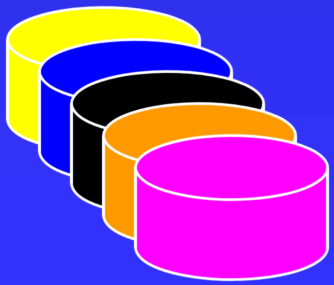
Data Placement on Tertiary Storage

Sunil Prabhakar

Joint work with Jiangtao Li

Purdue University



Introduction

- With current hardware, performance for data-intensive applications is constrained by I/O
- Random I/O performance is largely determined by the latency.
- For tertiary storage the latency is even more critical than for secondary storage.
- Given the current trends this problem will be aggravated in the foreseeable future.

Reducing Latency

- Caching
- Prefetching
- Scheduling
- Parallel I/O
- Compression
- Placement (replication & prefetching)
- …

Tertiary Storage Placement

- Goal:
  - Reduce switching of media
  - Reduce seek latency
- Two sub-problems:
  - Medium allocation
  - Intra-medium placement

Related Work

- Specific domains (arrays, RDBMS, Images)
- Most placement work has focused on intra-medium placement.
- Recent work for tertiary storage has addressed the allocation problem, but under the assumption of independent access probabilities.
- This is not always a valid assumption (e.g. web pages, online manuals, multimedia databases)

Problem addressed

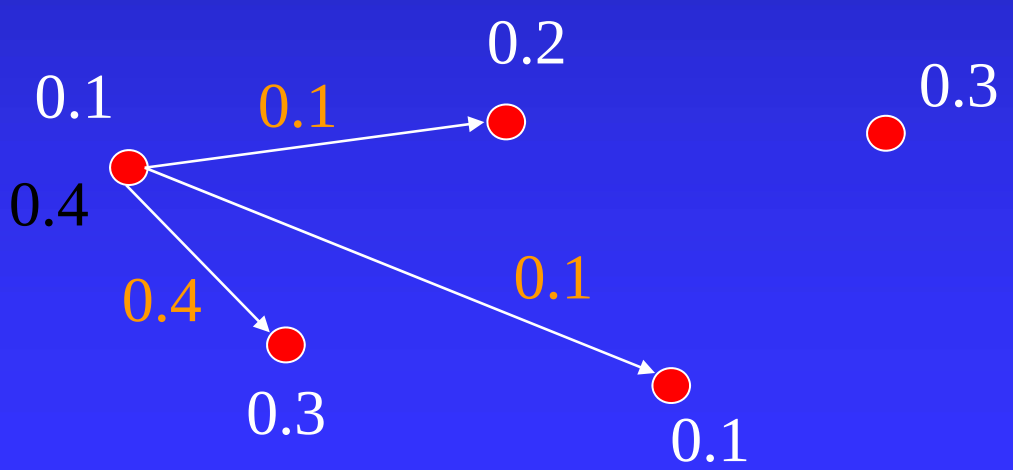
- Design of placement schemes with non-independent access patterns.
- Initially assume that the access pattern is known
- Focus on the allocation problem -- existing techniques for intra-medium placement
- Additional issues addressed:
  - Replication
  - Impact of secondary storage
  - prefetching

Access Patterns

- Use the notion of a browsing graph
  - Nodes represent objects
  - Node labels give the probability that an object is independently accessed
  - Directed edges between nodes have labels giving the probability that the edge will be traversed. E.g. edge $a \rightarrow b$ with probability p_{ab} represents the fact that object b will be accessed following an access for object a with probability p_{ab}.

Browsing Graph

- Birth Probability
- Edge Probability
- Death Probability



Data Placement Schemes

1. ***Birth Probability*** Scheme
   - Place objects in decreasing order of birth probability (independent placement)
   - This is known to be optimal if we ignore relationships between objects [2].
2. ***Static Probability*** Scheme
   - Same as above, except that we use the static probability for determining placement (cf Google).

Data Placement Schemes

3. ***Edge Merge*** Scheme
   - Place strongly connected neighbors on the same medium
   - Edges are merged in decreasing order of probability.
   - Birth probability and edge probabilities of merged object are calculated.
   - Merge edges as long as the total size of the merged objects is smaller than medium capacity.
   - Merged objects are now allocated to media in decreasing order of the cumulative static probability.

Data Placement Schemes

5. ***Hot Edge Merge*** Scheme

- Identical to Edge Merge, except that only edges that have more than a threshold probability are merged.
- Objective is to produce media with very high probability of being loaded permanently.

Data Placement Schemes

6. ***Birth Hop*** Scheme

- Initially, place highest birth probability object on an empty medium.
- Repeatedly add the object with the highest birth probability or edge probability from objects already on that medium.
- Once the medium is full, repeat the above steps for the remaining objects.

7. ***Static Hop*** Scheme

- Identical to above scheme, except that we use static probability instead of birth probability.

