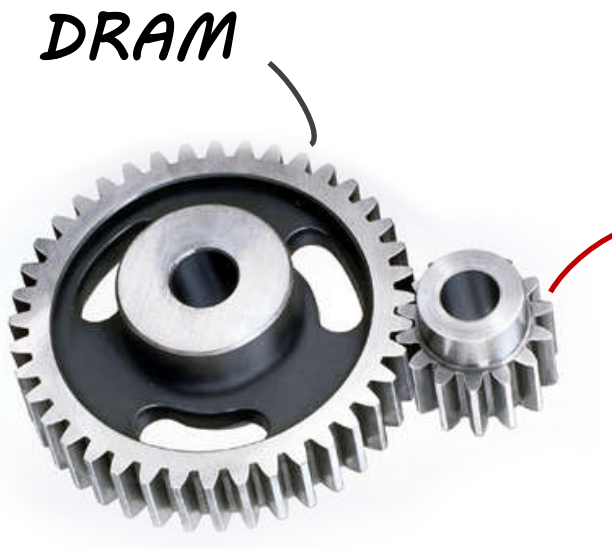
# NightWatch: Integrating Transparent Cache Pollution Control into Dynamic Memory Allocation Systems

Rentong Guo[1], Xiaofei Liao[1], Hai Jin[1],
Jianhui Yue[2], Guang Tan[3]

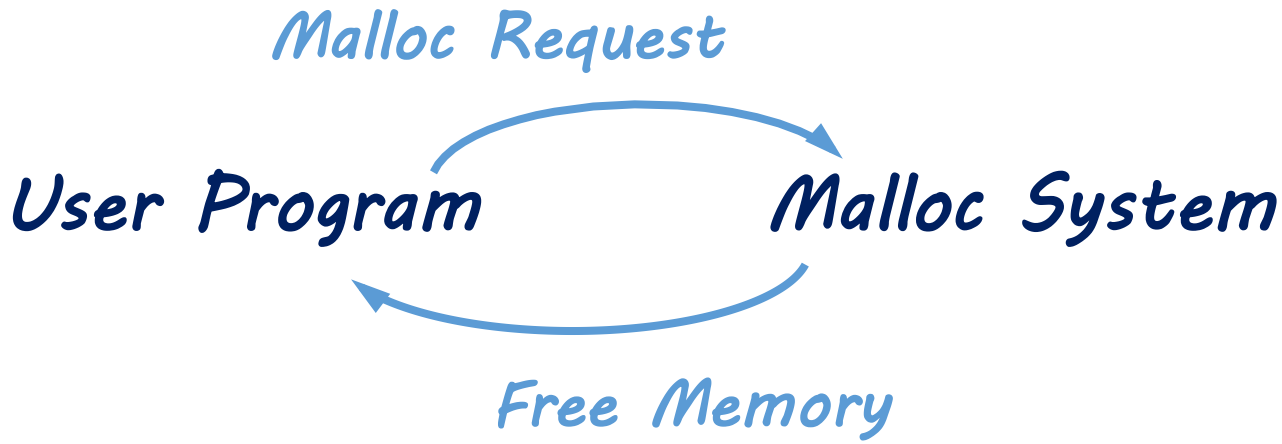
[1]Huazhong University of Science and Technology
[2]Auburn University
[3]SIAT, Chinese Academy of Sciences

