Spring 2021

CS 424/624 Reliable Software Systems

Lecture 2: Empirical Study

Prof. Ryan Huang

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Administrivia

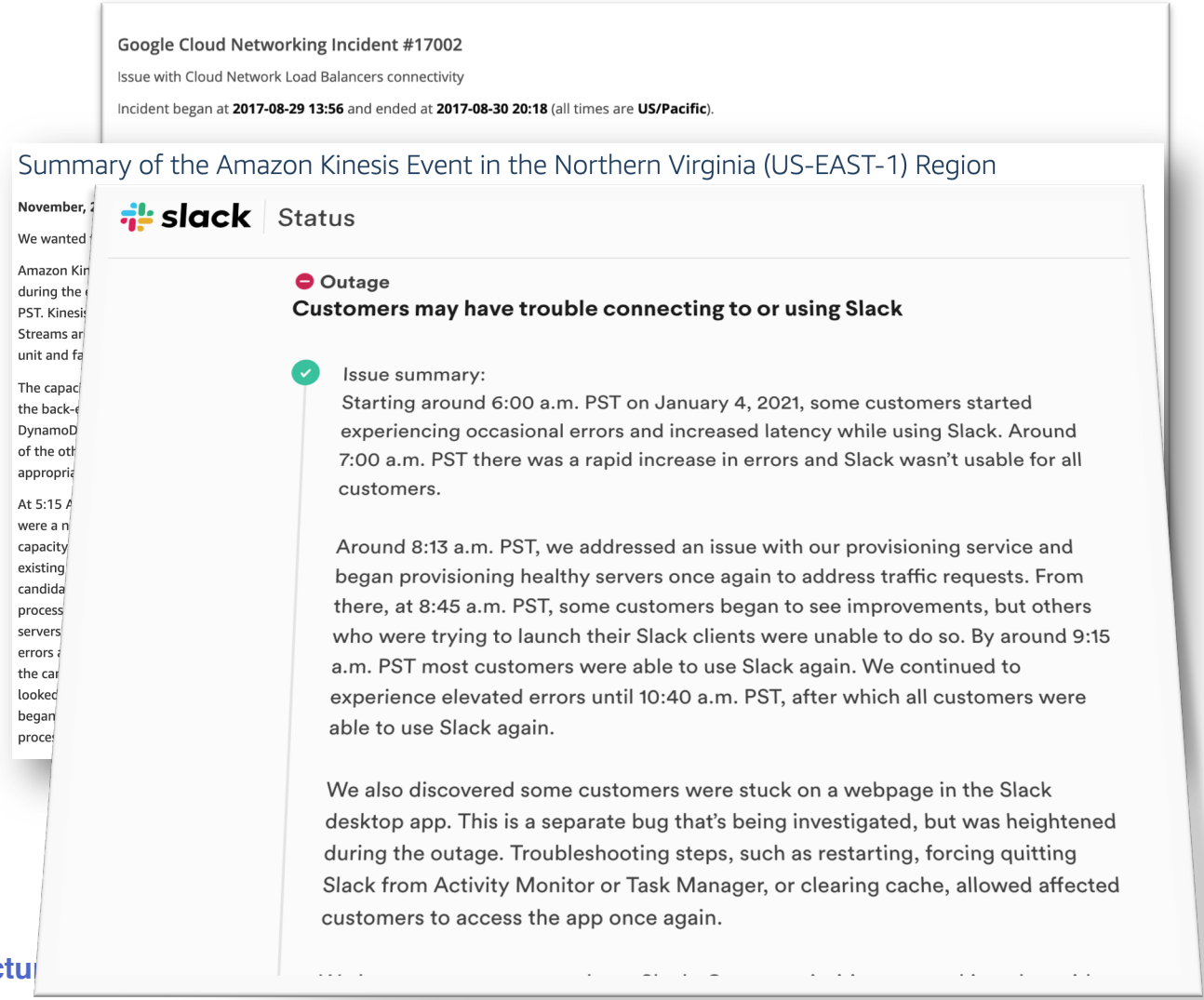
- **Sign up paper presentations**
- **Look for course project teammates**
- **Join “#paper-discussion” channel on Slack**
  - Can use Anonymous bot

