Coordinated Omission

in NoSQL Database Benchmarking

Steffen Friedrich, Wolfram Wingerath, Norbert Ritter

University of Hamburg
Department of Informatics
Databases and Information Systems

March 6, 2017

NoSQL Performance Evaluation ?

Cooper et al.:
Benchmarking Cloud Serving Systems with YCSB, SoCC'10, ACM, 2010

YCSBs load generation

```
while (_opsdone < _opcount) {
  long startTime = System.nanoTime();
  Status status = _db.read( table, key, fields, result );
  long endTime = System.nanoTime();

  _measurements.measure("READ", (int)( (endTime - startTime) / 1000));




}
```

YCSBs load generation

```
_targetOpsTickNanos = (long) (1 000 000 000 / targetThroughput)
long overallStartTime = System.nanoTime();

while (_opsdone < _opcount) {




  _opsdone++;

  long deadline = overallStartTime + _opsdone * _targetOpsTickNanos;
  long now = System.nanoTime();
  while((now = System.nanoTime()) < deadline) {
    LockSupport.parkNanos( deadline - now );
  }
}
```

YCSBs load generation

```
_targetOpsTickNanos = (long) (1 000 000 000 / targetThroughput)
long overallStartTime = System.nanoTime();

while (_opsdone < _opcount) {
```

What if

latency > _targetOpsTickNanos ?

now > deadline ?

```
  _opsdone++;

  long deadline = overallStartTime + _opsdone * _targetOpsTickNanos;
  long now = System.nanoTime();
  while((now = System.nanoTime()) < deadline) {
    LockSupport.parkNanos( deadline - now );
  }
}
```

The Coordinated Omission Problem

“a conspiracy we’re all a part of”

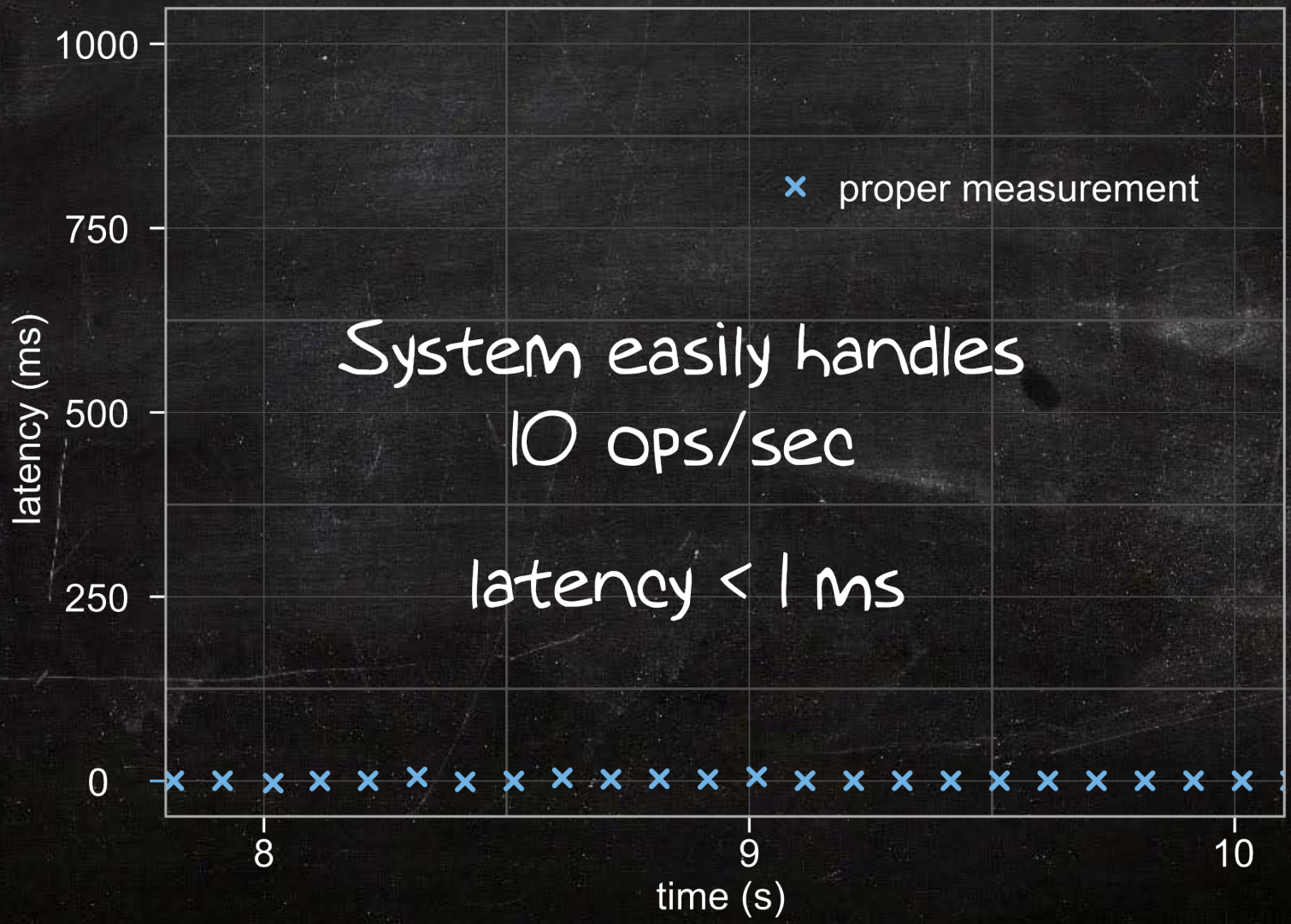
Gil Tene, Azul Systems:

How NOT to Measure Latency, QCon, 2013 - 2016

infoq.com/presentations/latency-response-time

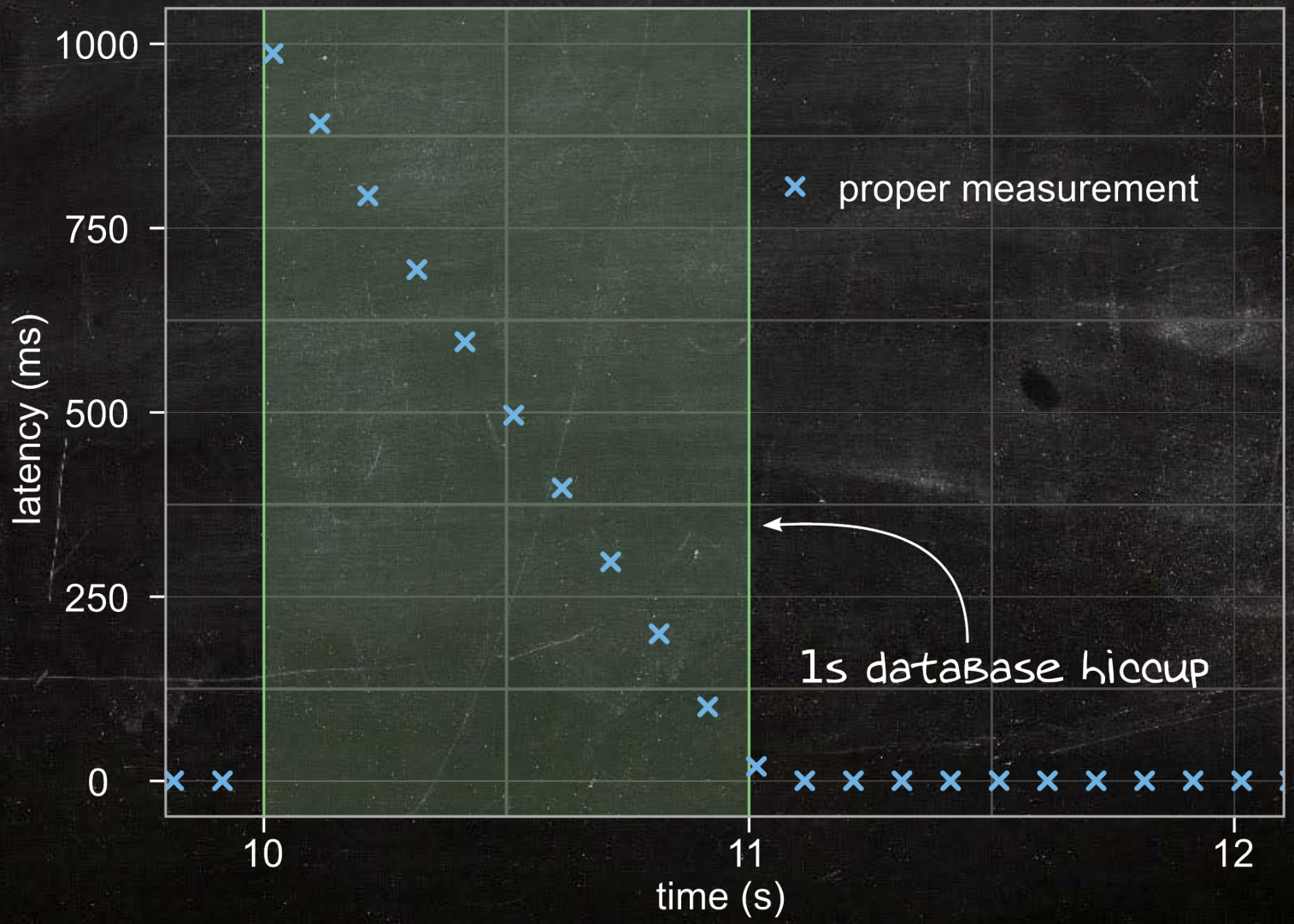
The Coordinated Omission Problem

Example I



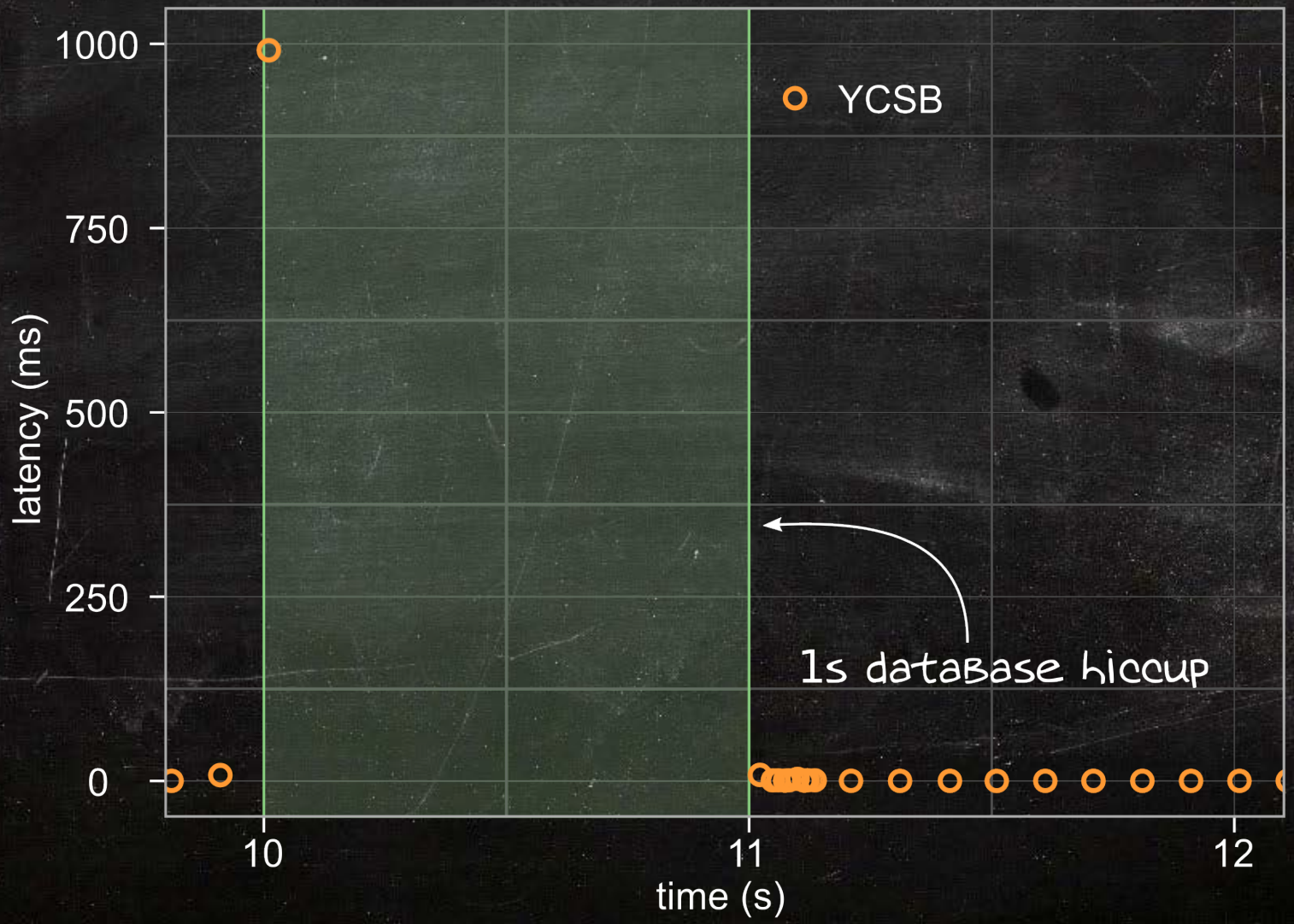
The Coordinated Omission Problem

Example II



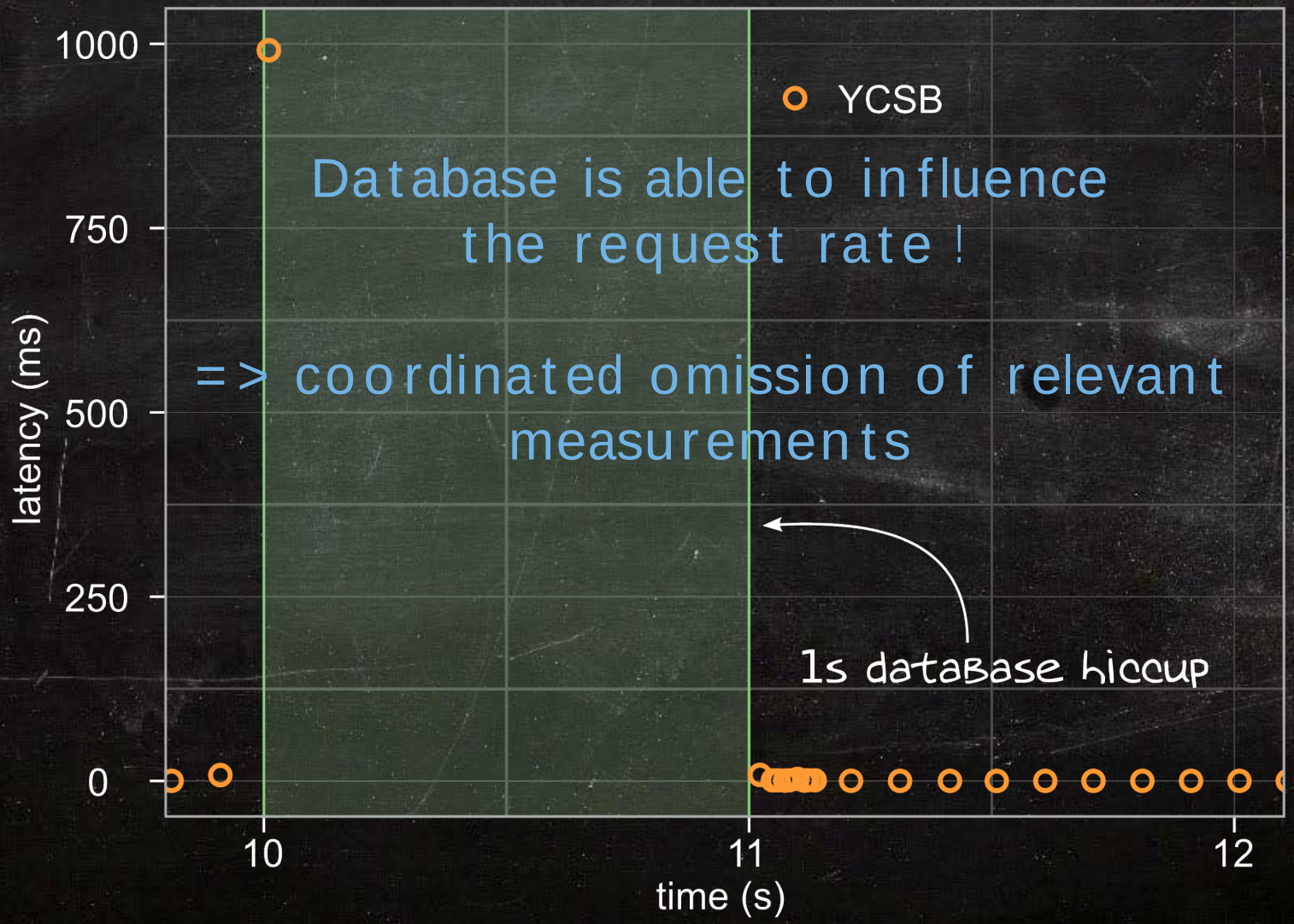
