Methods of Partitioning a Parallel Episodic Memory

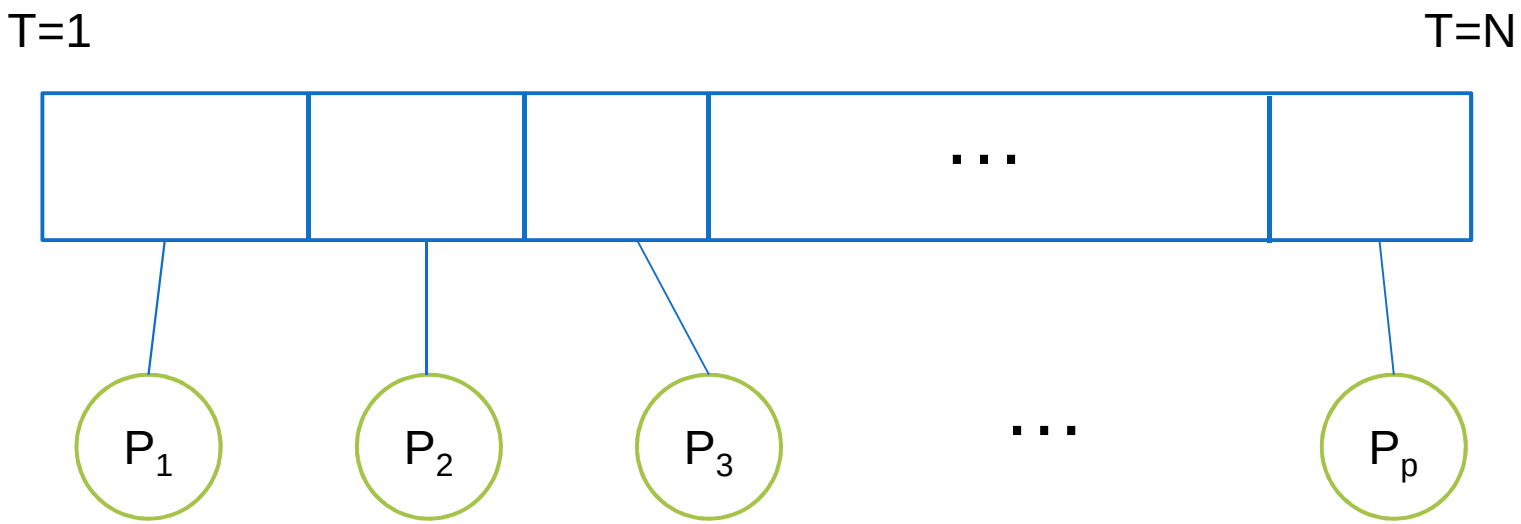
Aaron Mininger
mininger@umich.edu
and
James Kirk
University of Michigan

Motivation

- For long-lived agents, worst-case epmem query times grow linearly
- Parallelizing epmem would allow long-lived agents to remain reactive longer
- Evaluate how effective parallelizing epmem would be

