Анализ автовекторизации в LLVM

# Конфигурация

CPU: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz

Instruction set: MMX, SSE, SSE2, SSE3, SSE4.1, SSE4.2, EM64T, AES, AVX, AVX2, FMA3

# Исследуемый цикл

for (int i=0; i < N; i++) r[i] = a[i] + b[i];

# Сравнение LLVM IR

|  |  |
| --- | --- |
| Без векторизации и раскрутки | Без векторизации, с раскруткой |
| for.body: ; preds = %for.body, %entry  %indvars.iv = phi i64 [ 0, %entry ], [ %indvars.iv.next, %for.body ]  %add.ptr.i = getelementptr inbounds float, float\* %0, i64 %indvars.iv  %3 = load float, float\* %add.ptr.i, align 4, !tbaa !7  %add.ptr.i14 = getelementptr inbounds float, float\* %1, i64 %indvars.iv  %4 = load float, float\* %add.ptr.i14, align 4, !tbaa !7  %add = fadd float %3, %4  %add.ptr.i12 = getelementptr inbounds float, float\* %2, i64 %indvars.iv  store float %add, float\* %add.ptr.i12, align 4, !tbaa !7  %indvars.iv.next = add nuw nsw i64 %indvars.iv, 1  %exitcond.not = icmp eq i64 %indvars.iv.next, 1048576  br i1 %exitcond.not, label %for.cond.cleanup, label %for.body, !llvm.loop !9 | for.body: ; preds = %for.body, %entry  %indvars.iv = phi i64 [ 0, %entry ], [ %indvars.iv.next.3, %for.body ]  %add.ptr.i = getelementptr inbounds float, float\* %0, i64 %indvars.iv  %3 = load float, float\* %add.ptr.i, align 4, !tbaa !7  %add.ptr.i14 = getelementptr inbounds float, float\* %1, i64 %indvars.iv  %4 = load float, float\* %add.ptr.i14, align 4, !tbaa !7  %add = fadd float %3, %4  %add.ptr.i12 = getelementptr inbounds float, float\* %2, i64 %indvars.iv  store float %add, float\* %add.ptr.i12, align 4, !tbaa !7  %indvars.iv.next = or i64 %indvars.iv, 1  %add.ptr.i.1 = getelementptr inbounds float, float\* %0, i64 %indvars.iv.next  %5 = load float, float\* %add.ptr.i.1, align 4, !tbaa !7  %add.ptr.i14.1 = getelementptr inbounds float, float\* %1, i64 %indvars.iv.next  %6 = load float, float\* %add.ptr.i14.1, align 4, !tbaa !7  %add.1 = fadd float %5, %6  %add.ptr.i12.1 = getelementptr inbounds float, float\* %2, i64 %indvars.iv.next  store float %add.1, float\* %add.ptr.i12.1, align 4, !tbaa !7  %indvars.iv.next.1 = or i64 %indvars.iv, 2  %add.ptr.i.2 = getelementptr inbounds float, float\* %0, i64 %indvars.iv.next.1  %7 = load float, float\* %add.ptr.i.2, align 4, !tbaa !7  %add.ptr.i14.2 = getelementptr inbounds float, float\* %1, i64 %indvars.iv.next.1  %8 = load float, float\* %add.ptr.i14.2, align 4, !tbaa !7  %add.2 = fadd float %7, %8  %add.ptr.i12.2 = getelementptr inbounds float, float\* %2, i64 %indvars.iv.next.1  store float %add.2, float\* %add.ptr.i12.2, align 4, !tbaa !7  %indvars.iv.next.2 = or i64 %indvars.iv, 3  %add.ptr.i.3 = getelementptr inbounds float, float\* %0, i64 %indvars.iv.next.2  %9 = load float, float\* %add.ptr.i.3, align 4, !tbaa !7  %add.ptr.i14.3 = getelementptr inbounds float, float\* %1, i64 %indvars.iv.next.2  %10 = load float, float\* %add.ptr.i14.3, align 4, !tbaa !7  %add.3 = fadd float %9, %10  %add.ptr.i12.3 = getelementptr inbounds float, float\* %2, i64 %indvars.iv.next.2  store float %add.3, float\* %add.ptr.i12.3, align 4, !tbaa !7  %indvars.iv.next.3 = add nuw nsw i64 %indvars.iv, 4  %exitcond.not.3 = icmp eq i64 %indvars.iv.next.3, 1048576  br i1 %exitcond.not.3, label %for.cond.cleanup, label %for.body |

