# SO YOU HAVE TO BUILD SOME ANALYTICS

# OR

# LET'S NOT BUILD SOME ANALYTICS IF WE CAN HELP IT

# WHO AM I?

1. Consultant - Scale Assembly
2. Work on Ruby apps primarily
3. Have built 3 different "large" Analytics systems

# WHO IS THIS FOR

1. Programmers Who Want To Build An Analytics Application
2. Programmers Who Wonder HOW To Build An Analytics Application
3. A Third Point Would Really Tie This Slide Together

# WHAT IS AN ANALYTICS APPLICATION?

“an·a·lyt·ics: information resulting from the systematic analysis of data or statistics.”
A Hasty Google Search

Business Intelligence? Software Metrics?

Usually a mix of both.

# WHY WOULD I NEED TO BUILD ONE?

Haha because someone asked you to.

# SRSLY THO, WHY DO I NEED TO BUILD ONE?

Typically because the time to produce the results of an ad-hoc query exceed the amount of time a viewer is willing to wait for it.

In other words...

# QUERIES LIKE THIS

```
SELECT * FROM (
  SELECT customers.id, SUM(invoices.amount) AS invoice_amt
    FROM invoices
      INNER JOIN customers ON invoices.customer_id = customers.id
    GROUP BY invoices.customer_id
  ) t ORDER BY t.invoice_amt DESC LIMIT 10
```

# NO LONGER FINISH IN LESS THAN 30 SECONDS [1]

[1] Yes, people will wait 30 seconds for a page to load if it can make them a lot of money. It's in a footnote, totally accurate!

# CHOOSE YOUR OWN ~~MIS~~ADVENTURE

You are in a DARK CAVE. The MANAGER has asked you to build an ANALYTICS APPLICATION.

> look

To the NORTH is the comforting yet boring DATABASE YOU ARE ALREADY USING. To the SOUTH is THE LAND OF BIG DATA. You smell Hadoop from that direction.

# HOW DO I CHOOSE?

Optimizing the existing solution or choosing a new solution is a difficult problem. You need constraints.

“THE MORE CONSTRAINTS ONE IMPOSES, THE MORE ONE FREES ONE'S SELF. AND THE ARBITRARINESS OF THE CONSTRAINT SERVES ONLY TO OBTAIN PRECISION OF EXECUTION.

PRESENTING...

# A HANDY CHECKLIST FOR DECIDING HOW TO BUILD YOUR ANALYTICS

1. How often does the data change?
2. Does the data need to be "realtime"?[2]
3. Does total amount of data fit into a single machine's memory?
4. How fast does your data grow?
5. Do you know in advance all the questions you intend to answer?
6. Can you tolerate approximate answers?

[2] Ugh.

You may have noticed that all of those questions dealt with the DATA, not the DATABASE.

We are concerned with the data and how it must be presented, not the mechanical characteristics of a given database or software package. Details like it's massively parallel! or its single machine performance is incredible! are not really relevant to us at this stage.

# EXAMPLES (1):

I am a programmer named Aardvark. I have been told to build an analytics solution for my Jetpack business. My manager reviews a monthly report that tells him who the top 10 customers are for each of the past 12 months. We have 200,000 customers. The page that loads this report takes 5 minutes to run.

# EXAMPLES (1):

Aardvark does not need a new system. The manager looks at this report monthly, so create a cron job that runs it every month and saves it to a flat file. Load the flat file when the page is viewed.

Eventually the manager will want the page updated on demand, so add a button that queues a job to regenerate the report.

Also, Aardvark should check out database indexes.

# EXAMPLES (2):

I am a programmer named Belvedere. I work at a hot startup that sells handlebar mustaches to Pokemon. We have 2M customer records, and are adding between 8,000-10,000 a day. Our CTO wants to run SQL queries to research user behavior, but honestly we're terrified to touch the database for fear of blocking production traffic.

# EXAMPLES (2):

Belvedere does not need a new system. Add a read replica to the database that is intended for Business Intelligence; the CTO is already willing to write SQL, just give them a safe place to run it.

# EXAMPLES (3):

I am a programmer named Chaucer. I work at a mobile phone company where we have to give customers reports on total minutes used, total number of phone numbers called, and total amount of data used, over any length of time greater than 1 minute. This data is used for both reporting and billing purposes. We have 10M customer records, that generate (on average) 1,000 call records per month, and add 80,000 new customers per month.

I may already have a drinking problem.

# EXAMPLES (3):

Chaucer MAY need a new system. The combination of a steady growth rate of data, and the requirement to access data over any arbitrary time slice, means that the data cannot be easily optimized for something like ad-hoc SQL queries.

What techniques can Chaucer bring to bear here?

# CHEATING YOUR DATA WITH NO REMORSE

You can cheat your data problems by exploiting how the data is written and how the data is read. By using the questions from the checklist from earlier, you can use the natural read, write and storage characteristics of the data to determine where you can sacrifice flexibility for sustainability.

# TECHNIQUE 1: RECOGNIZE COLD DATA

If the data is written but never changed, you can roll it up into more manageable aggregates as it is written. (Just make sure you can regenerate those aggregates from the source data. Aggregation is information loss!)

```
BEGIN;
INSERT INTO call_records (number, customer_id, duration, date)
        VALUES ('555-555-5555', 10, 10, '2016-08-23');
INSERT INTO monthly_usage (date, customer_id, duration)
        VALUES ('2016-08-23', 10, 10)
      ON DUPLICATE KEY UPDATE duration = duration + 10;
COMMIT;
```

# TECHNIQUE 2: SCOFF AT REALTIME

If the data must be written quickly but isn't going to be acted upon until some period of time later (>10 seconds, let's say), do your critical operations now and queue the rest to be stored in an analytics database later.[3]

[3] May cause queue overflow and lagged queue, consult your doctor before queueing everything.

