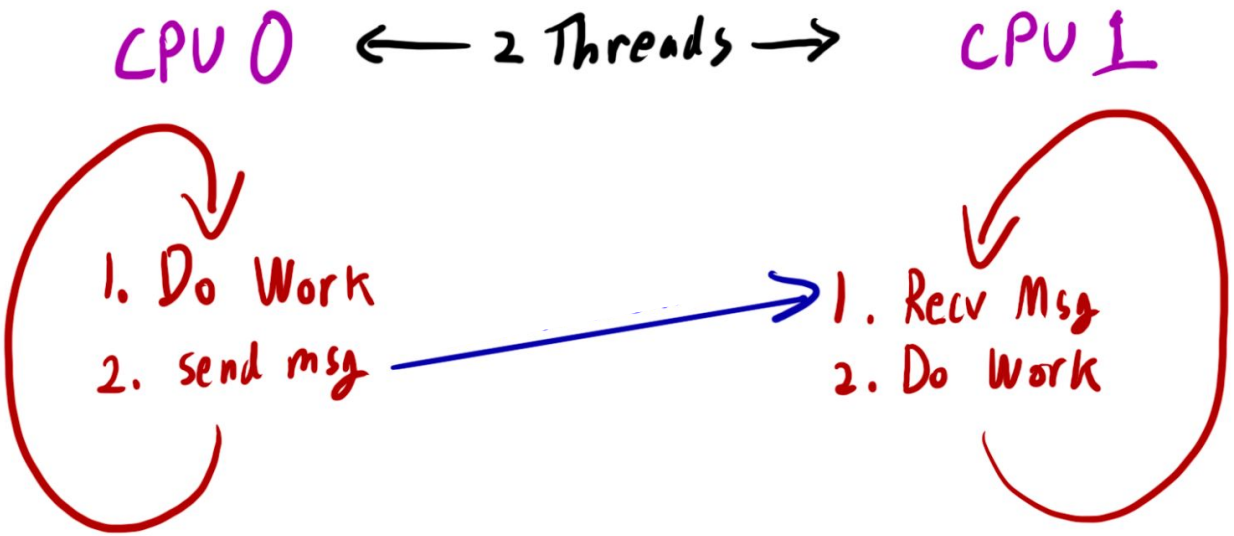


Do Work: Allocates + frees memory on per-thread heaps

Is This Better as one process or two?

Virtual Memory

A Huge Step Backwards

Daniel Bittman (he/him)

HPTS 2024

Virtual Memory is Evil

Semantics

Performance

Security

Flat, Uniform.

Contiguous, infinite,

Isolated, Fast

Flat, Uniform.

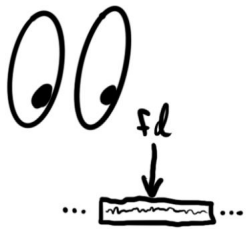
non-uniform Access latency
Page faults, discontinuous allocation,
holes, Varying memory coherence,

Contiguous, infinite,

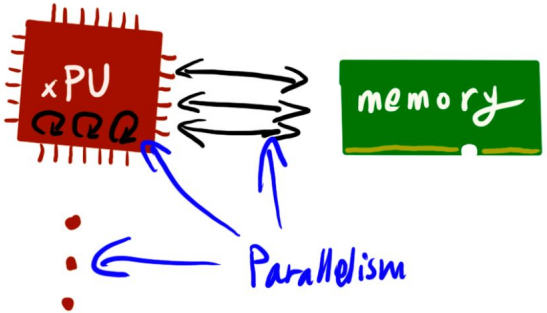
infallible malloc, Spatial Safety, temporal
Safety, buffer overflow, ROP,

Isolated, Fast

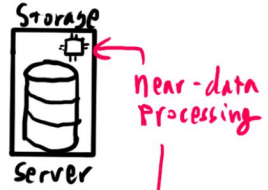
TLB miss, Shootdown, Meltdown,
Cross-machine coherence, local execution
context, Page table walking


