Московский физико-технический институт (государственный университет)
Физтех-школа радиотехники и компьютерных технологий
Кафедра информатики и вычислительной техники

Выпускная квалификационная работа (бакалаврская работа)

Исследование вызовов математических функций, их профилирование и оптимизация компилятором LCC

Работу выполнил: Оберман И.Е.
Научный руководитель: Шампаров В.Е.

Москва, 2023

Математические функции

Тест	Затраты	Функция
527.cam4	9,73%	exp
554.roms	5,59%	exp
544.nab	15,15%	exp
416.gamess	4,09%	exp
459.GemsFDTD	3,61%	exp
538.imagick	12,93%	floor
538.imagick	12,83%	ceil
527.cam4	7,57%	log
544.nab	3,90%	log
526.blender	2,90%	sin, cos
554.roms	2,01%	log
527.cam4	2,72%	pow
554.roms	1,53%	pow

Результаты профилирования вызовов математических функций в тестах пакета SPEC CPU2017r

```
for(int t=0; t<(int)tris.size(); t++) {
    for (int i = 0; i < 3; ++i) {

        ...

        phi[0] = 2.*asin( (mvmFloat)(0.5* norm(u[1]-u[2]) )  );
        phi[1] = 2.*asin( (mvmFloat)(0.5* norm(u[0]-u[2]) )  );
        phi[2] = 2.*asin( (mvmFloat)(0.5* norm(u[0]-u[1]) )  );

        ...

        c[0] = (sinh*sin(h-phi[0]))/(sin(phi[1])*sin(phi[2]))-1.;
        c[1] = (sinh*sin(h-phi[1]))/(sin(phi[0])*sin(phi[2]))-1.;
        c[2] = (sinh*sin(h-phi[2]))/(sin(phi[0])*sin(phi[1]))-1.;

        ...

        s[0] = sqrt((float)(1.-c[0]*c[0]));
        s[1] = sqrt((float)(1.-c[1]*c[1]));
        s[2] = sqrt((float)(1.-c[2]*c[2]));

        ...

    }
}
```

Пример активного использования математических функций (SPEC CPU2017r 526.blender)

Математические функции

- Описаны в библиотеке math.h
- Часто вызываются в вычислительных задачах

Гипотеза: часто вызываются с одними и теми же значениями аргументов

- Стандартизированы
- Известно наличие/отсутствие побочных эффектов
- Результаты и поведение детерминированы

Вывод: многократное вычисление одного и того же результата

Следствие: достаточно вычислить 1 раз

Идея оптимизации

Если выявлено, что функция часто вызывается с 1 конкретным значением freq_value, то заменяем ее вызовы на вычисленный заранее результат retval

```
...some code...
res = sin(x);
...some code...
```

```
...some code...
if (x == freq_value)
    res = retval;
    // retval = sin(freq_value)
else
    res = sin(x);
...some code...
```

Цель работы

Исследовать необходимость, возможность и полезность применения оптимизации замены вызовов математических функций на проверку равенства их аргументов и часто повторяющихся значений и реализовать саму оптимизацию в оптимизирующем компиляторе lcc

Задачи:

- Анализ применимости предлагаемой оптимизации
- Реализация алгоритма предлагаемой оптимизации в оптимизирующем компиляторе lcc
- Проведение замеров влияния оптимизации на целевые программы

Поиск контекста применения

- **544.nab**: 15,15% на exp(x)
  из них 50% exp(0.0)

- **538.imagick** – *неприменима*
- **526.blender**: 2.9% на sin(x), cos(x)
  из них 64% sin(0.0)
  и 63% cos(0.0)
  Оптимизация неприменима из-за замены sin(x), cos(x) → sincos(x, &sin, &cos)

- **554.roms**: 2,01% на log(x)
  из них 18% log(86956.5)

Тест	Затраты	Функция
527.cam4	9,73%	exp
554.roms	5,59%	exp
544.nab	15,15%	exp
416.gamess	4,09%	exp
459.GemsFDTD	3,61%	exp
538.imagick	12,93%	floor
538.imagick	12,83%	ceil
527.cam4	7,57%	log
544.nab	3,90%	log
526.blender	2,90%	sincos
554.roms	2,01%	log
527.cam4	2,72%	pow
554.roms	1,53%	pow

Результаты профилирования вызовов математических функций в тестах пакета SPEC CPU2017r

