МЦСТ ЭЛЬБРУС

Агрессивная inline-подстановка функций для VLIW архитектур

Волконский В.Ю., к.т.н.
Нейман-Заде М.И., к.ф.-м.н.
Ермолицкий А.В., к.т.н.
Четверина О.А.
Маркин А.Л.

Москва 2015

МЦСТ ЭЛЬБРУС

inline-подстановка

```
int foo( int x) {
  return (x == 0) ? 0 : x - 1;
}
int bar( int y) {
  return foo(y) + foo(0) + foo(y+1);
}
```

```
int bar( int y) {
  int tmp;
  if (y   == 0) tmp  = 0; else tmp  = y       - 1;
  if (0   == 0) tmp += 0; else tmp += 0       - 1;
  if (y+1 == 0) tmp += 0; else tmp += (y + 1) - 1;
  return tmp;
}
```

МЦСТ ЭЛЬБРУС

inline-подстановка

```
int bar( int y) {
  int tmp;
  if (y   == 0) tmp  = 0; else tmp  = y       - 1;
  if (0   == 0) tmp += 0; else tmp += 0       - 1;
  if (y+1 == 0) tmp += 0; else tmp += (y + 1) - 1;
  return tmp;
}
```

```
int bar( int y) {
  int tmp;
  if (y == 0) tmp  = 0; else tmp = y - 1;
  if (y+1 != 0) tmp += y;
  return tmp;
}
```

inline-подстановка

Преимущества:

- Уменьшение количества вызовов
- Удаление передачи параметров и результата
- Дополнительные возможности для оптимизаций

Недостатки:

- Рост размера кода
- Замедление компиляции
- Негативное влияние на подсистему памяти

МЦСТ ЭЛЬБРУС

inline для VLIW

- дорогие операции вызова
- смешивание кода из разных процедур
- более оптимальное использование параллельных ресурсов
- важно время компиляции

МЦСТ ЭЛЬБРУС

Агрессивная inline-подстановка

Классическая постановка задачи:

$$T_e \rightarrow min$$
$$S' \leq \alpha S$$

Классическое приближённое решение:

- Использование профильной информации
- Ограничение на размер подставляемых функций
- Подстановка вызовов в порядке уменьшения счётчиков

МЦСТ ЭЛЬБРУС

Сбалансированная inline-подстановка

Постановка задачи:

$$T_e^3 T_c \rightarrow min$$

$$T_e = \sum_i T_i$$

$$T_c = \alpha + \beta N^2$$

Вес inline-подстановки:

$$W = -3 \frac{\Delta T_e}{T_e} - \frac{\Delta T_c}{T_c}$$

МЦСТ ЭЛЬБРУС

Ключевые особенности

- Учёт оптимизаций
- Высокая масштабируемость
- Умеренный рост кода
- Умеренная зависимость от профильной информации

МЦСТ ЭЛЬБРУС

Учёт оптимизаций

- Code scheduling
- Global copy propagation
- Constant propagation
- Redundant forks elimination
- Dead code elimination
- Software pipelining
- Other loop optimizations

МЦСТ ЭЛЬБРУС

Пример учёта оптимизаций

```
int foo( int a, int b) {
  if( a == 1 ) {
    if( b == 1 ) {
      <code1>
    } else {
      <code2>
    }
  } else {
    <code3>
  }
}

int bar( ... ) {
  ...
  x = foo( 1, 2);
  ...
}
```

```
int bar( ... ) {
  ...
  x = <code2>;
  ...
}
```