fd

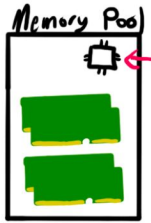

fd

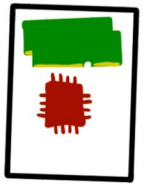

xPU
memory
Parallelism

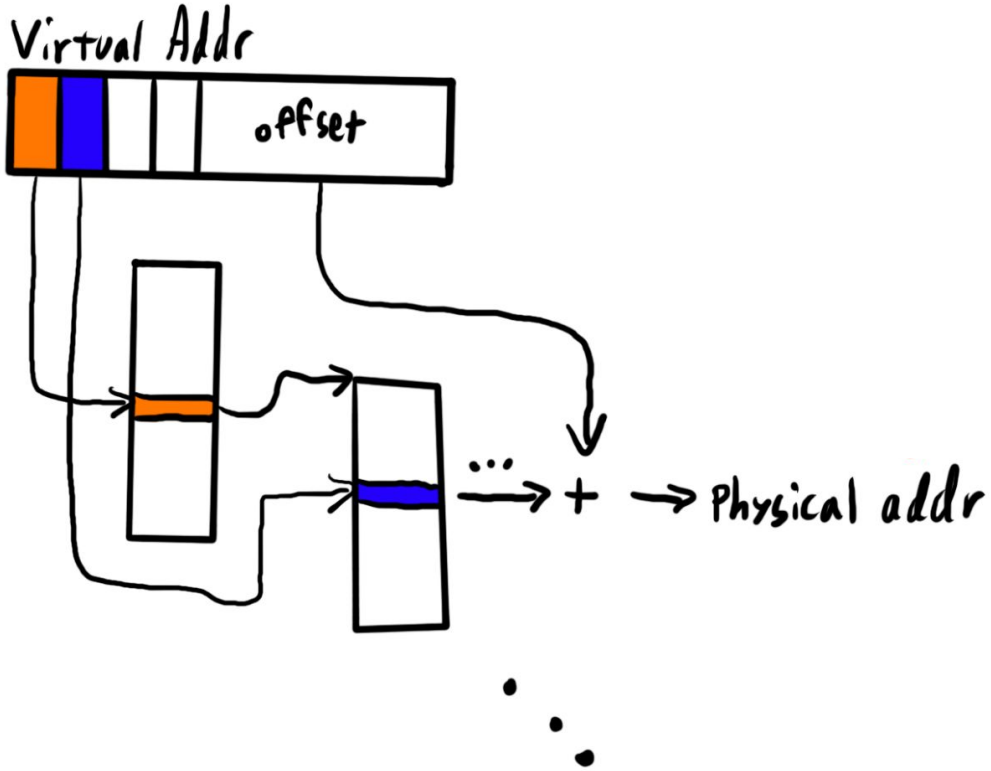

fd

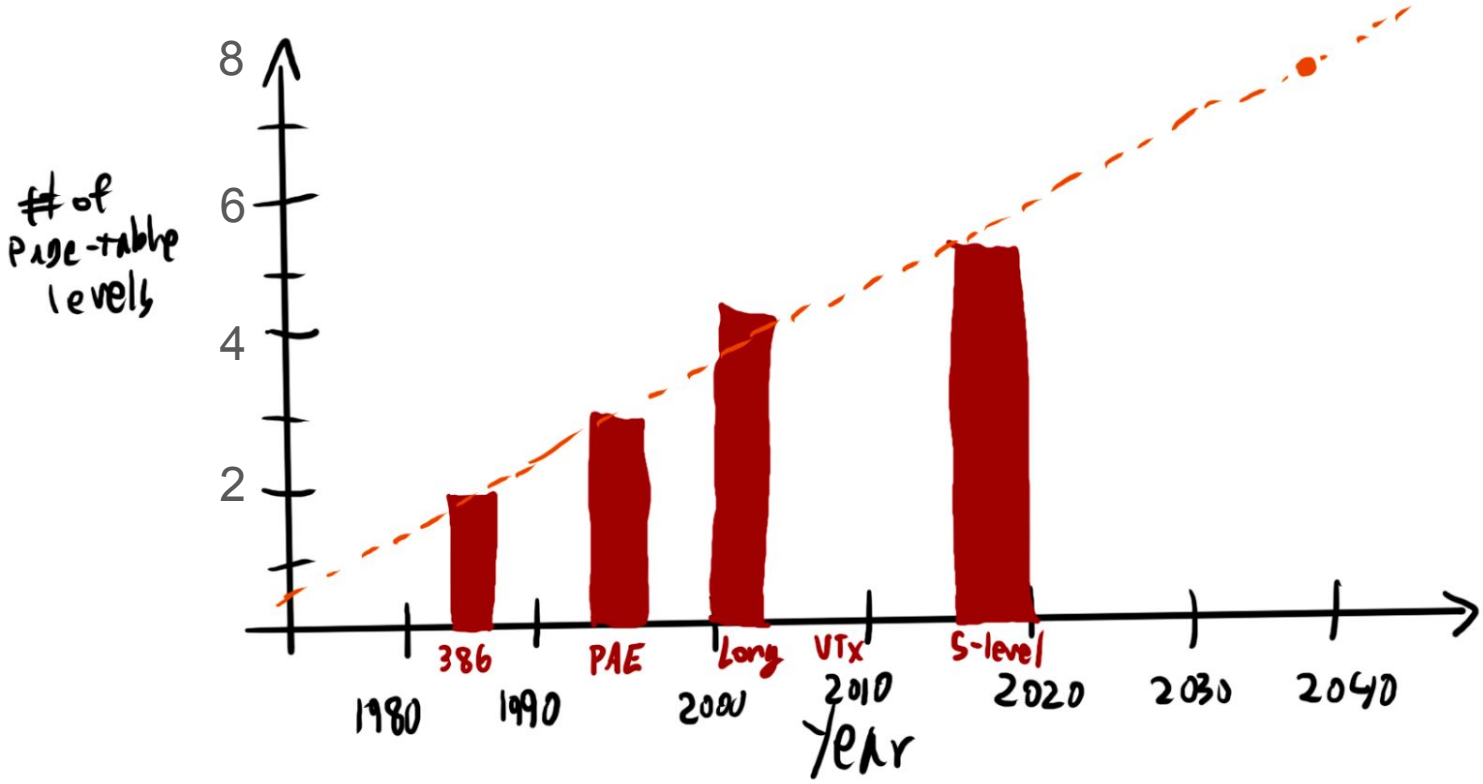

