The Spatter Benchmark

Or: Benchmarking and Modeling Sparse Memory Accesses for Heterogeneous Systems

Patrick Lavin, Jeffrey Young, Jason Riedy, Rich Vuduc
Purpose

• Dense memory access is well understood, but it is difficult to predict how a memory system will respond in irregular scenarios

  • Indirection, poor spatial and temporal locality

• Spatter allows us to view performance changes across architectures, so that we can better understand their differences

• You can’t understand what you can’t measure
Spatter

- Memory benchmark based on a scatter/gather kernels
  - Scatter $Y[j[:]] = X[:]$  
  - Gather $Y[:] = X[i[:]]$
  - SG $Y[j[:]] = X[i[:]]$

- Designed to model sparse data movement in applications like SuperLU and kernels like SpGEMM

- Includes effects of indirection and random or sparse access
Configuration

• Backend - OpenMP, OpenCL, or CUDA

• Work per thread (work item)

• CUDA (or OpenCL) block size

• Buffer sizes, cache-ability, access stride
Examples
• Gather elements of x, then do a dot product with data in A.

```python
for (i in range(nrows)):
    indices ← row[i] : row[i+1]
    gather(tmp, x, col[indices])
    y[i] = dot_prod(val[indices], tmp)
```
Examples - CSC SpMV

- Scale some column of A by the value in x, then scatter-accumulate into y.

```
for (i in range(ncols)):
    indices ← col[i] : col[i+1]
    tmp ← vector_scale(val[indices], x[i])
    scatter_accum(y, row[indices], tmp)
```
Examples - SpGEMM

- Scatter-accumulate columns of A corresponding to non-zero entries in a column of B into a dense SPA buffer. Gather SPA into C.

```
for (j in range(ncols) :
    SPA = 0  //dense accumulation buffer
    for non-zero B(k,j) :
        scatter_accum(SPA, A(:,k)*B(k,j))
        gather(C.val, SPA)
        gather(C.row, which(SPA))
```

Algorithm from Buluç and Gilbert: Parallel Sparse Matrix-Matrix Multiplication and Indexing: Implementation and Experiments
https://doi.org/10.1137/110848244
Examples - Vectorization

- Some forms of vectorization may naturally lead to Gather/Scatter operations

```python
for (j in range(N)):
    for (i in range(4)):
        out[j] += data[i, j]

for (j = 0; j < N; j+=8):
    for (i in range(4)):
        gather_accum_stride(temp,j+i, 8, 4)  //gather 8 elements,
                                            //gap size 4

out[j:j+8] = temp
```

Column-Major
Example - SuperLU

- SuperLU spends a large portion of its runtime on just scattering data.
Benchmark Output
Performance Exploration

Uniform Stride

Gather Bandwidth
Tesla P100, CUDA Backend

Scatter Bandwidth
BDW, OpenMP Backend

Power8, OpenMP Backend

Sparsity
- 1
- 2
- 4
- 8
- 16
- 32
- 64
- 128

Eff. Bandwidth (% of BabelStream) vs Work per thread
Uniform Stride Access

**Gather**

- **Sparsity**
- **Effective Bandwidth (% of BabelStream)**

**Scatter**

- **Gather**
- **Scatter**

|----------------|---------|------------|-----------|---------|-----------|------------|---------|----------------|--------------|

**Impact of Access Sparsity**
Random Access

Gather, Uniform

Sparsity vs Effective Bandwidth (% of BabelStream)

Device–Backend
- K40c–CUDA
- P100–CUDA
- Titan Xp–CUDA

Sparsity vs Effective Bandwidth (% of BabelStream)

Device–Backend
- BDW–OMP
- KNL–OMP
- Power8–OMP
- SNB–OMP
- ThunderX2–OMP
Energy Efficiency

Uniform Stride

Gather

Scatter

Device:
- BDW–OMP
- GV100–CUDA
- K40c–CUDA
- KNL–OMP
- P100–CUDA
- Power8–OMP
- SNB–OMP
- ThunderX2–OMP
- Titan Xp–CUDA
What’s Next?

- Partner with industry to run on upcoming systems
- Evaluation of slightly more complex synthetic traces
  - Mostly stride-1
  - Write collisions
- Gather/Scatter traces from real (DOE) mini-apps
- Measure impact of vector length (SVE and AVX) on generated code (and therefore cache performance)
- CILK backend for EMU, FPGA-specific OpenCL Backend
- More general kernels, with accumulation and a length buffer
- **Simplify to present a STREAM-like result**
More Info

• **Spatter.io**
  - Documentation
  - Guide to easily plot your GPU against ours

• ArXiv Pre-print
  - Spatter: A Benchmark Suite for Evaluating Sparse Access Patterns

• Code
  - [https://github.com/hpcgarage/spatter](https://github.com/hpcgarage/spatter)
Acknowledgements

This material is based upon work supported by the National Science Foundation under Award #1710371.
The Spatter Benchmark
(spatter.io)

Or: Benchmarking and Modeling Sparse Memory Accesses for Heterogeneous Systems

Patrick Lavin, Jeffery Young, Jason Riedy, Rich Vuduc