Look-Ahead SLP: Auto-vectorization in the presence of commutative operations

Vasileios Porpodas[1], **Rodrigo Rocha**[2] and Luís Góes[3]

Intel Santa Clara, USA[1]
University of Edinburgh, UK[2]
PUC Minas, Brazil[3]

EuroLLVM 2018

Legal Disclaimer & Optimization Notice

Straight-Line Code Vectorizer

- LLVM and GCC provide SLP
  - Superword Level Parallelism [Larsen PLDI'00]
  - Bottom-Up SLP

Straight-Line Code Vectorizer

- LLVM and GCC provide SLP
  - Superword Level Parallelism [Larsen PLDI'00]
  - Bottom-Up SLP
- SLP and loop-vectorizer complement each other:

Straight-Line Code Vectorizer

- LLVM and GCC provide SLP
  - Superword Level Parallelism [Larsen PLDI'00]
  - Bottom-Up SLP
- SLP and loop-vectorizer complement each other:
  - Even if loop vectorizer fails, SLP could partly succeed

Straight-Line Code Vectorizer

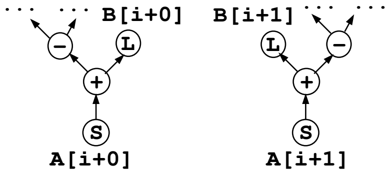
- LLVM and GCC provide SLP
  - Superword Level Parallelism [Larsen PLDI'00]
  - Bottom-Up SLP
- SLP and loop-vectorizer complement each other:
  - Even if loop vectorizer fails, SLP could partly succeed
- Compilers usually run SLP after the Loop Vectorizer

Why commutative operations matter ?

- Operands can be reordered
- It can change the shape of the DAG
- Since SLP operates on DAGs, it affects the SLP's vectorization effectiveness

```
sub1 = ... - ...;
sub2 = ... - ...;
A[i+0] = sub1   + B[i+0];
A[i+1] = B[i+1] + sub2;
```

DAGs



SLP not effective in more complex cases

- SLP reordering not effective for:

SLP not effective in more complex cases

- SLP reordering not effective for:
  1. Opcode mismatch further up the graph

SLP not effective in more complex cases

- SLP reordering not effective for:
  1. Opcode mismatch further up the graph
  2. Load address mismatch further up the graph

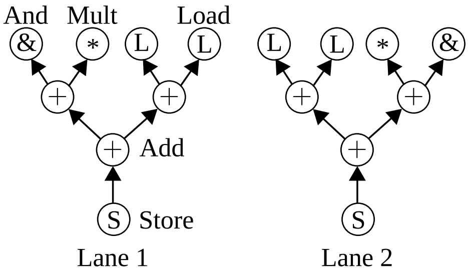
SLP not effective in more complex cases

- SLP reordering not effective for:
  1. Opcode mismatch further up the graph
  2. Load address mismatch further up the graph
  3. Reodering across chains of commutative operations*

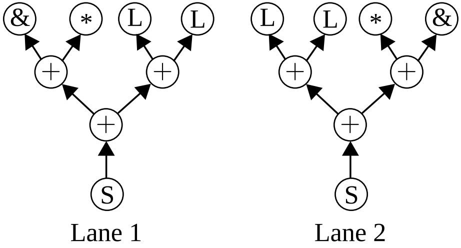
SLP not effective in more complex cases

- SLP reordering not effective for:
  1. Opcode mismatch further up the graph
  2. Load address mismatch further up the graph
  3. Reodering across chains of commutative operations*
- Look-Ahead SLP (LSLP) provides a solution to all three.