Storage
Server
near-data
Processing

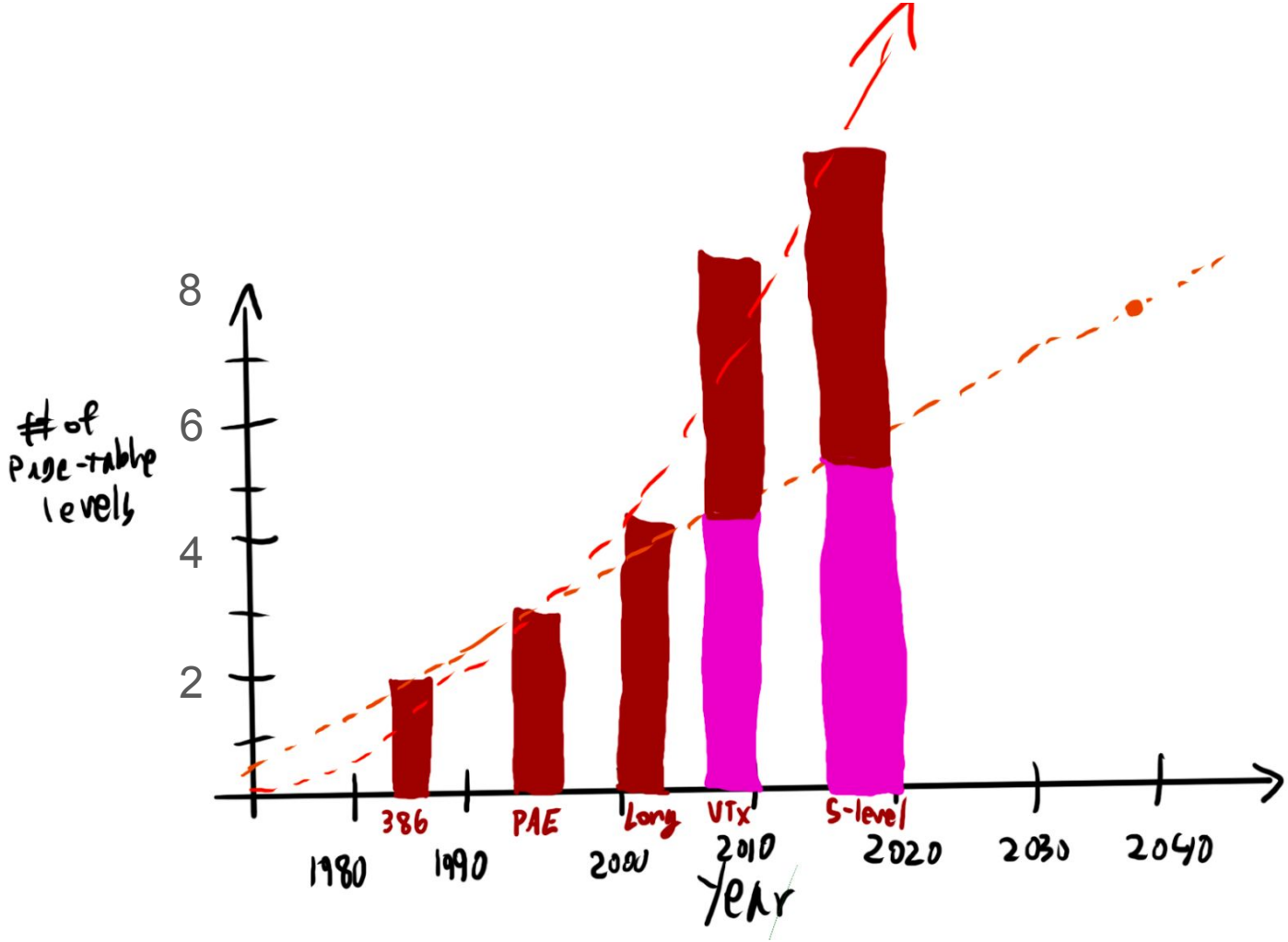

interconnect
In-net
Processing

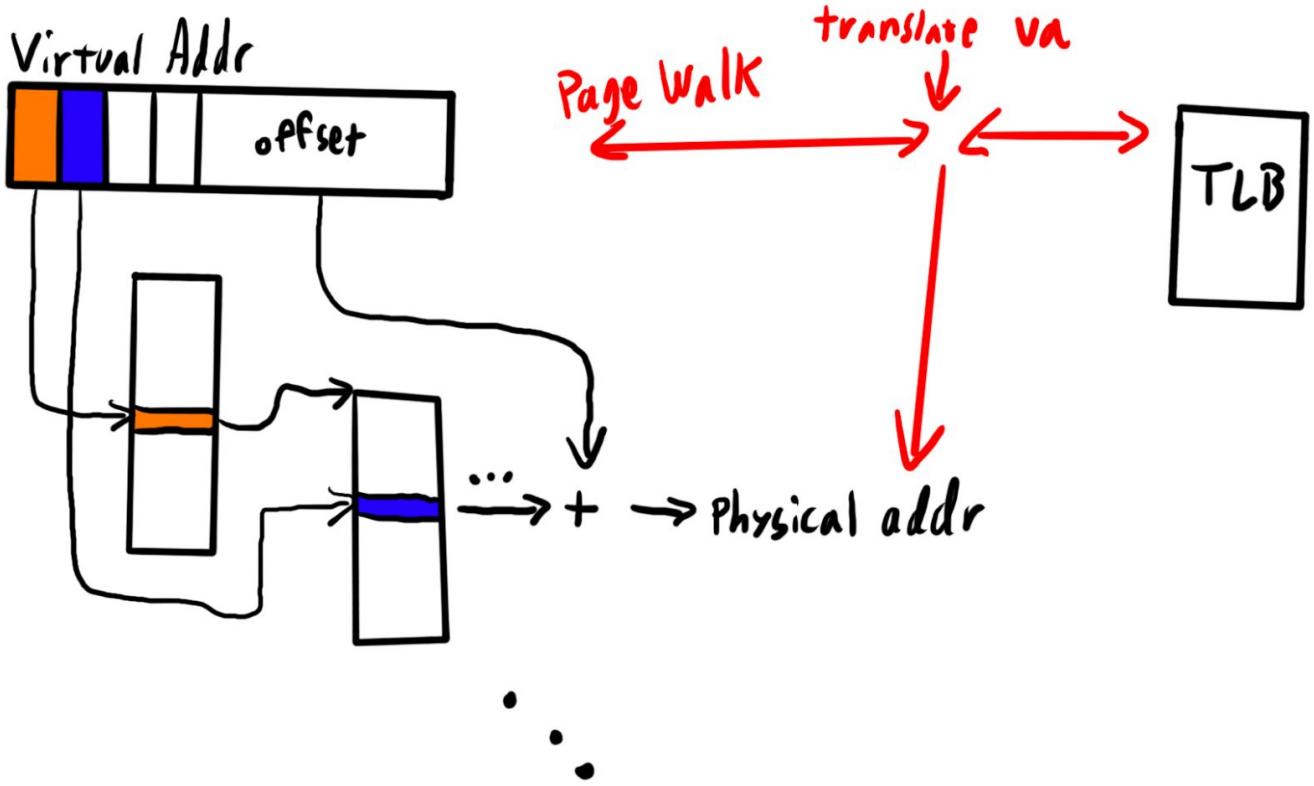

Memory Pool

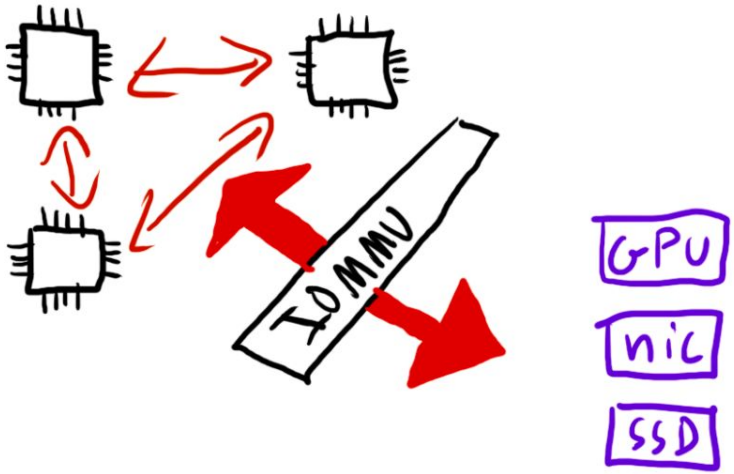


