

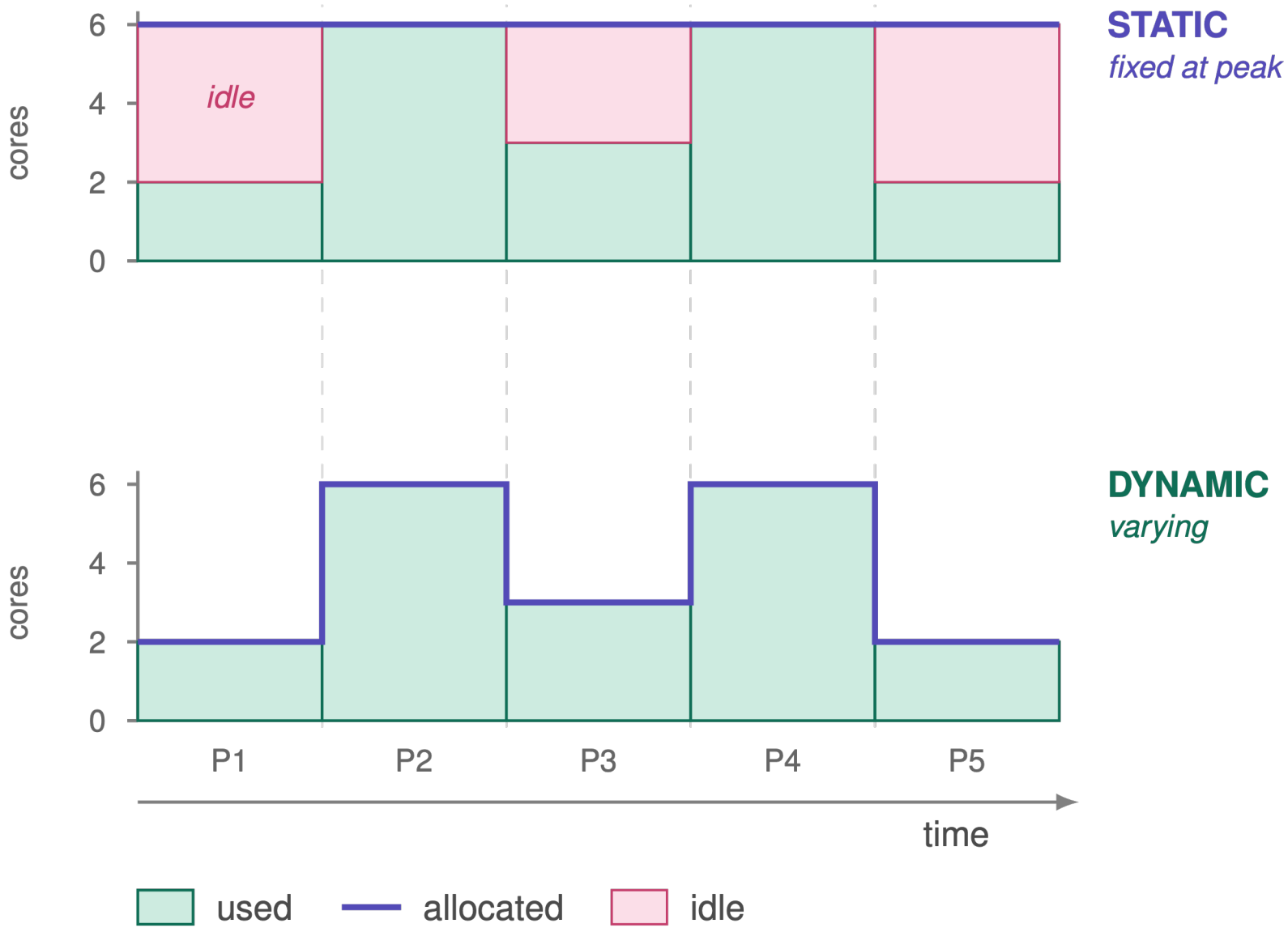
Leibniz-Rechenzentrum
der Bayerischen Akademie der Wissenschaften

