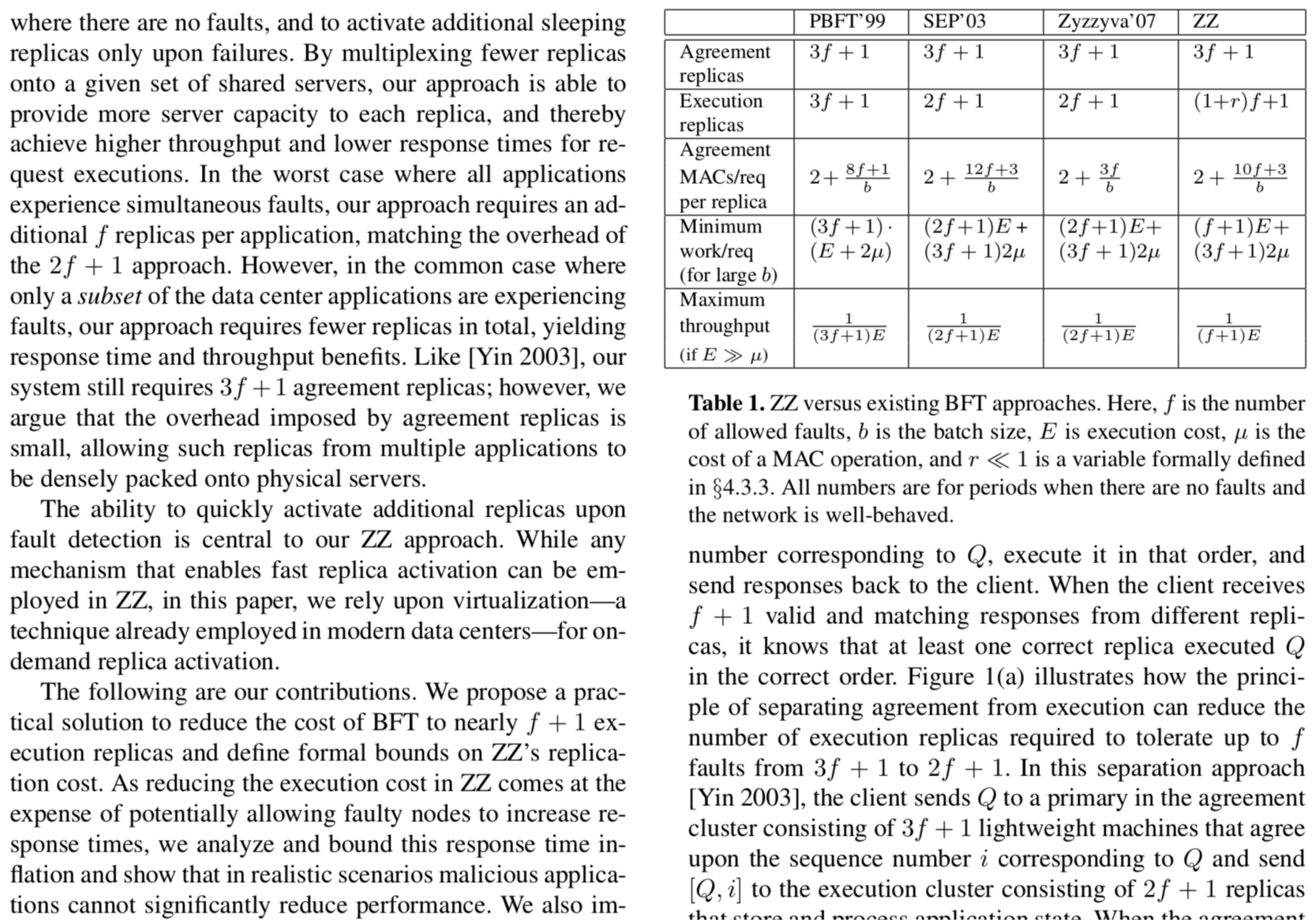
Writing Well

CSCI 8901:
Research & Evaluation Methods

Prof. Tim Wood
GWU

Scientific Papers

So boring…



where there are no faults, and to activate additional sleeping replicas only upon failures. By multiplexing fewer replicas onto a given set of shared servers, our approach is able to provide more server capacity to each replica, and thereby achieve higher throughput and lower response times for request executions. In the worst case where all applications experience simultaneous faults, our approach requires an additional f replicas per application, matching the overhead of the $2f+1$ approach. However, in the common case where only a *subset* of the data center applications are experiencing faults, our approach requires fewer replicas in total, yielding response time and throughput benefits. Like [Yin 2003], our system still requires $3f+1$ agreement replicas; however, we argue that the overhead imposed by agreement replicas is small, allowing such replicas from multiple applications to be densely packed onto physical servers.

