


UNIVERSITEIT GENT

Space-Efficient 64-bit Java Objects through Selective Typed Virtual Addressing

Kris Venstermans, Lieven Eeckhout, Koen De Bosschere
Department of Electronics and Information Systems
Ghent University - Belgium

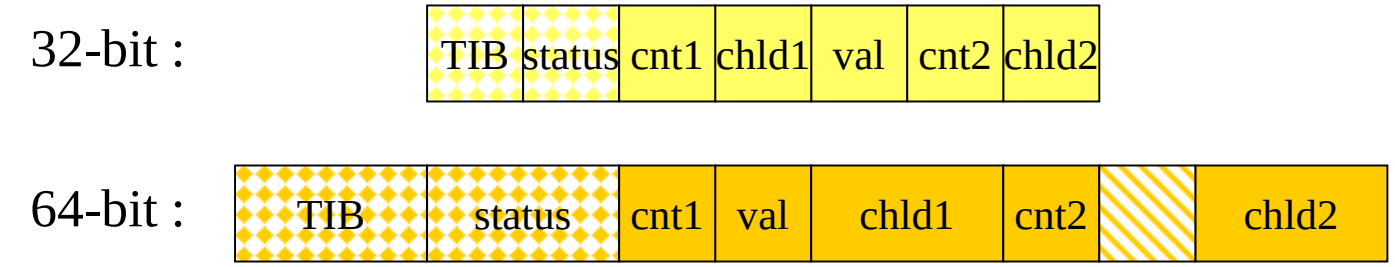
HiPEAC COMPILATION ARCHITECTURE

4th International Symposium on Code Generation and Optimization
March 26-29th , 2006. New York, NY

How big are Java objects ?

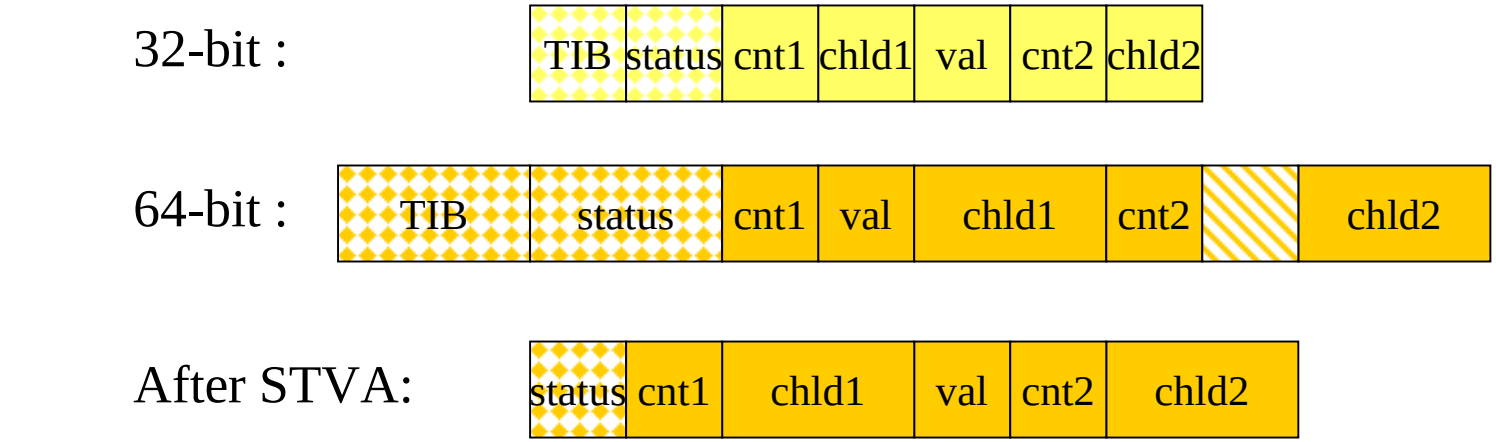
```
class Example {
    int counter1;
    Example child1;
    byte val;
    int counter2;
    Example child2;
}
```

Objects are bigger then they look: besides member fields also header fields

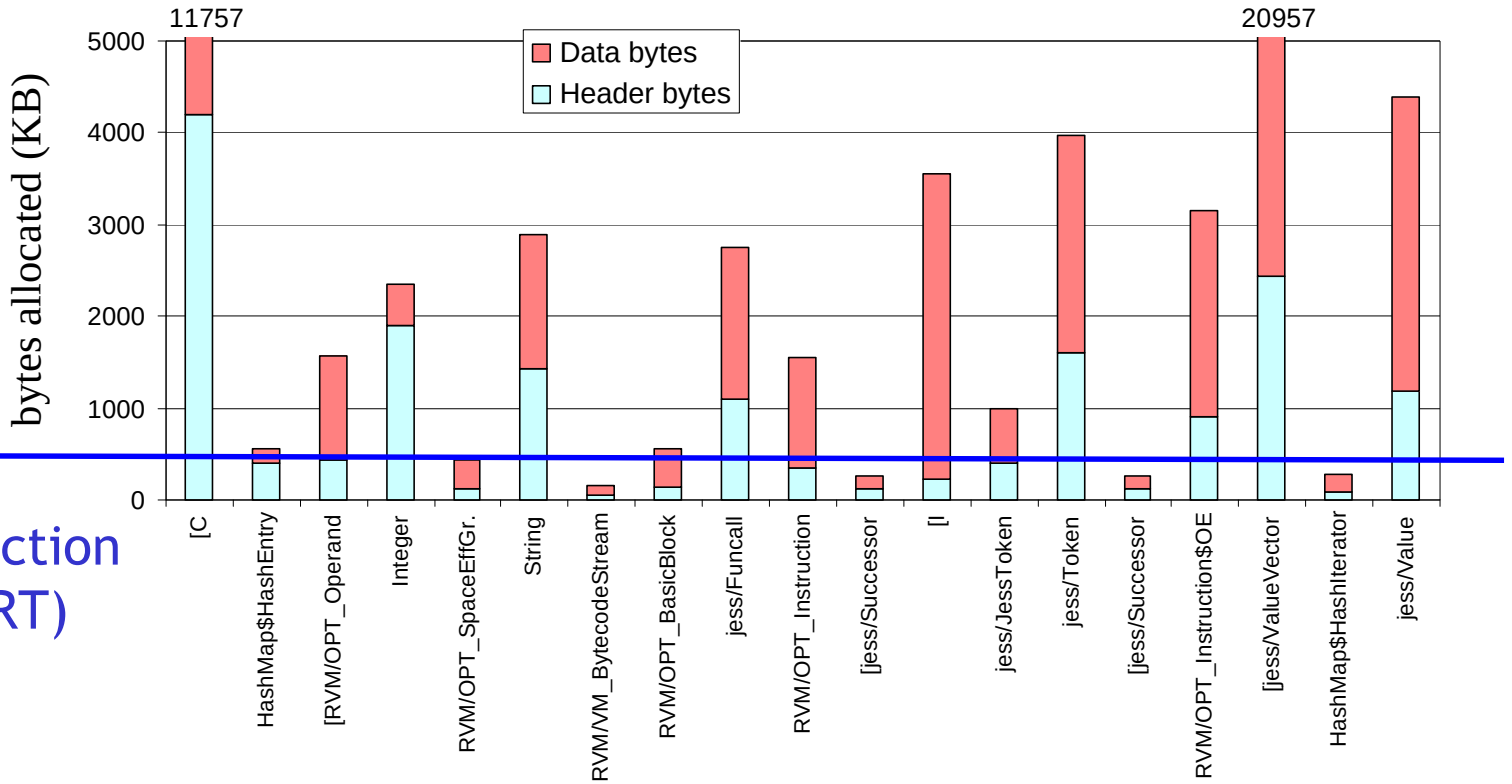


… but we can reduce the header fields

```
class Example {
    int counter1;
    Example child1;
    byte val;
    int counter2;
    Example child2;
}
```

