ML-Compiler-Bridge: Interfacing ML and Compilers

S. VenkataKeerthy[1], **Siddharth Jain**[1],

Umesh Kalvakuntla[1], G Pranav Sai[1], Rajiv S Chitale[1], Eugene Brevdo[2], Albert Cohen[2], Mircea Trofin[2], Ramakrishna Upadrasta[1]

IIT Hyderabad[1], Google[2]

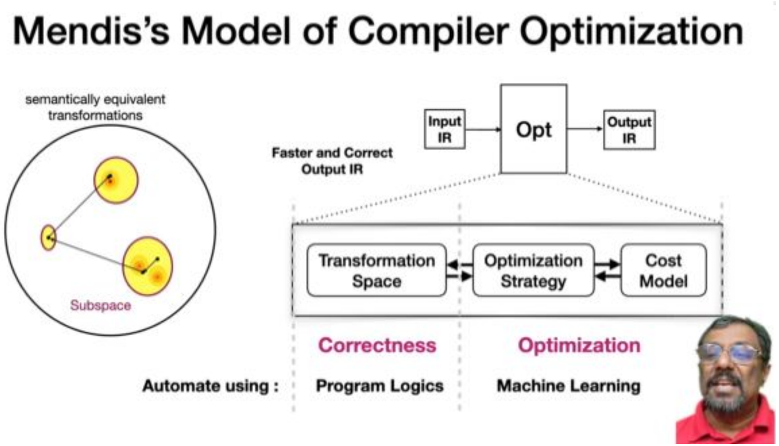
భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Eighth LLVM Performance Workshop at CGO
2nd March 2024

ML, ML everywhere!

- Impact of ML for *hard, heuristic-based* compiler optimizations

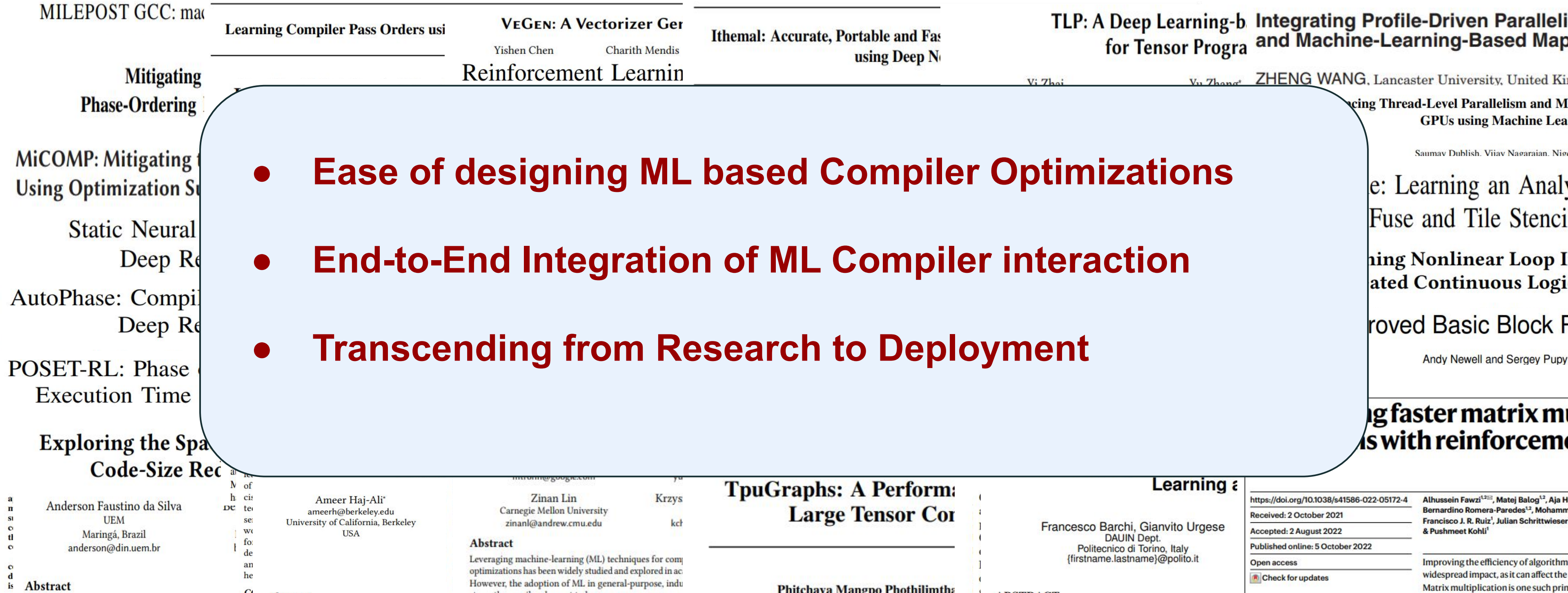
Compiler 2.0 (CGO’22 & LCTES’20 Keynotes) by Prof. Saman Amarasinghe



https://www.youtube.com/watch?v=w_sX9aZoZxg

ML, ML everywhere!

200+ works on using ML for Compiler Optimizations in the recent years!



- **Ease of designing ML based Compiler Optimizations**
- **End-to-End Integration of ML Compiler interaction**
- **Transcending from Research to Deployment**

ML for Compiler Optimizations

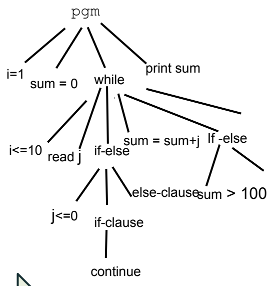
Benchmarks

Synthetic programs

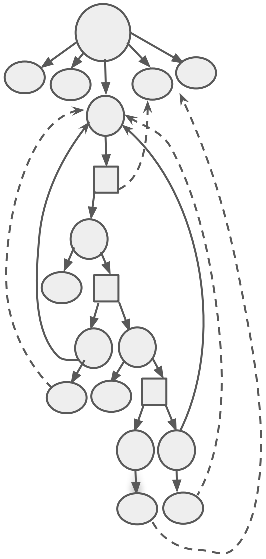
Program Generation

Fuzzers

Generate/Select Programs

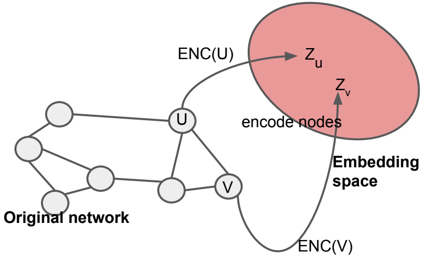



```
S1.  i =1
S2.  sum=0
P3.  while i<=10 do
S4.   read j
P5.   if j<=0 then
S6.      continue
S7.   sum=sum+j
P8.   if sum>100 then
S9.       break
S10.  i=i+1
S11. print sum
```



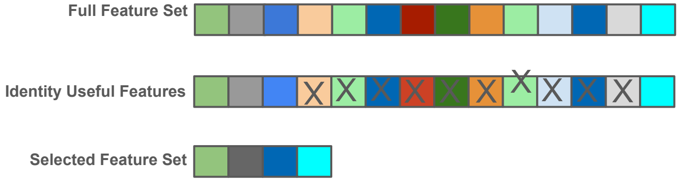
Choose Representations → AST/IR/PDG/…

Embeddings



or/and

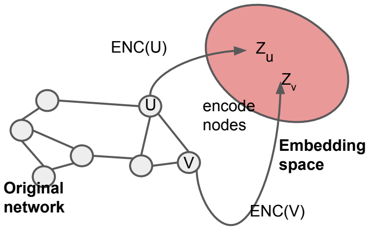
Feature Selection



Represent programs as vectors

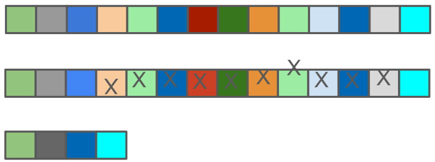
ML for Compiler Optimizations

Embeddings

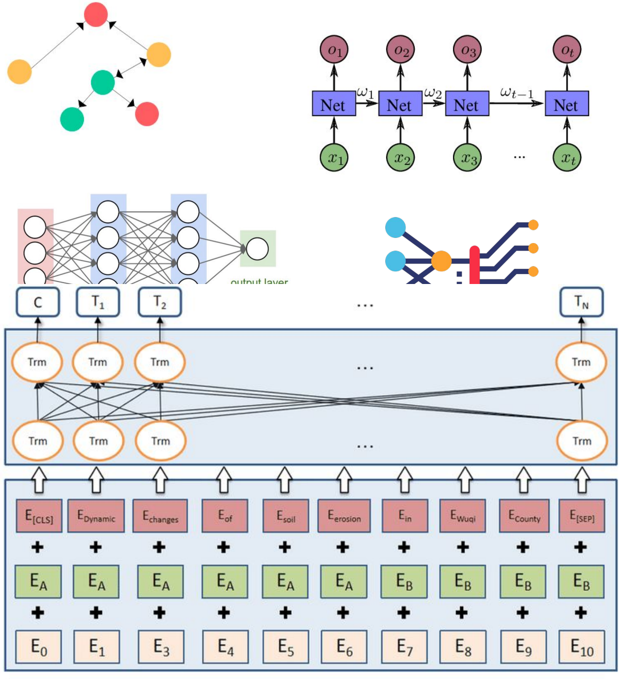


or/and

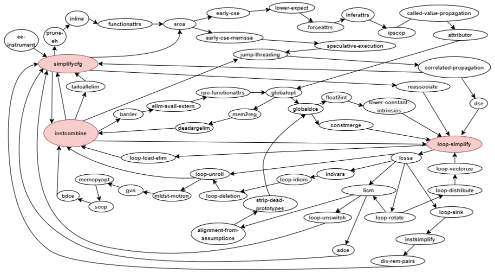
Feature Selection



Represent programs as vectors

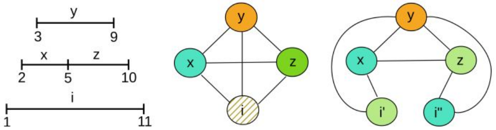


Choose ML model(s)



