Microsoft

The state of JIT in PostgreSQL

Daniel Gustafsson

PGConf.NYC 2023

Daniel Gustafsson

daniel@yesql.se

Daniel Gustafsson

daniel@yesql.se

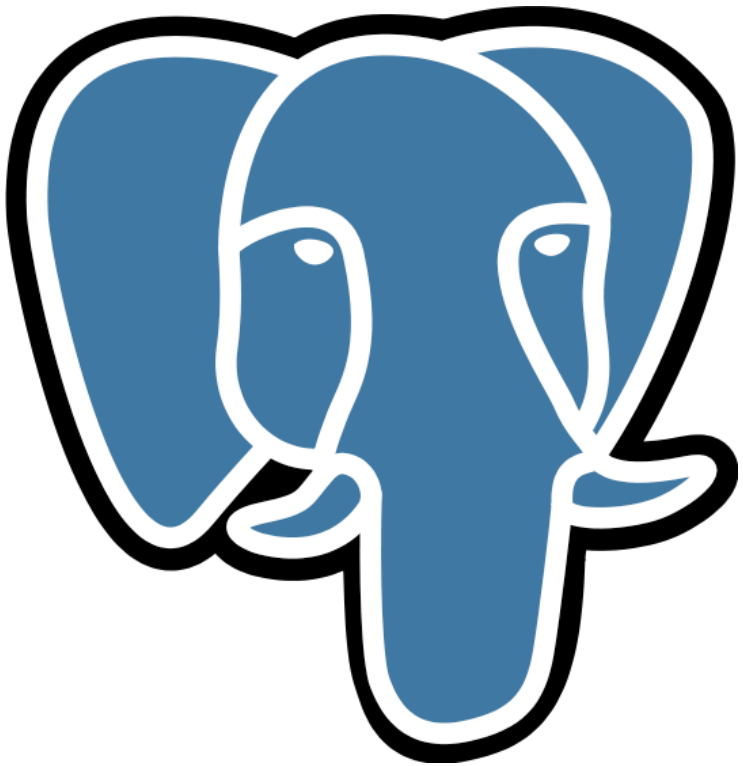

Microsoft
Microsoft Office Professional
Diskette 1 - Installationsdiskette
1
Windows-Serie
Diskettenformat: HD (1,44 MB)
Version 4.3 ★
Installationsanleitung:
1. Starten Sie Microsoft Windows.
2. Legen Sie Diskette 1 - Installationsdiskette - in Laufwerk A: ein.
3. Wählen Sie im Programm-Manager aus dem Menü Datei den Befehl Ausführen.
4. Geben Sie a:\setup ein, und drücken Sie die EINGABETASTE.
D16927
© 1993-1995 Microsoft Corporation. Alle Rechte vorbehalten.
Disk Assy D16927

Contributor, Committer

Web, Stockholm PUG, etc etc..

Nordic PGDay, PGConf EU

FOSDEM PGDay



curl://

Contributor, Committer

Security Team

Side A

What is JIT

Concepts

History

Side B

JIT in postgres

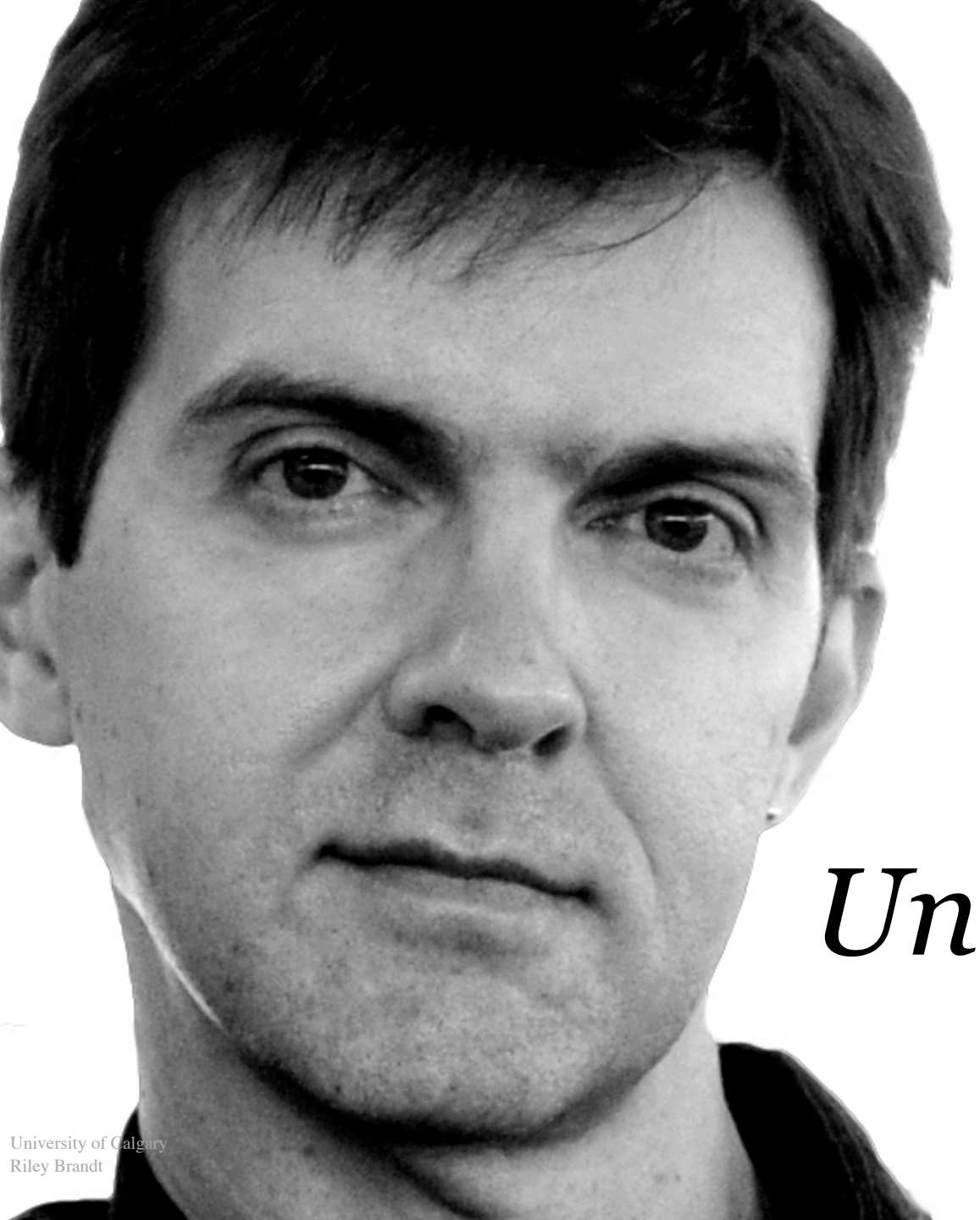
What & How

Current state

Future plans

JIT

General concepts



Professor

John Aycock

University of Calgary

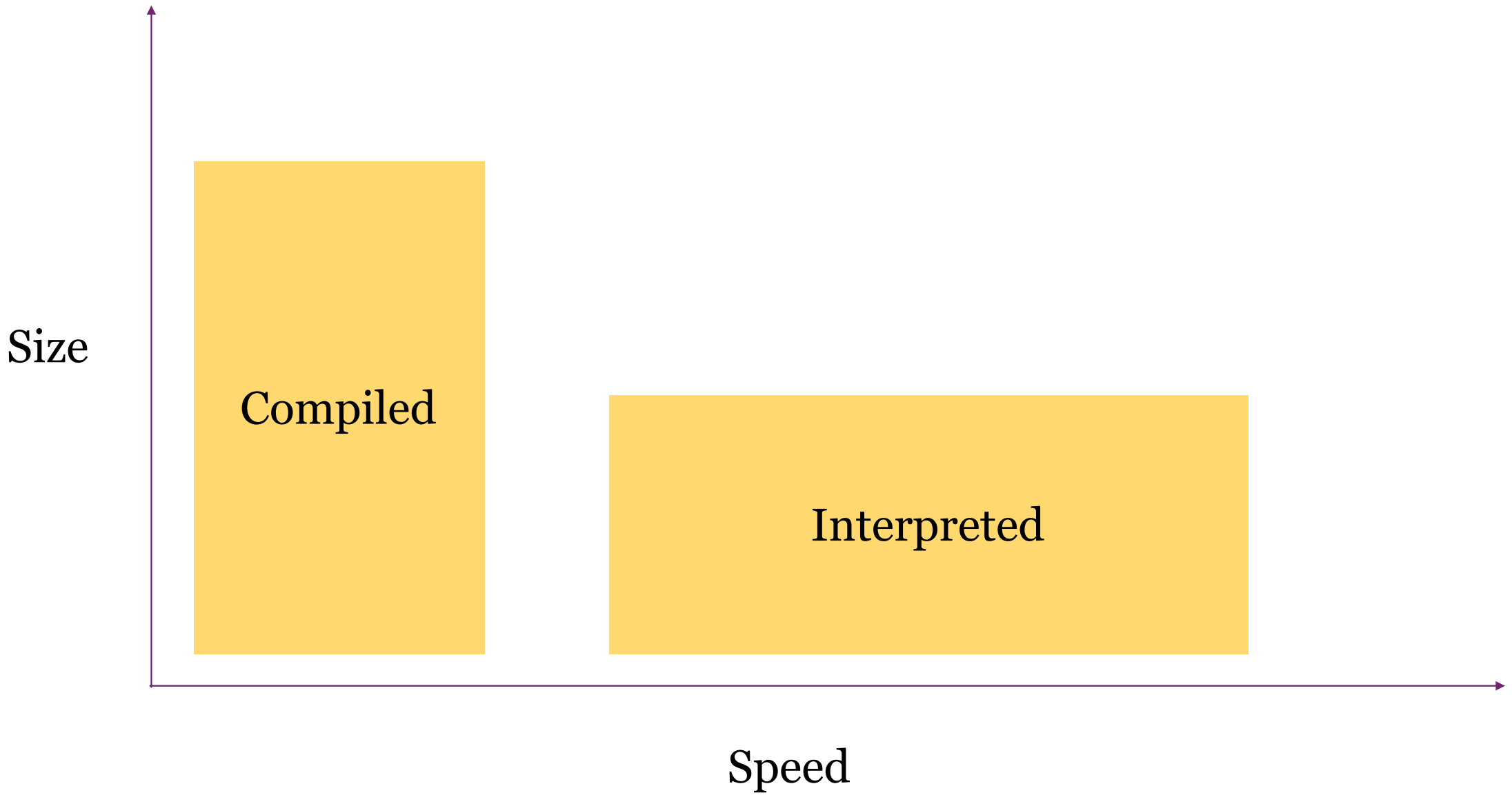
Strictly speaking, JIT compilation systems ("JIT Systems" for short) are completely unnecessary.

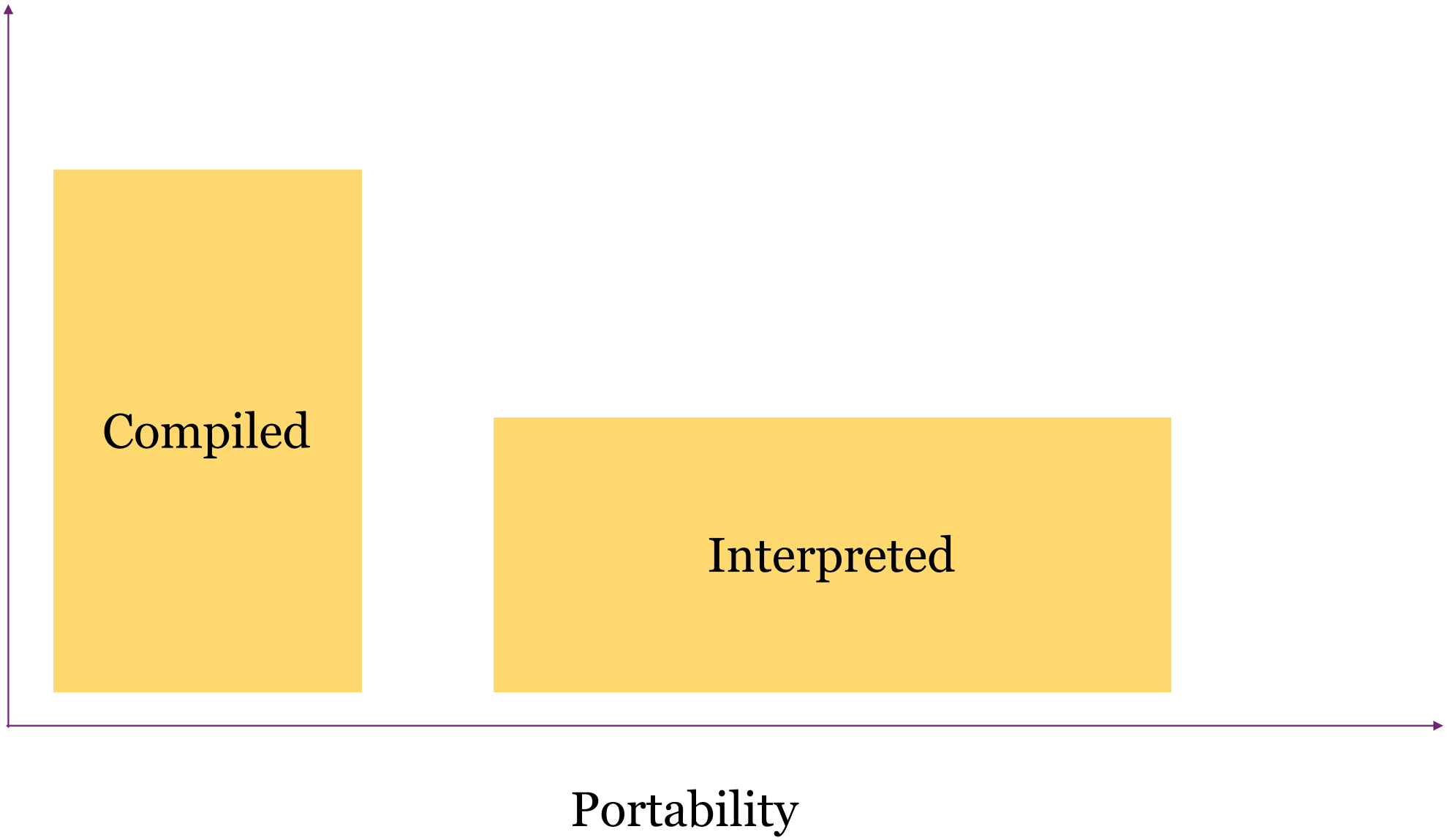
-- Prof. John Aycock
A Brief History of Just-in-Time

Strictly speaking, JIT compilation systems ("JIT Systems" for short) are completely unnecessary.

They are only a means to improve the time and space efficiency of programs.