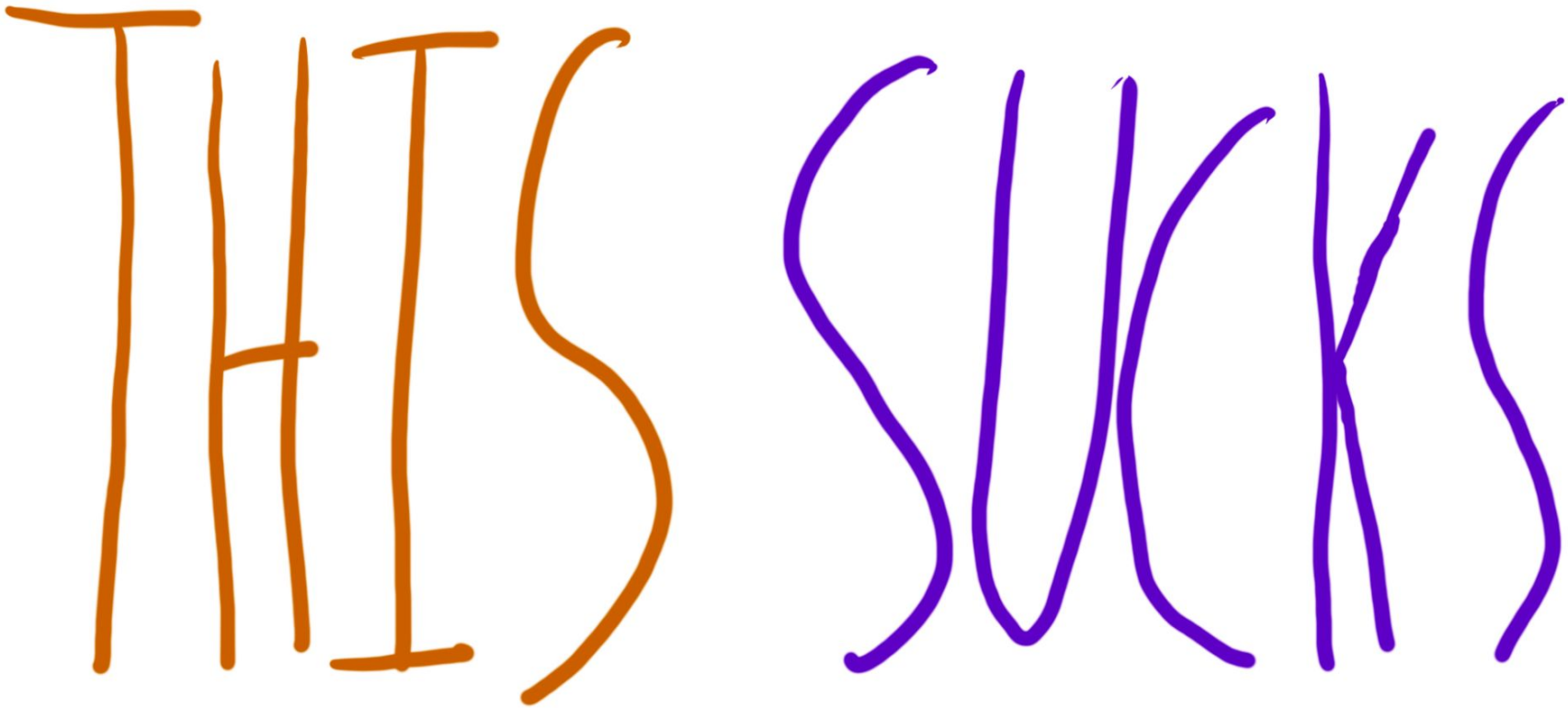

Virtual Addr
offset
+
Physical addr


# of Page-table levels
8
6
4
2
386
PAE
Long
VTx
5-level
1980
1990
2000
2010
2020
2030
2040
Year


