MLMQ
Mat Luke Message Queuing

Advanced Systems Lab – Milestone 1

Matthias Ganz & Lukas Elmer

HS 13/14

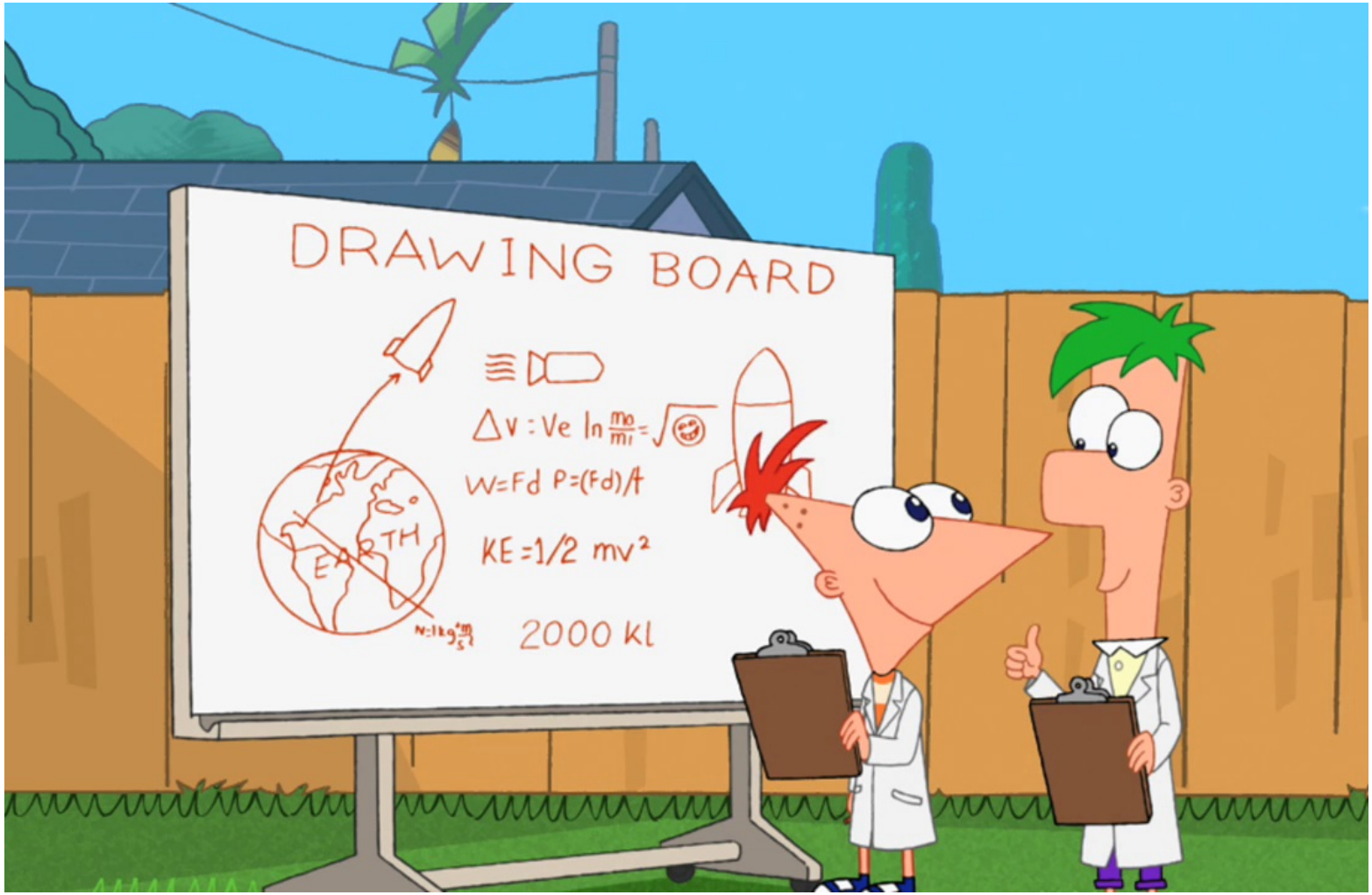
ETHZ


United States
Atlantis

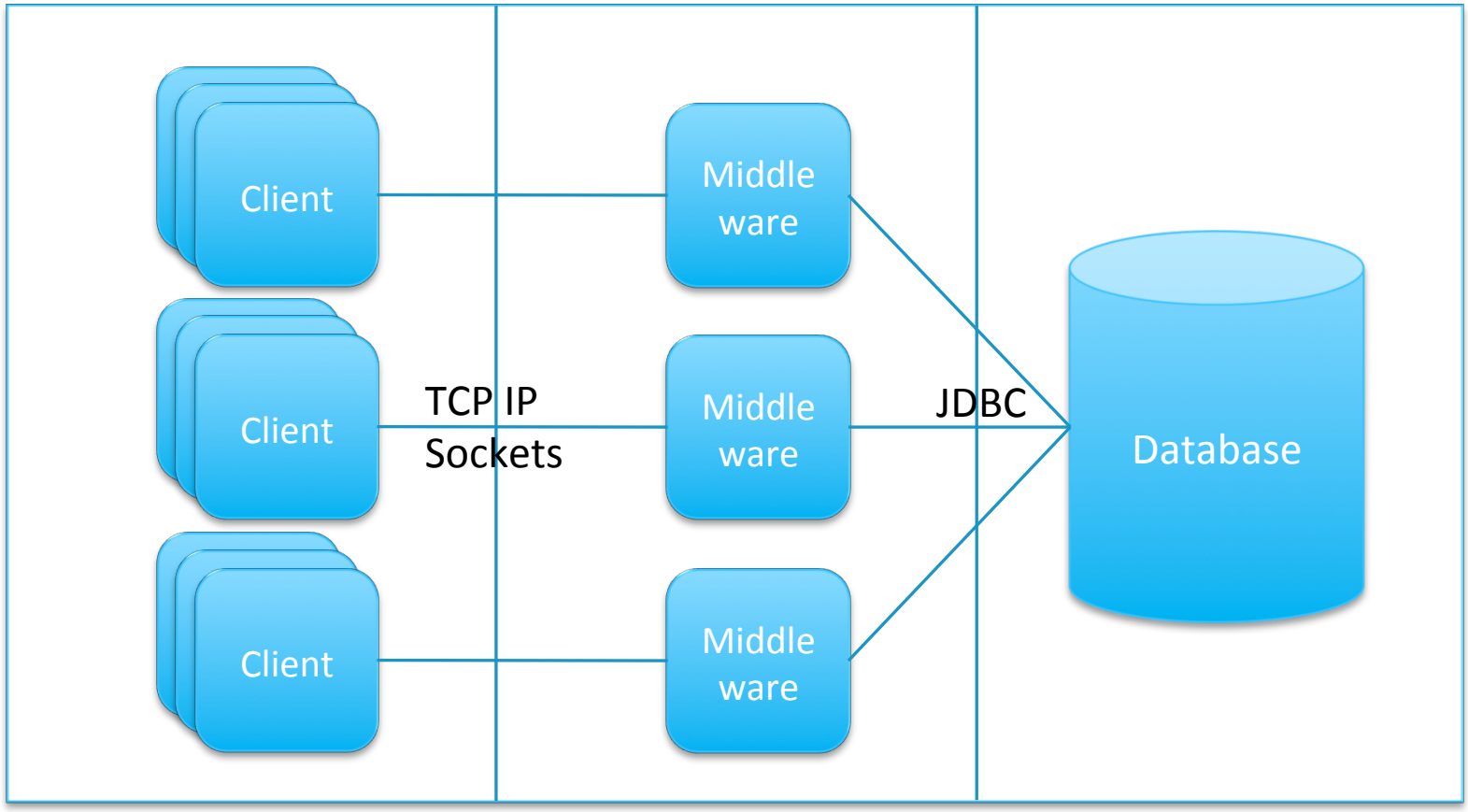
Agenda

* Design
* Experiments
* Lessons Learned about the System

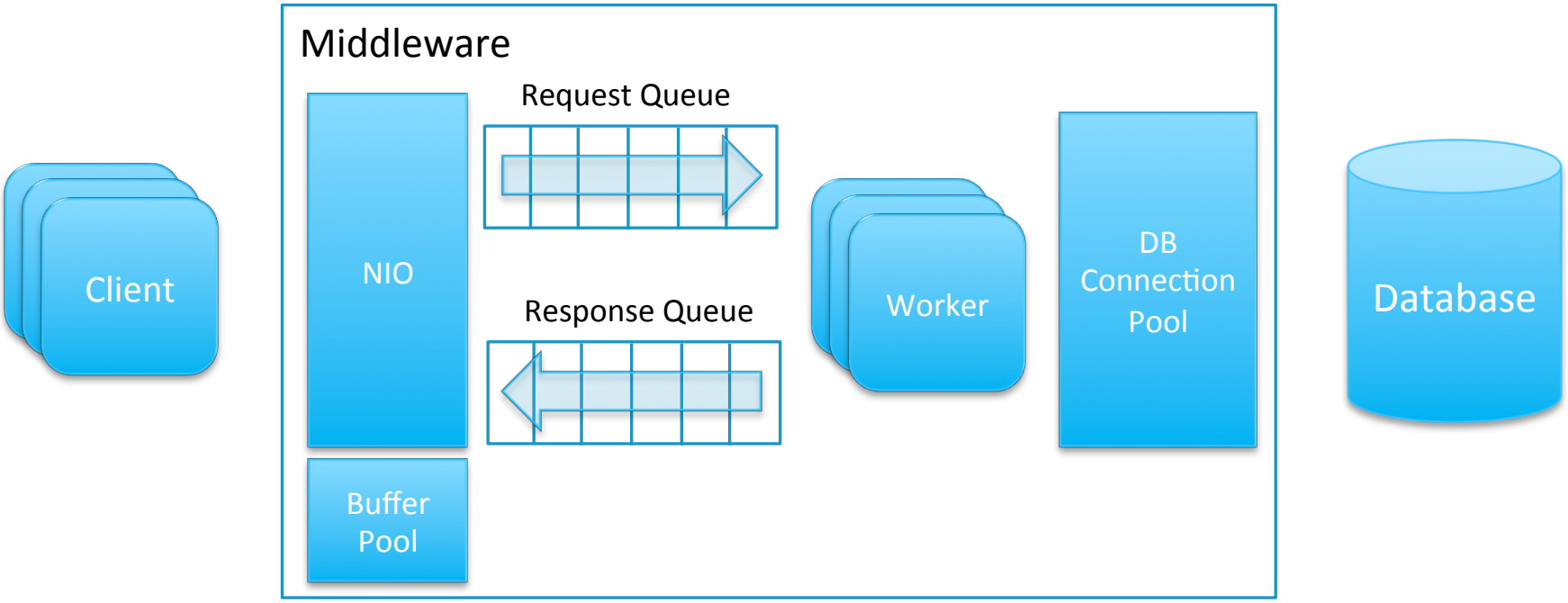
Design



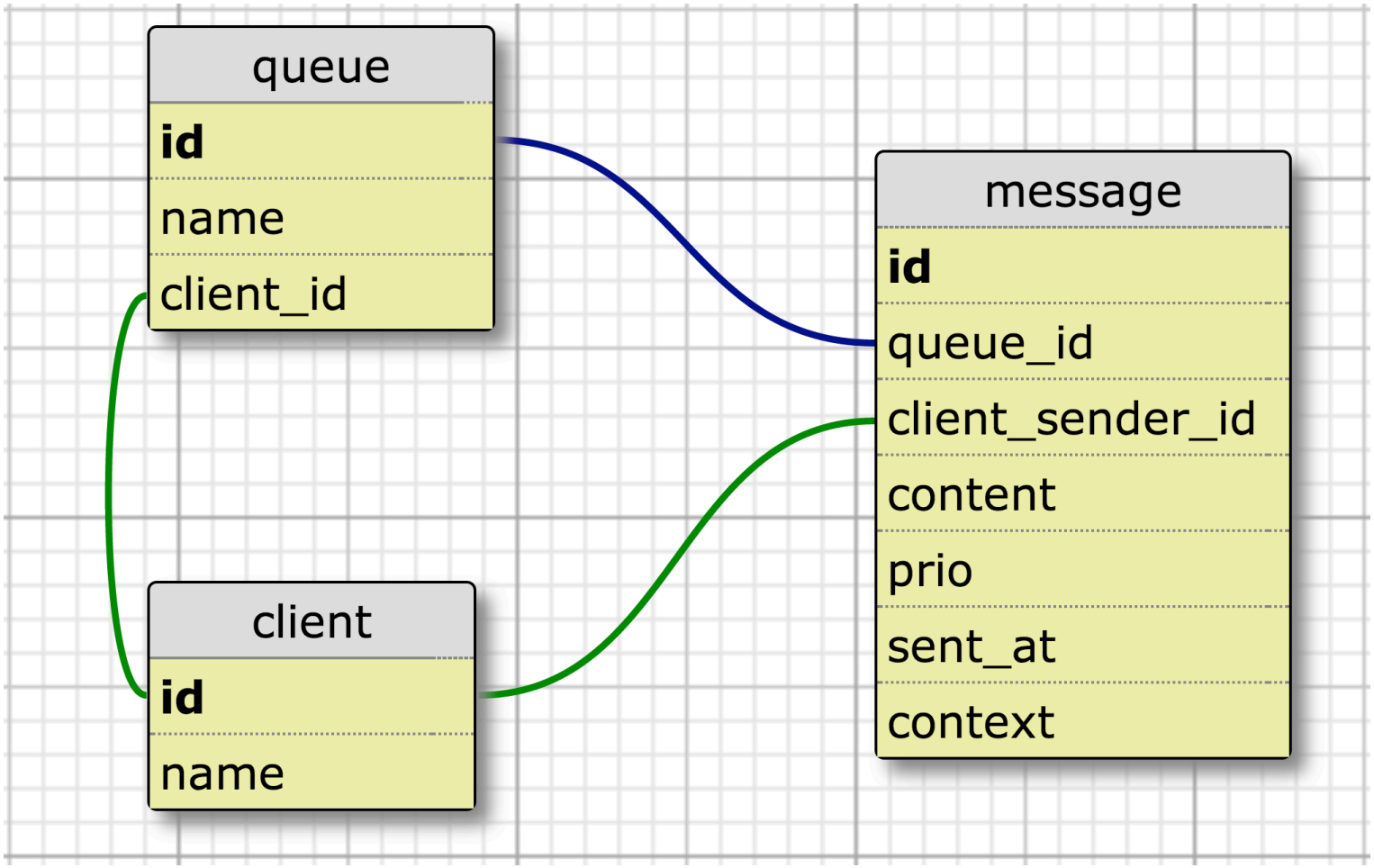
Design – Overview



Design – Middleware



Design – Database



Design – Database Interface

* Indices
  * Queue: client_id
  * Client: name
  * Message: queue_id, prio, sent_at, client_sender_id, context
* Important stored procedures
  * peekMessage
  * dequeueMessage
    * Locks a specific record for dequeuing
* Prepared statements & auto commit

Design – Client Interface

ClientDto register()
ClientDto lookupClient(String clientName)

QueueDto lookupClientQueue(long clientId)
QueueDto createQueue(String queueName)