-- Prof. John Aycock
A Brief History of Just-in-Time


Size
Compiled
Interpreted
Speed


Compiled
Interpreted
Portability

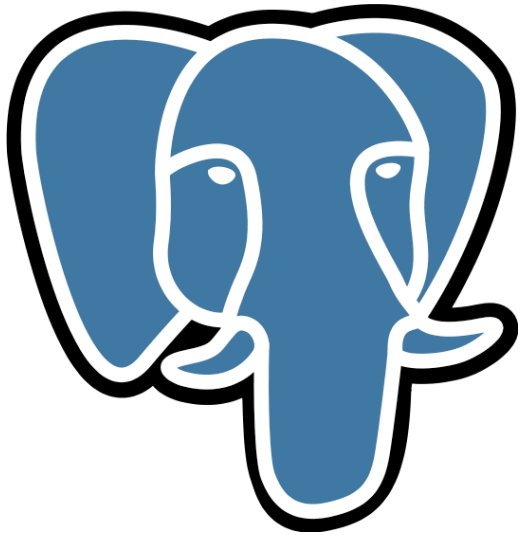
Tradeoff Management

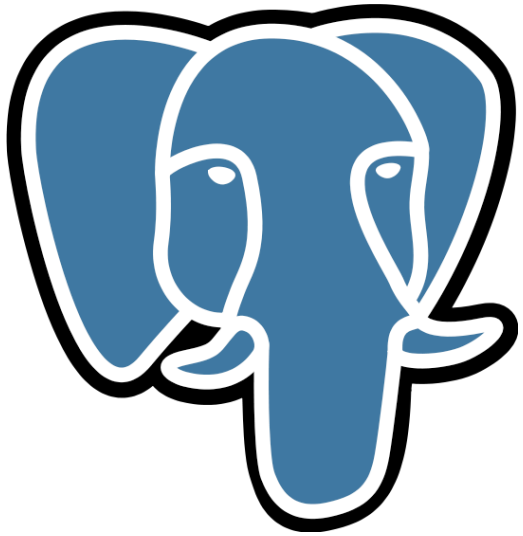
*E*xecutable size

*S*peed of execution

*P*ortability

*R*untime feedback





parser

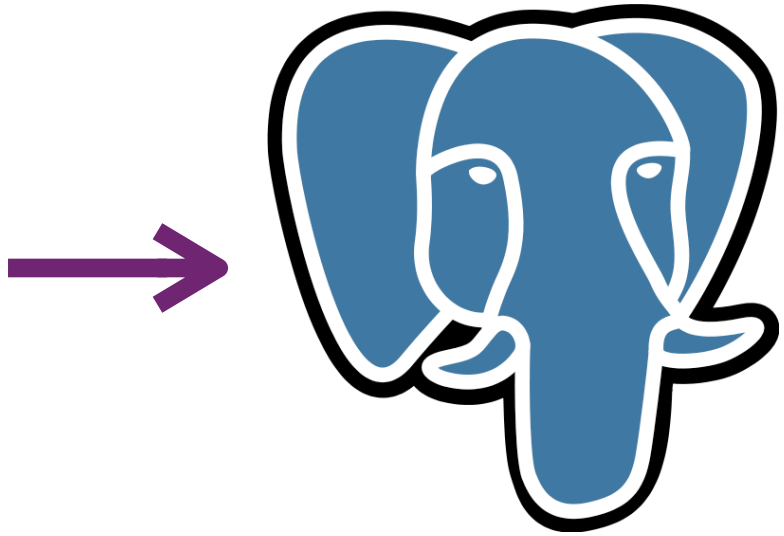
planner

executor

misc

data

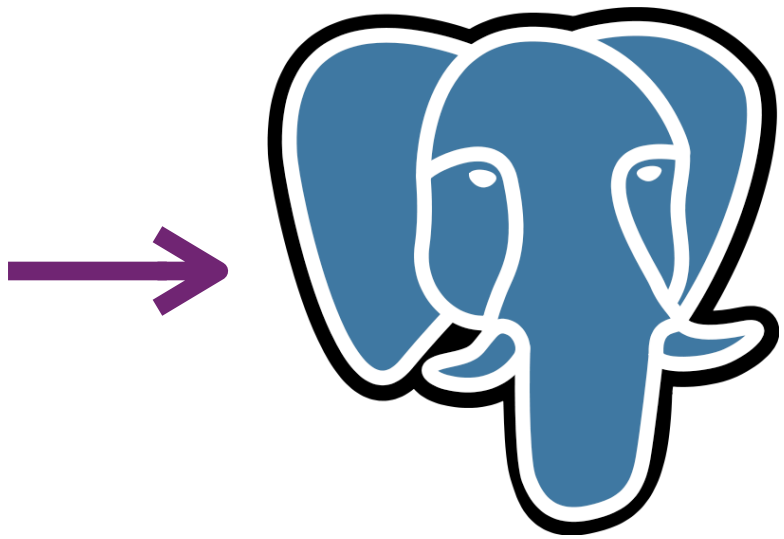
```
SELECT
      lorem
FROM
      ipsum;
```

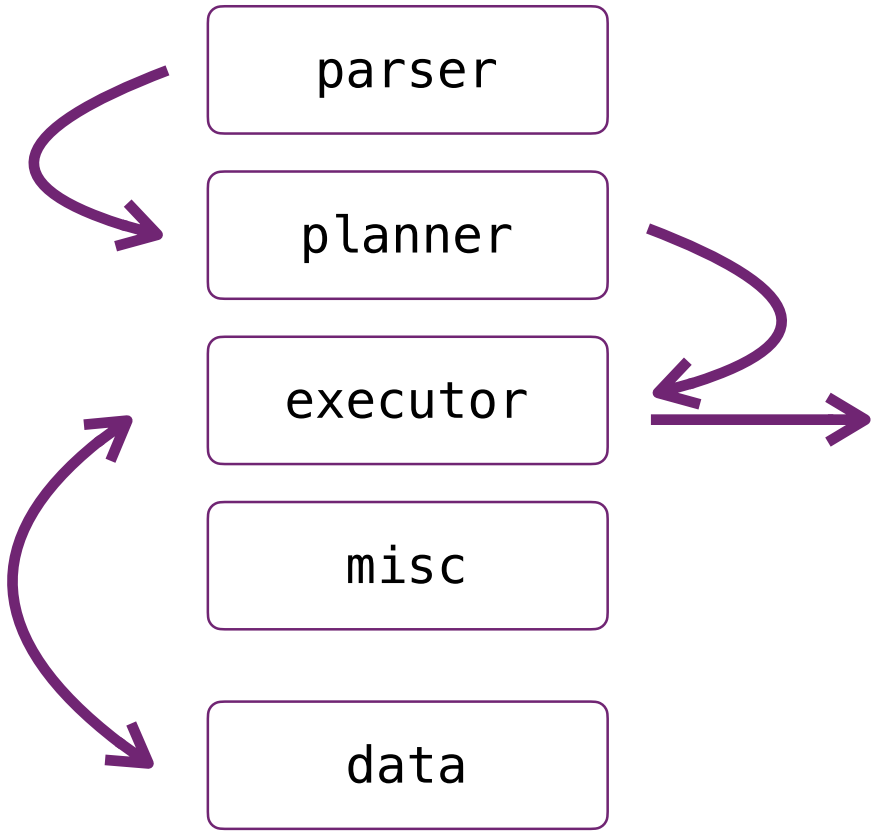


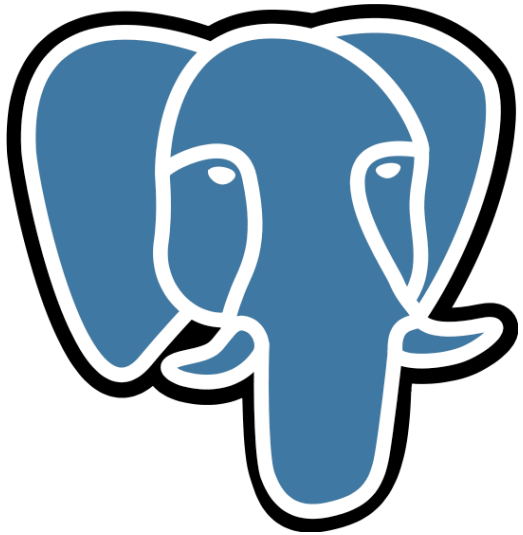
- parser
- planner
- executor
- misc

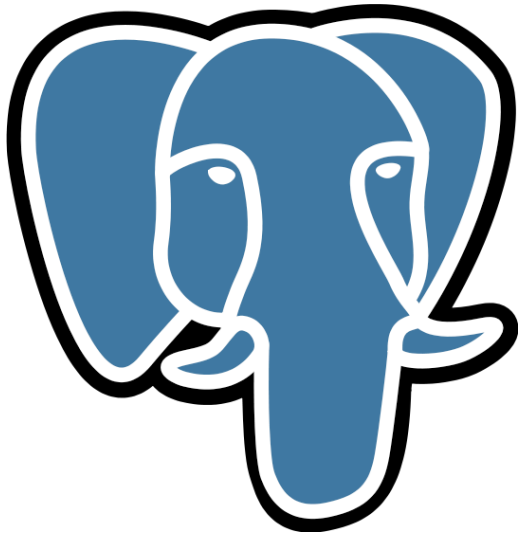
- data

```
SELECT
      lorem
FROM
      ipsum;
```



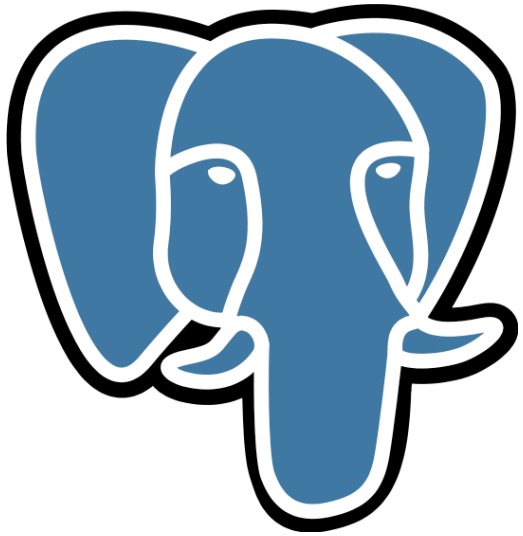




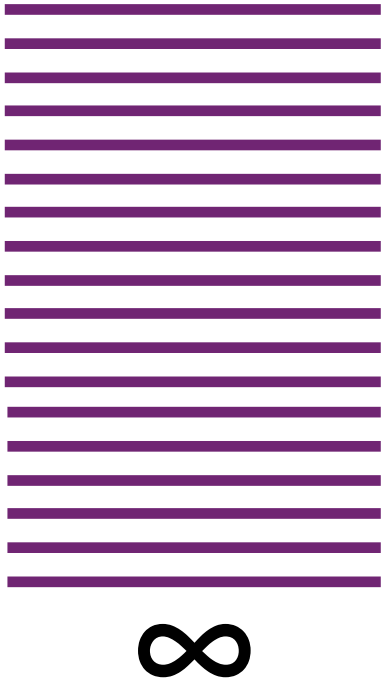


query_lookup()

data

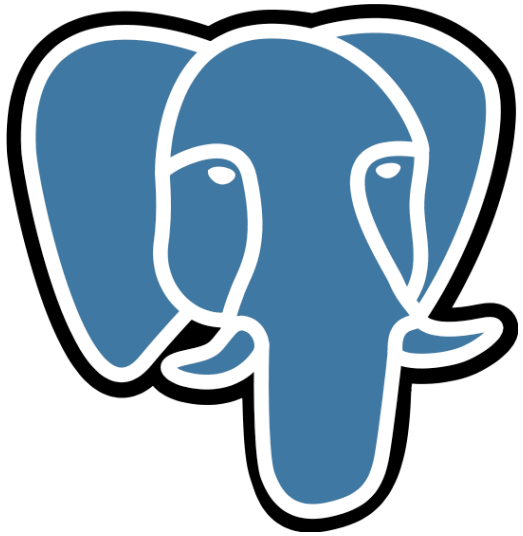


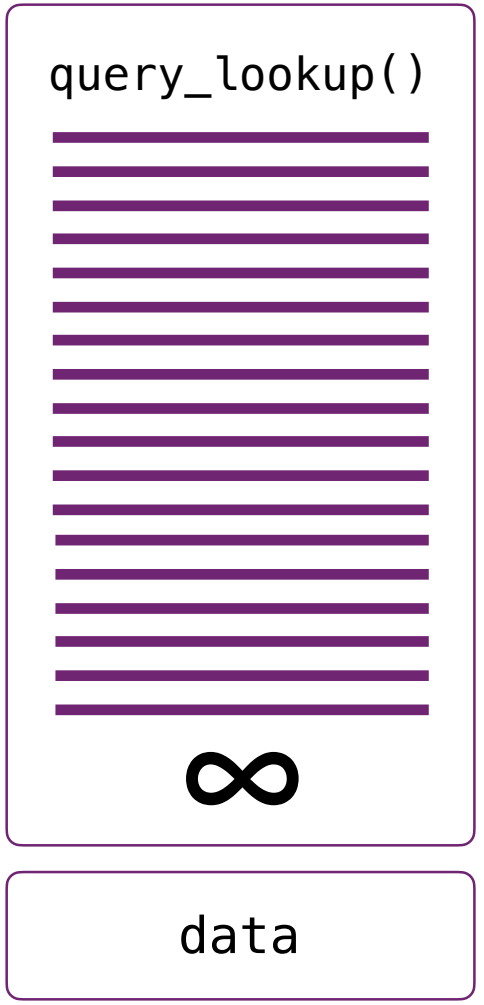
query_lookup()



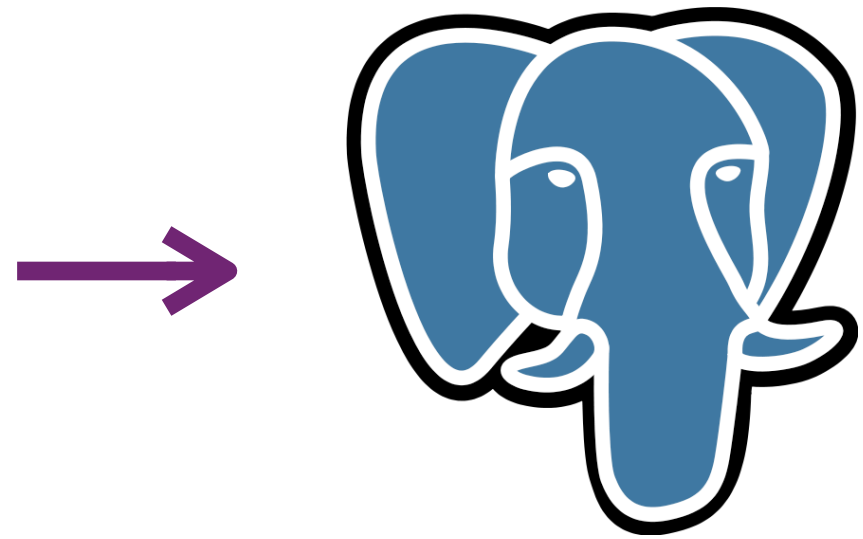
data

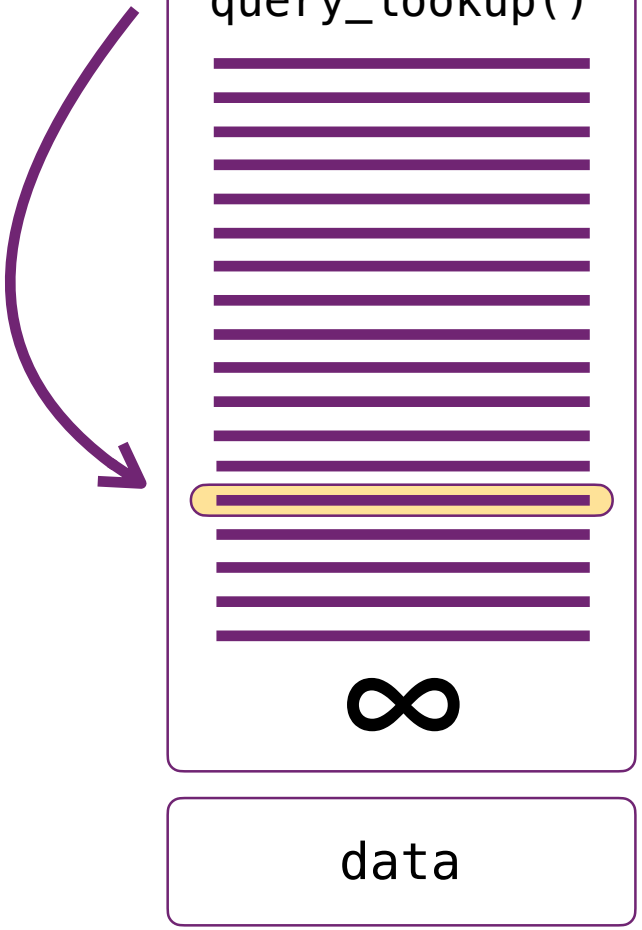
```
SELECT
      lorem
FROM
      ipsum;
```





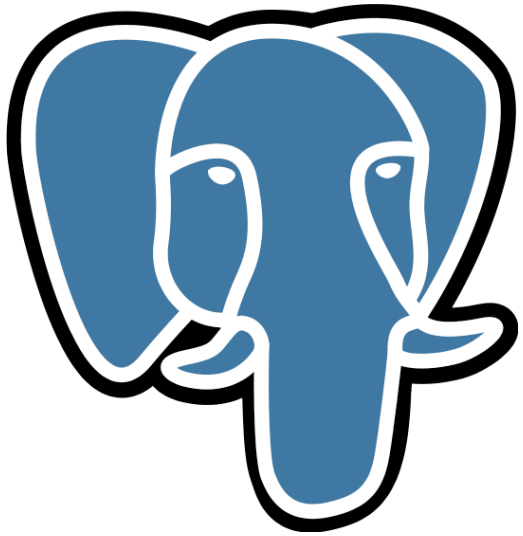
```
SELECT
      lorem
FROM
      ipsum;
```

