How Frappe V14 is bLaZinGLy fast*

* without Rewriting it in Rust

Ankush Menat
Engineer

Frappe

Improvements

- Lower response times
- Lower memory usage
- Lower asset bundle size
- Lower disk usage
- Faster "feels"

How we identify bottlenecks

- Bug reports from paying customers or community
- Frappe Monitor logs using ELK stack
- Sentry
- Developers scratching their itch*

Common patterns

	Knowns	**Unknowns**
Known	Specific known bottlenecks	Overheads from libraries and abstractions boundaries
Unknown	Our own "small" unknown overheads	*Good luck!*

Common patterns

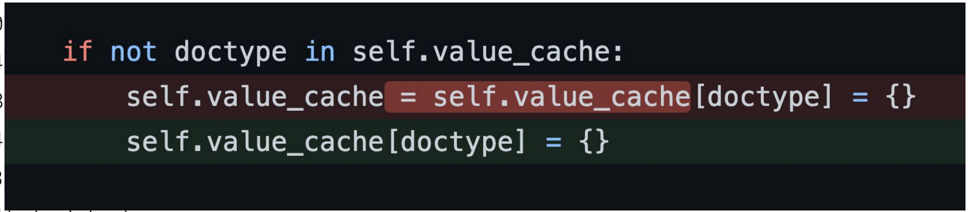
	Knowns	**Unknowns**
Known	Specific known bottlenecks	Overheads from libraries and abstractions boundaries
Unknown	Our own "small" unknown overheads	*Good luck!*

“Stock Entry submission is slow”

1. Find entry point: **`stock_entry.submit()`**
2. **`%prun sales_invoice.submit()`**
3. …
4. profit?

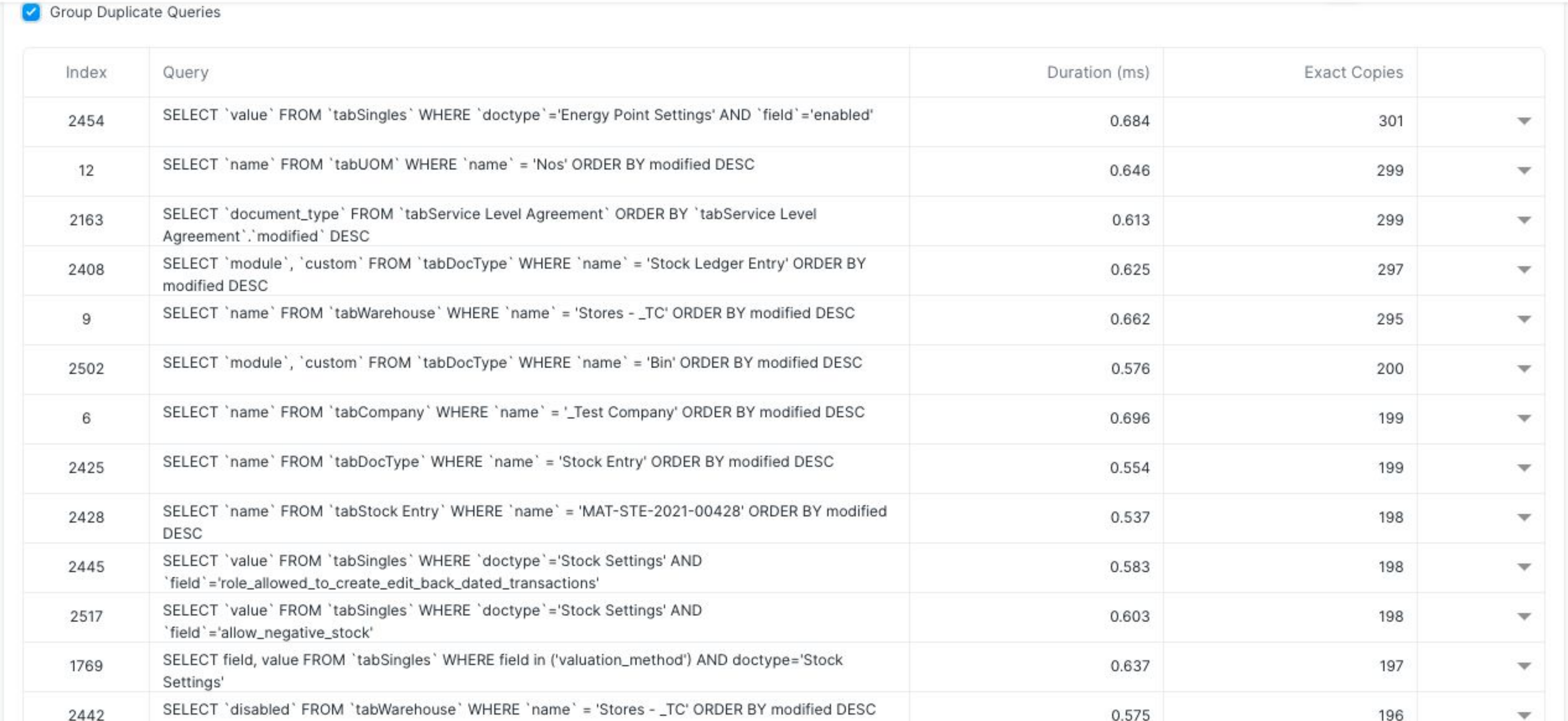
“Stock Entry submission is slow”

```
      17314183 function calls (17062515 primitive calls) in 18.350 seconds

 Ordered by: cumulative time

 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
  181/1    0.001    0.000   18.350   18.350 {built-in method builtins.exec}
  100/1    0.000    0.000   18.350   18.350 document.py:927(submit)
  100/1    0.000    0.000   18.350   18.350 document.py:915(_submit)
  199/1    0.000    0.000   18.350   18.350 document.py:280
  199/1    0.007    0.000   18.350   18.350 document.py:284
2792/15    0.031    0.000   16.988    1.133 document.py:848
2792/15    0.025    0.000   16.809    1.121 document.py:114
2792/15    0.017    0.000   16.791    1.119 document.py:113
  501/3    0.002    0.000   16.516    5.505 document.py:854(<lambda>)
  299/1    0.007    0.000   12.864   12.864 document.py:984(run_post_save_methods)
      1    0.000    0.000   12.519   12.519 stock_entry.py:94(on_submit)

    ---- clipped to keep relevant output ---

    299    0.002    0.000    1.791    0.006 document.py:815(_validate_links)
    398    0.042    0.000    1.760    0.004 base_document.py:522(get_invalid_links)
```



```
if not doctype in self.value_cache:
    self.value_cache = self.value_cache[doctype] = {}
    self.value_cache[doctype] = {}
```