Табл 1.

|  |  |
| --- | --- |
| Без векторизации и раскрутки | С векторизацией, без раскрутки |
| for.body: ; preds = %for.body, %entry  %indvars.iv = phi i64 [ 0, %entry ], [ %indvars.iv.next, %for.body ]  %add.ptr.i = getelementptr inbounds float, float\* %0, i64 %indvars.iv  %3 = load float, float\* %add.ptr.i, align 4, !tbaa !7  %add.ptr.i14 = getelementptr inbounds float, float\* %1, i64 %indvars.iv  %4 = load float, float\* %add.ptr.i14, align 4, !tbaa !7  %add = fadd float %3, %4  %add.ptr.i12 = getelementptr inbounds float, float\* %2, i64 %indvars.iv  store float %add, float\* %add.ptr.i12, align 4, !tbaa !7  %indvars.iv.next = add nuw nsw i64 %indvars.iv, 1  %exitcond.not = icmp eq i64 %indvars.iv.next, 1048576  br i1 %exitcond.not, label %for.cond.cleanup, label %for.body, !llvm.loop !9 | vector.body: ; preds = %entry, %vector.body  %index = phi i64 [ %index.next, %vector.body ], [ 0, %entry ]  %3 = getelementptr inbounds float, float\* %0, i64 %index  %4 = bitcast float\* %3 to <4 x float>\*  %wide.load = load <4 x float>, <4 x float>\* %4, align 4, !tbaa !7, !alias.scope !9  %5 = getelementptr inbounds float, float\* %1, i64 %index  %6 = bitcast float\* %5 to <4 x float>\*  %wide.load24 = load <4 x float>, <4 x float>\* %6, align 4, !tbaa !7, !alias.scope !12  %7 = fadd <4 x float> %wide.load, %wide.load24  %8 = getelementptr inbounds float, float\* %2, i64 %index  %9 = bitcast float\* %8 to <4 x float>\*  store <4 x float> %7, <4 x float>\* %9, align 4, !tbaa !7, !alias.scope !14, !noalias !16  %index.next = add i64 %index, 4  %10 = icmp eq i64 %index.next, 1048576  br i1 %10, label %for.cond.cleanup, label %vector.body, !llvm.loop !17  for.cond.cleanup: ; preds = %vector.body, %for.body  ret void  for.body: ; preds = %entry, %for.body  %indvars.iv = phi i64 [ %indvars.iv.next, %for.body ], [ 0, %entry ]  %add.ptr.i = getelementptr inbounds float, float\* %0, i64 %indvars.iv  %11 = load float, float\* %add.ptr.i, align 4, !tbaa !7  %add.ptr.i14 = getelementptr inbounds float, float\* %1, i64 %indvars.iv  %12 = load float, float\* %add.ptr.i14, align 4, !tbaa !7  %add = fadd float %11, %12  %add.ptr.i12 = getelementptr inbounds float, float\* %2, i64 %indvars.iv  store float %add, float\* %add.ptr.i12, align 4, !tbaa !7  %indvars.iv.next = add nuw nsw i64 %indvars.iv, 1  %exitcond.not = icmp eq i64 %indvars.iv.next, 1048576  br i1 %exitcond.not, label %for.cond.cleanup, label %for.body, !llvm.loop !20 |

Табл 2.

