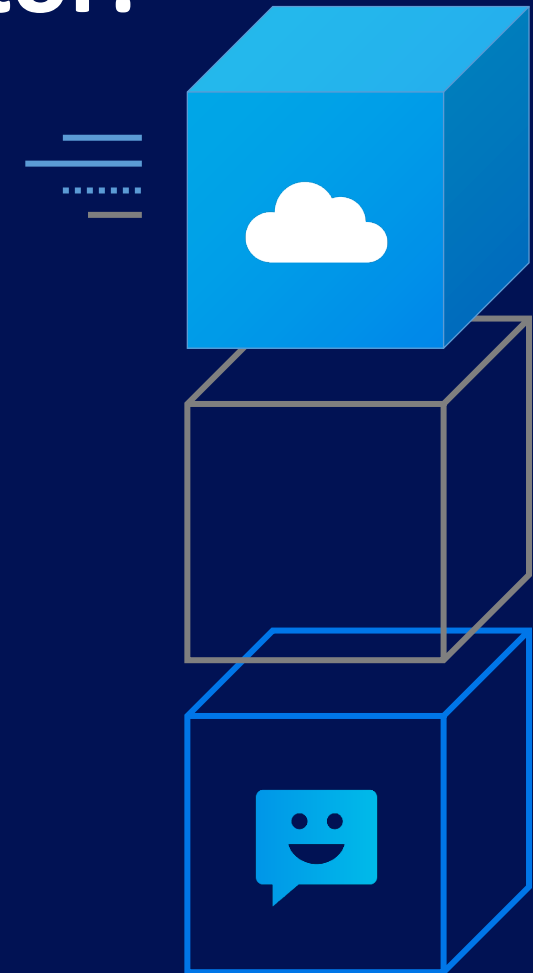


Apache ZooKeeper and Apache Curator: Meet the Dining Philosophers

ApacheCon NA 2022

Paul Brebner
Instaclustr by NetApp





Who Am I?



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[®]

I live in Australia!

A ZooKeeper Walks Into a Pub...



APACHE
ZooKeeperTM

Actually an Outback Pub

Outback = “Out of the back of ...”, e.g. “Back O’ Bourke”



Google Maps



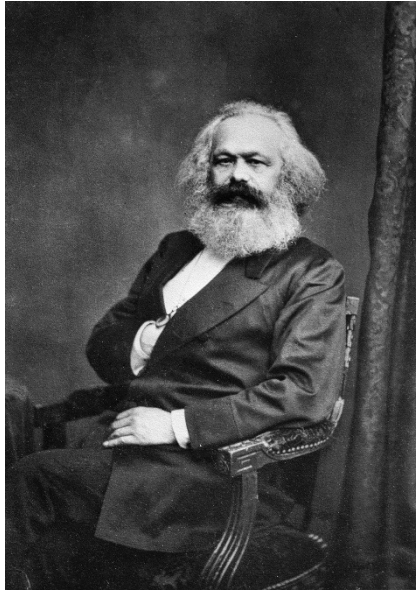
The Pub



Daly Waters Pub in outback Northern Territory, Australia

(Source: [Wikimedia Commons](#))

And Meets Some Unexpected Guests...



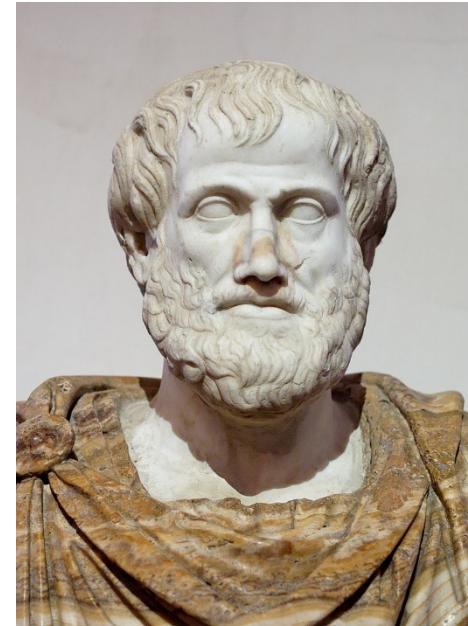
Karl (Marx)

**Ludwig
(Wittgenstein)**



**Niccolo
(Machiavelli)**

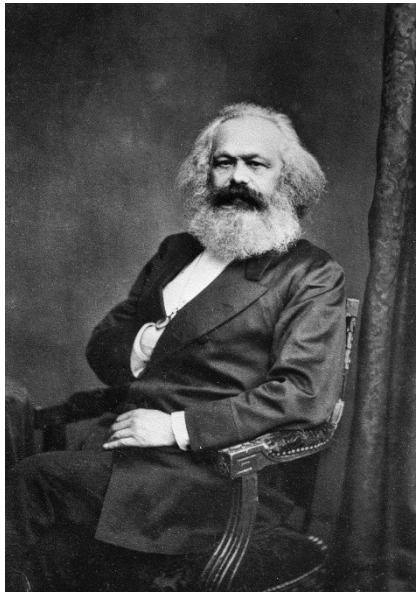
Aristotle



Confucius

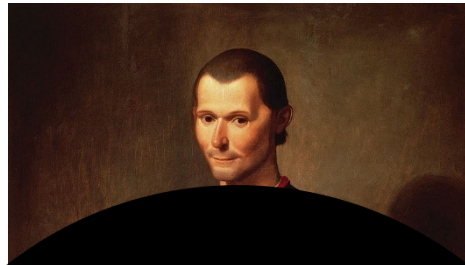


...Who Appear to Be Fighting Over Cutlery



Karl (Marx)

**Ludwig
(Wittgenstein)**



Aristotle



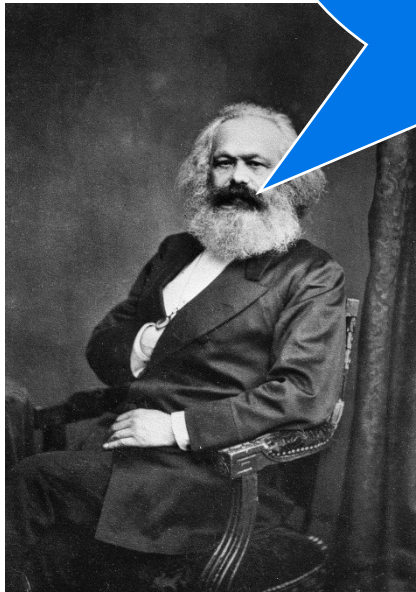
Confucius



(Source: Wikimedia Commons/Shutterstock)

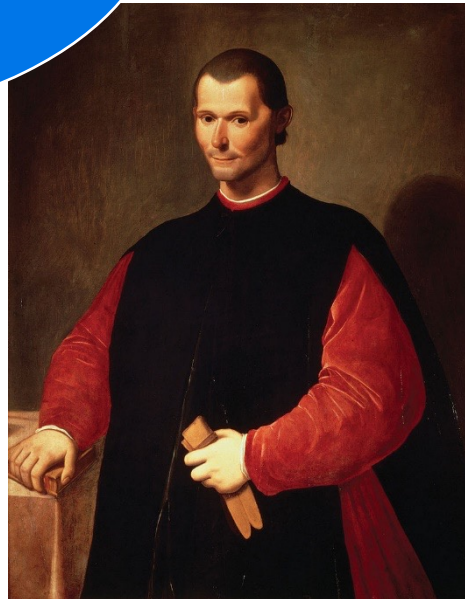


“Ludwig, I’m hungry please lend me a fork.”



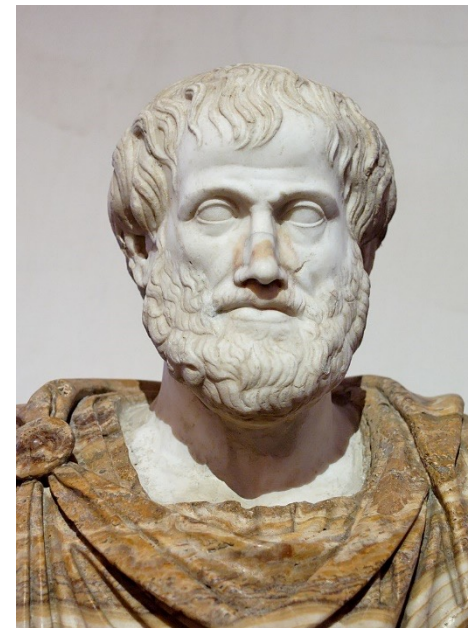
Karl (Marx)

**Ludwig
(Wittgenstein)**



**Niccolo
(Machiavelli)**

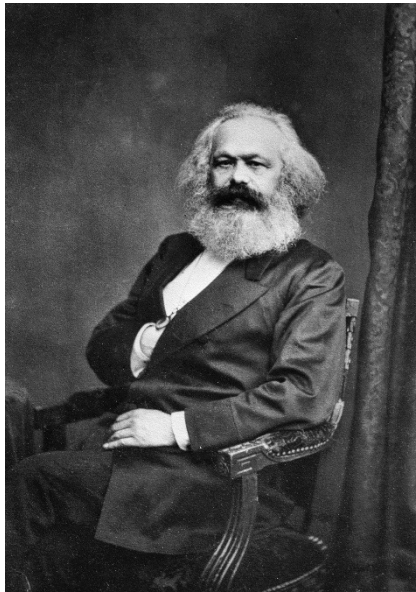
Aristotle



Confucius



“Karl, I don’t fully understand what you mean by the word ‘fork.’”



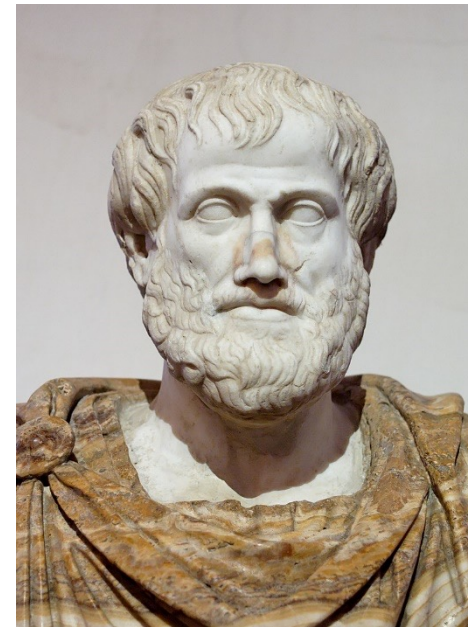
Karl (Marx)

**Ludwig
(Wittgenstein)**



**Niccolo
(Machiavelli)**

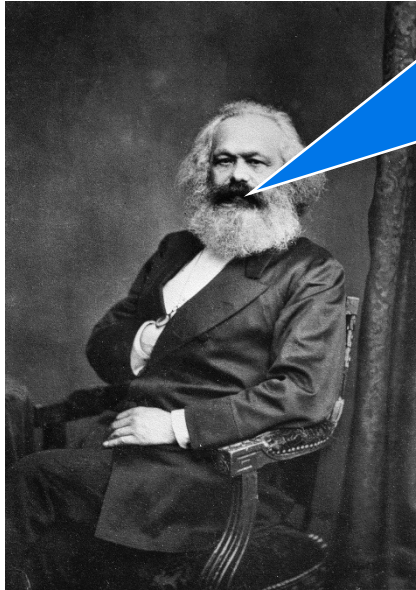
Aristotle



Confucius

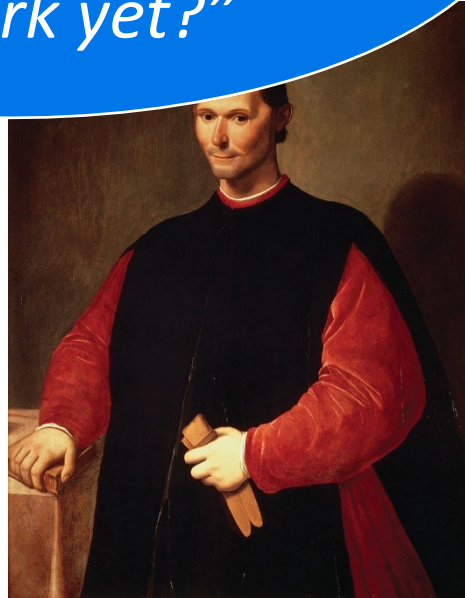


“Ludwig, you are not sharing fairly! Niccolo, have you finished with your fork yet?”



