Span Queries: a Declarative Map/Reduce Approach to Scale-up and Scale-out Inferencing

Nick Mitchell, Paul Castro, Mudhakar Srivastava

{nickm, castrop, msrivats}@us.ibm.com

IBM Research

What if we had a SQL for GenAI?

SQL

- SQL lets apps prepare the backend for future queries
- SQL lets apps separate concerns of imperative app logic and declarative data logic
- SQL lets app express bulk analytical queries

How can we apply this to GenAI?

- Map/Reduce
- Spans
- Dependent/independent sub-sequences

A Span Query is an expression tree over g, x, +

g: generate

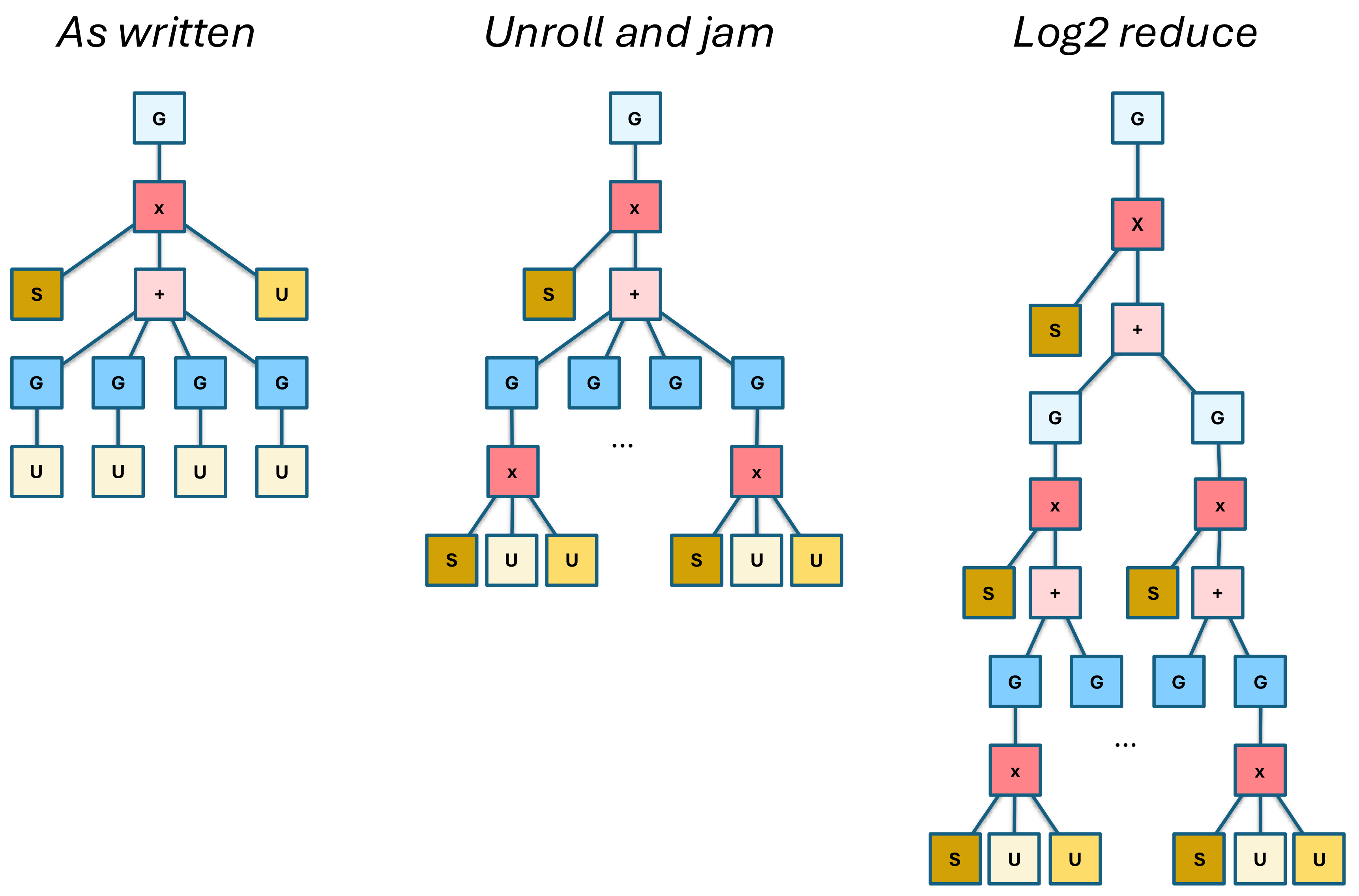
x: depends-on/attend-to

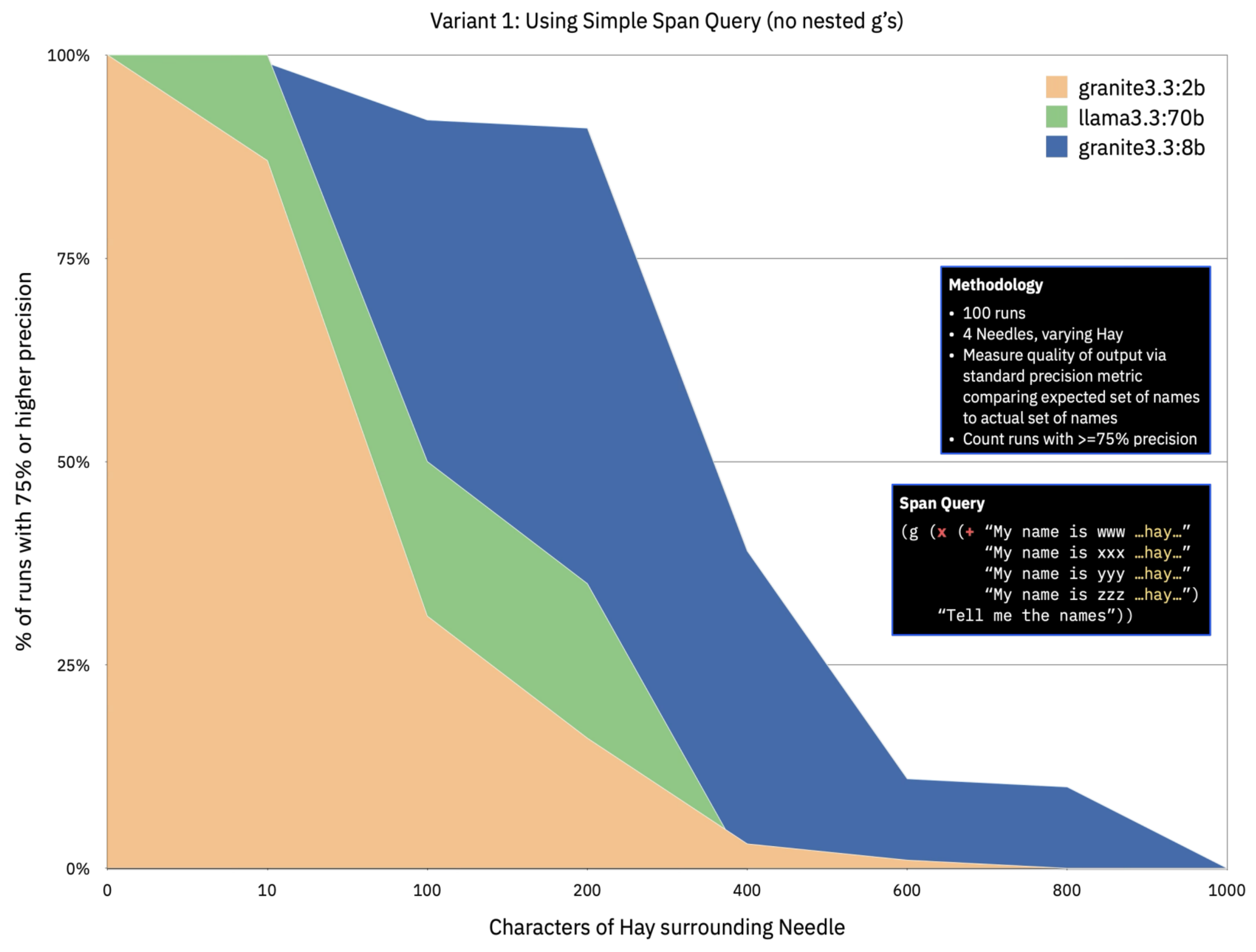
+: independent

Textual Representation *(note: not proposing as DSL)*

```
(g (x (system "A good email is…")
      (+ (g (user "an introductory email"))
         (g (user "an introductory email"))
         (b (user "an introductory email"))
         (g (user "an introductory email"))
      (user "I am applying to IBM")))
```

