Arm SVE Users Meeting

SVE Experiences at Arm Research

Alex Rico

Supercomputing 2017
Denver, CO, USA

Gather / Scatter

arm

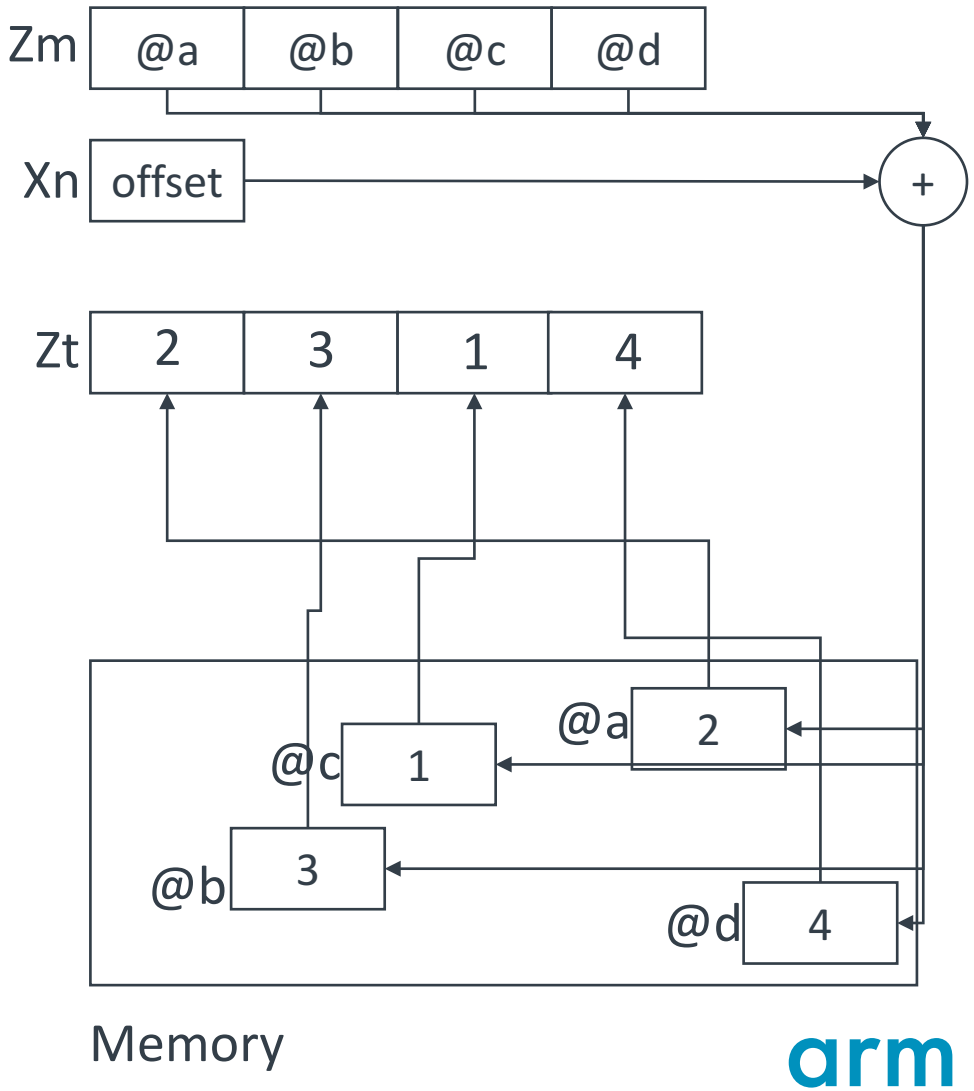
Gather/Scatter Operations are Good and Evil

Enable vectorization of codes with non-adjacent accesses on adjacent lanes

Performance implementation dependent

Worst case one separate access per element

```
LD1D <Zt>.D, Ps/Z [<Xn>, <Zm>.D]
```

