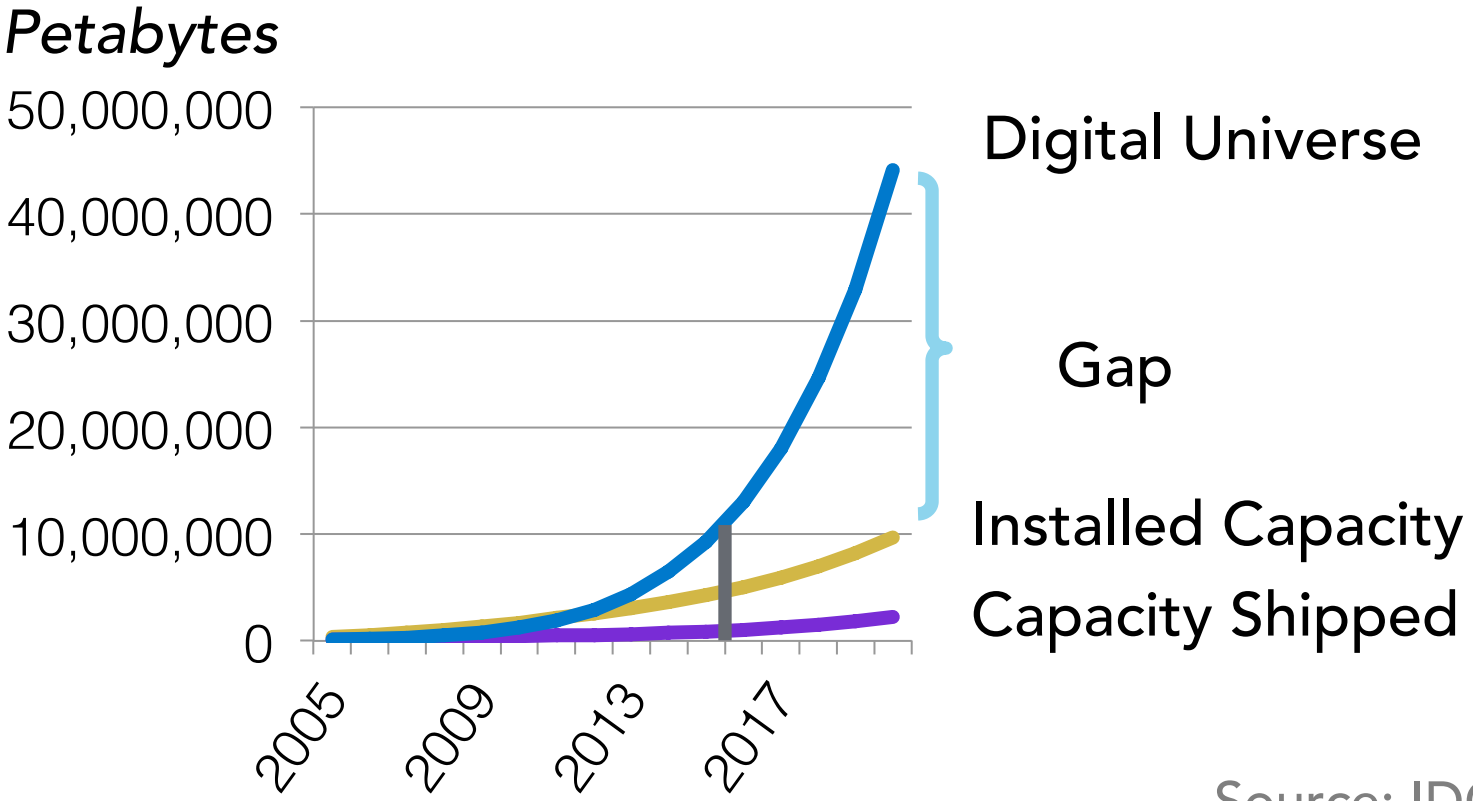
Progress & Challenges for Virtual Memory Management

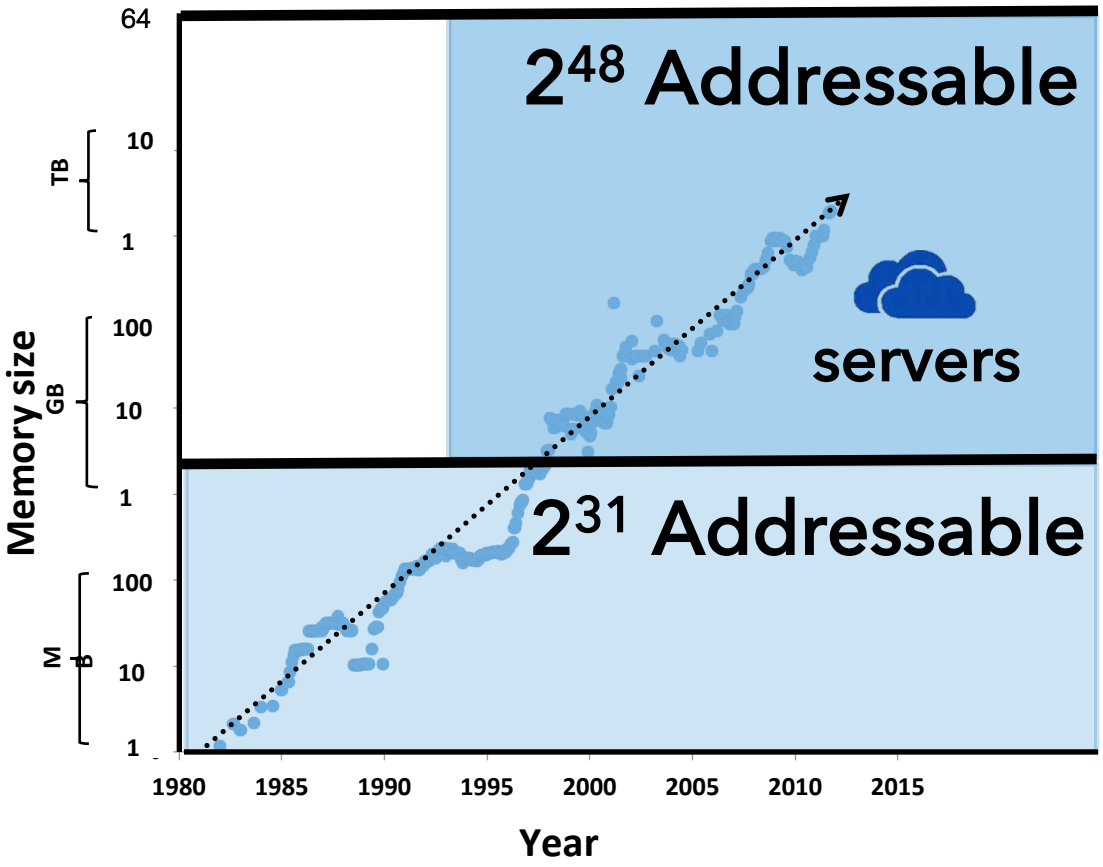
Kathryn S. McKinley Microsoft Research

The Storage Gap



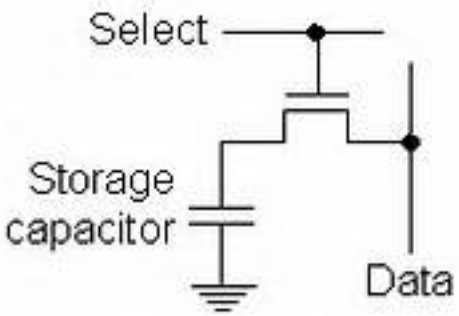
Source: IDC

Memory capacity for $10,000



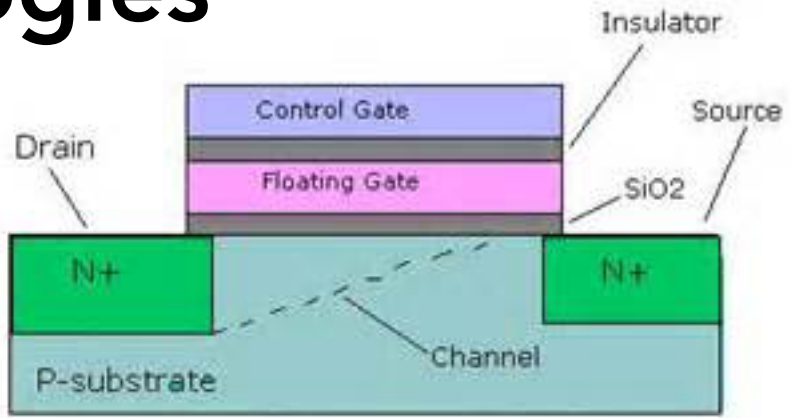
*Inflation-adjusted 2011 USD, from: jcmit.com

Memory technologies



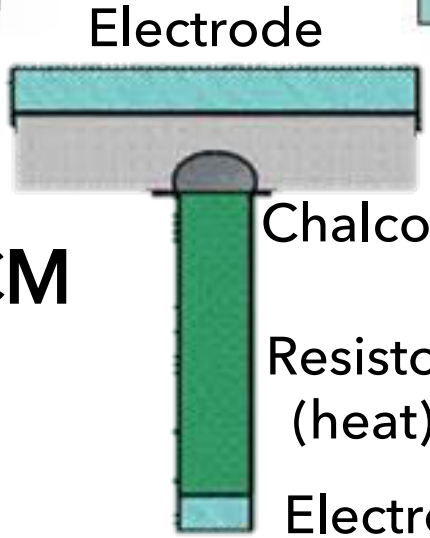
DRAM

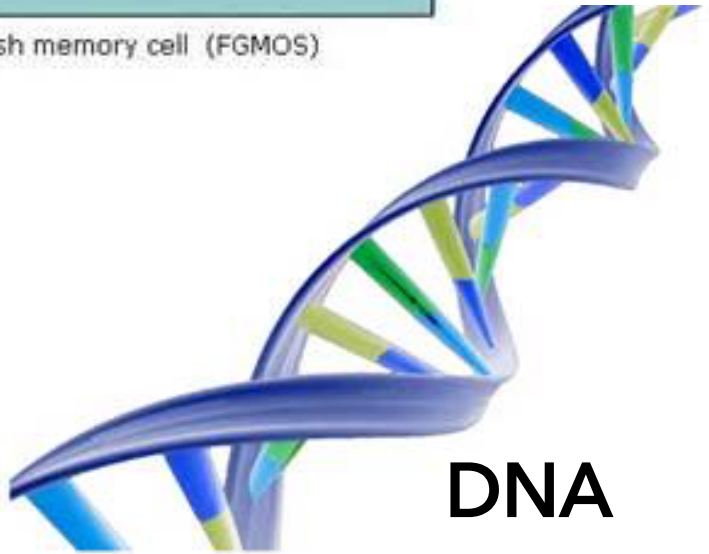
FLASH



A flash memory cell (FGMOS)

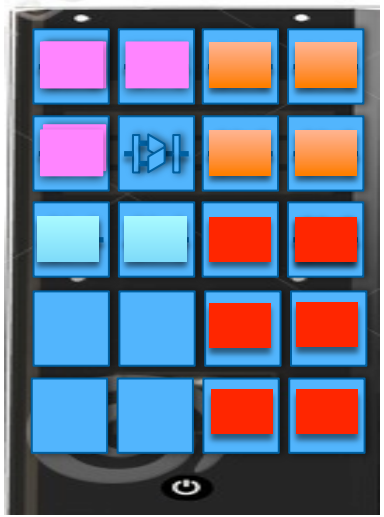
PCM





DNA

Heterogeneous Memory Systems



Mix of
DRAM, Flash, NVM

Memory coupled with
FPGA & special purpose ASIC

Server software

Mobile + Web + Cloud

traditional batch
throughput
+
interactive latency

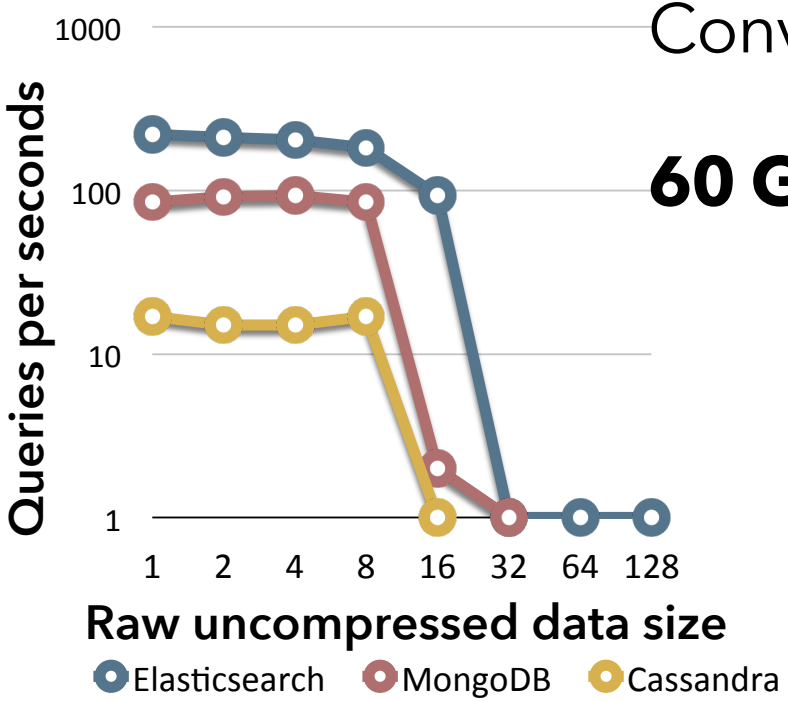
Scaling up & out

Basic
Server Architecture

aggregator

Server workloads scaled to memory sizes & CPUs

Applications must fit in memory



Conviva customer search logs

60 GB Amazon EC2 server, one core

Corollary

no swapping

Source: R. Agarwal [NSDI'15, NSDI'16]

