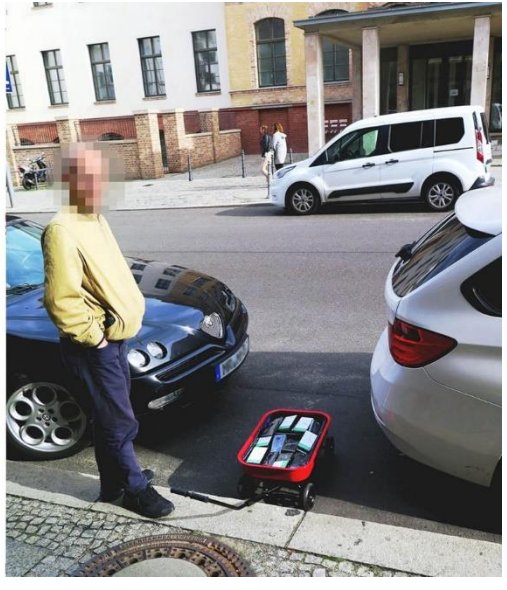
BRIAN BARRETT SECURITY 02.03.2020 01:44 PM

An Artist Used 99 Phones to Fake a Google Maps Traffic Jam

With his "Google Maps Hack," artist Simon Weckert draws attention to the systems we take for granted—and how we let them shape us.





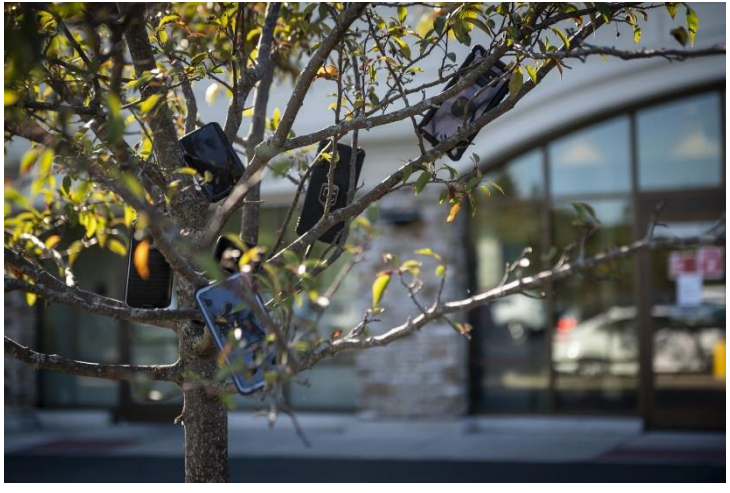


Technology

Amazon Drivers Are Hanging Smartphones in Trees to Get More Work

By Spencer Soper

Someone places several devices in a tree located close to the station where deliveries originate. Drivers in on the plot then sync their own phones with the ones in the tree and wait nearby for an order pickup. The reason for the odd placement, according to experts and people with direct knowledge of Amazon's operations, is to take advantage of the handsets' proximity to the station, combined with software that constantly monitors Amazon's dispatch network, to get a split-second jump on competing drivers.



Reason-2: For Profit (To Solve Problems)

- System is low-level, under other components
  - New features: migration, snapshot, replay, fault tolerant, etc.
  - Performance: hardware features, kernel primitives, system-level profiling, consistency model, etc.
  - Security: buffer overflow, ROP/JOP, side-channel, etc.

- Chance to have a start-up

THE PRINCIPLES

General Design Principles: 1/3

- Adopt sweeping simplifications
- Avoid excessive generality
- Avoid rarely used components
- Be explicit
- Decouple modules with indirection
- Design for iteration

General Design Principles: 2/3

- End-to-end argument
- Escalating complexity principle
- Incommensurate scaling rule
- Keep digging principle
- Law of diminishing returns
- Open design principle

General Design Principles: 3/3

- Principle of least astonishment
- Robustness principle
- Safety margin principle
- Unyielding foundations rule

Specific Area Principles (1/2)

- Atomicity: Golden rule of atomicity
- Coordination: One-writer principle
- Durability: The durability mantra
- Security: Minimize secrets
- Security: Complete mediation
- Security: Fail-safe defaults

Specific Area Principles (2/2)

- Security: Least privilege principle
- Security: Economy of mechanism
- Security: Minimize common mechanism

Design Hints

- Exploit brute force
- Instead of reducing latency, hide it
- Optimize for the common case
- Separate mechanism from policy

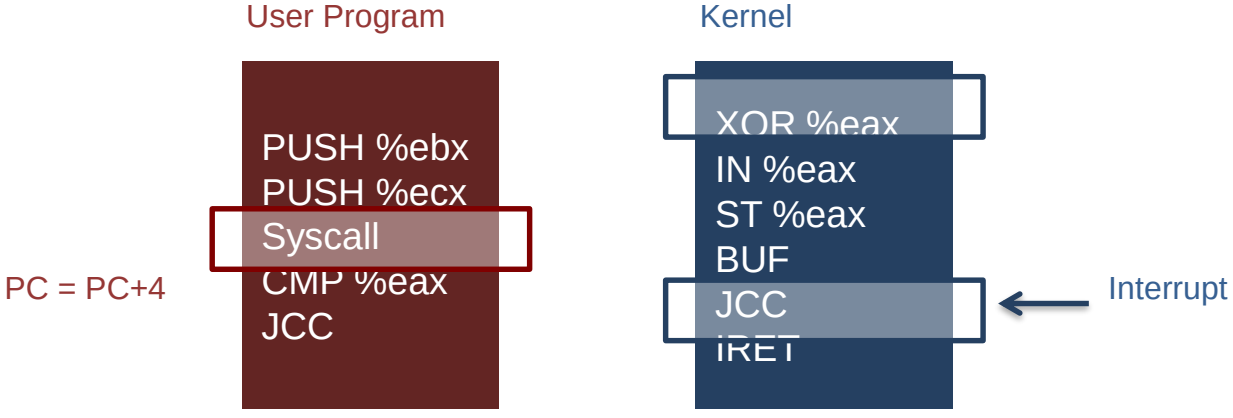
Richard P. Gabriel

WORSE IS BETTER

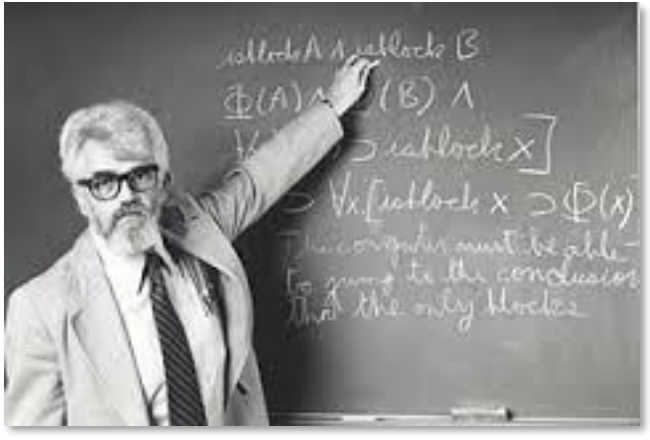
https://www.dreamsongs.com/RiseOfWorseIsBetter.html

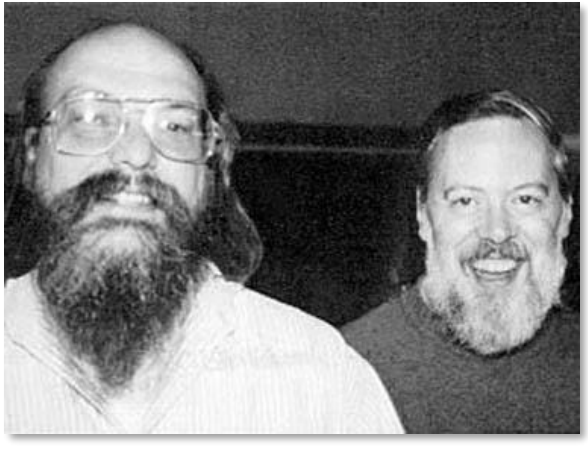
Worse is Better

- How to handle interrupt during a system-call
  - Must save user states: just a single PC is not enough
  - There are still states in the kernel, e.g., I/O buffers
  - Two choices: back out or press forward?



