Eidetic Systems

David Devecsery, Michael Chow, Xianzheng Dou,
Jason Flinn, Peter Chen

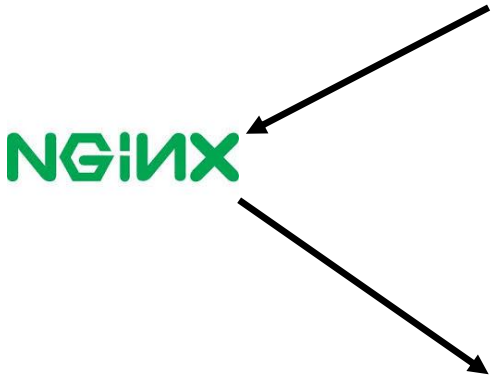
University of Michigan

What is an Eidetic System?

Eidetic – Having “Perfect memory” or “Total Recall”

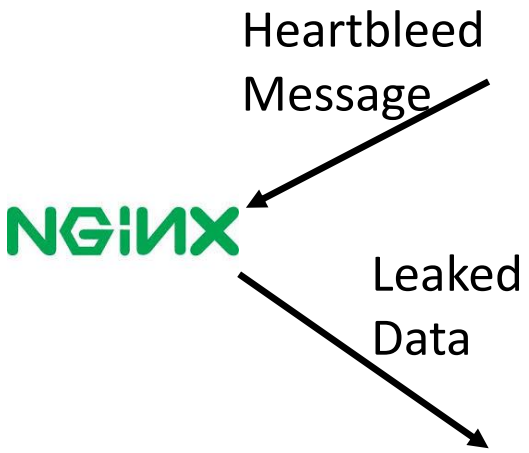
Eidetic System – A system which can recall and trace through the lineage of any past computation

Motivation - Heartbleed



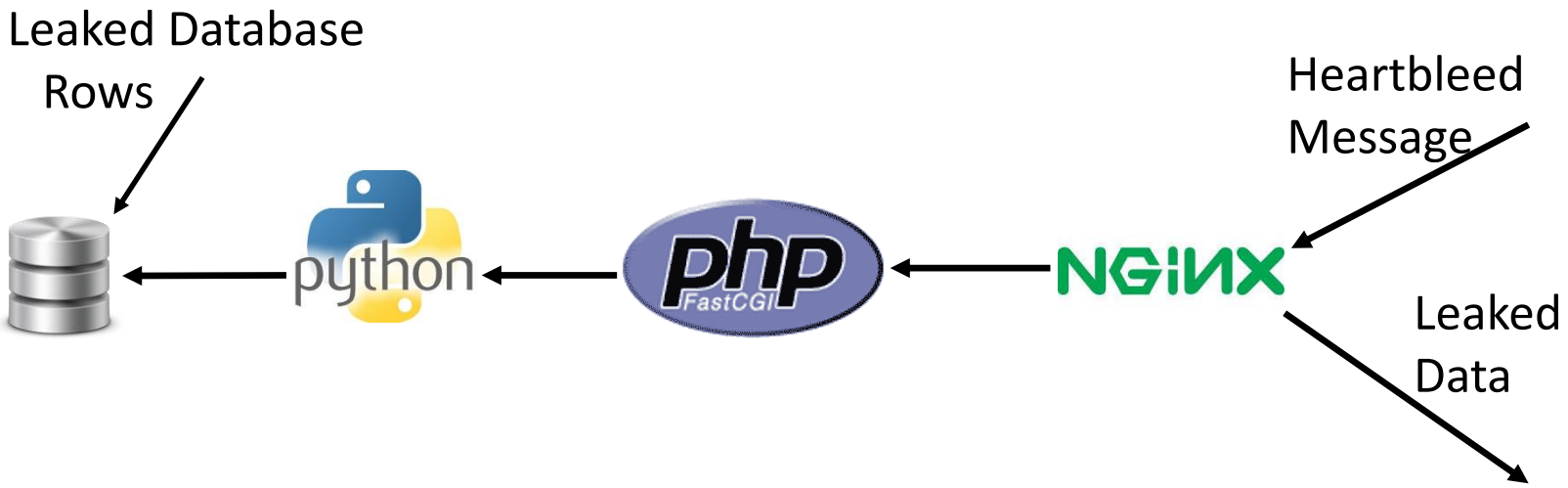
- Was Heartbleed exploited?
- What data was leaked?

Motivation - Heartbleed



- Was Heartbleed exploited? - Yes
- What data was leaked?

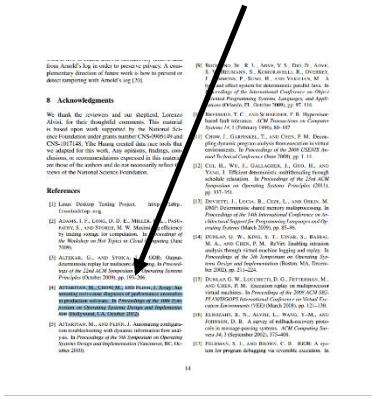
Motivation - Heartbleed



- Was Heartbleed exploited? - Yes
- What data was leaked?

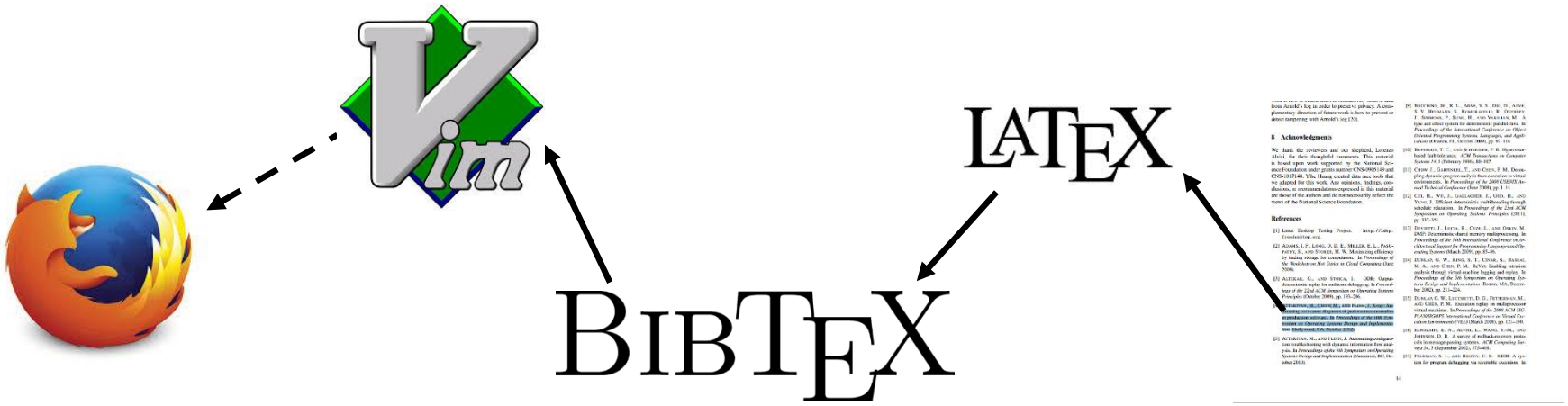
Motivation – Wrong Reference

Bad Citation



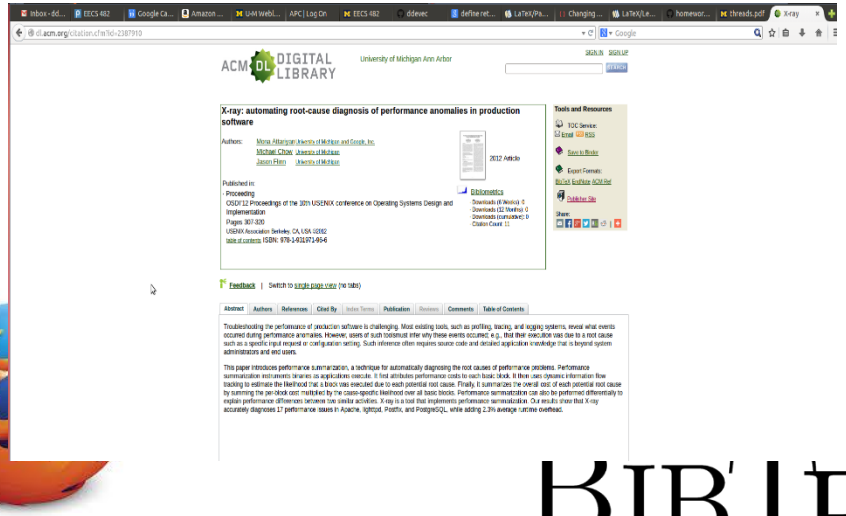
- How did I get the wrong citation?

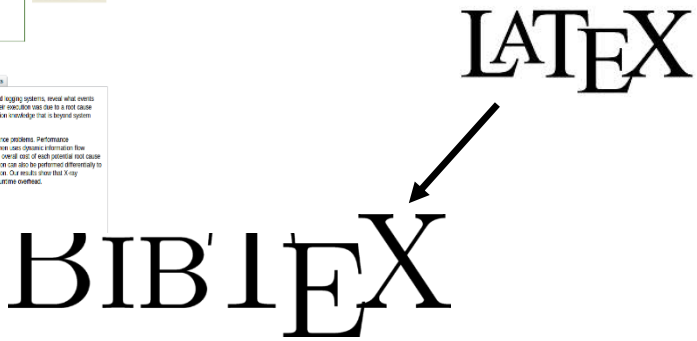
Motivation – Wrong Reference

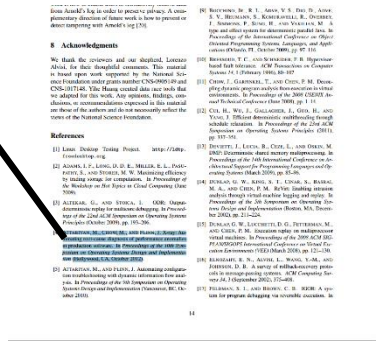


- How did I get the wrong citation?

