System-wide Memory Defragmenter Without Killing any application

Name: Pintu Kumar

Email: pintu.k@samsung.com

Samsung R&D Institute India - Bangalore

CONTENT

- Objective
- Introduction
- Memory Reclaim Techniques in Kernel
- System-wide Memory Reclaim Techniques
- Experimentation Results
- Summary
- Conclusion

OBJECTIVE

- To quickly recover entire system memory in one shot without killing or closing already running application.
- To reduce memory fragmentation to some extent.
- To avoid higher-order allocation to enter slow path again and again.
- To provide interface to user space for quickly reclaiming entire system memory as much as possible.
- To bring back the entire system memory to a stage where it looks like fresh reboot.

INTRODUCTION

- **Memory fragmentation?**
  - Non availability of higher order contiguous pages, although there are lots of free pages in smaller order which are not contiguous.

 # cat /proc/buddyinfo

	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone Normal	972	352	171	25	0	0	0	0	0	0	0

Higher-order pages

Free Memory = (972*1 + 352*2 + 171*4 + 25*8) = 2560*4K = 10MB

Although we have 10MB memory free, still the request for 2^4 order (16*4K = 64K contiguous block) may fail.
This situation is known as external memory fragmentation.

- **To measure fragmentation level across each order, following formula can be used:**

$$FragLevel(\%) = \frac{TotalFreePages - \sum_{i=j}^{N} (2^i . k_i)}{TotalFreePages} X100$$

TotalFreePages	**=**	**Total number of free pages in each Node**
N	**=**	**MAX_ORDER - 1 ➔ The highest order of allocation**
j	**=**	**the desired order requested**
i	**=**	**page order ➔ 0 to N**
Ki	**=**	**Number of free pages in ith order block**

- **Cat /proc/buddyinfo can be used to measure the fragmentation level.**
- **We have developed a user-space utility to measure the overall fragmentation level of the system.**
- **OUTPUT is shown below:**

Order	2-Power	Nr Pages	Free Pages	Frag Level (%)
0	1	972	972	**0%**
1	2	352	704	**37%**
2	4	171	684	**65%**
3	8	25	200	**92%**
4	16	0	0	**100%**
5	32	0	0	**100%**
6	64	0	0	**100%**
7	128	0	0	**100%**
8	256	0	0	**100%**
9	512	0	0	**100%**
10	1024	0	0	**100%**
Total			**2560**	**81%**

Average value

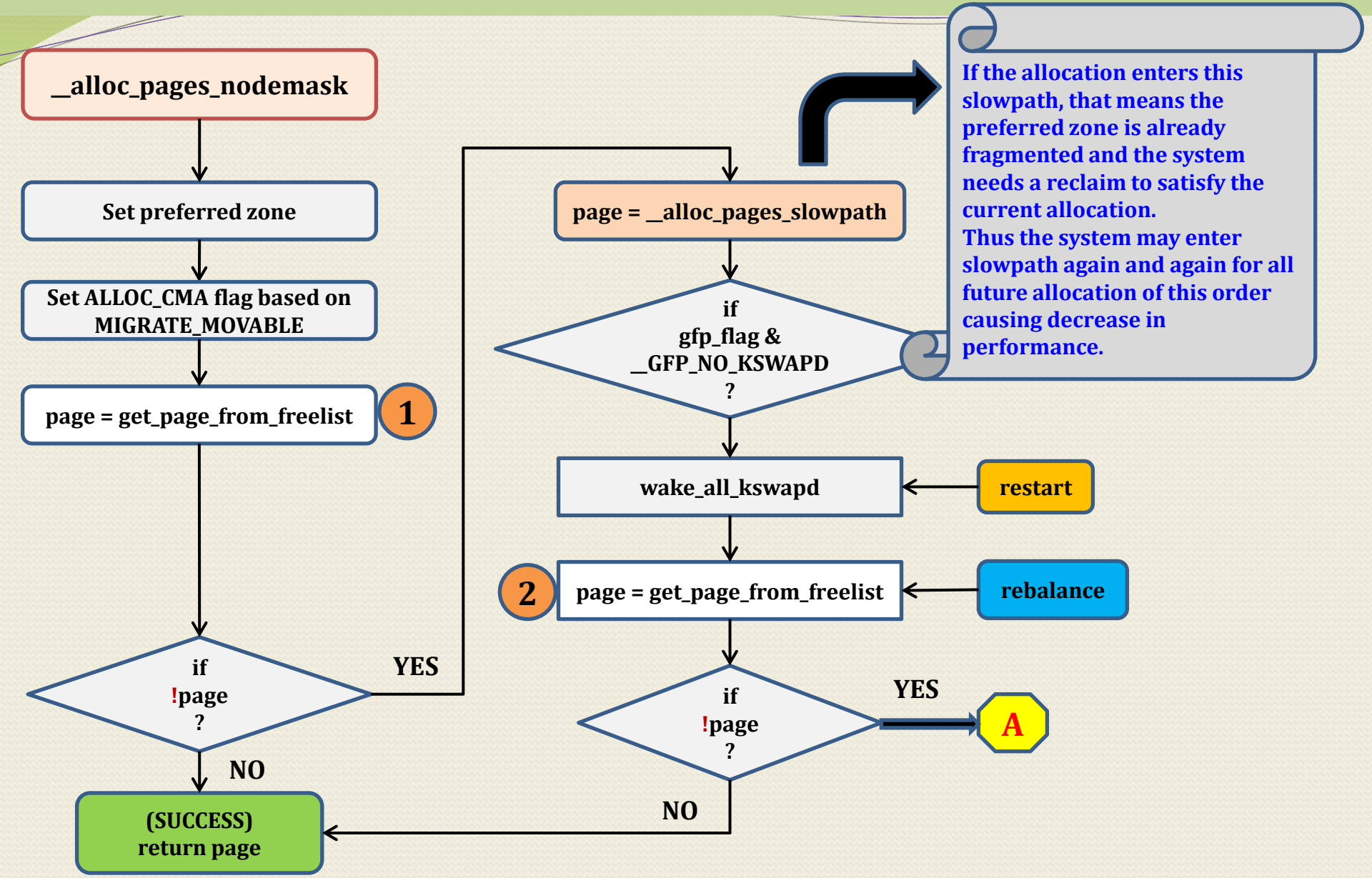
- **However, if COMPACTION is enabled, the fragmentation level can be measured directly using:**
- **cat /sys/kernel/debug/extfrag/unusable_index**

