Paxos made transparent

Heming Cui[+*], Rui Gu[*], Cheng Liu[*], Tianyu Chen[x], and Junfeng Yang[*]
[+]The University of Hong Kong [*]Columbia University [x]Tsinghua University

Paxos

- distributed consensus protocol
- used by SMR

SMR(State Machine Replication)

Models program as deterministic state machine

- states - important data
- transitions - execution on input

Runs replicas of program to ensure high availability

SMR(State Machine Replication)

Pros:

- great fault tolerance

Cons:

- assumes deterministic execution
- existing SMR systems provides too narrow interface

CRANE

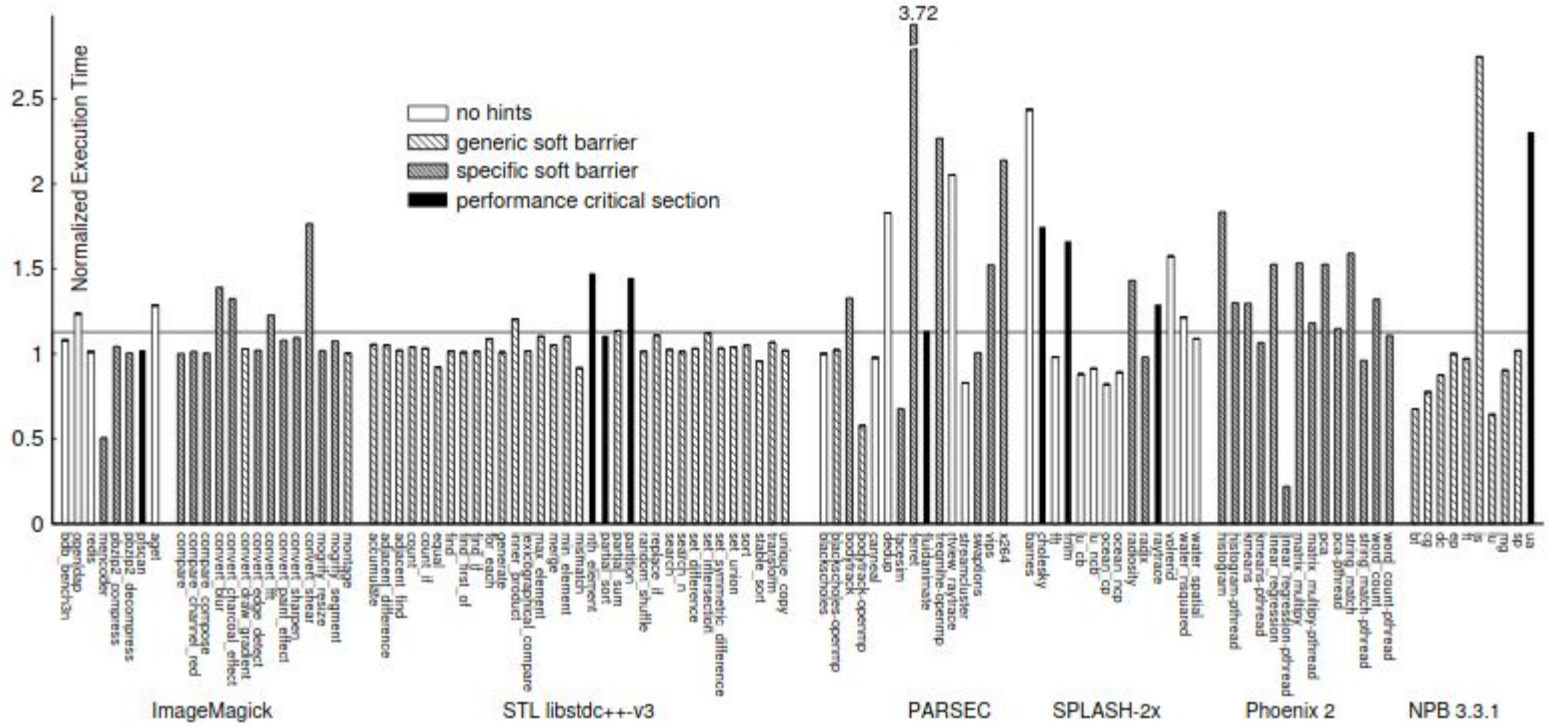
- SMR system that transparently replicates general server program
- deterministic multithreading (PARROT)
- time bubbling (network input timing)
- CRIU (checkpoint and restore process state)
- LXC (file system state)

developers focus on developing system,

not its replication

DMT - deterministic multithreading

- logical time
- CRANE uses PARROT
- PARROT is very efficient, average overhead is 12.7%


Normalized Execution Time
2.5
2
1.5
1
0.5
0
3.72
no hints
generic soft barrier
specific soft barrier
performance critical section
ImageMagick
STL libstdc++-v3
PARSEC
SPLASH-2x
Phoenix 2
NPB 3.3.1

Time of arrival of request

Physical time of arrival may differ for each replica leading to different execution sequence.