The Coordinated Omission Problem

Example III



The Coordinated Omission Problem

Example III



The Coordinated Omission Problem

Example

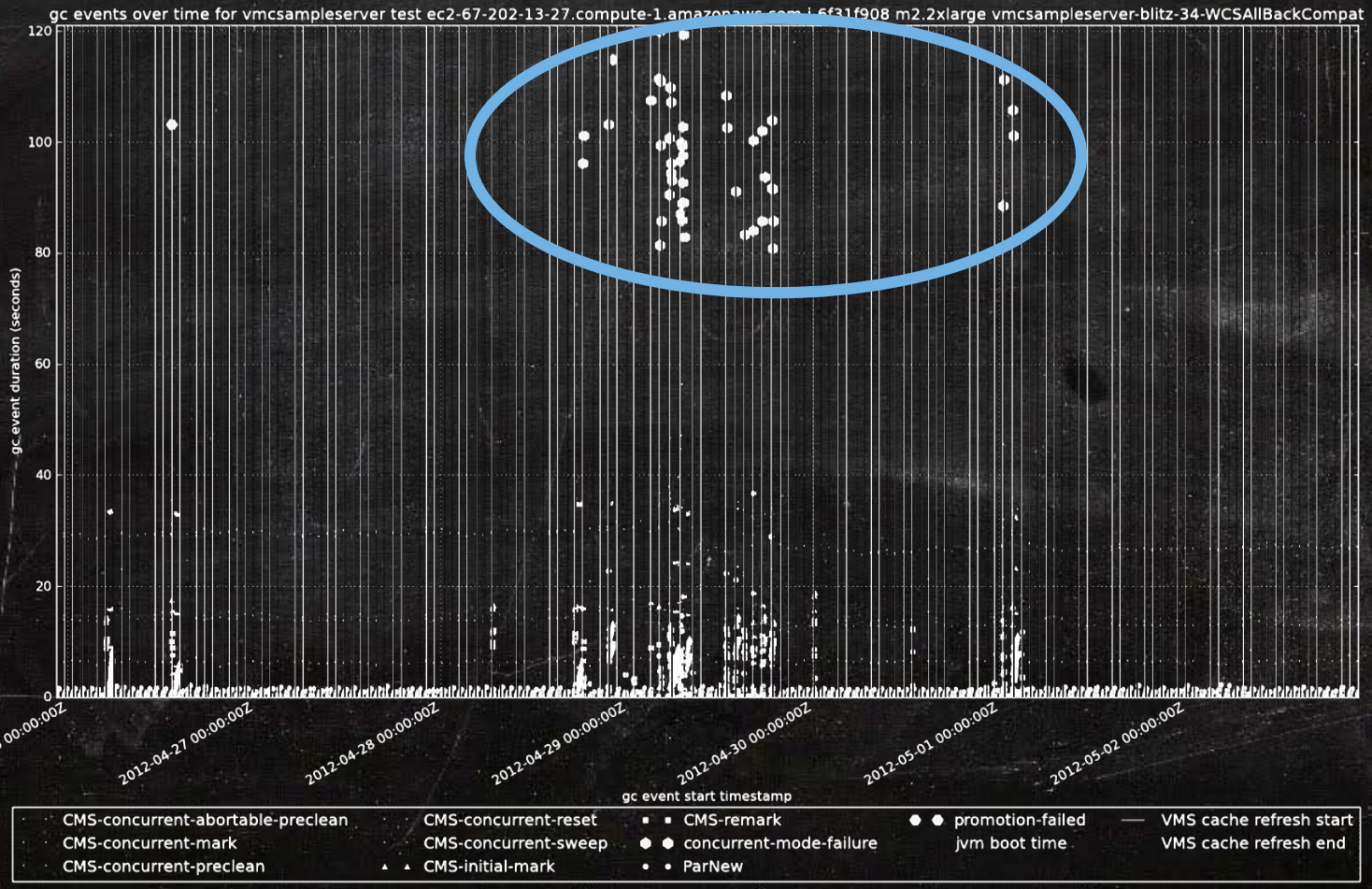
The Results:

	AVG.	90%ile	99%ile	Max
No Hiccup	0.92	1.133	1.649	8.423
Hiccup	17.43	7.539	603.647	903.679
Hiccup YCSB	4.39	4.711	6.599	902.143