Tree representation



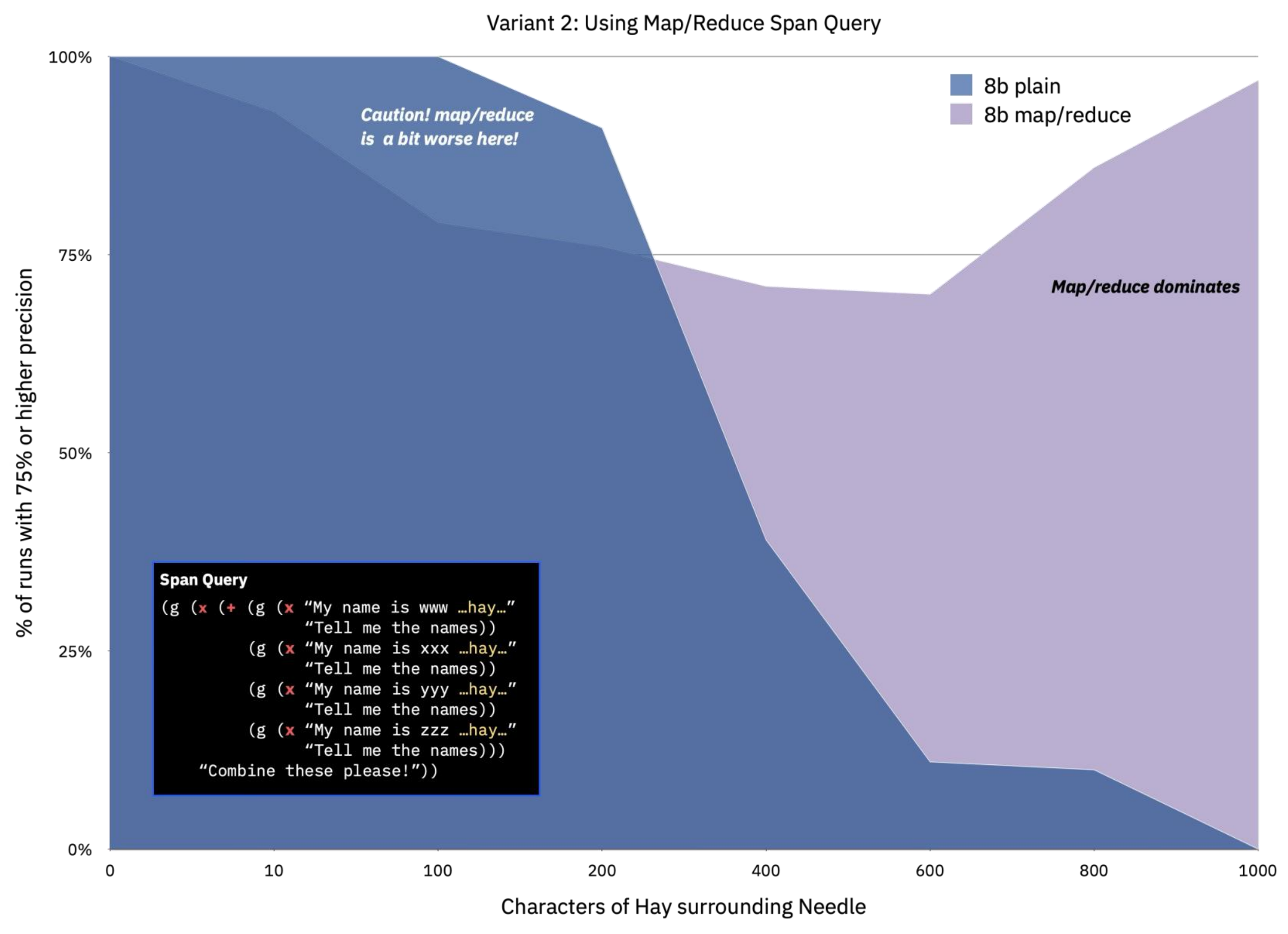

Variant 1: Using Simple Span Query (no nested g's)

Methodology

- 100 runs
- 4 Needles, varying Hay
- Measure quality of output via standard precision metric comparing expected set of names to actual set of names
- Count runs with >=75% precision

Span Query

```
(g (x (+ "My name is www …hay…"
         "My name is xxx …hay…"
         "My name is yyy …hay…"
         "My name is zzz …hay…")
      "Tell me the names"))
```