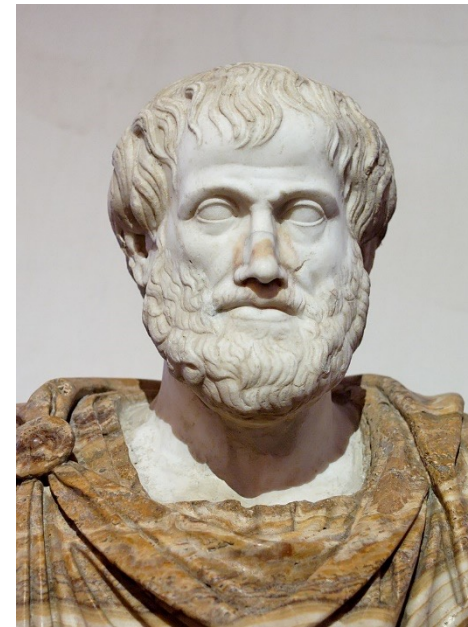
Karl (Marx)

**Ludwig
(Wittgenstein)**



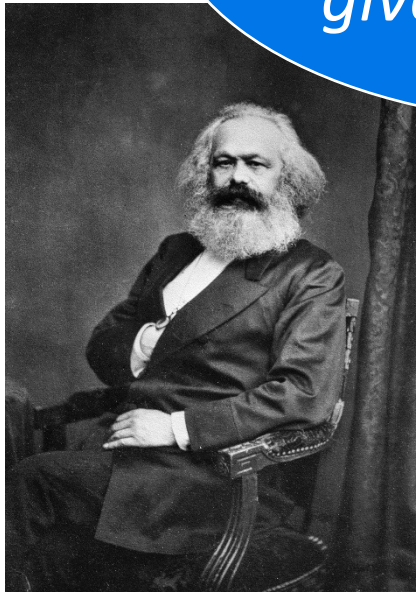
**Niccolo
(Machiavelli)**

Aristotle



Confucius

"No, it's mine! I will never ever give you my fork."

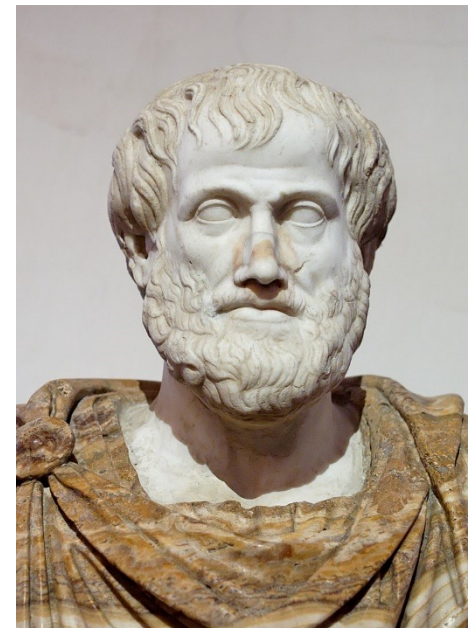


Karl (Marx)

**Ludwig
(Wittgenstein)**



**Niccolo
(Machiavelli)**



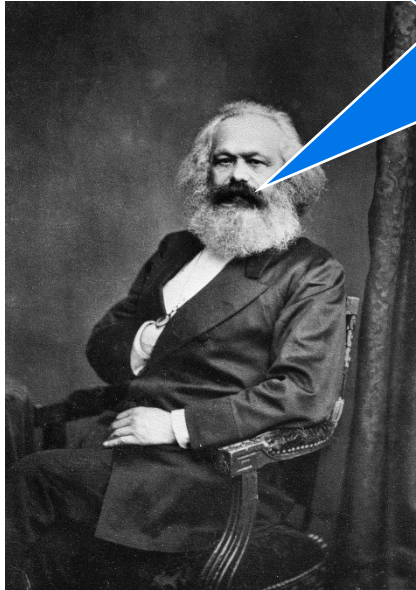
Aristotle



Confucius

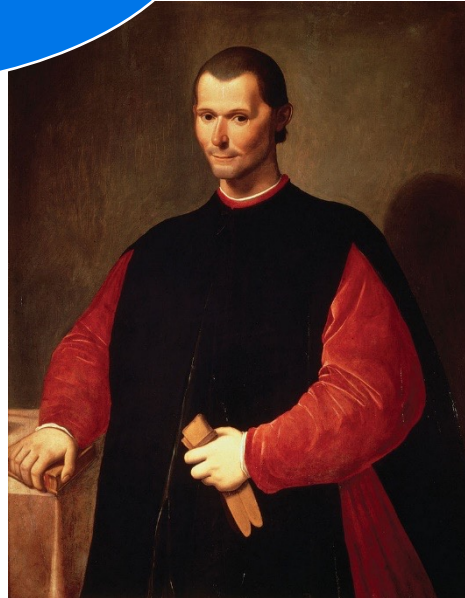


“Who invented these silly rules anyway?”



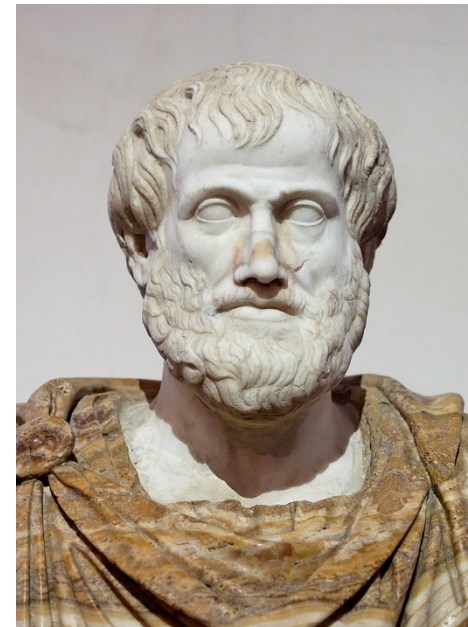
Karl (Marx)

**Ludwig
(Wittgenstein)**



**Niccolo
(Machiavelli)**

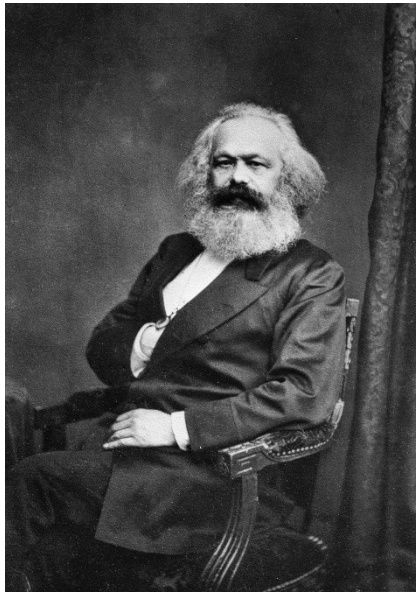
Aristotle



Confucius



*“Dijkstra—a
‘computer scientist’—
whatever that is!”*



Karl (Marx)

**Ludwig
(Wittgenstein)**



**Niccolo
(Machiavelli)**



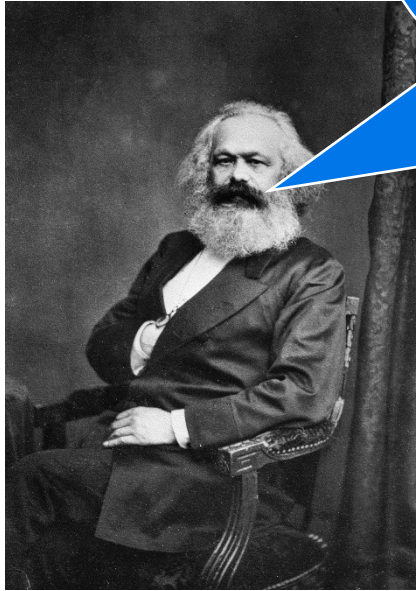
Aristotle



Confucius



“Why can’t I stage a proletarian revolution and take control of all the means of production— I mean forks?”



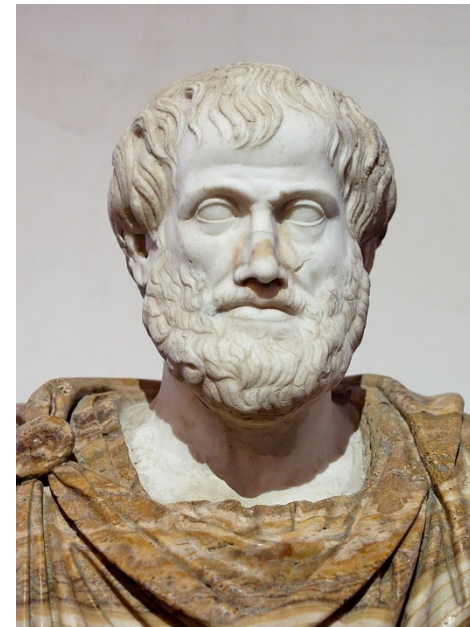
Karl (Marx)

**Ludwig
(Wittgenstein)**



**Niccolo
(Machiavelli)**

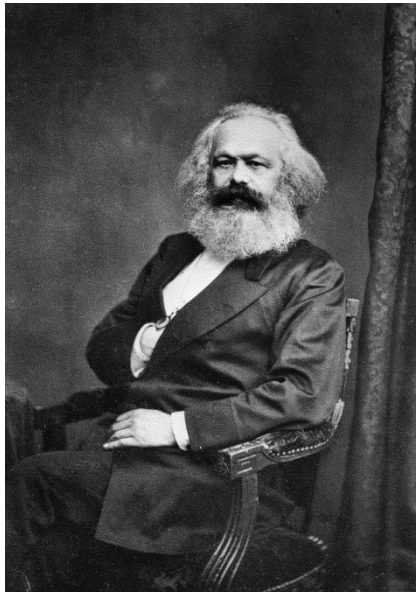
Aristotle



Confucius

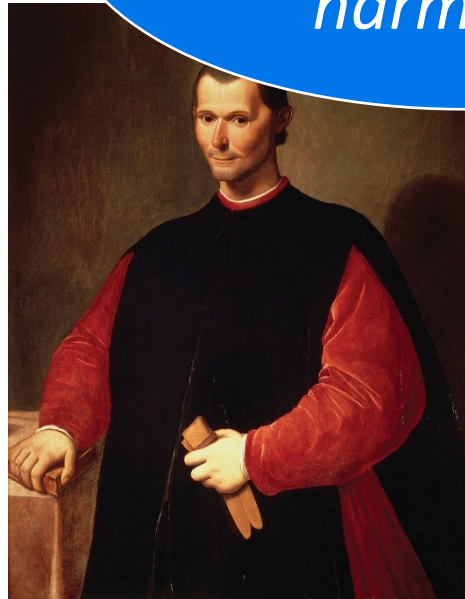


"Ritual leads to greater social harmony"

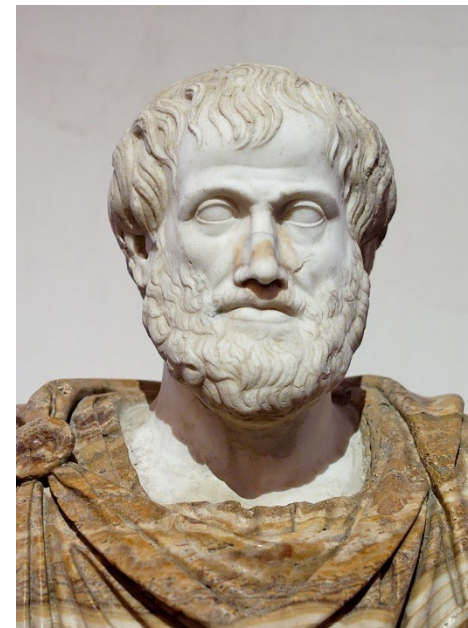


Karl (Marx)

Ludwig (Wittgenstein)



Niccolo (Machiavelli)



Aristotle



Confucius

The ZooKeeper, being trained in the harmonious running of zoos, goes over to see if he can help out with their resource sharing problem...

