Stony Brook University

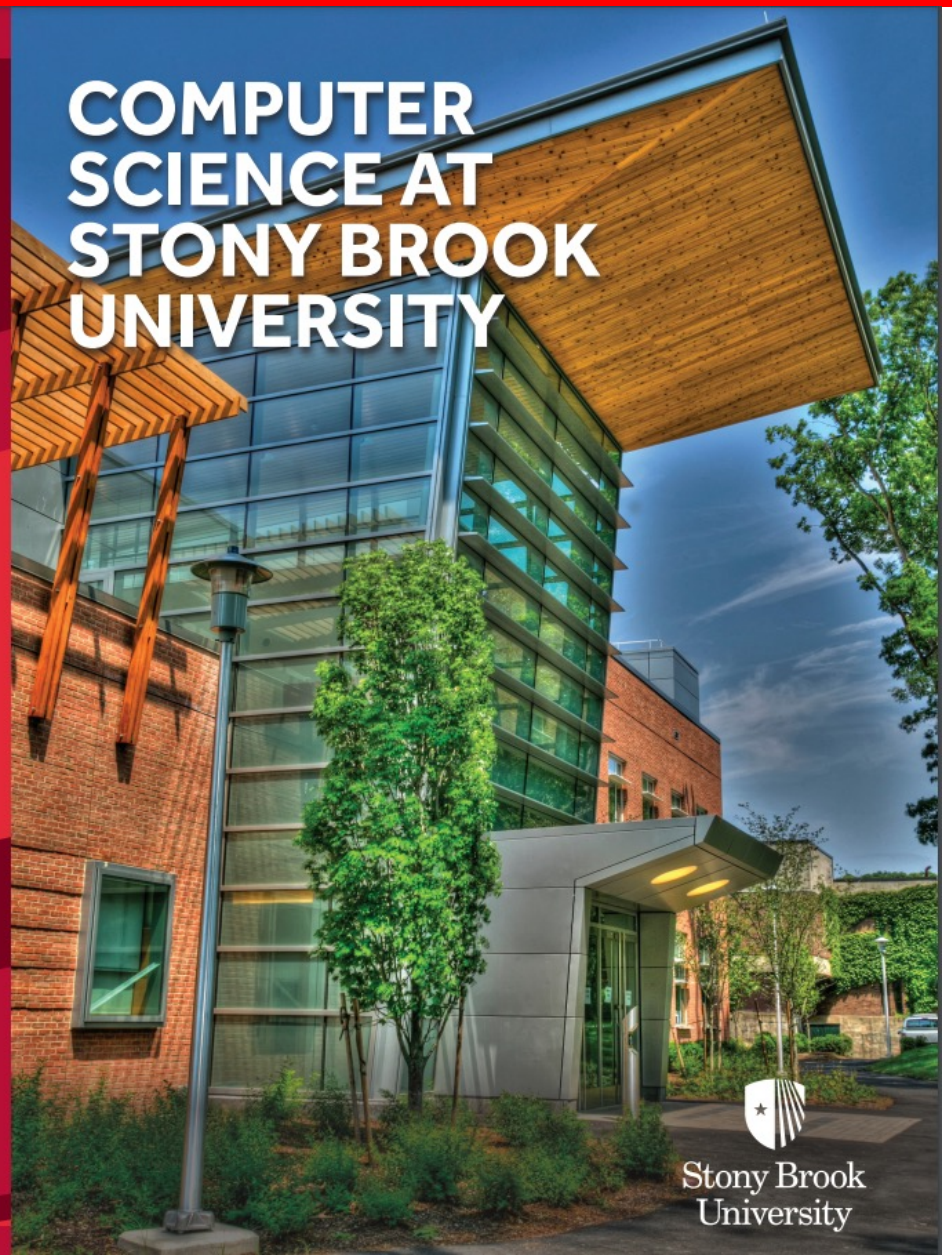
Towards Adaptive Transaction Processing in Untrusted Environments

Mohammad Javad Amiri

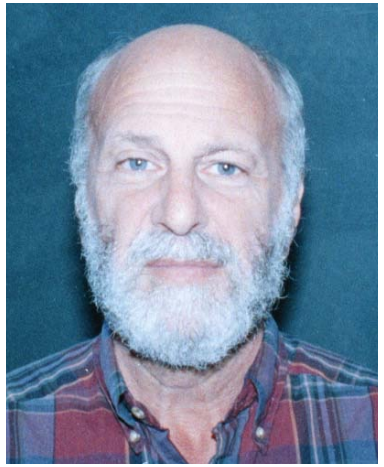
Divy Agrawal, Amr El Abbadi, Boon Thau Loo, Dahlia Malkhi, Ryan Marcus, Mo Sadoghi, Chenyuan Wu, Haoyun Qin, Bhavana Mehta