Motivation – Wrong Reference

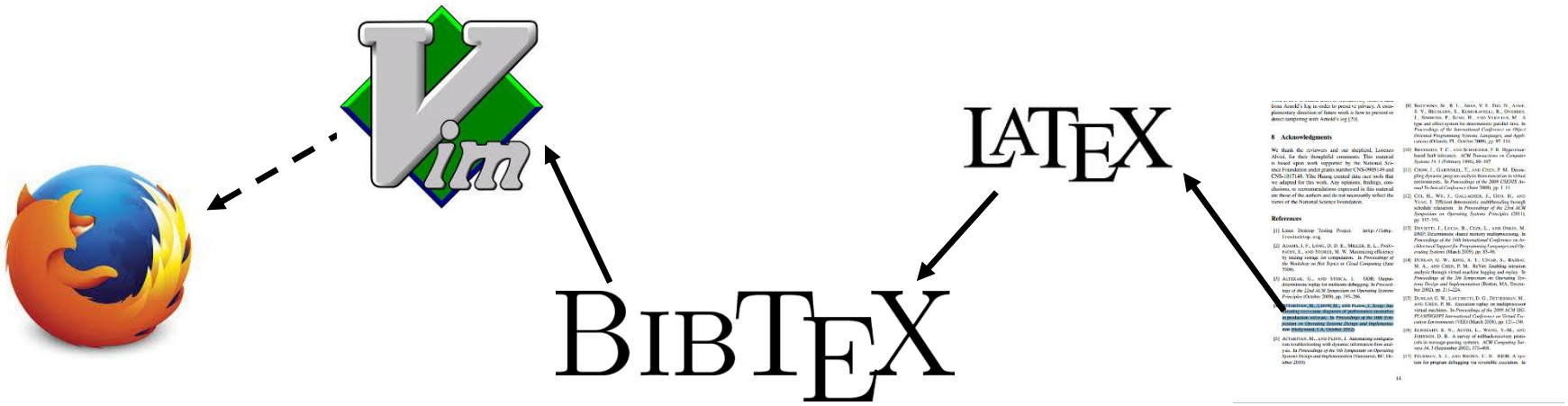






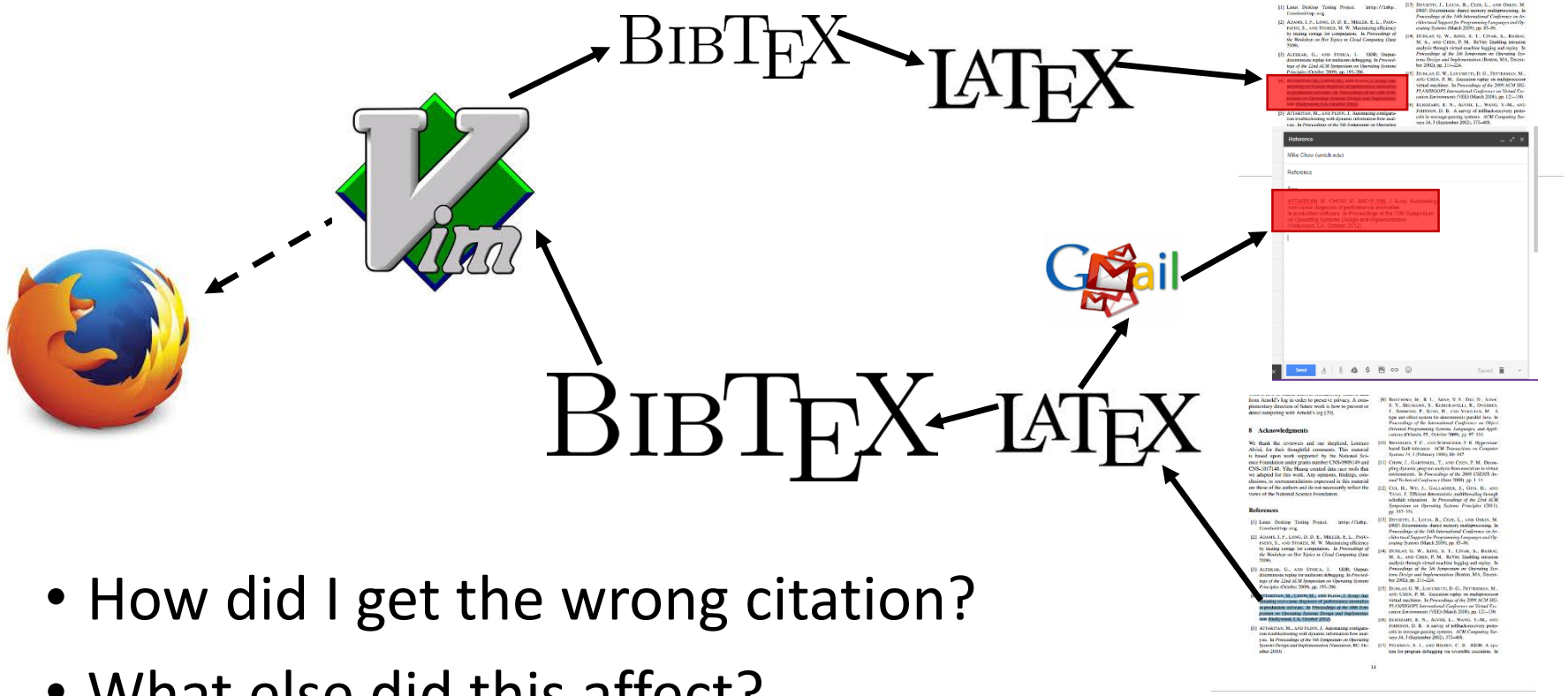
- How did I get the wrong citation?

Motivation – Wrong Reference



- How did I get the wrong citation?
- What else did this affect?

Motivation



- How did I get the wrong citation?
- What else did this affect?

Arnold

- First practical eidetic computer system
  - Efficiently records & recalls all user-space computation
    - Process register/memory state
    - Inter-process communication
  - Handles lineage queries
    - What data was affected?
    - What states and outputs were affected?
- Targeted towards desktop/workstation use
- Reasonable overheads
  - Record 4 years of data on $150 commodity HD
  - Under 8% performance overhead on most benchmarks

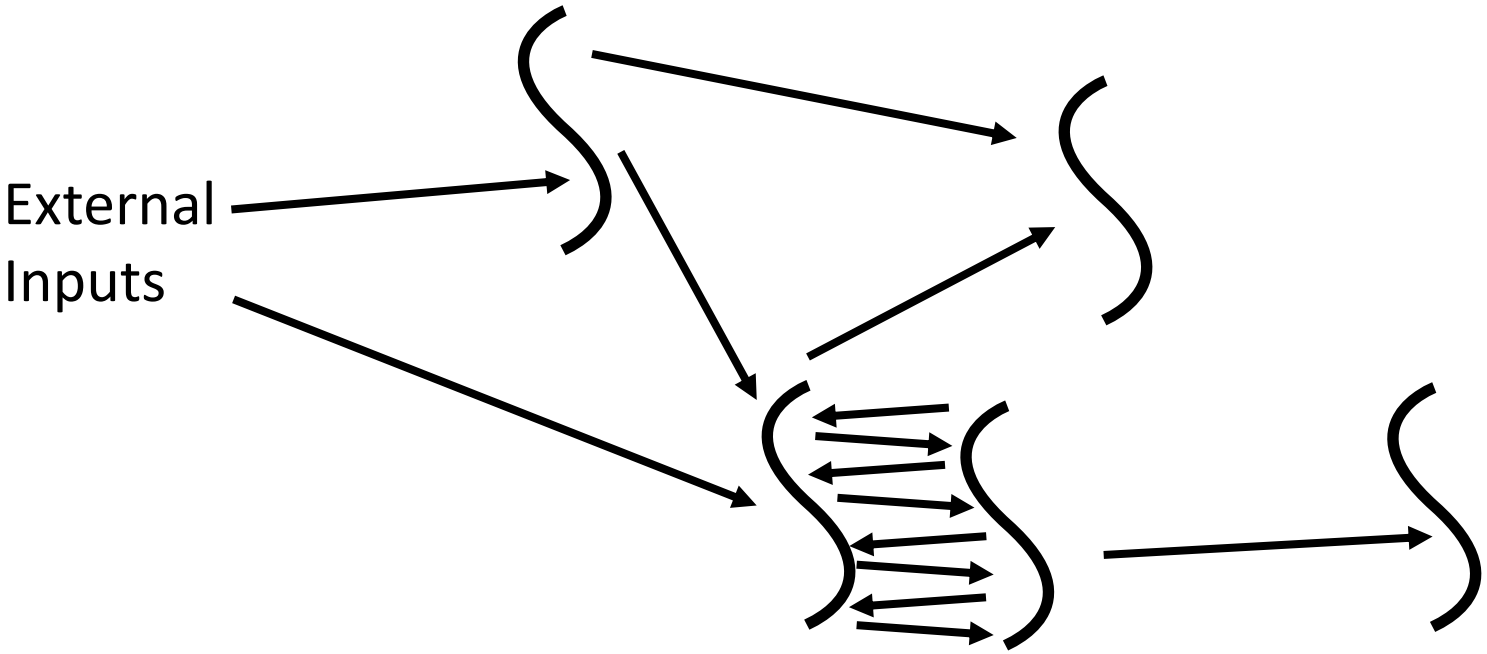
Overview

- Introduction
- Motivation
- How Arnold remembers all state
- How Arnold supports lineage queries
- Conclusion

