Big

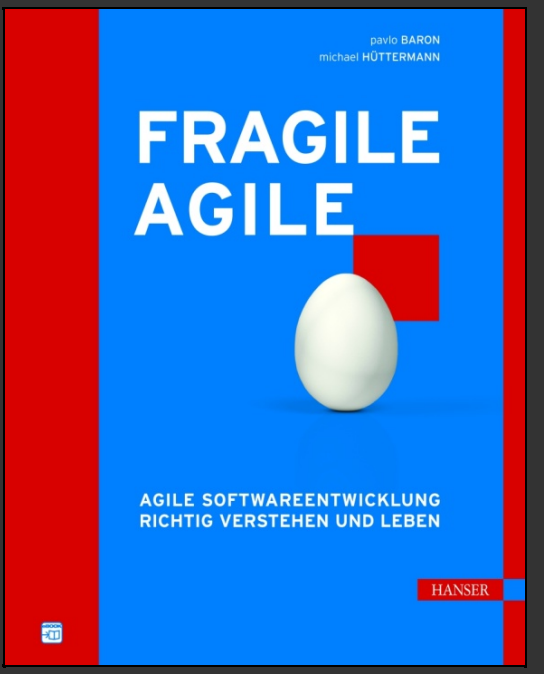
Big

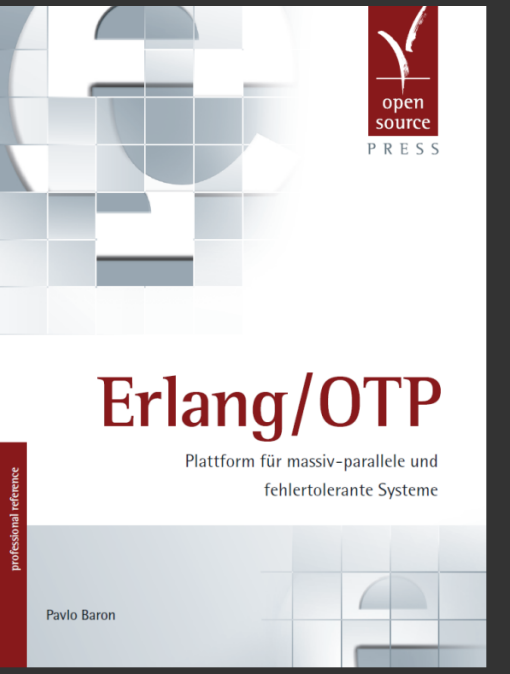
Big

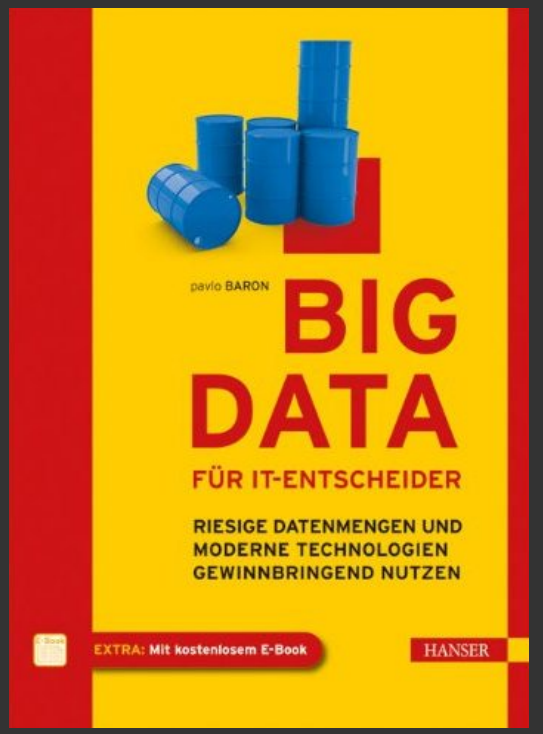
Big

Big

data developer







Pavlo Baron

pavlo.baron@codecentric.de
@pavlobaron

Disclaimer:

I will not flame-war, rant, bash or do anything similar around concrete tools here.

I'm almost tired of doing that

Hey, dude.

I'm a big data developer. We have enormous data.

I continuously read articles on highscalability.com

Big data
is easy.

It's like normal
data, but, well,
even bigger.