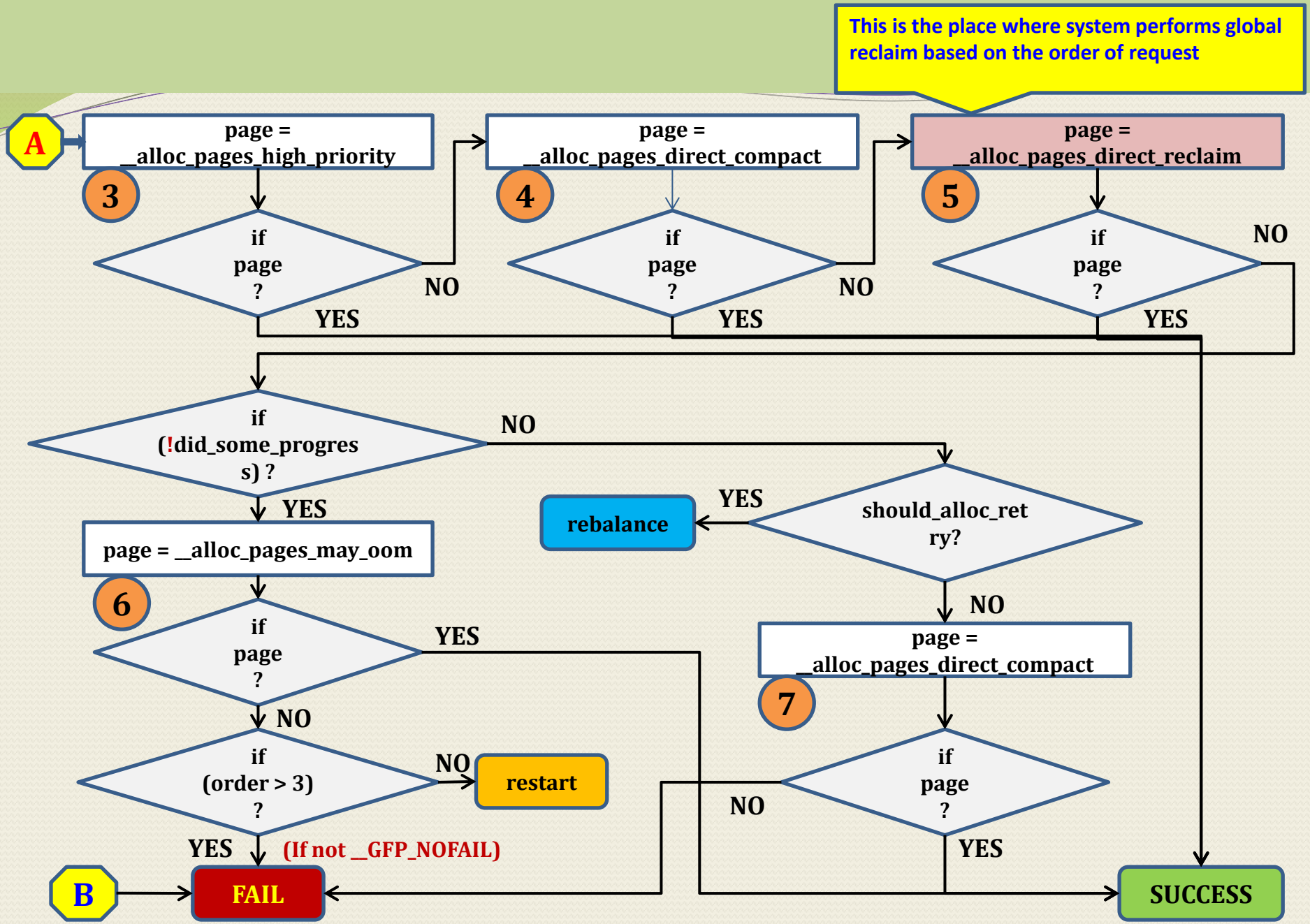
Node 0, zone Normal	0.000	3.797	6.547	9.219	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Order	Index	FragLevel (%)
0	0.000	**0.00**
1	0.379	**37.90**
2	0.654	**65.40**
3	0.921	**92.10**
4	1.000	**100.00**
5	1.000	**100.00**
6	1.000	**100.00**
7	1.000	**100.00**
8	1.000	**100.00**
9	1.000	**100.00**
10	1.000	**100.00**
Average		**81.40**

- ✓ **Here, to get the fragmentation level, just multiply the unusable index value by 100.**
- ✓ **You can observe that the results obtained by our frag level calculation in previous slide and this usable index is almost same.**
- ✓ **Soon we will contribute this utilities to open source.**

