
speedshop

Determining Ruby Process Counts

speedshop

3 ActiveRecord Mistakes That Slow Down Rails Apps: Count, Where and Present

by **Nate Berkopec** (@nateberkopec) of ***speedshop*** (who?), a Rails performance consultancy.

Summary: Many Rails developers don't understand what causes ActiveRecord to actually execute a SQL query. Let's look at three common cases: misuse of the count method, using where to select subsets, and the present? predicate. You may be causing extra queries and N+1s through the abuse of these three methods. *(2778 words / 12 minutes)*

OUTLINE

1. Understanding the math
2. What the math means
3. Math Check!
4. Real world application

HOW MANY SERVERS DO YOU NEED?

300 REQUESTS PER MINUTE

250 MILLISECOND AVERAGE

1 SECOND 95TH PERCENTILE

LITTLE'S LAW

ITEMS IN SYSTEM = ARRIVAL RATE X TIME IN SYSTEM


ALL CHECKSTANDS ARE
ACCESSIBLE TO CUSTOMERS
WITH DISABILITIES
2
4
6
8
10
12
13

WAIT TIMES
UTILIZATION
10 CUSTOMERS ENTER LINE PER MINUTE
TAKES 1 MINUTE TO CHECKOUT CUSTOMER
20 CHECKOUT COUNTERS

AN EXAMPLE WITH SIMPLE NUMBERS

100 REQUESTS PER MINUTE

250 MILLISECOND RESPONSE TIMES

5 SERVERS

5 WORKER PROCESSES PER

WHAT DOES IT MEAN

TRUE

IN THE LONG RUN

4 95TH PERCENTILE REQUESTS IN A ROW
HAPPENS 1 IN 100,000 REQUESTS
3 TIMES PER DAY AT 200 RPM

UNDERSTANDING
REQUEST QUEUES IN RUBY WEB APPS

SCALING REDUCES WAIT TIMES

NOT RESPONSE TIMES

AUTOSCALING
MUST BE BASED ON WAIT TIMES

HOW TO FIND REQUEST QUEUE TIMES

EASY TO SEE UNDERSCALING
HARD TO SEE OVERSCALING

PREDICTABLE RESPONSE TIMES ARE SCALABLE RESPONSE TIMES

STORY:

REDUCING WAIT TIMES ON A HIGH 95-TH APP

WHAT WE KNOW SO FAR

WIP = LATENCY * ARRIVAL RATE

SCALING REDUCES QUEUE

PREDICTABLE IS SCALABLE

HOW UNICORN WORKS

TWITTER, 2008

» 600 requests/second

» 180 application processes (mongrel)

» About 300ms average server response time

SHOPIFY, 2013

» Shopify receives 833 requests/second.

» They average a 72ms response time

» They run 53 application servers with a total of 1172 application processes (!!!) with NGINX and Unicorn.

