SAND2015-1822C

Exceptional service in the national interest

Sandia National Laboratories

Parallel Graph Coloring

Erik G. Boman, Siva Rajamanickam

Sandia National Labs

SIAM CS&E '15

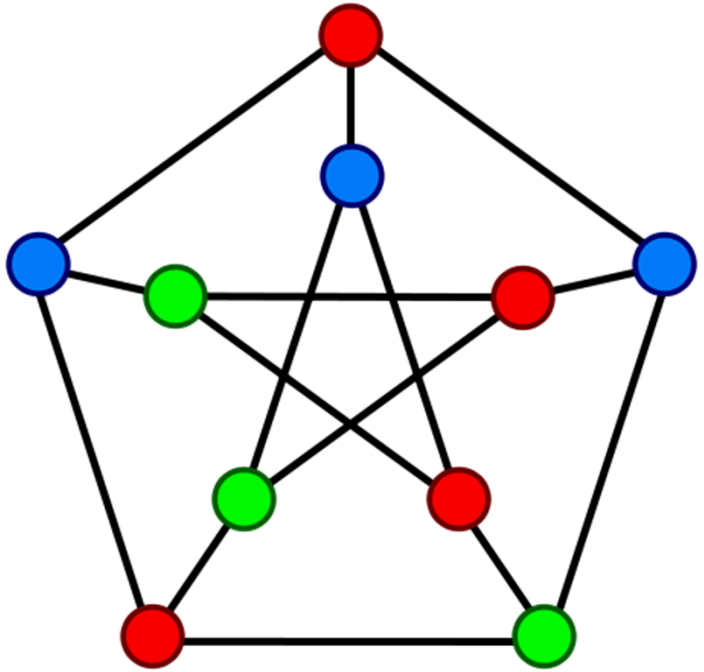
U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Background: Graph Coloring

Sandia National Laboratories

- Given a graph G(V,E), find a coloring c:V➔N such that no two adjacent vertices have the same color.
- We wish to (approximately) minimize the number of colors.
- Exact solution is NP-hard
- But linear-time greedy methods work well in practice!



Applications: Parallel scheduling, register allocation, sparse matrix ordering, preconditioning, AD

Software

Sandia National Laboratories

- Often embedded within other software (not exposed).
- General-purpose libraries:
  - Zoltan(2): Load balancing and combinatorial scientific computing.
  - ColPack: Focus on automatic differentiation.
  - cuSparse 7: GPU only.
- Easy to implement serial greedy but parallel is harder
- Our goal: Portable parallel shared-memory coloring for wide range of architectures (multicore, MIC, GPU)

Parallel Algorithms

Sandia National Laboratories

- The serial greedy algorithm is inherently sequential; very difficult to parallelize.
- The two most important parallel algorithms:
  - Jones-Plassmann (’93): Color sequence of independent sets.
  - Gebremedhin-Manne (‘00): Speculative/iterative coloring. To improve parallelism, allow some mistakes (coloring conflicts); go back and fix them later.
- We prefer GM due to less synchronization
  - #colors are similar for both.
  - GM was shown faster on distributed-memory (Bozdag et al, 2008).
  - Conjecture this is true on shared-memory, too.

Multithreaded: Iterative Greedy Algorithm

Sandia National Laboratories

Proc IterativeGreedy ($G = (V, E)$)
U is set of vertices to be colored, and R to be recolored
while U is not empty
    1. Speculatively color vertices
    for $v \in U$ **in parallel**
        for each neighbor w of v
            Mark color[w] as forbidden to v
        Assign smallest available value to color[v]
    2. Detect conflicts and create recoloring list
    for $v \in U$ **in parallel**
        for each neighbor w of v
            if color[w] = color[v]
                add higher-numbered vertex to R
    $U = R$
end proc

Credit: Gebremedhin & Pothen

Implementation Issues

Sandia National Laboratories

- Conflict resolution options:
  - Serial: When #conflicts is small, recolor these vertices in serial.
  - Parallel: Iterate and recolor until all conflicts resolved.
- How to find smallest available color:
  - Typically, a “forbidden” array is used but this requires a lot of memory
    - max-degree is often too pessimistic upper bound, especially for power-law graphs.
    - Limited fast memory on GPU.
    - Dynamic reallocation too expensive on Xeon Phi and GPU.
  - We chose to examine a fixed chunk of colors (64) at a time. May need multiple passes.

