Introduction to Tensorflow Serving

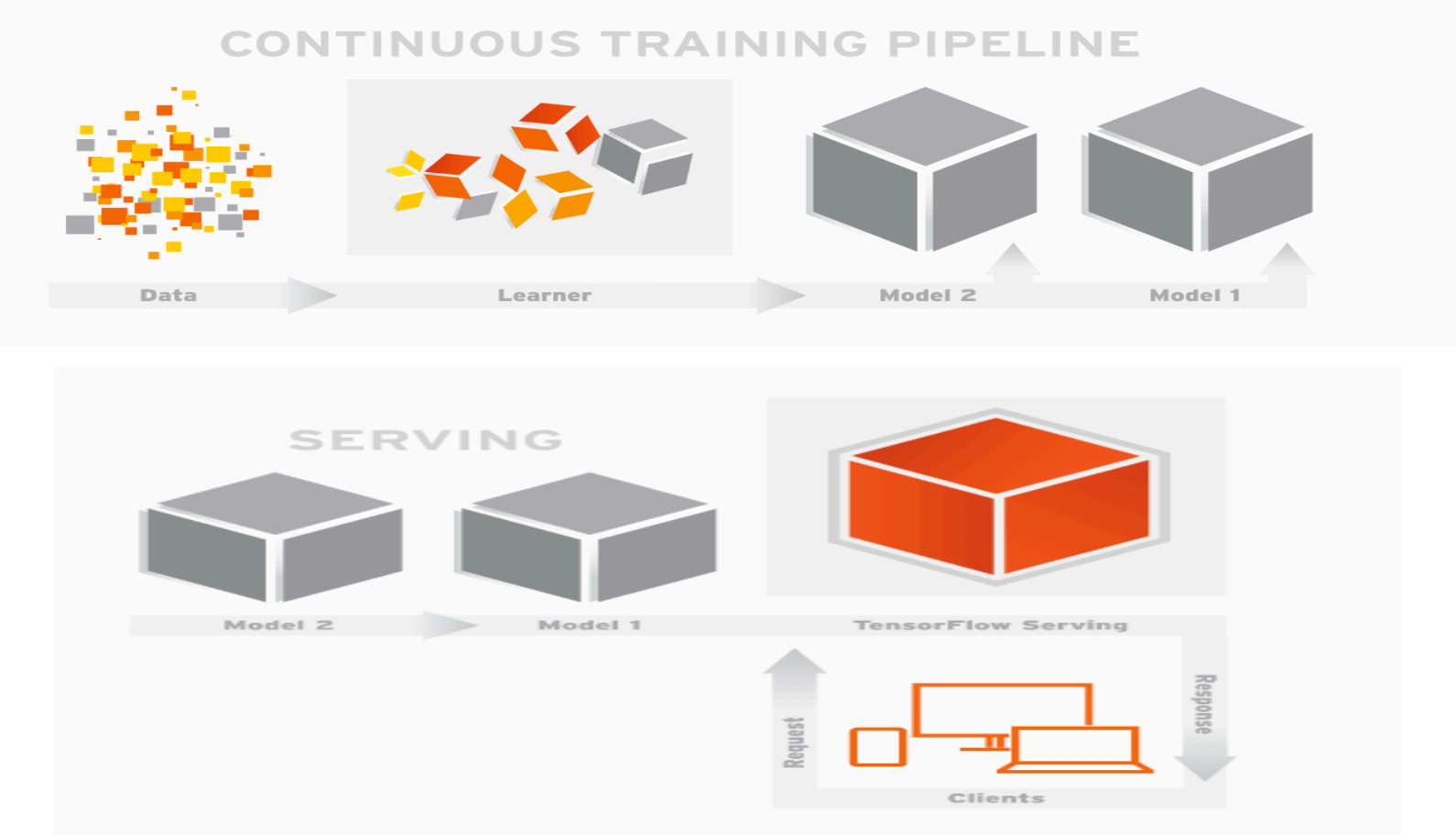
京东广告部 - 包勇军 曾凡喜 蒋在帆

2016.10

What is model server

- 定义：Model inference 的服务化
- 工业级应用
  - 广告排序
  - 商品推荐
  - 新闻推荐

Model Serving Production Environments



Why we need model serving

- Why separated module
  - 技术需求
    - 性能
      - 平响
      - QPS
    - 功能
      - 模型更新
      - 在线模型实验
      - 集群化
      - 服务化

The Old Story

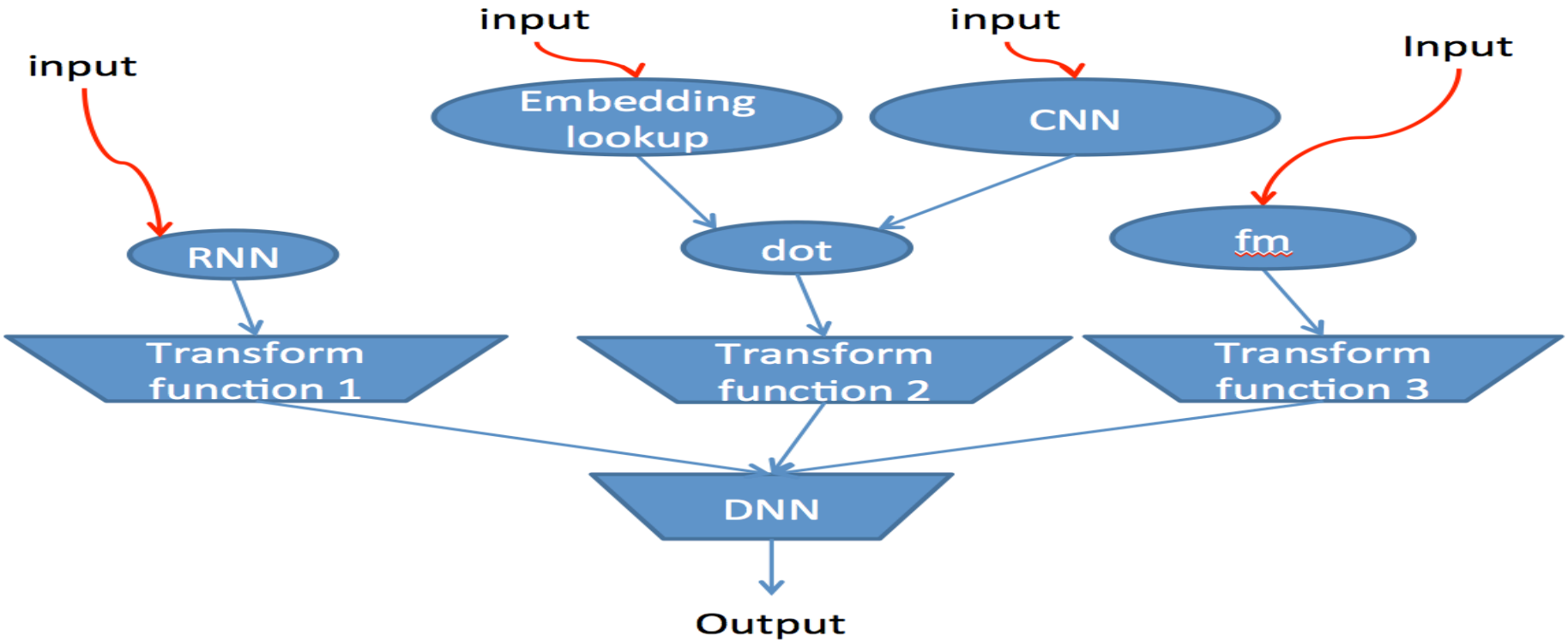
- Offline
  - Liblinear
  - LibFM
  - Mahout
  - MLlib
- Model
- Server