# TECHNIQUE 3: KNOW WHAT YOU NEED TO KNOW AHEAD OF TIME

If you know the questions you need to answer beforehand, you can know what sorts of data you can safely ignore at write time. If some function exists like:

```
var f = function affectsReports(data){ ... } // returns true or false
```

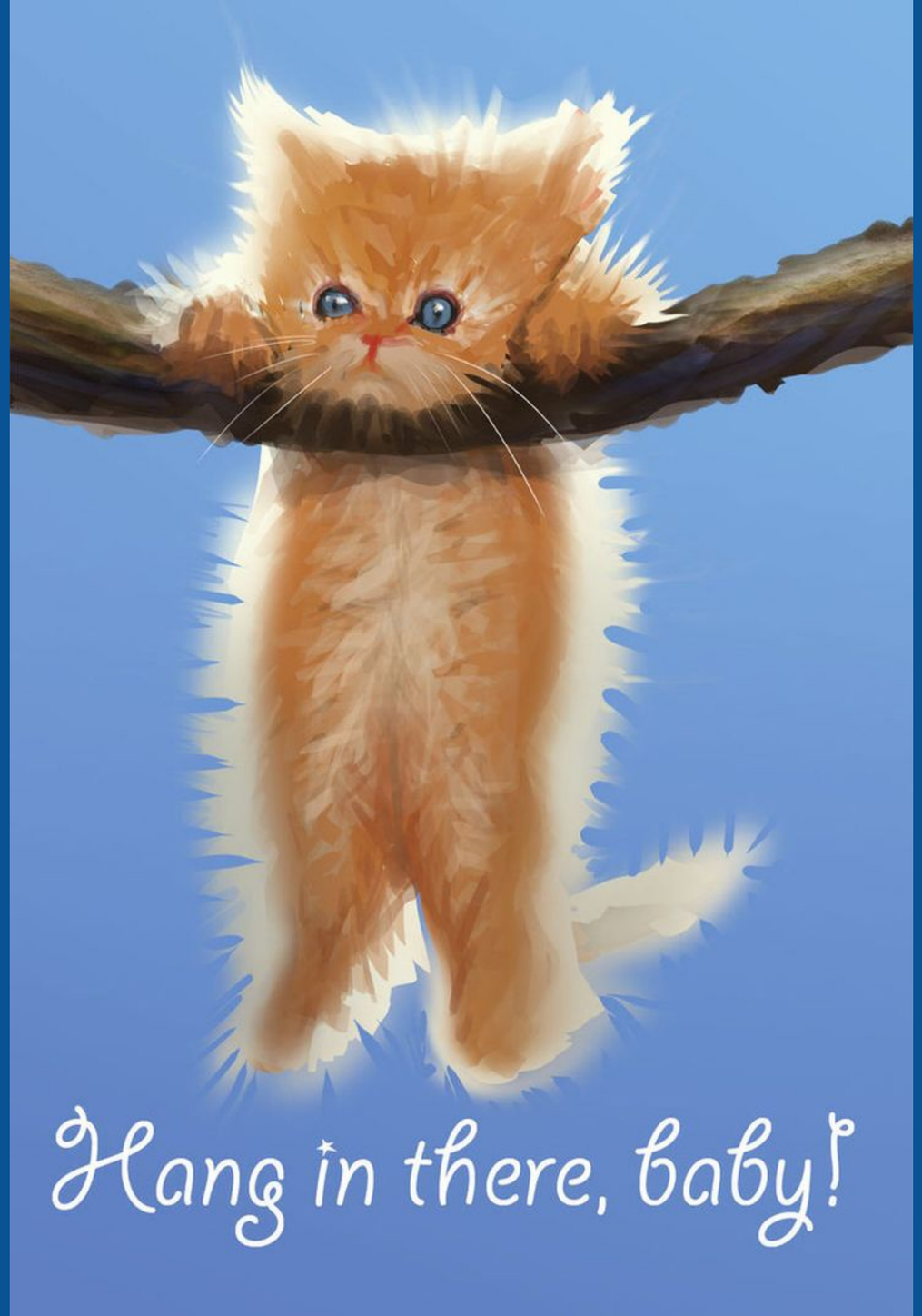
then you can tell at write time whether this data is even relevant to your analytics concerns. The more you can know ahead of time, the more you can cut down data size.

# TECHNIQUE 9001: RECOGNIZE IMPOSSIBILITIES WHEN CONFRONTED WITH THEM

If you are told you must build a system which presents real time analytics for 5,000 payloads per second each of size 200kb on a single machine, realize that someone just asked you to process business data at 1 gigabyte per second.[4]

Use what you know about your data, and the capacities of the system that it lives on, to make good judgements on how your system is even able to live in the reality we exist in.

[4] Soulver is amazing for this


Hang in there, baby!

# THANKS! HERE'S SOME LINKS...

1. Guerilla Capacity Planning
2. Don't Use Hadoop - your data isn't that big
3. Latency Numbers Every Programmer Should Know
   ( Interactive )
4. HighScalability
5. Soulver

Honestly this is just kind of a grab bag of topics.

# AND ALSO...
# COME TO PLIBMTTBHGATY
# AND TRY SOMETHING NEW

REGISTRATION LINK

# BONUS ROUND:
## WHO WANTS TO TALK ABOUT HYPERLOGLOG?!?!

1. HyperLogLog Wiki
2. Original Paper
3. Google Alterations
4. Antirez post
5. hyperll gem
6. ccard-lib