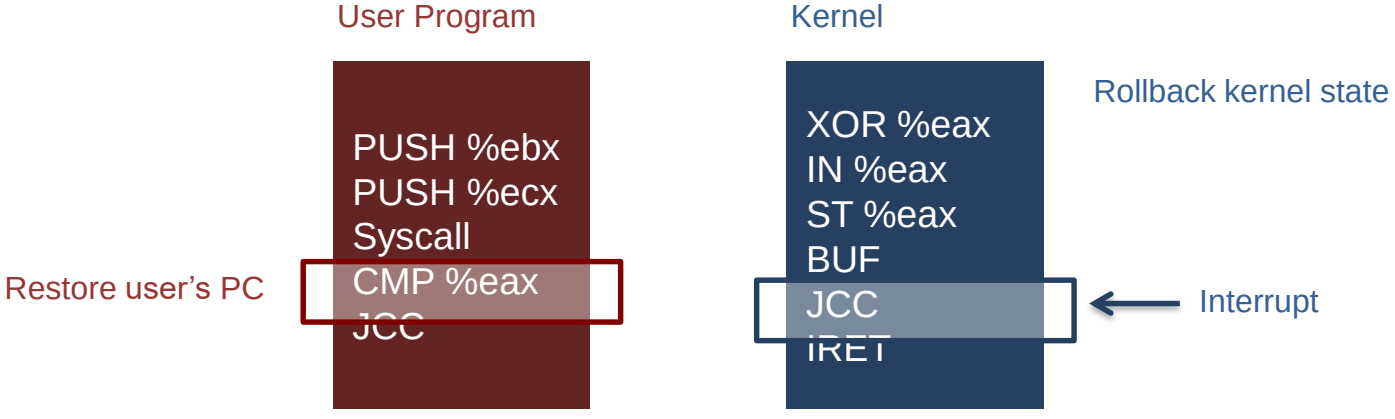
MIT Guys *VS.* New Jersey Guys (Bell Lab)





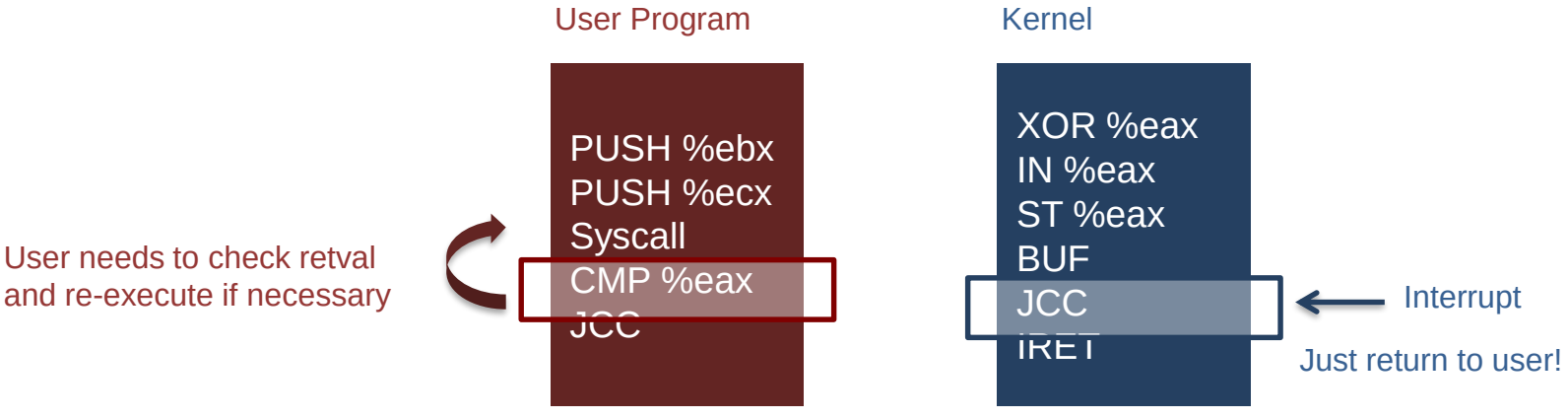
MIT Guys: We Do the Right Thing!

- Back out and restore the user program PC
- Re-execute the system call after resume
- Rollback all the kernel states, e.g., the buffer



New Jersey Guys: We Do the Simple Thing!

- Sys-calls always finish, but sometimes return an error
- The user needs to check the error code and retry in case
- Because the right thing is too complex, and happens rarely



Simplicity VS. The Right Thing

	MIT Guys	New Jersey Guys
Simplicity :	Simple implementation is more important than simple interface	Simple implementation is more important than simple interface
Correctness :	Incorrectness is not allowed	It's better to be simple than correct
Consistency :	Inconsistency is not allowed	It can be sacrificed for simplicity
Completeness :	Must cover as many as possible	It can be sacrificed for simplicity

MIT :	Correctness = Consistency > Completeness > Simplicity
New Jersey :	Simplicity > Anything else

MapReduce: Simplified Data Processing on Large Clusters, OSDI'04

CASE STUDY: MAP-REDUCE

What Did People Do Before MapReduce?

- A user wants to analyze big data (in TB)
  - He will think about distributing the task to 1000 machines
  - Split the data to 1,000 pieces for each machine
  - Send the data and code to 1,000 machines
  - Control the 1,000 machines to run
  - Collect the 1,000 outputs
  - Calculate the 1,000 outputs to get the final result
- Now, consider following scenarios:
  - Machine failures
  - Machine number scales to 10,000 (even more failures…)

MapReduce Provides:

- Automatic parallelization and distribution
- Fault-tolerance
- I/O scheduling
- Status and monitoring

- And a simple interface to use: map() / reduce()

Programming Model

- map (in_key, in_value) -> list(out_key, intermediate_value)
  - Processes input key/value pair
  - Produces set of intermediate pairs

- reduce (out_key, list(intermediate_value)) -> list(out_value)
  - Combines all intermediate values for a particular key
  - Produces a set of merged output values (usually just one)

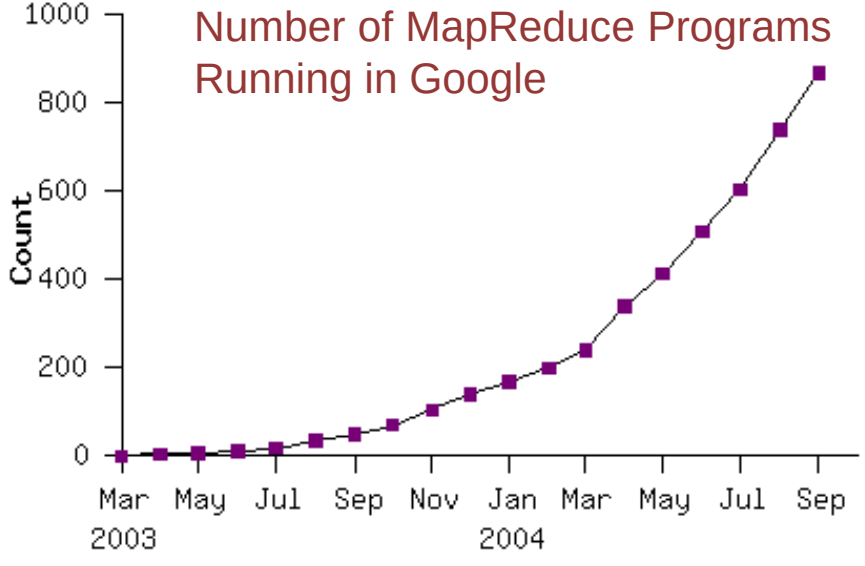
- *Inspired by similar primitives in **LISP** and other languages*

Example: Count Word Occurrences

```
1  map(String input_key, String input_value) {
2    // input_key: document name
3    // input_value: document contents
4    for each word w in input_value:
5      EmitIntermediate(w, "1");
6  }
7
8
9  reduce(String output_key, Iterator intermediate_values) {
10   // output_key: a word
11   // output_values: a list of counts
12   int result = 0;
13   for each v in intermediate_values:
14     result += ParseInt(v);
15   Emit(AsString(result));
16 }
```