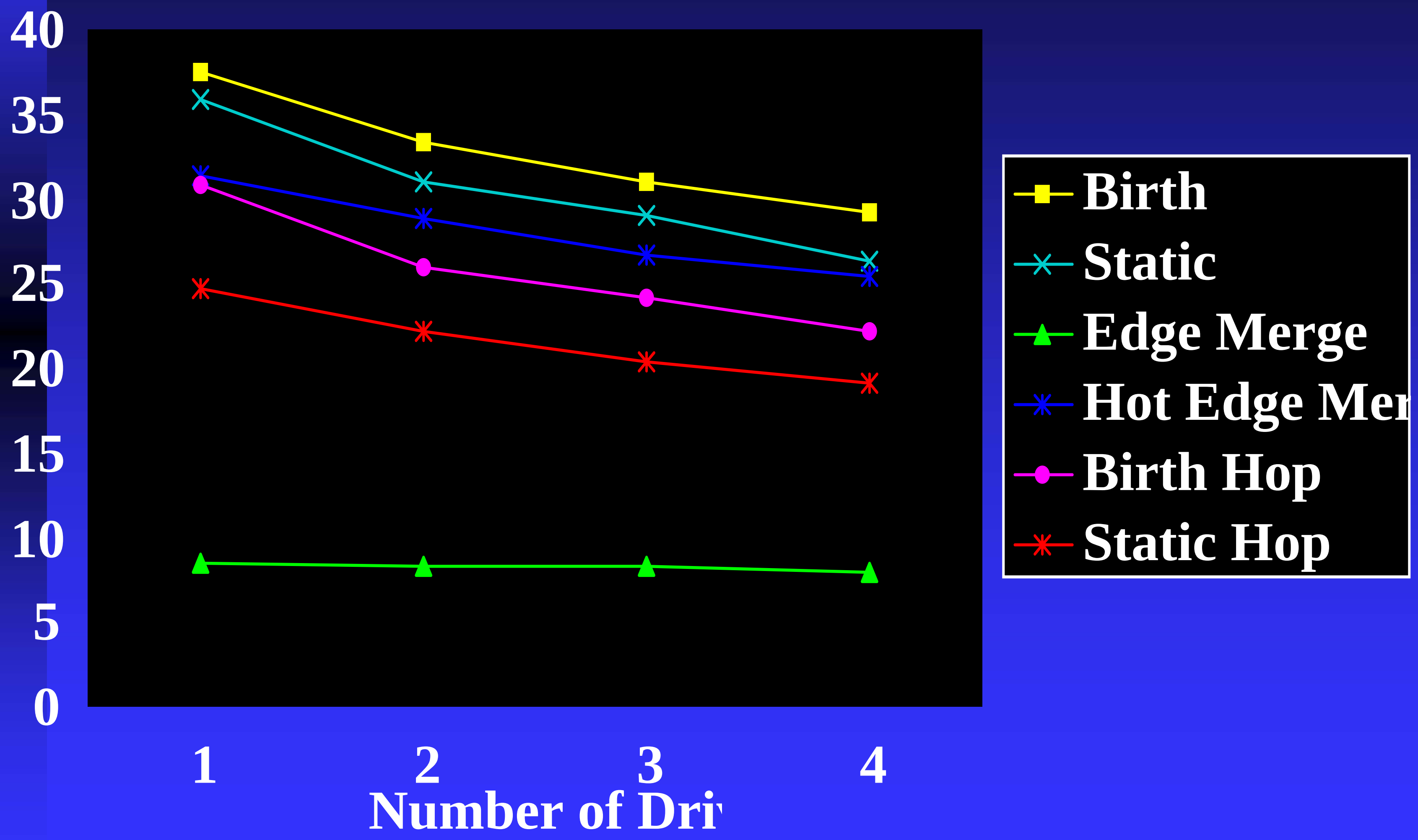
Other Issues

- Adaptive Placement
  - Use observed pattern of access to periodically reorganize data placement.
- Impact of Secondary Storage
  - Handle as above -- “observe” pattern at tertiary level.
- Replication
  - No cost replication when objects belong to multiple clusters -- use available free space.
- Prefetching
  - Once a medium is loaded on a drive, some high probability objects are prefetched.

Experimental Results

- Simulation (CSIM) of Ampex DST drives.
- 10,000, objects (100MB each)
- 2000 tapes of 2GB each -- 4TB total
- 4, 5GB disks -- 20 GB total
- Birth probability follows a Zipf distribution
- Objects divided into clusters (5 and 20 per cluster)
- 5% of objects are outliers
- Death probability uniformly chosen (0.05 -- 0.2)
- Edge probabilities are uniformly distributed.
- Average response time for 1000 requests.