QueueDto lookupClientQueue(String queueName)
void deleteQueue(long id)

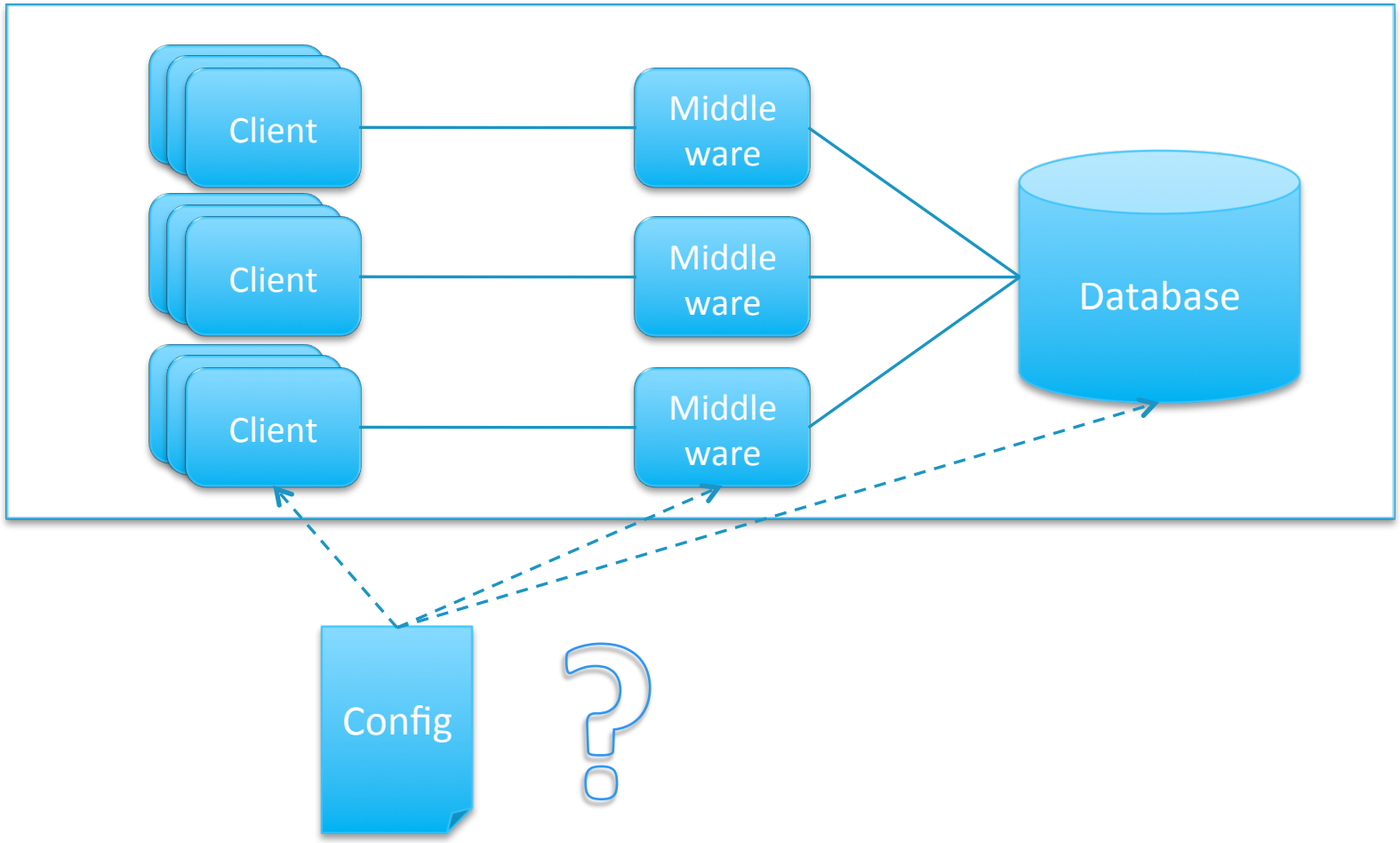
void sendMessage(long queueId, byte[] content, int prio)
void sendMessage(long[] queueIds, byte[] content, int prio)
void sendMessageToClient(long clientId, byte[] content, int prio)
long sendRequestToClient(long client, byte[] content, int prio)
long sendResponseToClient(long clientId, long context, byte[] content, int prio)
int queuesWithPendingMessages(List<QueueDto> queues, int maxNumQueues)
MessageDto peekMessage(MessageQueryInfoDto messageQueryInfo)
MessageDto dequeueMessage(MessageQueryInfoDto messageQueryInfo)

Experiments

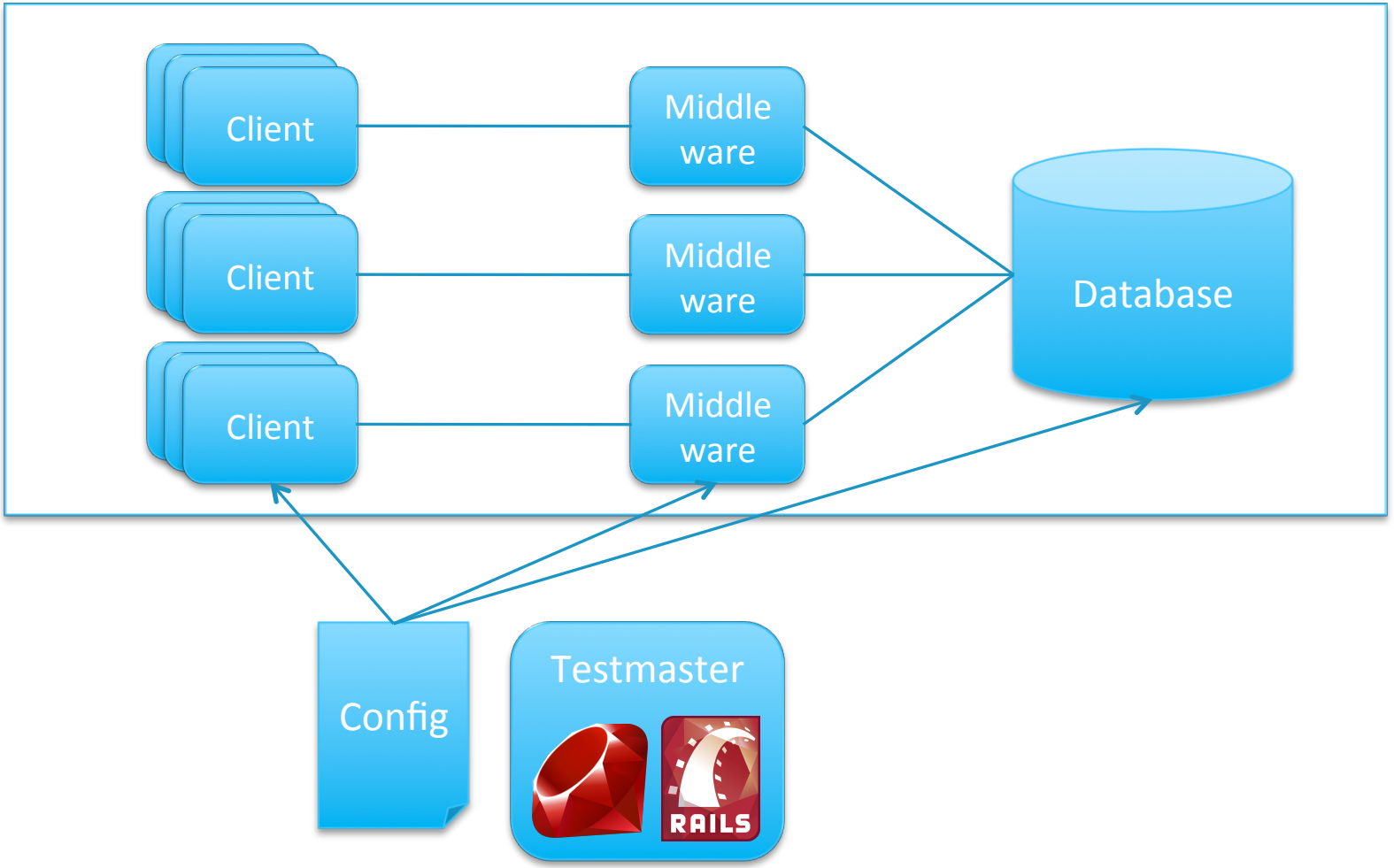
Experiments – Summary

* Stability
  * 2h Test
* Best configuration
  * Throughput vs. response time
  * 2^k Test
* Bottlenecks, optimization Points
  * Which component spends how much time
* System limit
  * Load tests

