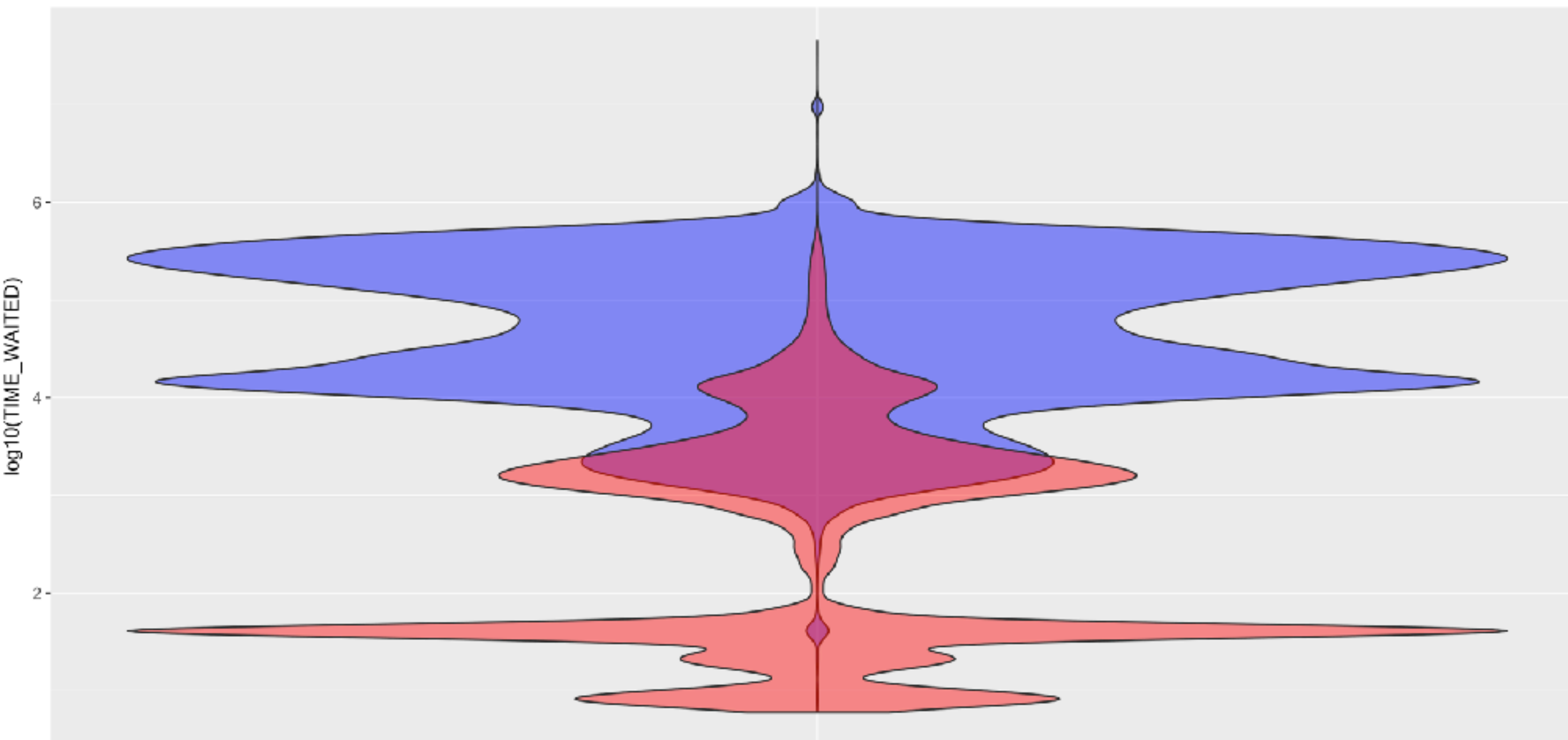
Visualizing ASH

John Beresniewicz

NoCOUG 2018



Agenda

- What is ASH?
  - Mechanism and properties
  - Usage: ASH Math, Average Active Sessions
- ASH Visualizations
  - EM Performance: Wait class details, Top Activity,
  - EM Treemaps: Enterprise Loadmap, ASH Analytics
- Visual Investigation of ASH dumps using R and ggplot

What is ASH?

- Active Session History (V$ACTIVE_SESSION_HISTORY)
- Session State Objects sampled every 1000 ms by MMNL
- ACTIVE (non-idle) session state data captured into circular memory buffer
- Latch-less and efficient, sampling is independent of session activity
- “Fix-up” mechanism updates recent samples with new data (TIME_WAITED not known when sampled session is waiting)

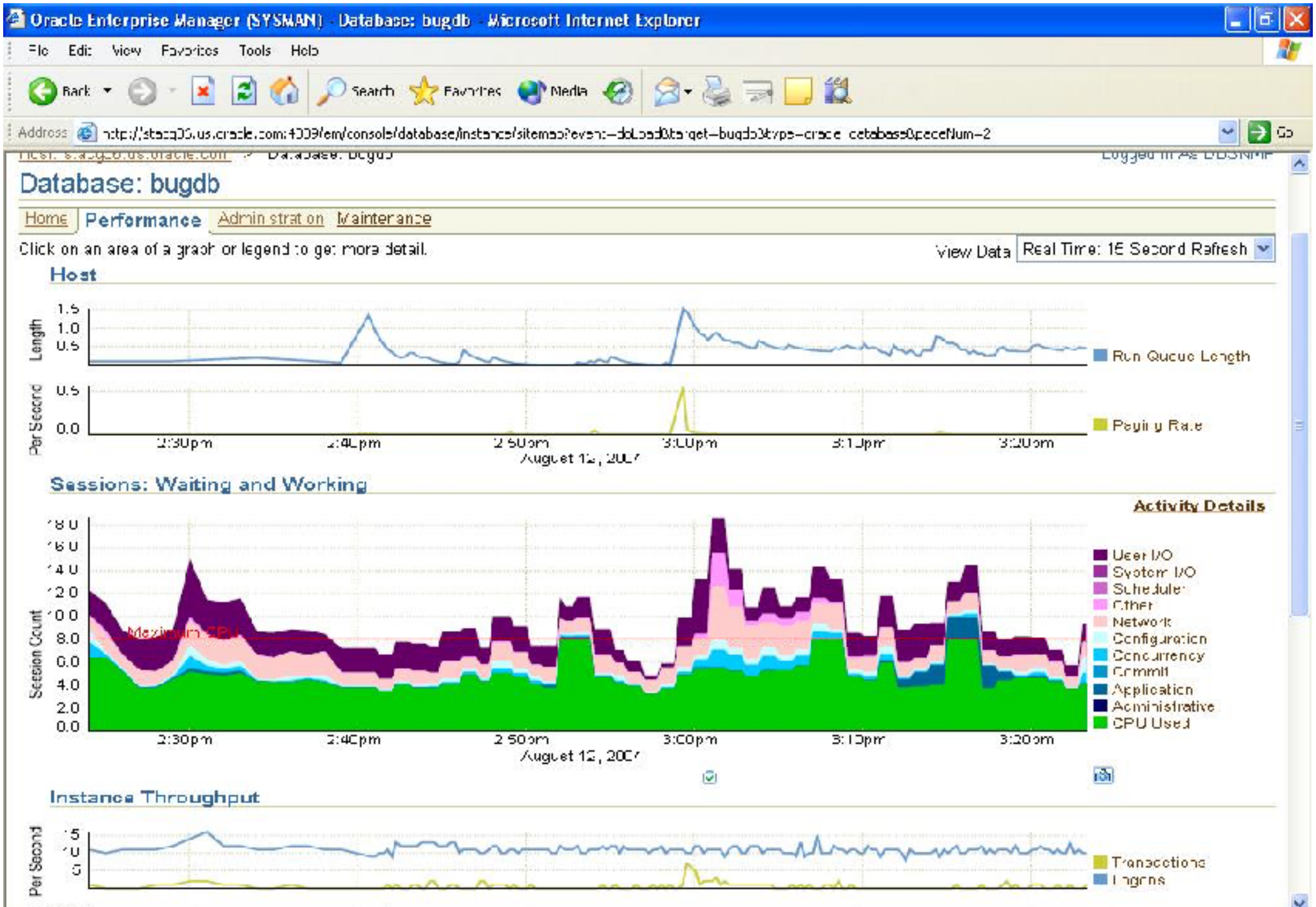
Standard usage: ASH Math

- **COUNT(*) = DB Time (seconds)**
- ASH is a FACT table where each row equals one second of session DB Time
- Use GROUP BY to break down DB Time by any of the many dimensions of ASH
- Aggregate ASH samples to Decompose DB Time over various dimension combinations

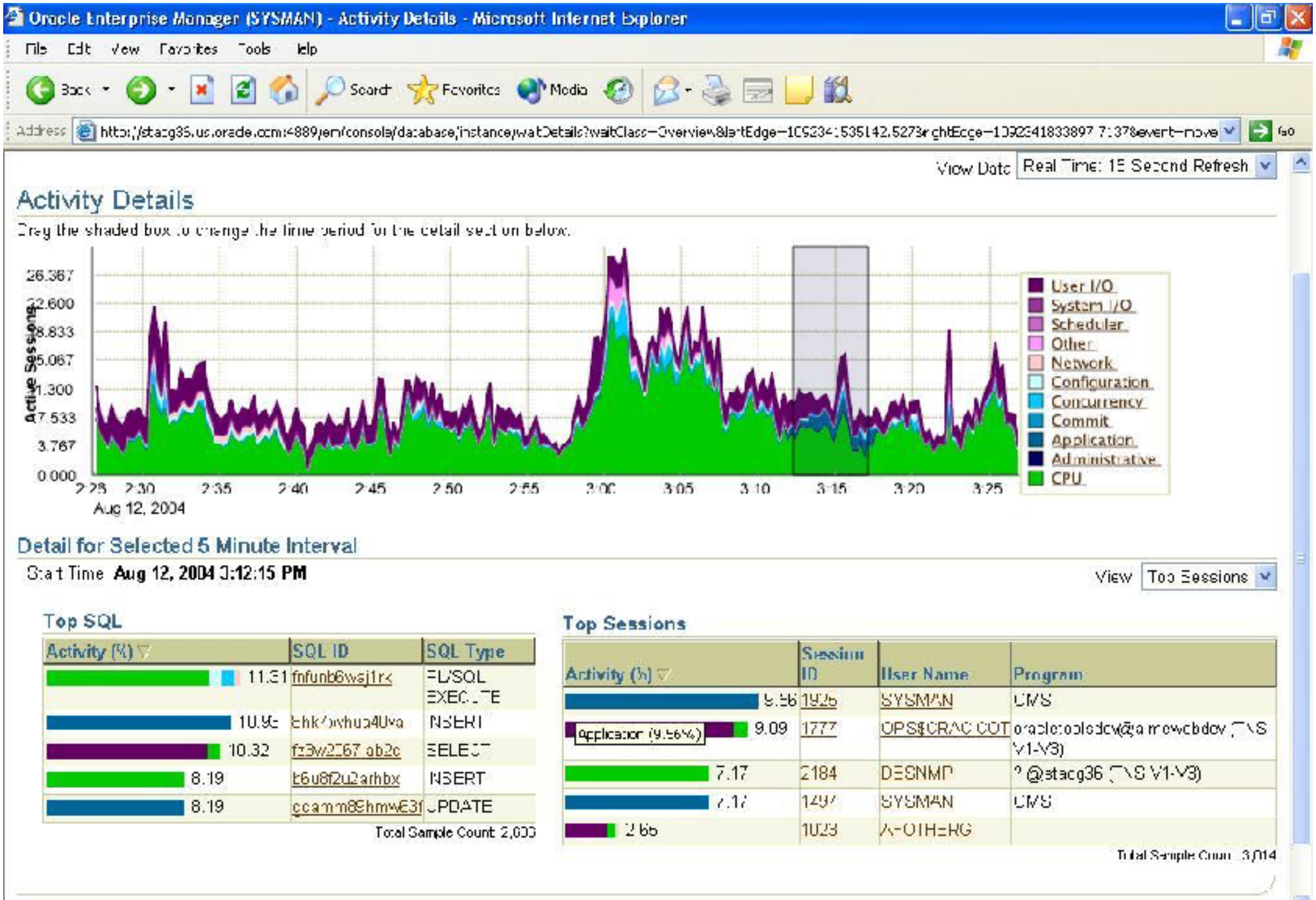
Visualizing ASH in EM

- Performance Page > Wait Class drill-down
- Top Activity Page
- ASH Analytics
- SQL Monitor

