Auto-reconfiguration for Latency Minimization in CPU-based DNN Serving

Ankit Bhardwaj
MIT CSAIL

Amar Phanishayee
Meta

Deepak Narayanan
NVIDIA

Ryan Stutsman
University of Utah

Motivation

Serving small DNN models on CPU:

- Consumes lower power
- Cost-effective
- Easy to scale across multiple machines

Increasing core counts and memory bandwidth make them suitable for serving small models.

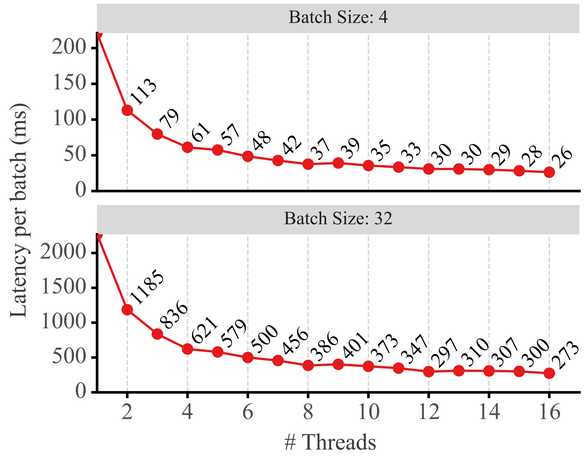
Identifying Configuration is Challenging

<1, T, B> to <*i, t, b*> is challenging

- Model specific parallelism
- Batch-specific gains
- Heterogeneous hardware

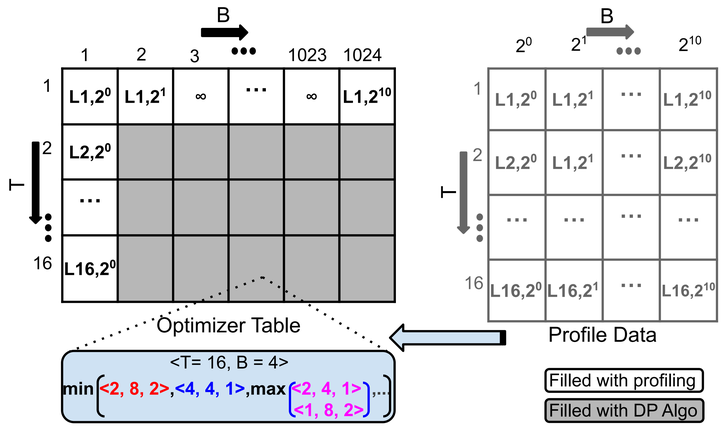
Identifying a configuration that works for a specific model and a given batch size on a machine can be challenging.

Reducing Latency with Intra-op Parallelism



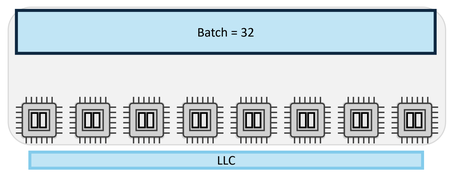
Diminishing returns on adding more cores

Dynamic Programming to the Rescue

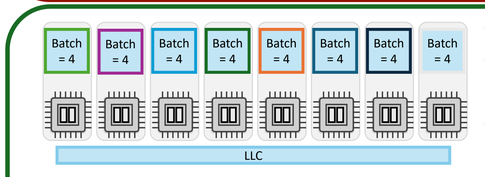


- Profile a few configurations for each model
- Predict the rest using a 2D-Knapsack solution

Multi-instance Serving for Lower Latency



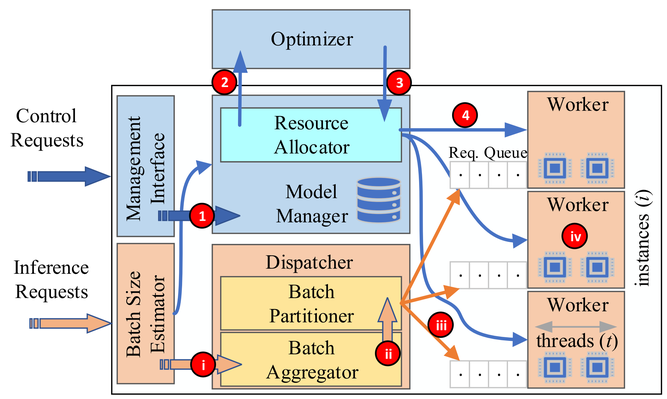
- Batch Size (B) = 32
- Cores (T) = 16
- Single Instance Latency = 273 ms



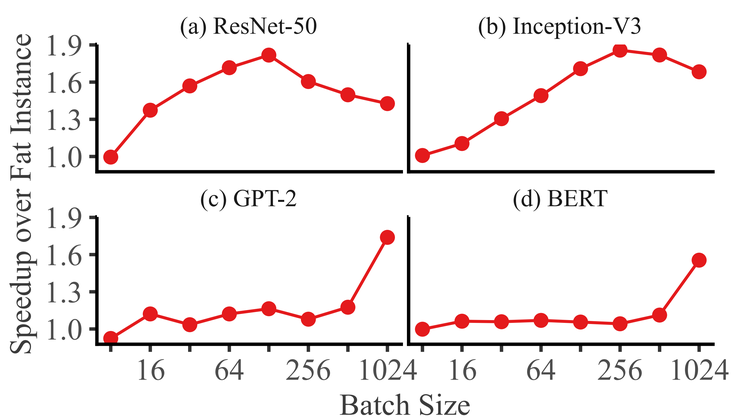
- Batch Size (B) = 32
- Cores (T) = 16
- 8 Instance (4b, 2t) Latency = 113 ms

Same Batch, Same Cores, but Multiple Instances -> 2.4x Speedup

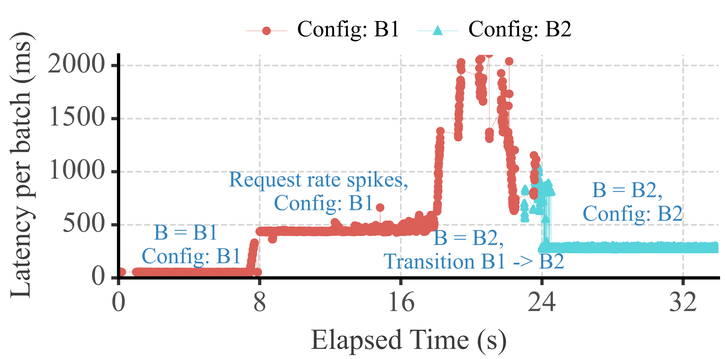
Packrat Architecture



Multi-instance Speedups



Automated Reconfiguration



Conclusion

- Pure intra-op parallelism on CPUs shows diminishing returns.
- Multi-instance execution improves performance.
- Packrat automatically finds optimal configurations and dynamically adjusts the serving system.
- Achieves up to **3.2x** latency speedup.