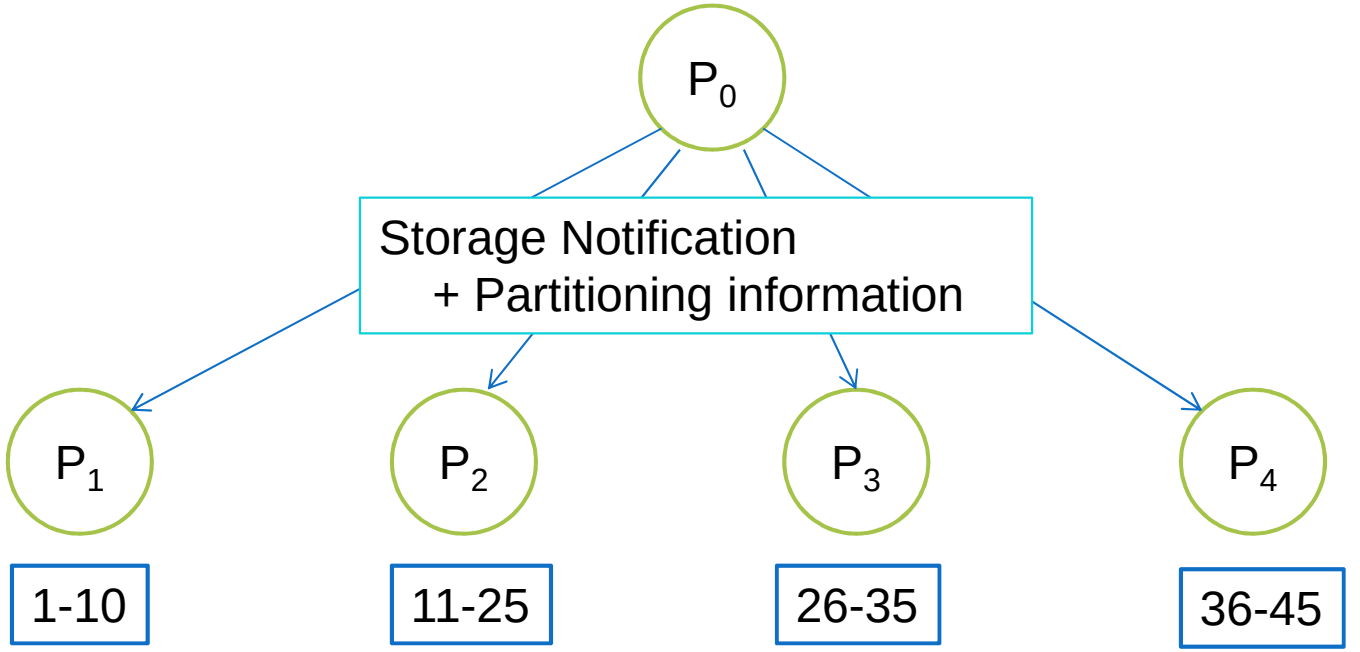
Parallel Implementation

- Epmem is partitioned and spread among the worker processors



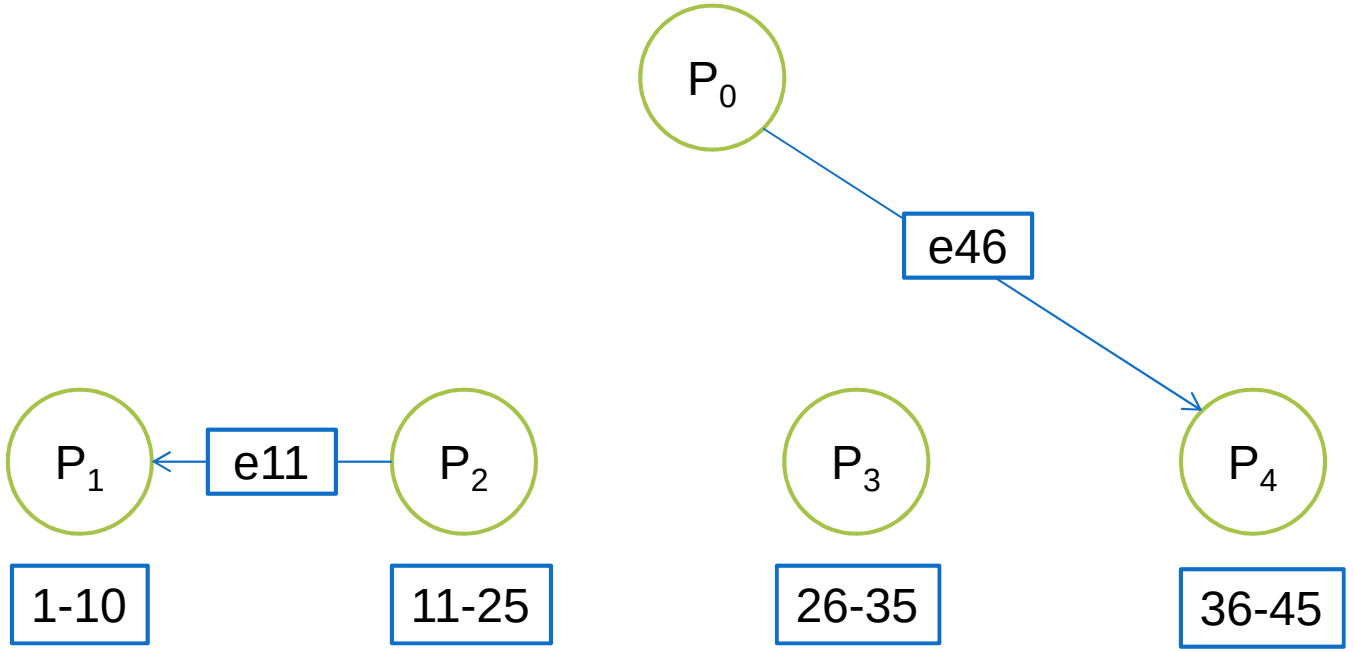
Parallel Storage

- The master notifies all workers that a new episode is being stored
- Includes information about the current partitioning scheme



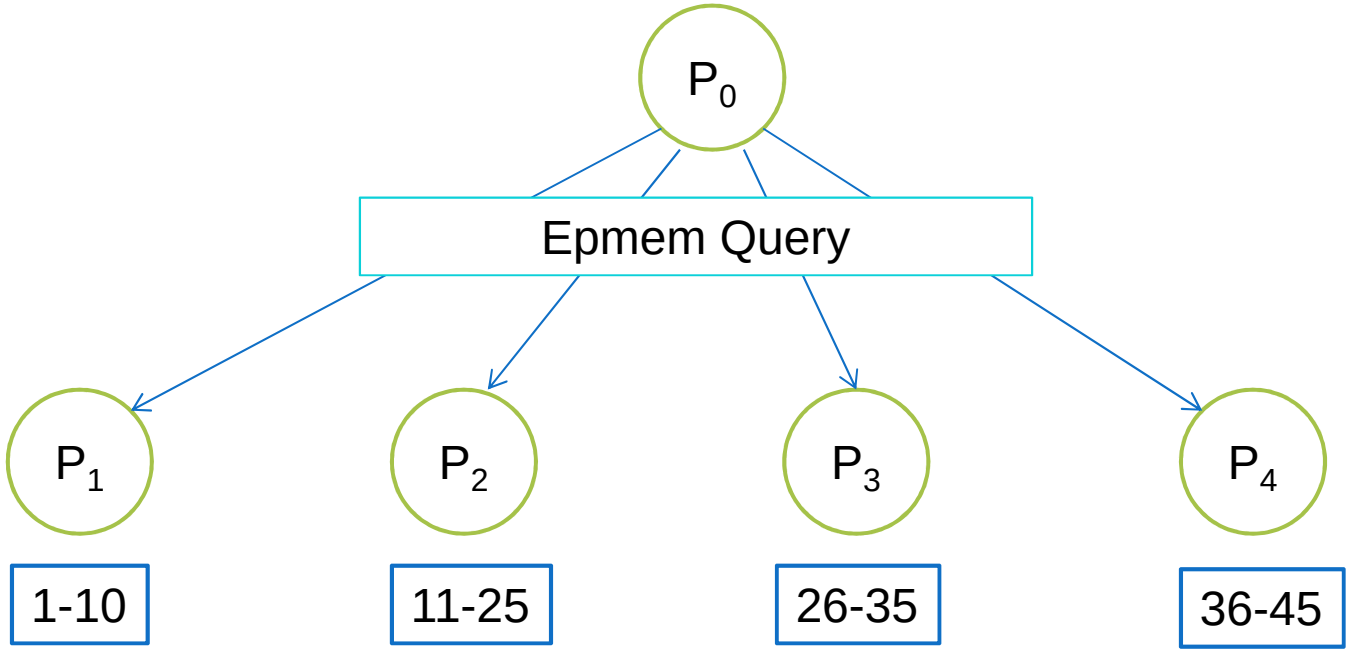
Parallel Storage

- Each worker does a local decision to send its oldest episode to the next processor
- At most 1 episode is passed down by each worker