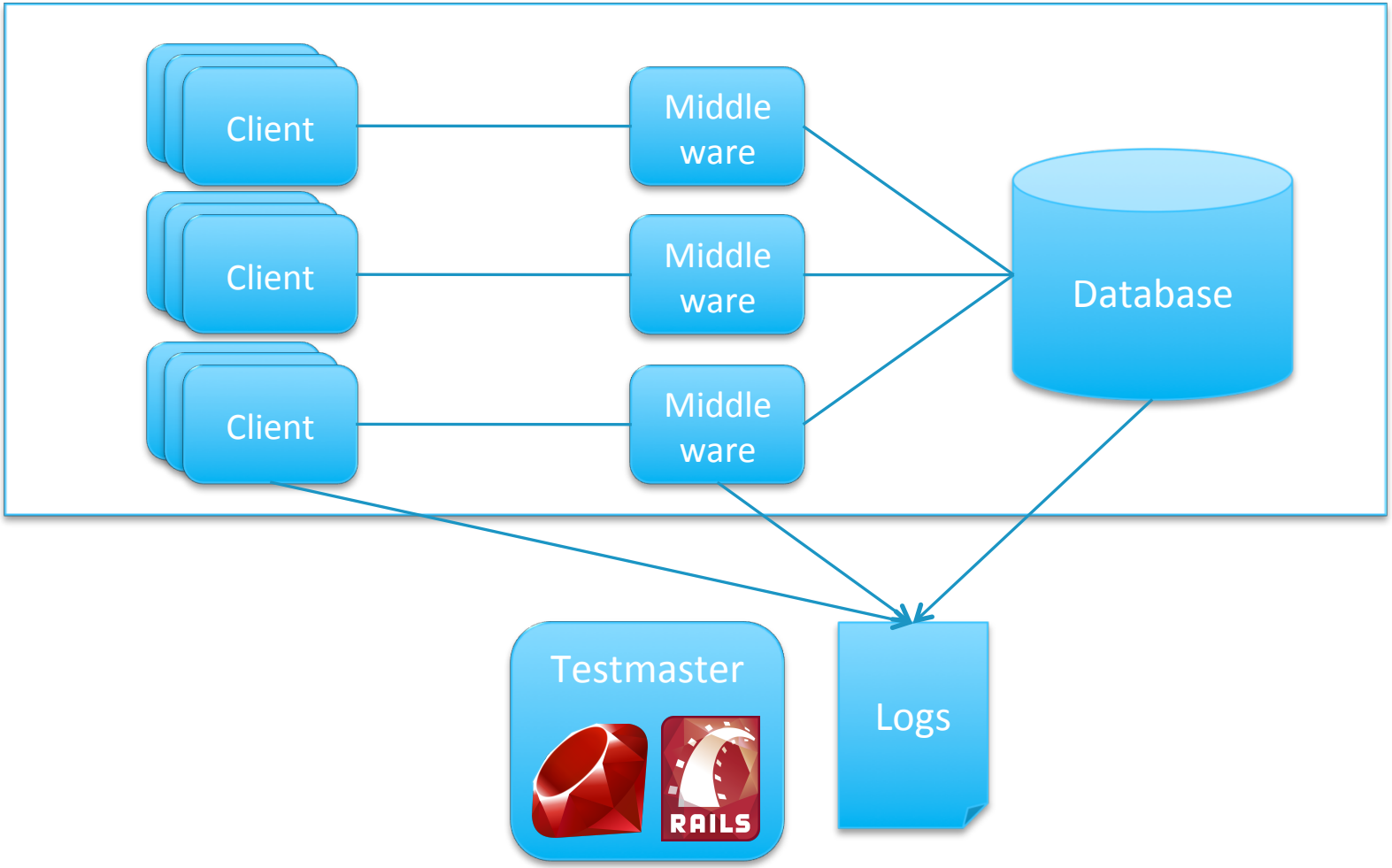
Experiments – Setup



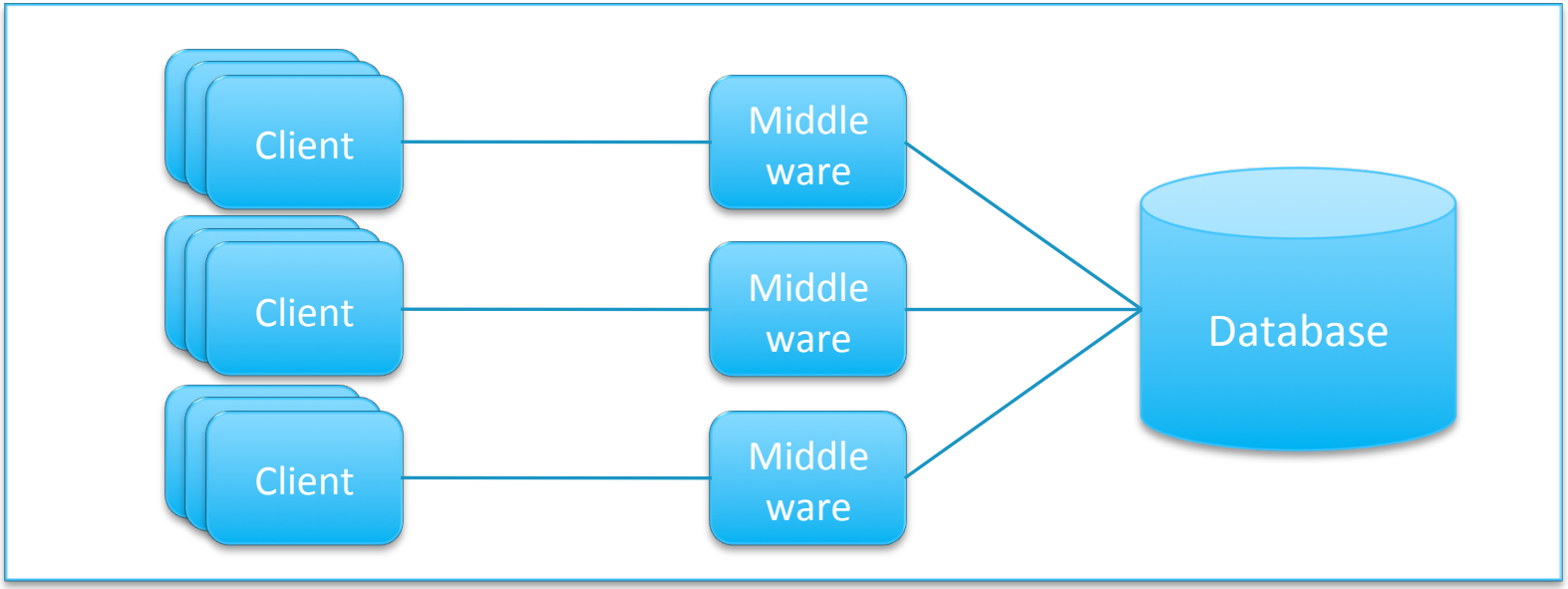
Experiments – Setup



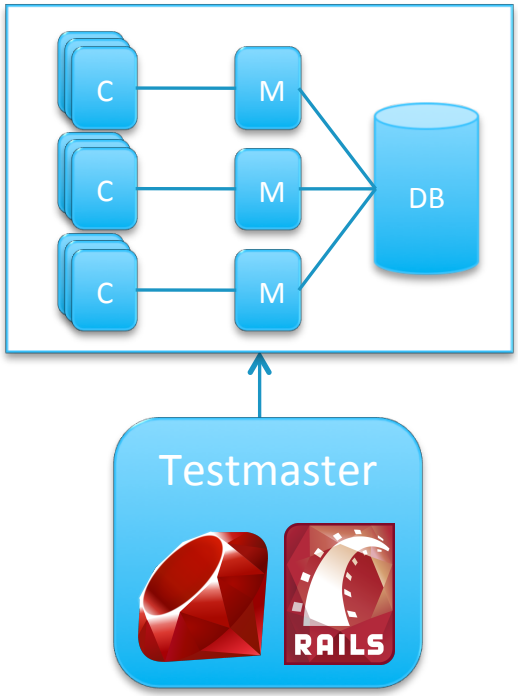
Experiments – Setup



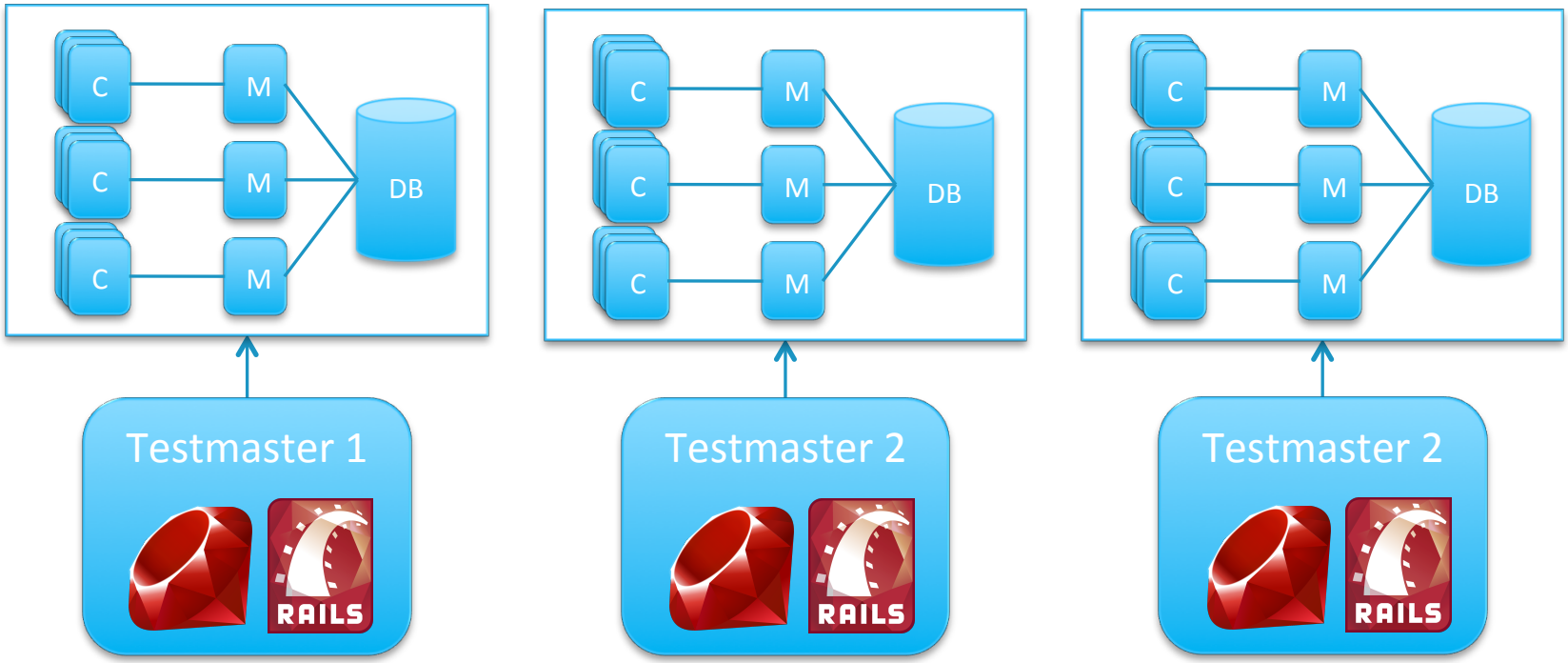
Experiments – Setup



Experiments – Sequential Test