Пример учёта оптимизаций

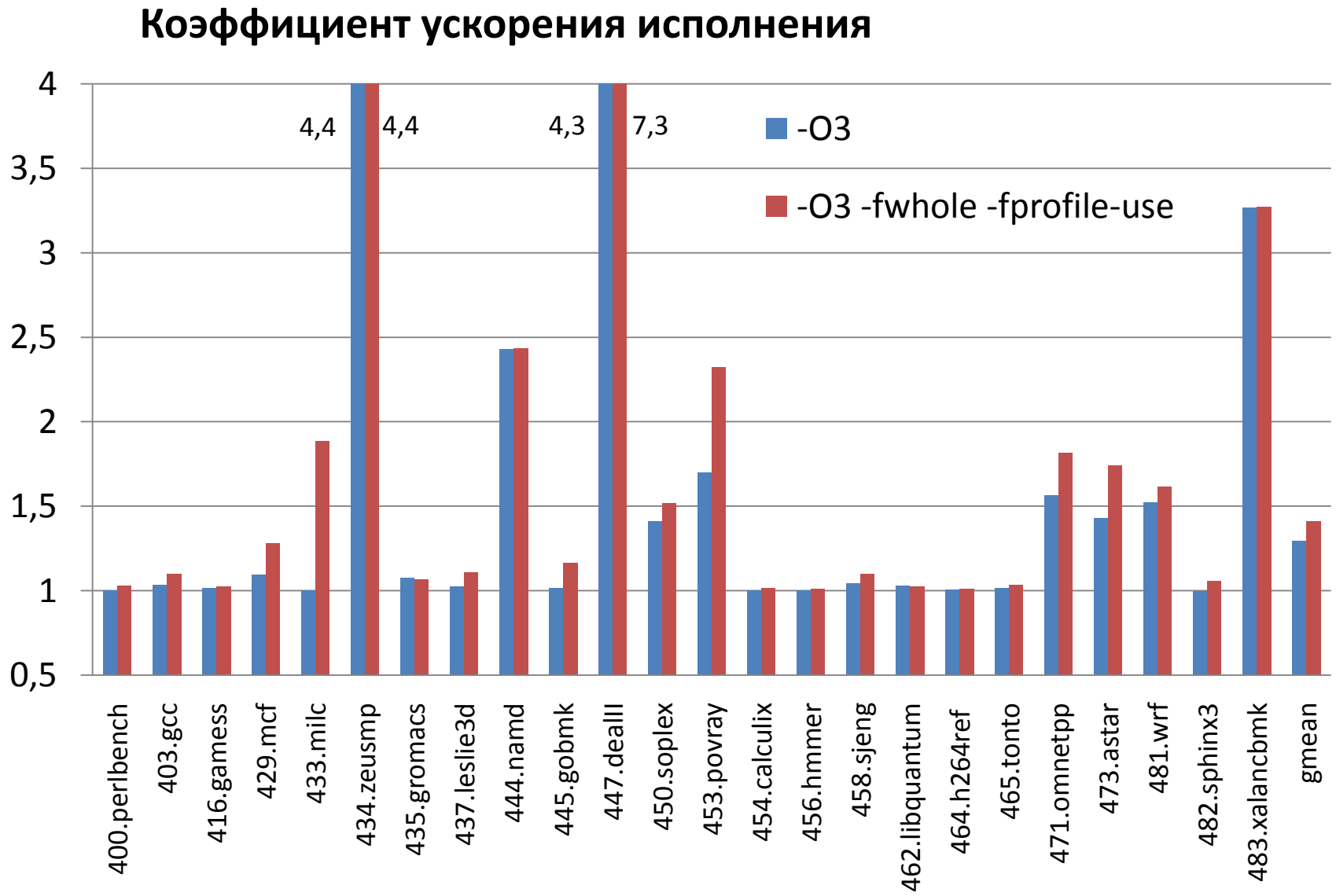
```
void foo( int n, ...) {
  for( i=0; i<n; i++ )
    <loop body>
  ...
}

int bar( ... ) {
  ...
  foo( N, ...);
  foo( N, ...);
  foo( N, ...);
  ...
}
```

```
int bar( ... ) {
  ...
  for( i=0; i<N; i++ ) {
    <loop body (1)>
    <loop body (2)>
    <loop body (3)>
  }
  ...
}
```

