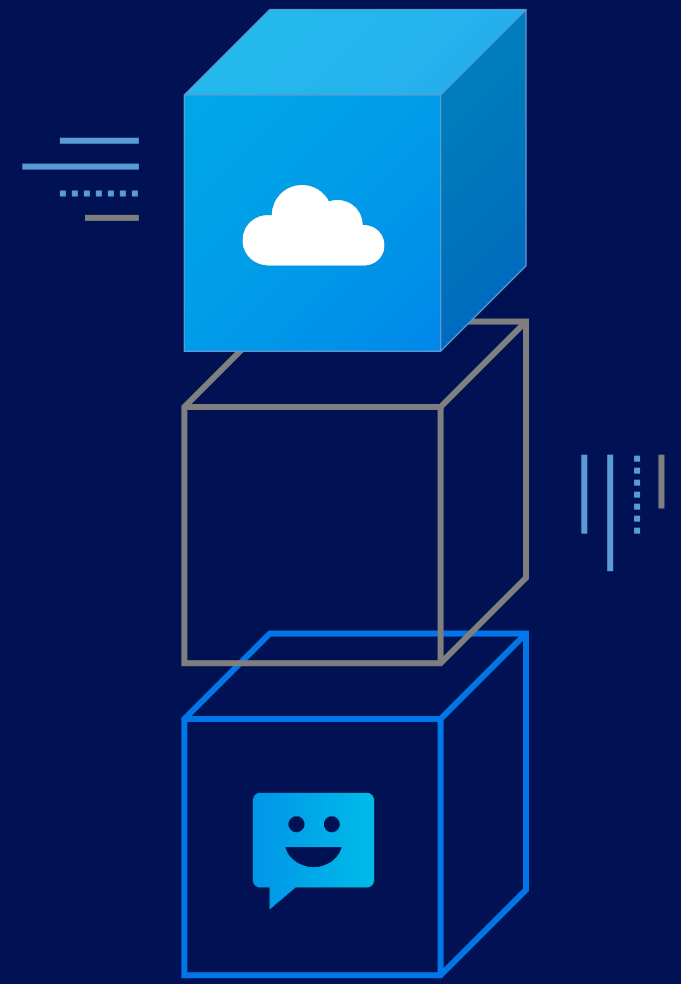
instaclustr

The Impact of Hardware and Software Version Changes on Apache Kafka® Performance and Scalability

ApacheCon NA Performance Engineering Track 2022



Paul Brebner, Hendra Gunadi, and more!
Instaclustr by NetApp

Who Am I?

instaclustr

Previously

- R&D in distributed systems and performance engineering

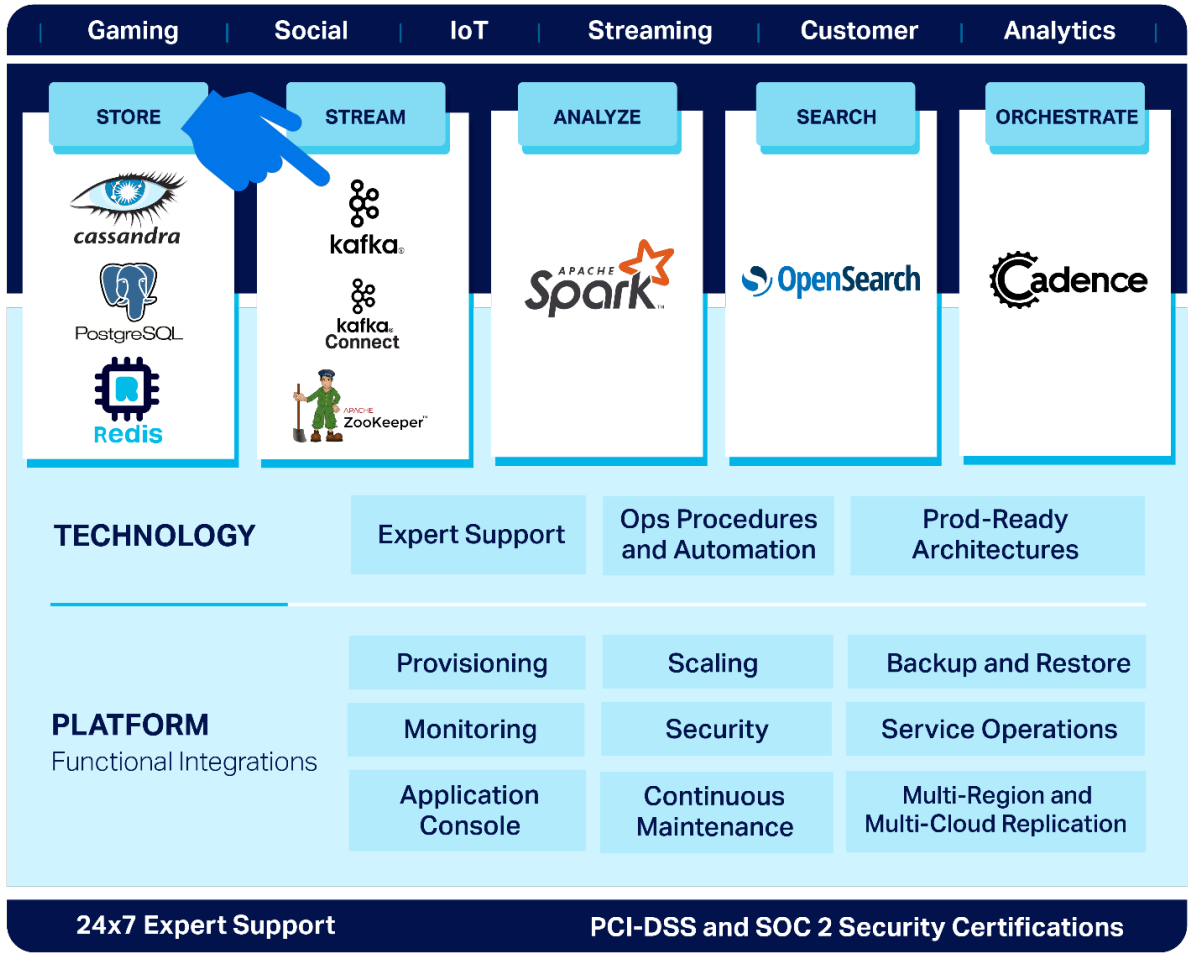
Last 5 years

- Technology Evangelist for Instaclustr by NetApp
- 100+ Blogs, Demo Applications, Talks
- Benchmarking and Performance Insights
- Open Source Technologies including
  - Apache Cassandra®, Spark™, ZooKeeper, Kafka®
  - OpenSearch®, Redis™, PostgreSQL®
  - Uber's Cadence®

Instaclustr Managed Platform

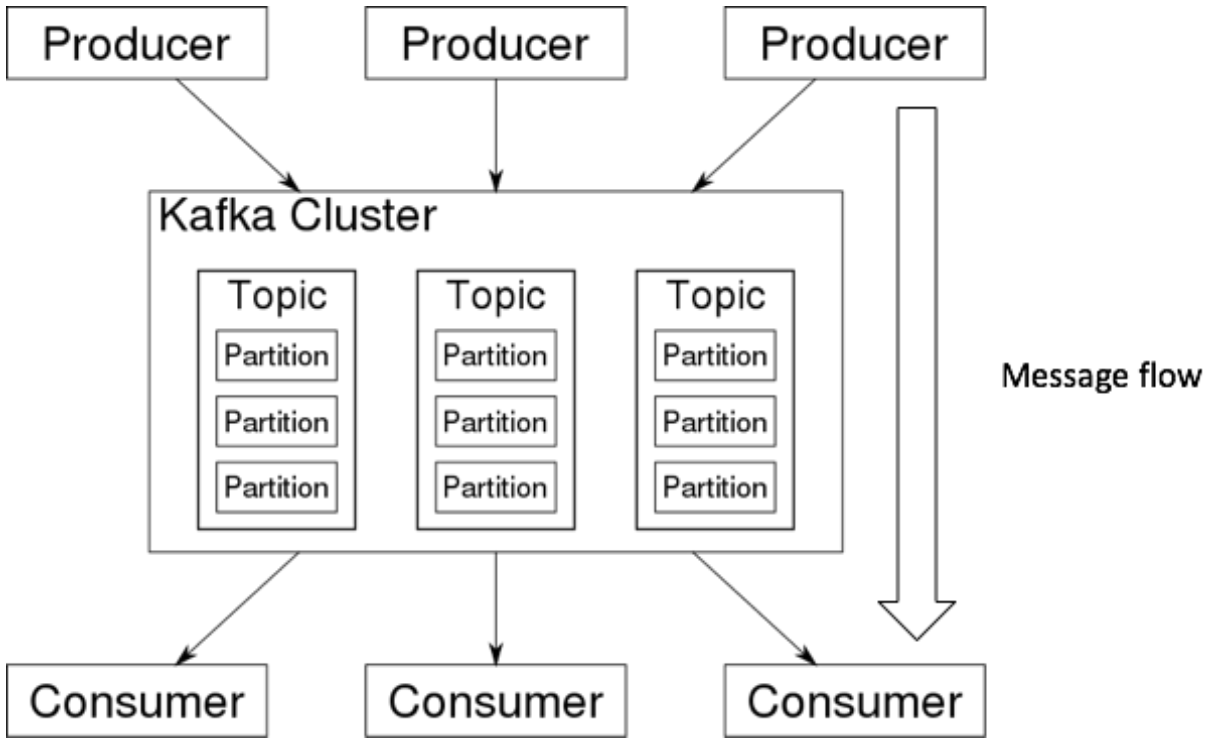
Cloud Platform for Big Data Open Source Technologies

Focus of this talk is on Apache Kafka®



What is Kafka?

Kafka is a distributed streams processing system, it allows distributed producers to send messages to distributed consumers via a Kafka cluster.