The Old Story

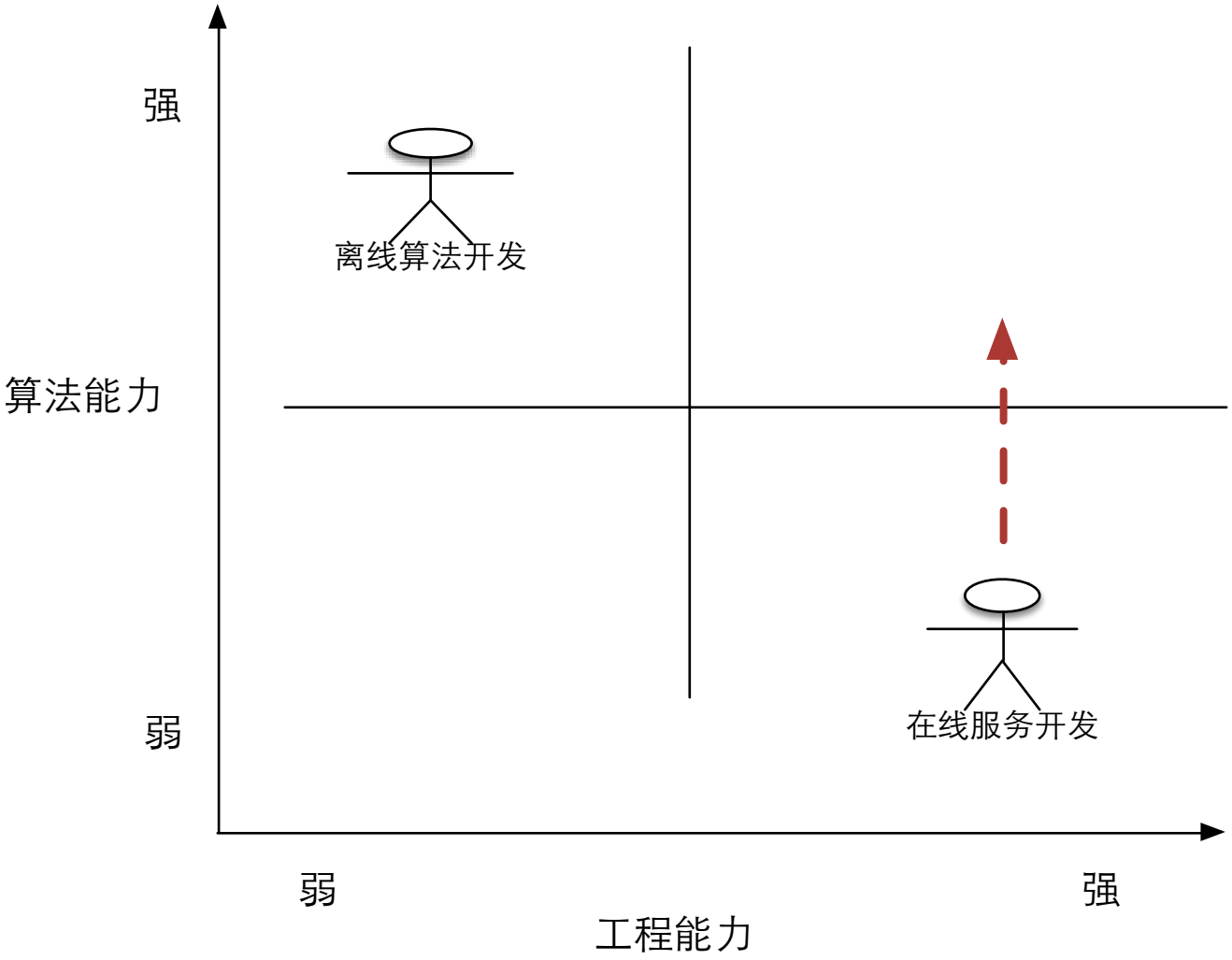
- Supported Algorithms
  - Logistic Regression
  - Factorization Machine
  - SVM
  - RandomForest
  - GBDT

The New Challenges

- 复杂的网络结构
  - 更多更复杂的算子
  - 经常变动的网络结构



The New Challenges



The New Challenges

- Server Requirements
  - 灵活的算法支持
    - 多变的网络结构
  - 高性能
    - 平响
    - 吞吐
  - 生产条件下实验支持
    - 模型管理
    - 模型更新
  - Offline/online model transfer
    - **算法**开发效率, script language based
    - 服务执行效率， system language based

TensorFlow

TensorFlow

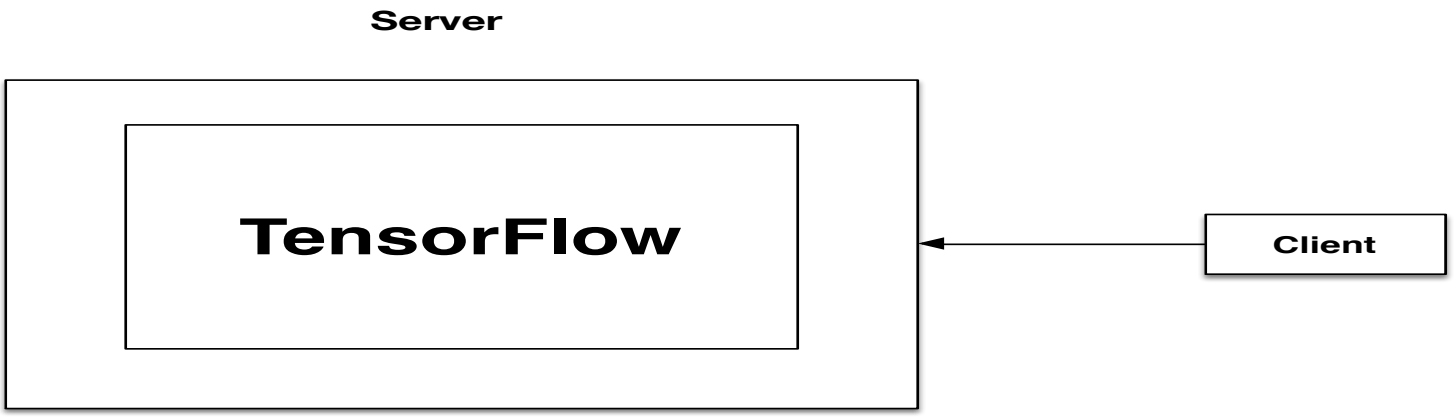
- Tensorflow
  - Google的第二代深度学习系统
  - Tensorflow is an interface for expressing machine learning algorithms and an implementation for executing such algorithms.
  - Tensorflow as an open source library for numerical computation using data flow graphs.

