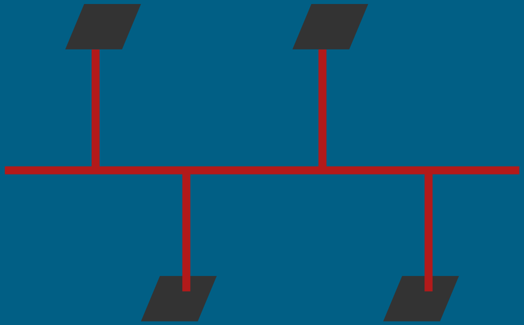
Inter-VM data exfiltration

The art of cache timing covert channel on x86 multi-core



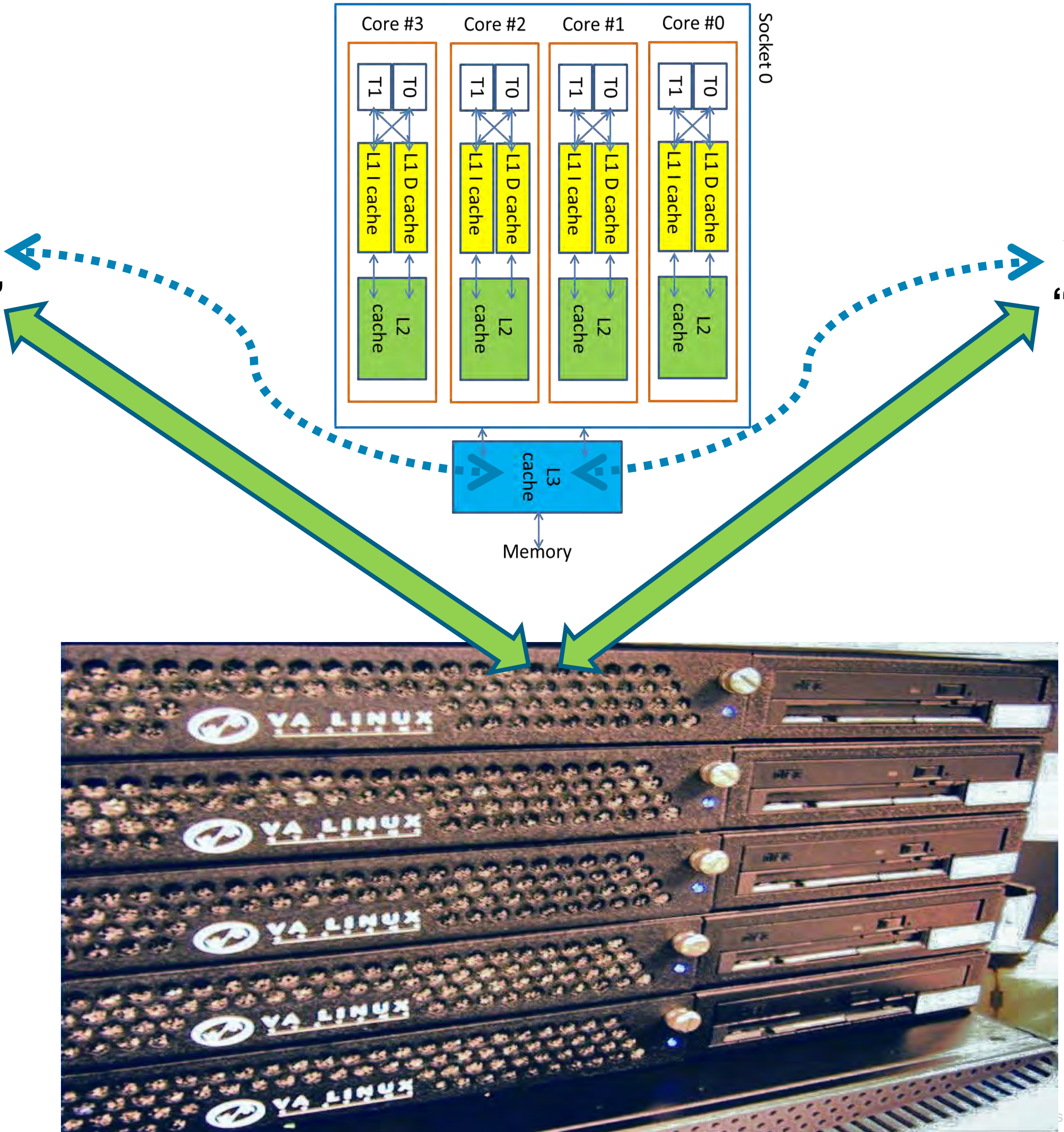
Etienne Martineau

Kernel Developer

August 2015

VM #2
"server"

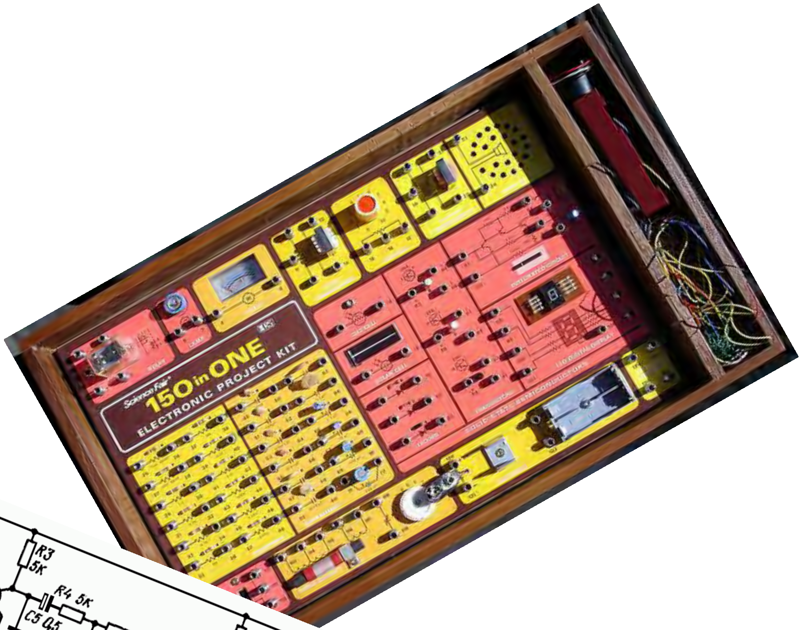
VM #1
"client"

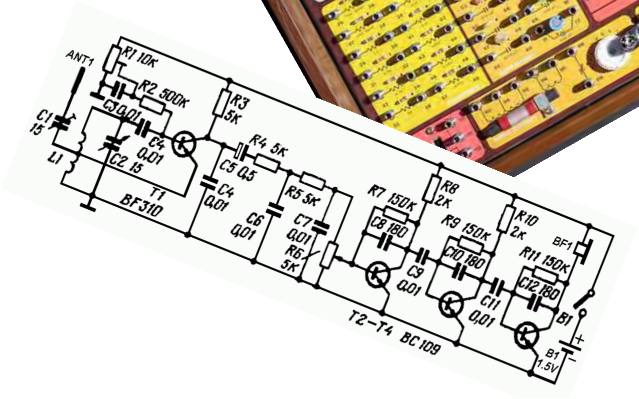


Disclaimer

- Research… own time… my opinions… not my employers…
- The information and the code provided in this presentation is to be used for educational purposes only.
- I am in no way responsible for any misuse of the information provided.
- In no way should you use the information to cause any kind of damage directly or indirectly.

About me



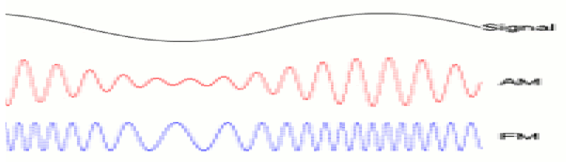


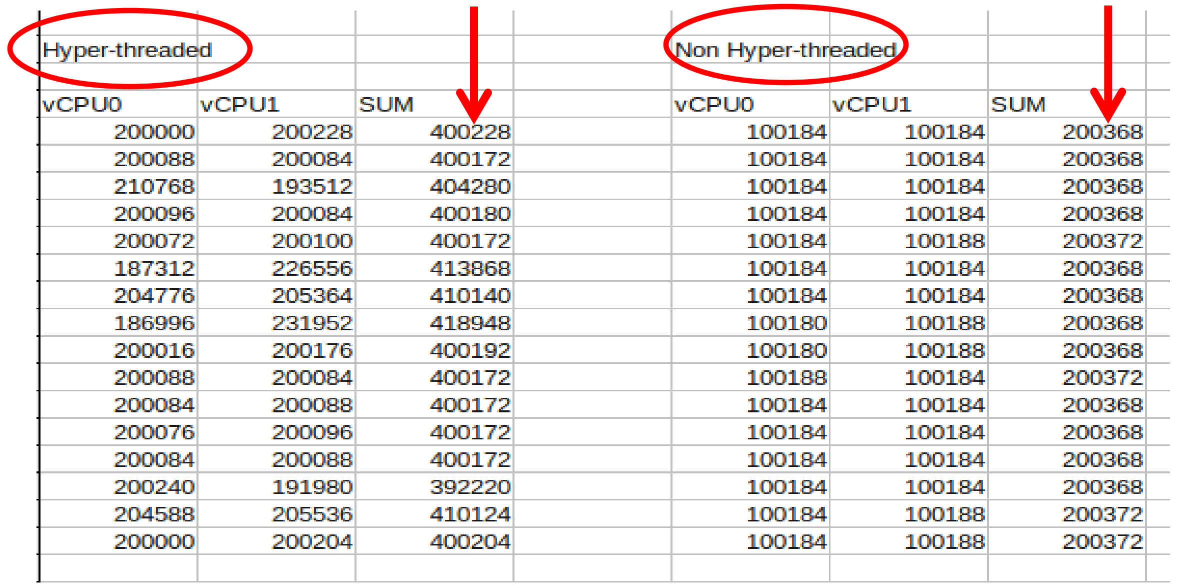
$$\nabla \cdot \vec{E} = \frac{\rho}{\varepsilon_0} = 4\pi k \rho \qquad \oint \vec{E} \cdot d\vec{A} = \frac{q}{\varepsilon_0}$$

$$\nabla \cdot \vec{B} = 0 \qquad \oint \vec{B} \cdot d\vec{A} = 0$$

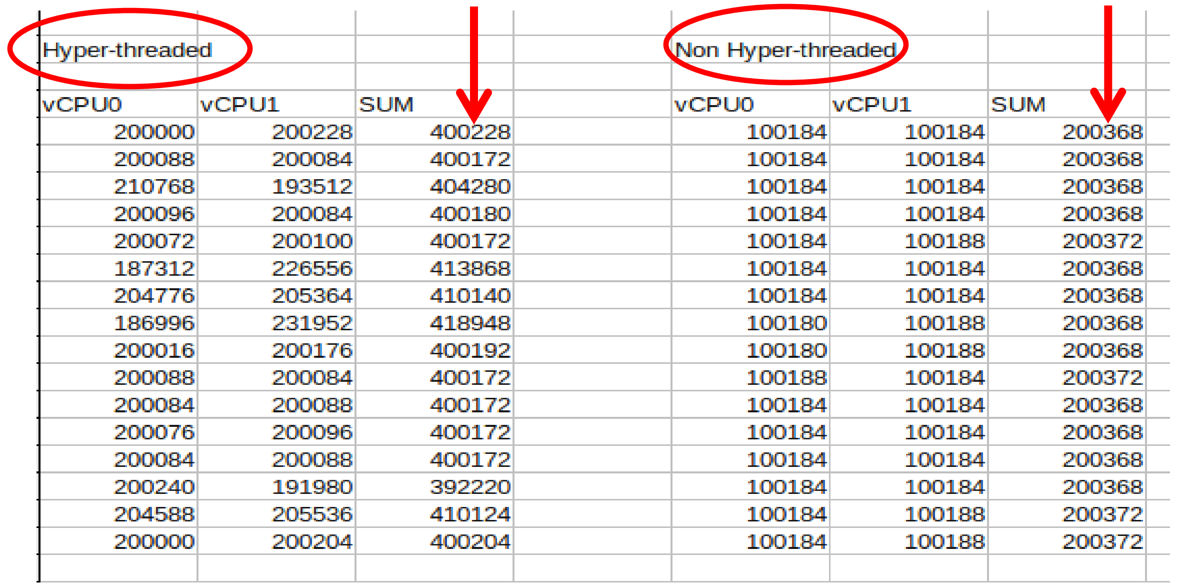
$$\nabla \times \vec{E} = -\frac{\partial B}{\partial t} \qquad \oint \vec{E} \cdot d\vec{s} = -\frac{d\Phi_B}{dt}$$

$$\nabla \times \vec{B} = \frac{\vec{J}}{\varepsilon_0 c^2} + \frac{1}{c^2}\frac{\partial E}{\partial t} \qquad \oint \vec{B} \cdot d\vec{s} = \mu_0 i + \frac{1}{c^2}\frac{\partial}{\partial t}\int \vec{E} \cdot d\vec{A}$$





Hyper-threaded				Non Hyper-threaded		
vCPU0	vCPU1	SUM		vCPU0	vCPU1	SUM
200000	200228	400228		100184	100184	200368
200088	200084	400172		100184	100184	200368
210768	193512	404280		100184	100184	200368
200096	200084	400180		100184	100184	200368
200072	200100	400172		100184	100188	200372
187312	226556	413868		100184	100184	200368
204776	205364	410140		100184	100184	200368
186996	231952	418948		100180	100188	200368
200016	200176	400192		100180	100188	200368
200088	200084	400172		100188	100184	200372
200084	200088	400172		100184	100184	200368
200076	200096	400172		100184	100184	200368
200084	200088	400172		100184	100184	200368
200240	191980	392220		100184	100184	200368
204588	205536	410124		100184	100188	200372
200000	200204	400204		100184	100188	200372

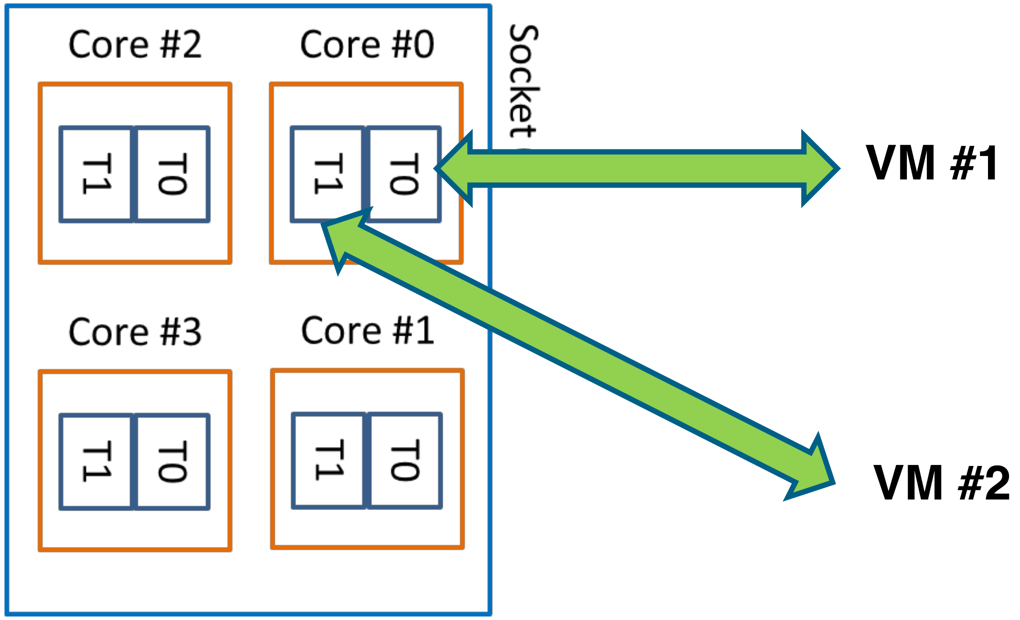


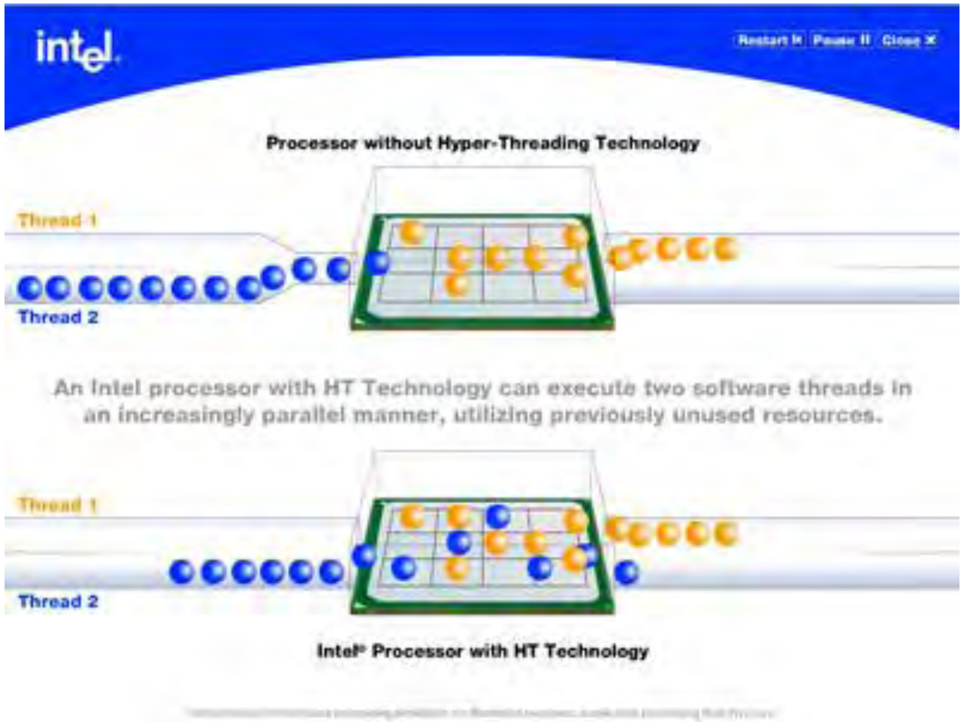
Hyper-threaded

vCPU0	vCPU1	SUM
200000	200228	400228
200088	200084	400172
210768	193512	404280
200096	200084	400180
200072	200100	400172
187312	226556	413868
204776	205364	410140
186996	231952	418948
200016	200176	400192
200088	200084	400172
200084	200088	400172
200076	200096	400172
200084	200088	400172
200240	191980	392220
204588	205536	410124
200000	200204	400204

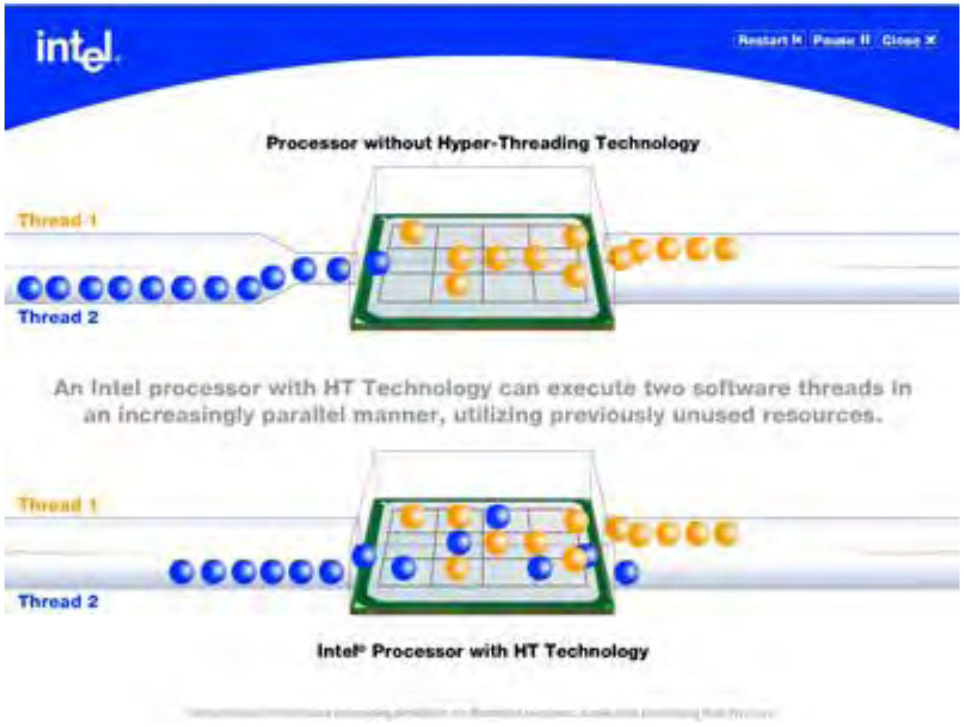
Non Hyper-threaded

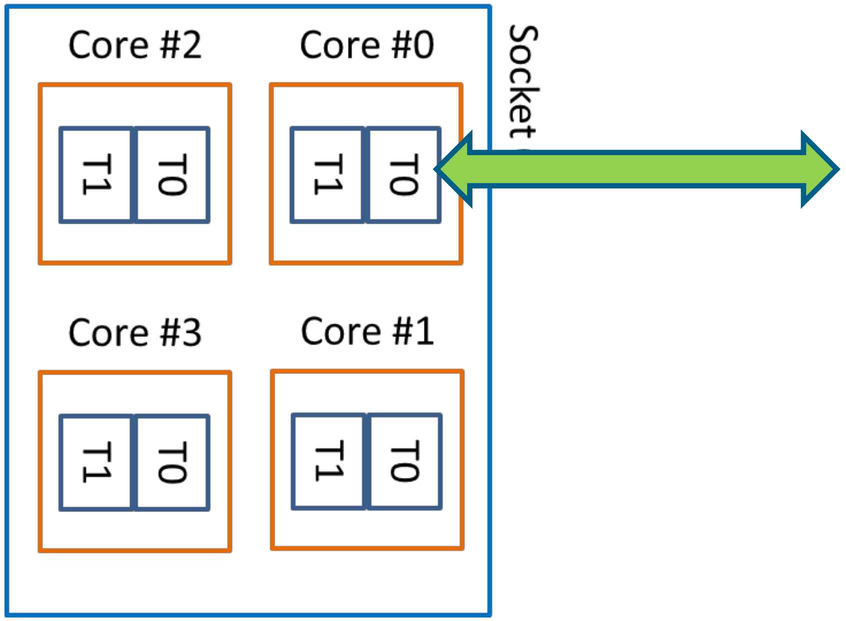
vCPU0	vCPU1	SUM
100184	100184	200368
100184	100184	200368
100184	100184	200368
100184	100184	200368
100184	100188	200372
100184	100184	200368
100184	100184	200368
100180	100188	200368
100180	100188	200368
100188	100184	200372
100184	100184	200368
100184	100184	200368
100184	100184	200368
100184	100184	200368
100184	100188	200372
100184	100188	200372




intel
Restart
Pause
Close
Processor without Hyper-Threading Technology
Thread 1
Thread 2
Thread 1
Thread 2
Intel® Processor with HT Technology

An Intel processor with HT Technology can execute two software threads in an increasingly parallel manner, utilizing previously unused resources.

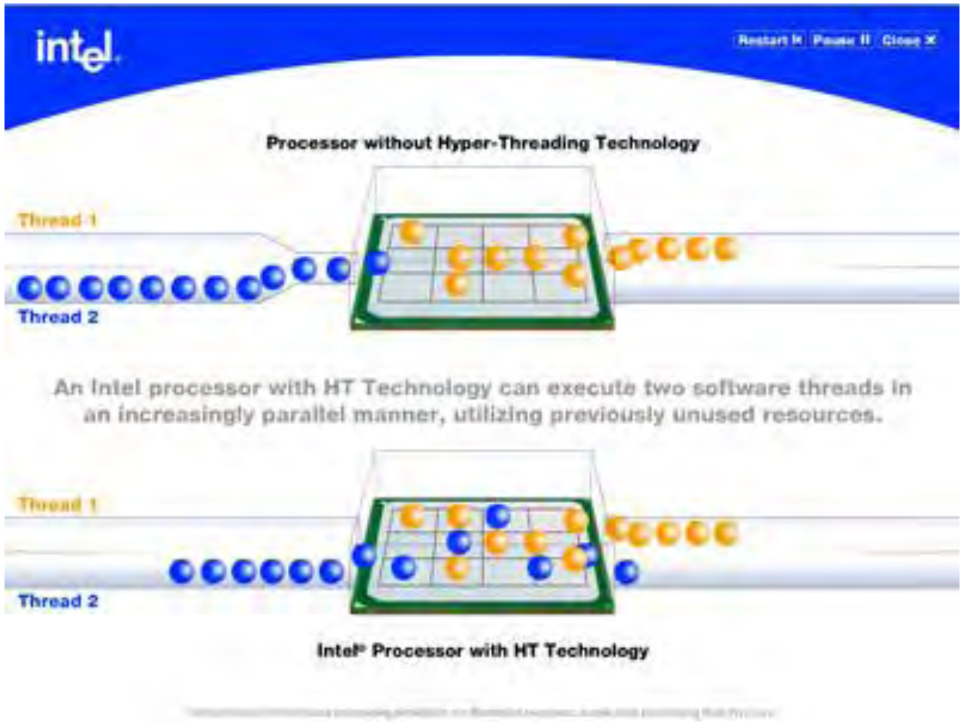

intel
Restart Pause Close
Processor without Hyper-Threading Technology
Thread 1
Thread 2
An Intel processor with HT Technology can execute two software threads in an increasingly parallel manner, utilizing previously unused resources.
Thread 1
Thread 2
Intel® Processor with HT Technology



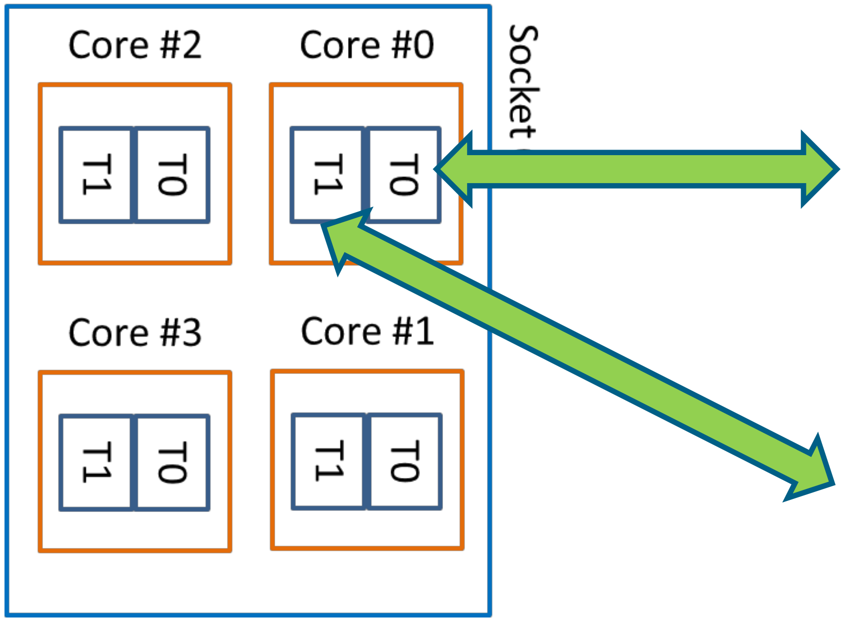
VM#1 Modulate a contention pattern

1 | 0 | 0 | 0 | 1

MUL | NOP | NOP | NOP | MUL


intel
Restart Pause Close
Processor without Hyper-Threading Technology
Thread 1
Thread 2
Thread 1
Thread 2
Intel® Processor with HT Technology

An Intel processor with HT Technology can execute two software threads in an increasingly parallel manner, utilizing previously unused resources.

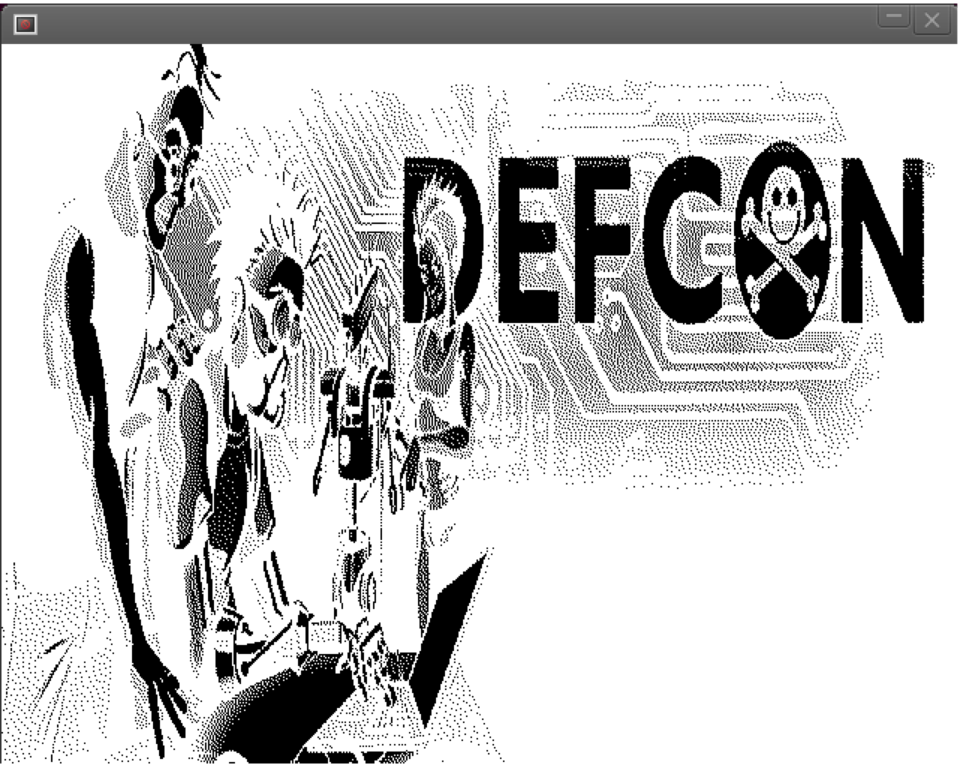


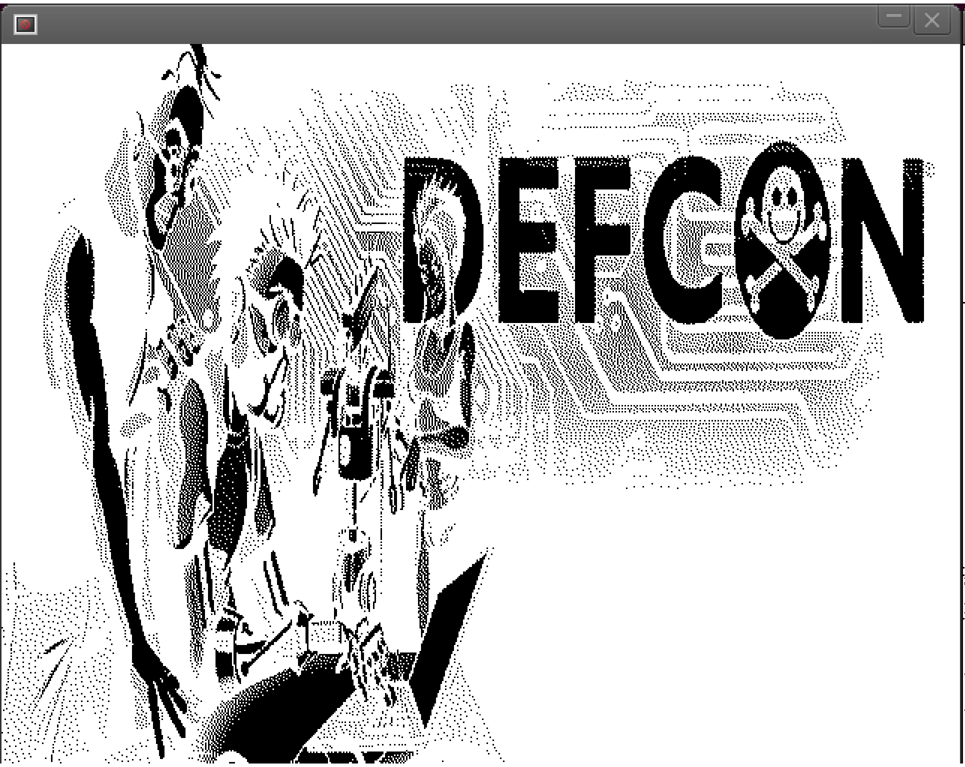
VM#1 Modulate a contention pattern

```
 1  |  0  |  0  |  0  |  1
MUL | NOP | NOP | NOP | MUL
```

VM#2 Detect BUS contention

```
Slow| Fast| Fast| Fast| Slow
 1  |  0  |  0  |  0  |  1
```


