A Way Forward: Enabling Operating System Innovation in the Cloud

Dan Schatzberg, James Cadden, Orran Krieger, Jonathan Appavoo
Boston University

There has been a lot of
innovation in cloud computing

In contrast, there has been relatively little innovation in the operating system

Do operating systems matter?

Do operating systems matter? Yes!

Memcached Latency

Component	Description	Mean latency (μs)	99 %ile latency (μs)	Overall share
DC Fabric	Propagation delay	< 1	-	-
	Single Switch	1-4	40-60	1%
	Network Path†	6	150	7 %
Endhost	Net. serialization	1.3	1.3	1.4 %
	DMA	2.6	2.6	3 %
	Kernel (incl. lock contention)	**76**	**1200-2500**	**86-95** %
Application	Application*	2	3	2 %
	Total latency	88	1356-2656	100 %

Reproduced from Kapoor et al. Chronos predictable low latency for data center applications. SOCC 2012

How can we enable OS innovation in the cloud?

Modify Existing OS

Clean Slate

Modify Existing OS

Pros

✓ Large software ecosystem

Clean Slate

Modify Existing OS

Pros

✓ Large software ecosystem

Cons

- Large and complex

Clean Slate

Modify Existing OS

Pros

✓ Large software ecosystem

Cons

- Large and complex

Clean Slate

✓ Complete control

Modify Existing OS

Pros

✓ Large software ecosystem

Cons

- Large and complex

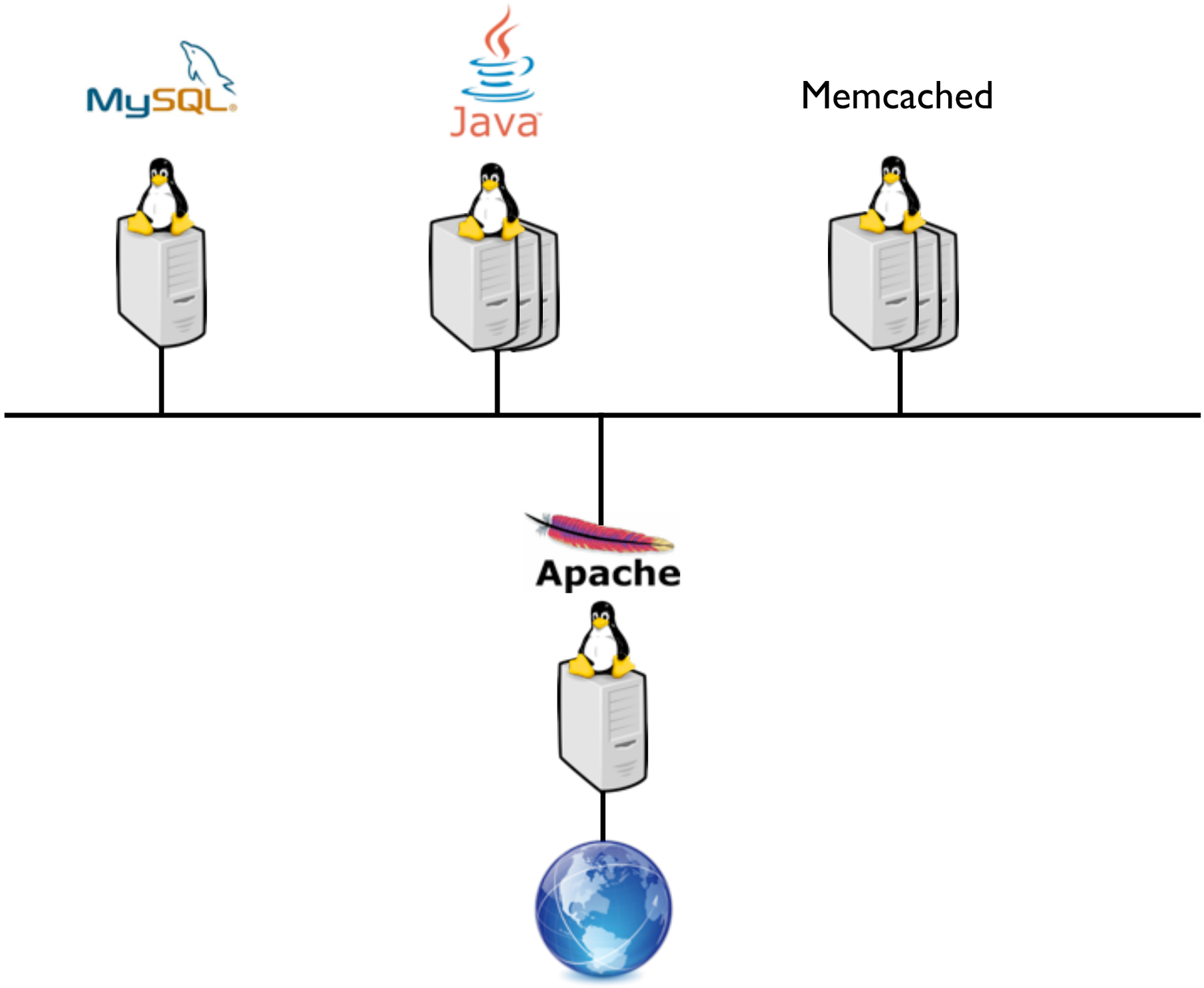
Clean Slate

✓ Complete control

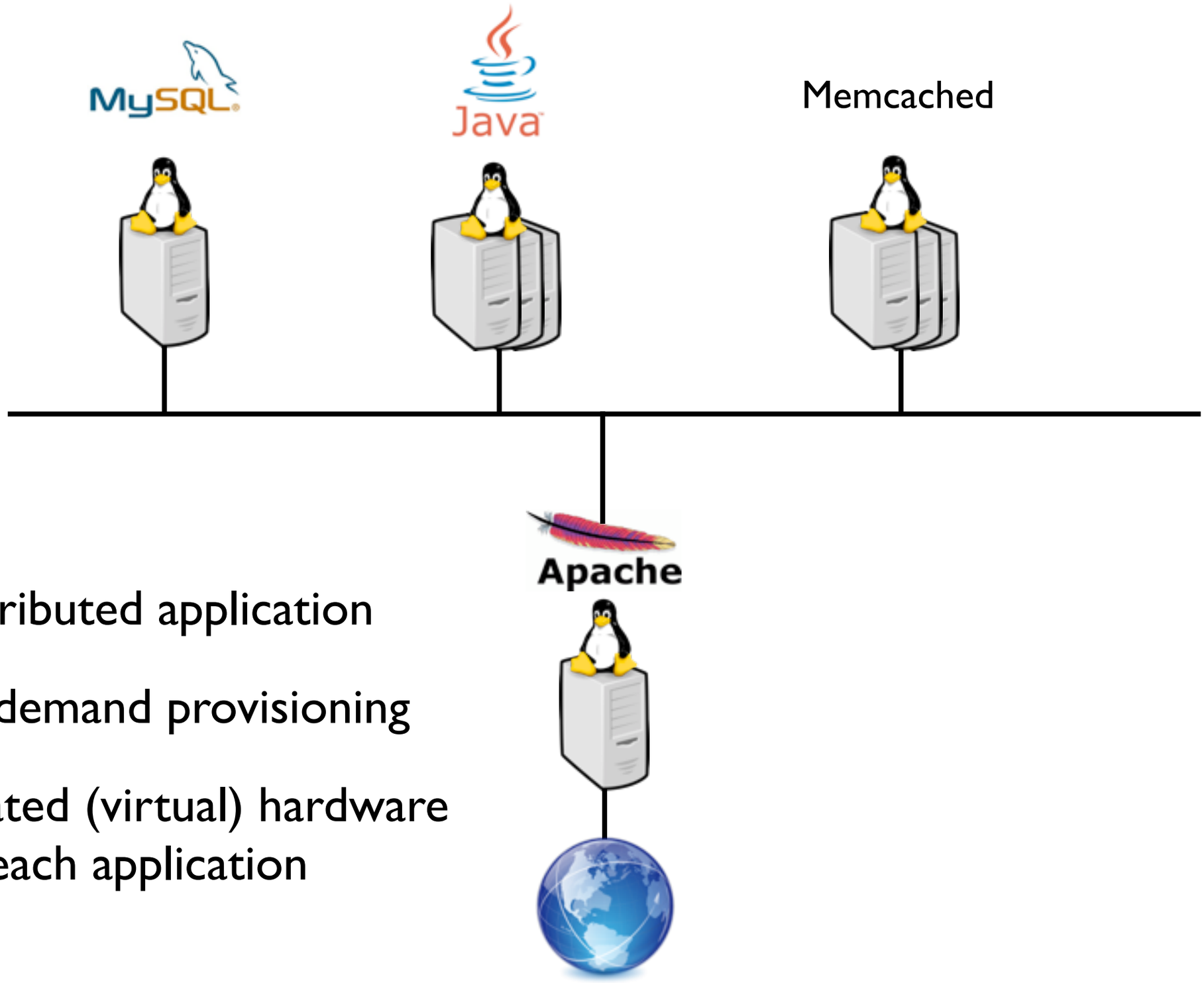
- Need to reproduce legacy functionality

