# From Lookup to Reasoning

*An allocation-aware resource advisor for Chameleon, and a two-wing benchmark that tests it*

Vivek Rai
OSRE 2026 Fellowship
UC Santa Cruz OSPO · Summer of Reproducibility

Project: Chameleon Cloud
Mentor: Paul Marshall
Final review · 24 August 2026

# What researchers use a testbed for



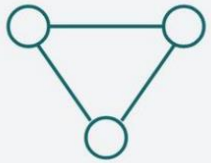
### Compute at scale

Train and evaluate on GPUs nobody owns individually



### Edge and sensing

Cameras, sensors and small devices, on real hardware



### Systems research

Reconfigure the network, the kernel, the whole stack

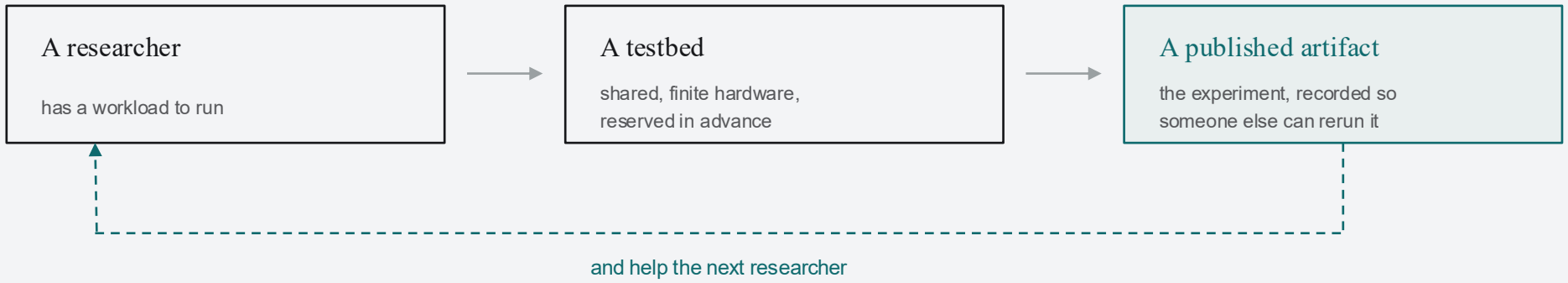


### Reproducibility

Rerun someone else's experiment, exactly

Chameleon is one such testbed: publicly funded, open to researchers, and the one this work is built on.

# One way researchers use a shared testbed



# But first, every experiment starts with the same five decisions

| DECISION | EXAMPLE |
| --- | --- |
| Which device or node type | `raspberrypi4-64, gpu_a100_pcie` |
| Is one free right now | 22 of 36 free |
| Which image and runtime | `arm64, runtime=nvidia` |
| How the lease reaches the container | `reservation_id=` |
| For how long, and how many | 2 hours, 1 device |

- None of this is the science. It is what stands in front of it.
- A device that fits the job may still be reserved by someone else, or powered down for maintenance.

# This work is on Chameleon, and starts from its documentation assistant

WHAT CHAMELEON PROVIDES

- Bare-metal and edge hardware across seven sites, reserved in advance.
- **Trovi**, where a researcher publishes the artifact for an experiment so others can rerun it.
- A **documentation assistant**, deployed today, answering testbed questions from the docs.

WHAT THAT ASSISTANT CANNOT DO

1. **Name the hardware.** No inventory of the device or node types that exist.
2. **See what is free.** No view of current reservations.
3. **Point to real experiments.** It reads documentation, never the artifacts.

So it cannot answer what a user actually asks: *what should I reserve, and for how long?*

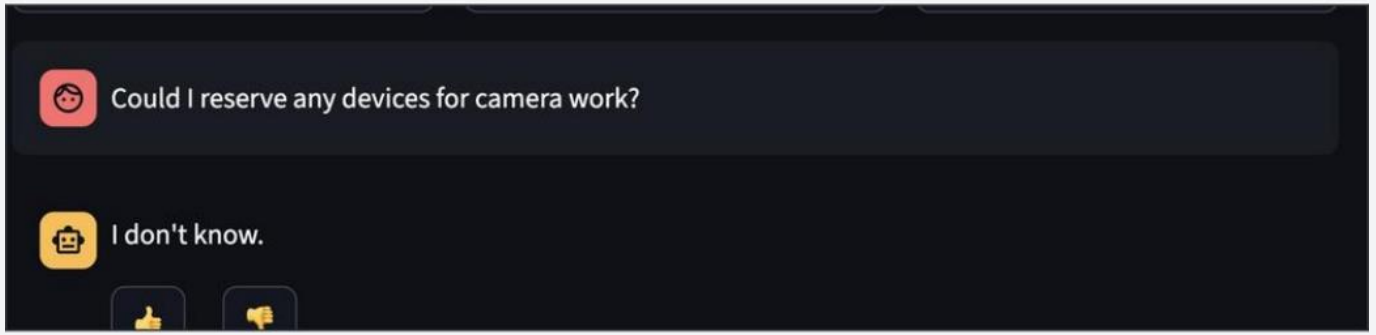
DEFINITION

**Trovi:** Chameleon's public repository of experiment artifacts. A researcher who finishes an experiment publishes it here so others can rerun it.

DEFINITION

**Bare metal vs edge:** Two kinds of Chameleon hardware. Bare metal is a whole server, held exclusively. Edge (CHI@Edge) is small devices such as Raspberry Pis and Jetsons, reserved and driven as containers. Each has its own reservation call, and the two are not interchangeable.