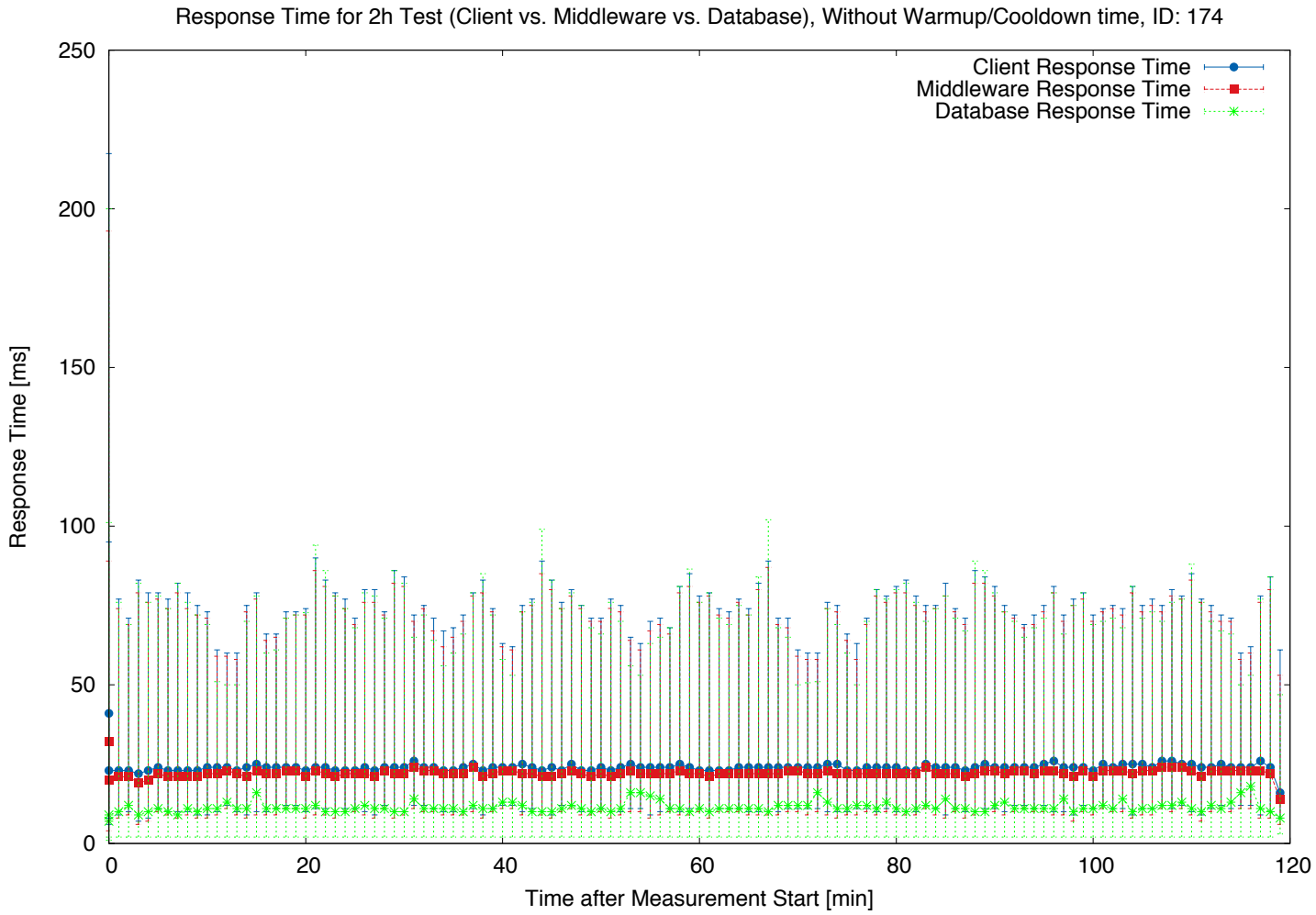
Experiments – Parallel Tests



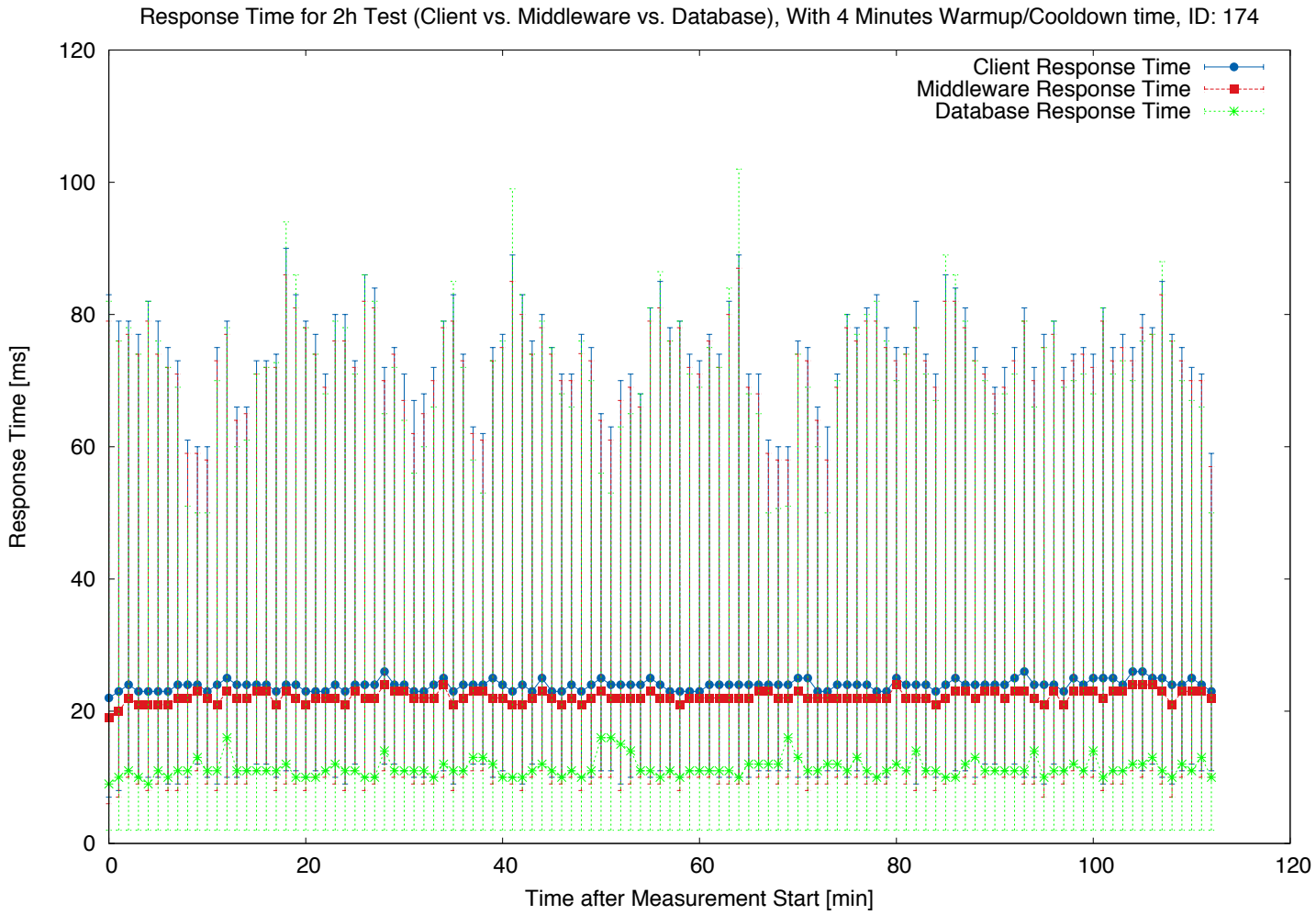
Cannot to 20 Amazon EC2 instances

→ Request to Increase Amazon EC2 Instance Limit

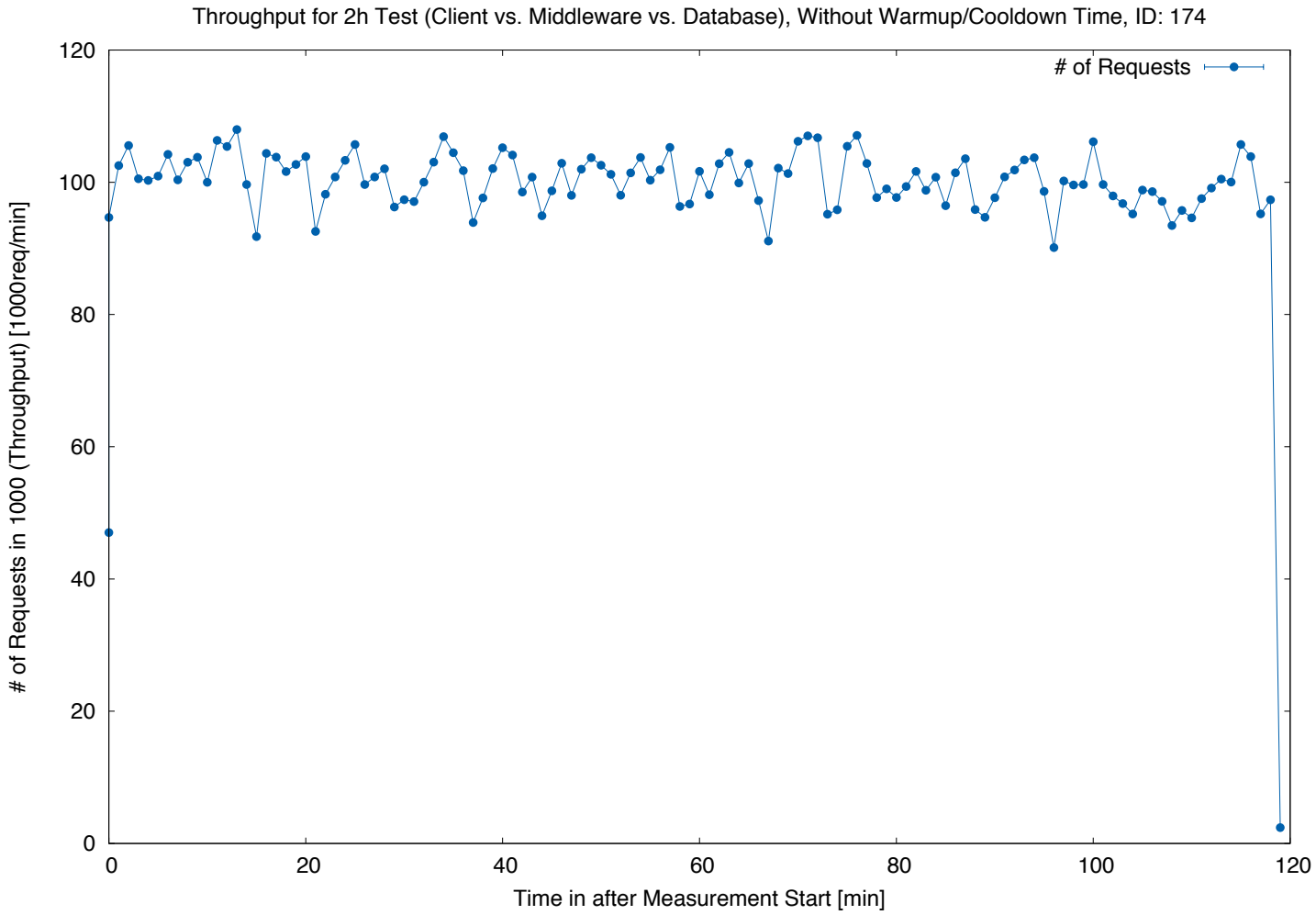
Experiments – 2h Test without warmup / cooldown time, response time



