

+

+

# How to Log All Filesystem Operations

(while only logging 1% of them)

David F. Bacon

IBM T.J. Watson Research Center

+

1

## Introduction

Problem: Logging for transparent recovery.

- Pessimistic: Borg et al (Nixdorf/Auragen Fault-Tolerant Unix)
- Optimistic: Strom and Yemini (Optimistic Recovery) and variants.

We show that the technique of Null Logging described by Strom, Bacon, and Yemini is highly successful in at least one environment: single-user workstation filesystems.

# Process Model

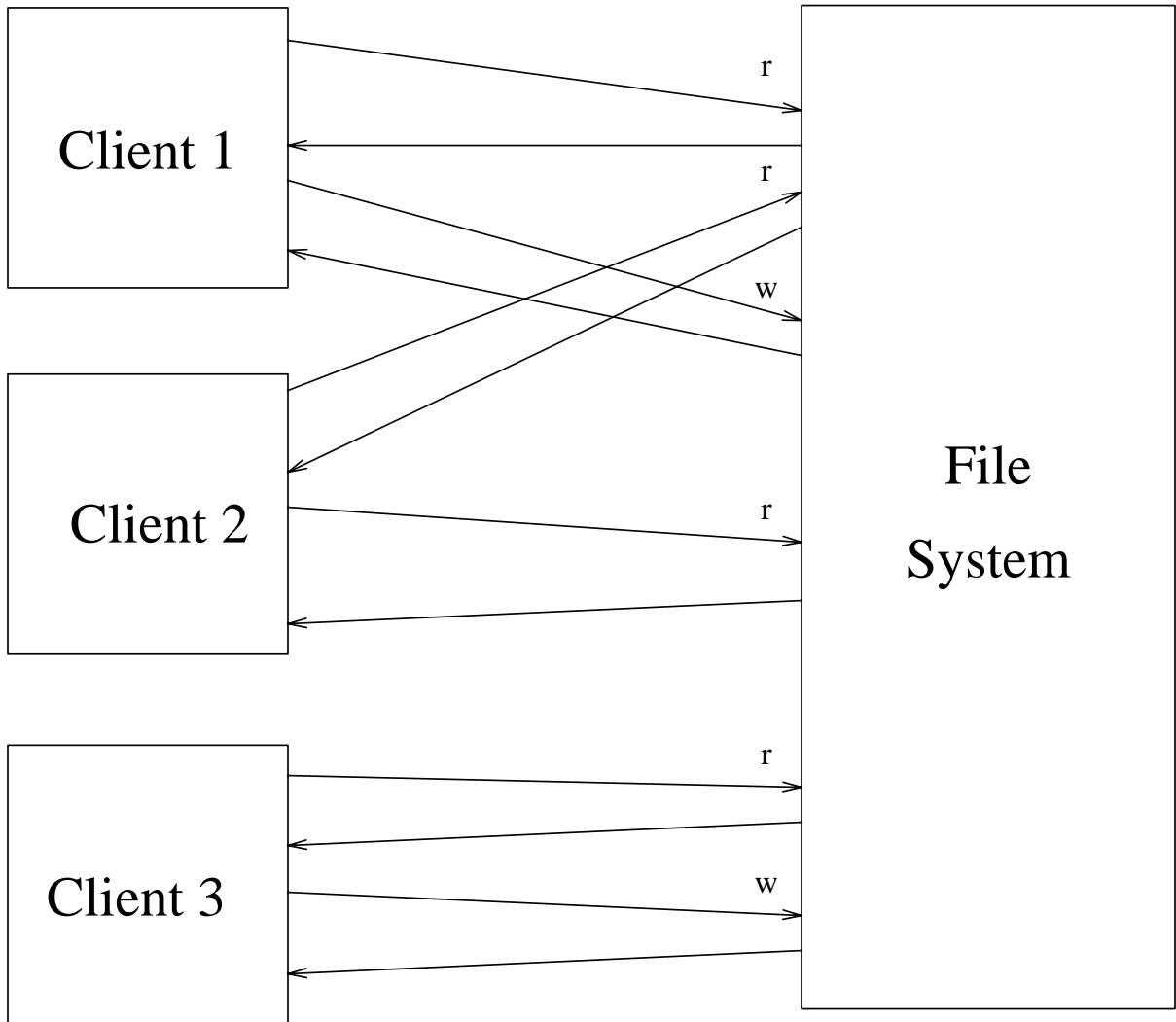
Our model assumes:

- CSP-like process structure
- piece-wise determinism
- all non-deterministic events are messages (WLG)
- fail-stop
- stable storage

+

+

# Filesystem Model



+

4

## Logging

- data (volatile logging)
- arrival order (null logging)

Logging of the arrival order is the only remaining bottleneck for committing (optimistic) or sending (pessimistic) messages.

