Expanding CXL Software Ecosystem through HMSDK on Linux

MEMORY FABRIC FORUM | OCP GLOBAL SUMMIT | OCT 15-17, 2024 SAN JOSE, CA

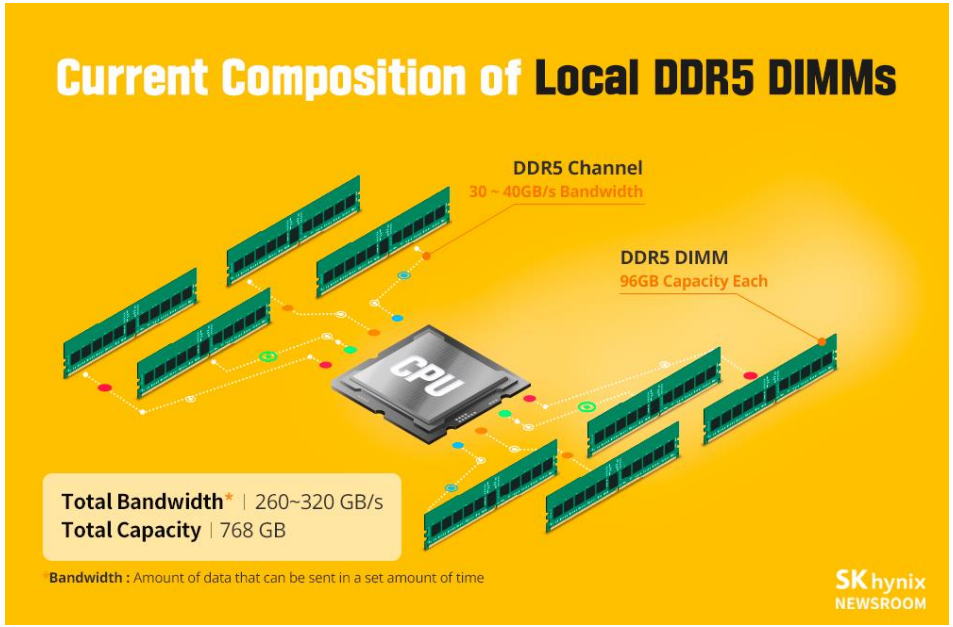
Expanding CXL Software Ecosystem through HMSDK on Linux

Honggyu Kim

SK hynix

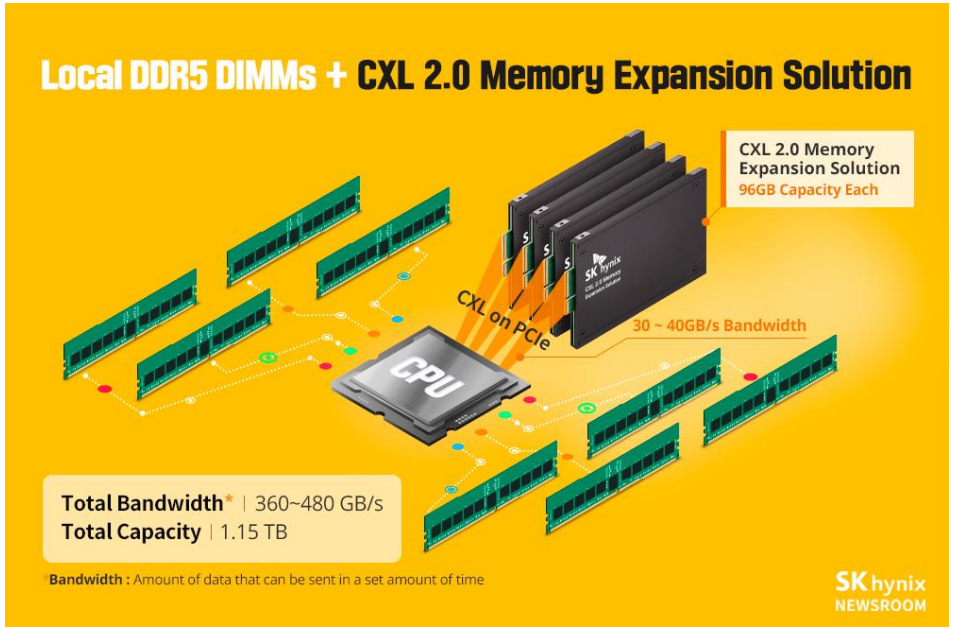
Conventional (Homogeneous) Memory System

- Most systems have the same type of memory in their DIMM slots.
  - It just works without software support.



Heterogeneous Memory System

- CXL memory allows bandwidth/capacity expansion beyond DIMM slot limits.
  - But it requires software support for efficient use.



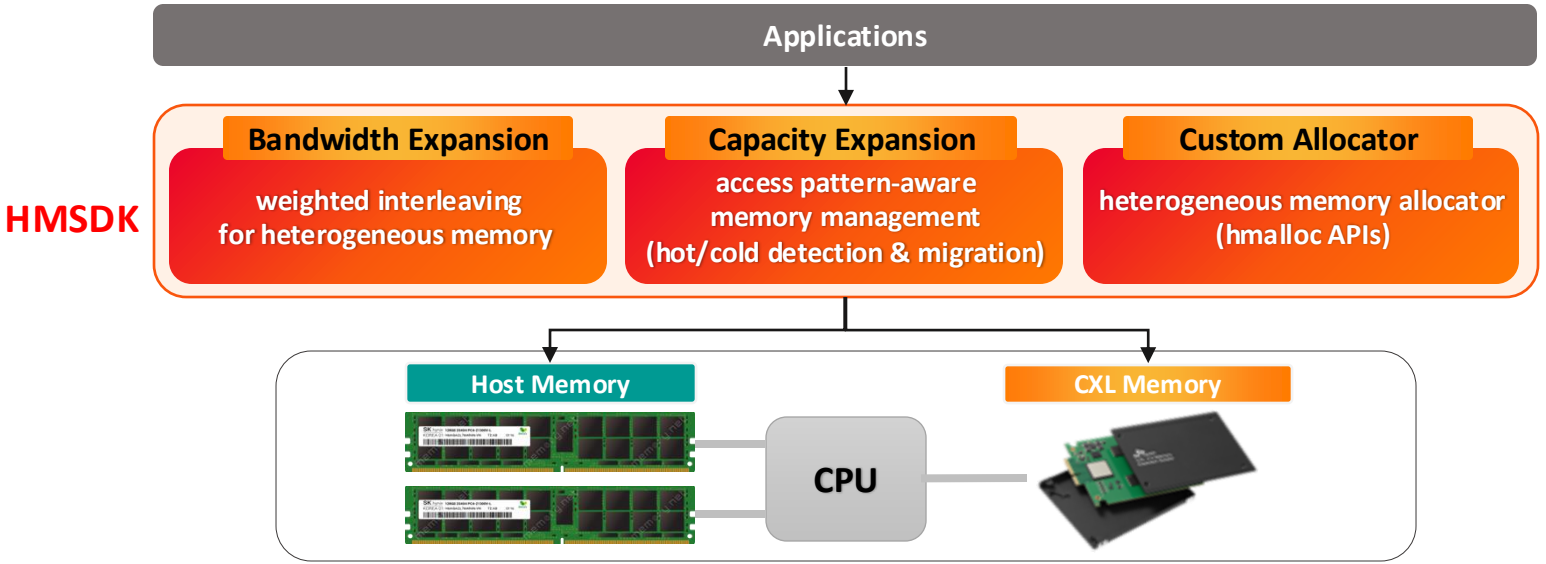
Software Support for CXL Memory

- CXL driver level support.
  - In Linux kernel, CXL device driver is at ***linux/drivers/cxl***.

- Memory management(mm) support for the efficient use.
  - Based on NUMA abstraction
    - CXL memory is detected as a cpuless NUMA node.
  - We're working on this under HMSDK project.

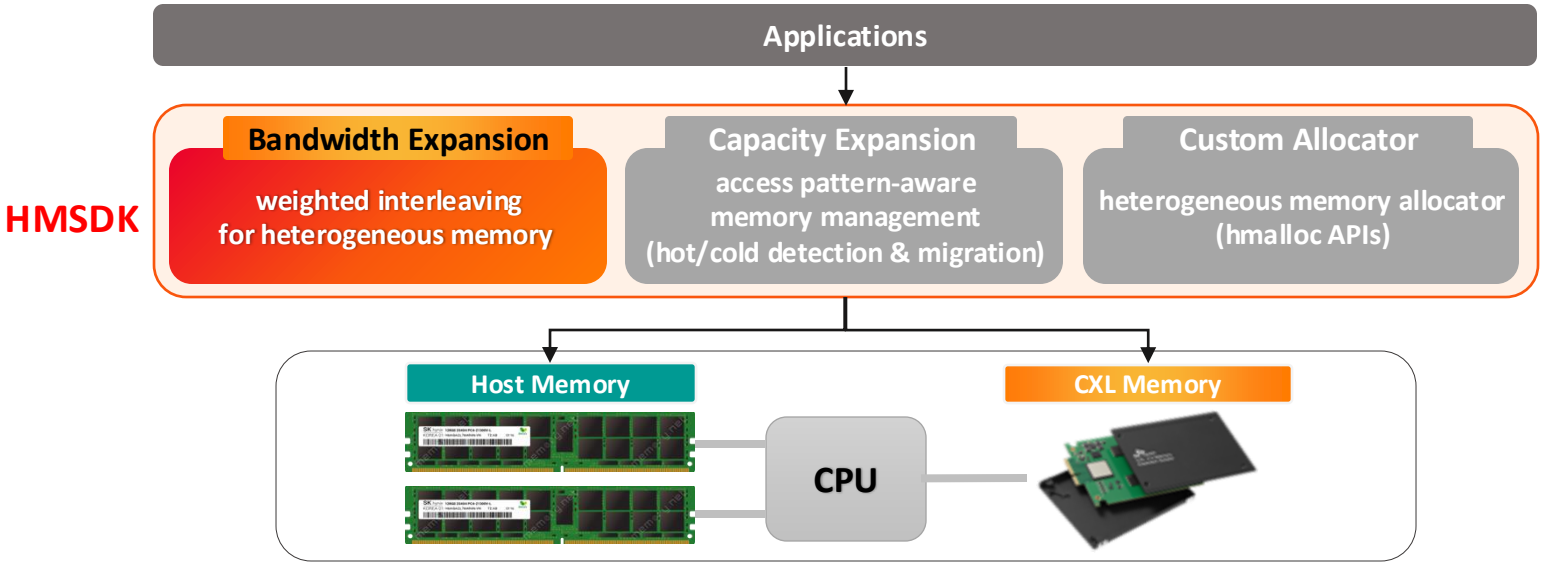
HMSDK

- Heterogeneous Memory Software Development Kit



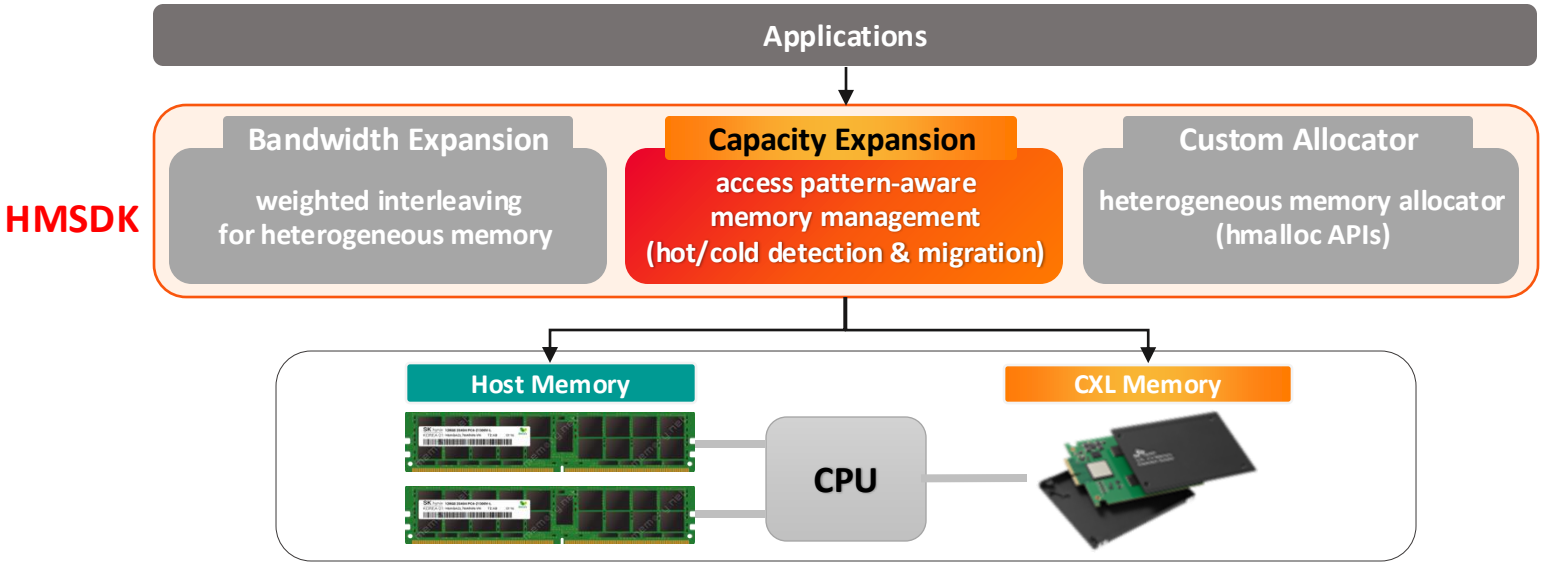
HMSDK

- **Bandwidth Expansion** is used when target workloads are **bandwidth hungry**.
  - Achieve speed up for bandwidth-intensive workloads.



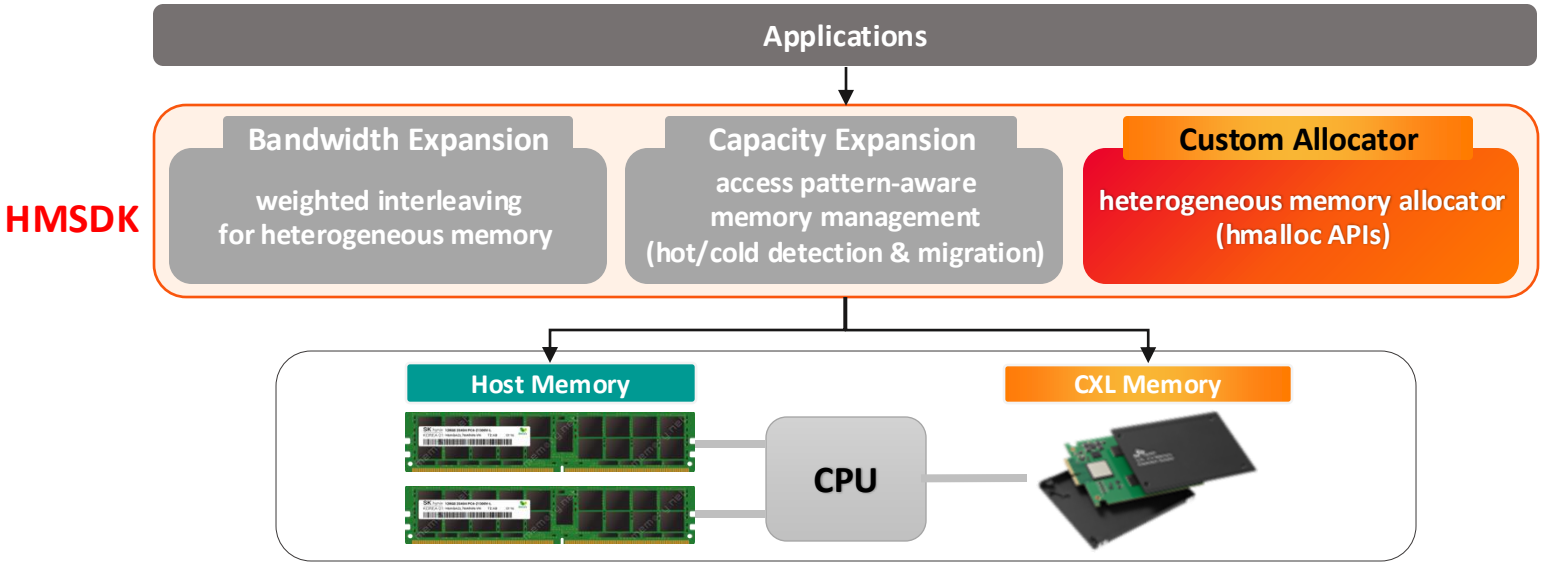
HMSDK

- **Capacity Expansion** is used when target workloads **needs more capacity**.
  - Expand capacity with CXL memory minimizing the additional latency issue.



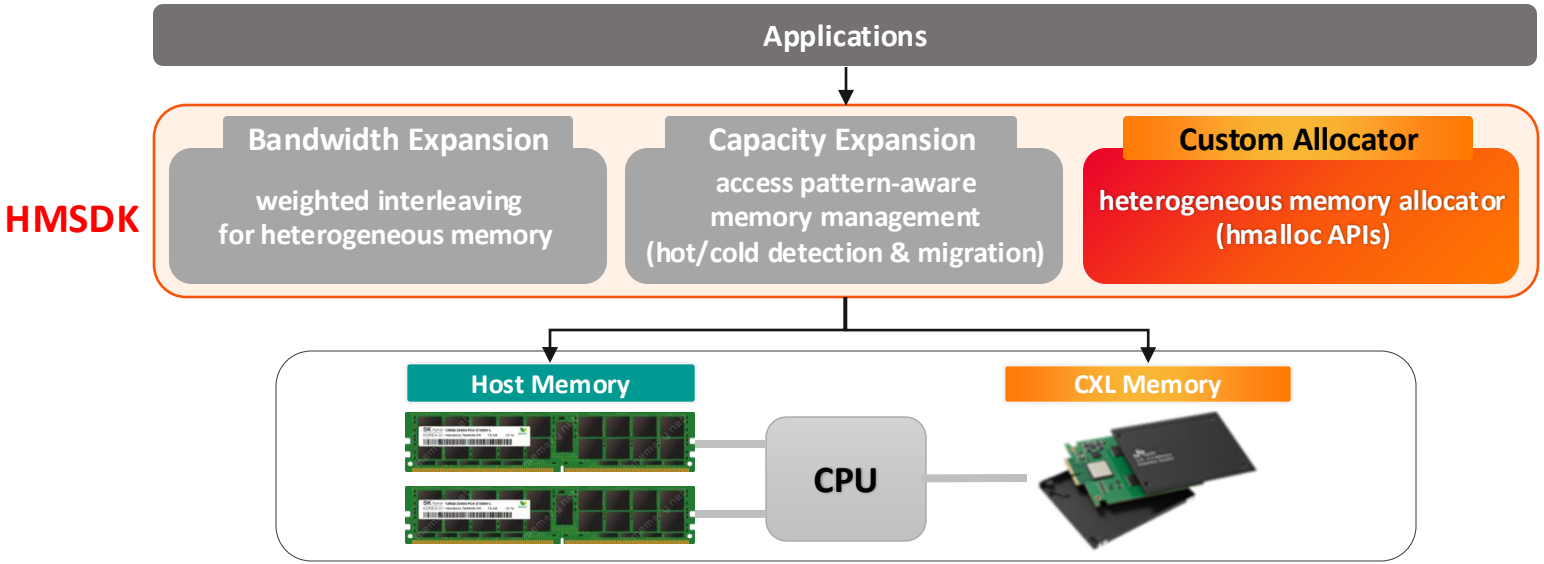
HMSDK

- **Custom Allocator** is used when **users have knowledge of their programs**.
  - With customized programs using **hmalloc APIs** such as ***hmalloc()***, ***hfree()***, etc.



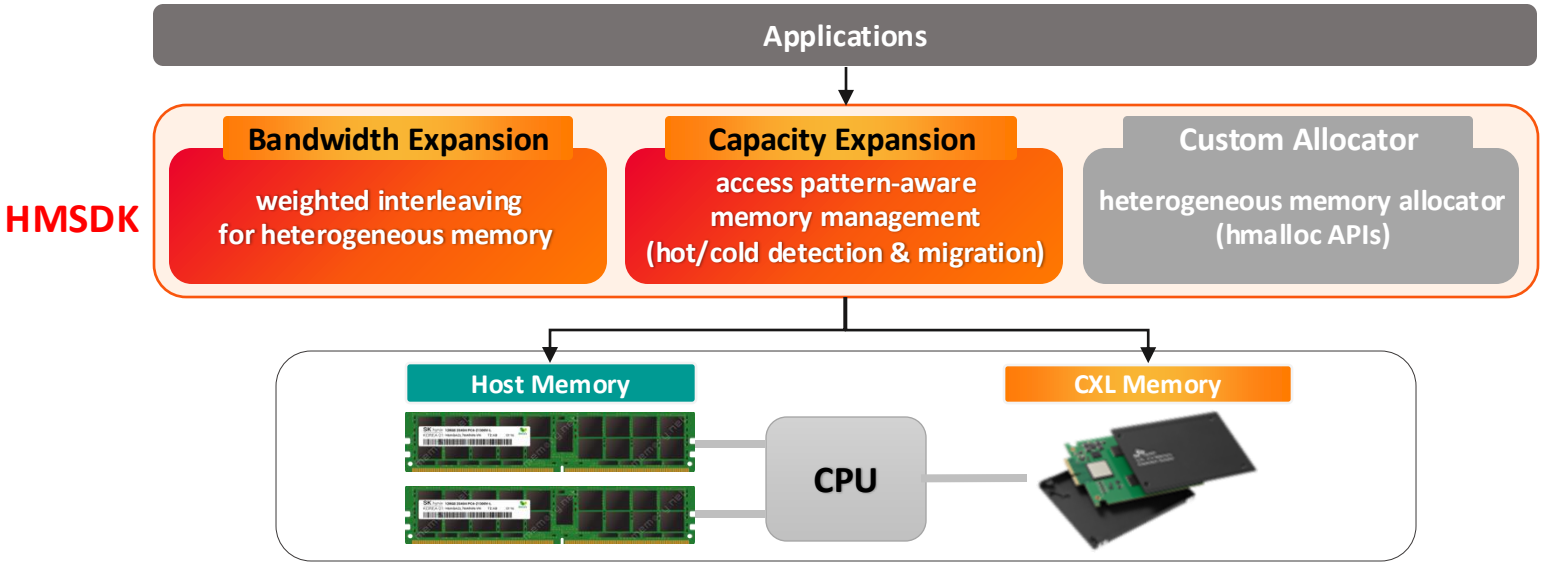
HMSDK

- Custom allocator **requires software modification**.



HMSDK

- Bandwidth and capacity expansion **DO NOT require software modification**.
  - Those are OS level techniques.



Bandwidth and Capacity Expansion

- No free lunch for bandwidth and capacity expansion.
  - Unlike DIMM attached host memory.

- Bandwidth can be expanded with default 1:1 interleaving.
  - But may not utilize the full bandwidth of host memory.

- Capacity can be expanded but there can be performance issues.
  - Frequently accessed data might stay in CXL memory.

Bandwidth and Capacity Expansion

- No free lunch for bandwidth and capacity expansion.
  - Unlike DIMM attached host memory.

- Bandwidth can be expanded with default 1:1 interleaving.
  - But may not utilize the full bandwidth of host memory.

    -> Weighted interleave

- Capacity can be expanded but there can be performance issues.
  - Frequently accessed data might stay in CXL memory.

