QoSFlow: Ensuring Service Quality of Distributed Workflows Using Interpretable Sensitivity Models

Md Hasanur Rashid[1], Jesun Firoz[2], Nathan R. Tallent[2], Luanzheng Guo[2], Meng Tang[3], Dong Dai[1]

[1]University of Delaware, [2]Pacific Northwest National Laboratory, [3]Illinois Institute of Technology

DIRLAB

Workflows drive modern science

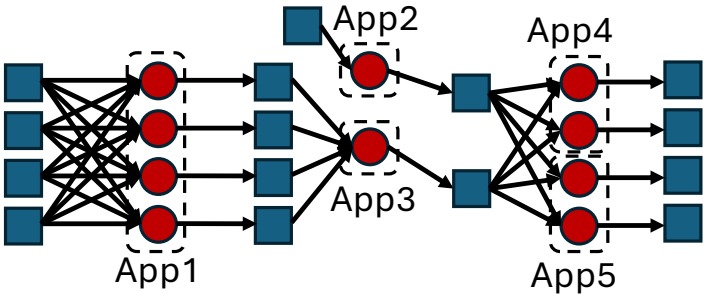
- Automated experiments, digital twins, data-driven pipelines
- Workflows ≣ **DAG** of applications

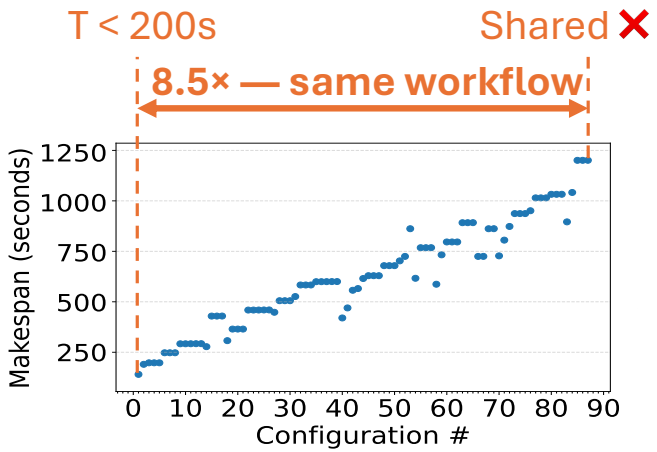
Users need explicit QoS

- *"Finish by deadline T"*
- *"Use ≤ N nodes"*
- *"Avoid storage tier X"*

Same workflow & data → wildly different runtimes

- 1000 Genome workflow at 10 nodes:
  - **140 s → 1201 s** depending on storage choice
- That's an **8.5× spread** with no code or data change





Can we recommend a config that meets a QoS constraint — and explain why?

DIRLAB

[C1] Path-dependent behavior

Critical path *shifts* with storage / concurrency choices

[C2] Resource sensitivity

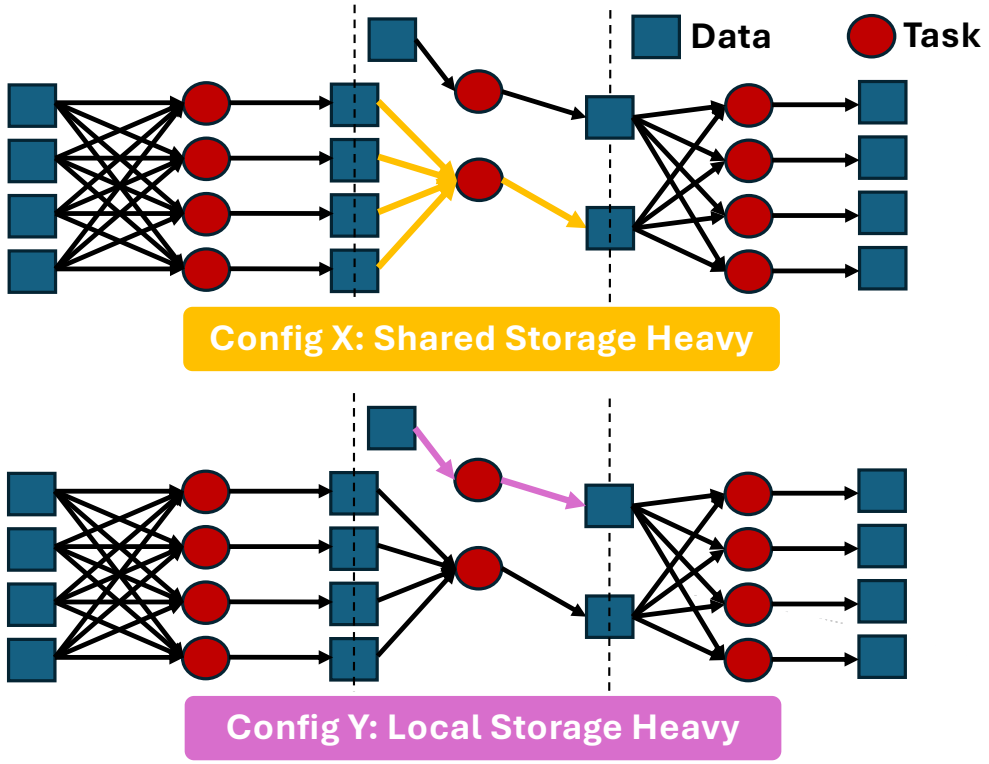
Which knobs matter? Which don't?

