Open Source AceCon
2021 智能云边开源峰会
AI x Cloud Native x Edge Computing
人工智能 × 云原生 × 边缘计算

TVM backend的机器学习推理加速

项目MLInferBooster介绍

陈铁军 VMware – OCTO, ATG
黄雍涛 VMware – OCTO, ATG
徐梓桐 VMware – OCTO, ATG

Project MLInferBooster

Agenda

- Background – our focus
- Project MLInferBooster Introcution
- Next

ML/AI upstream frameworks

Problem area

- Limited AI accelerators enablement
  - Types - GPU/FPGA
  - Vendors - Nvidia, AMD and Xilinx
  - ❖ Other accelerators ?
- Limited AI performance
  - Focuses on training
  - ❖ Differentiated optimization technologies ?
    - ❑ Training vs Inference
- Limited to native environment
  - Host = Target
  - Running on real AI accelerator
  - ❖ Cross arches? No HW accelerators?

SW Accelerator

Graph compiler

- The levels of the ML pyramid
  - The low-level libraries
  - Deep learning frameworks
  - Compiler
- The Graph compiler
  - What
  - The goal
  - Projects
    - ❑ TensorRT
    - ❑ XLA
    - ❑ Glow
    - ❑ TVM
    - ❑ …

Project MLInferBooster

The backend accelerator – TVM

- TVM
  - “Apache TVM is a compiler stack for deep learning systems.”
- Why
  - Open source
  - TVM supports most AI/ML frameworks
  - TVM targets various types of AI accelerators
    - Including CPU
  - Cross-compiling
    - Host /= Target
  - Good performance
    - *Only for inference now*