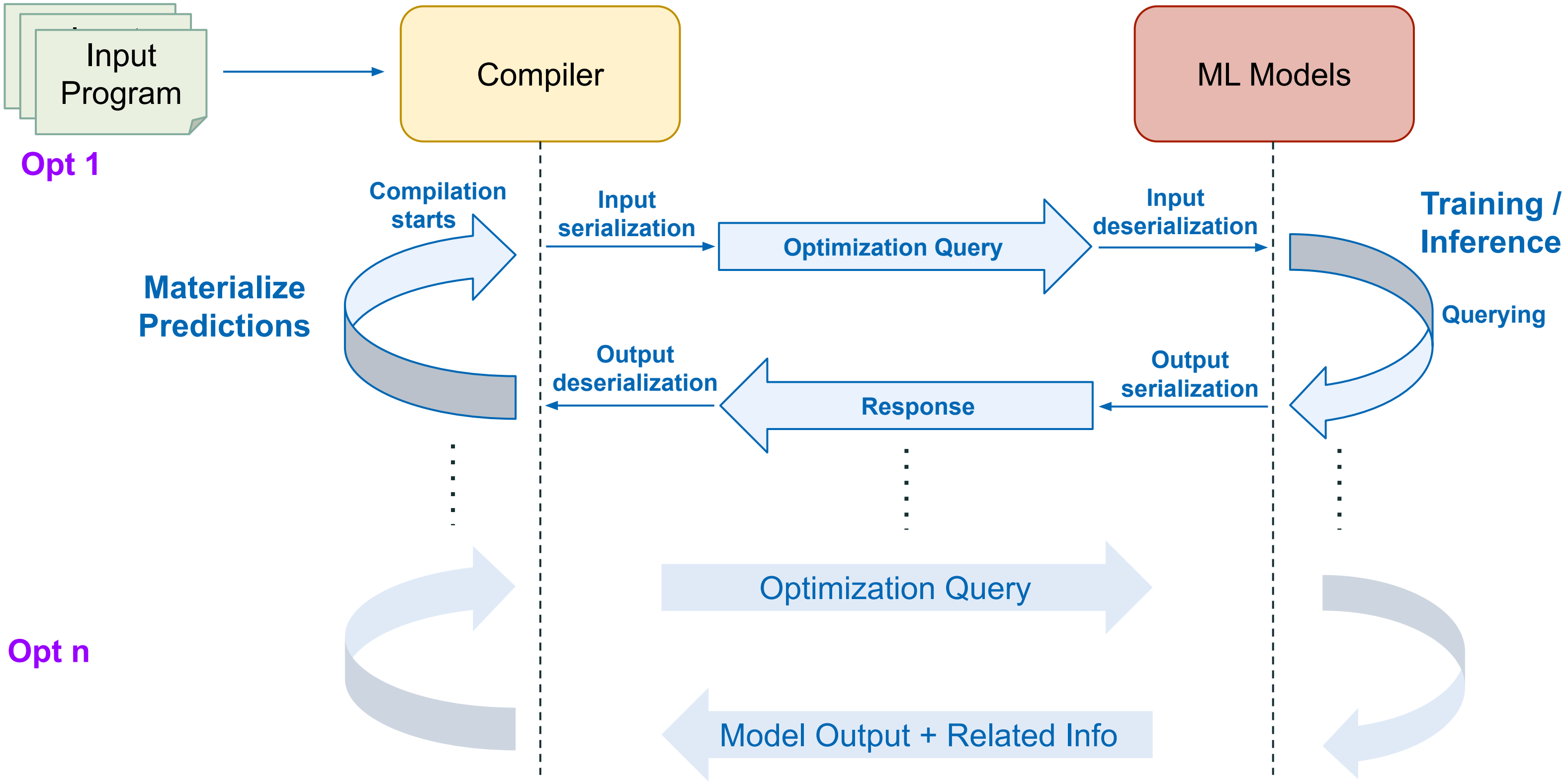
```
// (a) Loop1
int x[N] , y[N], a[N];
for (int i=1; i<N; ++i)
{
    x[i+1]=x[i-1]+x[i+1];
    a[i+1]=(a[i-1]+a[i])
            /2.0 ;
}
```

```
// (b) Loop1 :Distributed
int x[N],y[N],a[N];
for (int i=1; i<N; ++i)
   x[i+1] = x[i-1]+x[i+1];
for (int i=1; i<N; ++i)
   a[i+1] = (a[i-1]+a[i])
              /2.0;
```