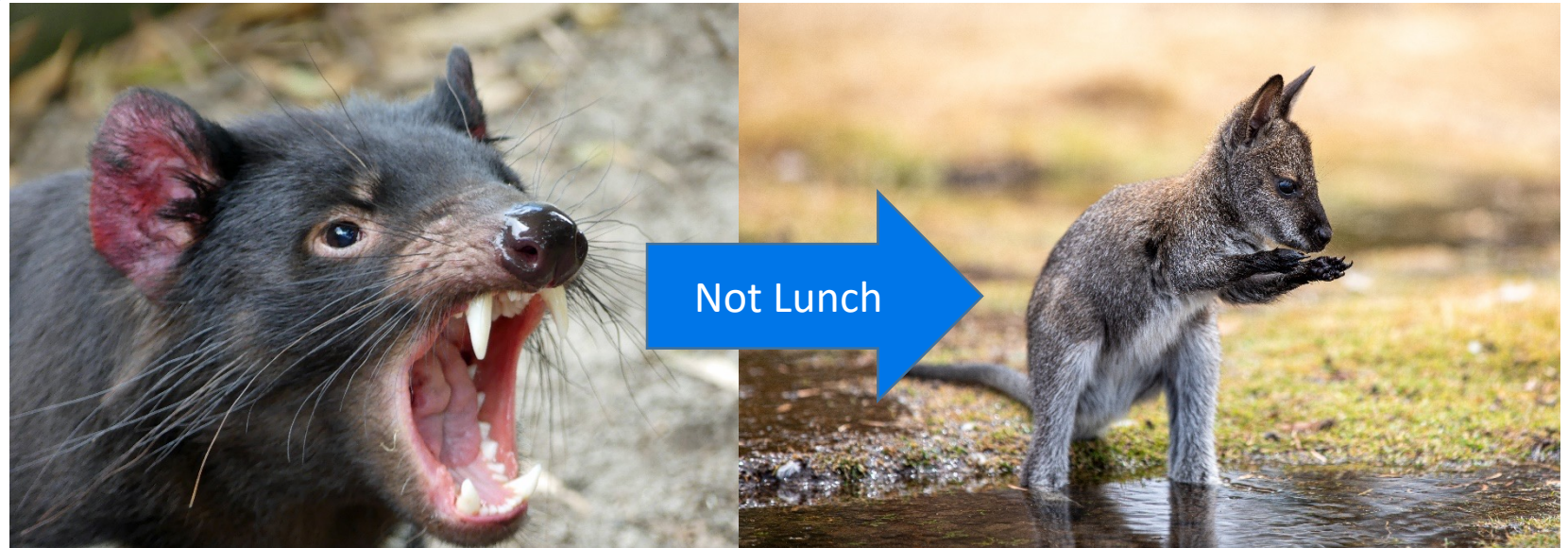


Gentlemen, perhaps I can be of assistance. Given my often complex and dangerous job as a ZooKeeper I'm sure I can help!

APACHE
ZooKeeper™

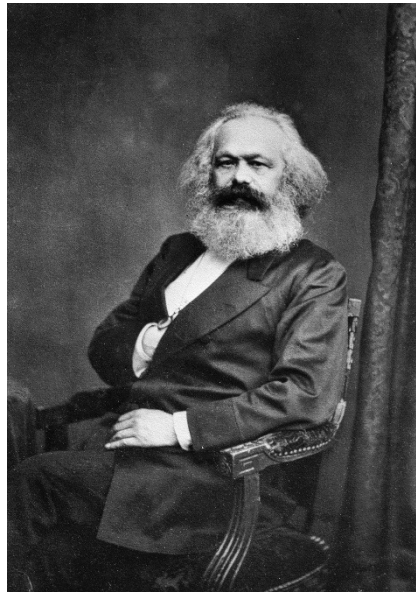


None of
my Tasmanian
Devils have eaten any
of the Wallabies—yet!



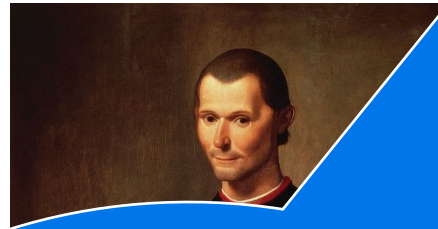
The Dining Philosophers Reply

(for that was the name of their club)

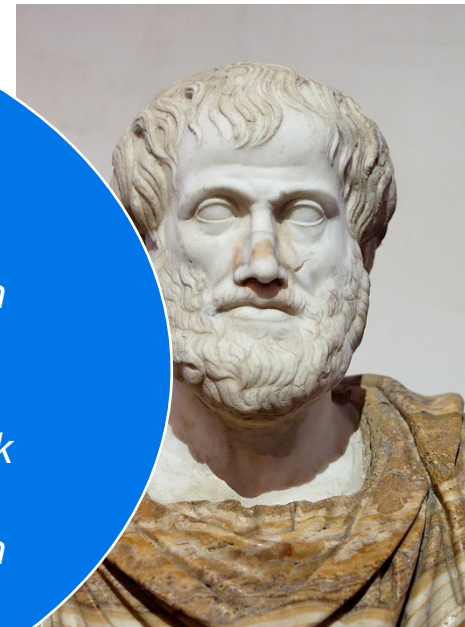


Karl (Marx)

**Ludwig
(Wittgenstein)**



Aristotle

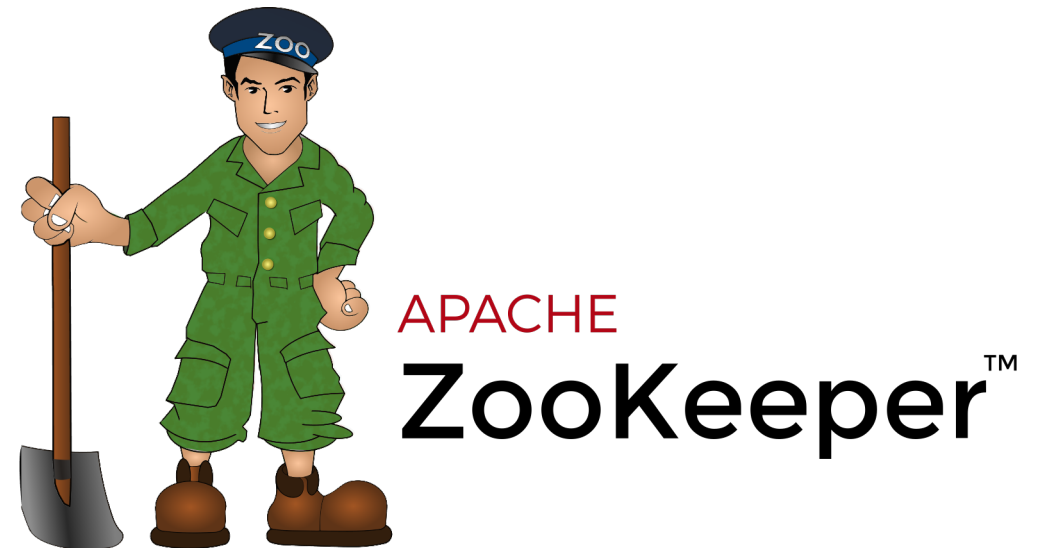


Confucius

*Sir, thank you very much!
We accept your kind offer,
as we are very Hungry!
Although we have no conception
of how you will solve our
intractable problem.
But then at least we can get back
to the more serious matters of
Thinking and Eating, hopefully in
approximately equal amounts.*


Introducing the ZooKeeper

- **A Mature Apache Technology**
 - De facto technology for distributed systems coordination and meta-data storage
 - Originated as a sub-project of Hadoop
 - Used by lots of projects including Kafka
- **Instaclustr Uses ZooKeeper**
 - For managed Kafka clusters
 - For our managed High Availability PostgreSQL® service (distributed config store for Patroni)
 - And offers it as ...



An Instaclustr Managed Service





Now part of Spot by NetApp

[Platform](#) [Pricing](#) [Services](#) [Partners](#) [Resources](#) [Contact us](#)






[Free Trial](#) [Log In](#)


// Managed and Hosted

Instaclustr for Apache ZooKeeper™

Instaclustr provides a fully managed service for Apache ZooKeeper™—SOC 2 certified and hosted on [AWS](#). Easily coordinate and manage your distributed applications with the help of Instaclustr Managed ZooKeeper.

[Contact Us](#) [Get Started >](#)



Introducing the ZooKeeper

Optimized for

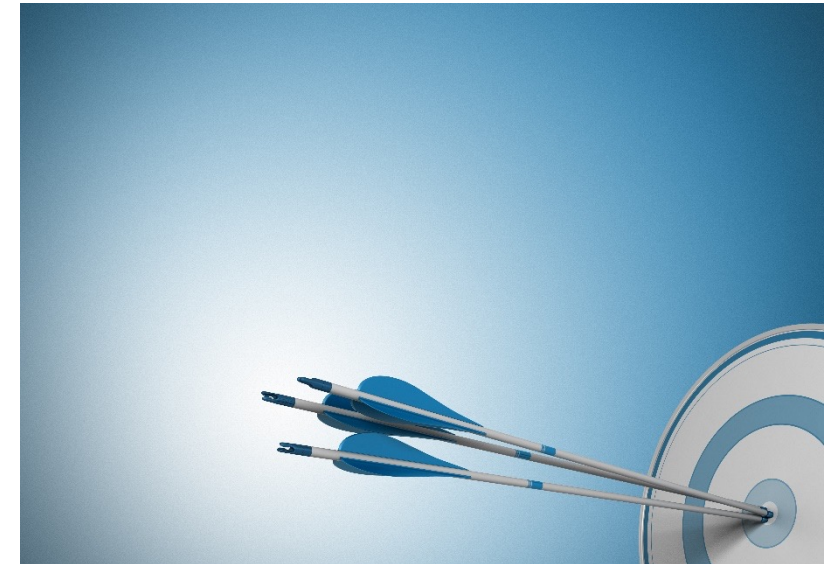
1. Consistency
2. Availability
3. Performance



APACHE
ZooKeeperTM

Consistency With ZAB

- **Single leader for atomic writes**
- **Multiple replicas**
- **ZAB**
 - ZooKeeper Atomic Broadcast
 - Protocol for sequential ordering and consistency



(Source: Shutterstock)

Availability



- **No Clusters, but Ensembles**
 - 1 Leader, multiple replicas
- **If the leader fails then another replica takes over**
- **Ensemble available if majority ($> \frac{1}{2}$) of servers running**
- **Ensembles should have an odd numbers of servers, e.g.**
 - 3 servers supports 1 server failure (2 out of 3 running)
 - 5 servers support 2 server failures (3 out of 5 running)
 - 7 servers support 3 server failures (4 out of 7 running)
- **Reads go to any server (so will succeed if ≥ 1)**
- **Writes go to leader only (and succeed if $> \frac{1}{2}$ servers running)**