Model is Widely Applicable

- Example uses: Distributed grep, distributed sort, web link-graph reversal, term-vector per host, web access log states, inverted index construction, document clustering, machine learning, statistical machine translation, …

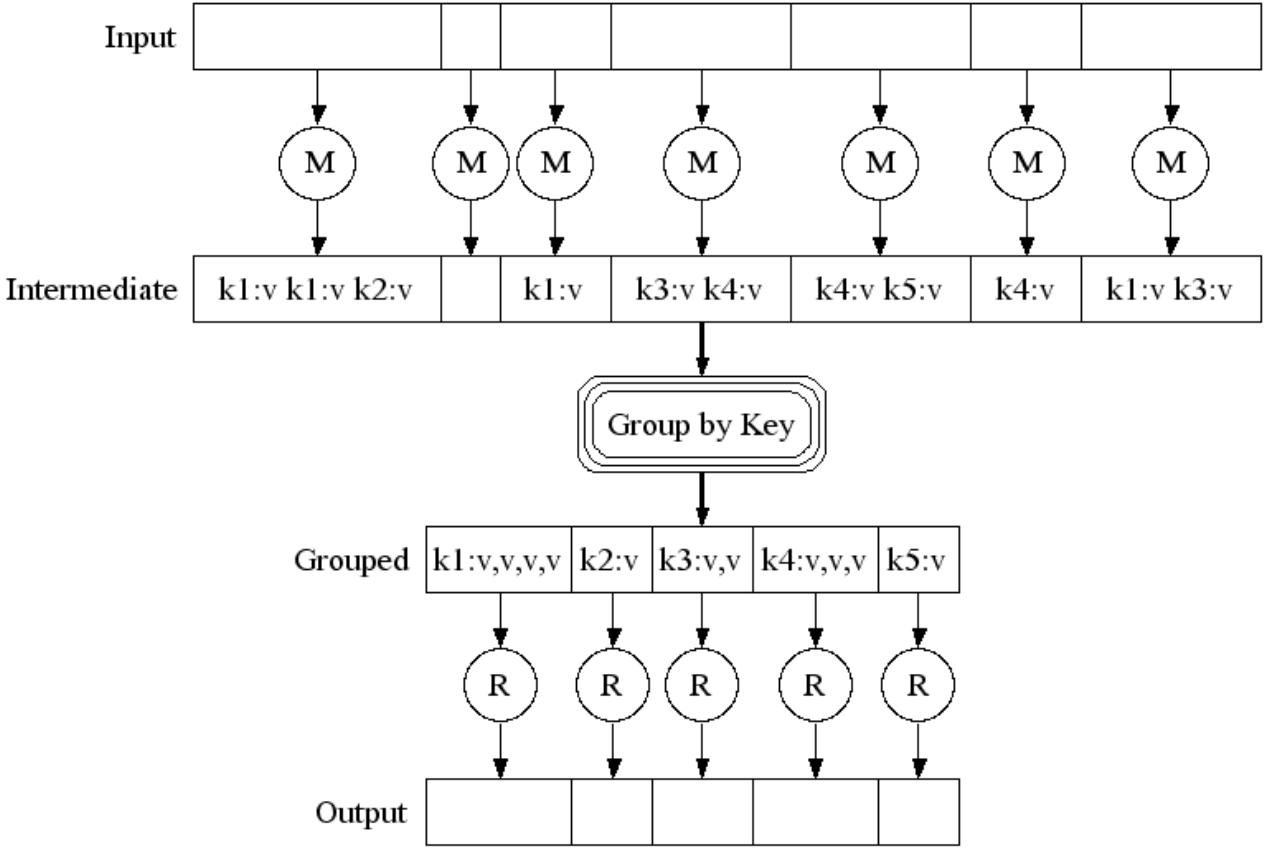


Implementation Overview

- Thousands of PC: 2-CPU, 2-4 GB memory
- Limited bisection bandwidth
- Storage is on local IDE disk (cheap, but likely to fail)
- GFS: distributed FS managers data (SOSP’03)
- Job scheduling system

- Implemented as a C++ library, linked to user’s apps

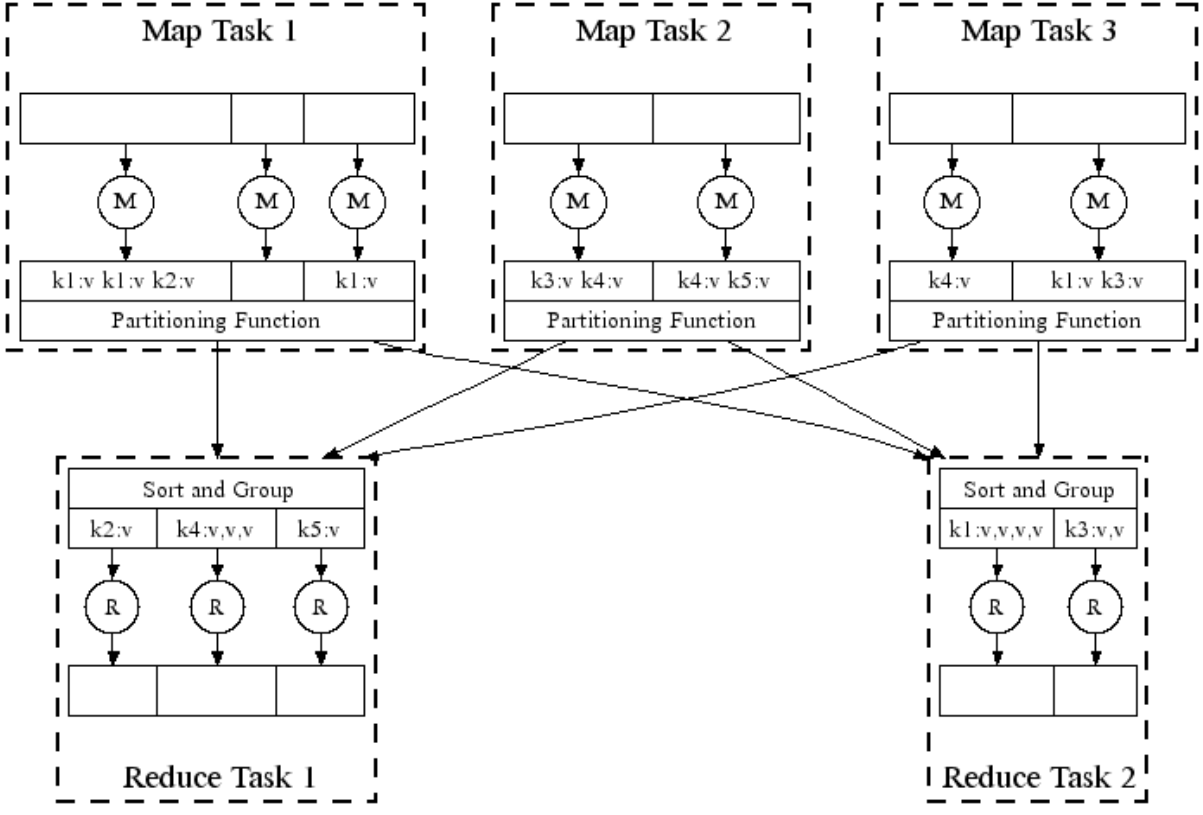
Execution


Input
M
M
M
M
M
M
M
Intermediate
k1:v k1:v k2:v
k1:v
k3:v k4:v
k4:v k5:v
k4:v
k1:v k3:v
Group by Key
Grouped
k1:v,v,v,v
k2:v
k3:v,v
k4:v,v,v
k5:v
R
R
R
R
R
Output