# of Page-table levels
8
6
4
2
386
PAE
Long
VTx
5-level
1980
1990
2000
2010
2020
2030
2040
Year


Virtual Addr
offset
translate va
Page Walk
TLB
+
Physical addr

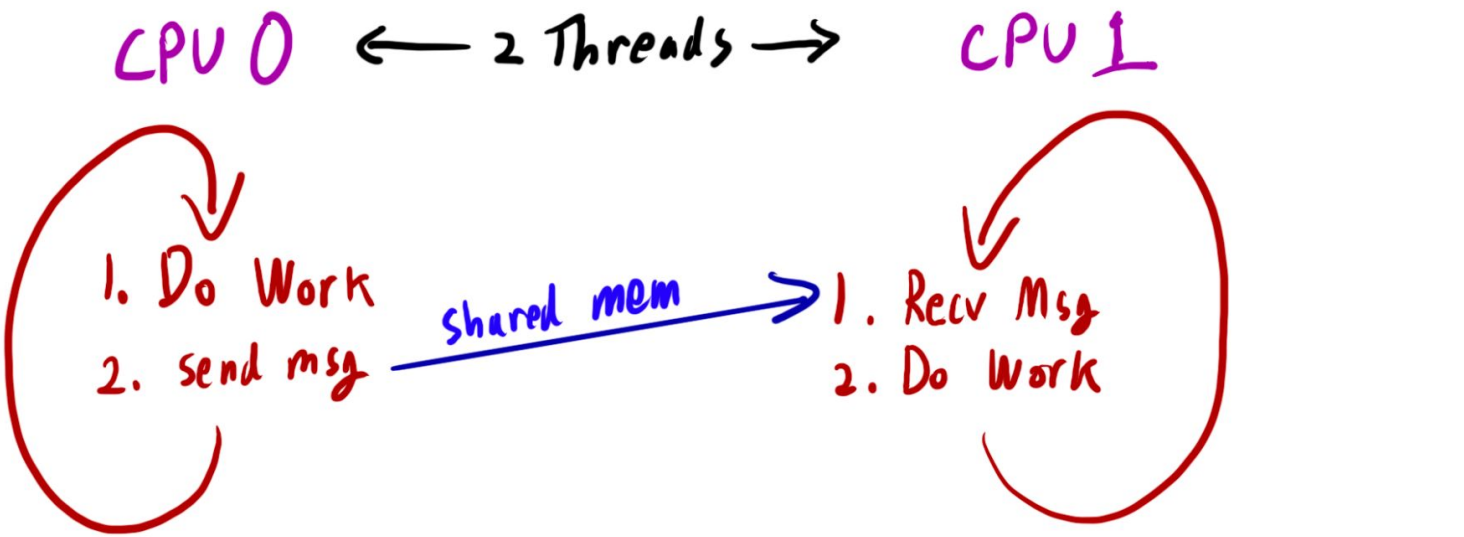