DEFCON

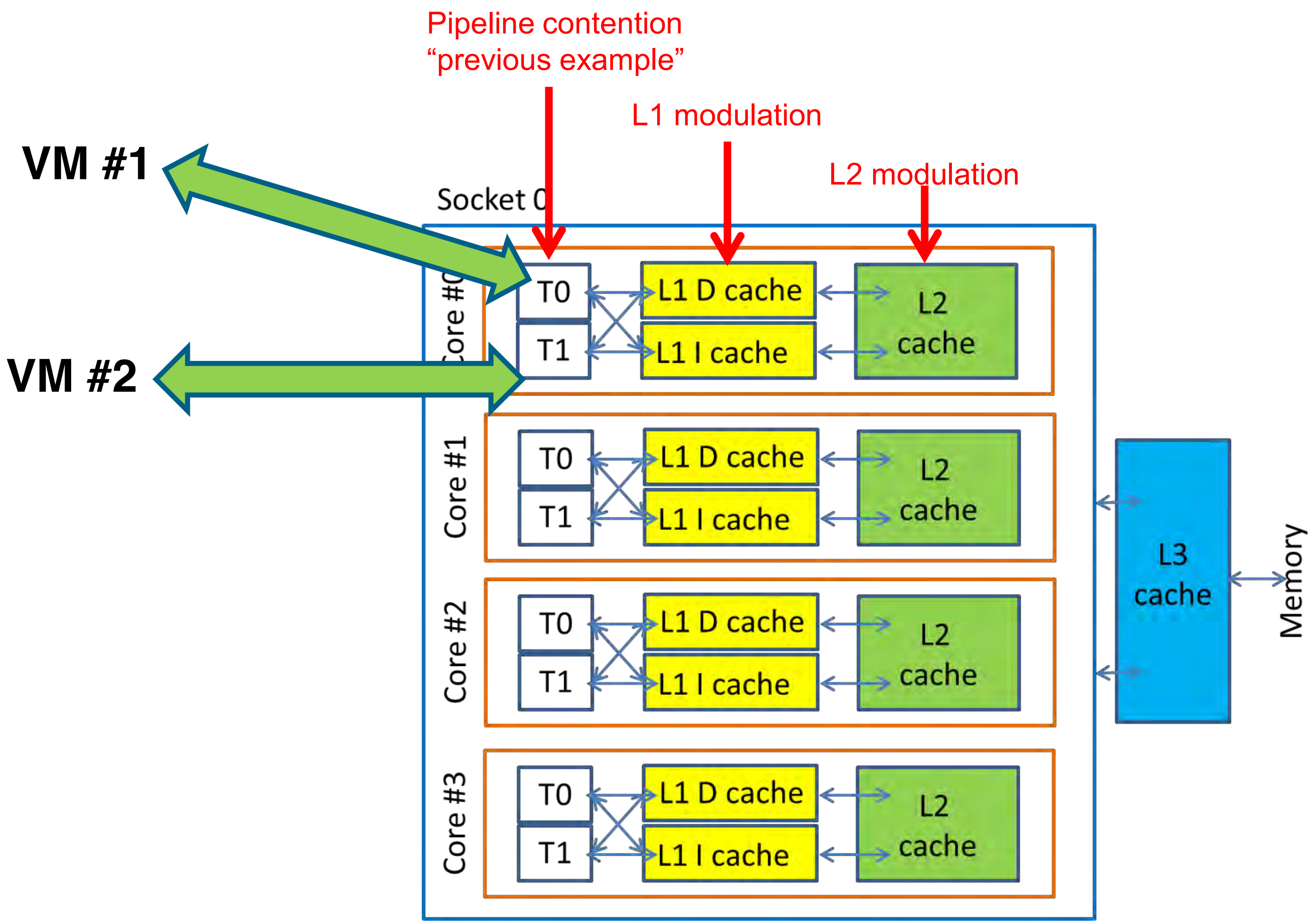

DEFCON

Video #1

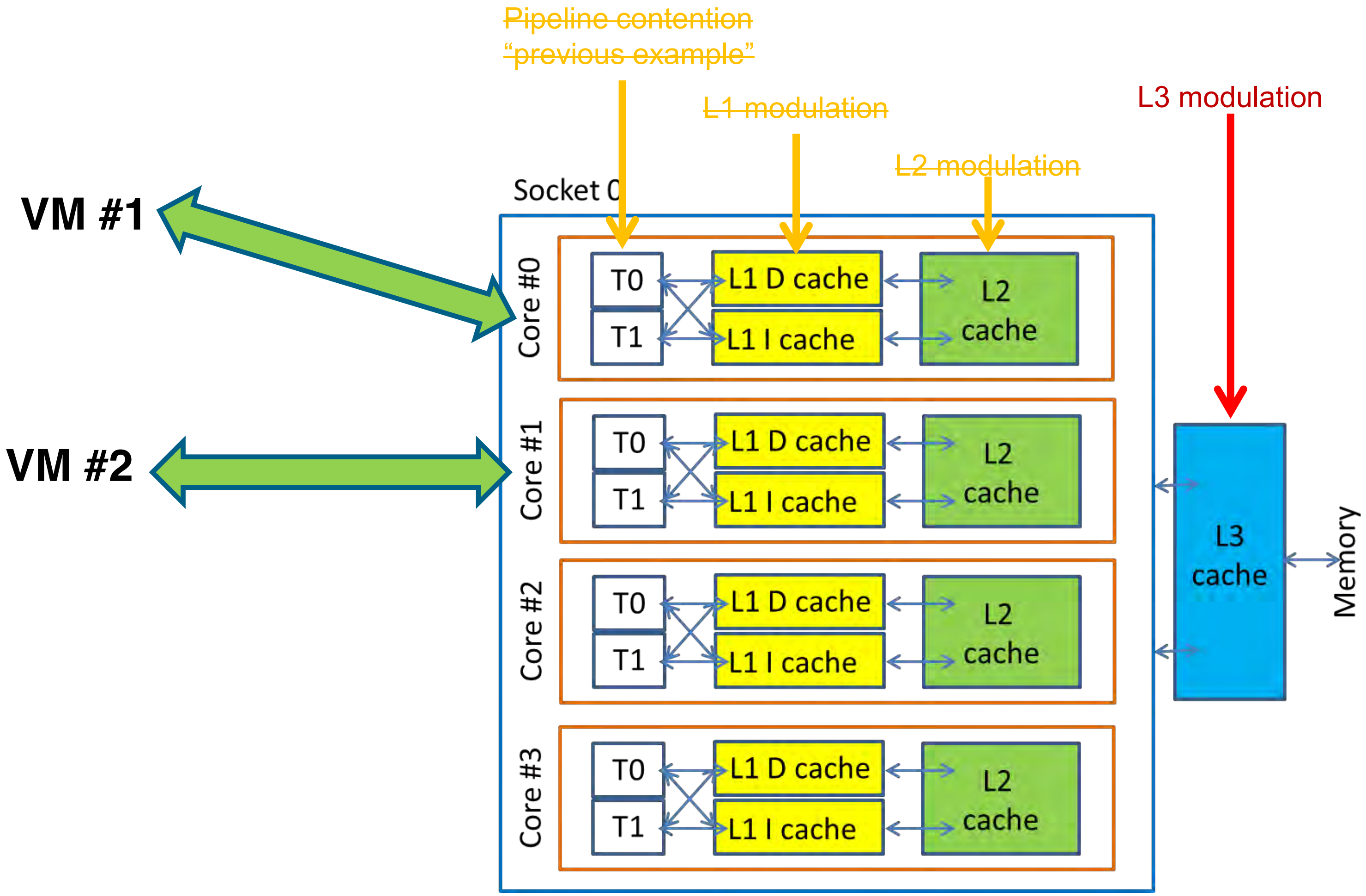
Overview

- Goal
  - Practical implementation (not just some research stuff)
- How
  - Abusing X86 shared resources
  - Cache line encoding / decoding
  - Getting around the HW pre-fetcher
  - Data persistency and noise. What can be done?
  - Guest to host page table de-obfuscation. The easy way
  - High precision inter-VM synchronization: ➔All about timers
- Detection / Mitigation

Shared resource: HT enabled

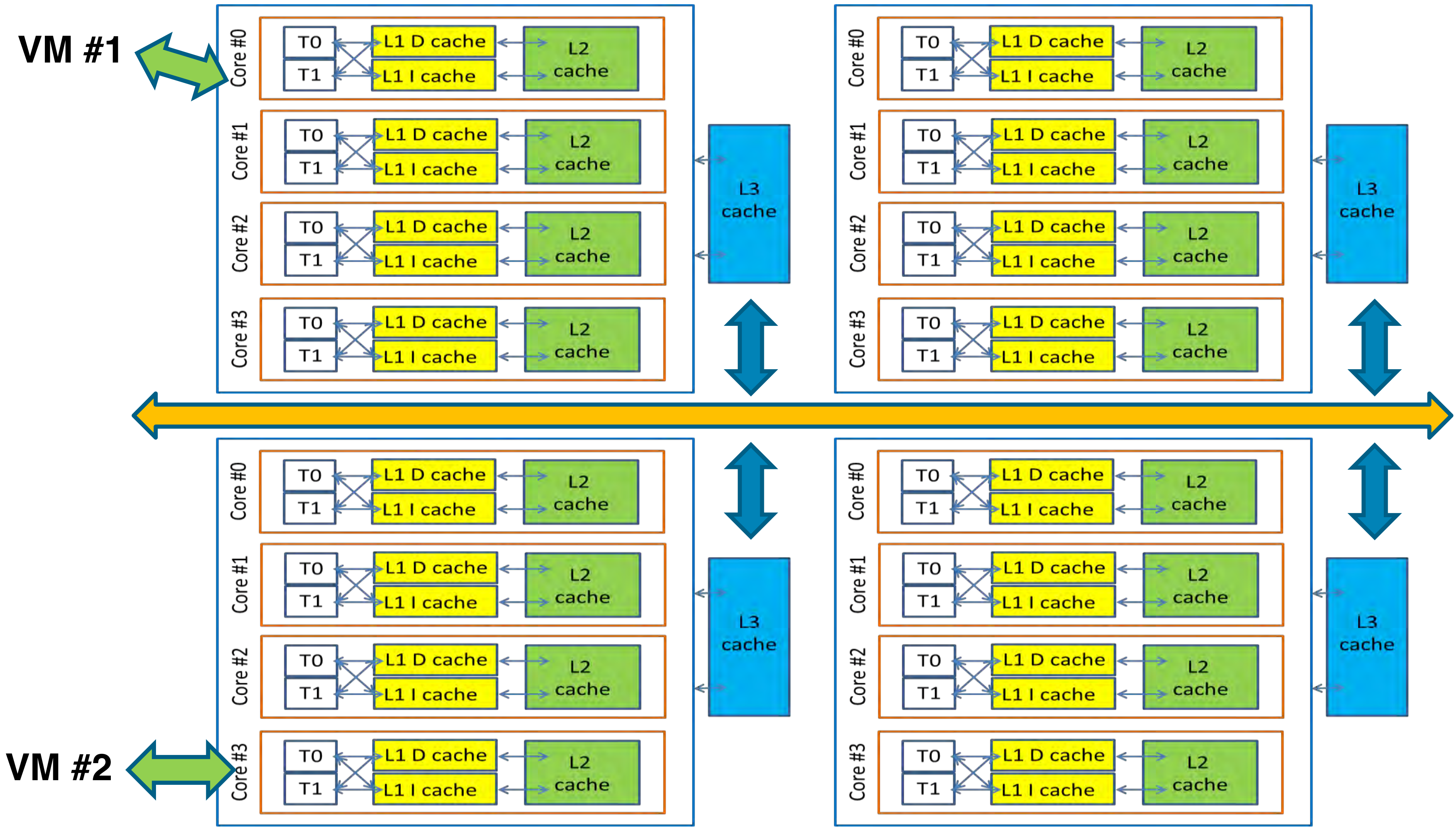


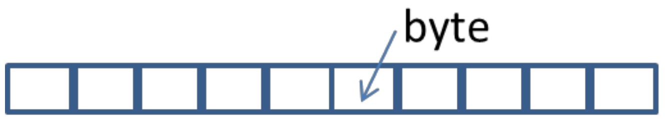
Shared resource: HT disabled

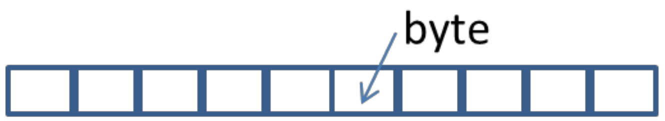


http://it.slashdot.org/story/05/05/17/201253/hyper-threading-linus-torvalds-vs-colin-perciv

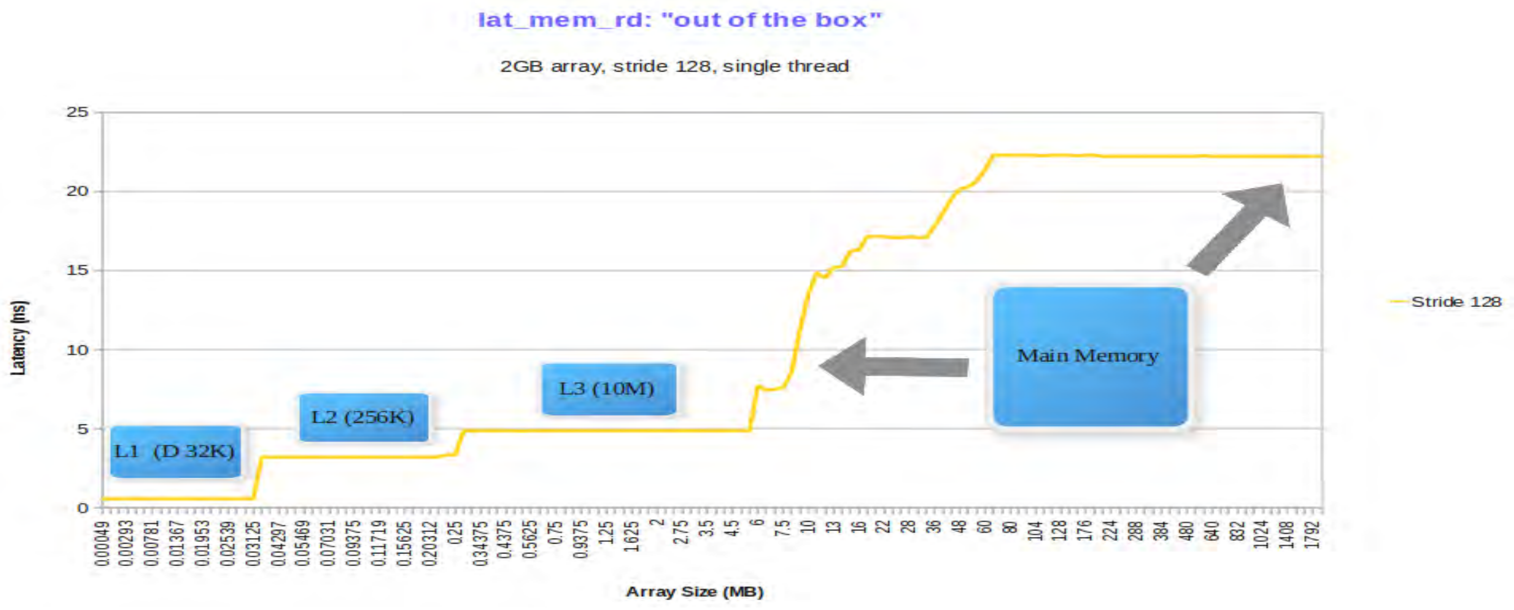
Shared resource: Multi socket

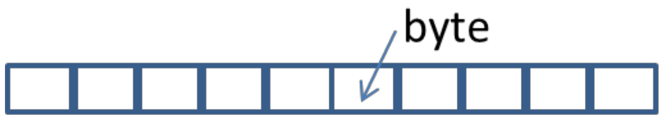



byte
Cache line (64 bytes)

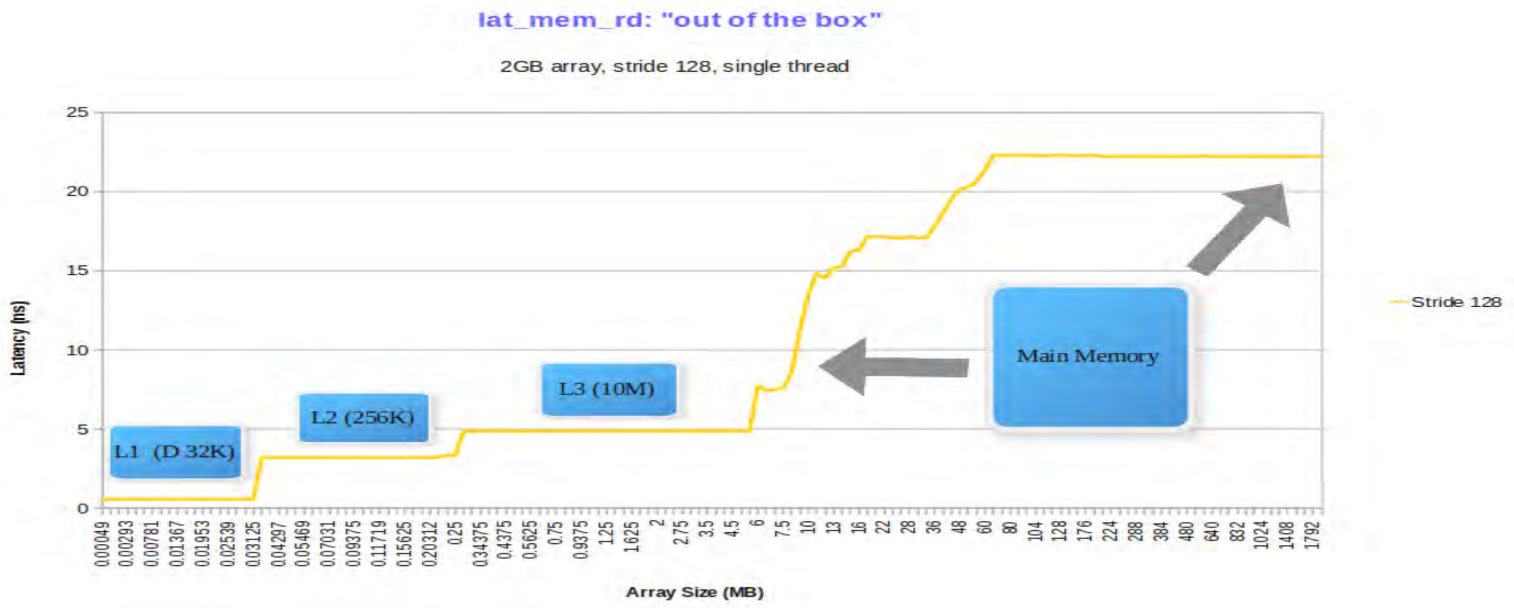


Cache line (64 bytes)

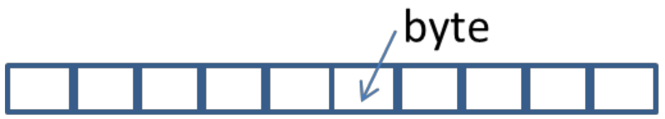




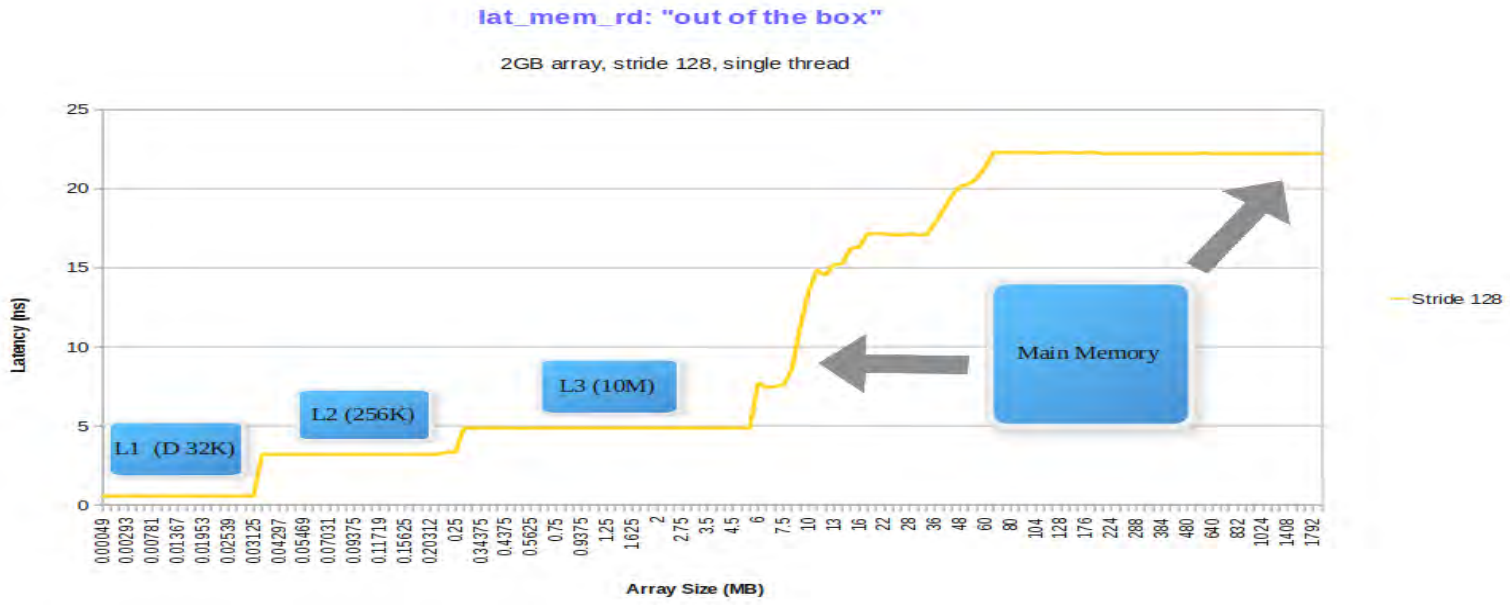
Cache line (64 bytes)



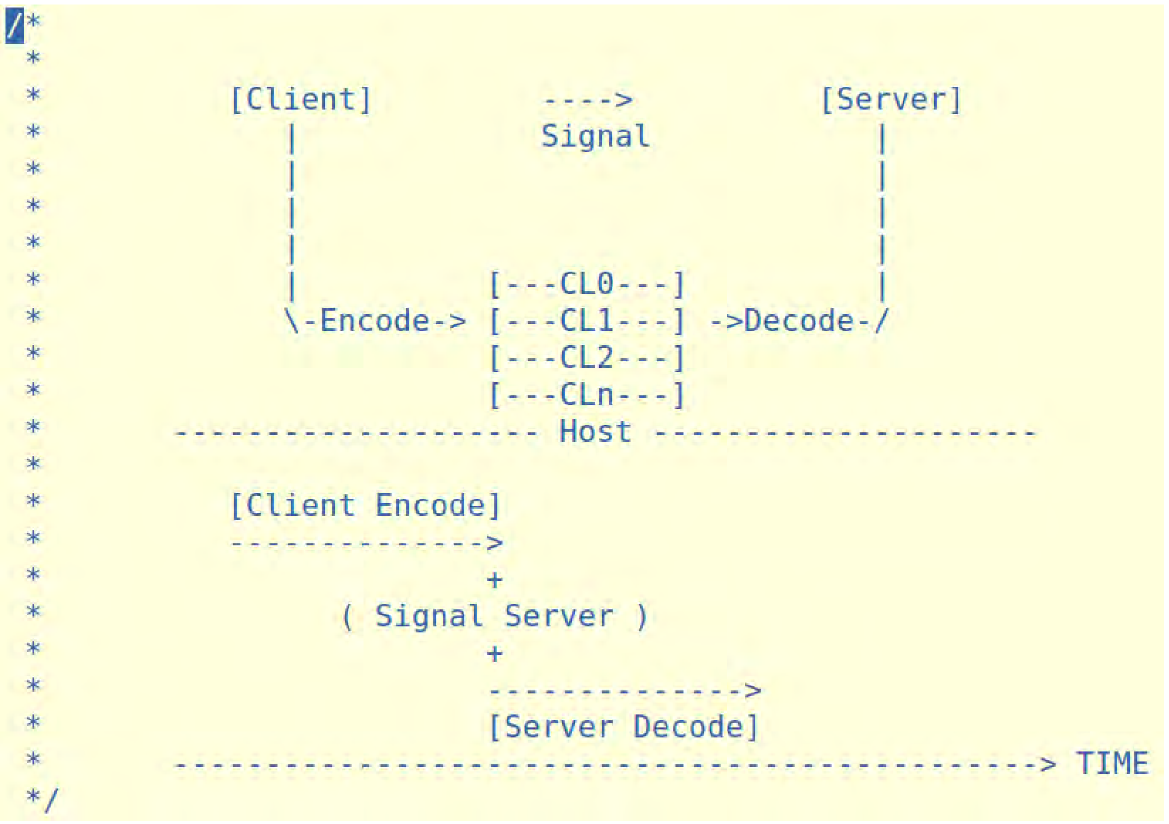
```
VM#1 encode a pattern in cache line
 CL0 |  CL1 |  CL2 |  CL3 |  CL4
  1  |   0  |   0  |   0  |   1
Load | Flush| Flush| Flush| Load
```



Cache line (64 bytes)

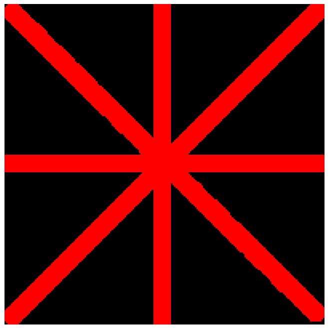


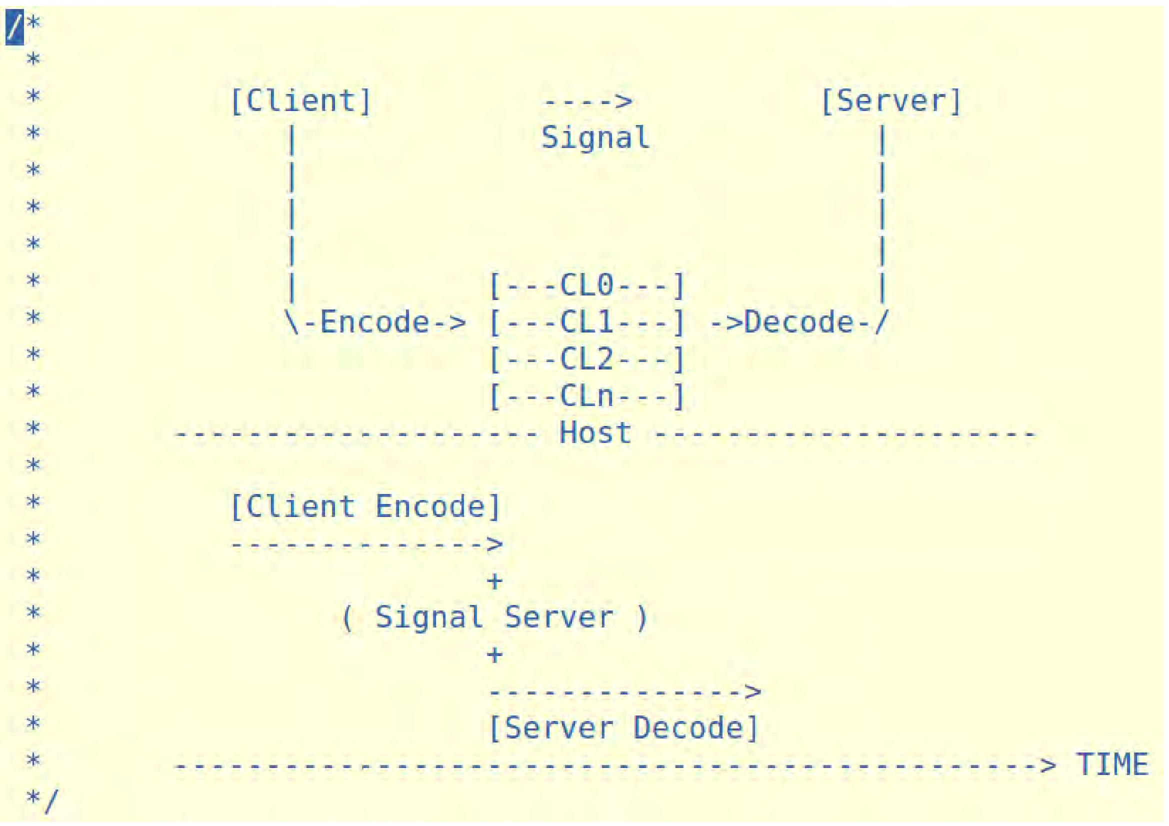
```
VM#1 encode a pattern in cache line
 CL0 |  CL1 |  CL2 |  CL3 |  CL4
  1  |   0  |   0  |   0  |   1
Load | Flush| Flush| Flush| Load

VM#2 decode the cache line access time
 CL0 |  CL1 |  CL2 |  CL3 |  CL4
 Fast| Slow | Slow | Slow |  Fast
  1  |   0  |   0  |   0  |   1
```



```
/*
 *
 *          [Client]         ---->          [Server]
 *             |             Signal            |
 *             |                               |
 *             |                               |
 *             |                               |
 *             |          [---CL0---]          |
 *             \-Encode-> [---CL1---] ->Decode-/
 *                        [---CL2---]
 *                        [---CLn---]
 *       -------------------- Host ---------------------
 *
 *          [Client Encode]
 *          -------------->
 *                        +
 *                ( Signal Server )
 *                        +
 *                        -------------->
 *                        [Server Decode]
 *       ------------------------------------------------> TIME
 */
```

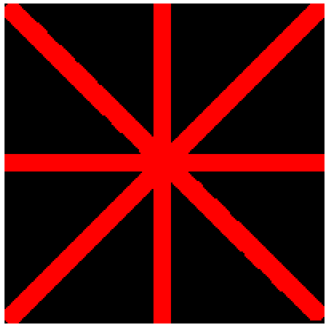
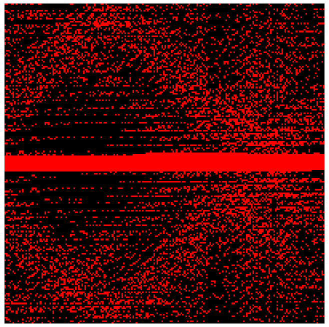
- **NO VM**
- Simple Client / Server test program
- Cache Line from shared memory directly
- Mutex for inter-process signaling
- Client encode a pattern



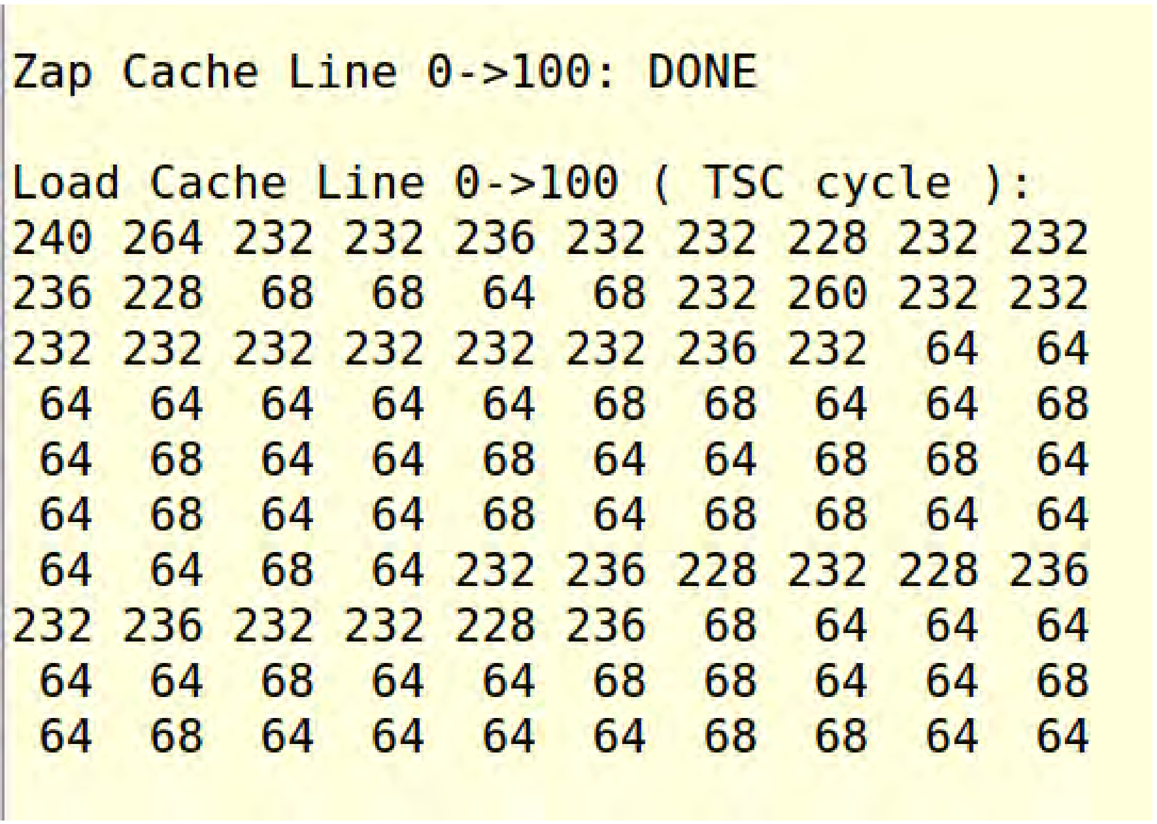


```
/*
 *
 *          [Client]         ---->          [Server]
 *             |             Signal            |
 *             |                               |
 *             |                               |
 *             |                               |
 *             |          [---CL0---]          |
 *             \-Encode-> [---CL1---] ->Decode-/
 *                        [---CL2---]
 *                        [---CLn---]
 *       -------------------- Host ---------------------
 *
 *          [Client Encode]
 *          -------------->
 *                        +
 *                ( Signal Server )
 *                        +
 *                        -------------->
 *                        [Server Decode]
 *       ------------------------------------------------> TIME
 */
```

- **NO VM**
- Simple Client / Server test program
- Cache Line from shared memory directly
- Mutex for inter-process signaling
- Client encode a pattern
- Server decode
- ➔Something weird?

