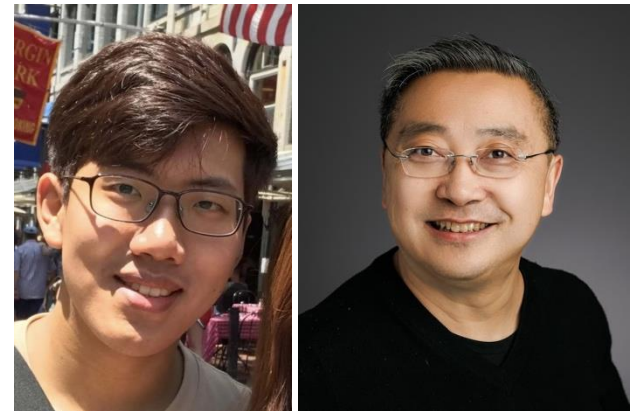


Serve Programs, Not Prompts

In Gim and Lin Zhong
Yale University

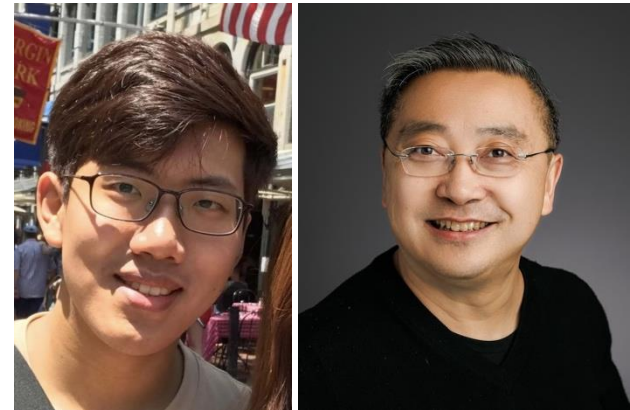


Future LLM serving systems should

Serve Programs, Not Prompts

To efficiently support emerging AI applications

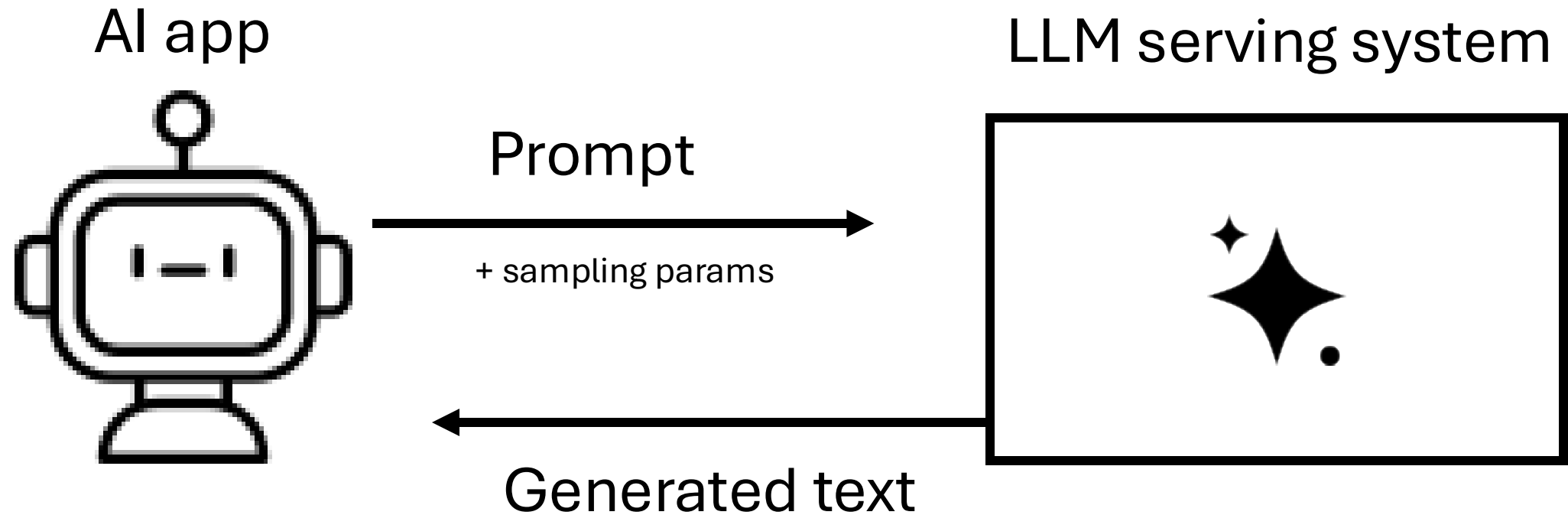
In Gim and Lin Zhong
Yale University



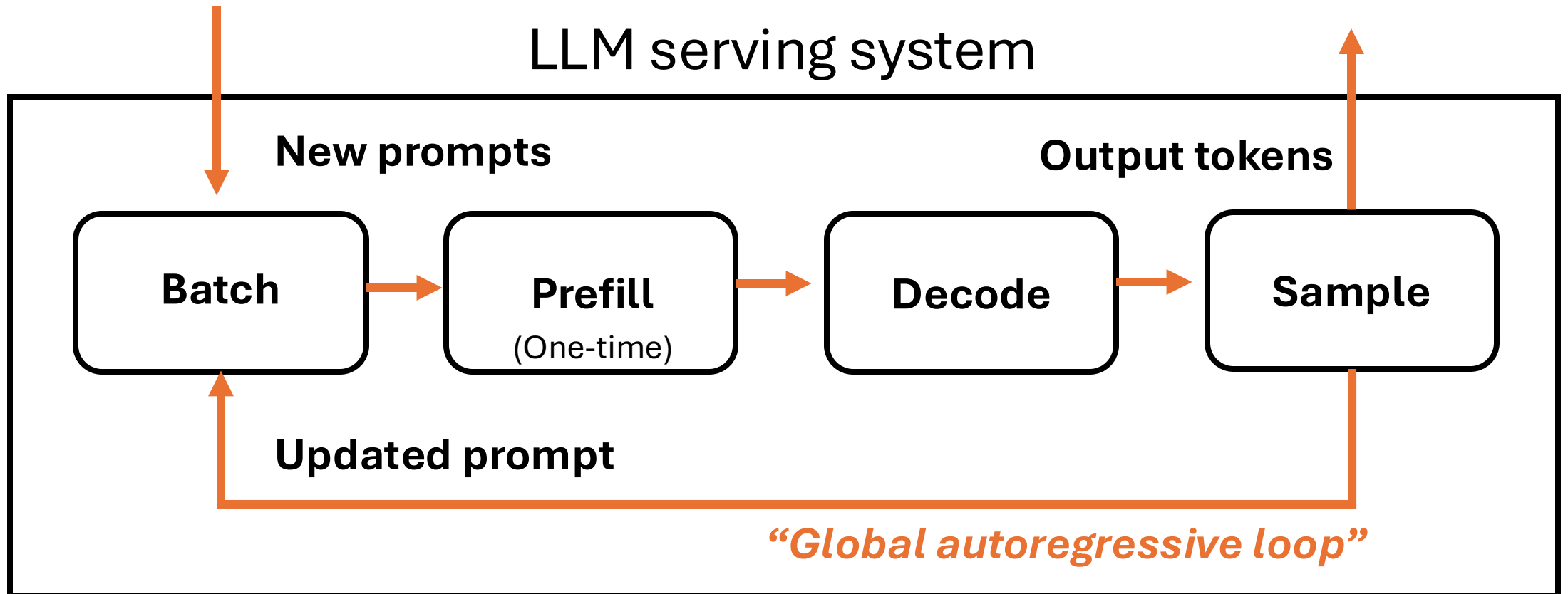
The only API you need to build a million-dollar startup

```
curl https://api.openai.com/v1/responses \  
-d "Write a one-sentence bedtime story about a unicorn."
```