[C3] QoS → schedule

How to translate *"finish in T seconds"* into a config?

[C4] Interpretability

Why did the model pick this config?



Existing tools optimize but don't explain — users need both

DIRLAB

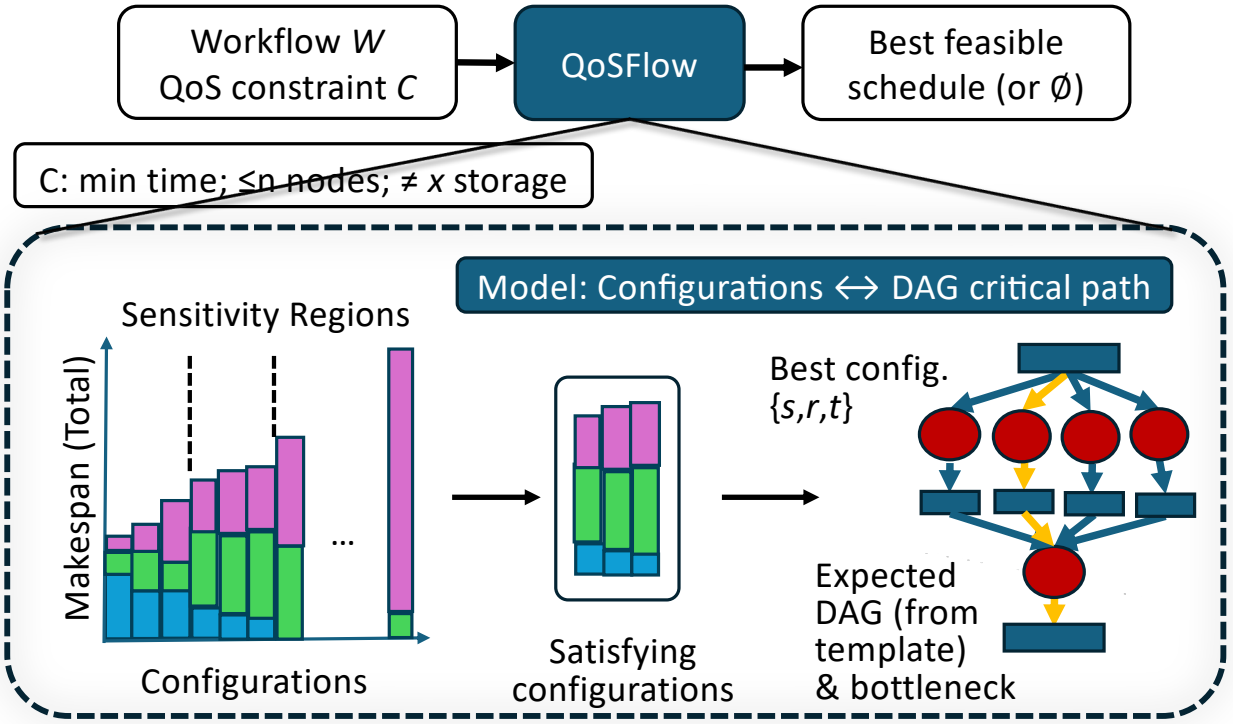
Our approach: *I/O-centric, interpretable sensitivity modeling*