ENVATO, 2013

» Envato receives 115 requests per second

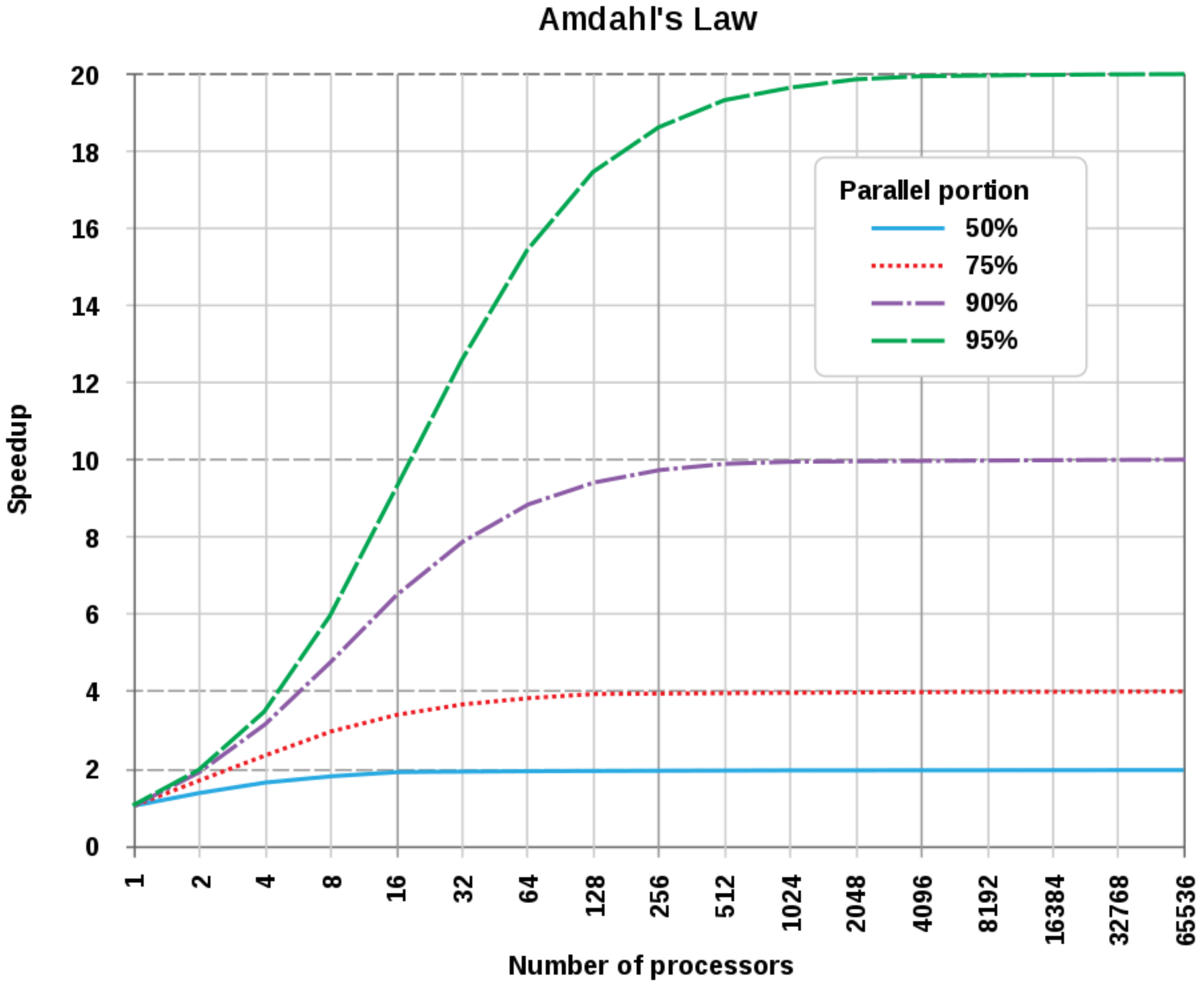
» They run an average of 147ms response time

» They run 45 processes

WHY THE PROCESS?

THREADS VS PROCESSES

AMDAHL'S LAW


Amdahl's Law
Speedup
Number of processors
0
2
4
6
8
10
12
14
16
18
20
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Parallel portion
50%
75%
90%
95%


GVL

WHAT % SPENT HOLDING GVL?

BREAKING THE LAWS

DIVIDING SERVERS UP - BIG OR SMALL?

WHERE DO REQUESTS QUEUE?

LOAD BALANCER BEHAVIOR

PASSENGER BEHAVIOR (UNIQUE CASE)

FILLING UP RESOURCES (CPU/MEM)

SERVERS

CONTAINER STORY

OTHER SERVICES
BACKGROUND JOBS

OTHER SERVICES
DB, JRUBY, LANGUAGES

CHECKIN: WHAT WE KNOW

LITTLE'S LAW
HOW REQUEST QUEUES WORK
HOW TO PROVISION
HOW TO SAVE $$$

THANK YOU!

@NATEBERKOPEC

MORE TACTICS AT:

RAILSSPEED.COM