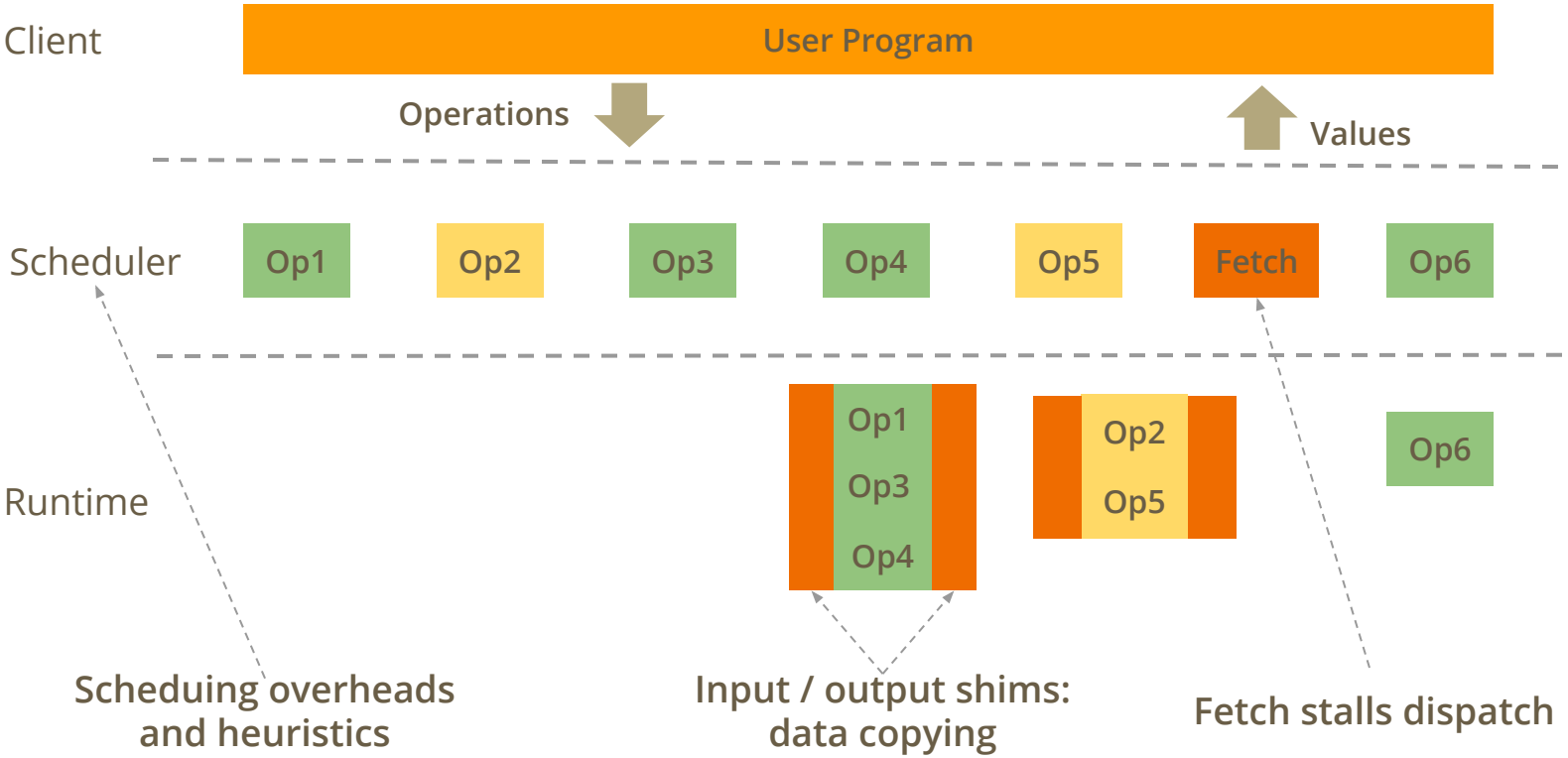
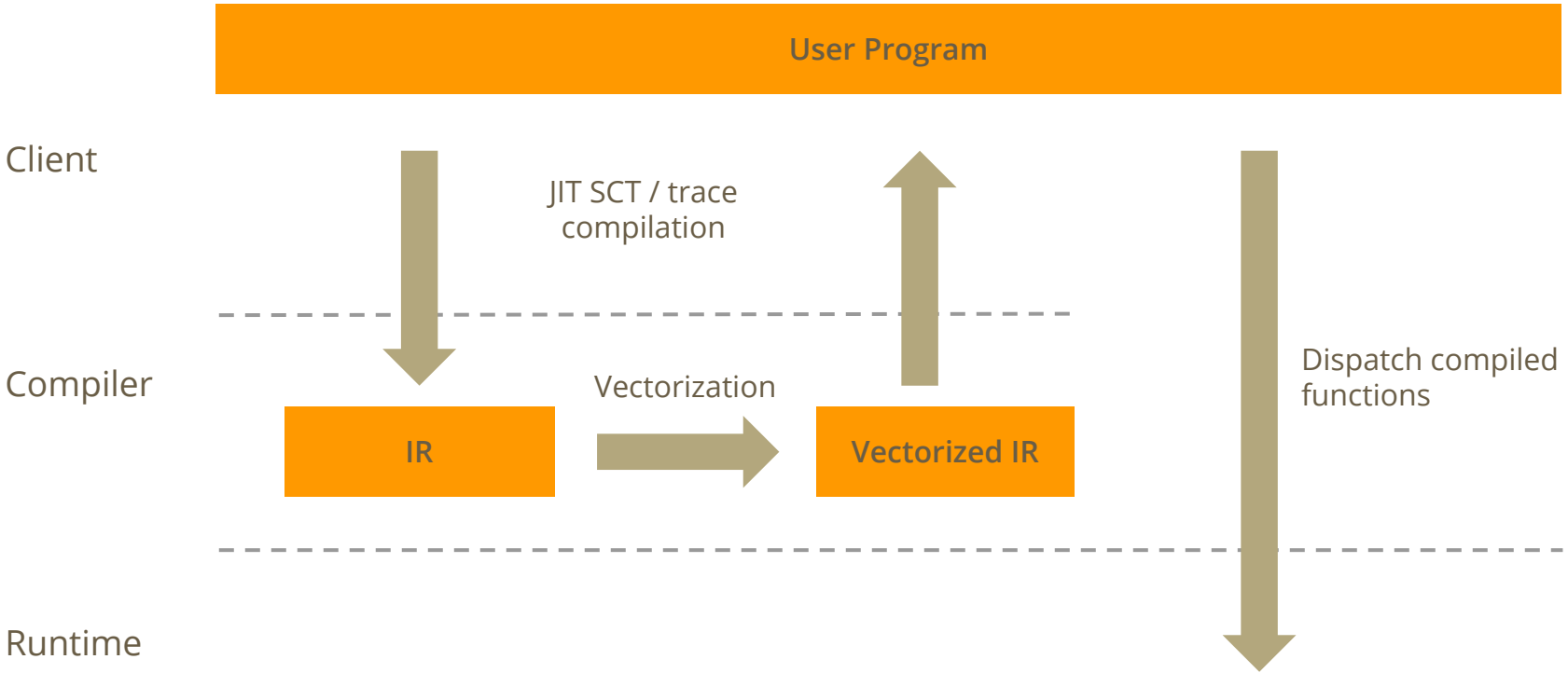
Static Automatic Batching In TensorFlow