“Stock Entry submission is slow”



✓ Group Duplicate Queries

Index	Query	Duration (ms)	Exact Copies
2454	SELECT `value` FROM `tabSingles` WHERE `doctype`='Energy Point Settings' AND `field`='enabled'	0.684	301
12	SELECT `name` FROM `tabUOM` WHERE `name` = 'Nos' ORDER BY modified DESC	0.646	299
2163	SELECT `document_type` FROM `tabService Level Agreement` ORDER BY `tabService Level Agreement`.`modified` DESC	0.613	299
2408	SELECT `module`, `custom` FROM `tabDocType` WHERE `name` = 'Stock Ledger Entry' ORDER BY modified DESC	0.625	297
9	SELECT `name` FROM `tabWarehouse` WHERE `name` = 'Stores - _TC' ORDER BY modified DESC	0.662	295
2502	SELECT `module`, `custom` FROM `tabDocType` WHERE `name` = 'Bin' ORDER BY modified DESC	0.576	200
6	SELECT `name` FROM `tabCompany` WHERE `name` = '_Test Company' ORDER BY modified DESC	0.696	199
2425	SELECT `name` FROM `tabDocType` WHERE `name` = 'Stock Entry' ORDER BY modified DESC	0.554	199
2428	SELECT `name` FROM `tabStock Entry` WHERE `name` = 'MAT-STE-2021-00428' ORDER BY modified DESC	0.537	198
2445	SELECT `value` FROM `tabSingles` WHERE `doctype`='Stock Settings' AND `field`='role_allowed_to_create_edit_back_dated_transactions'	0.583	198
2517	SELECT `value` FROM `tabSingles` WHERE `doctype`='Stock Settings' AND `field`='allow_negative_stock'	0.603	198
1769	SELECT field, value FROM `tabSingles` WHERE field in ('valuation_method') AND doctype='Stock Settings'	0.637	197
2442	SELECT `disabled` FROM `tabWarehouse` WHERE `name` = 'Stores - _TC' ORDER BY modified DESC	0.575	196