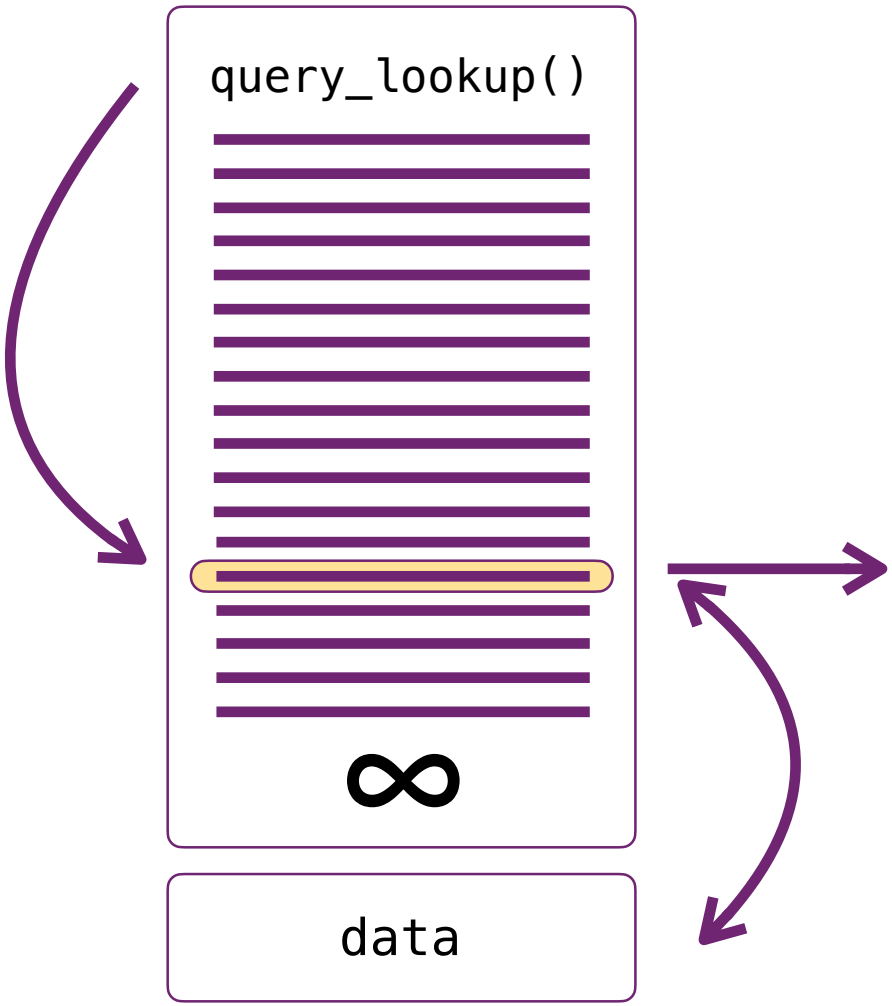


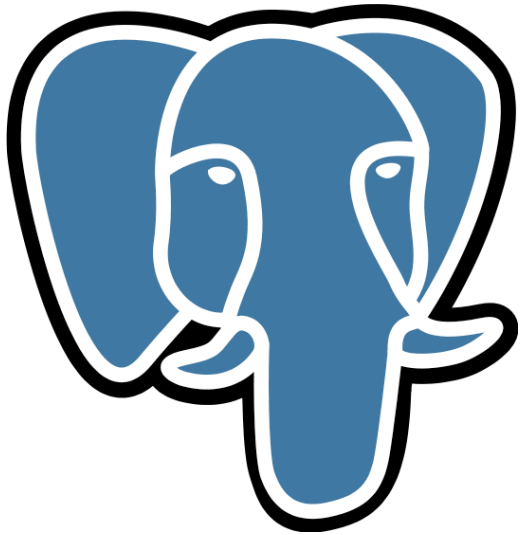


SELECT_lorem_FROM_ipsum.EXE

```
SELECT
      lorem
FROM
      ipsum;
```







parser

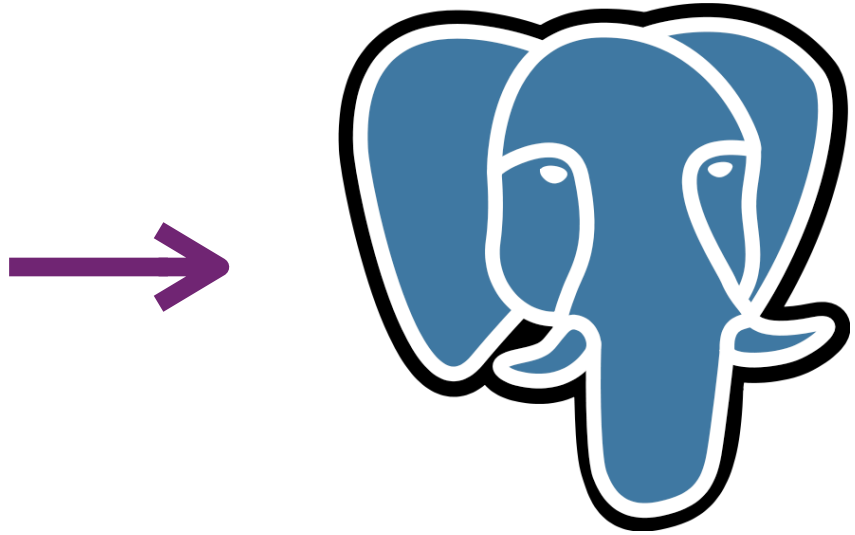
planner

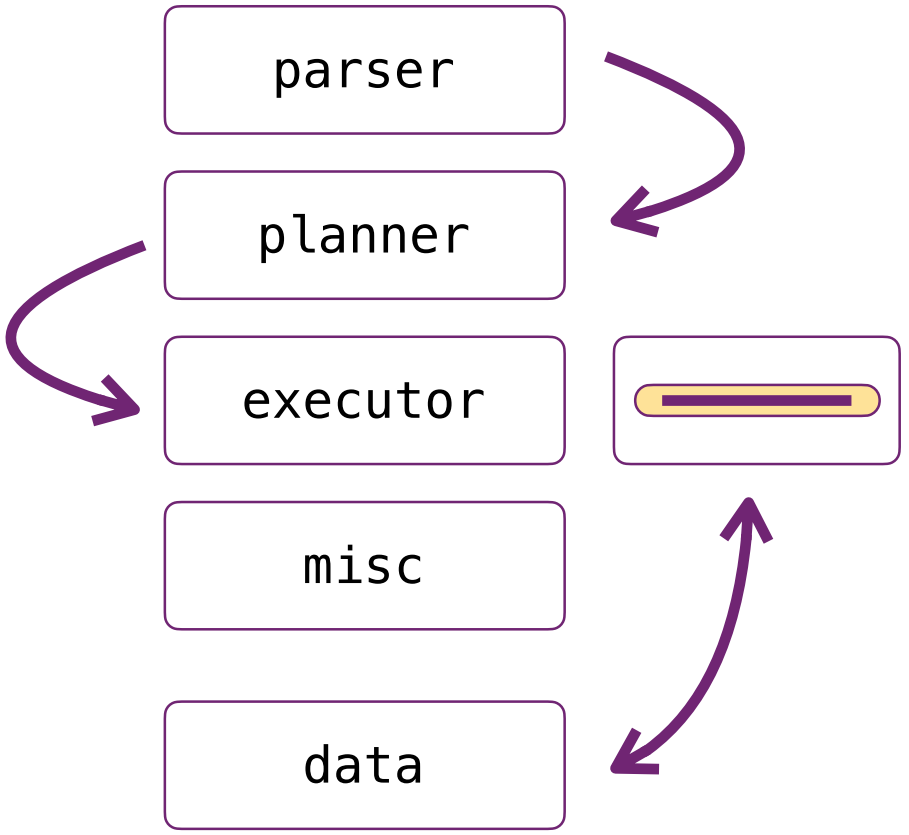
executor

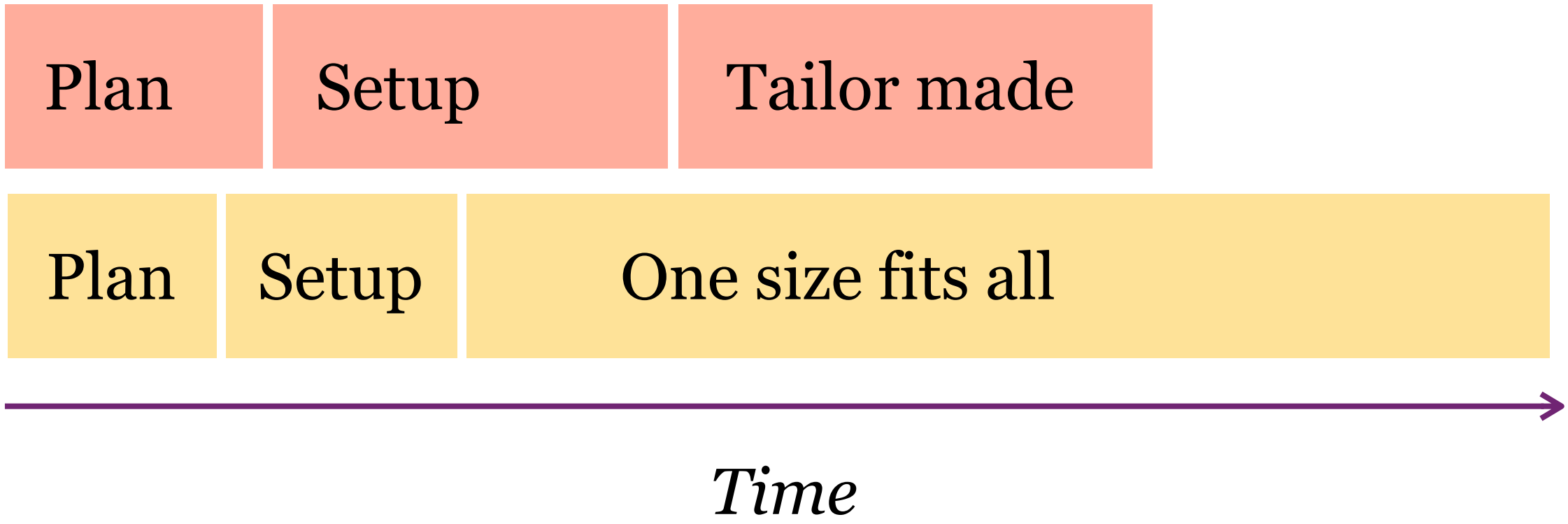
misc

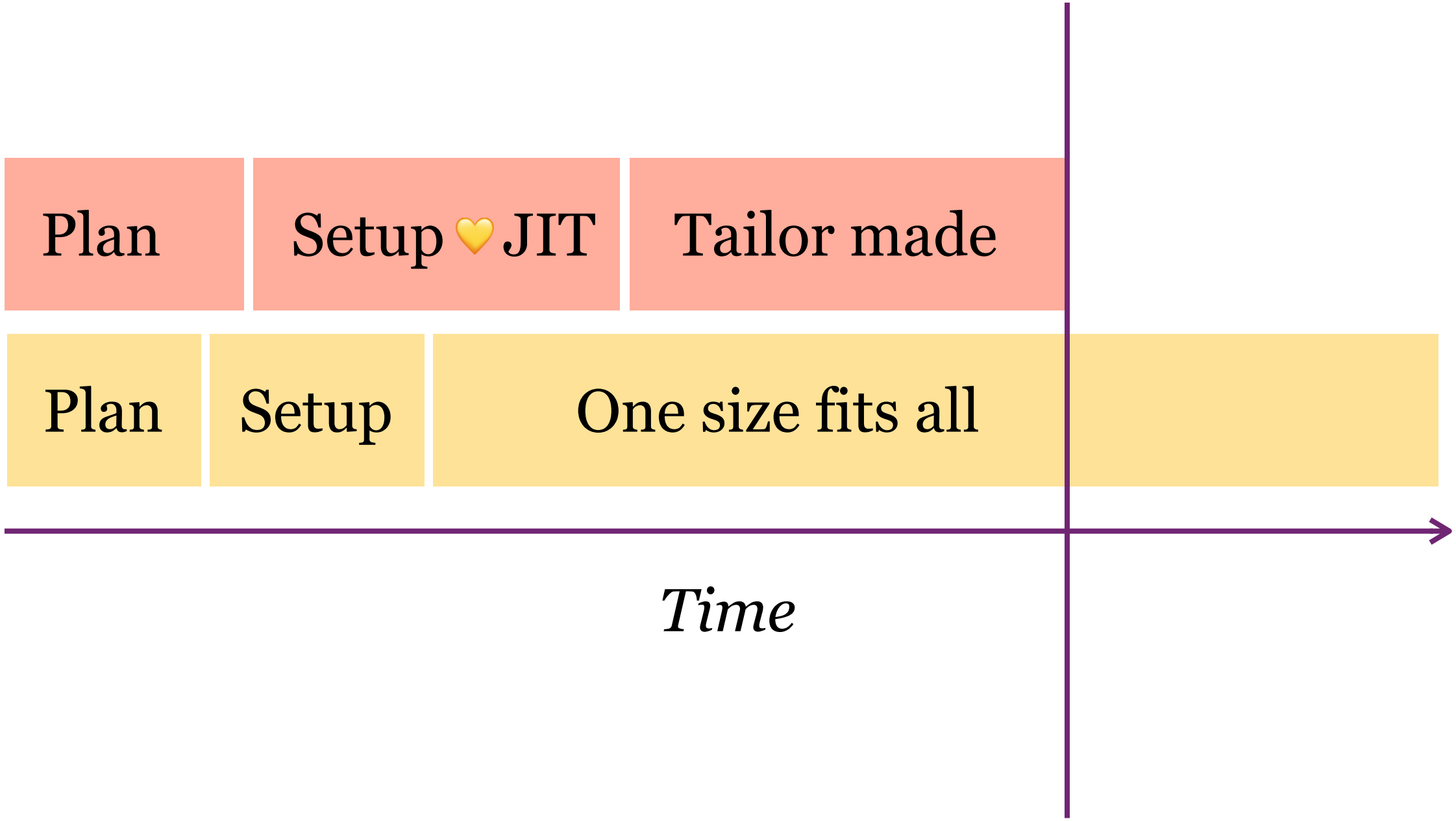
data

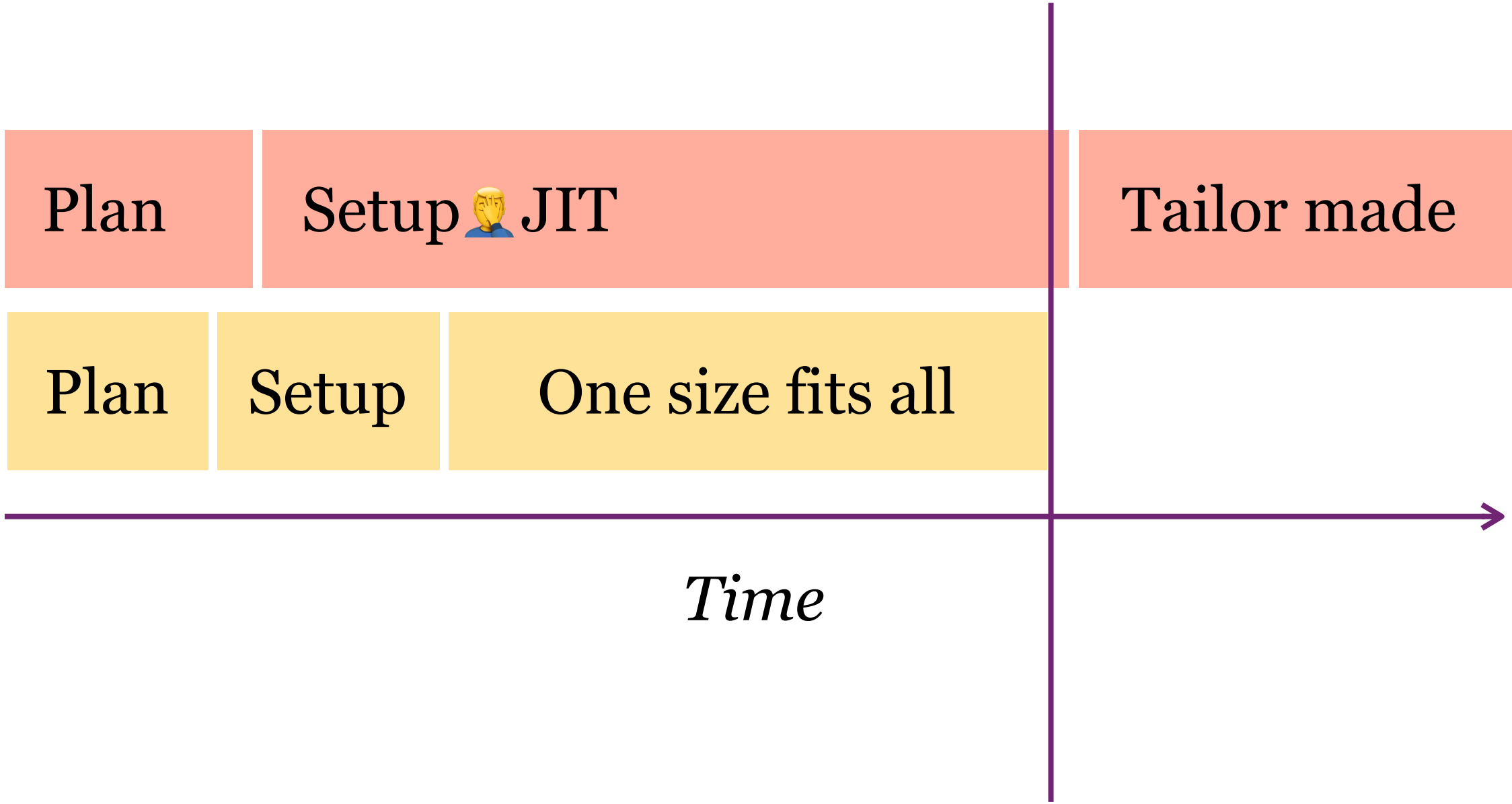
```
SELECT
      lorem
FROM
      ipsum;
```






Plan
Setup
Tailor made
Plan
Setup
One size fits all
Time


Plan
Setup 💛 JIT
Tailor made
Plan
Setup
One size fits all
Time


Plan
Setup🤦JIT
Tailor made
Plan
Setup
One size fits all
Time

https://sv.wikipedia.org/wiki/Minidator#/media/Fil:Pdp7-oslo-2005.jpeg


https://flickr.com/photos/warshawski/9414014122

Ken Thompson

Faces of Open Source
Peter Adams

Regular Expression Search Algorithm

KEN THOMPSON
Bell Telephone Laboratories, Inc., Murray Hill, New Jersey

A method for locating specific character strings embedded in character text is described and an implementation of this method in the form of a compiler is discussed. The compiler accepts a regular expression as source language and produces an IBM 7094 program as object language. The object program then accepts the text to be searched as input and produces a signal every time an embedded string in the text matches the given regular expression. Examples, problems, and solutions are also presented.