|  |  |
| --- | --- |
| Без векторизации и раскрутки | С векторизацией и раскруткой |
| for.body: ; preds = %for.body, %entry  %indvars.iv = phi i64 [ 0, %entry ], [ %indvars.iv.next, %for.body ]  %add.ptr.i = getelementptr inbounds float, float\* %0, i64 %indvars.iv  %3 = load float, float\* %add.ptr.i, align 4, !tbaa !7  %add.ptr.i14 = getelementptr inbounds float, float\* %1, i64 %indvars.iv  %4 = load float, float\* %add.ptr.i14, align 4, !tbaa !7  %add = fadd float %3, %4  %add.ptr.i12 = getelementptr inbounds float, float\* %2, i64 %indvars.iv  store float %add, float\* %add.ptr.i12, align 4, !tbaa !7  %indvars.iv.next = add nuw nsw i64 %indvars.iv, 1  %exitcond.not = icmp eq i64 %indvars.iv.next, 1048576  br i1 %exitcond.not, label %for.cond.cleanup, label %for.body, !llvm.loop !9 | vector.body: ; preds = %entry, %vector.body  %index = phi i64 [ %index.next.1, %vector.body ], [ 0, %entry ]  %3 = getelementptr inbounds float, float\* %0, i64 %index  %4 = bitcast float\* %3 to <4 x float>\*  %wide.load = load <4 x float>, <4 x float>\* %4, align 4, !tbaa !7, !alias.scope !9  %5 = getelementptr inbounds float, float\* %3, i64 4  %6 = bitcast float\* %5 to <4 x float>\*  %wide.load24 = load <4 x float>, <4 x float>\* %6, align 4, !tbaa !7, !alias.scope !9  %7 = getelementptr inbounds float, float\* %1, i64 %index  %8 = bitcast float\* %7 to <4 x float>\*  %wide.load25 = load <4 x float>, <4 x float>\* %8, align 4, !tbaa !7, !alias.scope !12  %9 = getelementptr inbounds float, float\* %7, i64 4  %10 = bitcast float\* %9 to <4 x float>\*  %wide.load26 = load <4 x float>, <4 x float>\* %10, align 4, !tbaa !7, !alias.scope !12  %11 = fadd <4 x float> %wide.load, %wide.load25  %12 = fadd <4 x float> %wide.load24, %wide.load26  %13 = getelementptr inbounds float, float\* %2, i64 %index  %14 = bitcast float\* %13 to <4 x float>\*  store <4 x float> %11, <4 x float>\* %14, align 4, !tbaa !7, !alias.scope !14, !noalias !16  %15 = getelementptr inbounds float, float\* %13, i64 4  %16 = bitcast float\* %15 to <4 x float>\*  store <4 x float> %12, <4 x float>\* %16, align 4, !tbaa !7, !alias.scope !14, !noalias !16  %index.next = or i64 %index, 8  %17 = getelementptr inbounds float, float\* %0, i64 %index.next  %18 = bitcast float\* %17 to <4 x float>\*  %wide.load.1 = load <4 x float>, <4 x float>\* %18, align 4, !tbaa !7, !alias.scope !9  %19 = getelementptr inbounds float, float\* %17, i64 4  %20 = bitcast float\* %19 to <4 x float>\*  %wide.load24.1 = load <4 x float>, <4 x float>\* %20, align 4, !tbaa !7, !alias.scope !9  %21 = getelementptr inbounds float, float\* %1, i64 %index.next  %22 = bitcast float\* %21 to <4 x float>\*  %wide.load25.1 = load <4 x float>, <4 x float>\* %22, align 4, !tbaa !7, !alias.scope !12  %23 = getelementptr inbounds float, float\* %21, i64 4  %24 = bitcast float\* %23 to <4 x float>\*  %wide.load26.1 = load <4 x float>, <4 x float>\* %24, align 4, !tbaa !7, !alias.scope !12  %25 = fadd <4 x float> %wide.load.1, %wide.load25.1  %26 = fadd <4 x float> %wide.load24.1, %wide.load26.1  %27 = getelementptr inbounds float, float\* %2, i64 %index.next  %28 = bitcast float\* %27 to <4 x float>\*  store <4 x float> %25, <4 x float>\* %28, align 4, !tbaa !7, !alias.scope !14, !noalias !16  %29 = getelementptr inbounds float, float\* %27, i64 4  %30 = bitcast float\* %29 to <4 x float>\*  store <4 x float> %26, <4 x float>\* %30, align 4, !tbaa !7, !alias.scope !14, !noalias !16  %index.next.1 = add nuw nsw i64 %index, 16  %31 = icmp eq i64 %index.next.1, 1048576  br i1 %31, label %for.cond.cleanup, label %vector.body, !llvm.loop !17  for.cond.cleanup: ; preds = %vector.body, %for.body  ret void  for.body: ; preds = %entry, %for.body  %indvars.iv = phi i64 [ %indvars.iv.next.3, %for.body ], [ 0, %entry ]  %add.ptr.i = getelementptr inbounds float, float\* %0, i64 %indvars.iv  %32 = load float, float\* %add.ptr.i, align 4, !tbaa !7  %add.ptr.i14 = getelementptr inbounds float, float\* %1, i64 %indvars.iv  %33 = load float, float\* %add.ptr.i14, align 4, !tbaa !7  %add = fadd float %32, %33  %add.ptr.i12 = getelementptr inbounds float, float\* %2, i64 %indvars.iv  store float %add, float\* %add.ptr.i12, align 4, !tbaa !7  %indvars.iv.next = or i64 %indvars.iv, 1  %add.ptr.i.1 = getelementptr inbounds float, float\* %0, i64 %indvars.iv.next  %34 = load float, float\* %add.ptr.i.1, align 4, !tbaa !7  %add.ptr.i14.1 = getelementptr inbounds float, float\* %1, i64 %indvars.iv.next  %35 = load float, float\* %add.ptr.i14.1, align 4, !tbaa !7  %add.1 = fadd float %34, %35  %add.ptr.i12.1 = getelementptr inbounds float, float\* %2, i64 %indvars.iv.next  store float %add.1, float\* %add.ptr.i12.1, align 4, !tbaa !7  %indvars.iv.next.1 = or i64 %indvars.iv, 2  %add.ptr.i.2 = getelementptr inbounds float, float\* %0, i64 %indvars.iv.next.1  %36 = load float, float\* %add.ptr.i.2, align 4, !tbaa !7  %add.ptr.i14.2 = getelementptr inbounds float, float\* %1, i64 %indvars.iv.next.1  %37 = load float, float\* %add.ptr.i14.2, align 4, !tbaa !7  %add.2 = fadd float %36, %37  %add.ptr.i12.2 = getelementptr inbounds float, float\* %2, i64 %indvars.iv.next.1  store float %add.2, float\* %add.ptr.i12.2, align 4, !tbaa !7  %indvars.iv.next.2 = or i64 %indvars.iv, 3  %add.ptr.i.3 = getelementptr inbounds float, float\* %0, i64 %indvars.iv.next.2  %38 = load float, float\* %add.ptr.i.3, align 4, !tbaa !7  %add.ptr.i14.3 = getelementptr inbounds float, float\* %1, i64 %indvars.iv.next.2  %39 = load float, float\* %add.ptr.i14.3, align 4, !tbaa !7  %add.3 = fadd float %38, %39  %add.ptr.i12.3 = getelementptr inbounds float, float\* %2, i64 %indvars.iv.next.2  store float %add.3, float\* %add.ptr.i12.3, align 4, !tbaa !7  %indvars.iv.next.3 = add nuw nsw i64 %indvars.iv, 4  %exitcond.not.3 = icmp eq i64 %indvars.iv.next.3, 1048576  br i1 %exitcond.not.3, label %for.cond.cleanup, label %for.body, !llvm.loop !19 |

