Towards Mass Storage Systems with Object Granularity

Koen Holtman

CERN/CMS

Peter van der Stok

Eindhoven University of technology

Ian Willers

CERN/CMS

NASA and IEEE Mass Storage Conference

March 27-30, 2000

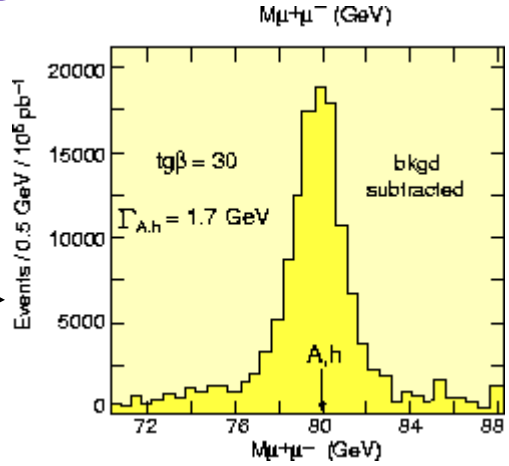
Introduction

- Many applications analyse sets of KB -- MB size objects
- Tapes work most efficiently with MB -- GB size files
  - This leads to mass storage systems with **file granularity**
- Investigated potential benefits of mass storage systems with **object granularity**
  - The application stores and accesses objects
  - The MSS maps the objects to files
  - The MSS can **re-map** objects to new files, to re-optimise if the object access patterns change
- Investigation into benefits done by developing an architecture for an MSS with object granularity
  - Architecture incorporates solutions to associated scalability and fragmentation problems
  - Architecture based on simulation and implementation studies

Application

- Object granularity data analysis
  - Append-only dataset
- Multi-TB set of objects, object size KB -- MB
- Query:

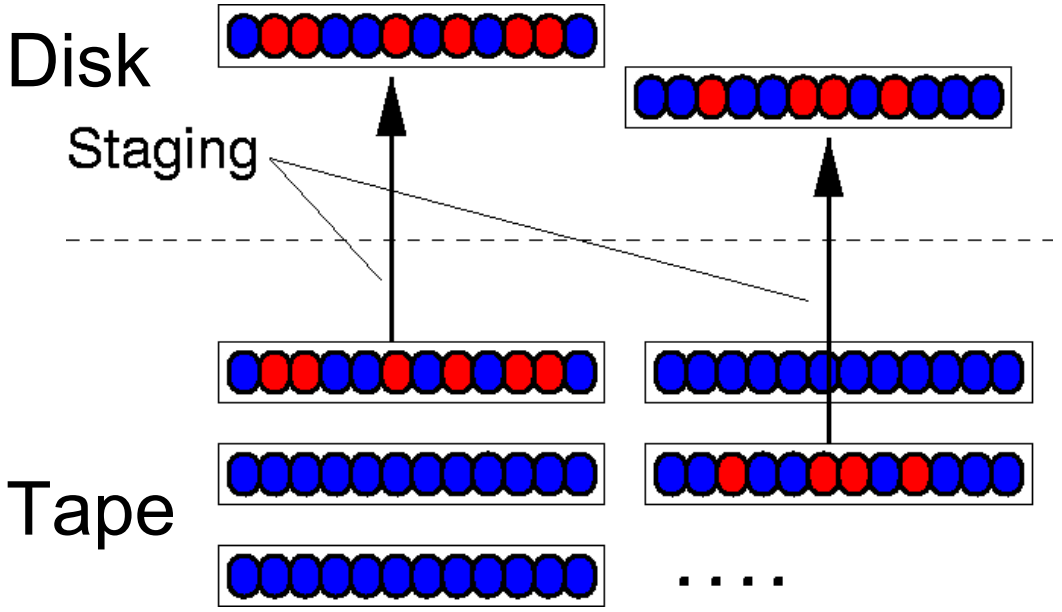
  for_each(o in S) { h:=h + f(o) }



- Computation is order-independent, trivial to create sub-queries and run in parallel
- In High Energy Physics, S can be 10^4 objects out of 10^9 objects