IOMMU
GPU
NIC
SSD


THIS SUCKS

TLB Coherence

Linux TLB ~~optimizations~~ complications

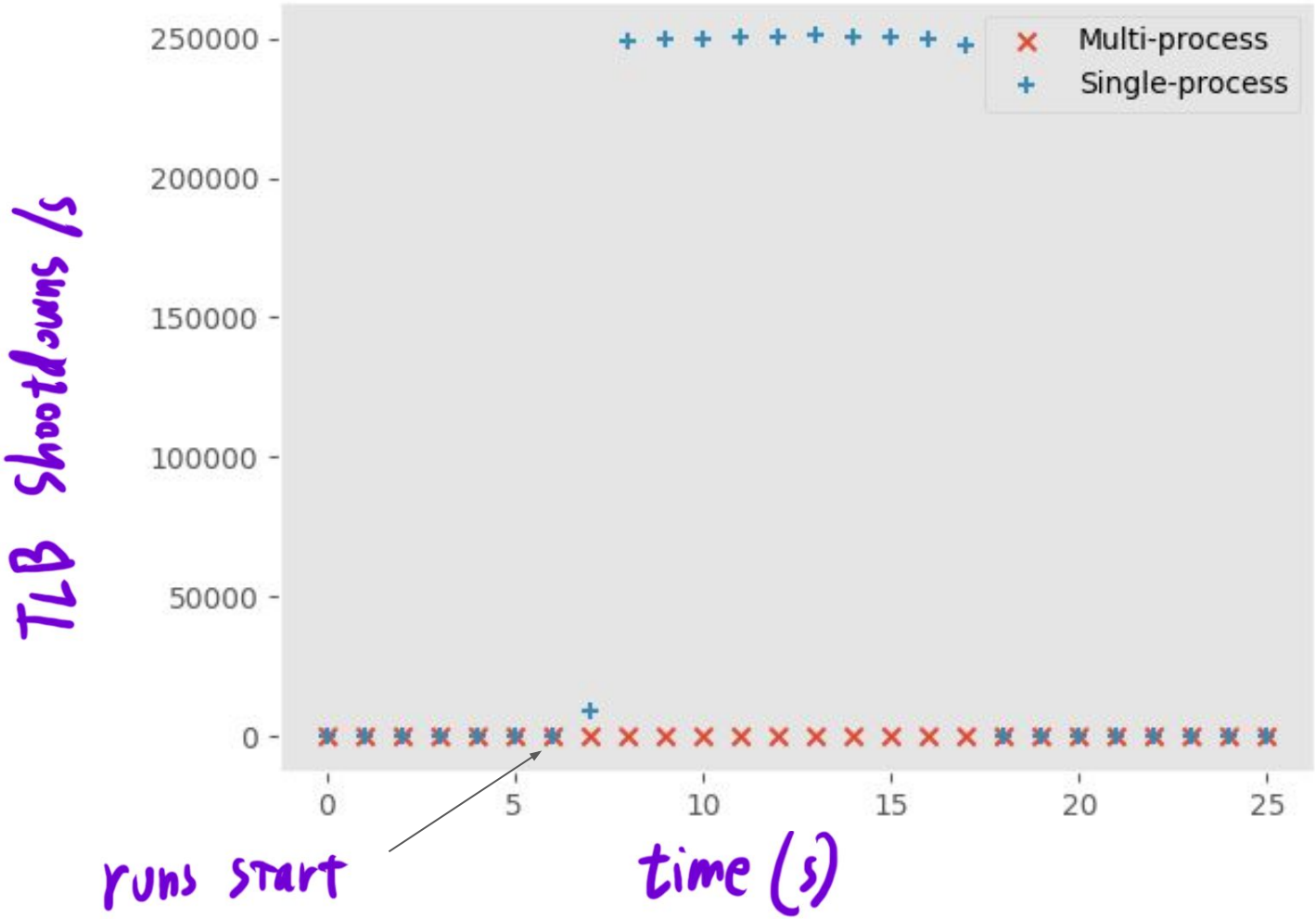
>10K LoC



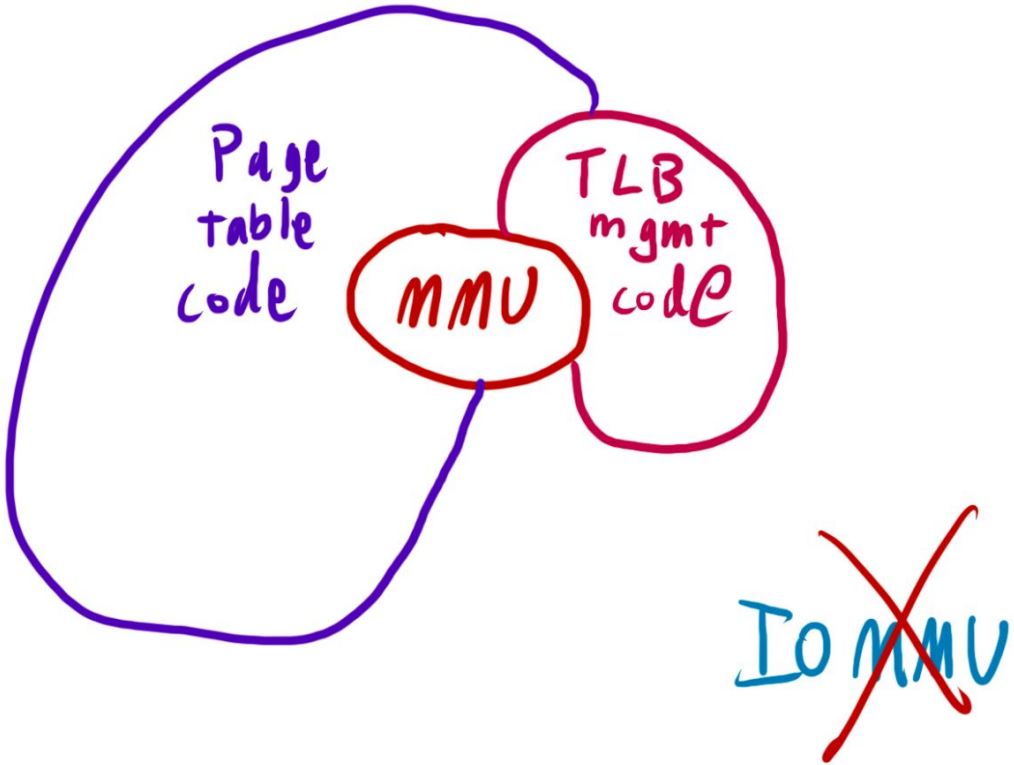
calls munmap(2)