# Asked exactly that question, the deployed system says nothing




---

*"Given a plain-language workload, name hardware that can **do the work**, at a site that carries it, that is **free now**, and emit a reservation that submits."*

# Contribution 1 of 3

1 **Hardware and live state**

what exists, and what is free

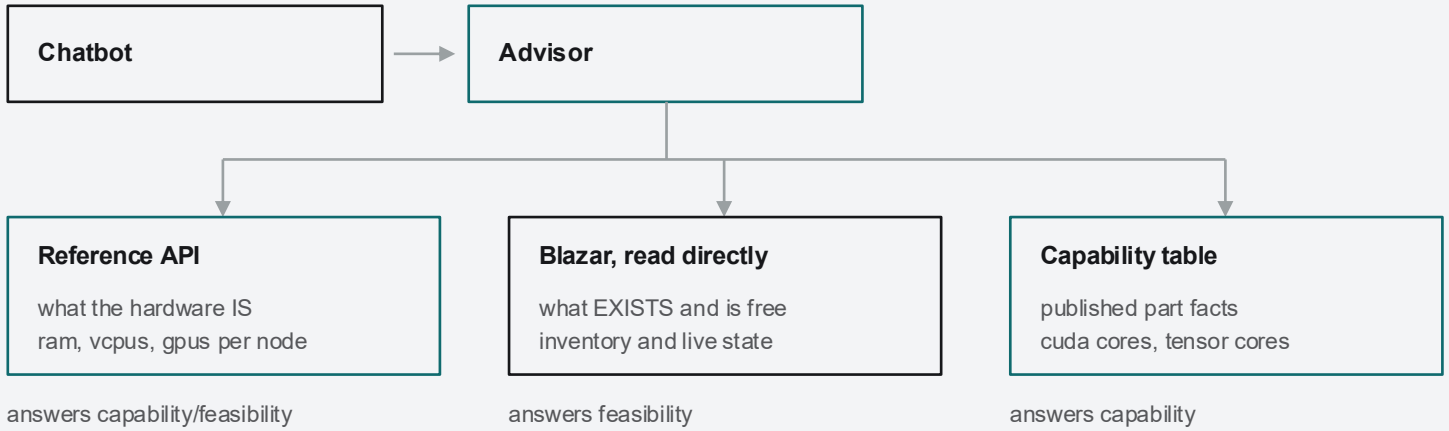
2 **Artifacts**

what real experiments actually ran on

3 **A benchmark, in two wings**

so we can tell whether any of it helps

# Surveying what hardware exists, and what is free




- **The advisor answers on the chatbot's behalf.** It fires only on resource questions; everything else stays on the documentation path.
- **The three sources are kept apart on purpose.** Every row of the capability table records which of them it came from, so a reader can tell a measurement from a published spec.
- **CHI@Edge is not in the reference API.** It lists zero edge devices, so edge inventory and availability are read from Blazar directly.

DEFINITION

**Blazar:** OpenStack's reservation service, and the component Chameleon uses to hold hardware over a time window. Asking Blazar is how you learn what is free right now, as opposed to what exists.

DEFINITION

**Capability vs feasibility:** Two different questions about one device. Capability is whether it can do the job at all; feasibility is whether you can get it right now. A device can be perfectly capable and completely unavailable.

# Contribution 2 of 3

1 **Hardware and live state**

what exists, and what is free

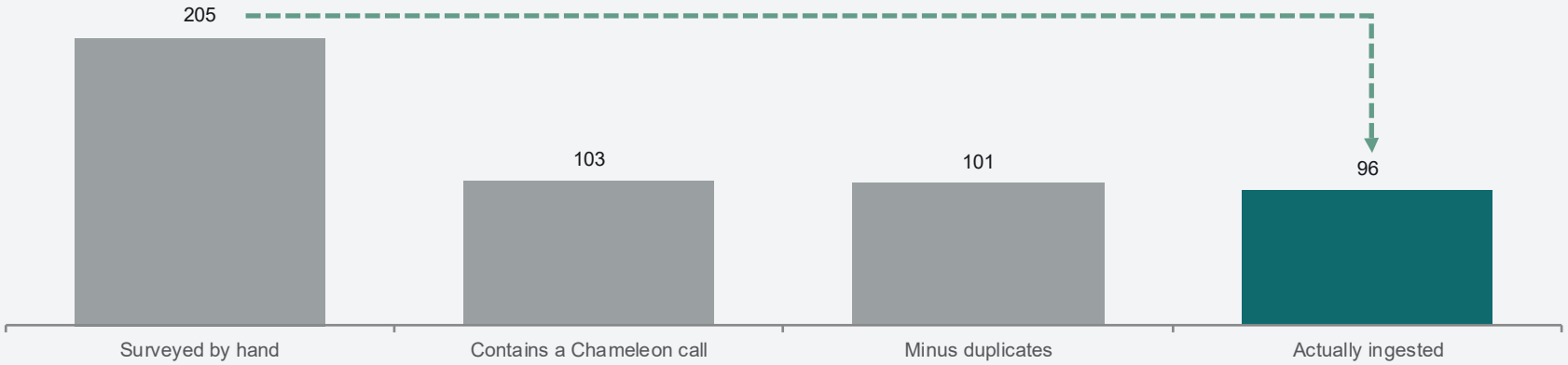
2 **Artifacts**

what real experiments actually ran on

3 **A benchmark, in two wings**

so we can tell whether any of it helps

# Which artifacts we have surveyed so far




INCLUDED IF

- Its Python parses, and the syntax tree contains a call to a Chameleon provisioning API: `add_device_reservation`, `add_node_reservation`, `create_server`.
- A definition such as `def create_server(...)` does not count.

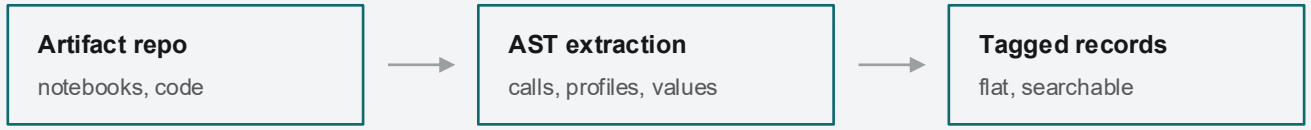
WHAT THE CORPUS NOW GROUNDS

- **The edge part** is grounded on the ingested **6 CHI@Edge artifacts**.
- **The bare-metal part** is grounded on **90 artifacts** real bare-metal and KVM notebooks and repos together with 14 recorded of testbed state and a 29-row capability table.
- The advisor reads that corpus at question time and keeps its own notes on the use cases it is asked about, so the grounding it draws on grows with use.

# How artifacts are ingested




Chunking an artifact buries those few lines among everything else the experiment does.

