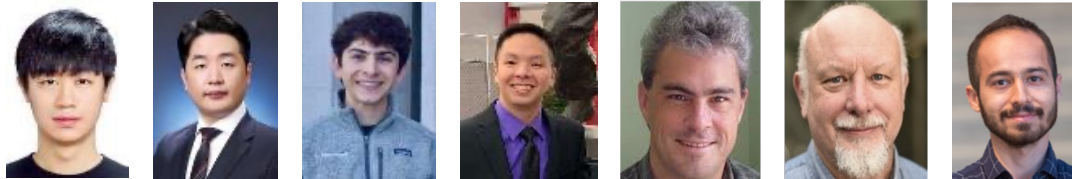


LLM Compiler for Parallel Function Calling

Sehoon Kim*, Suhong Moon*, Ryan Tabrizi, Nicholas Lee,
Michael Mahoney, Kurt Keutzer, **Amir Gholami**

University of California Berkeley

LlamaIndex Webinar, 2024

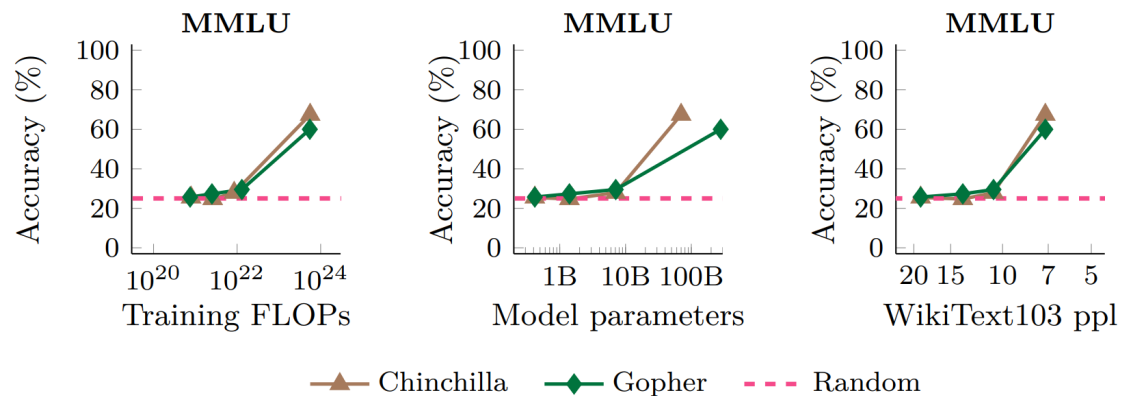


Emergent Properties of Large Language Models

- **Emergent properties** are only present in larger models, but not in small models.
- Opens up a new way of **few-shot learning** to solve more complex problems with in-context examples

1 Translate English to French: ← *task description*
2 sea otter => loutre de mer ← *examples*
3 peppermint => menthe poivrée
4 plush girafe => girafe peluche
5 cheese => ← *prompt*

Few-shot example



Chain-of-Thoughts Reasoning

A series of intermediate steps (i.e. chain-of-thoughts) improves LLM performance on complex reasoning tasks

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thoughts Reasoning

A series of intermediate steps (i.e. chain-of-thoughts) improves LLM performance on complex reasoning tasks

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

Are LLMs All We Need?

- LLMs still suffer from **math, knowledge cutoff, and hallucination**.
- This motivates us to equip LLMs with **external tools** to supplement these limitations.



241 - (-241) + 1



241 - (-241) + 1 is equivalent to $241 + 241 + 1$, which simplifies to $483 + 1$. So 241 - (-241) + 1 is equal to 484. ..?



You

what is apple's stock price today?



ChatGPT

As of my last update in April 2023, I don't have the capability to access or retrieve real-time data, including current stock prices. To find the latest stock price of Apple, I recommend checking a reliable financial news website, a stock market app, or a financial newspaper. These sources are regularly updated with the most recent stock prices and market information.



LLMs are bad at math

LLMs have knowledge cutoff

Function-Augmented LLMs

Toolformer: A pioneering work on augmenting tools (i.e. functions) to LLMs

- Let LLM output special tokens when it needs to use external functions, when we stop the LLM execution and run the functions ourselves, whose outputs are then appended back to the LLM

Out of 1400 participants, 400 (or `[Calculator(400 / 1400)`
`→ 0.29]` 29%) passed the test.

The name derives from "la tortuga", the Spanish word for
`[MT("tortuga") → turtle]` turtle.

The Brown Act is California's law `[WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.]` that requires legislative bodies, like city councils, to hold their meetings open to the public.

Function-Augmented LLMs

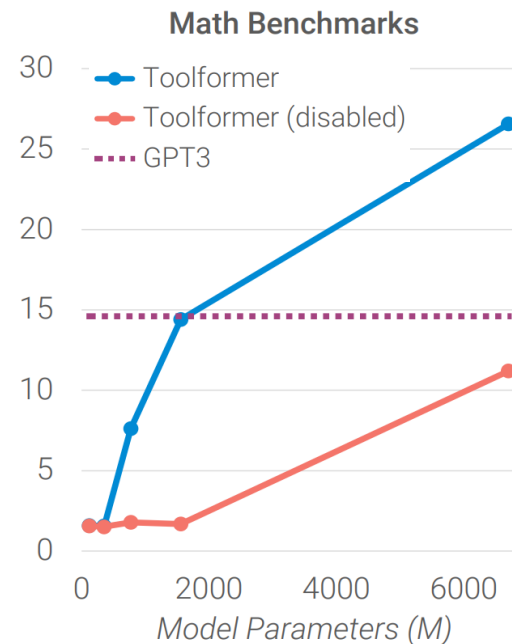
Toolformer: A pioneering work on augmenting tools (i.e. functions) to LLMs

- Let LLM output special tokens when it needs to use external functions, when we stop the LLM execution and run the functions ourselves, whose outputs are then appended back to the LLM

Out of 1400 participants, 400 (or `Calculator(400 / 1400)`
→ 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for
`MT("tortuga")` → turtle] turtle.

The Brown Act is California's law `WikiSearch("Brown Act")` → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.



ReAct: Reason and Action

ReAct: a pioneering work in enabling function calling + reasoning

- **Sequential generation** of Reasoning (i.e. “Thought”) and function calling(i.e. “Action”)
- Widely adapted in well-known LLM frameworks including LlamaIndex and LangChain

Query: Were Scott Derrickson and Ed Wood of the same nationality?

Thought 1: I need to search Scott Derrickson.

Action 1: search(Scott Derrickson)

Observation 1: Scott Derrickson (born July 16, 1966) is an **American filmmaker**. He is best known for his work in the horror genre, directing films such as The Exorcism of Emily Rose (2005), Sinister (2012) and The Black Phone (2022)...

Thought 2: I need to search Ed Wood.

Action 2: search(Ed Wood)

Observation 2: Edward Davis Wood Jr. (October 10, 1924 – December 10, 1978) was **an American filmmaker**, actor, screenwriter, and pulp novel author...