Today's LLM serving systems are designed for text completion

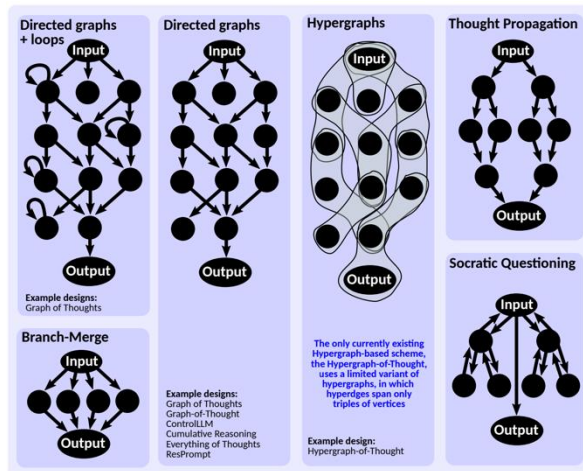


Today's LLM serving systems are designed for text completion



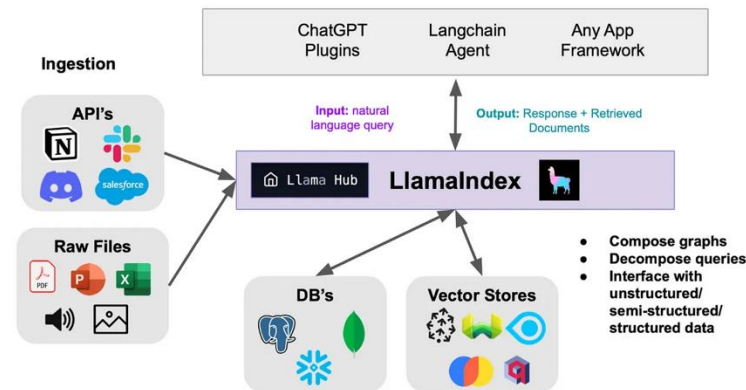
LLM usages are evolving beyond simple text completion

Non-linear generation



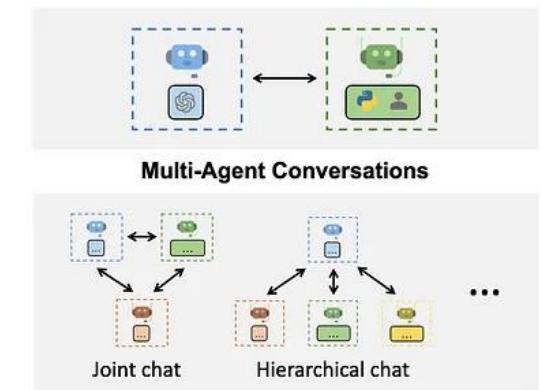
E.g., Graph-of-Thought

Augmented generation



E.g., LlamaIndex

Agentic problem solving



Flexible Conversation Patterns

E.g., AutoGen

LLM usages are evolving beyond simple text completion

Non-linear generation

Augmented

Language Model Cascades:
Token-level uncertainty and beyond

Type-Constrained Code Generation with Language Models

NIELS MÜNDLER*, ETH Zurich, Switzerland

JINGXUAN HE*, UC Berkeley, USA

HAO WANG, UC Berkeley, USA

KOUSHIK SEN, UC Berkeley, USA

DAWN SONG, UC Berkeley

MARTIN VECHEV, ETH Z

Aditya Krishna Narasimhan

Wittawat Jitkrittum

Aditya Krishna Menon

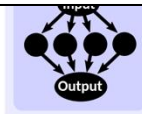
Sanjiv Kumar

Google Research, New York

Multi-Agent Conversations

VALIDATING LARGE LANGUAGE MODELS WITH RELM

Michael Kuchnik¹ Virginia Smith¹ George Amvrosiadis¹

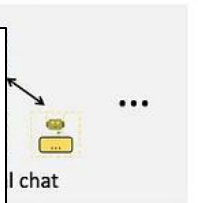


Example designs:
Graph of Thoughts
Graph-of-Thought
ControlLLM
Cumulative Reasoning
Everything of Thoughts
ResPrompt

the hypergraph of thoughts uses a limited variant of hypergraphs, in which hyperedges span only triples of vertices

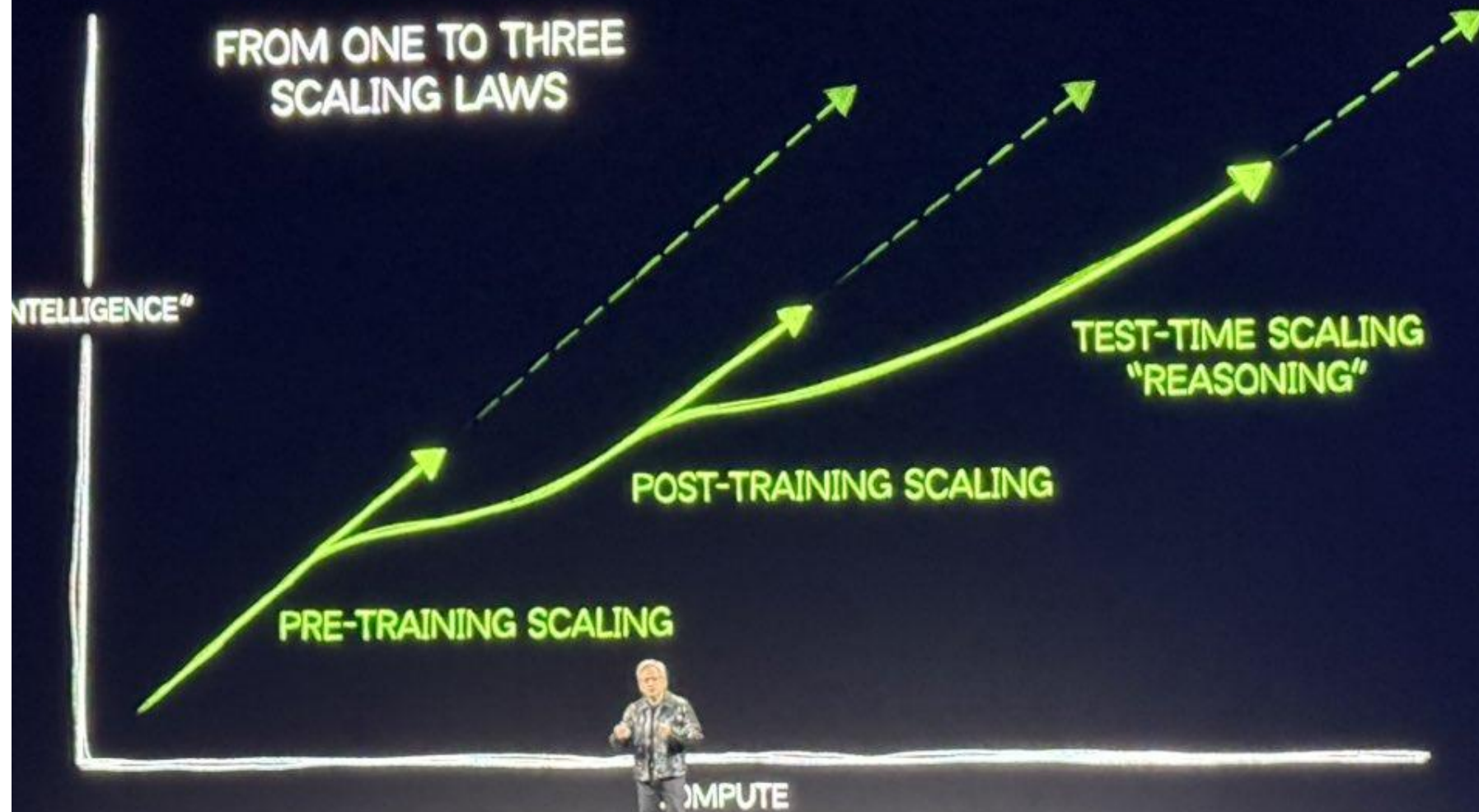
Example design:
Hypergraph-of-Thought

E.g., Graph-of-Thou



Patterns

Gen



The disconnect between LLM applications and serving systems

1. **Inference** inefficiency from missing application-level optimizations
2. **Integration** friction with the external data and tools
3. **Implementation** challenges for the custom generation workflows

The only API you need to build a million-dollar startup?

```
curl https://api.openai.com/v1/responses \  
-d "Write a one-sentence bedtime story about a unicorn."
```