# Malloc System

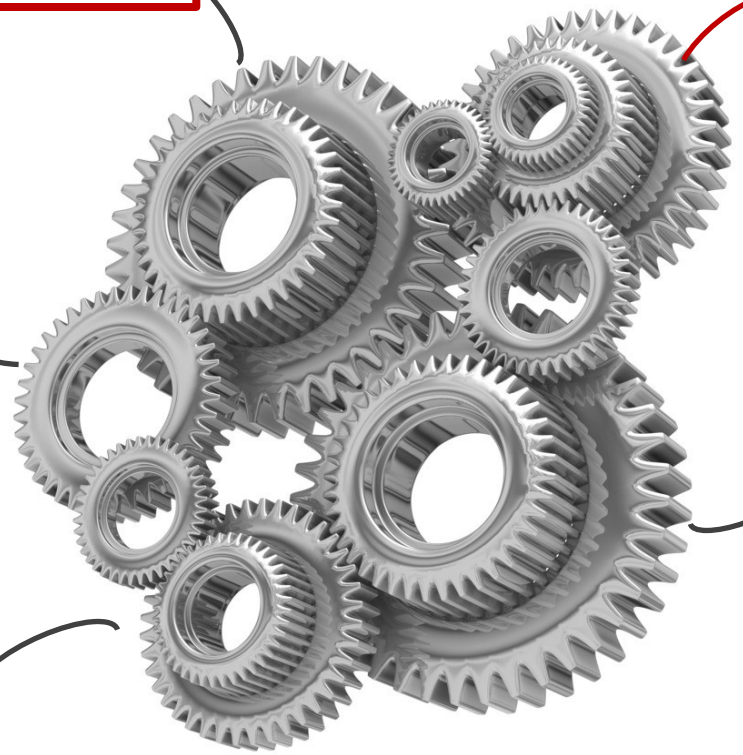


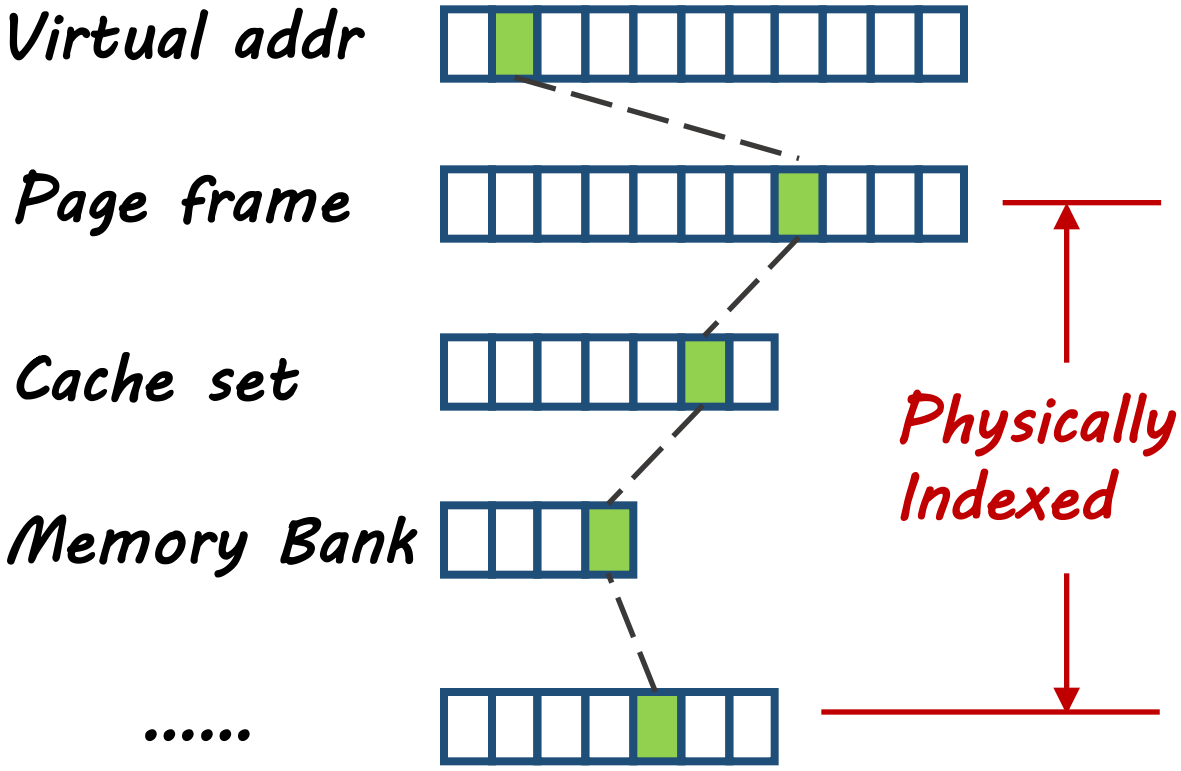

A system managing main memory

```
int* chunk = malloc(size);
```



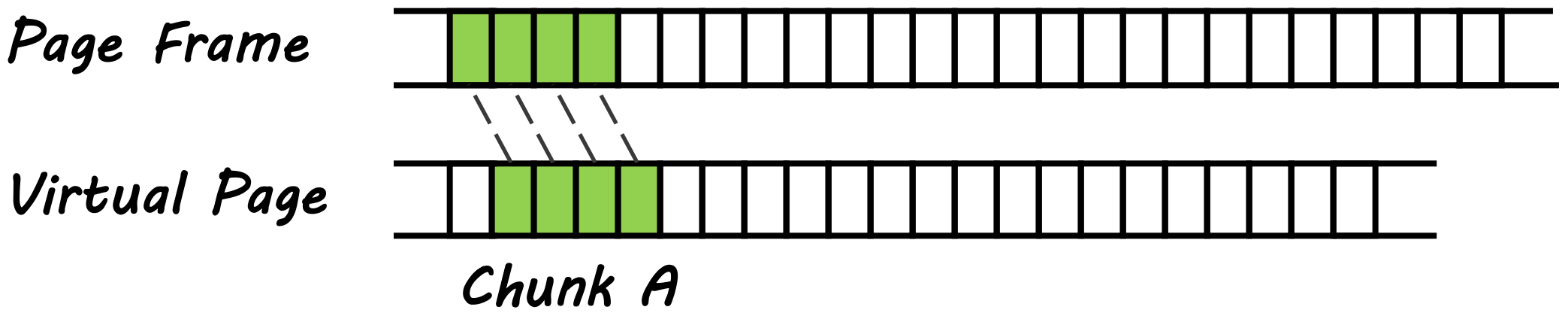
# The Whole Picture



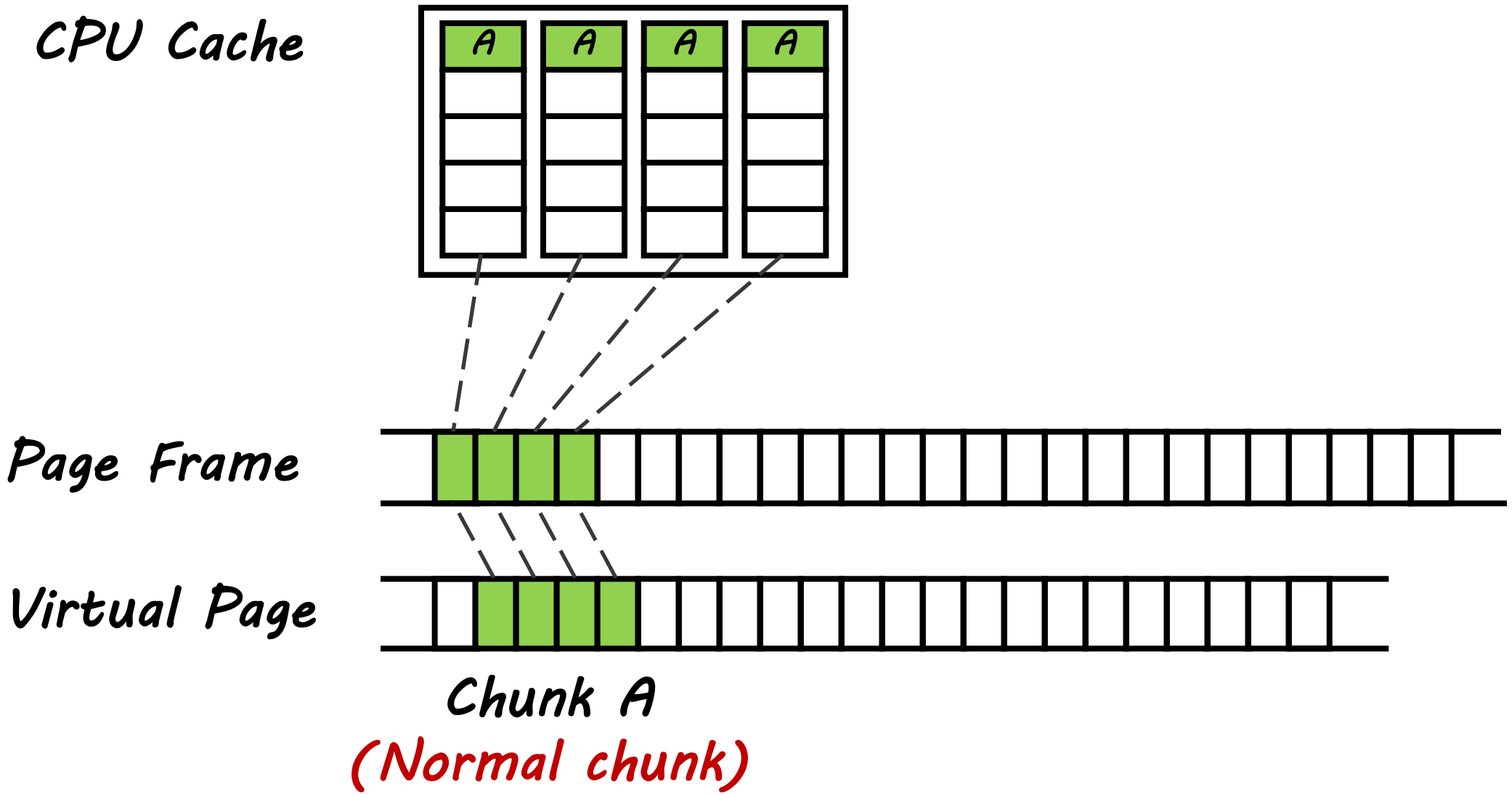




A system allocating resources across multiple hardware layers

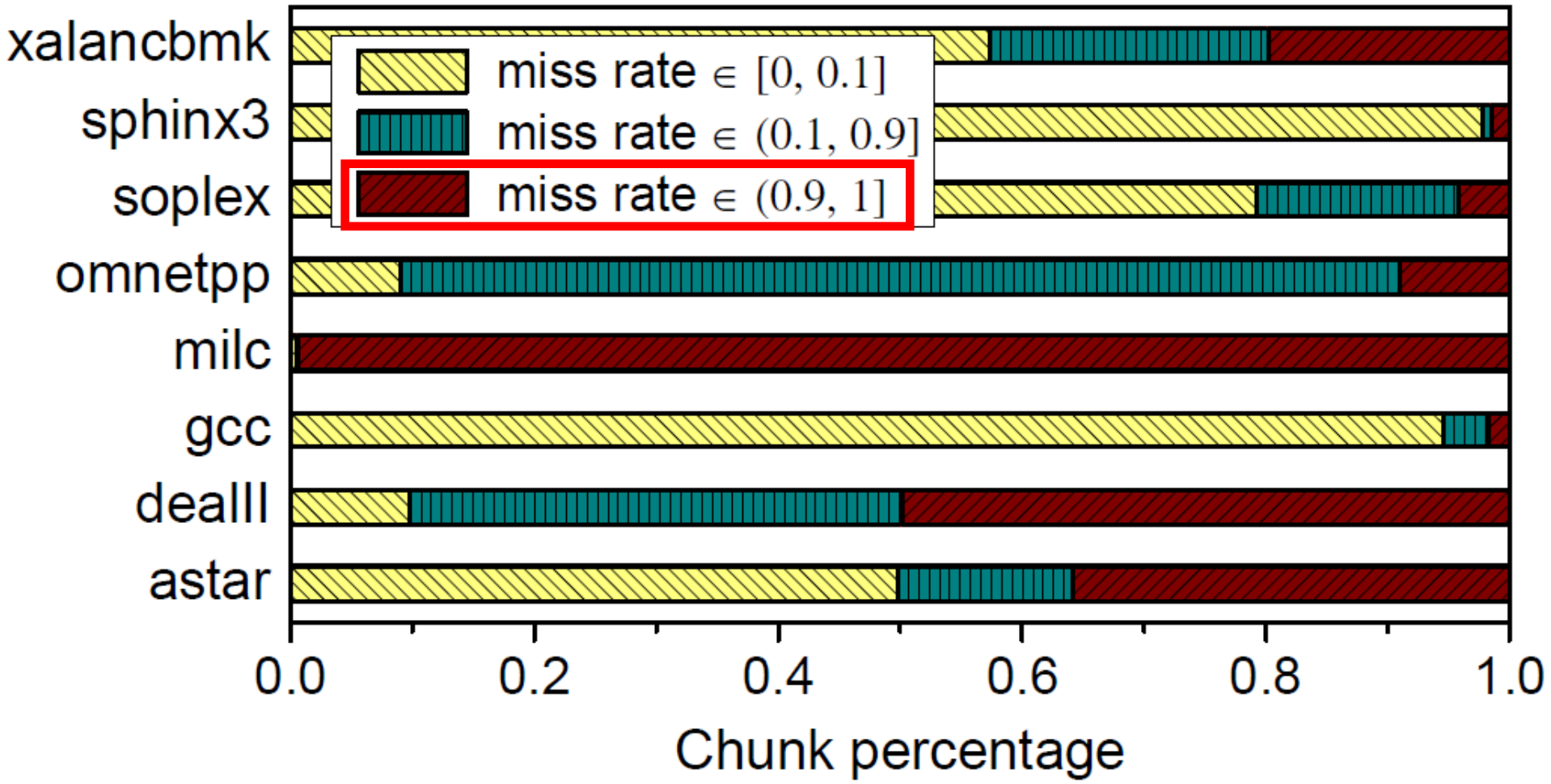
# Cache Resource Allocation



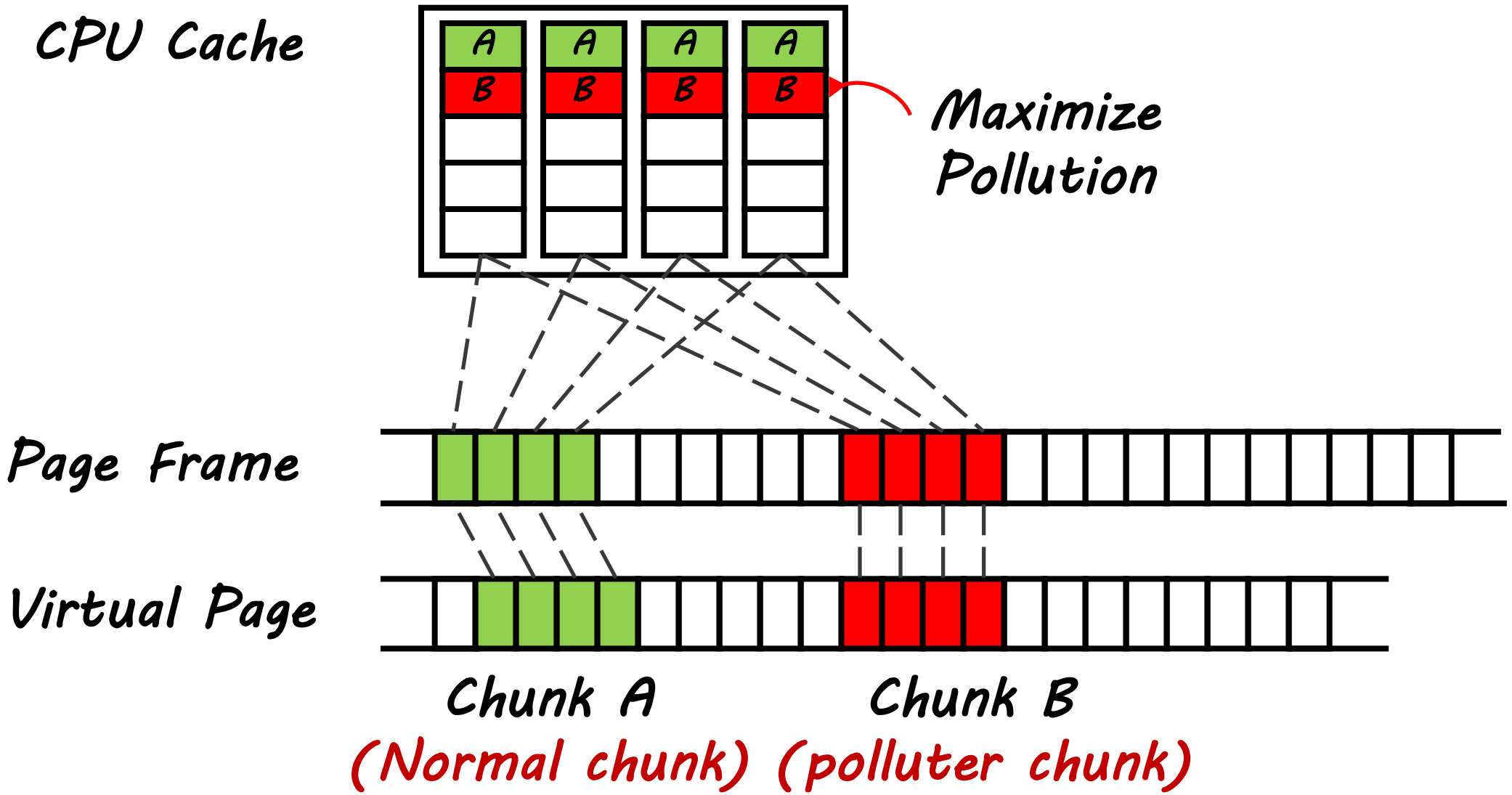
