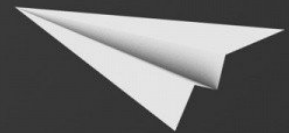


My code is slow, and why you should care

Musings of a software developer



Jason Chalom
contact@jasonchalom.com

Speed

Stability

Reliability



WHY?

- Trust in the software
- Growth

Focus:

What is your reason?

Find the objective and scope it

A futuristic digital dashboard with various gauges and displays. On the left, a vertical speedometer with yellow numbers 60, 70, 80, and 100. At the top center, a large digital display shows '146,667' in yellow. To its right are icons for a battery and a flame. Below the main display, a red curved line and a blue bar chart are visible, with the number '147' in red. At the bottom, there are two circular gauges labeled 'FUEL' and 'ELEC' with red needles. The 'FUEL' gauge has markings for 1/4, 1/2, and 3/4. The 'ELEC' gauge has markings for 20 and 40. The background is dark with blue and yellow highlights.

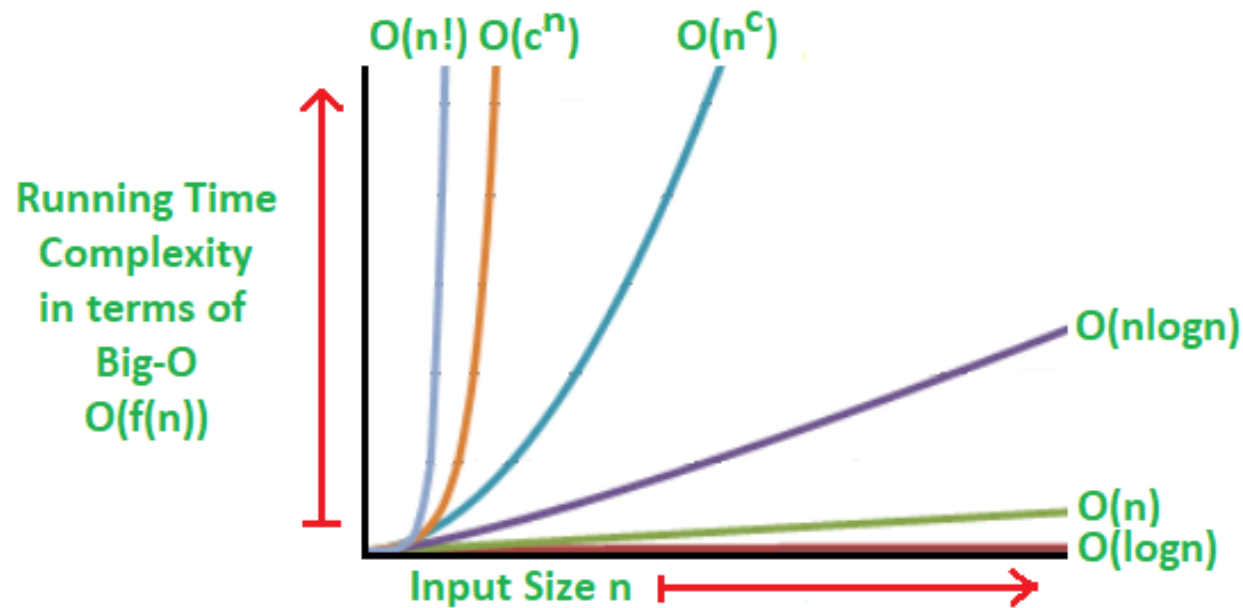
Measure and

measure some more

Types of Performance

- Computational
 - Big O, Omega, Phi
- Data loading speeds
- Architecture and system modelling
- Dependencies and third party reliance
- Precision