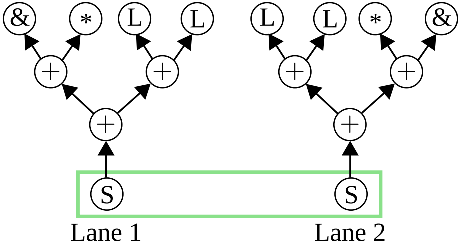
Reordering Chains of Commutative Operations

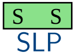


Reordering Chains of Commutative Operations



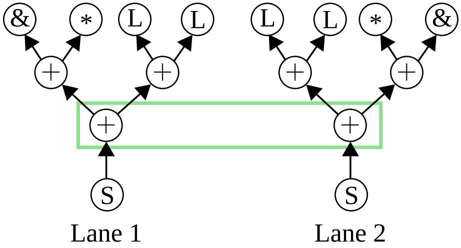
Reordering Chains of Commutative Operations

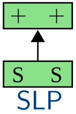




🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

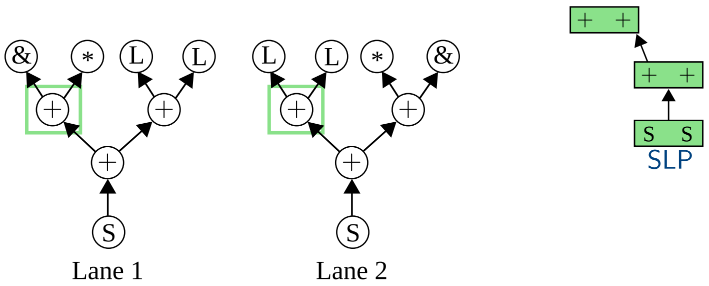
Reordering Chains of Commutative Operations





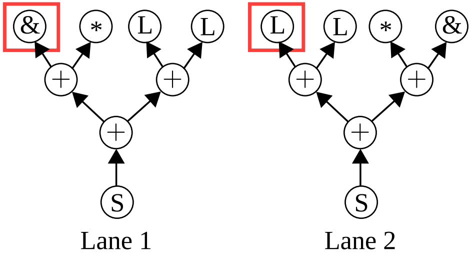
🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

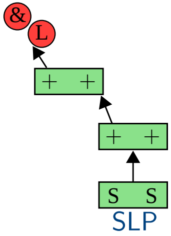
Reordering Chains of Commutative Operations



● Non−Vectorizable 🟩 Vectorizable +/−#Cost

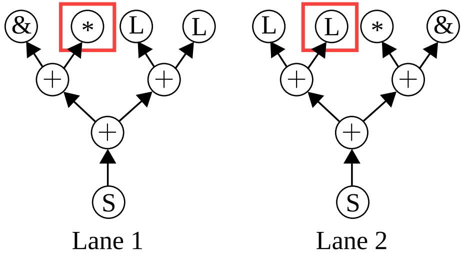
Reordering Chains of Commutative Operations

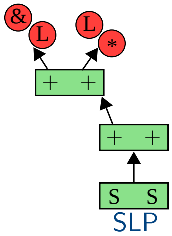




🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

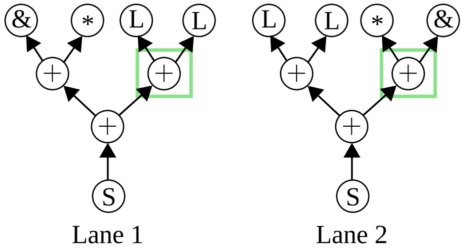
Reordering Chains of Commutative Operations

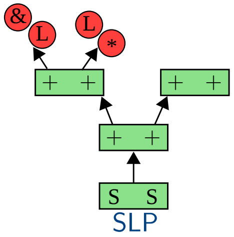




🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

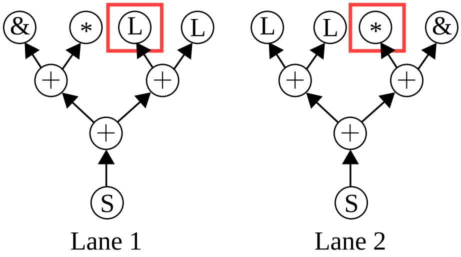
Reordering Chains of Commutative Operations

