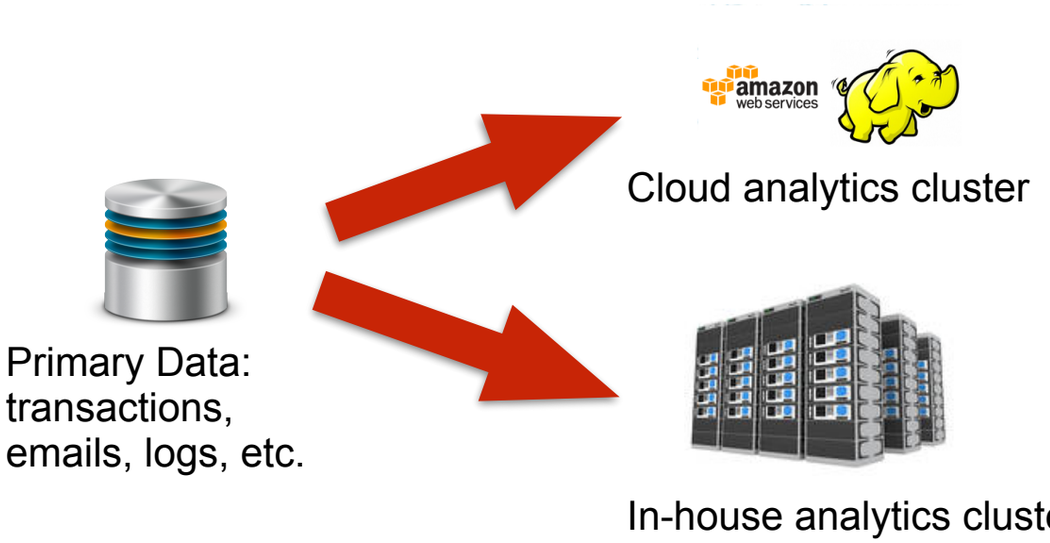
Data Sharing Made Easier through Programmable Metadata

Zhe Zhang
IBM Research

Remzi Arpaci-Dusseau
University of Wisconsin-Madison

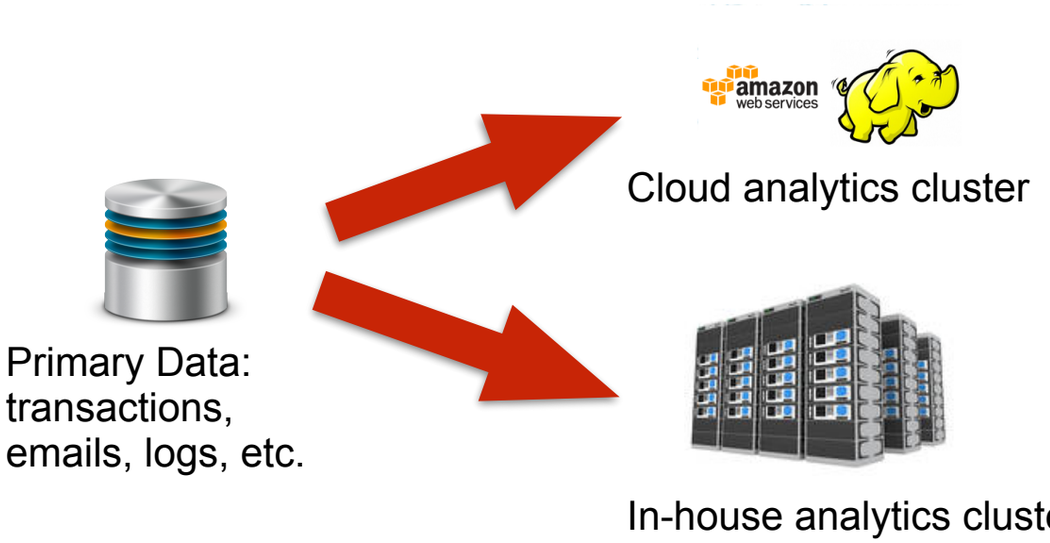
How do applications share data today?