File-granularity MSS

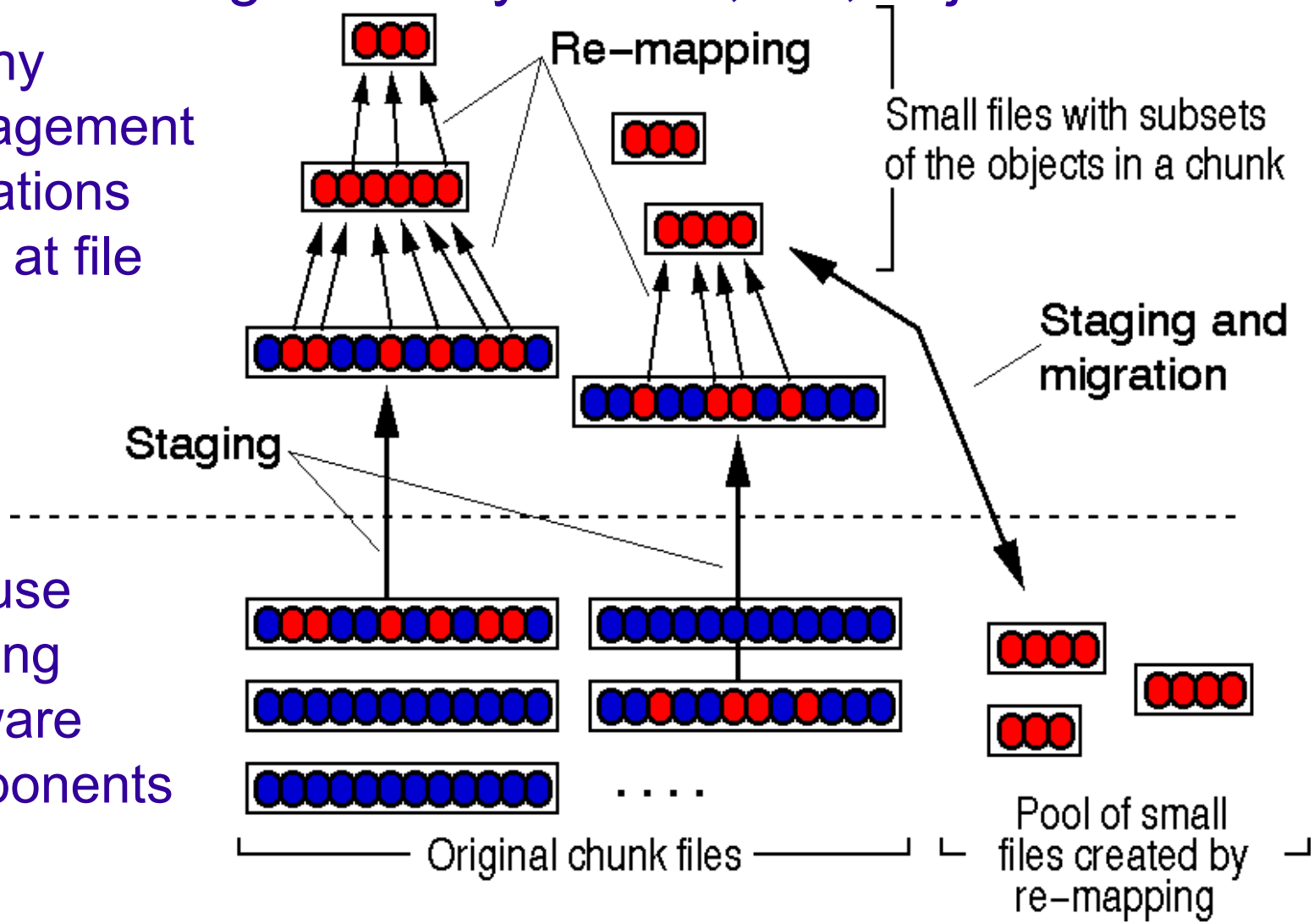
- Make a fixed mapping of objects to files
- Staging is in units of files
  - Red objects are hit by a query