FAR BEYOND

MAKE STONY BROOK DB GREAT AGAIN

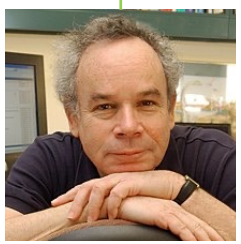


Phil Lewis

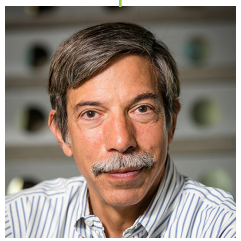


Art Bernstein

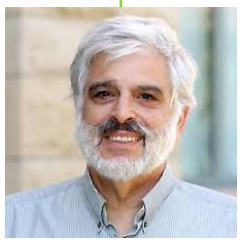
- Databases and Transaction Processing: An Application-Oriented Approach
- Concurrency in programming and database systems



Avi Silberschatz



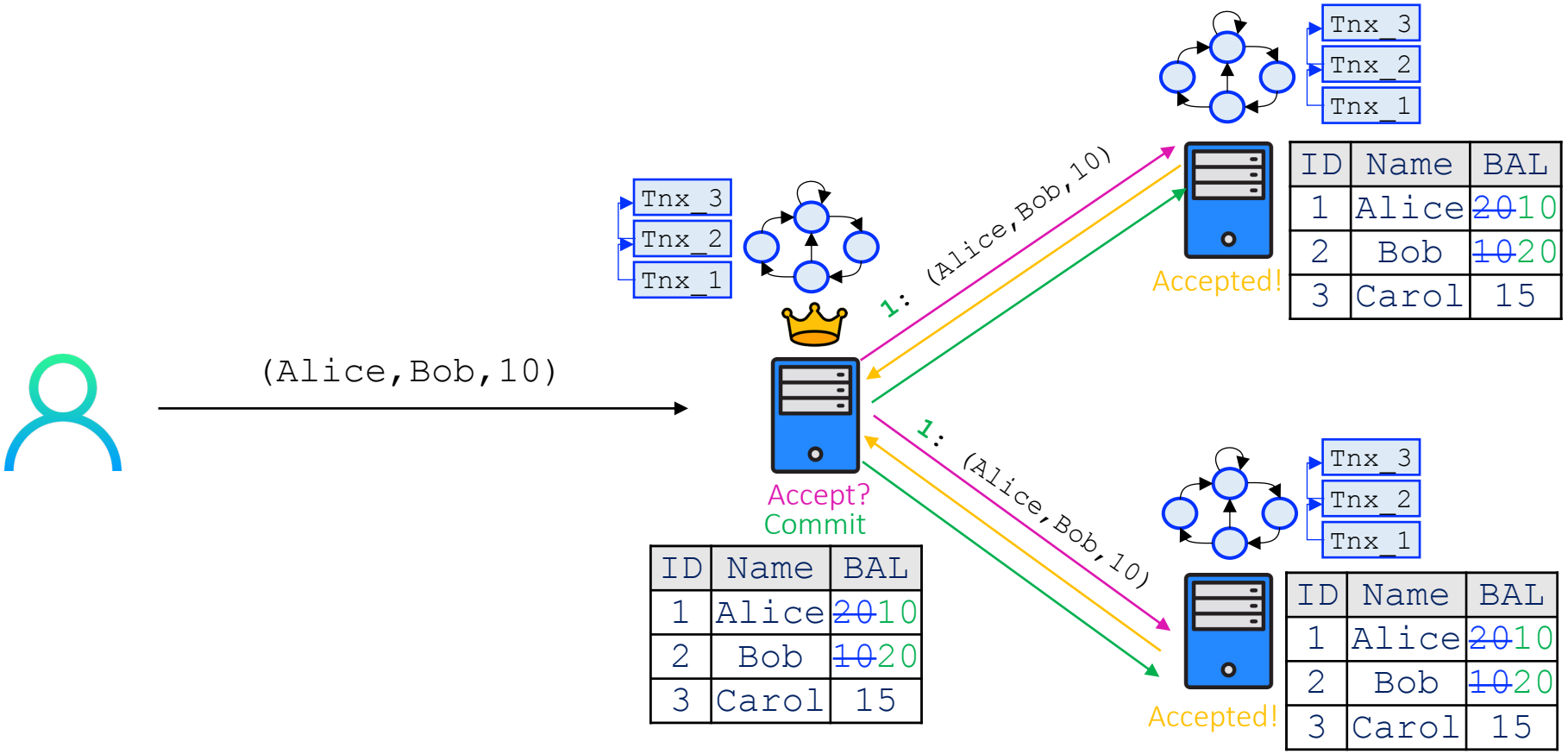
Fred Schneider



Jeffrey Ullman

Divy Agrawal

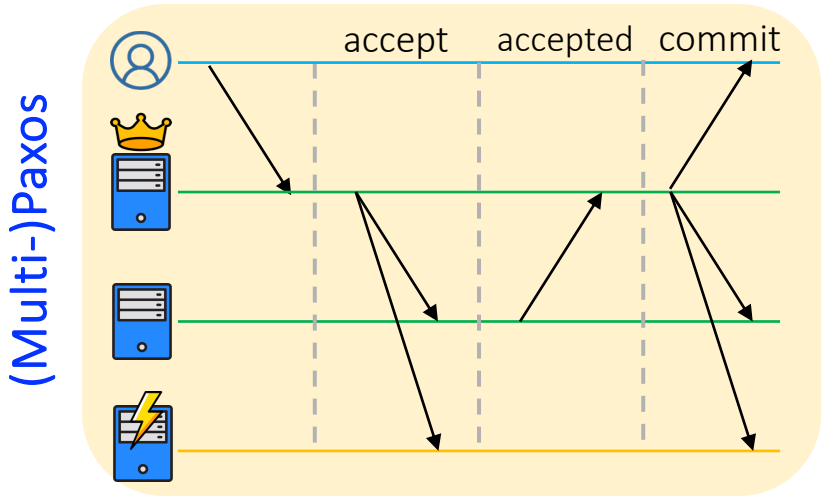
Fault-tolerant transaction processing



