Отчёт по задаче 2
Подсчёт количества cache miss для операции матричного умножения в зависимости от порядка итерирования

	L1 load	L1 store	L1 cache	L2 load	L2 store	L2 cache	Theor	Theor/L1cache	Theor/L2cache
0 | ijk	38320	7	64711	*297*	6	*8628*	1250000	19,3166540465	144,877144182
1 | ikj	13501	7	62359	*427*	5	*6284*	500000	8,01808880835	79,5671546785
2 | kij	35844	6	70209	*583*	7	*10544*	500000	7,12159409762	47,4203338392
3 | jik	38562	8	76661	*306*	8	*8092*	1250000	16,3055530191	154,473554128
4 | jki	15037	10	65078	*266*	5	*5913*	2000000	30,7323519469	338,23778116
5 | kji	*28126*	11	78740	*1022*	6	*16409*	2000000	25,4000508001	121,884331769

Некоторые значения(L1 load для опции 5 и полностью L2 load и L2 cache) выделены курсивом. Это значит, что значения, полученные для этих измерений очень сильно менялись между тестами: для L1 load от 22300 до 35971, для некоторых значений L2 — в 2 - 4 раза. В таблице приведены усреднённые результаты.

Теоретические значения промахов стабильно выше реальных. Если сравнивать полученные результаты между собой, то для одного уровня кэша должны примерно совпадать количества промахов с опциями 0 и 3, 1 и 2, 4 и 5. Но и это не совпадает с теорией.

Теперь сравним полученные значения промахов кэша и времени выполнения. И если у теоретических значений промахов можно было выделить хотя бы частичные совпадения с измерениями времени, то для реальных значений какие-либо взаимосвязи найти не удаётся вообще.

Можно сделать вывод, что:
1) реальные промахи кэша существенно отличаются от теоритических
2) связать время выполнения с количеством промахов явно не удаётся.

Небольшое примечание к коду: при компиляции с -fsanitize=address не удалось добиться корректного выполнения. Но программа так же находила утечки памяти и при удалении из main() всех строчек кроме PAPI_library_init(PAPI_VER_CURRENT); (с PAPI_shutdown(); или без); программа без PAPI с этим ключом компилировалась и работала. Поэтому я склонен считать, что это проблема не программы, а локальной машины. Была попытка проверить это на полюсе, но там наиболее рабочим компилятором является xlc(gcc «ругался» на примерно половину ключей компиляции), а для него аналога -fsanitize=address я не нашёл.