ulm university universität ulm

Systems Support For Efficient State-Machine Replication

Gerhard Habiger
Franz J. Hauck

FBSYS Herbst 2019

2019-11-21

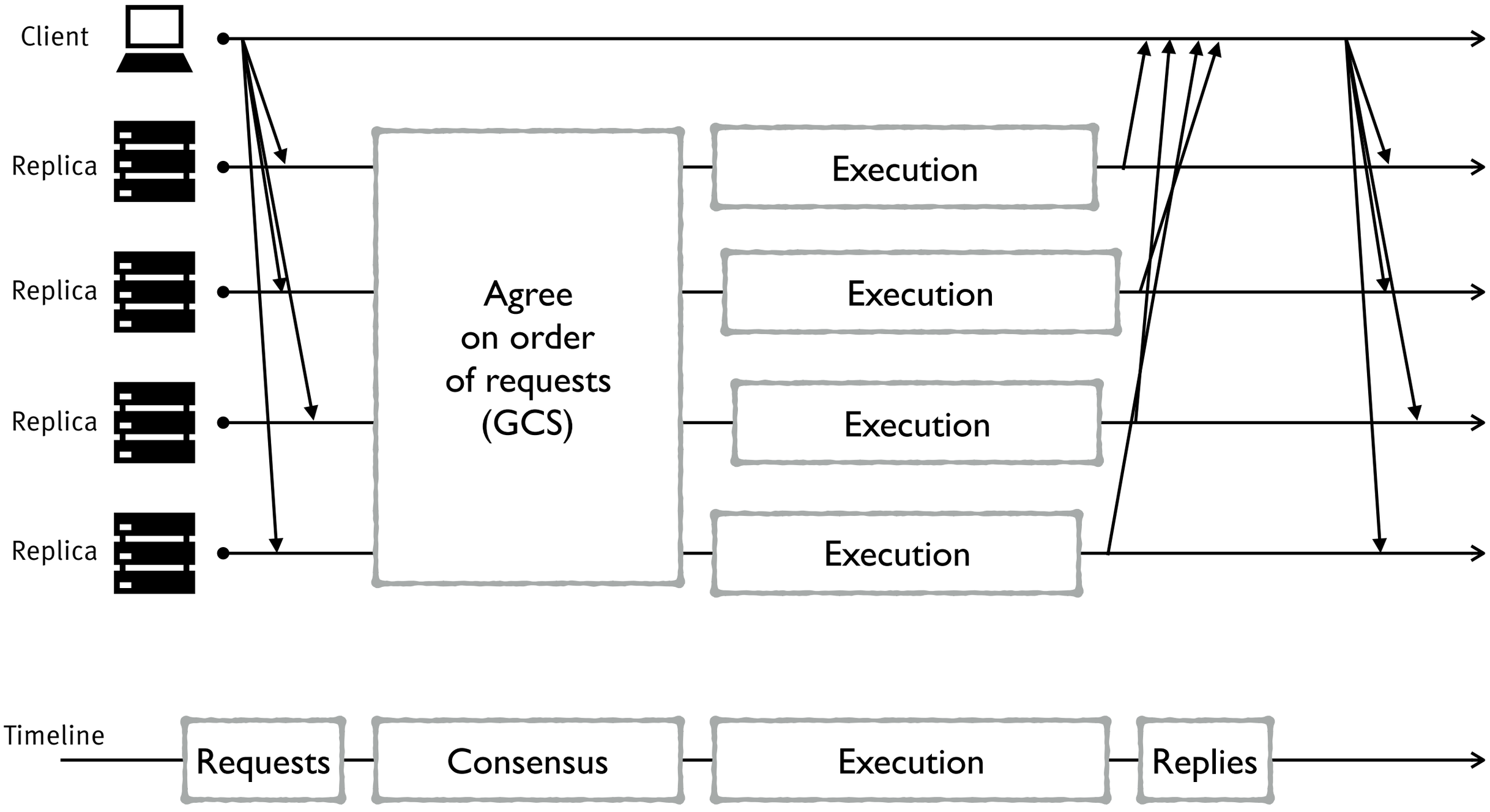
SMR

- Service modelled as deterministic finite state machine
- DSFM replicated across multiple servers (“replicas”)
- Incoming requests have to be totally ordered