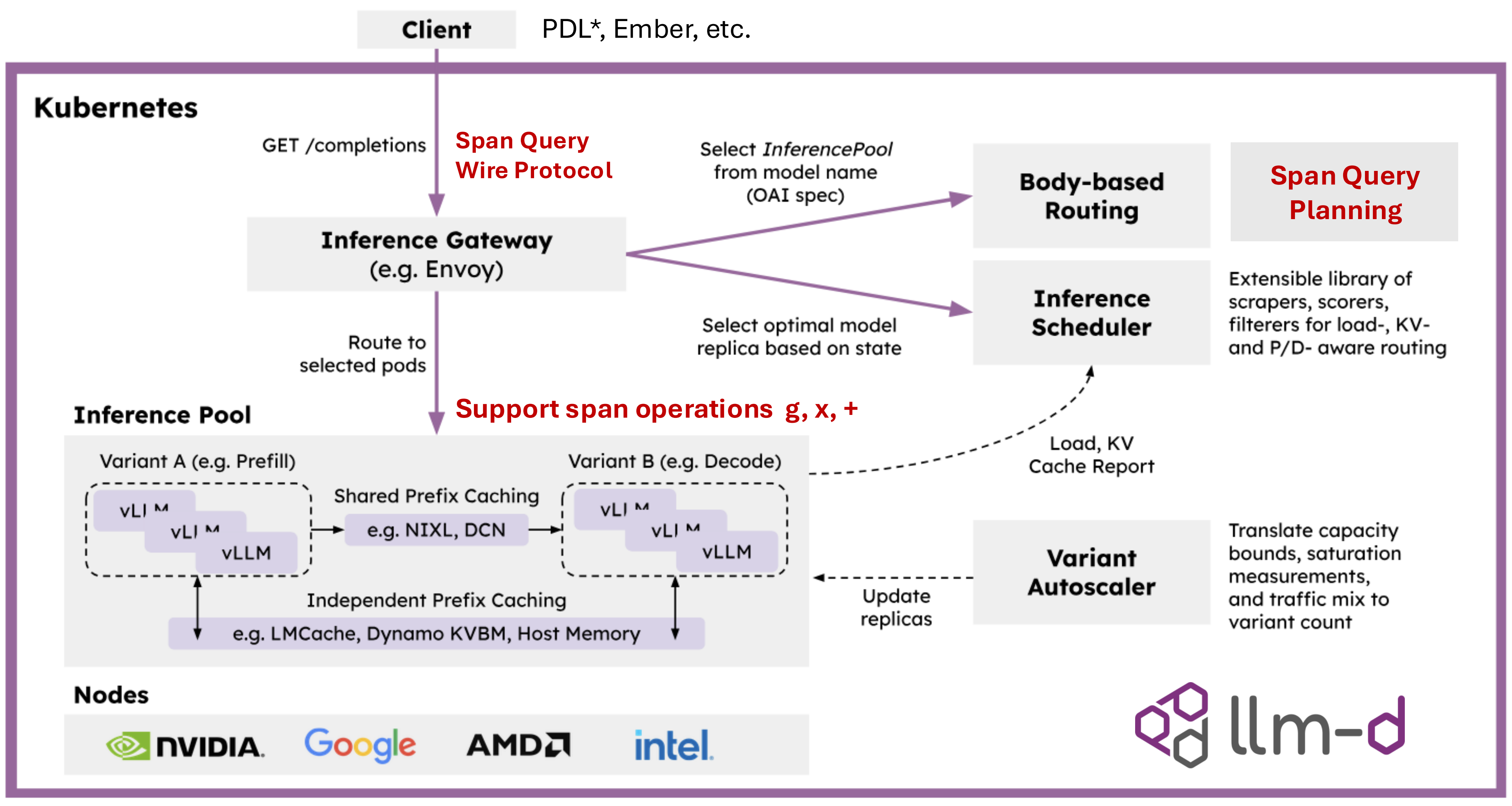

Variant 2: Using Map/Reduce Span Query

Span Query

```
(g (x (+ (g (x "My name is www …hay…"
               "Tell me the names))
         (g (x "My name is xxx …hay…"
               "Tell me the names))
         (g (x "My name is yyy …hay…"
               "Tell me the names))
         (g (x "My name is zzz …hay…"
               "Tell me the names)))
      "Combine these please!"))
```

Strike Points Across the Stack

1. How can **vLLM scale-up** better when given the dependence relations implicit in a span query?
2. How can **llm-d scale-out** better in light of a map/reduce query structure?
3. Does the backend benefit from "**prepared statements** I.e. being given, in advance, templated queries?
4. Can we **simplify client libraries** by leveraging a SQL-like separation of concerns?
5. Can we **consolidate inference scaling patterns** aroun queries? How many of them can be expressed as queries?


Client
PDL*, Ember, etc.
Kubernetes
GET /completions
Span Query Wire Protocol
Select InferencePool from model name (OAI spec)
Body-based Routing
Span Query Planning
Inference Gateway (e.g. Envoy)
Inference Scheduler
Extensible library of scrapers, scorers, filterers for load-, KV- and P/D- aware routing
Select optimal model replica based on state
Route to selected pods
Support span operations g, x, +
Inference Pool
Variant A (e.g. Prefill)
Variant B (e.g. Decode)
vLLM
Shared Prefix Caching
e.g. NIXL, DCN