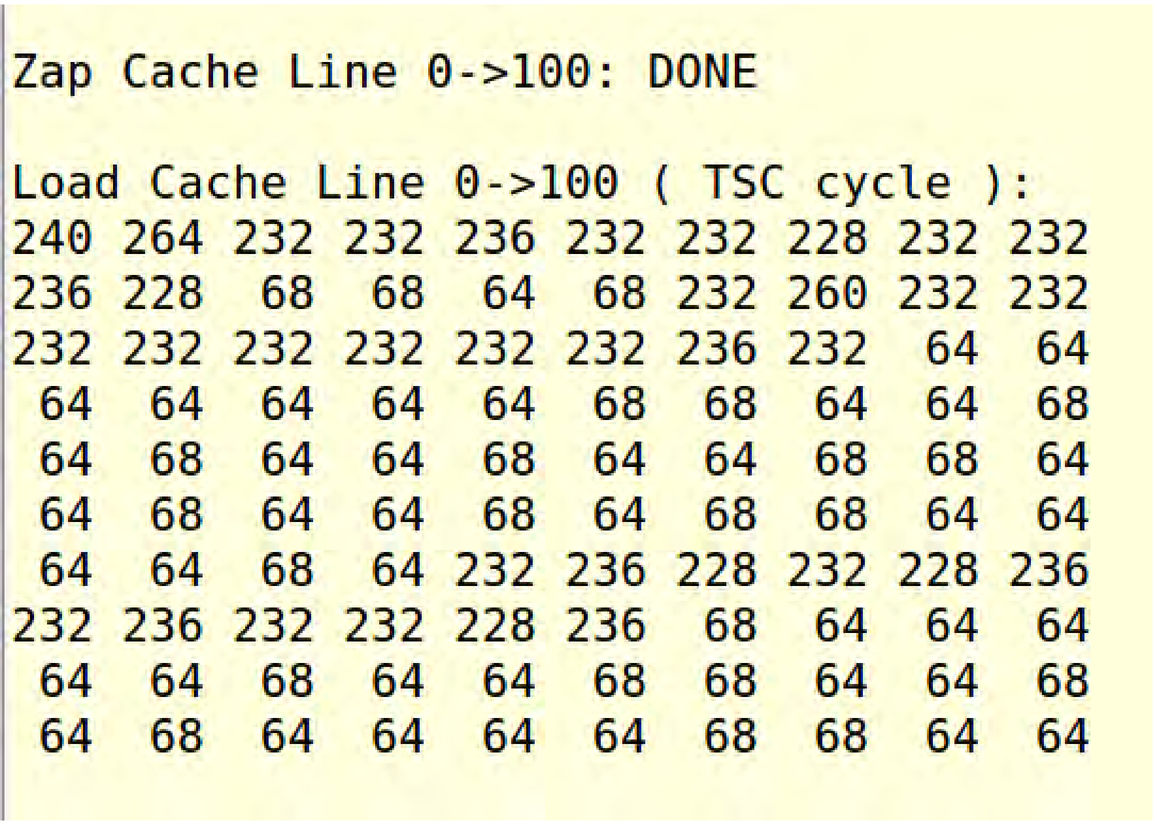



- Simple test:
- Flush CL0 -> CL100
- Measure CL access time for CL0 -> CL100
- ➔ Long latency for all CL



```
Zap Cache Line 0->100: DONE

Load Cache Line 0->100 ( TSC cycle ):
240 264 232 232 236 232 232 228 232 232
236 228  68  68  64  68 232 260 232 232
232 232 232 232 232 232 236 232  64  64
 64  64  64  64  64  68  68  64  64  68
 64  68  64  64  68  64  64  68  68  64
 64  68  64  64  68  64  68  68  64  64
 64  64  68  64 232 236 228 232 228 236
232 236 232 232 228 236  68  64  64  64
 64  64  68  64  64  68  68  64  64  68
 64  68  64  64  64  64  68  68  64  64
```

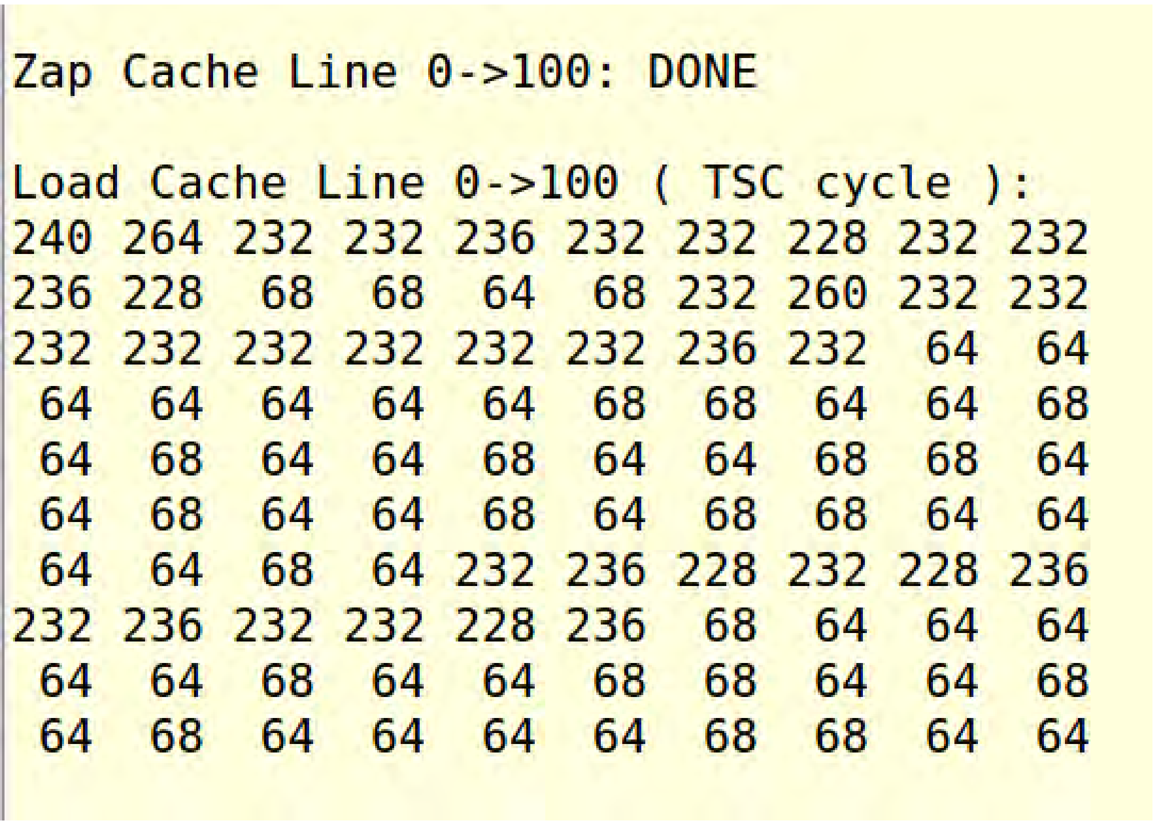
- Simple test:
- Flush CL0 -> CL100
- Measure CL access time for CL0 -> CL100
- ➔ Long latency for all CL
- ???



```
Zap Cache Line 0->100: DONE

Load Cache Line 0->100 ( TSC cycle ):
240 264 232 232 236 232 232 228 232 232
236 228  68  68  64  68 232 260 232 232
232 232 232 232 232 232 236 232  64  64
 64  64  64  64  64  68  68  64  64  68
 64  68  64  64  68  64  64  68  68  64
 64  68  64  64  68  64  68  68  64  64
 64  64  68  64 232 236 228 232 228 236
232 236 232 232 228 236  68  64  64  64
 64  64  68  64  64  68  68  64  64  68
 64  68  64  64  64  64  68  68  64  64
```

- Simple test:
- Flush CL0 -> CL100
- Measure CL access time for CL0 -> CL100
- ➔ Long latency for all CL
- ???

Prefetching in general means bringing data or instructions from memory into the cache **before they are needed**



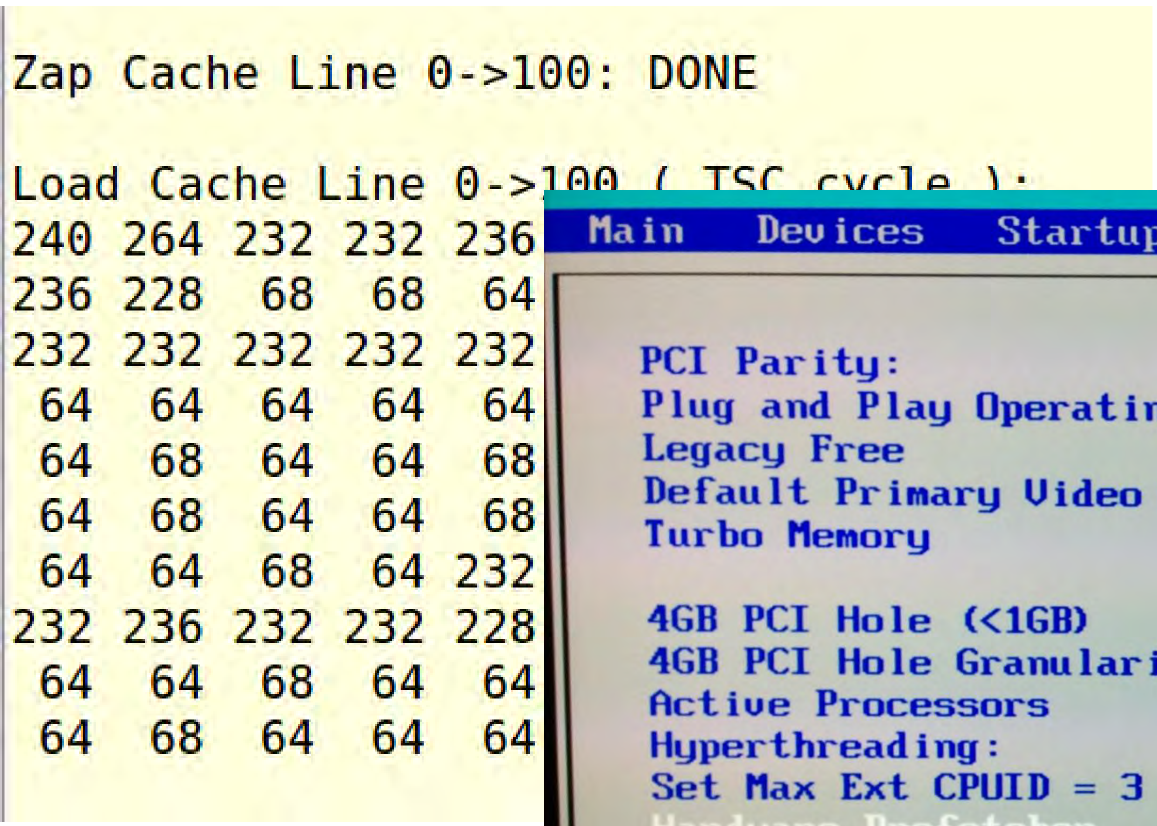
```
Zap Cache Line 0->100: DONE

Load Cache Line 0->100 ( TSC cycle ):
240 264 232 232 236 232 232 228 232 232
236 228  68  68  64  68 232 260 232 232
232 232 232 232 232 232 236 232  64  64
 64  64  64  64  64  68  68  64  64  68
 64  68  64  64  68  64  64  68  68  64
 64  68  64  64  68  64  68  68  64  64
 64  64  68  64 232 236 228 232 228 236
232 236 232 232 228 236  68  64  64  64
 64  64  68  64  64  68  68  64  64  68
 64  68  64  64  64  64  68  68  64  64
```

- Simple test:
- Flush CL0 -> CL100
- Measure CL access time for CL0 -> CL100
- ➔ Long latency for all CL
- ???

Prefetching in general means bringing data or instructions from memory into the cache **before they are needed**

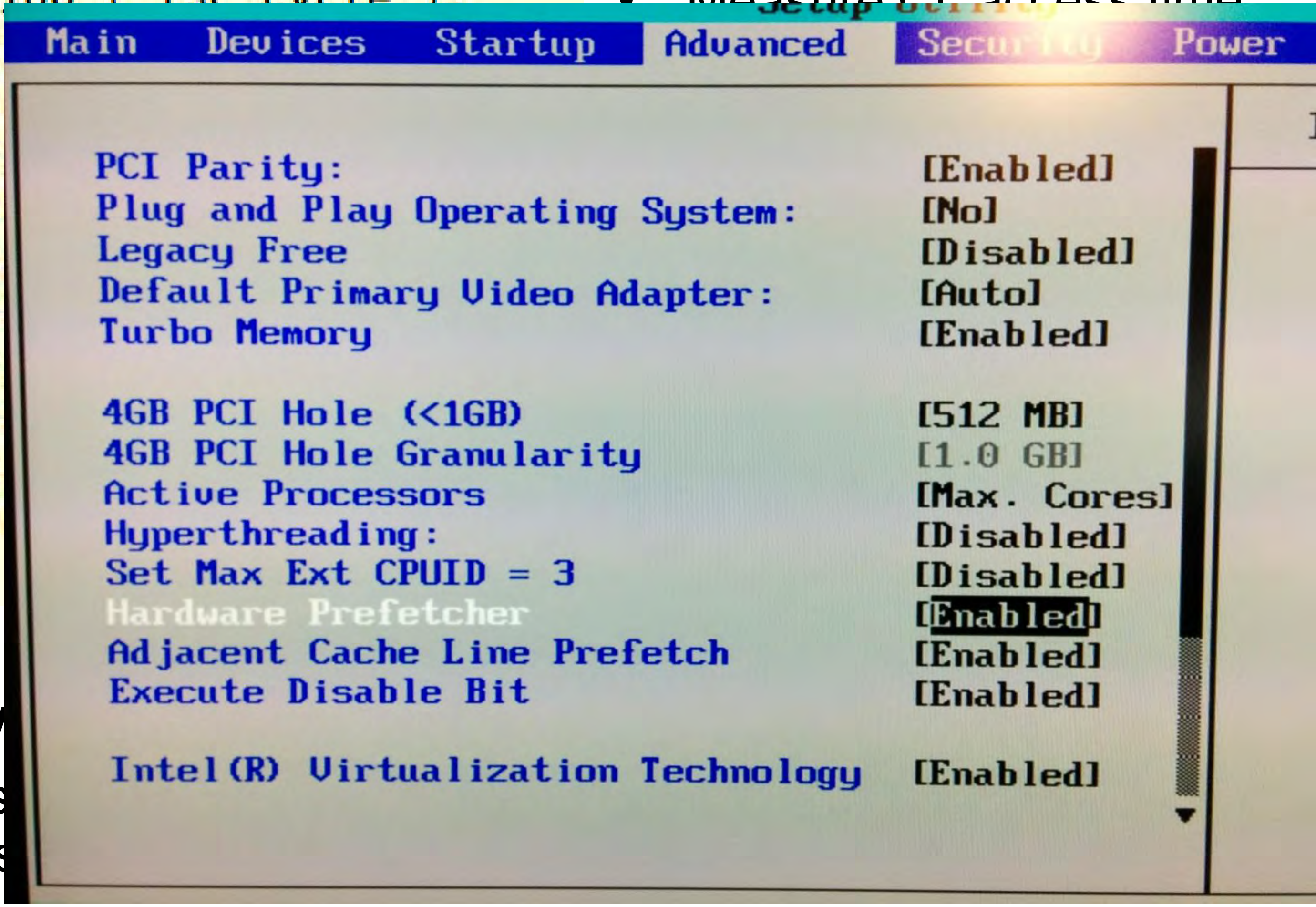
The Core™ i7 processor and Xeon® 5500 series processors, for example, have some prefetchers that bring data into the L1 cache and some that bring data into the L2.

There are also different algorithms – some monitor data access patterns for a particular cache and then **try to predict what addresses will be needed in the future.**



```
Zap Cache Line 0->100: DONE

Load Cache Line 0->100 ( TSC cycle ):
240 264 232 232 236
236 228  68  68  64
232 232 232 232 232
 64  64  64  64  64
 64  68  64  64  68
 64  68  64  64  68
 64  64  68  64 232
232 236 232 232 228
 64  64  68  64  64
 64  68  64  64  64
```

- Simple test:
- Flush CL0 -> CL100
- Measure CL access time

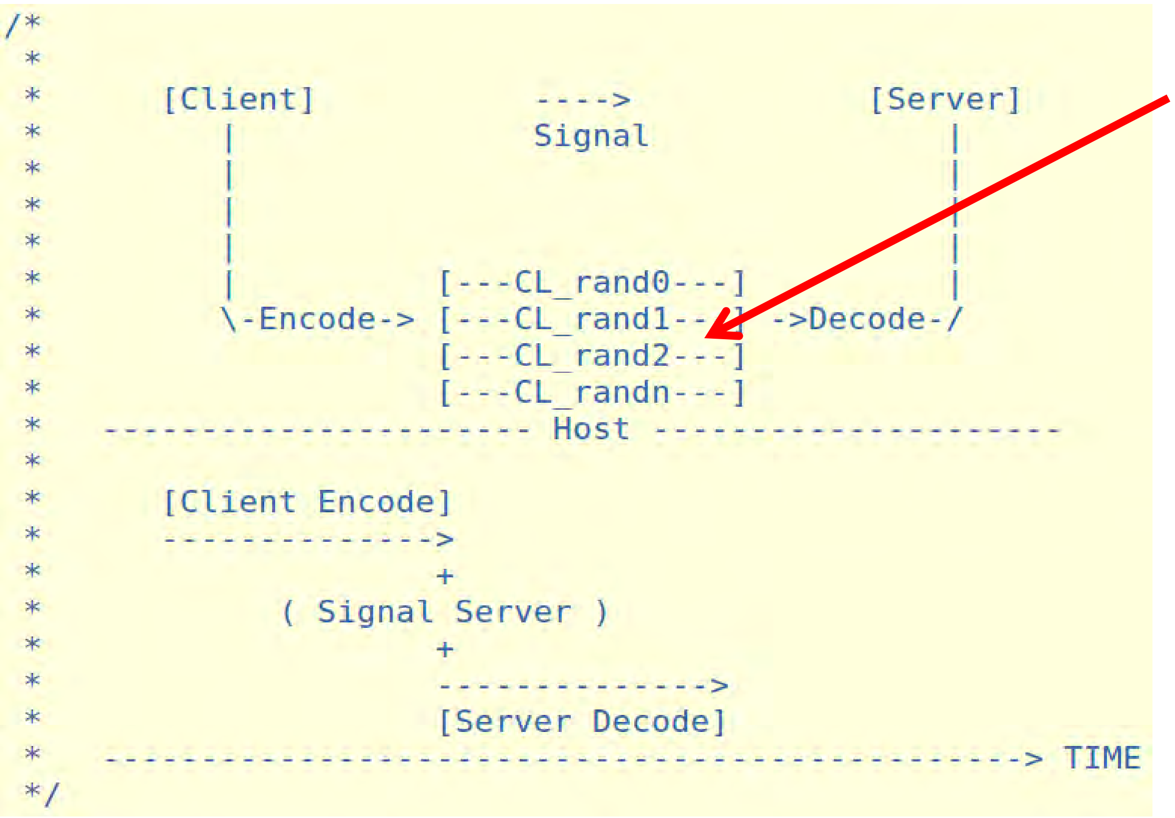


```
PCI Parity:                                  [Enabled]
Plug and Play Operating System:              [No]
Legacy Free                                  [Disabled]
Default Primary Video Adapter:               [Auto]
Turbo Memory                                 [Enabled]

4GB PCI Hole (<1GB)                          [512 MB]
4GB PCI Hole Granularity                     [1.0 GB]
Active Processors                            [Max. Cores]
Hyperthreading:                              [Disabled]
Set Max Ext CPUID = 3                        [Disabled]
Hardware Prefetcher                          [Enabled]
Adjacent Cache Line Prefetch                 [Enabled]
Execute Disable Bit                          [Enabled]

Intel(R) Virtualization Technology           [Enabled]
```

Prefetching in general
the cache **before they**

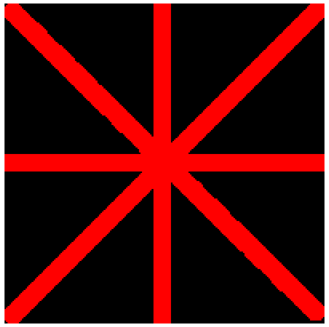
The Core™ i7 process
have some prefetchers
data into the L2.

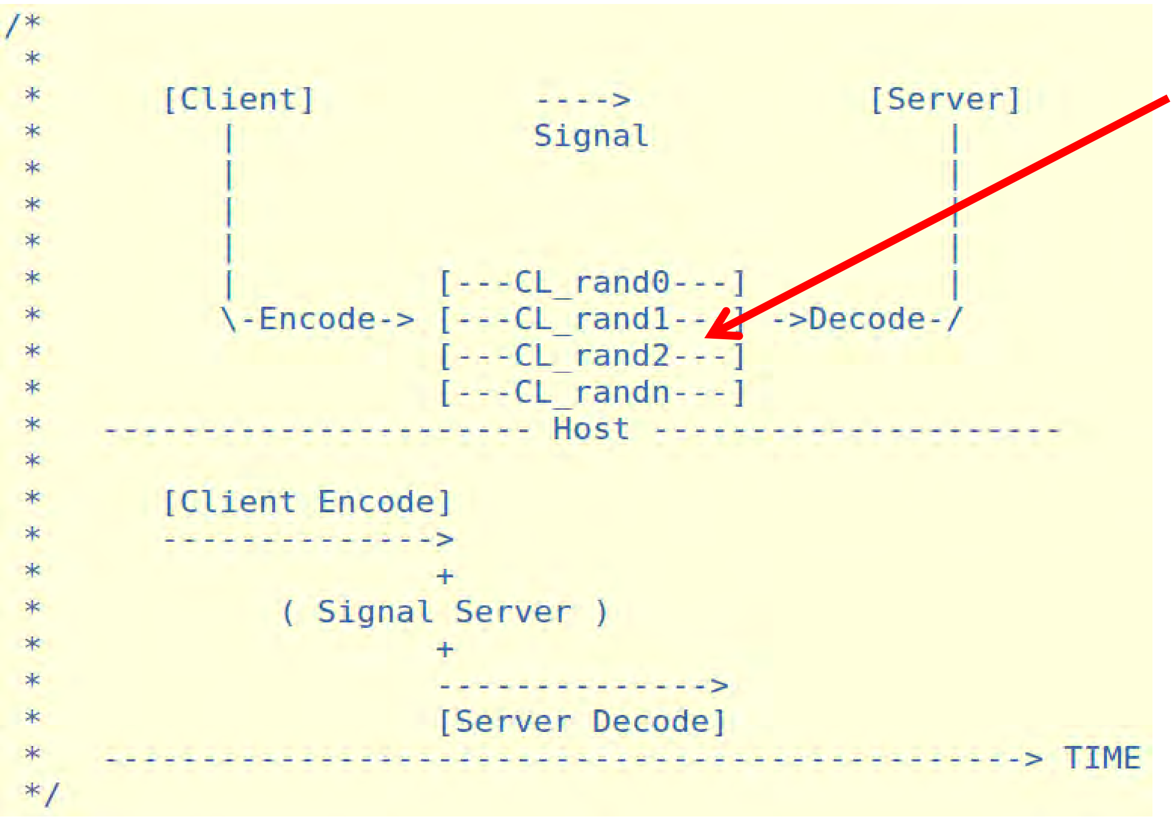
There are also different algorithms – some monitor data access patterns for a particular cache and then **try to predict what addresses will be needed in the future.**



```
/*
 *
 *     [Client]              ---->              [Server]
 *        |                  Signal                 |
 *        |                                         |
 *        |                                         |
 *        |                                         |
 *        |             [---CL_rand0---]            |
 *        \-Encode-> [---CL_rand1---] ->Decode-/
 *                   [---CL_rand2---]
 *                   [---CL_randn---]
 *   ---------------------- Host --------------------
 *
 *     [Client Encode]
 *     -------------->
 *                   +
 *          ( Signal Server )
 *                   +
 *                   -------------->
 *                   [Server Decode]
 *   ------------------------------------------------> TIME
 */
```

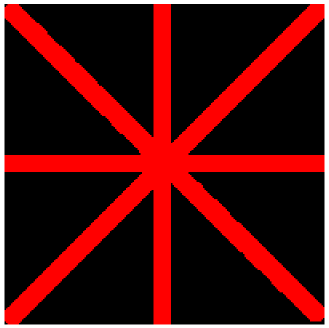
- Simple trick that randomized CL access

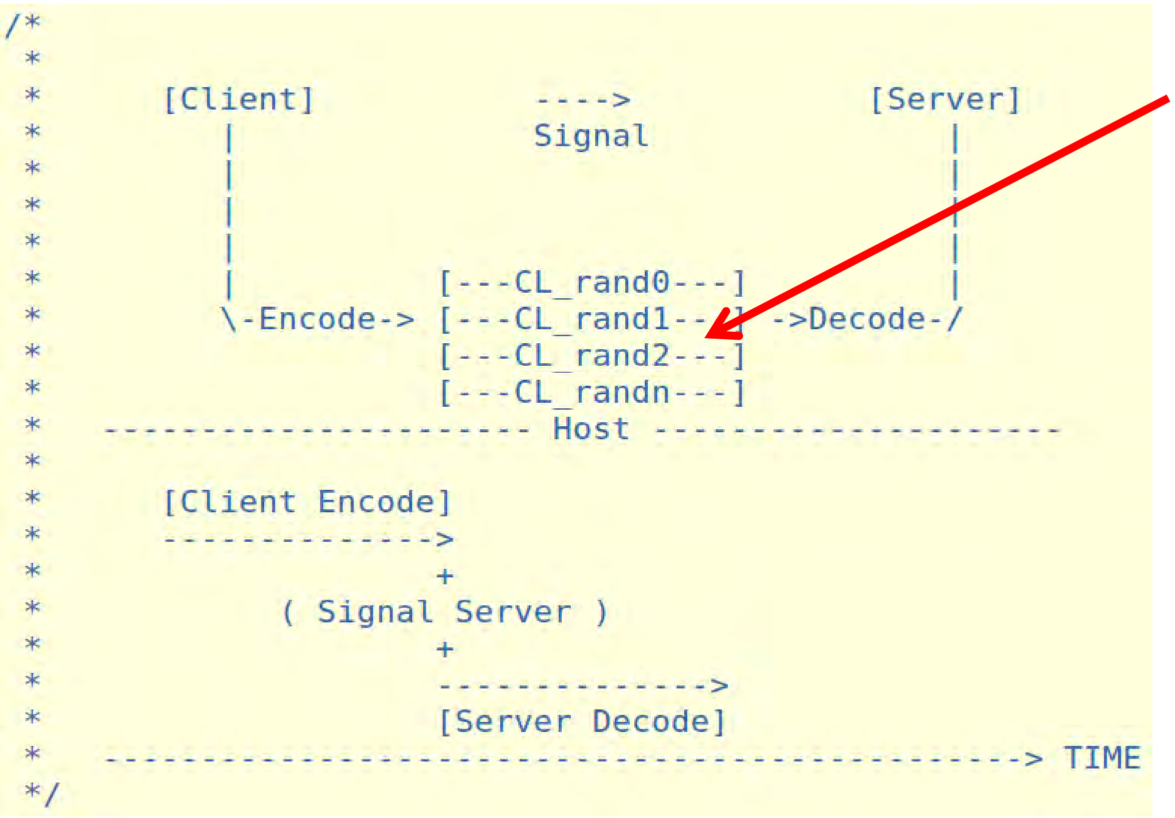




```
/*
 *
 *     [Client]            ---->             [Server]
 *        |                Signal                |
 *        |                                      |
 *        |                                      |
 *        |                                      |
 *        |           [---CL_rand0---]           |
 *        \-Encode-> [---CL_rand1---] ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   -------------------- Host --------------------
 *
 *     [Client Encode]
 *     -------------->
 *                    +
 *          ( Signal Server )
 *                    +
 *                    -------------->
 *                    [Server Decode]
 *   ---------------------------------------------> TIME
 */
```

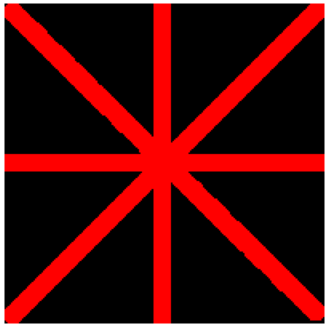
- Simple trick that randomized CL access
- CL access random within a page

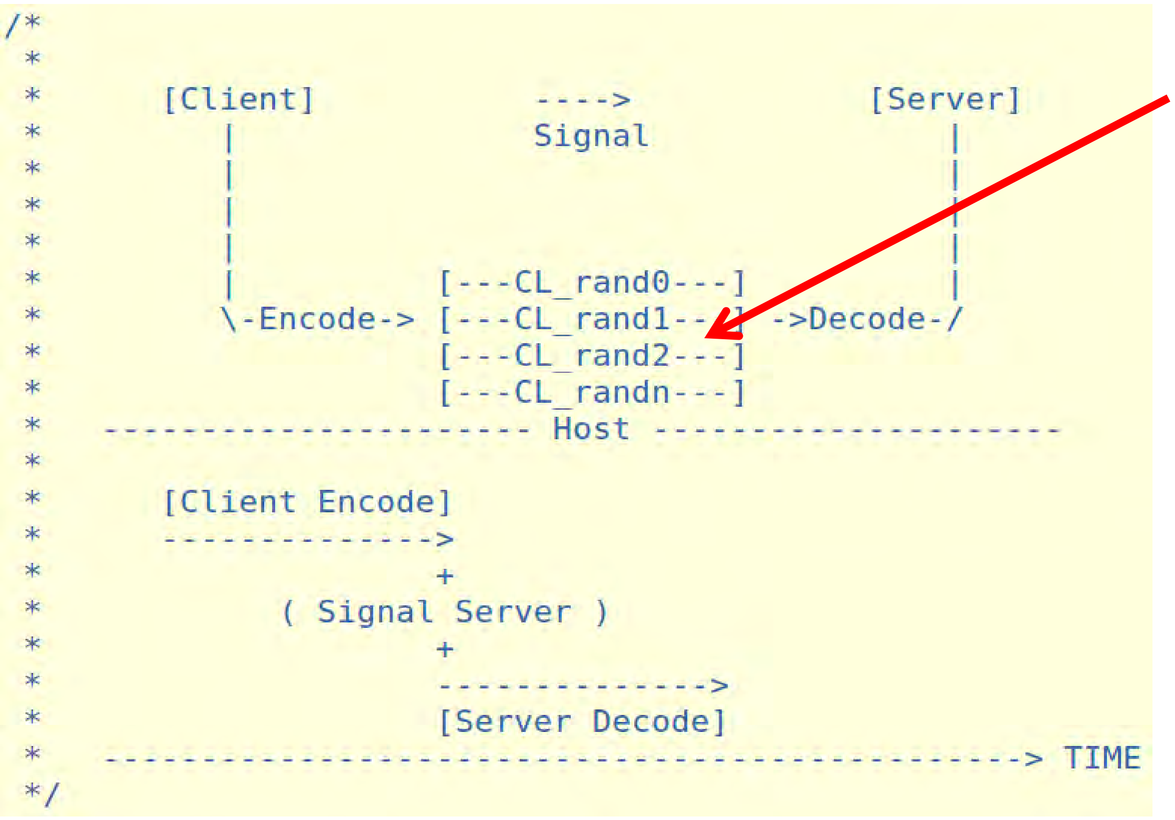




```
/*
 *
 *      [Client]            ---->            [Server]
 *         |                Signal              |
 *         |                                    |
 *         |                                    |
 *         |                                    |
 *         |          [---CL_rand0---]          |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *    ---------------------- Host --------------------
 *
 *      [Client Encode]
 *      -------------->
 *                    +
 *            ( Signal Server )
 *                    +
 *                    -------------->
 *                    [Server Decode]
 *    ------------------------------------------------> TIME
 */
```

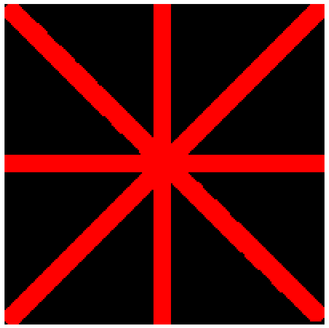
- Simple trick that randomized CL access
- CL access random within a page
- CL access random across pages

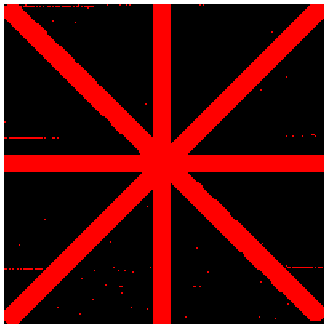


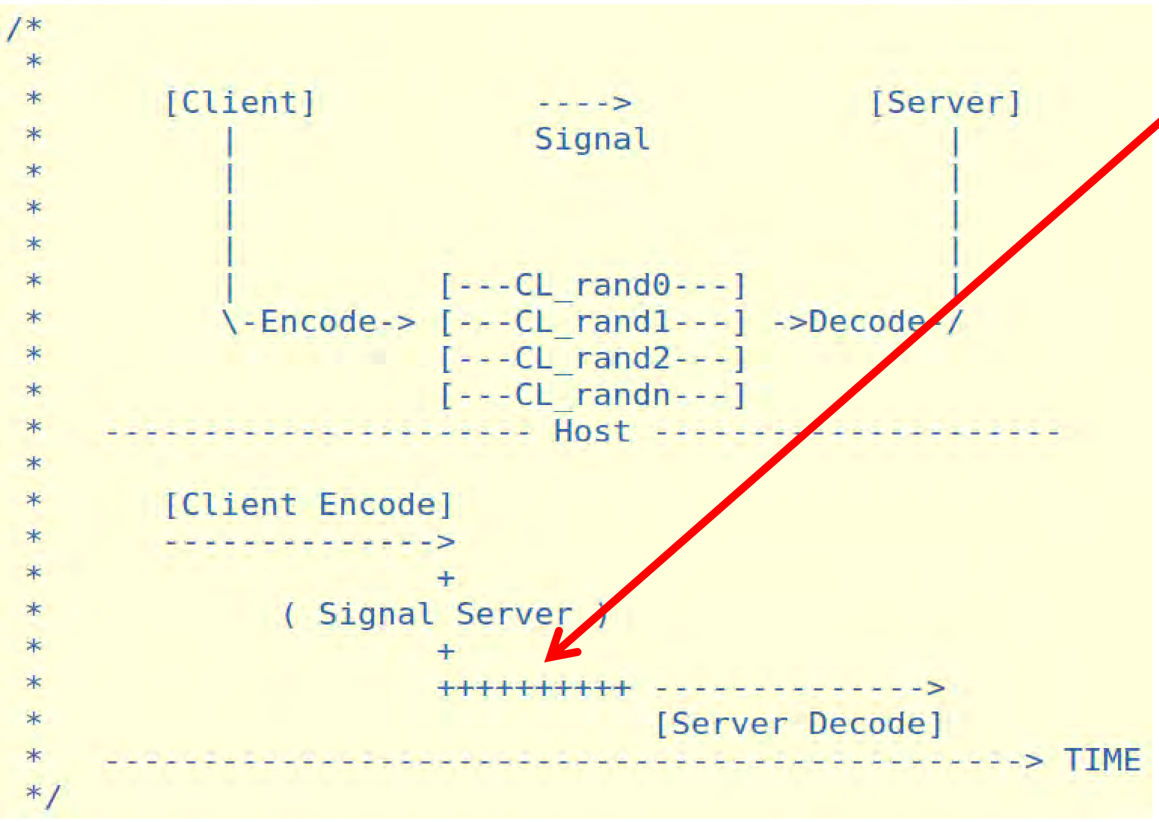


```
/*
 *
 *     [Client]            ---->            [Server]
 *        |                Signal               |
 *        |                                     |
 *        |                                     |
 *        |                                     |
 *        |           [---CL_rand0---]          |
 *        \-Encode-> [---CL_rand1---] ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   --------------------- Host ---------------------
 *
 *     [Client Encode]
 *     -------------->
 *                    +
 *           ( Signal Server )
 *                    +
 *                    -------------->
 *                    [Server Decode]
 *   ----------------------------------------------> TIME
 */
```

- Simple trick that randomized CL access
- CL access random within a page
- CL access random across pages
- This apparently manage to confuse the HW prefetcher!