A Diagnostic Tool for Phase-Aware Resource Allocation in Malleable HPC Workloads

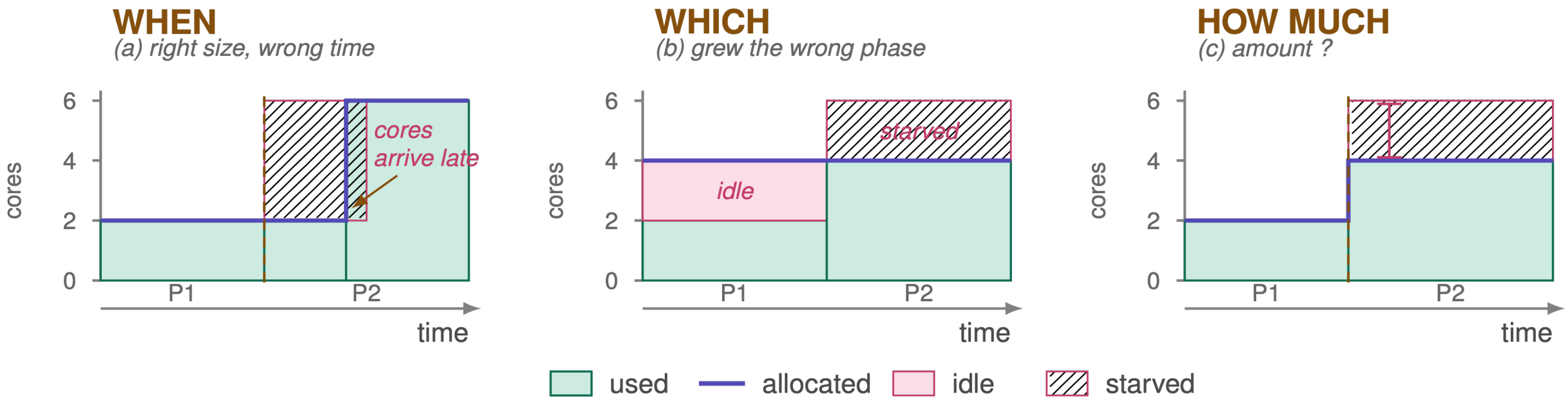
Arjun Parab (LRZ), Amir Raoofy (LRZ), Josef Weidendorfer (TUD)
Leibniz Supercomputing Centre (LRZ), TUD Dresden University of Technology (TUD)

Dresden | 01.10.2026

Motivation

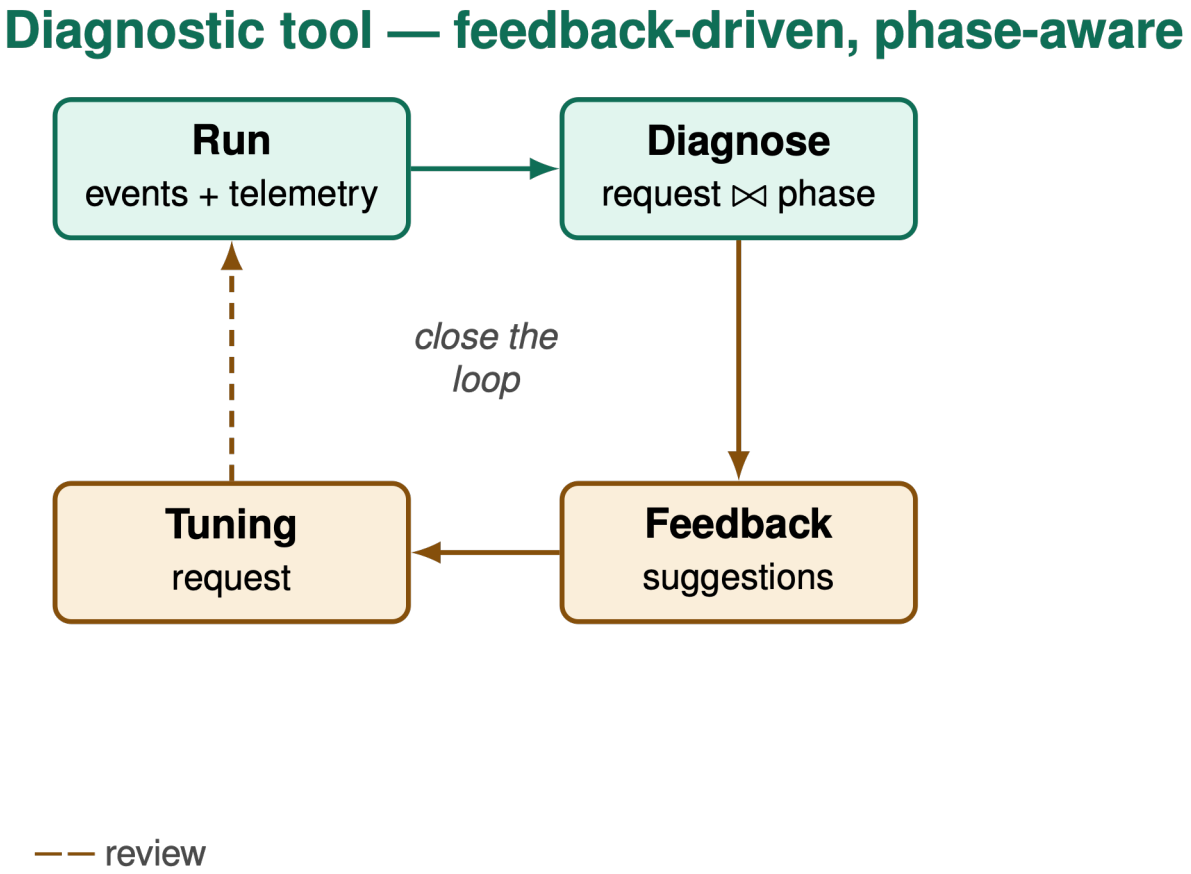

cores
6
4
2
0
idle
STATIC
fixed at peak
cores
6
4
2
0
DYNAMIC
varying
P1
P2
P3
P4
P5
time
used
allocated
idle

Problems


WHEN
(a) right size, wrong time
cores
arrive late
WHICH
(b) grew the wrong phase
idle
starved
HOW MUCH
(c) amount ?
cores
6
4
2
0
P1
P2
time
used
allocated
idle
starved