KEY WORDS AND PHRASES: search, match, regular expression
CR CATEGORIES: 3.74, 4.49, 5.32

Faces of Open Source
Peter Adams

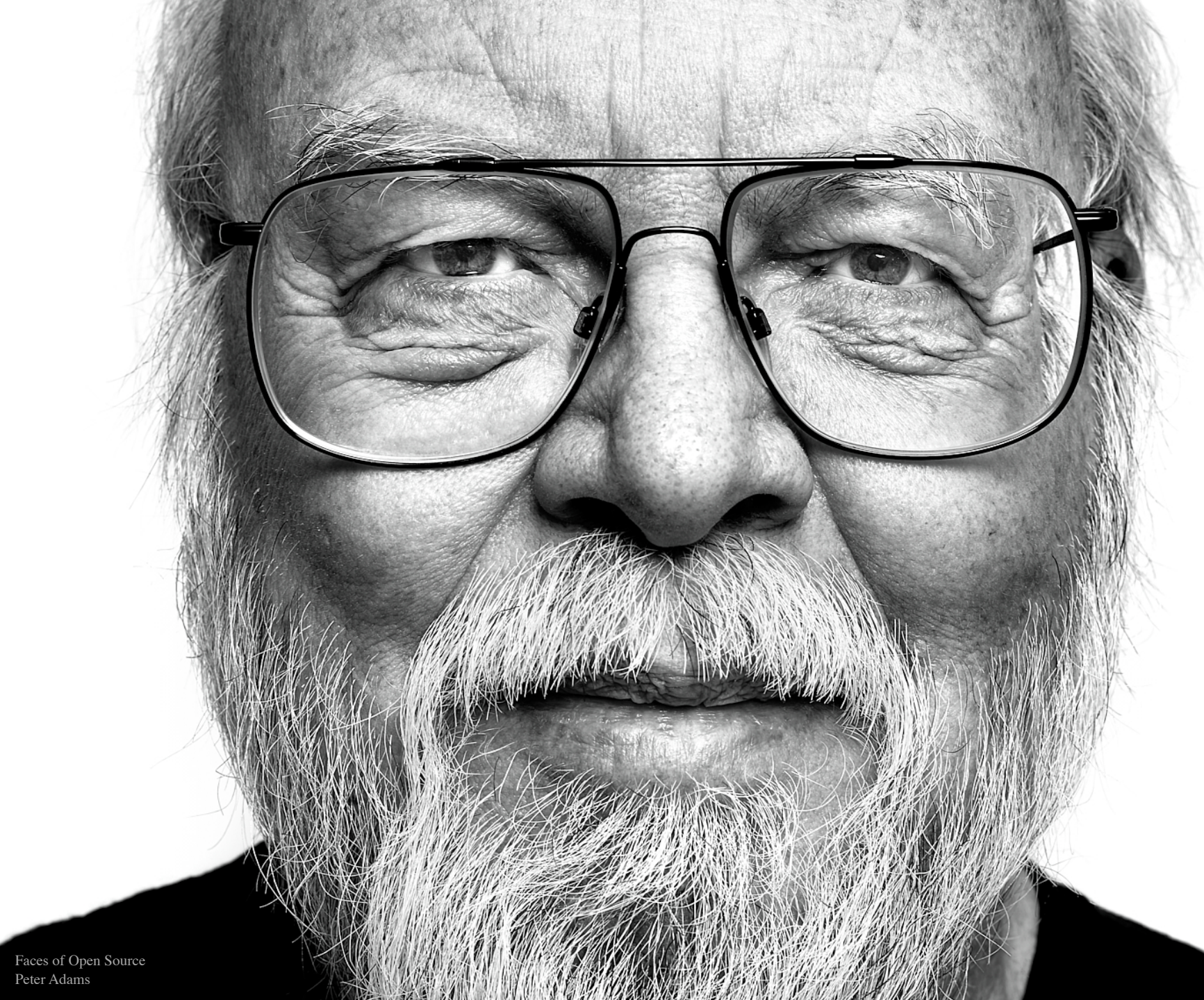
Regular Expression Search Algorithm

KEN THOMPSON
Bell Telephone Laboratories, Inc., Murray Hill, New Jersey

A method for locating specific character strings embedded in character text is described and an implementation of this method in the form of a compiler is discussed. The compiler accepts a regular expression as source language and produces an IBM 7094 program as object language. The object program then accepts the text to be searched as input and produces a signal every time an embedded string in the text matches the given regular expression. Examples, problems, and solutions are also presented.

KEY WORDS AND PHRASES: search, match, regular expression
CR CATEGORIES: 3.74, 4.49, 5.32

Faces of Open Source
Peter Adams

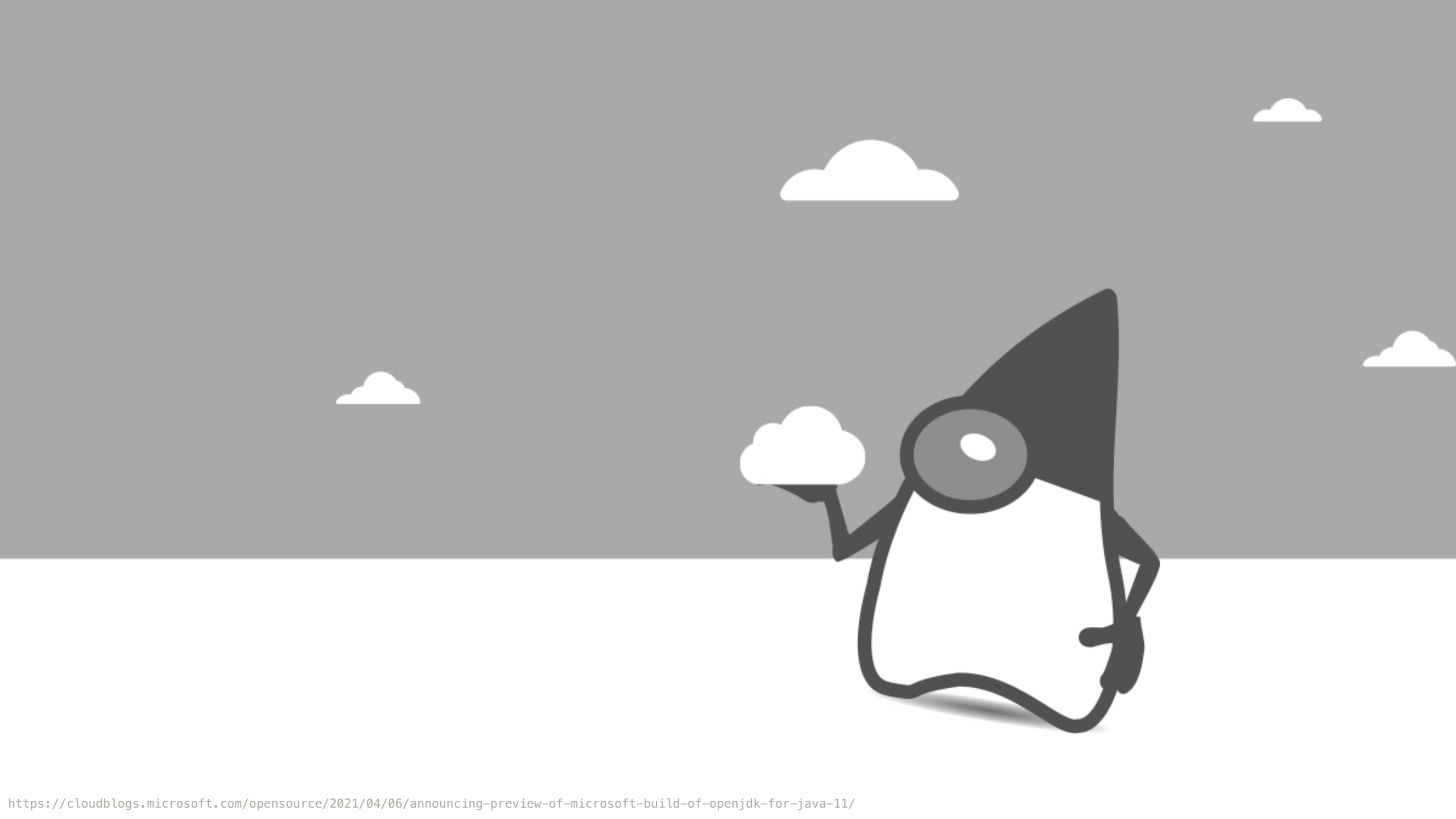


James Gosling

Faces of Open Source
Peter Adams

Java isn't just slow, it's really slow, surprisingly slow

-- *Paul Tyma, Why are we using Java again? Communications of the ACM 41, 1998*



https://cloudblogs.microsoft.com/opensource/2021/04/06/announcing-preview-of-microsoft-build-of-openjdk-for-java-11/

JIT

in PostgreSQL

History

Introduced in v*11*

Enabled by default in v*12* if compiled with LLVM

`pg_stat_statements` counters in v*15*

Support for new LLVM releases

History

Introduced in v*11*

Enabled by default in v*12* if compiled with LLVM

`pg_stat_statements` counters in v*15*

Support for new LLVM releases

Capabilities

*T*uple deforming

*E*xpression Evaluation

*I*nlining

Tuple deforming

*T*he act of reading a tuple from storage into a memory structure

*K*nowing the number of columns and their types can remove branches

Expression Evaluation

`WHERE` clauses

*T*urning an evaluation of a specific expression into native code only for that case

Inlining

*R*educing the overhead of postgres' extensible nature

*D*uplicating operators is unwanted

*E*xisting function compiled to IR by clang and bitcode installed

Configuration

```
jit_above_cost = <int>

jit_inline_above_cost = <int>

jit_optimize_above_cost = <int>
```

`jit_above_cost = n`

All queries with a cost higher than *n* get JIT compiled without optimization

Compiled with `-O0`

`jit_inline_above_cost = n`

Inlining will be performed for queries with a cost higher than *n*

`jit_optimize_above_cost = n`

All queries with a cost higher than *n* get JIT compiled *with* optimization

```
postgres=#
```

```
postgres=# SHOW jit_above_cost;
 jit_above_cost
----------------
 100000
(1 row)

postgres=#
```

```
postgres=# SHOW jit_above_cost;
 jit_above_cost
----------------
 100000
(1 row)

postgres=# EXPLAIN (TIMING OFF, ANALYZE)
           SELECT SUM(relpages) FROM pg_class;
                          QUERY PLAN
-----------------------------------------------------------------
 Aggregate  (cost=18.18..18.19 rows=1 width=8)
            (actual rows=1 loops=1)
   ->  Seq Scan on pg_class  (cost=0.00..17.14 rows=414 width=4)
                             (actual rows=421 loops=1)
 Planning Time: 0.112 ms
 Execution Time: 0.188 ms
(4 rows)
```

```
postgres=# SHOW jit_above_cost;
 jit_above_cost
----------------
 100000
(1 row)

postgres=# EXPLAIN (TIMING OFF, ANALYZE)
           SELECT SUM(relpages) FROM pg_class;
                          QUERY PLAN
-----------------------------------------------------------------
 Aggregate  (cost=18.18..18.19 rows=1 width=8)
            (actual rows=1 loops=1)
   ->  Seq Scan on pg_class  (cost=0.00..17.14 rows=414 width=4)
                             (actual rows=421 loops=1)
 Planning Time: 0.112 ms
 Execution Time: 0.188 ms
(4 rows)
```

```
postgres=#
```

```
postgres=# SET jit_above_cost TO 10;
SET

postgres=#
```

```
postgres=# SET jit_above_cost TO 10;
SET

postgres=# EXPLAIN (TIMING OFF, ANALYZE)
           SELECT SUM(relpages) FROM pg_class;
                                QUERY PLAN
-------------------------------------------------------------------------
 Aggregate  (cost=18.18..18.19 rows=1 width=8)
            (actual rows=1 loops=1)
   ->  Seq Scan on pg_class  (cost=0.00..17.14 rows=414 width=4)
                             (actual rows=421 loops=1)
 Planning Time: 0.115 ms
 JIT:
   Functions: 3
   Options: Inlining false, Optimization false,
            Expressions true, Deforming true
 Execution Time: 8.843 ms
(7 rows)
```

When is it useful?

*L*ong running queries

*C*PU-bound workloads

*A*mortizing the cost of the JIT compilation process

When is it useful?

*A*nalytical queries

*W*hen it transparently made the query run faster..

*O*ptimization, not functionality

Extensions?

THE DESIGN OF POSTGRES

Michael Stonebraker and Lawrence A Rowe

Department of Electrical Engineering
and Computer Sciences
University of California
Berkeley, CA 94720

Abstract

This paper presents the preliminary design of a new database management system, called POSTGRES, that is the successor to the INGRES relational database system The main design goals of the new system are to

1) provide better support for complex objects,

2) provide user extendibility for data types, operators and access methods,

3) provide facilities for active databases (i e , alerters and triggers) and inferencing including forward- and backward-chaining,

4) simplify the DBMS code for crash recovery,

5) produce a design that can take advantage of optical disks, workstations composed of multiple tightly-coupled processors, and custom designed VLSI chips, and

6) make as few changes as possible (preferably none) to the relational model

THE DESIGN OF POSTGRES

Michael Stonebraker and Lawrence A Rowe

Department of Electrical Engineering
and Computer Sciences
University of California
Berkeley, CA 94720

Abstract

This paper presents the preliminary design of a new database management system, called POSTGRES, that is the successor to the INGRES relational database system The main design goals of the new system are to

1) provide better support for complex objects,

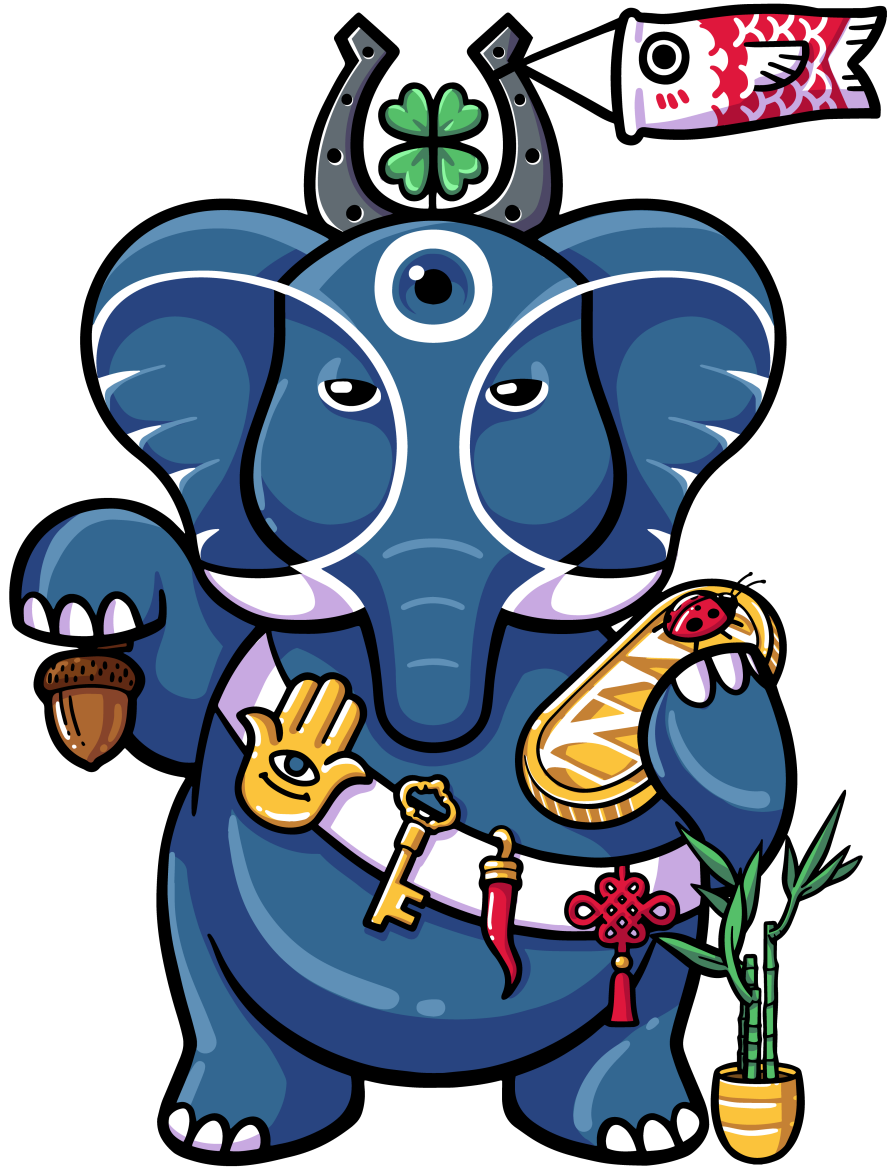
2) provide user extendibility for data types, operators and access methods,

3) provide facilities for active databases (i e , alerters and triggers) and inferencing including forward- and backward-chaining,

4) simplify the DBMS code for crash recovery,

5) produce a design that can take advantage of optical disks, workstations composed of multiple tightly-coupled processors, and custom designed VLSI chips, and

6) make as few changes as possible (preferably none) to the relational model



PostgreSQL
VERSION 13

Extensions!