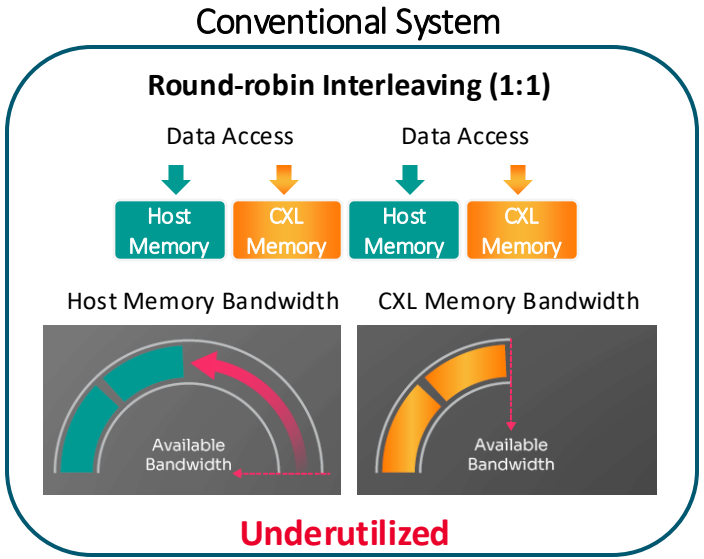
    -> Memory access pattern based migration

Bandwidth Expansion

Achieve speed up for bandwidth-intensive workloads

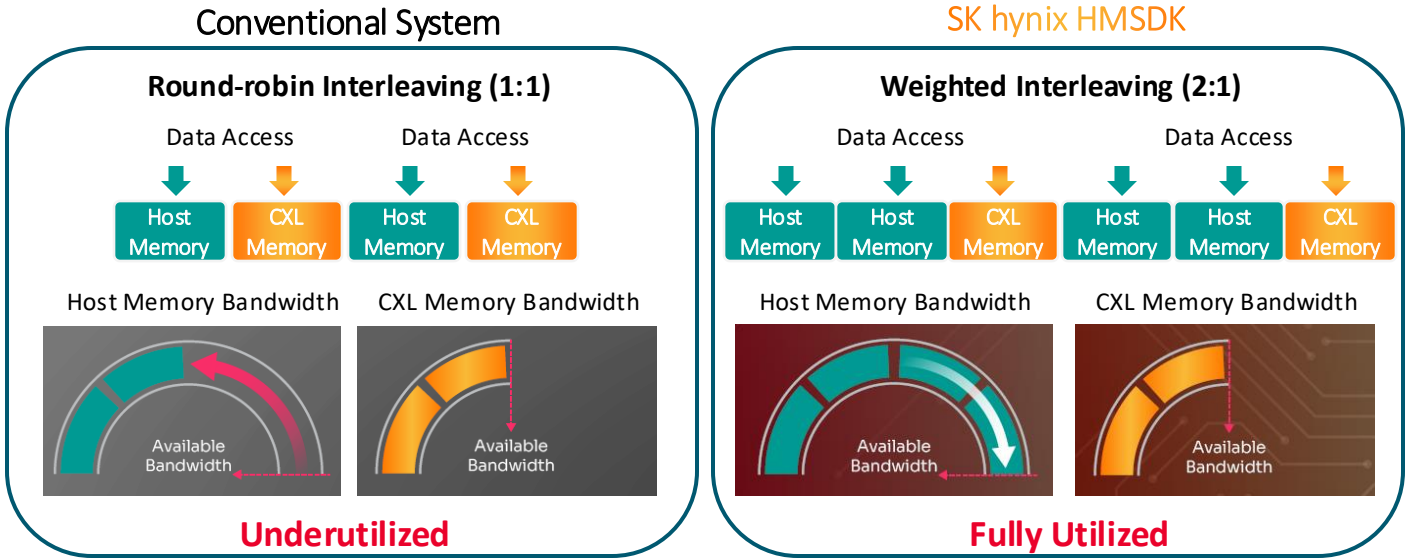
Bandwidth Expansion (via round-robin interleave)

- Bandwidth can be expanded with conventional 1:1 interleaving.
  - But it may not utilize the full bandwidth of host memory.



Bandwidth Expansion (via weighted interleave)

- Bandwidth can be expanded with conventional 1:1 interleaving.
  - But it may not utilize the full bandwidth of host memory.
  - Weighted interleaving is needed to handle the bandwidth difference.



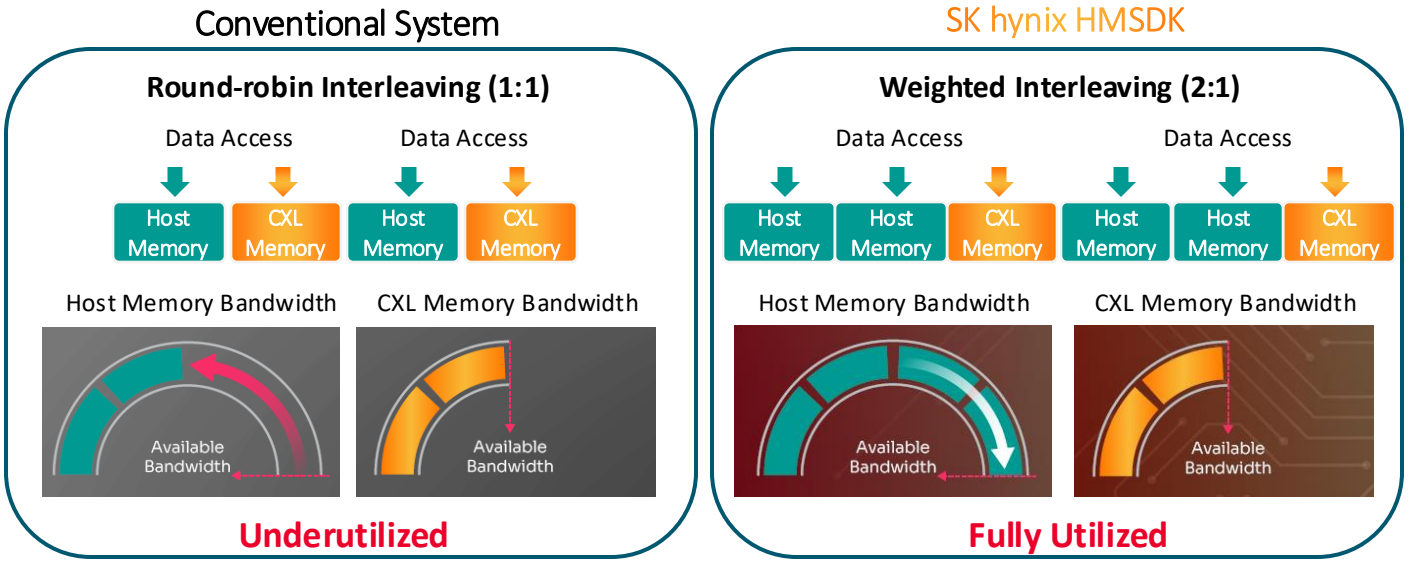
Bandwidth Expansion (via weighted interleave)

- The kernel interface: sysfs interface (**available from v6.9**)
  - ***/sys/kernel/mm/mempolicy/weighted_interleave/nodeN***
    - Where ***N*** is a node number.
    - It stores a value for weight.

- The user interface: ***numactl -w/--weighted-interleave <nodes>*** option.
  - Set the mempolicy to ***MPOL_WEIGHTED_INTERLEAVE***.
  - Set ***nodemask*** for the given NUMA ***nodes***.

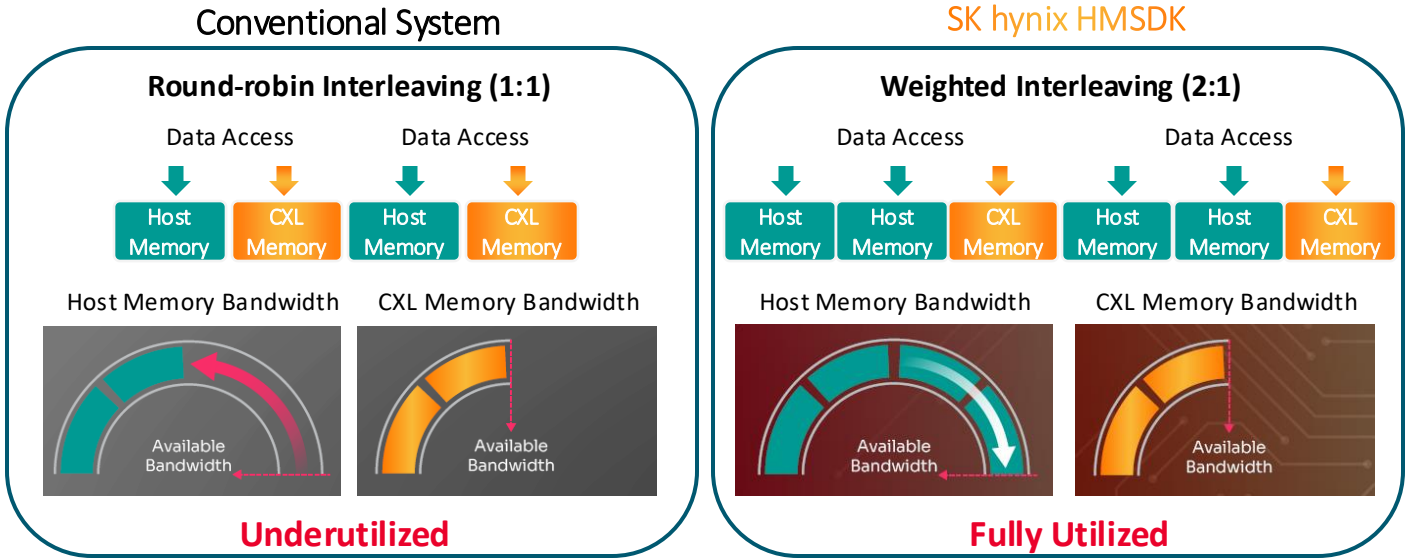
Bandwidth Expansion (via weighted interleave)

- Bandwidth can be expanded with weighted interleaving. (e.g. 2:1 interleaving)
  - `echo 2 > /sys/kernel/mm/mempolicy/weighted_interleave/node0  # DRAM node`
  - `echo 1 > /sys/kernel/mm/mempolicy/weighted_interleave/node1  # CXL node`



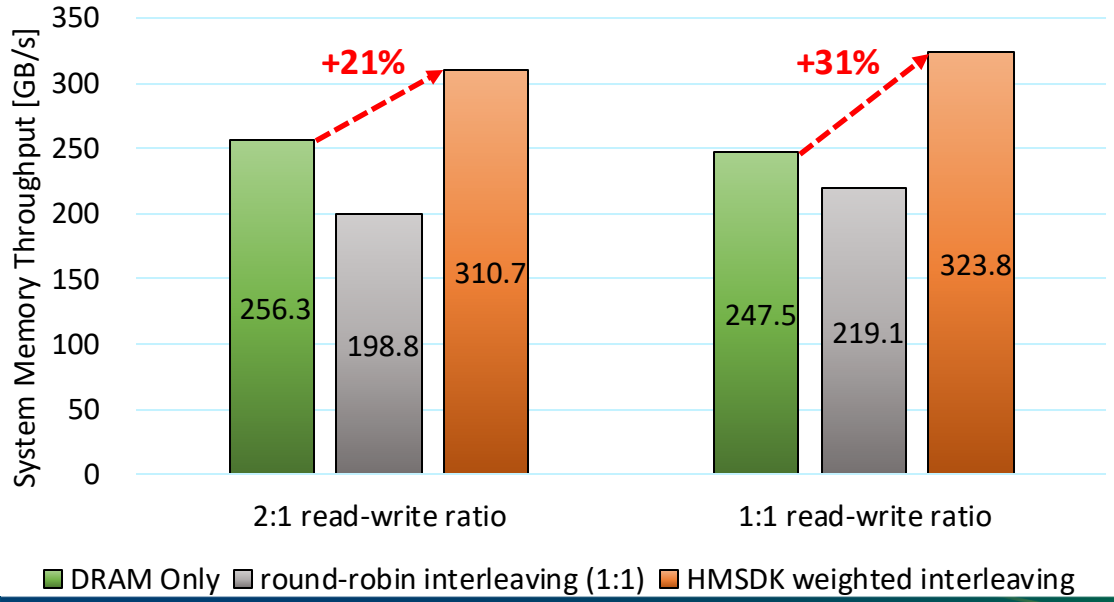
Bandwidth Expansion (via weighted interleave)

- Then run the target program, ***<prog>***, with ***numactl*** as follows.
  - ***numactl --weighted-interleave 0,1 <prog>***
    - It reads weights for **node0** and **node1** then run ***<prog>*** in weighted interleave mode.



Bandwidth Expansion (via weighted interleave)

- Experimental result using Intel MLC tool.
  - 8:3 weighted interleaving for 8ch DDR5 DRAM + 4ch CMM-DDR5

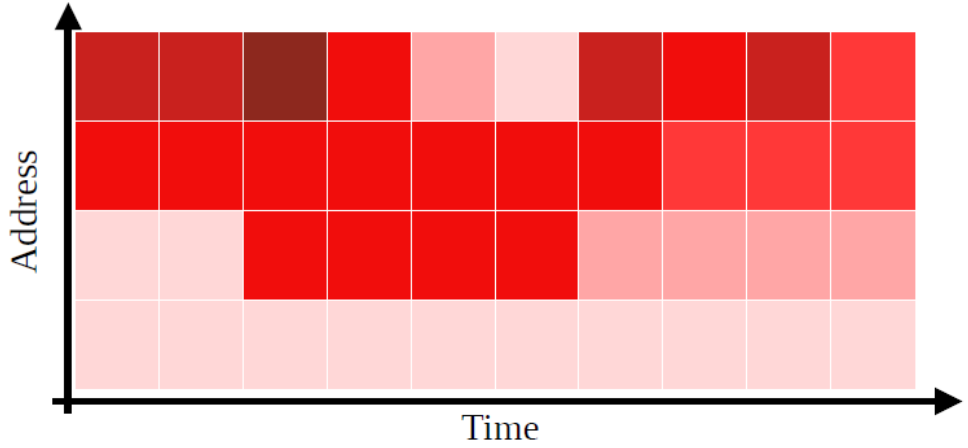


Capacity Expansion

Expand capacity with CXL memory
minimizing the additional latency overhead.

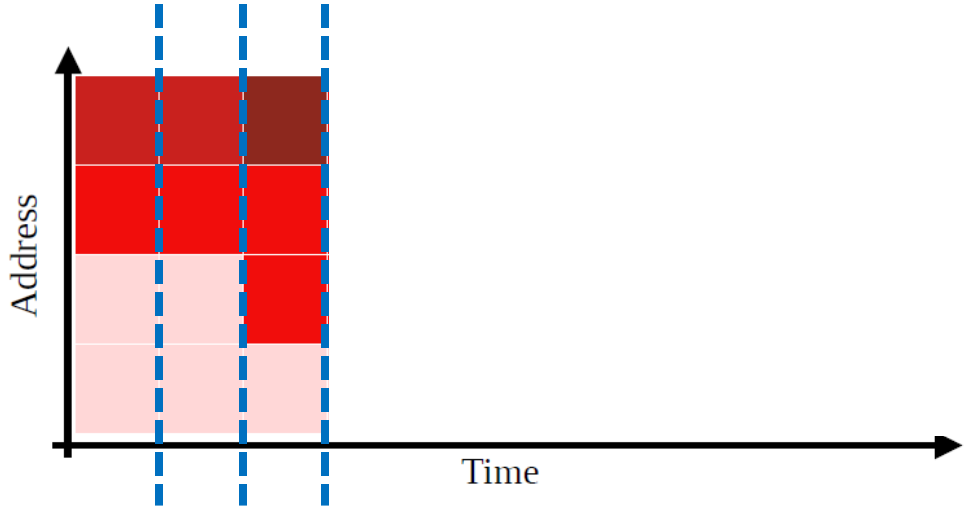
DAMON: Data Access MONitor

- DAMON is a data access monitoring framework in the Linux kernel.
  - Supported from Linux v5.15, released on Nov 2021.
- DAMON allows memory access checks in upper bounded overhead for scalability.
  - But scarifies some accuracy.



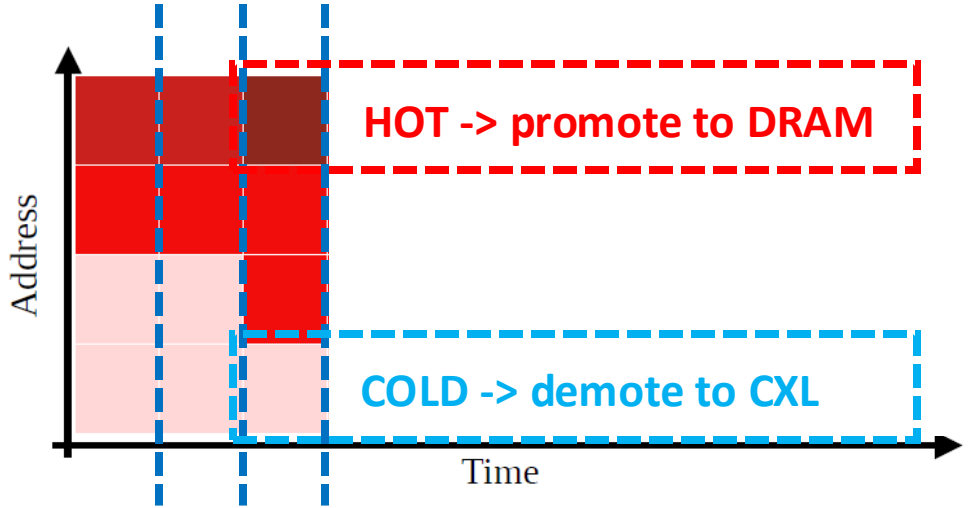
DAMOS: DAMON based Operation Schemes

- DAMOS is an access-aware system optimizations engine.
  - Make memory management decision based on DAMON.



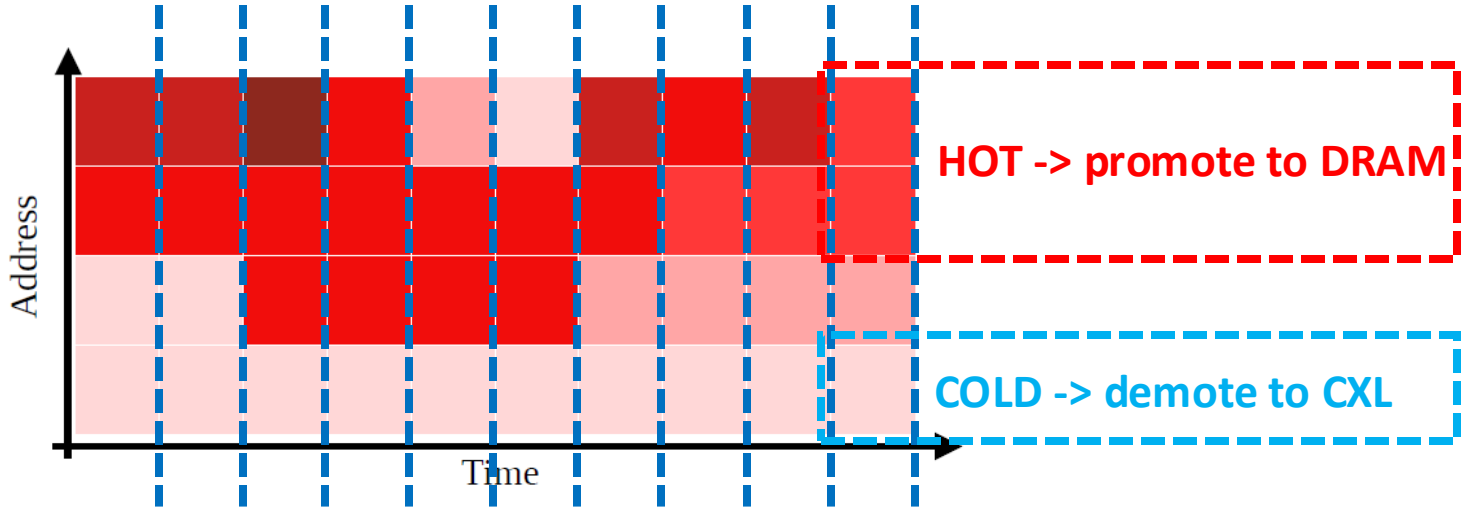
DAMOS: DAMON based Operation Schemes

- DAMOS is an access-aware system optimizations engine.
  - Make memory management decision based on DAMON.



DAMOS: DAMON based Operation Schemes

- DAMOS is an access-aware system optimizations engine.
  - Make memory management decision based on DAMON.



DAMOS: DAMON based Operation Schemes

- DAMOS is an access-aware system optimizations engine.
  - Make memory management decision based on DAMON.

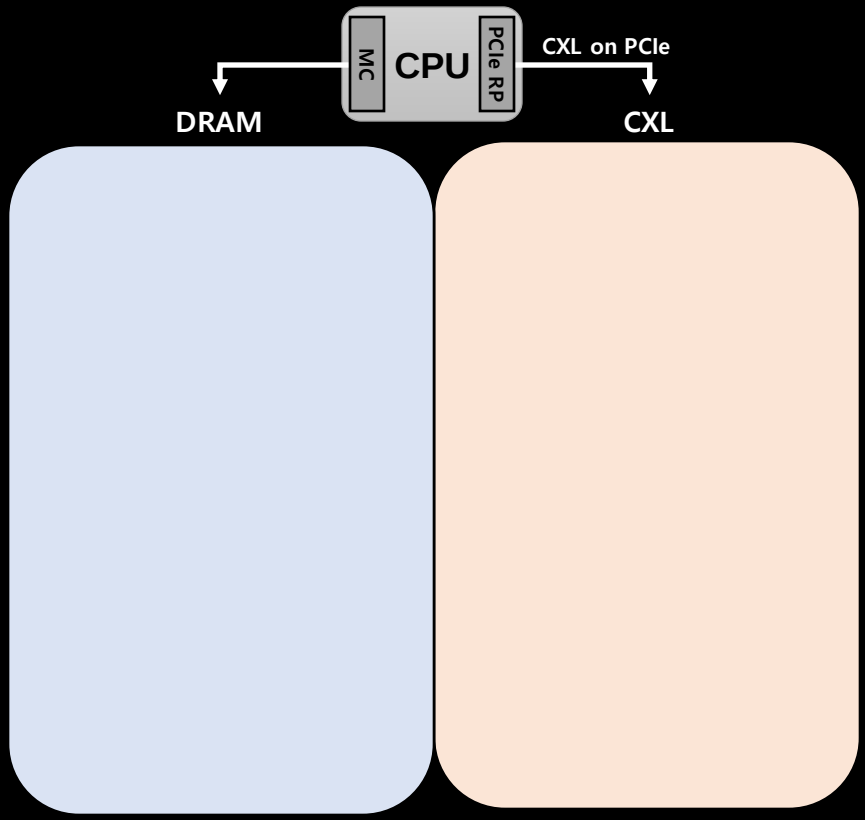
```
enum damos_action {
    DAMOS_WILLNEED,
    DAMOS_COLD,
    DAMOS_PAGEOUT,
    DAMOS_HUGEPAGE,
    DAMOS_NOHUGEPAGE,
    DAMOS_LRU_PRIO,
    DAMOS_LRU_DEPRIO,


    DAMOS_STAT,
    NR_DAMOS_ACTIONS,
};
```

DAMOS: DAMON based Operation Schemes

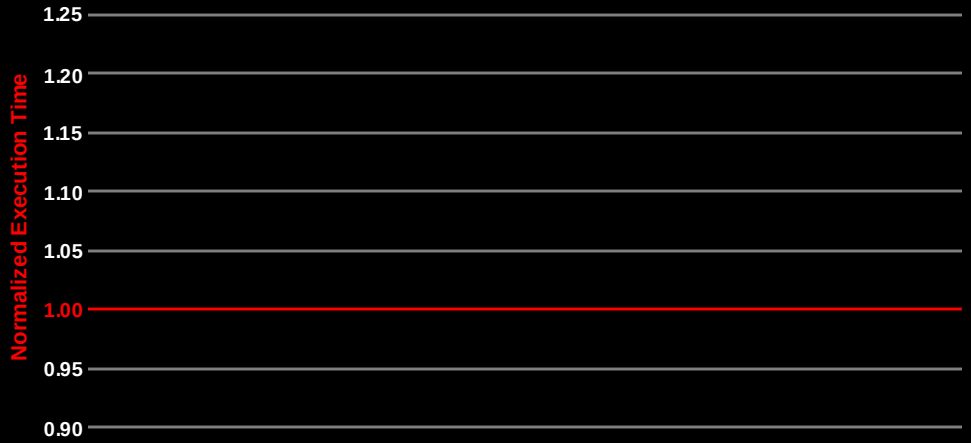
- DAMOS is an access-aware system optimizations engine.
  - Make memory management decision based on DAMON.
  - HMSDK implemented migration actions. (**available from Linux v6.11**)

