SITE RELIABILITY ENGINEERING HOW GOOGLE RUNS PRODUCTION SYSTEMS

Hope is not a strategy

SYSADMINS VS SRE

- inderect cost of ops/dev vs 'breaking the wall'
- direct cost of manual interaction vs automative same approach to hire SRE as SW

PILLARS OF SRE

- Ensuring a Durable Focus on Engineering
- Pursuing Maximum Change Velocity Without Violating a Service’s SLO
- Monitoring
- Emergency Response
- Change Management
- Demand Forecasting and Capacity Planning
- Provisioning
- Efficiency and Performance

CH2 THE PRODUCTION ENVIRONMENT AT GOOGLE, FROM THE VIEWPOINT OF AN SRE

- Borg (ancestor of k8s)
- Colossus (successor of Google File System)
- Bigtable (sql like)
- Spanner (sql with consistence around the world)
- Blobstore
- Global Software Load Balancer
- ▪ geo dns load balancing
- ▪ user sercvice level
- ▪ RPC load balancing

PART 2 OF LIST SERVICES

- Chubby (distributed lock system)
- Borgmon (same approach as prometheus)
- Protocols
- ■ Stubby (RPC)
- ■ protobufs (similar to Apache’s Thrift)
- Shakespear (case to study)

PART II PRINCIPLES

CH3 EMBRACING RISK

Site Reliability Engineering seeks to balance the risk of unavailability with the goals of rapid innovation and efficient service operations

MANAGING RISK

- The cost of redundant machine/compute resources
- The opportunity cost

MEASURING SERVICE RISK

- Time-based availability = uptime/(uptime + downtime)
- Aggregate availability = successful requests/total requests

IDENTIFYING THE RISK TOLERANCE

- What level of availability is required?
- Do different types of failures have different effects on the service?
- How can we use the service cost to help locate a service on the risk continuum?
- What other service metrics are important to take into account?

MOTIVATION FOR ERROR BUDGETS

- Software fault tolerance
- Testing
- Push frequency
- Canary duration and size

Our practice is then as follows:

- Product Management defines an SLO, which sets an expectation of how much uptime the service should have per quarter.
- The actual uptime is measured by a neutral third party: our monitoring system.
- The difference between these two numbers is the “budget” of how much “unreliability” is remaining for the quarter.
- As long as the uptime measured is above the SLO—in other words, as long as there is error budget remaining—new releases can be pushed.

CH4 SERVICE LEVEL OBJECTIVES

- An SLI is a service level indicator—a carefully defined quantitative measure of some aspect of the level of service that is provided.
- An SLO is a service level objective: a target value or range of values for a service level that is measured by an SLI
- Finally, SLAs are service level agreements: an explicit or implicit contract with your users that includes consequences of meeting (or missing) the SLOs they contain.

Indicators for diff sercices could be diff too:

- User-facing serving systems: availability, latency, and throughput
- Storage systems: latency, availability, and durability
- Big data systems: throughput and end-to-end latency.

CH5 ELIMINATING TOIL

Toil is the kind of work tied to running a production service that tends to be manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as a service grows

- Manual
- Repetitive
- Automatable
- Tactical
- No enduring value
- O(n) with service growth

Our SRE organization has an advertised goal of keeping operational work (i.e., toil) below 50% of each SRE’s time. At least 50% of each SRE’s time should be spent on engineering project work that will either reduce future toil or add service features.

CH6 MONITORING DISTRIBUTED SYSTEMS

- Analyzing long-term trends
- Comparing over time or experiment groups
- Alerting
- Building dashboards
- Conducting ad hoc retrospective analysis (i.e., debugging)

Some tips:

- Symptoms Versus Causes
- Black-Box Versus White-Box
- The Four Golden Signals
- ■ Latency
- ■ Traffic
- ■ Errors
- ■ Saturation
- Choosing an Appropriate Resolution for Measurements

CH7 THE EVOLUTION OF AUTOMATION AT GOOGLE

“If we are engineering processes and solutions that are not automatable, we continue having to staff humans to maintain the system. If we have to staff humans to do the work, we are feeding the machines with the blood, sweat, and tears of human beings. Think The Matrix with less special effects and more pissed off System Administrators.” - Joseph Bironas

CH8 RELEASE ENGINEERING

Philosophy:

- Self-Service Model
- High Velocity
- Hermetic Builds
- Enforcement of Policies and Procedures
  Continuous Build and Deployment:

CH9 SIMPLICITY

- System Stability Versus Agility
- The Virtue of Boring
- ■ Push back when accidental complexity is introduced
- ■ Constantly strive to eliminate complexity in systems
- I Won’t Give Up My Code!
- The “Negative Lines of Code” Metric
- Minimal APIs
- Modularity
- Release Simplicity

PART III PRACTICES

CH10 PRACTICAL ALERTING FROM TIME-SERIES DATA

There is a deep reference of 'prometheus-like' borgmon system

CH11 BEING ON-CALL

- primary and a secondary on-call rotation
- Balance in Quantity: 25% time on call
- Balance in Quality: sufficient time to deal with any incidents and follow-up activities (postmortems) - 2 issue per 12h