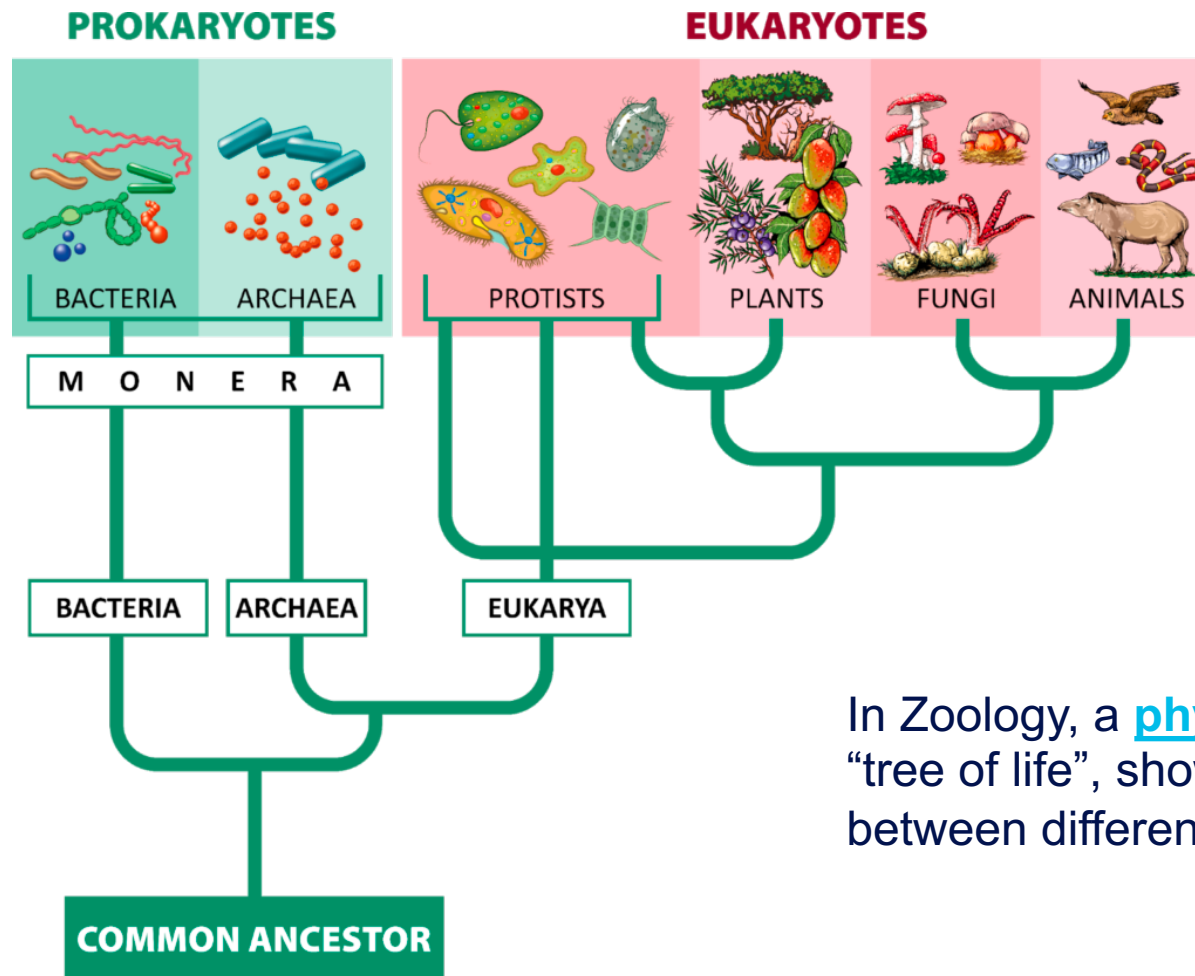
Performance

- **Objects are in memory**
- **Reads are fast, but**
- **Writes are slower due to writing disk log**
- **Read intensive workloads are best**
- **Objects should be small**
 - Written in Java
 - GC can become an overhead
- **What are objects?**



(Source: Shutterstock)

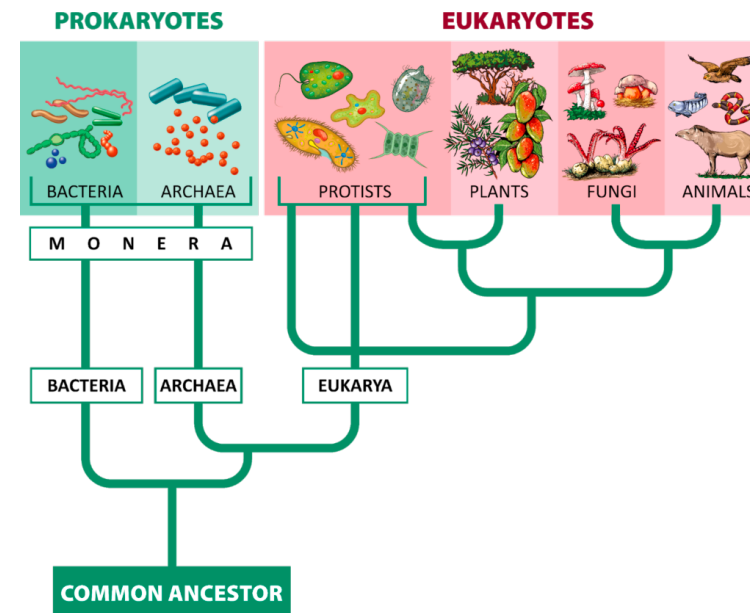
ZNODES



In Zoology, a [phylogenetic tree](#), or “tree of life”, shows the relationship between different species

Trees

- Appropriately (for a ZooKeeper) the only data structure is a tree (of life)
- Hierarchical namespace:
 - E.g. “/Animalia/Chordata/Mammalia/Marsupialia/Dasyuromorphia/Dasyuridae/Sarcophilus/TasmanianDevil”
- C.f. File system
 - Each node can have data and children
 - Nodes are called ZNODES
 - But “nodes” in an Ensemble are called “servers”
 - ZNODES have
 - Version control
 - Atomic reads/writes
 - Replicated (in ensemble)



ZNODES

- **Persistent or Ephemeral**
- **Persistent**
 - Default
 - Outlive client connection that created them
- **Ephemeral**
 - Transient
 - Exist only for the duration of the client session that created them
 - Useful for participant discovery and election triggering once client dies



(Source: Shutterstock)

Operations on ZNODES

■ Operations

- Create
- Delete
- Exists
- getData
- getChildren
- setData

■ Operations Are Low Level

- Just create and delete ZNODES in hierarchical namespace and
- Read and write data to/from them atomically
- Enable simple naming, configuration and group membership services
- Example recipes for higher operations



(Source: Shutterstock)

Wombats are also “low-level” –
They live under ground.

ZNODES: Watches

- **Clients can set watches on ZNODES**
- **Change to ZNODE triggers watch and sends notification to client**
- **But only once! A one-time trigger**
- **Set another watch for further notifications**



(Source: Shutterstock)

The Dining Philosophers Problem

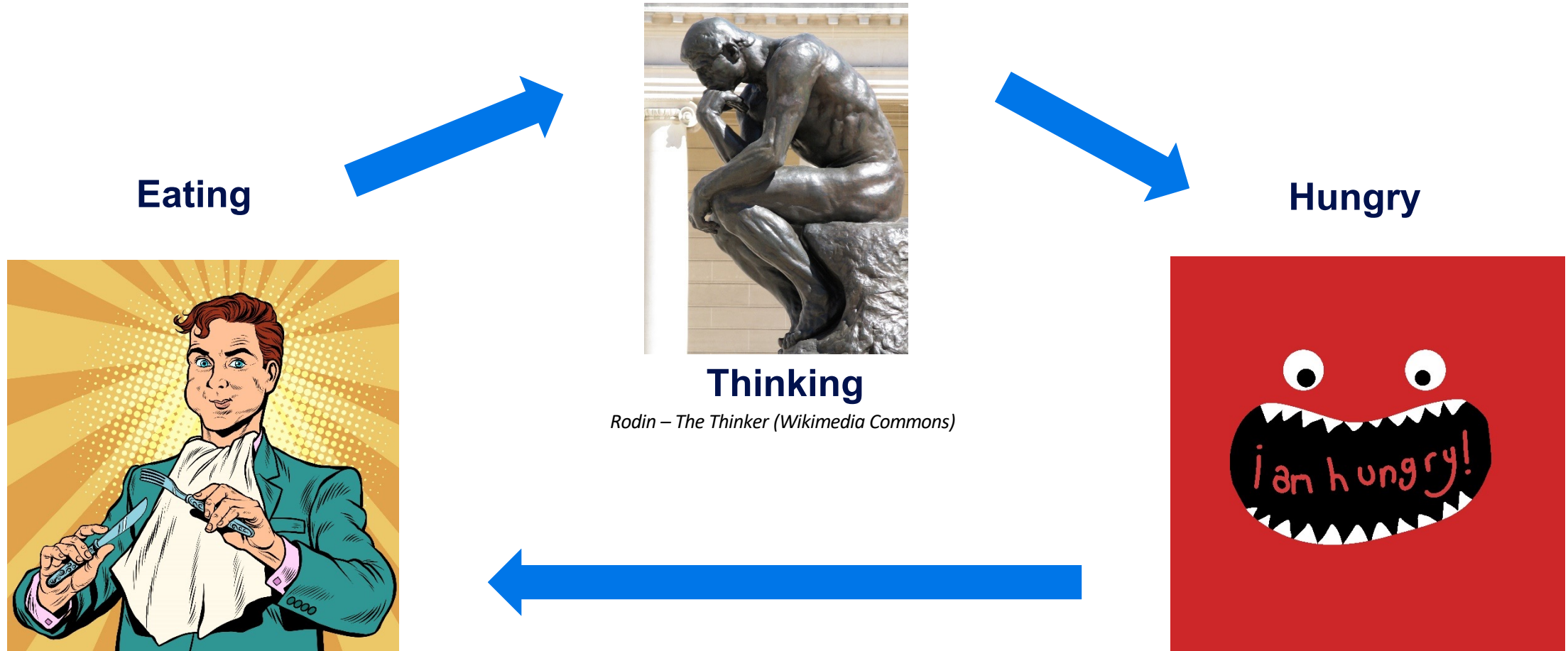


The Dining Philosophers Problem

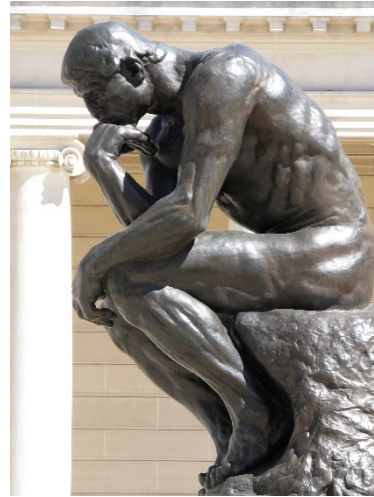
- Classic distributed systems concurrency and synchronization problem
- Dijkstra 1965
- Limited resources—Forks (1 per Philosopher)
- Multiple consumers (Philosophers)
- N Philosophers (5) seated around circular table
- Each has an (infinite) bowl of food in front of them
- Fork on left and right, same number as Philosophers
- To eat, a Philosopher must have **both left and right forks**
 - No talking allowed
 - No eating with fingers or only one fork (think of them as chopsticks)
 - No stealing forks from your neighbour (or anyone else) if they are holding it



Algorithm



Eating



Thinking

Rodin – The Thinker (Wikimedia Commons)

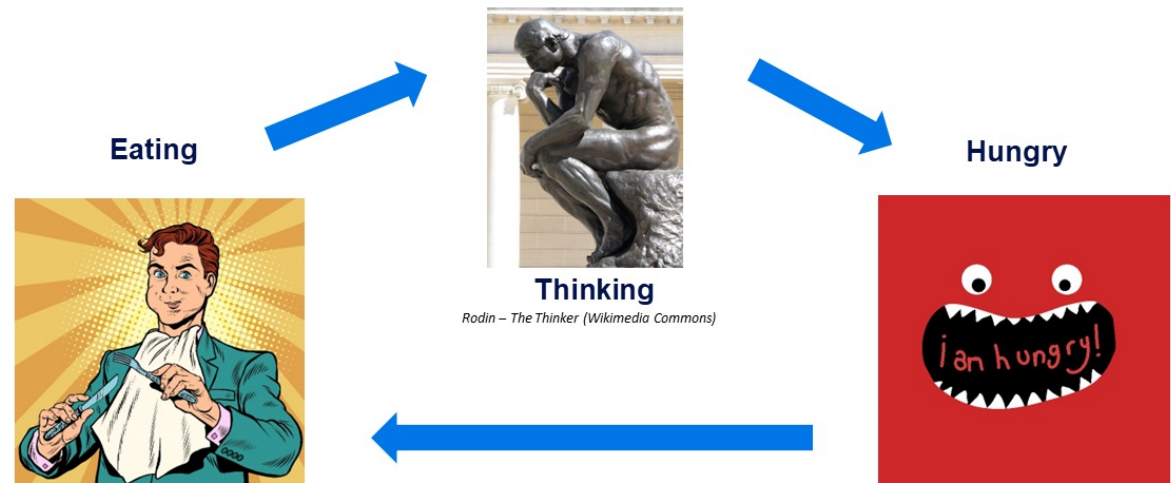
Hungry



Algorithm

Philosophers alternate between Thinking, Wanting to Eat (Hungry), and Eating

- Think
 - “Think” for a random period of time
- Hungry
 - Enter “Hungry” state
 - Wait until the left fork is free and take it
 - Wait until the right fork is free and take it
- Eating
 - “Eat” for a random period of time
 - Put both forks back on the table
- Start from Thinking again