Табл 3.

# Анализ LLVM IR

1. При раскрутке уменьшается количество итераций цикла, при этом в одну итерацию компилятор вставляет на самом деле код сразу нескольких подряд. Грубо говоря, такой код:

for (int i = 0; i < 8; i++) { x[i] = y[i]+z[i]; }

можно раскрутить в:

for (int i = 0; i < 4; i++) { x[i] = y[i]+z[i]; x[i+1] = y[i+1]+z[i+1]; }

1. При векторизации также уменьшается количество итераций исходного цикла, но несколько искусственным образом: инструкции не дублируются, вместо этого внутрь цикла вставляется вложенный цикл. При использовании только одного вычислительного узла для выполнения такой подход особого смысла не имеет, а вот при возможности параллельного исполнения может быть эффективным за счет того, что такой вложенный цикл может быть скомпилирован таким образом, что его итерации будут выполняться параллельно. Таким образом, по сути, общее количество итераций не изменяется.
2. При использовании и векторизации, и раскрутки, следовательно, могут раскручиваться итерации векторизованного цикла, т.е. происходит уменьшение числа итераций.

# Измерение времени исполнения

|  |  |  |
| --- | --- | --- |
| **Конфигурация** | **Время, мс** | **Ускорение** |
| Без векторизации и раскрутки | 1349.54 | - |
| Без векторизации, с раскруткой | 1351.62 | 0,99 |
| С векторизацией, без раскрутки | 1173.93 | 1,15 |
| С векторизацией, с раскруткой | 1053.08 | 1,28 |

# Анализ времени исполнения

Наибольшее ускорение наблюдается при использовании обоих подходов. Объяснение этому приводится в описании этих подходов. Впрочем, ускорение не слишком большое из-за относительной простоты вычислений, возможно, это вообще можно считать «погрешностью».