A Way Forward

Typical Cloud Web Application



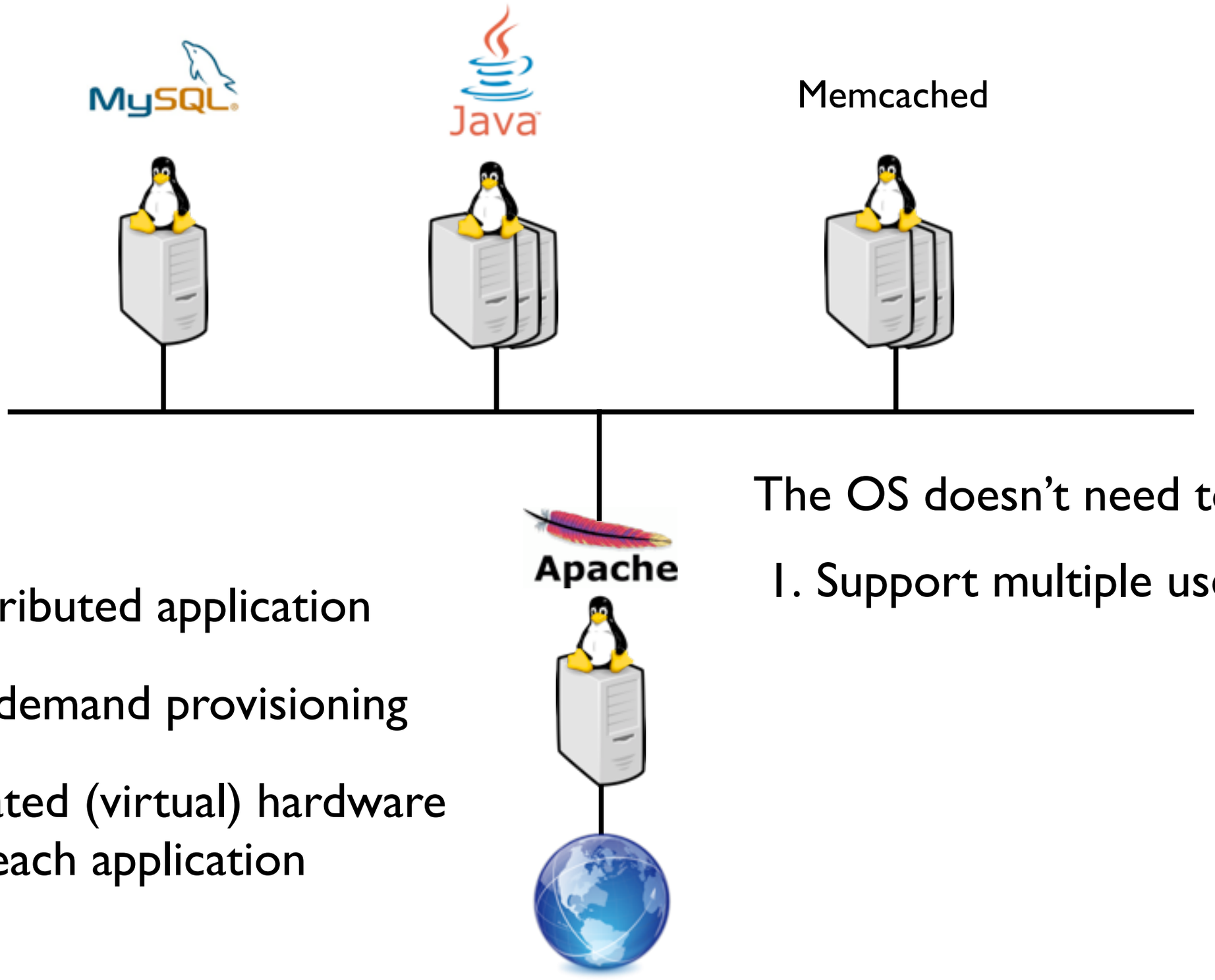
Typical Cloud Web Application



- Distributed application
- On demand provisioning
- Isolated (virtual) hardware for each application

What is the role of the operating system?

Typical Cloud Web Application

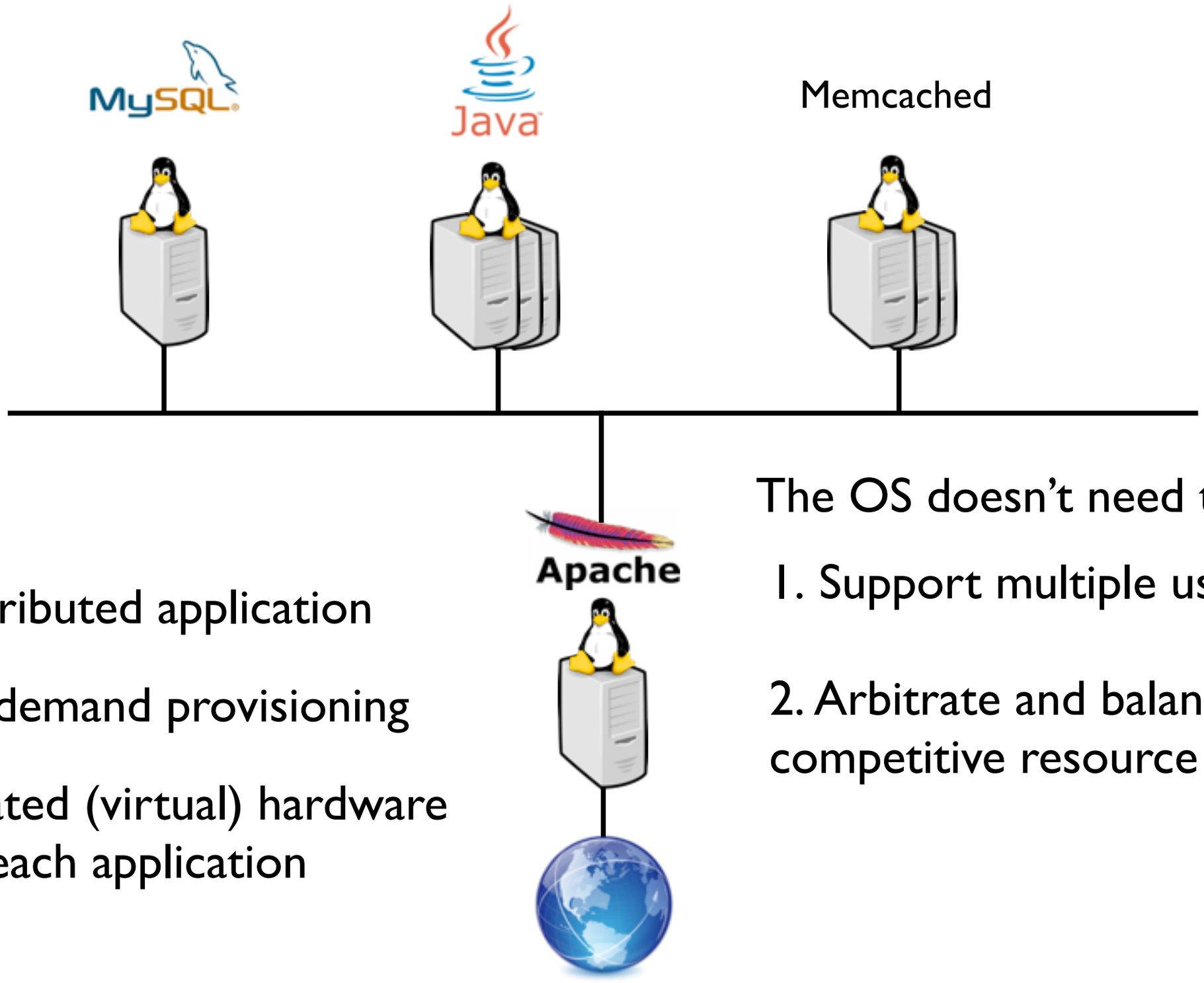


- Distributed application
- On demand provisioning
- Isolated (virtual) hardware for each application

The OS doesn't need to:

1. Support multiple users

Typical Cloud Web Application



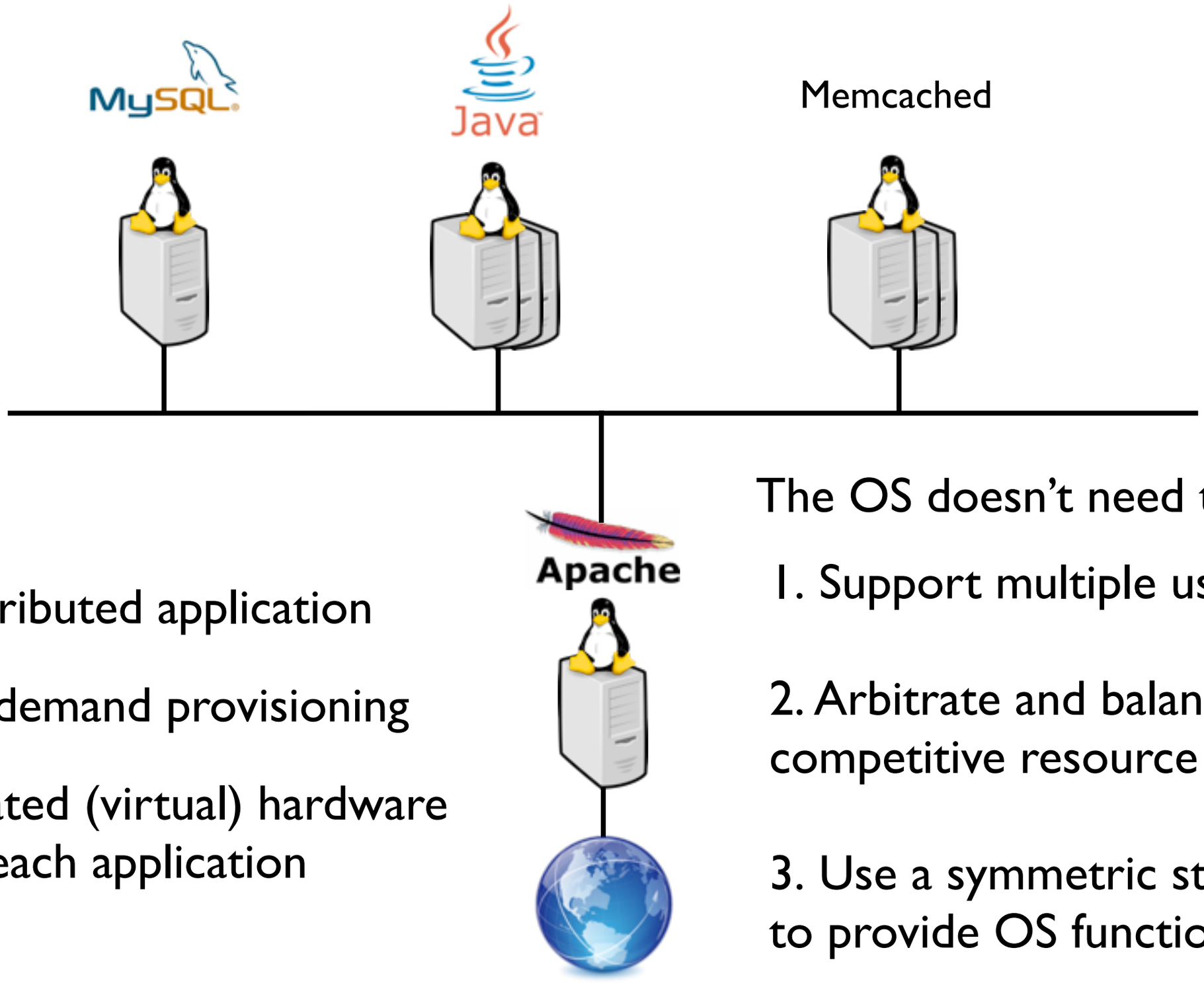
- Distributed application
- On demand provisioning
- Isolated (virtual) hardware for each application

The OS doesn't need to:

1. Support multiple users

2. Arbitrate and balance competitive resource usage

Typical Cloud Web Application



- Distributed application
- On demand provisioning
- Isolated (virtual) hardware for each application

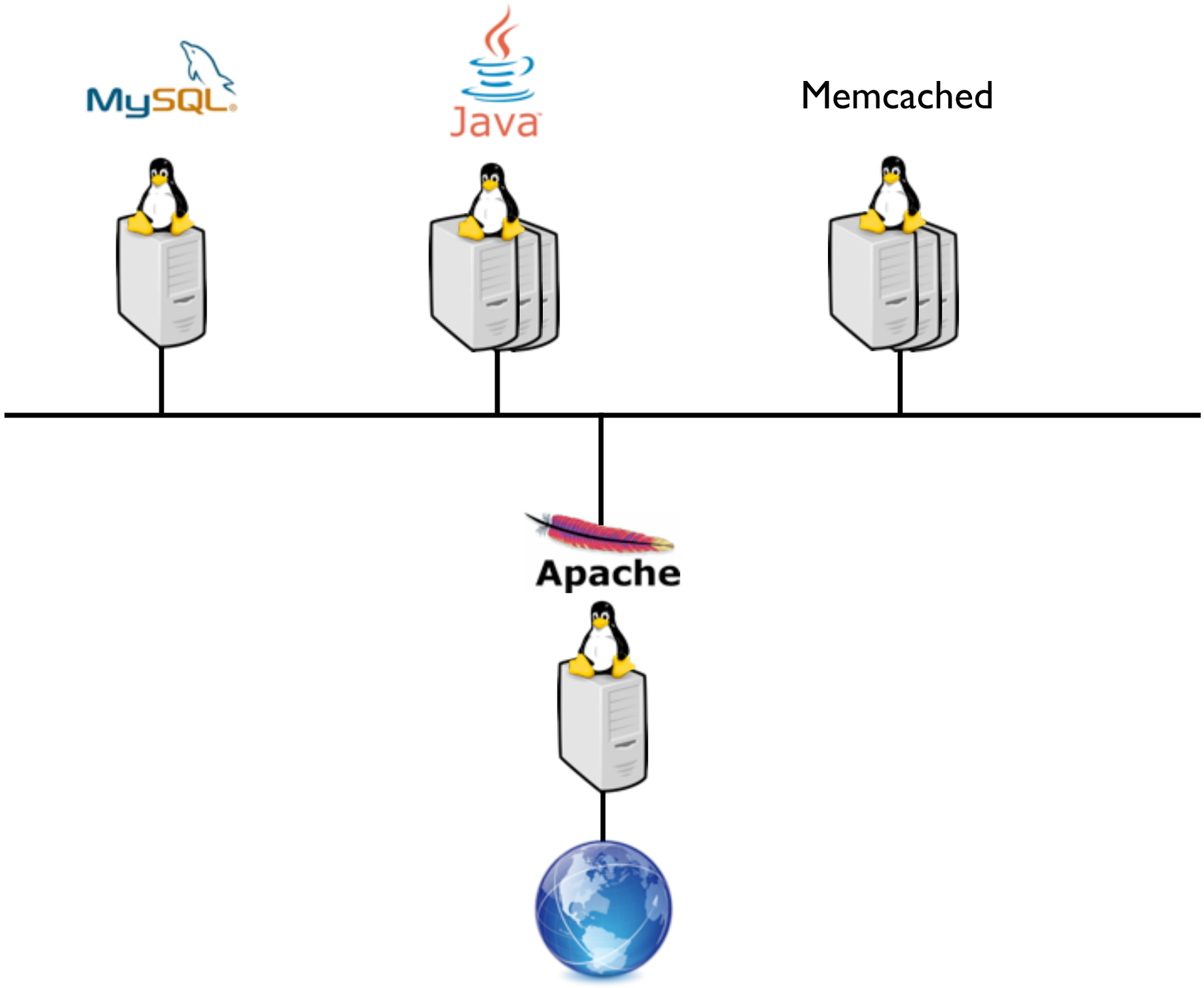
The OS doesn't need to:

1. Support multiple users

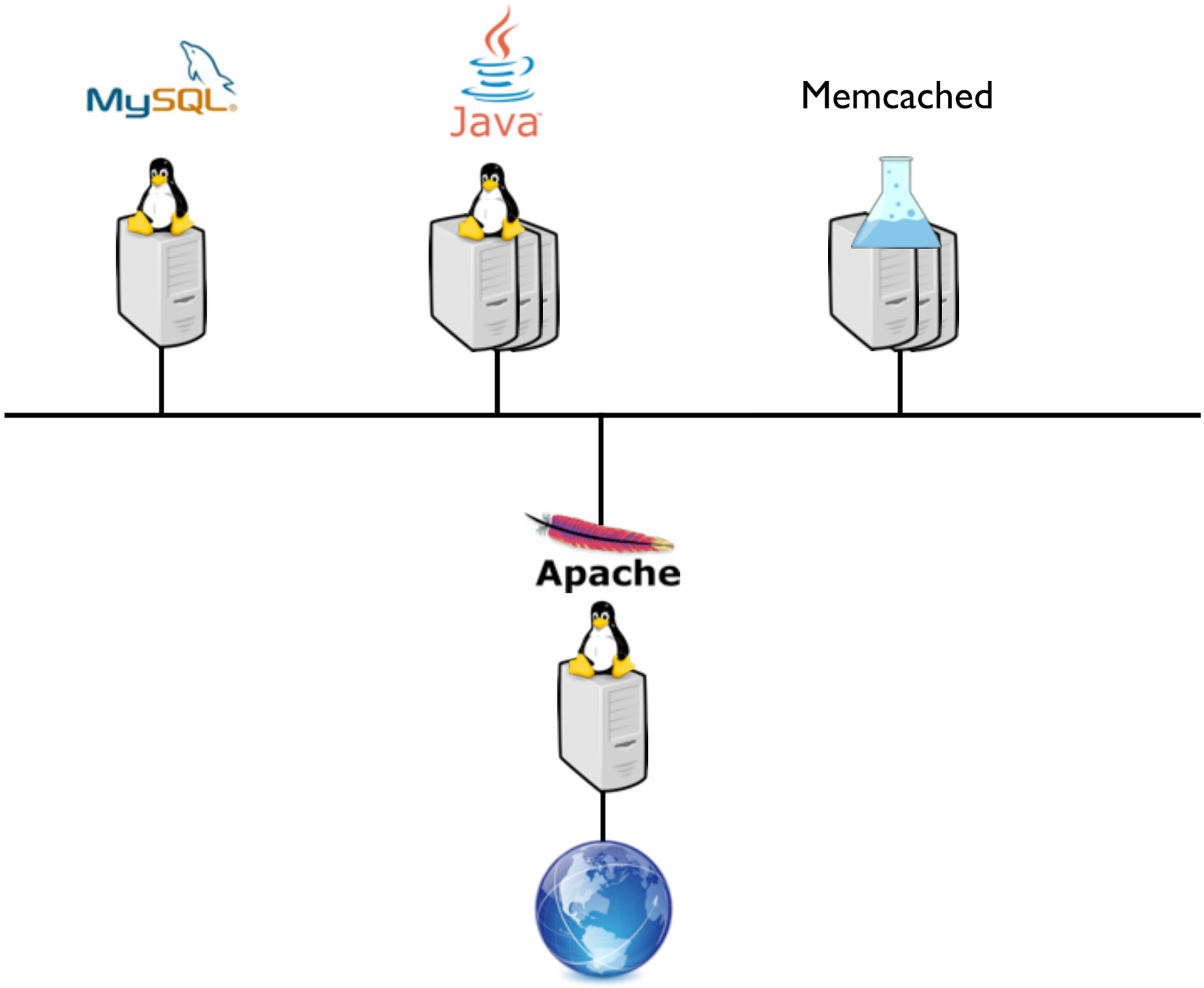
2. Arbitrate and balance competitive resource usage

