Beware

By

J Crowcroft from work by Malte Schwarzkopf & Frank McSherry

Computer Laboratory & Unaffiliated

University of Cambridge
tl;dr #1

• Network speed may not matter with a Spark based stack, but it does matter to higher performance analytics stacks, and for graph processing especially.

• By moving from a 1G to a 10G network, we see a 2x-3x improvement in performance for timely dataflow.
tl;dr #2

• A well balanced distributed system offers performance improvements even for graph processing problems that fit into a single machine;

• running things locally isn't always the best strategy
tl;dr #3

- PageRank performance on GraphX is primarily system bound. We see a 4x-16x performance increase when using timely dataflow on the same hardware, which suggests that GraphX (and other graph processing systems) leave an alarming amount of performance on the table.
fn pagerank(graph: &G, vertices: usize, alpha: f32) {
    // mutable per-vertex state
    let mut src = vec![0f32; vertices];
    let mut dst = vec![0f32; vertices];
    let mut deg = vec![0f32; vertices];
    // determine vertex degrees
    for (x, _) in graph.edges() { deg[x] += 1f32; }
    // perform 20 iterations
    for _iteration in (0 .. 20) {
        // prepare src ranks
        for vertex in (0 .. vertices) {
            src[vertex] = alpha * dst[vertex] / deg[vertex];
            dst[vertex] = 1f32 - alpha;
        }
        // do the expensive part
        for (x, y) in graph.edges() { dst[y] += src[x]; }
    }
}
Impl #1: Send everything
Impl #2: Worker-level aggregation
Impl #3: Process-level aggregation
Some Baseline figures

![Runtime comparison chart]

- Yellow: Spark, as published
- Aqua: GraphX, as published
- Blue: GraphX, our cluster
- Red: Laptop, 1 thread (simple)
- Purple: Laptop, 1 thread (smart)

Twenty pagerank iterations, baseline measurements.
## System

<table>
<thead>
<tr>
<th>System</th>
<th>source</th>
<th>cores</th>
<th>twitter</th>
<th>rv</th>
<th>uk</th>
<th>2007</th>
<th>05</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spark</td>
<td>GraphX paper (<a href="https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-gonzalez.pdf">link</a>)</td>
<td>16x8</td>
<td>857s</td>
<td>1759s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphX</td>
<td>GraphX paper (<a href="https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-gonzalez.pdf">link</a>)</td>
<td>16x8</td>
<td>419s</td>
<td>462s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphX</td>
<td>measured on our cluster</td>
<td>16x8</td>
<td>334s</td>
<td>362s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single thread</td>
<td>COST paper (<a href="https://www.usenix.org/conference/hotos15/workshop-program/presentation/mcsherry">link</a>)</td>
<td>1</td>
<td>300s</td>
<td>651s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(simpler)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single thread</td>
<td>COST paper (<a href="https://www.usenix.org/conference/hotos15/workshop-program/presentation/mcsherry">link</a>)</td>
<td>1</td>
<td>110s</td>
<td>256s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(smarter)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Timely dataflow impl

### Twenty pagerank iterations on one machine, multiple threads.

<table>
<thead>
<tr>
<th>System</th>
<th>cores</th>
<th>twitter rv</th>
<th>uk_2007_05</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timely dataflow</td>
<td>1</td>
<td>350.7s (11.33s)</td>
<td>442.2s (8.90s)</td>
</tr>
<tr>
<td>Timely dataflow</td>
<td>2</td>
<td>196.5s (6.39s)</td>
<td>297.3s (5.67s)</td>
</tr>
<tr>
<td>Timely dataflow</td>
<td>4</td>
<td>182.4s (6.12s)</td>
<td>192.0s (3.78s)</td>
</tr>
<tr>
<td>Timely dataflow</td>
<td>8</td>
<td>107.6s (3.70s)</td>
<td>137.1s (3.29s)</td>
</tr>
<tr>
<td>Timely dataflow</td>
<td>12</td>
<td>95.0s (3.32s)</td>
<td>114.5s (2.65s)</td>
</tr>
</tbody>
</table>
Now you can have multiple ...
Conclusions 1

• As we have seen, the three implementations (GraphX and the two timely dataflow ones) have different bottleneck resources.
• GraphX does more compute and is CPU-bound even on the 1G network, whereas the leaner timely dataflow implementations become CPU-bound only on the 10G network.
• Drawing conclusions about the scalability or limitations of either system based on the performance of the other is likely misguided.
Conclusions 2

• Fast 10G networks *do* help reduce the runtime of parallel computations by significantly more than 2-10%: we've seen speedups up to 3x going from 1G to 10G.

• However, the structure of the computation and the implementation of the data processing system must be suited to fast networks, and different strategies are appropriate for 1G and 10G networks.

• For the latter, being less clever and communicating sometimes actually helps.
Conclusions 3

• Distributed data processing makes sense even for graph computations where the graph fits into one machine.

• When computation and communication are overlapped sufficiently, using multiple machines yields speedups up to 5x (e.g., on twitter_rv, 1x8 vs. 16x8). Running everything locally isn't necessarily faster.
Conclusions 4

• Can make PageRank run **16x faster per iteration** using **distributed timely dataflow** than using GraphX (from 12.2s to 0.75s per iteration).

• This tells us something about how much scope for improvement there is even over numbers currently considered state-of-the-art in research!
For more details&followup

- See

- Ack to Malte Schwarzkopf & Frank McSherry