State Machine Replication: a replicated service whose state is mirrored across different deterministic replicas

- Assign order to each client request in the global service history and execute it in that order

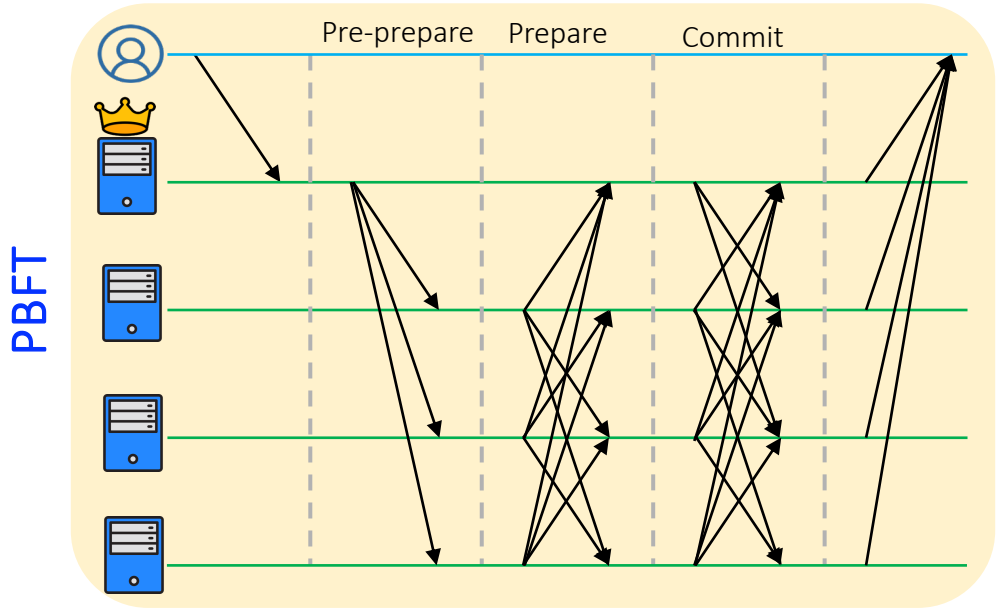
Crash fault-tolerant protocol: (Multi-)Paxos



- Requires 2f+1 nodes to be able to tolerate f failures
- How to deal with Byzantine failure?
  - nodes exhibit arbitrary, potentially malicious, behavior
  - Potential causes: software bugs, hardware failures, malicious attacks