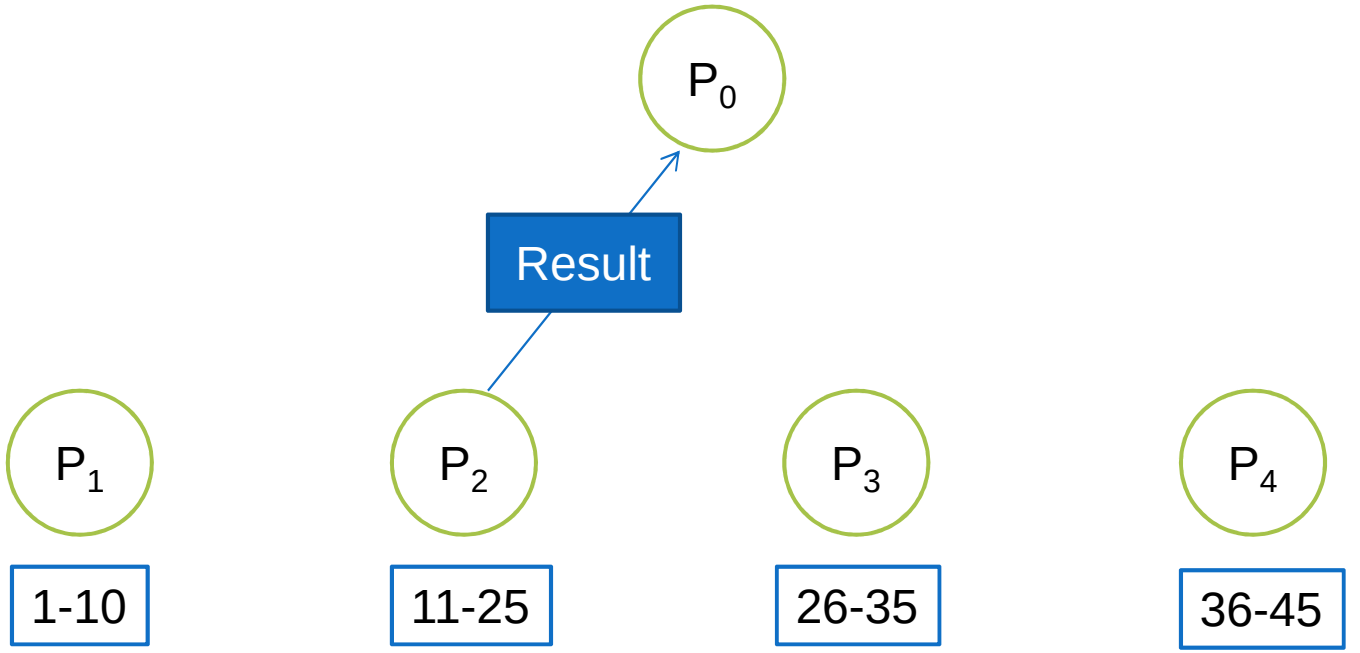
Parallel Retrieval

- The master sends the cue to every worker



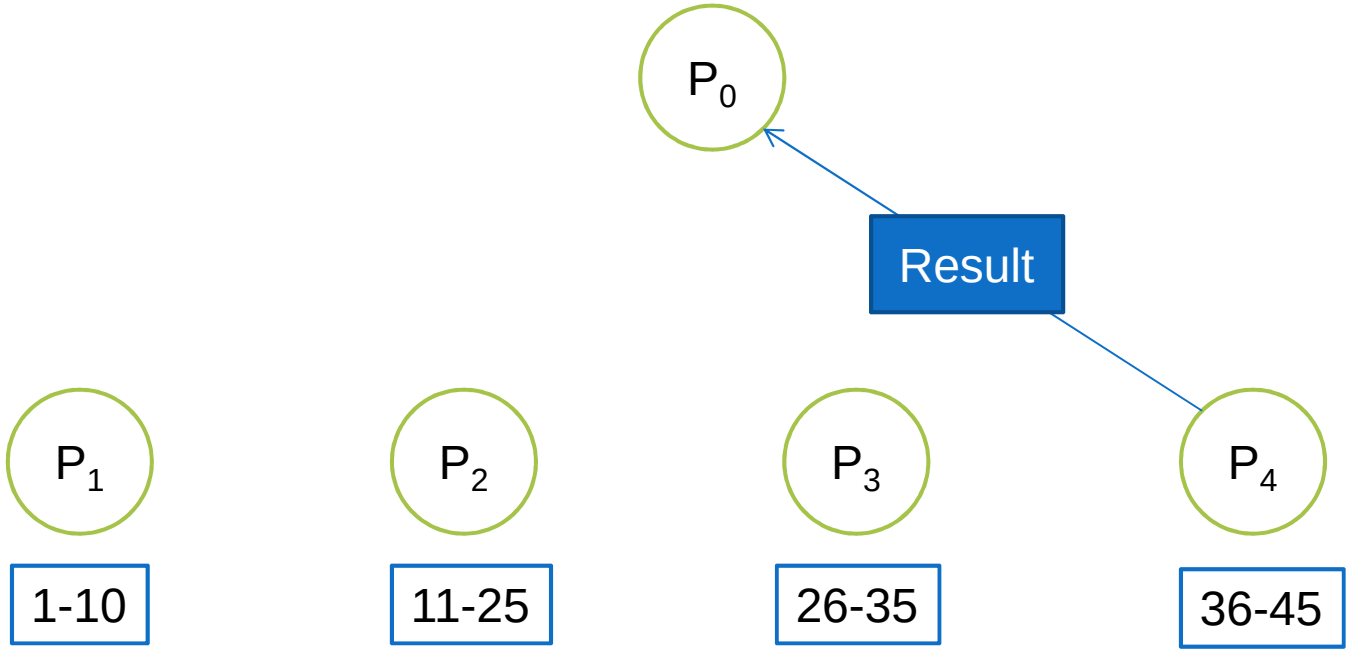
Parallel Retrieval

- The master sends the cue to every worker
- Each worker reports its result back



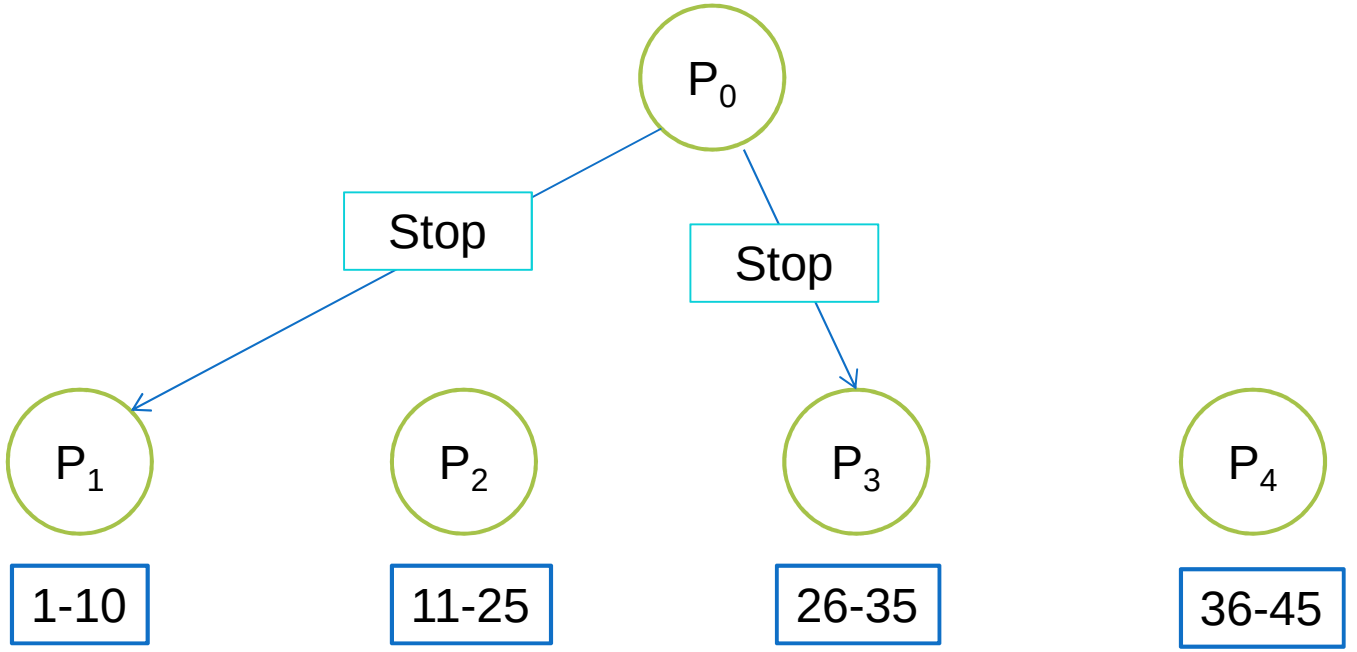
Parallel Retrieval

- The master sends the cue to every worker
- Each worker reports its result back
- The underlying search algorithm remains unchanged



Parallel Retrieval

- The master sends the cue to every worker
- Each worker reports its result back
- If a global best has been found, the master tells the rest to stop

