Michael Morris

ECE 2230-1

MP6 Performance Analysis

12/4/2023

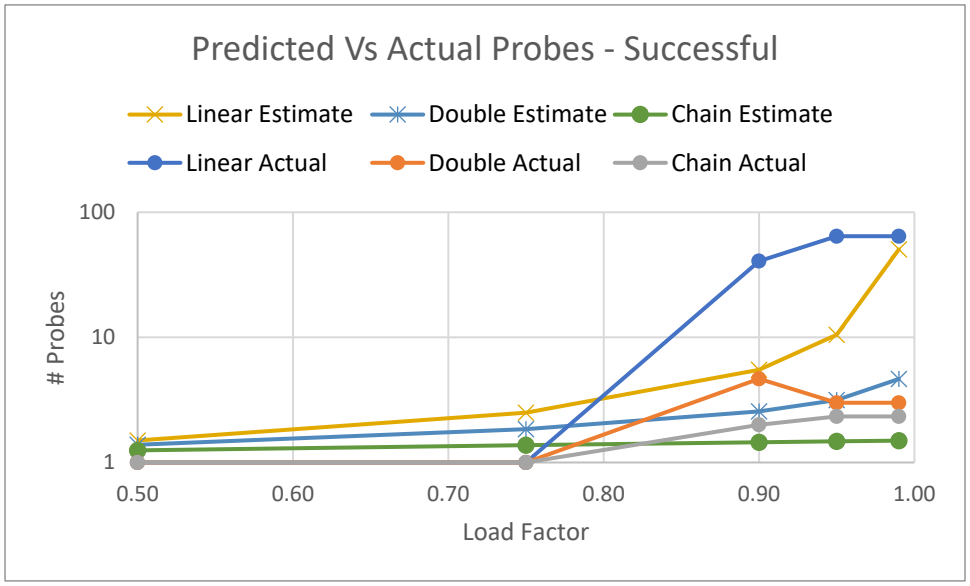
1

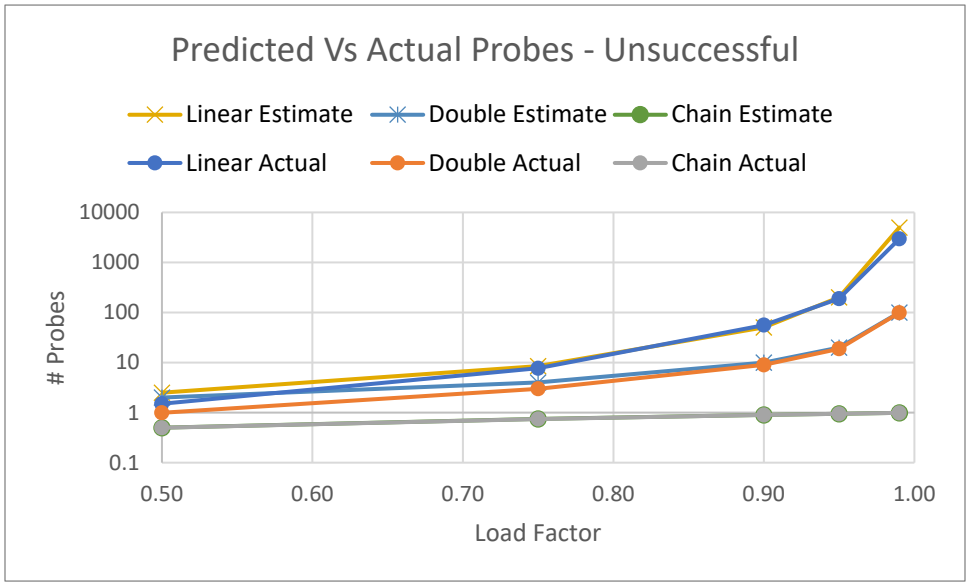
		Predicted		Actual		
	Load	**Successful**	**Unsuccessful**	**Successful**	**Unsuccessful**	**Avg Success. Probes**
Linear	0.99	50.5	5000.5	64.3333	2964.53	43.595
	0.95	10.5	200.5	64.3333	189.219	9.66169
	0.90	5.5	50.5	40.6667	55.9097	4.77955
	0.75	2.5	8.5	1	7.70367	1.52295
	0.50	1.5	2.5	1	1.50388	0.503876
Double	0.99	4.65169	100	3	98.9107	3.6866
	0.95	3.1534	20	3	18.8473	2.18667
	0.90	2.55843	10	4.6667	8.99742	1.57383
	0.75	1.84839	4	1	2.98566	0.85494
	0.50	1.38629	2	1	0.99172	0.387756
Chain	0.99	1.495	0.99	2.3333	0.989299	0.494598
	0.95	1.475	0.95	2.3333	0.946197	0.475185
	0.90	1.45	0.9	2	0.897214	0.449791
	0.75	1.375	0.75	1	0.74441	0.376444
	0.50	1.25	0.5	1	0.4986	0.249756