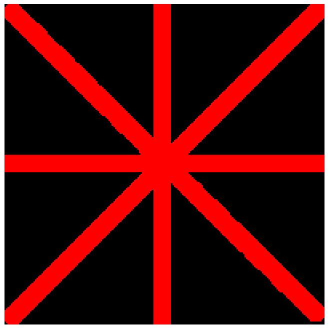


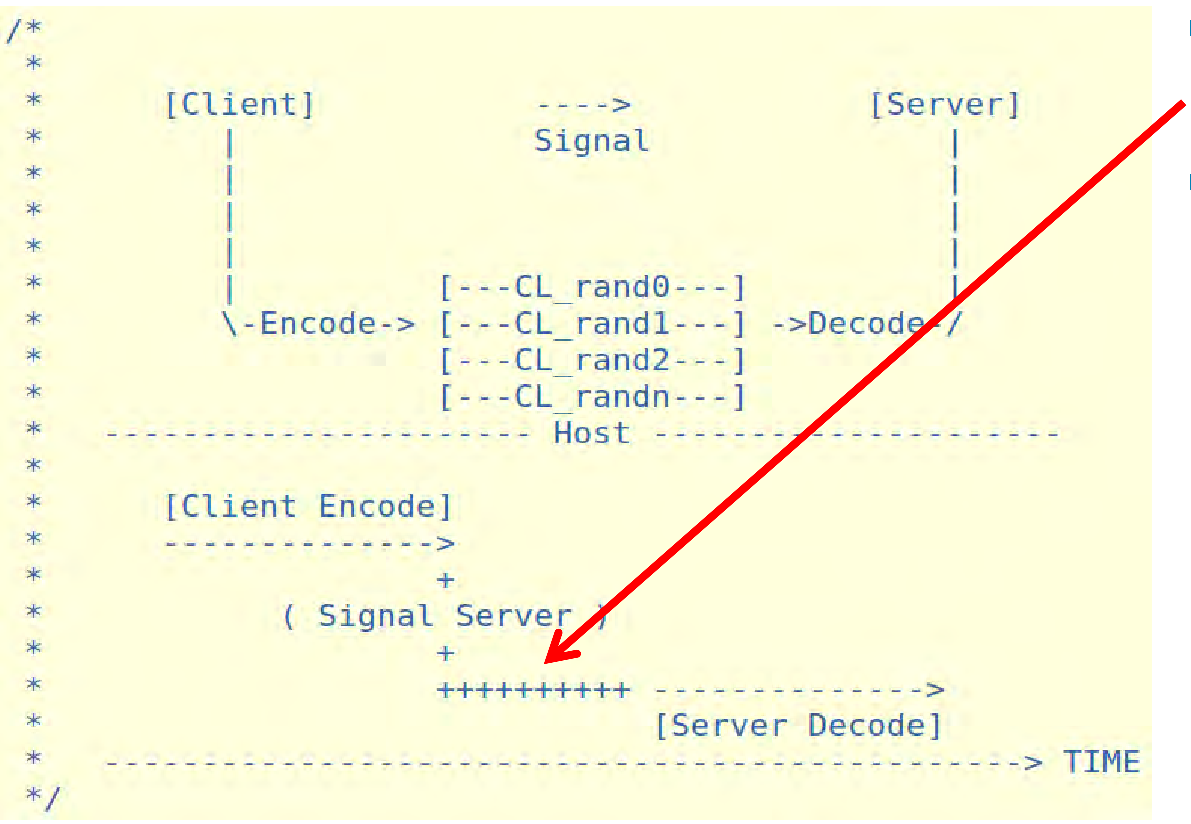




```
/*
 *
 *     [Client]           ---->          [Server]
 *        |               Signal              |
 *        |                                   |
 *        |                                   |
 *        |                                   |
 *        |            [---CL_rand0---]       |
 *        \-Encode->   [---CL_rand1---]  ->Decode-/
 *                     [---CL_rand2---]
 *                     [---CL_randn---]
 *   ---------------------- Host ----------------------
 *
 *     [Client Encode]
 *     -------------->
 *                     +
 *           ( Signal Server )
 *                     +
 *                     ++++++++++ ------------->
 *                                [Server Decode]
 *   -------------------------------------------------> TIME
 */
```

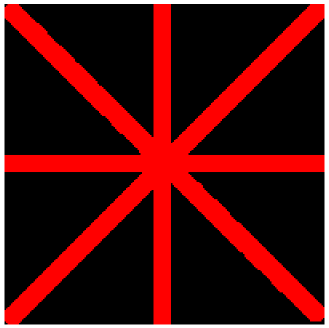
- What happen if we wait longer before decoding?

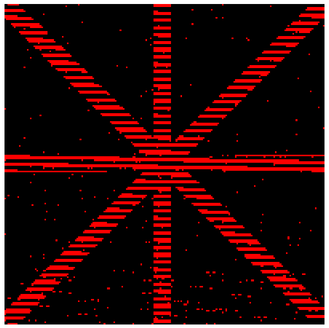


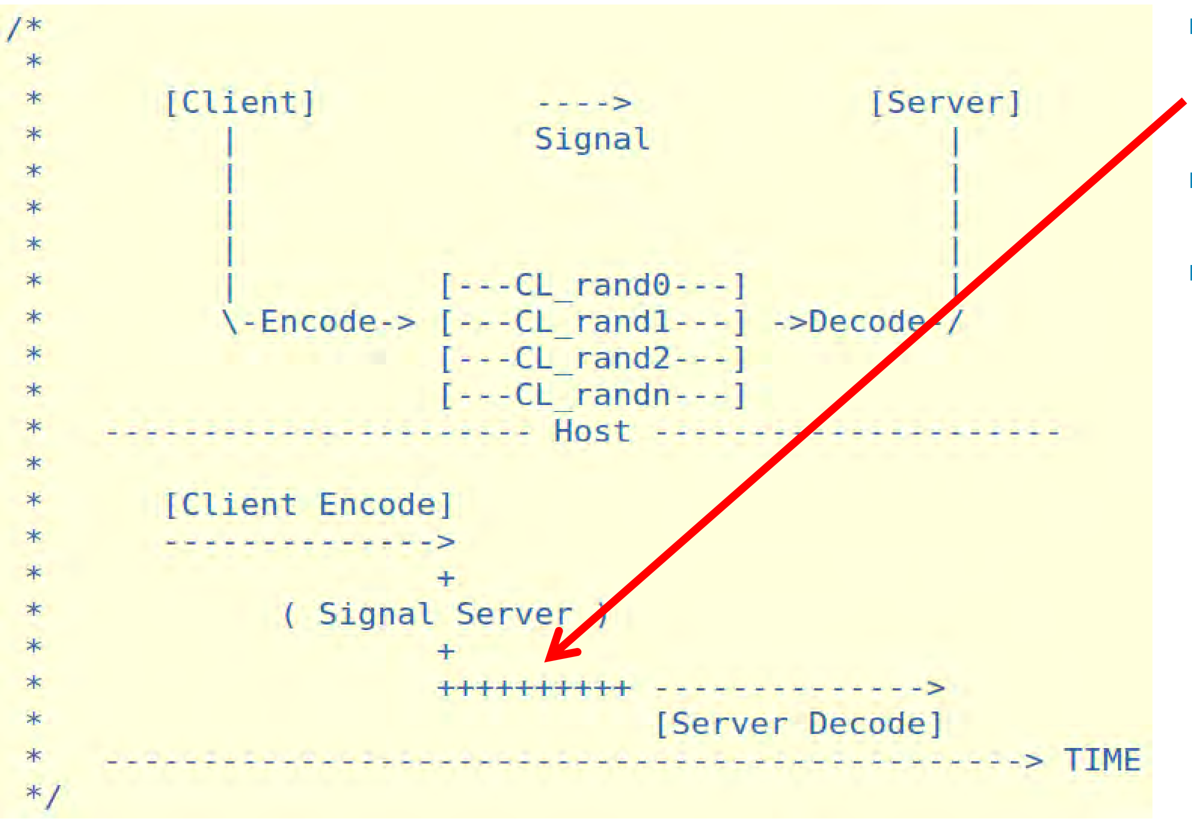


```
/*
 *
 *     [Client]           ---->           [Server]
 *        |               Signal              |
 *        |                                   |
 *        |                                   |
 *        |                                   |
 *        |           [---CL_rand0---]        |
 *        \-Encode-> [---CL_rand1---] ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   ---------------------- Host ----------------------
 *
 *     [Client Encode]
 *     --------------->
 *                    +
 *           ( Signal Server )
 *                    +
 *                    ++++++++++ ------------->
 *                               [Server Decode]
 *   -------------------------------------------------> TIME
 */
```

- What happen if we wait longer before decoding?
- Wait

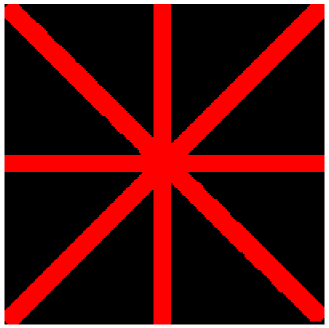


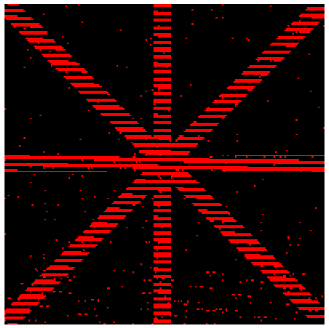


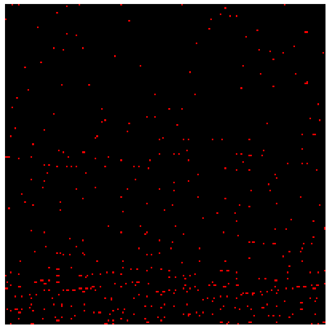


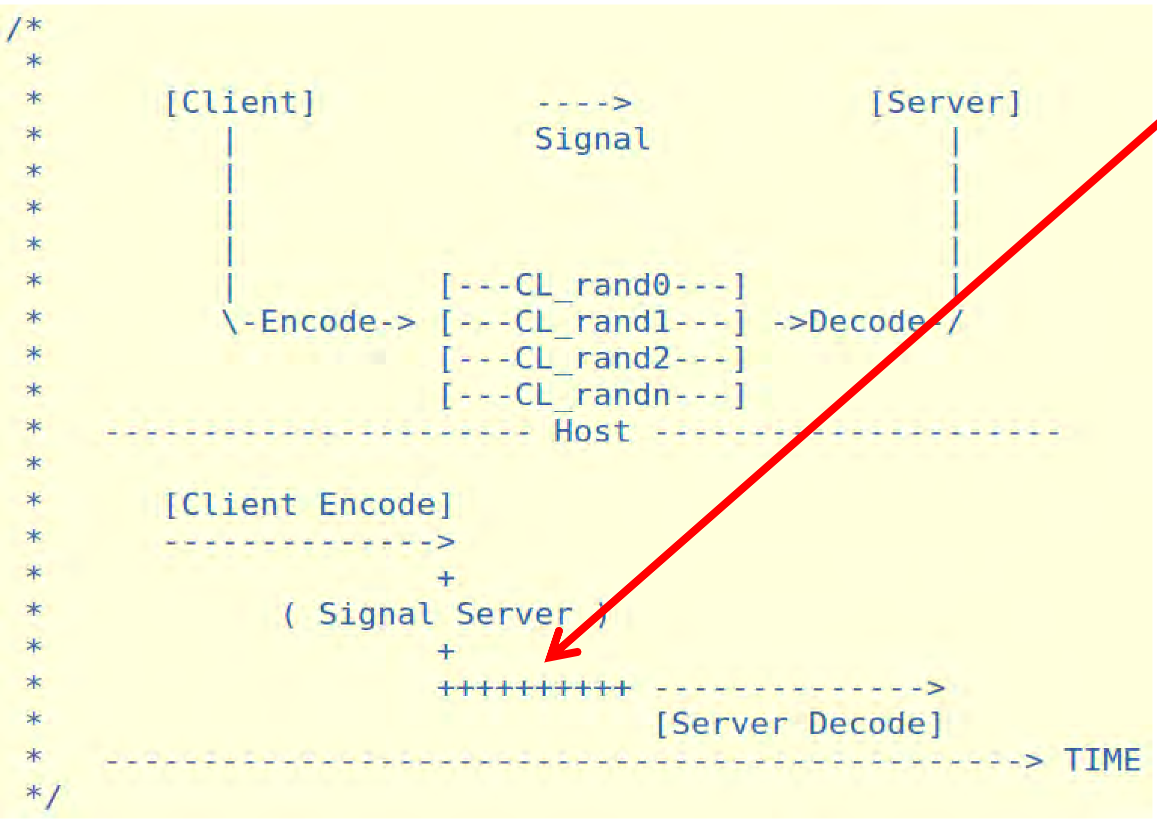
```
/*
 *
 *       [Client]           ---->           [Server]
 *          |               Signal              |
 *          |                                   |
 *          |                                   |
 *          |                                   |
 *          |          [---CL_rand0---]         |
 *          \-Encode-> [---CL_rand1---] ->Decode-/
 *                     [---CL_rand2---]
 *                     [---CL_randn---]
 *    ----------------------- Host ----------------------
 *
 *       [Client Encode]
 *       --------------->
 *                      +
 *             ( Signal Server )
 *                      +
 *                      ++++++++++ -------------->
 *                                 [Server Decode]
 *    ---------------------------------------------------> TIME
 */
```

- What happen if we wait longer before decoding?
- Wait
- Wait



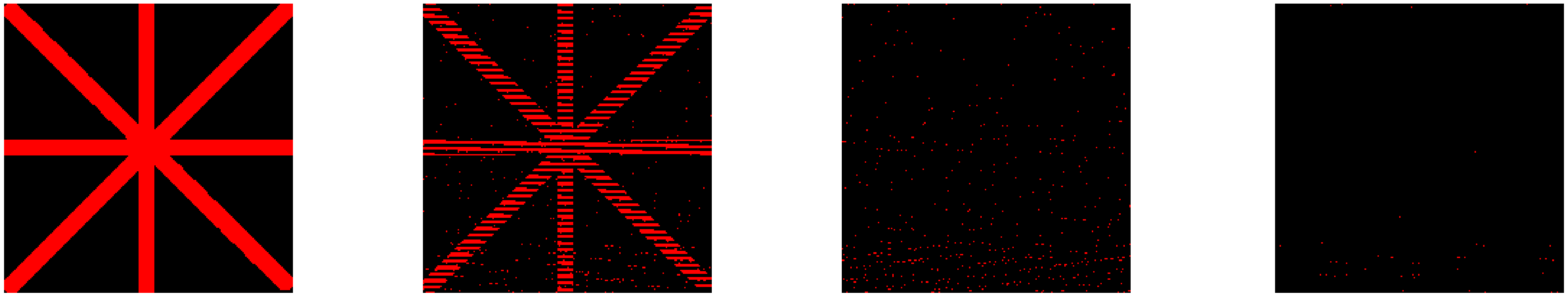


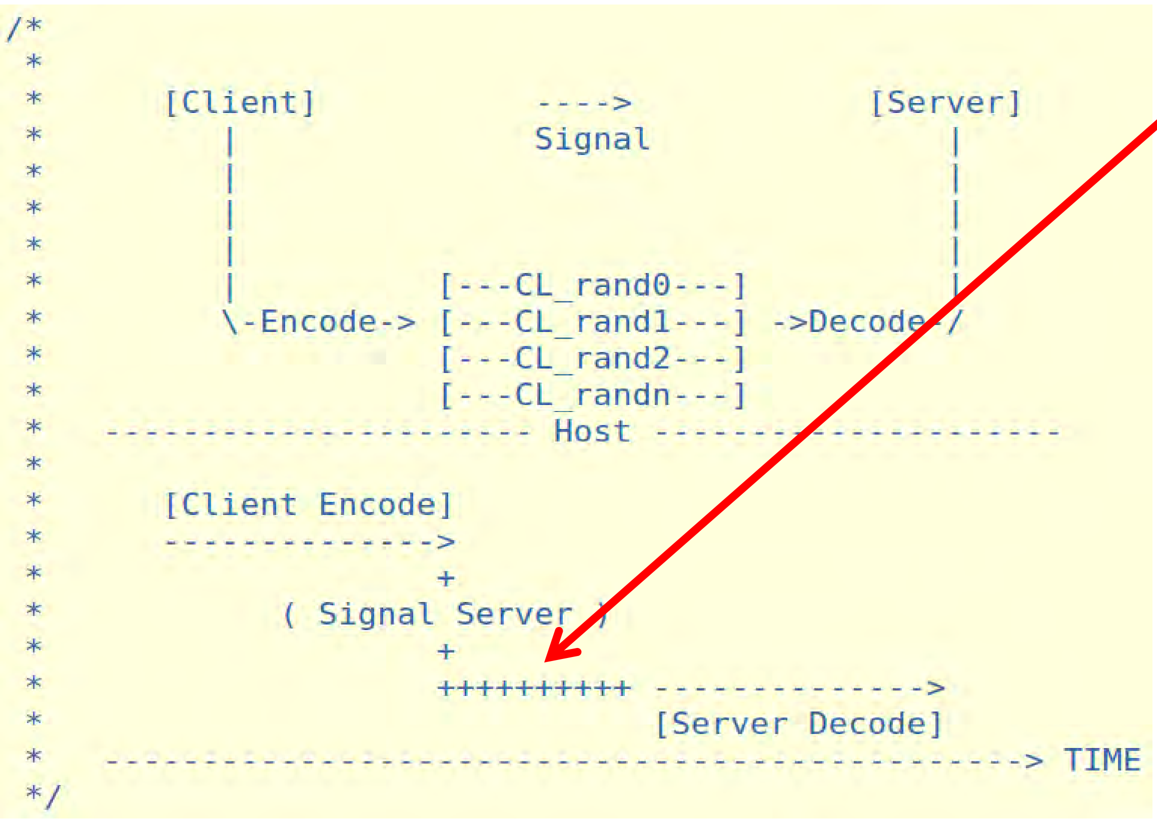




```
/*
 *
 *     [Client]              ---->             [Server]
 *        |                  Signal               |
 *        |                                       |
 *        |                                       |
 *        |                                       |
 *        |             [---CL_rand0---]          |
 *        \-Encode-> [---CL_rand1---] ->Decode-/
 *                   [---CL_rand2---]
 *                   [---CL_randn---]
 *   ----------------------- Host ---------------------
 *
 *     [Client Encode]
 *     --------------->
 *                    +
 *           ( Signal Server )
 *                    +
 *                    ++++++++++ -------------->
 *                               [Server Decode]
 *   ---------------------------------------------------> TIME
 */
```

- What happen if we wait longer before decoding?
- Wait
- Wait
- Wait

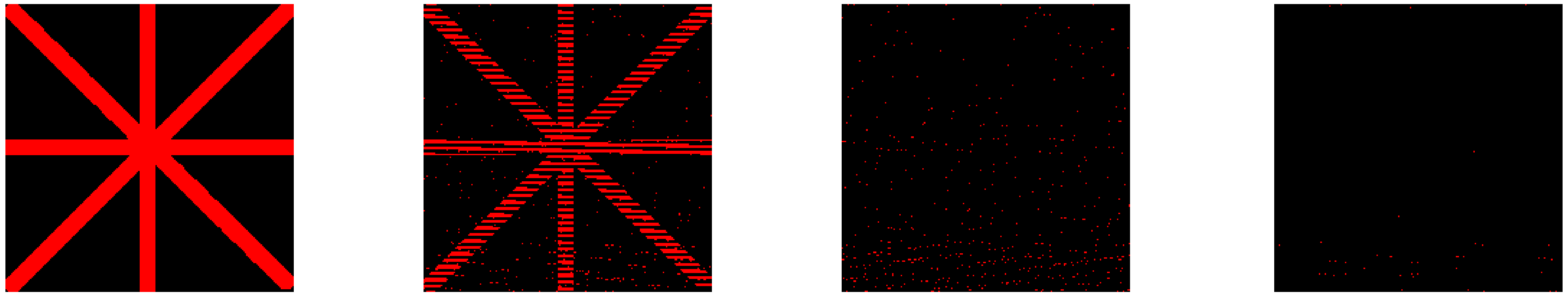




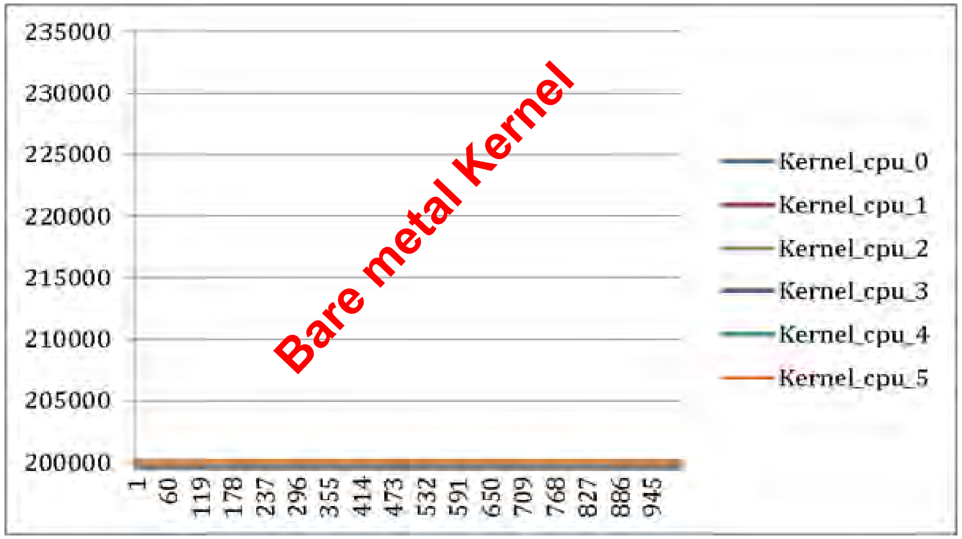
```
/*
 *
 *      [Client]            ---->             [Server]
 *         |                Signal                |
 *         |                                      |
 *         |                                      |
 *         |                                      |
 *         |           [---CL_rand0---]           |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                     [---CL_rand2---]
 *                     [---CL_randn---]
 *    ----------------------- Host ---------------------
 *
 *      [Client Encode]
 *      --------------->
 *                     +
 *             ( Signal Server )
 *                     +
 *                     ++++++++++ ------------->
 *                                 [Server Decode]
 *    ---------------------------------------------------> TIME
 */
```

- What happen if we wait longer before decoding?
- Wait
- Wait
- Wait

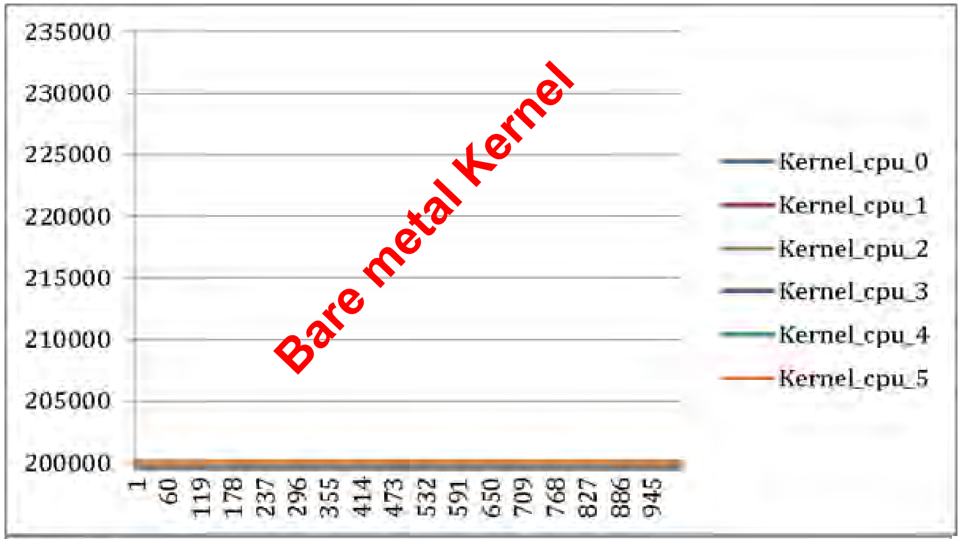
- **Encoded data in the cache evaporates pretty quickly.**



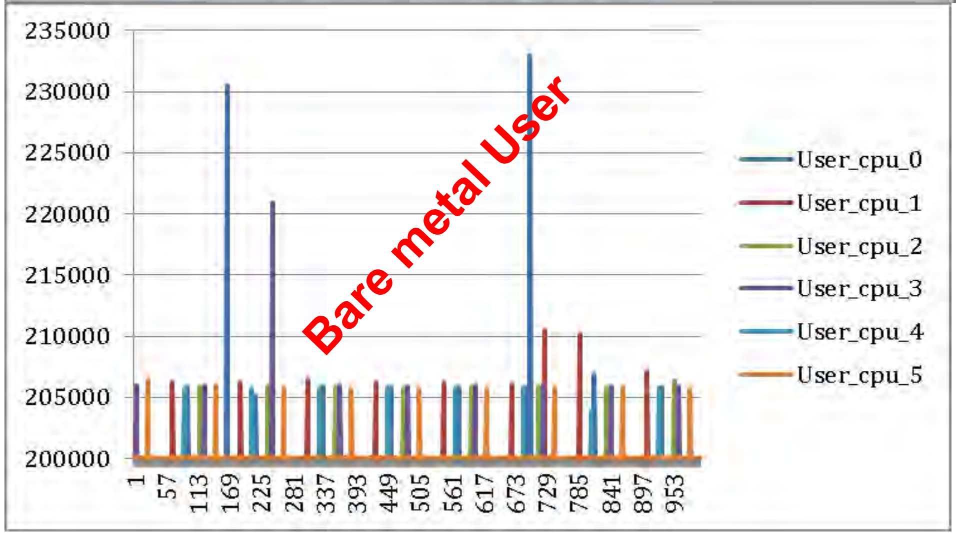
Noise


Bare metal Kernel
235000
230000
225000
220000
215000
210000
205000
200000
1
60
119
178
237
296
355
414
473
532
591
650
709
768
827
886
945
Kernel_cpu_0
Kernel_cpu_1
Kernel_cpu_2
Kernel_cpu_3
Kernel_cpu_4
Kernel_cpu_5

Noise

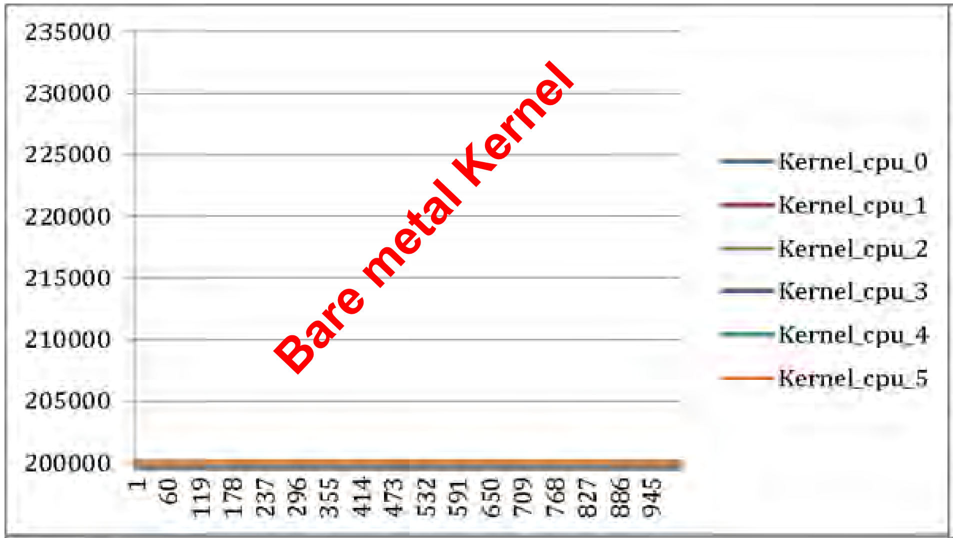


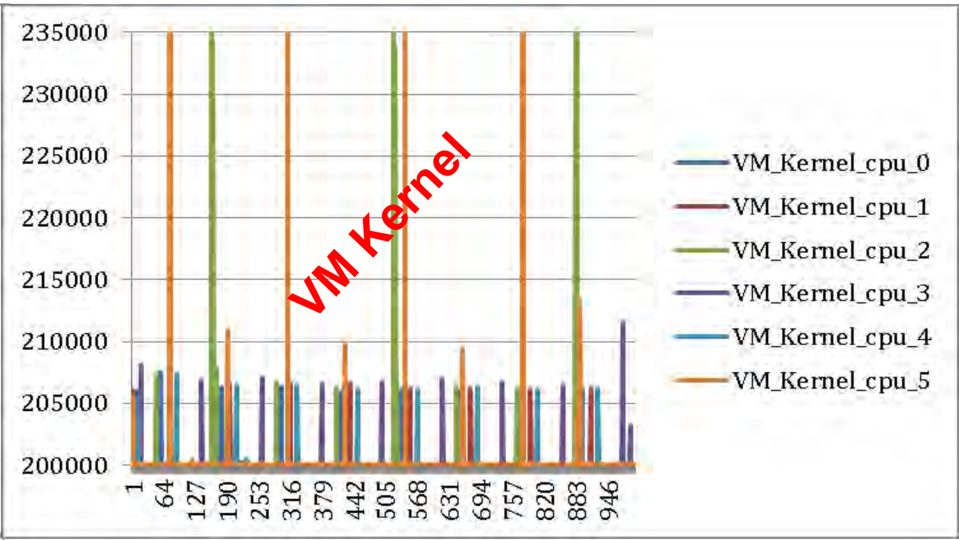
235000
230000
225000
220000
215000
210000
205000
200000
1
60
119
178
237
296
355
414
473
532
591
650
709
768
827
886
945
Kernel_cpu_0
Kernel_cpu_1
Kernel_cpu_2
Kernel_cpu_3
Kernel_cpu_4
Kernel_cpu_5
Bare metal Kernel