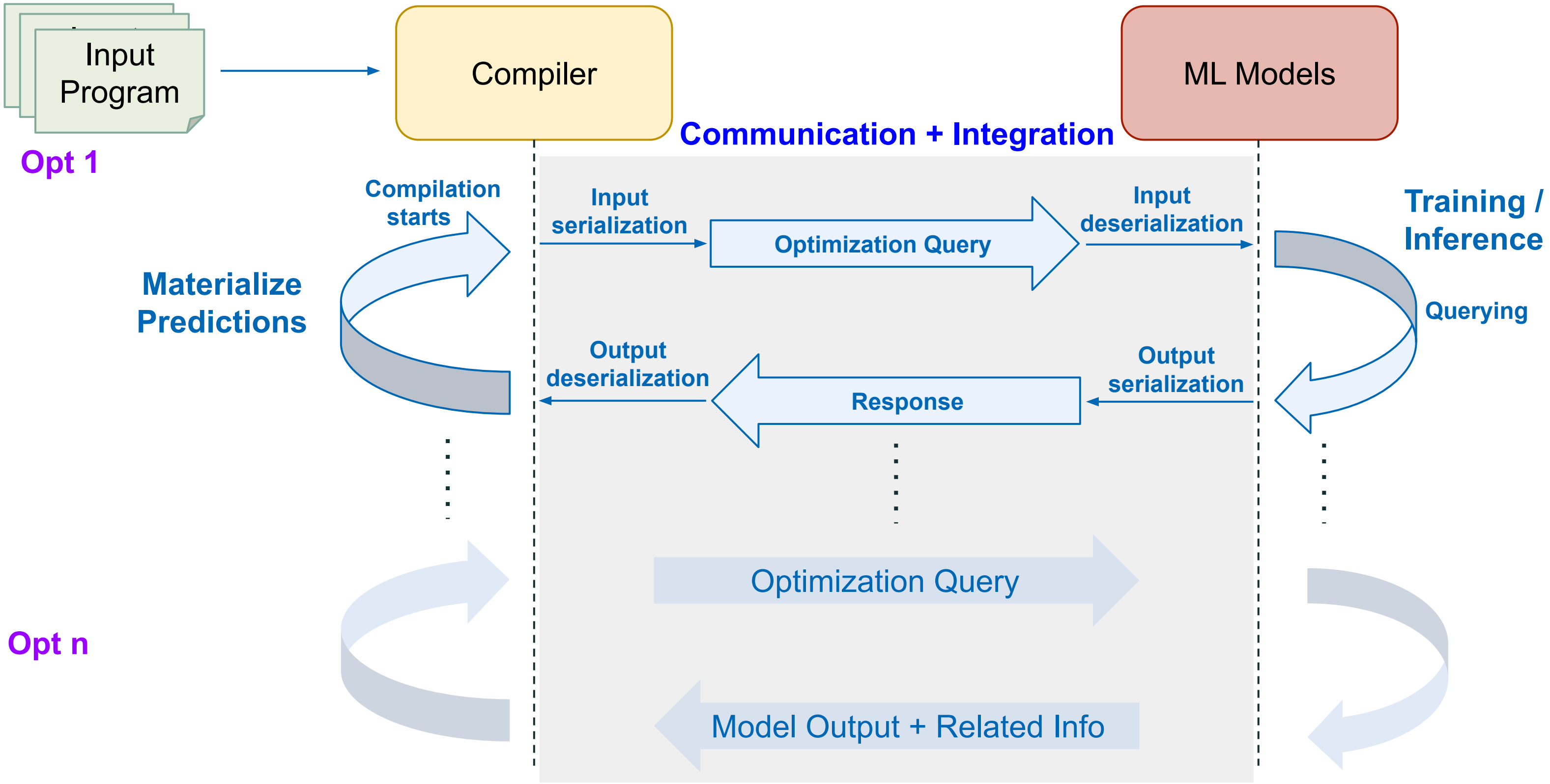


Perform Optimizations

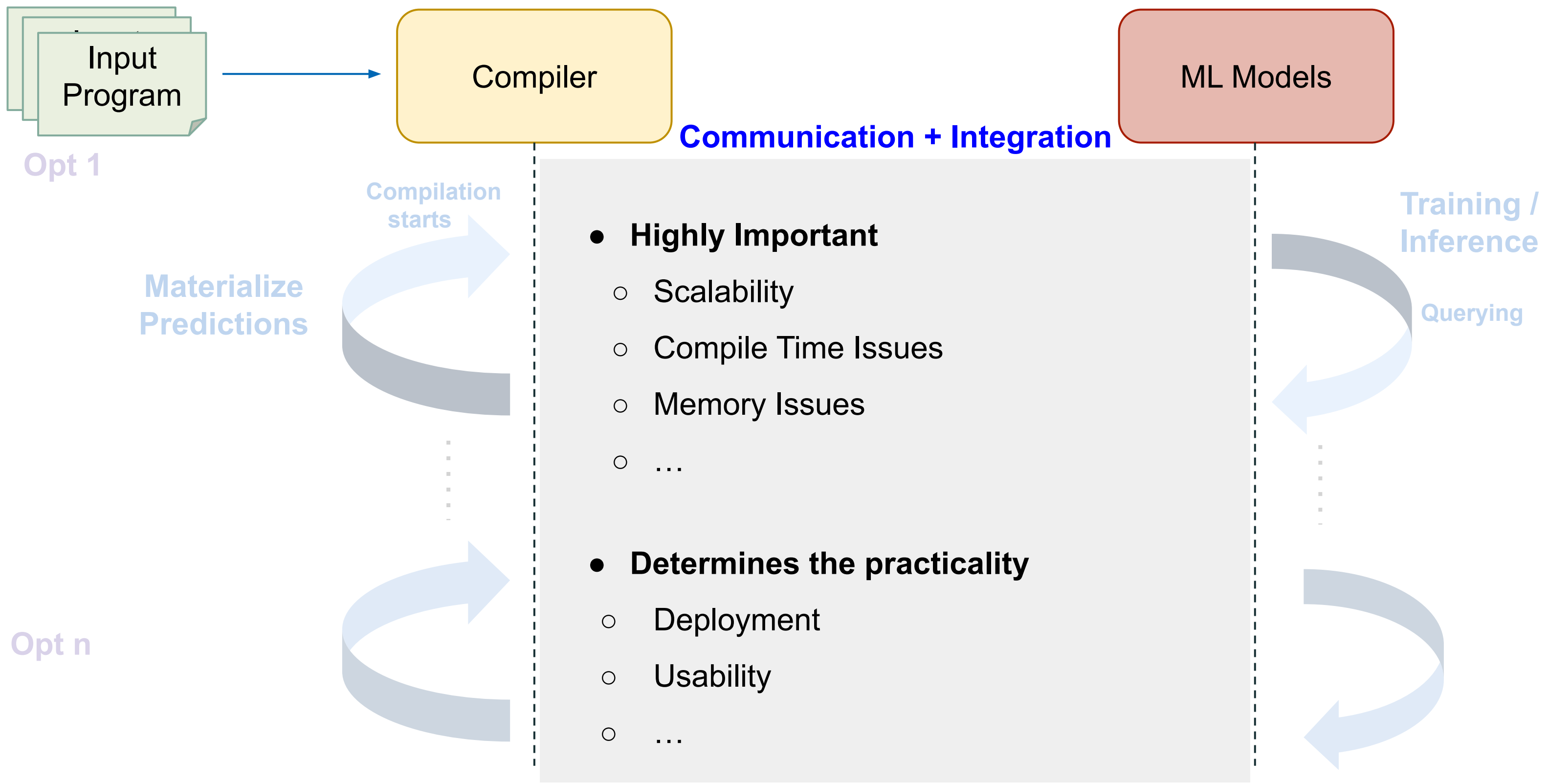
ML-Compiler Interaction



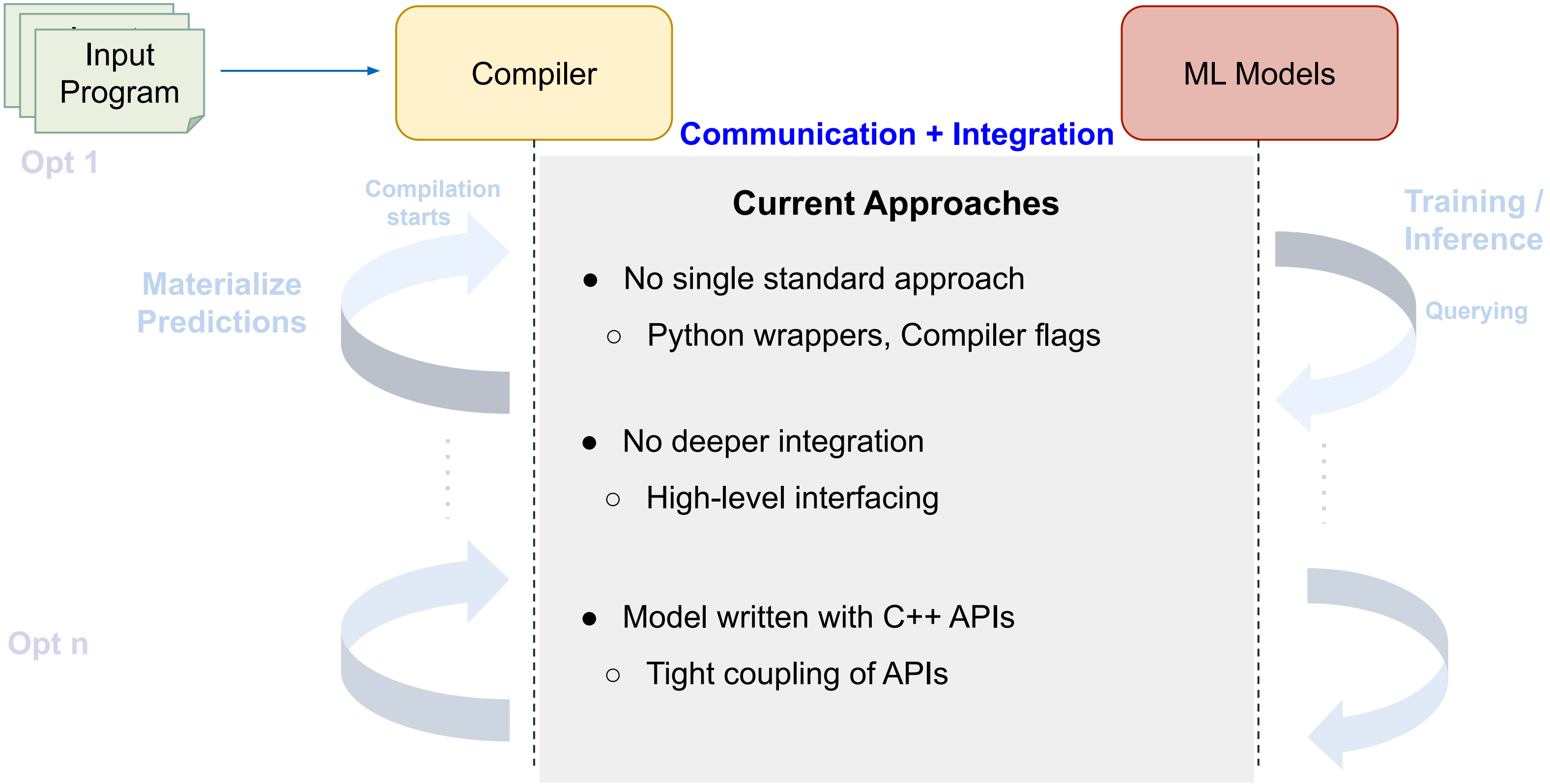
ML-Compiler Interaction



ML-Compiler Interaction



ML-Compiler Interaction



Current Approaches

- No single standard approach
  - Python wrappers, Compiler flags
- No deeper integration
  - High-level interfacing
- Model written with C++ APIs
  - Tight coupling of APIs

Current Limitations

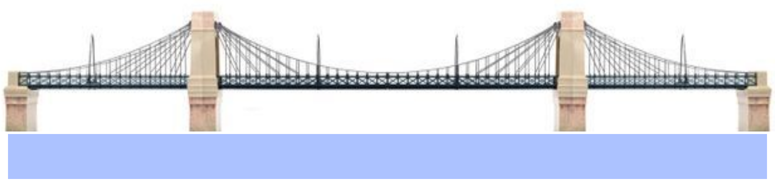
Scalability	Integration	Programmability	Portability
• Python/C++ wrappers • 6x – 100x slowdown	Not all outputs can be communicated via flags	Models written in C++ are not ML developer friendly	Support for diverse ML frameworks
Phase Ordering, Loop Distribution, …	**Register Allocation, Instruction Scheduling, …**	**RLLib, SciPy, …**	**TF, PyTorch, JAX, …**