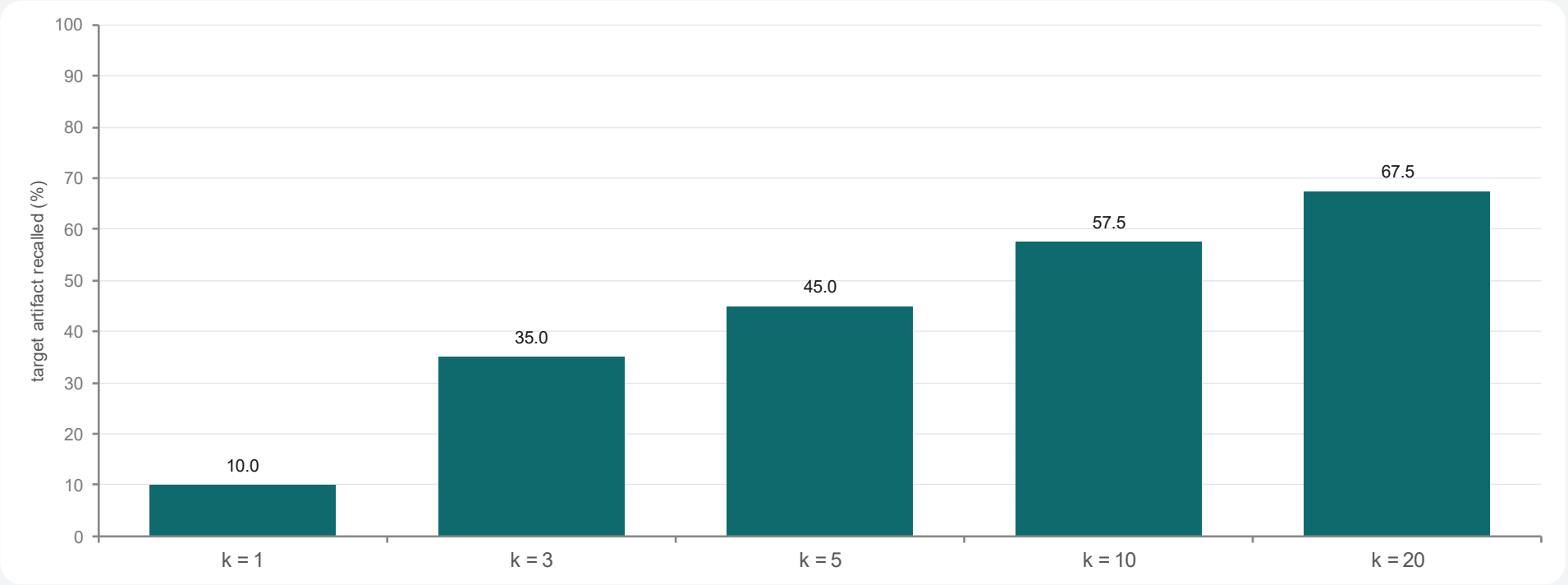



Tags record which site an artifact ran on and which API generation it used, so retrieval can filter on both.

- Most of an artifact is the experiment itself: model code, data handling, analysis. Only a few lines provision hardware.
- That part is structured: which call, which node or device type, which image, which duration.
- Extraction is deterministic, so the corpus rebuilds byte-identical.

> THE CLAIM
>
> *An artifact is mostly experiment code, and only a few lines of it provision hardware. Chunked as prose, those lines are buried among everything else. Treated as a structured evidence record, the same artifact becomes usable grounding rather than noise.*

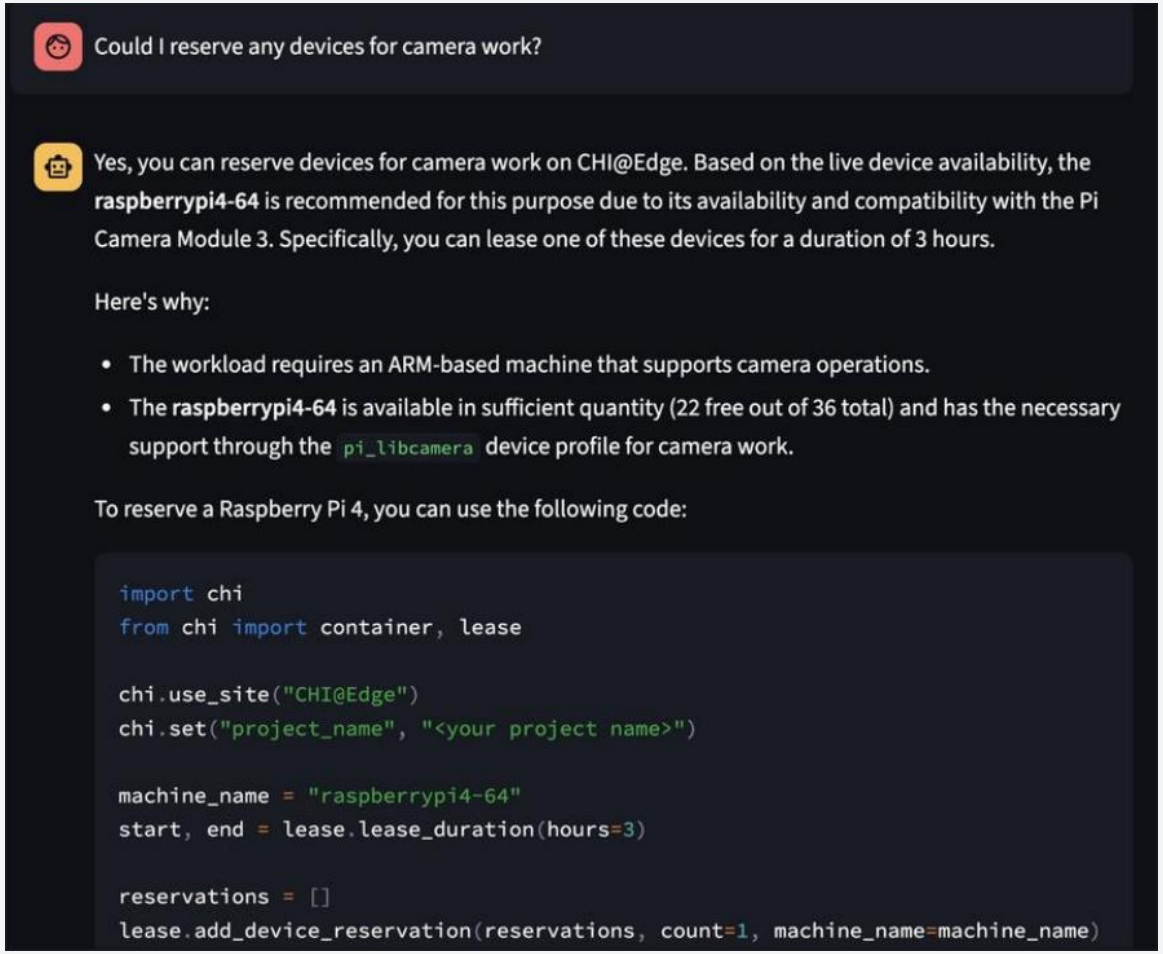
# Artifacts usage by advisor




Number of artifacts allowed to advisor
v/s
percentage of getting right set of artifacts

# The same question, asked again

1. Decide whether this is a resource question at all.
2. Read live availability at question time.
3. Recommend a node or device type, a free unit, a count and a duration.
4. Emit ready-to-run `python-chi`.
5. Link the artifacts the recommendation came from.