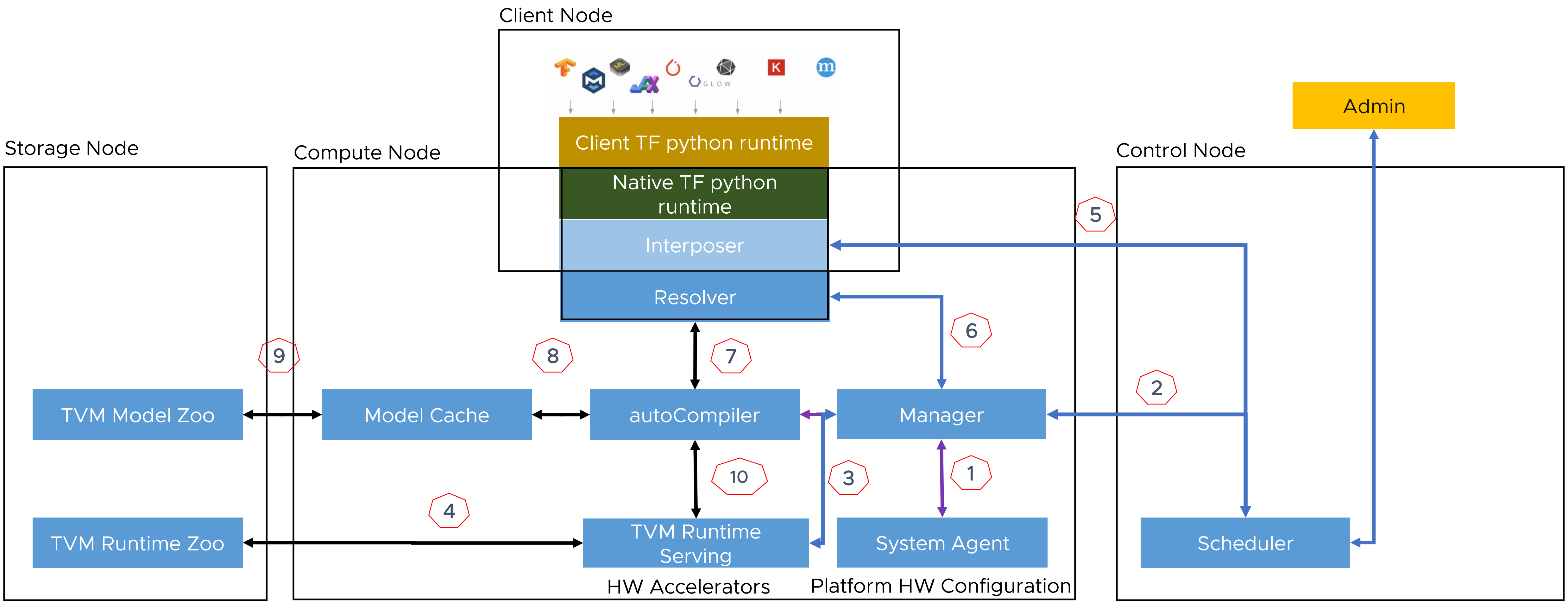
Project MLInferBooster

Solution

- Focus
  - ML Inference
- Target
  - MI Inference Acceleration by TVM
- Goal
  - Build a TVM Serving System
    - □ Backend
    - □ Automated
    - □ Unified server architecture
- How
  - Interpose ML framework python API
  - TVM progressing – Auto {detecting, compiling, scheduling, inferencing, etc}
    - □ HW accelerator type
    - □ Model & Mode info – {input, output} layer, shape, etc
    - □ TVM API : ML framework API