Thought 3: They are both American filmmakers.

Action 3: finish(yes)

Example Failure Case with ReAct

- ReAct can get into infinite loop of function calling

Query: Were John Smith and Ed Wood of the same nationality?

Thought 1: I need to search John Smith

Action 1: search(John Smith)

Observation 1: There are multiple matches for John Smith. This could refer to ...

Thought 2: I need to search John Smith

Action 2: search(John Smith)

Observation 2: There are multiple matches for John Smith. This could refer to ...

Thought 3: I need to search John Smith

Action 3: search(John Smith)

Observation 3: There are multiple matches for John Smith. This could refer to ...



Challenges with ReAct

With these approaches it is very challenging to develop large scale intelligent software:

- **Accuracy:** Concatenating intermediate observations can affect the execution flow of the LLM, **potentially reducing accuracy**
- **Serial Execution:** Not possible to run multiple tools in parallel
- **Reliability:** Intermediate results can affect the LLMs ability to keep track of the task
- **Testability:** Hard to create unit tests for specific paths of the code
- **Long Term Planning:** Current LLMs are not good at long term planning
- **Debugging:** Requires manually reading intermediate thoughts/observations and reasoning why the LLM got the wrong results
- **Fault Tolerance:** Hard to recover from wrong LLM decisions (no replanning)

But recent LLMs are very good at simple function calling that involves few calls!

Let's take a step back

Recent LLMs are very good at simple function calling!

- 💡 What if we had a way to break the problem into more bounded function calls?
- That is exactly what a good programmer does when writing large scale code:
 - We break our code into smaller pieces that are easy to reason about, debug, and test
 - We then write a controller logic that makes the appropriate calls to each of these smaller pieces with the appropriate try/except and error handling



Illustration Credit: [Isaac Rodriguez](#)

Incorporating a Systems View Helps Resolve These

Need a Compiler to orchestrate these

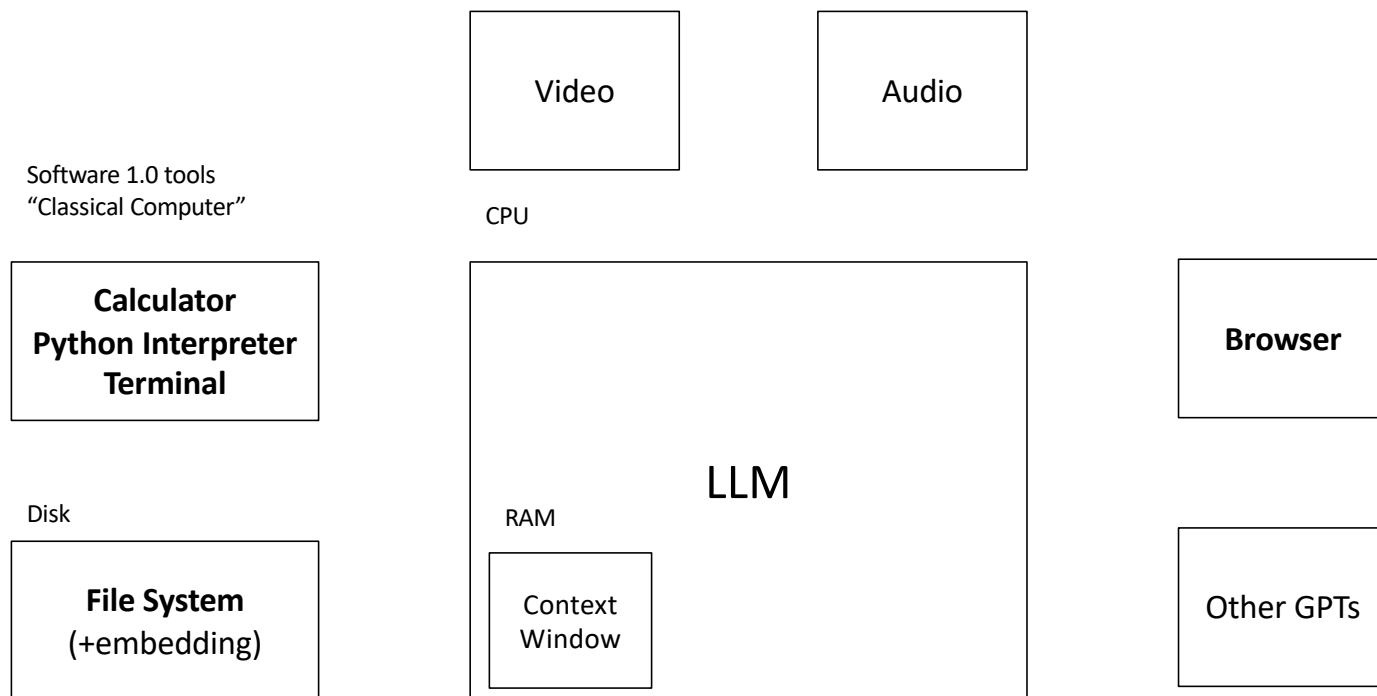


Illustration Credit: [Andrej Karpathy](#)

Incorporating a Systems View Helps Resolve These

Need a Compiler to orchestrate these

You can even imagine orchestrating the execution to a different LLMs depending on the difficulty