0. Computational



$O(n!), O(c^n), O(n^c)$ - Worst

$O(n \log n)$ - Bad

$O(n)$ - Fair

$O(\log n)$ - Good

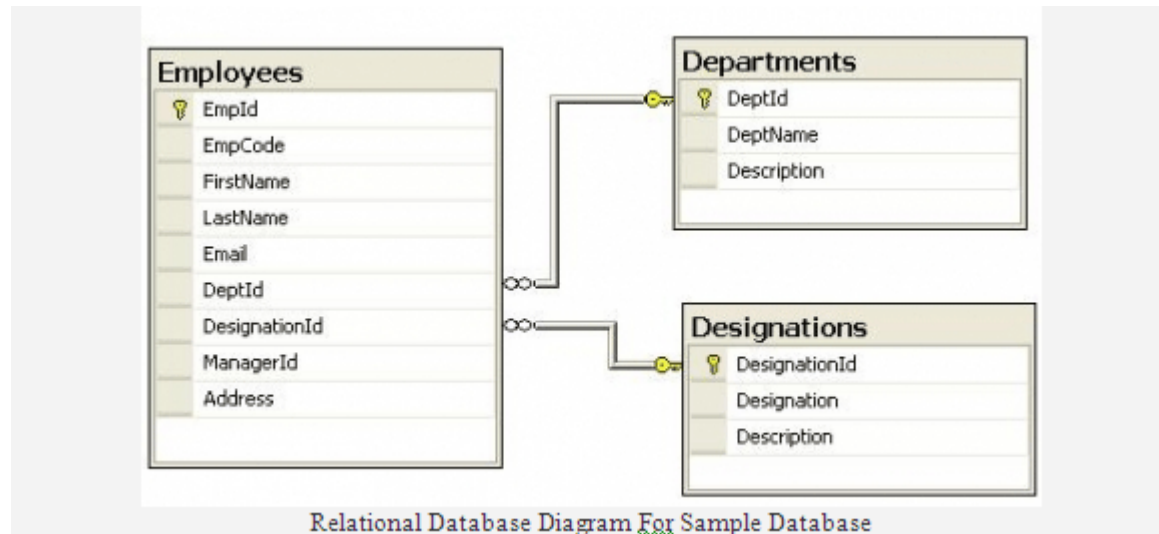
$O(1)$ - Best

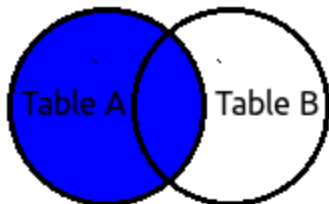
1. Data

The usual suspect

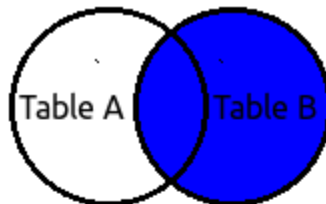


Understand your data architecture

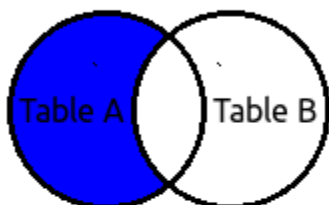




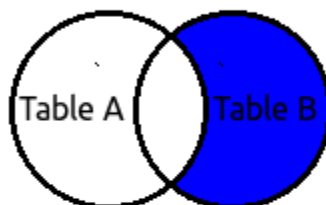
SELECT [list] FROM
[Table A] A
LEFT JOIN
[Table B] B
ON A.Value = B.Value



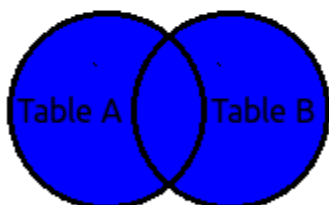
SELECT [list] FROM
[Table A] A
RIGHT JOIN
[Table B] B
ON A.Value = B.Value



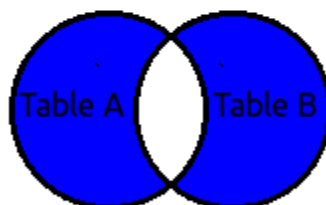
SELECT [list] FROM
[Table A] A
LEFT JOIN
[Table B] B
ON A.Value = B.Value
WHERE B.Value IS NULL



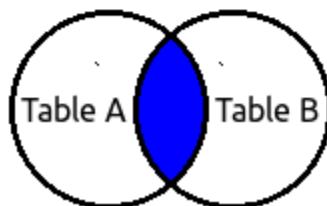
SELECT [list] FROM
[Table A] A
RIGHT JOIN
[Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL