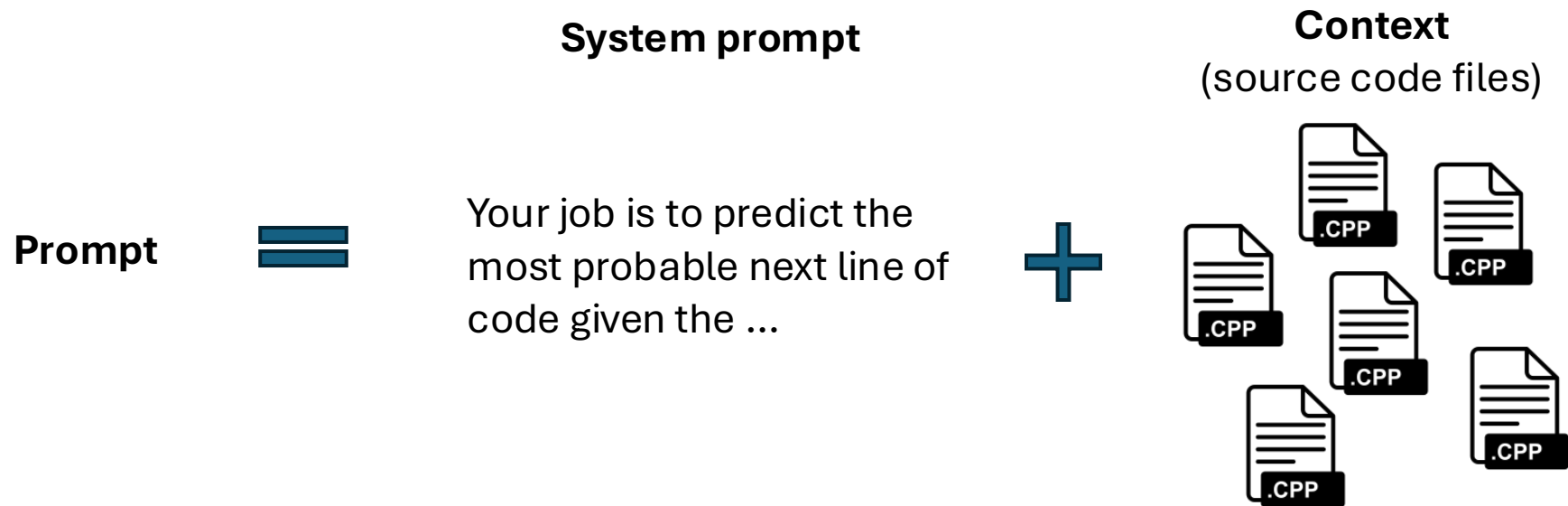
Imagine building an AI code completion tool

```
1  from __future__ import print_function
2  import argparse
3  import torch
4  import torch.nn as nn
5  import torch.optim as optim
6  import numpy as np
7  import matplotlib
8  matplotlib.use('Agg')
9  import matplotlib.pyplot as plt
10
11 class Sequence(nn.Module):
    def __init__(self):
        super(Sequence, self).__init__()
        self.lstm1 = nn.LSTMCell(1, 51)
        self.lstm2 = nn.LSTMCell(51, 51)
        self.linear = nn.Linear(51, 1)

    def forward(self, input, future = 0):
        outputs = []
        h_t = torch.zeros(input.size(0), 51, dtype=torch.double)
```

Challenge 1. Inference efficiency

Scenario: User's input updates the suggestion.

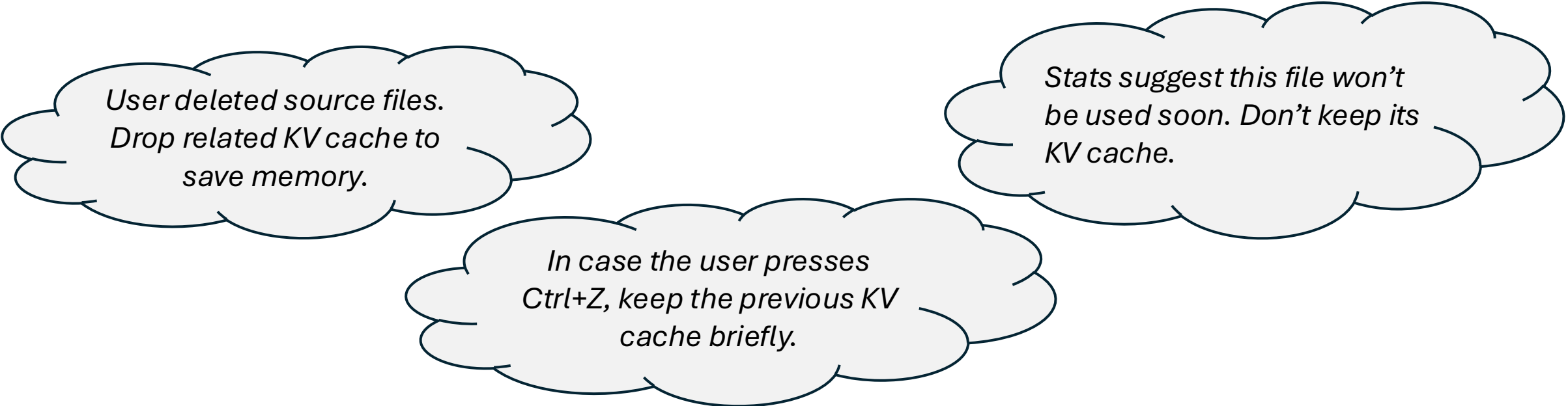


Due to the large overlaps in prompts,
Inter-request KV cache reuse is essential for efficiency

Challenge 1. Inference efficiency

KV cache management is often automatic, system-wide policy
(e.g., OpenAI's prompt caching, vLLM's automatic prefix caching)

This precludes application-driven optimizations



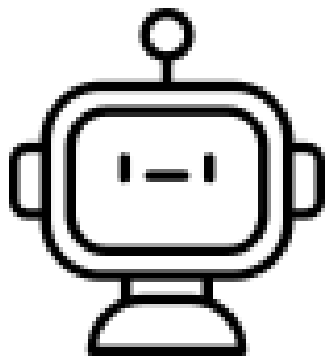
*User deleted source files.
Drop related KV cache to
save memory.*

*In case the user presses
Ctrl+Z, keep the previous KV
cache briefly.*

*Stats suggest this file won't
be used soon. Don't keep its
KV cache.*

Challenge 2. Integration efficiency

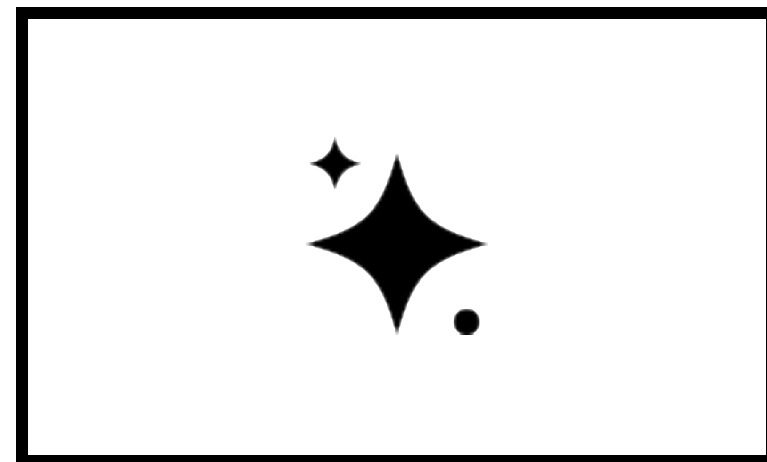
Scenario: Augment generation with relevant API documentation



Prompt: “If unsure about an API during generation, output **<read>API name</read>** and pause”