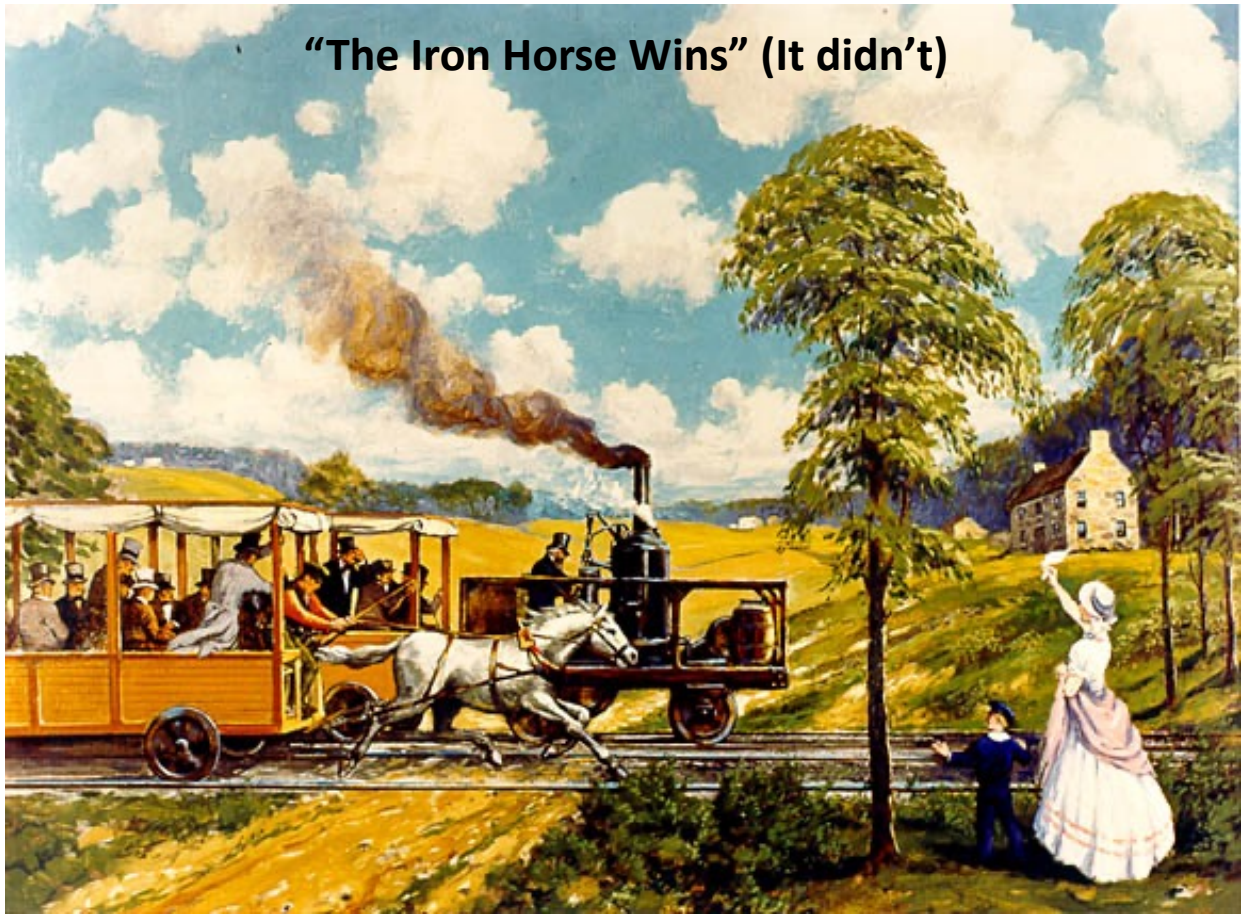


And Change:
Hardware and Software

instaclustr

Some changes have “obvious” performance impacts – e.g. Horse → Steam

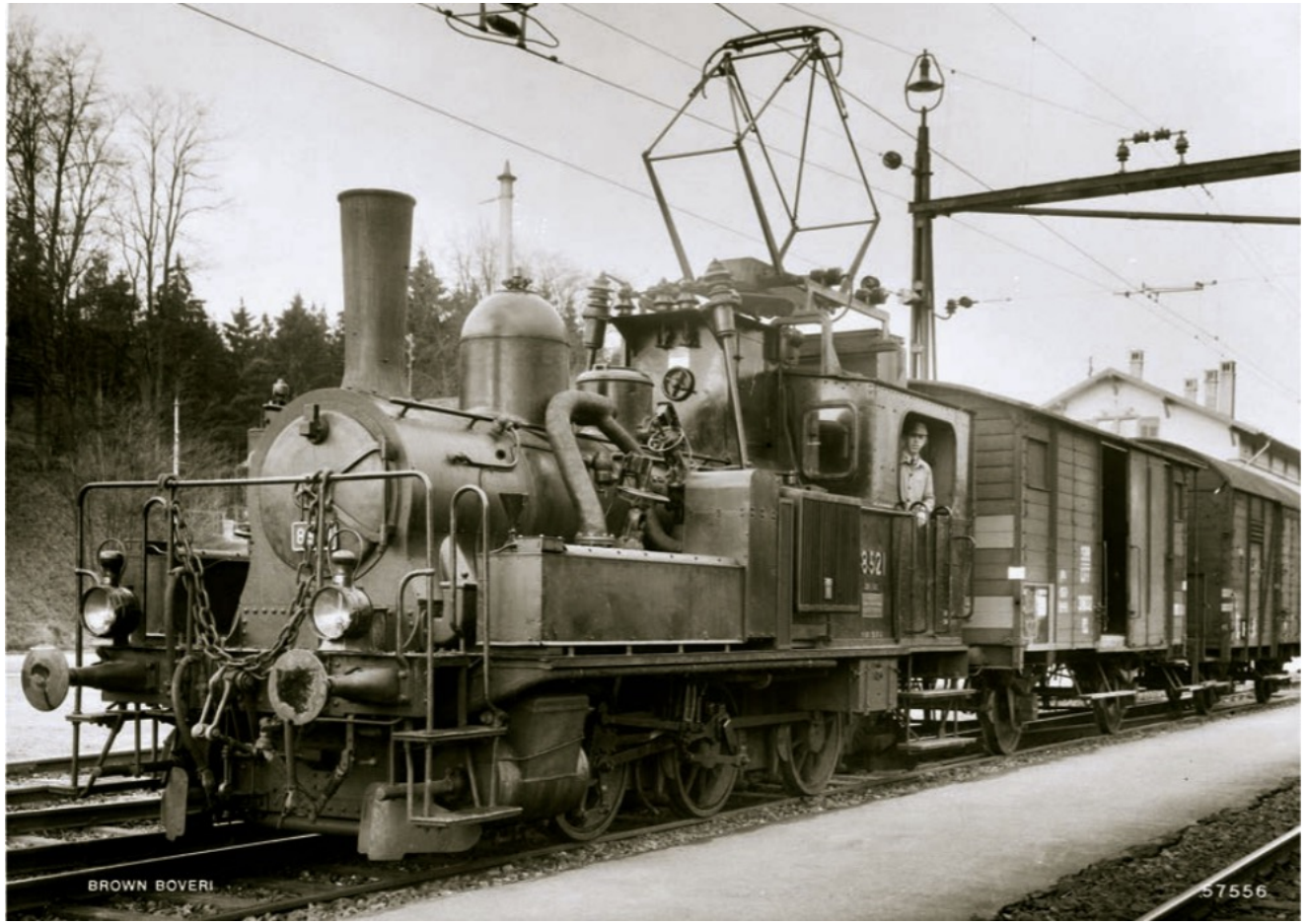
instaclustr



“The Iron Horse Wins” (!) Race in 1830 between steam locomotive and a horse, horse won (due to mechanical failure), but was obviously inferior. Source: https://www.fhwa.dot.gov/rakeman/1830.htm

Others are not so obvious – e.g. Electric + Steam Locomotive

instaclustr



https://en.wikipedia.org/wiki/Electric-steam_locomotive#/media/File:SBB_Ee_3-3_8521.png

Part 1: Hardware Change

(Source: Shutterstock)

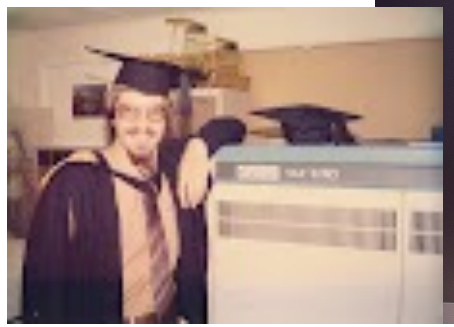
instaclustr

Hardware Change: CISC

VAX 11/780

CISC = Complex Instruction Set Computer

University of Waikato NZ
1980-85

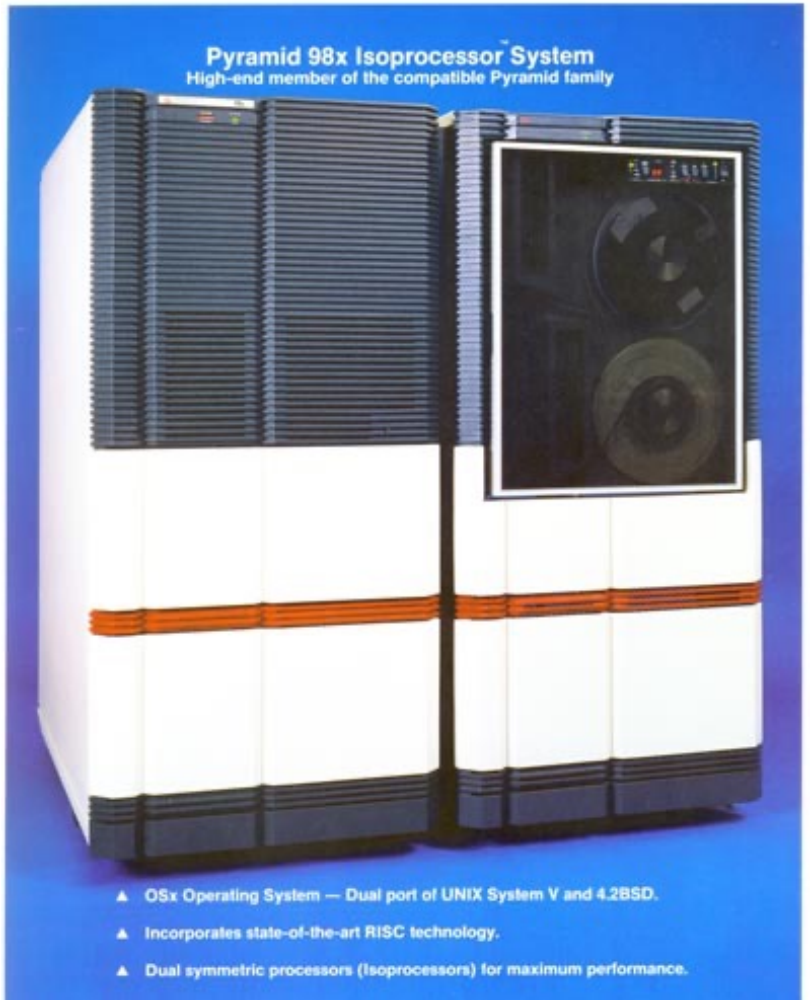


CISC to RISC

Pyramid Technology

RISC = Reduced Instruction Set Computer

UNSW Sydney Australia
2nd half of 1980s



More Recently: Intel PC

https://commons.wikimedia.org/wiki/File:HP_OMEN_X_900_Gaming_Desktop_PC.jpg

Intel PC to iPhone?!

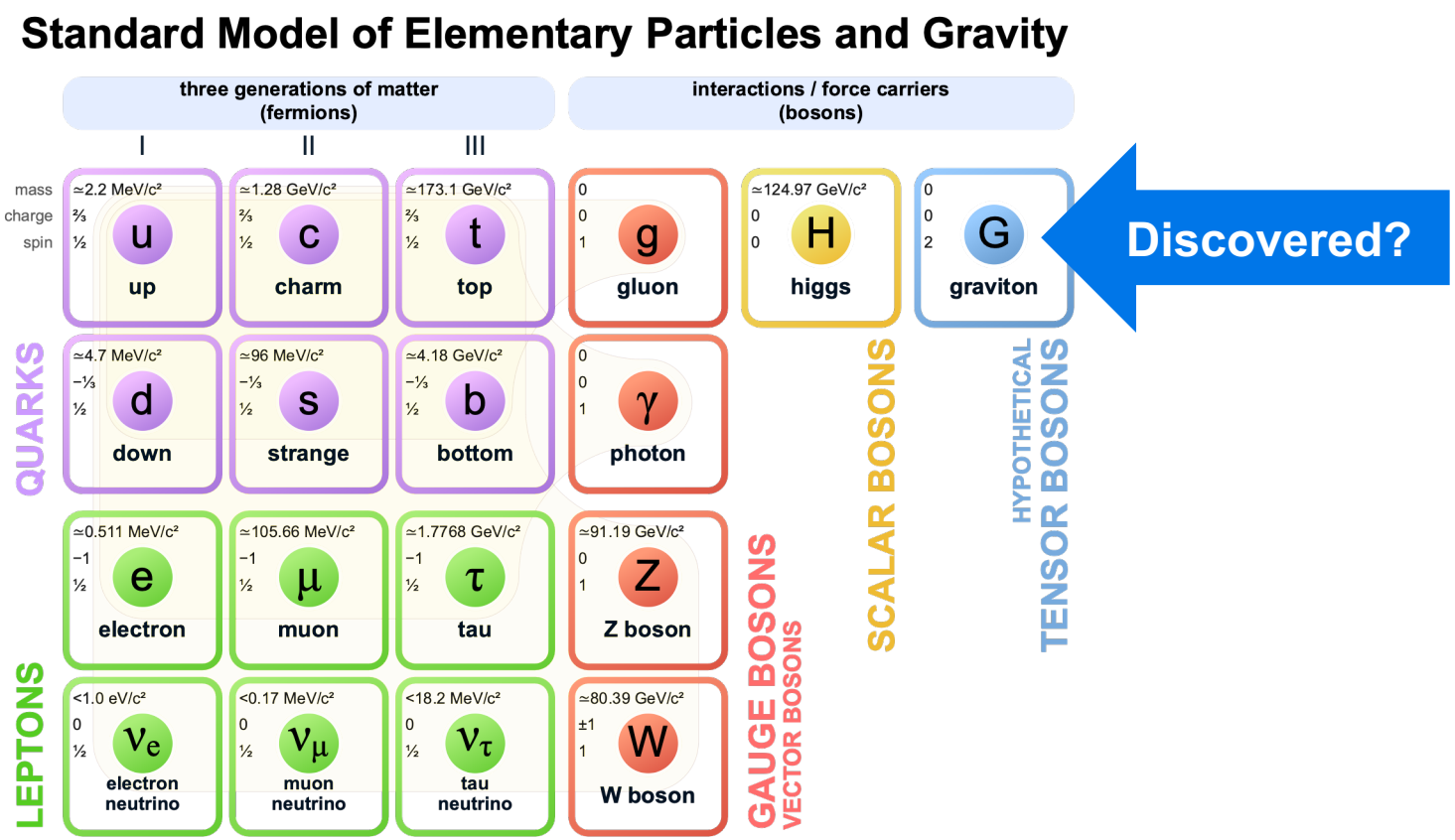
https://commons.wikimedia.org/wiki/File:IPhone_12_Pro_Max_-_3.jpg

Acorn BBC Micro Computer: 1980s

https://www.classic-computers.org.nz/collection/BBCb-1920x.jpg

Fast Forward 40 Years: Have *Gravitons* Been Discovered?