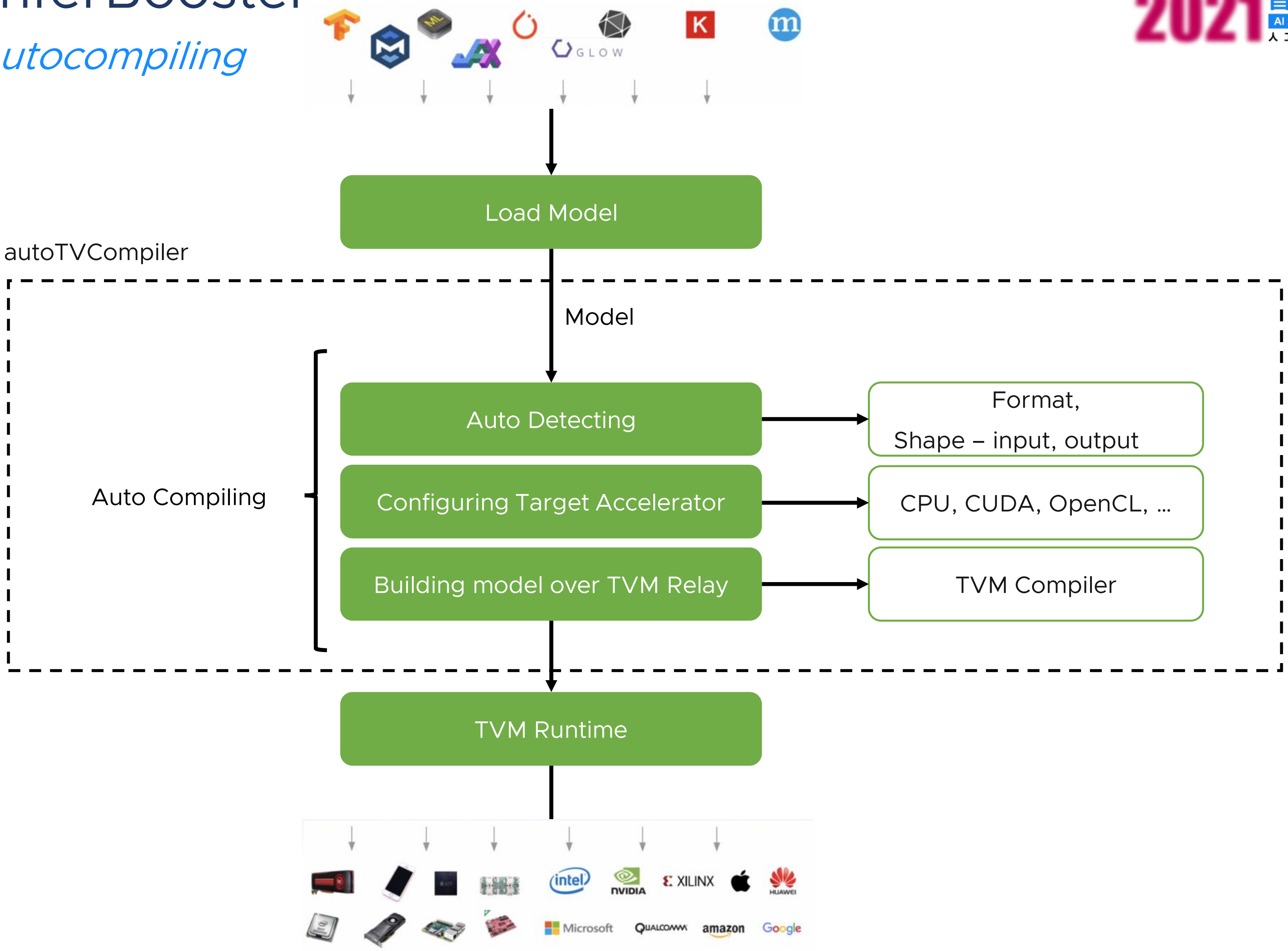
Project MLInferBooster

Architecture Overview



Project MLInferBooster

Components - autocompiling



Load Model

autoTVCompiler

Model

Auto Compiling