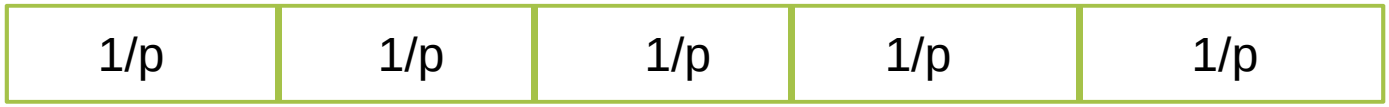


Partitioning

- Decide how to spread the episodes among the processors
  - Worst case is a search through all episodes
  - Results must be biased towards recency
  - Characteristics of the agent have a large impact on possible speedup

Partitioning Strategies

- Even



- Exponential

- Shift

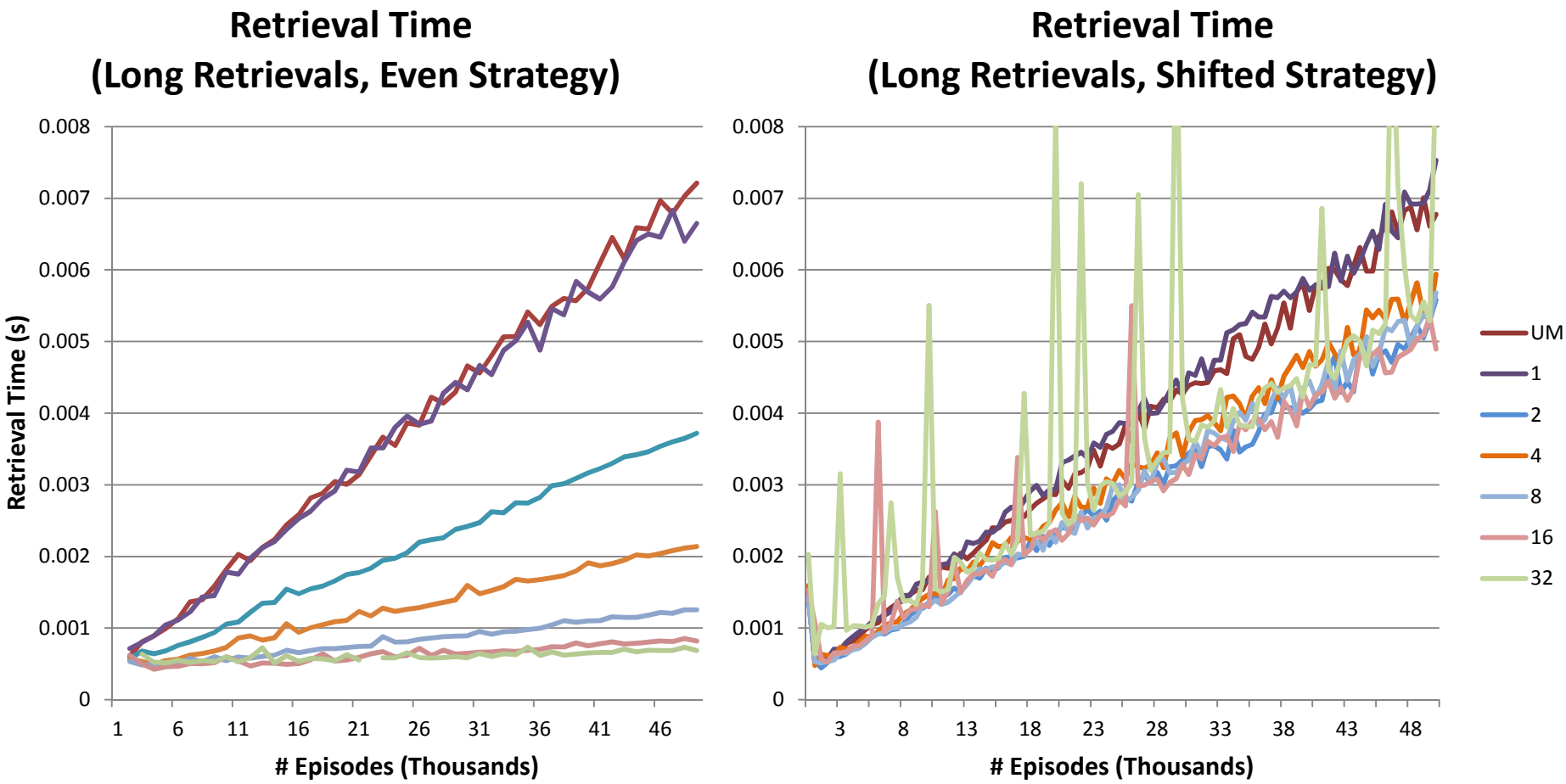
Experiments

- Tests using 1-32 processors and an unmodified (UM) baseline comparison
- Supercomputing cluster (flux)
- Ran for 50,000 cycles
- Performed a query at every 1,000 cycles
- Evaluated storage times at every 1,000 cycles