3. Use a symmetric structure to provide OS functionality

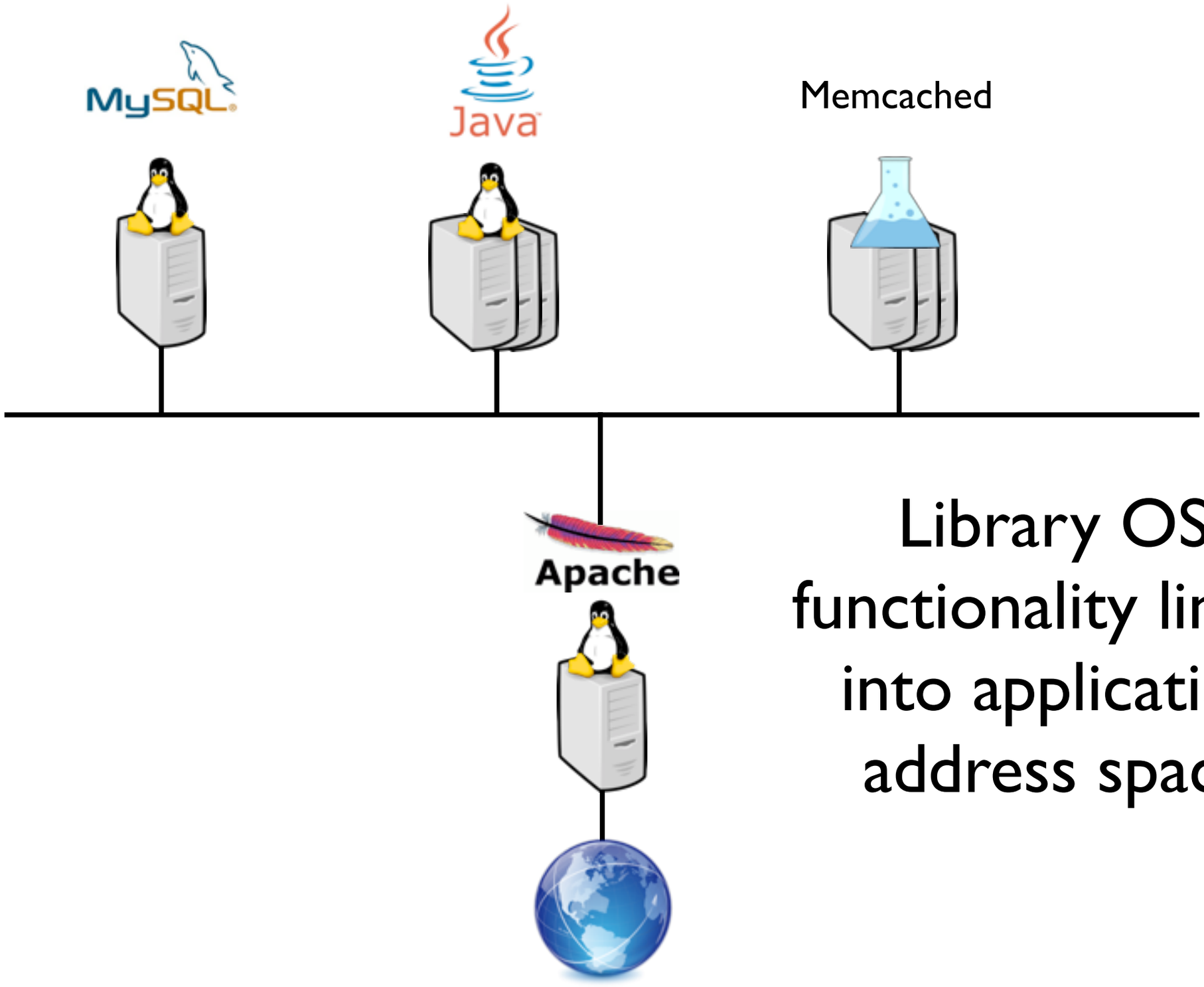
Typical Cloud Web Application



Typical Cloud Web Application

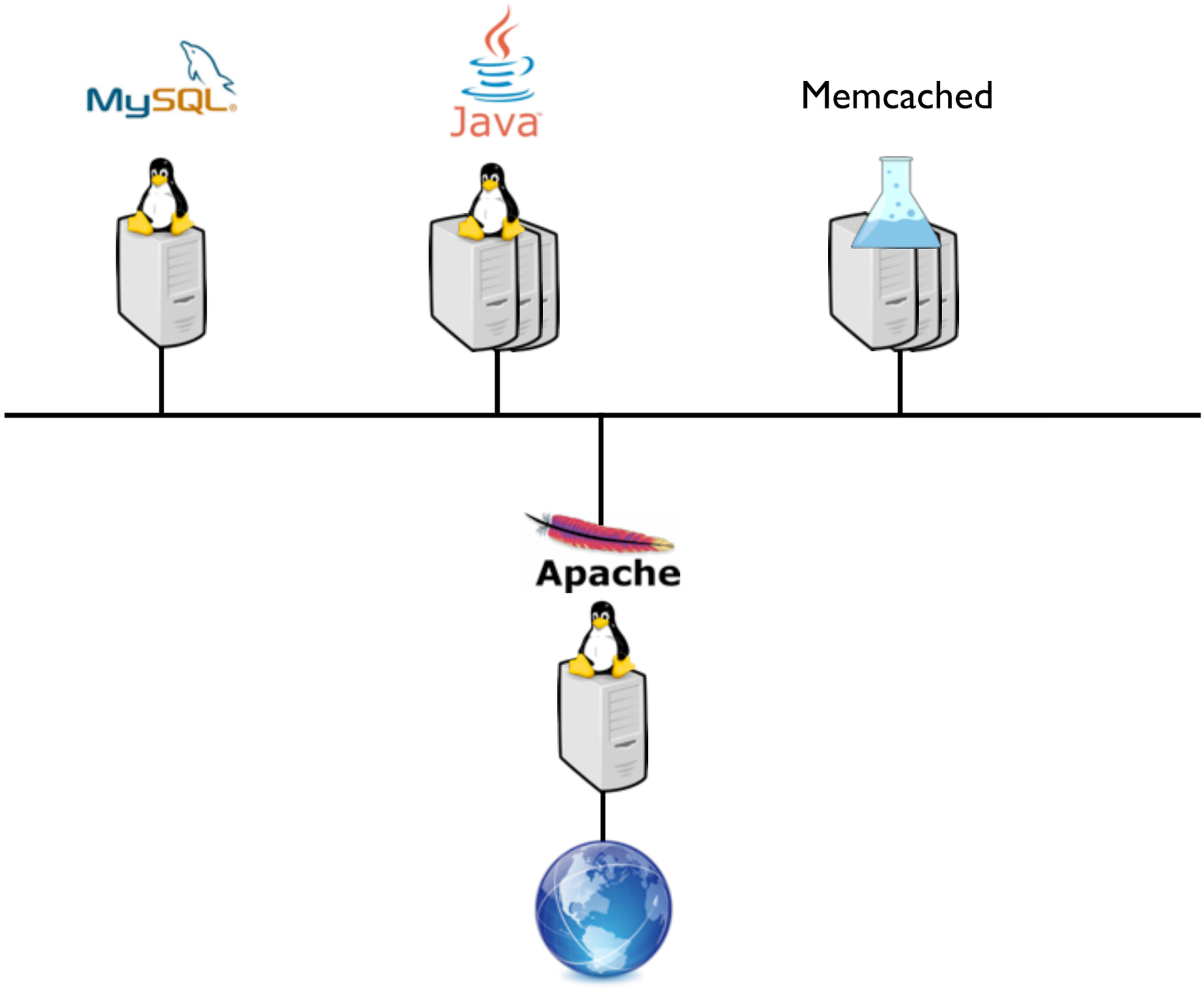


Typical Cloud Web Application

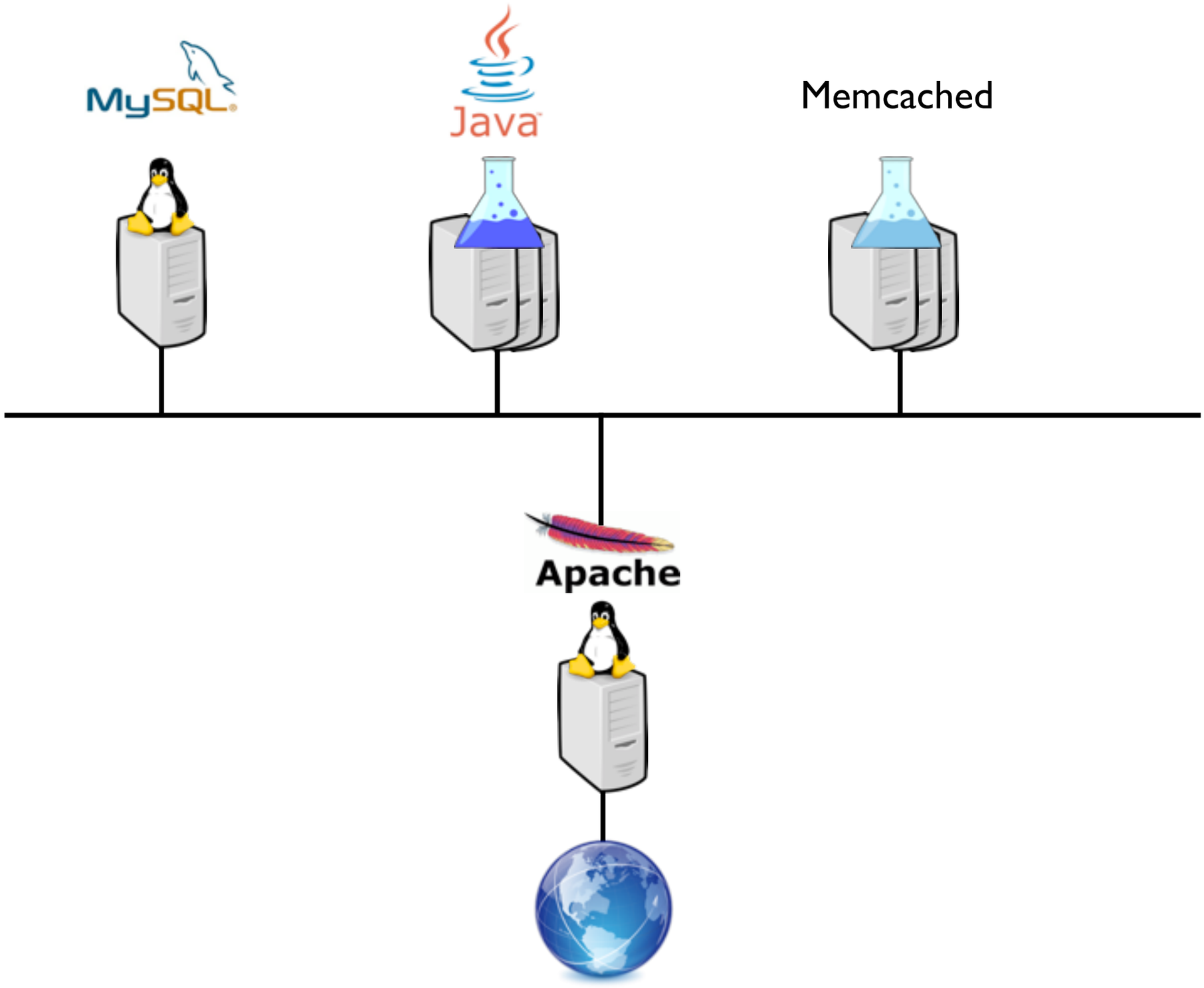


Library OS: functionality linked into application address space

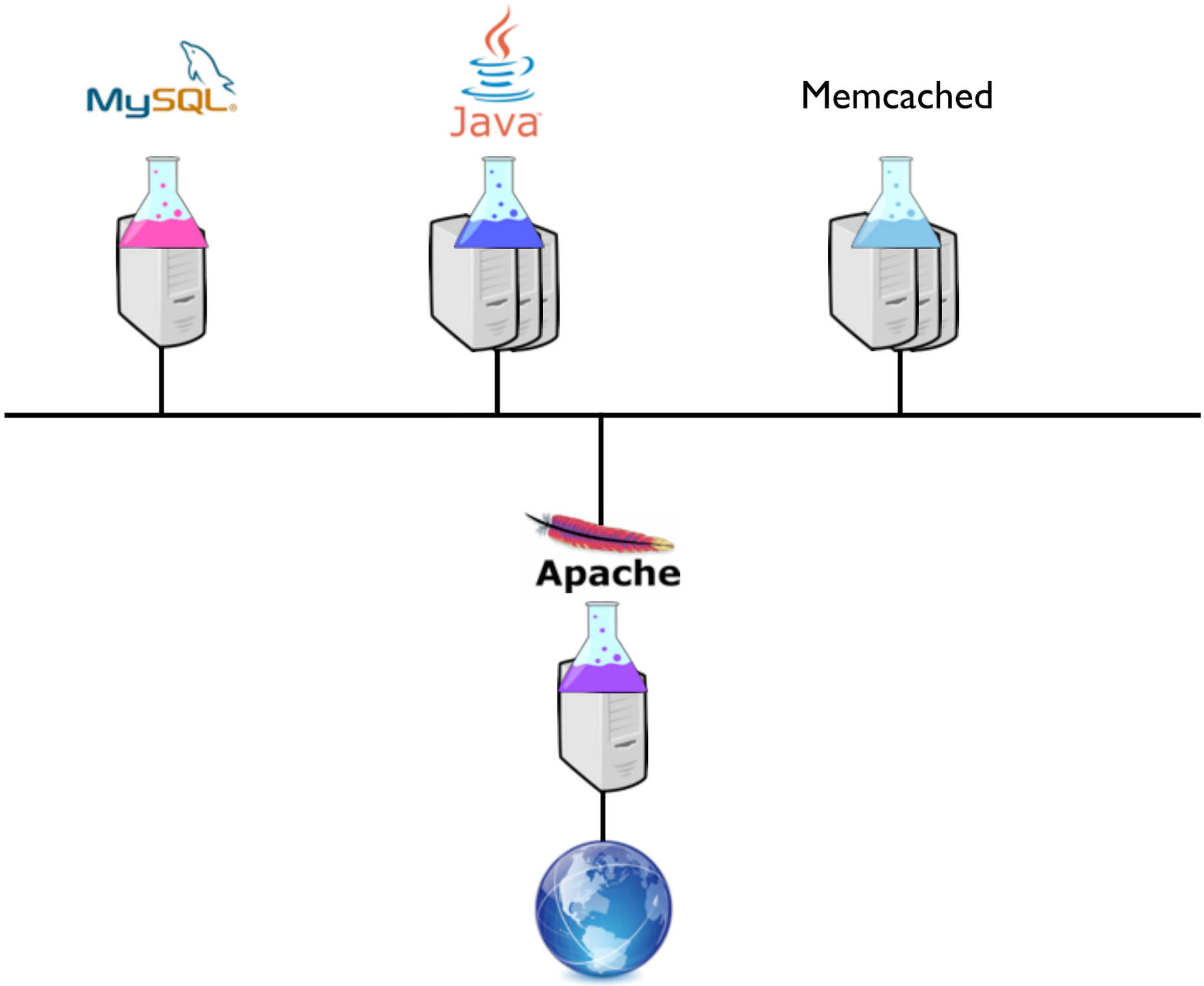
Typical Cloud Web Application



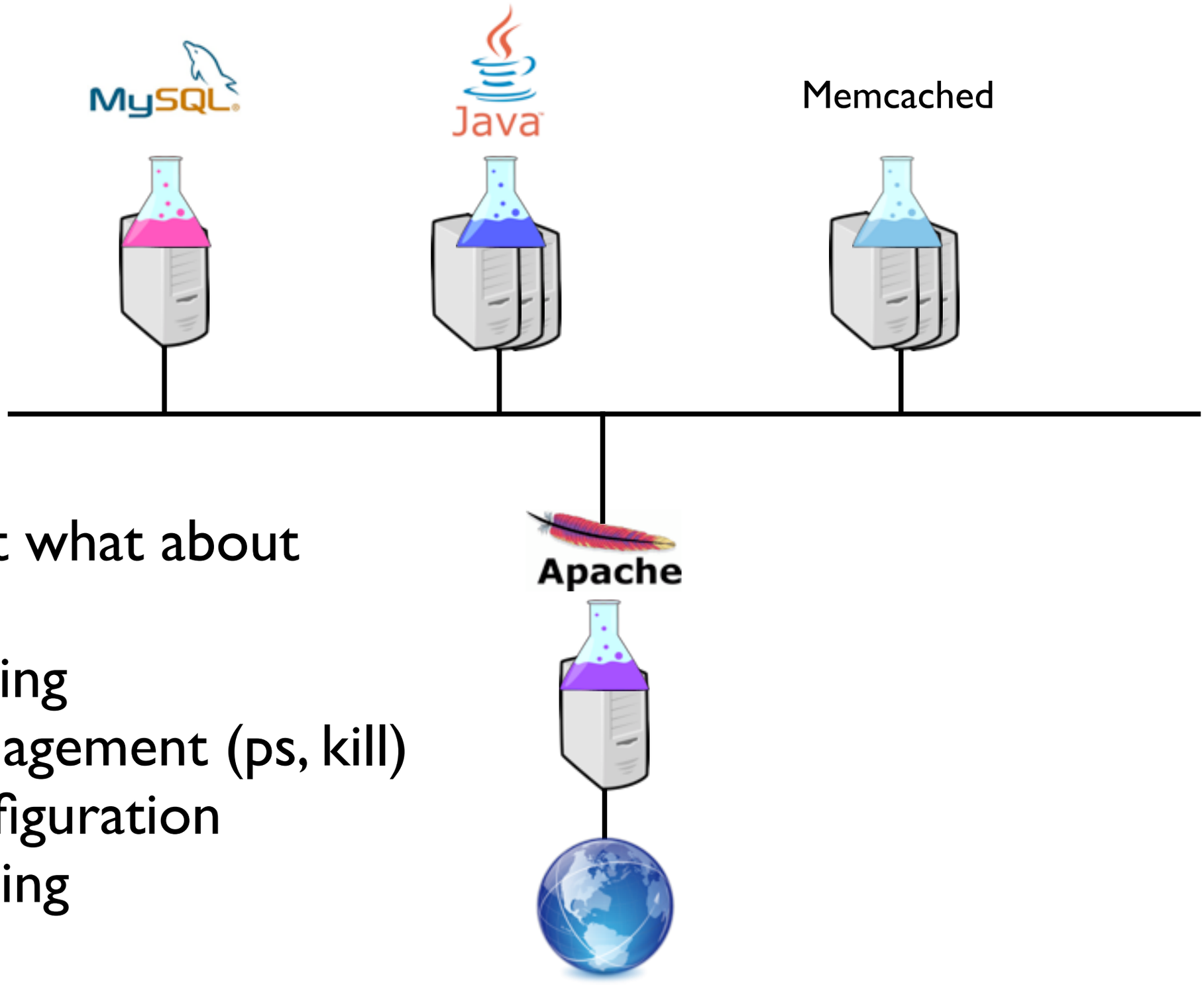