- If there are few red objects, this is inefficient in staging and caching

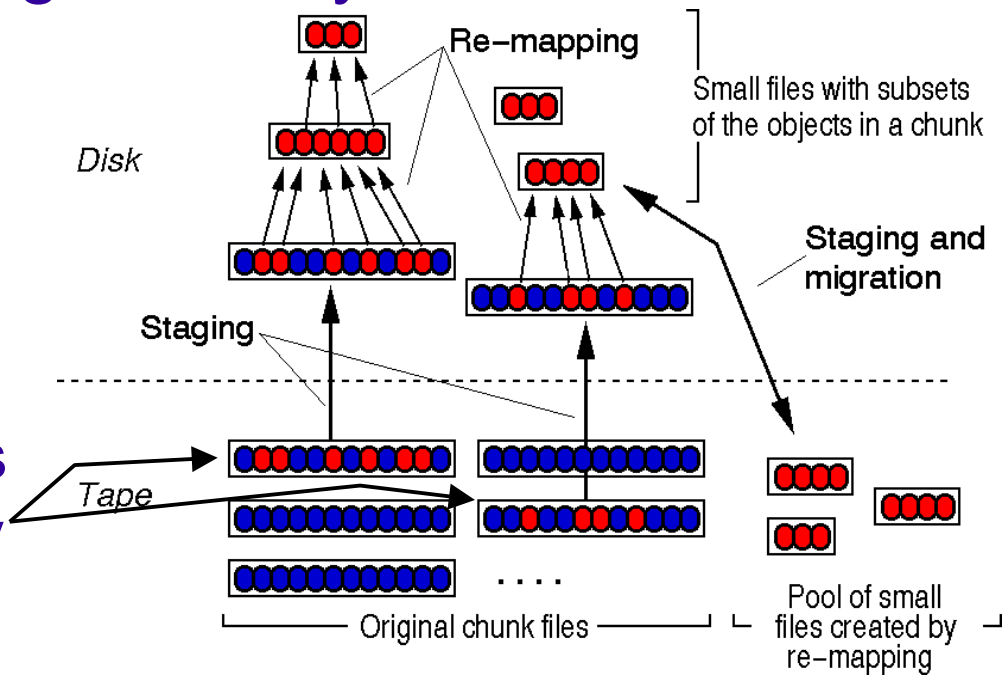
Our Object gran. MSS

- 3 levels of granularity: chunk, file, object
  - Many management operations done at file level
  - Re-use existing software components



When is it useful?

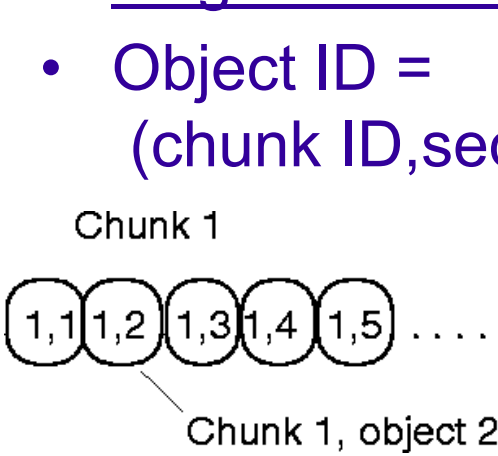
- Conditions under which our object granularity MSS outperforms file granularity MSS:
- Repetitive access condition
- Sparse access condition
  - Amount of red objects in chunks hit by query should be <= 30%