Auto Detecting → Format, Shape – input, output

Configuring Target Accelerator → CPU, CUDA, OpenCL, …

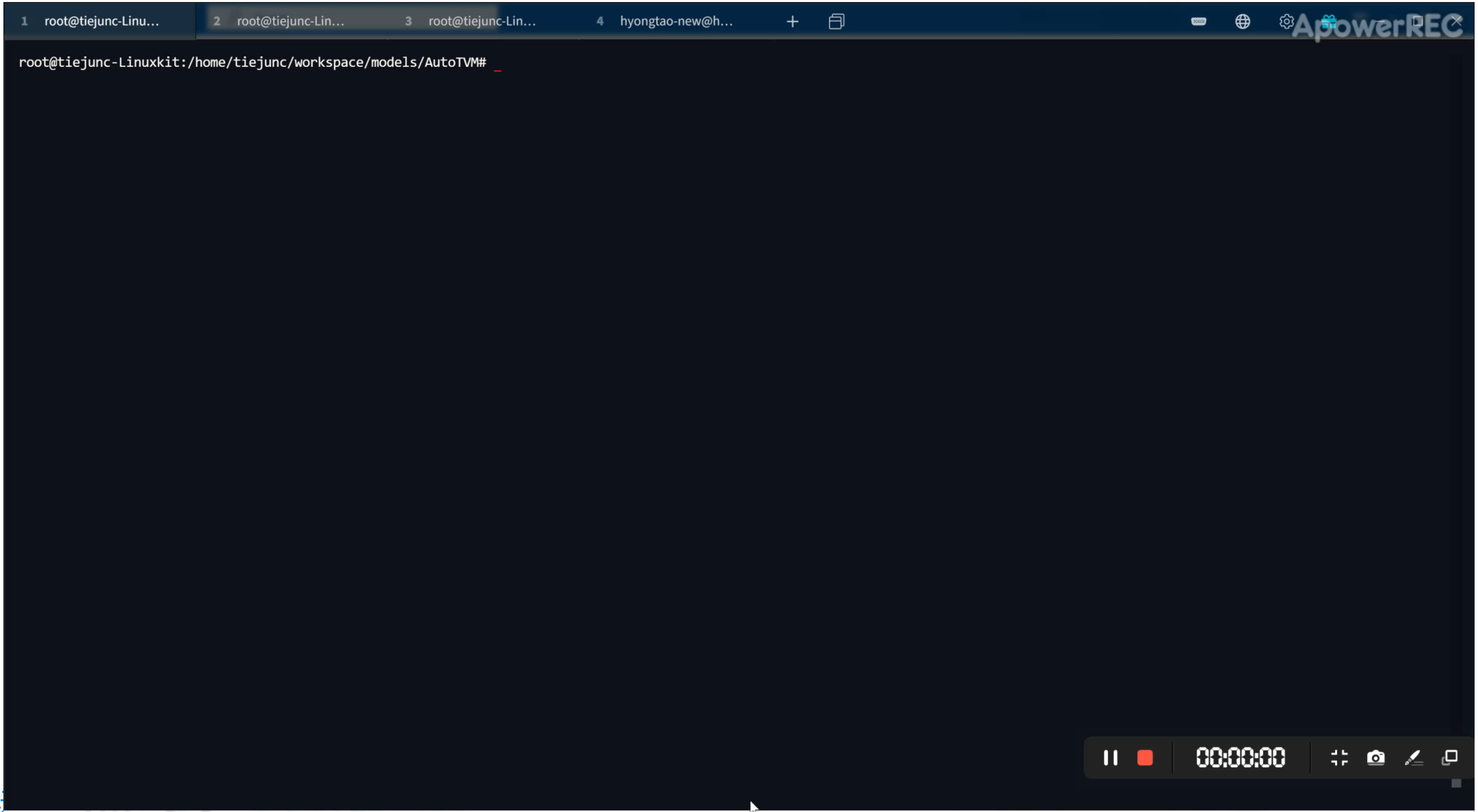
Building model over TVM Relay → TVM Compiler

