





Rage DB

Building a Graph Database in Anger

Sixteenth TUC Meeting - June 24, 2023

Lose seconds off your graph queries with this one weird trick¹

Doctors² hate him!
Developer builds a property graph server from home.

-  Faster than a speeding bullet train
-  Connect from anywhere via HTTP
-  Use a programming language to query
-  Apache license, version 2.0

 Try it right now using Docker!



RAGE DB

(1) Put everything in memory (2) Computer Science PhDs.

🔗 main ▾ ragedb / LICENSE.txt

 ragedb/ragedb is licensed under the **Apache License 2.0**

A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- ✓ Commercial use
- ✓ Modification
- ✓ Distribution
- ✓ Patent use
- ✓ Private use

Limitations

- ✗ Trademark use
- ✗ Liability
- ✗ Warranty

This is not legal advice. [Learn more about repository licenses.](#)

 **maxdemarzi** first commit

Latest commit

👤 1 contributor

177 lines (150 sloc) | 9.93 KB

```
1
2           Apache License
3           Version 2.0, January 2004
4           http://www.apache.org/licenses/
```

Blog Posts

maxdemarzi.com

APR 12 2022

LEAVE A COMMENT

RAGE
EDIT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 25: DATES IN C++ AND FASTER IMPORTS



Back in [February](#), we added the ability to load a CSV file and alter the contents while importing it. We also added Date support to [RageDB](#) using a Lua library. This was a masterful job of copy and paste and got us lots of functionality very quickly. When we timed the import for LDBC SNB SF10 it came in at 28 minutes. Which wasn't bad, but wasn't great. Let's try to speed that up today.

JUL 14 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 4: CREATING AND RETRIEVING NODES



When I was first introduced to graph databases I had a hard time [understanding](#) them. When a node gets created, where does it go? There are no tables in graph databases, so I was missing that logical arrangement of rows and columns. It made me a little paranoid. Like what if I lost them? I built some projects storing all the data in Postgres first, so if anything happened to the graph I could rebuild it. That [main objective security feature](#) [comment](#) was the biggest boost. We're taking another walk through a door and arranging all nodes and properties of each node in a set of vectors, but we get straight to work.

Continue reading...
Tags: [graph-databases](#) [graph-databases](#) [software-technology](#)

JUL 13 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 3: NODE TYPES



[Jim Morrison](#) died in 1971. I wasn't even alive then. I don't know about [The Doors](#) until the [Velvet Hammer](#) film from the 90s. I was still too young to fully understand them but I became a fan of the music nonetheless. In the last few months I've learned more about doors. I learned about this concept of [gate](#) [and](#) [the](#) [way](#) [doors](#) [work](#). The idea being that one way doors can only be [opened](#) [through](#) [one](#) [way](#). But if you don't love the decision [and](#) [walk](#) [through](#) [the](#) [door](#) [back](#) [and](#) [back](#) [to](#) [where](#) [you](#) [started](#).

Continue reading...
Tags: [graph-databases](#) [graph-databases](#) [node](#) [software-technology](#)

JUL 12 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 2: SHARDS ARE OK!



When I was a new Java developer I would sometimes wake up in the middle of the night hyperventilating and covered in sweat. Usually from a nightmare about [Java](#) [and](#) [fighting](#) [with](#) [pom.xml](#). We dream of Software, but does Software dream? I don't know, but I hope when Maven goes to [download](#) [the](#) [pom.xml](#) [and](#) [wakes](#) [up](#) [answering](#) [thinking](#) [about](#) [Cache](#) [and](#) [Checksum](#)... I have it!

Rather I should say "I don't" because I used to look for help on [stackoverflow](#) [and](#) [on](#) [my](#) [own](#) [machine](#) [for](#) [C++](#) [projects](#) [by](#) [Jason](#) [Tanner](#) which made the nightmare stop. We start our new project by [deciding](#) [whether](#) [to](#) [use](#) [a](#) [library](#) [and](#) [removing](#) [a](#) [few](#) [C++](#) [related](#) [things](#) [we](#) [won't](#) [be](#) [using](#). Which the [idea](#) [for](#) [all](#) [the](#) [details](#), I only understood half of it, but it was enough. There I learned about [C++](#).

Continue reading...
Tags: [graph-databases](#) [graph-databases](#) [node](#) [software-technology](#)

JUL 08 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS



I could not decide if I wanted a Phoenix or [Cerberus](#) as a mascot for this new project, so I went with both. The image you see above is what I ended up with. It's absolutely outrageous. It's perfectly what I want. That was going to be the name by the way "Outrageous DB" but it was kind of long, so I went with "Rage DB" instead. Right about now you may be thinking, wait, what are we doing, what is this new project? What is going on?

Continue reading...
Tags: [graph-databases](#) [graph-databases](#) [node](#) [software-technology](#)

JUL 29 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 8: QUERIES WITH LUA



[Amie Brundin](#) writes a monster of a blog post the other day [about SQL](#). This is my favorite part.

To take an example close to my heart: [Differential database](#) is a database engine that includes support for automatic parallel execution, horizontal scaling and incrementally maintained views. It table [caching](#) [and](#) [was](#) [mostly](#) [written](#) [by](#) [a](#) [single](#) [person](#). [Materialized](#) [views](#) [support](#) [for](#) [SQL](#) [and](#) [various](#) [data](#) [sources](#). To date, that has taken [~100kloc](#) (not including dependencies) and I estimate ~10-20 engineer-years. Just covering SQL to [the](#) [logical](#) [plan](#) [takes](#) [~20kloc](#), more than ten times the entirety of differential database.

Similarly, [right](#) [looks](#) [to](#) [have](#) [~20kloc](#) [and](#) [duckdb](#) [~10kloc](#). The count for duckdb doesn't even include the parser that they [recently](#) [borrowed](#) [from](#) [postgres](#), which at [~10kloc](#) is much larger than the entire [public](#) [condition](#) [for](#) [lua](#).

Amie Brundin – [Against SQL](#)

Continue reading...
Tags: [graph-databases](#) [graph-databases](#) [software-technology](#)

JUL 27 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 7: PERFORMANCE



I don't know why, but I need things to be fast. Nothing drives me up the wall more than waiting a long time for some database query to finish. It's some kind of disease that you. So today we're going to do site performance checking to see where RageDB is at. So far at we can do it across nodes and relationships, but fast enough to get a signal.

Continue reading...
Tags: [graph-databases](#) [graph-databases](#) [software-technology](#)

JUL 26 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 6: RELATIONSHIPS



Good relationships are hard. They don't just happen. They take time, patience and about two thousand lines of code to work together. It's not convenient I have it right, maybe one of you out there has a better design we can implement. In the [original design](#) I was storing the full relationships complete with identifying code bits, properties and type information. This time relationships are only temporarily created when requested, and we're just going to store their pieces in different nodes. Let's dive in to the code.