Current Limitations

m x n problem 🙁

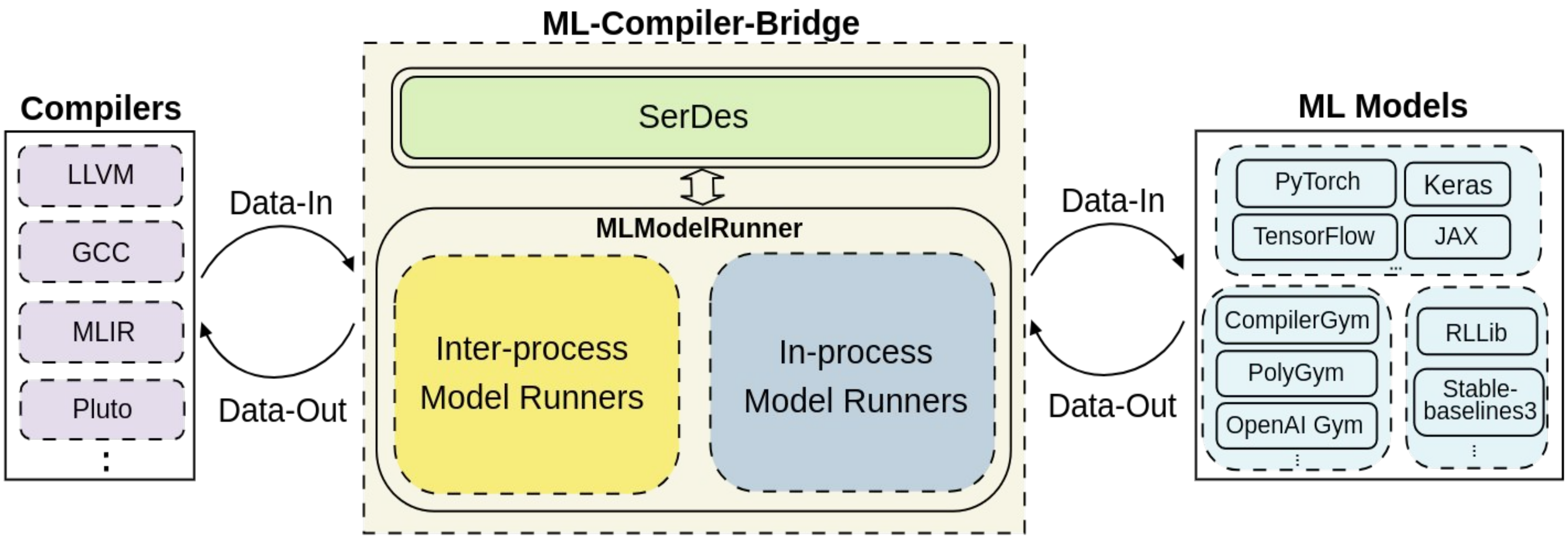
Our Proposal

ML-Compiler-Bridge

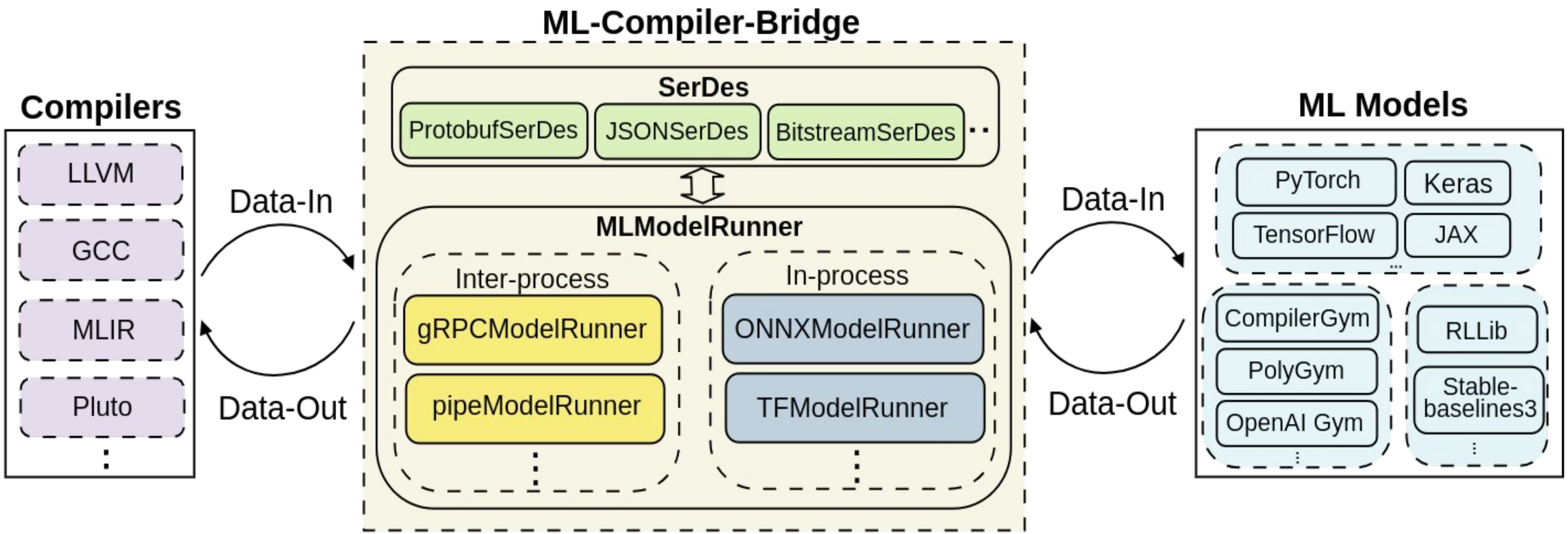


m + n 😀

ML-Compiler-Bridge



ML-Compiler-Bridge

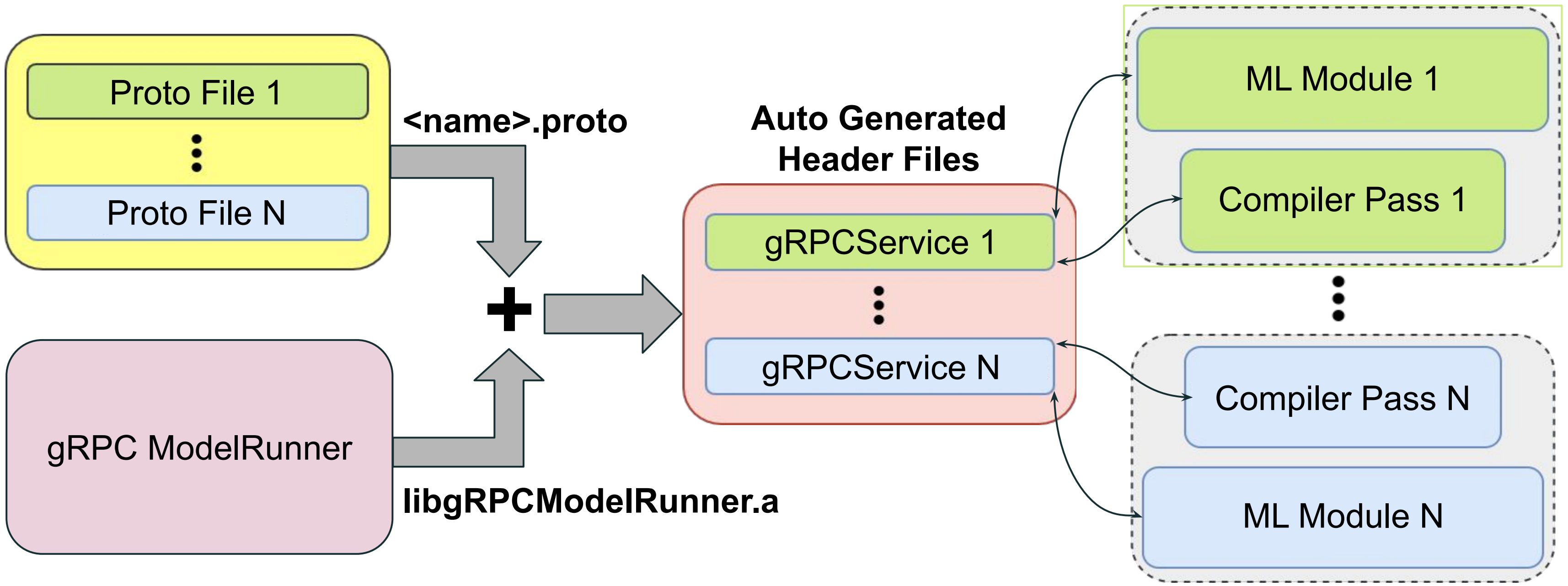


Model Runners: Medium of Communication

Two Broad Model Runners

Inter-Process Model Runners	In-Process Model Runners
Compiler and the ML model runs as two concurrent processes.	ML model is part of the compiler
• gRPC • Unix-style Named Pipes	• ONNX C++ Runtime • TensorFlow AOT model
Designed for **Training**	Designed for **Inference**