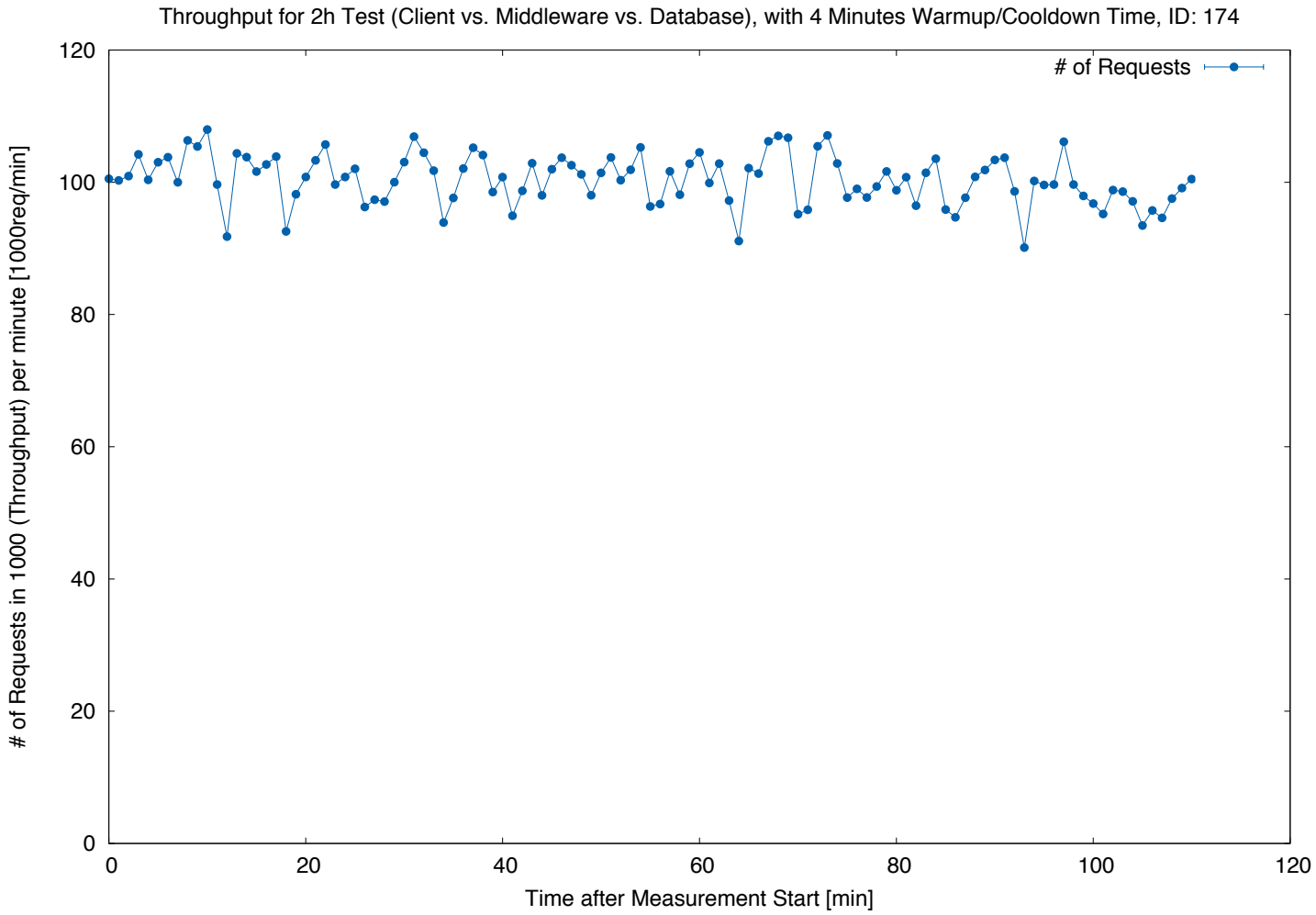
Experiments – 2h Test with 4 minutes warmup / cooldown time, response time



Experiments – 2h Test without warmup / cooldown time, throughput



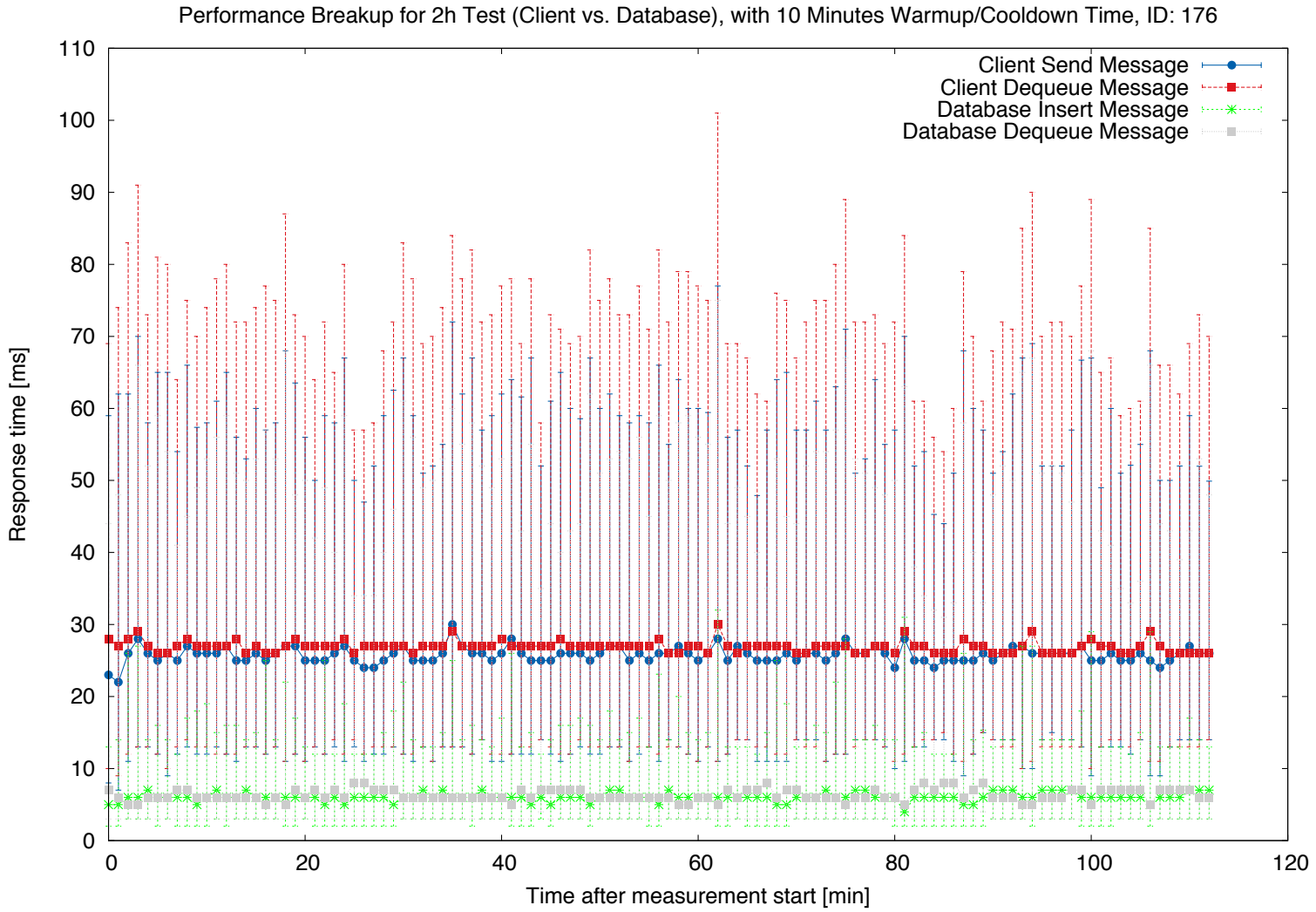
Experiments – 2h Test with 4 minutes warmup / cooldown time, throughput



