A Comprehensive Study of Characterizing Program Execution Time

Young-Kyoon Suh

February 16, 2016

1 Experiment Notes

Task Length	Description	Time Length
Regular PUT experiment. Refer to Sections 2, 3, and 4.		
PUT1∼PUT64	Runs of 1000 samples (on `sodb12`).	2013-10-14 ∼ 2013-10-15
PUT128∼PUT2048	Runs of 300 samples (on `sodb12`).	2013-12-12 ∼ 2013-12-21
PUT4096	A run of 300 samples (on `sodb12`).	2014-06-23 ∼ 2014-07-10
PUT8192	Runs of 40/260 samples (on `sodb12`).	2015-04-23 ∼ 2015-04-27 / 2015-10-31 ∼ 2015-11-24
PUT16384	Runs of 40/260 samples (on `sodb12`).	2015-04-23 ∼ 2015-04-23 / 2015-11-25 ∼ 2016-01-14

Table 1: Notes on the regular PUT data used for the histograms

Task Length	Description	Time Length
Regular PUT experiment. Refer to Section 5.		
PUT1	A run of 20k samples on `sodb9`.	2015-12-15 ∼ 2015-12-15
PUT2	A run of 20k samples on `sodb10`.	2015-12-15 ∼ 2015-12-15
PUT16	Runs of 2k, 4k, 8k, 16k, and 32k samples (on `sodb12`).	2016-01-25 ∼ 2016-02-09
Dual PUT experiment. Refer to Section 6.		
PUT4096	A run of 500 samples on `sodb8`.	2015-11-08 ∼ 2015-12-25
PUT2	A run of 1k samples on `sodb9`.	2015-12-27 ∼ 2015-12-27
PUT64	A run of 1k samples on `sodb10`	2015-12-27 ∼ 2015-12-27
PUT4∼PUT32	A run of 1k samples on `sodb9`.	2016-01-27 ∼ 2016-01-31

Table 2: Notes on the new PUT experiments

2 Summary of the EMPv4 data

EMPv4: Running PUT with a specific task length under a controlled environment, with i) daemon processes disabled, ii) the NTP daemon process activated, iii) major CPU features (turbo and speedstep) disabled, and iv) an up-to-date Linux version (RHEL 6.0) installed.

	Num. of Samples	Minimum (msec)	Maximum (msec)	Average (msec)	Std. Dev. (msec)
PUT1	1,000	999.0	1,005.0	1,002.4	0.73
PUT2	1,000	1,996.0	2,007.0	2,004.5	1.38
PUT4	1,000	4,004.0	4,012.0	4,008.6	1.64
PUT8	1,000	8,014.0	8,023.0	8,018.1	1.72
PUT16	1,000	16,029.0	16,041.0	16,034.3	1.86
PUT32	1,000	32,064.0	32,084.0	32,068.2	2.05
PUT64	1,000	64,129.0	64,145.0	64,135.0	2.27
PUT128	300	128,244.0	128,260.0	128,251.2	2.32
PUT256	300	256,494.0	256,523.0	256,502.3	3.29
PUT512	300	512,995.0	513,152.0	513,005.1	9.41
PUT1024	300	1,025,997.0	1,026,141.0	1,026,012.4	11.43
PUT2048	300	2,051,981.0	2,052,156.0	2,052,012.0	11.19
PUT4096	300	4,105,451.0	4,105,629.0	4,105,526.0	25.98
PUT8192	40 (last Apr)	8,207,870.0	8,207,967.0	8,207,918.0	21.03
PUT8192	260 (Nov)	8,210,940.0	8,211,196.0	8,211,049.0	36.60
PUT16384	40 (last Apr)	16,415,757.0	16,415,964.0	16,415,810.3	40.43
PUT16384	260 (Nov)	16,422,028	16,422,389	16,422,153.0	52.54

Table 3: PT statistics by EMPv4 (See Table 1.)

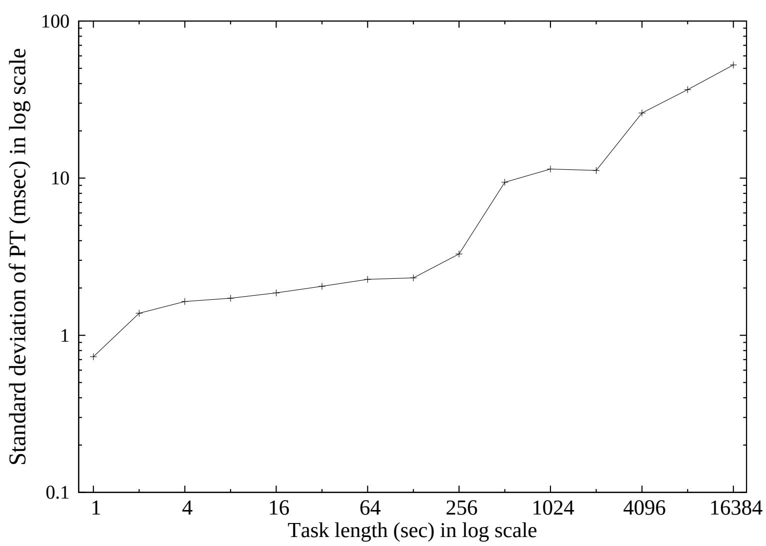
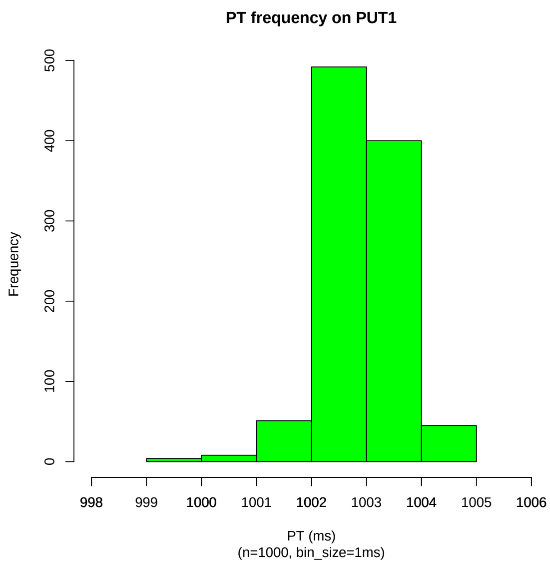


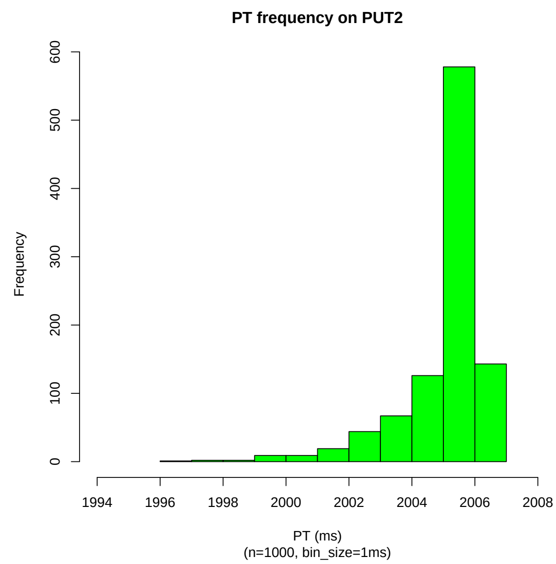
Figure 1: Std. dev. of PT over increasing task length (See Table 1.)

3 Histograms on the EMPv4 Data

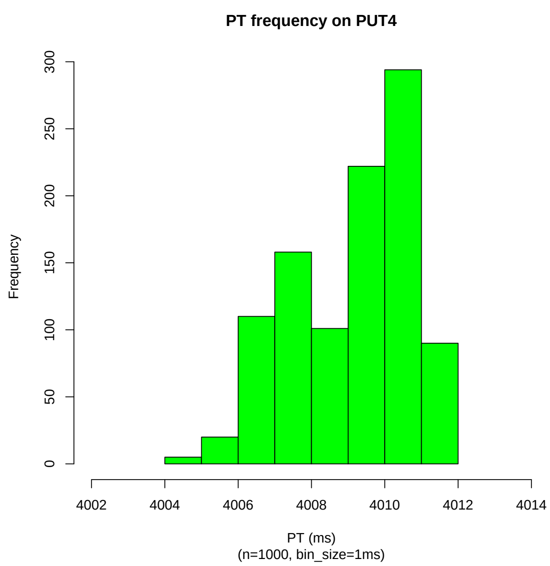
The base data of the following histograms are from Table 1.



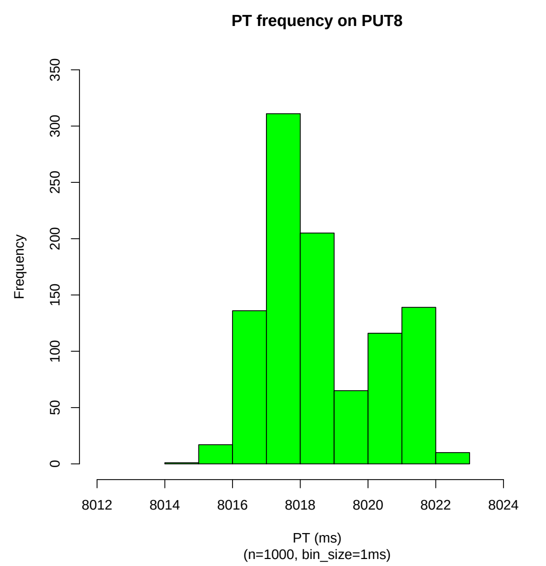
(a) PT frequency on PUT1



(b) PT frequency on PUT2

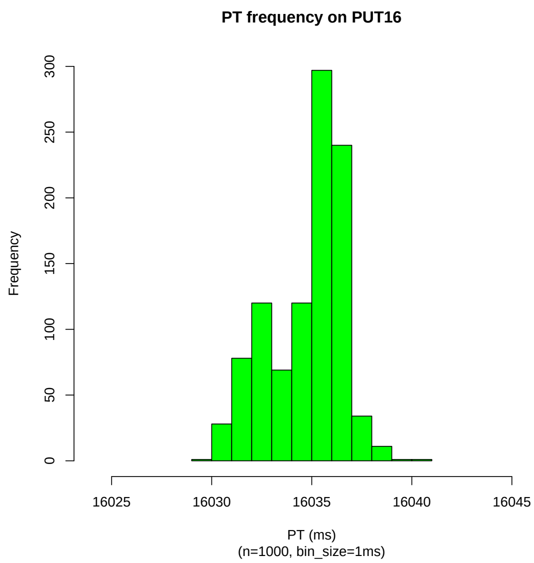


(c) PT frequency on PUT4

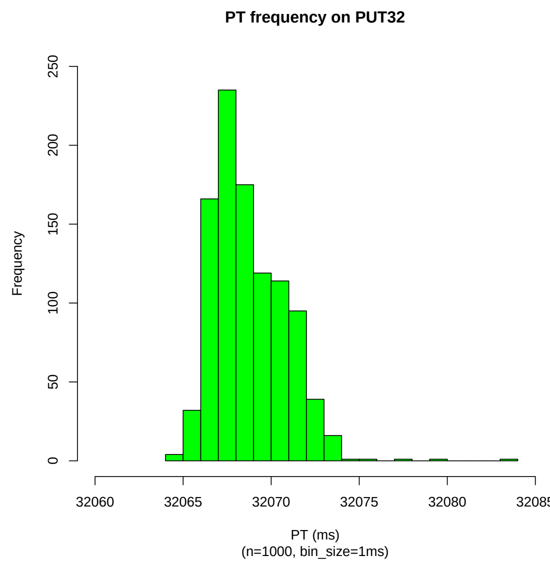


(d) PT frequency on PUT8

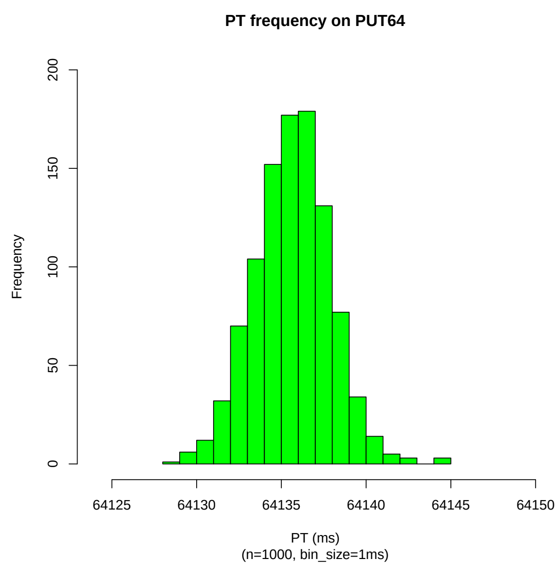
Figure 2: PT Histograms of PUT1 ... PUT8



(a) PT frequency on PUT16

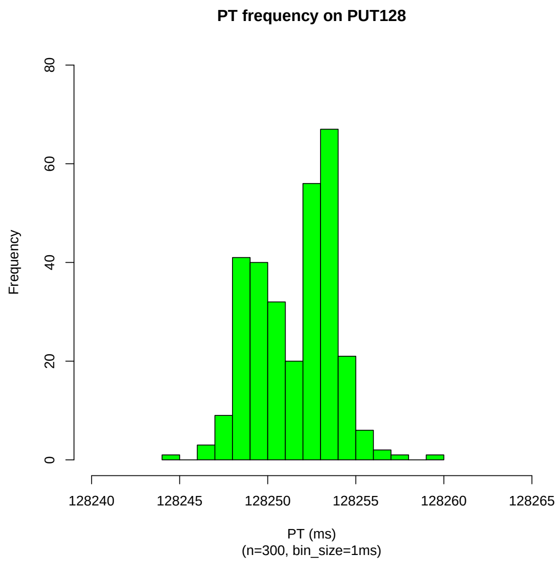


(b) PT frequency on PUT32

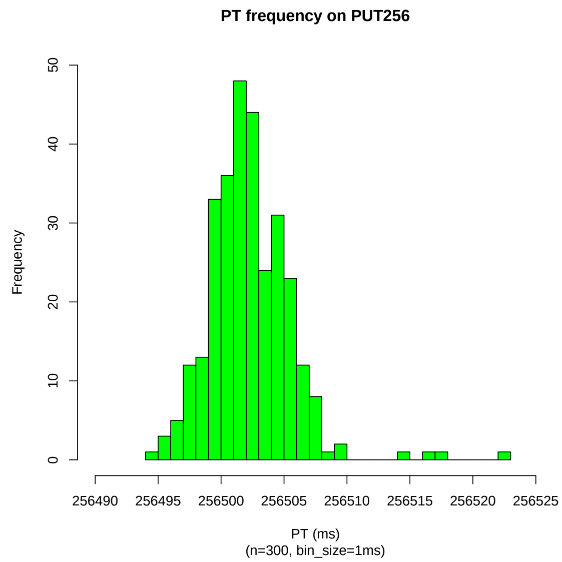


(c) PT frequency on PUT64

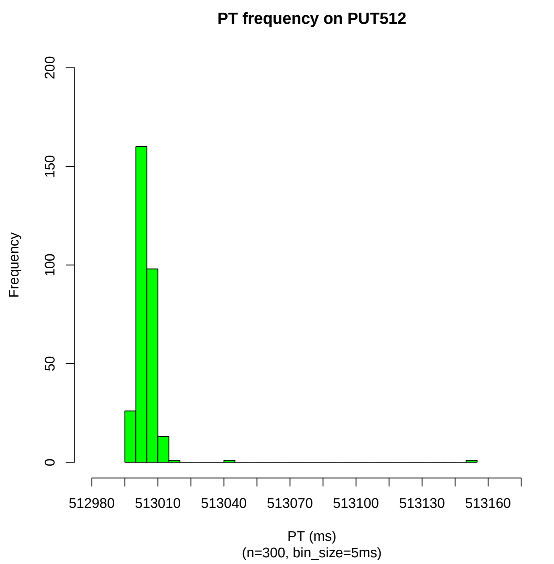
Figure 3: PT Histograms of PUT16 ... PUT64