Typical Cloud Web Application



Typical Cloud Web Application



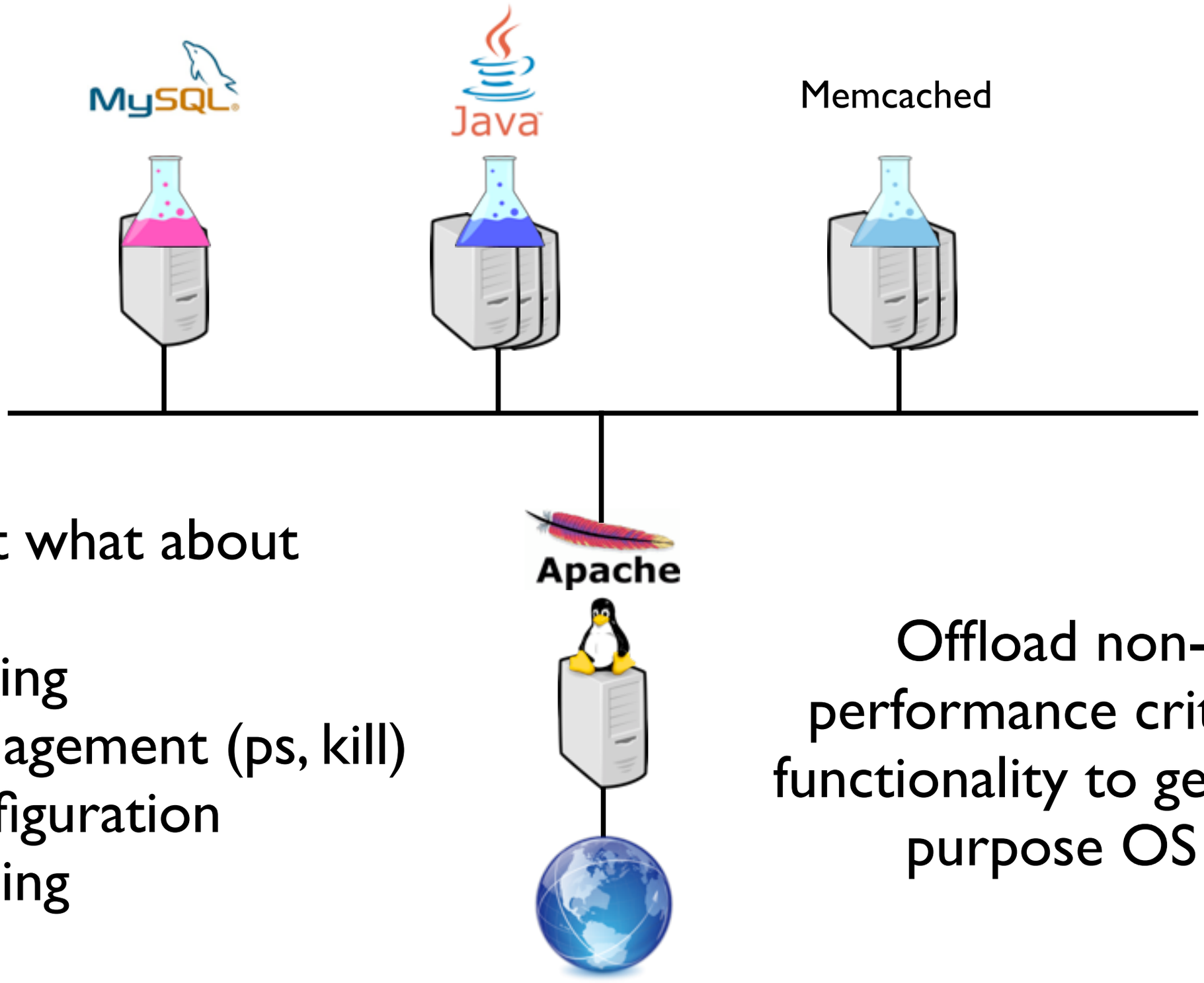
Typical Cloud Web Application



But what about

- logging
- management (ps, kill)
- configuration
- tooling

Typical Cloud Web Application

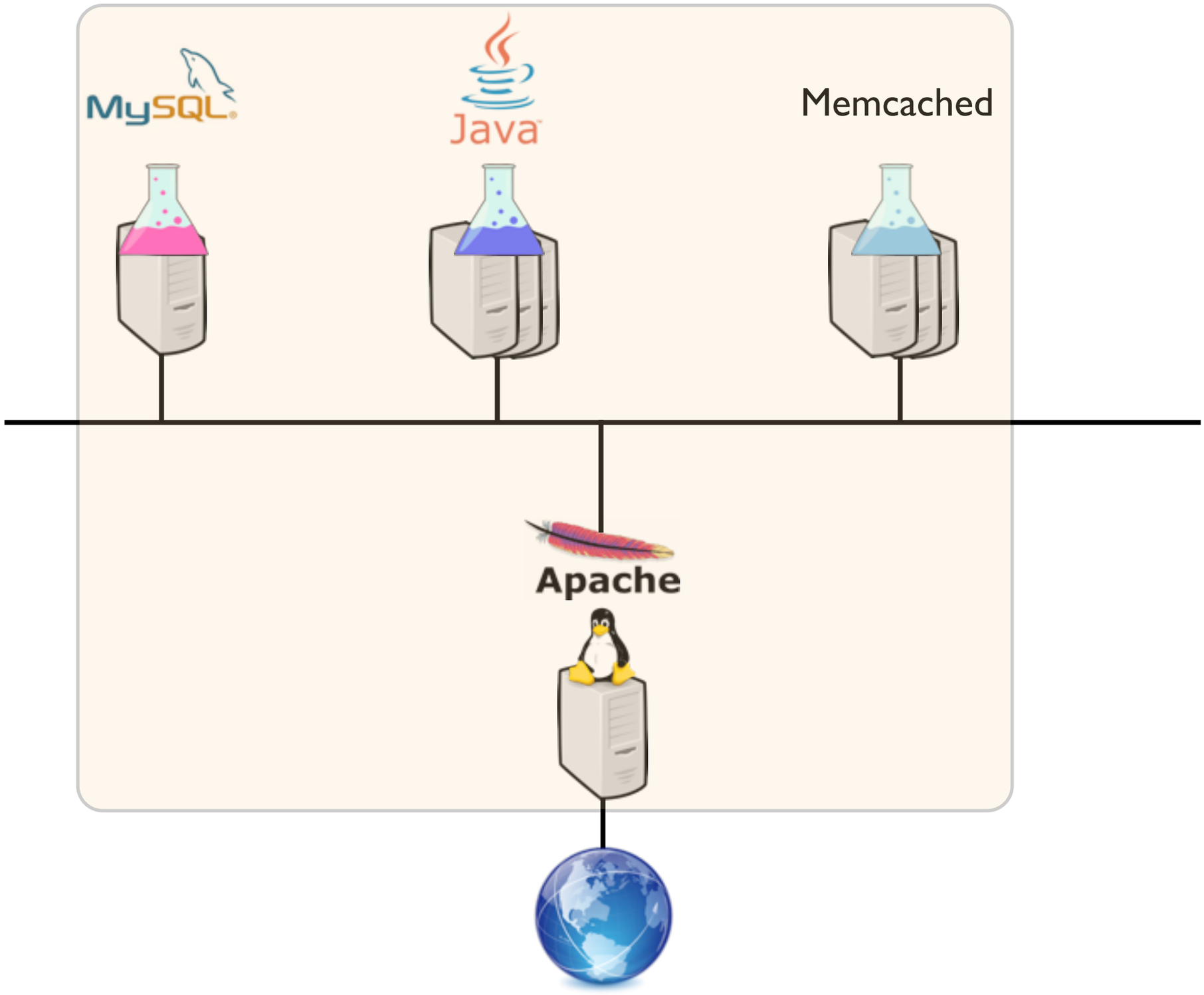


But what about

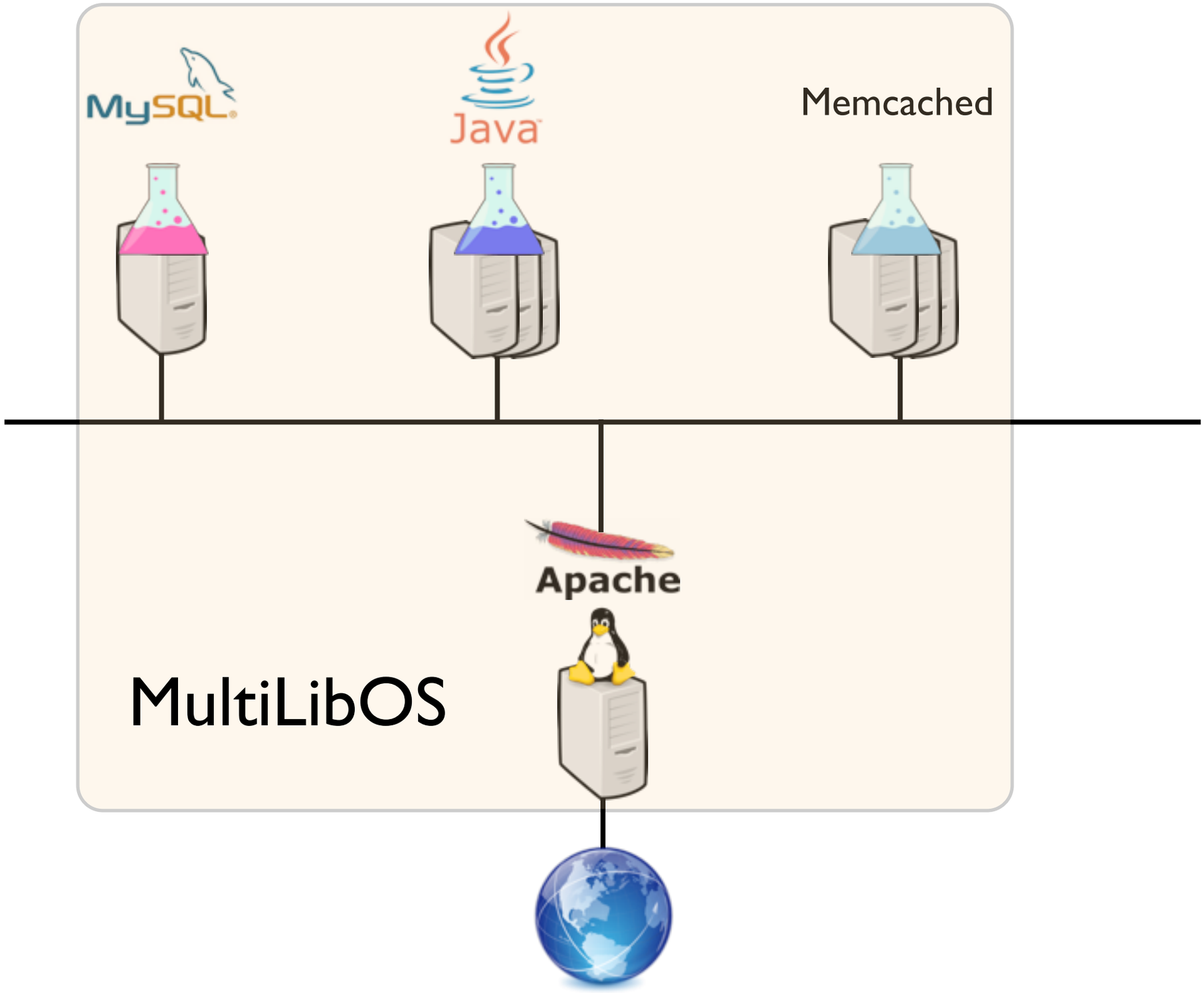
- logging
- management (ps, kill)
- configuration
- tooling

Offload non-performance critical functionality to general purpose OS

Typical Cloud Web Application



Typical Cloud Web Application



A MultiLibOS is a tightly integrated composition of general purpose operating systems and specialized library operating systems

There are many different ways a MultiLibOS might be integrated

There are many different ways a MultiLibOS might be integrated

- Distributed Shared Memory
- Message Passing
- Distributed Namespace (9p)

