Mutltiprogramming, Multithreading and Event-driven programming

Azza Abouzied

What is a process

A bundle of things:

1. A thread of execution
2. Program counter, registers, stack (execution history)
3. Address space
4. Resources managed by the kernel
    - Pointers to system resources: e.g. file descriptors
    - Privileges & User
    - Current working directory
    - Bookkeeping stuff: resource tracking; address-space management (e.g. process id, current state, ...)

What happens on a context switch?

Context switching, the switching from one runnable task to another.

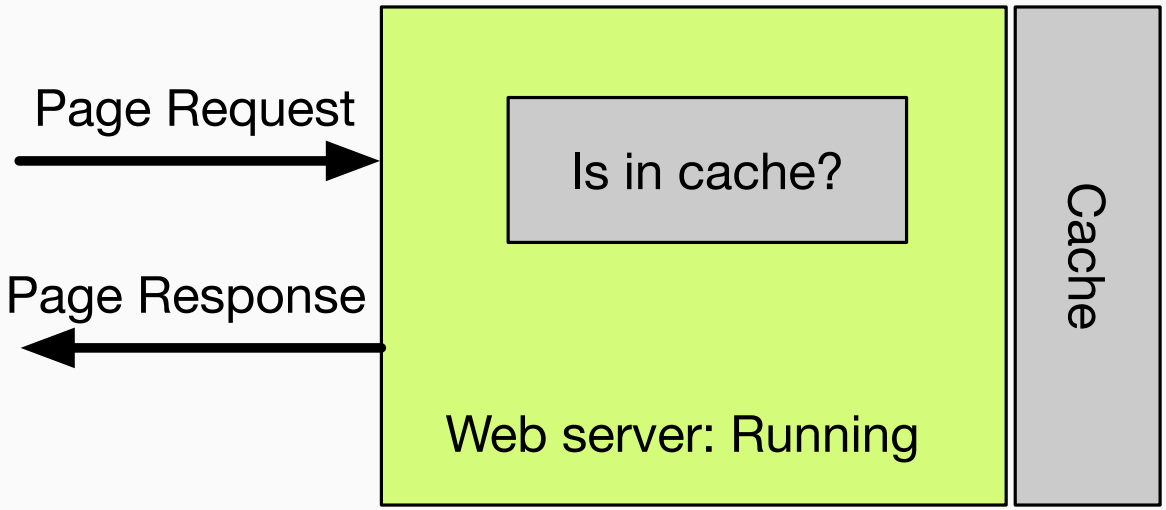
Occurs at interrupts, system calls, any preemption

Kernel sets up the environment for the process to continue working.

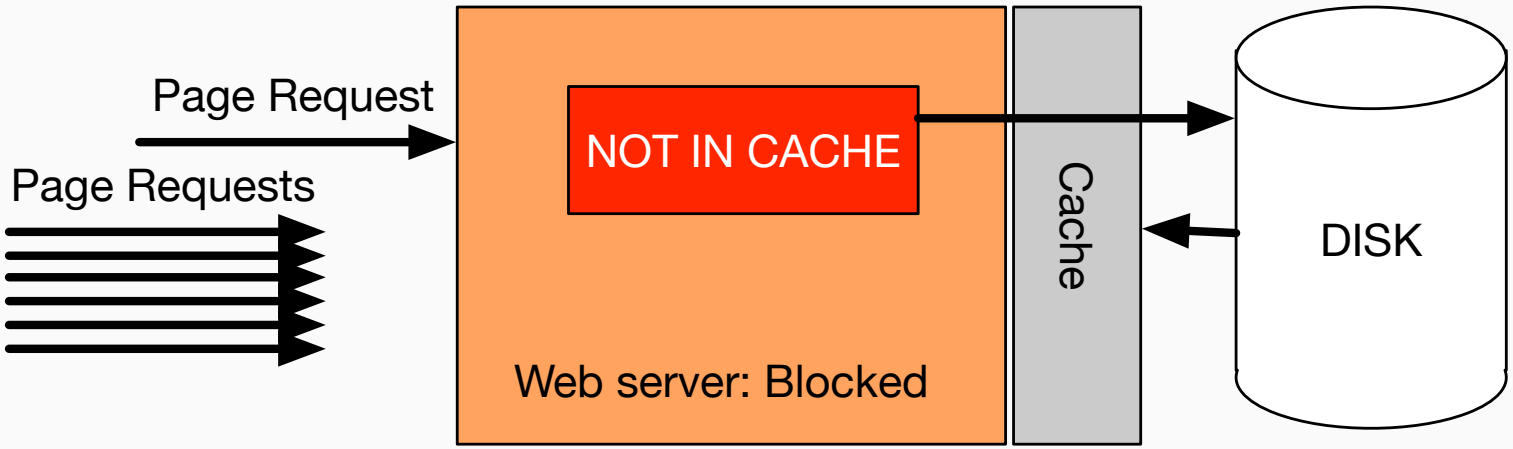
Is the process the right level of abstraction?