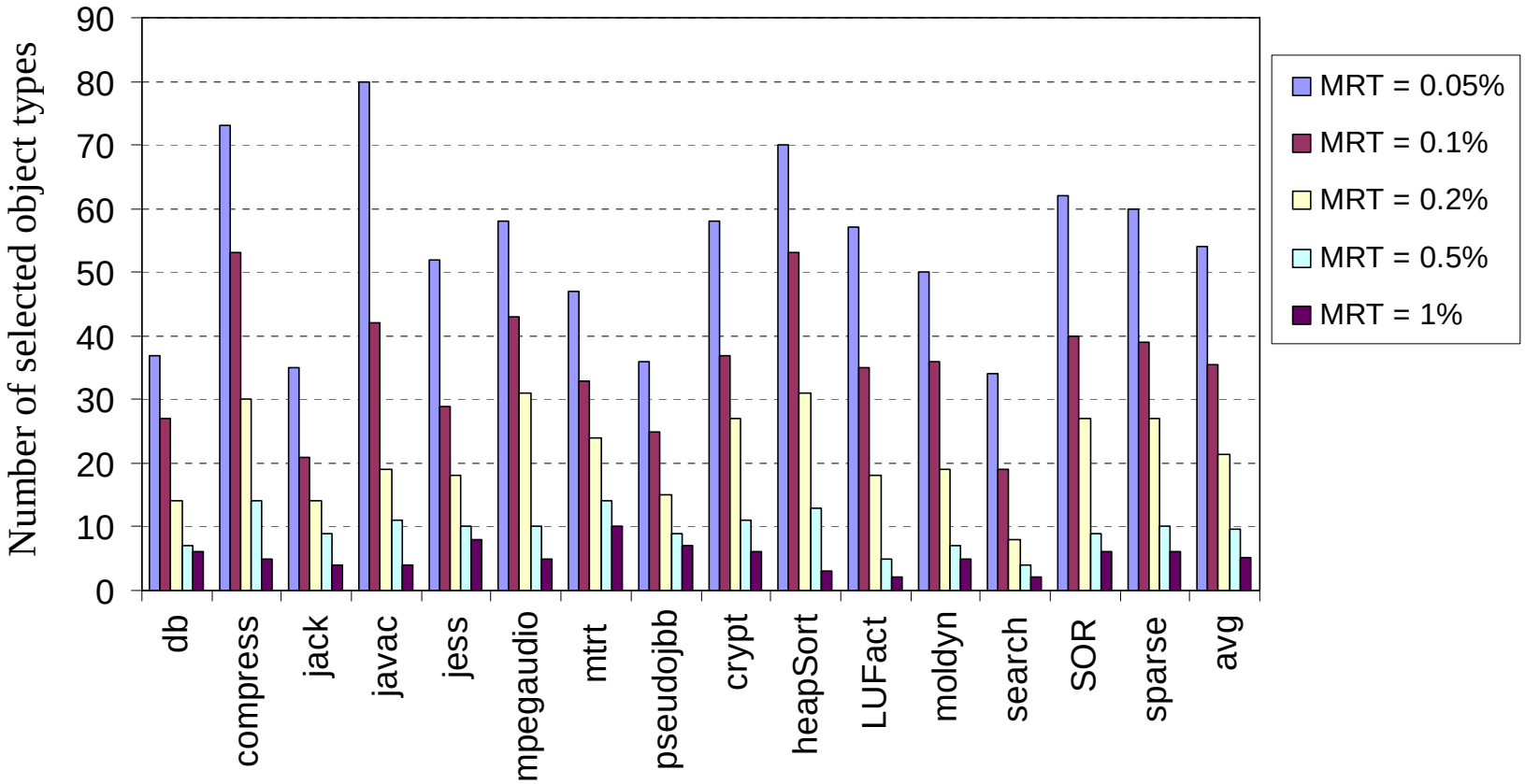


Not all headers are equally important

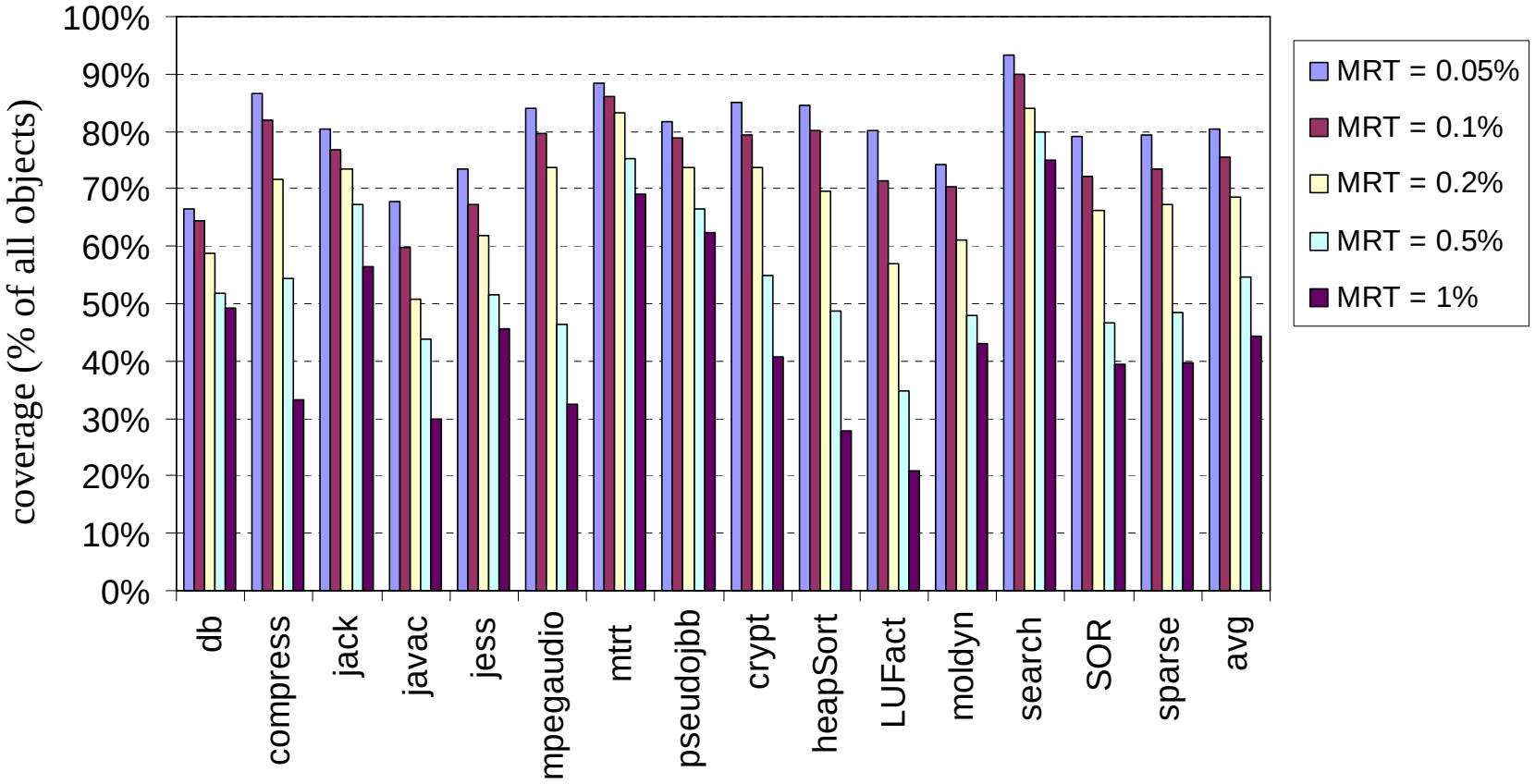


Memory reduction threshold (MRT)

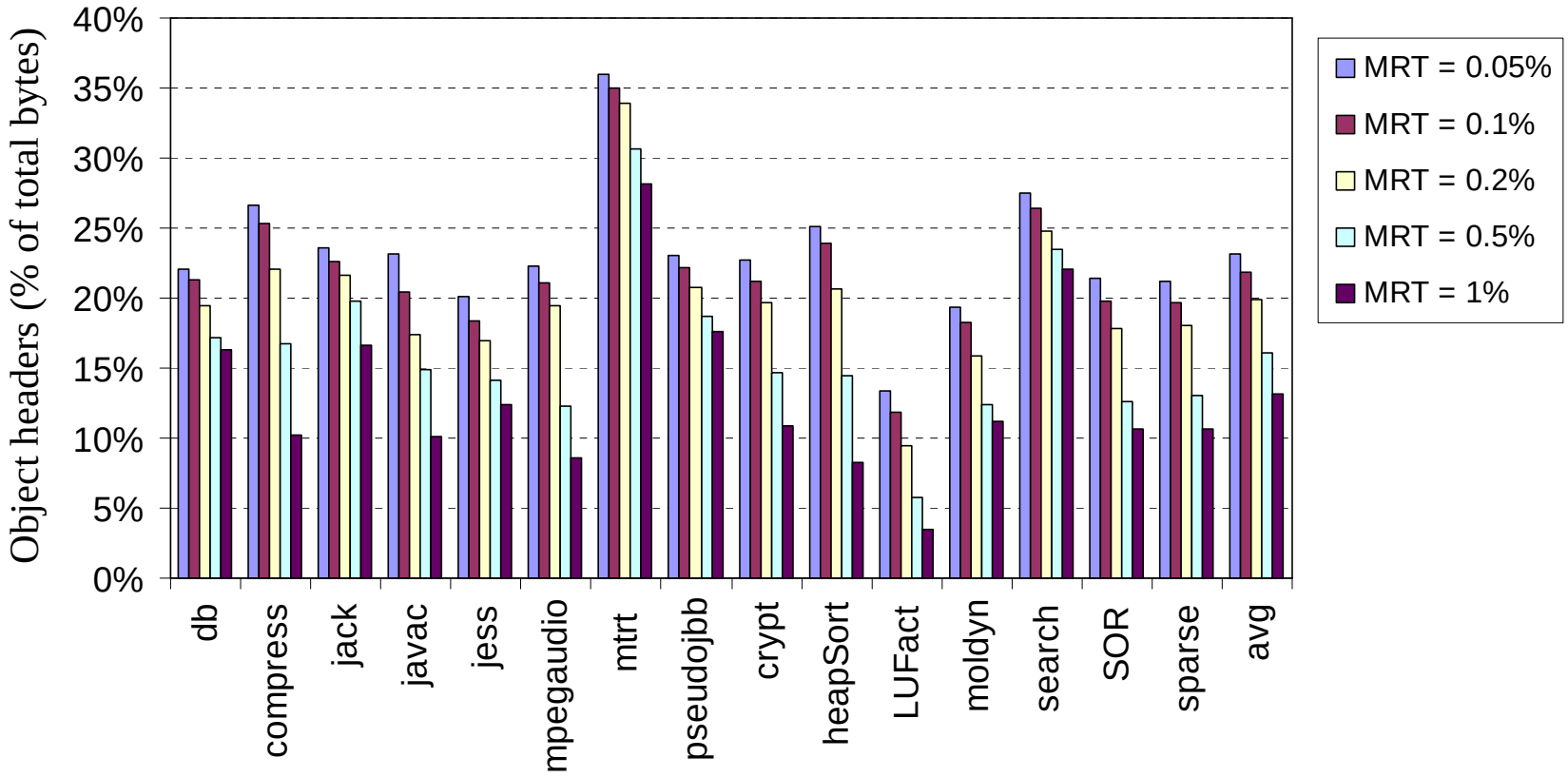
Only a small amount of object types gets selected by our criterium