Three contributions *(addressing C1–C4):*

1. **Sensitivity-driven regions** *[C1, C2]*
   Map configs to DAG critical paths; collapse equivalents

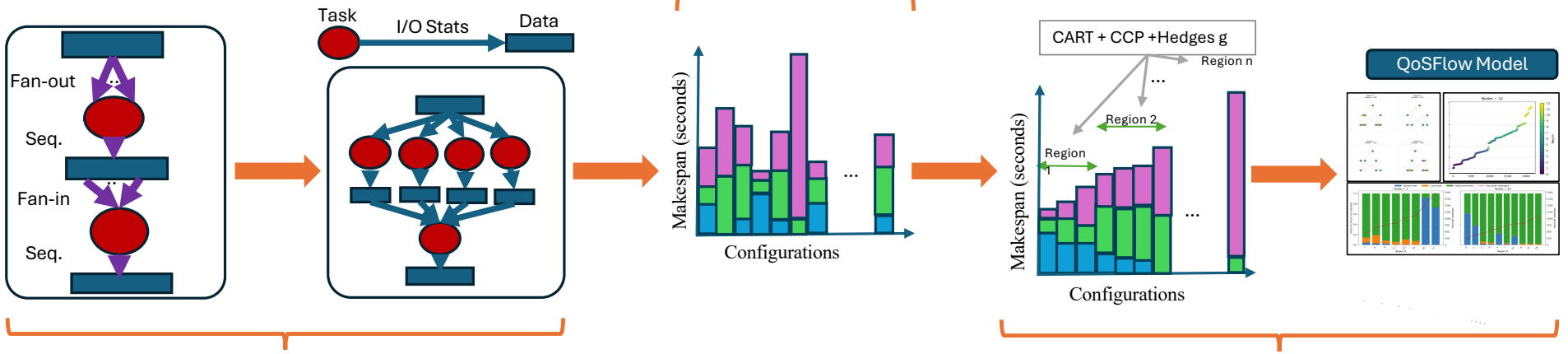
2. **Interpretable rules** *[C4]*
   Per-stage decisions marked *critical* or *flexible*

3. **QoS-driven selection** *[C3]*
   Constraints → configs (or Ø if infeasible)



DIRLAB

Estimate: Critical-path makespan per config



Model: DAG template + scale projection (no re-execution)

Partition: CART regions + interpretable rules

① Decompose each stage's I/O

- Stage-in, Execution, Stage-out
- Identify I/O and data movement cost

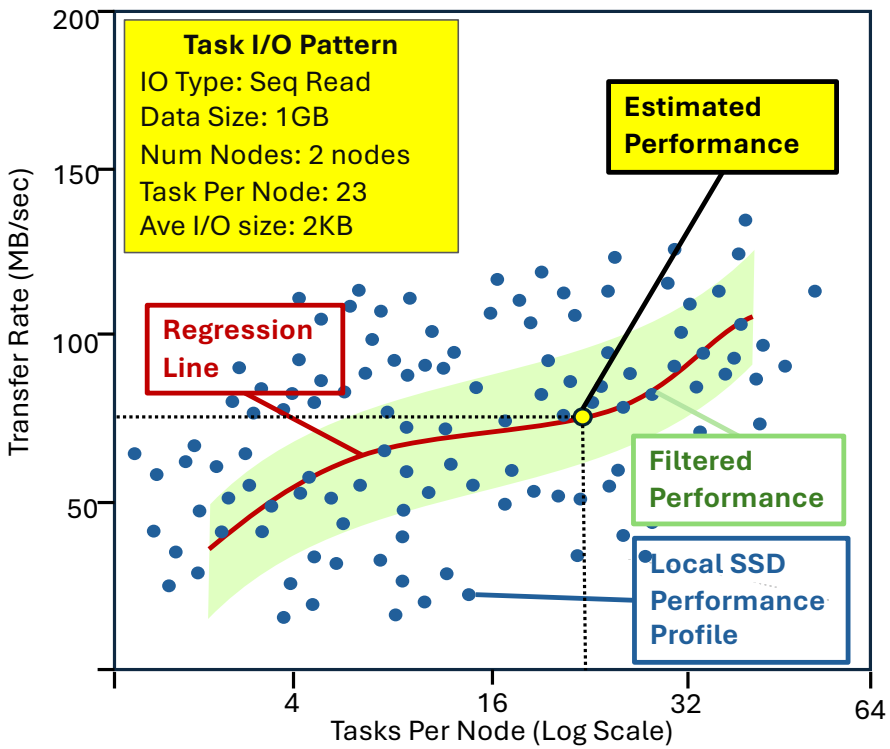
② Match to storage profile *(→ see right)*