*E*xtensions built with PGXS against a JIT enabled server get bitcode compiled and installed

*C*an be inlined like core postgres code

```
$ ls lib/bitcode/pg_stat_statements*
lib/bitcode/pg_stat_statements.index.bc

lib/bitcode/pg_stat_statements:
pg_stat_statements.bc
```

```
$ ls lib/bitcode/pg_stat_statements*
lib/bitcode/pg_stat_statements.index.bc

lib/bitcode/pg_stat_statements:
pg_stat_statements.bc

$ llvm-dis lib/bitcode/pg_stat_statements/pg_stat_statements.bc

; ModuleID = 'lib/bitcode/pg_stat_statements/pg_stat_statements.bc'
source_filename = "pg_stat_statements.c"
target datalayout = "e-m:o-p270:32:32-p271:32:32-p272:64:64-i64:64-
target triple = "x86_64-apple-macosx12.0.0"
```

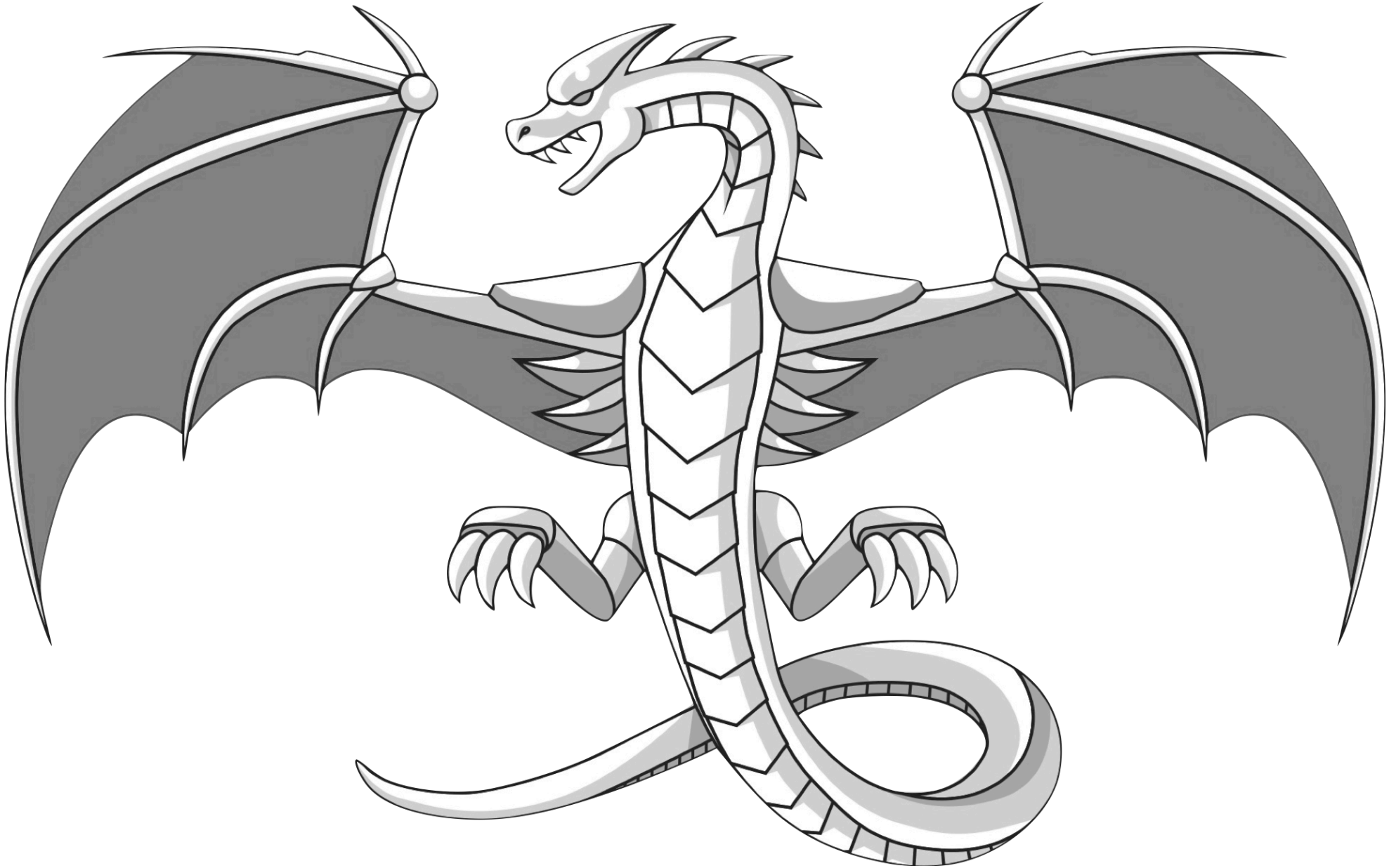
```
$ ls lib/bitcode/pg_stat_statements*
lib/bitcode/pg_stat_statements.index.bc

lib/bitcode/pg_stat_statements:
pg_stat_statements.bc

$ llvm-dis lib/bitcode/pg_stat_statements/pg_stat_statements.bc

; ModuleID = 'lib/bitcode/pg_stat_statements/pg_stat_statements.bc'
source_filename = "pg_stat_statements.c"
target datalayout = "e-m:o-p270:32:32-p271:32:32-p272:64:64-i64:64-
target triple = "x86_64-apple-macosx12.0.0"
```

Extensions?



https://llvm.org/img/LLVMWyvernBig.png

https://llvm.org/img/LLVMWyvernBig.png

PostgreSQL

VERSION 14

```
jit_provider = llvmjit
```

```
$ otool -L lib/llvmjit.dylib
lib/llvmjit.dylib:
    /usr/local/opt/llvm@15/lib/libLLVM.dylib
    /usr/lib/libc++.1.dylib
    /usr/lib/libSystem.B.dylib
```

```
$ otool -L lib/llvmjit.dylib
lib/llvmjit.dylib:
    /usr/local/opt/llvm@15/lib/libLLVM.dylib
    /usr/lib/libc++.1.dylib
    /usr/lib/libSystem.B.dylib
```

```
jit_provider = BYO_jit
```

```
void
_PG_jit_provider_init(JitProviderCallbacks *cb)
{
    cb->reset_after_error = byo_reset_after_error;
    cb->release_context = byo_release_context;
    cb->compile_expr = byo_compile_expr;
}
```

```
void
_PG_jit_provider_init(JitProviderCallbacks *cb)
{
    cb->reset_after_error = byo_reset_after_error;
    cb->release_context = byo_release_context;
    cb->compile_expr = byo_compile_expr;
}
```

A small matter of programming

```
void
_PG_jit_provider_init(JitProviderCallbacks *cb)
{
    cb->reset_after_error = byo_reset_after_error;
    cb->release_context = byo_release_context;
    cb->compile_expr = byo_compile_expr;
}
```

A small matter of programming

(For some value of small)

The Road Ahead

No.

No.

James Coleman

Stampede of the JIT compilers

To: PostgreSQL Hackers, Cc: David Pirotte

<CAAaqYe-g-Q0Mm5H9QLcu8cHeMwok+HaxS4-UC9Oj3bK3

Hello,

We recently brought online a new database cluster, and in th
of ramping up traffic to it encountered a situation where a
query (analyzing helped with this, but I think the issue is
relevant) resulted in that query being compiled with JIT, an
large number of backends were running that same shape of que
them JIT compiling it. Since each JIT compilation took ~2s,

No.

JC

James Coleman

Stampede of the JIT

KK

Konstantin Knizhnik

One more problem with JIT

To: PostgreSQL Hackers

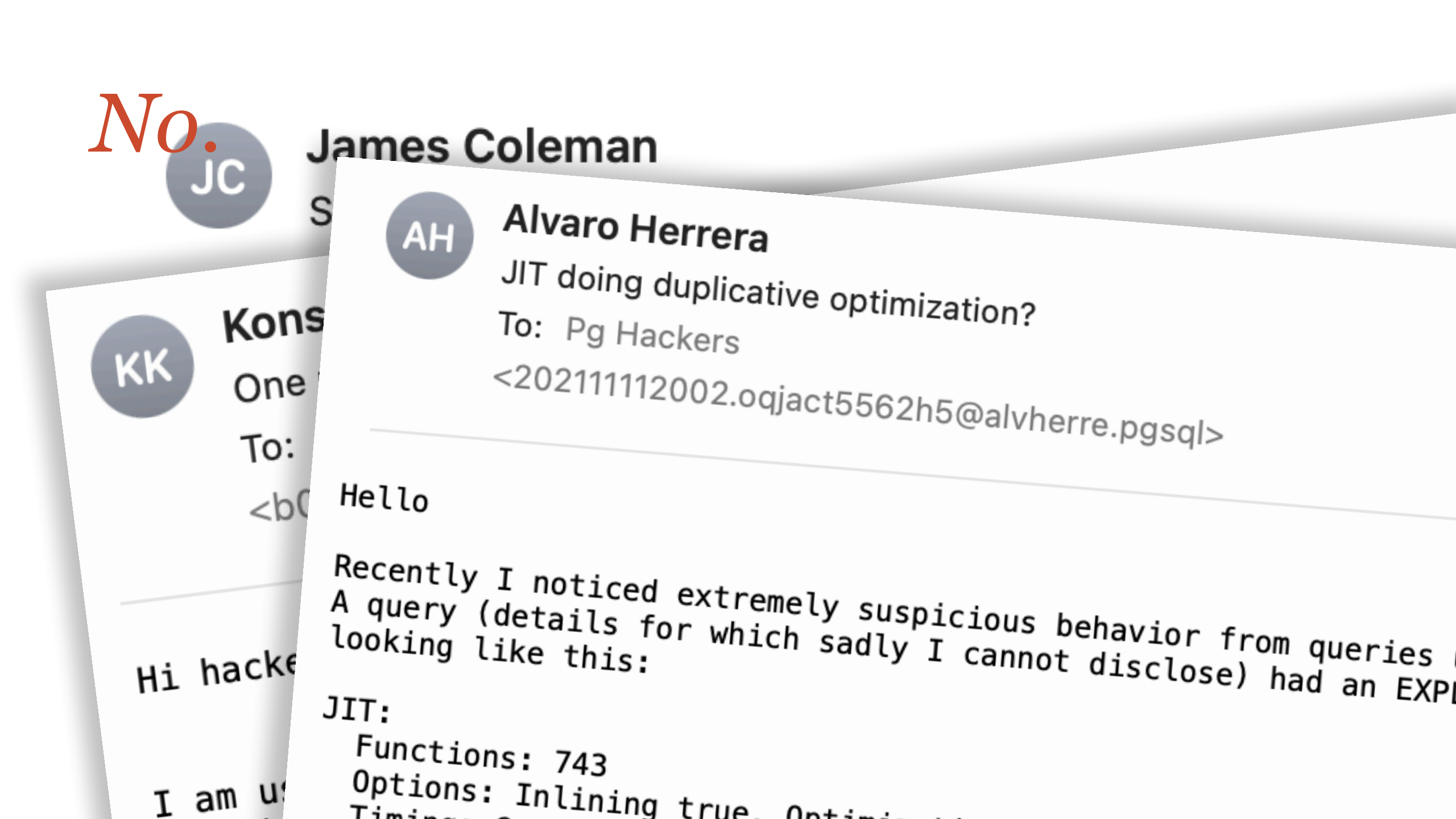
<b0900d21-594a-57a7-d345-9f8e3987e068@garret.ru>