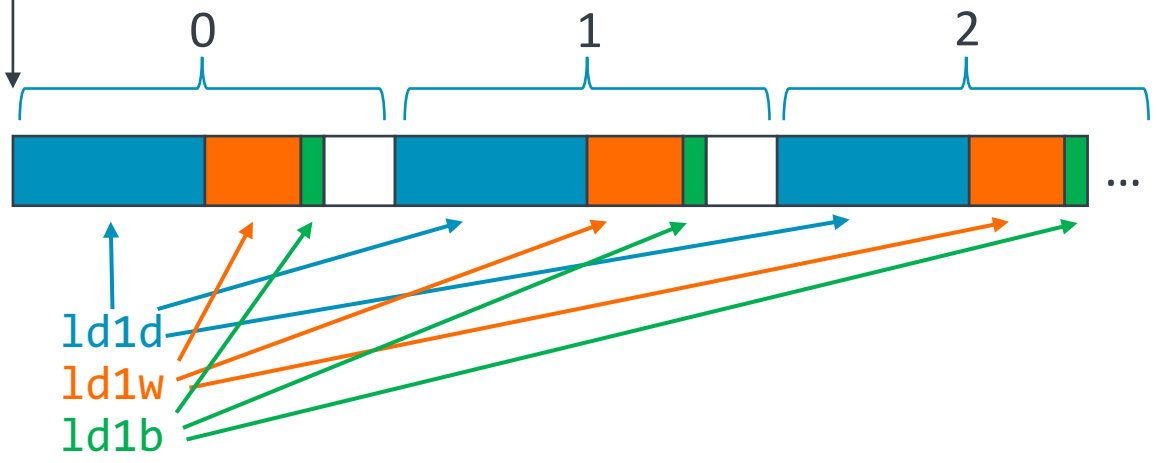


arm

Array of Structures vs. Structure of Arrays

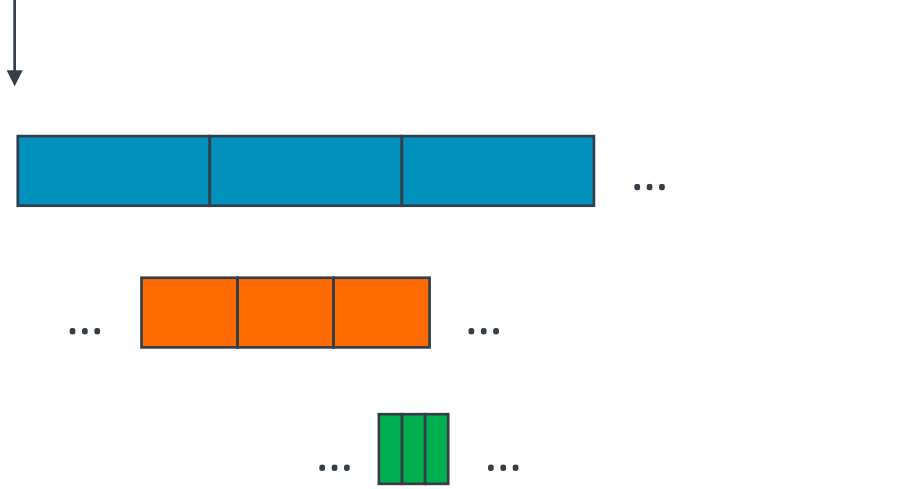
```
typedef struct {
    uin64_t num_projects;
    float caffeine;
    bool vim_nemacs;
} HPCProgrammer_t;

HPCProgrammer_t programmers[N];
```




```
typedef struct {
    uin64_t num_projects[N];
    float caffeine[N];
    bool vim_nemacs[N];
} HPCProgrammer_t;

HPCProgrammer_t programmers;
```



arm

SVE Non-Temporal Vector Instructions

- `LDNT1D { <Zt1>.D }, <Pglo>/Z, [<Xn|SP>, <Xm>, LSL #3]`
- `STNT1D { <Zt1>.D }, <Pglo>, [<Xn|SP>{, #<simm4>, MUL VL}]`

From the ARM ARM (Architecture Reference Manual):

*Non-temporal contiguous load and stores include a **hint to the memory system** that this is a "streaming" access, and the memory locations **are not expected to be accessed again soon** so do not need to be retained in local caches.*