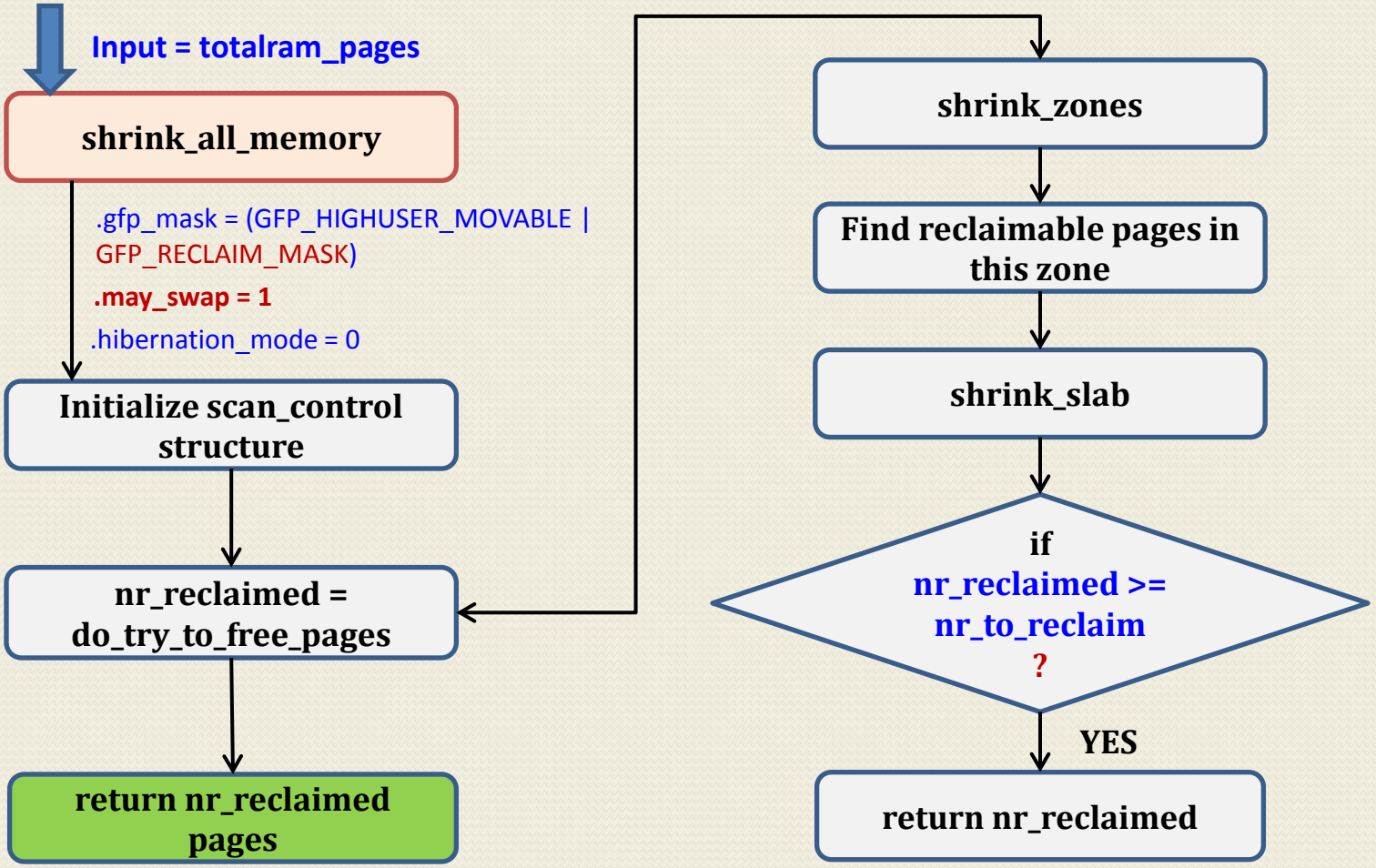
MEMORY RECLAIM TECHNIQUES IN KERNEL




This is the place where system performs global reclaim based on the order of request
A
page = _alloc_pages_high_priority
3
if page ?
NO
YES
page = __alloc_pages_direct_compact
4
if page ?
NO
YES
page = __alloc_pages_direct_reclaim
5
if page ?
NO
YES
if (!did_some_progres s) ?
NO
YES
page = __alloc_pages_may_oom
6
if page ?
YES
NO
if (order > 3) ?
NO
restart
YES
(If not __GFP_NOFAIL)
B
FAIL
rebalance
YES
should_alloc_ret ry?
NO
page = __alloc_pages_direct_compact
7
if page ?
NO
YES
SUCCESS

