WORM & Freerun Experiments, Random & Sequential; TTCS

Charts v2

Thursday, July 18, 2019 6:30 AM

Experiments (6x)

- Baseline Freerun Sequential ← Baseline (so w/o cipher scheduling)
- Scheduled Freerun Sequential
- Scheduled WORM Sequential ← READ ONLY RESULTS
- Baseline Freerun Random ← Baseline (so w/o cipher scheduling)
- Scheduled Freerun Random
- Scheduled WORM Random ← READ ONLY RESULTS

Freerun == baseline WORM, hence 6x tests instead of 8x!

Only used to measure forward

Standalone Ciphers

A	Chacha8
B	Chacha20
C	Freestyle Fast
D, etc.	Other ciphers

Cipher Pairs (scheduled)

X	Chacha8 × Chacha20
Y	Chacha20 × Freestyle Fast
Z	Chacha8 × Freestyle Fast

↑ primary   swap ↑

Ratios

1	1/4th swap, 3/4ths primary
2	2/4ths swap, 2/4ths primary
3	3/4ths swap, 1/4th primary

Schedules

M	Mirrored
S	Selective
F, G, H	Forward 0, 1, 2 (respectively)

Examples

A, B, C } Baseline results are simple

A_R, C_W, $Z2_R$, $FZ2_R$ } For when reads and writes appear on the same graph (shorthand iff schedule not specified)

MX_1 ← also other contextual subscripts

MX1	Mirrored schedule chacha 8 × chacha 20 w/ 1/4th storage crypted w/ swap cipher
SX2	Selective schedule chacha 8 × chacha 20 w/ 1/2 of storage crypted w/ swap cipher
SZ1	Selective schedule chacha 8 × Freestyle Fast w/ 1/4th storage crypted w/ swap cipher
FY3	Forward schedule chacha20 × Freestyle Fast w/ 3/4th storage crypted w/ swap cipher
HY3	
etc.	

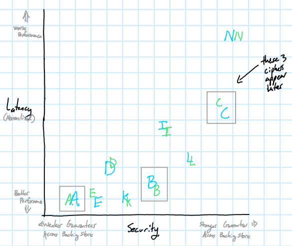
Notes

- All examples could exist for more than just 40MB results
- Normalized Y axis to max(Y) where appropriate **(GCM)**
- Expand into multiple charts w/ all combinations for dissertation
- Security of cipher pairs can be calculated w/: $S = c_1 + |c_1 - c_2|\lambda$, λ = ratio
- Y axis normalization is taking the GCM of all max(y) for each graph, multiply each graph's Y values by GCM/max(y)

Figures to include

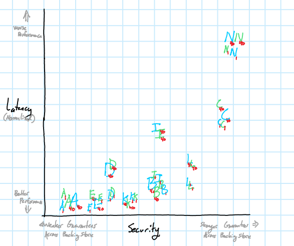
- 1.1 alt & 1.3 alt (↑ 1.2 alt if 1k is completely different)
- 2.1 alt
- 7, 3.2
- 3.3
- 4
- 6

1.1 Baseline cipher performance, 40MB reads, no scheduling



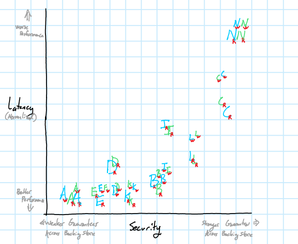
1.2 Baseline cipher performance, 40MB vs 1KB reads, no scheduling

(could separate into own charts instead)



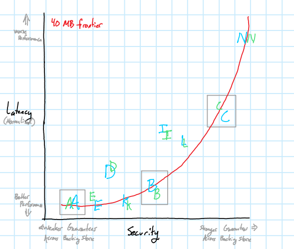
1.3 Baseline cipher performance, 40MB reads vs writes, no scheduling

(could separate into own charts instead)



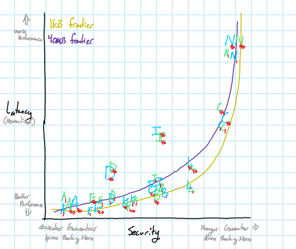
Argues that a rich tradeoff space exists between stream ciphers in a storage context

1.1-alt Baseline cipher performance, 40MB reads, no scheduling



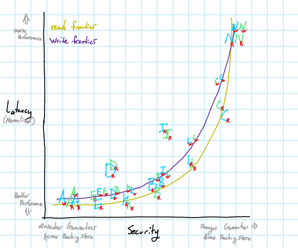
1.2-alt Baseline cipher performance, 40MB vs 1KB reads, no scheduling

(could separate into own charts instead)

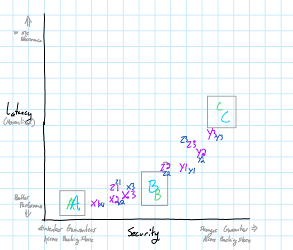


1.3-alt Baseline cipher performance, 40MB reads vs writes, no scheduling

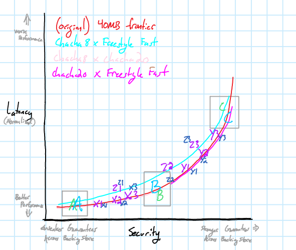
(could separate into own charts instead)