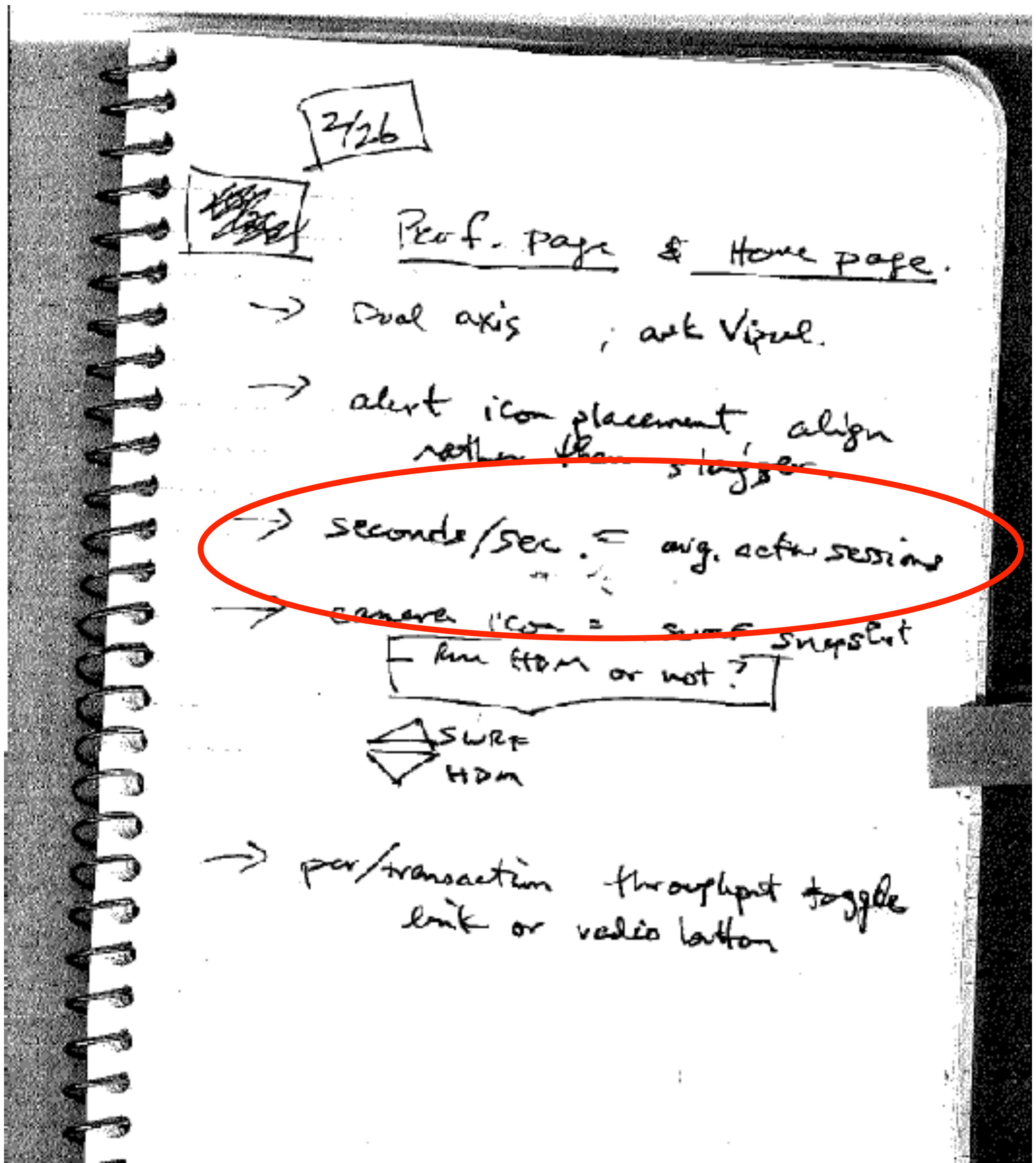
EM Performance Page



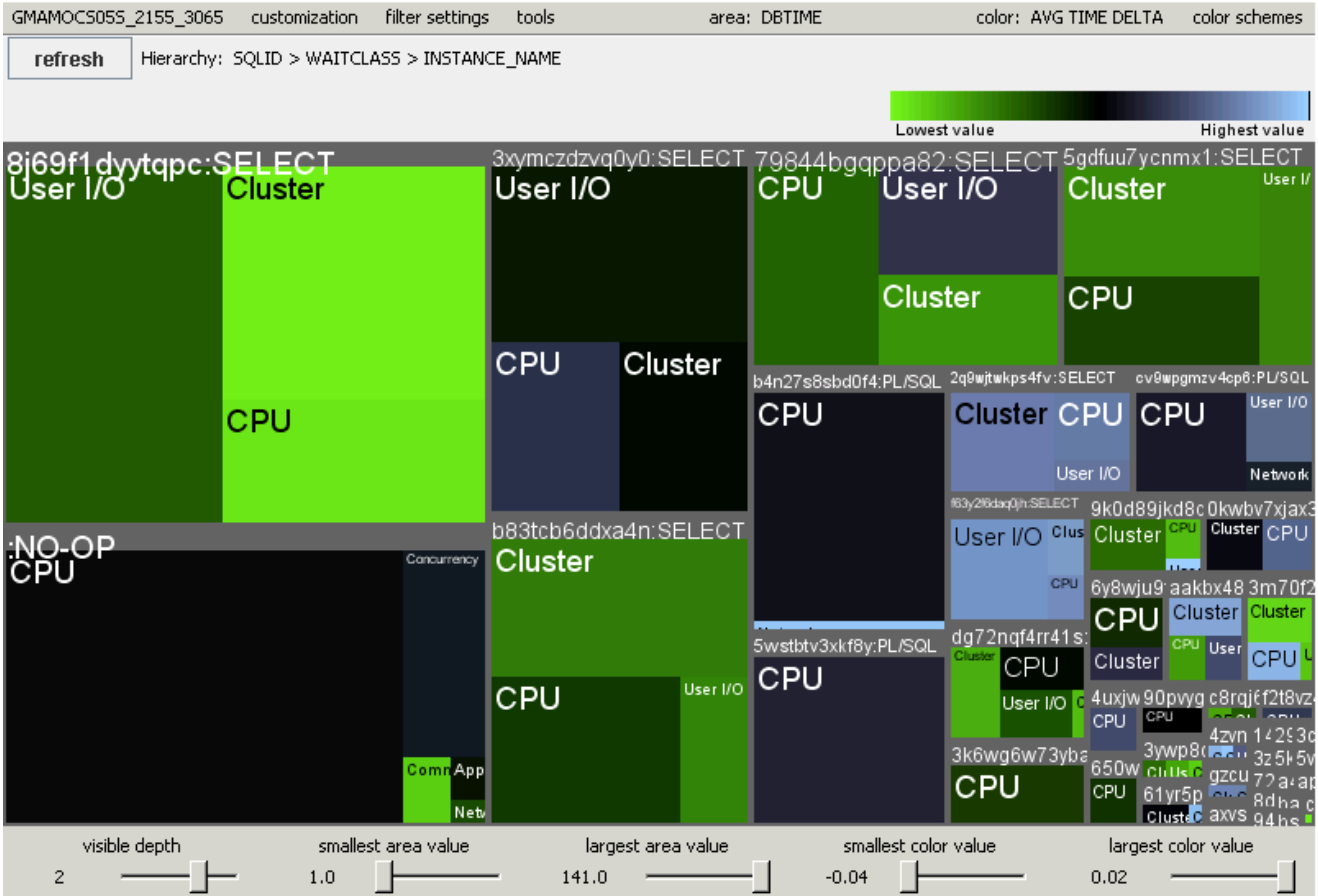
EM Top Activity - ASH

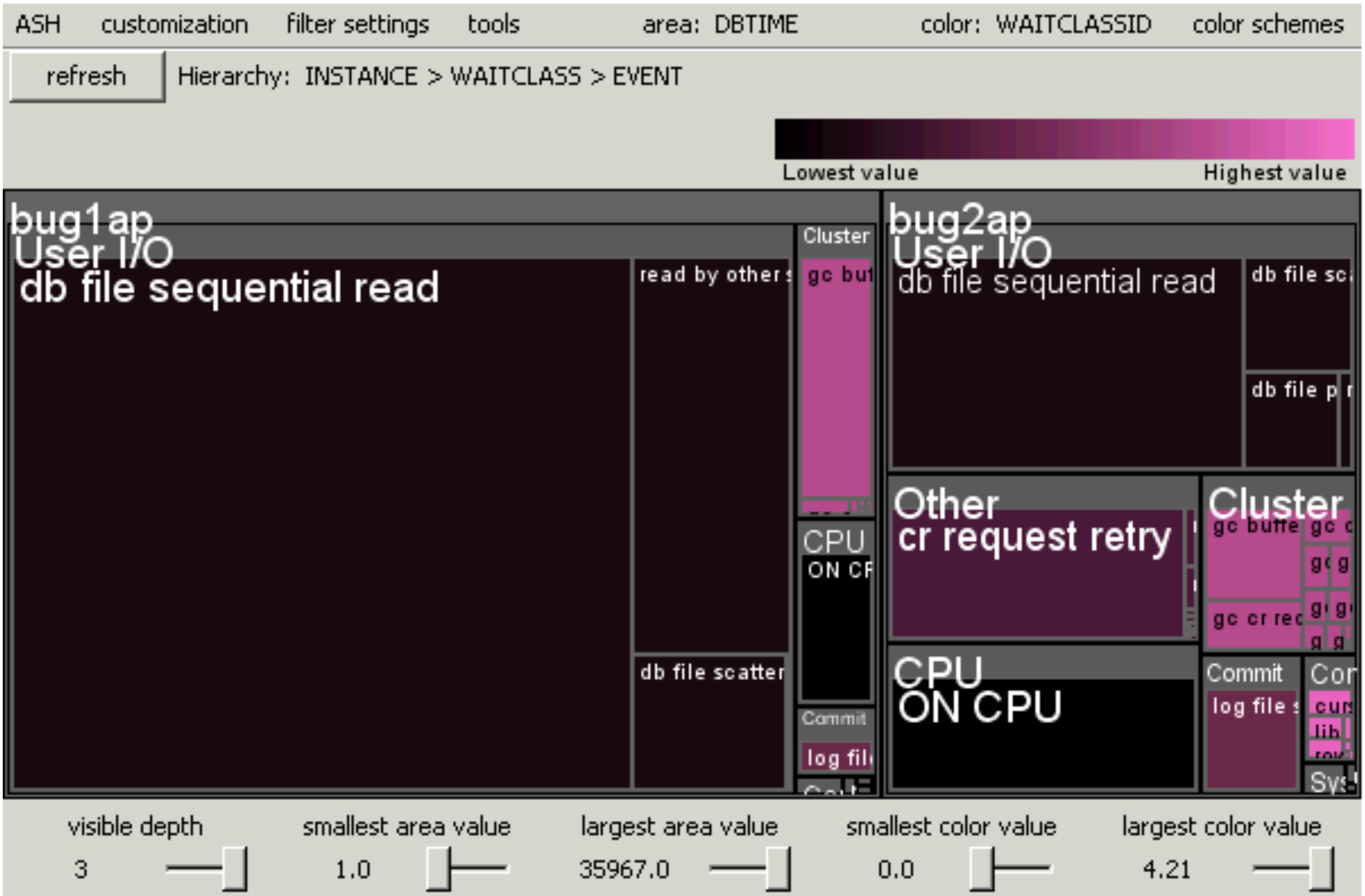


Average Active Sessions

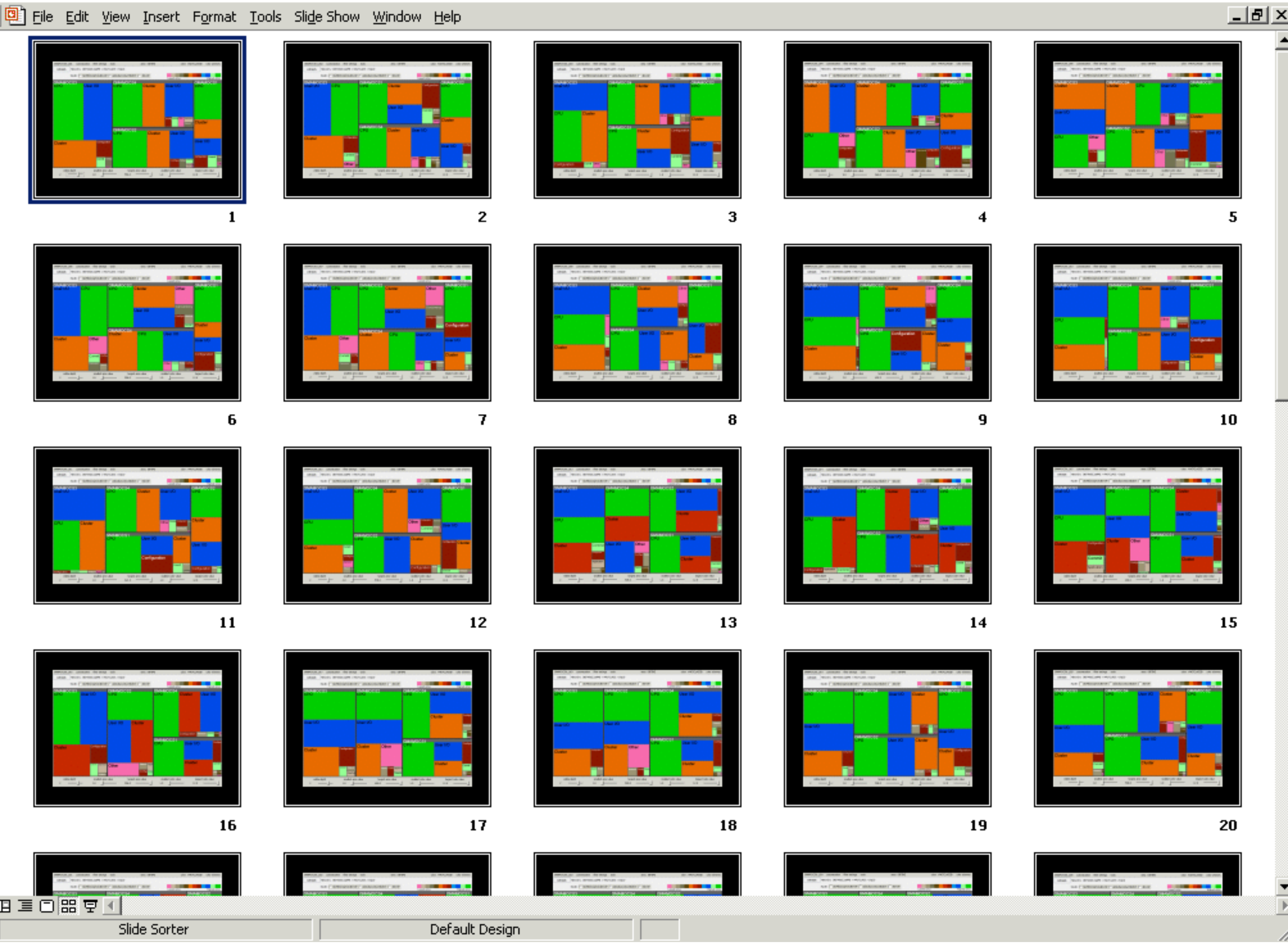


ASH Treemaps

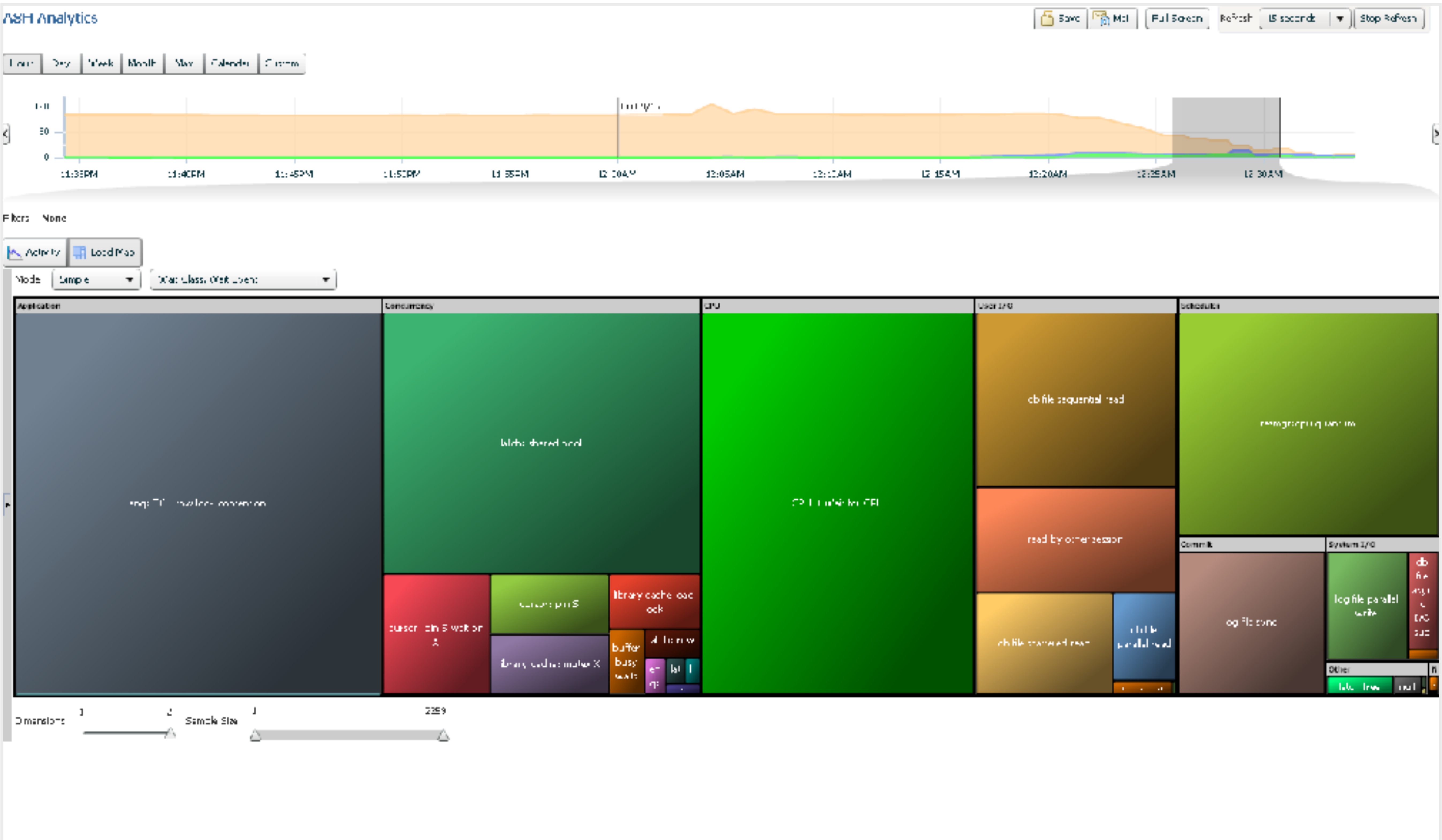

GMAMOCS055_2155_3065
customization
filter settings
tools
area: DBTIME
color: AVG TIME DELTA
color schemes
refresh
Hierarchy: SQLID > WAITCLASS > INSTANCE_NAME
Lowest value
Highest value
8j69f1dyytqpc:SELECT
User I/O
Cluster
CPU
3xymczdzvq0y0:SELECT
User I/O
CPU
Cluster
79844bgqppa82:SELECT
CPU
User I/O
Cluster
5gdfuu7ycnmx1:SELECT
Cluster
CPU
b4n27s8sbd0f4:PL/SQL
CPU
2q9wjtwkps4fv:SELECT
Cluster
CPU
User I/O
cv9wpgmzv4cp6:PL/SQL
CPU
User I/O
Network
:NO-OP
CPU
Concurrency
App
b83tcb6ddxa4n:SELECT
Cluster
CPU
User I/O
5wstbtv3xkf8y:PL/SQL
CPU
dg72nqf4rr41s
CPU
User I/O
3k6wg6w73yba
CPU
visible depth
2
smallest area value
1.0
largest area value
141.0
smallest color value
-0.04
largest color value
0.02



ASH
customization
filter settings
tools
area: DBTIME
color: WAITCLASSID
color schemes
refresh
Hierarchy: INSTANCE > WAITCLASS > EVENT
Lowest value
Highest value
bug1ap
User I/O
db file sequential read
read by other s
db file scatter
Cluster
gc buf
CPU
ON CP
Commit
log fil
bug2ap
User I/O
db file sequential read
db file sc
db file p
Other
cr request retry
Cluster
gc buffe
gc cr req
CPU
ON CPU
Commit
log file s
Con
cur
lib
visible depth
3
smallest area value
1.0
largest area value
35967.0
smallest color value
0.0
largest color value
4.21


File Edit View Insert Format Tools Slide Show Window Help
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Slide Sorter
Default Design

EM ASH Analytics



Exploring ASH dumps visually using R and ggplot

Tools and raw data

- Tools: Anaconda, R, RStudio, ggplot2, dplyr (Hadley Wickham's stuff)
- Data: some old 10g ASH dumps
  - 10.2.0.1 Enterprise Edition
  - 4-node RAC ASH dump > 4 separate trace files

Data import and preparation

- Getting started was easy:

```
ashDF <- read.csv(ashfilepath,header=FALSE)
```

- HOWEVER: real analysis required significant data munging
  - multiple trace files per dump
  - join in wait event and wait class names
  - import microsecond sample times
  - add state boolean: time_waited > 0 = WAIT else CPU
  - add “CPU” wait-class

DB Time over Time (AAS)

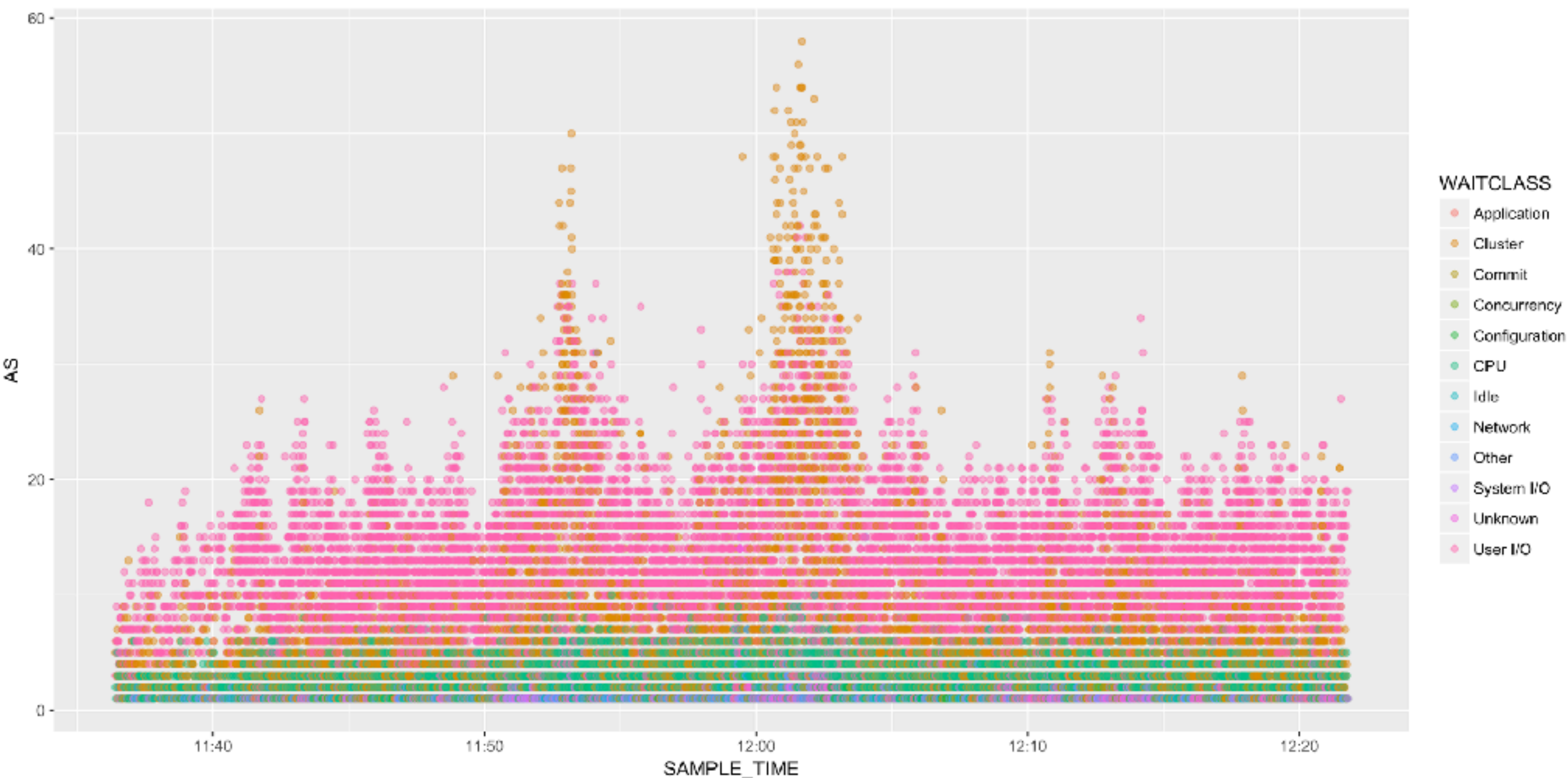
What and why?

- Standard usage:

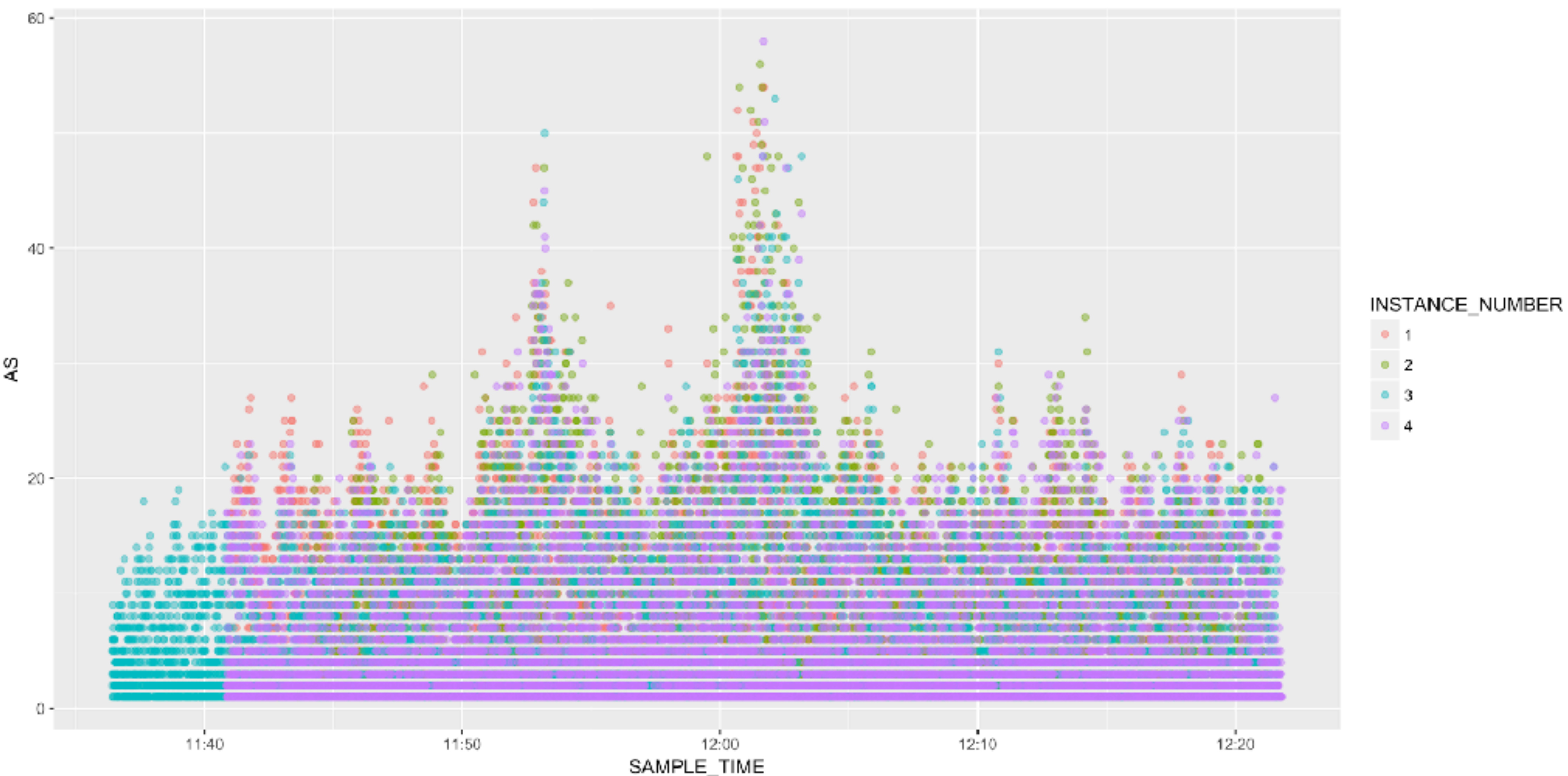
  Aggregate ASH samples into Average Active Sessions

- Investigate consistency across RAC instances
- Investigate sampler independence, sample id consistency?
- Compute AAS over wait classes by 1-minute intervals