Continue reading...
Tags: [graph-databases](#) [graph-databases](#) [software-technology](#)

JUL 19 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 5: PROPERTIES



Whenever you [start](#) [on](#) [the](#) [game](#) [Minecraft](#), you start by getting properties, changing them and making sure anything that uses them plays a key sum. That's also true of graph databases. Finding a property, changing a property, filtering on a property and sometimes even deleting properties can be really expensive. Part of the issue is that it was decided at some point that it would be great if any nodes of the graph could have different properties and it was decided that the same property name, the [same](#) [property](#) [name](#) [has](#) [to](#) [be](#) [the](#) [same](#).

AUG 24 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 12: FINDING THINGS



Knowing yourself is the beginning of all wisdom
Aristotle

At some point in your life you will look at the mirror and not recognize the person staring back at you. It's not only the way you look, but the choices you make and the results of those choices that will seem puzzling. You may decide right then and there to embark in the most important quest of your life. Finding yourself.

Nodes and Relationships don't get the luxury of having mirrors, but we still need to find them every once in a while. So far all we are able to do is get a node by a label and key or use the key to get a node or relationship. This is great and covers a lot of use cases, but sometimes the primary id is not what we are looking for. We may be looking for all nodes of a type that share a property value equal to something, or greater than something else. How can we go about finding the answers in these types of queries in our graph?

Continue reading...
Tags: [graph-databases](#) [graph-databases](#) [node](#) [software-technology](#)

AUG 16 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 11: TESTING



Heidi has been testing software since we met in college back in 1989. Today she is the head of QA at SAP Market Intelligence managing hundreds of people. I don't know how she does it but she is the kind of person that breaks every piece of software she touches, even when she doesn't want to... like the in-flight entertainment system on a long flight to France. About 10 years ago, I made a terrible mistake and asked her to test a web application I had written. She ripped it apart in minutes. I felt like the worst developer in the world, and I probably won't be all. I never wanted to test that way again, so I got better at writing tests. How I've only ranked second now, right after that dev who doesn't test at all.

Continue reading...
Tags: [graph-databases](#) [graph-databases](#) [node](#) [software-technology](#)

AUG 09 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 10: NULLS



We decided that our Nodes and Relationships will have Schema in our database. If we create a User Node Type and set an "age" property type as an Integer, then all User nodes will have that property as an Integer. This also seems simple enough, but what happens if we truly don't know the value of a property for a node? The user hasn't had an age yet, or for one of many perfectly valid reasons do not know what their age is? Or what if that age property that was once set is deleted? What do we do now?

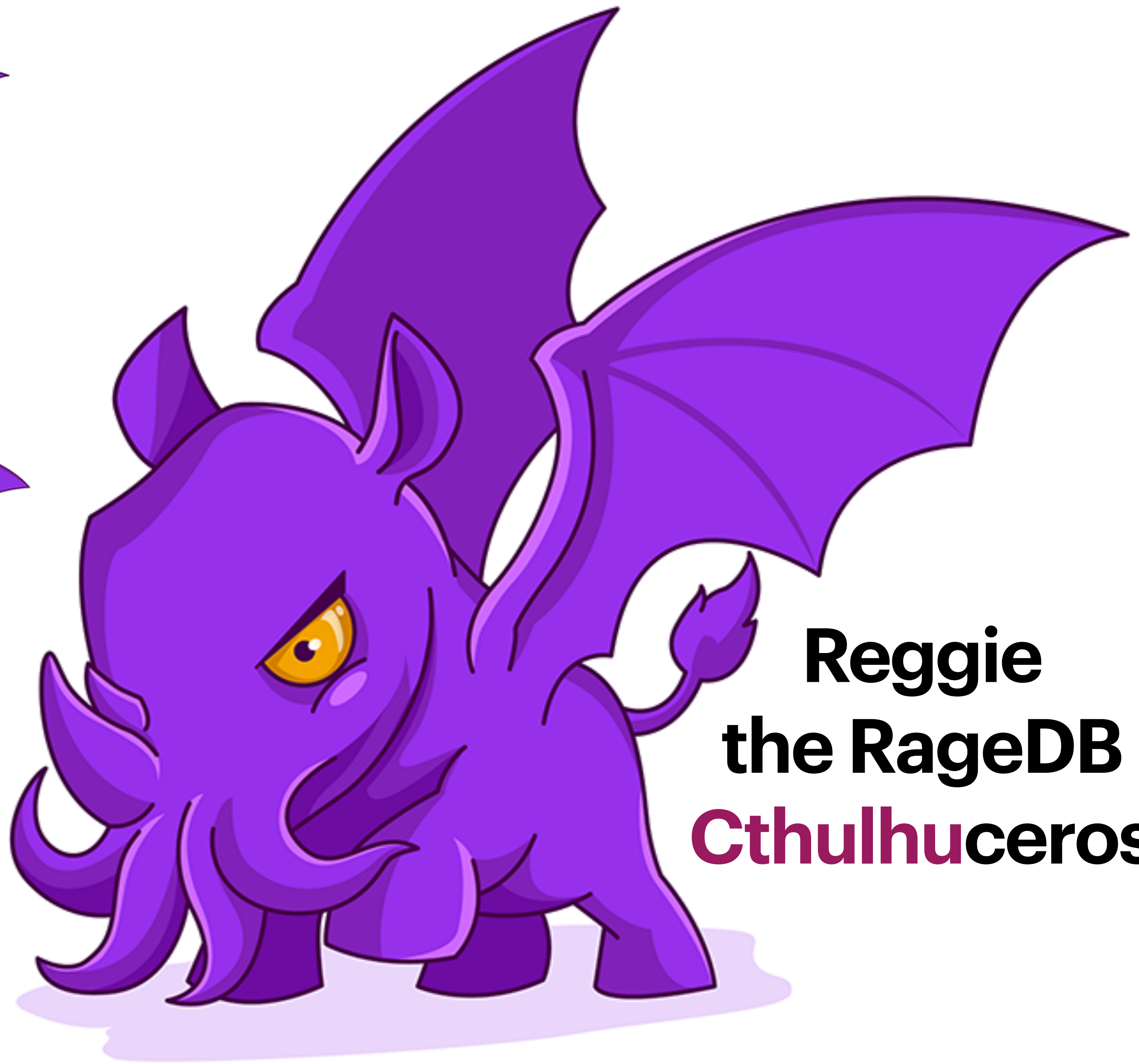
In [part 4](#) of this series we talked about how we store properties, but not really how we delete them. In my first design I had "properties" as a list indicating the data in its longer form to take. So for things it would be an empty string, for integers it was the integer value followed, etc., but what do we do about Boolean? It's either true or false, it doesn't lend itself to a boolean style value... and for the ones that do, does it even make sense?