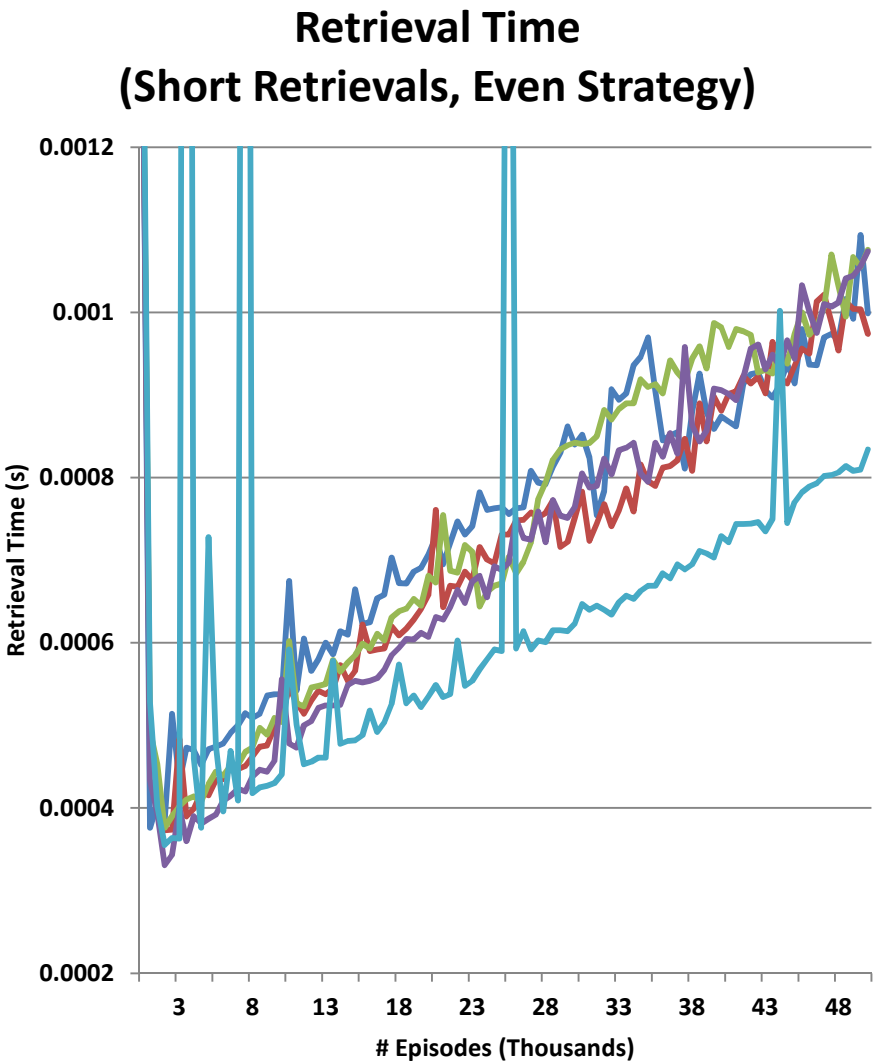
Experiments

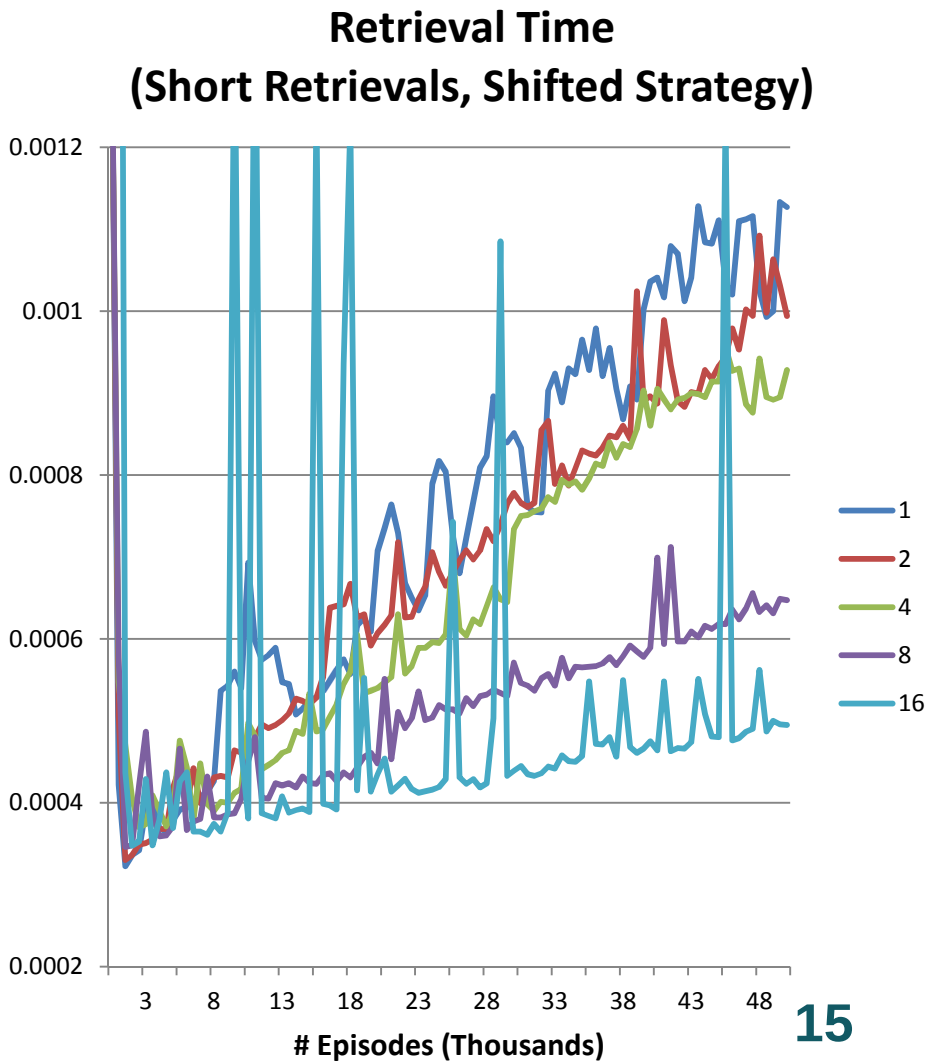
- Retrieval Types
  - Long – Oldest 10%
  - Short – Most recent 10%
  - Random – Even distribution