SELECT [list] FROM
[Table A] A
FULL OUTER JOIN
[Table B] B
ON A.Value = B.Value



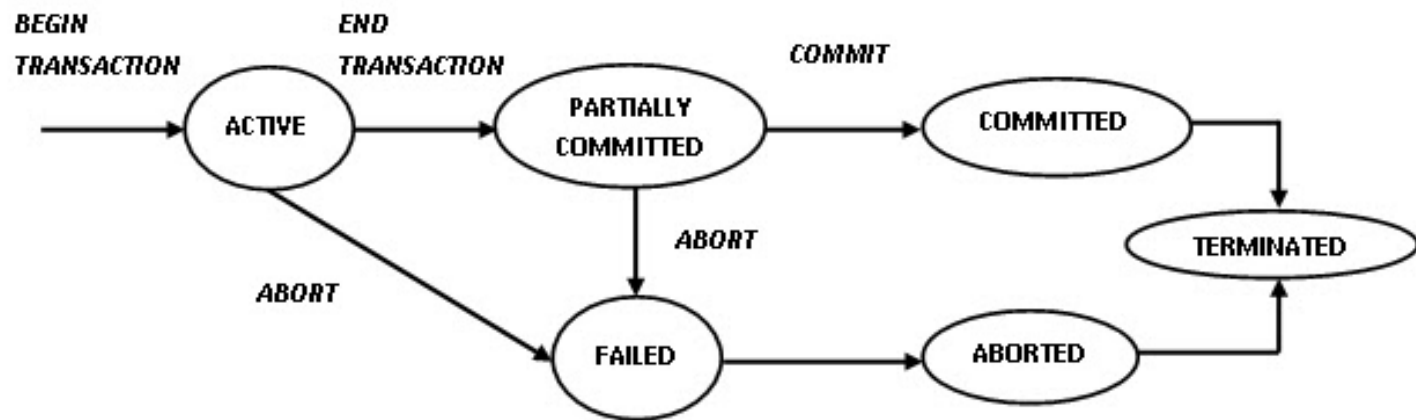
SELECT [list] FROM
[Table A] A
FULL OUTER JOIN
[Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL
OR B.Value IS NULL



SELECT [list] FROM
[Table A] A
INNER JOIN
[Table B] B
ON A.Value = B.Value

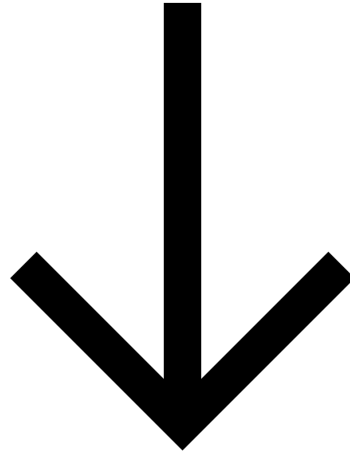
Transactional databases

and batch processing



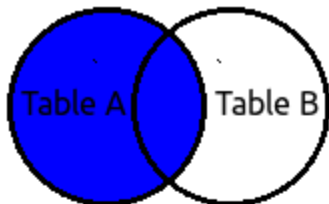
```
1 batch = 15000 items
2
3 for element in batch
4     do_some_work()
5
6 completed_at_date = Date.now
7 ▾ for element in batch
8     update_batch_date_in_database(completed_at_date)
```

```
1 def update_batch_date_in_database(date, element)  
2     element.update!(batch_date: date)  
3 end
```

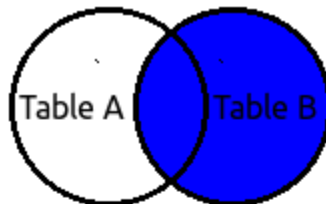


```
• 1 elements.update_all(batch_date: date)
```

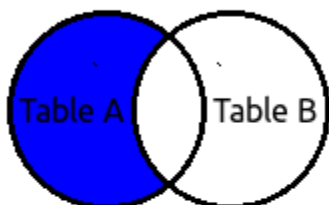
Early reduction



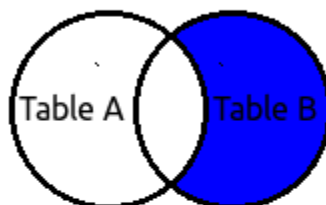
SELECT [list] FROM
[Table A] A
LEFT JOIN
[Table B] B
ON A.Value = B.Value