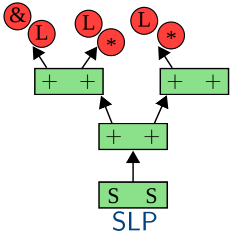




🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

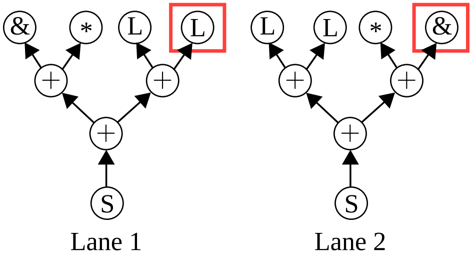
Reordering Chains of Commutative Operations

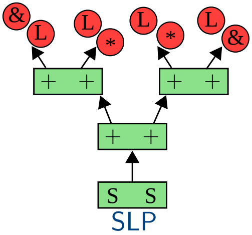




🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

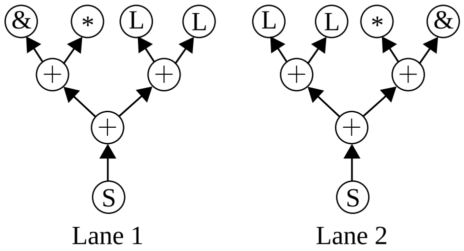
Reordering Chains of Commutative Operations

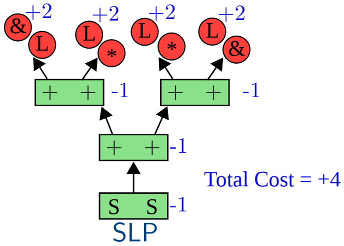




🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

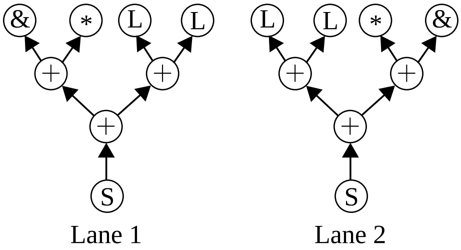
Reordering Chains of Commutative Operations

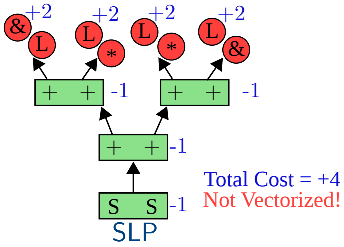




🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

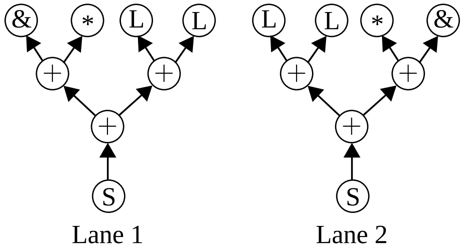
Reordering Chains of Commutative Operations

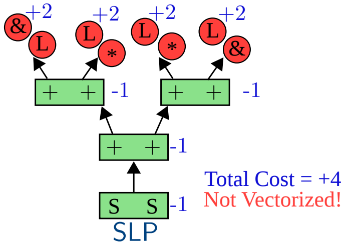


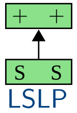


🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations

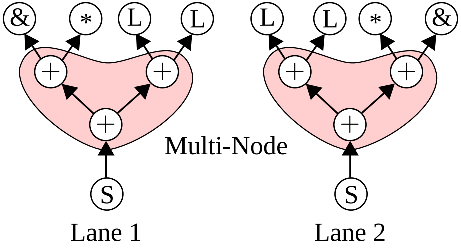


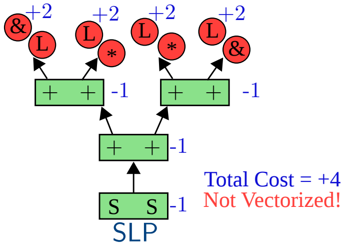




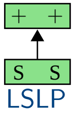
🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations



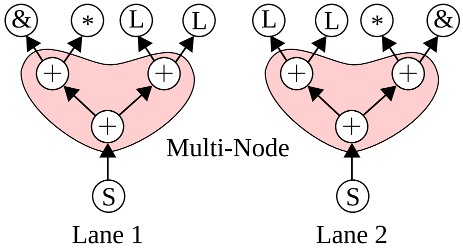


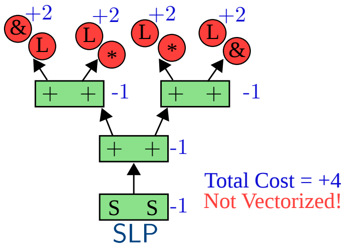
Total Cost = +4
Not Vectorized!