Long Retrievals



Short Retrievals



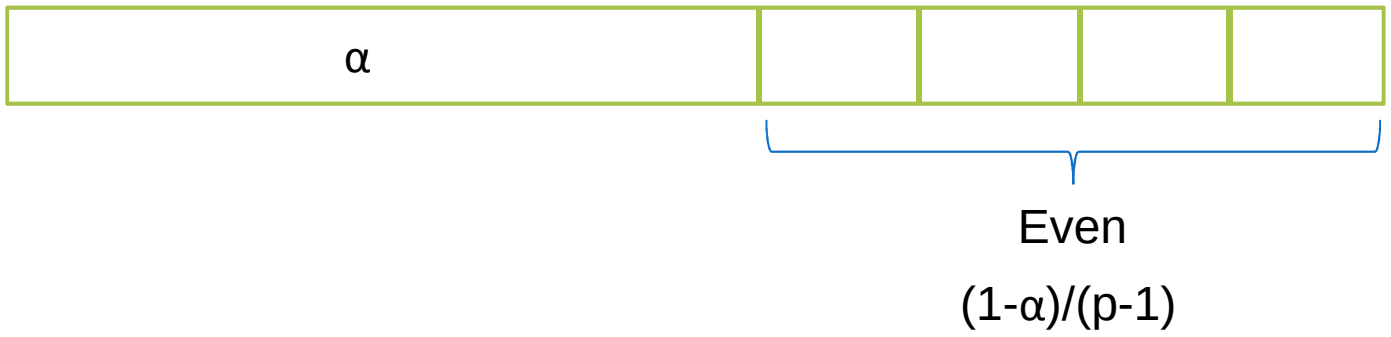


Random Retrievals

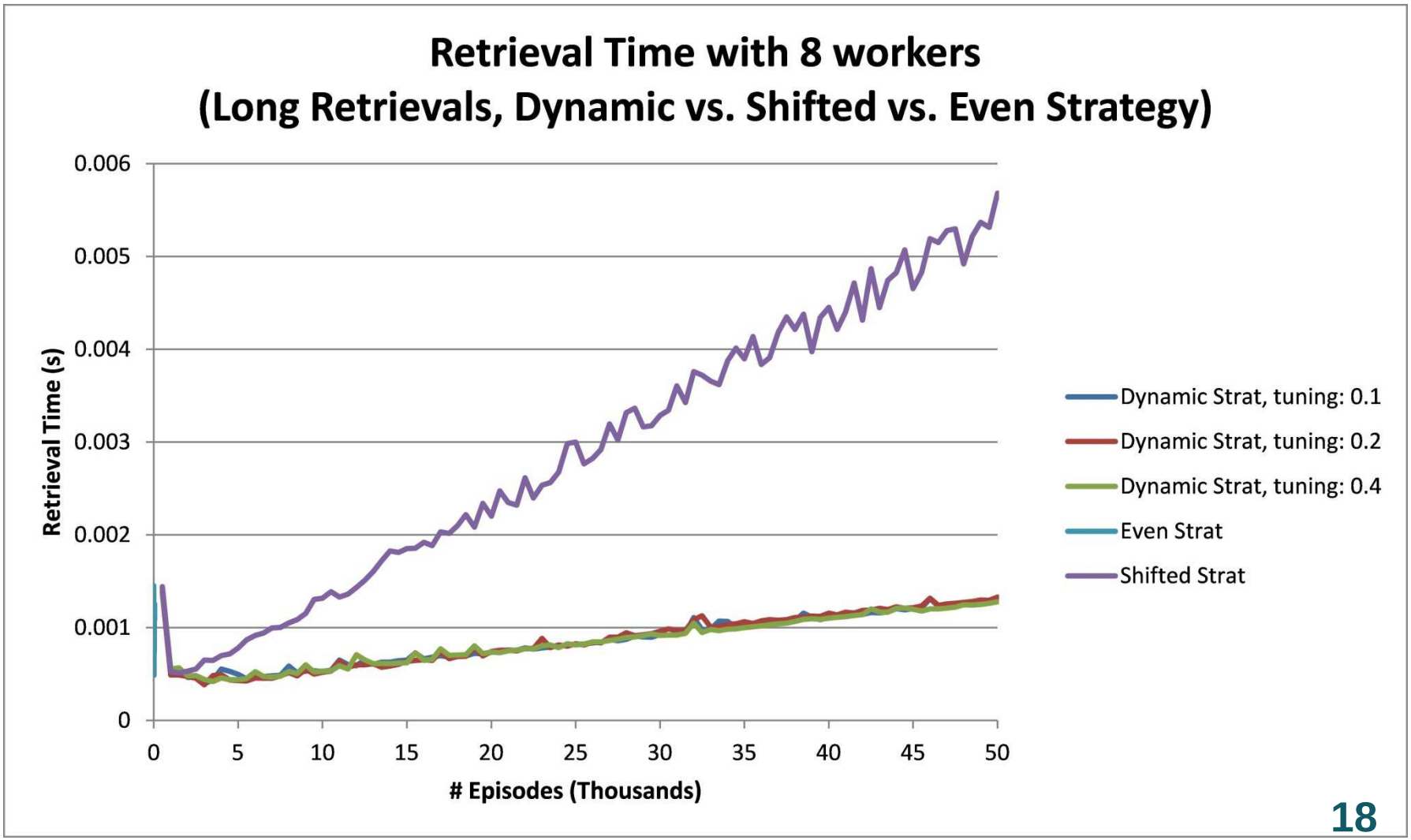
- Even strategy bounds the worst case
- Shifted strategy does not do well

Dynamic Partitioning

- Tune alpha based on the past performance
  - Long retrievals – reduce alpha
  - Short retrievals – increase alpha

