Canary in a Coal Mine:
Building infrastructure resiliency with canary data reloads

Ann Kilzer – Site Reliability Engineer
I help people get jobs.
Indeed Products

60 countries

30 languages

200 million UVs

20 million jobs
 indeed

what

job title, keywords or company name

where

San Francisco, CA

Find Jobs

771,136 new jobs in the last 7 days
Search job sites, newspapers, associations and company career pages.

Indeed también está disponible en español

Indeed helps people get jobs: Over 10 million stories shared
Site Reliability Engineer - New
Imx - San Francisco, CA
14000+ (210)
115,000+ (157)
120,000+ (133)
130,000+ (71)
Full-time (973)
Contract (18)
Temporary (5)
Internship (1)
Part-time (1)
San Francisco, CA (217)
Redwood City, CA (17)
San Mateo, CA (12)
Hayward, CA (9)
San Francisco Bay Area, CA (6)
more >
Search Engine
Search Engine

- Java
- Lucene
- Homogenous servers
- Multiple datacenters
Job Index

- Example of a “data artifact”
  - Inverted Index and related files
  - Published by Index Builder
Data Artifacts

- Read-optimized data segments
- Lucene, csv, other binary formats
- Published periodically by builders
- Labeled by generation: G1, G2, G3...
Artifact Distribution

Jobseekers
Requests
Servers
Heap
Disk
Artifact Distribution

Jobseekers

Requests

Servers

Heap

Disk
Job Index Heap Usage

memUsed

Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep

3,600,000,000 4,000,000,000 3,600,000,000 4,800,000,000 5,200,000,000 5,600,000,000
In six months:

- Heap usage grew from 3.7 GB to around 6 GB
- Remained around 15 million jobs
- Per-job cost increased from 273 bytes to around 400 bytes
Job Index Artifact Growth

- Quantity
  - Aggregation
  - Employer posts
- Metadata
  - Parallel lookup tables
  - Job refinements, estimated salaries, etc.
Job Index growth caused full GC loops

**Graph:**
- **Y-axis:** memUsed/memMax
- **X-axis:** Time (1m)
- **Values:**
  - 6 PM: memUsed/memMax
  - 7 PM: memUsed/memMax
  - 8 PM: memUsed/memMax (increases)
  - 9 PM: memUsed/memMax (returns to baseline)
  - 10 PM: memUsed/memMax (returns to baseline)
Sponsored Jobs Info
Sponsored Jobs Info Data Artifact Growth

Size (MB)

1/1/16 2/1/16 3/1/16 4/1/16 5/1/16 6/1/16 7/1/16 8/1/16
Sponsored Jobs Info Data Artifact OOME

- August 2016
  - Artifact sizes gradually rising
  - Peak Traffic
  - High heap usage → OOME across three US datacenters
The Cost of an OOME

- Site outage for US and Canada
  - Complete outage for 50 minutes, partial for 1 hour 30 minutes
  - Estimated losses:
    - Many hundreds of thousands of job clicks
    - Many thousands of applies
We did not help people get jobs
How can we prevent catastrophic failure?
How can we prevent catastrophic failure?
Canary Reloading
1. Load the artifact
2. If successful, “bless” the run
3. After blessing, other servers load the artifact
4. If unsuccessful: try again
Jobseekers

Requests

Servers

Heap

Disk

Artifact Distribution

A

B

C

wait

wait

G1

G2

G1

G1

G1

G1

G2

G2

G2

G2
Artifact Distribution

Jobseekers
Requests
Servers
Heap
Disk
Artifact Distribution

Jobseekers
Requests
Servers
Heap
Disk

A
B
C

G1
G2
A server crash does not imply a reload failure.
Consistent Distributed Lock Service

I see A is working!

owner: A
blessed: no
timeout: in 2 minutes

A

B
wait

C
wait
A hasn't updated the lock. I'm going to clean it up!

owner: A
blessed: no
timeout: expired

Consistent Distributed Lock Service
Artifact Distribution
Worst Case

- $n - 1$ capacity
  - Healthcheck alerts if this situation persists
- Outdated data
Other applications

- Exceptions
- Corrupted Artifacts
- Segfaults
- Other system metrics indicating failure
The Algorithm
while not reloaded:
  try to claim the canary lock
  if I am the canary:
    execute the reloader
    bless and return
  if lock is expired, clean it up
  is the lock blessed?
    execute the reloader
  should I abort?
  break