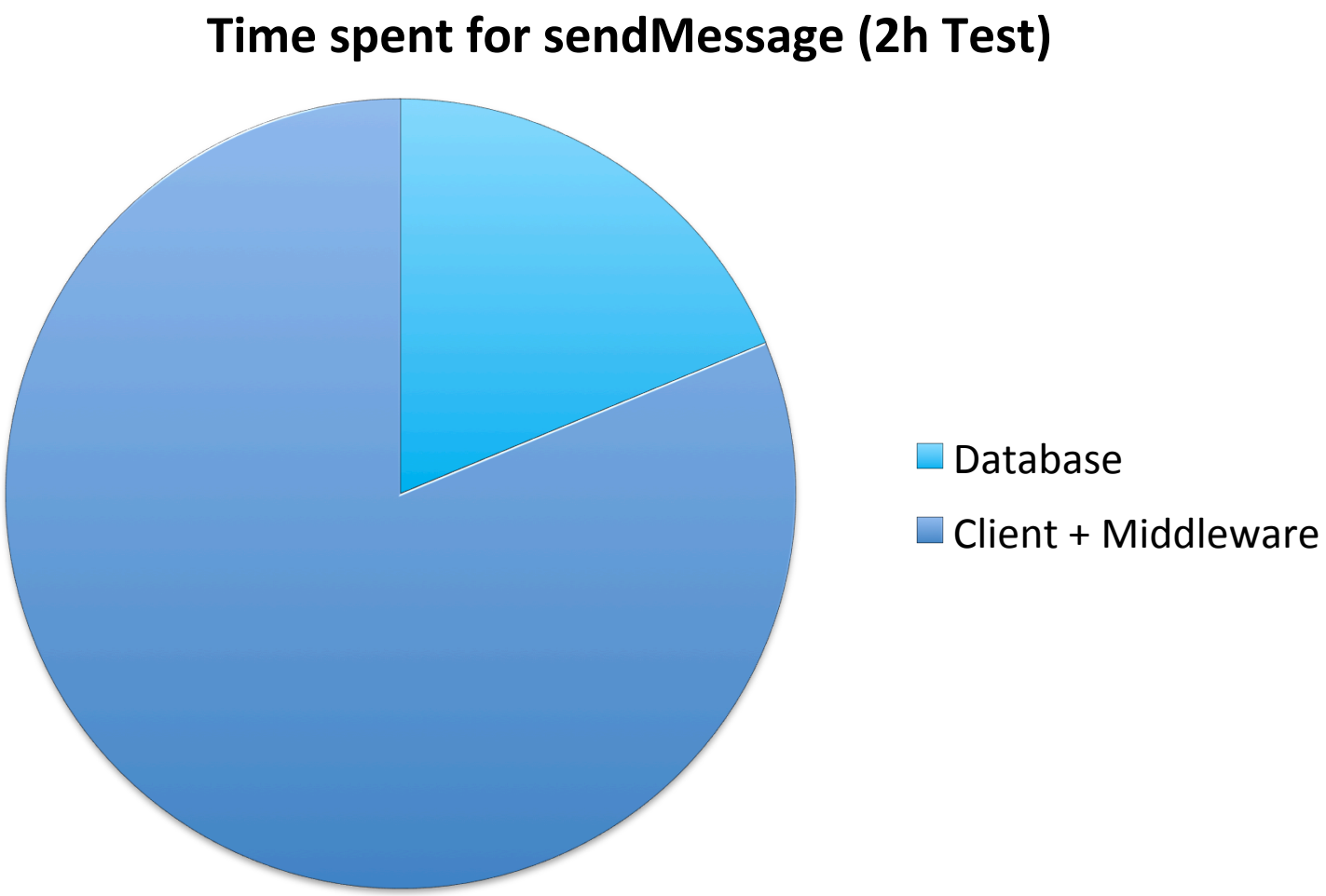
Experiments – 2h Test

* For the 2h Test configuration
  * For 95% of all **sendMessage** requests, the response time will be **under 50 ms**
  * For 95% of all **dequeueMessage** requests, the response time will be **under 62 ms**