+

+

Event	Op	Log Record
0	$r_1$	$r_1 \mid 0$
1	$r_1$	
2	$w_1$	
3	$r_1$	
4	$r_2$	$r_2 \mid 3$
5	$r_2$	
6	$r_3$	$r_3 \mid 1$
7	$r_3$	
8	$w_3$	
9	$w_4$	$w_4 \mid 2$

+

+

+

Run	Events	Null-Logged	Percentage
SU1	131246	124100	94.6%
SU2	68840	65902	95.7%
SU Total	200086	190002	95.0%
FS 1	2235574	2078142	92.9%
FS 2	1210219	1126197	93.0%
TS	1297275	1102723	85.0%

Null Logging Effectiveness: Predictor 1  
(Same Sender)

+

7

+

+

Event	Op	Log Record
0	$r_1$	
1	$r_1$	
2	$w_1$	$w_1 \mid (1,2)$
3	$r_1$	
4	$r_2$	
5	$r_2$	
6	$r_3$	
7	$r_3$	
8	$w_3$	$w_3 \mid (1,1) (2,2) (3,2)$
9	$w_4$	$w_4 \mid ()$

Logging Actions of the Read Predictor

+

8

+

+

Run	Events	Null-Logged	Percentage
SU 1	131246	120348	91.7%
SU 2	68840	65123	94.6%
SU Total	200086	185471	92.7%
FS 1	2235574	1894433	84.7%
FS 2	1210219	1045770	86.4%
TS	1297275	995047	76.7%

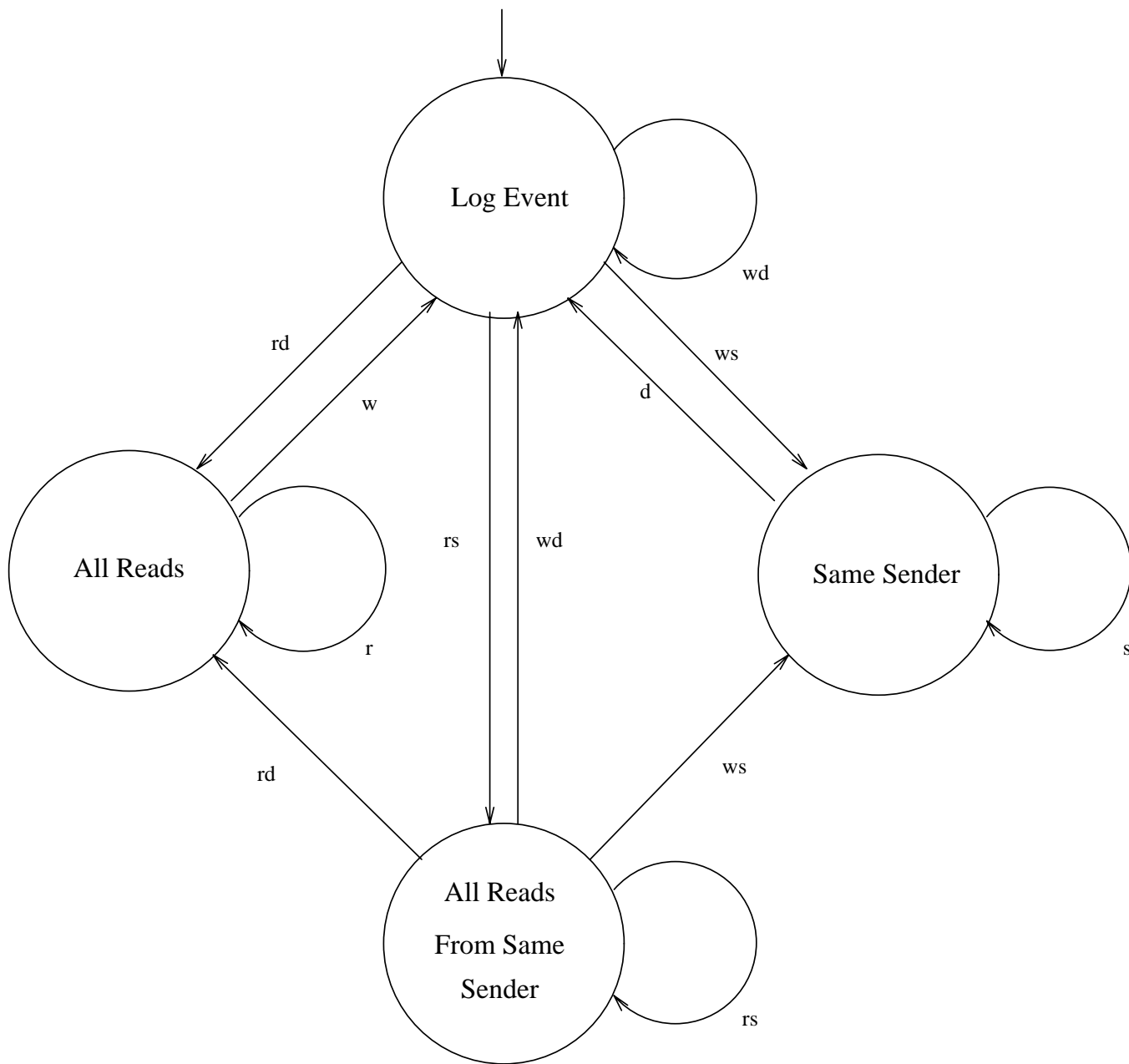
Null Logging Effectiveness: Predictor 2  
(Read)

