Practical Data Visualization

Visualization can be a valuable tool for analysing and understanding data (and the "reality" that we are trying to measure). It can be much richer than numeric metrics, providing valuable insights. It can also distort, camouflage, or hide information. This talk will provide some real-life examples of how to use and mis-use visualization.

Frank Rowand, Sony Network Entertainment

November 7, 2012

121106_0502

Why do I analyze data?

Debugging performance

- If I know the factor that is limiting performance I know where to focus efforts to improve it

Predicting behavior of future workloads

- If I can describe the factors that affect performance then I can predict whether a workload will perform adequately
  - examples:
    - batch throughput, transaction processing responsiveness, gaming, audio recording or playback, desktop interactivity

What am I trying to accomplish?

I am trying to

- remove noise from the data
- find a signal in the data
- solve a mystery
- explore a problem space
- gain insights into how and why a system is behaving
- not lie to myself

What this talk is not

Edward Tufte's books:

- The Visual Display of Quantitative Information
- Envisioning Information
- Visual Explanations

http://www.edwardtufte.com/tufte/books_vdqi

These books are highly recommended. This talk is much more humble....

Data Example

Task Switch Time

Which is Better?

```
           min   avg
           ---   ---
ts_11:      34    69
ts_12:      35    68
```

(hint: smaller is better)

Which is Better?

```
          min  avg  max
          ---  ---  ---
ts_11:     34   69  147
ts_12:     35   68  159
```

(conflicting metrics)

Which is Better?

	min	avg	max	std dev
ts_11:	34	69	147	21.9
ts_12:	35	68	159	28.0

(conflicting metrics)

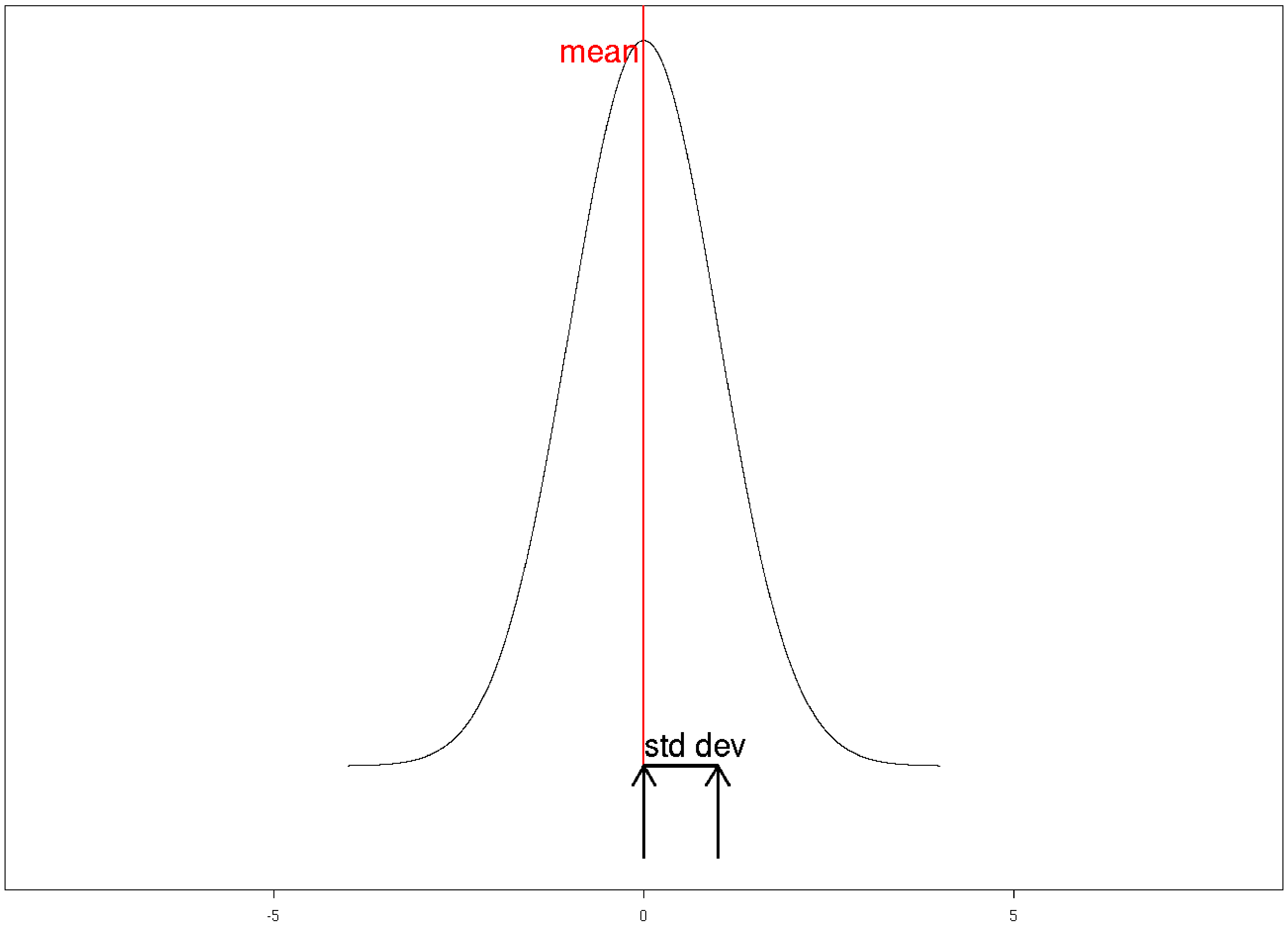
standard deviation - What is it?

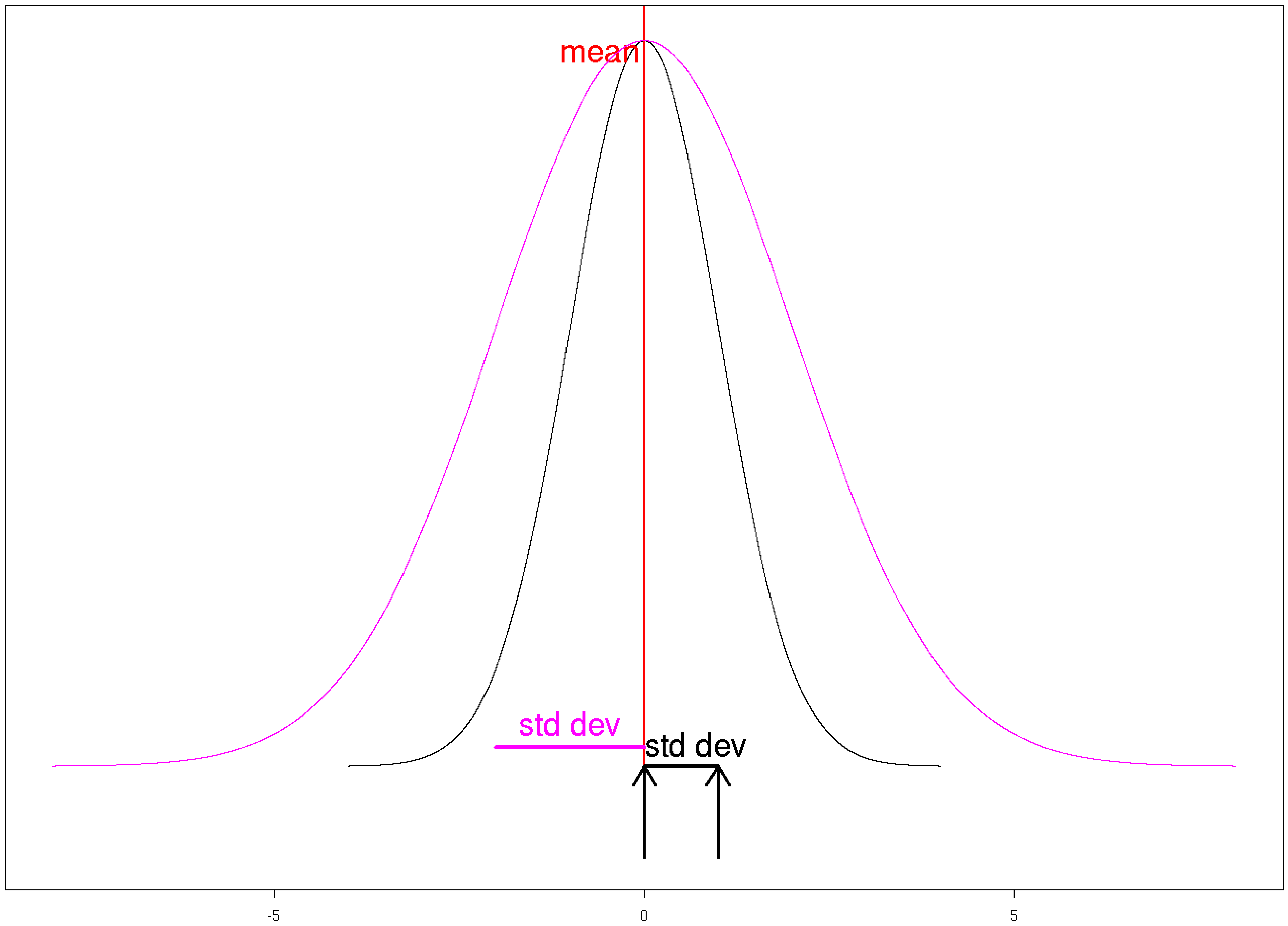
Intuitive description

A measure of the width of a frequency distribution.

A smaller standard deviation means the distribution of the data points is narrower.

A smaller standard deviation means the distribution of the data points is closer to the mean.


mean
std dev
-5
0
5


mean
std dev
std dev
-5
0
5

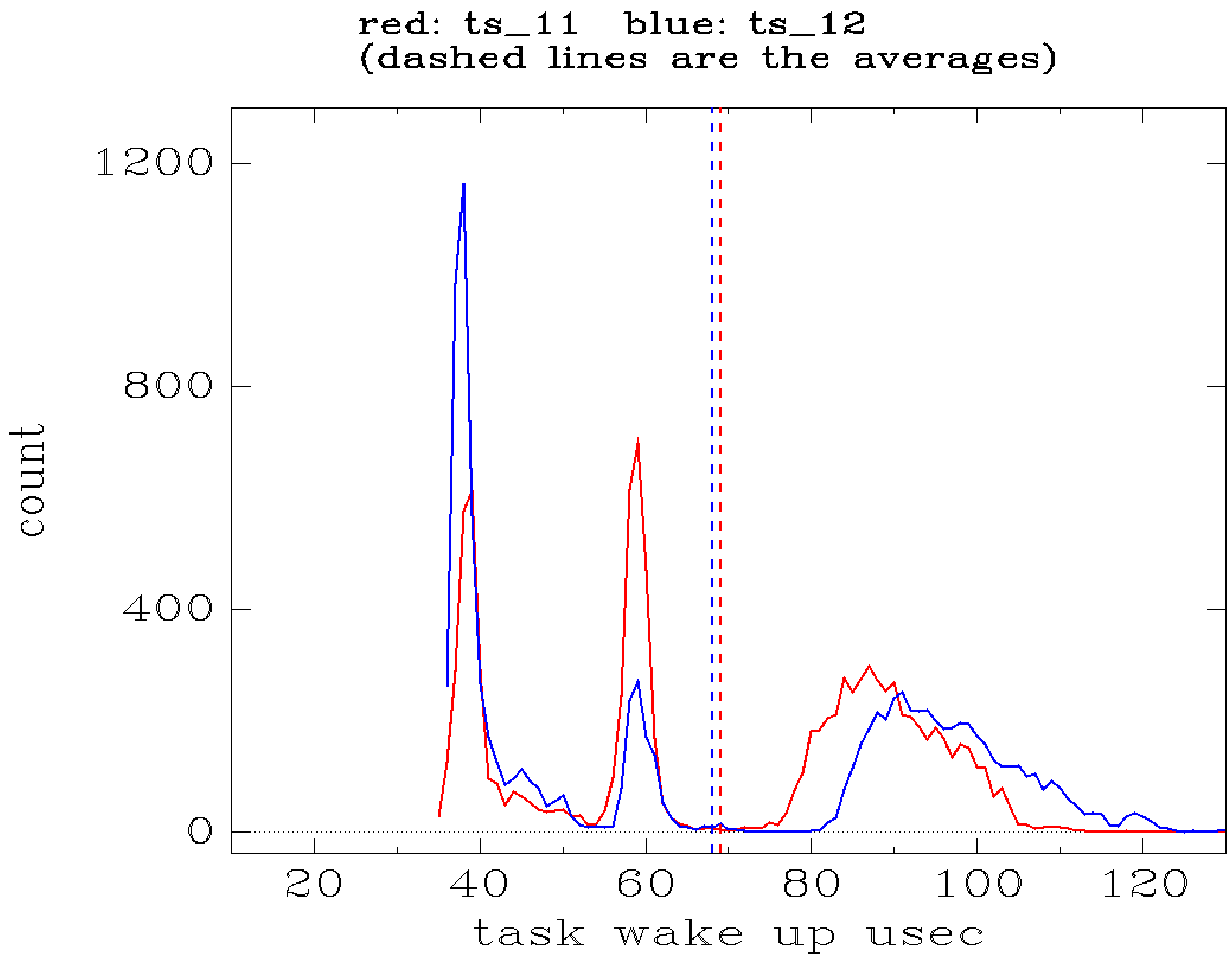
standard deviation – What is it?

If the data was a perfect “normal distribution”, it would be described by the curves on the previous slide.

The starting point for many statistics analysis is “assume a normal distribution”.

Which is Better?

	min	avg	max	std dev
ts_11:	34	69	147	21.9
ts_12:	35	68	159	28.0


red: ts_11 blue: ts_12
(dashed lines are the averages)
count
1200
800
400
0
20
40
60
80
100
120
task wake up usec

Answer

It depends.

Answer

It depends.

Real Answer:

- “Average is a meaningless metric”
- “Standard Deviation is a meaningless metric”

Answer

Trick question. The two sets of data, ts_11 and ts_12, are two different runs of the same test.

Real Answer:

- The graph explains difference between the two test runs better than the raw metrics.

Which is Better?

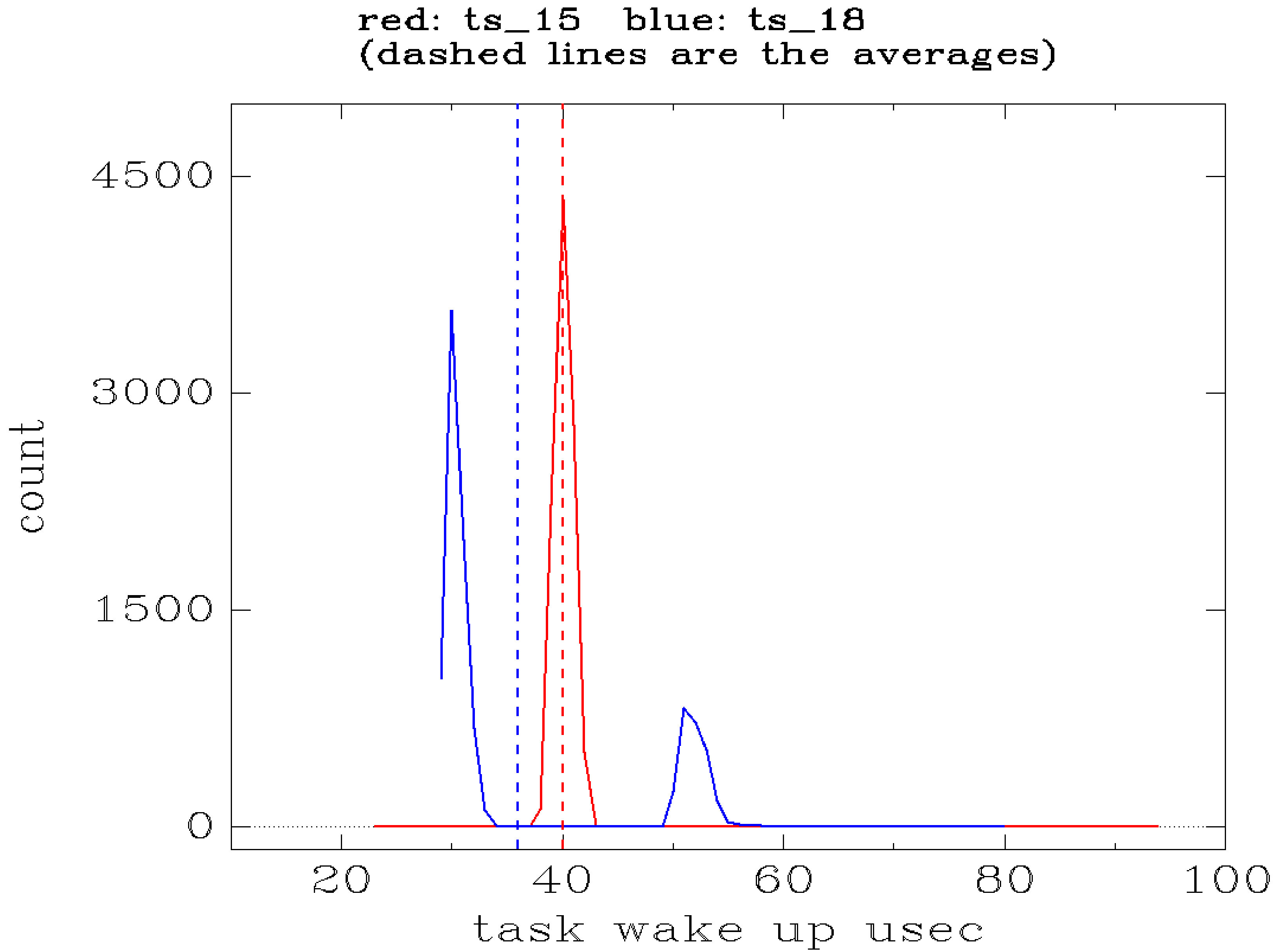
```
          min  avg  max
          ---  ---  ---

ts_15:     22   40   94

ts_18:     28   36   80
```

Which is Better?

	min	avg	max	std dev
ts_15:	22	40	94	1.2
ts_18:	28	36	80	9.6



red: ts_15 blue: ts_18
(dashed lines are the averages)
4500
3000
1500
0
count
20
40
60
80
100
task wake up usec

Answer

It depends.

What is more important for your use case?

Throughput -- average

Consistency / determinancy -- std dev, maximum

Worst case -- maximum

Answer

Real Answer:

- The graph explains difference between the two test runs better than the raw metrics.
- “Average is a meaningless metric”
- “Standard Deviation is a meaningless metric”

Average is a meaningless metric

If the different data data sets have different distributions

For hard real-time metrics, average is always meaningless (maximum is critical)

Standard Deviation is a meaningless metric

If the data is not a normal distribution

Meaningless Metrics

But Average and Standard Deviation can still be used as flags to get your attention or provide insights.

ASCII graphs can also be useful

Not sexy, but do not underestimate their power

Migration Algorithm 1

```
# producer cpu map:
#
#      0 0101010101010101010101010101010101010101010101010101010101010101010101
#     70 0101010101010101010101010101010101010101010101010101010101010101010101
#    140 0101010101010101010101010101010101010101010101010101010101010101010101
#    210 0101010101010101010101010101010101010101010101010101010101010101010101
#    280 0101010101010101010101010101010101010101010101010101010101010101010101
#    350 0101010101010101010101010101010101010101010101010101010101010101010101
#    420 0101010101010101010101010101010101010101010101010101010101010101010101
#    490 0101010101010101010101010101010101010101010101010101010101010101010101
#    560 0101010101010101010101010101010101010101010101010101010101010101010101
#    630 0101010101010101010101010101010101010101010101010101010101010101010101
#    700 0101010101010101010101010101010101010101010101010101010101010101010101
#
# consumer cpu map:
#
#      0 1010101010101010101010101010101010101010101010101010101010101010101010
#     70 1010101010101010101010101010101010101010101010101010101010101010101010
#    140 1010101010101010101010101010101010101010101010101010101010101010101010
#    210 1010101010101010101010101010101010101010101010101010101010101010101010
#    280 1010101010101010101010101010101010101010101010101010101010101010101010
#    350 1010101010101010101010101010101010101010101010101010101010101010101010
#    420 1010101010101010101010101010101010101010101010101010101010101010101010
#    490 1010101010101010101010101010101010101010101010101010101010101010101010
#    560 1010101010101010101010101010101010101010101010101010101010101010101010
#    630 1010101010101010101010101010101010101010101010101010101010101010101010
#    700 1010101010101010101010101010101010101010101010101010101010101010101010
```

Migration Algorithm 1

```
# producer cpu map:
#
#      0 0101010101010101010101010101010101010101
#     70 0101010101010101010101010101010101010101
#
# consumer cpu map:
#
#      0 1010101010101010101010101010101010101010
#     70 1010101010101010101010101010101010101010
```

Excessive migration is usually not good.

Migration Algorithm 2

producer and consumer always on same cpu,
instead of always on the other cpu

```
#                 -- producer --------------
# consumer          cpu 0           cpu 1
# --------        -----------   -----------
#  cpu 0                 5168             0
#  cpu 1                    0          4832
```

Migration Algorithm 2

producer and consumer always on same cpu,
instead of always on the other cpu

```
#                 -- producer -------------
# consumer          cpu 0           cpu 1
# --------        -----------   -----------
#  cpu 0                 5168             0
#  cpu 1                    0          4832
```

How often does migration occur?
  One migration per test run?
  One migration per message?

Migration Algorithm 2

```
# producer cpu map:
#
#      0 0010101010000000000100111111111111111111111111111111111111111111111111
#     70 1111111111111111111111111111111111111111111111111100000000000000000000
#    140 0000000000001111111111111111111111111111111111111111111111111111111111
#    210 1111111110001111111111111111111111111111111111111111111111111111111111
#    280 1111111111111111111111111111111111111111000000000000000000000000000000
#    350 0000000000000000000000000000000000000000000000000000000000000000000001
#    420 0011111111111111111111111111111111111111111111111111111111111111111111
#    490 1111111111111111111111111111110000000000000000000000000000000000000000
#    560 0000000000000000000000000000000000000000000000000000000000010011111111
#    630 1111111111111111111111111111111111111111111111111111111111111111111111
#    700 1111111111111111111100000000000000000000000000000000000000000000000000
#
# consumer cpu map:
#
#      0 0010101010000000000100111111111111111111111111111111111111111111111111
#     70 1111111111111111111111111111111111111111111111111100000000000000000000
#    140 0000000000001111111111111111111111111111111111111111111111111111111111
#    210 1111111110001111111111111111111111111111111111111111111111111111111111
#    280 1111111111111111111111111111111111111111000000000000000000000000000000
#    350 0000000000000000000000000000000000000000000000000000000000000000000001
#    420 0011111111111111111111111111111111111111111111111111111111111111111111
#    490 1111111111111111111111111111110000000000000000000000000000000000000000
#    560 0000000000000000000000000000000000000000000000000000000000010011111111
#    630 1111111111111111111111111111111111111111111111111111111111111111111111
#    700 1111111111111111111100000000000000000000000000000000000000000000000000
```

Migration Algorithm 2

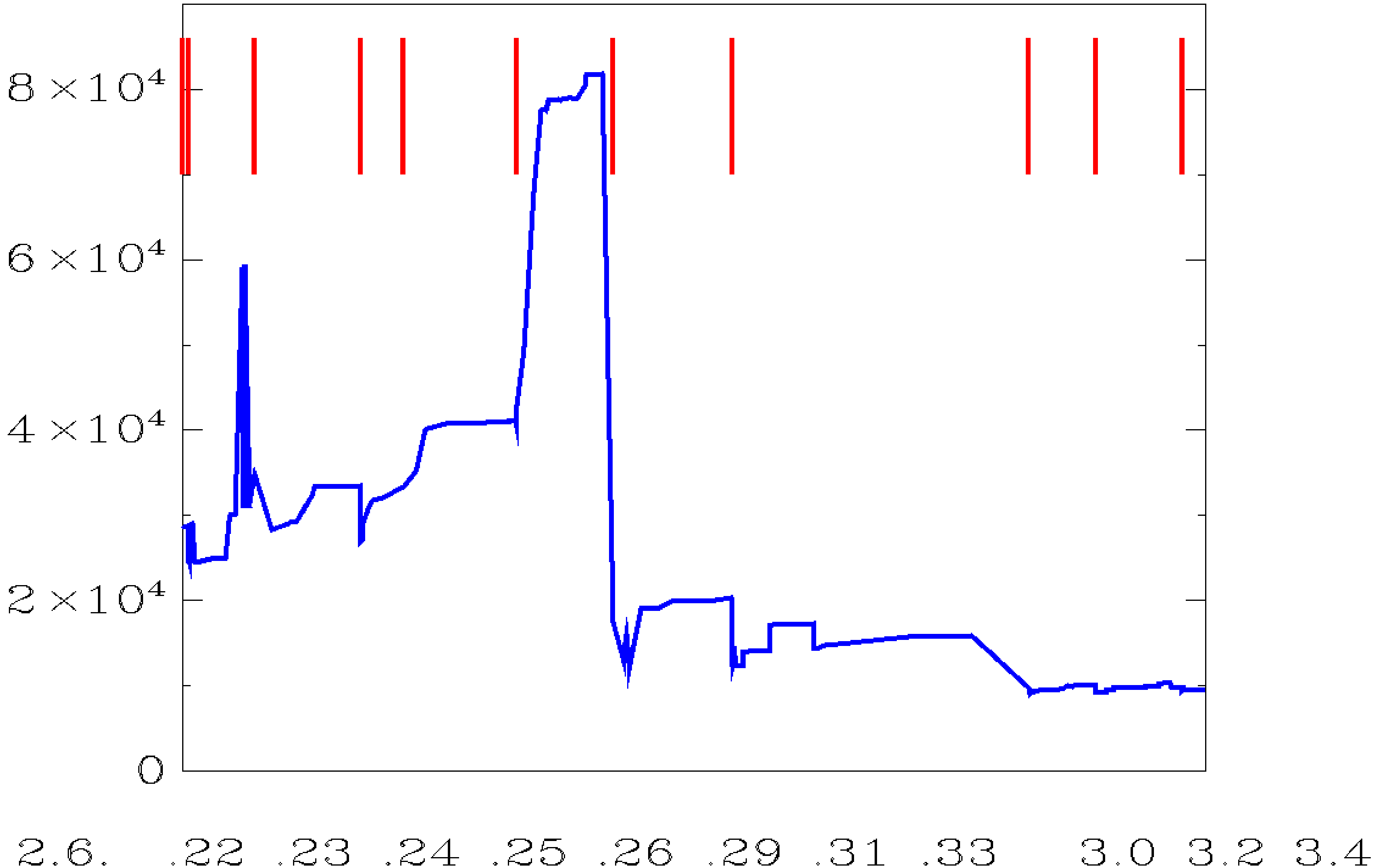
```
# producer cpu map:
#
#       0 00101010100000000001001111111111111111
#      70 11111111111111111111111111111111111111
#     140 00000000000011111111111111111111111111
#
# consumer cpu map:
#
#       0 00101010100000000001001111111111111111
#      70 11111111111111111111111111111111111111
#     140 00000000000011111111111111111111111111
```

Occasional migration, but producer and consumer always on same cpu

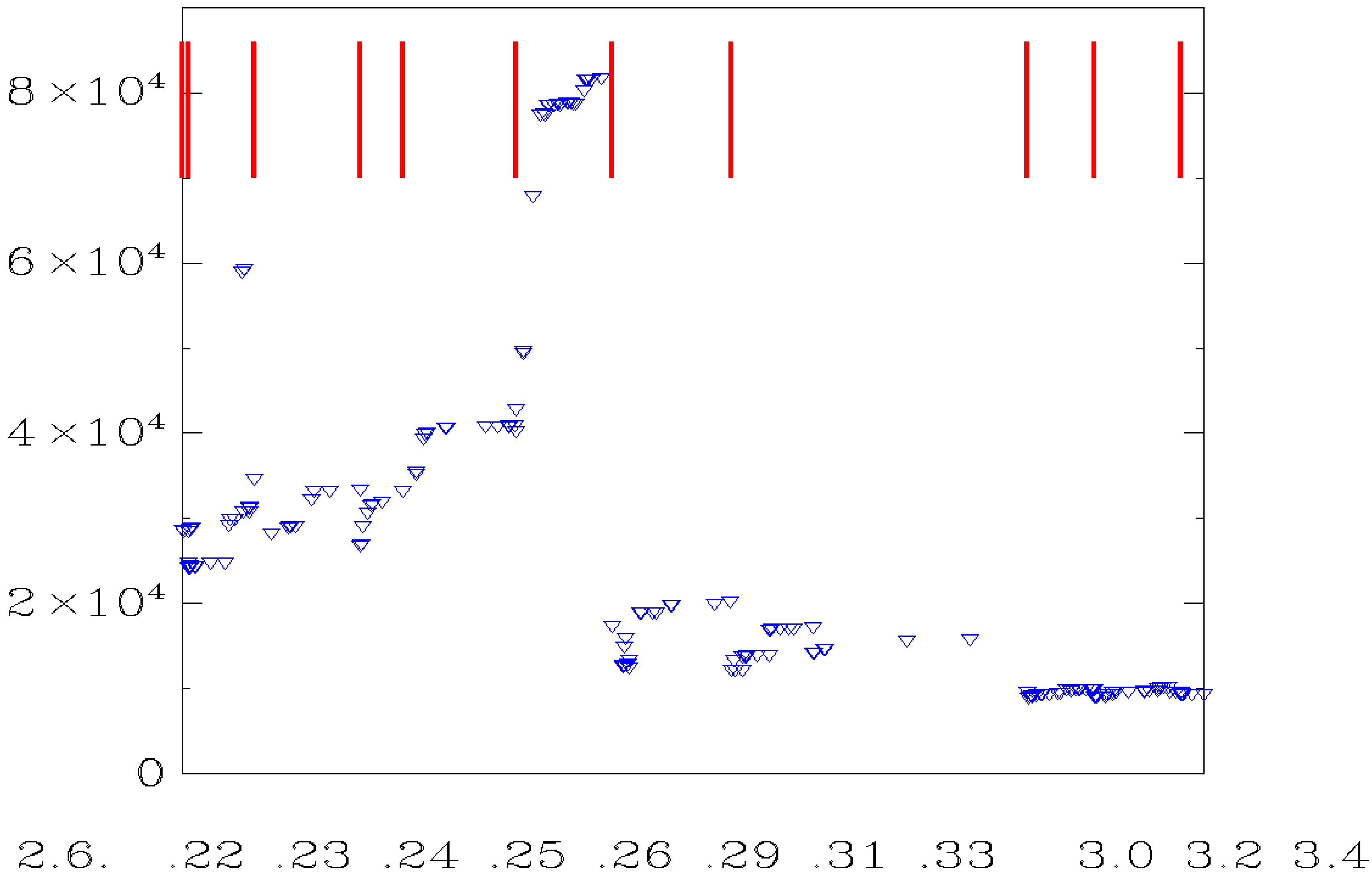
Lines vs Points

Which presentation provides a more informative graph?