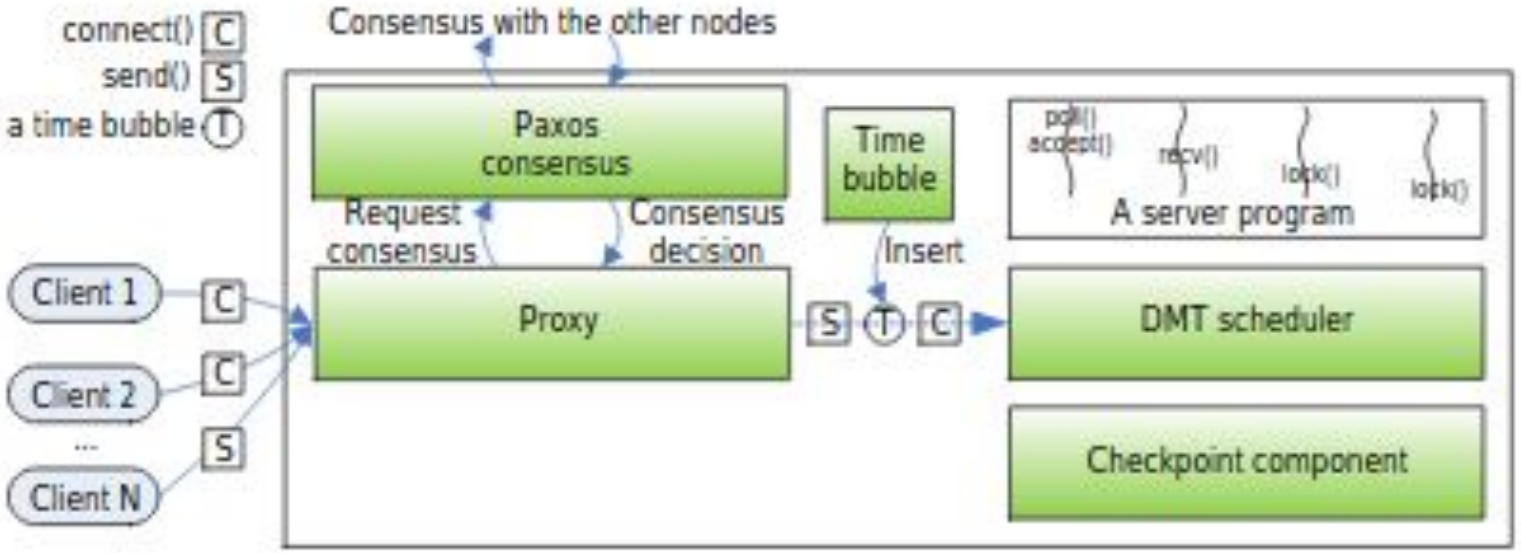
- many request don’t need admision time, because it is already deterministic
- practice: request arrive in burst

Time bubbling

- ensures that first request in burst is admitted deterministically at each replica
- enforces deterministic logical time

CRANE overview

- One replica is leader, rest are backup.
- Leader proposes order of request.
- If the leader fails, replicas run leader election.
- clients sends requests to leader and get responses.



CRANE architecture (CRANE components are in green)

Implementation

- interpose POSIX sockets and pthread synchronization by hijacking dynamically linked library.
- implements PAXOS protocol atop libevent.
- uses CRIU to checkpoint and restore process states and LXC for file system state

- CRANE enforces that replicas see the same order of clients socket calls
- CRANE enforces that poll, accept, recv are scheduled and returned in same sequence of logical time
- send are scheduled by DMT and does not need consensus

Proxy component

- on client socket request proxy invokes consensus
- forwards responses from the server
- on backup replica does not accept connection from clients and does not invoke consensus

PAXOS component

- receives client socket request from proxy
- only component that communicates with other replicas
- after component reaches consensus, each PAXOS component notifies its own proxy

DMT component

- PARROT as DMT scheduler
- intercept pthread synchronization and enforces round-robin
- PARROT does not resolve data races. CRANE tolerates data races that have fail-stop consequences.

