AARHUS UNIVERSITET

Microservices and DevOps

Scalable Microservices

Stability Antipatterns

Henrik Bærbak Christensen

AARHUS UNIVERSITET

Release It!

- Written by what is obviously a highly skilled practitioner ☺

- Definitely not written by an academic ☺? ☹ ?
  - I have had to write most of the definitions to pinpoint the terminology used…
  - Each section does *not* follow a pattern template
    - (name, problem, solution, forces)

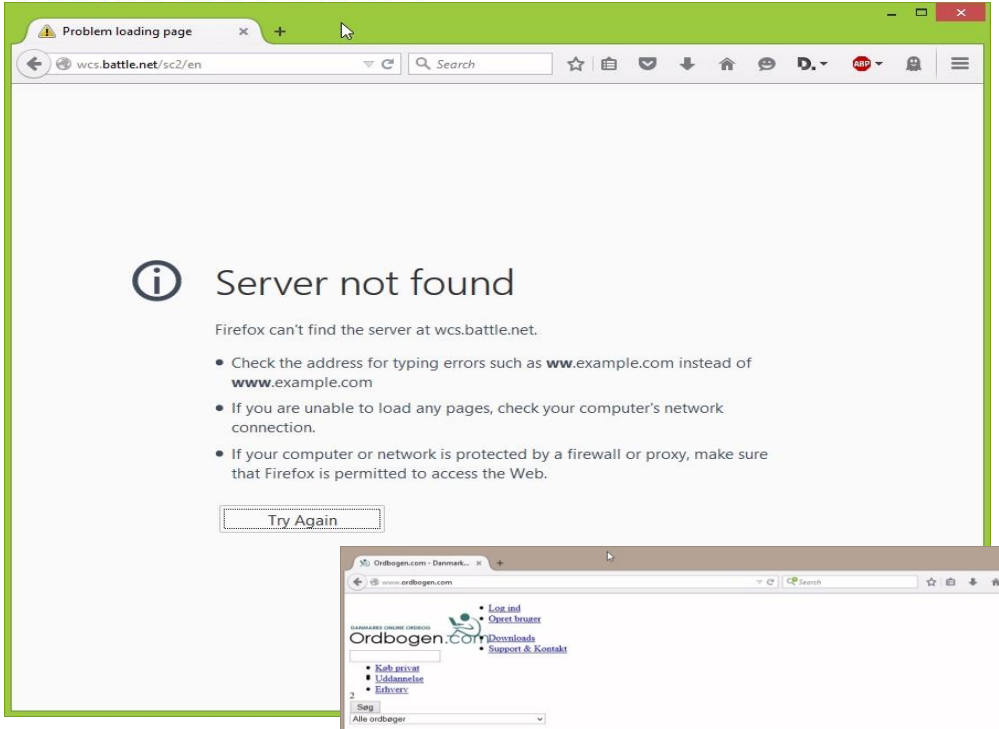
Punch line…

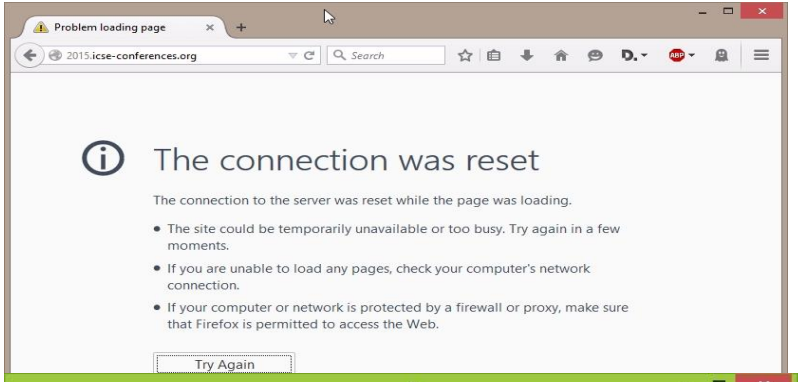
- *Cynical software expects bad things to happen and is never surprised when they do. Cynical software doesn't even trust itself.*

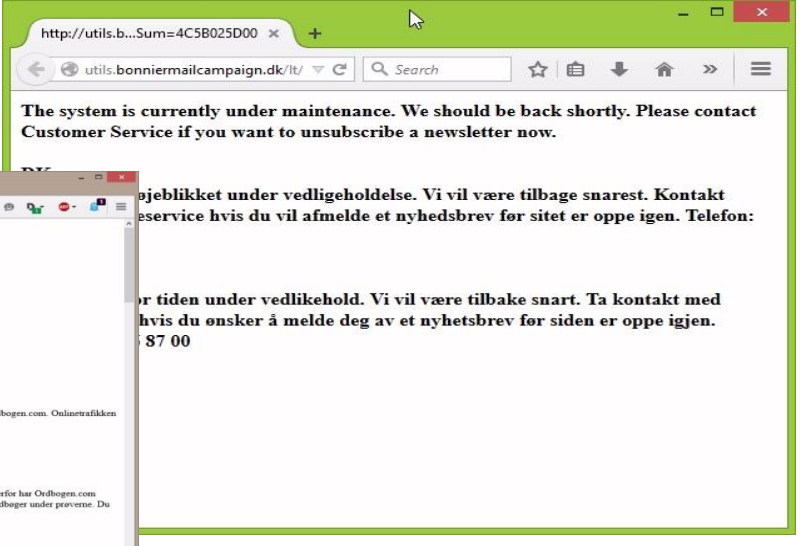
- *A highly stable design usually costs the same to implement as an unstable one.*
  - Hm? I think Nygard means 'the total cost over lifetime' here…

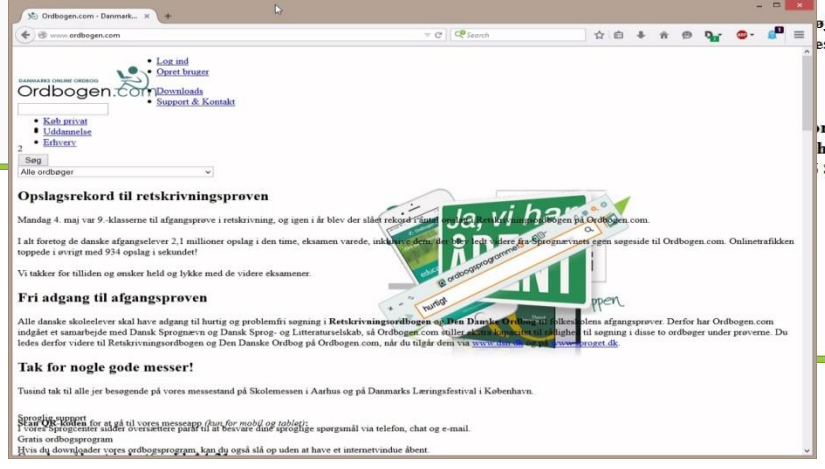
AARHUS UNIVERSITET

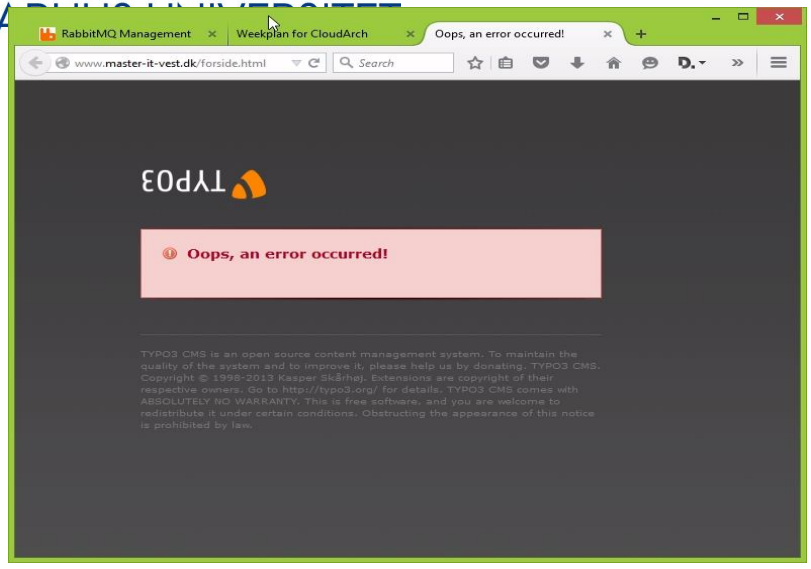
About a half year of 'ups'