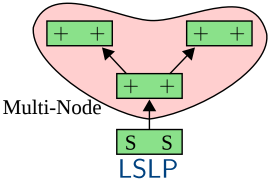
🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations



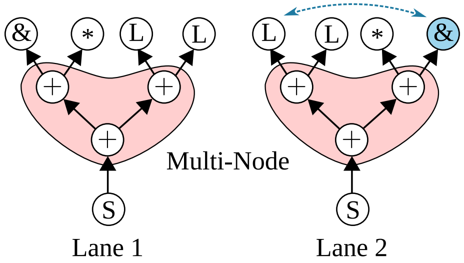


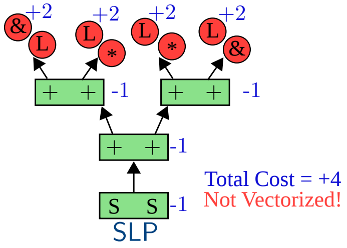
Total Cost = +4
Not Vectorized!

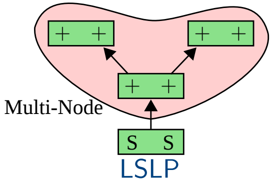


🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations

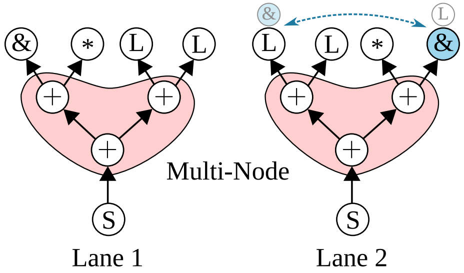


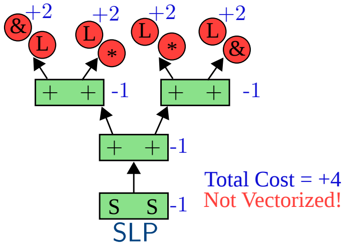


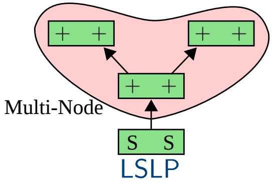


🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations

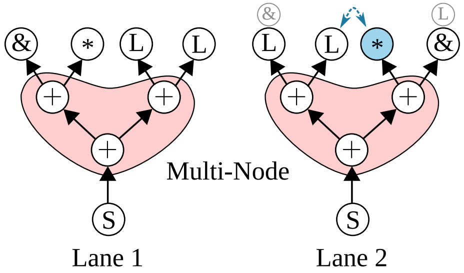


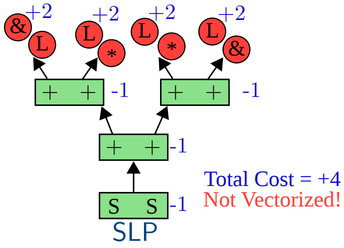