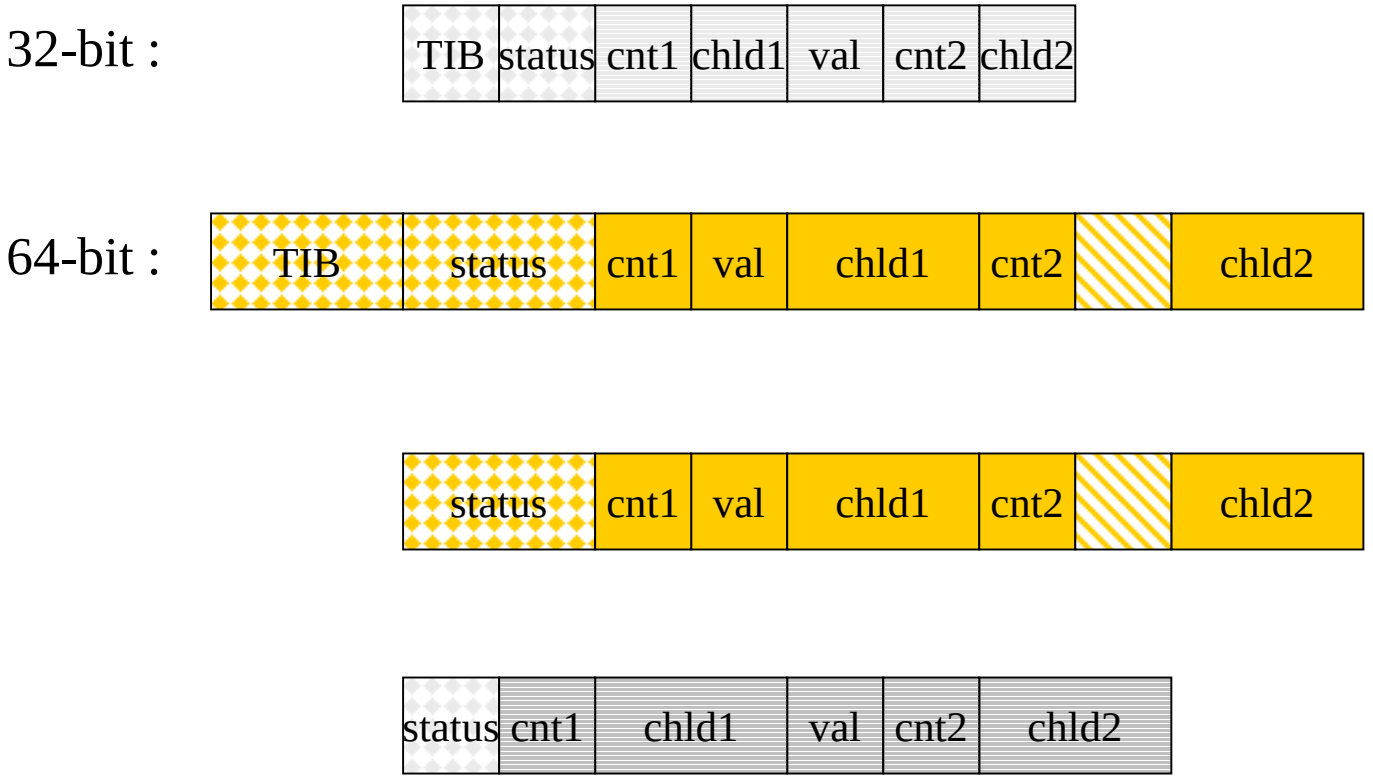
So those small amount of types cover most objects



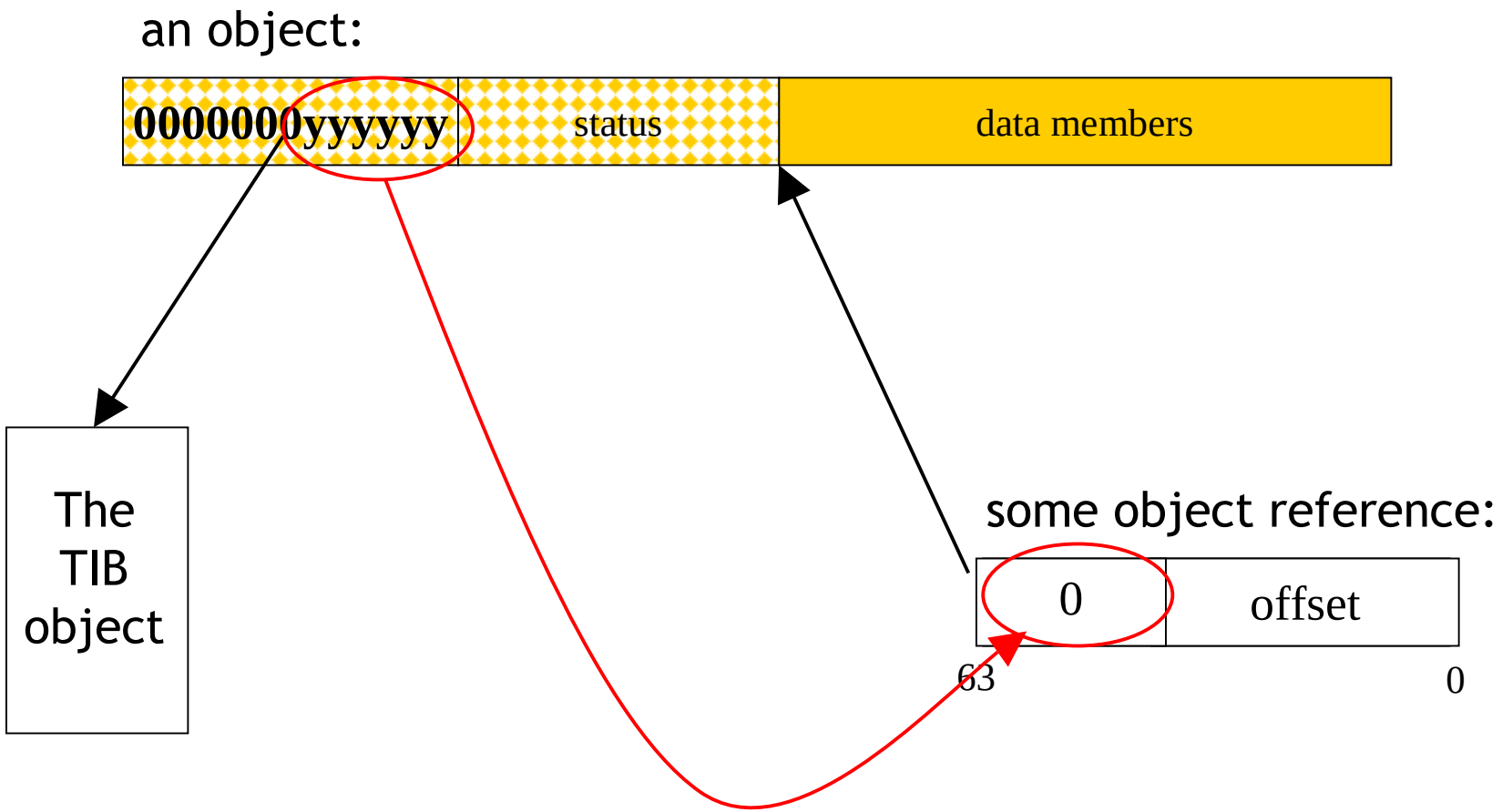
Together the headers of these few types occupy more than 15% of all space