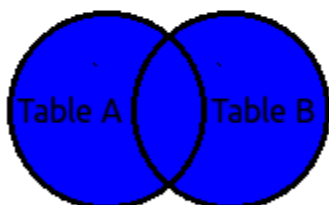
SELECT [list] FROM
[Table A] A
RIGHT JOIN
[Table B] B
ON A.Value = B.Value



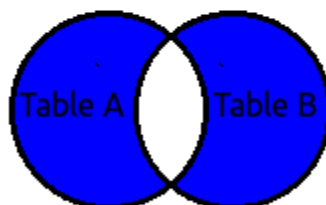
SELECT [list] FROM
[Table A] A
LEFT JOIN
[Table B] B
ON A.Value = B.Value
WHERE B.Value IS NULL



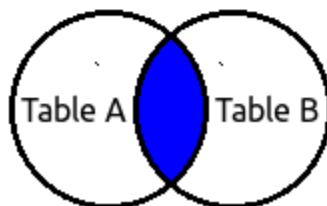
SELECT [list] FROM
[Table A] A
RIGHT JOIN
[Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL



SELECT [list] FROM
[Table A] A
FULL OUTER JOIN
[Table B] B
ON A.Value = B.Value



SELECT [list] FROM
[Table A] A
FULL OUTER JOIN
[Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL
OR B.Value IS NULL



SELECT [list] FROM
[Table A] A
INNER JOIN
[Table B] B
ON A.Value = B.Value

A high-angle, long-exposure photograph of a busy train station. Two trains are visible on adjacent tracks. The train on the left is a grey and red locomotive with the number 9529. The train on the right is a yellow and grey locomotive with the number 2057 and the destination 'Churchgate' displayed. A massive crowd of people is walking between the tracks, their figures blurred due to motion. The station is covered by a large roof structure with hanging lights. In the foreground, there are two black signs with white numbers '2' and '3'.

**Rely on your
framework**

A photograph of a snowy landscape with railroad tracks curving into the distance. The tracks are covered in snow and footprints. A railroad crossing sign is visible on the right side of the tracks. The sky is clear and blue.

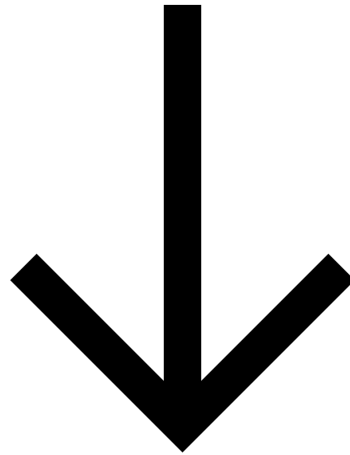
Dont rely on your
framework

Offloading the computation

One solution now is
not a solution later
on

The magic ``.pluck``

```
1 Users -> UserDetails
2 Users have company
3 UserDetails have name
4
5 User.join(:user_details).where(company: 'Next45', name: 'Bob')
```



```
1 user_ids = User.where(company: 'Next45').pluck(:user_detail_id)
2 results = UserDetail.where(id: user_ids).where(name: 'Bob')
```


2. Architecture



Why solution x ?

NoSQL

A photograph of a long, straight asphalt road stretching into the distance. The road has a dashed yellow center line and white dashed lines on the left side. The landscape is flat and arid, with sparse, dry vegetation and small trees on the sides. The sky is overcast and grey. The text "Flat structures" is overlaid in the center of the image.