Time bubbling component

Activate in two cases:

1. on a server's bootstrap to make sure that server programs reches same initial state across replicas and wait for input
2. on timeout if server has not received any request

For each time bubble DMT scheduler promises to run N_{clock} synchronization

Checkpoint component

- checkpoint component is invoked every minute
- CRIU saves process registers and memory
- LXC backup working directory
- proxy and DMT component does not need checkpoint

CRANE synchronization wrappers PARROT pthread wrapper

```
1: int pthread_mutex_lock_wrapper(mu) {
2:     DMT.get_turn();
3:     check_add_timebubble(); // NOP in Parrot. Called in Crane.
4:     while (pthread_mutex_trylock(mu))
5:         DMT.wait(mu);
6:     DMT.put_turn();
7:     return 0; // Error handling code ommitted for clarity.
8: }
```

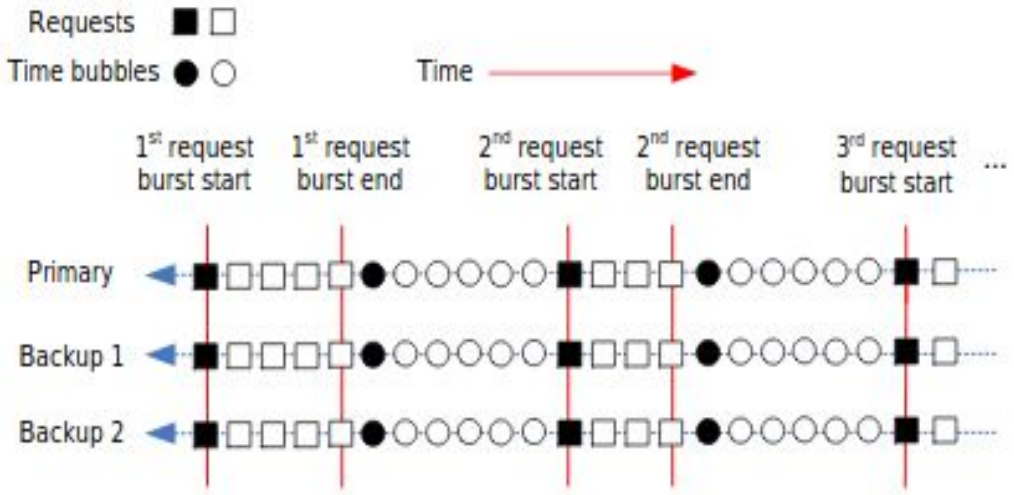
CRANE check_add_timebubble

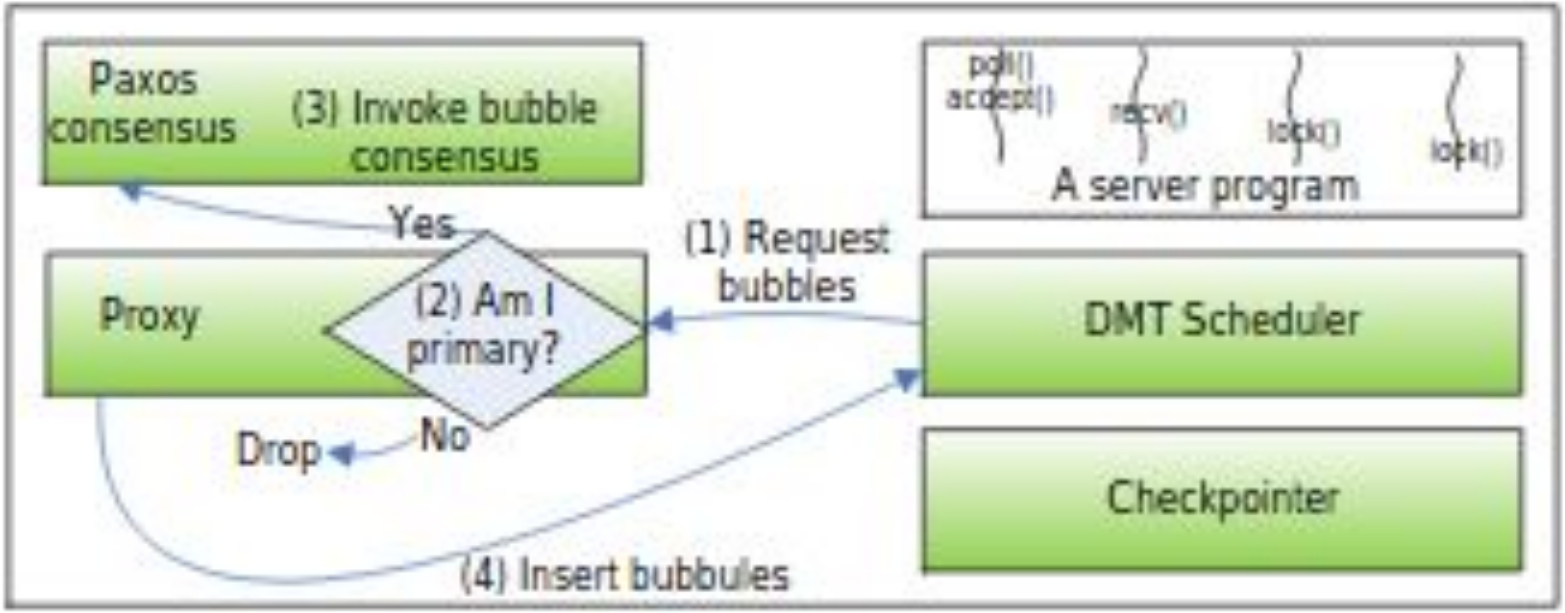
```
1:  void check_add_timebubble(mu) {
2:    while (paxos_seq.empty()) {
3:      usleep(...);
4:      request_time_bubble();
5:    }
6:    if (paxos_seq.head().type == TIME_BUBBLE)
7:      paxos_seq.head().decrease();
8:    else
9:      DMT.signal(paxos_seq.head());
10: }
```