The Coordinated Omission Problem

Real World **GC Event Chart**

80-120 seconds GC pauses @ Netflix Cassandra Cluster 2012

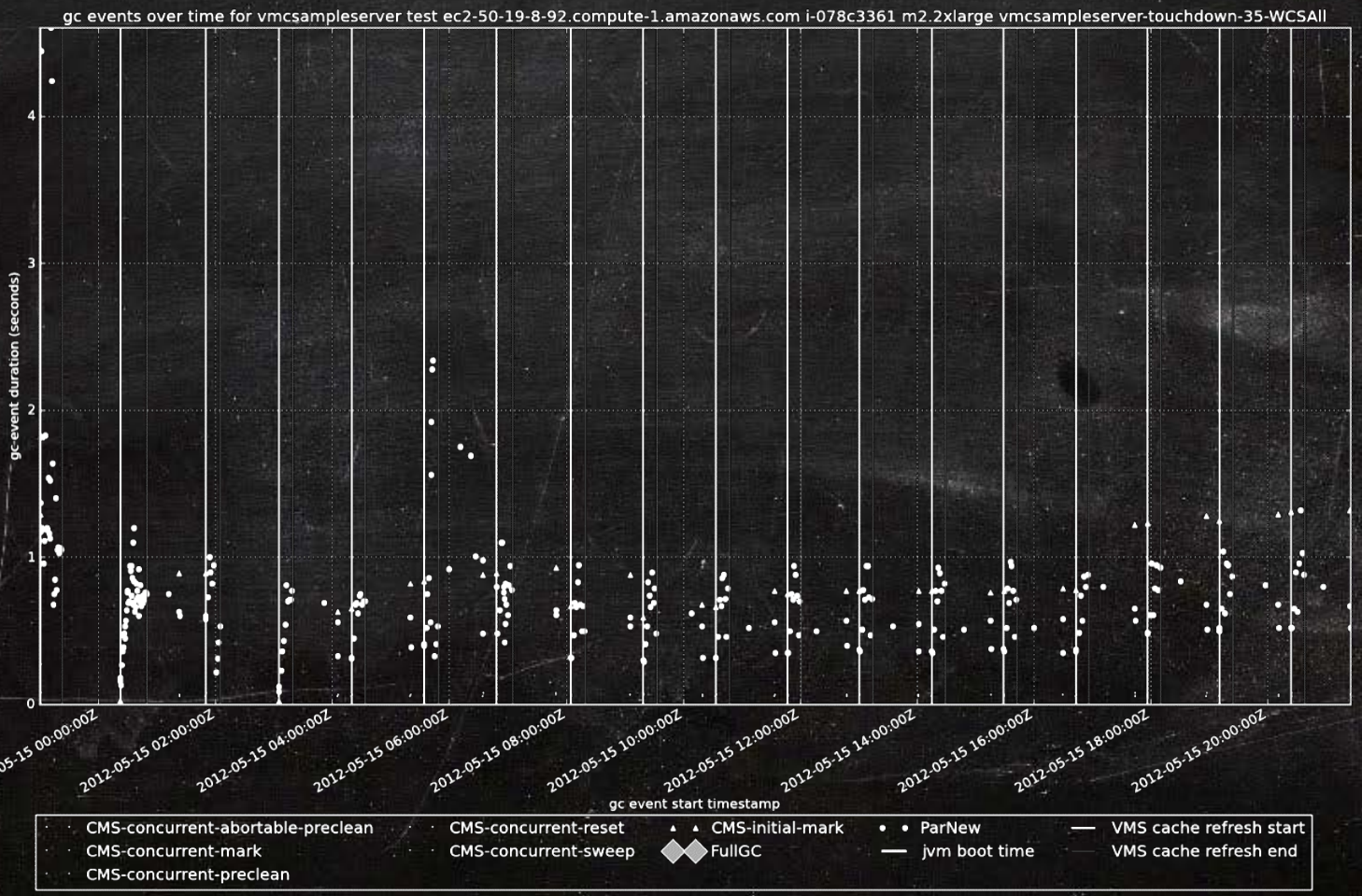


http://techblog.netflix.com/2013/05/garbage-collection-visualization.html

The Coordinated Omission Problem

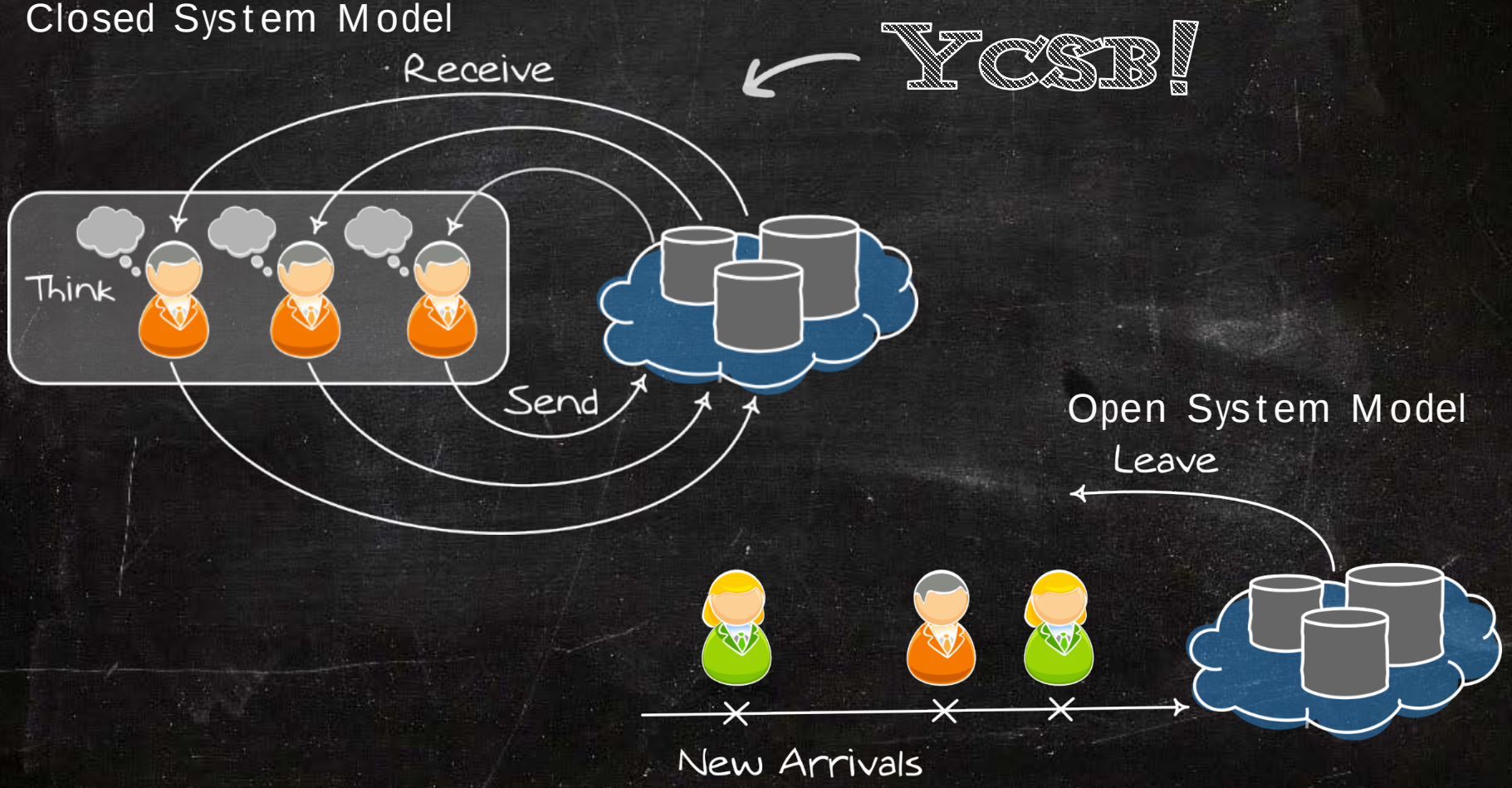
Real World **GC Event Chart**

After fixing the problem: max 5 seconds GC pauses



http://techblog.netflix.com/2013/05/garbage-collection-visualization.html

Closed VS. Open System Model



Schröder et al. Open Versus Closed: A Cautionary Tale, 2006

Closed VS. Open System Model

Schröder et al. surveyed system models in various web related workload generators:

- "Most of these generators/benchmarks assume a closed system model"
- "For many of these workload generators, it was quite difficult to figure out which system model was being assumed"
- "Principle (i): For a given load, mean response times are significantly lower in closed systems than in open systems."