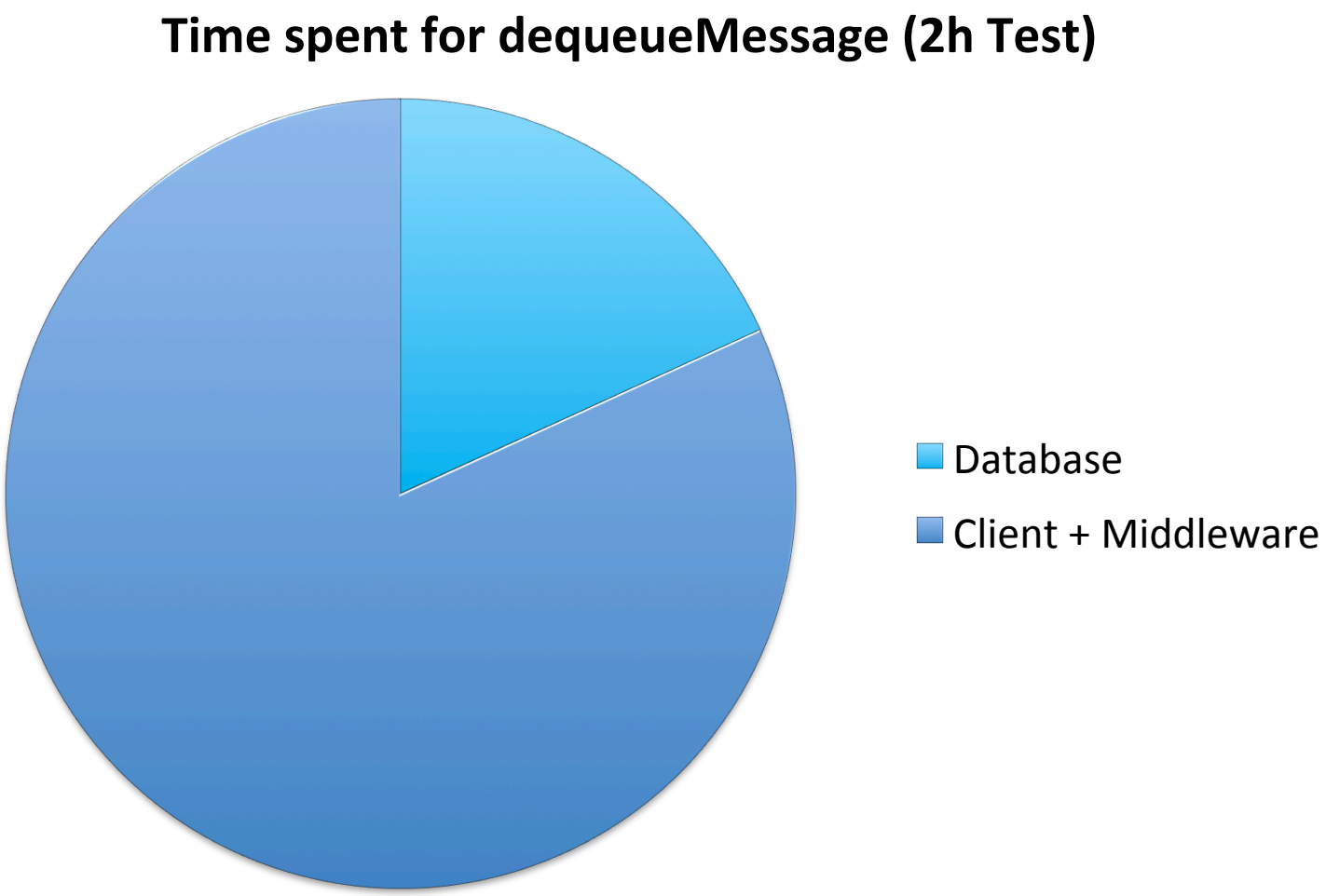
Experiments – Results, 2h Test



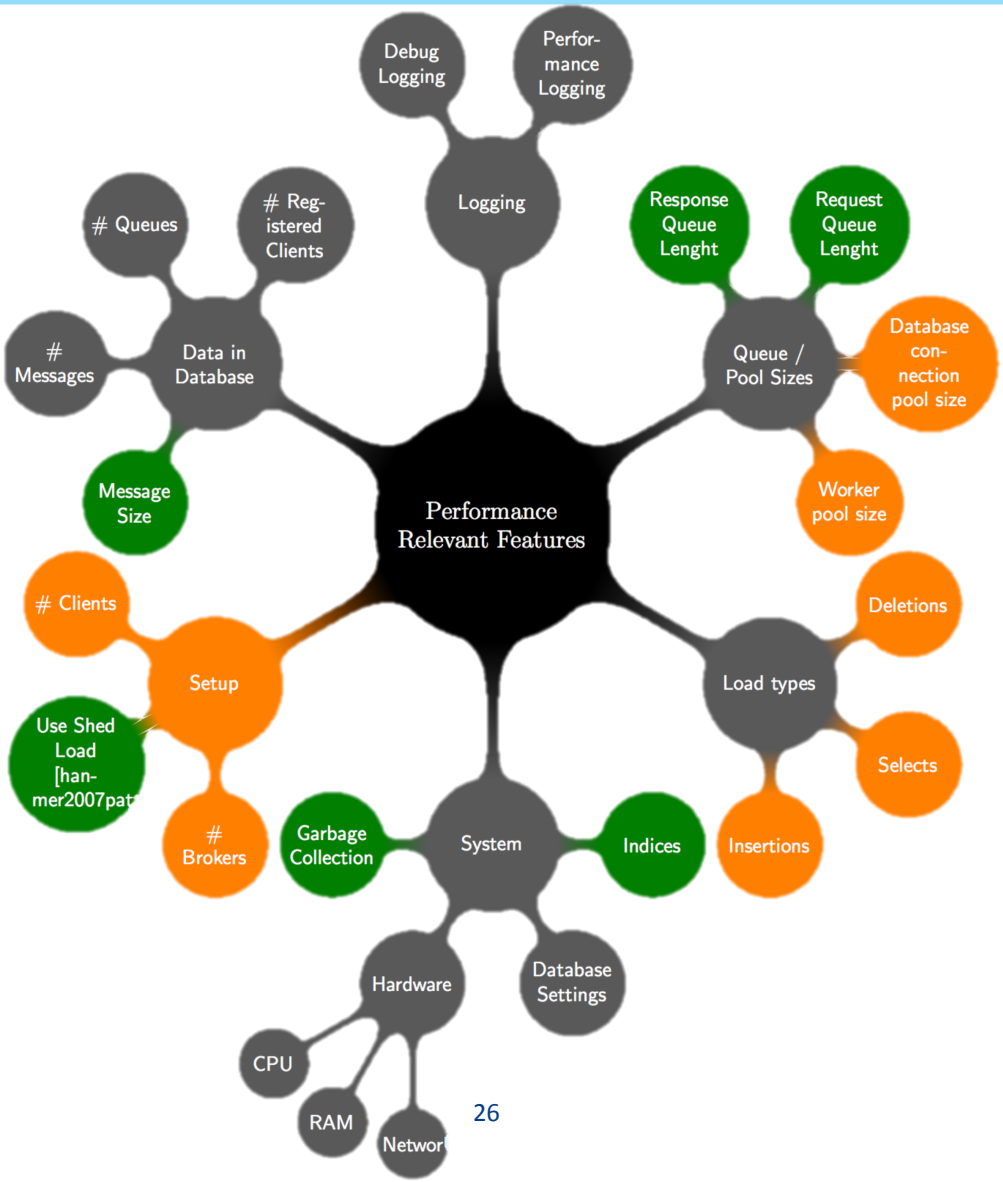
Experiments – Results, 2h Test



Experiments – Results, 2h Test

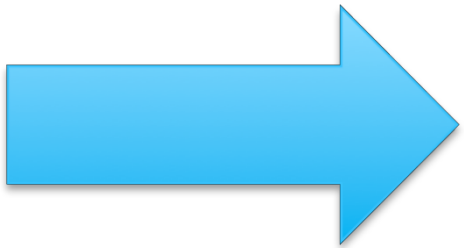


Experiments – Factors



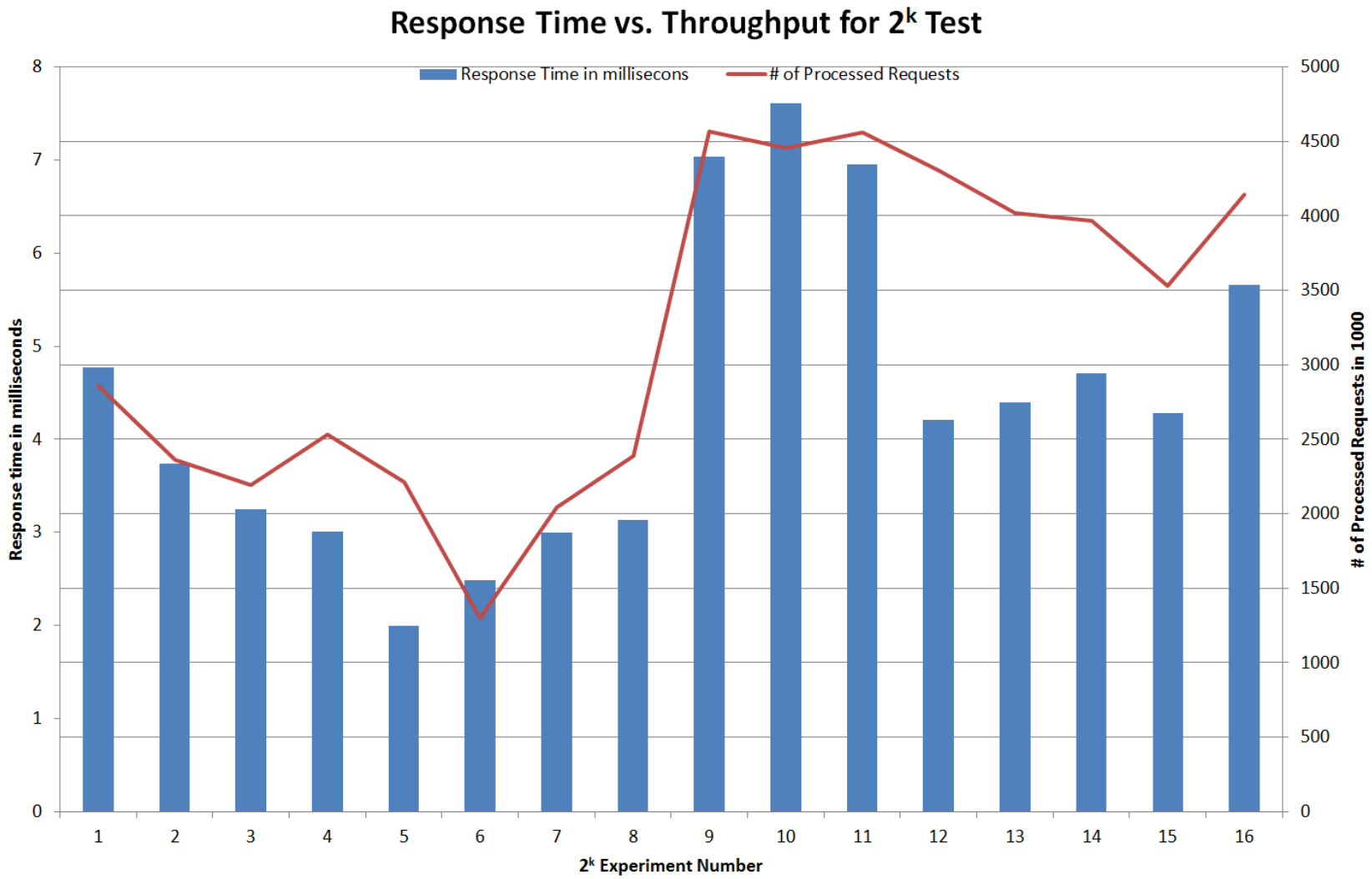
Experiments – Primary Factors

* # Clients
* # Brokers
* DB Connection Pool Size
* Worker Pool Size



2^k Test

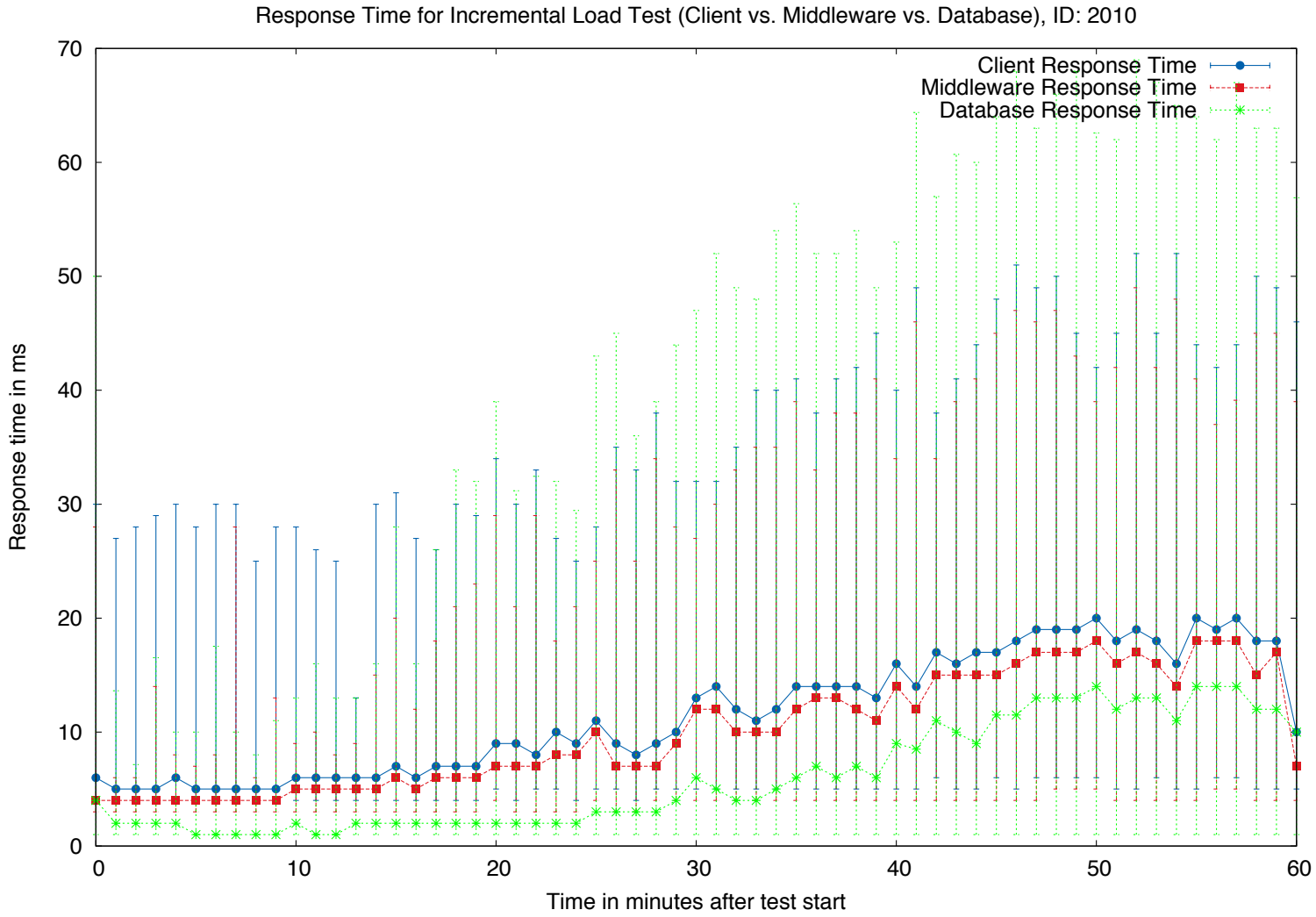
Experiments – Results: 2^k Test



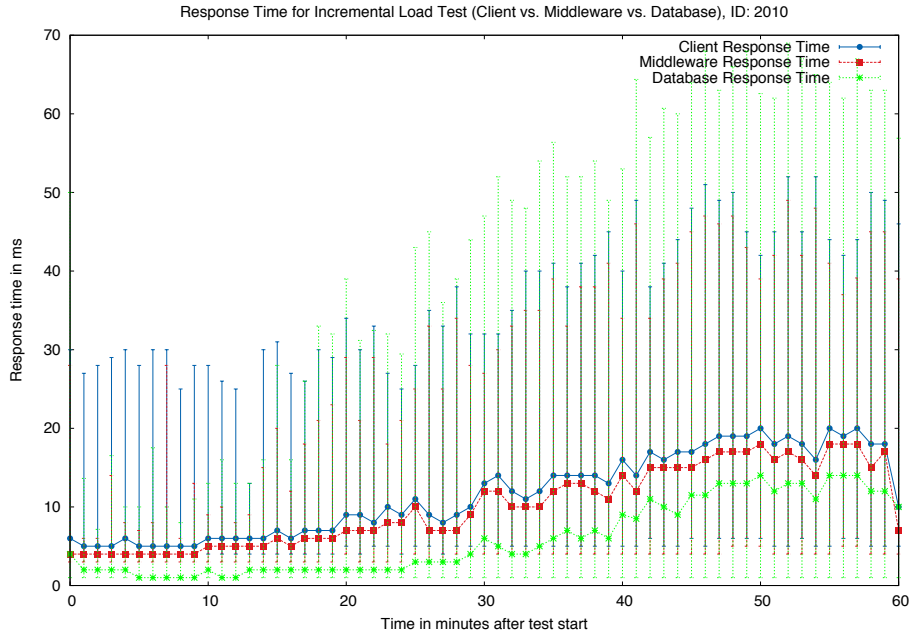
Experiments – Results: 2^k Test

* Interpretation
  * Higher throughput ➔ higher response time

Experiments – Incremental Load Test, Response Time