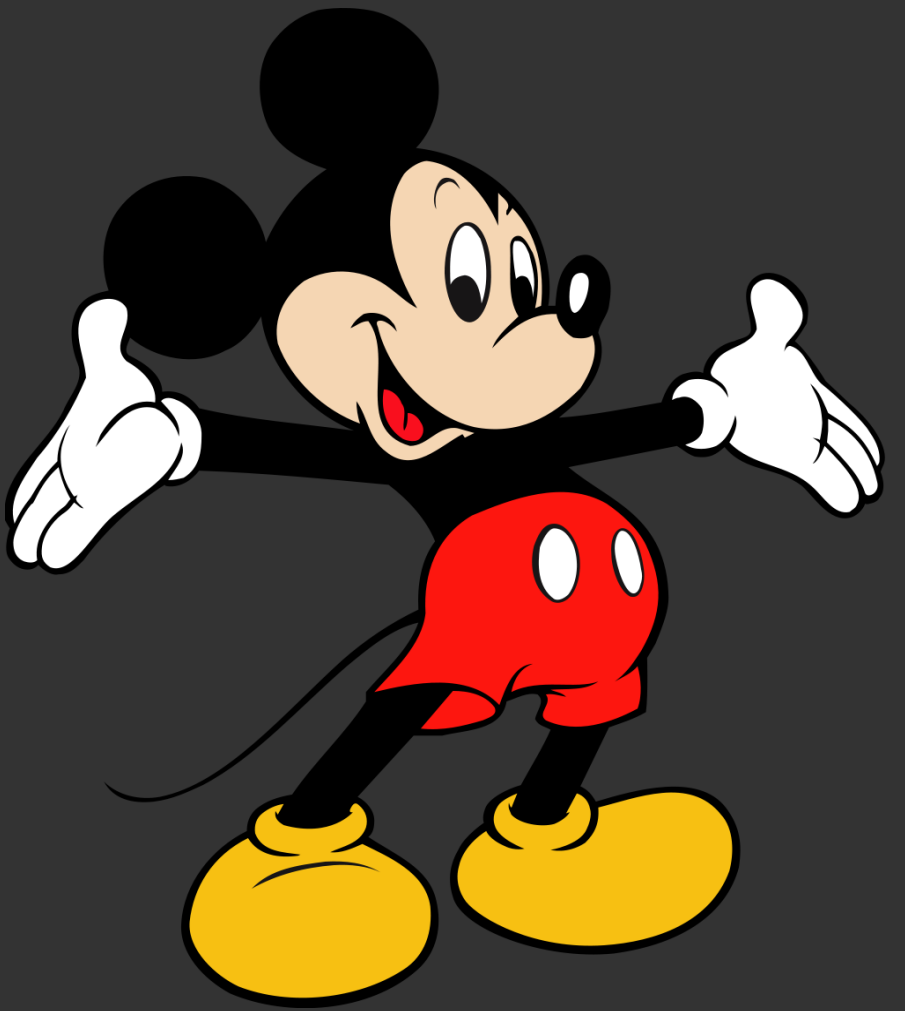
Aha. So this is not big, right?

That's big, huh?

Peta bytes of data every hour on different continents, with complex relations and with the need to analyze them in almost real time for anomalies and to visualize them for your management.

You can easily call this big data.

Everything below this you can call Mickey Mouse data



Good news is: you can intentionally grow – collect more data. And you should!

np, dude!

Data is data.

Same principles

Really?

Let's consider storage, ok?

np, dude!

Storage is just a database

Really?

Storage capacity of one single box, no matter how big it is, is limited

np, dude!

You're talking about SQL, right?

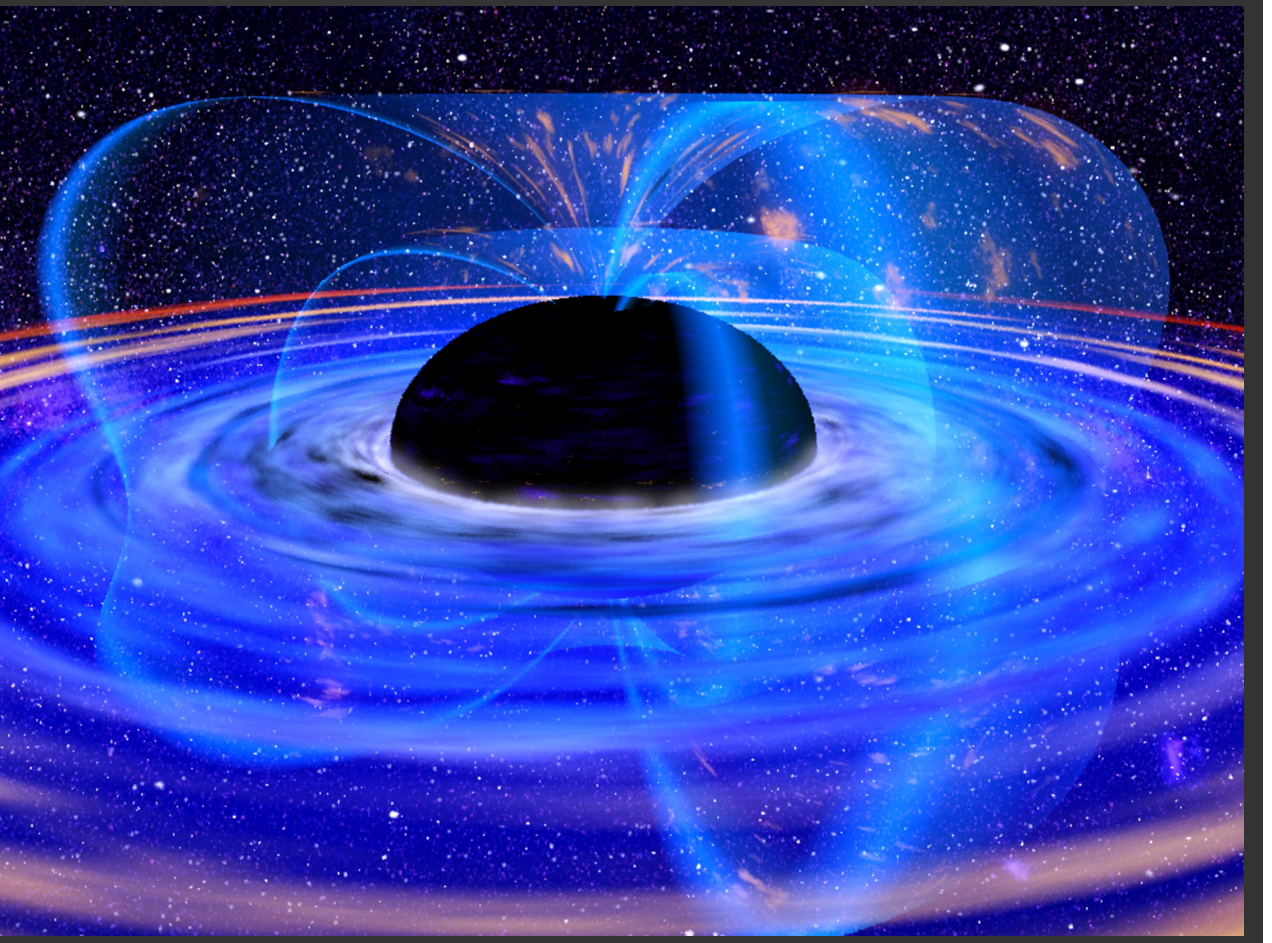
SQL is boring grandpa stuff and doesn't scale