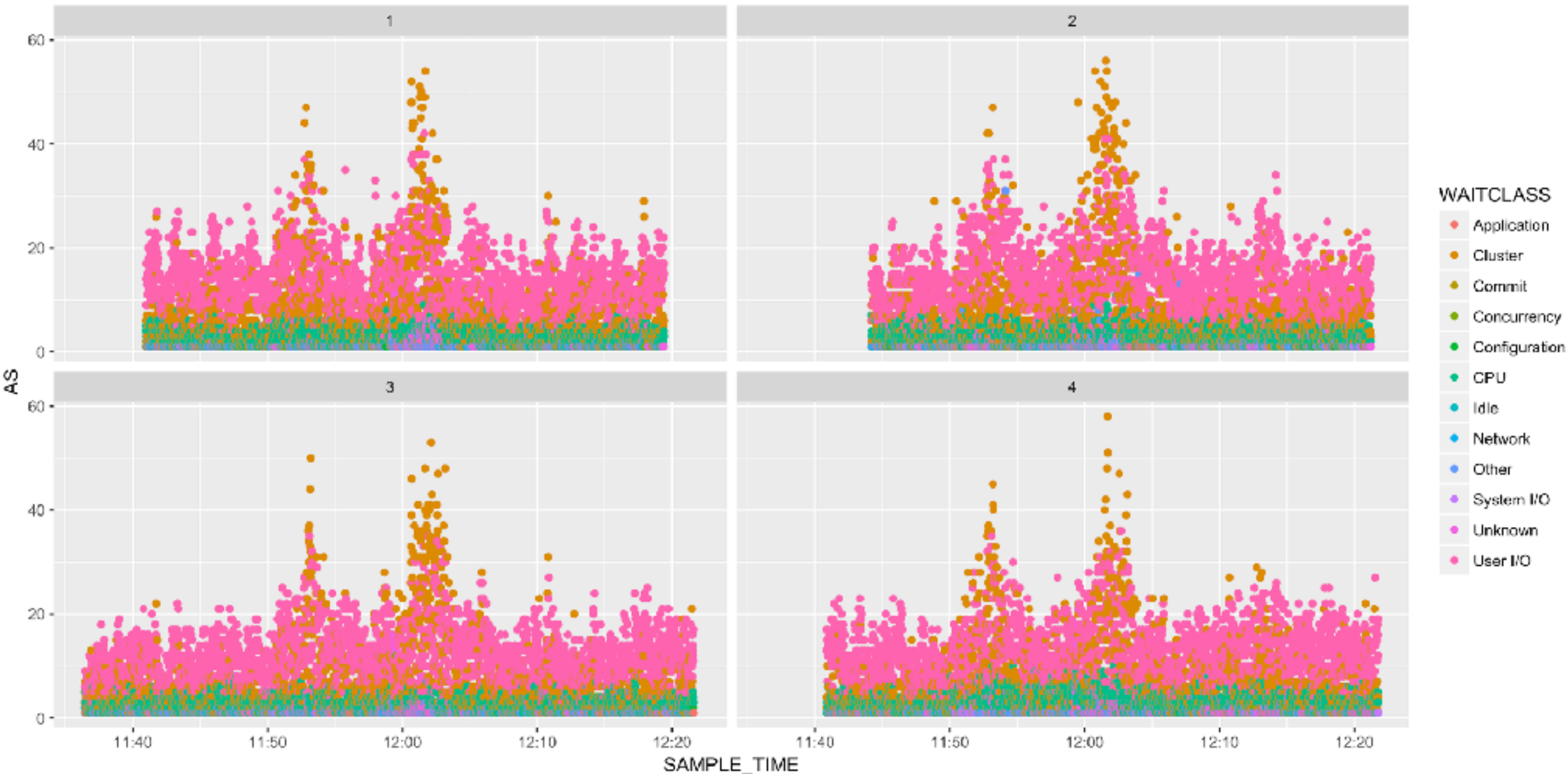
x = sample time, color wait class



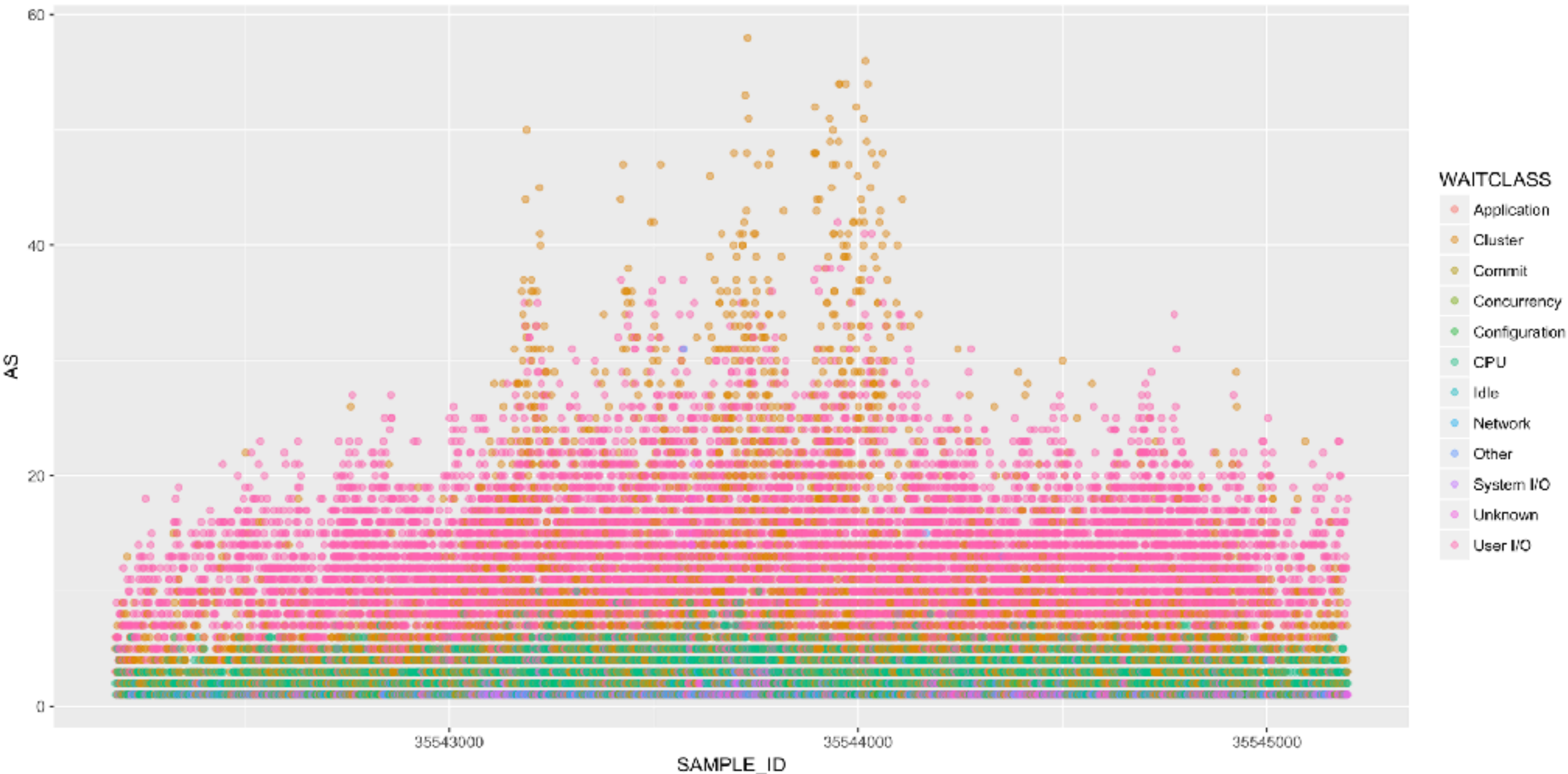
color by instance



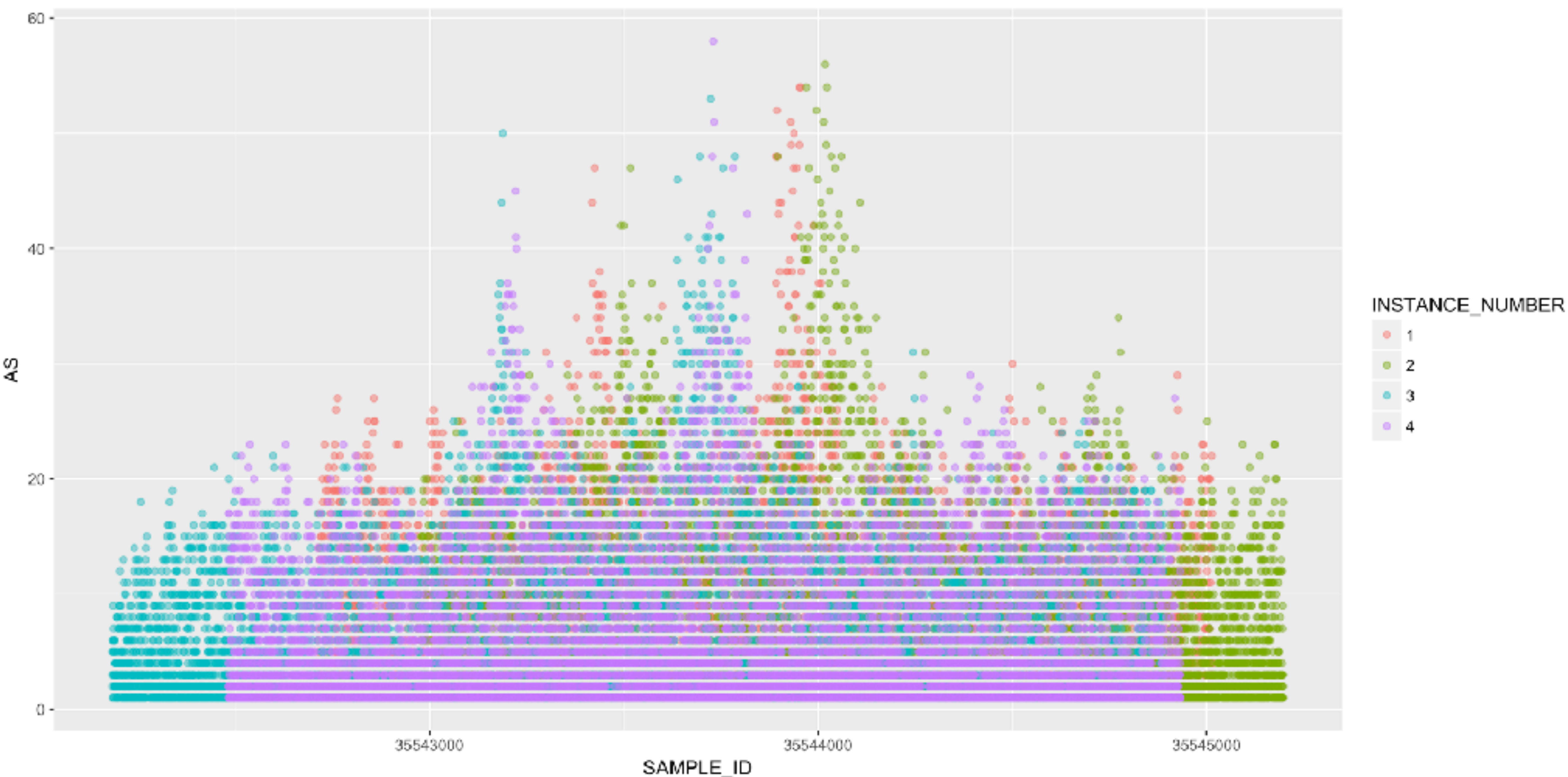
color wait class, facet instance



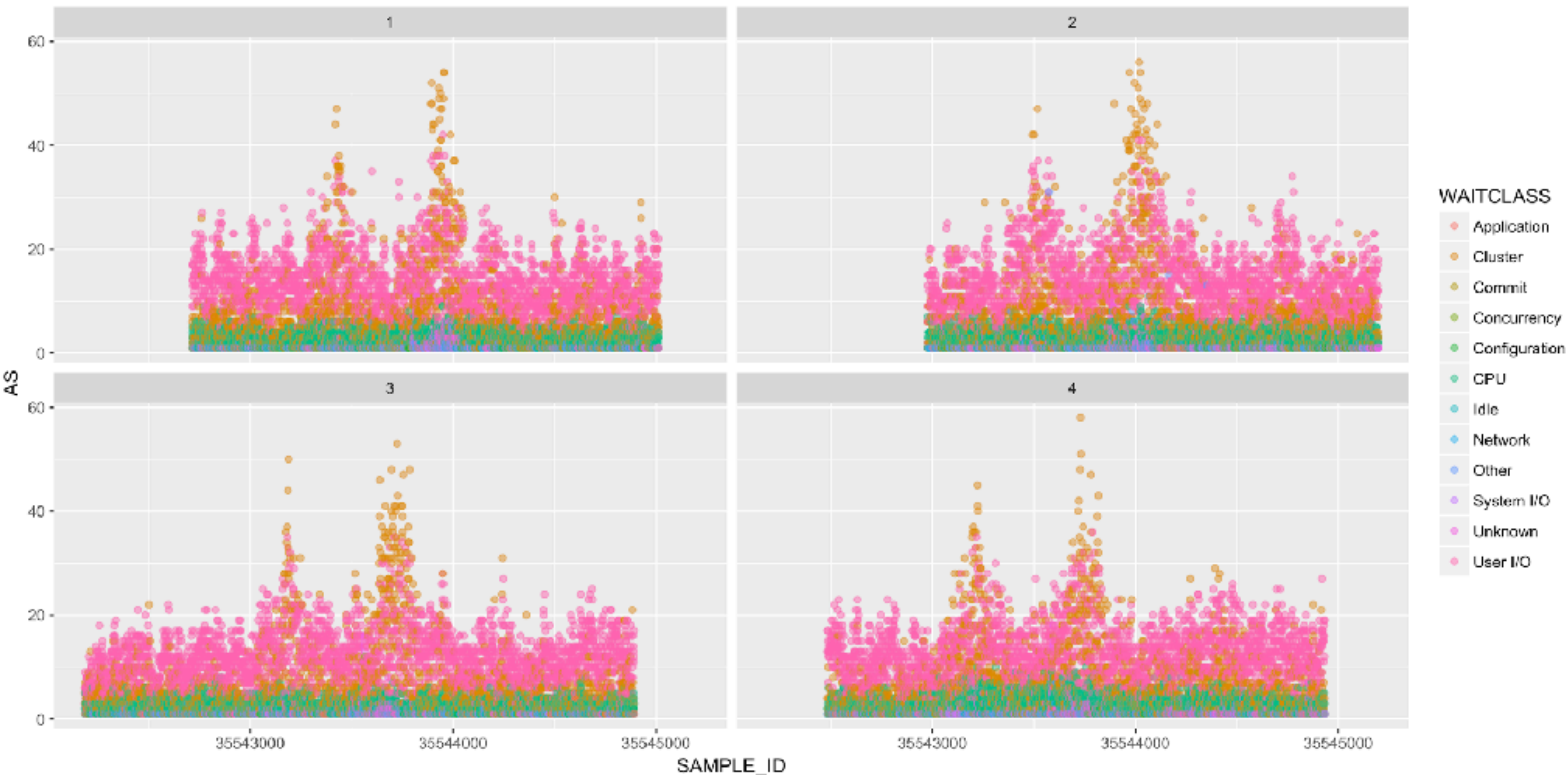
x = sample id, color wait class



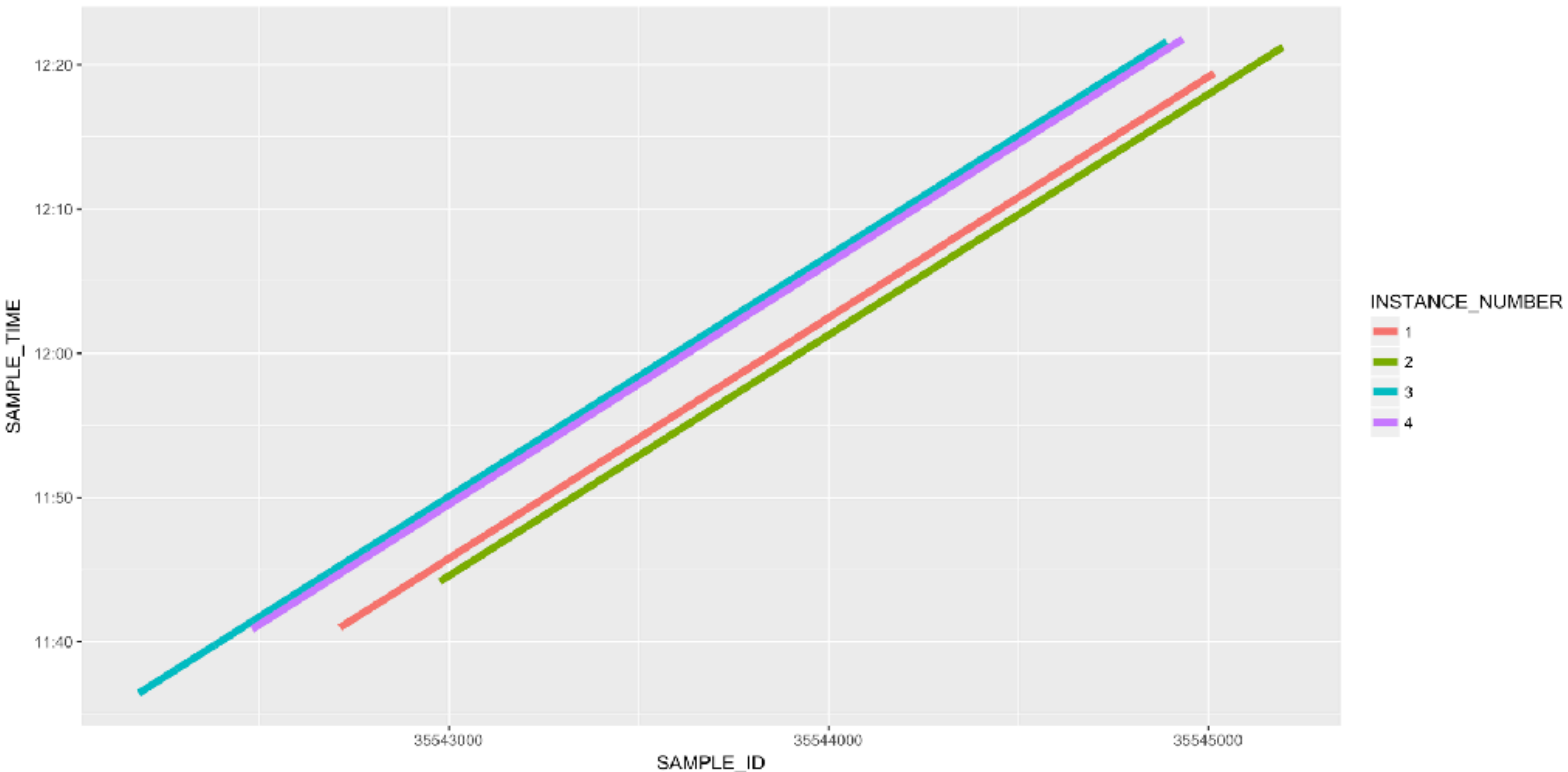
color by instance, hmmm



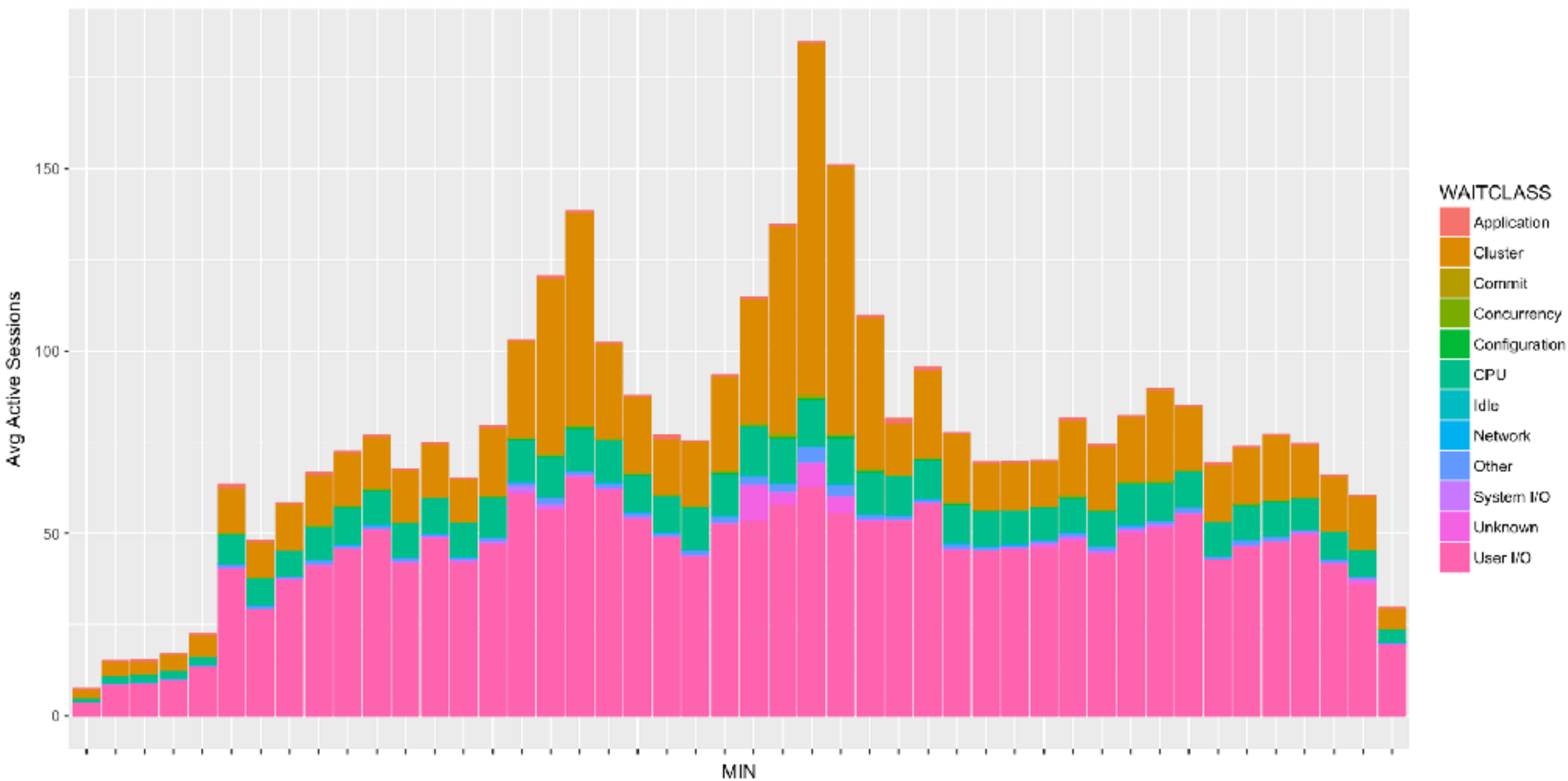
facet instance, color wait class



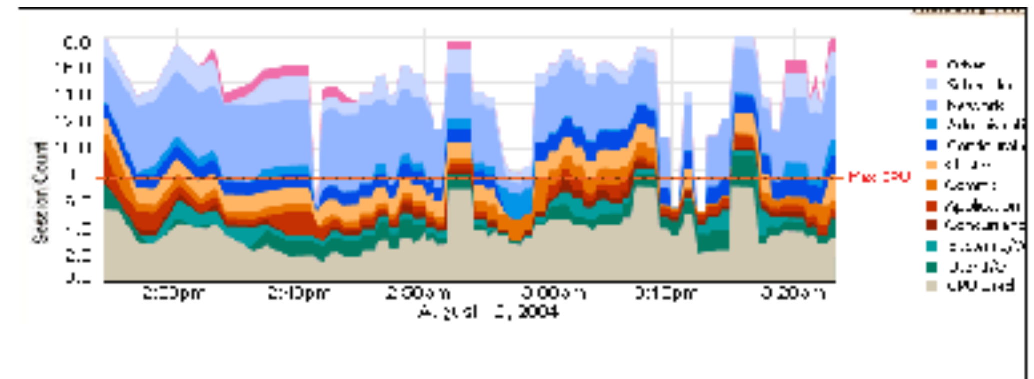
sample id vs. sample time

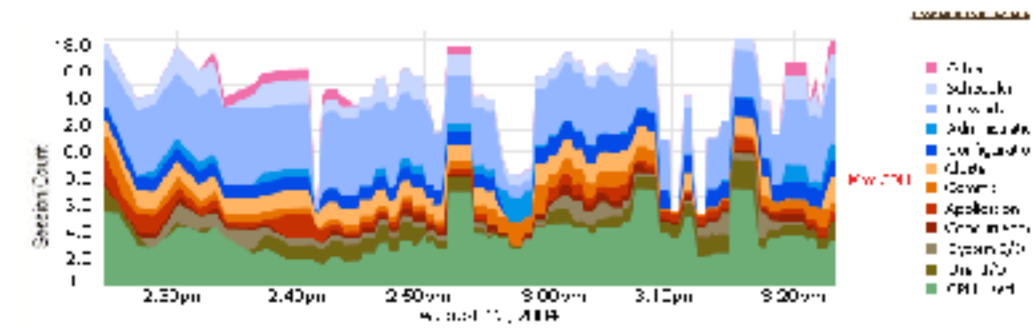


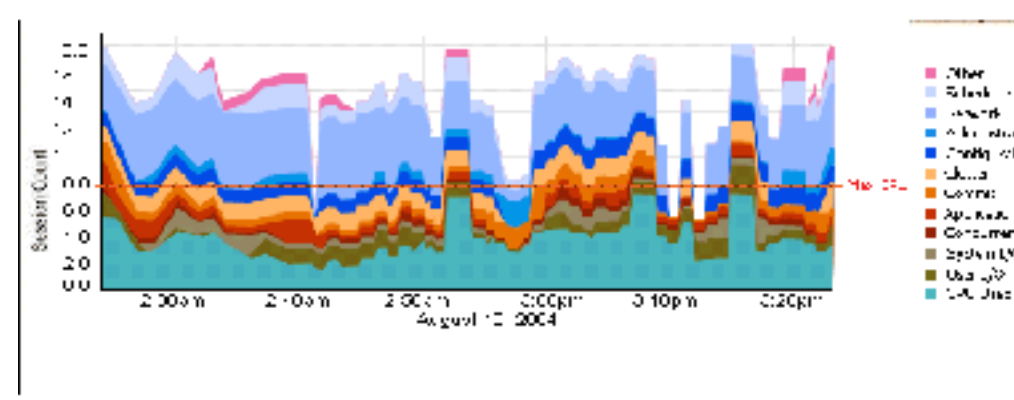
AAS by minute, color wait class

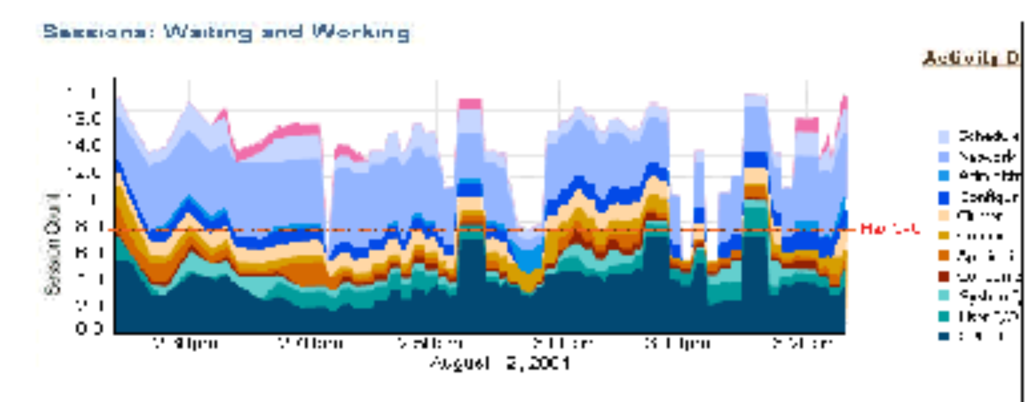


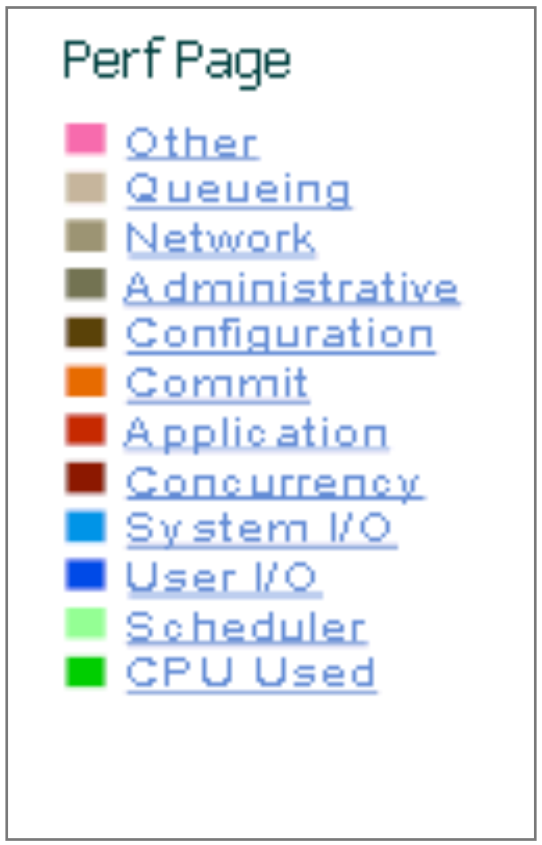
Wait class colors











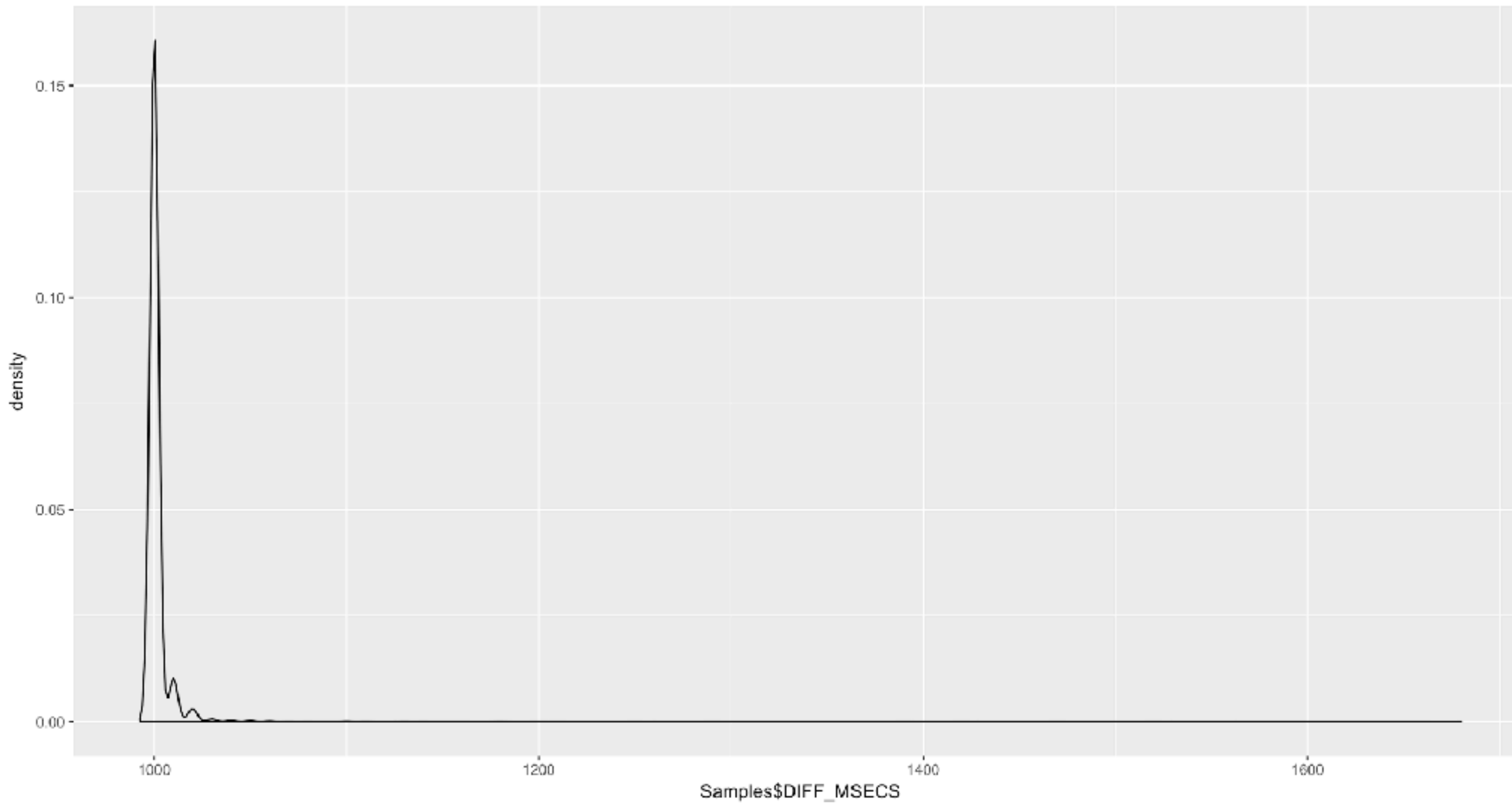
It took significant effort to finalize the wait class color scheme.

ASH Sampler timing

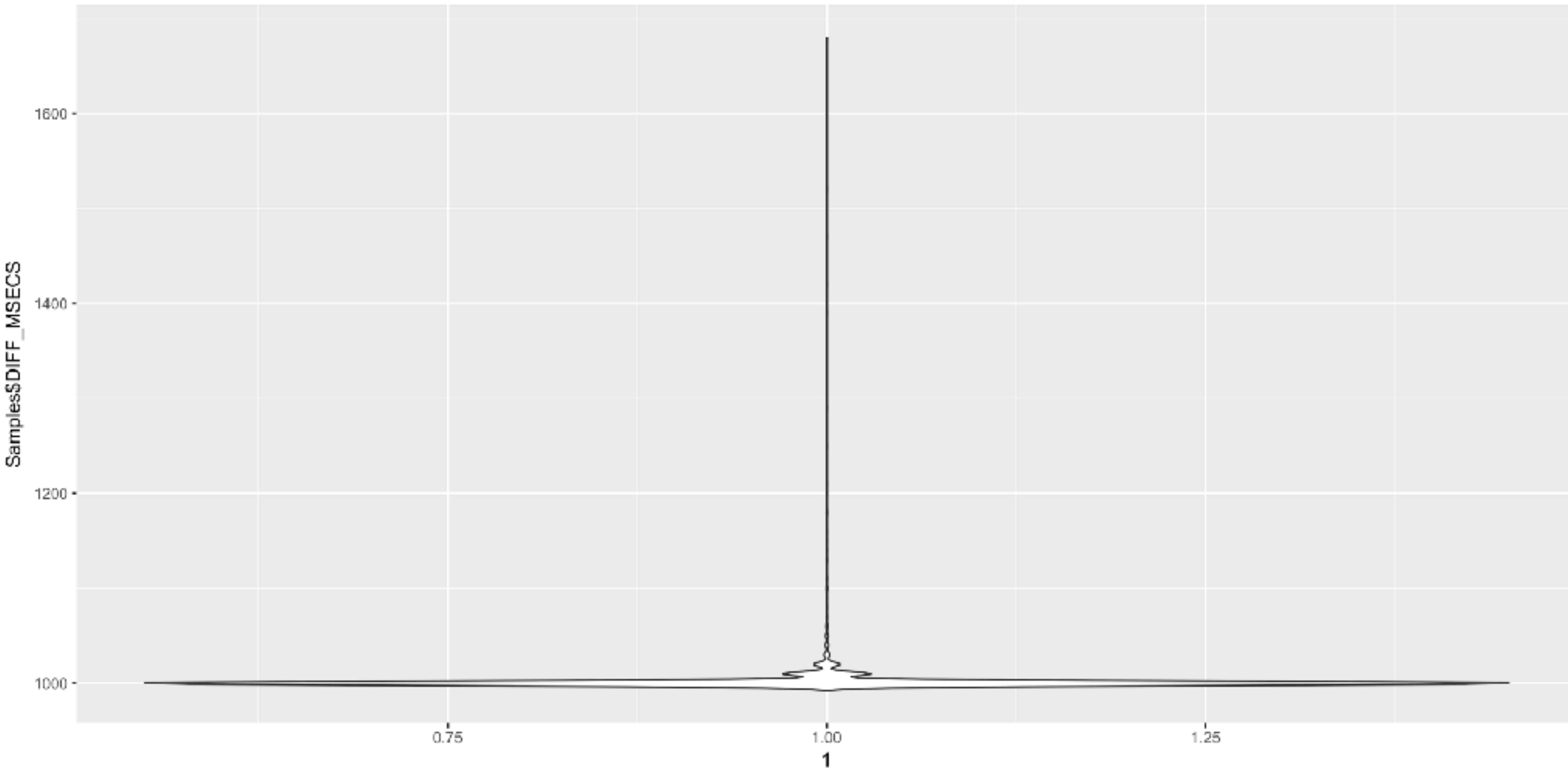
What and why?

- Micro-second sample times in ASH dump data
- Does the sampler do well at keeping to fixed interval?
- Use inter-sample time diffs to analyze sampler consistency
- diff_msecs = sample_time - lag(sample_time)
  expressed in microseconds, grouped by instance number

