Laboratory for Advanced Systems Software

ZZ and the Art of Practical Byzantine Fault Tolerance

Timothy Wood, Rahul Singh, Arun Venkataramani, Prashant Shenoy, and Emmanuel Cecchet

University of Massachusetts Amherst

Data Center Fault Tolerance

- Data Centers
  - Run many critical applications
  - Virtualization
- Many potential causes of failure
  - Hardware crashes
  - Software bugs
  - Malicious attacks
- High availability
  - protect against crashes
  - Need **f+1** replicas to tolerate **f** node failures
- **Byzantine Fault Tolerance**
  - Protects against arbitrary malicious behavior
  - Need **3f+1** replicas to tolerate **f** faulty nodes
  - Considered too expensive to actually use…

BYZANTINE FAULT TOLERANCE

- Protocol to allow a collection of replicas to act like a single correct one, even if there are malicious faults

- Fault Model
  - Replicas can be arbitrarily malicious
  - Cannot subvert cryptographic messages
  - Only **f** replicas can be faulty at any one time

- Must guarantee
  - Safety: any response given to a client is correct
  - Liveness: any request will eventually get a response

PROTECTING N APPLICATIONS, F=1

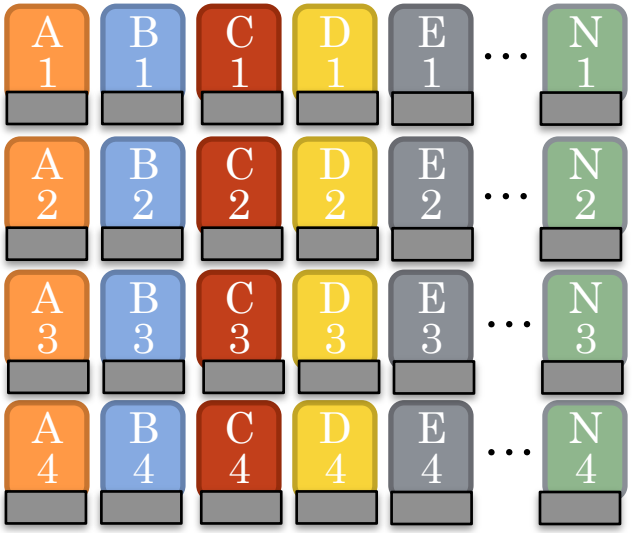
No Protection

N servers

Crash Fault Tolerance

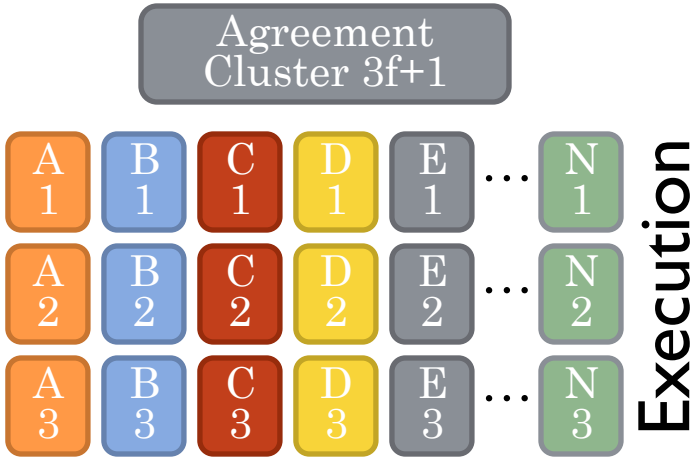
N(f+1)

Byzantine Fault Tolerance



N(3f+1)

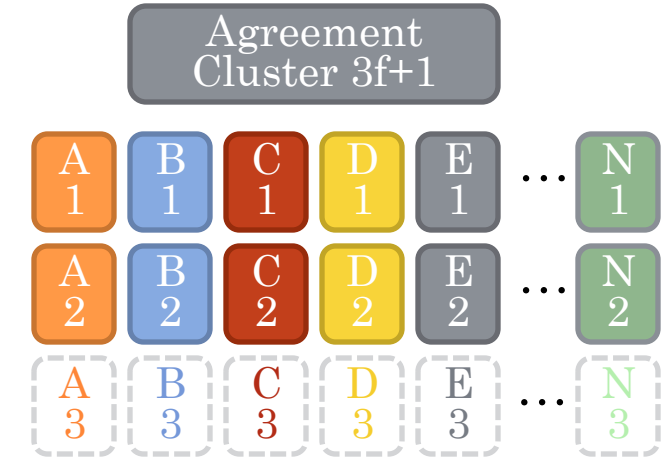
Separate Agree/Execution



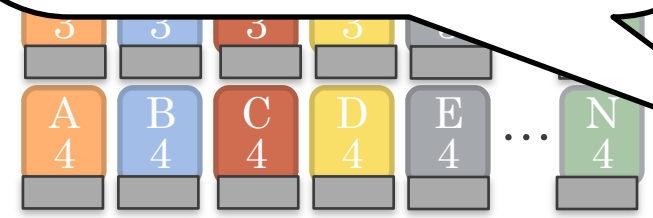
N(2f+1)

PROTECTING N APPLICATIONS, F=1

Can we reduce the cost of BFT even further?