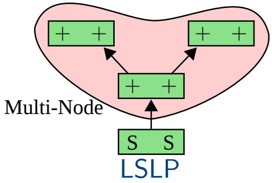


🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations

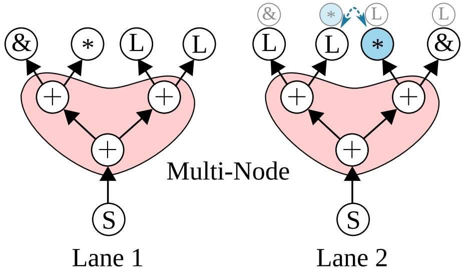


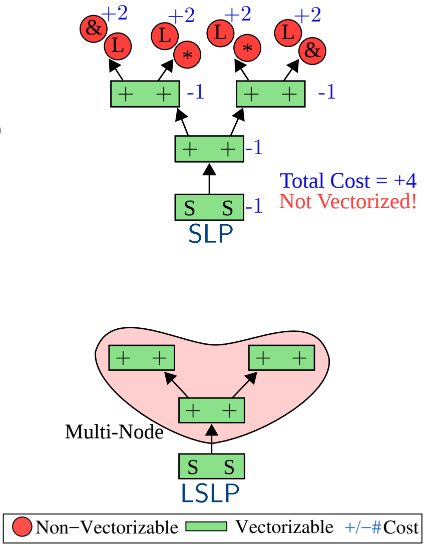




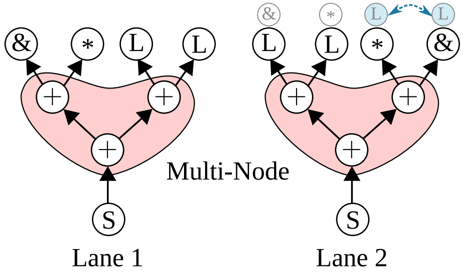
🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

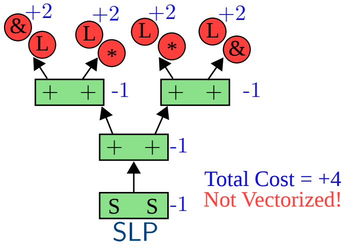
Reordering Chains of Commutative Operations



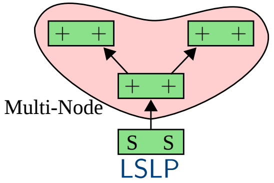


Reordering Chains of Commutative Operations



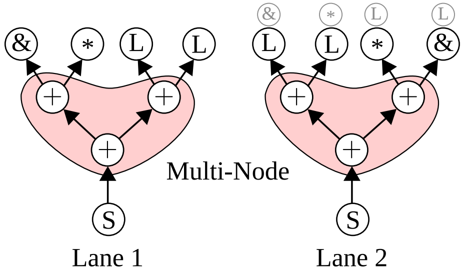


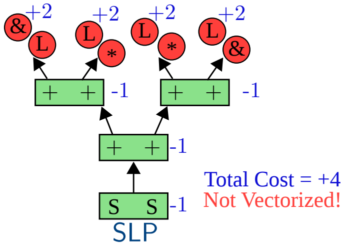
Total Cost = +4
Not Vectorized!