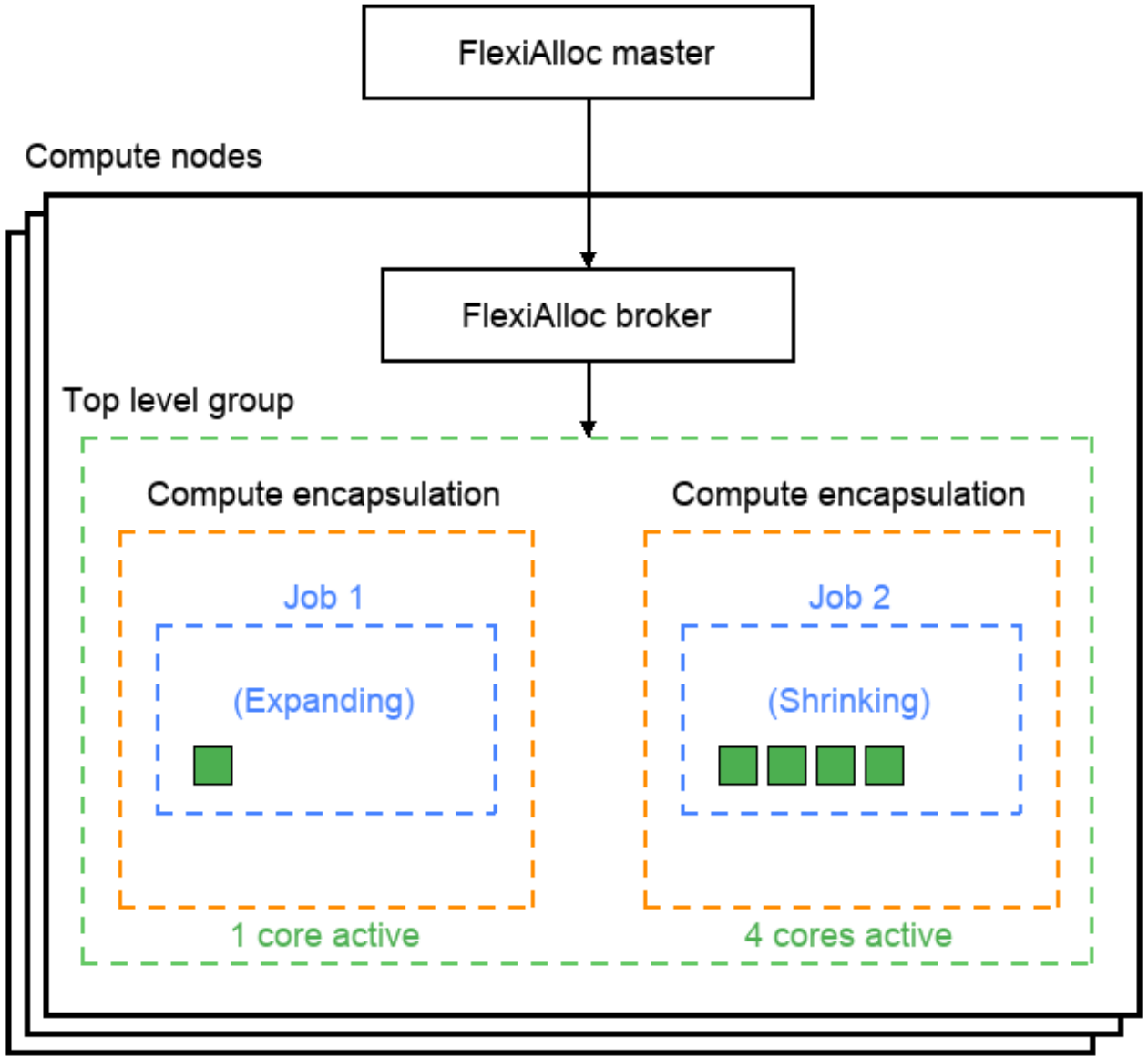
Proposed solution

- FlexiAlloc
- Diagnostic Tool
- Feedback + Tuning



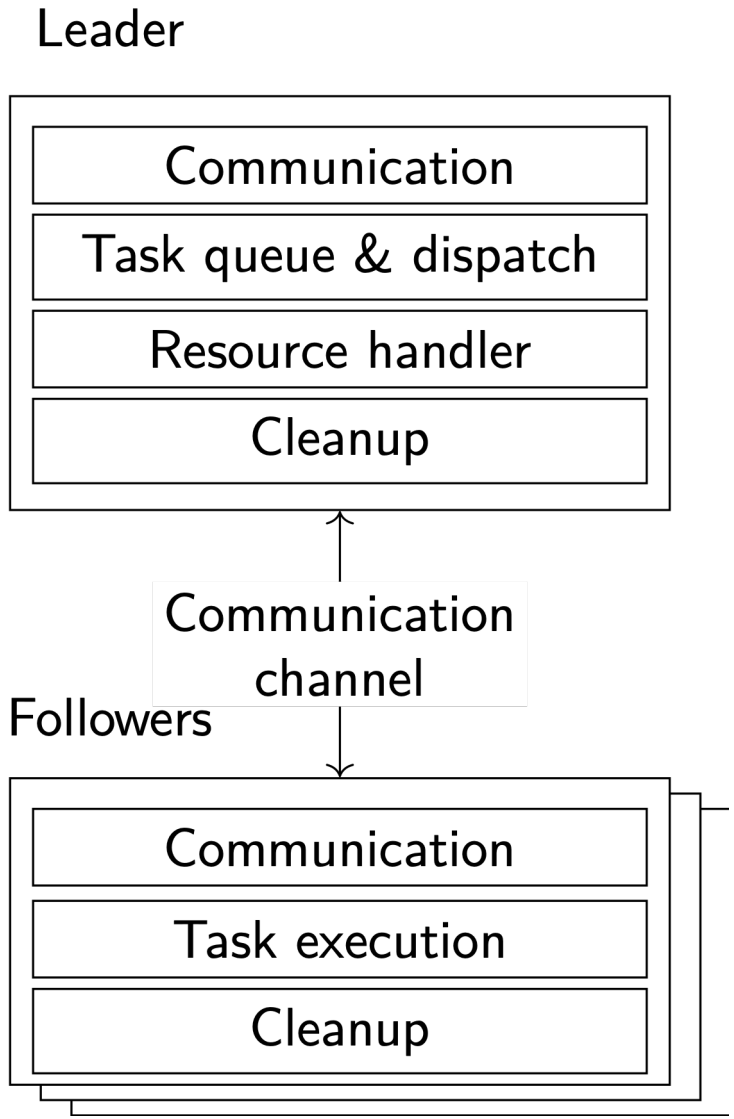
Flexible Allocation (FlexiAlloc)

- Dynamic resource management framework
- Compute encapsulation
- Fast and flexible prototyping
- Studying scheduling strategies



Leader-Follower Programming Model (LFPM)

- Restructure SPMD to support dynamic allocations
- LFPM introduces new abstract roles
