Accelerating Large Scale Deep Learning Inference through DeepCPU at Microsoft

2019 USENIX Conference on Operational Machine Learning

Minjia Zhang, Samyam Rajbandari, Wenhan Wang, Elton Zheng, Olatunji Ruwase, Jeff Rasley, Jason Li, Junhua Wang, Yuxiong He

Microsoft AI and Research

Highlights

- **DeepCPU**, the fastest deep learning serving library for recurrent neural networks (RNNs) on CPUs
- SLT (Scenario, Library, Technique) driven methodology
- 10x lower latency and cost than existing framework
- Ship DL models with great latency/cost reduction in Microsoft

Deep Learning Serving Challenges

- Long serving latency blocks deployment
- Support advance models while meeting latency SLA and saving cost

DL Scenarios	Original Latency	Latency Target
MRC Model A	~100ms	< 10ms
MRC Model B	~107ms	< 10ms
Ranking Model	10ms for [query, 1 passage] x 150 passages	< 5ms
Query rewriting	~51ms	< 5ms

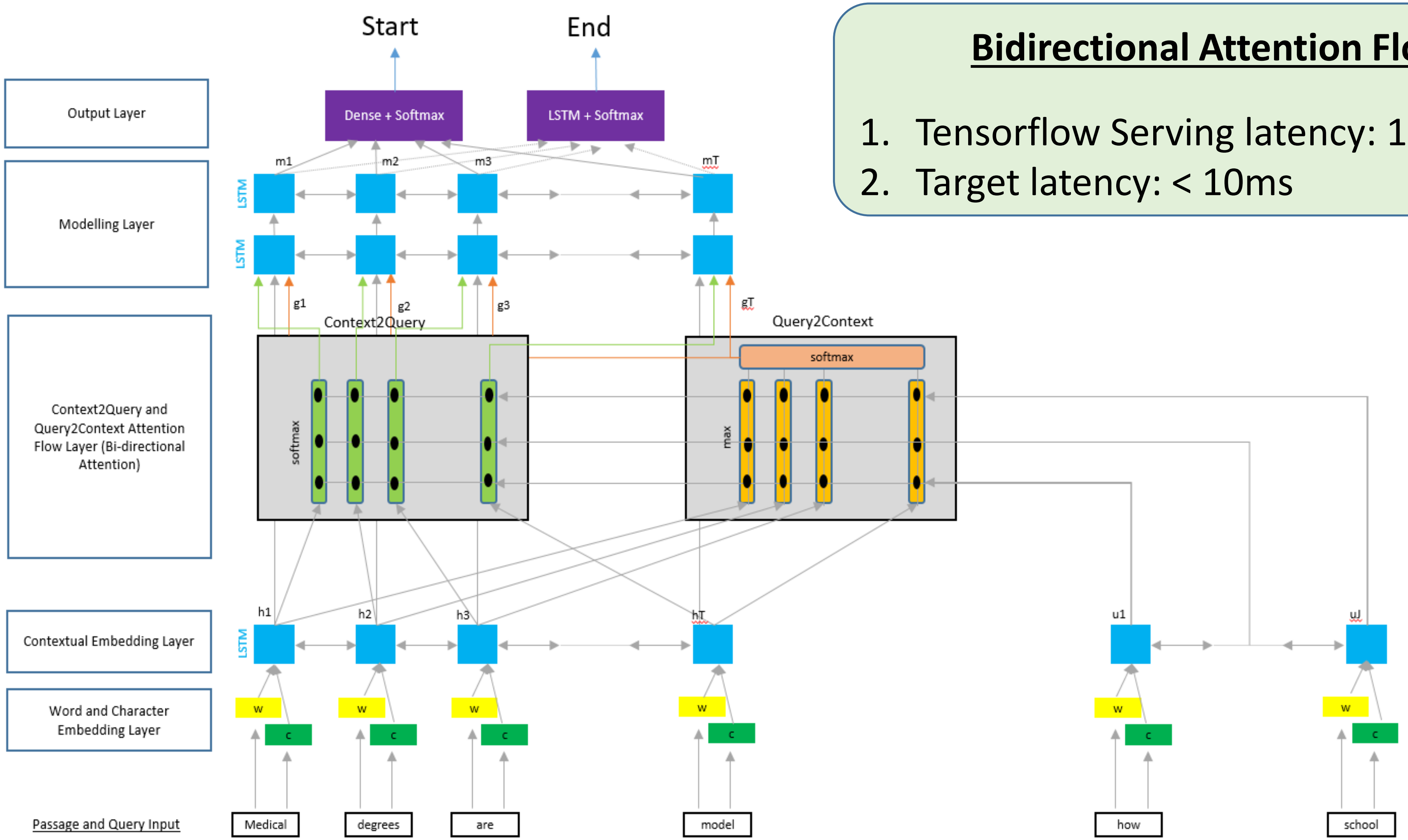