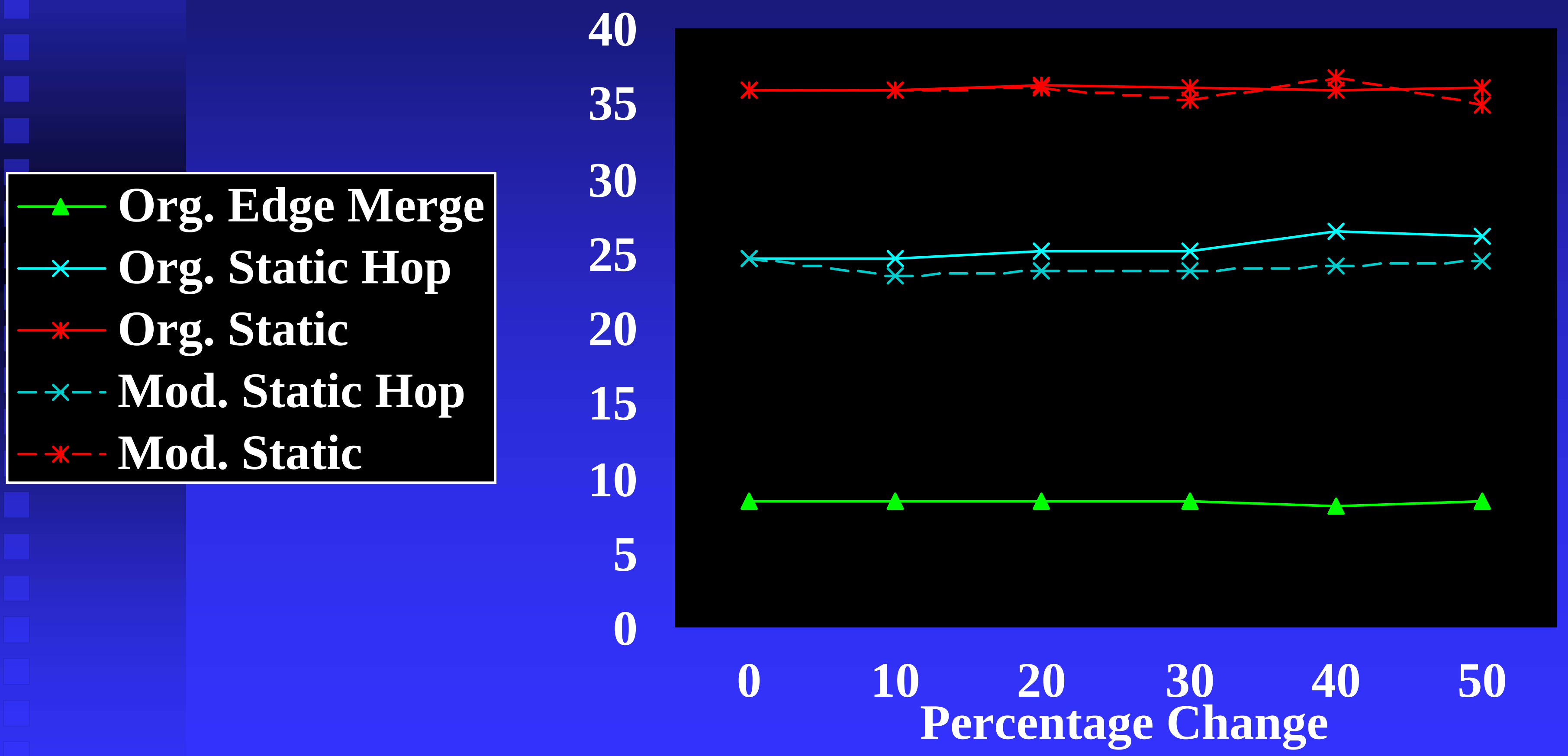
Performance



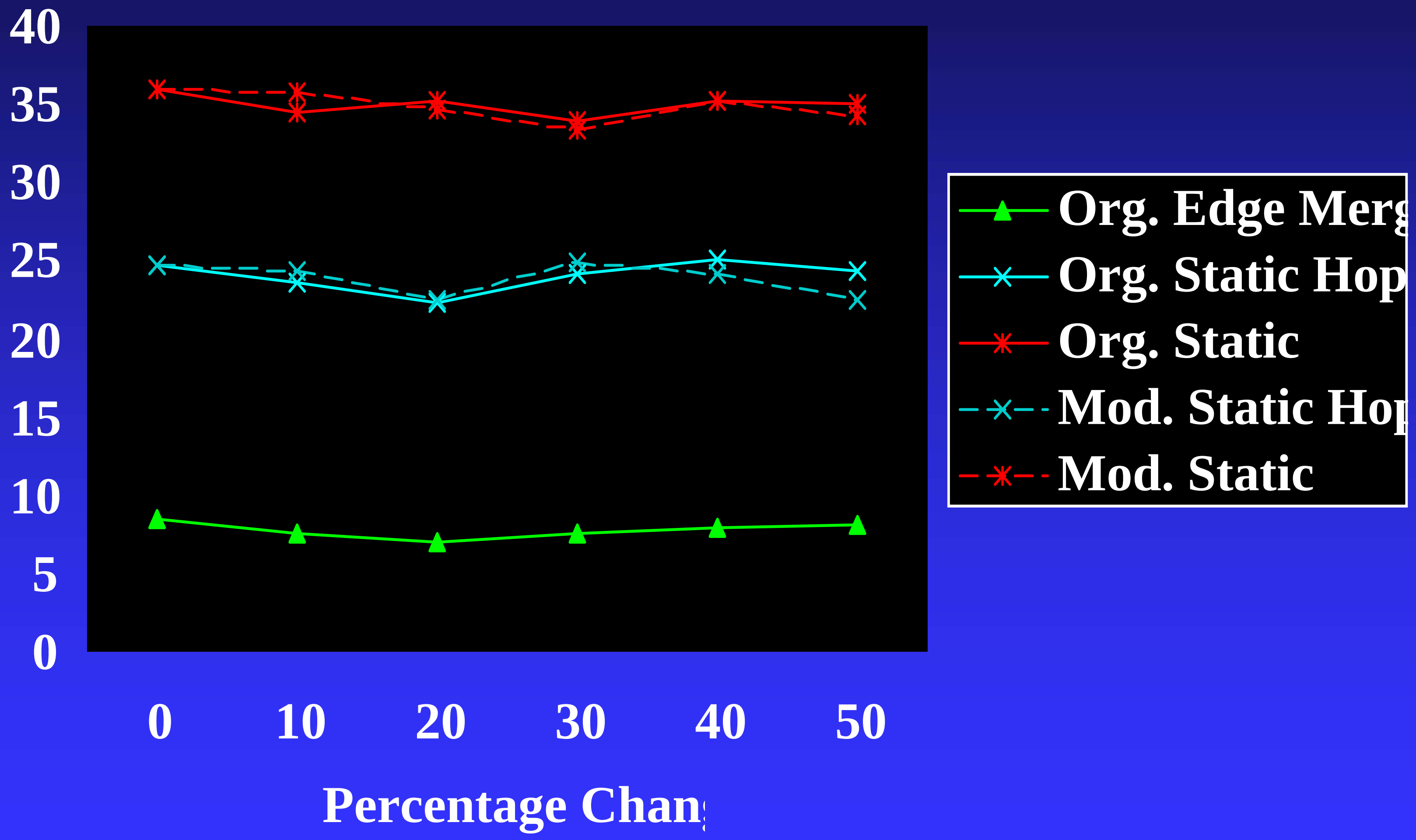
Sensitivity to Access Pattern

- Study the impact of variations in the access pattern.
- Consider variations in:
  - Node probabilities
  - Edge probabilities
  - Cluster compositions
- Test with original placement and also with modified placement (based upon observed pattern).

