arm

Timing-agnostic SVE Analysis of geophysics kernel
Background on geophysical stencils

<table>
<thead>
<tr>
<th>Sync/Comms-avoiding strategies</th>
<th>Leveraging hardware features</th>
<th>Productivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Blocking (mainly spatial) but traction for temporal algorithms</td>
<td>• Heterogeneous architectures.</td>
<td>• High-level, DSL-like approaches (e.g. Devito, Patus, Yask)</td>
</tr>
<tr>
<td>• Runtime systems (e.g. task-based)</td>
<td>• SIMD.</td>
<td>• Directive-based, source-to-source</td>
</tr>
<tr>
<td>• Runtime systems (e.g. task-based)</td>
<td>• Mixed-precision (converged architectures).</td>
<td>• Auto-tuning, machine learning</td>
</tr>
</tbody>
</table>

**Refs**

<table>
<thead>
<tr>
<th>Sync/Comms-avoiding strategies</th>
<th>Leveraging hardware features</th>
<th>Productivity</th>
</tr>
</thead>
</table>

- Coupling of HPC and Numerical methods approaches: *Mercerat et al. (2009), Breuer et al. (2016).*
- Performance characterization and projection: *C.Andreolli et al. (2011), R.Cruz & M.Araya-Polo (2011).*
- Impact of the Absorbing Boundary Conditions and integration in complex applications: *M.Christen et al.(2011).*
<table>
<thead>
<tr>
<th>Company</th>
<th>Chip</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marvell (Cavium)</td>
<td>THUNDERX</td>
<td>Ampere (X-Gene)</td>
</tr>
<tr>
<td>Ampere (X-Gene)</td>
<td></td>
<td>Fujitsu</td>
</tr>
<tr>
<td>Huawei (HiSilicon)</td>
<td>Hi1612</td>
<td>Amazon (Annapurna)</td>
</tr>
<tr>
<td>EPI / SiPearl</td>
<td></td>
<td>Other</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Arm is ubiquitous

• Four NUMA Regions
• SVE-capable (512-bits)

• Various interfaces (more hierarchy)
• Diversity of accelerators
Theoretical roofline

Data from publicly available performance information.
What makes it a **Scalable Vector Extension**?

- **There is no preferred vector length**
  - The vector length (VL) is a hardware choice, 128-2048b, in increments of 128b
  - A Vector Length Agnostic (VLA) programming adjusts dynamically to the available VL

- **SVE addresses traditional barriers to auto-vectorization**
  - Software-managed speculative vectorization of uncounted loops
  - Extract more data-level parallelism (DLP) from existing C/C++/Fortran source code

- **SVE is a new approach to vectorization, not an iteration on existing ISAs (e.g. NEON)**
  - SVE is a separate, optional extension with a new set of instruction encodings
  - Initial focus is **HPC** and general-purpose server, not media/image processing
Dynamic Binary instrumentation

- Diversity of languages, frameworks, dependencies
- Region of Interest feature for full-fledged code

Walk through – source code modification

```c
int main(int argc, char *argv[]) {
    float* s_prev;
    float* s_next;
    float* s_vel;

    #define __START_TRACE() {asm volatile (".inst 0x2520e020");}
    #define __STOP_TRACE() {asm volatile (".inst 0x2520e040");}

    iso_init(nx,ny,nz,s_prev,s_next,s_vel);
    __START_TRACE();
    iso_compute(s_prev,s_next,s_vel,cval,nx,ny,nz,timesteps);
    __STOP_TRACE();

    for (i = SIZE; i < (nx - SIZE); i++) {
        for (j = SIZE; j < (ny - SIZE); j++) {
            #pragma clang loop vectorize(enable)
            for (k = SIZE; k < (nz - SIZE); k++) {
            
```

• Region of Interest feature to capture hotspots: `__START_TRACE() / __STOP_TRACE()`
• Standard pragmas, keywords to enhance vectorization (e.g. from LLVM)
Walk through - vectorization

- Add "+sve" to generate SVE instructions
- Use "-Rpass" or "opt-report" flag for compiler insights
Walk through - scripts

- Same binary, varying vector length
- Various post-processing scripts are included: “merge, analyze, flops-bytes ...”
Walk through - metrics

Native instr. breakdown

SVE instr. breakdown

SVE Mem operations

```
# SUMMARY
#---------------------------
# VL: 512 bits
# Memtrace file: merge.log
# Output: stdout

load/store/total, Total, SVE, non-SVE, SVE-contiguous, SVE-contig-disLanes, SVE-gather/scatter, SVE-gather/scatter-allLanes, SVE-gather/scatter-Lanes
load, 24291847, 16793088, 7498759, 16793088, 1469395, 2099136, 0, 0, 0
store, 2929421, 430592, 2498829, 430592, 376768, 53824, 0, 0, 0
total, 27221268, 17223680, 9997588, 17223680, 15070720, 2152960, 0, 0, 0
```
Acoustic stencil - 8-th order (ISO)

NEON Vectorization study

- NEON (128-bits)
- LLVM and GNU toolchain.
- Speedup up to x3.7
Dynamic instructions breakdown

• Same binary for all SVE runs
• Impact of Amdhal law for large vector length

⇒ Implementation-dependent
Acoustic stencil - **8-th order (ISO)**

**SVE instructions breakdown**

- Decrease of instruction count as we increase the vector length
- Ratios (Floating Point/ Memory or Loads/Stores instructions).

⇒ Stencil-based kernels characterization
Scalar instructions breakdown

- Instructions count reduction with larger SVE vector.
  => Fewer loop iteration (e.g. conditional branch – bcond)
SVE Vector lane utilization

- Number of bytes transferred
- Very useful for application characterization (SIMD lanes, scatter/gather operations ....)
Thank You
Danke
Merci
Merci
Merci
ありがとうございます
감사합니다
धन्यवाद
شكرًا
ধন্যবাদ
תודה