EXTENDING LOOPVECTORIZE TO SUPPORT OUTER LOOP VECTORIZATION USING VPLAN

Diego Caballero and Vectorizer Team, Intel Corporation.

April 16th, 2018 Euro LLVM Developers' Meeting. Bristol, UK.

Legal Disclaimer & Optimization Notice

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Key Takeaways

- Outer loop vectorization is a BIG missing feature in LV.
  - Project outlined at 2016 LLVM Dev Meeting. Concept of VPlan introduced to LV in early 2017.
- We present 5 patch series to incrementally support outer loop vectorization in LV.
- Implementation: the VPlan-native vectorization path – an alternative vectorization path in LV for outer loops where VPlan is built upfront in the vectorization pipeline.
  - Initially NFC for inner loop vectorization.
  - Main goal: gradually converge to a single path for inner and outer loop vectorization.
- Our proposal enables many opportunities for community participation in VPlan development.

Agenda

- VPlan Background.
- Proposal: Bring Outer Loop Vectorization to LV.
- Implementation: The VPlan-native vectorization path in LV.
- Participation opportunities for the LLVM community.

VPLAN BACKGROUND

What is VPlan?

New vectorization infrastructure for LV aimed at:

- Explicitly modeling and evaluating the cost of multiple vectorization scenarios, choosing the one with the best cost and materialize its vectorization decisions:

```
for(j = 0; j < N; j++)
  for(i = 0; i < M; i++) {
    a[j][i] = b[j][i] + c[j][i];
    d[j][i*2] = b[j][i] * C;
    d[j][i*2+1] = c[j][i] * D;
  }
```

Vectorize inner loop?

Vectorize outer loop?

Vectorize inner + outer loop?

SLP?

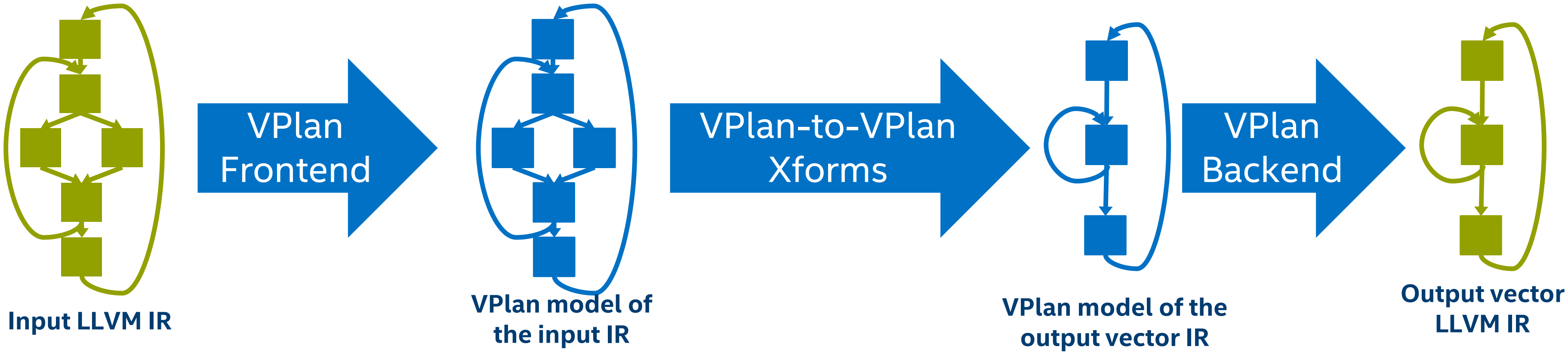
SLP-aware loop vectorization?

...

How does VPlan work?

A VPlan is a model of the output vector IR for a particular vectorization scenario:

- This model starts from the input IR.
- VPlan-to-VPlan transformations incrementally turn the input model into a model of the output vector IR.
- Input IR remains intact until the best vectorization plan is materialized on the input IR (i.e., the plan is executed).