DEFINITION

**python-chi:** The Python library researchers use to drive Chameleon: request a lease, reserve a node or device, launch on it.

# Three contributions

1 **Hardware and live state**

what exists, and what is free

2 **Artifacts**

what real experiments actually ran on

3 **A benchmark, in two wings**

so we can tell whether any of it helps

# One benchmark, two wings




| CHECKER GROUP | THE QUESTION IT ASKS | EDGE | CHAMELEON |
|---|---|---|---|
| feasibility | Is the top pick real, free, and enough of it? | 228 | 328 |
| capability | Did you rank only hardware that can do the job? | 358 | 261 |
| safety | Did you avoid the calls that break things? | 199 | 138 |
| mechanism | Right API calls, wired together correctly? | 115 | 125 |
| specifics | Right site, image, hours, node type? | 115 | 123 |

Checker instances, not items. The two wings are never blended: different hardware, different syntax. Compare their shape against each other, never their scores.

# A checker is one automated test on the answer

THE ITEM

"Reserve a Pi with a camera for two hours."

ANSWER A · PASSES 5 OF 5

```
Use raspberrypi4-64, which supports
the pi_libcamera profile.
22 of 36 are free right now.

lease.add_device_reservation(
    reservations, count=1,
    machine_name="raspberrypi4-64")
start, end = lease.lease_duration(hours=2)
Container(reservation_id=lease_id)
```

ANSWER B · PASSES 3 OF 5

(wrong call)

```
lease.add_node_reservation(
    reservations, count=1,
    node_type="raspberrypi4-64")
start, end = lease.lease_duration(hours=2)
server.create_server(...)
```

(wrong call)

WHAT EACH CHECKER READS, AND ITS VERDICT

| CHECK GROUP | READS | A | B |
|---|---|---|---|
| **Mechanism** | the calls, and how they connect | pass | fail |
| **Specifics** | the literal strings and numbers | pass | pass |
| **Safety** | what is absent: no bare-metal call | pass | fail |
| **Feasibility** | the device against saved availability | pass | pass |
| **Capability** | the device against the hardware table | pass | pass |

B names a device that exists, is free, and can drive a camera. It reserves it with the bare-metal call, which cannot reserve an edge device at all. Getting the values right and the mechanism wrong is the common failure, and it is why the groups are never averaged into one score.

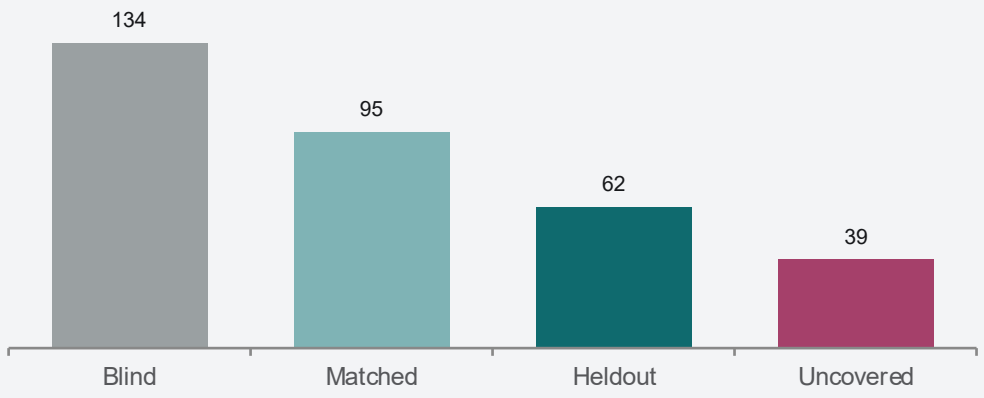
Across both wings the harness holds **reusable checker rules** instantiated against every item. ·

# Each question is asked four times, changing only the context

| CONDITION | PAIRS | WHAT IS PASTED ABOVE THE PROMPT | WHAT IT MEASURES |
|---|---|---|---|
| Blind | **134** | nothing at all | what the system manages unaided |
| Matched | **95** | the artifact that reserves a Pi camera | the answer is present in the context |
| **Heldout** | **62** | artifacts for other edge work: an SSH image, a sensor hat | whether anything carries over from a neighbouring artifact |
| Uncovered | **39** | artifacts, none of which document the device asked for | behaviour when no artifact describes the hardware at all |

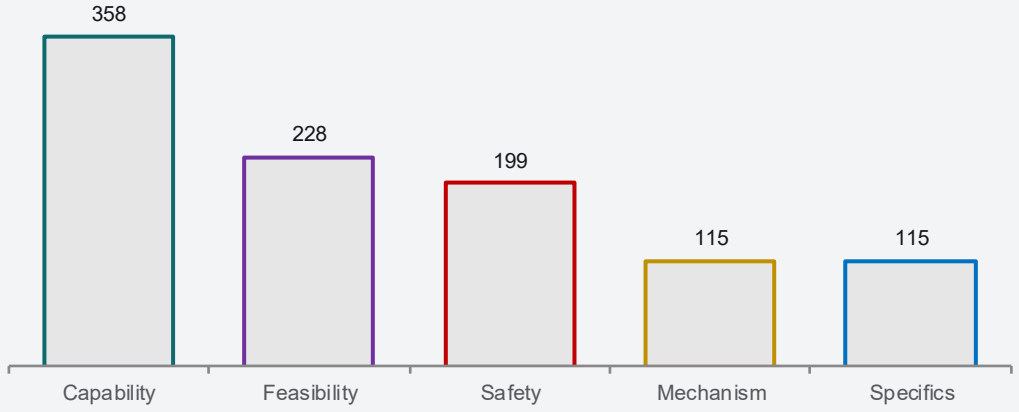
A *pair* is one item asked under one condition, so 330 pairs in total. Not every item has all four: an item only gets a condition when the right context exists for it.