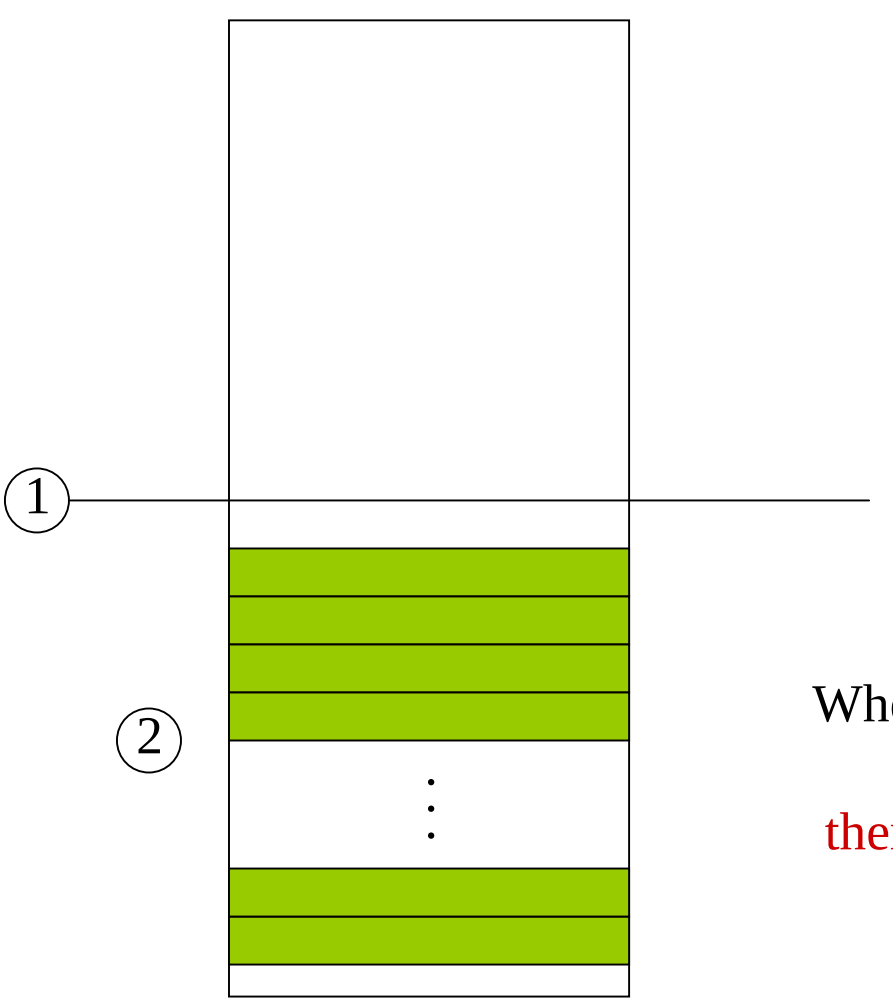
Step 1: Removing TIB through TVA: Typed Virtual Addressing