–Syncing data between storage systems:
- Commonly used big data workflow
- Slow, stale and strenuous

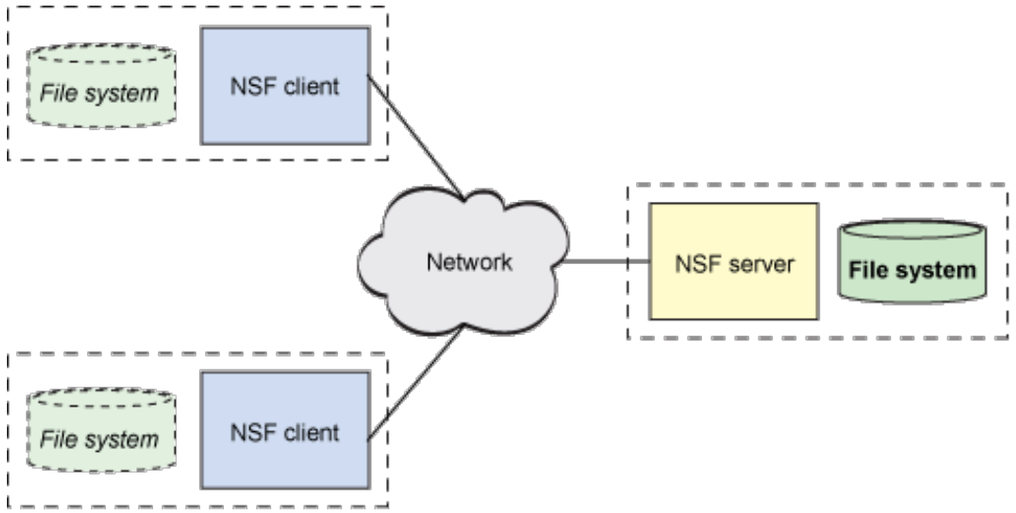


How do applications share data today?

– Syncing data between storage systems:
  - Commonly used big data workflow
  - Slow, stale and strenuous

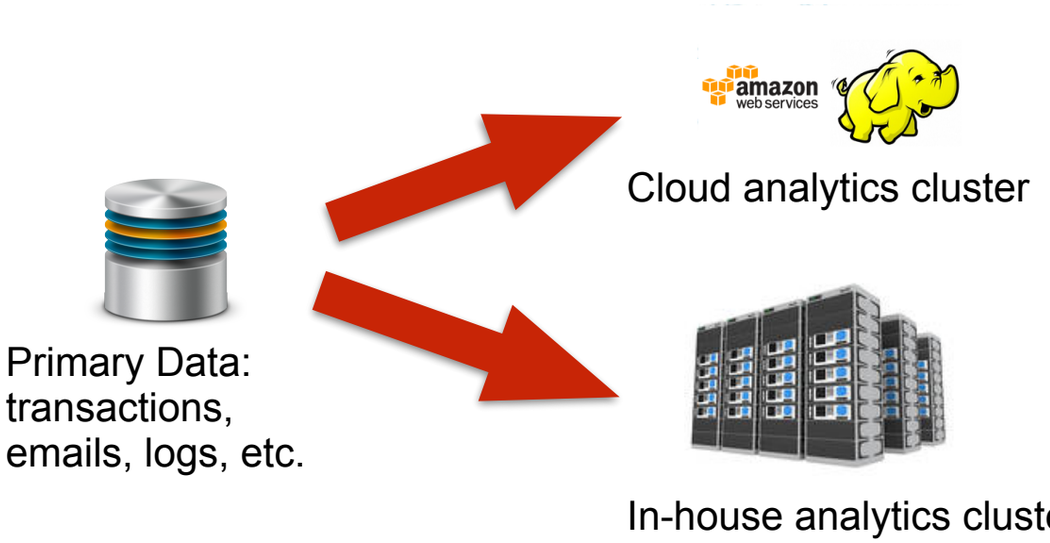


– Mounting and using shared storage systems:
  - Difficult to serve heterogenous workloads
  - Heavy workload on centralized name nodes