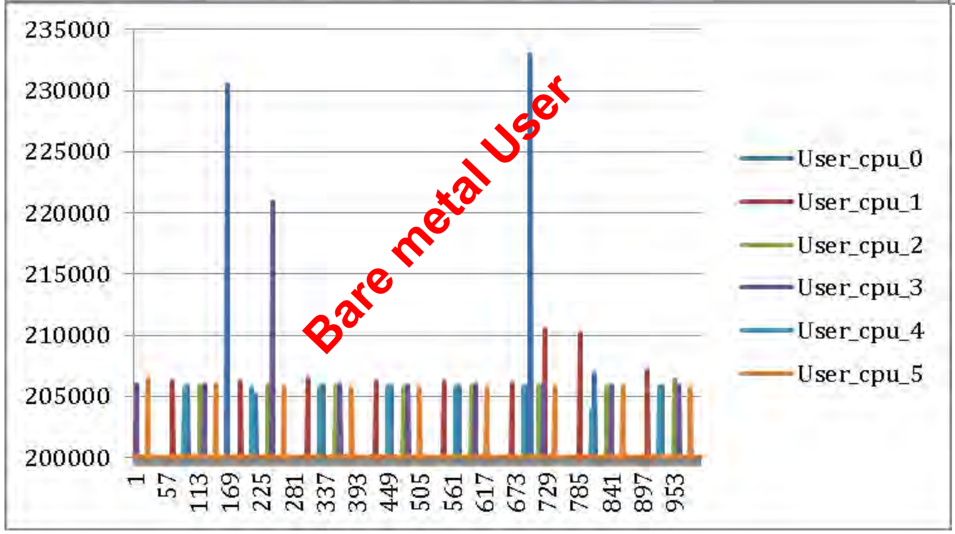


235000
230000
225000
220000
215000
210000
205000
200000
1
57
113
169
225
281
337
393
449
505
561
617
673
729
785
841
897
953
User_cpu_0
User_cpu_1
User_cpu_2
User_cpu_3
User_cpu_4
User_cpu_5
Bare metal User

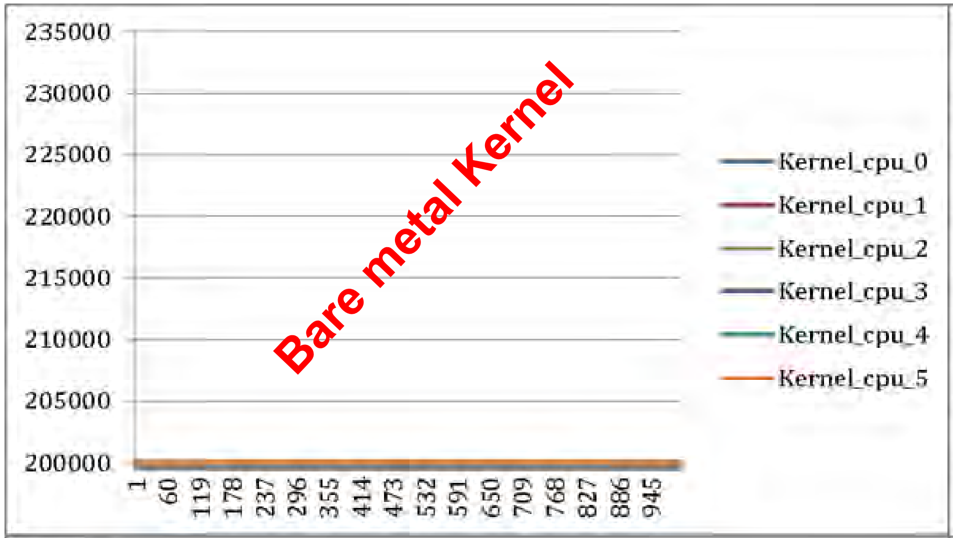
Noise


235000
230000
225000
220000
215000
210000
205000
200000
1
60
119
178
237
296
355
414
473
532
591
650
709
768
827
886
945
Kernel_cpu_0
Kernel_cpu_1
Kernel_cpu_2
Kernel_cpu_3
Kernel_cpu_4
Kernel_cpu_5
Bare metal Kernel

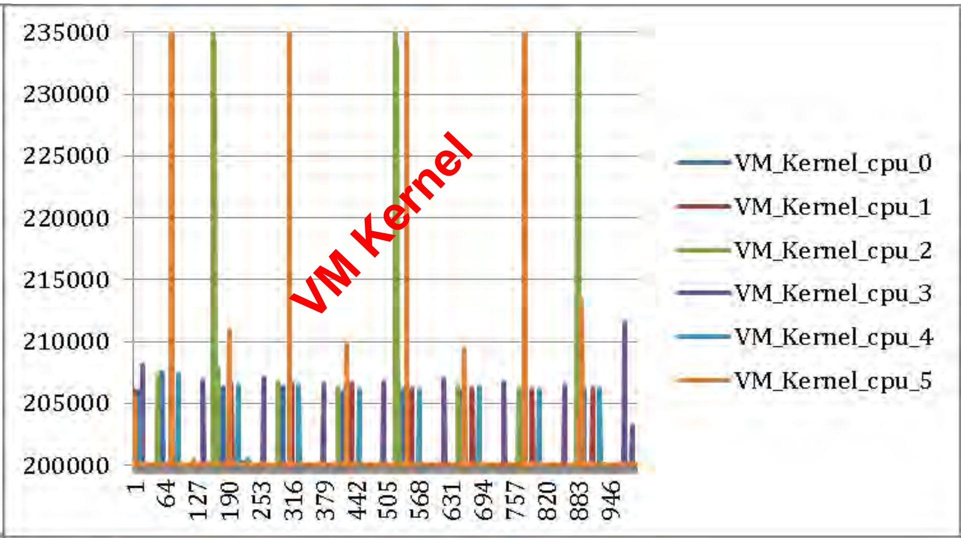

235000
230000
225000
220000
215000
210000
205000
200000
1
64
127
190
253
316
379
442
505
568
631
694
757
820
883
946
VM_Kernel_cpu_0
VM_Kernel_cpu_1
VM_Kernel_cpu_2
VM_Kernel_cpu_3
VM_Kernel_cpu_4
VM_Kernel_cpu_5
VM Kernel


235000
230000
225000
220000
215000
210000
205000
200000
1
57
113
169
225
281
337
393
449
505
561
617
673
729
785
841
897
953
User_cpu_0
User_cpu_1
User_cpu_2
User_cpu_3
User_cpu_4
User_cpu_5
Bare metal User

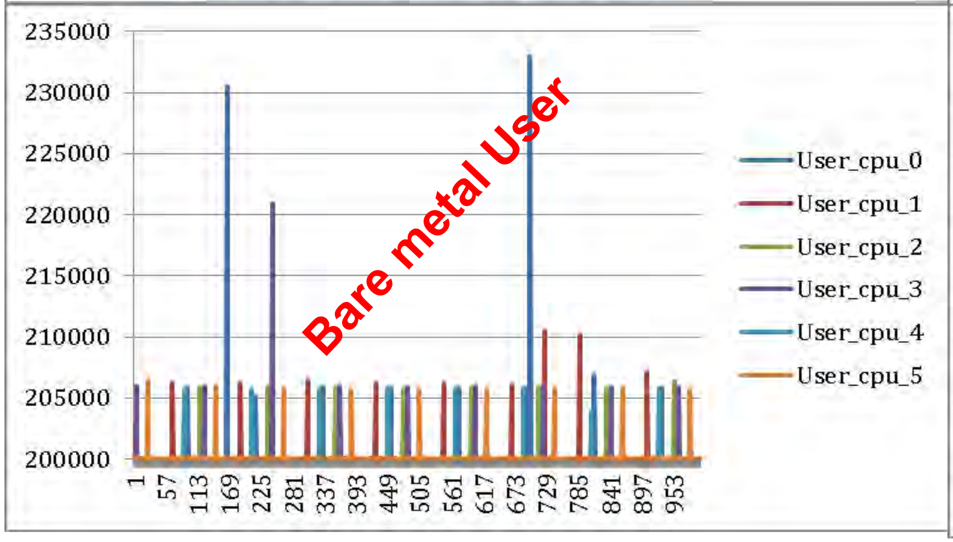
Noise



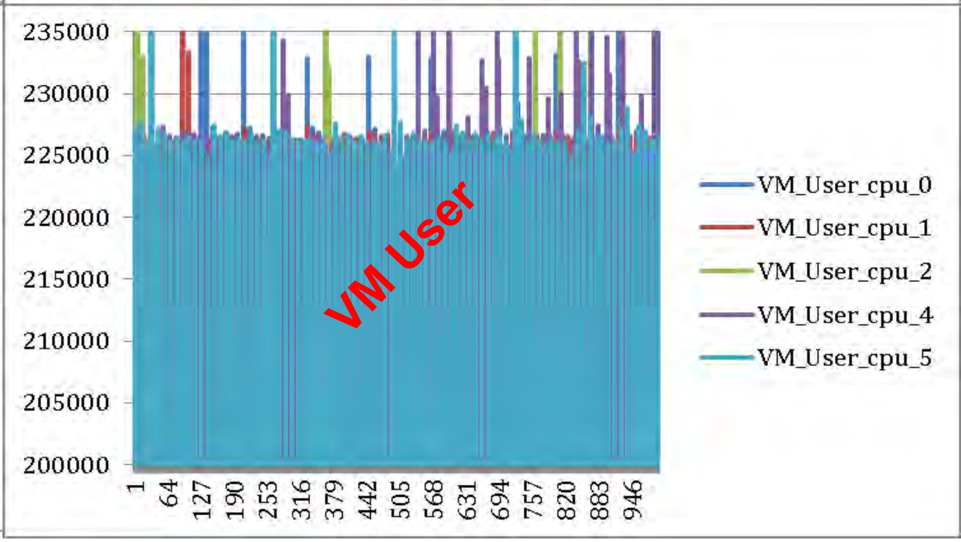
235000
230000
225000
220000
215000
210000
205000
200000
1
60
119
178
237
296
355
414
473
532
591
650
709
768
827
886
945
Kernel_cpu_0
Kernel_cpu_1
Kernel_cpu_2
Kernel_cpu_3
Kernel_cpu_4
Kernel_cpu_5
Bare metal Kernel



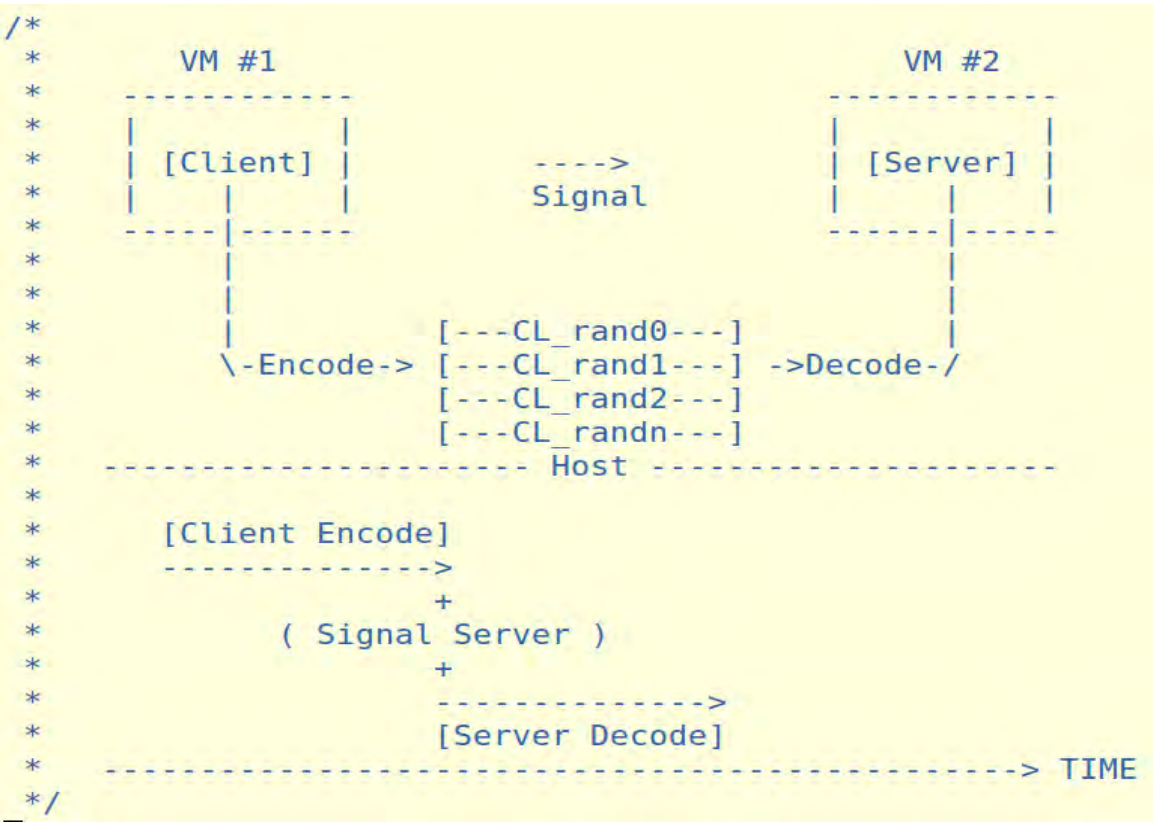
235000
230000
225000
220000
215000
210000
205000
200000
1
64
127
190
253
316
379
442
505
568
631
694
757
820
883
946
VM_Kernel_cpu_0
VM_Kernel_cpu_1
VM_Kernel_cpu_2
VM_Kernel_cpu_3
VM_Kernel_cpu_4
VM_Kernel_cpu_5
VM Kernel



235000
230000
225000
220000
215000
210000
205000
200000
1
57
113
169
225
281
337
393
449
505
561
617
673
729
785
841
897
953
User_cpu_0
User_cpu_1
User_cpu_2
User_cpu_3
User_cpu_4
User_cpu_5
Bare metal User

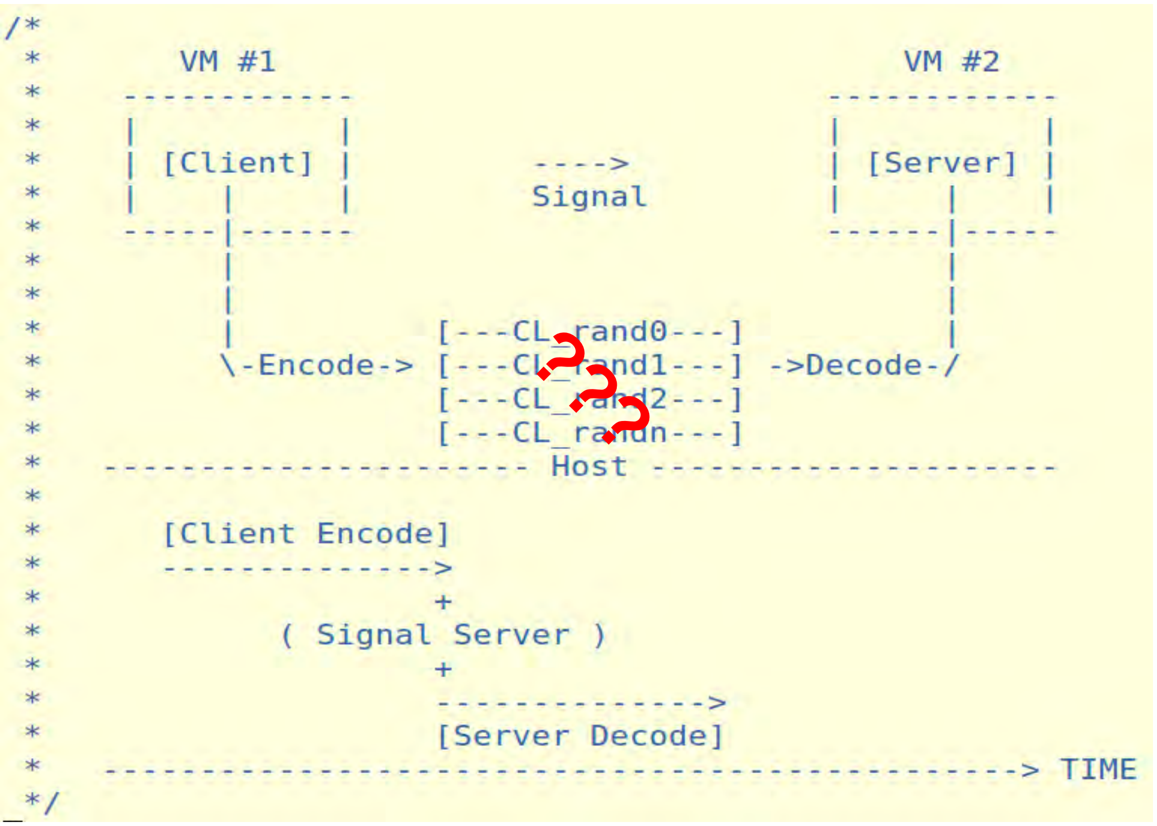


235000
230000
225000
220000
215000
210000
205000
200000
1
64
127
190
253
316
379
442
505
568
631
694
757
820
883
946
VM_User_cpu_0
VM_User_cpu_1
VM_User_cpu_2
VM_User_cpu_4
VM_User_cpu_5
VM User



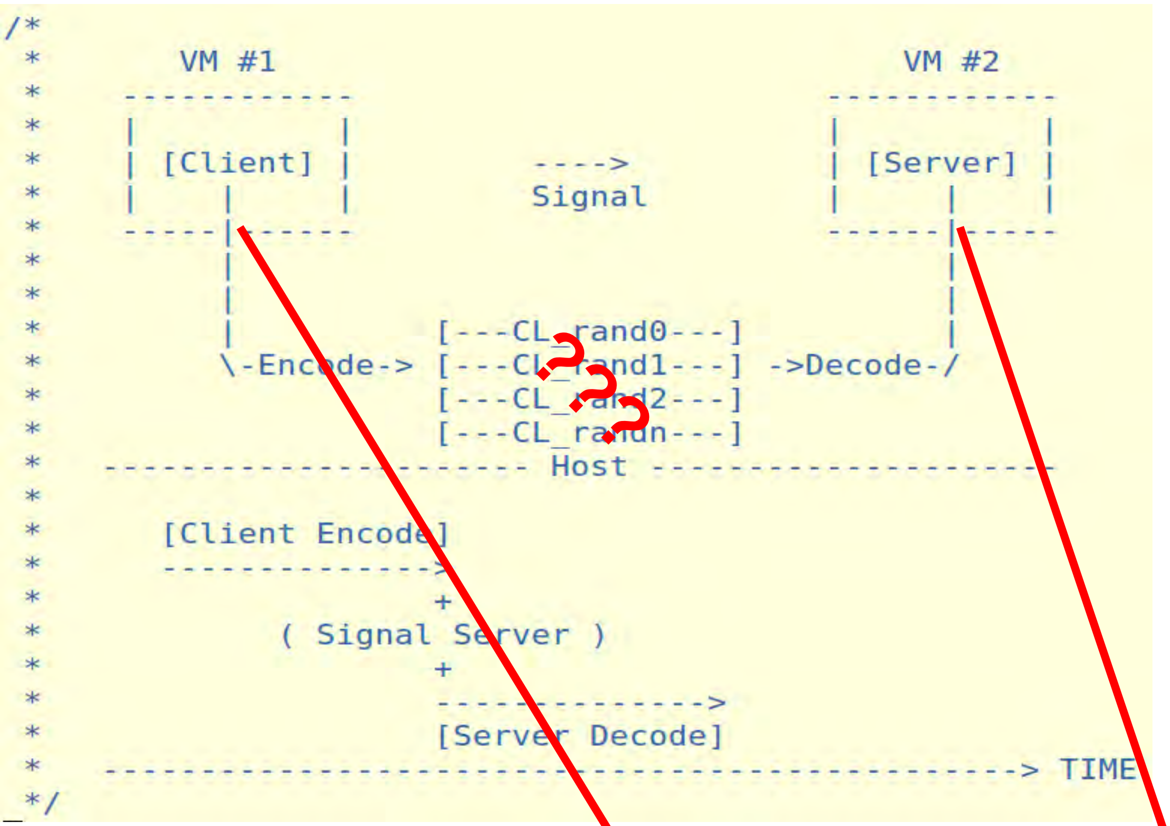
```
/*
 *       VM #1                                  VM #2
 *    ------------                          ------------
 *    |          |                          |          |
 *    | [Client] |          ---->           | [Server] |
 *    |    |     |          Signal          |     |    |
 *    -----|------                          ------|-----
 *         |                                      |
 *         |                                      |
 *         |           [---CL_rand0---]           |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                     [---CL_rand2---]
 *                     [---CL_randn---]
 *   ---------------------- Host ---------------------
 *
 *      [Client Encode]
 *      --------------->
 *                     +
 *            ( Signal Server )
 *                     +
 *                     --------------->
 *                     [Server Decode]
 *   ----------------------------------------------> TIME
 */
```

- Client in **VM#1**, Server in **VM#2**



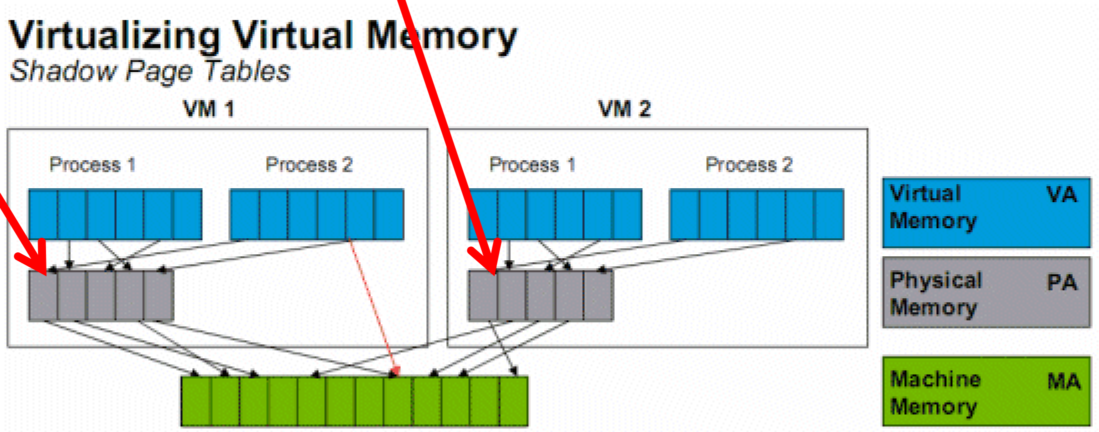
```
/*
 *       VM #1                                          VM #2
 *     ------------                                  ------------
 *     |          |                                  |          |
 *     | [Client] |            ---->                 | [Server] |
 *     |    |     |            Signal                |    |     |
 *     -----|------                                  ------|-----
 *          |                                              |
 *          |                                              |
 *          |             [---CL_rand0---]                 |
 *          \-Encode->    [---CL_rand1---]  ->Decode-/
 *                        [---CL_rand2---]
 *                        [---CL_randn---]
 *    ---------------------- Host ---------------------
 *
 *       [Client Encode]
 *       --------------->
 *                      +
 *             ( Signal Server )
 *                      +
 *                      -------------->
 *                      [Server Decode]
 *    ----------------------------------------------------> TIME
 */
```

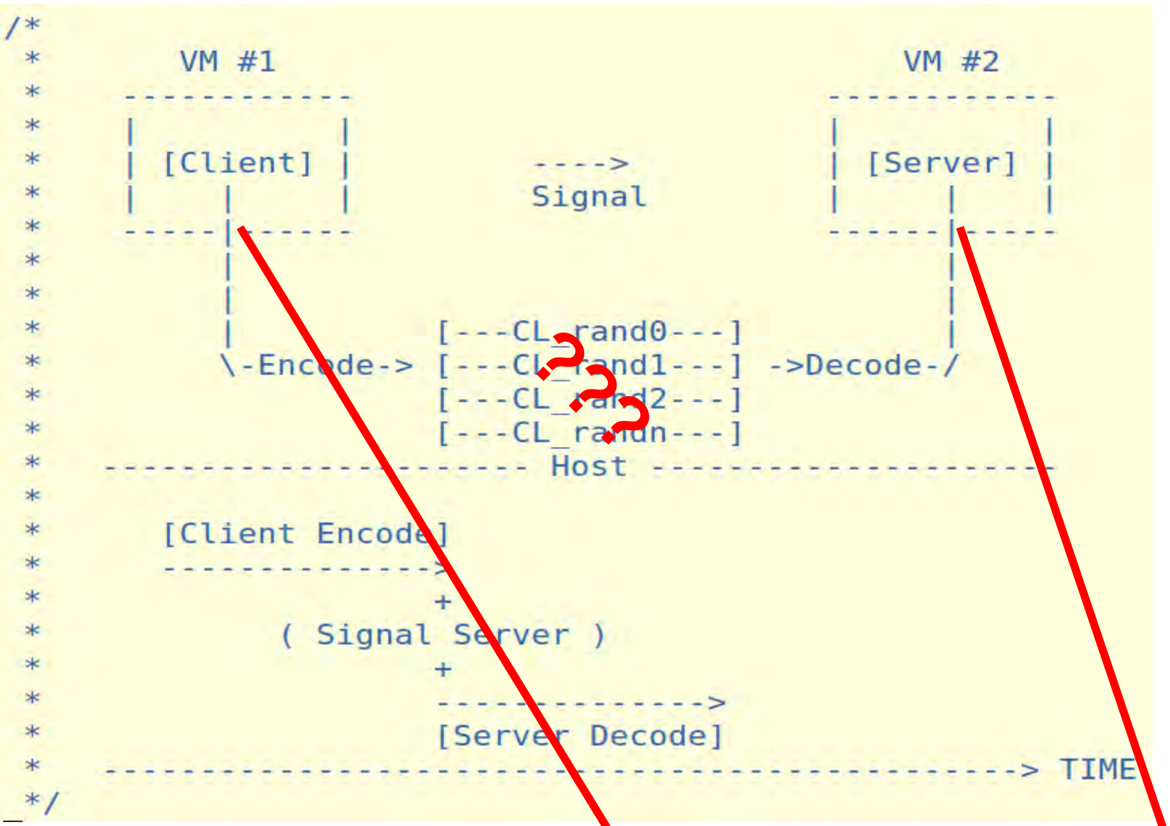
- Client in **VM#1**, Server in **VM#2**
- L2 OR L3 cache are tagged by the physical address but **in a VM the physical address that you see has nothing to do with the real physical address on bare metal** that the cache is using.



```
/*
 *       VM #1                                    VM #2
 *     ------------                           ------------
 *     |          |                           |          |
 *     | [Client] |        ---->              | [Server] |
 *     |    |     |        Signal             |     |    |
 *     -----|------                           ------|-----
 *          |                                       |
 *          |                                       |
 *          |           [---CL_rand0---]            |
 *          \-Encode-> [---CL_rand1---] ->Decode-/
 *                      [---CL_rand2---]
 *                      [---CL_randn---]
 *    ---------------------- Host ---------------------
 *
 *       [Client Encode]
 *       --------------->
 *                      +
 *              ( Signal Server )
 *                      +
 *                      --------------->
 *                      [Server Decode]
 *    ------------------------------------------------> TIME
 */
```

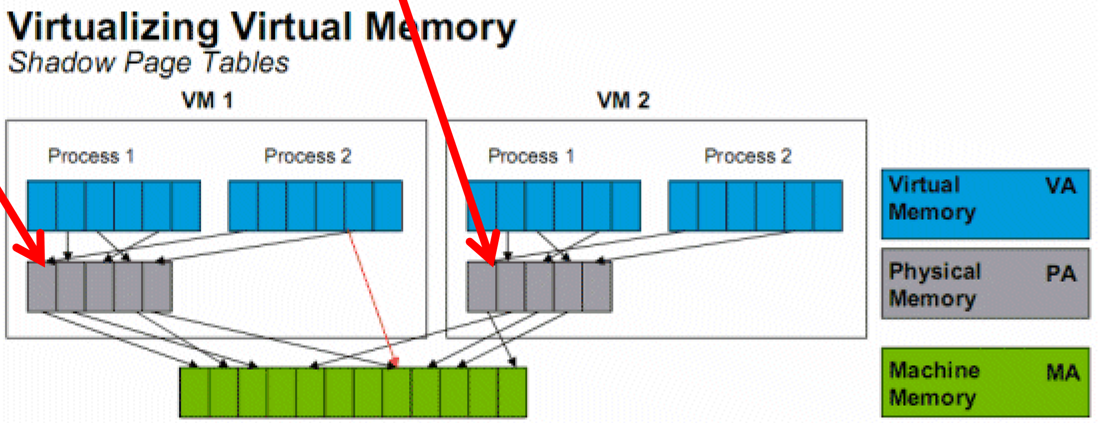
- Client in **VM#1**, Server in **VM#2**
- L2 OR L3 cache are tagged by the physical address but **in a VM the physical address that you see has nothing to do with the real physical address on bare metal** that the cache is using.
- There is another layer of translation





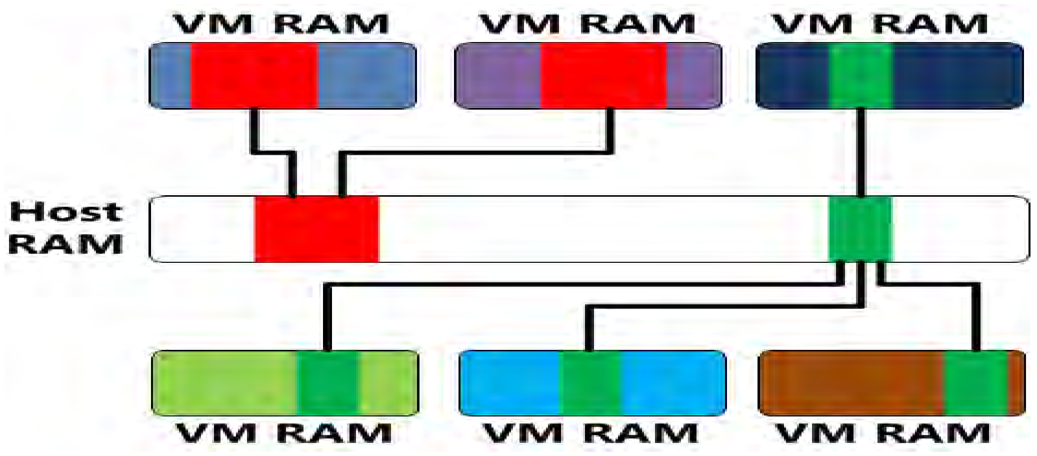
```
/*
 *       VM #1                                         VM #2
 *     ------------                                 ------------
 *     |          |                                 |          |
 *     | [Client] |          ---->                  | [Server] |
 *     |    |     |          Signal                 |    |     |
 *     -----|------                                 -----|------
 *          |                                            |
 *          |                                            |
 *          |           [---CL_rand0---]                 |
 *          \-Encode->  [---CL_rand1---]  ->Decode-/
 *                      [---CL_rand2---]
 *                      [---CL_randn---]
 *   ---------------------- Host ----------------------
 *
 *      [Client Encode]
 *      --------------->
 *                      +
 *              ( Signal Server )
 *                      +
 *                      -------------->
 *                      [Server Decode]
 *   ---------------------------------------------------> TIME
 */
```

- Client in **VM#1**, Server in **VM#2**
- L2 OR L3 cache are tagged by the physical address but **in a VM the physical address that you see has nothing to do with the real physical address on bare metal** that the cache is using.
- There is another layer of translation
- **This is a complex problem to solve**



Page de-duplication

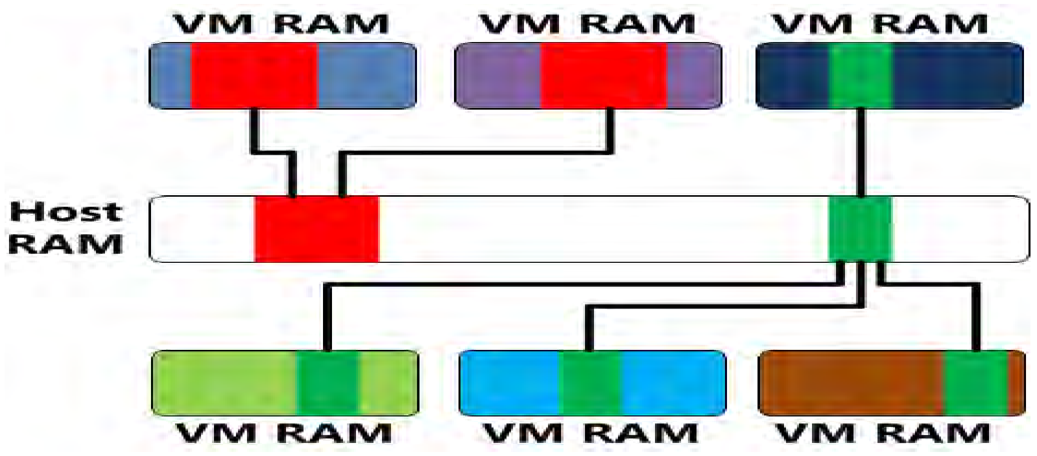
KSM enables the kernel to examine two or more already running programs and compare their memory. If any memory regions or pages are identical, **KSM merge them into a single page physical page on bare-metal host kernel.**



Page de-duplication

KSM enables the kernel to examine two or more already running programs and compare their memory. If any memory regions or pages are identical, **KSM merge them into a single page physical page on bare-metal host kernel.**

If one of the programs wants to modify a shared page KSM kicks in and un-merge it.