Оценка эффекта на модельном тесте

```
#ifdef CHECK_OPT
    for( i = 0; i < ARR_LEN; i++ )
    {
        loc = src1[i];
        if ( loc == FREQ_VAL )
        {
            real_retval = RETVAL;
        } else
        {
            real_retval = MATH_FUNC( loc );
        }
        dst1[i] = loc + real_retval;
    }
#else
    for( i = 0; i < ARR_LEN; i++ )
    {
        loc = src1[i];
        real_retval = MATH_FUNC( loc );
        dst1[i] = loc + real_retval;
    }
#endif
```

На разработанном модельном тесте проверен эффект от предлагаемой оптимизации:

- MATH_FUNC – рассматриваемая математическая функция
- FREQ_VAL – частое значение аргумента
- RETVAL – возвращаемое значение для этого аргумента

Оценка эффекта на модельном тесте

Результаты модельного теста:

Функция	Минимальная доля FREQ_VAL, при которой оптимизация дает ускорение
acos	< 1,00%
acosh	< 1,00%
asin	< 1,00%
asinh	< 1,00%
atan	< 1,00%
atanh	< 1,00%
cos	< 1,00%
cosh	4,00%
exp	4,00%
log	< 1,00%
log10	< 1,00%
log2	< 1,00%
sin	< 1,00%
sinh	4,00%
sqrt	нет
tan	< 1,00%
tanh	32,00%

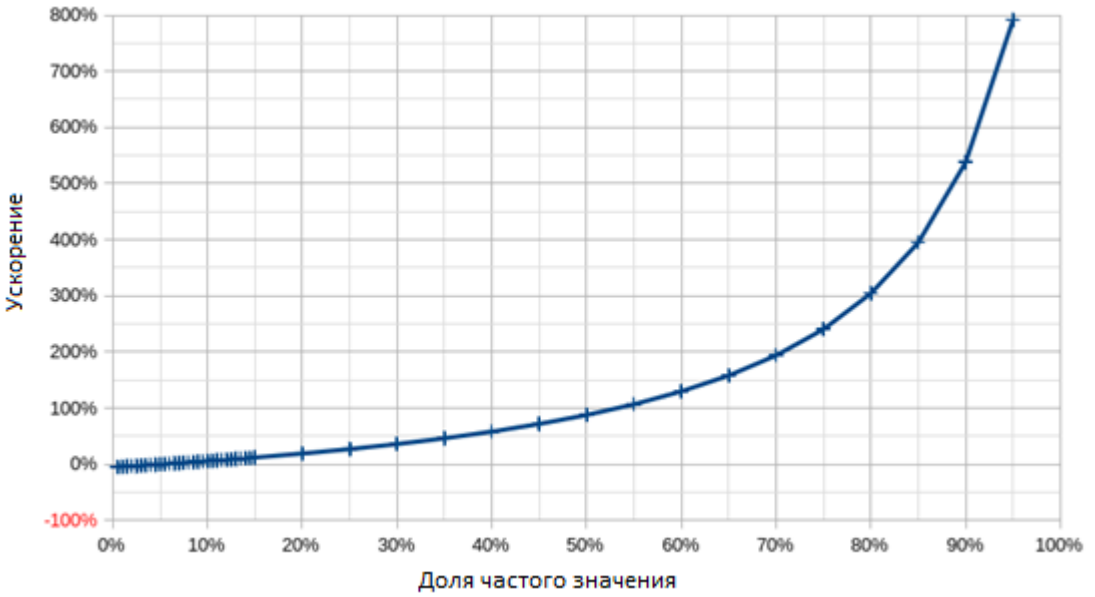


График зависимости ускорения исполнения модельного теста для функции exp(x) от доли значения 0.0 среди аргументов

Оценка эффекта на SPEC CPU

Из результатов модельного теста получаем следующие теоретические оценки ускорения:

- **544.nab**:
  - 15,15% времени на вычисление exp(x)
  - 50% из них - exp(0,0)
  - из модельного теста ускорение для exp(x) при 50% значений: +74,40%
  - следовательно, экономия времени: 6,49%
  - ускорение: +6,95%
- **554.roms**:
  - аналогично +0,36% → не выходит за пределы погрешности

Профилирование

До исполнения программы невозможно выявить самые частые значения аргументов математических функций

Для этого использован механизм **профилирования** - сбора интересующих нас характеристик работы программы при исполнении

```
...some code...
if (x == freq_value)
    res = retval;
    // retval = sin(freq_value)
else
    res = sin(x);
...some code...
```