- Lookup by I/O type, data size, parallelism, transfer size
- Read estimated transfer rate off the profile

③ Aggregate level-by-level

- Max across parallel stages within a level
- Sum across DAG levels

→ **makespan + critical path** per config



Score every config without running it — and identify which path is critical

DIRLAB

Key insight: regions, not single configs

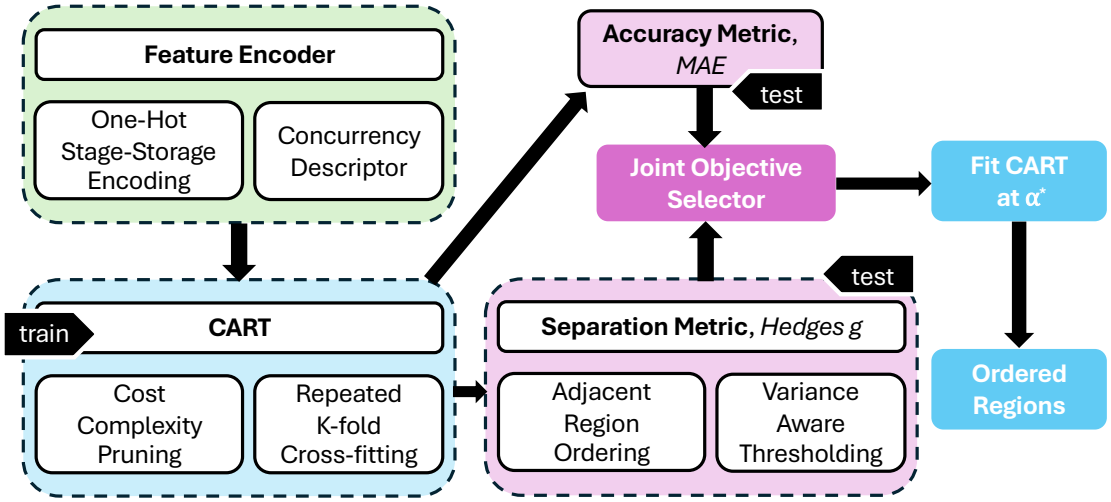
Configs sharing a critical path achieve similar makespan while off-path choices are flexible

① Classify sensitivity

- Critical-path stages → **critical**
- Off-path stages → **flexible**

② Partition with CART *(→ see right)*

- Each tree leaf → one performance region
- Joint objective avoids two failure modes:
  - MAE alone → over-segments on **noise**
  - Hedges' g alone → trivial, **coarse regions**



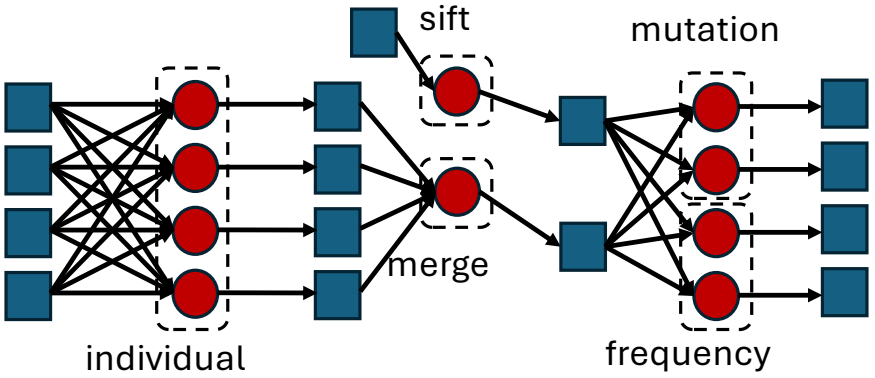
③ Order regions by median makespan

→ low intra-region variance, clear inter-region steps

From thousands of configs → a handful of interpretable, ordered regions

1000 Genomes (1kgenome) — bioinformatics, data-intensive

- Per-chromosome data extraction, SNP scoring, frequency analysis
- Variable I/O patterns across stages → **storage tier choice matters**



