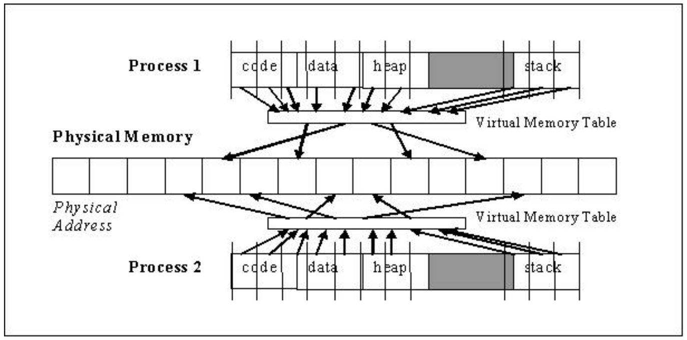
More Memory, More Problems:

Towards De-virtualizing the Virtual Memory…and More!

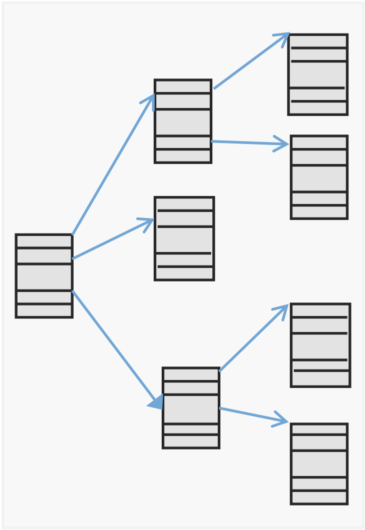
Arman Shanjani, En-Ui (Annie) Lin, Michael Vaughn,
Xiangjin Wu

"Virtual memory was invented in a time of scarcity. Is it still a good idea?"
– Charles Thacker, 2010 ACM Turing Award Lecture.

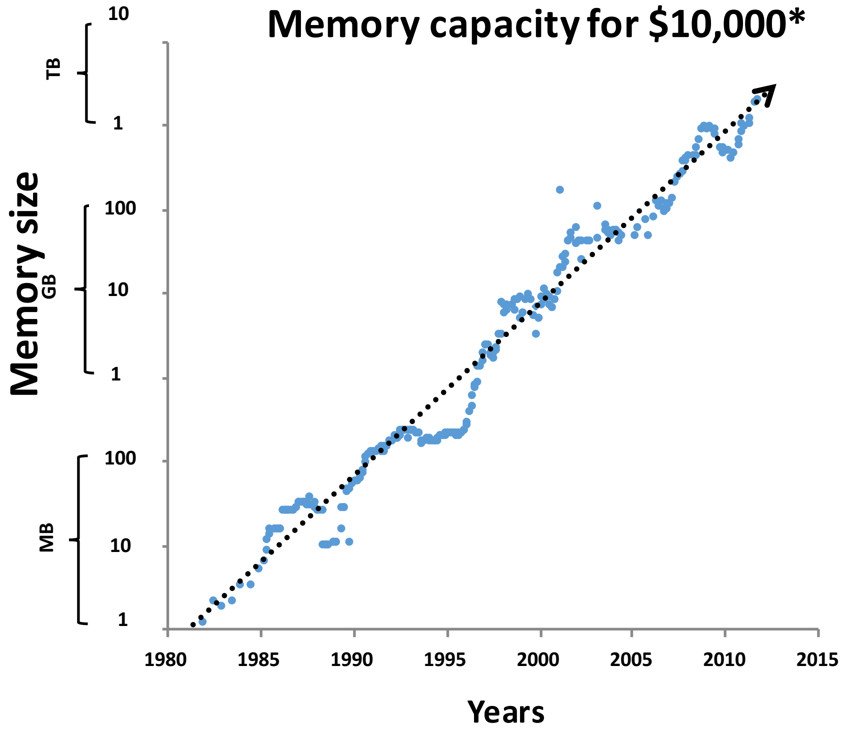


Why Virtual Memory?

- Demand Paging
- Program size not limited by main memory
- Protection
- Shared Memory
- Etc.



What’s Wrong, Then?