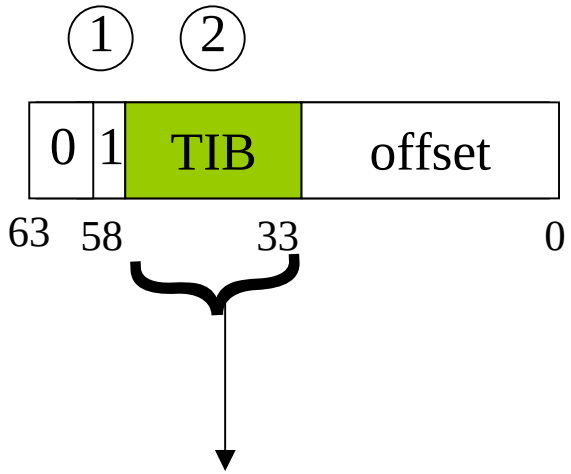


Basic principle TVA



Selective Typed Virtual Addressing



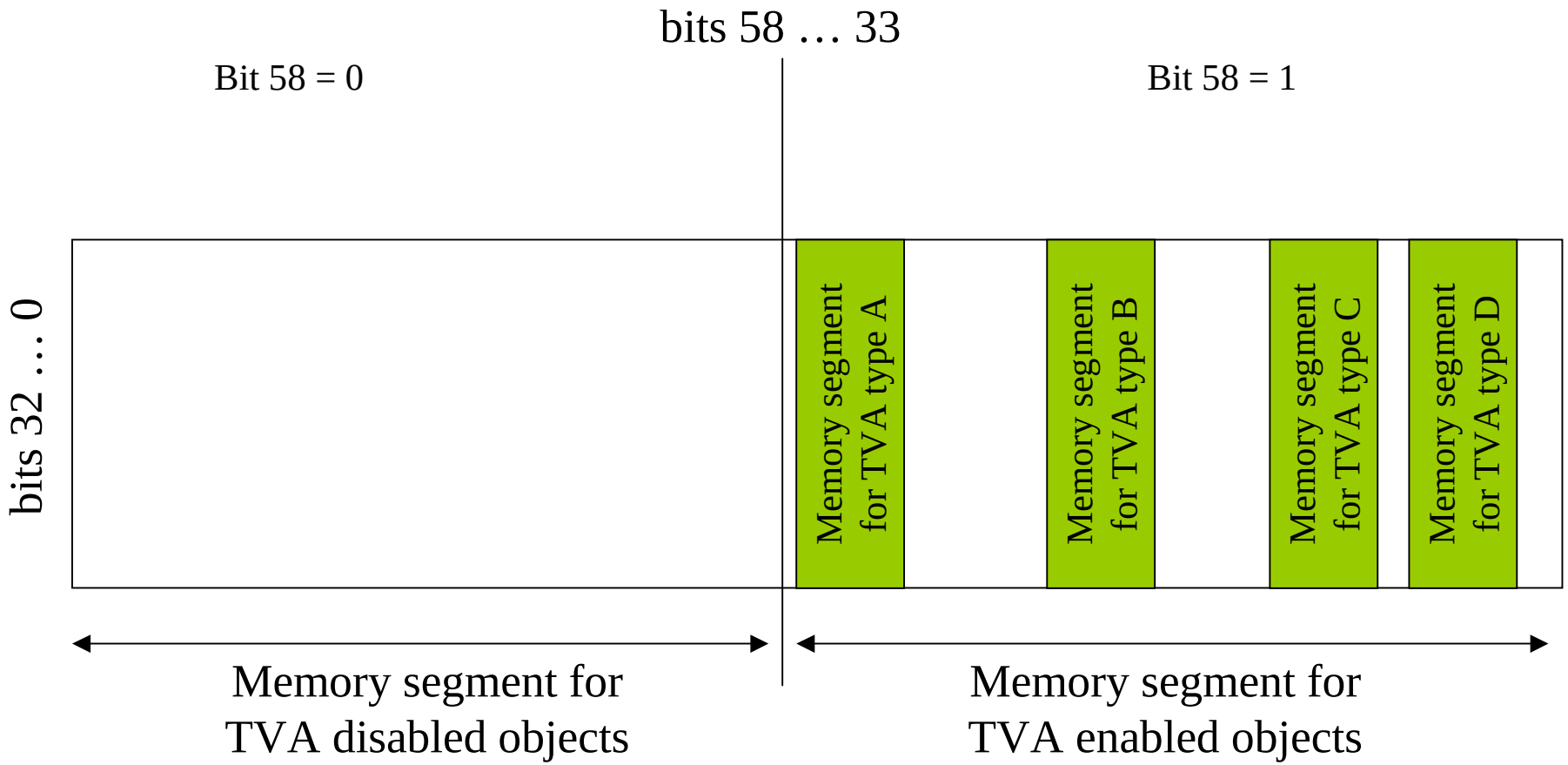


25 bits + 3 bits alignation = 28 bits

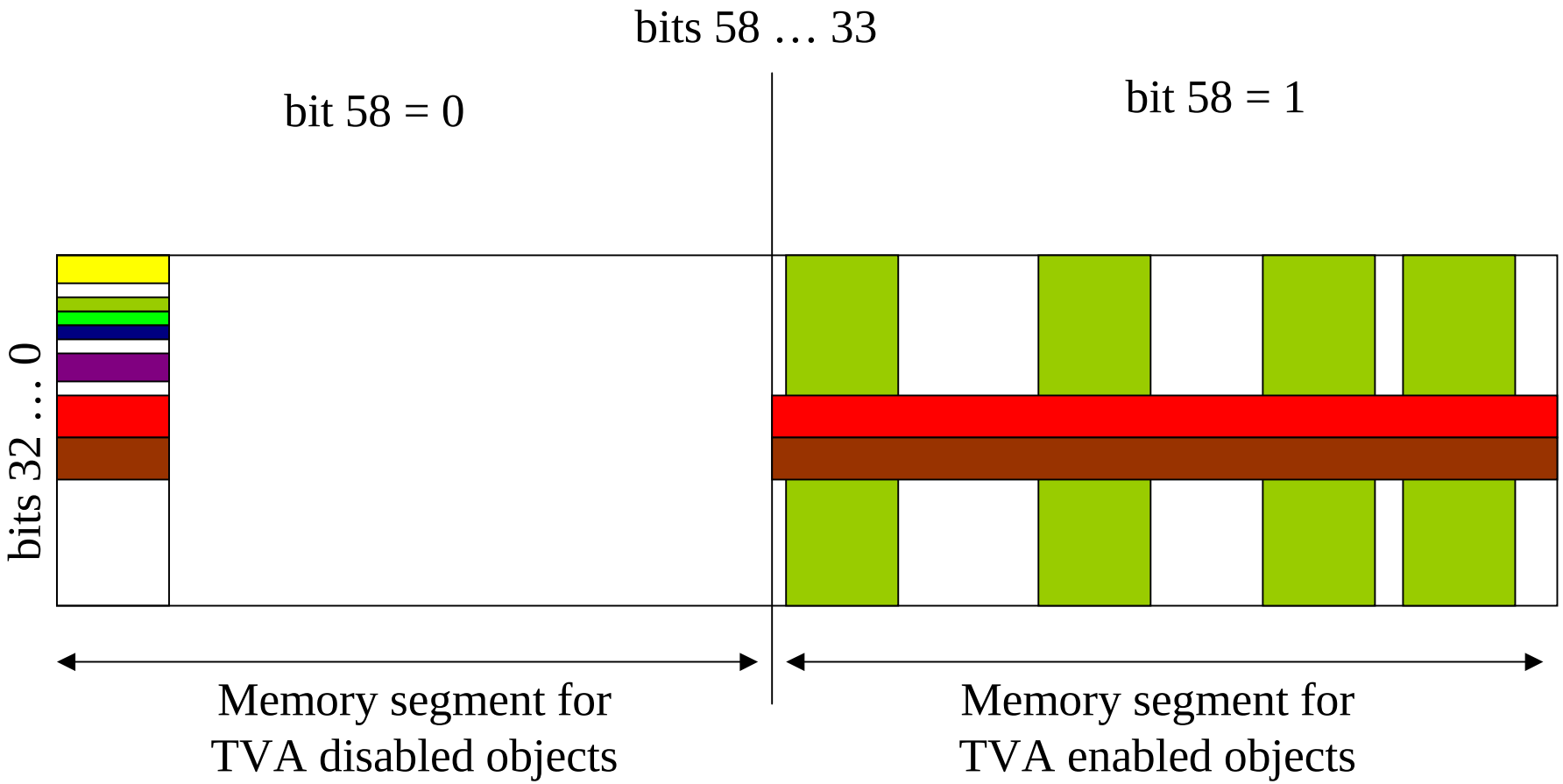
or 256 MB adressing space for TIBs