  - Leader
  - Follower



LFPM Comparison with MPI

```
...
print("Hello from rank ", rank, " of ",
size)
...
```

```
mpirun -n 4 ./hello
```

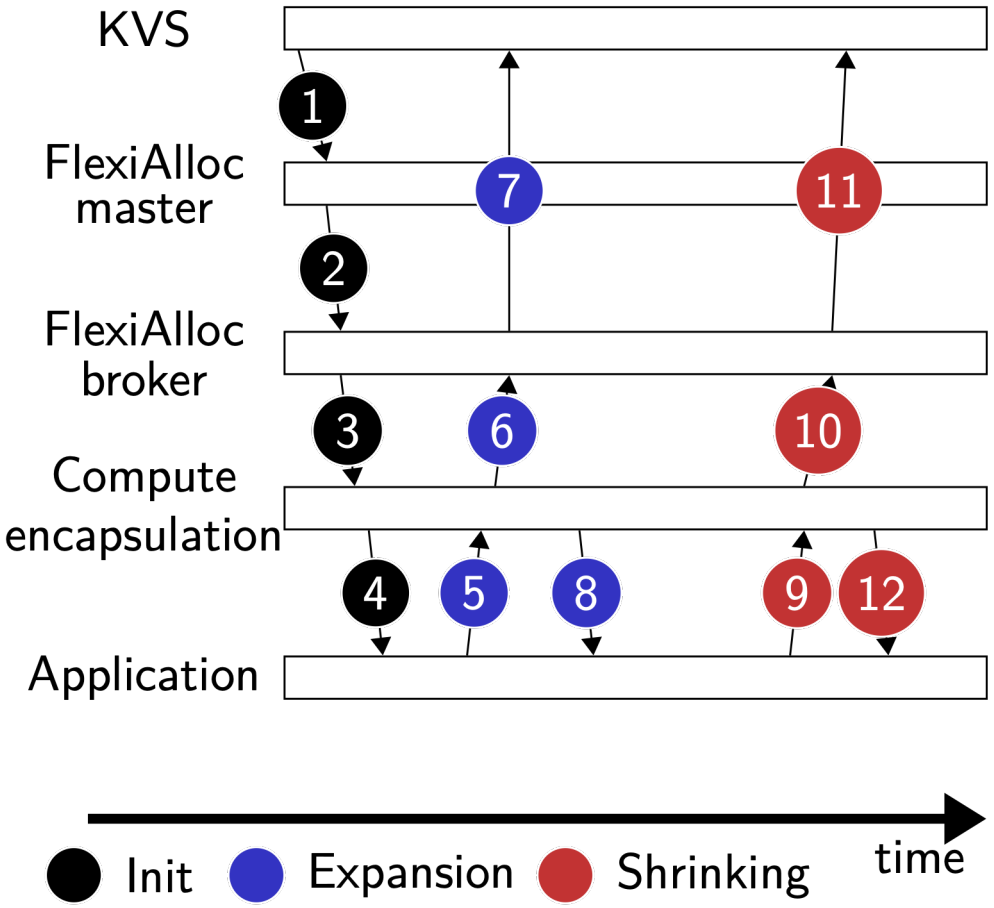
```
Hello from rank 0 of 4
Hello from rank 1 of 4
Hello from rank 2 of 4
Hello from rank 3 of 4
```

```
role_discovery():
  leader() or follower()
  ...
leader():
  print(f"Hello from: {identity}")
  ...
follower():
  print(f"Hello from: {identity}")
  ...
```