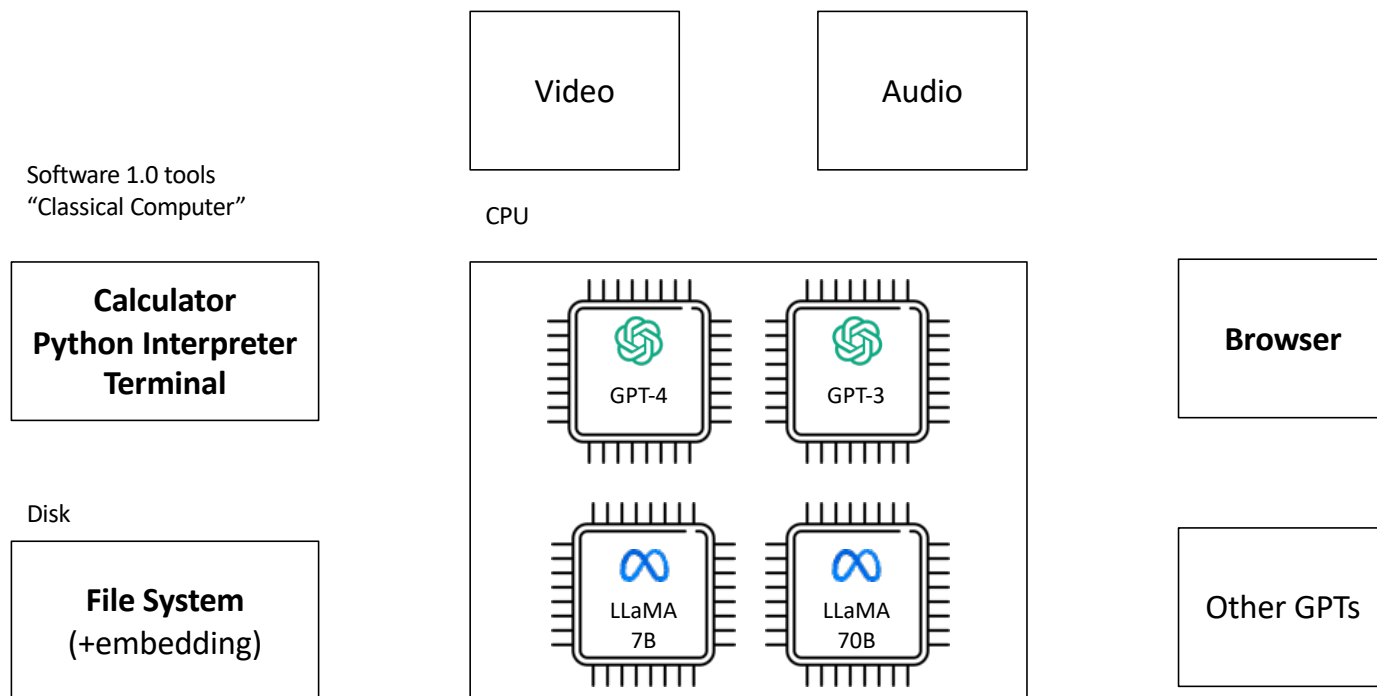
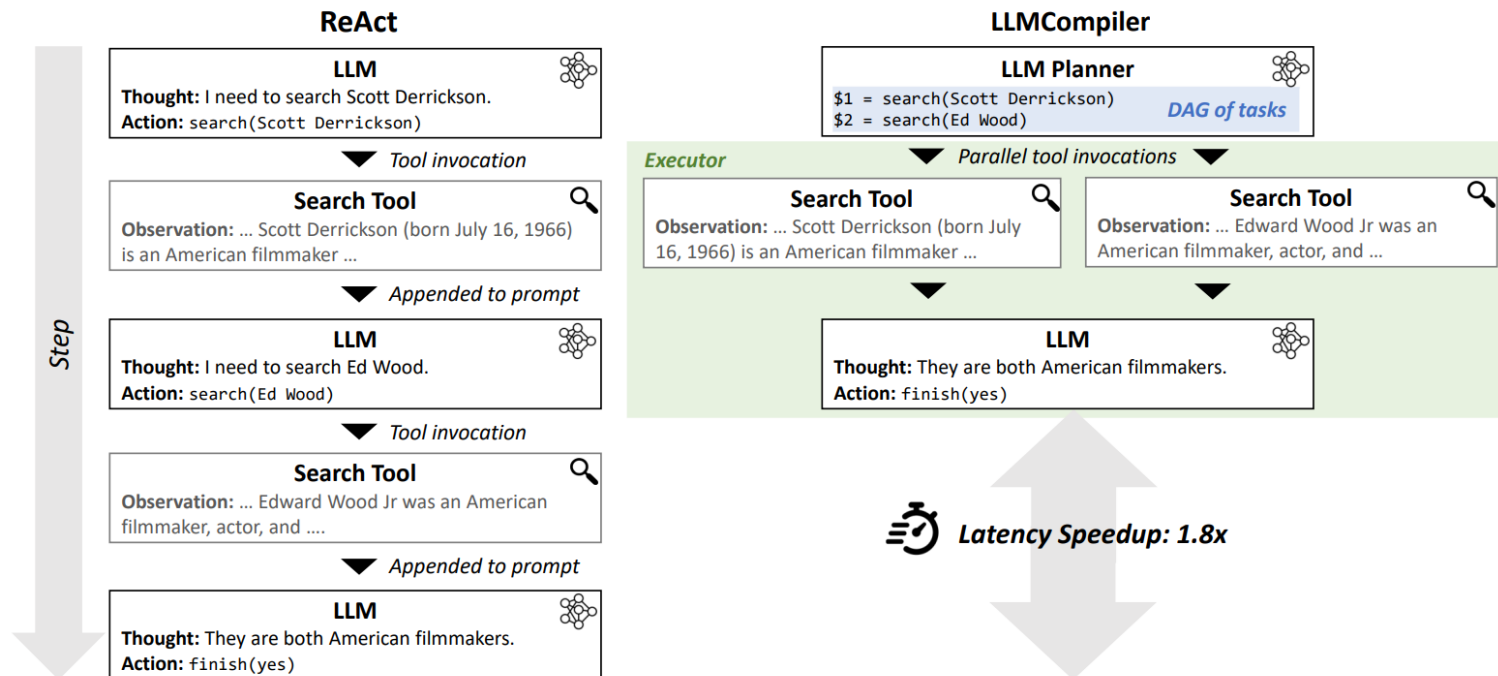


Illustration Credit: [Andrej Karpathy](#)

LLMCompiler: Towards Parallel Function Calling

- **Compiler** tailored for LLM function calling
 - **Efficiently orchestrate various function calls** and handle their dependencies.

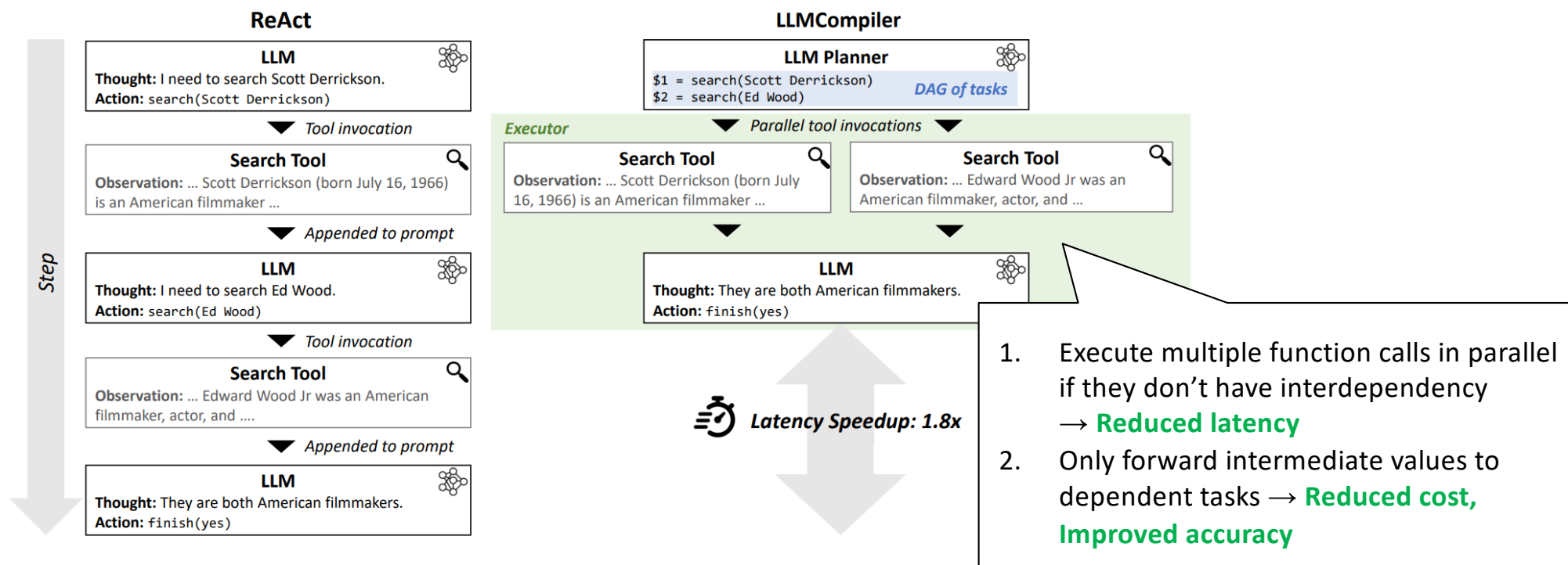
HotpotQA Question: Were Scott Derrickson and Ed Wood of the same nationality?