Remembering State

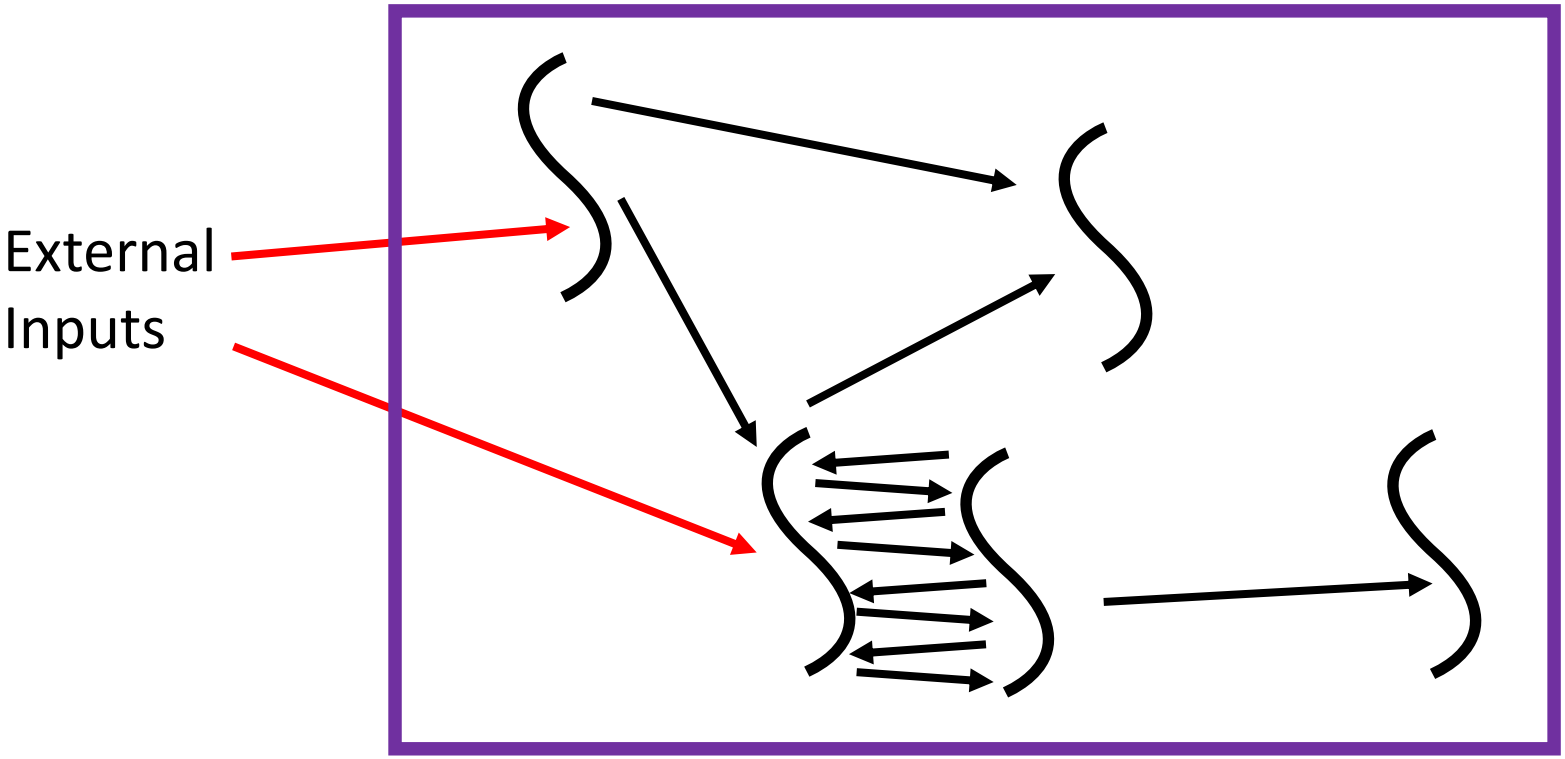
- Requirements:
  - Store years of state on a single disk
    - Memory/register space within a process
    - Inter process communication
    - File state
  - Recall any state in reasonable time
- Solution:
  - Deterministic record & replay
    - “Process group” based replay
    - “Process graph” to track inter-process lineage
  - Log compression

Recording Granularity



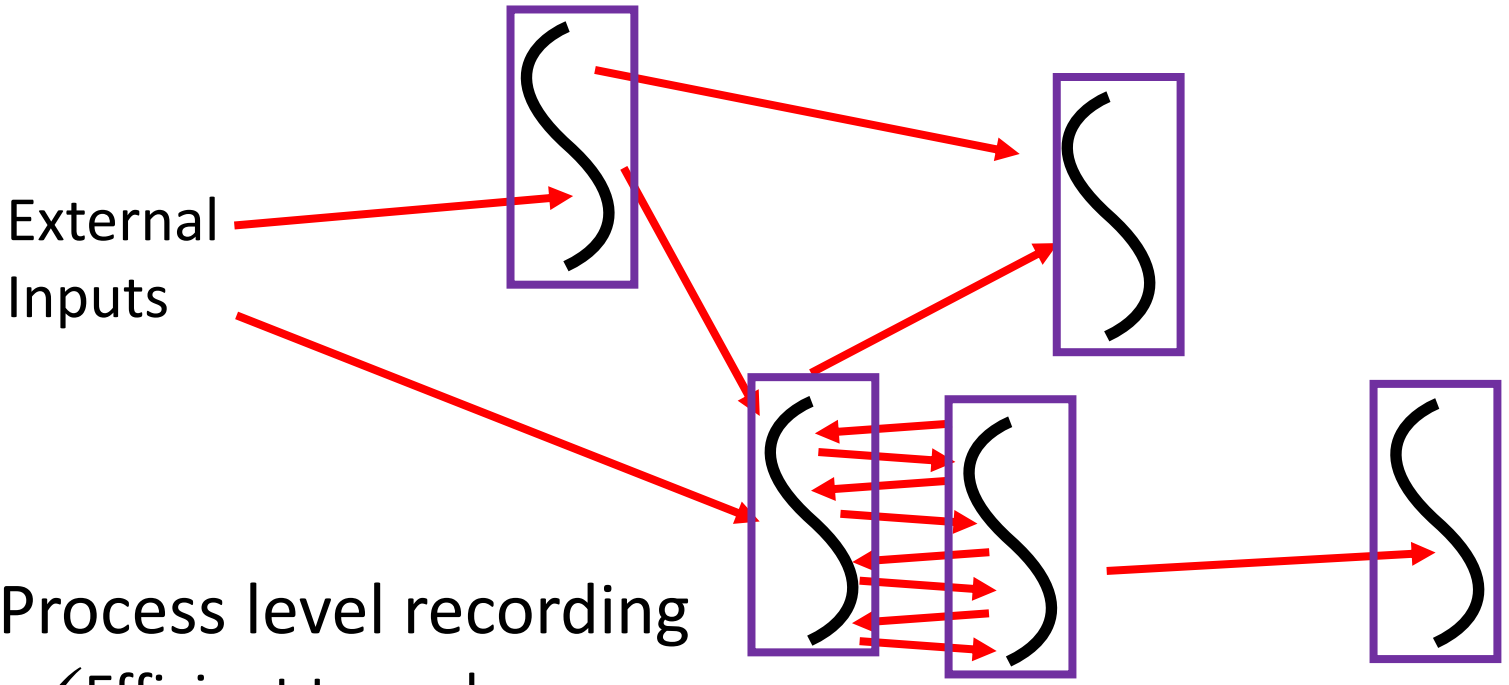
- What granularity is best to record our system?

Recording Granularity



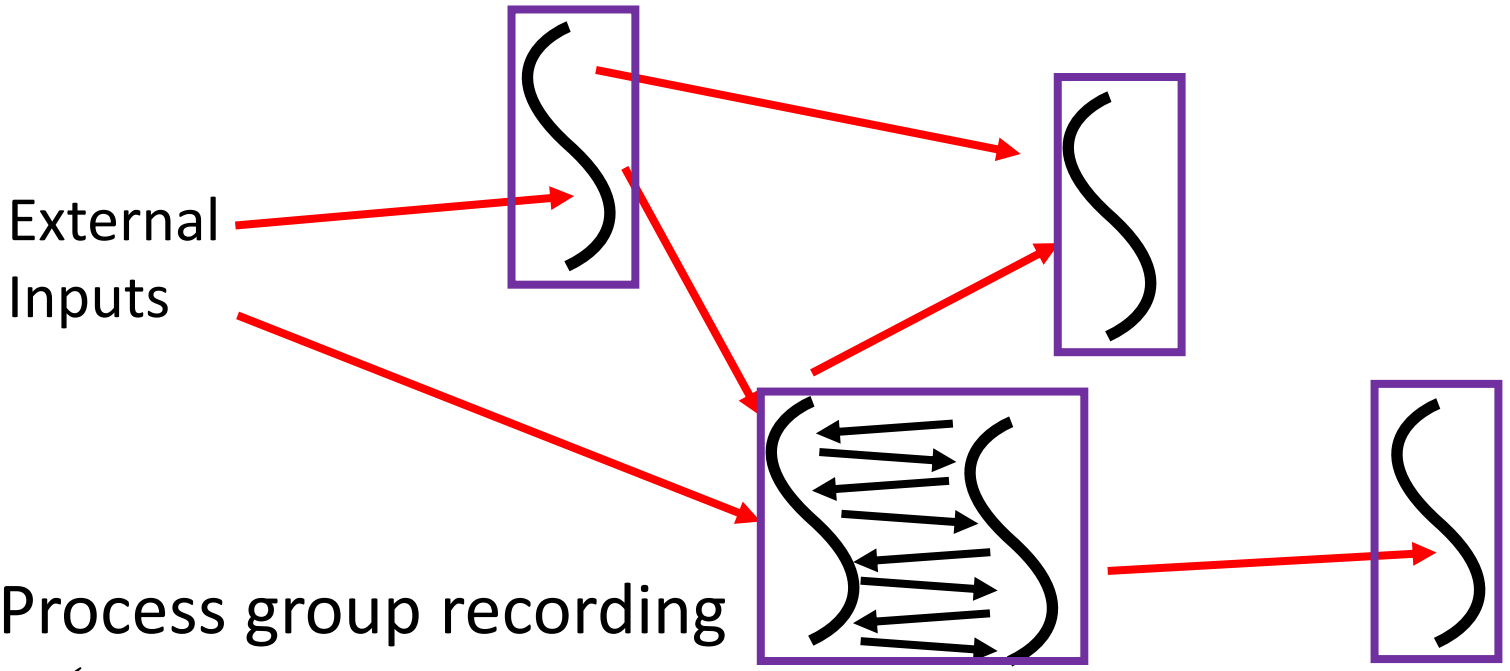
- Whole system recording
  - ✓Low space overhead
  - ×Costly to replay