When inspecting a candidate type for STVA:
first allocate the TIB,
then select the corresponding TVA memory segment

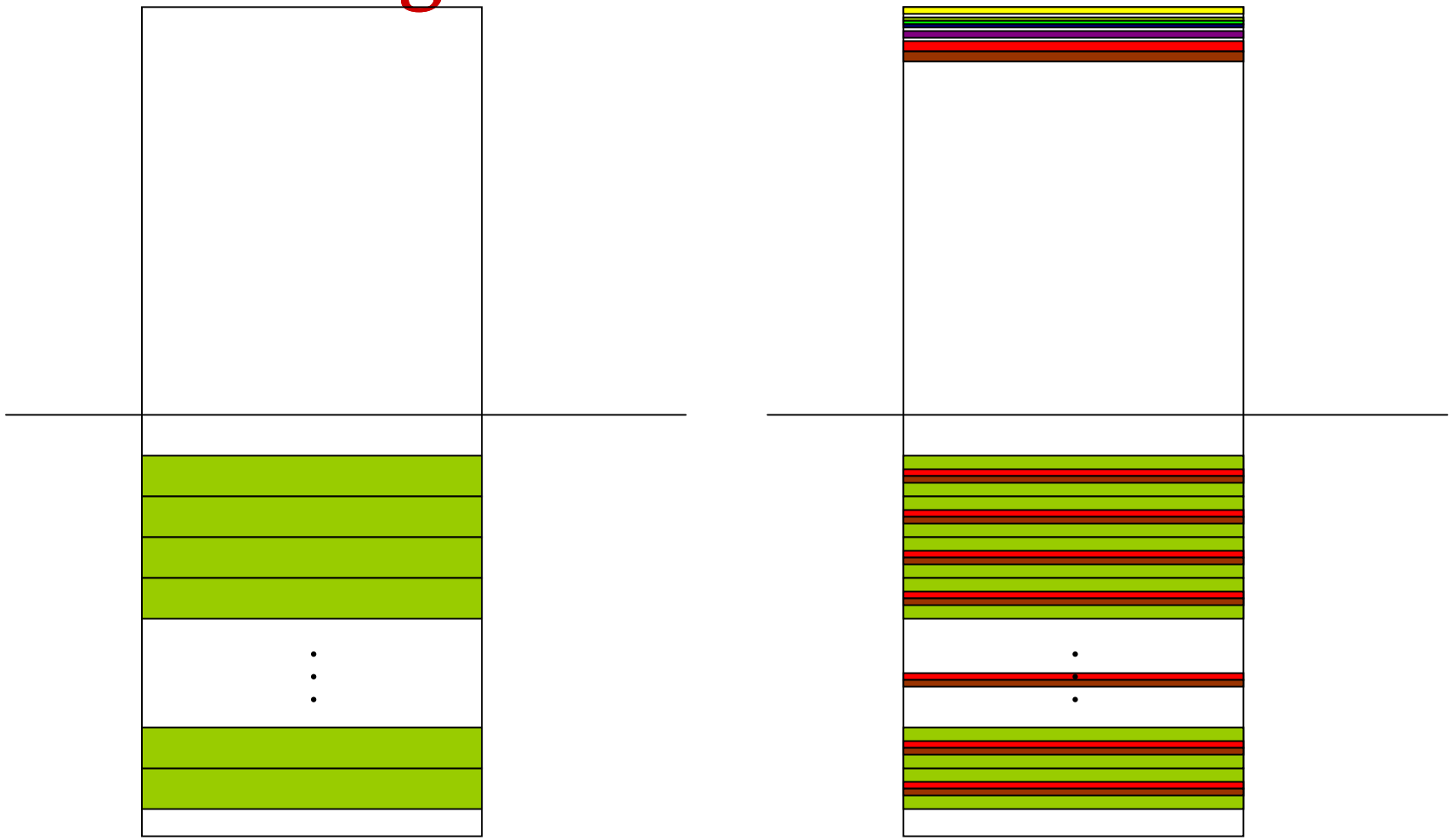
STVA's virtual address space



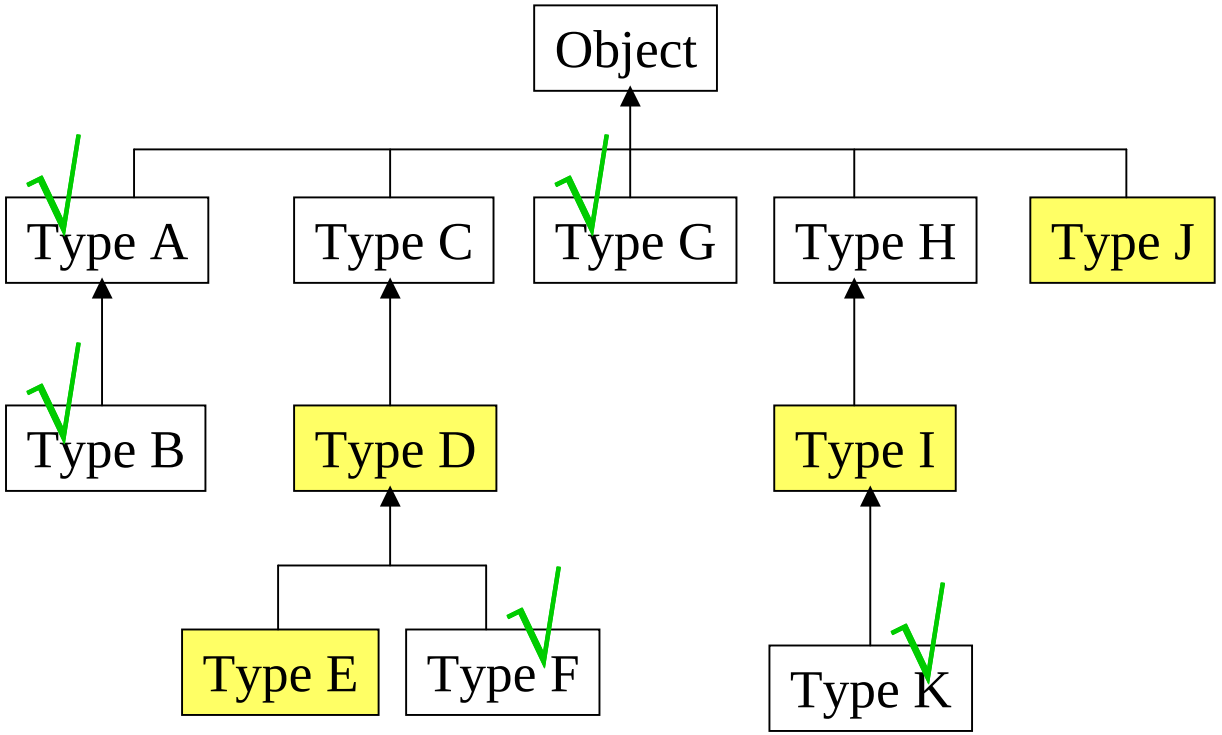
… and the VM spaces, where are they?