TensorFlow

- Why Popular
  - 强大社区支持
  - 工业级品质
  - 异构环境
  - 分布式

Tensorflow Serving

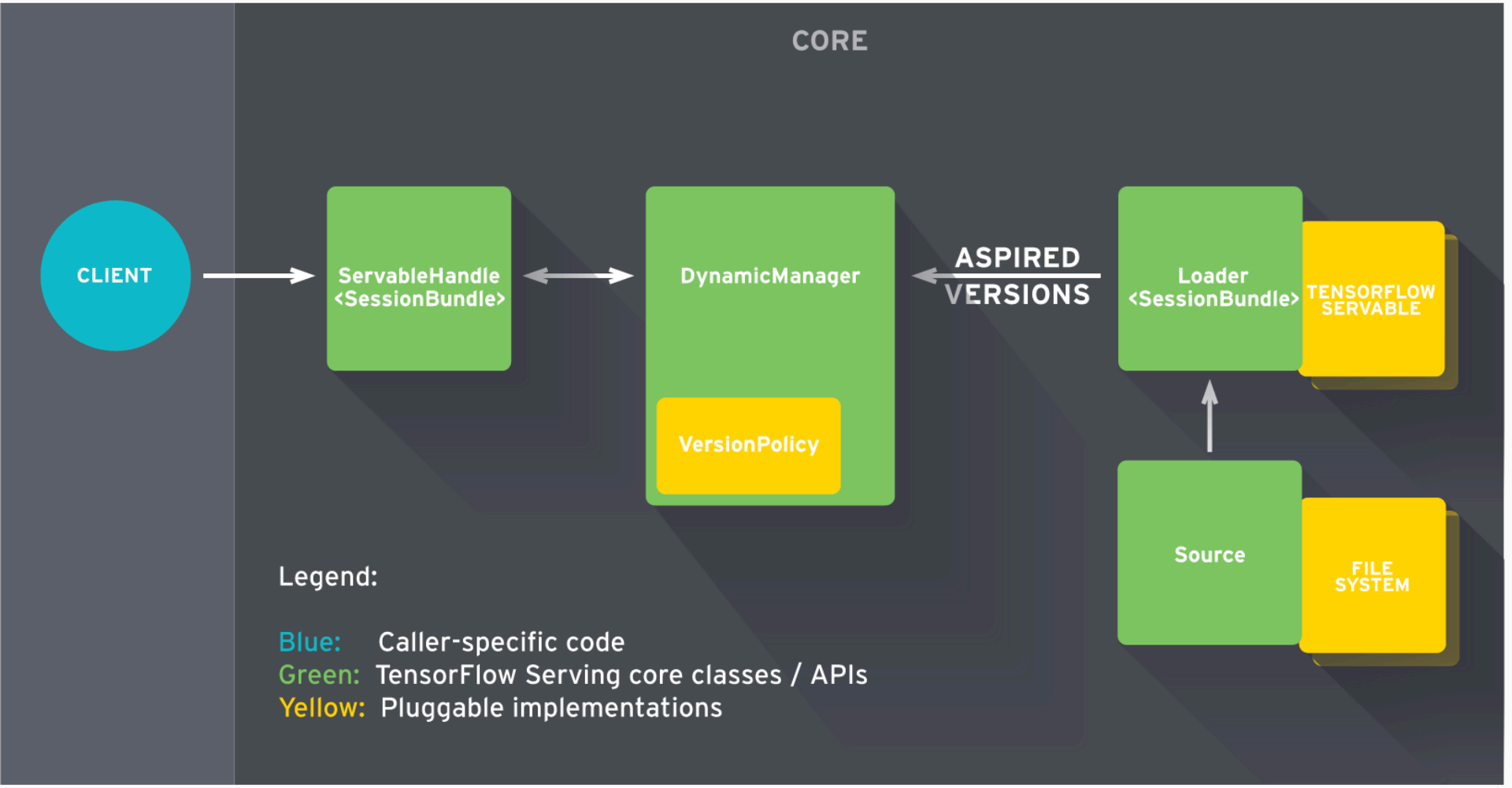
TensorFlow Serving is a flexible, high-performance serving system for machine learning models, designed for production environments. TensorFlow Serving makes it easy to deploy new algorithms and experiments, while keeping the same server architecture and APIs. TensorFlow Serving provides out-of-the-box integration with TensorFlow models, but can be easily extended to serve other types of models and data.



Tensorflow Serving

- Advantages
  - Performance
    - GPU/CPU
    - Batch requests with configurable latency for throughput
    - Distributed computing
  - Extensible, plugin mechanism
  - Production supports
    - Flexible model version managements
    - Powerful experiments supports
    - Seamless integration with tensorflow
    - Cross language supports

Tensorflow Serving Architecture



Tensorflow Serving Architiechture

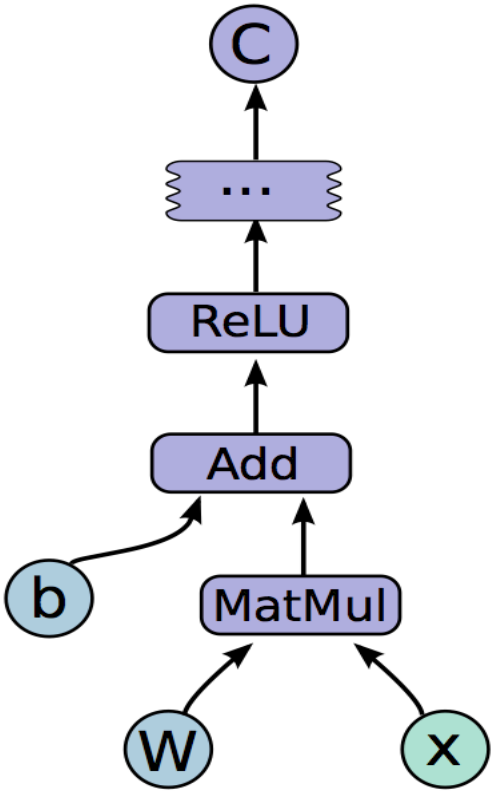
- Key Concepts
  - Model
    - Include algorithms and weights
    - Represented as One or **More** servables
  - Servable
    - the computation object
  - Servable Version
  - Servable Stream
    - Sequence of versions of a servable, in increasing order