ITEM AND CONDITION PAIRS




Types of conditions questions were asked in

CHECKER INSTANCES BY GROUP, EDGE WING




Types of checkers (tests)

# Three experiments, one benchmark

| EXPERIMENT | THE QUESTION | WING | ARMS | ITEMS |
|---|---|---|---|---|
| **Artifacts** | If we hand the system the right artifact, does the answer improve? | Edge | **4** | **95** |
| **Advisor** | If the system has to find the artifact and the live state itself, does the answer improve? | Edge | **2** | **134** |
| **Advisor** | Does the same result hold on bare metal, where the answer is a ranked list of node types? | Bare-metaal | **2** | **136** |

In all three, every item is answered by every arm, and everything is held fixed except the one thing named. Same questions, same checkers, same scoring pass.

Both advisor experiments run `Meta-Llama-3.3-70B-Instruct` on the TACC Tejas endpoint, which is what the deployed system uses, under an identical system prompt. The artifact experiment adds a second system so that result is not about one model.

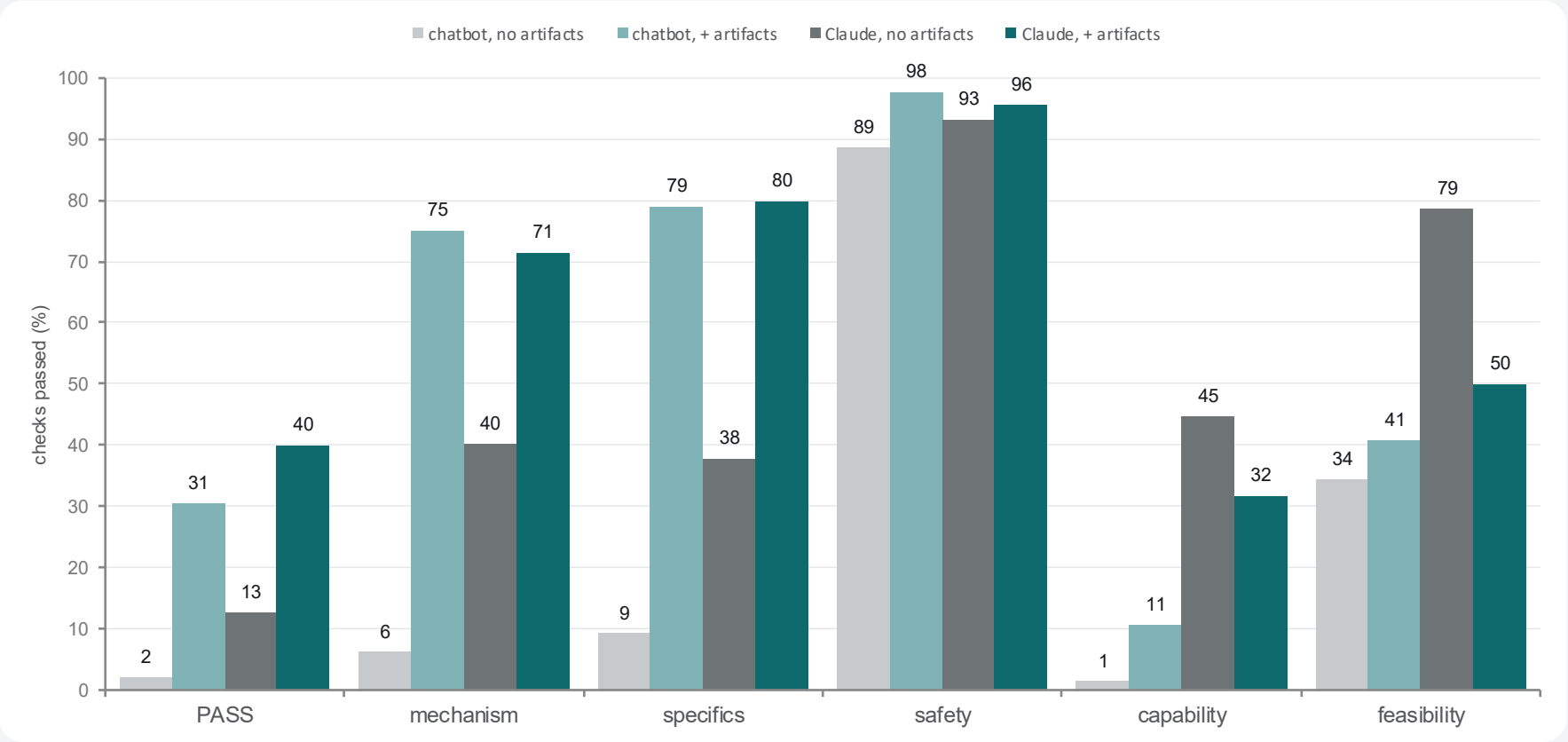
## HOW TO READ EVERY CHART THAT FOLLOWS

- no grounding: the model answering unaided
- grounded, edge wing
- grounded, bare-metal wing
- the failure case: hardware no artifact documents

Lighter shades of the same colour are steps toward that grounding, never a different subject. Every chart is a vertical bar chart with the same axis treatment, so a bar height means the same thing on every slide