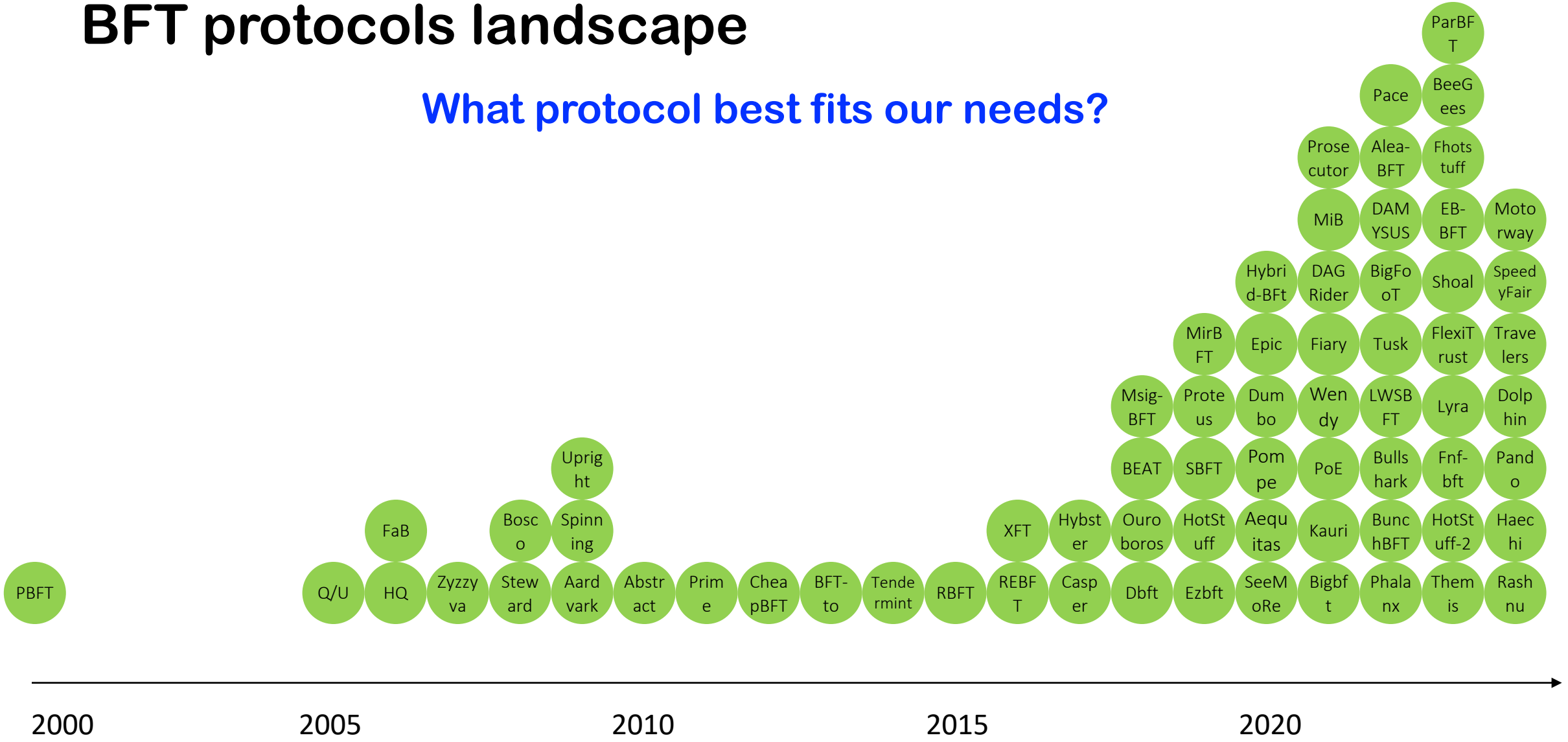
Byzantine fault-tolerant protocol: PBFT

- Nodes can fail arbitrarily, including deviating from the protocol
- Require 3f+1 nodes to tolerate f concurrent failures
- E.g., PBFT



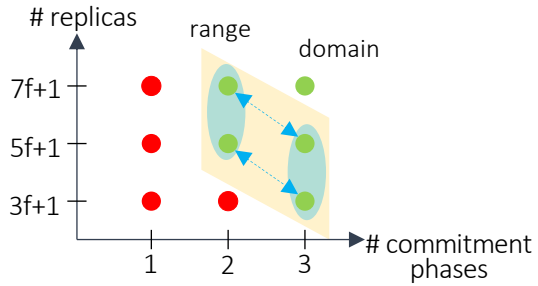
BFT protocols landscape

What protocol best fits our needs?



