Deutsche Bank

Cache miss optimization in ultra low latency applications

23-24 October 2014

Евгений Шевкопляс
eugene.shevkoplyas@db.com

Passion to Perform

2014 CEE-SEC(R) Software Engineering Conference in Russia

Как написать быстрое приложение?

- Использовать алгоритмы с лучшей сложностью
- Уметь находить места, которые можно ускорить
- Уметь замерять различные части программы

Как измерить время?

— Time stamp counter

```
uint64_t rdtsc()
{
    uint32_t a,d;
    asm volatile("rdtsc" : "=a" (a), "=d" (d));
    return ((uint64_t)a) | (((uint64_t)d)<<32;
}
```

Время: 7 nsec (24)

Примеры:

```
std::string s = “1234567890”;
CHECKPOINT(“std::string");
char buf[32];
sprintf(buf, "%d", 1234567890);
CHECKPOINT("sprintf");
std::stringstream sstrm;
sstrm << 1234567890;
CHECKPOINT(“sstream");
struct timeval tm;
gettimeofday(&tm, NULL);
CHECKPOINT("gettimeofday");
rdtsc();
CHECKPOINT("rdtsc");
CHECKPOINT("nothing");
```

```
main.cpp(17) std::string: 0.579 us(2008)

main.cpp(20) sprintf: 1.18 us(4100)

main.cpp(23) sstream: 2.24 us(7776)

main.cpp(26) gettimeofday: 49.6 ns(172)

main.cpp(28) rdtsc: 13.8 ns(48)
main.cpp(29) nothing: 6.92 ns(24)
```

Пример 1: “Наведи порядок!”

Простая реализация, или прототип

```
std::vector<double> values;
double result = 0.0;
int N = 10000000;

for (size_t i = 1; i < N; ++i) {
    result += values[i-1]*values[i];
}
```

Простая реализация, или прототип

```
std::vector<double> values;
double result = 0.0;
int N = 10000000;

for (size_t i = 1; i < N; ++i) {
    result += values[i-1]*values[i];
}
```

Время: 28 msec

Добавим проверку на неинициализированные значения

```
Double operator * (const Double & d) const
{
        Double result;
        if (!m_empty && !d.m_empty) {
            result.m_value = m_value * d.m_value;
            result.m_empty = false;
        } else {
            print_error();
        }
        return result;
 }

void Double::print_error() const
{
    std::stringstream sstrm;
    sstrm  << __FILE__ << ":" << __LINE__
      << “ Uninitialized value!" << std::endl;
    std::cout << sstrm.str() << std::endl;
}
```

Добавим проверку на неинициализированные значения

```
Double operator * (const Double & d) const
{
        Double result;
        if (!m_empty && !d.m_empty) {
            result.m_value = m_value * d.m_value;
            result.m_empty = false;
        } else {
            print_error();
        }
        return result;
 }

void Double::print_error() const
{
    std::stringstream sstrm;
    sstrm  << __FILE__ << ":" << __LINE__
      << “ Uninitialized value!" << std::endl;
    std::cout << sstrm.str() << std::endl;
}
```

Время: 30.1 msec (+2.1 msec)

Такой вариант написать быстрее

```
Double operator * (const Double & d) const
{
      Double result;
      if (!m_empty && !d.m_empty) {
          result.m_value = m_value * d.m_value;
          result.m_empty = false;
      } else {
          std::stringstream sstrm;
          sstrm  << __FILE__ << ":" << __LINE__
            << “ Uninitialized value!" << std::endl;
          std::cout << sstrm.str() << std::endl;
      }
      return result;
}
```

Такой вариант написать быстрее

```
Double operator * (const Double & d) const
{
	Double result;
	if (!m_empty && !d.m_empty) {
	    result.m_value = m_value * d.m_value;
	    result.m_empty = false;
	} else {
	    std::stringstream sstrm;
	    sstrm  << __FILE__ << ":" << __LINE__
	      << “ Uninitialized value!" << std::endl;
	    std::cout << sstrm.str() << std::endl;
	}
	return result;
}
```

Время: 40.3 msec (+12.3, было +2,1)

Такой вариант написать быстрее

```
Double operator * (const Double & d) const
{
	Double result;
	if (!m_empty && !d.m_empty) {
	    result.m_value = m_value * d.m_value;
	    result.m_empty = false;
	} else {
	    std::stringstream sstrm;
	    sstrm  << __FILE__ << ":" << __LINE__
	      << “ Uninitialized value!" << std::endl;
	    std::cout << sstrm.str() << std::endl;
	}
	return result;
}
```

Время: 40.3 msec (+12.3, было +2,1)

В 6 раз больше накладных расходов!