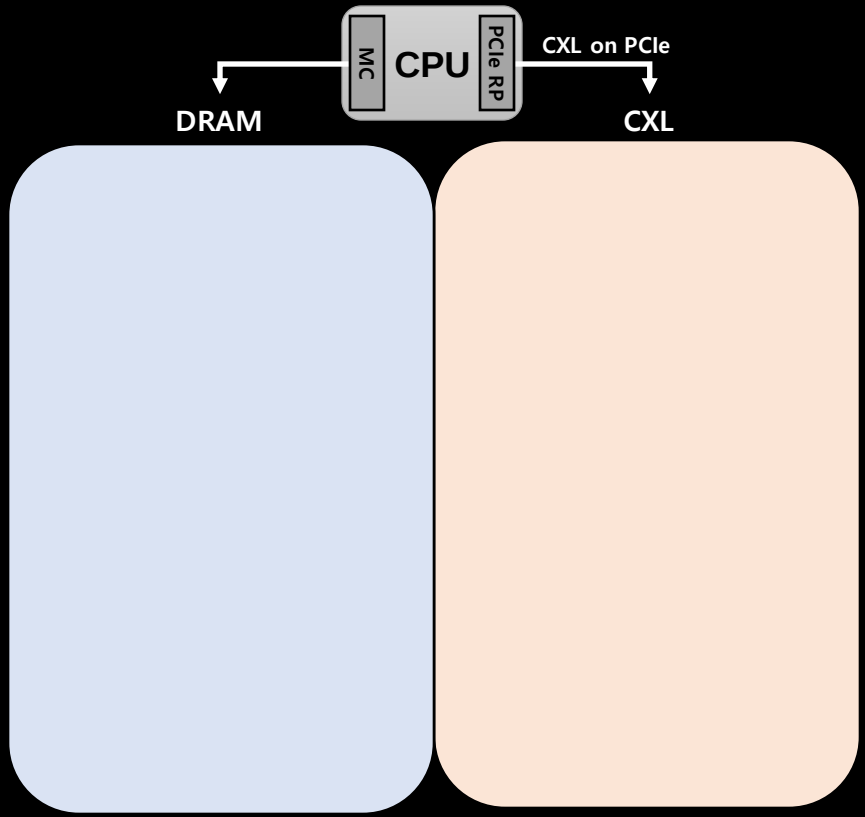
```
enum damos_action {
    DAMOS_WILLNEED,
    DAMOS_COLD,
    DAMOS_PAGEOUT,
    DAMOS_HUGEPAGE,
    DAMOS_NOHUGEPAGE,
    DAMOS_LRU_PRIO,
    DAMOS_LRU_DEPRIO,
    DAMOS_MIGRATE_HOT,  // promotion
    DAMOS_MIGRATE_COLD, // demotion
    DAMOS_STAT,
    NR_DAMOS_ACTIONS,
};
```