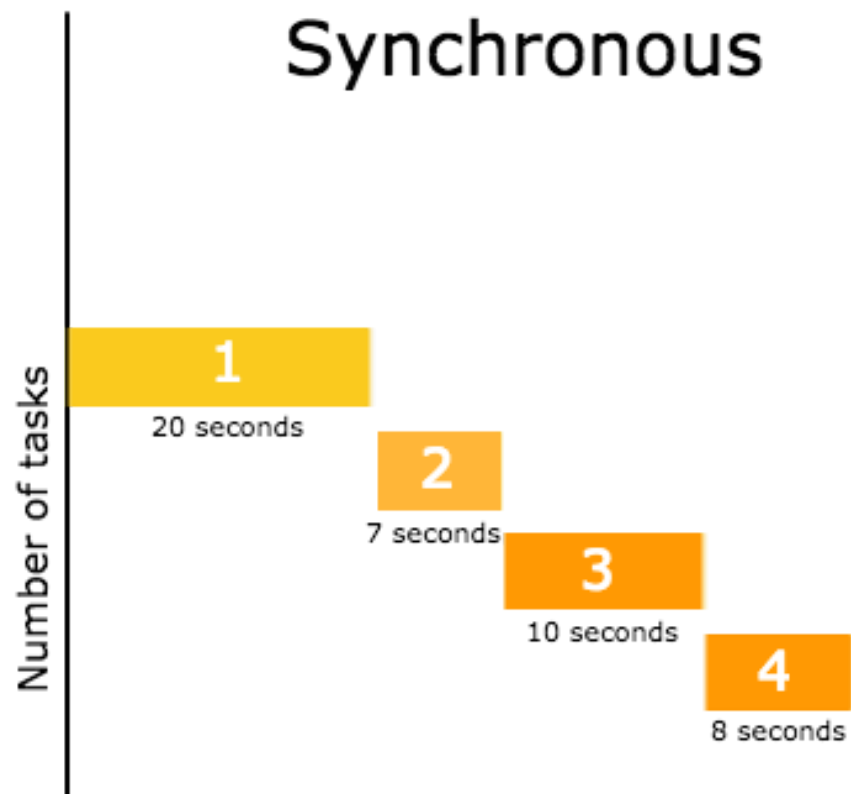
Flat structures



ElasticSearch

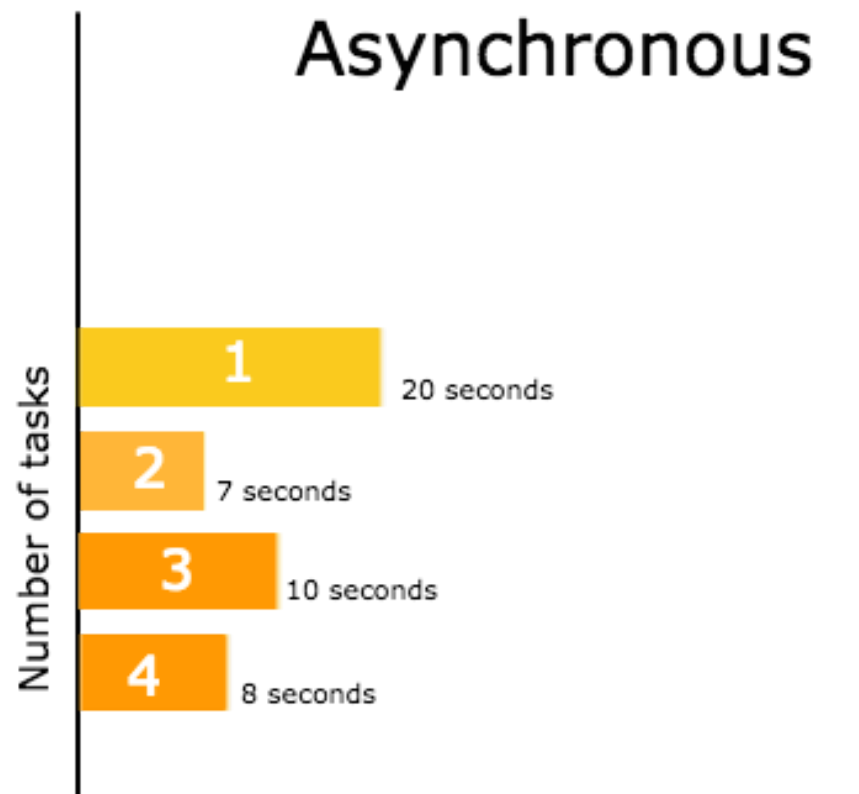
**Async rather than
sync**

Synchronous



Total time taken by the tasks.