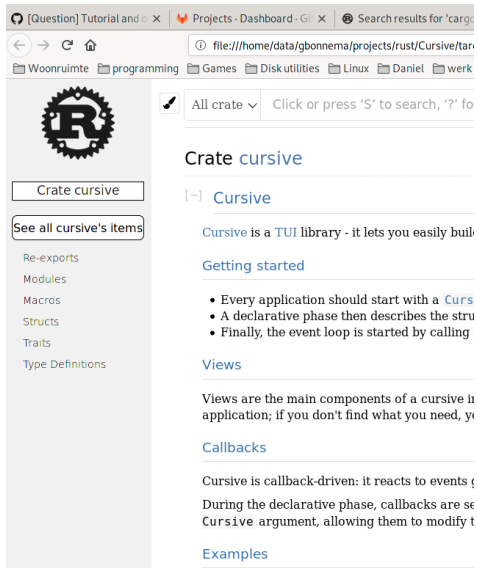
LLM serving system



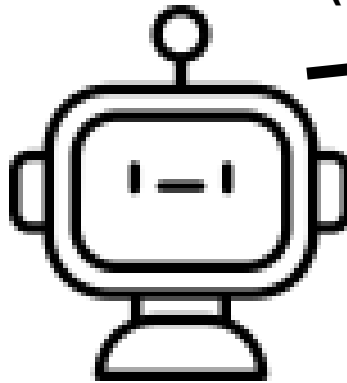
Challenge 2. Integration efficiency

Each external interaction induces two extra boundary crossings

Data/Tool



(3) Retrieve data



(1) Original prompt



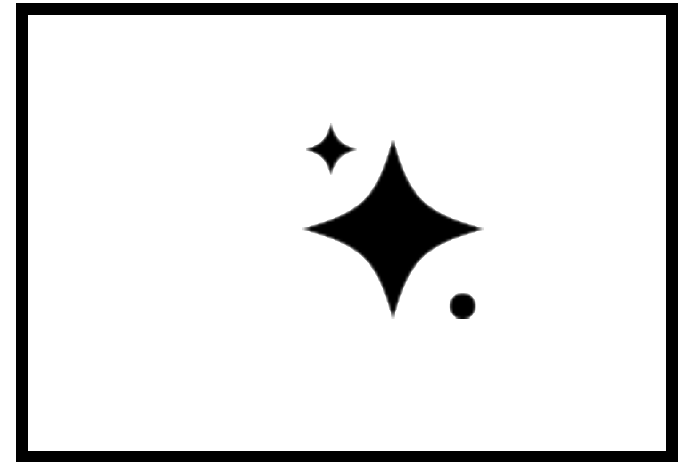
(2) <read> ...



(4) updated prompt

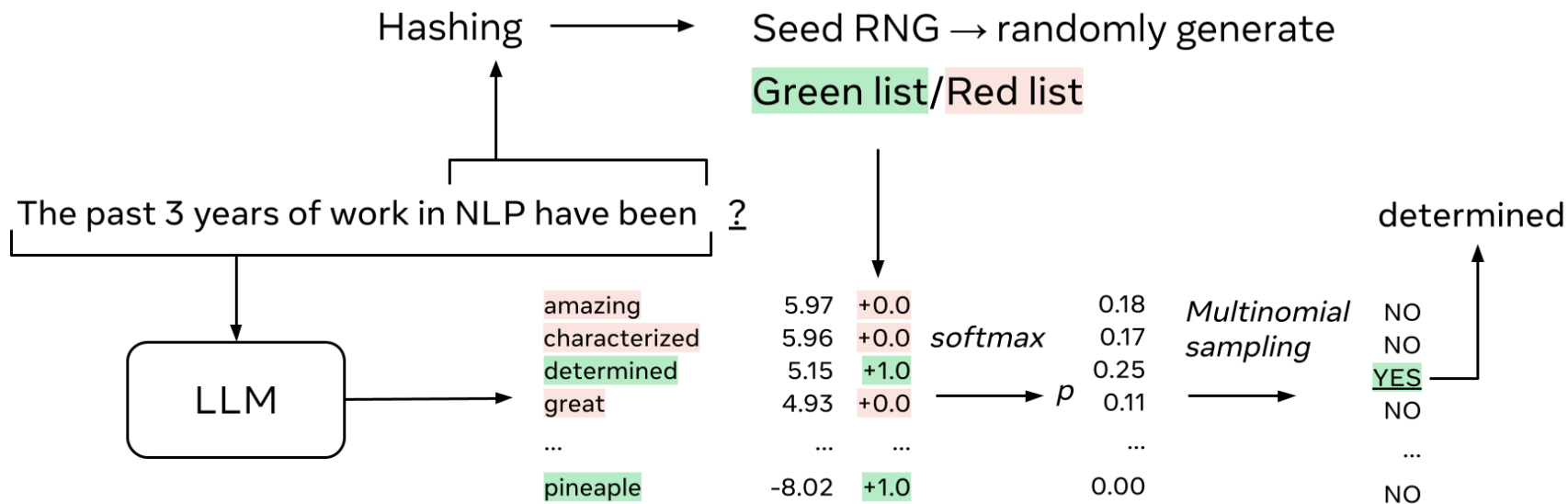


LLM serving system



Challenge 3. Implementation efficiency

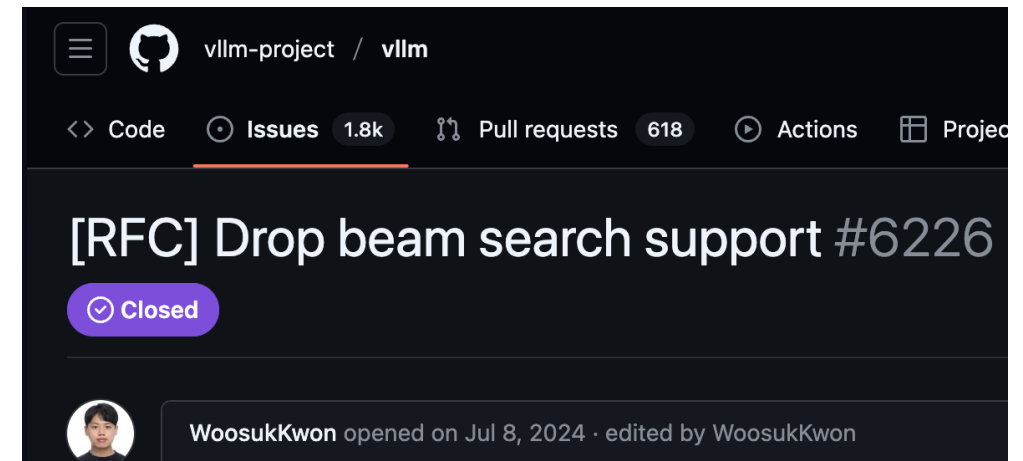
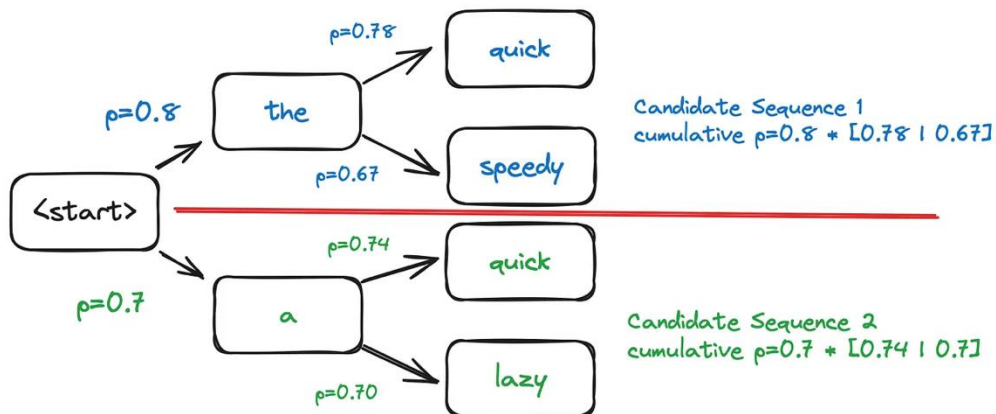
Scenario: Watermark output suggestions for IP protection



Challenge 3. Implementation efficiency

Current LLM serving systems confound **generation process** and **model computation** by employing a global autoregressive loop.