STVA-enabled spaces are non-contiguous

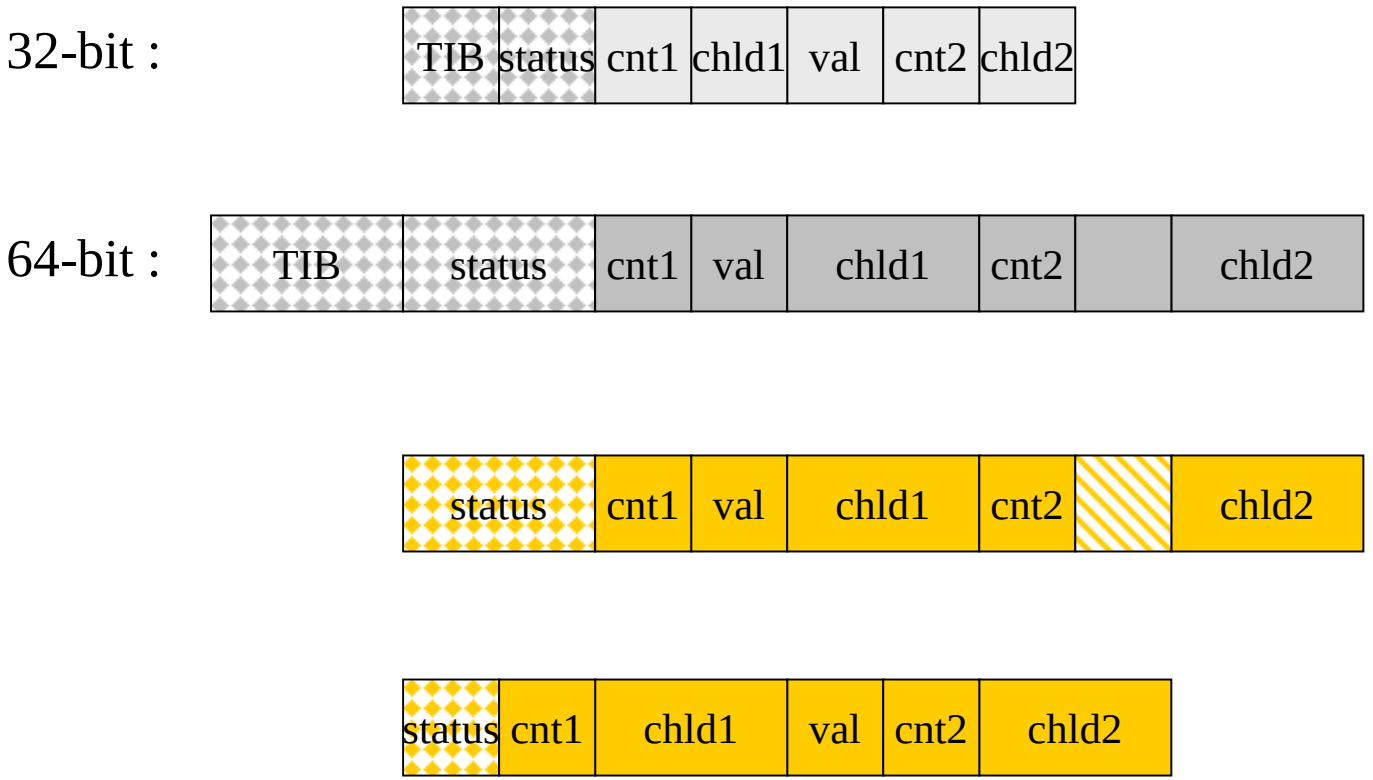


TIB access requires runtime check of STVA-bit

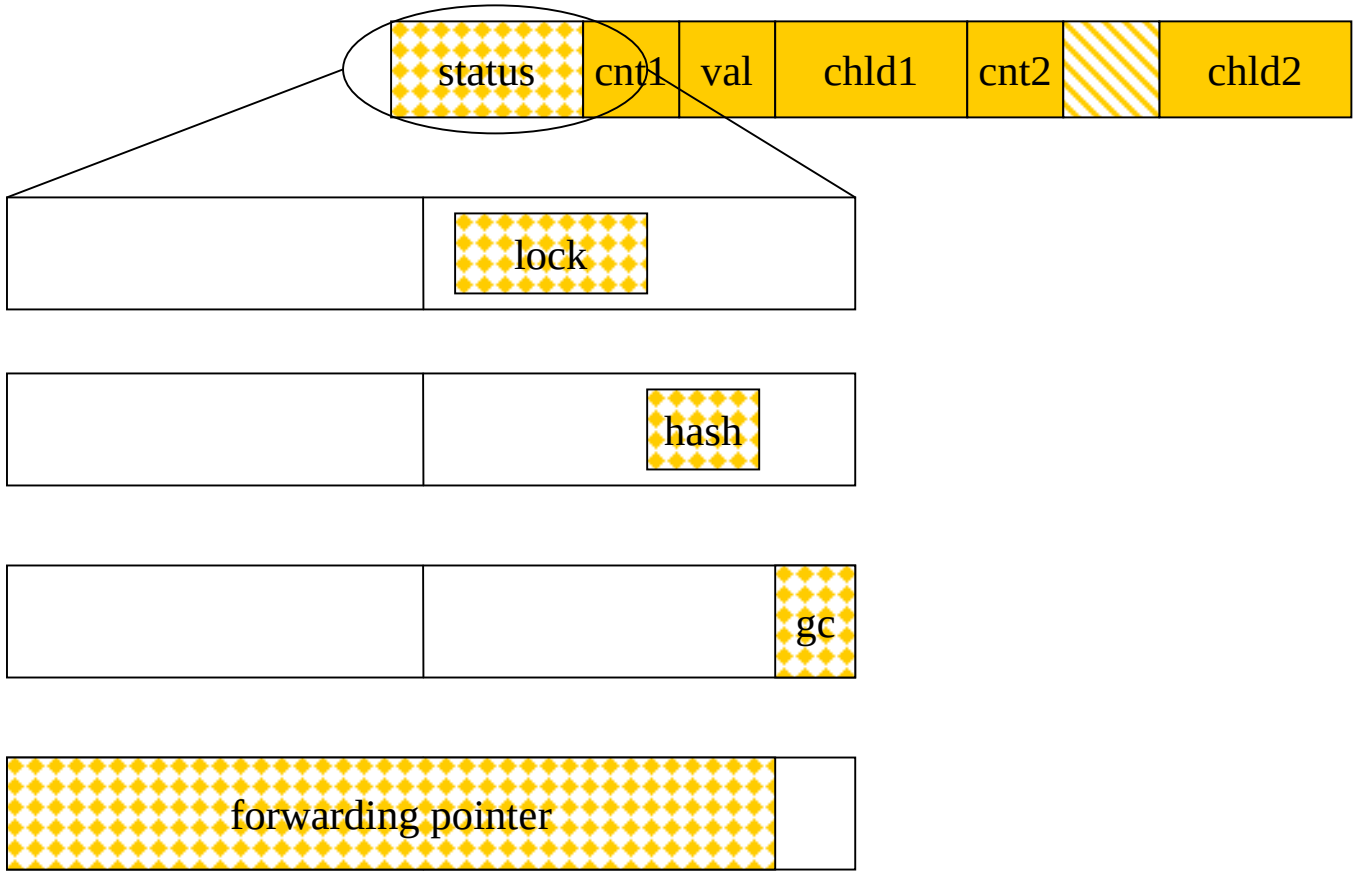


√ : except if no subtype is selected for STVA

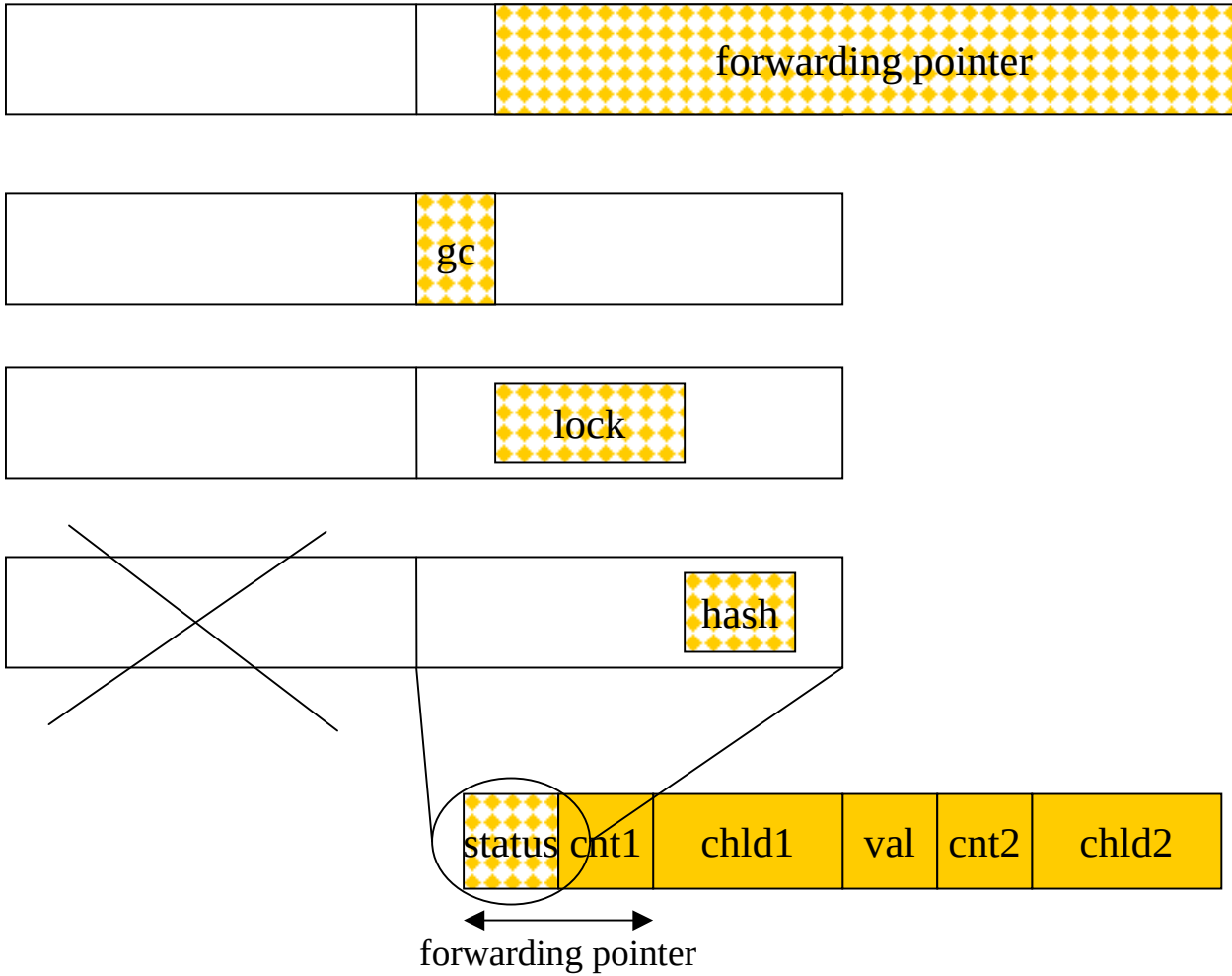
Step 2: Shrinking the status field



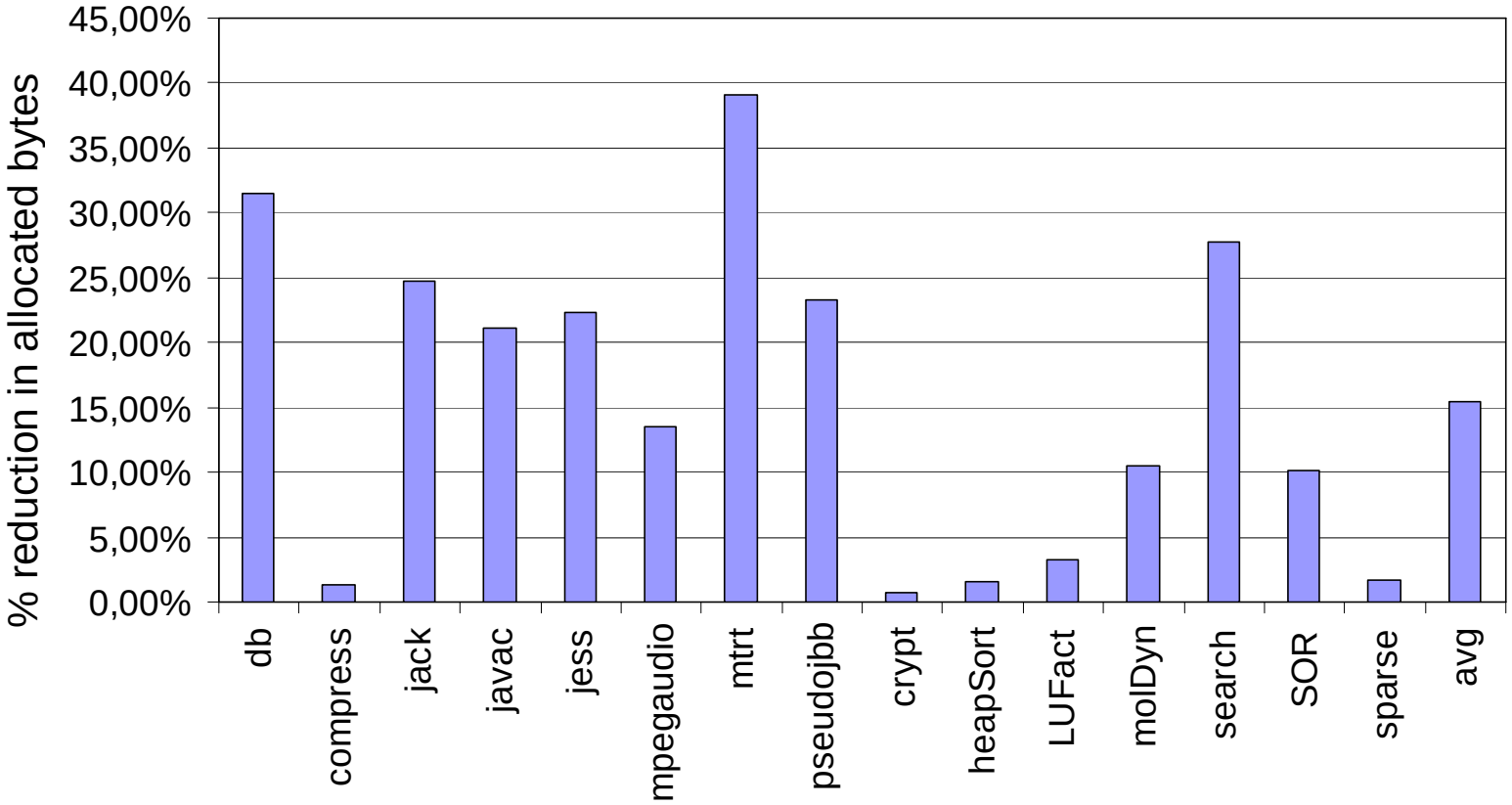
What's in a status field ?