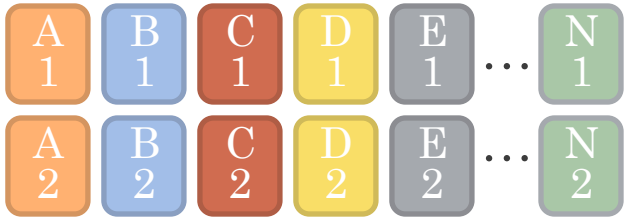


~N(f+1)



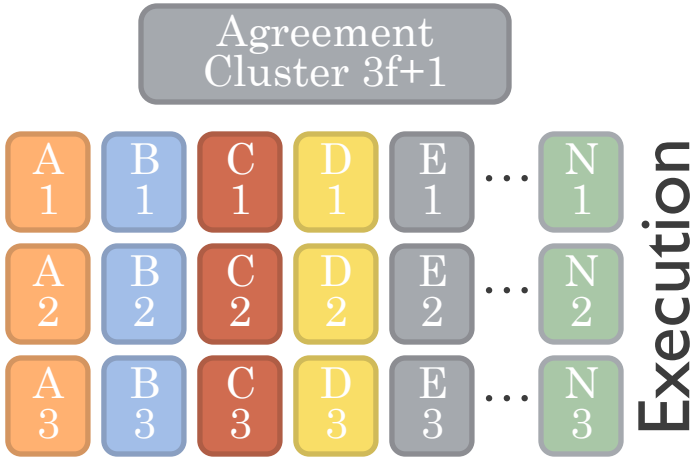
N(3f+1)

Crash Fault Tolerance



N(f+1)

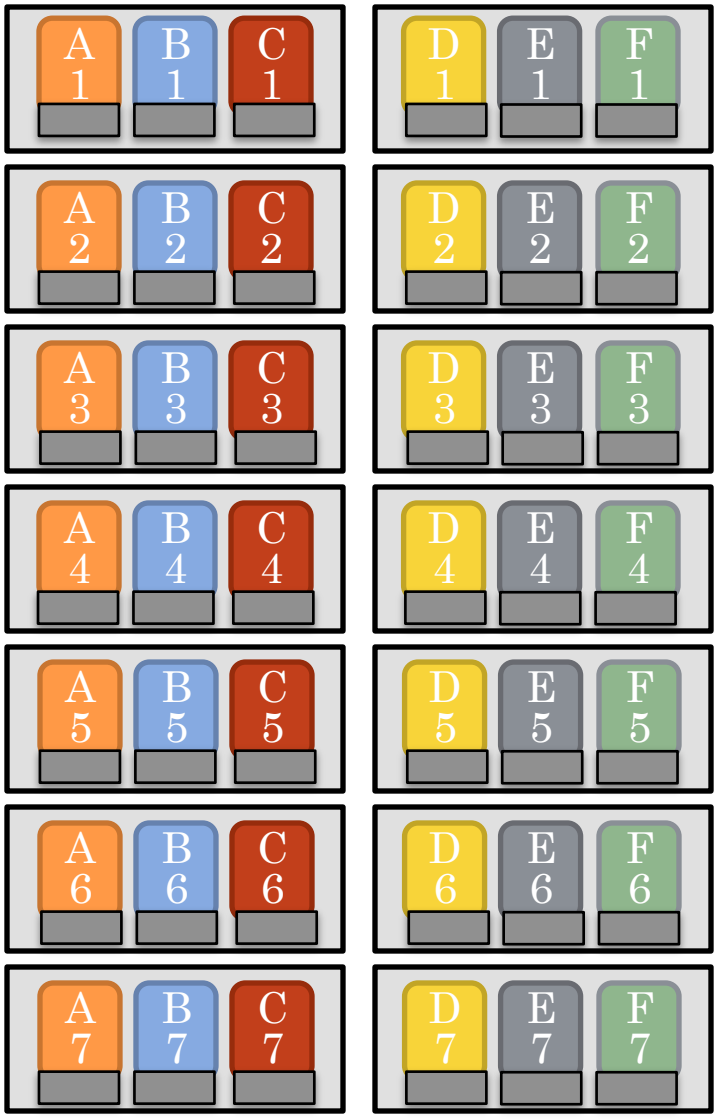
Separate Agree/Execution



N(2f+1)

