Fast Parallel Linear Programming

Dylan Foster (djf244)

October 7, 2015

Linear programming is a fundamental tool in fields including complexity theory and algorithm design, optimization, and operations research. A linear program is a specific type of optimization problem in which one is asked to optimize a linear objective function subject to linear constraints. A typical linear program might look like the following:

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b, \end{aligned}$$

where $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$. Geometrically, we can think of the linear program as asking us to search inside a polytope (a higher-dimensional polyhedron) defined by A and b for a point x that maximizes the function $c^T x$. Computer scientists have been solving linear programs with the exponential-time *simplex method* since World War II, but in 1979 Leonid Khachiyan exhibited the first truly polynomial time linear programming algorithm. Khachiyan's *ellipsoid algorithm* has a better theoretical guarantee than the simplex method but good theoretical guarantees don't always translate to good practical performance; Practitioners continued using the simplex method because it ran far faster on real data.

Since Khachiyan much effort has gone into efficiently solving linear programs efficiently. [1] initiated research into parallel linear programming algorithms with strong theoretical guarantees. They showed that if one is willing to accept an algorithm whose performance scales poorly with the precision to which the LP is solved, one can excellent runtime with respect to the size of the input data. Furthermore, their algorithm is amenable to parallelization. Parallel LP solvers with improved theoretical guarantees have since been obtained.

We propose an implementation of the theoretical state of the art [2]. It remains to be seen if this algorithm will have good practical performance or if, like the ellipsoid method, it is primarily of interest to theoreticians. [2] ran preliminary experiments which suggest that their approach performs favorably compared to other parallel solvers in terms of number of iterations. We hope to learn the following:

1. How well can the algorithm perform in terms of more concrete performance measures, such as absolute runtime and GFLOP/s?

2. How does the algorithm perform compared to the best serial LP solvers?

3. Is the algorithm amenable to the optimization techniques we have discussed in class? For example, can we easily tweak the algorithm to achieve good cache performance or is poor stride somehow inherent in this approach?

References

[1] Luby, Michael, and Noam Nisan. "A parallel approximation algorithm for positive linear programming." Proceedings of the twenty-fifth annual ACM Symposium on Theory of Computing. ACM, 1993.

[2] `http://arxiv.org/abs/1407.1925`