Coordinated Omission Correction

since YCSB Version 0.2.0 RC 1, June 2015


```
while (_opsdone < _opcount) {
```

=> intended measurement interval

```
  _measurements.measure("INTENDED_READ", (int)( (endTime - _deadline) / 1000));

  _opsdone++;

  _deadline = overallStartTime + _opsdone * _targetOpsTickNanos;

                              ...

}
```

Coordinated Omission Correction

since YCSB Version 0.2.0 RC 1, June 2015


```
while (_opsdone < _opcount) {
```

=> intended measurement interval

```
  _measurements.measure("INTENDED_READ", (int)( (endTime - _deadline) / 1000));

  _opsdone++;

  _deadline = overallStartTime + _opsdone * _targetOpsTickNanos;
```

=> but still influences the request rate !

```
}
```


N SQL MARK
Scalable NoSQL-Benchmarking
nosqlmark.informatik.uni-hamburg.de
YCSB!
Scala
akka

Scalable NoSQL-Benchmarking

nosqlmark.informatik.uni-hamburg.de

- Scaling YCSB compatible workloads to multiple benchmarking nodes

  => Automatically aggregated results
- Compatible to YCSB database interface layer
- Closed and Open System Model

Coordinated Omission Avoidance
in NoSQLMark

```
implicit val ec = context.system.dispatchers.lookup("blocking-io-dispatcher")

case DoOperation => {
  val operation = workload.nextOperation
  val startTime = System.nanoTime
  val future = Future {
    sendRequest(operation)
  }
  future.onComplete {
    case Success(status) => {
      val endTime = System.nanoTime
      measurementActor ! Measure(operation.name, (endTime - startTime) / 1000)
    }
    case Failure(ex) => {
      log.error(ex, "Error occured during operation {}", operation.name)
    }
  }
  ...
```

asynchronous load generation !

Coordinated Omission Validation with

Single-node inconsistent key-value Store

Originally developed to validate consitency measurement approaches

- consistent single-node backend
- multi-node behaviour
- tunable staleness
- globaly consistent logfile

github.com/steffenfriedrich/SickStore

Wingerath, Friedrich, Gesser, Ritter:
Who Watches the Watchmen? BTW 2015

Coordinated Omission Validation with

Single-node inconsistent key-value Store

New Feature:
Simulation of maximum throughput and database hiccups

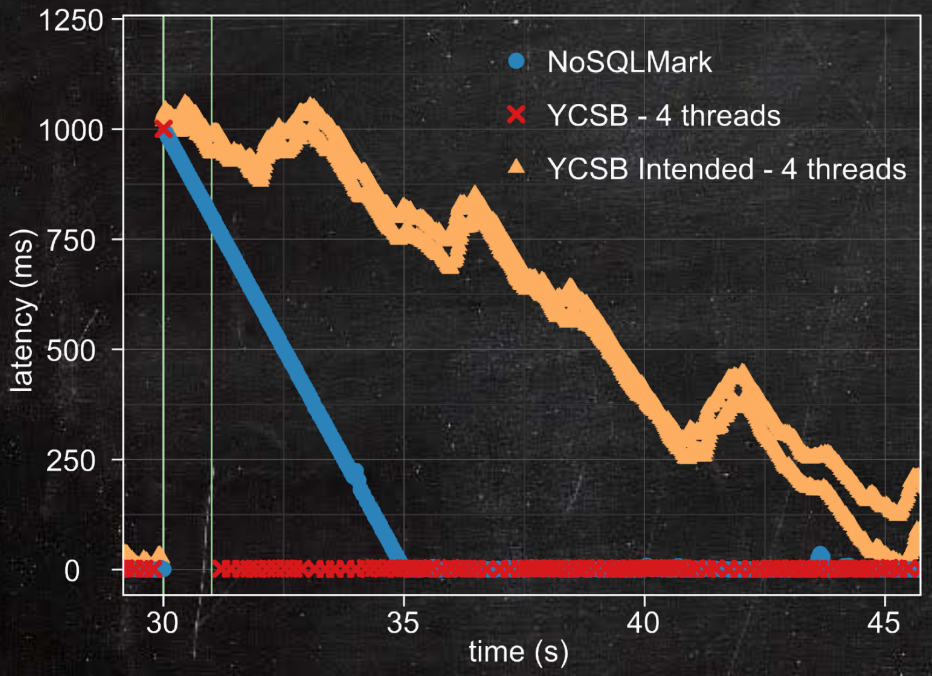
1. compute theoretical waiting time T_i of request i in the databse system
2. calling client thread has to sleep for T_i