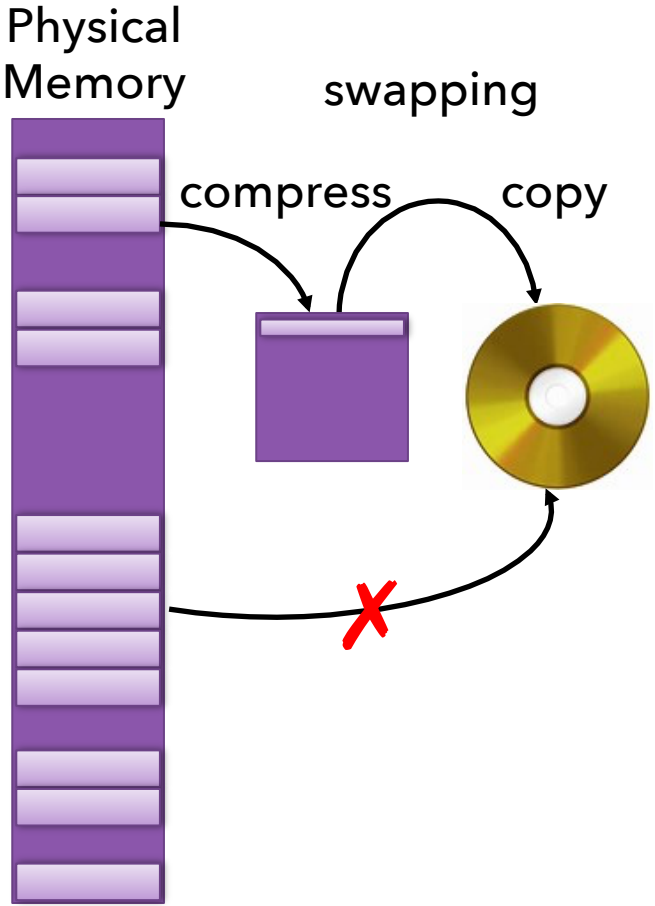
Road map

- Fitting in memory
- Better big memory hardware
- Implications for operating system memory management

Data in memory is compressible

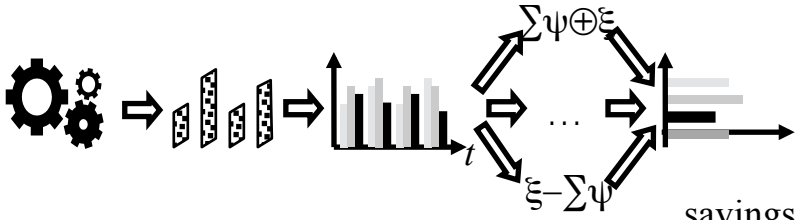
It's now cheaper to compress a page and write it out than write an uncompressed page

Apple & VMware do it

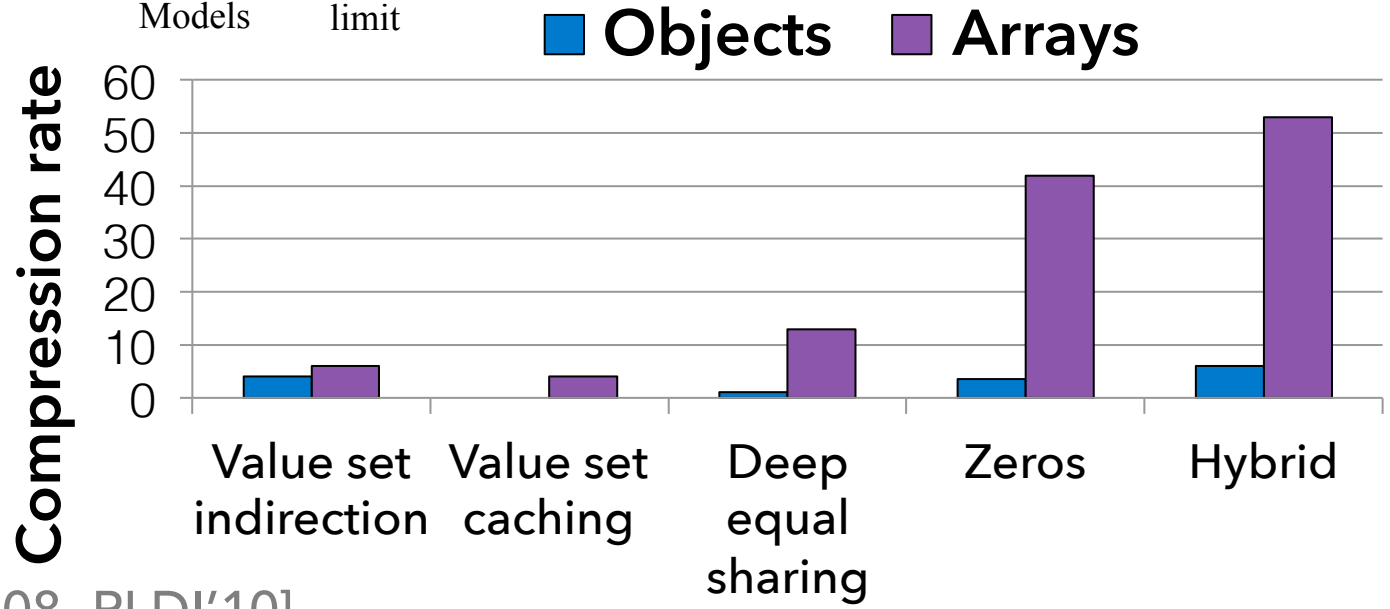


Can we compute on compressed data?

Java limit study



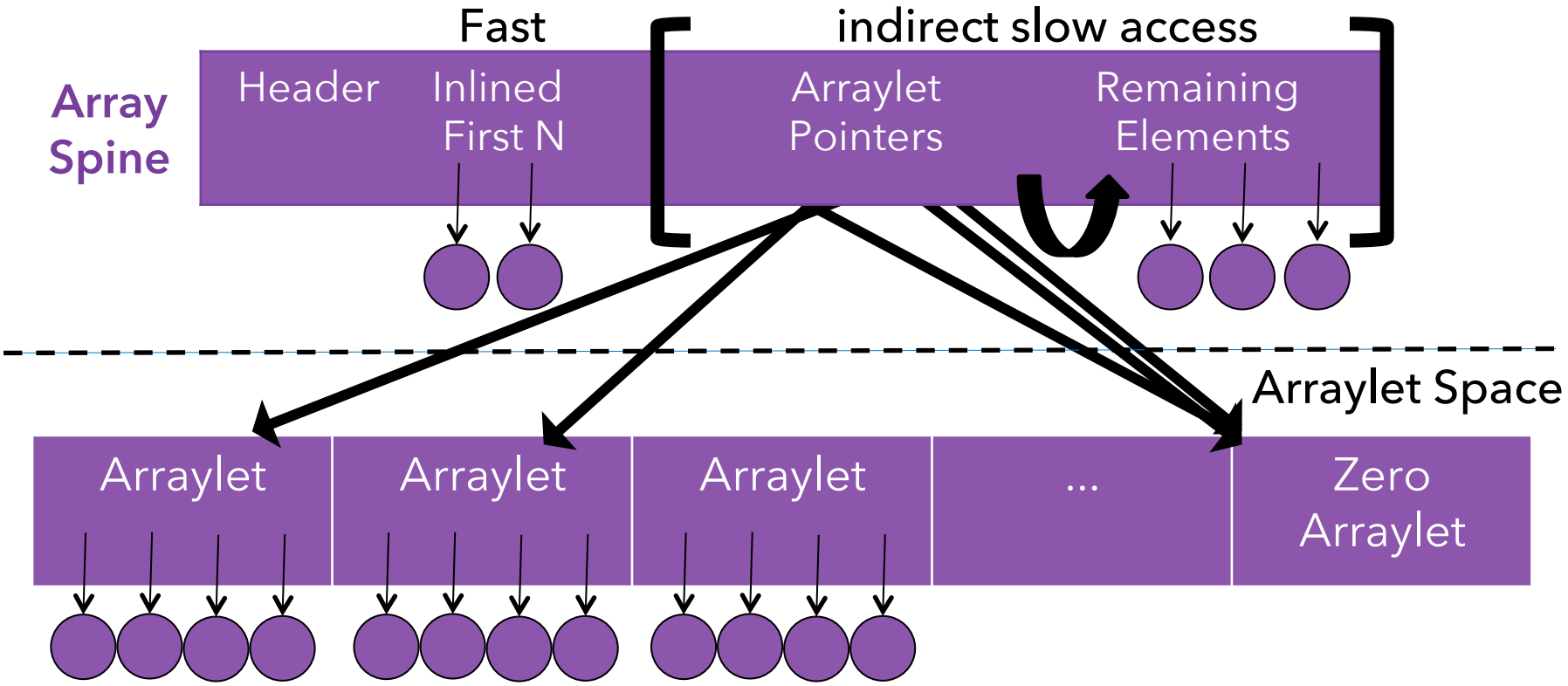
run program Heap dumps Models savings limit



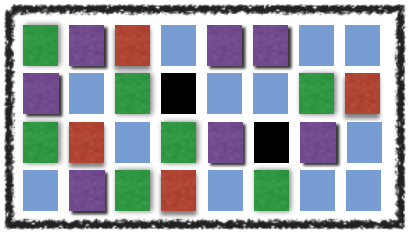
Sartor et al. [ISMM'08, PLDI'10]

Reality of compressed arrays

up to 49% compression, but 6% on average

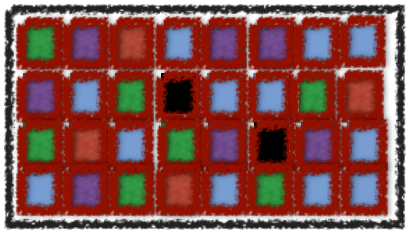


Are databases compressible?



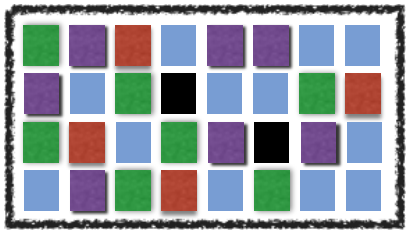
Search( )

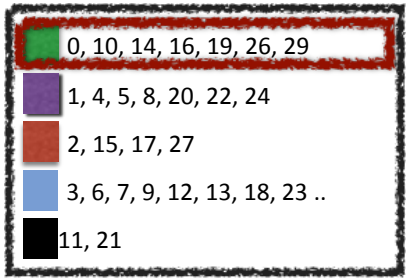
Data Scans



Low storage
Low Throughput

Indexes

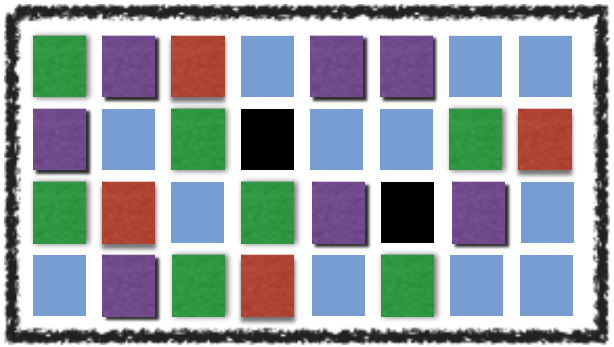


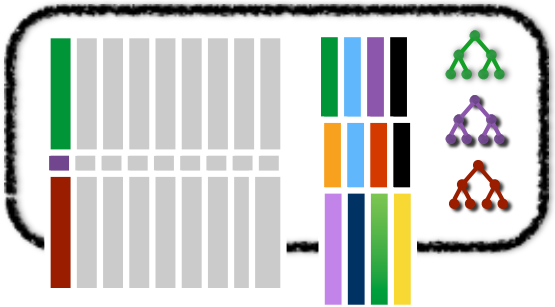


High storage
High Throughput

Source: R. Agarwal [NSDI'15, NSDI'16]

Databases are compressible!





Succinct insight

compression index serves as search index

- Burrows-Wheeler Transform (bzip) suffix arrays
- New data structures & query algorithms

Source: R. Agarwal [NSDI'15, NSDI'16]

Advantages

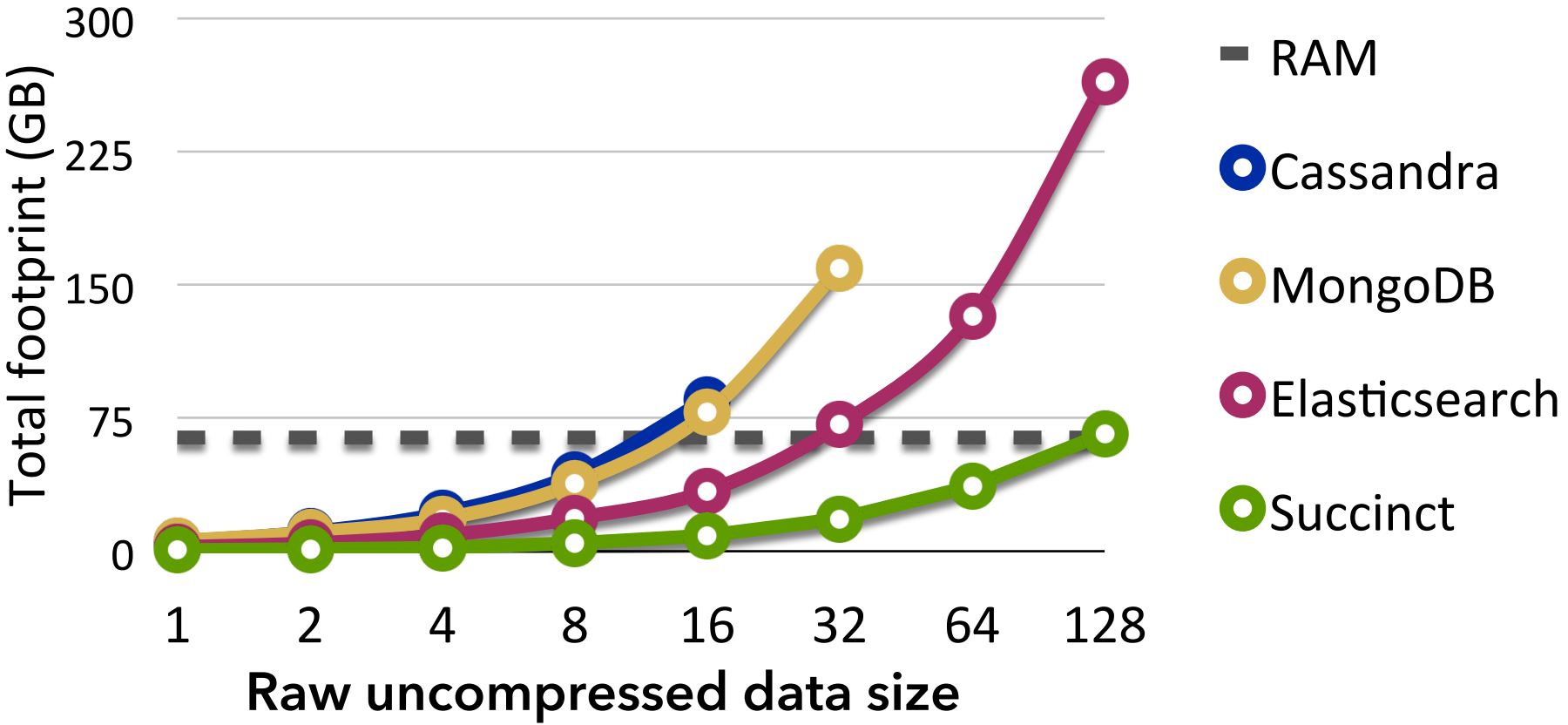
- Compression index is the data index!
- No scans
- No decompression
- Queries search, range, random access, regular expressions