Methodology

- **Improve existing general-purpose DL frameworks?**
- **Customized optimization with effective reuse**
- **Co-development of Scenario, Library, and Technique (SLT)**
- Scenario
  - Apply customized optimization, striking for best performance
  - Think out of box, not limited by existing framework
- Library
  - Collection of generic building blocks that speed up customized optimization
  - Framework independent -- can benefit multiple DL frameworks
- Technique
  - One technique could benefit multiple library components and many scenarios
  - Parallelism, scheduling, and locality optimization on CPU at no cost in accuracy

Outline

- Real-World Scenarios with DeepCPU-Powered RNN-Based Models
- Library Features
- Optimization Techniques
- How is DeepCPU Utilized?

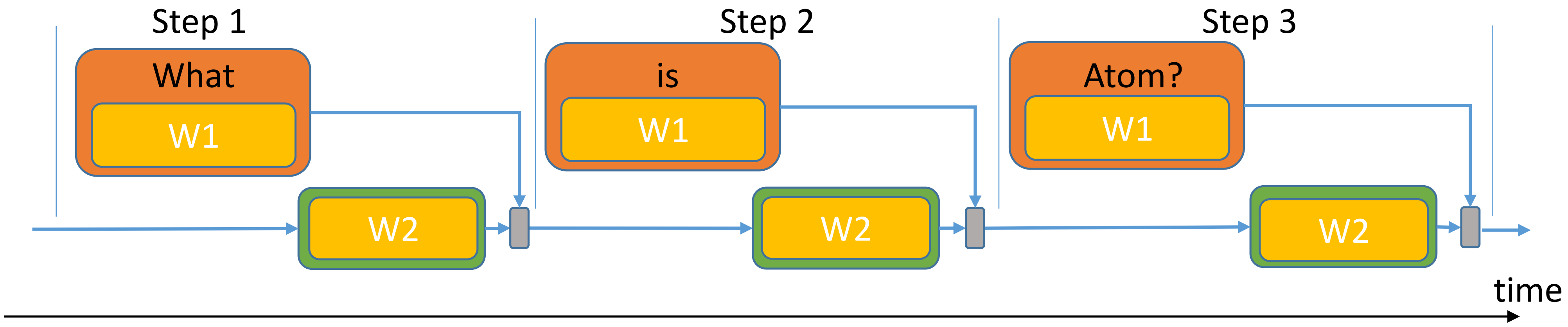
Scenario 1: Question Answering



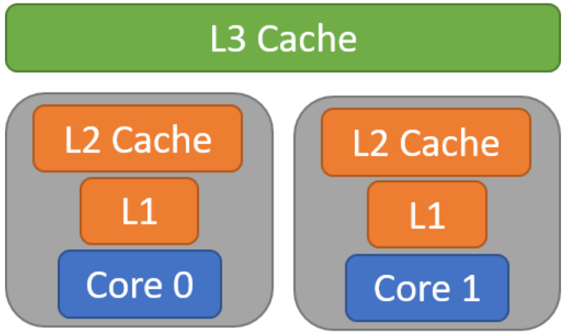
Bidirectional Attention Flow Model (BiDAF)