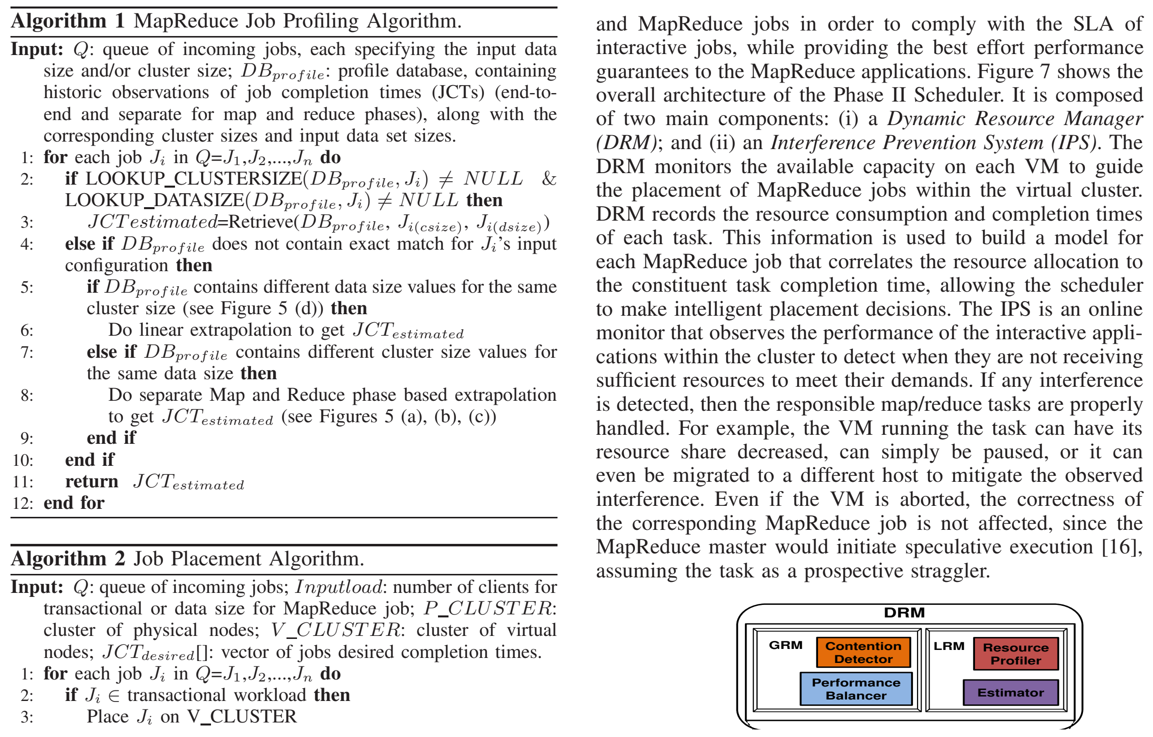
The ability to quickly activate additional replicas upon fault detection is central to our ZZ approach. While any mechanism that enables fast replica activation can be employed in ZZ, in this paper, we rely upon virtualization—a technique already employed in modern data centers—for on-demand replica activation.

The following are our contributions. We propose a practical solution to reduce the cost of BFT to nearly $f+1$ execution replicas and define formal bounds on ZZ's replication cost. As reducing the execution cost in ZZ comes at the expense of potentially allowing faulty nodes to increase response times, we analyze and bound this response time inflation and show that in realistic scenarios malicious applications cannot significantly reduce performance. We also im-

	PBFT'99	SEP'03	Zyzzyva'07	ZZ
Agreement replicas	$3f+1$	$3f+1$	$3f+1$	$3f+1$
Execution replicas	$3f+1$	$2f+1$	$2f+1$	$(1+r)f+1$
Agreement MACs/req per replica	$2+\frac{8f+1}{b}$	$2+\frac{12f+3}{b}$	$2+\frac{3f}{b}$	$2+\frac{10f+3}{b}$
Minimum work/req (for large b)	$(3f+1)\cdot(E+2\mu)$	$(2f+1)E+(3f+1)2\mu$	$(2f+1)E+(3f+1)2\mu$	$(f+1)E+(3f+1)2\mu$
Maximum throughput (if $E \gg \mu$)	$\frac{1}{(3f+1)E}$	$\frac{1}{(2f+1)E}$	$\frac{1}{(2f+1)E}$	$\frac{1}{(f+1)E}$