Limitations

- More CPU operations for data access
- No scans
- No in-place updates
- Preprocessing time

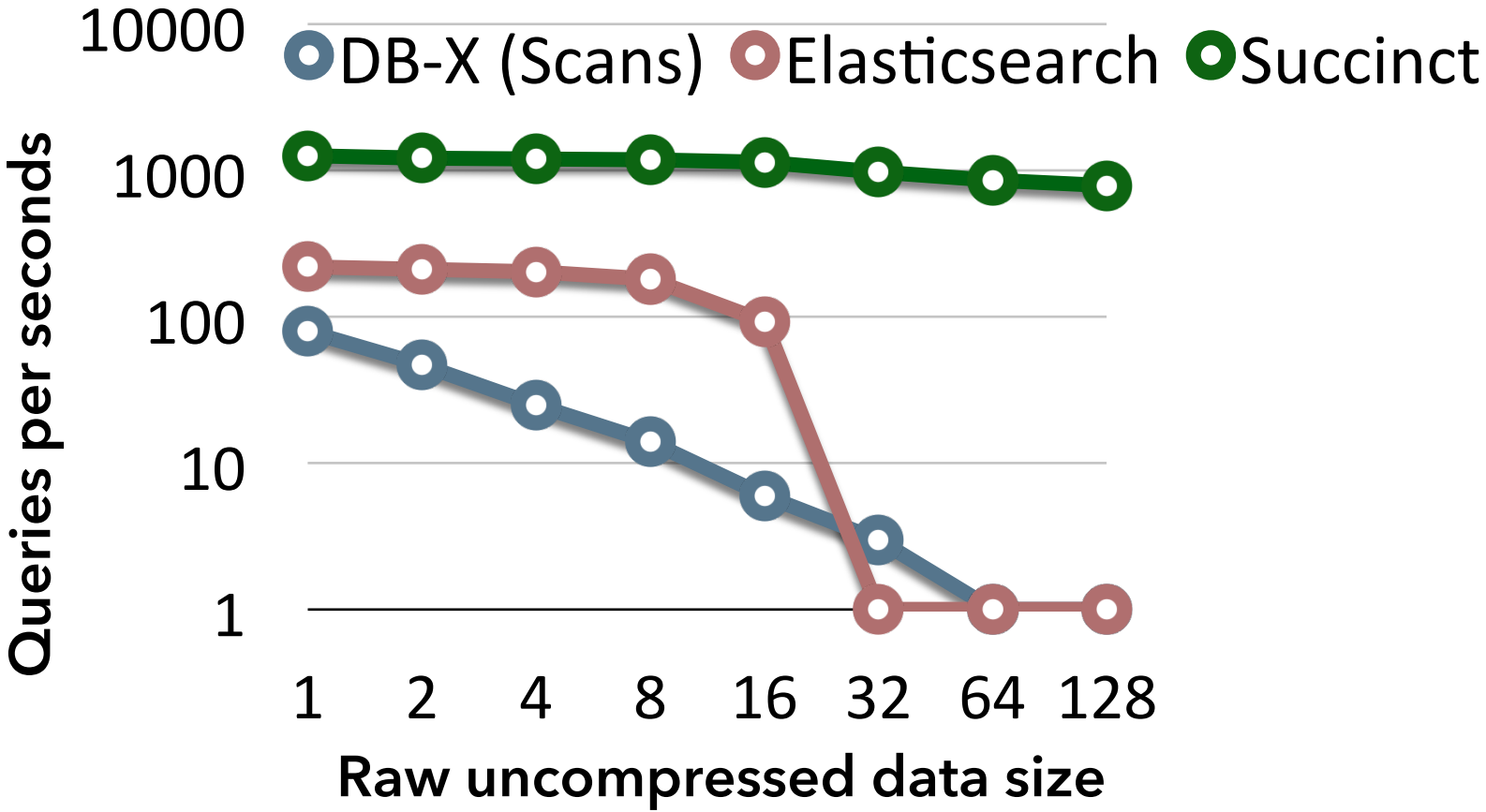
Source: R. Agarwal [NSDI'15, NSDI'16]

Order of magnitude more data



Source: R. Agarwal [NSDI'15, NSDI'16]

Throughput


10000
1000
100
10
1
DB-X (Scans)
Elasticsearch
Succinct
Queries per seconds
1
2
4
8
16
32
64
128
Raw uncompressed data size

Fitting in memory is necessary, but not sufficient

Road map

- Fitting in memory
- Better big memory hardware
- Implications for operating system memory management