TVM Runtime

Project MLInferBooster

Demo A1

Open Source AceCon 2021 智能云边开源峰会 AI x Cloud Native x Edge Computing 人工智能 × 云原生 × 边缘计算



```
root@tiejunc-Linuxkit:/home/tiejunc/workspace/models/AutoTVM# _
```

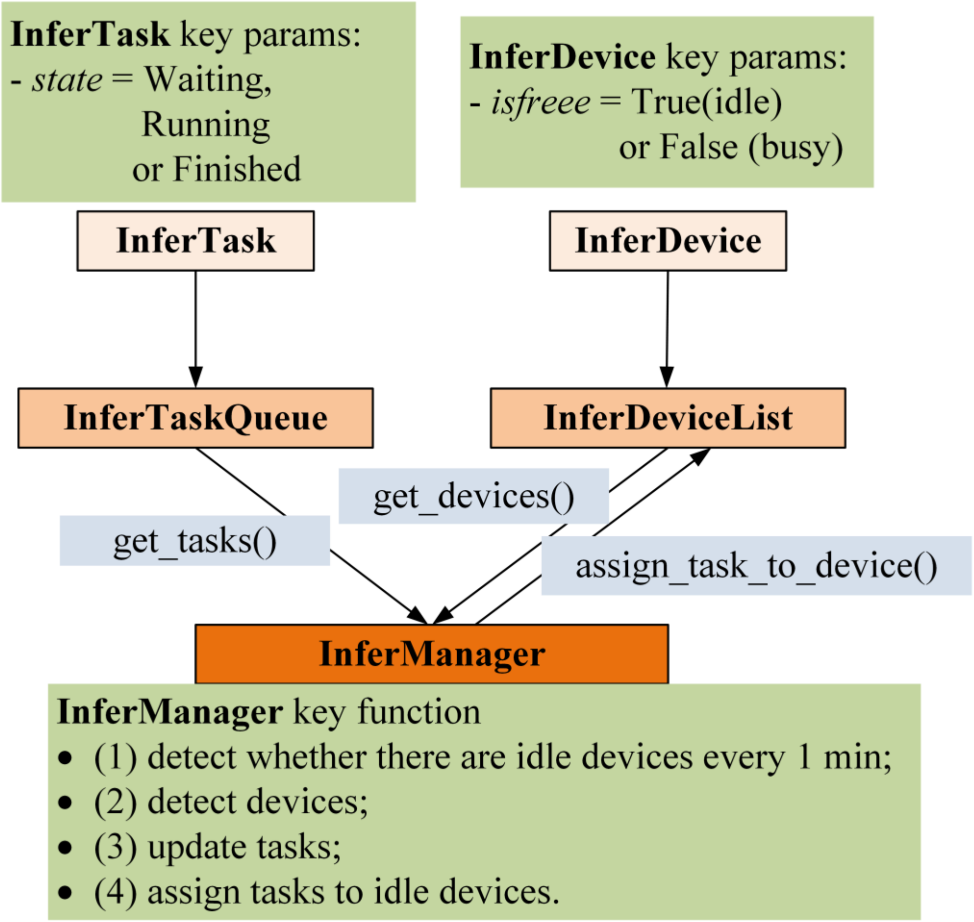
边缘计算分论坛

Project MLInferBooster

Demo A2

Project MLInferBooster

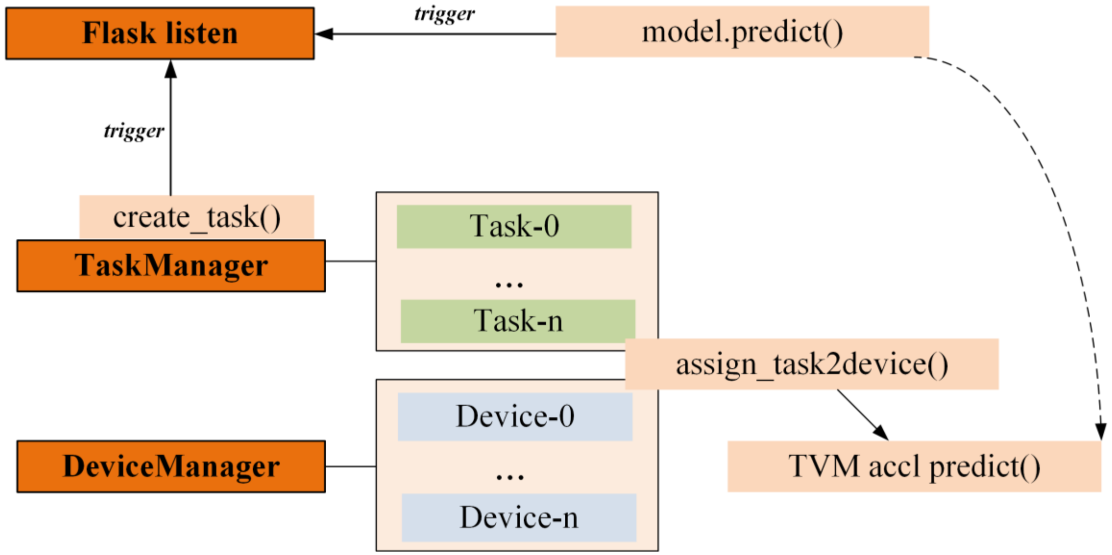
Components - Scheduler



InferTask key params:
- *state* = Waiting,
  Running
  or Finished

InferDevice key params:
- *isfreee* = True(idle)
  or False (busy)

InferManager key function
- (1) detect whether there are idle devices every 1 min;
- (2) detect devices;
- (3) update tasks;
- (4) assign tasks to idle devices.