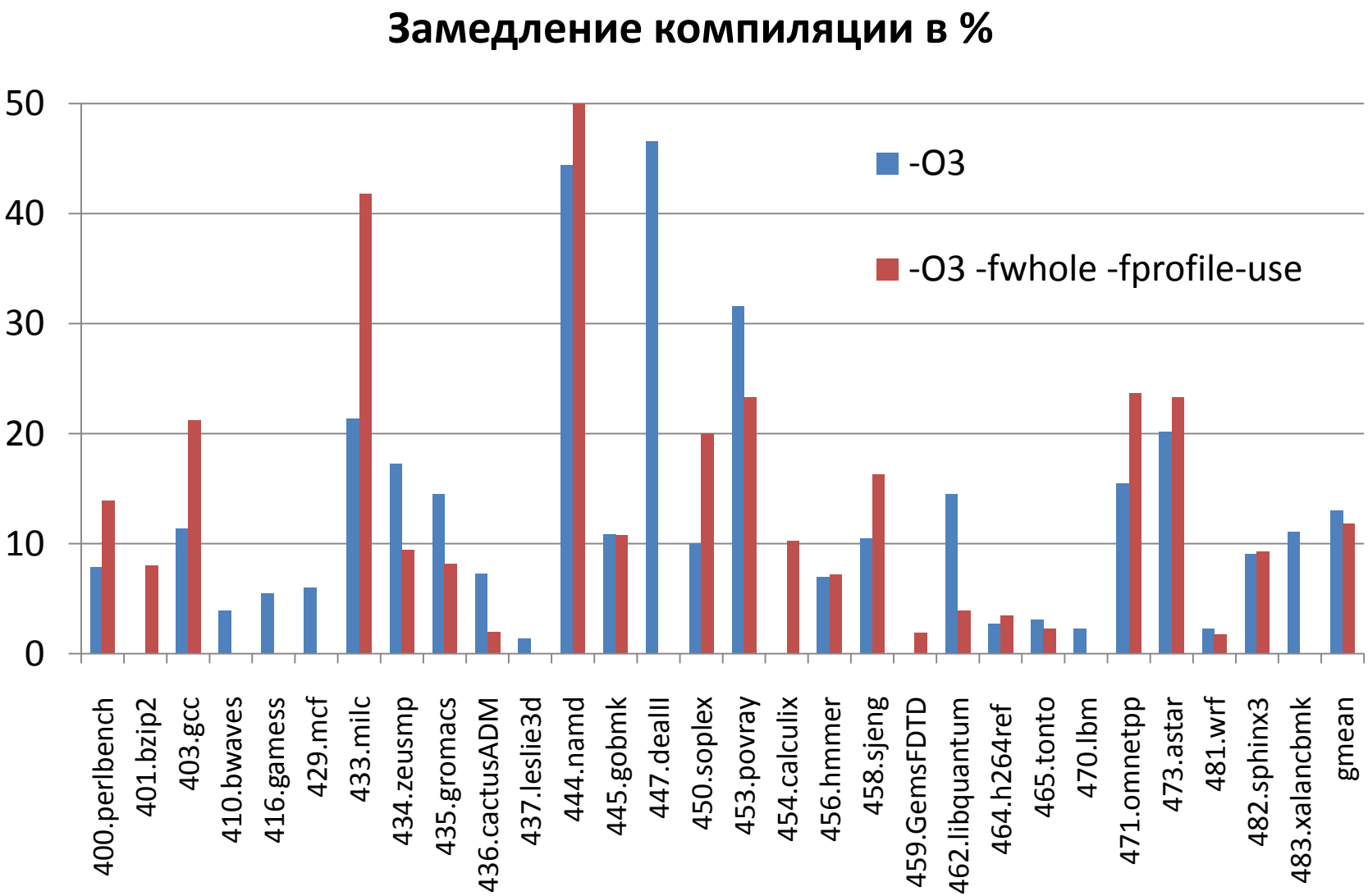
МЦСТ ЭЛЬБРУС

Экспериментальные результаты