This makes it difficult to implement & manage generation strategies that deviates from the usual generation process (e.g., beam search)



Optimizations are global policy

```
[--speculative-config SPECULATIVE_CONFIG]
```

```

[ -v m s e e ] [ -h ] [ -m d o d M O D E L ]
[ -t s i s a d b , c l a s s i f y p r e f i x r e s t , e m b e d d i n g , g e n e t o o , j e w a r , d e a c e r , s i a n c o p i t o n ]
[ -t o k e n i z e r T O K E N I Z E R ]
[ -t o k e n i z e r m o d e ] [ -j o u t c o u r t u m i n t a s L e a n e ]
[ -t s e t m e m o c o s t e ] [ -n o t s e t m e m o c o s t e ]
[ -t g e p i u t o f o r t s A o u t A o u t M o K i n d I S T a n g ]
[ -s e e s I S E D ] [ -t r c o n f i g p a t h H F _ C O N F I G _ P A T H ]
[ -a l l o w e d _ l o c a l m e d i a p a t h A L L O W E D _ L O C A L _ M E ( B A , P _ P A T H ) ]
[ -t e a t o r e v I S I O N ] [ -c o s t e v a t o r c o s t _ R E I D E N ]
[ -n o s c a t e r p r o c s _ S C A L I N G ] [ -n o p e t h e t o c s _ T H E T A ]
[ -q u a n t i z e r m o d e K N E I D Z E R , R E V I O N ]
[ -x m s d e l t a m a x _ M O D E L L E N ]
[ -c a n r e n e r a n a n c y m e m l i b t a s j a t s a n d b y s c o m p r e s s e d t e m o n e s s e p e s s e d i p t e r m s _ m s d l i g e m m _ g i t s , f i l t e r g g i t s , b i t b l a s a d _ m a r t i n , g i t s _ m a r t i n , Z a , h u p e s _ m a r t i n , d o m o l o p m o c _ m a r t i n , n e u r o _ n _ q u a n t _ m a p 4 b i t , f i l t e r , a p p a y u a r t c o c h a c o p u _ m i n , t h e n e ]
[ -e n t c o s e n g e ] [ -n o e n t c o s e n g e ]
[ -m a x e n t l e n t a p a t h m a x _ R E Q _ L E N , L O _ T _ C A P T U R E ]
[ -m a x l o g p a r m a x _ L O O P R O B ]
[ -d i s a b l e s l i d i n g _ w i n d o w ] [ -n o d i s a b l e s l i d i n g _ w i n d o w ]
[ -d i s a b l e t h e c a s e a d d t r ] [ -n o d i s a b l e t h e c a s e a d d t r ]
[ -s k i p t o k e n i z e r i n t ] [ -n o s k i p t o k e n i z e r i n t ]
[ -e n a b l e p r o m p t _ e m b e d s ] [ -n o e n a b l e p r o m p t _ e m b e d s ]
[ -s e r v e r m o d e n a m e S E R V _ M O D E L _ N A M E [ S E R V _ D _ M O D E L _ N A M E _ ] ]
[ -d i s a b l e t e a r n e o u t p u t p r i n t ]
[ -c o n f i g f o r m a t ( a u t d i s , m e s s a g e ) ] [ -t h e t o k a n H F _ T O K E N ]
[ -t r o u t e r s H F _ O U T R O U T S ]
[ -o v e r r i d e n e u r o n c o n f i g O V E R R I D E _ N E U R O N _ C O N F I G ]
[ -o v e r r i d e p o o l c o n f i g O V E R R I D E _ P O O L _ C O N F I G ]
[ -l o g g i n g p r o c e s s o r p a r t e r L O G G I N G _ P R O C E S S O R _ P A T T E R N ]
[ -g e n e r a t o n c o n f i g G E N E R A T I O N _ C O N F I G ]
[ -o v e r r i d e g e n e t o n c o n f i g O V E R R I D E _ G E N E R A T I O N _ C O N F I G ]
[ -e n a b l e s l e e p _ m o d e ] [ -n o e n a b l e s l e e p _ m o d e ]
[ -m o d e l i n g ( a u t c u l t i v a t e r o m e m o ) ]
[ -m o d e l f o r m a t f a t h , k i t a f e m m o t u p e a c t u a l u m m y t e m o r l e , r a t h a r d e d , s i t a t g u f , z i h a n d i t y m , i s t a L u n a ] [ i t r e a m e r t u n a l , s e e e m e , c h a r d e d t a s t e s a f e t e m o ]
[ -d o w n l o a d a d d D O W N L O A D _ D I R ]
[ -m o d e l d e l e t e m o d e c o n f i g M O D E L _ L O A D E R , E X T R A _ C O N F I G ]
[ -i g n o r e p a r t t e r s I G N O R E _ P A T T E R N S [ I G N O R E _ P A T T E R N S _ ] ]
[ -u s e t d r o m l o a d ] [ -n o u s e t d r o m l o a d ]
[ -d i s a b l e p a t e r n c o m p e r p a t t Q U O R A _ A D A P T E R _ N A M E , O R _ P A T H ]
[ -p t l o a d m a p l o c a t i o n P T _ L O A D _ M A P _ L O C A T I O N ]
[ -g u i d e d e c o d i n g b i t k a n d ( a u t g u a i d e m a x t o m a t _ n o m a t _ n o r e c e r , p a t t e m , x g r a m m e r ) ]
[ -g u i d e d e c o d i n g _ d i s a b l e t h e f a i l t a c k ] [ -n o g u i d e d e c o d i n g _ d i s a b l e t h e f a i l t a c k ]
[ -g u i d e d e c o d i n g _ d i s a b l e t h e w h i t e s p a c e ] [ -n o g u i d e d e c o d i n g _ d i s a b l e t h e w h i t e s p a c e ]
[ -g u i d e d e c o d i n g _ d i s a b l e t h e a d d i t i o n a l p r o p e r t y ] [ -n o g u i d e d e c o d i n g _ d i s a b l e t h e a d d i t i o n a l p r o p e r t y ]
[ -e n a b l e m e a n o n i n g ] [ -n o e n a b l e m e a n o n i n g ]
[ -m e a n o n i n g p a r a m e t e r d i s t r i b u t i o n , i _ g a n n e t , a v e r a g e ]
[ -d i s t r i b u t e m e x e c u t o r b a n d k a n d ( e m m a t , l a u n c h e r , m p u , a y , a m i , N o n ) ]
[ -p a t e m p a r a m e t e r s i z e P A T T E R N _ P A R A M E T E R _ S I Z E ]
[ -m e m o r y p a r a m e t e r s i z e T E M P O R A _ P A R A M E T E R _ S I Z E ]
[ -d a t a p a r a m e t e r s i z e D A T A _ P A R A M E T E R _ S I Z E ]
[ -d a t a p a r a m e t e r s i z e D A T A _ P A R A M E T E R _ S I Z E _ L O C A L ]
[ -d a t a p a r a m e t e r s i z e D A T A _ P A R A M E T E R _ S I Z E _ L O C A L _ I S S ]
[ -d a t a p a r a m e t e r s i z e D A T A _ P A R A M E T E R _ S I Z E _ P O R T ]
[ -m a x e x p e r t p a r a m e t e r s i z e M A X _ P A R A M E T E R _ S I Z E ]
[ -d a t a p a r a m e t e r s i z e M A X _ P A R A M E T E R _ S I Z E _ L O A D I N G _ W O R K E R S ]
[ -n o y w e d e n e u r o n m a i g n i t ] [ -n o n o y w e d e n e u r o n m a i g n i t ]
[ -d i s a b l e t h e c u r t o m a l l m e d a c e ] [ -n o d i s a b l e t h e c u r t o m a l l m e d a c e ]
[ -w o r k e r s i z e W O R K E R _ C L S ]
[ -w o r k e r e x t e n s i o n s i z e W O R K E R _ E X T E N S I O N _ C L S ]
[ -b l o c k i z e i t b , h l i z d , a l l i z d ]
[ -g a p m e m o r y u t i l i z a t i o n G A P _ M E M O R Y _ U T I L I Z A T I O N ]
[ -s w a p s p a c e S W A P _ S P A C E ]
[ -i v c a s e t h e g e p i u t o f o r t s A o u t m o d e , a d a p t e r , a d a p t e r ]
[ -n u m g a p b l o c k s o v e r r i d e N U M _ O P _ B L O C K S _ O V E R R I D E ]
[ -e n a b l e m e c a c h e c a c h i n g ] [ -n o e n a b l e m e c a c h e c a c h i n g ]
[ -p r e f i x c a c h e t r a n s h a n d i n g p a t t e m a d a p t e r ]
[ -c p u o f f l o a d g b C P U _ O F F L O A D _ G B ]
[ -c a l c u l a t e s c a l e m e ] [ -n o c a l c u l a t e s c a l e m e ]
[ -t o k e n i z e r p o o l s i z e T O K E N I Z E R _ P O O L _ S I Z E ]
[ -t o k e n i z e r p o o l s i z e T O K E N I Z E R _ P O O L _ T Y P E ]
[ -t o k e n i z e r p o o l s e a r c o n f i g T O K E N I Z E R _ P O O L _ S E A R C O N F I G ]
[ -l i m i t m e m p a r p r o m p t L I M I T _ M A P _ P R O M P T _ P A T T E R N ]
[ -m e m p r o c e s s o r k e a n g M E M _ P R O C E S S O R _ K E A N G ]
[ -d i s a b l e m e m p r o c e s s o r c a c h e ] [ -n o d i s a b l e m e m p r o c e s s o r c a c h e ]
[ -e n a b l e b i o r a ] [ -n o e n a b l e b i o r a ]
[ -e n a b l e b i o r a b i a s ] [ -n o e n a b l e b i o r a b i a s ]
[ -m a x l o r a m a x _ L O R R S ] [ -m a x l o r a m a x _ L O R R _ L O A D I N G _ F A C T O R S ]
[ -m a x e n a v e r a b i l i t y L O R R _ E X T R A _ V O C A B _ S I Z E ]
[ -o r d g e p i u t o f o r t s A o u t I n g ]
[ -l o n g l e n t r a i n f i l t e r l o n g L O N G _ L O R R _ S C A L I N G _ F A C T O R S [ L O N G _ L O R R _ S C A L I N G _ F A C T O R S _ ] ]
[ -m a x k e p u f o r m a x _ O P _ L O A D ]
[ -f u l l y t h a r d e d f o r a s ] [ -n o f u l l y t h a r d e d f o r a s ]
[ -f u l l y t h a r d e d f o r a s p a t t e r ] [ -n o f u l l y t h a r d e d f o r a s p a t t e r ]
[ -m a x p r o m p t s i z e p a r a m e t e r m a x _ P R O M P T _ A D A P T E R _ I S S ]
[ -m a x p r o m p t s i z e p a r a m e t e r m a x _ P R O M P T _ A D A P T E R _ T O K E N ]
[ -d e v i c e ( a u t o c p u , z u a d a , t p u n e u r o n , t p u n e u r o ) ]
[ -s p e c u l a t i v e c o n f i g S P E C U L A T I V E _ C O N F I G ]
[ -s h o w h a d m e m o r y t r a c e f o r e v e n t s S H O W _ H I D E N _ M E T R I C S _ F O R V E R S I O N ]
[ -o t t p r a c e s
```