HMSDK Capacity Expansion

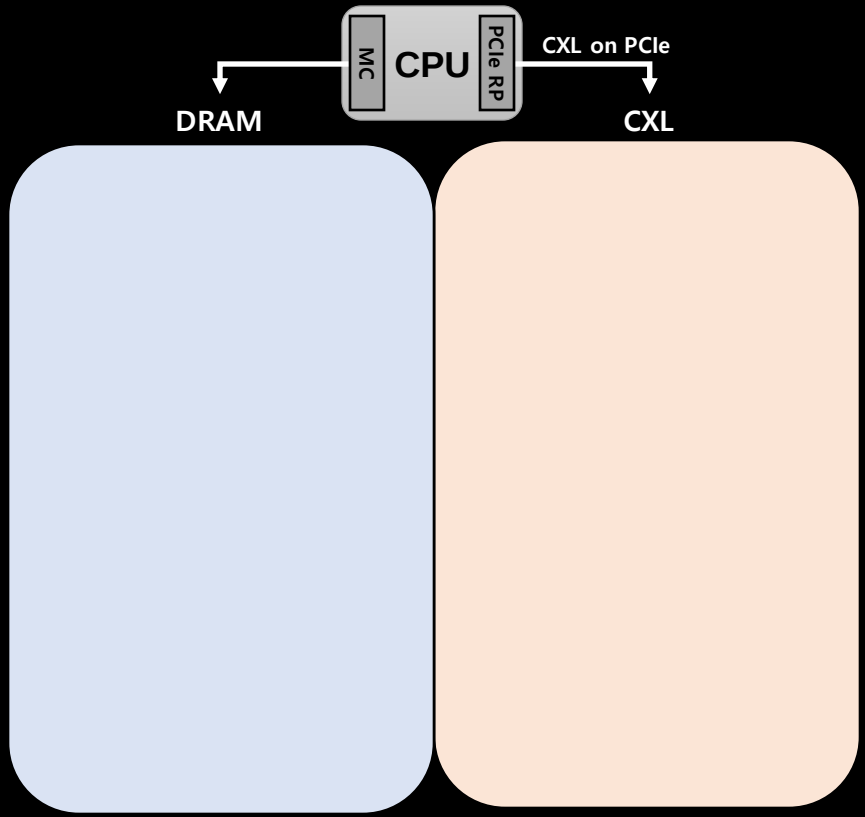


HMSDK Capacity Expansion

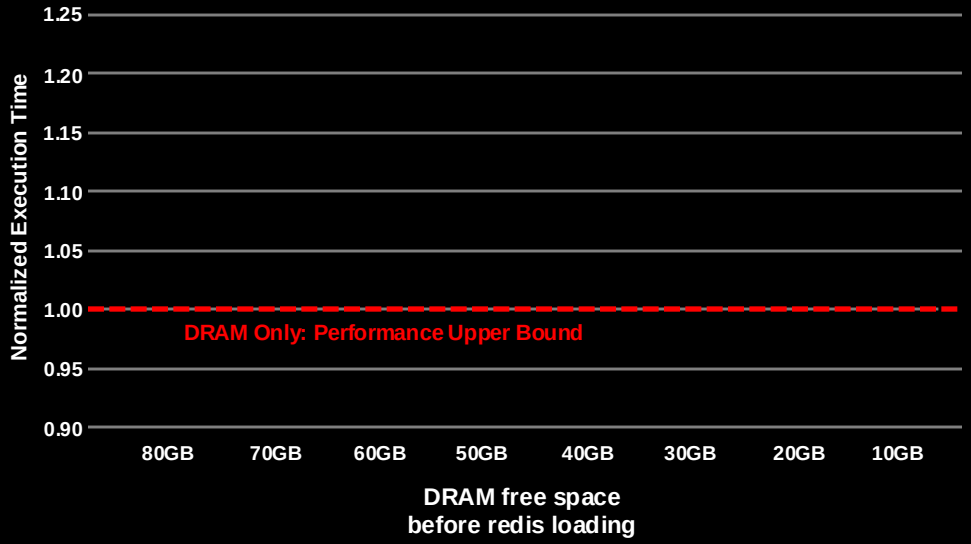


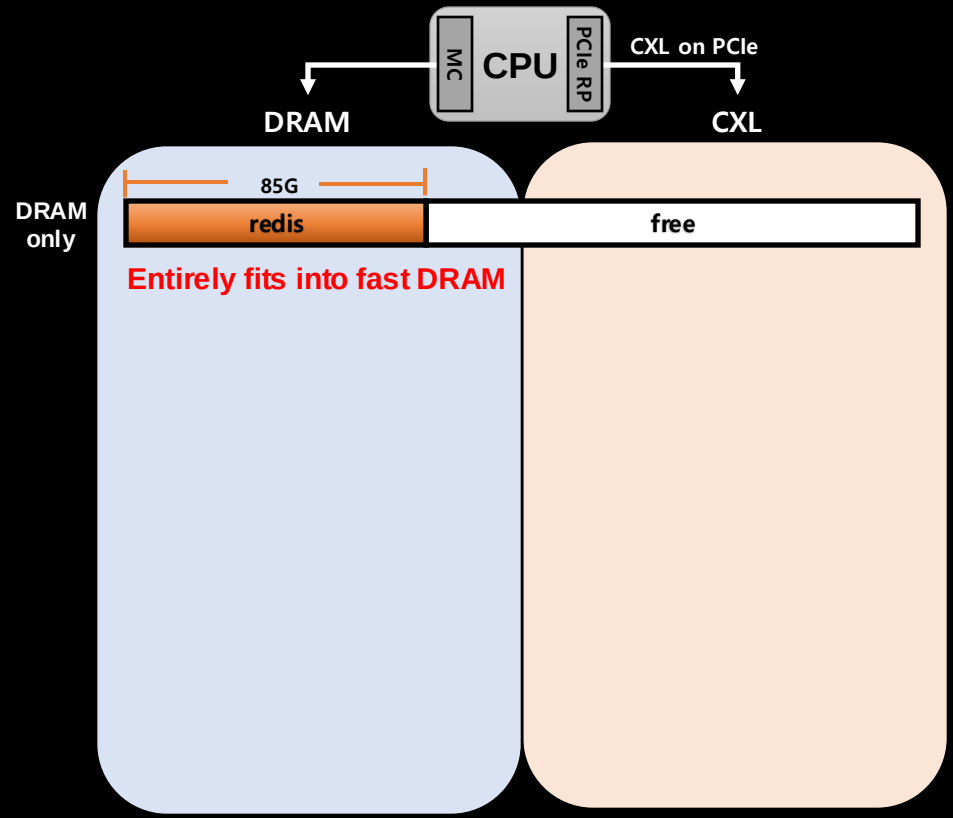


HMSDK Capacity Expansion

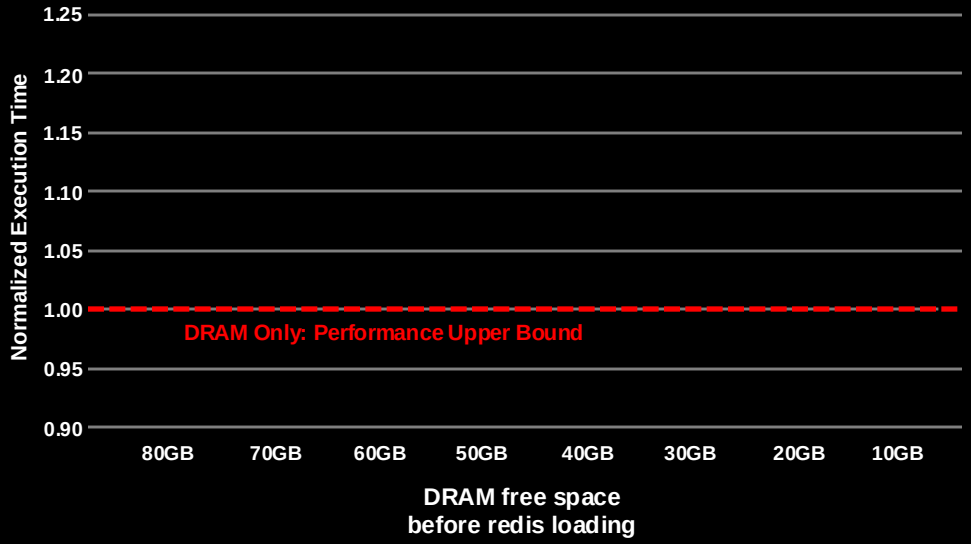


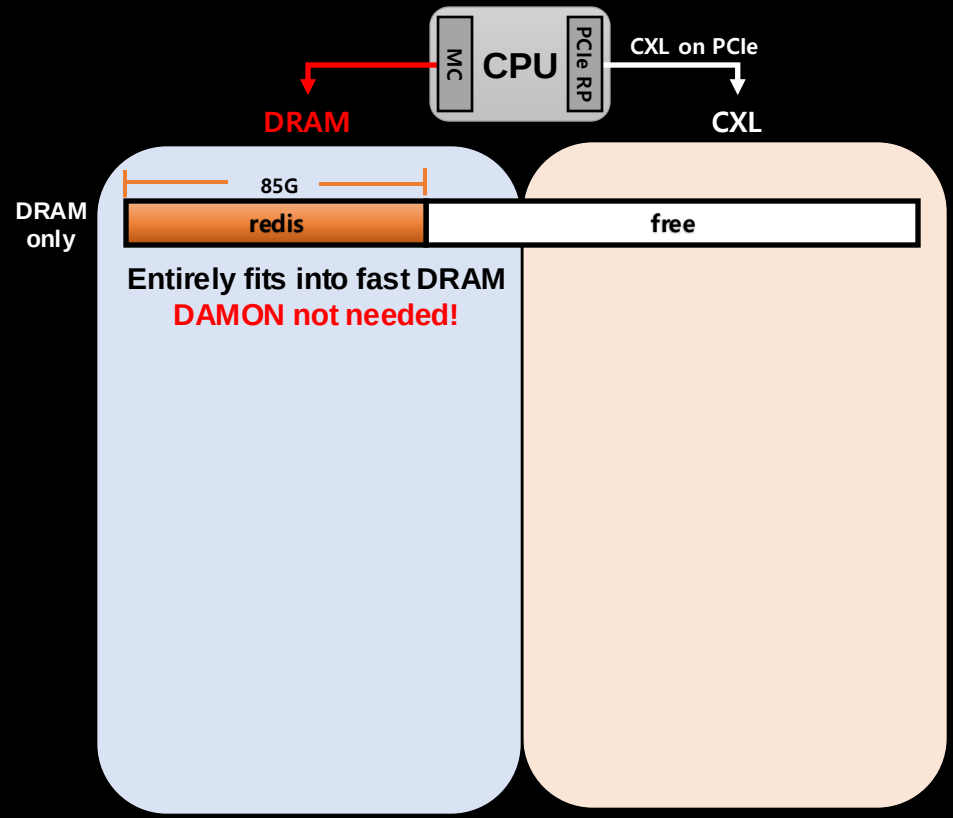
HMSDK Capacity Expansion



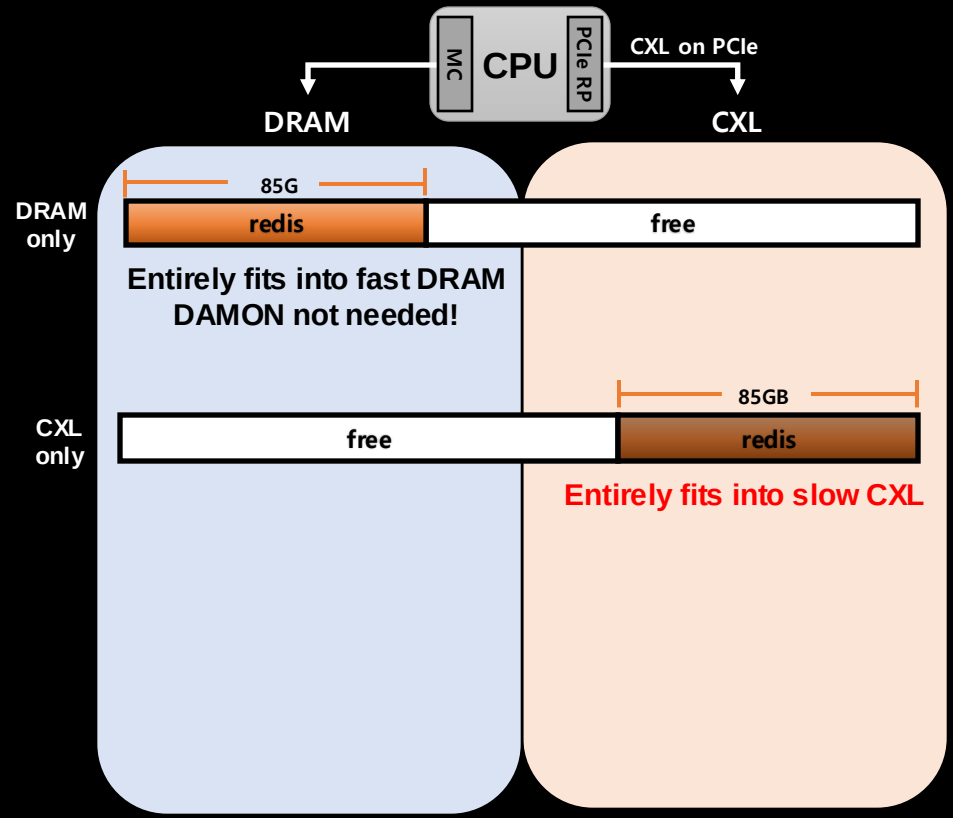


HMSDK Capacity Expansion

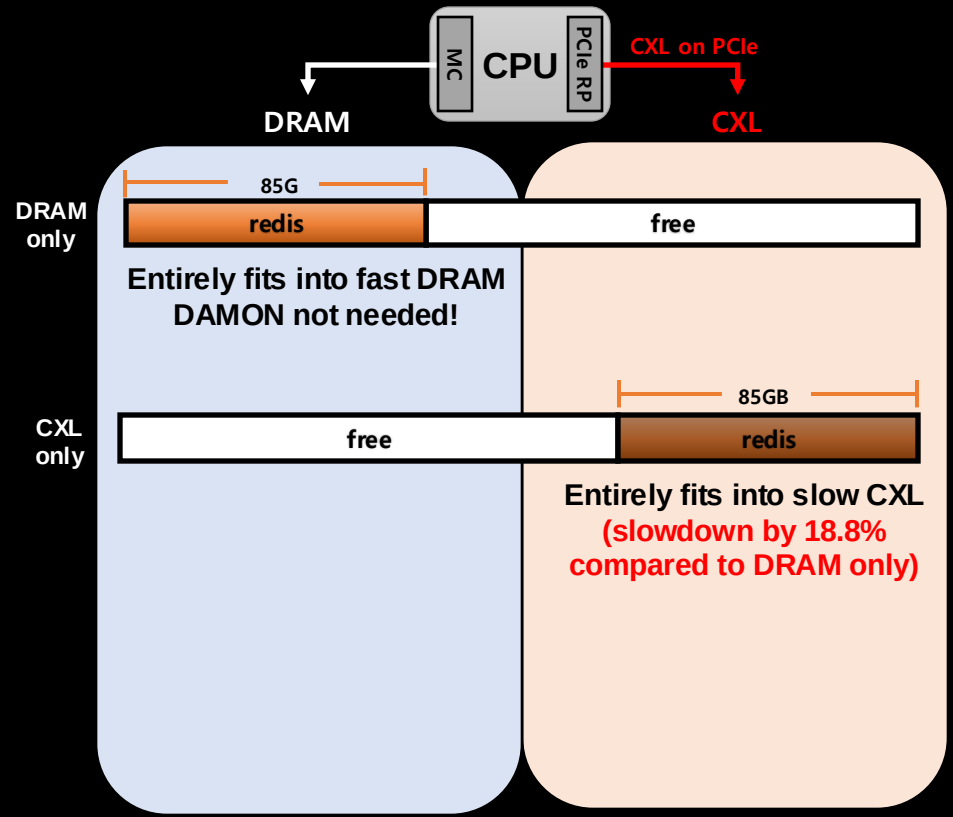




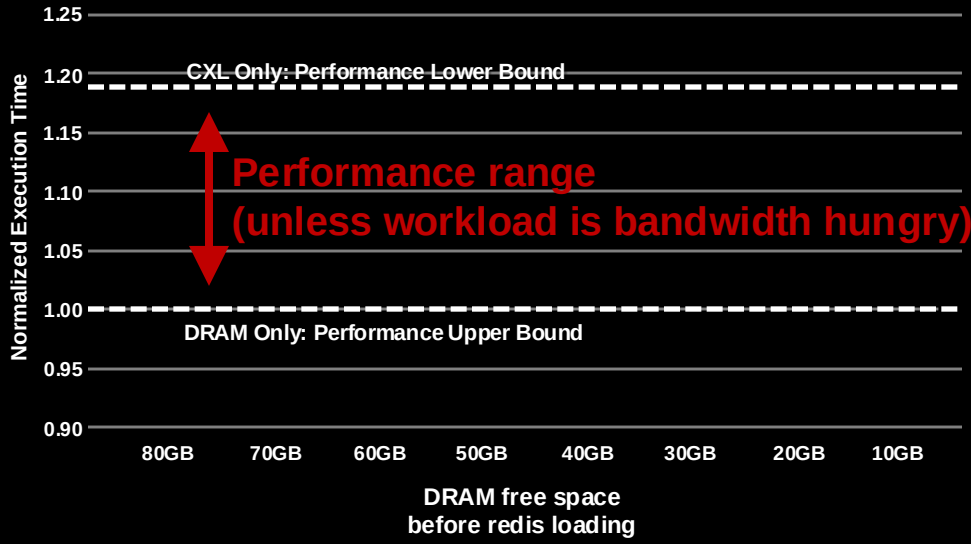
HMSDK Capacity Expansion

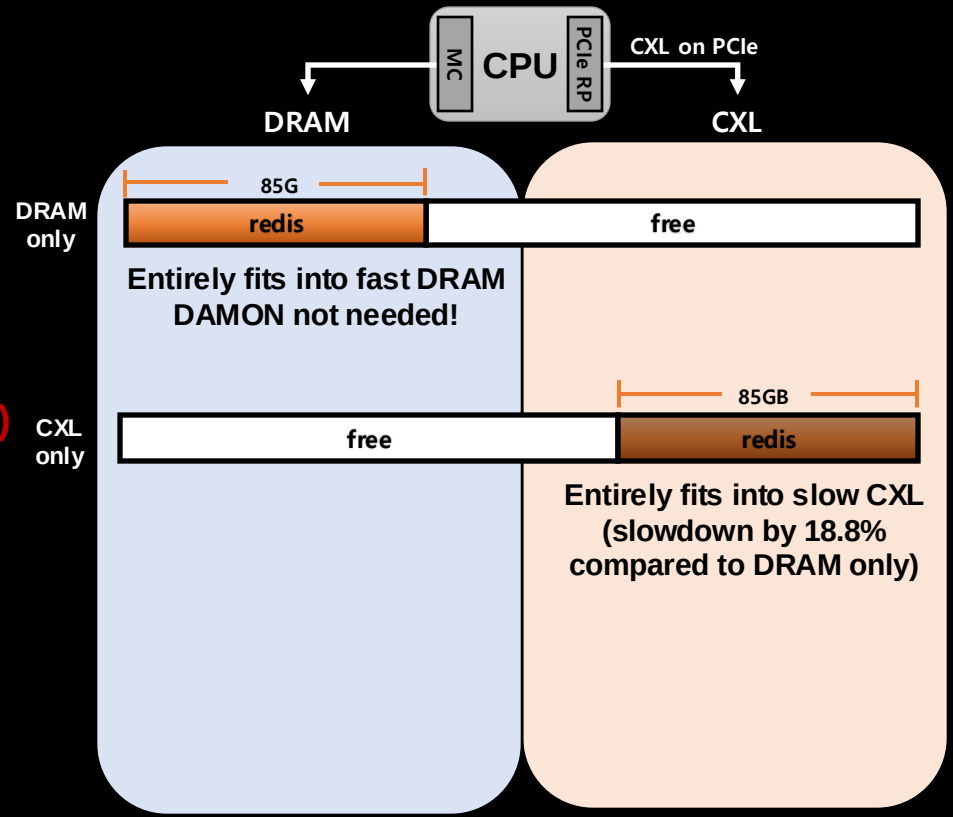


HMSDK Capacity Expansion

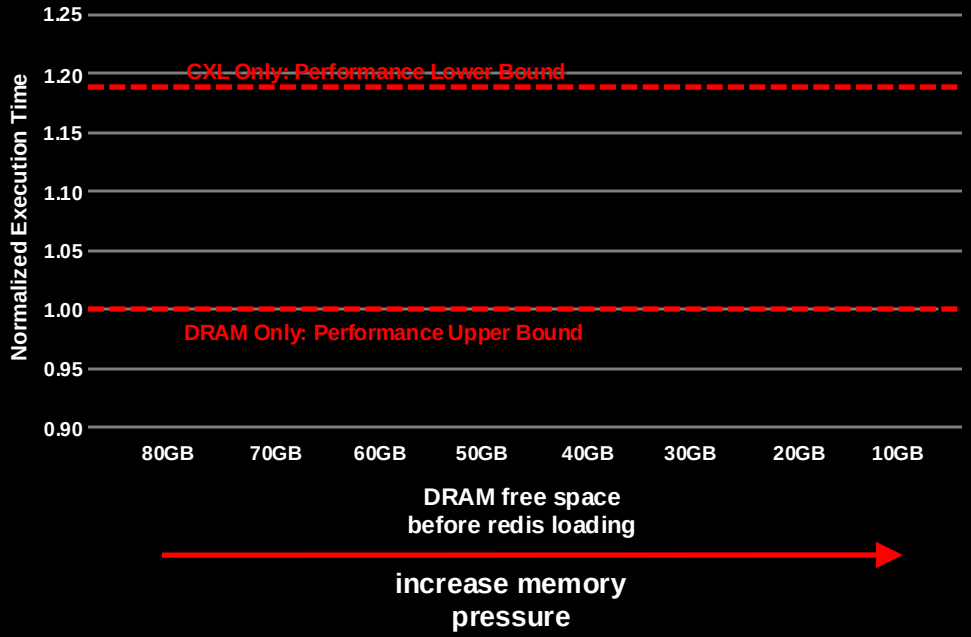


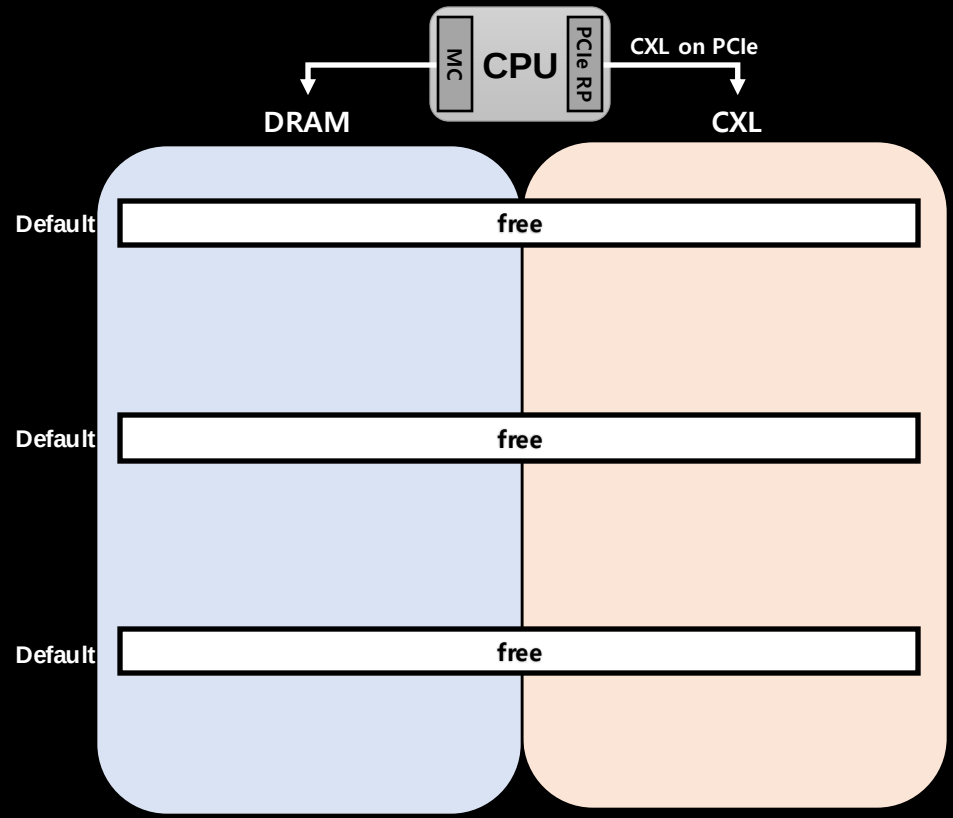
HMSDK Capacity Expansion



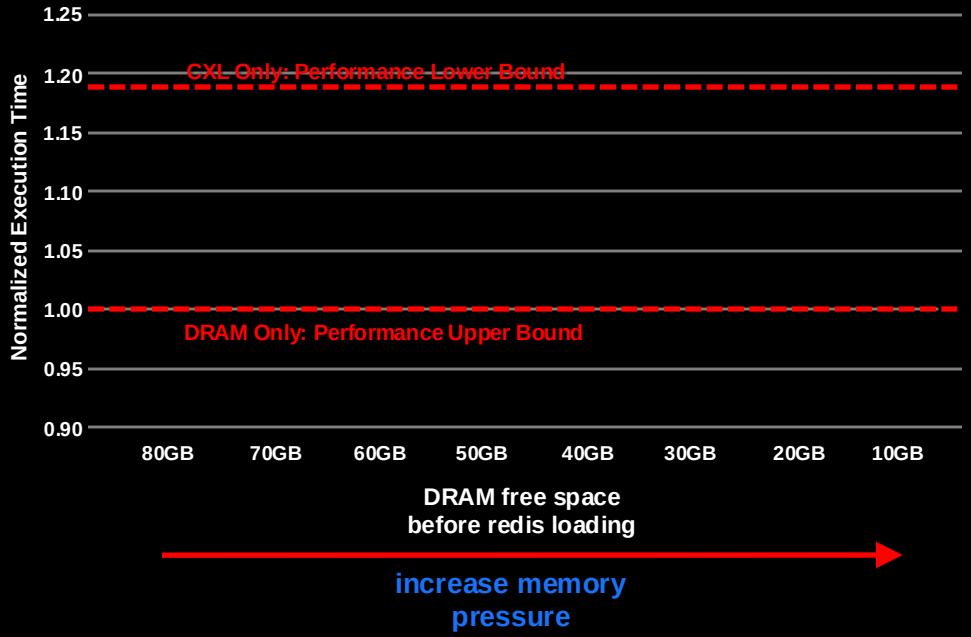


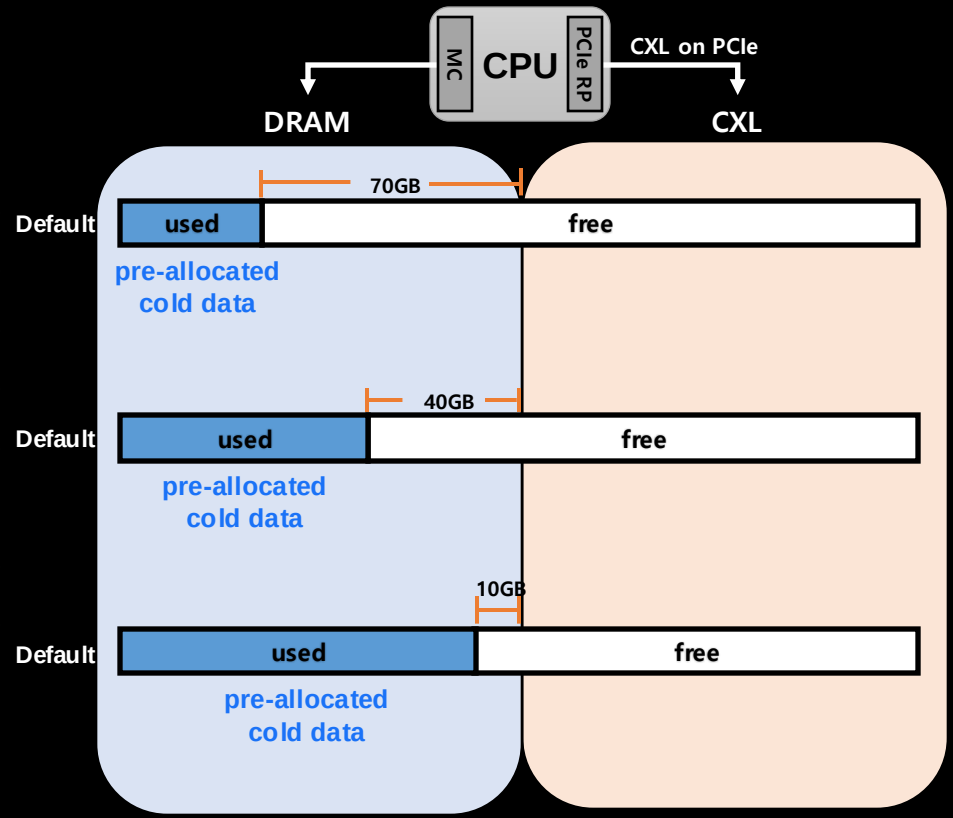
HMSDK Capacity Expansion



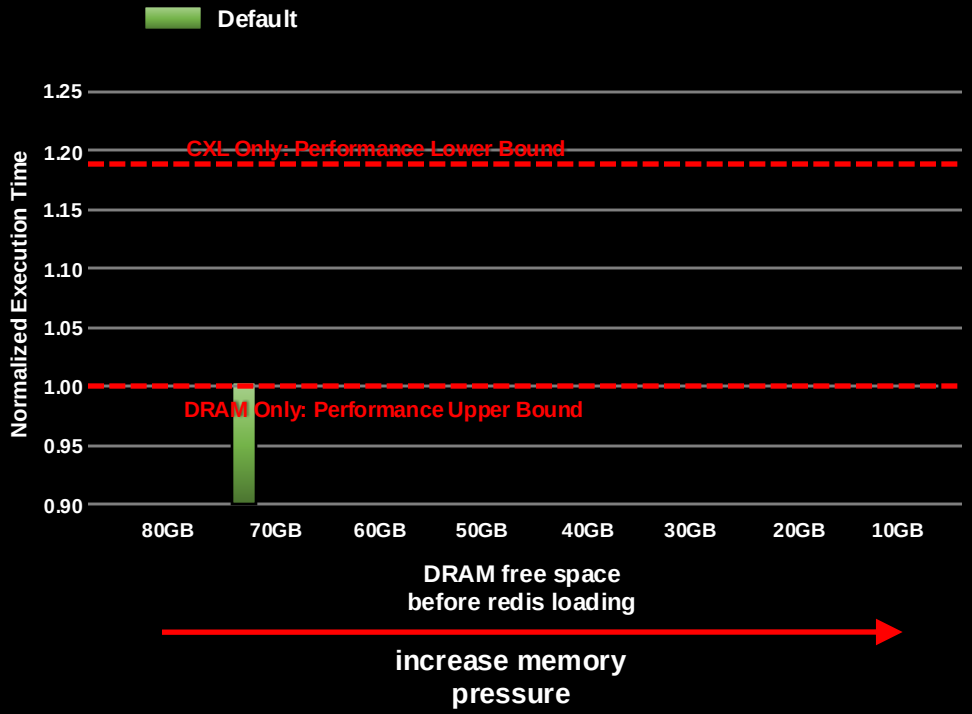


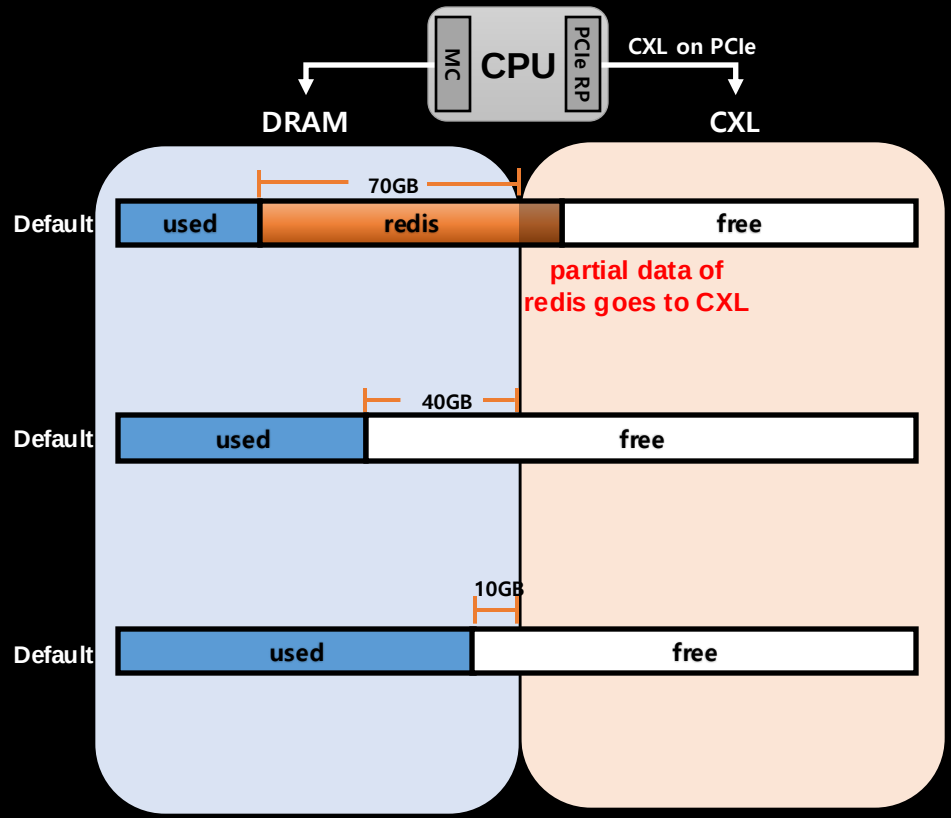
HMSDK Capacity Expansion





HMSDK Capacity Expansion

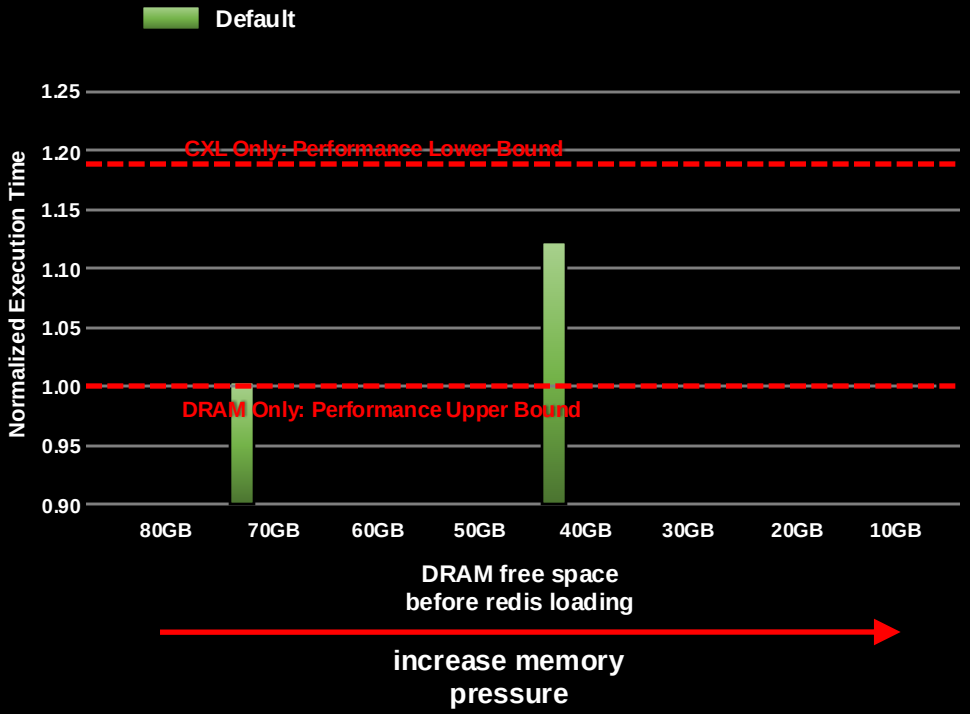


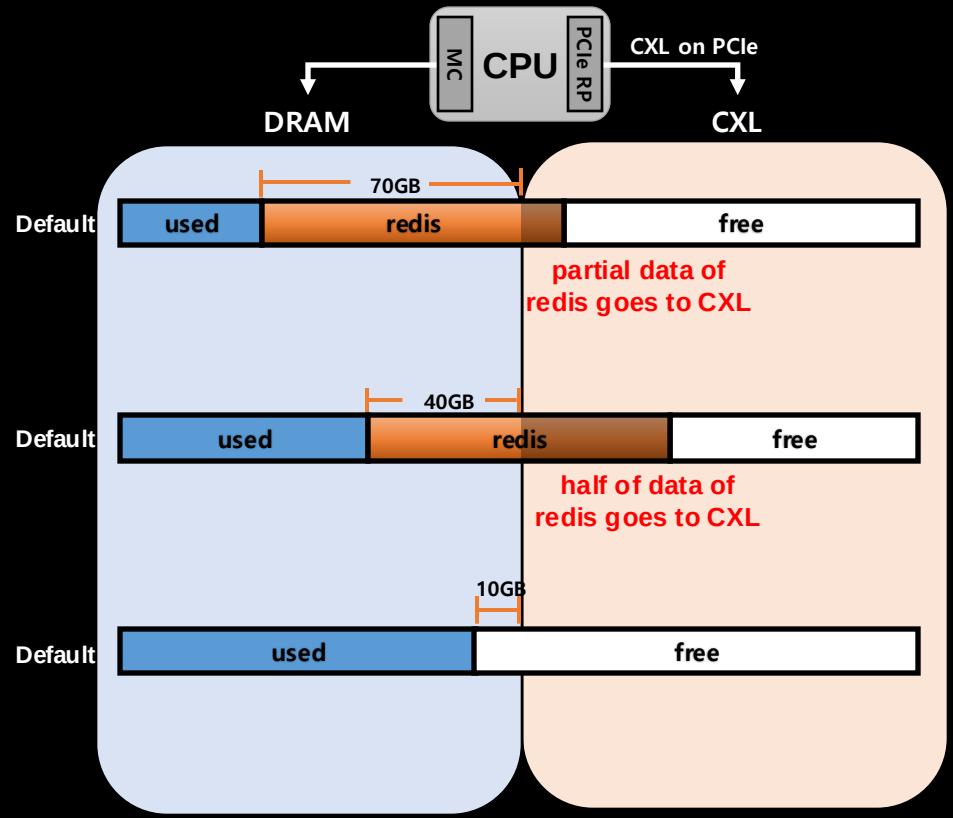


<Default>

1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

HMSDK Capacity Expansion

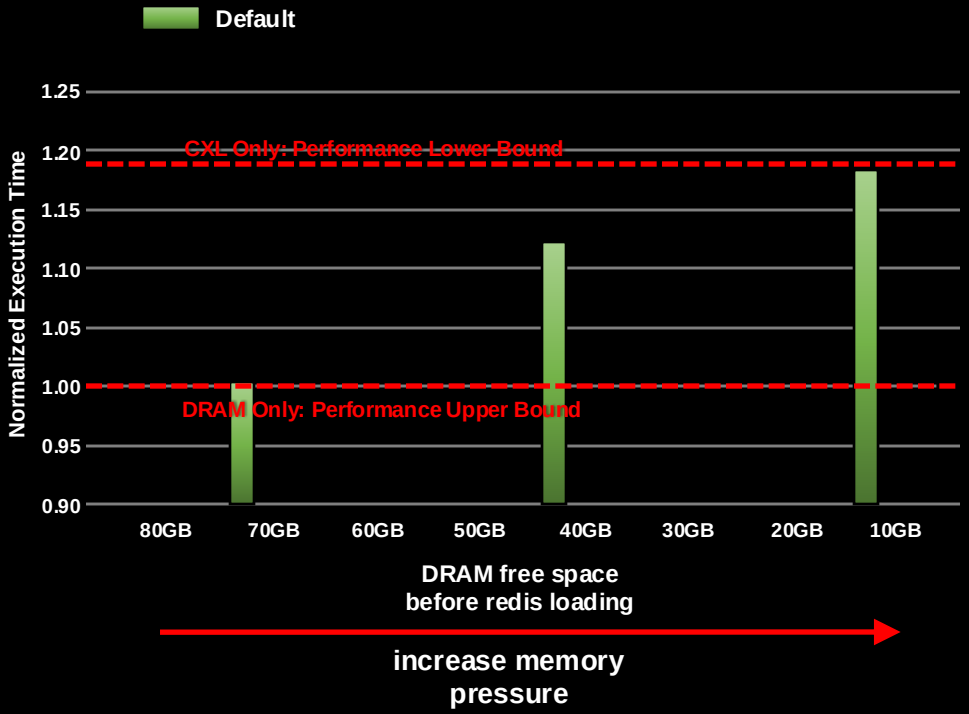


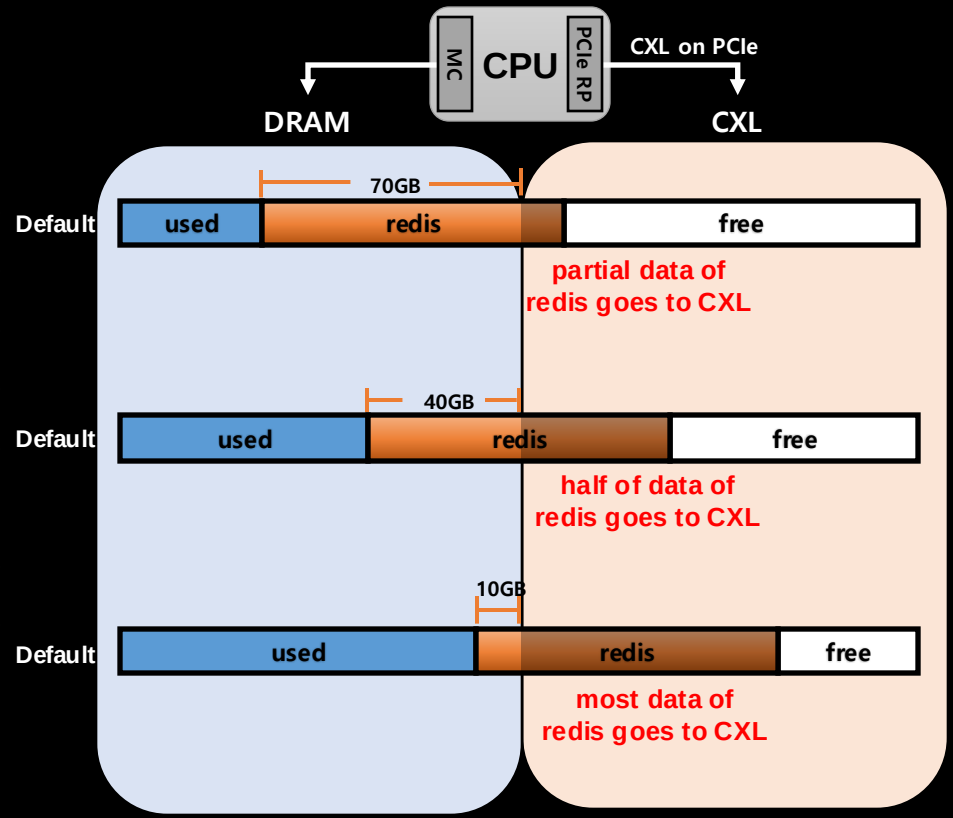


<Default>