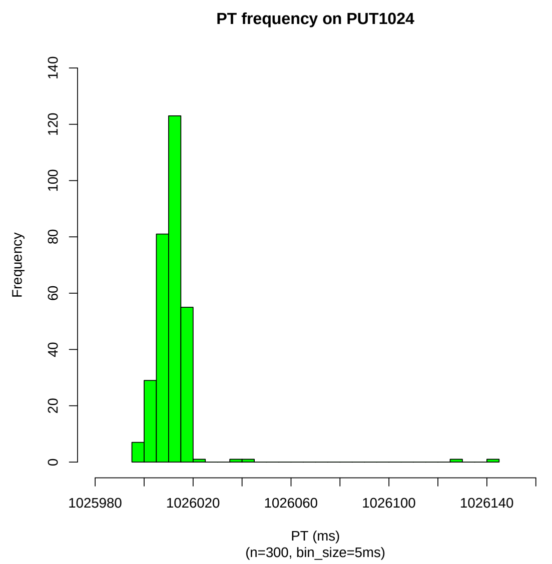
(a) PT frequency on PUT128



(b) PT frequency on PUT256

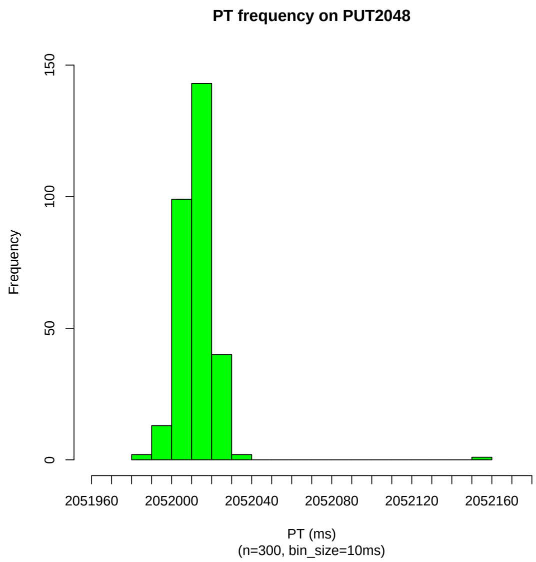


(c) PT frequency on PUT512

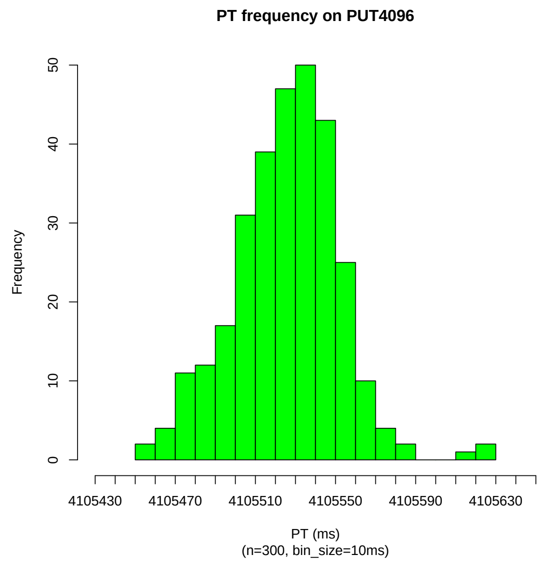


(d) PT frequency on PUT1024

Figure 4: PT Histograms of PUT128 ... PUT1024

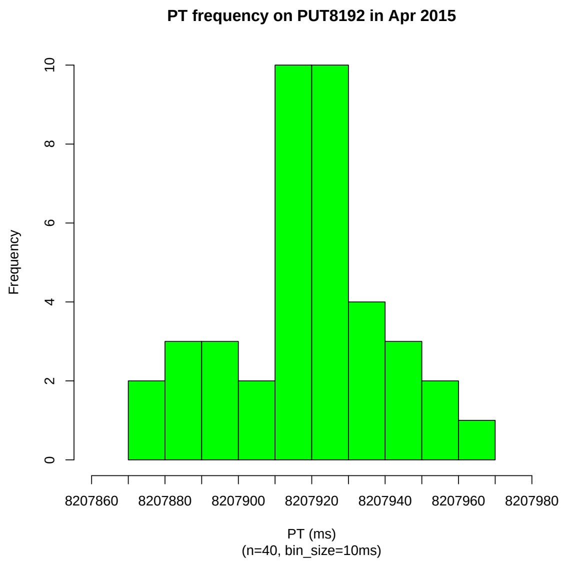


(a) PT frequency on PUT2048

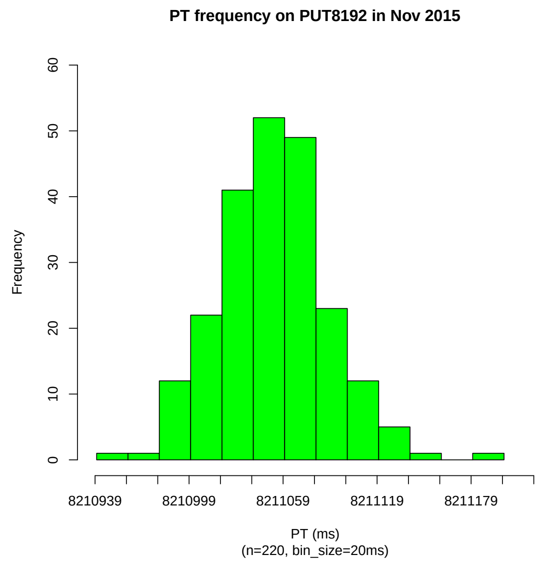


(b) PT frequency on PUT4096

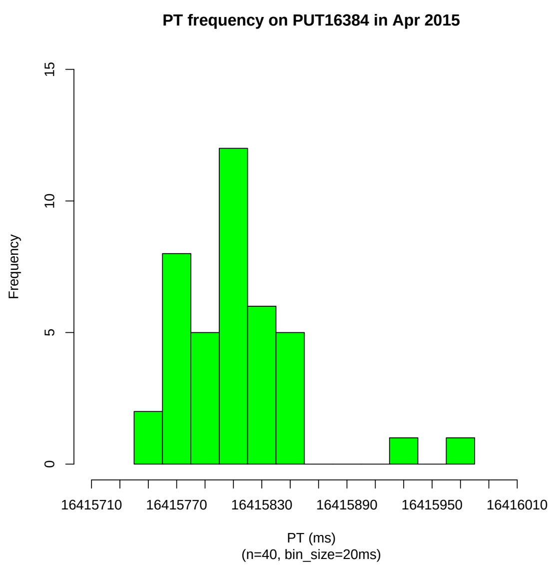
Figure 5: PT Histograms of PUT2048 and PUT4096



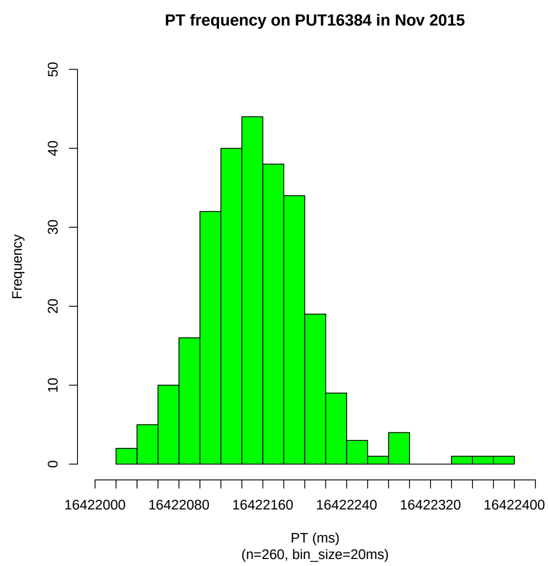
(a) PT frequency on PUT8192 with 40 samples (See Table 1.)



(b) PT frequency on PUT8192 with 260 samples (See Table 1.)



(c) PT frequency on PUT16384 with 40 samples (See Table 1.)

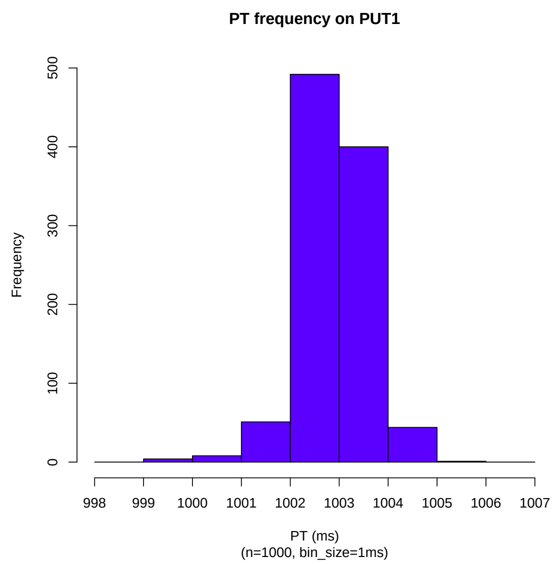


(d) PT frequency on PUT16384 with 260 samples (See Table 1.)

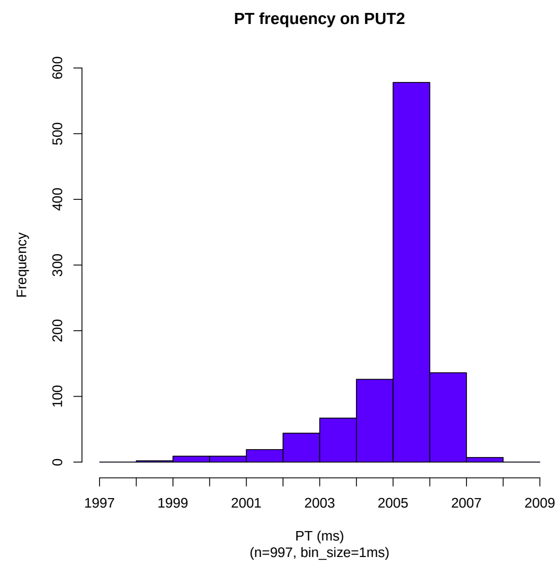
Figure 6: PT Histograms of PUT8192 and PUT16384

4 Histograms on the EMPv5 Data

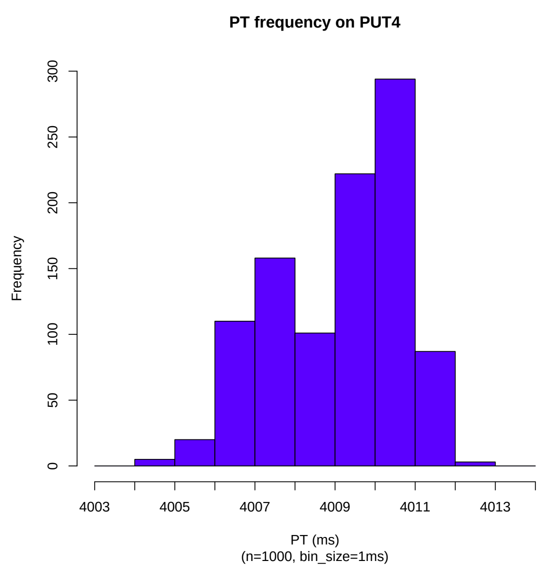
The base data of the following histograms are from Table 1. EMPv5(-relaxed) trims outliers from the data of each PUT by EMPv4. To be more specific, for each run of PUT an outlier is determined as the one above and below the average ± *five[1]* standard deviations computed from the EMPv4 data.



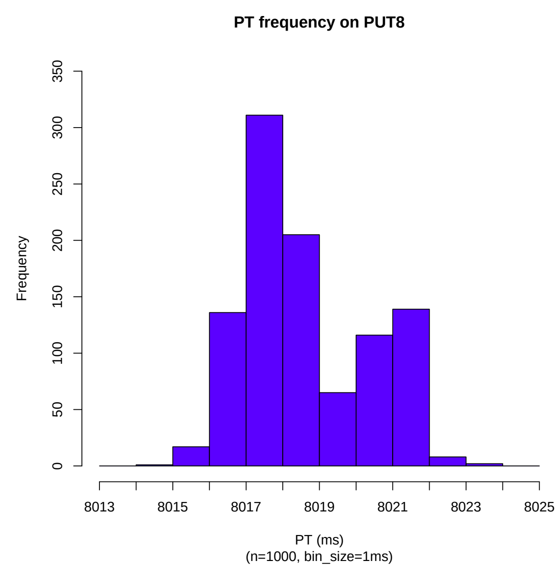
(a) PT frequency on PUT1



(b) PT frequency on PUT2



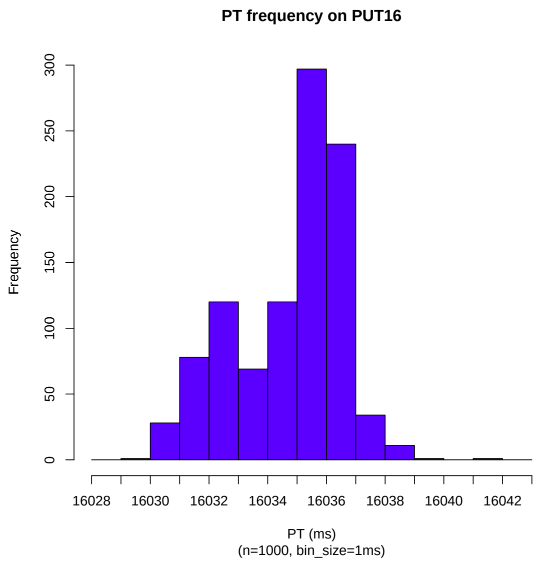
(c) PT frequency on PUT4



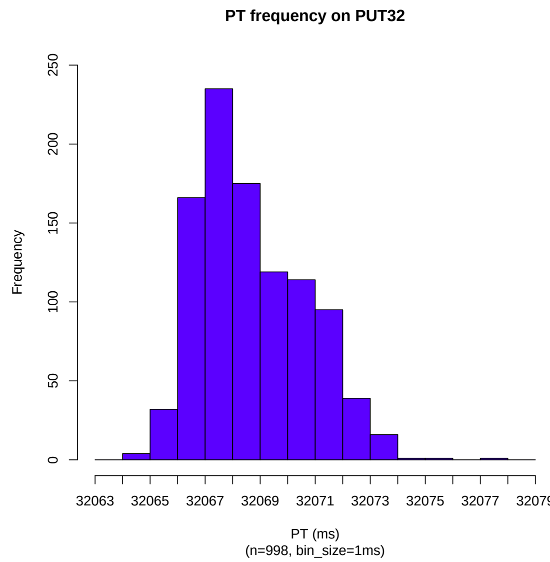
(d) PT frequency on PUT8

Figure 7: PT Histograms of PUT1 ... PUT8

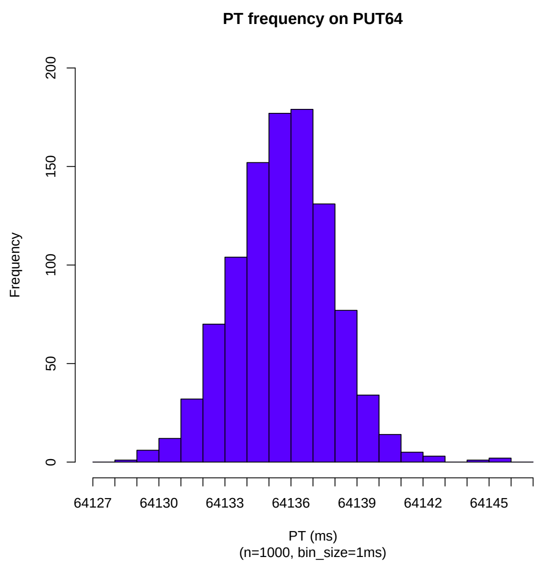
[1]In the stricter version, we use *two*.



(a) PT frequency on PUT16

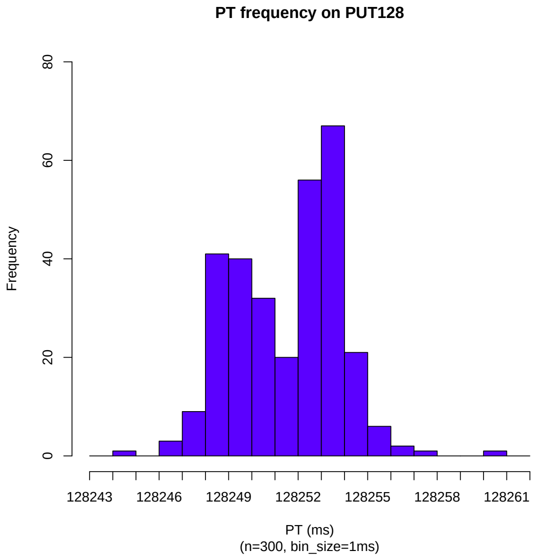


(b) PT frequency on PUT32

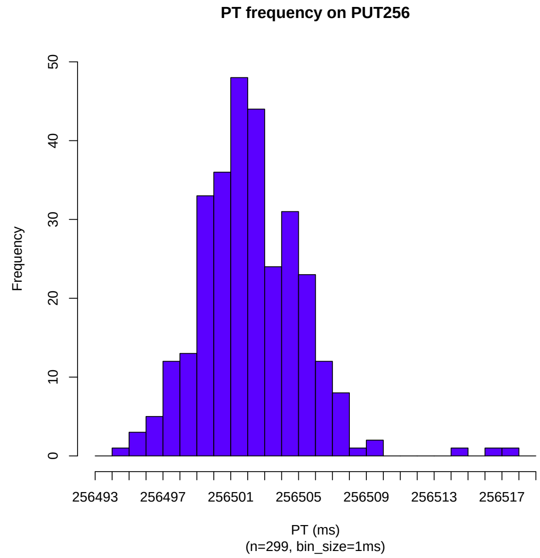


(c) PT frequency on PUT64

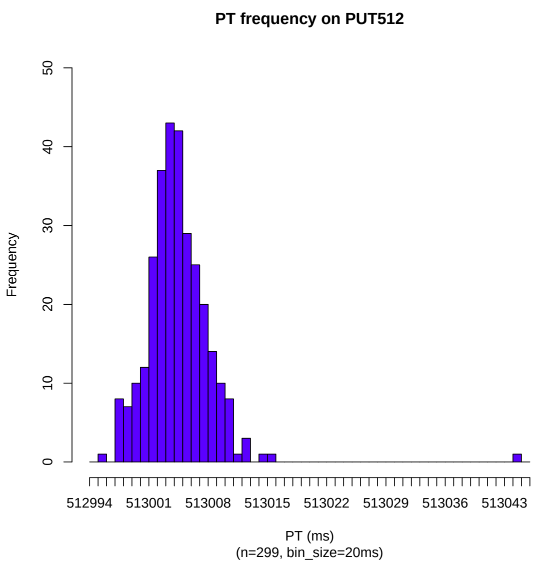
Figure 8: PT Histograms of PUT16 ... PUT64