instaclustr



https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles_%2B_Gravity.svg

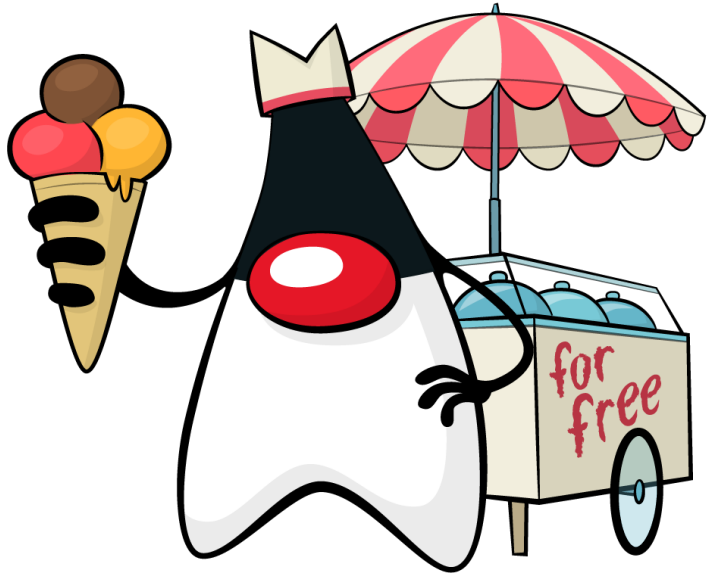
Fast Forward 40 Years to the AWS Graviton2

instaclustr

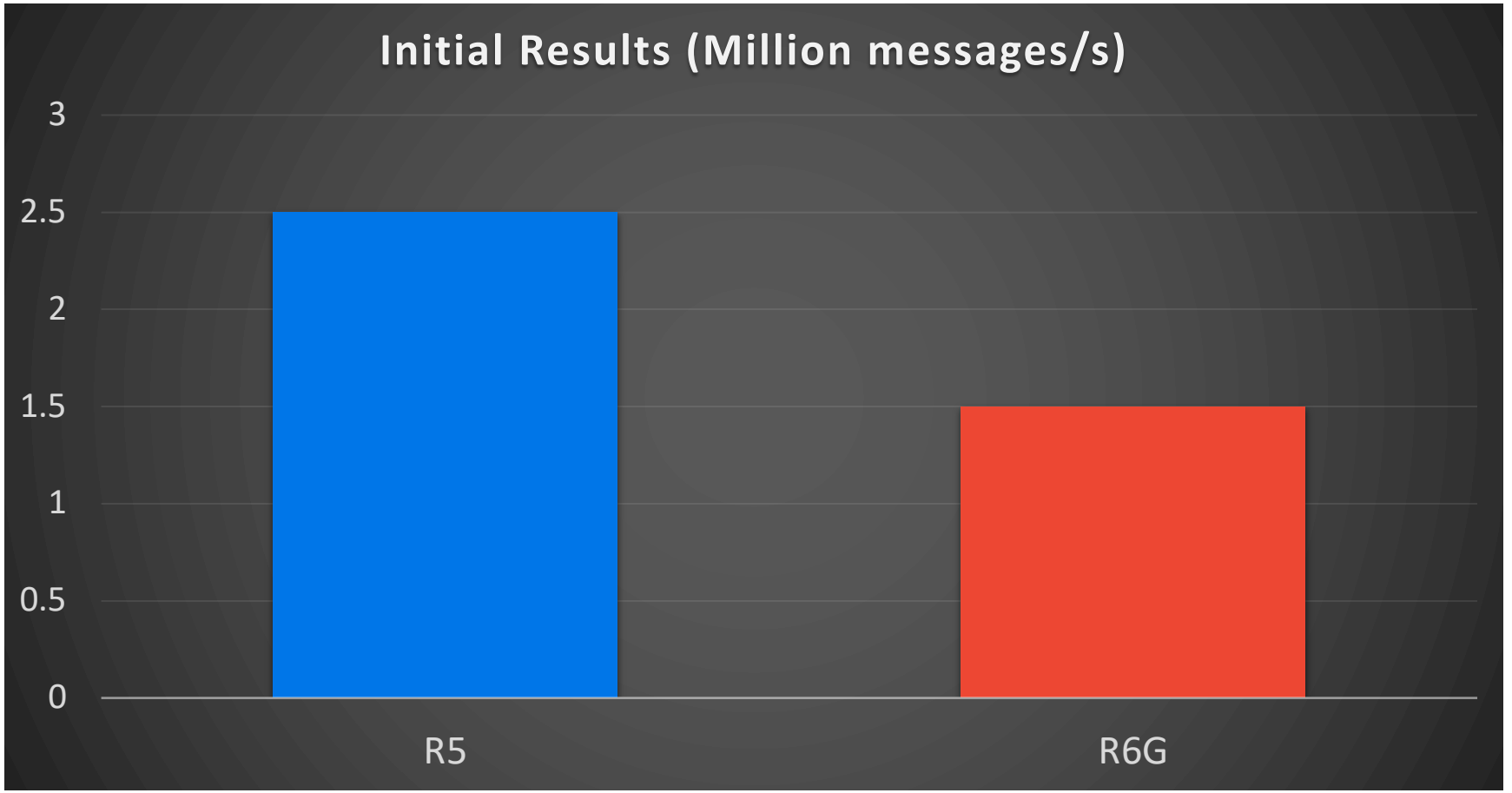
- **RISC for Servers!**
  - ARM-based New AWS instance types
  - Designed by ARM, formerly **Advanced RISC Machines** and originally **Acorn RISC** Machine
    - made the BBC micro
- **Real Cores**
  - 64 Cores, 256GB RAM per CPU
  - No hyperthreading
  - Each vCPU = 1 physical core
- **Benchmarking**
  - Reported to be up to 40% faster than Intel and AMD
- **Less Power Consumption**
  - So Cheaper and Faster

Initial Benchmarking

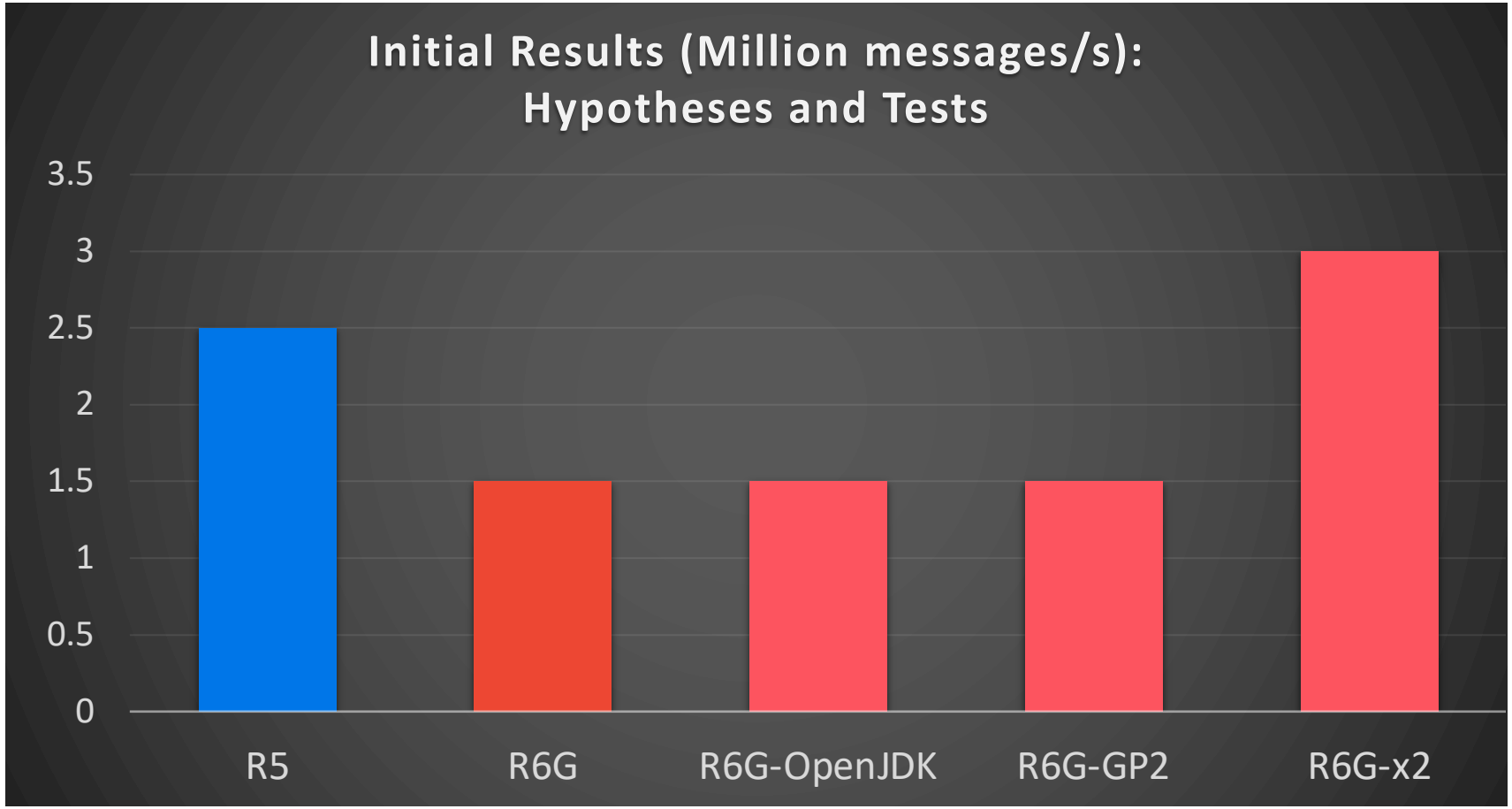
- **Apache Kafka® deployed on**
- **R5 (Intel) vs. R6g (Graviton2) instances**
- **New R6g configuration:**
  - AWS Gp3 disks
  - Java 11 OpenJDK → Amazon Corretto
  - Client → Broker encryption enabled
- **Hoping for easy and large performance gains…**



Initial Results: 40% Worse!