# Cache Resource Allocation



# Data Chunks Have Different Access Locality Pattern

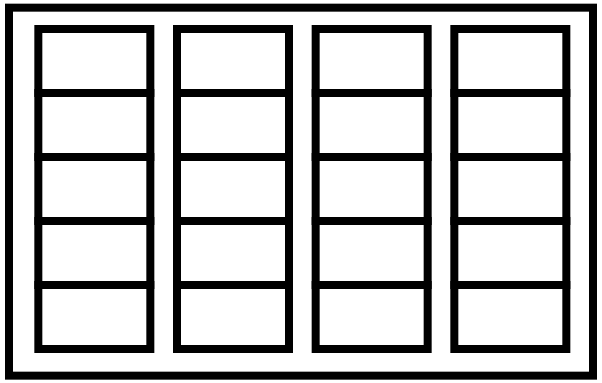


# Cache Resource Allocation



# Cache Resource Allocation

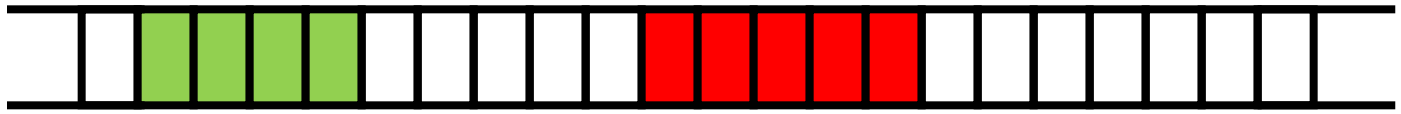
CPU Cache



Page Frame



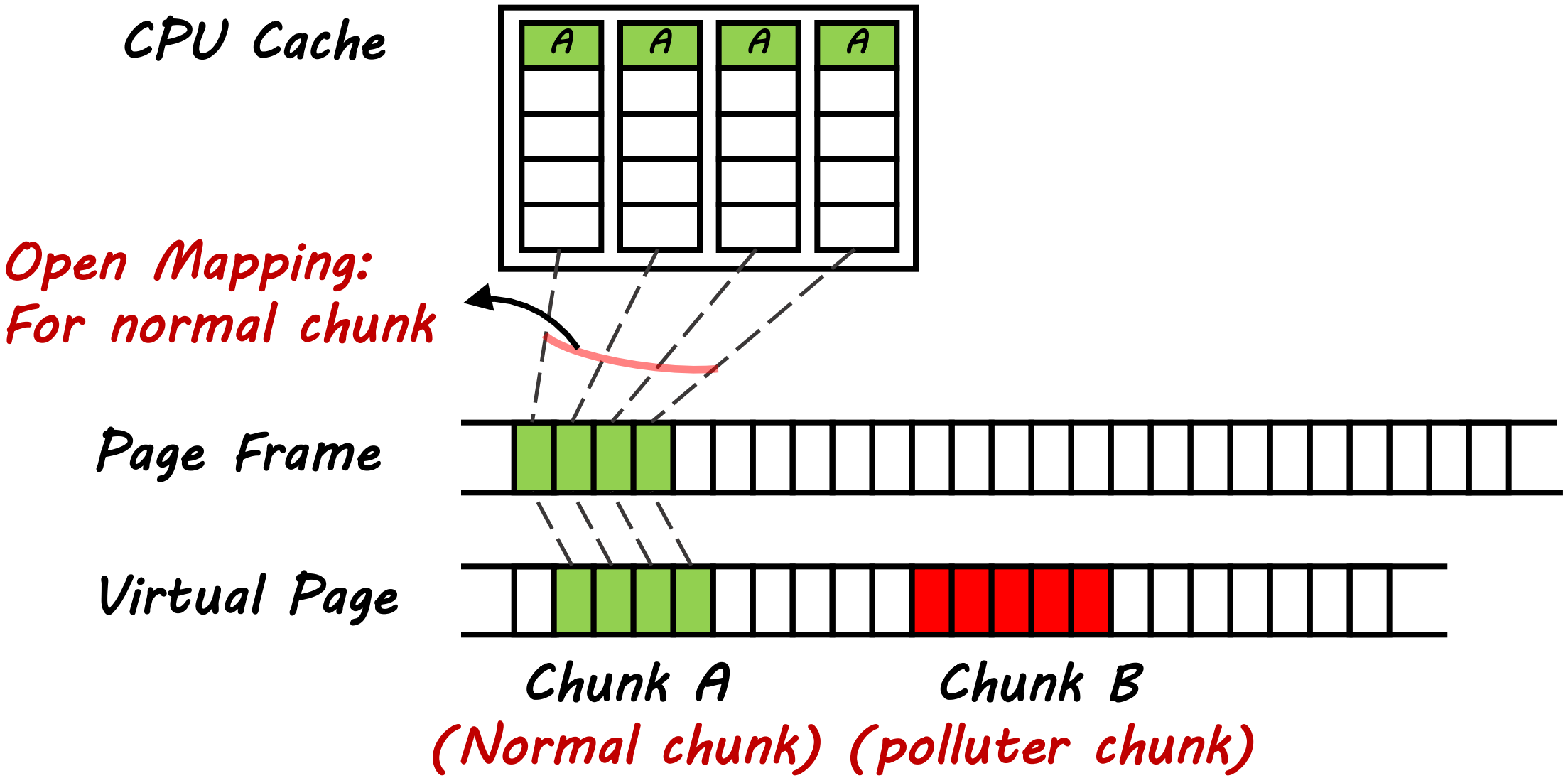
Virtual Page



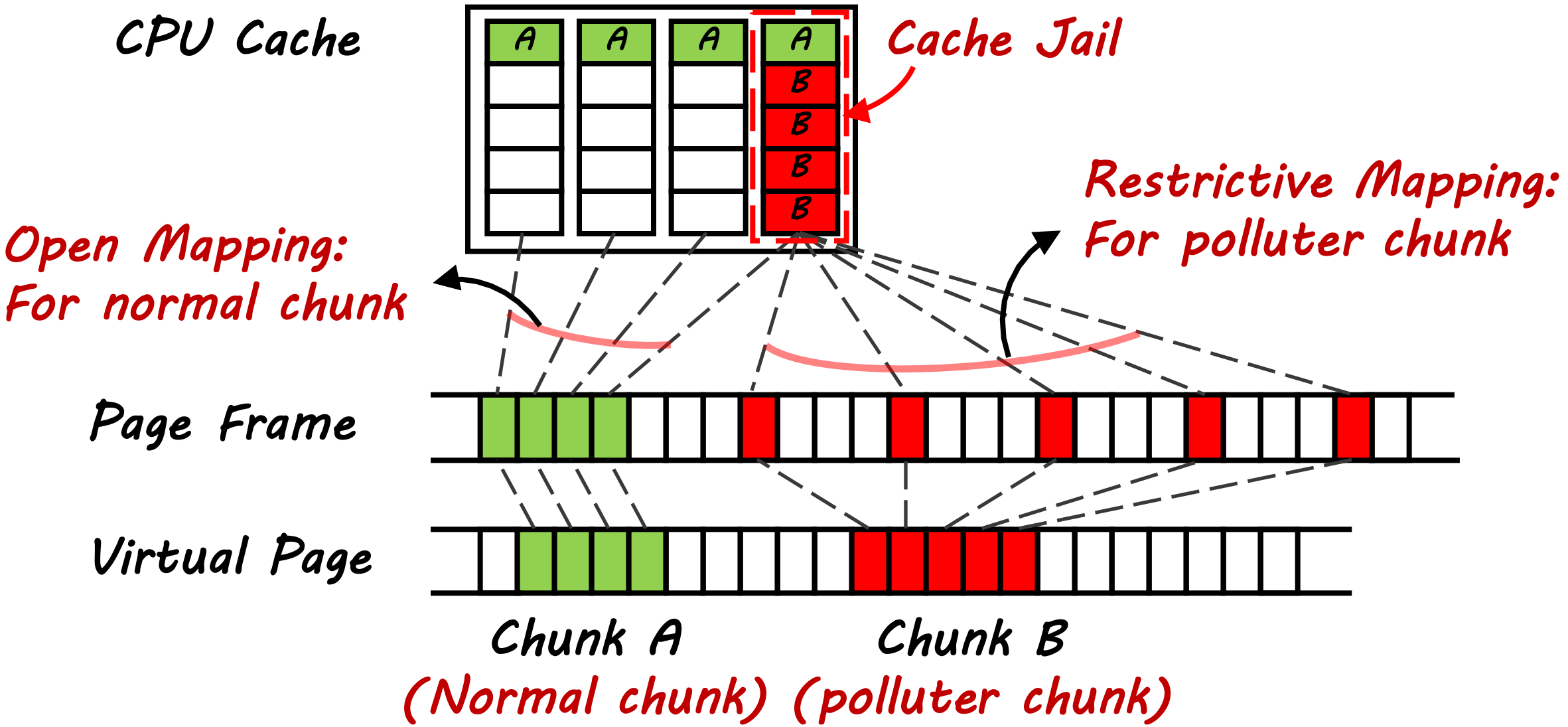
Chunk A (Normal chunk)

Chunk B (polluter chunk)