Oh.

When you expect big data, you need to scale very far and thus build on distribution and combine theoretically unlimited amount of machines to one single distributed storage

There is no way around.

Except you invent the BlackHoleDB



np, dude!

NoSQL scales and is cool.

They achieve this through sharding, ya know?

Sharding hides this distribution stuff

Hem.

Building upon distribution is much harder than anything you've seen or done before

Except you fed a crowd with 7 breads and walked upon the water

np, dude!

From three of CAP, I'll just pick two.

As easy as this

Yeah, but...

The only thing that is absolutely certain about distributed systems is that parts of them will fail and you will have no idea where and what the hell is going on

So your P must be a given in a distributed system. And you want to play with C vs. A, not just take black or white

np, dude!

Sharding works seamlessly. I don't need to take care of anything

Seriously?

For example, one of the hardest challenges with big data is to distribute/shard parts over several machines still having fast traversals and reads, thus keeping related data together.

Valid for graph and any other data store, also NoSQL, kind of

Another hard challenge with sharding is to avoid naive hashing.

Naive hashing would make you depend on the number of nodes and would not allow you to easily add or remove nodes to/from the system

And still, the trade-off between data locality, consistency, availability, read/write/search speed, latency etc. is hard

np, dude!

NoSQL would write asynchronously and do map/reduce to find data

Of course.

You will love eventual consistency, especially when you need a transaction around a complex money transfer

np, dude!

I don't have money to transfer. But I need to store lots of data.

I can throw any amount of it at NoSQL, and it will just work

Really?

So you'd just throw something into your database and hope it works?

What if you throw and miss the target?

Data locality, redundancy, consistent hashing and eventual consistency combined with use case driven storage design are key principles in succeeding with a huge distributed data storage.

That's big data development

How about data provisioning?

np, dude!

It's database being the bottle neck, not my web servers

When you have thousands or millions parallel requests per second, begging for data, the first mile will (also) quickly become the bottle neck.

Requests will get queued and discarded as soon as your server doesn't bring data fast enough through the pipe

np, dude!

I'll get me some bad-ass sexy hardware

I bet you will.

But under high load, your hardware will more or less quickly start to crack

You'll burn your hard disks, boards and cards. And wires. And you'll heat up to a maximum

It's not about sexy hardware, but about being able to quickly replace it.

Ideally while the system keeps running

But anyway.

To keep the first mile scalable and fast, would lead to some expensive network infrastructure.

You need to get the maximum out of your servers in order to reduce their number

np, dude!

I will use an event driven C10K problem solving awesome web server. Or I'll write one on my own

Maybe.

But when your users are coming from all over the world, it won't help you much since the network latency from them to your server will kill them

You would have to go for a CDN one day, statically pre-computing content.

You would use their infrastructure and reduce the number of hits on your own servers to a minimum

np, dude!

I'll push my whole platform out to the cloud. It's even more flexible and scales like hell

Well...

You cannot reliably predict on which physical machine and actually how close to the data your program will run.