➡ All replicas yield same state after receiving the same requests

➡ Can tolerate Byzantine faults

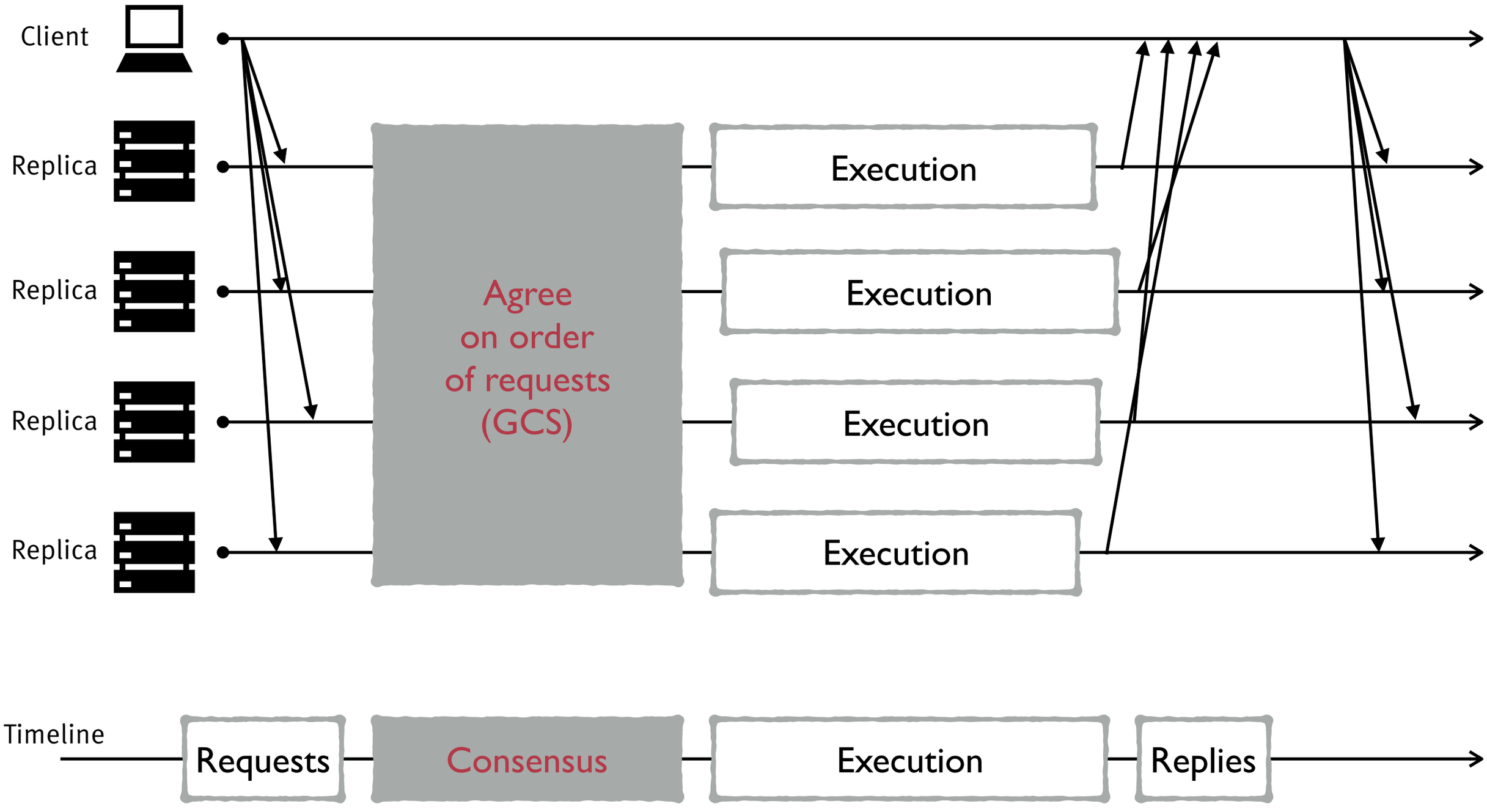
SMR


Client
Replica
Replica
Replica
Replica
Agree on order of requests (GCS)
Execution
Execution
Execution
Execution
Timeline
Requests
Consensus
Execution
Replies

Efficient SMR

- Topics we are looking at:
- GCS —› Consensus with trusted components
- Execution —› Parallelized request handling; multithreading
- Variable request arrival rates —› Adapt to current load
- Checkpointing/Recovery —› Re-enable for parallel execution