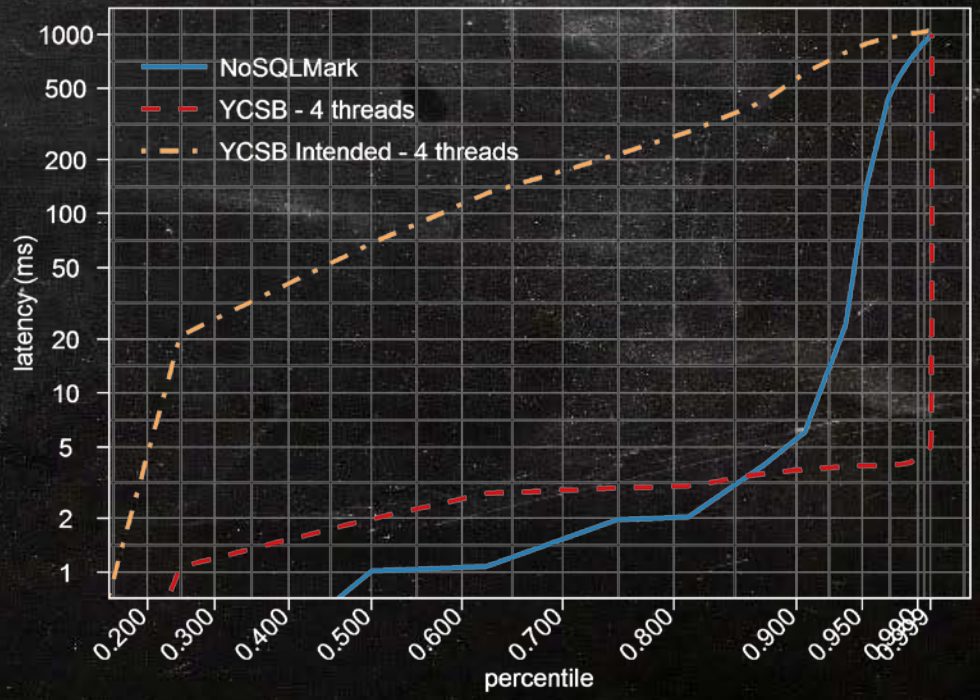
Experimental Validation: SickStore

Benchmark: 90 000 ops, target = 1000 ops/sec,

SickStore: 1 second hiccup, max throughput = 1250 ops/sec,

80% of max throughput



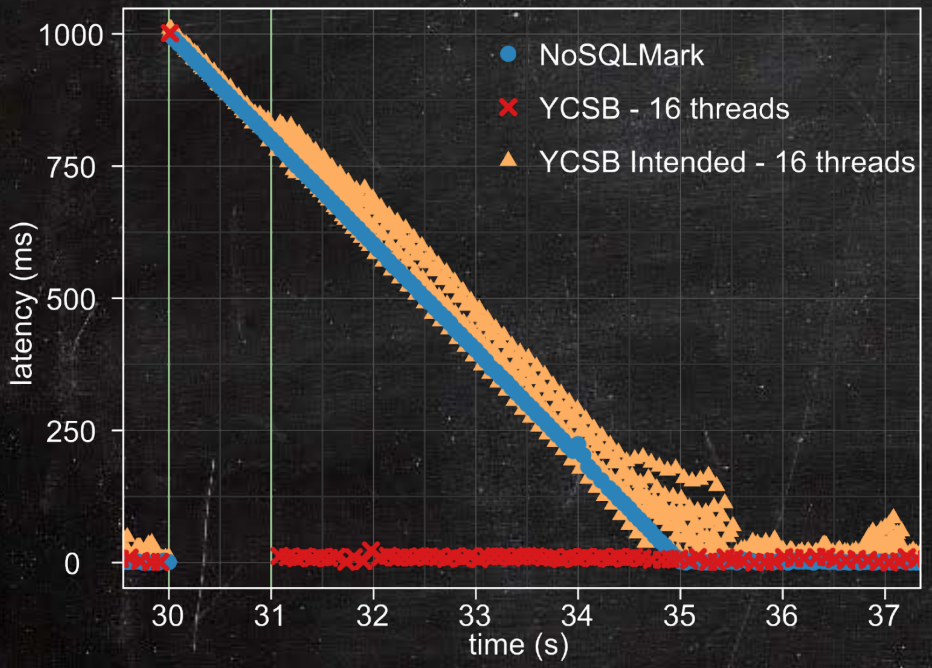


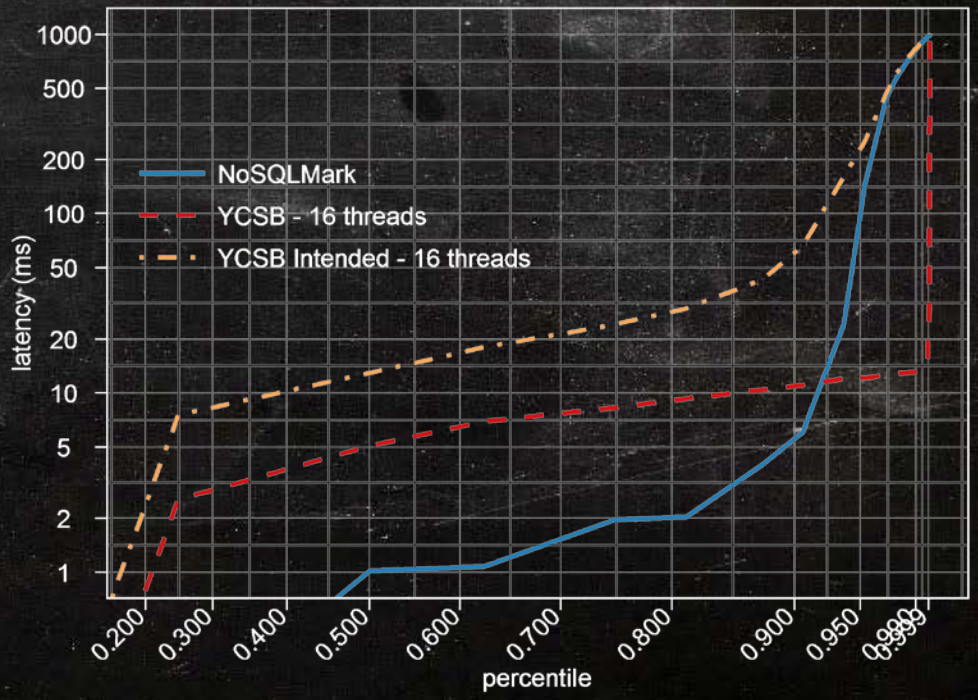
	YCSB	NoSQLMark	YCSB Intended
AVG.:	2 ms	29 ms	180 ms

Experimental Validation: SickStore

Benchmark: 90 000 ops, target = 1000 ops/sec,

SickStore: 1 second hiccup, max throughput = 1250 ops/sec,



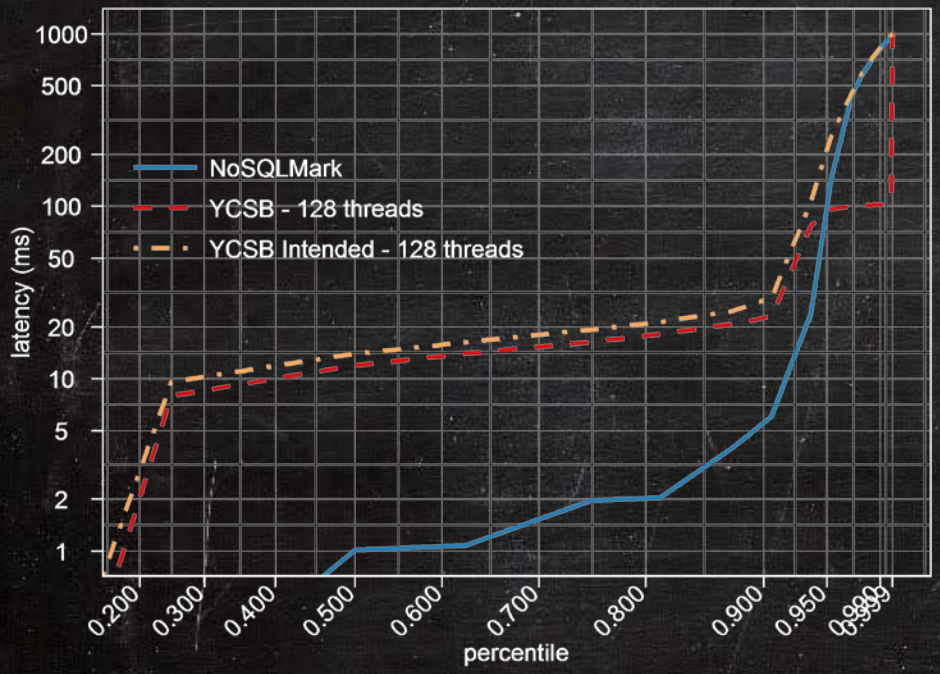


	YCSB	NoSQLMark	YCSB Intended
AVG.:	6 ms	29 ms	49ms

Experimental Validation: SickStore

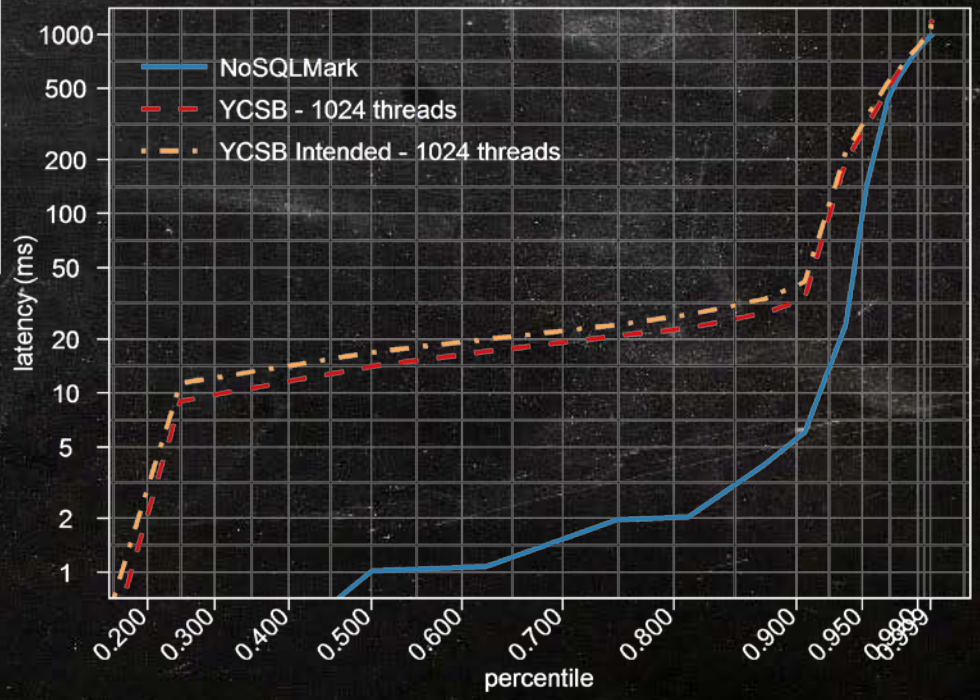
Benchmark: 90 000 ops, target = 1000 ops/sec,