1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

HMSDK Capacity Expansion

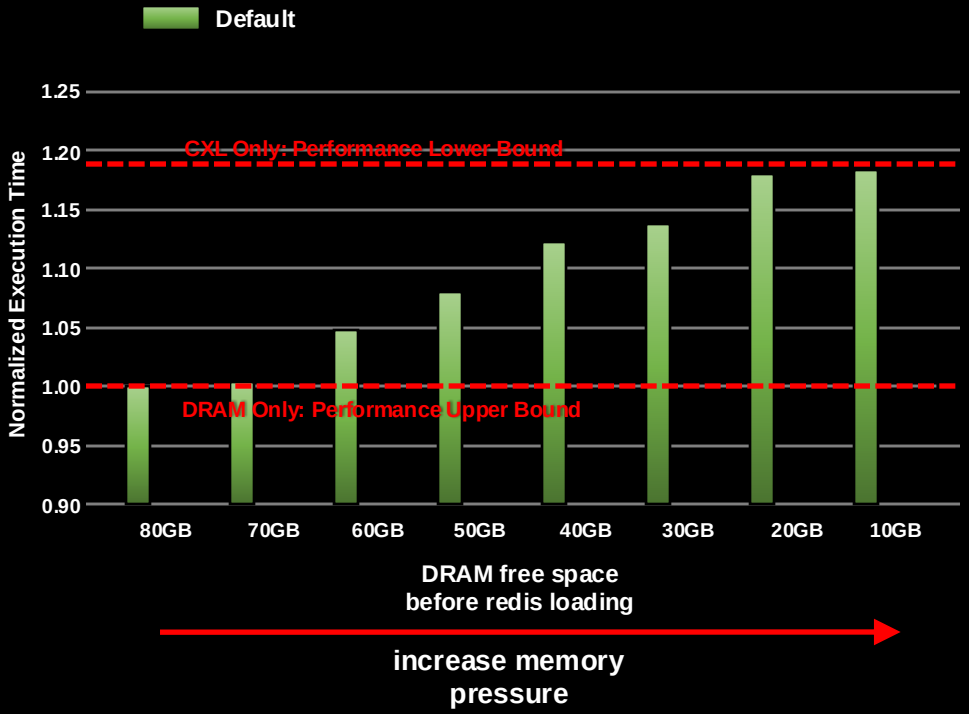


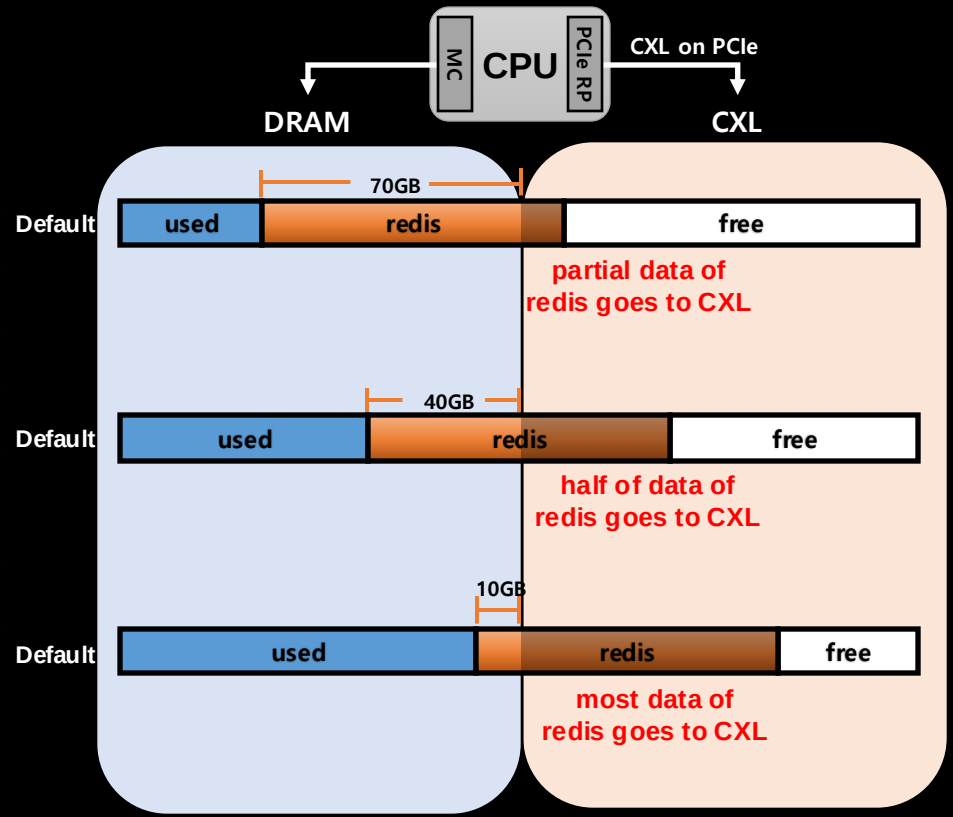


<Default>

1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

HMSDK Capacity Expansion

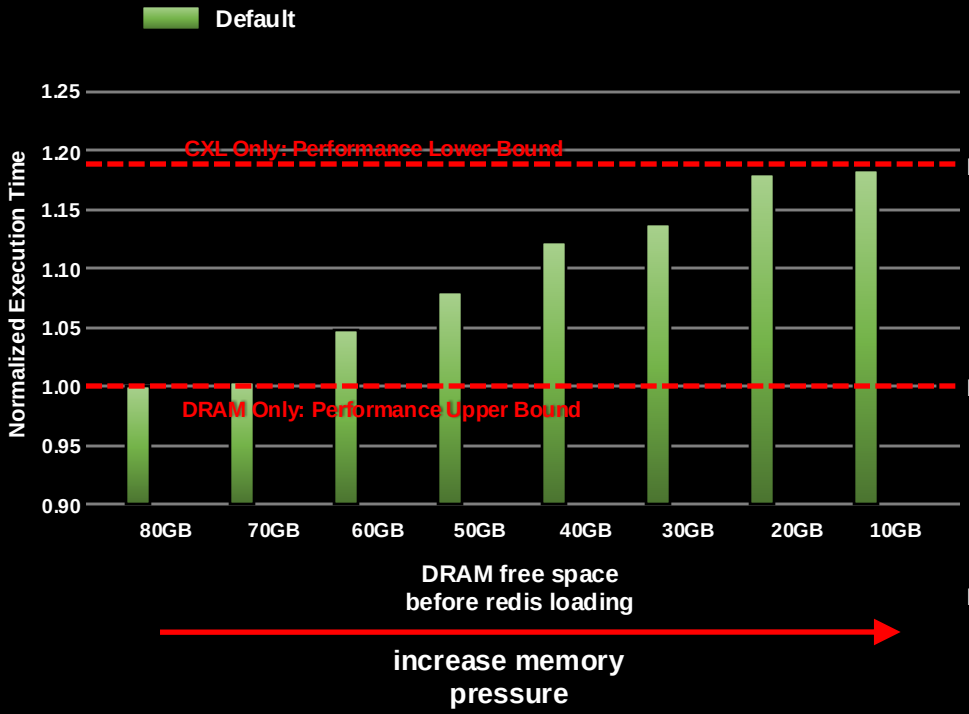


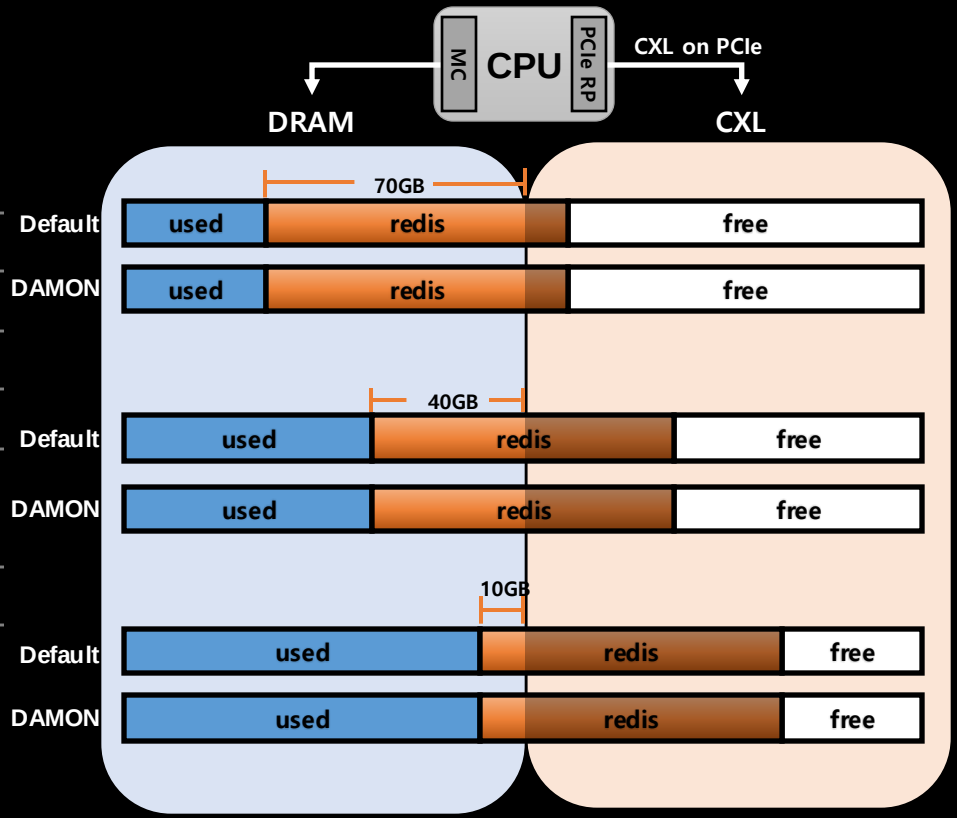


<Default>

1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

HMSDK Capacity Expansion



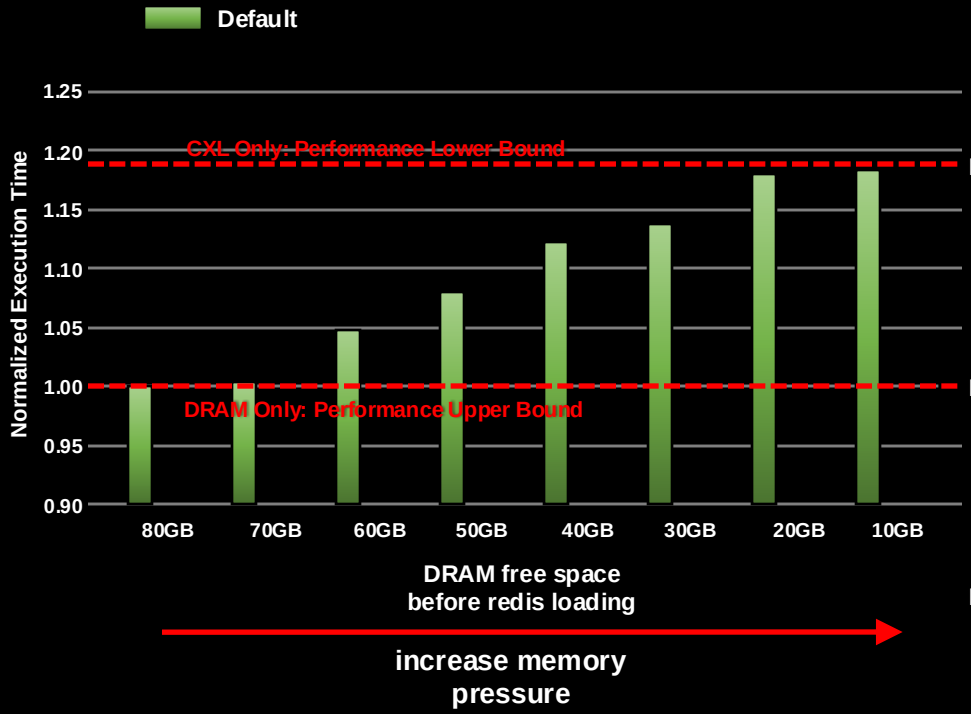


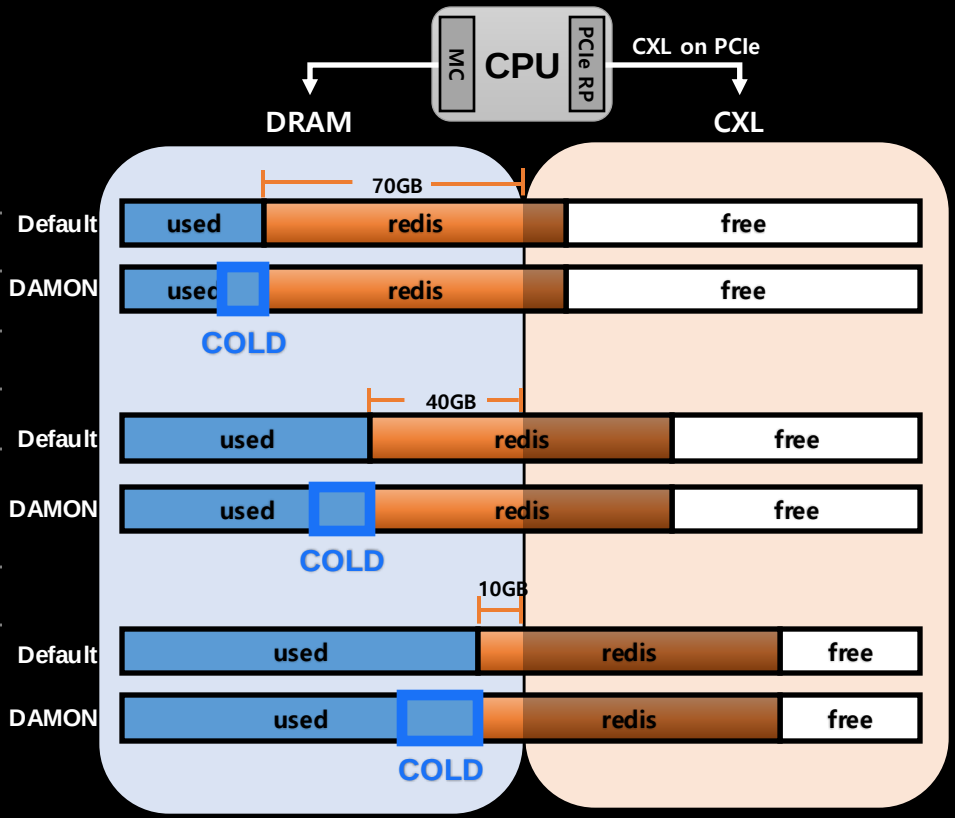
<Default>

1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

HMSDK Capacity Expansion





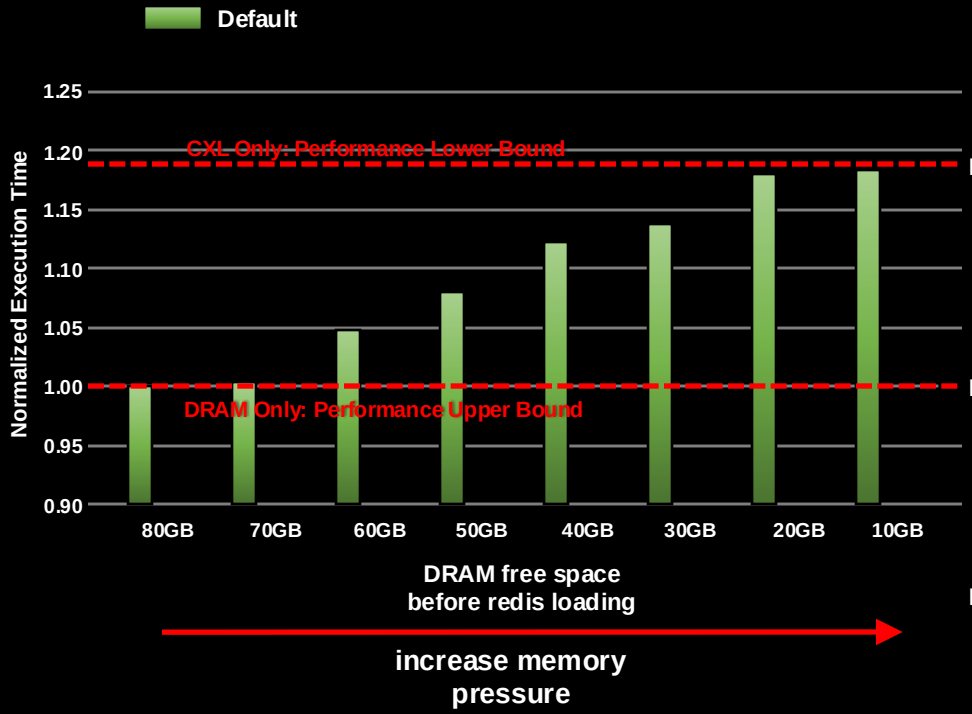
<Default>

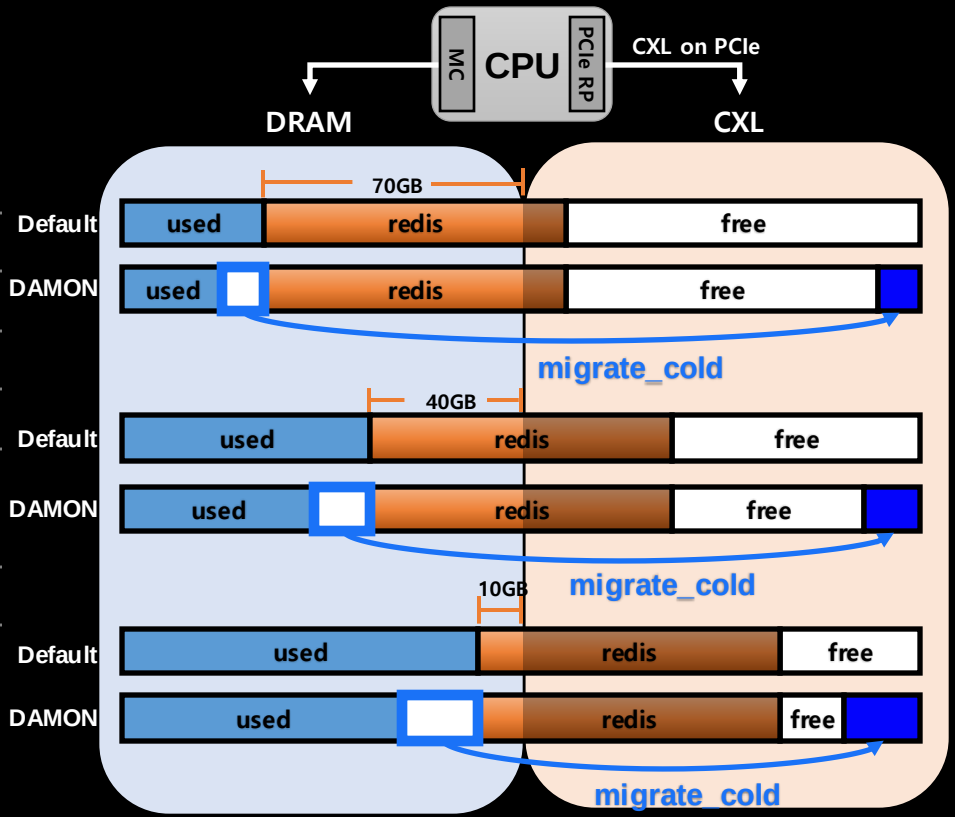
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

1. Demote cold data from DRAM to CXL memory.

HMSDK Capacity Expansion





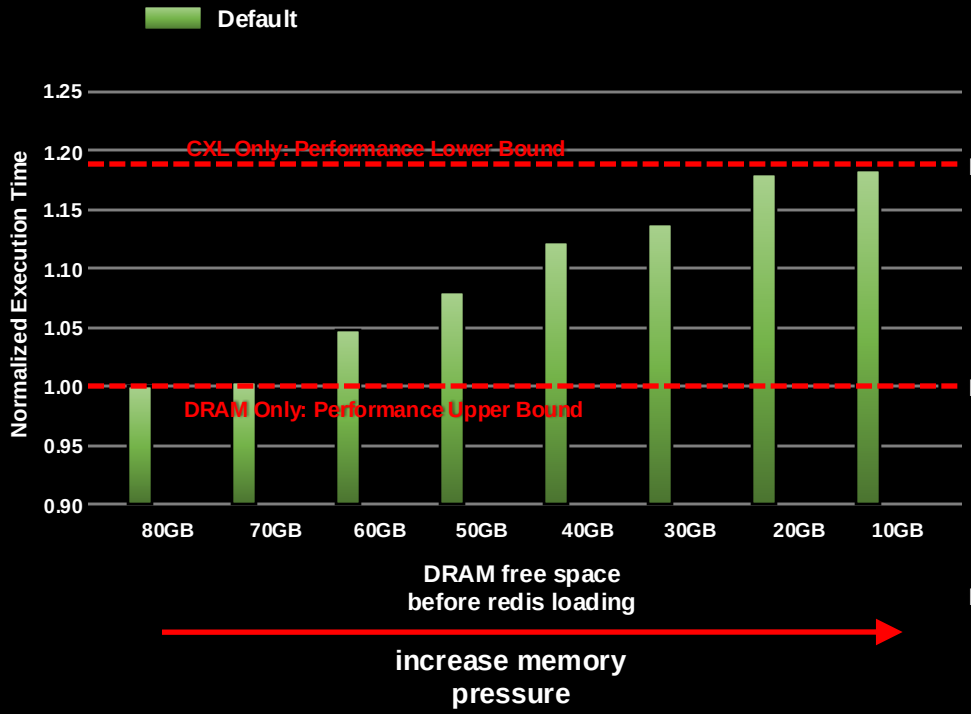
<Default>

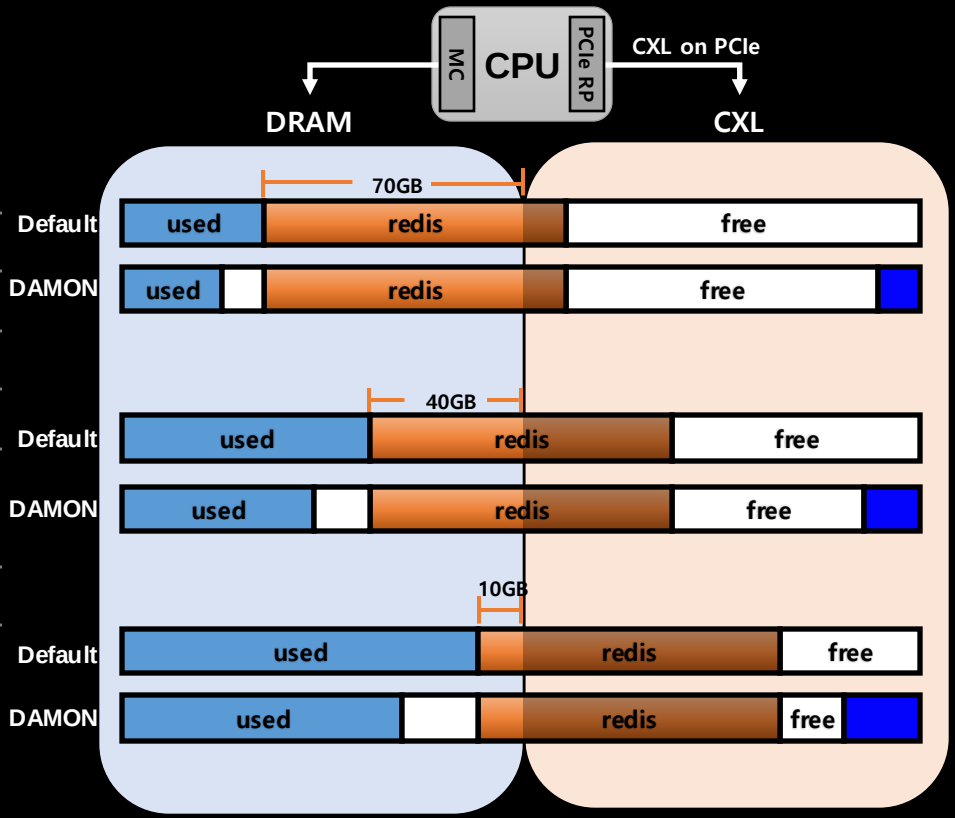
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

1. Demote cold data from DRAM to CXL memory.

HMSDK Capacity Expansion

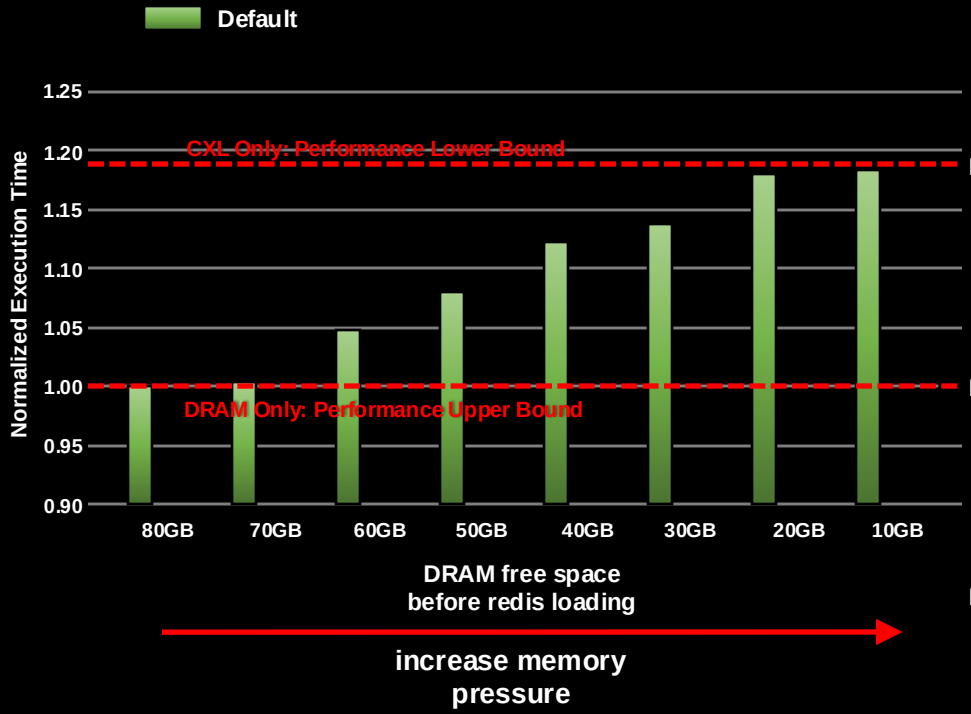


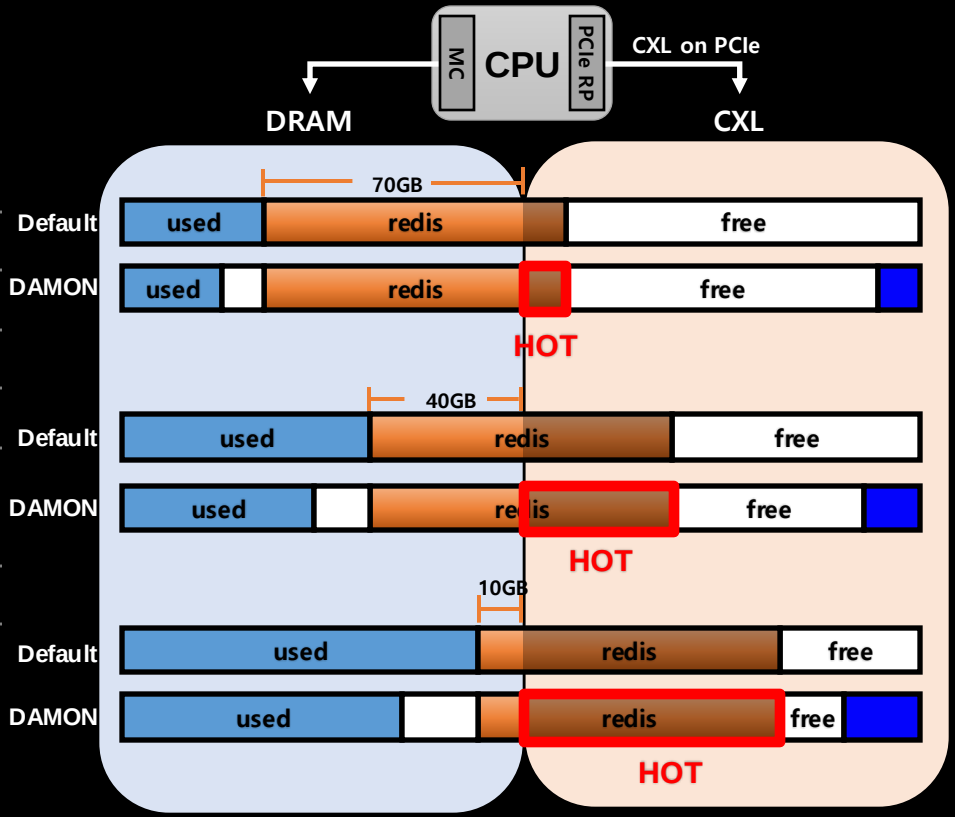


<Default>
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>
1. Demote cold data from DRAM to CXL memory.