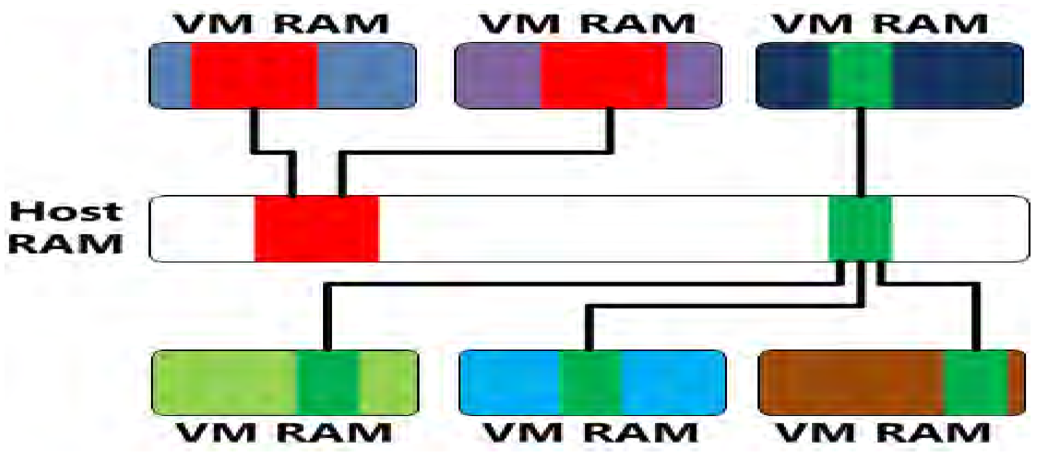


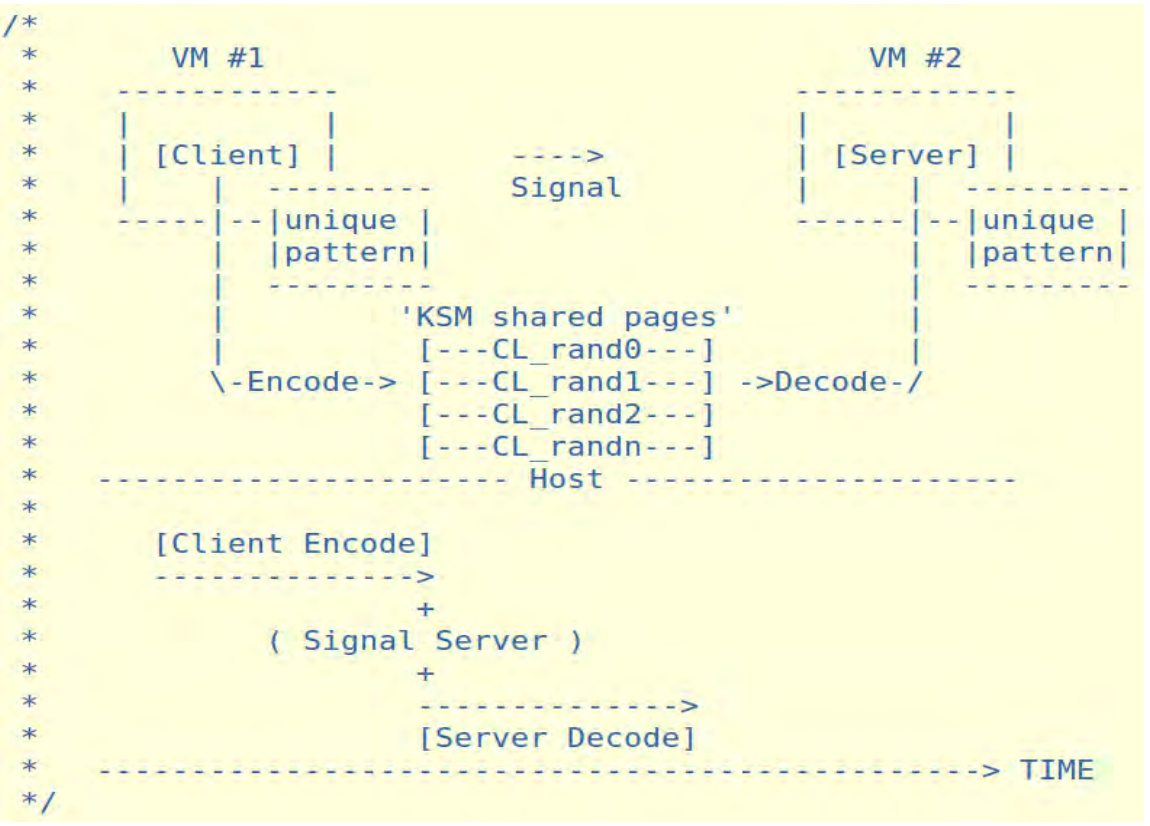
Page de-duplication

KSM enables the kernel to examine two or more already running programs and compare their memory. If any memory regions or pages are identical, **KSM merge them into a single page physical page on bare-metal host kernel.**

If one of the programs wants to modify a shared page KSM kicks in and un-merge it.

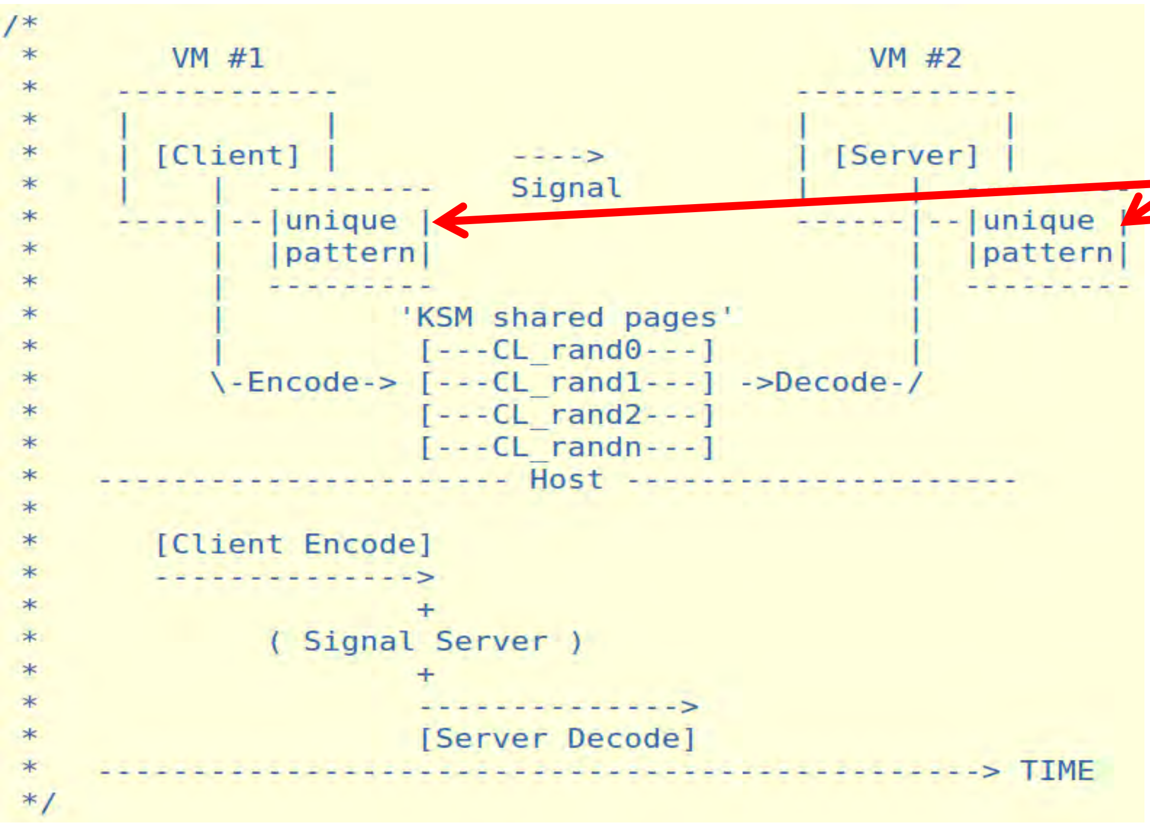
This is useful for virtualization with KVM. Once the guest is running **the contents of the guest operating system image can be shared when guests are running the same operating system or applications.**





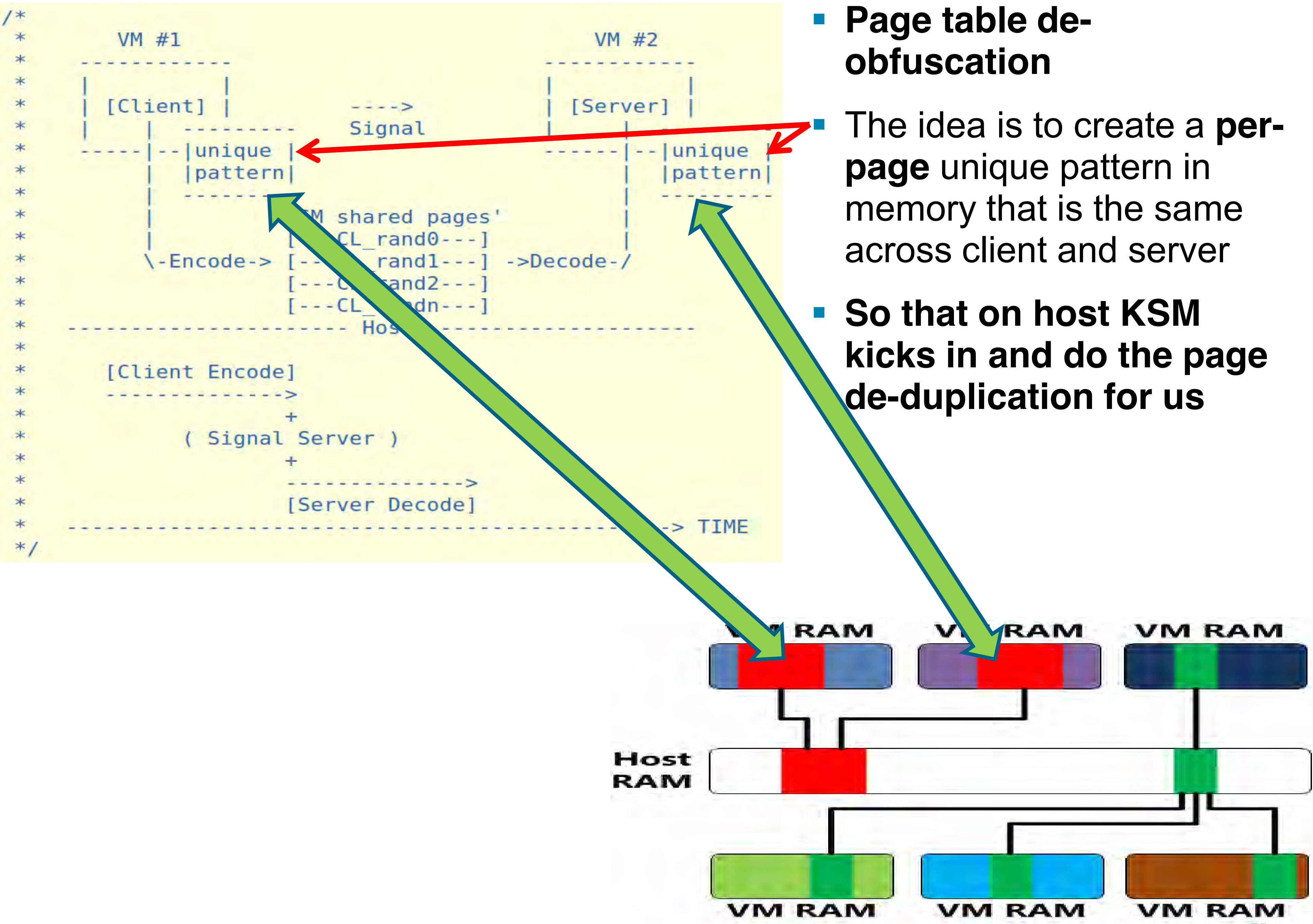
```
/*
 *       VM #1                                VM #2
 *    ------------                        ------------
 *    |          |                        |          |
 *    | [Client] |         ---->          | [Server] |
 *    |    |  ---------    Signal         |     |  ---------
 *    -----|--|unique |                   ------|--|unique |
 *         |  |pattern|                         |  |pattern|
 *         |  ---------                         |  ---------
 *         |         'KSM shared pages'         |
 *         |          [---CL_rand0---]          |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   ---------------------- Host ----------------------
 *
 *      [Client Encode]
 *      -------------->
 *                    +
 *            ( Signal Server )
 *                    +
 *                    -------------->
 *                    [Server Decode]
 *   -----------------------------------------------> TIME
 */
```

- **Page table de-obfuscation**



```
/*
 *       VM #1                                 VM #2
 *    ------------                         ------------
 *    |          |                         |          |
 *    | [Client] |         ---->           | [Server] |
 *    |    |  ---------    Signal          |     |  ---------
 *    -----|--|unique |                    ------|--|unique |
 *         |  |pattern|                          |  |pattern|
 *         |  ---------                          |  ---------
 *         |         'KSM shared pages'          |
 *         |          [---CL_rand0---]           |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   ---------------------- Host ----------------------
 *
 *      [Client Encode]
 *      --------------->
 *                    +
 *            ( Signal Server )
 *                    +
 *                    -------------->
 *                    [Server Decode]
 *   ------------------------------------------------> TIME
 */
```

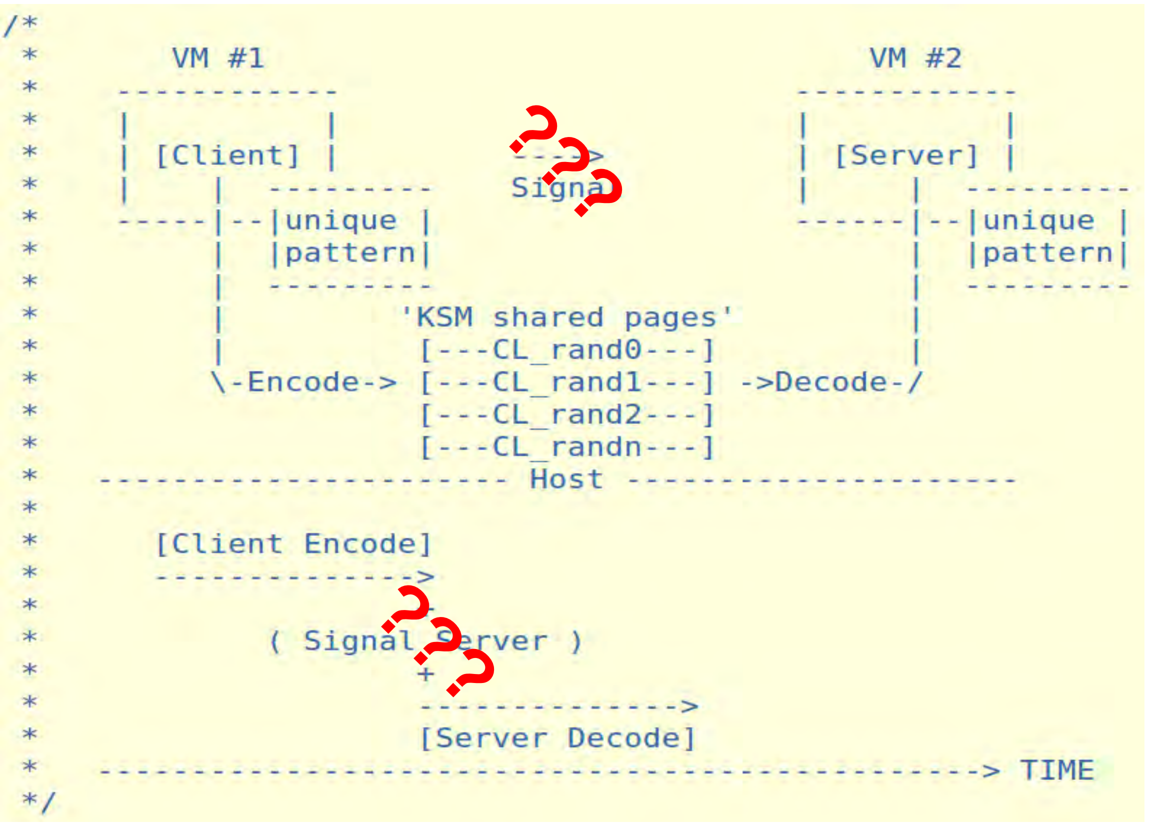
- **Page table de-obfuscation**
- The idea is to create a **per-page** unique pattern in memory that is the same across client and server



```
/*
 *     VM #1                                        VM #2
 *   ------------                                 ------------
 *   |          |                                 |          |
 *   | [Client] |          ---->                  | [Server] |
 *   |    |   ---------    Signal                 |    |   
 *   -----|--|unique  |                           ------|--|unique  |
 *        |  |pattern|                                 |  |pattern|
 *        |   -------                                  |   ---------
 *        |          'VM shared pages'                 |
 *        |          [---CL_rand0---]                  |
 *        \-Encode-> [---CL_rand1---] ->Decode-/
 *                   [---CL_rand2---]
 *                   [---CL_randn---]
 *  ---------------------- Host ----------------------
 *
 *     [Client Encode]
 *     -------------->
 *                   +
 *           ( Signal Server )
 *                   +
 *                   -------------->
 *                   [Server Decode]
 *  ---------------------------------------------> TIME
 */
```

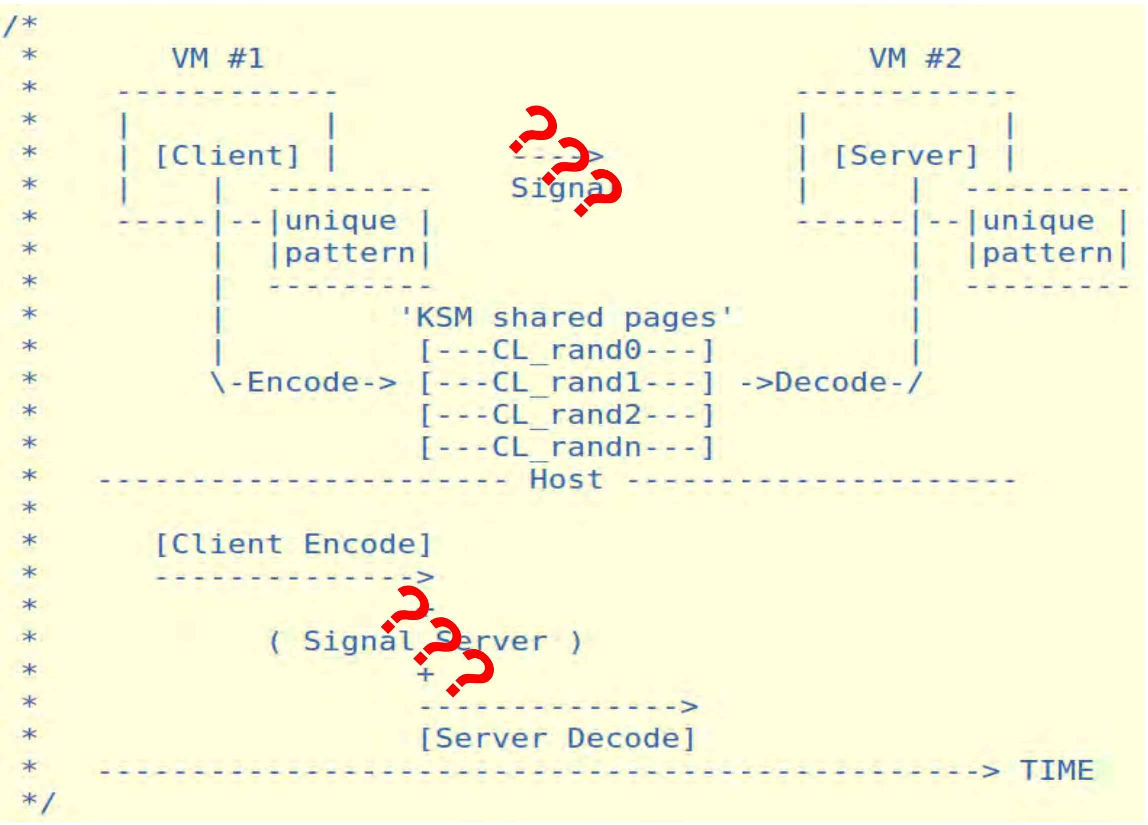
- **Page table de-obfuscation**
- The idea is to create a **per-page** unique pattern in memory that is the same across client and server
- **So that on host KSM kicks in and do the page de-duplication for us**

```
/*
 *       VM #1                                VM #2
 *    ------------                        ------------
 *    |          |                        |          |
 *    | [Client] |         ---->          | [Server] |
 *    |    |  ---------    Signal         |     |  ---------
 *    -----|--|unique |                   ------|--|unique |
 *         |  |pattern|                         |  |pattern|
 *         |  ---------                         |  ---------
 *         |         'KSM shared pages'         |
 *         |          [---CL_rand0---]          |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   ---------------------- Host ---------------------
 *
 *      [Client Encode]
 *      -------------->
 *                    +
 *            ( Signal Server )
 *                    +
 *                    -------------->
 *                    [Server Decode]
 *   -----------------------------------------------> TIME
 */
```



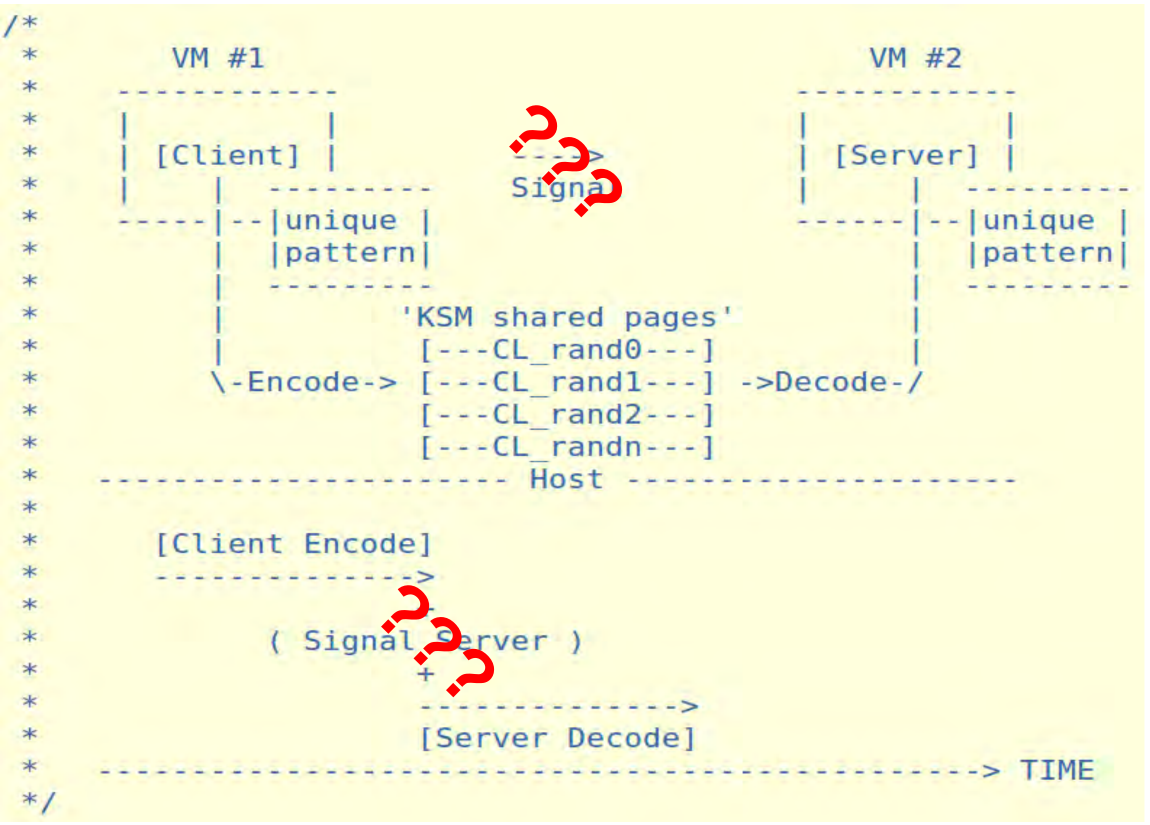
```
/*
 *       VM #1                                VM #2
 *    ------------                        ------------
 *    |          |                        |          |
 *    | [Client] |         ---->          | [Server] |
 *    |    |  ---------    Signal         |     |  ---------
 *    -----|--|unique |                   ------|--|unique |
 *         |  |pattern|                         |  |pattern|
 *         |  ---------                         |  ---------
 *         |         'KSM shared pages'         |
 *         |          [---CL_rand0---]          |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   ---------------------- Host ---------------------
 *
 *      [Client Encode]
 *      -------------->
 *                    +
 *            ( Signal Server )
 *                    +
 *                    --------------->
 *                    [Server Decode]
 *   ------------------------------------------------> TIME
 */
```

- **There is no synchronization primitive across processes running in different VM ???**



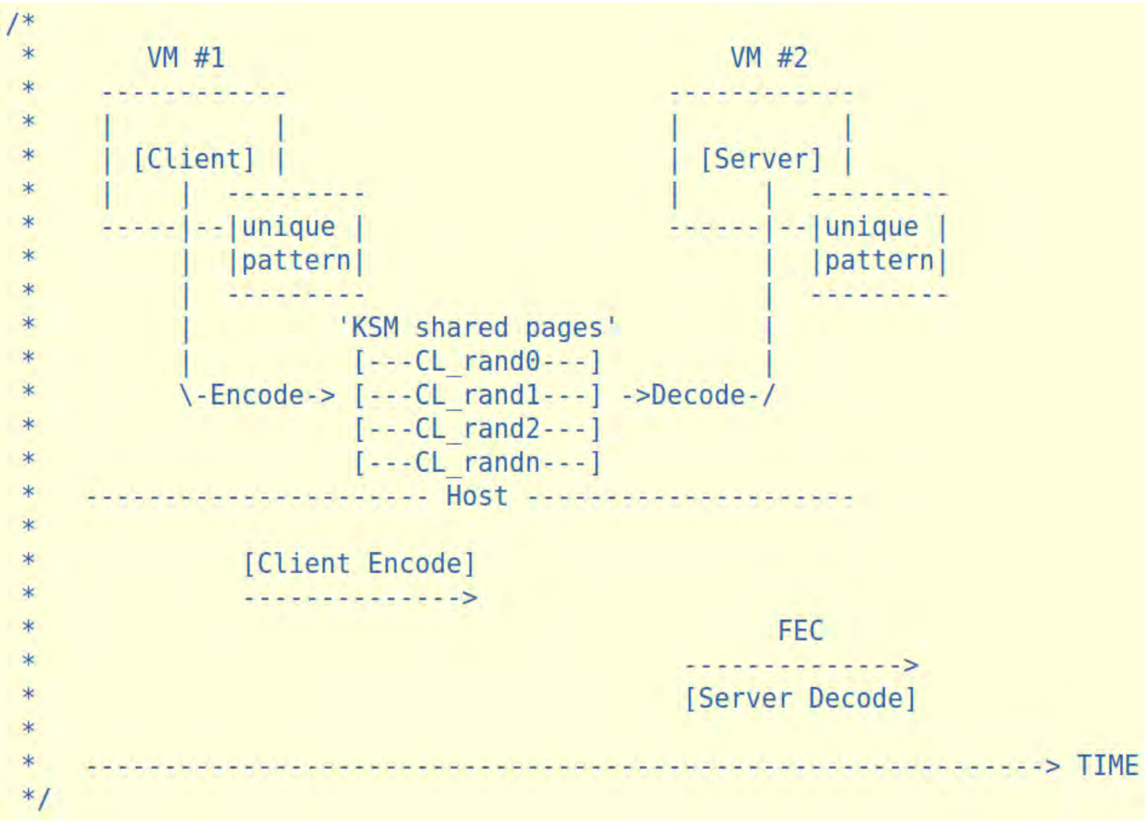
```
/*
 *        VM #1                                      VM #2
 *    -----------                                -----------
 *    |         |                                |         |
 *    | [Client] |            ---->              | [Server] |
 *    |    |    ---------     Signal             |      |    ---------
 *    -----|--|unique |                          ------|--|unique  |
 *         |  |pattern|                                |  |pattern|
 *         |  ---------                                |  ---------
 *         |          'KSM shared pages'               |
 *         |           [---CL_rand0---]                |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                     [---CL_rand2---]
 *                     [---CL_randn---]
 *   --------------------- Host ---------------------
 *
 *    [Client Encode]
 *    -------------->
 *                   +
 *         ( Signal Server )
 *                   +
 *                   -------------->
 *                     [Server Decode]
 *   ---------------------------------------------> TIME
 */
```

- **There is no synchronization primitive across processes running in different VM ???**
- *In reality there is mechanism to do that (EX ivshmem) but this is not enabled in production env*



```
/*
 *        VM #1                                    VM #2
 *     ------------                            ------------
 *     |          |                            |          |
 *     | [Client] |          ---->             | [Server] |
 *     |    |    ---------   Signa             |     |     ---------
 *     -----|--|unique |                       ------|--|unique |
 *          |  |pattern|                             |  |pattern|
 *          |   ---------                            |   ---------
 *          |          'KSM shared pages'            |
 *          |           [---CL_rand0---]             |
 *          \-Encode-> [---CL_rand1---] ->Decode-/
 *                      [---CL_rand2---]
 *                      [---CL_randn---]
 *    ---------------------- Host --------------------
 *
 *     [Client Encode]
 *     --------------->
 *                      +
 *           ( Signal Server )
 *                      +
 *                      -------------->
 *                      [Server Decode]
 *    ------------------------------------------> TIME
 */
```

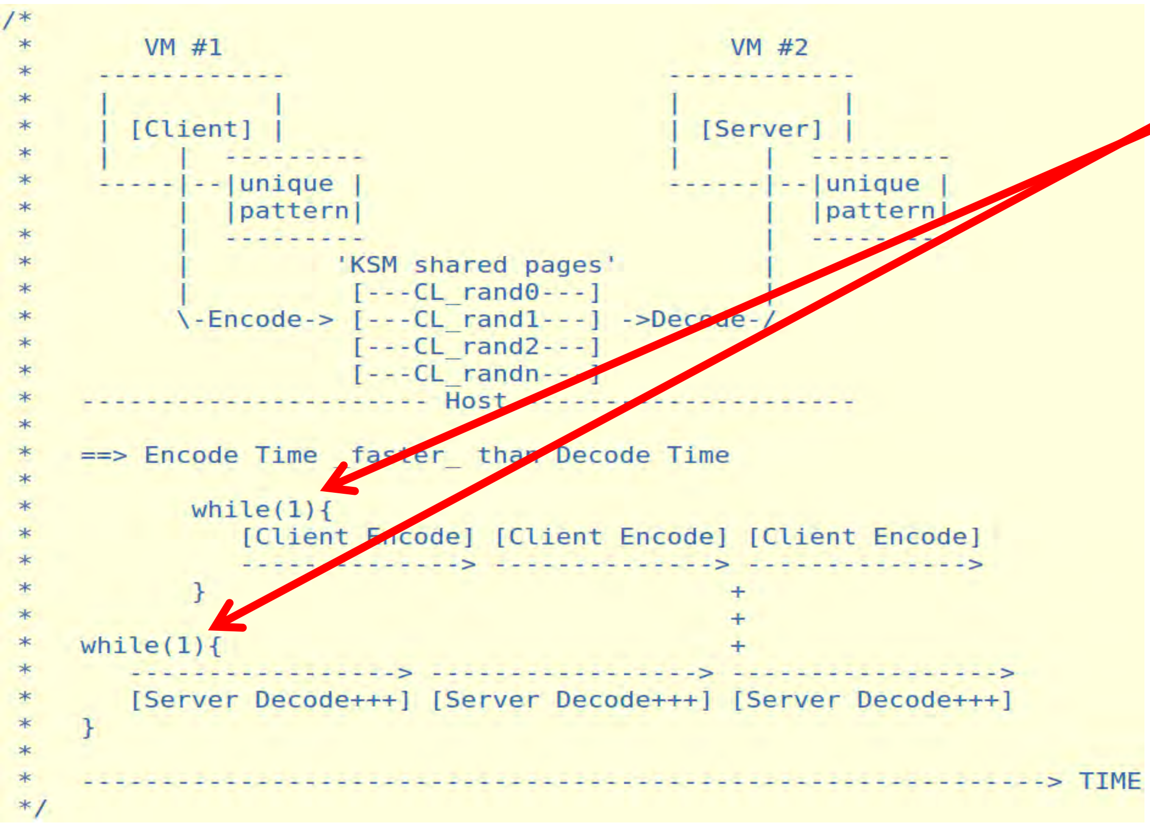
- **There is no synchronization primitive across processes running in different VM ???**
- *In reality there is mechanism to do that (EX ivshmem) but this is not enabled in production env*
- **We need something to replace the mutex**



```
/*
 *       VM #1                                 VM #2
 *    ------------                         ------------
 *    |          |                         |          |
 *    | [Client] |                         | [Server] |
 *    |    |  ---------                    |     |  ---------
 *    -----|--|unique |                    ------|--|unique |
 *         |  |pattern|                          |  |pattern|
 *         |  ---------                          |  ---------
 *         |         'KSM shared pages'          |
 *         |          [---CL_rand0---]           |
 *         \-Encode-> [---CL_rand1---]  ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   ---------------------- Host  ---------------------
 *
 *              [Client Encode]
 *              -------------->
 *                                                FEC
 *                                          -------------->
 *                                          [Server Decode]
 *
 *   --------------------------------------------------------------> TIME
 */
```

Option #1

- Forget about the synchronization aspect and hope for the best
- With error correction we can achieve some data transmission.
- Very low bit rates
- CPU consumption is low



```
/*
 *      VM #1                                     VM #2
 *    ------------                            ------------
 *    |          |                            |          |
 *    | [Client] |                            | [Server] |
 *    |    |  ---------                       |     |  ---------
 *    -----|--|unique  |                      ------|--|unique  |
 *         |  |pattern|                             |  |pattern|
 *         |  ---------                             |  ---------
 *         |         'KSM shared pages'             |
 *         |          [---CL_rand0---]              |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   ---------------------- Host ----------------------
 *
 *   ==> Encode Time  _faster_  than Decode Time
 *
 *          while(1){
 *             [Client Encode] [Client Encode] [Client Encode]
 *             ---------------> ---------------> --------------->
 *          }                                  +
 *                                             +
 *   while(1){                                 +
 *      -----------------> -----------------> ------------------>
 *      [Server Decode+++] [Server Decode+++] [Server Decode+++]
 *   }
 *
 *   -------------------------------------------------------------------> TIME
 */
```