SickStore: 1 second hiccup, max throughput = 1250 ops/sec,



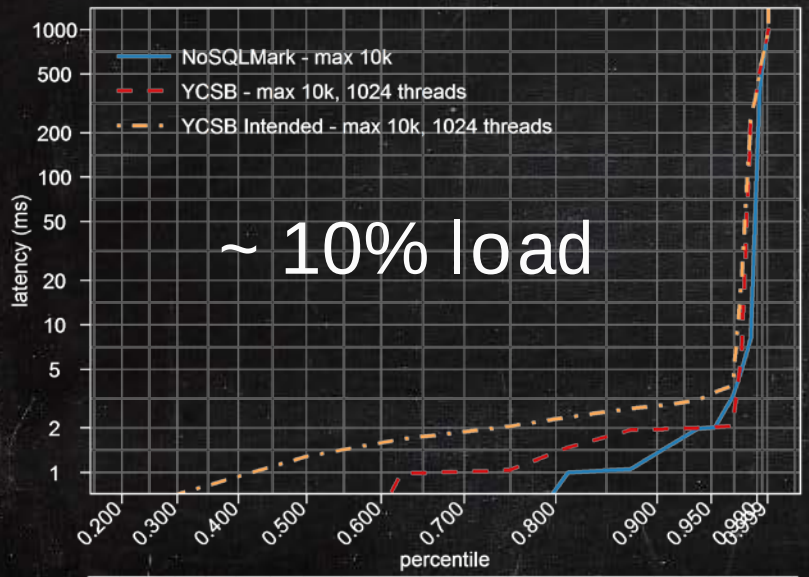
	YCSB	NoSQLMark	YCSB Intended
AVG.:	19 ms	29 ms	44ms

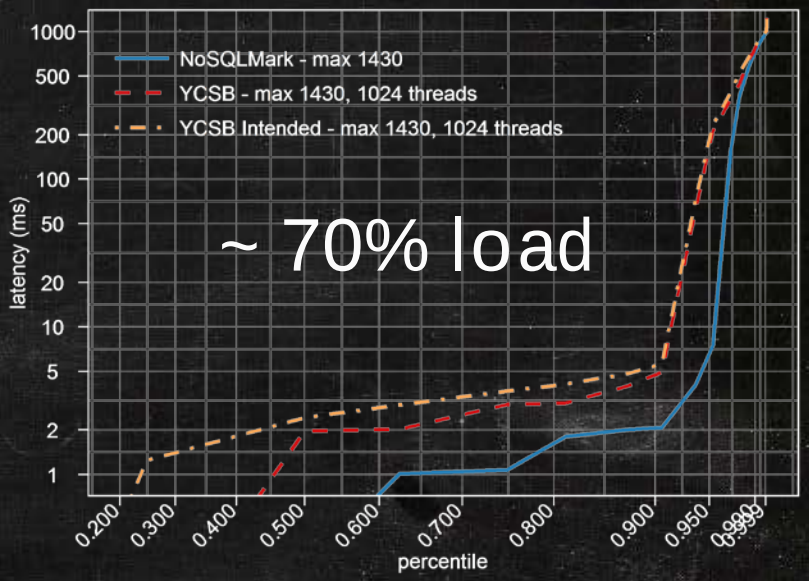
	YCSB	NoSQLMark	YCSB Intended
AVG.:	49 ms	29 ms	54ms