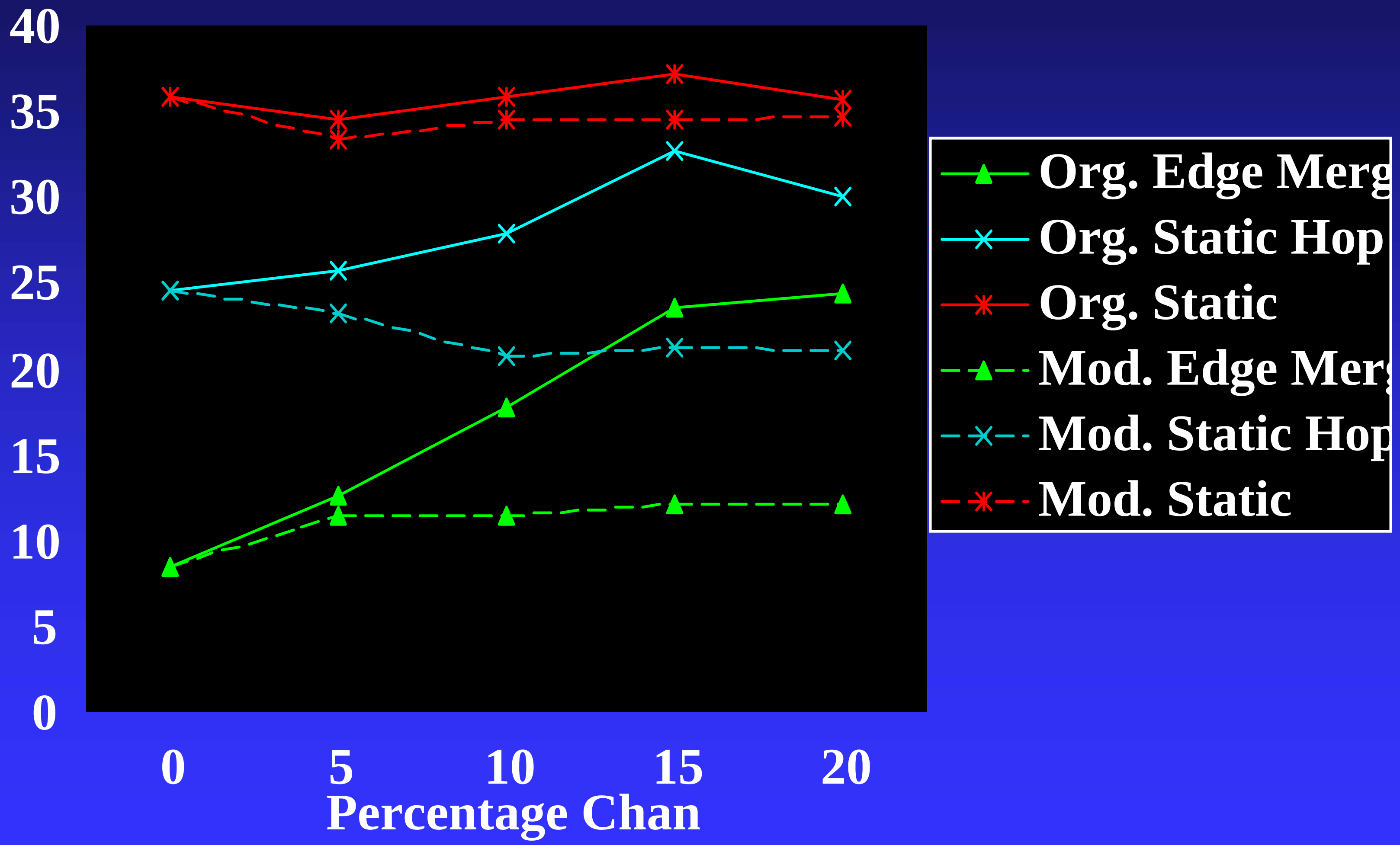
Variations in Edge Probabilities



Variations in Node Probabilities



Variations in Node Clusters



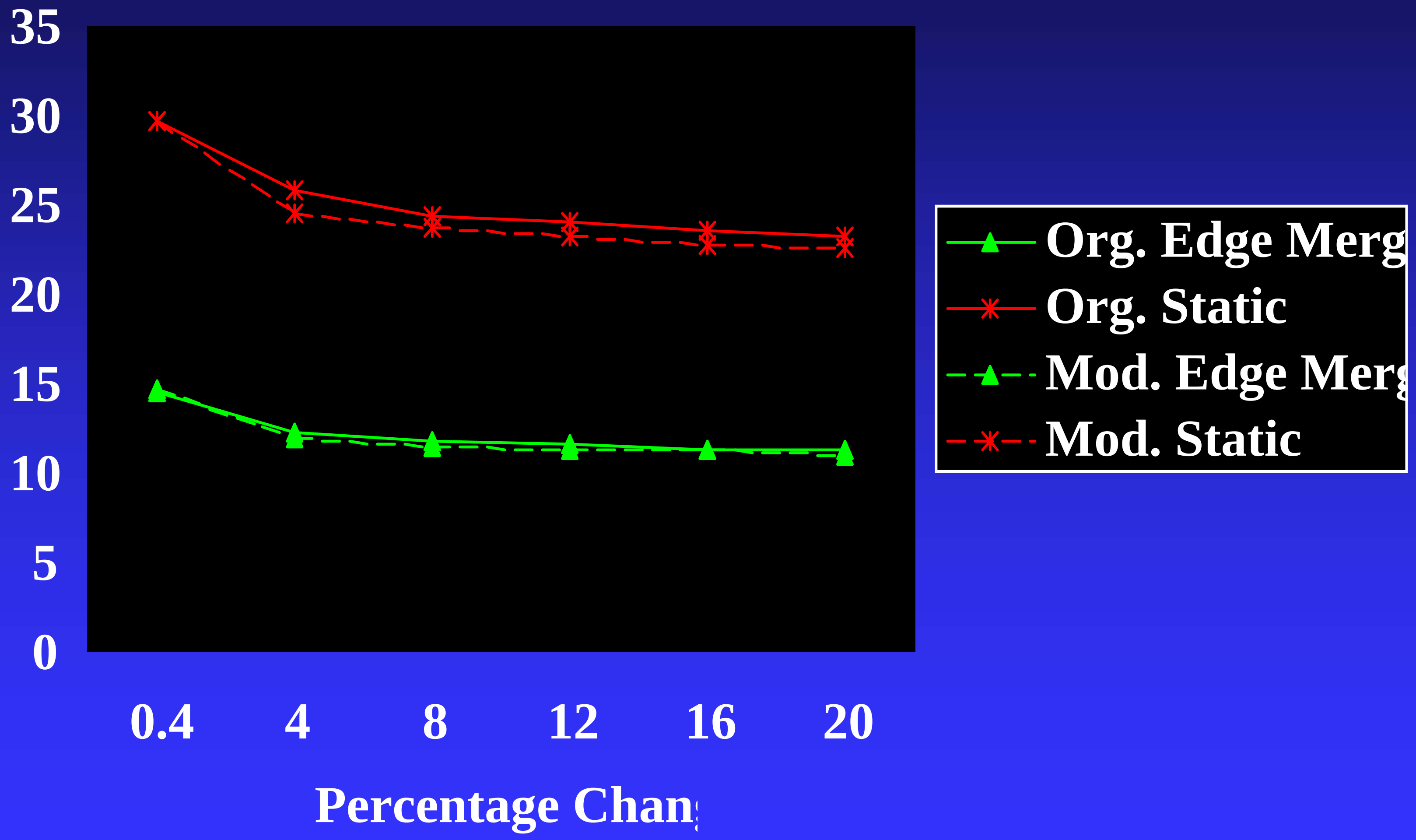
Access Pattern Variations

- The node and edge probabilities are less critical than the cluster composition!
- Therefore, it is important to be able to recognize the related objects.
- Changes in node and edge probabilities should not trigger re-organization -- Edge Merge is especially insensitive to these.

Impact of Secondary Storage

- The presence of a secondary storage buffer can have a significant impact on placement.
- High probability objects are likely to be cached on disk.
- We handle this situation by simply placing objects based upon the "effective" access pattern at the tertiary level.
- Experiment with various cache sizes.