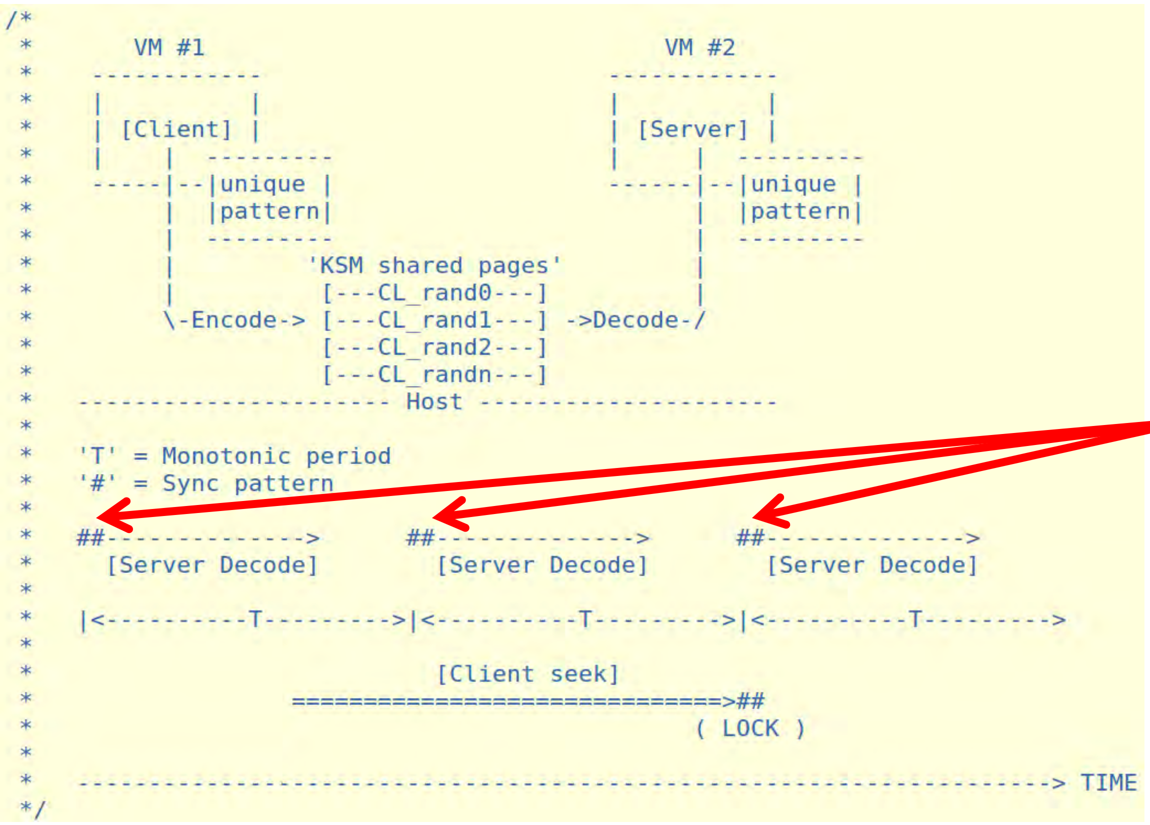
Option #2

- Busy loop on each side
- Client faster than Server
- At some point there will be an overlap and the server will pickup the signal
- CPU consumption is **High**
- OK bit rates
- **We want <1% CPU usage to remain undetected.**

```
/*
 *        VM #1                                 VM #2
 *     ------------                          ------------
 *     |          |                          |          |
 *     |  [Client] |                         |  [Server] |
 *     |    |  ---------                     |     |  ---------
 *     -----|--|unique |                     ------|--|unique |
 *          |  |pattern|                           |  |pattern|
 *          |  ---------                           |  ---------
 *          |          'KSM shared pages'          |
 *          |           [---CL_rand0---]           |
 *          \-Encode-> [---CL_rand1---] ->Decode-/
 *                      [---CL_rand2---]
 *                      [---CL_randn---]
 *    ---------------------- Host ---------------------
 *
 *    'T' = Monotonic period
 *    '#' = Sync pattern
 *
 *    ##-------------->       ##-------------->       ##-------------->
 *      [Server Decode]         [Server Decode]         [Server Decode]
 *
 *    |<----------T--------->|<----------T--------->|<----------T--------->
 *
 *                              [Client seek]
 *                    ================================>##
 *                                                ( LOCK )
 *
 *    ----------------------------------------------------------------------> TIME
 */
```

Option #3

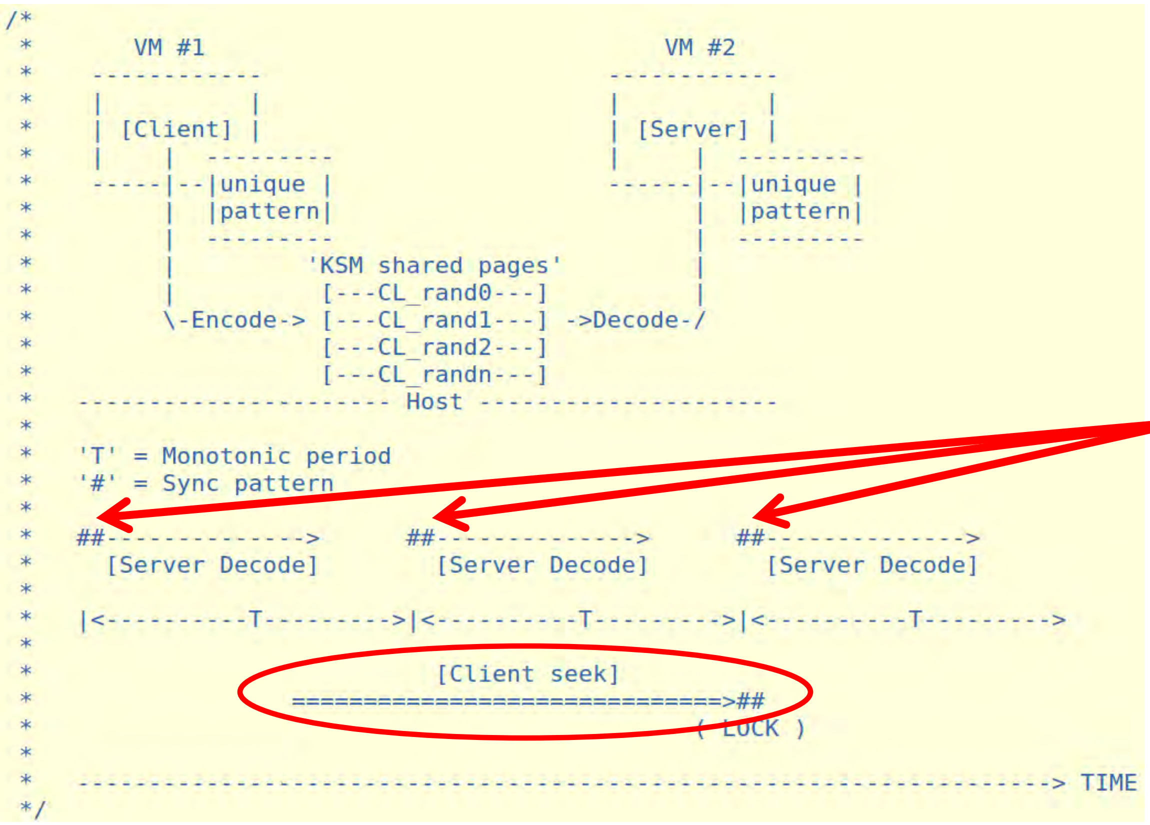
- Define a common period 'T'
- Client-Server lock into phase



```
/*
 *        VM #1                                  VM #2
 *      ------------                          ------------
 *      |          |                          |          |
 *      | [Client] |                          | [Server] |
 *      |    |   ---------                    |     |   ---------
 *      -----|--|unique |                     ------|--|unique |
 *           |  |pattern|                           |  |pattern|
 *           |   ---------                          |   ---------
 *           |          'KSM shared pages'          |
 *           |           [---CL_rand0---]           |
 *           \-Encode-> [---CL_rand1---] ->Decode-/
 *                       [---CL_rand2---]
 *                       [---CL_randn---]
 *     ---------------------- Host ---------------------
 *
 *     'T' = Monotonic period
 *     '#' = Sync pattern
 *
 *     ##-------------->       ##-------------->       ##-------------->
 *       [Server Decode]         [Server Decode]         [Server Decode]
 *
 *     |<---------T--------->|<---------T--------->|<---------T--------->
 *
 *                         [Client seek]
 *               ================================>##
 *                                            ( LOCK )
 *
 *     ----------------------------------------------------------------------> TIME
 */
```

Option #3

- Define a common period 'T'
- Client-Server lock into phase
- Server sends a sync pattern



```
/*
 *      VM #1                                VM #2
 *    ------------                         ------------
 *    |          |                         |          |
 *    | [Client] |                         | [Server] |
 *    |    |   ---------                   |     |   ---------
 *    -----|--|unique  |                   ------|--|unique  |
 *         |  |pattern|                        |  |pattern|
 *         |   ---------                       |   ---------
 *         |          'KSM shared pages'       |
 *         |           [---CL_rand0---]        |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                     [---CL_rand2---]
 *                     [---CL_randn---]
 *   ---------------------- Host ---------------------
 *
 *   'T' = Monotonic period
 *   '#' = Sync pattern
 *
 *   ##-------------->      ##-------------->      ##-------------->
 *     [Server Decode]        [Server Decode]        [Server Decode]
 *
 *   |<----------T--------->|<----------T--------->|<----------T--------->
 *
 *                        [Client seek]
 *                 ================================>##
 *                                              ( LOCK )
 *
 *   ------------------------------------------------------------------> TIME
 */
```

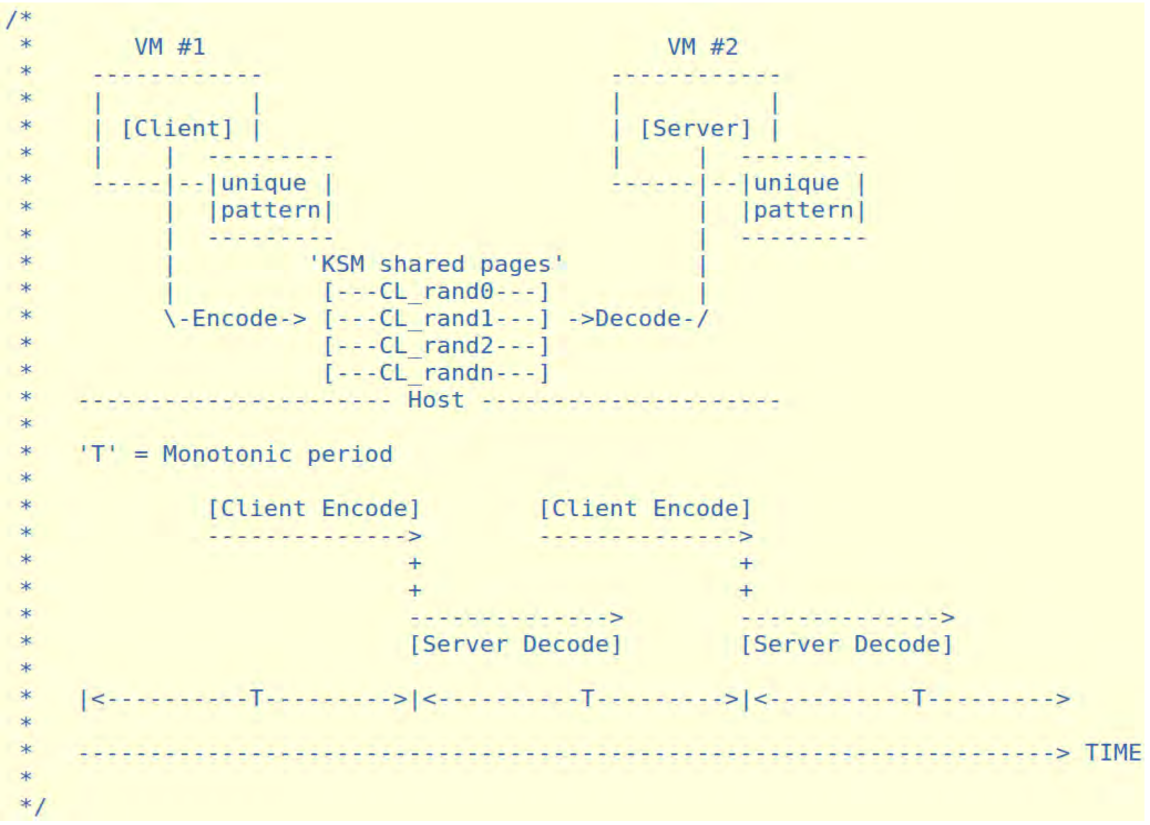
Option #3

- Define a common period 'T'
- Client-Server lock into phase
- Server sends a sync pattern
- Client sweep over the period in search for the sync

```
/*
 *       VM #1                                 VM #2
 *    ------------                         ------------
 *    |          |                         |          |
 *    | [Client] |                         | [Server] |
 *    |    |  ---------                    |     |  ---------
 *    -----|--|unique |                    ------|--|unique |
 *         |  |pattern|                          |  |pattern|
 *         |  ---------                          |  ---------
 *         |         'KSM shared pages'          |
 *         |          [---CL_rand0---]           |
 *         \-Encode-> [---CL_rand1---]  ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   ---------------------- Host ----------------------
 *
 *   'T' = Monotonic period
 *
 *            [Client Encode]         [Client Encode]
 *            -------------->         -------------->
 *                          +                       +
 *                          +                       +
 *                          -------------->         -------------->
 *                          [Server Decode]         [Server Decode]
 *
 *   |<----------T--------->|<----------T--------->|<----------T--------->
 *
 *   ---------------------------------------------------------------------> TIME
 *
 */
```

Option #3

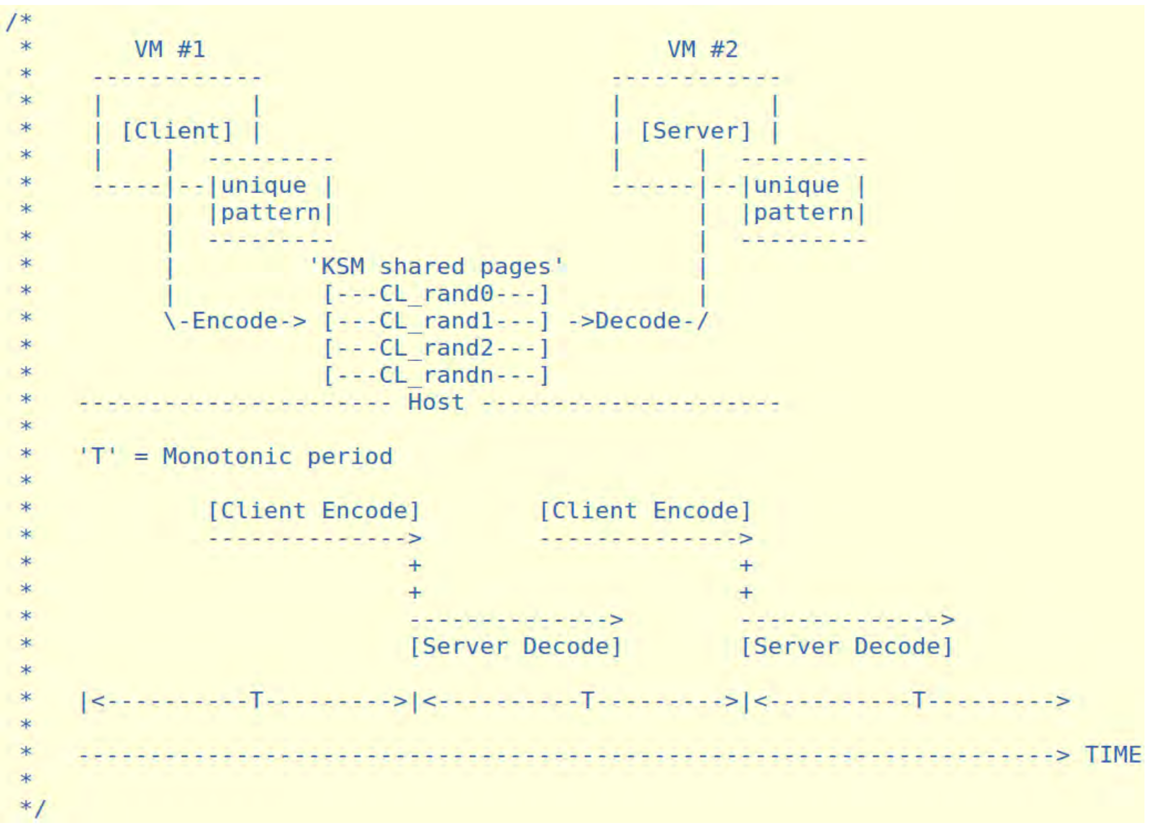
- Once the sync is found the phase is adjusted are we are ready for transmission



```
/*
 *       VM #1                                 VM #2
 *    ------------                         ------------
 *    |          |                         |          |
 *    | [Client] |                         | [Server] |
 *    |    |  ---------                    |     |  ---------
 *    -----|--|unique |                    ------|--|unique |
 *         |  |pattern|                          |  |pattern|
 *         |  ---------                          |  ---------
 *         |         'KSM shared pages'          |
 *         |          [---CL_rand0---]           |
 *         \-Encode-> [---CL_rand1---]  ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   ---------------------- Host  ---------------------
 *
 *   'T' = Monotonic period
 *
 *            [Client Encode]         [Client Encode]
 *            -------------->         -------------->
 *                          +                       +
 *                          +                       +
 *                          -------------->         -------------->
 *                          [Server Decode]         [Server Decode]
 *
 *   |<----------T--------->|<----------T--------->|<----------T--------->
 *
 *   ---------------------------------------------------------------------> TIME
 *
 */
```

Option #3

- Once the sync is found the phase is adjusted are we are ready for transmission
- For that to work we need a **monotonic pulse**



```
/*
 *      VM #1                                VM #2
 *    ------------                         ------------
 *    |          |                         |          |
 *    | [Client] |                         | [Server] |
 *    |    |  ---------                    |     |  ---------
 *    -----|--|unique |                    ------|--|unique |
 *         |  |pattern|                          |  |pattern|
 *         |  ---------                          |  ---------
 *         |          'KSM shared pages'         |
 *         |           [---CL_rand0---]          |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                     [---CL_rand2---]
 *                     [---CL_randn---]
 *   ---------------------- Host ---------------------
 *
 *   'T' = Monotonic period
 *
 *            [Client Encode]         [Client Encode]
 *            --------------->        --------------->
 *                           +                       +
 *                           +                       +
 *                           --------------->        --------------->
 *                           [Server Decode]         [Server Decode]
 *
 *   |<----------T--------->|<----------T--------->|<----------T--------->
 *
 *   ----------------------------------------------------------------------> TIME
 *
 */
```

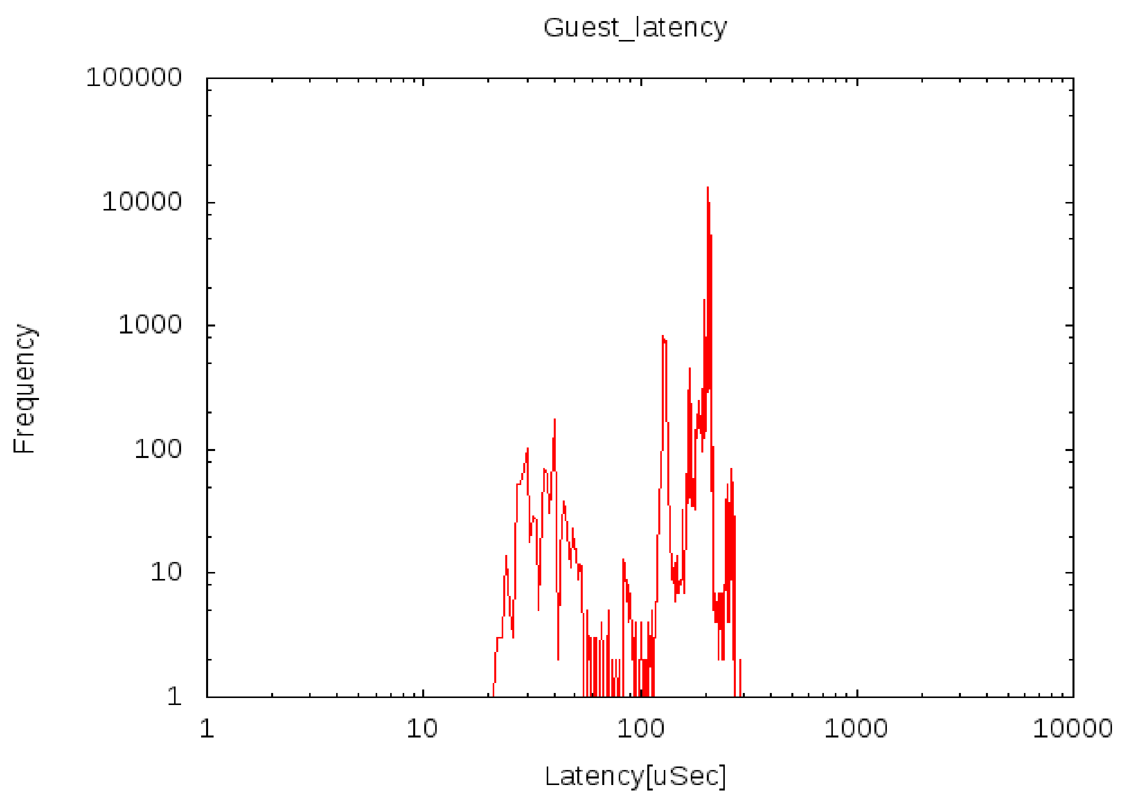
Option #3

- Once the sync is found the phase is adjusted are we are ready for transmission.
- For that to work we need a **monotonic pulse**
- Some jitter but not too much (*Lots of noise in VMs ➔ data evaporates out of the cache very quickly*)

- How to achieve a monotonic pulse?

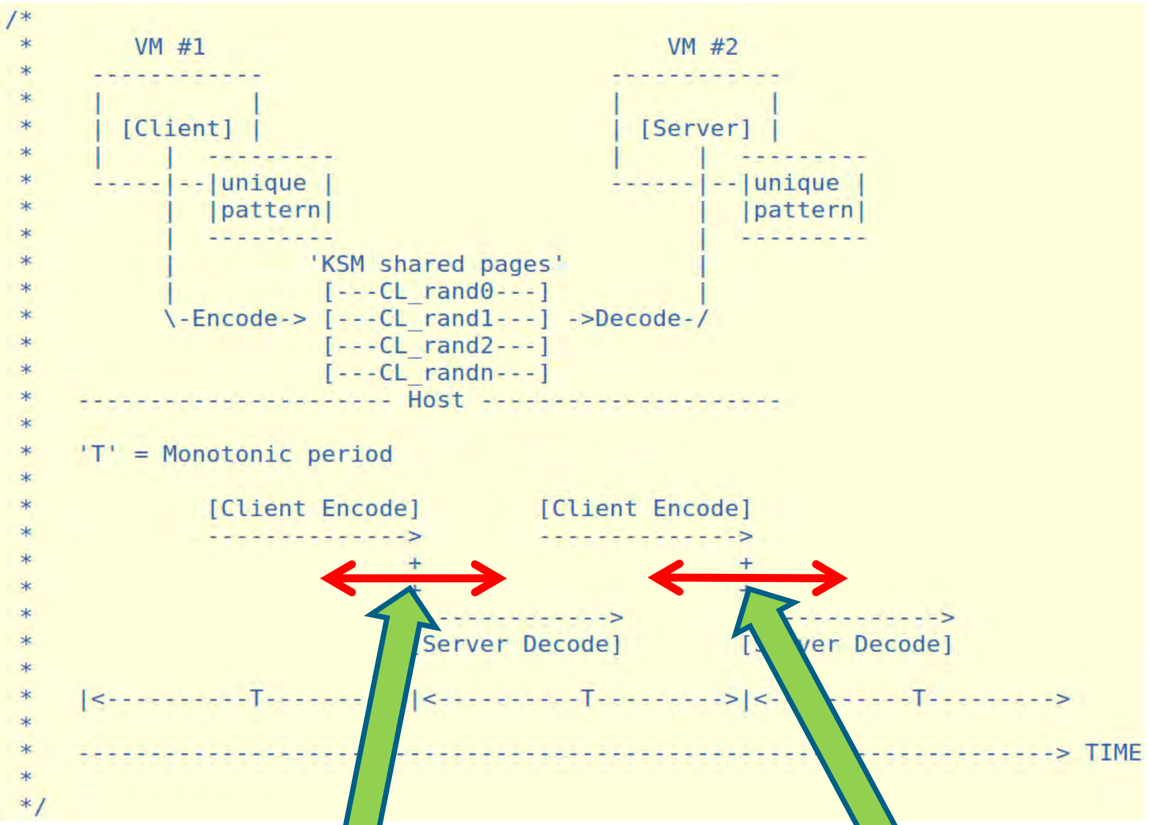
- How to achieve a monotonic pulse?
- Timers

- How to achieve a monotonic pulse?
- Timers
- Why timers?
- We need to sleep ➔ Avoid detection (< 1% CPU usage)

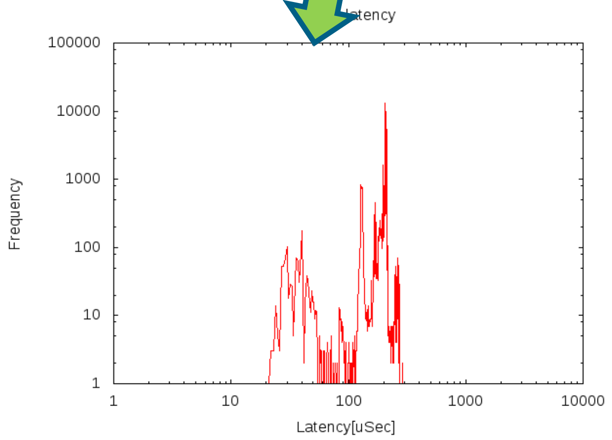


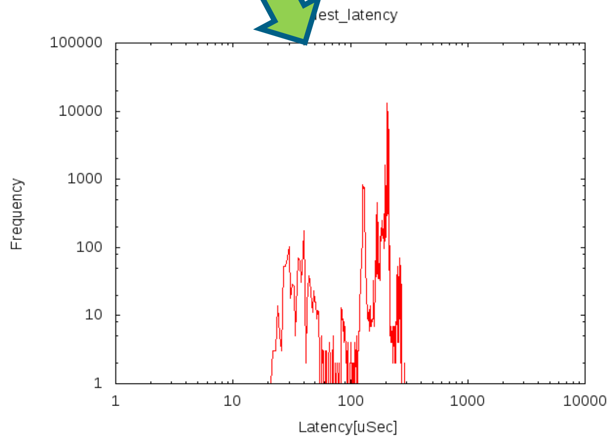
- How to achieve a monotonic pulse?
- Timers
- Why timers?
- We need to sleep ➔ Avoid detection (< 1% CPU usage)

- Jitter comes from both VM



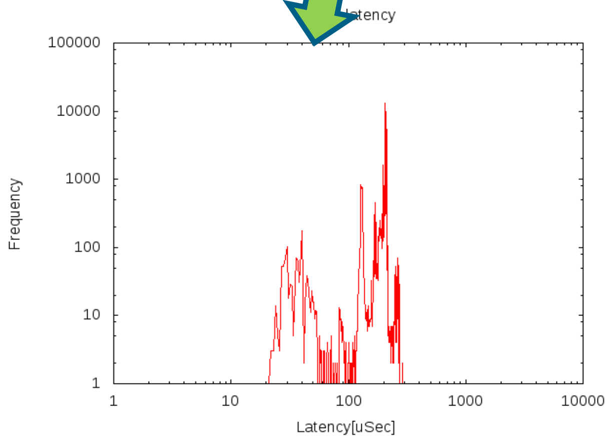
```
/*
 *        VM #1                                  VM #2
 *     ------------                          ------------
 *     |          |                          |          |
 *     | [Client] |                          | [Server] |
 *     |    |   ---------                    |      |   ---------
 *     -----|--|unique |                     ------|--|unique |
 *          |  |pattern|                            |  |pattern|
 *          |   ---------                           |   ---------
 *          |          'KSM shared pages'           |
 *          |           [---CL_rand0---]            |
 *          \-Encode-> [---CL_rand1---] ->Decode-/
 *                      [---CL_rand2---]
 *                      [---CL_randn---]
 *    ---------------------- Host ---------------------
 *
 *    'T' = Monotonic period
 *
 *            [Client Encode]         [Client Encode]
 *            --------------->        --------------->
 *                           +                       +
 *                           +                       +
 *                            ------------->          ------------->
 *                           [Server Decode]         [Server Decode]
 *
 *    |<----------T--------->|<----------T--------->|<----------T--------->
 *
 *    -----------------------------------------------------------------------> TIME
 *
 */
```

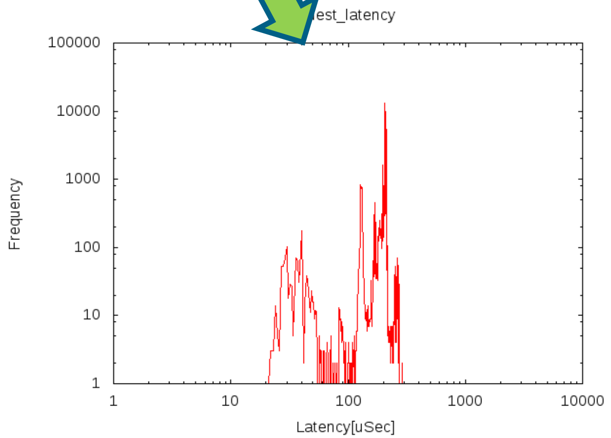




```
/*
 *       VM #1                                VM #2
 *    ------------                         ------------
 *    |          |                         |          |
 *    | [Client] |                         | [Server] |
 *    |    |  ---------                    |     |  ---------
 *    -----|--|unique |                    ------|--|unique |
 *         |  |pattern|                          |  |pattern|
 *         |  ---------                          |  ---------
 *         |         'KSM shared pages'          |
 *         |          [---CL_rand0---]           |
 *         \-Encode-> [---CL_rand1---]  ->Decode-/
 *                    [---CL_rand2---]
 *                    [---CL_randn---]
 *   ---------------------- Host ----------------------
 *
 *   'T' = Monotonic period
 *
 *            [Client Encode]        [Client Encode]
 *            --------------->       --------------->
 *                           +                      +
 *                           +                      +
 *                           --------------->       --------------->
 *                           [Server Decode]        [Server Decode]
 *
 *   |<----------T--------->|<----------T--------->|<----------T--------->
 *
 *   ---------------------------------------------------------------------> TIME
 *
 */
```

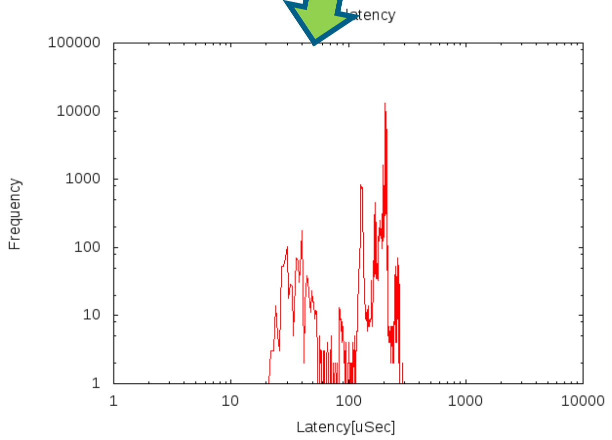
- Jitter comes from both VM
- **Too much jitter**