BFT protocols design space and design dimensions

- Design space
  - A set of dimensions to analyze BFT protocols
- Design choices
  - Trade-offs between dimensions
  - A set of one-to-one functions, each maps protocols in its domain to protocols in its range
- Focus on partially synchronous BFT protocols



Amiri, M. J., Wu, C., Agrawal, D., El Abbadi, A., Loo, B. T., & Sadoghi, M. The Bedrock of Byzantine Fault Tolerance: A Unified Platform for BFT Protocol Analysis, Implementation and Experimentation, NSDI'24 **[Outstanding Paper Award]**

Design space of BFT protocols

Protocol structure

P1. Commitment strategy
P2. Number of commitment phases
P3. View-change
P4. Checkpointing
P5. Recovery
P6. Types of clients

Environmental Settings

E1. Number of replicas
E2. Communication topology
E3. Authentication
E4. Responsiveness, synchronization, and timers

Quality of Service

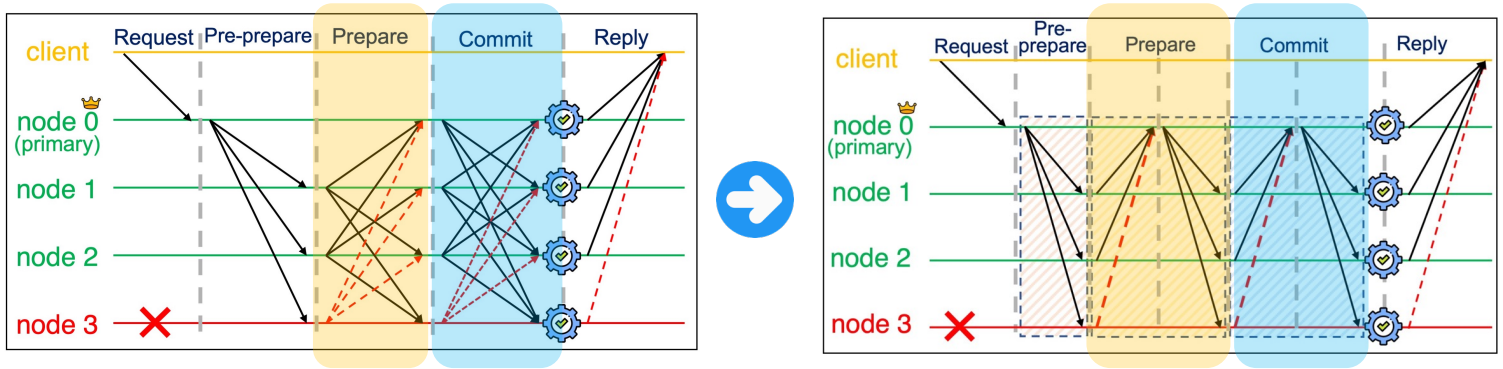
Q1. Order-fairness
Q2. Load balancing

Performance Optimization

O1. Out-of-order processing
O2. Request pipelining
O3. Parallel ordering
O4. Parallel execution
O5. Read-only requests processing
O6. Separating ordering and execution
O7. Trusted hardware
O8. Request/reply dissemination