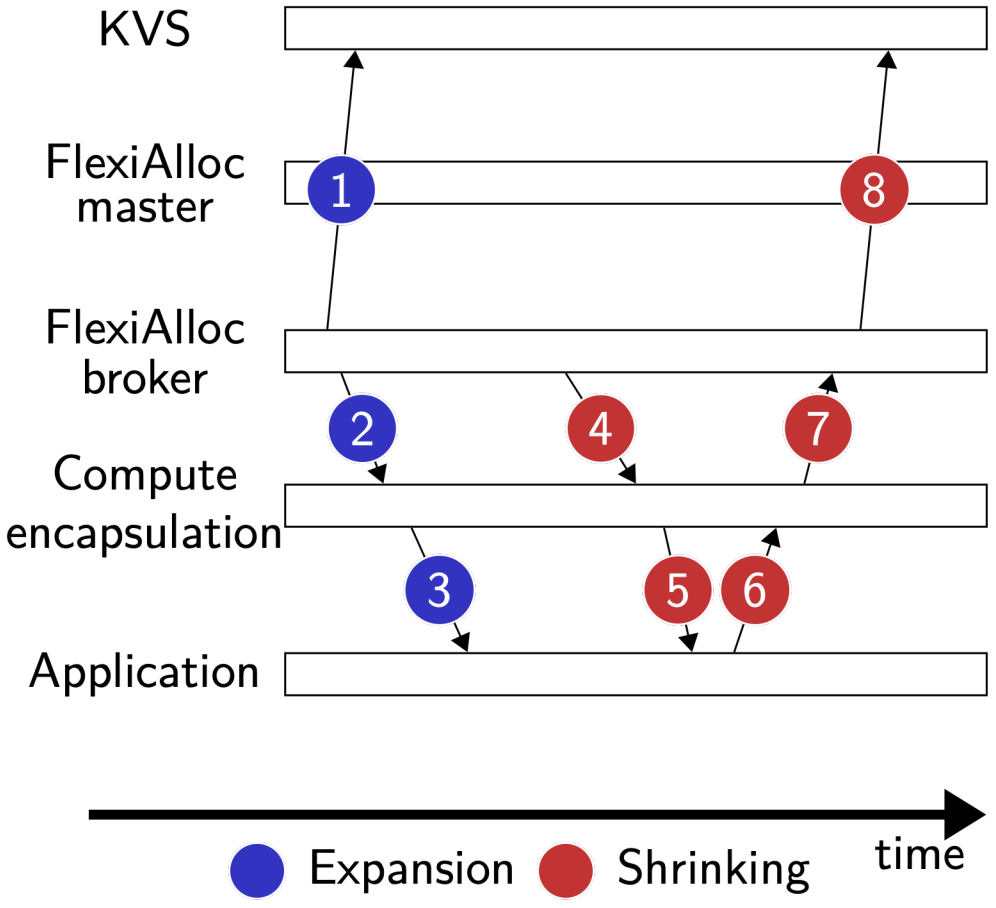
```
flexialloc job submit 4 'python3 hello.py'
```

```
Hello from: leader
Hello from: follower-wolpy-1234-1
Hello from: follower-wolpy-1235-2
Hello from: follower-wolpy-1236-3
```