Tensorflow Serving Architiechture

- Key Concepts
  - Loader
    - Manage the servable's lifecycle
    - Standardize the loading and unloading a servable
  - Source
    - Abstraction of model storage
  - Aspired versions
    - Versions that should be loaded

Tensorflow Server Inside

- Graph Engine



Tensorflow Server Inside

- Graph Engine Concepts
  - The graph represents a dataflow computation by a directed graph
  - Each node has 0 or more inputs/outputs, and represents the instantiation of operation
  - Values flow along edges are tensors(multi-dim array)
  - An operation has a name and represents a abstract computation. Operation can have attributes

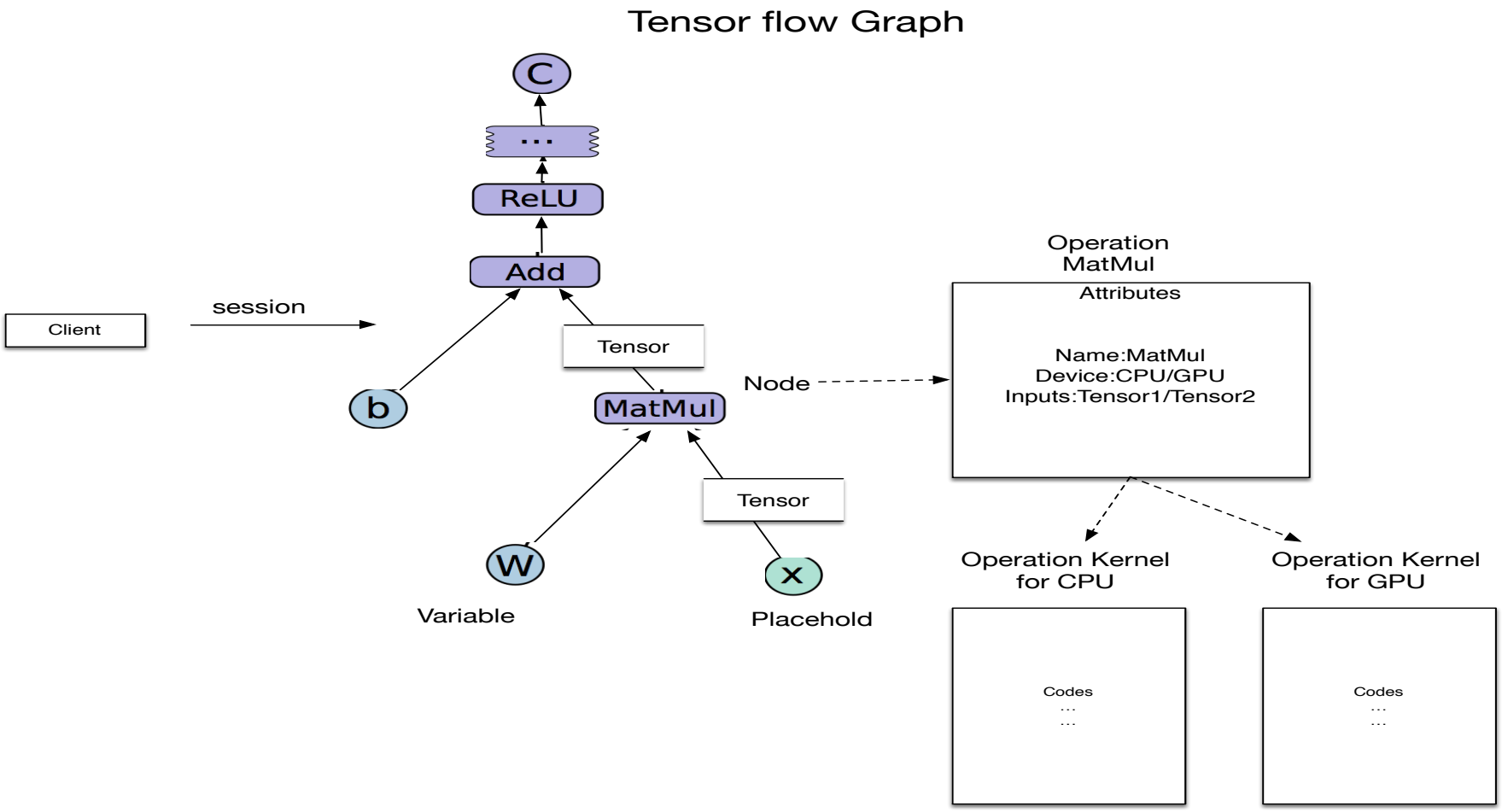
Tensorflow Server Inside

- Graph Engine Concepts
  - All attributes must be provided at graph construction time in order to instantiate a node to perform the operation
  - A kernel is a particular implementation of an operation that can run on a particular device.
  - Client use session to interact with TF system.

Tensorflow Server Inside

- Graph Engine Concepts
  - Variable is a special kind of operation that returns a handle to a persistent mutable tensor that typically stores model's parameters
  - Placeholds