LLM serving paradigm should change

Model-centric

How fast (and how many) can it generate new tokens?



Application-centric

How efficiently does it support the application logic?

Our vision: “programmable” LLM serving system

We have seen similar problems/solutions in networking, OS, and security before:

- eBPF
- SDN
- OPA

Can we take the same philosophy in building the LLM serving system?

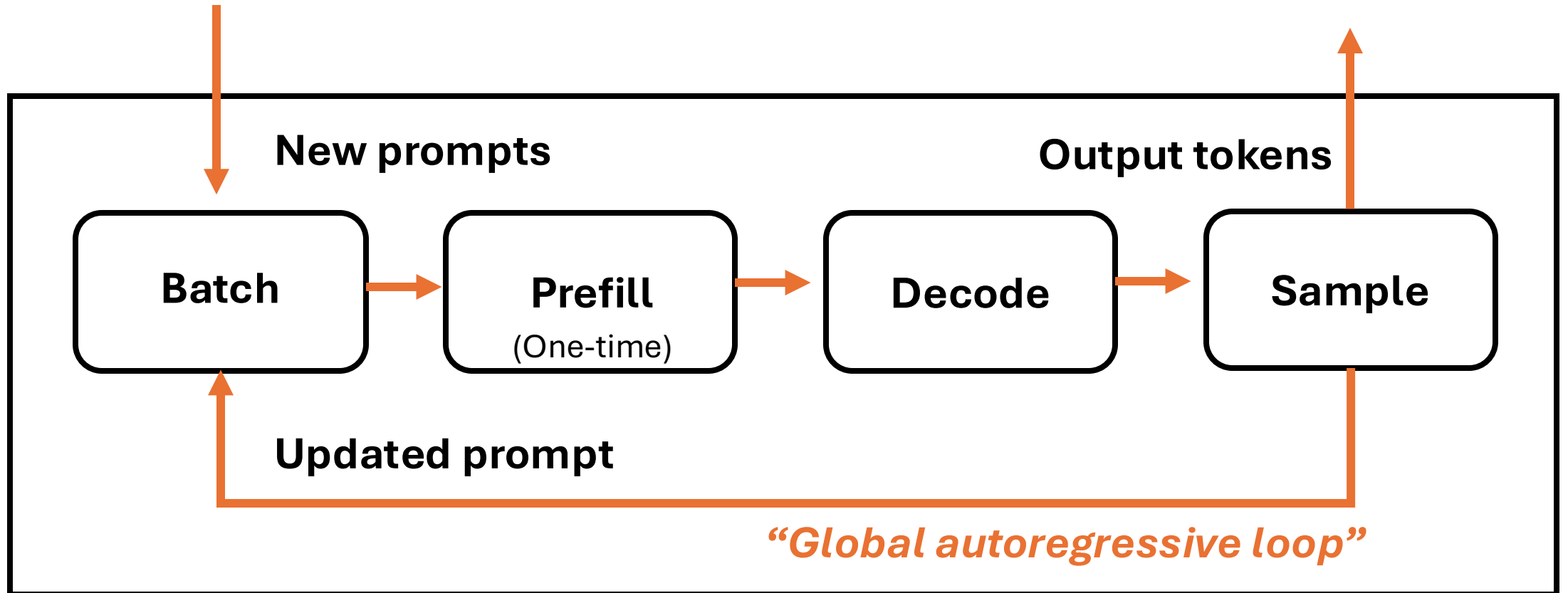
Our proposal

Symphony is a *programmable* LLM serving system that:

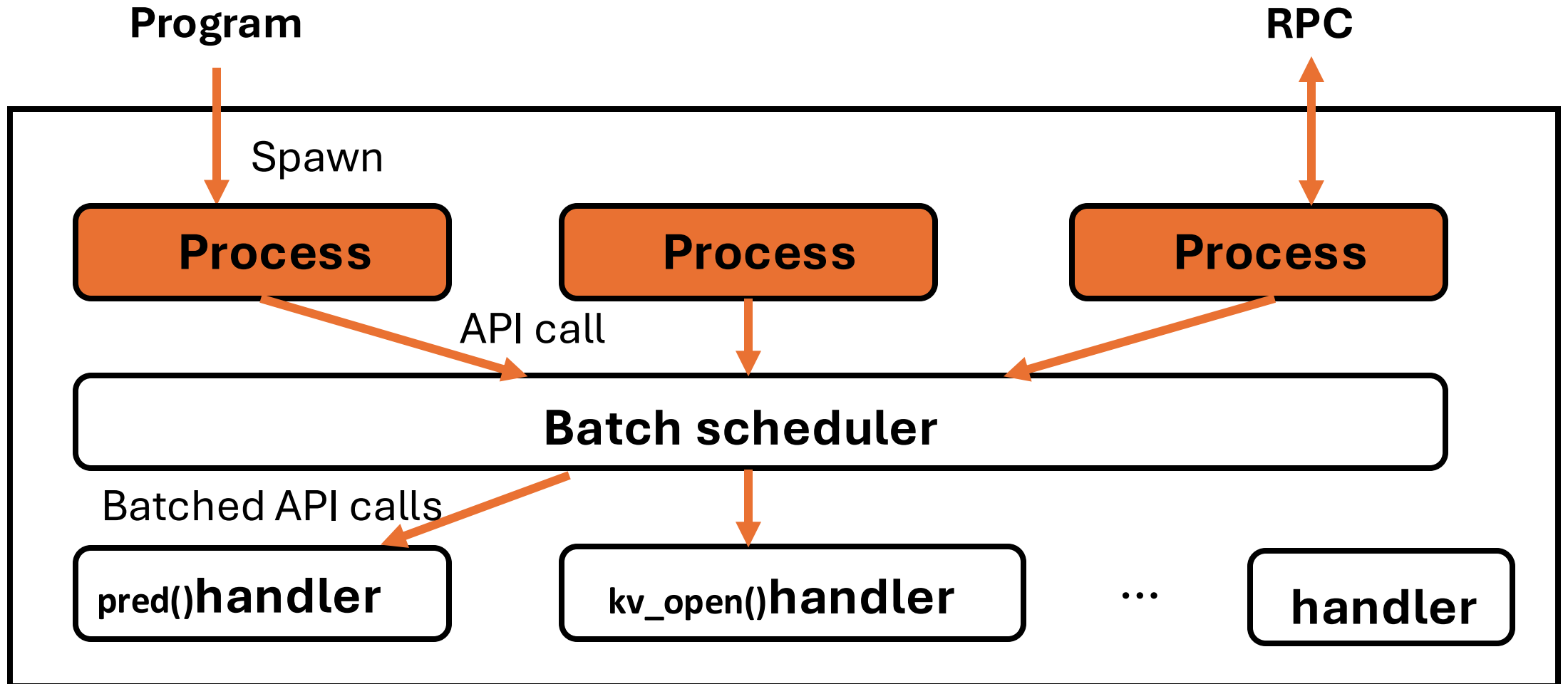
(1) **Decouples** control (e.g., autoregressive loop) and the model computation from the generation process