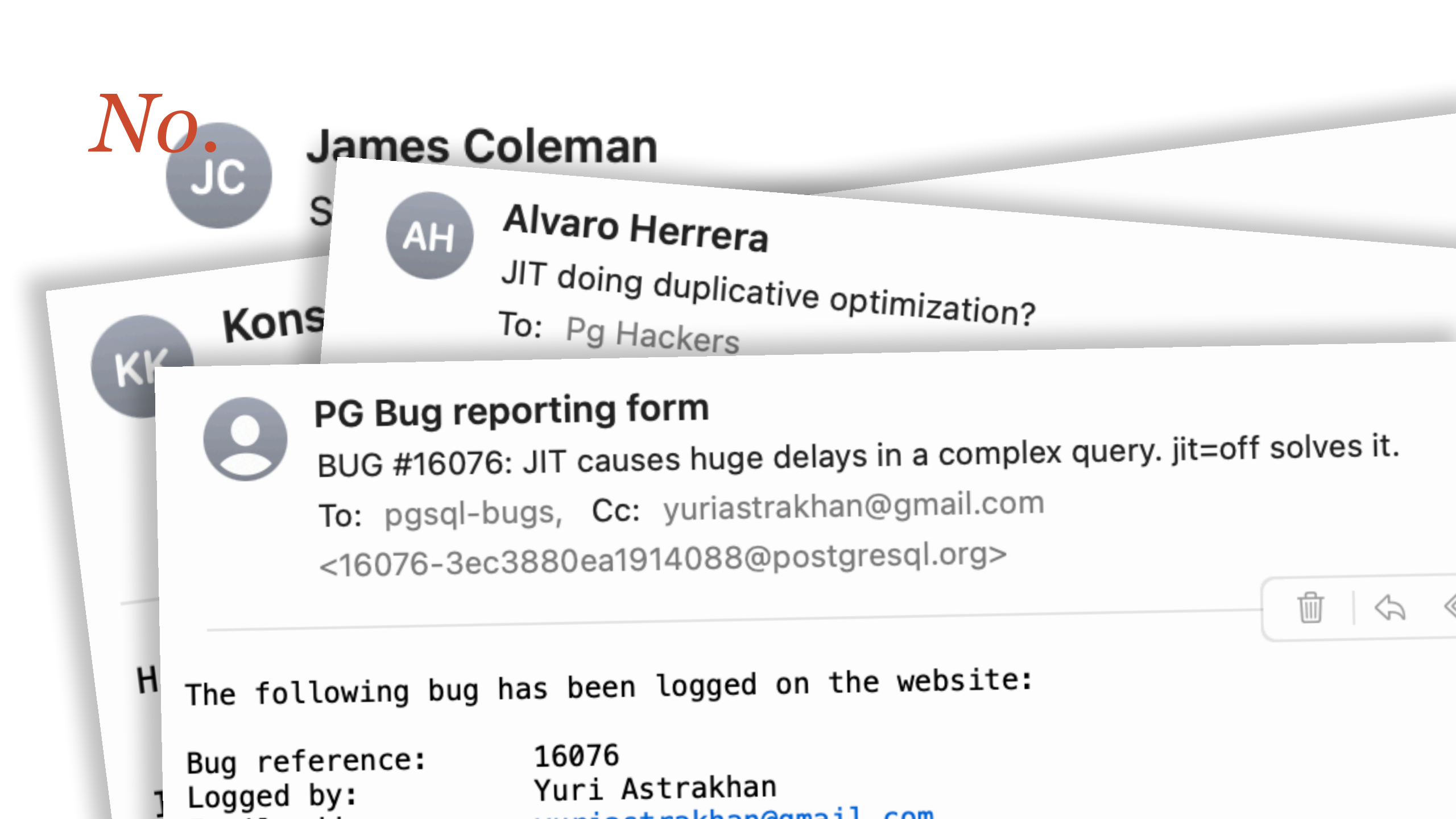
Hi hackers,

I am using pg_embedding extension for Postgres which implement

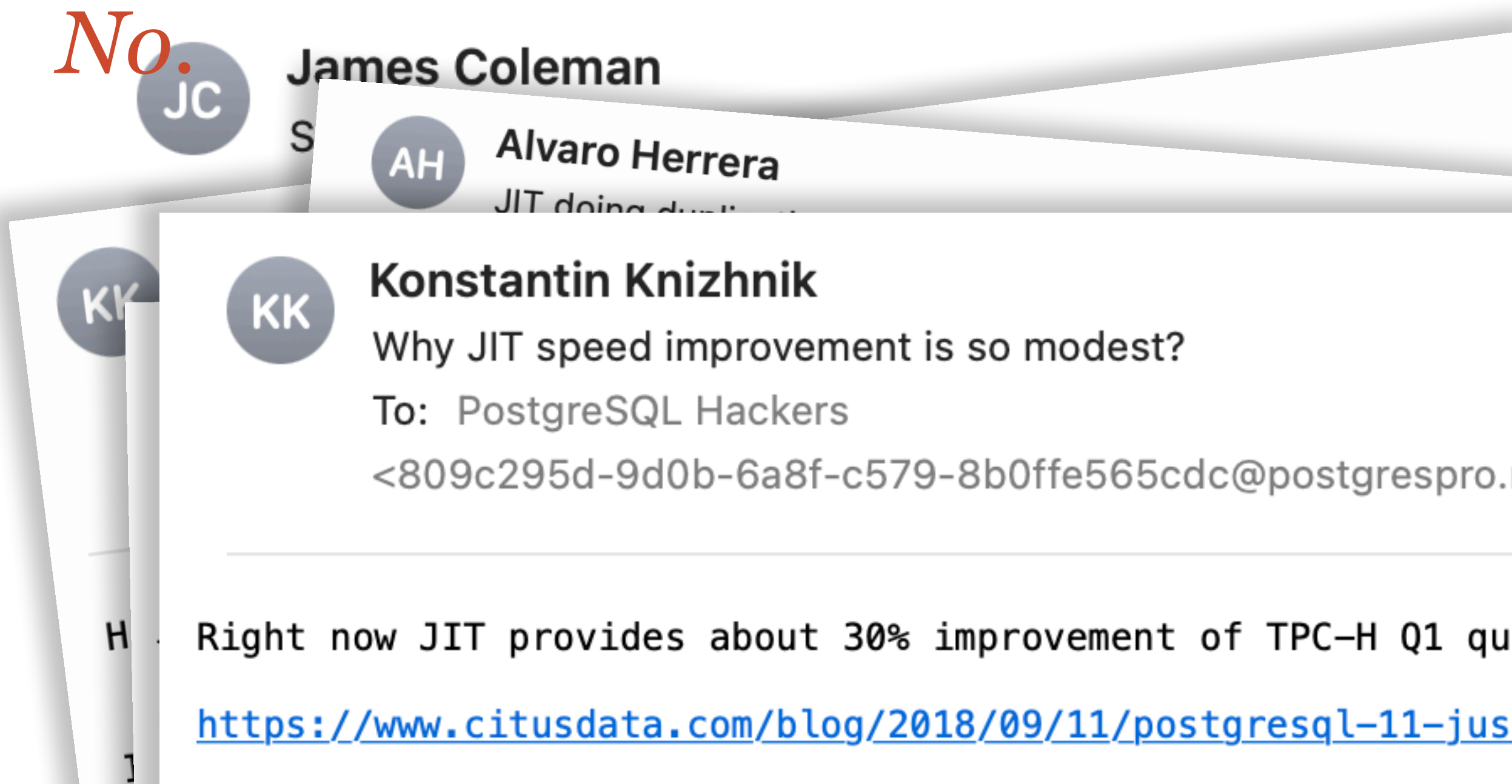
query looks something like this:

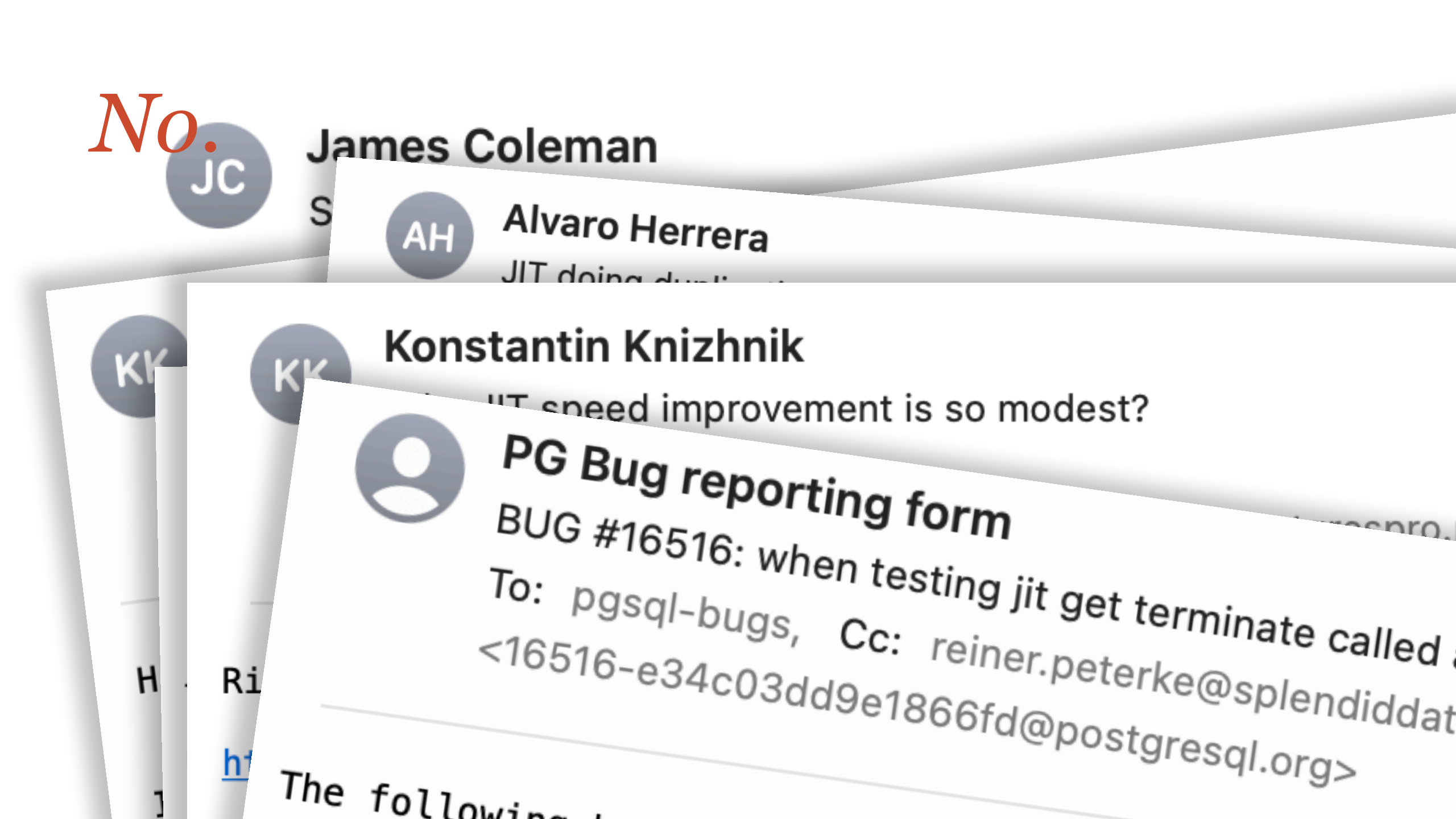
ay_cosine_dist(embedding, ai <=> ARRAY[0.024


No.
JC
James Coleman
KK
Kons
One
To:
<bC
Hi hacke
I am us
AH
Alvaro Herrera
JIT doing duplicative optimization?
To: Pg Hackers
<202111112002.oqjact5562h5@alvherre.pgsql>
Hello
Recently I noticed extremely suspicious behavior from queries
A query (details for which sadly I cannot disclose) had an EXPL
looking like this:
JIT:
Functions: 743
Options: Inlining true, Optimi


No.
JC
James Coleman
AH
Alvaro Herrera
JIT doing duplicative optimization?
To: Pg Hackers
KK
Kons
PG Bug reporting form
BUG #16076: JIT causes huge delays in a complex query. jit=off solves it.
To: pgsql-bugs, Cc: yuriastrakhan@gmail.com
<16076-3ec3880ea1914088@postgresql.org>
The following bug has been logged on the website:
Bug reference: 16076
Logged by: Yuri Astrakhan

No.


JC
James Coleman
AH
Alvaro Herrera
KK
Konstantin Knizhnik
Why JIT speed improvement is so modest?
To: PostgreSQL Hackers
<809c295d-9d0b-6a8f-c579-8b0ffe565cdc@postgrespro.
Right now JIT provides about 30% improvement of TPC-H Q1 qu
https://www.citusdata.com/blog/2018/09/11/postgresql-11-jus


No.
JC
James Coleman
AH
Alvaro Herrera
KK
Konstantin Knizhnik
speed improvement is so modest?
PG Bug reporting form
BUG #16516: when testing jit get terminate called
To: pgsql-bugs, Cc: reiner.peterke@splendiddat
<16516-e34c03dd9e1866fd@postgresql.org>
The

People are unlikely to notice/report cases when the JIT (including costing) works fine. But they sure are annoyed when it makes the wrong choice.

-- *Tomas Vondra*

BIKESHED

ADVISORY

WARNING

Whats wrong?

*I*t's crashing my database!

*I*t's making my query slow!

*S*eriously?!.. NOT NOW!

Whats wrong?

Out of memory due to leaks

It's making my query slow!

Seriously?!.. NOT NOW!

Whats wrong?

Out of memory due to leaks

Lack of caching of expressions

Seriously?!.. NOT NOW!

Whats wrong?

Out of memory due to leaks

Lack of caching of expressions

Coarse cost model

Don't crash

Out of memory

*M*emory "leak" in LLVM with inlining

*S*ome queries greatly affected, others not at all

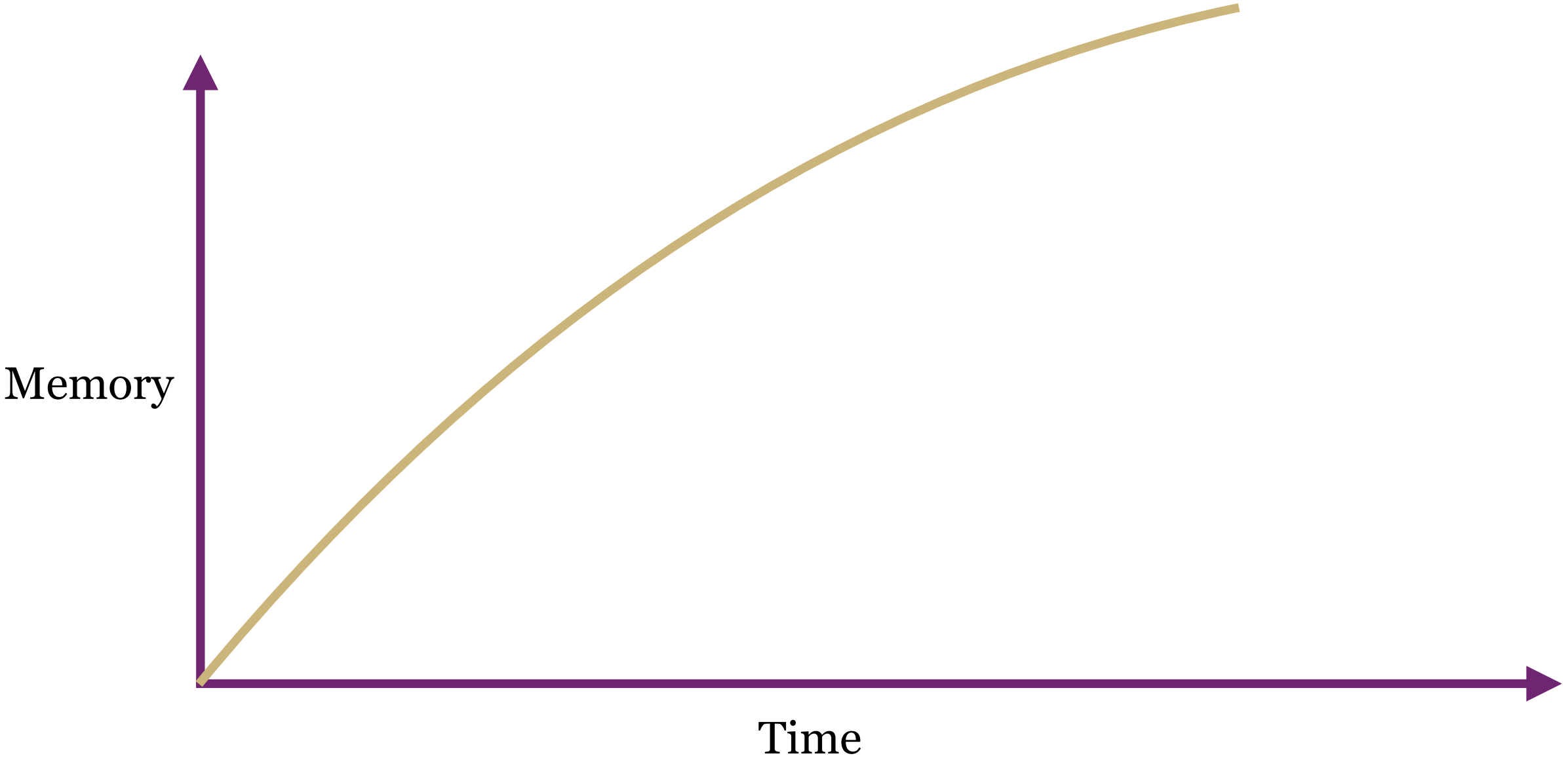
*F*ix is to forcibly evict from LLVM periodically

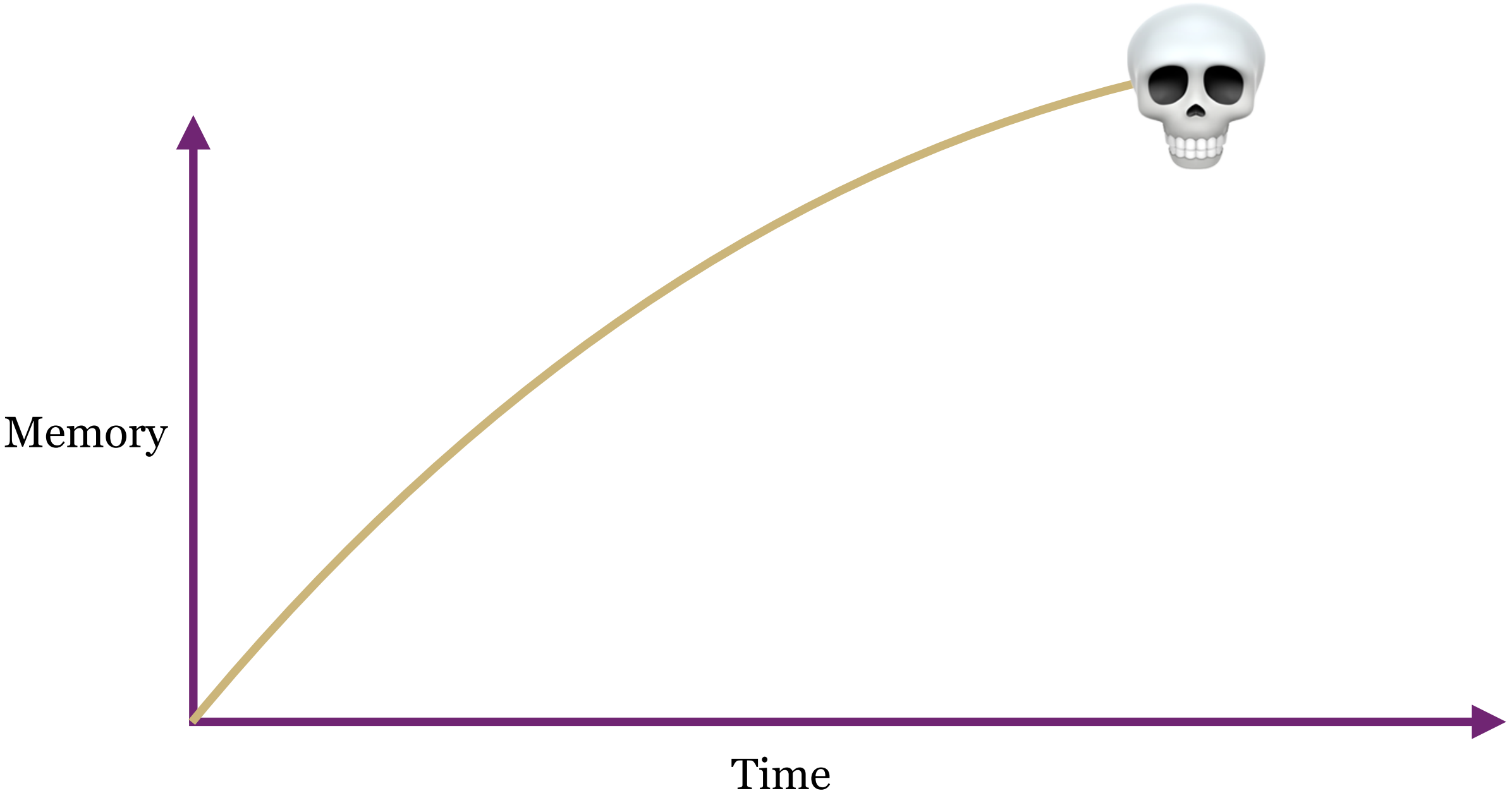
author Daniel Gustafsson <dgustafsson@postgresql.org>
Wed, 27 Sep 2023 11:02:21 +0000 (13:02 +0200)
commit 9dce22033d5d2813e9f8e7d595f57ee5a38b3f8e