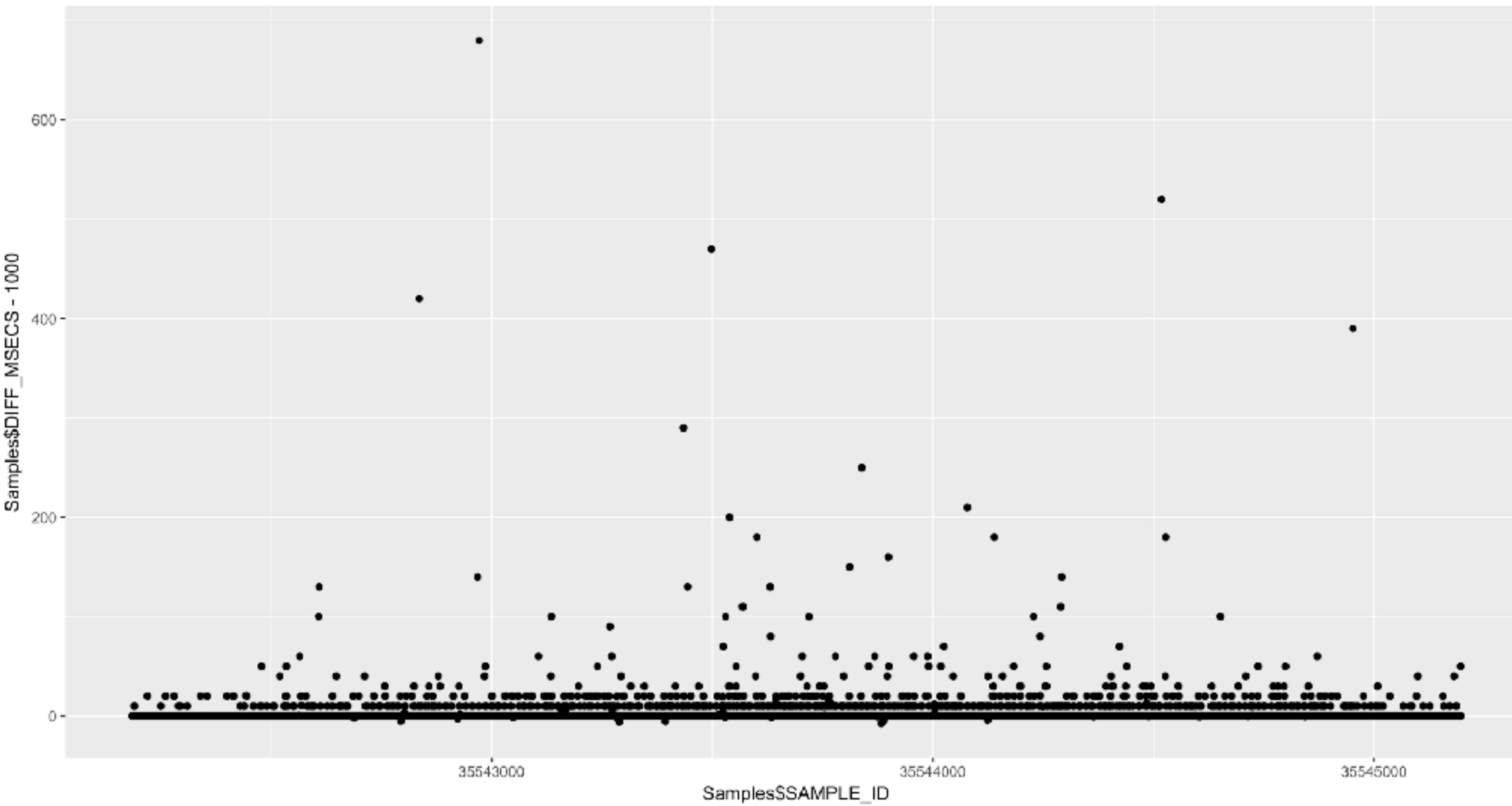
density plot of diff_msecs



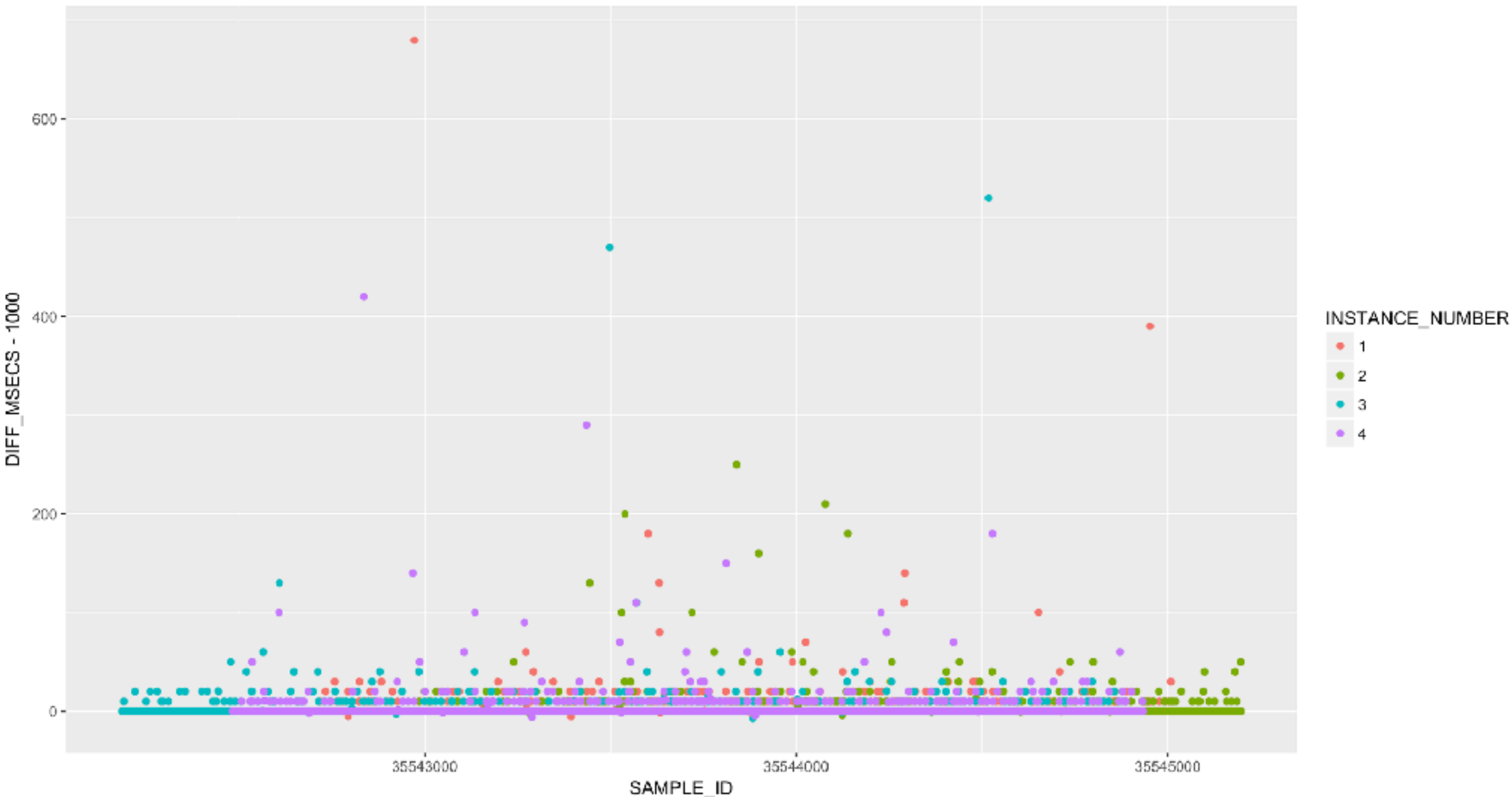
violin plot of diff_msecs



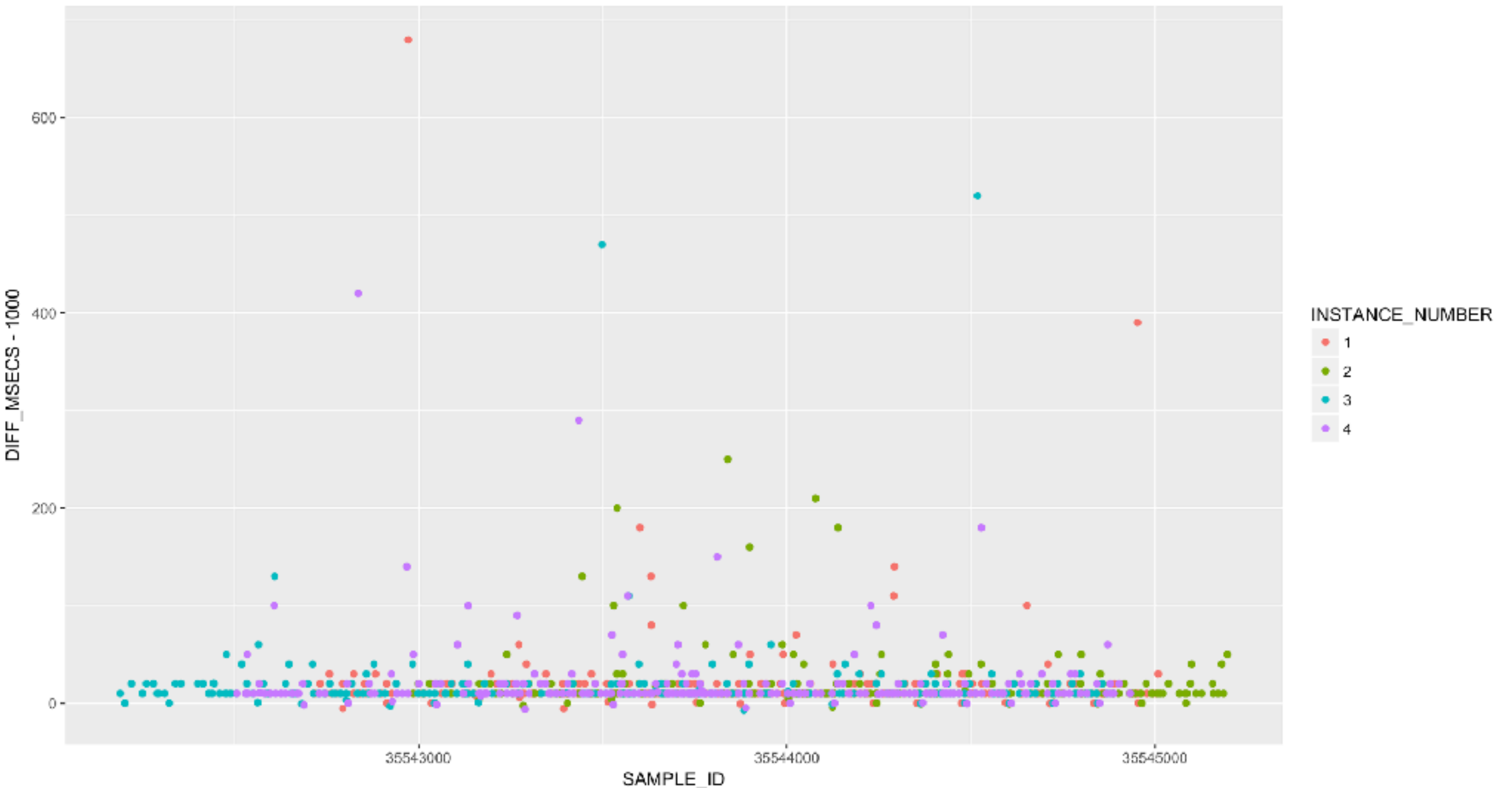
plot diff msec - 1000



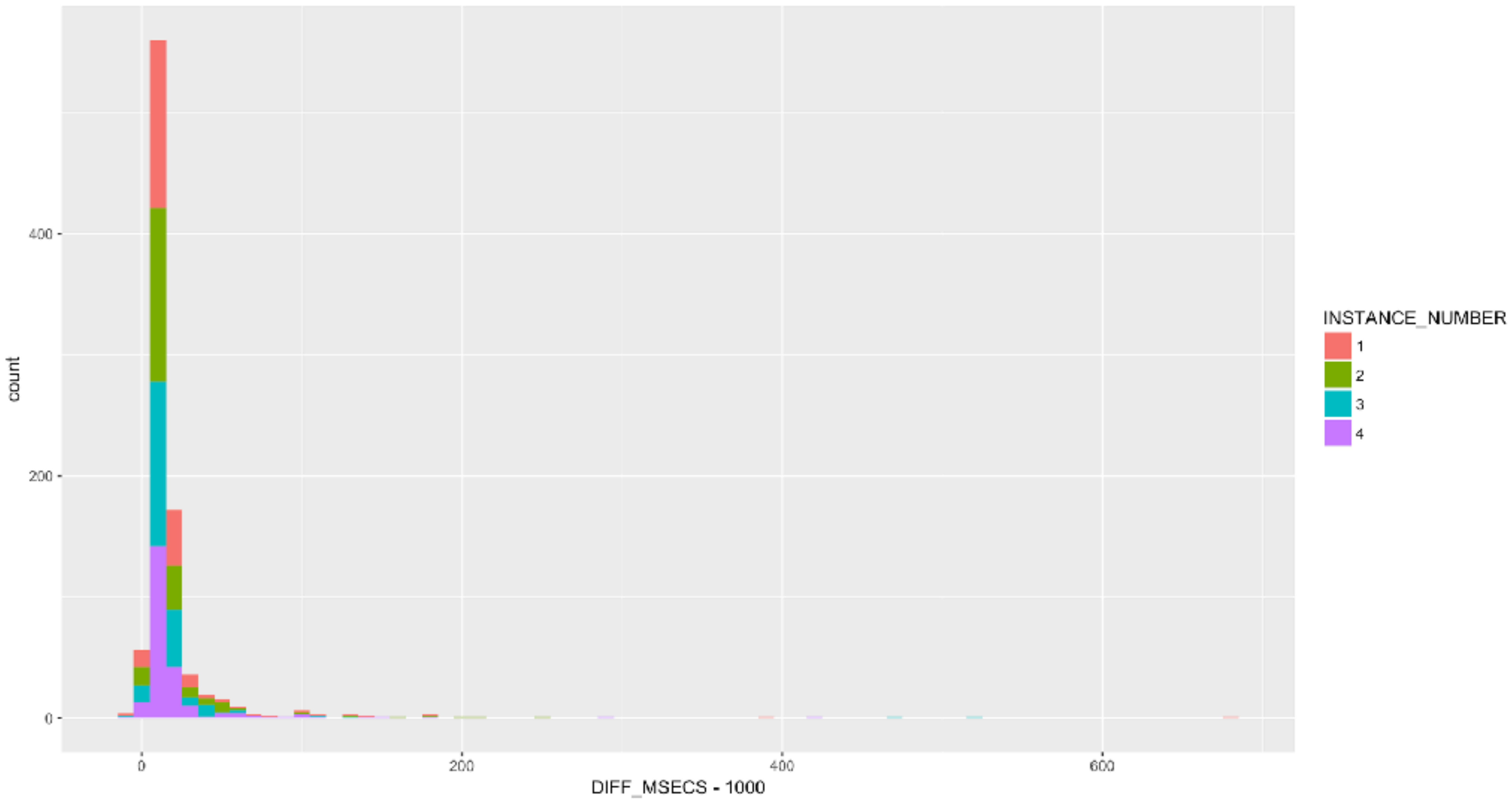
color by instance



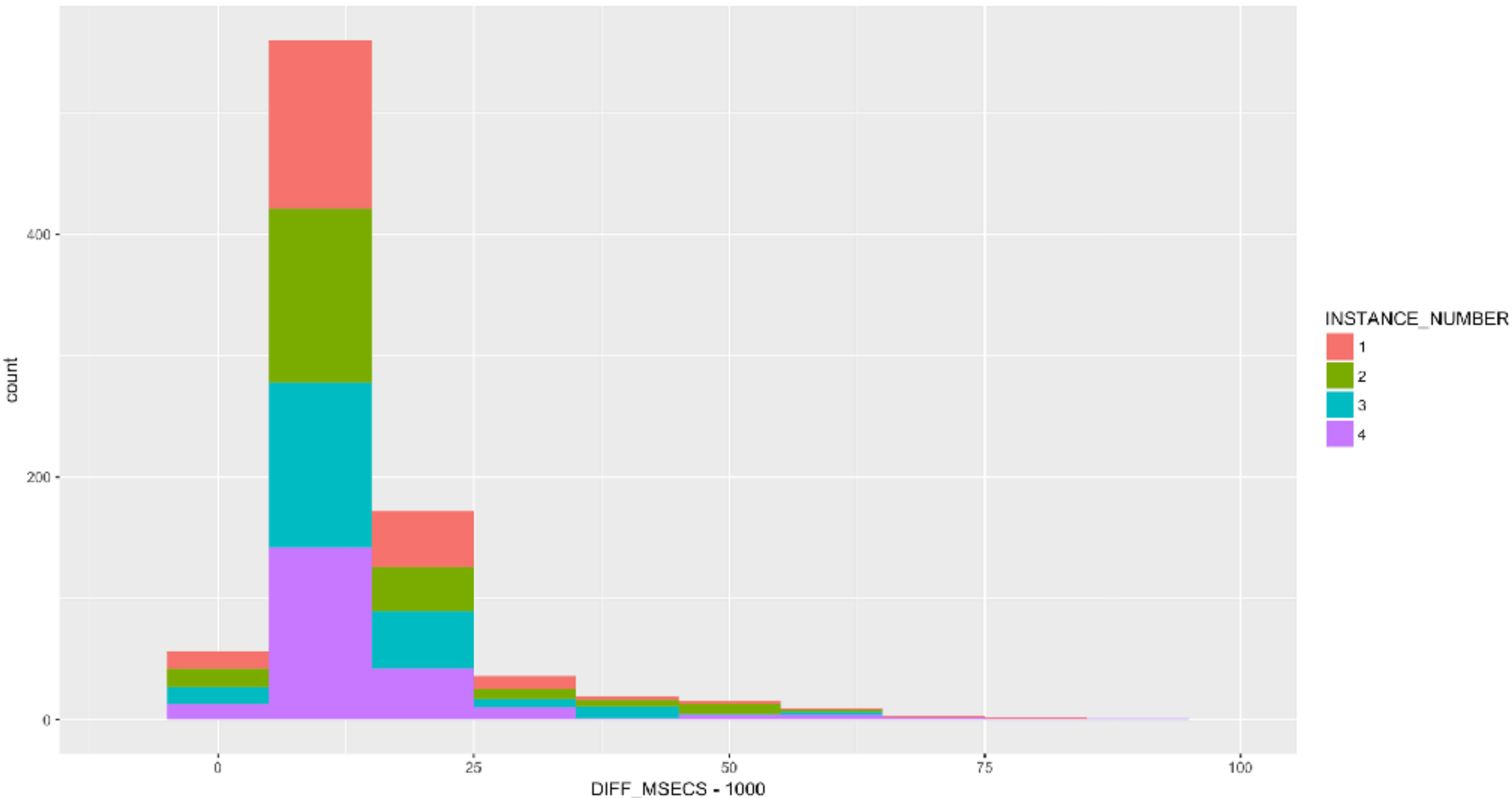
filter “good” ones out



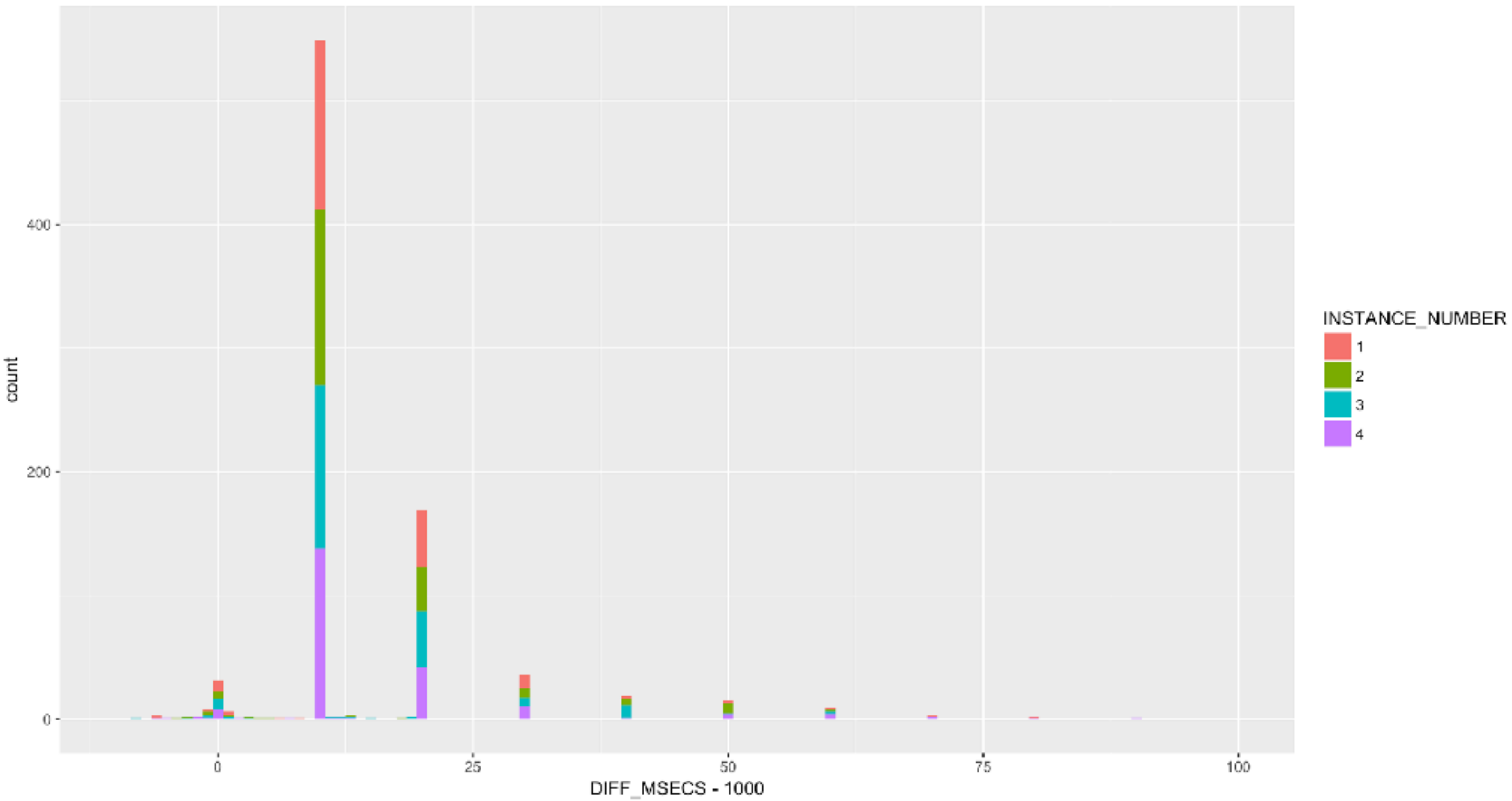
skewed distribution with long tail