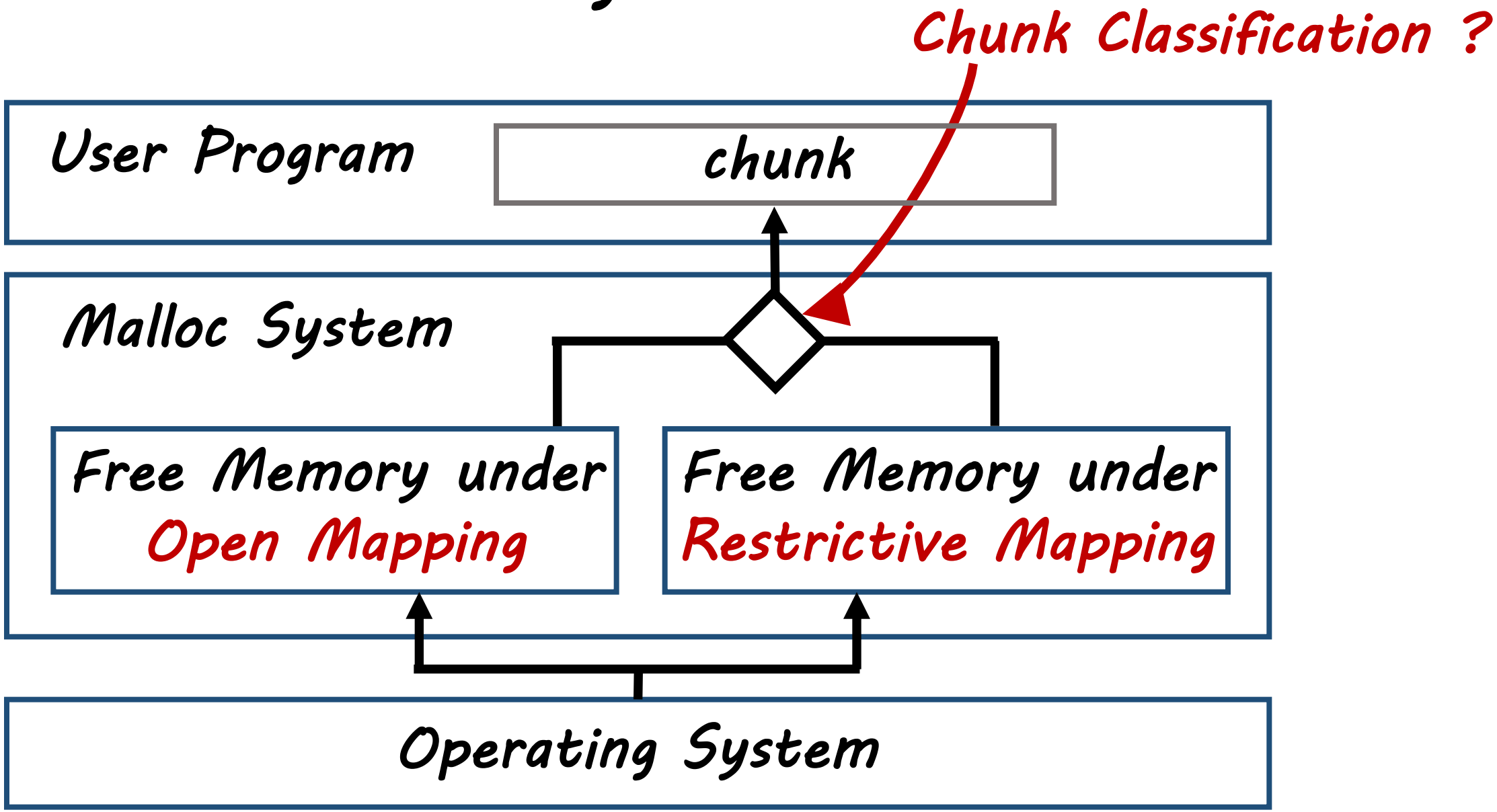
# Cache Resource Allocation



# Cache Resource Allocation



# The Big Picture



# Chunk Classification

Polluter Chunk

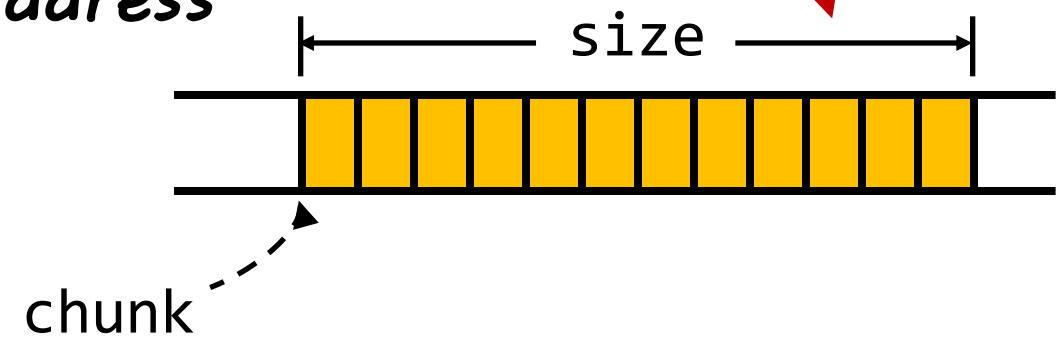
?

Normal Chunk

```
int* chunk = malloc(size);
```

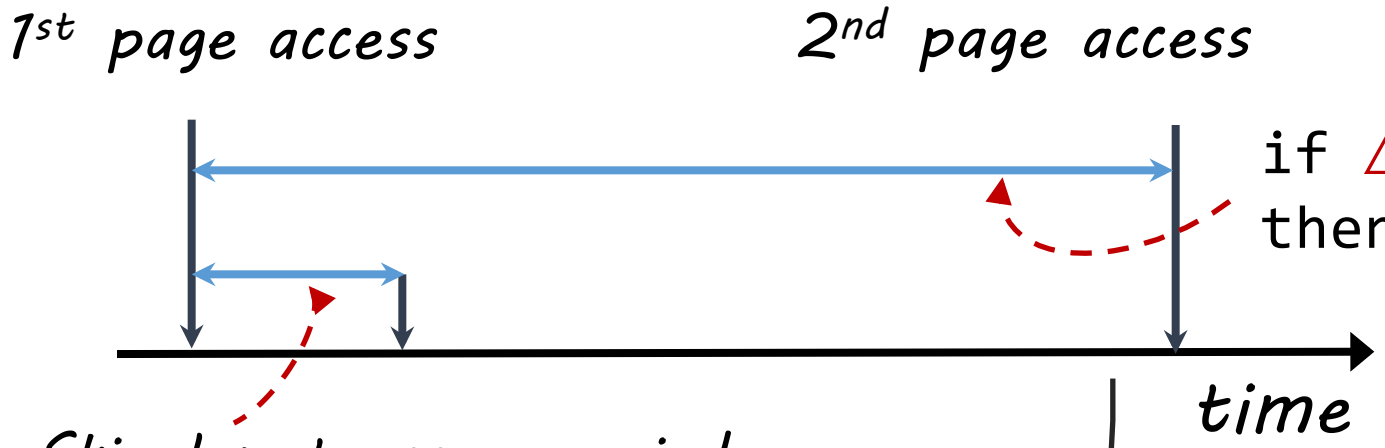
Sampling data access of this region, and estimate locality

Virtual Address



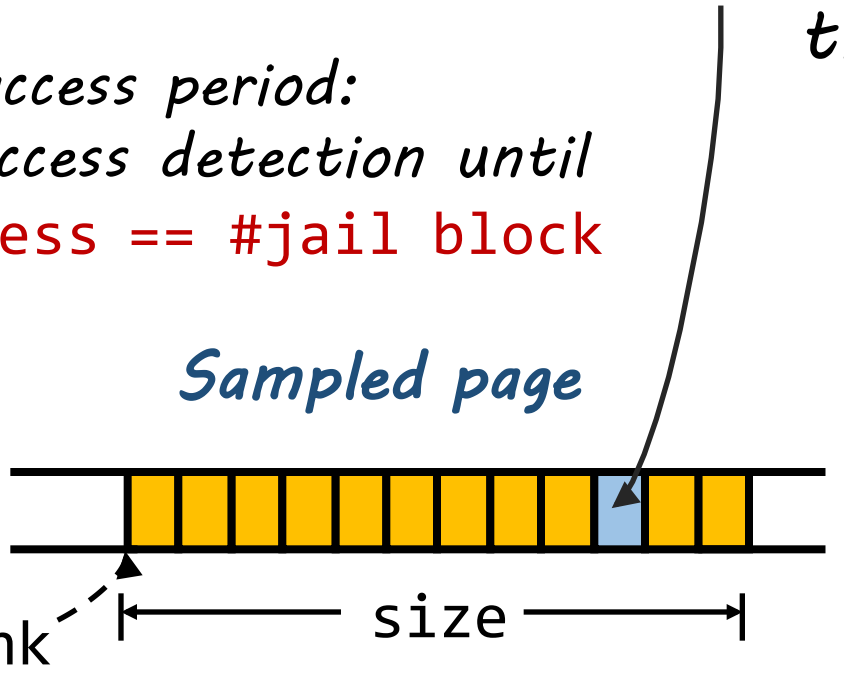

The sampling should be Lightweight, and should be built upon commodity hardware support