“Background job crashing from memory usage”

1. Find entry point:

   `repost_entries()`

2. **`@memory_profiler.profile`**

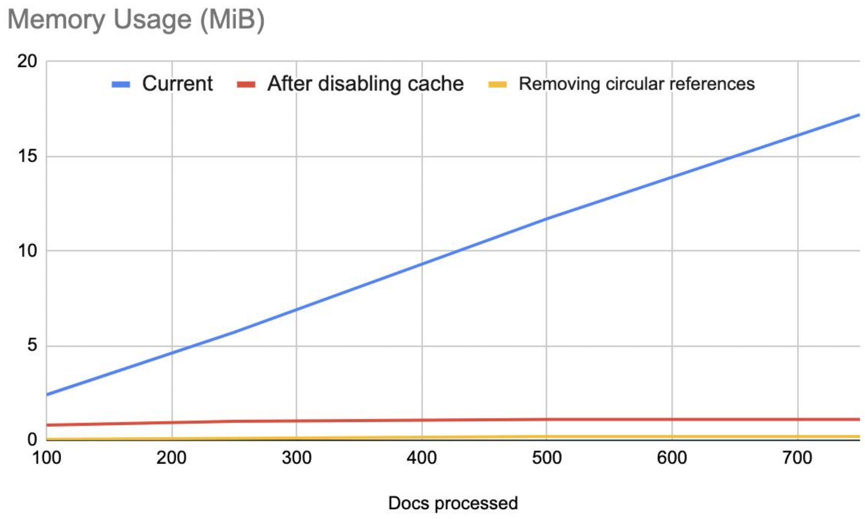
```
Line #    Mem usage    Increment  Occurrences   Line Contents
============================================================
     3   38.816 MiB   38.816 MiB           1   @profile
     4                                         def process_docs():
     5   38.828 MiB    0.012 MiB           1       for name in docs_to_process:
     6  192.453 MiB  152.625 MiB           1           doc = frappe.get_doc(“type”, name)
     7  204.853 MiB   12.40  MiB           1           doc.process()
```

Frappe

“Background job crashing from memory usage”

1. Find entry point:

   `repost_entries()`
2. **`@memory_profiler.profile`**
3. Keep narrowing down till you find the root cause.
4. …
5. profit?



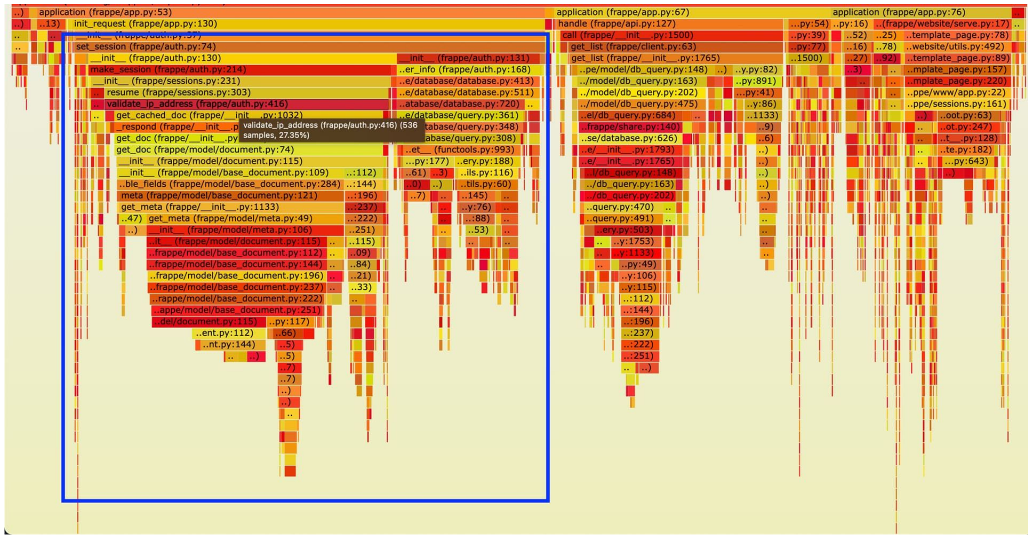
Frappe

Common patterns

	Knowns	**Unknowns**
Known	Specific known bottlenecks	Overheads from libraries and abstractions boundaries
Unknown	Our own "small" unknown overheads	*Good luck!*

