Poster at EuroSys '08, Glasgow, UK

Okeanos - Reliable Archival Storage for Heterogeneous Stream Data

Andromachi Hatzieleftheriou and Stergios V. Anastasiadis
Department of Computer Science, University of Ioannina, GREECE

Introduction

- A general-purpose stream archival facility could serve as a building block for a variety of applications, e.g.
  - network packet monitoring
  - urban traffic control
- General monitoring case:
  - messages received from massive numbers of sensors
  - reception at potentially different rates
  - data should be stably stored on disk
- Existing systems' inadequacies:
  - Traditional systems (such as relational DBs):
    - not engineered to efficiently store continuous stream data automatically generated from sensors in real time
  - Modern stream storage servers:
    - basically designed to store stream files of limited size for repetitive playback
    - inefficient to constantly accumulate continuous stream data for archival purposes
- Monitoring sensors may generate:
  - high-resolution video and audio streams at large rates
  - intermittent variations in environmental conditions at much lower rates
- Summarizing received data should:
  - be stably stored on the storage facility
  - not compromise the sequential playback performance

Journaling Filesystems

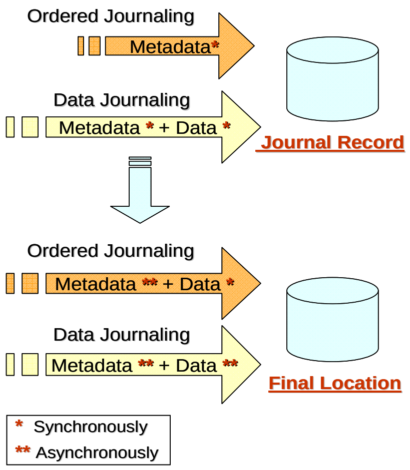
- Preserve filesystem consistency across system crashes at minimal recovery time
  - improvement of operation reliability
- Two alternative journaling modes can be used:
  - metadata-only logging (ordered journaling)
  - write-ahead logging of file contents (data journaling)
- Data journaling negatively affects disk throughput in sequential write workloads
  - doubly stores data at both journal record and the final location in the file system structures
- According to aggregate workload characteristics:
  - Ordered journaling → efficient for sequential access
  - Data journaling → efficient for random access

Stream Rate Characteristics

- Efficient and reliable storage of multiple concurrent streams
  - aggregate workload → random-access behavior
  - appends corresponding to individual streams → perfectly sequential
- Data received from low-rate streams:
  - significant overhead for the immediate movement from memory to the final disk location
  - disk penalized with small writes
  - flushing data to the final location can be deferred to a later more convenient time
- Data received from higher-rate streams:
  - larger amount of data
  - can be moved directly to their final destination on disk without compromising the efficiency of the storage device

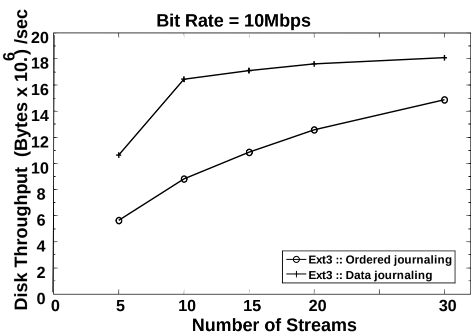
Journaling Inefficiencies

- Operation sequence in ordered journaling:
  - At each journal commit interval, before the journal record is updated (metadata only), data is flushed to the final location
  - At each pdflush wake-up interval, both data and metadata are flushed to their final locations
  - within journal record:
    - metadata written synchronously
    - sequential writes → efficiency
    - small amount of data → efficiency
  - within final location:
    - metadata written asynchronously
    - data written synchronously
    - continuous disk traffic → inefficient for small writes
- Operation sequence in data journaling:
  - At each journal commit interval, the journal record updated (both data and metadata)
  - At each pdflush wake-up interval, both data and metadata are flushed to their final locations
  - within journal record
    - both data and metadata written synchronously
    - sequential writes → efficiency
    - large amount of data → inefficient for large volumes of data
  - within final location
    - both data and metadata written asynchronously
    - deferred writes (write-coalescing) → efficient for small writes