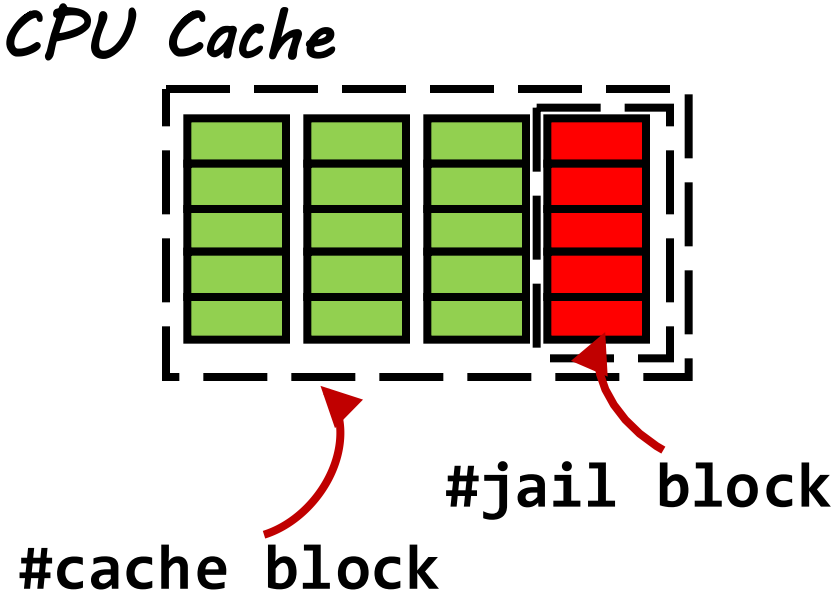
# Sampling Chunk Access



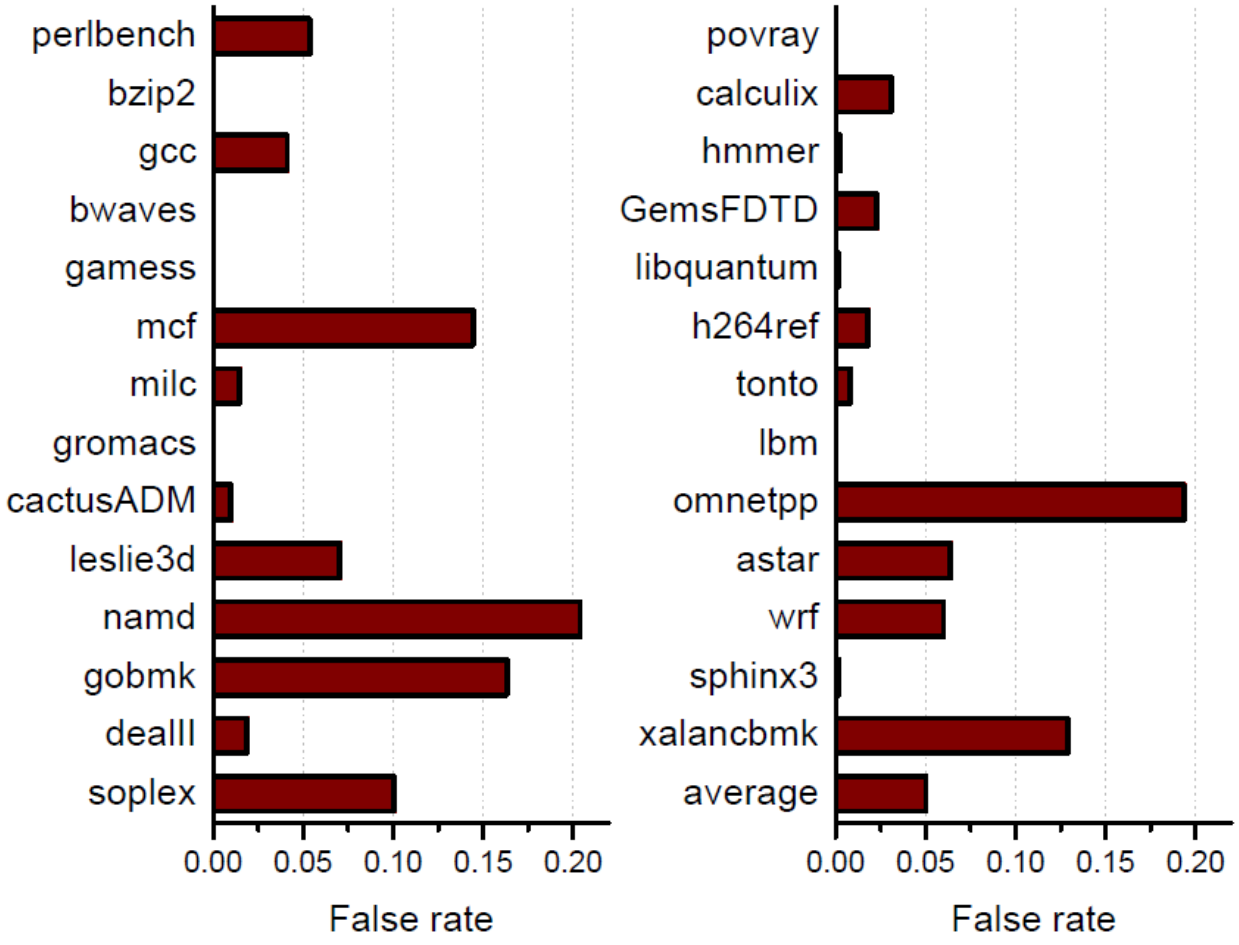

if △cache miss > #cache block
then 2nd page access is cache miss

Skip burst access period:
Stop page access detection until
△cache access == #jail block



# Sampling Chunk Access




Cache miss estimation false rate

1 million samples per program

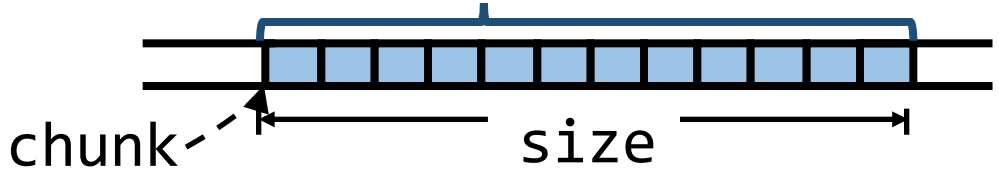
Average false rate: 6.0%

"`if △cache miss > #cache block then 2nd page access is cache miss`" is **conservative** estimation for cache miss.

Cache Miss → Cache Hit

# Intra-Chunk Locality Similarity

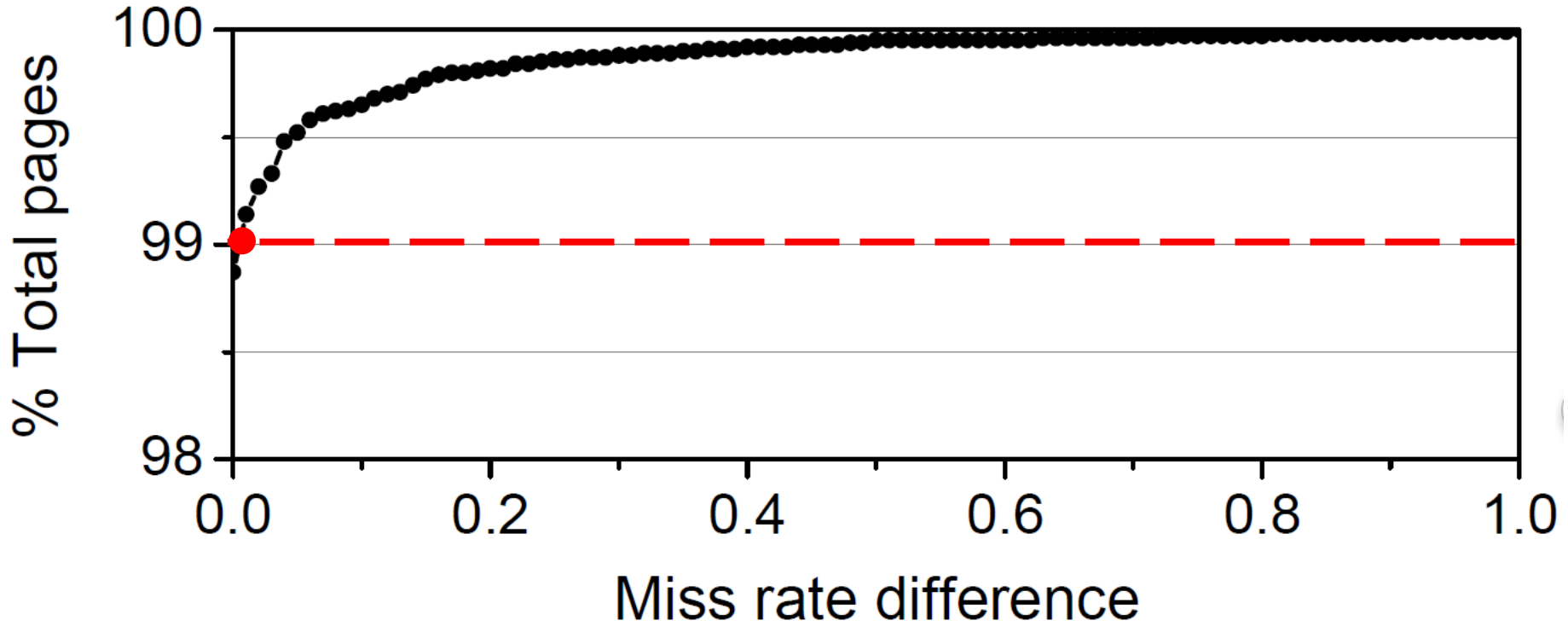
*Do we need to sample every page of a chunk?*




*only if pages differ significantly in their locality properties*

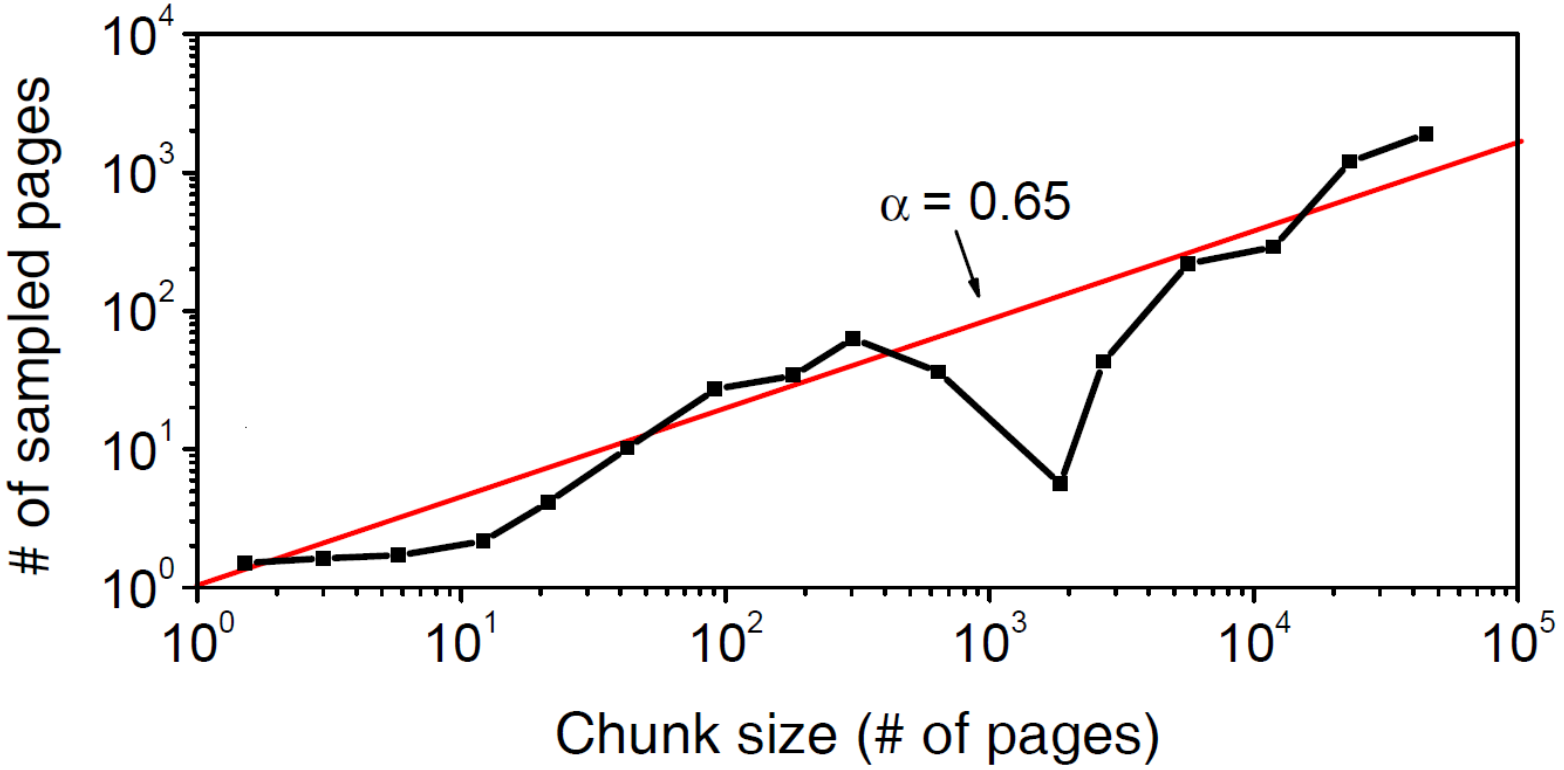
```
img->mb_data
    = calloc(img->FrameSizeInMbs, sizeof(Macroblock));
......
/* encode a picture */
while (NumberOfCodedMBs < img->total_number_mb){
    ......
    /* encode a macroblock in img->mb_data */
    encode_one_macroblock ();
    NumberOfCodedMBs++;
}
```