# Edge Results

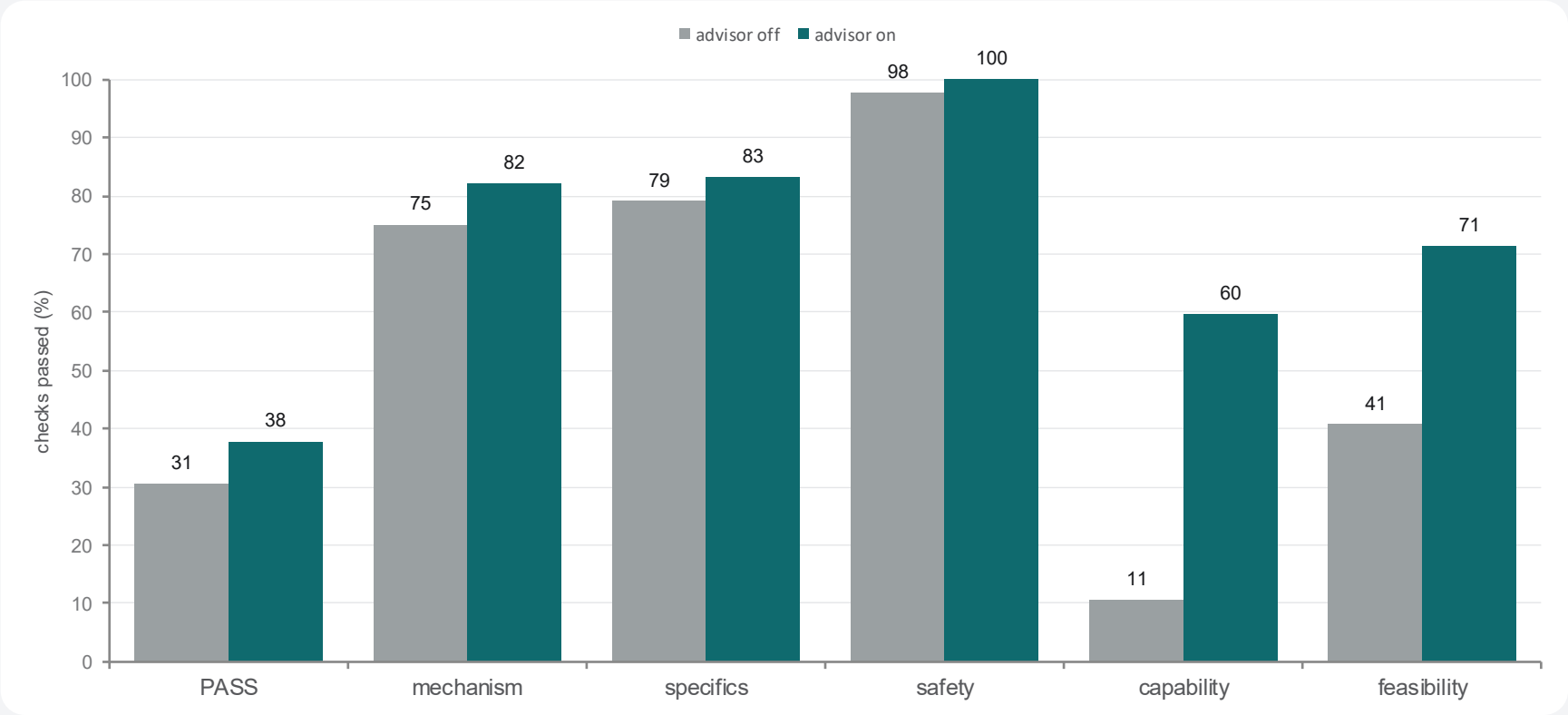
# Given the right artifact, both systems improve sharply (Edge)




Chatbot/Claude + Artifacts on Edge benchmark

**In practice.** Fully correct answers go 2 % to 31 % for the chatbot and 13 % to 40 % for Claude, on the same 95 matched questions. Grounding in artifacts is not a crutch for a weak model.

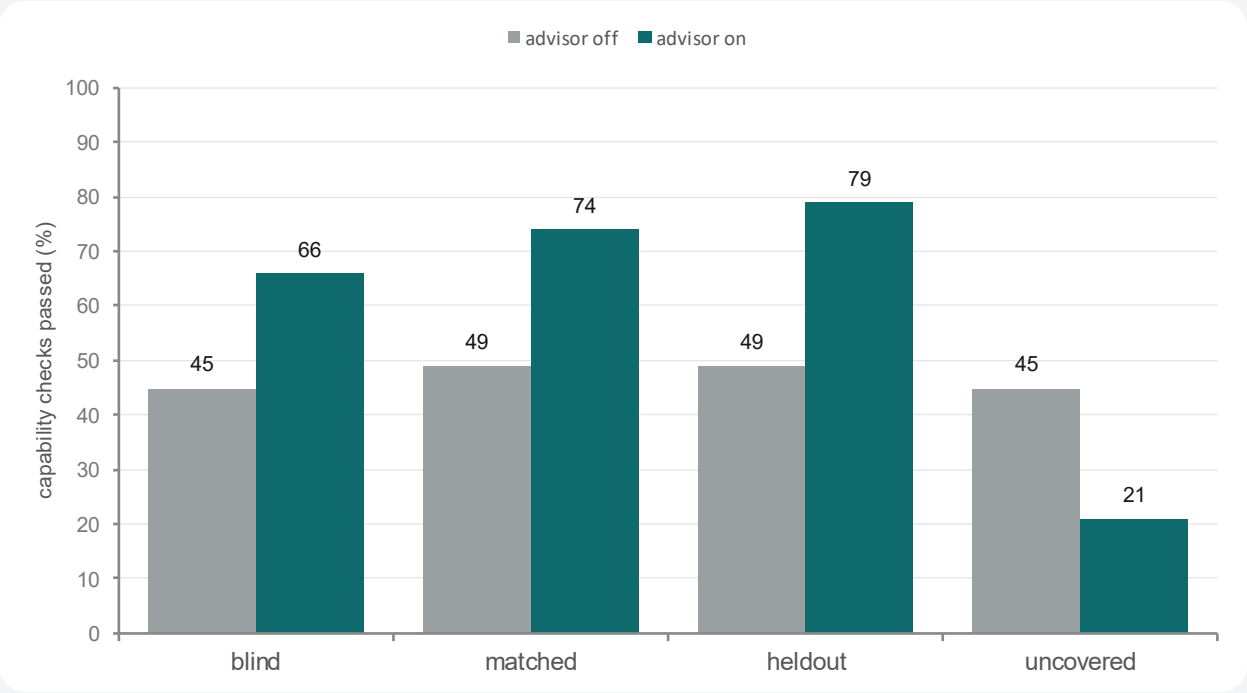
# With the advisor on, capability and feasibility groups move sharply




Chatbot + Artifacts + Advisor on Edge benchmark

**In practice.** The assistant now names a device that exists, that is free, in the right call, far more often than before.