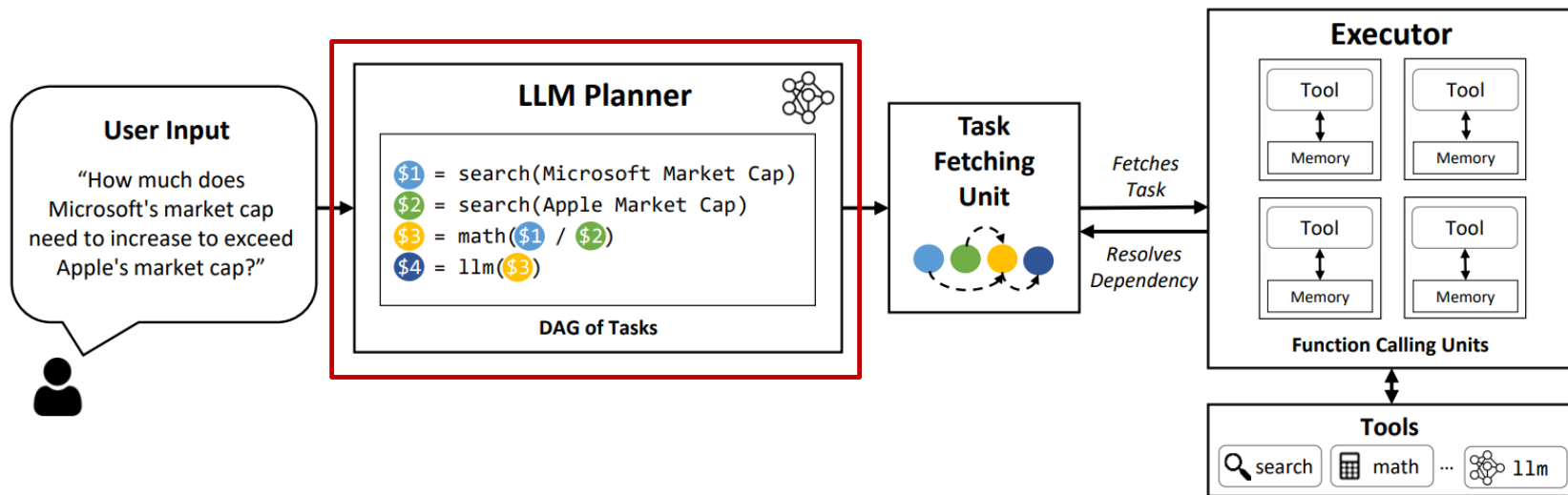
LLMCompiler: Towards Parallel Function Calling

- **Compiler** tailored for LLM function calling
 - **Efficiently orchestrate various function calls** and handle their dependencies.

HotpotQA Question: Were Scott Derrickson and Ed Wood of the same nationality?