Continue reading...
Tags: [graph-databases](#) [graph-databases](#) [node](#) [software-technology](#)

AUG 03 2021
LEAVE A COMMENT

LET'S BUILD SOMETHING OUTRAGEOUS – PART 9: DOCKER





Reggie
the RageDB
Cthulhuceros



Please allow me to
INTRODUCE

MYSELF

I'M Here

**because Neo4j
never went Public**

Max De Marzi

@maxdemarzi

maxdemarzi.com

GitHub.com/maxdemarzi



MAX DE MARZI

Graphs, Graphs, and nothing but the Graphs

VIDEOS SERVICES CONTACT ABOUT HOME

JAN 04 2012
30 COMMENTS

NEOGRAPHY

EDIT

GETTING STARTED WITH RUBY AND NEO4J

Getting started with Ruby and Neo4j is very easy.
Follow these steps and you'll be up and running in no time.

First we install the neography gem:

Using Bundler:

```
1 | echo "source 'http://rubygems.org'"  
2 | gem 'neography' " > Gemfile  
3 | bundle install
```

Without Bundler:

```
1 | gem install neography
```

Then we'll add our tasks to a Rakefile, download Neo4j and start it:

```
1 | echo "require 'neography/tasks'" > Rakefile  
2 | rake neo4j:install  
3 | rake neo4j:start
```

Eleven Years

In the Graph game





NOT an Academic



Work the Field



A black and white photograph of a measuring tape. The tape is coiled and laid out on a light-colored surface. The numbers on the tape are clearly visible, ranging from 82 to 113. The word "Scale" is written in a large, bold, black font across the center of the image, partially overlapping the tape. The background is a plain, light-colored surface.

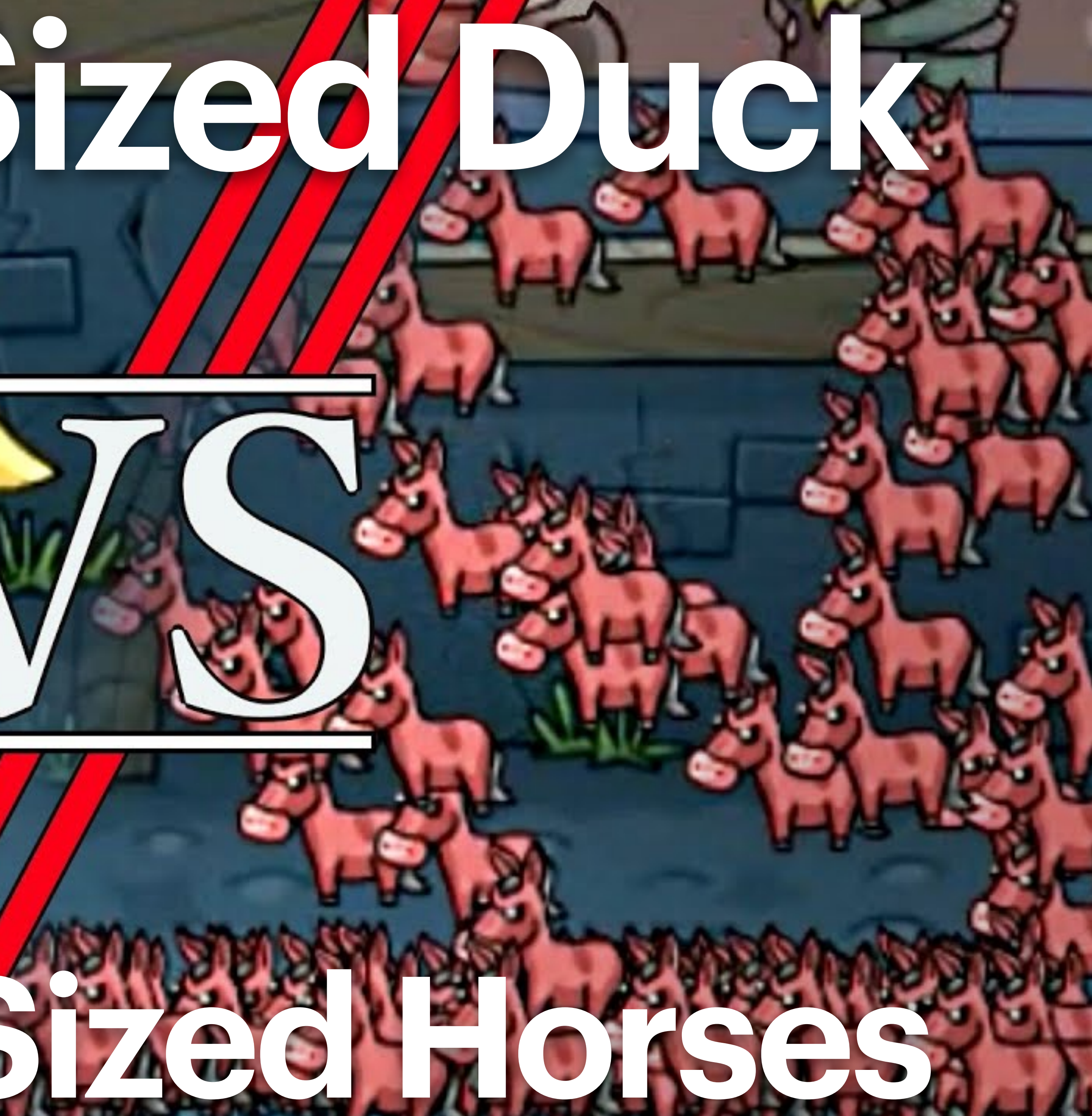
Scale

1 Horse Sized Duck



V S

100 Duck Sized Horses



intel.

XEON®

Intel® Xeon® Processor codenamed Sierra Forest

First Xeon processor with Efficient-core (E-Core)