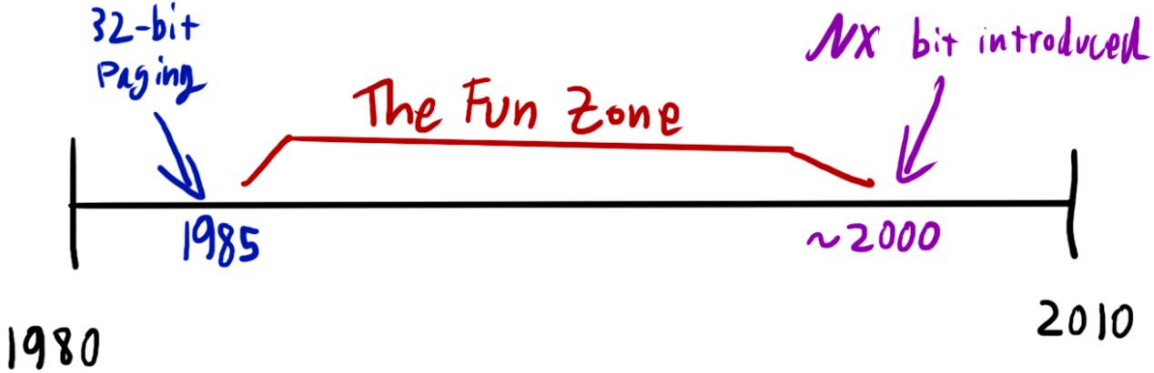
Do Work: Allocates + frees memory on per-thread heaps

Is This Better as one process or two?


TLB Shootdowns /s
250000
200000
150000
100000
50000
0
Multi-process
Single-process
0
5
10
15
20
25
runs start
time (s)

Attack Surface




32-bit
Paging
The Fun Zone
NX bit introduced
1985
~2000
1980
2010

Infinite Memory

and

Infallible Malloc

Infinite Memory and Infallible Malloc

vm.overcommit_memory = thoughts and prayers

Infinite Memory

and

Infallible Malloc

How many libc functions allocate memory?

```
char* buffer = NULL;
while(buffer == NULL){
    buffer = malloc(sizeof(char) * 100);
}
```

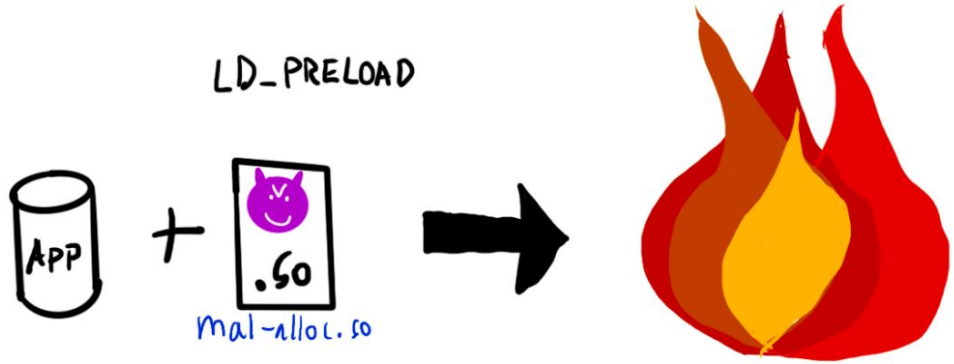
```
char* buffer = NULL;
while(buffer == NULL){
    buffer = malloc(sizeof(char) * 100);
}
```