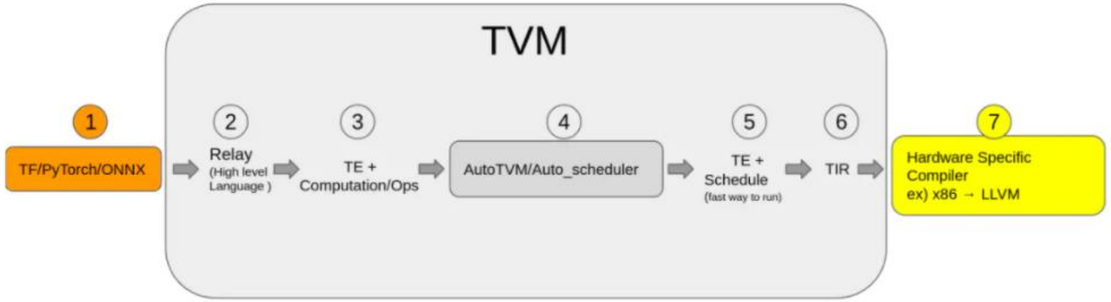


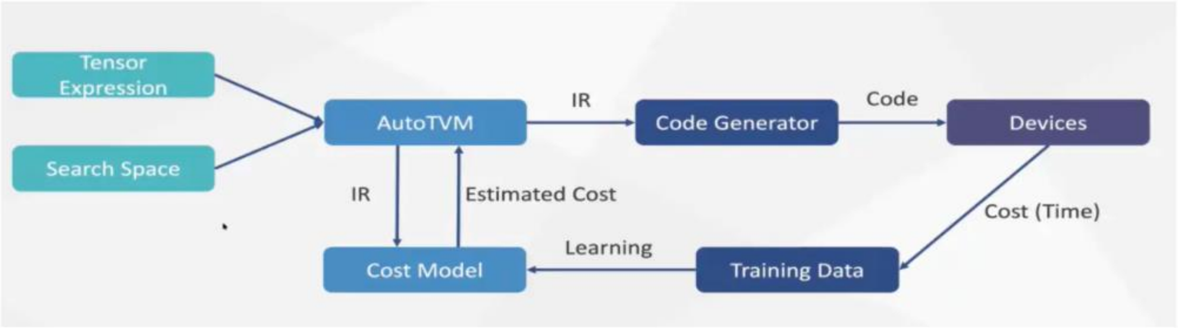
Project MLInferBooster

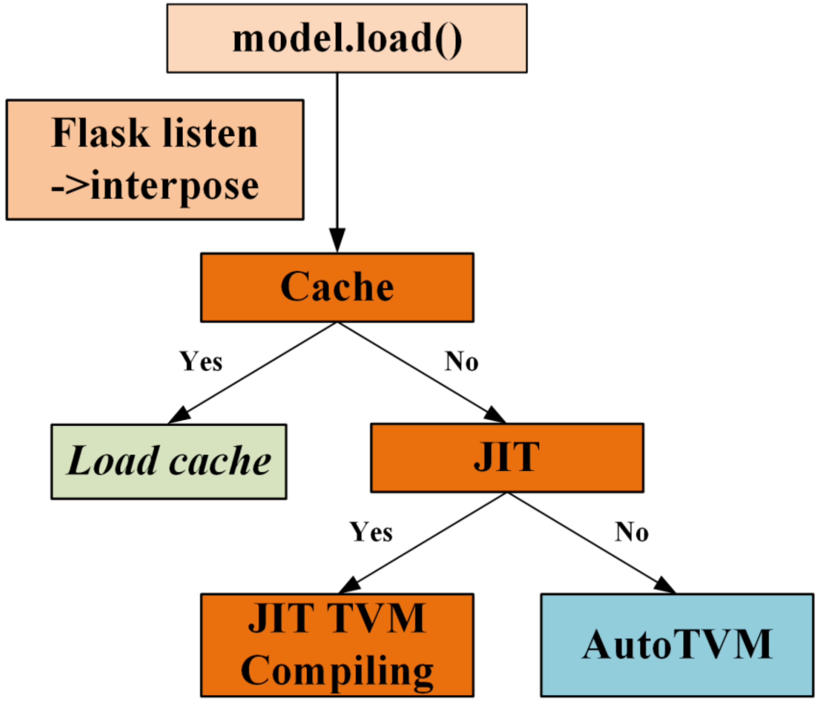
Demo Auto-Schedule

Project MLInferBooster

Components - AutoTVM







model.load()

Flask listen
->interpose

Cache

Yes

No

Load cache

JIT

Yes

No

JIT TVM
Compiling

AutoTVM

参考资料：
[1] 《Learning to Optimize Tensor Programs》NIPS-2018，陈天奇。
[2] https://zhuanlan.zhihu.com/p/37181530　陈天奇 知乎。

Project MLInferBooster

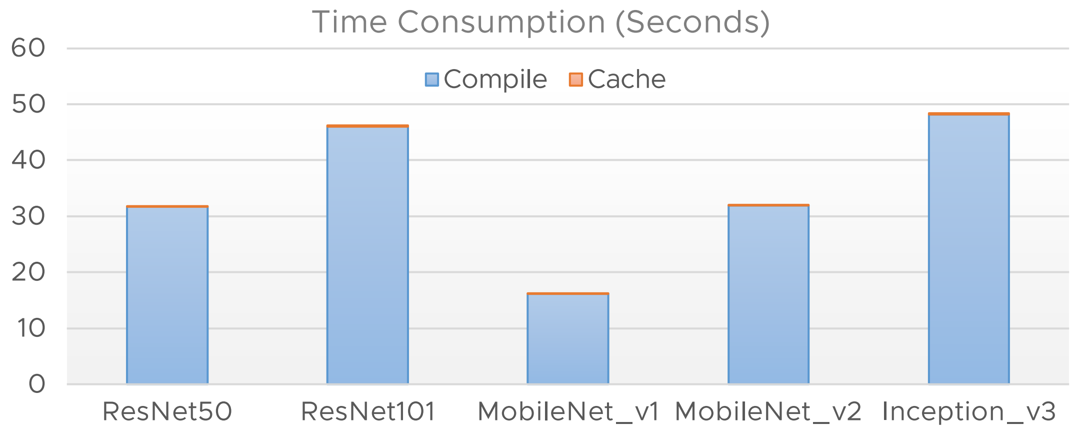
Demo AutoTVM tuning

Project MLInferBooster

Components - Model Cache

- Objectives
  - Cache the compiled model information
  - Mapping mechanism
  - Least Frequently Used (LFU) cache replacement policy

- Benefits
  - Avoid recompile of the same model and save time
  - Apply efficient strategy to prevent cache overflow