Case Study: A Web Server

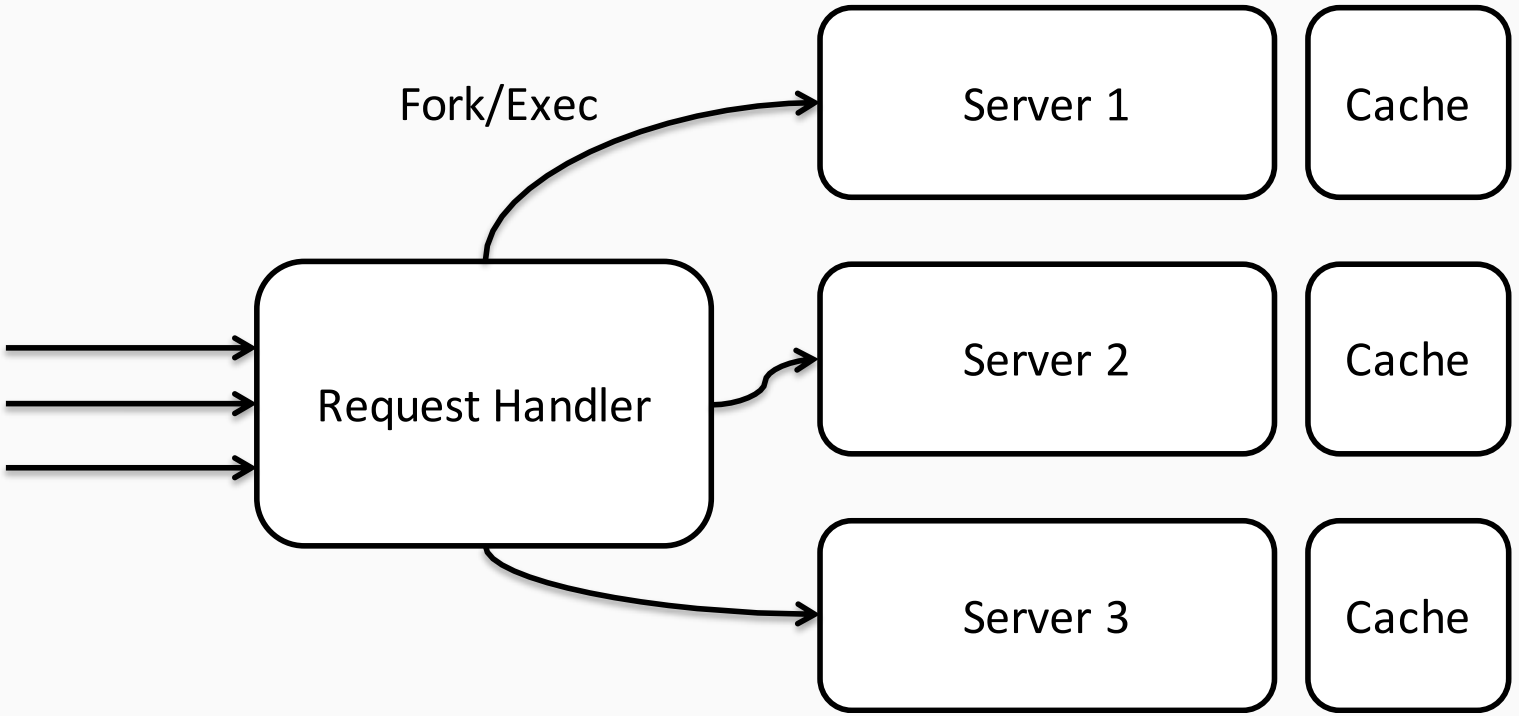
The Single Process Web Server



