

ArgoCon
NORTH AMERICA

Cache Me If You Can

Optimizing Expensive Tasks in Argo DAGs

Chetan Sharma

ArgoCon North America 2026

About the Speaker



Chetan Sharma

Senior Software Engineer, Google

I specialize in large-scale data processing systems, including streaming and batch pipelines, and data storage infrastructure.

Today's flow

ArgoCon NORTH AMERICA

FROM REPEATED WORK TO PRACTICAL CACHE DECISIONS

01

The reuse problem

Different requests. Repeated intermediate work.

02

Our research

What we learned across documents, slides and video.

03

Argo integration

Native memoization, safe reuse and coordination.

04

What we measured

Hit value, lookup overhead and retention under pressure.

05

Practical choices

Choose an approach by the work it actually avoids.

What is worth keeping—and where should reuse happen?

One document, several questions

ArgoCon NORTH AMERICA

REUSE PATTERN / ONE SOURCE, DIFFERENT QUESTIONS

SOURCE DOCUMENT

Research services invoice

doc-000 · unchanged input

Supplier	ABC Company
Currency	USD
Payment terms	Thirty days after receipt

REUSABLE STAGES

Shared document processing

01 **OCR**

02 **Parse**

03 **Embed**

Reuse when inputs match

QUESTION-SPECIFIC WORK

Two different requests

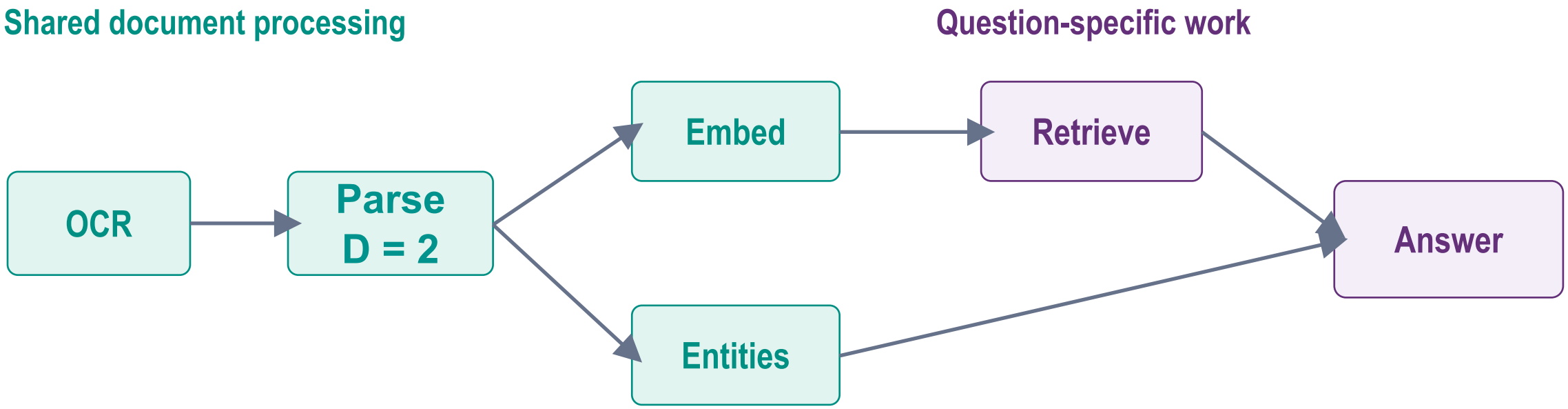
Q1 **What is the supplier name?**
ABC Company

Q2 **What are the payment terms?**
Thirty days after receipt

Different questions can reuse the same document-level work.

Reuse inside an Argo DAG (Directed Acyclic Graph)

Nodes are tasks. Arrows are dependencies. D = number of immediate downstream consumers.



An intermediate DAG result can be reused even when the final question changes.

An expensive result can be evicted too early

ArgoCon NORTH AMERICA

Illustrative two-slot cache with equal-size results

1 · CACHE FULL

Older OCR output
Expensive to recreate

Recent parse output
Cheap to recreate

2 · CHEAP RESULT ARRIVES

New cheap output C

↓

LRU removes older OCR output

3 · OCR RETURNS

CACHE MISS

↓

OCR runs again

LRU remembers recency, not the recomputation penalty of the next miss.

What makes a result worth keeping?