Memory capacity for $10,000*

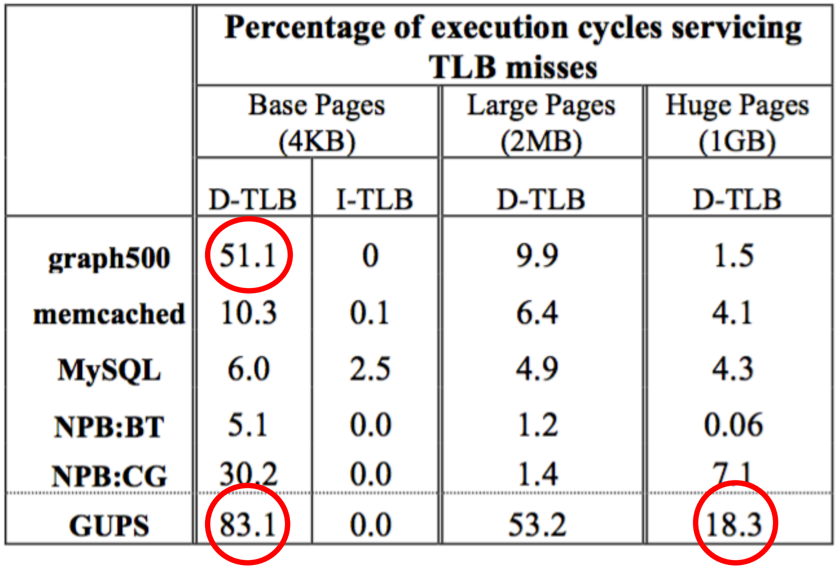
*Inflation-adjusted 2011 USD, from: jcmit.com

Year	Processor	L1 DTLB entries
1999	Pent. III	72
2001	Pent. 4	64
2008	Nehalem	96
2012	IvyBridge	100
2015	Broadwell	100

1.75 entries / year !

[ISCA ’15]

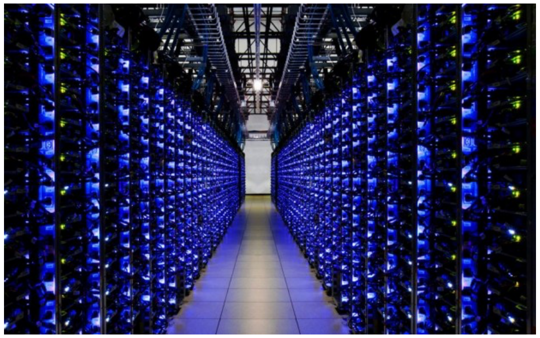
What’s Wrong, Then?



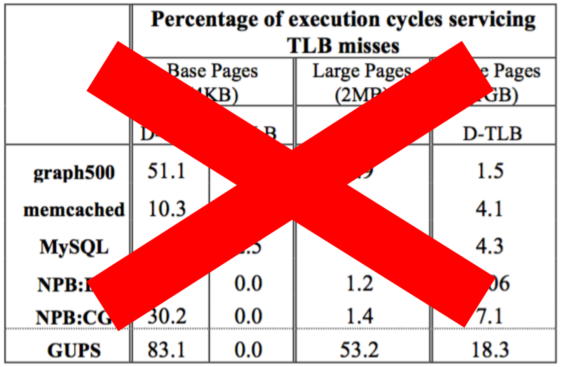
	Percentage of execution cycles servicing TLB misses			
	Base Pages (4KB)		Large Pages (2MB)	Huge Pages (1GB)
	D-TLB	I-TLB	D-TLB	D-TLB
graph500	51.1	0	9.9	1.5
memcached	10.3	0.1	6.4	4.1
MySQL	6.0	2.5	4.9	4.3
NPB:BT	5.1	0.0	1.2	0.06
NPB:CG	30.2	0.0	1.4	7.1
GUPS	83.1	0.0	53.2	18.3

[ISCA ’13]

Who Doesn’t Need Virtual Memory?