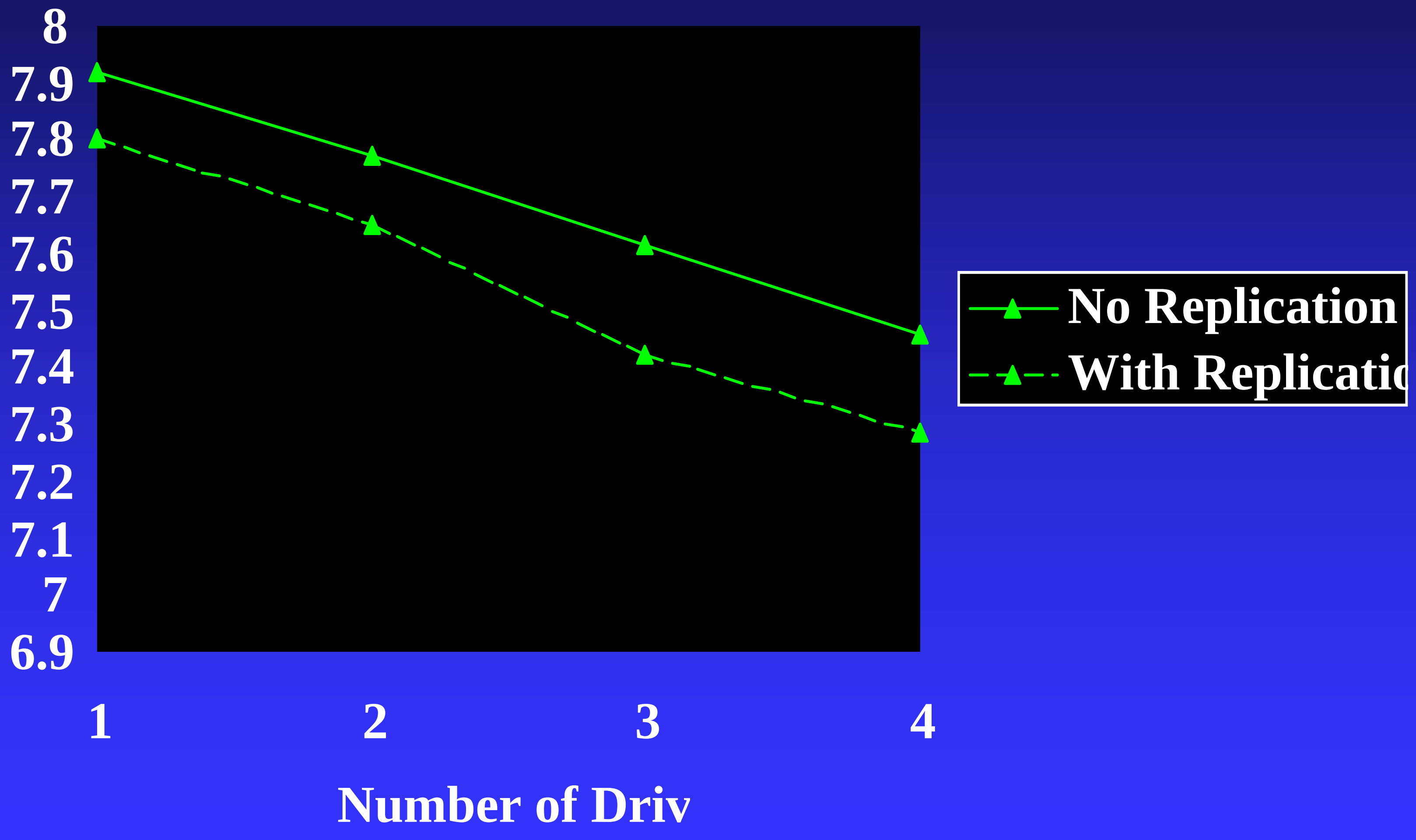
Secondary Storage



Replication

- Objects that belong to more than one cluster cause problems.
- We propose to make replicas of objects (one for each cluster that the object belongs to).
- Since large clusters of related objects are placed placed together, it is quite likely that extra space is left over.