FlexiAlloc Workflow



Application scaling

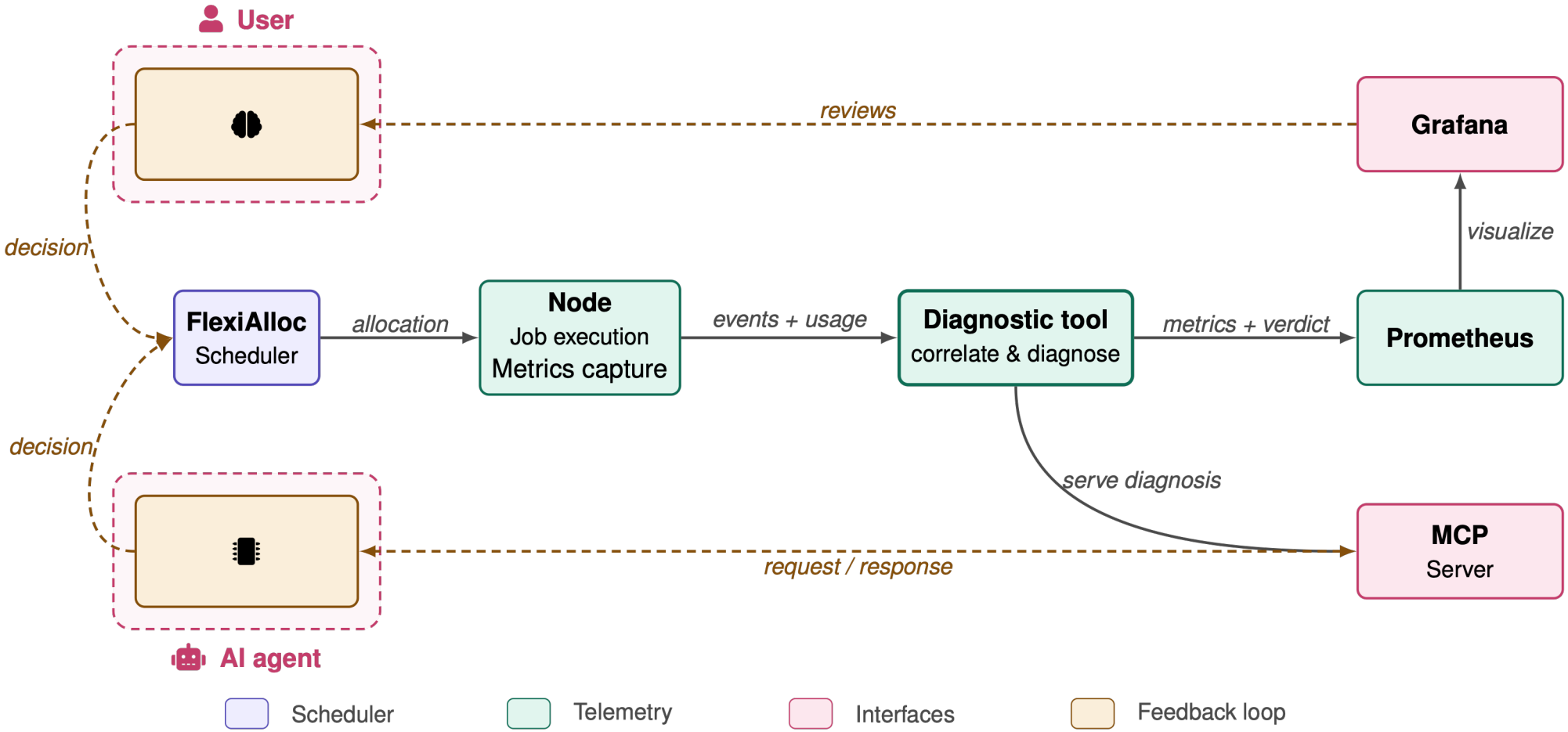


Scheduler scaling

Requirements

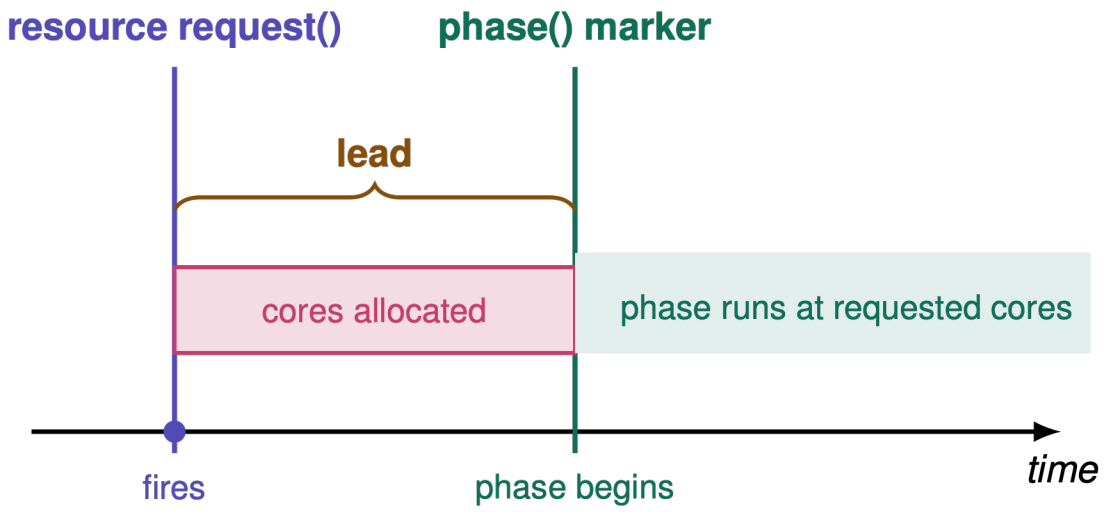
- Linking domains
- Correlation on shared timeline
- Diagnosis
- Output options
- Actionable suggestion

Architecture



Annotation and Scenario

Annotate: resource request() before each `phase()` marker



Good ⇒ cores granted within 20% of allocation time
Ok ⇒ 20–50% (little idle)
Early ⇒ >50% (idle well before use)
Late ⇒ cores arrive after the phase starts (under-provisioned)