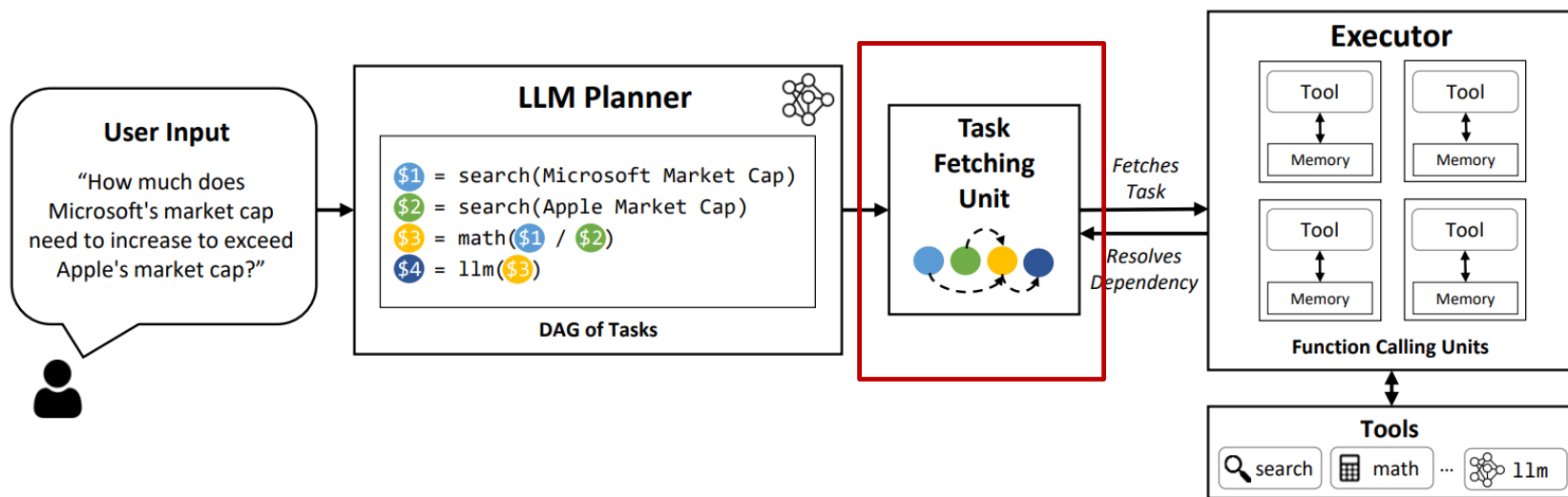


LLMCompiler: Towards Parallel Function Calling



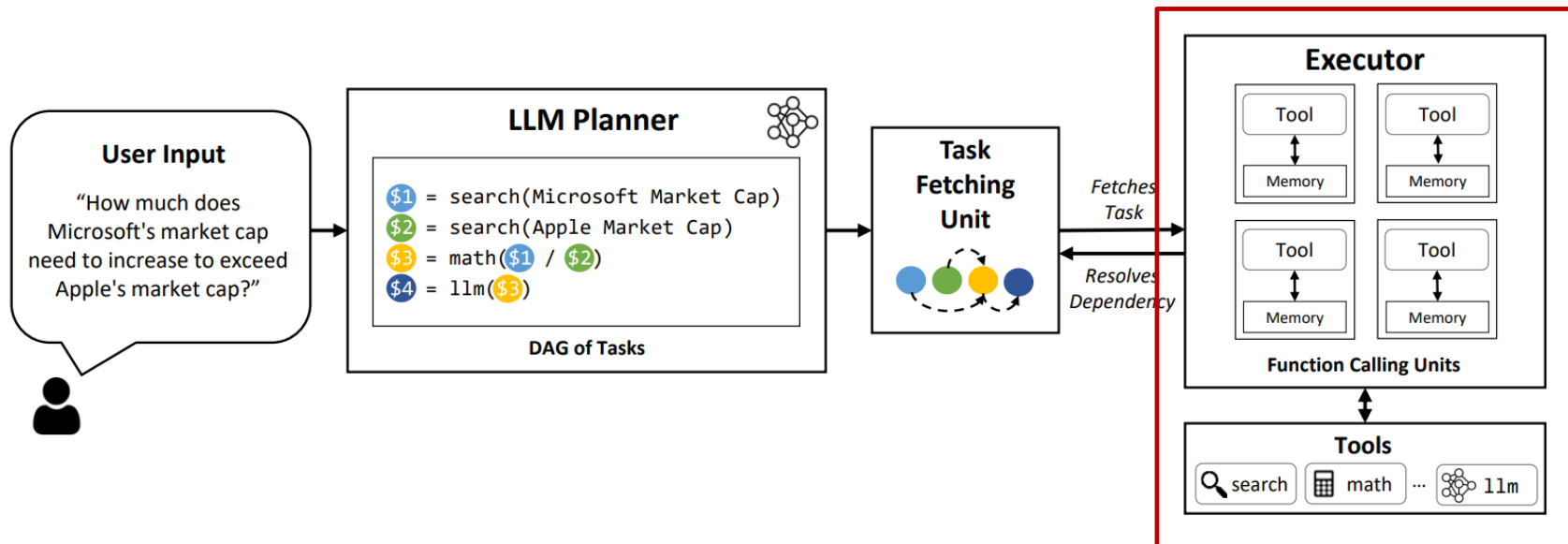
1. LLM Planner: Given a user input, it automatically identifies the necessary tasks, their input arguments, as well as the dependencies between them using the sophisticated reasoning capability of LLMs

LLMCompiler: Towards Parallel Function Calling



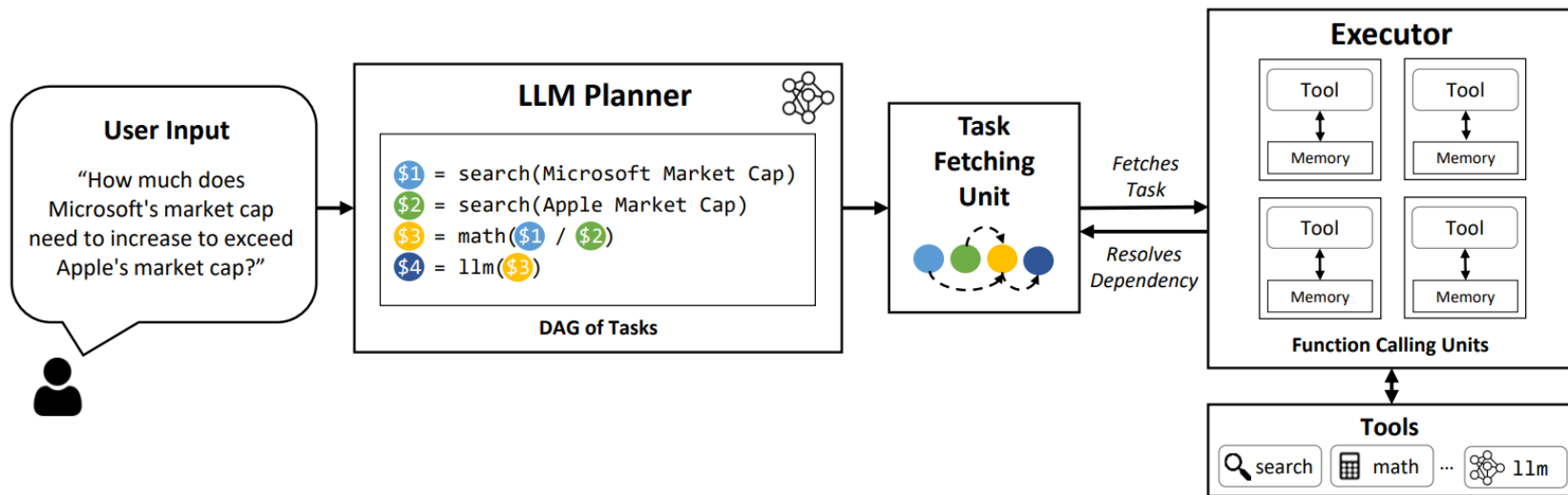
2. Task Fetching Unit: Similar to instruction fetching mechanism in contemporary computer architectures, it (1) replaces variables with the actual outputs from preceding tasks and (2) fetches tasks to the Executor as soon as they are ready for (parallel) execution.

LLMCompiler: Towards Parallel Function Calling



3. Executor: It is equipped with the tools that the user provides, and it delegates the task to the associated tool. **Tools** can be simple functions like a calculator, Wikipedia search, or API calls, or they can even be LLM agents that are tailored for a specific task.

LLMCompiler: Towards Parallel Function Calling



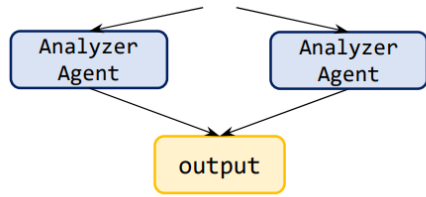
Simple Configuration

Users only need to supply:

1. **Tool Definitions:** Same as other frameworks (ReAct, OpenAI function calling, etc.)
2. **In-context Examples for the Planner:** Examples of how the Planner should behave, that can aid the Planner LLM in generating the appropriate dependency graph in the correct format for incoming inputs

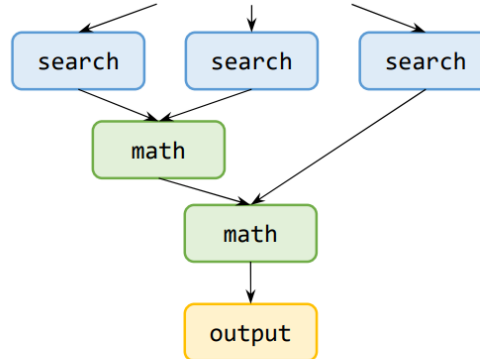
Dependency Graph Examples

Analyze Apple and Microsoft's latest 10-K form and compare their sales forecast.