SCALING TO F=2

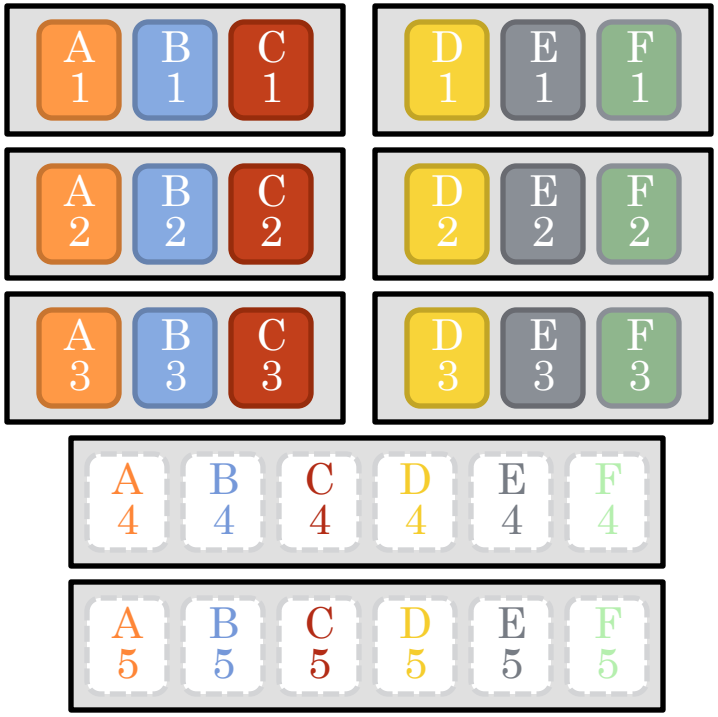
Traditional BFT



N(3f+1) = 7N

ZZ

Cras Agreement Cluster 3f+1 rance

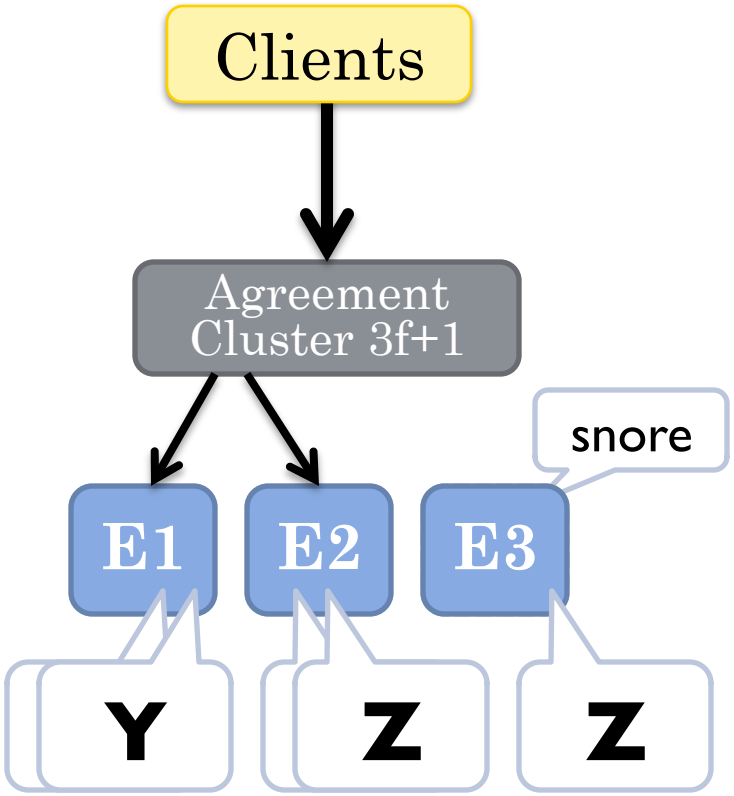


N(f+1) active = 3N

N(f) asleep = 2N

ZZ: F+1 BFT EXECUTION

- **Fault-free case:** f+1 exec replicas can make progress
  - Reduces resource consumption in data center
- **Fault detection:** if execution replicas disagree
- **Wake up:** spawn new execution replicas
  - Obtain application state and replay requests
- **Shutdown:** eliminate faulty replicas
  - Must reduce cost back to f+1