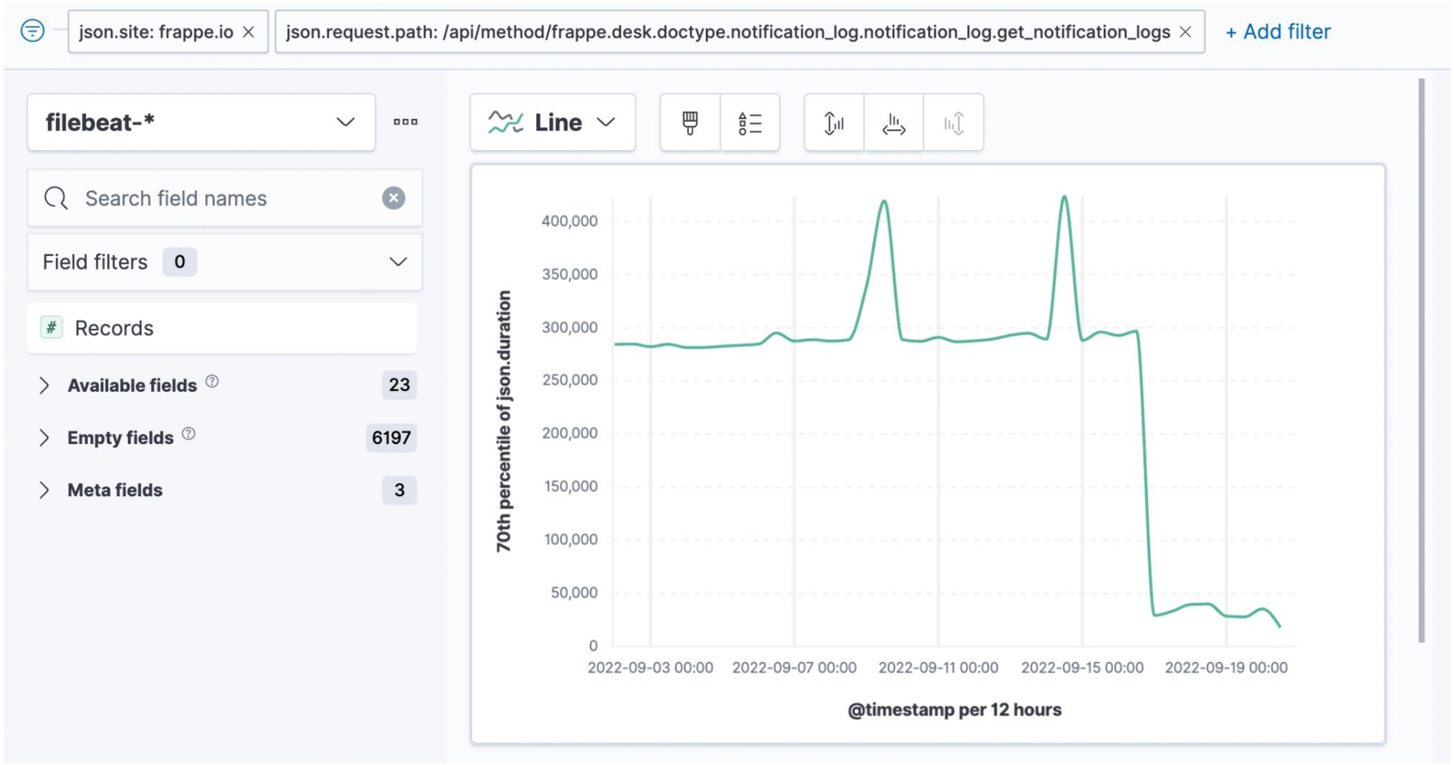
“Response times seem slow”

1. Entry point: **??**
2. Need high level overview.
3. Use the *right* tool - `py-spy`.
4. …
5. 30-50% reduction in overheads. profit?