HPC Cluster

- Dual AMD EPYC 7502 (64 cores)
- 256 GB DDR4-3200 memory
- 6× NVIDIA A100 GPU per node
- HDR-100 InfiniBand (100 Gb/s)

Three storage tiers *(the QoS knob)*

- **BeeGFS** — remote parallel filesystem
- **NVMe SSD** — node-local, >1 GB/s
- **tmpFS** — main-memory backed

*Also evaluated: **PyFLEXTRKR** (atmospheric science, 9 stages) and **DeepDriveMD** (ML-steered MD, iterative) — see paper for details.*

DIRLAB

FSF (Fastest-Storage First)

↳ prefer high-bandwidth tiers

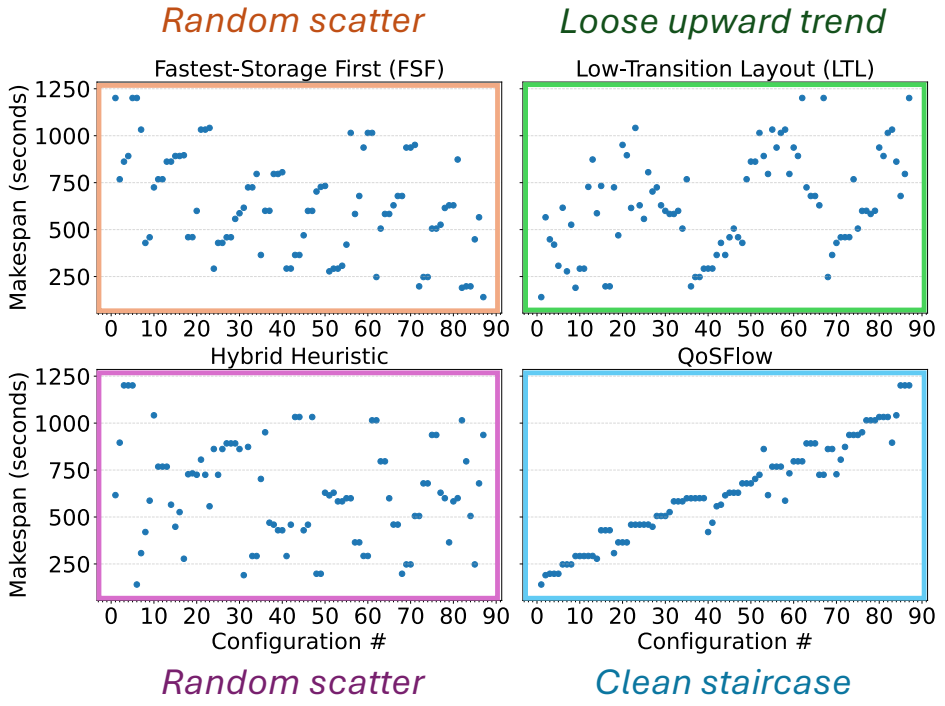
LTL (Low-Transition Layout)

↳ minimize cross-tier data movement

Hybrid (FSF ⊕ LTL)