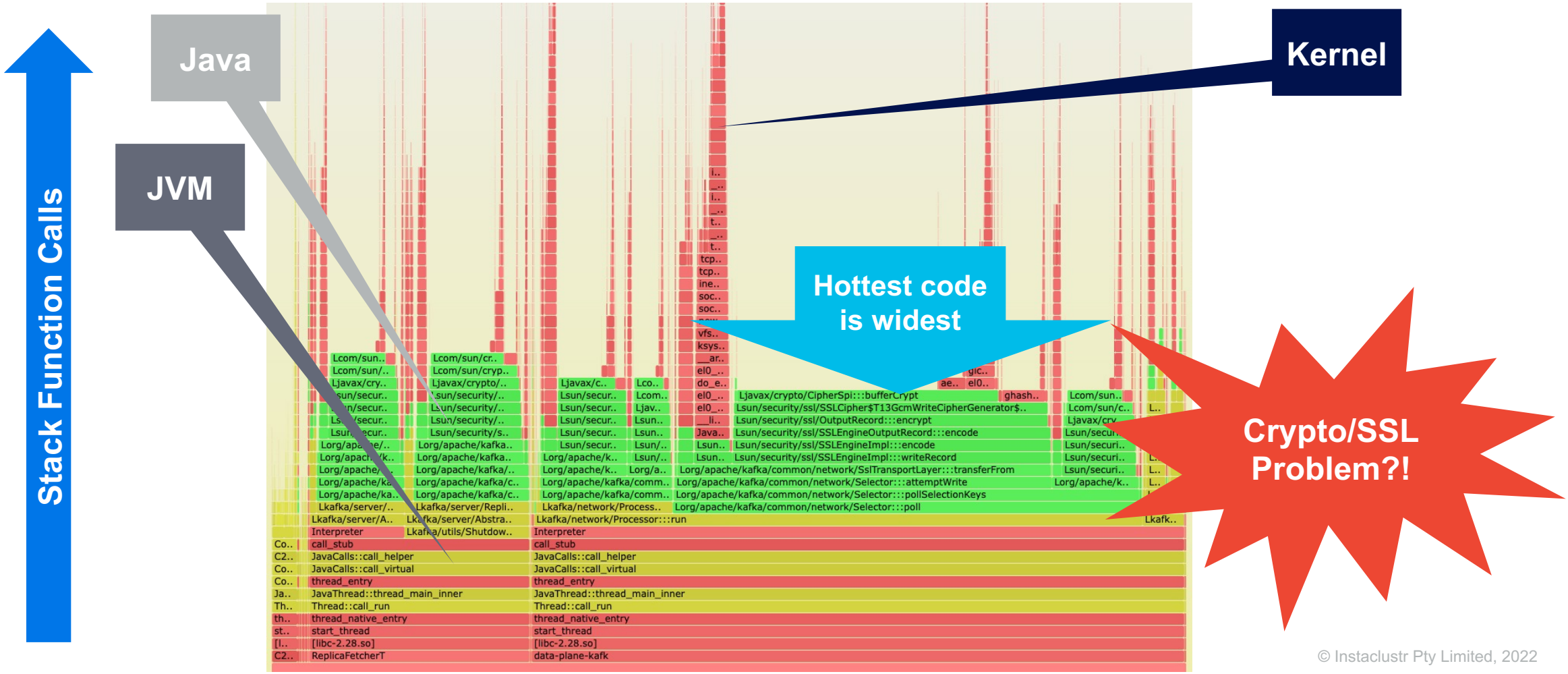


Why? Hypotheses and Tests



Deep Dive: CPU Profiling with Flame Graphs

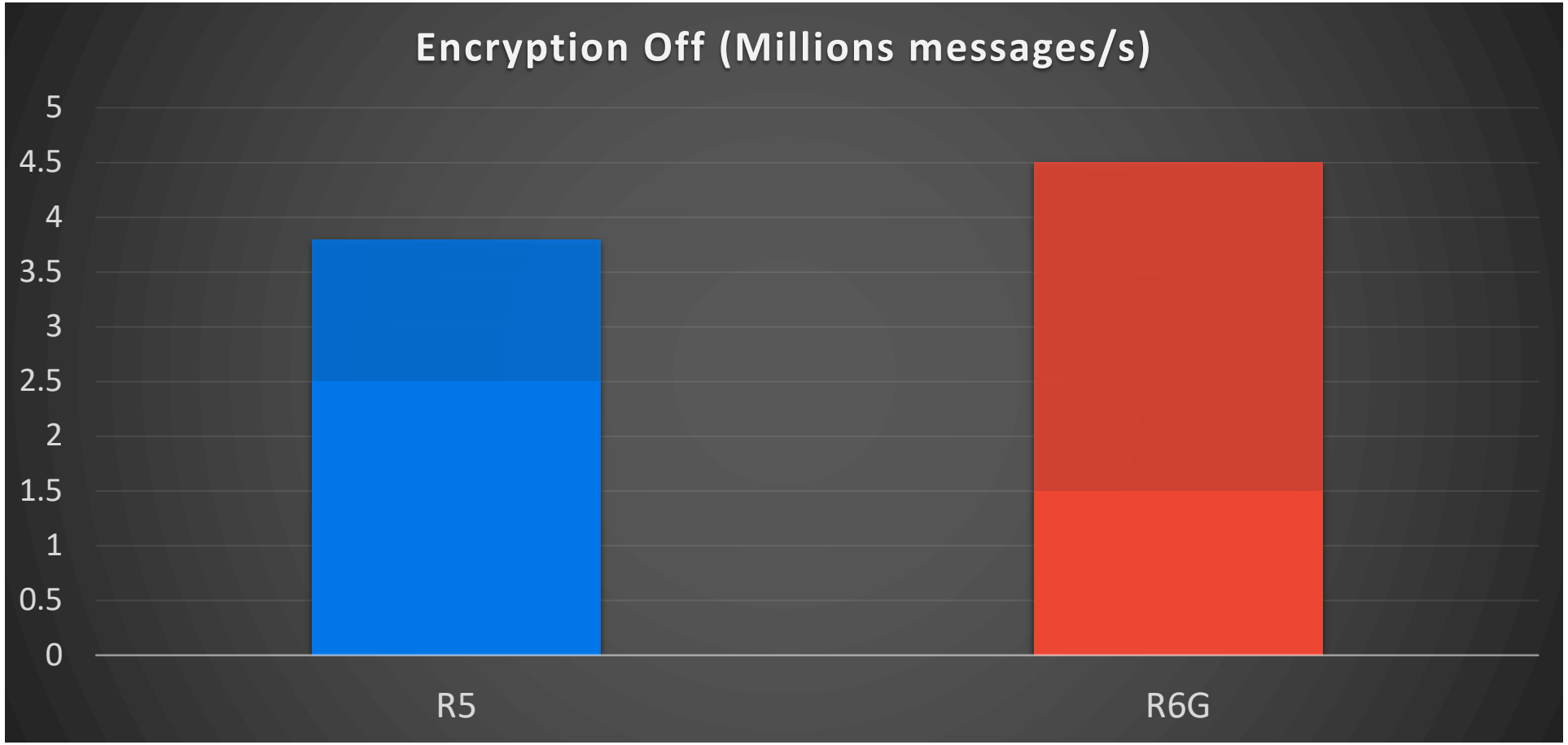
instaclustr



Encryption Off: Better!

200% improvement, 18% better than R5

instaclustr



Solution? ACCP

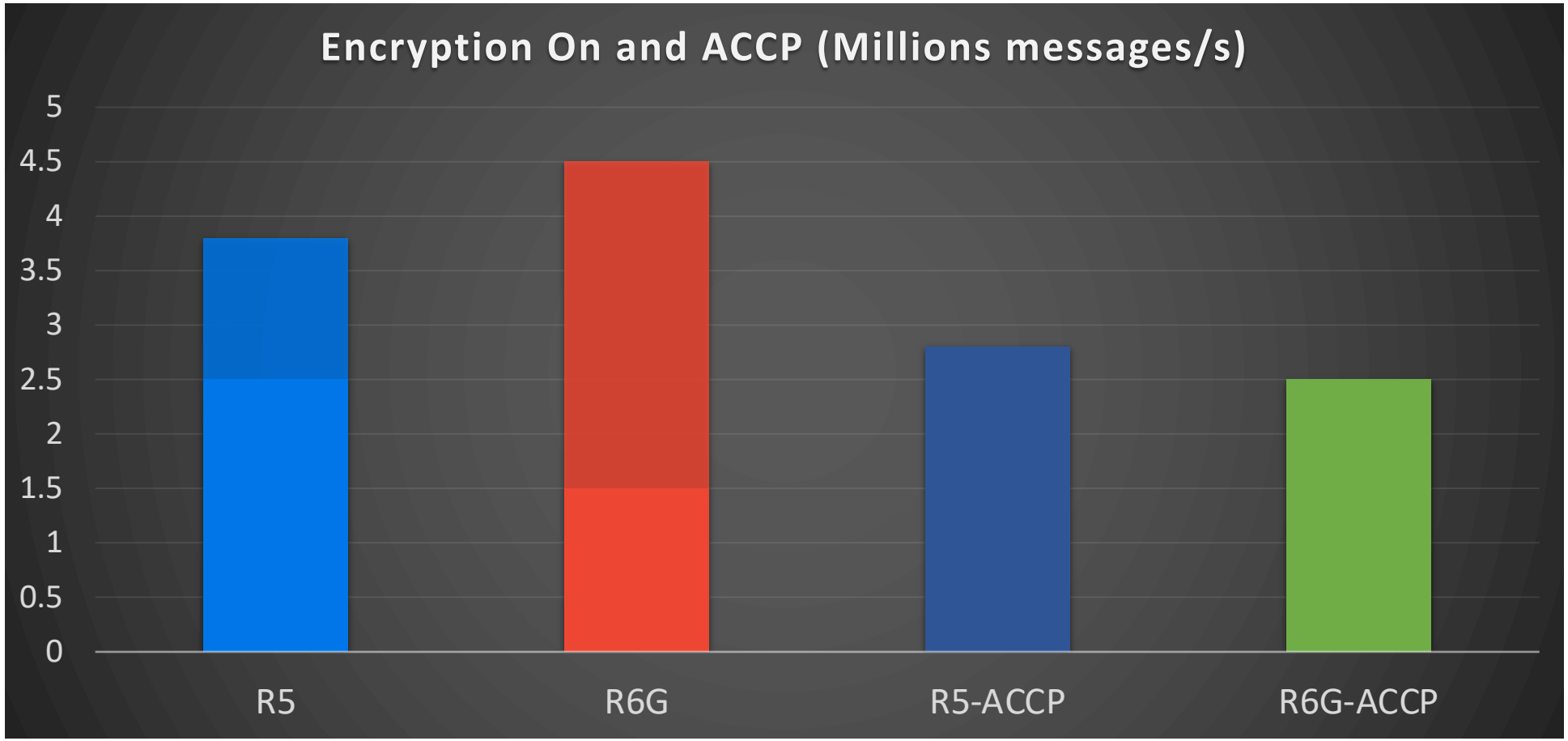
instaclustr

- **Cryptography obviously has a big overhead**
  - but we still need it turned on…
- **An alternative?**
  - Try Amazon Corretto Crypto Provider (ACCP)

Encryption On and ACCP

Comparable Performance → Cheaper

instaclustr



Explanation?

- **OpenJDK/Amazon Corretto encryption on Graviton was slow**
  - Due to lack of support for Intel operation that sped up cryptography
- **Amazon Corretto Crypto Provider (ACCP)**
  - Uses OpenSSL, written in C, and faster!
- **ACCP no longer needed**
  - As there's a patch for OpenJDK (JDK-8267993 & JDK-8271567)

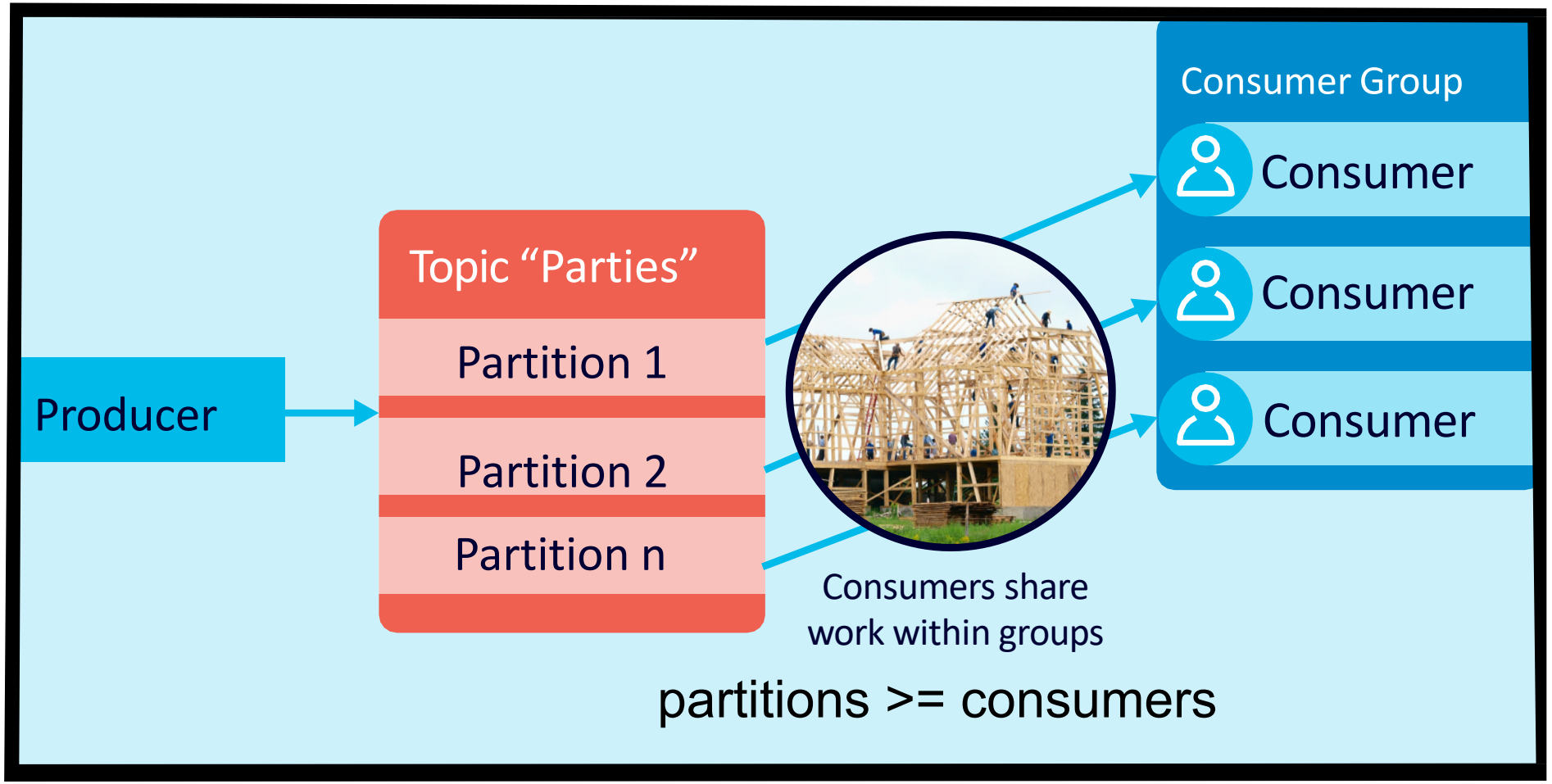
Part 2: Software Change

```
162   /**
163    * Create a topic.
164    * Wait until the leader is elected and the metadata is propagated to all brokers.
165    * Return the leader for each partition.
166    */
167   def createTopic(
168     topic: String,
169     numPartitions: Int = 1,
170     replicationFactor: Int = 1,
171     topicConfig: Properties = new Properties,
172     listenerName: ListenerName = listenerName,
173     adminClientConfig: Properties = new Properties
174   ): scala.collection.immutable.Map[Int, Int] = {
175     if (isKRaftTest()) {
176       resource(createAdminClient(brokers, listenerName, adminClientConfig)) { admin =>
177         TestUtils.createTopicWithAdmin(
178           admin = admin,
179           topic = topic,
180           brokers = brokers,
181           numPartitions = numPartitions,
182           replicationFactor = replicationFactor,
183           topicConfig = topicConfig
184         )
185       }
186     } else {
187       TestUtils.createTopic(
188         zkClient = zkClient,
189         topic = topic,
190         numPartitions = numPartitions,
191         replicationFactor = replicationFactor,
192         servers = servers,
193         topicConfig = topicConfig
194       )
195     }
196   }
197
```