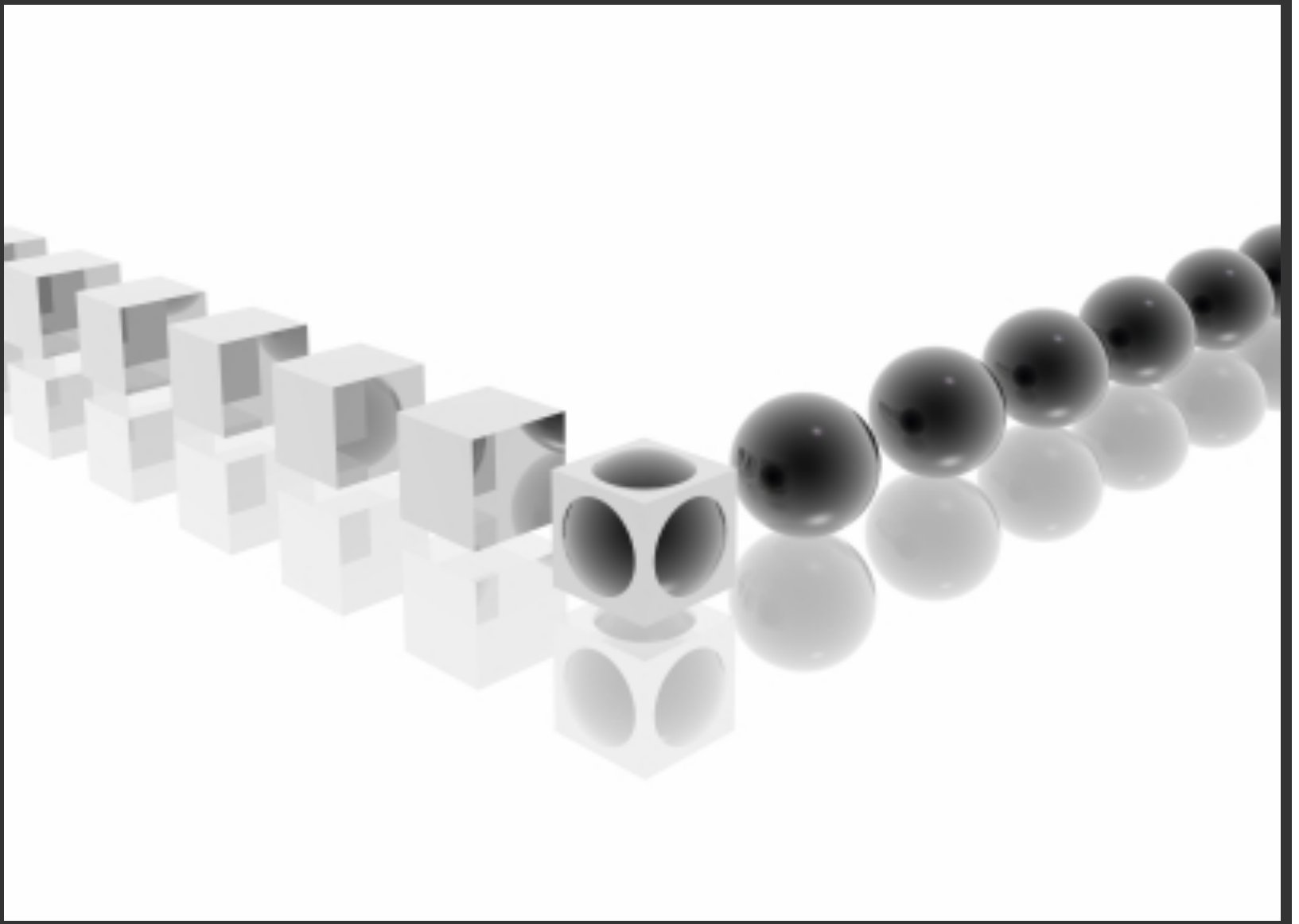
Whenever virtual machines or storage fragments get moved, your world stops

You can easily force data locality and shorter stop-the-world-phases by paying higher bills

Data locality, geographic spatiality, dedicated virtualization and content pre-computability combined with use case driven cloudification are key principles in succeeding with provisioning of huge data amounts.

That's big data development

Let's talk about processing, ok?



np, dude!