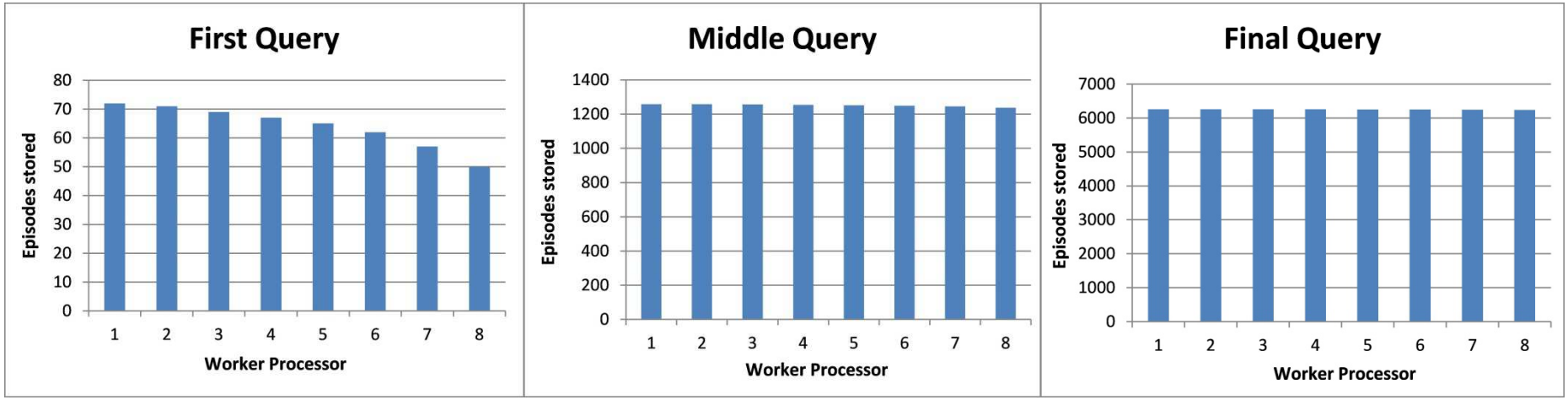


Dynamic Partitioning

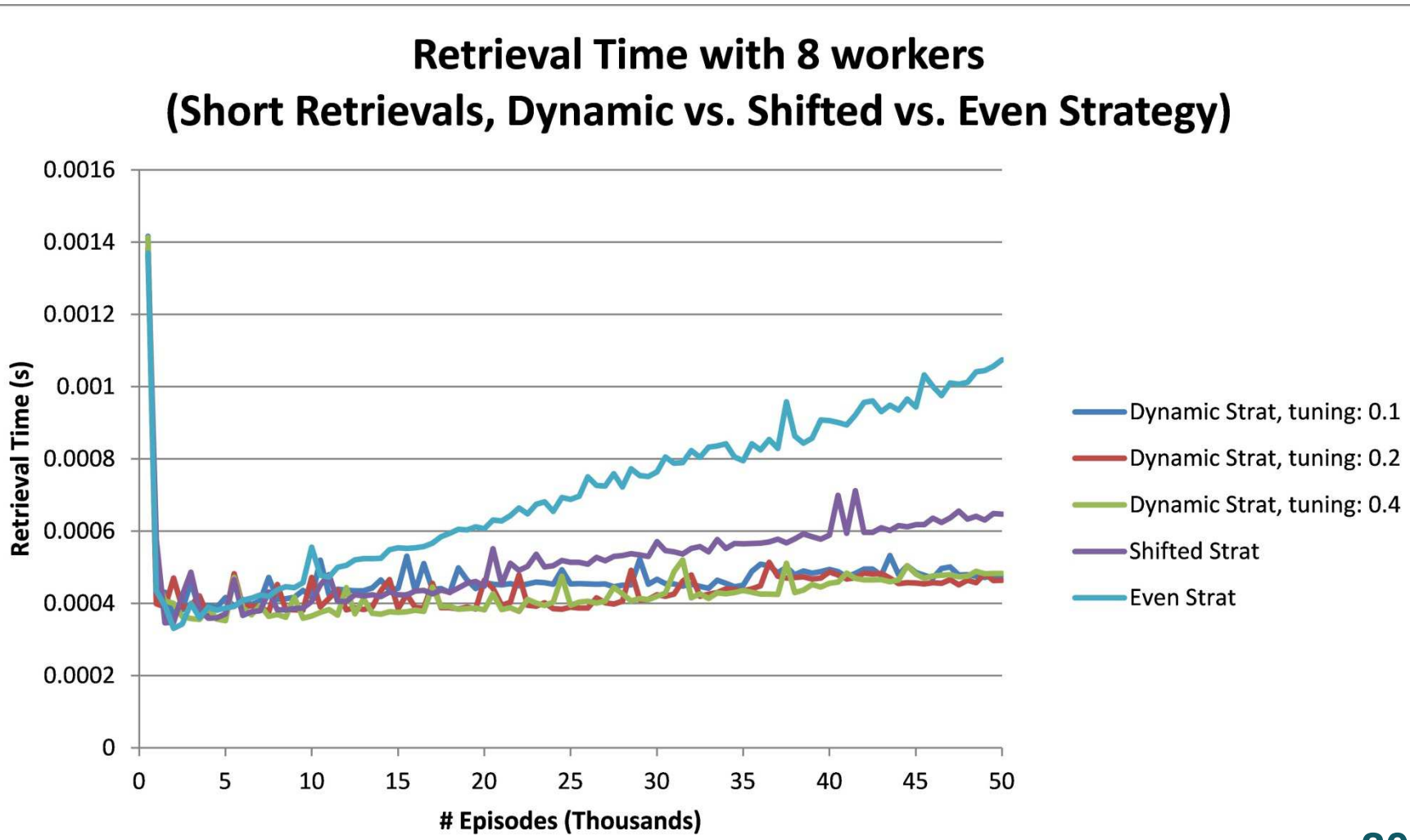


Dynamic Partitioning

- Long Retrievals

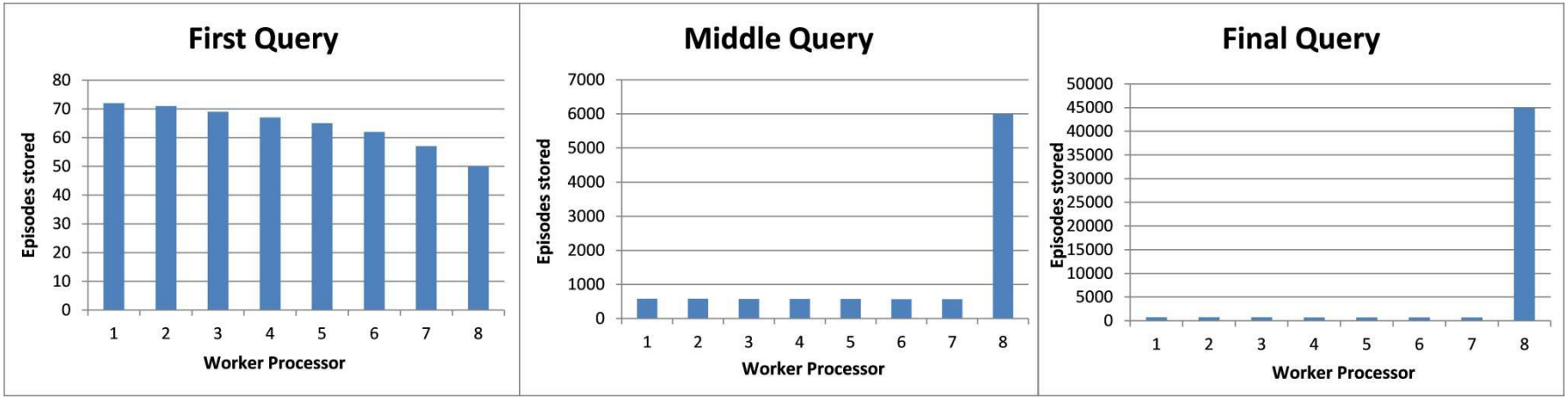


Dynamic Partitioning



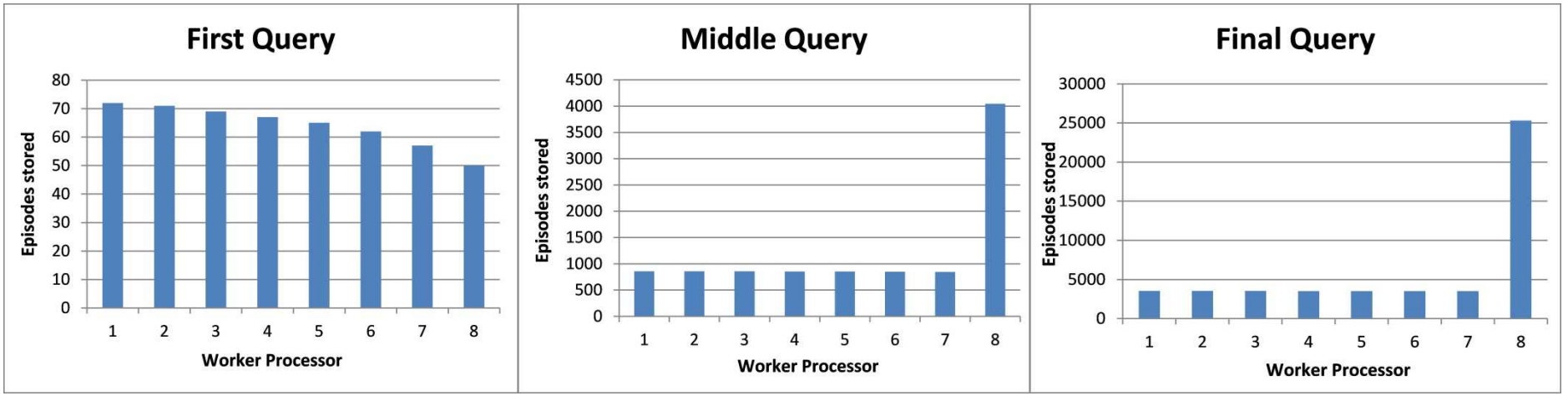
Dynamic Partitioning

- Short Retrievals

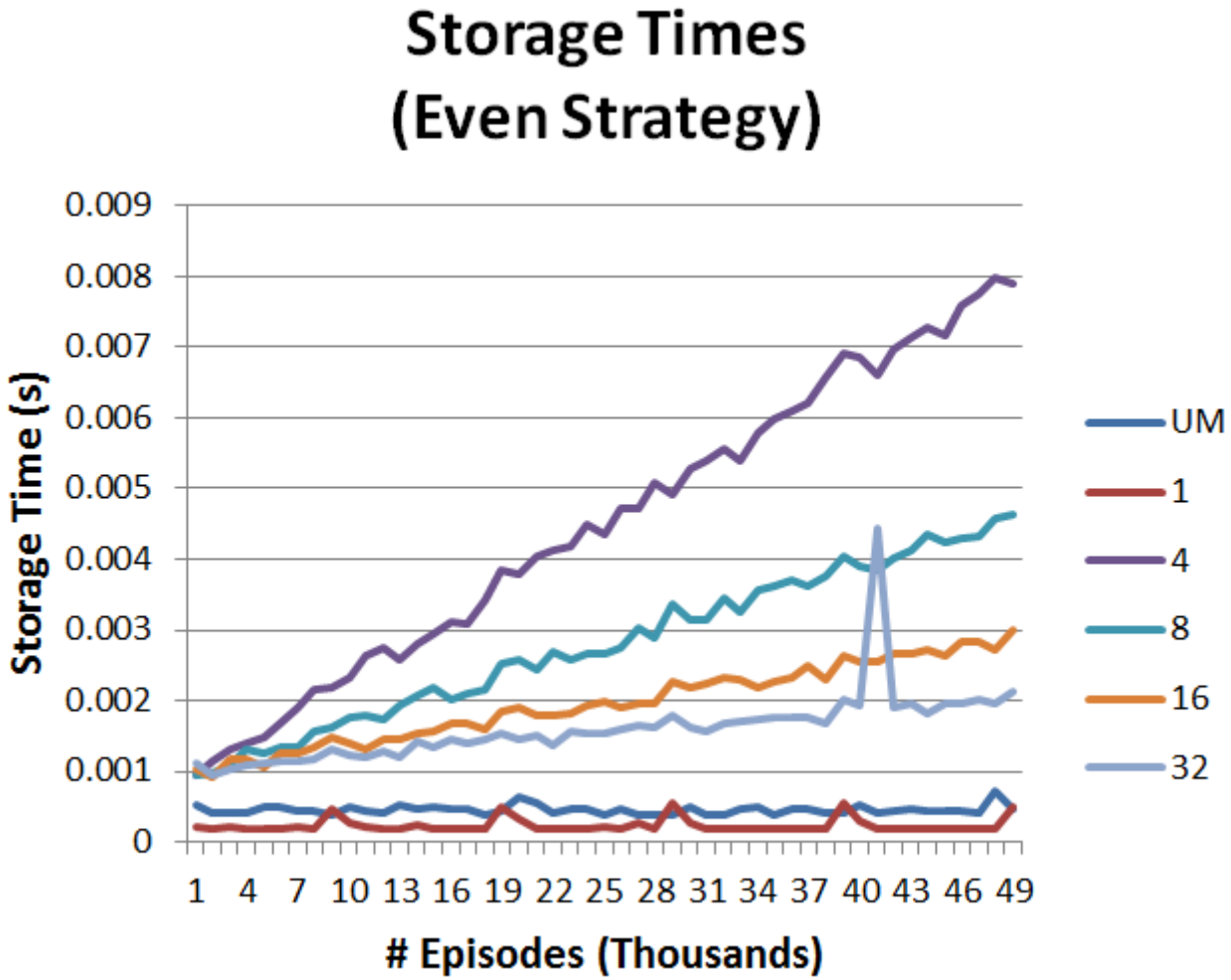


Dynamic Partitioning

- Random Retrievals



Effect on Storage Time



Nuggets

- Successfully implemented a parallel version of epmem
- Imposes minimal overhead for queries
- Created a single strategy that adjusts to different use cases

Coal

- Did not evaluate on complex agents
- Expensive storage times
- Poor speedup in most cases

Long Retrievals, Dynamic Partitioning

# Procs	1	2	4	8	16	32
Speedup	1	1.80	3.32	5.49	8.86	10.51