Importance of Studying System Failures

- **We often learn more about reliability design from systems that failed than systems that succeeded**
  - Many surprises…
- **Study ≫ number game**
- **A good study on system failures yields deep insights that inspire solutions**
  - Which problems are important but overlooked, what the common patterns are, etc.
  - E.g., the concept of *margin of safety*

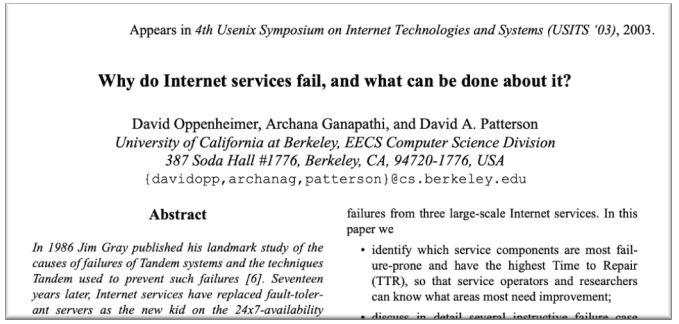
A Standard Practice in Industry

- **Regular meetings to review and analyze all major incidents**
  - A written report to summarize what went wrong, how the issue was resolved, what to improve, etc.
  - Typically called Root Cause Analysis (*RCA*) or Postmortem Analysis
  - Reports of very serious outages are usually posted publicly



A Common Research Topic In Academia

- **Numerous papers that study failures on different kinds of systems**
  - Some focus on a sub-type of failures



Appears in *4th Usenix Symposium on Internet Technologies and Systems (USITS '03)*, 2003.

Why do Internet services fail, and what can be done about it?

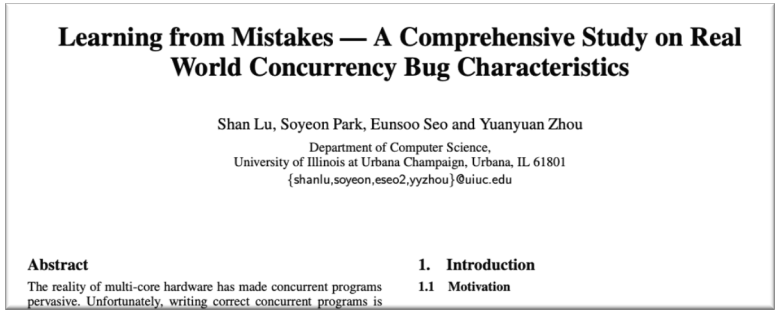
David Oppenheimer, Archana Ganapathi, and David A. Patterson
University of California at Berkeley, EECS Computer Science Division
387 Soda Hall #1776, Berkeley, CA, 94720-1776, USA
{davidopp,archanag,patterson}@cs.berkeley.edu

Abstract

In 1986 Jim Gray published his landmark study of the causes of failures of Tandem systems and the techniques Tandem used to prevent such failures [6]. Seventeen years later, Internet services have replaced fault-tolerant servers as the new kid on the 24x7-availability

failures from three large-scale Internet services. In this paper we

- identify which service components are most failure-prone and have the highest Time to Repair (TTR), so that service operators and researchers can know what areas most need improvement;



Learning from Mistakes — A Comprehensive Study on Real World Concurrency Bug Characteristics

Shan Lu, Soyeon Park, Eunsoo Seo and Yuanyuan Zhou
Department of Computer Science,
University of Illinois at Urbana Champaign, Urbana, IL 61801
{shanlu,soyeon,eseo2,yyzhou}@uiuc.edu

Abstract

The reality of multi-core hardware has made concurrent programs pervasive. Unfortunately, writing correct concurrent programs is

1. Introduction

1.1 Motivation

Why Does a Cloud-Scale Service Fail Despite Fault-Tolerance?