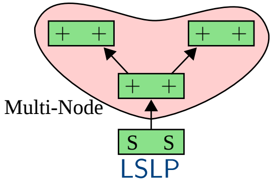


🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations

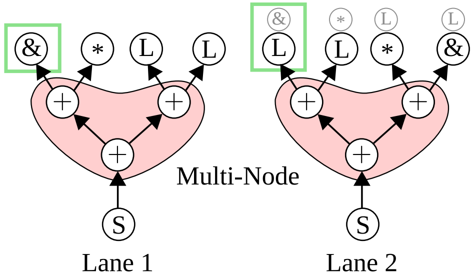


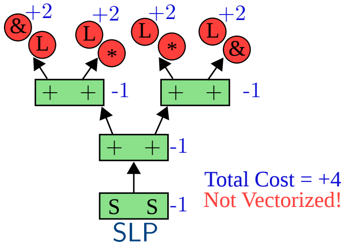


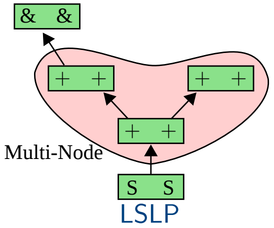


🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations

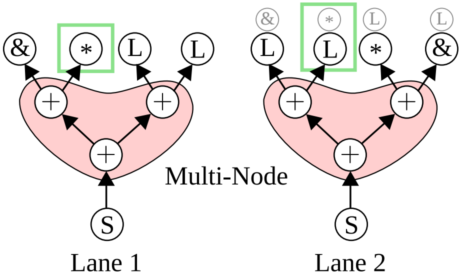


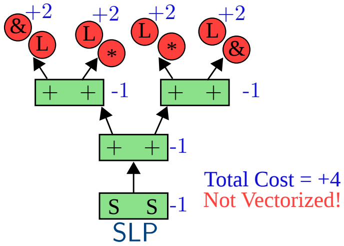


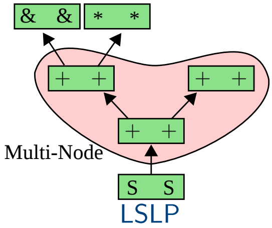


🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations

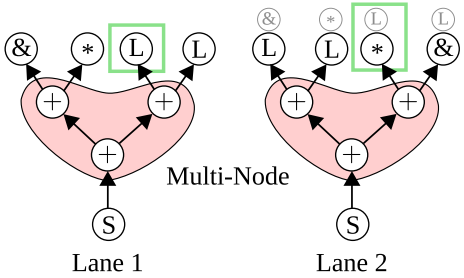


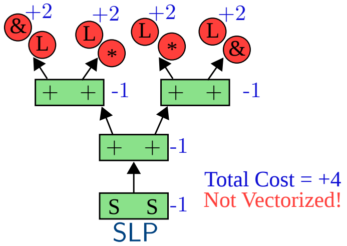


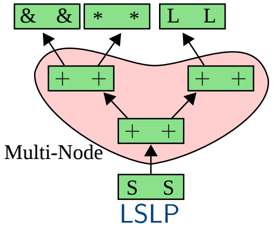


🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations

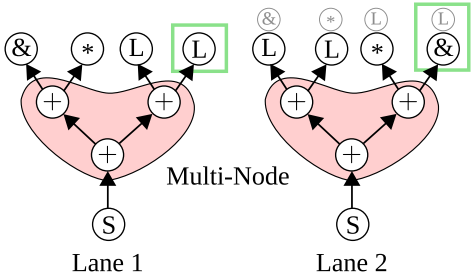


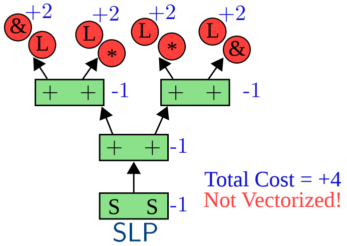


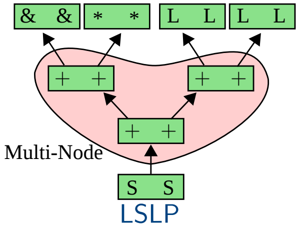


🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations

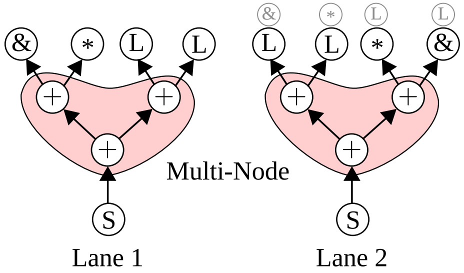


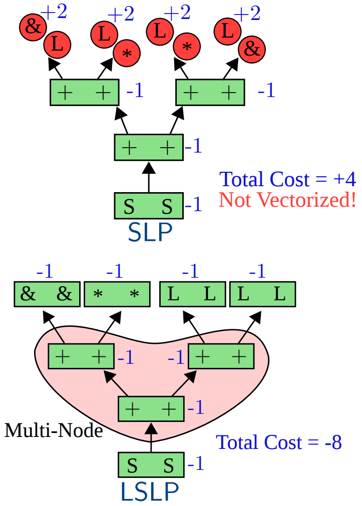




🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

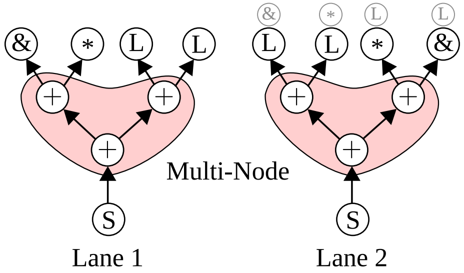
Reordering Chains of Commutative Operations