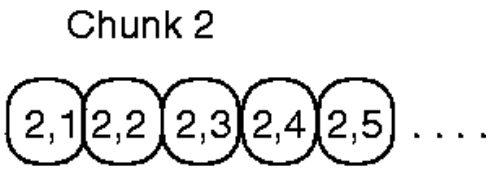


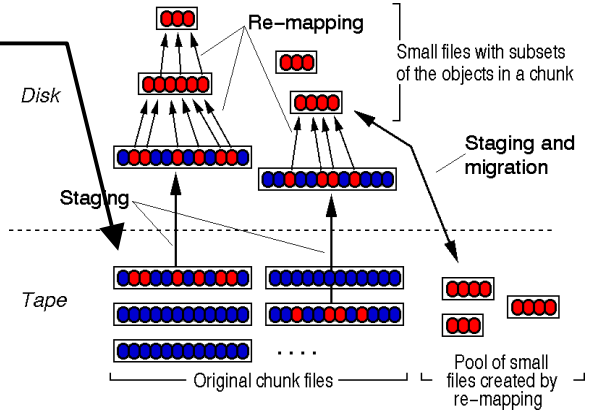
- Usefulness of keeping small files on tape:
  - Less useful than hoped in our system/workload scenarios
  - Useful if disk cache small (<=4%) or if many queries huge

Filling and indexing

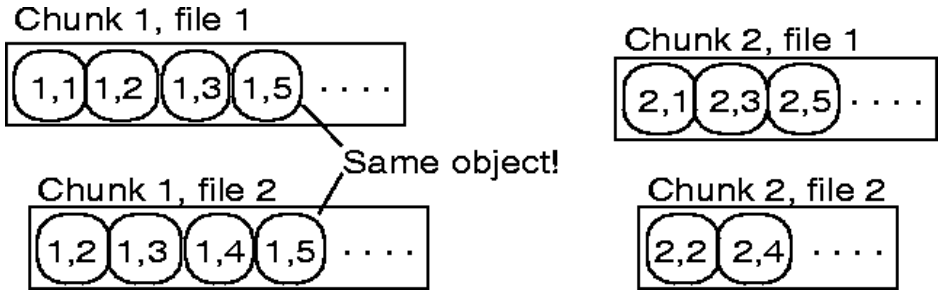
- Append-only: application programmer fills system with new **chunks** (sets of objects)
- Original chunk file created for each new chunk
- Object ID = (chunk ID,sequence number in chunk)





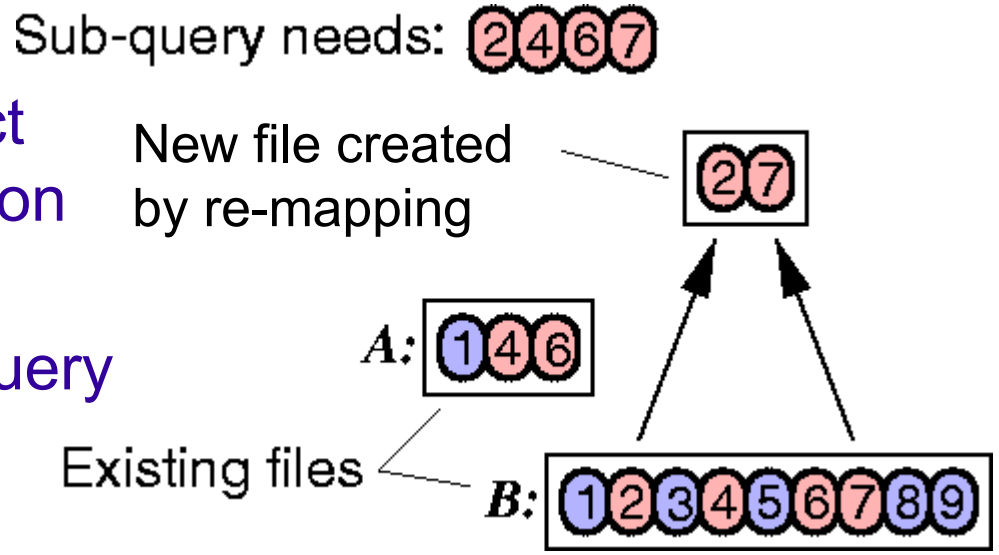


- Up to ~20 files per chunk (in our simulations)
- Every file stores a subset of the objects in a chunk, in sequence order