1. Tensorflow Serving latency: 107ms **(non-shippable)**
2. Target latency: < 10ms

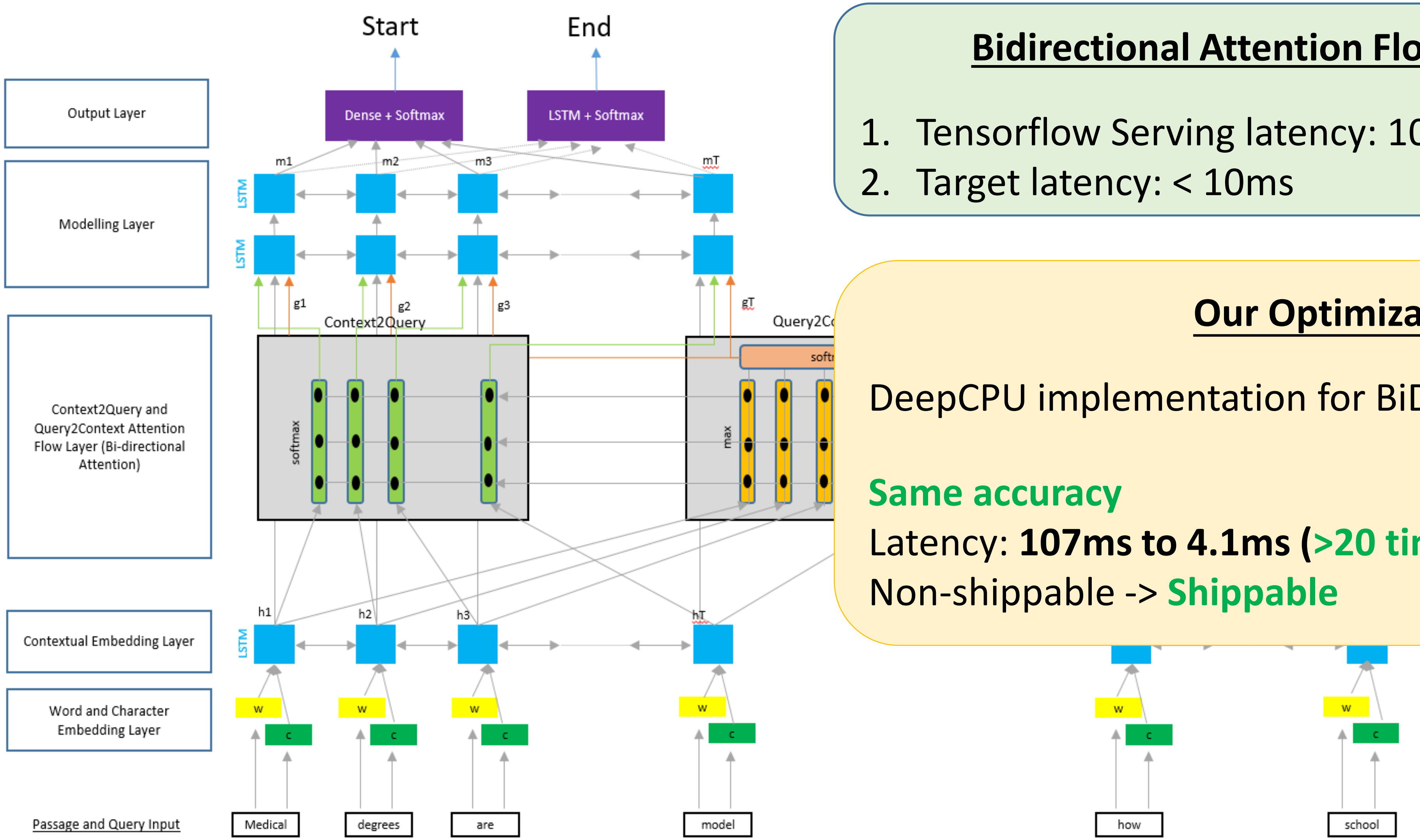
RNN Performance Bottleneck



Performance Critical Factors	Implications
Limited Parallelism **(small batch size)**	Poor Scalability
Poor Data Locality	Poor Scalability and Performance due to reading data from slow memory