ArgoCon
NORTH AMERICA

THREE INPUT SIGNALS

BOUNDED RESULT CACHE

D **Downstream impact**

Immediate consumers in the supplied DAG

C **Recomputation cost**

Observed stage execution time

F **Invocation frequency**

Calls to the same task type

COMBINE AT INSERTION

RETENTION POLICY

Score each completed result

$$S = D + 2C + 0.01F$$

Higher score = greater retention priority

When the payload budget fills, remove lower-priority results.

Research across three workloads

ArgoCon NORTH AMERICA

OUR INITIAL RESEARCH · TASK-RESULT CACHING IN PLAN–ACT WORKFLOWS

MP-DocVQA

MULTIPAGE DOCUMENT QA

Evidence spans document pages.

1–20 pages per document

Highly variable reuse

1–85 questions per document can revisit source processing.

SlideVQA

PRESENTATION QA

Questions refer to slide decks.

1–13 questions per deck

Moderate reuse

Related questions revisit the same deck; answers differ.

VideoMME

VIDEO UNDERSTANDING

Visual and temporal questions.

3 questions per video

Limited reuse

A short reuse window before the next video arrives.

Equal finite capacity within each benchmark • Latency + reuse behavior + answer quality

Preprint: Workload-Aware Caching for Multi-Agent Systems · **arXiv:2607.20495**

Lower latency across workloads



ArgoCon
NORTH AMERICA

OUR RESEARCH · MEAN QUERY LATENCY / LRU LATENCY · LRU = 100% · LOWER IS BETTER

MP-DocVQA

41.8% lower than LRU



SlideVQA

51.3% lower than LRU



VideoMME

19.9% lower than LRU



Lowest latency among the finite policies tested. Answer quality is evaluated separately.

Preprint: Workload-Aware Caching for Multi-Agent Systems · **arXiv:2607.20495**

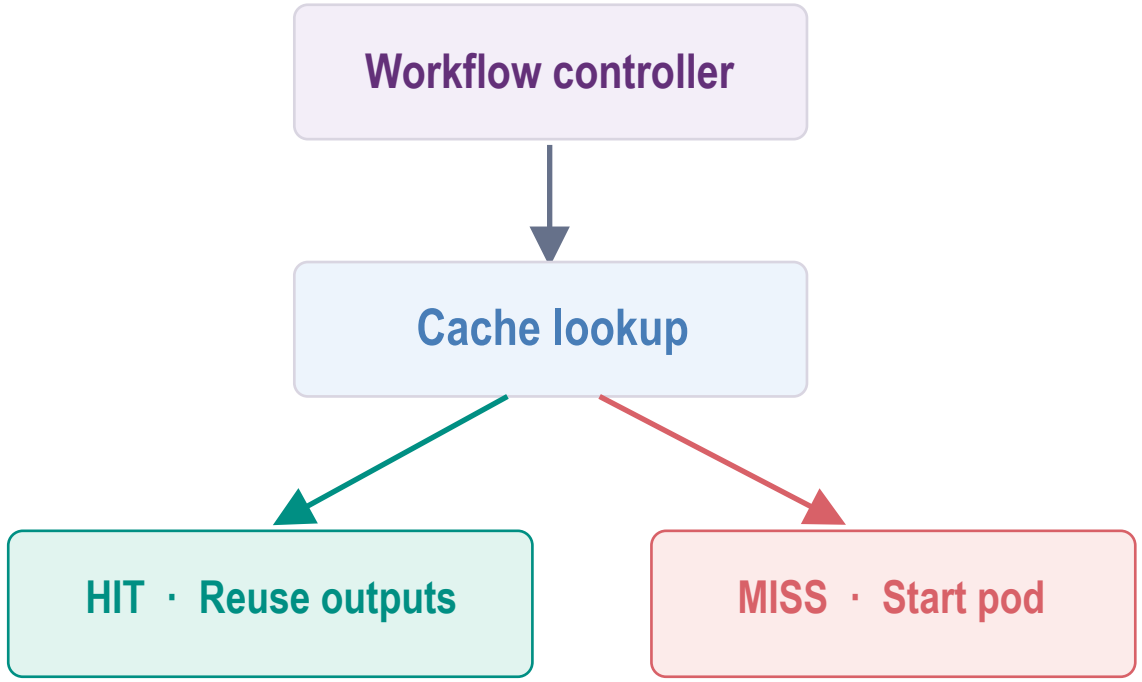
Native Argo memoization



The Argo controller checks the key before creating the task pod.