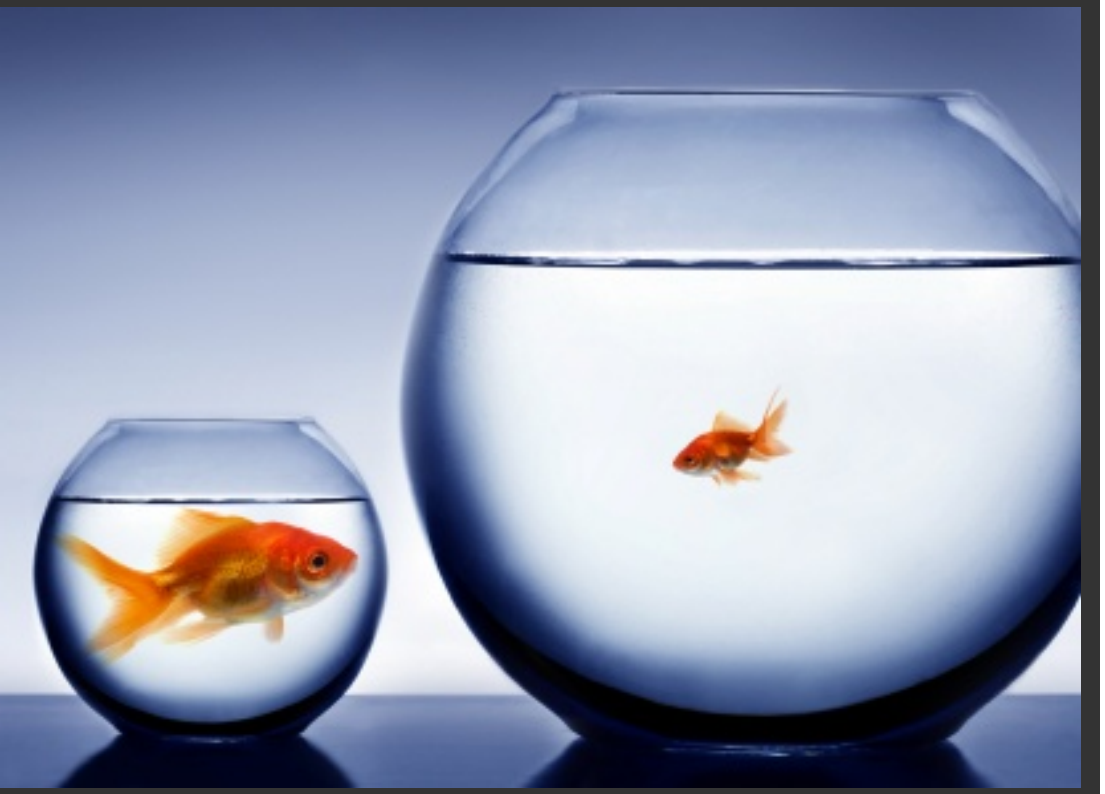
All easily done by a map/reduce tool

Almost agreed.

map/reduce has two basic phases:
even “map” and “reduce”

The slowest of those two is definitely “split”.

Moving data from one huge pile to another before map/reduce is damn expensive



np, dude!

I'll write my data straight to the storage of my map/reduce tool.

It will then tear

It can.

But what if you need to search during the map phase or even afterwards – full-text, meta?

np, dude!

I'll use a cool indexing search engine or library.

It can find my data in a snap

Would it?

A very hard challenge is to partition the index and to couple its related parts to the corresponding data.

With data locality of course, having index pieces on the related machines. It doesn’t help you much to find data through index while nodes holding it are unavailable

Data and index locality and direct filling of data pots as data flies by combined with use case driven technology usage are key principles in succeeding with processing of huge data amounts.

That's big data development

So, how about analytics, dude?

np, dude!

It's classic use case for map/reduce.

I can do this afterwards and on the fly

Are you sure?

So, you expect one tool to do both, real-time and post fact analytics?

What did you smoke last night?

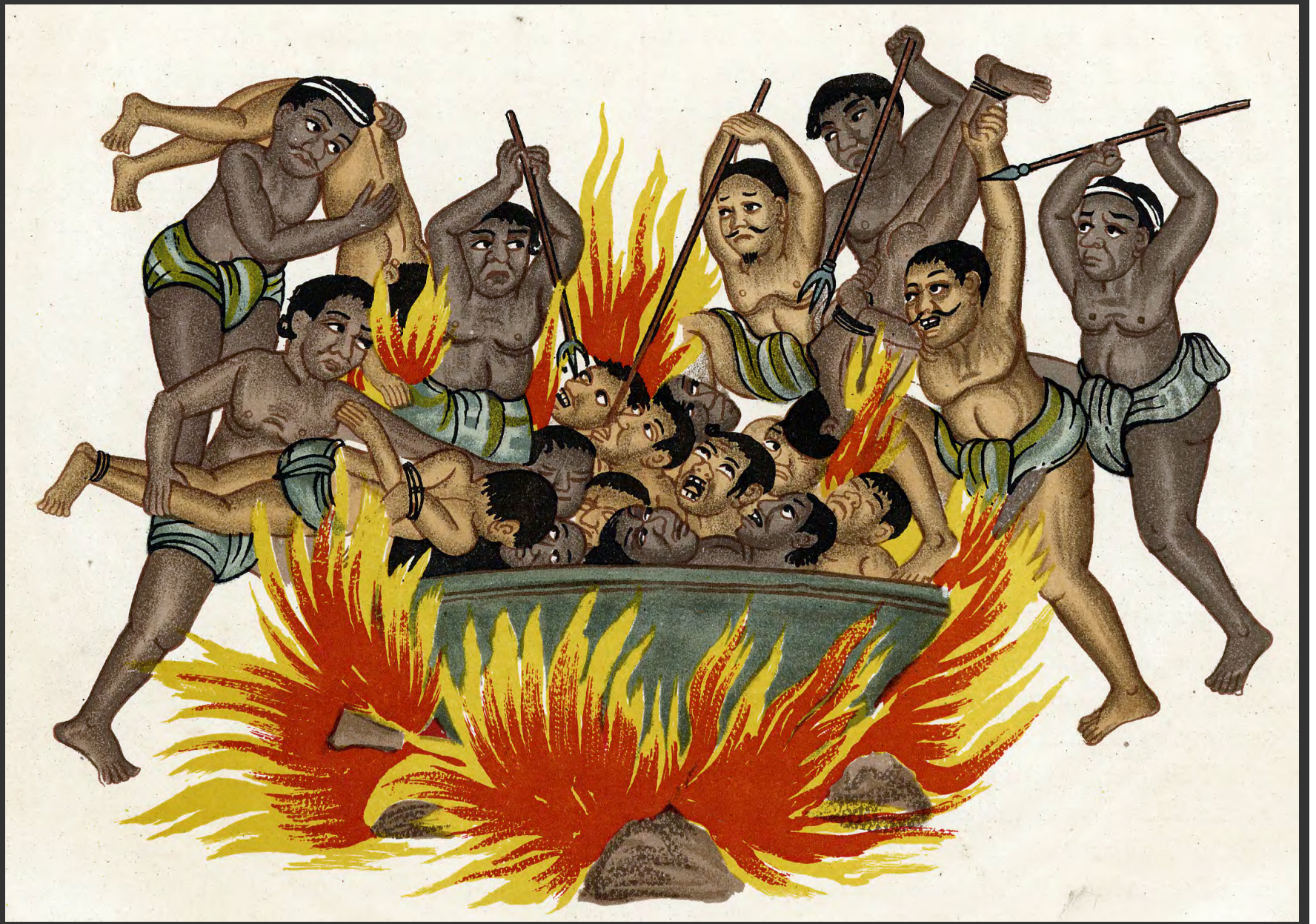
You don't want to believe in map/reduce in (near) real-time, don't you?