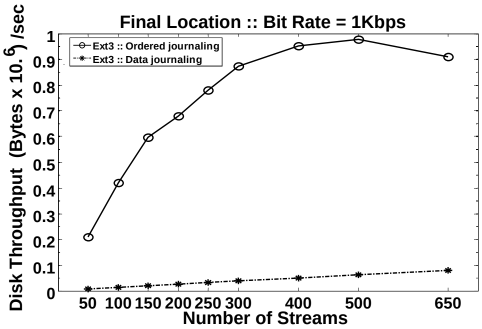
High-Rate Stream Results

- Total disk traffic:
  - random-access behavior
  - ordered journaling better than data journaling

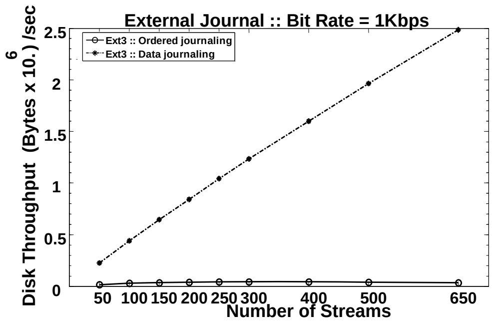


Low-Rate Stream Results

- Final location traffic:
  - Ordered journaling
    - random access
    - more data than needed
  - Data journaling
    - only needed data logged



- Journal traffic:
  - Ordered journaling
    - only metadata logged
  - Data journaling
    - sequential writes
    - more data than needed



In Progress

- For each individual stream the system should automatically:
  - identify the most appropriate journaling approach
  - adjust its behavior according to the varying features of the stream over time
- Data journaling inefficient for low-rate streams
  - Possible solution: differential logging

References

[1] P. J. Desnoyers and P. Shenoy, *Hyperion: High Volume Stream Archival for Retrospective Querying*, USENIX Annual Technical Conference, June 2007

[2] V. Prabhakaran et al., *Analysis and Evolution of Journaling File Systems.* USENIX Annual Technical Conference, April 2005, pp. 105-120.

Contact Information

- Please contact {axatzhel, stergios}@cs.uoi.gr
- More information on this and related projects can be obtained at www.srg.cs.uoi.gr

This program is supported by Interreg IIIA Greece-Albania Neighboring Grant No 303090/YD7631/03-07-2007
Intersafe: Teaching and research in computer science with applications to public safety

This project is cofunded by ERDF and national funds through the GREECE-ALBANIA neighboring program

Okeanos - Reliable archival storage for heterogeneous stream data[1]

Andromachi Hatzieleftheriou*
Stergios V. Anastasiadis
Department of Computer Science
University of Ioannina, GREECE
{axatzhel, stergios}@cs.uoi.gr

The prevalence of continuous monitoring processes for system management purposes and general physical site safety make stream processing applications highly relevant in modern computing infrastructures. Recently proposed stream management engines demonstrate the feasibility of flexibly applying time-series operators on massive numbers of streams in real time as their data arrive to the system. In principle, dropping prices in computer hardware should also make possible the storage of high-resolution or numerous streams for entire months or years.

Prior research has made the case that traditional systems (such as relational databases or general-purpose file systems) are not engineered to efficiently store continuous stream data that are automatically generated from sensors in real time [1]. Similarly, modern stream storage servers are basically designed to store stream files of limited size for repetitive playback rather than constantly accumulating continuous stream data for archival purposes. In the general monitoring case, we are interested in receiving messages from massive numbers of sensors at potentially different rates and reliably storing their data on disk files before acknowledging their reception as successful. Some sensors may generate high-resolution video and audio streams at large rates while others may send intermittent variations in environmental conditions at much lower rates. Across all these heterogeneous cases, we need the received data to be stably stored on the same storage facility without compromising the sequential playback performance required for effective visualization or statistics-gathering processing.