Sampling today, shipping 1st half of 2024

Excellent silicon health

Silicon power-on;
Operating systems booted
in <18 hours

Lead vehicle for Intel 3

144 processor cores

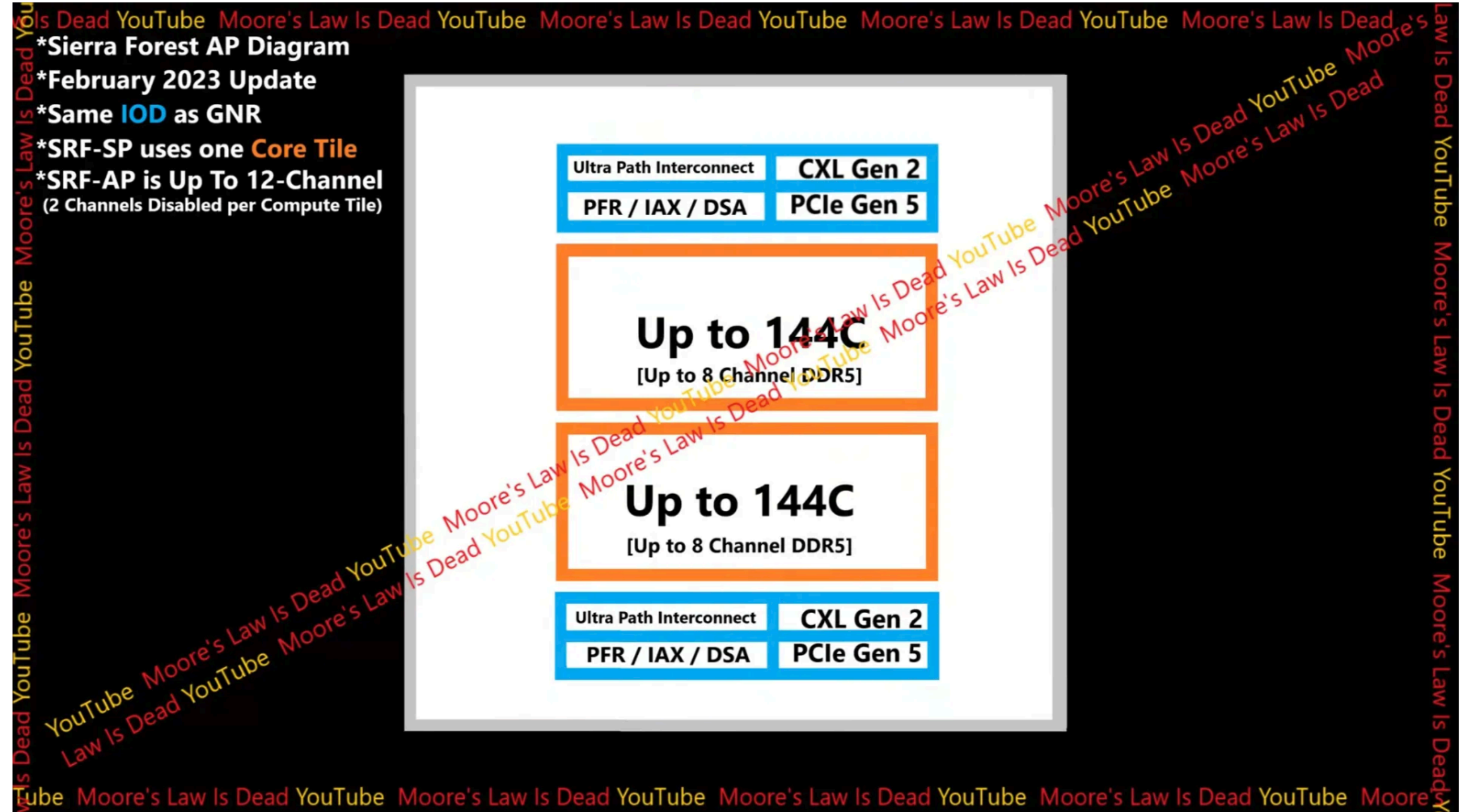
144

New class of Xeon

Built for cloud-optimized
workloads

Below is the diagram for Sierra Forest-AP, featuring 2 tiles for a total of 288 cores. The customer in question wants the same CPU configuration, with just 1 tile disabled (144-cores). And that 144-core CPU is probably going to be marketed by Intel as Sierra Forest-SP.

288 Cores

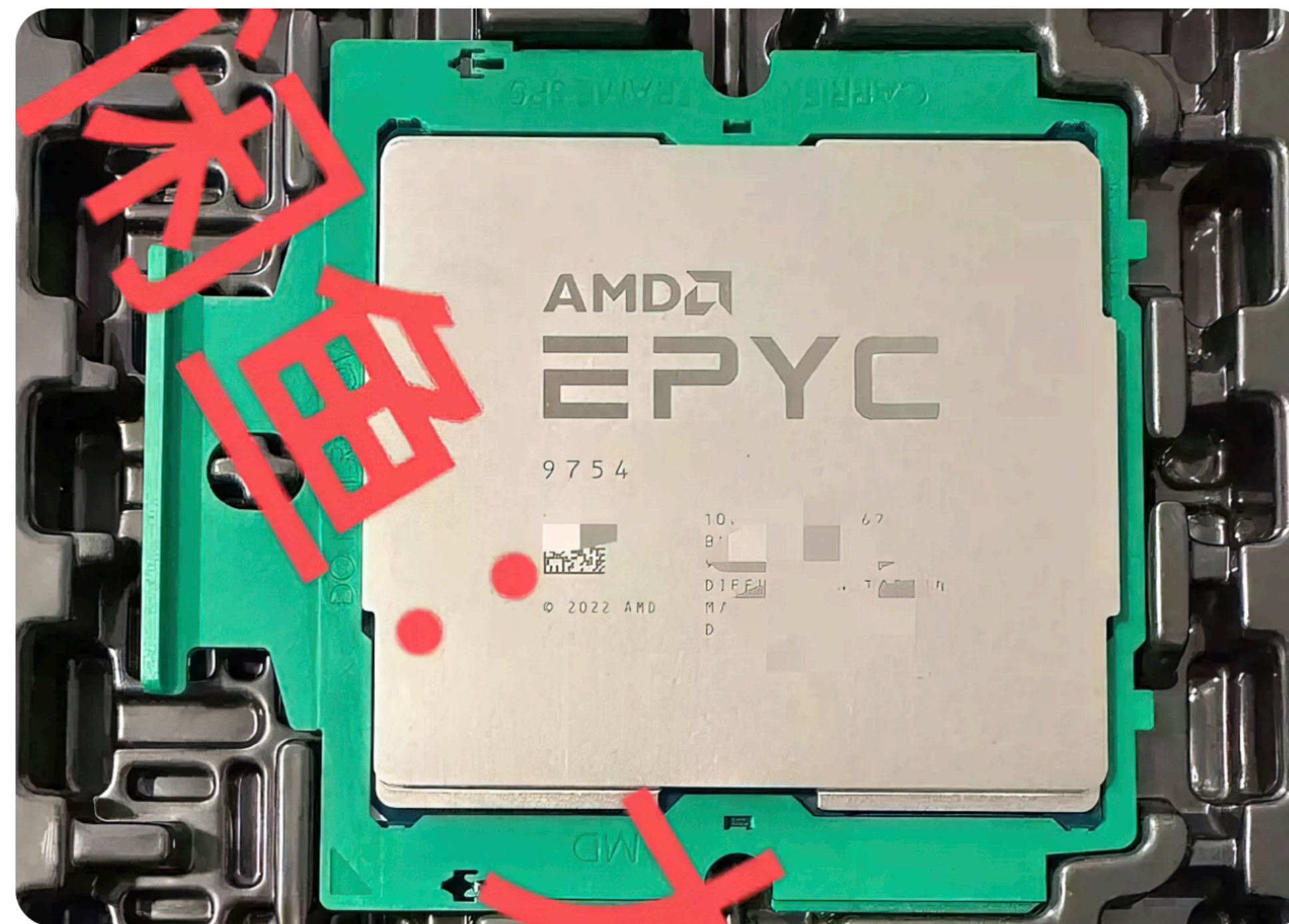


Hardware

AMD EPYC 9754 "Bergamo" CPU With 128 Zen 4C Cores Being Sold For \$5500 US, Less Than Half Its Official Price

Hassan Mujtaba · Jun 20, 2023 11:11 AM EDT · Copy Shortlink

96



AMD recently introduced its brand new Bergamo CPU lineup which features the flagship EPYC 9754 chip with 128 Zen 4C Cores. This chip is now being sold at the Chinese 3rd party seller, Goofish.