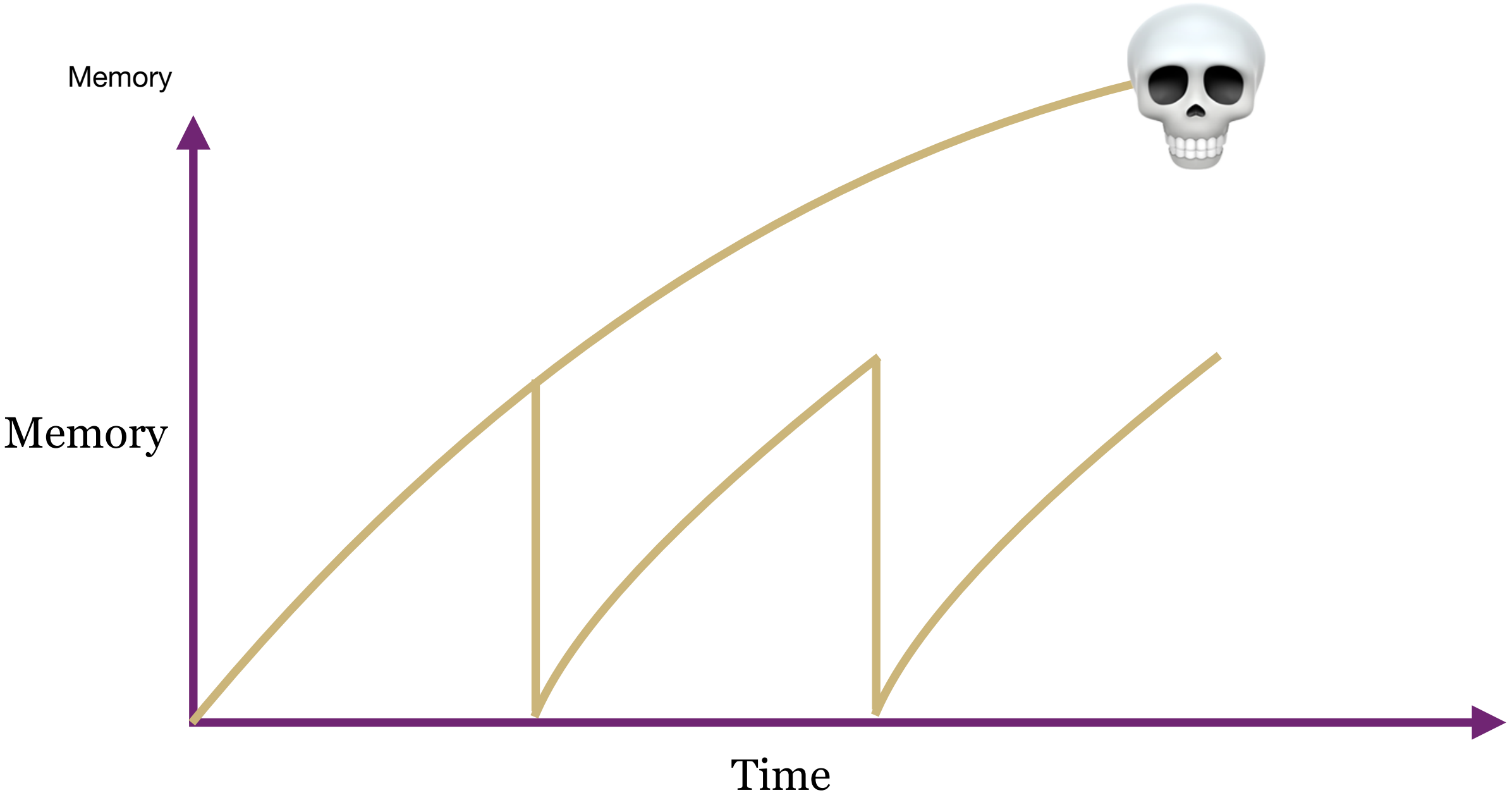
llvmjit: Use explicit LLVMContextRef for inlining

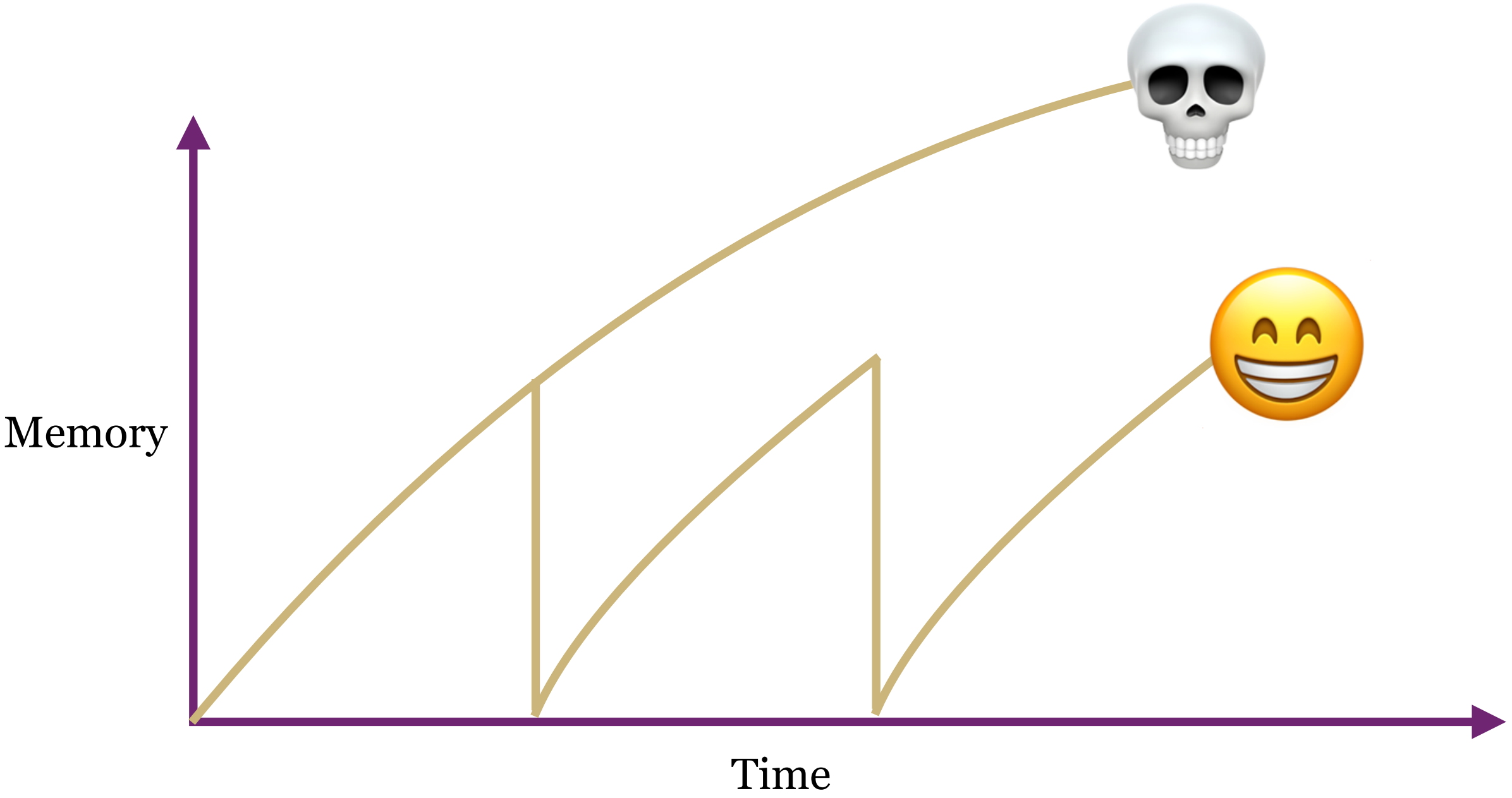
When performing inlining LLVM unfortunately "leaks" types (the types survive and are usable, but a new round of inlining will recreate new structurally equivalent types). This accumulation will over time amount to a memory leak which for some queries can be large enough to trigger the OOM process killer.

To avoid accumulation of types, all IR related data is stored in an LLVMContextRef which is dropped and recreated in order to release all types. Dropping and recreating incurs overhead, so it will be done only after 100 queries. This is a heuristic which might be revisited, but until we can get the size of the context from LLVM we are flying a bit blind.


Memory
Time


Memory
Time


Memory
Memory
Time


Memory
Time

Future work

*B*ackpatching to stable releases

*I*nterrogating LLVM for size and only recreate when over threshold

*L*imit to queries with inlining

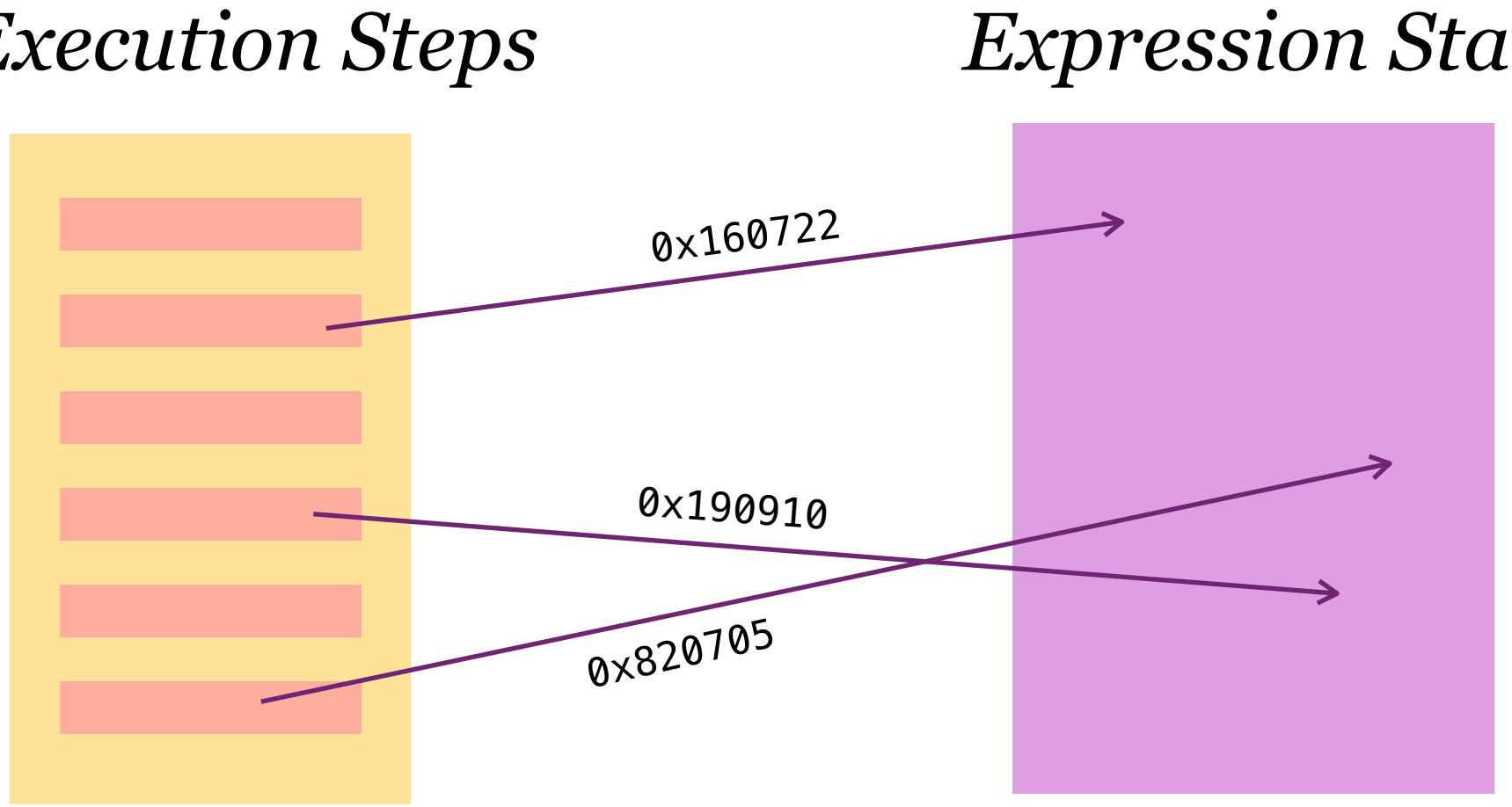
Be fast(er)

