A Data Driven Approach to TCP Tuning

By: Spencer Stingley & Wes Hardaker

Motivation

There are three major motivations for this research

- To improve the TCP throughput of the production DNS server via kernel parameters
- To develop a repeatable methodology to validate improvements to the setup, using statistical methods to prevent bias
- To test the effectiveness of the DIINER testbed

Parameter Selection

Goal: improve TCP throughput

Mechanism: all the parameters in /proc/sys of the DNS server

The general strategy was to address

- **Memory allocation parameters**
- **Frequency scaling methods**
- **Any seemingly relevant tcp parameters**

A very helpful resource in finding determining the effect of various parameters is:

- https://sysctl-explorer.net/

Varying Conditions

Measurements taken after *dnsperf* roughly stabilized (~5 minutes).

- Mean Latency
- Mean Connection Latency
- Queries per second
- Queries Lost

Two DNSperf conditions shown today:

- Low Utilization:	12 clients,	12 threads,	100 in-flight
- High Utilization:	13,000 clients,	400 threads,	100,000 in-flight

Note: case2 reached the maximum capacity and couldn't handle more output

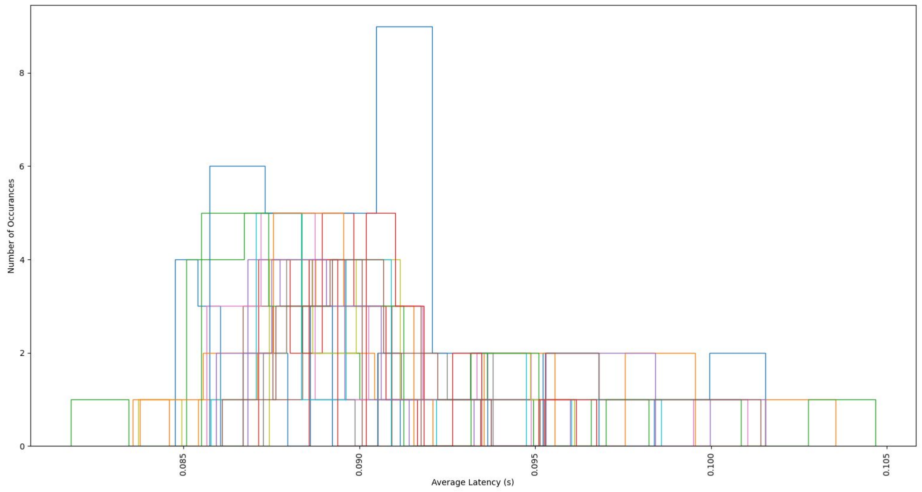
Consistency in the Conditions

- DNSperf needs to run for roughly 5 minutes to reduce the variance on the trials and let timeout messages steady out on larger workloads
- DNSperf needs to be tuned to the specific machine to be able to increase the server load to maximum capacity.
    - Max utilization of the receiving server achieved: ~98%
- To quantify the server utilization, the average cpu usage over a 30 second interval was taken at 3 times over 5 trials at baseline conditions