The Single Process Web Server



The server with N processes



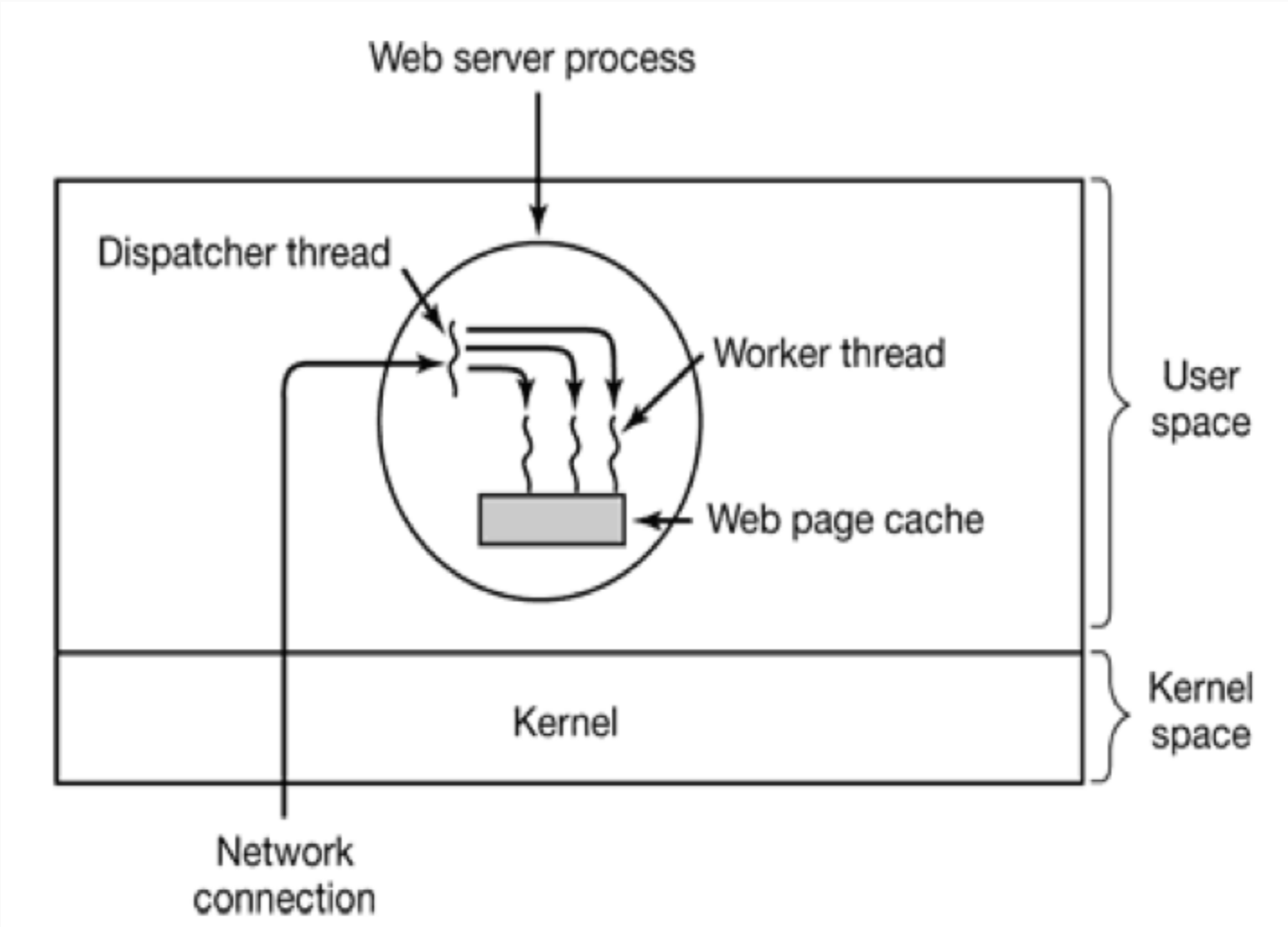
Why is this bad?