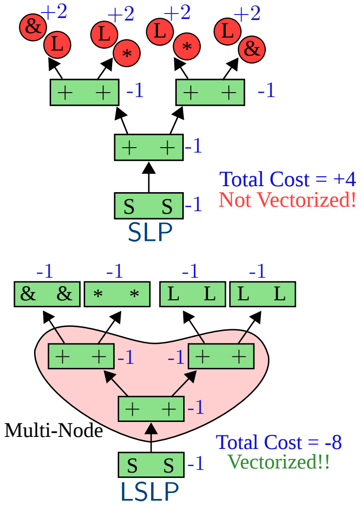




🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Reordering Chains of Commutative Operations





🟥 Non−Vectorizable 🟩 Vectorizable +/−#Cost

Experimental Setup

- Target: Intel Core i5-6440HQ Skylake CPU
- Kernels from SPEC CPU2006
- We evaluated the following cases:

Experimental Setup

- Target: Intel Core i5-6440HQ Skylake CPU
- Kernels from SPEC CPU2006
- We evaluated the following cases:
  1. All loop, SLP and LSLP vectorizers disabled (O3)

Experimental Setup

- Target: Intel Core i5-6440HQ Skylake CPU
- Kernels from SPEC CPU2006
- We evaluated the following cases:
  1. All loop, SLP and LSLP vectorizers disabled (O3)
  2. O3 + SLP enabled but No Reordering (SLP-NR)

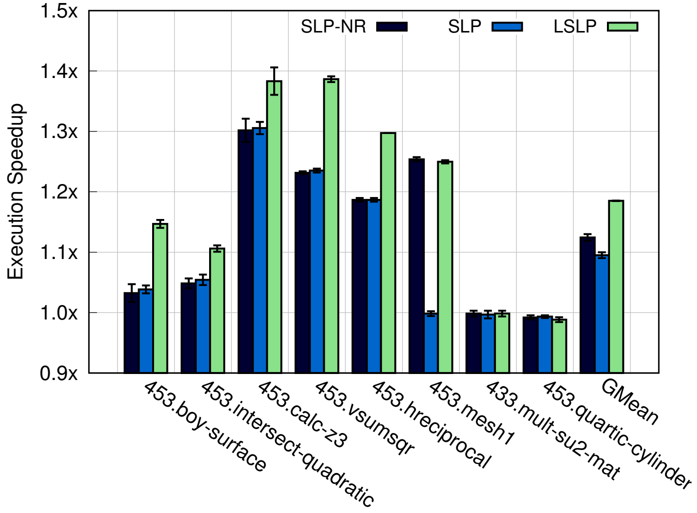
Experimental Setup

- Target: Intel Core i5-6440HQ Skylake CPU
- Kernels from SPEC CPU2006
- We evaluated the following cases:
  1. All loop, SLP and LSLP vectorizers disabled (O3)
  2. O3 + SLP enabled but No Reordering (SLP-NR)
  3. O3 + SLP enabled (SLP)

Experimental Setup

- Target: Intel Core i5-6440HQ Skylake CPU
- Kernels from SPEC CPU2006
- We evaluated the following cases:
  1. All loop, SLP and LSLP vectorizers disabled (O3)
  2. O3 + SLP enabled but No Reordering (SLP-NR)
  3. O3 + SLP enabled (SLP)
  4. O3 + LSLP enabled (LSLP)

Performance (normalized to O3)



Conclusion

- LSLP introduces an effective scheme for dealing with commutative operations.
- Look-Ahead SLP: Auto-vectorization in the presence of commutative operations (CGO 2018)