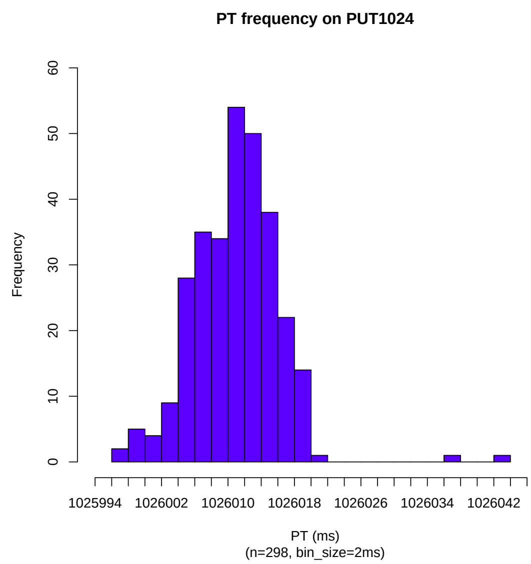
(a) PT frequency on PUT128



(b) PT frequency on PUT256

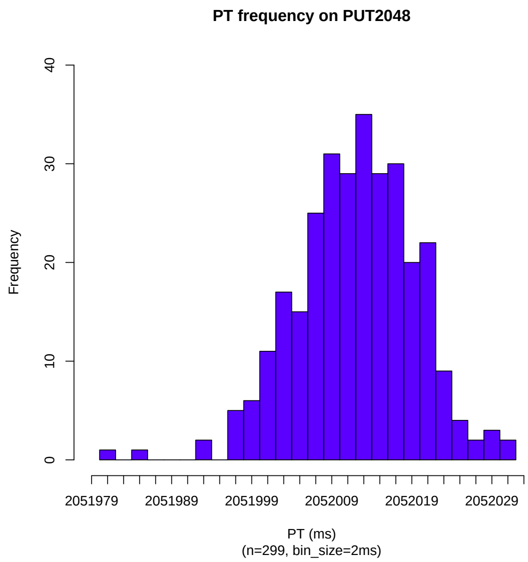


(c) PT frequency on PUT512

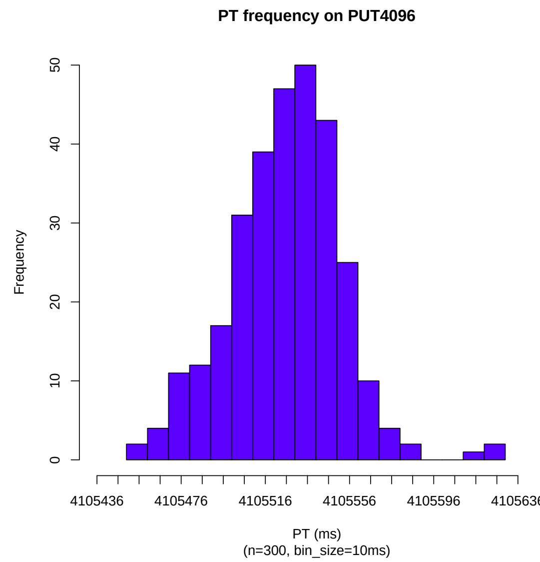


(d) PT frequency on PUT1024

Figure 9: PT Histograms of PUT128 ... PUT1024

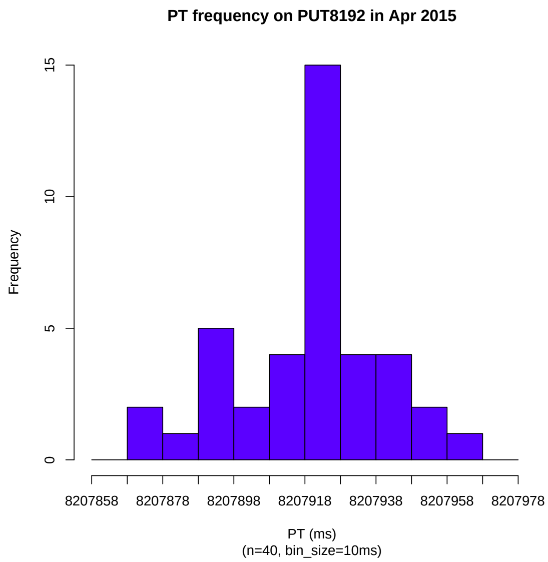


(a) PT frequency on PUT2048

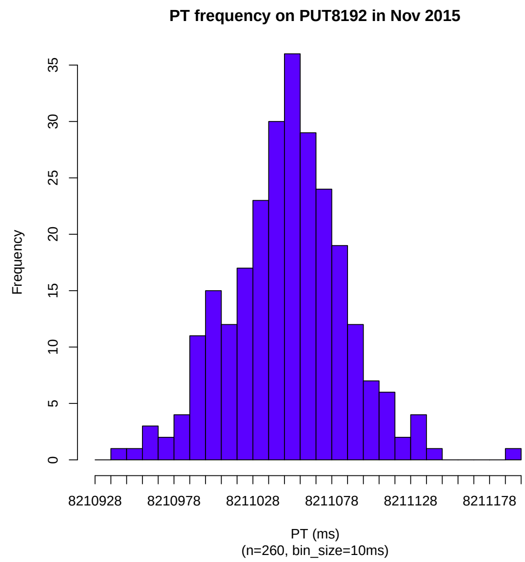


(b) PT frequency on PUT4096

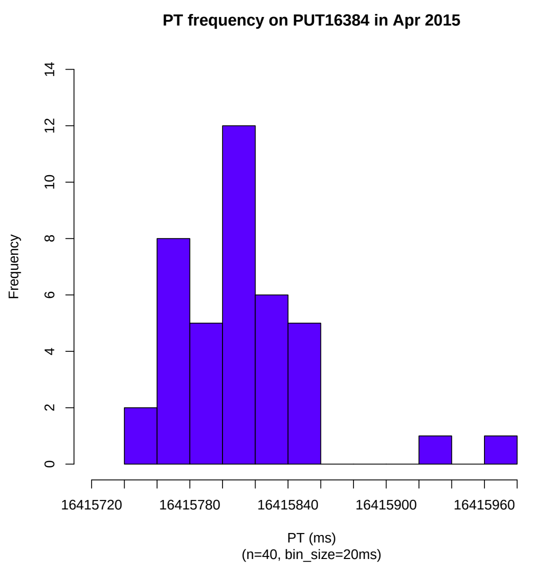
Figure 10: PT Histograms of PUT2048 and PUT4096



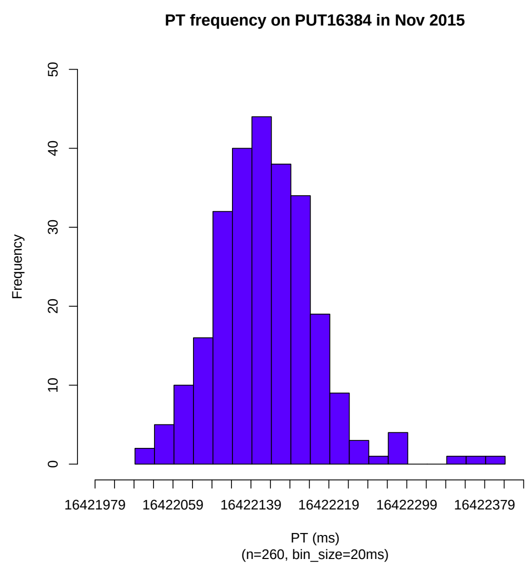
(a) PT frequency on PUT8192 with 40 samples (See Table 1.)



(b) PT frequency on PUT8192 with 260 samples (See Table 1.)



(c) PT frequency on PUT16384 with 40 samples (See Table 1.)



(d) PT frequency on PUT16384 with 260 samples (See Table 1.))

Figure 11: PT Histograms of PUT8192 and PUT16384

5 Sample Size vs. Standard Deviation of PT

The base data of the following histograms are from Table 2.

5.1 PUT1 and PUT2

Table 4 exhibits varying standard deviations over increasing sample size on PUT1 and PUT2. EMPv4 is applied to the table's data.

Num. of Samples	Std. Dev. (msec)	
	PUT1	PUT2
1,000	1.07	1.40
2,000	1.06	1.39
3,000	1.07	1.38
4,000	1.07	1.37
5,000	1.07	1.40
6,000	1.06	1.70
7,000	1.06	1.65
8,000	1.07	1.62
9,000	1.07	1.60
10,000	1.07	1.58
11,000	1.08	1.57
12,000	1.08	1.56
13,000	1.08	1.54
14,000	1.08	1.53
15,000	1.08	1.52
16,000	1.08	1.51
17,000	1.08	1.50
18,000	1.08	1.50
19,000	1.08	1.50
20,000	1.08	1.49

Table 4: Std. Dev. of PUT1 and PUT2 over increasing sample size

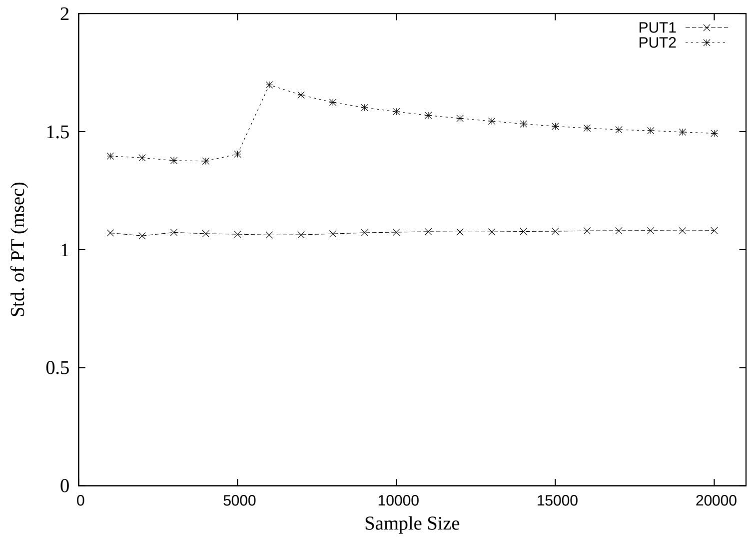


Figure 12: Std. dev. of PT on PUT1 and PUT2 over increasing sample size

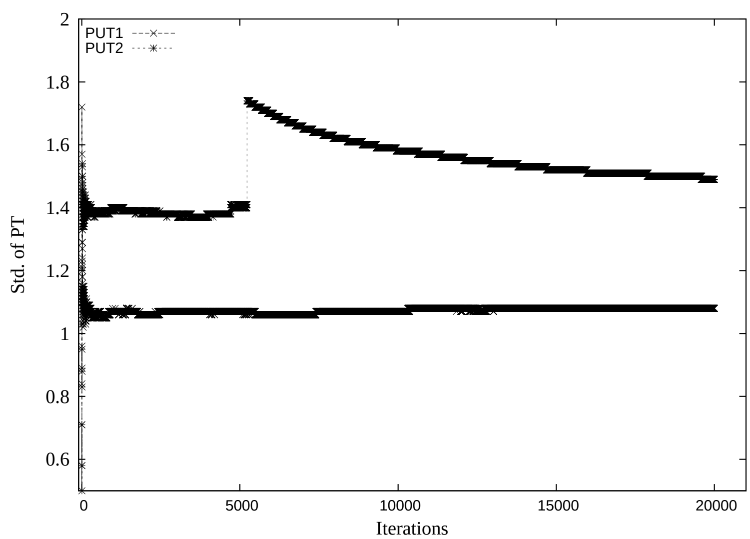
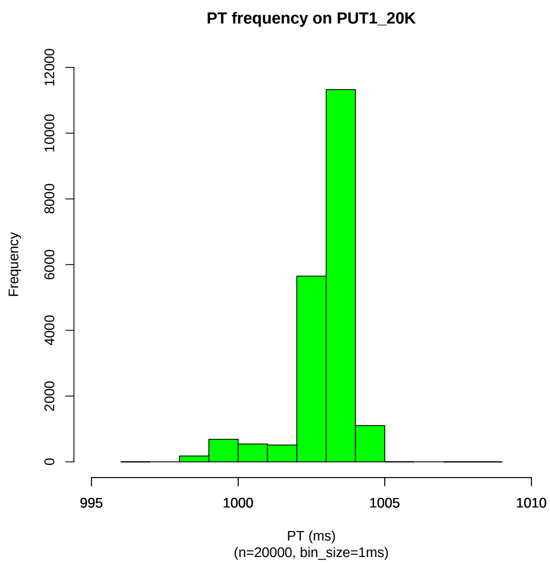


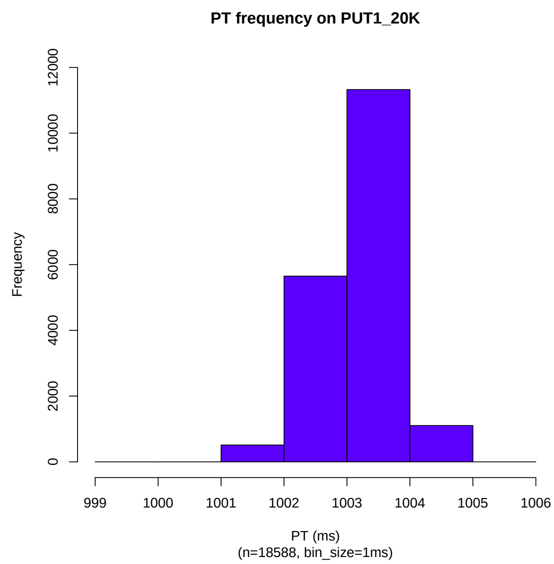
Figure 13: Std. dev. of PT on PUT1 and PUT2 over increasing sample size

PUT2	Program Time
`incr_work`	2078 msecs (at the 5276th iteration)
Daemon Processes	**Program Time**
`md0_raid1`	1 msec
`proc_monitor`	198 msecs
`rhn_check`	460 msecs
Total	659 msecs

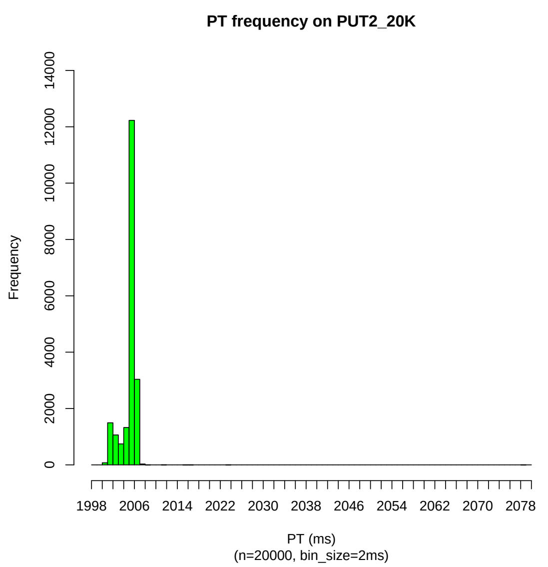
Table 5: The daemon processes captured at the hike of PUT2



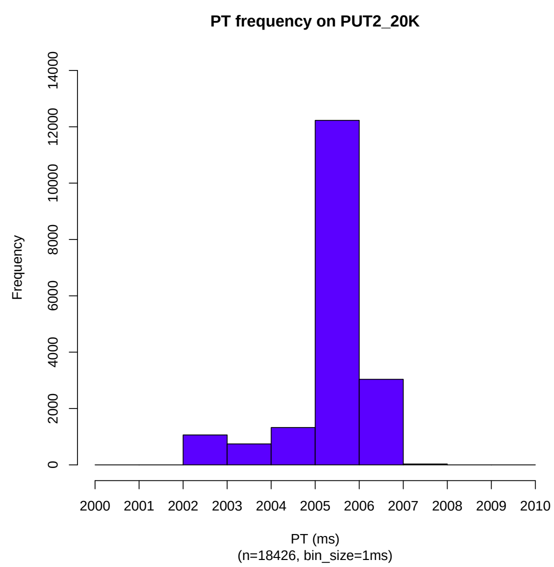
(a) PT frequency on PUT1 by EMPv4 (See Table 2.)



(b) PT frequency on PUT1 by EMPv5 (See Table 2.)



(c) PT frequency on PUT2 by EMPv4 (See Table 2.)



(d) PT frequency on PUT2 by EMPv5 (See Table 2.)

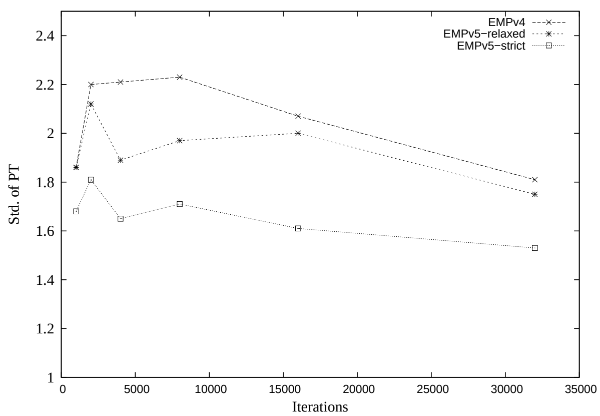
Figure 14: PT Histograms of PUT1 and PUT2 by 20,000 trials

5.2 PUT16

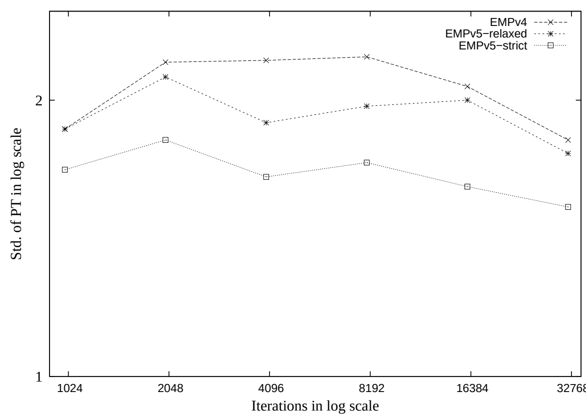
In this experiment we ran PUT16 up to 32,000 from 1,000 times by a factor of two. The relaxed version of EMPv5 (called *EMPv5-relaxed*) uses *five* standard deviations whereas its strict version (called *EMPv5-strict*) does *two* standard deviations for a vertical gap below and above the average. (Young: 2k samples seem most appropriate to represent the whole population of PUT16, in that the standard deviations by EMPv5 on the 2k sample size are almost at peak compared to those of the other sample sizes.)