Scenario example

```
config(job_id="job-1050", app="cfd-solver",
       initial_cores=2, allocation_time=5,
       redistribution=2)

#  ...

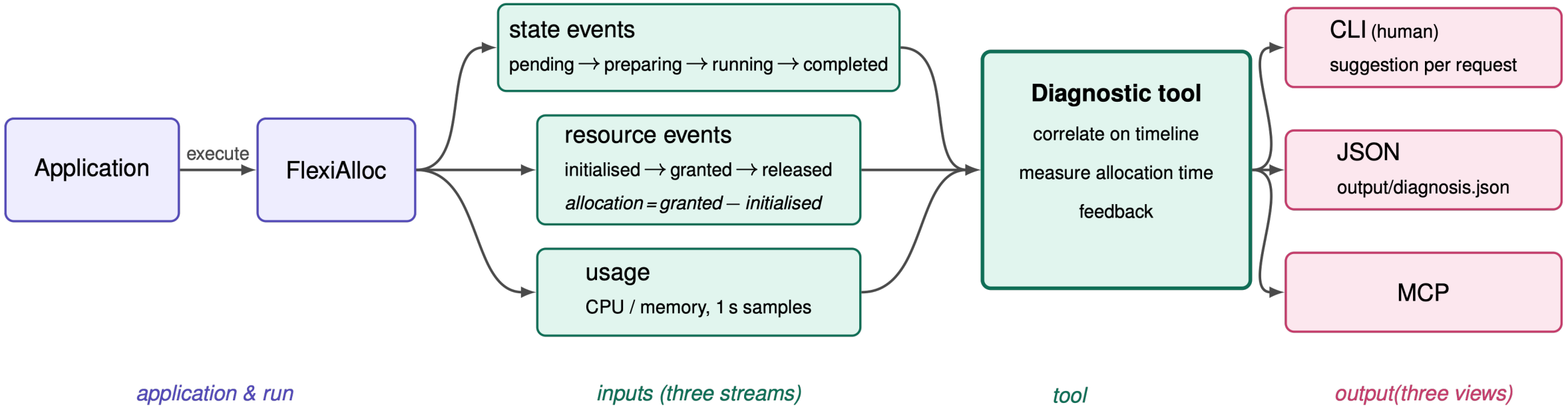
request("solve_A", cores=6, lead=5)

#  ...

phase("solve_A", dur=30, ...
      redistribution=2,
      redistribution_with_allocation=5)

phase("reduce_1", dur=15,...)
#  ...
```

Workflow


Application
execute
FlexiAlloc
state events
pending → preparing → running → completed
resource events
initialised → granted → released
allocation = granted − initialised
usage
CPU / memory, 1 s samples
Diagnostic tool
correlate on timeline
measure allocation time
feedback
CLI (human)
suggestion per request
JSON
output/diagnosis.json
MCP
application & run
inputs (three streams)
tool
output(three views)

Initial request placement

Initial request placement

Phase	Op	Cores	Lead	Arrival	Feedback	Suggestion
solve_A	expand	2 → 6	5	on time	good	–
solve_B	expand	6 → 12	7	−2 s	ok	lead 7 → 5
solve_C	expand	12 → 16	20	−15 s	early	lead 20 → 5
solve_D	shrink	16 → 8	2	+3 s	late	lead 2 → 5
solve_E	expand	8 → 16	1	+4 s	late	lead 1 → 5

Invisible in usage metrics alone

`solve_E` ran under-provisioned

`solve_C` idled cores (granted early).

Feedback: **1 good** · **1 ok** · **1 early** · **2 late**

Feedback Loop: after improvement

Applying suggestions

Phase	Op	Cores	Lead	Arrival	Feedback
solve_A	expand	2 → 6	5	on time	**good**
solve_B	expand	6 → 12	5	on time (was 7)	**good**
solve_C	expand	12 → 16	5	on time (was 20)	**good**
solve_D	shrink	16 → 8	5	on time (was 2)	**good**
solve_E	expand	8 → 16	5	on time (was 1)	**good**

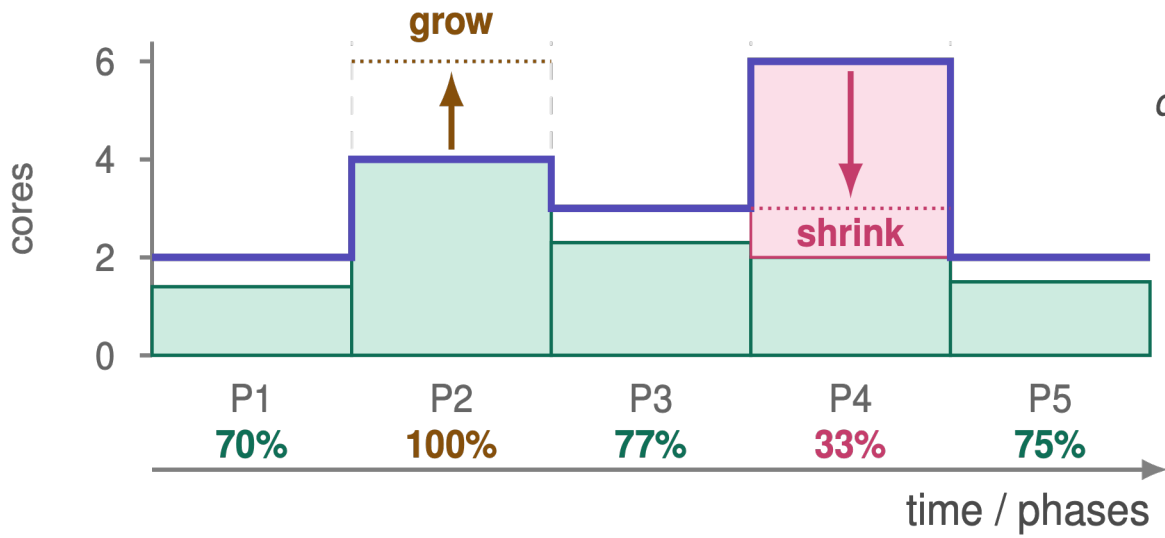
`solve_E` no longer starves

`solve_C` no longer idles cores.