Recording Granularity



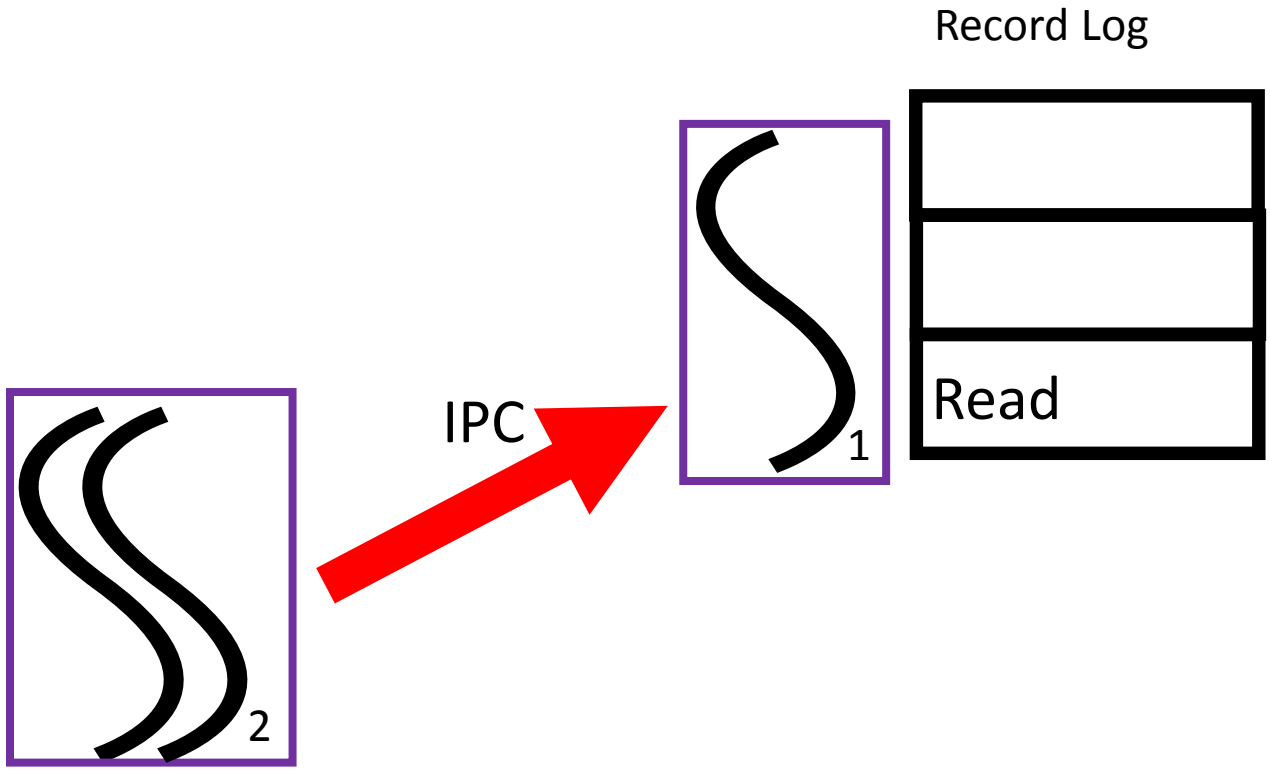
- Process level recording
  - ✓Efficient to replay
  - ×Uses extra disk space
  - ×No Inter-process tracking

Recording Granularity

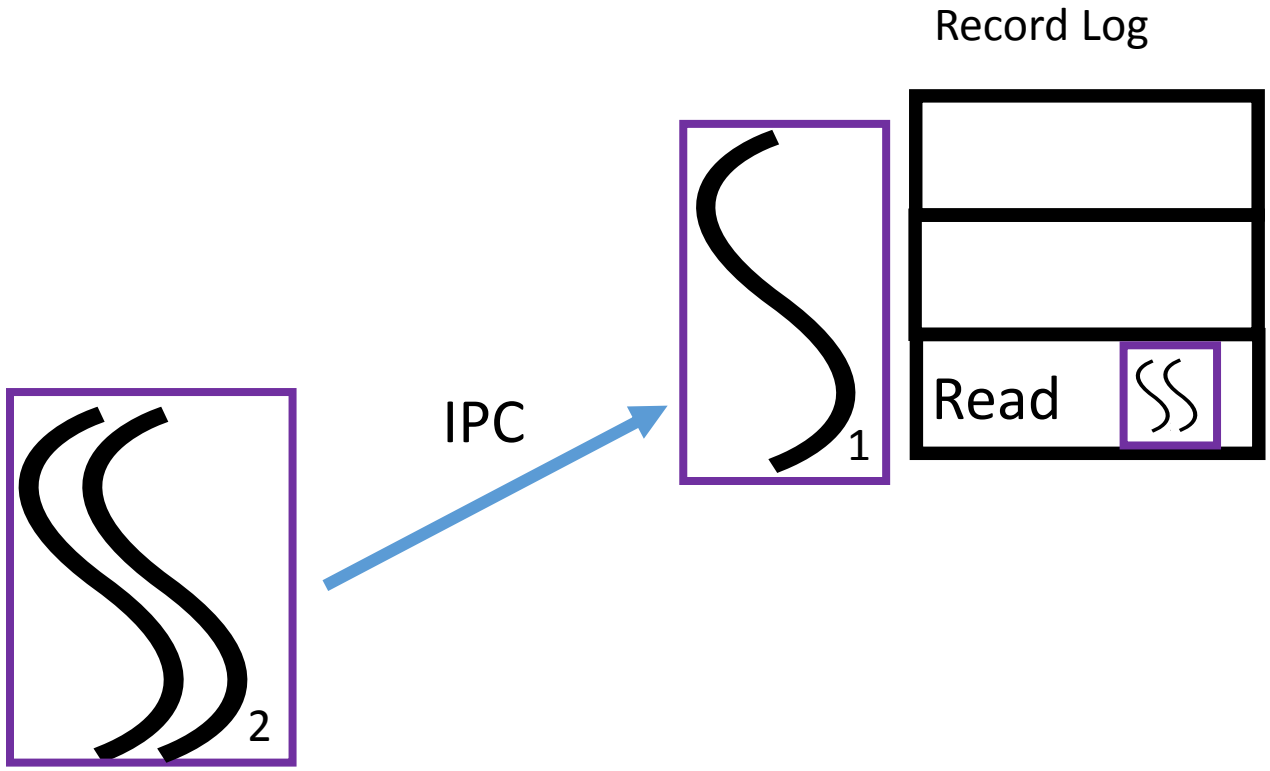


- Process group recording
  - ✓Efficient to replay
  - ✓Reasonable disk space
  - ×No Inter-process tracking