Realtime means REAL TIME, damn it!

It doesn't mean "as fast as possible" or "while you order a pizza".

Time is time, and realtime is all about doing something in fixed, prescribed time or just DIAF

‘Cause distribution has got your soul



np, dude!

I'll get me some rocket fast hardware

I'm sure you will. But:

You cannot predict and fix the map/reduce time.

You cannot ensure the completeness of data.

You cannot guarantee causality knowledge

If you need to predict better,
to be able to know about data/event causality, to be fast you need to CEP data streams as data flies by.

There is no (simple, fast) way around

But the most important thing is:

None of the BI tools you know will adequately support your NoSQL data store, so you're all alone in the world of proprietary immature tool combinations.

The world of pain.

np, dude!

My map/reduce tool can even hide math from me, so I can concentrate on doing stuff

There is no point in fearing math/statistics/ML. You just need it

Separation of immediate and post fact analytics and CEP of data streams as data flies by combined with use case driven technology usage and statistical knowledge are key principles in succeeding with analytics of huge data amounts.

That's big data development

Oh, we forgot visualization

np, dude!

I just have no idea about it

Me neither.

I just know that you can't visualize huge data amounts using classic spreadsheets. There are better ways, tools, ideas to do this – find them

That's big data development

hem, dude...

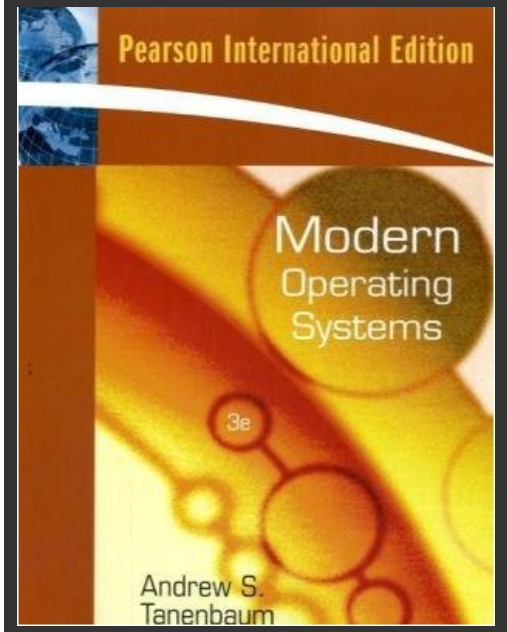
You're a smart-ass.

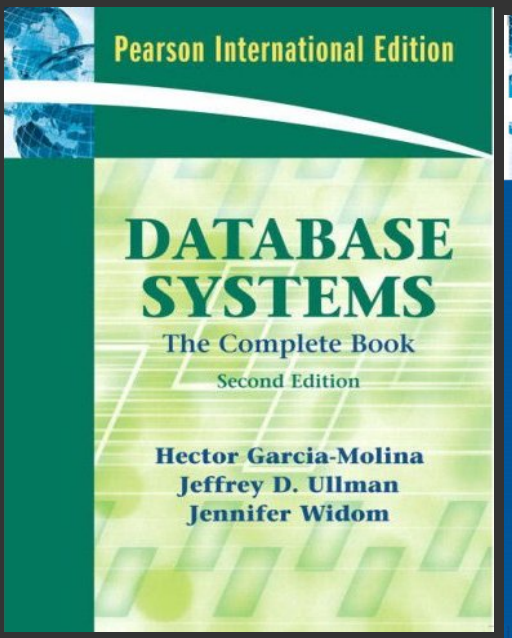
Is it that you want to say?