VPlan Representation

Hierarchical CFG:

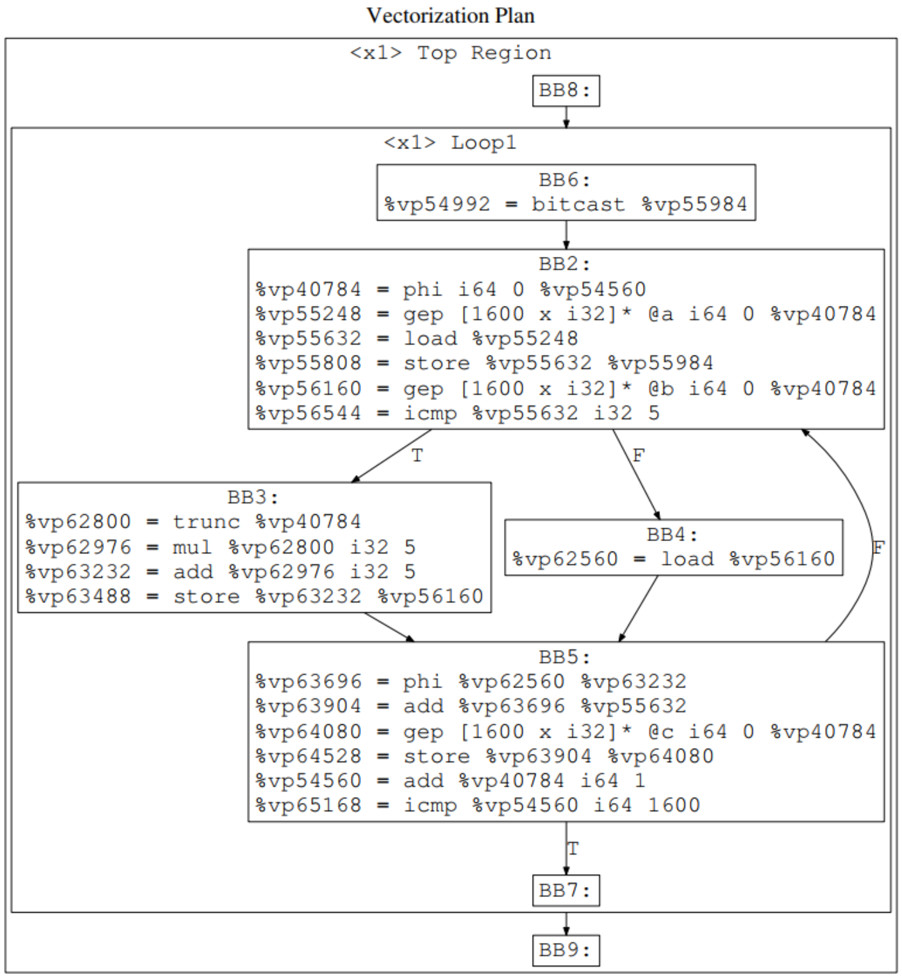
- VPBasicBlock and VPRegion.

Instruction-level representation:

- VPValue/VPInstructions (VPRecipes).

Same look and feel as LLVM IR:

- VPlan CFG ≈ LLVM CFG (despite the regions).
- VPValue/VPInstruction ≈ Value/Instruction.
  - VPBuilder ≈ IRBuilder.



VPlan Work Done So Far

Previous Talks:

- “Extending LoopVectorizer towards supporting OpenMP 4.5 SIMD and outer loop auto-vectorization.” Hideki Saito (2016 LLVM Meeting). https://www.youtube.com/watch?v=XXAvdUwO7kQ
  - Outline of the project. High level strategy.
- “Introducing VPlan to the Loop Vectorizer.” Gil Rapaport (2017 Euro LLVM Meeting). https://www.youtube.com/watch?v=IqzJRs6tb7Y
  - Introducing VPlan as a refactoring effort to model vectorization decisions.
- “Vectorizing Loops with VPlan – Current State and Next Steps.” Ayal Zaks (2017 LLVM Meeting). https://www.youtube.com/watch?v=BjBSJFzYDVk
  - Introducing VPInstructions to model existing predication using VPlan.