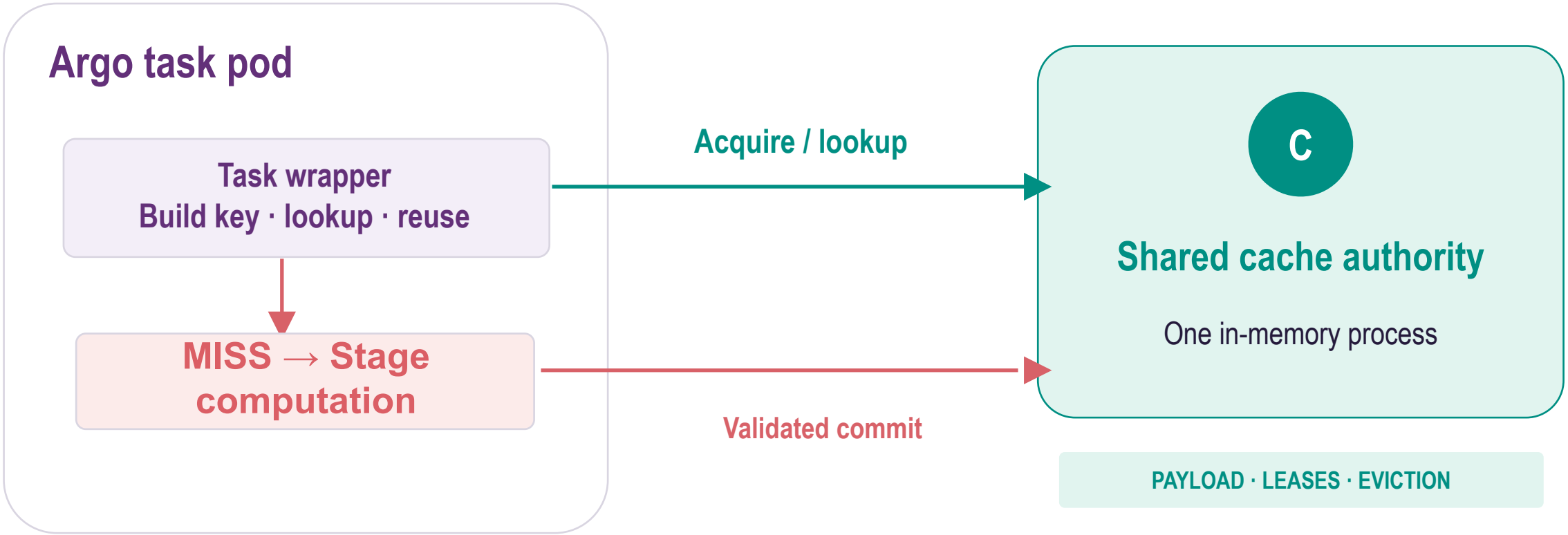
TEMPLATE MEMOIZATION

```
memoize:
  key:
"{{inputs.parameters.cache_key}}"
  maxAge: "24h"
  cache:
    configMap:
      name: stage-results
```



A native HIT can skip the task pod altogether.

A shared bounded cache for task outputs



External HIT: the task pod already exists when the lookup happens.

Which outputs are safe to reuse?

ArgoCon NORTH AMERICA

CORRECT REUSE NEEDS THE RIGHT IDENTITY, INTACT OUTPUTS AND AN ELIGIBLE OPERATION

Source + revision

Exact immutable source content

Parent outputs

Hashes of upstream result content

MATCH KEY + VERIFY OUTPUT

Content hashes check the bytes—not whether the answer is true.