Num. of Samples	Std. Dev. (msec)		
	EMPv4	EMPv5-relaxed	EMPv5-strict
1,000	1.86	1.86	1.68
2,000	2.20	2.12	1.81
4,000	2.21	1.89	1.65
8,000	2.23	1.97	1.71
16,000	2.07	2.00	1.61
32,000	1.81	1.75	1.53

Table 6: Standard deviations of PUT16 over increasing sample size



(a) Standard deviations of PT on PUT16

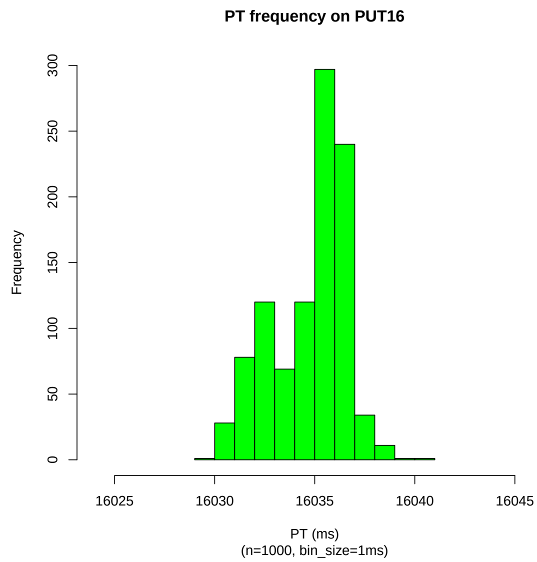


(b) Standard deviations of PT on PUT16, in log scale

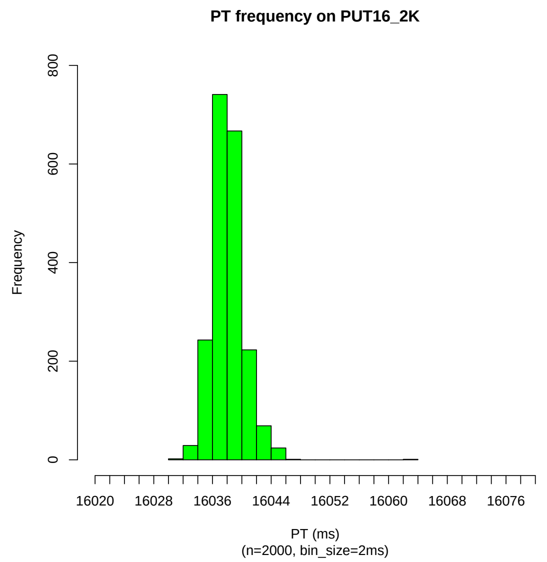
Figure 15: Standard deviations of PT on PUT16 over increasing sample size

5.3 Histograms by EMPv4

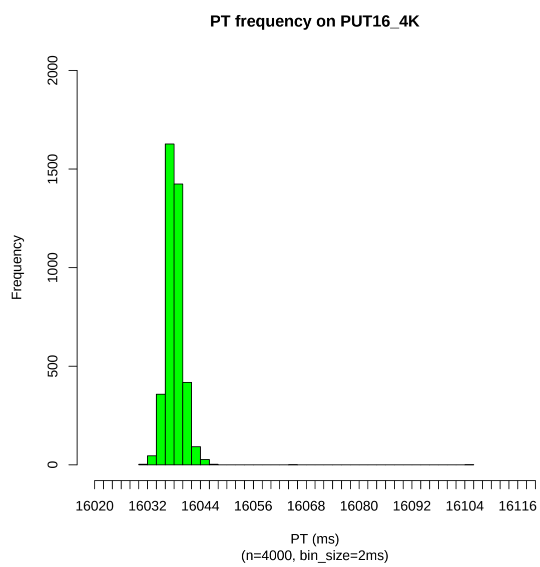
We apply EMPv4 to the runs of PUT16 as mentioned above. The following histograms are the results of EMPv4.



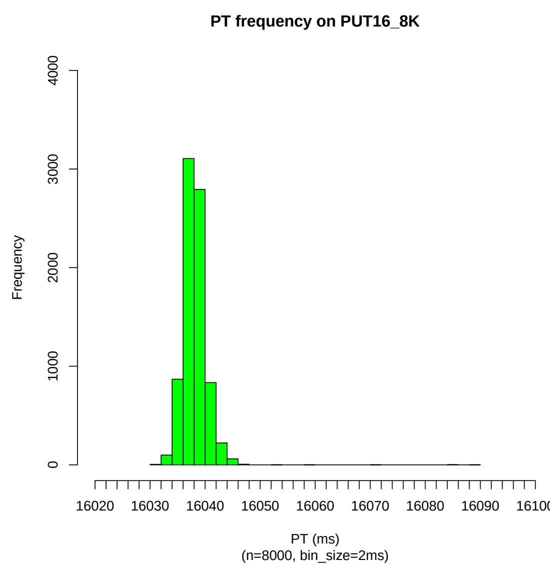
(a) PT frequency on PUT16 with 1k samples



(b) PT frequency on PUT16 with 2k samples

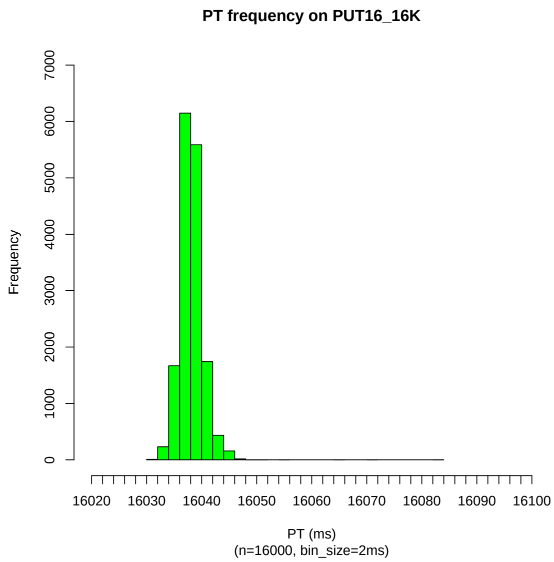


(c) PT frequency on PUT16 with 4k samples

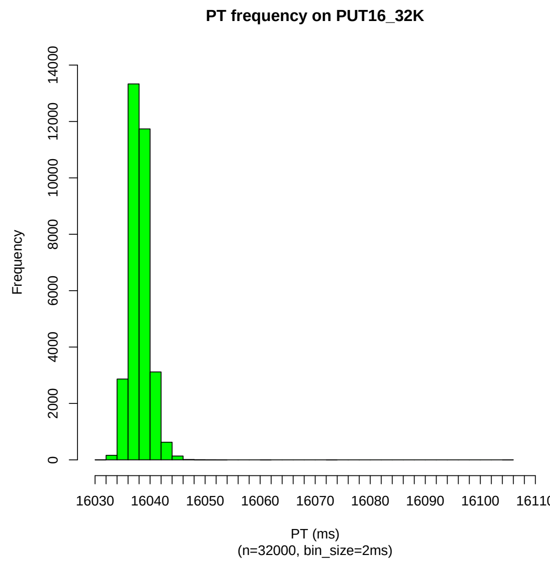


(d) PT frequency on PUT16 with 8k samples

Figure 16: PT histogram of PUT16 by EMPv4, with the sample size increasing from 1k to 8k



(a) PT frequency on PUT16 with 16k samples

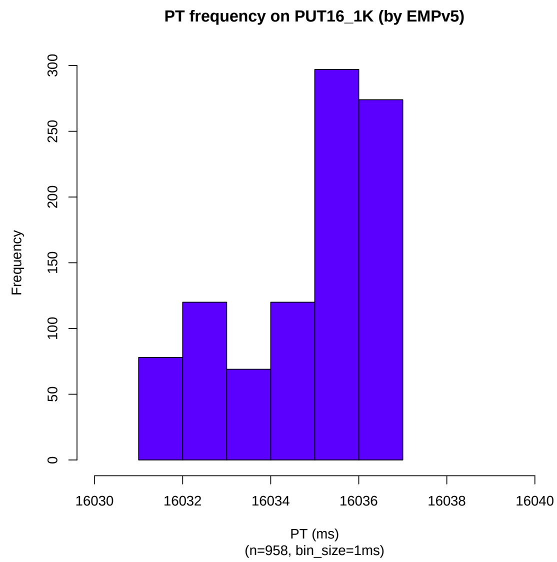


(b) PT frequency on PUT16 with 32k samples

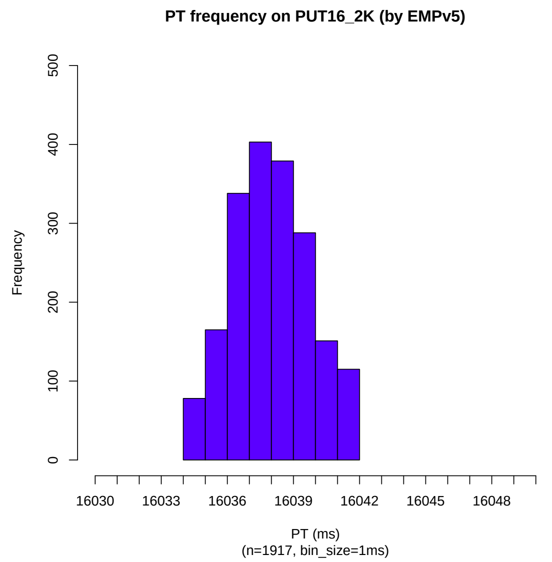
Figure 17: PT histogram of PUT16 by EMPv4, with the sample size increasing from 16k to 32k

5.4 Histograms by EMPv5

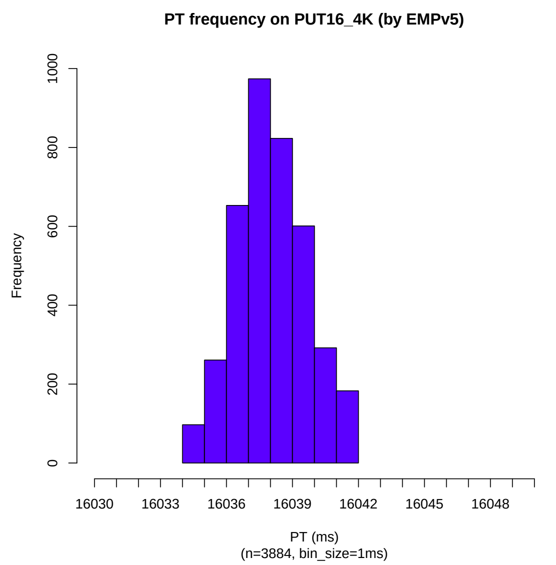
We now apply EMPv5 to the same data of PUT16. To be more specific, we use EMPv5-strict, by which the following histograms are obtained.



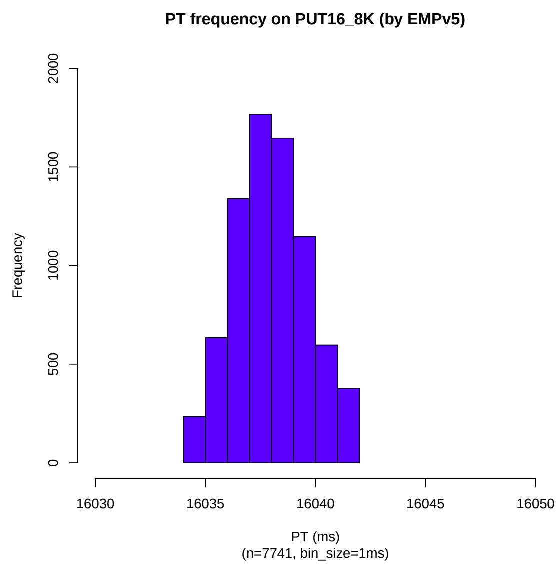
(a) PT frequency on PUT16 with 1k samples



(b) PT frequency on PUT16 with 2k samples

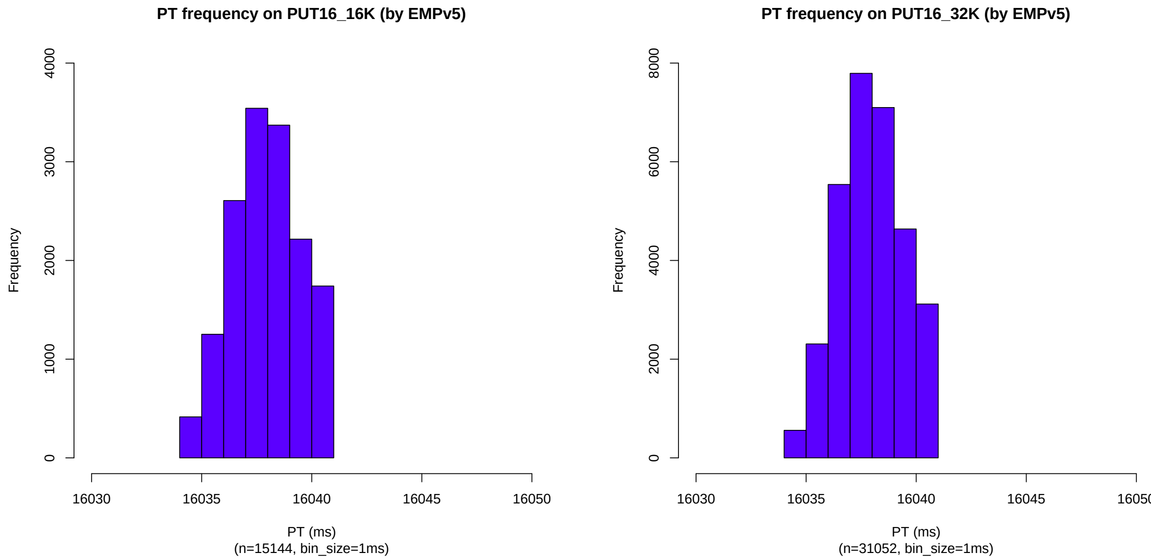


(c) PT frequency on PUT16 with 4k samples



(d) PT frequency on PUT16 with 8k samples

Figure 18: PT histogram of PUT16 by EMPv5, with the sample size increasing from 1k to 8k



(a) PT frequency on PUT16 with 16k samples

(b) PT frequency on PUT16 with 32k samples

Figure 19: PT histogram of PUT16 by EMPv5, with the sample size increasing from 16k to 32k

6 Dual PUT Experiment

In this section we study the characteristics of program times measured in dual PUT experiment. Dual experiment is devised to see whether "internal dependency" of measured times within the same run exists. For this dual experiment, we measure and compare the execution time of the first half (part I) and second half (part II) of each PUT. Specifically, we compute correlation coefficients between corresponding measured times of parts I and II, within the same run of each PUT. In this experiment we expect that little dependency will be observed within the same run for any PUT.

Note that after initial analysis on the data of dual PUT2, PUT64, and PUT4096, we conducted additional experiments of dual PUT4, PUT8, PUT16, and PUT32. It was because some dependency (a correlation efficient of 0.3) was observed at PUT2 and we wanted to check out from which such dependency began between PUT2 and PUT64.

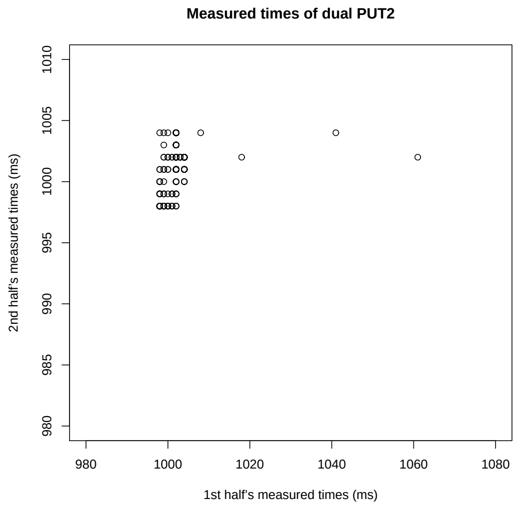
The base data of the following table and histograms are from Table 2.

	Correlation Coefficient	Sample Size
PUT2	0.3	1,000
PUT4	-0.07 (-0.15 except max)	1,000
PUT8	0.8	1,000
PUT16	0.3	1,000
PUT32	-0.01	1,000
PUT64	-0.01	1,000
PUT4096	-0.01	500

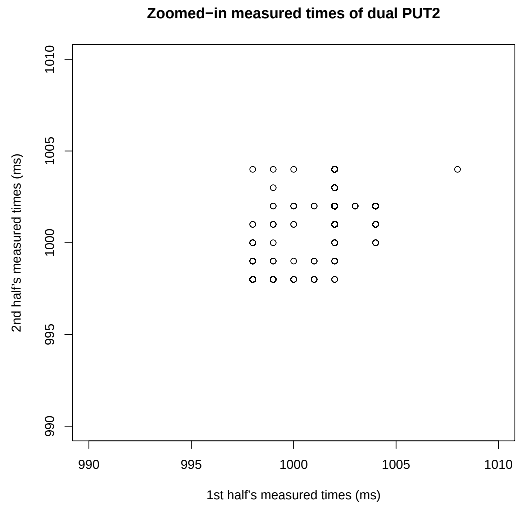
Table 7: Overall statistics of dual PUT experiment

6.1 Scatter Plots

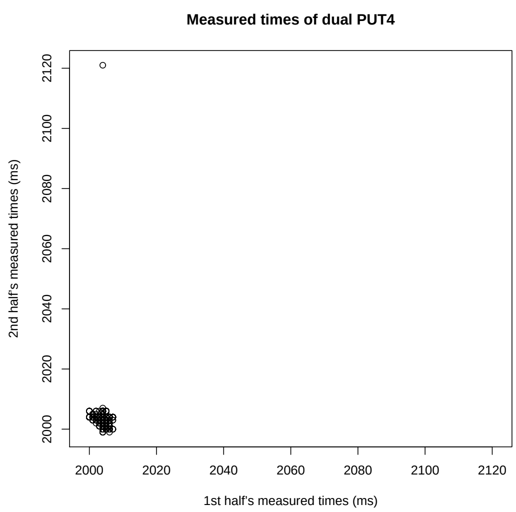
In this section we plot measures times of dual PUT experiments. We provide not only scatter plots of raw data but also those of focused clouds to further look inside.