Участок кода после оптимизации

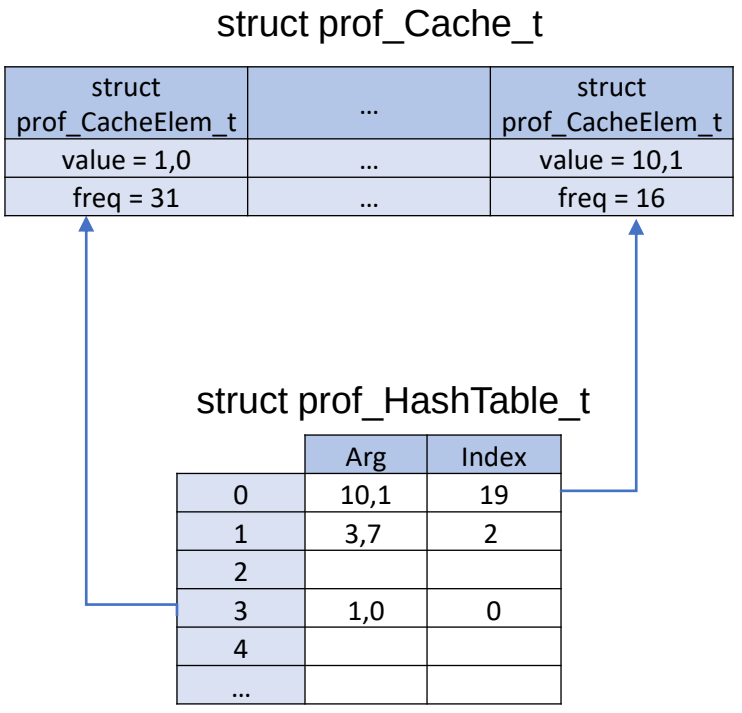
Профилирование

Для выделения самых частых значений была реализована структура данных кэш:

Внешнее устройство кэша:

- Элемент кэша – struct prof_CacheElem_t
  - значение элемента value
  - частота появления freq

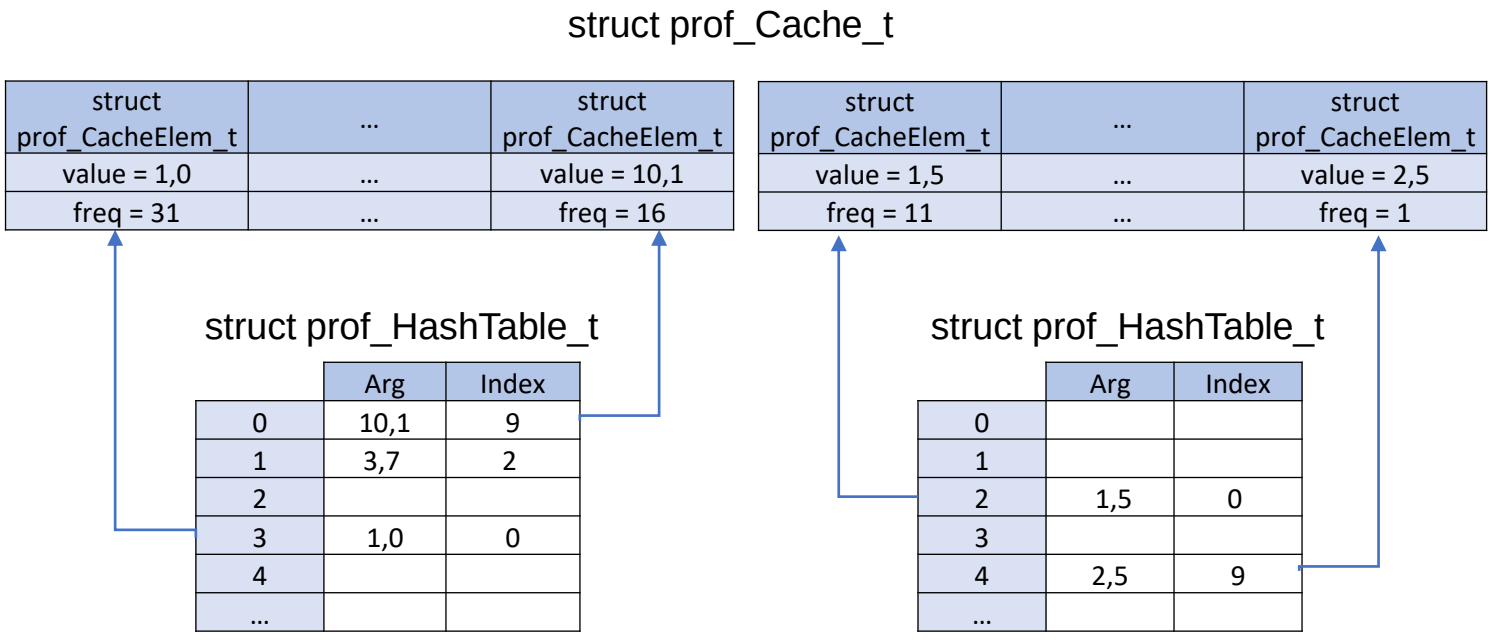
- Сам кэш – struct prof_Cache_t:
  - всего 20 элементов
  - массив для хранения
  - хэш-таблица «значение элемента – индекс массива»
  - политика замещения LFU