HMSDK Capacity Expansion





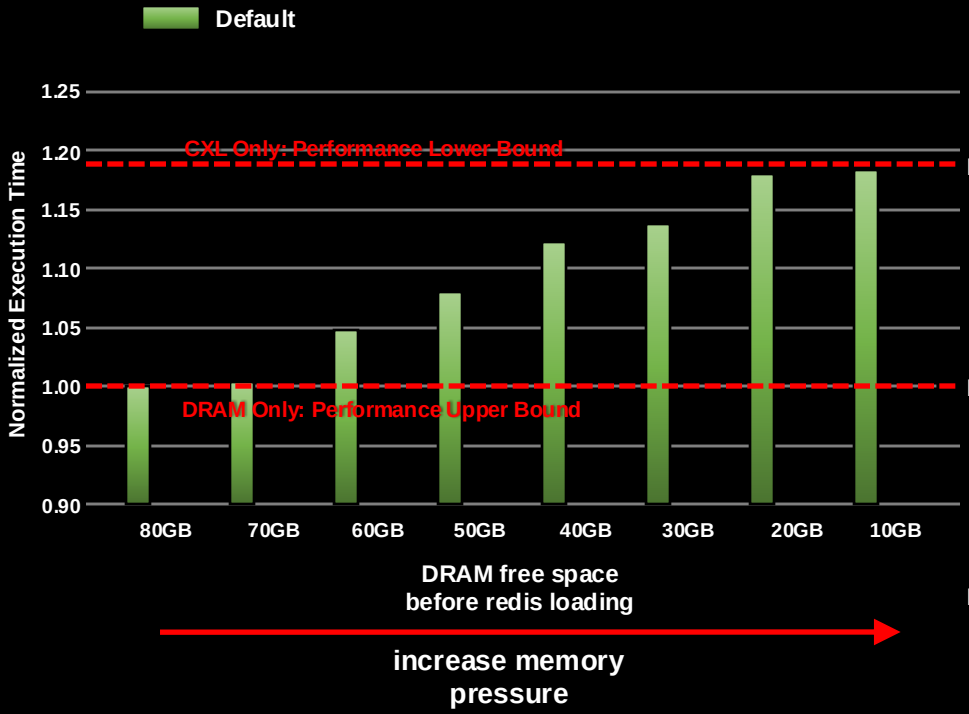
<Default>

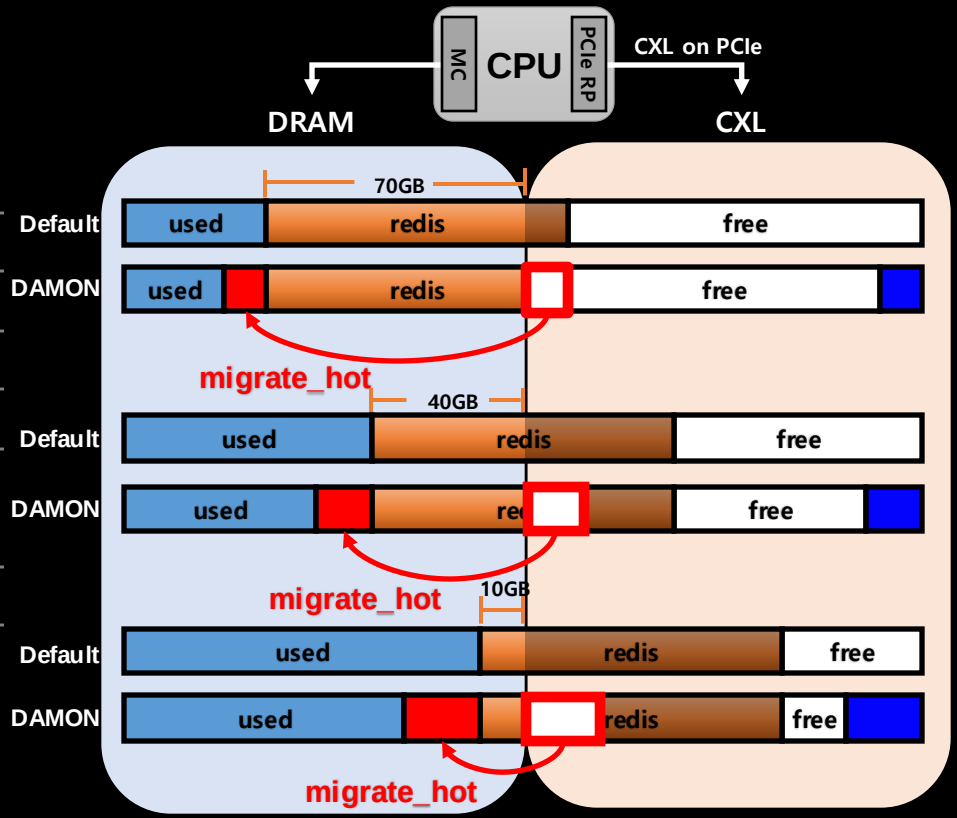
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

1. Demote cold data from DRAM to CXL memory.
2. Promote more redis data to fast DRAM.
   (while keeping cold data on CXL memory)

HMSDK Capacity Expansion

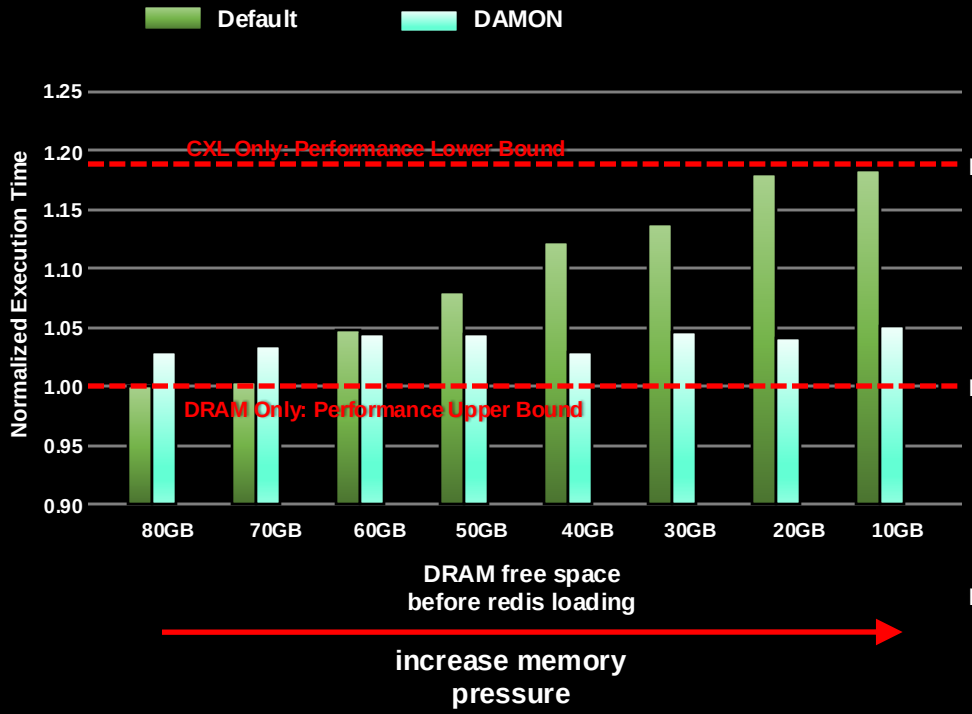


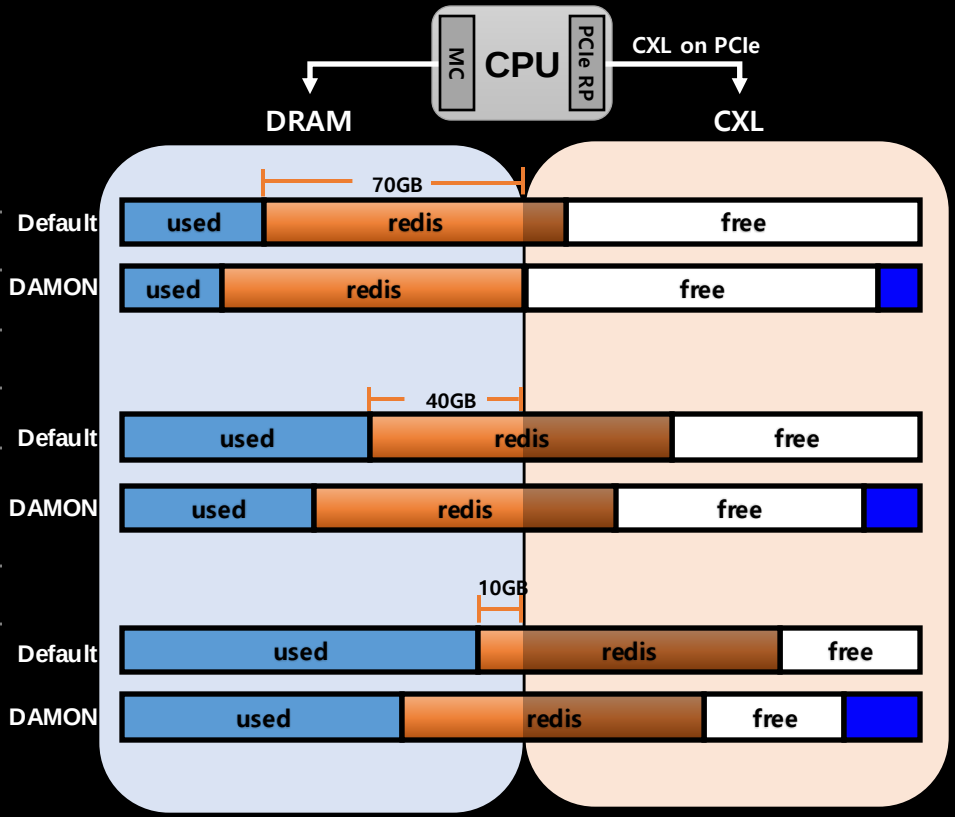


<Default>
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>
1. Demote cold data from DRAM to CXL memory.
2. Promote more redis data to fast DRAM.
   (while keeping cold data on CXL memory)

HMSDK Capacity Expansion





<Default>

1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

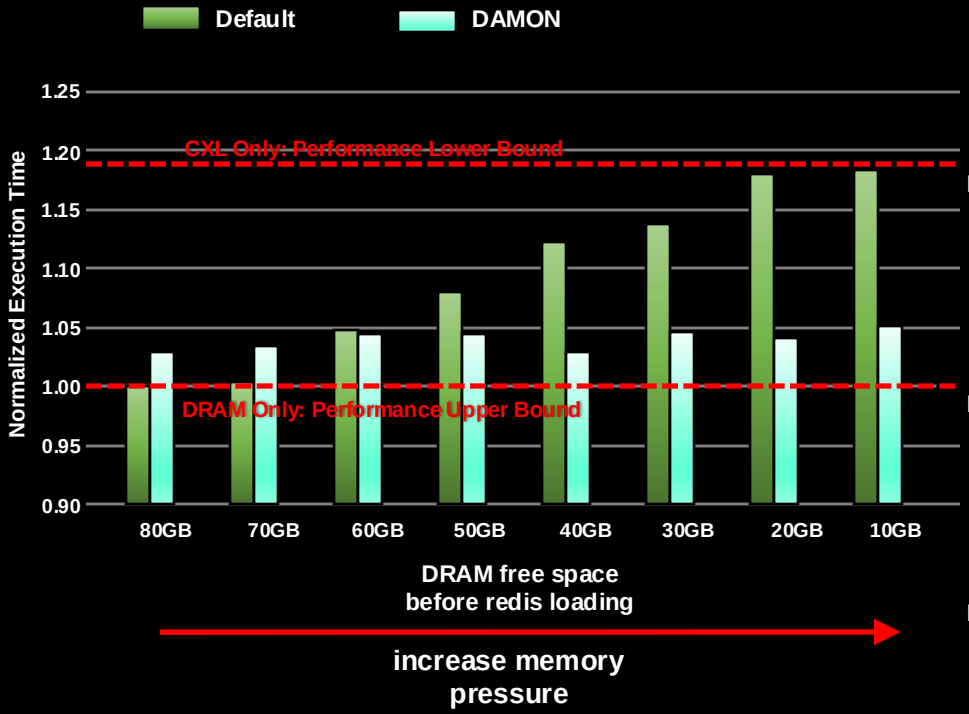
1. Demote cold data from DRAM to CXL memory.
2. Promote more redis data to fast DRAM.
   (while keeping cold data on CXL memory)

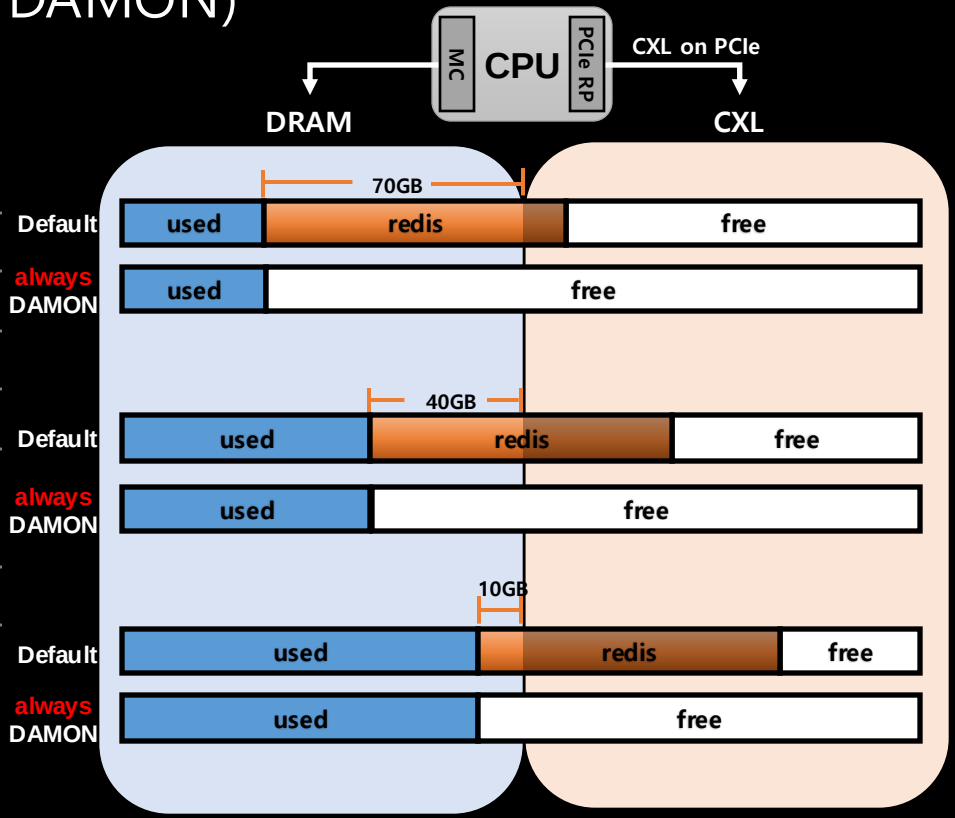
HMSDK Capacity Expansion Evaluation

- The DAMON was turned on just before the memory accesses via YCSB.
  - This is more for evaluation.

- In the real world, the DAMON will be running always on the system.
  - So other cold data can be demoted earlier.
  - Even before the workload(redis) was loaded.

HMSDK Capacity Expansion (always DAMON)





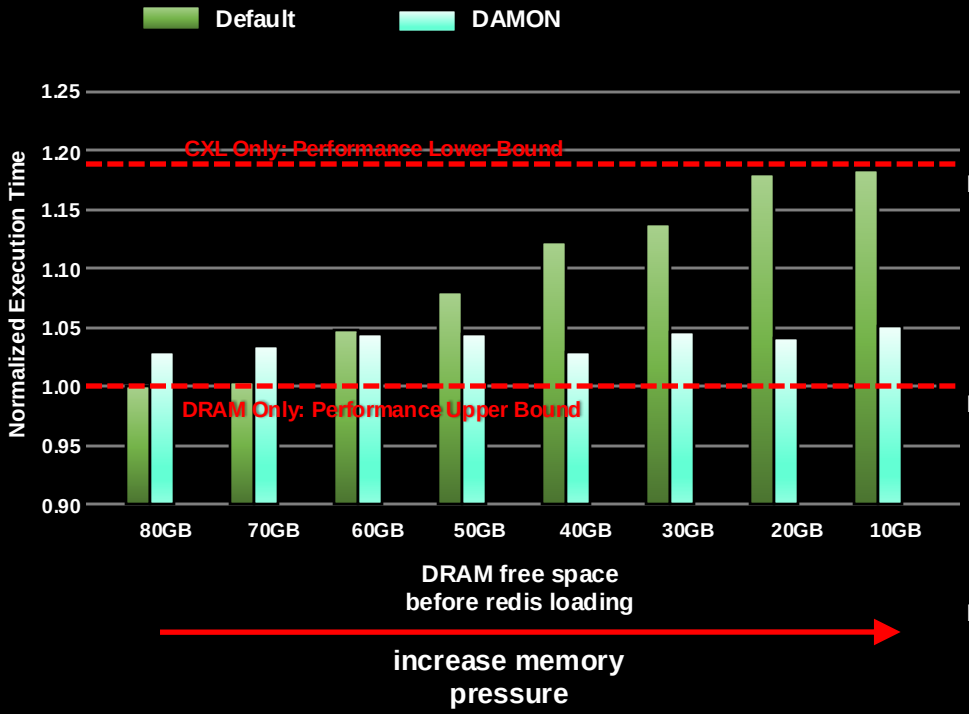
<Default>

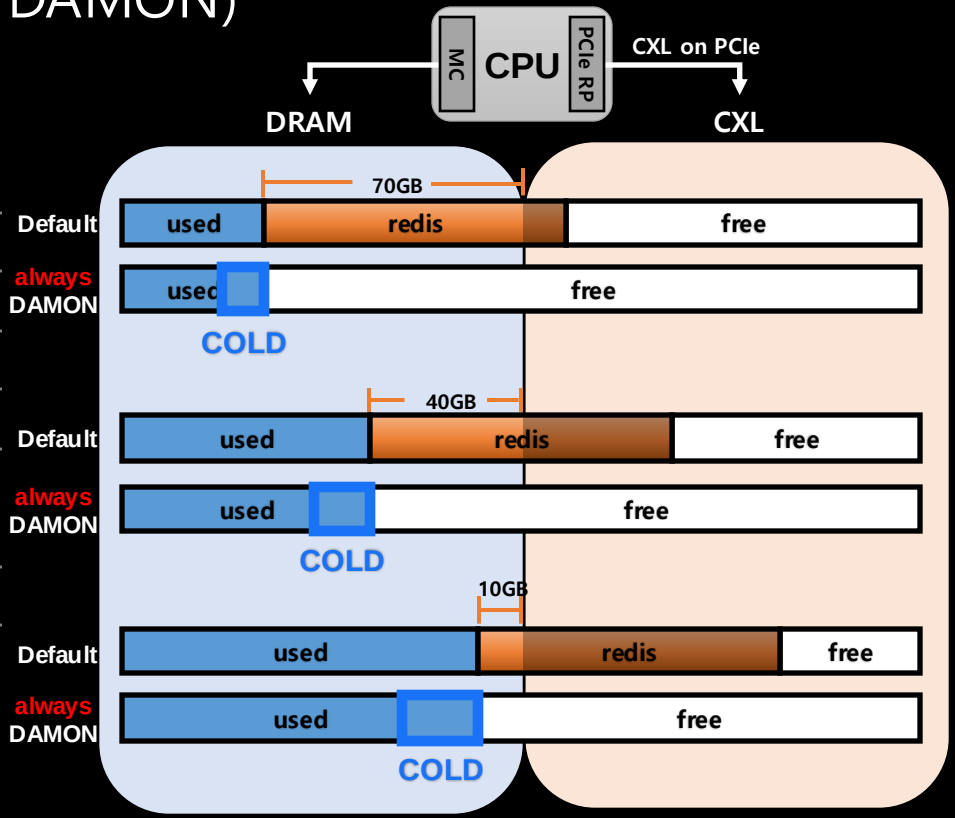
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

1. Demote cold data from DRAM to CXL memory.

HMSDK Capacity Expansion (always DAMON)





