Distributed Parallel Datalog for Lifted Planning

Oliver Joergensen Dominik Drexler Jendrik Seipp

HSDIP Workshop, ICAPS 2026

Linköping University, Sweden

Lifted Planning

- Grounding can be infeasible: exponentially many ground atoms and actions
- Lifted planners search directly on the first-order representation
- Core computations are Datalog queries: successor generation, axioms, FF-RPG heuristic

state expansion = Datalog program evaluation

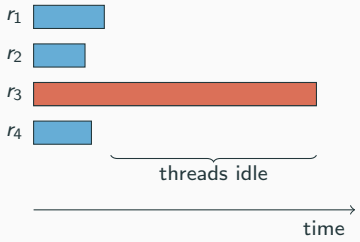
Parallelism Today: Threads

Tyr: thread-parallel Datalog evaluation

- rules of a stratum evaluated concurrently
- 2–6× speedup on 8 cores

Bottleneck: rule skew

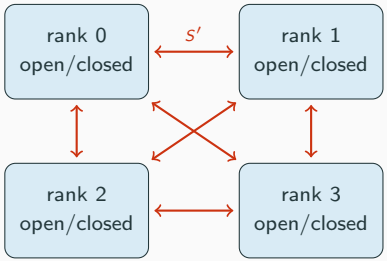
- one rule dominates the stratum
- extra threads sit idle



Our Idea: Distribute the Search

N MPI ranks, private memory, message passing

- each rank: own open list, closed set, Datalog engine
- work distribution function $\omega : \mathcal{S} \to \{0, \ldots, N-1\}$
- successors forwarded to their owner rank
- owner alone deduplicates and expands



no shared memory $\Rightarrow$ no rule-skew bottleneck, scales beyond one node

Which Hash Function?

$$\omega(S) = h(\phi(S)) \bmod N$$

Uniform hash (Zobrist)

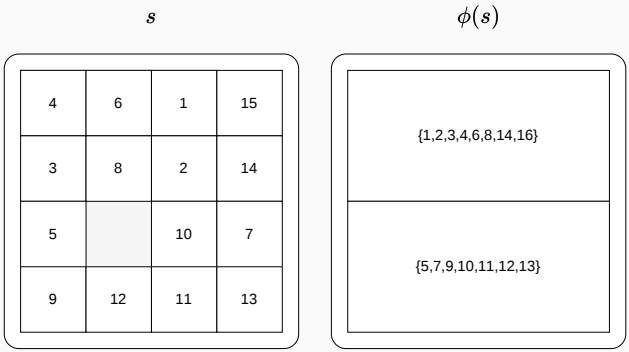
- perfect load balance
- $\approx \frac{N-1}{N}$ of states sent away

Structure-aware hash (AZH)

- co-locates related states
- built from DTGs

Problem: DTGs require a fully grounded task

Abstraction-Based Hashing, Illustrated



moves within a half: same feature $\phi(S)$, no communication

Invariants as a Lifted Proxy

A planning invariant ι: lifted atom templates such that exactly one ground instance holds per group, in every reachable state

$$\iota = \{\mathsf{at}(x, l),\ \mathsf{in}(x, t)\}$$

- rigid variable x (package): selects the mutex group
- counted variables l, t: members within the group

- computable from the lifted task — no grounding
- mutex groups underlie the finite-domain variables of DTGs

Invariant-Based Zobrist Hashing

Abstraction: atoms covered by a selected invariant

$$\phi_{\mathcal{I}}(S) = \{(i, p) \mid p \in \overline{P}(S),\ p \text{ in a mutex group of } \iota_i\}$$

Hash and router:

$$H_{\mathcal{I}}(S) = \bigoplus_{(i,p)\in\phi_{\mathcal{I}}(S)} v(i, p) \qquad \omega_{\mathcal{I}}(S) = H_{\mathcal{I}}(S) \bmod N$$

- $v(i, p)$: deterministic FNV-1a hash — no shared random table
- incremental update per transition: $|\mathit{eff}(\bar{a})|$ XOR operations