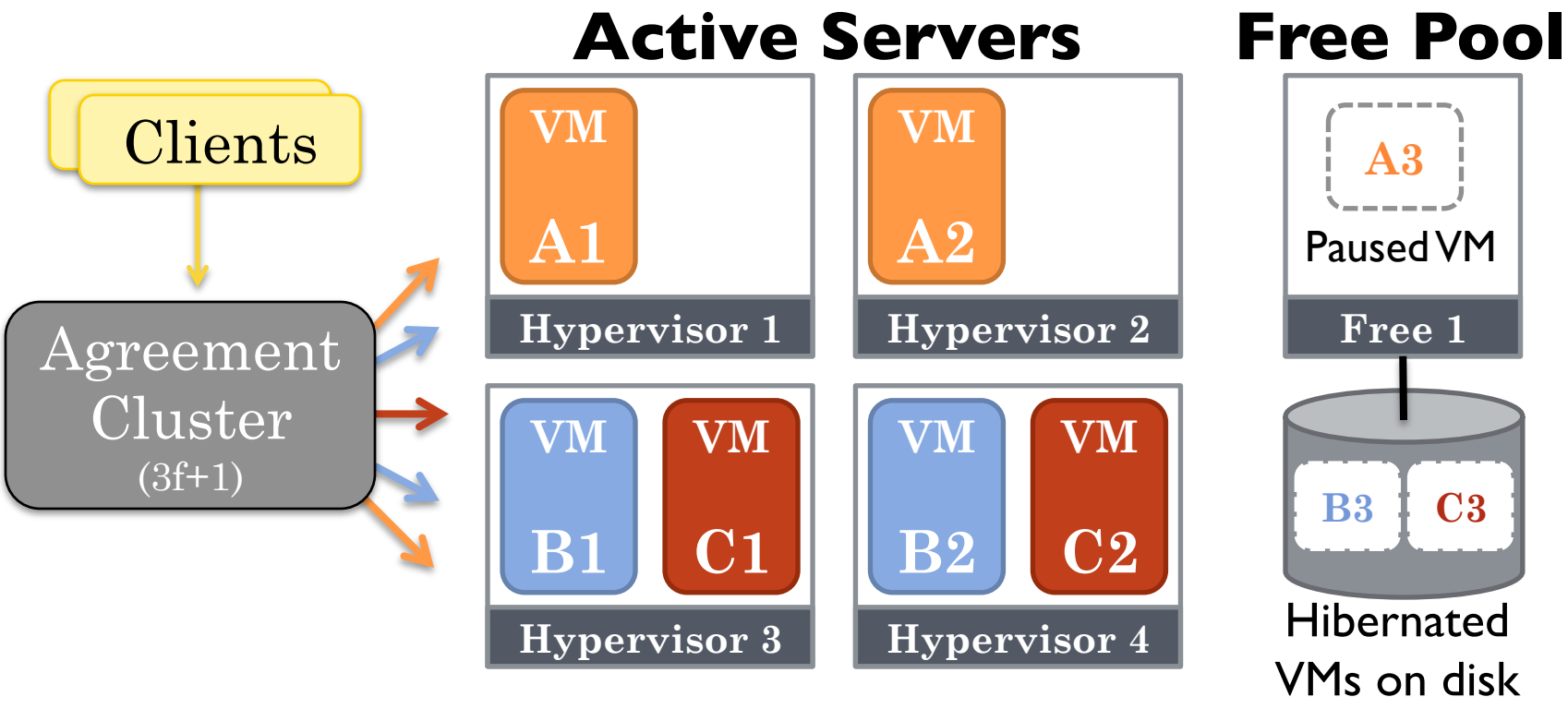


Outline

- Introduction
- **ZZ Design**
- Evaluation
- Conclusions

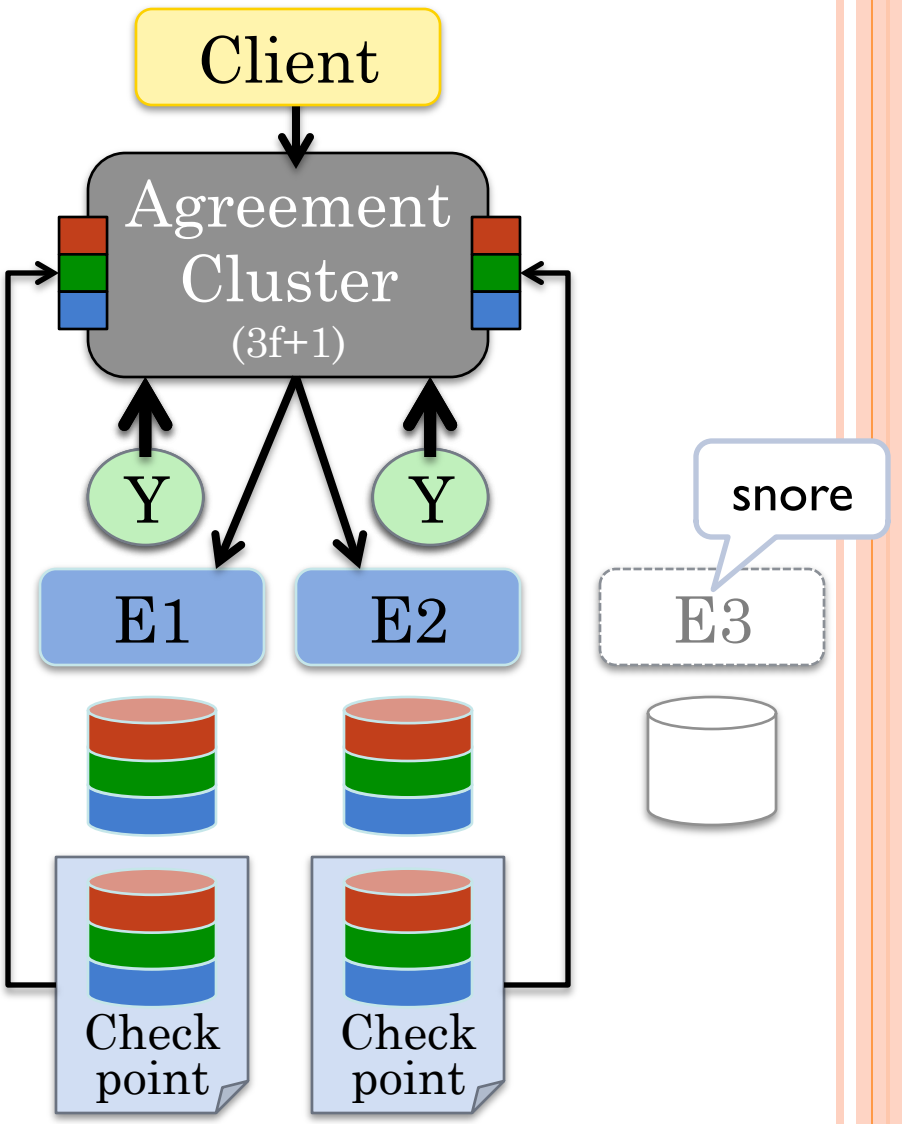
ZZ BFT Data Center

- Host multiple BFT applications in data center
- Each server runs multiple BFT virtual machines
  - Replicas for the same application cannot be colocated
- Spare replicas can be paused or hibernated