Tensorflow Server Inside

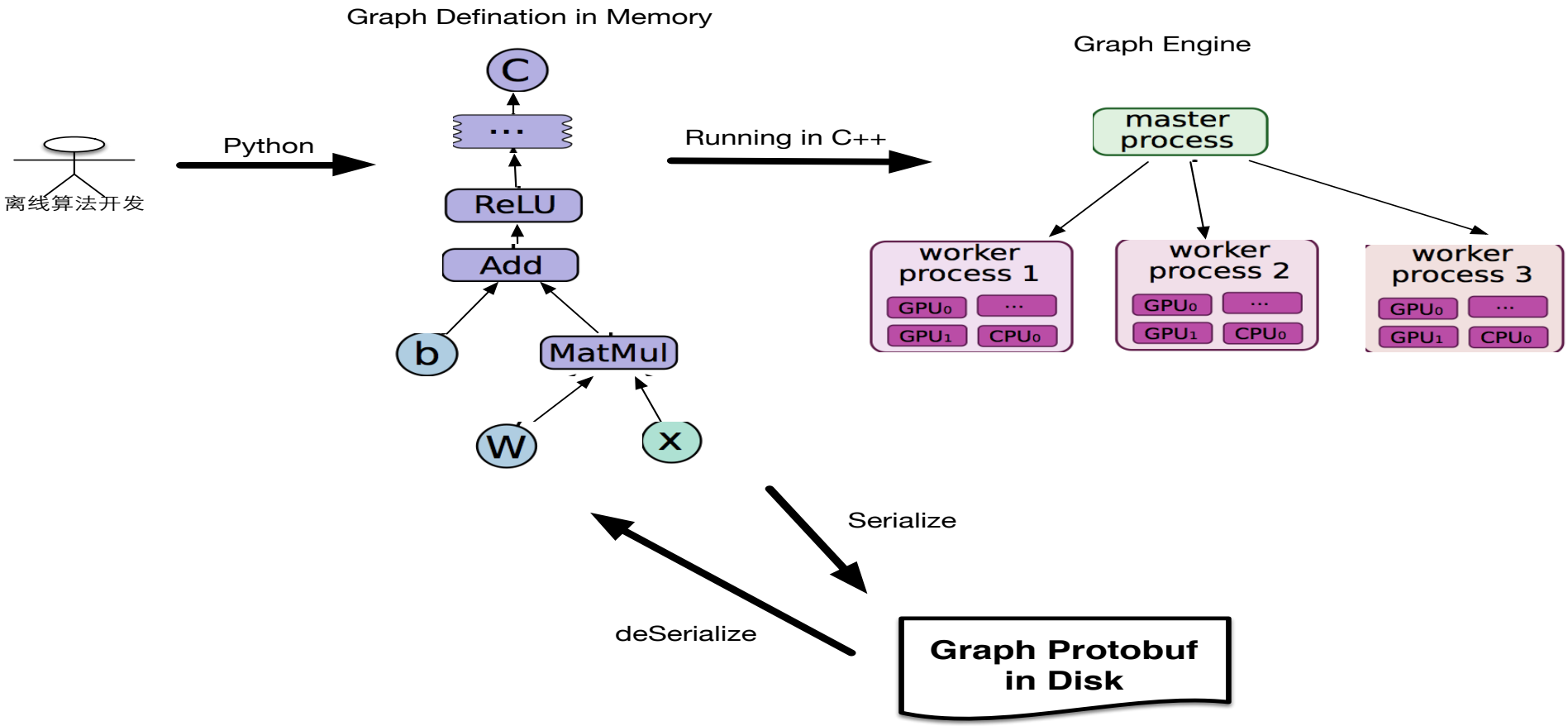


Tensorflow Server Inside

- Graph engine
  - TF所有的操作都是通过构建Graph，增加Node来完成
  - Benefits
    - Better optimization with full computation network
    - Computation is an entity
      - checkpoint
      - Distributed
    - Expression, Execution, Easy cross language support
      - C++ backend engine
      - Python frontend wrapper

Tensorflow Server Inside

- Offline/online Model transfer



Tensorflow Server Inside

- Offline/online model transfer
  - Other extensions?
    - Torch
    - https://github.com/torch/torch7/blob/6acd647cfc6617d09dedea1d00385fef6c9e3984/doc/serialization.md

Tensorflow Serving Steps

- Ideas：
  - Offline： Python generate Graph
  - Online: C++ inference Graph
  - 平衡开发效率和执行效率
- Serving Steps
  - 模型的导出和存储
  - Graph的加载和还原
  - Graph的计算

Tensorflow Serving Demo

Export the model

```
from tensorflow.contrib.session_bundle import exporter
...
export_path = sys.argv[-1]
print 'Exporting trained model to', export_path
saver = tf.train.Saver(sharded=True)
model_exporter = exporter.Exporter(saver)
model_exporter.init(
    sess.graph.as_graph_def(),
    named_graph_signatures={
        'inputs': exporter.generic_signature({'images': x}),
        'outputs': exporter.generic_signature({'scores': y})})
model_exporter.export(export_path, tf.constant(FLAGS.export_version), sess)
```

Tensorflow Serving Demo

Serve the model

```
int main(int argc, char** argv) {
  ...

  std::unique_ptr<ServerCore> core;
  TF_CHECK_OK(ServerCore::Create(
      config, std::bind(CreateSourceAdapter, source_adapter_config,
                        std::placeholders::_1, std::placeholders::_2),
      &CreateServableStateMonitor, &LoadDynamicModelConfig, &core));
  RunServer(port, std::move(core));

  return 0;
}
```

Tensorflow Serving Demo

- Run

```
$>mkdir /tmp/monitored
$>cp -r /tmp/mnist_model/00000001 /tmp/monitored
$>bazel build //tensorflow_serving/model_servers:tensorflow_model_server
$>bazel-bin/tensorflow_serving/model_servers/tensorflow_model_server --en
able_batching --port=9000 --model_name=mnist --model_base_path=/tmp/monit
ored
```

Tensorflow Server Inside

- Exported Model
  - properties
    - Protobuf based， language independent
    - With graph definitions, recoverable
      - Theano
      - Caffe
      - Torch
    - Version supports
      - Epoch based
      - Time based

TensorFlow Server Inside

- Model Details
  - Checkpoints
    - Directory structure

```
# Directory overview
00000000/
          assets/
          export.meta
          export-?????-of-?????
```

Tensorflow Server Inside

- Exported Model Contains
  - Version
  - Assets
  - MetaGraphDef
    - Graph definition
    - Signatures, specifies inputs and outputs used at interference time
    - Other metadata,
  - A checkpoint of variables of the graph

TensorFlow Server Inside

- Model version /Experiments Support
  - Multi models can be served simultaneously
  - VersionPolicy manage multi versions of model.
    - Supporting gradual rollout, in which v2 can be discovered, loaded, experimented, or reverted to while serving v1.
    - Supporting teardown v1 before bringing up v2 for minimizing resource usage.
    - customize

Tensorflow Server Inside

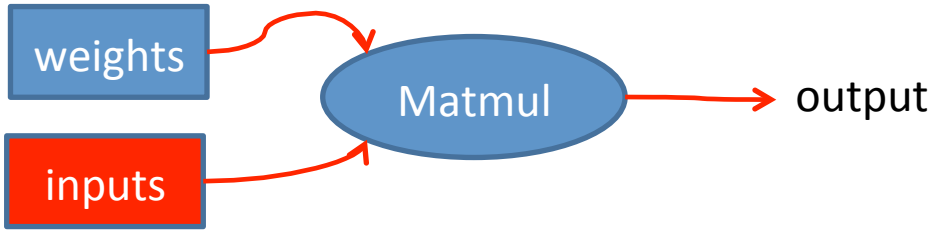
- Graph Engine Plugin Interface
  - Implementation/Register/build
  - Operation 对应了Graph中的Node，定义OP的输入，输出
  - OpKernel实现了OP的运算逻辑
  - Steps:
    - Defines the op's interface
    - Implements the kernel
    - Registering the op
    - Implement the gradient