blue: insertions



blue: insertions



Lines vs. Points

Lines can emphasize trends, changes in direction
Lines can hide detail

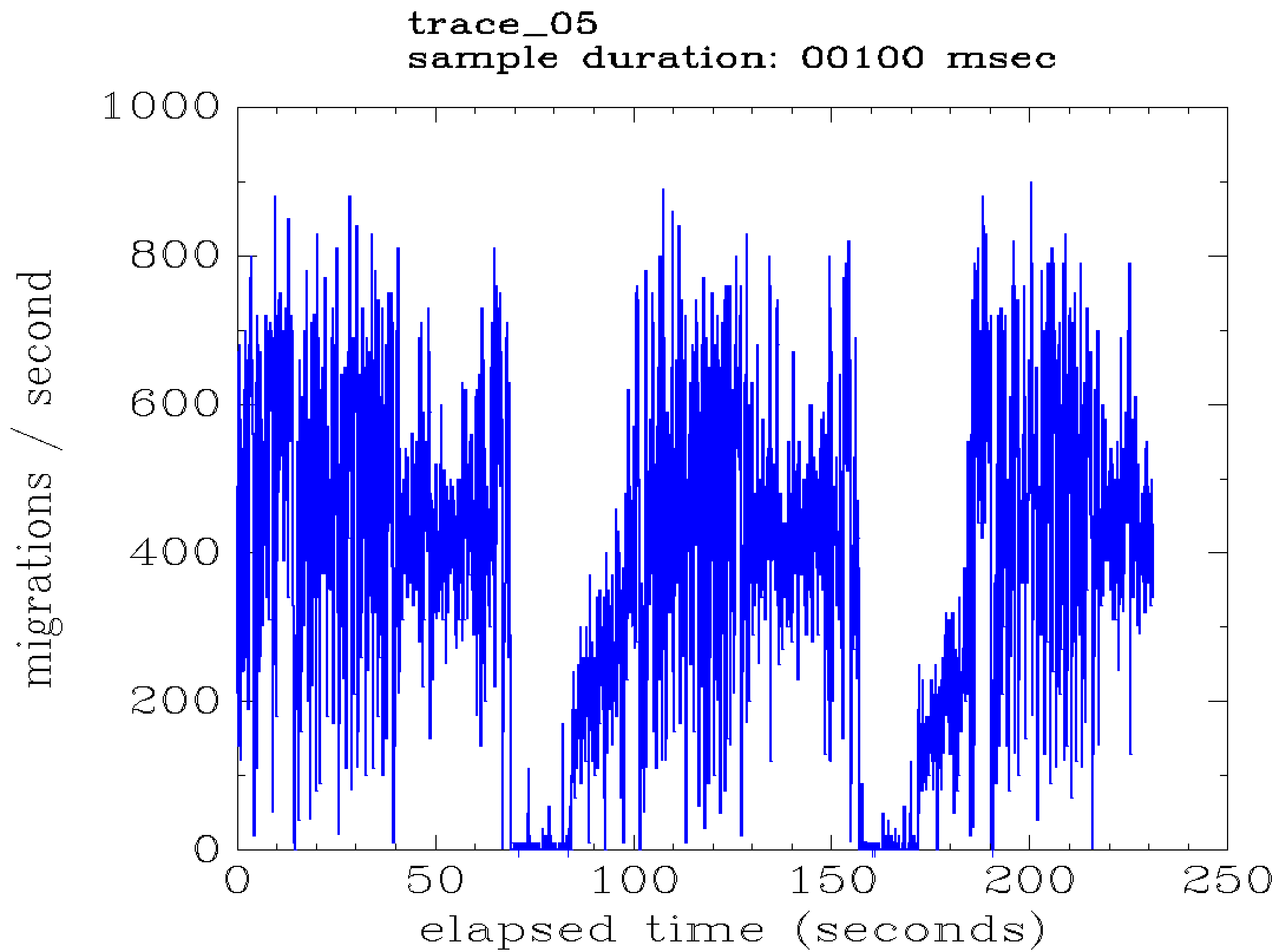
Points can expose detail

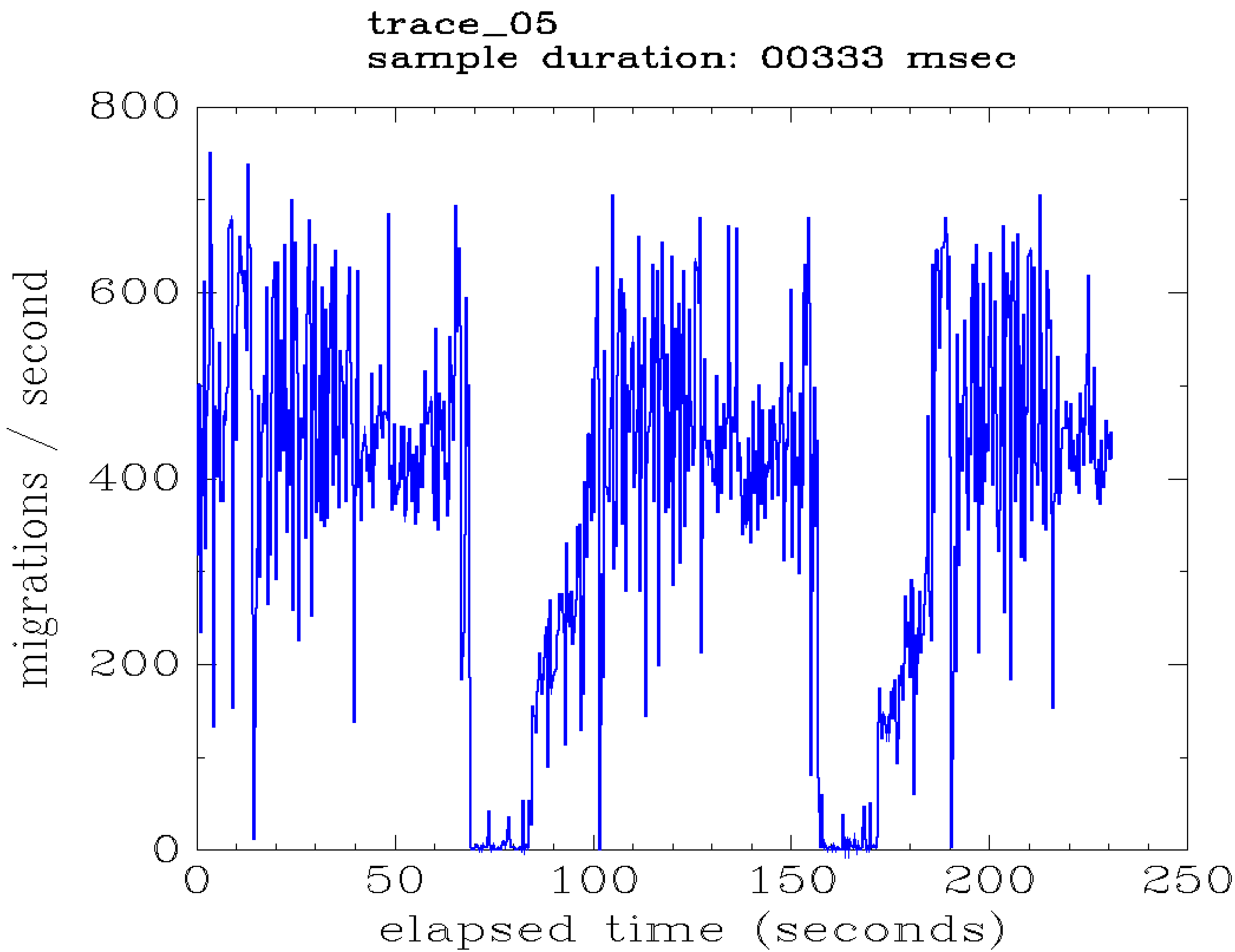
Data Example

Scheduler task migrations per second

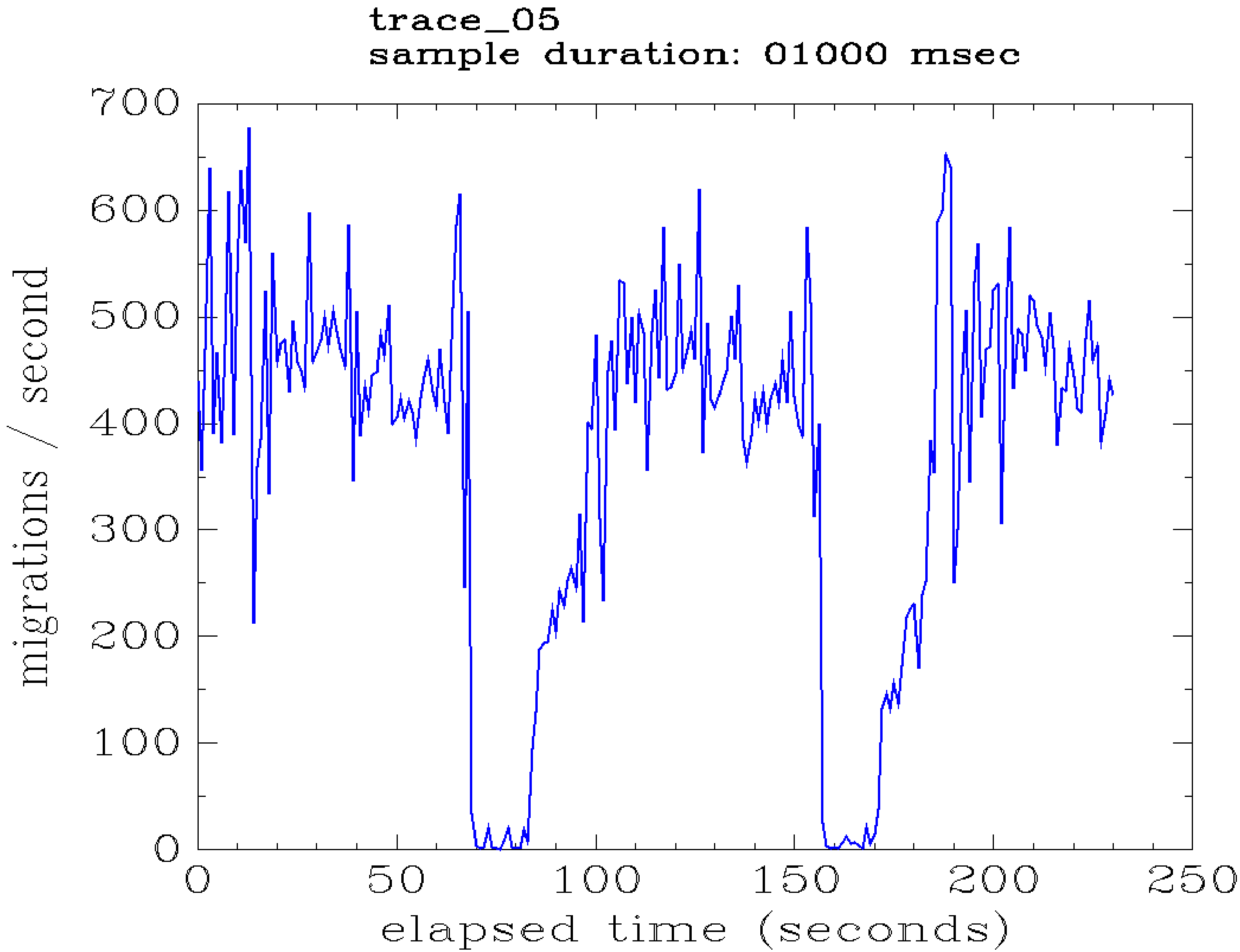
Sampling Period

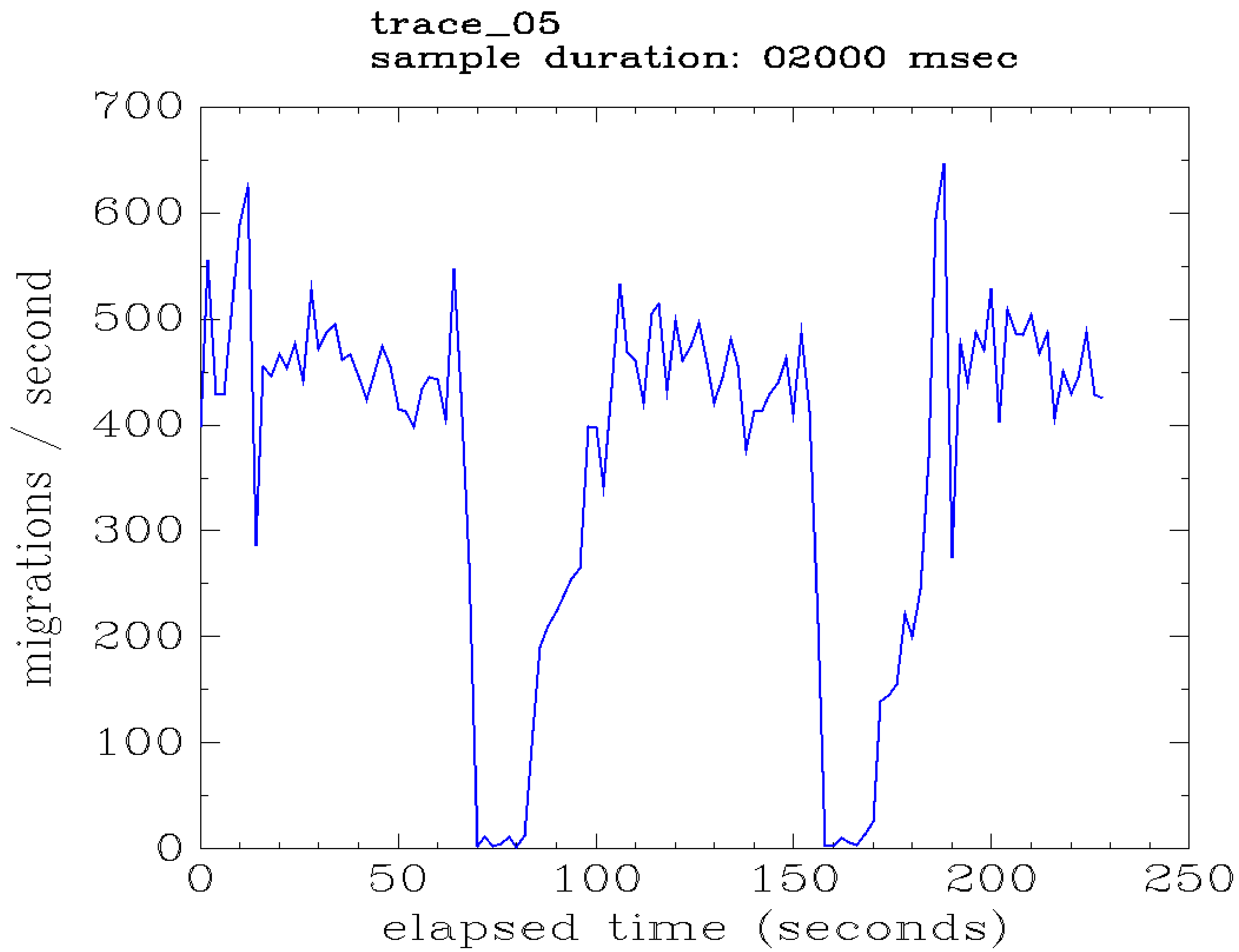
Number of events sampled per data point graphed

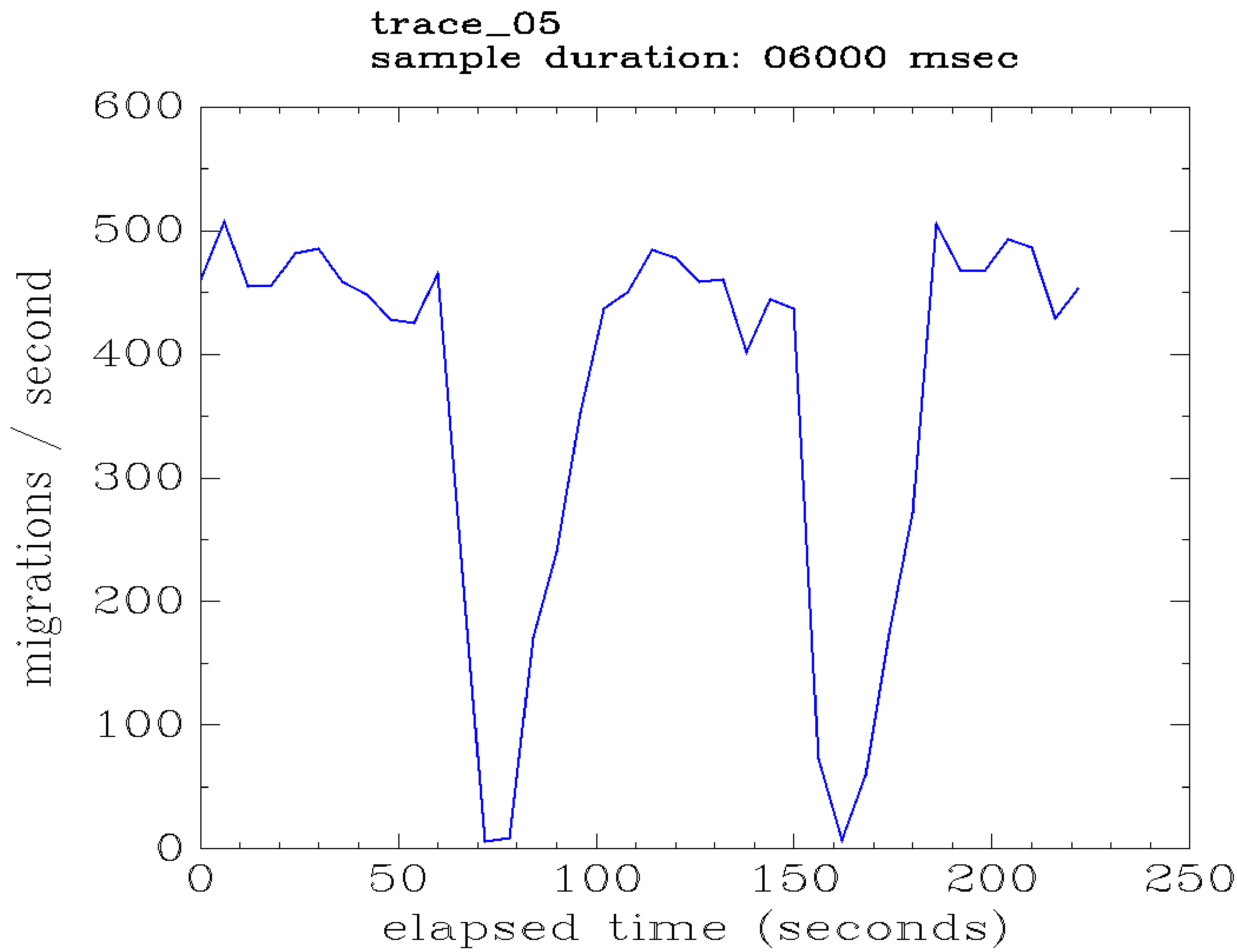

trace_05
sample duration: 00100 msec
1000
800
600
400
200
0
0
50
100
150
200
250
migrations / second
elapsed time (seconds)

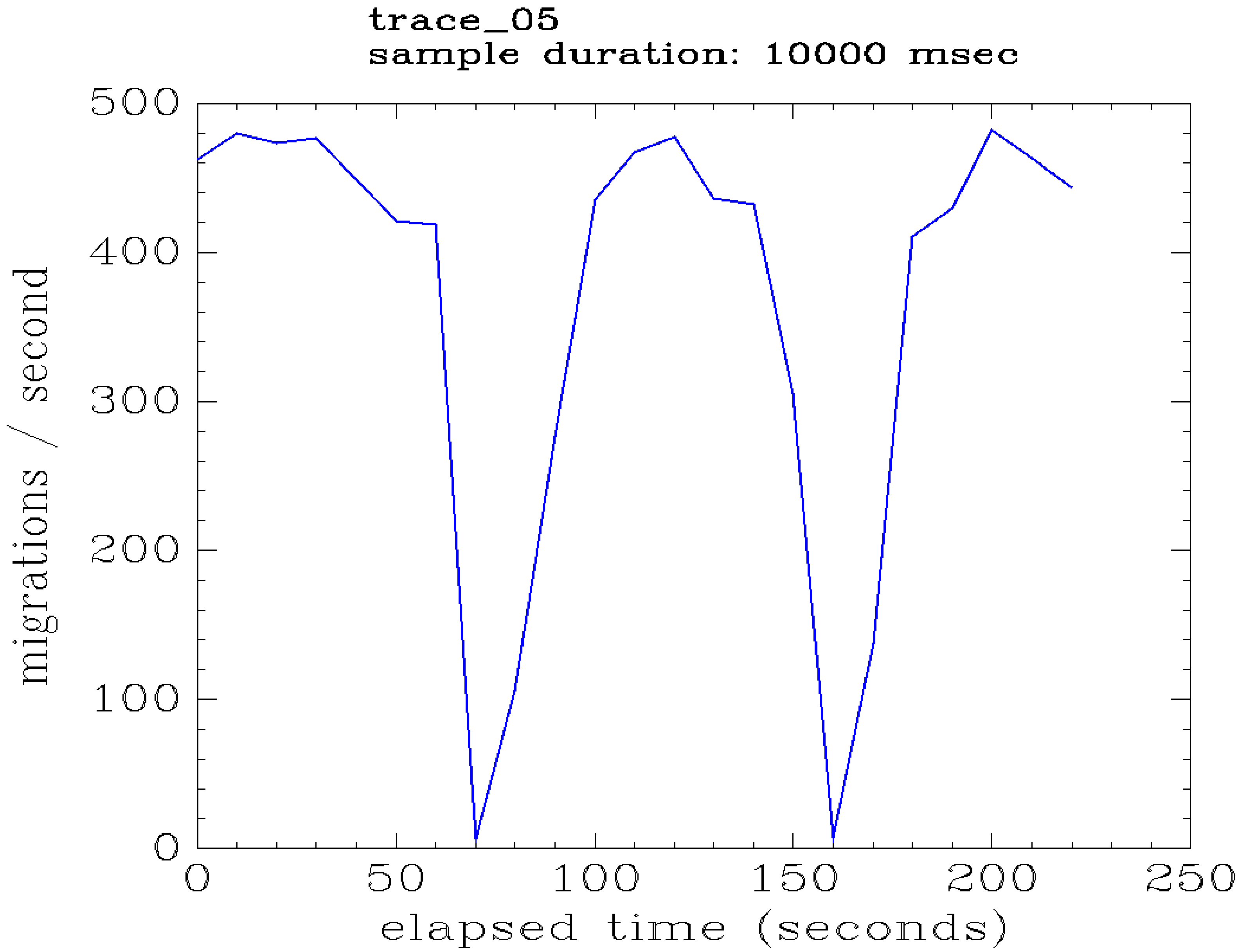


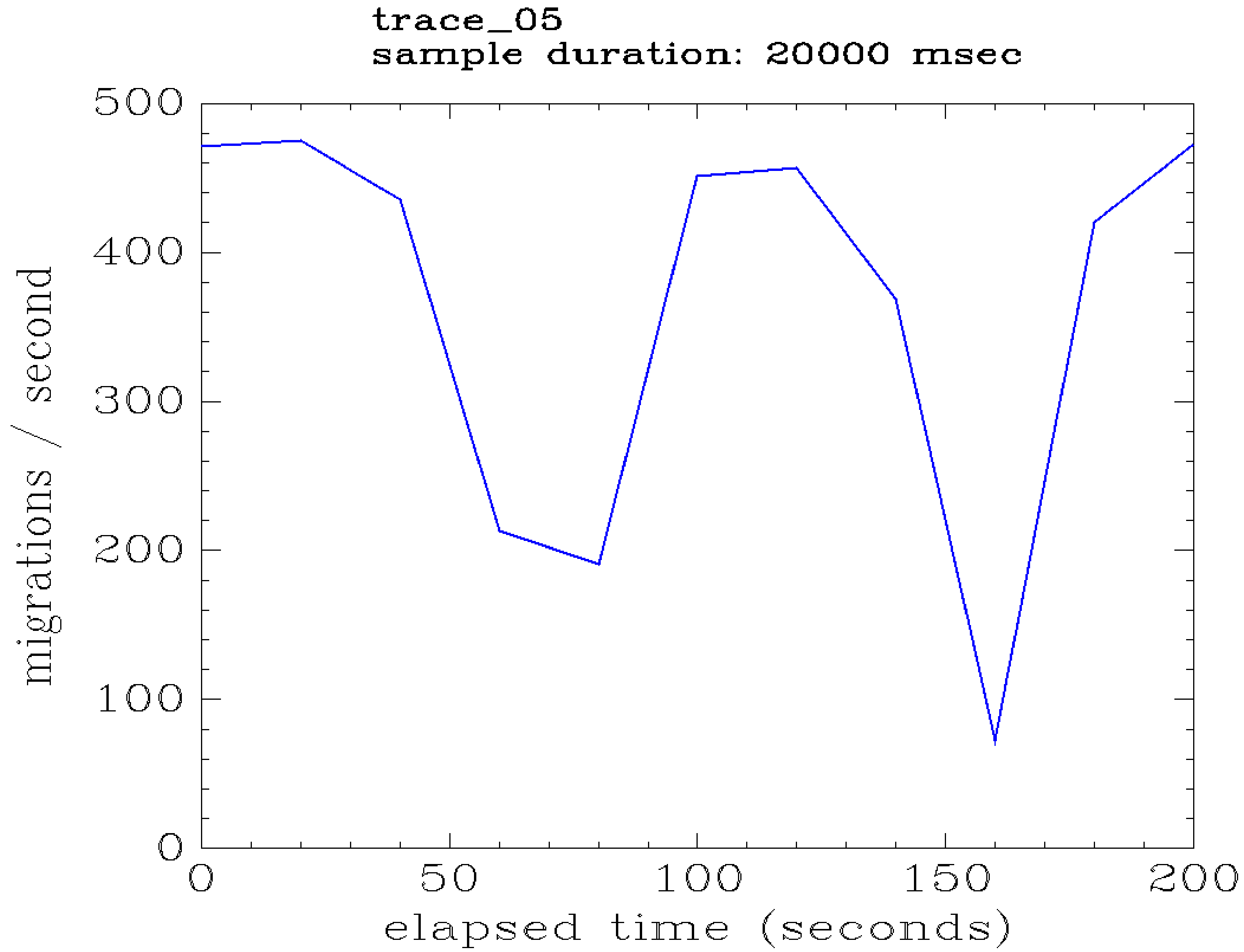
trace_05
sample duration: 00333 msec
800
600
400
200
0
migrations / second
0
50
100
150
200
250
elapsed time (seconds)