A server with several processes is generally a bad idea:

1. Expensive Context Switching between processes
2. Independent Address spaces when caches need to be shared![1]
3. Limited # of processes that can be issued (Kernel sets these limits)

[1] Okay, okay ... you can declare a shared memory space between processes but we haven't gotten there yet ;)

The Multithreaded Server



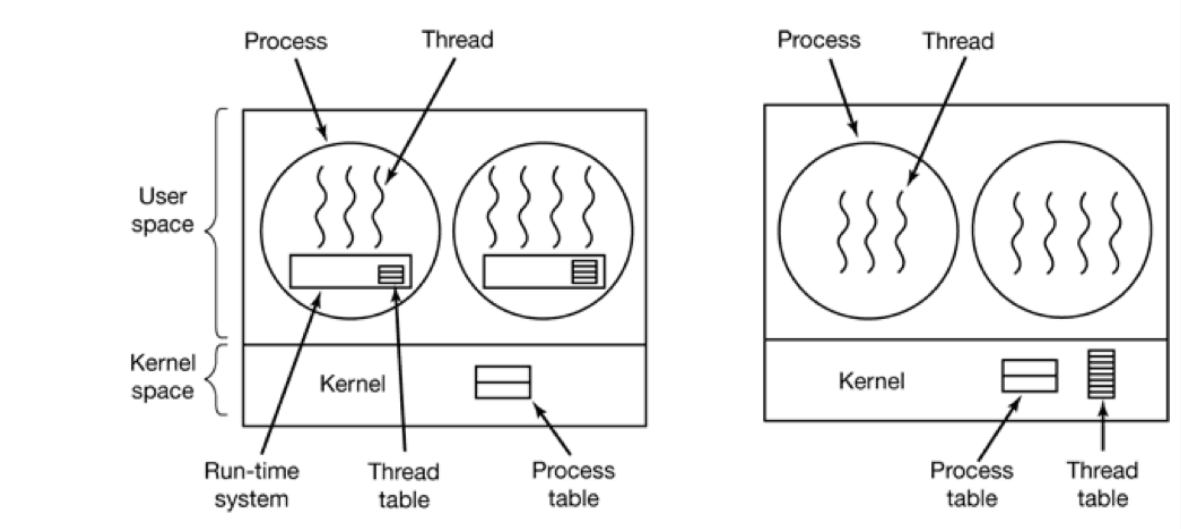
A Thread vs. a process

What do you need to take care of during a context switch between one user process and another and between two threads in a user process?

Process	Thread
Everything in the thread	Program Counter
Address space	Stack
Global variables	Registers
Open files	State
Child processes	
Pending alarms	
Registered Interrupts/Signals & handlers	
Bookkeeping stuff (pids, priveleges, ...)	

The devil in the details

How are threads implemented actually matters?



1. User-level threads (Many to one) vs. Kernel-level threads (One to one).

2. Linux, Mac OS, WindowsXP, Solaris provide 1-1 mappings.

3. There are many Many to few implementations as well.

What are the advantages and disadvantages of the different models?

Wait! There is more!