Approach to deal under stress:

- Intuitive, automatic, and rapid action
- Rational, focused, and deliberate cognitive functions
  on-call resources are:
- Clear escalation paths
- Well-defined incident-management procedures
- A blameless postmortem culture

Avoiding Inappropriate Operational Load:

- Operational Overload
- ■ measure load
- ■ fix waste alerts
- ■ give back the pager
- A Treacherous Enemy: Operational Underload
- ■ ???
- ■ DiRT (Disaster Recovery Training)
- ■ “Wheel of Misfortune” exercises

CH12 EFFECTIVE TROUBLESHOOTING

- Problem Report (expected behavior, the actual behavior, how to reproduce the behavior)
- Triage (make the system work as well as it can under the circumstances (stop bleeding))
- Examine
- Diagnose
- ■ Simplify and reduce
- ■ Ask “what,” “where,” and “why”
- ■ What touched it last
- ■ Specific diagnoses
- Test and Treat

Negative Results Are Magic

- Negative results should not be ignored or discounted
- Experiments with negative results are conclusive
- Tools and methods can outlive the experiment and inform future work
- Publishing negative results improves our industry’s data-driven culture
- Publish your results

CH13 EMERGENCY RESPONSE

This chapter dedicated to some real exercise for detailed explanation how to deal in case of problem

- Test-Induced Emergency
- Change-Induced Emergency
- Process-Induced Emergency

and general plan:

- Details
- Response
- Findings:
- What went well / What we learned

Learn from the Past. Don’t Repeat It:

- Keep a History of Outages
- Ask the Big, Even Improbable, Questions: What If…?
- Encourage Proactive Testing

CH14 MANAGING INCIDENTS

Google loves komitets! Elements of Incident Management Process:

- Recursive Separation of Responsibilities
- ■ Incident Command
- ■ Operational Work
- ■ Communication
- ■ Planning
- A Recognized Command Post
- Live Incident State Document
- Clear, Live Handoff

Best Practices for Incident Management:

- Prioritize
- Prepare
- Trust
- Introspect
- Consider alternatives
- Practice
- Change it around

CH15 POSTMORTEM CULTURE: LEARNING FROM FAILURE

- User-visible downtime or degradation beyond a certain threshold
- Data loss of any kind
- On-call engineer intervention (release rollback, rerouting of traffic, etc.)
- A resolution time above some threshold
- A monitoring failure (which usually implies manual incident discovery)

Collaborate and Share Knowledge

- Real-time collaboration
- An open commenting/annotation system
- Email notifications

Introducing a Postmortem Culture

- Postmortem of the month
- Google+ postmortem group
- Postmortem reading clubs
- Wheel of Misfortune

CH16 TRACKING OUTAGES

- Escalator/Outalator
- Aggregation
- Tagging
- Analysis

CH17 TESTING FOR RELIABILITY

- Traditional Tests
- ■ Unit tests
- ■ Integration tests
- ■ System tests
- ■ ○ Smoke tests
- ■ ○ Performance tests
- ■ ○ Regression tests
- Production Tests
- ■ Configuration test
- ■ Stress test
- ■ Canary test

CH18 SOFTWARE ENGINEERING IN SRE

- design and create software with the appropriate considerations
- embedded in the subject matter easily understand the needs and requirements of the tool being developed
- A direct relationship with the intended user—fellow SREs—results in frank and high-signal user feedback

CH19 LOAD BALANCING AT THE FRONTEND

Load Balancing Using DNS

- Recursive resolution of IP addresses
- Nondeterministic reply paths
- Additional caching complications

Load Balancing at the Virtual IP Address IPv4+GRE

CH20 LOAD BALANCING IN THE DATACENTER

- A Simple Approach to Unhealthy Tasks: Flow Control
- A Robust Approach to Unhealthy Tasks: Lame Duck State
- Limiting the Connections Pool with Subsetting
- ■ Picking the Right Subset (A Subset Selection Algorithm: Deterministic Subsetting)
- Load Balancing Policies
- ■ Simple Round Robin
- ■ Least-Loaded Round Robin
- ■ Weighted Round Robin

CH21 HANDLING OVERLOAD

- The Pitfalls of “Queries per Second”
- Per-Customer Limits
- Client-Side Throttling
- Criticality
- Utilization Signals
- Handling Overload Errors
- ■ per-request retry budget
- ■ per-client retry budget

CH22 ADDRESSING CASCADING FAILURES

Preventing Server Overload

- Load test the server's capacity limits, and test the failure mode for overload
- Serve degraded results
- Instrument the server to reject requests when overloaded
- Instrument higher-level systems to reject requests, rather than overloading servers

CH23 MANAGING CRITICAL STATE: DISTRIBUTED CONSENSUS FOR RELIABILITY

Replica state machine and Paxos
ACID (Atomicity, Consistency, Isolation, and Durability) vs BASE (Basically Available, Soft state, Eventual consistency)

CH24 DISTRIBUTED PERIODIC SCHEDULING WITH CRON

Storing the State

CH25 DATA PROCESSING PIPELINES

Data pipelines:

- coroutines
- DTSS communication files
- UNIX pipe
- ETL pipelines
- “Big Data,” or “datasets that are so large and so complex that traditional data processing applications are inadequate.”