Architecture

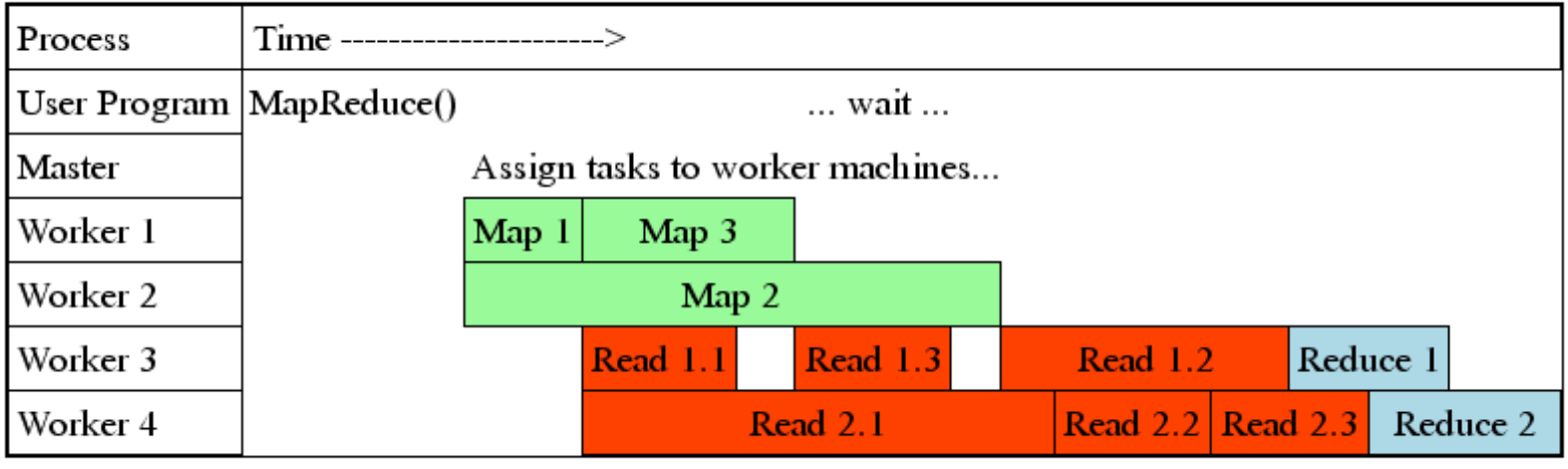
- Master node
- Worker node

Parallel Execution



Task Granularity and Pipelining

- Fine granularity tasks: many more map tasks than machines
  - Minimizes time for fault recovery
  - Can pipeline shuffling with map execution
  - Better dynamic load balancing



Fault Tolerance: Handled via Re-execution

- On worker failure:
  - Detect failure via periodic heartbeats
  - Re-execute completed and in-progress map tasks
  - Re-execute in progress reduce tasks
  - Task completion committed through master
- Master failure:
  - Could handle, but not yet (master failure unlikely)

- Robust: lost 1,600 of 1,800 machines once, but finished fine

Refinement: Redundant Execution

- Slow workers significantly lengthen completion time
  - Other jobs consuming resources on machine
  - Bad disks with soft errors transfer data very slowly
  - Weird things: processor caches disabled (!!)
- Solution: Near end of phase, spawn backup copies of tasks
  - Whichever one finishes first "wins"
- Effect: Dramatically shortens job completion time

Refinement: Locality Optimization

- Master scheduling policy:
  - Asks GFS for locations of replicas of input file blocks
  - Map tasks typically split into 64MB (== GFS block size)
  - Map tasks scheduled so GFS input block replica are on same machine or same rack
- Effect: 1,000s of machines read input at local disk speed
  - Without this, rack switches limit read rate

Refinement: Skipping Bad Records

- Map/Reduce functions sometimes fail for particular inputs
  - Best solution is to debug & fix, but not always possible
  - On seg fault:
    - Send UDP packet to master from signal handler
    - Include sequence number of record being processed
  - If master sees two failures for same record:
    - Next worker is told to skip the record
- Effect: Can work around bugs in third-party libraries

Other Refinements (see paper)

- Sorting guarantees within each reduce partition
- Compression of intermediate data
- Combiner: useful for saving network bandwidth
- Local execution for debugging/testing
- User-defined counters

Conclusions

- MapReduce has proven to be a useful abstraction
- Greatly simplifies large-scale computations at Google
- Fun to use: focus on problem, let library deal with messy details

Principles in MapReduce

- Principle of least astonishment
  - The user only needs to define map/reduce functions
- Optimize for the common case
  - List the most common scenarios, not all of them
- Robustness principle

Criticism: “MapReduce: A Major Step Backwards”

Prof. David DeWitt

Wisconsin University

Michael Stonebraker

CTO of Informix
Professor in MIT
Professor in UC Berkeley

Criticism: “MapReduce: A Major Step Backwards”

- A giant step backward in the programming paradigm for large-scale data intensive applications
- A sub-optimal implementation, in that it uses brute force instead of indexing
- Not novel at all — it represents a specific implementation of well known techniques developed nearly 25 years ago
- Missing most of the features that are routinely included in current DBMS
- Incompatible with all of the tools DBMS users have come to depend on

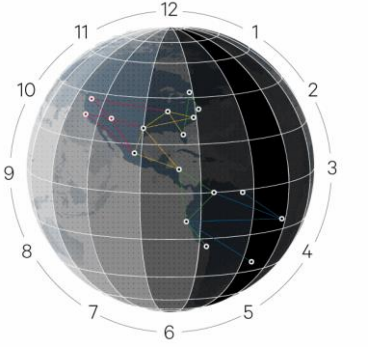
- Well, these are all “the right things”, but…

After MapReduce

- Hadoop
  - An open-source implementation of MapReduce
  - Again, very bad performance at first
  - But widely used!
    - Cassandra, Hbase, Hive, PIG, Spark, ZooKeeper, …
  - Keep being optimized by both academy and industry
    - Till today

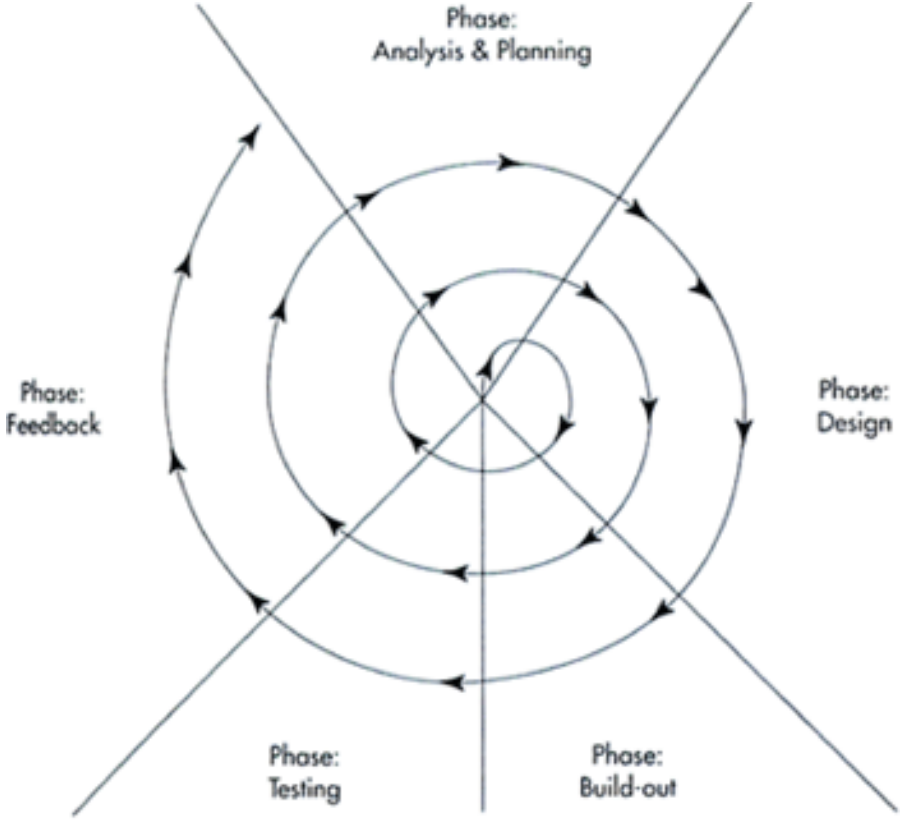
After MapReduce: Google

- MapReduce retired around 2007
- 2010: Pregel [SIGMOD'10] for graphic computing
- 2012: Spanner [OSDI'12]
  - Global distributed database using *TrueTime* API
- 2014: DataFlow
  - Map-Reduce is retired in Google



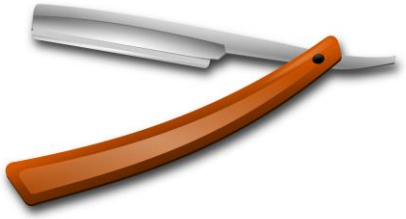
Another Principle

- Design for Iteration
- It’s hard, if possible, to do things right in the 1st time



SO, HOW TO BE MORE “KISS”?