Tensorflow Server Inside

- Other plugins
  - Model sourcer/loader
  - Model upgrades policy
  - Model assets
  - Service Monitor
  - Request batching policy
  - Servable
    - 任何有数据更新需求的服务

Tensorflow Server Inside- OP Plugin examples

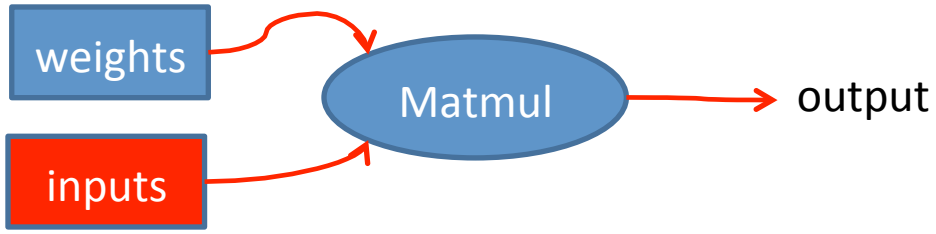
- **Plugin Examples：　注册MatMul Opkernel**




```
REGISTER_KERNEL_BUILDER(                                                \
    Name("MatMul").Device(DEVICE_CPU).TypeConstraint<T>("T"),           \
    MatMulOp<CPUDevice, T, false /* cublas, ignored for CPU */>);       \
```

MatMul as an example—Register Op

- 注册MatMul Operation




```
REGISTER_OP("MatMul")
    .Input("a: T")
    .Input("b: T")
    .Output("product: T")
    .Attr("transpose_a: bool = false")
    .Attr("transpose_b: bool = false")
    .Attr("T: {half, float, double, int32, complex64, complex128}")
    .SetShapeFn(shape_inference::MatMulShape)
    .Doc(R"doc(
Multiply the matrix "a" by the matrix "b".

The inputs must be two-dimensional matrices and the inner dimension of
"a" (after being transposed if transpose_a is true) must match the
outer dimension of "b" (after being transposed if transposed_b is
true).

*Note*: The default kernel implementation for MatMul on GPUs uses
cublas.

transpose_a: If true, "a" is transposed before multiplication.
transpose_b: If true, "b" is transposed before multiplication.
)doc");
```

MatMul as an example—Register Gradient Op

MatMulGrad:

```
Status MatMulGrad(const AttrSlice& attrs, FunctionDef* g) {
  return MatMulGradCommon("MatMul", "transpose_a", "transpose_b", attrs, g);
}
REGISTER_OP_GRADIENT("MatMul", MatMulGrad);
```

MatMulGradCommon:

```
if (!ta && !tb) {
  return MatMulGradHelper(g, opname, attr_adj_x, attr_adj_y, "dz", false, "y",
                          true, "x", true, "dz", false);
}
```

Gradients:dx, dy, dz

$Z = x * y$

$dx = dz * \partial z/\partial x = dz * y$

$dy = \partial z/\partial y * dz = x * dz$

x0="dz"

x1="y"

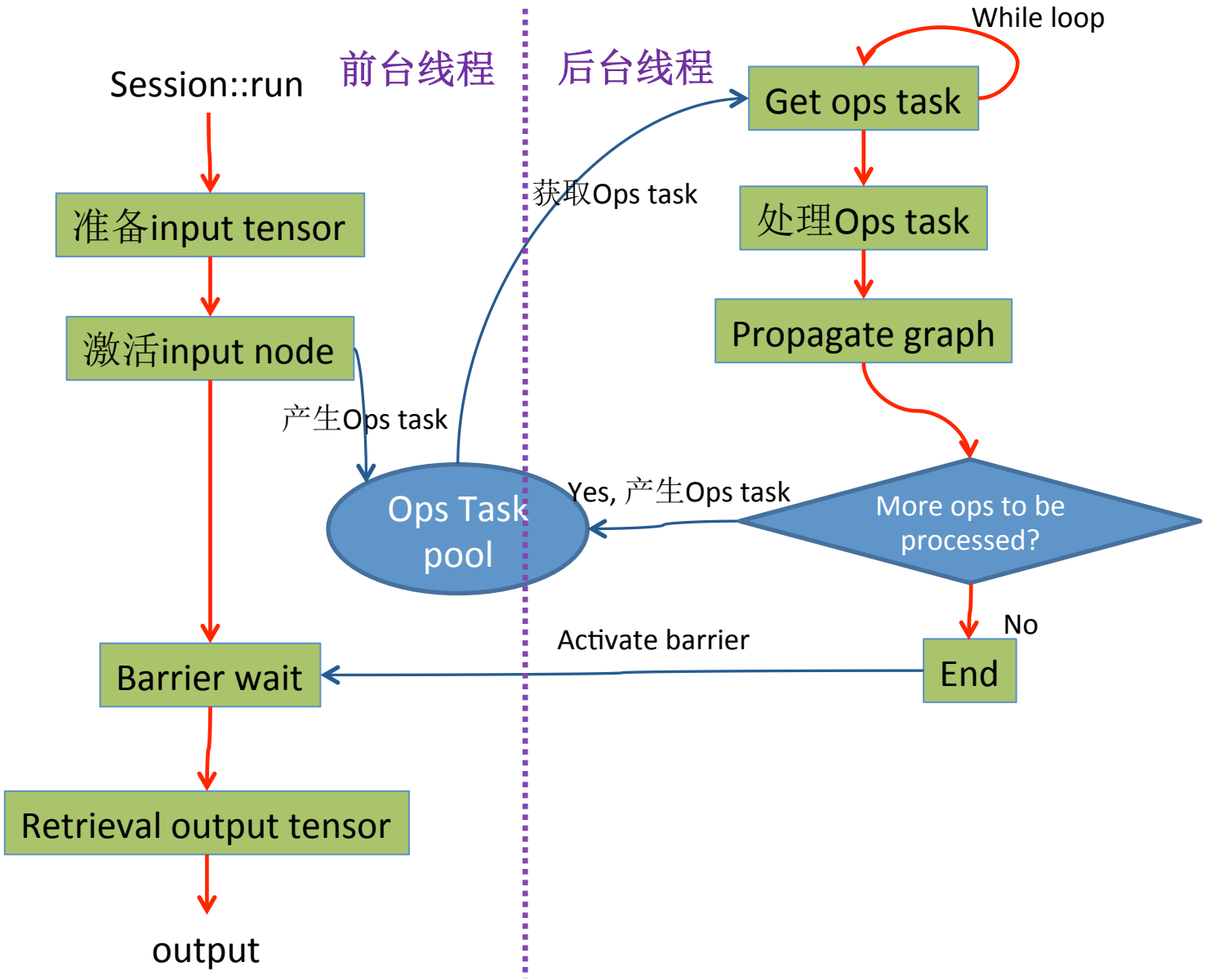
y0="x"