Drawbacks of Periodic Pipelines in Distributed Environments:

- Monitoring Problems in Periodic Pipelines
- “Thundering Herd” Problems
- Moiré Load Pattern

CH26 DATA INTEGRITY: WHAT YOU READ IS WHAT YOU WROTE

Trade-offs between:

- uptime
- latency
- scale
- velocity
- privacy

How Google SRE Faces the Challenges of Data Integrity

The 24 Combinations of Data Integrity Failure Modes:

- First Layer: Soft Deletion
- Second Layer: Backups and Their Related Recovery Methods
- Overarching Layer: Replication
- 1T Versus 1E: Not “Just” a Bigger Backup
- Third Layer: Early Detection
- Knowing That Data Recovery Will Work

GENERAL PRINCIPLES OF SRE AS APPLIED TO DATA INTEGRITY

- General Principles of SRE as Applied to Data Integrity
- Trust but Verify
- Hope Is Not a Strategy
- Defense in Depth

CH28 RELIABLE PRODUCT LAUNCHES AT SCALE

committees! google loves committees
Case of study: NORAD (the North American Aerospace Defense Command) to host a Christmas-themed website that tracked Santa’s progress around the world

LAUNCH COORDINATION ENGINEERING

- Breadth of experience
- Cross-functional perspective
- Objectivity

SETTING UP A LAUNCH PROCESS

- Lightweight
- Robust
- Thorough
- Scalable
- Adaptable

Tactics

- Simplicity
- A high touch approach
- Fast common paths
- The Launch Checklist

PART IV MANAGEMENT

ACCELERATING SRES TO ON-CALL AND BEYOND

Recommended patterns	**Anti-patterns**
Designing concrete, sequential learning experiences for	

students to follow|Deluging students with menial work (e.g., alert/ticket triage) to train them; “trial by fire”| |Encouraging reverse engineering, statistical thinking, and working from fundamental principles|Training strictly through operator procedures, checklists, and playbooks| |Celebrating the analysis of failure by suggesting postmortems for students to read|Treating outages as secrets to be buried in order to avoid blame| |Creating contained but realistic breakages for students to fix using real monitoring and tooling|Having the first chance to fix something only occur after a student is already on-call| |Role-playing theoretical disasters as a group, to intermingle a team’s problem-solving approaches|Creating experts on the team whose techniques and knowledge are compartmentalized| |Enabling students to shadow their on-call rotation early, comparing notes with the on-caller|Pushing students into being primary on-call before they achieve a holistic understanding of their service| |Pairing students with expert SREs to revise targeted sections of the on-call training plan|Treating on-call training plans as static and untouchable except by subject matter experts| |Carving out nontrivial project work for students to undertake, allowing them to gain partial ownership in the stack|Awarding all new project work to the most senior SREs, leaving junior SREs to pick up the scraps| Creating Stellar Reverse Engineers and Improvisational Thinkers:

- Reverse Engineers: Figuring Out How Things Work
- Statistical and Comparative Thinkers: Stewards of the Scientific Method Under Pressure
- Improv Artists: When the Unexpected Happens
- Tying This Together: Reverse Engineering a Production Service
  Five Practices for Aspiring On-Callers:
- A Hunger for Failure: Reading and Sharing Postmortems
- Disaster Role Playing
- Break Real Things, Fix Real Things
- Documentation as Apprenticeship
- Shadow On-Call Early and Often

DEALING WITH INTERRUPTS

did you read 'Time management for system admins'?

EMBEDDING AN SRE TO RECOVER FROM OPERATIONAL OVERLOAD

Phase 1: Learn the Service and Get Context

- Identify the Largest Sources of Stress
- Identify Kindling

Phase 2: Sharing Context

- Write a Good Postmortem for the Team
- Sort Fires According to Type

Phase 3: Driving Change

- Start with the Basics
- Get Help Clearing Kindling
- Explain Your Reasoning
- Ask Leading Questions

COMMUNICATION AND COLLABORATION IN SRE

So important, i'll leave this blank

THE EVOLVING SRE ENGAGEMENT MODEL

SRE Engagement: What, How, and Why The SRE Engagement Model:

- System architecture and interservice dependencies
- Instrumentation, metrics, and monitoring
- Emergency response
- Capacity planning
- Change management
- Performance: availability, latency, and efficiency

The PRR Model

LESSONS LEARNED FROM OTHER INDUSTRIES

Preparedness and Disaster Testing

- Relentless Organizational Focus on Safety
- Attention to Detail
- Swing Capacity
- Simulations and Live Drills
- Training and Certification
- Focus on Detailed Requirements Gathering and Design
- Defense in Depth and Breadth

ME

Maxim Ermolenko tg - @flomko presentation
https://github.com/FloMko/sre-workshop