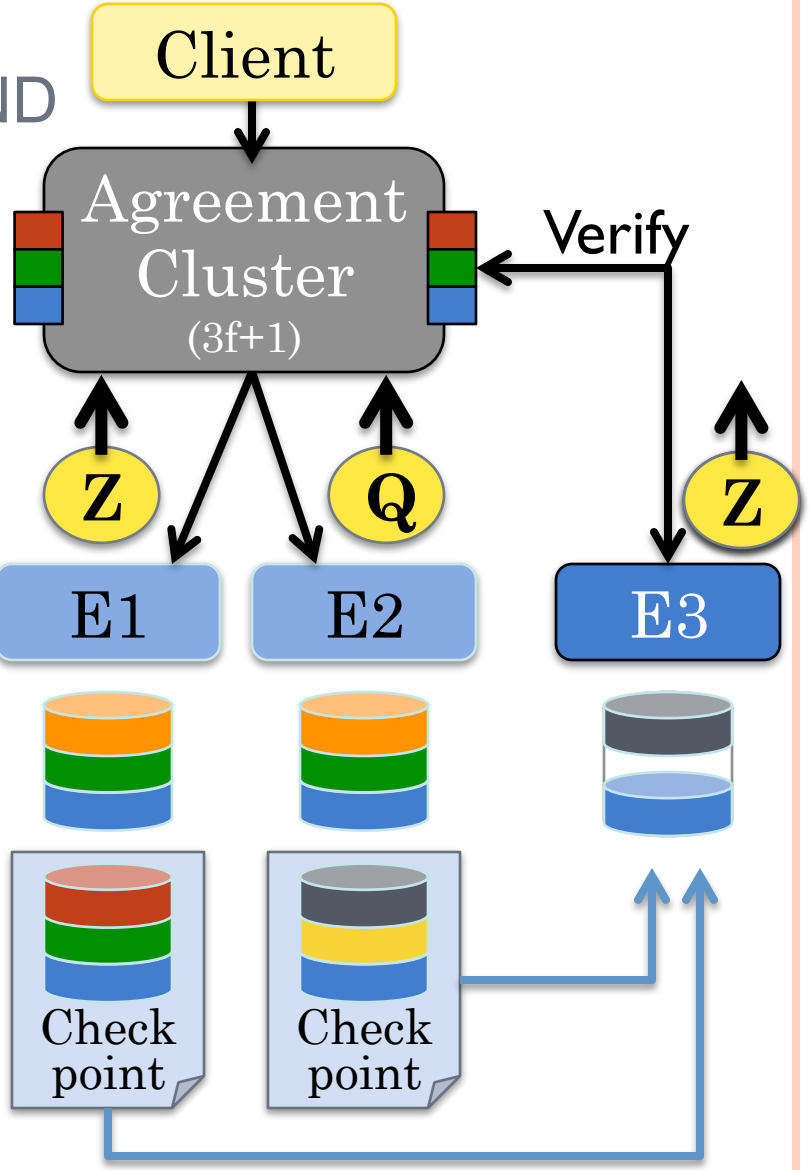
Graceful Execution

- Periodically make checkpoints
  - Agreement cluster must verify consistency
- Continue processing requests as long as responses agree
- Extra replica sleeps and has no application state