y0="dz"

```
*g = FDH::Define(
    // Arg defs
    {"x: T", "y: T", "dz: T"},
    // Ret val defs
    {"dx: T", "dy: T"},
    // Attr defs
    {{"T: {half, float, double}"}},
    // Nodes
    {
        {{"dx"},
         opname,
         {x0, x1},
         {{"T", "$T"}, {attr_adj_x, ax0}, {attr_adj_y, ax1}}},
        {{"dy"},
         opname,
         {y0, y1},
         {{"T", "$T"}, {attr_adj_x, ay0}, {attr_adj_y, ay1}}},
    });
```

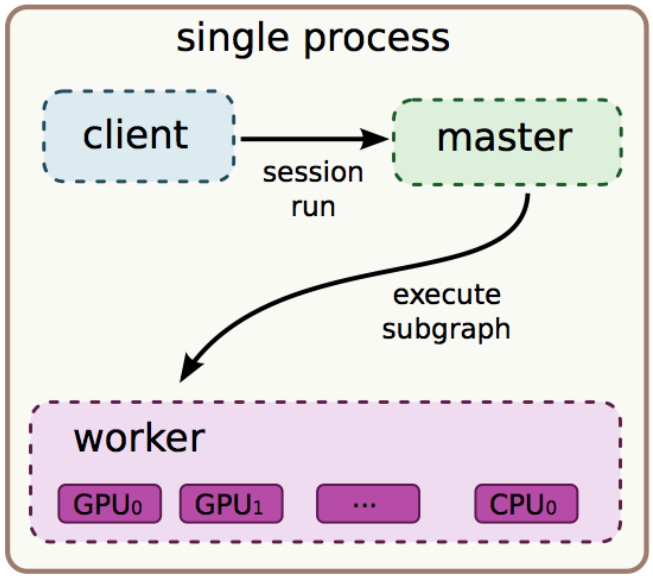
Tensorflow Server inside

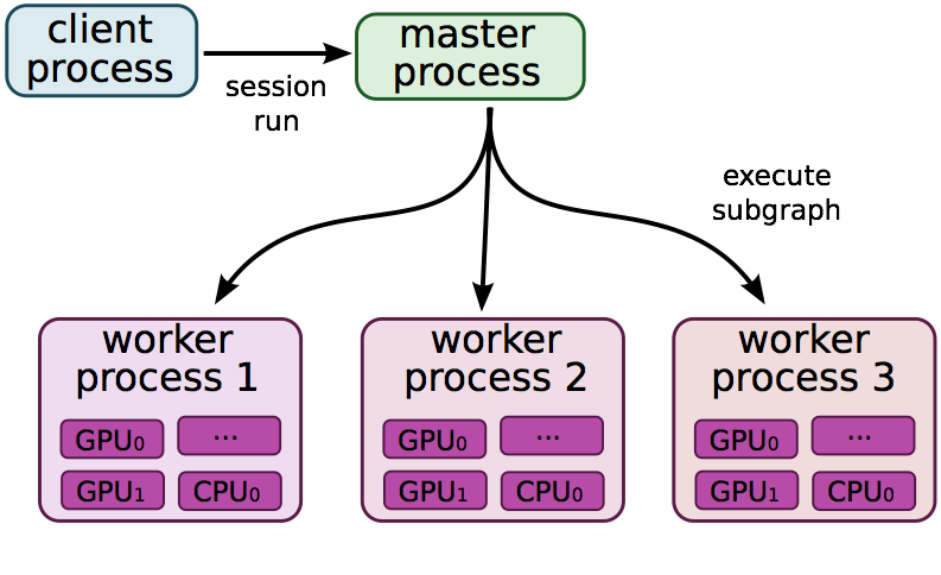
- Graph Engine execution
  - Key points
    - Analysis node dependencies
    - Each node is a computation task
    - Queuing the ready node when its number of dependencies drops to zero.
    - Task can be asynchronous


Session::run
前台线程
后台线程
While loop
Get ops task
获取Ops task
处理Ops task
准备input tensor
Propagate graph
激活input node
产生Ops task
Ops Task pool
Yes, 产生Ops task
More ops to be processed?
No
Activate barrier
End
Barrier wait
Retrieval output tensor
output