↳ combine both

Policy	Pairwise Concordance
FSF	0.378
Hybrid	0.503
LTL	0.751
QoSFlow	**0.956**

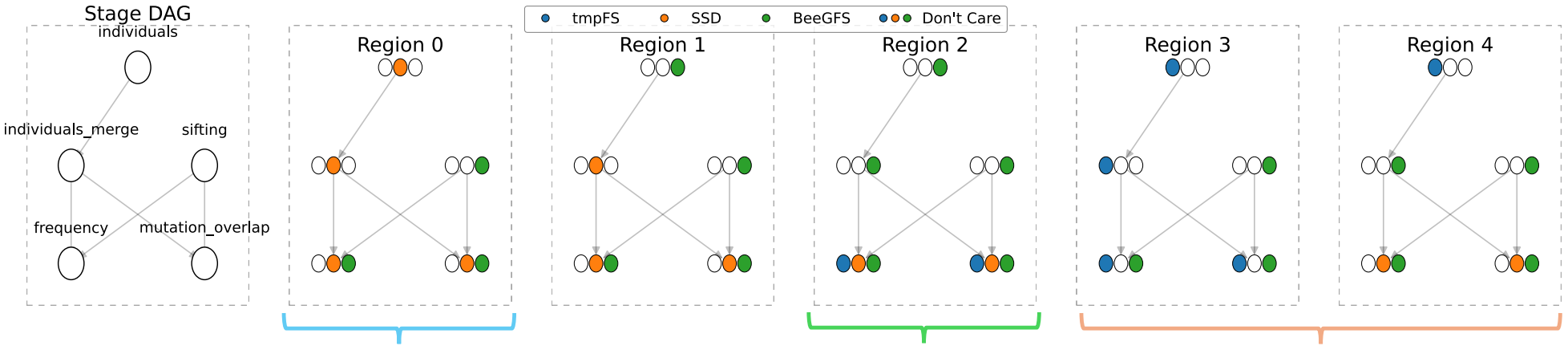


QoSFlow orders configs near-perfectly — +27.38% over the best heuristic (LTL)

DIRLAB

ordered by region's median makespan

Fastest

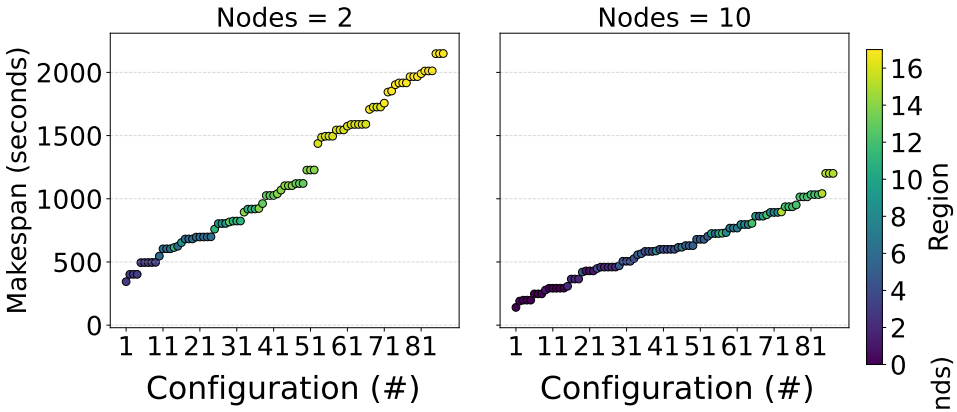


- individuals, merge → **SSD** *(critical)*
- sifting → **BeeGFS** *(critical)*
- frequency, mutation → **SSD or BeeGFS** *(flexible)*