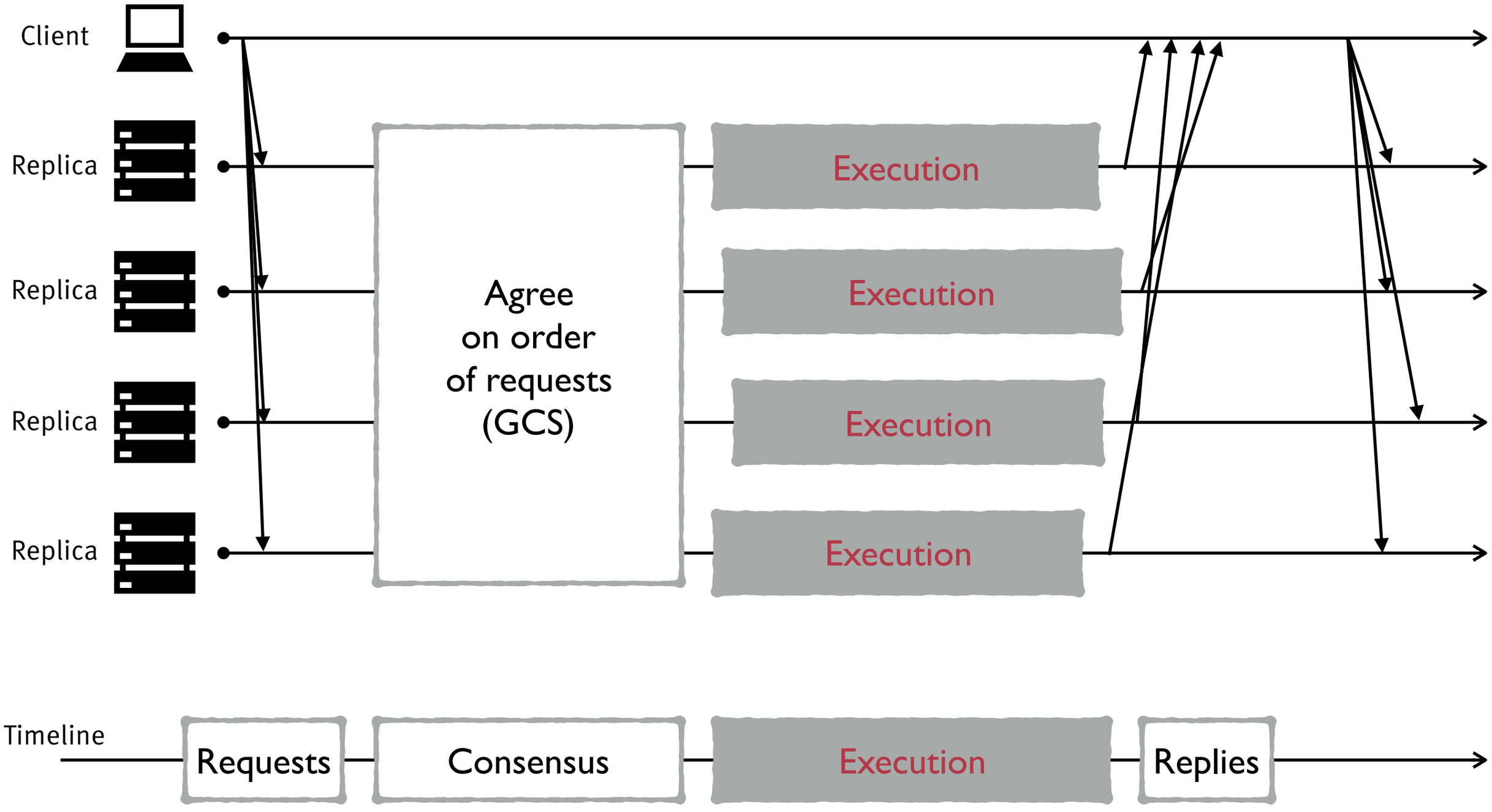
SMR


Client
Replica
Replica
Replica
Replica
Agree on order of requests (GCS)
Execution
Execution
Execution
Execution
Timeline
Requests
Consensus
Execution
Replies

Trusted Components

- Deploying trusted components in the GCS can reduce overhead and increase performance
- Example: EBAWA
  - Contains several liveness and safety bugs we detected and fixed
  - New algorithm supporting trusted components: *Aphousia* (not yet published)
- Additional problem: Swapping of consensus algorithms in current systems hard (usually hardwired)
  - Research on modularizing SMR