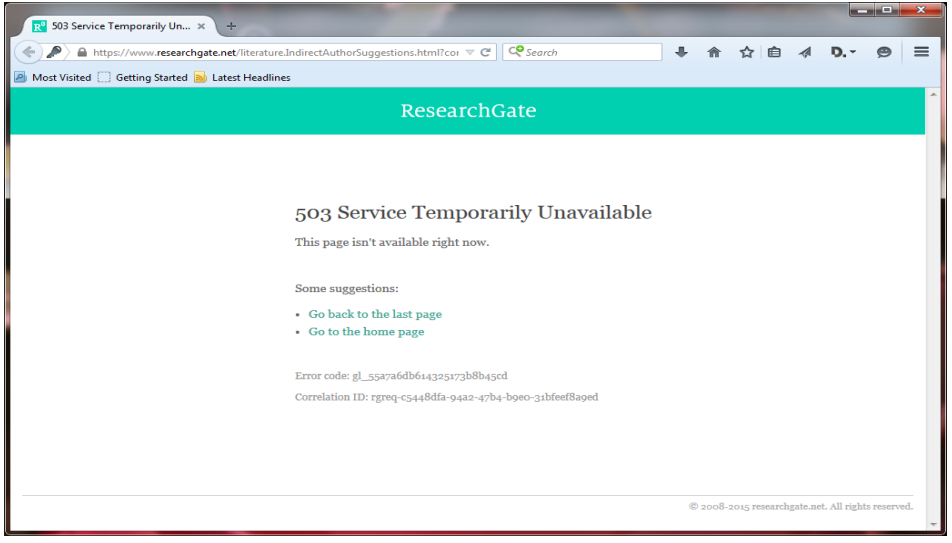


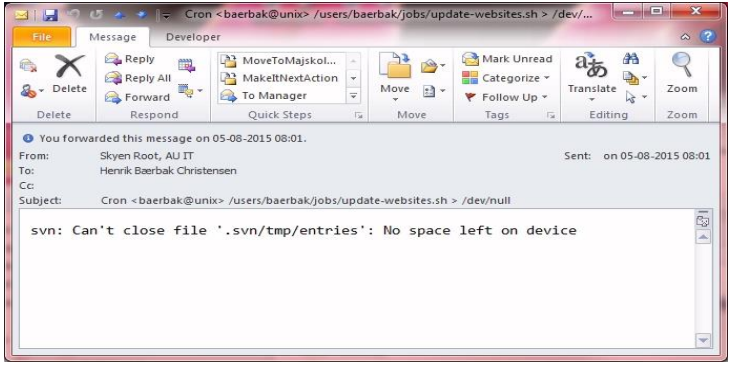




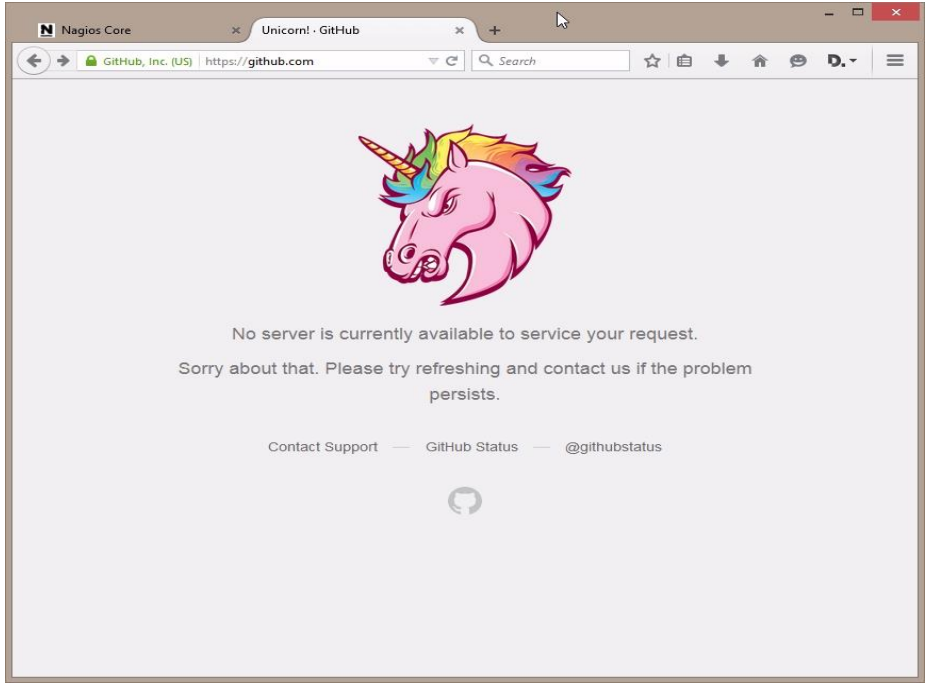


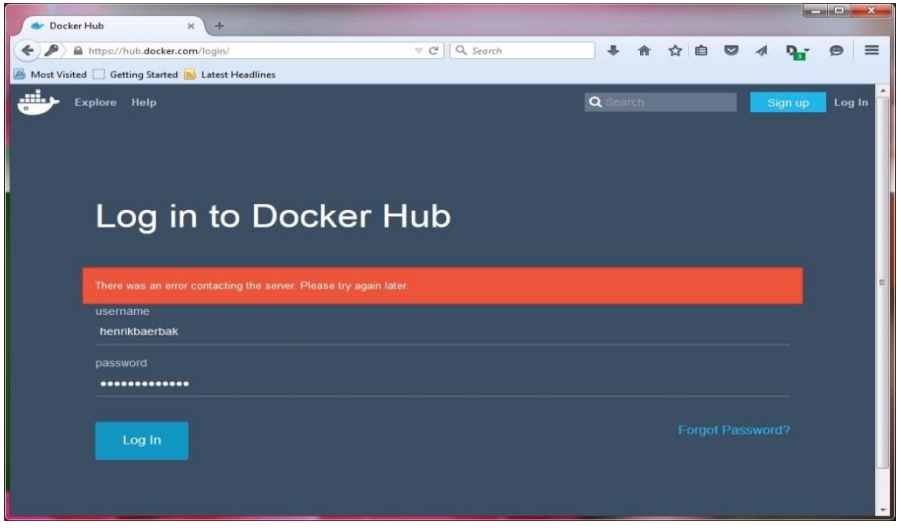

RabbitMQ Management
Weekplan for CloudArch
Oops, an error occurred!
www.master-it-vest.dk/forside.html
Search
TYPO3
Oops, an error occurred!
TYPO3 CMS is an open source content management system. To maintain the quality of the system and to improve it, please help us by donating. TYPO3 CMS. Copyright © 1998-2013 Kasper Skårhøj. Extensions are copyright of their respective owners. Go to http://typo3.org/ for details. TYPO3 CMS comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to redistribute it under certain conditions. Obstructing the appearance of this notice is prohibited by law.