Philosophers are silent, can't “negotiate” or “demand” forks

- Insufficient forks for everyone to be Eating at same time
- So some Philosophers will have to wait for one or both forks while Hungry

Problems and Solutions

■ Problems

- Starvation
- Deadlock
- Race Conditions
- Fairness

■ Good Solutions

- Minimize Hungry time
- Maximize Eating concurrency

■ Simplest Solutions

- Semaphores, timeouts around fork waiting, and a centralized Waiter (or ZooKeeper)
- But watch out as the waiter can become the bottleneck



(Source: Shutterstock)

My Version Has a Boss and Timeouts

Use leader election to elect a Boss Philosopher

- **Think**
 - Check if I am the leader, if I am, then announce a topic of my choice.
 - “Think” for a random period of time.
 - If I was the leader, give up the leadership.
- **Hungry**
 - Enter “Hungry” state.
 - **Wait (with timeout)** until the left fork is free and take it.
 - If I have the left fork then **wait (with timeout)** until the right fork is free and take it.
- **Eating**
 - If I have both forks, then “Eat” for a random period of time.
 - If I have any forks then put them back on the table.
 - Start from the top again.



(Source: Shutterstock)

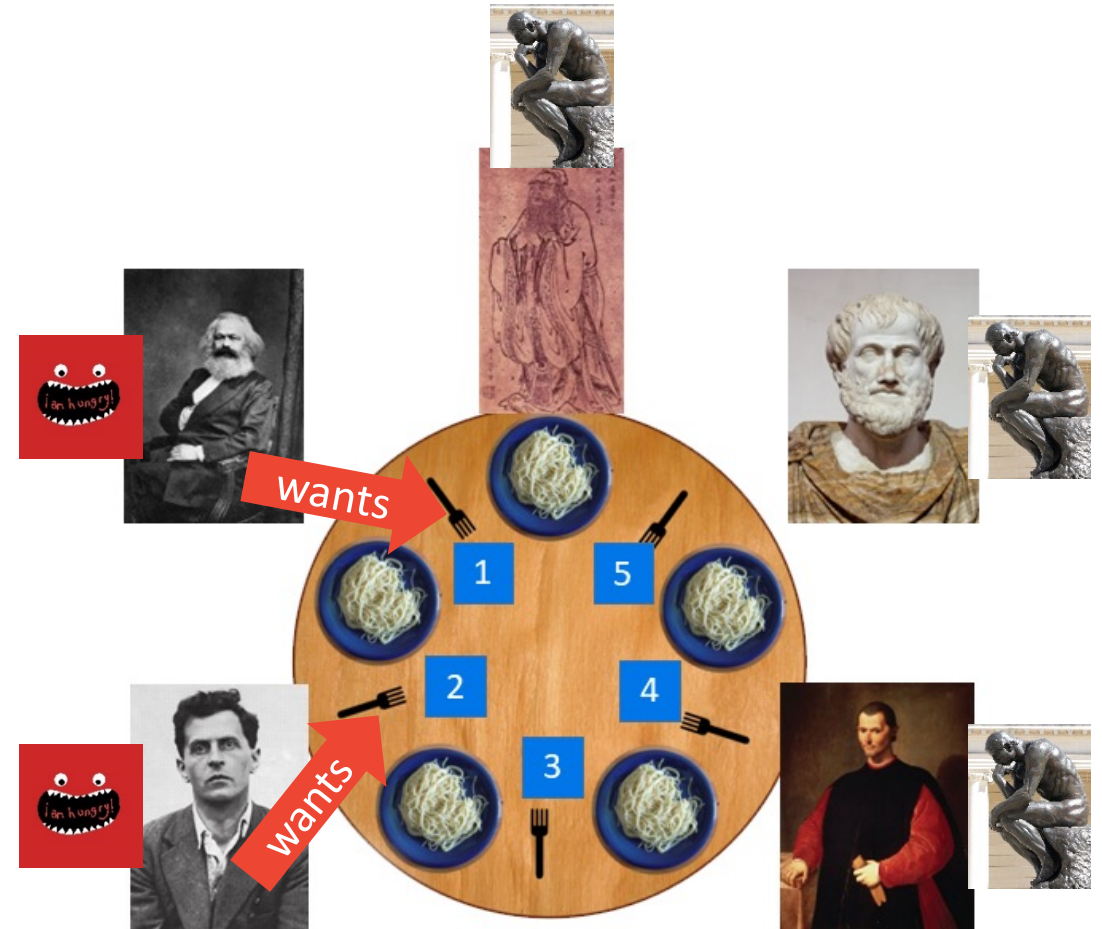
Example Trace

- Run Boss Election... [this takes some time]
- Marx is Thinking...
- Machiavelli is Thinking...
- Wittgenstein is Thinking...
- Aristotle is Thinking...
- Confucius is Thinking...



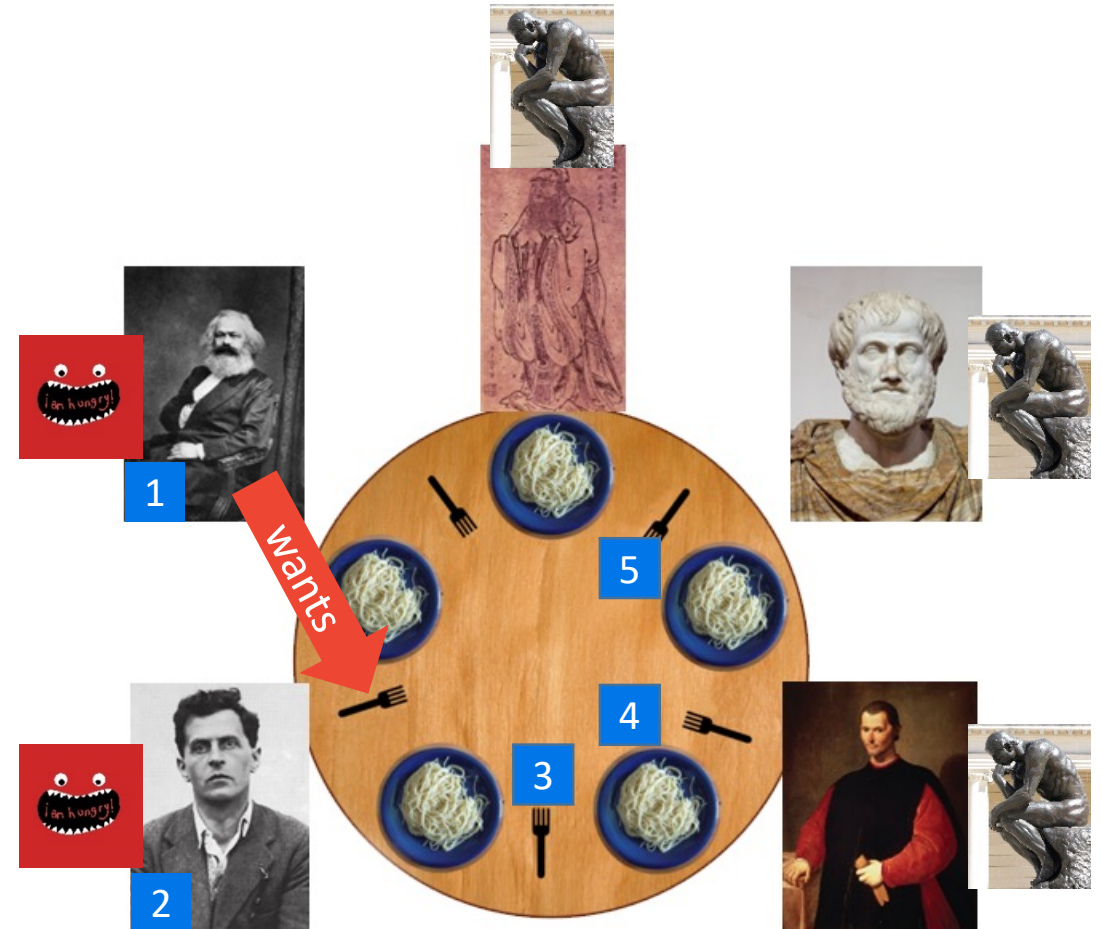
Example Trace

- Marx is Hungry, wants left fork 1
- Wittgenstein is Hungry, wants left fork 2



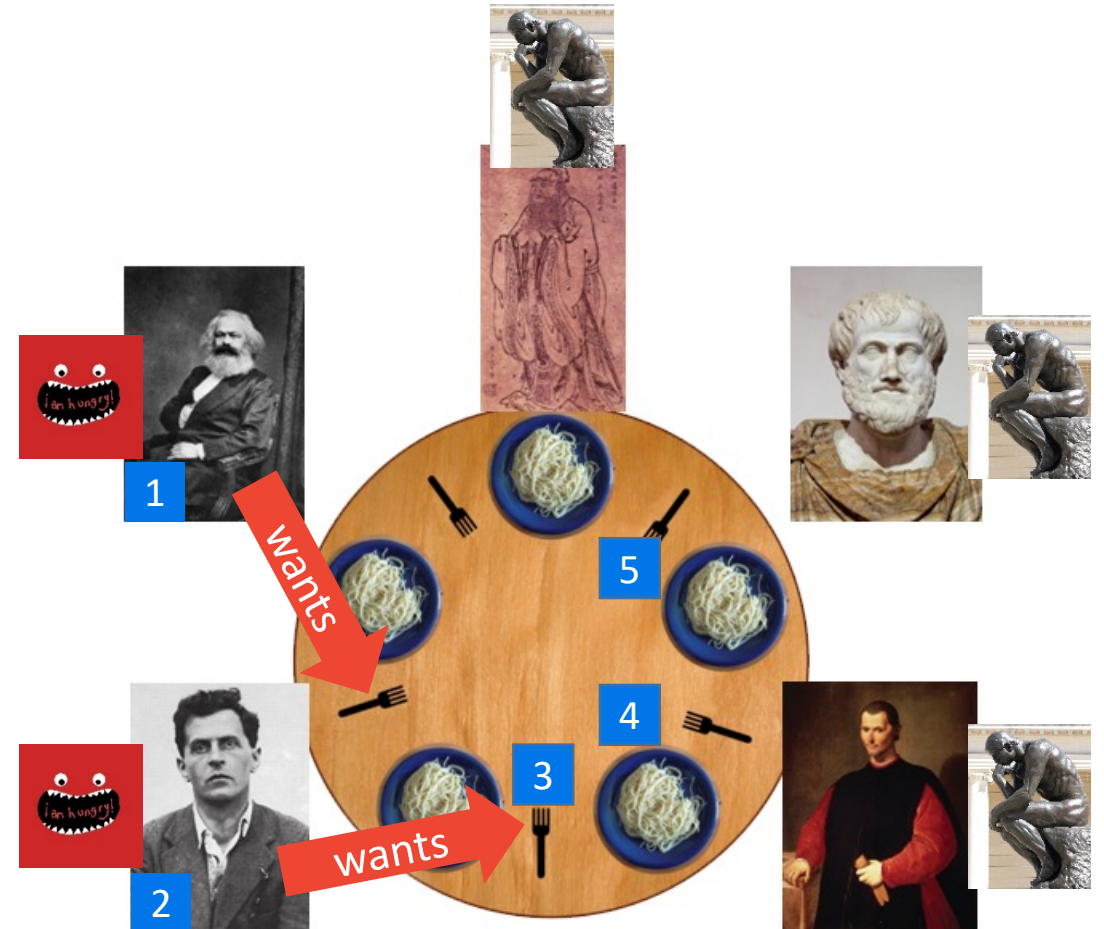
Example Trace

- Marx got left fork 1
- Marx wants right fork 2
- Wittgenstein got left fork 2
 - (Marx misses out!)



