Outline

Profiling Result 1
- Flat Combining 2
- CCSynch 3
- FC-PQ-BHeap 4
- Mutex 5

Code Change 6
- Shared Counter 7

Flat Combining

Function	CPU Time	Clockticks	Instructions Retired	CPI Rate	Module
lock	44.490s	1.11×10^{11}	1.30×10^{9}	85.489	dlock
bench code	4.682s	1.17×10^{10}	1.27×10^{10}	0.918	dlock
[vmlinux]	0.069s	1.49×10^{8}	6.27×10^{7}	2.368	vmlinux
thread::current	0.037s	9.68×10^{7}	1.98×10^{7}	4.889	dlock

Function	Retiring	Front-End Bound	Bad Speculation	Back-End Bound
lock	0.90%	0.10%	0.00%	99.10%
bench code	30.90%	0.30%	0.80%	68.00%
[vmlinux]	34.30%	57.20%	11.40%	0.00%
thread::current	87.80%	100.00%	0.00%	0.00%

Profiling Result

Flat Combining

CCSynch

FC-PQ-BHeap

Mutex

Code Change

Shared Counter

Reason for `blackbox`

Reason for the change

CCSynch

Profiling Result
Flat Combining
CCSynch
FC-PQ-BHeap
Mutex
Code Change
Shared Counter
Reason for `blackbox`
Reason for the change

FC-PQ-BHeap

Profiling Result
Flat Combining
CCSynch
FC-PQ-BHeap
Mutex
Code Change
Shared Counter
Reason for `blackbox`
Reason for the change

Mutex

Profiling Result
Flat Combining
CCSynch
FC-PQ-BHeap
Mutex
Code Change
Shared Counter
Reason for `blackbox`
Reason for the change

Outline

Profiling Result 1

- Flat Combining 2
- CCSynch 3
- FC-PQ-BHeap 4
- Mutex 5

Code Change 6

- Shared Counter 7

Shared Counter

1. Remove `blackbox` for accessing the data
2. Change the blackbox position

```diff
- while black_box(loop_limit) > 0 {
-   *data += 1;
- }
+ while loop_limit > 0 {
+   *black_box(&mut *data) += 1;
+   loop_limit -= 1;
+ }
```

Profiling Result
Flat Combining
CCSynch
FC-PQ-BHeap
Mutex
Code Change
Shared Counter
Reason for `blackbox`
Reason for the change

Shared Counter

Reason for `blackbox`

1. `loop_limit` => the length of **Critical Section**.
2. Compiler will optimize the code to something like `*data += loop_limit;`, which will make varying the `loop_limit` not affecting the length of **Critical Section**.

Reason for the change

1. I want to mimic more access to the shared variable (hopefully something like `inc (rax)` in assembly).
2. The previous version contains too much overhead for doing the loop.