Premise:

The worst case for cache conflict misses is always worse for way based partitioning than it is for set based partitioning.

Simple Case:

There are no conflict misses in a fully associative cache. On the other hand, a direct mapped cache with the same cache size; will give worst performance in terms of conflict misses.

A more detailed case study:

Consider the following cache:

Size = 2 MB = 2048 KB = 512 Pages

Ways = 16

Way Size = 2 ^ 17 B = 128 KB = 32 Pages

Sets = 2 ^ 5 = 32

Set Size = 16 Pages

We want to create dedicated space for a benchmark whose working set size is slightly more than one fourth of the total cache size i.e.

Working set size = 132 Pages

We create a dedicated partition for this benchmark equal to one fourth of the cache size. Using set partitioning:

Number of sets for partition = 8

Using way partitioning:

Number of ways for partition = 4

Lets assume that pages are allocated uniformly to the benchmark such that the cache space is evenly utilized. Moreover, lets assume that the benchmark has perfectly sequential memory access pattern in which all memory access are made at the granularity of cache block size.

For this deterministic case, we will have the following utilization of the cache space:

Ways	0	7	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S00	0 129	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120
S01	1 130	9	..	..	..	..	..	..	..	..	..	..	..	..	..	121
S02	2 131	10	..	..	..	..	..	..	..	..	..	..	..	..	..	122
S03	3 132	11	..	..	..	..	..	..	..	..	..	..	..	..	..	123
S04	4	12	..	..	..	..	..	..	..	..	..	..	..	..	..	124
S05	5	13	..	..	..	..	..	..	..	..	..	..	..	..	..	125
S06	6	14	..	..	..	..	..	..	..	..	..	..	..	..	..	126
S07	7	15	23	31	39	47	55	63	71	79	87	95	103	111	119	127

Ways	0	1	2	3
S00	0 128	32	64	96
S01	1 129	…	…	…
S02	2 130	…	…	…
S03	3 131	…	…	…
S04	4 132	…	…	…
S05	5	…	…	…
S06	6	…	…	…
S07	7	…	…	…
S08	8	…	…	…
S09	9	…	…	…
S10	10	…	…	…
S11	11	…	…	…
S12	12	…	…	…
S13	13	…	…	…
S14	14	…	…	…
S15	15	…	…	…
S16	16	…	…	…
S17	17	…	…	…
S18	18	…	…	…
S19	19	…	…	…
S20	20	…	…	…
S21	21	…	…	…
S22	22	…	…	…
S23	23	…	…	…
S24	24	…	…	…
S25	25	…	…	…

Ways	0	1	2	3
S26	26	…	…	…
S27	27	…	…	…
S28	28	…	…	…
S29	29	…	…	…
S30	30	…	…	…
S31	31	63	95	127

Conclusion:

In the worst case of both set partitioning and way partitioning, the performance is going to be the same i.e. 100% cache misses

Hypothesis:

On average, way partitioning is worse than set partitioning

Scenario:

Consider a benchmark with working set size which is equal to the cache partition size:

Working set size = 128 Pages

Lets assume that the benchmark has a random memory access pattern in which memory references are made at the granularity of cache blocks i.e. 64 bytes. Let us also assume a random page allocation policy. Such a policy would make some of the cache sets to be more heavily populated than others. In such a case, there will be (on average) more conflict misses in the case of way partitioning as compared to the case of set partitioning.

Experiment:

Create the cache using associative arrays

Allocate pages at random and map them to cache sets

Assume LRU replacement policy

Generate random memory requests for cache blocks and keep track of references / misses

Repeat the experiment multiple times and plot the PDF of cache miss-rate for set partitioning vs way partitioning

Expected Outcome:

The miss-rate curve for way partitioning will be worse

Simplification:

Assume that the cache consists of 256 blocks. It has 16 sets and 16 ways

64 blocks are allocated to the benchmark using:

Way partitioning : 16 sets x 4 ways

Set partitioning : 4 sets x 16 ways