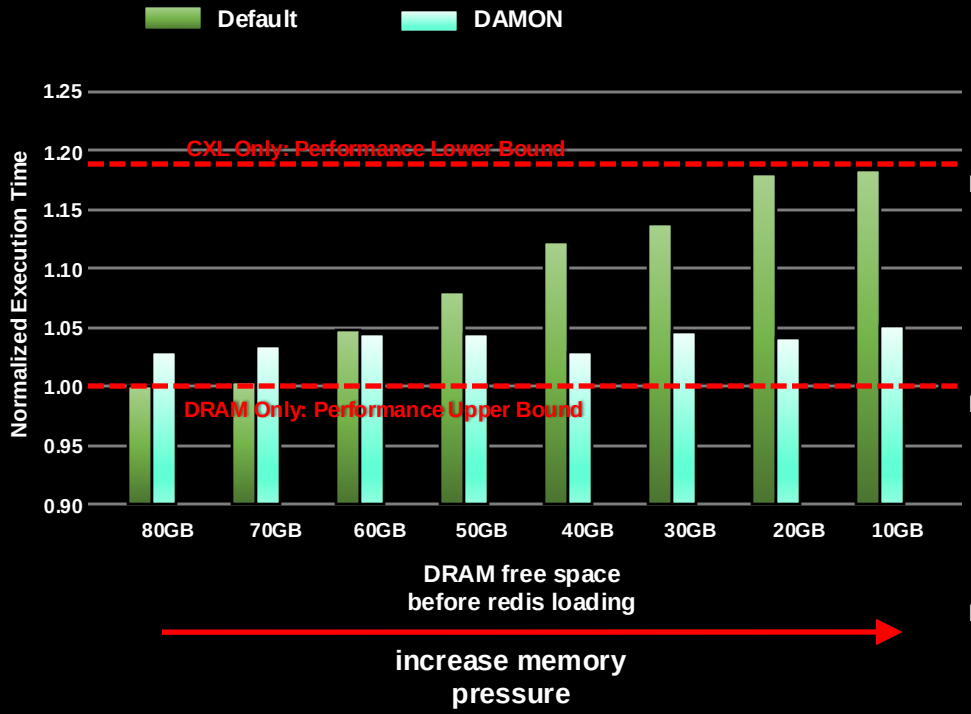
<Default>

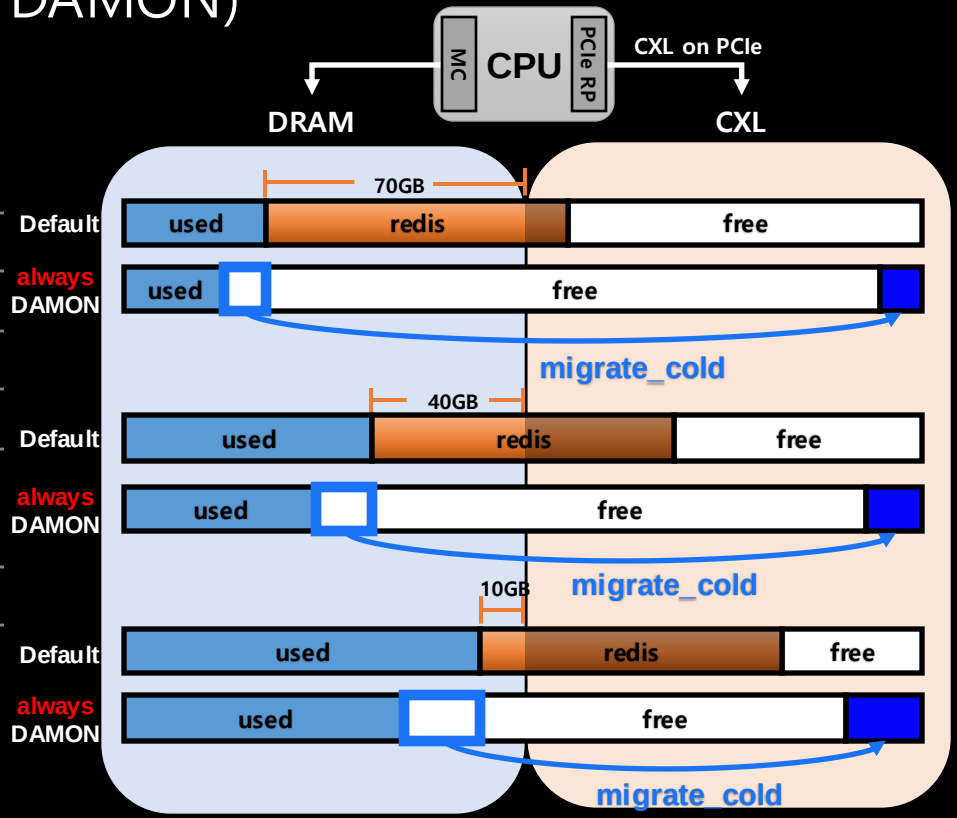
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

1. Demote cold data from DRAM to CXL memory.

HMSDK Capacity Expansion (always DAMON)

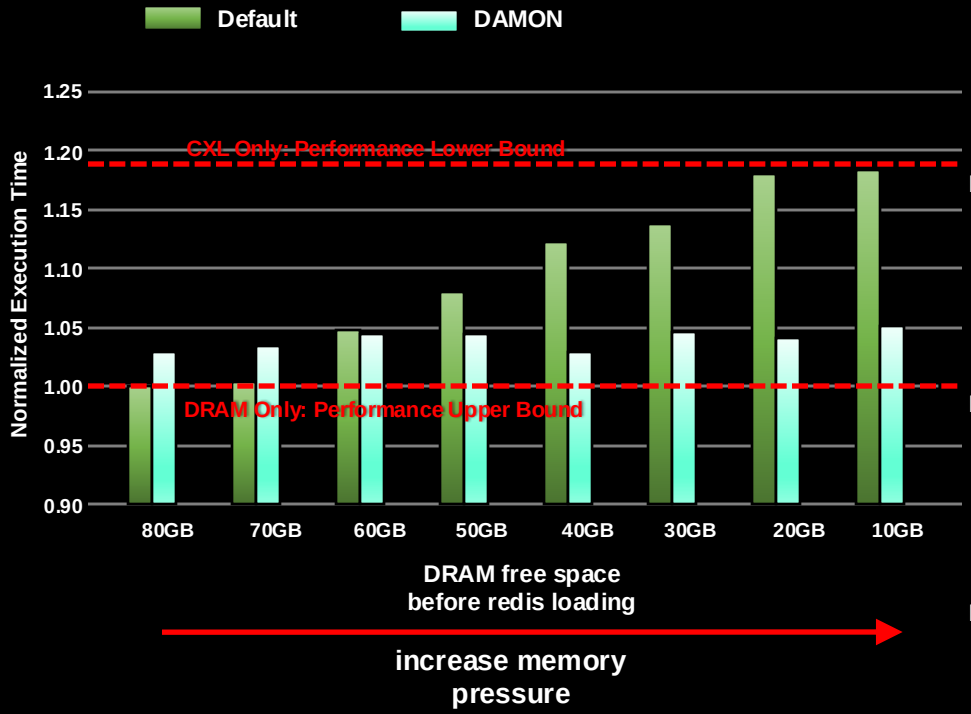


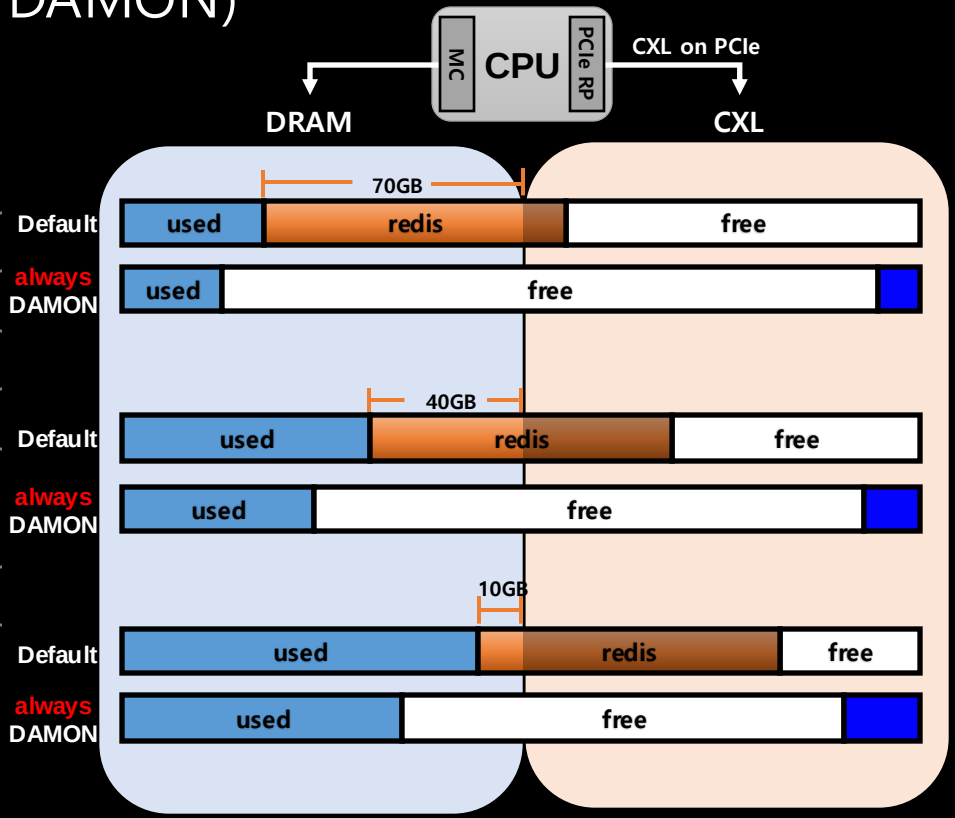


<Default>
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>
1. Demote cold data from DRAM to CXL memory.

HMSDK Capacity Expansion (always DAMON)





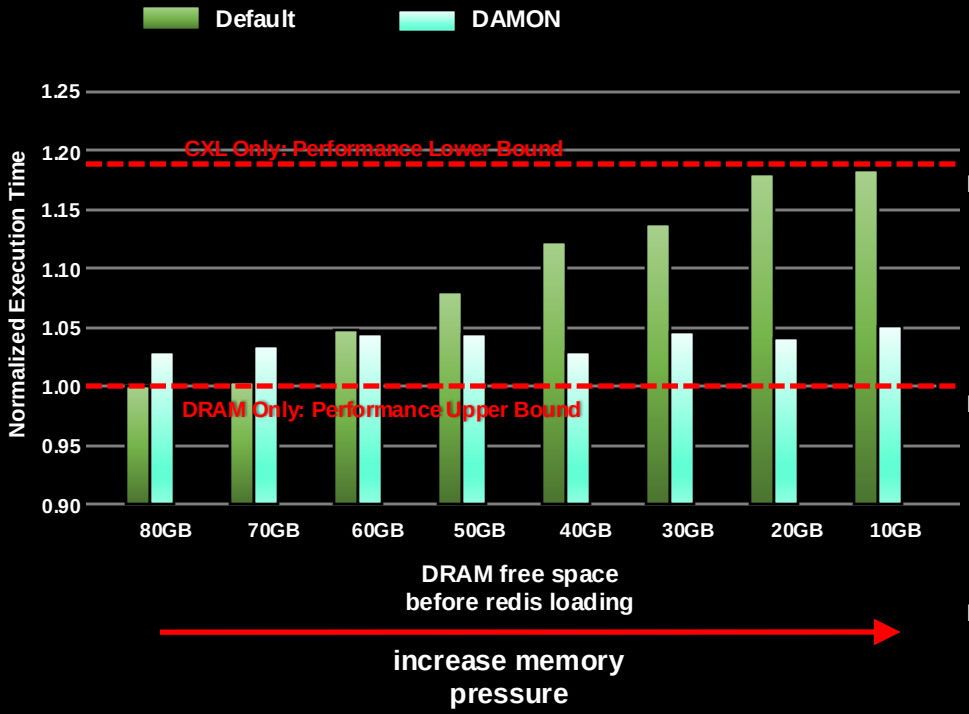
<Default>

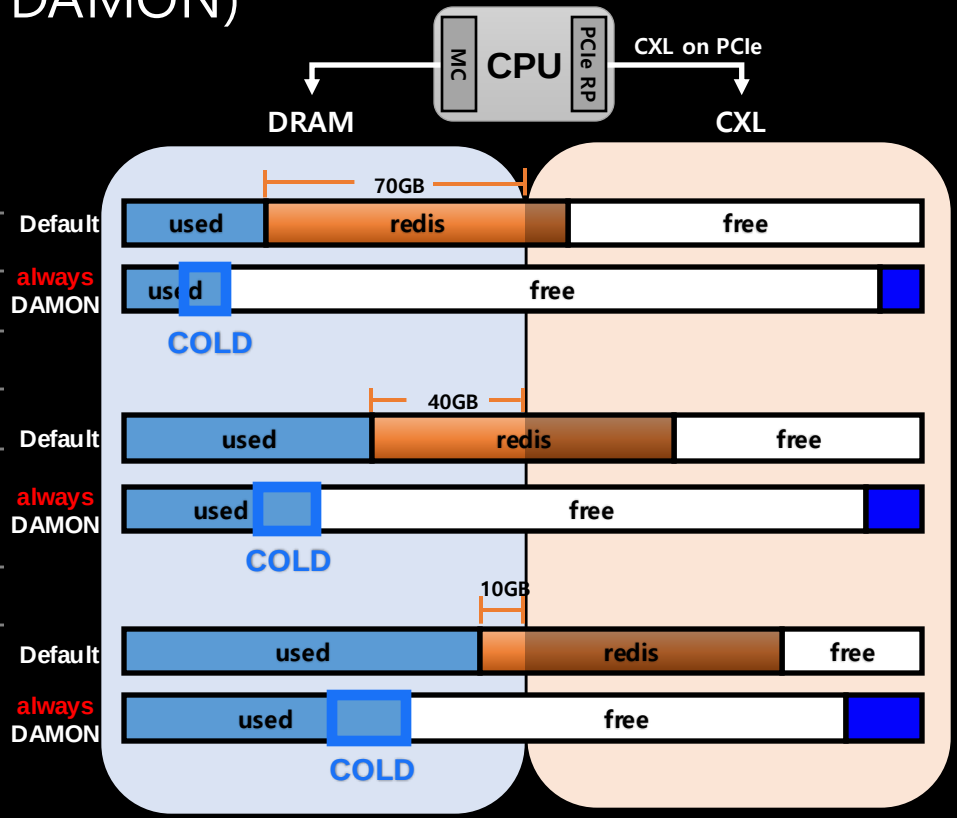
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

1. Demote cold data from DRAM to CXL memory.

HMSDK Capacity Expansion (always DAMON)





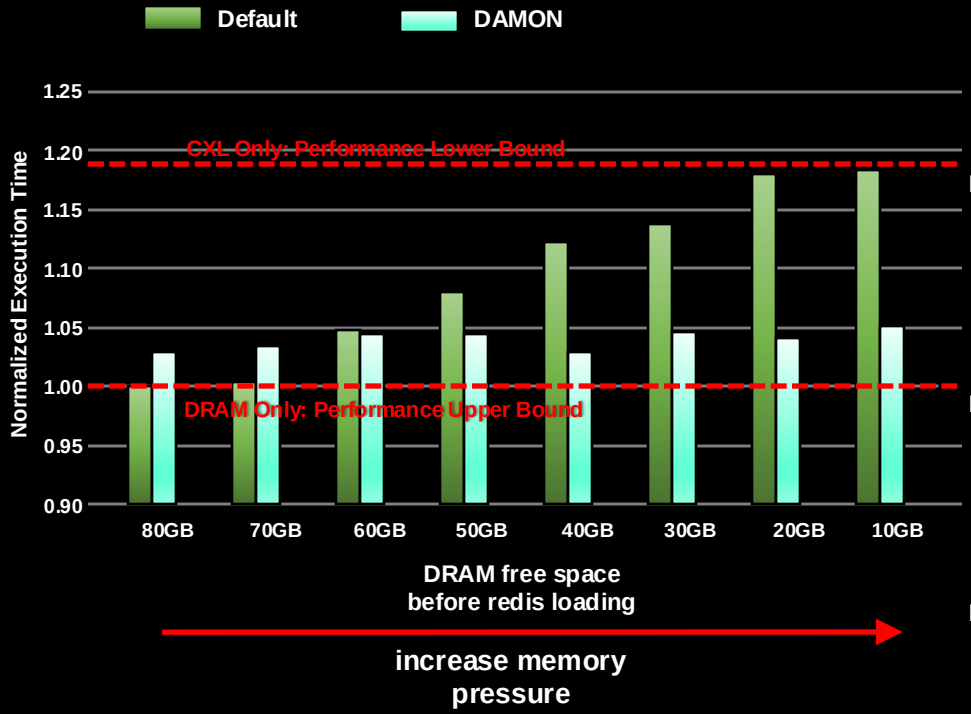
<Default>

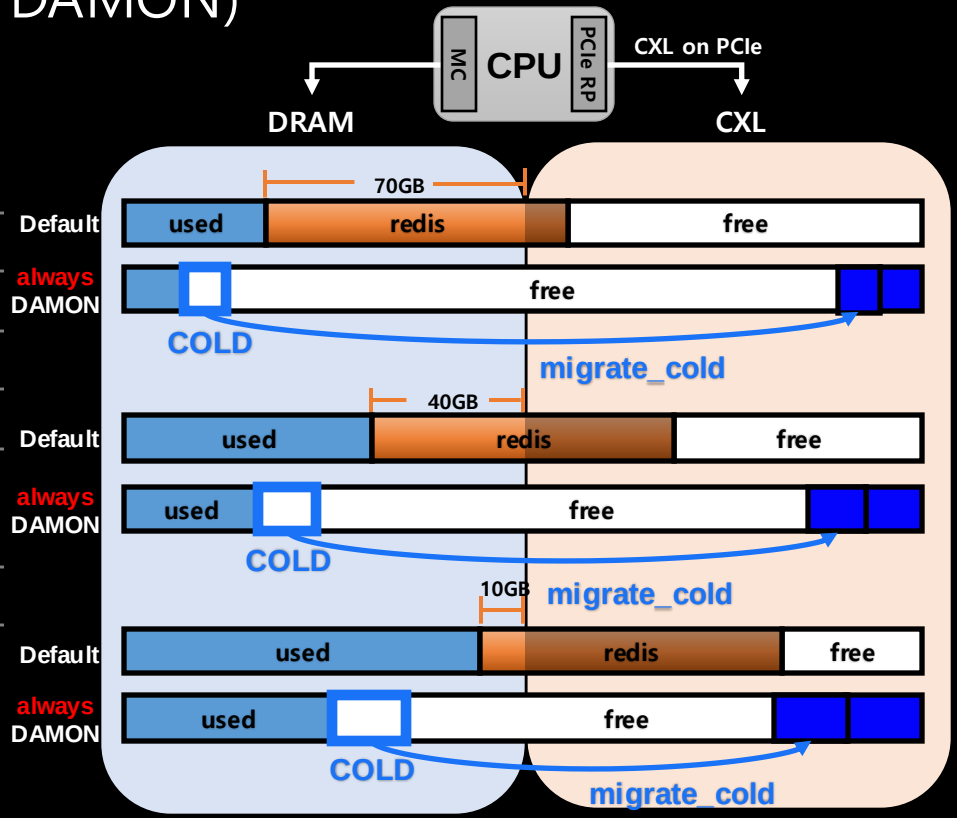
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

1. Demote cold data from DRAM to CXL memory.
2. Keep demote cold data

HMSDK Capacity Expansion (always DAMON)

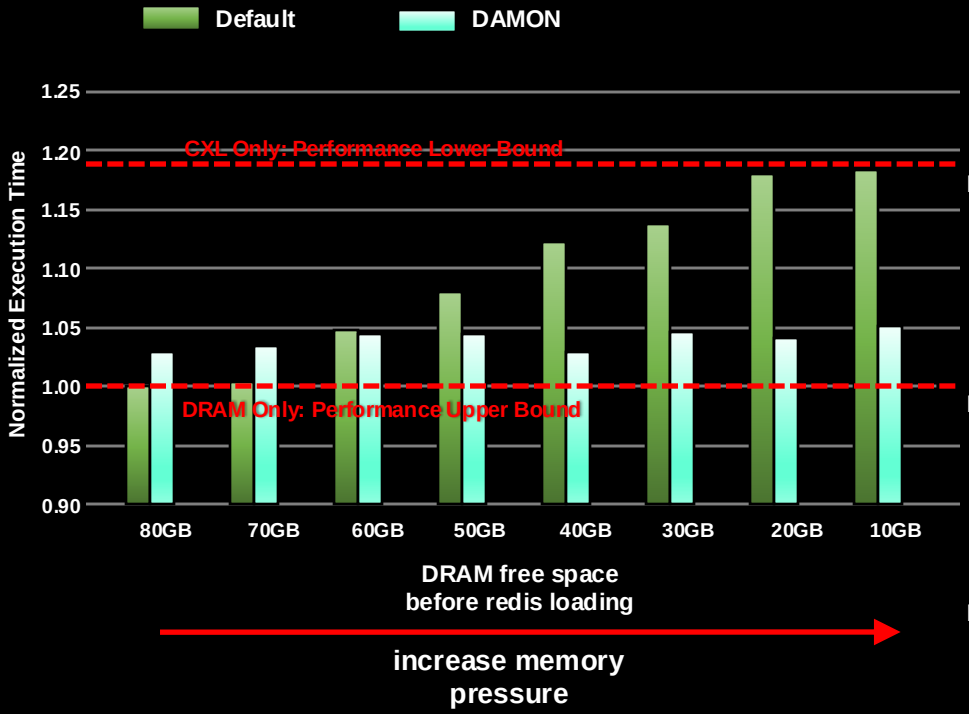


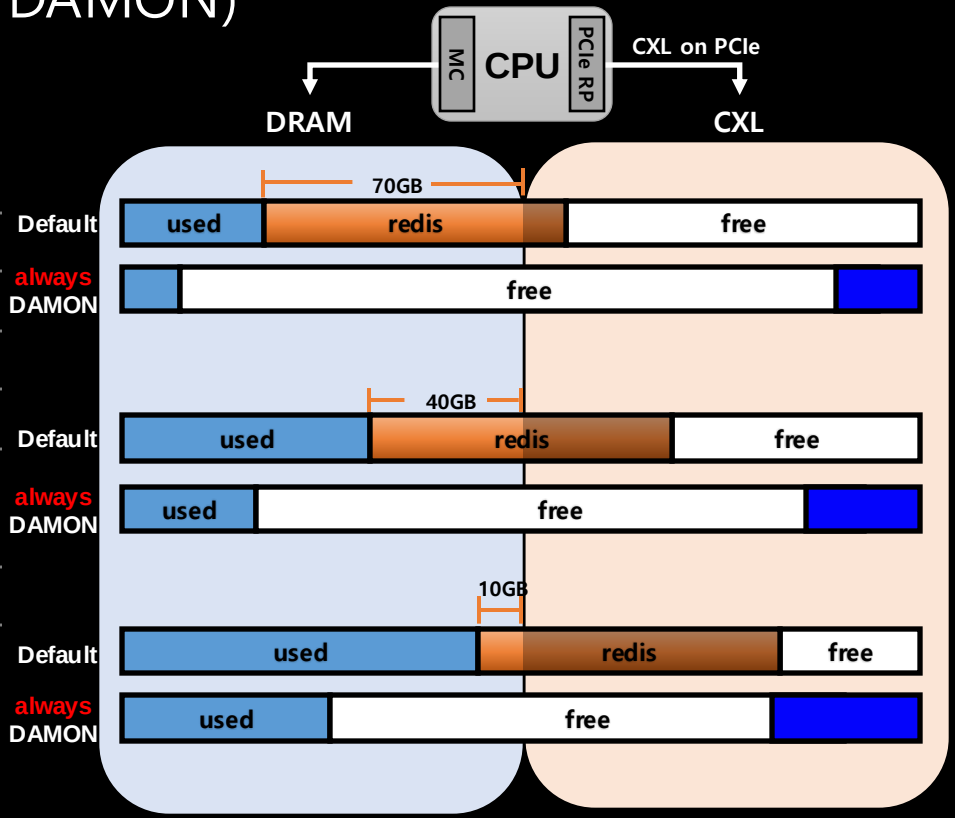


<Default>
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>
1. Demote cold data from DRAM to CXL memory.
2. Keep demote cold data

HMSDK Capacity Expansion (always DAMON)