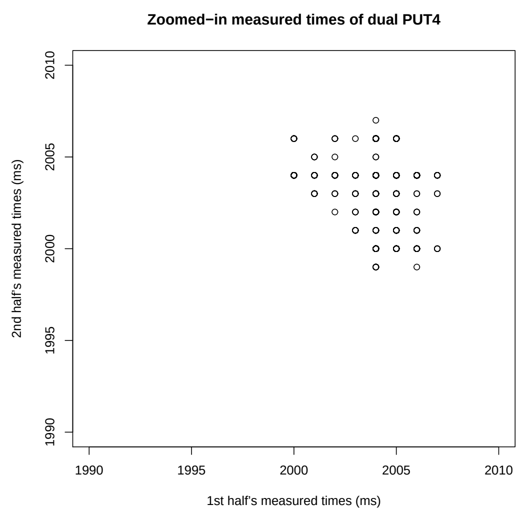
(a) Measured times on dual PUT2



(b) Zoomed-in measured times on dual PUT2

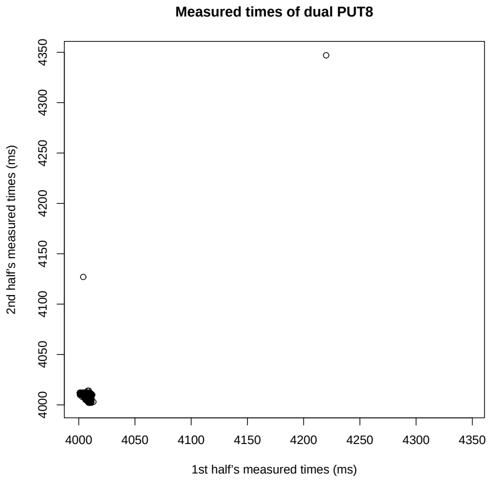


(c) Measured times on dual PUT4

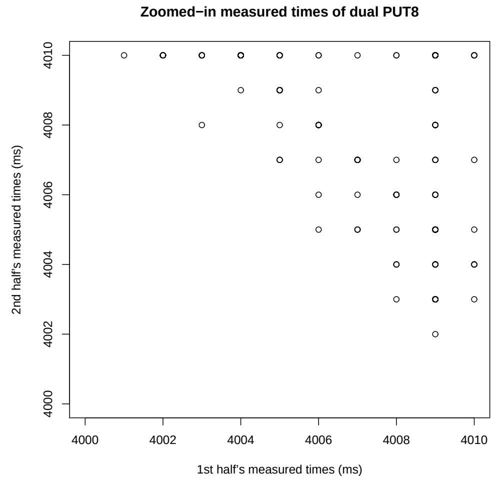


(d) Zoomed-in measured times on dual PUT4

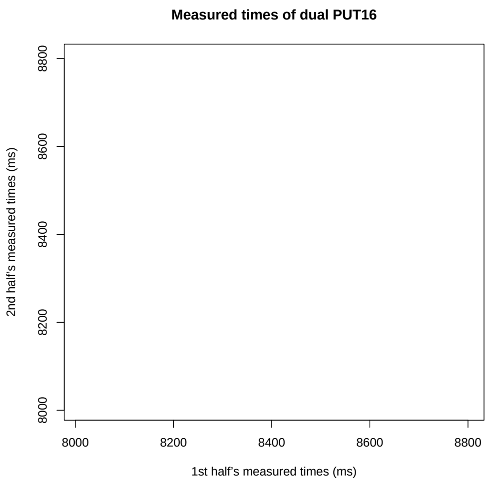
Figure 20: Scatter plots on dual PUT2∼PUT8



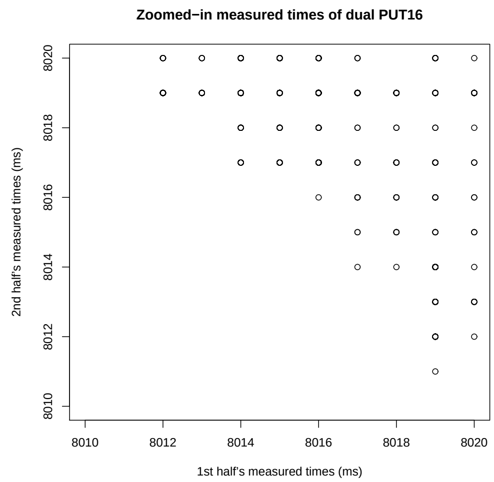
(a) Measured times on dual PUT8



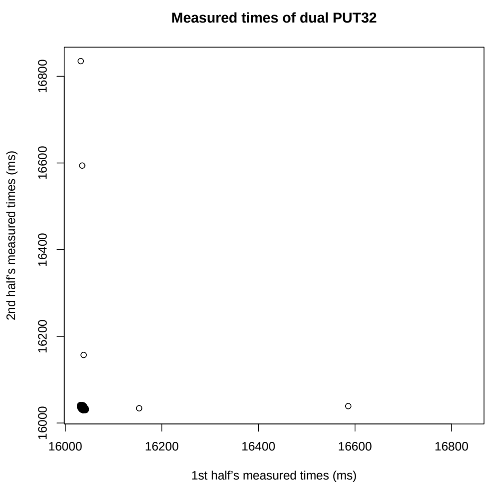
(b) Zoomed-in measured times on dual PUT8



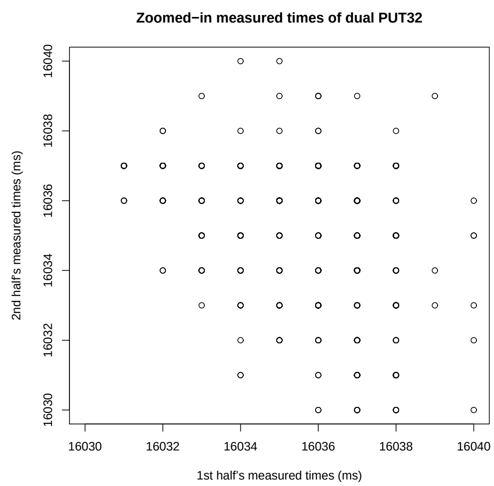
(c) Measured times on dual PUT16



(d) Zoomed-in measured times on dual PUT16

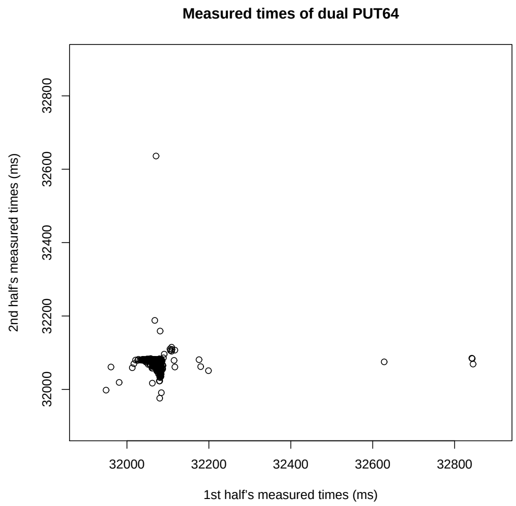


(e) Measured times on dual PUT32

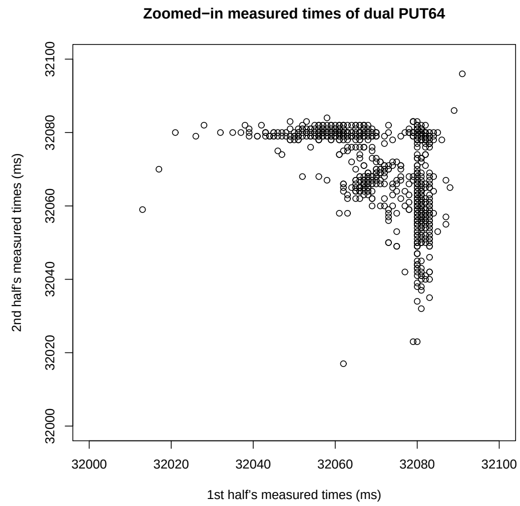


(f) Zoomed-in measured times on dual PUT32

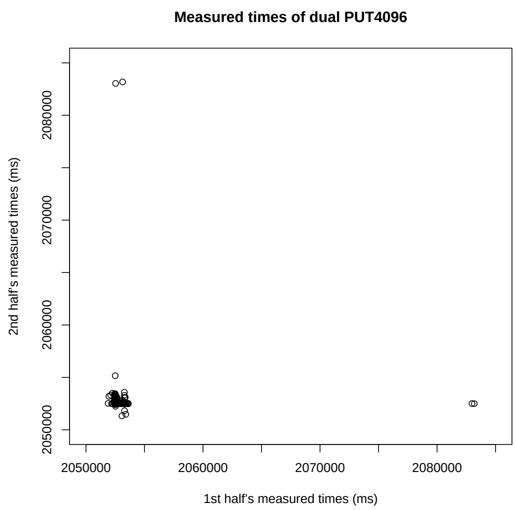
Figure 21: Scatter plots on dual PUT16∼PUT64



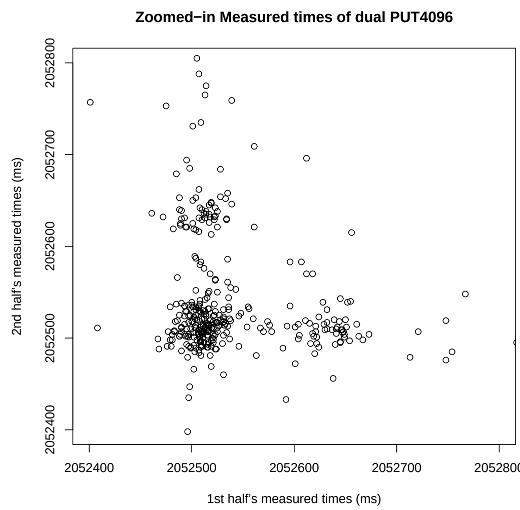
(a) Measured times on dual PUT64



(b) Zoomed-in measured times on dual PUT64



(c) Measured times on dual PUT4096

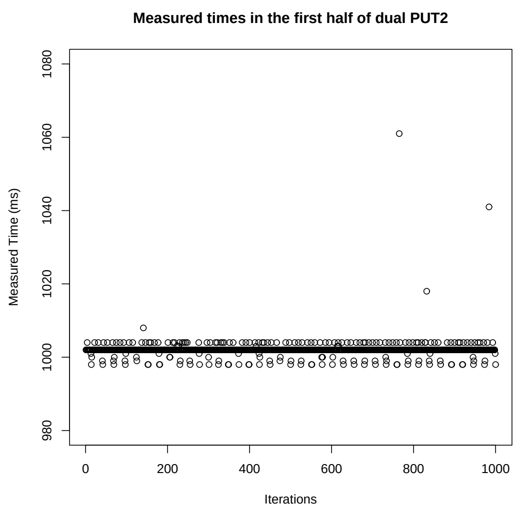


(d) Zoomed-in measured times on dual PUT4096

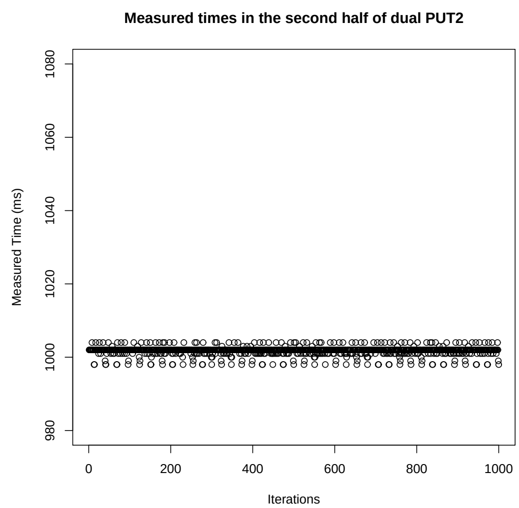
Figure 22: Scatter plots on dual PUT64∼PUT4096

6.2 Program Time Comparison

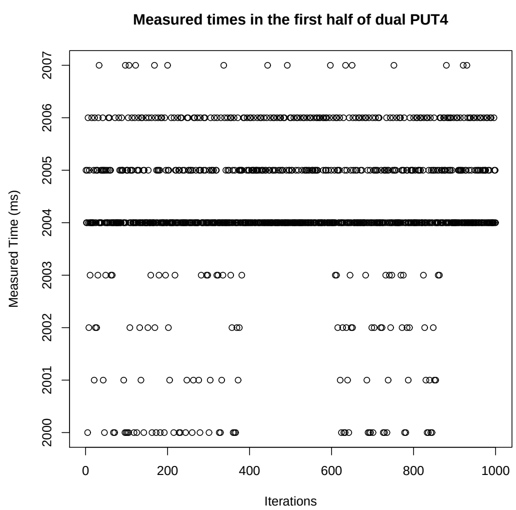
In this section we perform one-to-one comparison on measured times of parts I and II for the same iteration of each PUT.



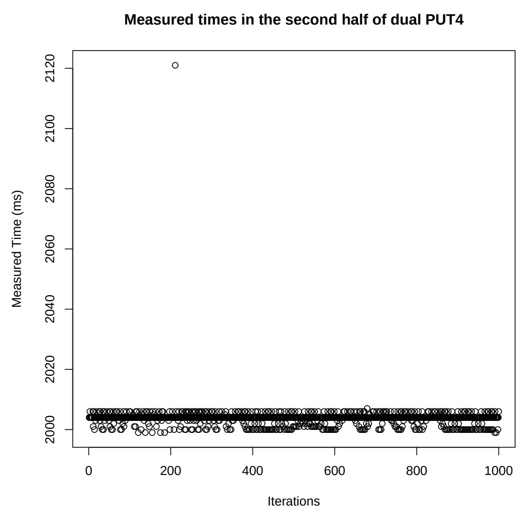
(a) 1st half's measured times on dual PUT2



(b) 2nd half's measured times on dual PUT2

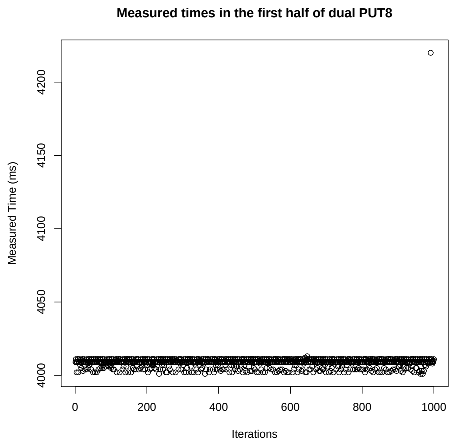


(c) 1st half's measured times on dual PUT4

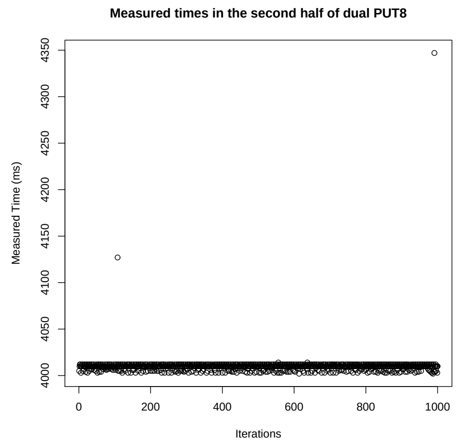


(d) 2nd half's measured times on dual PUT4

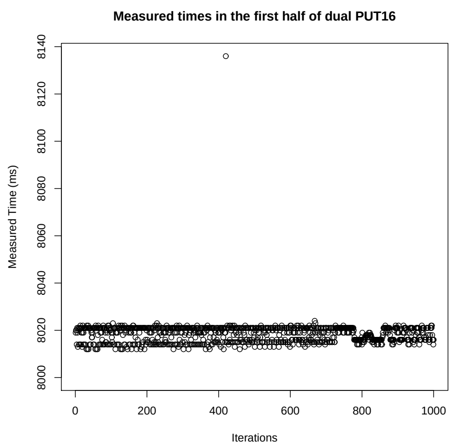
Figure 23: Measured time comparison on dual PUT2∼PUT8



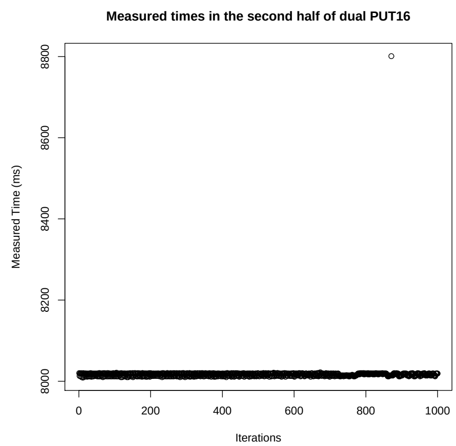
(a) 1st half's measured times on dual PUT8



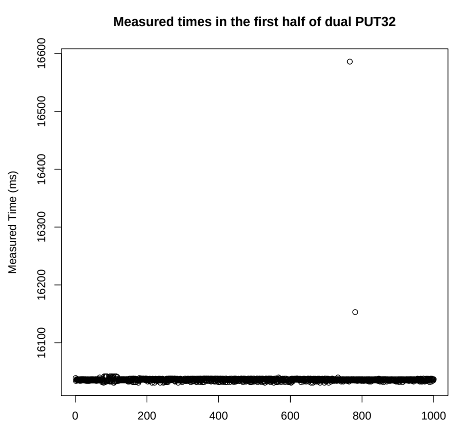
(b) 2nd half's measured times on dual PUT8



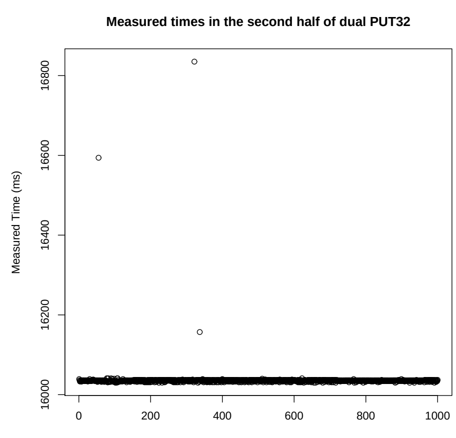
(c) 1st half's measured times on dual PUT16