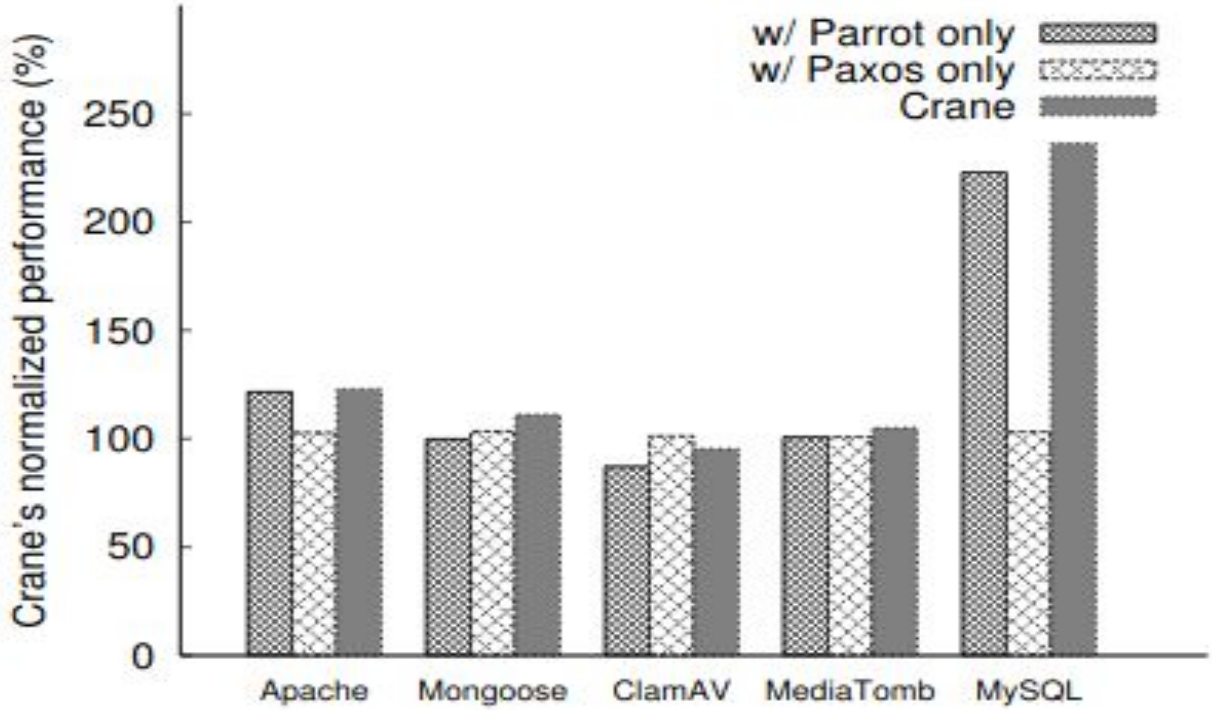
CRANE wrapper for recv

```
1: int recv_wrapper(sockfd, ...) {
2:     DMT.get_turn();
3:     DMT.wait(sockfd);
4:     int nbytes = recv(sockfd, ...);
5:     paxos_seq.dequeue(nbytes);
6:     DMT.put_turn();
7:     return nbytes;
8: }
```