Code + model

Implementation and model revision

Inputs + sharing

Relevant inputs; production requires authorized sharing

HIT

Matching full key + valid stored result → reuse

MISS

No valid matching result → compute

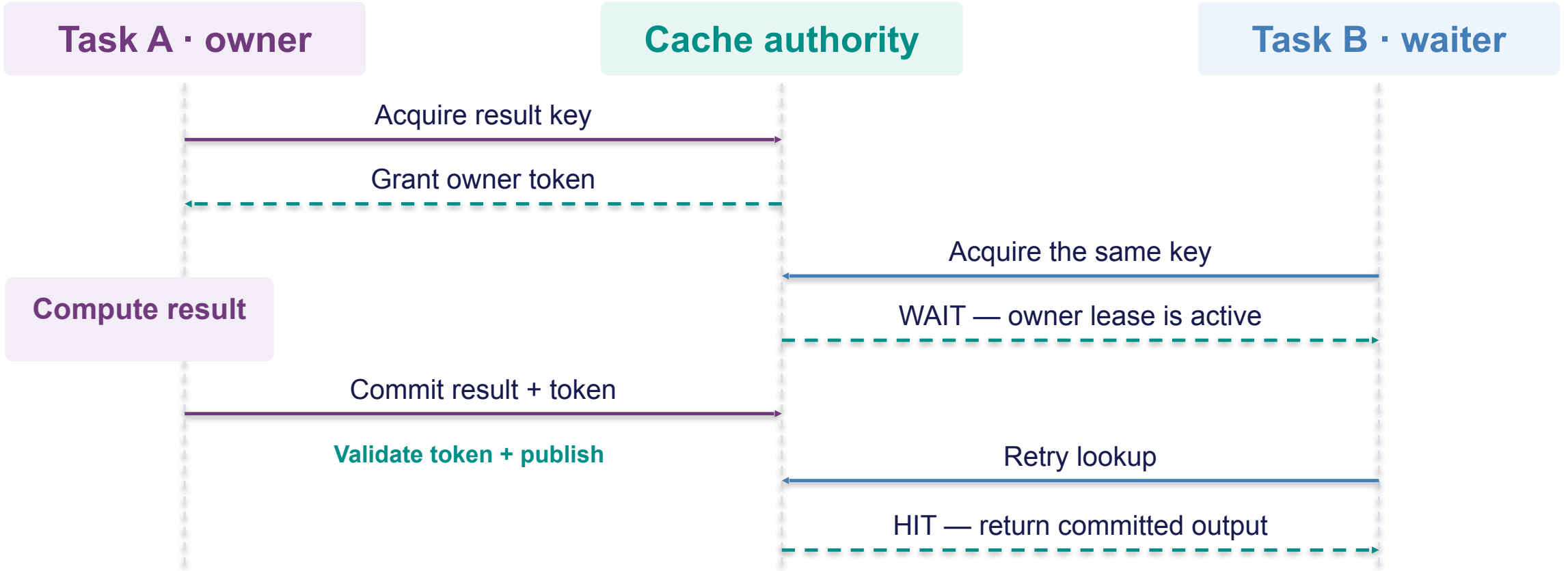
BYPASS

Required side effect → execute

Cache correctness and final-answer quality are different checks.

Concurrent requests for one result

SAME RESULT KEY · ONE OWNER LEASE · WAITERS RETRY AFTER PUBLICATION



One owner computes; waiters reuse the result after a valid commit.

Three questions about reuse

ArgoCon NORTH AMERICA

01

Which hits are valuable?

Models-loaded CPU worker

Models stay in memory
No per-stage pod startup

Query latency
Expensive recomputations

02

What does a hit avoid?

Real Argo DAG

No evictions
Matching reuse outcomes

Workflow duration
Task-pod creation

03

How long must results survive?

Controlled-delay study

Fixed request releases
Different miss penalties

Workflow duration
Retained payload

What to keep. Where to reuse it. How long it must remain available.

What matters more: cache hits or costly misses?

MODELS-LOADED CPU WORKER · NO PER-STAGE POD STARTUP · SAME BOUNDED PAYLOAD CAP

35.7%

lower mean paired query latency vs LRU

52.4%

fewer OCR computations versus LRU

Despite a lower stage-hit rate

Reused stage results ÷ all requested stages

LRU	Workload-aware
21.4%	19.0%

Each stage counts once—not by its cost.
This is not answer accuracy.

Fewer hits can still save more expensive work.

Same reuse. Different workflow overhead.

ArgoCon NORTH AMERICA

REAL CPU ARGO DAG · MATCHING REUSE · NO EVICTIONS

Same request trace • Same reused stages • Same stage computations

Mean workflow duration

External cache = 100% · Lower is better



32.3% lower workflow duration

Task pods created

External cache = 100% · Lower is better



50% fewer task pods

The same reuse does not guarantee the same execution overhead.

A hit can skip computation—or its task pod

ArgoCon NORTH AMERICA

REPEATED-DOCUMENT OCR CHECK · NO EVICTIONS IN EITHER CACHE

First request in every configuration: start the task pod and compute OCR.