AMD's Flagship Bergamo Chip, The EPYC 9754 128-Core CPU, Is Being Sold For \$5500 US

Trending Stories

Starfield Doesn't Have Land Vehicles or Fishing, Todd Howard Confirms

46 Active Readers

Final Fantasy XVI Resolution Drops as Low as 720P in Performance Mode, Uses AMD FSR 1

17 Active Readers

The Xbox Series X Is Considered by Microsoft Its Mid-Gen Refresh, as the Xbox Series S Is Their Standard Machine

17 Active Readers

Jim Keller's Tenstorrent Wants To Compete With NVIDIA's AI GPUs Using RISC-V Based AI CPUs

15 Active Readers

Intel 14th Gen Raptor Lake-S "Desktop" & Raptor Lake-HX "Laptop" CPU Refresh Confirmed

12 Active Readers

Popular Discussions

AMD CEO Teases ROCm Support Coming To Radeon Consumer GPUs Soon

1503 Comments

128 Cores For \$5500

Moar core blimey: 384-core AMD EPYC Venice server chip with Zen 6 architecture enters the rumor mill **384!!!**

A well-known leaker has made some extraordinary claims about the future of AMD's server chips, EPYC. Apparently, there is a generation coming within a few years, codenamed "Venice", that will offer well over 200 cores per part and may even include an astonishing 384-core SKU. The Zen 6-based chips will also apparently feature custom memory upgrades.

Daniel R Deakin, 05/01/2022   ... Zen AMD Leaks / Rumors

Workstation Business

The YouTube channel **Moore's Law Is Dead** has been a frequent source of leaks and rumors in the past but this latest one from show host Tom will definitely require a pinch of salt, especially as it discusses naming schemes and configurations that are potentially



AMD's EPYC lineup for 2024/25 will purportedly be codenamed "Venice" and will use Zen 6 microarchitecture. (Image source: AMD/Unsplash - edited)



6TB RAM



Distributed Systems

At @SIGMODConf #SIGMOD2023 #PODS2023 starts with excellent keynote by @jure of @StanfordEng

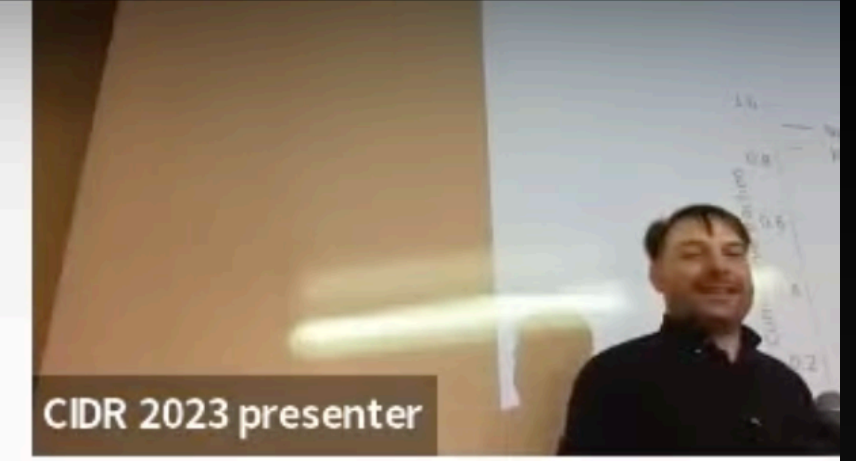
Databases as Graphs: Predictive Queries for Declarative Machine Learning



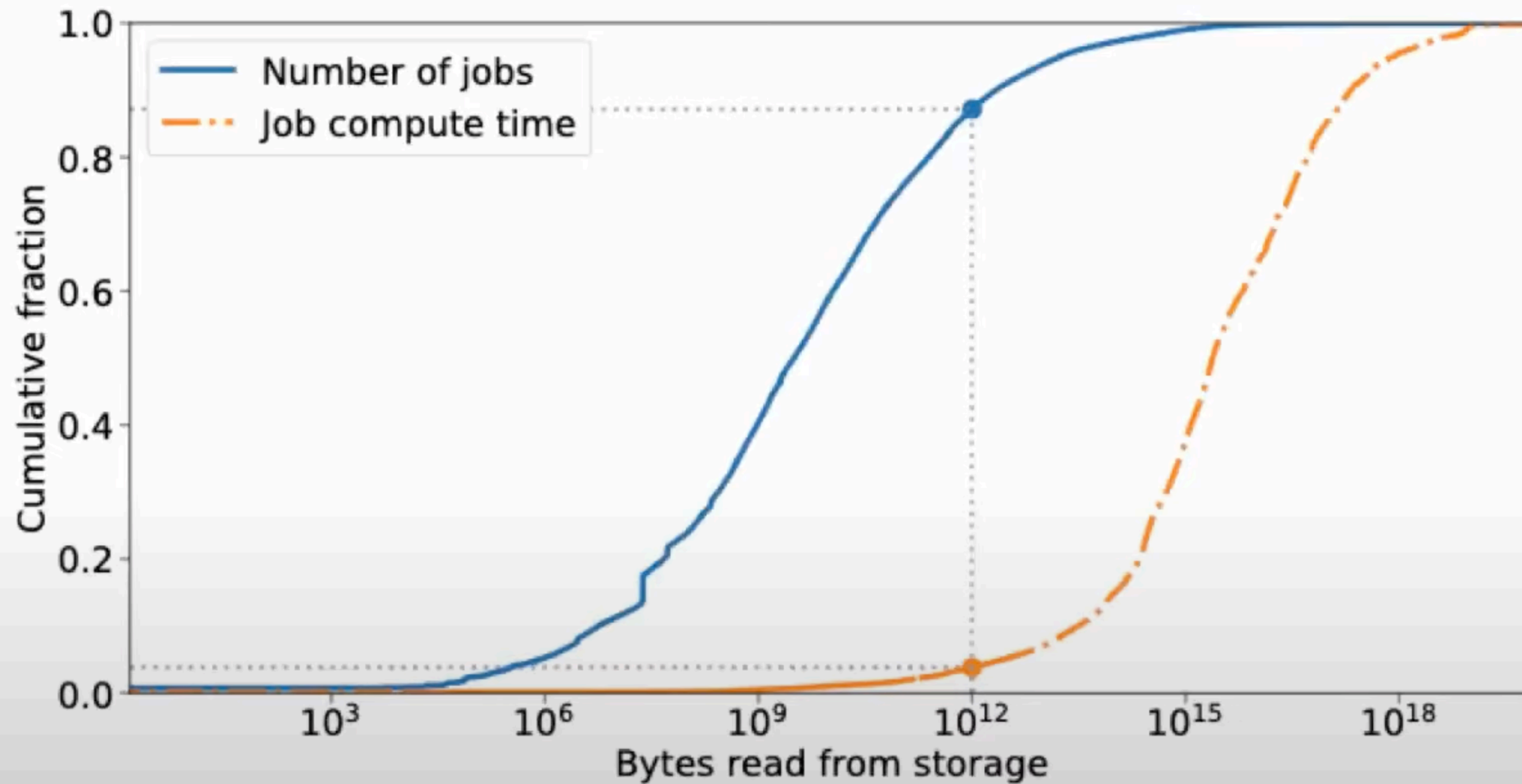
- ### Neural Message Passing Scheme
- Data-dependent computation graphs
 - Generalization of **any** neural network architecture

A new paradigm of how we define neural networks!