Профилирование

Внутреннее устройство кэша:

- 2 половины кэша по 10 элементов
- Очистка нижней половины раз в 2000 элементов для обработки антипаттернов
- Пример: …XYXYXY… при полностью заполненном кэше



Алгоритм профилирования

Запускается по опции –mathfunc с дополнительной опцией –mathprofile_gen

Для каждого вызова рассматриваемой математической функции вставить перед ее вызовом функцию запоминания аргумента в кэш:

1) Если кэш еще не создан – создать

2) Выполнить поиск элемента в обеих хэш-таблицах:
   - Если нашли, то увеличить его счетчик
     - Если счетчик стал больше, чем у элемента выше – поменять местами
   - Если не нашли – выполнить процедуру вставки
     - Если верхняя половина не полностью заполнена – вставить в ее конец
     - Иначе, если нижняя половина не полностью заполнена – вставить в ее конец
     - Иначе удалить последний элемент нижней половины и вставить на его место

3) При выходе из программы распечатать самые частые значения кэшей в файл, если выполняются следующие условия:
   - Если частота самого частого аргумента больше пороговой
   - Если общее количество вызовов данной функции больше 500

Алгоритм оптимизации

Запускается по опции –mathfunc с дополнительной опцией –mathprofile_use

1) Получить из профиля *рассматриваемые* функции, их частые значения аргументов и соответствующие возвращаемые значения

2) Для каждого вызова каждой *рассматриваемой* функции func и самого частого значения ее аргумента freq_val с соответствующим возвращаемым значением retval заменить этот вызов на следующий код:

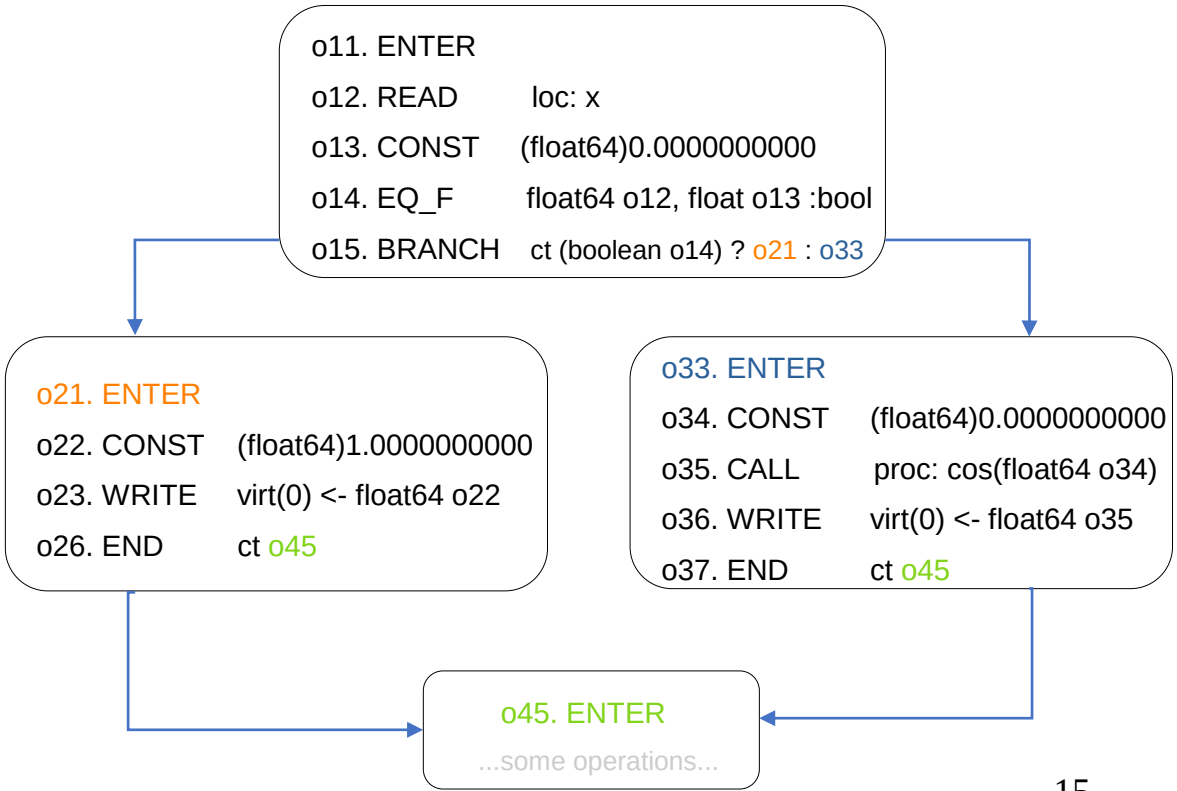
```
if (x == freq_val)
    res = retval;
else
    res = func(x);
```