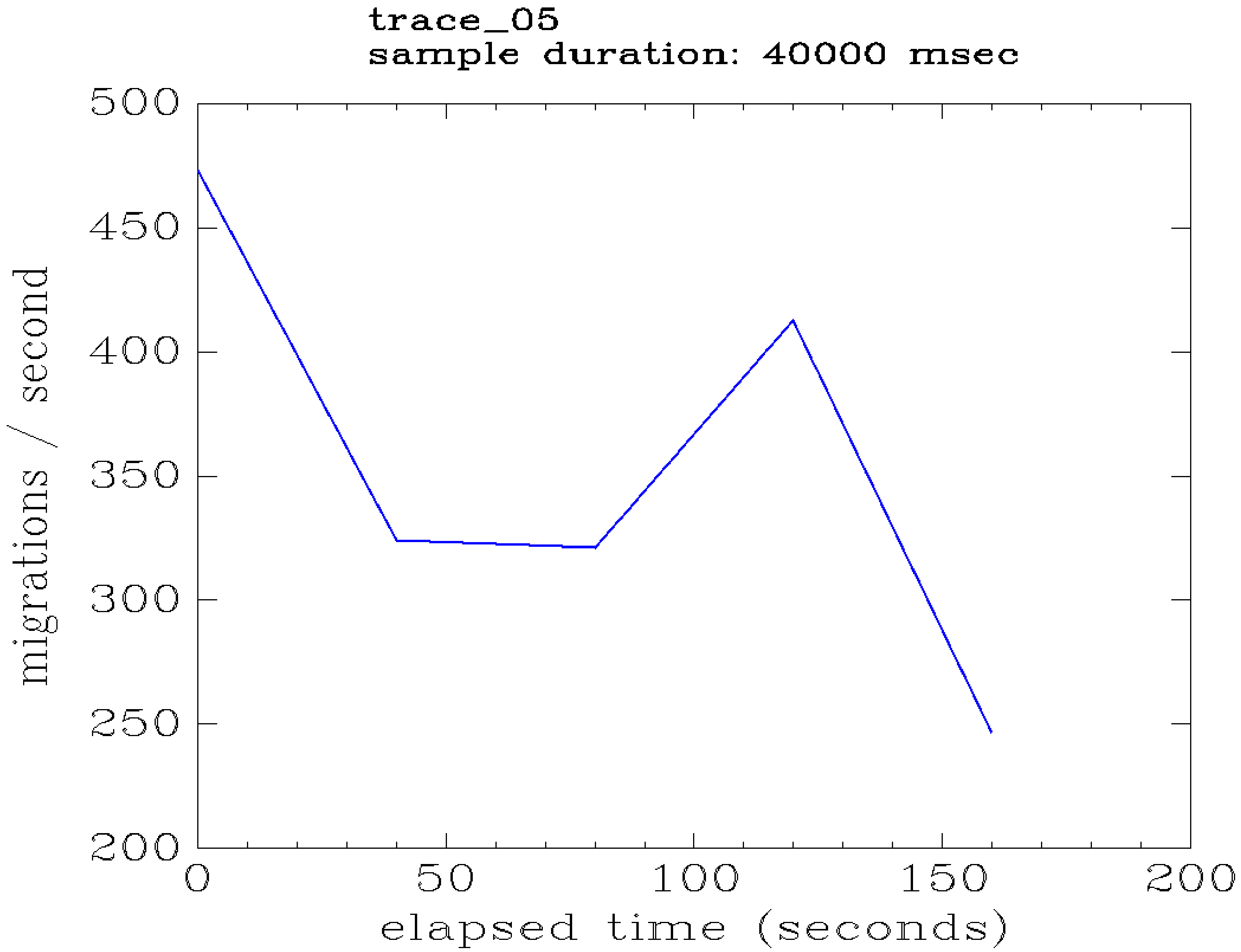

trace_05
sample duration: 01000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
600
700
0
50
100
150
200
250


trace_05
sample duration: 02000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
600
700
0
50
100
150
200
250


trace_05
sample duration: 06000 msec
600
500
400
300
200
100
0
0
50
100
150
200
250
migrations / second
elapsed time (seconds)

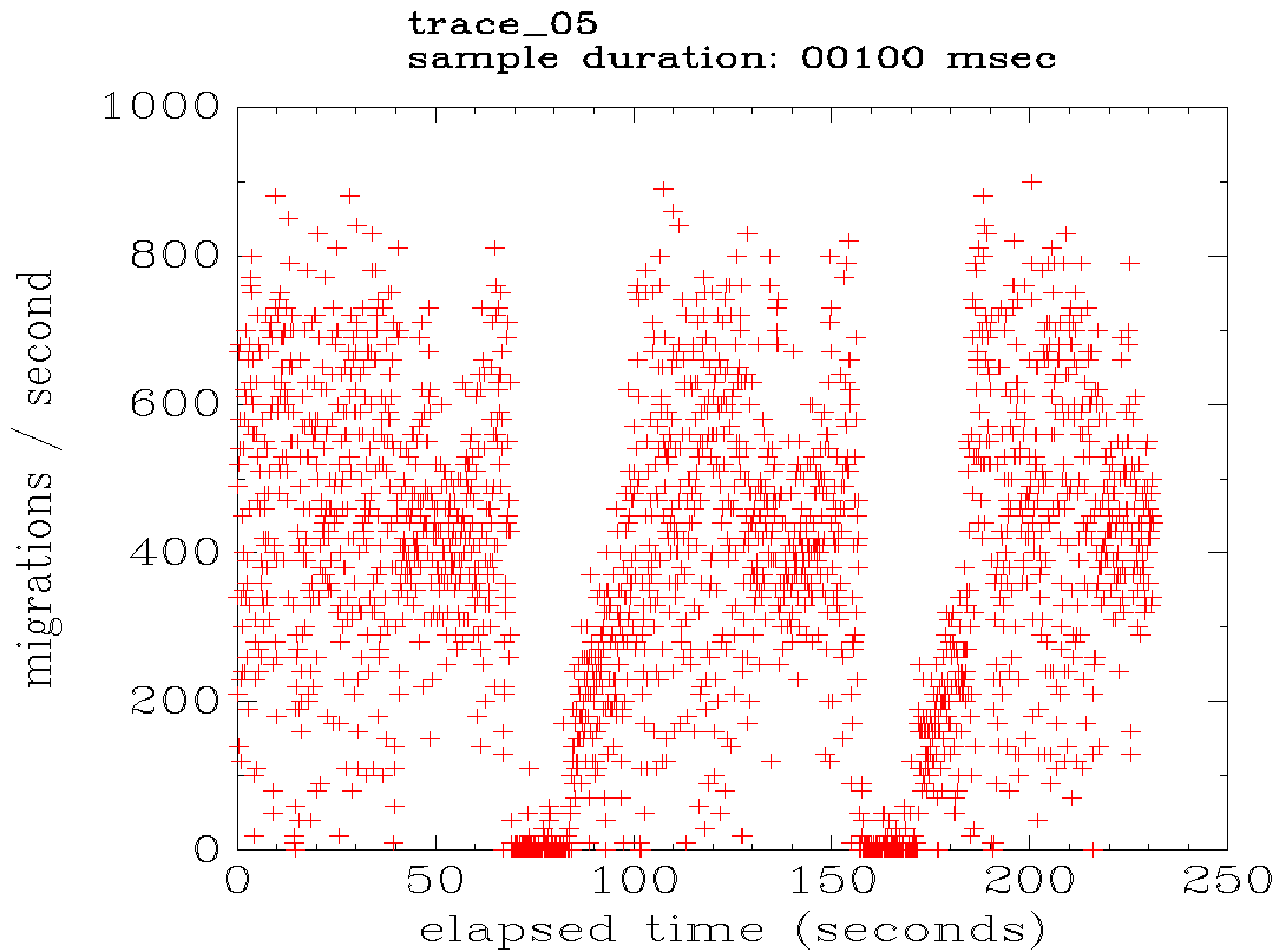

trace_05
sample duration: 10000 msec
500
400
300
200
100
0
0
50
100
150
200
250
migrations / second
elapsed time (seconds)

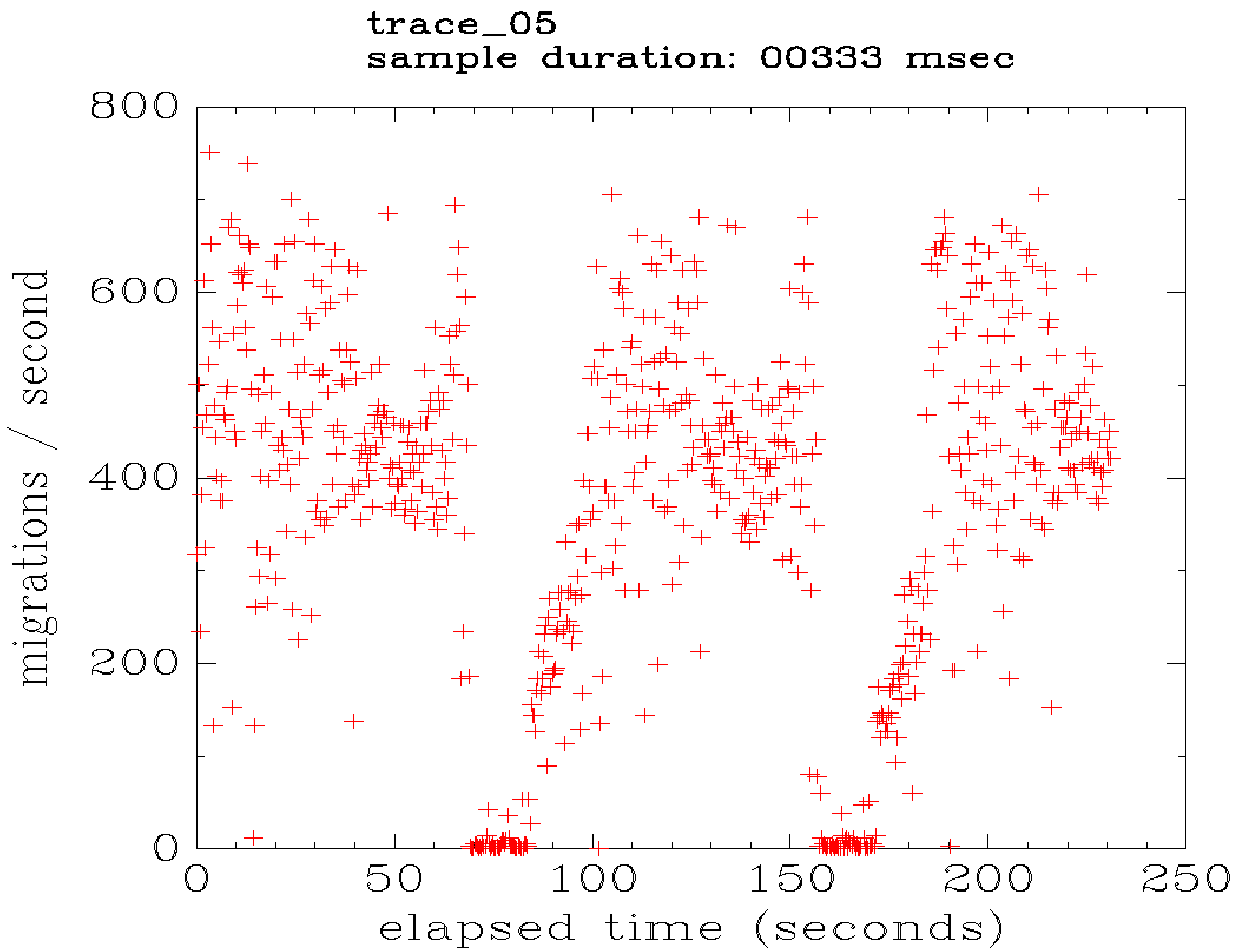

trace_05
sample duration: 20000 msec
500
400
300
200
100
0
0
50
100
150
200
migrations / second
elapsed time (seconds)

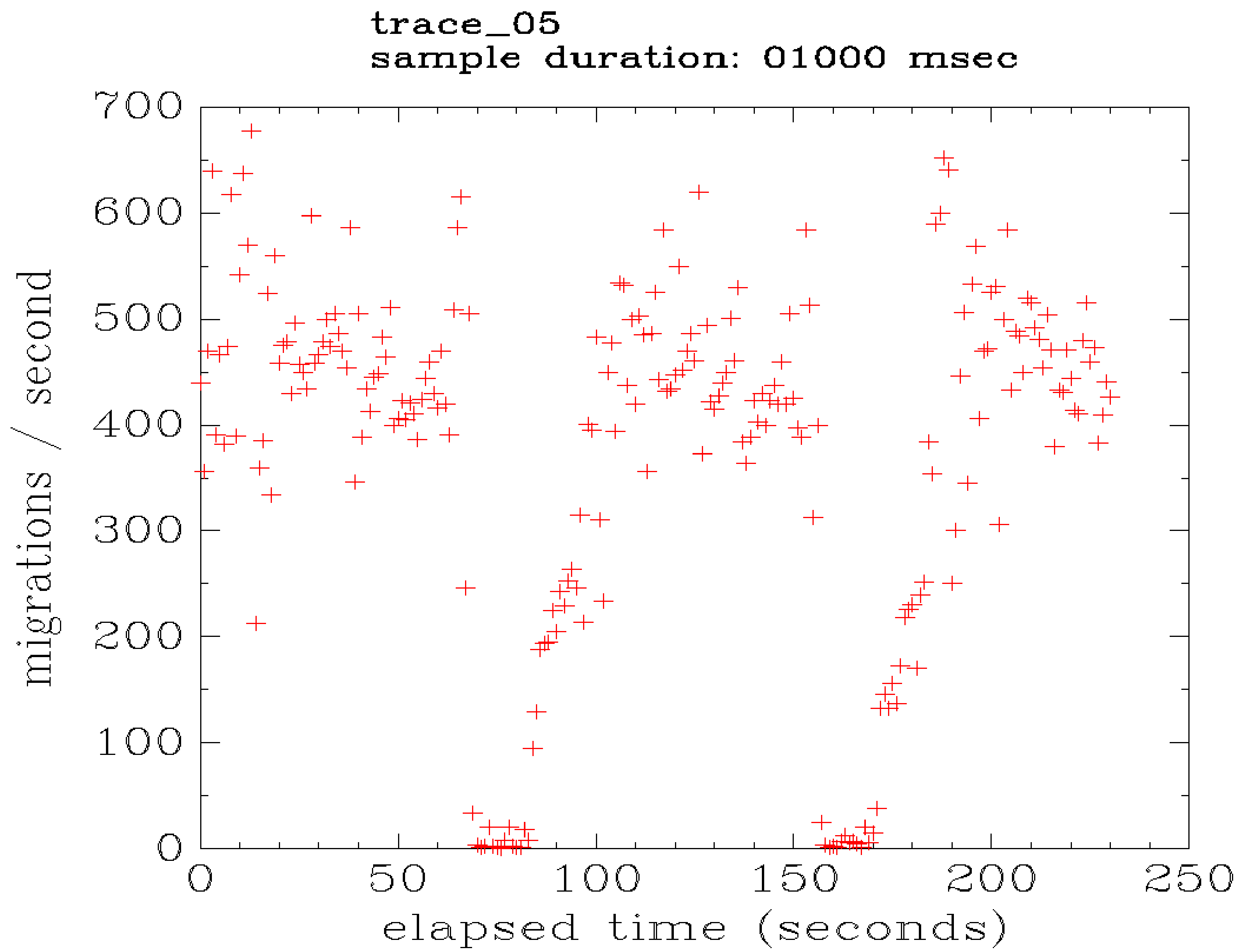

trace_05
sample duration: 40000 msec
500
450
400
350
300
250
200
0
50
100
150
200
migrations / second
elapsed time (seconds)

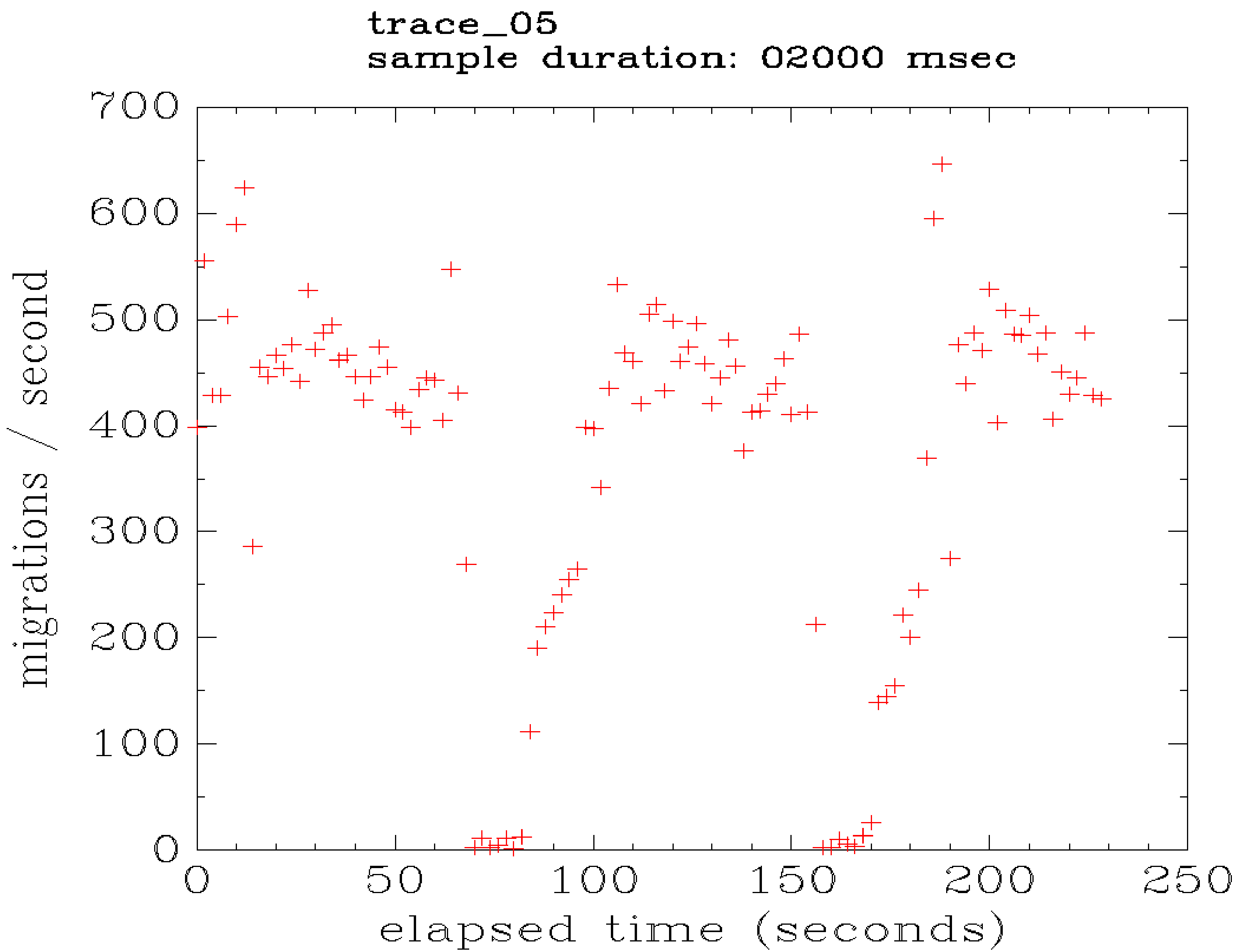
Lines vs Points

another example

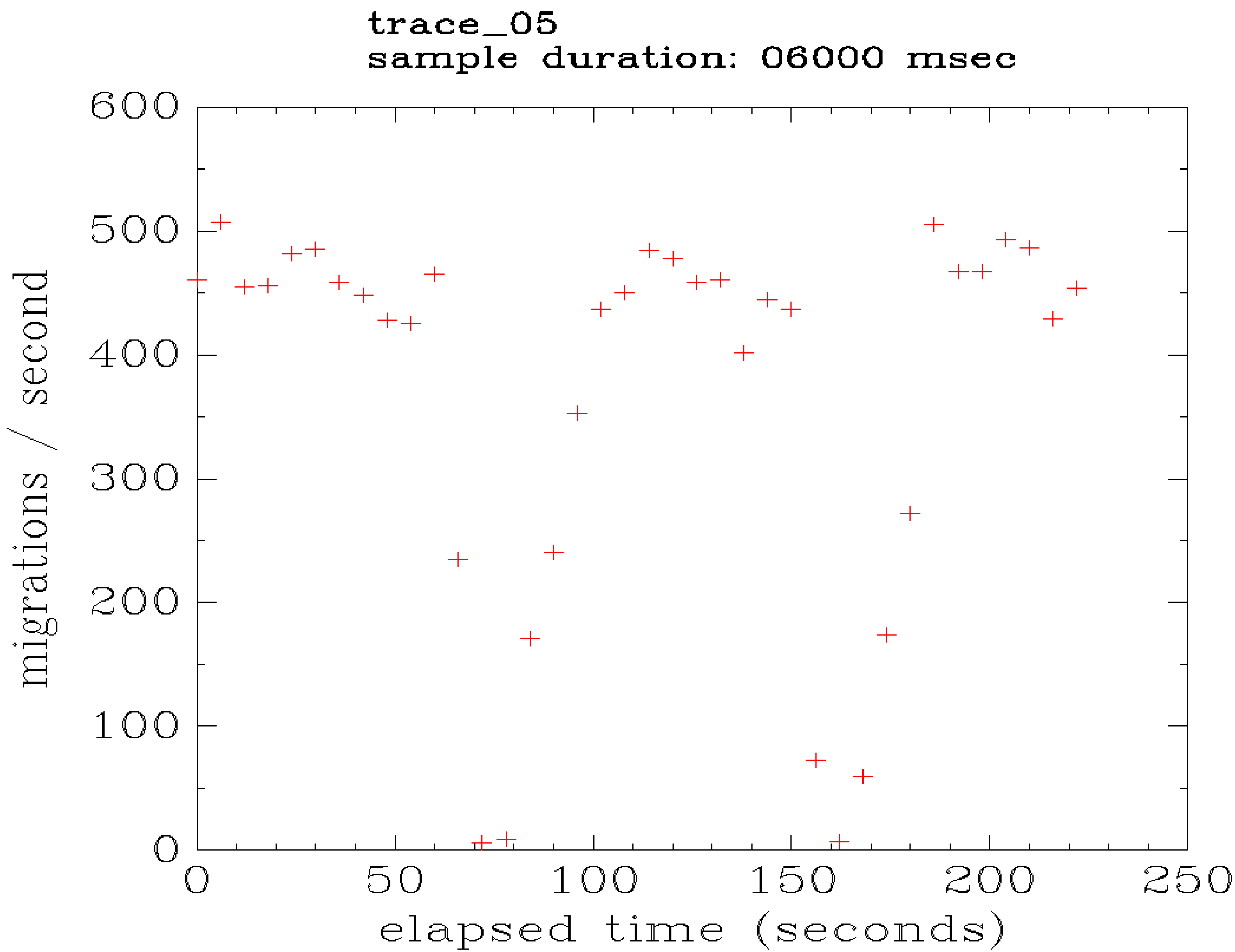

trace_05
sample duration: 00100 msec
migrations / second
elapsed time (seconds)
0
200
400
600
800
1000
0
50
100
150
200
250

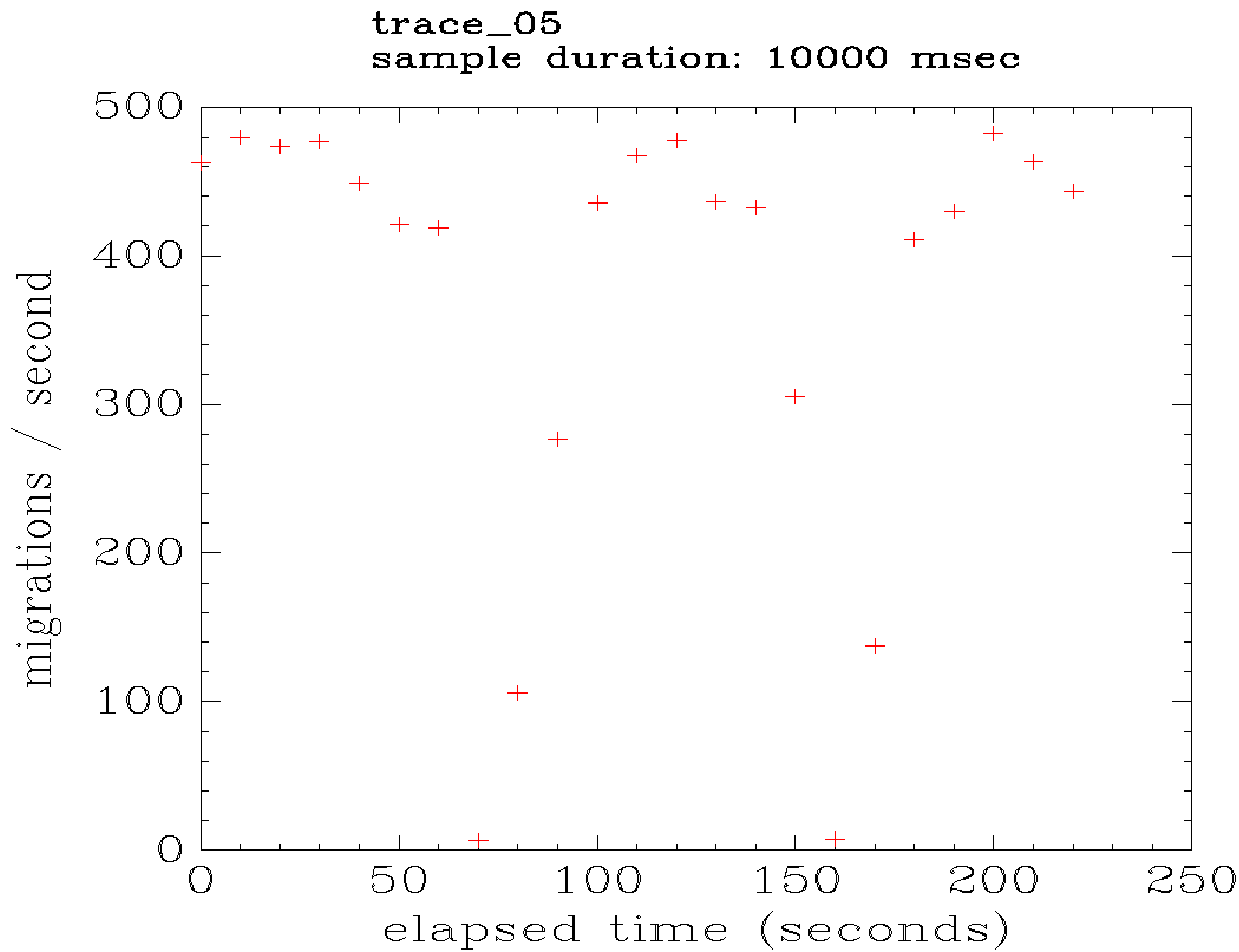

trace_05
sample duration: 00333 msec
800
600
400
200
0
0
50
100
150
200
250
migrations / second
elapsed time (seconds)

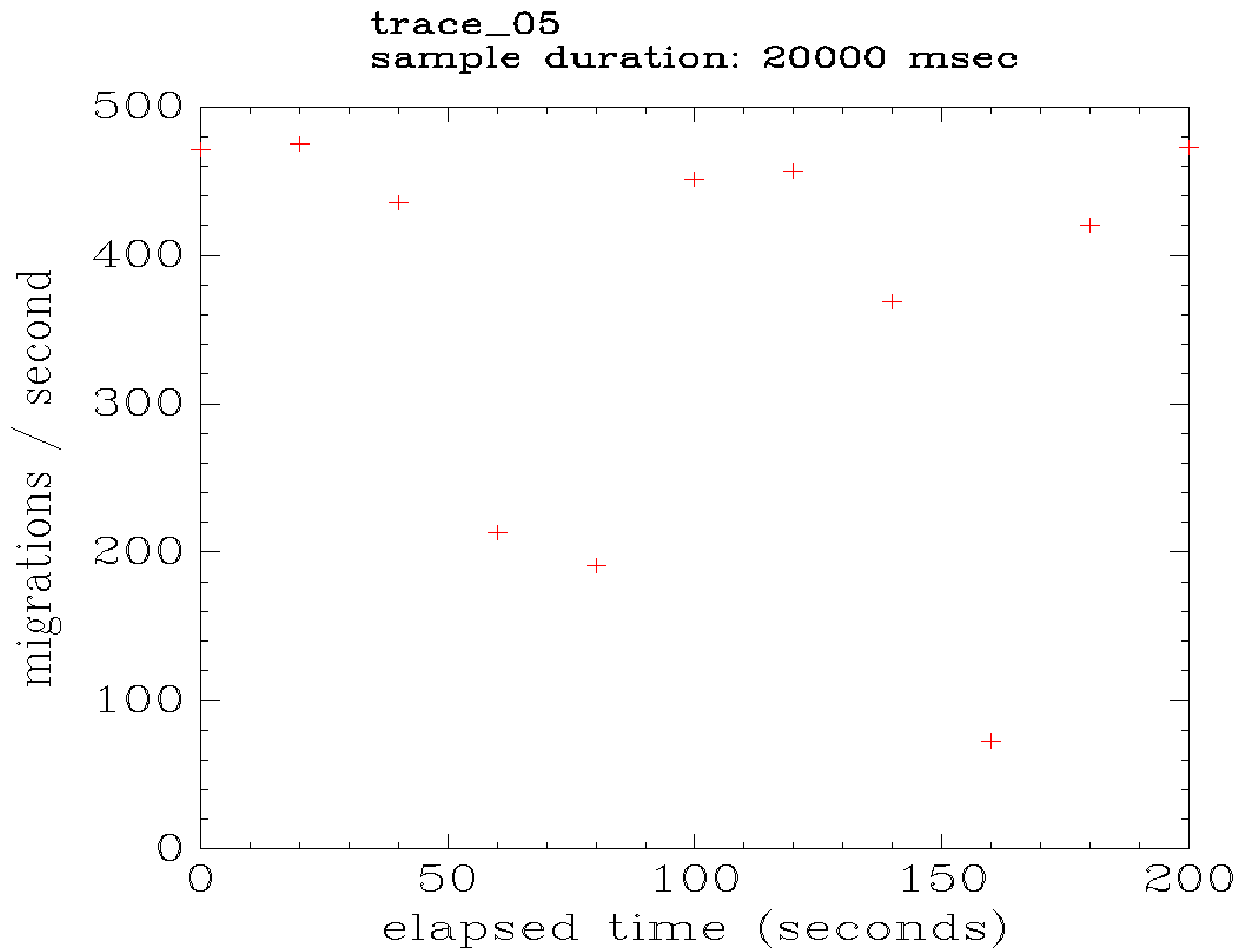

trace_05
sample duration: 01000 msec
700
600
500
400
300
200
100
0
migrations / second
0
50
100
150
200
250
elapsed time (seconds)