Inter-process Model Runners: gRPC



gRPC Model Runner - proto description

```
syntax = "proto3";
// Package name for current optimization
package helloMLBridgegRPC;
// Service class RPC declarations
service HelloMLBridgeService {
 // RPC to query compiler
 rpc queryCompiler(ActionRequest) returns (TensorResponse) {}
 // RPC to get Advice from model
 rpc getAdvice(TensorResponse) returns (ActionRequest) {}
}
// Data structures for request and response messages
message TensorResponse { repeated float tensor = 1; }
message ActionRequest { int32 action = 1; }
```

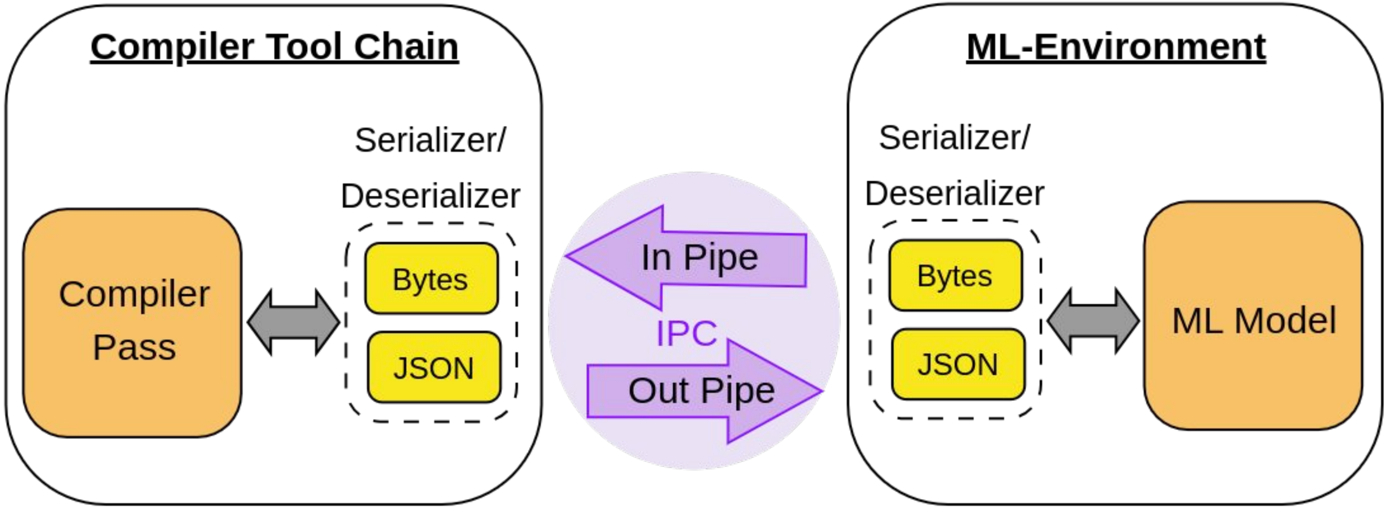
RPC Service Class

RPC to Query Compiler

RPC to Query Model

Request/Response Data Structure

Inter-process Model Runners: Pipes



Pipe Model Runner - Internals

```
...
void *PipeModelRunner::evaluateUntyped() override
 auto *data = SerDes->getSerializedData();
send(data);
 auto *reply = receive();

 return SerDes->deserializeUntyped(reply);
}
...
```

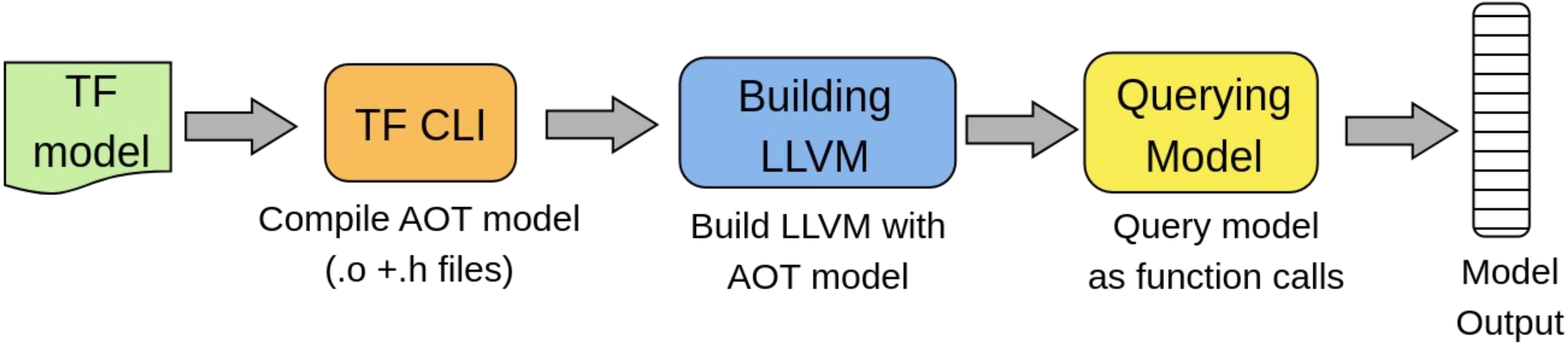
Send data on pipe

Get serialized data from SerDes

Read received data on pipe (**Blocking**)