Kokkos Programming Environment

Sandia National Laboratories

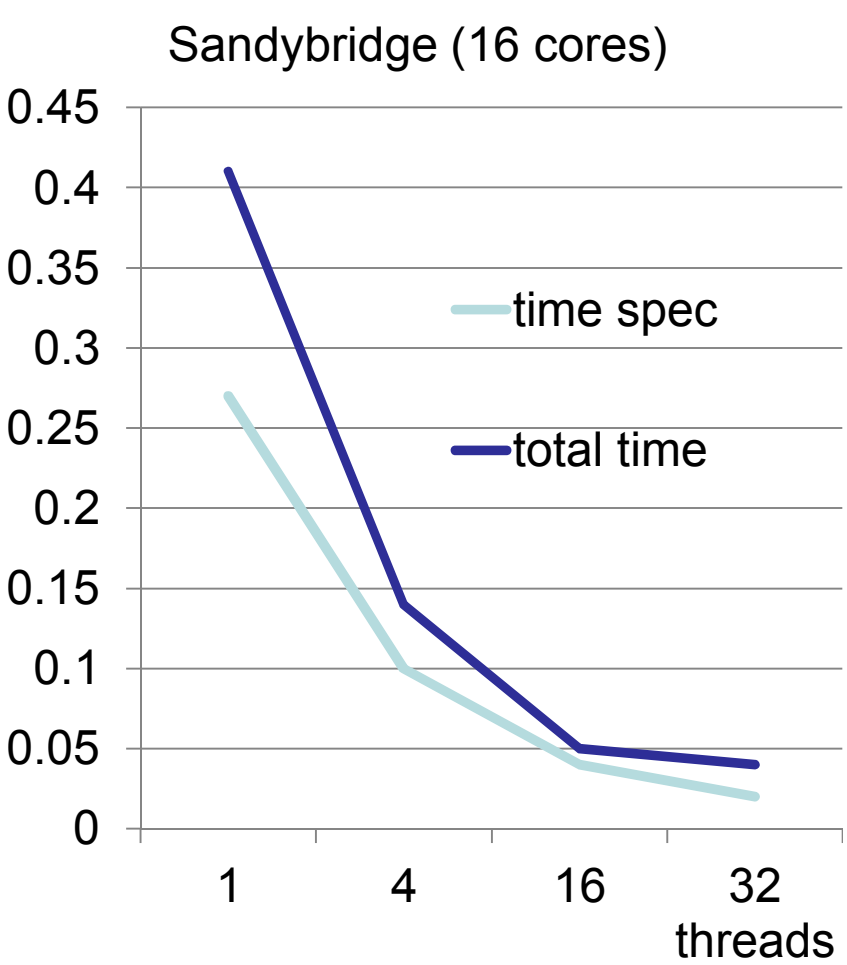
Challenge: How to write parallel code that is portable to CPU, MIC, GPU, and future architectures?