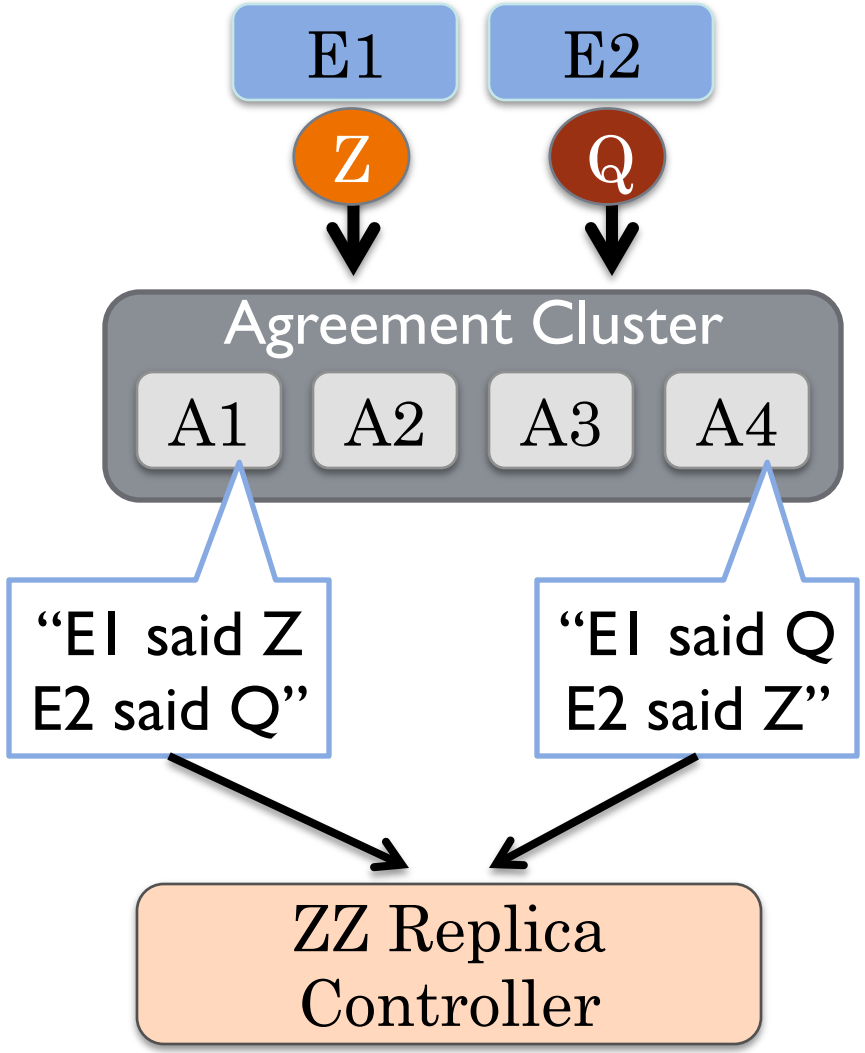
Recover State on Demand

- Fault detected if replicas disagree
- New replicas need state to start processing requests
  - Can't trust execution replicas!
  - Transferring full state too slow!
- Attempt to begin replaying requests from last checkpoint
- Obtain state as needed for each request
- Use checkpoint digests from agreement cluster for state verification



Preventing Spurious Wakeups

- Starting new replicas is expensive
  - Causes downtime and increases replication cost
- Only have second hand information
  - Using MACs not signatures
  - Agreement nodes can be faulty too
- Some faults do not require a wakeup to make progress



Blocking & Non-Blocking Faults

- Only wake up for faults that prevent ZZ from making progress
- ZZ Replica Controller must decide

Non-Blocking

	A1	A2	A3	A4
E1	Q	Q	Q	Q
E2	P	Q	Q	Q
E3	Not needed			

Response of E2 as reported by A3

Blocking

	A1	A2	A3	A4
E1	P	Q	Q	Q
E2	P	P	P	Q
E3	P	P	P	Q

Need E3 to help diagnose fault

Assigning Blame

- Must reduce active replicas back to f+1
- **Non-Blocking Faults**
  - f+1 agreement nodes match
  - Do not have enough information to "convict"
- **Blocking Faults**
  - Cannot make progress
  - Usually* have enough information to convict immediately
- **Theorem:** if a wake up occurs, ZZ will shut down at least one faulty replica

	A1	A2	A3	A4
E1	Q	Q	Q	Q
E2	P	Q	Q	Q
E3	Not needed			

Non-Blocking

	A1	A2	A3	A4
E1	P	Q	Q	Q
E2	P	P	P	Q
E3	P	P	P	Q

Blocking

Outline

- Introduction
- ZZ Design
- **Evaluation**
- Conclusions

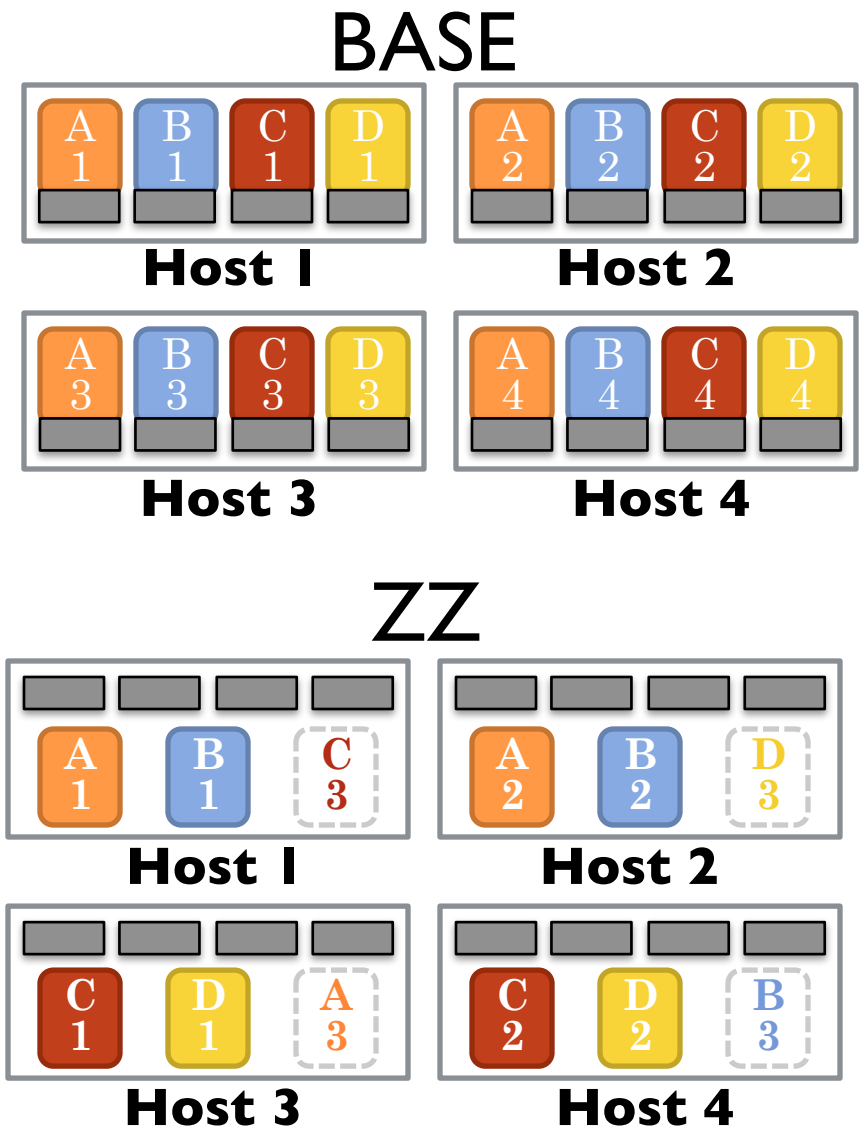
IMPLEMENTATION

- Modification of the BASE software library
  - Separated agreement and execution
  - Added fault detection
  - Optimized replica recovery
  - Ideas could be applied to other agreement protocols
- Xen virtualization platform
  - Paused / hibernated VMs used for sleeping replicas
- Simplified checkpoint creation with ZFS
  - Can use file system level snapshots instead of making application modifications