# Intra-Chunk Locality Similarity



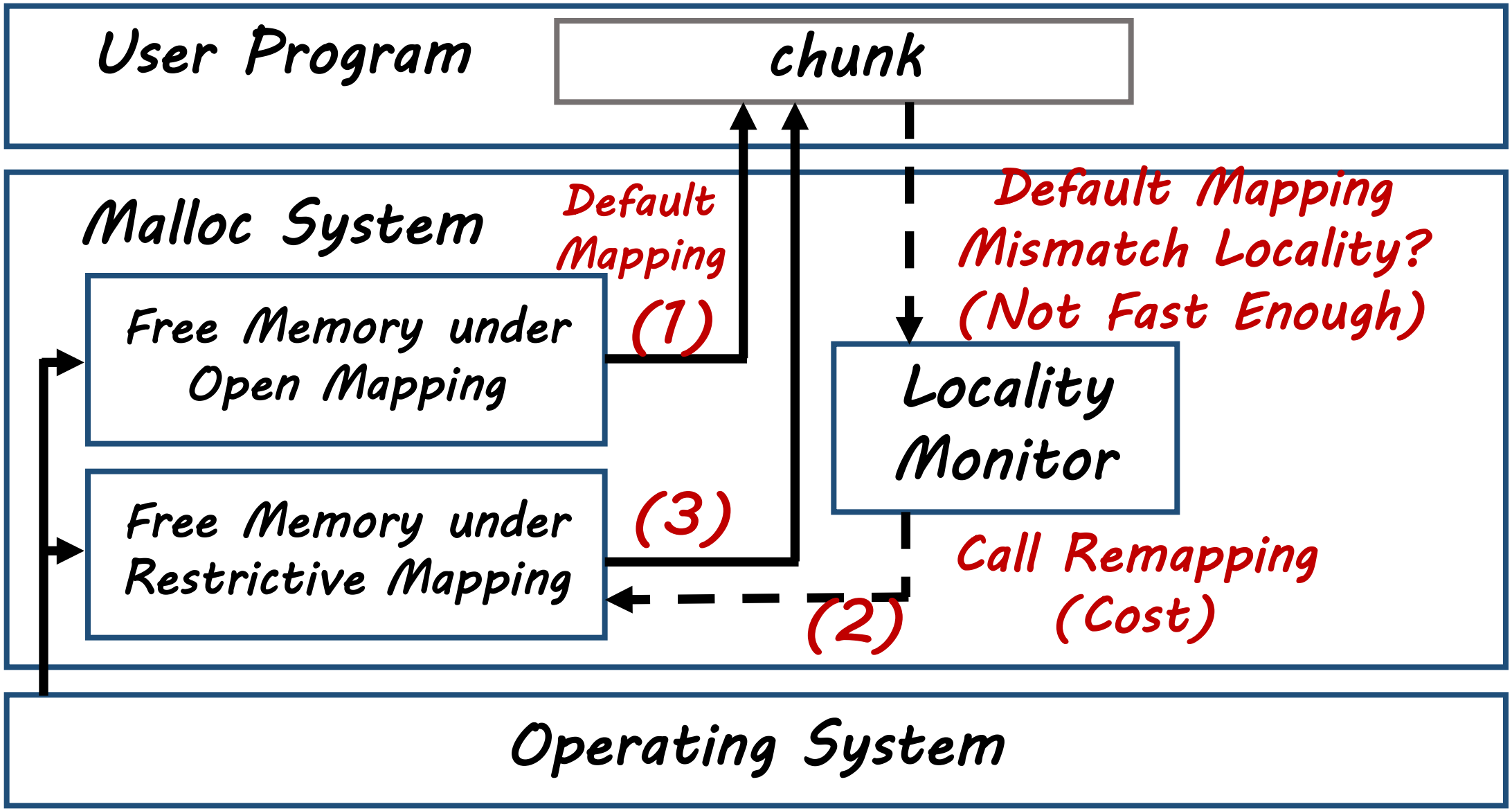

For the 27 programs tested:
Within chunks, 99% pages have a similar cache miss rate.

# Intra-Chunk Locality Similarity




For a chunk with N pages, only $N^{0.65}$ pages need to be sampled to guarantee >95% monitoring accuracy

# Is An Efficient Monitor Enough?



# Chunk Type Prediction

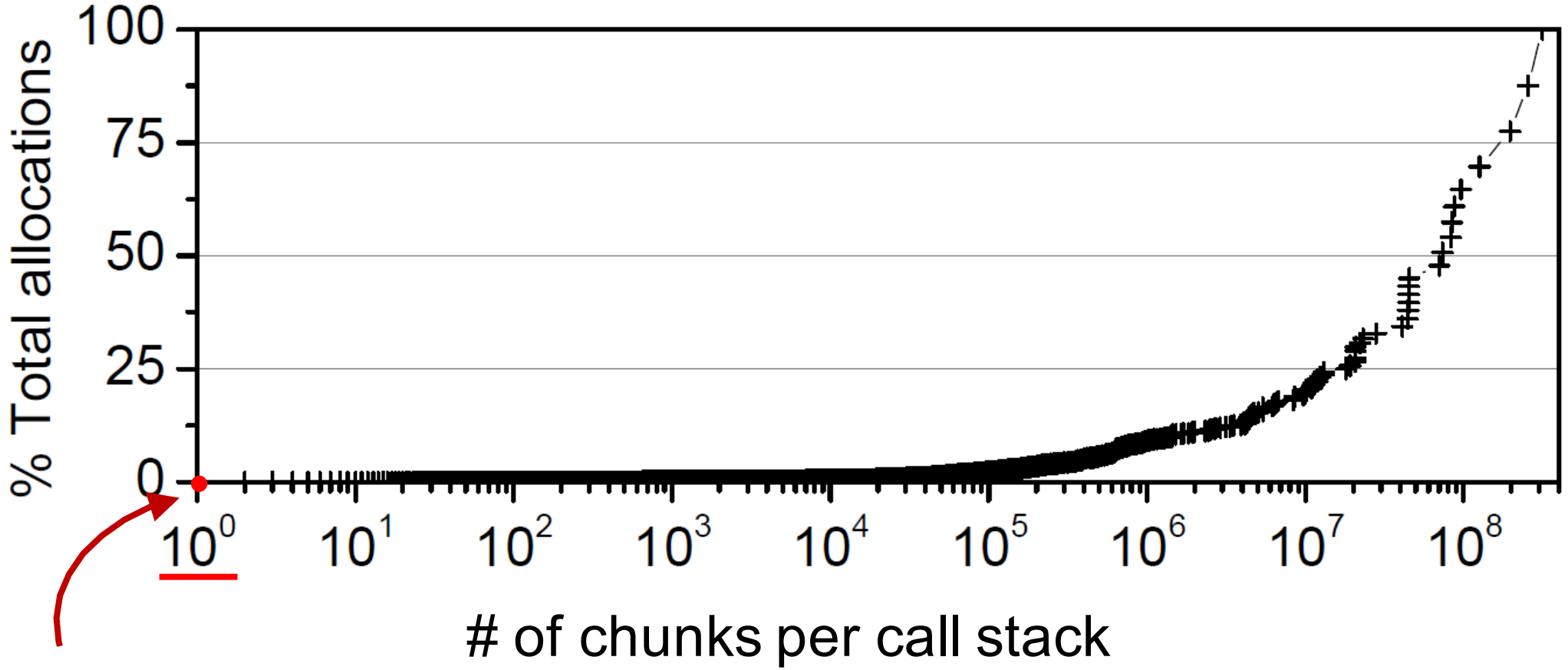
## Can we know the Chunk's type BEFORE it is used?

```
for (img->number=0; img->number < input->no_frames;
        img->number++) {
    ……
    buf = malloc (xs * ys * symbol_size_in_bytes);
    /* read one frame */
    read(p_in, buf, bytes_y);
    /* convert file read buffer to sour
    buf2img(imgY_org_frm, buf, xs, ys,
    ……
    free (buf);
}
```