Пример 2: “А Вася дома?”

Вызов функции, которая, возможно, ничего не делает

```
for (const auto & module : modules) {
        module.check_errors();
}
```

Вызов функции, которая, возможно, ничего не делает

```
for (const auto & module : modules) {
        module.check_errors();
}
```

```
void Module::check_errors() const
{
    if (m_errors == 0) {
        return;
    }

    // big business logic
    std::stringstream sstrm;
    sstrm << __FILE__ << ":" << __LINE__
          << " Our module has "
          << m_errors << " errors" << std::endl;
    std::cerr << sstrm.str() << std::endl;
}
```

Замеряем:

```
for (const auto & module : modules) {
        module.check_errors();
}
```

Время: 29.6 msec

Замеряем:

```
for (const auto & module : modules) {
      module.check_errors();
}
```

Время: 29.6 msec

Добавляем «лишнюю» проверку

```
for (const auto & module : modules) {
       if (module.has_errors()) {
          module.check_errors();
       }
}
```

Замеряем:

```
for (const auto & module : modules) {
      module.check_errors();
}
```

Время: 29.6 msec

Добавляем «лишнюю» проверку

```
for (const auto & module : modules) {
       if (module.has_errors()) {
          module.check_errors();
       }
}
```

Время: 6.6 msec

В 4.5 раза быстрее!

Пример 3: “В тесноте, да не в обиде”

Как расположить данные?

```
#define MAX_LEVEL 50


struct Book1
{
    size_t size;
    int prices[MAX_LEVEL];
    int volumes[MAX_LEVEL];
};
```

```
struct Book2
{
    struct Level
    {
        int price;
        int volume;
    };
    size_t size;
    Level levels[MAX_LEVEL];
};
```

Запись (N=10000, size = 1)

```
for (size_t i = 0; i < N; ++i) {
    auto & book = books1[i];
    for (size_t j = 0; j < size; ++j) {
        book.prices[j] = ++v;
        book.volumes[j] = ++v;
    }
}
```

```
for (size_t i = 0; i < N; ++i) {
    auto & book = books2[i];
    for (size_t j = 0; j < size; ++j) {
        auto & level = book.levels[j];
        level.price = ++v;
        level.volume = ++v;
    }
}
```

Запись (N=10000, size = 1)

```
for (size_t i = 0; i < N; ++i) {
    auto & book = books1[i];
    for (size_t j = 0; j < size; ++j) {
        book.prices[j] = ++v;
        book.volumes[j] = ++v;
    }
}
```

```
for (size_t i = 0; i < N; ++i) {
    auto & book = books2[i];
    for (size_t j = 0; j < size; ++j) {
        auto & level = book.levels[j];
        level.price = ++v;
        level.volume = ++v;
    }
}
```

Чтение

```
for (size_t i = 0; i < N; ++i) {
    const auto & book = books1[i];
    for (size_t j = 0; j < size; ++j) {
        sum += book.prices[j];
        sum += book.volumes[j];
    }
}
```

```
for (size_t i = 0; i < N; ++i) {
    const auto & book = books2[i];
    for (size_t j = 0; j < size; ++j) {
        const auto & level = book.levels[j];
        sum += level.price;
        sum += level.volume;
    }
}
```

Запись (N=10000, size = 1)

```
for (size_t i = 0; i < N; ++i) {
    auto & book = books1[i];
    for (size_t j = 0; j < size; ++j) {
        book.prices[j] = ++v;
        book.volumes[j] = ++v;
    }
}
```

115 usec

```
for (size_t i = 0; i < N; ++i) {
    auto & book = books2[i];
    for (size_t j = 0; j < size; ++j) {
        auto & level = book.levels[j];
        level.price = ++v;
        level.volume = ++v;
    }
}
```

42.9 usec

Чтение

```
for (size_t i = 0; i < N; ++i) {
    const auto & book = books1[i];
    for (size_t j = 0; j < size; ++j) {
        sum += book.prices[j];
        sum += book.volumes[j];
    }
}
```

```
for (size_t i = 0; i < N; ++i) {
    const auto & book = books2[i];
    for (size_t j = 0; j < size; ++j) {
        const auto & level = book.levels[j];
        sum += level.price;
        sum += level.volume;
    }
}
```