- No manual feature engineering needed:
 - ML over tables **without explicit joins**
 - Neurons of the neural network learn to perform joins
- End-to-end data-driven ML



At Google, 90% of all analytics workloads operate on less than 1 TB of data.



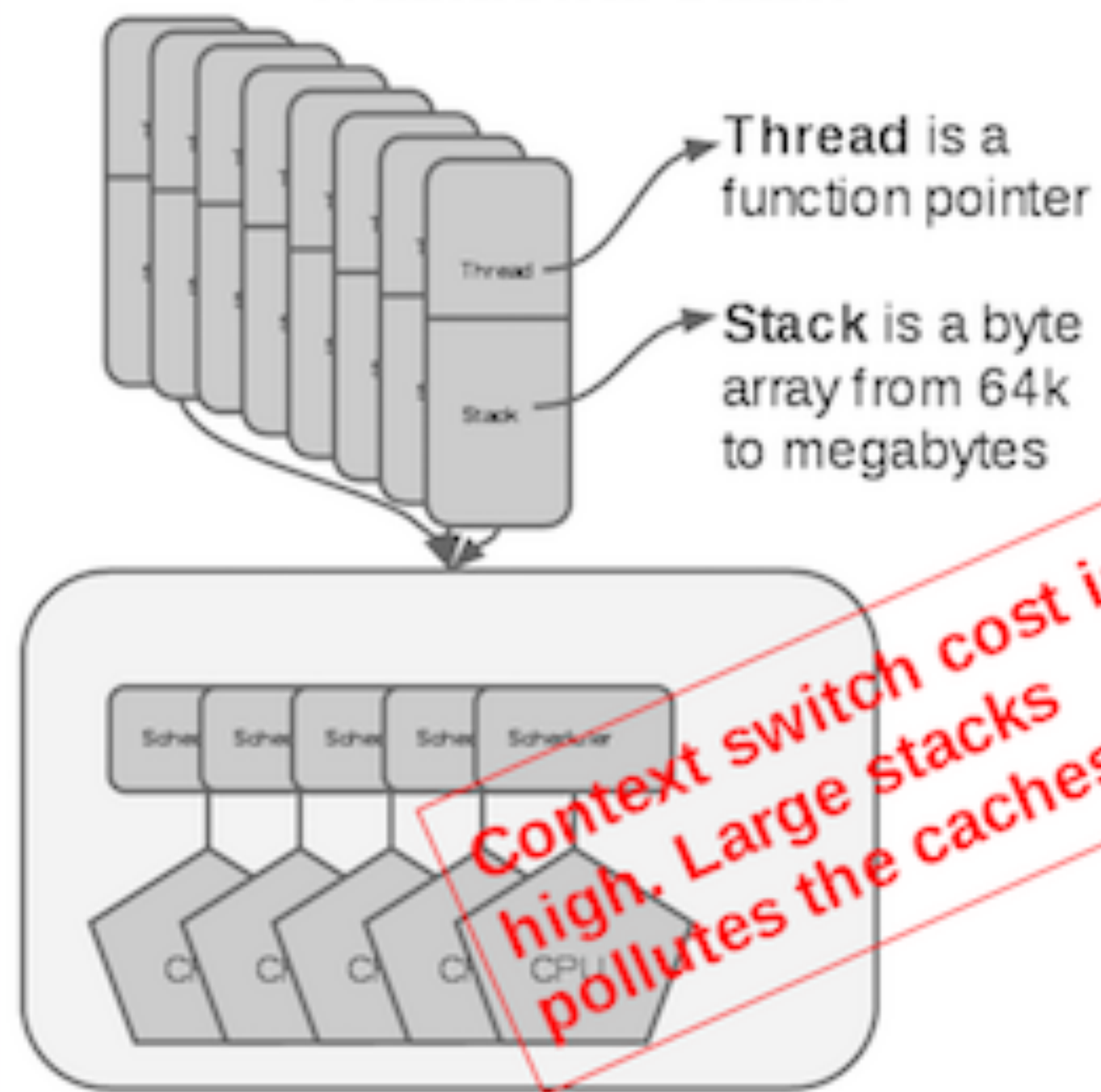
**Distribute On Cores
not on Servers**

Seastar Framework

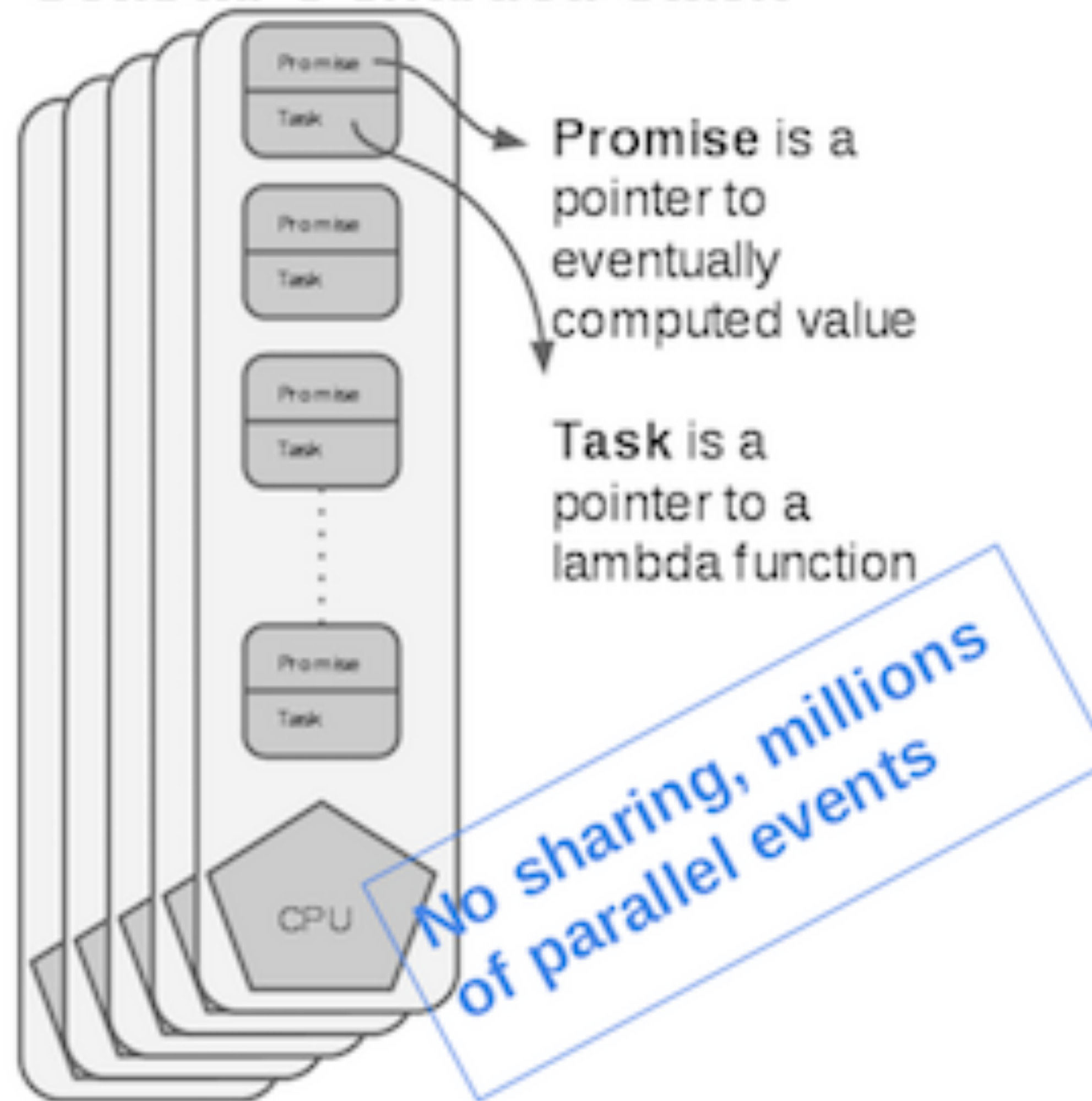
- Shared Nothing
- Message Passing
- Futures and Promises
- High Performance
Networking