# Grounding helps only where the corpus covers the hardware




Chatbot scores on different artifact conditions

WHY: NO ARTIFACT DOCUMENTS ANY ACCELERATOR ON CHI@EDGE

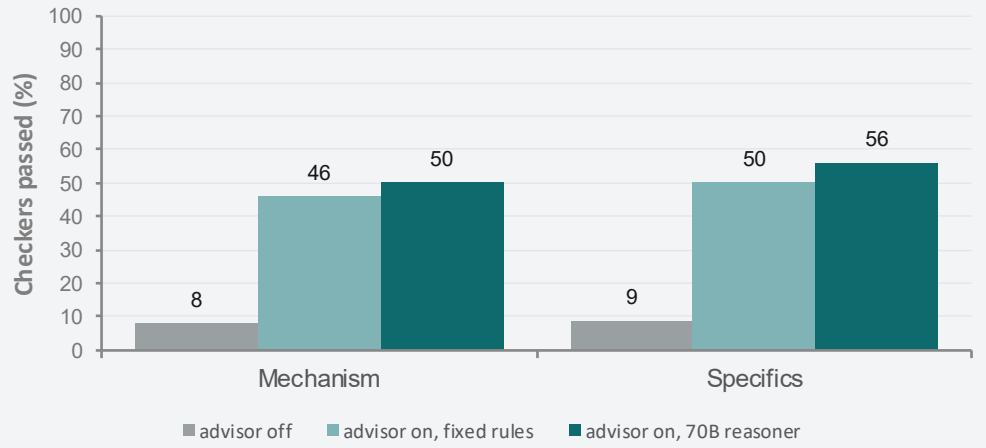
| DEVICE TYPE | DEVICES | ARTIFACTS |
|---|---|---|
| raspberrypi4-64 | 36 | 4 |
| raspberrypi5 | 10 | 1 |
| jetson-nano | 12 | 0 |
| jetson-xavier-nx-devkit-emmc | 4 | 0 |
| jetson-orin-nano-devkit-nvme | 3 | 0 |
| jetson-agx-orin-devkit-64gb | 1 | 0 |
| coral-dev | 2 | 0 |

**On uncovered hardware the advisor makes the answer worse: 45 % to 21 %.**

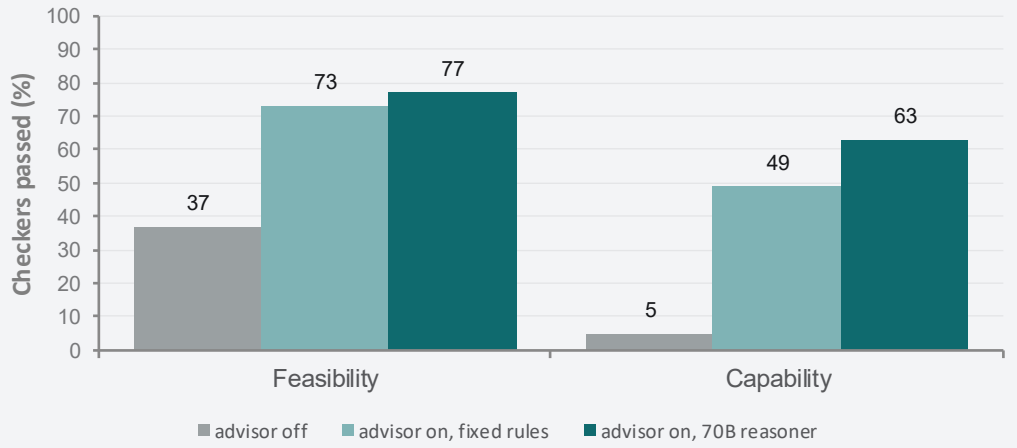
# Most of the gain comes from the advisory, and some from reasoning over it