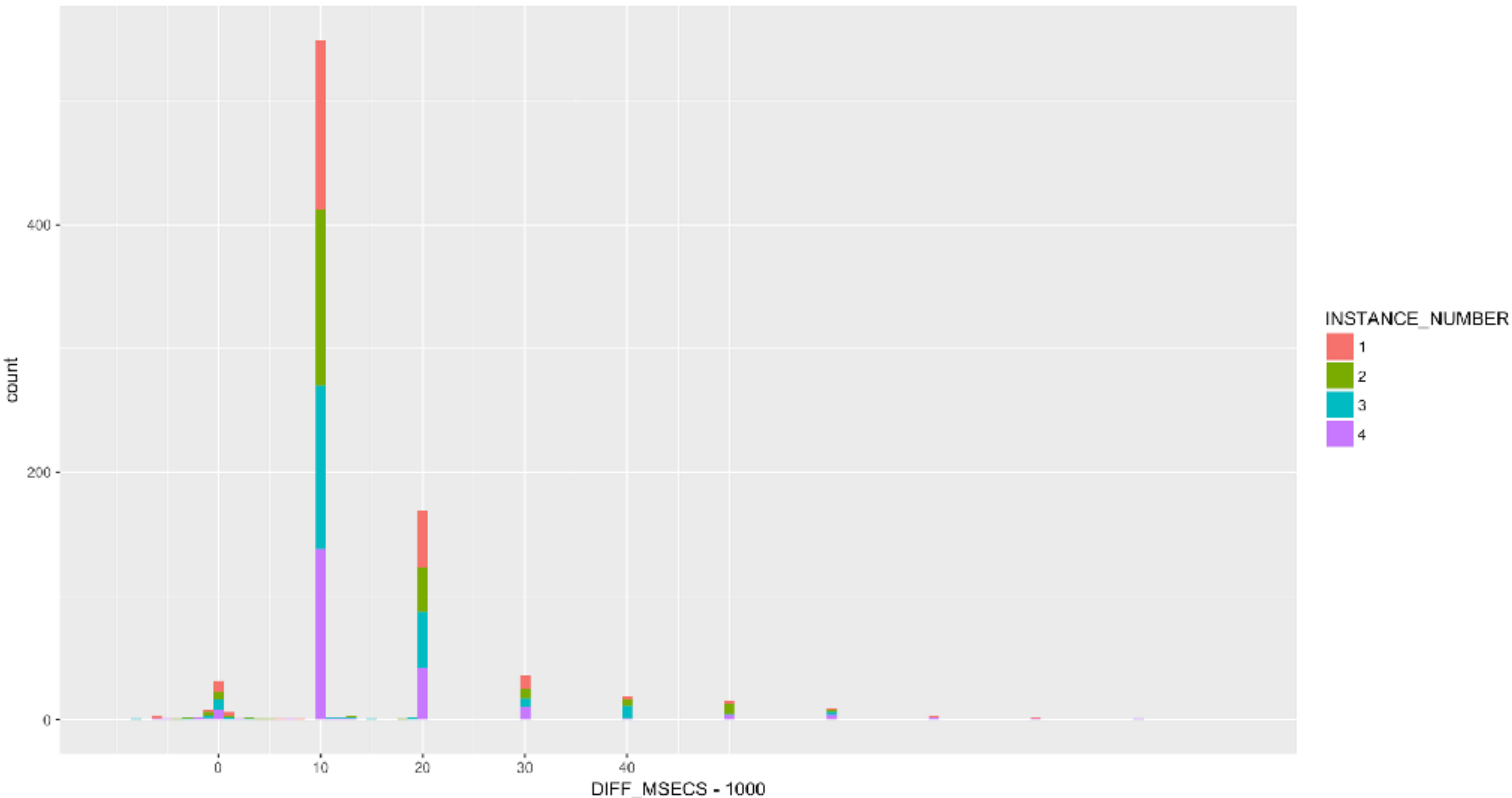
zoom in on the action



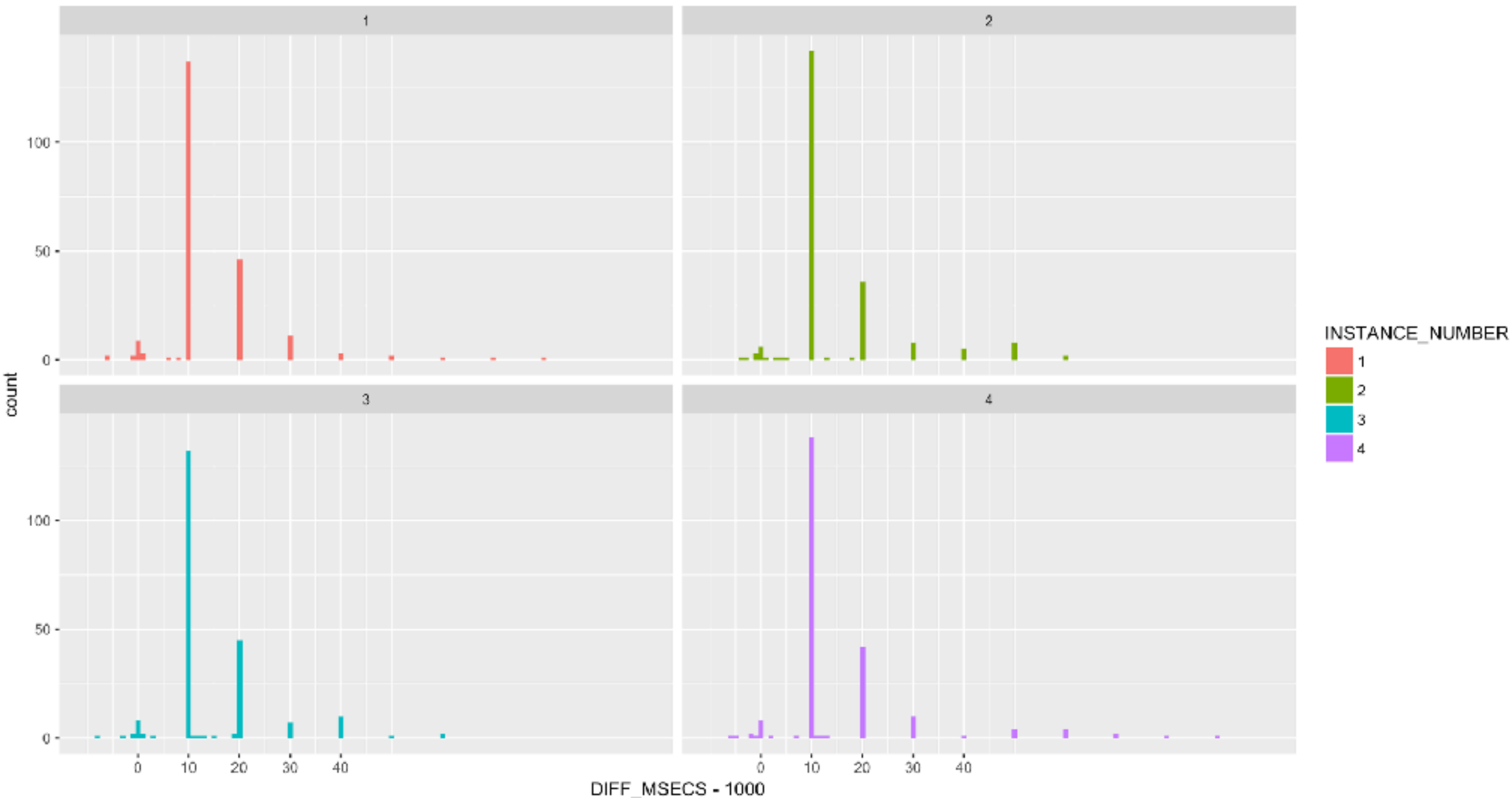
diffs in 10 ms intervals?



ha! (serialization, scheduler?)

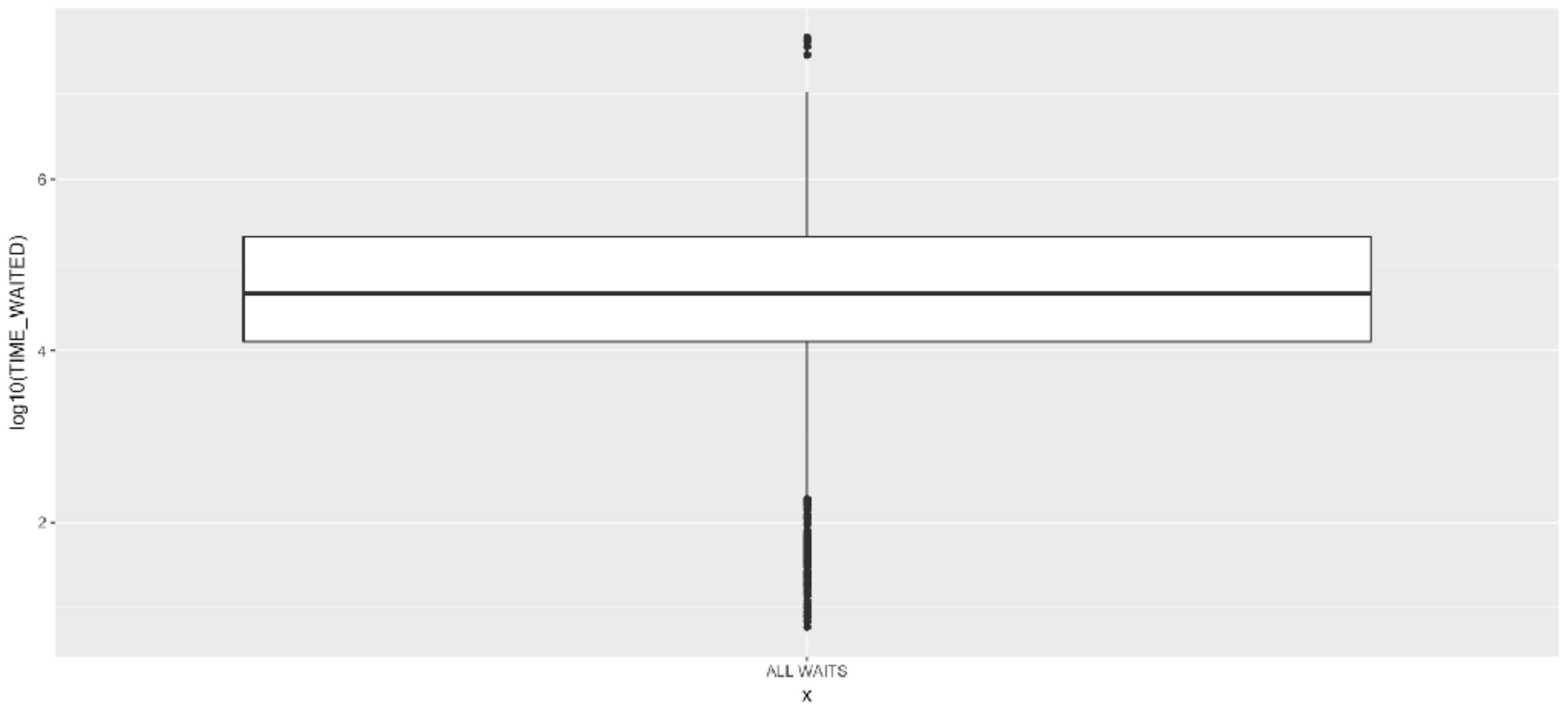


consistency across samplers

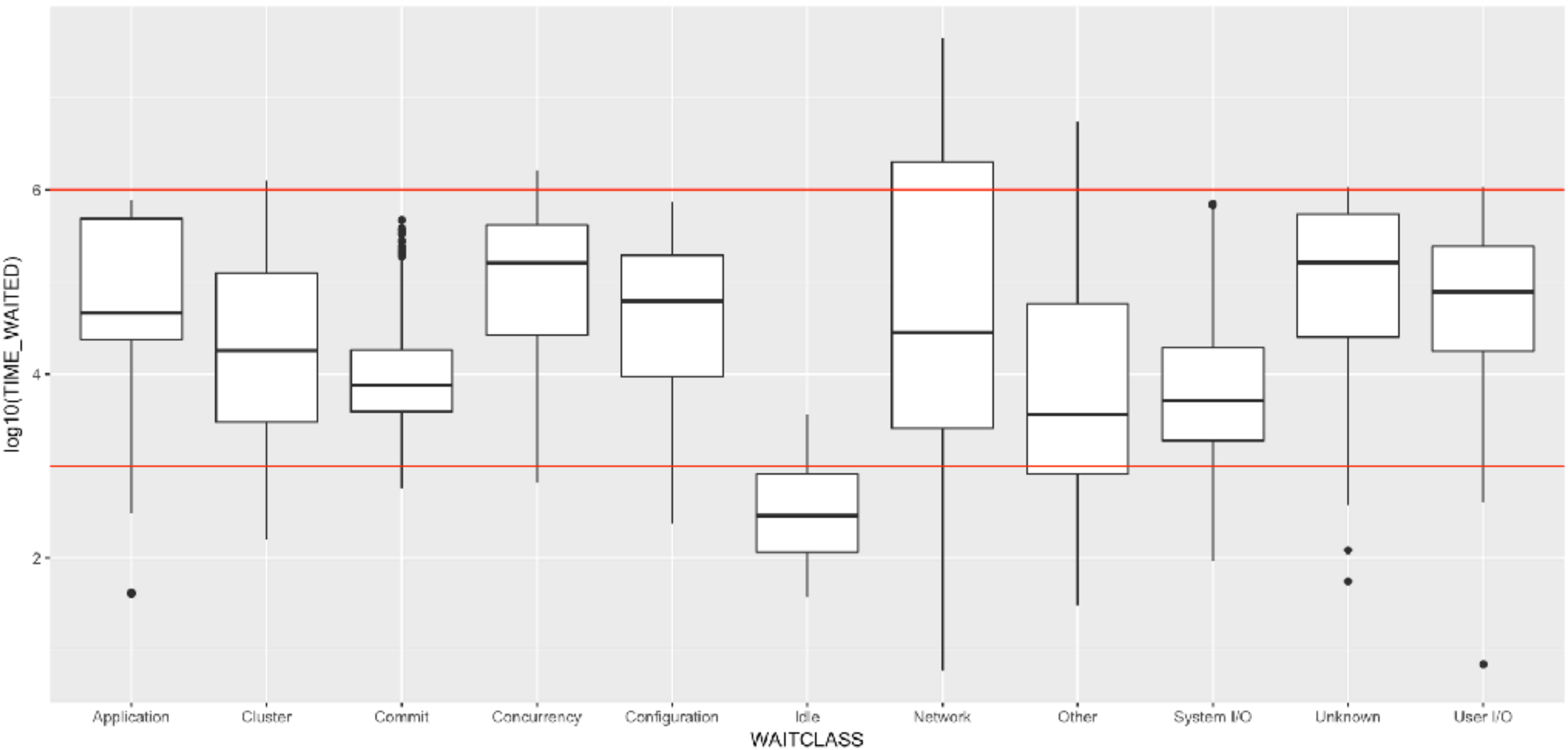


See what we got:
observed event counts

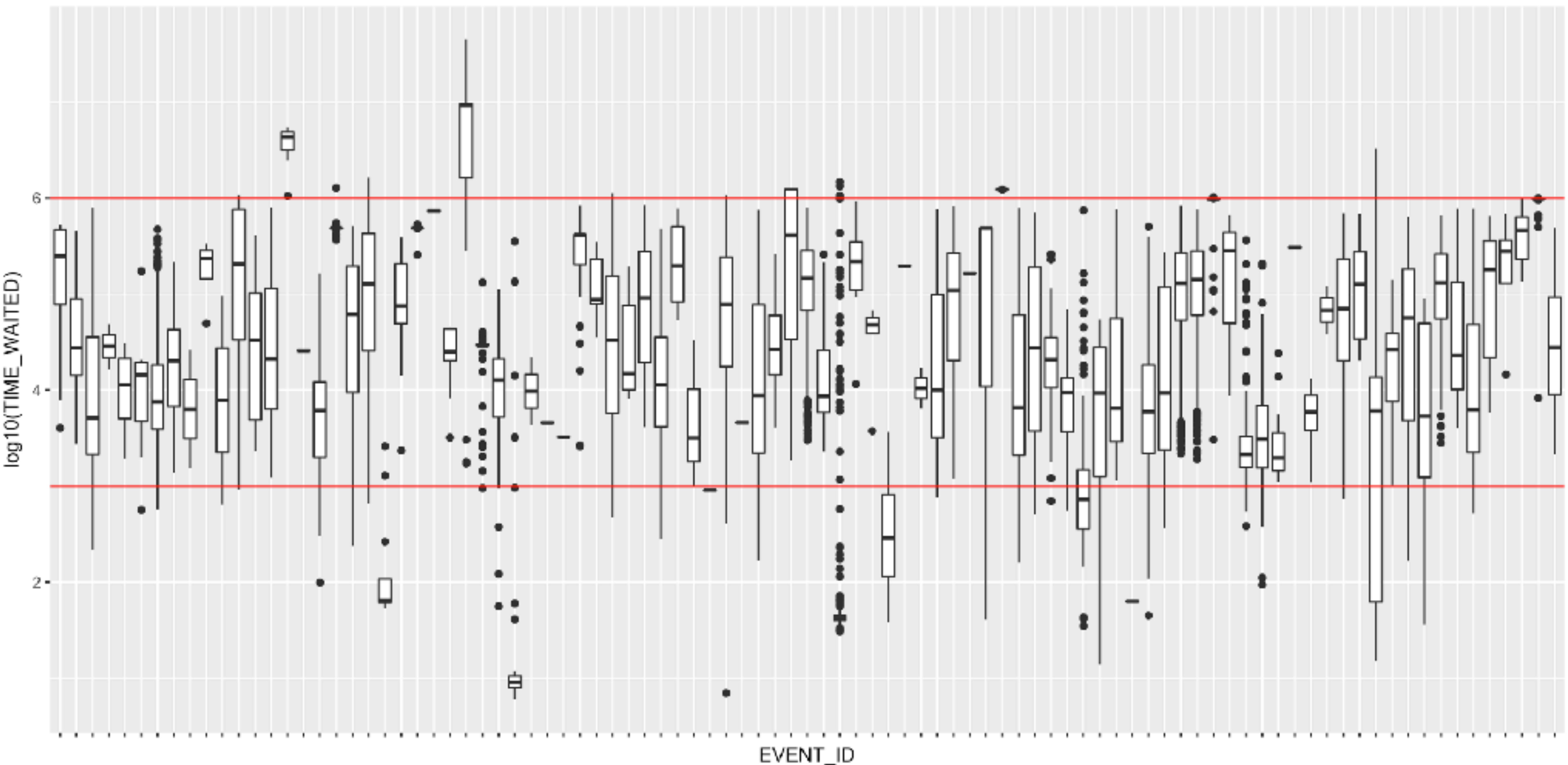
Box plot of log10(TIME_WAITED)