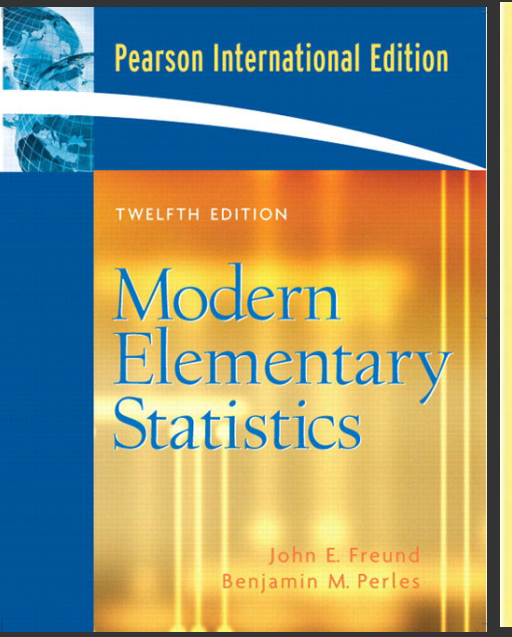
Almost.

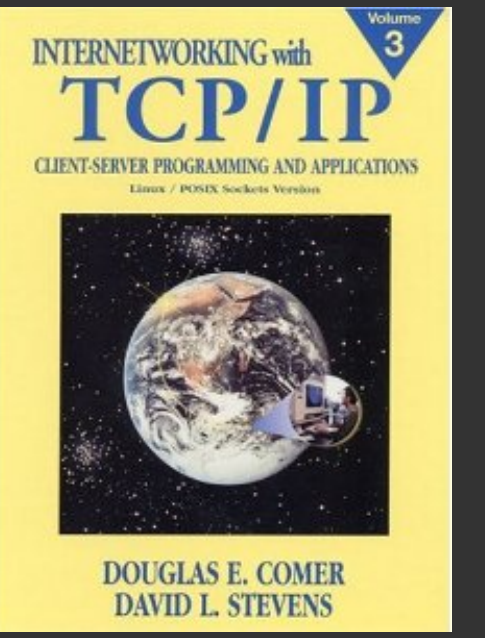
In one of my humble moments
I would suggest you to do the following:

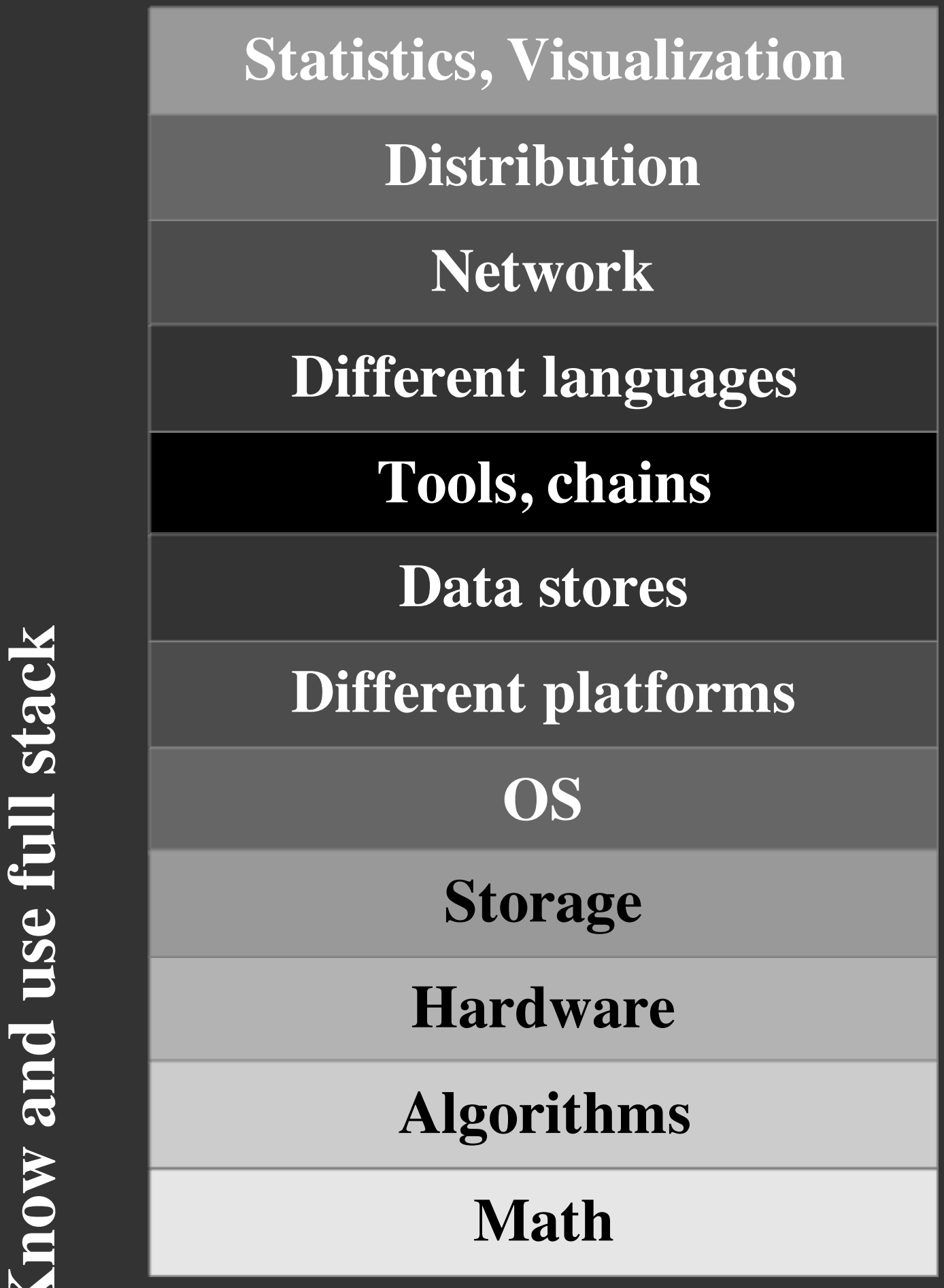
Stop thinking you gain adequately deep knowledge through reading half-baked blog posts. Get yourself some of those:










