Algorithmic Paradigms

New algorithmic paradigms that promote reasoning about parallel performance and lead to provable performance guarantees, while allowing algorithms to be mapped onto diverse parallel and distributed environments, and optimizing resource usage including compute cycles, communication, input-output (I/O), memory hierarchies, and energy;

Challenges

1) New paradigms
   - Hadoop/MR, HBP, BSP, etc. work well in some cases.
   - What about for more inherently irregular settings like graph algorithms?
   - Beat MapReduce in practice.

Challenges

2) Better fundamental algorithms
   - e.g., single source shortest paths.

Challenges

3) Coping with big data

- Sequential doesn’t even work – goal is to solve the problem at all, not beat sequential

Challenges

4) Getting data in and out of the cloud

- Currently very unpredictable
- How to make performance guarantees?
- Algorithm must specify I/O demands as a resource requirement?

Challenges

5) Parameter-oblivious algorithms

- How? What do they look like?
- What are the limits?

Challenges

6) Energy tunable algorithms
- If there is even any reason to tune.
- How can you specify such an algorithm?
- Parameter oblivious?

Challenges

7) Data structures

- The modularity is great for sequential programs.
- Want structures more interesting than arrays
- What does a program have to look like to get good performance when data structures are there? How do we analyze it?

Challenges

8) Performance models are necessary

- HTMs & synchronization
  - What other primitives should we work with?
- GPUs
- etc.

Challenges

9) System software optimization

- Scheduling
- Data placement
- Resource allocation.

Challenges

10) Scaling from small to huge with few knobs

- What should an algorithm description look like?

Challenges

15% of papers in STOC/FOCS/SODA about parallelism