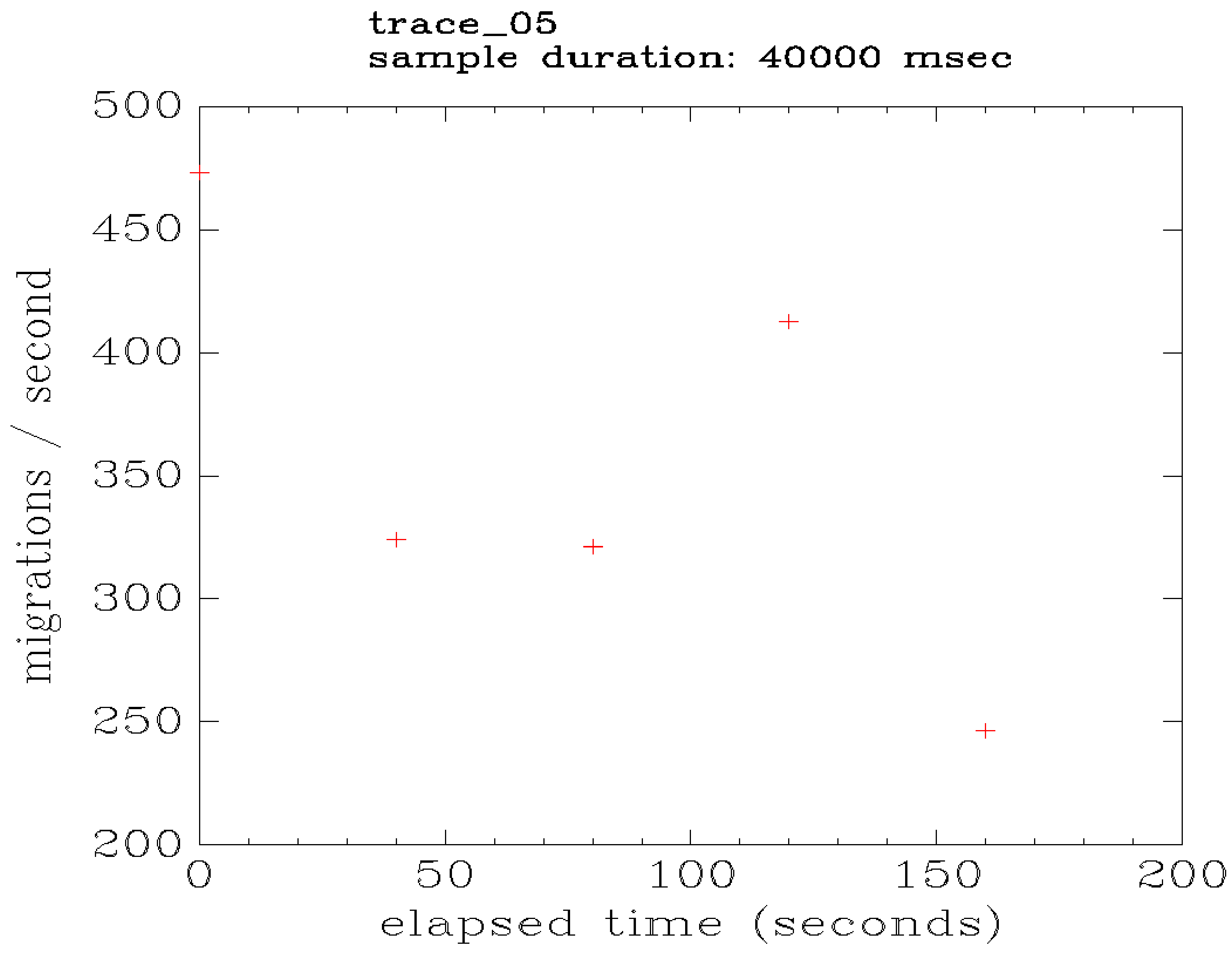


trace_05
sample duration: 02000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
600
700
0
50
100
150
200
250


trace_05
sample duration: 06000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
600
0
50
100
150
200
250


trace_05
sample duration: 10000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
0
50
100
150
200
250

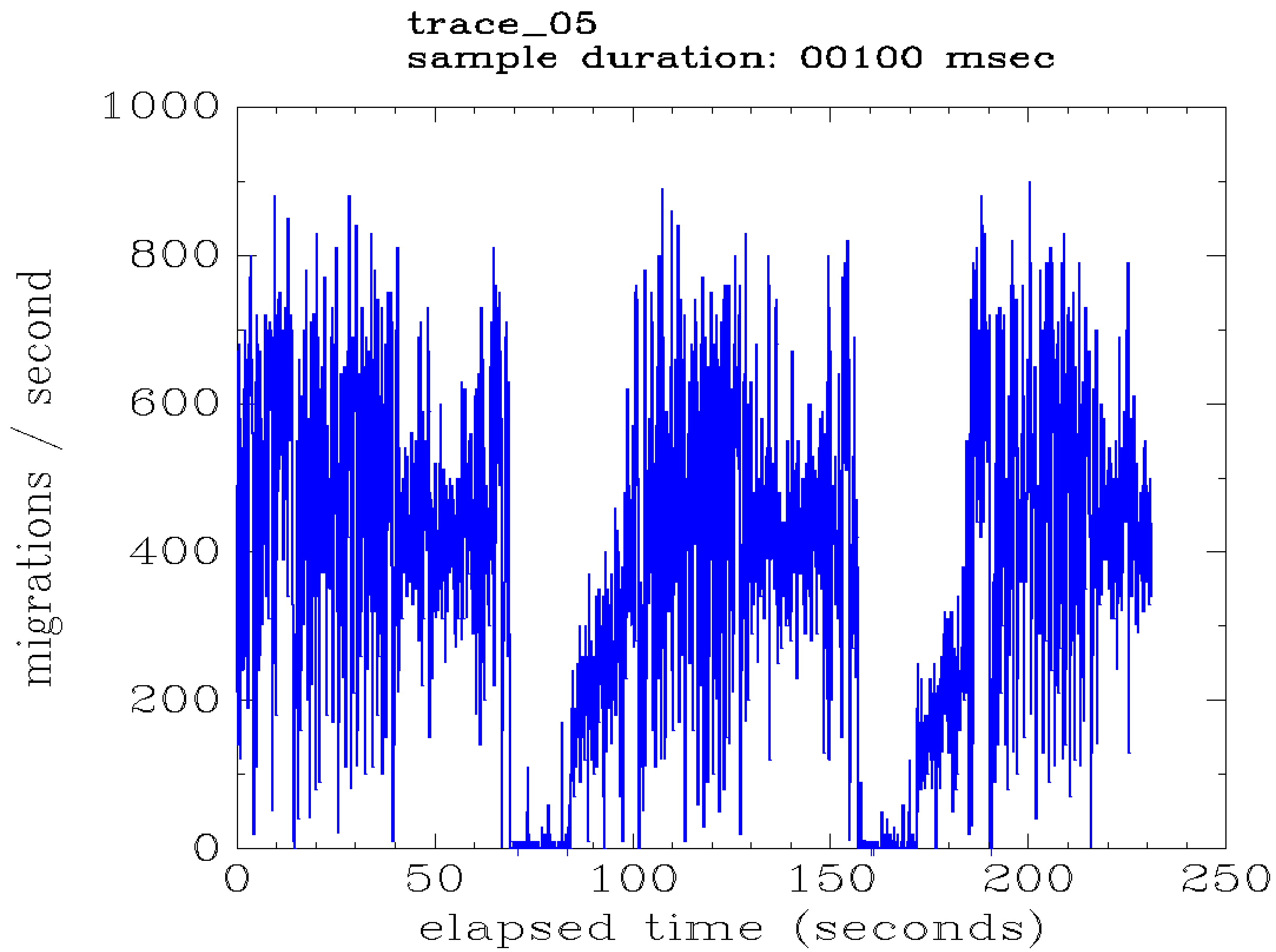

trace_05
sample duration: 20000 msec
500
400
300
200
100
0
migrations / second
0
50
100
150
200
elapsed time (seconds)

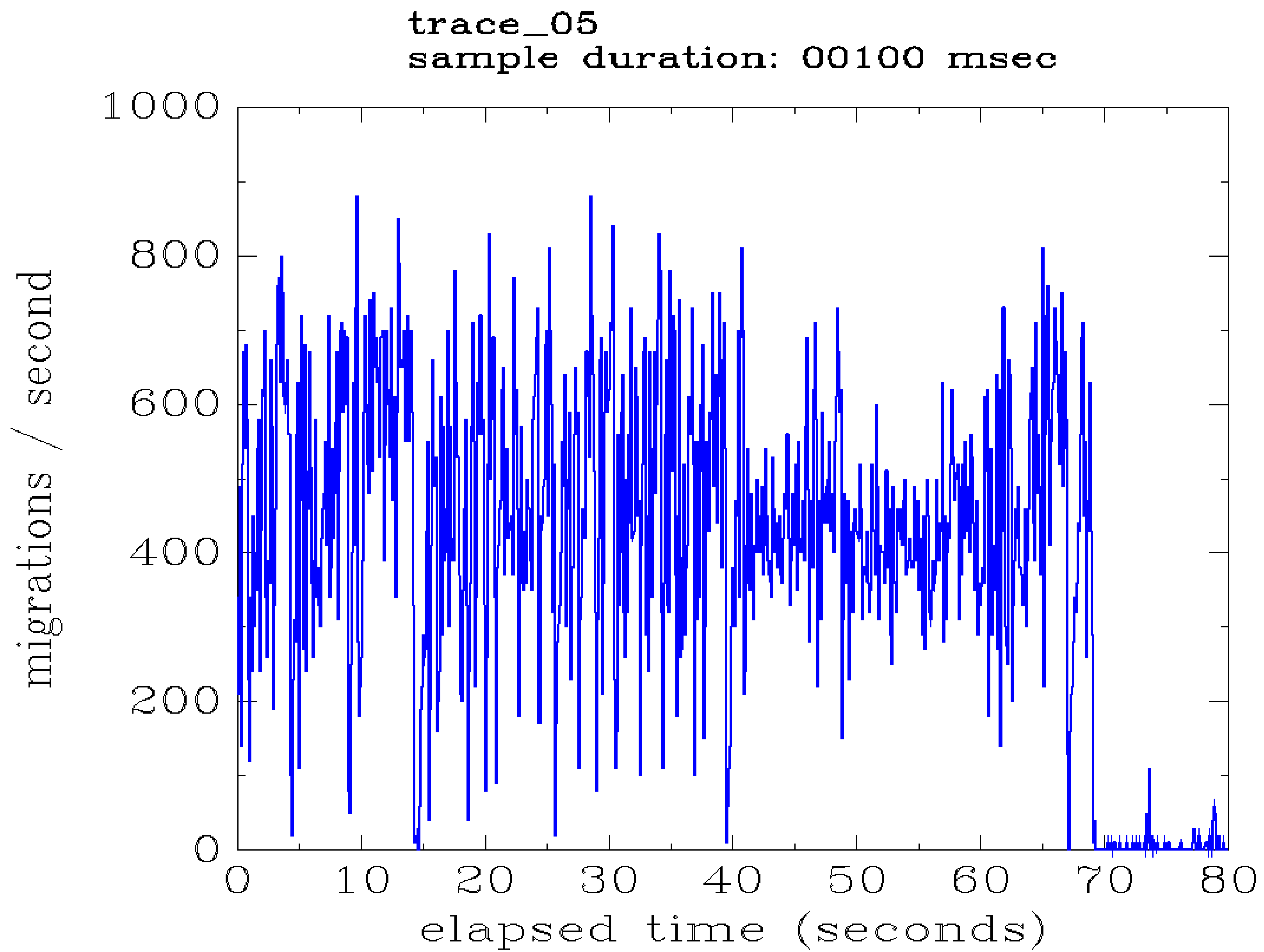


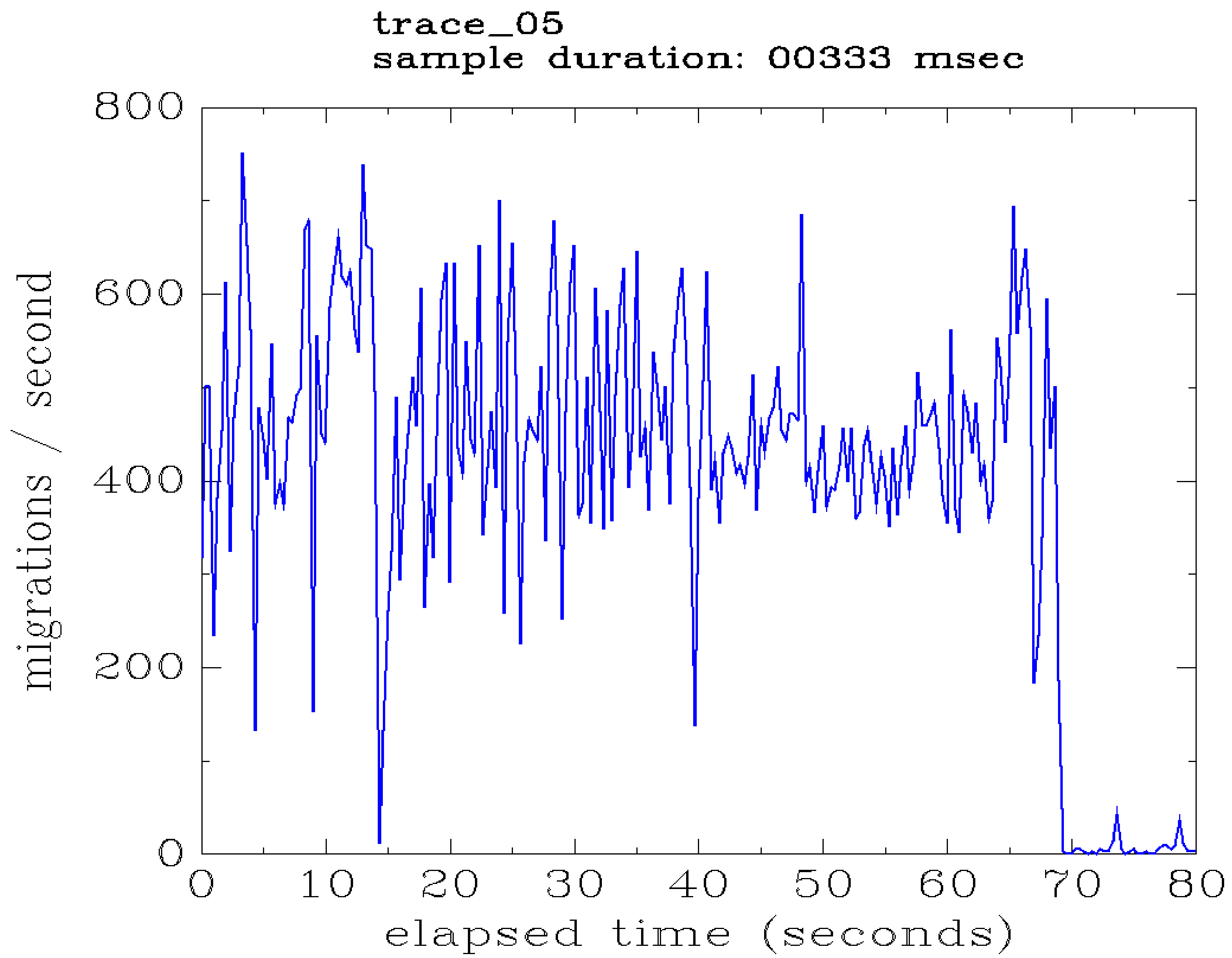
trace_05
sample duration: 40000 msec
500
450
400
350
300
250
200
0
50
100
150
200
elapsed time (seconds)
migrations / second

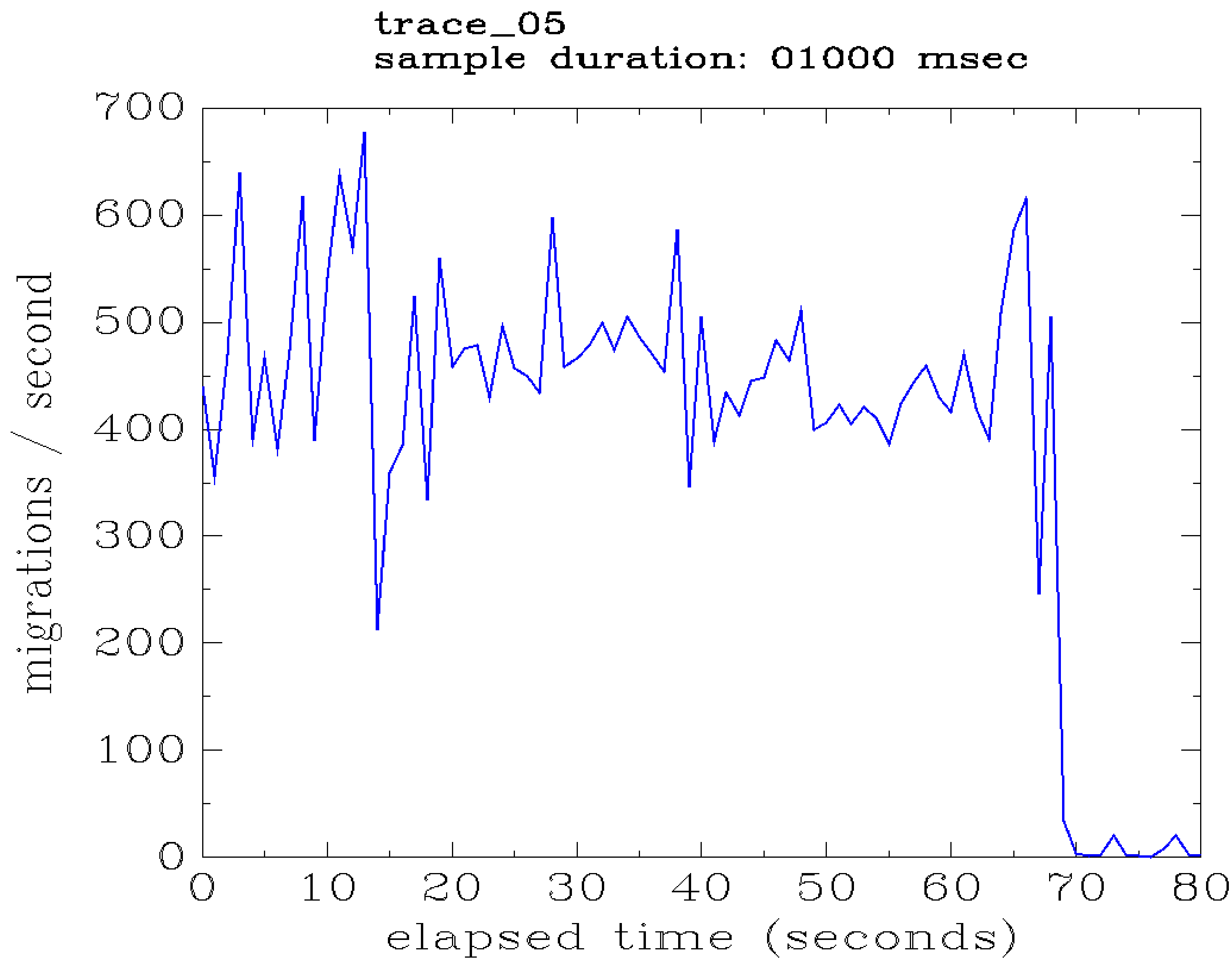
Rescale

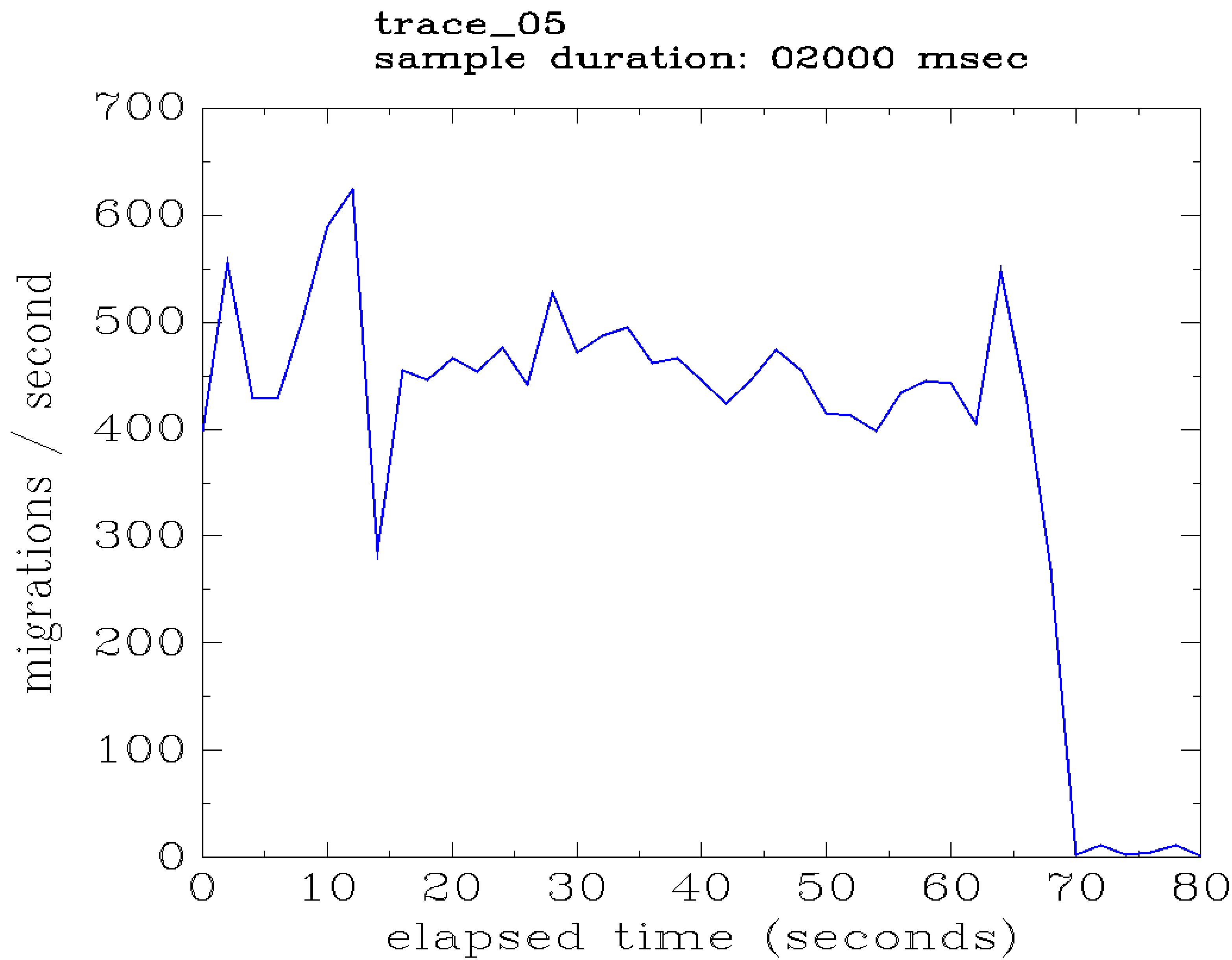
Change scale to reveal detail

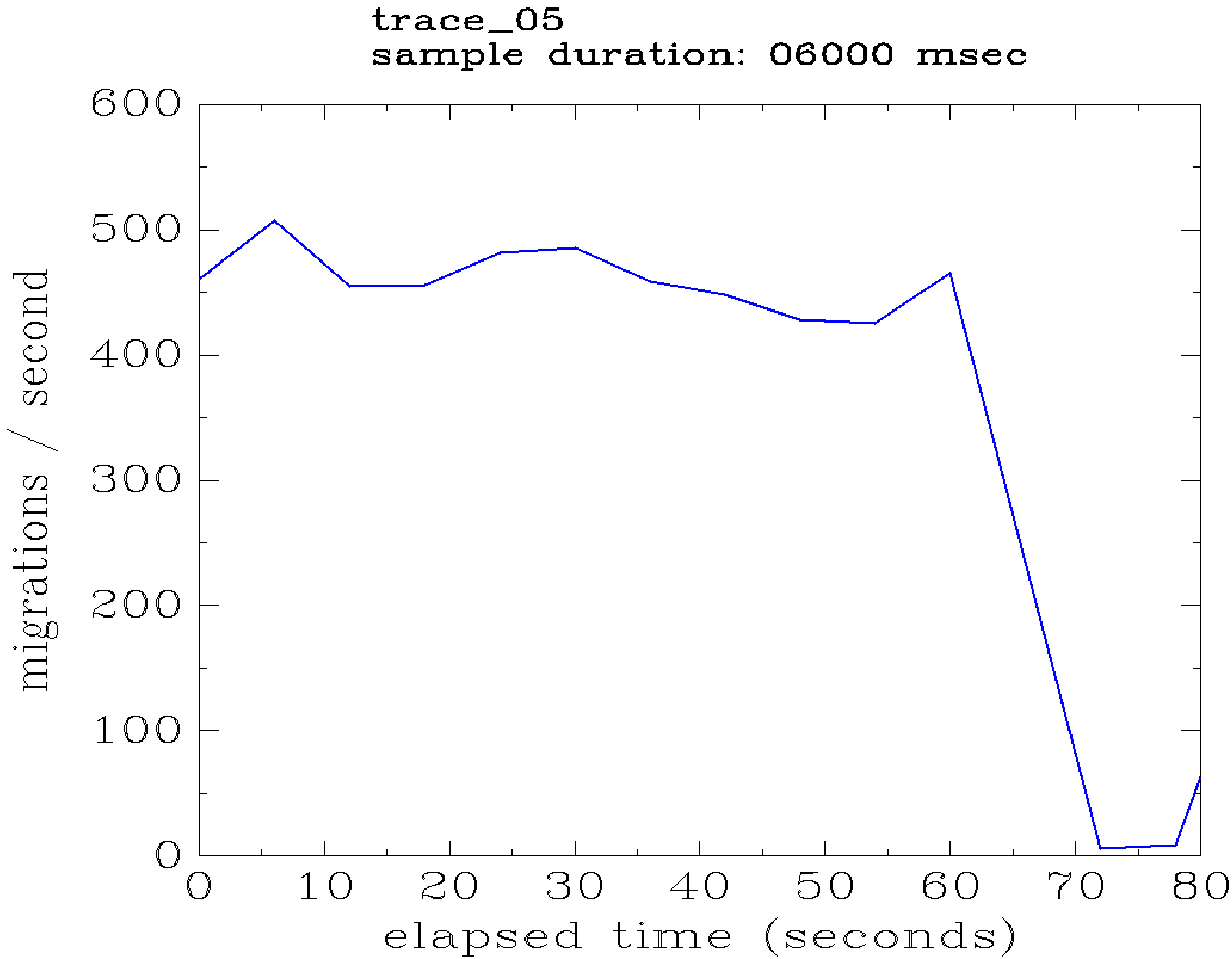

trace_05
sample duration: 00100 msec
1000
800
600
400
200
0
0
50
100
150
200
250
migrations / second
elapsed time (seconds)