Re-mapping operation

- Based on object copy, not move
- Done at the file level

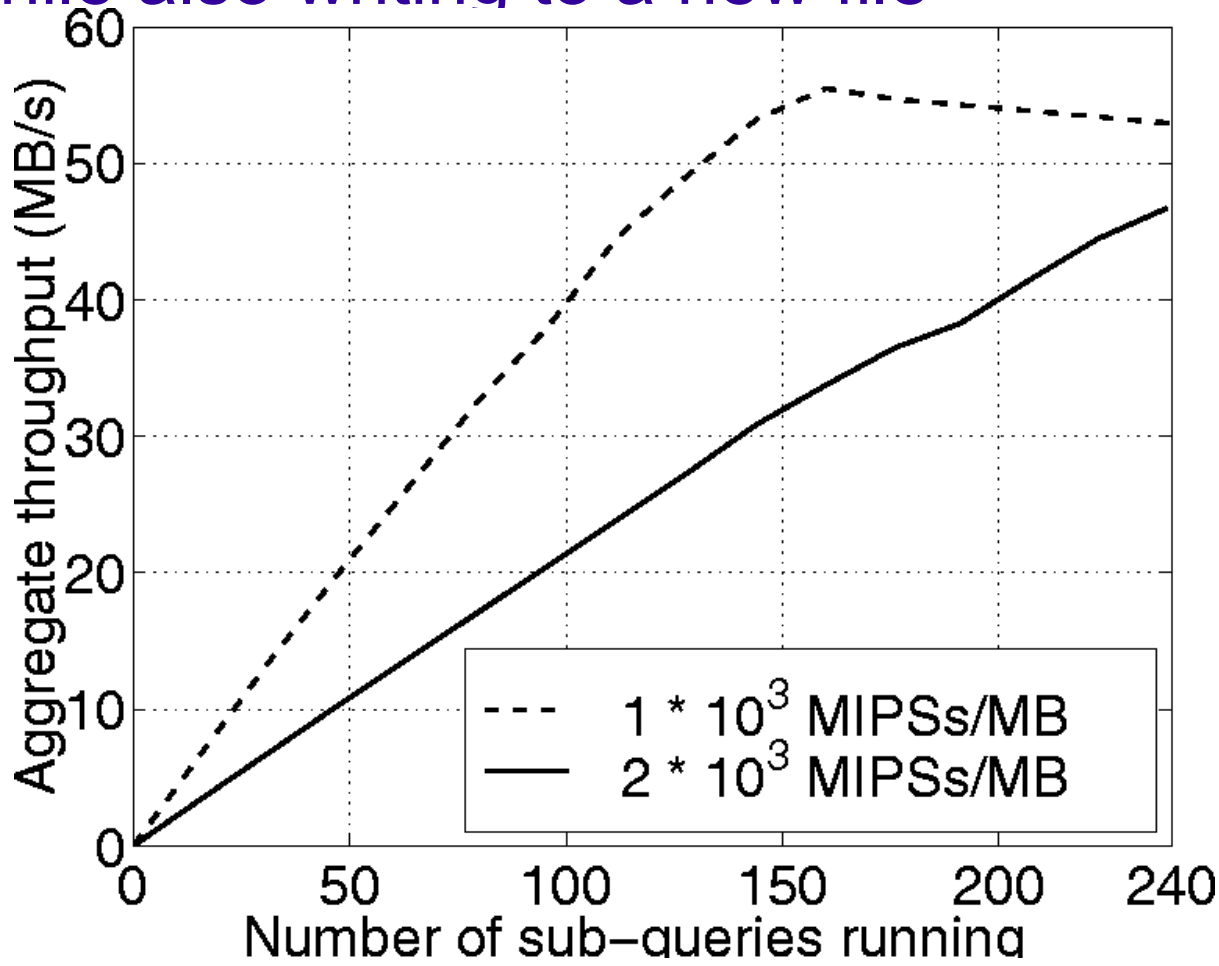


- Done as a side-effect of sub-query execution
- Triggered by sparse reading of the sub-query
- Tuned to <33%

- Preserves order of objects, to avoid fragmentation
- Objects already in a small file are not copied, to avoid unnecessary duplication
- Files are deleted by cache replacement algorithm