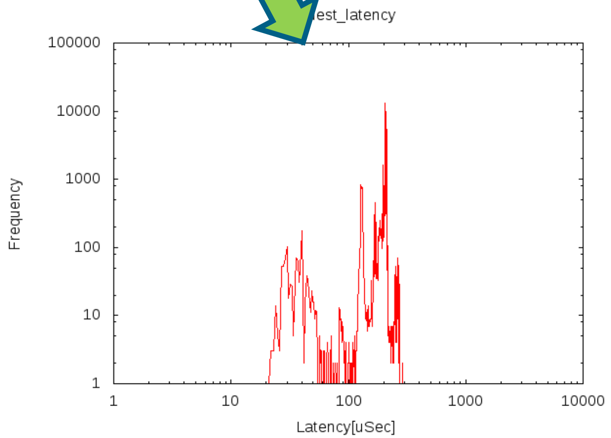


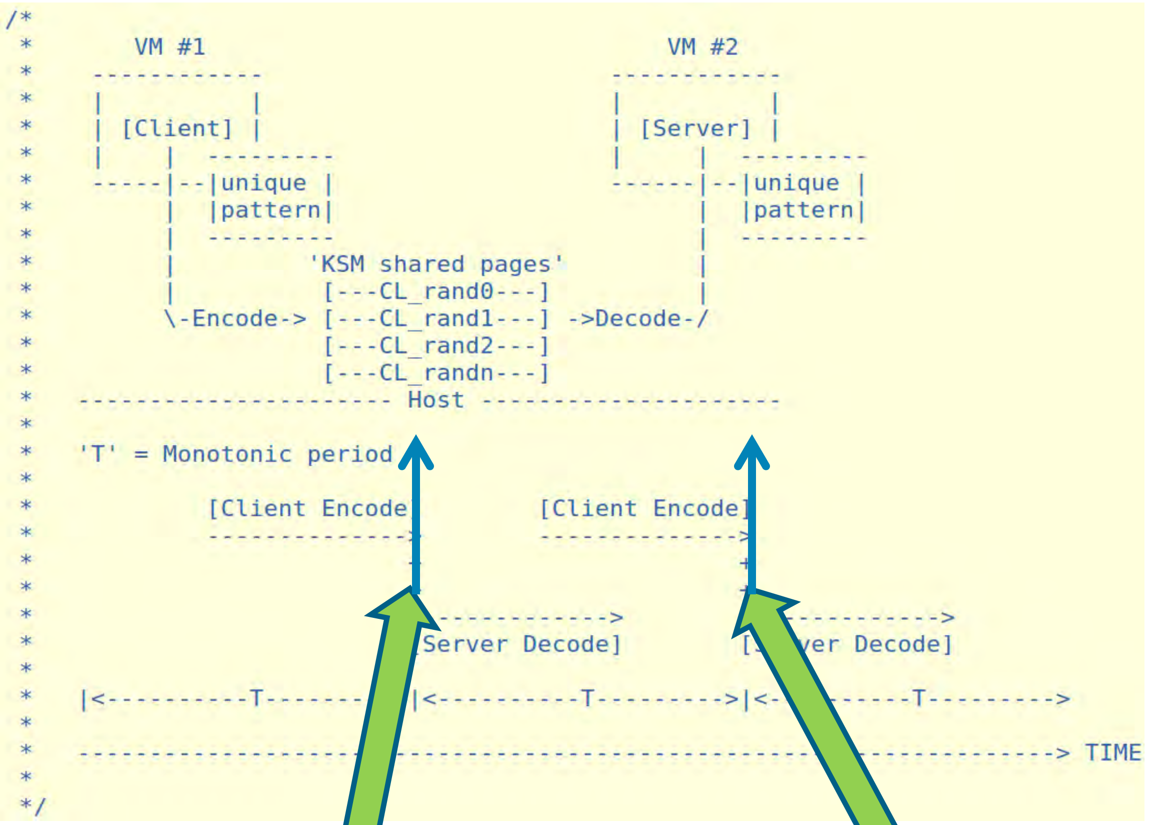


```
/*
 *       VM #1                                 VM #2
 *     ------------                          ------------
 *     |          |                          |          |
 *     | [Client] |                          | [Server] |
 *     |    |   ---------                    |      |   ---------
 *     -----|--|unique  |                    ------|--|unique  |
 *          |  |pattern|                            |  |pattern|
 *          |   ---------                           |   ---------
 *          |         'KSM shared pages'            |
 *          |          [---CL_rand0---]             |
 *          \-Encode-> [---CL_rand1---] ->Decode-/
 *                     [---CL_rand2---]
 *                     [---CL_randn---]
 *    ---------------------- Host --------------------
 *
 *    'T' = Monotonic period
 *
 *             [Client Encode]          [Client Encode]
 *             --------------->         --------------->
 *                            +                        +
 *                            +                        +
 *                            -------------->          -------------->
 *                            [Server Decode]          [Server Decode]
 *
 *    |<----------T--------->|<----------T---------->|<----------T---------->
 *
 *    ----------------------------------------------------------------------> TIME
 *
 */
```

- The idea here is to do padding up to some value above the maximum jitter

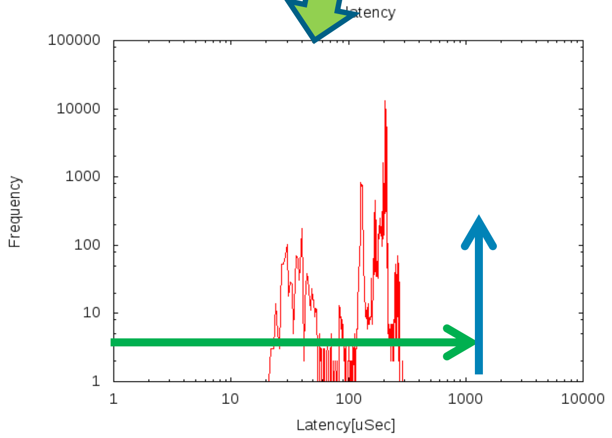


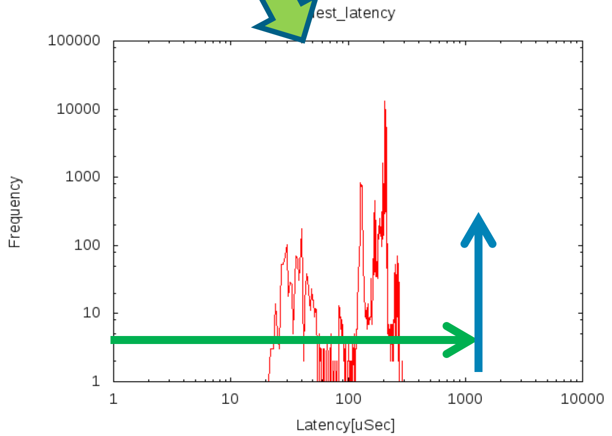


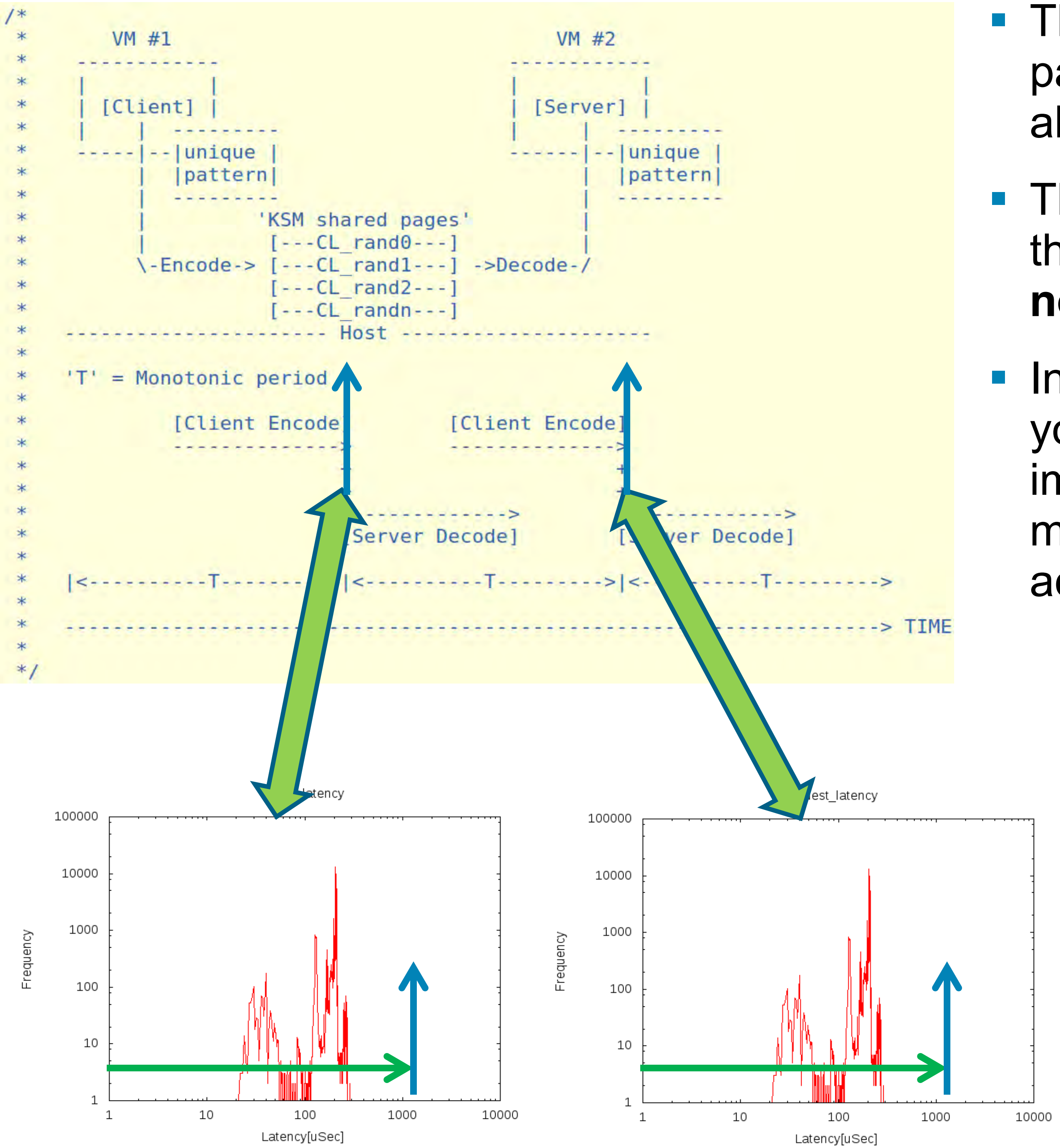


```
/*
 *        VM #1                              VM #2
 *     ------------                       ------------
 *     |          |                       |          |
 *     | [Client] |                       | [Server] |
 *     |    |   ---------                 |    |   ---------
 *     -----|--|unique  |                 -----|--|unique  |
 *          |  |pattern|                       |  |pattern|
 *          |   ---------                      |   ---------
 *          |         'KSM shared pages'       |
 *          |          [---CL_rand0---]        |
 *          \-Encode-> [---CL_rand1---] ->Decode-/
 *                     [---CL_rand2---]
 *                     [---CL_randn---]
 *    --------------------- Host --------------------
 *
 *    'T' = Monotonic period
 *
 *             [Client Encode]         [Client Encode]
 *             --------------->        --------------->
 *                            +                       +
 *                            +                       +
 *                            --------------->        --------------->
 *                            [Server Decode]         [Server Decode]
 *
 *    |<----------T--------->|<----------T--------->|<----------T--------->
 *
 *    -----------------------------------------------------------------> TIME
 *
 */
```

- The idea here is to do padding up to some value above the maximum jitter

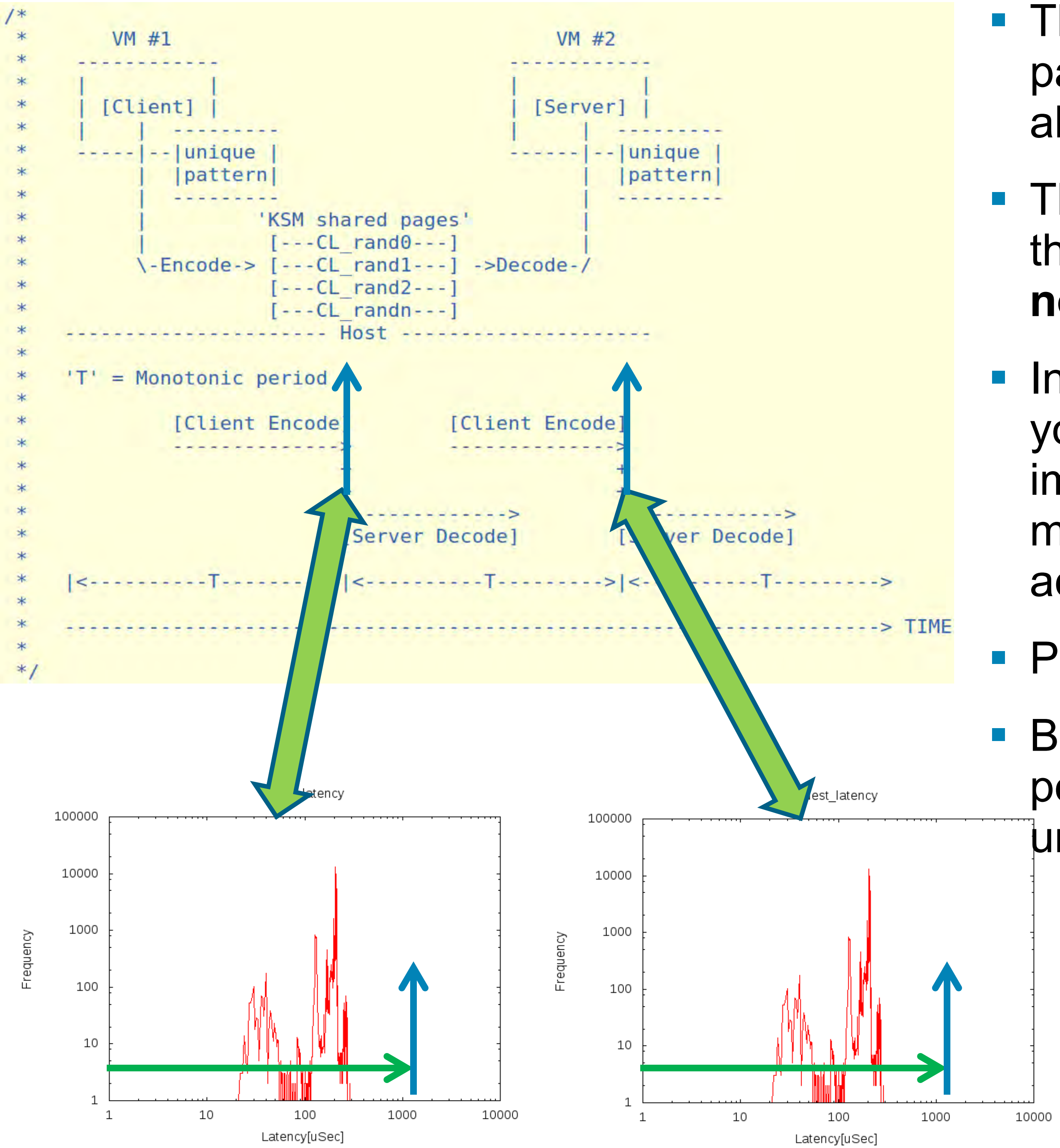






```
/*
 *      VM #1                               VM #2
 *   ------------                        ------------
 *   |          |                        |          |
 *   | [Client] |                        | [Server] |
 *   |    |  ---------                   |      |  ---------
 *   -----|--|unique |                   ------|--|unique |
 *        |  |pattern|                         |  |pattern|
 *        |  ---------                         |  ---------
 *        |          'KSM shared pages'        |
 *        |            [---CL_rand0---]        |
 *        \-Encode-> [---CL_rand1---] ->Decode-/
 *                   [---CL_rand2---]
 *                   [---CL_randn---]
 *  ---------------------- Host --------------------
 *
 *  'T' = Monotonic period
 *
 *         [Client Encode]        [Client Encode]
 *         ---------------->      ---------------->
 *                         +                       +
 *                         +                       +
 *                          ------------->          ------------->
 *                          [Server Decode]         [Server Decode]
 *
 *  |<----------T--------->|<----------T--------->|<----------T--------->
 *
 *  ----------------------------------------------------------------------> TIME
 *
 */
```

- The idea here is to do padding up to some value above the maximum jitter
- The problem here is that the **padding is subject to noise**
- In other word more time you spend trying to immunize yourself to noise more noise you end up accumulating

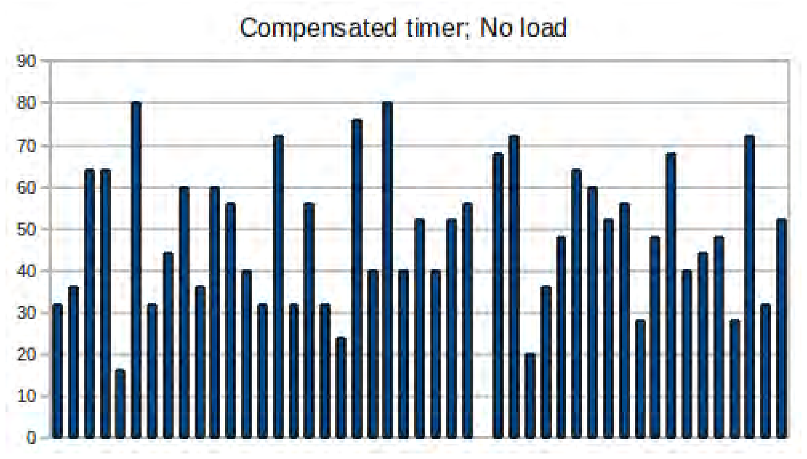


```
/*
 *       VM #1                                  VM #2
 *    ------------                          ------------
 *    |          |                          |          |
 *    | [Client] |                          | [Server] |
 *    |    |  ---------                     |      |  ---------
 *    -----|--|unique |                     -------|--|unique |
 *         |  |pattern|                            |  |pattern|
 *         |  ---------                            |  ---------
 *         |          'KSM shared pages'           |
 *         |           [---CL_rand0---]            |
 *         \-Encode-> [---CL_rand1---] ->Decode-/
 *                     [---CL_rand2---]
 *                     [---CL_randn---]
 *    ---------------------- Host --------------------
 *
 *    'T' = Monotonic period
 *
 *             [Client Encode]        [Client Encode]
 *             --------------->       --------------->
 *                            +                      +
 *                            +                      +
 *                             ------------->         ------------->
 *                             [Server Decode]        [Server Decode]
 *
 *    |<----------T---------->|<----------T--------->|<----------T--------->
 *
 *    ------------------------------------------------------------------> TIME
 *
 */
```

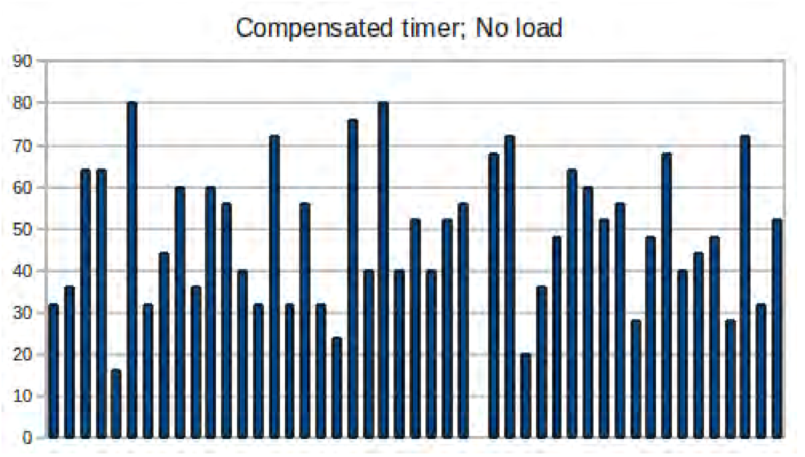
- The idea here is to do padding up to some value above the maximum jitter
- The problem here is that the **padding is subject to noise**
- In other word more time you spend trying to immunize yourself to noise more noise you end up accumulating
- Padding consume CPU
- By stretching the timer period it's easy to stay under 1% of CPU usage

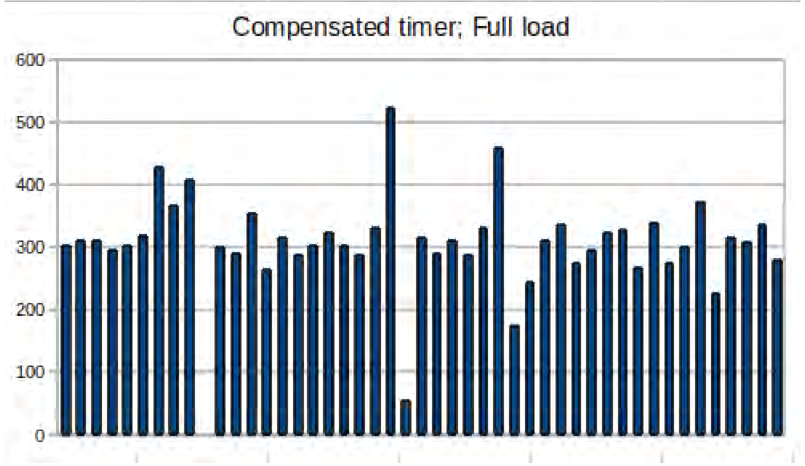
- It’s a tricky problem but at the end I got it right!

- It’s a tricky problem but at the end I got it right!
- In short the padding is using a calibrated software loop that is kept in check with the TSC

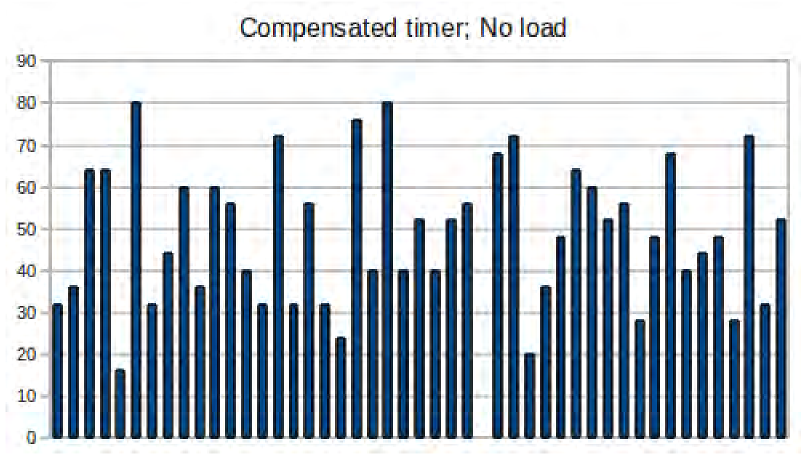


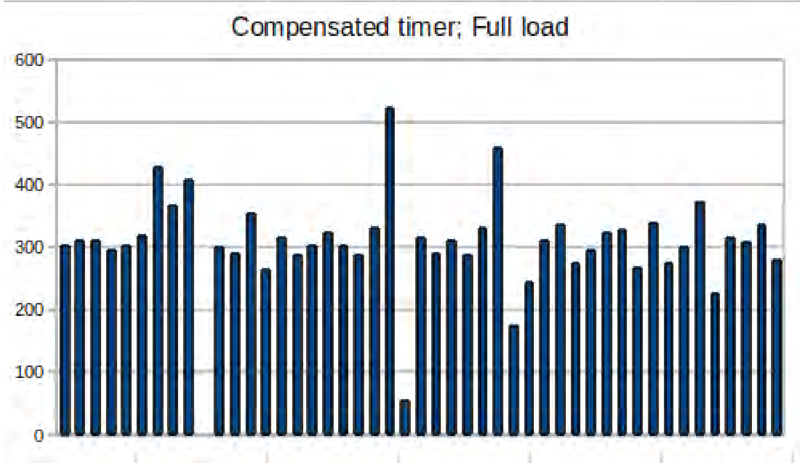
- It’s a tricky problem but at the end I got it right!
- In short the padding is using a calibrated software loop that is kept in check with the TSC
- Assume 2.4Ghz machine;
- On a idle system:
  ~50 cycle ➔ 20 nSec





- It’s a tricky problem but at the end I got it right!
- In short the padding is using a calibrated software loop that is kept in check with the TSC
- Assume 2.4Ghz machine;
- On a idle system:
  ~50 cycle ➔ 20 nSec
- On a loaded system
  ~300 cycle ➔120 nSec





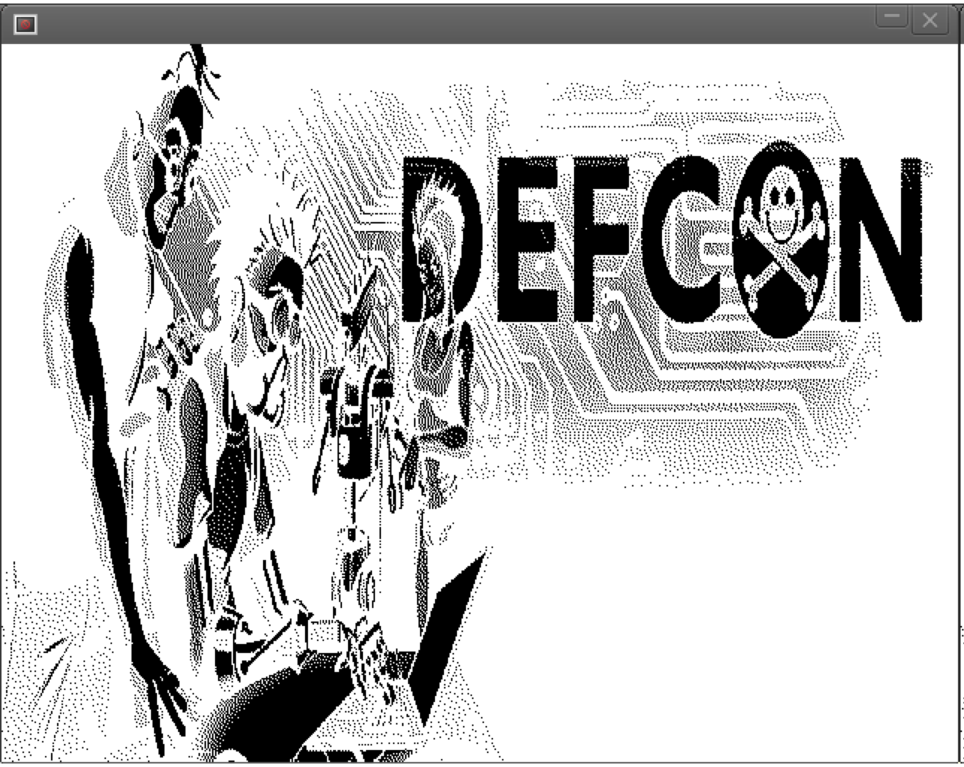
- It’s a tricky problem but at the end I got it right!
- In short the padding is using a calibrated software loop that is kept in check with the TSC
- Assume 2.4Ghz machine;
- On a idle system:
  ~50 cycle ➔ 20 nSec
- On a loaded system
  ~300 cycle ➔120 nSec

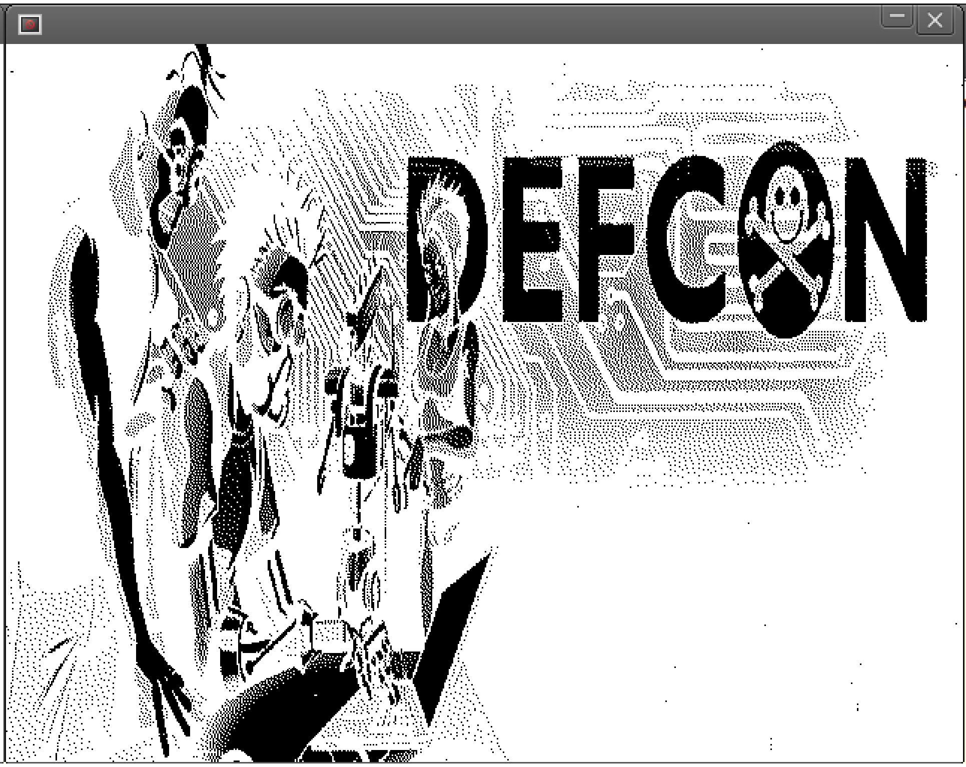
Timers:

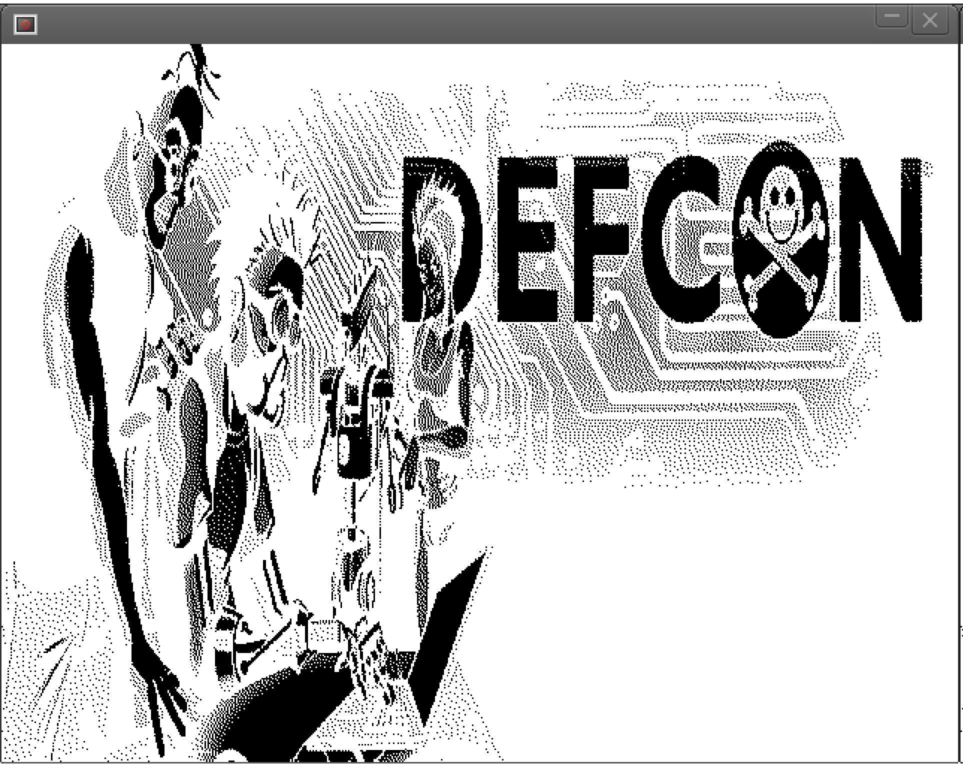
- 100uSec = 240 000 cycle
- 10uSec = 24 000 cycle (best case)

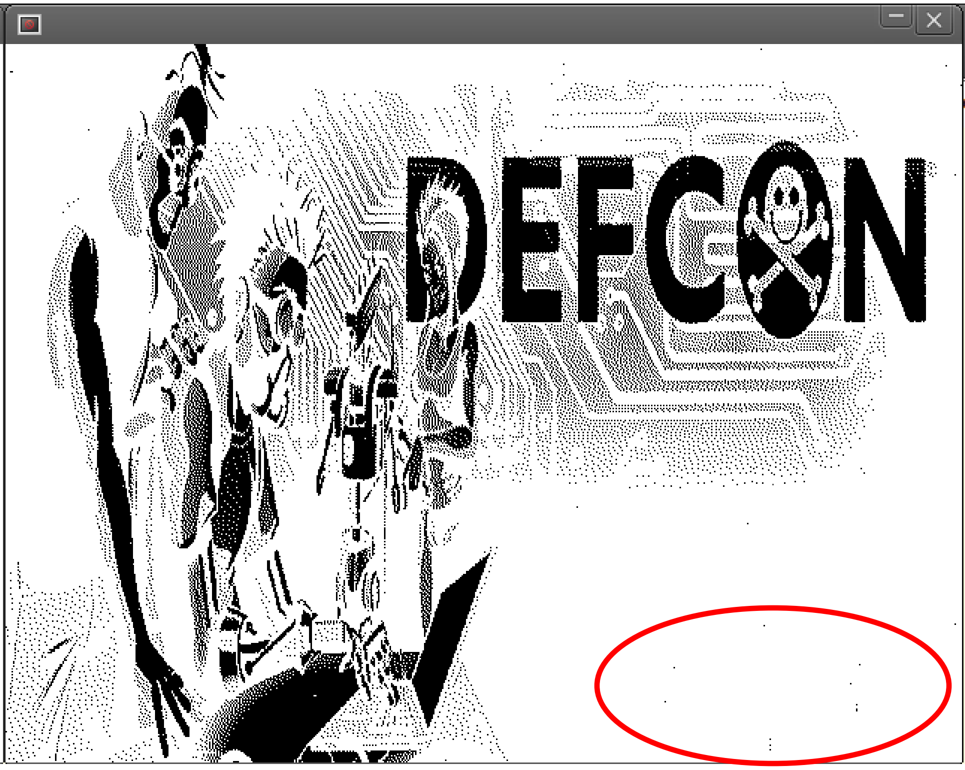
Recap

- Encoding / decoding based on memory access time
  - (1 = slow, 0 = fast)
- Got rid of the HW prefetching (without disabling it from BIOS!)
  - (randomized the access to cache lines / pages)
- Physical memory pages that are shared across VM
  - Thanks to KSM ☺
- PLL and high precision inter-VM synchronization
  - (Compensated timer <120 nSec jitter)
- Time for a demo!


DEFCON


DEFCON


DEFCON


DEFCON

Video #2

Video #3

Mitigation

- Disable page-deduplication (KSM) / Per-VM policy
  - No inter-VM shared read-only pages
  - Flush ‘clflush’ and reload won’t work
  - No OS / Application fingerprinting (de-duplication page-fault)
  - Higher memory cost
- X86 ‘clflush’ instruction: Privilege?
  - Microcode?
- Co-location policy (per-core / per-socket / per-box)

Detection

- Hardware counter
- Inter-VM scheduling “abnormality”
- TSC related “abnormality”

Thank you!