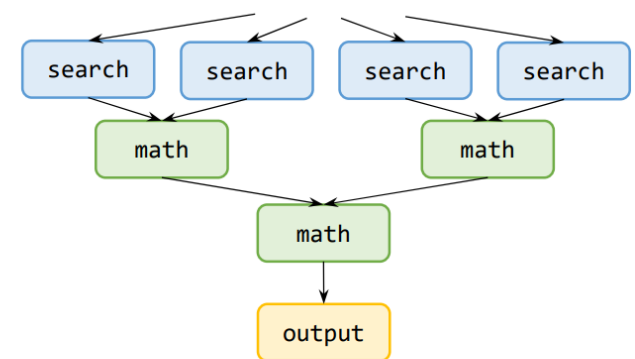
(a)

If Stanford and UCLA were to merge, would they have more Nobel laureates than UC Berkeley?



(b)

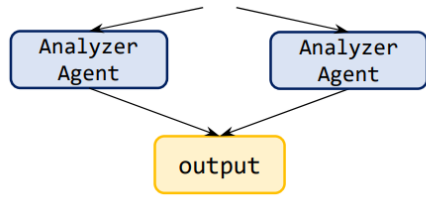
Which has higher total healthcare expenses, Florida or New York, considering both public and private sectors?



(c)

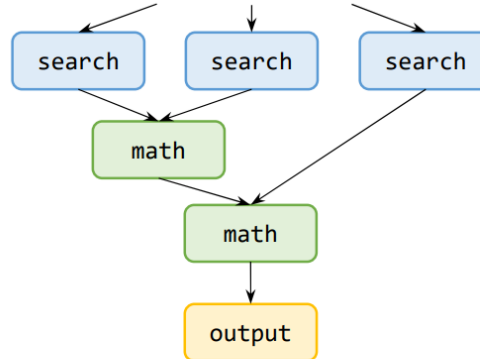
Pattern 1: Embarrassingly Parallel Function Calling

Analyze Apple and Microsoft's latest 10-K form and compare their sales forecast.



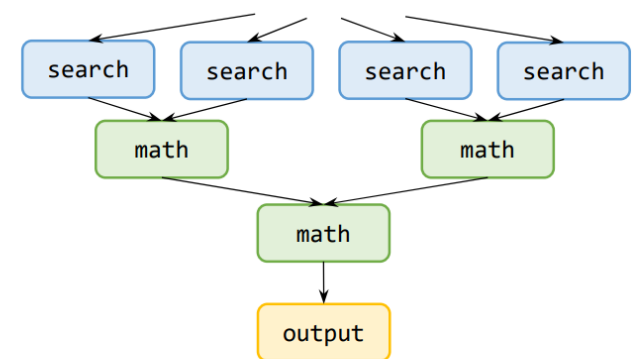
(a)

If Stanford and UCLA were to merge, would they have more Nobel laureates than UC Berkeley?



(b)

Which has higher total healthcare expenses, Florida or New York, considering both public and private sectors?



(c)

Pattern 1: Embarrassingly Parallel Function Calling

HotpotQA (Comparison)

e.g. Were Scott Derrickson and Ed Wood of the same nationality?

- 2-way parallelizable workload
- Tool: Wikipedia Search

Big Bench -- Movie Recommendation

e.g. Which movie among the options is the most similar to Mission Impossible, The Silence of the Lambs, American Beauty, and Star Wars Episode IV - A New Hope?

Options: Austin Powers International Man of Mystery, Alesha Popovich and Tugarin the Dragon, In Cold Blood, Rosetta

- 8-way parallelizable workload
- Tool: Wikipedia Search

Pattern 1: Embarrassingly Parallel Function Calling

Benchmark	Method	GPT (Closed-source)			LLaMA-2 70B (Open-source)		
		Accuracy (%)	Latency (s)	Speedup	Accuracy (%)	Latency (s)	Speedup
HotpotQA	ReAct	61.52	-	-	50.48	-	-
	ReAct [†]	62.47	7.12	1.00×	52.25	13.44	1.00×
	OAI Parallel Function	62.05	4.42	1.61×	-	-	-
	LLMCompiler	62.00	3.95	1.80×	55.28	9.58	1.40×
Movie Rec.	ReAct	45.93	-	-	73.20	-	-
	ReAct [†]	70.60	20.47	1.00×	73.00	33.37	1.00×
	OAI Parallel Function	77.00	7.42	2.76×	-	-	-
	LLMCompiler	77.60	5.47	3.74×	74.40	11.83	2.82×

ReAct[†] : With additional prompt that avoids repetitive function calls and early stopping

Open-source Model Support

- LLMCompiler supports both **closed-source models** (e.g. GPT) and **open-source models** (e.g. LLaMA). On the other hand, OpenAI's parallel function calling is only supported with GPT.

Pattern 1: Embarrassingly Parallel Function Calling

Benchmark	Method	GPT (Closed-source)			LLaMA-2 70B (Open-source)		
		Accuracy (%)	Latency (s)	Speedup	Accuracy (%)	Latency (s)	Speedup
HotpotQA	ReAct	61.52	-	-	50.48	-	-
	ReAct [†]	62.47	7.12	1.00×	52.25	13.44	1.00×
	OAI Parallel Function	62.05	4.42	1.61×	-	-	-
	LLMCompiler	62.00	3.95	1.80×	55.28	9.58	1.40×
Movie Rec.	ReAct	45.93	-	-	73.20	-	-
	ReAct [†]	70.60	20.47	1.00×	73.00	33.37	1.00×
	OAI Parallel Function	77.00	7.42	2.76×	-	-	-
	LLMCompiler	77.60	5.47	3.74×	74.40	11.83	2.82×

ReAct[†] : With additional prompt that avoids repetitive function calls and early stopping

Latency

- By avoiding sequential reasoning and function calling processes, LLMCompiler achieves up to **1.8x and 3.7x speedup** on each dataset compared to **ReAct**.
- Interestingly, LLMCompiler shows a **speedup of up to 35%** compared to **OpenAI parallel function calling**. One speculation is that there might be additional overheads behind the scene for validating the function and argument names, etc.

Pattern 1: Embarrassingly Parallel Function Calling

Benchmark	Method	GPT (Closed-source)			LLaMA-2 70B (Open-source)		
		Accuracy (%)	Latency (s)	Speedup	Accuracy (%)	Latency (s)	Speedup
HotpotQA	ReAct	61.52	-	-	50.48	-	-
	ReAct [†]	62.47	7.12	1.00×	52.25	13.44	1.00×
	OAI Parallel Function	62.05	4.42	1.61×	-	-	-
	LLMCompiler	62.00	3.95	1.80×	55.28	9.58	1.40×
Movie Rec.	ReAct	45.93	-	-	73.20	-	-
	ReAct [†]	70.60	20.47	1.00×	73.00	33.37	1.00×
	OAI Parallel Function	77.00	7.42	2.76×	-	-	-
	LLMCompiler	77.60	5.47	3.74×	74.40	11.83	2.82×

ReAct[†] : With additional prompt that avoids repetitive function calls and early stopping

Accuracy

- LLM Compiler achieves **better accuracy** compared to ReAct.
 - Common issues with ReAct is **repetitive function calls and early stopping**, which are also reported in the original paper. LLMCompiler avoids this by setting up a plan ahead-of-time, and by avoiding unexpected interference of intermediate observations in every reason-and-action iteration.
- LLMCompiler achieves **on-par accuracy** compared to OpenAI parallel function calling.

