

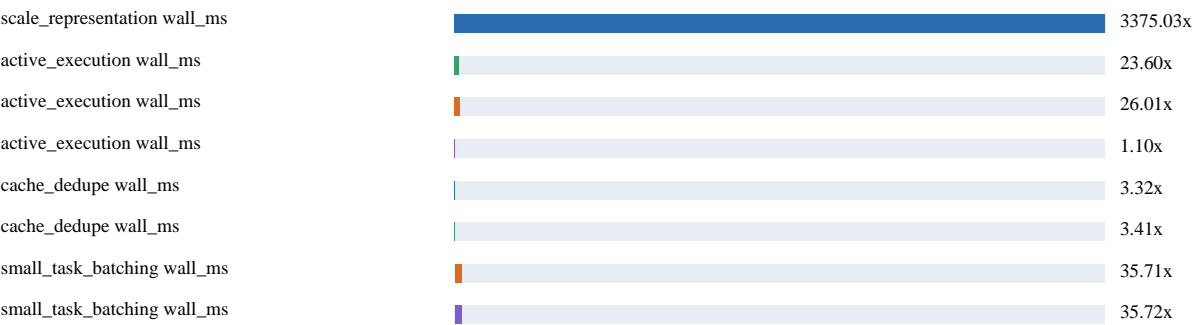
Yool Safe-Speed Benchmark V2

Comparacao entre instrucao normal, V1 high-throughput e V2 safe-speed.
Run date: 2026-05-21
Python: 3.14.3

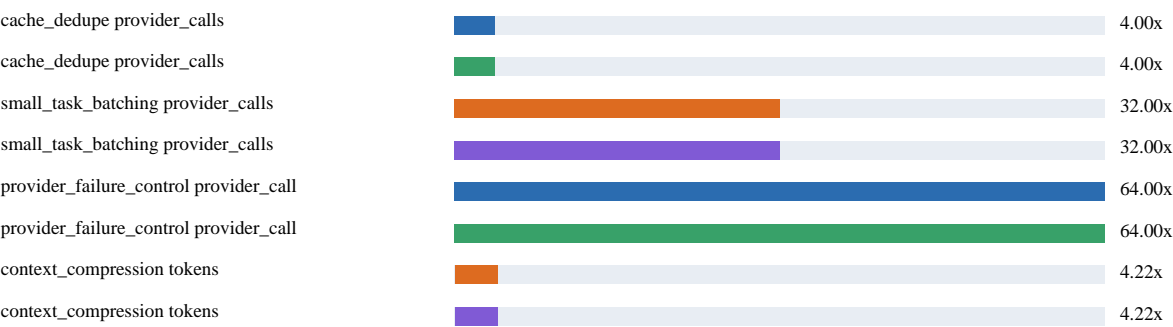
Executive Summary

A V2 mantem o ganho estrutural da V1 para escala massiva e adiciona controles seguros de velocidade: cache por input/receipt, LaneWorkerPool adaptativo, backoff com jitter, circuit breaker por provedor, batching, compressao de contexto, roteamento local e speculative execution apenas para tarefas idempotentes.

Speed gains (x)



Provider/token reduction (x)



Scale Representation

Profile	Tasks	Wall ms	Throughput/s	Calls	Cache	Blocked	Tokens
normal instruction	131,072	174.49	751175.4	0	0	0	0
V1 high-throughput	1,048,576	0.08	13851731550.8	0	0	0	0
V2 safe-speed	1,048,576	0.05	20281939643.6	0	0	0	0

Active Execution

Profile	Tasks	Wall ms	Throughput/s	Calls	Cache	Blocked	Tokens
normal instruction	1,024	5615.25	182.4	1,024	0	0	0
V1 high-throughput	1,024	237.91	4304.2	1,024	0	0	0
V2 safe-speed	1,024	215.90	4742.9	1,024	0	0	0

Cache Dedupe

Profile	Tasks	Wall ms	Throughput/s	Calls	Cache	Blocked	Tokens
normal instruction	256	417.68	612.9	256	0	0	0
V1 high-throughput	256	429.38	596.2	256	0	0	0
V2 safe-speed	256	125.84	2034.4	64	192	0	0

Small Task Batching

Profile	Tasks	Wall ms	Throughput/s	Calls	Cache	Blocked	Tokens
normal instruction	512	729.98	701.4	512	0	0	0
V1 high-throughput	512	730.34	701.0	512	0	0	0
V2 safe-speed	512	20.44	25043.8	16	0	0	0

Provider Failure Control

Profile	Tasks	Wall ms	Throughput/s	Calls	Cache	Blocked	Tokens
normal instruction	64	0.02	3062203.2	192	0	0	0
V1 high-throughput	64	0.01	7529406.0	192	0	0	0
V2 safe-speed	64	0.51	126033.9	3	0	63	0

Context Compression

Profile	Tasks	Wall ms	Throughput/s	Calls	Cache	Blocked	Tokens
normal instruction	1	0.10	10090.8	0	0	0	5,016
V1 high-throughput	1	0.07	13440.9	0	0	0	5,016
V2 safe-speed	1	0.16	6265.7	0	0	0	1,188

Gains Table

Scenario	Baseline	Improved	Metric	Ratio	Gain
scale_representation	normal instruction	V2 safe-speed	wall_ms	3375.03x	99.97%
active_execution	normal instruction	V1 high-throughput	wall_ms	23.60x	95.76%
active_execution	normal instruction	V2 safe-speed	wall_ms	26.01x	96.16%
active_execution	V1 high-throughput	V2 safe-speed	wall_ms	1.10x	9.25%
cache_dedupe	normal instruction	V2 safe-speed	wall_ms	3.32x	69.87%
cache_dedupe	normal instruction	V2 safe-speed	provider_calls	4.00x	75.00%
cache_dedupe	V1 high-throughput	V2 safe-speed	wall_ms	3.41x	70.69%
cache_dedupe	V1 high-throughput	V2 safe-speed	provider_calls	4.00x	75.00%
small_task_batching	normal instruction	V2 safe-speed	wall_ms	35.71x	97.20%
small_task_batching	normal instruction	V2 safe-speed	provider_calls	32.00x	96.88%
small_task_batching	V1 high-throughput	V2 safe-speed	wall_ms	35.72x	97.20%
small_task_batching	V1 high-throughput	V2 safe-speed	provider_calls	32.00x	96.88%

provider_failure_control	normal instruction	V2 safe-speed	provider_calls	64.00x	98.44%
provider_failure_control	V1 high-throughput	V2 safe-speed	provider_calls	64.00x	98.44%
context_compression	normal instruction	V2 safe-speed	tokens	4.22x	76.32%
context_compression	V1 high-throughput	V2 safe-speed	tokens	4.22x	76.32%

Interpretation

V1 ja resolve escala com lazy batch_spawn. V2 melhora a velocidade real de entrega ao reduzir chamadas repetidas e overhead de orquestracao. O circuit breaker e o backoff sao importantes porque velocidade sem controle aumenta risco de rate limit ou bloqueio por provedor. A V2 acelera evitando trabalho, nao forçando chamadas infinitas.

Limitations

Os numeros medem mecanica local. Provedores reais podem variar por rate limit, latencia, tamanho de contexto e politica de cache. Para trabalho CPU-bound puro em Python, subprocessos ou extensoes nativas ainda sao melhores que threads por causa do GIL.