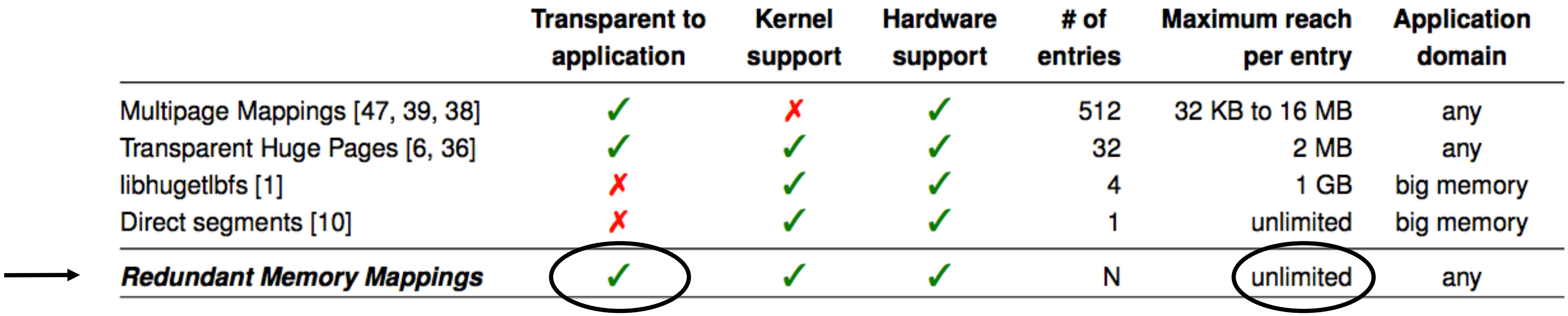


- No virtual machines
- Huge amounts of memory
- Same protection permissions across (almost) all virtual address space

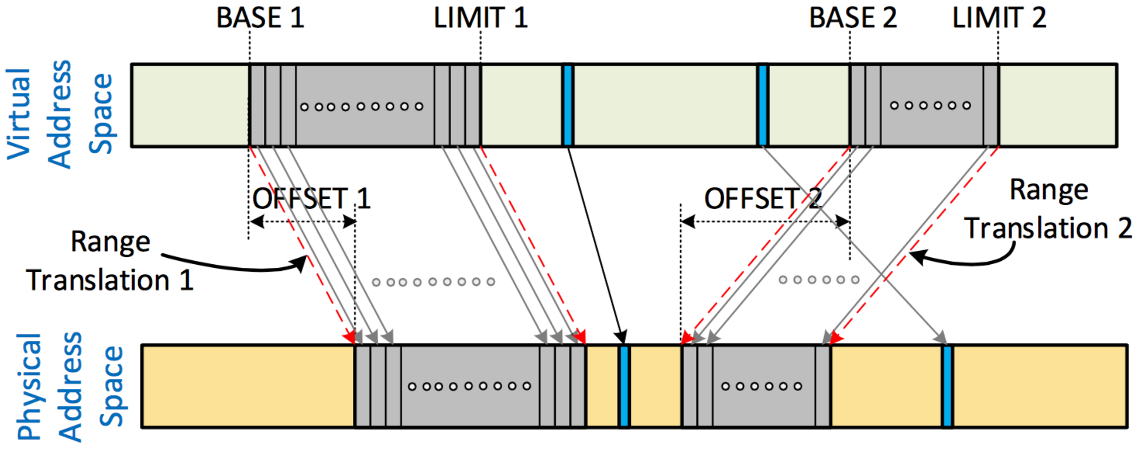


	Percentage of execution cycles servicing TLB misses			
	Base Pages ([illegible]KB)		Large Pages (2M[illegible]	[illegible]e Pages [illegible]GB)
	D[illegible]	[illegible]B	[illegible]	D-TLB
graph500	51.1	[illegible]	[illegible]	1.5
memcached	10.3	[illegible]	[illegible]	4.1
MySQL	[illegible]	[illegible]	[illegible]	4.3
NPB:[illegible]	[illegible]	0.0	1.2	[illegible]06
NPB:CG	30.2	0.0	1.4	7.1
GUPS	83.1	0.0	53.2	18.3