45 seconds

Asynchronous



Total time taken by the tasks.
20 seconds

3. Dependencies

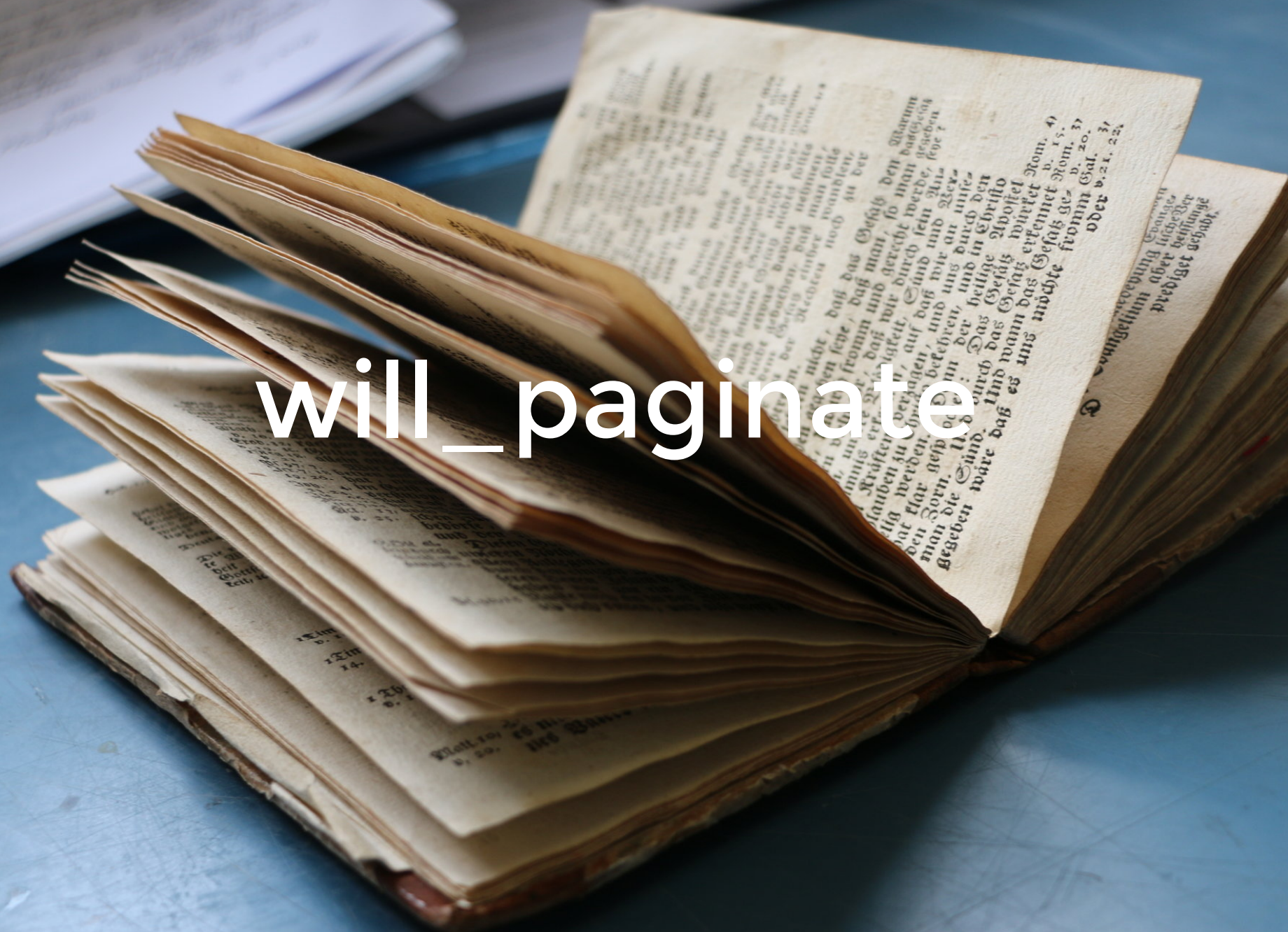
Pros

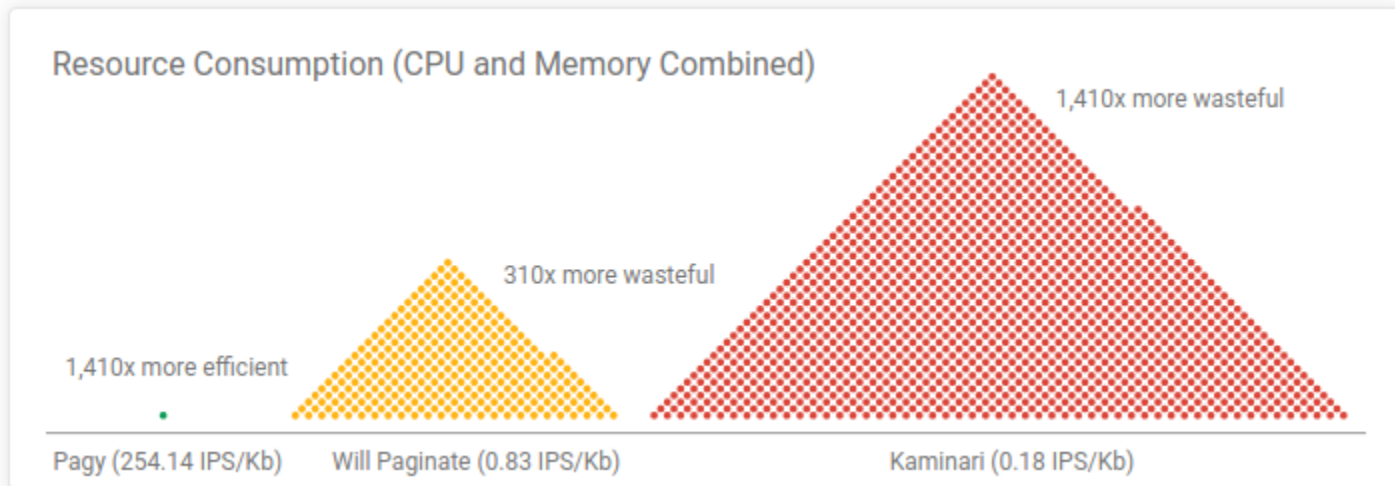