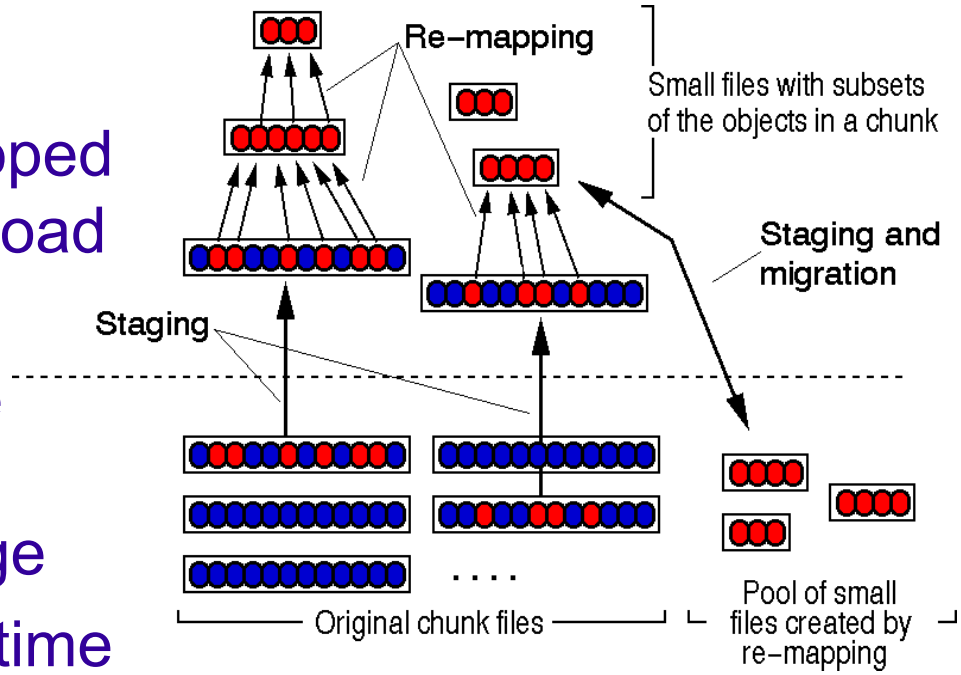
I/O scalability

- One sub-query can read concurrently from many files on disk, while also writing to a new file
  - Use 'bursty sequential reading' optimisation
  - Test: each sub-query reads objects concurrently from 3 files, re-maps 10% of the objects to a new file



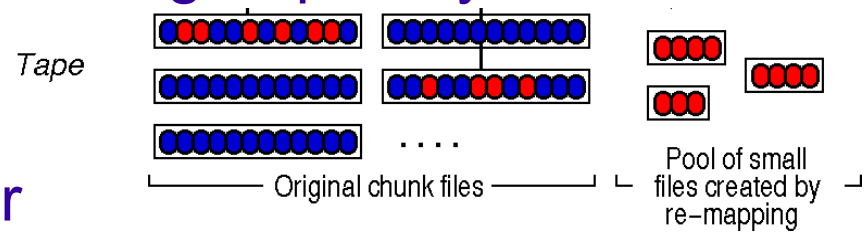
Small files on tape

- Idea is to re-stage small files instead of original chunk files
  - Less useful than hoped in our system/workload scenarios
  - Useful if disk cache small (<=4%) or if many queries huge
  - In practice, a lot of time is spent in writing small files to tape, almost as much time as is saved in staging
  - Use of tape pool...
  - Selection of small files to migrate… size <=20%
    - Maybe a better selection heuristic can be found