The event-driven server

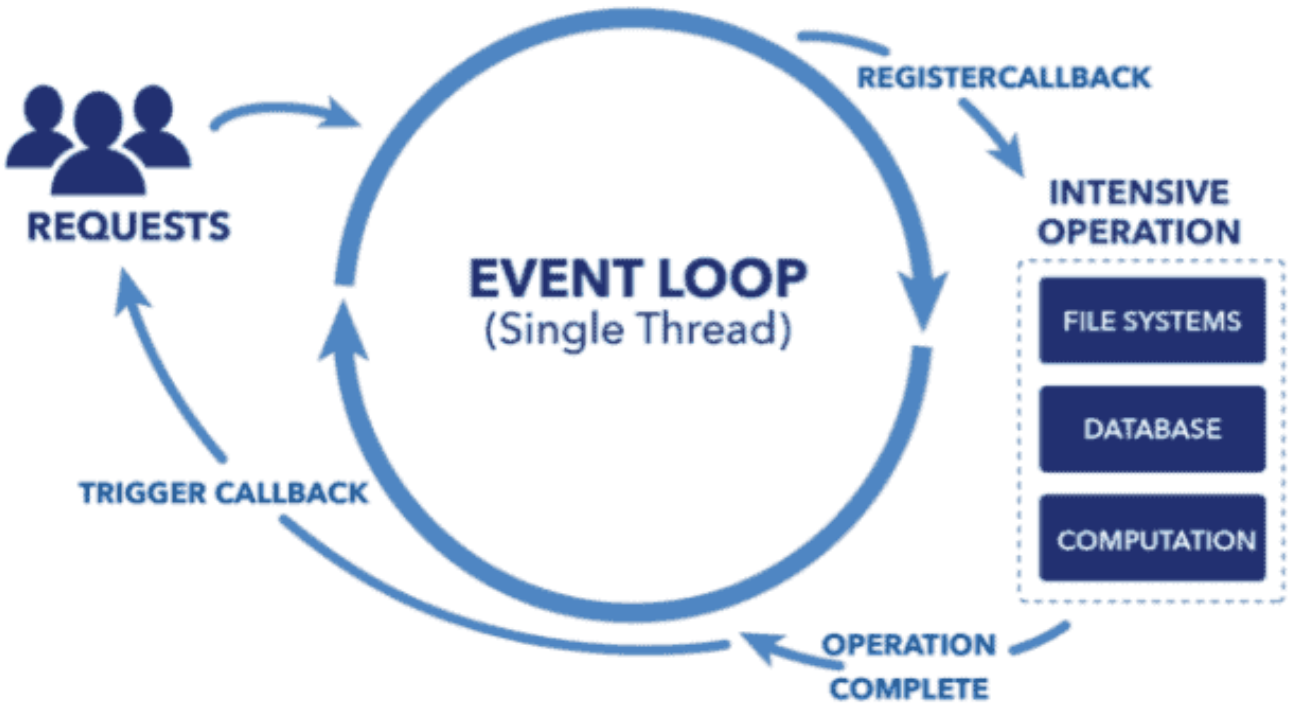
The problem with our first server was that IO resulted in blocking calls.

1. The OS can provide non-blocking/asynchronous IO calls

2. We can build the process in a way that behaves like interrupt handling except we call it event handling.

There are event-driven servers in the wild: Node.js, memcached, etc.

The Node.js Server



Event-driven or Multithreaded?

Not an easy debate to settle:

- 1995, **Why threads are a bad idea?** John Ousterhout (UC Berkeley, Sun Labs)
- 2003, **Why events are a bad idea?** Van Behren, Condit, Brewer (UC Berkeley)

Why did we favor events in 1995?