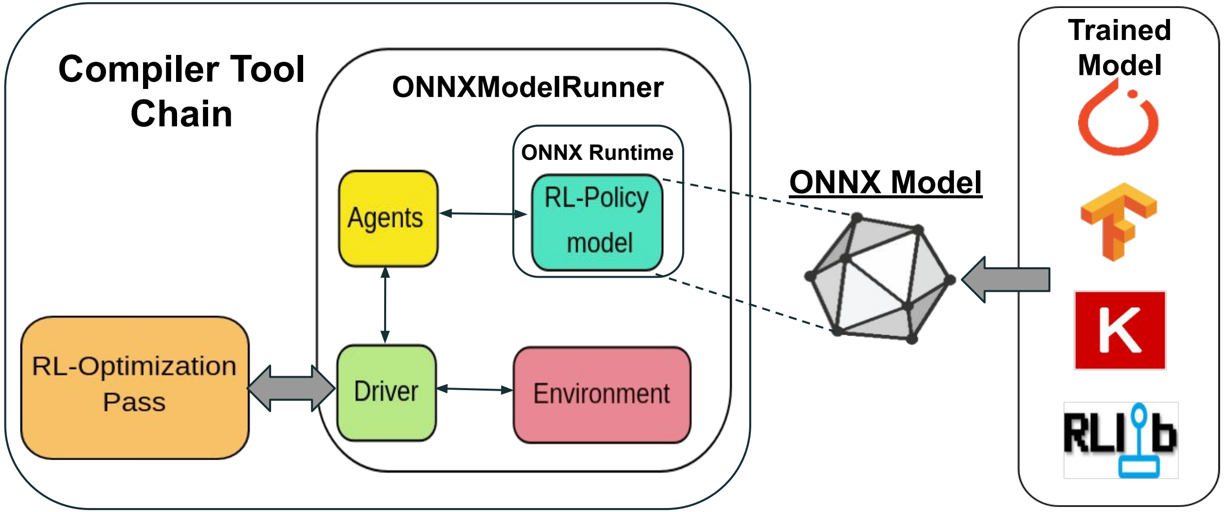
Deserialized received data and return

In-process Model Runners: TensorFlow AOT



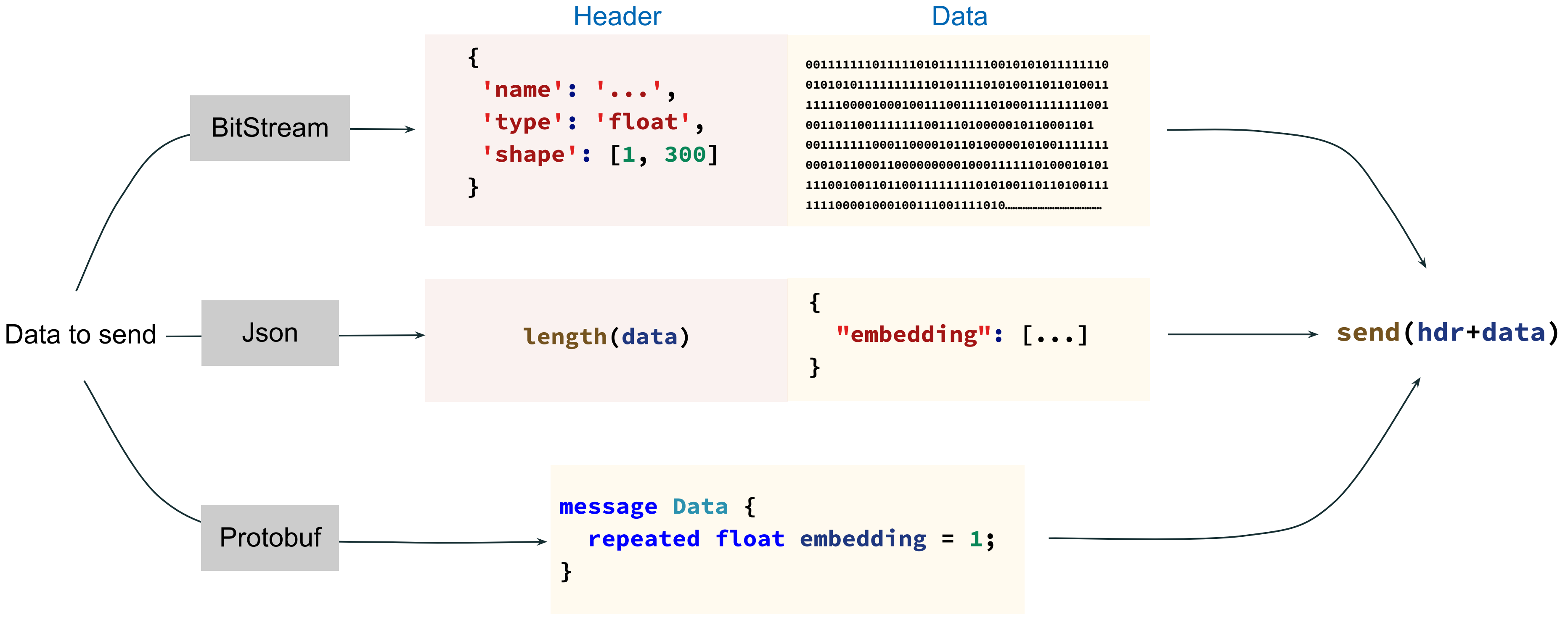
In-process Model Runners: ONNX

ONNX - Framework neutral, interoperable infrastructure for trained model integration



ONNX. Open Neural Network Exchange. 2017, https://github.com/onnx/onnx

SerDes: Serialization-Deserialization Module



Comparison of Different Model Runners

	gRPC	Pipes	ONNX	TF-AOT
Multithreaded Compilation	✖	✖	✔	✔
Distributed Training	✔	✖	–	–
Single process (Model is part of the compiler)	✖	✖	✔	✔
Auto-serialization	✔	✔	–	–
Communication Robustness	✖	✖	✔	✔
ML Framework Independent	✔	✔	✔	✖

Using ML-Compiler-Bridge (C++)

```cpp
#include "MLCompilerBridge/MLModelRunner.h"
#include "MLCompilerBridge/yourMLModelRunner.h"

// Instantiate the required model runner with SerDes type
MLModelRunner *MLRunner = std::make_unique<yourModelRunner>(Arg,
      SerDes::Kind::yourSerDesType);
// Process Input Features
std::pair<std::string, InType> p = ... // Input
MLRunner->populateFeatures(p);
// Get ML Advice/Output
OutType advice = MLRunner->evaluate<OutType>();
// Use the obtained advice
...
```

Creating the Model Runner Instance

Populating feature to be sent to Model

Querying Model for Advice

Multi-Language Support: Python

```python
import CompilerInterface as CI

# Instantiate the required CompilerInterface with serdes type
interface = CI.YourCompilerInterface(Arg, yourSerdesType)
while True:
    ...
    # Populates buffer with advice
    interface.populate_buffer(advice)
    # Send buffer data to compiler and wait for next request
    response = interface.evaluate()
    ...
    # Break on condition
```

Creating CompilerInterface Instance

Populating buffer with advice data

Responding to compiler with advice

Multi-Language Support: C

```c
#include "MLModelRunner/C/ONNXModelRunner.h"
#include "MLModelRunner/C/PipeModelRunner.h"

// Instantiate the required model runner with SerDes type
PipeModelRunnerWrapper *pmr = createPipeModelRunner
      ("plutopipe.out", "plutopipe.in", config);
// Process Input Features
float *features = ... // Input
populateFloatFeatures(pmr, "tensor", features, n);
// Get ML Advice/Output
int advice = evaluateIntFeatures(pmr);
// Use the obtained advice
...
```

Creating Pipe Model Runner Instance

Populating feature to be sent to Model

Querying Model for Advice

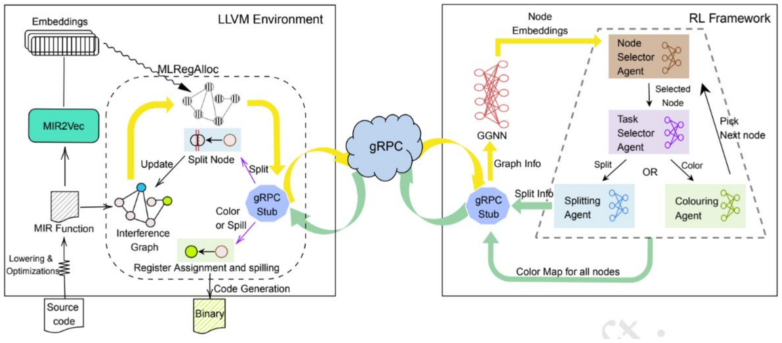
Adding New Model Runners + SerDes

```cpp
#include "MLModelRunner/MLModelRunner.h"

class NewModelRunner : public
MLModelRunner {
public:
 // Custom ModelRunner Constructor
 NewModelRunner();
 virtual ~NewModelRunner();
private:
 // Function to establish communication
 void *evaluateUntyped() override;
 // Functions to send and receive data
 void send(void *data);
 void *receive();
};
```

```cpp
#include "SerDes/baseSerDes.h"

class NewSerDes : public BaseSerDes {
public:
 NewSerDes() :BaseSerDes(BaseSerDes::Kind::NewSD){};
 void setFeature(const std::string name, const int value)
override;
 void setFeature(const std::string name, const float value)
override;
                              ...
 void *getSerializedData() override;
 void cleanDataStructures() override;
 private:
 void *deserializeUntyped(void *data) override;
};
```

Supports Wider Use-Cases...

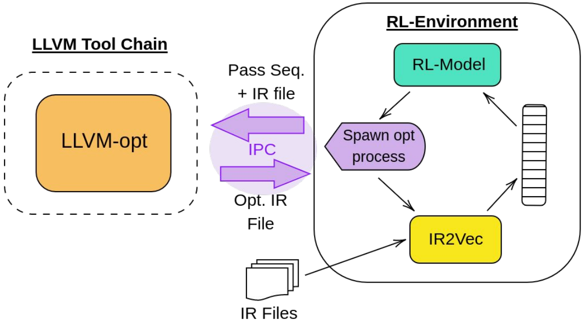
RL4ReAl - Register Allocation



- Communication: gRPC based multiple times
- Agents: Multiple hierarchical agents
- Model Type: PyTorch (GNN + FCNN)
- Model Input: Interference graph + node embedding
- Model Output: Colour map

VenkataKeerthy, et al., RL4ReAl: Reinforcement Learning for Register Allocation. CC 2023.

POSET-RL - Phase Ordering

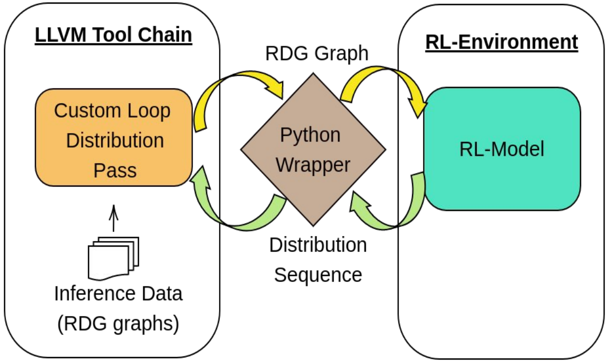


- Communication: Opt flag based multiple times
- Agent: Single agent
- Model Type: PyTorch (FCNN)
- Model Input: IR2Vec vectors
- Model Output: Pass sequence

Jain, et al., POSET-RL: Phase ordering for Optimizing Size and Execution Time using Reinforcement Learning. ISPASS 2022

Supports Wider Use-Cases…

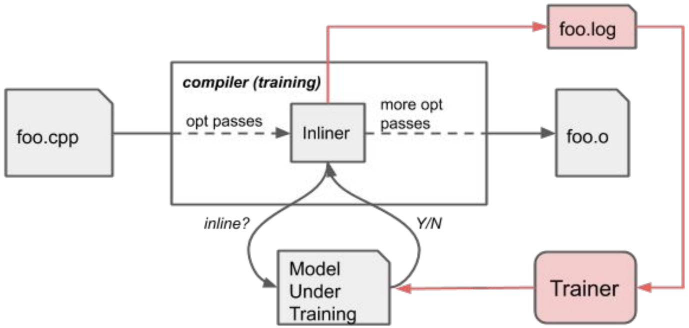
Loop Distribution



- Communication: Python Wrapper based once at end
- Agents: Multiple agents
- Mode Type: PyTorch (GNN + FCNN)
- Model Input: IR2Vec vectors
- Model Output: Distribution sequence

Jain, et al., "Reinforcement Learning assisted Loop Distribution for Locality and Vectorization", LLVM-HPC 2022.