Traditional stack



SeaStar's sharded stack





Queries



A Dozen to a Hundred



1-3 Money Queries

Monster Queries

- Complex Business Logic
- Bidirectional Traversals
- Involve relationship properties
- Weighted Shortest Paths
- Fan out Exploration Queries
- Queries through Dense Nodes



Queries that make



your Query Optimizer Cry





PGQL

- READ ONLY
- RPQs
- No GRAPH CONSTRUCT/PROJECT:
- NOT COMPOSABLE YET

ORACLE PGX

G CORE

ADVISES

- CREATE-READ
- RPQs
- GRAPH CONSTRUCT/PROJECT:
- COMPOSABLE

NO IMPLEMENTATIONS YET

Cypher

- CREATE-READ-UPDATE-DELETE
- No RPQs
- GRAPH CONSTRUCT/PROJECT:
- COMPOSABLE

- Neo4j DB
- Cypher for SPARK/Gremlin
- Agraph
- Memgraph
- Redis Graph
- inGraph
- SAP HANA Graph
- Cypher.PL



NEW FUSED

GQL

- CREATE-READ-UPDATE-DELETE
- RPQs
- GRAPH CONSTRUCT/PROJECT:
- COMPOSABLE

MAY 25 2020

1 COMMENT

DATABASE, RANDOM

EDIT

DECLARATIVE QUERY LANGUAGES ARE THE IRAQ WAR OF COMPUTER SCIENCE



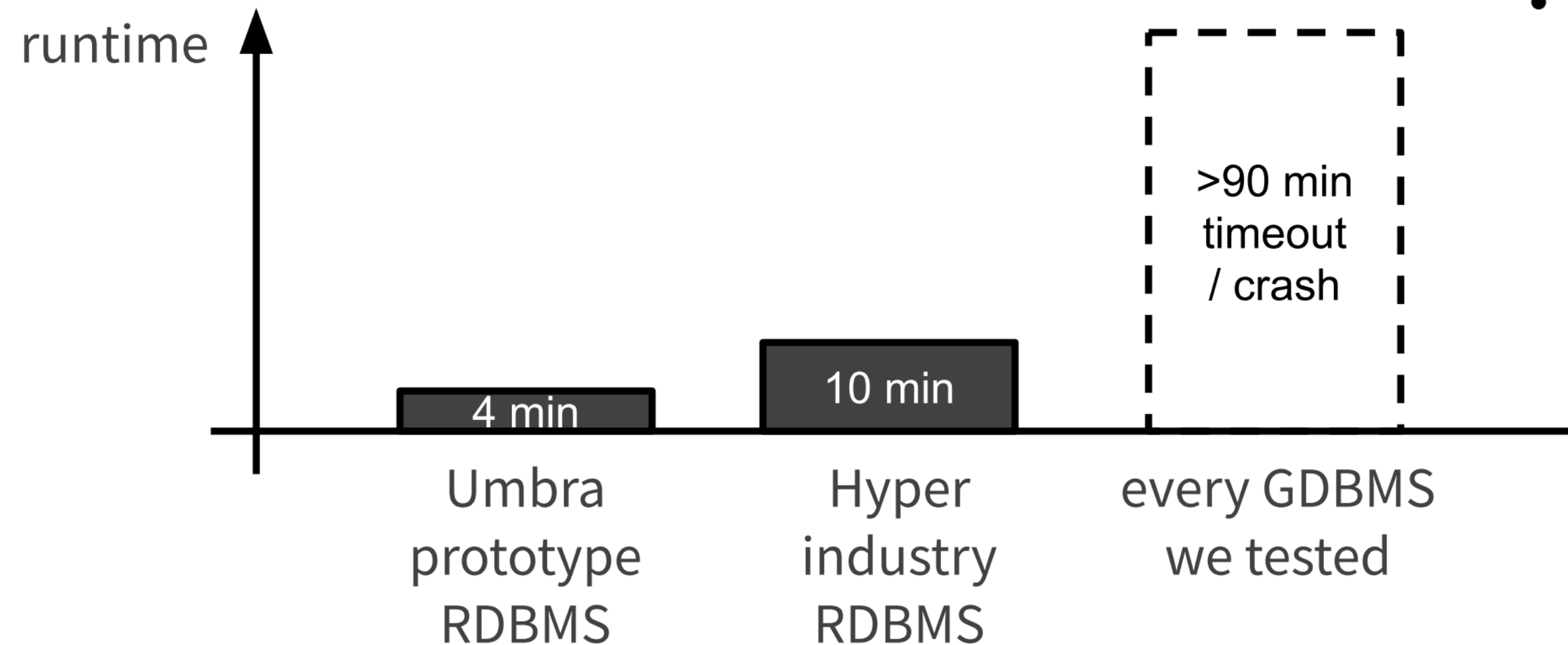
44 Different Queries

- There are at least 44 different ways to write:
“Find the customers who decreased their purchase amounts on their most recent order”
- 30 Unique Timings
- At least 30 ways for the Query Planner and Optimizer to execute