Example Trace

- Wittgenstein wants right fork 3



Example Trace

- Wittgenstein got right fork 3
- Wittgenstein is Eating



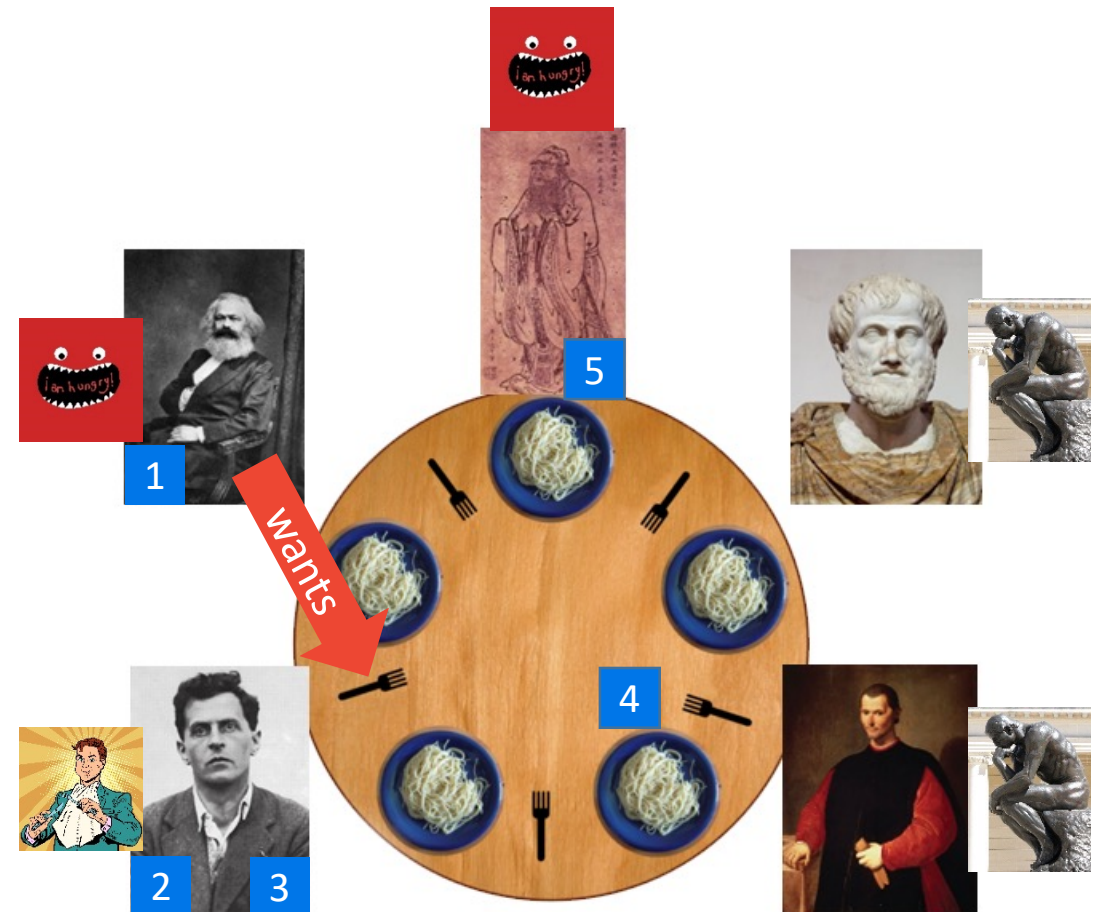
Example Trace

- Confucius is Hungry, wants left fork 5



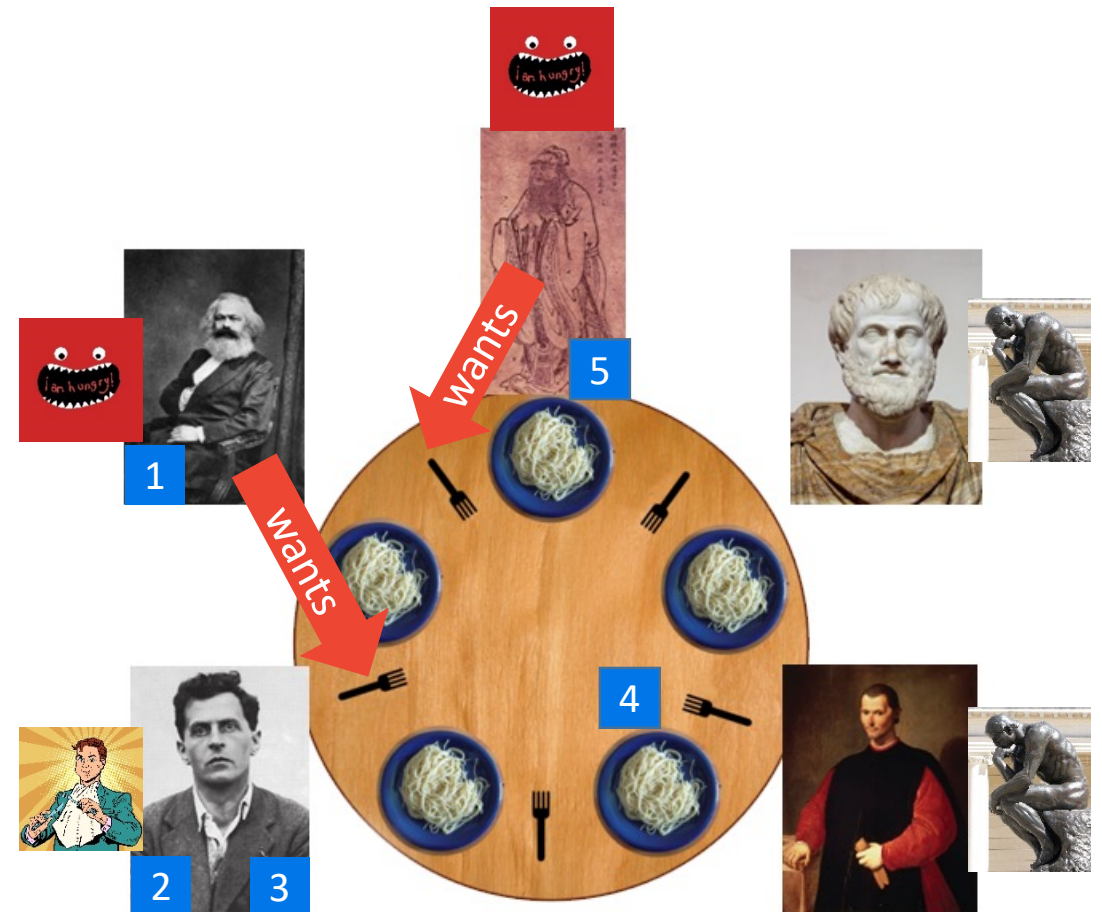
Example Trace

- Confucius got left fork 5



Example Trace

- Confucius wants right fork 1



Example Trace

- *** Marx gave up waiting for right fork 2
- Marx putting left fork back: 1
- Marx is Thinking...
- Marx is the BOSS!
- Marx says “Everyone now think about Aesthetics”



Example Trace

- Confucius got right fork 1
- Confucius is Eating



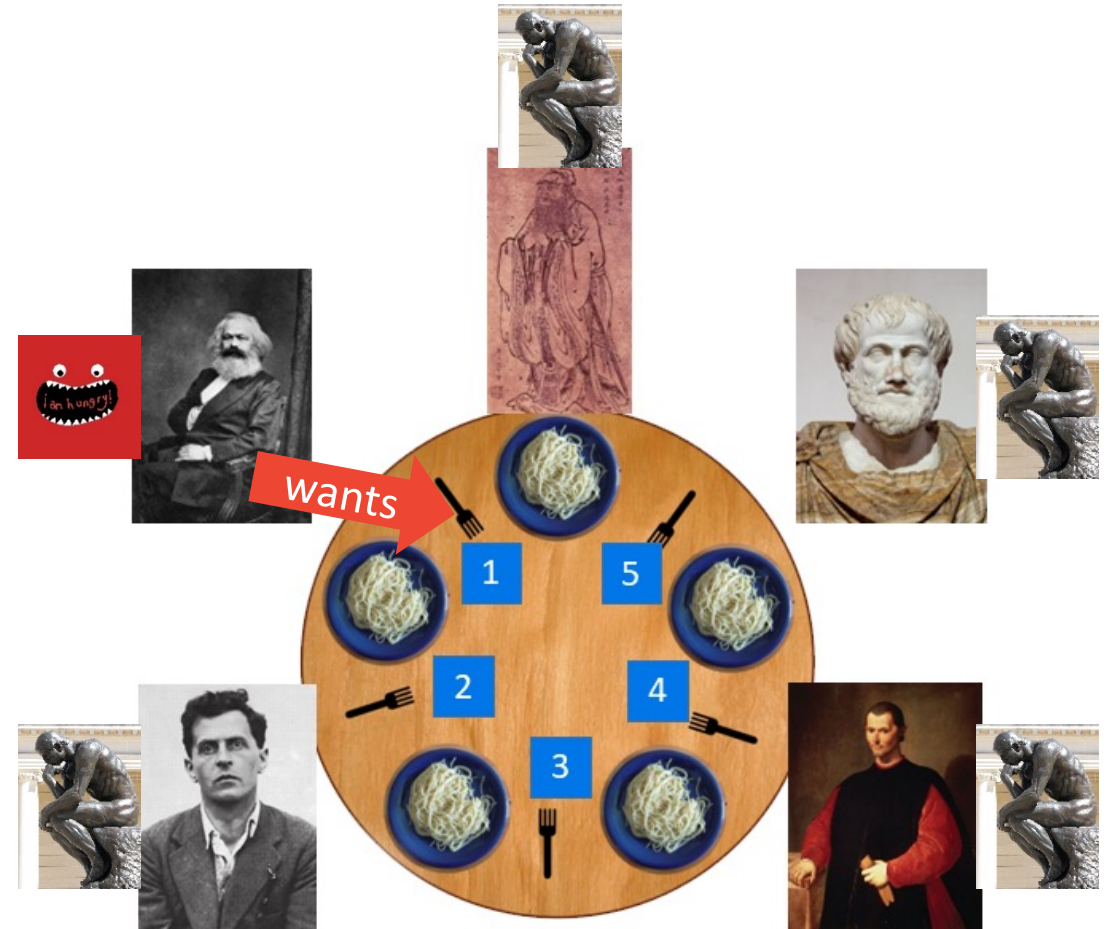
Example Trace

- Confucius finished eating!
- Putting both forks back: 5, 1
- Confucius is Thinking...
- Wittgenstein finished eating!
- Putting both forks back: 2, 3
- Wittgenstein is Thinking...