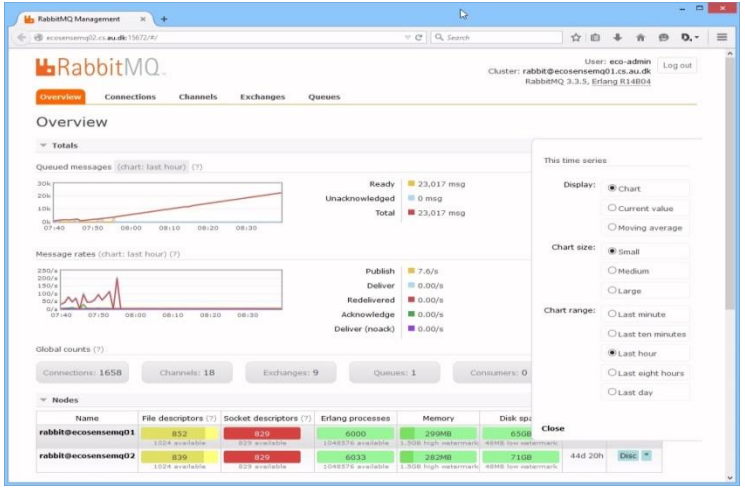

503 Service Temporarily Un...
https://www.researchgate.net/literature.IndirectAuthorSuggestions.html?co
Search
Most Visited
Getting Started
Latest Headlines
ResearchGate
503 Service Temporarily Unavailable
This page isn't available right now.
Some suggestions:
Go back to the last page
Go to the home page
Error code: gl_55a7a6db614325173b8b45cd
Correlation ID: rgreq-c5448dfa-94a2-47b4-b9e0-31bfeef8a9ed
© 2008-2015 researchgate.net. All rights reserved.


Cron <baerbak@unix> /users/baerbak/jobs/update-websites.sh > /dev/...
File
Message
Developer
Reply
Reply All
Forward
Delete
MoveToMajskol...
MakeItNextAction
To Manager
Move
Mark Unread
Categorize
Follow Up
Translate
Zoom
Delete
Respond
Quick Steps
Move
Tags
Editing
Zoom
You forwarded this message on 05-08-2015 08:01.
From: Skyen Root, AU IT
Sent: on 05-08-2015 08:01
To: Henrik Bærbak Christensen
Cc:
Subject: Cron <baerbak@unix> /users/baerbak/jobs/update-websites.sh > /dev/null
svn: Can't close file '.svn/tmp/entries': No space left on device

AARHUS UNIVERSITET







More recent

47 MIN. SIDEN | EMIL ELLER

Lige nu kan man ikke se corona-testsvar på sundhed.dk eller i MinSundhed

Her til morgen er det ikke muligt at se svar på coronatest på
MinSundhed.

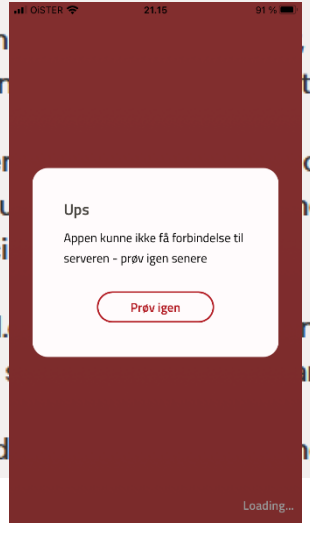
Det skriver myndighederne på sundhed.dk.

Derfor kan man , hvis man skal fremvis
coronapas. Man tus for, om man tidlige

- Det er desvær coronapas for covid-1 covid-19-
prøvesvar på su ndhed lige nu. Det er st pas for
covid-19-vacci

Ifølge sundhed. ne, at der er overbelas s Statens
Serum Institut, ar hos.

Der er ingen tid ge problemet kommer til at vare.



En mexicaner kigger på sin telefon mandag i Mexico By.
(Foto: ALFREDO ESTRELLA © Ritzau Scanpix)