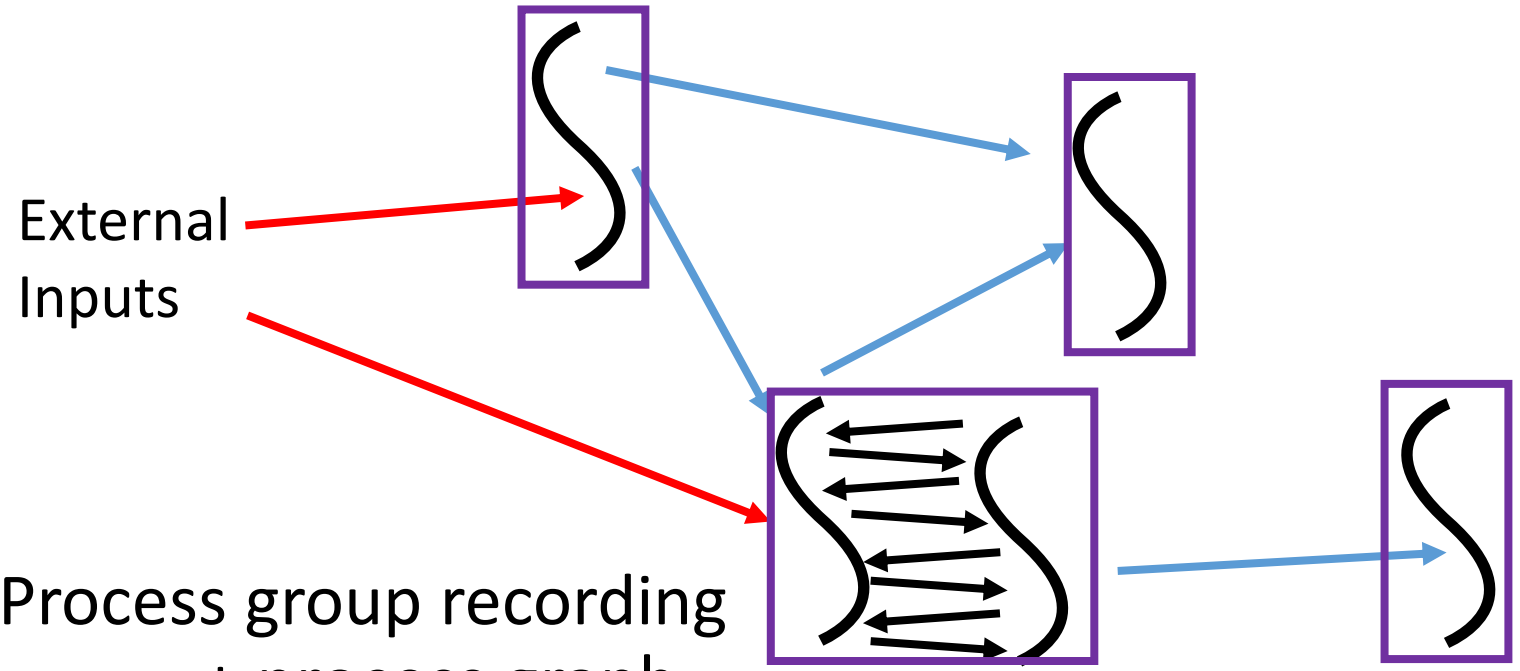
Implementation – Process Graph



Implementation – Process Graph

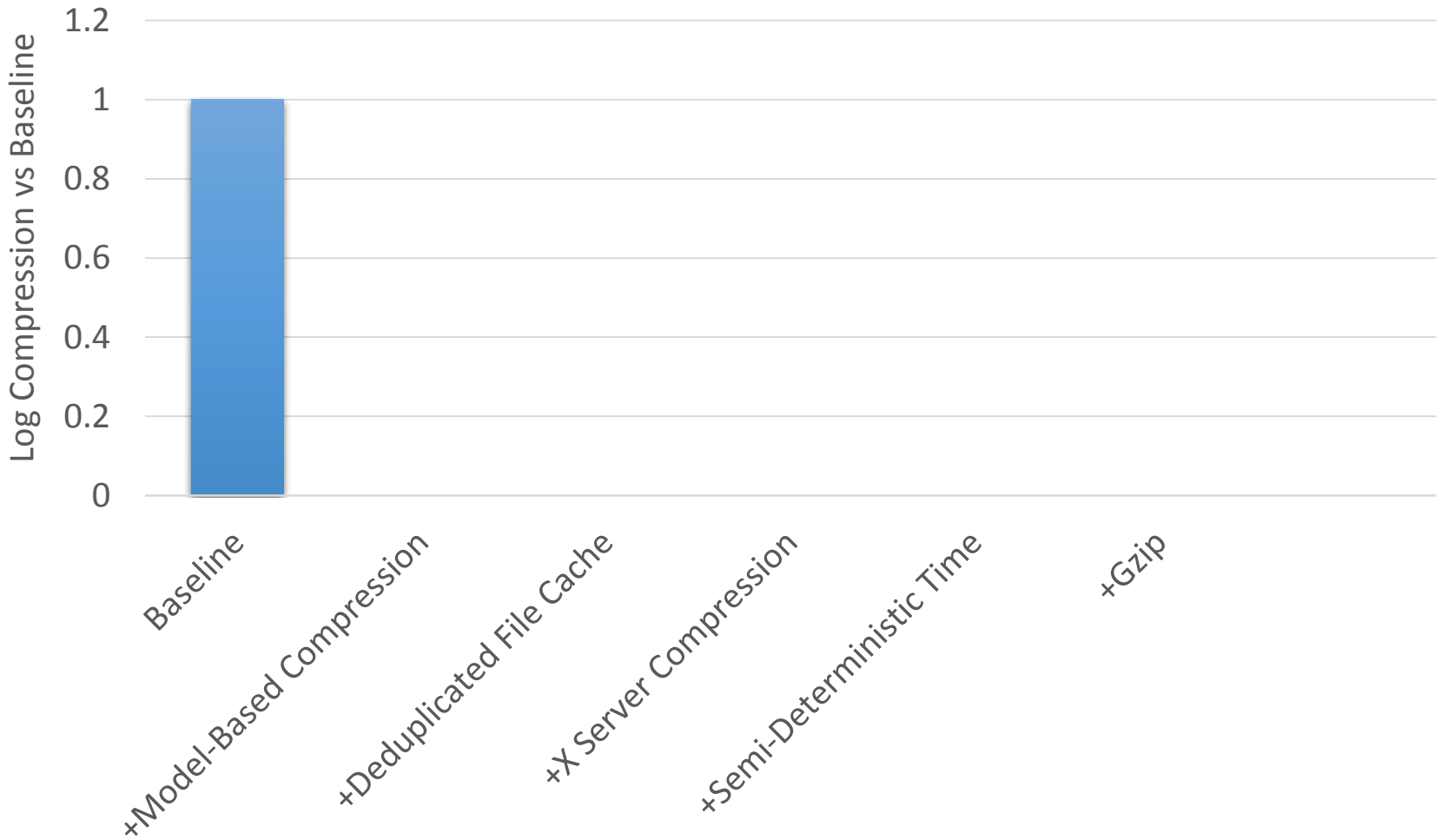


Recording

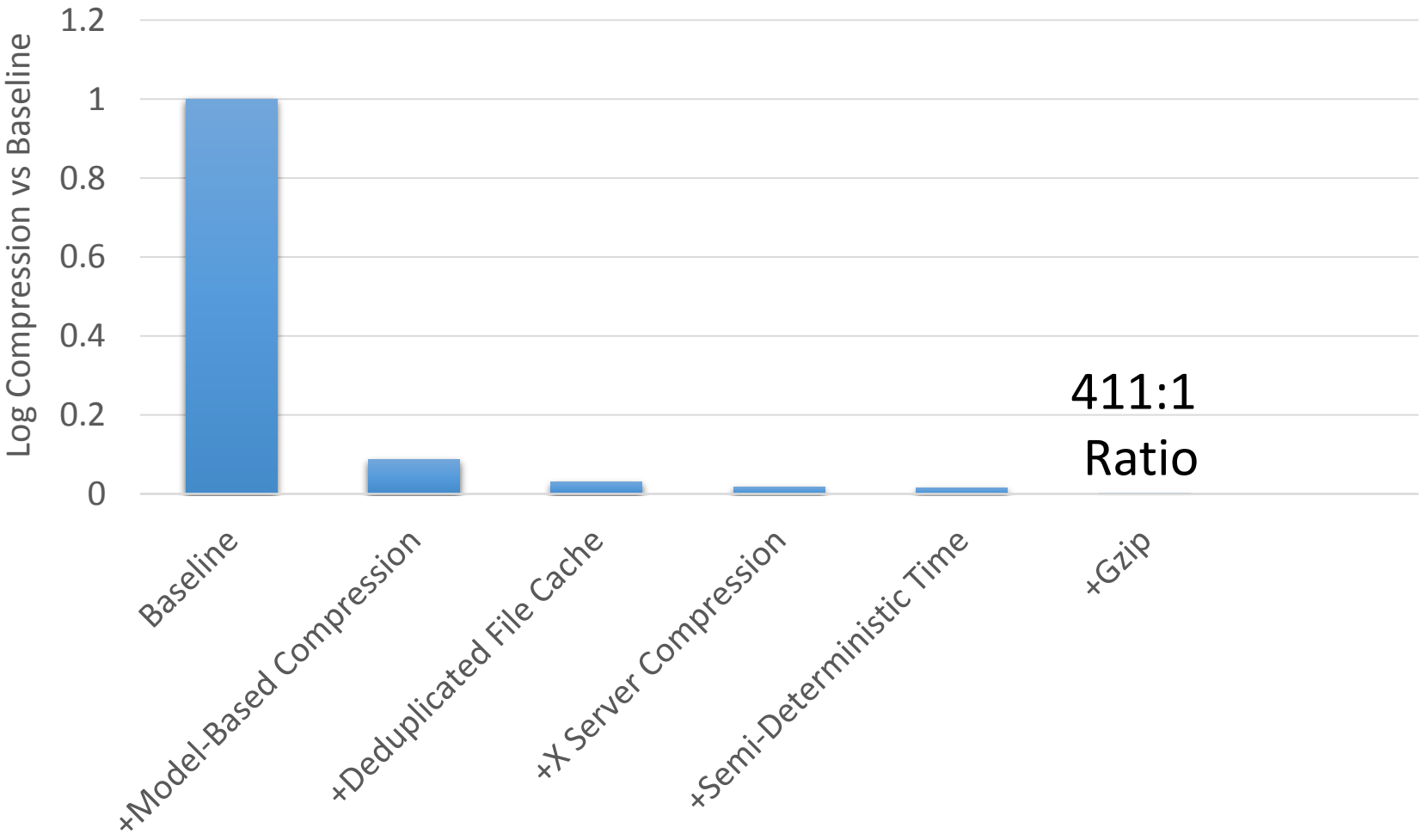


- Process group recording
  + process graph
  - ✓Efficient to replay
  - ✓Reasonable disk space
  - ✓Inter-process tracking