+

9

+

+



+

+

+

Event	Op	Log Record
0	$r_1$	$r_1 \mid (1,0)$
1	$r_1$	
2	$w_1$	
3	$r_1$	
4	$r_2$	$r_2 \mid (1,3)$
5	$r_2$	
6	$r_3$	
7	$r_3$	
8	$w_3$	$w_3 \mid (2,1) (3,2)$
9	$w_4$	$w_4$

+

+

+

Run	Events	Null-Logged	Percentage
SU 1	131246	129152	98.4%
SU 2	68840	68141	99.0%
SU Total	200086	197293	98.6%
FS 1	2235574	2144626	95.9%
FS 2	1210219	1168060	96.5%
TS	1297275	1200002	92.5%

Null Logging Effectiveness: Predictor 3  
(Hybrid)

+

+

+

Event	Op	Anticipatory	Plain Hybrid
0	$r_1$	$r_1 \mid (1,0)$	$r_1 \mid (1,0)$
1	$r_1$		
2	$w_1$	$w_1 \mid (1,1)$	
3	$r_1$		
4	$r_2$	$r_2 \mid (1,1)$	$r_2 \mid (1,3)$
5	$r_2$		
6	$r_3$	$r_3 \mid (2,1)$	
7	$r_3$		
8	$w_3$	$w_3 \mid (3,1)$	$w_3 \mid (2,1) (3,2)$
9	$w_4$	$w_4$	$w_4$

+

+

+

Run	Events	Null-Logged	Percent	Anticipatory	Percent
SU 1	131246	130078	99.1%	16876	12.9%
SU 2	68840	68437	99.4%	6252	9.1%
SU Total	200086	198515	99.2%	23128	11.6%
FS 2	1210219	1177363	97.2%	197935	16.3%
TS	1297275	1210718	93.3%	352776	27.1%

Null Logging Effectiveness: Predictor 4 (Hybrid with Anticipatory Logging)

Note: Assumes anticipatory records stable after one intervening event.

+

## Conclusions

Null logging is highly effective on workstations: over 99%.

Volatile Logging + Null Logging can greatly improve commit latency of both optimistic and pessimistic recovery, and can substantially improve performance of pessimistic recovery.

Null logging should be further studied in time-sharing environments and for less specialized processes.

Anticipatory logging is a nice idea, but involves tradeoffs – further study is required. (Similar to Lempel-Ziv encoding)

Adaptive predictor functions probably will be worth the cost in less predictable environments.

## Future Work

Do measurements on timesharing systems.

Examine possibility of adaptive protocol that switches between same-sender and round-robin predictor, similar to the Urn Protocol for carrier sense networks.

If timesharing systems present a problem, try applying these techniques at disk partition or even single-file granularities.

Examine the likelihood of success of anticipatory logging – how often are the writes fast enough?

