Malleable Flow for Time-Bounded Replica Consistency Control

THSS-ISE Lab

Yuqing Zhu[#1], Jianmin Wang[#2], Philip S. Yu[*#3]

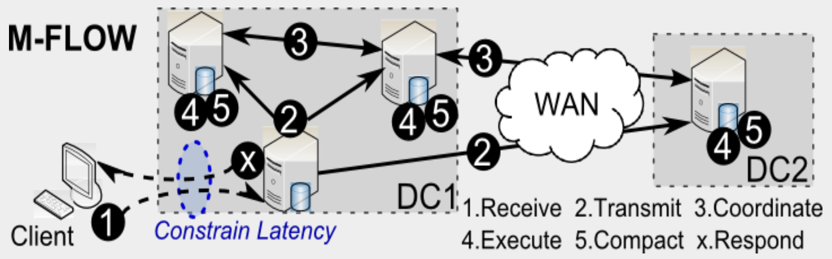
[#]Tsinghua University, School of Software

[*]University of Illinois at Chicago, Dept. of Computer Science

[1]zhuyuqing@gmail.com, [2]jimwang@tsinghua.edu.cn, [3]psyu@uic.edu

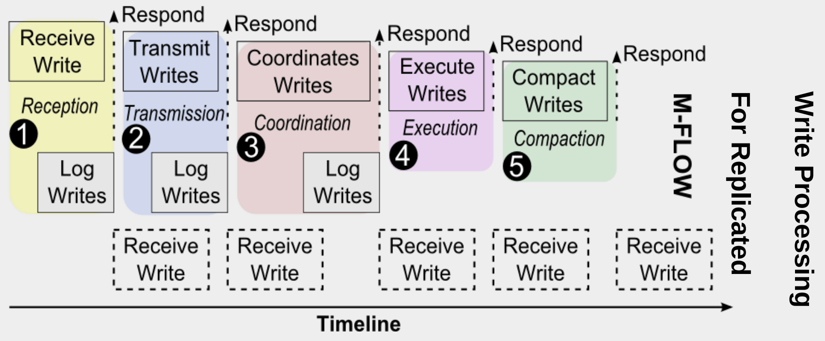
Tsinghua University

Big Picture


M-FLOW
Client
1
Constrain Latency
x
2
3
3
2
4 5
4 5
4 5
WAN
DC1
DC2
1.Receive 2.Transmit 3.Coordinate
4.Execute 5.Compact x.Respond

A system with malleable flow (M-flow) supports latency-bounded operations that maximize replica consistency within the given time. A better consistency status indicates a shorter latency for a following consistent read, or a more recent value returned by a following read within the same latency.

The M-flow replication process starts from when the request is received till when the response is sent. It is decomposed into six stages, i.e. reception, transmission, coordination, execution, compaction and acquisition.



The M-flow replication strategy allows update-anywhere, eager synchronous and lazy asynchronous replication simultaneously. The decomposed replication process enables this flexibility and the control of replica consistency (and latency) by reforming a suitable execution process with carefully selected stages and writes.

* This work is supported by the National Science and Technology Major Special Projects of China under Grant No. 2010ZX01042-002-002-01, and the National Natural Science Foundation of China under Grant No. 61073005.

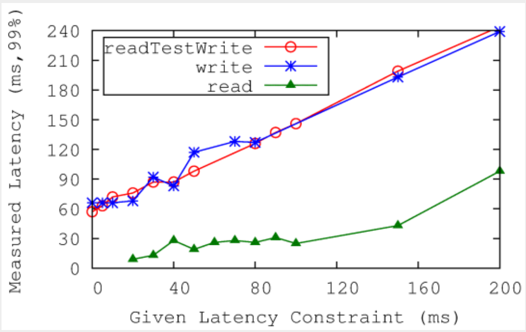
Motivation

- Replication is important for availability, scalability and reliability in large-scale systems
- Trade-off between replica consistency and availability, latency exists
- Guaranteeing best replica consistency within a given latency
  - ○ Not provided yet
  - ○ But highly DESIRABLE

Key Idea

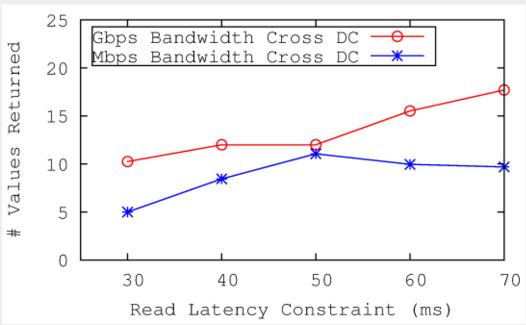
- To decompose the replication process into stoppable stages and then into a directed graph of ordered minute steps
- To reform an execution flow by finding a path of steps in the graph that
  - ○ Meets the time requirement
  - ○ Maximizes replica consistency status
  - ○ Guarantees durability and fault tolerance