Ashish Agarwal
Google Brain

Dynamic Automatic Batching



Static Automatic Batching



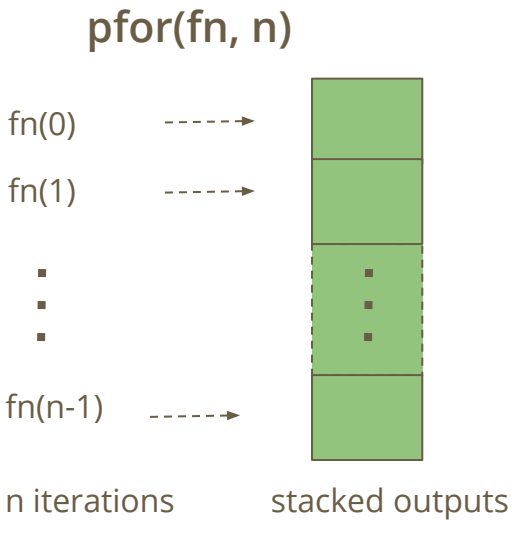
Loop Vectorization

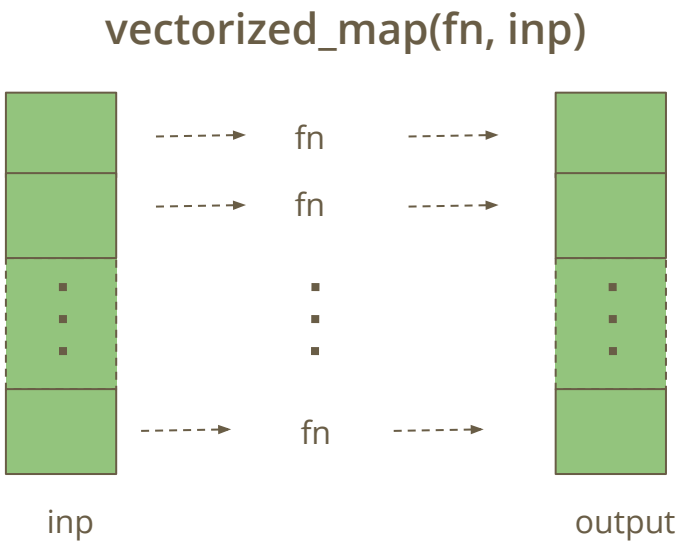
```
for i in range(n):
    c[i] = a[i] * b[i]
```

```
c = a * b
```

Tensor IR allows getting rid of loops!

Vectorization In TensorFlow





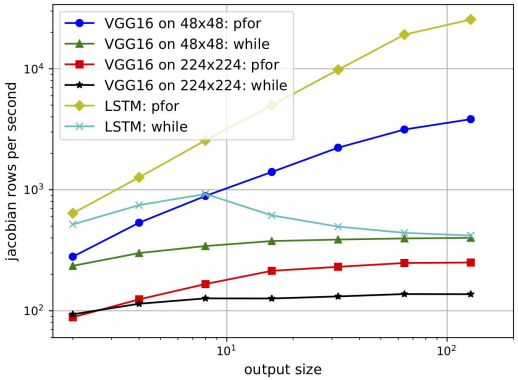
fn is the loop body. pfor semantically runs n iterations in parallel, and stacks their outputs.

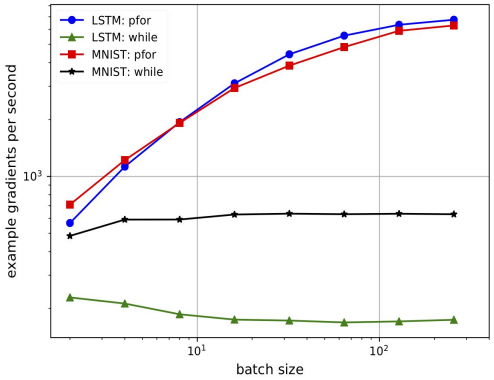
Maps fn on each row slice of inp. Similar to pfor(lambda i: fn(tf.gather(inp, i)), tf.shape(inp)[0]).

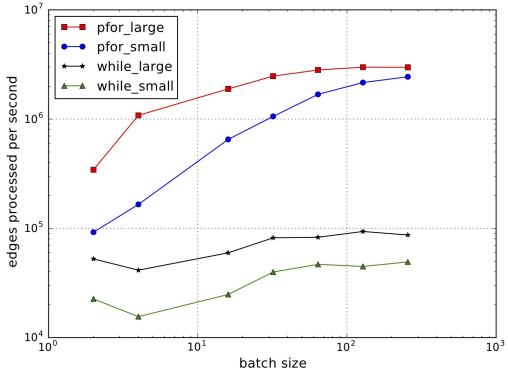
Vectorization In TensorFlow

```
# Forward pass auto-batching
tf.vectorized_map(model_fn, inputs)

# Per-example gradients
tf.vectorized_map(lambda z: tf.gradients(model_fn(z), variables), inputs)

# Jacobian
tf.vectorized_map(lambda z: tf.gradients(z, variables), outputs)

# TensorFlow jacobian API
tf.GradientTape.jacobian(output, inp)
tf.GradientTape.batch_jacobian(output, inp)
```