Pattern 1: Embarrassingly Parallel Function Calling

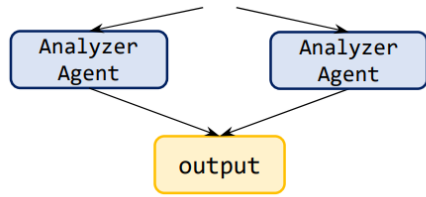
Benchmark	Method	In. Tokens	Out. Tokens	Cost (\$/1k)	Cost Red.
HotpotQA	ReAct	2900	120	5.00	1.00×
	OAI Parallel Function	2500	63	2.66	1.87×
	LLMCompiler	1300	80	1.47	3.37×
Movie Rec.	ReAct	20000	230	20.46	1.00×
	OAI Parallel Function	5800	160	6.14	3.33×
	LLMCompiler	2800	115	3.04	6.73×

Cost

- By reducing the repetitive LLM calls and concatenation of every observation, LLMCompiler achieves **3.4x and 6.7x cost reduction** compared to ReAct.

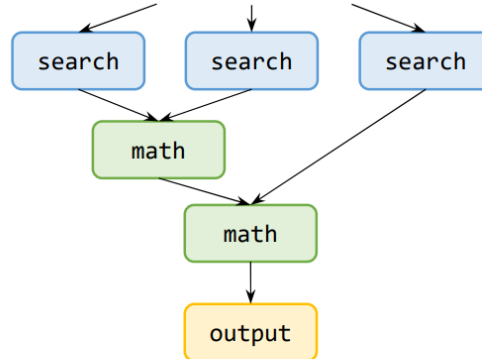
Dependency Graph Examples

Analyze Apple and Microsoft's latest 10-K form and compare their sales forecast.



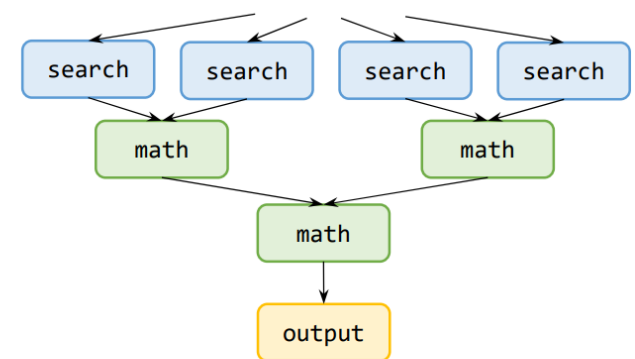
(a)

If Stanford and UCLA were to merge, would they have more Nobel laureates than UC Berkeley?



(b)

Which has higher total healthcare expenses, Florida or New York, considering both public and private sectors?



(c)

LLMCompiler Needs a Replanning Capability

1. In various applications, the execution graph can only be constructed based on the execution results of the previous function calling, which is unknown a priori.

e.g. Buying an item on the web – you cannot plan which items to buy before entering the web page

2. Often time, plans may fail or return unexpected outputs, which cannot be anticipated in the planner phase.

e.g. Search failure

LLMCompiler Needs a Replanning Capability

1. In various applications, the execution graph can only be constructed based on the execution results of the previous function calling, which is unknown a priori.

e.g. Buying an item on the web – you cannot plan which items to buy before entering the web page

The planner plans out a part of the entire plan, see the intermediate results, and plan out the rest

2. Often time, plans may fail or return unexpected outputs, which cannot be anticipated in the planner phase.

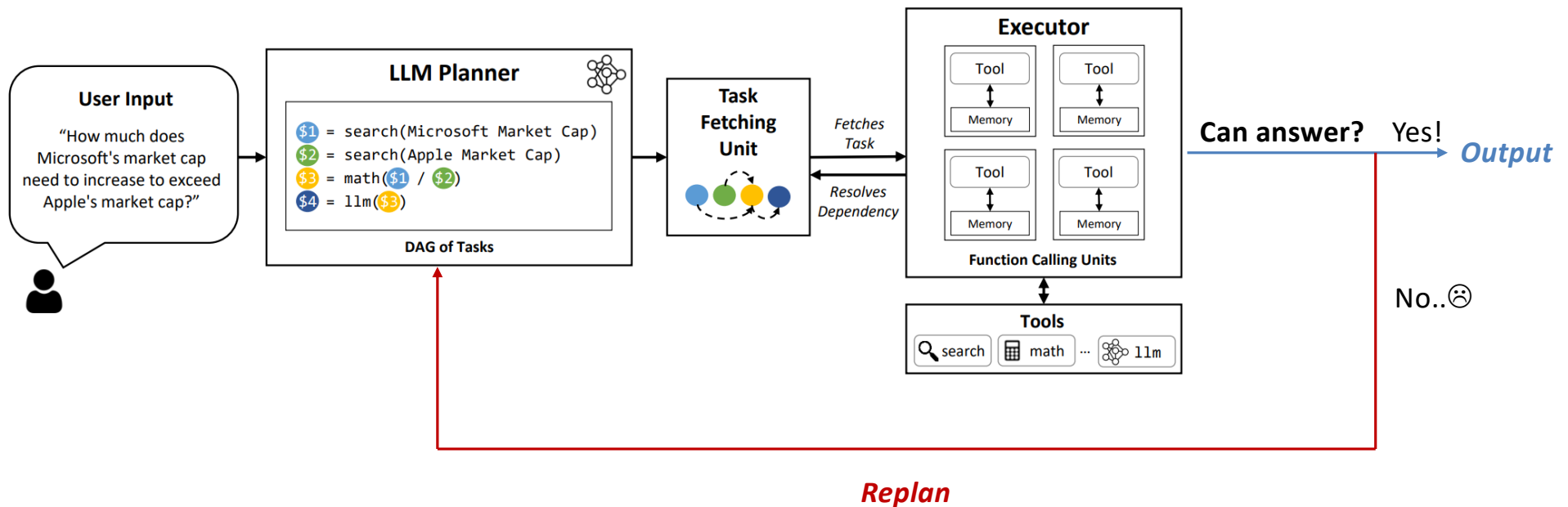
e.g. Search failure

The planner adjusts the original plan if it fails during execution

“Replanning can be a solution”