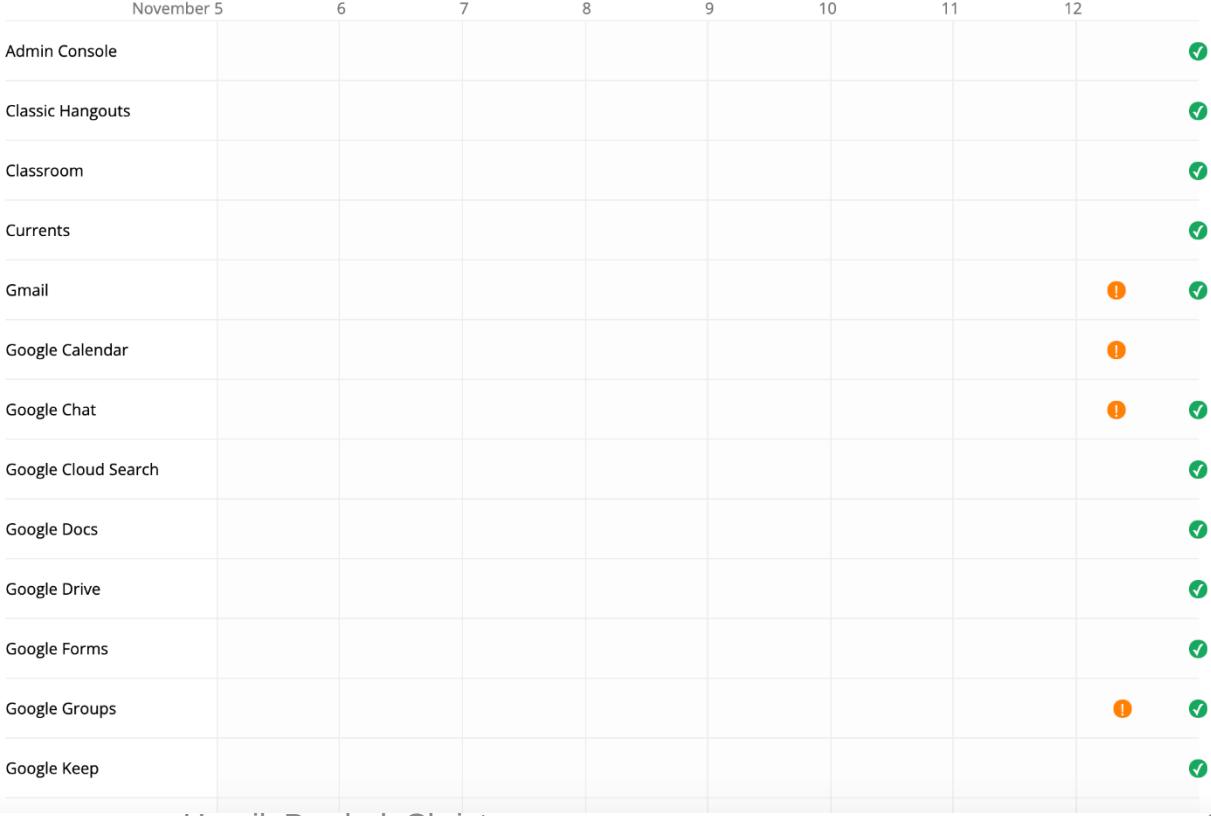
AARHUS UNIVERSITET

Google Workspace

Google Workspace Status Dashboard

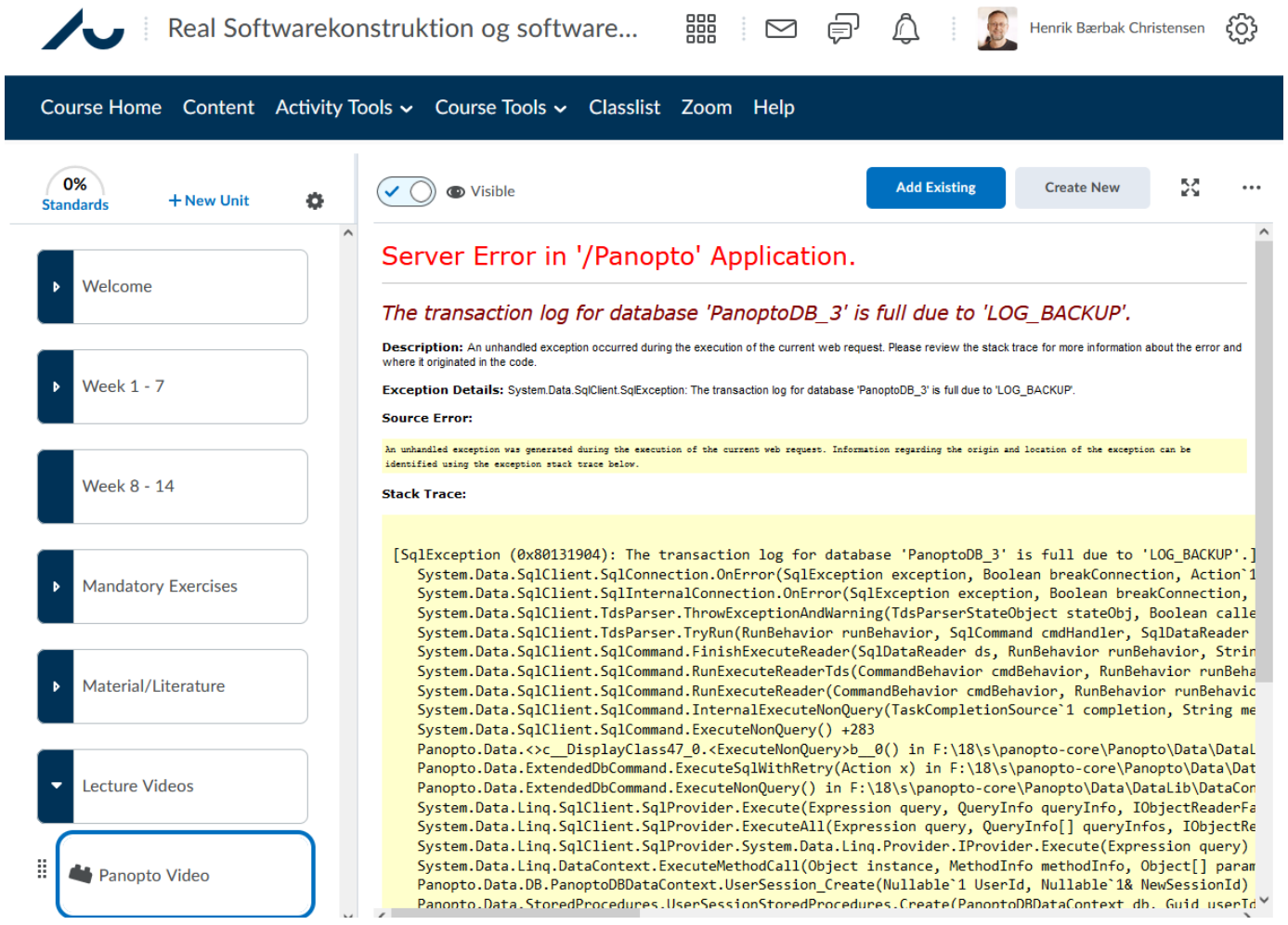
We are currently investigating an issue affecting user access to multiple services affecting users in Europe.

This page provides status information on the services that are part of Google Workspace. Check back here to view the current status of the services listed below. If you are experiencing an issue not listed here, please contact Support. Learn more about what's posted on the dashboard in this FAQ. For additional information on these services, please visit https://workspace.google.com/. For incidents related to Google Analytics, visit the Google Ads Status Dashboard.



AARHUS UNIVERSITET

And



AARHUS UNIVERSITET

Akin to Nygard §1

- MSDO 2020 Crunch broke down. Why?
  - No try catch around ‘network.close()’ in my finally() clause ☹

```
} finally {
  // Bug? Guard against com.github.dockerjava.api.exception.NotFoundException:
  // {"message":"could not find an available, non-overlapping IPv4 address pool among the defaults to assign to the network"}
  try {
    logger.info("method=assess, context=finally");

    if (network != null) network.close();
    if (daemon != null) daemon.close();
```

  - Was due to the ‘only 16 networks available’ default docker setup

AARHUS UNIVERSITET

Another one

- Crunch3.baerbak.com
  - The central submission server
- Fixed IP in my own range, provided by AUIT
- During maintenance September 2016 they *shut down this range's DNS name servers* without telling me!
- Result: *All* my servers could not see the internet!
  - And no warning at all!

Morale: My software in production *change even when I do not change my software!*

Definitions

- *Transaction*: Abstract unit of work processed by the system
  - *More akin to 'story' (XP) or 'use case'*
- *System:* Complete, interdependent set of hardware, applications, networks, power supplies, and services required to process transactions
- *Impulse:* Rapid shock to the system
- *Stress:* Force applied to the system over extended time

Reusing *mechanical engineering* terminology

Definitions

- *Strain*: Changed shape when stress is applied.
- *Longevity:* Ability to keep processing transactions for a long time [long = more than timespan between deployments]
- *Crack:* essentially synonym with failure
- *Failure mode:* the trigger, the way cracks propagate, and the result of the damage
  - *Safe failure modes* are better than unpredictable failure modes!
- *Crackstoppers:* failure modes that keep cracks away from indispensable system parts

Examples

- Straining the system
  - Requests get queue up
    - Queues grow
      - RAM exhausted – start swapping to disk
    - Requests get queue up much much faster…
    - … disaster…

Stability

- ***Accept that failure will happen!*** (‘Design for failure’)

- *Safe failure mode:* A failure mode that contains the damage and protects the rest of the system.

- *Stability (resilience, longevity):* Ability to keep processing for a long time even when there are transient impulses, persistent stresses, or component failures

- *Analogy: Crumble zones in modern cars…*

AARHUS UNIVERSITET

Compare to Bass, 3rd Ed

- *Availability (1):* Property of software that it is there and ready to carry out its task when you need it to be.
- Avizienis el al.: *Dependability* is the ability to avoid failures that are more frequent and more severe than is acceptable

Stability (resilience, longevity): Ability to keep processing for a long time even when there are transient impulses, persistent stresses, or component failures

- *Availability (2):* Ability of a system to mask or repair faults such that the cumulative service outage period does not exceed a required value over a specified time interval