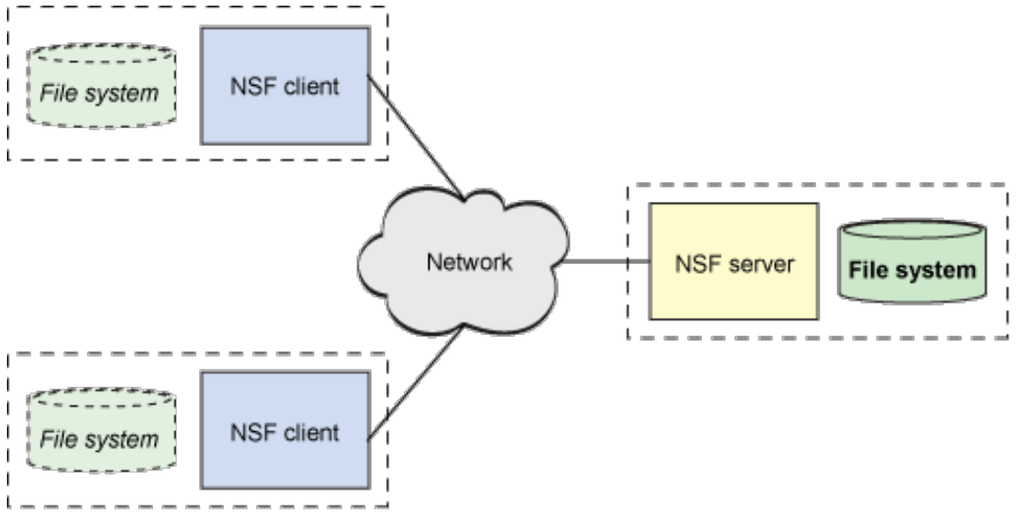


How do applications share data today?

–Syncing data between storage systems:
- Commonly used big data workflow
- Slow, stale and strenuous



–Mounting and using shared storage systems:
- Difficult to serve heterogenous workloads
- Heavy workload on centralized name nodes



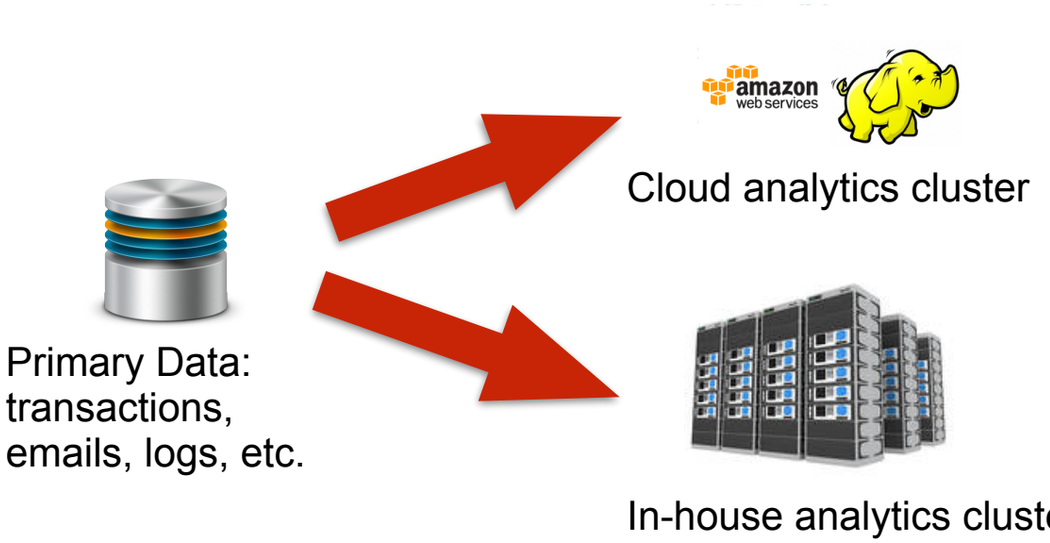
Observations

–Data *always* written and read through the same storage system (filesystem, DB, etc.)
- Metadata updated with writes
- Metadata used in reads

