Compare Self Increment

Hao Deng

October 2023

1 Introduction

In this assignment, we want to compare ++i and i++ in the aspect of time cost. Specifically, we conduct experiments on the following data type in c++:

1. int32_t
2. int64_t
3. uint32_t
4. uint64_t
5. float
6. double

For each of data type above, we initiate a variable i, iterate i++ and ++i for 1,000,000,000 times, and record the cost of time. Then, we increase the number of iterations to 2,000,000,000 for validation, as we expect increasing number of iterations should make the difference in cost of time more obvious.

2 Experiment Result

	int32_t	int64_t	uint32_t	uint64_t	float	double
i++	1.814s	1.84s	1.931s	1.798s	2.439s	2.447s
++i	1.921s	1.758s	2.009s	1.834s	2.409s	2.421s

Table 1: Experiment 1 with 1,000,000,000 Iterations

	int32_t	int64_t	uint32_t	uint64_t	float	double
i++	1.63487s	1.61803s	1.61033s	1.691s	2.297s	2.278s
++i	1.629s	1.62375s	1.621s	1.664s	2.272s	2.273s

Table 2: Experiment 2 with 1,000,000,000 Iterations

	int32_t	int64_t	uint32_t	uint64_t	float	double
i++	2.107s	1.84503s	1.73646s	1.76154s	2.36107s	2.33805s
++i	1.77012s	1.87006s	1.76105s	1.765s	2.33404s	2.35251s

Table 3: Experiment 3 with 1,000,000,000 Iterations

	int32_t	int64_t	uint32_t	uint64_t	float	double
i++	3.51s	3.479s	3.46863s	3.51463s	4.7737s	4.79662s
++i	3.545s	3.71774s	3.47994s	3.49362s	4.77167s	4.88403s

Table 4: Experiment 1 with 2,000,000,000 Iterations

	int32_t	int64_t	uint32_t	uint64_t	float	double
i++	3.66952s	3.47606s	3.4491s	3.51314s	4.77913s	4.73401s
++i	3.46276s	3.4686s	3.51654s	3.4075s	4.67226s	4.85311s

Table 5: Experiment 2 with 2,000,000,000 Iterations

	int32_t	int64_t	uint32_t	uint64_t	float	double
i++	3.67897s	3.50403s	3.94903s	3.782s	4.807s	5.138s
++i	3.49355s	3.61006s	3.775s	3.543s	4.85103s	4.905s

Table 6: Experiment 3 with 2,000,000,000 Iterations

3 Analysis

From the tables above, we notice that costs of time for i++ and ++i are about the same for each data type. We can't tell which is better from the tables above.
After searching online, we found a plausible explanation to this result. Theoretically, ++i can never be worse than i++ because i++ potentially make a copy of i. However, modern compiler can optimize the codes most of the time. Therefore, the result we obtained showed little difference between i++ and ++i. If we want to perform further experiments, we have to find a way to prevent compiler optimizing the codes automatically.
Notice that float and double cost more time than int and uint and that there isn't a significant difference between the costs of time for 32-bit int and 64-bit int.