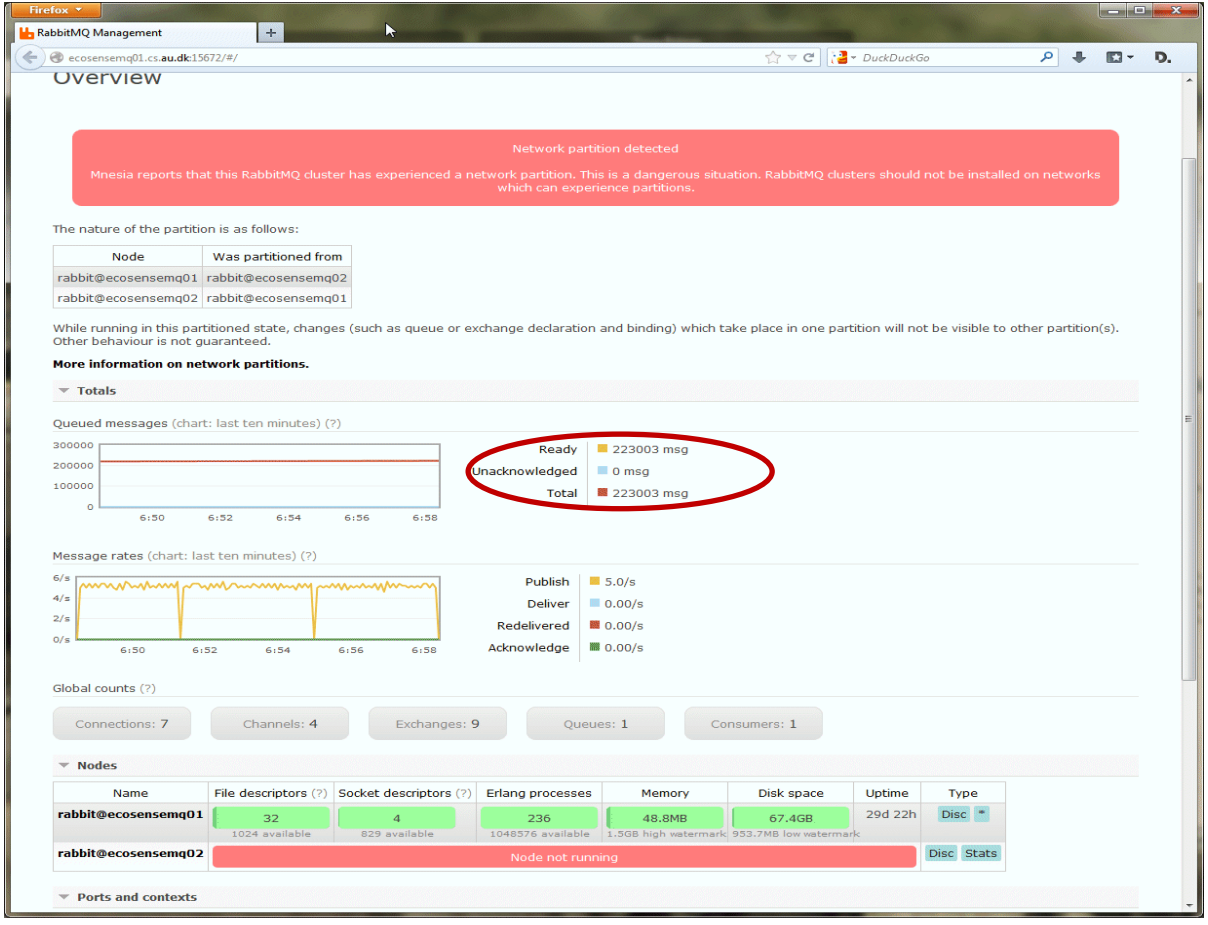
AARHUS UNIVERSITET

Discussion

- *Student report: “As my program is running on Amazon EC2, it is always available”*
  - What is the ‘system’ definition here?

- Ragnarok (my phd project) crashed due to an unhandled ”disk full” error and left the persistent data structure in an inconsistent state
  - Impulse, strain, crack, ?
  - Potential *safe failure mode*?

Discussion


Firefox
RabbitMQ Management
ecosensemq01.cs.au.dk:15672/#/
DuckDuckGo
Overview
Network partition detected
Mnesia reports that this RabbitMQ cluster has experienced a network partition. This is a dangerous situation. RabbitMQ clusters should not be installed on networks which can experience partitions.
The nature of the partition is as follows:
Node
Was partitioned from
rabbit@ecosensemq01
rabbit@ecosensemq02
rabbit@ecosensemq02
rabbit@ecosensemq01
While running in this partitioned state, changes (such as queue or exchange declaration and binding) which take place in one partition will not be visible to other partition(s). Other behaviour is not guaranteed.
More information on network partitions.
Totals
Queued messages (chart: last ten minutes) (?)
300000
200000
100000
0
6:50
6:52
6:54
6:56
6:58
Ready 223003 msg
Unacknowledged 0 msg
Total 223003 msg
Message rates (chart: last ten minutes) (?)
6/s
4/s
2/s
0/s
Publish 5.0/s
Deliver 0.00/s
Redelivered 0.00/s
Acknowledge 0.00/s
Global counts (?)
Connections: 7
Channels: 4
Exchanges: 9
Queues: 1
Consumers: 1
Nodes
Name
File descriptors (?)
Socket descriptors (?)
Erlang processes
Memory
Disk space
Uptime
Type
rabbit@ecosensemq01
32
1024 available
4
829 available
236
1048576 available
48.8MB
1.5GB high watermark
67.4GB
953.7MB low watermark
29d 22h
Disc
rabbit@ecosensemq02
Node not running
Disc
Stats
Ports and contexts

AARHUS UNIVERSITET

Discussion

- Oct 20 2013
  - Grundfos dorm server did not send data for 26 hours
    - Normally 70KB/s
  - On Oct 21 it went online again, pumping all accumulated data into Karibu during 51 minutes (70Kb/s x 26 hours = lot!)
    - Impulse, strain, crack, ?

AARHUS UNIVERSITET

Chain of failure

Cracks propagate!

- At each step
  - It can be accelerated
  - It can be slowed
  - It can be stopped
- And…
  - High level complexity → more directions for cracks to propagate
  - Tight coupling → accelerate cracks

AARHUS UNIVERSITET

Crack analysis

- Process: Look at every
  - External/remote call
  - Every I/O
  - Every use of resources
  - Every expected output
  - Every mutex/synchronized/critical region
- … and ask

What are *all* the ways this can go wrong?

Analyze types of impulses and stresses

Examples

- What if I cannot make the initial connection?
- What if it takes 10 minutes?
- What if I get disconnected?
- What if it takes 2 min to respond?
- What if 100.000 queries arrive at any one time?
- What if my disk is full when I try to log the error message from the SQLException that happened because the network during the update transaction while…?

Stability Antipatterns

14 root causes that create, accelerate or multiply cracks and lead to system failures

AARHUS UNIVERSITET

Integration Points

- *Integration Points:* Points in the software that integrate with external systems [My shot at a definition!]
  - Sockets, process, pipe, RPC, file access, database call, feed, …
  - Even calls to *critical regions* ala *synchronized methods*

- Lessons to remember
  - Every integration point will *eventually fail* in some way
  - Failures take several forms. You will not get nice error messages
  - Failures propagate quickly
  - *Know when to open up abstractions*
    - Cracks does not respect your nice role boundaries ☺

AARHUS UNIVERSITET

Ex: RabbitMQ Java library

- Thanks for nothing…

```
public class ShutdownSignalException
extends java.lang.RuntimeException
implements SensibleClone<ShutdownSignalException>
```

Encapsulates a shutdown condition for a connection to an AMQP broker. Depending on HardError when calling `getReference()` we will either get a reference to the Connection or Channel instance that fired this exception.