more flexibility for final two stages

tmpFS → **BeeGFS** transition *increases data movement cost*

Users see which storage decisions matter — and where they have flexibility

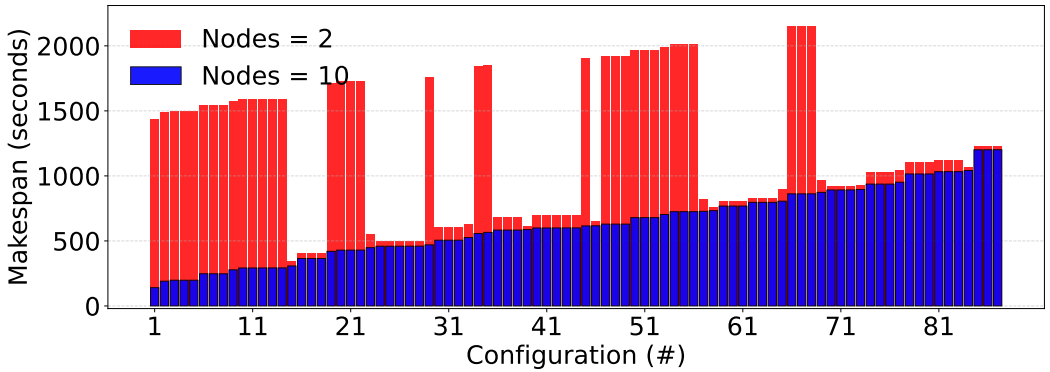


✅ **Regions stay tight at different scale**

- forms clear inter-region steps
- finds good partitions regardless of scale

Region structure is robust — rankings are scale-specific

vary node scale: ***10 → 2***

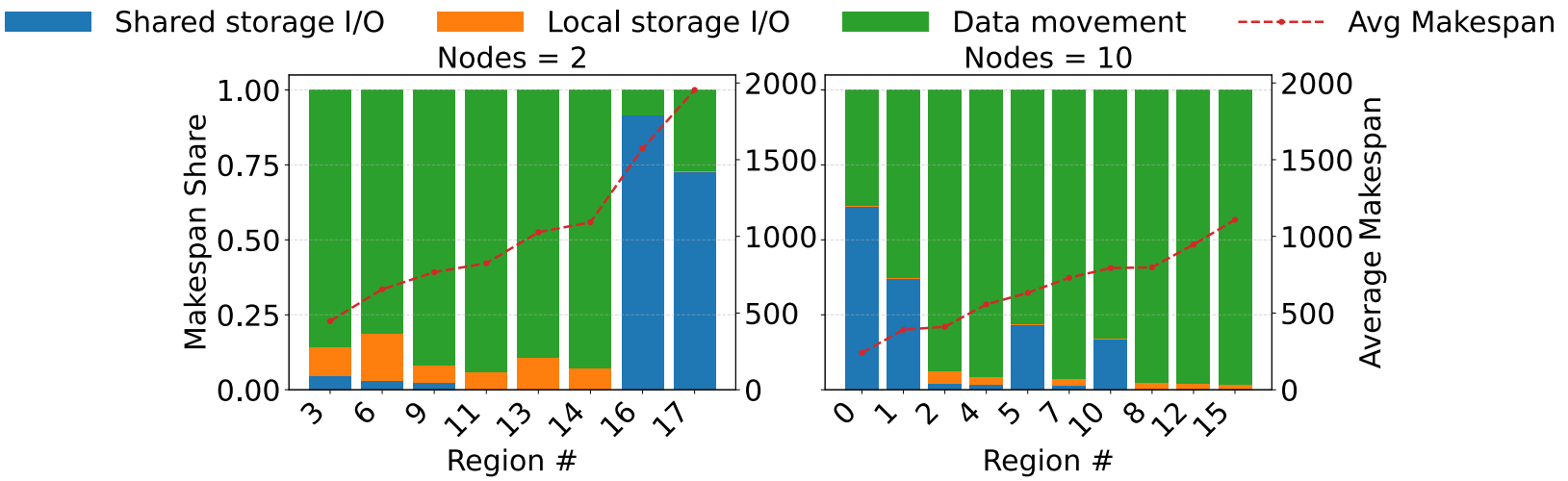


⚠️ **Configurations re-rank across scales**

- A top config at 10 nodes can fall to middle at 2 nodes
- Bottleneck shifts as parallelism changes

→ **regions must be formed per scale**

DIRLAB



2 nodes, Data movement dominates

- Top regions are movement-bound
- Stage-in/out cost is the limiting factor

10 nodes, Execution I/O dominates

- Top regions are shared/local I/O bound
- More tasks per stage → more concurrent I/O pressure

The dominant bottleneck shifts with scale — QoSFlow makes the shift visible

Q1 · Capacity-bound ✅
"Best config given ≤ N nodes?"
→ Returns optimal config at constraint

Q2 · Allowed tiers ✅
"Best config using only tiers {A, B}?"
→ Restricts feasible space, ranks

Q3 · Deadline + exclusion ⚠️
"Meet deadline T, but exclude tier X?"
→ **Correctly returns infeasible**

Q4 · Tier fallback ✅
"Tier X is down — best alternative?"
→ Excludes X, re-ranks, returns best

QoSFlow handles diverse QoS queries — including correctly flagging infeasibility

DIRLAB

Interpretable, scale-aware QoS modeling for distributed workflows

1. Workflow QoS is an I/O problem

Storage-tier choice causes **8.5×** makespan variation on the same workflow

2. Sensitivity → Regions → Recommendations

Partition the config space into performance-equivalent regions with interpretable rules

3. Region transferability is workflow-dependent

Robust *per scale* — but rankings shift across scales; QoSFlow makes that visible

+27.38% better ordering than best heuristic *(1kgenome)*

Approach validated on 3 real workflows

Paper

Artifact

Md Hasanur Rashid · mrashid@udel.edu

Thank you · Questions?

University of Delaware · Pacific Northwest National Laboratory · Illinois Institute of Technology