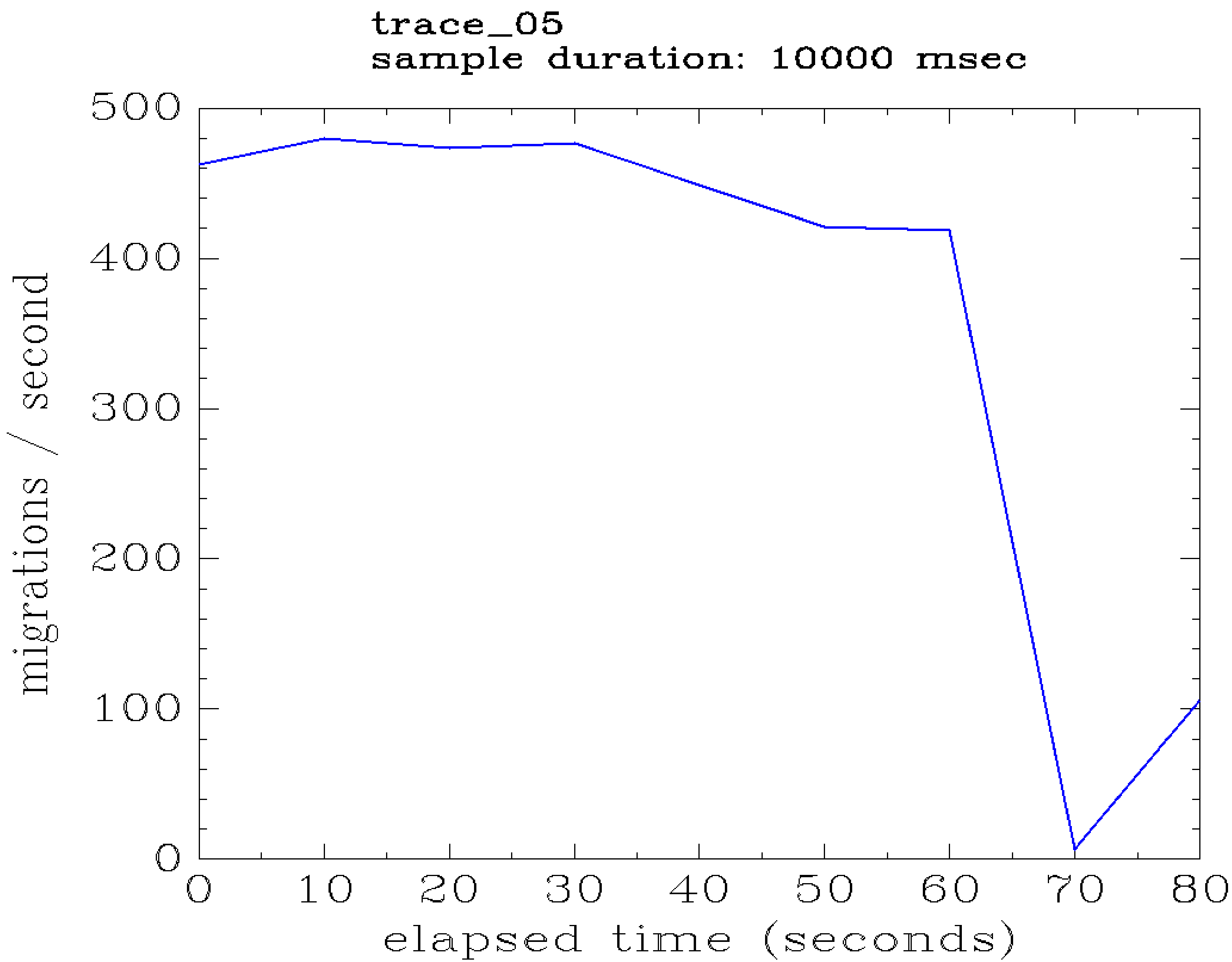

trace_05
sample duration: 00100 msec
migrations / second
elapsed time (seconds)
0
200
400
600
800
1000
0
10
20
30
40
50
60
70
80

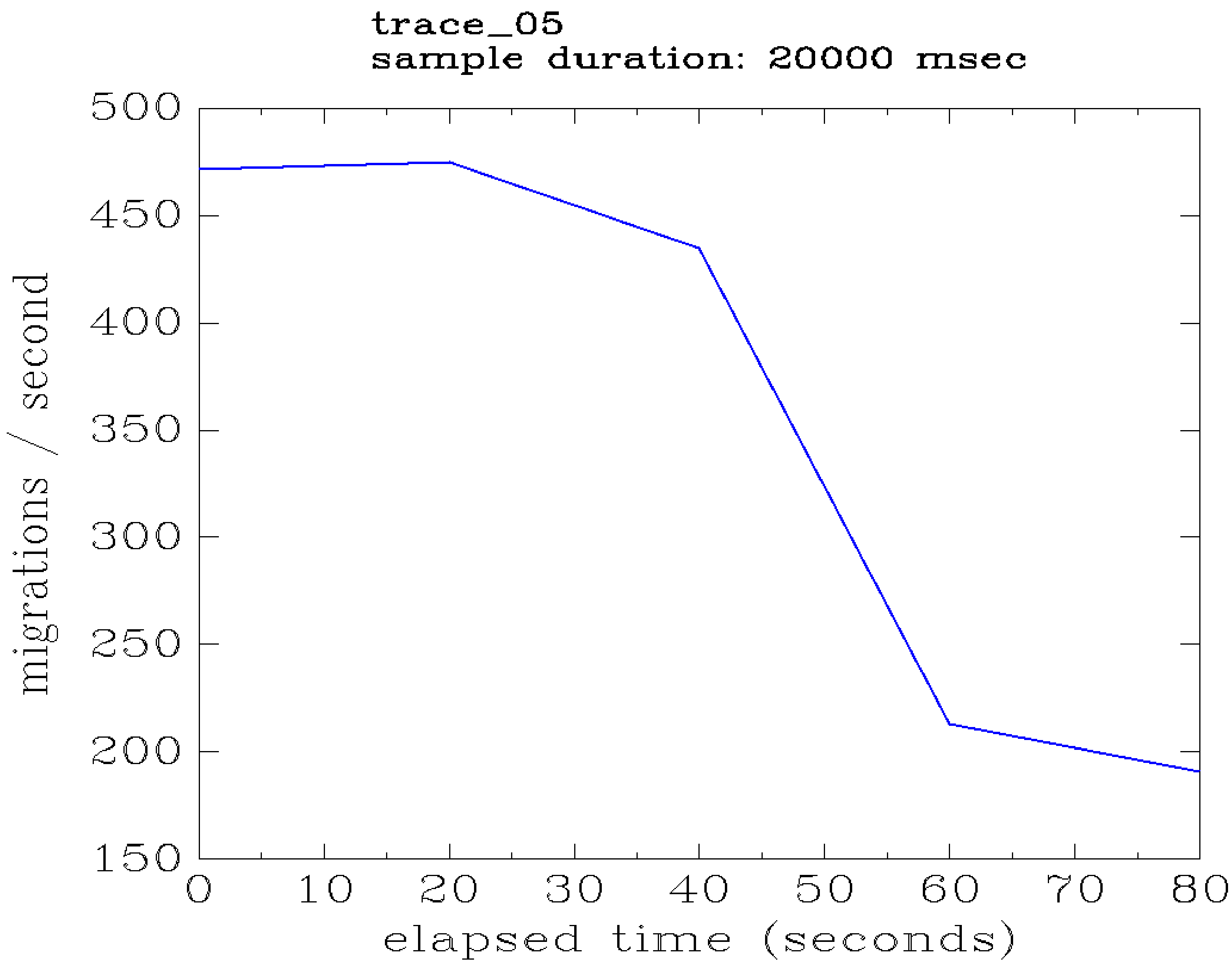

trace_05
sample duration: 00333 msec
800
600
400
200
0
migrations / second
0
10
20
30
40
50
60
70
80
elapsed time (seconds)

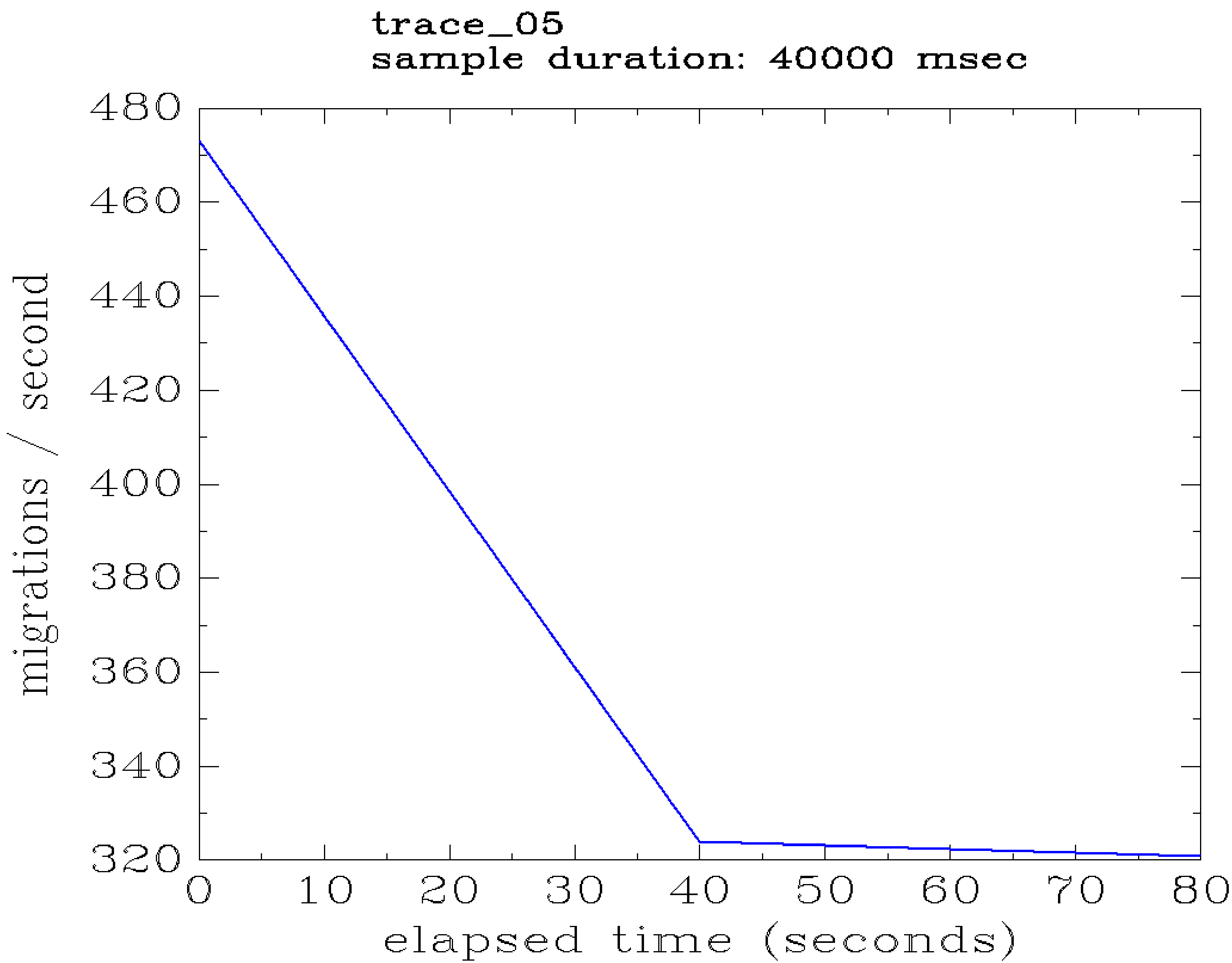

trace_05
sample duration: 01000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
600
700
0
10
20
30
40
50
60
70
80


trace_05
sample duration: 02000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
600
700
0
10
20
30
40
50
60
70
80


trace_05
sample duration: 06000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
600
0
10
20
30
40
50
60
70
80

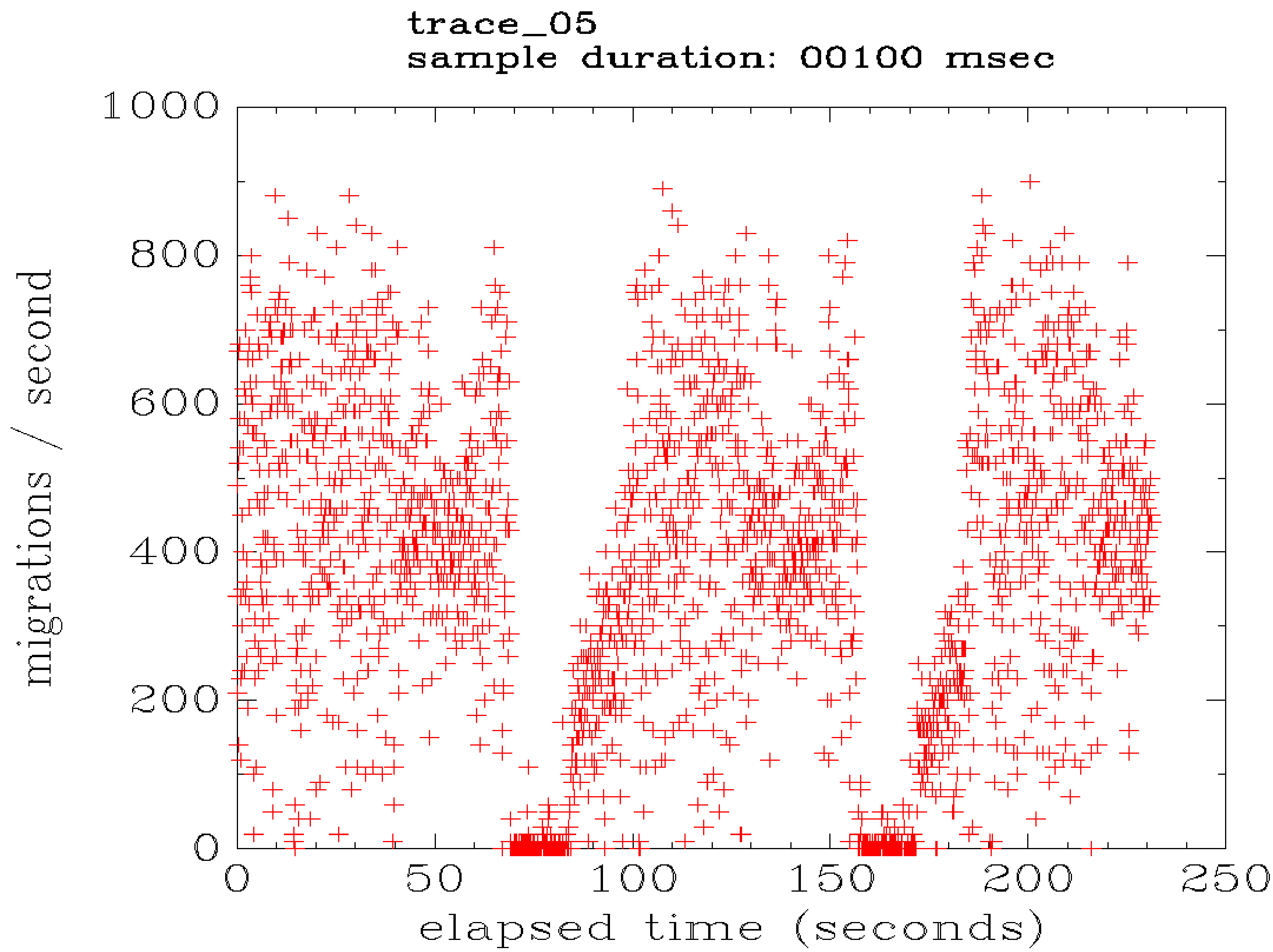

trace_05
sample duration: 10000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
0
10
20
30
40
50
60
70
80

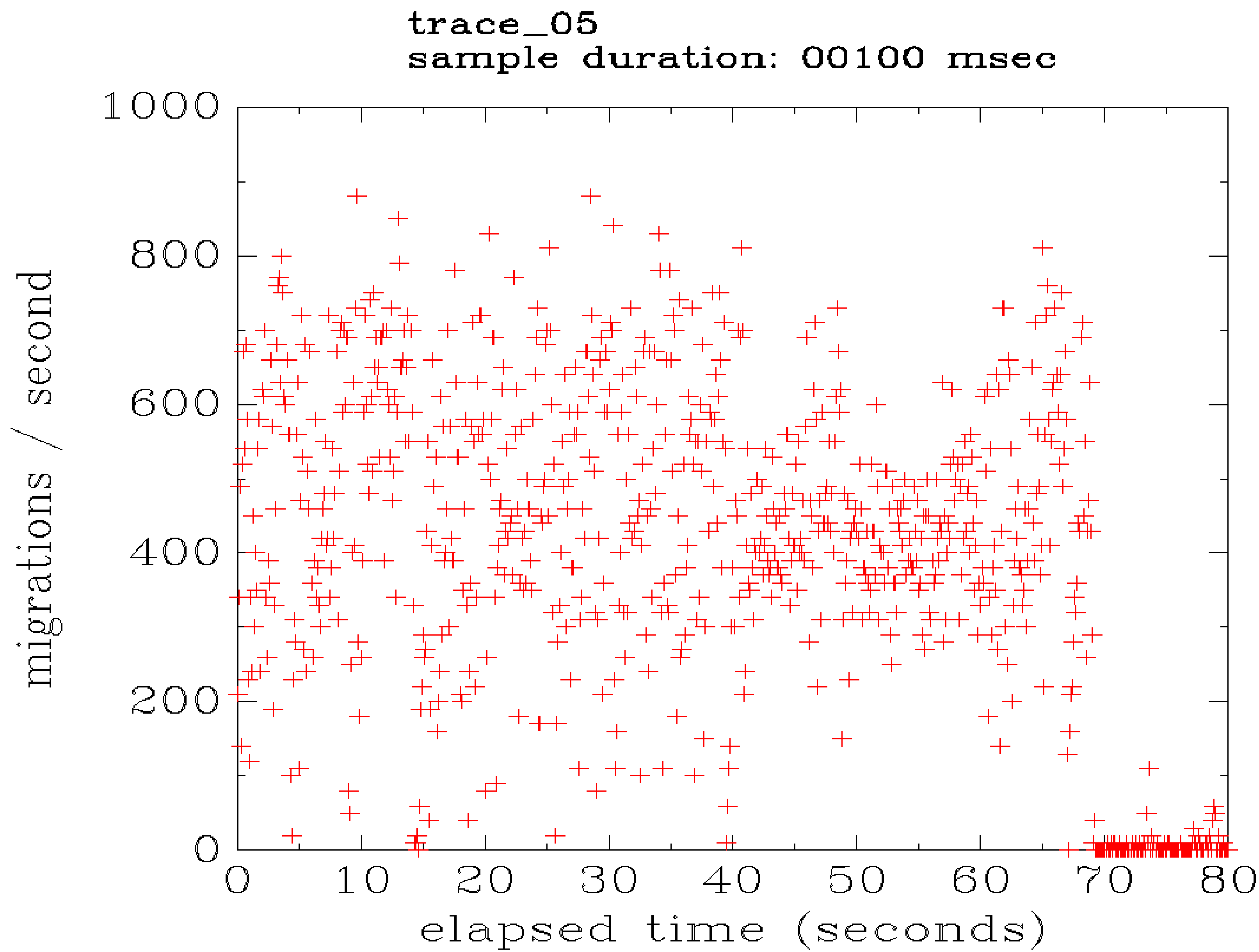

trace_05
sample duration: 20000 msec
500
450
400
350
300
250
200
150
0
10
20
30
40
50
60
70
80
migrations / second
elapsed time (seconds)

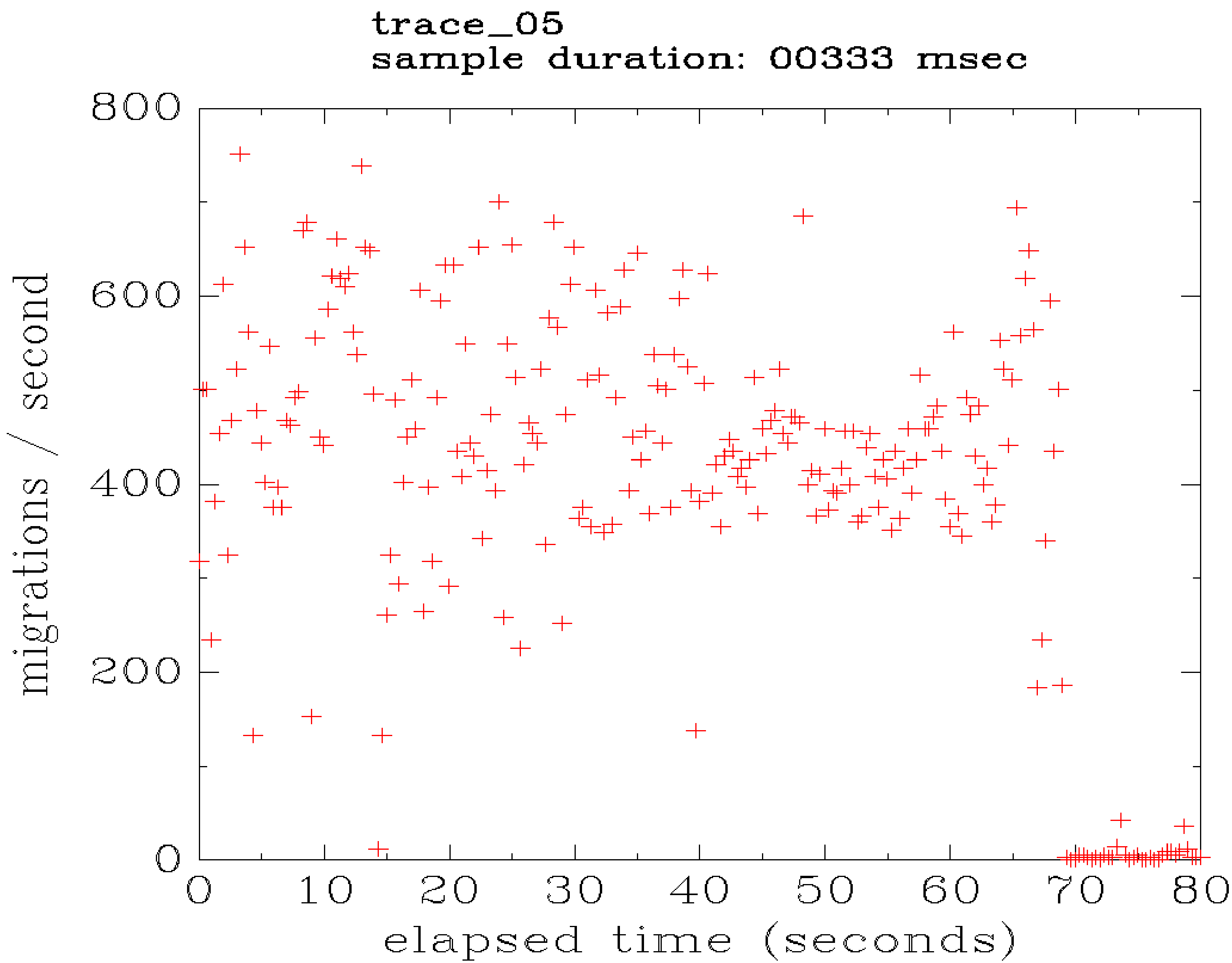

trace_05
sample duration: 40000 msec
480
460
440
420
400
380
360
340
320
0
10
20
30
40
50
60
70
80
migrations / second
elapsed time (seconds)

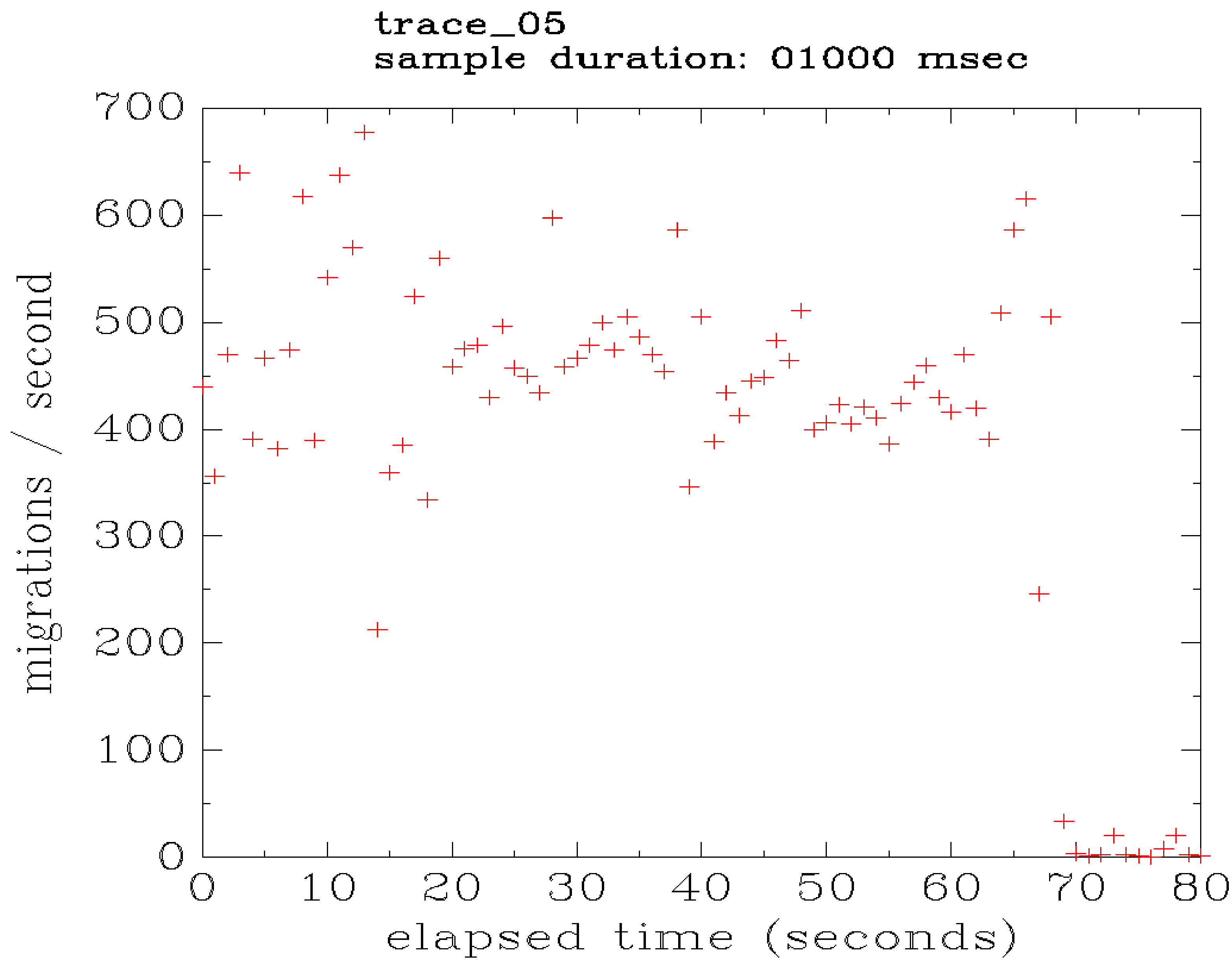
Lines vs Points

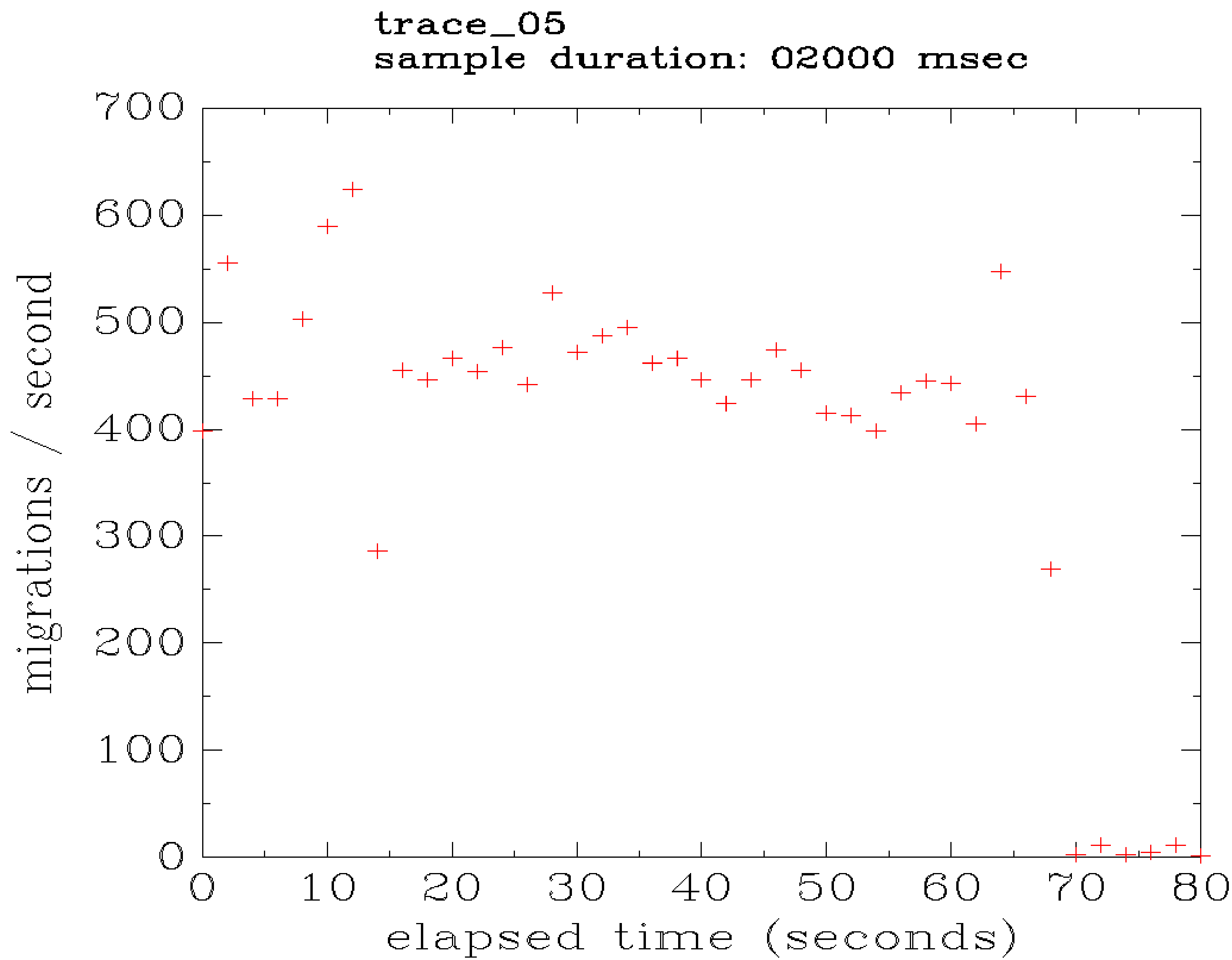
again...