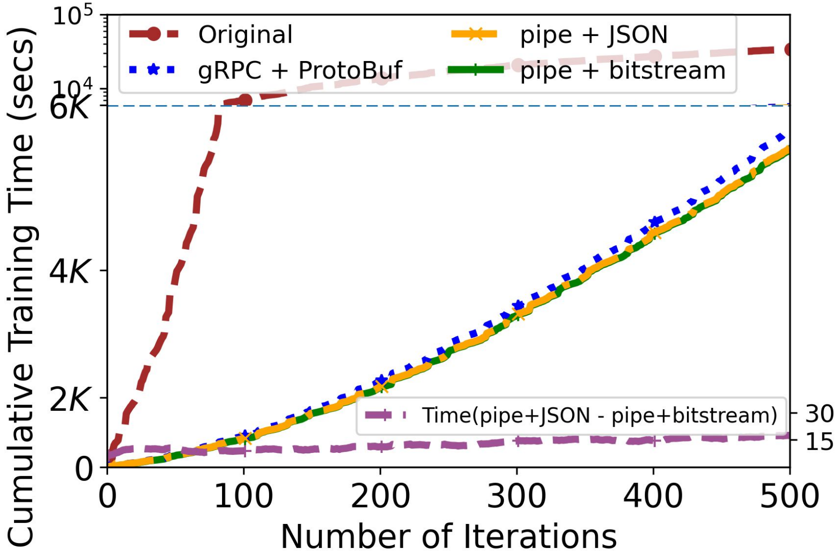
LLVM ML-Inliner



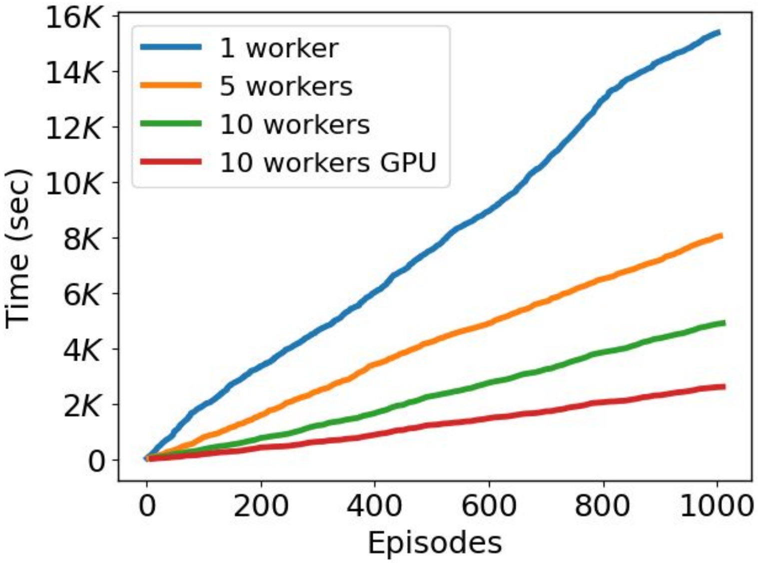
- Communication: Precompiled TF model
- Agents: Single agent
- Mode Type: TensorFlow (FCNN)
- Model Input: Feature vector
- Model Output: Binary (yes/no)

Trofin, et al. "MLGO: a machine learning guided compiler optimizations framework." arXiv 2021.