Note: each resized phase now pays redistribution-with-allocation

Feedback: **5 good** · 0 ok · 0 early · 0 late

Which and How much



diagnose →

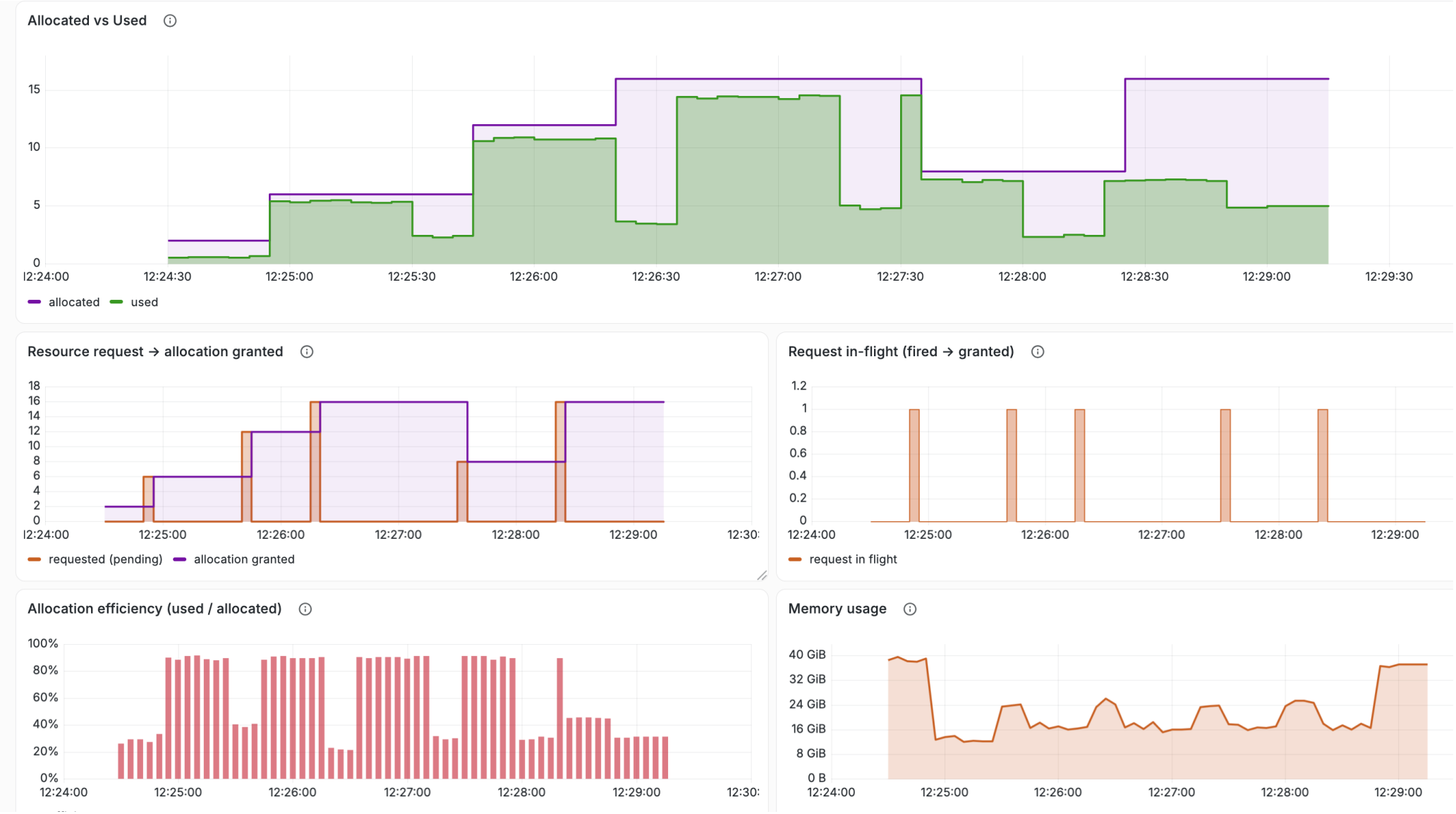
Phase	u	Feedback	Suggestion
P1	70%	ok	-
P2	100%	saturated	grow +2
P3	77%	ok	-
P4	33%	over-prov.	shrink→3
P5	75%	ok	-

▲ $u \geq 90\%$: grow $60\% < u < 90\%$: hold ▭ $u \leq 60\%$: shrink

per-phase u = *used / allocated drives* both *decisions*

Visualisation

Dashboard



Summary

- Malleability's benefit is real but conditional
- Tool correlates requests, reveals timing misalignments
- Review + Realigning requests, larger gains

Future Work

- Integration of FlexiAlloc
- LLM-agent-driven + scheduler-automated
- Measured redistribution marker

Arjun.Parab@lrz.de