

Автоматизированный анализ производительности и регрессии компилятора в системе LLVM LNT

Владимир Милосердов
371 гр.

Научный руководитель
Немешев М.Х.

Научный консультант
Якушкин С. И. (Synopsys)

СПбГУ, 2018

Цель

Улучшение фреймворка LNT для автоматизации отслеживания регрессий с целью улучшения эффективности процесса разработки

Описание

Предметная область: Анализ регрессий при совершенствовании оптимизаций компилятора

Задачи:

- Кто виноват: компилятор или программа?
- Анализ производительности и функциональное тестирование
- Унифицировать запуск различных тестов
- Сбор данных

LLVM LNT

LNT Database Suite nts Baselines

nts / LNT-Broadwell-AVX2-O3_clang_DEV_x86_64:1344 / Run Results

- Machine Info
- Run Info
- View Options
- Report
- execution_time

Runs:

- 04/10/2018 07:38:52
- 04/10/2018 06:49:07
- 04/10/2018 05:25:43
- 04/10/2018 04:07:55

Compare To:

- 04/10/2018 07:38:52
- 04/10/2018 06:49:07
- 04/10/2018 05:25:43
- 04/10/2018 04:07:55

Status Group	#	# (B)
Performance Regressions	0	8
Performance Improvements	1	16
Added Tests	0	11
Unchanged Tests	2554	2520
Total Tests	2555	

Run-Over-Run Changes Detail

Performance Improvements - execution_time	Δ	Previous	Current	σ	Δ (B)	σ (B)
MultiSource/Benchmarks/Fhourstones/fhourstones	-11.92%	0.7474	0.6583	0.0327	-3.97%	0.0327

Run-Over-Baseline Changes Detail

Performance Regressions - execution_time	Δ (B)	Baseline	Current	σ (B)	Δ	σ
SingleSource/Benchmarks/Misc-C++/Large/ray	35.64%	1.7880	2.4253	0.1123	-0.45%	0.1123
MultiSource/Benchmarks/Pttrdist/ks/ks	27.86%	1.0638	1.3602	0.0347	-4.28%	0.0347
SingleSource/Benchmarks/Misc/flops-1	22.02%	0.8919	1.0883	0.0054	-0.68%	0.0054
SingleSource/Benchmarks/Misc/flops	16.97%	4.5174	5.2839	0.0788	1.13%	0.0788

Задачи

- Улучшить генерацию диагностических сообщений компилятора, построенного на базе LLVM
- Проверить эффективность выявления регрессий с использованием диагностических сообщений
- Добавить возможность визуализации диагностических сообщений проходов компилятора в системе LLVM LNT
- Улучшить представления данных тестирования в LLVM LNT

Пример

```
void f1(int *x, int end) {  
    end >>= 2;  
    for (int i = 1; i < end; i++)  
        x[i] += 10;  
}
```

test.cxx:3:1: remark: loop vectorized

test.cxx:3:1: remark: loop not vectorized: uncountable iteration count

Сравнение диагностической информации двух прогонов бенчмарка

```
define internal i32 @rb_scope_hello:_"  
(i32 %self, i8* %sel, i32 %what) {  
MainBlock:  
  
etc...  
  
%2 = load i8** @0  
%3 = call i32 @rb_str_new(i8* getelementptr...  
%4 = load i32* %0  
%5 = call i32 @rb_str_new(i8* getelementptr...  
%6 = call i32 (i32, ...) @rb_str_new_fast(...  
%7 = getelementptr i32* %argv, i32 0  
store i32 %6, i32* %7  
%8 = call i32 @vm_dispatch(...  
  
etc...
```

```
--- !Missed  
Pass:      slp-vectorizer  
Name:      InequableTypes  
Function:   foo  
Args:  
- String:   'Cannot SLP vectorize list:  
- String:   'parts of scalar instructio  
- InstructionOpcode1: add  
- String:   ' and '  
- InstructionOpcode2: phi  
...  
--- !Missed  
Pass:      slp-vectorizer  
Name:      NotPossible  
Function:   foo  
Args:  
- String:   'Cannot SLP vectorize list:
```

```
38c int a = 0;  
39c int b = 5;  
39d int c = prev_count;  
39e if (a > c)  
39f     prev_count = b;  
39g else  
3a0     c = b;  
3a1 b = prev_count;  
3a2 prev_count++;
```

licm

```
38c int a = 0;  
39c int b = 5;  
39d int c = prev_count;  
39e if (a > c)  
39f     prev_count = b;  
39g else  
3a0     c = b;  
3a1 b = prev_count;  
3a2 prev_count++;
```

Licm fail

Baseline

Test Data

[Summary data table](#)

Name	compile_time				execution_time								Log
Test	Prev	Current	%	Δ	σ	MAD	Prev	Current	%	Δ	σ	MAD	
MultiSource/Benchmarks/Prolangs-C/allroots/allroots-link	0.0918	0.0900	-1.96%	-0.0018	-	-	-	-	-	0.0000	-	-	Log
MultiSource/Benchmarks/Prolangs-C/unix-tbl/unix-tbl-link	0.2811	0.3015	7.26%	0.0204	-	-	-	-	-	0.0000	-	-	Log
MultiSource/Benchmarks/Prolangs-C/gnugo/gnugo	3.3230	3.6131	8.73%	0.2901	-	-	82.4819	91.9274	11.45%	9.4455	-	-	Log
MultiSource/Benchmarks/Prolangs-C/football/football	5.1747	5.7475	11.07%	0.5728	-	-	0.2140	0.2491	16.40%	0.0351	-	-	Log
MultiSource/Benchmarks/Prolangs-C/football/football-link	0.1516	0.1695	11.81%	0.0179	-	-	-	-	-	0.0000	-	-	Log
MultiSource/Benchmarks/Prolangs-C/fixoutput/fixoutput	0.3263	0.3802	16.52%	0.0539	-	-	0.2288	0.2659	16.22%	0.0371	-	-	Log
MultiSource/Benchmarks/Prolangs-C/gnugo/gnugo-link	0.3573	0.4187	17.18%	0.0614	-	-	-	-	-	0.0000	-	-	Log
MultiSource/Benchmarks/Prolangs-C/allroots/allroots	0.4740	0.5563	17.36%	0.0823	-	-	0.3613	0.4227	16.99%	0.0614	-	-	Log
MultiSource/Benchmarks/Prolangs-C/fixoutput/fixoutput-link	0.1167	0.1387	18.85%	0.0220	-	-	-	-	-	0.0000	-	-	Log
MultiSource/Benchmarks/Prolangs-C/unix-tbl/unix-tbl	5.5225	7.9669	44.26%	2.4444	-	-	0.1820	0.2106	15.71%	0.0286	-	-	Log