Zach Pomper, Max Johnson
15-418

Checkpoint Report

**WORK COMPLETED:** We have finished and corrected sequential implementations of our target algorithms, and have implemented our desired supporting scripts. Among these scripts are a visualizer, I/O parsing scripts, a validator, a random input generator, and build scripts. We wrote sequential BFD and WalkPack bin packers in C++, which will hopefully be a good precursor to implementing them in CUDA. While we did not mention BFD in our initial project proposal, we realized that it would be an important comparison measure against the RIT paper we’ve been referencing. We have a preliminary CUDA implementation of WalkPack as well. We have also spent a decent amount of time playing with tuning parameters governing the behavior of WalkPack in an attempt to improve the quality of solutions it generates: We have made progress in this arena but it’s not fully there just yet.

**DELIVERABLE PROGRESS:** We feel we’re on track to complete our deliverables, but suspect that the goalposts will have to be moved in terms of the problems addressed by our solution. It appears that the randomized heuristics implemented in our reference paper are even less effective than we had thought, so our goal is now to show that a randomized algorithm can achieve acceptable quality while being more parallelizable than traditional solutions (we had originally planned for our implementation to be both more optimal and more parallel-friendly). We are still hopeful that we will be able to double up on parallelization with a hybrid MPI-CUDA solution (we have accounts on componium, a dual-GP102 computer club machine in Cyert Hall).

**POSTER SESSION:** Our poster plans have not changed since our proposal. We are planning on showing the usual assortment of speedup graphs, as well as a few visual comparisons from our visualizer output.

**SCHEDULE:**

<table>
<thead>
<tr>
<th>Week</th>
<th>Dates</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10/29–11/04</td>
<td>Write project proposal. Create framework and test input generator.</td>
</tr>
<tr>
<td>1</td>
<td>11/05–11/11</td>
<td>Write sequential implementation of Best-Fit Decreasing. Start sequential implementation of WalkPack. Create visualizer and output analyzer.</td>
</tr>
<tr>
<td>2</td>
<td>11/12–11/18</td>
<td>Refine sequential implementation of WalkPack. Start parallel</td>
</tr>
</tbody>
</table>
We have very successfully worked together to complete the tasks so far, and will continue to work on each task as a team rather than individually.

**CONCERNS:** For the most part, we know what needs to be done and have the tools to do it. We were somewhat disappointed to learn that CUDA vector methods are only executable in host code, but this is not an insurmountable setback. We’ve had more trouble than expected making the sequential WalkPack generate solutions of sufficient quality. We may have to accept lower quality than expected to focus our time on parallelizing the algorithm. Otherwise, we’re feeling good about the project.

**PRELIMINARY RESULTS:**

**UNIFORM CASE:**
This test case consists of 1000 objects with sizes selected uniformly at random from [1, 10000] and bins of size 10000. The average object size is 5000, half of the bin size. Based on this, we'd expect this case to be solvable with approximately 500 bins. BFD roughly matches that with 508 bins used. By visual inspection, BFD produces a high-quality solution. This is consistent with the results of [1], which found that BFD performs well on uniformly random cases. Our implementation of WalkPack produces a lower-quality solution than BFD at 548 bins, which is expected based on the results of [1]. Visually inspecting the output of WalkPack shows erratic placement, which is expected due to the random nature of the algorithm. We observe some clearly poorly-chosen bins including bins with a single small object. This is simply a virtue of the algorithm; WalkPack will sometimes make choices that increase the bin count in ways that
seem "obviously" bad. We may investigate heuristics to clean these sparsely-filled bins during a final pass.

Figure 1 (left). WalkPack Results for the uniform case. 548 bins used.
Figure 2 (right). BFD Results for the uniform case. 508 bins used.

TWO-THIRDS CASE:
This test case consists of 1000 objects with sizes selected uniformly at random from [2000, 3000] and bins of size 10000. It is called the "2/3 case" in [1], though is might more accurately be called the "3/4 case". This is considered an adversarial case for BFD, as the algorithm will fill the first bins with the largest objects, which will result in many bins with only 3 objects, though we'd expect an optimal solution to use many bins with 4 objects based on the average object size. As expected, BFD produces many bins with only three objects. Our implementation of WalkPack still performs worse than BFD, but by a much smaller margin. WalkPack involves several heuristics with multiple parameters, which we have yet to tune to reach an acceptable quality. We are expecting WalkPack to produce results of significantly better quality than BFD in the 2/3 case. For our preliminary results we see a marked improvement in the relative quality between WalkPack and BFD while going from the uniform case to the 2/3 case.
Figure 3 (left). WalkPack Results for the 2/3 case. 286 bins used.
Figure 4 (right). BFD Results for the 2/3 case. 279 bins used.