Evaluation: BFT Data Center Setup

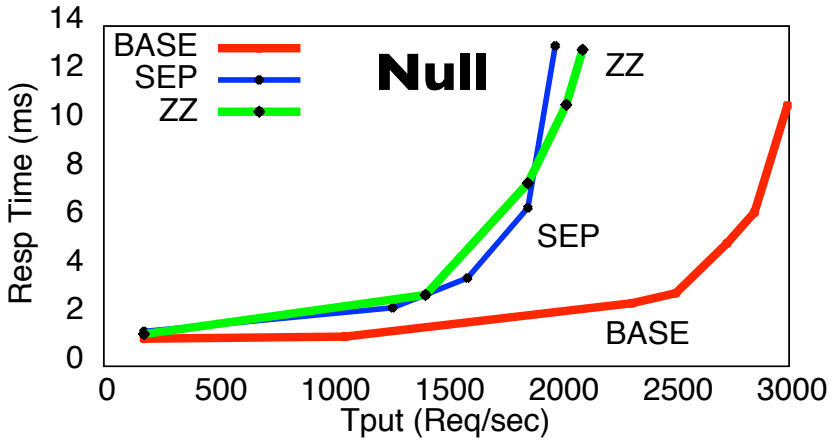
- Four host machines
- Four web apps (A-D)
- f=1

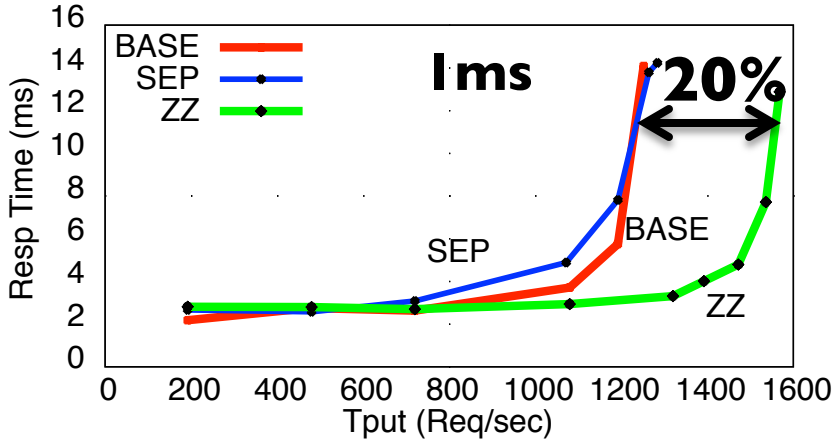
- BASE
  - 4 replicas per server
  - Each performs agreement and execution
  - Masks faults
- ZZ
  - 4 agreement replicas per server
  - 2 execution replicas
  - 1 sleeping standby replica



EVALUATION: EXECUTION COST MATTERS

- Adjust per request execution cost
- ZZ has higher throughput when resources constrained

