Improving the robustness of our service

Daniel Gomes
What is Arquivo.pt?

Web pages preserved since 1996

Public search service

Information in several languages
Brief history of Arquivo.pt

2007: Project launch
2010: Search prototype publicly available

9/2013: Service collapsed due to hardware malfunction

- Data loss of 17% (17 TB)
- Crawling interruptions
- Suspension of search service

2014 - 2016: Recovery and improving robustness
Now, we can share our experience.
Arquivo.pt system overview
Our web archiving workflow is mainly automatic

1. **Live-web data**
2. **Crawler**
3. **Web-archived data (ARC files)**
4. **Indexer**
5. **Indexes**
6. **Reproduction of preserved web-content**
7. **Temporal Search**
Arquivo.pt is a medium-size web archive

Hardware

85 servers

Archived data

4 billion files

468 TB (ARC files, indexes, replication)

Estimated data growth

72 TB/year
5 measures to improve the robustness of Arquivo.pt
Hardware and software architecture shifted to *Shared-Nothing* (#1)
Design-to-fail: the failure of a single equipment cannot jeopardize the service

Centralized: blade server enclosures + storage arrays

Distributed (shared-nothing): independent rack servers
Inefficient physical space management at the data center with blade systems

Space that was never used

Space still occupied after servers disabled
Independent rack servers

Only operational servers occupy physical space

Physical space is released as servers break
Perform load tests immediately after buying to induce failures

Open source tools: bonnie (disk), stress (CPU), memtest (Memory)

Bathtub curve: identify Early Failures during the warranty period
Segregate development from production networks

Private network

Gateway between networks

Public network

Development environment

Quality assurance and production environments
Reinforced replication policies (#2)
Tape

Offline backup
Bundle backup to tape every 4 months
ARC files, indexes

Random test recoveries from tape

Data recovery from tape is very slow
Hard disks

Online backups

Redundant server disks (RAID-5)

All data is replicated across 2 independent servers

ARC files, indexes, software

Daily backup during crawl on live hard disks

Lose at most 1 day of crawled data
Distant location backups

Tapes moved to distant geographical location

Lisbon to Porto: 275 KM

ARC files copied to the Internet Archive through the Internet

Lisbon to California: 9 000 KM
Monitor the service (#3)
Monitoring tools fail

The service is broke but we didn’t know

So we did not fix it

Who monitors the monitoring tools?
Use redundant monitoring tools

Hardware failures
  Vendor tools are not enough

Hardware resources
  Cacti and Ganglia

Service availability
  Nagios and Uptime Robot (external)

Access statistics
  Awstats and Google Analytics (external)
Induce faults to test monitoring!

It’s better to identify problems when you are ready for them
Quality Assurance for software development (#4)
People get tired from doing repeatedly the same (testing). *Computers don’t.*
Code testing: automatize

Compilation: the code is well written!

Unit: does what it supposes to do!

Functional: makes the service work

Simulate user workflows (e.g. search for an archived page)

Many free and powerful tools to automatize testing

SeleniumHQ, SauceLabs, Jenkins, SonarCube
Workload capacity testing: automatize

Establish minimum thresholds for new service release

Jmeter

Workload average: 3 responses/second

Speed average: 5 seconds per response
Security testing: automatize

It’s not “if we get attacked”, it’s “when we get attacked”

OWASP Zed Attack Proxy (ZAP)

Expert reviews
Usability testing: conducted by skilled professionals

What is the use of a service that users cannot use?

Identify the problems that really affect the service

Most technical problems are reflected on usability obstacles

Help from Human Computer Interaction group from University of Lisbon and UX training
Document and test procedures (#5)
Different types of documentation for different purposes

*Wiki*: internal procedures

*GitHub*: software

*Reports*: analysis

*Internal and external presentations*: collaborations

*Scientific and technical publications*: peer-review
Test the documentation

Installations of software components from scratch

Procedures executed by colleagues based on existing documentation without help
Open source everything we do

github.com/arquivo

Increases responsibility

Increases software quality
Results
Crawling and indexing are stable
Search availability in 2016

100%
Recovering our users

4 090 users per month (average)

Gaining new users

90% are new users
Lessons learned

Strict *Shared-nothing* architecture for hardware and software

Replicate data on multiple distinct media

Software development without proper Quality Assurance leads to waste of resources

Test everything, every time, automatically.

Accept staff rotation and proactively prepare for it