Table 1. ZZ versus existing BFT approaches. Here, f is the number of allowed faults, b is the batch size, E is execution cost, μ is the cost of a MAC operation, and $r \ll 1$ is a variable formally defined in §4.3.3. All numbers are for periods when there are no faults and the network is well-behaved.

number corresponding to Q, execute it in that order, and send responses back to the client. When the client receives $f+1$ valid and matching responses from different replicas, it knows that at least one correct replica executed Q in the correct order. Figure 1(a) illustrates how the principle of separating agreement from execution can reduce the number of execution replicas required to tolerate up to f faults from $3f+1$ to $2f+1$. In this separation approach [Yin 2003], the client sends Q to a primary in the agreement cluster consisting of $3f+1$ lightweight machines that agree upon the sequence number i corresponding to Q and send $[Q, i]$ to the execution cluster consisting of $2f+1$ replicas that store and process application state. When the agreement



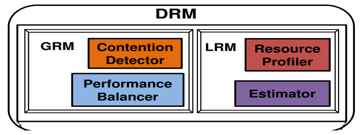
Algorithm 1 MapReduce Job Profiling Algorithm.

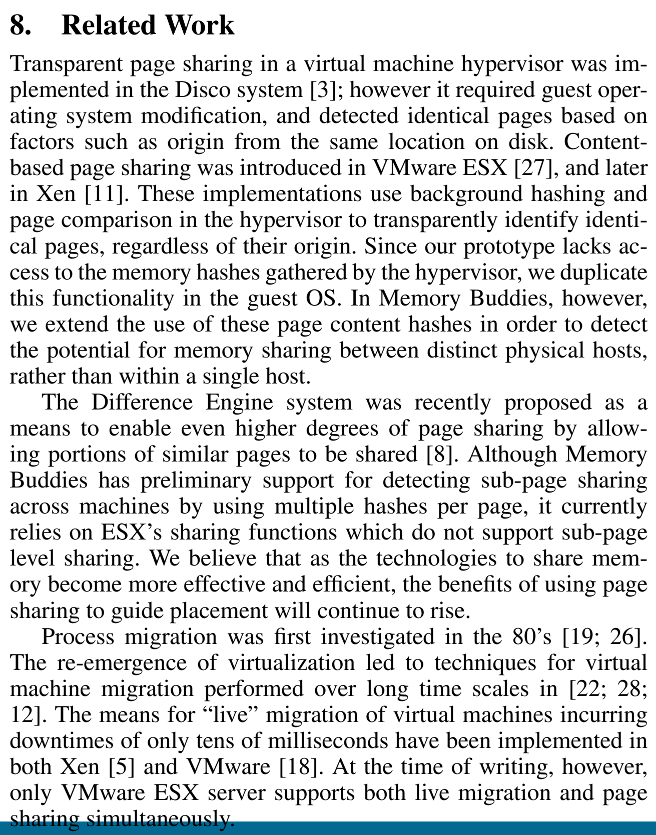
```
Input: $Q$: queue of incoming jobs, each specifying the input data
    size and/or cluster size; $DB_{profile}$: profile database, containing
    historic observations of job completion times (JCTs) (end-to-
    end and separate for map and reduce phases), along with the
    corresponding cluster sizes and input data set sizes.
 1: for each job $J_i$ in $Q$=$J_1$,$J_2$,...,$J_n$ do
 2:   if LOOKUP_CLUSTERSIZE($DB_{profile}, J_i$) $\neq NULL$ &
      LOOKUP_DATASIZE($DB_{profile}, J_i$) $\neq NULL$ then
 3:     $JCTestimated$=Retrieve($DB_{profile}$, $J_{i(csize)}$, $J_{i(dsize)}$)
 4:   else if $DB_{profile}$ does not contain exact match for $J_i$'s input
      configuration then
 5:     if $DB_{profile}$ contains different data size values for the same
        cluster size (see Figure 5 (d)) then
 6:       Do linear extrapolation to get $JCT_{estimated}$
 7:     else if $DB_{profile}$ contains different cluster size values for
        the same data size then
 8:       Do separate Map and Reduce phase based extrapolation
          to get $JCT_{estimated}$ (see Figures 5 (a), (b), (c))
 9:     end if
10:   end if
11:   return $JCT_{estimated}$
12: end for
```