Call stack

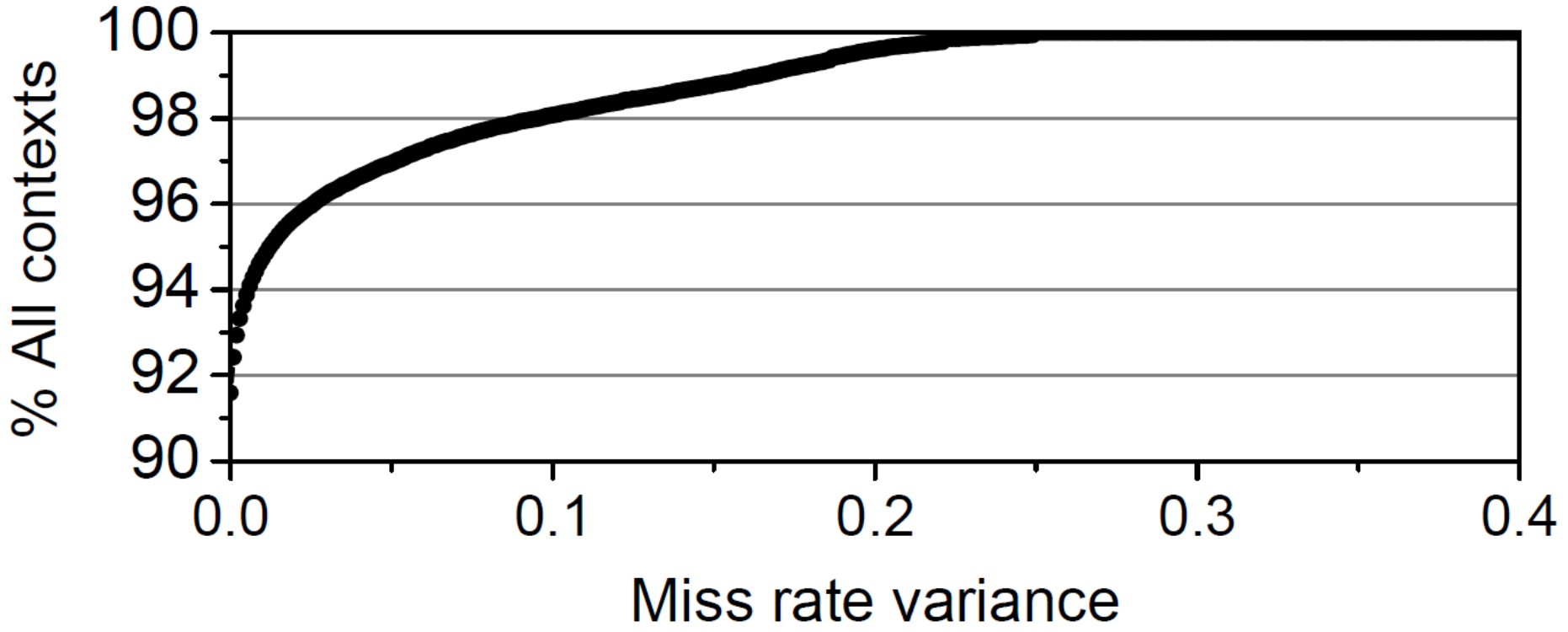
```
malloc()    0x3FF..2E
ld_frame()  0x80A3633
……
main()      0x8048757
_start()    0xAF9C37
```

# Enough Opportunity for Prediction




Chunks that do not share call stack

# Inter-Chunk Locality Similarity




Over 90% of the chunks have a same miss rate with other chunks that share the same call stack

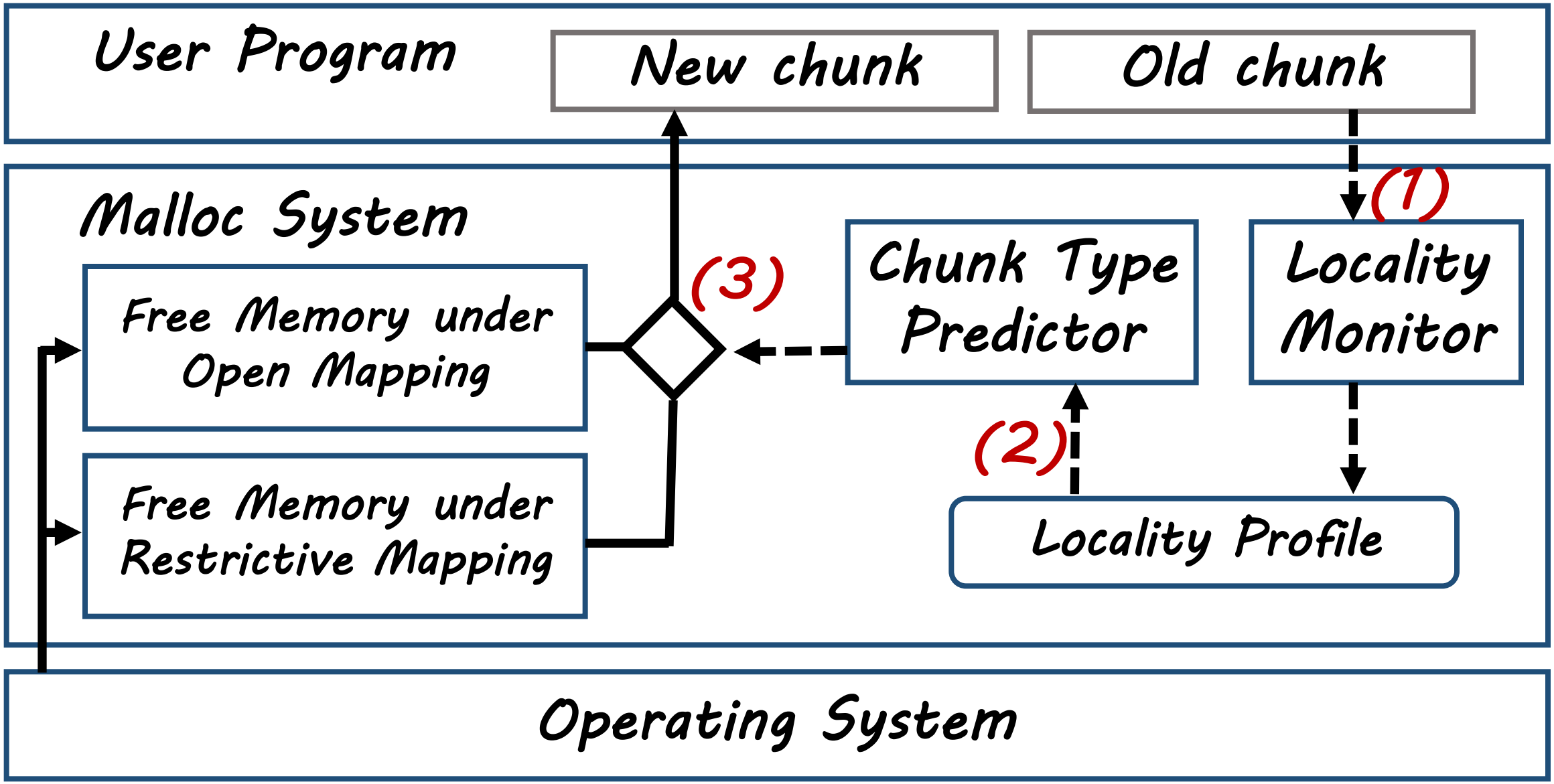
# Chunk Type Prediction Accuracy

| Benchmark | Succ. rate | Benchmark | Succ. rate |
|---|---|---|---|
| perlbench | 99.8% | povray | 99.7% |
| bzip2 | 95.8% | calculix | 92.9% |
| gcc | 96.2% | hmmer | 99.5% |
| bwaves | 99.3% | GemsFDTD | 100.0% |
| gamess | 98.6% | libquantum | 77.8% |
| milc | 99.7% | h264ref | 92.2% |
| gromacs | 100.0% | tonto | 100.0% |
| cactusADM | 100.0% | omnetpp | 100.0% |
| leslie3d | 100.0% | astar | 97.5% |
| namd | 58.1% | wrf | 99.4% |
| gobmk | 92.8% | sphinx3 | 100.0% |
| dealII | 93.6% | xalancbmk | 100.0% |
| soplex | 94.3% | **Average** | **95.5%** |

27 Programs

Average Prediction Success Rate:
95.5%

# Put Everything Together

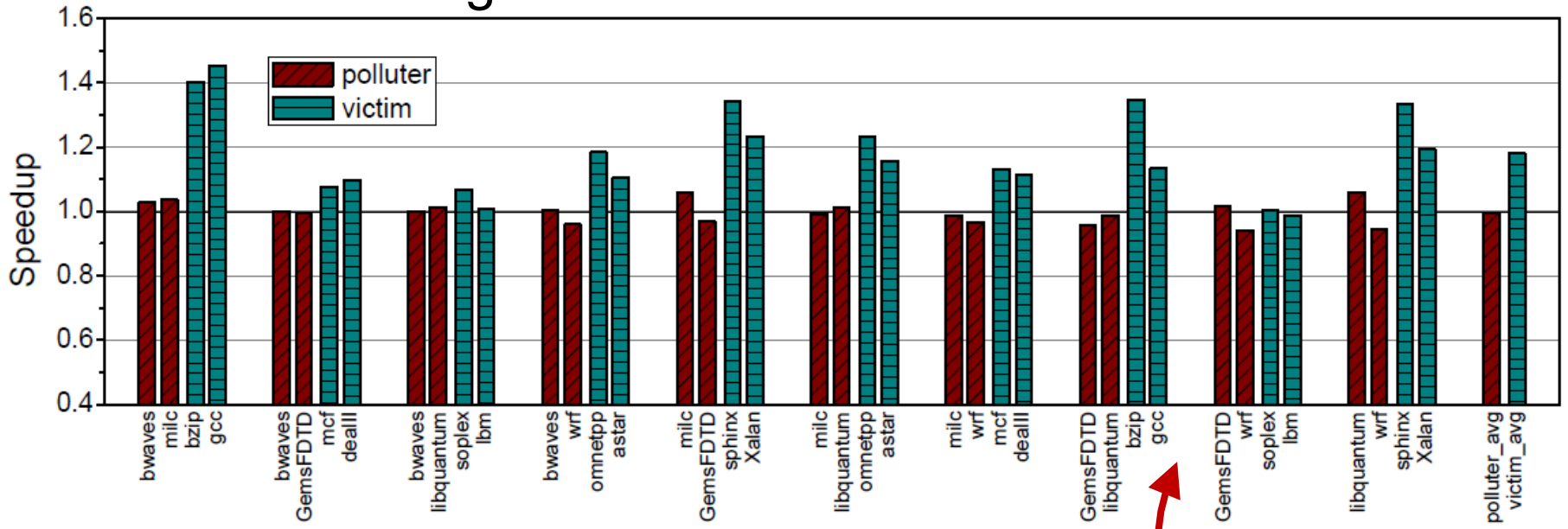


# Experiment Setup

| Benchmark Program Classifications | | | |
|---|---|---|---|
| Category | Cache sensitivity (Slowdown with 1/8 Cache ) | cache access rate (#access per 1k cycle) | Programs |
| **Polluter** | < 10% | > 5 | 410.bwaves 433.milc 459.GemsFDTD 462.libquantum 481.wrf |
| **Victim** | > 20% | -- | 401.bzip2 403.gcc 429.mcf 447.dealII 450.soplex 470.lbm 471.omnetpp 473.astar 482.sphinx3 483.xalancbmk |
| **Neutral** | [10%, 20%] | < 5 | 400.perlbench 416.gamess 435.gromacs 436.cactusADM 437.leslie3d 444.namd 445.gobmk 453.povray 454.calculix 456.hmmer 464.h264ref 465.tonto |