- Storage overhead is also likely to be small.

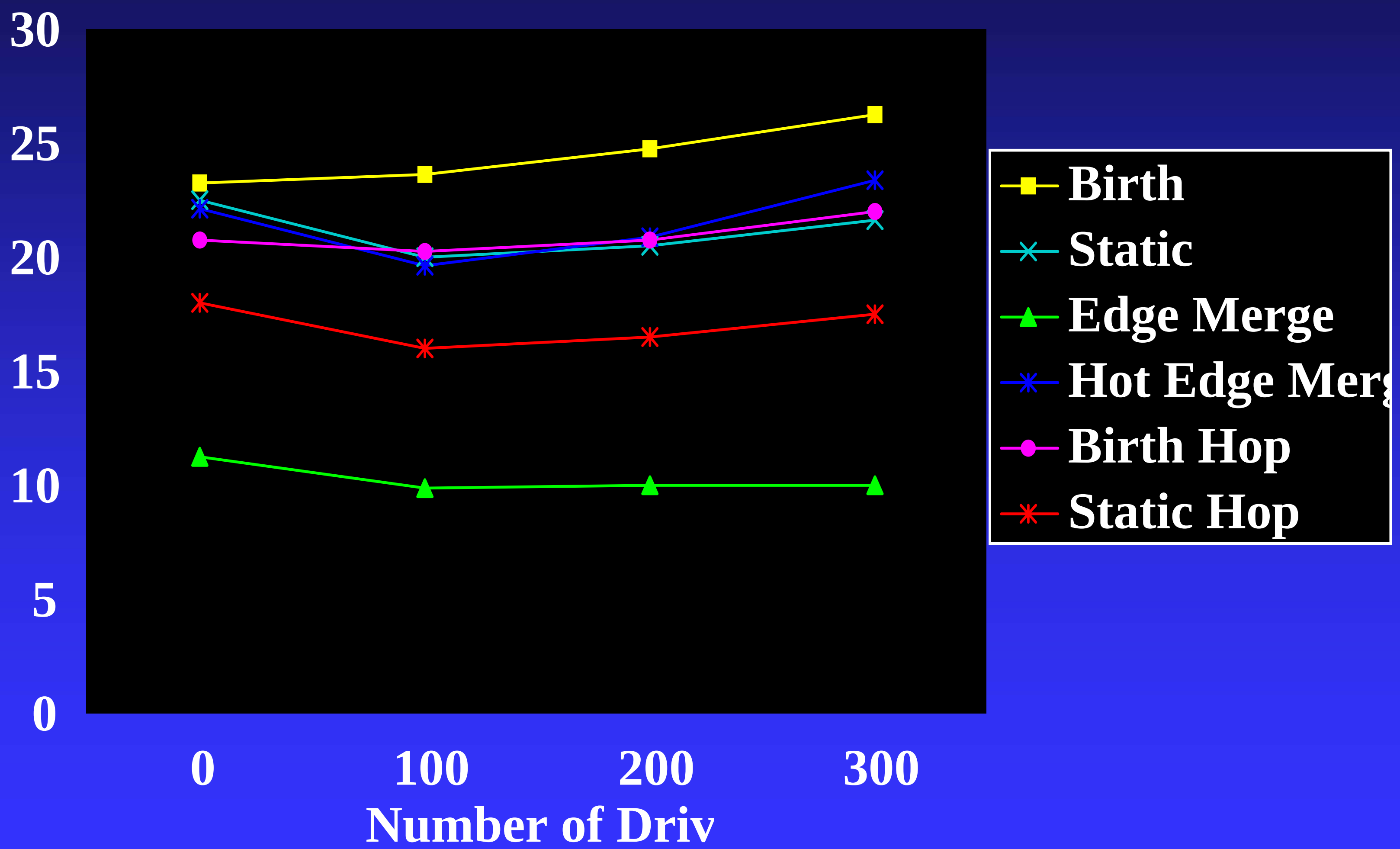
Replication



Prefetching

- Since related objects are clustered on the same medium -- prefetching is very promising.
- Trade-off.
- Experiment 1:
  - Always prefetch data not on disk
  - Vary max prefetch size per medium