(d) 2nd half's measured times on dual PUT16

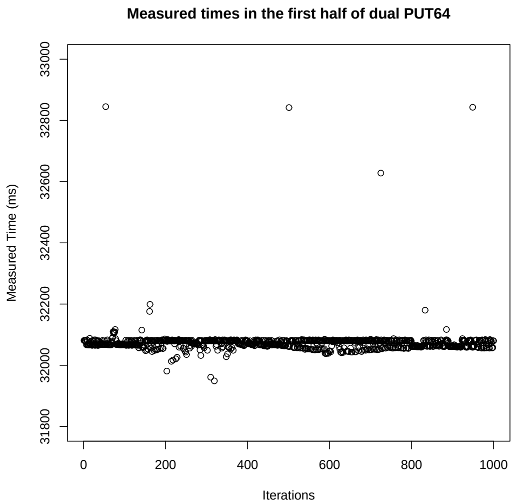


(e) 1st half's measured times on dual PUT32

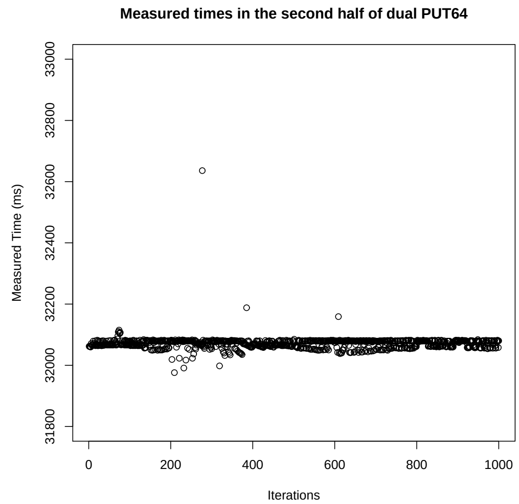


(f) 2nd half's measured times on dual PUT32

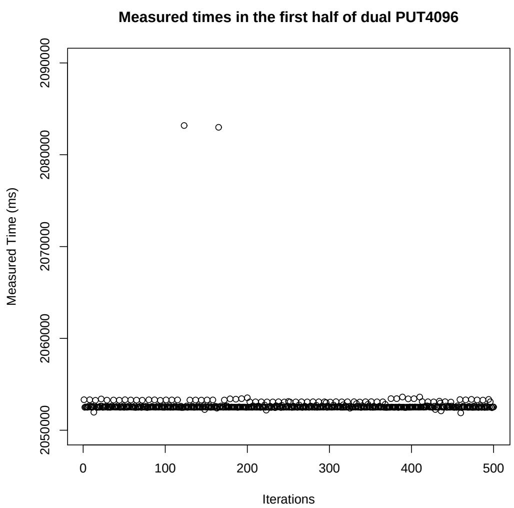
Figure 24: Measured time comparison on dual PUT8∼PUT16



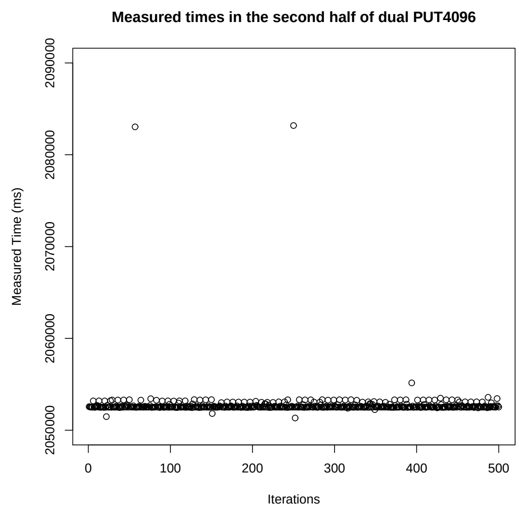
(a) 1st half's measured times on dual PUT64



(b) 2nd half's measured times on dual PUT64



(c) 1st half's measured times on dual PUT4096

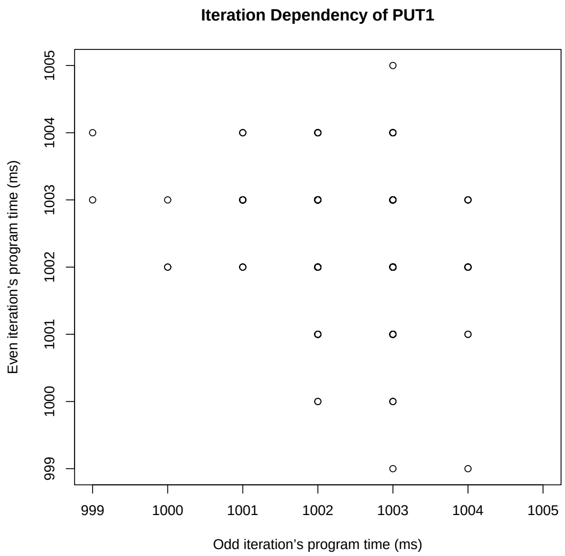


(d) 2nd half's measured times on dual PUT4096

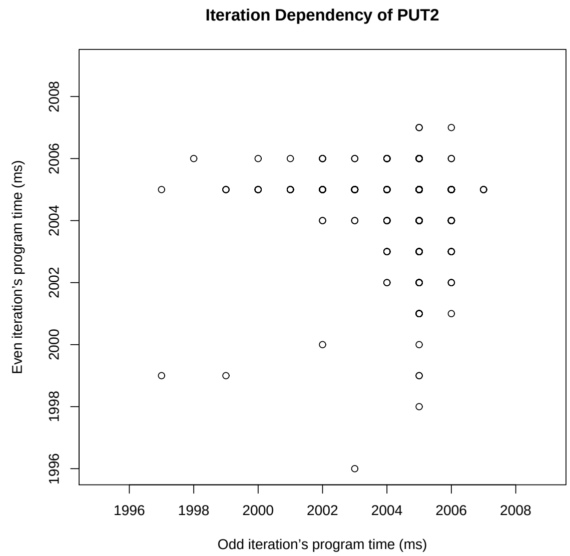
Figure 25: Measured time comparison on dual PUT64 and PUT4096

6.3 Successive Iterations' Dependency

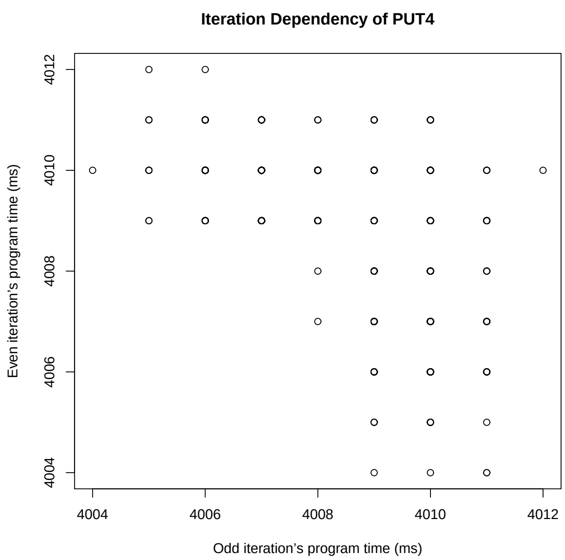
In this section we plot measured times of each iteration pair consisting of odd and even iterations. Specifically, the measured times at adjacent, odd and even iterations consist of x and y coordinates and plotted.



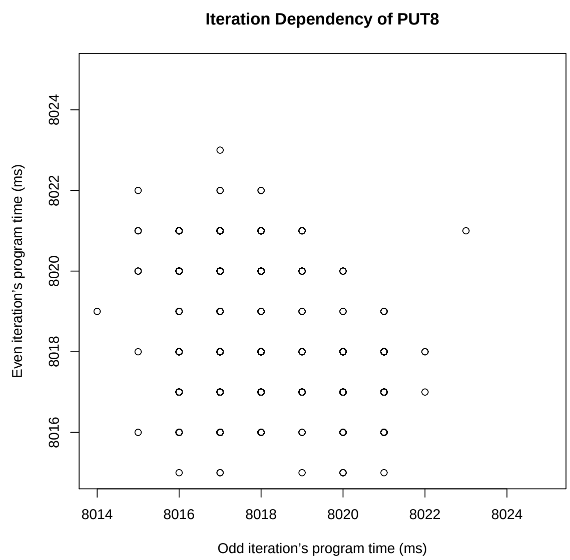
(a) Scatter plot on PUT1



(b) Scatter plot on PUT2

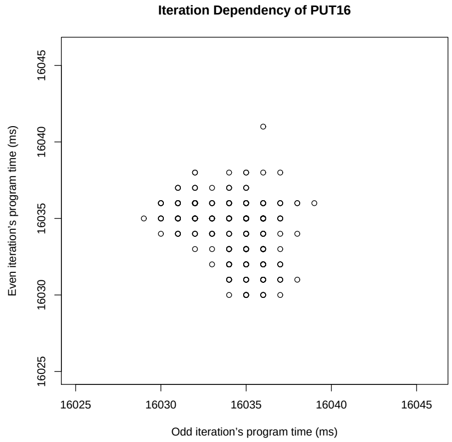


(c) Scatter plot on PUT4

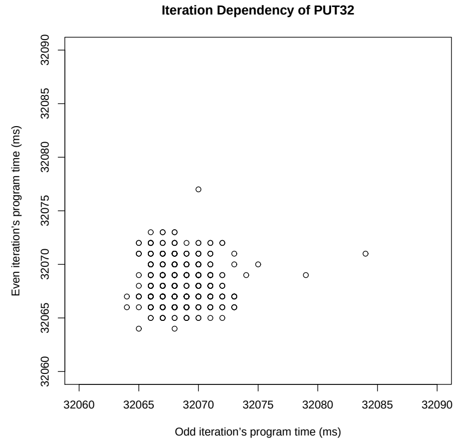


(d) Scatter plot on PUT8

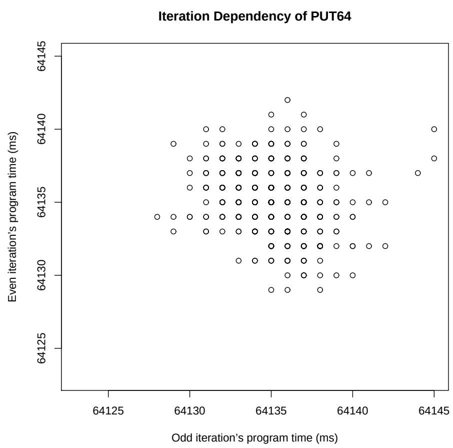
Figure 26: Iteration dependency on PUT1∼PUT8



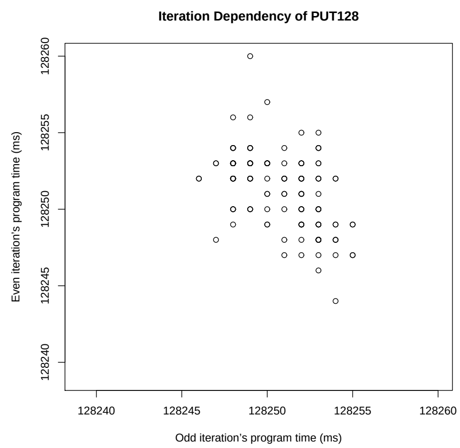
(a) Scatter plot on PUT16



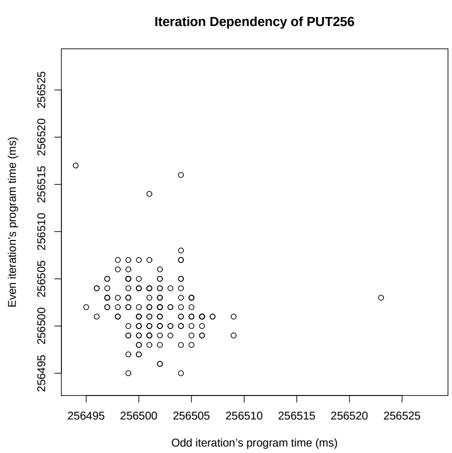
(b) Scatter plot on PUT32



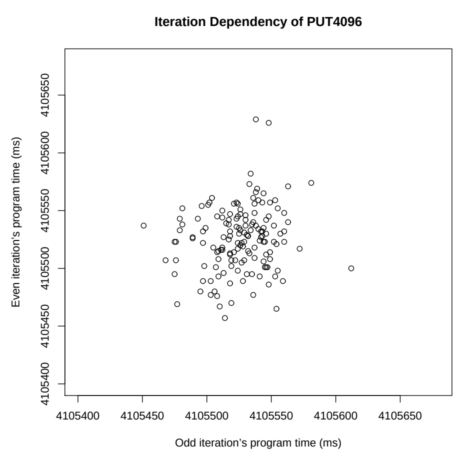
(c) Scatter plot on PUT64



(d) Scatter plot on PUT128

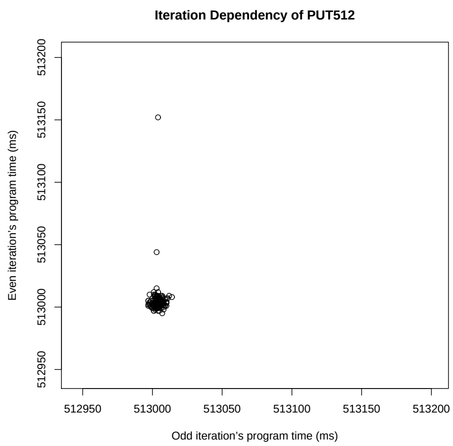


(e) Scatter plot on PUT256

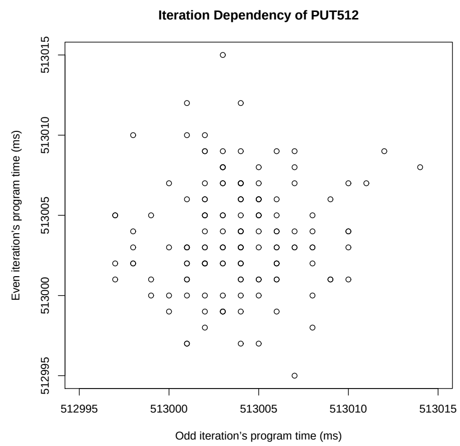


(f) Scatter plot on PUT4096

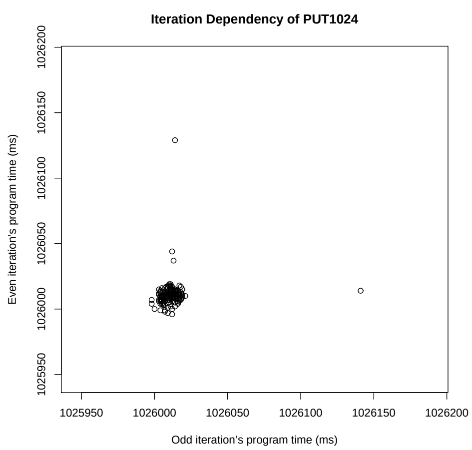
Figure 27: Iteration Dependency on PUT16∼PUT256 and PUT4096



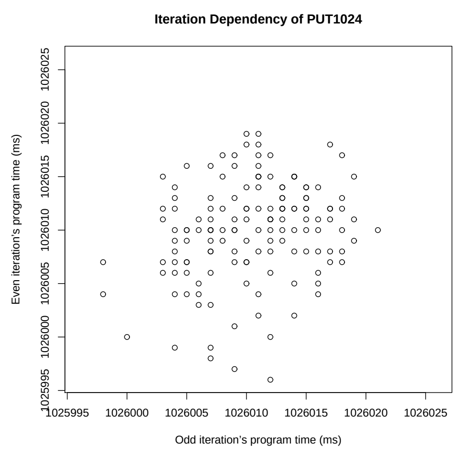
(a) Scatter plot on PUT512



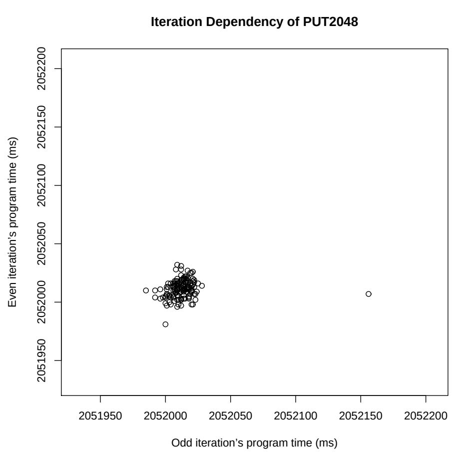
(b) Scatter plot on trimmed data of PUT512