Occam's Razor



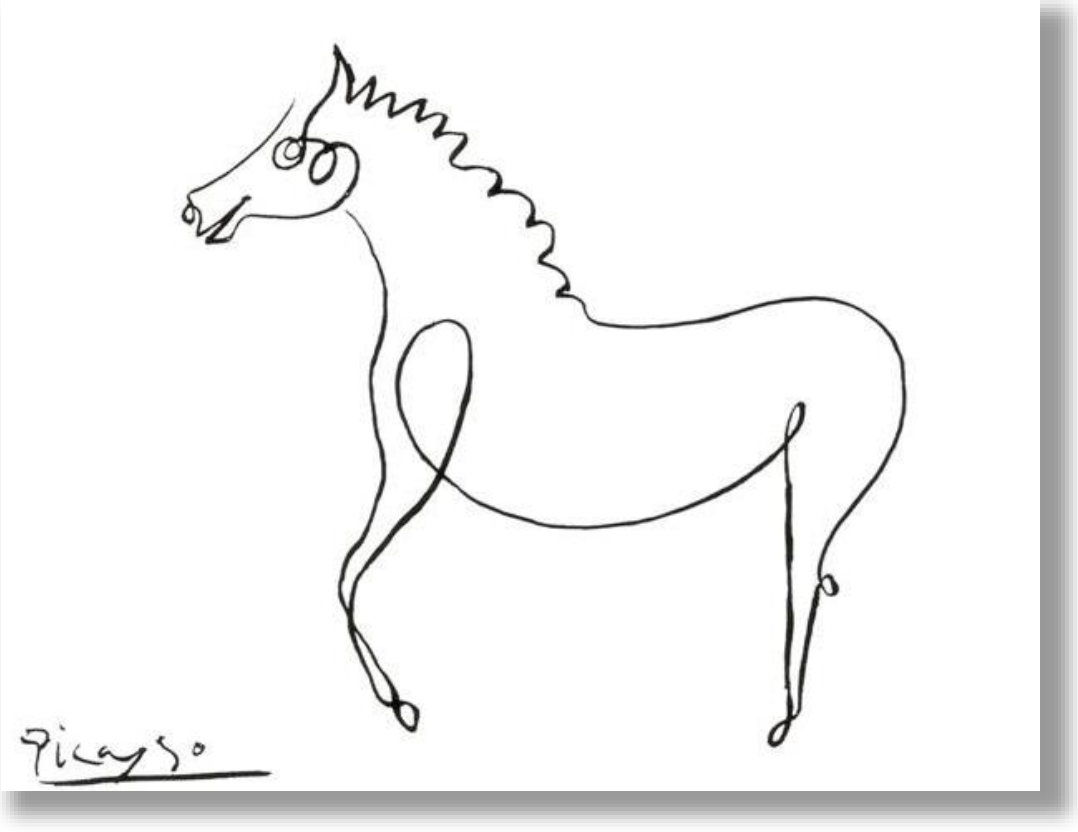
- Aka., law of parsimony
- Now, chose between following two equations:

$$(x-2)^2(y+2x-8)^2(y-2x+4)^2(x-4)^2(y-2x+8)^2(y+2x-16)^2\left(\left(y-3\right)^2+\left|x-\tfrac{11}{2}\right|+\left|x-\tfrac{13}{2}\right|-1\right)^2$$
$$\cdot\left(\left(y-x+5\right)^2+|x-8|+|x-9|-1\right)^2\left(\left(y+x-11\right)^2+|x-8|+|x-9|-1\right)^2\left(\left(y-x+7\right)^2+|x-9|+|x-10|-1\right)^2$$
$$\cdot\left(\left(y-3\right)^2+|x-9|+|x-10|-1\right)^2(x-11)^2(y+x-15)^2(x-13)^2(x-14)^2\left(\left(y-2\right)^2+|x-14|+|x-16|-2\right)^2$$
$$\cdot\left(\left(y-3\right)^2+|x-14|+|x-15|-1\right)^2\left(\left(y-4\right)^2+|x-14|+|x-16|-2\right)^2(x-18)^2\left(\left(y-4\right)^2+|x-17|+|x-19|-2\right)^2(x-21)^2$$
$$\cdot\left(\left(y-4\right)^2+|x-20|+|x-22|-2\right)^2\left(\left(y-2\right)^2+|x-20|+|x-22|-2\right)^2\left(\left(y-\tfrac{1}{2}x+\tfrac{17}{2}\right)^2+|x-23|+|x-25|-2\right)^2$$
$$\cdot\left(\left(y+\tfrac{1}{2}x-\tfrac{29}{2}\right)^2+|x-23|+|x-25|-2\right)^2(x-28)^2\left(\left(y+\tfrac{1}{4}x-11\right)^2+|x-28|+|x-30|-2\right)^2\left(\left(y-\tfrac{1}{8}x+\tfrac{1}{4}\right)^2+|x-28|+|x-30|-2\right)^2$$
$$\cdot\left(\left(y+\tfrac{5}{8}x-\tfrac{83}{4}\right)^2+|x-28|+|x-30|-2\right)^2(x-31)^2\left(\left(y-2\right)^2+|x-31|+|x-33|-2\right)^2\left(\left(y-3\right)^2+|x-31|+|x-32|-1\right)^2$$
$$\cdot\left(\left(y-4\right)^2+|x-31|+|x-33|-2\right)^2\left(\left(y-\tfrac{1}{2}x+15\right)^2+|x-34|+|x-36|-2\right)^2\left(\left(y-3\right)^2+|x-34|+|x-36|-2\right)^2$$
$$\cdot\left(\left(y-\tfrac{1}{2}x+14\right)^2+|x-34|+|x-36|-2\right)^2\left((x-38)^2+(y-3)^2-1\right)^2(x-40)^2(y+x-44)^2(x-42)^2(y-2x+84)^2$$
$$\cdot(y+2x-92)^2\left(\left(y-3\right)^2+\left|x-\tfrac{87}{2}\right|+\left|x-\tfrac{89}{2}\right|-1\right)^2(x-46)^2(y+x-50)^2(x-48)^2\left(\left(y-\tfrac{1}{2}x+\tfrac{43}{2}\right)^2+|x-49|+|x-51|-2\right)^2$$
$$\cdot\left(\left(y+\tfrac{1}{2}x-\tfrac{55}{2}\right)^2+|x-49|+|x-51|-2\right)^2(x-52)^2\left(\left(y-2\right)^2+|x-52|+|x-54|-2\right)^2\left(\left(y-3\right)^2+|x-52|+|x-53|-1\right)^2$$
$$\cdot\left(\left(y-4\right)^2+|x-52|+|x-54|-2\right)^2+\left(y^2-6y+8+\sqrt{y^4-12y^3+52y^2-96y+64}\right)^2=0$$

