Narrow-band Topology Optimization on a Sparsely Populated Grid

SIGGRAPH Asia 2018  Presented at TOP Webinar

Haixiang Liu*1  Yuanming Hu*2  Bo Zhu23  Wojciech Matusik2  Eftychios Sifakis1

1University of Wisconsin-Madison  2MIT CSAIL  3Dartmouth College

*equally contributed
Design domain: 3D density field (array) of values in $[0,1]$

\[
\begin{align*}
\min \; & S(\rho, u) = u^T K(\rho) u \\
\text{s.t.} \; & F(\rho, u) = K(\rho) u - f = 0 \\
& V(\rho) - \hat{V} \leq 0,
\end{align*}
\]

Minimal compliance objective (deformation)

Linear FEM

Volume constraint

[Sigmund, A 99 line topology optimization code written in Matlab]
Topography Optimization

Density Update

FEM Solve
Grid Representation of Density
14 million voxels

Single GPU

[Wu et al. 2017]

1 billion voxels

HPC Cluster

8000 cores

[Aage et al. 2017]

1 billion voxels

Ours

Single Workstation

56 cores
11,520,000,000 background voxels (3TB memory)
1,040,875,347 **active** voxels (8.6%)
Virtual Memory System

CPU

Page Table & TLB

Physical Memory

Virtual memory

Physical memory

Touched

Unused

Page table & mapping
Sparsely Paged Grid [Setaluri et al. 2014]
Narrowband tracking

Sparse blocks of 4x4x8 nodes
Up to 3.1x acceleration
Narrowband Evolution

Filter

Expand

Evolve & Shrink
Multigrid FEM Solver

MGPCG Time Complexity: $O(n)$
Aggressively Optimized Multigrid FEM Solver

- Optimized fine-level operator
- Matrix-free Galerkin-coarsened operator construction
- Improved eight-color Gauss-Seidel smoother

...
Vectorization & Fused multiply and add (FMA)

CPU
- Float point unit
- 2xFMA a x b + c

Page Table & TLB

Physical Memory

4.2G Hz x 2 FMA/cycle x 16 FLOPS/FMA x 4 cores = 538G FLOPs
Fine Level Stiffness Operator

\[ Ku = f \]

- Stiffness matrix
- Nodal displacement (Unknown)
- External force

Matrix-Free

\[ 2 \times 24^2 = 1,152 \text{ vectorized FMA inst. per node} \]
Fine Level Stiffness Operator
Fine Level Stiffness Operator
Main Memory
35.8 GB/s
256 cyc latency

L3 cache 2M/core
134.4 GB/s
42 cyc latency

L2 cache 256KB
268.8 GB/s
12 cyc latency

L1 data cache 32KB
403.2 GB/s
4 cyc latency

L2 Unified TLB (STLB)
4 KB/2MB pages - 1536 entries
1G pages - 16 entries

L1 Data TLB
4 KB pages - 64 entries
2/4 MB pages - 32 entries
1G pages - 4 entries

Integer Physical Registers
8 bytes per entry, 180 entries
1 cyc latency

Vector Physical Registers
32 byte entries, 168 entries
1 cyc latency

Execution Engine
4.20 GHz
134.4 G FLOP/s, i.e. 806.4 GB/s bandwidth requirement

* Figures are not drawn to scale.
* Instruction caches are omitted.
* Main memory BW is shared by all cores.

Closer to CPU, smaller capacity, lower latency, higher bandwidth.
Fine Level Operator

**SIMD Gather** is expensive
Access pattern is regular
**Blend** is cheap

```c
template <int di,int dj,int dk,class SPG_array>
__m512 VectorGet<di,dj,dk>(SPG_Array a,int64_t offset)
```
Matrix-Free Galerkin Coarsening

\[ K^{2h} = R_{2h \rightarrow h} K^h P_{h \rightarrow 2h} \]
Matrix-Free Galerkin Coarsening

Top level is matrix free
Matrix-matrix multiply is memory bound
Matrix-Free Galerkin Coarsening

\[ K^{4h} e_i = R_{2h \rightarrow 4h} R_{h \rightarrow 2h} K^{h} P_{h \rightarrow 2h} P_{2h \rightarrow 4h} e_i \]
Matrix-Free Galerkin Coarsening

\[ K^{4h} e_i = R_{2h \rightarrow 4h} R_{h \rightarrow 2h} K^h P_{h \rightarrow 2h} P_{2h \rightarrow 4h} e_i \]
Matrix-Free Galerkin Coarsening

Recursive algorithm
Stack allocated variables
Cell material is read once/level
Total time: 113.9 sec for 1.04 Billion voxels
~1.26 TFLOPS on Skylake SP
Modified Eight-Color Gauss-Seidel Smoother

Coupled Diagonal Smoothing
Shuffled Data Storage
CPU
Float point unit
2xFMA a x b + c

Page Table & TLB

L1 Cache

L2 Cache

L3 Cache

Physical Memory
CPU talks to caches in the unit of 64B cachelines.
Modified Eight-Color Gauss-Seidel Smoother

Original Data Layout

Shuffled Data Layout

64-byte Cacheline

Comparing to matrix-free multiply
same amount memory access but 1/8 computation

Memory bound
Modified Eight-Color Gauss-Seidel Smoother

Effective bandwidth: 68GB/s (out of 120 GB/s)
Bicycle Wheels

- 53.56M voxels
- 984x984x204 background grid
- 1% volume fraction
- ~65 TopOpt iterations

: Dirichlet boundary

: loads
Plane Wing

401.53M voxels
1696x342x1971 background grid
20% volume fraction
43 TopOpt iterations
Summary:

1. Use a **sparse** grid to compactly store irregular geometry
2. Track the evolution of the structure using a **narrowband**
3. Do aggressive **low-level optimization** to speed up FEM solve
...and parallelism.

Moore’s law continues

Single-core performance stops growing

Instead of “better” cores, we have “more” cores (with wider SIMD)
Conclusion

Data Layout: Narrowband domain evolution
Vectorization & Data Layout: Optimized fine-level operator
Vectorization: Matrix-free Galerkin-coarsening
Data Layout: SPGrid-specific sparse matrix storage
Vectorization & Data Layout: Improved eight-color Gauss-Seidel smoother
Vectorization & Data precision: Mixed-precision scheme

Performance in numerical computation is mostly about data. Good data layout and effective vectorization are keys to performance.

We need better language & compiler to reduce the development cost for high-performance computer graphics.
...and we did write a compiler for that!

Taichi: a language for high-performance computation on spatially sparse data structures
(SIGGRAPH Asia 2019)

Write high-performance CPU/GPU solvers on sparse grids, in a programming language like Python!

https://github.com/taichi-dev/taichi
Code: https://github.com/yuanming-hu/spgrid_topo_opt

Thanks!
Questions are welcome!