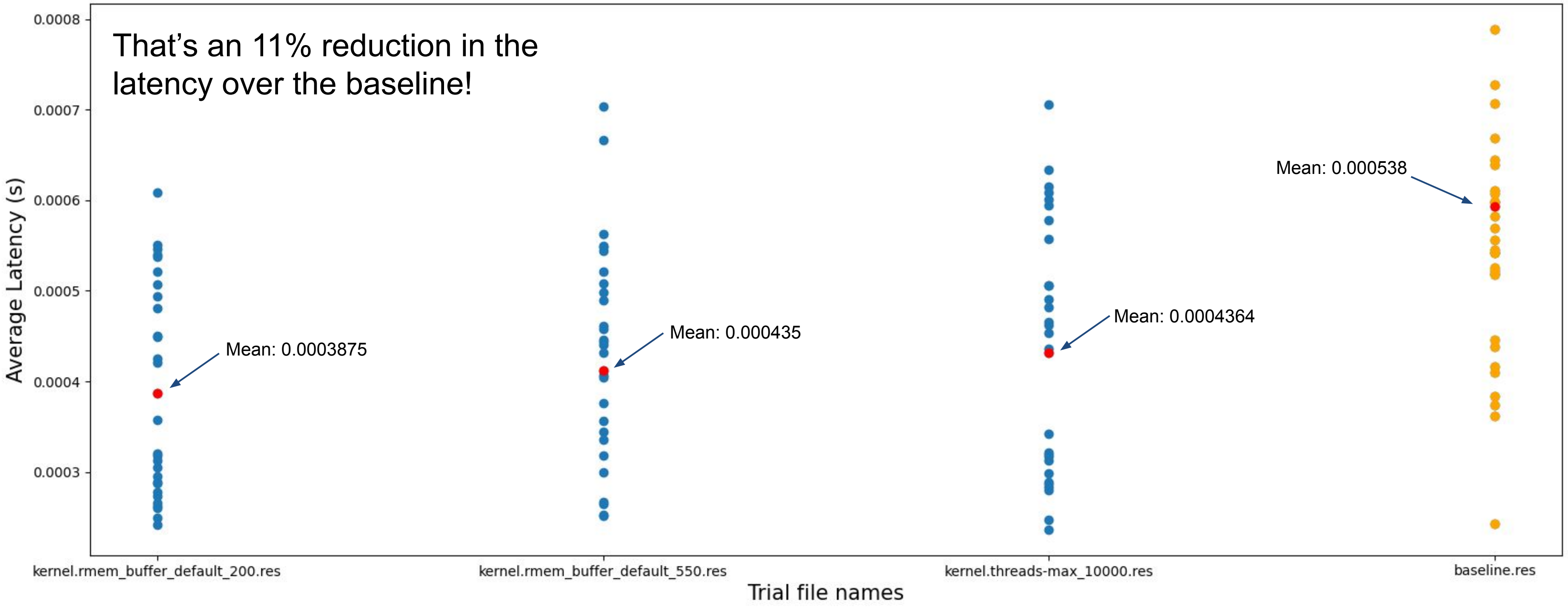
Statistical Analysis

Since the dependent variable is continuous, the data is approximately normal, the groups are independent, and the variances are relatively homogeneous, the anova test should be used to determine if there is any statistical significance

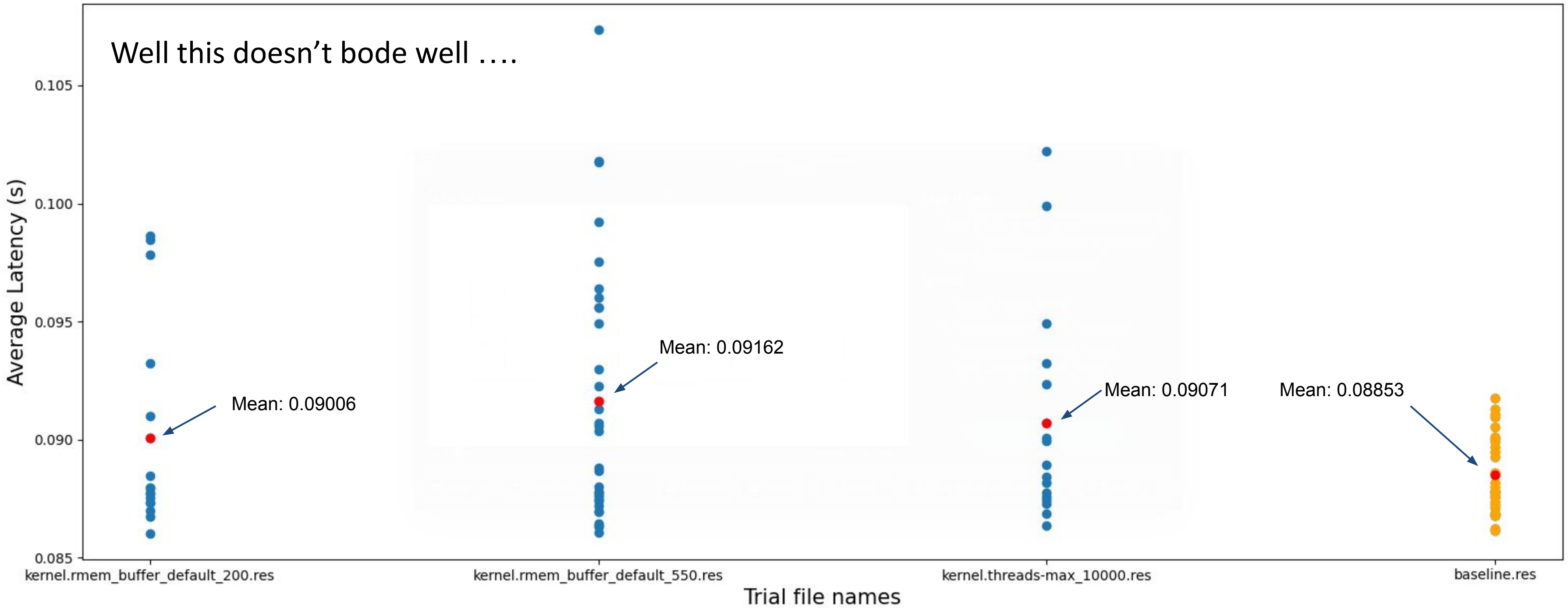
The ANOVA test will not tell tell us if any particular parameter has significance, simply if there is some present. A post hoc test like Dunn's or Tukey's should be applied to find the significant parameters