Algorithm 2 Job Placement Algorithm.

```
Input: $Q$: queue of incoming jobs; $Inputload$: number of clients for
    transactional or data size for MapReduce job; $P\_CLUSTER$:
    cluster of physical nodes; $V\_CLUSTER$: cluster of virtual
    nodes; $JCT_{desired}$[]: vector of jobs desired completion times.
 1: for each job $J_i$ in $Q$=$J_1$,$J_2$,...,$J_n$ do
 2:   if $J_i \in$ transactional workload then
 3:     Place $J_i$ on V_CLUSTER
```

and MapReduce jobs in order to comply with the SLA of interactive jobs, while providing the best effort performance guarantees to the MapReduce applications. Figure 7 shows the overall architecture of the Phase II Scheduler. It is composed of two main components: (i) a *Dynamic Resource Manager (DRM)*; and (ii) an *Interference Prevention System (IPS)*. The DRM monitors the available capacity on each VM to guide the placement of MapReduce jobs within the virtual cluster. DRM records the resource consumption and completion times of each task. This information is used to build a model for each MapReduce job that correlates the resource allocation to the constituent task completion time, allowing the scheduler to make intelligent placement decisions. The IPS is an online monitor that observes the performance of the interactive applications within the cluster to detect when they are not receiving sufficient resources to meet their demands. If any interference is detected, then the responsible map/reduce tasks are properly handled. For example, the VM running the task can have its resource share decreased, can simply be paused, or it can even be migrated to a different host to mitigate the observed interference. Even if the VM is aborted, the correctness of the corresponding MapReduce job is not affected, since the MapReduce master would initiate speculative execution [16], assuming the task as a prospective straggler.





8. Related Work

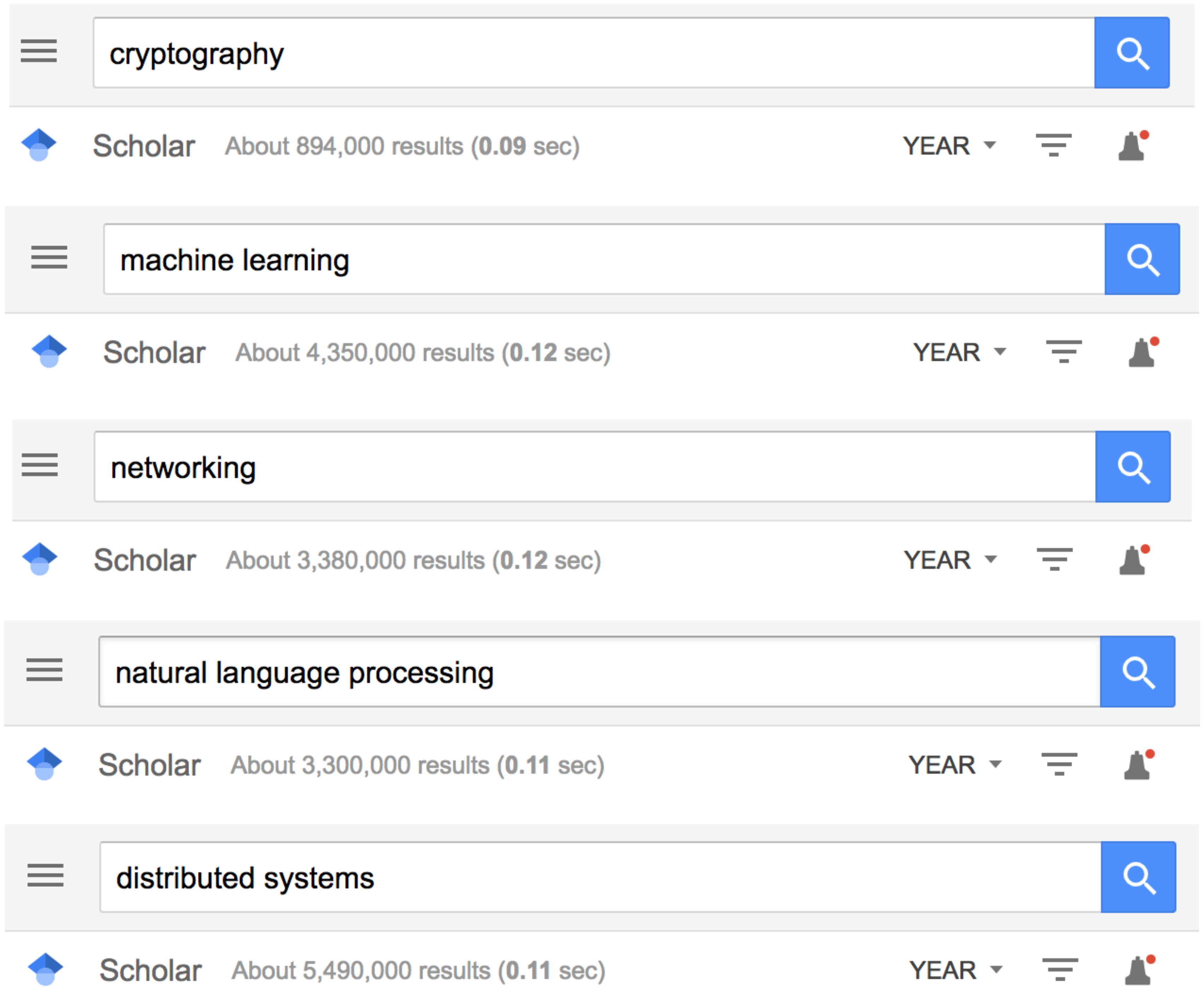
Transparent page sharing in a virtual machine hypervisor was implemented in the Disco system [3]; however it required guest operating system modification, and detected identical pages based on factors such as origin from the same location on disk. Content-based page sharing was introduced in VMware ESX [27], and later in Xen [11]. These implementations use background hashing and page comparison in the hypervisor to transparently identify identical pages, regardless of their origin. Since our prototype lacks access to the memory hashes gathered by the hypervisor, we duplicate this functionality in the guest OS. In Memory Buddies, however, we extend the use of these page content hashes in order to detect the potential for memory sharing between distinct physical hosts, rather than within a single host.