SYSTEM-WIDE MEMORY RECLAIM TECHNIQUES

#if defined CONFIG_HIBERNATION || CONFIG_SHRINK_MEMORY



#endif

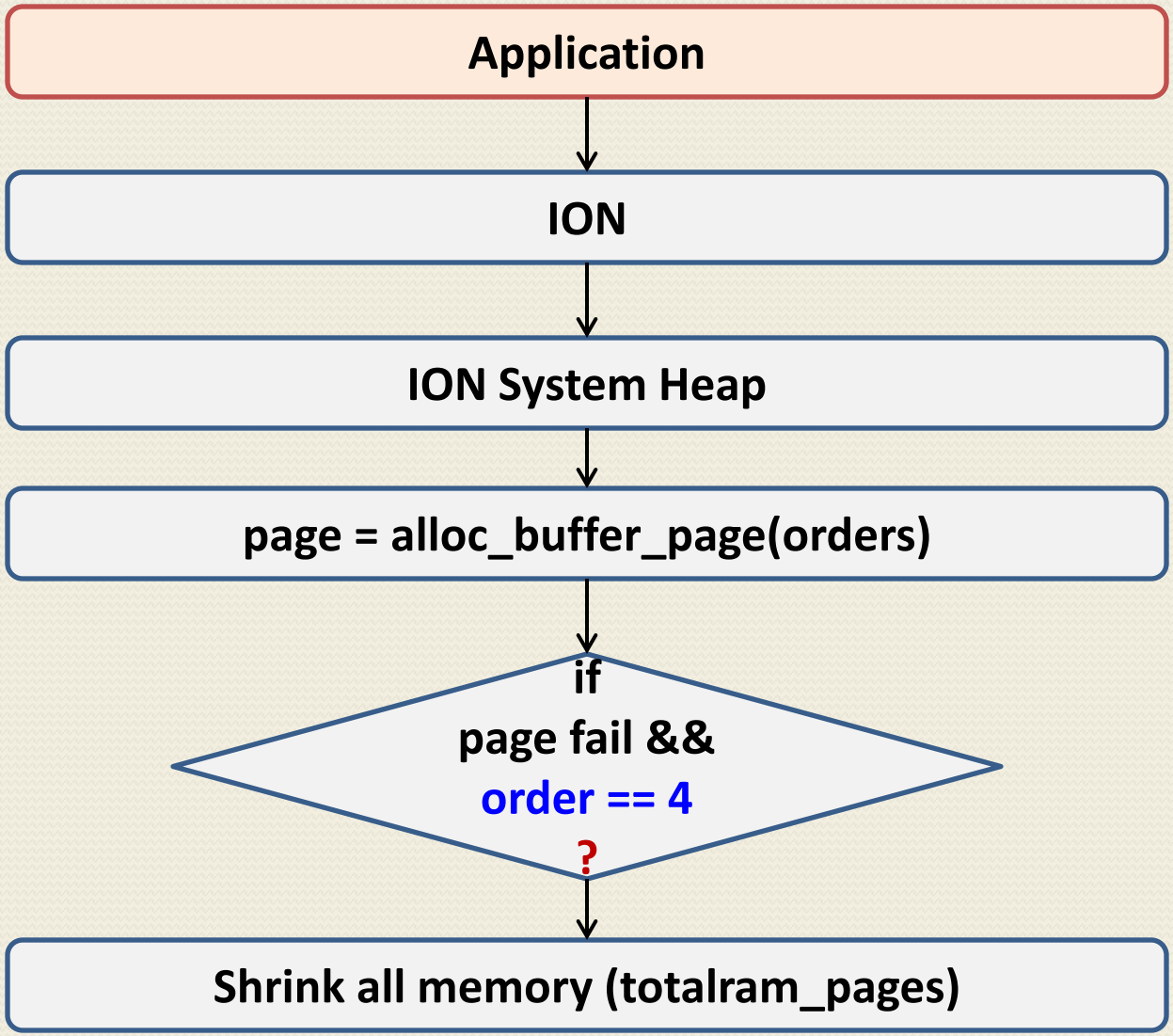
- System-wide memory reclaim in kernel can be performed using the shrink_all_memory() under mm/vmscan.c
- It takes only one input: no. of pages to be reclaimed. In our case we pass the entire system memory.
- It can perform entire system-wide reclaim across all zones, in one shot.
- It can reduce fragmentation by bringing back high-order pages quickly, and avoid slowpath.
- Currently shrink_all_memory is used only during hibernation case: kernel/power/snapshot.c: hibernate_preallocate_memory().
- We can use this function to invoke system-wide reclaim even from user-space or any other kernel sub-system.

Shrink Memory From User Space

```
int shrink_memory(struct shrink_status *status)
{
	int memfree1,memfree2;
	int totalfreed = 0;
	int ntimes = 0;

	while (ntimes < 10) {
		fprintf(stderr,". ");
		memfree1 = get_free_memory();
		system("echo 1 > /proc/sys/vm/shrink_memory");
		sleep(1);
		system("echo 1 > /proc/sys/vm/compact_memory");
		sleep(1);
		memfree2 = get_free_memory();
		totalfreed = totalfreed + (memfree2 - memfree1);
		ntimes++;
	}
	status->total_recovered = totalfreed;
	return 0;
}
```

Shrink Memory from ION driver



orders[] = {8, 4, 0}

EXPERIMENTATION RESULTS – USER SPACE

Test Results: ARM: Device 1

RAM: 512MB

Kernel Version: 3.4

Scenario1: After initial boot-up.

BEFORE:

free -tm	total	used	free	shared	buffers	cached
Mem:	468	390	78	0	16	172
-/+ buffers/cache:		201	267			
ZRAM Swap:	0	0	0			
Total:	468	390	78			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone Normal	217	86	24	24	8	2	2	3	1	2	17

AFTER:

free -tm	total	used	free	shared	buffers	cached
Mem:	468	217	250	0	0	21
-/+ buffers/cache:		195	272			
ZRAM Swap:	0	0	0			
Total:	468	217	250			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone Normal	246	230	97	40	16	3	6	3	5	4	57

Output of memory shrinker after boot-up:

```
sh-3.2# ./memory_shrinker.out
Total Memory: 468 MB
  Used Memory: 390 MB
  Free Memory: 78 MB
 Cached Memory: 189 MB
--------------------------------
  Used Memory: 216 MB
  Free Memory: 252 MB
 Cached Memory: 22 MB
--------------------------------
Total Memory Recovered: 174 MB
```

- After initial boot-up, free memory was: 78MB
- Total memory recovered (10 iterations), by memory shrinker: 174MB.
- Final free memory becomes: ~250MB

Memory Fragmentation Results:

BEFORE:

Zone:	Normal
Order	**Fragmentation[%]**
0	0.00%
1	1.00%
2	1.90%
3	2.30%
4	3.30%
5	3.90%
6	4.30%
7	4.90%
8	6.80%
9	8.10%
10	13.20%
Overall	**4.52%**

AFTER:

Zone:	Normal
Order	**Fragmentation[%]**
0	0.00%
1	0.30%
2	1.00%
3	1.60%
4	2.10%
5	2.50%
6	2.60%
7	3.20%
8	3.80%
9	5.80%
10	9.00%
Overall	**2.90%**

- **Initial boot-up fragmentation level was: 4.52%**
- **With memory shrinker fragmentation level becomes: 2.90%**

Scenario2: After many application launch.