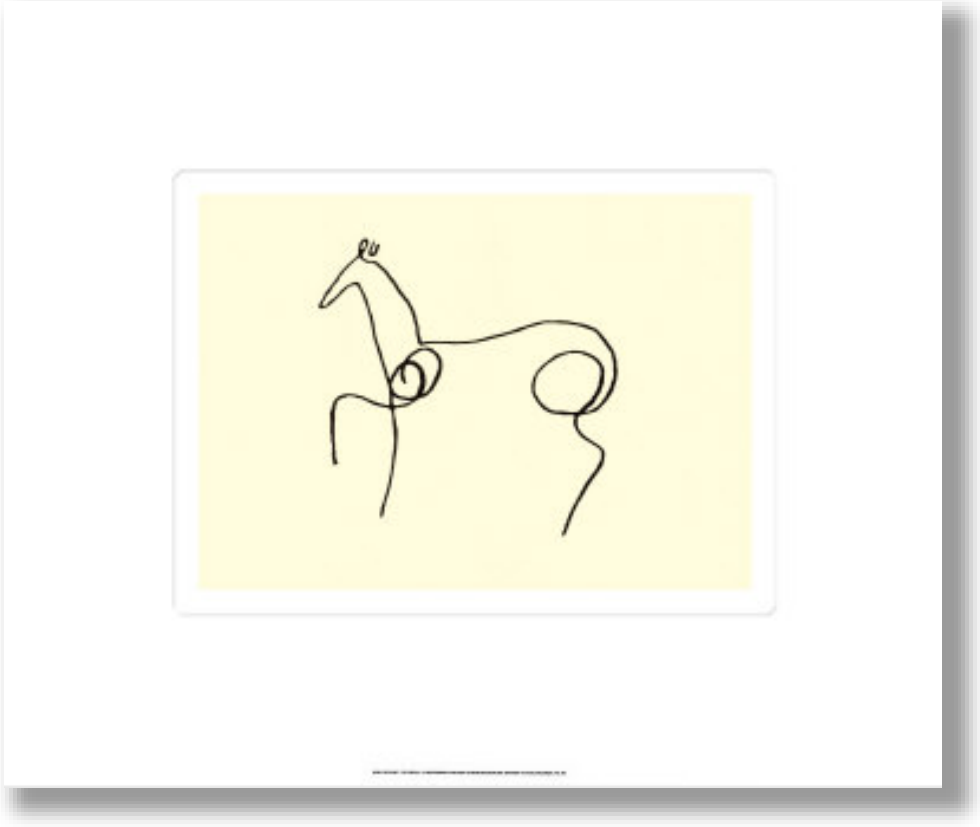
$$E = mc^2$$

Horse, by Picasso

Another Horse, by Picasso



Yet Another Horse, by Picasso



What does KISS cost?

CASE STUDY: SHELLSHOCK (BASH ATTACK)

ShellShock Attack

- Found by Stéphane Chazelas
- CVE-2014-6271 & CVE-2014-7169

```
-0-[xiayubin@MBA tmp]$ env VAR='() { :;}; echo Bash is vulnerable!' bash -c "echo Bash Test"
Bash is vulnerable!
Bash Test
```

The Vulnerability

- A function is defined in a var, which is exported
- A child bash copies all of its parent's env vars
- The child will execute some code in a var during the copy!

```
$ export X='() { echo "inside X"; }; echo "outside X";'
$ bash
outside X
$ env
…
X=() {  echo “inside X” }
…
```

The Vulnerability

- env VAR='() { :;}; echo Bash is vulnerable!' bash -c "echo Bash Test”
  - ‘:’ means ‘/bin/true’, which returns true and return

- Now please explain this test by yourselves
  - What would be executed during bash forking?
  - What can an attacker do if she can define such VAR?

Why Big Deal?

- On apache server, it is typical to fork a child process for each HTTP client
  - Use bash *env* variables a lot

- A DHCP client may get malicious value from a malicious DHCP server

Why ShellShock Ever Exist?

- Three principles of UNIX programming [1]:
  - Write programs that do one thing and do it well
  - Write programs to work together
  - Write programs to handle *text streams*, because that is a universal interface

[1] http://en.wikipedia.org/wiki/Unix_philosophy

Interesting Shell

$ ls -l *.c

“ls”, “-l”, "*.c” # *.c =>“foo.c” “bar.c”

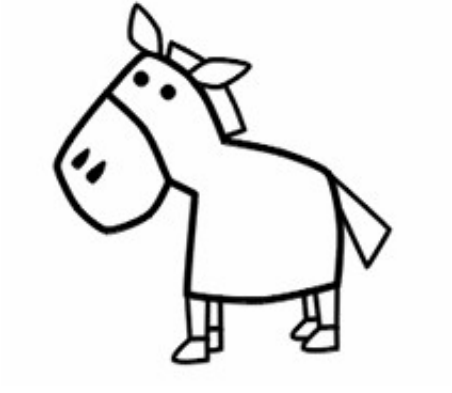
“ls”, “-l”, “foo.c” “bar.c”

What if a file with name “-l”?

>-1

The Simpler, The Better?

- No…
- The point is: how to find a right way to be simple?



How to make the kernel simpler?

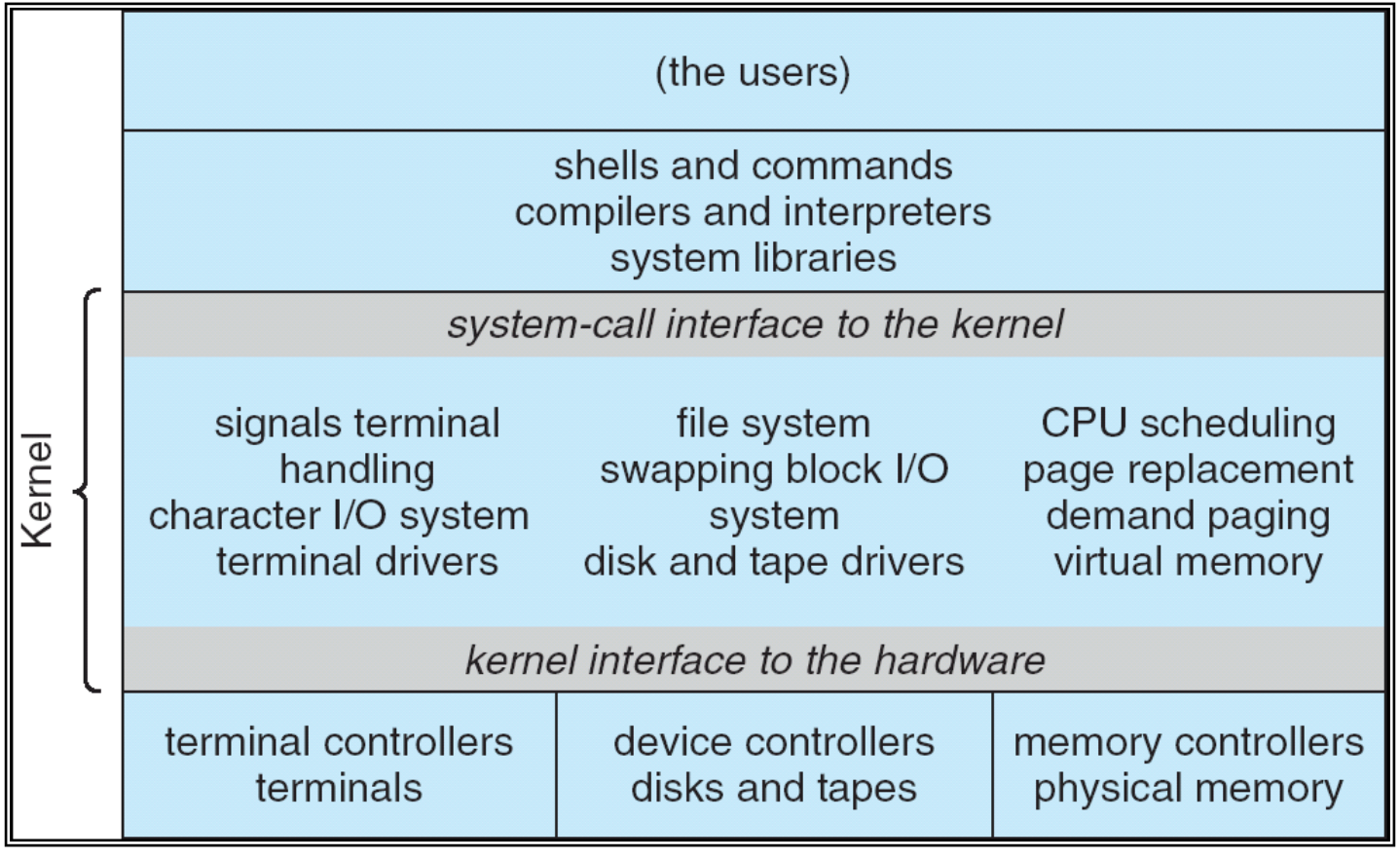
ADOPT KISS TO KERNEL

Monolithic Kernel

- A monolithic kernel is an operating system architecture where the entire operating system is working in kernel space and is alone in supervisor mode