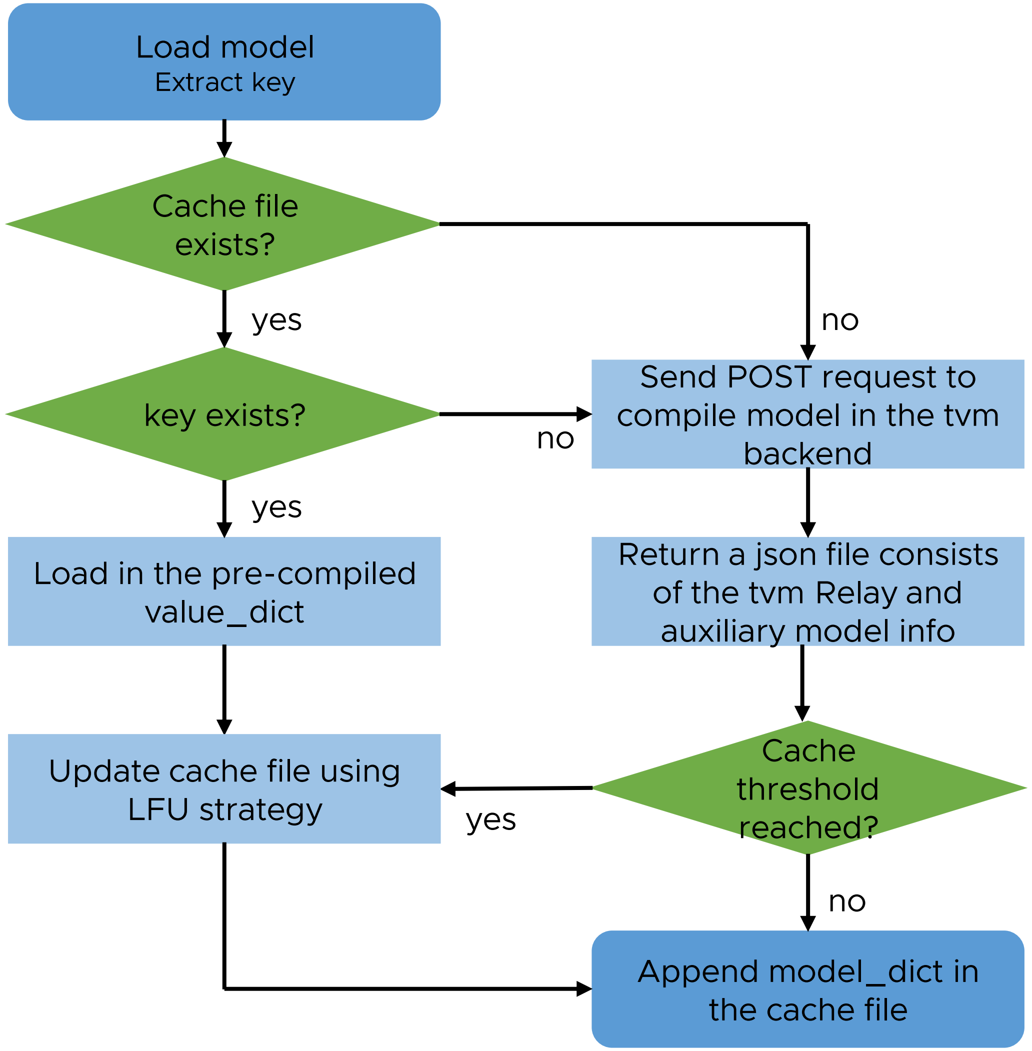
Project MLInferBooster

Simple mapping mechanism & workflow

Model_dict = {key : value_dict}

key : model_name#model_createtime

value_dict : { ‘tvm_relay’ : -----.so,
‘input_layer’ : -----,
‘input_layer_dtype’ : -----,
‘output_layer’ : -----,
‘target’ : -----,
‘freq’ : 1}


Load model
Extract key
Cache file exists?
yes
no
key exists?
no
yes
Send POST request to compile model in the tvm backend
Load in the pre-compiled value_dict
Return a json file consists of the tvm Relay and auxiliary model info
Update cache file using LFU strategy
yes
Cache threshold reached?
no
Append model_dict in the cache file

Project MLInferBooster

LFU cache replacement policy

Model is already compiled

key	value_dict					
	freq	Relay	target	input_layer	output_layer	input_layer_dtype
model_1#date	2	--	--	--	--	--
model_2#date	1	--	--	--	--	--
model_3#date	1	--	--	--	--	--

drop

append

freq = freq +1

model_1#date	3	--	--	--	--	--

A new model comes in when cache threshold has been reached

key	value_dict					
	freq	Relay	target	input_layer	output_layer	input_layer_dtype
model_2#date	1	--	--	--	--	--
model_3#date	1	--	--	--	--	--
model_1#date	3	--	--	--	--	--