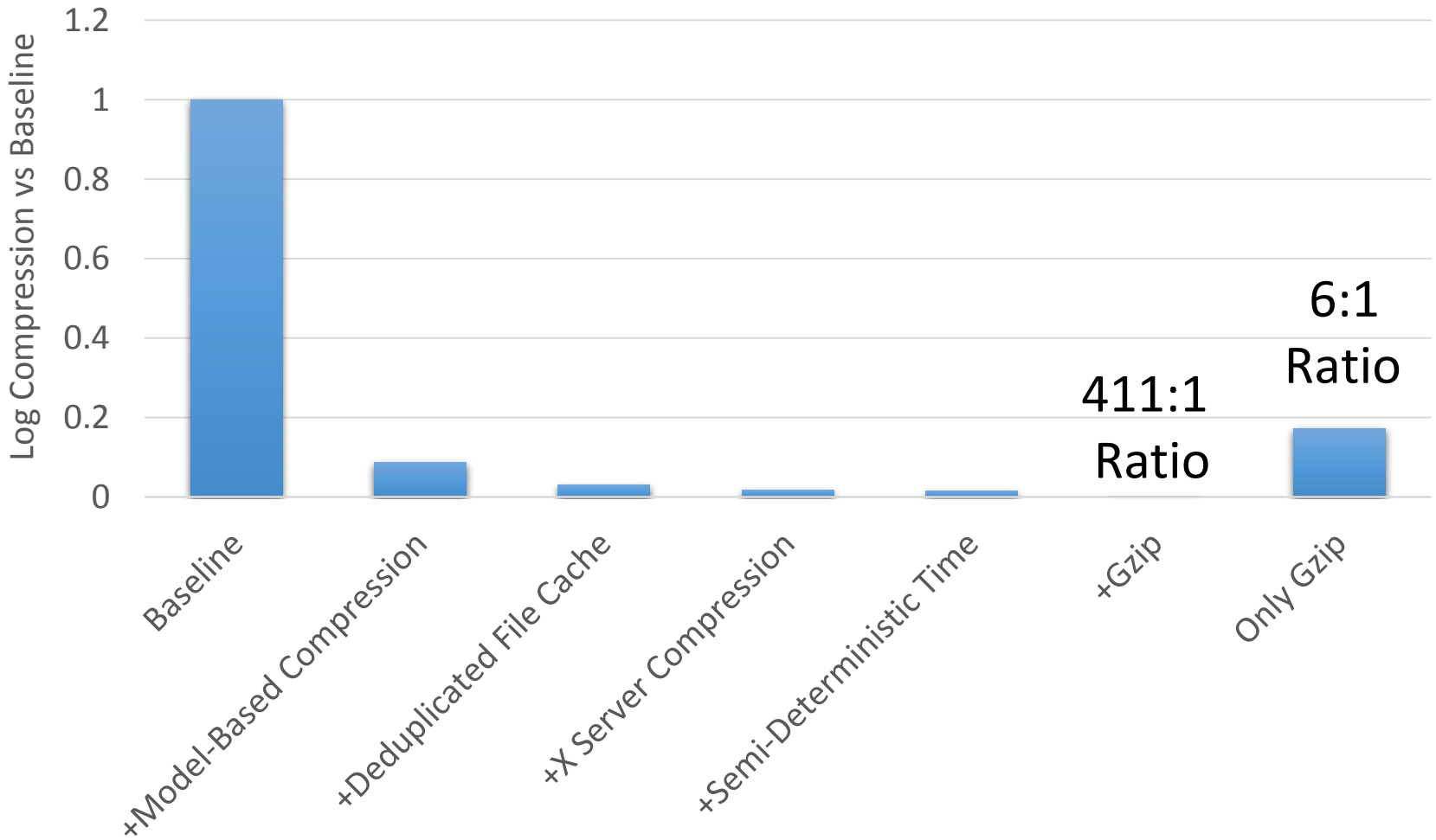
Space Optimizations



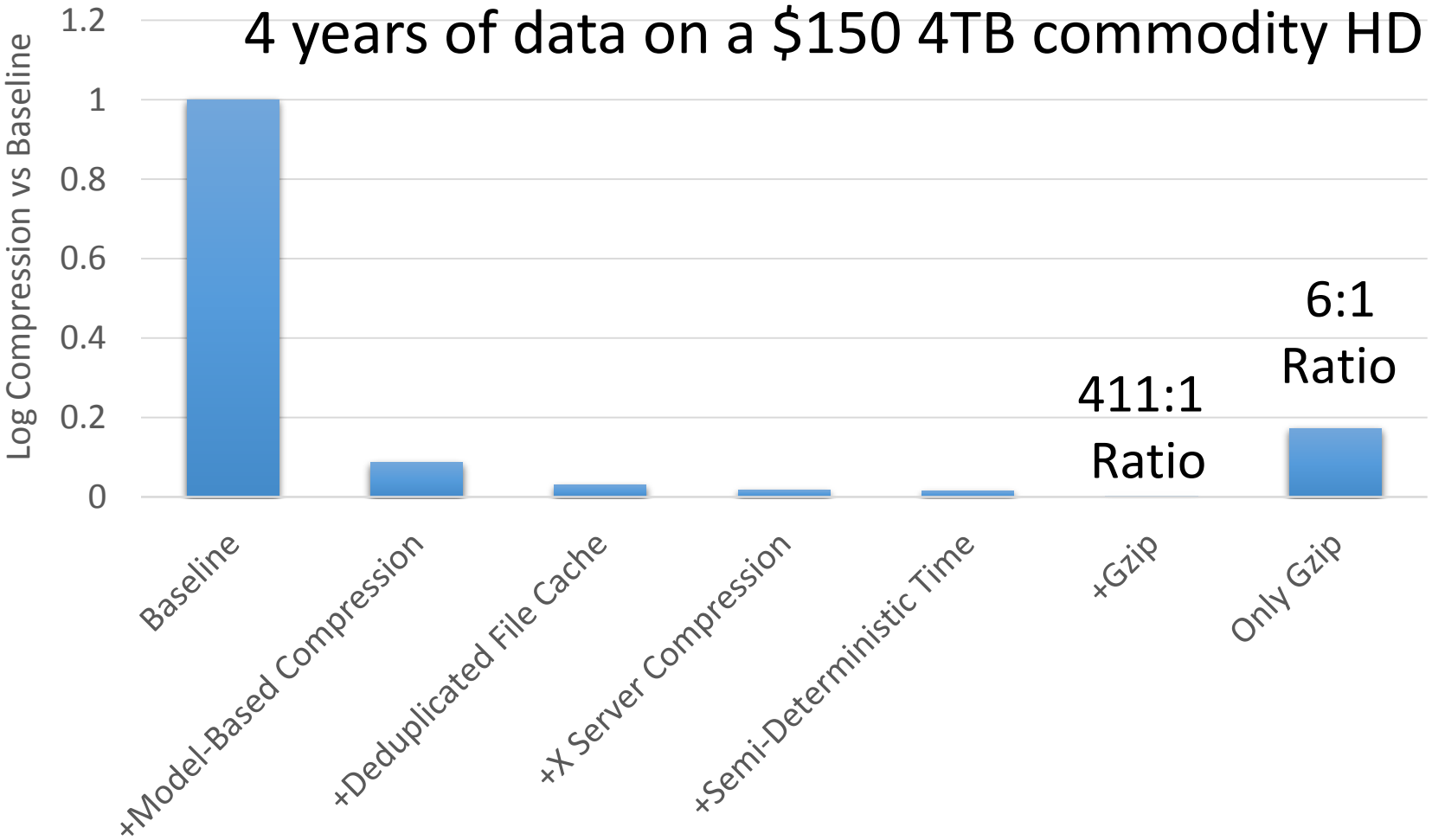
Space Optimizations



Space Optimizations



Space Optimizations



Model-Based Compression

- Formulate a model of a typical execution
  - Only record deviations from that model

```
ret_val  =  sys_read (fd, buffer, count);
```

usually equal

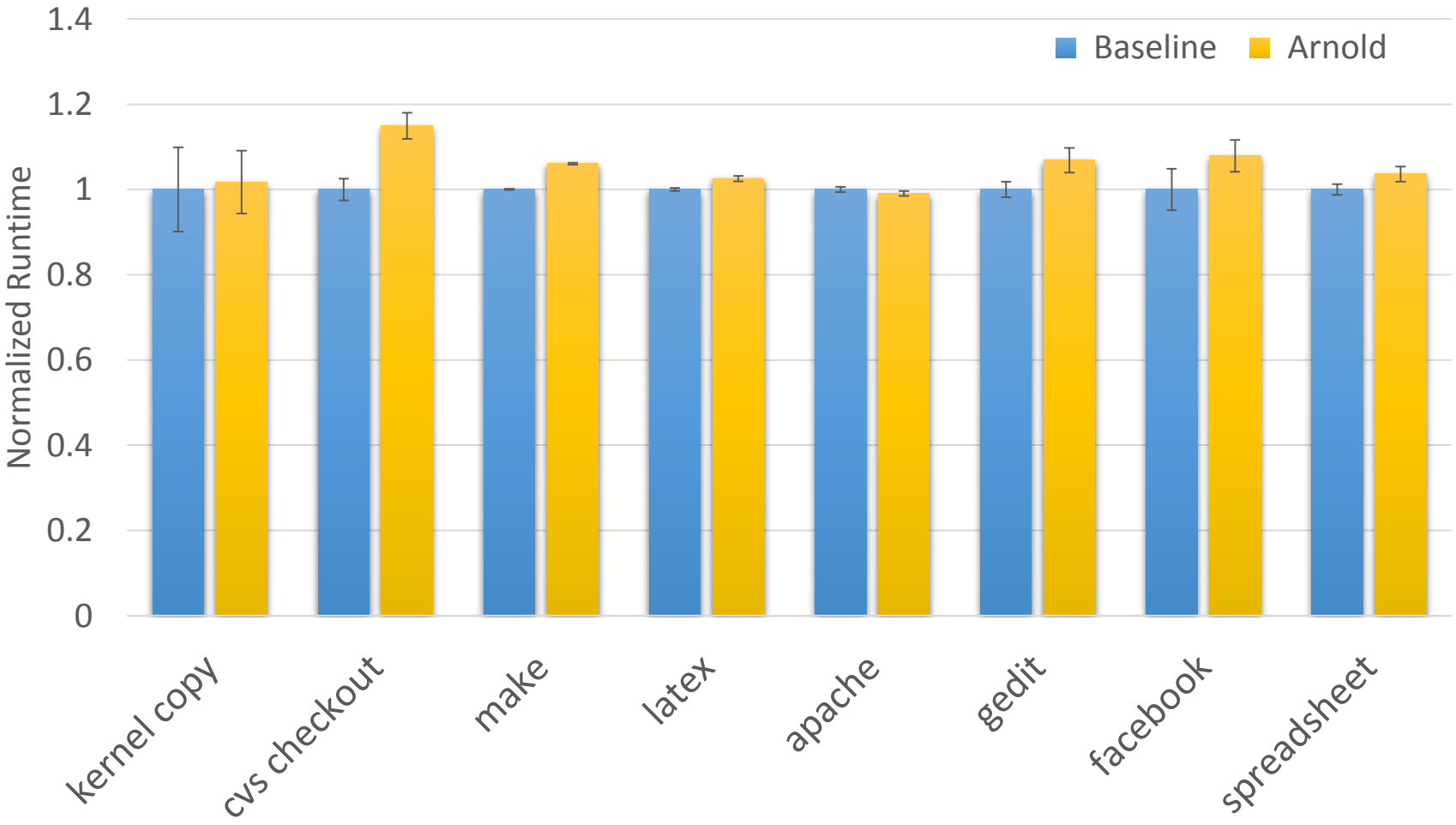
- Idea: Partial determinism
  - Encourage the program to conform to the model

Semi-Deterministic Time

- Frequent time queries are non-deterministic
- Use partially deterministic clock at record time
  - Real time clock & deterministic clock
  - Bound deviation

```
if (deterministic_clock – real_time_clock < threshold) {
        adjust deterministic_clock
        record deviation
}
return deterministic_clock
```

Performance Evaluation



Overview

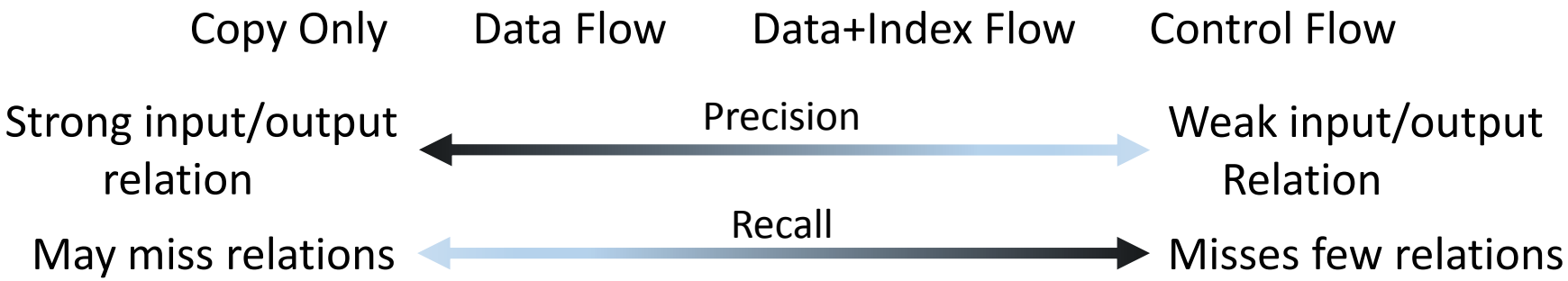
- Introduction
- Motivation
- How Arnold remembers all state
- How Arnold supports lineage queries
- Conclusion