Previous Work

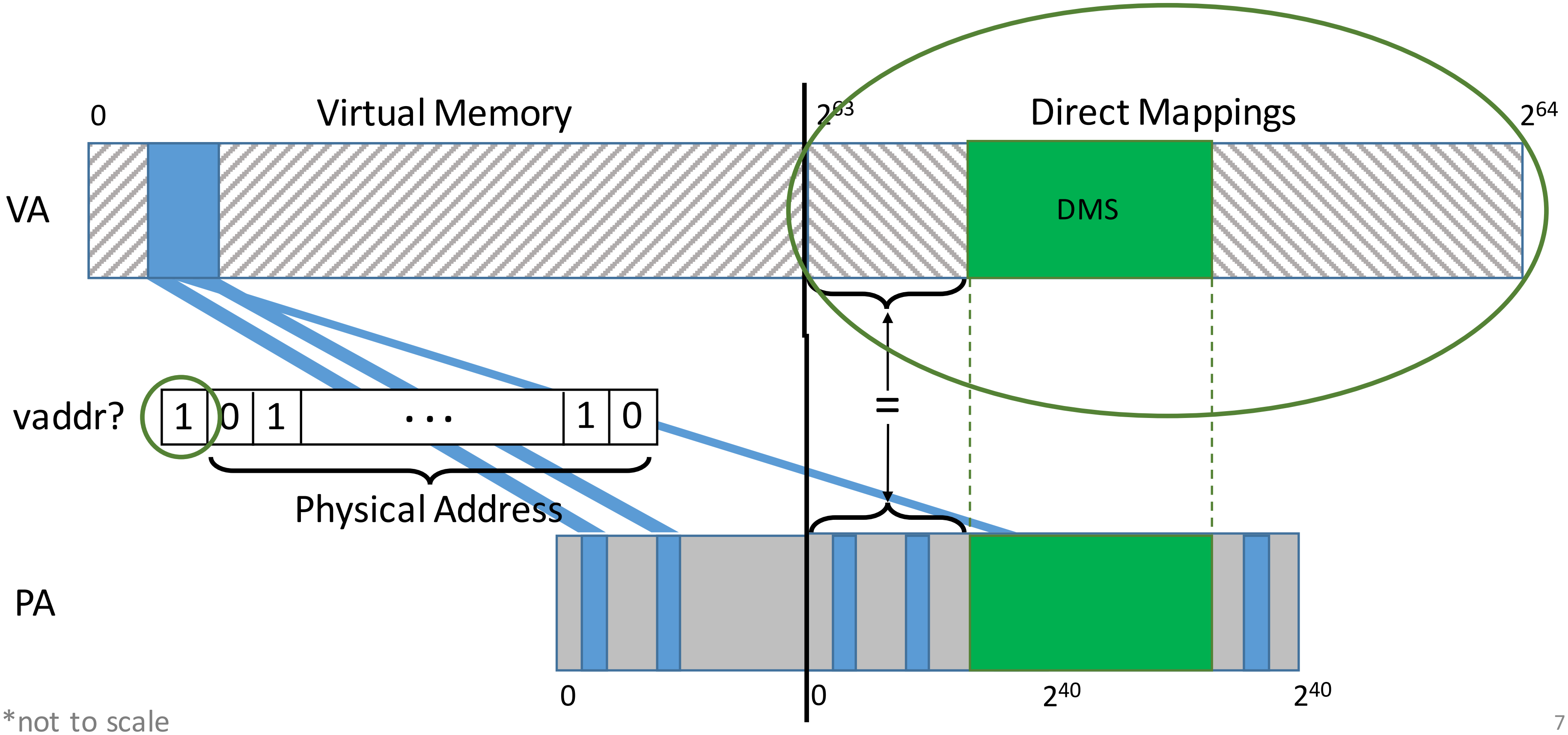


	Transparent to application	Kernel support	Hardware support	# of entries	Maximum reach per entry	Application domain
Multipage Mappings [47, 39, 38]	✓	✗	✓	512	32 KB to 16 MB	any
Transparent Huge Pages [6, 36]	✓	✓	✓	32	2 MB	any
libhugetlbfs [1]	✗	✓	✓	4	1 GB	big memory
Direct segments [10]	✗	✓	✓	1	unlimited	big memory
Redundant Memory Mappings	✓	✓	✓	N	unlimited	any