trace_05
sample duration: 00100 msec
migrations / second
elapsed time (seconds)
0
200
400
600
800
1000
0
50
100
150
200
250

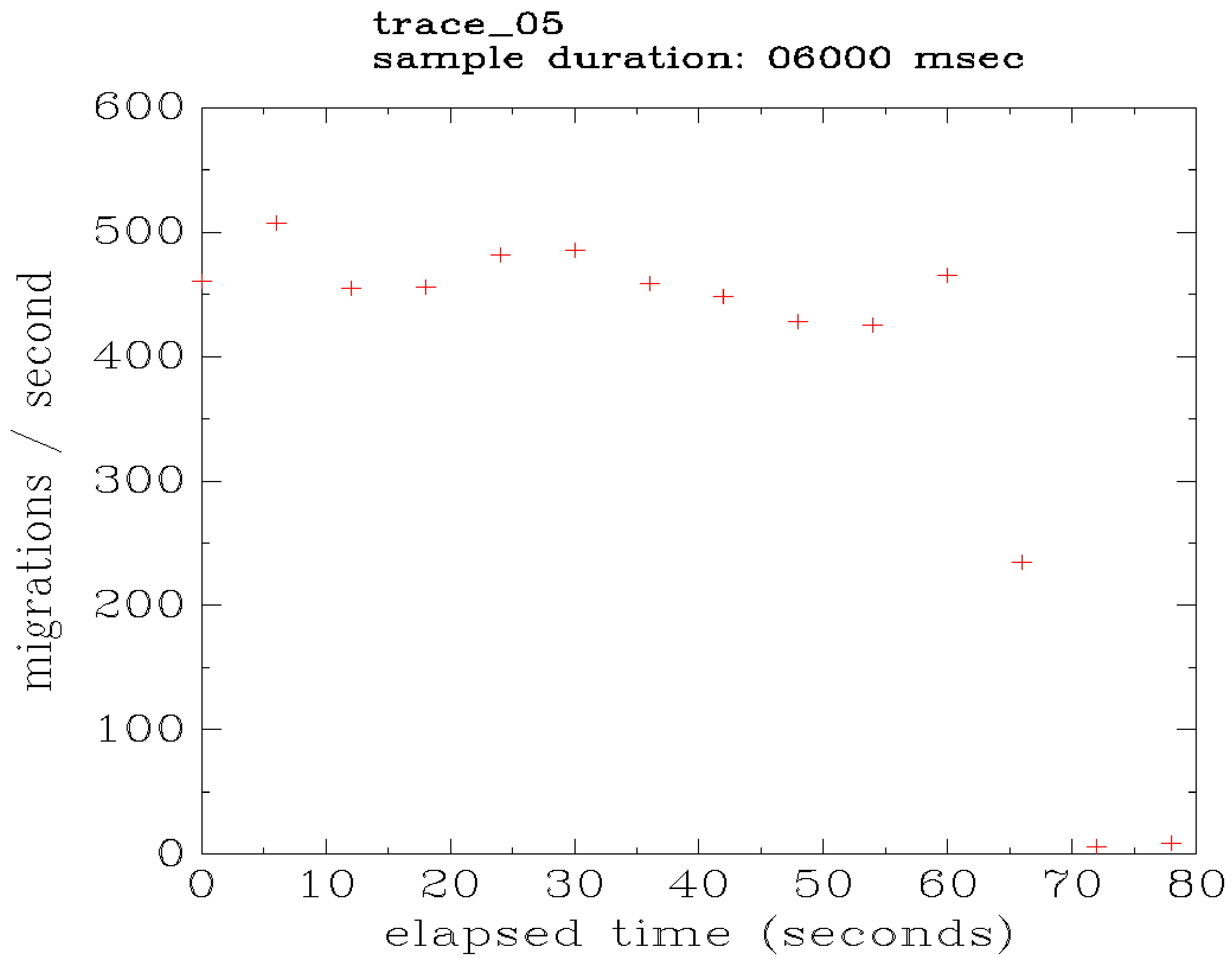

trace_05
sample duration: 00100 msec
migrations / second
1000
800
600
400
200
0
0
10
20
30
40
50
60
70
80
elapsed time (seconds)


trace_05
sample duration: 00333 msec
migrations / second
elapsed time (seconds)
800
600
400
200
0
0
10
20
30
40
50
60
70
80

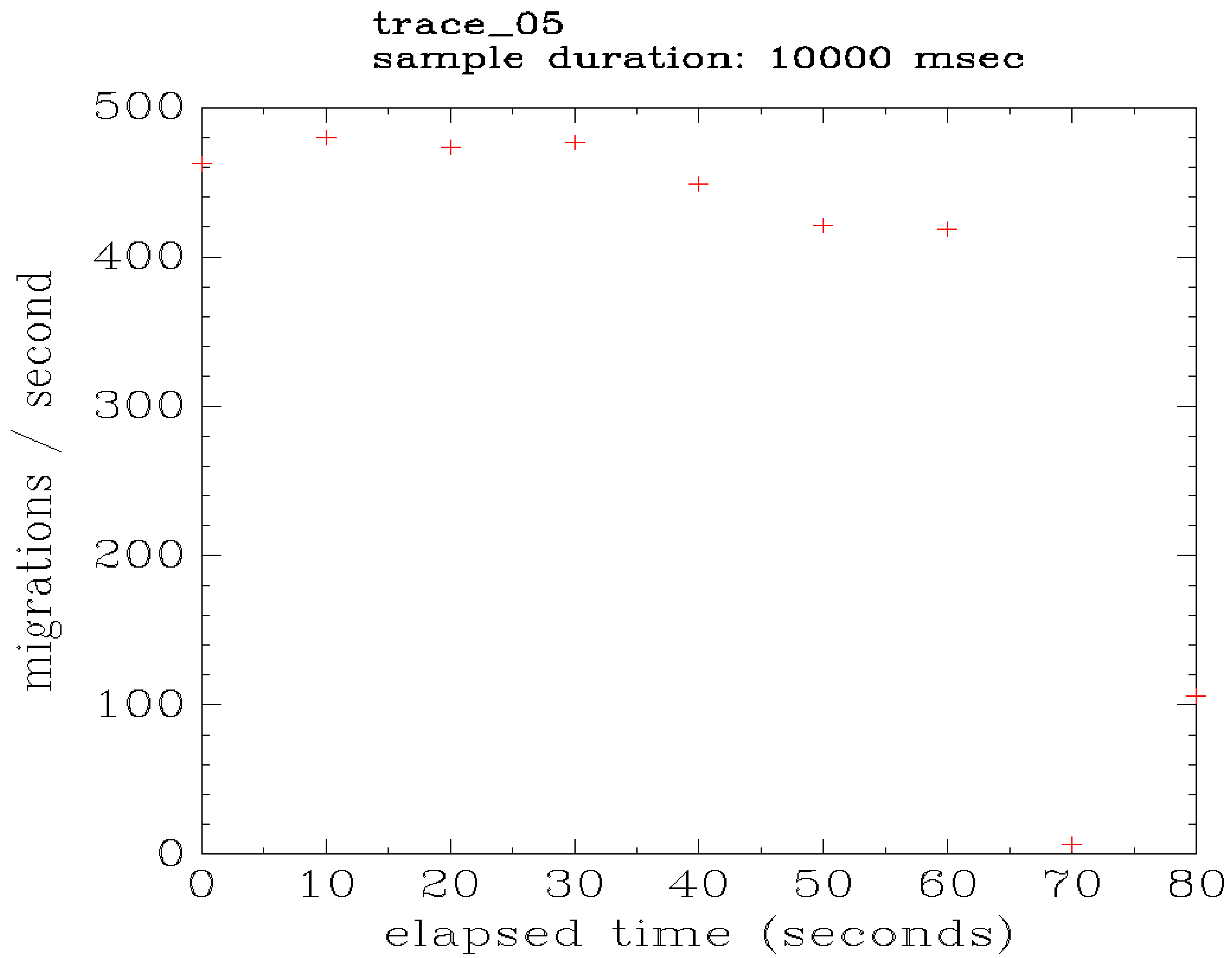

trace_05
sample duration: 01000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
600
700
0
10
20
30
40
50
60
70
80

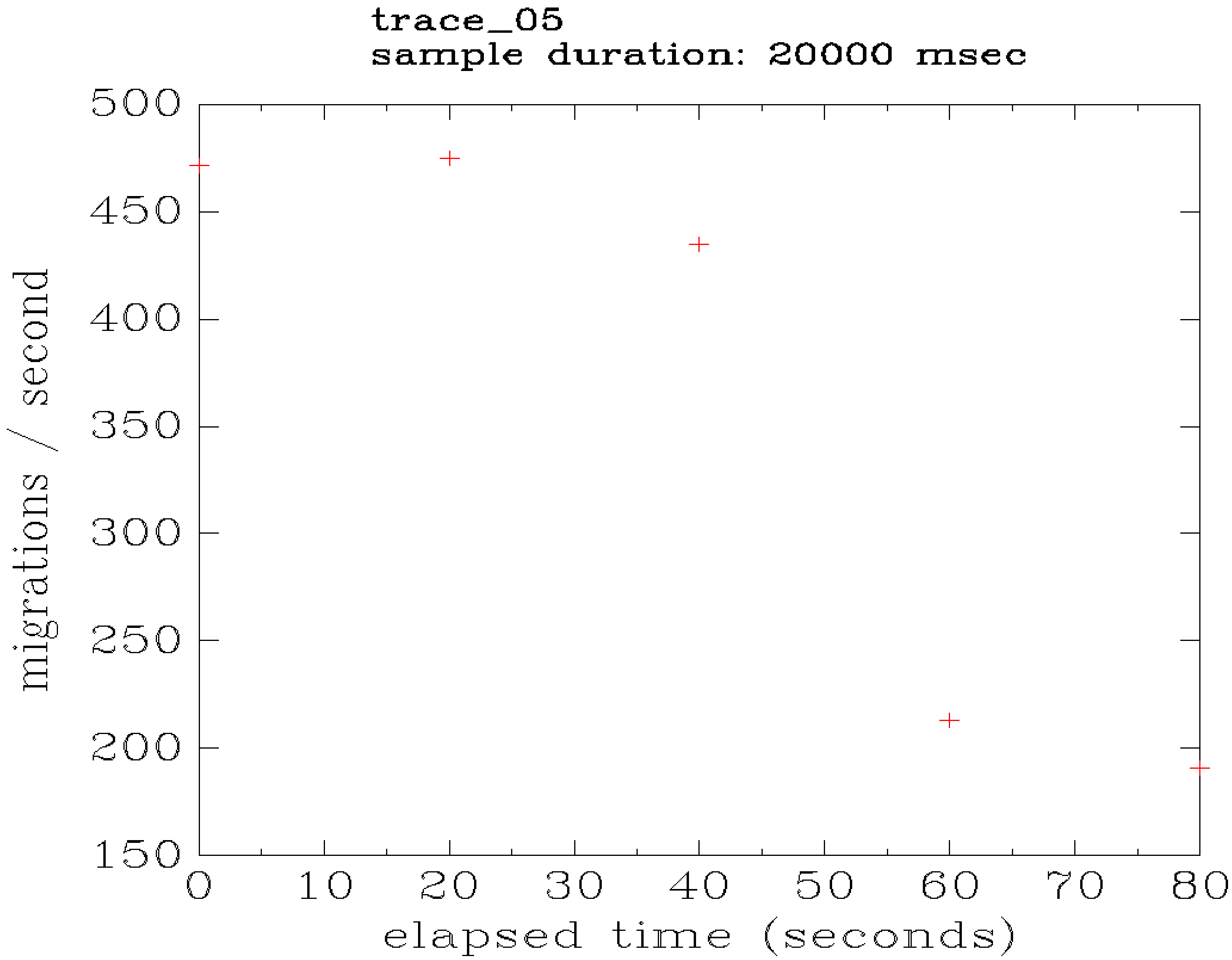


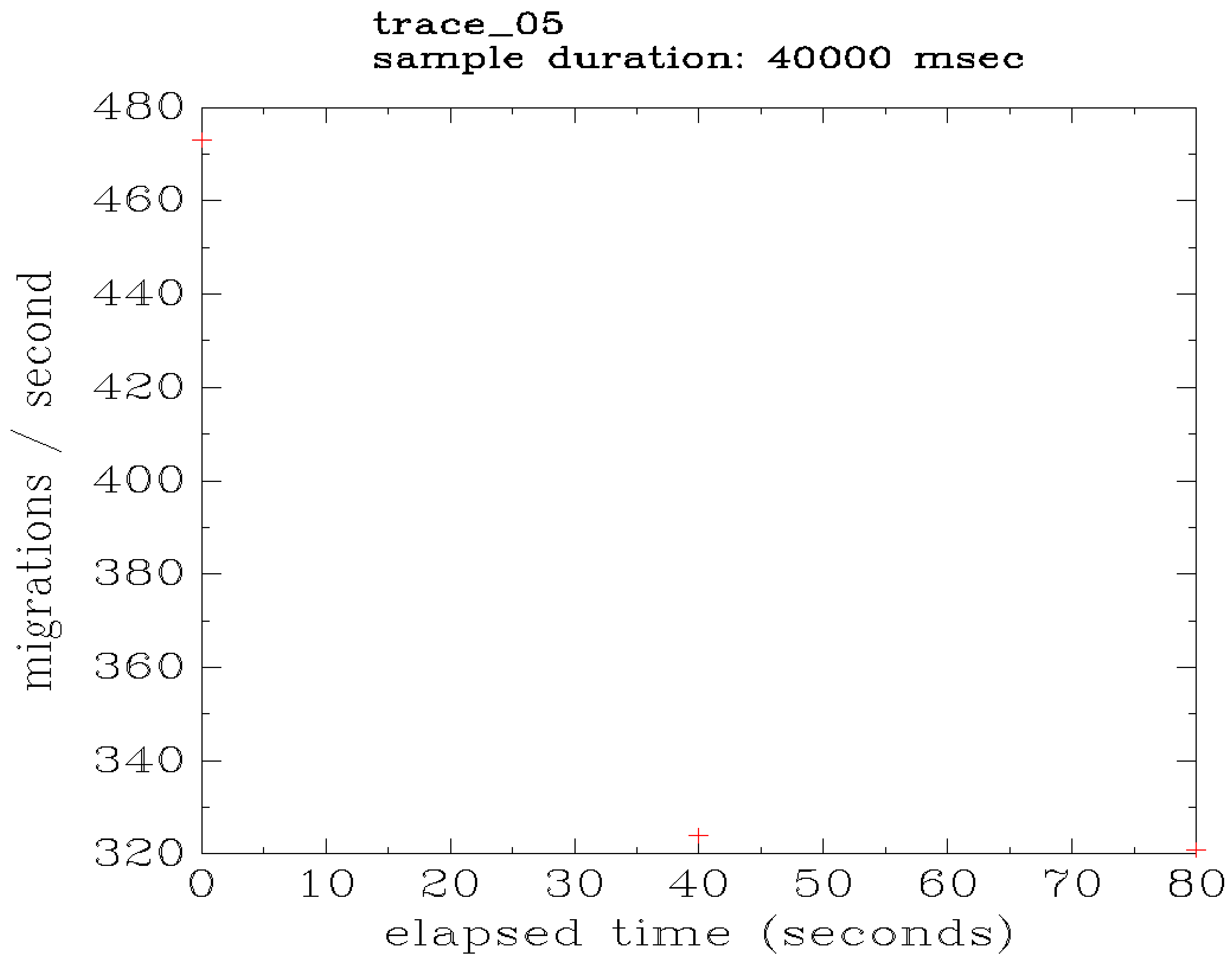
trace_05
sample duration: 02000 msec
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
migrations / second
elapsed time (seconds)



trace_05
sample duration: 06000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
600
0
10
20
30
40
50
60
70
80


trace_05
sample duration: 10000 msec
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
migrations / second
elapsed time (seconds)


trace_05
sample duration: 20000 msec
500
450
400
350
300
250
200
150
0
10
20
30
40
50
60
70
80
migrations / second
elapsed time (seconds)

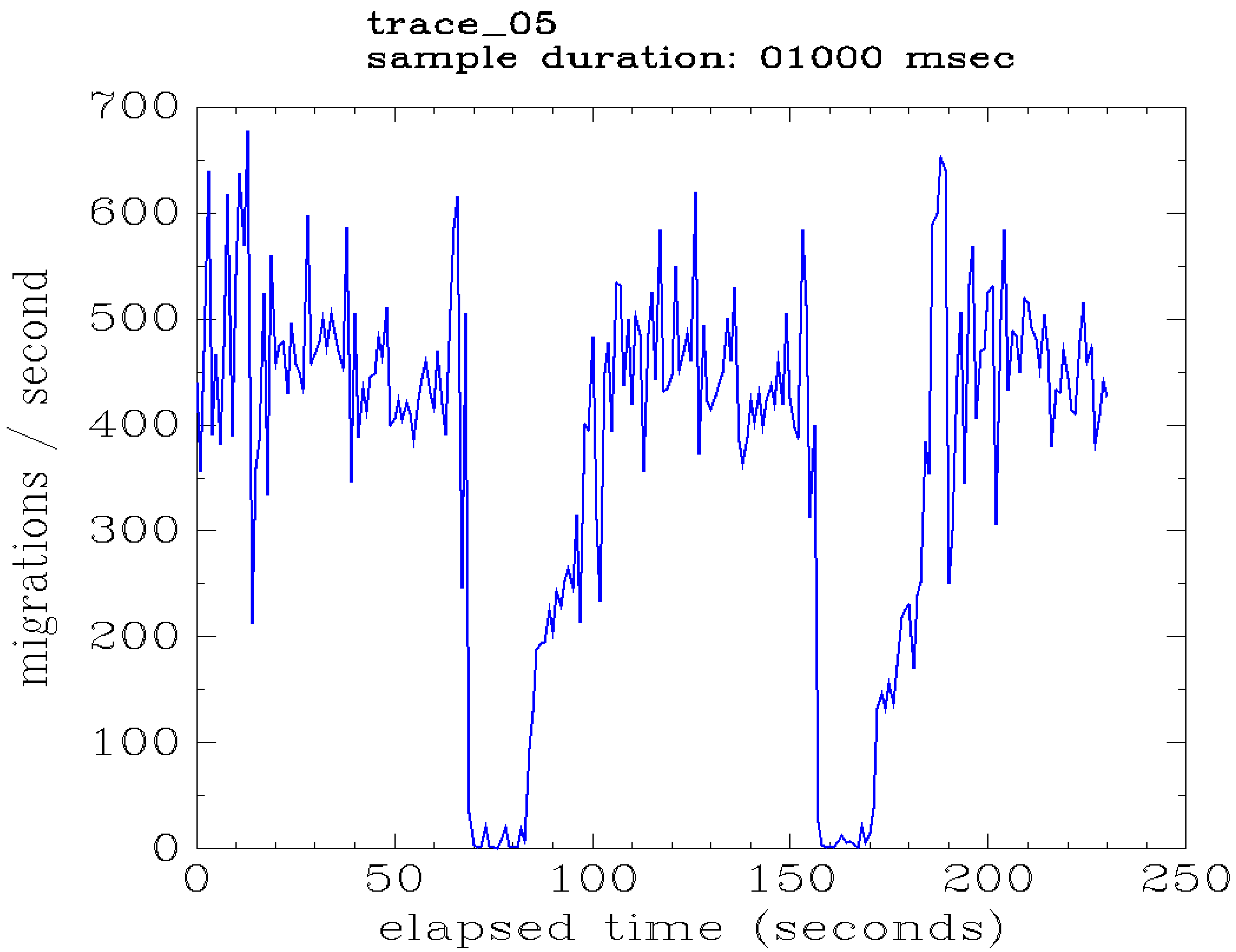

trace_05
sample duration: 40000 msec
migrations / second
elapsed time (seconds)
480
460
440
420
400
380
360
340
320
0
10
20
30
40
50
60
70
80

Transformation between domains

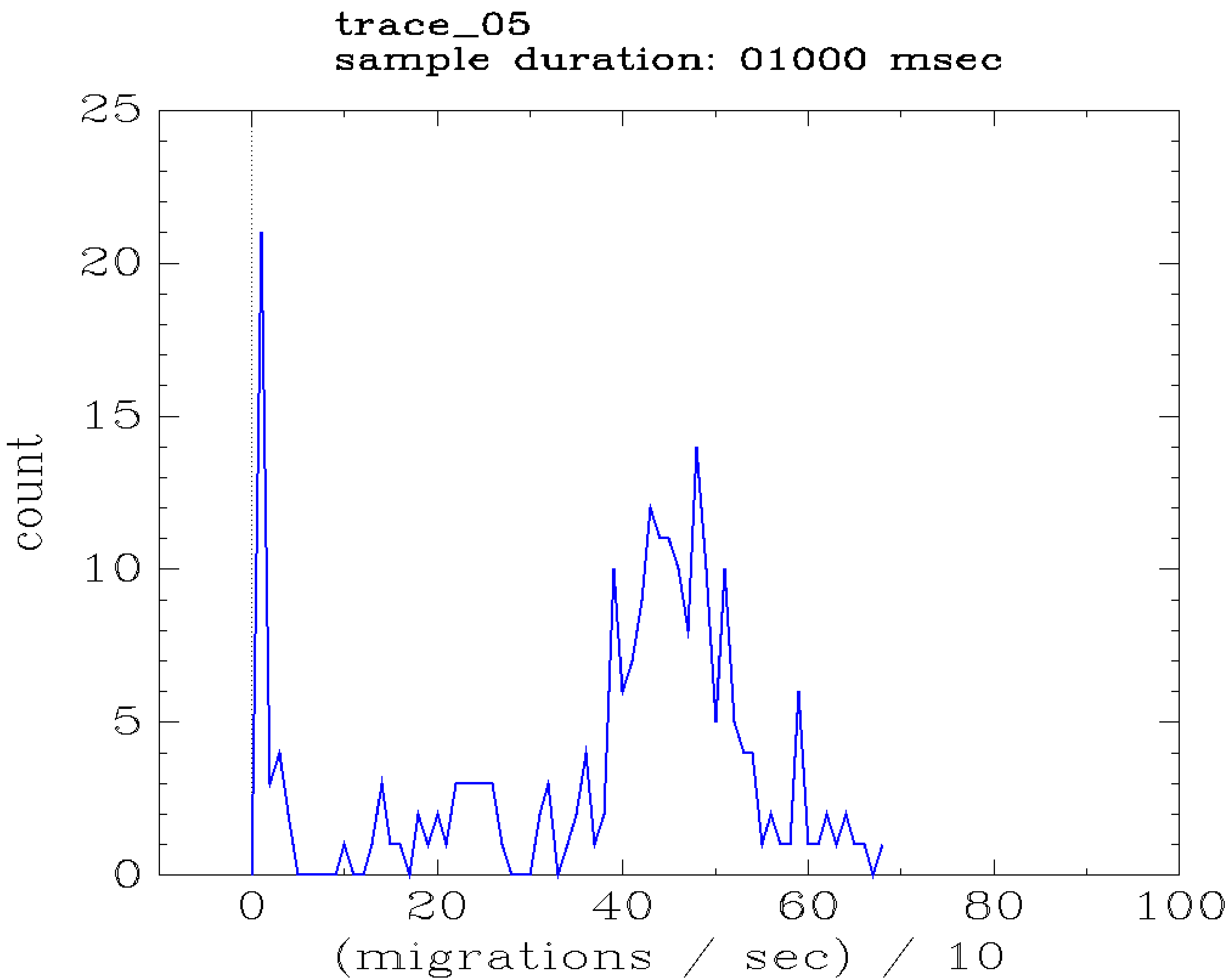
May reveal different information

In this example:

- time domain to histogram transformation shows significance of bi-modal nature of data more clearly....



trace_05
sample duration: 01000 msec
migrations / second
elapsed time (seconds)
0
100
200
300
400
500
600
700
0
50
100
150
200
250


trace_05
sample duration: 01000 msec
count
(migrations / sec) / 10
0
5
10
15
20
25
0
20
40
60
80
100

Scatterplot

Graph two metrics for each sample

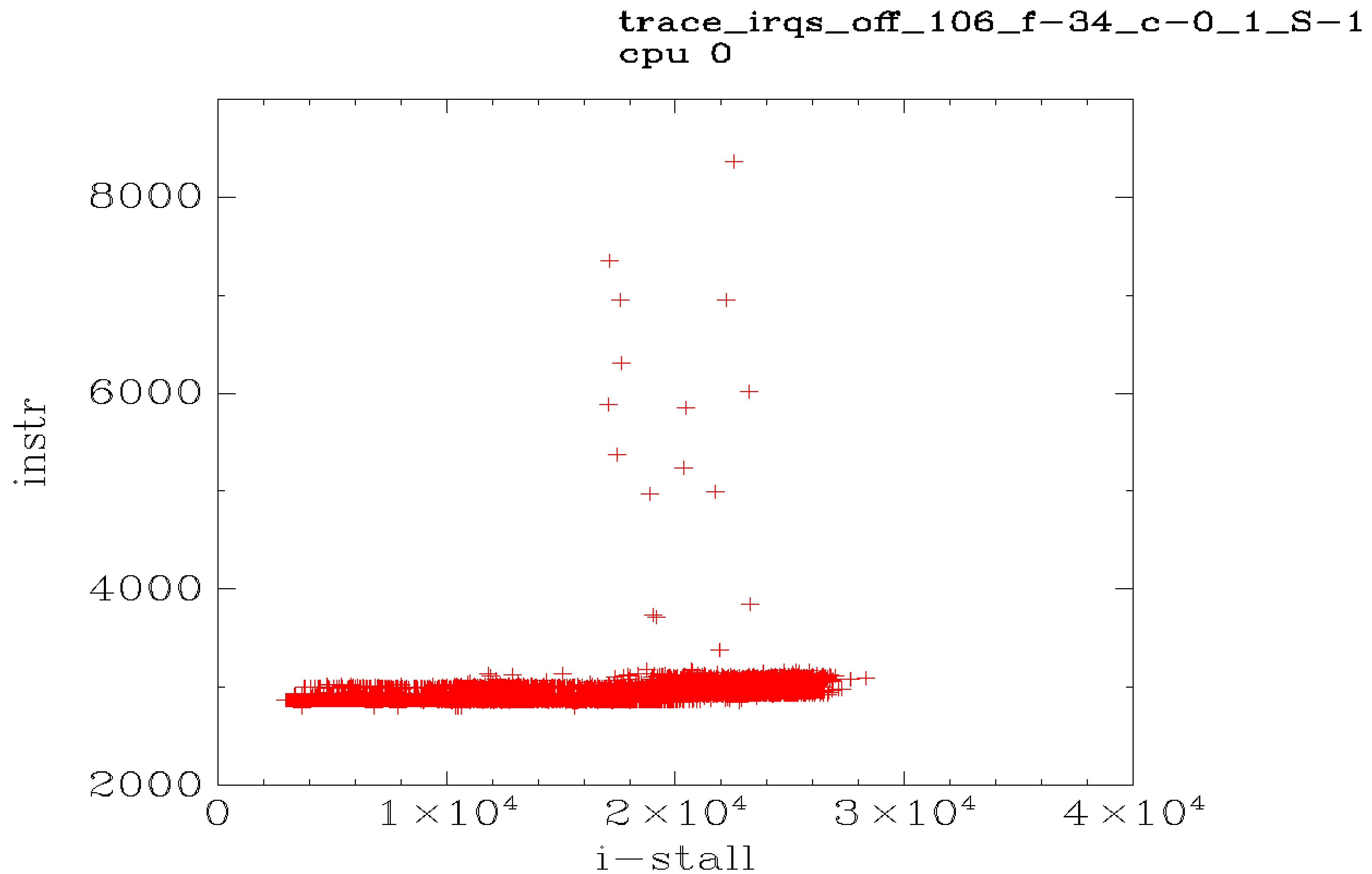
Requires logging of each sample instead of updating counter(s) for each sample.