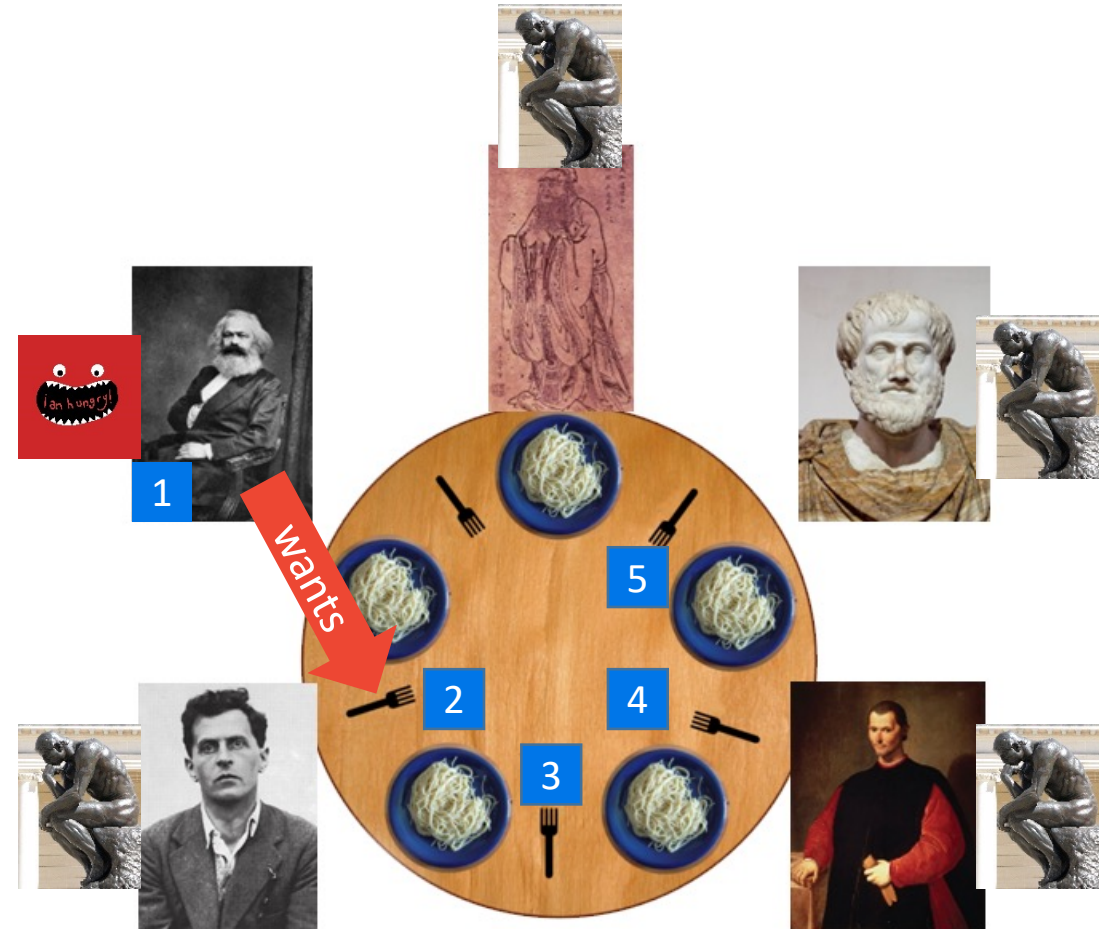
Example Trace

- Marx is no longer the BOSS
- Run Boss Election
- Marx is Hungry, wants left fork 1



Example Trace

- Marx got left fork 1
- Marx wants right fork 2



Marx Is Finally Happy

- Marx got right fork 2
- Marx is Eating (finally!!!)



Implementation with Apache Curator

A Curator also walks into a Pub...



**Curators organize and interpret exhibits—
even Zoo animals and Zookeepers!**

Implementation with Apache Curator



Dangerous Animals:

- Snakes
- Stingers
- Crocs
- Spiders
- Sharks
- Bluebottles
- Shells
- Ticks
- Ants
- Centipedes
- Cassowaries...



(Source: Shutterstock)



Don't Panic!

You are actually more likely to be injured by a horse in Australia

Apache Curator

- High-level Java client for Apache ZooKeeper

Koalas are also “high-level” – they live in trees



(Source: Shutterstock)



Apache Curator



▪ Lots of recipes

- Elections
- Locks
- Barriers
- Counters
- Catches
- Nodes/Watches
- Queues



Koalas only need 1 recipe: Eucalyptus leaves



(Source: Shutterstock)



Create and Start Client

```
int sleepMsBetweenRetries = 100;
int maxRetries = 3;
String zk = "127.0.0.1:2181";
RetryPolicy retryPolicy = new
RetryNTimes(maxRetries, sleepMsBetweenRetries);

CuratorFramework client = CuratorFrameworkFactory
    .newClient(zk, retryPolicy);
client.start();
```



Curator Recipes Used

- **Leader Latch for elections**
- **Shared Lock for forks**
- **Shared Counter for metrics**



Leader Latch



```
// For each Philosopher thread
LeaderLatch leaderLatch = new LeaderLatch(client, "/mutex/leader/topic", threadName);
leaderLatch.start();
// Think
if (leaderLatch.hasLeadership())
    System.out.println("I'm now the BOSS, think about X");
// Think for a while ...
// If thread was the leader then relinquish leadership, but rejoin pool again.
if (leaderLatch.hasLeadership())
{
    System.out.println("I'm no longer the BOSS");
    leaderLatch.close();
    leaderLatch = new LeaderLatch(client, "/mutex/leader/topic", threadName);
    leaderLatch.start();
}
// Hungry, try to eat, etc.
```



Shared Lock for Fork Sharing

```

int numForks = 5; // number of forks = number of Philosophers
static InterProcessSemaphoreMutex[] forks = new InterProcessSemaphoreMutex[numForks];

// Create forks
for (int i=0; i < numForks; i++)
    forks[i] = new InterProcessSemaphoreMutex(client, "/mutex/fork" + i);

// In each Philosopher thread:
// Hungry, want left fork
int leftFork = 0;
boolean gotLeft = forks[leftFork].acquire(maxForkWaitTime, TimeUnit.MILLISECONDS);

// to release a fork
forks[leftFork].release();

```



(Source: Shutterstock)

Shared Counter for Metrics

(using optimistic concurrency – try until success)

```
int meals = new SharedCount(client, "/counters/meals", 0);
try {
    meals.start();
    meals.setCount(0);
} catch (Exception e1) { e1.printStackTrace(); }
...
// In each Philosopher thread:
// Each time a Philosopher eats, increment the meal counter
// trySetCount returns true if it succeeds, else false and try again
boolean success = false;
while (!success) {
    int before = meals.getCount();
    success = meals.trySetCount(before + 1); }
...
// At the end of Dining, print out total meals
System.out.println("Total meals = " + meals.getCount());
```



(Source: Shutterstock)

Results: Baseline, Single ZK Server

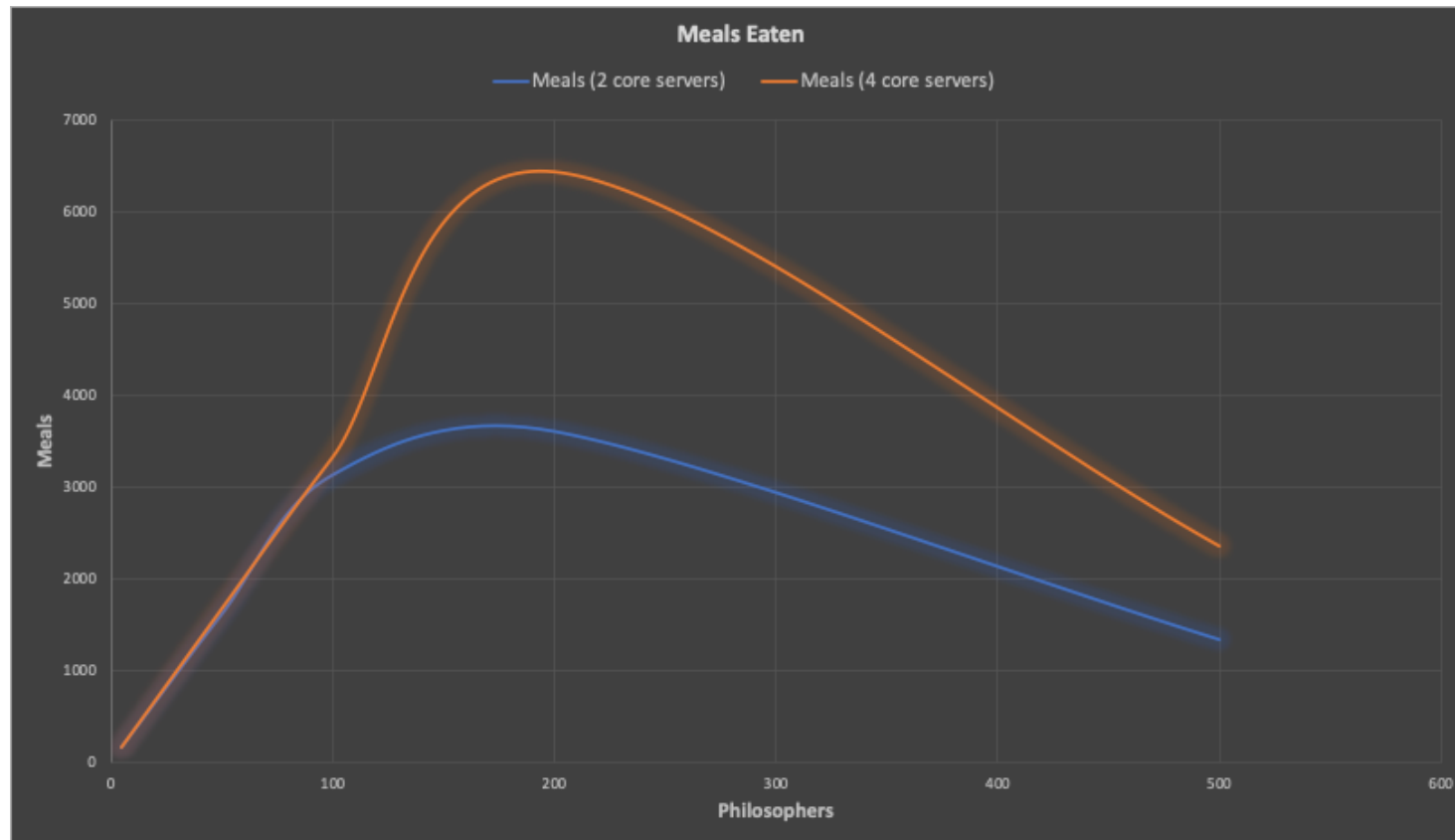
- **5 Philosophers, very low load on server so “best case” for this version of the algorithm**
- **Utilization for each state:**
 - Philosophers Thinking = 61%
 - Philosophers Hungry = 12%
 - Philosophers Eating = 27%
- **In “perfect” world with no fork contention**
 - Thinking and Eating would be 50% and Hungry 0%

Results: 3 server ZK Ensemble



(Source: Maritza Rios – Wikimedia Commons)