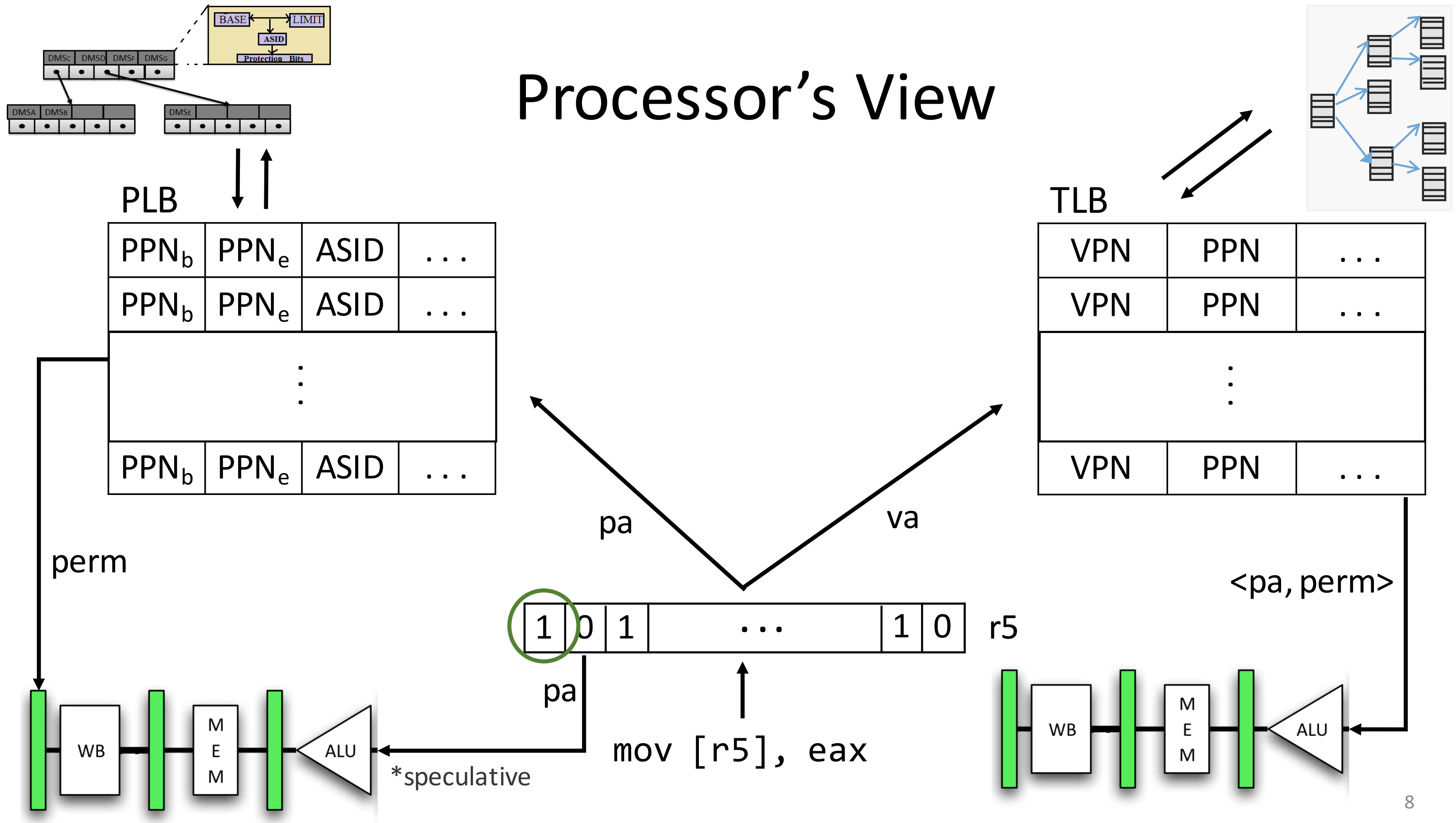
[ISCA '15]

Design



*not to scale

Processor’s View



PLB

PPN$_b$	PPN$_e$	ASID	. . .
PPN$_b$	PPN$_e$	ASID	. . .
⋮			
PPN$_b$	PPN$_e$	ASID	. . .