No cache

ON THE REPEATED REQUEST

Start task pod

↓

Run OCR again

↓

Return result

Nothing is skipped

External cache

ON THE REPEATED REQUEST

Start task pod

↓

Cache HIT

↓

Reuse result

OCR computation skipped

Native memoization

ON THE REPEATED REQUEST

Controller HIT

↓

Reuse recorded output

↓

Task pod skipped

OCR + task pod skipped

A hit only avoids work that has not already happened.

What if expensive results must survive a gap?

SEPARATE RETENTION STUDY · FIXED REQUEST RELEASES

Controlled delays represent recomputation cost in real Argo workflows.

A	→ Other outputs	→ A returns	→ More outputs	→ A returns
Expensive result	Cheap arrivals	Reuse opportunity	Cheap arrivals	Reuse opportunity

Native idle cleanup

Time-based removal
Observed below the ceiling

External workload-aware

Priority-based retention
Hard result-payload limit

Native cleanup OFF

Longer-retention reference
May exceed the ceiling

Can keeping A until its next use outweigh the external cache's overhead?

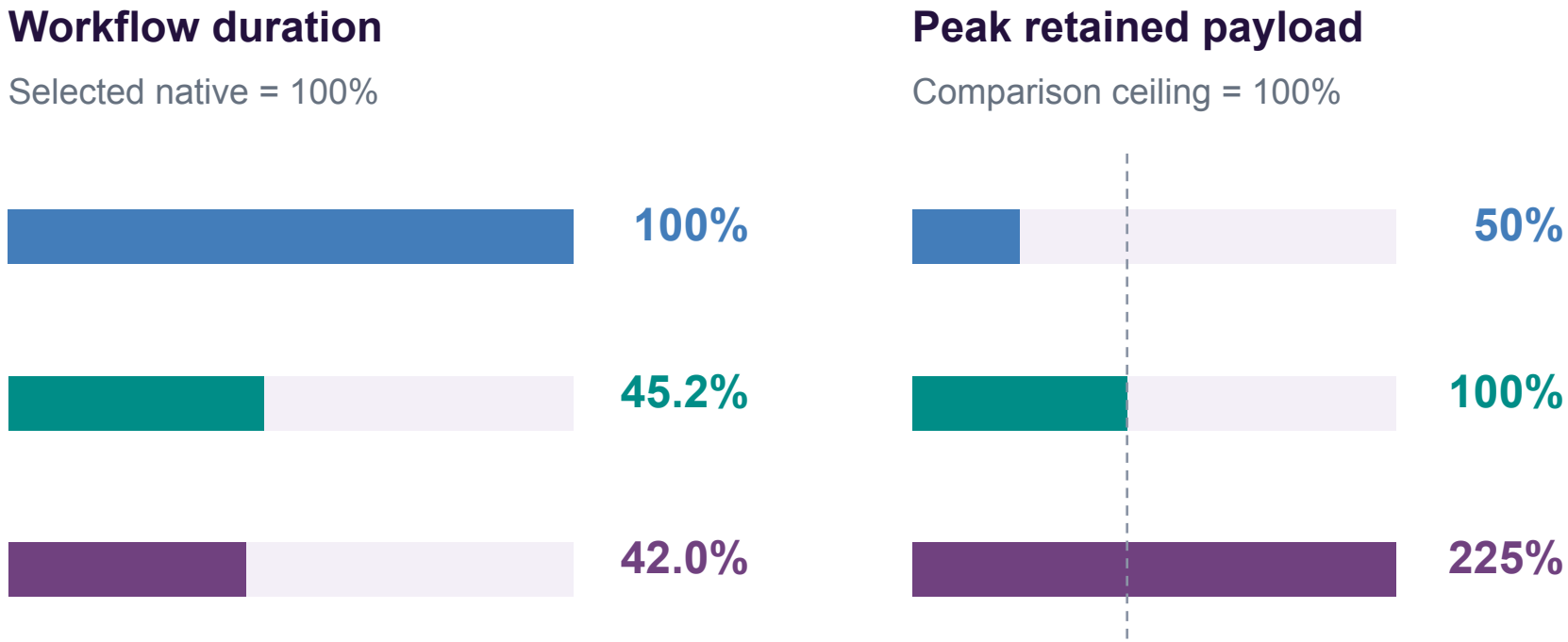
Keeping the costly result paid off under this ceiling

ArgoCon NORTH AMERICA

54.8%

lower mean workflow duration versus selected native idle cleanup

Retention configuration



Keeping everything was faster—but exceeded the payload ceiling.