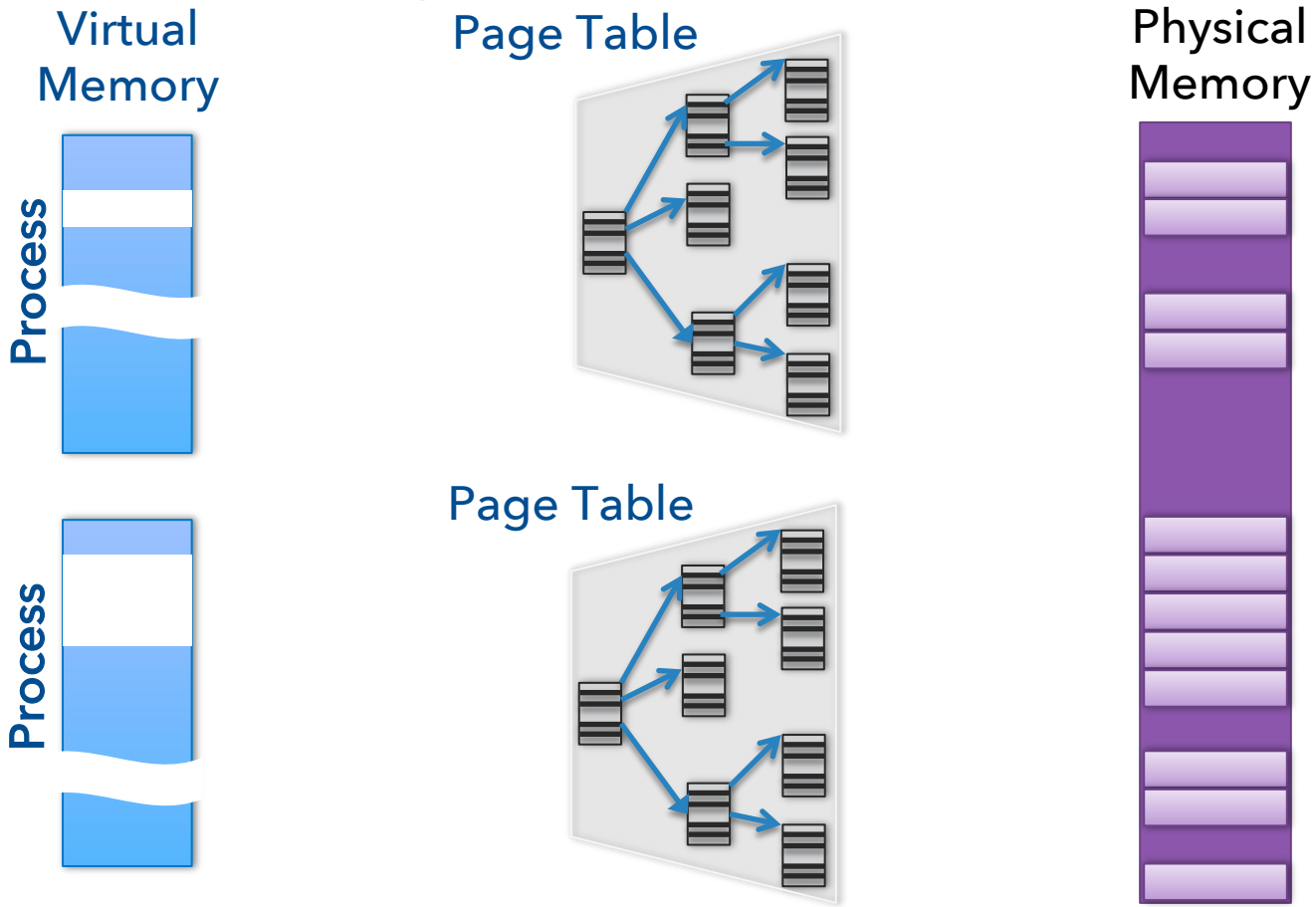
Scaling up & out

Basic
Server Architecture

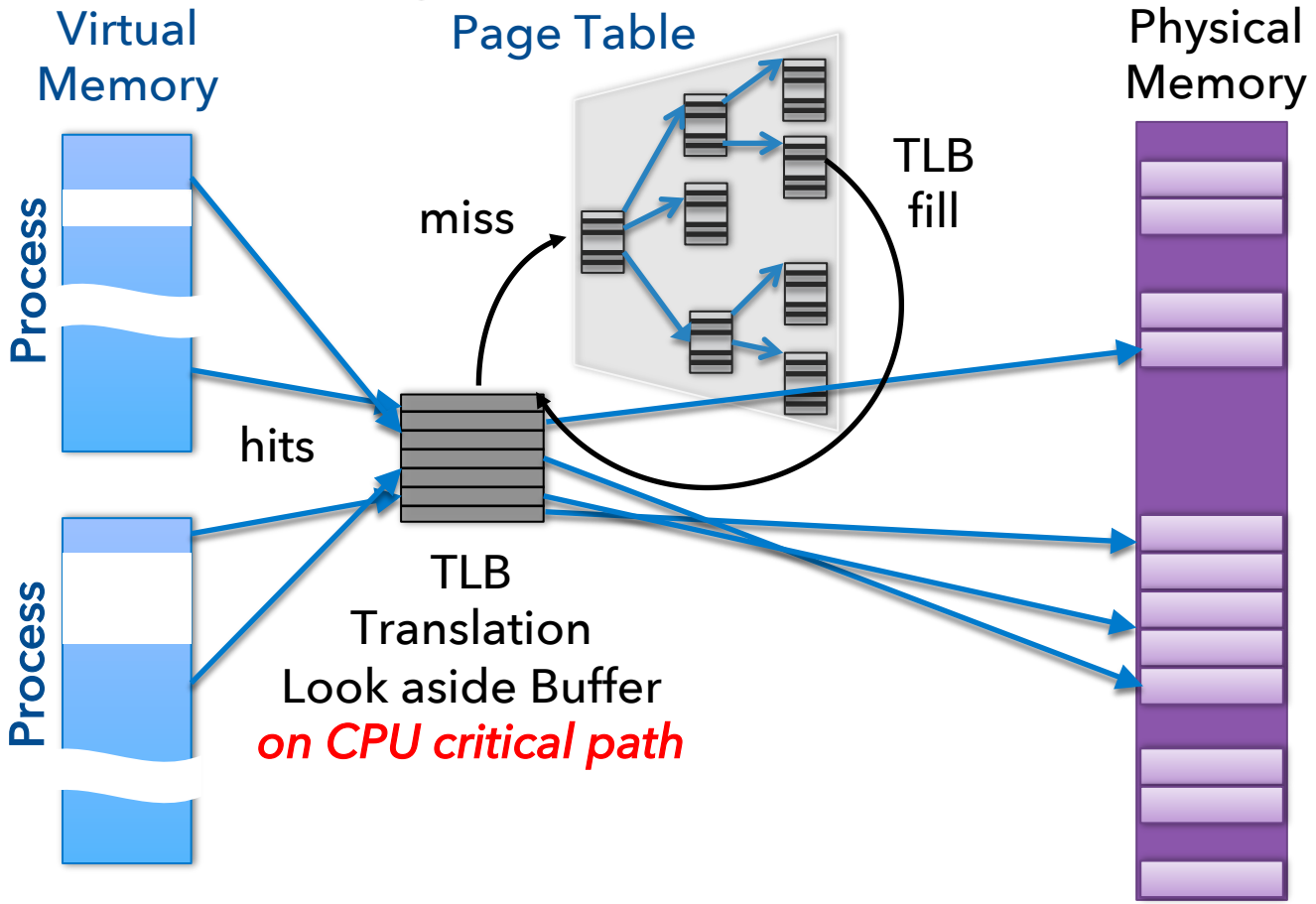
aggregator

Server workloads scaled to memory sizes & CPUs

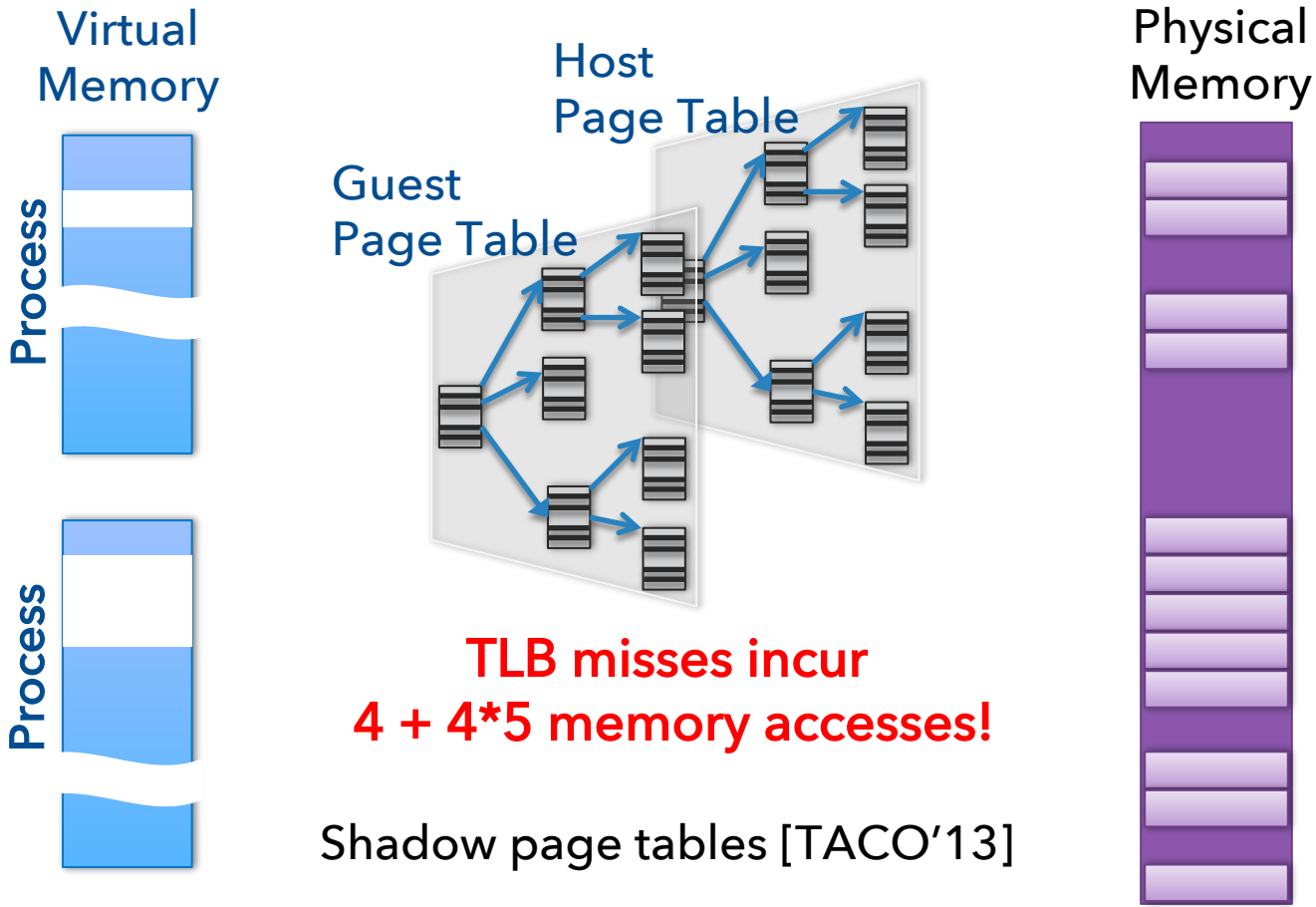
Hardware organization



Hardware organization



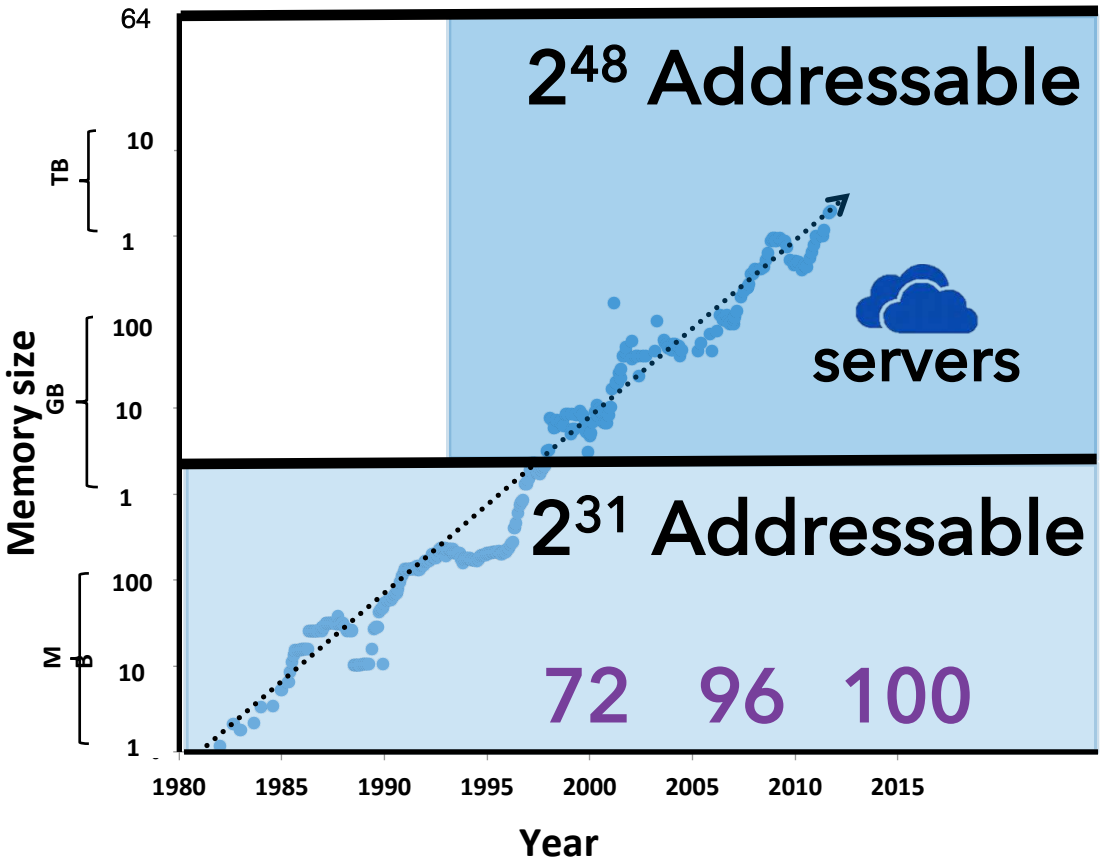
Virtualization exacerbates TLB limitations

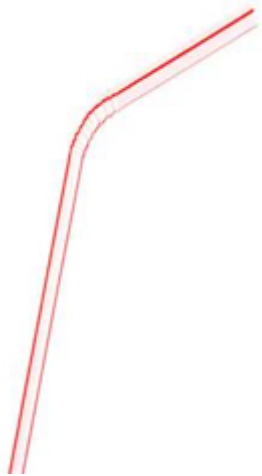


TLB misses incur
4 + 4*5 memory accesses!

Shadow page tables [TACO'13]

Memory capacity for $10,000





TLB L1 Entries

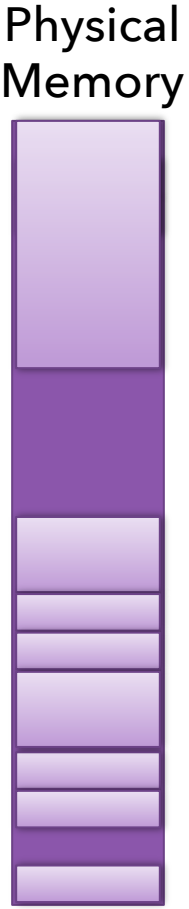
*Inflation-adjusted 2011 USD, from: jcmit.com

Bigger pages extend TLB reach

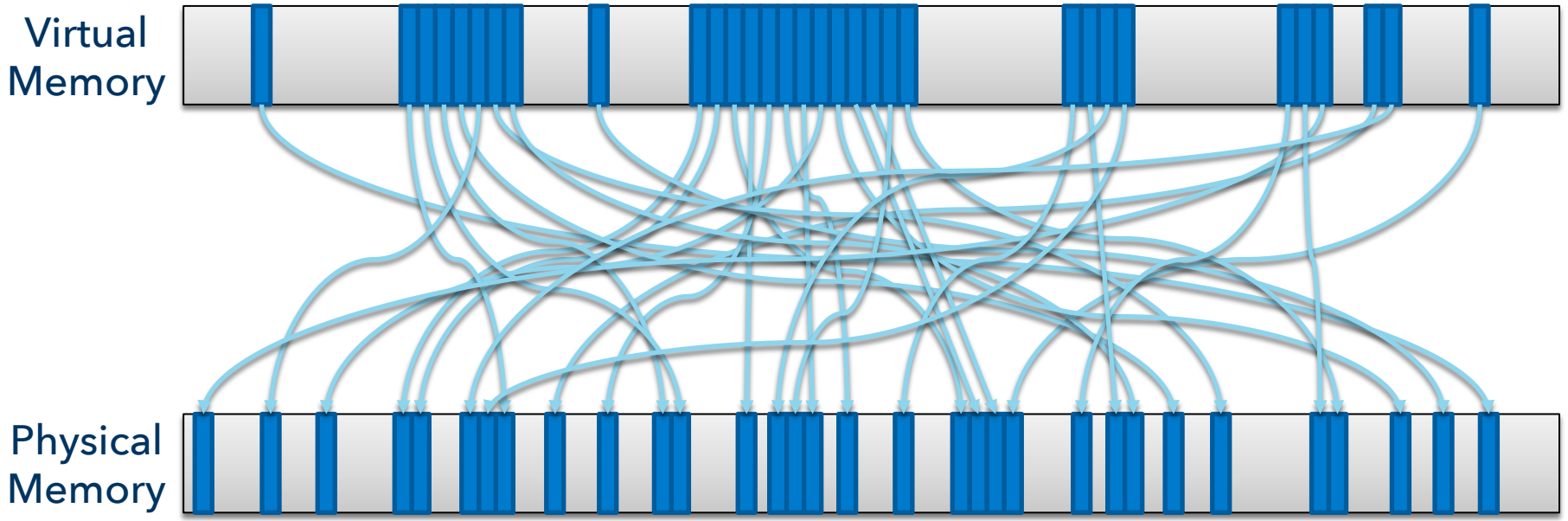
1 MB 1 GB *aligned*

- Explicit
  Programmer burden
- Transparent
  Internal fragmentation up to .5 page size depending on policy

One size does not fit all!