Optimization Results



Bidirectional Attention Flow Model (BiDAF)

1. Tensorflow Serving latency: 107ms **(non-shippable)**
2. Target latency: < 10ms

Our Optimization

DeepCPU implementation for BiDAF

Same accuracy

Latency: **107ms to 4.1ms (>20 times speedup)**

Non-shippable -> **Shippable**

Scenario 2: Text Similarity Ranking

- Generate text similarities using deep learning model
- Model: word embedding + encoding with GRUs + conv + max-pool
- Latency SLA: 5ms for <query, top 150 passages>
- Tensorflow serving latency — non-shippable
  - single <query, passage> pair: 10ms
  - <query, 150 passages>: fan-out to 150 machines

- Our optimizations — shippable, save machines
  - <query, 150 passages>: 5ms, one machine (>100x throughput gain)
  - Reduce thousands of machines and millions of infrastructure costs

Optimization Results

	Scenarios	Original Latency	Latency Target	Optimized Latency	Latency reduction	Throughput improvement
1	MRC Model A	~100ms	10ms	9ms	>10X	> 10X
2	MRC Model B	~107ms	10ms	4.1ms	>20X	> 50X
3	Neural Ranking Model A	10~12ms for [query, 1 doc] x 33 docs	6ms	1.5ms for [query, 1 doc]; <6ms for [query, 33 docs]	>6X	> 30X
4	Neural Ranking Model B	10ms for [query, 1 passage] x 150 passages	5ms	<1ms for [query, 1 passage]; <5ms for [query, 150 passages]	>10X	> 100X
5	Query rewriting	51ms	5ms	4ms	>10X	> 3X

Throughput: 5x – 100x higher
Cost: reduced to 1% - 20% of original cost

Optimization Results Continued

	Scenarios	Original Latency	Latency Target	Optimized Latency	Latency reduction	Throughput improvement
⑥	Encoder Model A	~29ms	10ms	5.4ms	5X	5X
⑦	MRC Model C	~45ms for 1 [query, passage]	10ms	4.0ms for 1 [query, passage]; <8.5ms for 20 [query, passage]	11X	> 100X
⑧	Query tagging	9~16ms	3ms	0.95ms	10X	> 10X
⑨	Encoder Model B	~25ms for [query, 1 title url]	7ms for a batch size of 33	5.4ms for [query, 33 title url];	10X	> 100X
⑩	Classifier A	60ms	3ms	3ms	20X	20X
⑪	Classifier B	8ms	3ms	1ms	8X	8X

Latency: 5x – 20x faster, from impossible to ship to well fitting SLA
Capacity: serving 5x – 20x bigger models under the same latency SLA