BEFORE:

free -tm	total	used	free	shared	buffers	cached
Mem:	468	455	12	0	4	72
-/+ buffers/cache:		379	88			
ZRAM Swap:	93	34	59			
Total:	562	490	71			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone Normal	972	352	171	52	14	3	1	0	0	0	0

AFTER:

free -tm	total	used	free	shared	buffers	cached
Mem:	468	362	105	0	3	41
-/+ buffers/cache:		318	150			
ZRAM Swap:	93	90	3			
Total:	562	453	109			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone Normal	473	218	1316	802	373	102	31	9	2	3	0

Output of memory shrinker after application launch and moving them to background:

```
sh-3.2# ./memory_shrinker.out
Total Memory: 468 MB
  Used Memory: 457 MB
  Free Memory: 11 MB
 Cached Memory: 77 MB
--------------------------------
  Used Memory: 323 MB
  Free Memory: 145 MB
 Cached Memory: 45 MB
--------------------------------
Total Memory Recovered: 136 MB
```

- **After many application launch, free memory becomes: 12MB**
- **Total memory recovered (10 iterations), by memory shrinker: 136MB.**
- **Final free settles down to: ~105MB (because some memory are immediately consumed back by the running applications/services)**

Memory Compaction Results:

BEFORE:	AFTER:
sh-3.2# cat /proc/vmstat | grep compact compact_blocks_moved 40 compact_pages_moved 1816 compact_pagemigrate_failed 33865 compact_stall 510 compact_fail 192 compact_success 3	sh-3.2# cat /proc/vmstat | grep compact compact_blocks_moved 64 compact_pages_moved 3197 compact_pagemigrate_failed 59042 compact_stall 662 compact_fail 192 compact_success 3

- **Even after memory shrinker, compaction did not succeed.**
- **But lots of pages were moved, which resulted into creating more numbers of higher order pages.**

Memory Fragmentation Results:

BEFORE:

Zone:	Normal
Order	**Fragmentation[%]**
0	0.00%
1	30.10%
2	52.50%
3	74.30%
4	87.60%
5	94.80%
6	97.90%
7	100.00%
8	100.00%
9	100.00%
10	100.00%
Overall	**76.11%**

AFTER:

Zone:	Normal
Order	**Fragmentation[%]**
0	0.00%
1	1.70%
2	3.30%
3	22.60%
4	46.40%
5	68.60%
6	80.70%
7	88.10%
8	92.30%
9	94.20%
10	100.00%
Overall	**54.35%**

- **After many application launch fragmentation level becomes: 76.11%**
- **With memory shrinker, fragmentation level decreases to: 54.35%**

Test Results: ARM: Device 2

RAM: 512MB

Kernel Version: 3.10

Scenario1: After initial boot-up.

BEFORE:

free -tm	total	used	free	shared	buffers	cached
Mem:	460	429	31	0	24	176
-/+ buffers/cache:		229	231			
ZRAM Swap:	0	0	0			
Total:	460	429	31			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone Normal	445	197	88	22	5	3	2	1	2	3	4

AFTER:

free -tm	total	used	free	shared	buffers	cached
Mem:	460	271	188	0	6	59
-/+ buffers/cache:		205	254			
ZRAM Swap:	0	0	0			
Total:	460	271	188			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone Normal	298	348	489	1189	611	199	42	23	11	6	8

Output of memory shrinker after boot-up:

```
sh-3.2# ./memory_shrinker.out
Total Memory:  460 MB
  Used Memory:  429 MB
  Free Memory:  31 MB
 Cached Memory:  200 MB
--------------------------------
  Used Memory:  209 MB
  Free Memory:  251 MB
 Cached Memory:  66 MB
--------------------------------
Total Memory Recovered:  221 MB
```

- After initial boot-up, free memory was: 31 MB
- Total memory recovered (10 iterations), by memory shrinker: 221 MB.
- Final free memory becomes: ~188 MB (after services reclaimed back its memory)

Memory Fragmentation Results:

BEFORE:

Zone:	Normal
Order	**Fragmentation[%]**
0	0.00%
1	5.60%
2	10.50%
3	14.90%
4	17.10%
5	18.10%
6	19.30%
7	20.90%
8	22.50%
9	29.00%
10	48.30%
Overall	**18.75%**

AFTER:

Zone:	Normal
Order	**Fragmentation[%]**
0	0.00%
1	1.00%
2	11.20%
3	23.70%
4	39.40%
5	55.60%
6	66.30%
7	70.70%
8	75.40%
9	80.50%
10	86.40%
Overall	**46.38%**

- **Here, fragmentation level increases, because lots of lower order pages were recovered compared to higher order.**
- **Although final free memory increased from: 31MB to 188MB**

Scenario2: After many application launch.