- E.g., Linux, BSD…

The Problem: Monolithic Kernel is Too Large



Micro-Kernel Came to the Rescue

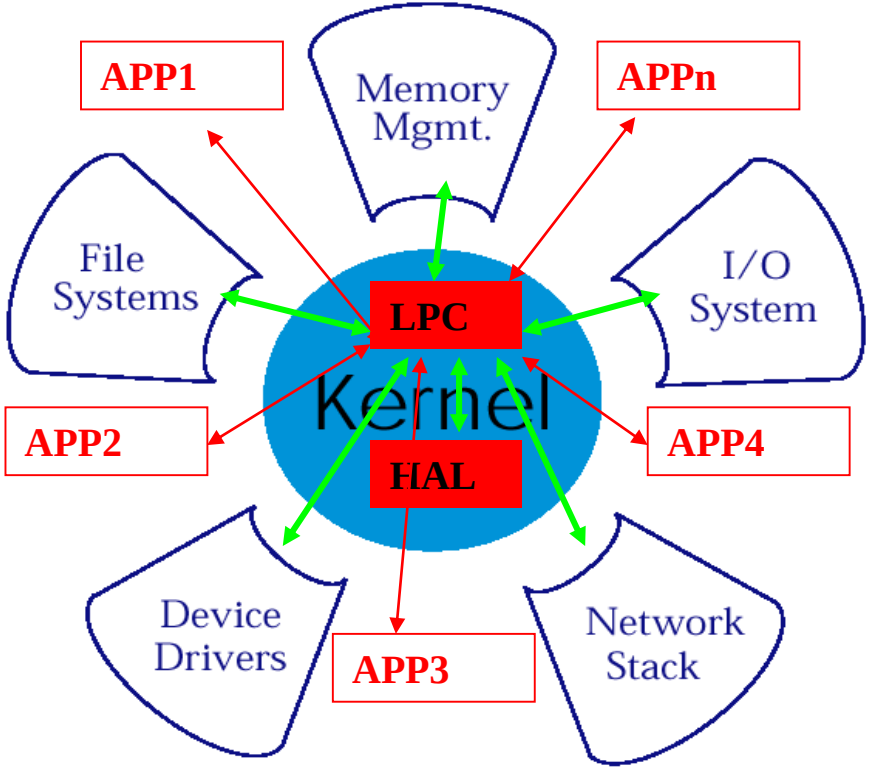
- Near-minimum amount of software that can provide the mechanisms needed to implement an operating system (OS)
- Low-level address space management, thread management, and inter-process communication (IPC) and so on …

- Mach, L4 kernel

Micro-Kernel

- Moves as much from the kernel into “user” space
  - Communication between user modules using message passing
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Detriments:
  - Performance overhead of user to kernel communication

Micro-Kernel Structure & Instances

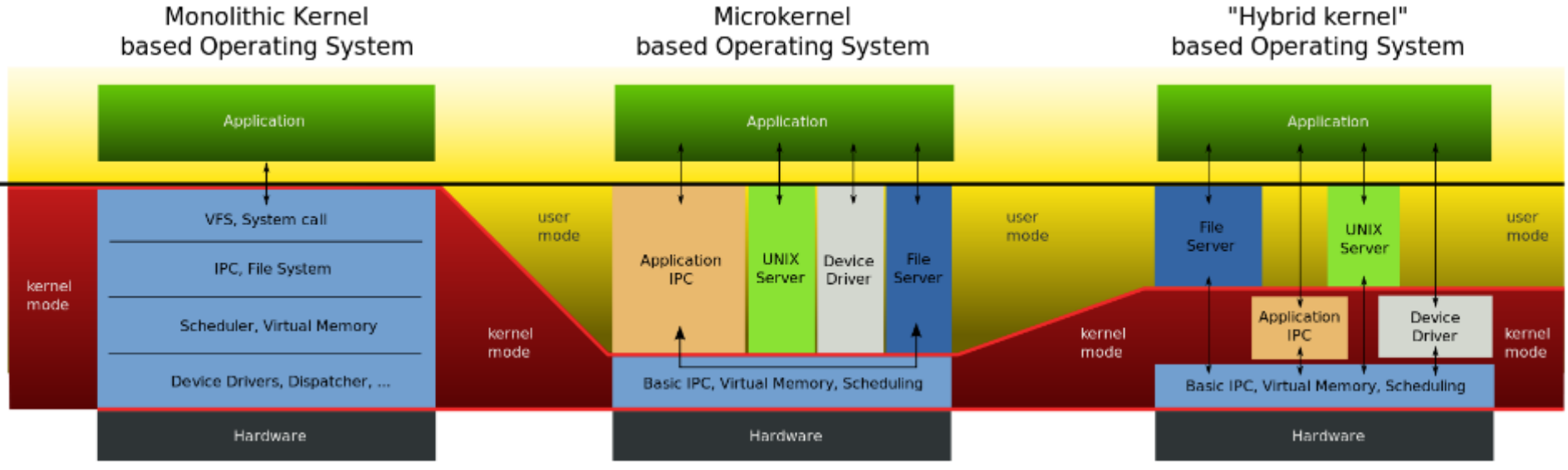


Hybrid-Kernel

- Kernel architecture based on combining aspects of microkernel and monolithic kernel architectures used in computer operating systems

- Windows NT kernel

Comparison


Monolithic Kernel
based Operating System
Application
VFS, System call
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
Hardware
kernel mode
user mode
Microkernel
based Operating System
Application
Application IPC
UNIX Server
Device Driver
File Server
Basic IPC, Virtual Memory, Scheduling
Hardware
"Hybrid kernel"
based Operating System
Application
File Server
UNIX Server
Application IPC
Device Driver
Basic IPC, Virtual Memory, Scheduling
Hardware

Kernels

Kernel classification (wikipedia)

Kernel (architecture)	Monolithic	Microkernel	**Exokernel**	hybrid

https://en.wikipedia.org/wiki/Kernel_(operating_system)

Exokernel: Motivation

- In traditional OS, only *privileged servers and the kernel* can manage system resources
- Un-trusted applications are required to interact with the hardware via some abstraction model
  - File systems for disk storage, virtual address spaces for memory, etc.

- Application demands vary widely!!
  - An interface designed to accommodate every application must anticipate all possible needs

Exokernel's Idea

Give un-trusted **applications** as much control over physical resources as possible

To force as few abstraction as possible on developers, enabling them to make as many decisions as possible about hardware abstractions.

- Let *the kernel* allocate the basic physical resources of the machine
- Let *each program* decide what to do with these resources

Exokernel separate *protection* from *management*

- They **protect** resources but delegate **management** to application

Exokernel: Design

- Kernel’s new role
  - Tracking ownership of resources
  - Ensuring resource protection
  - Revoking resource access
- Three techniques
  - Secure binding
  - Visible revocation
  - Abort protocol

Library Operating System

Most programs to be linked with *libraries* instead of communicating with the exokernel directly