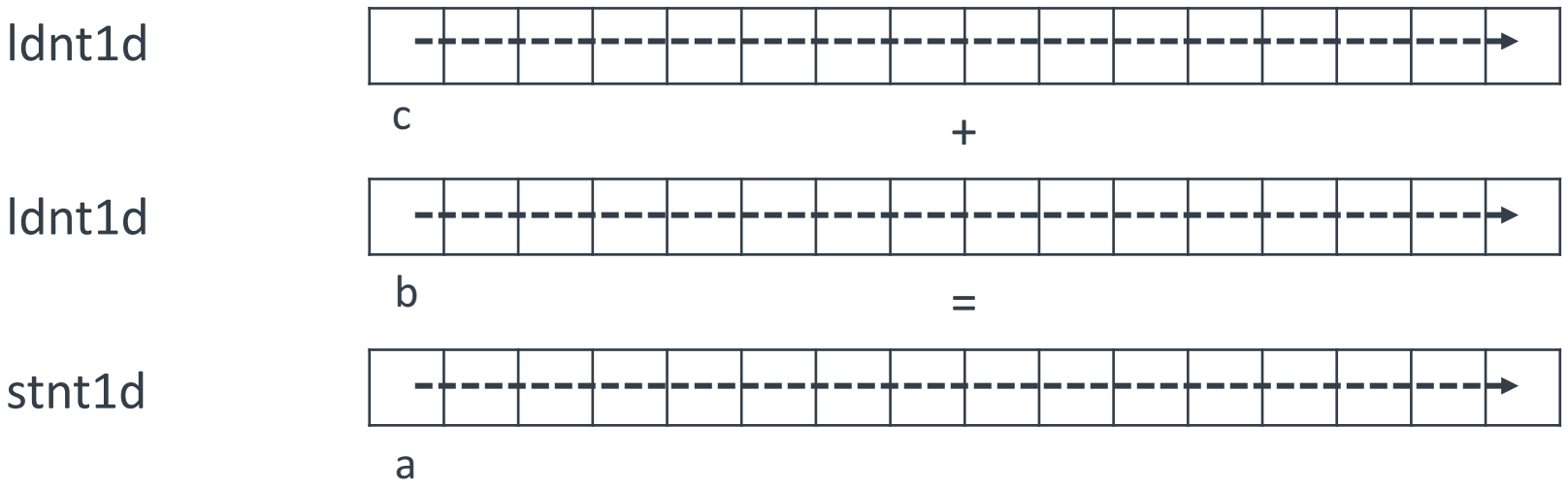
Being Non-temporal is Not Enough

Vector Addition

```
for (i=0; i<N; i++) {
    a[i] = b[i] + c[i];
}
```

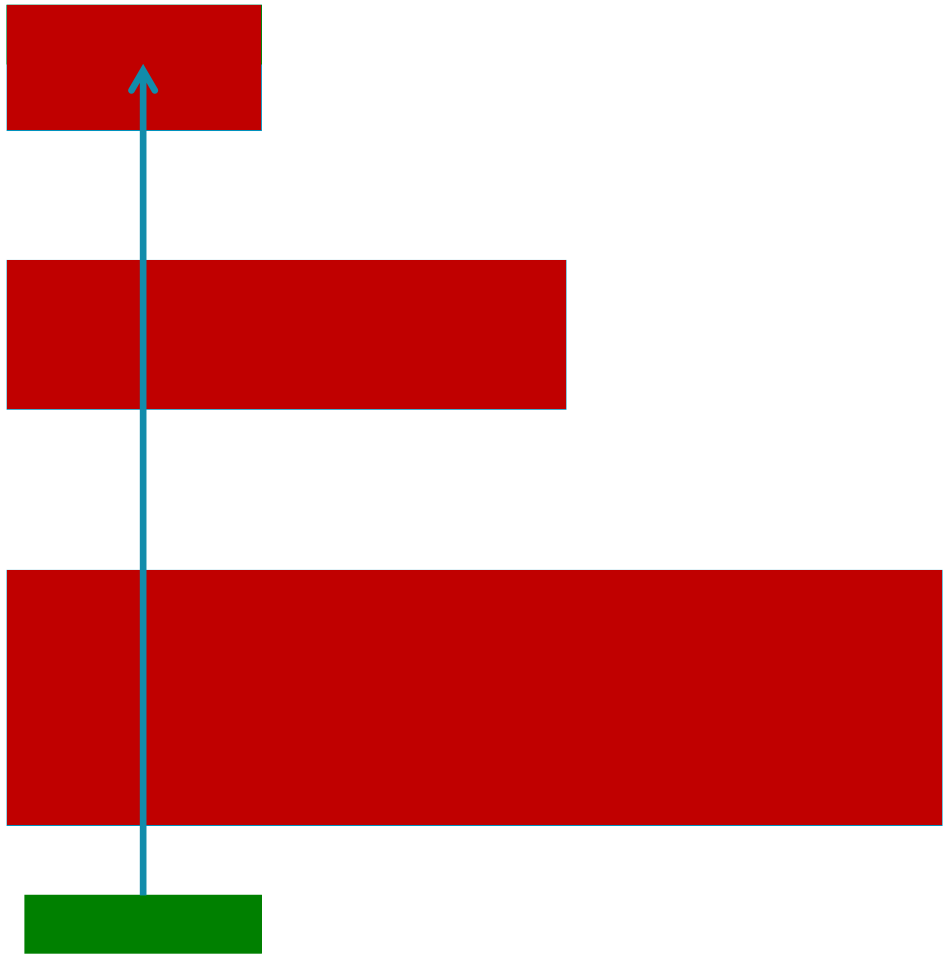
No benefit if all accesses are temporal

Target to leave space for *temporal* accesses

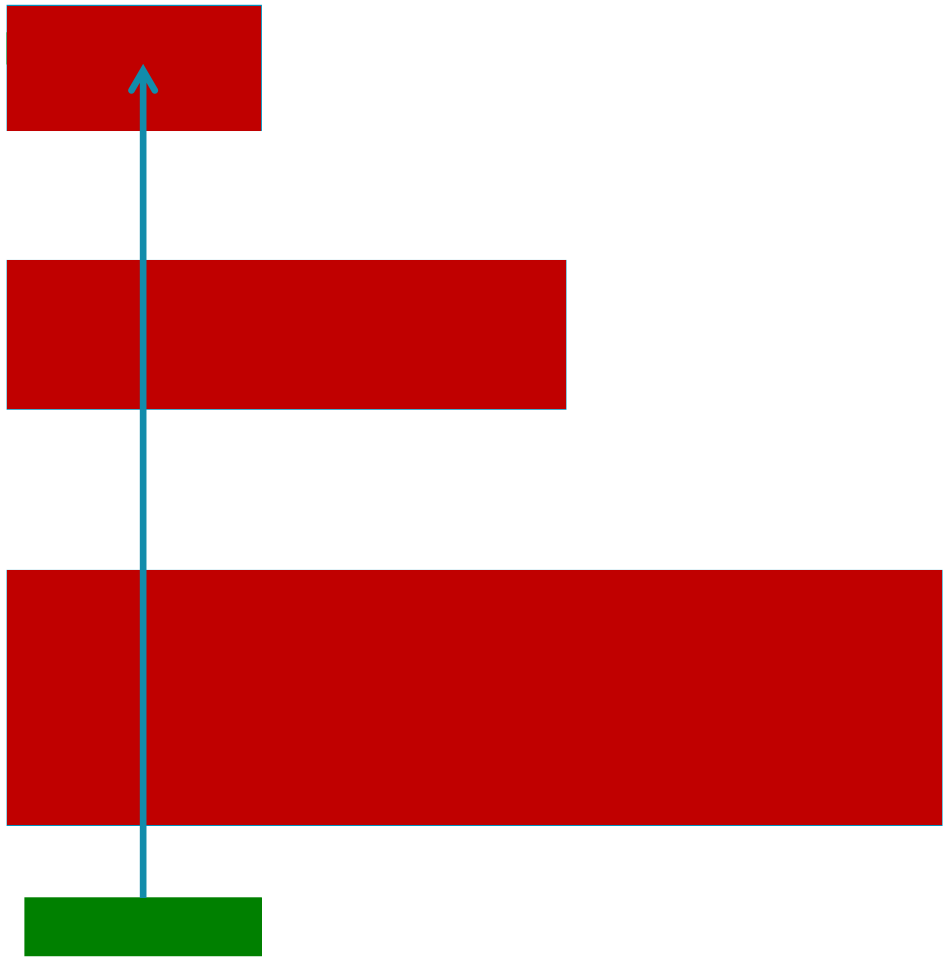


arm

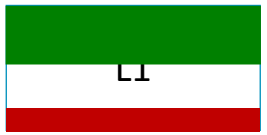
Mixed Temporal and Non-temporal

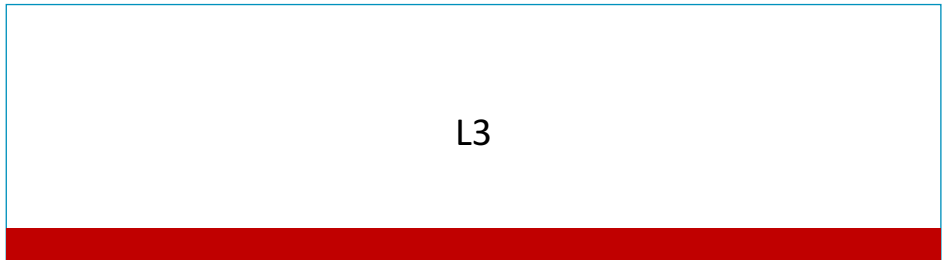


Mixed Temporal and Non-temporal



With non-temporal gather
And LRU allocation

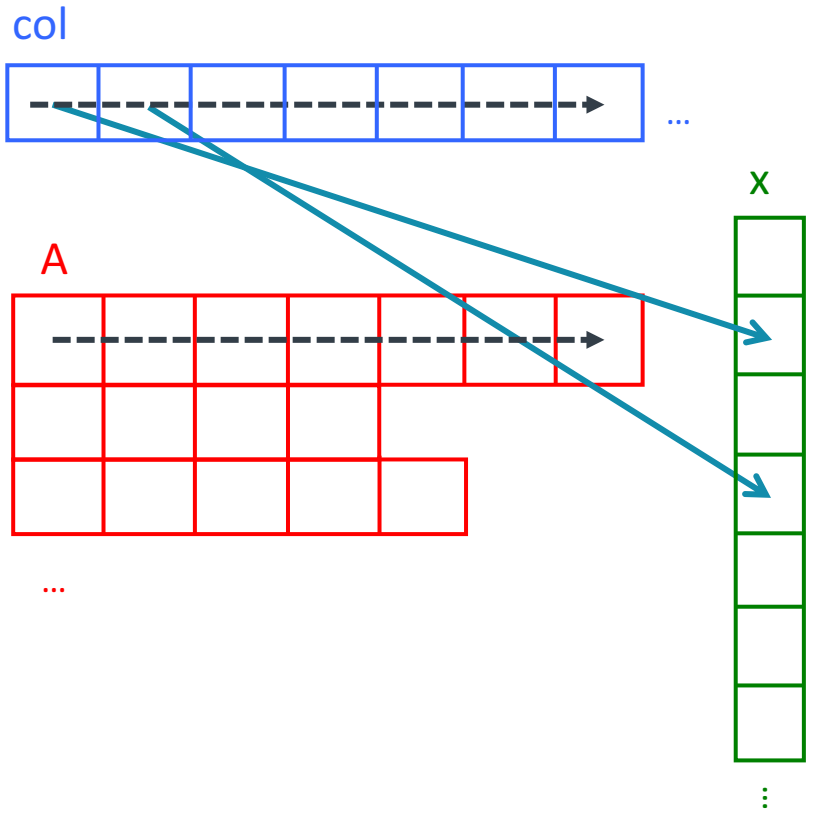




arm

Sparse Matrix Vector

```