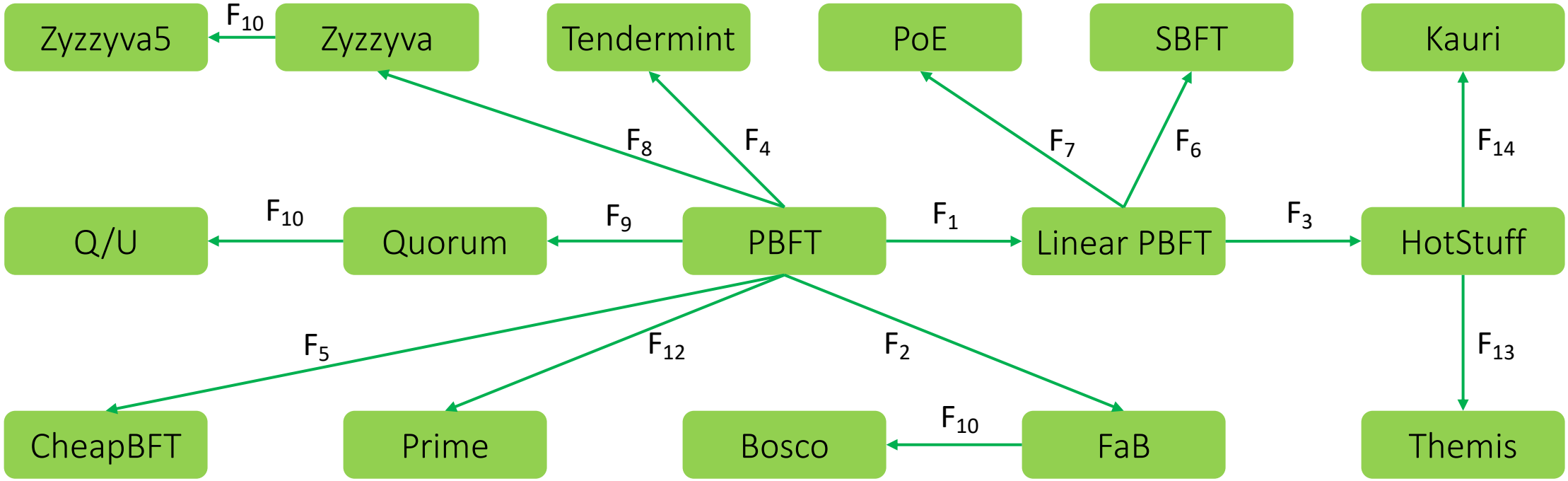
Design choices

1. Linearization
2. Phase reduction through redundancy
3. Leader rotation
4. Non-responsive leader rotation
5. Optimistic replica reduction
6. Optimistic phase reduction
7. Speculative phase reduction
8. Speculative execution
9. Optimistic conflict-free
10. Resilience
11. Authentication
12. Robust
13. Fair
14. Tree-based LoadBalancer



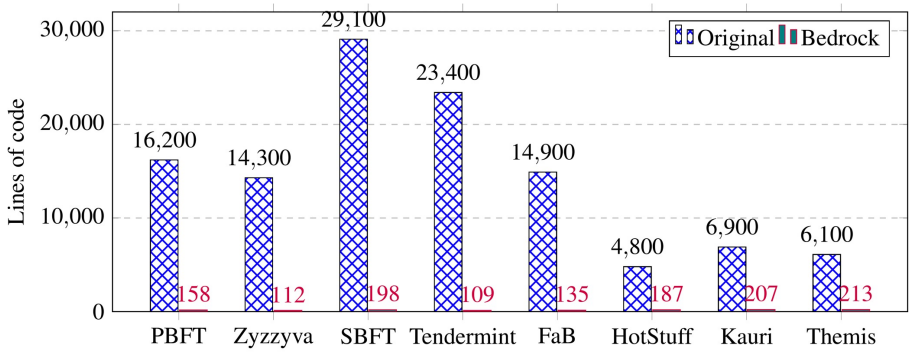
Derivation of protocols from PBFT using design choices

1. Linearization
2. Phase reduction through redundancy
3. Leader rotation
4. Non-responsive leader rotation
5. Optimistic replica reduction
6. Optimistic phase reduction
7. Speculative phase reduction
8. Speculative execution
9. Optimistic conflict-free
10. Resilience
11. Authentication
12. Robust
13. Fair
14. Tree-based LoadBalancer

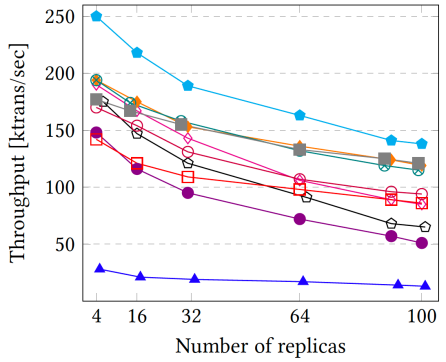


Bedrock platform

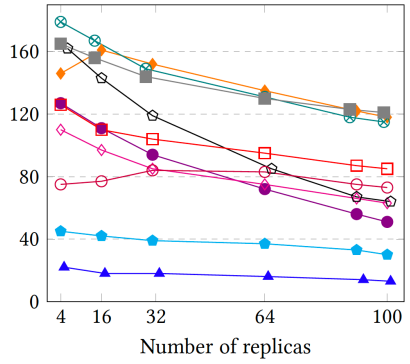
- Rapid prototyping of BFT protocols
  - Using a domain-specific language



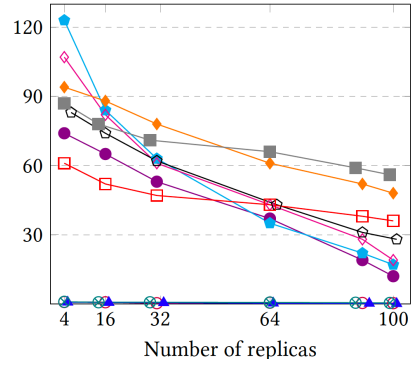
- Fair and Efficient experimental evaluation of BFT protocols