+

+

# How to Log All Filesystem Operations

(while only logging 1% of them)

David F. Bacon

IBM T.J. Watson Research Center

+

17

## Introduction

Problem: Logging for transparent recovery.

- Pessimistic: Borg et al (Nixdorf/Auragen Fault-Tolerant Unix)
- Optimistic: Strom and Yemini (Optimistic Recovery) and variants.

We show that the technique of Null Logging described by Strom, Bacon, and Yemini is highly successful in at least one environment: single-user workstation filesystems.

## Process Model

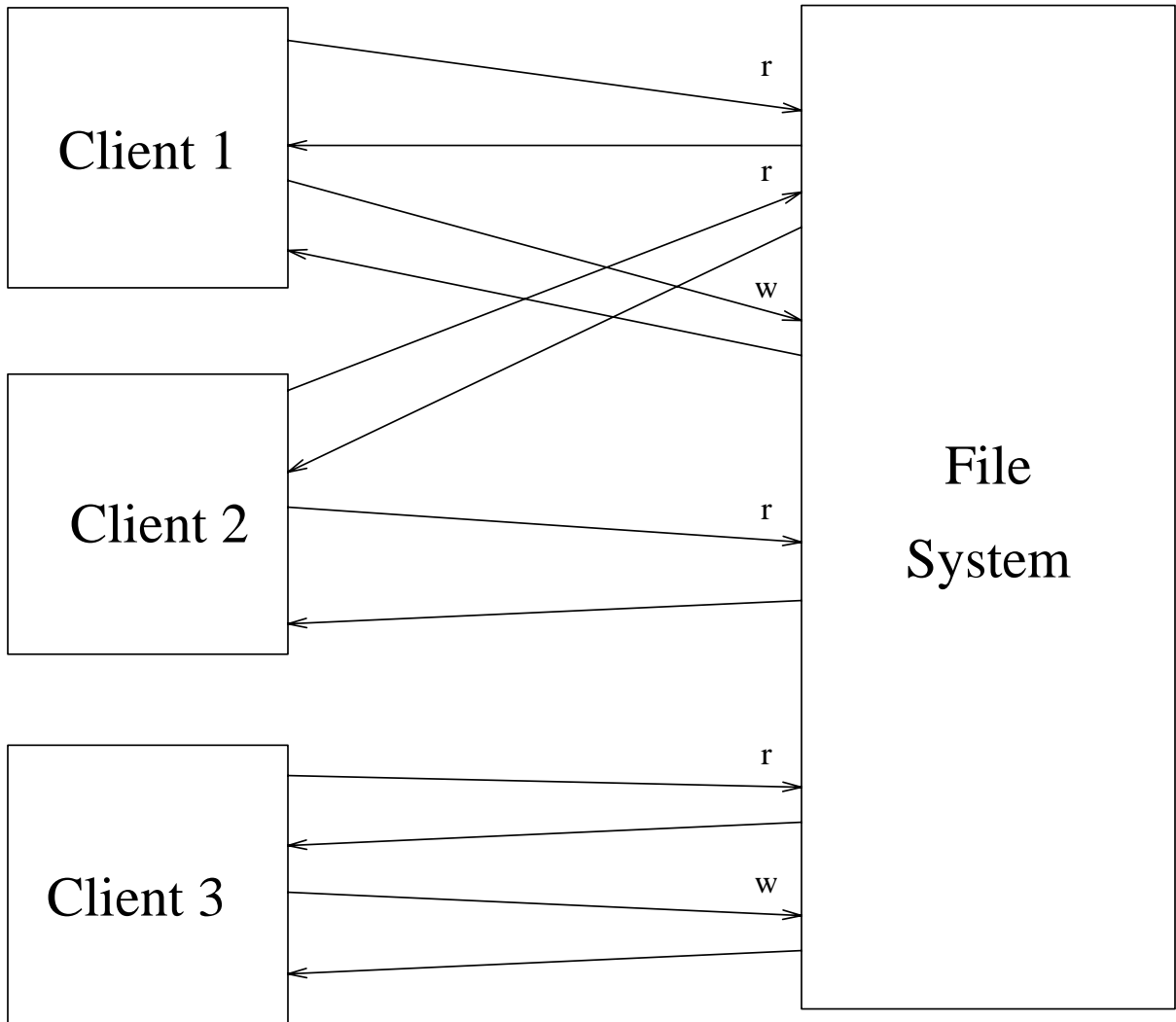
Our model assumes:

- CSP-like process structure
- piece-wise determinism
- all non-deterministic events are messages (WLG)
- fail-stop
- stable storage

+

+

# Filesystem Model



+

## Logging

- data (volatile logging)
- arrival order (null logging)

Logging of the arrival order is the only remaining bottleneck for committing (optimistic) or sending (pessimistic) messages.