- My perception is that failure conditions are often poorly documented. I need to understand the root cause…

- Finding the underlying problem, require *to open up the abstraction*
  - Ahh – plowing through MongoDB connector source code ☹

AARHUS UNIVERSITET

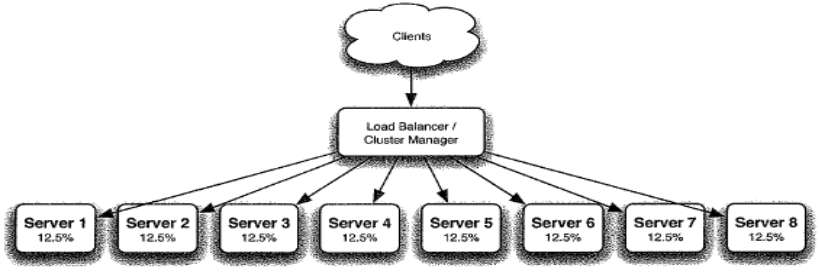
Integration Points

- In my mind ‘Integration Points’ is not really an anti-pattern

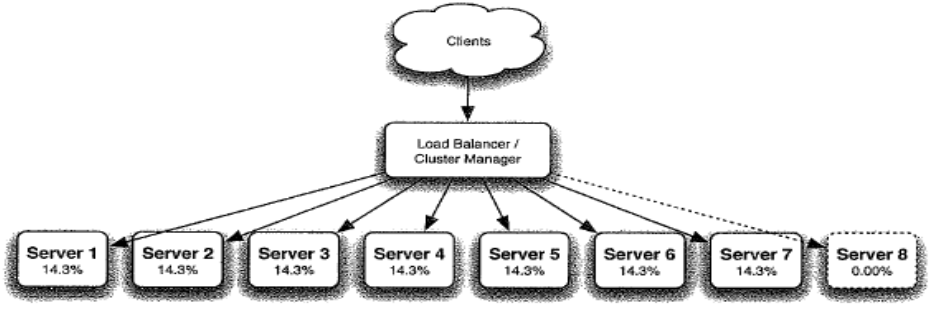
- It is more the central concept/term we will apply for all those ‘code segments’ that pose a stability issue, and can lead to all the other anti-patterns.

Chain Reactions

- Scaling
  - Horizontal: More machines (Google way)
  - Vertical: Bigger machine (Oracle way)
- Load-balancing over horizontal farm



- One server down: Load distributed



Chain Reactions

- If one server failed due to *load-related condition*, the rest of the pool are now more likely to fail!

- *Chain Reaction:* One failure (often resource/load related) in one server leads to increased probability of failures (often of similar type) in other servers [My shot at a definition!]

Ex: WoW Realms

- World of Warcraft
  - Realms safeguard against Chain Reactions
  - Real hard way of load-balancing…

AARHUS UNIVERSITET

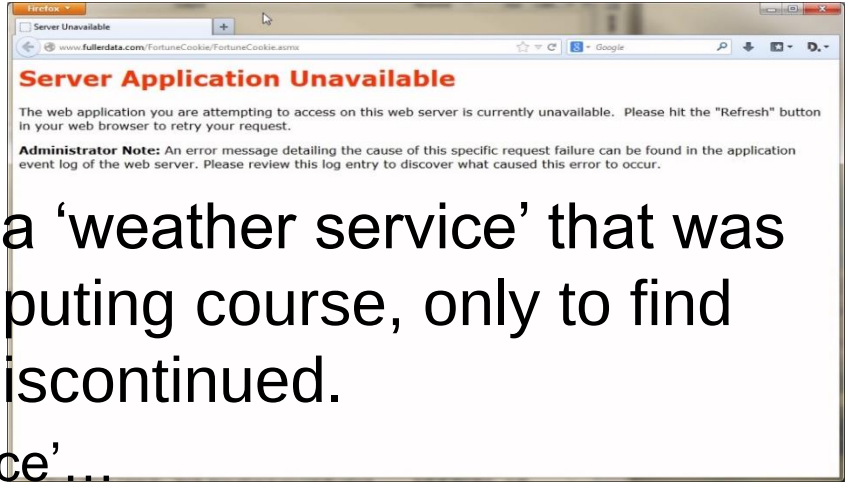
Cascading Failures

- *Cascading failure*: A failure that ”jumps the gap” from one service or layer to another [My shot at a definition!]
  - Example: Database failure, cascades into app server.

AARHUS UNIVERSITET

Example

- In a 2014 course I made a Facade service, delegating to a fortune cookie service I found on the internet
  - It, however, failed right after the seminar in which I introduced it ☹



- In MSDO, I wanted to reuse a ‘weather service’ that was introduced in my Cloud Computing course, only to find the underlying service was discontinued.
  - Rewrote it into the ‘quote service’…

Users

- Human users have a knack for creative destruction…

- Users means memory is consumed
- Users do unpredictable things to your apps
- Users may be malicious
- Users will gang up on you
  - ”Your site will get *slashdotted”*

  - *Slashdotted = posting a web URL on slashdot will cause a large surge in web traffic, which can overwhelm small sites…*

Users

- And then there are the ‘Unwanted users’
  - Bots that screen scrape your site to lure your prices
  - *Competitive intelligence*

- Nygard mention some tricks to block them of
  - Robots.txt only work with the nice ones
  - Firewalling them off
  - Sue them

AARHUS UNIVERSITET

Blocked Threads

- In the good old Amiga days, you knew a crash ☺

- Modern JVMs seldom crash the machine but just hangs in deadlocks, doing nothing useful.

- *Blocked threads:* All processes are blocked waiting on some impossible outcome

Example

- System uses ‘ObjectCache’ with ‘get()’
  - Interface says nothing about ‘synchronization’, but the implementation does…

```
public synchronized Object get(String id) {
  Object obj = items.get(id);
  if(obj == null) {
    obj = create(id);
    items.put(id, obj);
  }
  return obj;
}
```

Is slow, so they make a ‘Caching Proxy’:
If (incache) return cache;
else super.get()

- ‘create’ does external call to inventory system
- … which crash due to ‘unbalanced capacities’
- **Now, first thread to do cache miss calls ‘create’ which will wait for response that never comes (and never release lock on this)**
- **And all other will block as well, even if item in the cache…**

AARHUS UNIVERSITET

Self-Denial Attacks

- *Self-Denial Attacks:* Impulses that are generated from your own organization

- Ex
  - Winter storm: Lystrup skole webpage
    - *We do not know if school will be operating. Check back at 07.00 o'clock*
  - The Kähler vase incident
    - Put a specific vase on sale on a specific time
      - Which caused the web site to crash

Diablo III Launch May 15, 2012

- As well as ‘Pokemon No’ 2016

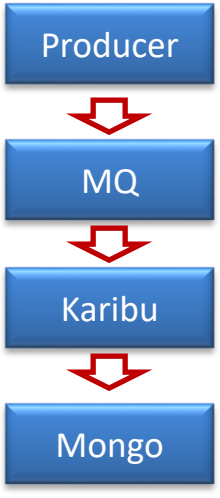
Scaling Effects

- *Scaling Effects:* Failures occur when increasing n, as scaling often goes as O(n^2) or worse
  - *Point-to-point communication* between servers scale as O(n^2)
    - Development environment: *one server* *1 connection*
    - Staging environment: *two servers* *4 connections*
    - Production environment: *16 servers* *256 connections !*
  - *Shared resources* are resources that *all* services must access to get work done (akin ‘singleton pattern’)
    - *Saturation → connection backlog*
    - *Backlog exceeds listen queue → failures*