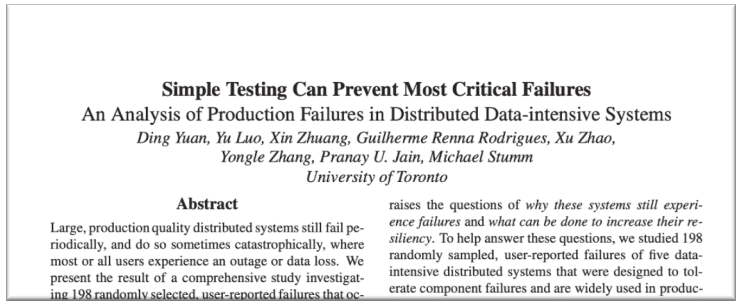
Peng Huang, Xinxin Jin, William J. Bolosky†, Yuanyuan Zhou

University of California, San Diego Microsoft Research† Microsoft‡

Abstract

The sheer scale and complexity of the cloud mean that even decades of research into fault-tolerance and software engineering for reliability, billions of dollars of invest-

tion [27, 45, 46, 51], path-redundant and failover networking [14, 39, 50], and app-specific fault handling logic [31] to detect, tolerate and recover from various faults in different layers of the systems. Additionally, careful software engineering, extensive testing and gradual roll-out



Simple Testing Can Prevent Most Critical Failures
An Analysis of Production Failures in Distributed Data-intensive Systems
Ding Yuan, Yu Luo, Xin Zhuang, Guilherme Renna Rodrigues, Xu Zhao, Yongle Zhang, Pranay U. Jain, Michael Stumm
University of Toronto

Abstract

Large, production quality distributed systems still fail periodically, and do so sometimes catastrophically, where most or all users experience an outage or data loss. We present the result of a comprehensive study investigating 198 randomly selected, user-reported failures that oc-

raises the questions of *why these systems still experience failures* and *what can be done to increase their resiliency*. To help answer these questions, we studied 198 randomly sampled, user-reported failures of five data-intensive distributed systems that were designed to tolerate component failures and are widely used in produc-

Why Does the Cloud Stop Computing?
Lessons from Hundreds of Service Outages

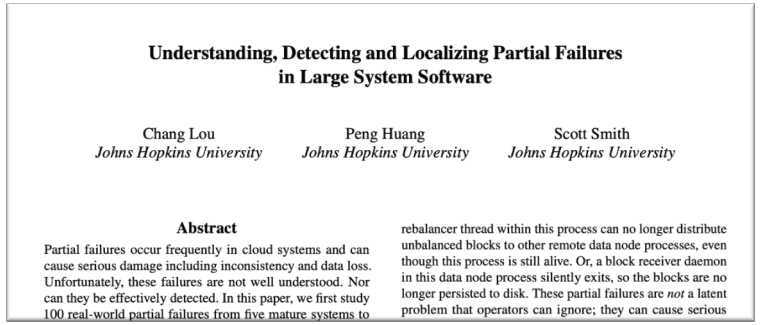
Haryadi S. Gunawi, Mingzhe Hao, and Riza O. Suminto
University of Chicago

Agung Laksono, Anang D. Satria, Jeffry Adityatama, and Kurnia J. Eliazar
Surya University

Abstract

We conducted a cloud outage study (COS) of 32 popular Internet services. We analyzed 1247 headline news and public post-mortem reports that detail 597 unplanned outages that

Not only do outages hurt customers, they also cause financial and reputation damages. Minutes of service downtimes can create hundreds of thousands of dollar, if not multi-million, of loss in revenue [29, 36, 89]. Company's



Understanding, Detecting and Localizing Partial Failures in Large System Software

Chang Lou
Johns Hopkins University

Peng Huang
Johns Hopkins University

Scott Smith
Johns Hopkins University

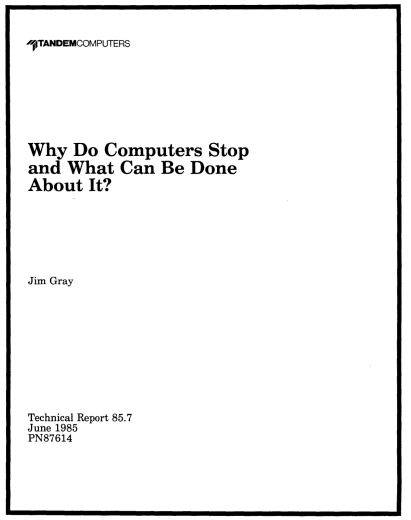
Abstract

Partial failures occur frequently in cloud systems and can cause serious damage including inconsistency and data loss. Unfortunately, these failures are not well understood. Nor can they be effectively detected. In this paper, we first study 100 real-world partial failures from five mature systems to

rebalancer thread within this process can no longer distribute unbalanced blocks to other remote data node processes, even though this process is still alive. Or, a block receiver daemon in this data node process silently exits, so the blocks are no longer persisted to disk. These partial failures are *not* a latent problem that operators can ignore; they can cause serious