PROPOSAL:
BRING OUTER LOOP VECTORIZATION TO LV

What Is Outer Loop Vectorization?

```
// Vectorize here!
for(i = 0; i < N; i++) {
  float accum = 0;

  for(j = 0; j < 5; j++) {
    accum += in[j][i] * filter[j];
  }

  out[i] = sqrtf(accum)/particles;
}
```

Vectorize the outer loop dimension:

- Outer loop iterations are executed in a vector way.
- Each vector iteration executes the iterations of the nested loops for all its vector lanes.

VF=4

1st vector iteration:

i = | 0 | 1 | 2 | 3 |

j = 0, 1, 2, 3, 4

2nd vector iteration:

i = | 4 | 5 | 6 | 7 |

j = 0, 1, 2, 3, 4

When to Apply Outer Loop Vectorization?

```
// Vectorize here!
for(i = 0; i < N; i++) {
  float accum = 0;

  for(j = 0; j < 5; j++) {
    accum += in[j][i] * filter[j];
  }

  out[i] = sqrtf(accum)/particles;
}
```

Outer loop vectorization could be beneficial when:

- Vectorization of the inner loop is not applicable or inefficient.
- Vectorizing the outer loop may lead to:
  - better memory access pattern (adjacent vector loads/stores vs gather/scatters).
  - covering wider hot spot (higher number of vectorized instructions).
- ...

Why Outer Loop Vectorization in LV?

Outer loop vectorization is a BIG missing feature in LV.

- Supported in GCC and ICC for several years.
- Required to properly support some programming models like OpenMP/OpenCL.

Long term goal: multi-dimensional vectorization (i.e., inner+outer+SLP). Outer loop vectorization is a good stepping stone.

- Knowing outer loop vectorization requirements early is essential for the long term design and development.

Proposal: Explicit Outer Loop Vectorization in LV

Introduce incremental support for **explicit** outer loop vectorization using VPlan.

- Explicit outer loop vectorization: the user guarantees with a pragma that vectorization is legal and instructs the compiler to vectorize the annotated loop.

```
#pragma clang loop vectorize(enable)
// OR
#pragma omp simd
for(i = 0; i< N; i++)
  for(j = 0; j< M; j++)
    a[j][i] = b[j][i] + c[j][i];
```

- Explicit outer loop vectorization is NFC for inner loop auto-vectorization.

Proposal: Explicit Outer Loop Vectorization in LV

Introduce incremental support for explicit outer loop vectorization **using VPlan**.

- Build VPlan upfront in the vectorizer pipeline.
- Implement transformations needed for outer loop vectorization as VPlan-to-VPlan transformations of the model of the input IR.
  - Without altering the input IR.

Proposal: Explicit Outer Loop Vectorization in LV

Introduce **incremental** support for explicit outer loop vectorization using VPlan.

- **5 patch series** with specific goals to progressively extend supported outer loops and vectorization technology:
  - Patch Series #1: Trivial Outer Loops w/ Trivially Uniform Branches.
  - Patch Series #2: Support for Trivially Divergent Branches.
  - Patch Series #3: Support for Non-Trivial Uniform Branches.
  - Patch Series #4: Support for divergent inner loops.
  - Patch Series #5: Support for multi-exit inner loops.

Patch Series #1: Trivial Outer Loops w/ Trivially Uniform Branches

Only support for trivial loops:

- All branches must be trivially uniform (loop nest invariant, no DA).
- Loops must be very simple:
  - Canonical IVs.
  - Simple bottom test condition: IV < loop nest invariant.
  - Inner loops must be uniform (all vector lanes will execute the same iterations).
- VF must be specified by the user.

```
#pragma omp simd simdlen(8)
for(i = 0; i < N; i++) {
  if (cond1) // Loop nest invariant
    for(j = 0; j < M; j++) {
      a[j][i] = b[j][i] + c[j][i];
    }
}
```

Patch Series #1: Trivial Outer Loops w/ Trivially Uniform Branches

New Vectorization Technology:

- VPlan H-CFG construction upfront in the vectorization pipeline.
- VPlan-based loop representation (VPLoopRegion) and loop analysis (VPLoopInfo).
- CG support for outer loops and preserving uniform control flow (multi BB vector loops).

```
#pragma omp simd simdlen(8)
for(i = 0; i < N; i++) {
  if (cond1) // Loop nest invariant
    for(j = 0; j < M; j++) {
      a[j][i] = b[j][i] + c[j][i];
    }
}
```

Patch Series #1: Trivial Outer Loops w/ Trivially Uniform Branches

Current Status

- Patch #1 (D40874): Add irreducible CFG detection for outer loops (Committed).
- Patch #2 (D42447): Detect outer loops for explicit vectorization (Under review).
- Patch #3 (D44338): Build plain CFG with simple recipes for outer loops (Under review).