The Difference Engine system was recently proposed as a means to enable even higher degrees of page sharing by allowing portions of similar pages to be shared [8]. Although Memory Buddies has preliminary support for detecting sub-page sharing across machines by using multiple hashes per page, it currently relies on ESX's sharing functions which do not support sub-page level sharing. We believe that as the technologies to share memory become more effective and efficient, the benefits of using page sharing to guide placement will continue to rise.

Process migration was first investigated in the 80's [19; 26]. The re-emergence of virtualization led to techniques for virtual machine migration performed over long time scales in [22; 28; 12]. The means for "live" migration of virtual machines incurring downtimes of only tens of milliseconds have been implemented in both Xen [5] and VMware [18]. At the time of writing, however, only VMware ESX server supports both live migration and page sharing simultaneously.

Scientific Papers

So many…



cryptography

Scholar About 894,000 results (**0.09** sec)

machine learning

Scholar About 4,350,000 results (**0.12** sec)

networking

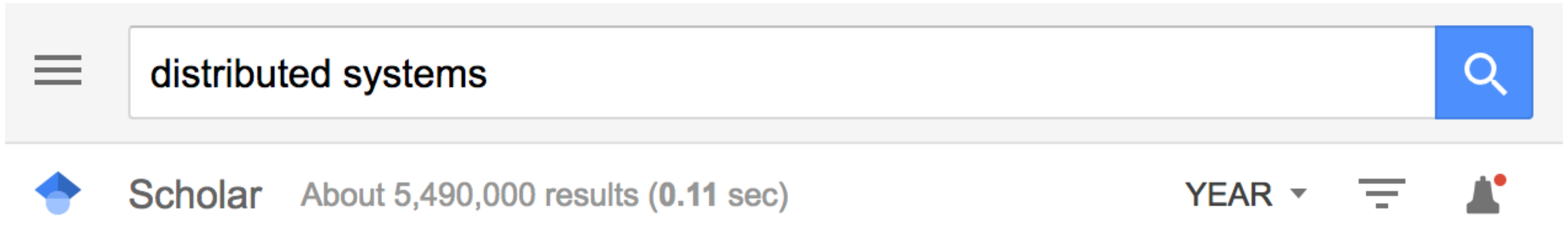
Scholar About 3,380,000 results (**0.12** sec)

natural language processing

Scholar About 3,300,000 results (**0.11** sec)

distributed systems

Scholar About 5,490,000 results (**0.11** sec)

