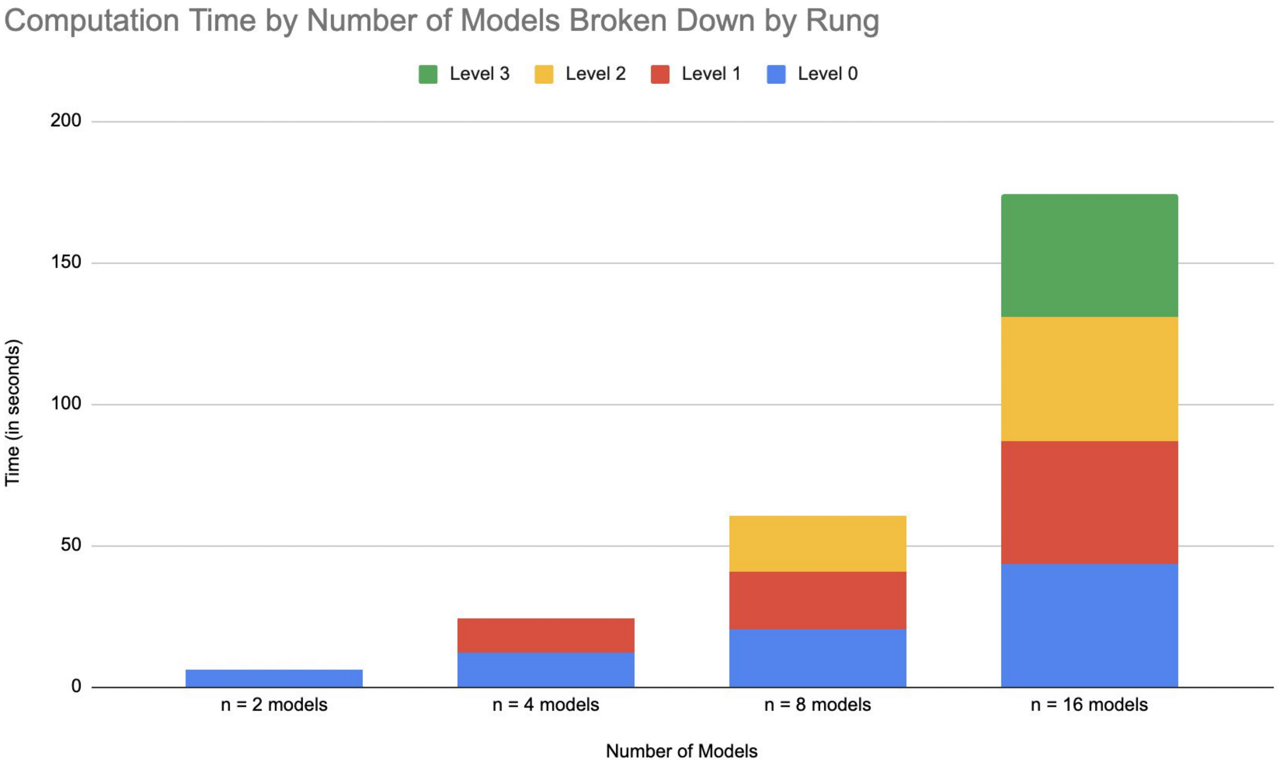
M4: ASH Parallel Design

Evan and Roy

ASH Overview + Sequential Methodology

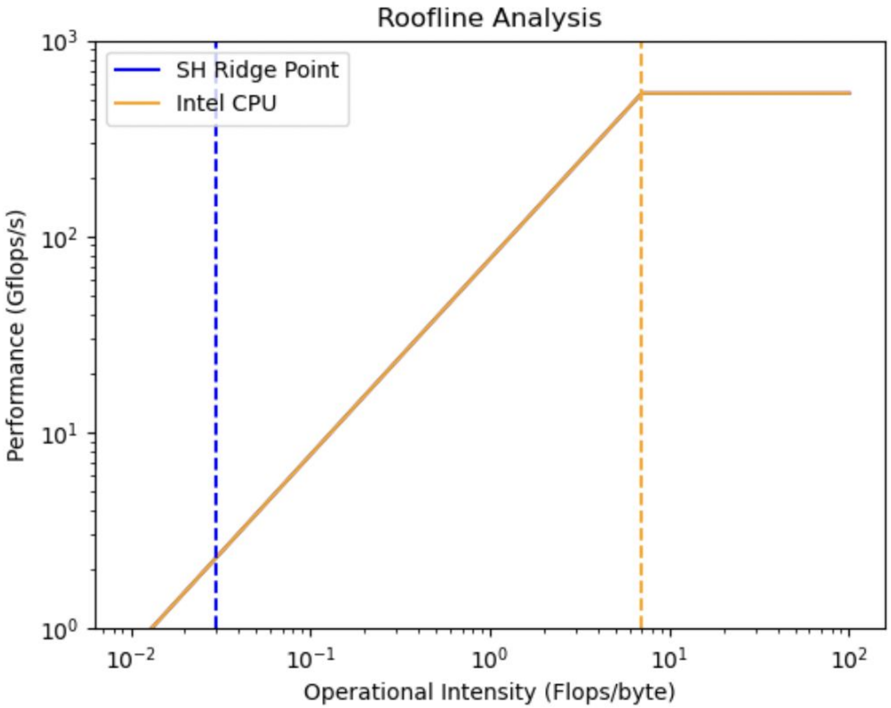
- ***Problem***: hyperparameters k2 **model performance** – however, optimizing has become increasingly challenging b/c **more complex datasets** + **high-dimensional search spaces**
- ***Solution***: Asynchronous Sequential Halving (ASH) is an algorithm for **efficiently tuning ML hyperparameters** by combing over the hyperparameter search space with **multiple cores simultaneously**
    - Random search > grid search b/c less exhaustive/expensive to get similar results **(~95%)**
- ***Sequential Halving (SH)****:* sequentially iterate over all models on the base rung and promote the top half to the next rung, then double the epochs and iterate over all models in next rung and promote, etc.

