PARALLELPROMPT

Extracting Parallelism from Large Language Model Queries

Steven Kolawole[1], Keshav Santhanam[2], Virginia Smith[1], Pratiksha Thaker[1]

[1]Carnegie Mellon University [2]Stanford University • NeurIPS 2025

Latent Semantic Parallelism

Current LLM Systems:

- Treat prompts as monolithic
- Optimize via token-level tricks
- Optimize via batch-level tricks
- Miss intra-query structure

The Insight:

Many prompts contain **independent subtasks** that can execute *in parallel*

10.3% of real prompts

contain latent parallelism

Real User Examples from Production Logs

REPEATED GENERATION

"Generate 10 variations of detailed room descriptions..."

→ **10 independent generation calls in parallel**

READING COMPREHENSION

"Rate these sentences 1-10: 1. The book is brown. 2. The book are brown..."

→ **Independent ratings executed in parallel**

8+ canonical categories + 400+ novel patterns discovered across 11 languages

Core Feature: Structured Schema Extraction

TEMPLATE

Task structure with placeholders

CONTEXT

Shared information across subtasks

DATA or *n*

Iteration inputs (list items, sentences, etc.)

Each of 37,070 prompts is annotated with this structured representation, enabling systematic parallelization strategies

Multi-Stage Curation Pipeline

358,000 prompts

LMSYS-Chat-1M + WildChat-1M • from public logs

↓ LLM-Assisted Extraction

Identify parallelizable structure • GPT-4 with task-specific prompts

↓ Schema Generation

Extract template + context + data • Structured representation

↓ Tiered Validation

Rule-based multilingual validation • High/medium confidence filtering

37,070 validated

10.3% yield • First real-world benchmark

Tiered Validation; ensuring quality at scale

Validation Rules:

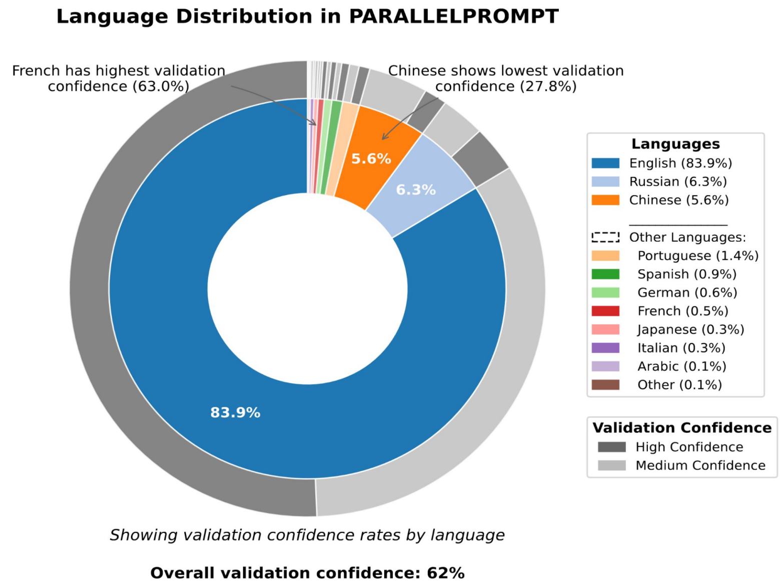
✓ Template-placeholder compatibility

✓ Data-count consistency

✓ Minimum parallelism threshold

✓ Language-specific constraints

✓ Mutual exclusivity checks



Validation Success Rates:

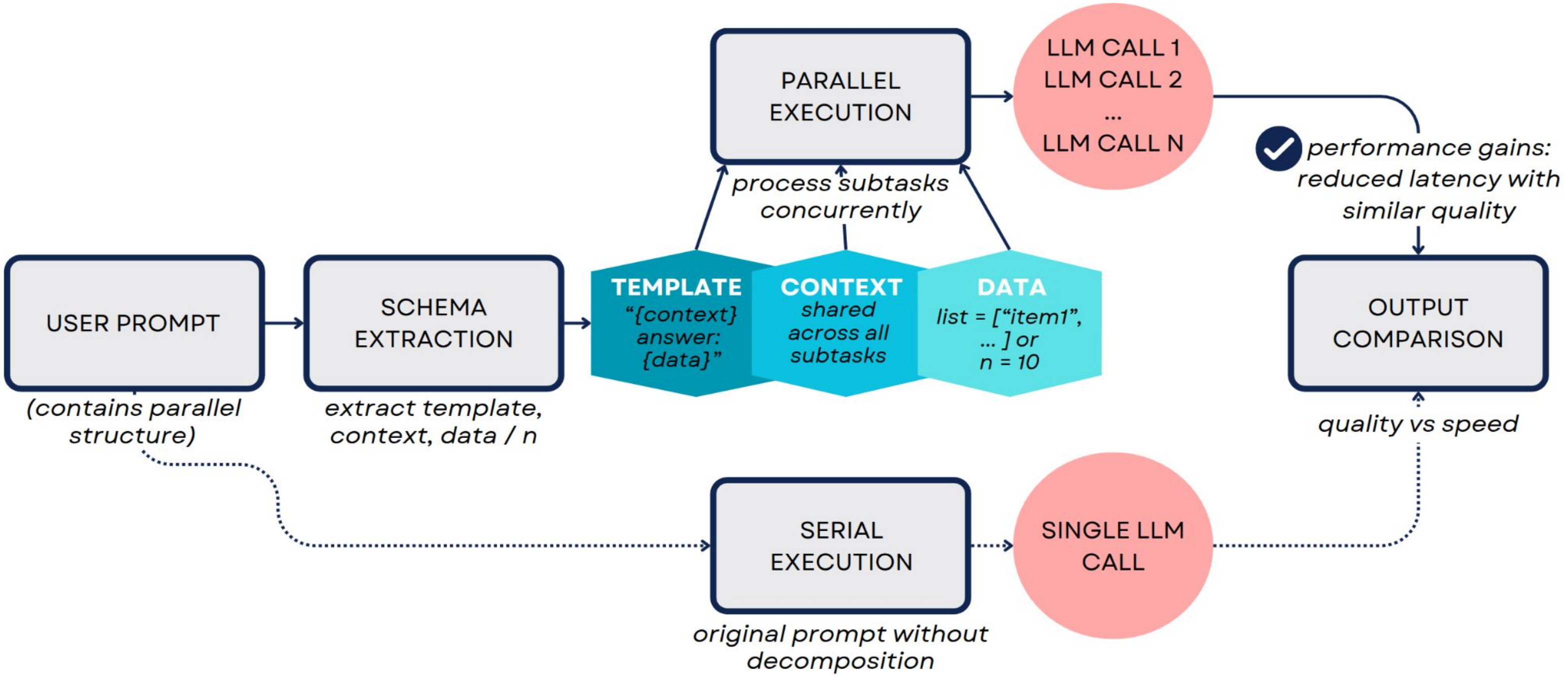
By Language:

- English: 55-63% (high)
- Chinese: 28-34% (lower)
- Japanese: 28-34% (lower)

By Category:

- Structured tasks (NER): Higher
- Creative tasks (Generation): Lower

Execution: Serial vs. Parallel Strategies



Schema decomposition → Parallel execution → Response aggregation

Performance Results: Significant Speedups

5.7×
Reading Comprehension

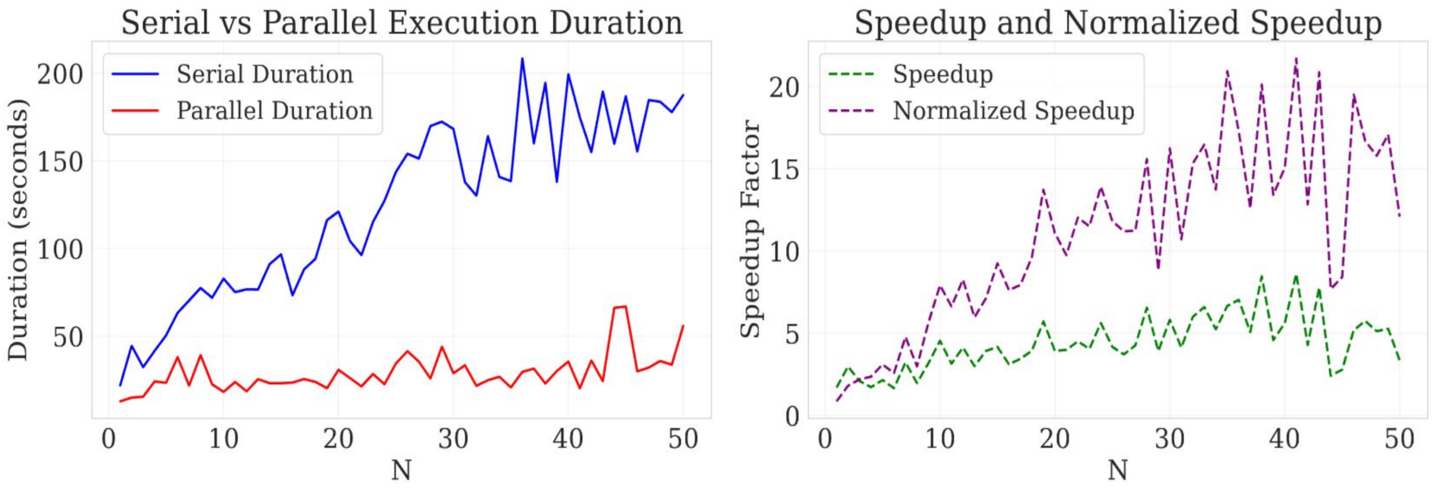
4.4×
Repeated Generation

92%
Quality Preserved

Evaluation Dimensions:

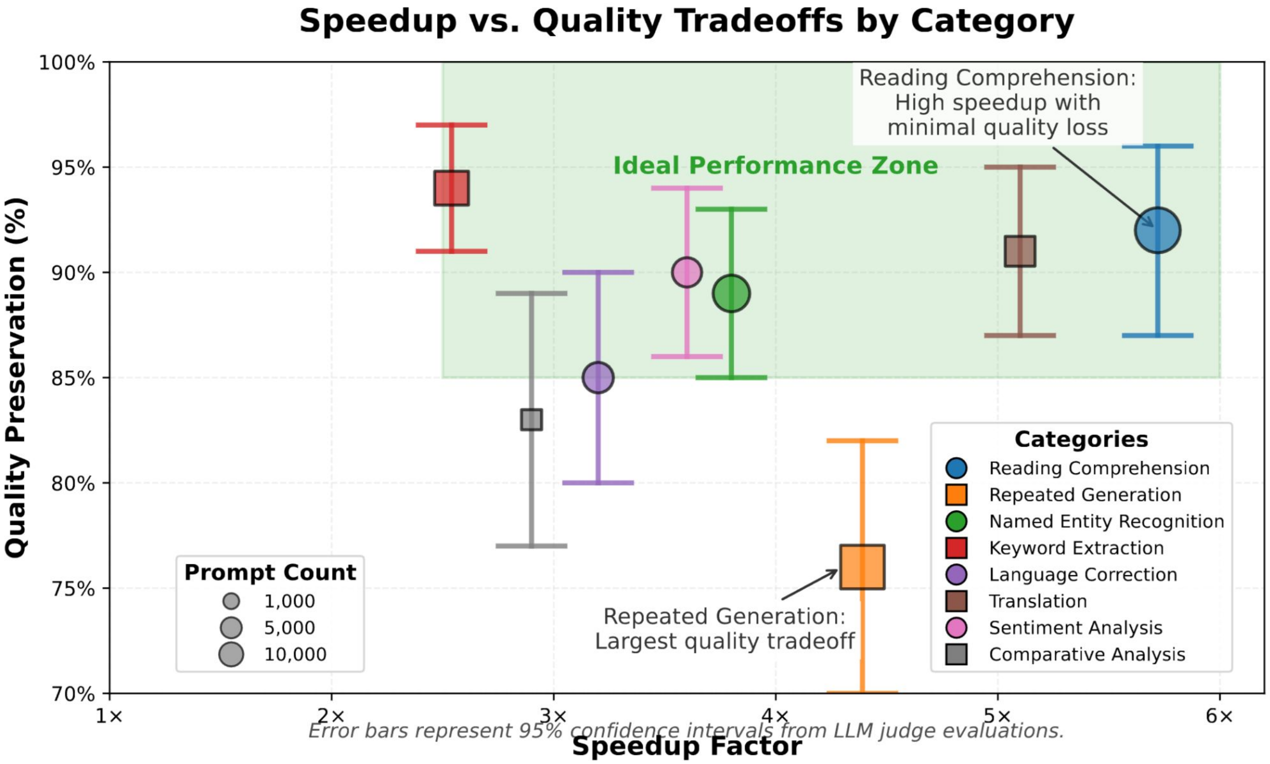
- Latency (normalized speedup)
- Structural adherence
- Semantic fidelity

Speedup Scales with Prompt Complexity



Greater parallelism opportunities → Higher speedups

Quality-Speed Tradeoffs



Minimal quality degradation across speedup ranges

Why Prior Methods Fail on Real Queries

Skeleton-of-Thought	Synthetic bullet-point outlines	Fails on natural language patterns
Tree-of-Problems	Assumes explicit problem structure	Misses implicit decomposition
PARALLELPROMPT	Real user prompts + structured schemas	**75%+ parsing success rate**

Key Insight: Real user prompts / data in the wild require robust schema extraction + multilingual validation, not heuristic pattern matching

Open Challenges & Limitations

Dependency Blindness

~25% of validated prompts have hidden dependencies between subtasks

Language-Specific Biases

Extraction methods favor Western languages (55-63% vs 28-34% success)

Creative Task Coverage

Lower validation rates for generation tasks vs. structured tasks

Future Directions:

• Task-adaptive parallelization strategies • Dependency detection models • Cross-lingual extraction robustness

Complementary to Existing Optimizations

Token-Level

Speculative decoding, KV-caching

Batch-Level

Dynamic batching, continuous batching

Query-Level

Intra-query parallelism (PARALLELPROMPT)

New optimization axis: Structure-aware execution can potentially combine with token/batch optimizations for compound speedups

Key Takeaways

1 **First Real-World Benchmark**

37,070 prompts with structured schemas across 11 languages

2 **Significant Speedups**

3-5× latency reduction with minimal quality loss

3 **New Optimization Paradigm**

Rethink execution as structured, parallelizable interface

📊 Dataset, Pipeline & Evaluation Suite available • arXiv:2506.18728