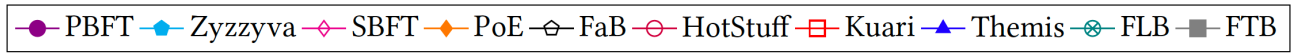
No faulty replica



Single faulty backup



Geo-distributed setup



Stony Brook University

```
protocol:
  general:
    leader: stable
    requestTarget: primary

  roles:
    - primary
    - nodes
    - client

  phases:
    - name: normal
      states:
        - idle
        - wait_prepare
        - wait_commit
        - executed
      messages:
        - name: request
          requestBlock: true
        - name: reply
          requestBlock: true
        - name: preprepare
          requestBlock: true
        - prepare
        - commit
    - name: view_change
      states:
        - wait_view_change
        - wait_new_view
      messages:
        - view_change
        - new_view
    - name: checkpoint
      messages:
        - checkpoint

  transitions:
    from:
      - role: client
        state: idle
        to:
          - state: executed
            update: sequence
            condition:
              type: msg
              message: reply
              quorum: 2f + 1
```

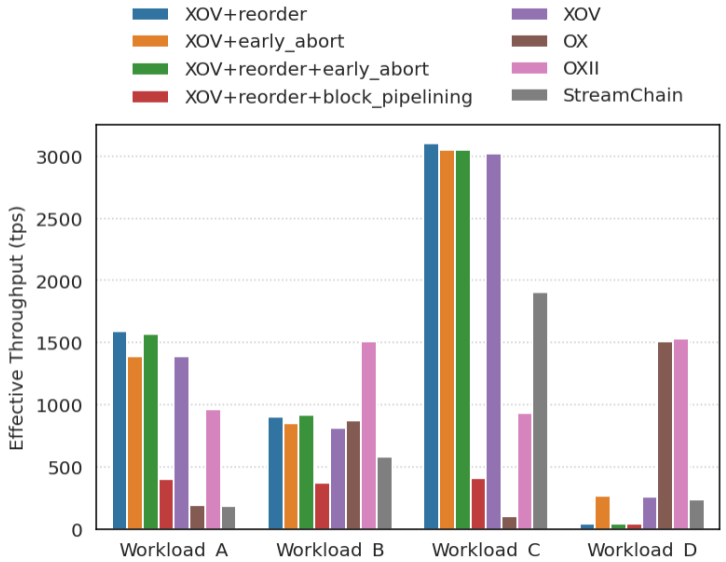
Beyond consensus protocols

- Transaction processing: ordering and execution
  - Concurrency control mechanism
  - Transaction reordering algorithms
  - Block size adaptation

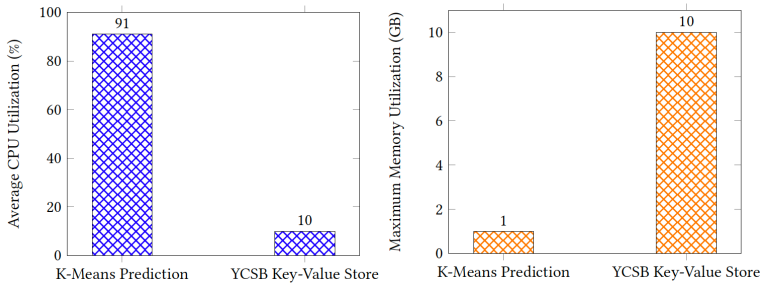
Wu, C., Mehta, B., Amiri, M. J., Marcus, R., & Loo, B. T. AdaChain: A Learned Adaptive Blockchain, VLDB'23

- Hardware resource management
  - Elasticity of disaggregated data center (DDC) infrastructure
  - Switching between DDC vs. non-DDC traditional setup
  - How to deal with the high overhead of switching?