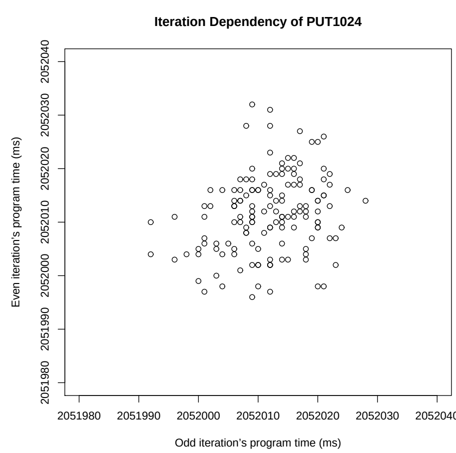
(c) Scatter plot on PUT1024



(d) Scatter plot on trimmed data of PUT1024

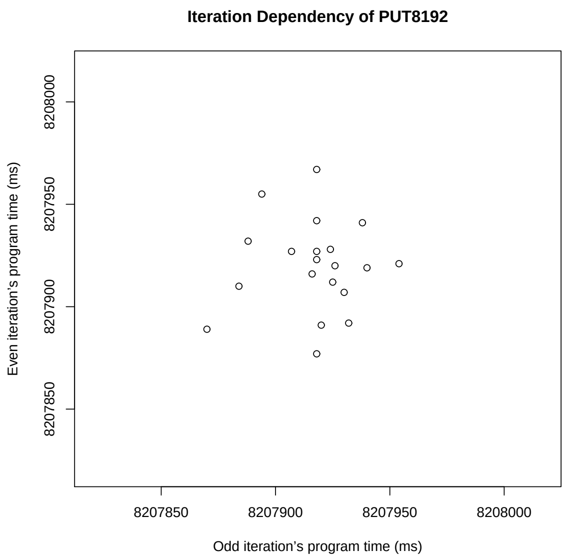


(e) Scatter plot on PUT2048

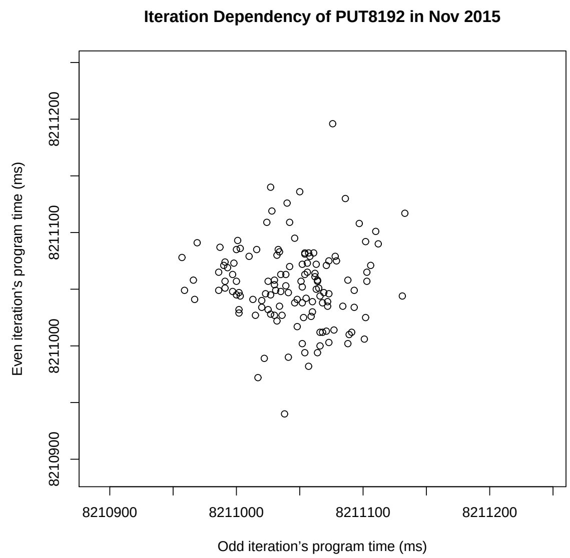


(f) Scatter plot on trimmed data of PUT2048

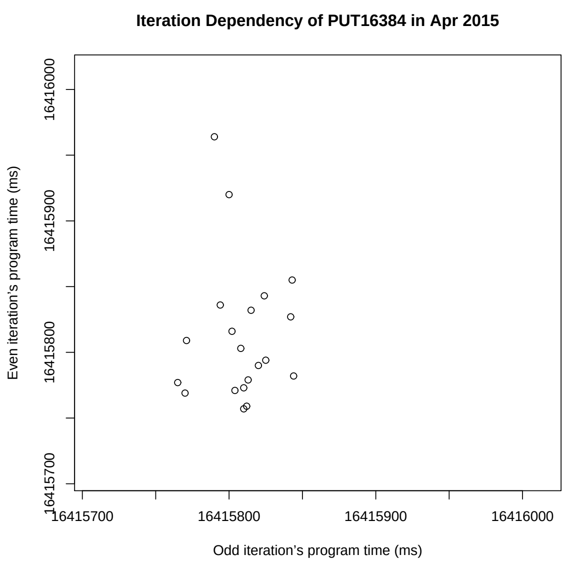
Figure 28: Iteration dependency on PUT512, PUT1024, and PUT2048



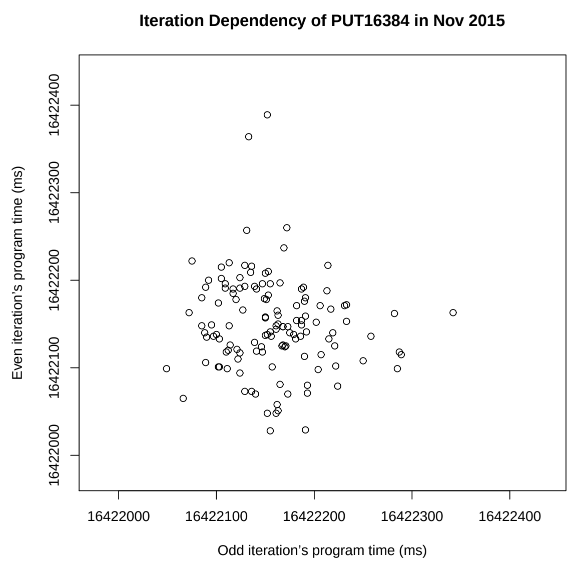
(a) Scatter plot on PUT8192 in Apr 2015



(b) Scatter plot on PUT8192 in Nov 2015



(c) Scatter plot on PUT16384 in Apr 2015



(d) Scatter plot on PUT16384 in Nov 2015

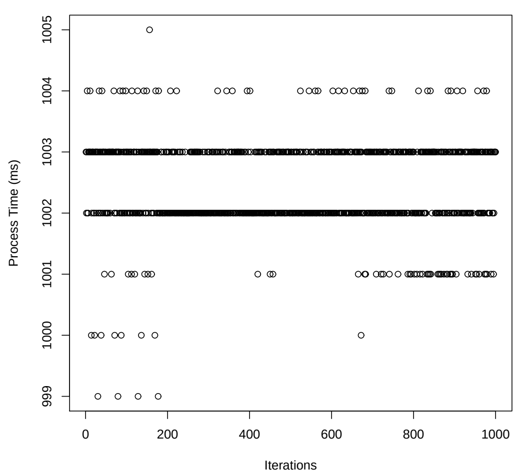
Figure 29: Iteration dependency on PUT8192∼PUT16384

7 Influence of Daemon Process on Program Time Measurement

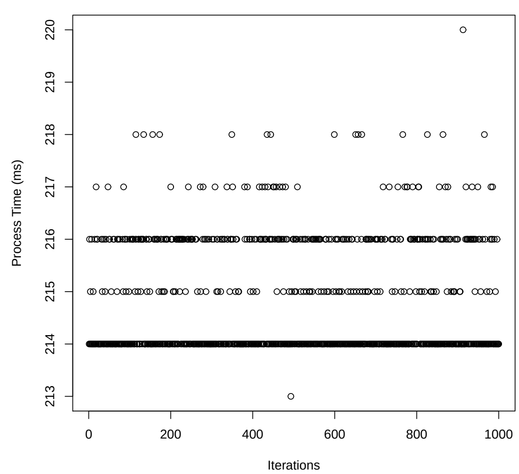
In this section we investigate correlations of program times between PUT and a group of daemon processes. The base data are from Table 1. It seems that the longer PUT, the stronger correlation of its PT with that of daemon processes.

PUT	Correlation Coefficient by EMPv4	Correlation Coefficient by EMPv5-relaxed
PUT1	-0.2	-0.2
PUT2	-0.005	-0.009
PUT4	-0.05	-0.05
PUT8	0.1	0.1
PUT16	0.1	0.1
PUT32	0.3	0.15
PUT64	0.2	0.2
PUT128	0.2	0.2
PUT256	0.4	0.4
PUT512	0.9	0.6
PUT1024	0.9	0.2
PUT2048	0.8	0.24
PUT4096	0.4	0.4
PUT8192 in Apr	0.4	0.4
PUT8192 in Nov	0.3	0.3
PUT16384 in Apr	0.4	0.8
PUT16384 in Nov	0.5	0.5

Table 8: Correlation Coefficients between Program Times of Daemon and PUT by EMPv4 and EMPv5

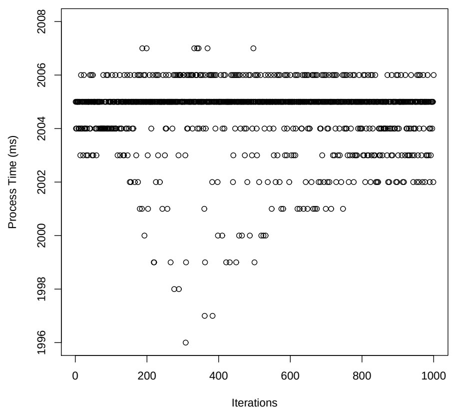


(a) PUT1's PTs

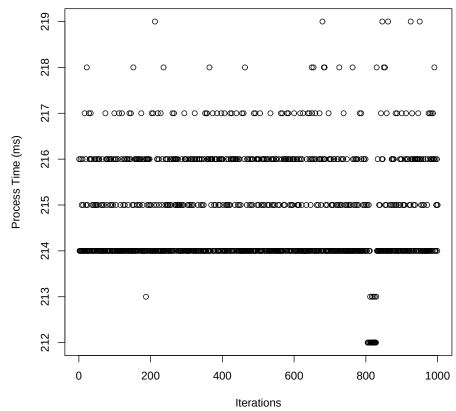


(b) PUT1's daemon PTs

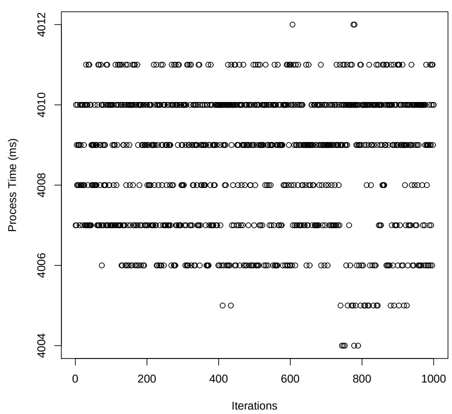
Figure 30: Program times between PUT1 vs. Daemon processes



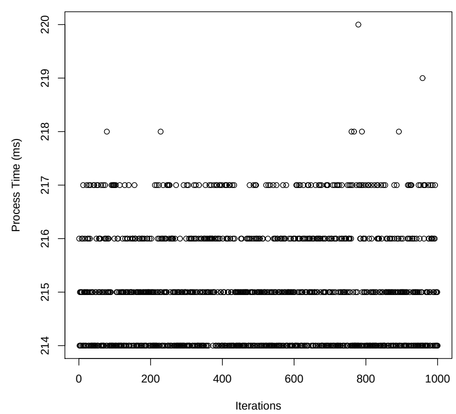
(a) PUT2's PTs



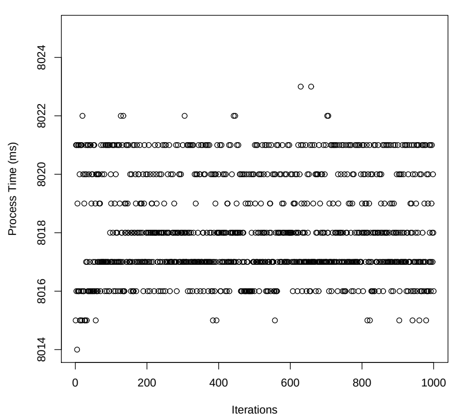
(b) PUT2's daemon PTs



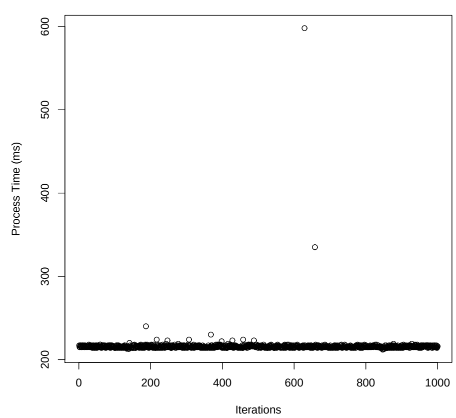
(c) PUT4's PTs



(d) PUT4's daemon PTs

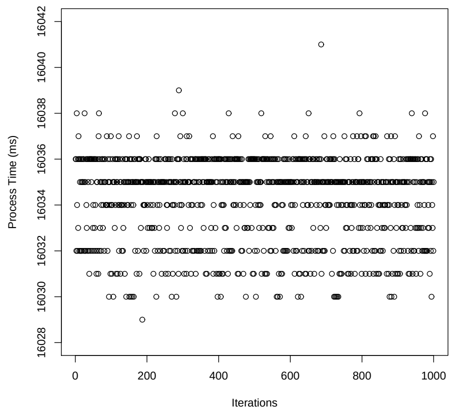


(e) PUT8's PTs

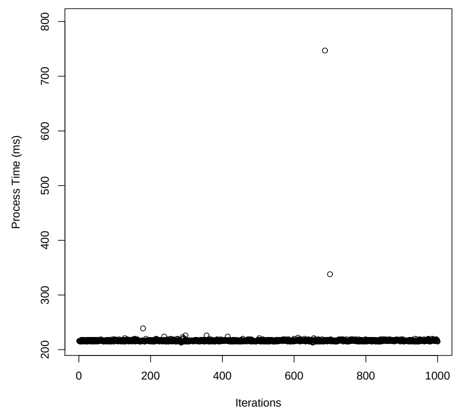


(f) PUT8's daemon PTs

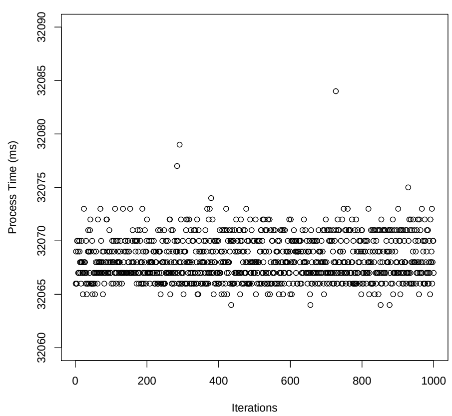
Figure 31: Program times between PUT2∼PUT8 vs. Daemon processes



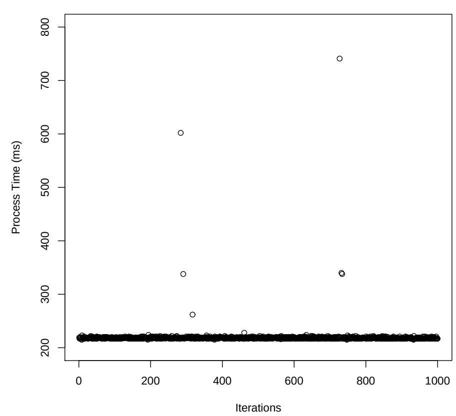
(a) PUT16's PTs



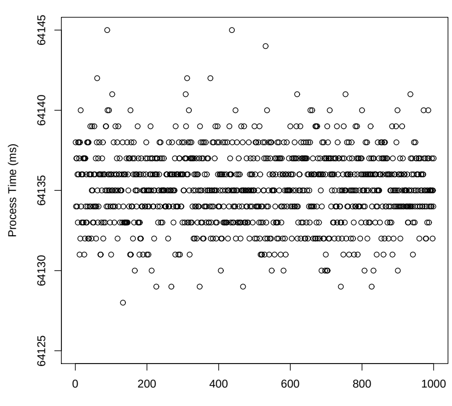
(b) PUT16's daemon PTs



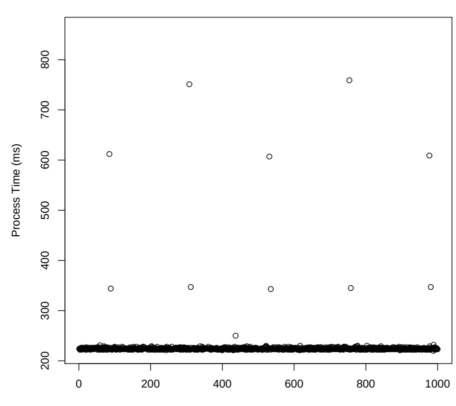
(c) PUT32's PTs



(d) PUT32's daemon PTs

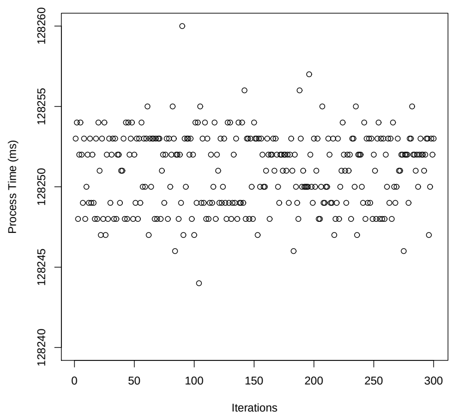


(e) PUT64's PTs

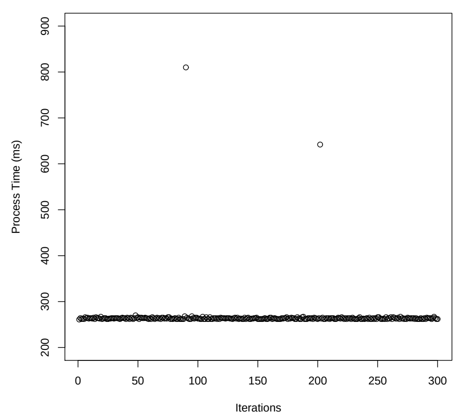


(f) PUT64's daemon PTs

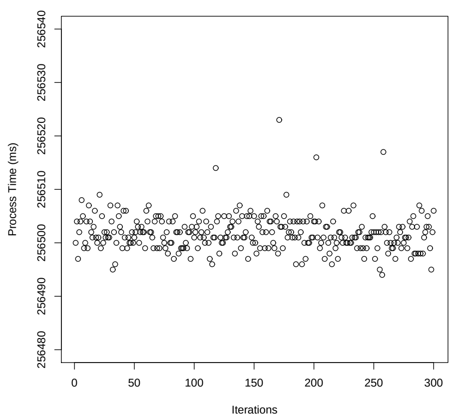
Figure 32: Program times between PUT16∼PUT64 vs. Daemon processes