Patch Series #2: Support for Trivially Divergent Branches

Improvements from previous series:

- Loops can have trivially divergent (not loop nest invariant) branches.
- Inner loops must still be uniform (all vector lanes would execute the same iterations).

```
#pragma omp simd simdlen(8)
for(i = 0; i < N; i++) {
  for(j = 0; j < M; j++) {
    if (a[j][i] < 0)
      a[j][i] = b[j][i] + c[j][i];
  }
}
```

Patch Series #2: Support for Trivially Divergent Branches

New Vectorization Technology:

- Basic VPlan-based predication algorithm.
- VPlan representation for predicates and VPInstruction masking.
- CG support for previous changes.

```
#pragma omp simd simdlen(8)
for(i = 0; i < N; i++) {
  for(j = 0; j < M; j++) {
    if (a[j][i] < 0)
      a[j][i] = b[j][i] + c[j][i];
  }
}
```

Patch Series #3: Support for Non-Trivial Uniform Branches

Improvements from previous series:

- Detection of non-trivial uniform branches: branches that are NOT loop nest invariant but are uniform.

New Vectorization Technology:

- VPlan-based Divergence Analysis.

```
#pragma omp simd simdlen(2)
for(i = 0; i < N; i++) {
  for(j = 0; j < M; j++) {
    // Uniform for i-loop vect.
    if (func(j))
      tmp = min;
    else
      tmp = input[j][i];

    output[j][i] = tmp * filter[j][i];
  }
}
```

Patch Series #4: Support for Divergent Inner Loops

```
#pragma omp simd simdlen(8)
for(i = 0; i < N; i++) {
  for(j = 0; j < i; j++) {
    a[j][i] = b[j][i] + c[j][i];
  }
}
```

VF=4: i = | 0 | 1 | 2 | 3 |

Lane 0: won't execute any 'j' iteration.

Lane 1: will execute 1 'j' iteration.

Lane 2: will execute 2 'j' iterations.

Lane 3: will execute 3 'j' iterations.

Improvements from previous series:

- Inner loops can be divergent (all vector lanes may NOT execute the same iterations).

Patch Series #4: Support for Divergent Inner Loops

New Vectorization Technology:

- VPlan-to-VPlan transformation to turn the divergent inner loop into uniform inner loop.
  - All vector lanes execute the same ‘j’ iterations.
  - Some vector lanes will be masked out for the whole ‘j’ loop body.

```
#pragma omp simd simdlen(8)
for(i = 0; i < N; i++) {
  jub = hmax(i);
  for(j = 0; j < jub; j++) {
    if (j < i)
      a[j][i] = b[j][i] + c[j][i];
  }
}
```

Patch Series #5: Support for Multi-Exit Inner Loops

```
#pragma omp simd simdlen(8)
for(i = 0; i < N; i++) {
  int accum = a[i];

  for(j = 0; j < M; j++) {
    if (accum > threshold)
      break;
    accum += b[j][i] + c[j][i];
  }
  a[i] = accum;
}
```

Improvements from previous series:

- Inner loops can have multiple exits.
- Outer loops must have a single exit.
  - We are not dealing with search loops.