Запись (N=10000, size = 1)

```
for (size_t i = 0; i < N; ++i) {
    auto & book = books1[i];
    for (size_t j = 0; j < size; ++j) {
        book.prices[j] = ++v;
        book.volumes[j] = ++v;
    }
}
```

115 usec

```
for (size_t i = 0; i < N; ++i) {
    auto & book = books2[i];
    for (size_t j = 0; j < size; ++j) {
        auto & level = book.levels[j];
        level.price = ++v;
        level.volume = ++v;
    }
}
```

42.9 usec

Чтение

```
for (size_t i = 0; i < N; ++i) {
    const auto & book = books1[i];
    for (size_t j = 0; j < size; ++j) {
        sum += book.prices[j];
        sum += book.volumes[j];
    }
}
```

49.7 usec

```
for (size_t i = 0; i < N; ++i) {
    const auto & book = books2[i];
    for (size_t j = 0; j < size; ++j) {
        const auto & level = book.levels[j];
        sum += level.price;
        sum += level.volume;
    }
}
```

23 usec

Почему?

Callgrind

Профилируем

```
> valgrind --tool=callgrind --simulate-cache=yes ./a.out
==22360== Callgrind, a call-graph generating cache profiler
…
> callgrind_annotate --auto=yes ./callgrind.out.22360 > callgrind.annotate
```

Пример 1: «Наведи порядок!»

```
-------------------------------------------------------------------------------
-- Auto-annotated source: Double.h
-------------------------------------------------------------------------------
      Ir             Dr          Dw     I1mr     D1mr D1mw ILmr   DLmr DLmw
       .              .           .        .        .    .    .      .    .    Double operator * (const Double & d) const
       .              .           .        .        .    .    .      .    .    {
       .              .           .        .        .    .    .      .    .      Double result;
39,999,996    19,999,998       0        0   2,500,001 0        0 2,500,001    .      if (!m_empty && !d.m_empty) {
19,999,998    19,999,998       .        .        .    .    .      .    .           result.m_value = m_value * d.m_value;
       .              .           .        .        .    .    .      .    .           result.m_empty = false;
       .              .           .        .        .    .    .      .    .      } else {
       .              .           .        .        .    .    .      .    .        print_error();
       .              .           .        .        .    .    .      .    .      }
       .              .           .        .        .    .    .      .    .      return result;
       .              .           .        .        .    .    .      .    .    }
```

Пример 1: «Наведи порядок!»

```
-------------------------------------------------------------------------------
-- Auto-annotated source: Double.h
-------------------------------------------------------------------------------
      Ir             Dr          Dw     I1mr     D1mr D1mw lLmr   DLmr DLmw
 69,999,993          0  59,999,994        1        0    0    1      .    .    Double operator * (const Double & d) const
       .             .           .        .        .    .    .      .    .    {
       .             .           .        .        .    .    .      .    .       Double result;
39,999,996    19,999,998          0        0  2,500,001    0    0  2,500,001  .    if (!m_empty && !d.m_empty) {
19,999,998    19,999,998          .        .        .    .    .      .    .       result.m_value = m_value * d.m_value;
 9,999,999             .          .        .        .    .    .      .    .         result.m_empty = false;
       .             .           .        .        .    .    .      .    .    } else {
       .             .           .        .        .    .    .      .    .       std::stringstream sstrm;
       .             .           .        .        .    .    .      .    .       sstrm << "Invalid usage of double in "
       .             .           .        .        .    .    .      .    .       << __FILE__ << ":" << __LINE__ << std::endl;
       .             .           .        .        .    .    .      .    .       std::cout << sstrm.str() << std::endl;
       .             .           .        .        .    .    .      .    .    }
19,999,998           .           .        .        .    .    .      .    .    return result;
79,999,992  69,999,993           .        .        .    .    .      .    .  }
```

Пример 2: «А Вася дома?»

```
-------------------------------------------------------------------------------
-- Auto-annotated source: main.cpp
-------------------------------------------------------------------------------
      Ir            Dr          Dw  I1mr   D1mr  D1mw ILmr   DLmr  DLmw
20,000,000          .           .     .      .     .     .      .     .       for (const auto & module : modules) {
30,000,000  10,000,000  0     0    625,001  0     0    625,001   .          if (module.has_errors() != 0) {
       .            .          .     .      .     .     .      .     .            module.check_errors();
       .            .          .     .      .     .     .      .     .          }
       .            .          .     .      .     .     .      .     .       }
```