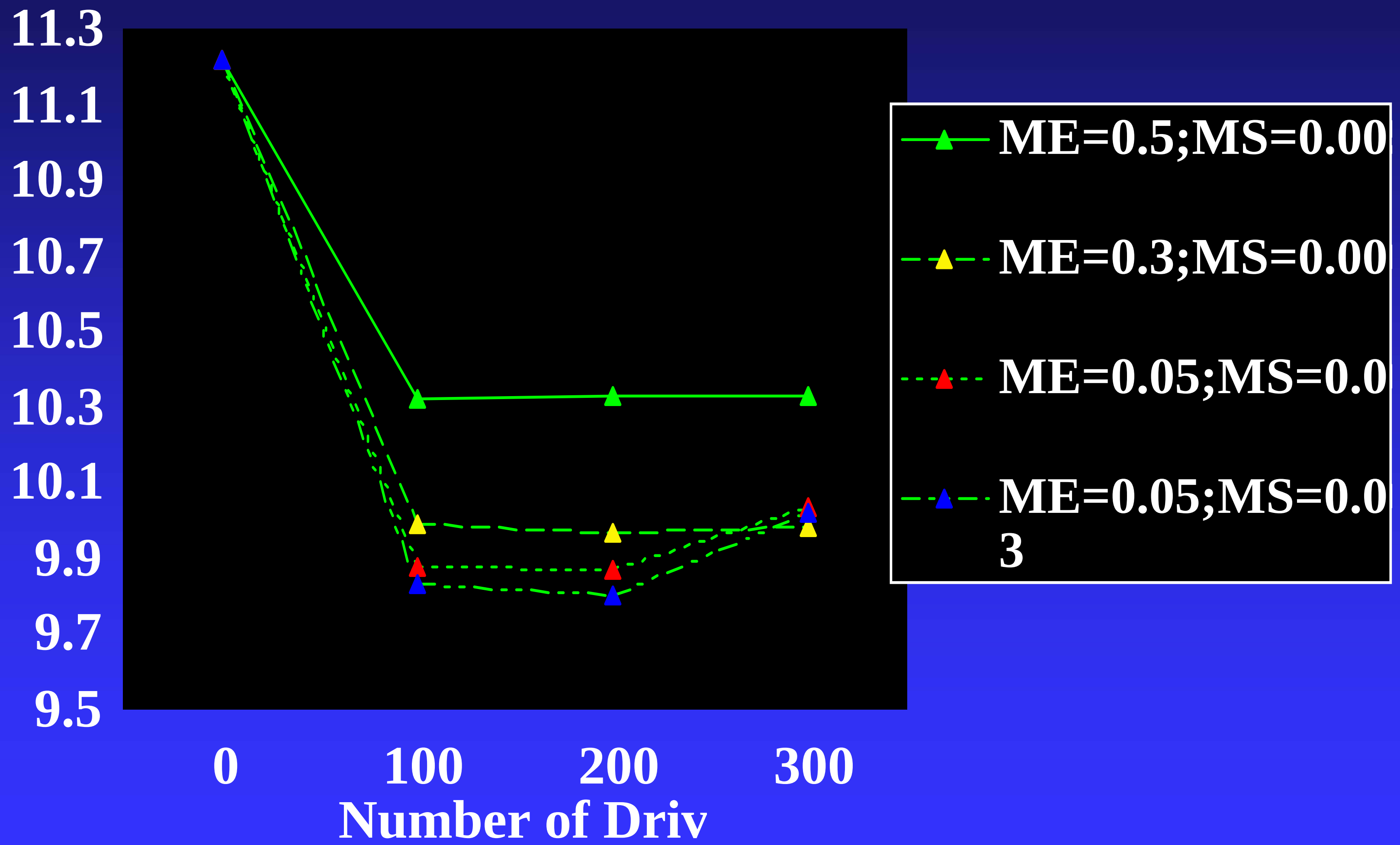
Prefetching



Prefetching (contd.)

- Edge merge is best suited for prefetching.
- Experiment 2:
  - Prefetch only if suitable object exists
  - Object has strong edge from current object, or
  - Object has high static
  - Set bounds for each:
    - ME - minimum edge probabilty
    - MS - minimum static probability

Prefetching



Conclusion

- If objects are accessed in a related fashion -- this information is valuable for placement.
- Proposed schemes (esp. Edge Merge) significantly outperform "optimal" schemes based upon independent access assumptions (77% better)
- Exact knowledge of access pattern is not critical -- only the relationship information is important.
- Adapting to changes in access patterns and handling unknown access patterns is easily achieved.

Conclusion (contd).

- Incorporating disk cache effects is handled in the same manner as adapting to changes in access patterns.
- Selective replication and prefetching are effective for the proposed schemes, resulting in significantly improve performance.