Today: Two Early Empirical Studies

- One is a classic work that basically started this line of research in systems community


TANDEM COMPUTERS

Why Do Computers Stop and What Can Be Done About It?

Jim Gray

Technical Report 85.7
June 1985
PN87614

- The second is one of first systematic studies on OS bugs

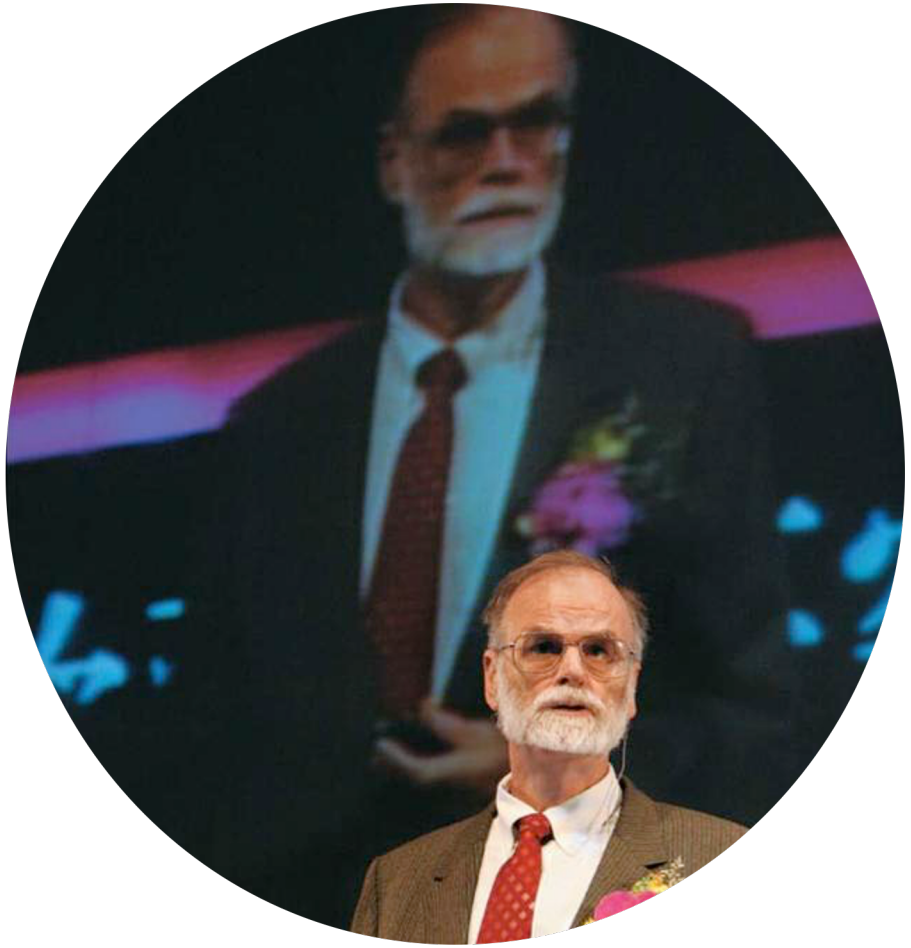

An Empirical Study of Operating Systems Errors

Andy Chou, Junfeng Yang, Benjamin Chelf, Seth Hallem, and Dawson Engler
Computer Systems Laboratory
Stanford University
Stanford, CA 94305
{acc, junfeng, bchelf, shallem, engler}@cs.stanford.edu