The MultiLibOS allows a tradeoff between effort and utility

Higher Effort

Lower Utility

Higher Utility

Lower Effort

Higher Effort

Lower Utility

Higher Utility

Existing General Purpose Systems

Lower Effort

Higher Effort

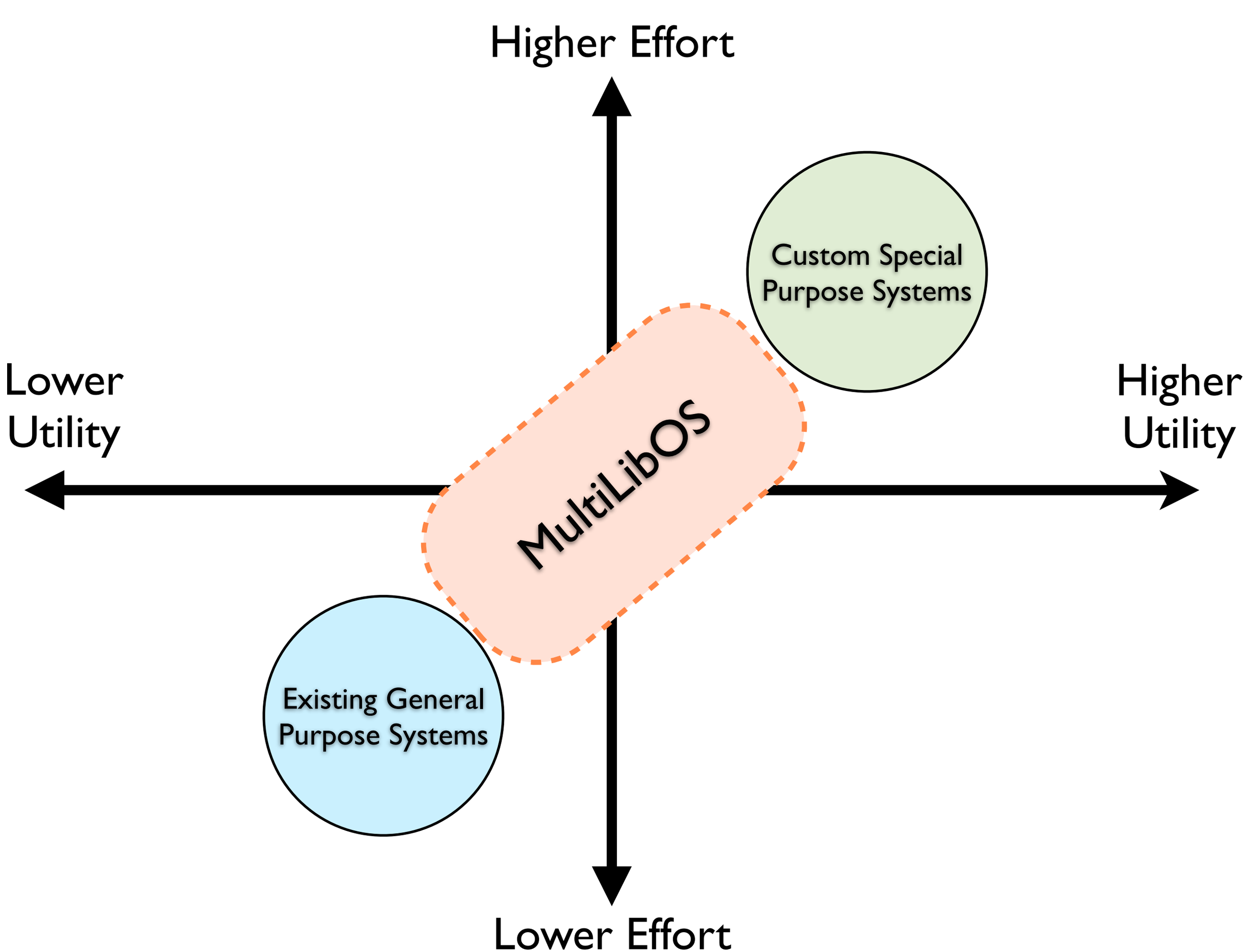
Custom Special Purpose Systems

Lower Utility