In our vision, a general-purpose stream archival facility could serve as a building block for a variety of applications in the entire range from network packet monitoring to urban traffic control with the appropriate indexing functionality built separately at a higher level when needed. In order to improve their operation reliability, general-purpose file systems apply journaling techniques to preserve metadata consistency across system crashes at minimal recovery time. Some of them additionally use write-ahead logging of file contents (or data journaling) in order to prevent small writes from penalizing disk access performance. Nevertheless, data journaling negatively affects disk throughput in sequential write workloads due to doubly storing data at both their temporary journal record and also their final location in the file system structures (Figure 1). Alternatively, disk writing of data to its final location before updating the corresponding metadata can provide consistency guarantees similar to those of data journaling without the extra overhead associated with the latter. Comparisons across different journaling methods with general-purpose file server traffic have shown that either ordered data writing or data journaling may lead to better performance depending on whether the aggregate workload is sequential or random-access [2]. In our undergoing research, we focus on the efficient and reliable storage of multiple concurrent streams whose aggregate workload demonstrates random-access behavior even though appends corresponding to individual streams may be perfectly sequential.

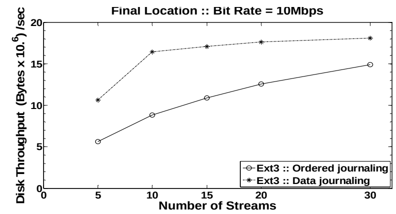


Figure 1. Total disk traffic when writing multiple high-rate streams of 10Mbps each on Ext3/Linux 2.6.18. The disk throughput of data journaling is twice that of ordered journaling and flattens out faster.

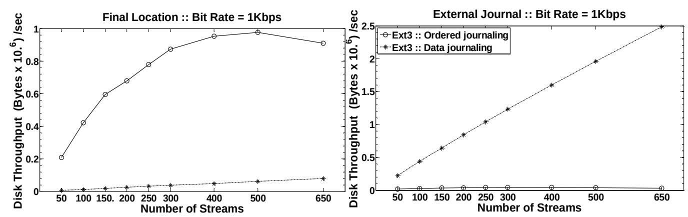


Figure 2. Journal and final location traffic generated from low-rate streams of 1Kbps. Although, in both journaling modes the disk is penalized with small writes, in contrast to the random-access final location writes, log records are written sequentially, which makes differential logging a possible solution to data journaling inefficiency.

Data journaling flushes data to stable log storage and easily restores it after a crash presuming that copying to the final location can be deferred to a later more convenient time. This is useful for low-rate streams that individually would incur significant overhead for their immediate movement from memory to their final disk location (Figure 2). Instead, the larger amounts of data received from higher-rate streams can be moved directly to their final destination on disk without compromising the efficient operation of the storage device. Ideally, the system should automatically identify the most appropriate journaling approach for each individual stream and adjust its behavior according to the varying features of the stream over time.

In conclusion, we motivate the necessity for building systems facilities for the archival storage of heterogeneous streams with different rate and content characteristics. Ideally, such facilities should be able to reliable store massive numbers of streams while requiring minimal recovery time across crashes and supporting low-overhead stream playback for the needs of visualization and statistical processing.

[1] P. J. Desnoyers and P. Shenoy, *Hyperion: High Volume Stream Archival for Retrospective Querying*, USENIX Annual Technical Conference, June 2007.
[2] V. Prabhakaran et al., *Analysis and Evolution of Journaling File Systems.* USENIX Annual Technical Conference, April 2005, pp. 105-120.

[1] Supported in part by an Interreg IIIA Greece-Albania Neighboring Grant No 303090/YD7631/03-07-2007.

* Graduate student