Patch Series #5: Support for Multi-Exit Inner Loops

New Vectorization Technology:

- VPlan-to-VPlan transformation to turn multi-exit inner loops into single exit inner loops.
  - Masking out vector lanes taking the early exit.

```
#pragma omp simd simdlen(8)
for(i = 0; i < N; i++) {
  int accum = a[i];

  exit_taken = false;
  for(j = 0; !exit_taken &&
             j < M; j++) {
    if (accum > threadshold)
      exit_taken = true;
    if (!exit_taken)
      accum += b[j][i] + c[j][i];
  }
  a[i] = accum;
}
```

Patch Series: Further Information

- Status Update on VPlan: Where we are currently, and what's ahead of us. December 2017. http://lists.llvm.org/pipermail/llvm-dev/2017-December/119522.html
- RFC: Proposal for Outer Loop Vectorization Implementation Plan, December 2017. http://lists.llvm.org/pipermail/llvm-dev/2017-December/119523.html

IMPLEMENTATION:
THE VPLAN-NATIVE VECTORIZATION PATH IN LV

Implementation Requirements

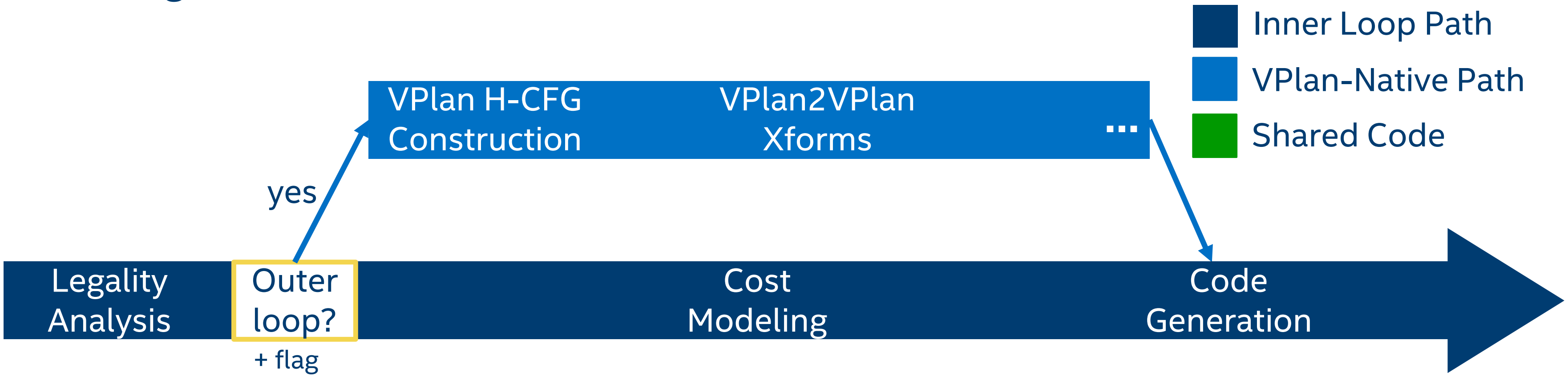
Introducing outer loop vectorization in LV is not an easy task!

- Do not destabilize inner loop vectorization.
  - Outer loop patches must be NFC for inner loop vectorization.
- Development must be incremental, flexible and cost efficient:
  - No massive code replacement.
  - Allow to continue the refactoring/porting of existing code in LV to VPlan.
  - Allow the development of new vectorization features.
- Single code base:
  - Share and extend existing code in LV to support outer loops.

Proposal: VPlan-Native Vectorization Path in LV

Introduce an alternative vectorization path in LV: the VPlan-native path.