Пример 2: «А Вася дома?»

```
-------------------------------------------------------------------------------
-- Auto-annotated source: main.cpp
-------------------------------------------------------------------------------
      Ir             Dr        Dw      I1mr    D1mr  D1mw ILmr   DLmr  DLmw
 20,000,000          .         .         .       .     .    .      .     .       for (const auto & module : modules) {
 30,000,000          0    10,000,000     .       .     .    .      .     .           module.check_errors();
190,000,000 80,000,000 60,000,000 625,001    0     1    625,001    .    => Module.cpp:Module::check_errors()
      .              .         .       .       .     .    .      .     .       }


-------------------------------------------------------------------------------
-- Auto-annotated source: Module.cpp
-------------------------------------------------------------------------------
      Ir             Dr        Dw    I1mr     D1mr D1mw ILmr    DLmr DLmw
      .              .         .       .       .     .    .       .     .     void Module::check_errors() const
80,000,000          0   60,000,000    .       .     .    .       .     .     {
30,000,000 10,000,000 0         0   625,001  0     0       625,001    .     if (m_errors == 0) {
      .              .         .       .       .     .    .       .     .         return;
      .              .         .       .       .     .    .       .     .     }
      .              .         .       .       .     .    .       .     .     std::stringstream sstrm;
      .              .         .       .       .     .    .       .     .     sstrm << __FILE__ << ":" << __LINE__;
      .              .         .       .       .     .    .       .     .     std::cerr << sstrm.str() << std::endl;
80,000,002 70,000,000 0       2       0     0      2          .     .     }
```

Пример 3: «В тесноте, да не в обиде»

```
--------------------------------------------------------------------------------
-- Auto-annotated source: main.cpp
--------------------------------------------------------------------------------
    Ir     Dr     Dw I1mr D1mr D1mw ILmr DLmr DLmw
20,000      .      .    .    .    .    .    .    .    for (size_t i = 0; i < N; ++i) {
     .      .      .    .    .    .    .    .    .        auto & book = books2[i];
     .      .      .    .    .    .    .    .    .        for (size_t j = 0; j < size; ++j) {
     .      .      .    .    .    .    .    .    .            auto & level = book.levels[j];
     .      .      .    .    .    .    .    .    .            level.price = ++v;
40,000      0 20,000    0    0 10,000    .    .    .            level.volume = ++v;
     .      .      .    .    .    .    .    .    .        }
     .      .      .    .    .    .    .    .    .    }

20,000      .      .    .    .    .    .    .    .    for (size_t i = 0; i < N; ++i) {
     .      .      .    .    .    .    .    .    .        auto & book = books1[i];
     .      .      .    .    .    .    .    .    .        for (size_t j = 0; j < size; ++j) {
     .      .      .    .    .    .    .    .    .            book.prices[j] = ++v;
40,000      0 20,000    0    0 20,000    .    .    .            book.volumes[j] = ++v;
     .      .      .    .    .    .    .    .    .        }
     .      .      .    .    .    .    .    .    .    }
```

Пример 3: «В тесноте, да не в обиде»

```
--------------------------------------------------------------------------------
-- Auto-annotated source: main.cpp
--------------------------------------------------------------------------------
   Ir        Dr     Dw I1mr D1mr D1mw ILmr DLmr DLmw
20,000       .      .    .    .     .     .    .     for (size_t i = 0; i < N; ++i) {
   .         .      .    .    .     .     .    .        auto const& book = books2[i];
   .         .      .    .    .     .     .    .        for (size_t j = 0; j < size; ++j) {
   .         .      .    .    .     .     .    .           auto const& level = book.levels[j];
 10,000  10,000   0    0  10,000    .     .    .           sum += level.price ;
20,000  10,000    .    .    .     .     .    .           sum += level.volume;
   .         .      .    .    .     .     .    .        }
   .         .      .    .    .     .     .    .     }

20,000       .      .    .    .     .     .    .     for (size_t i = 0; i < N; ++i) {
   .         .      .    .    .     .     .    .        auto const& book = books1[i];
   .         .      .    .    .     .     .    .        for (size_t j = 0; j < size; ++j) {
 10,000  10,000   0    0  10,000    .     .    .           sum += book.prices[j] ;
20,000  10,000   0    0  10,000    .     .    .          sum += book.volumes[j];
   .         .      .    .    .     .     .    .        }
   .         .      .    .    .     .     .    .     }
```

Заключение

Оптимизация использования кеша
даёт значительное ускорение
без изменения сложности алгоритма.

Наш опыт ускорения:
с 7 микросекунд до 0.3 микросекунд

Спасибо!

Q&A

eugene.shevkoplyas@db.com