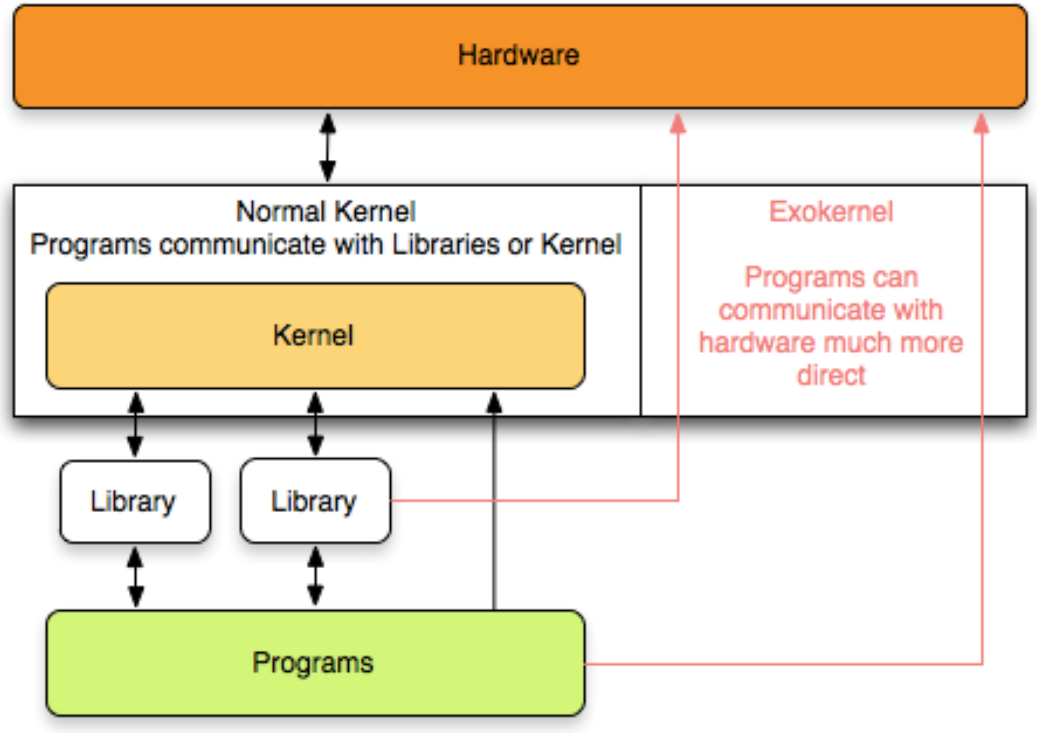
- The libraries hide low-level resources

An applications can choose the library which best suits its needs

The kernel only ensures that the requested resource is free, and the application is allowed to access it

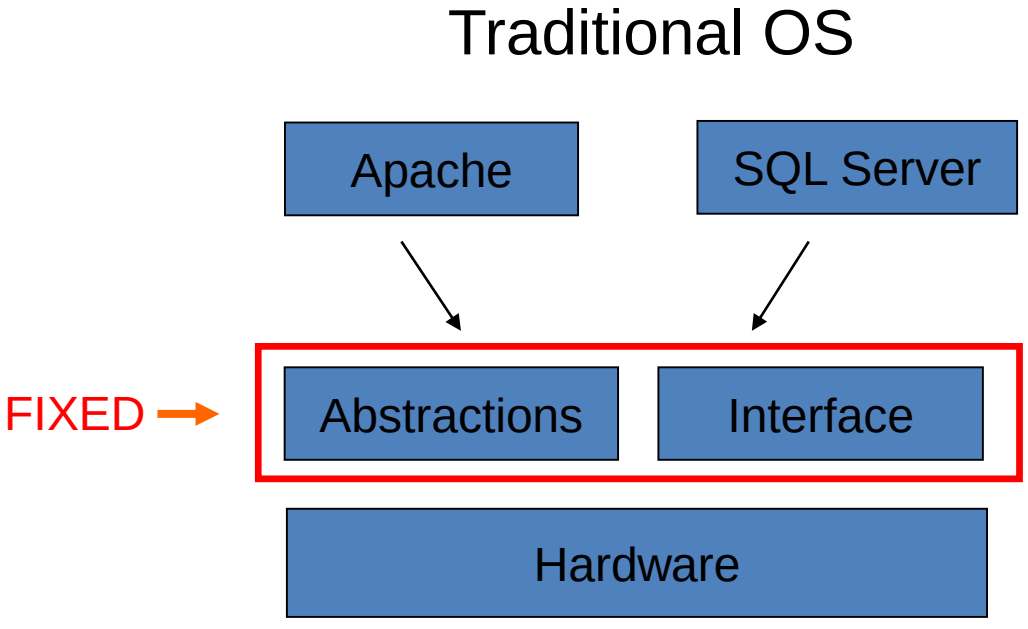
- Allow programmer to implement custom abstractions, omit unnecessary ones, most commonly to improve performance

Exokernel Example 1

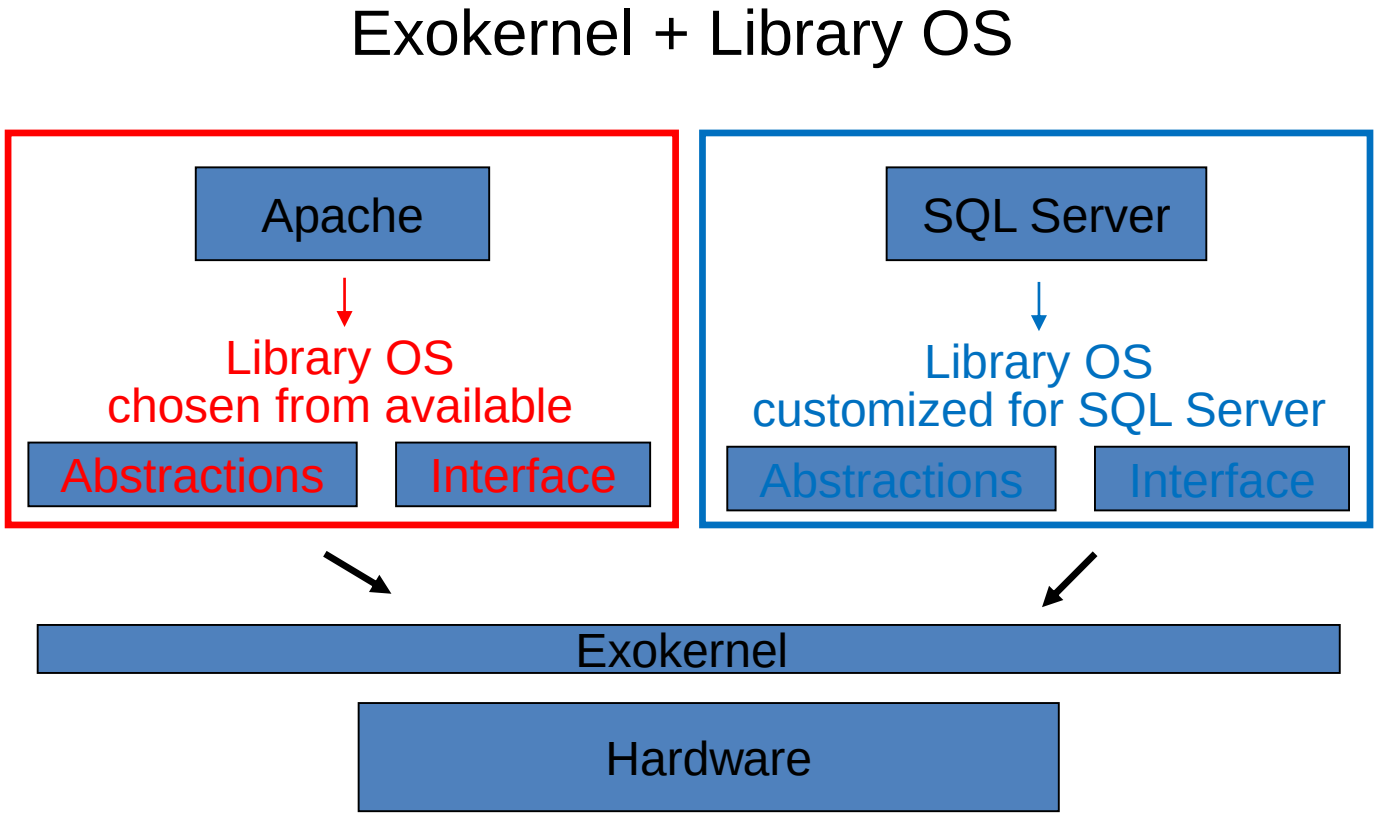


Exokernel give more direct access to the hardware, thus removing most abstractions

Exokernel Example 2



Exokernel Example 2



The way to read a paper and take note

P.R.O.S.E.C.

P.R.O.S.E.C.

- Problem
- Related work
- Observation
- Solution
- Evaluation
- Comments

P.R.O.S.E.C.

- **Problem**: What is the motivation? Why important?
- **Related work**: Why other work didn’t solve it?
- **Observation**: What is the key insight? Make it simple
- **Solution**: How does this work solve the problem?
- **Evaluation**: Does this work really solve the problem?
- **Comments**: What could be better? Your own opinion

Other Info

- Web site
  - http://ipads.se.sjtu.edu.cn/courses/csp
- TA
  - Bicheng Yang