Statistics, Visualization
Distribution
Network
Different languages
Tools, chains
Data stores
Different platforms
OS
Storage
Hardware
Algorithms
Math
Know and use full stack

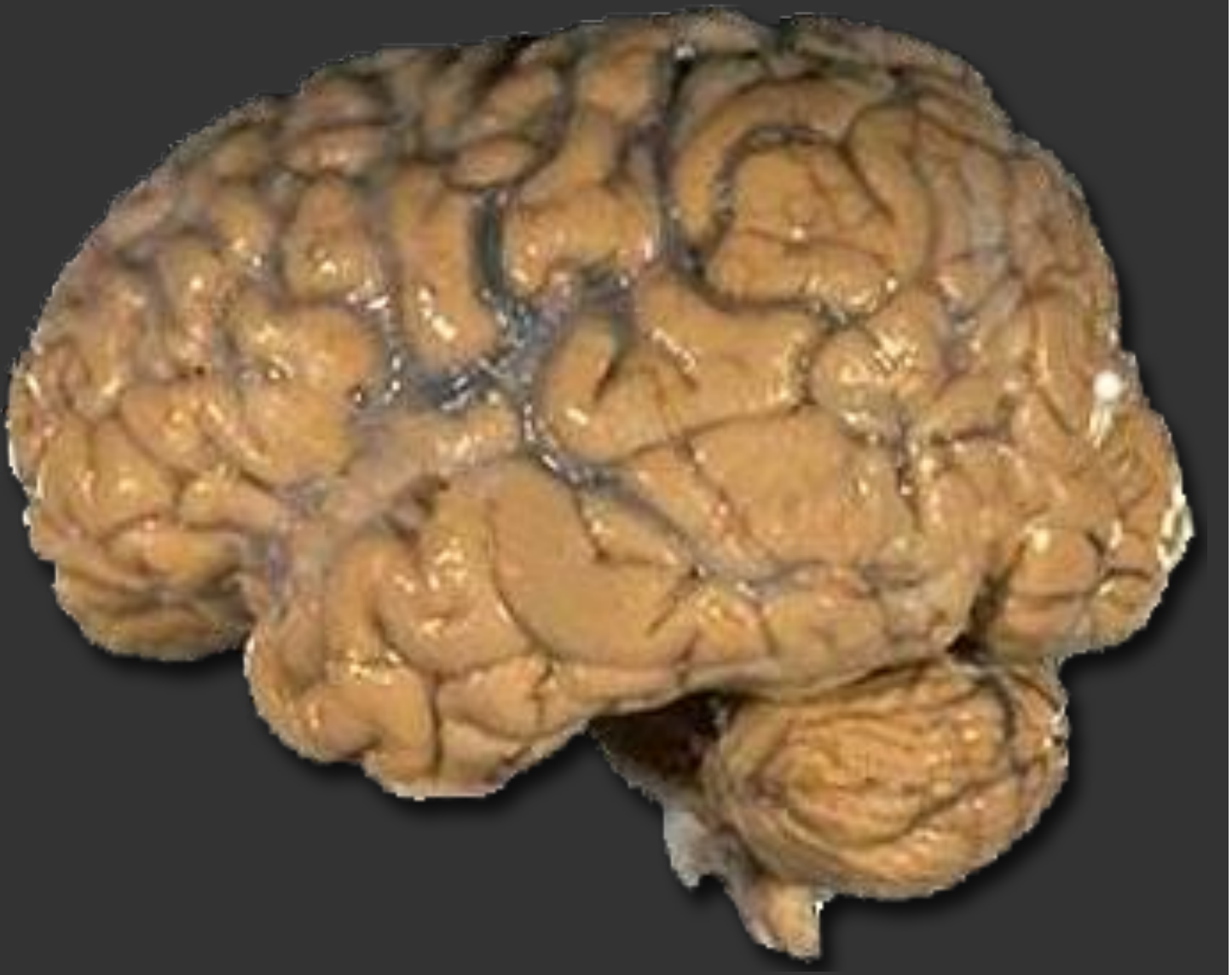
Know your point of pain.

You must be Twitter, Facebook or Google to have them all same time.

If you're none of them, you can have one or two. Or even none.

Go for them with the right chain tool

First and the most important tool in the chain is your brain



Thank you

Most images originate from
istockphoto.com

except few ones taken
from Wikipedia or Flickr (CC)
and product pages
or generated through public
online generators