LLMCompiler Needs a Replanning Capability

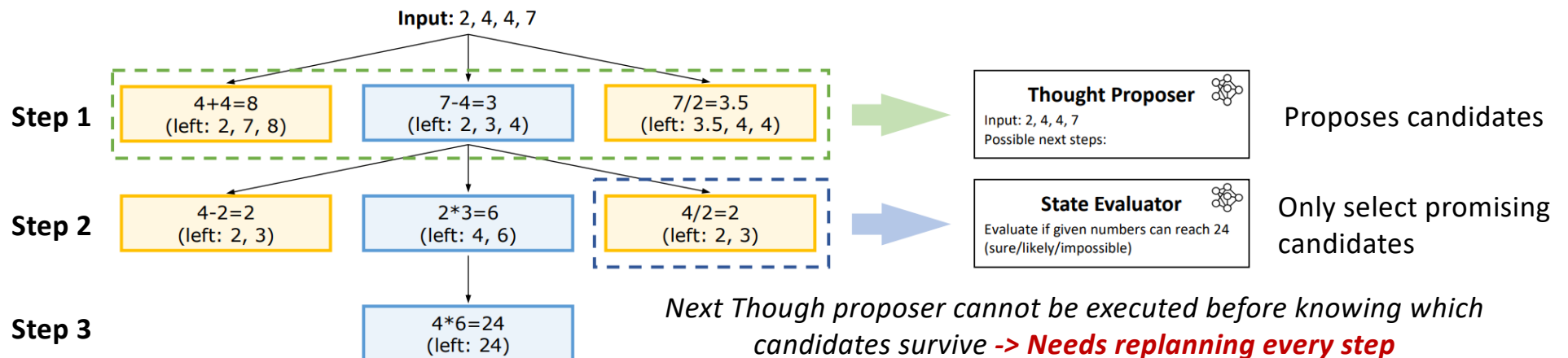
Replanning capability is needed in LLMCompiler!



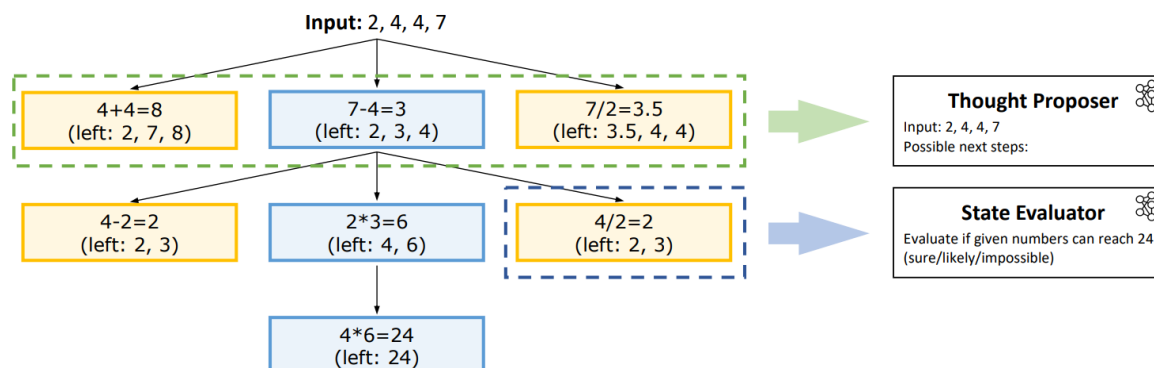
The Executor sends the intermediate results back to our LLM Planner.
Based on that, the Planner produces a new set of tasks

LLMCompiler Needs a Replanning Capability

e.g. **Game of 24**: using 4 numbers (e.g. 2, 4, 4, 7), make 24 with any permutation of those numbers, combined with operations (e.g. $(7-4)*2*4$)



LLMCompiler Needs a Replanning Capability



Benchmark	Method	GPT (Closed-source)			LLaMA-2 70B (Open-source)		
		Accuracy (%)	Latency (s)	Speedup	Accuracy (%)	Latency (s)	Speedup
Game of 24	Tree-of-Thoughts	74.00	241.2	1.00×	30.00	952.06	1.00×
	LLMCompiler	75.33	83.6	2.89×	32.00	456.02	2.09×

- Baseline Tree-of-Thought is extremely inefficient in latency since it needs to evaluate every candidate sequentially
- By executing them in parallel, LLMCompiler achieves **2-3x speedup**

Overall Latency and Accuracy Results

Benchmark	Method	GPT (Closed-source)			LLaMA-2 70B (Open-source)		
		Accuracy (%)	Latency (s)	Speedup	Accuracy (%)	Latency (s)	Speedup
HotpotQA	ReAct	61.52	-	-	50.48	-	-
	ReAct [†]	62.47	7.12	1.00×	52.25	13.44	1.00×
	OAI Parallel Function	62.05	4.42	1.61×	-	-	-
	LLMCompiler	62.00	3.95	1.80×	55.28	9.58	1.40×
Movie Rec.	ReAct	45.93	-	-	73.20	-	-
	ReAct [†]	70.60	20.47	1.00×	73.00	33.37	1.00×
	OAI Parallel Function	77.00	7.42	2.76×	-	-	-
	LLMCompiler	77.60	5.47	3.74×	74.40	11.83	2.82×
ParallelQA	ReAct	89.09	35.90	1.00×	59.59	15.47	1.00×
	OAI Parallel Function	87.32	19.29	1.86×	-	-	-
	LLMCompiler	89.38	16.69	2.15×	68.14	26.20	2.27×
Game of 24	Tree-of-Thoughts	74.00	241.2	1.00×	30.00	952.06	1.00×
	LLMCompiler	75.33	83.6	2.89×	32.00	456.02	2.09×

References

- [1] Shunyu Yao et al., Tree of Thoughts: Deliberate Problem Solving with Large Language Models, <https://arxiv.org/abs/2305.10601>
- [2] Shunyu Yao et al., ReAct: Synergizing Reasoning and Acting in Language Models, <https://arxiv.org/abs/2210.03629>
- [2] Zhilin Yang et al., HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering, <https://arxiv.org/abs/1809.09600>
- [3] Aarohi Srivastava et al., Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models, <https://arxiv.org/abs/2206.04615>
- [4] Wenhao Yu et al., IfQA: A Dataset for Open-domain Question Answering under Counterfactual Presuppositions, <https://arxiv.org/abs/2305.14010>
- [5] Lei Wang et al., Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models, <https://arxiv.org/abs/2305.04091>
- [6] Xuefei Ning et al., Skeleton-of-Thought: Large Language Models Can Do Parallel Decoding, <https://arxiv.org/abs/2307.15337>
- [7] Maciej Besta et al., Graph of Thoughts: Solving Elaborate Problems with Large Language Models, <https://arxiv.org/abs/2308.09687>
- [8] Binfeng Xu et al., ReWOO: Decoupling Reasoning from Observations for Efficient Augmented Language Models, <https://arxiv.org/abs/2305.18323>

Thank You

LLM Compiler for Parallel Function Calling

- Paper: <https://arxiv.org/abs/2312.04511>
- Official Codebase: <https://github.com/SqueezeAILab/LLMCompiler>
- LlamaIndex Integration: https://llamahub.ai/l/llama_packs-agents-llm_compiler?from=llama_packs