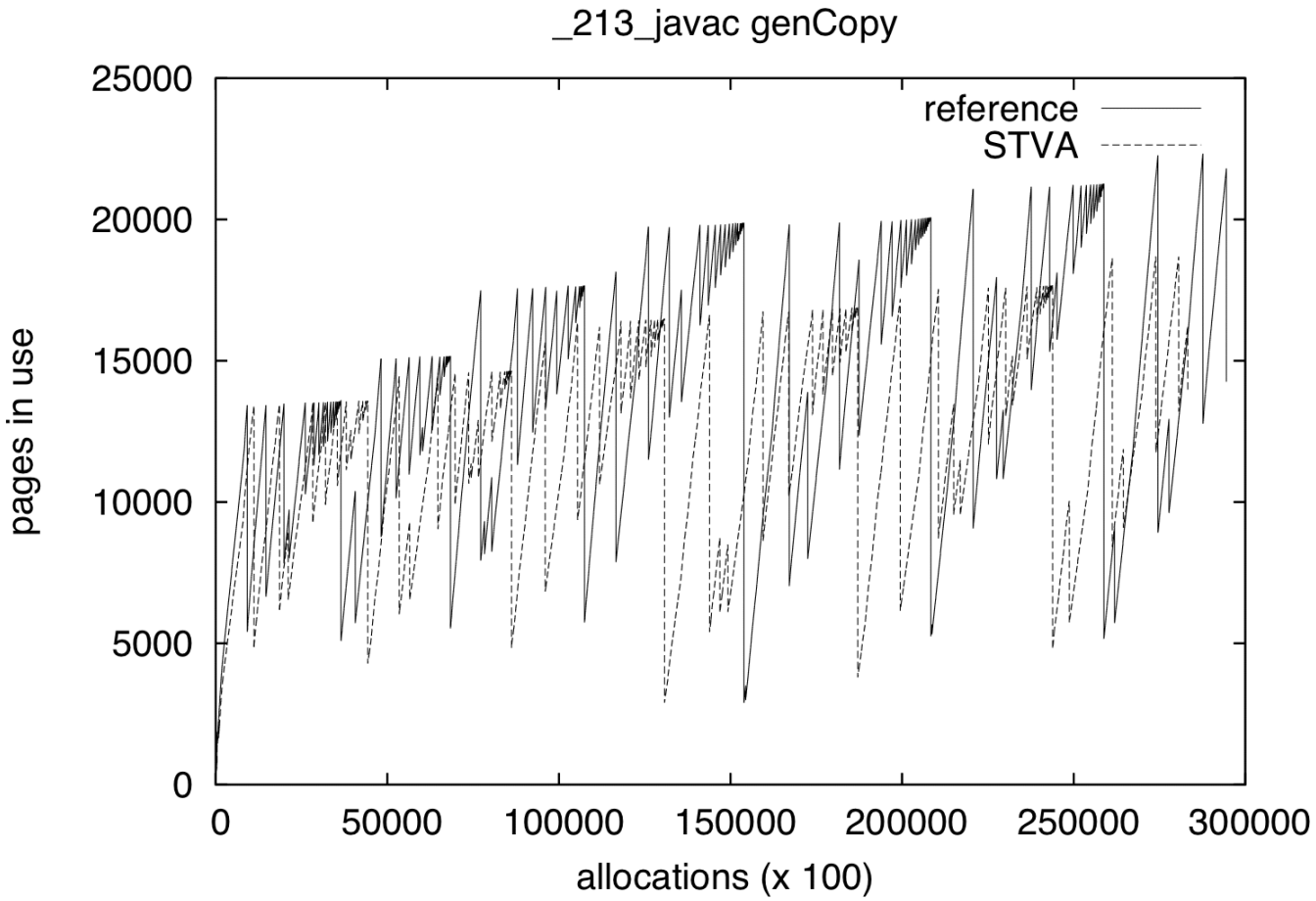
Relocating the forwarding pointer



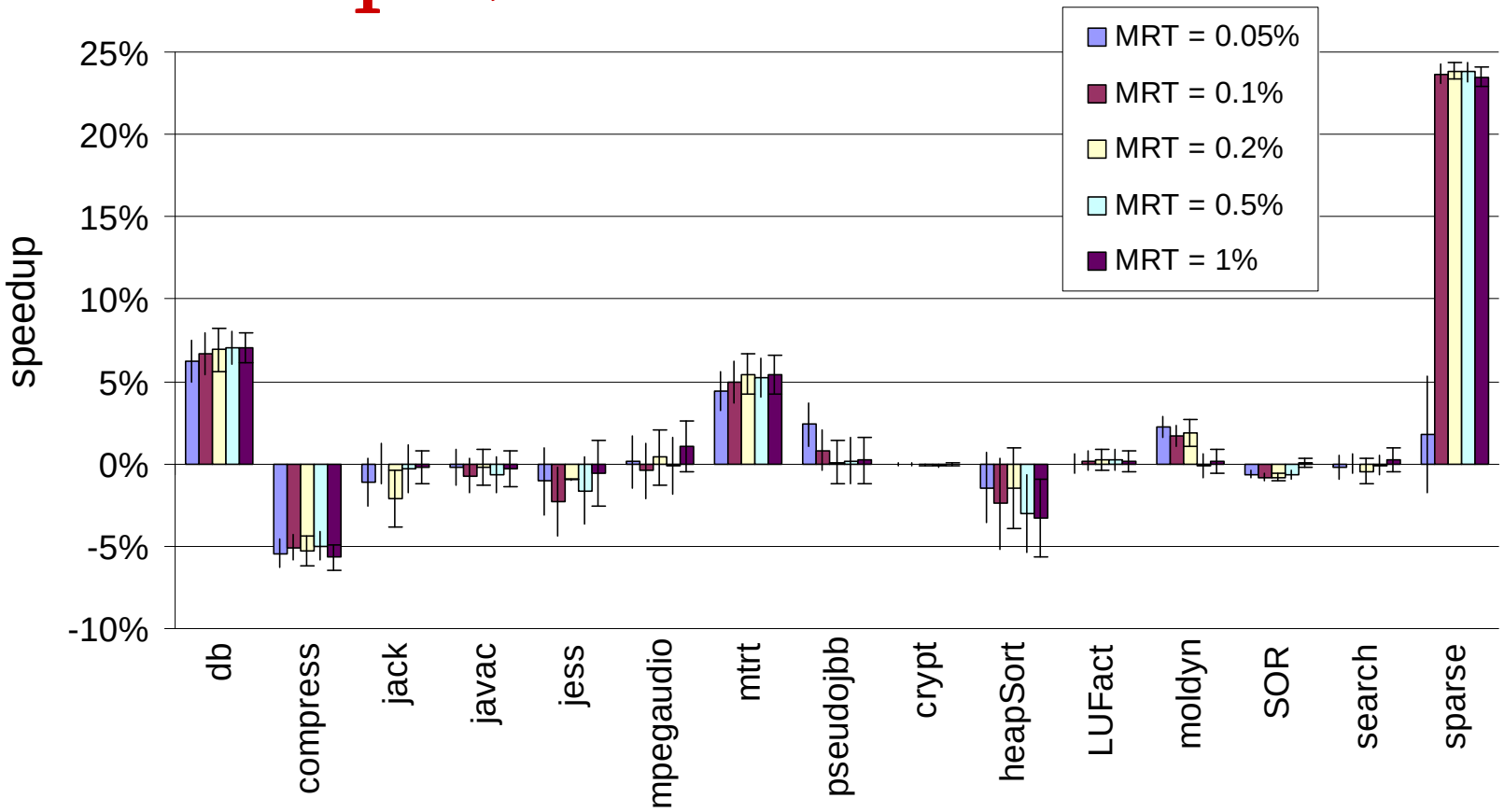
We reduce allocation on average with 15%



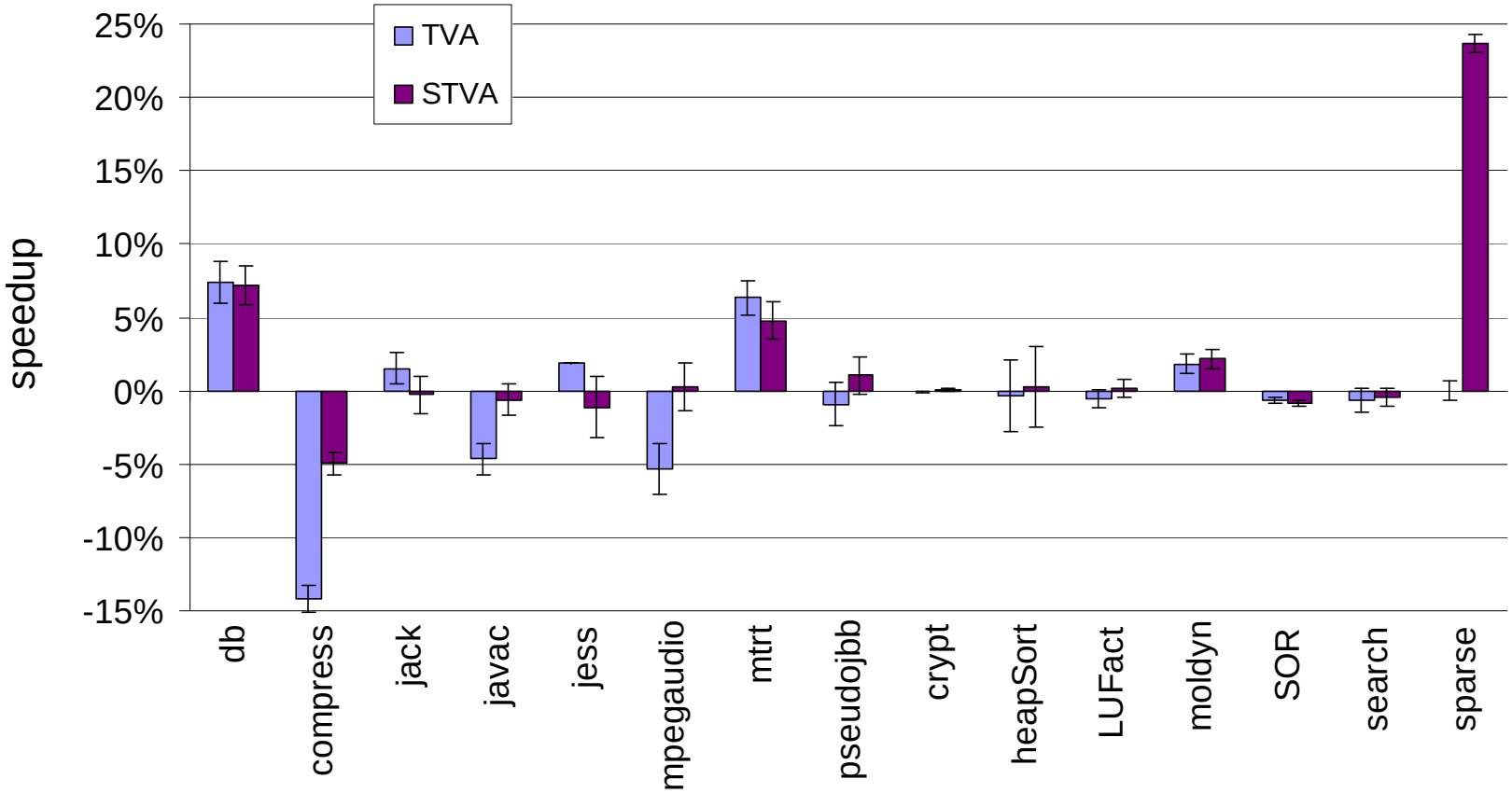
Beter heap occupancy, less GC



Performance: status quo, sometimes much faster



Performance: TVA vs. STVA



Related work

- 64-bit Pointer compression techniques:
  - **Adl-Tabatabai et al**. Improving 64-bit Java IPF performance by compressing heap references. CGO 2004.
  - **Lattner and Adve**. Transparent pointer compression for linked data structures. MSP 2005.

- Java header compression on 32-bit machines
  - **Bacon et al.** Space- and time-efficient implementation of the Java object model. ECOOP 2002.
  - **Shuf et al.** Exploiting prolific types for memory management and optimizations. POPL 2002.

- coallocating objects of the same type
  - **Dybvig et al.** Don't Stop the BIBOP: flexible and efficient storage management for dynamically-typed laguages. Tech.Rep. 1994

Conclusion: STVA does …

- give significant reduction in bytes allocated
- on average not have performance loss
- limit fragmentation