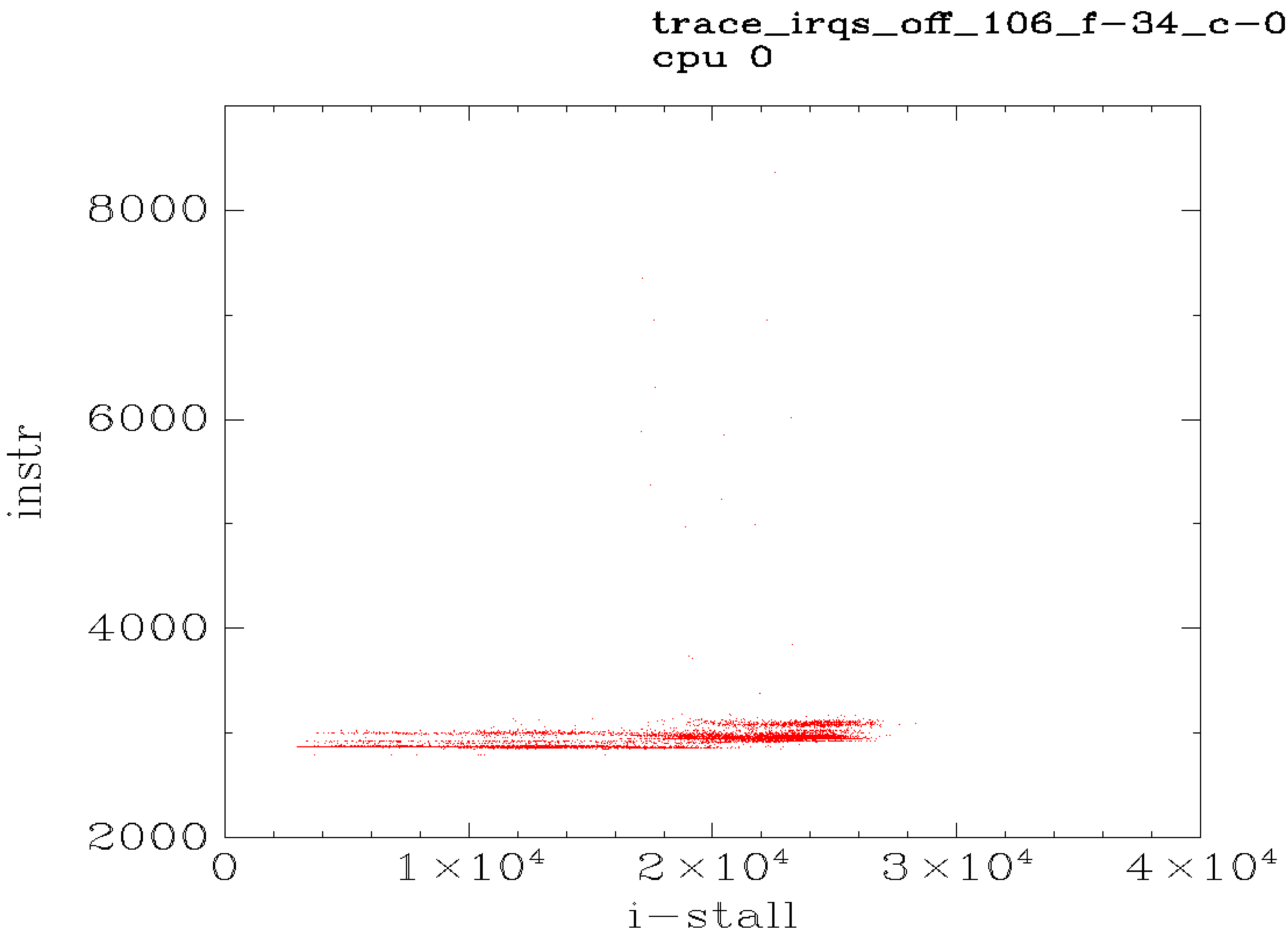
- More detailed information
- More overhead in data collection

Rescale

Individual samples: visibility vs details

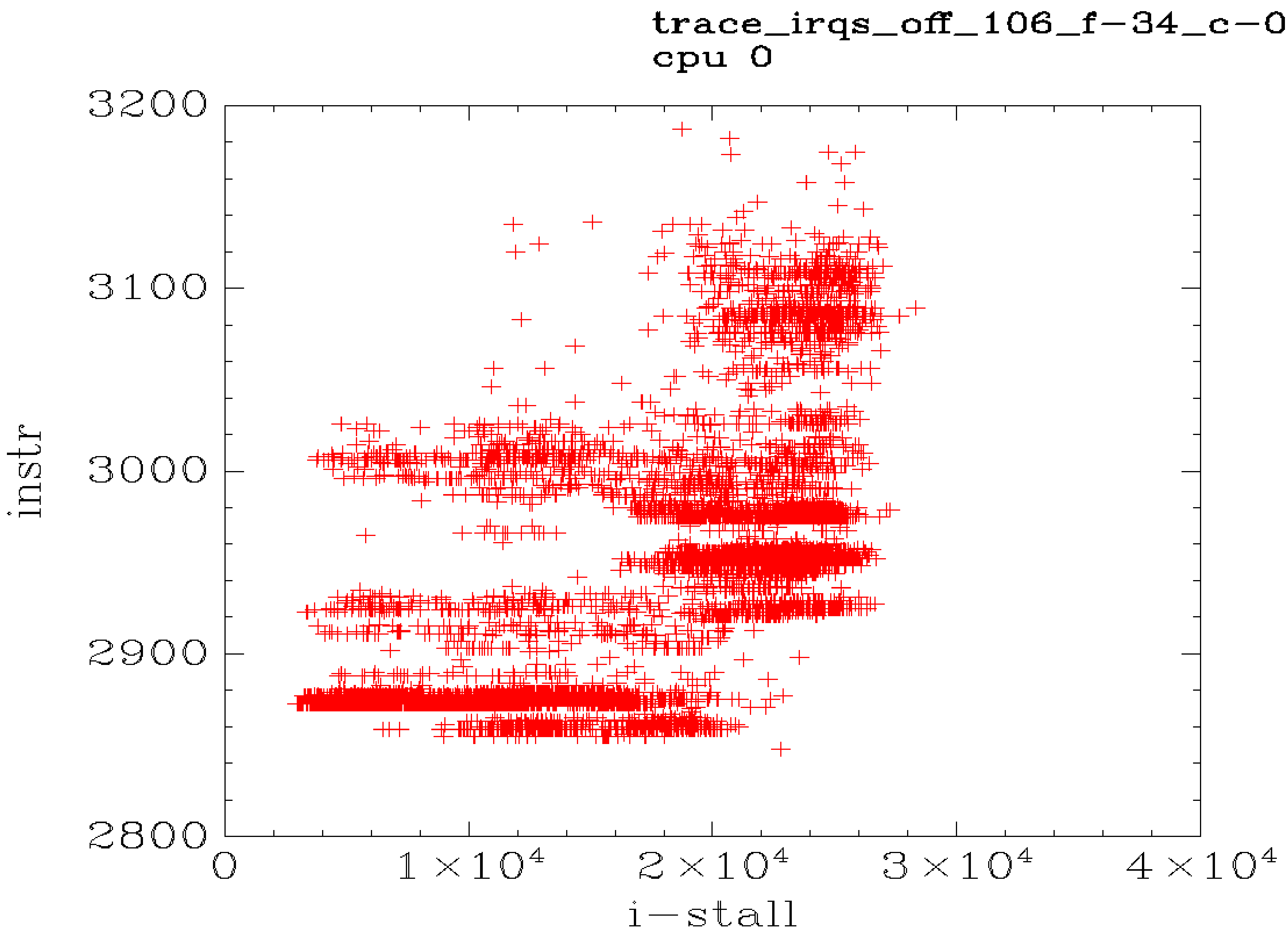
data: cost of do_local_timer()

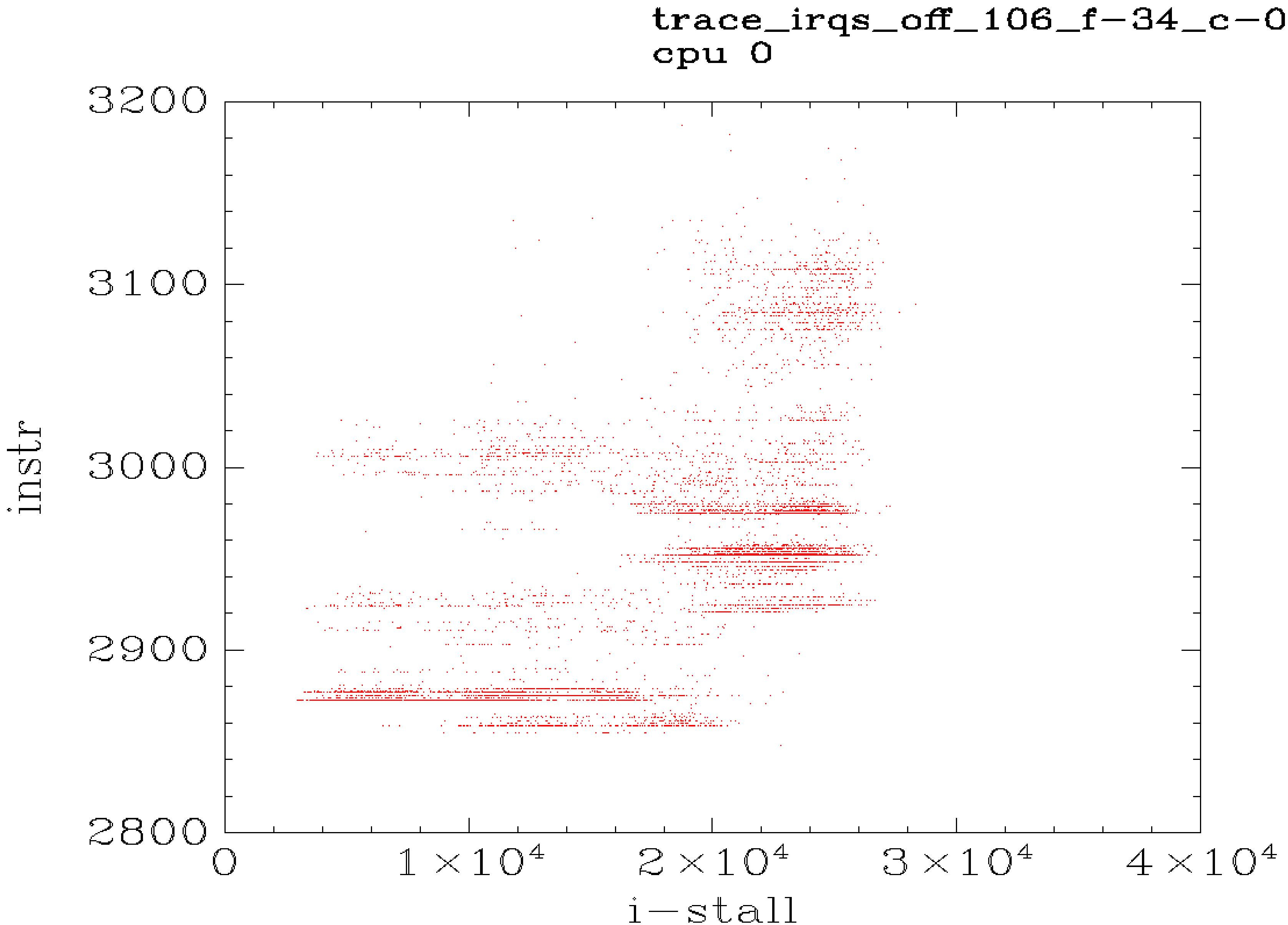

trace_irqs_off_106_f-34_c-0_1_S-1
cpu 0
instr
8000
6000
4000
2000
0
1×10⁴
2×10⁴
3×10⁴
4×10⁴
i-stall


trace_irqs_off_106_f-34_c-0_1_S-0
cpu 0
instr
8000
6000
4000
2000
0
1×10⁴
2×10⁴
3×10⁴
4×10⁴
i-stall

Rescale

Change scale to expose detail (2)


trace_irqs_off_106_f-34_c-0_1_S-1
cpu 0
3200
3100
3000
2900
2800
instr
0
1×10⁴
2×10⁴
3×10⁴
4×10⁴
i-stall

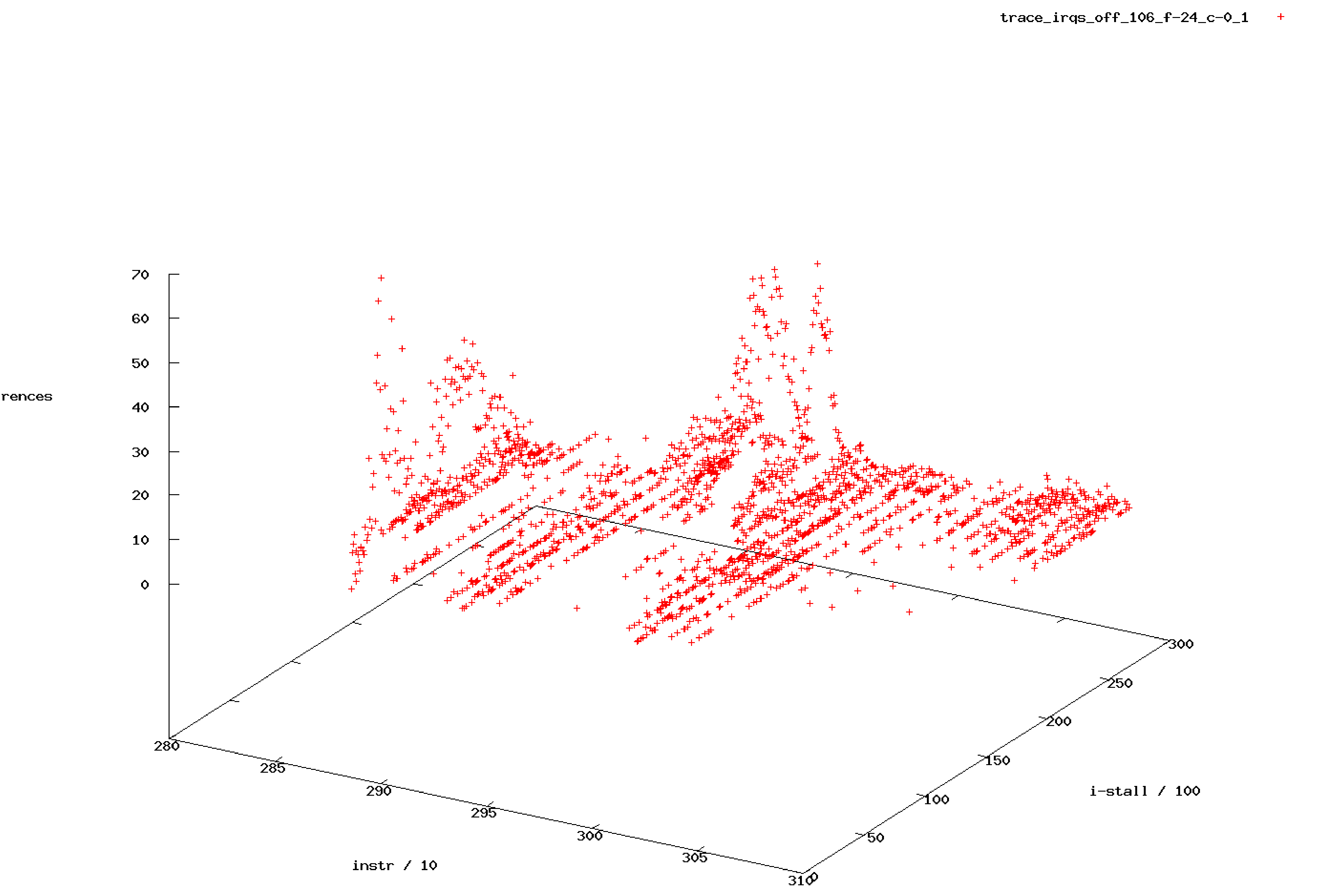


trace_irqs_off_106_f-34_c-0_1_S-0
cpu 0
instr
3200
3100
3000
2900
2800
0
1 × 10⁴
2 × 10⁴
3 × 10⁴
4 × 10⁴
i-stall

Additional Dimensions

Reveal additional information (intensity)

- 3-d (shown)
- 2-d with color gradient (not shown)


trace_irqs_off_106_f-24_c-0_1
rences
70
60
50
40
30
20
10
0
280
285
290
295
300
305
310
instr / 10
0
50
100
150
200
250
300
i-stall / 100

Finding trees in the forest

(1) Subset the data:

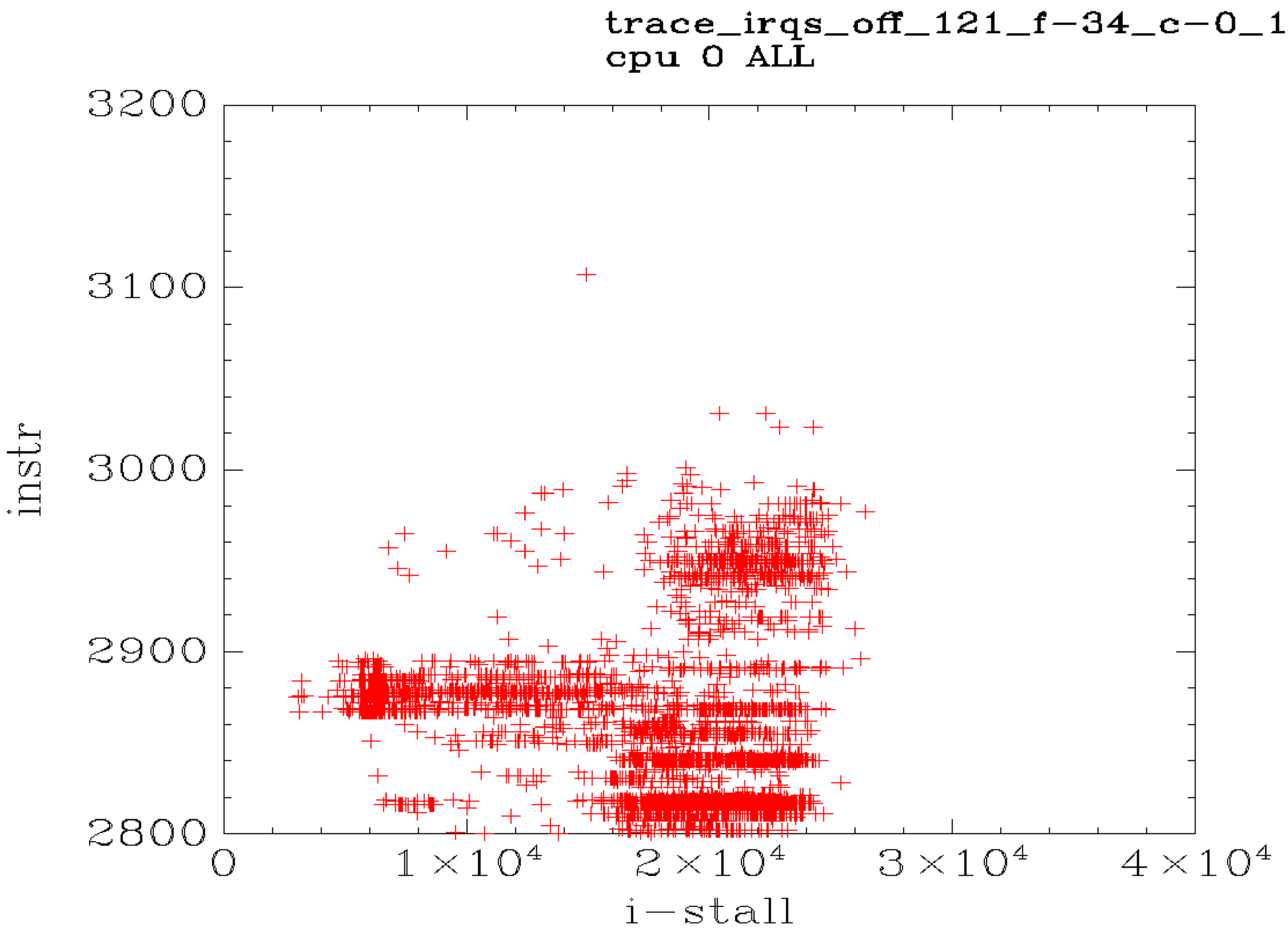
- all samples
- samples filtered by scheduling policy of interrupted task
- segment multi-modal distribution into several (nearly) normal distributions

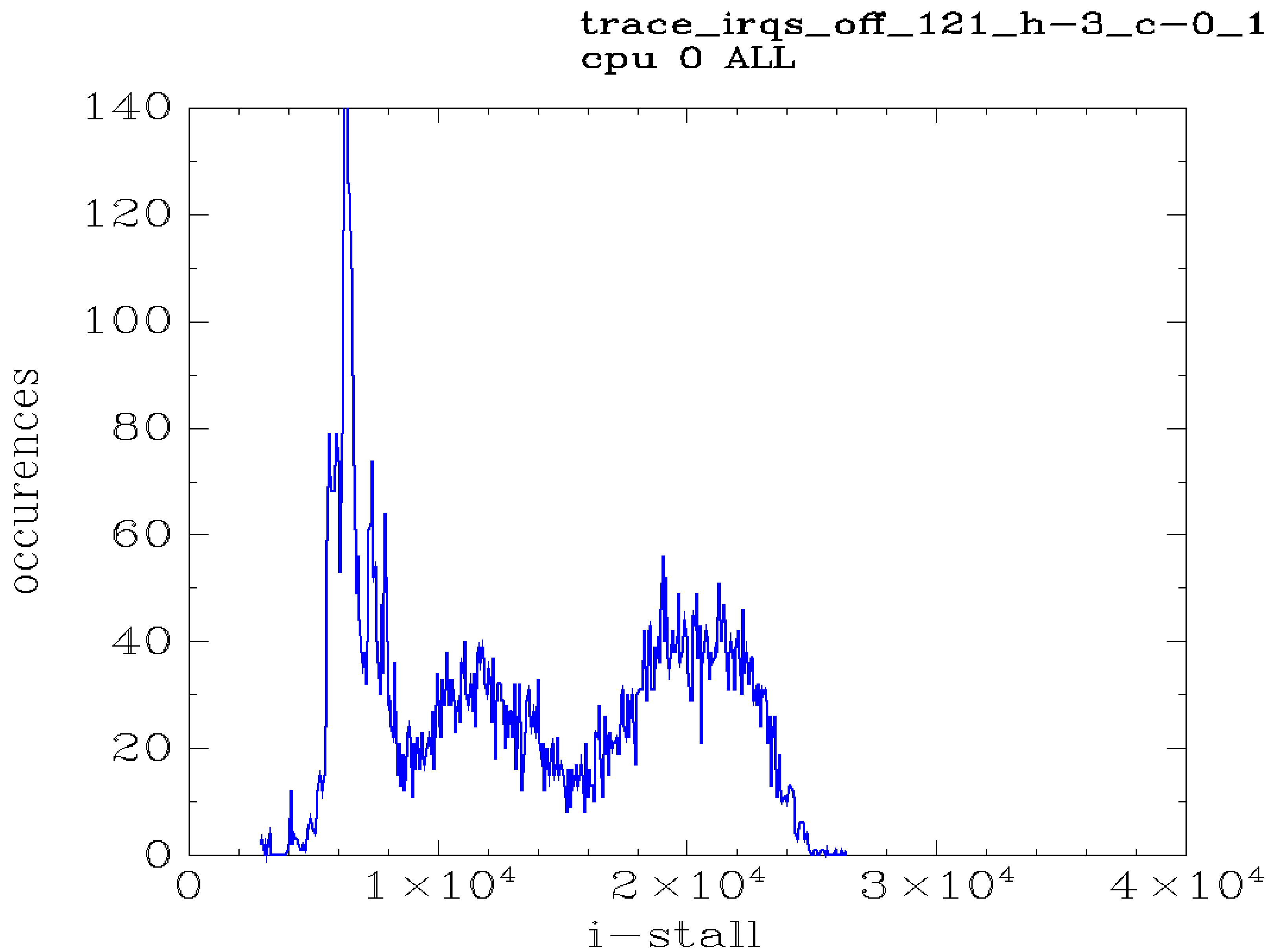
Finding trees in the forest

(2) Transformation between domains
and another way of showing intensity

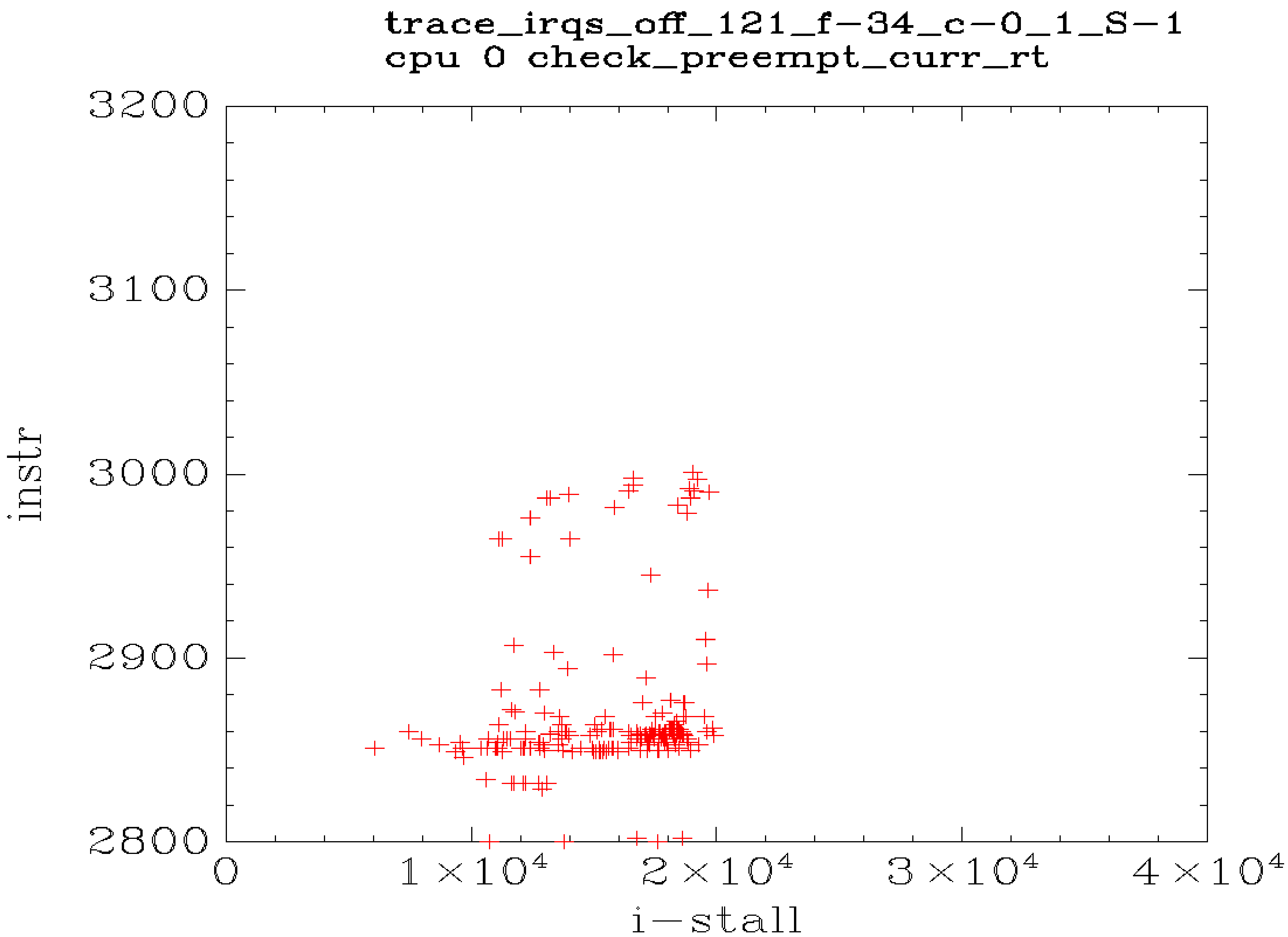
Puts focus on two different concepts:

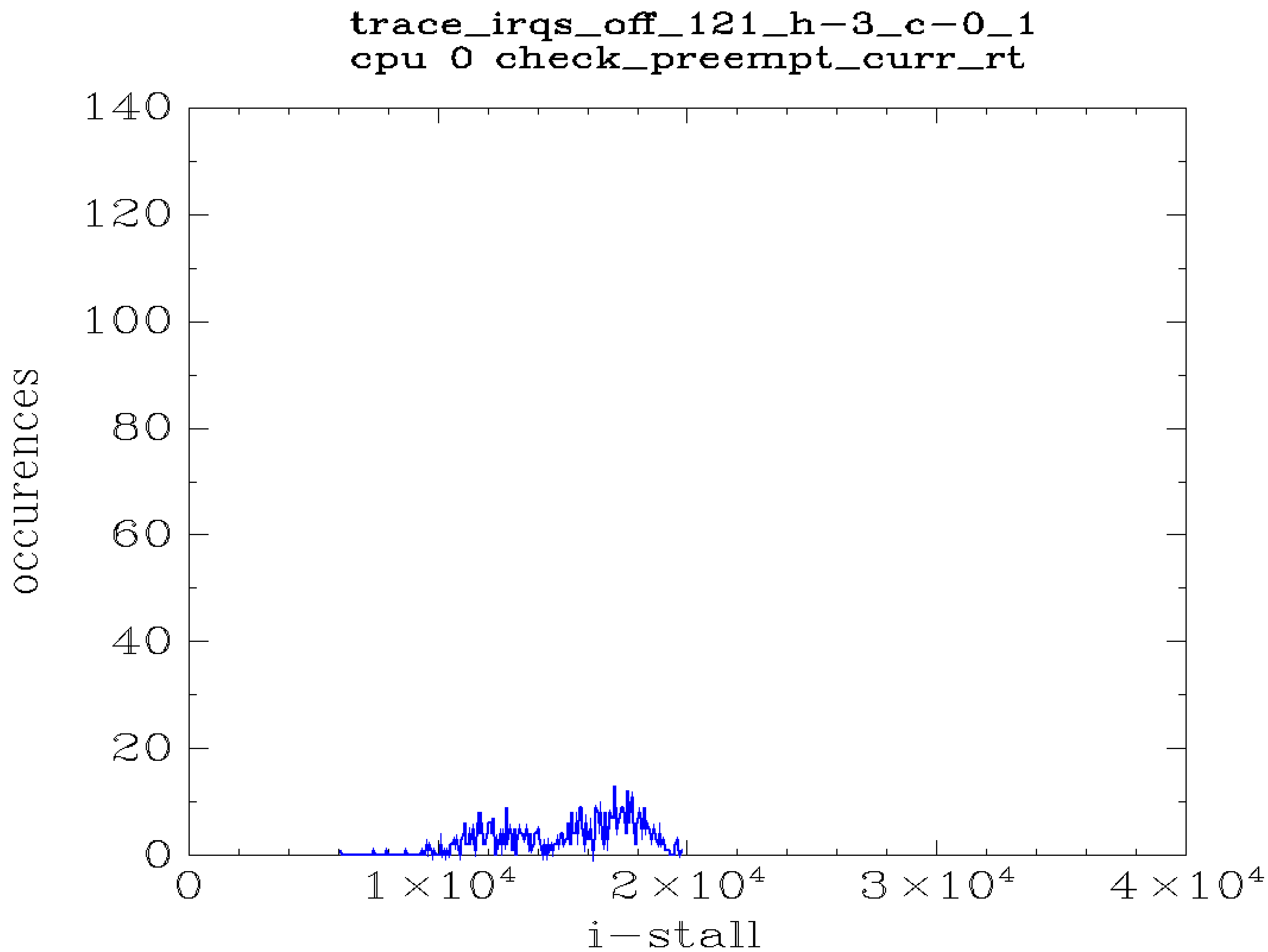
- # instructions vs # i-cache stall cycles
- frequency of # i-cache stall cycles

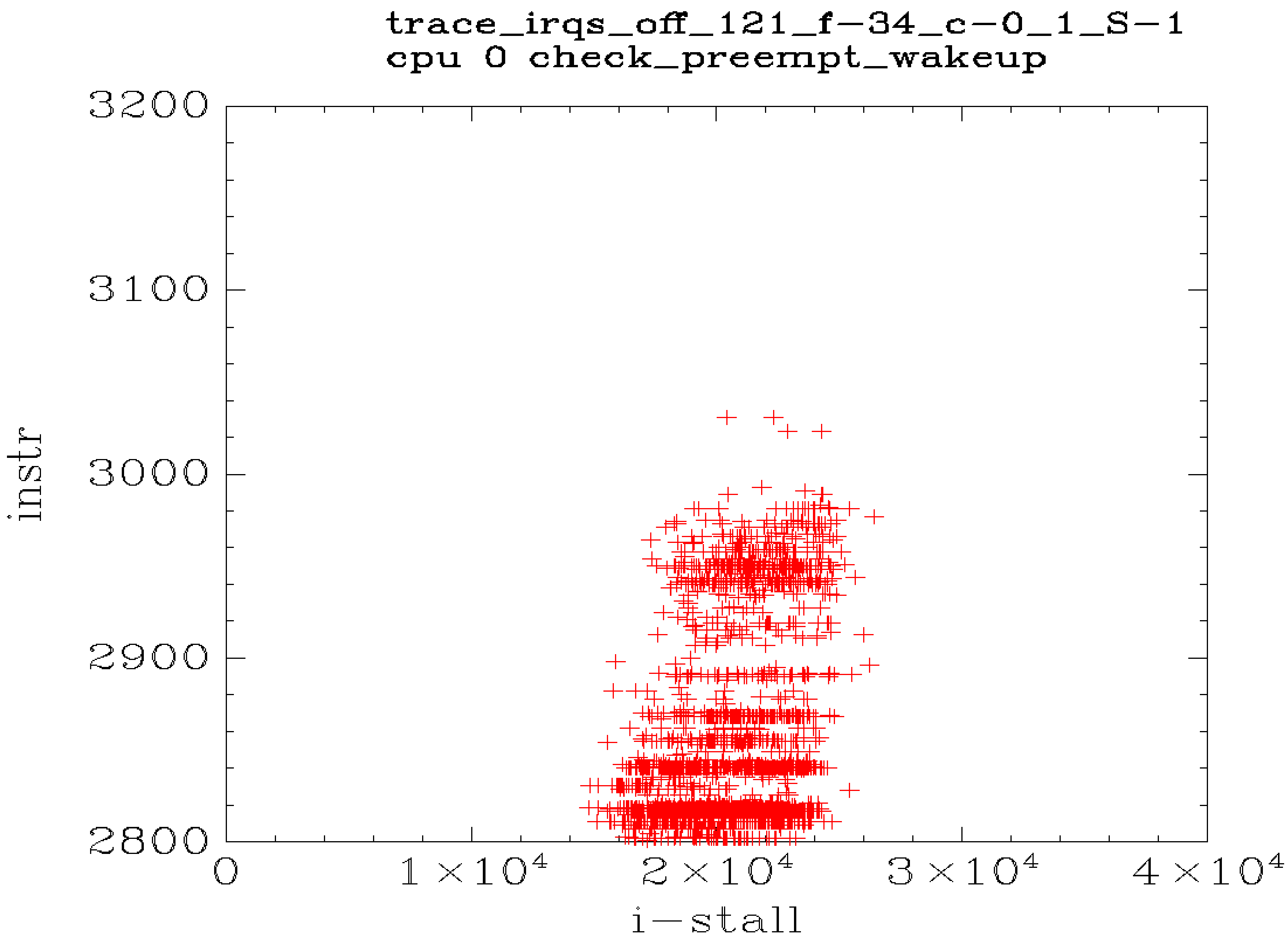

trace_irqs_off_121_f-34_c-0_1_S-1
cpu 0 ALL
3200
3100
3000
2900
2800
instr
0
1×10^4
2×10^4
3×10^4
4×10^4
i-stall

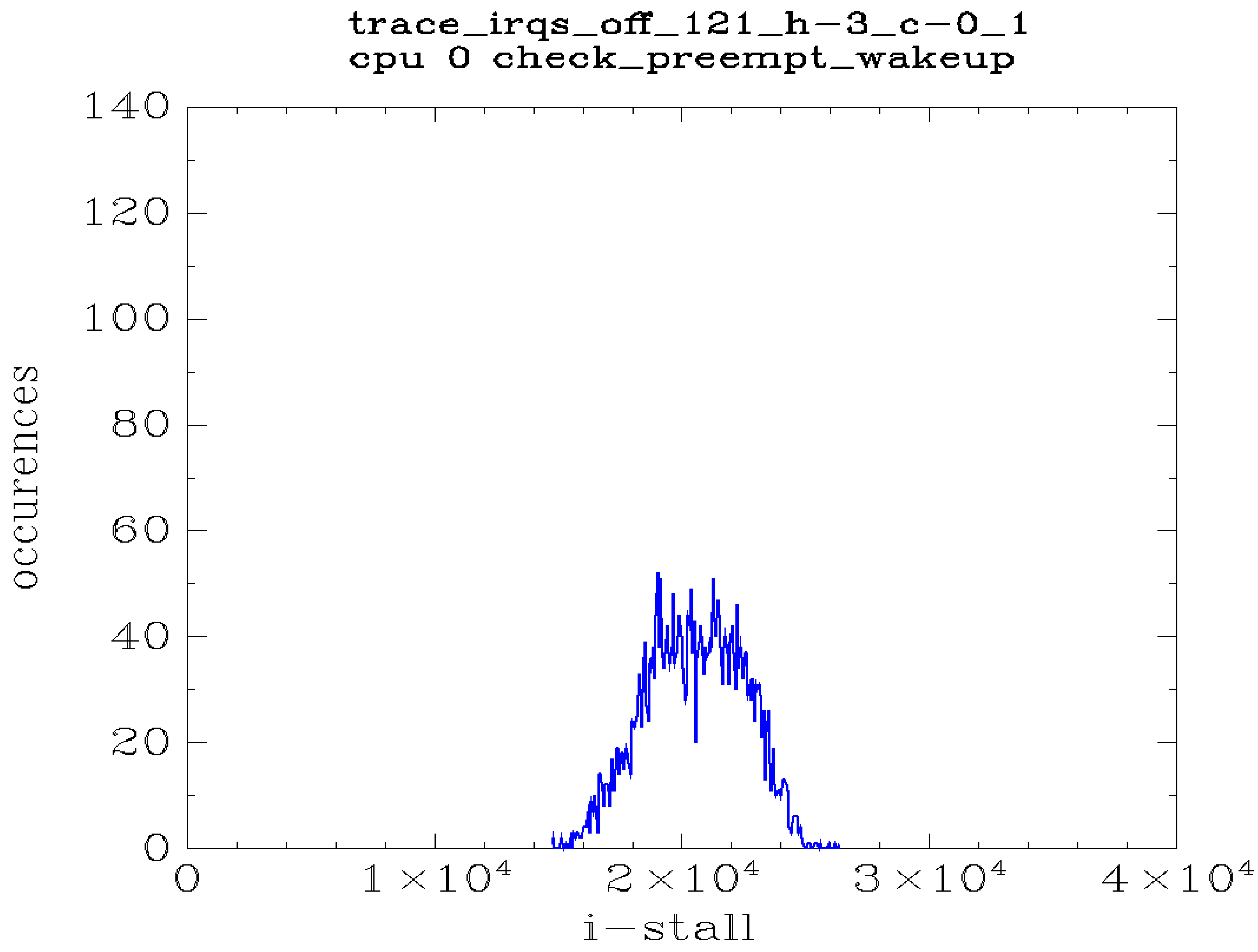


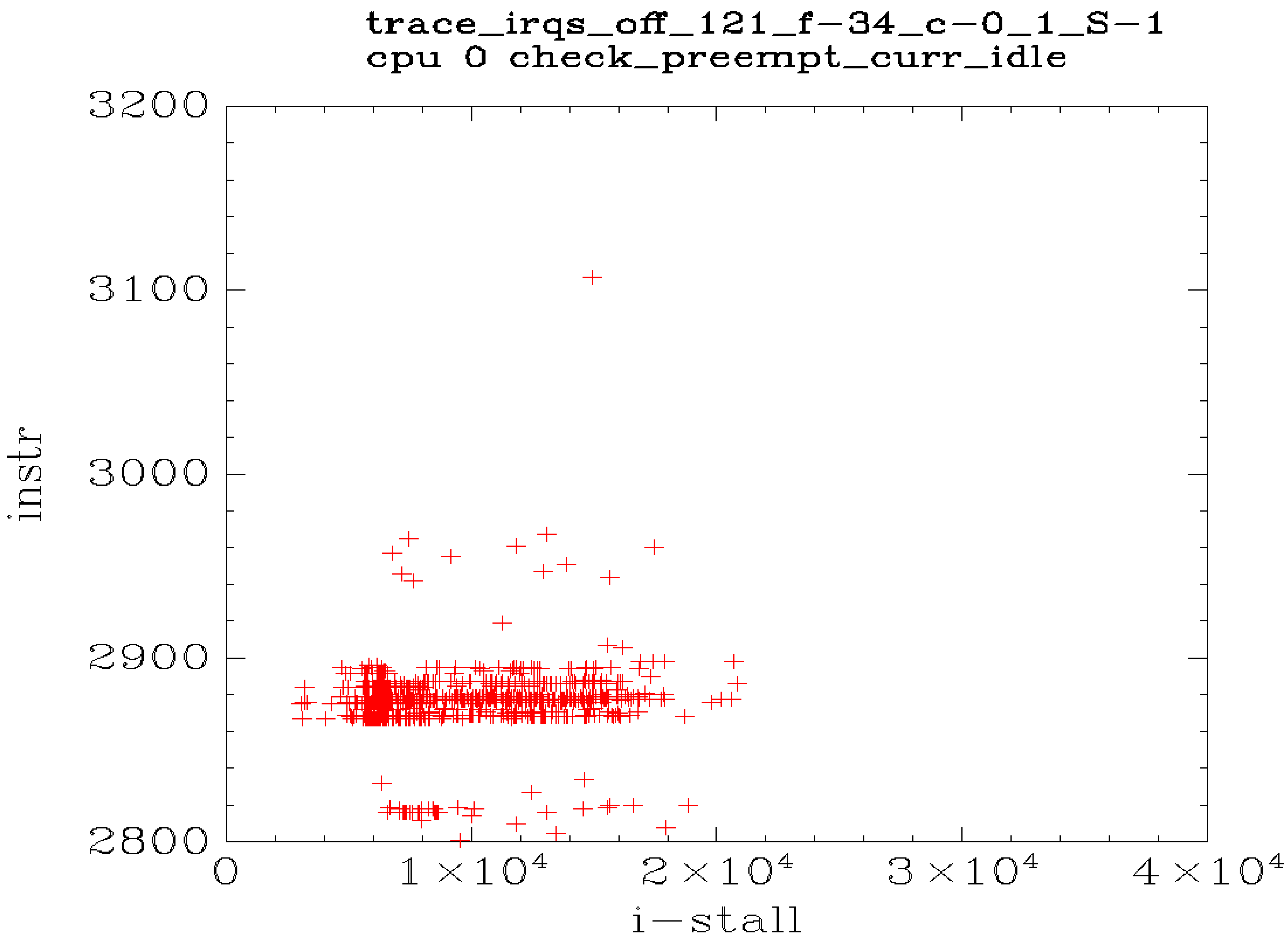
trace_irqs_off_121_h-3_c-0_1
cpu 0 ALL
140
120
100
80
60
40
20
0
occurences
0
1×10^4
2×10^4
3×10^4
4×10^4
i-stall

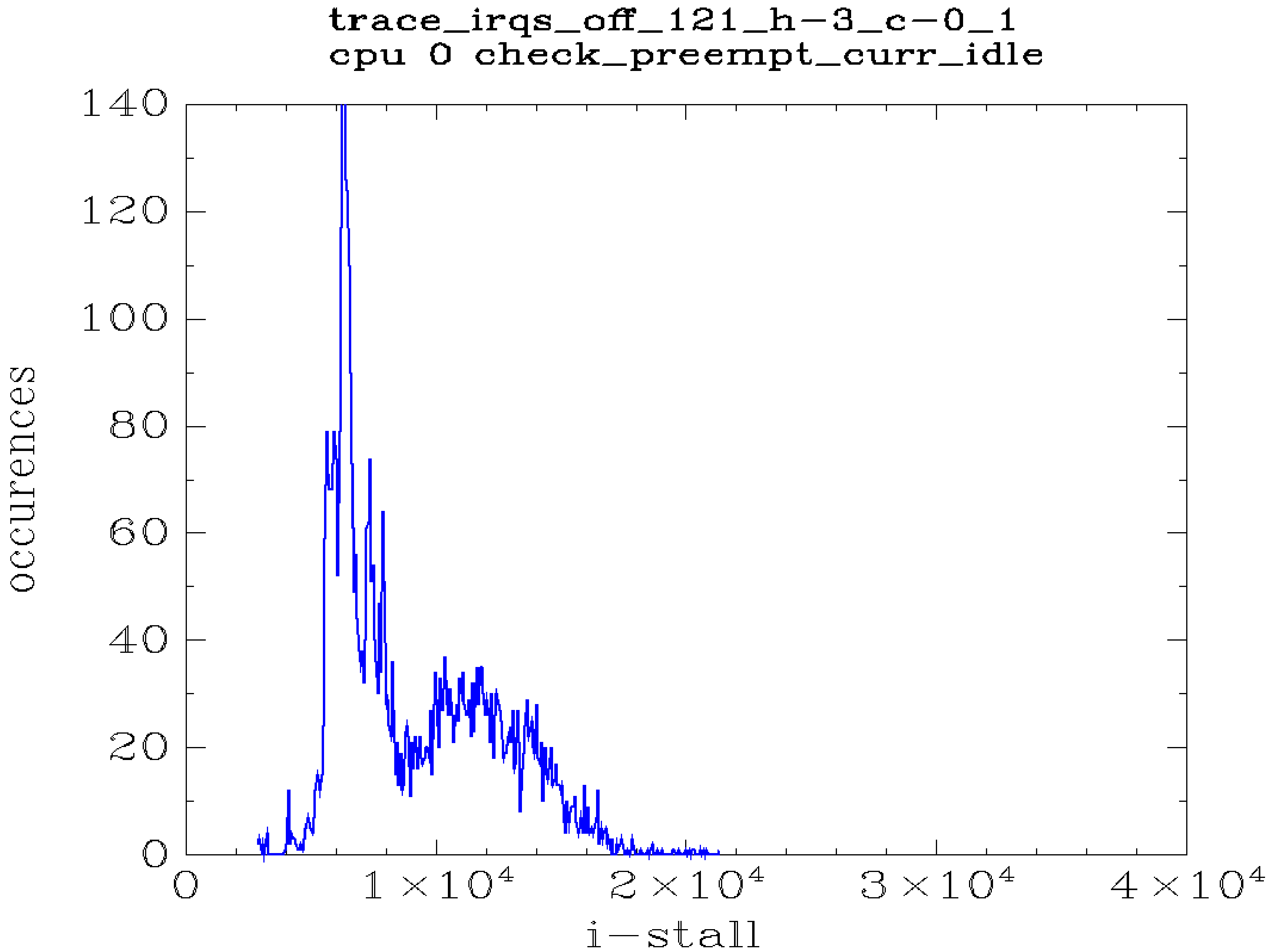

trace_irqs_off_121_f-34_c-0_1_S-1
cpu 0 check_preempt_curr_rt
3200
3100
3000
2900
2800
instr
0
1×10⁴
2×10⁴
3×10⁴
4×10⁴
i-stall


trace_irqs_off_121_h-3_c-0_1
cpu 0 check_preempt_curr_rt
occurences
i-stall
0
20
40
60
80
100
120
140
0
1×10^4
2×10^4
3×10^4
4×10^4


trace_irqs_off_121_f-34_c-0_1_S-1
cpu 0 check_preempt_wakeup
instr
3200
3100
3000
2900
2800
0
1×10^4
2×10^4
3×10^4
4×10^4
i-stall


trace_irqs_off_121_h-3_c-0_1
cpu 0 check_preempt_wakeup
occurences
i-stall
0
20
40
60
80
100
120
140
0
1×10⁴
2×10⁴
3×10⁴
4×10⁴


trace_irqs_off_121_f-34_c-0_1_S-1
cpu 0 check_preempt_curr_idle
3200
3100
3000
2900
2800
instr
0
1×10⁴
2×10⁴
3×10⁴
4×10⁴
i-stall



trace_irqs_off_121_h-3_c-0_1
cpu 0 check_preempt_curr_idle
occurences
i-stall
0
20
40
60
80
100
120
140
1×10⁴
2×10⁴
3×10⁴
4×10⁴

Comparing multiple tests

Metric: maximum IRQ disabled time

- 1 types of test for each kernel configuration

- 4 variants of kernel configuration
  - data for cpu 0
  - data for cpu 1

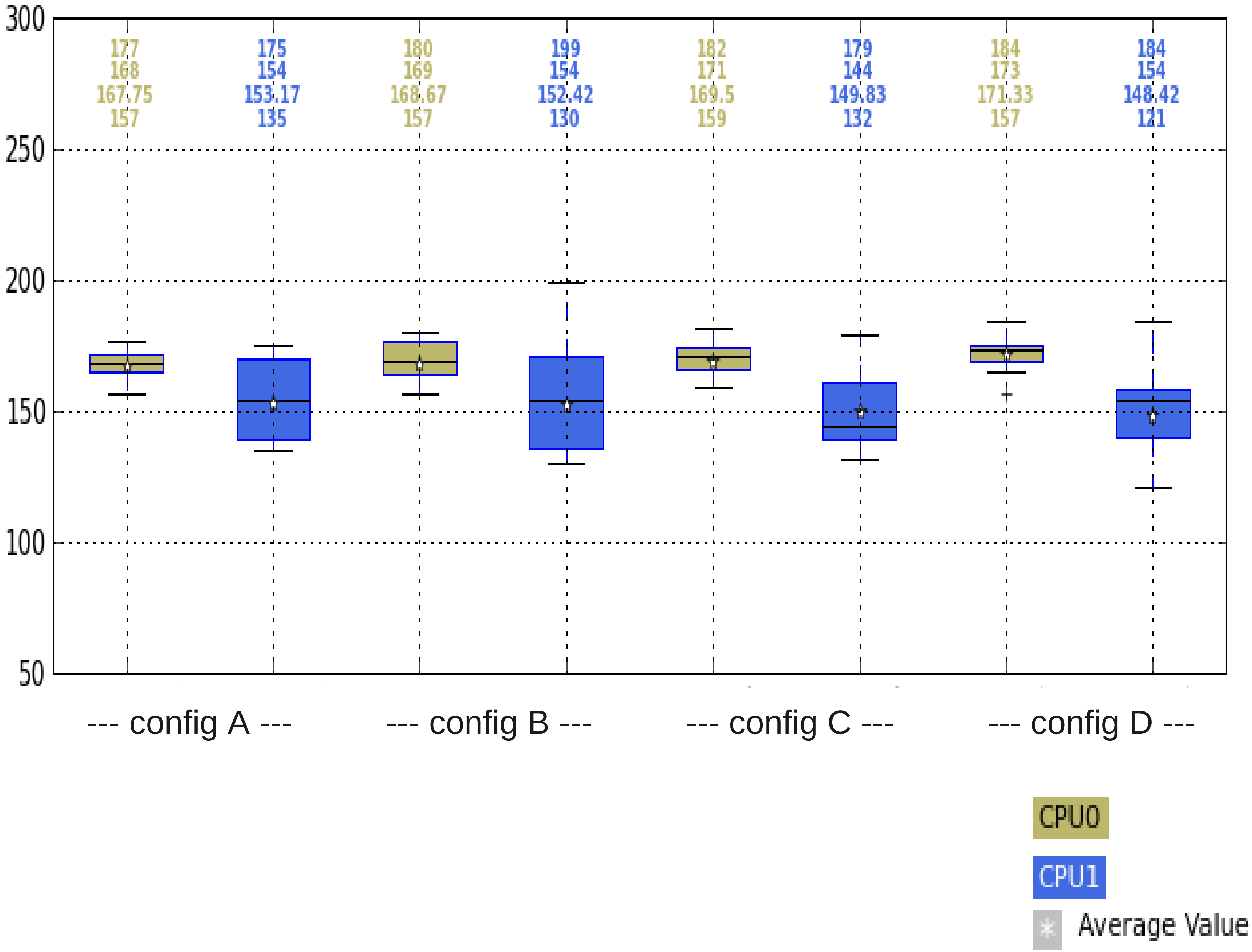
matplotlib box plot

box is: 25%, 50%, 75%
star inside box is the average

whisker end is most extreme value within
1.5 * (75% - 25%), each outlier would also be shown

Numbers above each box are:

- max
- median
- average
- min


300
250
200
150
100
50
177
168
167.75
157
175
154
153.17
135
180
169
168.67
157
199
154
152.42
130
182
171
169.5
159
179
144
149.83
132
184
173
171.33
157
184
154
148.42
121
--- config A ---
--- config B ---
--- config C ---
--- config D ---
CPU0
CPU1
Average Value

matplotlib box plot

An example plot showing outliers is available at:

- http://matplotlib.org/pyplots/boxplot_demo_06.hires.png

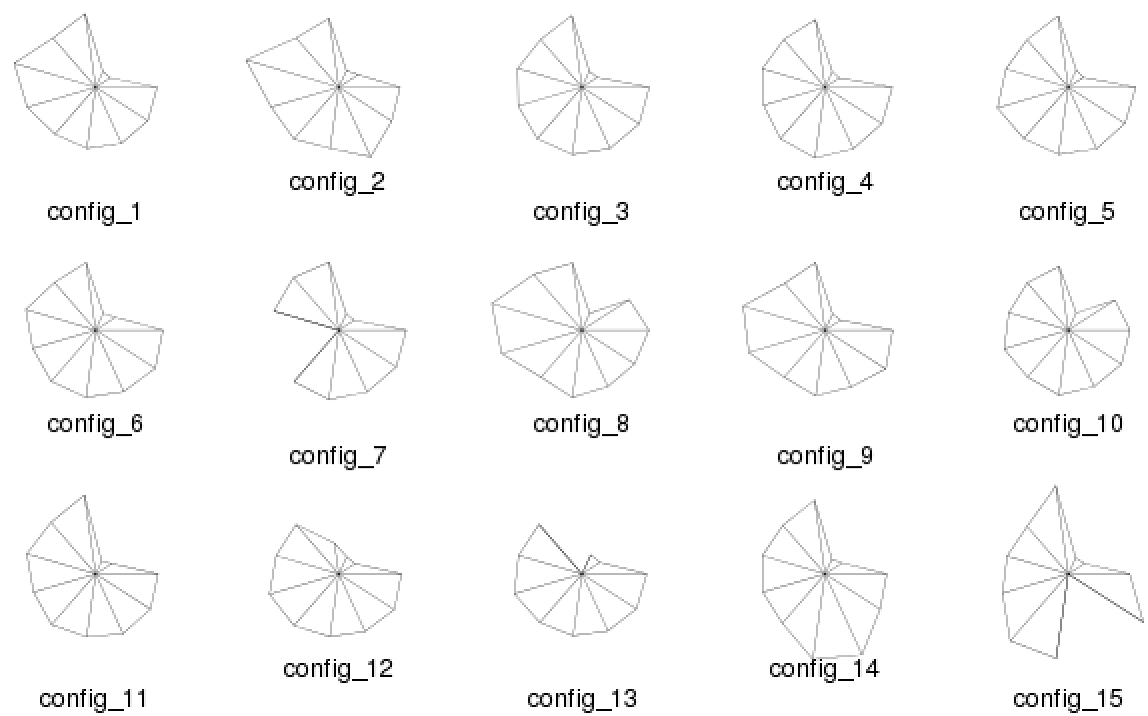
Comparing multiple dimensions

Metric: maximum IRQ disabled time

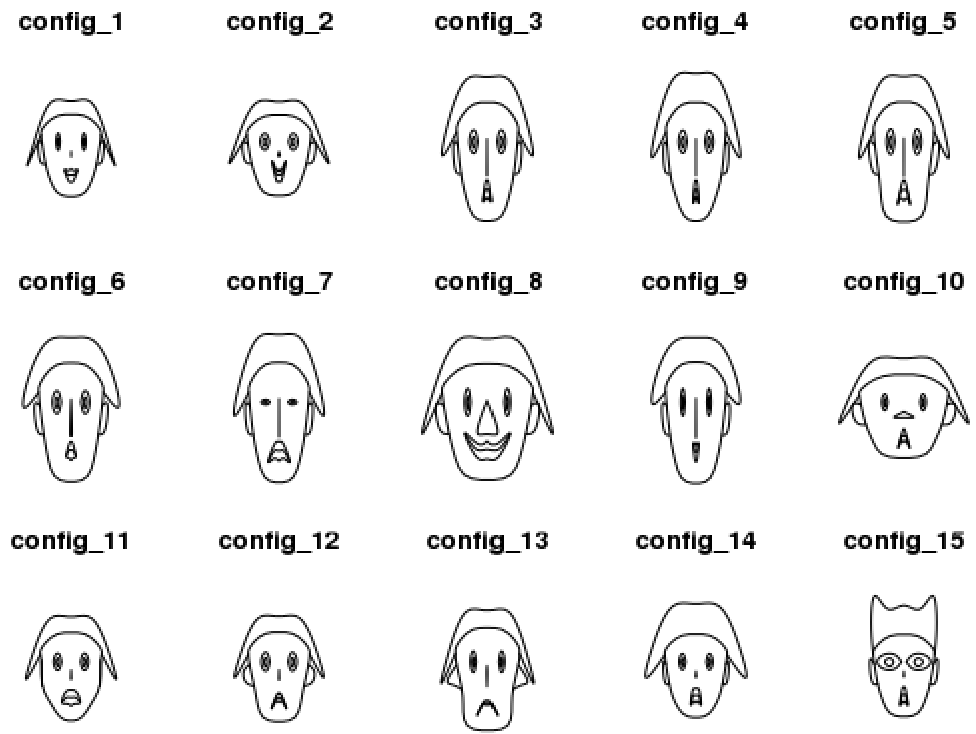
11 types of test for each kernel configuration

15 variants of kernel configuration

max IRQs disabled time
11 test cases



max IRQs disabled time
11 test cases



Final Thoughts

The simple answer is sometimes correct only through chance, dig deeper

Be creative in visualization

Visualization can hide or expose information

Ensure there is a physical meaning underlying the metrics and the visualization of the metrics

THE END

Thank you for your attention...

Questions?

How to get a copy of the slides

1) leave a business card with me

2) frank.rowand@am.sony.com