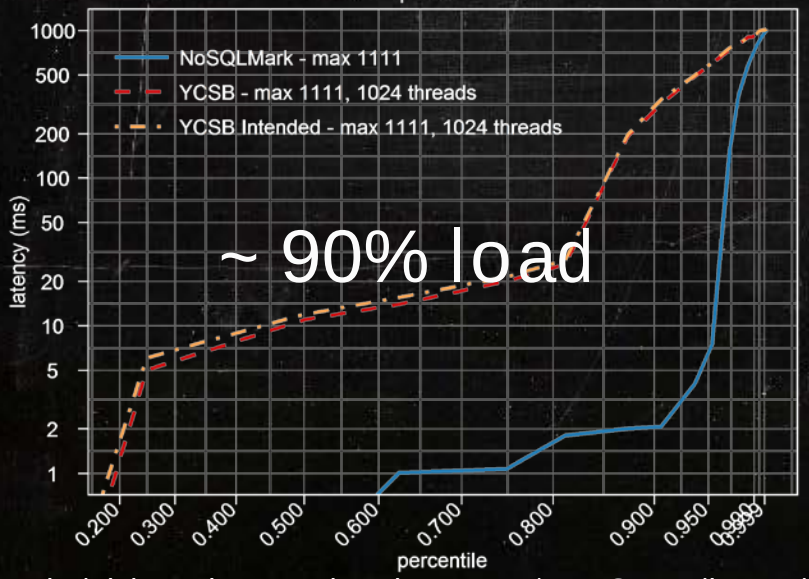


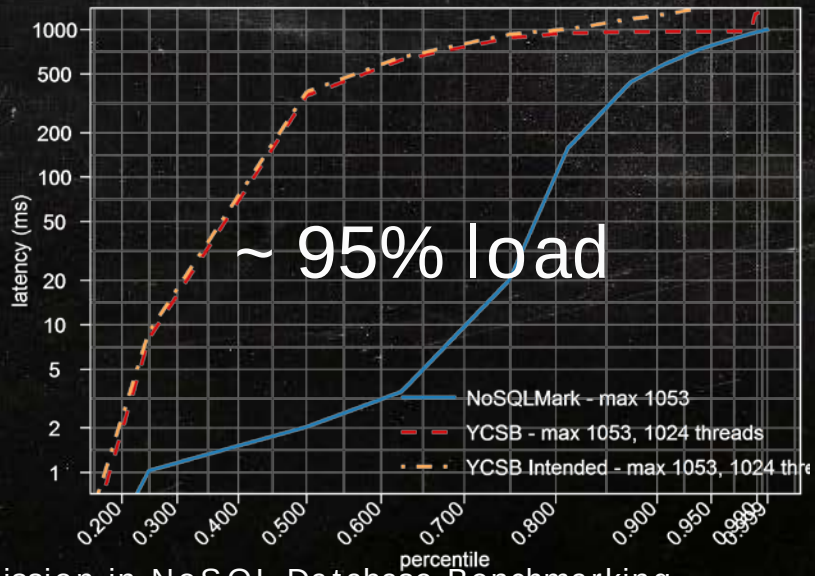
Experimental Validation: SickStore

Different max throughputs









Elasticity Benchmark with Cassandra

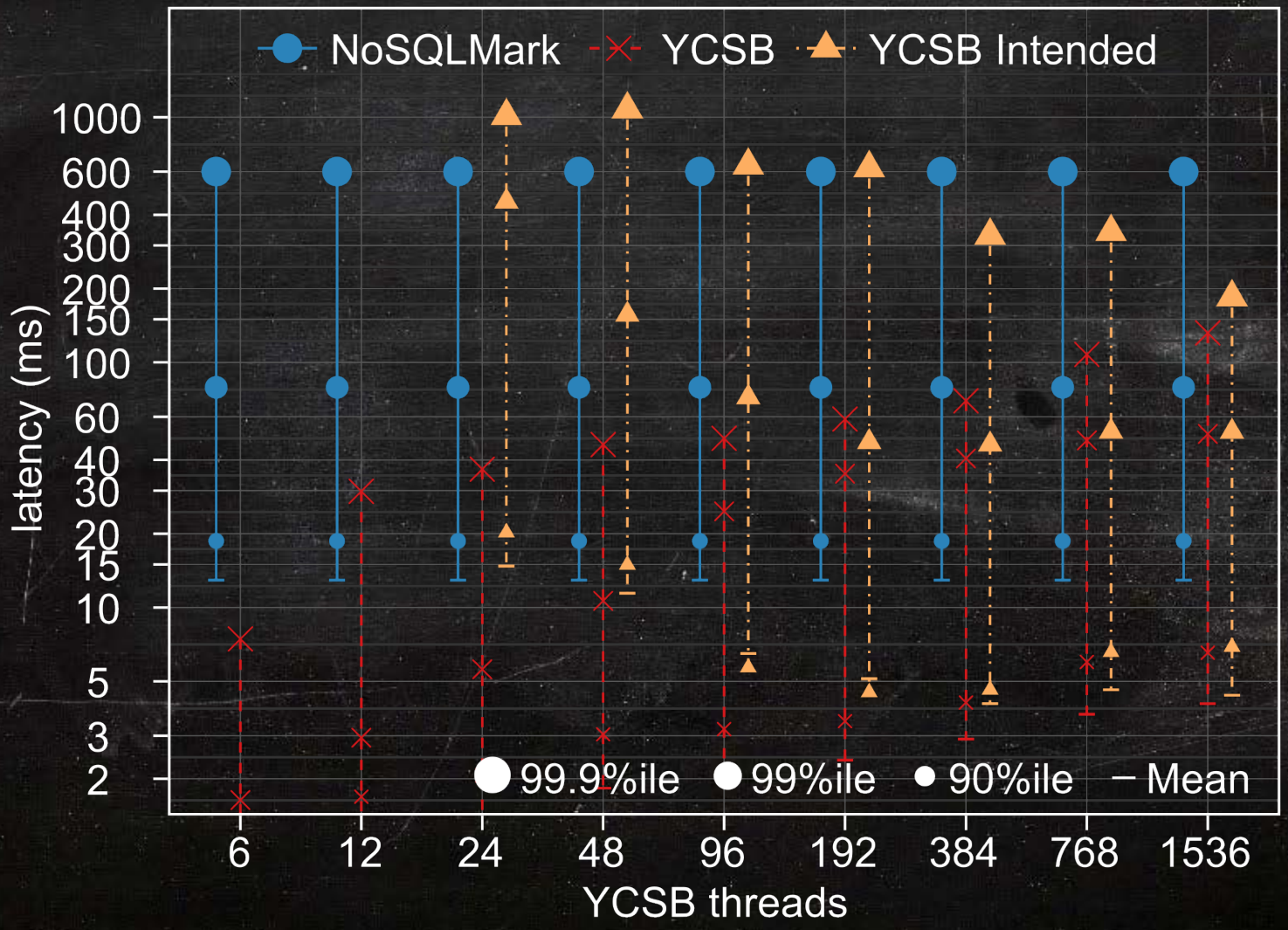
- one Cassandra node loaded with 10 million records
- after 5 min add second node
  => it starts serving after ~ 5 min
  => roughly the time it takes latency to stabilize
- run each experiment for max 15 min on a fresh Cluster

YCSB without intended measurement interval

Kuhlenkamp et al.: Benchmarking Scalability and Elasticity of Distributed Database Systems, VLDB, 2014

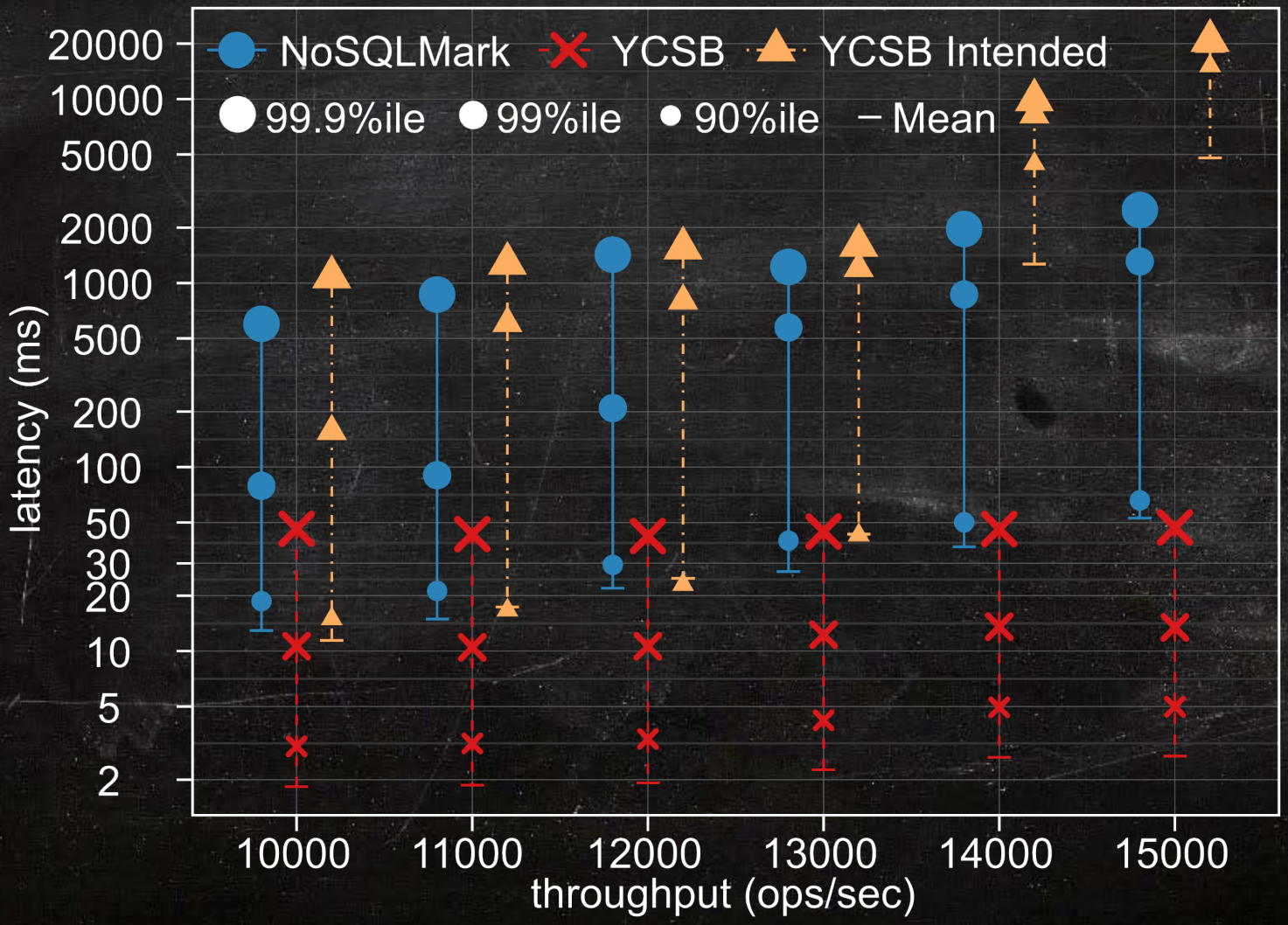
Elasticity Benchmark with Cassandra

target throughput = 10 000 ops/sec



Elasticity Benchmark with Cassandra

YCSB: 48 threads



Summary

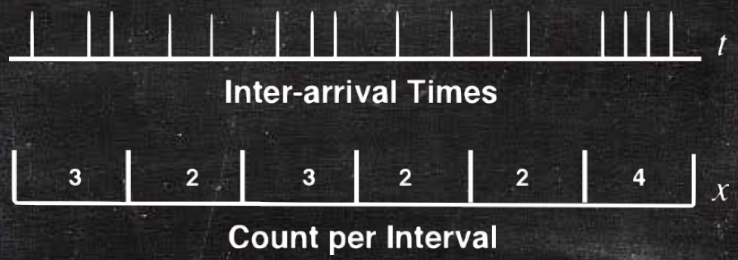
Benchmarking is hard and your latency values are probably lying to you!

Be aware of the system model underlying your load generators.

Further Improvement in NoSQLMark

More realistic distributions for request rate

- Exponential inter-request/arrival time => poisson distributed request/arrival rate



- Some authors consider Perato or hyper-exponential distributed inter-arrival time

James F. Brady & Neil J. Gunther: How to Emulate Web Traffic Using Standard Load Testing Tools, CoRR, 2016

Neil J. Gunther: **Load Testing Think Time Distributions, blogpost, 2010**
perfdynamics.blogspot.de/2010/05/load-testing-think-time-distributions.html