Tensorflow Server inside

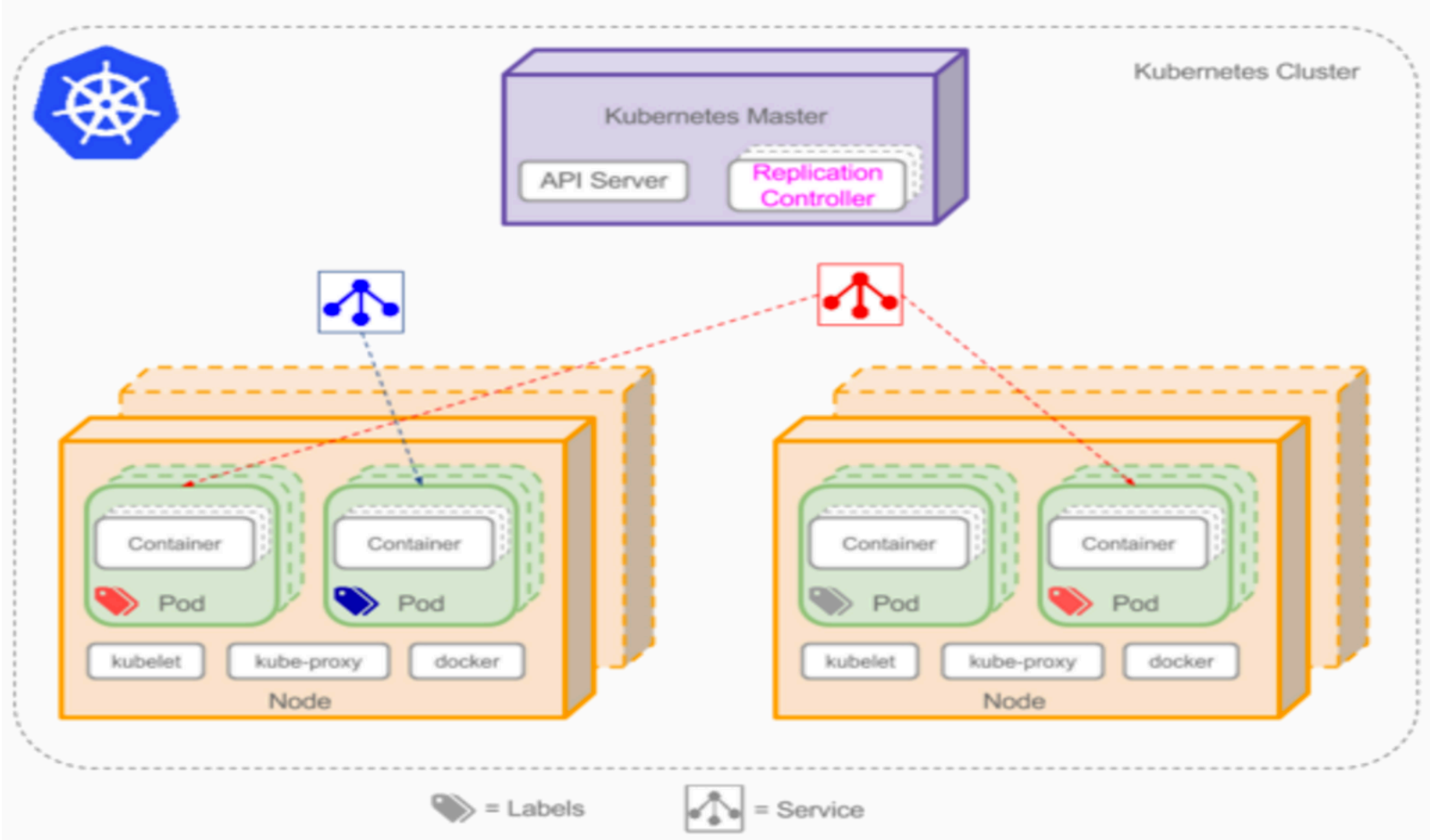
- Graph Engine execution
  - Distributed





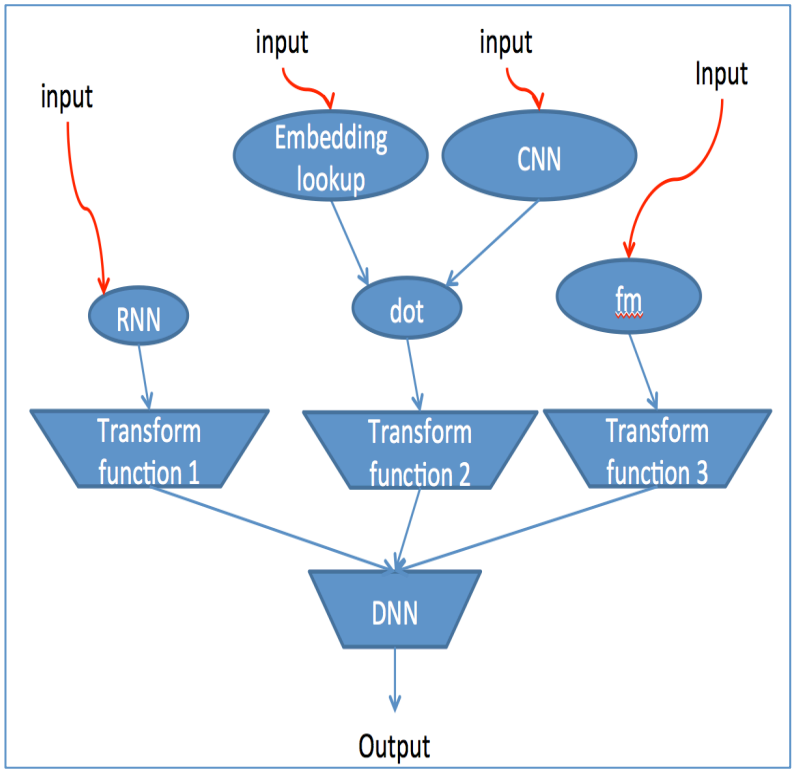
Tensorflow Server Inside

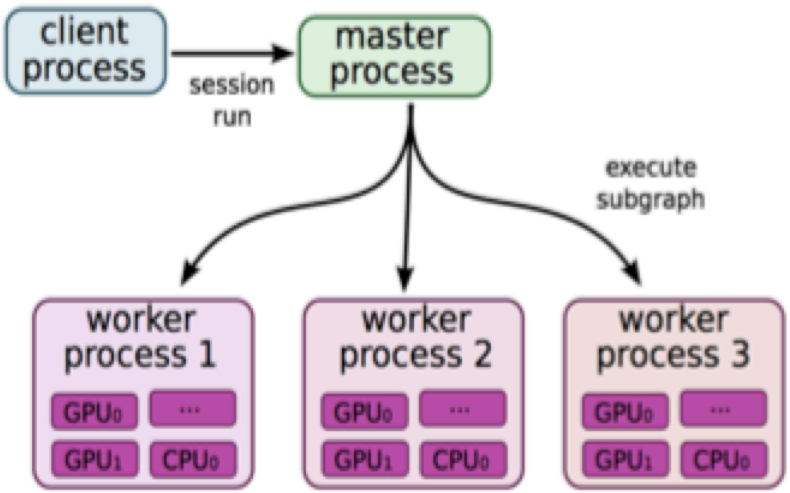
- Distributed computing-replication by Kubernetes



Tensorflow Server Inside

- Distributed computing-Distributed Graph





Tensorflow Serving

- 收益
  - 开发效率
    - 插件机制，扩展算法
    - Serving和training同一个执行引擎，实现一次
  - 性能
    - C++ code的执行引擎
    - GPU支持
    - 分布式支持

Tensorflow Serving

- Problem
  - Complexity
    - Too many abstractions for flexibility
      - 为了适应不同的环境，对各种“变”做了抽象
      - E.g. 模型加载做的抽象：Loader/Source/SourceAdapter/ SourceAdapterCreator/SessionBundle/ SessionBundleSourceAdapater/AspiredVersionManager
    - 去掉不必要的定制接口，简化
  - Not stable
    - Interface , concepts or implementations may be changed later.

What we did

- Large model support
  - 5X larger than original version model
- Large Scale sparse matrix computation support
- Engine performance boosting
- JD AD Deep Model support
  - LR-DNN-CNN-RNN mixture model
- Other JD environment supports
  - JIMDB based Embedding service
  - JD AD RemoteFS model transfer
  - JD AD Logging System Integration
- Others…

Thanks!

- Q&A.