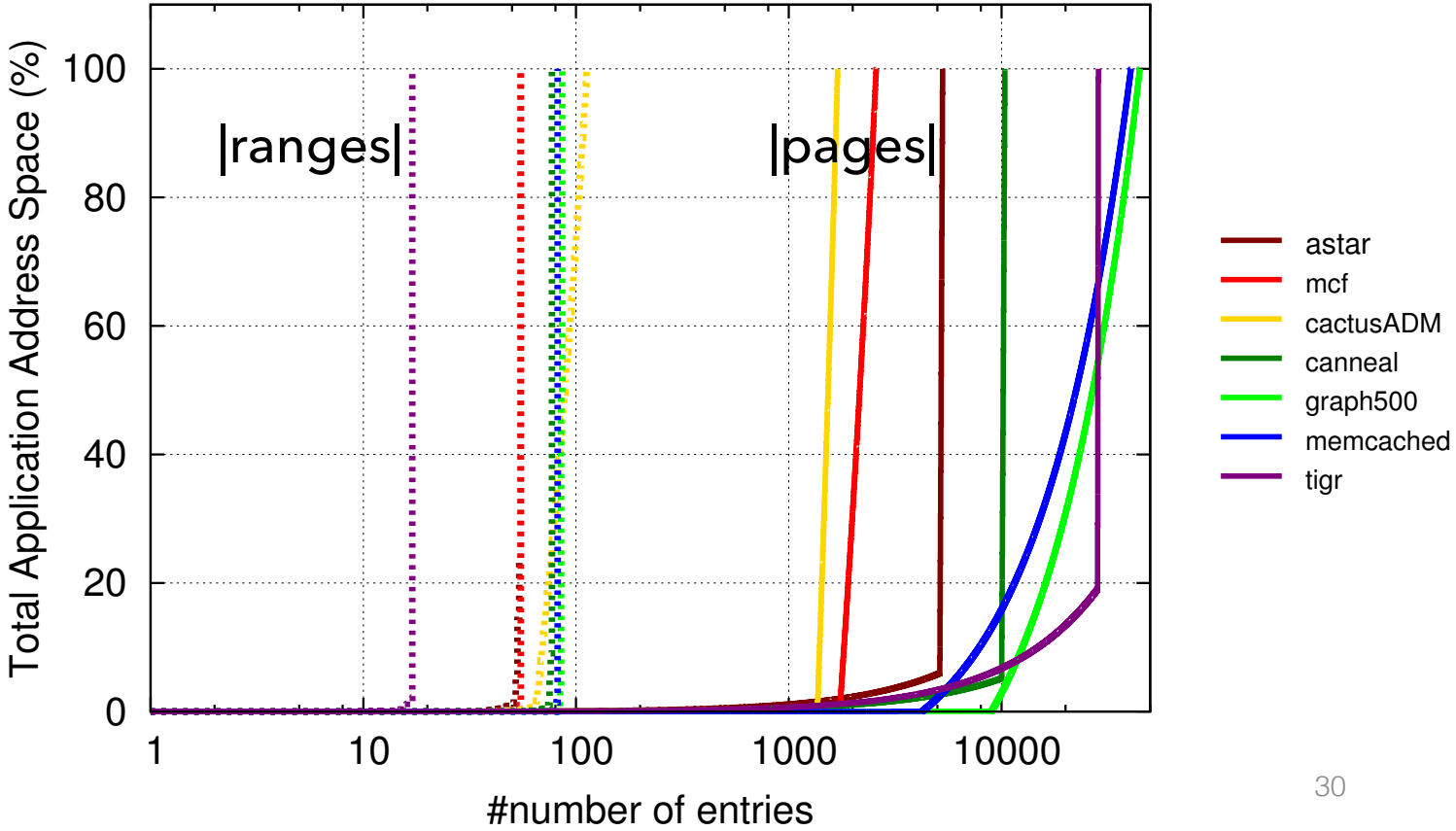
Paged memory



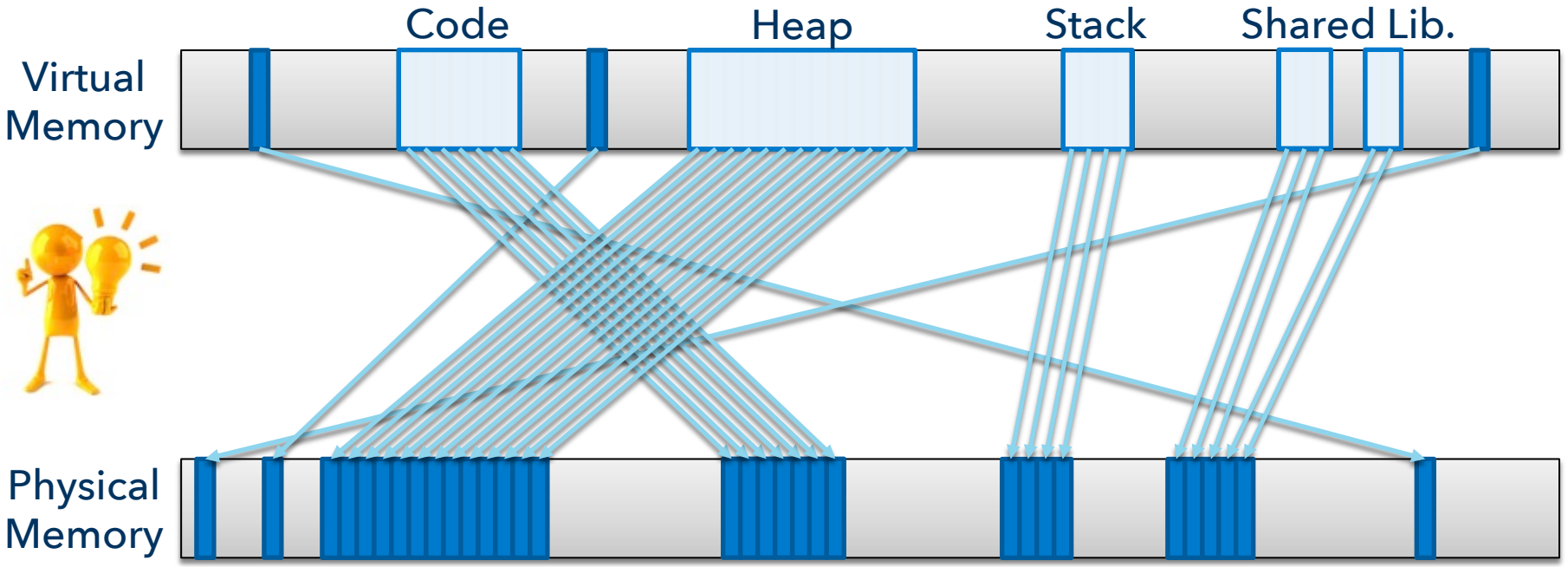
Key observation: ranges of contiguity

How much range contiguity is there?

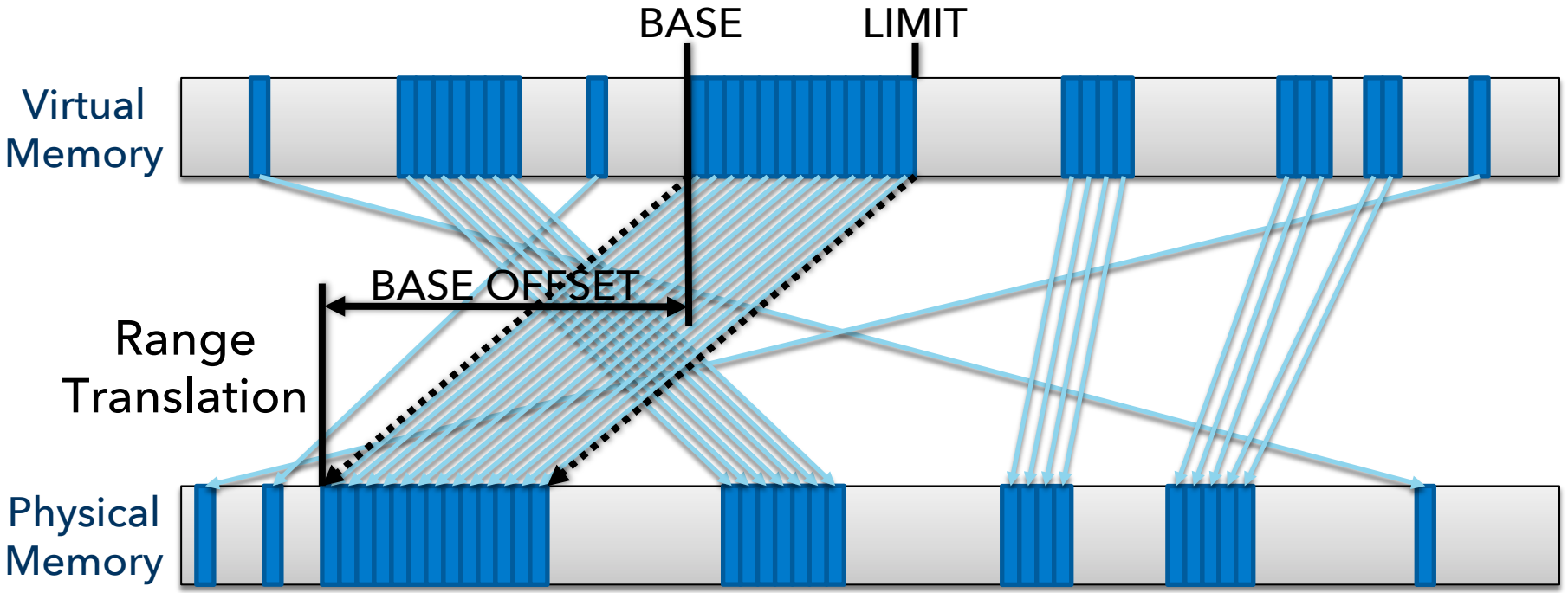
|Ranges| vs |Pages|



Idea: Redundant memory mappings

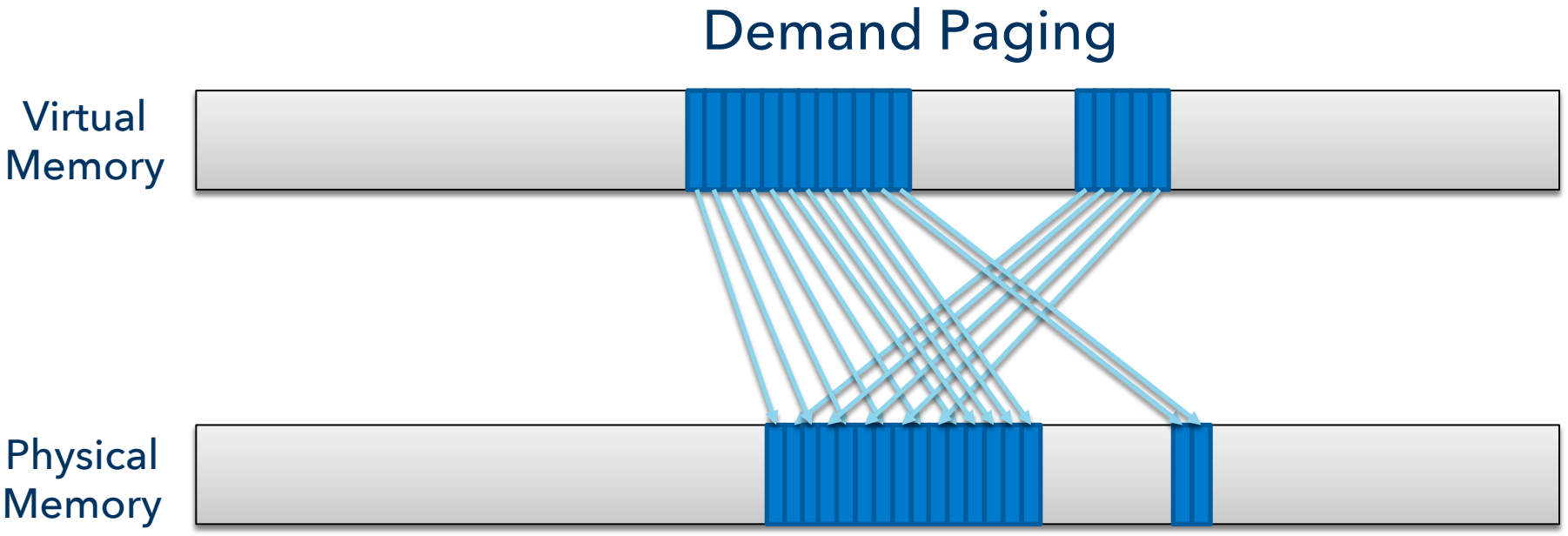


Range translations: Base, Limit, Offset



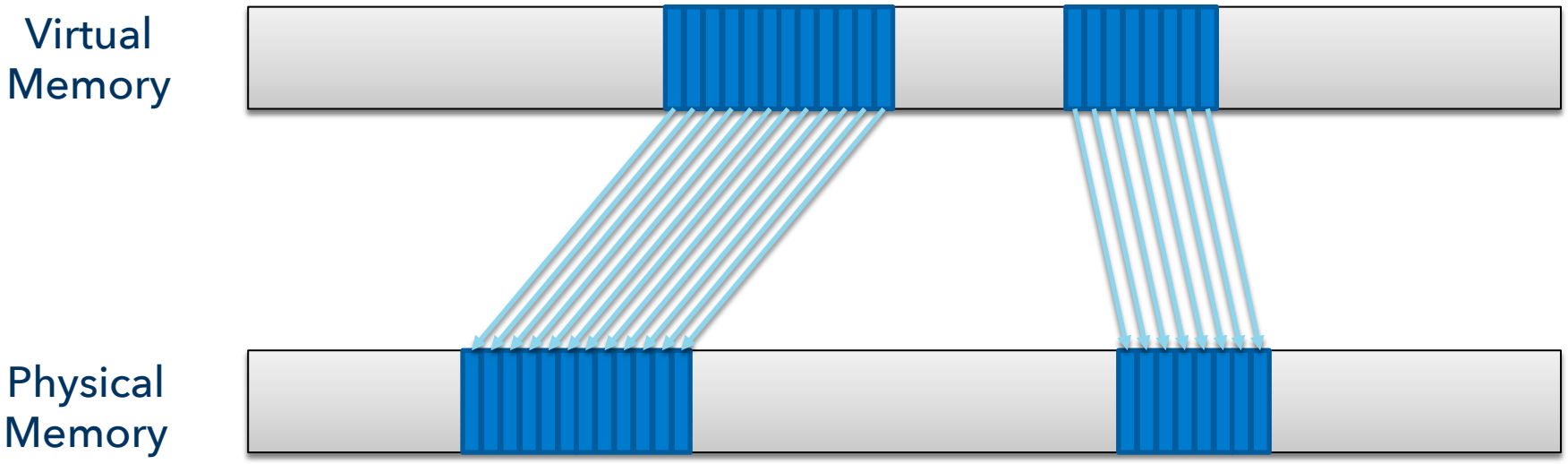
Range Translation: *maps contiguous virtual pages to contiguous physical pages with uniform protection*

OS memory management support



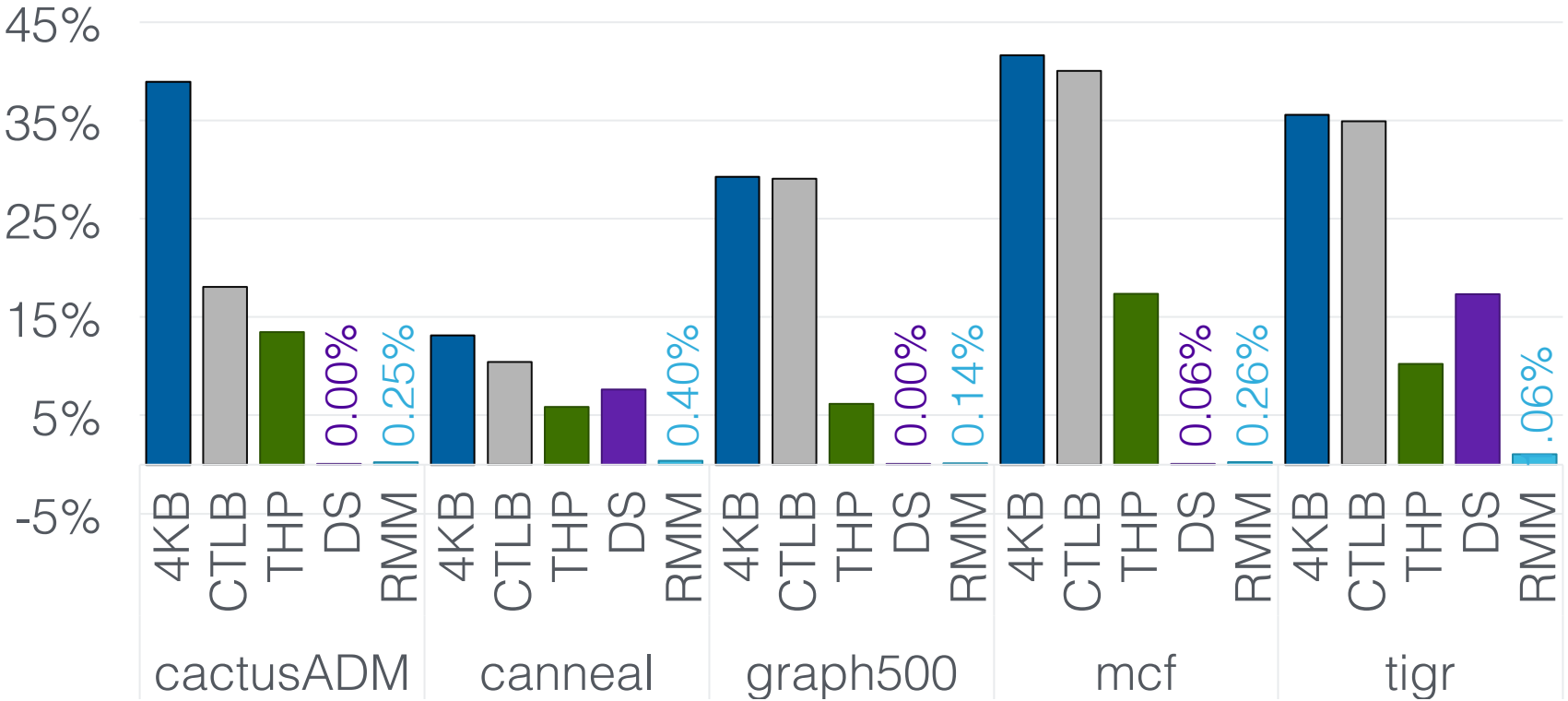
Does not facilitate physical range creation

