Zach Pomper, Max Johnson
15-418

Final Project Proposal


URL: http://www.contrib.andrew.cmu.edu/~msjohnso/418/

SUMMARY: We will implement and parallelize at least two heuristic algorithms in order to test the real-world utility of bin packing algorithms in nominally different complexity classes. We will start in a C++ and CUDA framework and attempt to extend this to multiple nodes using MPI if time permits.

BACKGROUND: Bin packing is a known NP-hard problem with applications in task scheduling, FPGA design, logistics, and virtual machine allocation. There exists an approximation algorithm with a proven upper bound of $1 + 11/9 \cdot$ optimal bins, though many randomized algorithms have averages much closer to optimal.

Briefly put, the bin packing problem is this: given a set of objects of various sizes and an unlimited supply of uniformly-sized bins, what is the assignment of objects to bins that minimizes the number of bins required to hold all objects? One approximation algorithm, Best-Fit Decreasing (BFD), is proven to find a solution that uses at most $1 + 11/9 \cdot$ optimal bins. This algorithm sorts the objects in decreasing size, then places each object in the first bin into which it fits.

One of the algorithms we will be implementing is WalkPack[1]. This algorithm employs randomization to improve on the quality of BFD. The basic sequential pseudocode is as follows:

Repeat for number of iterations
    Pick distinct random bins s,d
    Move all objects in d to s
    Destroy the now-empty d
    If size of objects in s > bin capacity
        Create a new bin b
        Move all objects which overflow s to b

To achieve reasonable quality and faster convergence, heuristics must be added to the selection of bins s and d. These are described in the following section. In [1], the authors parallelize this
algorithm by running it across the threads of the GPU and periodically choosing the best solution (fewest bins). All threads continue executing this algorithm on the new best solution. We plan on investigating other ways to parallelize this algorithm, such as assigning threads to bins or ranges of bins.

**THE CHALLENGE:** Bin packing is a challenging problem because the search space for optimal solutions explodes quickly with increasing numbers of bins, with near-optimal solutions only ever converging slowly. The parallelizability of the problem largely depends on the choice of algorithm used to approximate searching the whole space. We will start by implementing parallel WalkPack, which while simple to parallelize, exposes heuristics to reduce overall runtime and improve quality that will make parallelization more difficult. For example, between iterations, clearly sub-optimally filled bins, like bins with only one item, can be merged. Both the initial WalkPack algorithm and this kind of optimization require frequent changes in shared state, which is often difficult to implement efficiently on a GPU. Moreover, the random nature of WalkPack likely suffers from poor locality, as threads will reference many disparate bins over their lifetimes. Furthermore, simple mappings of threads to bins will fail to scale as optimal solutions rapidly cut away bins, potentially leading to divergent execution. In terms of constraints, GPU parallelization often makes dynamic scheduling and load balancing difficult, both of which are inherent problems in WalkPack.

**RESOURCES:** We will probably be using cluster computers on latedays and/or ghc, as well as computer club machines which we already have access to. We found a number of algorithmic descriptions that we can implement at our leisure. That said, we’ll be writing all of our code from scratch. The RIT paper is our primary reference on the WalkPack algorithm, with the other sources providing references for different approaches, should we choose to implement them.

Papers:
[1] [https://www.cs.rit.edu/~ark/students/amb4757/report.pdf](https://www.cs.rit.edu/~ark/students/amb4757/report.pdf)

Supplemental reading on more algorithms:
[https://link.springer.com/chapter/10.1007/978-3-319-47217-1_6](https://link.springer.com/chapter/10.1007/978-3-319-47217-1_6)
**GOALS AND DELIVERABLES:** We plan to implement and parallelize two algorithms for solving the bin packing problem. We will first implement WallPack in parallel. Once we have made significant progress in that implementation, we will decide on a second algorithm to parallelize. This decision will be informed by our experience with the first algorithm and will be based on the time remaining in the project. We have rough time and quality metrics from the paper describing the implementation of WallPack for GPU ([1]), which we would like to improve upon (hopefully significantly, given that we’re using low-level frameworks that will let us better optimize for our system configuration).

We will demonstrate speedup graphs and quality statistics for our implementations. These can be compared with the similar graphs from [1] to establish the quality of our work. We would like to create a visualizer that will give some intuition about the progress of the algorithm and the ultimate form of the solution.

**PLATFORM CHOICE:** We anticipate our workload will have a large amount of synchronization overhead and potentially poor locality. Since we would like our implementation to scale well with increasing numbers of bins (scaling memory demand), a GPU seems like a good parallel system to use. Accordingly, our familiarity with CUDA makes it the obvious choice of framework there. Additionally, implementations using multiple MPI nodes could reveal the usefulness of greedy algorithms. In terms of which GPU(s) in particular we will be using, we suspect that memory bandwidth and latency will be constraining factors in our implementation, but we have some flexibility in which GPU(s) we could use. Moreover, CUDA allows for a reasonable amount of flexibility in porting code to other systems, so we should be able to readily test the effects of different system configurations on performance of our algorithm(s).

**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. Start sequential implementation of WallPack.</td>
</tr>
<tr>
<td>2</td>
<td>11/12–11/18</td>
<td>Finish parallel implementation. Submit checkpoint.</td>
</tr>
<tr>
<td>Week</td>
<td>Date Range</td>
<td>Task Description</td>
</tr>
<tr>
<td>------</td>
<td>--------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>4</td>
<td>11/26–12/02</td>
<td>Write and optimize parallel implementation of second algorithm.</td>
</tr>
<tr>
<td>5</td>
<td>12/03–12/09</td>
<td>Attempt MPI implementations</td>
</tr>
<tr>
<td>6</td>
<td>12/10–12/15</td>
<td>Write final report, finishing touches, polish visualizer, submit final report.</td>
</tr>
</tbody>
</table>