When misses get cheaper, the benefit shrinks

CONTROLLED MISS-COST COMPARISON IN ARGO

Workload-aware improvement versus selected native idle cleanup

Higher miss penalty

PRIMARY STUDY

54.8% lower mean workflow duration

15.2% lower scheduled makespan

Lower miss penalty

SENSITIVITY CHECK

12.8% lower mean workflow duration

0.6% lower scheduled makespan

Fixed releases: finishing earlier does not bring the next request forward.

The value of retention depends on the cost of losing a result.

Choose the cache around the work it avoids



ArgoCon
NORTH AMERICA

AN EVALUATION SEQUENCE FOR YOUR WORKFLOW

01 **Start with safe reuse**

Equivalent inputs, valid outputs and permitted sharing.

02 **Measure native memoization**

A controller-level hit can avoid the task pod.

03 **Test bounded retention when space matters**

Look for costly results that return after intervening work.

04 **Validate the complete workflow**

Measure latency, expensive misses, task pods and payload.

A warm-process improvement does not guarantee a faster Argo workflow.

Three lessons about useful reuse



ArgoCon
NORTH AMERICA

01

Count the work saved—not just the hits

Avoiding an expensive miss can matter more than several cheap hits.

02

Check early enough to skip meaningful work

Reusing a result inside a pod is different from avoiding that pod.

03

Retain results that are costly and return

High recomputation cost alone does not guarantee future value.

Measure value, placement and retention together.

ArgoCon
NORTH AMERICA

Thank you!

Cache Me If You Can

Optimizing expensive tasks in Argo DAGs

Chetan Sharma

Senior Software Engineer, Google

Presentation slides

Add public slides URL before sharing

Appendix · Research across policies



ArgoCon NORTH AMERICA

Query latency relative to LRU = 100% · Lower is better

Policy	MP-DocVQA	SlideVQA	VideoMME
No cache	134.2%	138.3%	117.2%
FIFO	104.2%	105.6%	92.4%
ARC	81.5%	92.7%	93.6%
LFU	76.9%	71.1%	92.4%
LRU	100.0%	100.0%	100.0%
Workload-aware	**58.2%**	**48.7%**	**80.1%**
Unbounded	57.1%	47.1%	70.0%

Finite policies use equal capacity within each benchmark; unbounded is a reference.

Appendix · Models-loaded CPU percentage results by seed



ArgoCon NORTH AMERICA

Paired within-seed query-latency reduction vs LRU

Seed	No cache	Cost-only	Workload-aware	Unbounded
1	-5.7%	53.7%	53.2%	64.4%
2	-11.0%	48.6%	48.5%	59.3%
3	-32.6%	34.7%	32.2%	44.8%
4	-59.6%	9.2%	9.2%	28.1%
5	-18.5%	30.9%	35.5%	52.5%
Mean paired	**-25.5%**	**35.4%**	**35.7%**	**49.8%**

The headline uses mean within-seed percentages—not a ratio of pooled means.

Appendix · Bounded real-DAG results and their limits



ArgoCon
NORTH AMERICA

SEPARATE BOUNDED CPU DAG COMPARISON · ONE REQUEST TRACE

Duration: no cache = 100% | Task pods and evictions are counts

Configuration	Duration	Task pods	Evictions	Storage regime
No cache	100.0%	48	—	Not applicable
Native memoization	66.8%	28	0	Not byte-matched
External LRU	92.8%	48	10	Same hard payload cap
External cost-only	94.6%	48	23	Same hard payload cap
External workload-aware	**98.2%**	**48**	**29**	**Same hard payload cap**

5.8%

higher duration than LRU

Workload-aware in the same external-cache setup

66.9%

of the payload cap retained

Two costly outputs for a document that did not return

Expensive to recreate does not automatically mean valuable to retain.

Appendix · Retention repetitions and controls



ArgoCon NORTH AMERICA

Controlled waits: 120 s → 5 s expensive; 0.2 s cheap. Results: 2 KiB each; ceiling: 8 KiB.

Reductions versus selected native idle cleanup · 11 workflows per arm per repetition

Repetition	Aware duration	Native OFF duration	Aware makespan
1	55.0%	57.9%	15.2%
2	54.6%	58.0%	15.1%
3	54.7%	58.3%	15.2%

Peak payload: selected native 50% · aware 100% · native cleanup OFF 225%

Low-cost control (1 repetition): 12.8% lower mean duration · 0.6% lower makespan