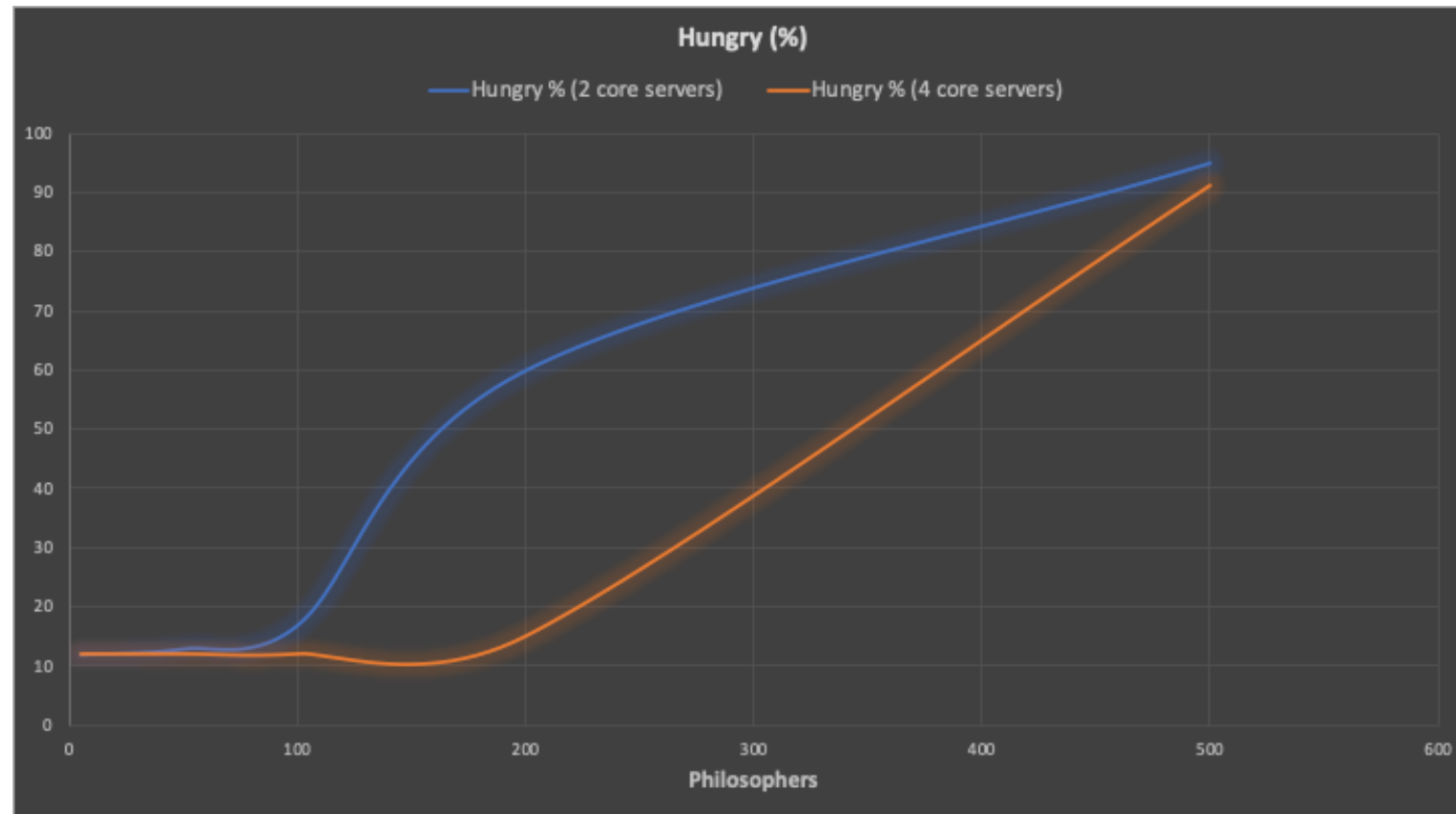
Load Testing on Ensembles



- 3 x large (2 vcpus) vs. xlarge (4 vcpus) EC2 servers (6 cores vs. 12 cores)
- 3600 meals (peak at 150 Philosophers) vs 6400 meals (peak at 200 Philosophers)

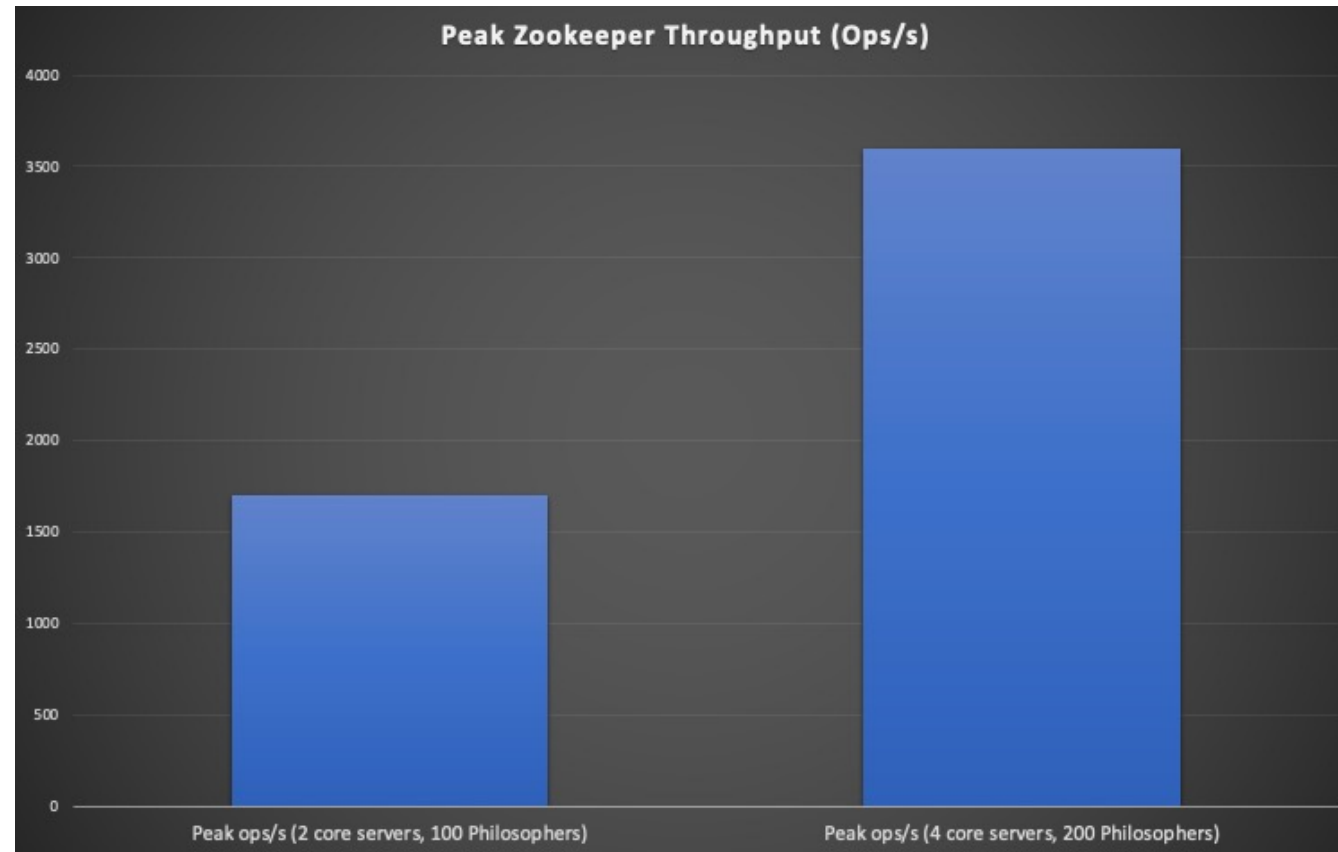
Increase in Hungry %:

Faster Increase for 2 cores



Zookeeper Throughput:

1,700 ops/s c.f. 3,600 ops/s





- **At maximum capacity, leader server saturated, replicas had spare capacity**
 - As write heavy workload
- **Availability**
 - Works as advertised
 - Killing 1/3 server, leader failed over immediately to replica
 - Killing 2/3 servers, Curator client error

Kafka leaves the Zoo(Keeper)

- **Kafka currently uses ZK to**
 - Store partition and broker metadata
 - Elect a broker as Kafka Controller
- **KIP-500 Replace ZK with self-managed Metadata Quorum KIP-595**
 - Kafka should be more scalable, faster and support millions of partitions (!)
- **Other projects also leaving ZooKeeper?**
 - Flink, Pinot
 - Cloud native deployments use Kubernetes etcd service
- **ZooKeeper vs. KRaft**
 - Latest results!
 - Performance Engineering Track, Thursday after lunch

(Source: Wikipedia)



(Source: Shutterstock)

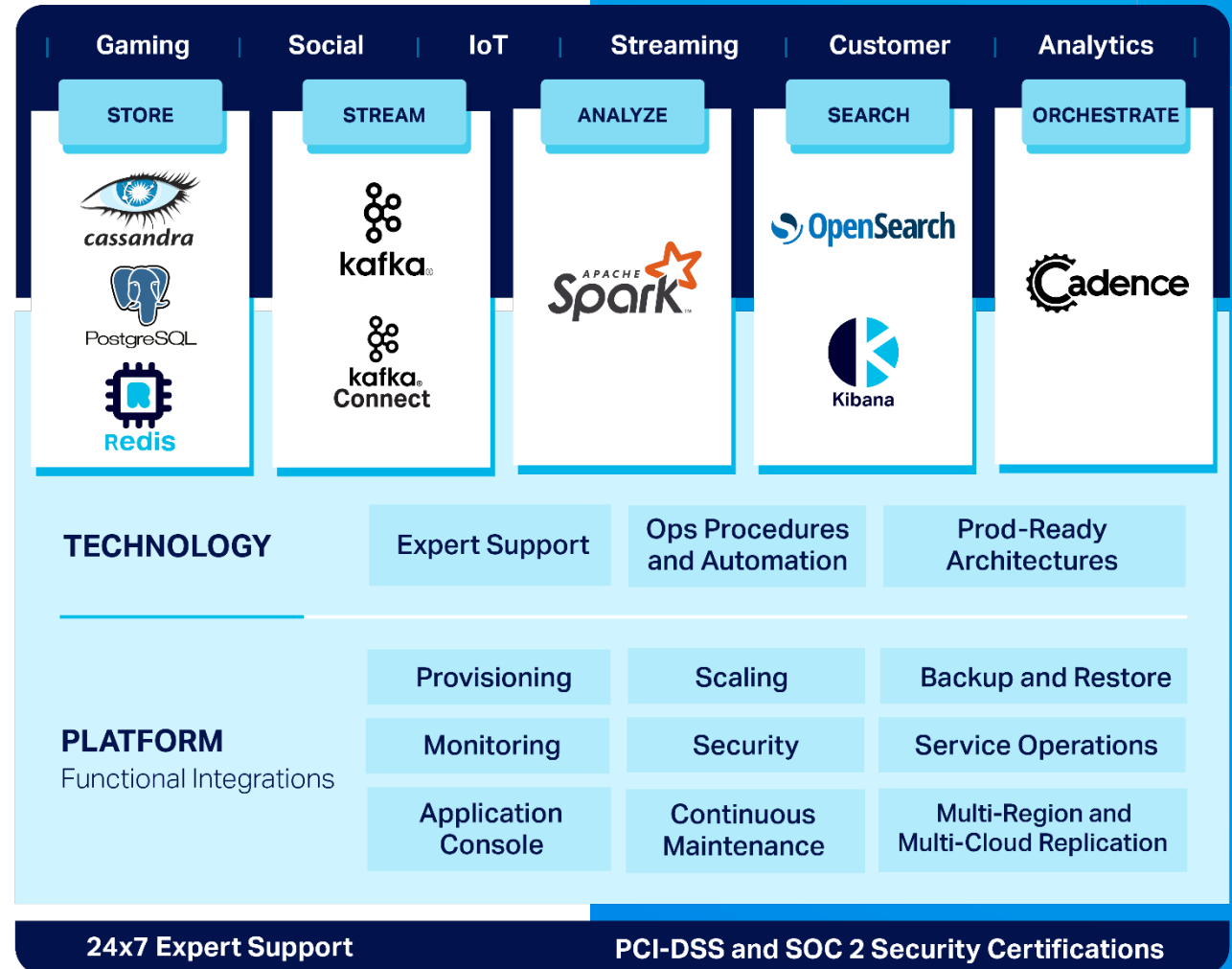
Instaclustr Managed Platform

A complete ecosystem to support your mission critical application.

Our other Managed Open Source Technologies +



APACHE
ZooKeeper[™]



Further Information

1

Blog: www.instaclustr.com/blog/apache-zookeeper-meets-the-dining-philosophers/

2

All My Blogs: www.instaclustr.com/paul-brebner/

3

Managed ZooKeeper: www.instaclustr.com/platform/managed-apache-zookeeper/



THANK YOU!

instacluster



www.instacluster.com



info@instacluster.com



[@instacluster](https://www.linkedin.com/company/instacluster)

© Instacluster Pty Limited, 2022

<https://www.instacluster.com/company/policies/terms-conditions/>

Except as permitted by the copyright law applicable to you, you may not reproduce, distribute, publish, display, communicate or transmit any of the content of this document, in any form, but any means, without the prior written permission of Instacluster Pty Limited