BEFORE:

free -tm	total	used	free	shared	buffers	cached
Mem:	460	440	20	0	5	65
-/+ buffers/cache:		369	90			
ZRAM Swap:	92	60	31			
Total:	552	501	51			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone Normal	1728	498	138	39	1	0	0	0	0	1	1

AFTER:

free -tm	total	used	free	shared	buffers	cached
Mem:	460	352	107	0	1	31
-/+ buffers/cache:		319	140			
ZRAM Swap:	92	92	0			
Total:	552	444	107			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone Normal	595	2884	1236	432	201	89	34	12	6	1	1

Output of memory shrinker after application launch and moving them to background:

```
sh-3.2# ./memory_shrinker.out
Total Memory:  460 MB
  Used Memory:  441 MB
  Free Memory:  19 MB
 Cached Memory:  72 MB
--------------------------------
  Used Memory:  327 MB
  Free Memory:  133 MB
 Cached Memory:  33 MB
--------------------------------
Total Memory Recovered:  114 MB
```

- After many application launch, free memory becomes: 19 MB
- Total memory recovered (10 iterations), by memory shrinker: 114 MB.
- Final free settles down to: ~107 MB (because some memory are immediately consumed back by the running applications/services)

Memory Compaction Results:

BEFORE:	AFTER:
sh-3.2# cat /proc/vmstat | grep compact compact_migrate_scanned 164681 compact_free_scanned 1064111 compact_isolated 33137 compact_stall 69 compact_fail 42 compact_success 19	sh-3.2# cat /proc/vmstat | grep compact compact_migrate_scanned 223633 compact_free_scanned 1116864 compact_isolated 54976 compact_stall 69 compact_fail 42 compact_success 19

- **Even after memory shrinker, compaction did not succeed.**
- **But lots of pages were migrated/scanned, which resulted into creating more numbers of higher order pages.**

Memory Fragmentation Results:

BEFORE:

Zone:	Normal
Order	**Fragmentation[%]**
0	0.00%
1	32.40%
2	52.10%
3	63.00%
4	69.20%
5	69.50%
6	69.50%
7	69.50%
8	69.50%
9	69.50%
10	79.60%
Overall	**58.53%**

AFTER:

Zone:	Normal
Order	**Fragmentation[%]**
0	0.00%
1	2.10%
2	23.00%
3	40.90%
4	53.40%
5	65.10%
6	75.40%
7	83.30%
8	88.80%
9	94.40%
10	96.20%
Overall	**56.60%**

- **Although, overall fragmentation becomes little less, in this case, but still it could recover large number of middle order pages.**

Test Results: Ubuntu 12.10

RAM: 768MB

Kernel Version: 3.10

Scenario1: After initial boot-up.

BEFORE:

free -tm	total	used	free	shared	buffers	cached
Mem:	749	697	51	0	27	252
-/+ buffers/cache:		417	332			
Physical Swap:	1021	0	1021			
Total:	1771	697	1073			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone DMA	2	2	2	0	3	1	2	1	1	1	0
Node 0, zone Normal	48	188	126	107	38	13	4	1	1	1	8

AFTER:

free -tm	total	used	free	shared	buffers	cached
Mem:	749	331	417	0	21	90
-/+ buffers/cache:		219	529			
Physical Swap:	1021	302	719			
Total:	1771	634	1136			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone DMA	1	1	2	1	3	3	2	2	3	1	2
Node 0, zone Normal	151	124	65	37	31	20	2	44	52	18	71

Output of memory shrinker after boot-up:

```
sh-3.2# ./memory_shrinker.out
Total Memory:   749 MB
  Used Memory:   698 MB
  Free Memory:   51 MB
 Cached Memory:   281 MB
--------------------------------
  Used Memory:   84 MB
  Free Memory:   665 MB
 Cached Memory:   112 MB
--------------------------------
Total Memory Recovered:   615 MB
```

- After initial boot-up, free memory was: 51 MB
- Total memory recovered (10 iterations), by memory shrinker: 615 MB.
- Final free memory becomes: ~417 MB (after some services/applications reclaimed back its memory)

Memory Fragmentation Results:

BEFORE:

Zone:	DMA	Normal
Order	**Fragmentation[%]**	**Fragmentation[%]**
0	0.00%	0.00%
1	0.10%	0.00%
2	0.50%	2.40%
3	1.20%	6.60%
4	1.20%	13.70%
5	5.50%	18.80%
6	8.40%	22.20%
7	19.80%	24.40%
8	31.30%	25.40%
9	54.20%	27.60%
10	100.00%	31.80%
Overall	**20.20%**	**15.72%**

AFTER:

Zone:	DMA	Normal
Order	**Fragmentation[%]**	**Fragmentation[%]**
0	0.00%	0.00%
1	0.00%	0.10%
2	0.00%	0.20%
3	0.20%	0.50%
4	0.40%	0.70%
5	1.70%	1.20%
6	4.20%	1.80%
7	7.50%	2.00%
8	14.10%	7.40%
9	33.90%	20.40%
10	47.10%	29.30%
Overall	**9.92%**	**5.78%**

- **Here, fragmentation reduces for both the zones.**
- **Plenty of higher order pages were recovered.**
- **Final free memory increased from: 51MB to 417MB**

Scenario2: After many application launch.

BEFORE:

free -tm	total	used	free	shared	buffers	cached
Mem:	749	685	63	0	5	87
-/+ buffers/cache:		593	156			
Physical Swap:	1021	445	576			
Total:	1771	1130	640			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone DMA	31	6	9	5	6	6	0	0	0	1	0
Node 0, zone Normal	4039	859	336	120	113	66	25	6	0	0	1

AFTER:

free -tm	total	used	free	shared	buffers	cached
Mem:	749	620	128	0	11	129
-/+ buffers/cache:		479	270			
Physical Swap:	1021	580	441			
Total:	1771	1201	569			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone DMA	89	84	75	58	43	19	8	2	1	1	0
Node 0, zone Normal	349	388	270	66	46	23	16	12	3	0	21