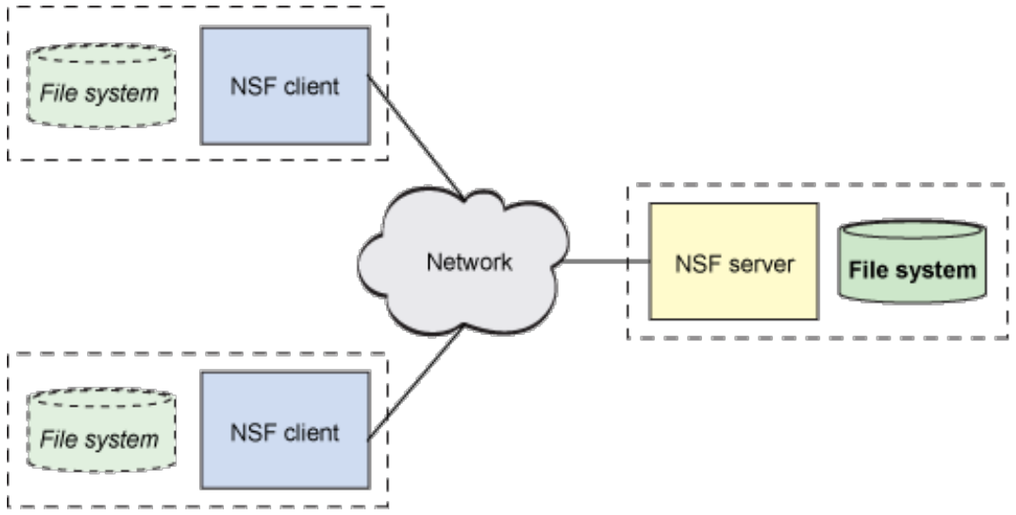
–Data produced in form *A* and consumed in form *B*?
- *View* DB records as a file?
- Analyze thousands of local log files as a single text file?

How do applications share data today?

–Syncing data between storage systems:
- Commonly used big data workflow
- Slow, stale and strenuous



–Mounting and using shared storage systems:
- Difficult to serve heterogenous workloads
- Heavy workload on centralized name nodes

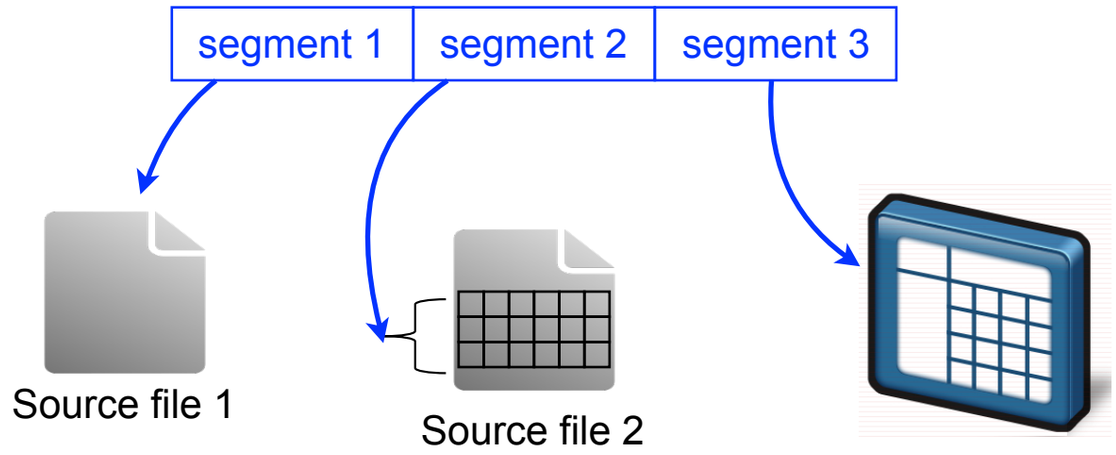


Observations

–Data *always* written and read through the same storage system (filesystem, DB, etc.)
- Metadata updated with writes
- Metadata used in reads

–Data produced in form *A* and consumed in form *B*?
- *View* DB records as a file?
- Analyze thousands of local log files as a single text file?