Querying Lineage

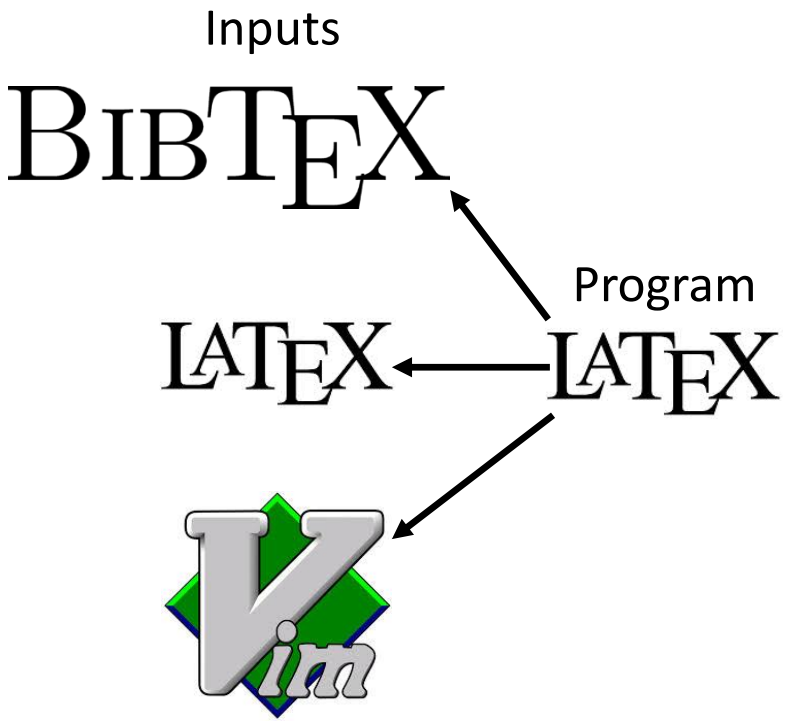
- Two types of queries:
  - Reverse: Where did this data come from?
  - Forward: What did this data affect?
- How does Arnold support these queries?
  - User specifies initial state
  - Trace the lineage of the computation
    - Intra-process tracking
    - Inter-process tracking

Intra-Process Lineage

- Use taint tracking for intra-process causality
  - Run retroactively, on recorded execution
  - Parallelizable
- Arnold supports several notions of causality:

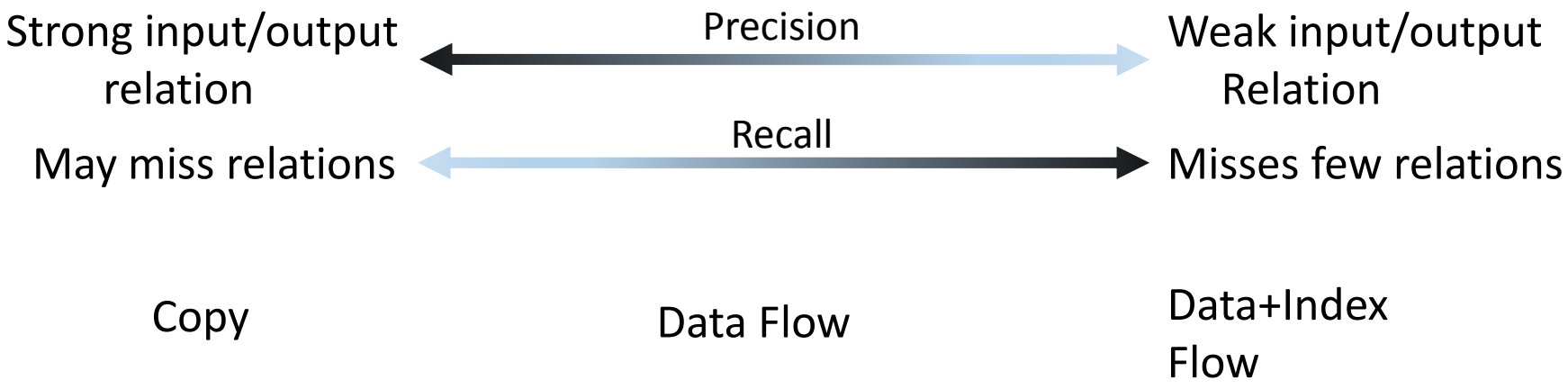


Intra-Process Lineage



Which linkage tool should Arnold use?

Intra-Process Lineage

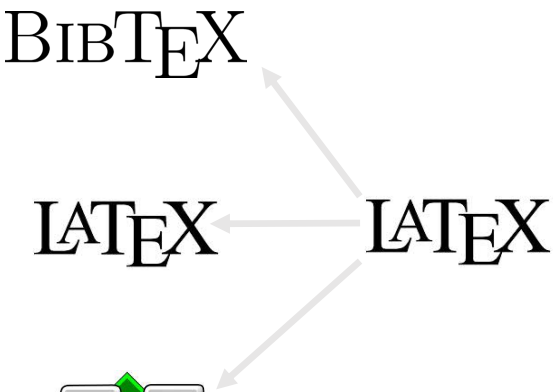


Intra-Process Lineage

Strong input/output relation ⟵ Precision ⟶ Weak input/output Relation

May miss relations ⟵ Recall ⟶ Misses few relations

Copy | Data Flow | Data+Index Flow



Intra-Process Lineage

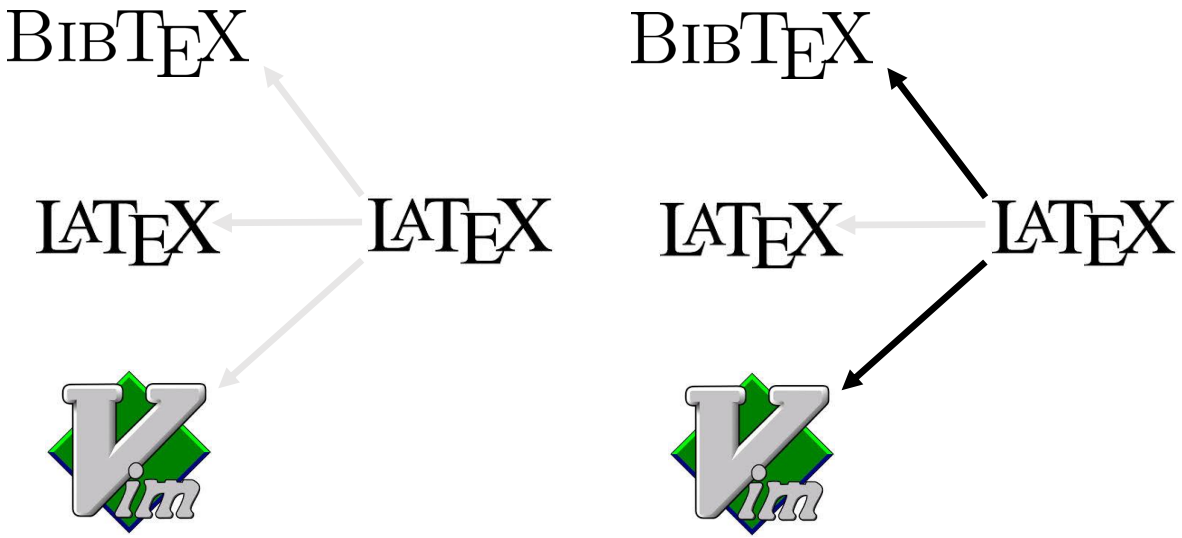
Strong input/output relation ⟵ Precision ⟶ Weak input/output Relation

