Нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий, математики и механики

Оптимизация вычисления формулы Блэка-Шоулза под архитектуру RISC-V

И.Н. Алпутов, Е.А. Панова, В.Д. Волокитин, И.Б. Мееров

Суперкомпьютерные дни в России 2024
23-24 сентября

Введение

- Известный бенчмарк – ***ценообразование опционов Блэка-Шоулза***
- Задача из области финансовой математики, представляет практический интерес
- Производительность ограничена подсистемой памяти (*memory-bound*)
- План:
  - Продемонстрировать классические техники оптимизации* под современные процессоры архитектуры x86 (Intel)
  - Применить их для современных CPU на базе архитектуры RISC-V
  - Оценить производительность, сравнить с архитектурой x86

*E. Panova, V. Volokitin, A. Gorshkov, I. Meyerov. Black-Scholes Option Pricing on Intel CPUs and GPUs: Implementation on SYCL and Optimization Techniques //Russian Supercomputing Days. – Cham : Springer International Publishing, 2022. – С. 48-62.

Опцион

- Задача – вычисление цены *европейского опциона колл*
- *Опцион* – контракт между двумя сторонами и
- Дает право купить *в будущем* у акцию по цене
- За это платит некоторую сумму
- – цена исполнения опциона
- – время экспирации опциона
- – цена опциона
- **Какова должна быть справедливая цена опциона?**

Модель Блэка-Шоулза*

- – цена акции в момент времени *t*
- – цена облигации в момент времени *t*
- – процентная ставка (считаем константой)
- – волатильность (считаем константой)
- – *Винеровский случайный процесс*, независимые приращения ,

*Black F., Scholes M. The pricing of options and corporate liabilities //Journal of political economy. – 1973. – Т. 81. – №. 3. – С. 637-654.

Формула Блэка-Шоулза

- – справедливая цена опциона
- – стартовая цена акции
- K – цена исполнения опциона
- T – время экспирации опциона
- σ – волатильность
- r – безрисковая процентная ставка
- – интегральная функция стандартного нормального распределения

Базовая версия на языке C++

```
const float SIG, R;  // volatility, interest rate

void GetOptionPrices (
  float *pT,   // maturity (input)
  float *pK,   // strike price (input)
  float *pS0,  // initial stock price (input)
  float *pC,   // option price (output)
  int N        // number of options
){
  for (int i = 0; i < N; i++)
  {
    float d1 = (std::log(pS0[i] / pK[i]) +
      (R + 0.5f * SIG * SIG) * pT[i]) / (SIG * std::sqrt(pT[i]));
    float d2 = d1 - SIG * std::sqrt(pT[i]);
    float erf1 = 0.5f + 0.5f * std::erf(d1 / std::sqrt(2.0f));
    float erf2 = 0.5f + 0.5f * std::erf(d2 / std::sqrt(2.0f));
    pC[i] = pS0[i] * erf1 - pK[i] * std::exp((-1.0f) * R * pT[i]) * erf2;
  }
}
```

Тип данных – *float (binary32)*

Структура массивов *(Structure of Arrays, SoA)*

$$\Phi(\quad) = \frac{1}{2} + \frac{1}{2} erf \frac{\quad}{\sqrt{2}}$$

Тестовая инфраструктура (x86)

Серверный узел x86	
CPU	2x Intel Xeon Silver 4310T (Ice Lake, 2.3 ГГц, 2x10 ядер, гипертрединг, AVX512)
RAM	64 ГБ, DDR4-2667
Операционная система	CentOS Linux 7
Компилятор	Intel oneAPI DPC++/C++ Compiler 2023.0.0

Оптимизации под архитектуру x86

1. **Векторные вычисления**
   - Библиотека OpenMP: *#pragma omp simd*
   - Мат. функции: *SVML (Short Vector Math Library)* из Intel Compiler
   - Ключ компиляции: *-march=icelake-server*
2. **Многопоточность**
   - OpenMP: *#pragma omp parallel for*
   - Статическое расписание
   - Ключ компиляции: *-fopenmp*
   - ***NUMA (Non-Uniform Memory Access)***

Non-Uniform Memory Access (NUMA)

- Два процессора на узле
- Память может быть физически выделена на одном процессоре, что даст 50% *медленного удаленного доступа*
- NUMA-friendly инициализация:

```
#pragma omp parallel for
for (int i = 0; i < N; i++) {
  // input arrays
  pT[i] = T0;  // T0, S0, K are constants
  pS0[i] = S0;
  pK[i] = K;
  // output array
  pC[i] = 0;
}
```