Frappe

“Response times seem slow”



json.site: frappe.io ×

json.request.path: /api/method/frappe.desk.doctype.notification_log.notification_log.get_notification_logs ×

+ Add filter

filebeat-*

Search field names

Field filters 0

Records

Available fields 23

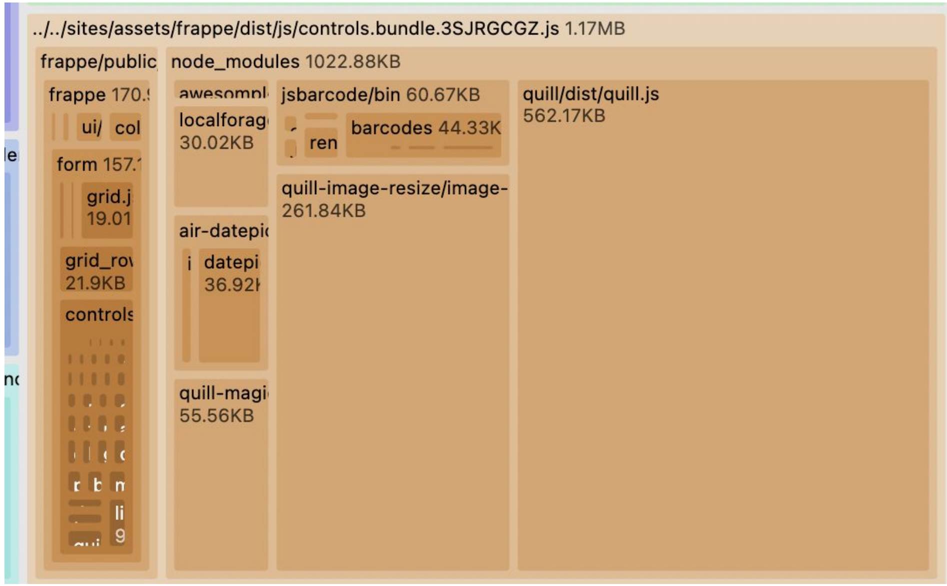
Empty fields 6197

Meta fields 3

Frappe

"Asset bundle size is huge"

1. Use the right tool - ESbuild meta files visualizer
2. Find duplicate, non-critical libraries and remove them
3. …
4. 5MB -> 3.4 MB. profit?



Frappe

"Too much disk usage"

1. Sort tables by usage, find top tables
2. Analyze tables.
3. …
4. 10%-50% drop in db usage..

~~Profit?~~

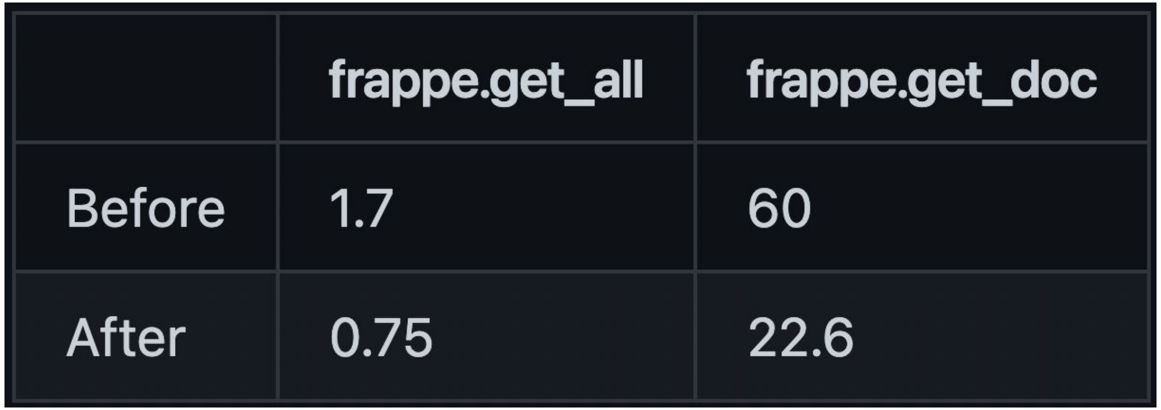
```
{
'name': '29438bf4aa',
'ref_doctype': 'Scheduled Job Log',
'docname': '33588173a2',
'data': '{\n "created_by":
"Administrator",\n "creation":
"2022-09-15 02:59:45.896342",\n
"updater_reference": null\n}',
 'doctype': 'Version'
}
```