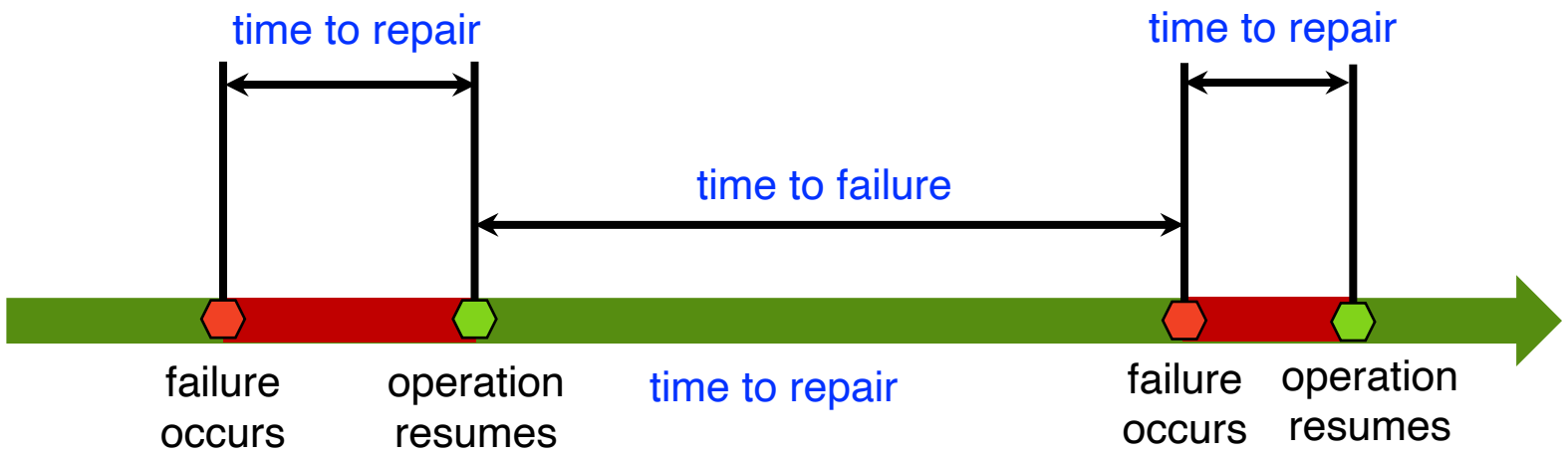
Why Do Computers Stop and What Can Be Done About It?

Author

- **Jim Gray**
  - A pioneer computer scientist
  - Received the Turing Award in 1998
    - *for seminal contributions to database and transaction processing research and technical leadership in system implementation*
  - "ACID", granular database locking, "five-minute rule" in caching, …
  - Disappeared with his sailboat in the waters in San Francisco in Jan 2007



Background: MTBF, MTTR, MTTF

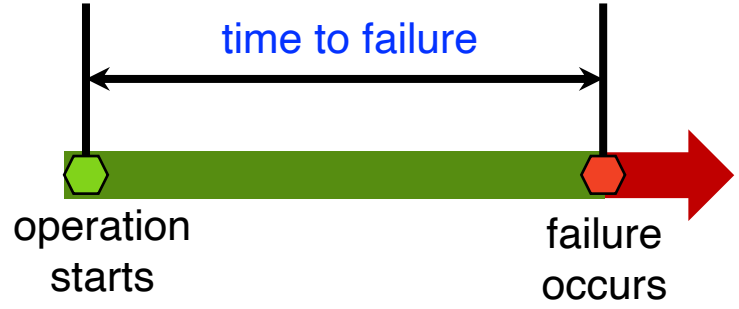


- **Mean Time *Between* Failure (MTBF)**
- **Mean Time To Recovery (MTTR)**

repairable system

- **Mean Time To Failure (MTTF)**

non-repairable system



Background: Reliability vs. Availability

- **Reliability: systems not doing wrong things**
  - proportional to MTBF
- **Availability: systems doing right things on time**
  - related to MTBF and MTTR

Definition 1

$$Avalability = \frac{E[uptime]}{E[uptime] + E[downtime]}$$

Definition 2

$$Avalability \approx \frac{MTBF}{MTBF + MTTR}$$

An Empirical Study of Operating Systems Errors

How Complex Systems Fail