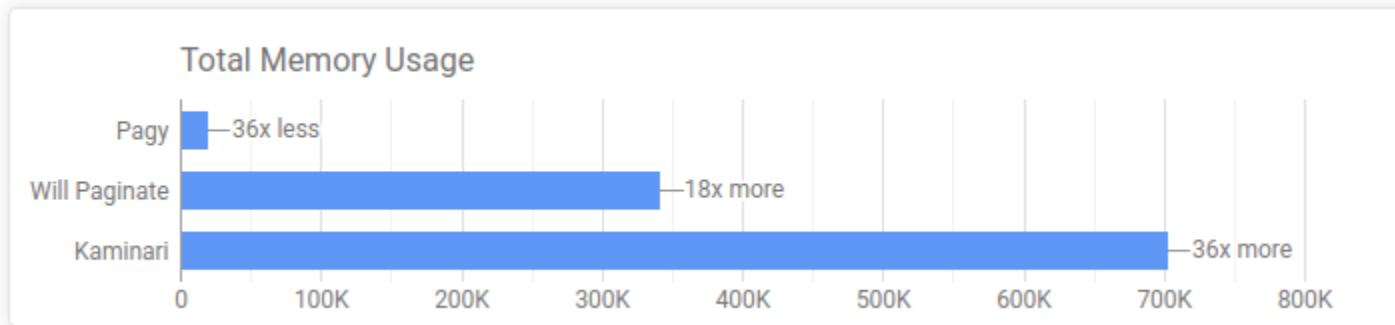
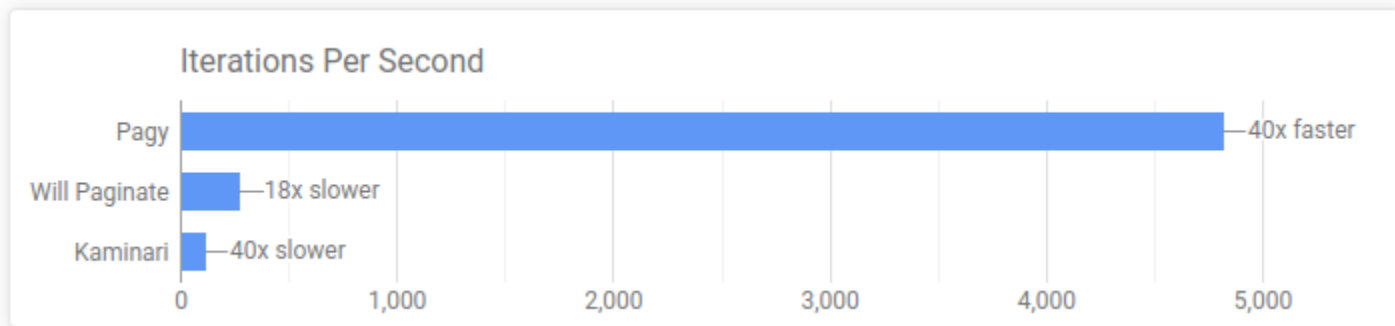
- Makes development faster
- Someone else has solved the problem
- Abstraction