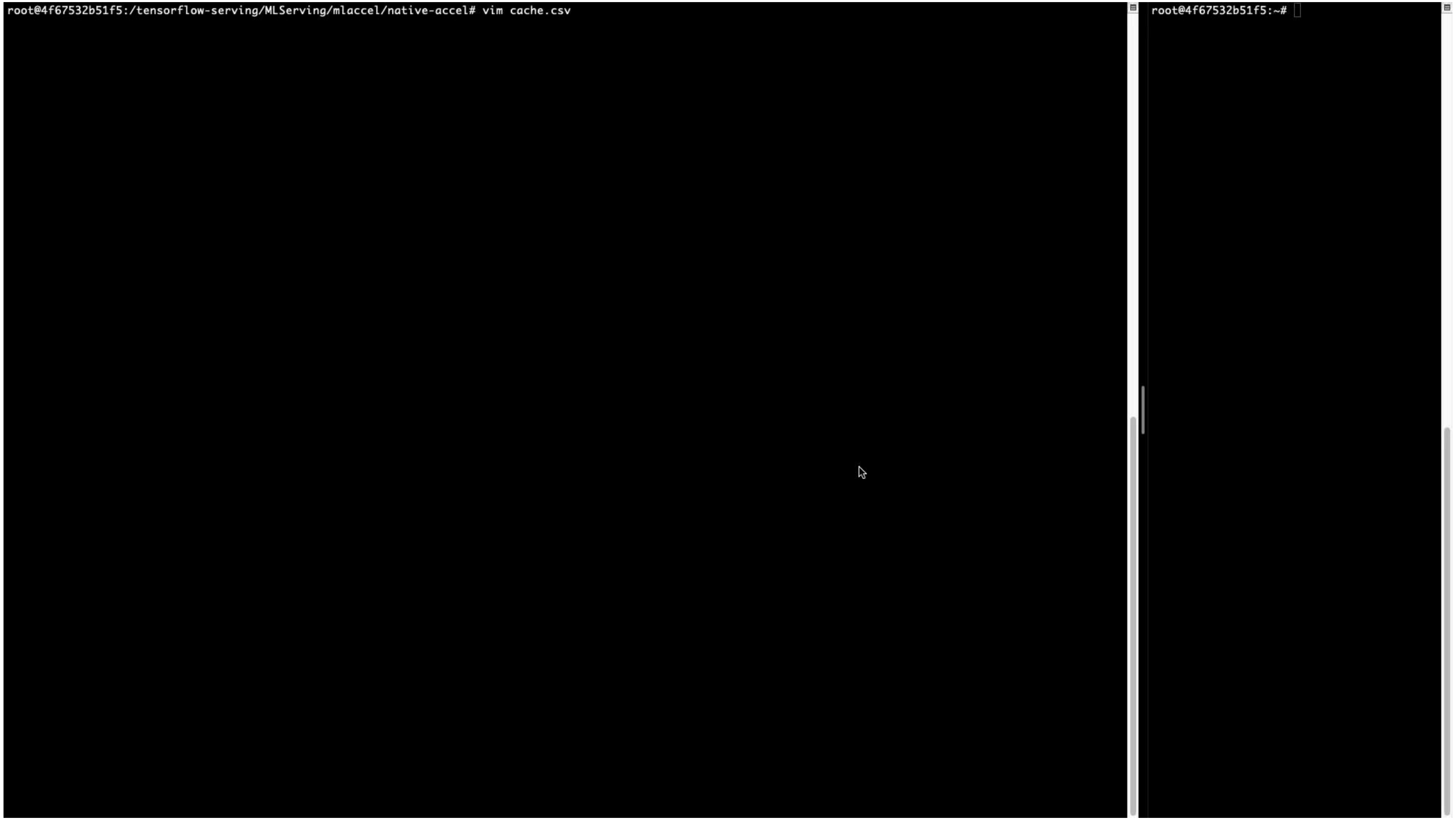
popitem (FIFO)

append

model_4#date	1	--	--	--	--	--

Project MLInferBooster

Demo Model Cache



```
root@4f67532b51f5:/tensorflow-serving/MLServing/mlaccel/native-accel# vim cache.csv
```

```
root@4f67532b51f5:~#
```

Project MLInferBooster

Next

- More ML upstream frameworks
- More ML serving system
- More ML models supported
  - ❑ Model conversion (TensorFlow/PyTorch → ONNX)
- K8s integration
  - KFServing

Open Source AceCon

2021 智能云边开源峰会

AI x Cloud Native x Edge Computing

人工智能 × 云原生 × 边缘计算

Thank You