Training Time Improvements

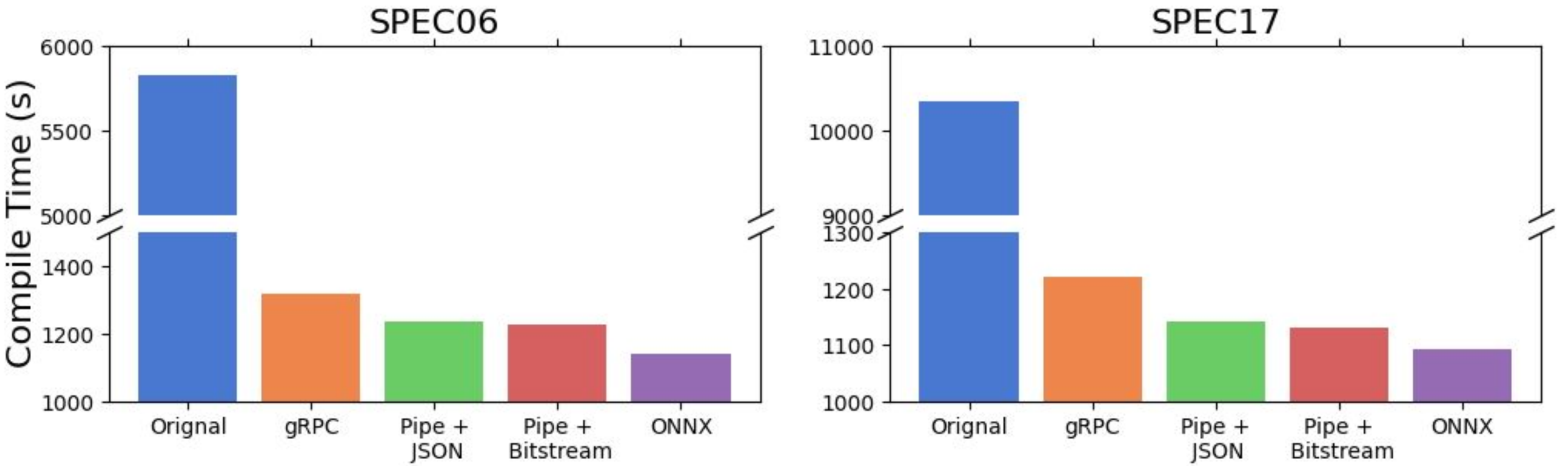


POSET-RL Training
Time Comparison

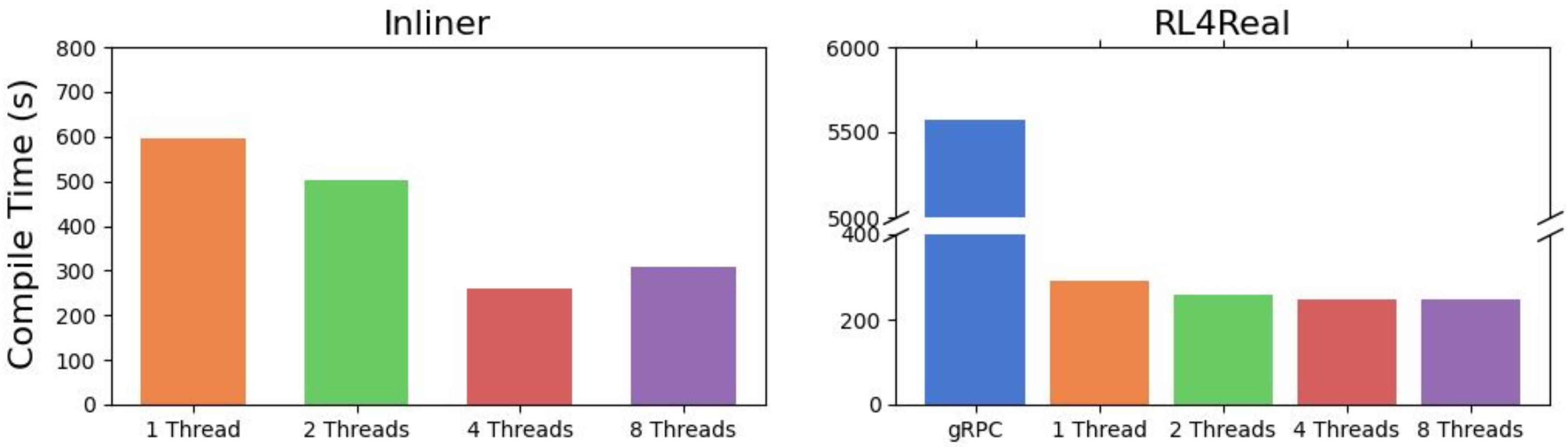


RL4ReAl Multi-worker
Training Time Comparison

Compile (Inference) Time Improvements: POSET-RL

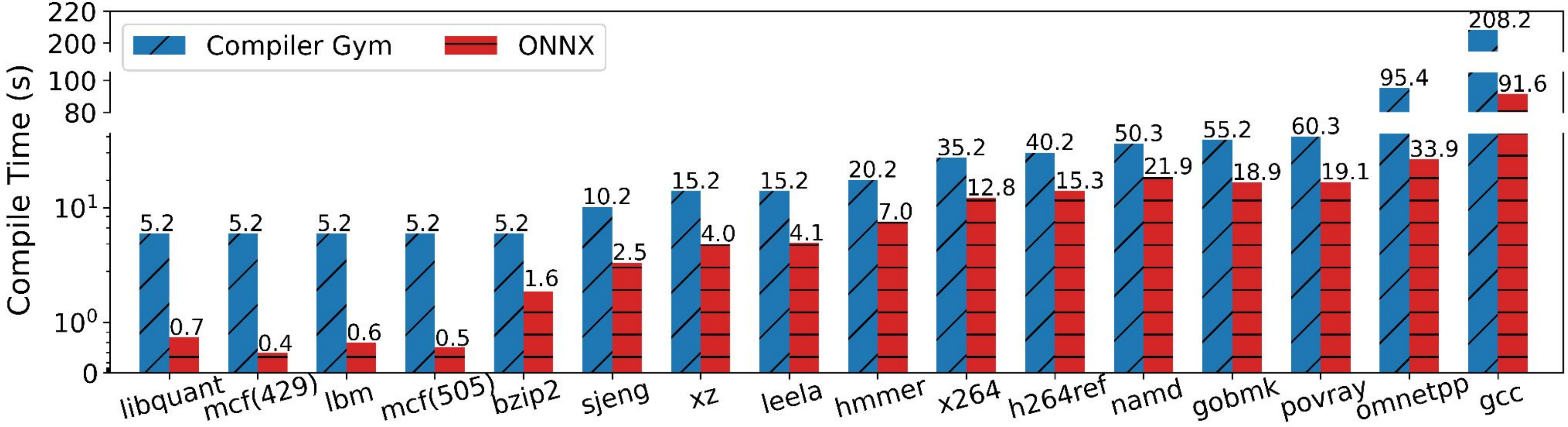


Support for Multi-threaded Compilation



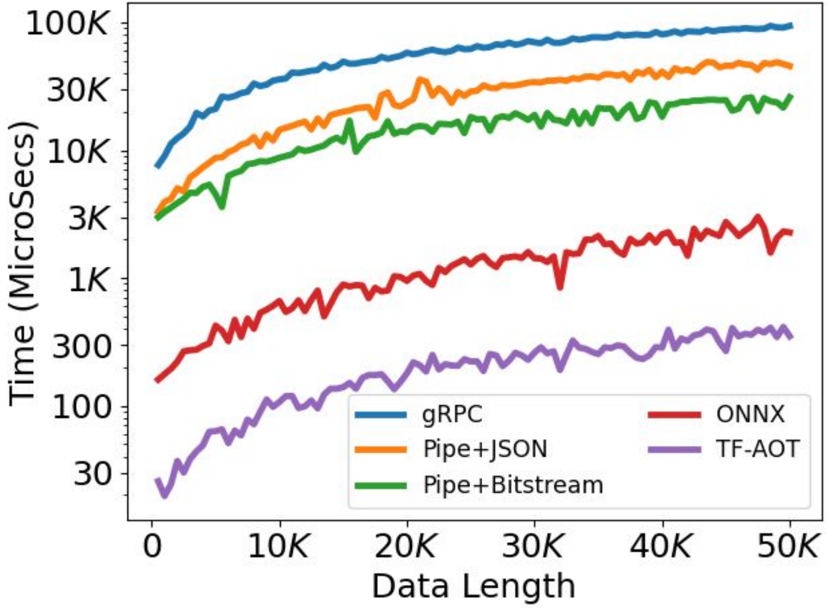
ML-Compiler-Bridge with CompilerGym

- Inference time comparison with CompilerGym's phase ordering model
- Model exported as ONNX model and queried using ONNXModelRunner



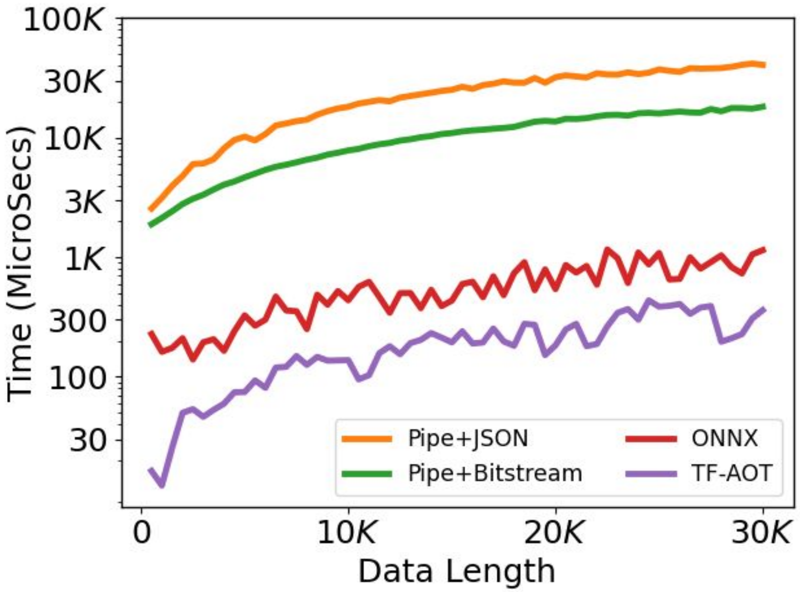
Cummins, et al. "CompilerGym: Robust, Performant Compiler Optimization Environments for AI Research." CGO 2022.

Performance of Individual Model Runners

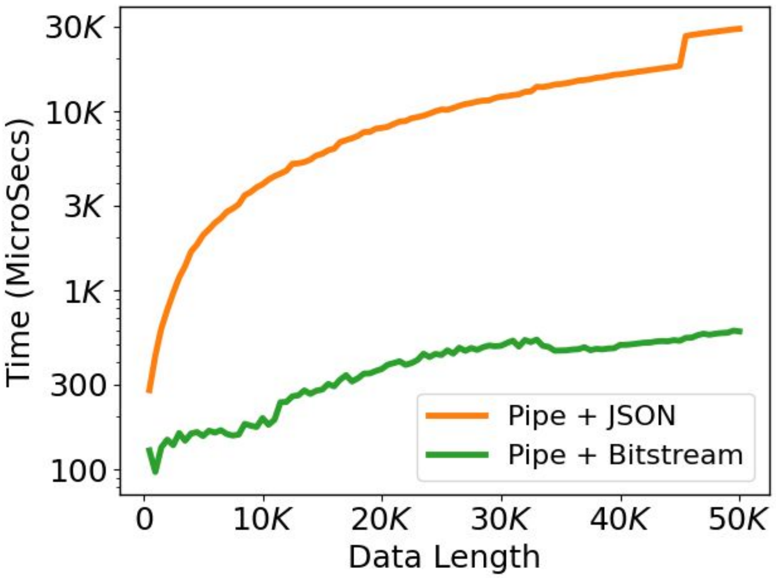


Round Trip-Times (RTT) for querying model with data of different lengths

Support for MLIR & Pluto



MLIR



Pluto

RTT for querying model with data of different lengths

Summary - ML-Compiler-Bridge

- Scalable, Lightweight suite of model runners and serializers
  - Supports Multiple Languages
  - Compiler and ML-Framework Independent
  - Supports deeper and high-level interfacing with compilers
- Plug-and-Play approach for ML based Compiler Optimizations
- Easier transition from research to deployment
- <u>We plan to upstream relevant portions to LLVM in addition to what is available</u>

Thank You!

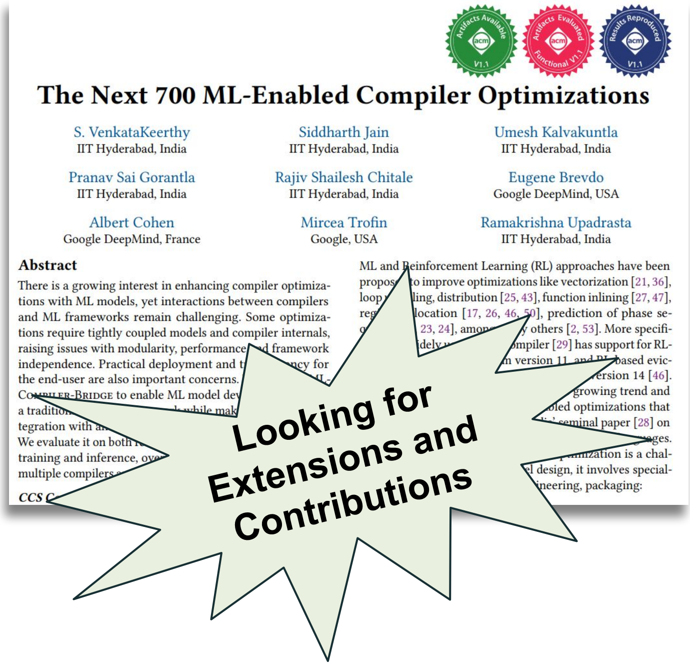
S. VenkataKeerthy | Siddharth Jain

https://svkeerthy.github.io | https://sid18996.github.io

Interested? Please get in touch with us

Visit our Poster @ C4ML (1800 hrs, Reception Area)

Code


The Next 700 ML-Enabled Compiler Optimizations

S. VenkataKeerthy, IIT Hyderabad, India
Siddharth Jain, IIT Hyderabad, India
Umesh Kalvakuntla, IIT Hyderabad, India
Pranav Sai Gorantla, IIT Hyderabad, India
Rajiv Shailesh Chitale, IIT Hyderabad, India
Eugene Brevdo, Google DeepMind, USA
Albert Cohen, Google DeepMind, France
Mircea Trofin, Google, USA
Ramakrishna Upadrasta, IIT Hyderabad, India

Abstract

There is a growing interest in enhancing compiler optimizations with ML models, yet interactions between compilers and ML frameworks remain challenging. Some optimizations require tightly coupled models and compiler internals, raising issues with modularity, performance … framework independence. Practical deployment and t… the end-user are also important concerns. …

https://compilers.cse.iith.ac.in/research/mlcompilerbridge