TLB

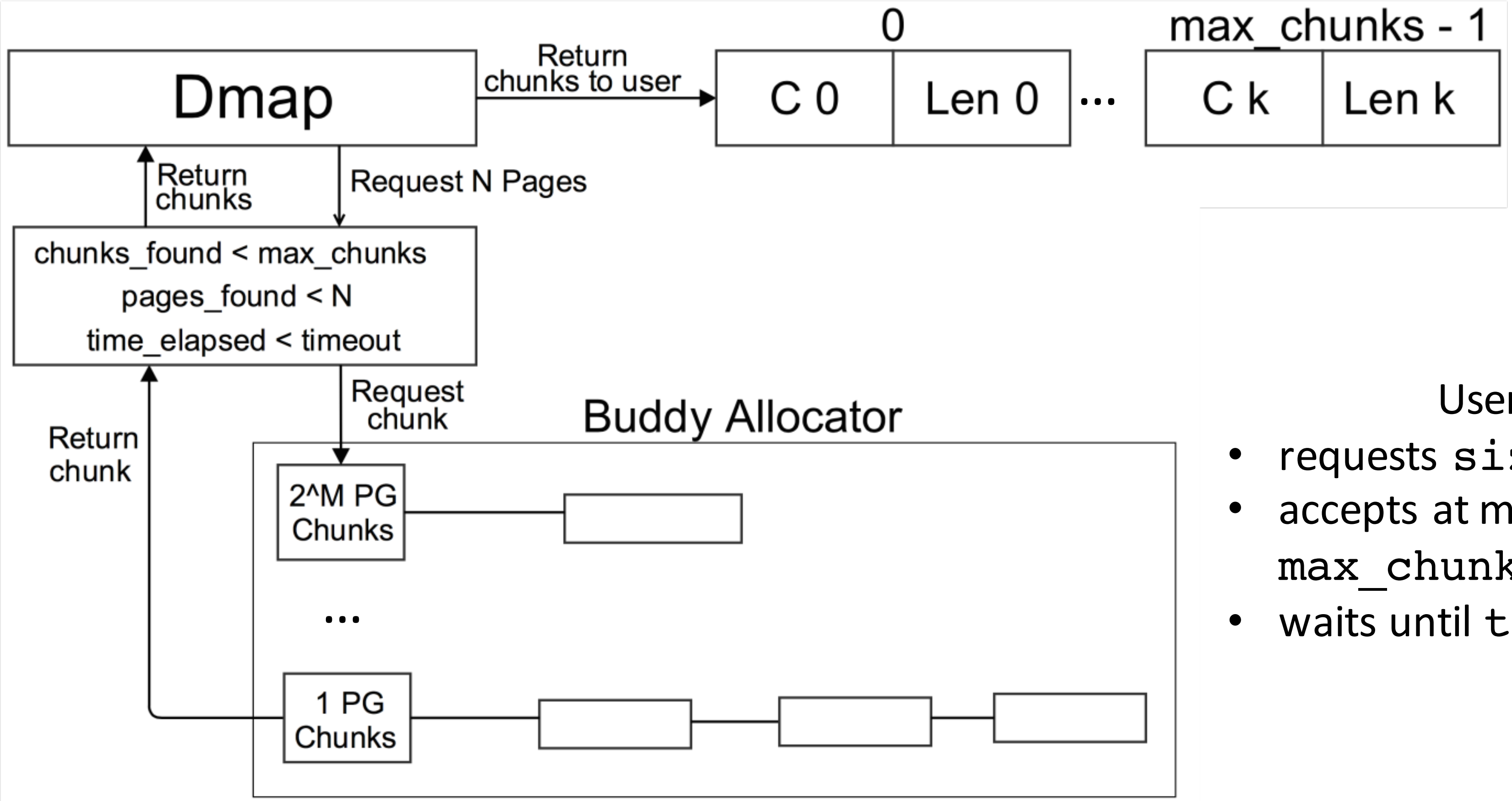
VPN	PPN	. . .
VPN	PPN	. . .
⋮		
VPN	PPN	. . .

```
mov [r5], eax
```

*speculative

System Call Interface

`dmap(size, max_chunks, timeout)`



Userspace:

- requests `size` bytes
- accepts at most `max_chunks` segments
- waits until `timeout`

Possible User-Level Policies

- Naive policy – simple replacement for mmap
- Initial policy – ask for all at the beginning
- Prediction policy – use a prediction algorithm based on prior use
- Reservation policy – request larger sizes than needed, reduce if necessary

Conclusion

- Virtual memory is abstraction for an older age
- TLB Reach is a big problem today
- Big companies cannot afford the performance penalty
- Directly-mapped segments provide performance benefits
- Export interface to user level
  - Apps customize policy