PROBLEM BASED BENCHMARKS

Guy Blelloch, Julian Shun, Kanat Tangwongsan, Harsha Vardhan Simhadri (CMU), Phillip Gibbons (Intel Labs)

MOTIVATION

There are many different approaches to programming an algorithm or application to run in parallel:

- Transactions
- Nested parallelism
- Map reduce
- Data parallelism
- Thread pools
- Futures
- PGAS
- Message passing
- Bulk synchronization
- Wait-free data structures
- Race-free algorithms
- Commutative operations
- Amorphous data parallelism
- Tuple space
- Automatic parallelism
- …

What is the best approach? How does a programmer decide which approach to use? How can we benchmark parallel programming approaches?

BENCHMARK GOALS

A set of "problem based benchmarks":
Must satisfy a particular input-output interface, but there are no rules on the techniques used

Measure the quality of solutions based on:

- **Performance and speedup** over a variety of input types and w.r.t. best sequential implementations
- **Quality of output.** Some benchmarks don't have a right answer or are approximations
- **Complexity of code.** Lines of code & other measures
- **Determinism.** Returns the same output on same input
- **Generic.** Code should be generic over types
- **Correctness guarantees**
- **Easily analyze performance**, at least approximately
- **Robustness at massive scale**

SELECTING BENCHMARKS: CRITERIA

Initial Focus: Application kernels with

- Wide coverage for "real world" problems
- Reasonably simple solutions (< 500 lines of code)
- Can test correctness or measure quality of output
- Scalable problem sizes
- Relevant for a variety of system scales, from a multicore server to a cloud data center

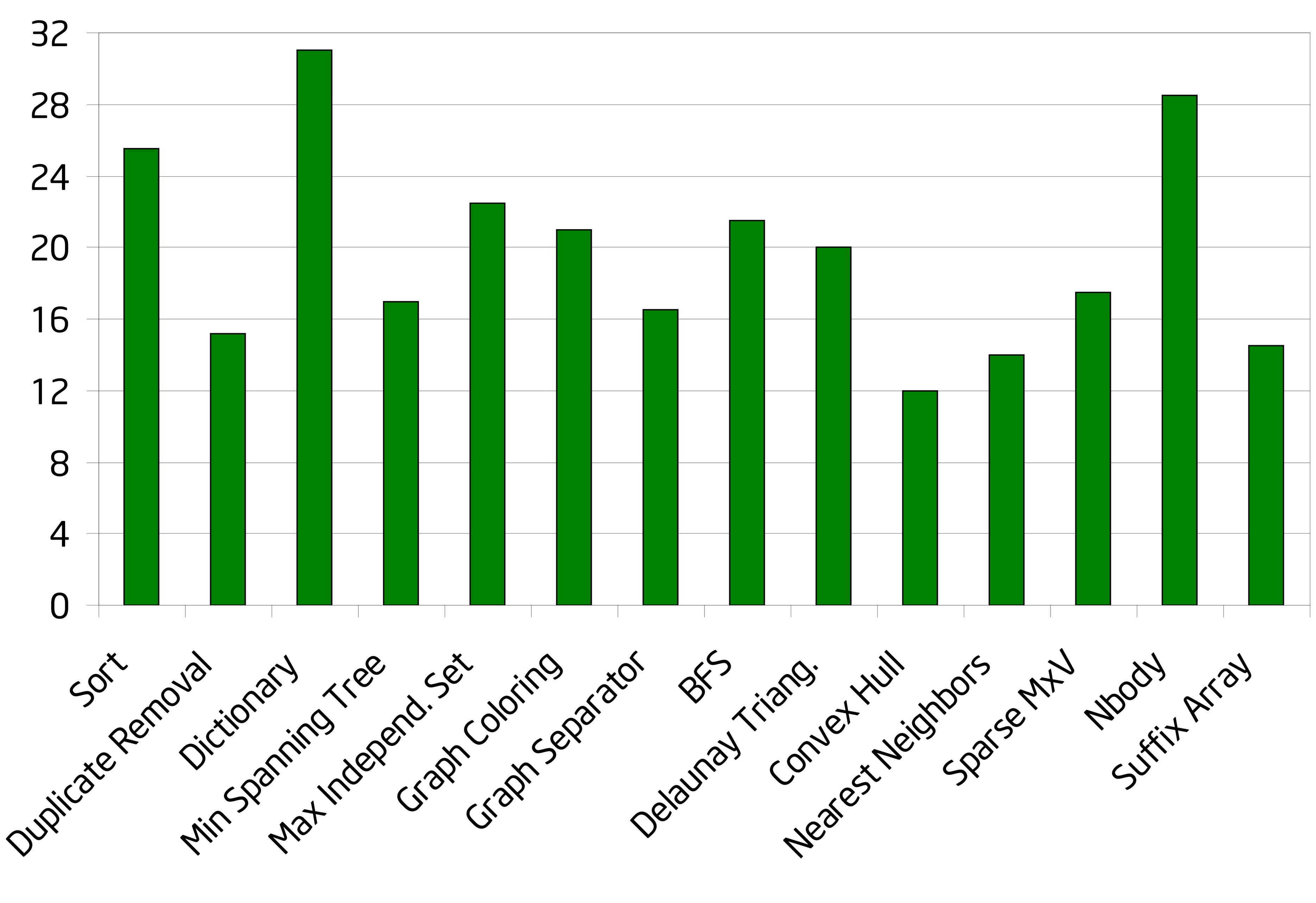
DOMAINS AND EXAMPLES

Sequences and strings: sorting, suffix arrays, seq. alignment
Graph algorithms: Min spanning tree, BFS, coloring, separators
Machine learning: Sparse SVM, K-means, Gibbs sampling, LASSO
Graphics: Ray Tracing, Micropoly Rendering
Geometry: Delaunay Triangulation, Nearest Neighbors, Nbody

INITIAL BENCHMARK RESULTS

Benchmark	LoC	Approach
Sort	230	Sampling, nested parallel
Duplicate Removal	122	Hashing
Dictionary	140	Deterministic hashing
Min Spanning Tree	162	Incremental, speculative
Max Independ. Set	63	Data parallel, random
Graph Coloring	45	Data parallel
Graph Separator	345	Nested parallel
BFS	45	Data parallel
Delaunay Triang.	325	Incremental, speculative
Convex Hull	93	Nested parallel
Nearest Neighbors	106	Nested parallel
Sparse MxV	22	Data parallel
Nbody	170	Nested parallel
Suffix Array	138	Data parallel

SPEEDUPS (32 CORE NEHALEM)



"Internally Deterministic Parallel Algorithms can Be Fast"
To appear in PPoPP'12

Intel Science & Technology Center for Cloud Computing

Carnegie Mellon University · Georgia Tech · intel · Princeton University · UC Berkeley