Eager paging



Increases range sizes, but can waste memory

It works: eliminates VM overheads



ISCA'15, IEEE MICRO Top Picks

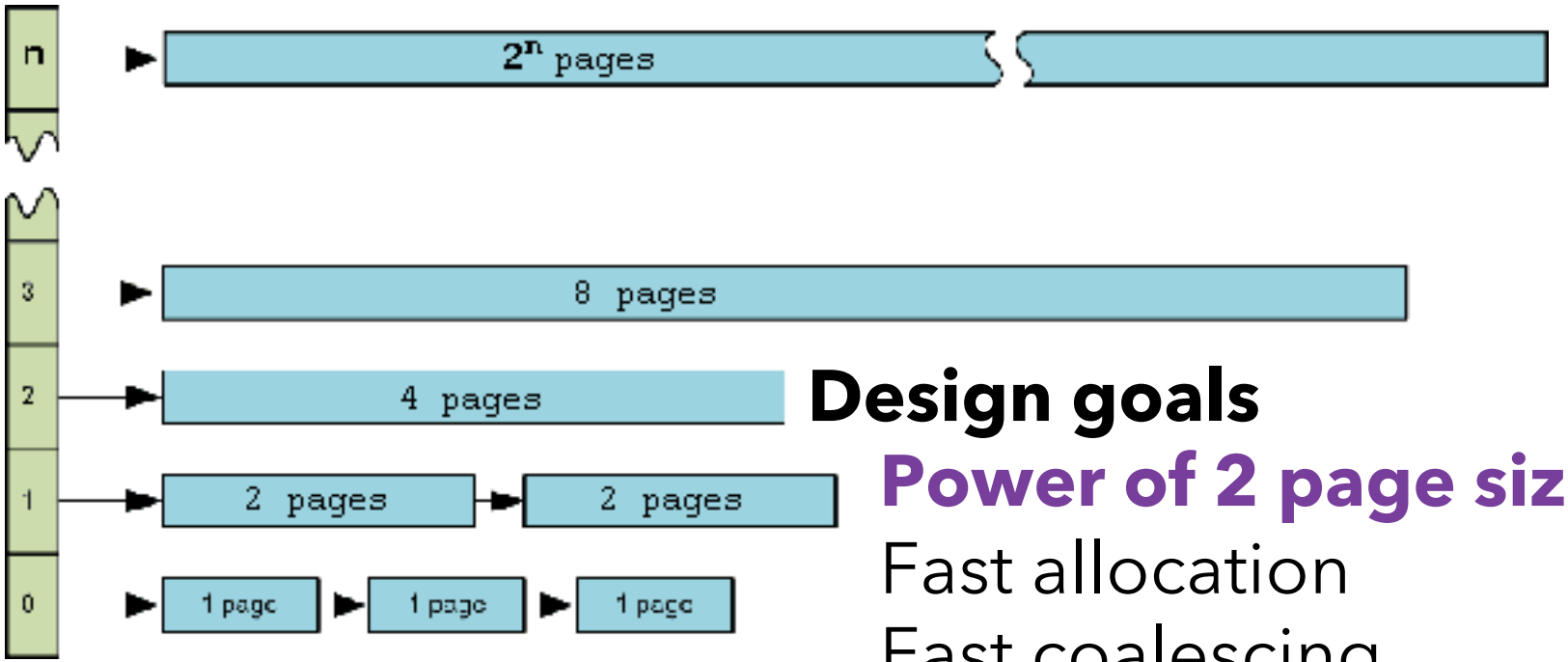
Road map

- Fitting in memory
- Better big memory hardware
- Implications for operating system memory management

Variable page sizes and ranges require contiguity

base pages do not

OS Buddy allocation



Design goals

Power of 2 page sizes

Fast allocation

Fast coalescing

Source: codeproject.com

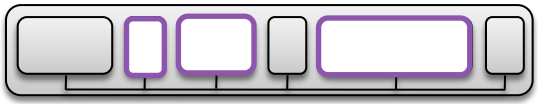
Memory management is a space tradeoff

Lessons from garbage collection

GC Fundamentals

Allocation

Free List



Bump Allocation

Identification

Tracing
(*implicit*)

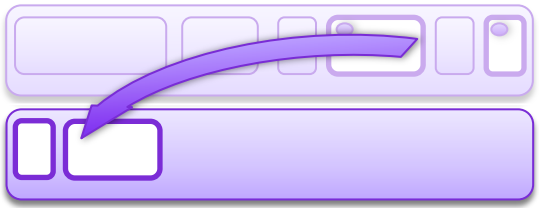
Reference Counting
(*explicit*)

Reclamation

Sweep-to-Free

Compact

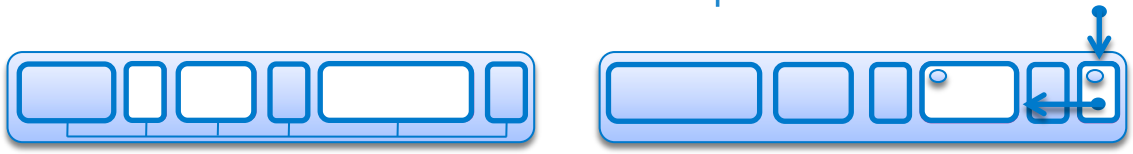
Evacuate



Canonical Garbage Collectors

Mark-Sweep [McCarthy 1960]

Free-list + trace + sweep-to-free

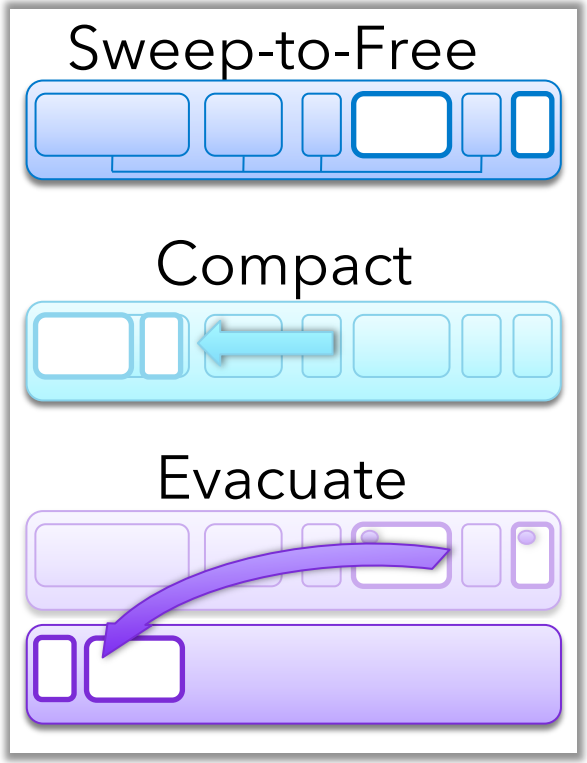


Mark-Compact [Styger 1967]

Bump allocation + trace + compact

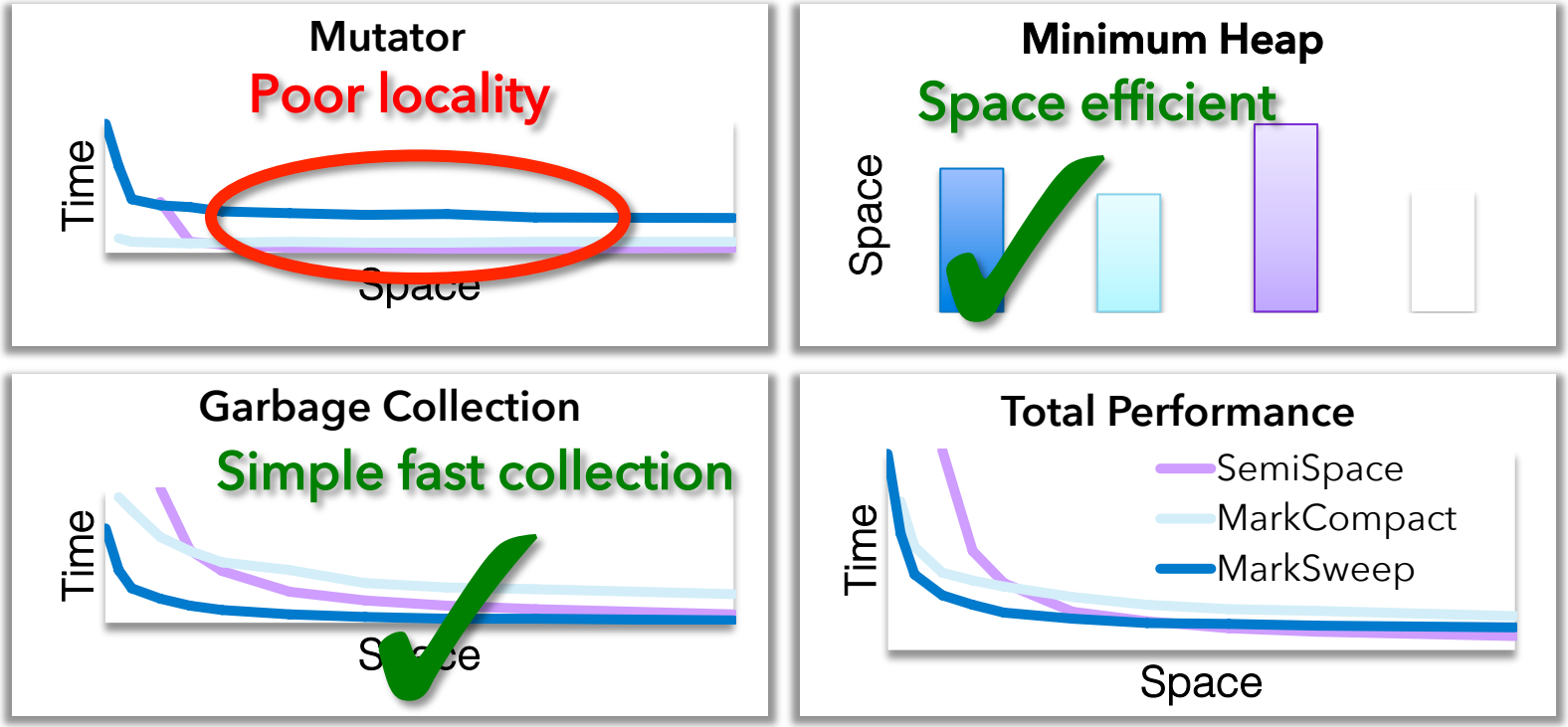
Semi-Space [Cheney 1970]