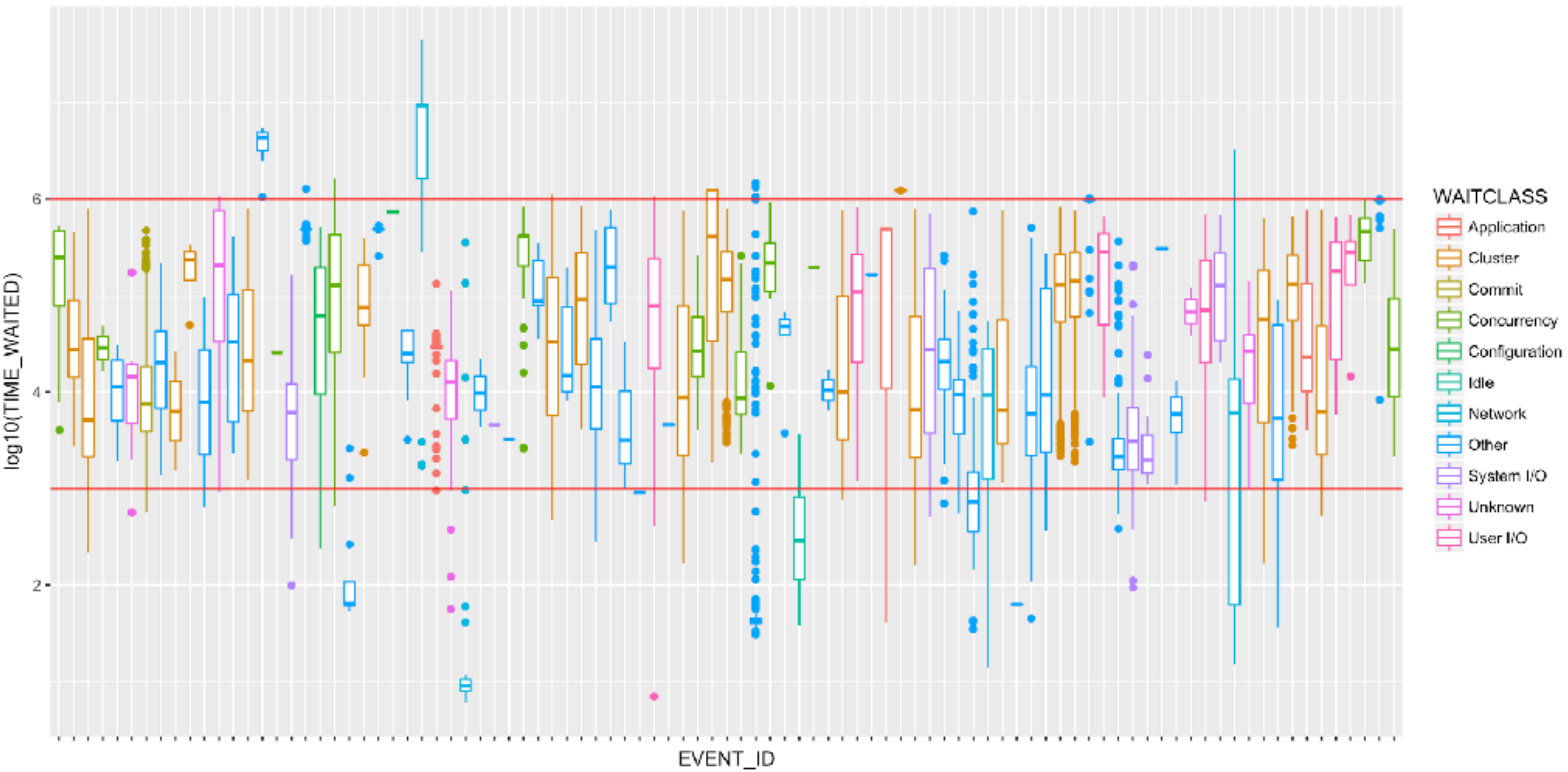
latency box plots by wait class



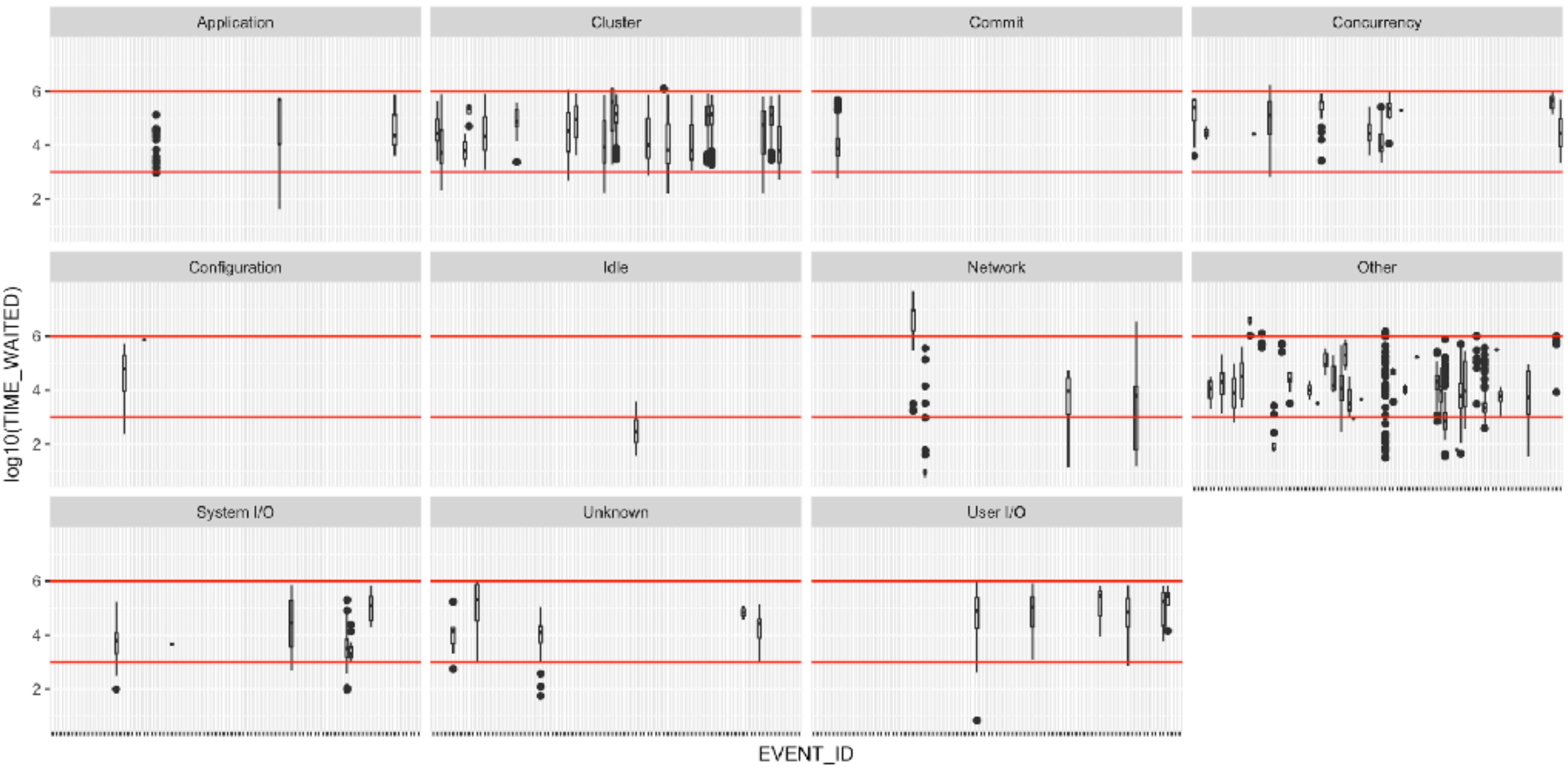
by event id



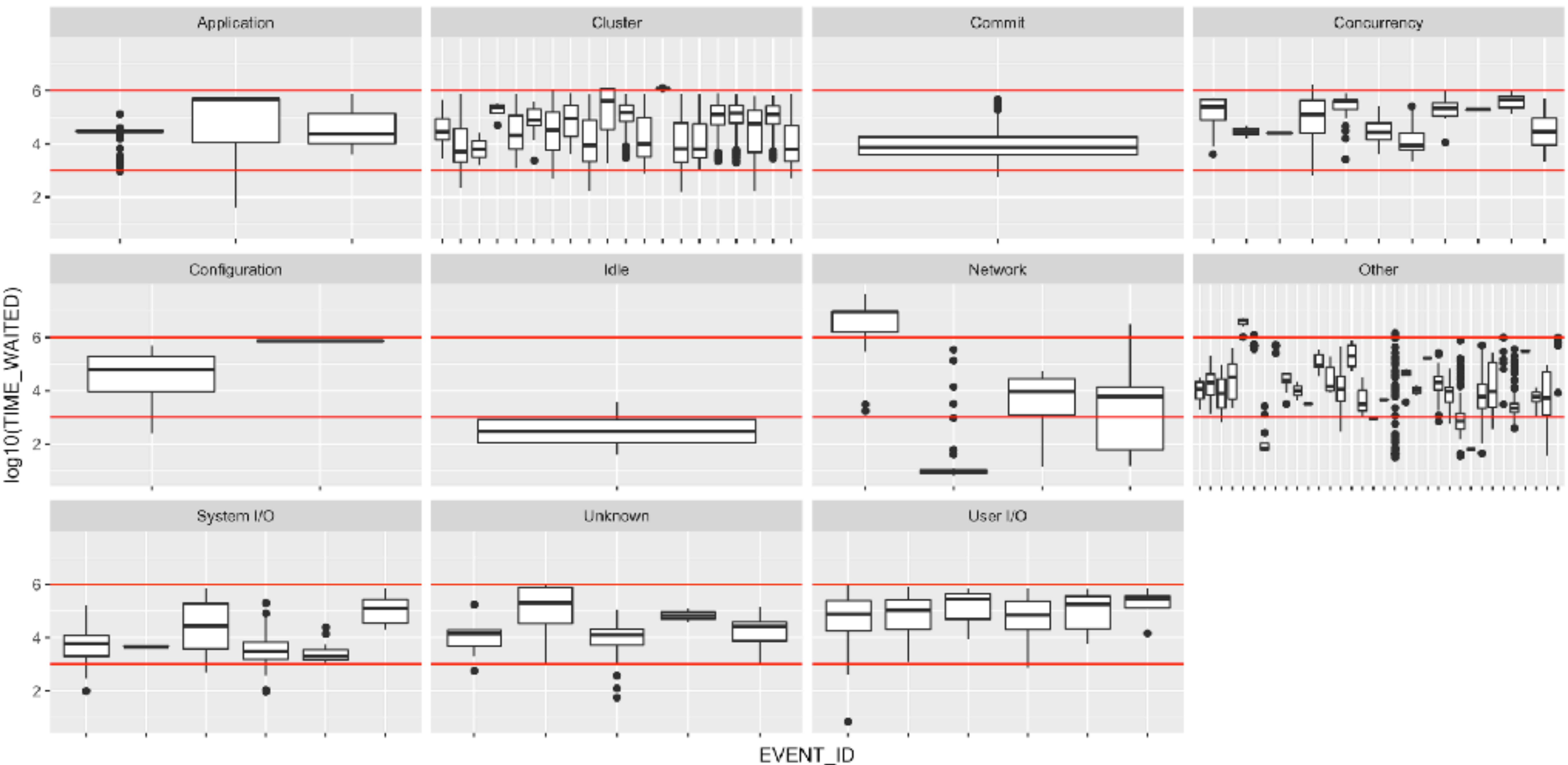
not helpful


log10(TIME_WAITED)
6
4
2
EVENT_ID
WAITCLASS
Application
Cluster
Commit
Concurrency
Configuration
Idle
Network
Other
System I/O
Unknown
User I/O

faceted by wait class



free x-axis so geoms fill space



See what could be:
estimated event counts

What and why?

- Patent US9633061B2 granted in 2017
  (Uri Shaft, Graham Wood & John Beresniewicz)

- Methods for determining event counts based on time-sampled data
  (i.e. ASH)

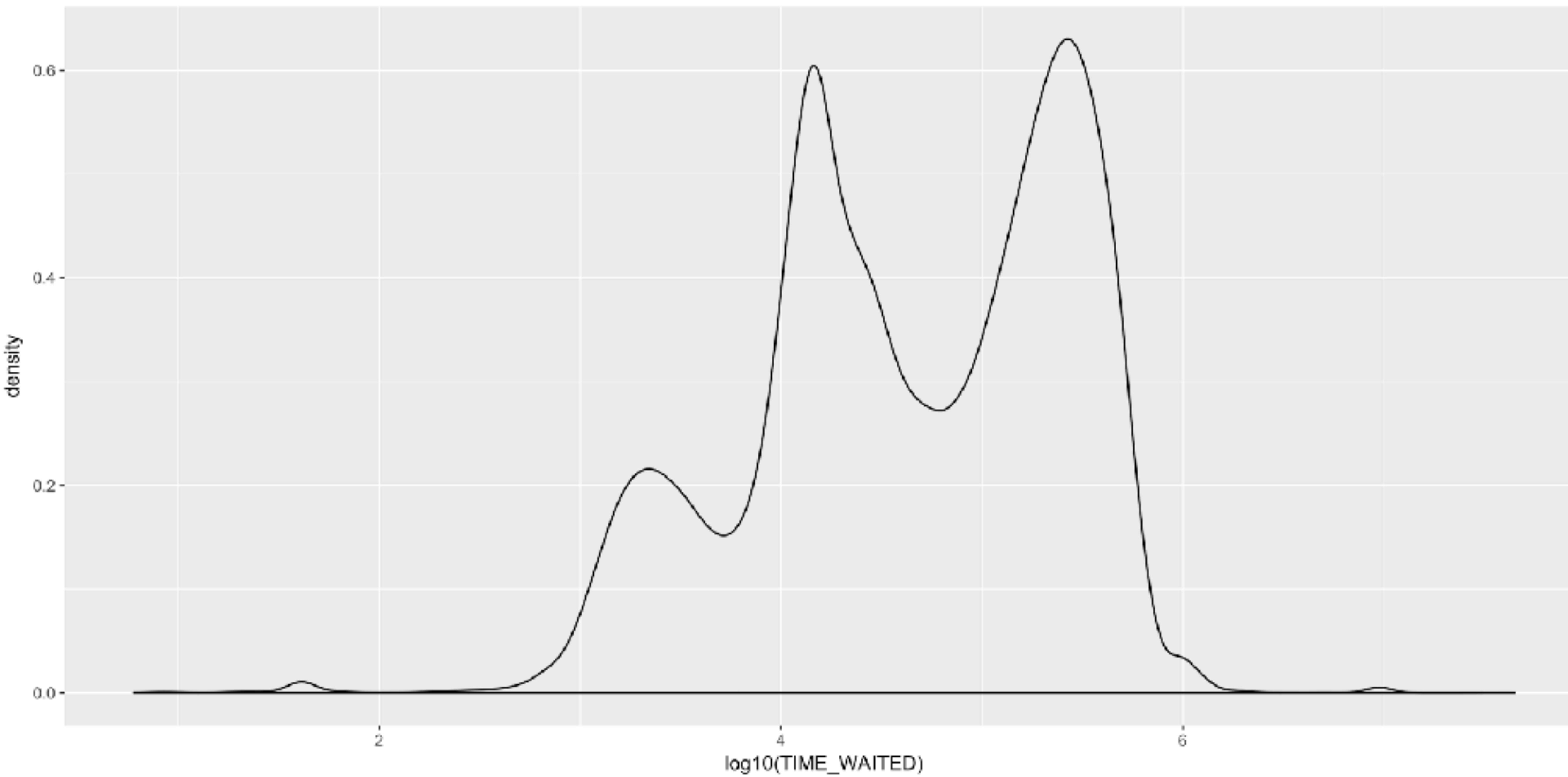
- Estimate average event latencies from samples:

  - est latency = est DBtime / est count

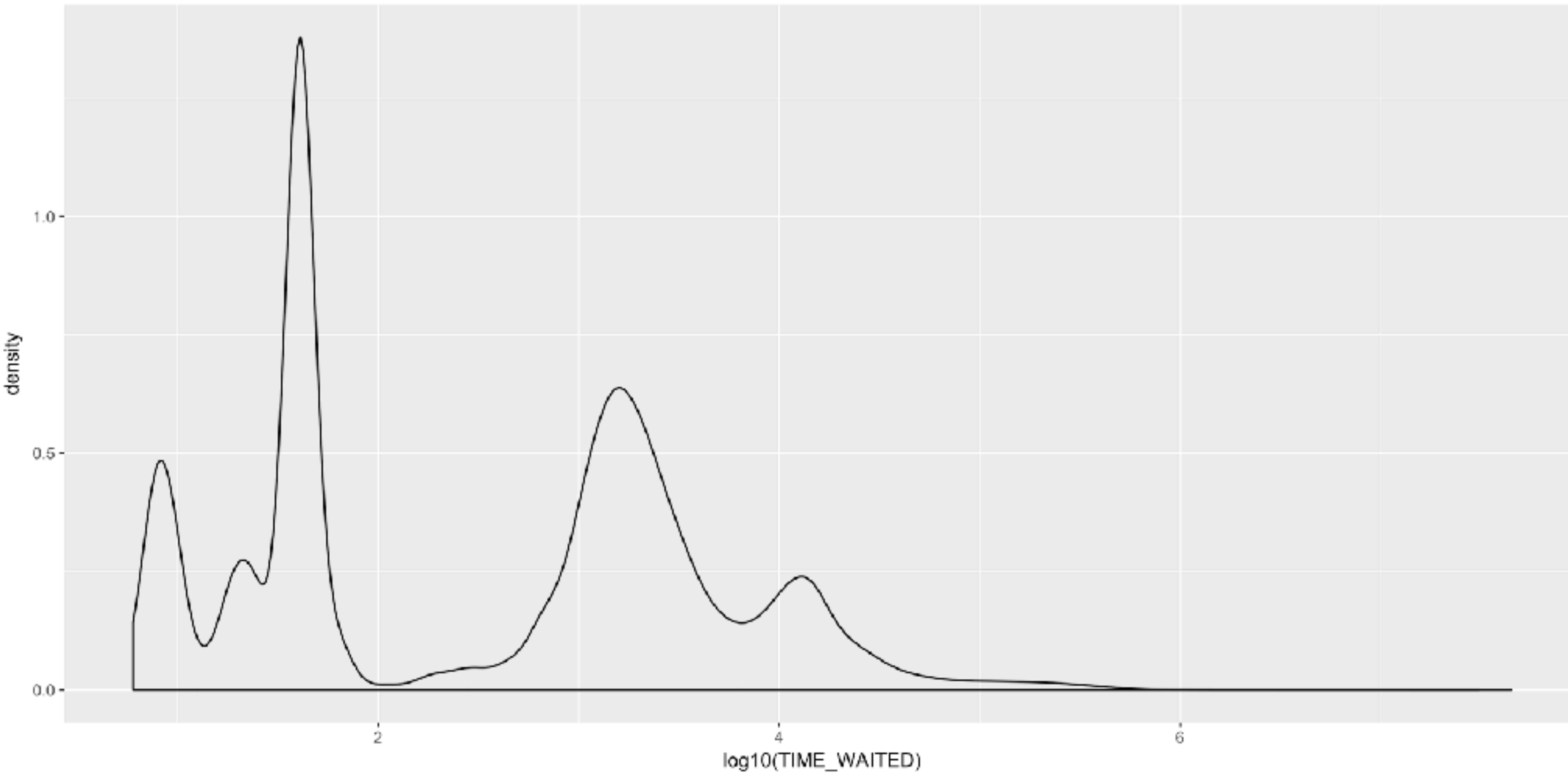
  - EST_COUNT = MAX(1000000 / TIME_WAITED, 1)
    (when 0 < time_waited, and sample interval = 1000 ms)