Invariant Selection

1. estimate each invariant's group size from object counts
2. rank invariants by estimate, ascending
3. greedily select mutex groups until granularity $\geq \beta N$

- oversampling factor $\beta > 1$ buffers against collisions
- predicate baseline: same scheme with predicates instead of groups

Experimental Setup

- Python prototype on top of Tyr (C++ Datalog core)
- Hard-To-Ground suite: 1058 instances
- fixed budget: $N \cdot k = 16$ cores, 30 min, 32 GiB

Tyr-16	1 search, 16 threads
Dist-1-16	16 ranks $\times$ 1 thread
Dist-4-4	4 ranks $\times$ 4 threads

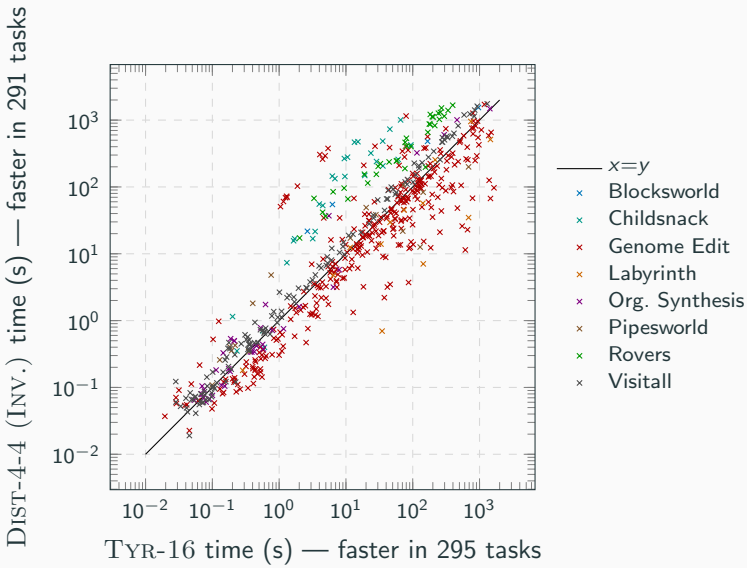
Rand.	uniform
Pred.	predicate hash
Inv.	invariant hash $\omega_{\mathcal{I}}$

Coverage and Memory

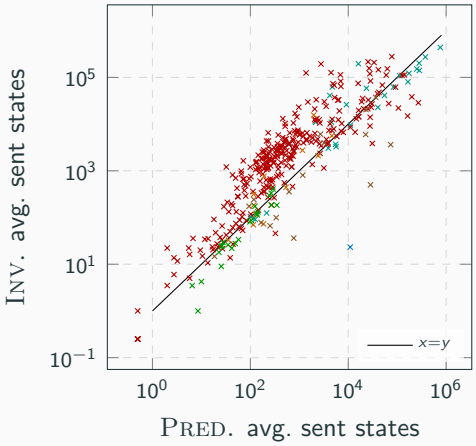
	#C	#M	MS
Powerlifted	623	257	439
Tyr-1-CPP	625	65	399
Tyr-16	**584**	72	303
Dist-1-16 (Pred.)	582	113	**327**
Dist-1-16 (Inv.)	550	56	312
Dist-4-4 (Inv.)	555	**35**	307
Dist-4-4 (Rand.)	532	54	295

- invariant router: fewest out-of-memory failures
- random router: worst across the board
- Python overhead: compare Tyr-1-CPP vs. prototype rows

Threads vs. Ranks Is Domain-Dependent



Communication: Inv. Sends More Than Pred.



Inv. sends more in 277 tasks,
Pred. in 95

- invariants track exactly the atoms that change most
- rarely changing predicates leave the hash unchanged
- granularity often coarser than ideal

Takeaways

- hash-distributed lifted search is feasible: first framework, computable without grounding
- no allocation of a fixed budget dominates: threads vs. ranks is domain-dependent

Future work

- C++ port (Implemented and showing promise)
- granularity sweep over β, alternative distribution strategies
- multi-node experiments allows using more cores that available on one CPU

Thank you! Questions?