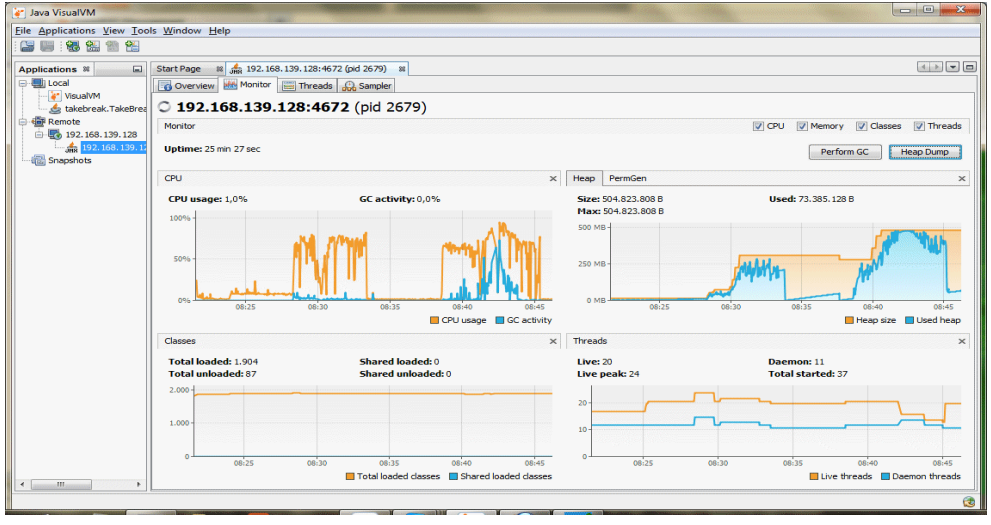
AARHUS UNIVERSITET

Unbalanced Capacities

- Example: *The Lyon airport stress test* I did in… Lyon airp.
  - Throttling up the producer rate of sending messages
    - *Strain* ☺



Karibu:
Pull message, convert, store in Mongo, repeat



- My daemon code was single-threaded, so *WTF???*
  - ”fetch msg from MQ, convert, store in MongoDB”

AARHUS UNIVERSITET

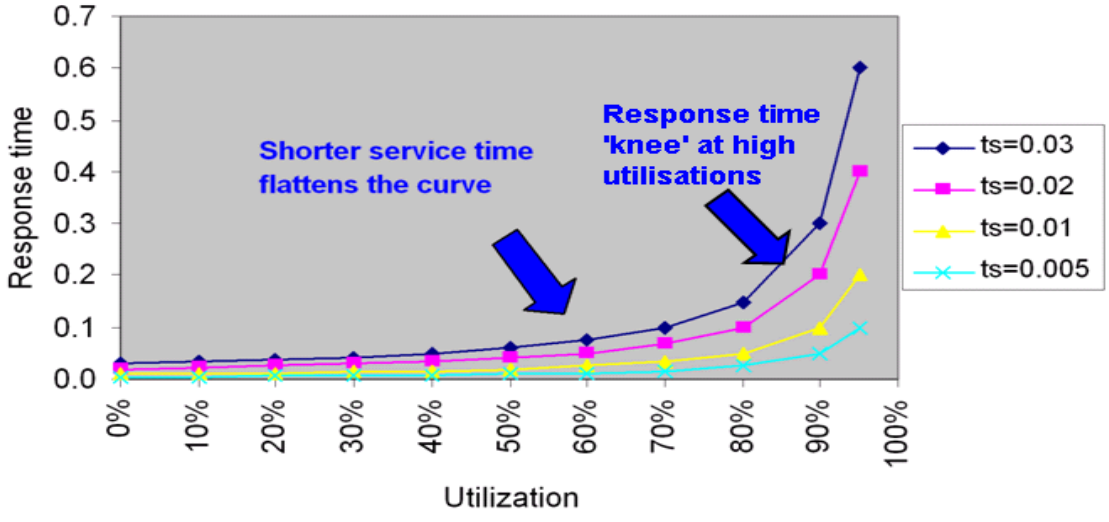
Unbalanced Capacities

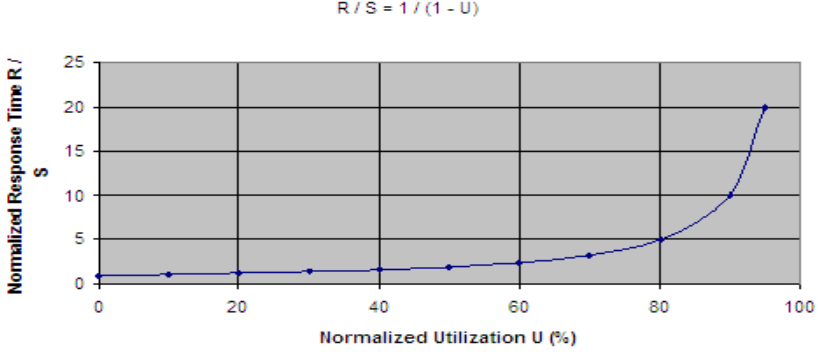
- It turned out that
  - RabbitMQ can deliver message at vastly higher speeds than MongoDB can ever store them (not surprising, but…)
  - The RabbitMQ Java connector/driver uses *prefetch*
    - Fetch next message even though the last has not been acknowledged by my daemon code yet!
    - (And where was that described in the documentation???)
    - Limit on prefetch: **default is ∞** ***thanks!***
- *Result: Chain reaction due to GC overload*
  - *Stemming from unbalanced capacities (mq vrs db)*

AARHUS UNIVERSITET

All systems have a limit

- All systems have a limit after which performance degrades rapidly



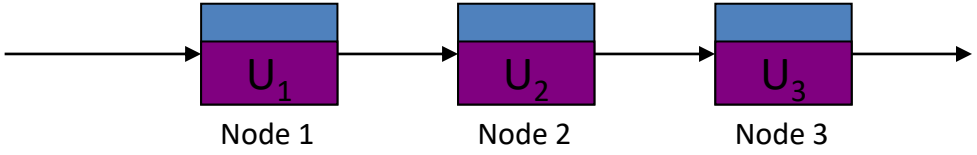


Around 70% utilization the response time exceeds 3 times the service time !

Balanced System and Bottleneck Analysis

- A **queuing system** is balanced if all nodes have the same equal utilization

$$U_1 = U_2 = U_3$$



- A software system can achieve its best possible, scalable performance if it's a balanced system
  - Therefore you should generally consider the node with highest utilization when you need to optimize a system (this is typically where the bottleneck is)
  - And generally stop when the system is balanced

AARHUS UNIVERSITET

Unbalanced Capacities

- *Unbalanced Capacity:* The mismatch between capacity of system parts that participate in handling transactions [own definition]

- Note
  - Depends on particular usage profile
    - Particularly subject to ‘Attacks of self denial’
      - That is, suddenly the usage profile change dramatically
        - » The Kähler vase case; Black Friday; Christmas sales…
  - Difficult to test as ratios often differ in QA and prod.
    - QA: often front-end is 1-1 but in prod. 10-1

Dogpile

- *Dogpile:* A large set of *starting/initializing* servers generate an impulse that generates a cascading failure [own definition]
- Or
  - Startup/periodic load may be much higher than steady-state load, triggering *circuit-breaker* tripping…
- Can also occur when external event trigger synchronization of traffic
  - Like pedestrian stoplights…

Dogpile

- Anyone from my SAiP fagpakke?
  - One important aspect of the load generator is the *gaussian delay* in the workload model
  - Why is that extremely important?
- How to avoid dogpiles?
  - Architectures must be crafted to acknowledge their existence and take preventive measures.