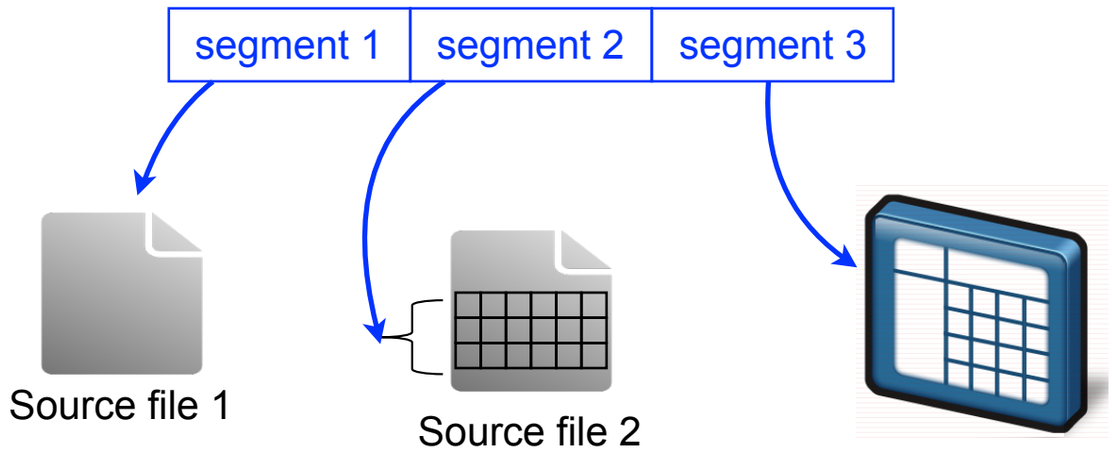
Programming the Metadata



Logical definition

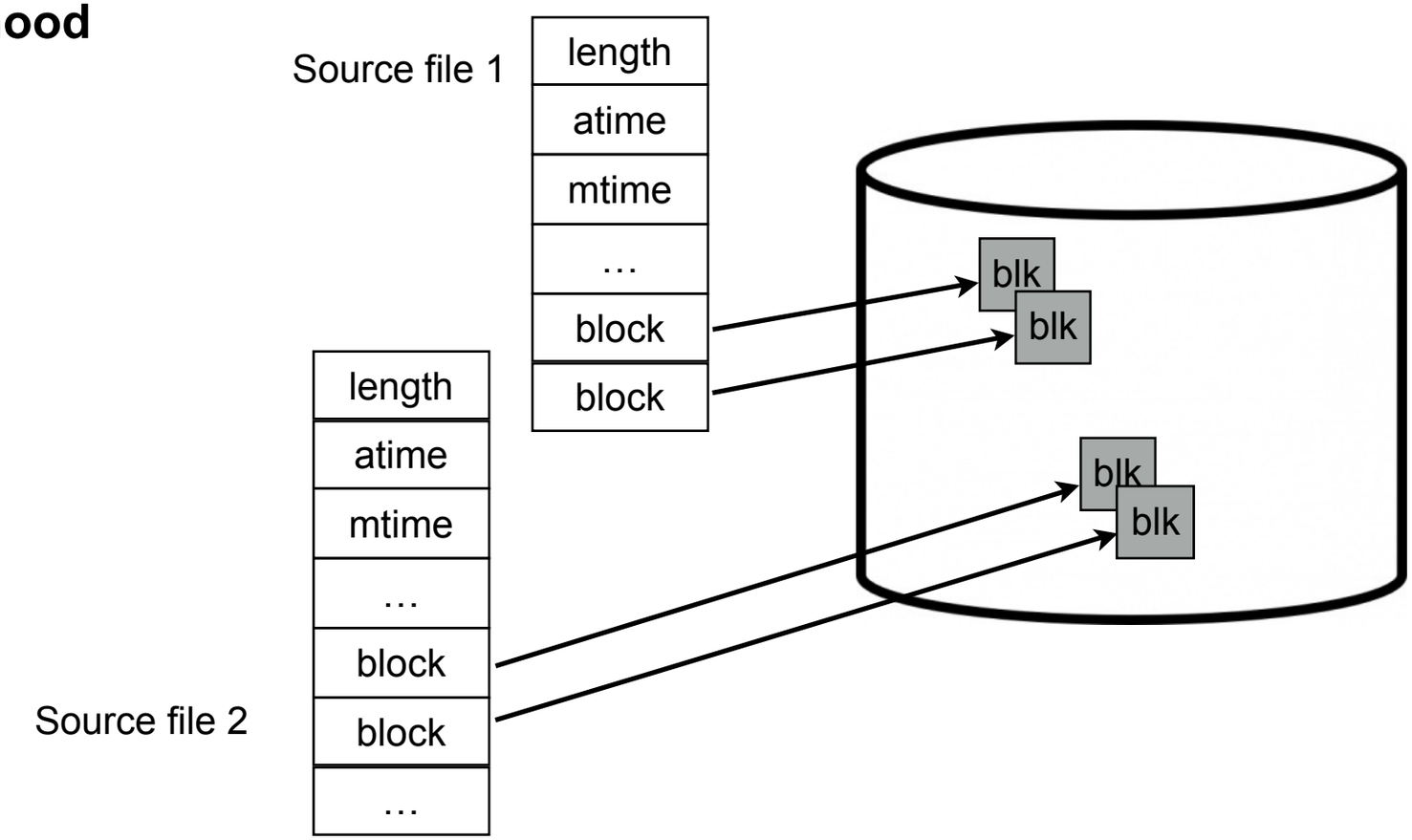
Under the hood

Programming the Metadata

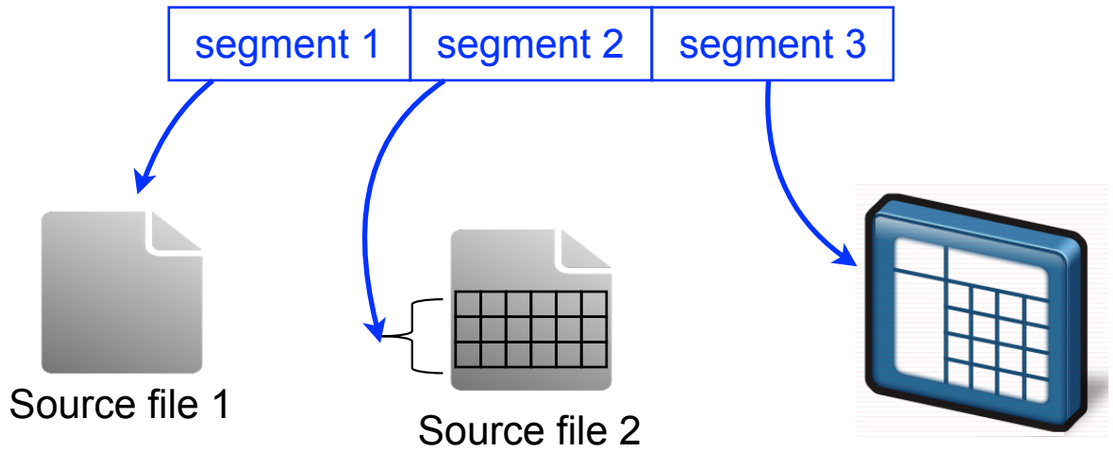


Logical definition

Under the hood

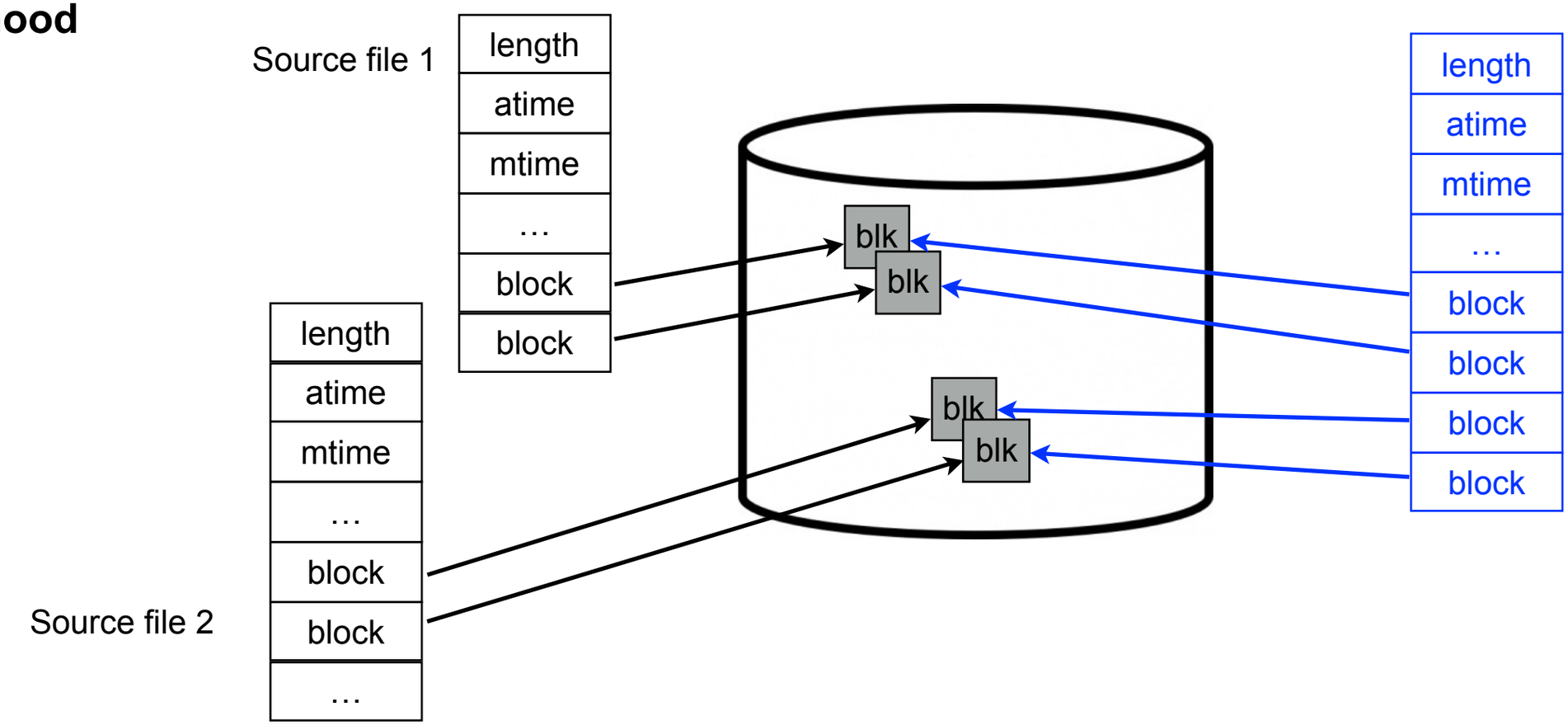


Programming the Metadata

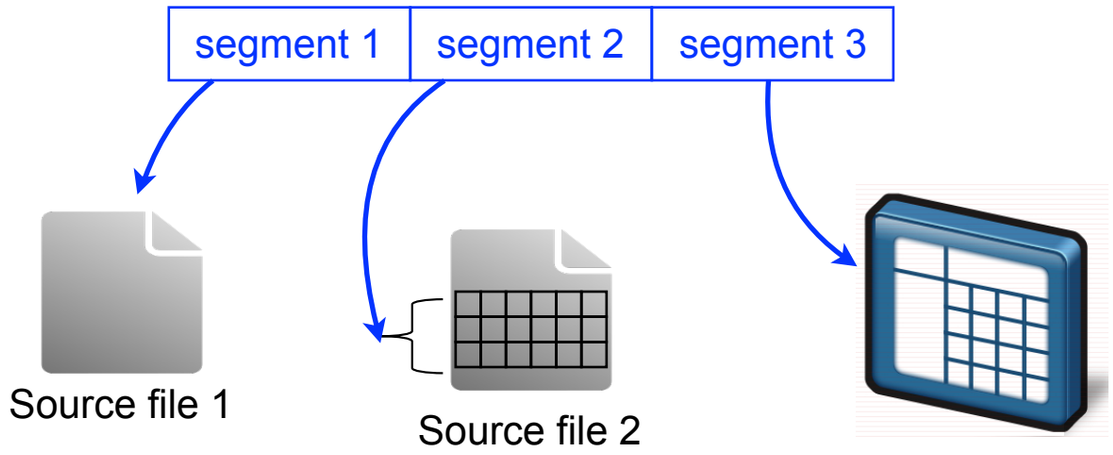


Logical definition

Under the hood