Bump allocation + trace + evacuate



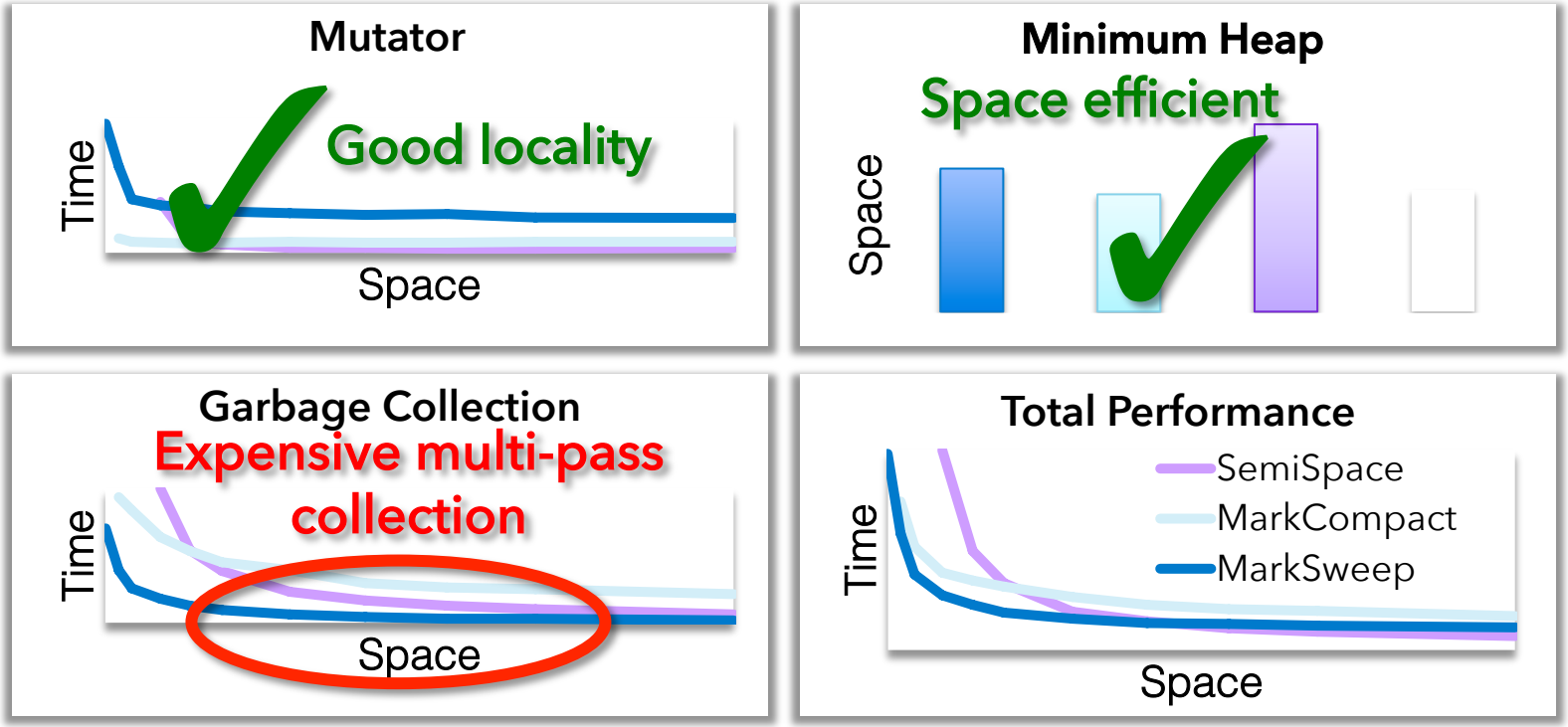
Mark-Sweep

Free List Allocation + Trace + Sweep-to-Free



Mark-Compact

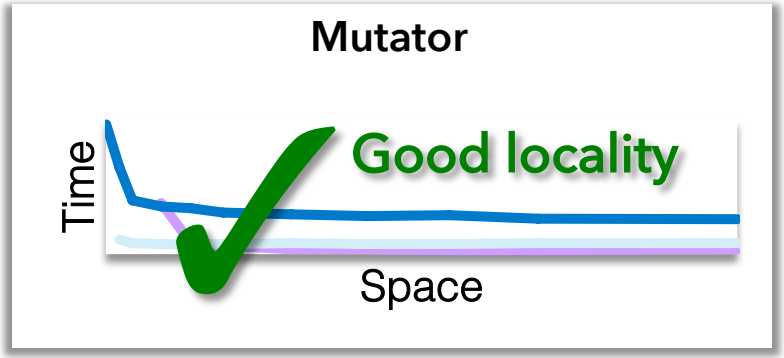
Bump Allocation + Trace + Compact

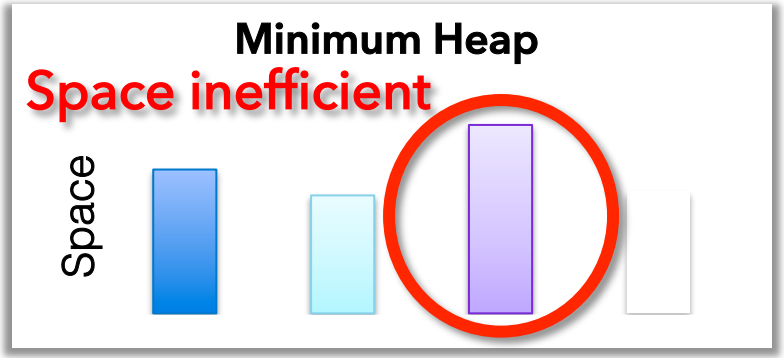


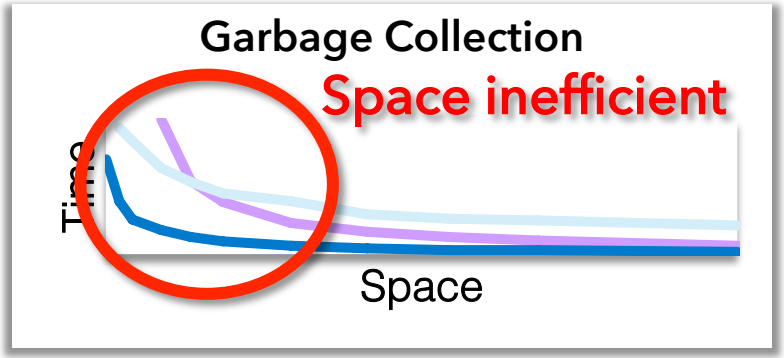
Actual data, taken from geomean of DaCapo, jvm98, and jbb2000 on 2.4GHz Core 2 Duo

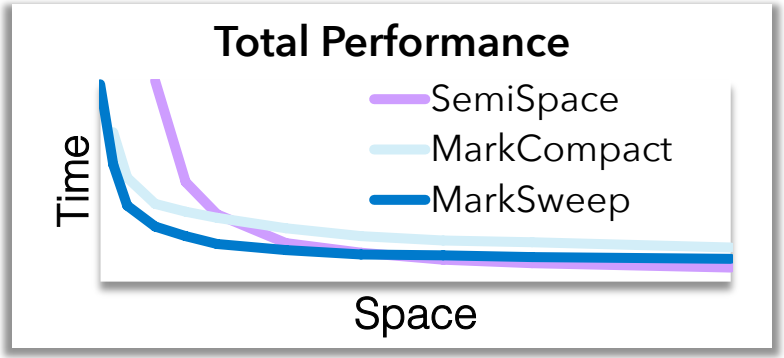
Semi-Space

Bump Allocation + Trace + Evacuate



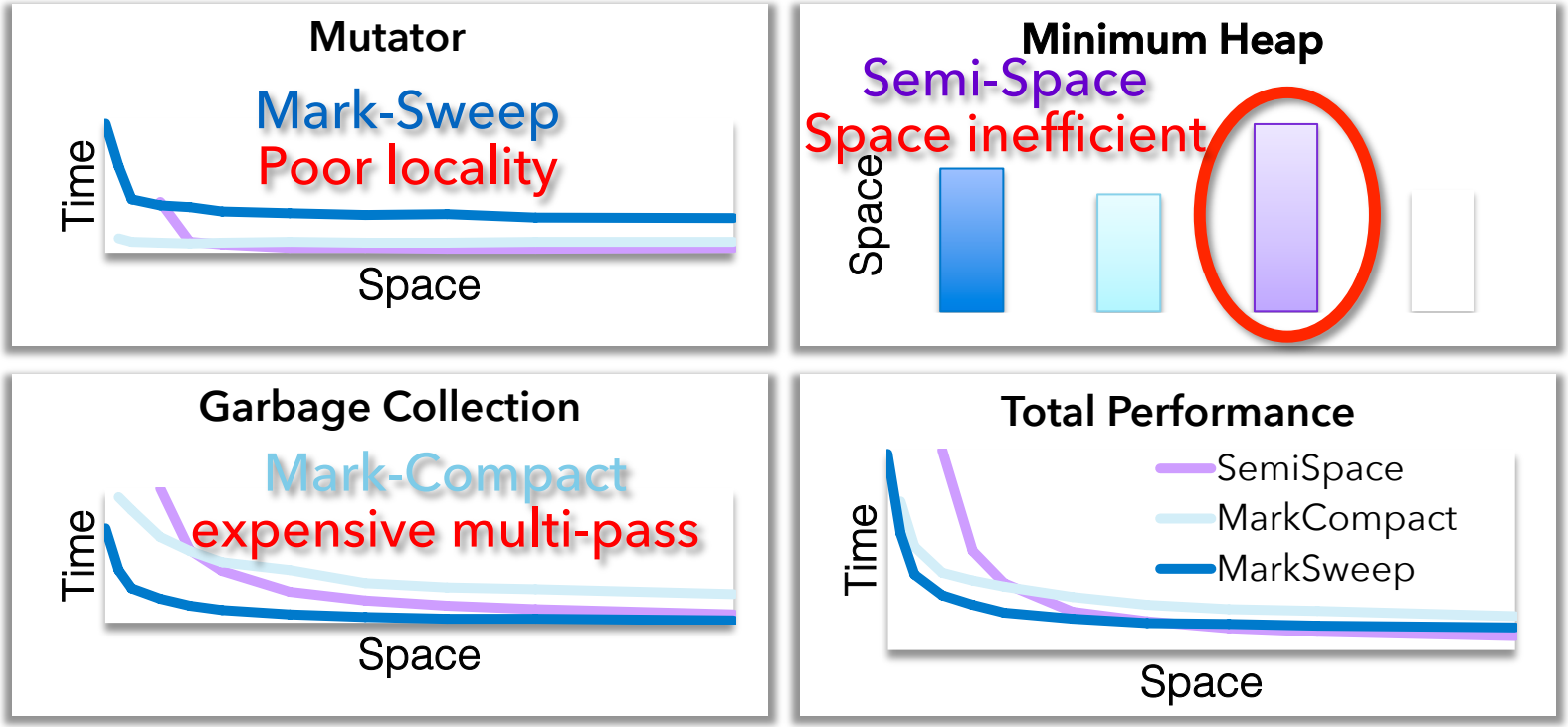






Performance Pathologies

Mark-Sweep, Mark-Compact, Semi-Space



Actual data, taken from geomean of DaCapo, jvm98, and jbb2000 on 2.4GHz Core 2 Duo

Mark-Region

with Sweep-To-Region

Mark-Sweep

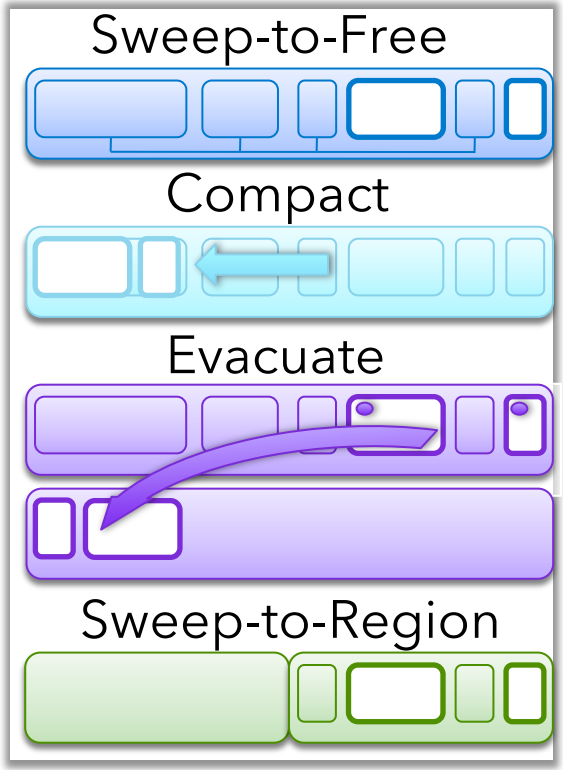
Mark-Compact

Semi-Space

Mark-Region

Bump + trace + sweep-to-region

Reclamation



Naïve Mark-Region

- Contiguous allocation into regions

  ✓ Excellent locality

  Objects cannot span regions

- Simply mark objects and their region
- Free unmarked regions

Region Size?

Large Regions

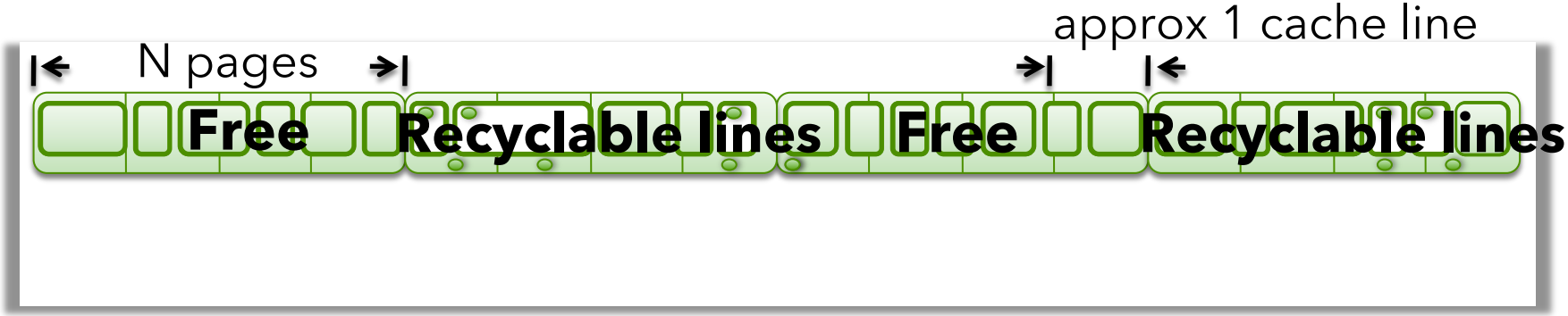
✓ More contiguous allocation ✗ Fragmentation (false marking)

Small Regions

✗ Increased metadata ✗ Fragmentation (can't fill blocks)

✗ Constrained object sizes

Hierarchy: lines and blocks



- TLB locality, cache locality
- Objects span lines

✓ Less fragmentation

- Block > 4 X max object size
- Lines marked with objects

✓ Fast common case

Allocation Policy
(Recycling)

- Recycle partially marked blocks first
  - ✓ Minimizes fragmentation
  - ✓ Maximizes contiguity
  - ✓ Shares free blocks among parallel thread allocators

Opportunistic Defragmentation

- Opportunistically *evacuate* fragmented blocks
  - Lightweight, uses same allocation mechanism
  - No cost in common case (specialized GC)
- Identify source and target blocks
- Evacuate objects in source blocks

  Allocate into target blocks
- Opportunistic: Leave in place if no space, or object pinned