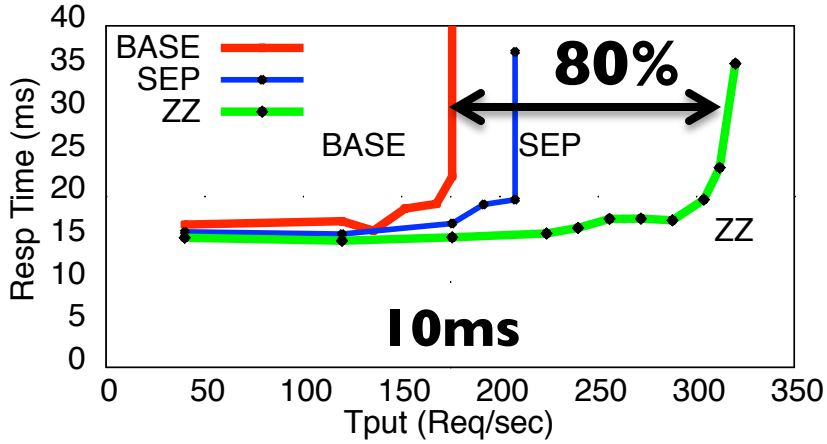




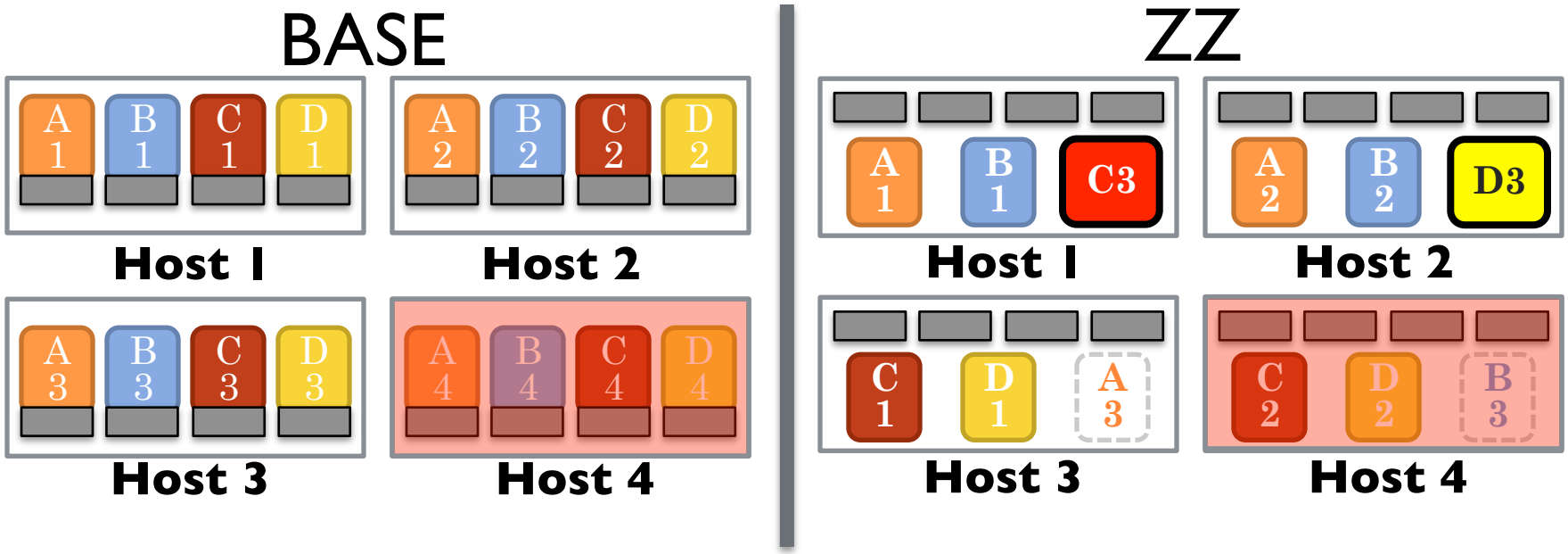
BASE: 3f+1

SEP: 2f+1

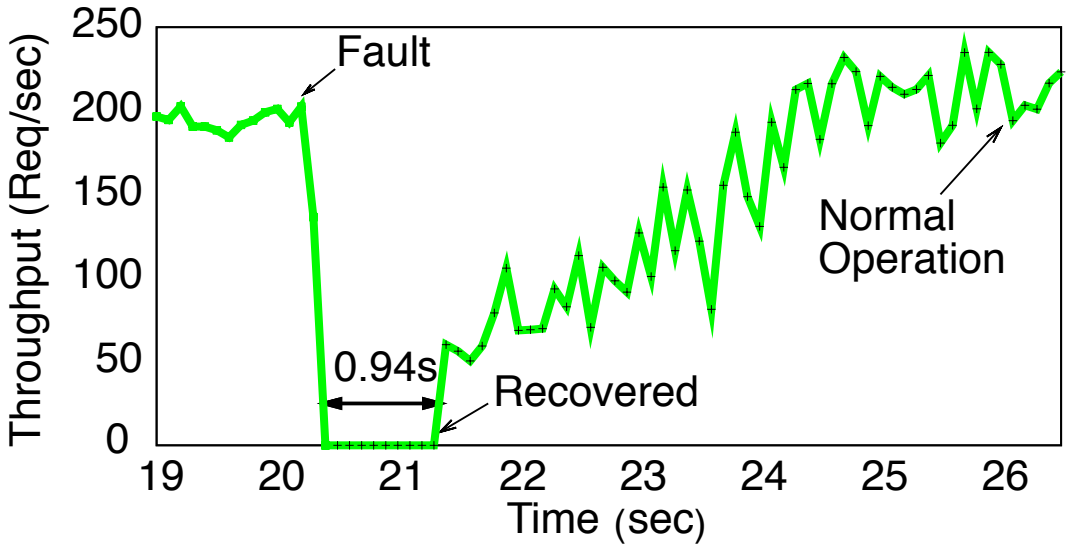
ZZ: f+1



EVALUATION: MULTIPLE FAILURES



EVALUATION: RECOVERY COST IN BFT NFS



- Downtime depends on amount of application state modified since last checkpoint
- Performance on subsequent requests is variable as state is obtained on demand

Related Work

- Byzantine Agreement – performance and robustness
  - Zyzzyva 07, Aardvark 09, Aliph 10
- Reducing Execution cost
  - Cheap Paxos 04 – low cost crash tolerance
  - SPARE 11 and ODRC 11 – reduce HW cost or improve performance
- ZZ
  - Could be used with optimized agreement protocol
  - Recovery time could be further reduced with proactive recovery

CONCLUSIONS

- Execution cost dominates agreement for real applications
- ZZ runs **f+1** active and **f** sleeping replicas
  - Saves resources when running multiple BFT applications
  - Improves performance if resources are constrained
- ZZ optimizes recovery after failure
  - Obtains state on demand
  - Only responds to blocking faults
- Additional details in paper
  - Overall replication cost
  - Response time inflation