+

+

Event	Op	Log Record
0	$r_1$	$r_1 \mid 0$
1	$r_1$	
2	$w_1$	
3	$r_1$	
4	$r_2$	$r_2 \mid 3$
5	$r_2$	
6	$r_3$	$r_3 \mid 1$
7	$r_3$	
8	$w_3$	
9	$w_4$	$w_4 \mid 2$

+

+

+

Run	Events	Null-Logged	Percentage
SU1	131246	124100	94.6%
SU2	68840	65902	95.7%
SU Total	200086	190002	95.0%
FS 1	2235574	2078142	92.9%
FS 2	1210219	1126197	93.0%
TS	1297275	1102723	85.0%

Null Logging Effectiveness: Predictor 1  
(Same Sender)

+

+

+

Event	Op	Log Record
0	$r_1$	
1	$r_1$	
2	$w_1$	$w_1 \mid (1,2)$
3	$r_1$	
4	$r_2$	
5	$r_2$	
6	$r_3$	
7	$r_3$	
8	$w_3$	$w_3 \mid (1,1) (2,2) (3,2)$
9	$w_4$	$w_4 \mid ()$

Logging Actions of the Read Predictor

+

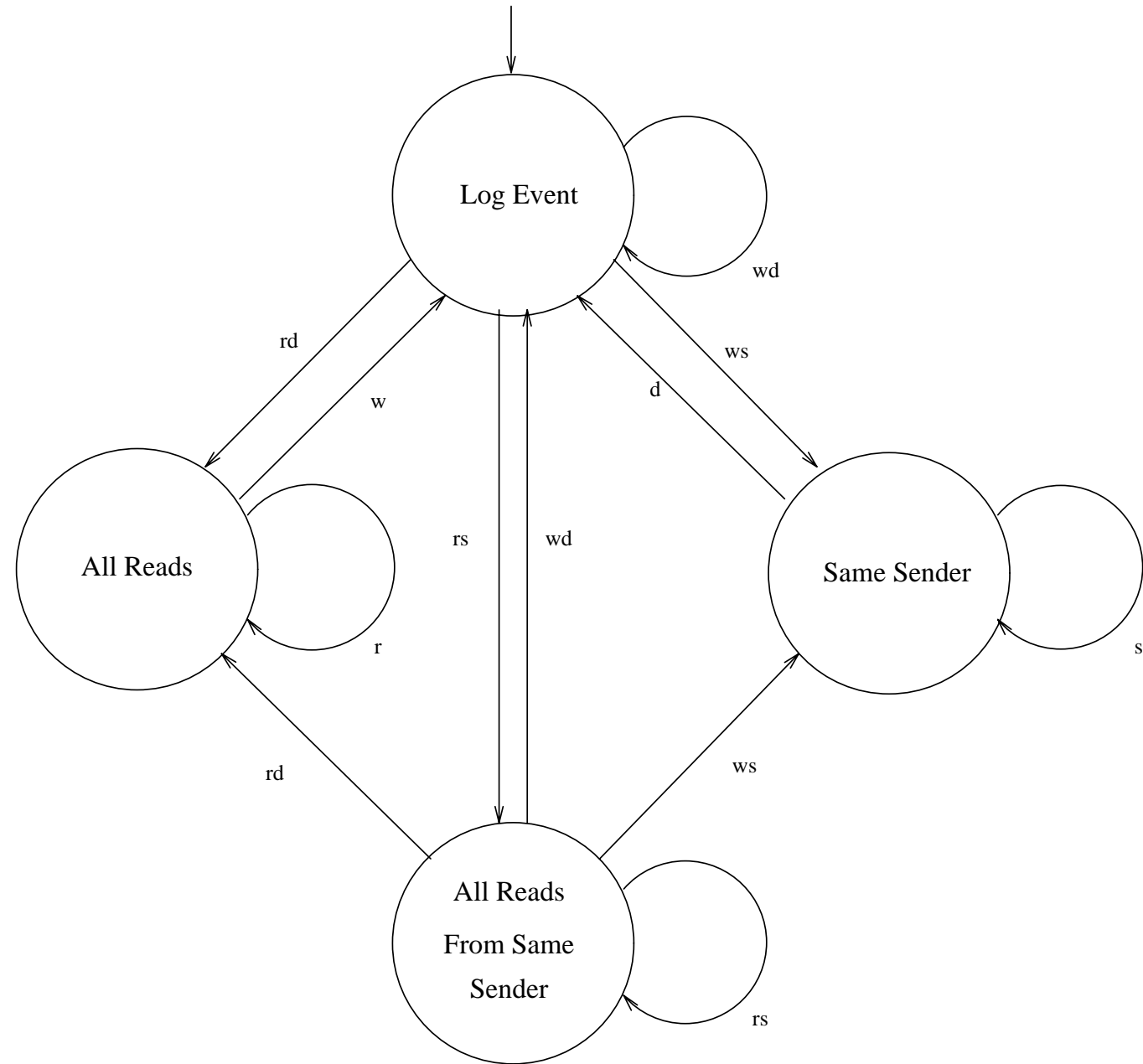
+

+

Run	Events	Null-Logged	Percentage
SU 1	131246	120348	91.7%
SU 2	68840	65123	94.6%
SU Total	200086	185471	92.7%
FS 1	2235574	1894433	84.7%
FS 2	1210219	1045770	86.4%
TS	1297275	995047	76.7%

Null Logging Effectiveness: Predictor 2  
(Read)

+



+

+

Event	Op	Log Record
0	$r_1$	$r_1 \mid (1,0)$
1	$r_1$	
2	$w_1$	
3	$r_1$	
4	$r_2$	$r_2 \mid (1,3)$
5	$r_2$	
6	$r_3$	
7	$r_3$	
8	$w_3$	$w_3 \mid (2,1) (3,2)$
9	$w_4$	$w_4$

+

+

+

Run	Events	Null-Logged	Percentage
SU 1	131246	129152	98.4%
SU 2	68840	68141	99.0%
SU Total	200086	197293	98.6%
FS 1	2235574	2144626	95.9%
FS 2	1210219	1168060	96.5%
TS	1297275	1200002	92.5%

Null Logging Effectiveness: Predictor 3  
(Hybrid)

+

+

+

Event	Op	Anticipatory	Plain Hybrid
0	$r_1$	$r_1 \mid (1,0)$	$r_1 \mid (1,0)$
1	$r_1$		
2	$w_1$	$w_1 \mid (1,1)$	
3	$r_1$		
4	$r_2$	$r_2 \mid (1,1)$	$r_2 \mid (1,3)$