Immix Mark-Region

128 byte lines, 32 KB blocks

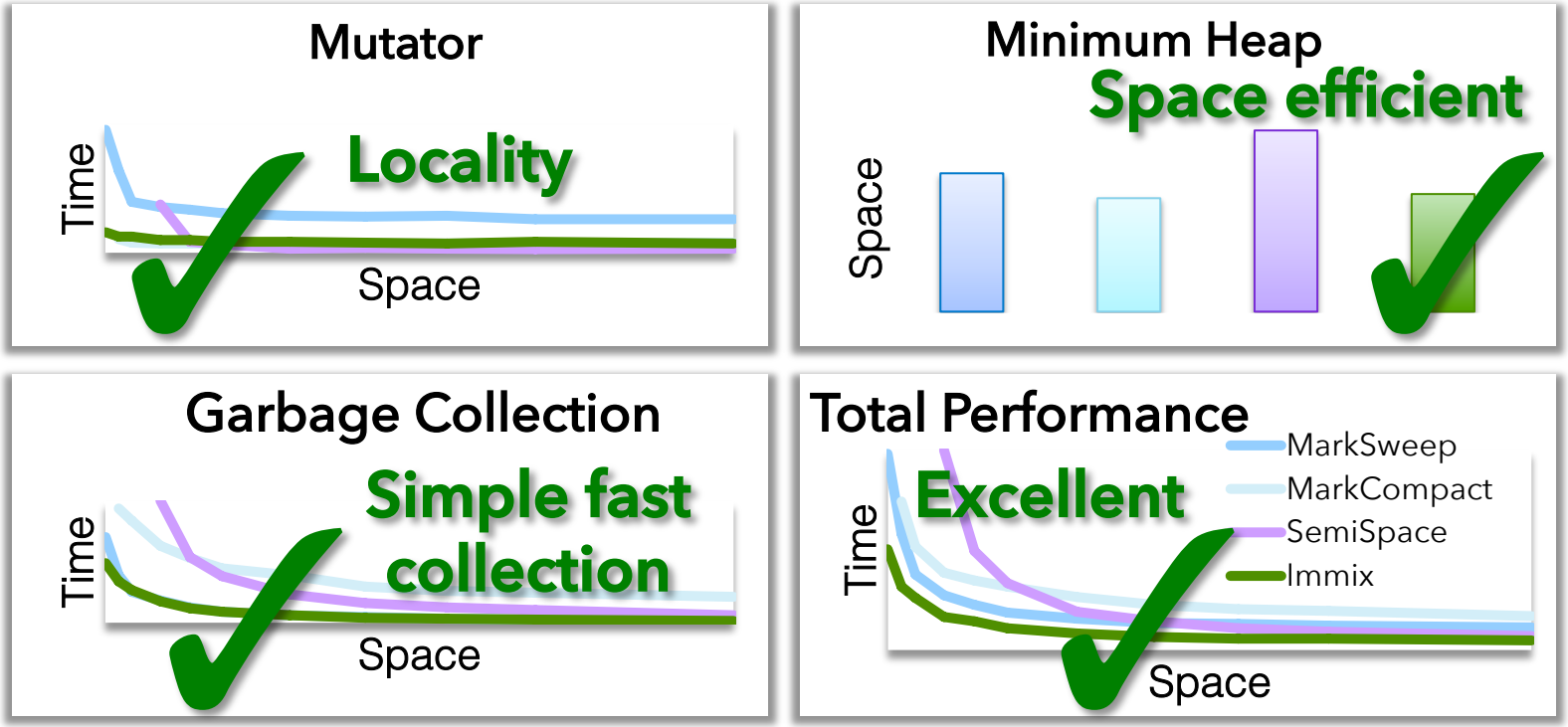
Opportunistic defragmentation triggered if there exist partially full blocks at memory exhaustion

Overflow allocation of medium objects (> 128 bytes)

Implicit marking

Immix

Bump Allocation + Trace + Sweep-to-Region



Actual data, taken from geomean of DaCapo, jvm98, and jbb2000 on 2.4GHz Core 2 Duo

Lessons for OS

Metadata Immix is more space & time efficient with 1.5% of metadata than all the zero metadata algorithms.

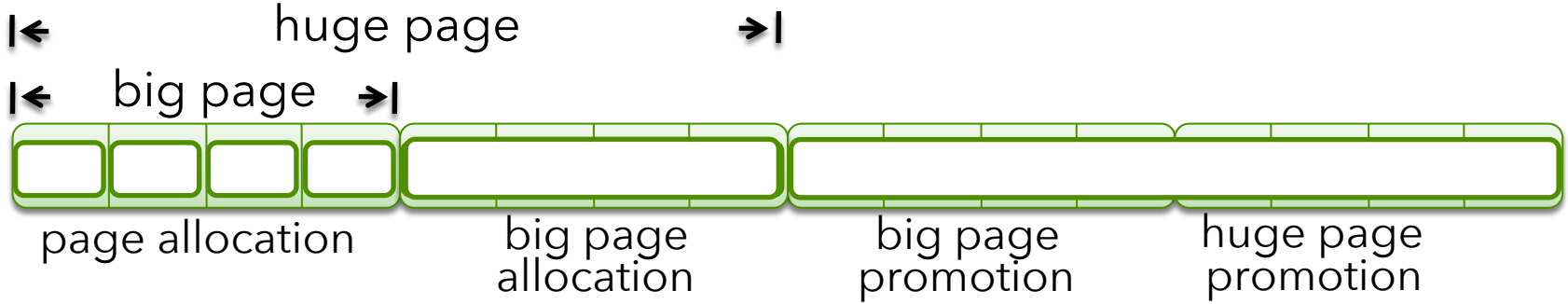
Contiguity Applications allocate & access sequential virtual addresses. Immix exploits this property.

Why not the OS?

Looking forward
Opportunities for OS memory management

Mark Region (MR) for Multiple Page Sizes

page sizes = region sizes



page = MR line

big page = multiple pages = MR block (aligned)

huge page = multiple big pages = multiple MR blocks, etc.

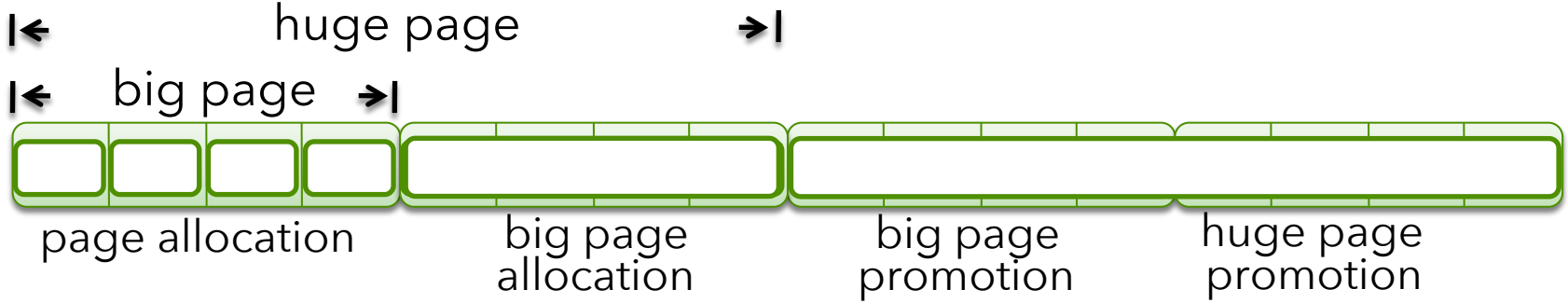
Fragmentation management - better, block statistics always correct

prevention allocate into occupied blocks w/ free pages

defrag target & source blocks

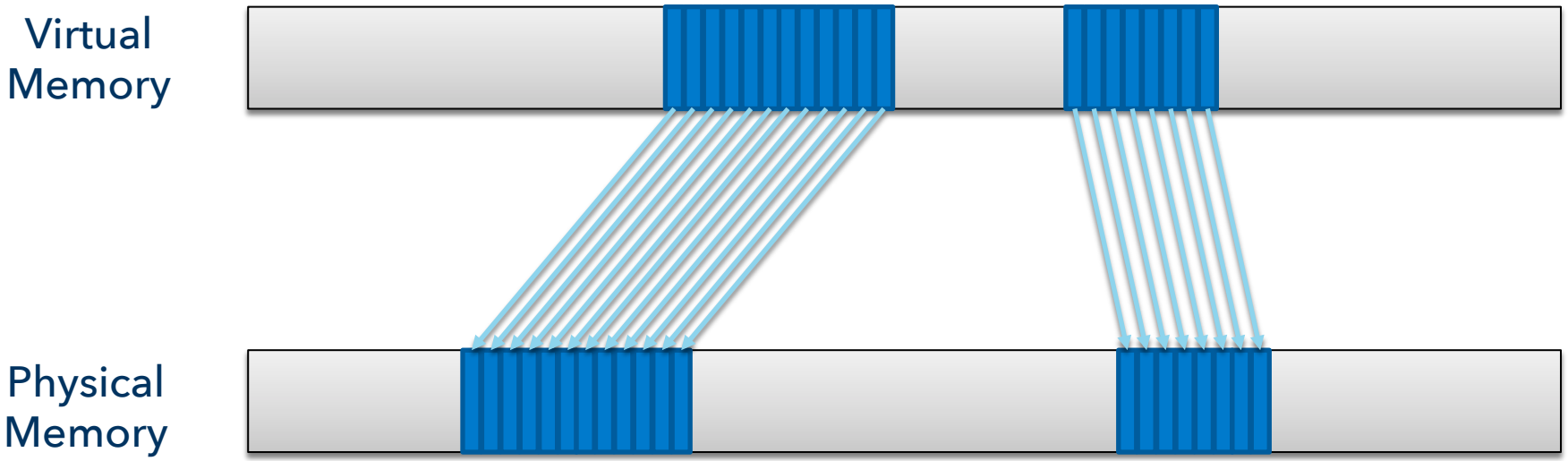
Incremental adoption in buddy allocator

buddy sizes = page sizes



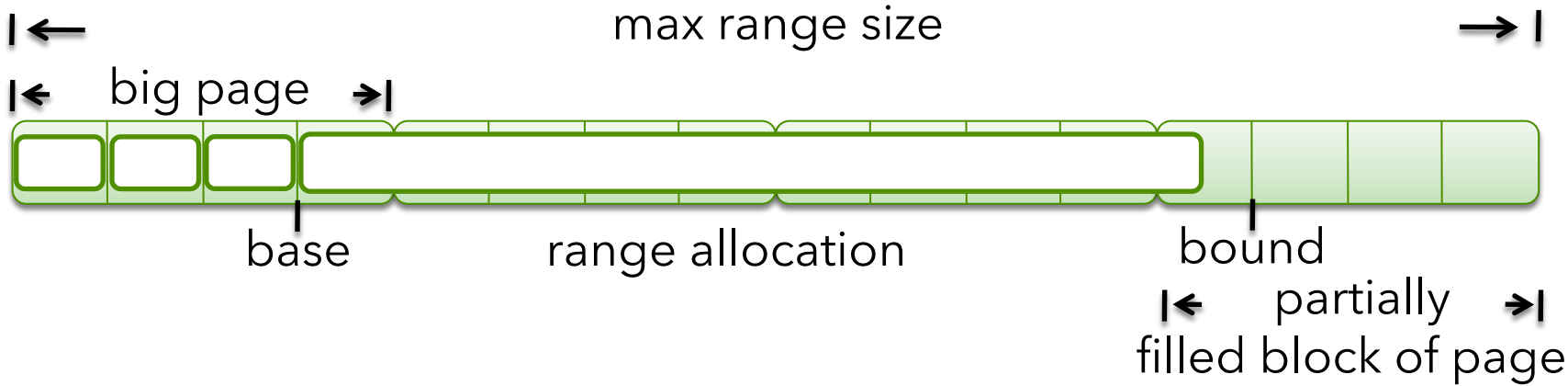
- **Eliminate all “buddy” sizes ≠ potential page sizes**
- **Add metadata for blocks and pages**
- **Add allocation and defragmentation policies**

What about range allocation?



Mark Region for Ranges

max range size = biggest block size



Range allocation – overflow allocation, harder due to variable sizes

Fragmentation management

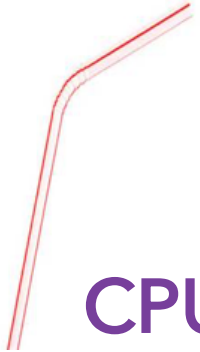
prevention same, allocate into occupied blocks w/ free pages

defrag same mechanisms with new incremental policies

that consider multiple page sizes and maximum range size

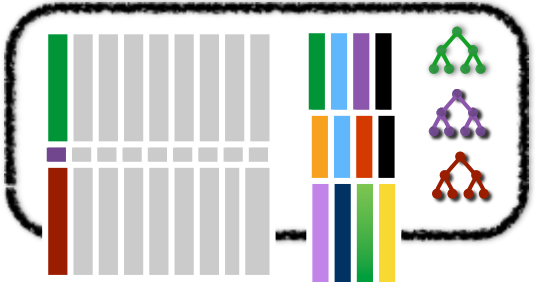
Lots of challenges

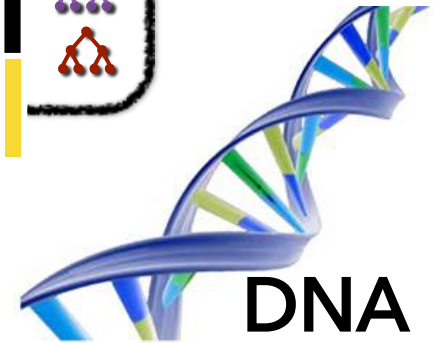
Heterogeneous memories



CPU constraints

memory management





DNA

Collaborators maybe YOU!

Steve Blackburn Karin Strauss

Vasileios Karakostas Jayneel Gandhi

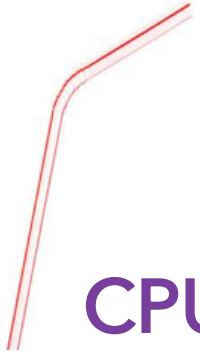
Mark D. Hill Michael M. Swift Osman S. Ünsal

Mario Nemirovsky Furkan Ayar Adrián Cristal

Jennifer Sartor Martin Hirzel Tiejun Gao

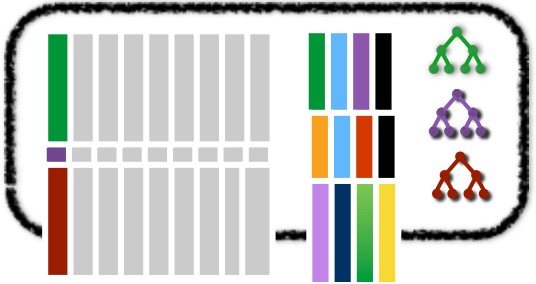
Lots of challenges

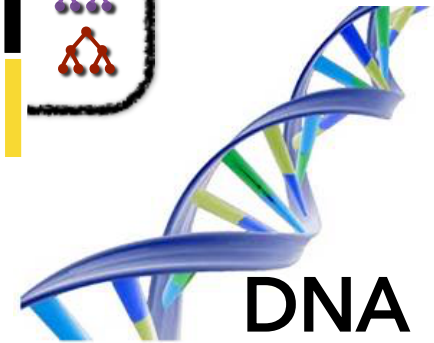
Heterogeneous memories



CPU constraints

memory management





DNA