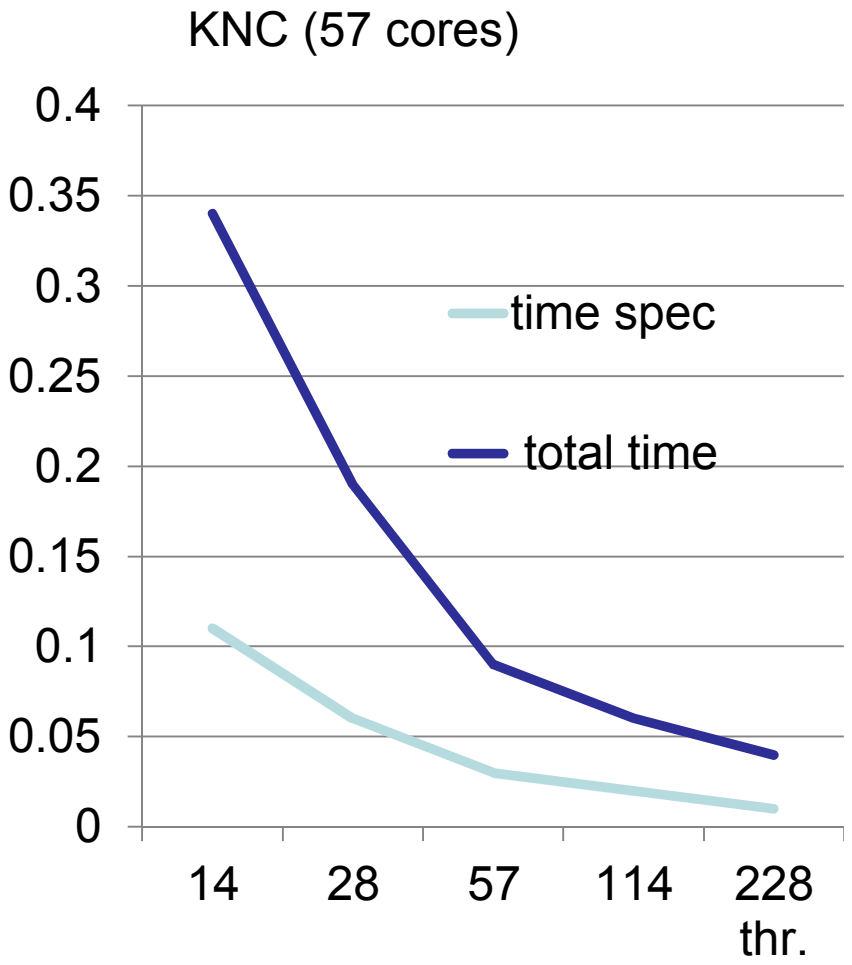
- We use Kokkos, a new library for performance-portable manycore programming

Key features:

- Write code once, many back-ends (e.g., OpenMP, CUDA)
- Portable C++11 library
- Data-parallel constructs well supported
- Helps with data placement ("first-touch")
- Tasks under development

Strong scaling (audikw1)





GPU Issues

Sandia National Laboratories

- Kokkos makes porting to GPU easy
  - Same source, no CUDA necessary
  - Need either UVM or explicit deep_copy from/to host/device
- Coloring algorithm:
  - #conflicts scales with #threads, so we may need many iterations to resolve all conflicts (serial resolution impractical)
  - Kernel launch time is high relative to coloring time
  - Have to assume data is already on the device

Conclusions

Sandia National Laboratories

- The speculative GM coloring method scales well on both multicore CPU and Intel Xeon Phi
- Observed #conflicts increases with concurrency (as expected)
- Quality (#colors) remains almost constant
- Useful paradigm for massively multithreaded graph algorithms
- GPU version in progress
- Kokkos provides portability and saves development time
- Plan open-source release
- Results are non-deterministic and non-repeatable. Get used to it!

Backup slides

Sandia National Laboratories

Serial Greedy Algorithm

Sandia National Laboratories

```
Procedure Greedy(G(V,E))
        Allocate  bool forbidden[]
        for each v in V do
                forbidden[*] = false
                for each w in adj(v) do
                        forbidden[color[w]] = true
                color[v] = min {i>0 | forbidden[i] == false}
End
```

Greedy coloring depends on the order vertices are visited. Good heuristics are Largest-First (LF) and Smallest-Last (SL). In parallel, we cannot impose any ordering (without limiting parallelism).

Results for CPU

Sandia National Laboratories

Platform: Intel Sandybridge, 16 cores
Graph: audikw1 (944K vertices, 39M edges)
Algorithm: GM with parallel conflict resolution

	1 thr.	4 thr.	16 thr.	32 thr.
Time speculative	0.27	0.10	0.04	0.02
Time conflict detection	0.13	0.05	0.02	0.02
Time conflict resolution	0	2e-5	3e-5	8e-5
Total time	0.41	0.14	0.05	0.04
#colors	54	60	60	63
#conflicts	0	3	27	42

Results for MIC

Sandia National Laboratories

Platform: Intel Xeon Phi (KNC, 57 cores)
Graph: audikw1 (944K vertices, 39M edges)
Algorithm: GM with parallel conflict resolution

	14 thr.	28 thr.	57 thr.	114 thr.	228 thr.
Time speculative	0.11	0.06	0.03	0.02	0.01
Time conflict detection	0.23	0.13	0.06	0.04	0.02
Total time	0.34	0.19	0.09	0.06	0.04
#colors	60	63	61	61	60
#conflicts	22	50	107	277	553
#iterations	1	1	1	2	2