Basic Algorithm
Basic Algorithm

while not reloaded:
  try to claim the canary lock
  if I am the canary:
    execute the reloader
    bless and return
  if lock is expired, clean it up
  is the lock blessed?
    execute the reloader
  should I abort?
    break
Try to claim a canary lock

- Consistent distributed key-value store
- Only one server can grab the lock
- Key: artifact name
- Value:
  - Owner: hostname
  - Blessed: false
  - Expiry: 2 minutes from now
Basic Algorithm

while not reloaded:
    try to claim the canary lock
    if I am the canary:
        execute the reloader
        bless and return

    if lock is expired, clean it up
    is the lock blessed?
    execute the reloader
    should I abort?
    break
If I am the Canary

- “Keep Alive” signal
- Start a background thread to continually update the lock
  - Key: artifact name
  - Value:
    - Owner: hostname
    - Blessed: false
    - Expiry: 2 minutes from now
If I am the Canary

- Execute the function with artifact reloading logic
- Ensure successful run
If I am the Canary

- When successful, update the lock to bless:
  - Owner: hostname
  - Blessed: true
  - Expiry: 24 hours from now
- Exit loop
while not reloaded:
  try to claim the canary lock
  if I am the canary:
    execute the reloader
    bless and return

if lock is expired, clean it up
is the lock blessed?
  execute the reloader
should I abort?
  break
If I’m not the Canary (Part 1)

- Is the lock expired?
  - Clean it up and restart the loop
- Gives every host a chance to grab the lock
while not reloaded:
    try to claim the canary lock
    if I am the canary:
        execute the reloader
        bless and return
    if lock is expired, clean it up
    is the lock blessed?
    execute the reloader
    should I abort?
    break
If I’m not the Canary (Part 2)

- Is the lock blessed?
  - Execute the reload function
  - break loop
while not reloaded:
    try to claim the canary lock
    if I am the canary:
        execute the reloader
        bless and return
    if lock is expired, clean it up
    is the lock blessed?
    execute the reloader
    should I abort?
    break
If I’m not the Canary (Part 3)

- Should I abort?
  - Safety vs. Liveness -- How long should we wait to load an artifact?
  - How long has a new artifact been available on disk?
  - Has SRE / Ops had time to react?
while not reloaded:
  try to claim the canary lock
  if I am the canary:
    execute the reloader
    bless and return
  if lock is expired, clean it up
  is the lock blessed?
    execute the reloader
  should I abort?
    break
Repeat!

- Repeat the loop until the reload is successful
Implementation at Indeed
Distributed Lock Service

● Key Value store
  ○ Key: artifact name
  ○ Value: {
    "owner": "hostname",
    "blessed": false,
    "expiry": 1498164251000
  }

● Atomic Writes

● Consistent

● Check-and-Set value 0 for leader election (First come, first serve)
Distributed Lock Service
Implementation at Indeed

- Job Index
- Generic Library
- Expired Jobs
- In progress:
  - Canary reload all the things
Over 4000 runs in production
Learning from Obstacles
Case Study: Complex Artifact

- Index data entangled with job sponsor data
  - Bad publish of minor component blocked canary reload
  - Don’t let your billing data get stale
Case Study: Complex Artifact

- Solution
  - Refactor & break up loads to simplify complex artifacts
  - Healthchecks on block time
Case Study: Bless and Die

- Surge in OOME in a datacenter
- Enough heap to bless, but not enough to run code immediately after
Case Study: Bless and Die

Solution:
- Wait to bless
- Test alloc
- System Introspection
Fine Tuning
Tips for success

- Infrastructure
  - Able to run at n -1 capacity
  - Periodically review heap space
Tips for success

● Good logging
  ○ Who is the canary
  ○ What is blessed

● Healthchecks
  ○ System block
  ○ Distributed lock dependency
Conclusion
Conclusion

- Canary Reloading is a technique that you can apply to safeguard data reloads
- And more
  - Exceptions
  - Other failure conditions
- No birds harmed in the creation of this tool
Q&A
Canary in a Coal Mine:
Building infrastructure resiliency with canary data reloads

Ann Kilzer – Site Reliability Engineer