# Performance Evaluations

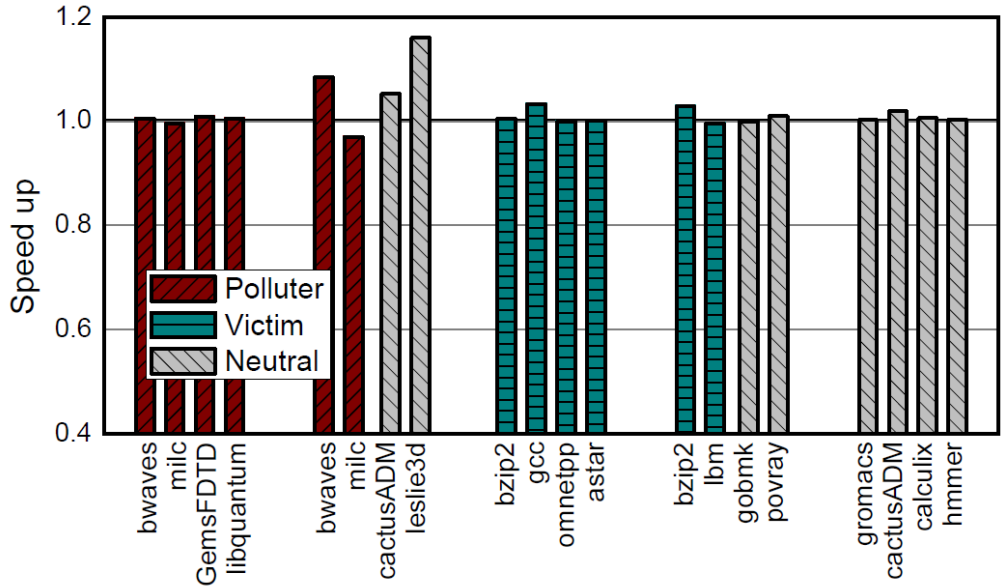
NightWatch+tcmalloc vs. tcmalloc




Victim

Polluter

Neutral

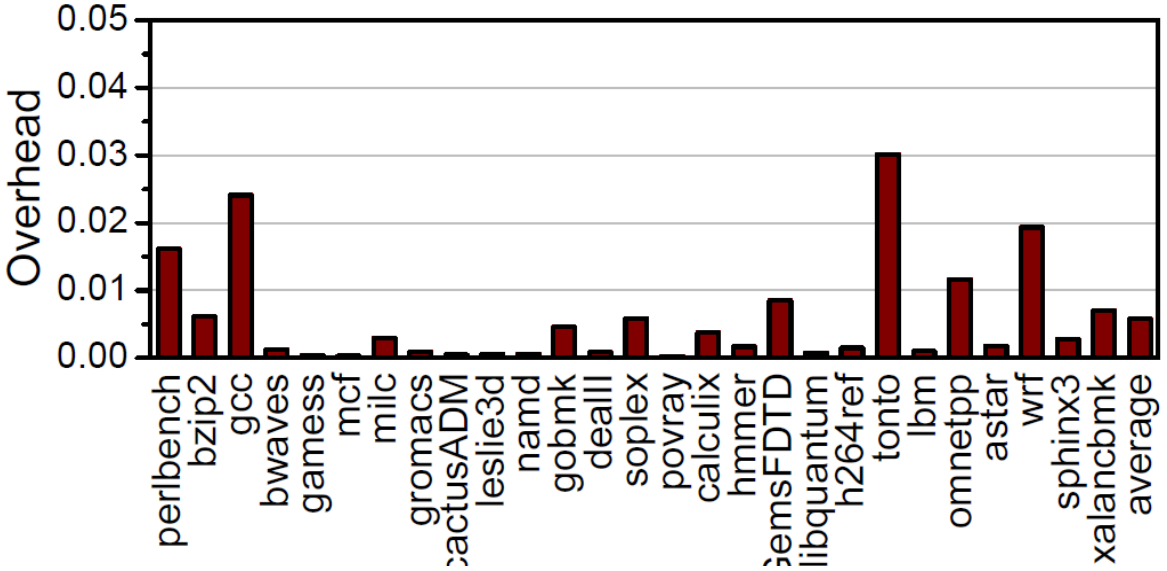



**Polluter + Victim**

Victims' average speedup 1.18,
highest speedup 1.45

NightWatch retains system performance when it cannot bring improvement

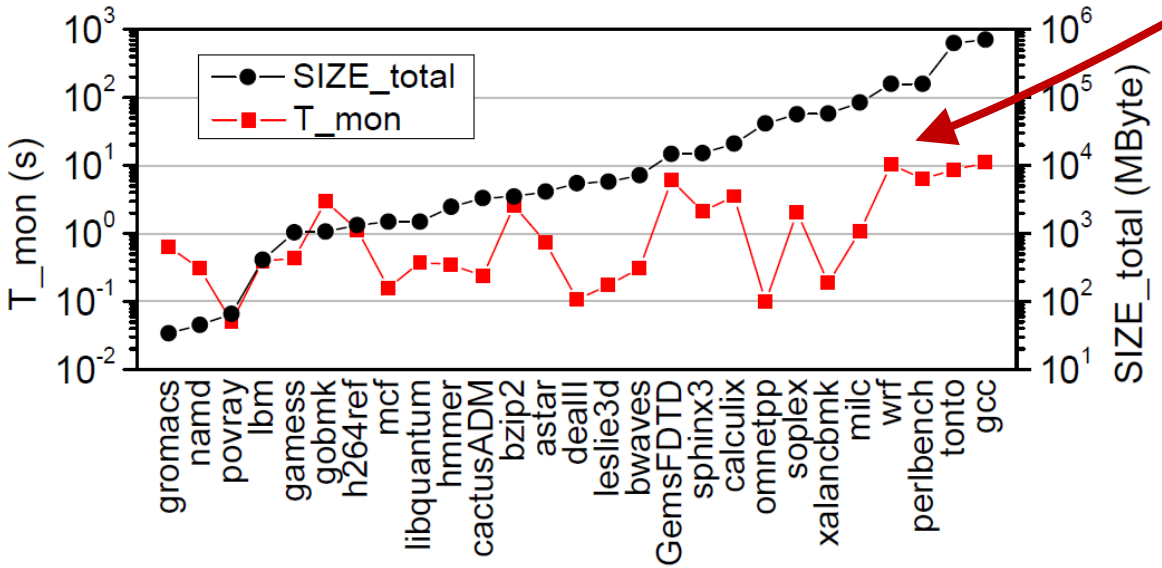
# System Overhead



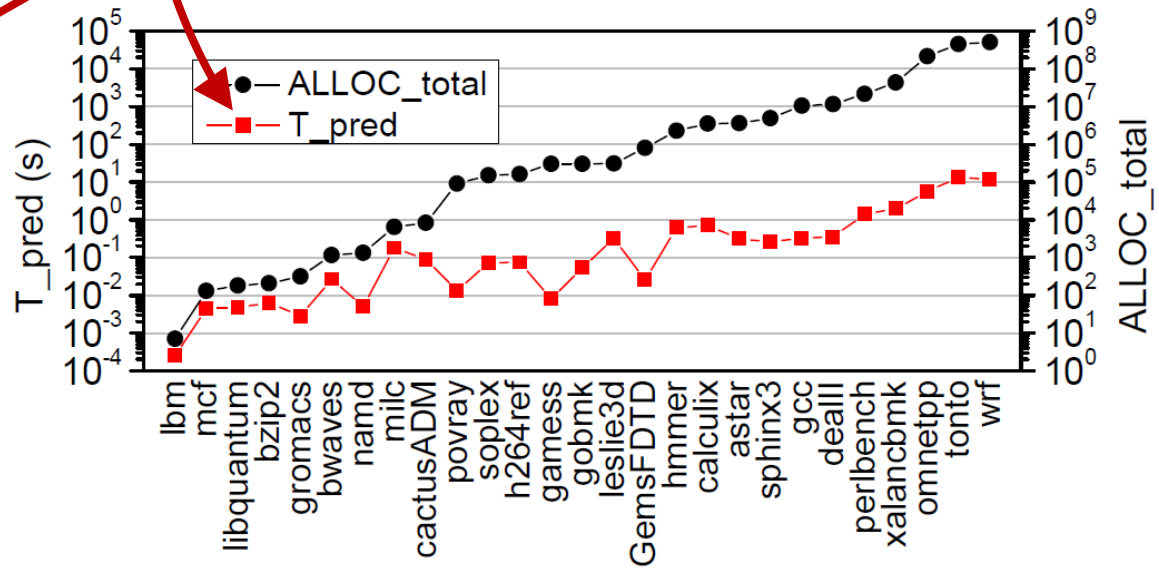

Overhead = $T_{NightWatch}$ / $T_{Total}$

Average overhead 0.57%,
Maximum overhead 3.02%

Scalability is guaranteed by
the Intra-Chunk Locality Similarity
And the Inter-Chunk Locality Similarity




Monitor's time cost as Sum(Chunk size) increases




Predictor's time cost as Sum(Chunk number) increases

# Conclusions

1. It is not only the memory matters in Malloc systems.
2. The Intra-Chunk and Inter-Chunk Locality Similarity make efficient chunk classification.
3. Integrating Cache Management into Malloc system offers notable performance improvement, with acceptable overhead.
4. Source code https://github.com/grtoverflow/pc-malloc

# Why the Name 'NightWatch'?

✗ Jon Snow and his brothers have contribution for this work.

√ The system helps the program protect the cache from being polluted.

# Questions?