instaclustr

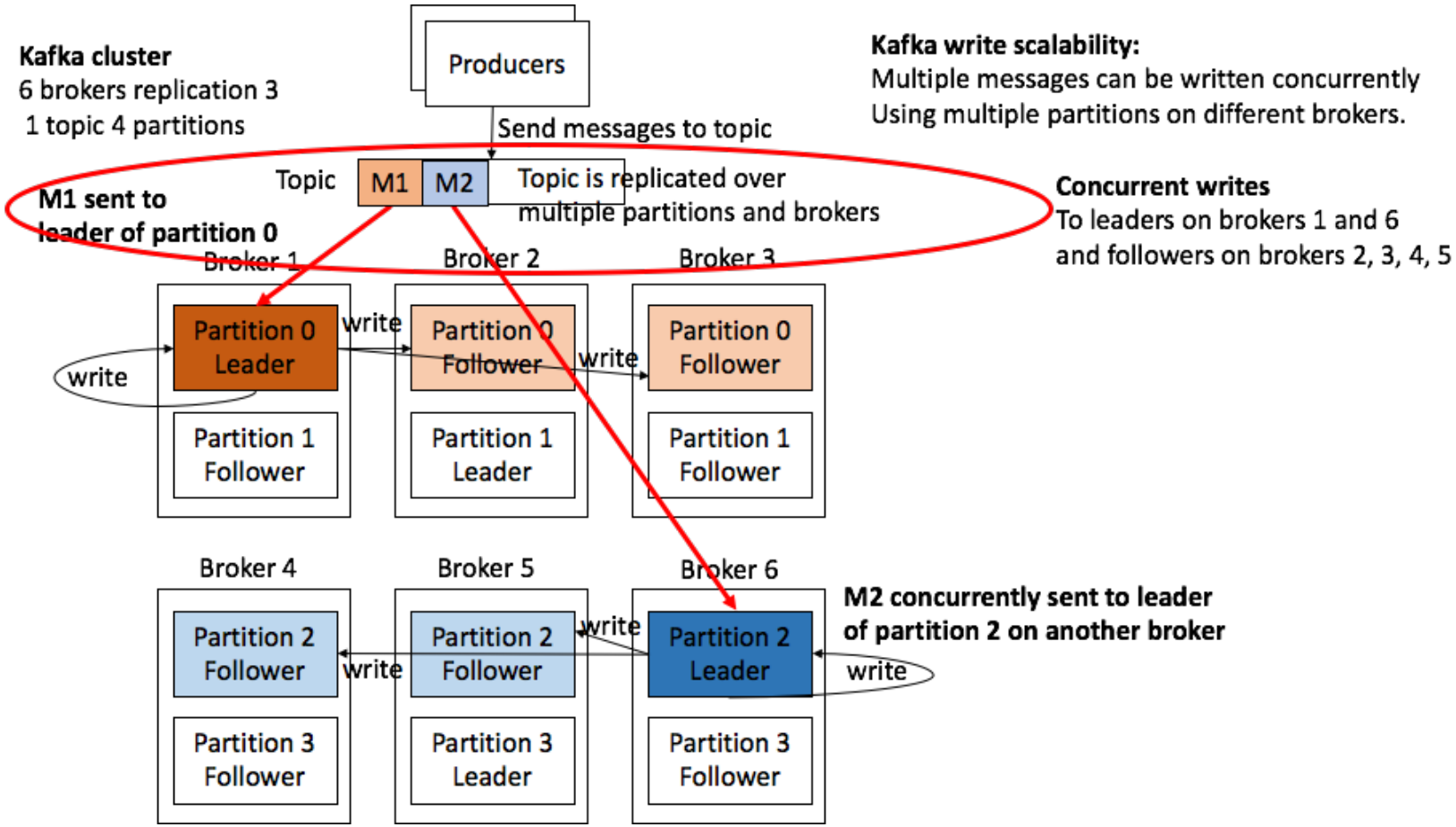
Kafka Topic Partitions Enable Consumer Concurrency

instaclustr



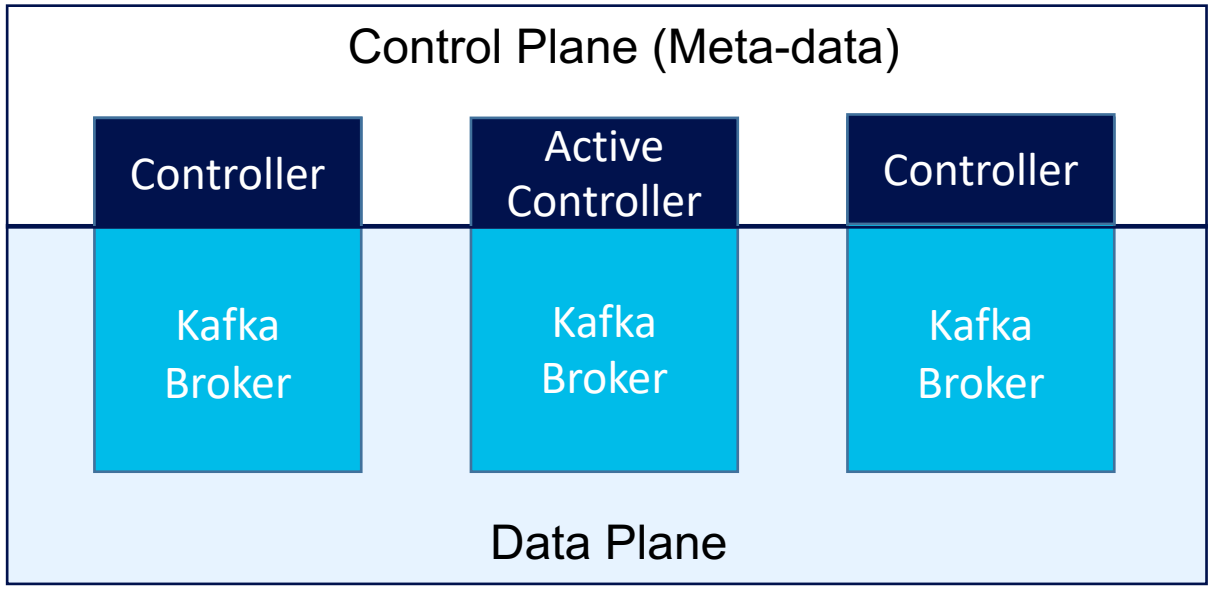
But Partitions Are Expensive:
Replication and Meta-Data Management

instaclustr



Kafka Controller

instaclustr



- **The Kafka Controller manages broker, topic, and partition meta-data—Kafka's "Brain"**
- **But which controller is active and where is the meta-data stored?**

(Source: Shutterstock)

Apache ZooKeeper®

instaclustr

ZooKeeper used for Controller election and storing meta-data

ZK 1

ZK 2
Leader

ZK 3

ZooKeeper Ensemble

Kafka Cluster Metadata

Changes, failures, and recovery → SLOW!

Controller	Active Controller	Controller
Kafka Broker	Kafka Broker	Kafka Broker

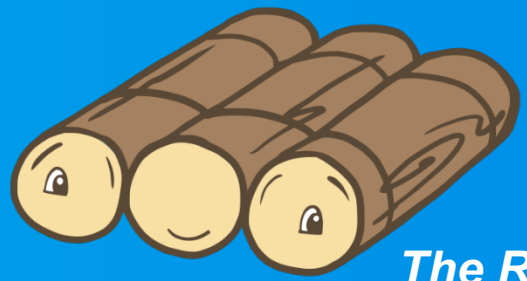
Kafka Cluster Metadata cached in brokers

Kafka Cluster

Kafka Cluster Data

Meta-data changes and recovery from failover are SLOW; Reads are fast due to caching

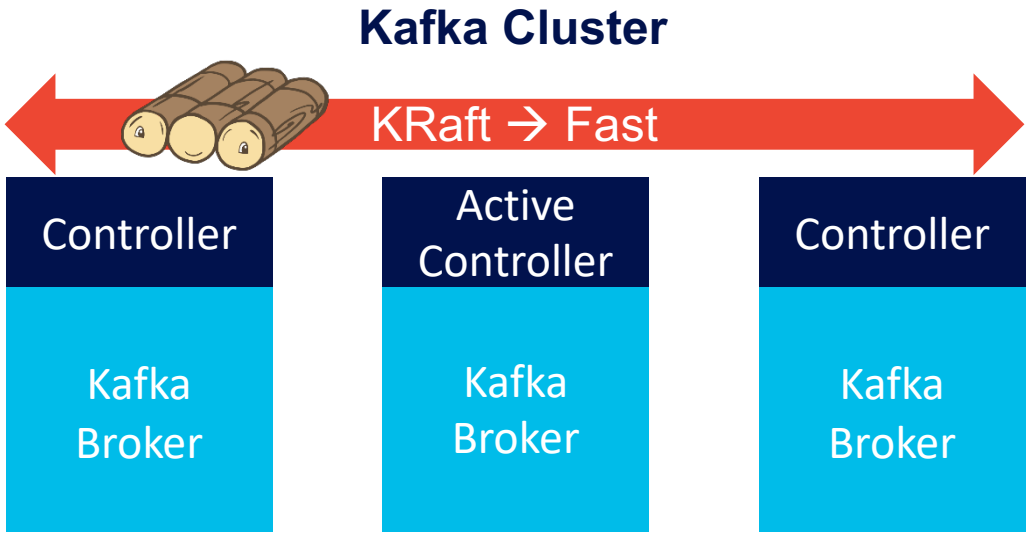
New KRaft Mode



The Raft Mascot

instaclustr

Kafka + Raft Consensus Algorithm = KRaft



- **Kafka Cluster Metadata—only stored in Kafka so fast and scalable**
- **Kafka Cluster Metadata replicated to all brokers, very fast failover**
- **Kafka Cluster Data**

Active Controller is Quorum Leader (using Raft to elect leader)

Hypotheses

What	ZooKeeper	KRaft
Reads and therefore data layer operations cached/replicated	FAST	FAST
Meta-data changes and recovery from failover	SLOW	FAST
Partitions per cluster	LESS	MORE
Robustness	GOOD	UNKNOWN

Experiment 1:
Message Throughput Benchmarking

instaclustr

Hypothesis:

There will be no or only minimal difference between ZooKeeper and KRaft message throughput

Why?