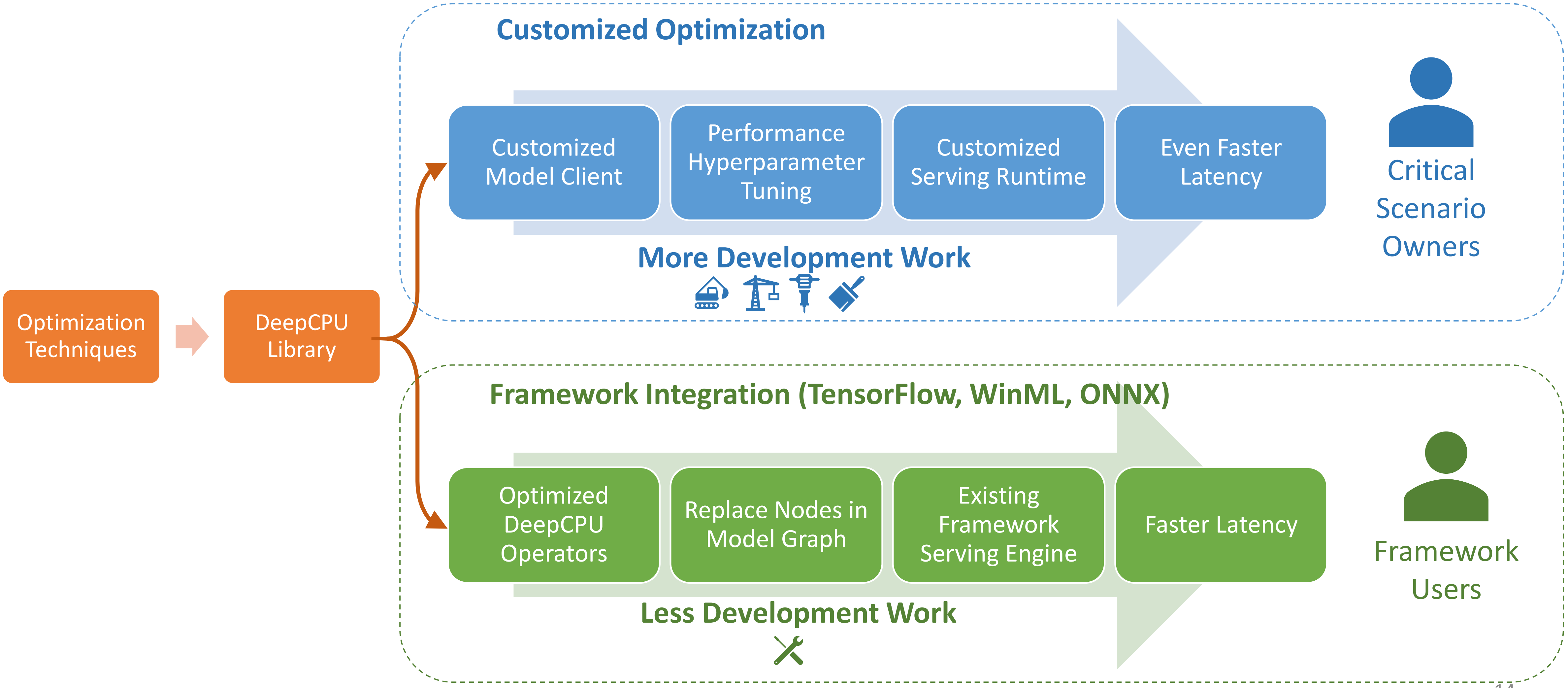
DeepCPU: Fast DL Serving Library on CPUs

- RNN family
  - GRU cell and GRU sequence
  - LSTM cell and LSTM sequence
  - Bidirectional and stacked RNN networks
- Fundamental building blocks and common DL Layers
  - Matrix multiplication kernels, activation functions
  - high-way network, max pool layer, MLP layer ……
- DL layers for MRC and conversation models
  - Variety of attention layers
  - seq2seq decoding with beam search ……

Optimization Techniques

Optimization	Our optimized library on CPU
Matrix computation	Cache-aware matrix kernels + Intel MKL
Activation functions	Vectorization + parallelization
Operation Fusing	Fuse operations to reduce data read/write
Affinity	Bind app thread to hardware thread cross-socket awareness
Locality	Private-cache-aware partitioning + weight-centric streamlining
Parallelism	Judicious parallelism considering workload, parallelism efficiency and load balancing
Task Scheduling	Priority over critical path Global optimization of DAG

How is DeepCPU Utilized?



DeepCPU: Make DL Serving Faster & More Efficient

Scenarios	Models	Usage	Impact
• Question Answering • Machine Reading Comprehension • Ranking • Query Rewriting • Query Tagging	• GRU/LSTM • Stacked RNN • Seq2Seq • Attention layers • Convolution • Highway network • MLP ……	• Customized optimization • Framework integration	• 10x faster • 10x larger models • 10x - 100x more throughput • 10x - 100x less cost

Thank You!

Questions?