A brief tour through the low load data



A brief tour through the high load data

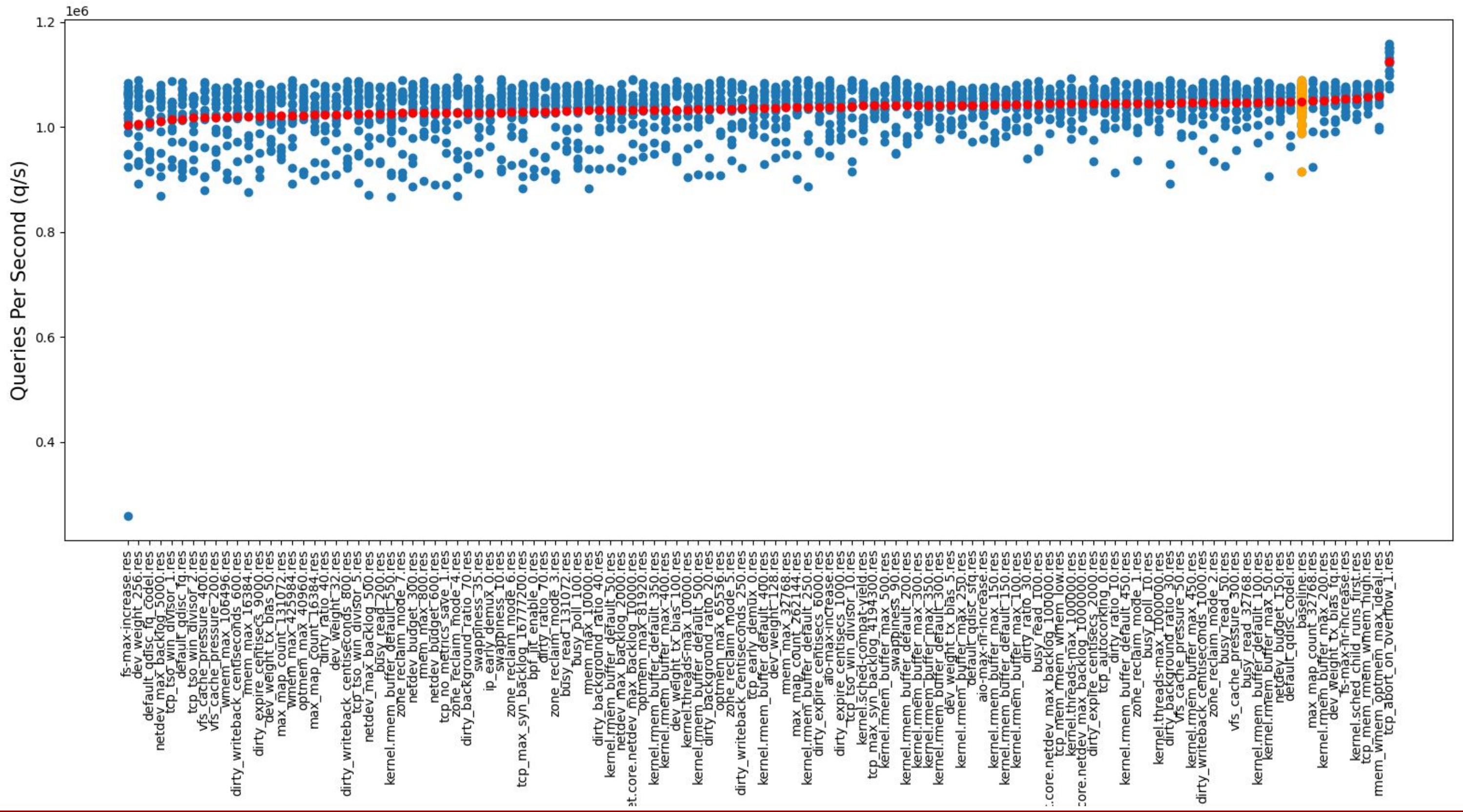


Statistically verifying our intuitions

The list of statistically significant kernel optimizations under minimal load:

Good news: If you aren't under extreme load, default settings are just fine!

A global view of the intense load data



Tuning Under Intense Load

tcp_abort_on_overflow = 1 does reduce the latency by roughly 5% and increases queries per second by about 8%, decreased lost packets by 98%

- Both Tukey's and dunn's post hoc tests verify its significance
- P-value of 0.06

Honorable Mentions:

- fs.file-max = 524288 is an improvement according to tukey
- dev_weight_tx_bias = 50 is useful according to dunn
- default_qdisc = fq_codel is bad according to dunn
- ip_early_demux = 0 is bad according to dunn
- tcp_tso_win_divisor < 3 is bad according to dunn
  - Increasing it doesn't appear to matter

Summary

- Be careful while collecting data!
  - Ensure testing under high load
  - Multiple trials are required
  - Study of statistical significance is critical
- Important tuning parameters:
  - **tcp_abort_on_overflow = 1**
  - Other parameters from the Ant Tuning DNS for TCP Queries Webpage:
    - https://ant.isi.edu/diiner/tcp/index.html
- Results will be published on the DIINER webpage
  - (I have written some very helpful python code to automate the analysis above)

Questions?