Bounded Time & Traded Consistency



Bounded Latencies (the given latency bounds versus the measured 99 percentile latencies):

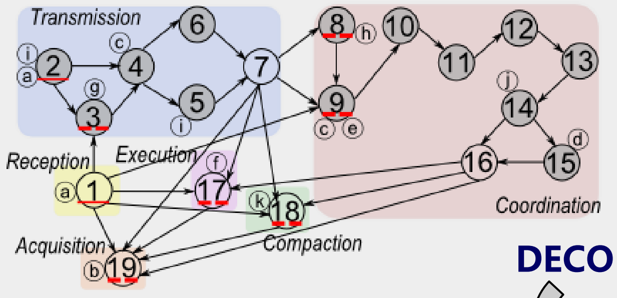
- M-flow can control the general trend on achieving the latency versus consistency trade-offs.



Consistency versus latency under different cross-DC bandwidths (the number of returned values versus the given read latency constraints):

- A larger latency bound for instantaneous reads following writes leads to a larger number of returned values.

The Malleable Flow (M-FLOW)



The decomposed steps and their execution orders form a directed graph, consisting of six partitions corresponding to the six stages.

DECOMPOSE

REFORM

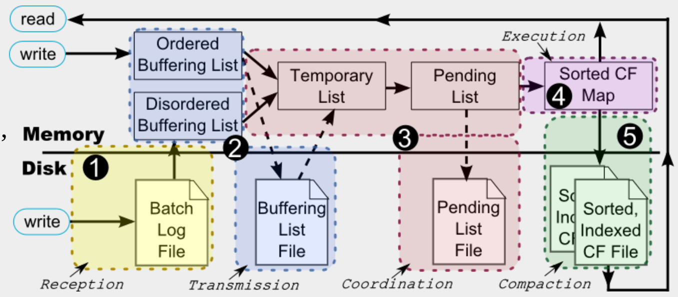
```
Algorithm : dCON Algorithm(finding the legal path with maximized consistency)
Input: $t_P$, $S_P$
Output: A path of nodes $S_P$
Initiate all $i$ s to 0;
if ⓕ then                                        // Executable writes
    Compute $S_P$'s path cost $C_P$;
    Estimate the number $m_e$ of executable writes within $t_P - C_P$;
    if $m_e > 0$ then
        Set $i_{12} \leftarrow$ MIN($m_e$,p-list write number);
        Add nodes 17 to $S_P$;                                 Execution
if ⓒ ∧ ⓔ ∧ ⓙ then                               // Coordinatable writes
    Compute $S_P$'s path cost $C_P$;
    Estimate the number $m_c$ of writes coordinatable and executable
    within $t_P - C_P$;
    if $m_c > 0$ then
        Set $i_7 \leftarrow$ MIN($\frac{1}{r}m_c$,b-list size+b-file write number);
        Set $i_{12} \leftarrow i_{12} + m_c$;
        if $i_7$ >b-list size then Add node 8 to $S_P$;
        Add nodes 9-14,16 to $S_P$;                            Coordination
if ⓒ ∧ ⓖ then                                   // Transmittable writes
    Compute $S_P$'s path cost $C_P$;
    Estimate the number $m_t$ of writes transmittable within $t_P - C_P$;
    if $m_t > 0$ then
        Set $i_4 = i_3 \leftarrow$ MIN($m_t$,bl-file write number);
        Add nodes 3-5, 7 to $S_P$;                             Transmission
if ⓚ then                                       // Compactable writes
    Compute $S_P$'s path cost $C_P$;
    Estimate the number $m_z$ of compactions within $t_P - C_P$;
    if $m_z > 0$ then
        Set $j_1, k_1$ based on $m_z$;
        Add nodes 18 to $S_P$;                                 Compaction
```

Principles for reformation:

- The path either covers or bypasses a stage
- The step subset is maximized with execution time bounded
- The set of writes processed within a step is maximized
  - ○ Computing from Stage execution, coordination, transmission, to Stage compaction
  - ○ Increasing executed and executable writes to improve the replica consistency status
- The path for a write covers Stage reception; that for a read covers Stage acquisition

Implementation over Cassandra

The storage architecture guarantees the durability of writes, and enables the execution flow to stop at the end of any stage.


read
write
Ordered Buffering List
Disordered Buffering List
Temporary List
Pending List
Execution
Sorted CF Map
4
Memory
Disk
1
2
3
5
write
Batch Log File
Buffering List File
Pending List File
Sorted, Indexed CF File
Reception
Transmission
Coordination
Compaction