Common patterns

	Knowns	**Unknowns**
Known	Specific known bottlenecks	Overheads from libraries and abstractions boundaries
Unknown	Our own "small" unknown overheads	*Good luck!*

Database client library

- PyMySQL (pure python) vs mariadb (C)
- ~2.5x faster execution just by swapping libraries



	frappe.get_all	frappe.get_doc
Before	1.7	60
After	0.75	22.6

Frappe

"`delete` not possible on large DBs"

1. `delete` query on ~95% of data in large tables just fail.
2. No amount of query optimization can fix this.

Fix: Copy 5% data to a temporary table and swap tables.

Frappe

On *Micro-optimizations*

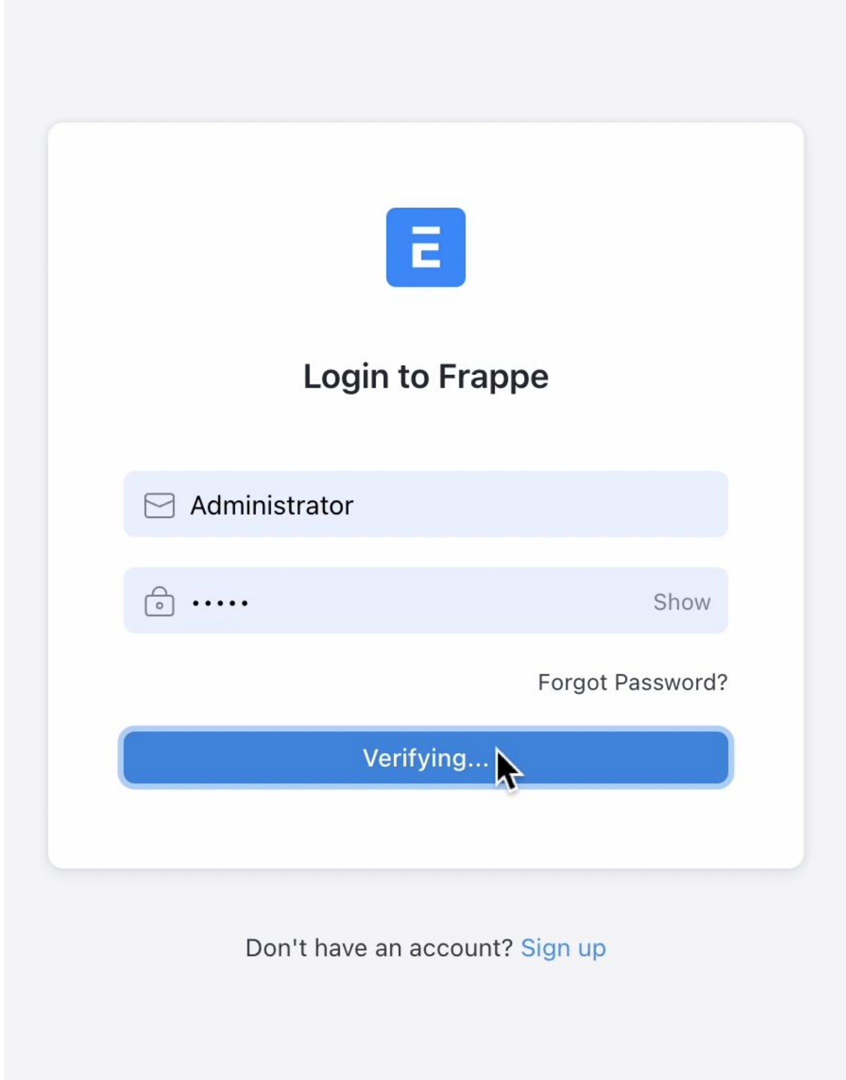
- `` `frappe._` `` is few microseconds faster.
- `` `doc.get` `` is few microseconds faster
- Pre-compiling regexps
- `get_cached_value` uses dict instead of doc.

“बूँद-बूँद से सागर भरता है”

(The water droplets will one day will an ocean.)

What about feels?