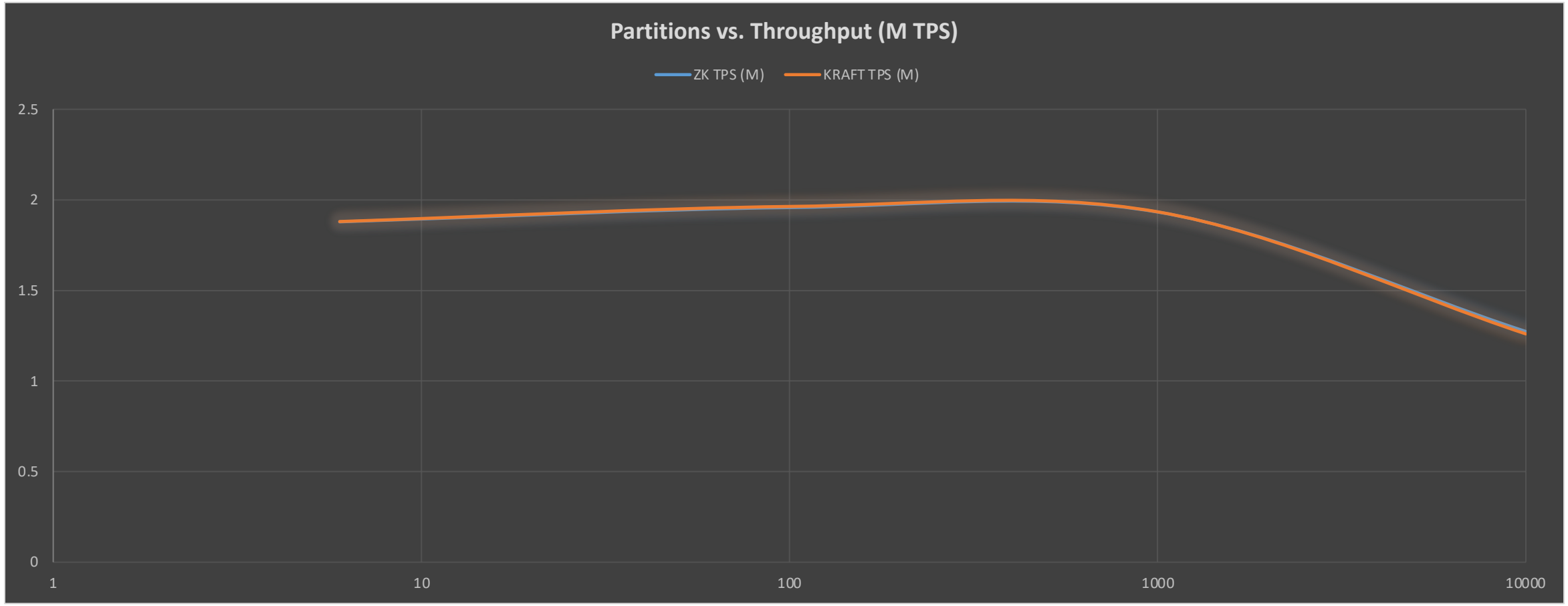
- **As ZK and Kraft are only concerned with meta-data management, not data workloads**
- **Kafka producers only need read-only access to partition meta-data**

How?

- **Kafka 3.1.1. on identical AWS R6G.large x 3 nodes clusters**

Performance:

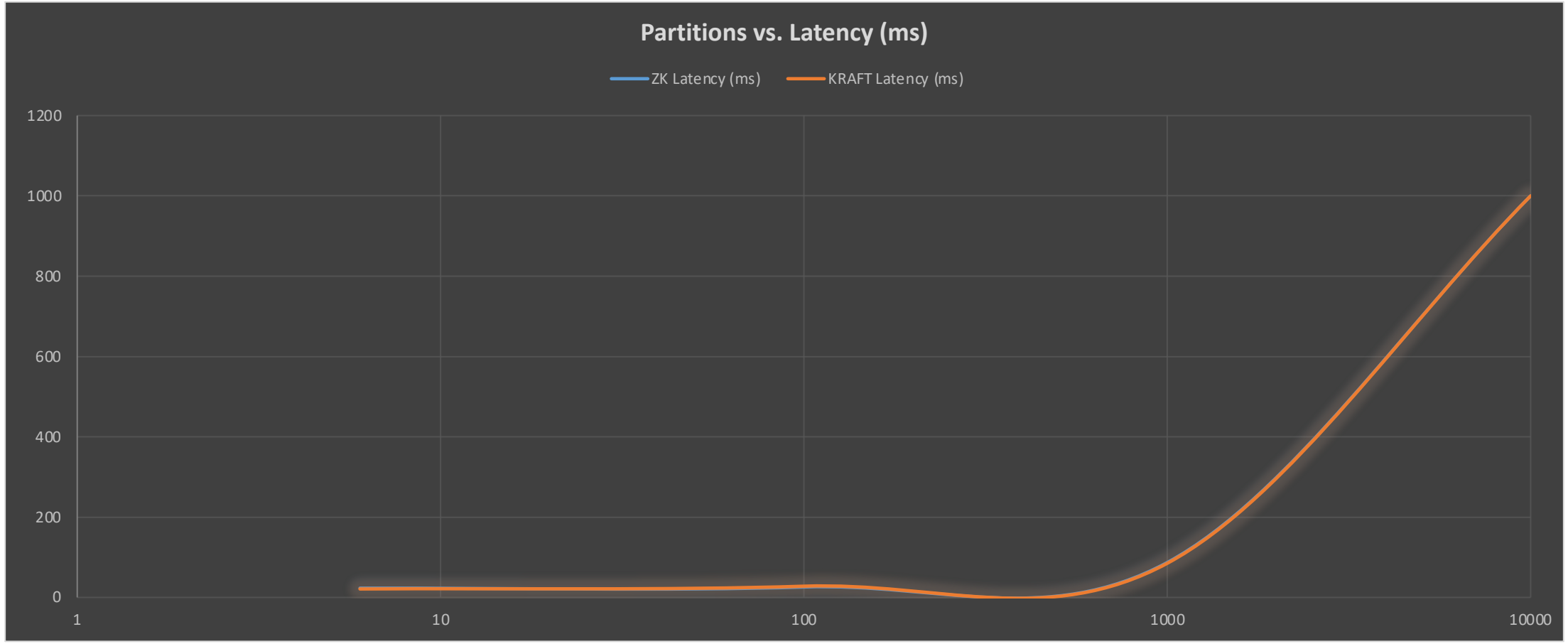
Partitions vs. Throughput (x-axis log)- identical, cliff > 1000



Partitions vs. Latency (ms):

Identical, Worse > 1000 Partitions

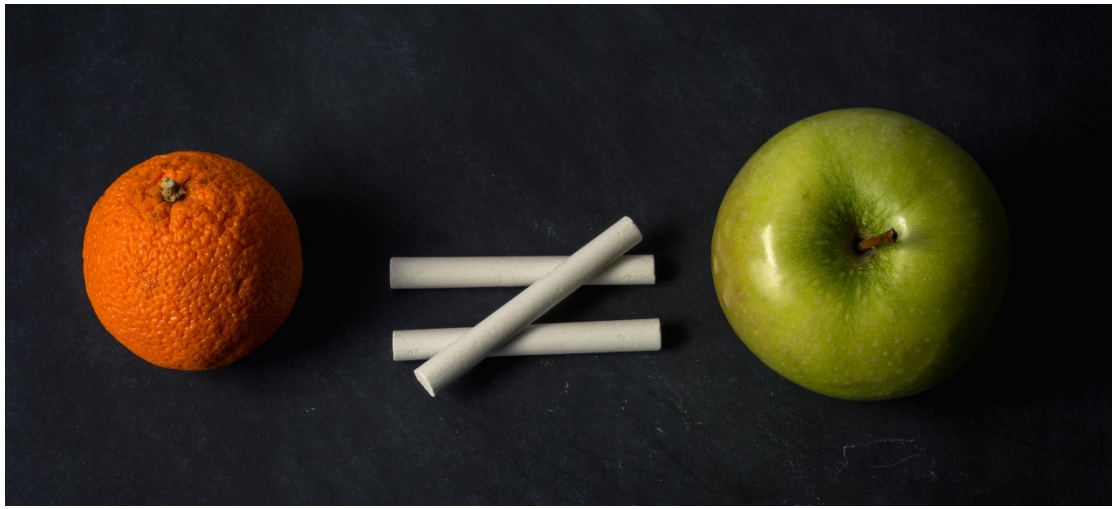
instaclustr



Comparison With Previous Experiments (2020)—Clusters

instaclustr

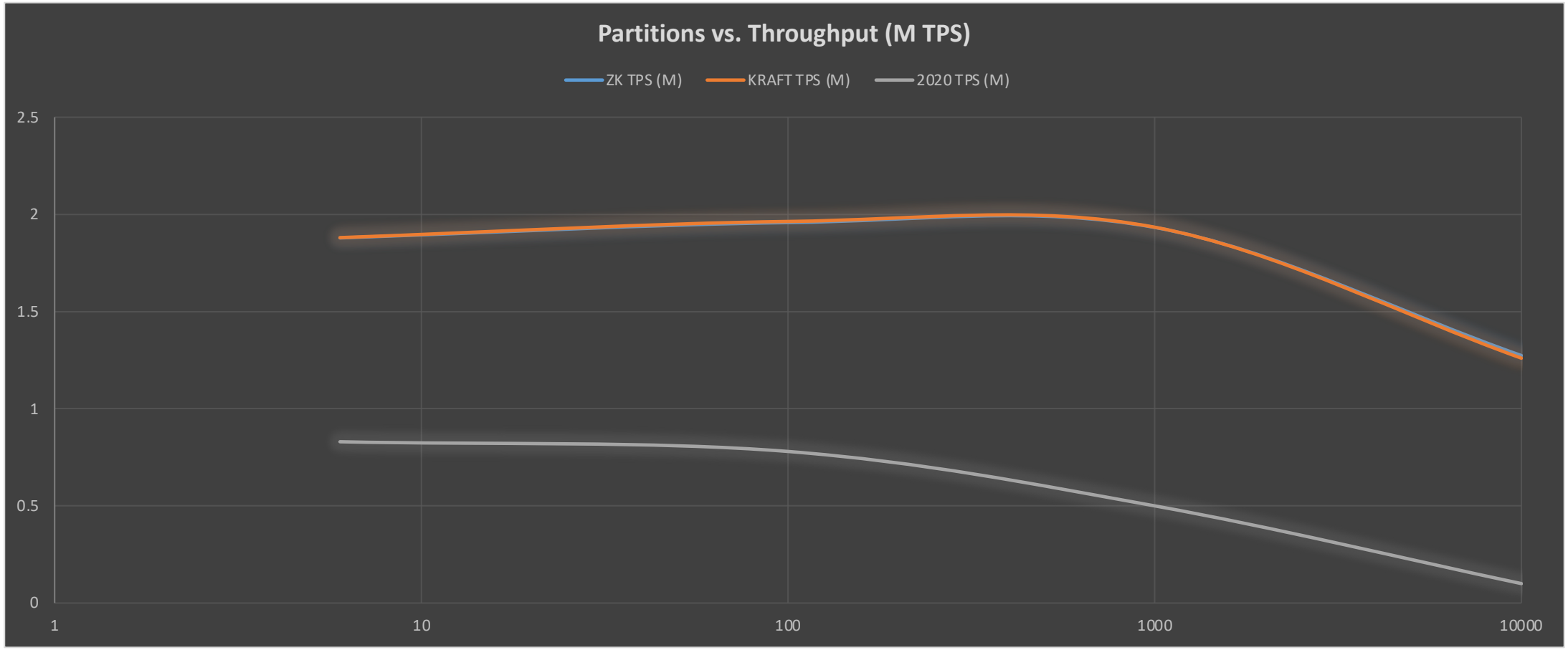
Configuration	Instances	Nodes	Total cores	RF	Kafka version	Date	bytes per msg
Original cluster	r5.xlarge	3	12	3	2.3	Jan-20	80
New cluster	r6g.large	3	6	3	3.1.1	Aug-22	8



(Source: Shutterstock)

- **Note apples-to-oranges comparison as almost everything is different!**
- **Also not directly comparable with results from 1st part of the talk**

Throughput higher and more scalable with increasing partitions c.f. 2020 results



Experiment 2

- How many partitions can we create?
- Can we create more on a KRaft cluster c.f. ZK cluster?
- How long does it take?
- RF=1 otherwise background CPU due to replication too high
  - 50% CPU load on clusters with 100 partitions and no data or workload