5	$r_2$		
6	$r_3$	$r_3 \mid (2,1)$	
7	$r_3$		
8	$w_3$	$w_3 \mid (3,1)$	$w_3 \mid (2,1) (3,2)$
9	$w_4$	$w_4$	$w_4$

+

+

+

Run	Events	Null-Logged	Percent	Anticipatory	Percent
SU 1	131246	130078	99.1%	16876	12.9%
SU 2	68840	68437	99.4%	6252	9.1%
SU Total	200086	198515	99.2%	23128	11.6%
FS 2	1210219	1177363	97.2%	197935	16.3%
TS	1297275	1210718	93.3%	352776	27.1%

Null Logging Effectiveness: Predictor 4 (Hybrid with Anticipatory Logging)

Note: Assumes anticipatory records stable after one intervening event.

+

## Conclusions

Null logging is highly effective on workstations: over 99%.

Volatile Logging + Null Logging can greatly improve commit latency of both optimistic and pessimistic recovery, and can substantially improve performance of pessimistic recovery.

Null logging should be further studied in time-sharing environments and for less specialized processes.

Anticipatory logging is a nice idea, but involves tradeoffs – further study is required. (Similar to Lempel-Ziv encoding)

Adaptive predictor functions probably will be worth the cost in less predictable environments.

## Future Work

Do measurements on timesharing systems.

Examine possibility of adaptive protocol that switches between same-sender and round-robin predictor, similar to the Urn Protocol for carrier sense networks.

If timesharing systems present a problem, try applying these techniques at disk partition or even single-file granularities.

Examine the likelihood of success of anticipatory logging – how often are the writes fast enough?