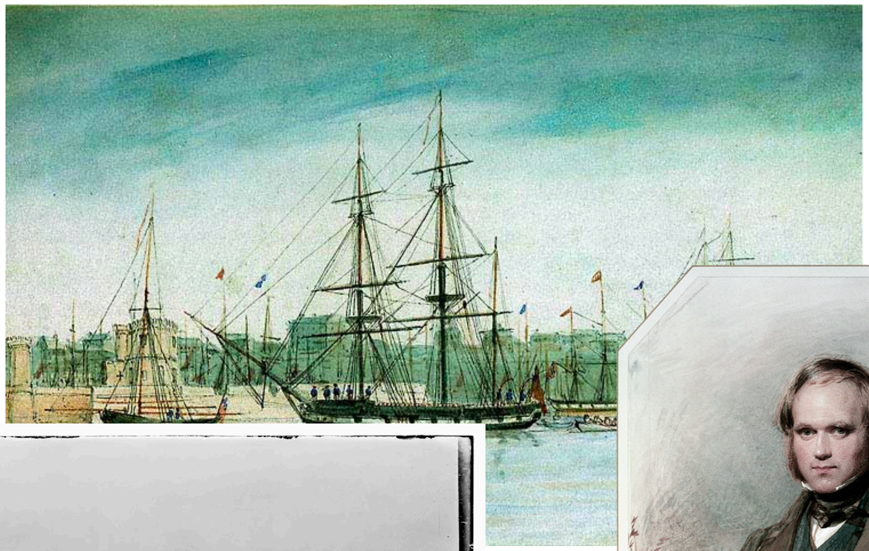
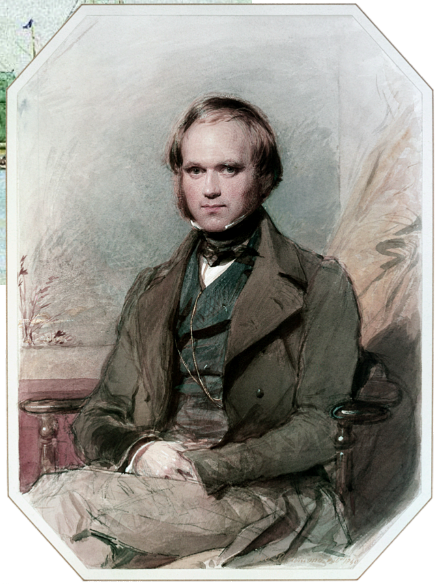

distributed systems

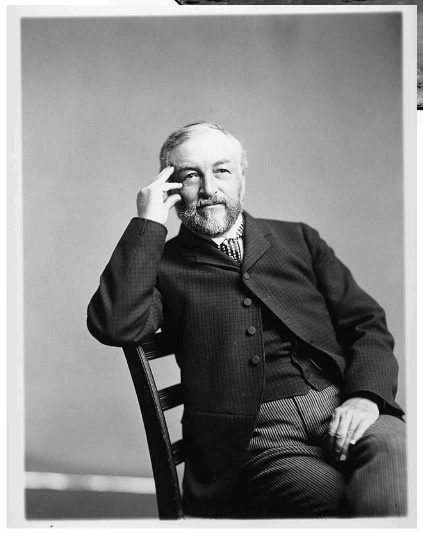
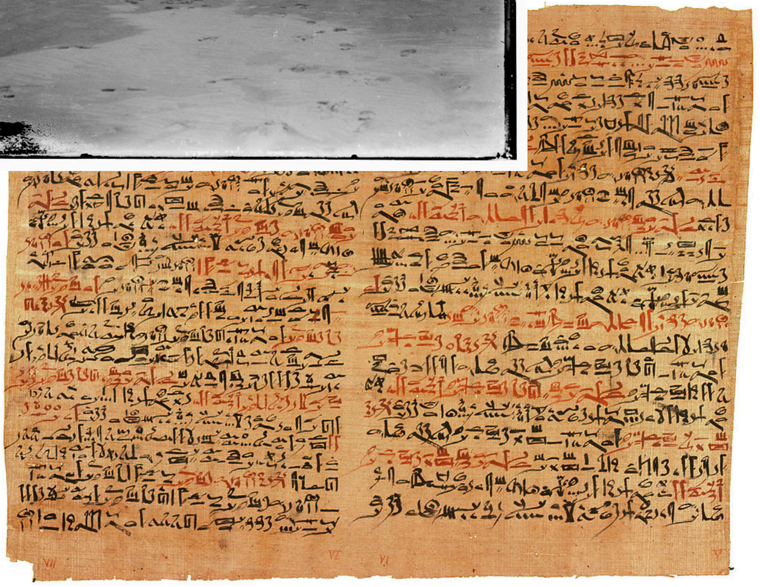
Scholar About 5,490,000 results (**0.11** sec)

Why would anyone even bother to read my paper?