- Performance Metric: Estimated Event Latency over past minute

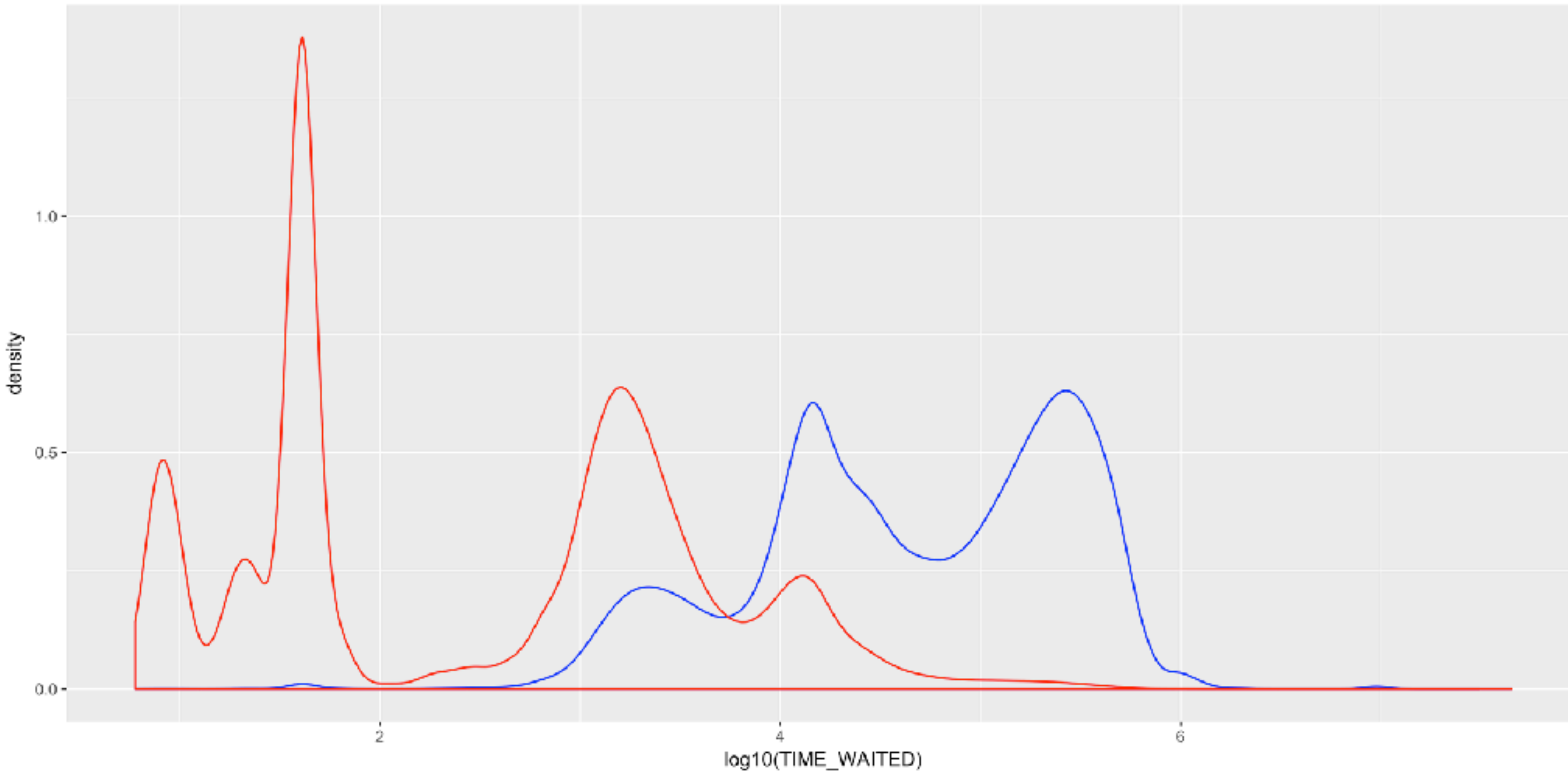
density plot: observed event count by latency



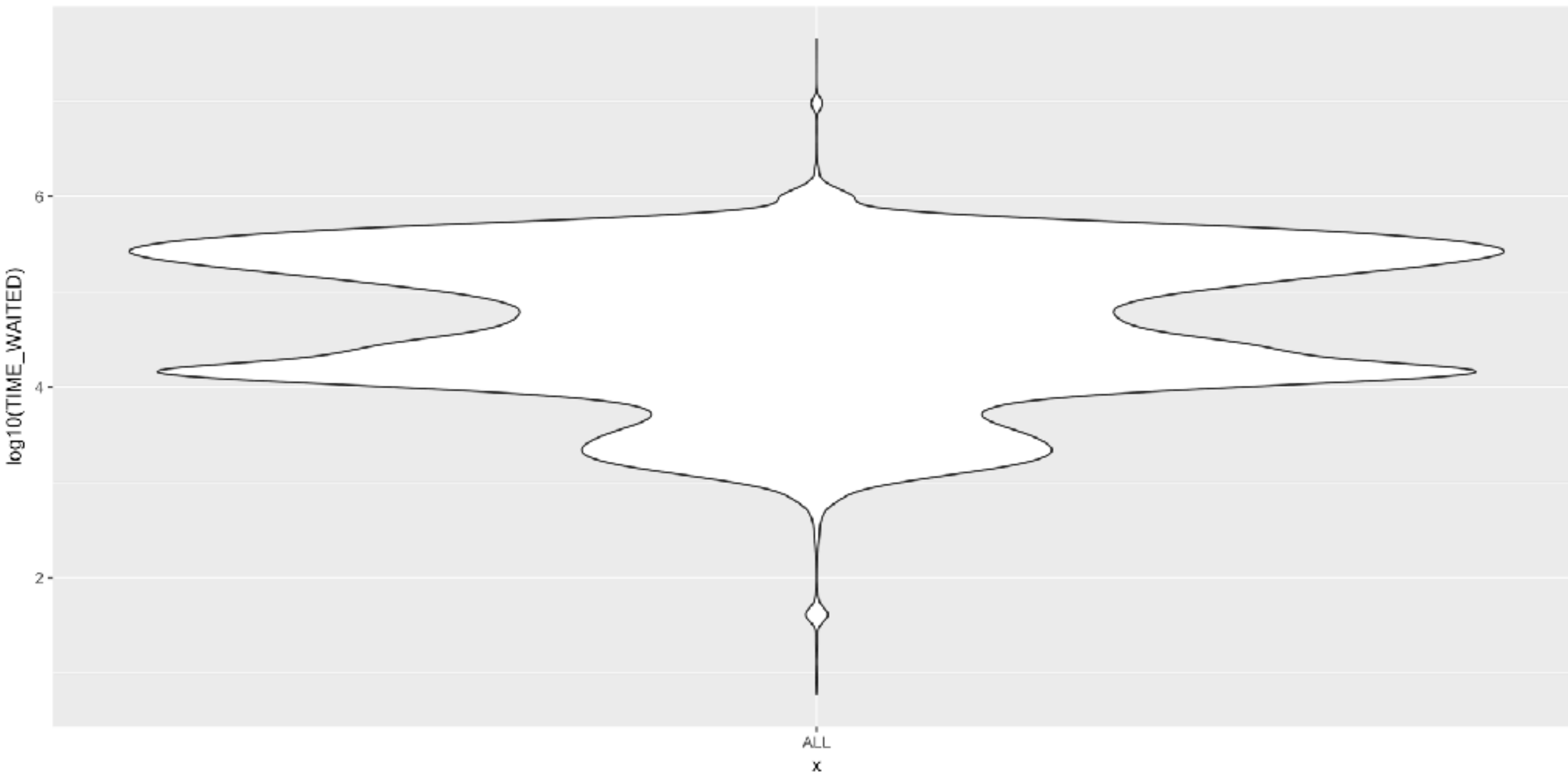
latency density weighted by estimated count



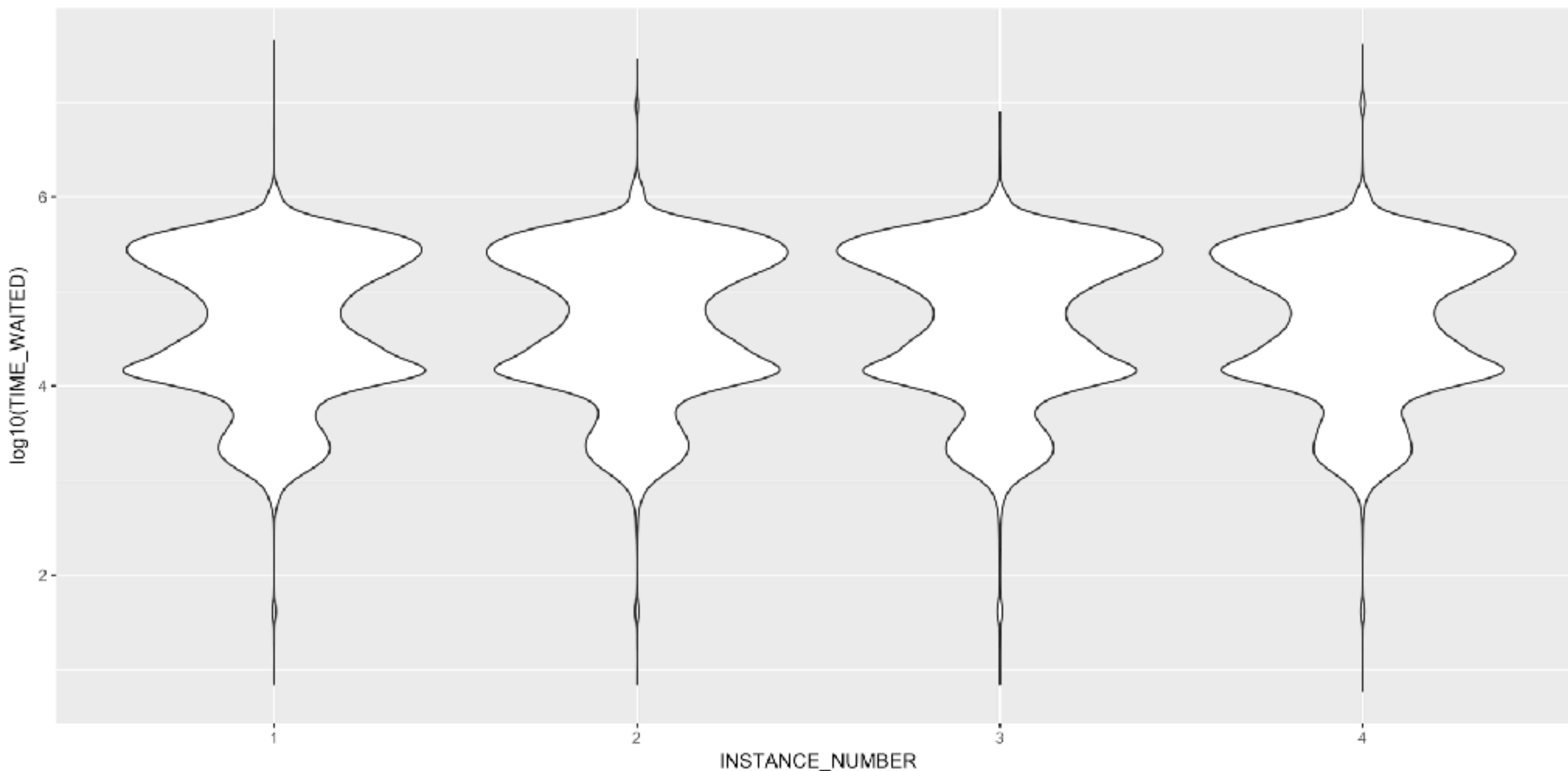
together
(blue = observed, red = estimated)



violin density plot, observed

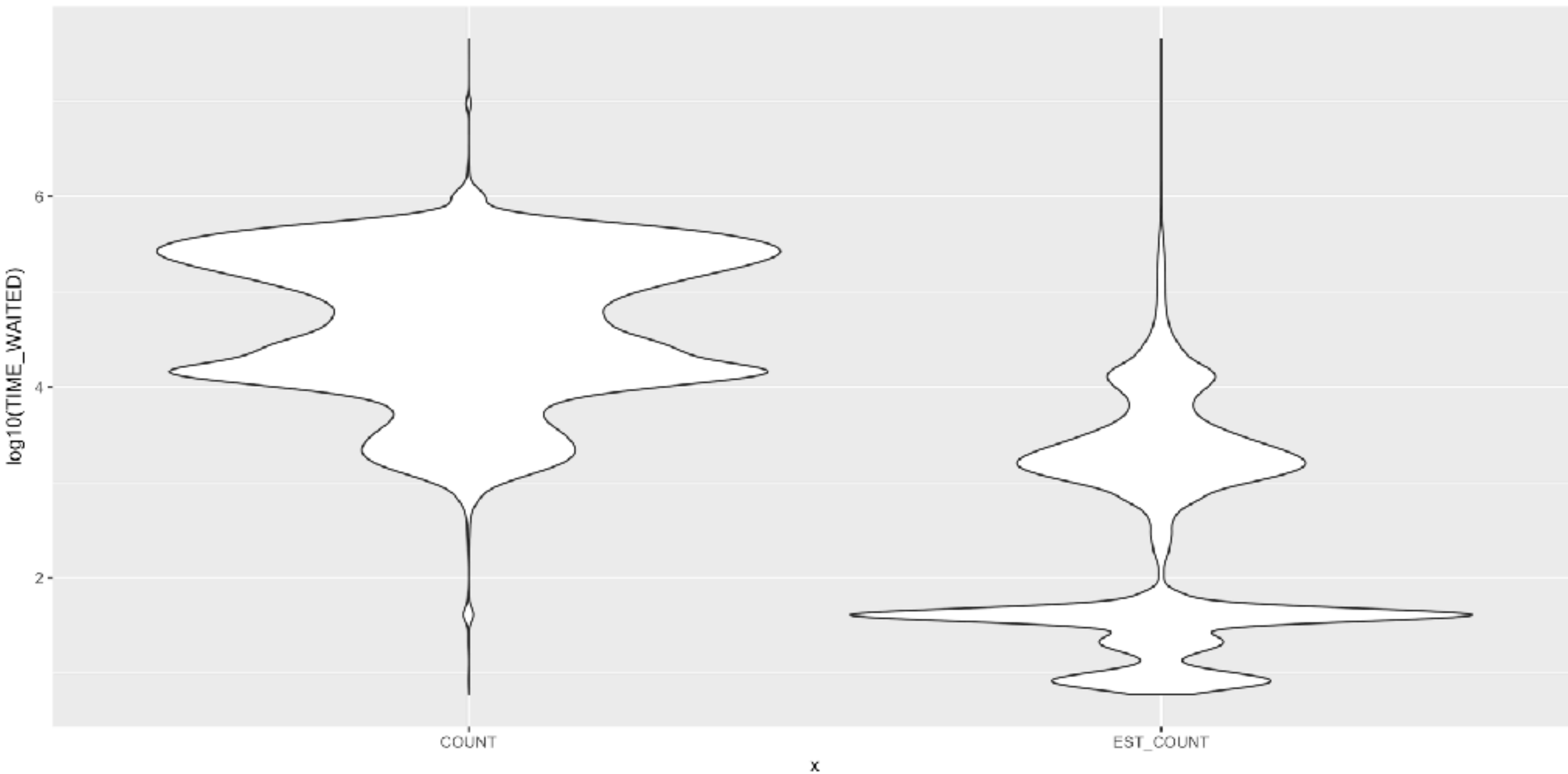


faceted by instance

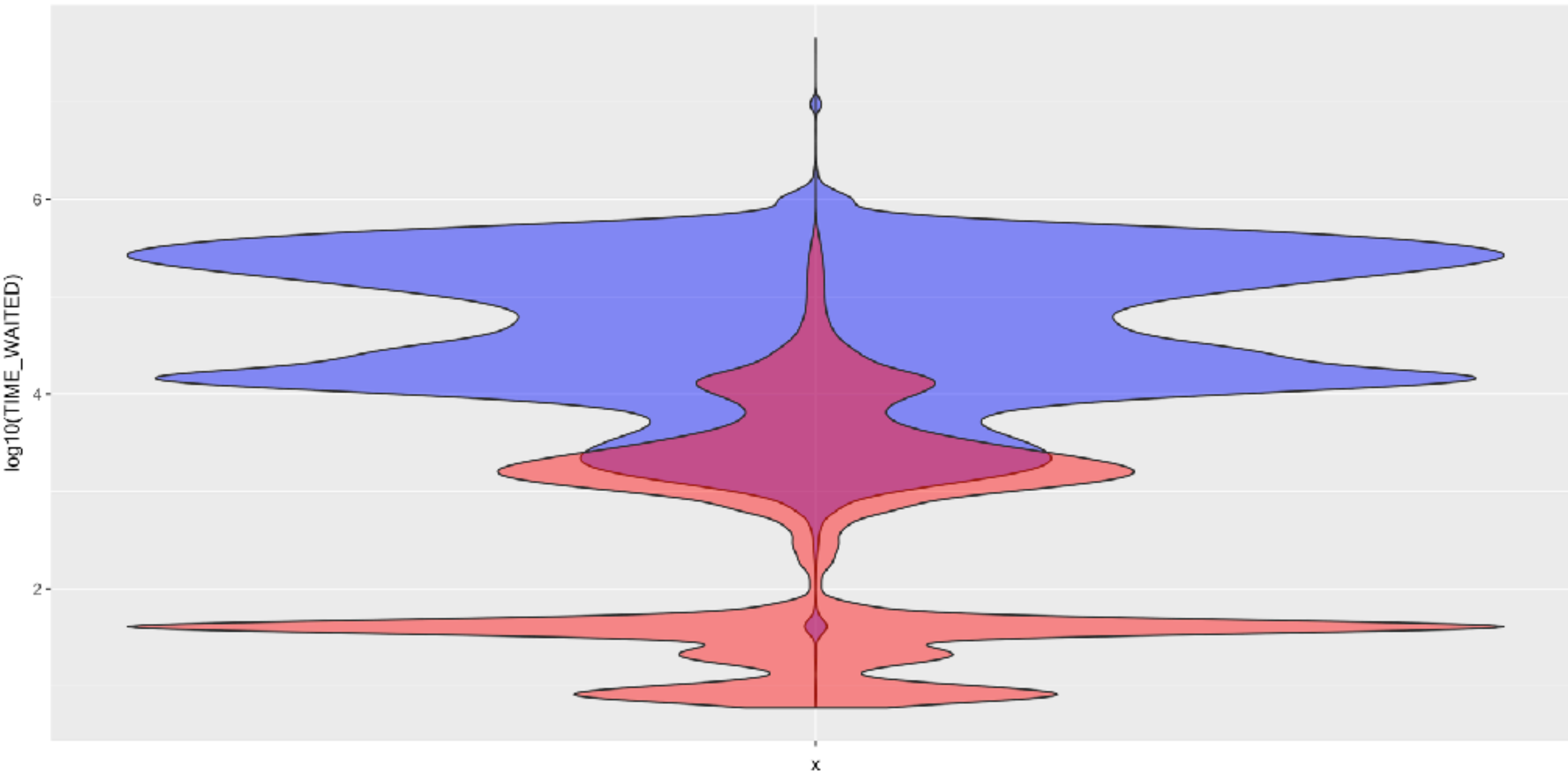


weighted by estimated count

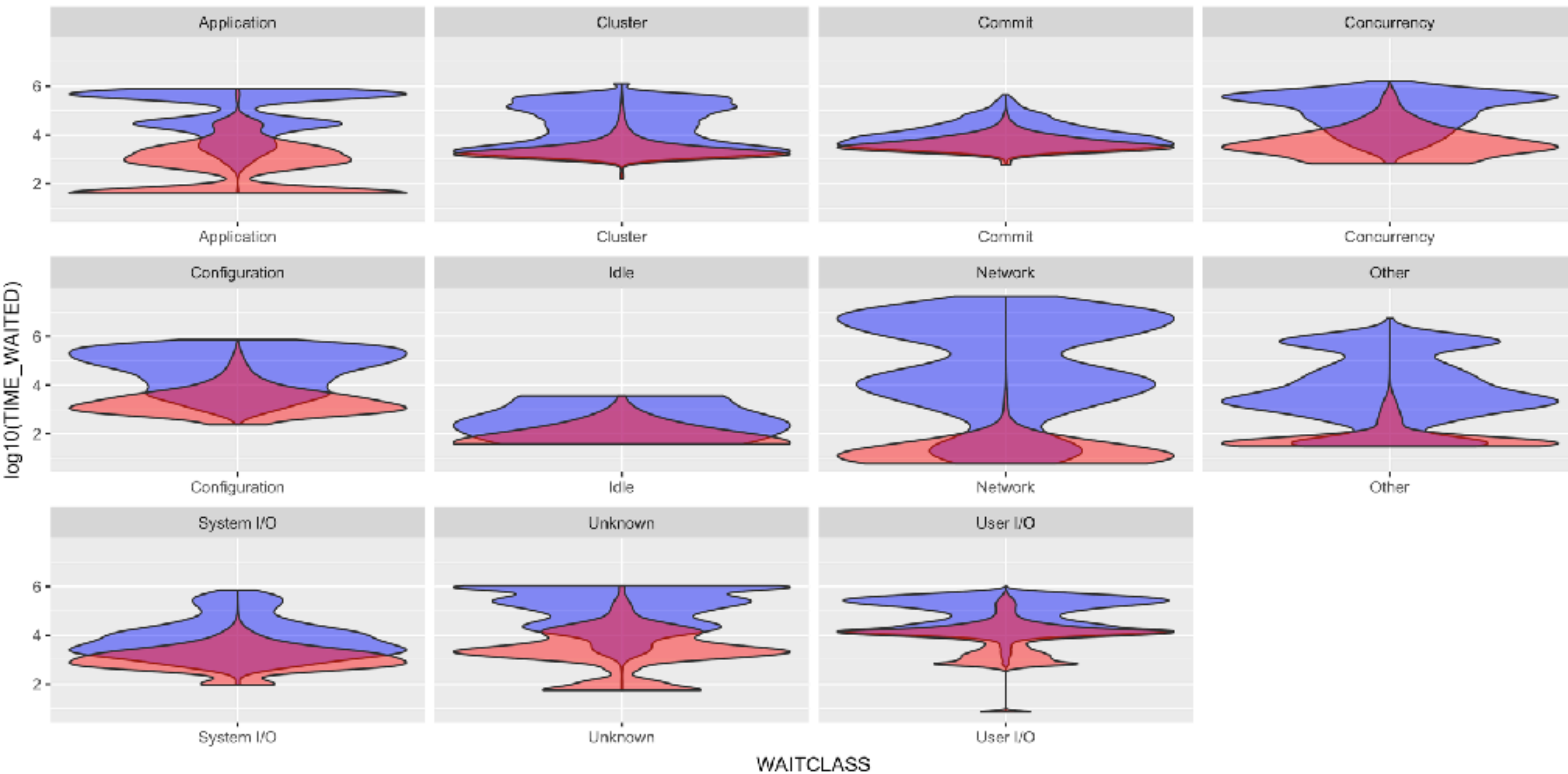
together, the balloon squeeze



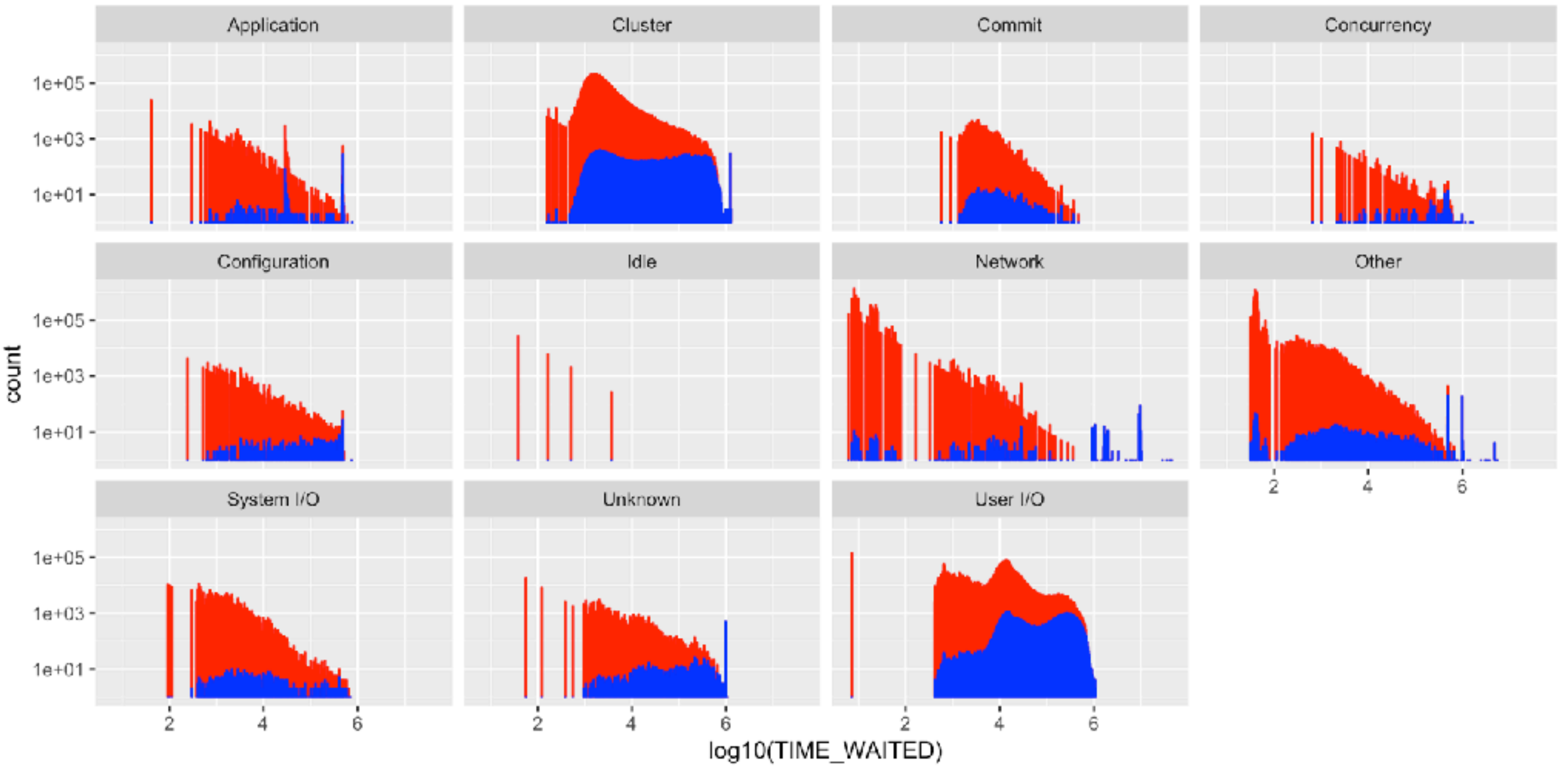
looks cool, but is it useful?