- Numbers don't translate 1-1 with "feels"
- UX improvements:
  - Skeleton loaders
  - Faster splash screens


Login to Frappe

Administrator

•••••

Tools revisited

1. **cProfile / %prun** - inbuilt profiler
2. **py-spy** - *sampling* profile best for overheads / in prod.
3. **Browser's Dev tools** - all things client side
4. **memory_profiler** / **scalene** - memory profiling
5. **Frappe Monitor** - monitoring response times and requests
6. **Frappe Recorder** - Analyzing SQL queries in any request.
7. **Custom scripts** - when everything else falls short.

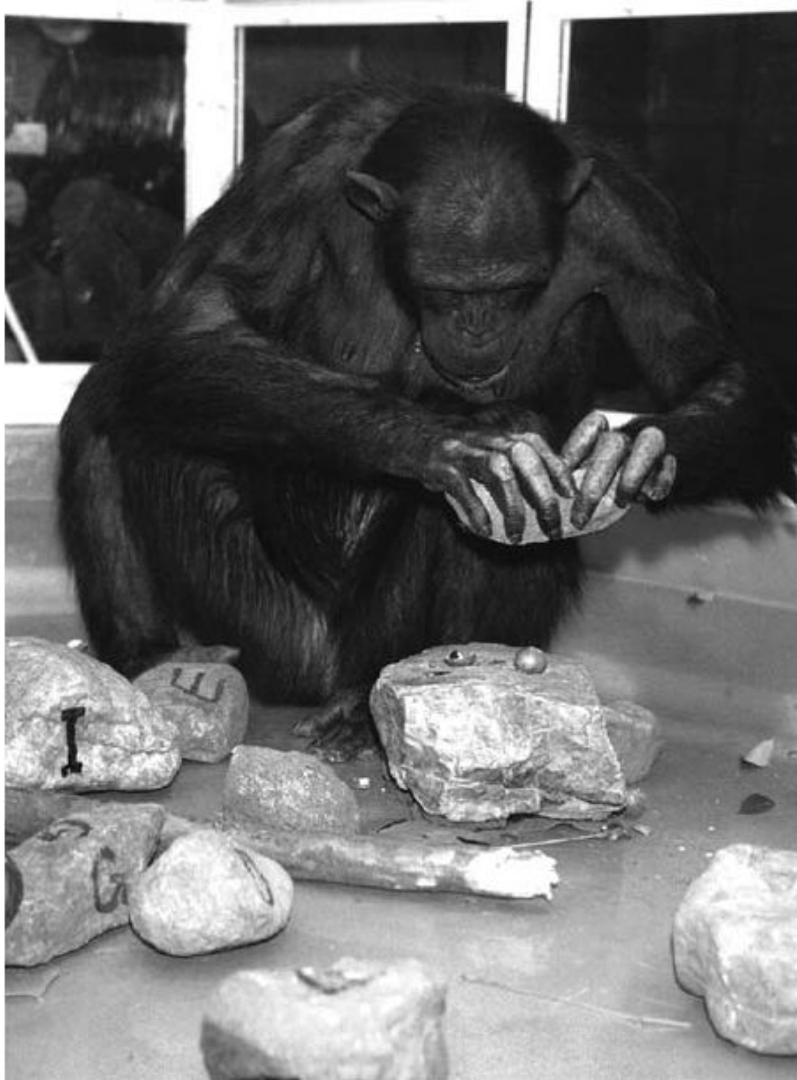
Non-exhaustive list of last year’s perf fixes

- 50% faster doc.as_dict
- Faster get_cached_doc()
- Faster workflow actions
- Faster desk routing
- Better DB indexes
- Faster website pages
- Single doc caching
- …

- Faster BOM update
- Faster reposting
- Faster invoice submission
- Faster PCV submission
- Faster barcode scan
- Faster variant selector
- Faster stock balance report
- …

Key takeaways

- Trust nothing, profile everything.
- Know thy abstractions.
- Learn some tools, write some tools. Life isn’t easy without them.
- Most performance issues are not deeply *technical*. Just need 🕵️‍♀️



The Road Ahead

- Performance regression tests
- Persistent DB connections
- Faster Desk routing
- Better client side caching
- More “***scratching the itch***”

⭐ and watch Frappe for updates.

Frappe

Questions?

Frappe