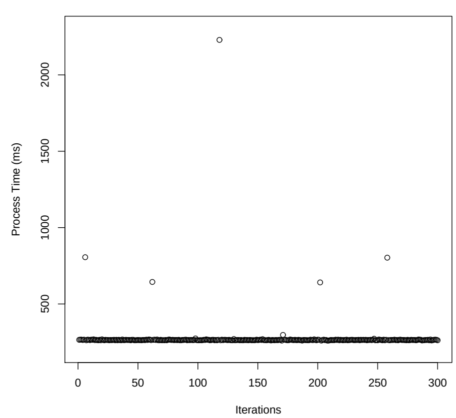
(a) PUT128's PTs



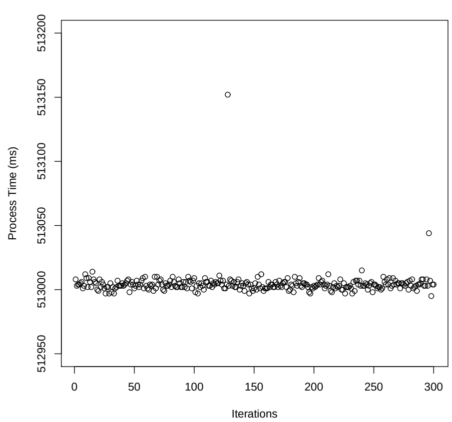
(b) PUT128's daemon PTs



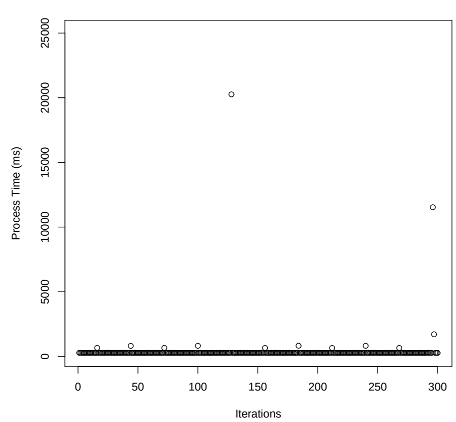
(c) PUT256's PTs



(d) PUT256's daemon PTs

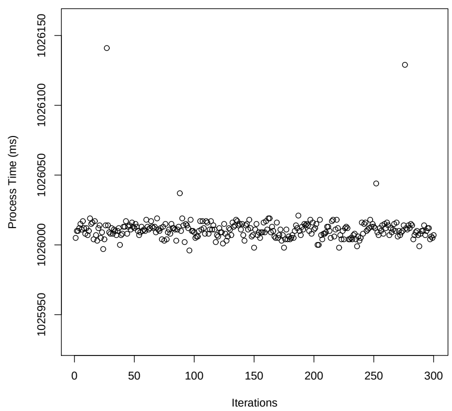


(e) PUT512's PTs

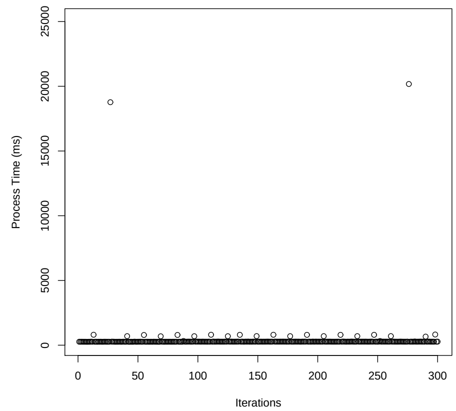


(f) PUT512's daemon PTs

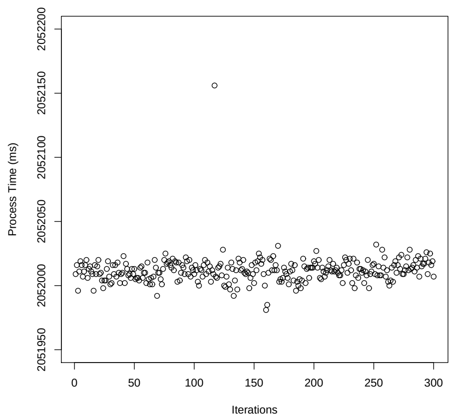
Figure 33: Program times between PUT128∼PUT512 vs. Daemon processes



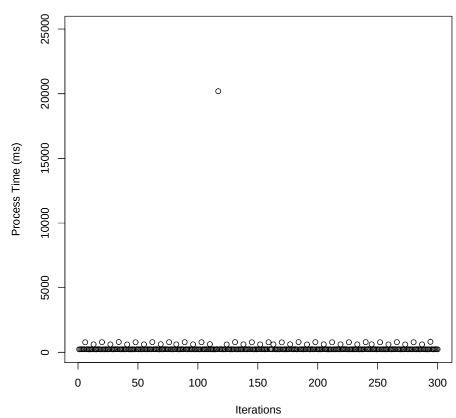
(a) PUT1024's PTs



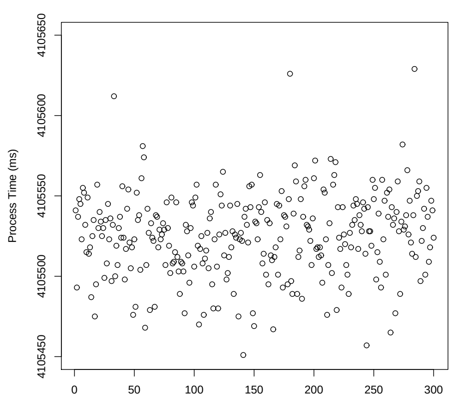
(b) PUT1024's daemon PTs



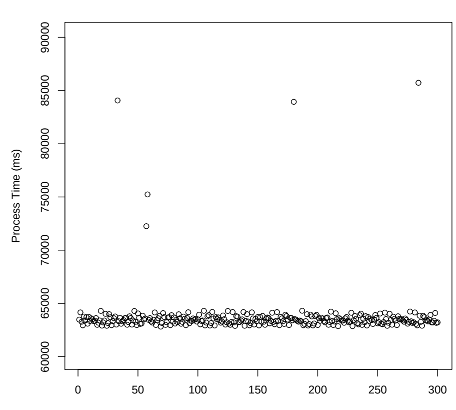
(c) PUT2048's PTs



(d) PUT2048's daemon PTs

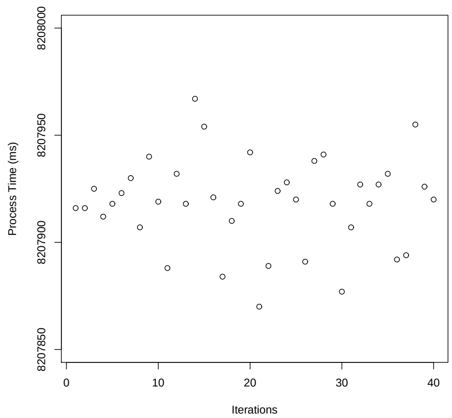


(e) PUT4096's PTs

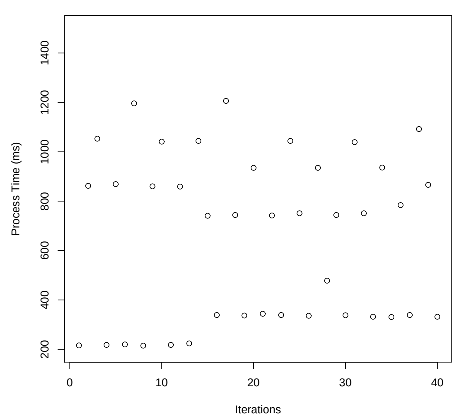


(f) PUT4096's daemon PTs

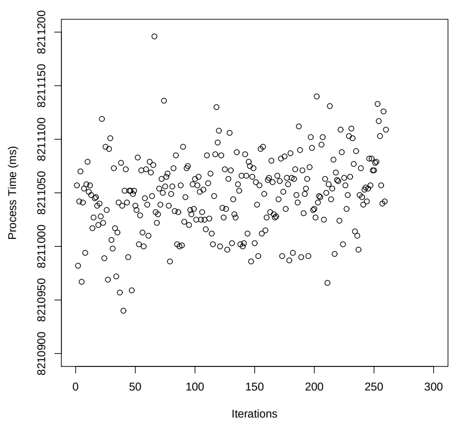
Figure 34: Program times between PUT1024∼PUT4096 vs. Daemon processes



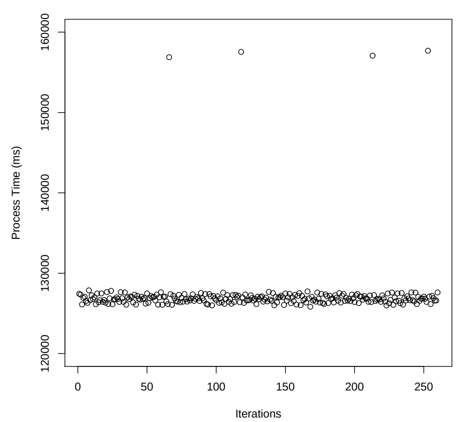
(a) PUT8192's PTs in Apr 2015



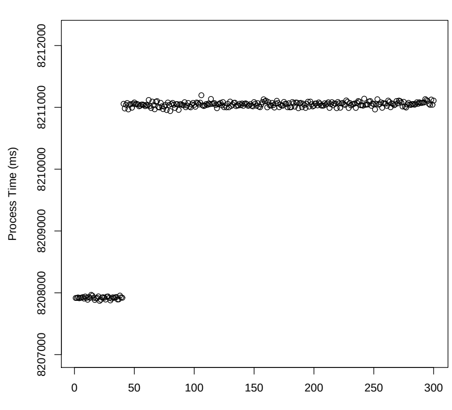
(b) PUT8192's daemon PTs in Apr 2015



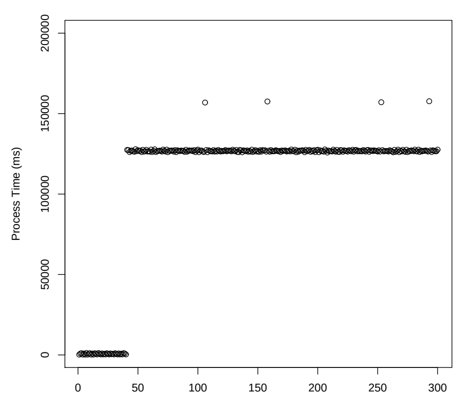
(c) PUT8192's PTs in Nov 2015



(d) PUT8192's daemon PTs in Nov 2015

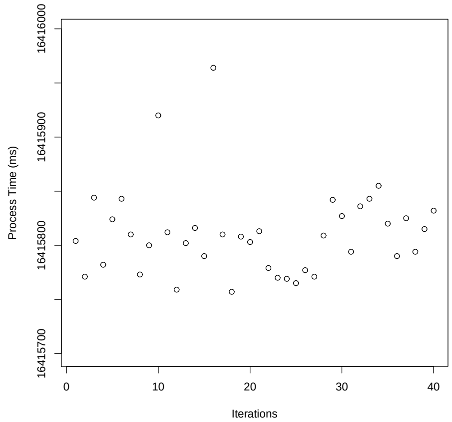


(e) Combined PUT8192's PTs

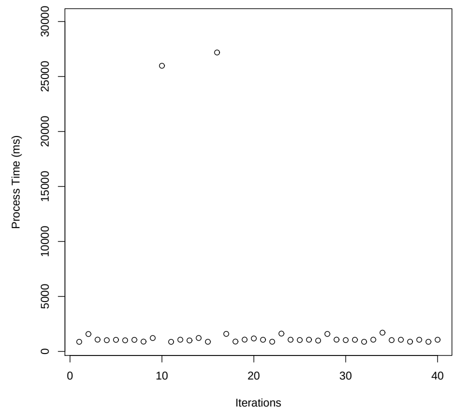


(f) Combined PUT8192's daemon PTs

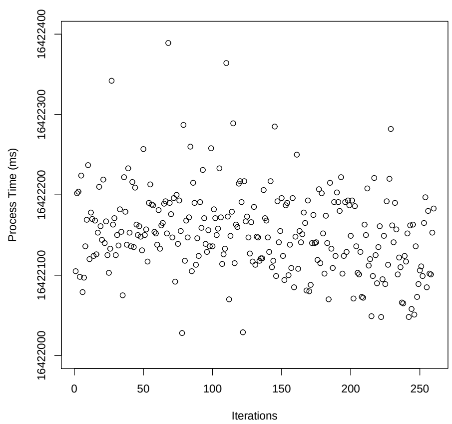
Figure 35: Program times between PUT8192 vs. Daemon processes



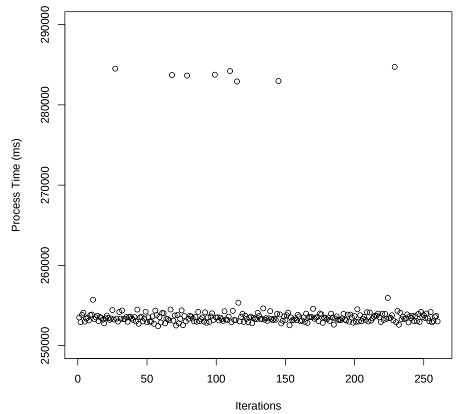
(a) PUT16384's PTs in Apr 2015



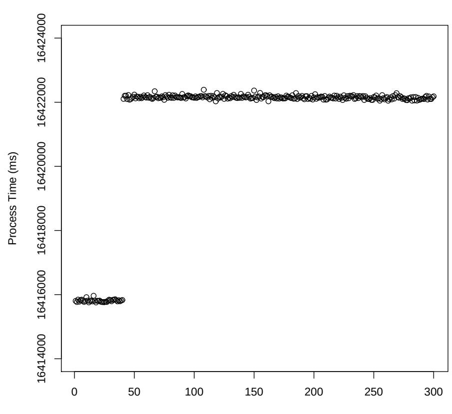
(b) PUT16384's daemon PTs in Apr 2015



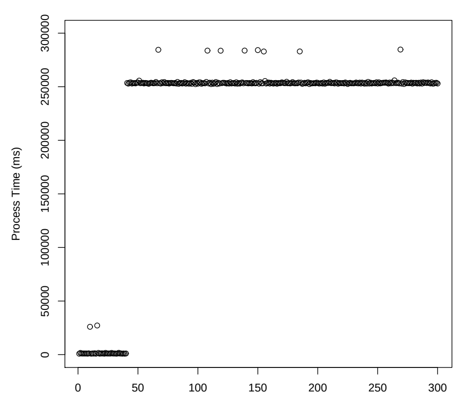
(c) PUT16384's PTs in Nov 2015



(d) PUT16384's daemon PTs in Nov 2015



(e) Combined PUT16384's PTs



(f) Combined PUT16384's daemon PTs

Figure 36: Program times between PUT16384 vs. Daemon processes

8 Conclusion

Below are the summarized observations by this study. (Order does not matter.)

1. Typically, the distribution of program times (PTs) of PUT is somewhat mixture of two models.
2. Outlier trimming does not well shape a normal distribution of PT.
3. For a short task length of PUT, it seems theres dependency between iterations in the same run of PUT.
4. Presence and activation of daemon processes strongly contribute to creating high variance in PT measurement.
5. The bigger task length, the stronger correlation between PTs of PUT and daemon processes.
6. When PUT is timed affects PT distribution, due to the presence of daemon processes whose running can't be controlled.
7. Measurement protocol is scalable with growing sample size and increasing task length.

9 Appendix

This appendix provides specific details of what daemon processes was captured and how much time was taken at a specific iteration revealing the most program time of a certain PUT.

9.1 Breakdown on Program Times of Daemon Processes

PUT256	Program Time
incr_work	256,514 msecs (at the 118th iteration)
Daemon Processes	Program Time
java	2 msecs
md0_raid1	4 msecs
jbd2/md0-8	1 msec
flush-9:0	10 msecs
proc_monitor	262 msecs
rhnsd	6 msecs
rhn_check	1,944 msecs
Total	2,229 msecs

Table 9: The daemon processes captured at the worst PT of PUT256

PUT256	Program Time
incr_work	513,152 msecs (at the 128th iteration)
Daemon Processes	Program Time
java	2 msecs
md0_raid1	51 msecs
jbd2/md0-8	27 msecs
flush-9:0	86 msecs
proc_monitor	270 msecs
rhnsd	6 msecs
rhn_check	19,820 msecs
Total	20,262 msecs

Table 10: The daemon processes captured at the worst PT of PUT512

PUT4096	Program Time
`incr_work`	4,105,629 msecs (at the 284th iteration)
Daemon Processes	Program Time
`events/0`	1 msec
`kblockd/0`	1 msec
`kslowd000`	31,710 msecs
`kslowd001`	31,782 msecs
`md0_raid1`	82 msecs
`jbd2/md0-8`	21 msecs
`flush-9:0`	79 msecs
`proc_monitor`	206 msecs
`rhnsd`	3 msecs
`ntpd`	1 msec
`java`	2 msecs
`rhn_check`	21,840 msecs
Total	85,728 msecs

Table 11: The daemon processes captured at the worst PT of PUT4096

PUT8192	Program Time
`incr_work`	8,207,884 msecs (at the 244th iteration)
Daemon Processes	Program Time
`kblockd/0`	3 msecs
`kslowd000`	31,710 msecs
`kslowd001`	31,782 msecs
`md0_raid1`	12 msecs
`jbd2/md0-8`	2 msecs
`proc_monitor`	204 msecs
`rhnsd`	6 msecs
`java`	1 msec
`rhsmcertd-worke`	114 msecs
`rhsmcertd-worke`	114 msecs
`rhn_check`	708 msecs
Total	64,656 msecs

Table 12: The daemon processes captured at the worst PT of PUT8192

Daemon Processes	Descriptions
`kslowd000` (`kslowd001`)	A kernel threads for performing things that take a relatively long time. "Typically, when processing something, these items will spend a lot of time, blocking a thread on I/O, thus making that thread unavailable for doing other work." (`http://www.mjmwired.net/kernel/Documentation/slow-work.txt`)
`rhn_check`	An external program for check for updates, run by `rhnsd`
`rhnsd`	"A background daemon process that periodically polls the Red Hat Network to see if there are any queued actions available. Typically started from the initialization (`init`) scripts in `/etc/init.d/rhnsd` when its time to poll the Red Hat Network server for available updates and actions. The default interval is every 240 minutes. The minimum polling interval is 60 minutes. Any network activity is done via the `rhn_check` utility." (`http://linuxcommand.org/man_pages/rhnsd8.html`)

Table 13: Descriptions of some daemon processes