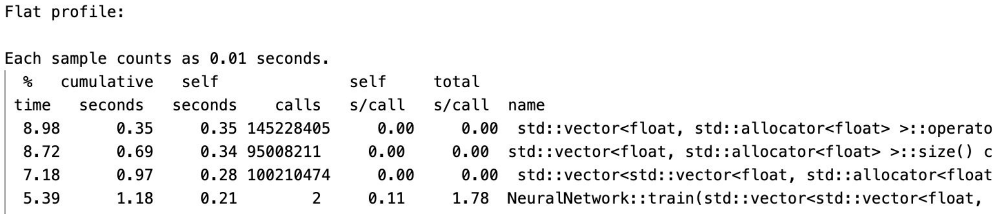
Sequential Baseline



Roofline Analysis

- **Nominal Peak Performance** of Intel CPU: 537.6 Gflop/s
- **Beta**: 76.8 GB/s
- **Optimal Ridge Point**: 7 Flops/byte
- The operational intensity of our bottleneck: 7500 flops/250000 bytes = 0.03 flops/byte
- **Performance**: 76.8 * 0.03 = 2.304 Gflop/s
- Gprof indicates **intensive memory management** and **movement**
- Supports hypothesis that increasing processes will be effective in **increasing operational intensity**

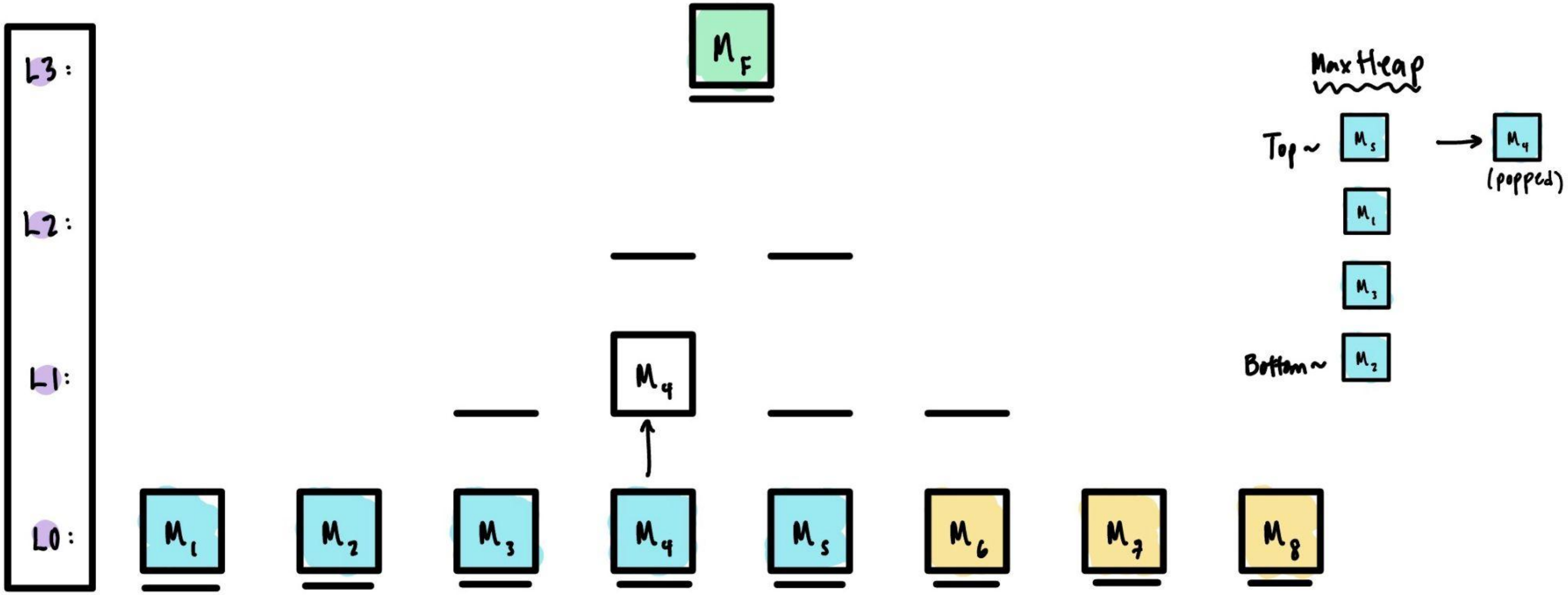




```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  s/call   s/call  name
  8.98      0.35     0.35 145228405     0.00     0.00  std::vector<float, std::allocator<float> >::operato
  8.72      0.69     0.34 95008211     0.00     0.00  std::vector<float, std::allocator<float> >::size() c
  7.18      0.97     0.28 100210474     0.00     0.00  std::vector<std::vector<float, std::allocator<float
  5.39      1.18     0.21        2     0.11     1.78  NeuralNetwork::train(std::vector<std::vector<float,
```

Parallel Design



Driver Code: MPI to 1) send model hyperparameter configurations to cores
2) communicate individual model results