Применение оптимизации

На промежуточном представлении EIR оптимизация выглядит следующим образом:

```
o11. ENTER
o13. READ       loc: x
o14. CALL       proc: cos(float64 o13)
o34. WRITE      virt(0) <- float64 o14
o26. END        ct o27
```

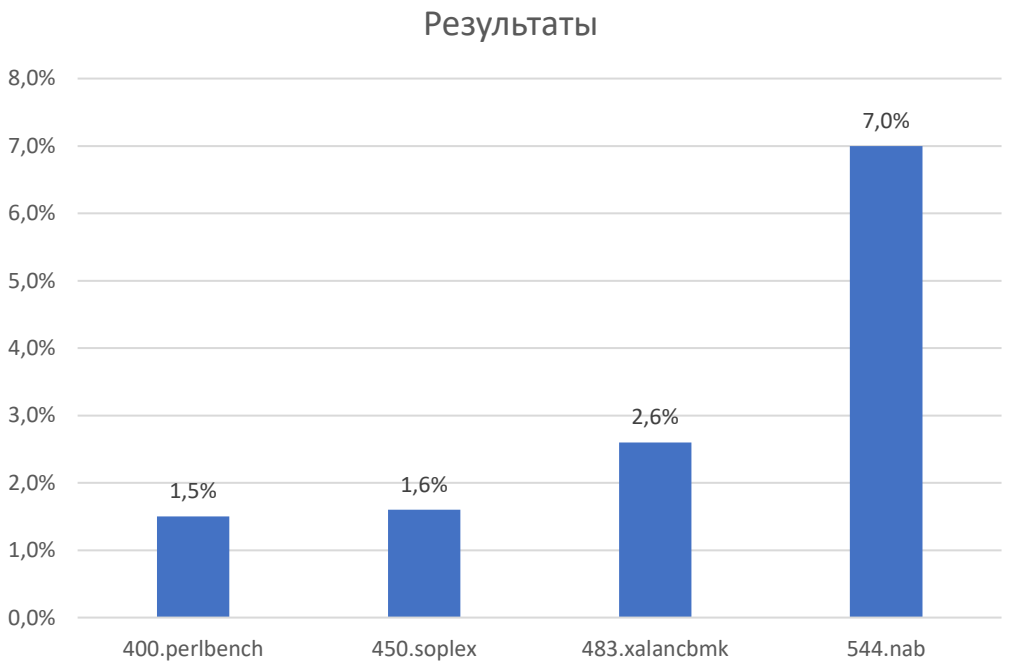
Участок графа потока управления (cfg) до применения оптимизации



Участок графа потока управления (cfg) после применения оптимизации

Проведение замеров

- SPEC CPU1995
  - не замедлились
- SPEC CPU2000
  - не замедлились
- SPEC CPU2006
  - 400.perlbench → +1,5%
  - 450.soplex → +1,6%
  - 483.xalancbmk → +2,6%
  - остальные не замедлились
- SPEC CPU2017r
  - 544.nab → +7,0%
  - остальные не замедлились



Результаты применения оптимизации на тестах пакета SPEC CPU

Результаты

- Найден контекст применения оптимизации замены вызовов математических функций на проверку равенства их аргументов и часто повторяющихся значений.
- Произведено теоретическое исследование эффекта от оптимизации, подтвердившее её перспективность.
- Удалось ускорить несколько задач:
  - 400.perlbench → +1,5%
  - 450.soplex → +1,6%
  - 483.xalancbmk → +2,6%
  - 544.nab → +7,0%.
- В данный момент оптимизация отлажена.