Independent Prefix Caching
e.g. LMCache, Dynamo KVBM, Host Memory
Load, KV Cache Report
Update replicas
Variant Autoscaler
Translate capacity bounds, saturation measurements, and traffic mix to variant count
Nodes
NVIDIA
Google
AMD
intel
llm-d

*could be span query client syntax

References

1. Thomas Merth, Qichen Fu, Mohammad Rastegari, and Mahyar Najibi. 2024. "Superposition prompting: improving and accelerating retrieval-augmented generation". In Proceedings of the 41st International Conference on Machine Learning (ICML'24), Vol. 235. JMLR.org, Article 1445, 35507–35527.
2. Automatic Prefix Caching, https://docs.vllm.ai/en/latest/features/automatic_prefix_caching.html
3. Zheng, Lianmin, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao et al. "SGLang: Efficient execution of structured language model programs." Advances in Neural Information Processing Systems 37 (2024): 62557-62583
4. Yao, Jiayi and Li, Hanchen and Liu, Yuhan and Ray, Siddhant and Cheng, Yihua and Zhang, Qizheng and Du, Kuntai and Lu, Shan and Jiang, Junchen. "CacheBlend: Fast Large Language Model Serving for RAG with Cached Knowledge Fusion", Proceedings of the Twentieth European Conference on Computer Systems 2025 (Eurosys '25)
5. LLM-D, https://github.com/llm-d/llm-d
6. PDL https://github.com/IBM/prompt-declaration-language
7. Ember https://github.com/pyember/ember

We are grateful to Martin Hirzel, Nathan Fulton, Louis Mandel, Hendrik Strobelt, and Mandana Vaziri for inspiring this work