Approaches Attempted

instaclustr

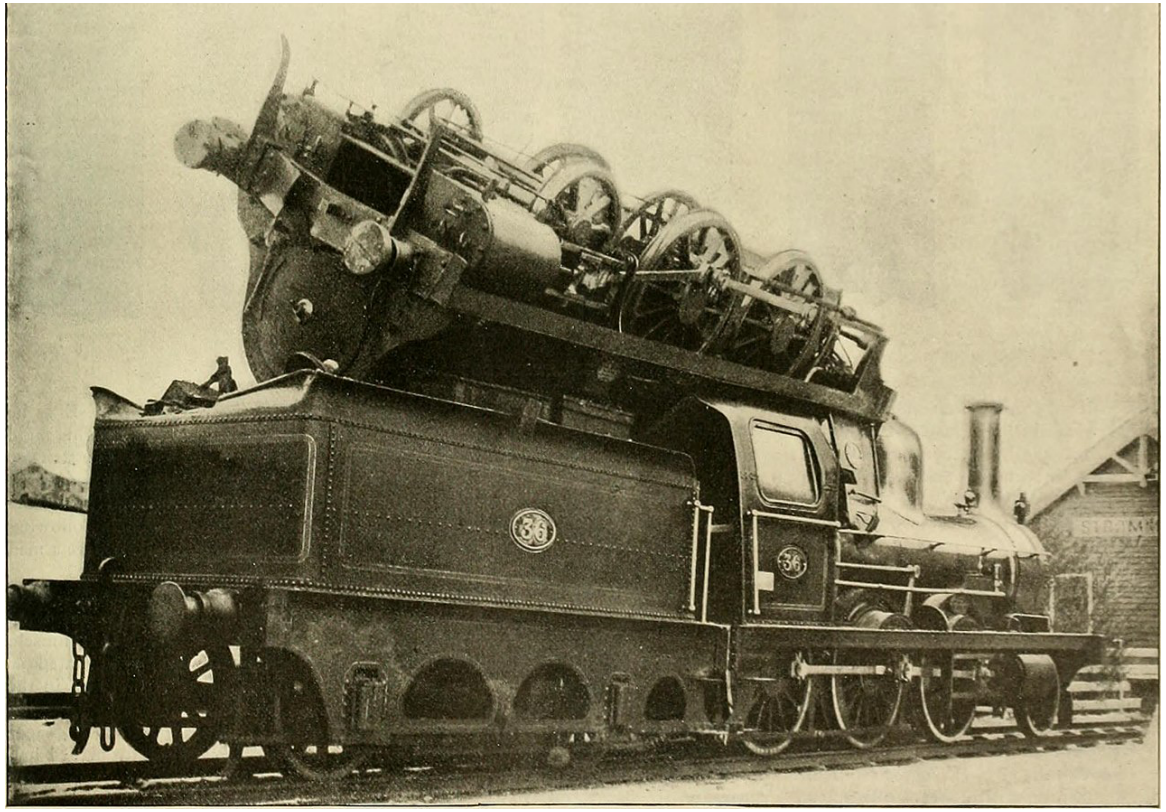
1. kafka-topics.sh -create topic with lots of partitions
2. kafka-topics.sh -alter topic with more partitions
3. curl with our provisioning API
4. script to create multiple topics with fixed (1000) partitions each

Problem!

All approaches failed eventually, some sooner rather than later...

Problem!

After some failures, the Kafka cluster was unusable, even after restarting Kafka



(Source: Commons Wikipedia)

Errors Included

panic("Shannon and Bill say this can't happen.");*

instaclustr

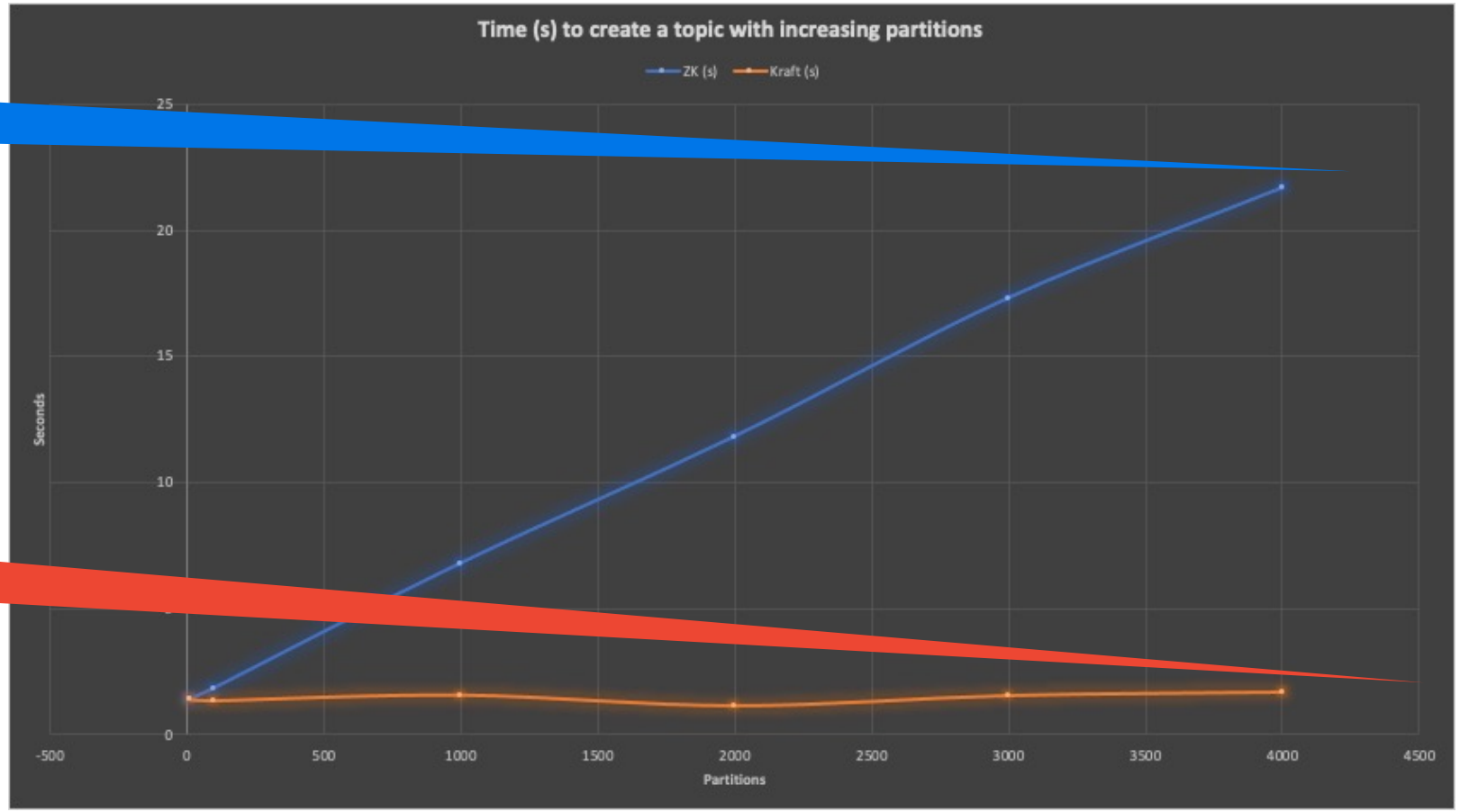
```
Error while executing topic command : The request timed out.
ERROR org.apache.kafka.common.errors.TimeoutException: The request timed out.

From curl: {"errors":[{"name":"Create
Topic","message":"org.apache.kafka.common.errors.RecordBatchTooLargeException
: The total record(s) size of 56991841 exceeds the maximum allowed batch size
of 8388608"}]}

org.apache.kafka.common.errors.DisconnectException: Cancelled createTopics
request with correlation id 3 due to node 2 being disconnected

org.apache.kafka.common.errors.DisconnectException: Cancelled
createPartitions request with correlation id 6 due to node 1 being
disconnected
```

* A historical error, “Shannon and Bill” = Bill Shannon – 1955-2020 (Sun, UNIX, J2EE)

Partition Creation Time

Eventual Timeout

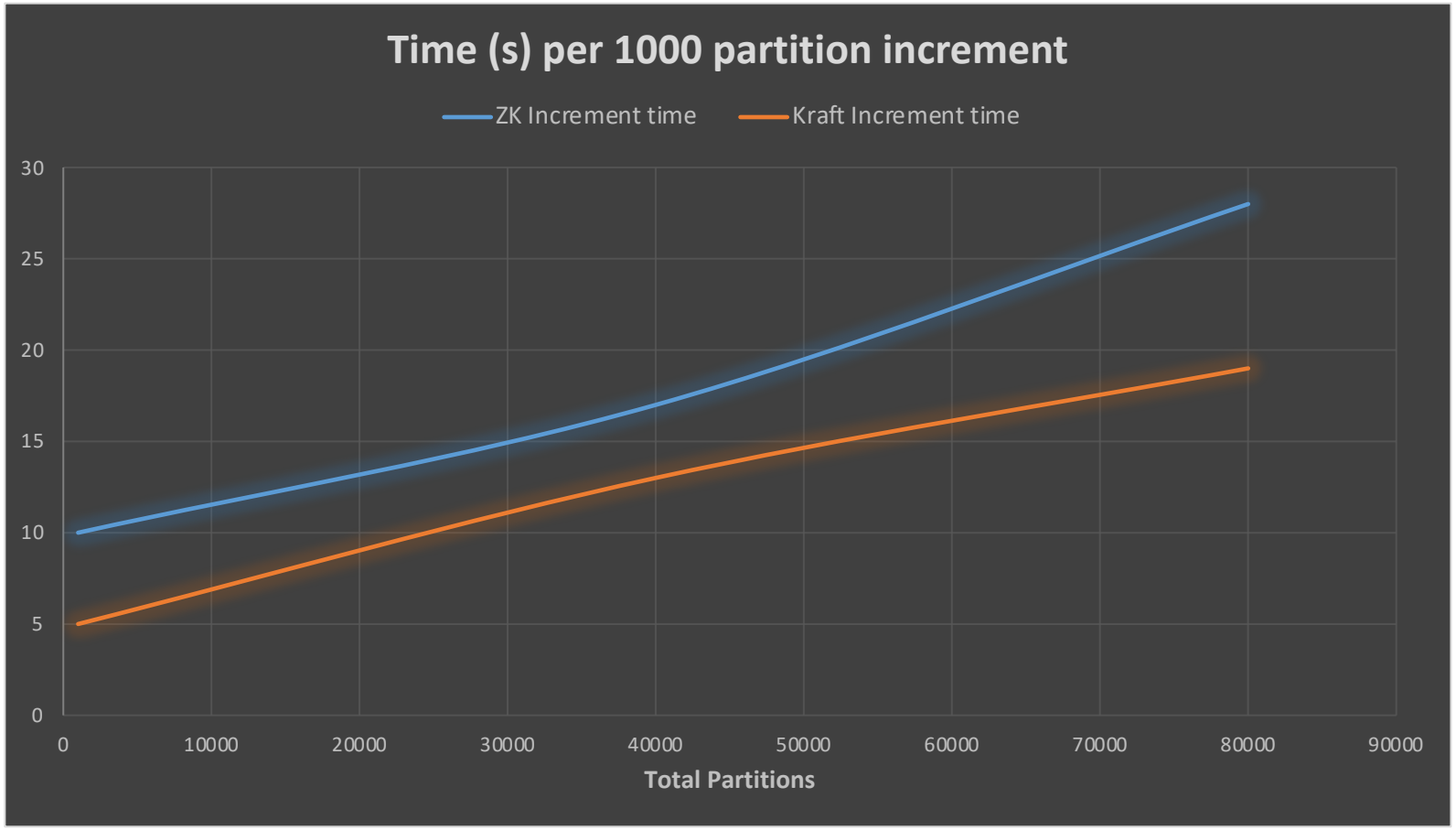
Eventual Failure



ZooKeeper O(n)

KRaft O(1)

Incremental Approach



Time per 1000 partition increment - increases with total partitions

ZooKeeper slower than KRaft

Slow process to create many partitions!

And eventual failure

Initial Conclusions?