Output of memory shrinker after application launch and moving them to background:

```
sh-3.2# ./memory_shrinker.out
Total Memory:  749 MB
  Used Memory:  704 MB
  Free Memory:  45 MB
 Cached Memory:   93 MB
--------------------------------
  Used Memory:  166 MB
  Free Memory:   583 MB
 Cached Memory:  142 MB
--------------------------------
Total Memory Recovered:  545 MB
```

- **After many application launch, free memory becomes: 45 MB**
- **Total memory recovered (10 iterations), by memory shrinker: 545 MB.**
- **Final free settles down to: ~ 128 MB (because some memory are immediately consumed back by the running applications/services)**

Memory Fragmentation Results:

BEFORE:

Zone:	DMA	Normal
Order	**Fragmentation[%]**	**Fragmentation[%]**
0	0.00%	0.00%
1	3.30%	25.20%
2	4.60%	36.30%
3	8.50%	45.20%
4	12.90%	51.60%
5	23.30%	63.50%
6	44.20%	77.50%
7	44.20%	88.10%
8	44.20%	93.20%
9	44.20%	93.20%
10	100.00%	69.64%
Overall	**29.95%**	**60.64%**

AFTER:

Zone:	DMA	Normal
Order	**Fragmentation[%]**	**Fragmentation[%]**
0	0.00%	0.00%
1	2.30%	0.70%
2	6.60%	3.40%
3	14.40%	7.00%
4	26.40%	8.90%
5	44.30%	11.40%
6	60.10%	14.00%
7	73.40%	17.50%
8	80.00%	22.80%
9	86.70%	25.50%
10	100.00%	25.50%
Overall	**44.93%**	**12.43%**

- **Here, fragmentation reduces drastically for Normal zone.**
- **But plenty of higher order pages were recovered from both the zones.**
- **Also Final free memory increases by 8-fold, from: 51MB to 417MB**

EXPERIMENTATION RESULTS – KERNEL SPACE

Test Results: ARM: Device 2

RAM: 512MB

Kernel Version: 3.10

Using: ION Driver in Kernel

Scenario3: After running many applications

BEFORE:

free -tm	total	used	free	shared	buffers	cached
Mem:	460	453	7	0	2	58
-/+ buffers/cache:		391	68			
ZRAM Swap:	92	0	92			
Total:	552	453	99			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone Normal	162	104	287	46	0	0	0	0	0	0	0

AFTER:

free -tm	total	used	free	shared	buffers	cached
Mem:	460	331	128	0	3	52
-/+ buffers/cache:		275	184			
ZRAM Swap:	92	85	7			
Total:	552	416	135			

buddyinfo	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Node 0, zone Normal	271	139	1563	1380	462	101	14	2	7	1	1

ION System Heap Results:

BEFORE:

```
client     pid     size
-----------------------------------------------
client:drm pid:1 size:79822848
-----------------------------------------------
total orphaned    0
total    79826944
deferred free    0
-----------------------------------------------
0 order 8 highmem pages in pool = 0 total
0 order 8 lowmem pages in pool = 0 total
0 order 4 highmem pages in pool = 0 total
0 order 4 lowmem pages in pool = 0 total
0 order 0 highmem pages in pool = 0 total
0 order 0 lowmem pages in pool = 0 total
```

- **No memory left in ION page pool.**
- **For every allocation request, system is bound to take slow-path.**
- **Application performance tends to degrade from this point.**

AFTER:

```
client     pid     size
-----------------------------------------------
client:drm pid:1 size:31883264
-----------------------------------------------
total orphaned    0
total    31883264
deferred free    0
-----------------------------------------------
0 order 8 highmem pages in pool = 0 total
9 order 8 lowmem pages in pool = 9437184 total
0 order 4 highmem pages in pool = 0 total
463 order 4 lowmem pages in pool = 30343168 total
0 order 0 highmem pages in pool = 0 total
117 order 0 lowmem pages in pool = 479232 total
```

- **Lots of order {8, 4, 0} pages were recovered.**
- **Performance is degraded only once, during recovery, but later it improves and stays for long time.**

Logs output during ION system heap allocation:

```
Kernel: [  495.009158]  [1:     Xorg:  331] [c1] [PINTU]: shrink_all_memory: nr_reclaimed: 26484
Kernel: [  495.752874]  [1:     Xorg:  331] [c1] [PINTU]: shrink_all_memory: nr_reclaimed: 12890
Kernel: [  495.806645]  [1:     Xorg:  331] [c1] [PINTU]: shrink_all_memory: nr_reclaimed: 1862
Kernel: [  495.826925]  [1:     Xorg:  331] [c1] [PINTU]: shrink_all_memory: nr_reclaimed: 669
Kernel: [  495.838600]  [1:     Xorg:  331] [c1] [PINTU]: shrink_all_memory: nr_reclaimed: 142
Kernel: [  495.847219]  [1:     Xorg:  331] [c1] [PINTU]: shrink_all_memory: nr_reclaimed: 122
Kernel: [  495.853767]  [1:     Xorg:  331] [c1] [PINTU]: shrink_all_memory: nr_reclaimed: 65
Kernel: [  495.859387]  [1:     Xorg:  331] [c1] [PINTU]: shrink_all_memory: nr_reclaimed: 82
Kernel: [  495.865089]  [1:     Xorg:  331] [c1] [PINTU]: shrink_all_memory: nr_reclaimed: 102
Kernel: [  495.868730]  [1:     Xorg:  331] [c1] [PINTU]: shrink_all_memory: nr_reclaimed: 47
Kernel: [  495.868757]  [1:     Xorg:  331] [c1] [PINTU]: Order:4, Total pages shrinked: 42465
```

- **Earlier, during ION system heap allocation, for every order-4 allocation, it fallback to order-0 allocation. Thus application performance will be degraded.**
- **With shrink memory during order-4 allocation failure, the fallback will happen only once. The next order-4 allocations will pass.**
- **Chances are that even order-8 allocation may pass, which will never happen in earlier case. Thus application launch performance can be increased and OOM can be delayed.**