(2) **Delegates** the control entirely to the user-provided programs, which we refer to as *LLM Inference Programs (LIPs)*

Prompt-serving architecture



Symphony architecture



LLM Inference Programs (LIPs)

(Some) APIs provided by Symphony:

```
// model computation
pred(kv_cache, tok_ids, ...)

// KV cache management
kv_open
kv_create
kv_remove

// tokenization
tokenize

// RPC
send_to_user
```

LIP code for basic autoregressive decoding

```
prompt = "Hello, how are you?"

kv = kv_create()
tok_ids = tokenize(prompt)

for (i = 0; i < MAX_TOKENS; i++) {
    dist = pred(kv, tok_ids, ...)
    sampled = argmax(dist)
    send_to_user(detokenize(sampled))
    tok_ids = [sampled]
}
kv_remove(kv)
```



Inference efficiency

→ LIPs customize KV cache management per application



Interaction efficiency

→ LIPs integrate arbitrary I/O and compute within the generation



Implementation efficiency

→ LIPs define novel decoding strategies without modifying the serving system

LIP code for prefix caching + parallel decoding

```
// load precomputed kv cache
prefix_kv = kv_open("sys_msg.kv");
suffixes = {
    tokenize(** query 1 */), ...
    tokenize(** query n */)
};
for (suffix : suffixes) {
    // fork prefix kv and thread
    kv = kv_fork(prefix_kv);
    pthread_create({
        pos = prefix_kv.len(), step = 0;
        t = suffix;
        // generate until eos token
        while (t != EOS) {
            d = pred(kv, t, pos + step);
            t = argmax(d);
            step++;
            print(detokenize(t));
        }
        kv_remove(kv);
    });
}
join_all_threads();
kv_close(prefix_kv);
```

```
//load piecomputed kv cache
prefix_kv = kv_open("sys_msg_kv");
suffixes = {
    tokenize("** query1 *"),...
    tokenize("** queryn *")
};
for(suffix : suffixes) {
    //fork prefix kv and thread
    kv = kv_fork(prefix_kv);
    pthread_create(&{
        pos = prefix_kv.len(), step = 0;
        t = suffix;
        //generate until eos token
        while(t != EOS) {
            d = pred(kv, t, pos + step);
            t = argmax(d);
            step++;
            print(debtokenize(t));
        }
        kv_remove(kv);
    });
}
join_all_threads();
kv_close(prefix_kv);
```



```
//load piecomputed kv cache
prefix_kv = kv_open("sys_msg_kv");
suffixes = {
    tokenize("** query1 *"),...
    tokenize("** queryn *")
};
for(suffix : suffixes) {
    //fork prefix kv and thread
    kv = kv_fork(prefix_kv);
    pthread_create(&{
        pos = prefix_kv.len(), step = 0;
        t = suffix;
        //generate until eos token
        while(t != EOS) {
            d = pred(kv, t, pos + step);
            t = argmax(d);
            step++;
            print(debtokenize(t));
        }
        kv_remove(kv);
    });
}
join_all_threads();
kv_close(prefix_kv);
```

```
//load piecomputed kv cache
prefix_kv = kv_open("sys_msg_kv");
suffixes = {
    tokenize("** query1 *"),...
    tokenize("** queryn *")
};
for(suffix : suffixes) {
    //fork prefix kv and thread
    kv = kv_fork(prefix_kv);
    pthread_create(&{
        pos = prefix_kv.len(), step = 0;
        t = suffix;
        //generate until eos token
        while(t != EOS) {
            d = pred(kv, t, pos + step);
            t = argmax(d);
            step++;
            print(debtokenize(t));
        }
        kv_remove(kv);
    });
}
join_all_threads();
kv_close(prefix_kv);
```

```
//load piecomputed kv cache
prefix_kv = kv_open("sys_msg_kv");
suffixes = {
    tokenize("** query1 *"),...
    tokenize("** queryn *")
};
for(suffix : suffixes) {
    //fork prefix kv and thread
    kv = kv_fork(prefix_kv);
    pthread_create(&{
        pos = prefix_kv.len(), step = 0;
        t = suffix;
        //generate until eos token
        while(t != EOS) {
            d = pred(kv, t, pos + step);
            t = argmax(d);
            step++;
            print(debtokenize(t));
        }
        kv_remove(kv);
    });
}
join_all_threads();
kv_close(prefix_kv);
```

Symphony

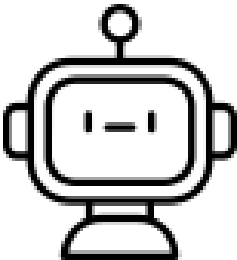
LLM serving system as OS

Symphony extends the responsibility of the LLM serving system to:

- (1) Abstracting resources (e.g., KV cache)
- (2) Scheduling LIPs
- (3) Handling I/O

We borrow familiar OS abstractions to describe Symphony's operation

AI app



```
prompt = "Hello, how are you?"  
  
kv = kv_create()  
tok_ids = tokenize(prompt)  
  
for (i = 0; i < MAX_TOKENS; i++) {  
    dist = pred(kv, tok_ids, _)  
    sampled = argmax(dist)  
    send_to_user(detokenize(sampled))  
    tok_ids = [sampled]  
}  
kv_remove(kv)
```