- Faster to create more partitions on KRaft c.f. ZK
- There's a limit of around 80,000 partitions on both ZK and KRaft clusters
- And Kafka fails!
- It's very easy and quick to kill Kafka on KRaft – just try and create a 100k partition topic

Experiment 3:
Reassign Partitions

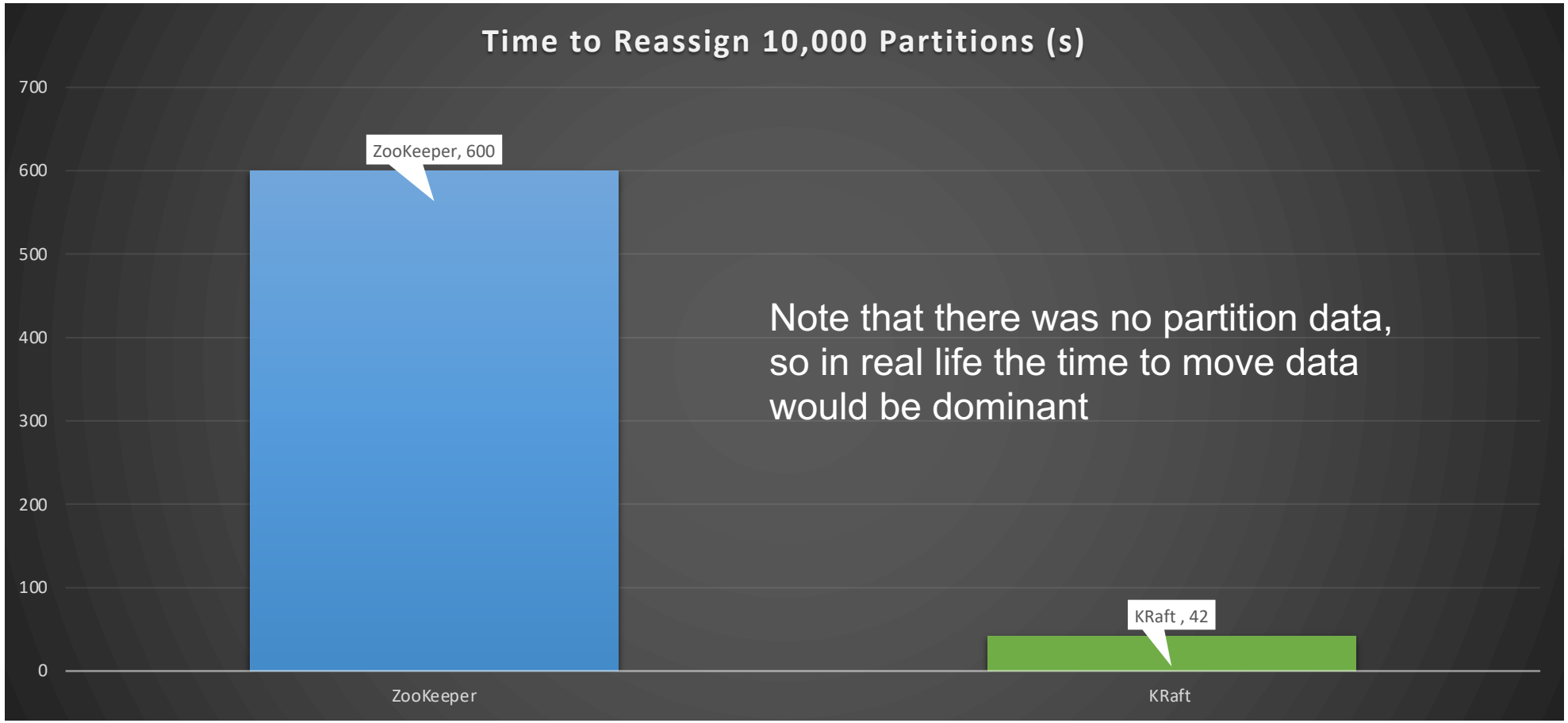
instaclustr

A common Kafka operation—if a server fails, you can move all of the leader partitions on it to other brokers

kafka-reassign-partitions.sh

- Run once to get a plan, and then again to actually move the partitions
- Moving partitions from 1 broker to the other 2 brokers
- 10,000 partitions, RF=2

The Answer to Life, the Universe, and Everything = 42s



Experiment 4: Maximum Partitions

instaclustr

- **Final attempt to reach 1 Million+ Partitions on a cluster (RF=1 only however)**
- **Used manual installation of Kafka 3.2.1. on large EC2 instance**
- **Hit limits at around 30,000 partitions:**

```
ERROR [BrokerMetadataPublisher id=1] Error publishing broker metadata at 33037
(kafka.server.metadata.BrokerMetadataPublisher)
java.io.IOException: Map failed
# There is insufficient memory for the Java Runtime Environment to continue.
# Native memory allocation (mmap) failed to map 65536 bytes for committing reserved memory.
```

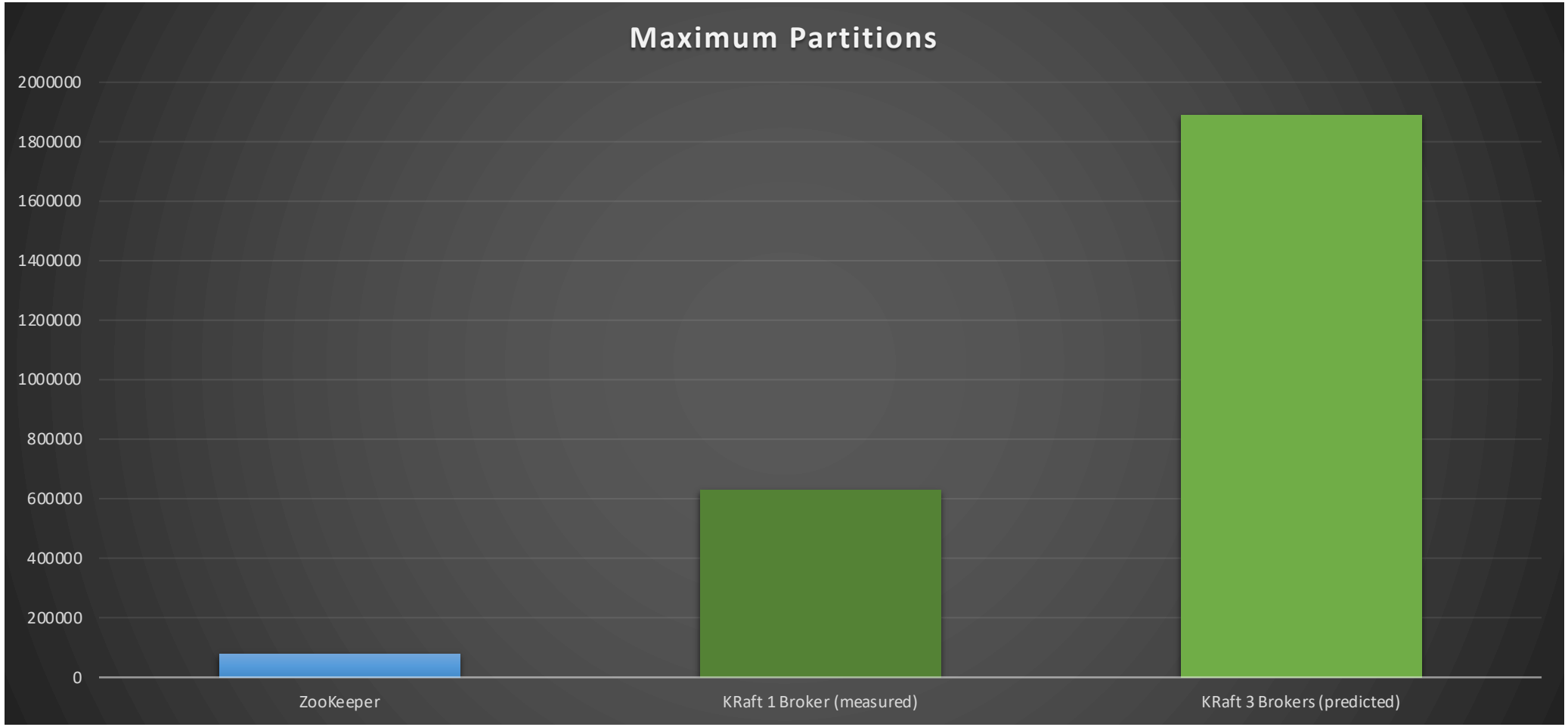
More RAM needed? No – didn't help.
More file descriptors? 2 descriptors used per partition. Only 65535 by default on Linux. Increased – still failed.

Experiment 4:
Maximum Partitions

instaclustr

- Plenty of spare RAM but out of memory error
- Googling found this:
  - KAFKA-6343 OOM as the result of creation of 5k topics (2017!)
  - Linux system setting: vm.max_map_count: Maximum number of memory map areas a process may have.
  - Each partition uses 2 map areas, default is 65530, allowing a maximum of only 32765 partitions.
- Set to a very large number, tried again…
- Now just get a normal memory error:
  - “java.lang.OutOfMemoryError: Java heap space”
- Tweaked JVM settings, and tried again…

1.9M Partitions > 1M → Success



But How About the Batch Error?

- **Still painfully slow to create this many partitions due to the batch error when creating too many partitions at once.**
- **This is a real bug: KAFKA-14204: QuorumController must correctly handle overly large batches**
- **Fixed in 3.3.3. (maybe, not tested)**

Use Cases for Lots of Partitions

1. Lots of topics! E.g. due to data model or security
2. High throughput
3. Slow consumers—shoppers with more groceries take longer at the checkout, so you need more checkouts to service shoppers

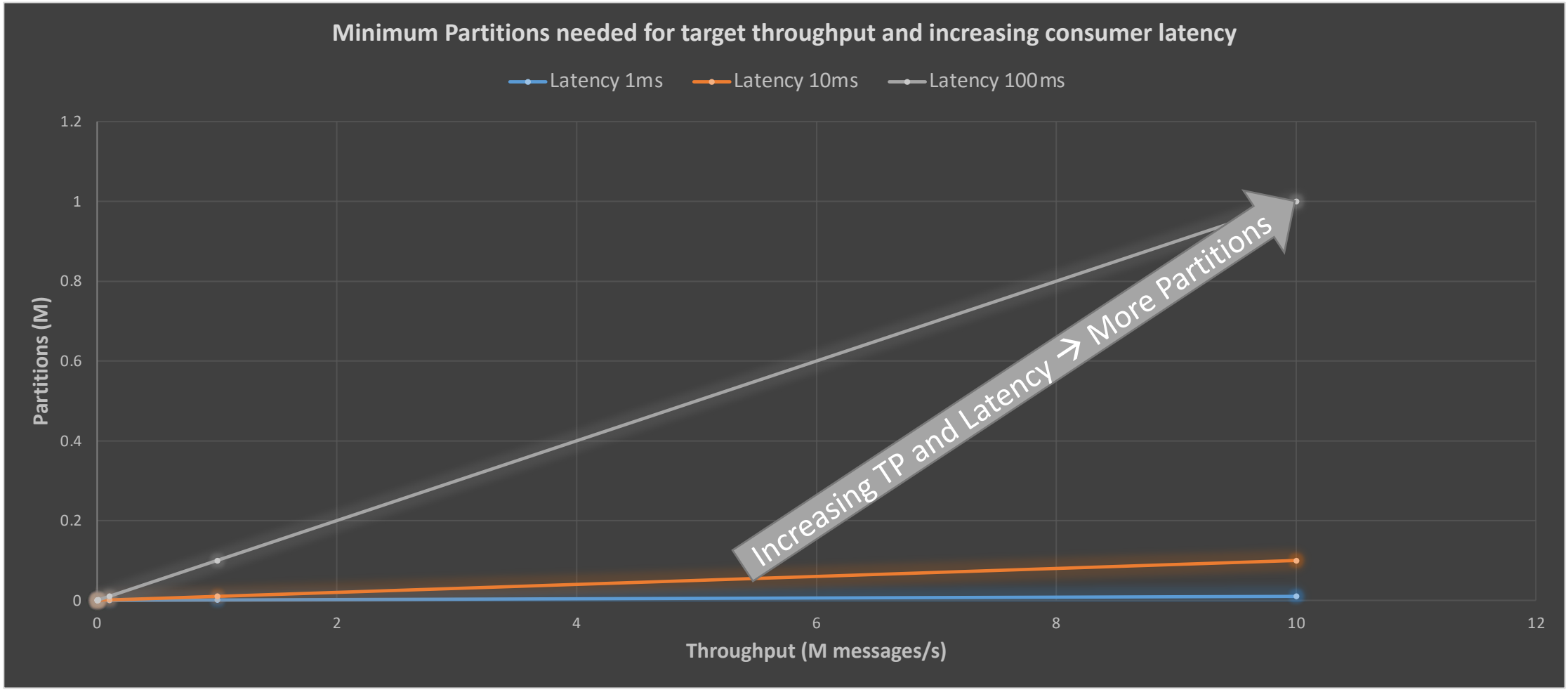
Possible problems?

- RF=3 → large clusters
- Lots of consumers
  - Consumer resources
  - Consumer group balancing performance
  - Key values >> partitions, etc.

(Source: AdobeStock)

Little's Law: Partitions = TP x RT
RT is Kafka consumer latency

instaclustr



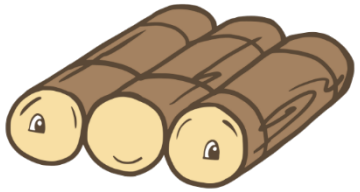
Conclusions

instaclustr

What	ZooKeeper 	KRaft 	Results
Reads and therefore data layer operations cached/replicated	FAST	FAST	Identical Confirmed
Meta-data changes	SLOW	FAST	Confirmed
Maximum Partitions	LESS	MORE	Confirmed
Robustness	YES	WATCH OUT	OS settings!

Kafka will soon abandon the Zoo(Keeper) on a KRaft!

instaclustr



Performance Engineering Takeaways For Apache Software?

instaclustr

(Source: Paul Brebner, Broken Hill, Australia)

- **Hardware and Software changes will cause performance surprises**
  - **potentially due to underlying layers**
- **Regular benchmarking, hypotheses, experiments, profiling, testing, etc**
  - **help improve community understanding and end-user experience of performance and scalability**
- **Open source cloud providers have a useful role to play in**
  - **performance assurance, and**
  - **for providing insights into running, optimizing and using Apache technologies at scale in production**

THANK YOU!

instaclustr

www.instaclustr.com

info@instaclustr.com

@instaclustr