Cons

- Have to maintain
- May have own internal bugs
- May no longer be maintained

will_paginate





*claimed by pagy

Metrics that were in place failed here

- Experimental results proved value of the change

Stability

Headroom

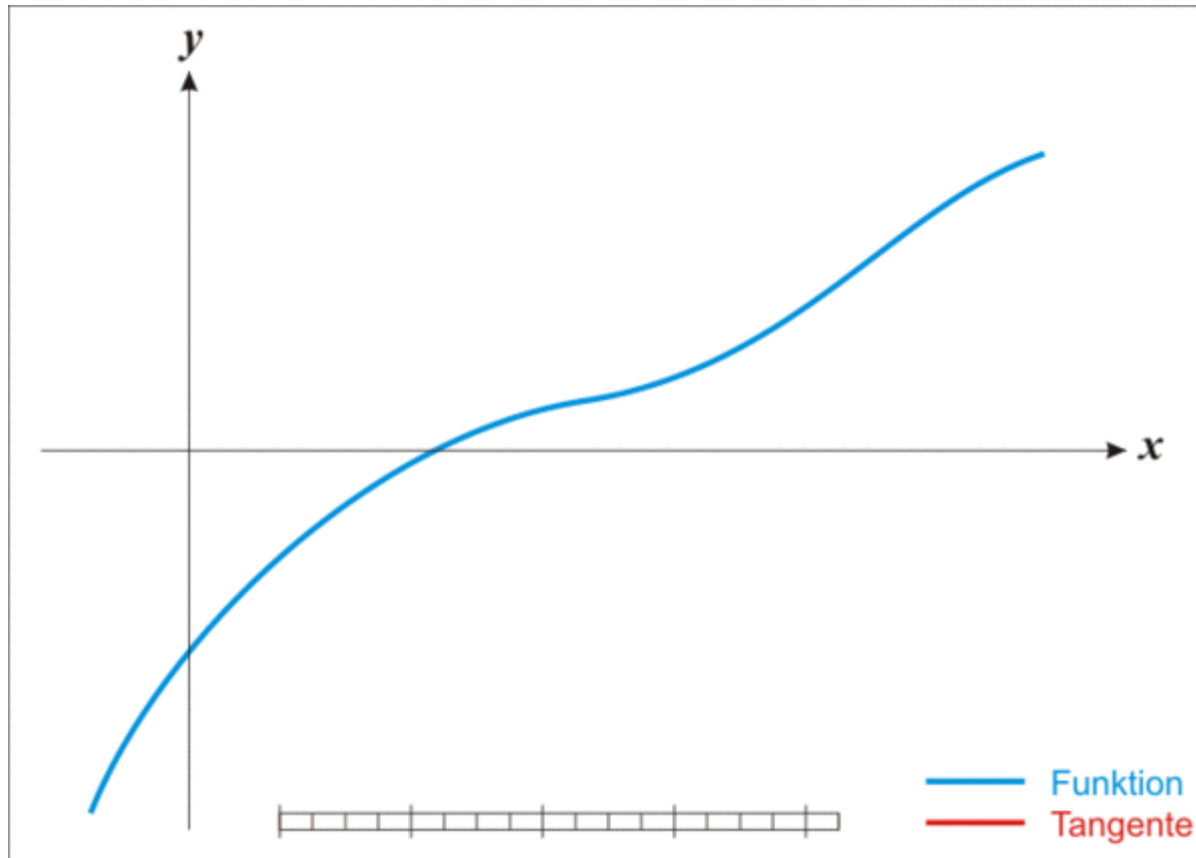
Less Bloat

Lessons:

- Sometimes do it yourself
- Experiment on hunches
- Always re-evaluate your strategies when they appear to not be working

4. Precision

Error of computation



Takeaways



Hmm Pizza ...

- Performance issues are healthy
- Solutions => time and cost
- Scaling does not always work
- Follow convention more than not
- Always trade-offs
- What works today may not tomorrow

Thank you