Пошаговая оптимизация под x86. Эксперименты

Версия	Время, с	*Ускорение*
Базовая версия	2,981	-
Векторизация	0,466	*6,4x*
Параллелизм	0,032	*14,6x*
Итого	**0,032**	***93,1x***

- опционов (1.5 ГБ)
- AVX512 + float => 8 элементов в zmm регистре, ускорение 6,4 раза
- 20 ядер на узле, ускорение 14,6 раз
- **Общее ускорение почти 2 порядка**
- **Производительность ограничена пропускной способностью кэша L3**

Тестовая инфраструктура (RISC-V)

Lichee Pi 4A	
CPU	TH1520 (T-Head, 1.85 ГГц, 4x XuanTie C910, RVV 0.7.1)
RAM	16 ГБ, LPDDR4X
Операционная система	Debian GNU/Linux 12
Компилятор	GNU 8.4.0

Оптимизация под архитектуру RISC-V

1. Многопоточность

Версия	Время, с	*Ускорение*
Базовая версия	15,196	-
Параллелизм	**3,690**	***4,12x***

2. Векторизация

- Векторные интринсики, lmul=4
- Длина векторного регистра 128 бит, ожидаемое ускорение
- **Нет векторных версий мат. функций**
- Выгрузка из векторных регистров перед вычислением мат. функций – дорого (замедление относительно скалярной версии)

Реализация математических функций

- Необходимы векторные функции ***exp***, ***log***, ***sqrt***, ***erf***
- ***sqrt*** для `binary32` – есть функция-интринсик
- Для остальных функций – полиномиальная аппроксимация
- Ограниченный набор аргументов из предметной области
  - Не рассматриваем особые значения аргументов (NaN, Inf, особые точки)
  - Считаем, что нет переполнений
- Не требуется высокая точность вычислений
- Векторные мат. функции реализованы для RISC-V и x86 (AXV512)

Реализация математических функций

- Экспонента (***exp***)
  - Источник: *Muller J. M. Elementary functions. – Birkhũser Boston, 2006.*
  - Ошибка 1 бит мантиссы
- Функция ошибок (***erf***)
  - Источник: *Abramovitz M., Stegun I. A. Handbook of Mathematical Functions, Nat //Bureau of Standards, NY. – 1964. – T. 832.*
  - Теоретическая ошибка в 7 десятичном знаке после запятой
- Натуральный логарифм (***log***)
  - Максимальная ошибка

Эксперименты

Версия		x86		RISC-V	
		Время, с	***Ускорение***	**Время, с**	***Ускорение***
Базовая версия (пользовательские мат. функции)		2,249	-	12,904	-
Векторизация (интринсики AVX512 или RVV 0.7.1)		0,350	***6,4x***	3,058	***4,22x***
Параллелизм	4 потока	0,120	***2,9x***	0,759	***4,02x***
	2x20 потоков	0,031	*11,3x*	-	-
Итого	**4 потока**	**0,120**	***18,7x***	**0,759**	***17,0x***
	2x20 потоков	**0,031**	***72,5x***	**-**	**-**

Выводы

- Бенчмарк – формула Блэка-Шоулза
- Была проведена оптимизация для архитектуры x86 (AVX512, 2x10 ядер)
- Векторизация, параллелизм на общей памяти – *ускорение 2 порядка*
- Использование векторизации на RISC-V затрудняется отсутствием векторных библиотек мат. функций
- Была написана пользовательская реализация мат. функций с точностью 5 десятичных знаков после запятой
- **Время работы для рассмотренных процессоров x86 и RISC-V сильно отличается (25 раз)**
- **Однако полученное ускорение от векторизации и параллелизма на 4 ядрах для процессоров x86 и RISC-V сопоставимо**
- *Открытый исходный код: https://github.com/PanovaElena/BS_benchmark_riscv*

Литература

- https://github.com/PanovaElena/BS_benchmark_riscv
- Black F., Scholes M. The pricing of options and corporate liabilities //Journal of political economy. – 1973. – Т. 81. – №. 3. – С. 637-654.
- E. Panova, V. Volokitin, A. Gorshkov, I. Meyerov. Black-Scholes Option Pricing on Intel CPUs and GPUs: Implementation on SYCL
  and Optimization Techniques //Russian Supercomputing Days. – Cham : Springer International Publishing, 2022. – С. 48-62.
- Hager G., Wellein G. Introduction to high performance computing for scientists and engineers. – CRC Press, 2010.
- O Mahony A., Hanzon B., Popovici E. The role of FPGAs in Modern Option Pricing techniques: A survey //Electronics. – 2024. – Т. 13. – №. 16. – С. 3186.