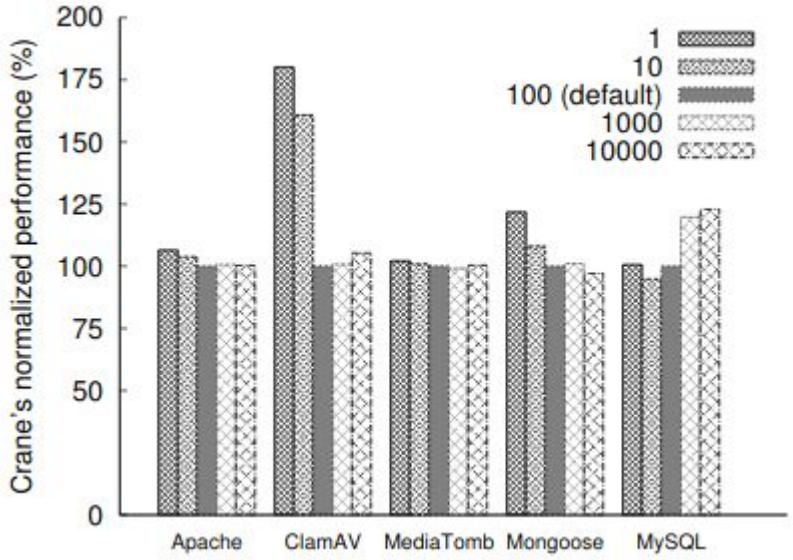
Time bubbling



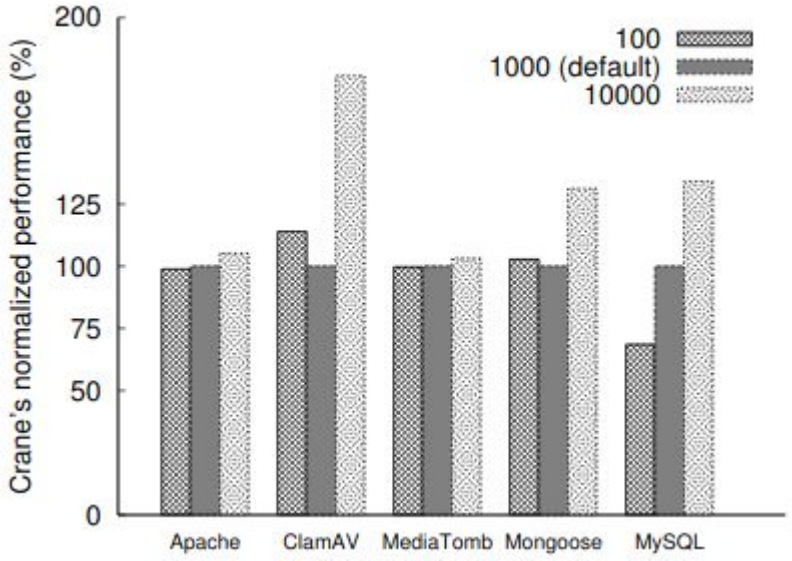

Paxos consensus
(3) Invoke bubble consensus
Yes
Proxy
(2) Am I primary?
No
Drop
(1) Request bubbles
poll()
accept()
recv()
lock()
lock()
A server program
DMT Scheduler
Checkpointer
(4) Insert bubbules


w/ Parrot only
w/ Paxos only
Crane
Crane's normalized performance (%)
250
200
150
100
50
0
Apache
Mongoose
ClamAV
MediaTomb
MySQL

CRANEs performance on different $W_{timeout}$



CRANE's performance on different N_{clock}



Program	# client socket calls	# time bubbles	%
Apache	3,000	450	13.04
ClamAV	18,000	1,173	6.12
MediaTomb	3,000	1,501	33.35
Mongoose	3,000	448	12.99
MySQL	6,750	573	7.82

Program	C p (ms)	R p (ms)	C fs (ms)	R fs (ms)
Apache	33	48	3,069	237
ClamAV	415	353	6,963	6,128
MediaTomb	17	27	2,852	213
Mongoose	15	31	1,294	169
MySQL	88	81	53,473	712

Table 2: ***Average time cost for*** **CRANE** ***'s checkpoint and restoring component.*** **"C p" means "Checkpoint process", "R p" means "Restore process", "C fs" means "Checkpoint file system", and "R fs" means "Restore file system".**

https://github.com/columbia/crane