Stories

So much better!

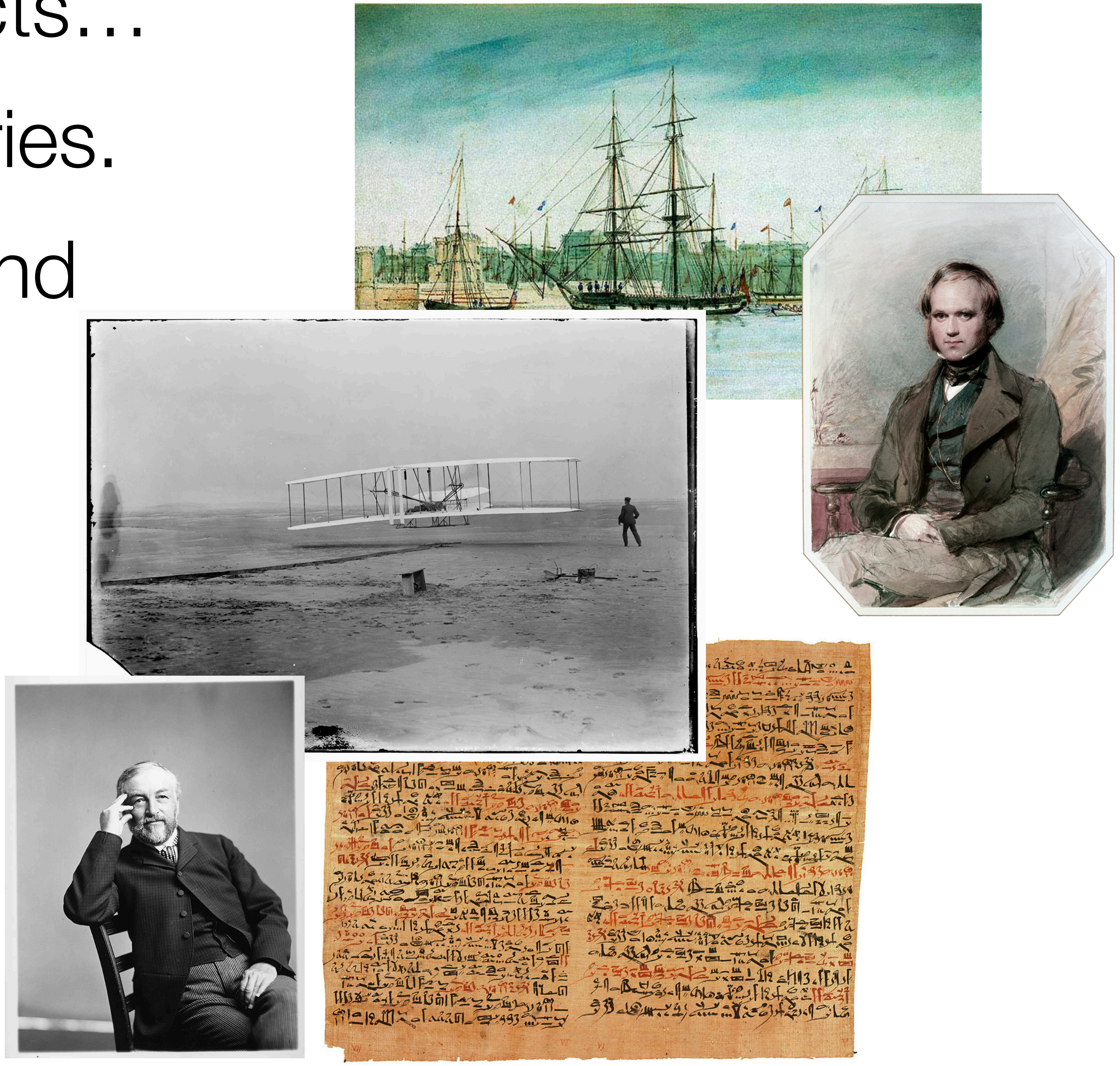
Stories

Science is about facts…

But people love stories.