Above 90% load factor, the linear probing method had an average measurement for successful searches that was greater than the predicted value. This discrepancy lessens as the load factor decreases, and at 75% the measured average number of probes for successful searches drops to 1. As the table becomes less congested, it becomes more likely that the hashed entries will be placed at their intended position. It should also be noted that the linear probing method had roughly the same predicted and measured average probes for unsuccessful searches.

Double hashing does not exhibit the same pattern as linear probing for successful searches, and measurements for both successful and unsuccessful searches generally match their predicted values. This is likely because when a collision occurs, the entry is rehashed and advanced to a different position, and that second position is likely to differ from the second position of other entries that collide with the same initial location. This creates a hash table that is more spread out and has less dense clusters.

Finally, the method of separate chaining also has predicted and measured values that are closely related. Unlike linear probing / double hashing, separate chaining contains a linked list for each table position. When collisions occur, the entry being placed travels down the list until it finds an available position (or its own key).





2

	Successful	
	Random	Sequential
Separate Chaining	2	0.271335
Double	4.6667	1.49022
Linear	40.6667	1750.33

	Unsuccessful	
	Random	Sequential
Separate Chaining	0.846171	0.85048
Double	5.71382	5.60728
Linear	22.8028	7835.47

The tables above show the average number of successful and unsuccessful searches by the retrieval driver for tables built with both random and sequential insertions. Separate chaining shows that both random and sequential insertion methods have a roughly consistent complexity class of O(1). This is also seen in successful tests using double hashing and sequential inserts. However, the remaining 3 double hashing tests (successful with random inserts, and unsuccessful with random / sequential inserts) resembles a complexity class of O(log(n)). Linear probing in all tests is more susceptible to primary clustering, and the results are significantly delayed from the previous tests. This is most apparent in both successful and unsuccessful tests with sequential inserts. As the list grows longer, he location at which a new entry can be inserted becomes further away, on average.

3

		%empty	Measured, Avg		Expected		After Rehash		
	Time	marked del	Successful	Unsuccessful	Successful	Unsuccessful	Successful	Unsuccessful	Time
50000	54.718	88.9006	12.0151	314.229	5.43502	49.2087	5.43605	46.0539	26.041
100000	944.719	98.2587	15.4387	1129.79	5.559461	52.41	5.73806	51.04	62.019
200000	3378.75	99.9536	19.4586	8874.86	5.56703	51.8496	5.60789	47.5687	93.968

As the number of trials increases from 50k to 100k and finally to 200k, the percent of empty position that have been marked as deleted increased from 88.9% to 99.95%. The number of probes for successful and unsuccessful searches also increases significantly above the expected value. After rehashing the tables, all tests exhibited probe numbers for successful and unsuccessful searches at the expected values.

Separate chaining does not experience these performance decreases as it does not have the same issues with deleting nodes. When a linked list is created at each node, deleting a node is accomplished by changing a couple pointers within the list around. When a search or insertion occurs at the same location, the entire list is scanned for duplicates and if none are found, the new node is placed at the back of the list. Contrasting this is the open addressing method which deals with collisions by continuing down the hash table until an available position is found for the new node. Deleting a node, however, causes all downstream nodes that previously collided with the deleted node to be unfindable without also conveying the now-empty node was actually deleted and searches / insertions should continue

beyond until an empty and non-deleted node is located. As more nodes are deleted, this causes significant delays as more nodes must be checked for the same operation.

4

[MB]	Table Size		
	65537	655373	1655377
Linear	2.62	22.50	56.21
Double	2.62	22.50	56.21
Chain	2.97	26.00	65.05

As we can see, at 65537 trials, separate chaining uses roughly 13.36% more memory than linear probing / double hashing methods. For 655373 and 1655377 trials, this increases to approximately 15.57%