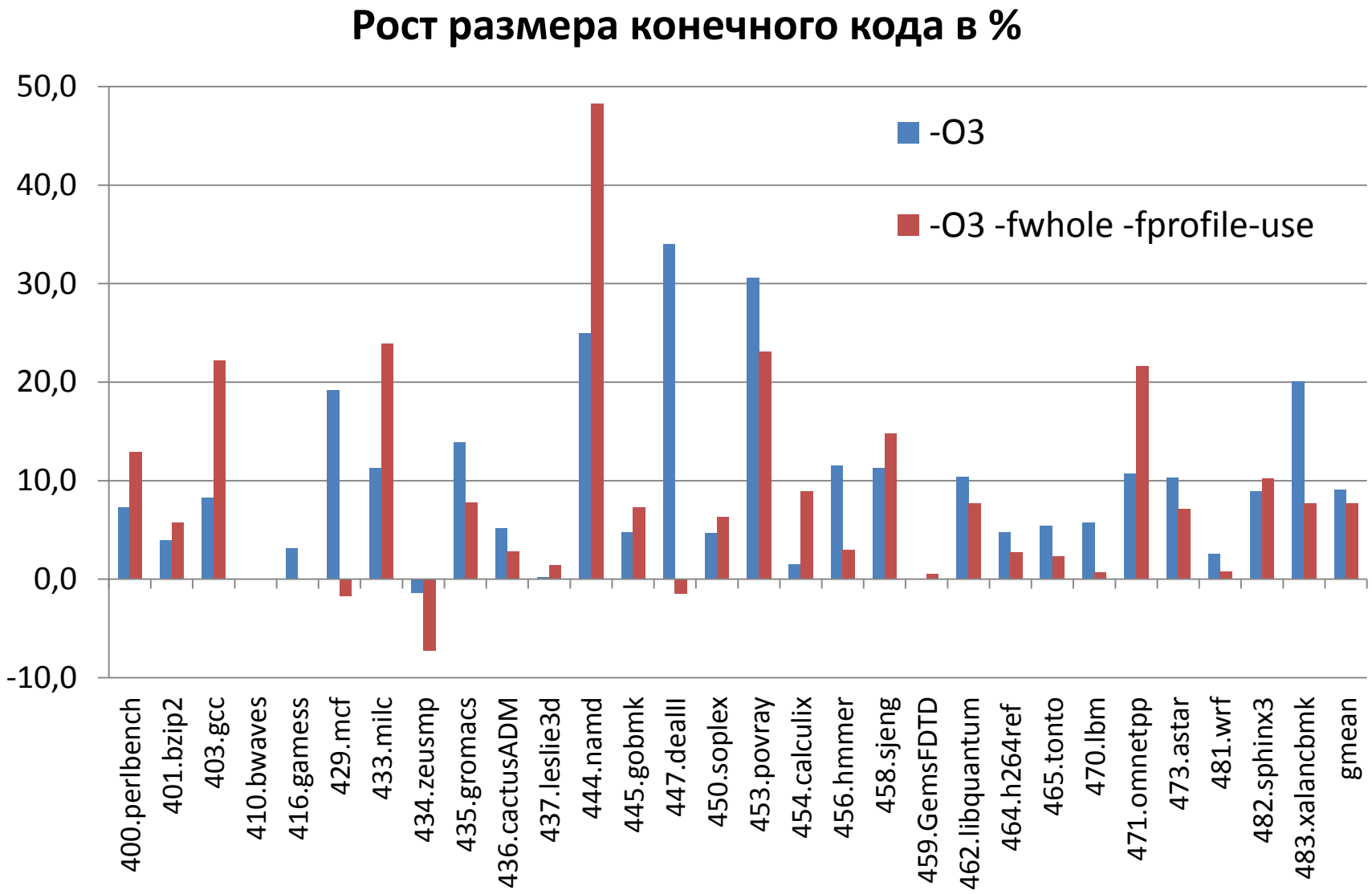
МЦСТ ЭЛЬБРУС

Экспериментальные результаты



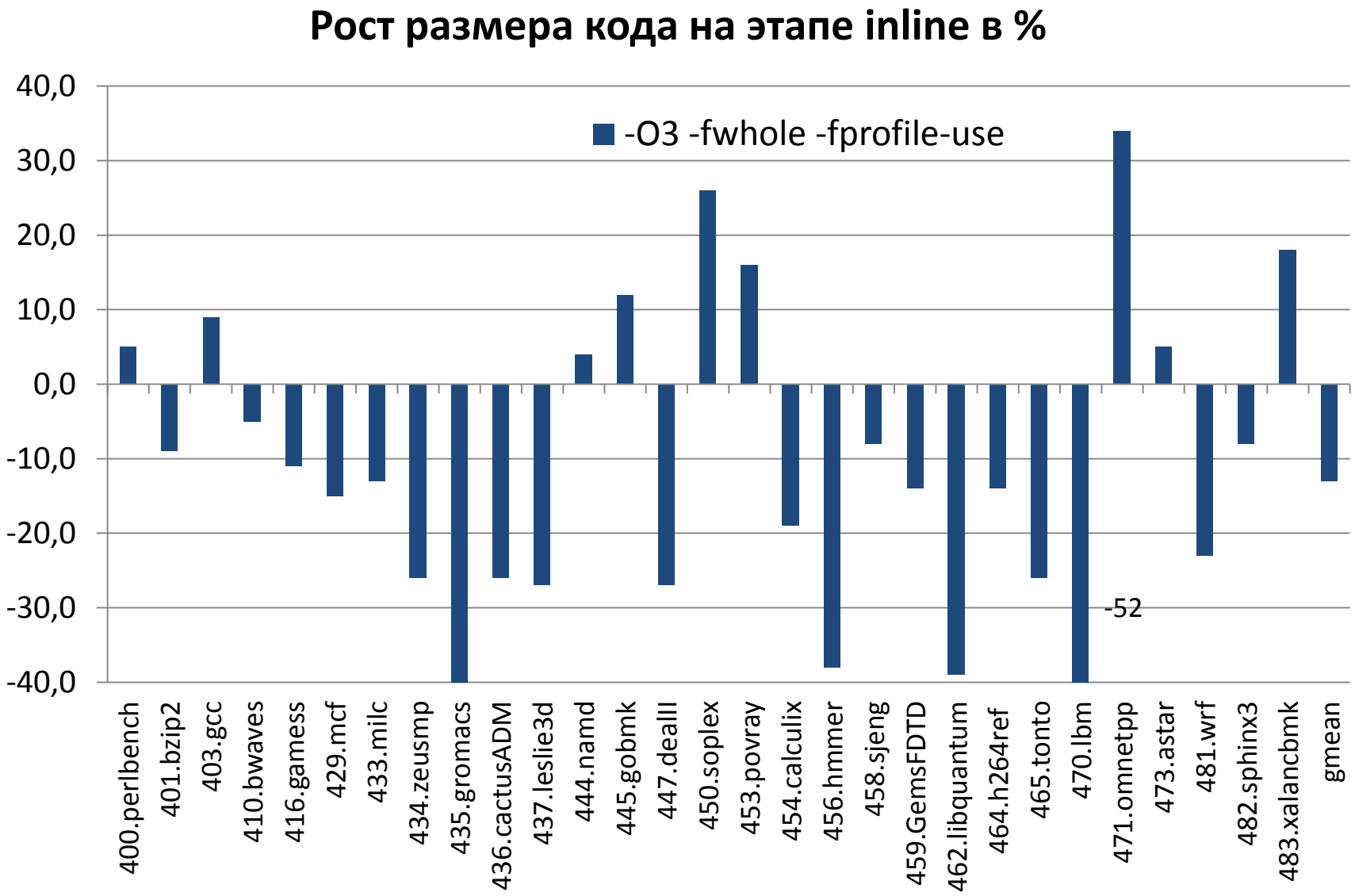
МЦСТ ЭЛЬБРУС

Экспериментальные результаты



МЦСТ ЭЛЬБРУС

Экспериментальные результаты



Вопросы?