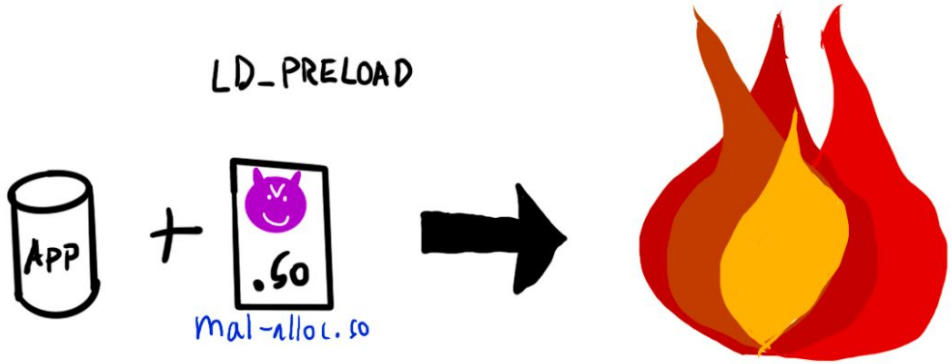
→ malloc never fails

→ oh, but it's UB not to check

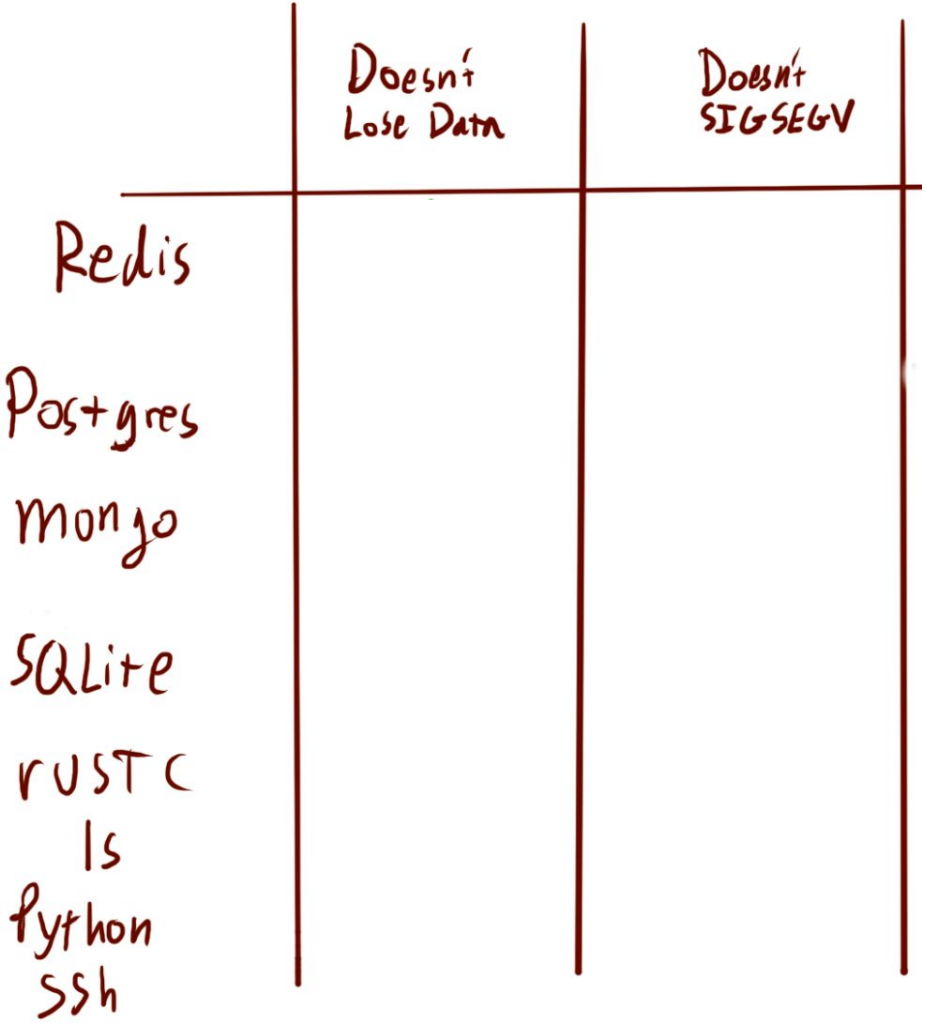
→ wait it totally does fail

∴ Don't worry about it, it's impossible to recover


LD_PRELOAD
APP
.so
mal-alloc.so




```
void *malloc(size_t size){
    if (should_i_fail_lol(size)) return NULL;
    //call real malloc
}
```


Doesn't Lose Data
Doesn't SIGSEGV
Redis
Postgres
mongo
SQLite
rustc
ls
Python
ssh

	Doesn't Lose Data	Doesn't SIGSEGV	Doesn't Exit	Stays Available
Redis				
Postgres				
mongo				
SQLite				
rustc				
ls				
python				
ssh				

	Doesn't Lose Data	Doesn't SIGSEGV	Doesn't Exit	Stays Available
Redis	✓			
Postgres	✓			
mongo	✓			
SQLite	✓			
rustc	?			
ls	N/A			
python	?			
ssh	✓			

	Doesn't Lose Data	Doesn't SIGSEGV	Doesn't Exit	Stays Available
Redis	✓	X	X	X
Postgres	✓	X	X*	X
mongo	✓	✓	X	X
SQLite	✓	✓	✓	✓
rustc	?	✓	X	X
ls	N/A	✓	X	X
python	?	X	X	X
ssh	✓	✓	X	X

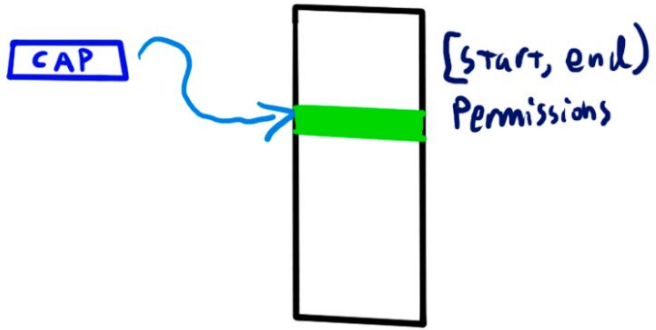
LD_PRELOAD = libmal-alloc.so sh

(✓ ✓ ✗ ✗)


Segments!
Logical Space
← Abstract Physical Memory


Segments!
Logical Space
← Abstract Physical Memory

CHERI


CAP
[start, end)
Permissions

Allocating Memory

sucks

```
my_poor_api() -> Result<..., AllocError>
```

```
a_more_correct_expression_of_the_api
```

So,

Dont.

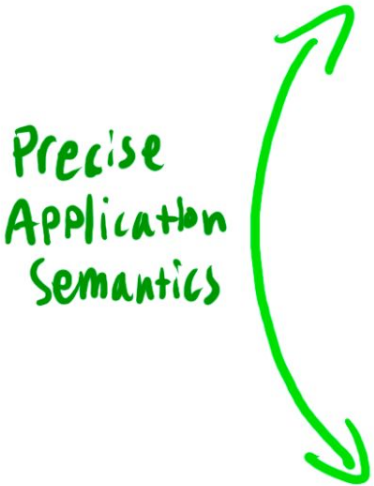

Twizzler

`&mut T`

`&T`

This reasoning is needed for C, C++ too

Depending on appending



Immutability changes everything

Safe, fast APIs for shared memory

→ Virtual Memory has bad Security + Semantics

Hidden Complexity is still complexity!

→ Virtual Memory has huge Performance overhead

And sometimes surprising overhead!

→ Allocating memory can fail

And Freeing memory can send messages!

→ Intel Page table depth followed Moore's Law ~2008

→ The OS + DB communities should go after the CPU designers

→ Just run SPDK on baremetal x86 segment mode.

Thank you,

Any Questions?

Twizzler.io
dbittman.com

Daniel Bittman (he/him)