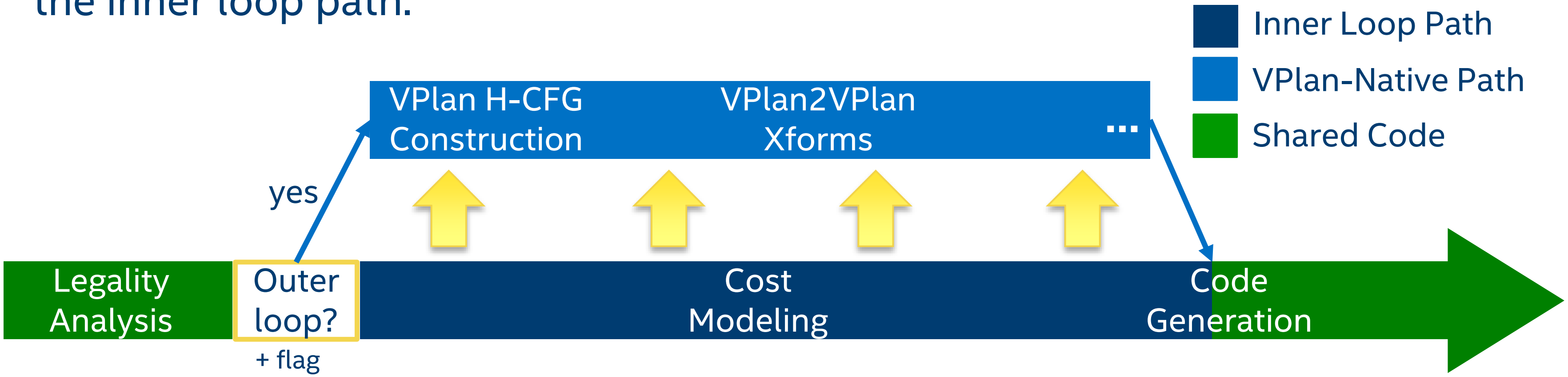
- Outer loops will be handled in the VPlan-native path (under a feature flag).
- In the VPlan-native path, VPlan is built upfront in the vectorization pipeline.
- Shared code between both paths will progressively increase until final convergence.



Convergence Towards a Single Vectorization Path

Main Goal: Incremental convergence towards a single VPlan-based vectorization path with support for inner and outer loops.

- Keeping both paths close will facilitate an incremental convergence without destabilizing the inner loop path.
- Convergence will be completed when the VPlan-native path is a superset of the inner loop path.



Advantages/Disadvantages: Reasonable Trade-off

Disadvantages:

- Alternative path for outer loop vectorization.

Advantages:

- Keeps inner loop vectorization path stable.
- Brings VPlan upfront in the vectorization pipeline.
  - Enables the development of new vectorization technology.
  - Enables more participation opportunities for the community.
- Allows to incrementally port existing algorithms to VPlan and extend them to support outer loops.

PARTICIPATION OPPORTUNITIES FOR THE LLVM COMMUNITY

Patch Series: Enabling Participation Opportunities

Our patch series will provide the basis for the community to participate in the design and development of VPlan.

- Patch Series #1: Trivial Outer Loops w/ Trivially Uniform Branches.
  - Any VPlan-to-VPlan transformation that involves modifying the H-CFG or the VPInstructions.
    - SLP-aware loop vectorization (ARM). RFC: http://lists.llvm.org/pipermail/llvm-dev/2018-February/121000.html
  - Remove the initial constraints of the supported outer loop nests.
- Patch Series #2: Support for Trivially Divergent Branches.
  - Advanced predication optimizations: e.g., all-one/all-zero vector mask bypasses (Region Vectorizer, Saarland U?).

Patch Series: Enabling Participation Opportunities

Our patch series will provide the basis for the community to participate in the design and development of VPlan.

- Patch Series #3: Support for Non-Trivial Uniform Branches.
  - Advanced divergence analysis technology (Region Vectorizer, Saarland U).
- Patch Series #4 & #5: Support for divergent inner loops and multi-exit inner loops.
  - Models to follow for other VPlan-to-VPlan transformations: search loop vectorization (OpenMP 'early_exit' clause), conditional last privates, etc.

Participation Beyond the Patch Series

- Outer loop auto-vectorization: extend legality analysis and cost model to outer loops.
- Refactoring components of the inner loop vectorizer so that they can be used in both the VPlan-native and the inner loop vectorizer paths.
- Final convergence: porting existing features in the inner loop path to the VPlan-native path (e.g., SinkScalar, IV type shrinking, etc.).
  - Comparison of VPlan-native vs. Inner loop path.
  - Follow-up: extend these features to support outer loops.
- Code reviews ☺.
- Many more…

Conclusion

- Outer loop vectorization is a BIG missing feature in LV.
- Our patch series bring incremental support for outer loop explicit vectorization to LV and the infrastructure to develop new vectorization technology.
- The VPlan-native vectorization path is a reasonable approach towards final convergence into a single and fully capable VPlan-based vectorization path.
- This proposal enables a lot of participation opportunities for the community to get involved with VPlan.

Software

diego.caballero@intel.com

BACKUP