program



Symphony

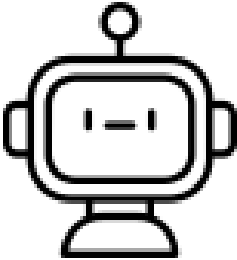
spawn



process

AI app

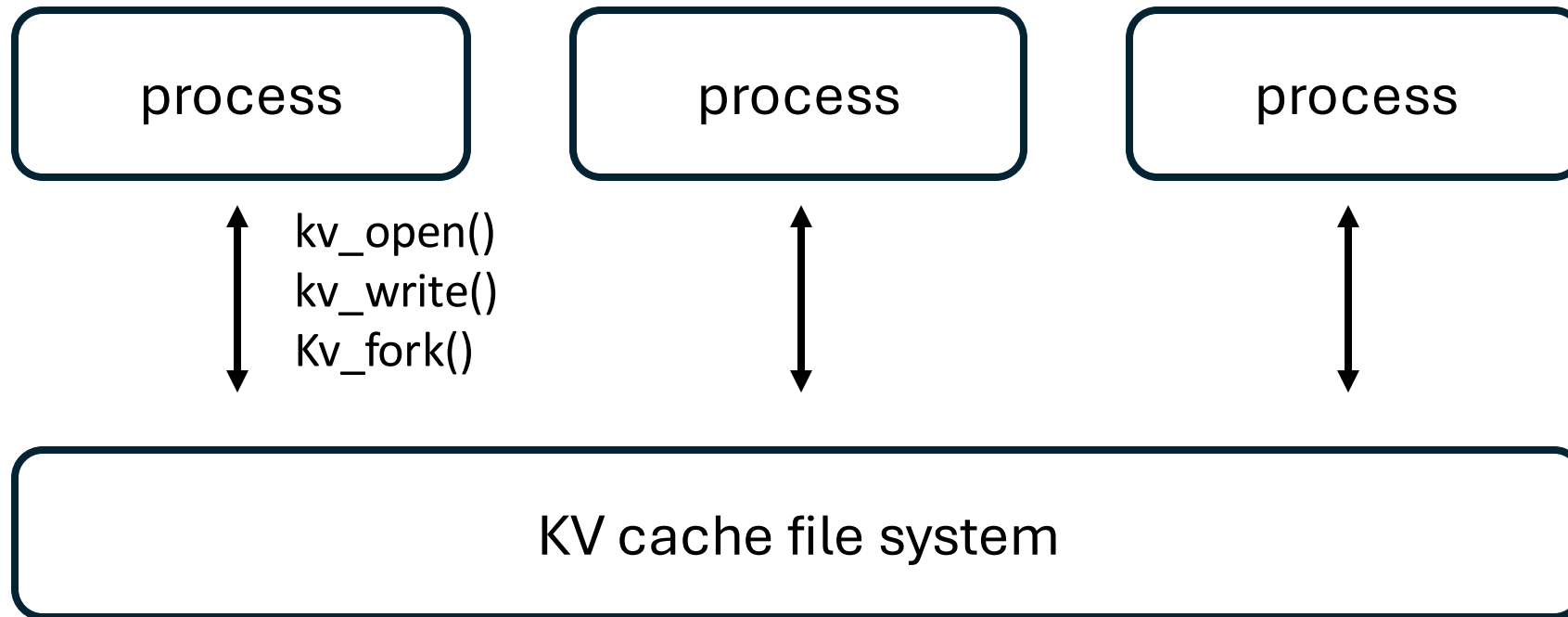
Symphony



RPC

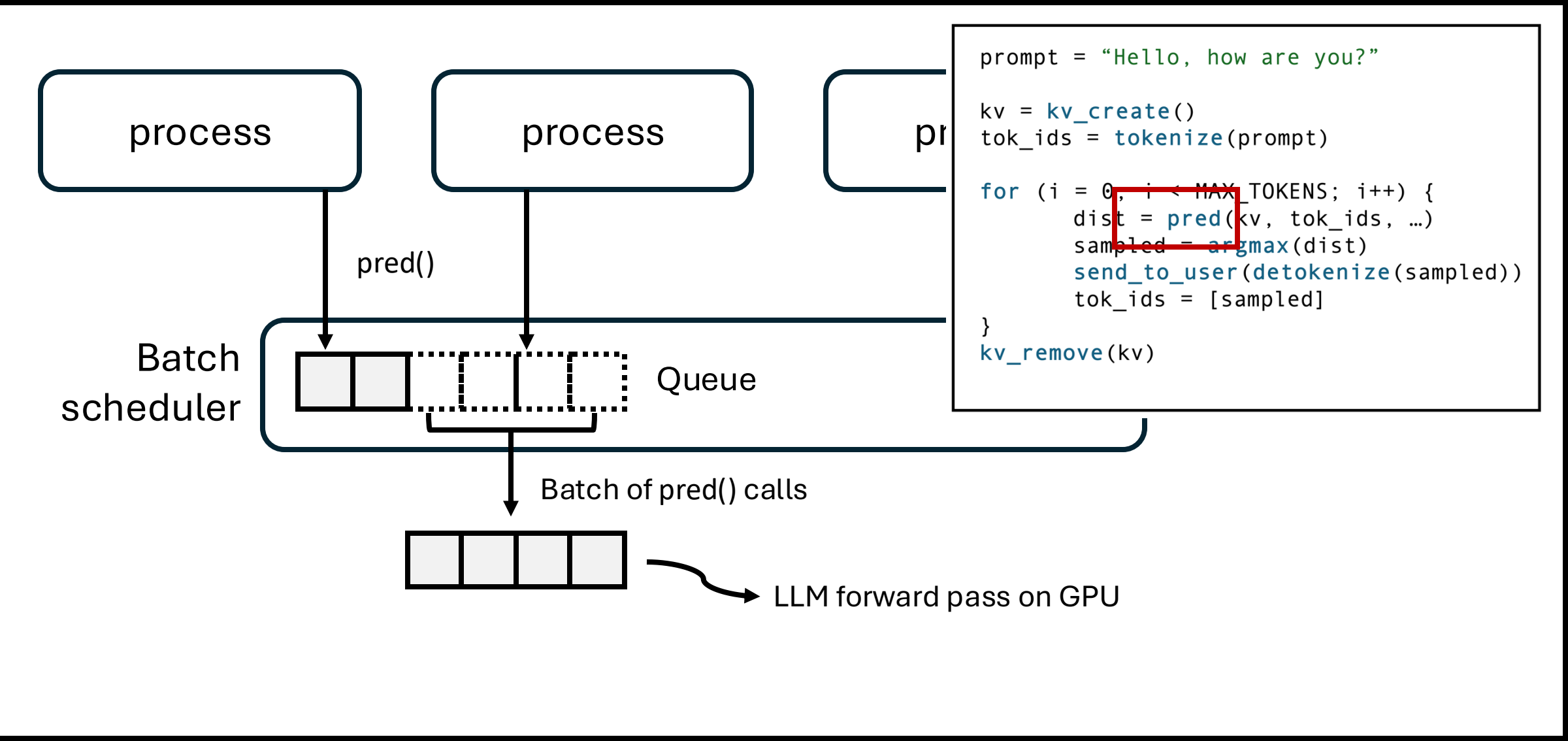


Symphony



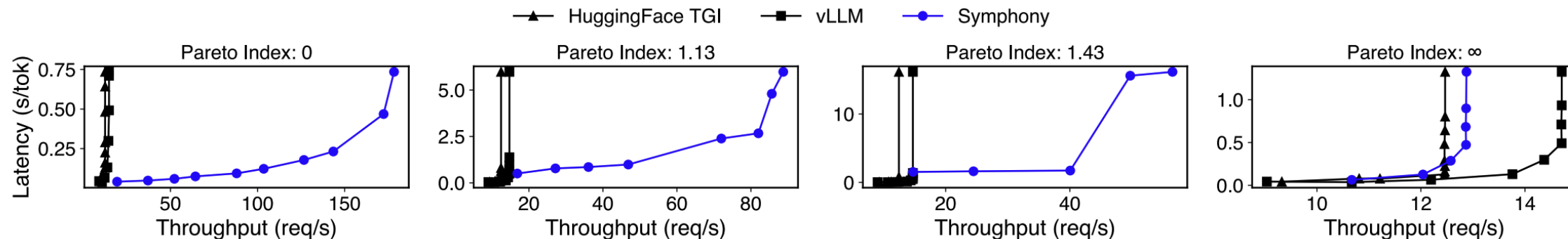
KV cache can be shared across processes

Symphony



Preliminary results

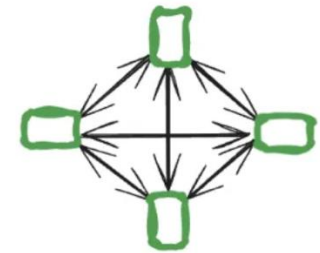
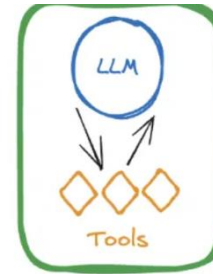
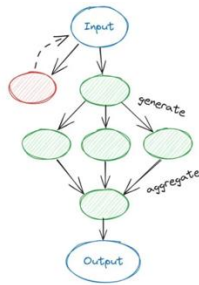
Benefits from application-specific optimization >> system overhead



User can exploit workload characteristics (Pareto Index) for optimal performance

Preliminary results

LIPs can implement wide spectrum of emerging LLM applications



Text completion

-4%-11%



Graph-of-Thought

15%-43%



ReACT agent

24%-



SWARM agents

55%-



End-to-end throughput improvements (baseline: vLLM/SGLang)

Challenges and open questions

- Security - What if LIPs are malicious?
- API design - What is the right interface granularity?