People remember and pay attention to stories

People are inspired by stories



Writing Science Stories

Your papers must be:

- Factual, accurate, unbiased, detailed, comprehensive

But they should also be

- Interesting
- Memorable
- Engaging

Introductions

Starting a Story

Introductions are by far the most important part of your papers*

How to make a good introduction?

- What do you put in yours?
- What do you remember reading from papers?

*in my opinion

Recipe: Introduction

The world is a terrible, terrible place.

Recipe: Introduction

The world is a terrible, terrible place.

But imagine how wonderful it could be if we could figure out how to do X!

Recipe: Introduction

The world is a terrible, terrible place.

But imagine how wonderful it could be if we could figure out how to do X!

My work helps us get one step closer to the magical dream world!

Recipe: Introduction

The world is a terrible, terrible place.

But imagine how wonderful it could be if we could figure out how to do X!

My work helps us get one step closer to the magical dream world!

Examples…

This presentation…

Papers are boring, unpleasant to read, and hard to write!

People love stories, if our paper can be a fun story, everyone will want to read it!

With the right recipe, we can write a good introduction to hook the reader and keep their attention!

VM Security

Virtual machines in the cloud are an easy target for attackers!

If we could scan for attacks beneath the OS layer we could detect more attacks and provide security as a service!

Our system, CRIMES, demonstrates the feasibility of efficiently scanning at this layer

Abstract

What is in an abstract?

Recipe: Abstract

This is a mini version of an intro

1. Set context
2. Show a problem
3. Review existing approaches
4. Name a solution
5. Hint at an evaluation

Let's look at some

Best paper award abstracts from:

- NIPS 2018
- EMNLP 2018
- OSDI 2018
- INFOCOM 2018
- Crypto 2018

Does it follow our pattern?

- 1. Set context
- 2. Show a problem
- 3. Review existing approaches
- 4. Name a solution
- 5. Hint at an evaluation

How much space for each section?

Abstracts

Content and style will vary by discipline!

This is why it is important to read papers in the conferences/journals that you want to publish in!

You must learn your communities norms and styles

Recipe: Writing a Paper

Write a 2 paragraph abstract

- High level brain dump of problem and goals

Add titles for all sections and subsections

Outline key sections

- One bullet point per paragraph

Sketch key figures

- System design, algorithm flow
- Predicted experimental results

All the sections

Sections (your paper probably won't include all)

- Abstract - min sentences, max 2 paragraphs (200 words)
- Intro - rest of first page, spills into second
- Problem Statement / Background and Motivation
- Approach / Design / System Model
- Implementation
- Experimental Design
- Evaluation
- Discussion
- Related Work (at start or end)
- Conclusions and Future Work
- Acknowledgements
- References
- Appendix

Tips

Don’t Stop

“It was a dark and stormy night…”

Better to vomit out ideas and clean them up later

- But try not to share a draft until after you have done cleanup

Read

I’m a pretty good writer…

- But I’m not sure why

One fact: I read a lot

- and read a lot more when I was younger

Pick a book and read it!

Book recommendations

The Book of Why - Judea Pearl

History of Indus Valley?

Psyched Up

Airport

Media Control - Chomsky

Getting Things Done - Alan

The Woman Smashed Codes

Bill Bryson

Timing?

How many hours does it take to write a paper?

It takes a lot of time!

I estimate I spend ~10 hours per page I write

- Includes planning what to write, writing, revising, drawing diagrams, revising, rewriting, and rewriting a few more times

To write equally well, you probably need more time than me

Paper deadline in 2 weeks?

- How much time to spend on experiments?
- How much time to spend on writing?
- Who can help with remaining tasks?

Write, then read, then revise

Get used to rereading and revising everything

- Emails, text messages, notes

Set Gmail to delay sending

- Gives you 10 seconds to fix a messed up email

Figure out what common errors you make

Try to be a perfectionist

- But don't let this slow your flow of ideas

Ask people to correct you

- Grammar snobs are annoyingly great!

Grammar Rules

Seven Deadly Sins of Writing

- https://www.hamilton.edu/academics/centers/writing/seven-sins-of-writing

With a partner, give us a short presentation on your grammar sin

- Don’t read from the sheet!
- Give us some CS-themed examples
- Tell us how these rules fit with CS technical writing