Scheduling and staging

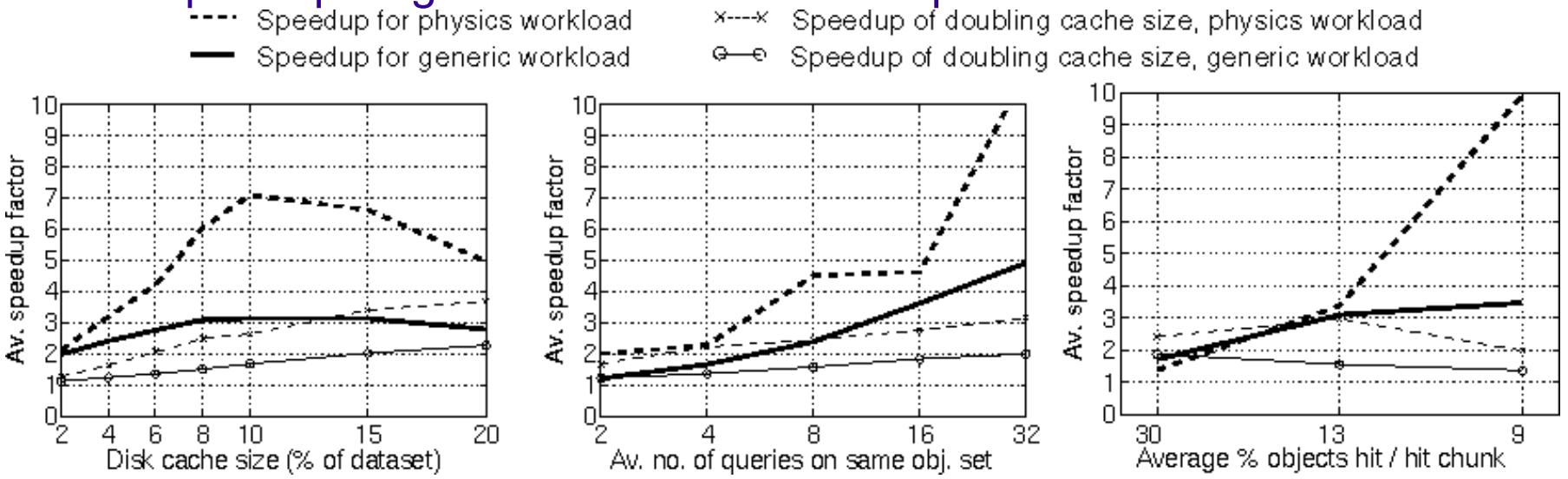
- Multi-user workload, users submit queries
- Query converted into sub-queries as soon as possible, to maximise concurrency and make demands visible
- Sub-query
  - (1) Blocks on condition 'all needed objects are in files on disk'
  - (2) Blocks on condition 'CPU and disk I/O becomes available'
  - (3) Executes
- Sub-queries blocked on tape are grouped by chunk into *clusters*, stager identifies a file which would un-block all sub-queries in the cluster



- Stager cycles over all tapes in a fixed order, all files on a tape that un-block a cluster are staged
  - Stager makes sure that all affected sub-queries get a chance to execute before staged files are deleted in cache replacement

Performance gains

- Using simulation we compared our object-granularity MSS to a normal file-granularity MSS over a large parameter range
- Multi-user workloads with queries requesting 0.03% to 6% of the object set, average object hit rates in hit chunks of 30% to 9% (sparse access condition)
- Speedups of 1 (none) to 52 found
- Speedups higher if workloads more repetitive



- Going to object-level granularity usually outperforms doubling the cache size

Conclusions

- Investigated potential benefits of mass storage systems with object granularity
- Work inspired by
  - impedance mismatch between our application needs and tape characteristics
  - Scenarios like ‘need 5 objects from every tape’
- Developed and validated an architecture for an object granularity MSS
  - Has solutions to fragmentation, scalability problems
  - 3 levels of granularity: chunk, file, object
  - Object re-mapping operation
- Identified conditions under which object granularity is beneficial

Discussion

- **Better than expected:** stability of system, speedups over wide range of workload parameters
- **Worse than expected:** keeping small files on tape not very beneficial in our scenarios
  - Maybe a better selection heuristic can be found
- **Future work:**
  - More implementation studies
  - Object granularity and re-mapping in wide-area distributed physics analysis systems
  - (maybe) Look at possible benefits of user hints
- **Related work:**
  - Data management by hand
  - StorHouse Atomic Data Store
  - View materialisation in relational databases and in data mining