Memory Fragmentation Results:

BEFORE:

Zone:	Normal
Order	**Fragmentation[%]**
0	0.00%
1	12.10%
2	25.10%
3	89.50%
4	100.00%
5	100.00%
6	100.00%
7	100.00%
8	100.00%
9	100.00%
10	100.00%
Overall	**75.15%**

AFTER:

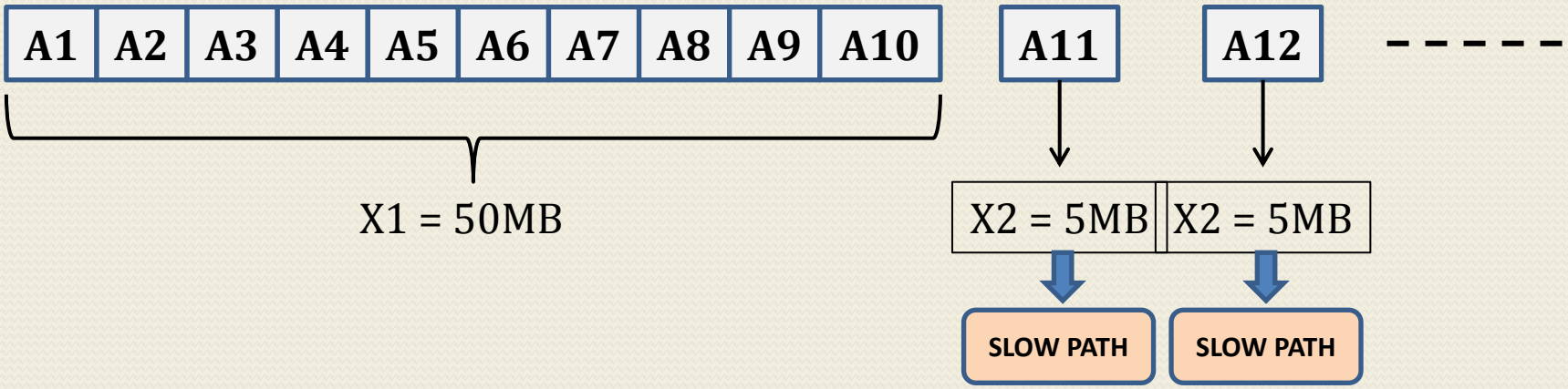
Zone:	Normal
Order	**Fragmentation[%]**
0	0.00%
1	0.80%
2	1.60%
3	20.50%
4	54.00%
5	76.50%
6	86.30%
7	89.10%
8	89.80%
9	95.30%
10	96.80%
Overall	**55.52%**

- **After many application launch fragmentation level becomes: 75.15%**
- **With ION memory shrinker, fragmentation level decreases to: 55.52%**

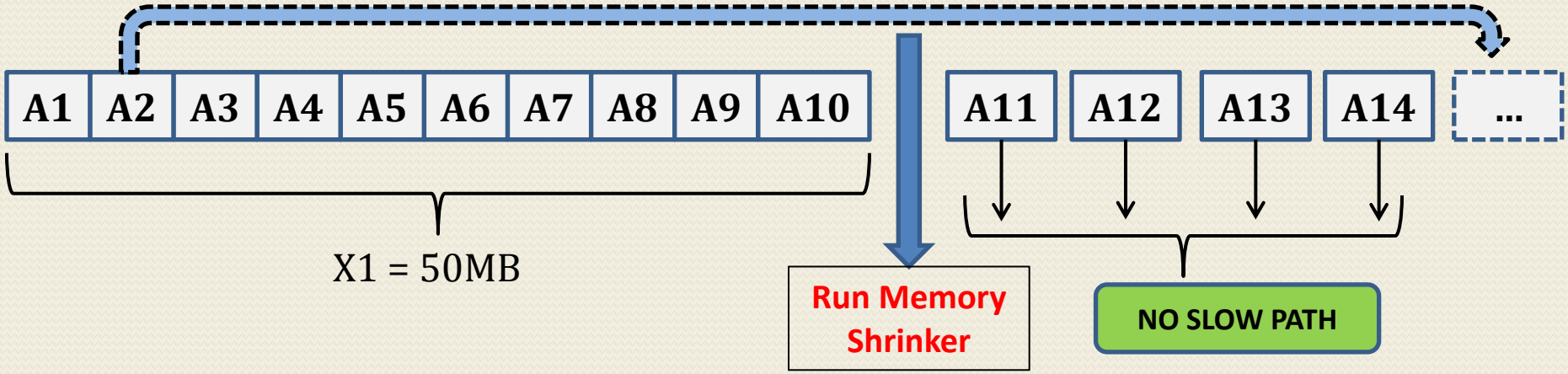
SUMMARY

Initial Free Memory = 50MB, Reclaimable memory = 150MB

Existing Approach:



New Approach:



CONCLUSION

- It can be developed as a system tool and invoked from user or kernel space.
- It can help in restoring the memory accumulated during initial boot-up, which may not be useful later.
- It can also help in finding out how much of the total memory can actually be reclaimed for each orders.
- It can help in allowing new application to launch without killing existing applications.
- This technique is already used in hibernation. Similarly for mobiles it can be used during system suspend.
- Similar technique is also used in memory cgroups in the name of force_reclaim and force_empty.
- It is more effective after heavy file transfer which make the system heavily fragmented. The caches accumulated here can be reclaimed.
- Memory shrinker patches and utilities developed here will be soon shared in the mainline for further review and improvements.

Thank You!

Questions??????