SMR

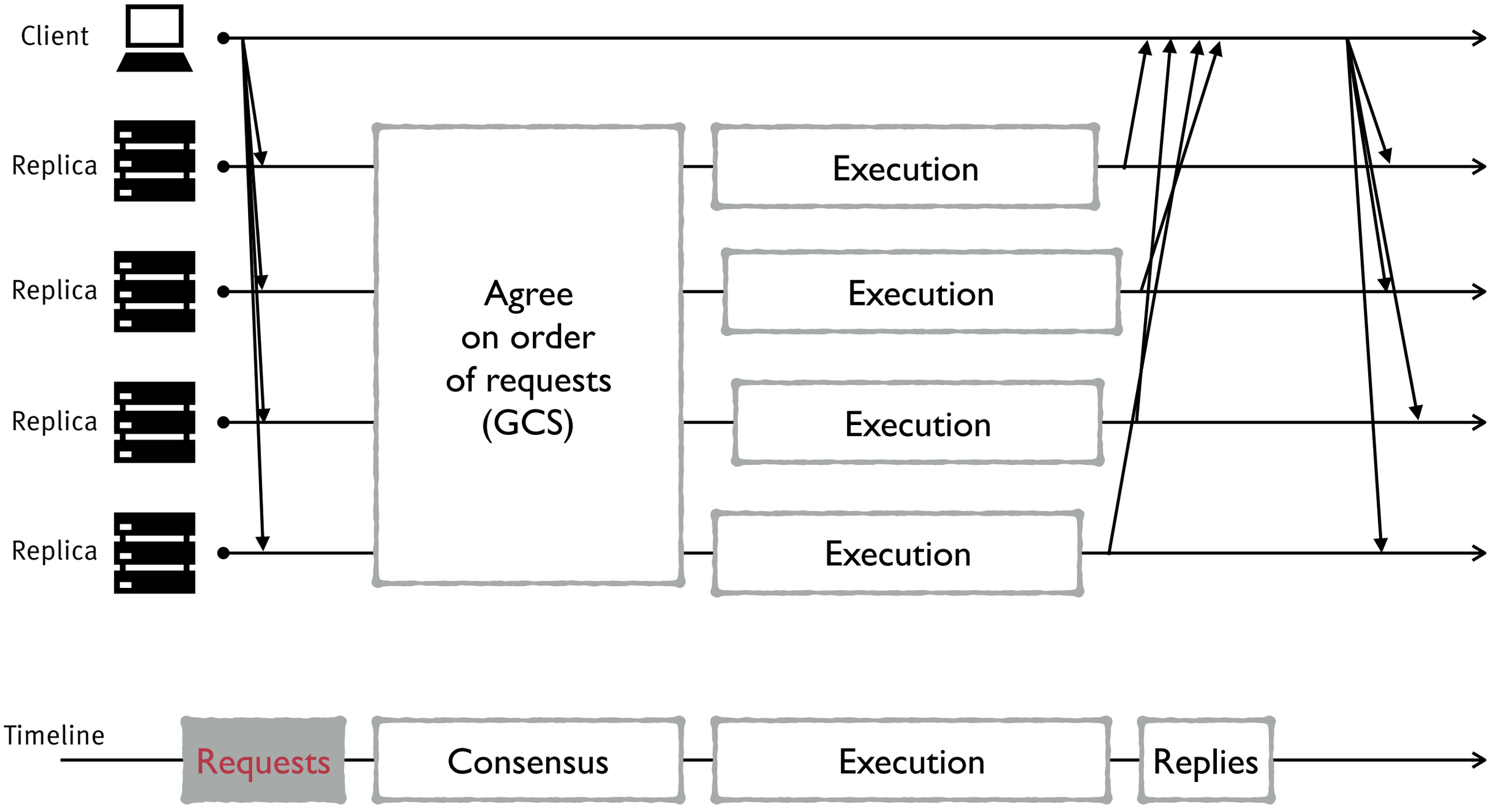

Client
Replica
Replica
Replica
Replica
Agree on order of requests (GCS)
Execution
Execution
Execution
Execution
Timeline
Requests
Consensus
Execution
Replies

Parallel Request Handling

- Main approaches:
  - Distinguish (in-)dependent requests, sequentialize remaining ones
  - Deterministic multithreading (DetMT)
- We use DetMT with Unified Deterministic Scheduler (UDS)
  - Parametrized deterministic scheduler
  - Allows reconfiguration during runtime

F. J. Hauck, G. Habiger, and J. Domaschka. 2016. UDS: A Novel and Flexible Scheduling Algorithm for Deterministic Multithreading. In 2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS). 177– 186.