May miss relations ⟵ Recall ⟶ Misses few relations

Copy

Data Flow

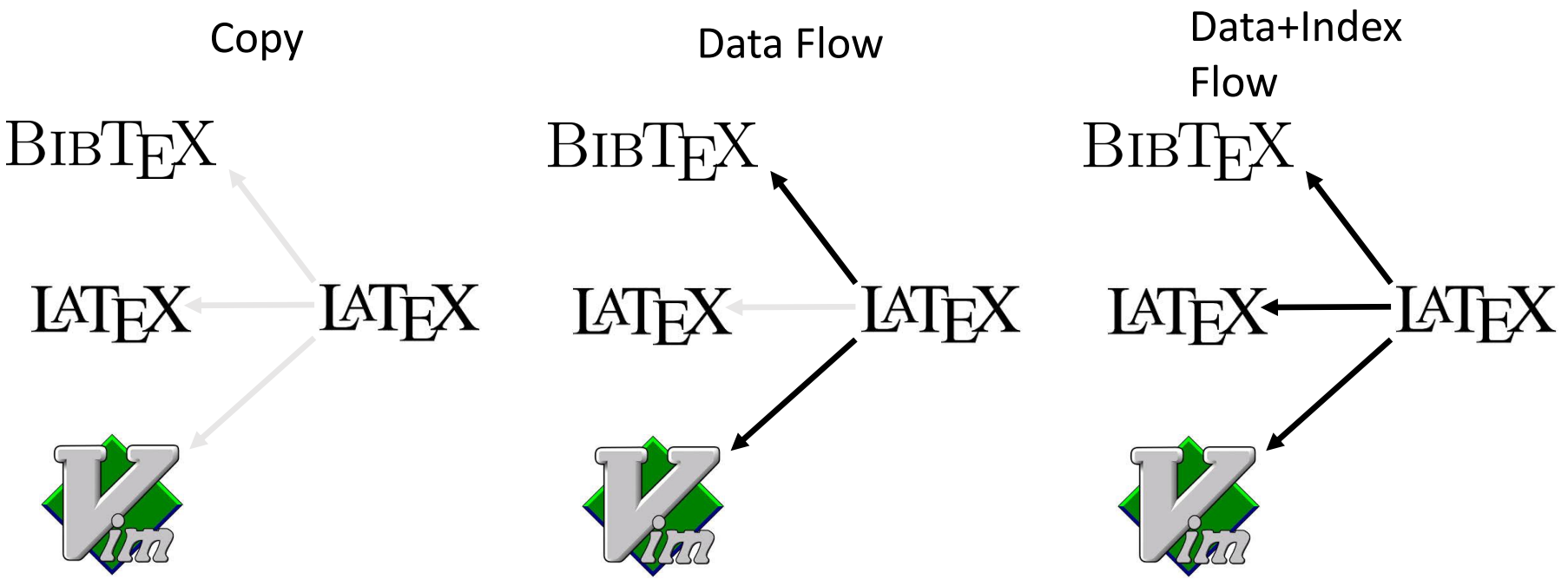
Data+Index Flow



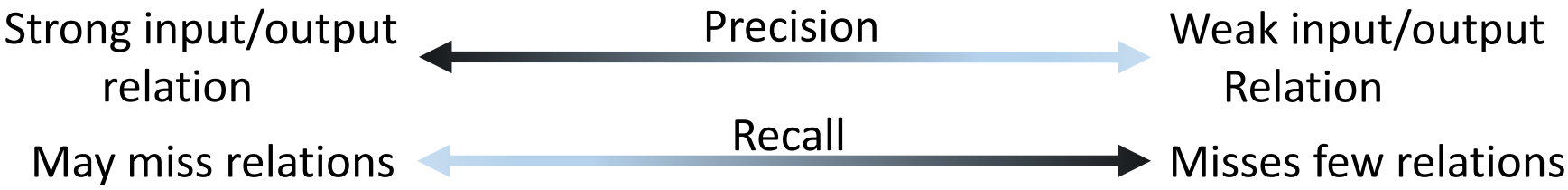
Intra-Process Lineage

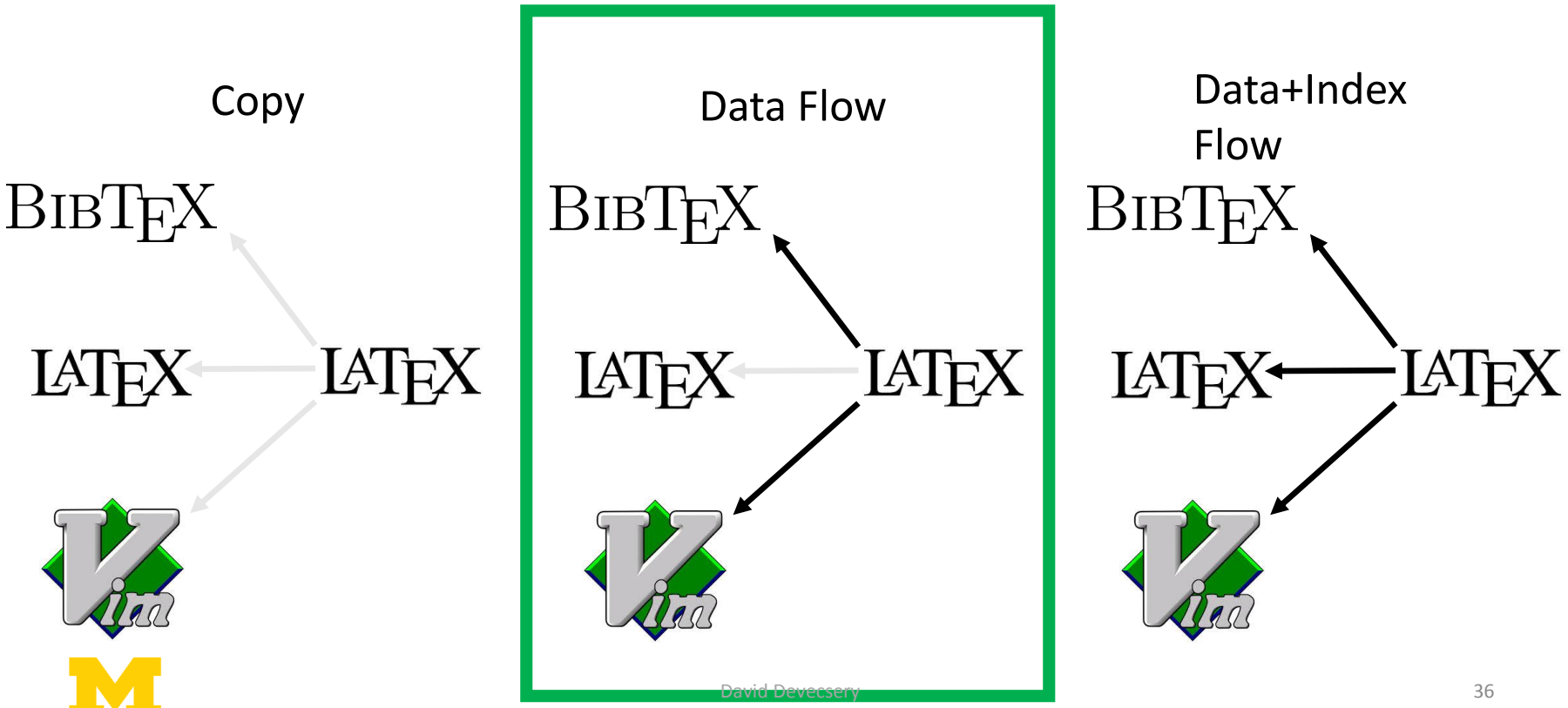
Strong input/output relation ⟵ Precision ⟶ Weak input/output Relation

May miss relations ⟵ Recall ⟶ Misses few relations



Intra-Process Lineage

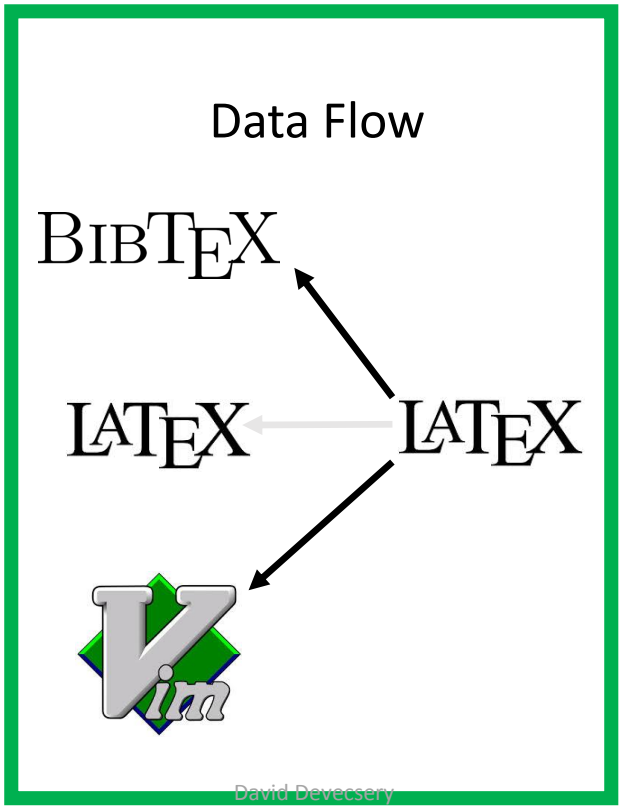




Intra-Process Lineage

Strong input/output relation ← Precision → Weak input/output Relation

May miss relations ← Recall → Misses few relations

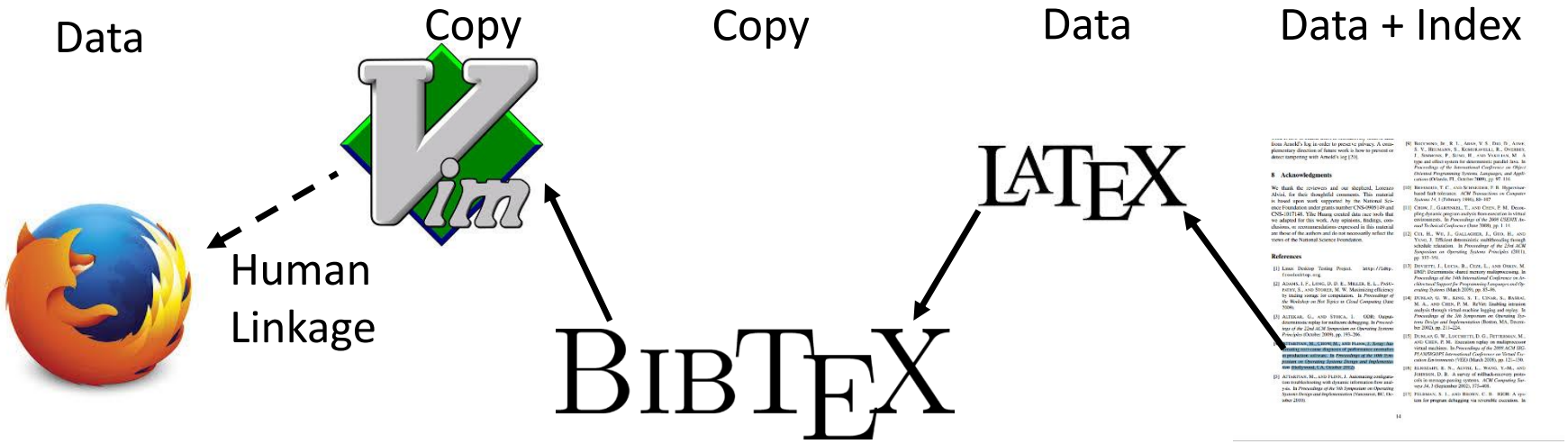


Arnold selects the most precise tool with at least one result

Inter-Process Lineage

- Two notions of inter-process linkage
- Process graph
  - Tracks lineage through inter-process communication
  - Precise
  - Captures group to group communication
- Human linkage
  - Handles relations between user inputs and outputs
  - Infers linkages based on data content and time
  - Imprecise – may have false negatives and false positives
  - Can capture linkages the process graph can miss

Evaluation – Wrong Reference



- Few false positives (font files, latex sty files, libc.so, libXt.so)
- No false negatives

Record Time	Replay Time	Replay + Pin Time	Query Time
96.1s	2.2s	70.0s	209.5s

Evaluation – Heartbleed

Data + Index Data + Index Data + Index

- No false positives or negatives

Record Time	Replay Time	Replay + Pin Time	Query Time
230.3s	0.4s	139.5s	235.1s

Conclusion

- Eidetic Systems are powerful tools
  - Complete vision into past computation
  - Answer powerful queries about state's lineage
- Arnold – First practical Eidetic System
  - Low runtime overhead
  - 4 years of computation on a commodity HD
  - Supports powerful lineage queries
- Code is released
  https://github.com/endplay/omniplay

Questions?

Backup Slides

Cloud Storage

- Future work
- Two approaches:
  - Statically served content
  - Distributed replay system

Related Works

- Execution Mining (Tralfamadore)
- DejaVu
- RAIL

User Study Log-Sizes

Users	Days	Groups Per Day	Storage Utilization (MB)			
			RAW File Cache	Logs	Filemap	Total
A	25	995	475	267	36	779
B	24	475	1095	936	339	2064
C	21	26122	869	350	690	1910
D	16	3339	1675	838	838	2594