facet by wait class, misleading

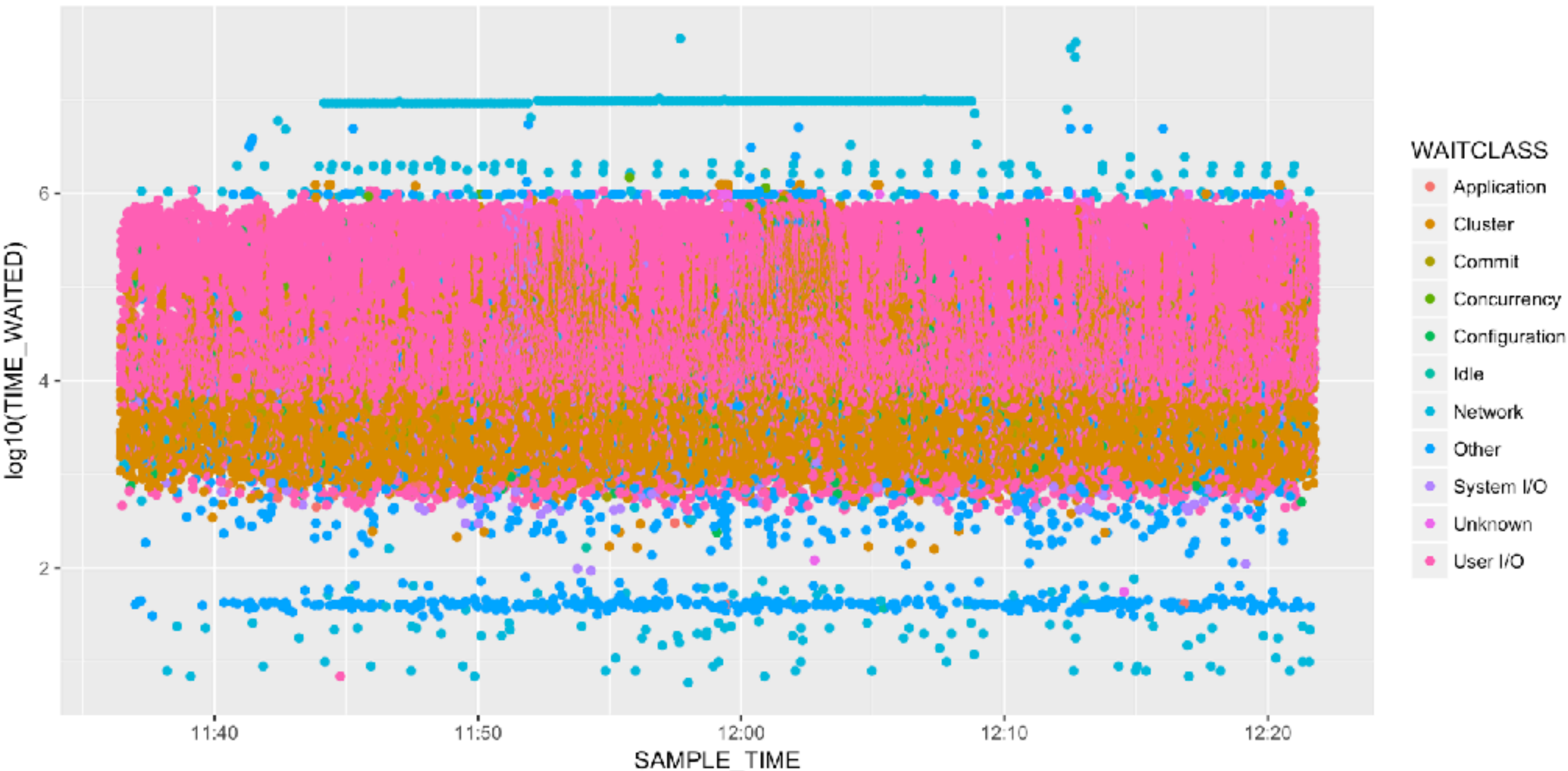


much better view

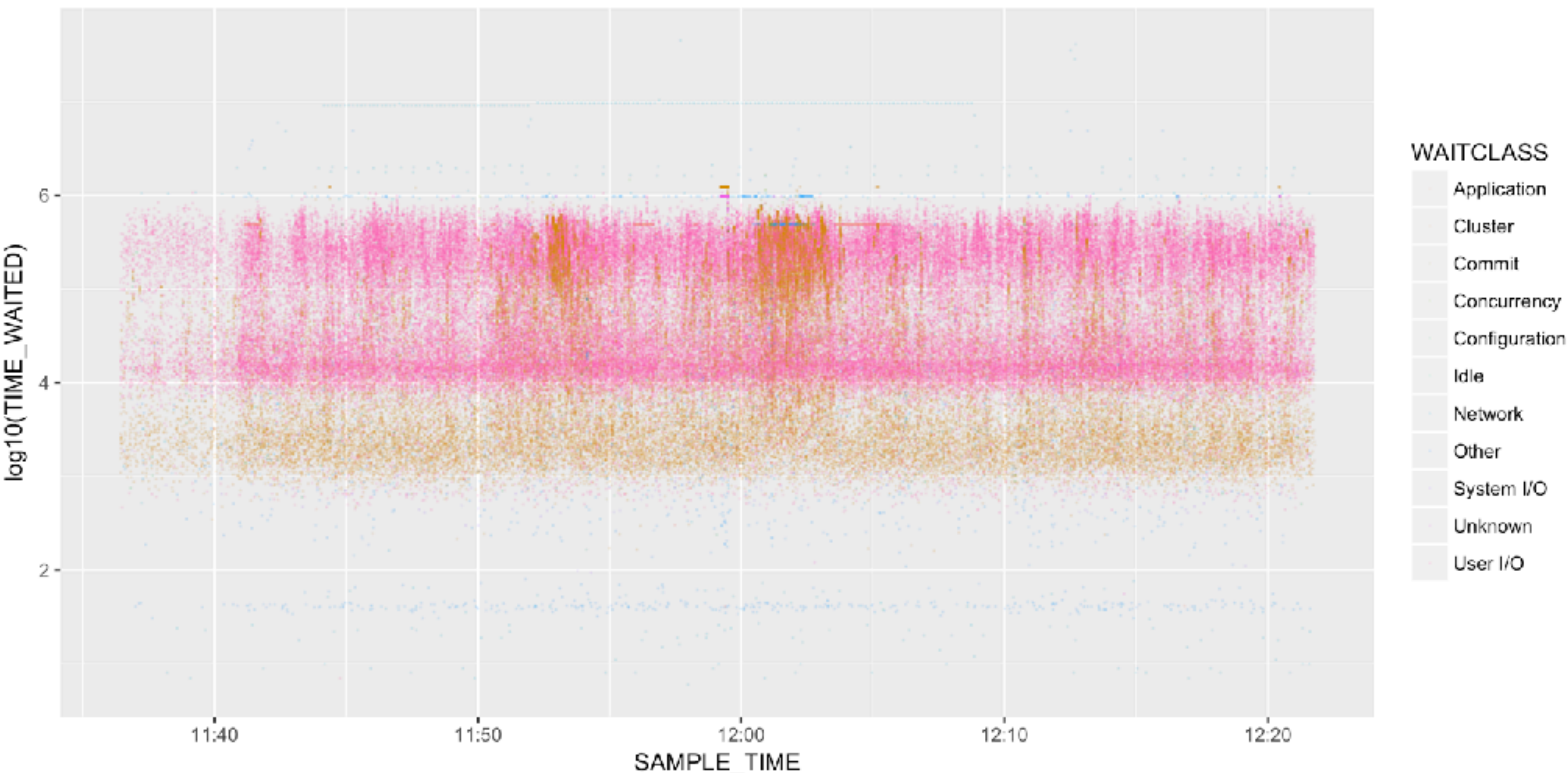


Back to raw data

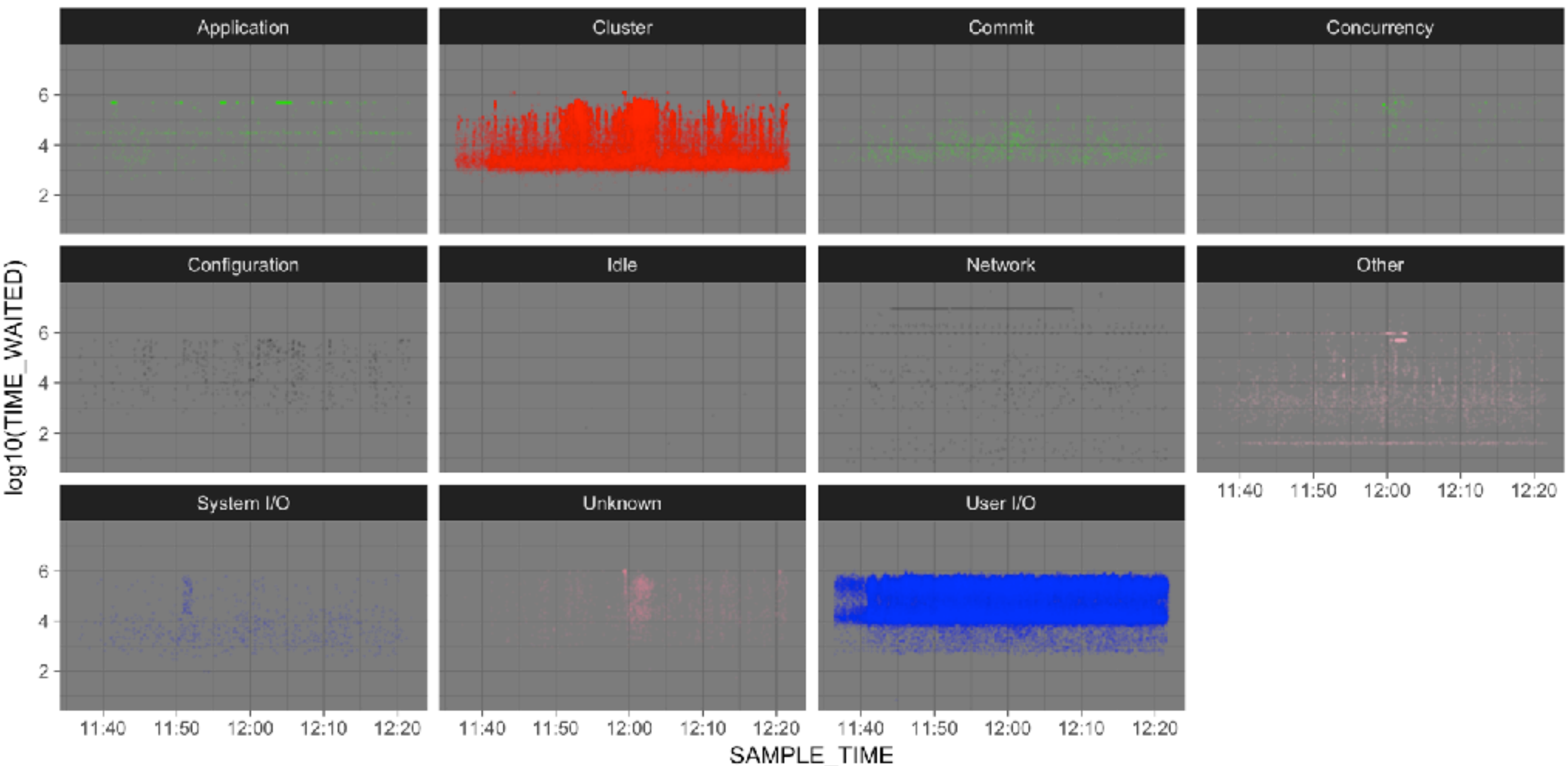
all samples over time - overplots



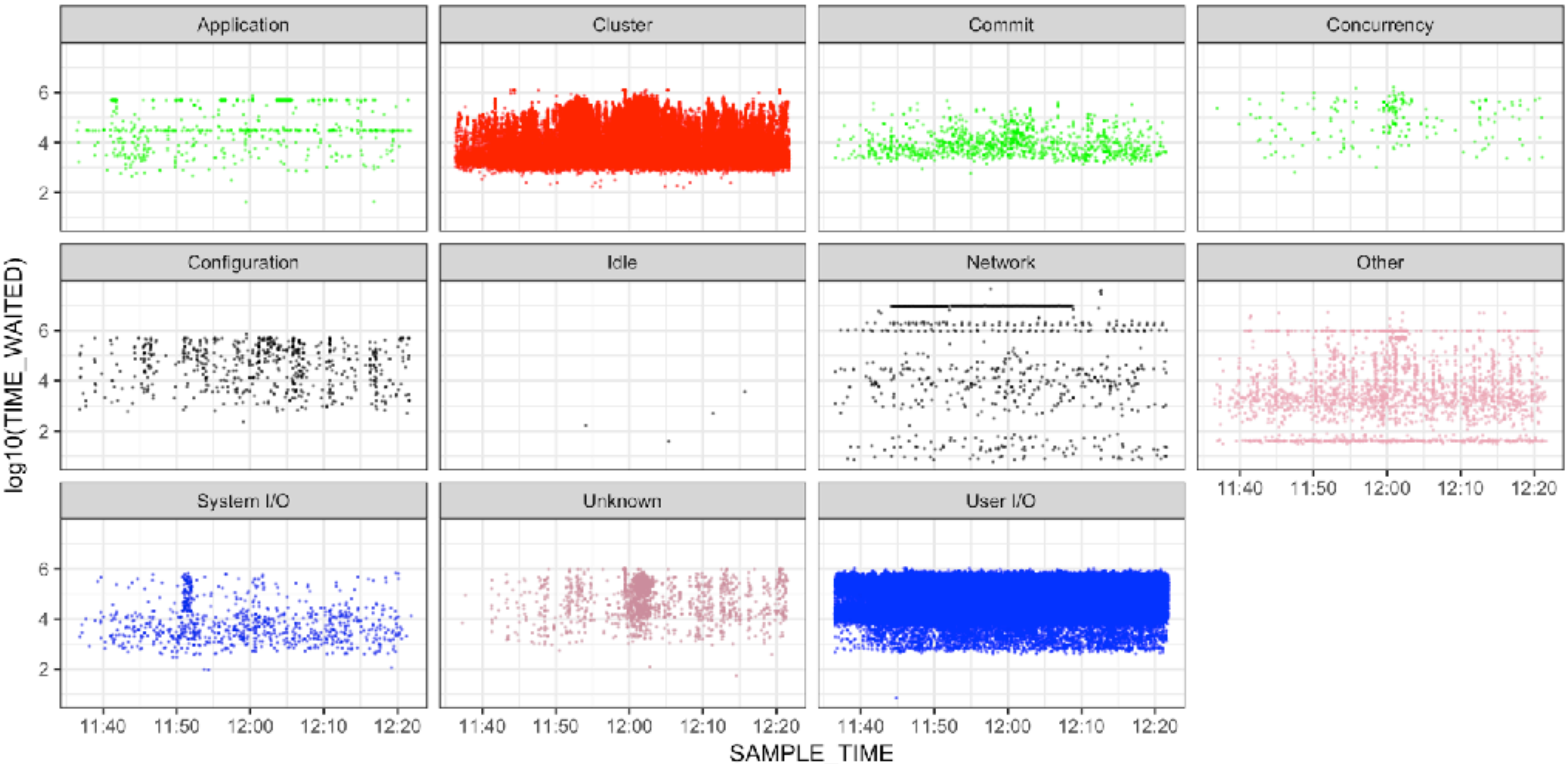
reduce point size and alpha



facet wait class, theme_dark

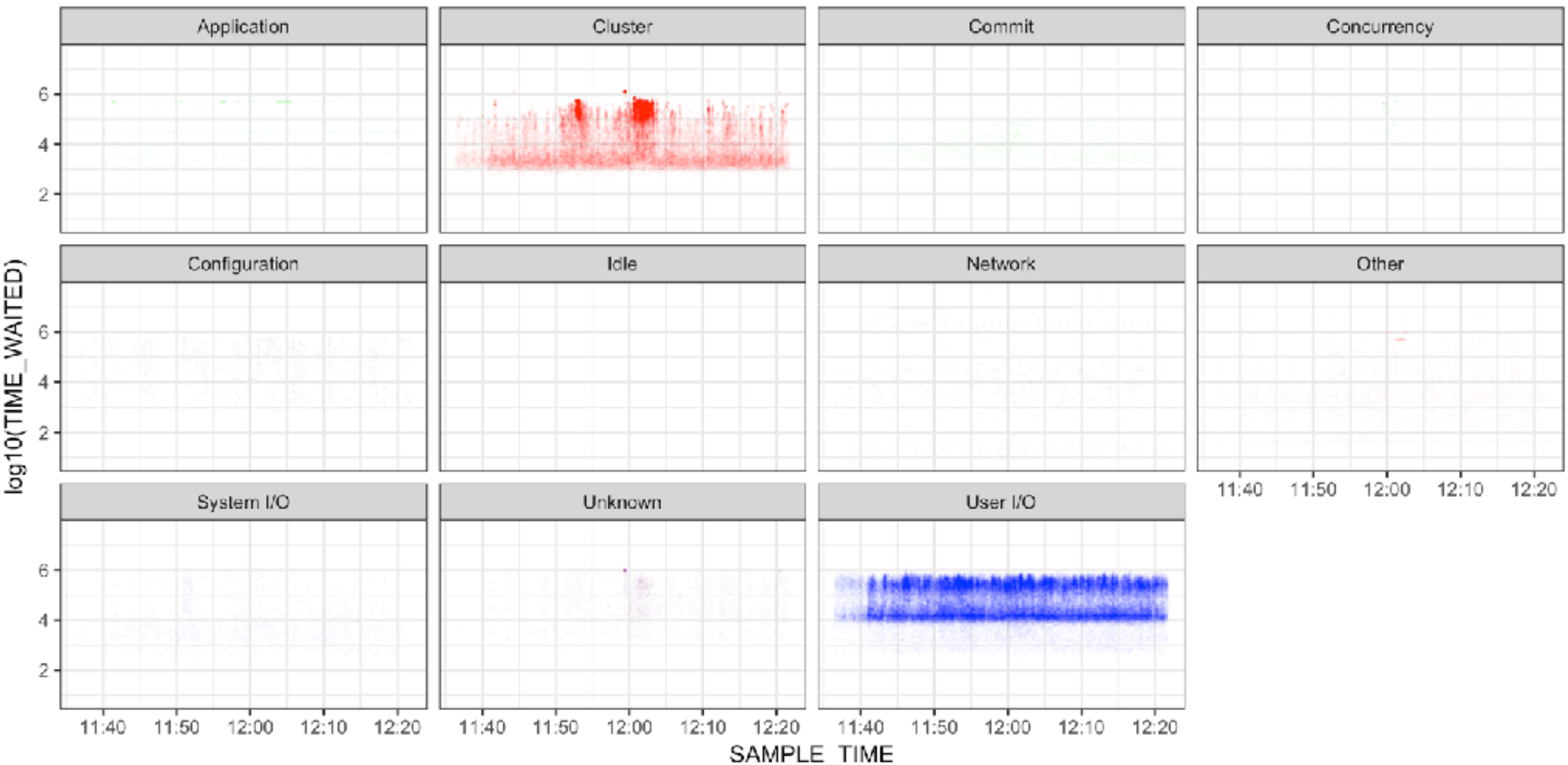


theme_bw

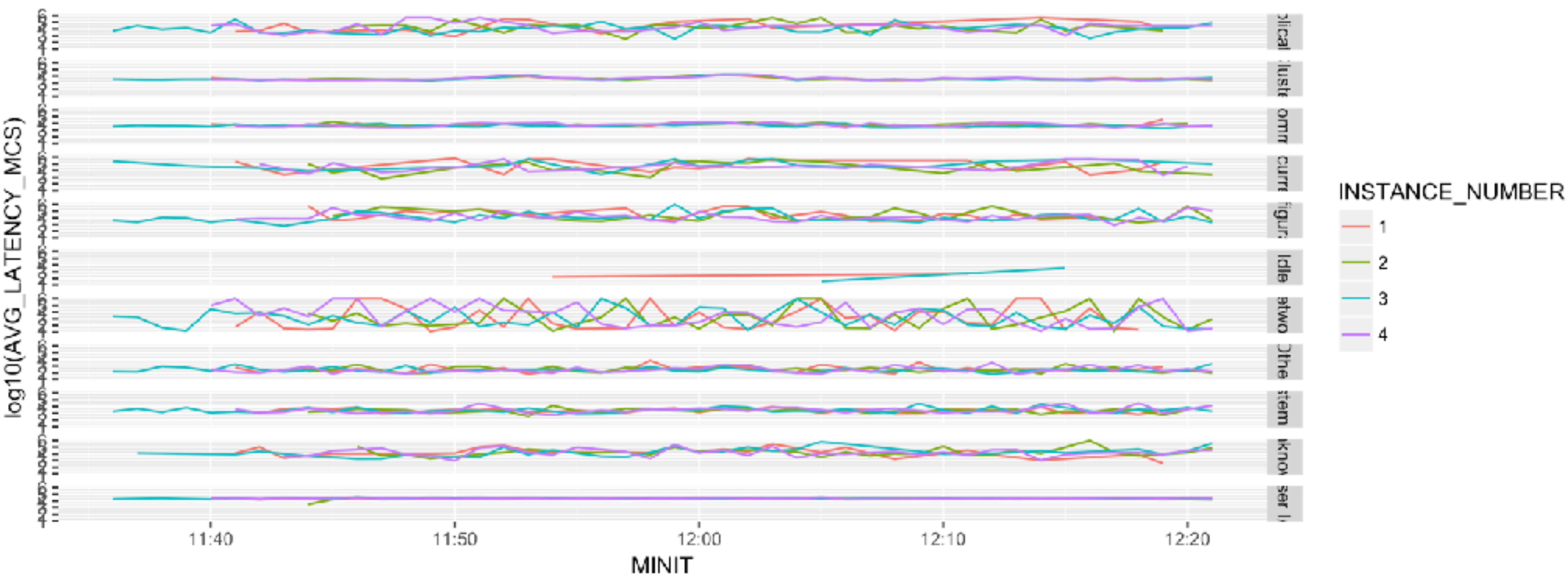


Application
Cluster
Commit
Concurrency
Configuration
Idle
Network
Other
System I/O
Unknown
User I/O
log10(TIME_WAITED)
6
4
2
11:40
11:50
12:00
12:10
12:20
SAMPLE_TIME

reduced alpha and point size



estimated 1 min avg latencies
color by instance, facet by wait class



ASH Visualization

- ASH data is highly dimensional and temporally fine-grained
- Ask questions and visualize to investigate
- Understand the visualization and whether it answers question
- Iterate visualizations by changing geometry attributes
- Experiment liberally, but be true to the data