for(m=row_start[j]; m<row_start[j+1]; m++)
          y[j] += A[m] * x[col[m]];
```




```
                                        ldnt1d    z3.d, p1/z, [x0]

whilelt p1.d, xzr, x4
ld1sw   z1.d, p1/z, [x2]                // z1 = &col[]
ld1d    z2.d, p1/z, [x1, z1.d, lsl #3]  // z2 = &x[&col[]]
ld1d    z3.d, p1/z, [x0]                // z3 = &A[]

fmla    z0.d, p1/m, z2.d, z3.d

add     x2, x2, x7      // add half vector reg length (in bytes)
addvl   x0, x0, 1       // add vector register length (in bytes)
subs    x4, x4, x8      // Remaining length
bgt     .L4

faddv   d0, p0, z0.d    // result for y[j]
```


arm

SVE Programming

Assembly	Full ISA Specification: The Scalable Vector Extension for Armv8-A Lots of worked examples in A sneak peek into SVE and VLA programming
Intrinsics	Arm C Language Extensions for SVE Arm Scalable Vector Extensions and application to Machine Learning
Compiler	Autovectorization – GCC, Arm Compiler for HPC, Cray, Fujitsu Help the compiler: OpenMP #pragma omp parallel for **simd**

arm

Test Your Code

Arm Instruction Emulator (ArmIE): Functional and instruction-level statistics

→ Docker image available!

gem5 SVE support

- Functional support
- Completing timing support in o3 core model

Docker demo