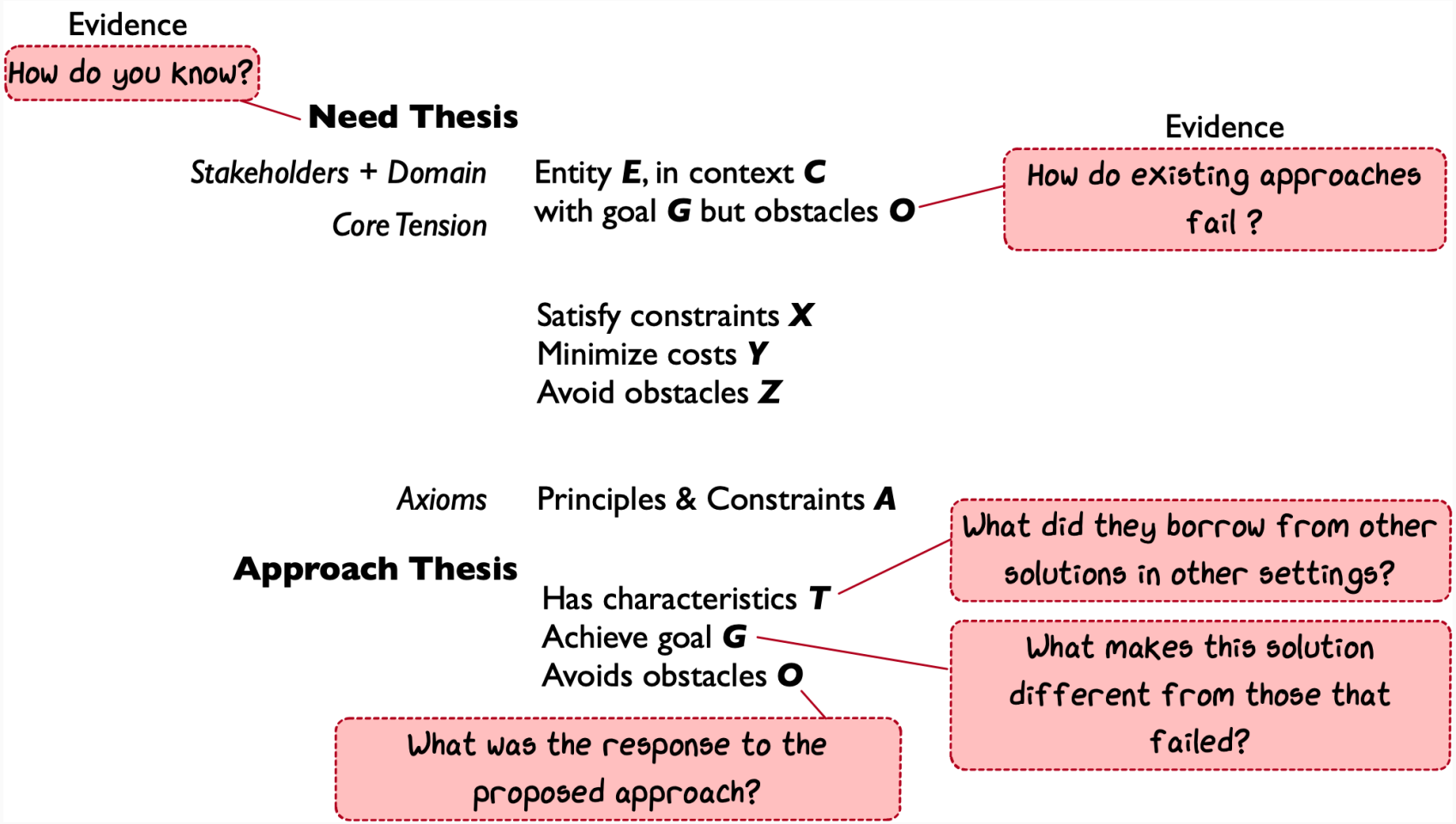
Which ones are easier to program?

Which ones require less resources?

Reading & Critiquing a Paper

How to read a paper in general?

The Design Arguments



FFS Annotated

A Fast File System for UNIX*

Marshall Kirk McKusick, William N. Joy†,
Samuel J. Leffler‡, Robert S. Fabry

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720

ABSTRACT

A reimplementation of the UNIX file system is described. The reimplementation provides substantially higher throughput rates by using more flexible allocation policies that allow better locality of reference and can be adapted to a wide range of peripheral and processor characteristics. The new file system clusters data that is sequentially accessed and provides two block sizes to allow fast access to large files while not wasting large amounts of space for small files. File access rates of up to ten times faster than the traditional UNIX file system are experienced. Long needed enhancements to the programmers' interface are discussed. These include a mechanism to place advisory locks on files, extensions of the name space across file systems, the ability to use long file names, and provisions for administrative control of resource usage.

Annotate the paper with the design arguments: Goals, Characteristics, Approach, Context, Needs, ... etc.

Critiquing

Read the blog post https://azzablogs.com/2019/01/23/how-to-write-a-critique-for-a-research-paper/

Use the design arguments to help you describe the work.

Great critiques anticipate future contexts and needs and re-examine the work in light of those.

1. Discuss Pros and Cons with the future in mind (or the present if the paper is from the past).

2. Does the work handle new technology (e.g. hardware, applications) in the horizon?

3. Does it scale to current workloads (consumer or enterprise)?

4. If not, why and how would you modify it?

Questions?