1. **46ms** Peso's GianLuca-copy
2. **46ms** Gianluca Again
3. **46ms** mByther 2
4. 50ms Gianluca Sartori
5. 60ms KevRiley
6. 60ms Paul Ireland
7. 60ms Kevriley 4
8. **60ms** Celko
9. 63ms Peso 3
10. 63ms KevRiley 5
11. 63ms giancula 3
12. 106ms Eric Pratley
13. 110ms Quan 2
14. 173ms girish Bhat
15. 280ms RobertFolkerts
16. 283ms Wm Brewer
17. 296ms Herman 2
18. 313ms AndyM
19. 330ms Peso
20. 330ms ivan.yong
21. 330ms Peso 4
22. 340ms andriy.zabavskyy
23. **343ms** maxdemarzi
24. 360ms Mark Marinovic
25. 360ms mByther 1
26. 376ms John McVay
27. 390ms Peso 2
28. 390ms Peter Brinkhaus
29. 390ms plamen
30. 423ms Alex Kuznetsov
31. 436ms Gustavo 2
32. 453ms back_of_the_fleet
33. **453ms** maxdemarzi
34. 560ms v_paronov
35. 610ms Steve Rowland
36. 610ms eemore-571761
37. 643ms Chris Howarth
38. 690ms Herman van Midden
39. 703ms jfortuny
40. 736ms Quan_L_Nguyen
41. 750ms Chris Howarth 2
42. 763ms Gustavo
43. 1030ms GSquared
44. **10766ms** Ramesh

GDBMS performance for subgraph queries

- Load the data: 100M vertices, 650M edges
- Run all 9 queries one-by-one (count number of matches)
- Environment: cloud VM, 370GB RAM, 48 vCPU cores



LDDBC BI SQL Queries

- The queries changed over time
 - Over 10x improvement gained by rewriting queries
 - *The optimizer should have been doing what we had to do by hand!*
 - Remove redundant joins with redundant relations
 - Common subquery elimination

10x faster

JAN 28 2019
2 COMMENTS

CYPHER, JAVA, PROBLEMS,
RANDOM

EDIT

NEO4J STORED PROCEDURES FOR DEVS THAT DON'T KNOW JAVA (YET)



Stored Procedure





Not Only Declarative

**Add a “real time”
Procedural Query Interface**

Lua

“Moon” in Portuguese

- Proven
 - Used in embedded systems and games
- Fast
 - Fastest scripting language I know of, and using LuaJIT
- Powerful, small and free (MIT)



Lua

As a Query Language

- Simple Queries

HEADERS BODY AUTHORIZATION 0 ACTION

Raw input ▾

```
1 NodeGet(["User", "Max"])
```

⋮ Response ×

200 OK

```
1 [-] [  
2 [-] {  
3     "id": 1027,  
4     "type": "User",  
5     "key": "Max",  
6 [-] "properties": {  
7     "age": 42,  
8     "name": "Max"  
9 }
```

Lua

As a Query Language

- Simple Queries
- Pipelined Queries

HEADERS BODY AUTHORIZATION 0 ACTIONS 0 CONFIG C

Raw input ▾

```
1 a = NodeGetId("User", "Max")
2 b = NodeGet("User", "Max")
3 c = NodeTypesGetCountByType("User")
4 d = NodePropertyGet("User", "Max", "name")
5 e = NodePropertyGetById(a, "age")
6 a, b, c, d, e
```

⋮ Response ×

200 OK

```
1 [-] [
2     1027,
3     [-] {
4         "id": 1027,
5         "type": "User",
6         "key": "Max",
7     [-] "properties": {
8         "age": 42,
9         "name": "max"
10    }
11 }
```

Lua

As a Query Language

- Simple Queries
- Pipelined Queries
- Complex Queries

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:7243/db/rage/lua
- Body:** Raw input
- Request Body (Raw input):**

```
1 names = {}
2 ids = NodeGetRelationshipsIds("User", "Max")
3 for k=1,#ids do
4     v = ids[k]
5     table.insert(names, NodePropertyGetById(v.node_id, "name"))
6 end
7 names
```
- Response:** 200 OK
- Response Body:**

```
1 [- [
2 [- [
3     "helene"
4 ]
5 ]
```

```
1 --ALL
2 get_city = function(person_id)
3     return NodeGetNeighbors("Person", person_id, Direction.OUT, "IS_LOCATED_IN")[1]
4 end
```

Then in our queries we can just reuse it:

```
1 --ALL
2 ldbc_snb_is01 = function(person_id)
3     local properties = NodeGetProperties("Person", person_id)
4     local city = get_city(person_id)
5     ...
```



Composable



Community

Let's build something
outrageous!




Follow RageDB on



 [Twitter](#)

Join us on  [Slack](#)




Help out on  [GitHub](#)



Script



 h: 0 m: 0 s: 2 ms: 211

```
1 count = 0
2 person = "person"..math.random(NodeTypesGetCountByType("Person"))
3 person_id = NodeGetId("Person", person)
4 liked_items = NodeGetRelationshipsIdsByIdForDirectionForType(person_id, Direction.OUT, "LIKES")
5 item_likes = LinksGetRelationshipsIds(liked_items, Direction.IN, "LIKES")
6
7 for item, likes in pairs(item_likes) do
8   other_person_likes = LinksGetRelationshipsIdsForDirectionForType(likes, Direction.OUT, "LIKES")
9   for other_person, likes2 in pairs(other_person_likes) do
10    for iter = 1, #likes2 do
11      count = count + 1
12    end
13  end
14 end
x 15 count
```

Bulk API

Response

[50131000]

On 4 Cores

```
./wrk -c 192 -t 32 -d 70s -s rage_get_node.lua --latency -R 190000
```

```
Running 1m test @ http://x.x.x.x:7243
```

```
32 threads and 192 connections
```

```
Thread Stats      Avg          Stdev         Max    +/- Stdev
```

```
  Latency        1.31ms    565.03us    11.08ms    74.35%
```

```
  Req/Sec        6.26k     400.79      8.67k     73.88%
```

```
Latency Distribution (HdrHistogram - Recorded Latency)
```

```
50.000%    1.23ms
```

```
75.000%    1.60ms
```

```
90.000%    1.99ms
```

```
99.000%    2.92ms
```

```
99.900%    5.84ms
```

```
99.990%    8.65ms
```

```
99.999%    9.88ms
```

```
100.000%   11.09ms
```

```
13297120 requests in 1.17m, 2.79GB read
```

```
Requests/sec: 189969.52
```

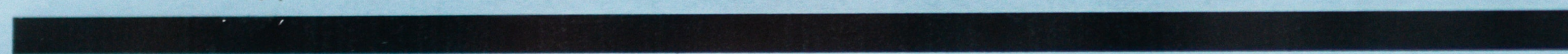
```
Transfer/sec:    40.88MB
```

190k Requests / Second



HELP

WANTED



Rage DB

An outrageous
graph database

@rage_database

ragedb.com

[GitHub.com/ragedb](https://github.com/ragedb)

hub.docker.com/u/ragedb



RAGE DB