

False sharing

Даниил Привалов. 381706-2

Описание проблемы производительности

Термин false sharing означает доступ к разным объектам в программе, разделяющим один и тот же блок кэш-памяти. False sharing в многопоточном приложении, когда в одном блоке оказываются переменные модифицируемые из разных потоков, ведет к снижению производительности и увеличению нагрузки на Cache coherence механизмы.

Задача приводящая к такой проблеме

Необходимо подобрать какую-то многопоточную программу для работы в разных потоках над отдельными частями структуры(класса).

Данная структура:

```
uint32_t d1;  
uint32_t d2;  
uint32_t d3;
```

В моем случае я просто модифицирую несколько таких структур параллельно по частям и суммирую все в результирующую структуру, такого же вида.

```
for (size_t i = 0; i < 20; i++)  
{  
    my_data[i].d1++;  
    other_data.d1 += my_data[i].d1 * (my_data[i].d1 + 1);  
}
```

Чтобы получить эту проблему, мне пришлось самому отключить выравнивание структуры командой `#pragma pack(push, 1)`.

Результаты тестирования


40 секунд на не выровненных данных против 15 на выровненных.

VTune показывает проблему и дает хорошую подсказку по ее исправлению:

Elapsed Time^②: 38.482s

Clockticks:	274,104,600,000	
Instructions Retired:	189,536,600,000	
CPI Rate ^② :	1.446	🚩
MUX Reliability ^② :	0.987	
➤ Retiring ^② :	29.4%	of Pipeline Slots
➤ Front-End Bound ^② :	16.6%	of Pipeline Slots
➤ Bad Speculation ^② :	11.4%	🚩 of Pipeline Slots
⌵ Back-End Bound ^② :	42.5%	🚩 of Pipeline Slots
⌵ Memory Bound ^② :	32.6%	🚩 of Pipeline Slots
➤ L1 Bound ^② :	8.9%	of Clockticks
L2 Bound ^② :	0.0%	of Clockticks
➤ L3 Bound ^② :	0.4%	of Clockticks
➤ DRAM Bound ^② :	6.3%	of Clockticks
⌵ Store Bound ^② :	39.6%	🚩 of Clockticks
Store Latency ^② :	41.0%	🚩 of Clockticks
False Sharing ^② :	14.3%	🚩 of Clockticks
Split Stores ^② :	0.0%	of Clockticks
DTLB Store Overhead ^② :	0.0%	of Clockticks
➤ Core Bound ^② :	9.9%	of Pipeline Slots

После выравнивания при помощи `alignas(CACHE_LINE_SIZE)` или отключения директивы `#pragma pack(push, 1)` мы получаем совершенно другую картину работы программы:

⌵ Elapsed Time [?] : 15.768s	
Clockticks:	111,141,800,000
Instructions Retired:	198,336,600,000
CPI Rate [?] :	0.560
MUX Reliability [?] :	0.978
⌵ Retiring [?] :	70.2%  of Pipeline Slots
⌵ Front-End Bound [?] :	12.7% of Pipeline Slots
⌵ Bad Speculation [?] :	0.0% of Pipeline Slots
⌵ Back-End Bound [?] :	17.0% of Pipeline Slots
⌵ Memory Bound [?] :	4.0% of Pipeline Slots
⌵ L1 Bound [?] :	6.6% of Clockticks
L2 Bound [?] :	0.1% of Clockticks
⌵ L3 Bound [?] :	0.0% of Clockticks
⌵ DRAM Bound [?] :	0.2% of Clockticks
⌵ Store Bound [?] :	0.2% of Clockticks
Store Latency [?] :	22.7% of Clockticks
False Sharing [?] :	0.0% of Clockticks
Split Stores [?] :	0.0% of Clockticks
DTLB Store Overhead [?] :	0.0% of Clockticks
⌵ Core Bound [?] :	13.0% of Pipeline Slots

Все же есть некоторые проблемы, но это потому что я пытался нагрузить потоки вычислениями и так же использовал сразу много структур для работы. Также у меня шло сплошное умножение, может в этом дело:

⌵ Retiring [?] :	70.2%  of Pipeline Slots
⌵ General Retirement [?] :	70.1%  of Pipeline Slots
⌵ FP Arithmetic [?] :	0.0% of uOps
Other [?] :	100.0%  of uOps

Закключение

Была разобрана проблема False sharing, сейчас компилятор не дает попасться на нее, возможно попасть на нее только если сам захочешь где-то упаковать структуру плотнее и не учтешь это при дальнейшем распараллеливании программы.

Параметры системы

CPU параметры:

- Name: Intel(R) Core(TM) Processor code named Broadwell (i5 – 5200U)
- Frequency: 2.2 GHz
- Logical CPU Count: 4

RAM: 4 GB

OS: Windows 10

Compiler: VS 2017