Lack of caching

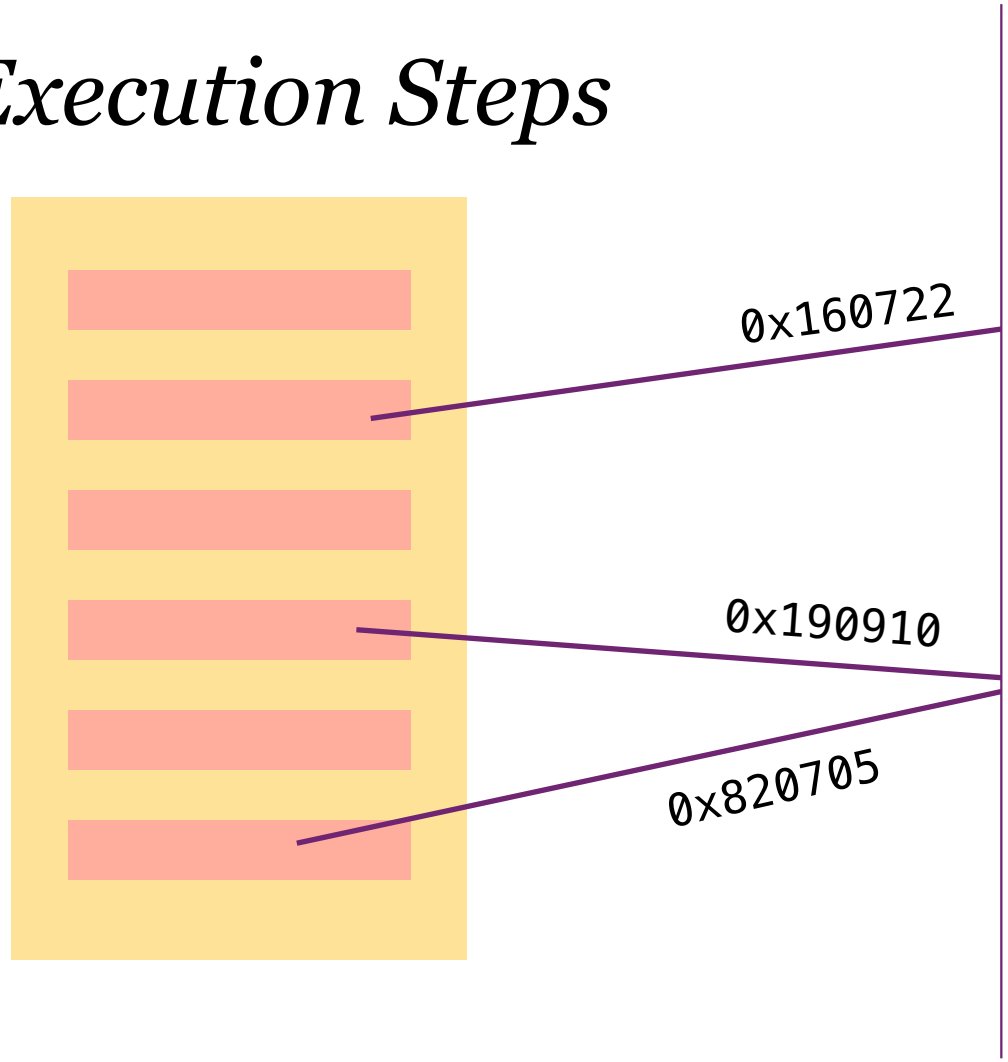
*H*ard problem to solve

*R*equires rewriting large parts of the executor

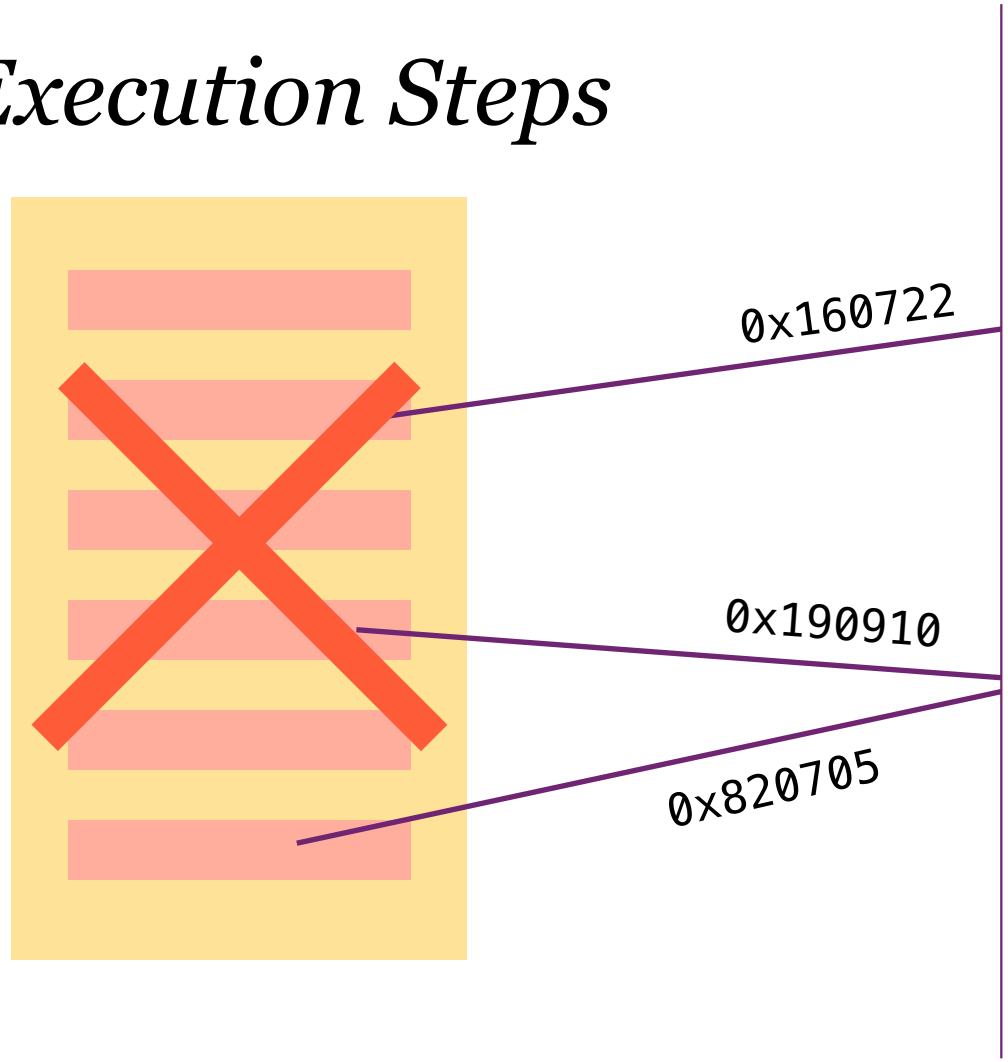
*C*urrently uses pointers to per-query memory allocations

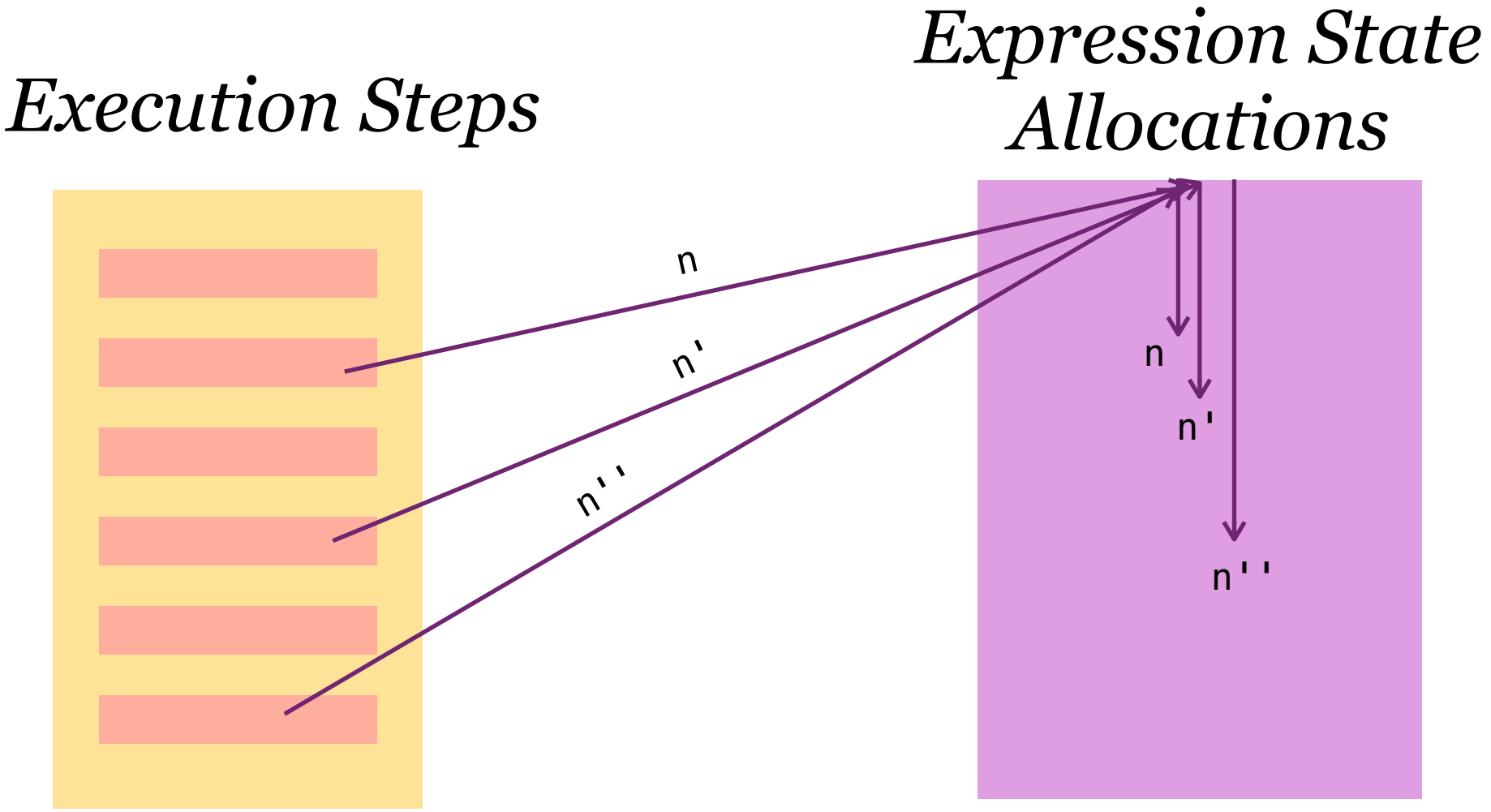

Execution Steps
Expression State
0x160722
0x190910
0x820705

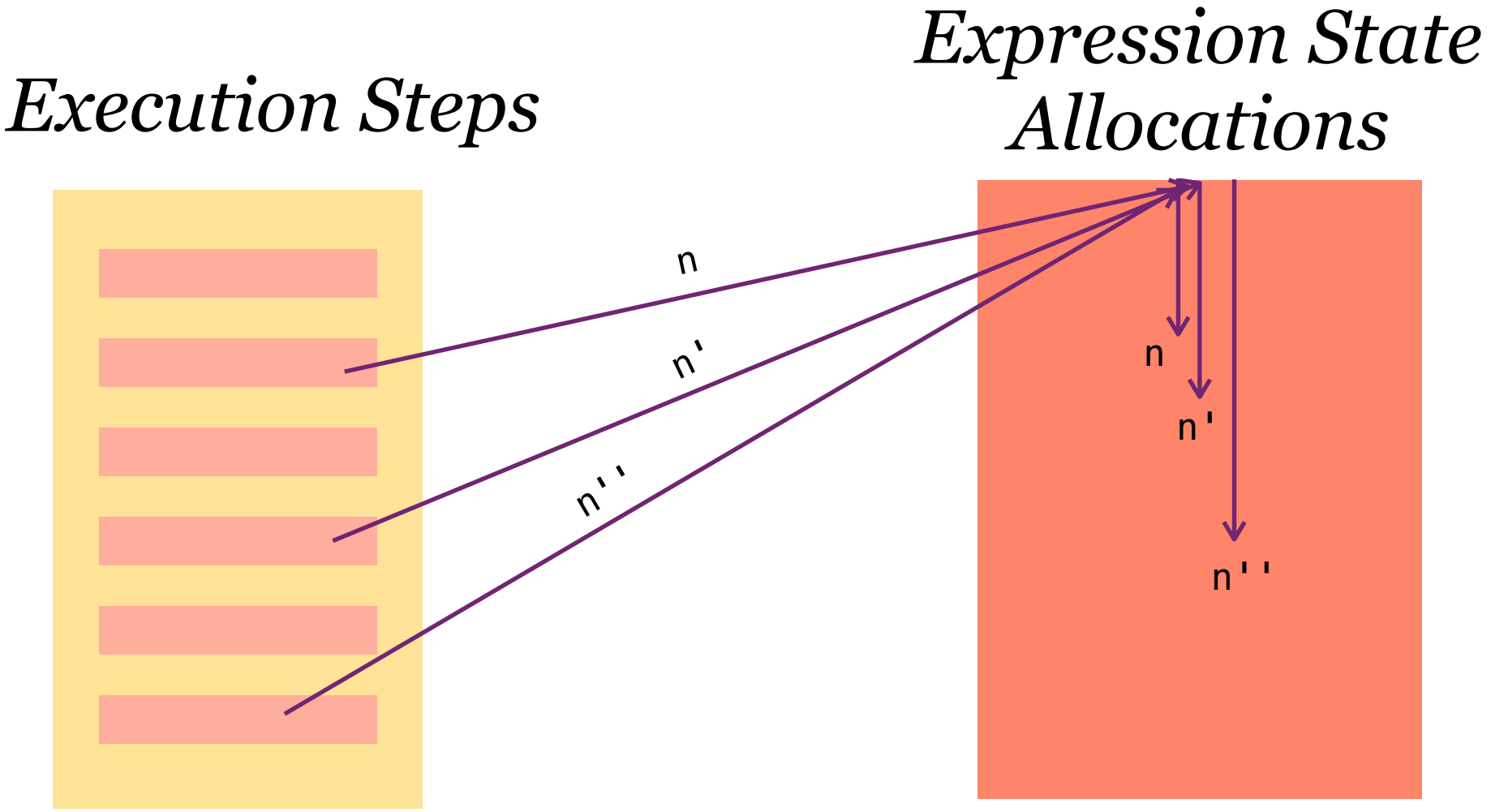
Execution Steps

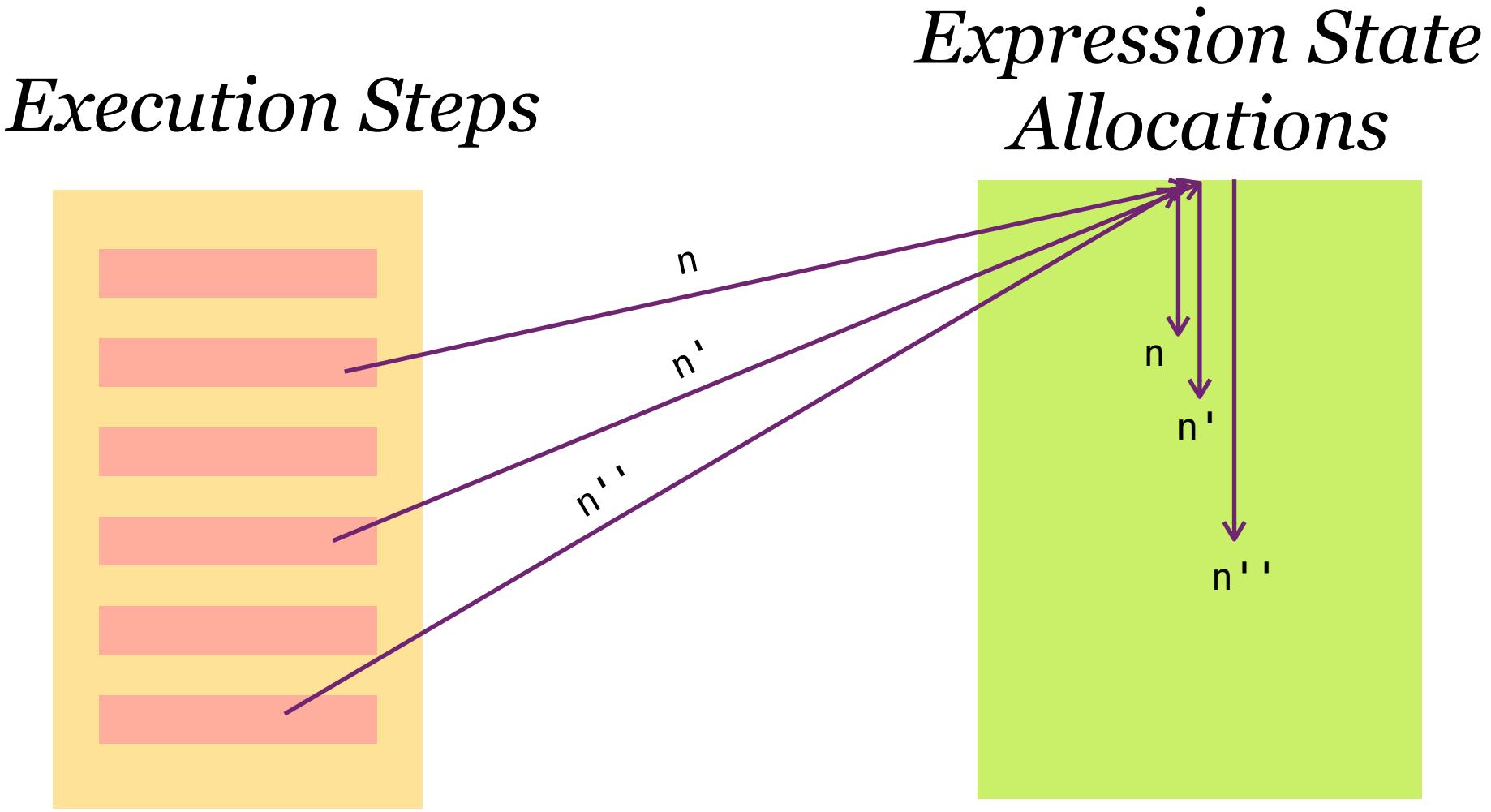


Execution Steps




Execution Steps
Expression State Allocations
n
n'
n''
n
n'
n''


Execution Steps
Expression State Allocations
n
n'
n''
n
n'
n''


Execution Steps
Expression State Allocations
n
n'
n''
n
n'
n''

Lack of caching

*P*atchset for relative pointers on -hackers soon, plan is to get a version of that into HEAD during v*17* cycle

```
$ git diff origin/master --stat | tail -1
89 files changed, 5929 insertions(+), 3732 deletions(-)
```

*C*aching is the next step, hopefully *something* during v*17*

Predict the future

.. JIT costing is disconnected from reality. It's a bit like trying to tune your radio with the volume control.

-- David Rowley 24/6 2023

Costing

*C*ost doesn't reflect potential JIT value

*E*conomy of scale, emitting code is expensive

*E*xpensive to determine equality of expressions

Costing

*M*uch of this is a general problem, which is excacerbated by JIT

Costing

*T*here are patches on -hackers but none which are actively worked on

*N*eeds to be revisited

*B*eing able to cache will greatly influence costing

Whats wrong?

Out of memory due to leaks

Lack of caching of expressions

Coarse cost model

Whats wrong?

*F*ixed

*L*ack of caching of expressions

*C*oarse cost model

Whats wrong?

Fixed *(there might be more crashes of course)*

Lack of caching of expressions

Coarse cost model

Whats wrong?

*F*ixed

*I*n progress

*C*oarse cost model

Whats wrong?

*F*ixed

*I*n progress

*N*ext on the list

TL;DR

*M*ature code

*C*an make many queries much faster

*N*ot perfect.. but we're working on it!

*H*elp us with testing and report bugs

Microsoft

Thank you!

daniel@yesql.se