Dogpile

- Even swarm has a dogpile akin-of problem
  - ‘depends_on’ attribute in compose-file dictate the starting order of services, but there is a difference between ‘running’ and ‘ready-for-work’ state…
- We will return to dockerfile HEALTHCHECK later
  - Can tell if a container is ‘ready-for-work’ rather than just running

Force Multiplier

- *Force Multiplier:* Control plane automation can overreact (multiply the force) upon detected failures, due to incorrect assumptions or missing/incorrect control data, leading to cascading failures [Own definition]
  - *Example:*
    - Reddit Goal: update zookeeper cluster
      - Shut down autoscaler, then update zookeeper cluster
    - Actual process
      - Package mgt system detects autoscaler missing, restarts it (!)
      - Autoscaler reads zookeeper (partially migrated) data
        - » Determine much less need for app and cache servers
        - » *So, starts shutting down a lot of servers*
  - The package management 'multiplied the force' on system…

AARHUS UNIVERSITET

Even Swarm

- I had one incident, a crashing version of SkyCave daemon sneaked by my pipeline test
- Upon redeploying the stack…
  - (I was experimenting with rollback, but obviously got it wrong)
  - … swarm just restarted services, saw them crash, and restarted
  - Eventually my machine would ‘Unsteady State’ crash

```
crunch@crunch3:~/proj/crunch4$ docker stack ps crunch-webui-stack
ID                  NAME                                    IMAGE                                NODE                DESIRED STATE       CURRENT STATE            ERROR
PORTS
o5f6ml9duyz6        crunch-webui-stack_crunchui.1           henrikbaerbak/private:crunchwebui    crunch3             Running             Running 2 minutes ago
kn9afbjdkezj        crunch-webui-stack_submissiondb.1       mongo:3.4                            crunch3             Running             Running 2 minutes ago
362ko934vmnt        crunch-webui-stack_crunchui.1           henrikbaerbak/private:crunchwebui    crunch3             Shutdown            Shutdown 18 hours ago
4anijf657vth         \_ crunch-webui-stack_crunchui.1       henrikbaerbak/private:crunchwebui    crunch3             Shutdown            Shutdown 25 hours ago
s75rilf58r74         \_ crunch-webui-stack_crunchui.1       henrikbaerbak/private:crunchwebui    crunch3             Shutdown            Shutdown 2 days ago
80czxcw9ntx7         \_ crunch-webui-stack_crunchui.1       henrikbaerbak/private:crunchwebui    crunch3             Shutdown            Shutdown 3 days ago
9jjsoxiddu9j        crunch-webui-stack_submissiondb.1       mongo:3.4                            crunch3             Shutdown            Complete 2 minutes ago
```

Safeguards

- Generally, the hardware/robotics guys are better at it…
  - Industrial robots have multiple layers of safeguards to prevent damage to people and facilities…
- So encode *safeguards* in software

The Governor Pattern

  - If 80% of system is suddenly gone, it is probably *you* that observes a distorted version of the world
  - If gap between observed and desired state is large, ask for human intervention/confirmation
  - Hysteresis:
    - Start VMs quickly, stop them slowly.
    - VMs takes time to start handling load, so do not start another one 10ms after – even if you see load not decreasing right now…

AARHUS UNIVERSITET

Slow Responses

- *Slow Response*: One system part starts to respond slowly
  - Easily triggers a ‘cascading failure’, as depending systems of course also starts responding slowly
  - Example:
    - Single-threaded SkyCave (‘socket.cpf’)
    - If user 1 invokes ‘quote’ and it takes 10 minutes to respond…
    - Then what will user 2, 3, 4, … experience?

AARHUS UNIVERSITET

SLA Inversion (*)

- *SLA Inversion:* The best SLA you can offer is the worst SLA of any external software system you depend upon.
  - If your system depends upon service X that offers only 88% availability, how can you promise 99,999%?
  - Example: The cookie service I relied upon had very low availability…
- (*) Removed from his second edition…

AARHUS UNIVERSITET

Unbounded Result Sets

- *Unbounded Result Sets:* Queries may return many more elements than expected

  - Query, loop-over-elements
  - Works well with 230 elements, but 5.000.000 ?
  - And the result?
    - Ressource strain, slow response, cascading failure, …
  - Always query with a limit !

AARHUS UNIVERSITET

A Similar AntiPattern

- *Not mentioned by Nygard directly, but*
- *Steady state:* For every mechanism that accumulate resources, some other mechanism must recycle that resource.
- Then we must also have AntiPattern
- *Unsteady state:* One or more mechanisms exist, that does not recycle resources but keeps accumulating them

Summary (*)

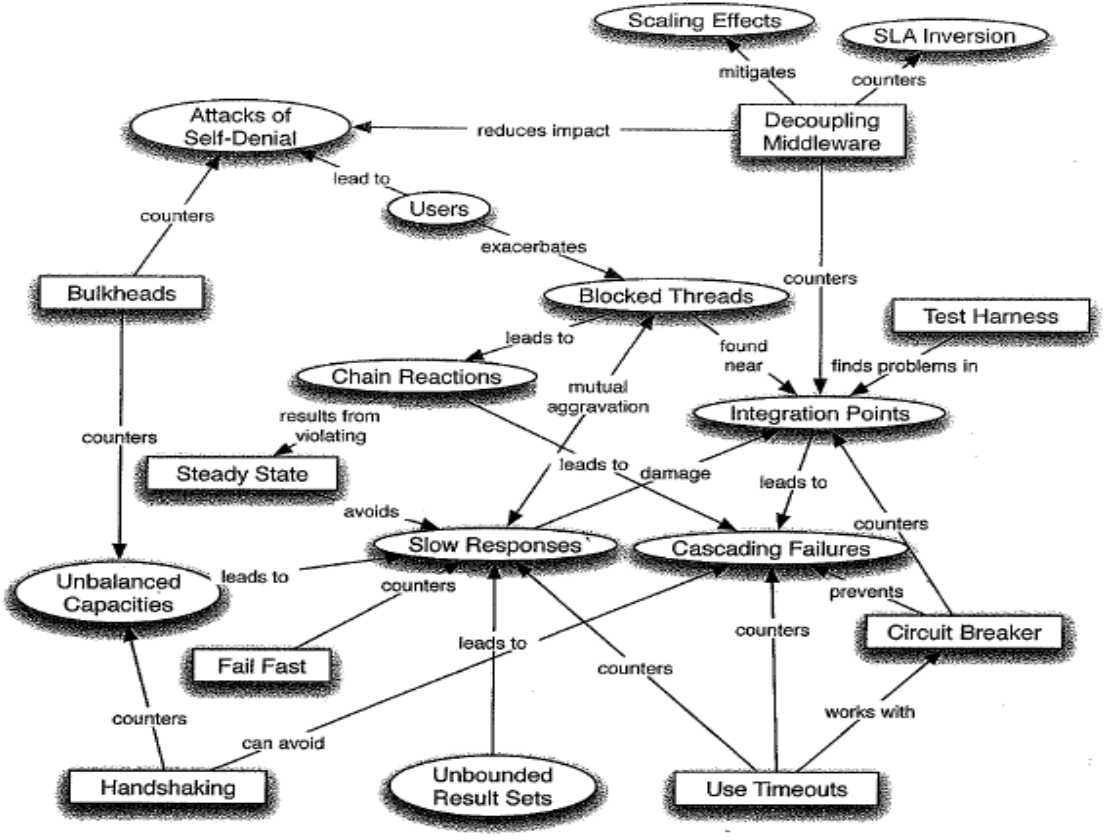


Figure 3.1: Interaction of Patterns and Antipatterns

First Ed.