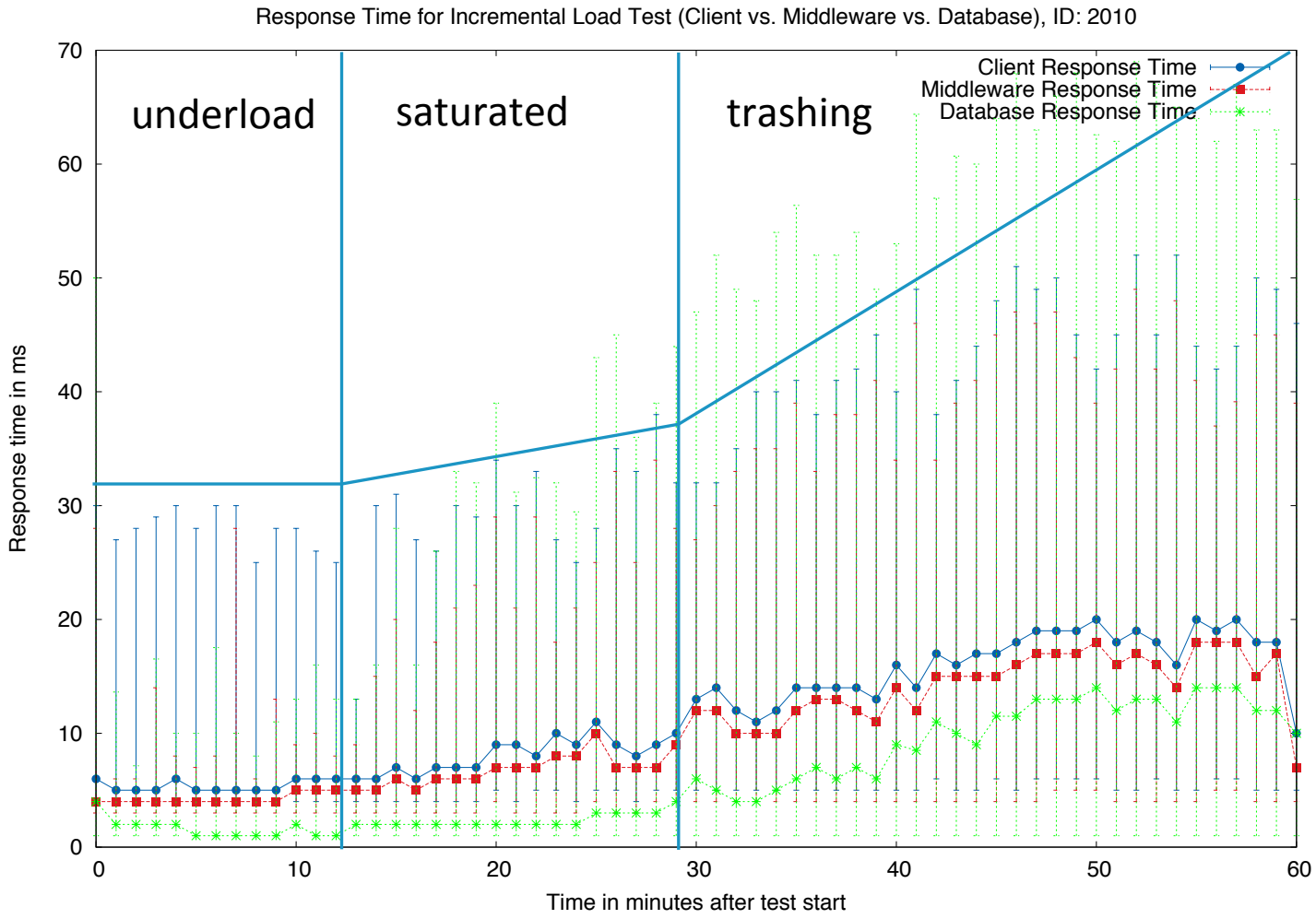


Experiments – Incremental Load Test, Response Time

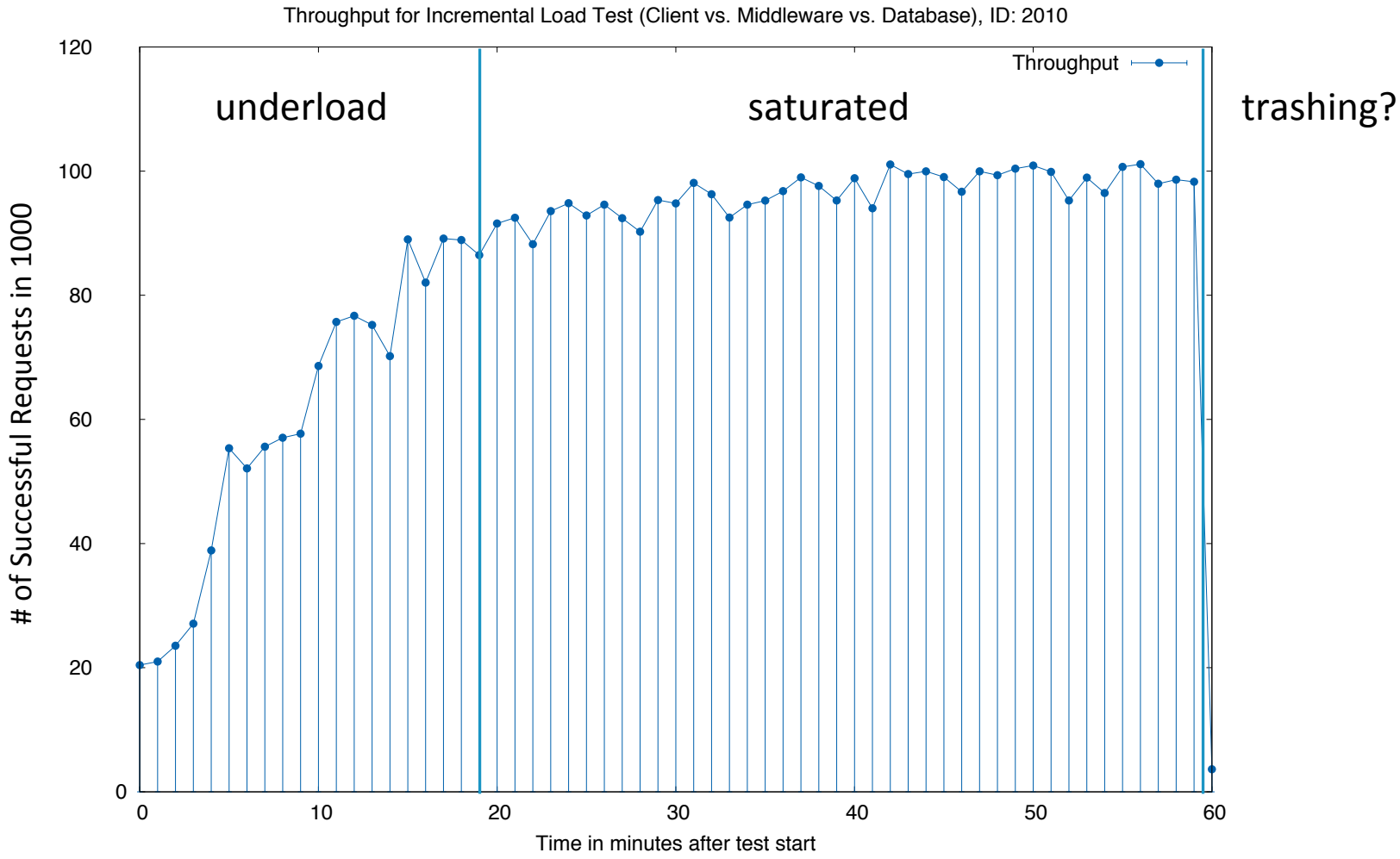


* Interpretation: Higher load ➔
  * Slower database
  * Higher database response time variance

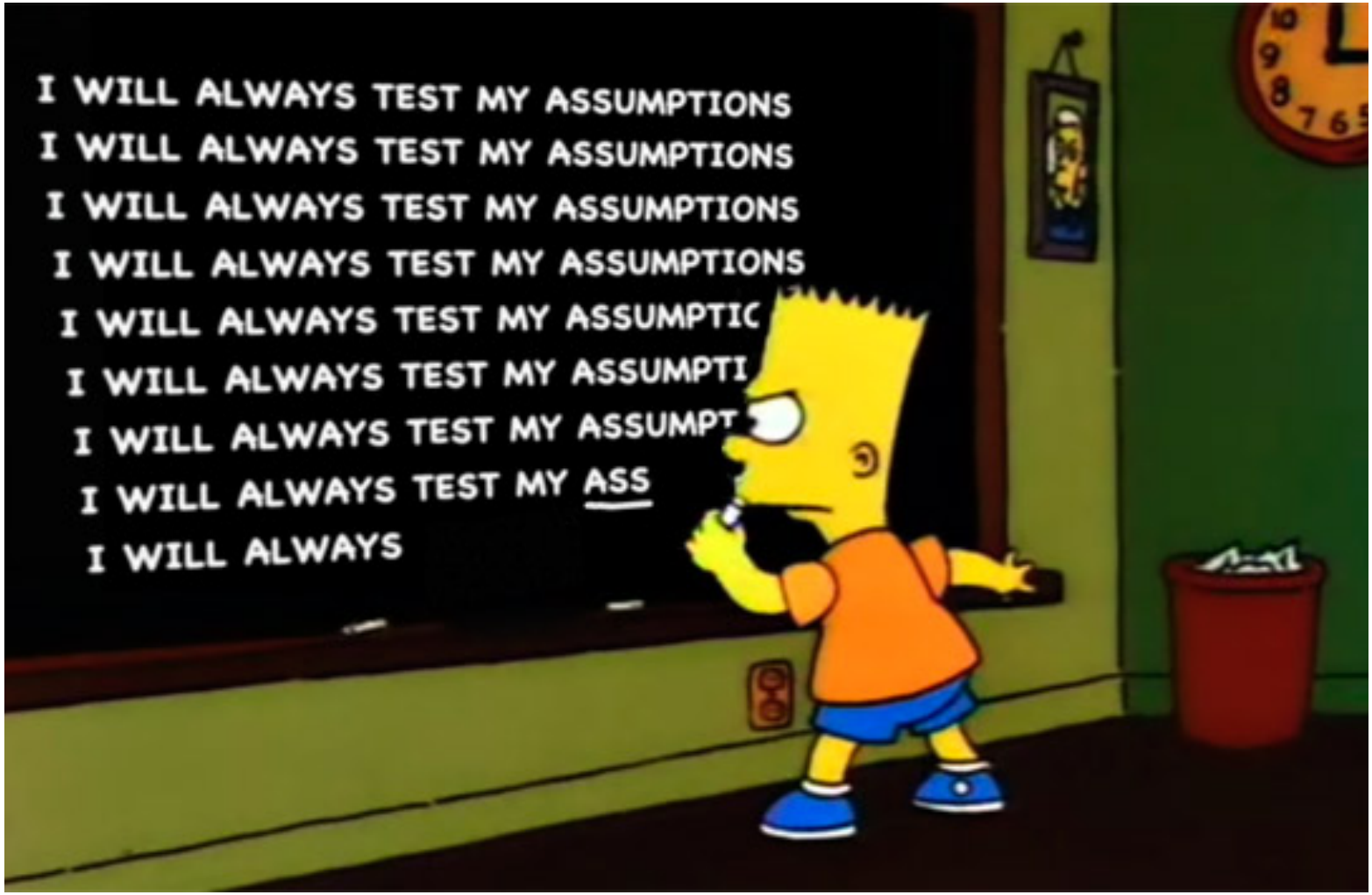
Experiments – Incremental Load Test, Response Time



Experiments – Incremental Load Test, Throughput



Lessons Learned about the System



Lessons Learned about the System

* Improvements during this milestone
  * Response time stability
  * Throughput
* Saturated system
  * => higher throughput
  * => higher variance in response time
* Potential bottleneck
  * Database

Thank you

Questions