SMR

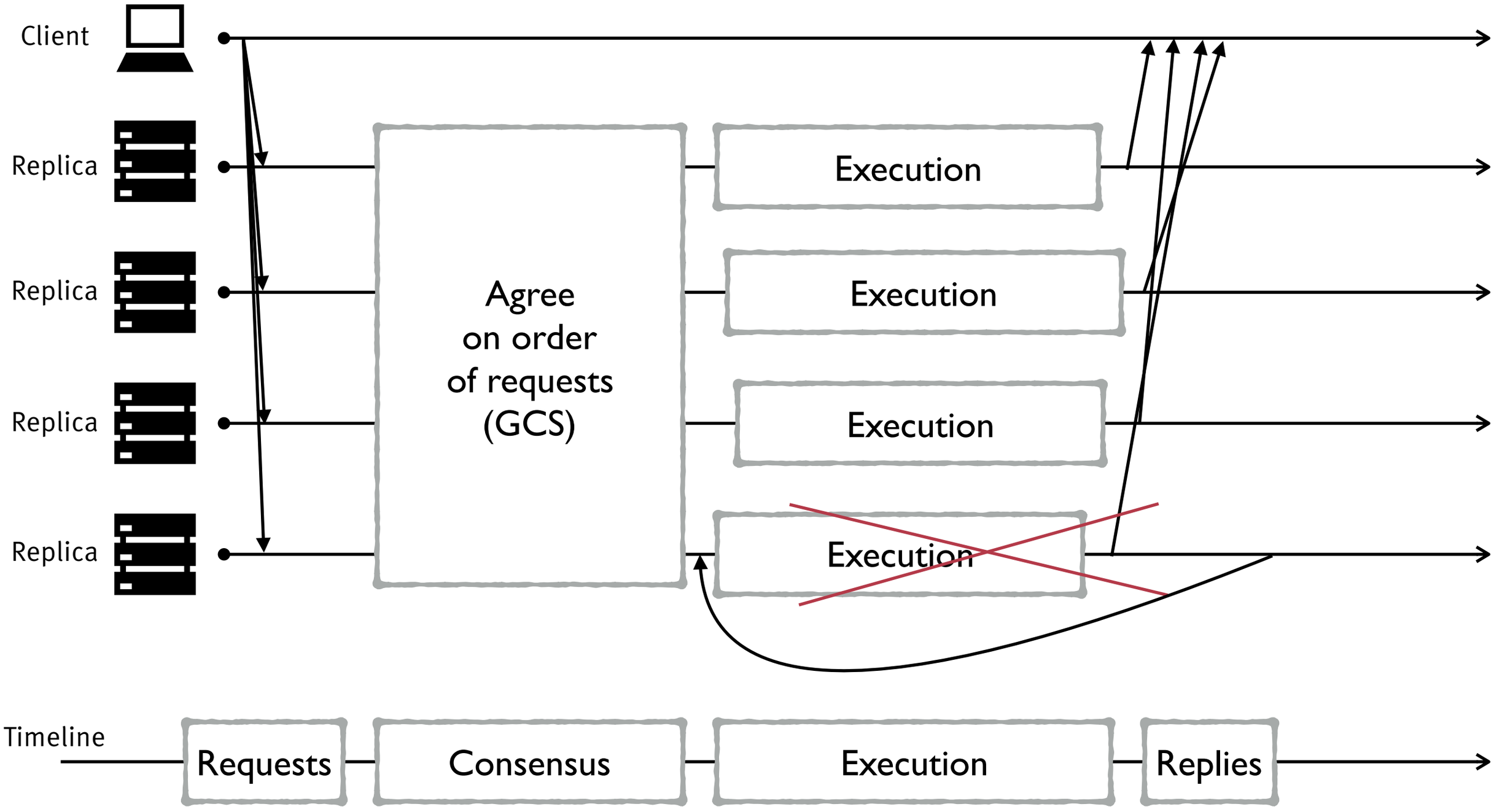

Client
Replica
Replica
Replica
Replica
Agree on order of requests (GCS)
Execution
Execution
Execution
Execution
Timeline
Requests
Consensus
Execution
Replies

Variable Load

- Request arrival rate usually not uniform over time
- But: Provisioning of resources (number of replicas, hardware of each replica, software configuration) usually fixed
- Scaling horizontally is difficult/pointless due to Consensus
- ➡ Scale hardware vertically
- ➡ Reconfigure system during runtime

G. Habiger, F. J. Hauck, J. Köstler, and H. P. Reiser. 2018. Resource-Efficient State-Machine Replication with Multithreading and Vertical Scaling. In 2018 14th European Dependable Computing Conference (EDCC). 87–94. https://doi.org/10. 1109/EDCC.2018.00024

SMR


Client
Replica
Replica
Replica
Replica
Agree
on order
of requests
(GCS)
Execution
Execution
Execution
Execution
Timeline
Requests
Consensus
Execution
Replies