2.1 Cipher pair performance vs baseline, 40MB reads, forward 0 scheduling, all ratios

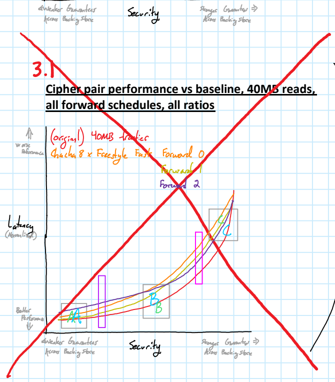


2.1-alt Cipher pair performance vs baseline, 40MB reads, forward 0 scheduling, all ratios



Demonstrates that we have a (novel) mechanism to navigate this tradeoff space

~~3.1 Cipher pair performance vs baseline, 40MB reads, all forward schedules, all ratios~~



Shows Forward scheduling potential advantages versus each other and baseline (frontier)

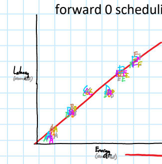
~~Valid comparison if the different workloads yield similar results?~~

Good graph for the second usecase w/ attempting to shift the storage into a "low power" mode by scheduling cipher swaps from freestyle fast to chacha 8 as the ratio shifts towards 1/4th of the backing store being freestyle

Also good for the fourth usecase w/ SSD EoL for the same reason: we can dynamically shift into a higher performance mode when the system determines it is necessary and shift back afterwards.

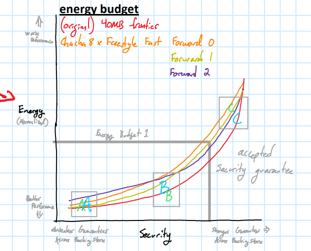
7 Linearity between latency and energy use, forward 0 scheduling, 1:1 ratios

Linearity Between Latency and Energy Use Across Ciphers on Given Workload

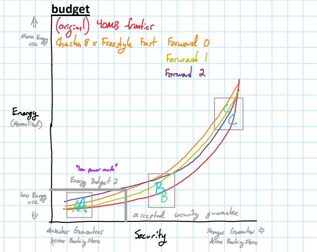


Goal: optimize security

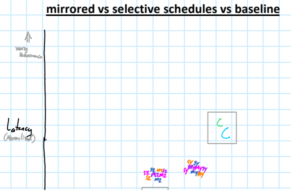
3.2-1 Cipher pair total energy use vs baseline, 40MB reads, all forward schedules, all ratios, large energy budget



3.2-2 Cipher pair total energy use vs baseline, 40MB reads, all forward schedules, all ratios, small energy budget



4 Cipher partition performance, 40MB reads, mirrored vs selective schedules vs baseline

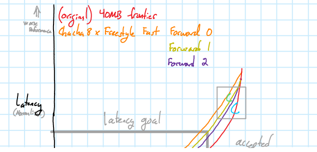


~~Also figure 11 from V1~~

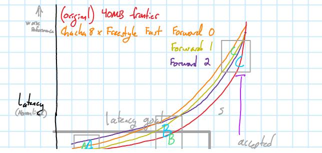
Good graph for the first and third usecases: VSRs and "panic mode". In the case of VSRs, lack of perf degradation despite presence of VSRs (up to a point). In the case of panic mode, see figure ~~11.~~6

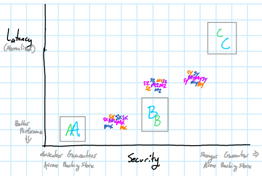
TELL selective & mirrored are parity w/ baseline (+ overhead)

3.3-1 Cipher pair performance vs baseline, 40MB reads, all forward schedules, all ratios, EOL latency goal



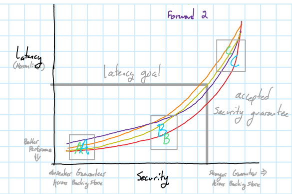
3.3-1 Cipher pair performance vs baseline at EOL, 40MB reads, all forward schedules, all ratios, EOL latency goal

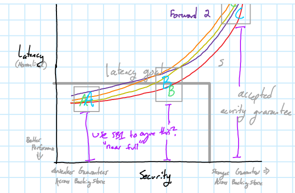




presence of VSRs (up to a point).
In the case of panic mode,
see figure ~~11.~~6

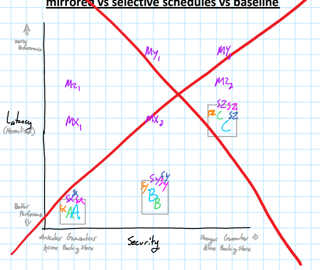
TELL selective & mirrored are parity
w/ baseline (+ overhead)





5

Cipher partition performance, 40MB writes, mirrored vs selective schedules vs baseline



Shows that mirrored is slow for writes
but as fast as baseline & Selective for
reads. Figure 6 ~~11~~ shows the benefit of
using Mirrored schedule.

Figure 6 merges:
~~- mirrored advantage~~
~~- ... frequency~~
- WORM R/W
- Freerun Sequential R/W
- Total time to schedule cipher swap of backing store (TTCS)

(2 writes turn to 4 mirrored)

40MB r/w
chacha8 x freestyle fast

1:1 ratio
all strategies

[Individual partitions in mirrored & selective must be same size as normal]

→ Does the backing store grow if in mirrored/selective?
→ If not, then tests involving them need to double the backing store size?

This should be the case for all tests if it isn't already

Ensure forward optimization only works w/ forward

6

Normalized Latency of Schedules on 40MB I/O, 1:1 ratios, all experiments