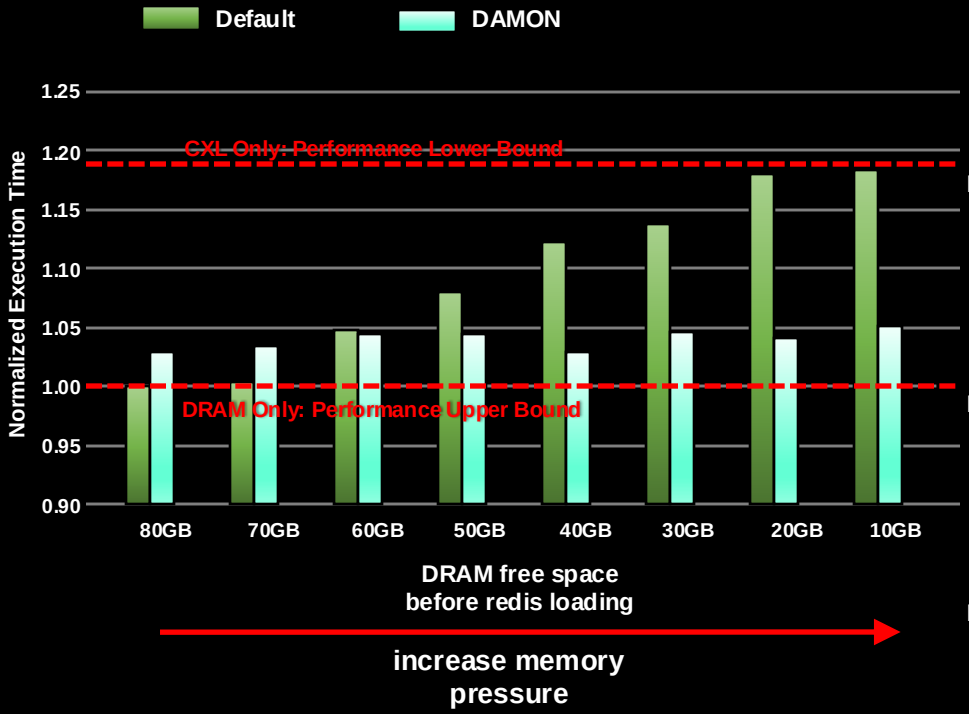


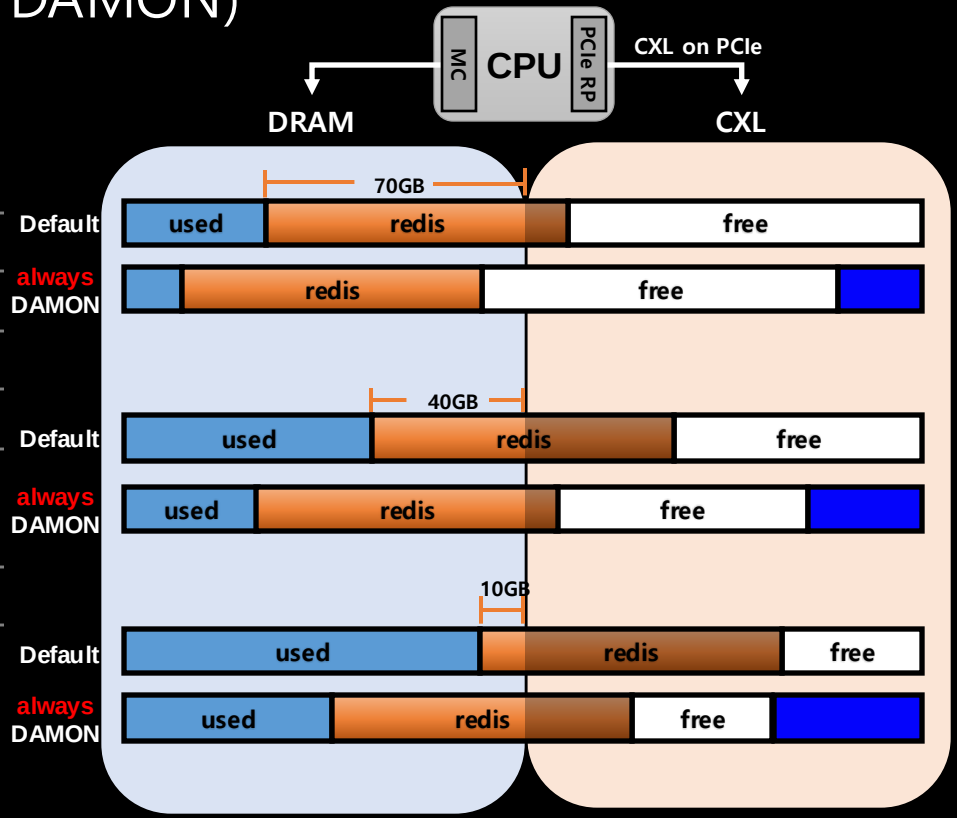


<Default>
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>
1. Demote cold data from DRAM to CXL memory.
2. Keep demote cold data

HMSDK Capacity Expansion (always DAMON)





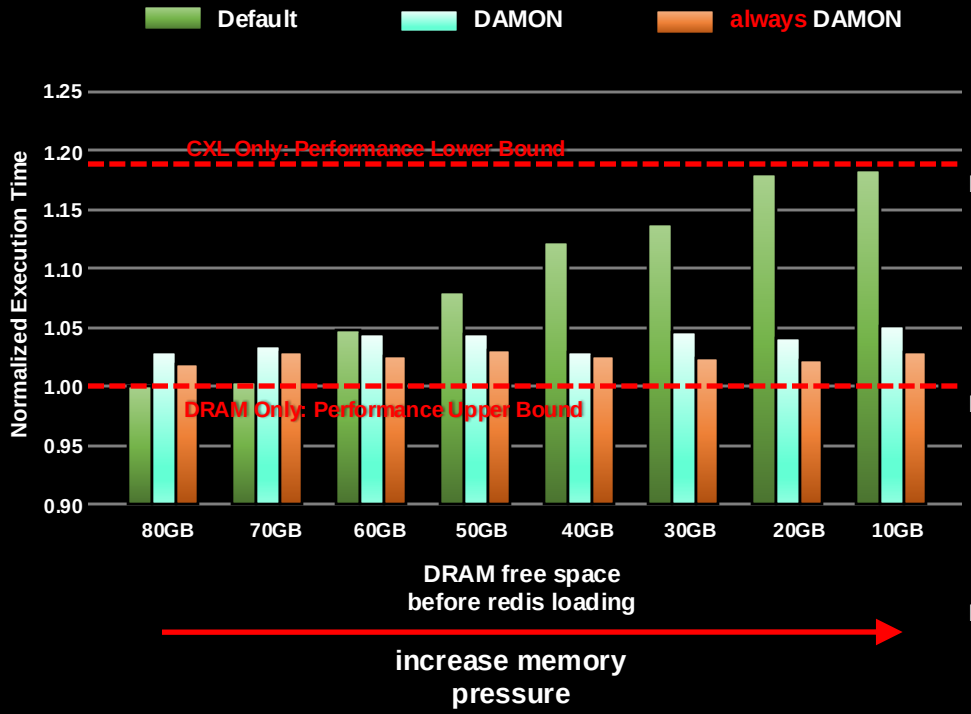
<Default>

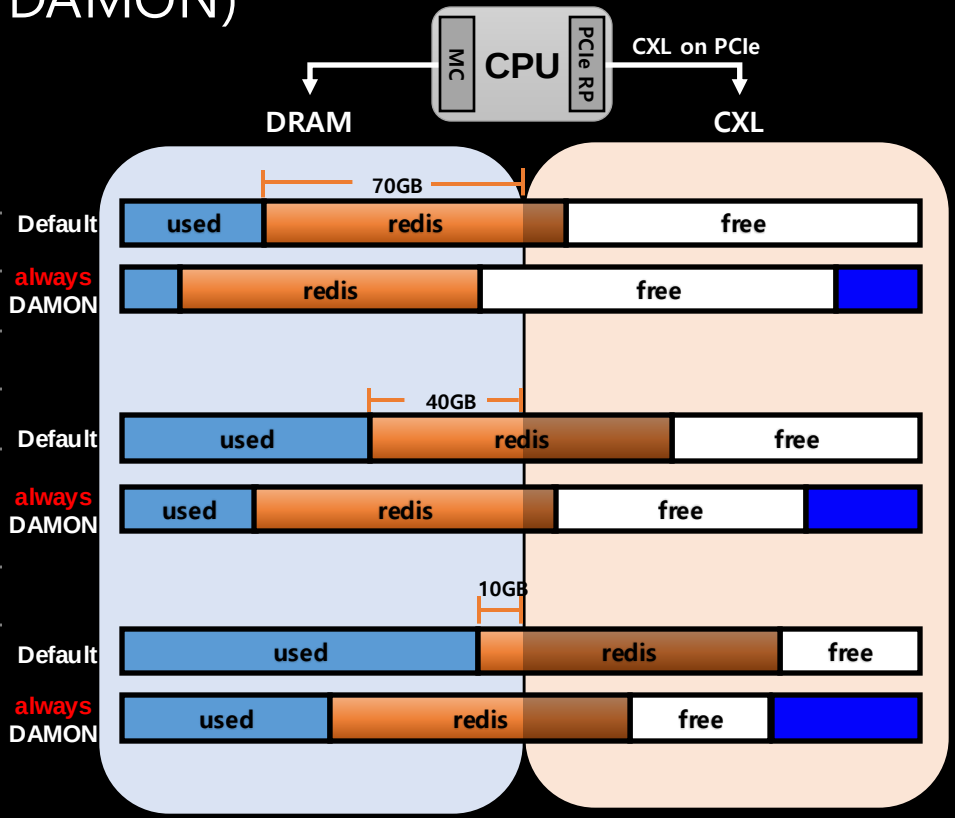
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

1. Demote cold data from DRAM to CXL memory.
2. Keep demote cold data
3. More redis data can be allocated on DRAM.

HMSDK Capacity Expansion (always DAMON)





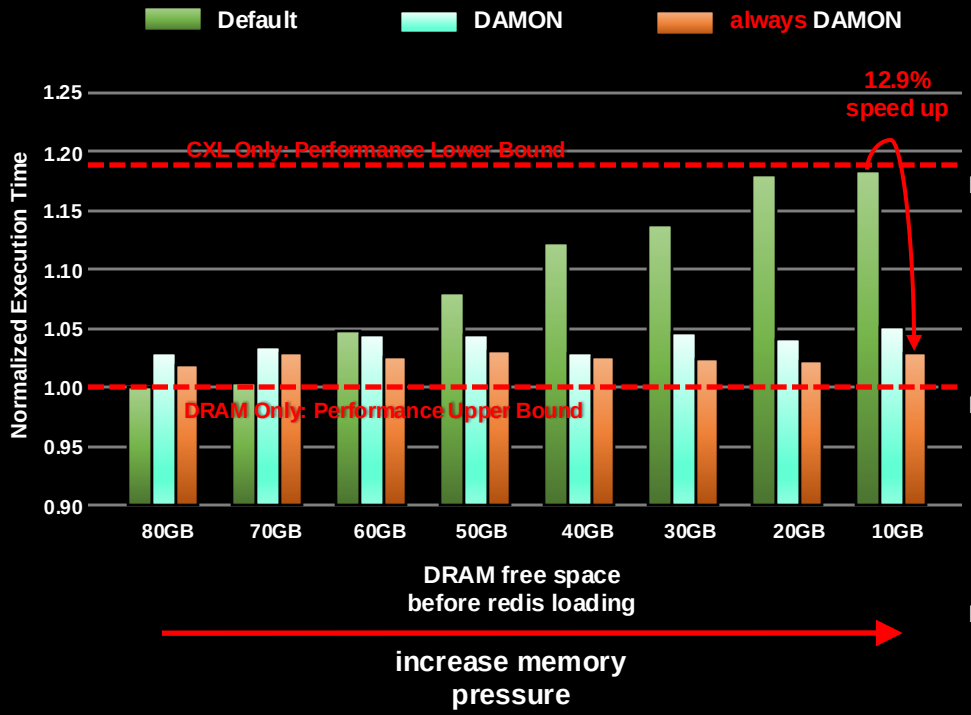
<Default>

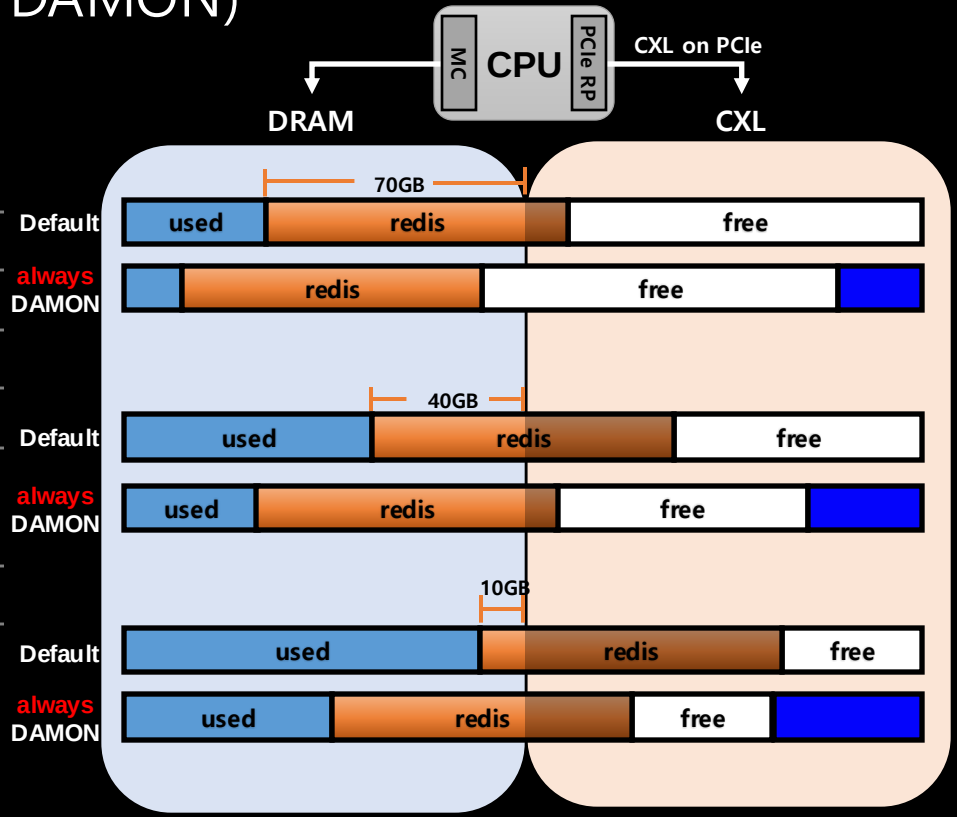
1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

1. Demote cold data from DRAM to CXL memory.
2. Keep demote cold data
3. More redis data can be allocated on DRAM.

HMSDK Capacity Expansion (always DAMON)





<Default>

1. DRAM is partially used by non-redis cold data.
2. Partial redis data is allocated on CXL memory.
   (due to insufficient space on DRAM)

<HMSDK>

1. Demote cold data from DRAM to CXL memory.
2. Keep demote cold data
3. More redis data can be allocated on DRAM.

Collaboration with DAMON community

- This work was done and presented at Open Source Summit Europe 2024.
  - DAMON Recipes: Ways to Save Memory Using a Linux Kernel Subsystem in the Real World
    - Presented by DAMON maintainer SeongJae Park@Meta and Honggyu Kim@SK hynix.

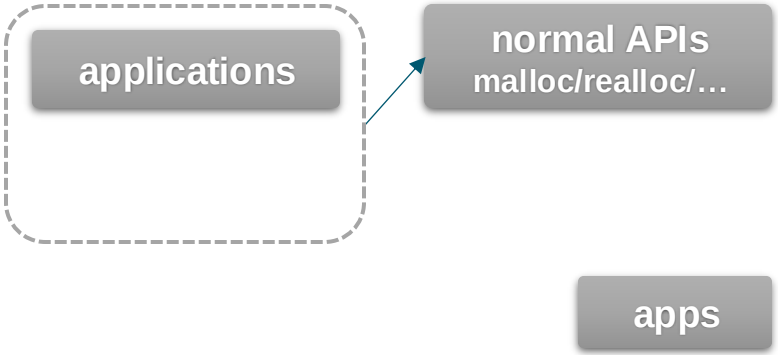
- The slide and video are available at
  - https://osseu2024.sched.com/event/1ej2S

Heterogeneous Memory Allocator (Custom Allocator)

With customized programs using hmalloc APIs.

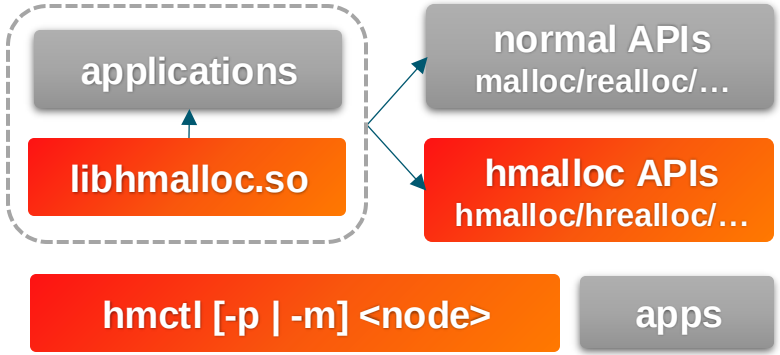
Heterogeneous Memory Allocator

- This can be used when users have knowledge of their program.
  - If users know which memory objects are cold,
    - then explicitly replace normal allocation APIs to hmalloc APIs.



Heterogeneous Memory Allocator

- This can be used when users have knowledge of their program.
  - If users know which memory objects are cold,
    - then explicitly replace normal allocation APIs to hmalloc APIs.
- HMSDK library usage (libhmalloc.so)
  - explicit allocation to a specific NUMA node (e.g. CXL memory)
    - e.g. ***hmalloc()***, ***hcalloc()***, ***hfree()***, etc.
  - ***numactl*** like ***hmctl*** tool
    - ***numactl*** applies process level mempolicy.
    - but ***hmctl*** applies mempolicy only to hmalloc area.
    - support ***--preferred*** and ***--membind*** modes.



Heterogeneous Memory Allocator

```
#include <hmalloc.h>
    ...
int main() {
    char *p = malloc(256 * MiB);
    char *hp = hmalloc(512 * MiB);
    memset(p, 'x', sz * MiB);
    memset(hp, 'x', hsz * MiB);
        ...
    getchar();  /* wait here */

    hfree(hp);
    free(p);
    return 0;
}
```

```
$ ./example &
[1] 49288
255 MiB is allocated by malloc().
511 MiB is allocated by hmalloc().
Press enter to stop.


$ numastat -c -p $!
Per-node process memory usage (in MBs) for PID 49288
(example)
         Node 0 Node 1 Node 2 Node 3 Total
         ------ ------ ------ ------ -----
Total       773      0      0      0   773
```

Heterogeneous Memory Allocator

```
#include <hmalloc.h>
    ...
int main() {
    char *p = malloc(256 * MiB);
    char *hp = hmalloc(512 * MiB);
    memset(p, 'x', sz * MiB);
    memset(hp, 'x', hsz * MiB);
        ...
    getchar();  /* wait here */

    hfree(hp);
    free(p);
    return 0;
}
```

```
$ ./example &
[1] 49288
255 MiB is allocated by malloc().
511 MiB is allocated by hmalloc().
Press enter to stop.


$ numastat -c -p $!
Per-node process memory usage (in MBs) for PID 49288
(example)
         Node 0 Node 1 Node 2 Node 3 Total
         ------ ------ ------ ------ -----
Total       773      0      0      0   773
```

Heterogeneous Memory Allocator

```
#include <hmalloc.h>
    ...
int main() {
    char *p = malloc(256 * MiB);
    char *hp = hmalloc(512 * MiB);
    memset(p, 'x', sz * MiB);
    memset(hp, 'x', hsz * MiB);
        ...
    getchar();  /* wait here */

    hfree(hp);
    free(p);
    return 0;
}
```

```
$ hmctl -m 2 ./example &
[1] 49288
255 MiB is allocated by malloc().
511 MiB is allocated by hmalloc().
Press enter to stop.


$ numastat -c -p $!
Per-node process memory usage (in MBs) for PID 49288
(example)
         Node 0 Node 1 Node 2 Node 3 Total
         ------ ------ ------ ------ -----
Total       261      0    512      0   773
```

Heterogeneous Memory Allocator

```
#include <hmalloc.h>
    ...
int main() {
    char *p = malloc(256 * MiB);
    char *hp = hmalloc(512 * MiB);
    memset(p, 'x', sz * MiB);
    memset(hp, 'x', hsz * MiB);
        ...
    getchar();  /* wait here */

    hfree(hp);
    free(p);
    return 0;
}
```

```
$ hmctl -m 3 ./example &
[1] 49288
255 MiB is allocated by malloc().
511 MiB is allocated by hmalloc().
Press enter to stop.


$ numastat -c -p $!
Per-node process memory usage (in MBs) for PID 49288
(example)
         Node 0 Node 1 Node 2 Node 3 Total
         ------ ------ ------ ------ -----
Total       261      0      0    512   773
```

Heterogeneous Memory Allocator

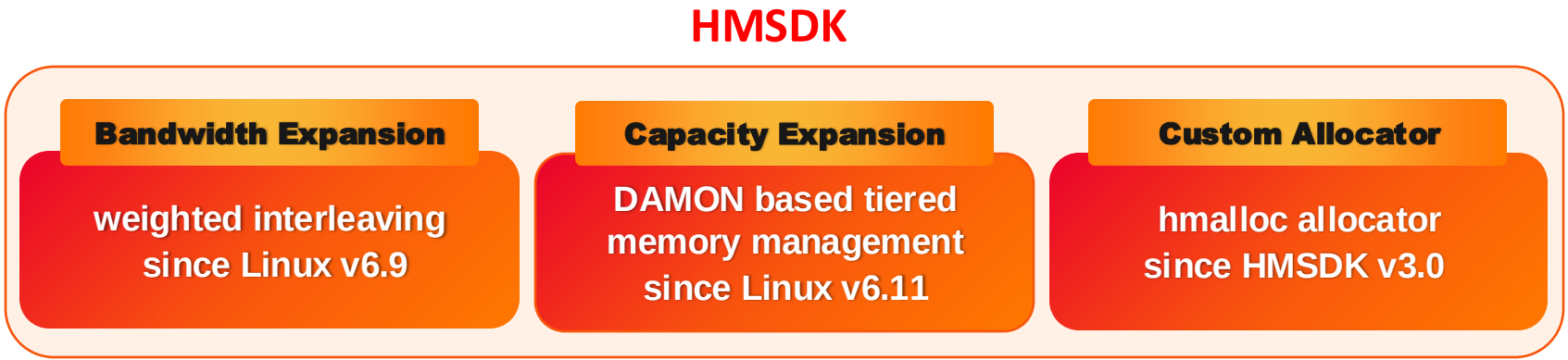
```c
#include <hmalloc.h>
    ...
int main() {
    char *p = malloc(256 * MiB);
    char *hp = hmalloc(512 * MiB);
    memset(p, 'x', sz * MiB);
    memset(hp, 'x', hsz * MiB);
        ...
    getchar();  /* wait here */

    hfree(hp);
    free(p);
    return 0;
}
```

```
$ numactl -p 1 hmctl -m 3 ./example &
[1] 49288
255 MiB is allocated by malloc().
511 MiB is allocated by hmalloc().
Press enter to stop.


$ numastat -c -p $!
Per-node process memory usage (in MBs) for PID 49288
(example)
         Node 0 Node 1 Node 2 Node 3 Total
         ------ ------ ------ ------ -----
Total         3    258      0    512   773
```

Conclusion

HMSDK v3.0 release

- HMSDK v3.0 was released Sep 2024 based on Linux v6.11!
  - Fully aligned with various open-source projects.
  - The official Linux kernel can directly be used for HMSDK.
  - https://github.com/skhynix/hmsdk/releases/tag/hmsdk-v3.0



HMSDK v3.0 release

- Linux kernel
  - weighted interleave (available from v6.9)
  - DAMON based tiered memory management (available from v6.11)

- numactl
  - -w/--weighted-interleave option (available on master)

- damo (DAMON user-space tool)
  - Interface for hot/cold migration (available from v2.4.0)
  - Support multiple kdamonds with yaml format.

Other Projects

- hwloc (Hardware locality)
  - Support HWLOC_MEMBIND_WEIGHTED_INTERLEAVE (available from hwloc-2.11.0)

- UMF (Unified Memory Framework)
  - Support UMF_NUMA_MODE_WEIGHTED_INTERLEAVE (proposed)

Conclusion

- We’re trying to expand CXL software ecosystem on Linux.
  - To make CXL memory usable by end users.
  - Such as software developers and system admins.

- New memory system requires software changes.
  - Especially for major open-source projects such as Linux kernel, etc.
  - Need to lower the huddle of using software techniques.

Thank You!

https://github.com/skhynix/hmsdk