Checkpointing

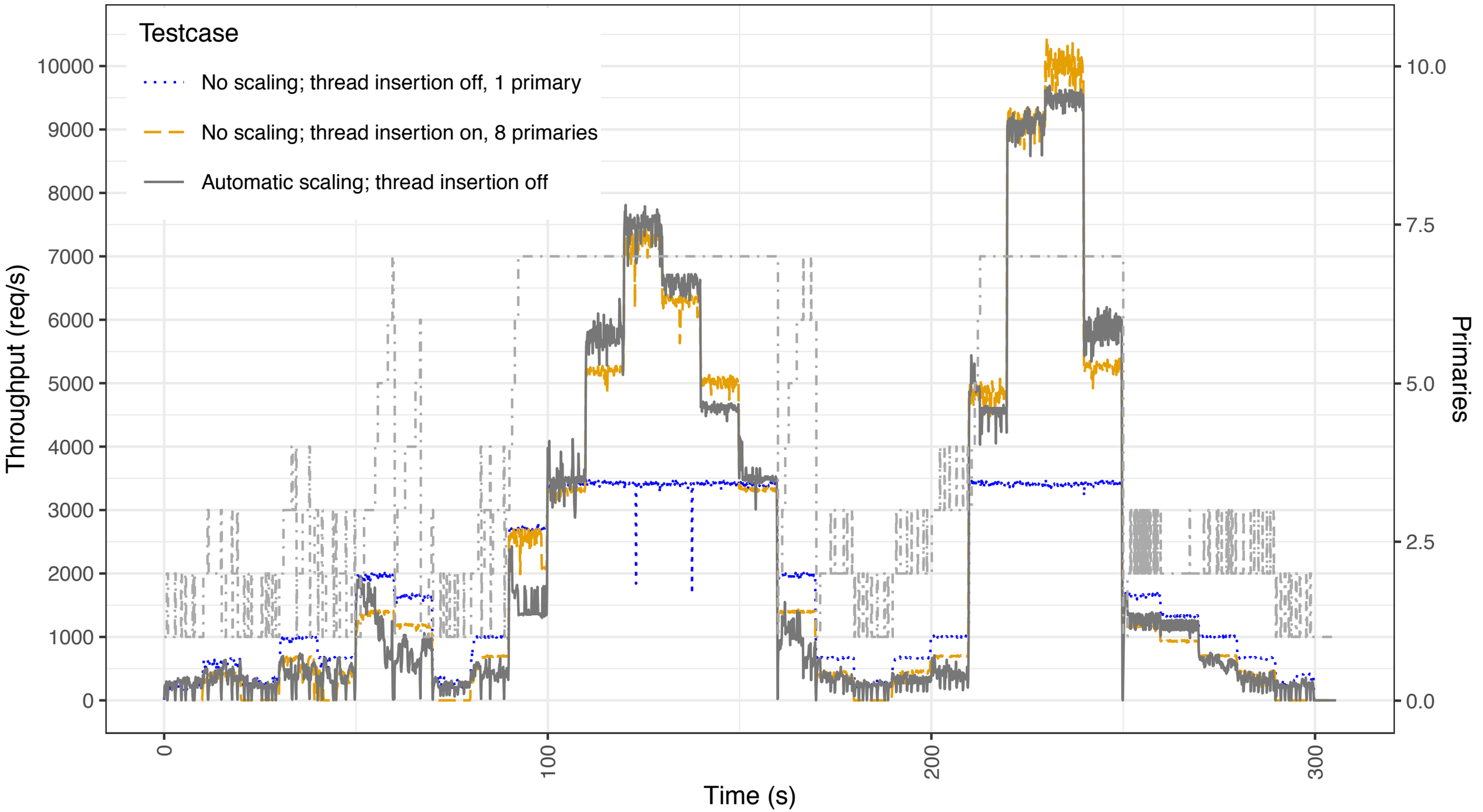
- Parallelizing request handling significantly complicates checkpointing
- Simplest approach: Stop-the-world (implemented)
- Advanced approaches: E.g. rolling snapshots during execution
  - tbd … still in the works

Reconfiguration Benefits

- Reconfiguring deterministic scheduler during runtime yields throughput and latency improvements
- Allows for best case performance in all scenarios
- Optimal resource usage in combination with dynamic vertical hardware scaling

(Publication currently in submission)

Reconfiguration Benefits



Conclusion

- Several parts of SMR can be improved
- Our goal: Easy (self-)configuring SMR system, including dynamic runtime adaptation
- 4 main topics we are looking at to achieve this:
  - Parallelized request handling via DetMT
  - GCS with trusted components
  - Dynamic vertical scaling and reconfiguration
  - Checkpointing during parallel execution