t

CODE GENERATION · 50 CORE ITEMS




RESERVATION DECISION · 84 ITEMS · CAPABILITY SUBSTANTIVE

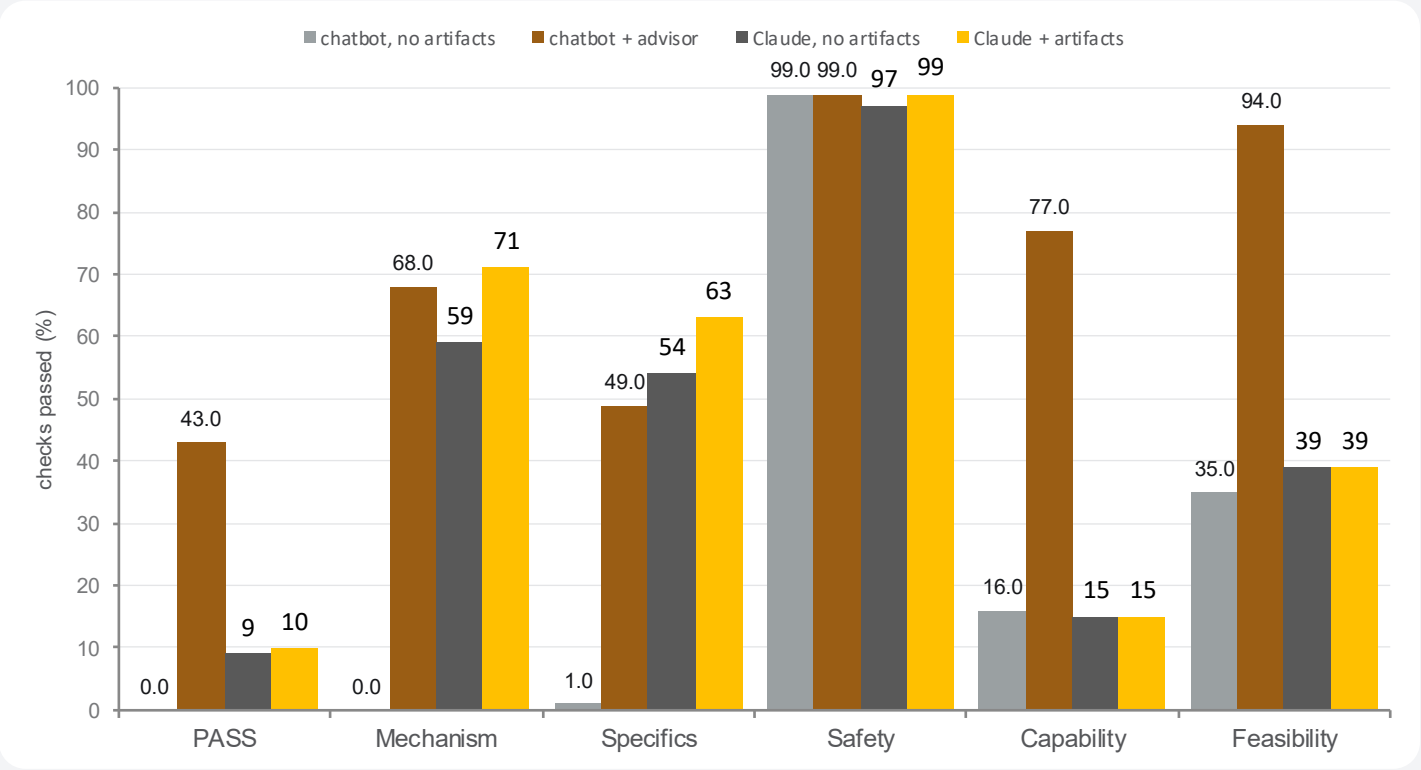



Chatbot scores with advisor being able to reason more

**In practice.** Most of the value is having the advisory at all. On substantive passes, capability runs 5 to 49 to 63: of the 58 point gain the fixed rules supply 44 and the 70B reasoner adds 14.

# Bare metal Results

# The advisor result holds on bare metal




Chatbot + Artifacts + Advisor on bare-metal benchmark

FULL WING, 136 ITEMS EACH ARM

| | |
|---|---|
| All checks pass, bare | 0 / 136 |
| **All checks pass, advisor** | **58 / 136** |

- **Zero regressions on the reservation suite.** The set of items passing only in the bare arm is empty.
- **Without reference material the model does not write python-chi at all.** Of 40 code answers, 26 were portal click-throughs and 10 were openstack CLI. With retrieved artifacts it satisfies 72 % of mechanism checks.

# Which claims the results actually carry

| CLAIM, AND THE VERDICT | WHAT THE RESULTS SUPPORT | WHAT THEY DO NOT SETTLE |
|---|---|---|
| **Artifacts are usable grounding**<br>SUPPORTED | Given the right artifact, fully correct answers go 2 % to 31 % for the chatbot and 13 % to 40 % for Claude, on the same 95 questions. | Artifacts teach how to write the reservation, not which hardware to pick. On that question they barely move the chatbot, 45 % to 44 %, and move Claude only modestly, 45 % to 57 %. |
| **The advisor improves the system**<br>SUPPORTED | All five check groups improve. Fully correct answers go from 5 of 134 to 23 of 134. | |
| **The advisor uses live availability**<br>SUPPORTED | It names a device that is actually free at the time of asking, and feasibility rises 37 % to 71 %. | |
| **Grounding helps only where the corpus covers the hardware**<br>SUPPORTED | On hardware that no artifact documents, the advisor makes hardware choice worse, 45 % to 21 %. | |
| **The result carries over to bare metal**<br>SUPPORTED | On 136 Chameleon-wing items the advisor takes full-item pass from 3 to 60, with zero regressions on the reservation suite. | |

**What the results say nothing about:** Safety was already near the ceiling in both wings, so little was learned there, and on the Chameleon wing it costs 2.9 points KVM is out of scope: it reserves flavors rather than hosts. And the hardware-choice percentages count the empty-answer passes described above, so the unaided figures read a little high.

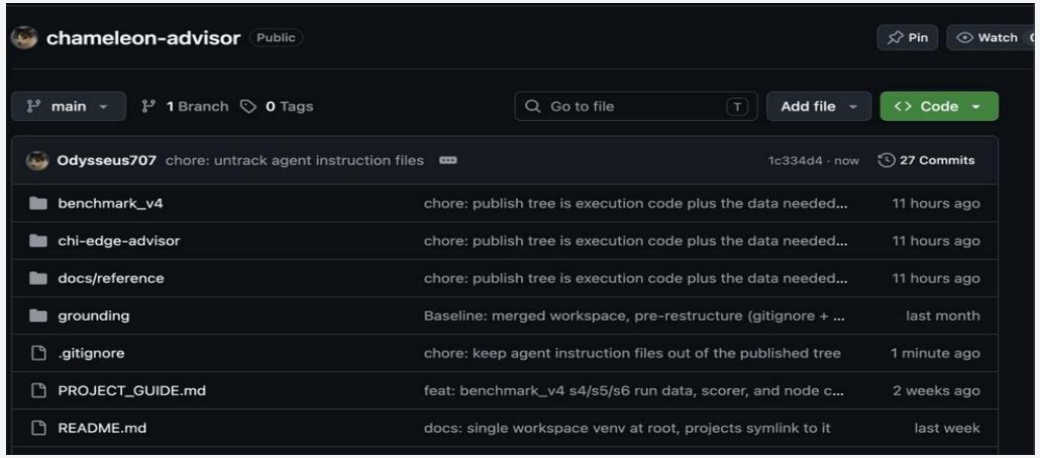
# What remains, and what is deliberately not claimed

- **Reserve the node, not just recommend it.** One confirmation, no copy-paste.
- **Production hosting.** The service needs credentials of its own, because it cannot act using each user's login.
- **Close the retrieval gap.** Within-cluster ranking and the top-3 budget are the two addressable parts.

## SHIPPED

- The advisor and both benchmark wings are public.
- The benchmark installs as a wheel, so anyone can `pip install` it and score a run. (still being tested)

github.com/Odysseus707/chameleon-advisor




## WHAT THE WORK DOES NOT CLAIM

- **That the two wings can be compared by score.** Different hardware, different suites. Compare their shape.

# Questions worth discussing

## RUNS THE SETUP ALREADY SUPPORTS

- **Tutorials or real experiments?** Feed the teaching artifacts, then the research ones, and see which kind carries the transferable part.
- **How many artifacts are enough?** One, three, five.
- **Does it hold on other models?** One model and one endpoint is the current limit on both wings.

## OPEN DECISIONS

- Should the advisor refuse to recommend hardware no artifact documents?
- How should the benchmark be distributed so others can run it?
- Should scoring ever submit a real lease? Today an answer is checked against a saved copy of availability rather than against the testbed, so a reservation the scheduler would refuse can still pass.
- Is the safety dip on the Chameleon wing worth the capability it buys?