Benchmarks

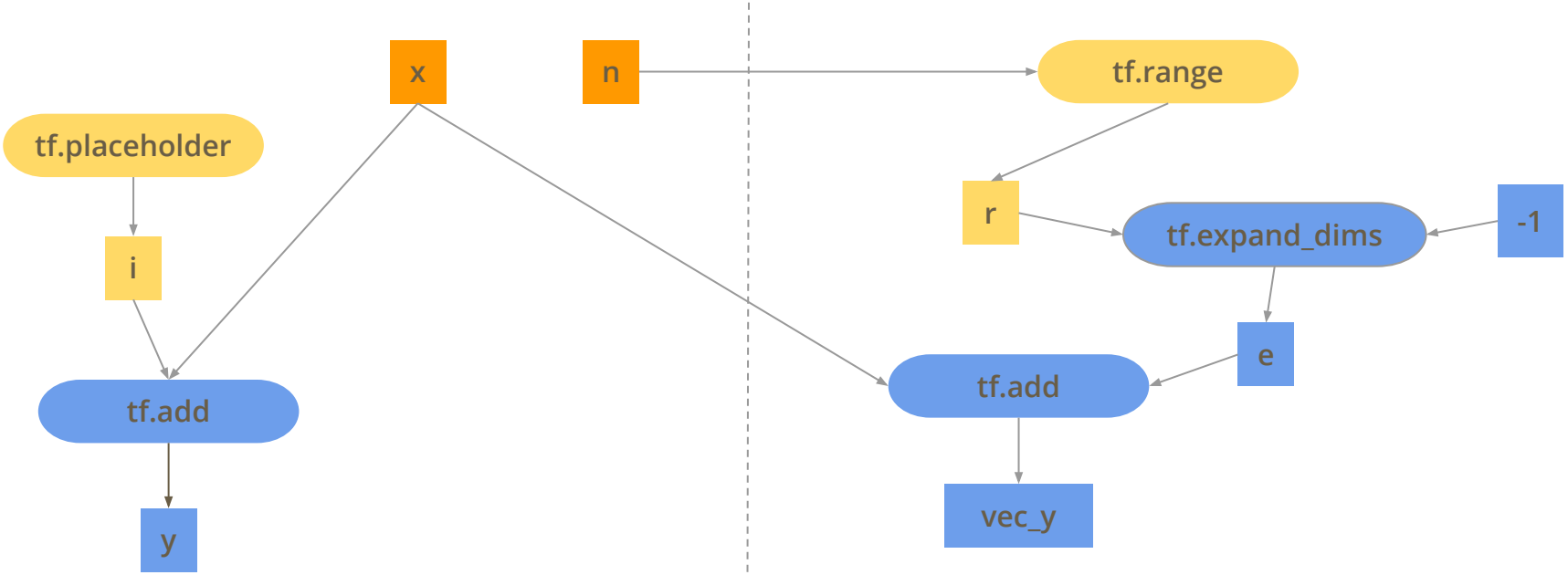






- Tested jacobians, per-example gradients, auto-batching on different models & platforms
- Up to 2 orders of magnitude speedups from vectorization
- Up to an order of magnitude speedup compared to dynamic batching
- On-par with manual batching

Vectorization In Action




```
# User code
x = tf.constant([1, 2])
n = 4
y = pfor(lambda i: x + i, n)
```

```
# Generated vectorized code
r = tf.range(4)
e = tf.expand_dims(r, -1)
vec_y = x + e
```

Vectorization Challenges

- Handling nested control flow
- Handling stateful operations
- Handling complex data structures
- Leveraging loop invariance

```
x = tf.random_uniform([128, 64, 64])
y = tf.random_uniform([128, 64, 64])
```

matmul(x[i], y[i]) ⟶

```
tf.matmul(x, y)  # BatchMatMul kernel
```

matmul(x[i], y[0]) ⟶

```
x_flat = tf.reshape(x, [128 * 64, 64])
out_flat = tf.matmul(x_flat, y[0])  # MatMul kernel
out = tf.reshape(out_flat, [128, 64, 64])
```

Thank You!