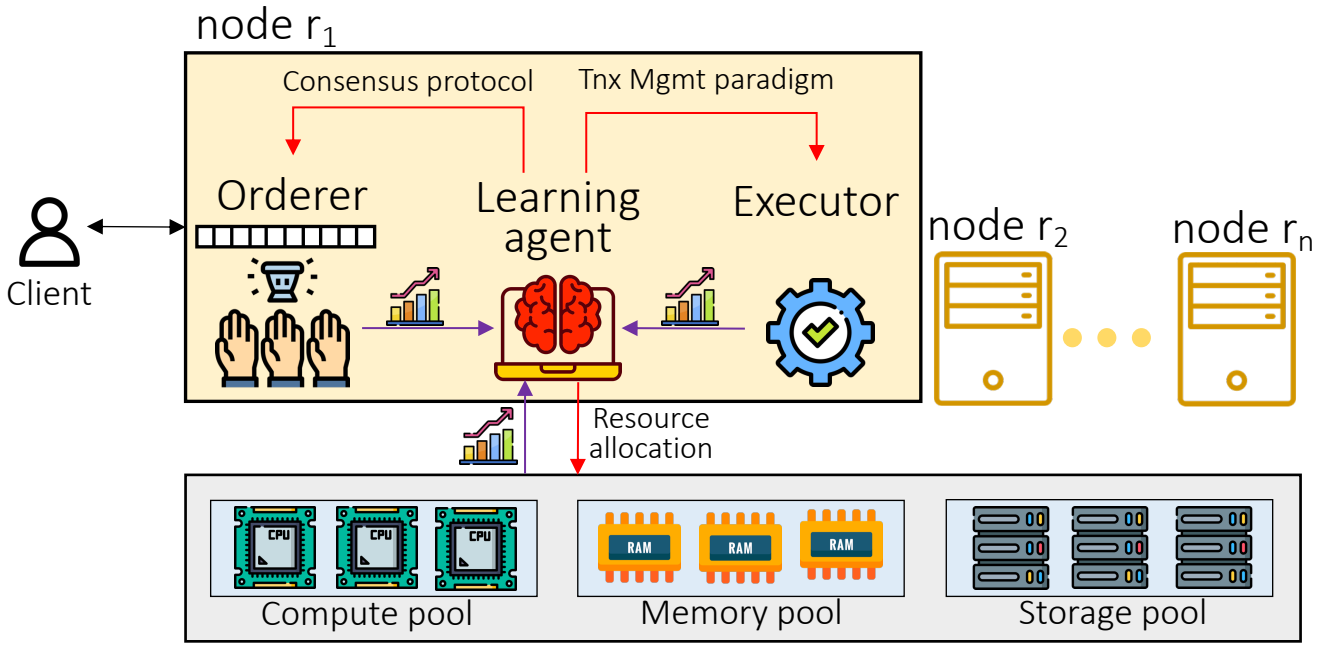
Wu, C., Amiri, M. J., Asch, J., Nagda, H., Zhang, Q., & Loo, B. T. FlexChain: an elastic disaggregated blockchain, VLDB'23



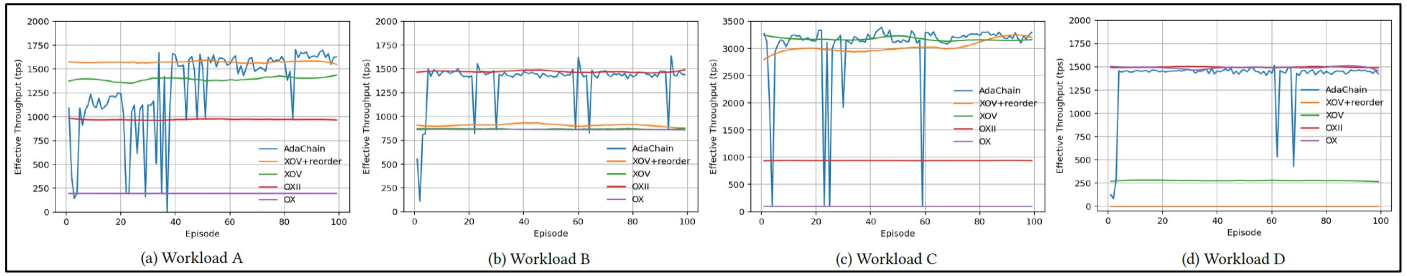
Workload	Write Ratio	Contention Level	Load	Compute Intensity
A	low	high	high	high
B	moderate	high	moderate	low
C	moderate	low	high	Very high
D	high	very high	moderate	Very low

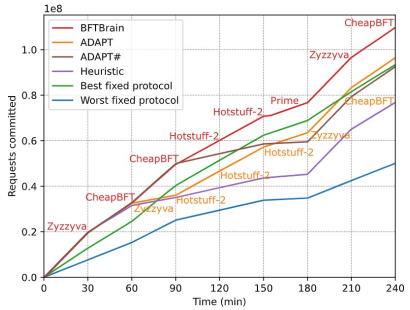


Full Stack Adaptivity



- Robust online data collection
  - No single trusted entity
  - Each node has a learning agent
  - Nodes agree on decisions
  - Featurizing faults and protocols
- Switch at runtime
  - Protocol, paradigm or resources
- Cross-layer adaptivity
  - Identify performance bottlenecks
  - Protocol/paradigm Compatibility





Wu, C., Amiri, M. J., Qin, H., Mehta, B., Marcus, R., & Loo, B. T. , Towards Full Stack Adaptivity in Permissioned Blockchains. VLDB'24


THANK YOU!
Questions?