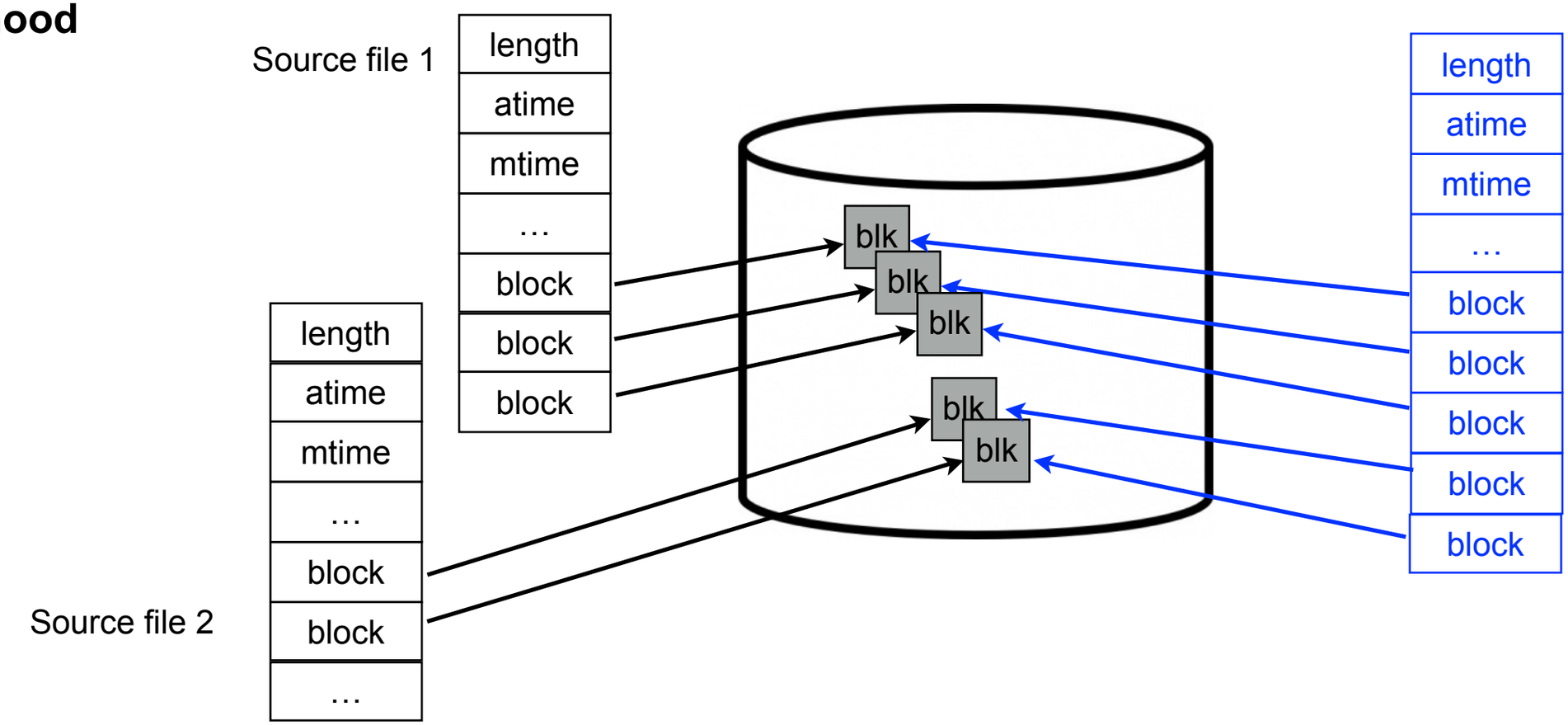


Programming the Metadata



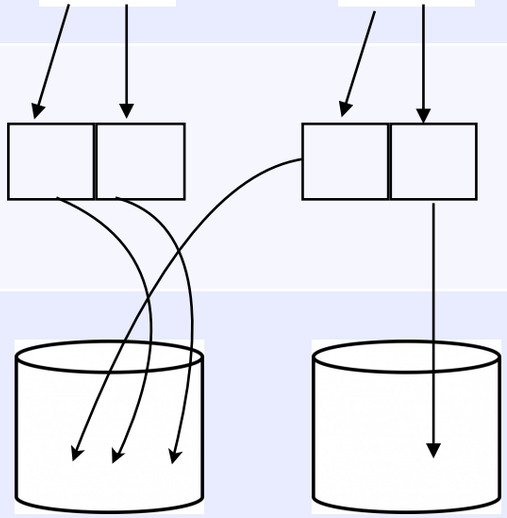
Logical definition

Under the hood



Challenges

- **API challenge: identification / namespace of source data**
  - How to define a file in VM1 to include a source file in VM2?
  - Granularity-based source file selection: 1 out of 10 lines of text?
  - Content-based source file selection: all lines containing certain keyword?
  - Arbitrary “SELECT * FROM * WHERE *” in source DB tables?
- **Performance challenge: frequent metadata updates**

Layers		Example of Liseners
Applications		• Map to destination file if keyword matches • Map every 1 line out of 10 lines of text to destination file
VFS		• Map entire file to destination file • Map every 1MB out of 10MB to destination file
Block storage		• All VFS listeners can be implemented on block layer with a reverse pointer from block to inode

Use Case: Distributed Live Analytics

- **hadoop dfs -composeFromLocal <configuration file> <path to HDFS file>**
- **Configuration file**
  slave1:/opt/IBM/*/*.log
  slave2:/var/*.log
  …
- **Challenges**
  - Informing NameNode of local file size changes
  - Balancing workload