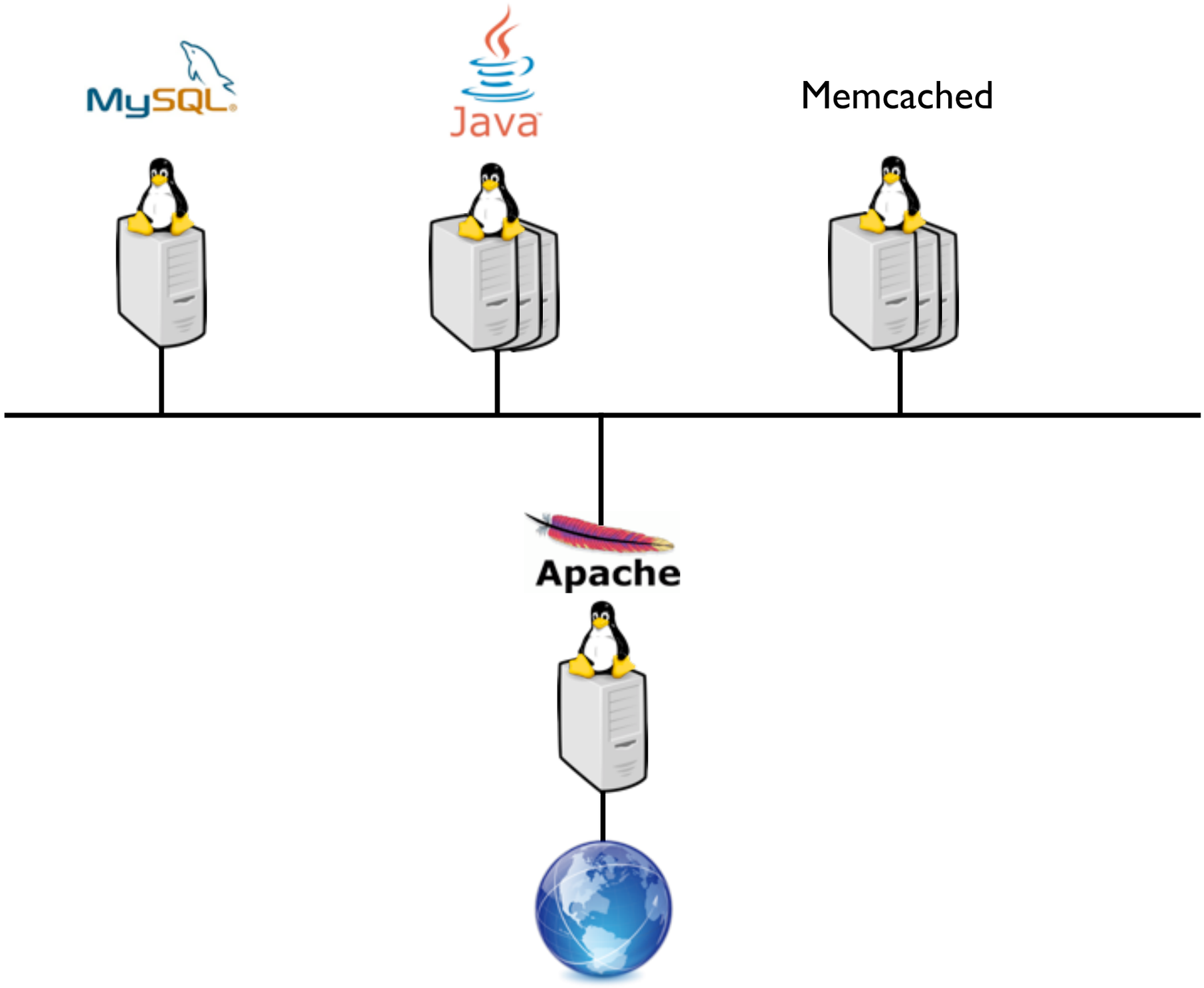
Higher Utility

Existing General Purpose Systems

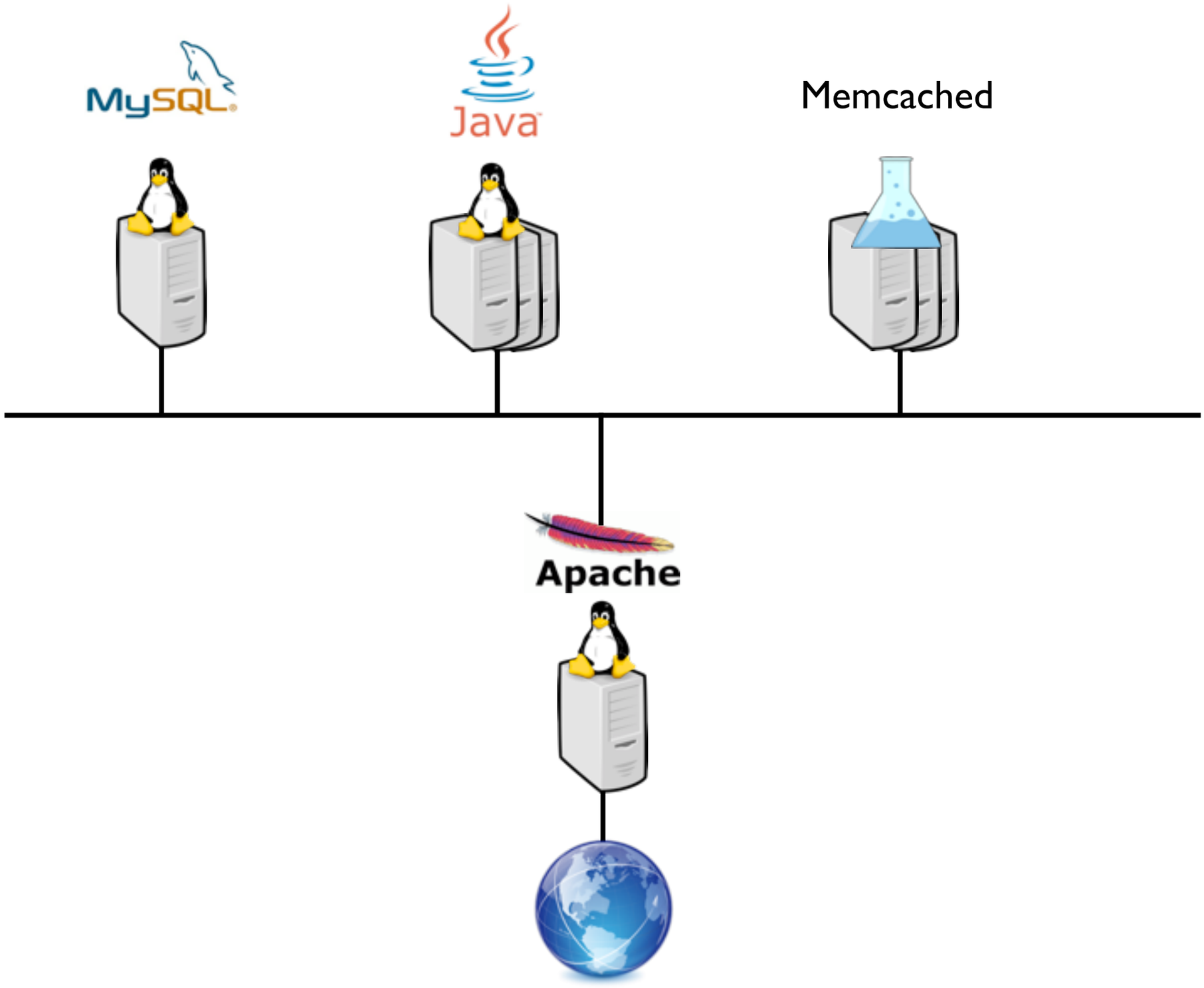
Lower Effort


Higher Effort
Custom Special Purpose Systems
Lower Utility
Higher Utility
MultiLibOS
Existing General Purpose Systems
Lower Effort

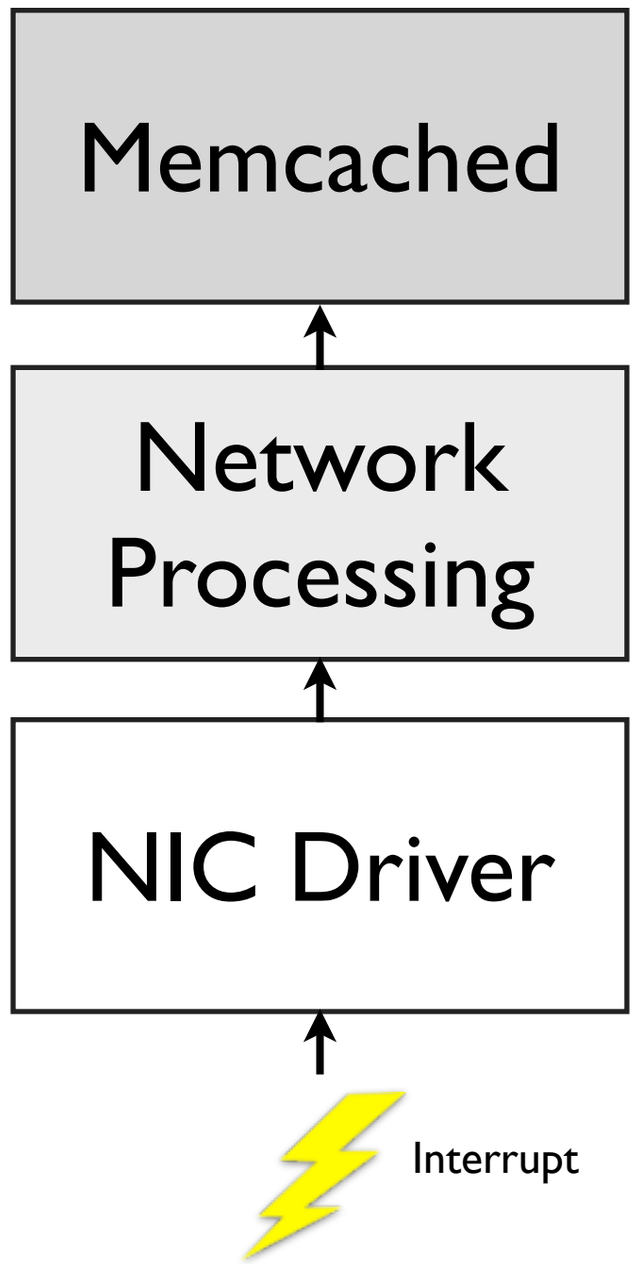
Typical Cloud Web Application



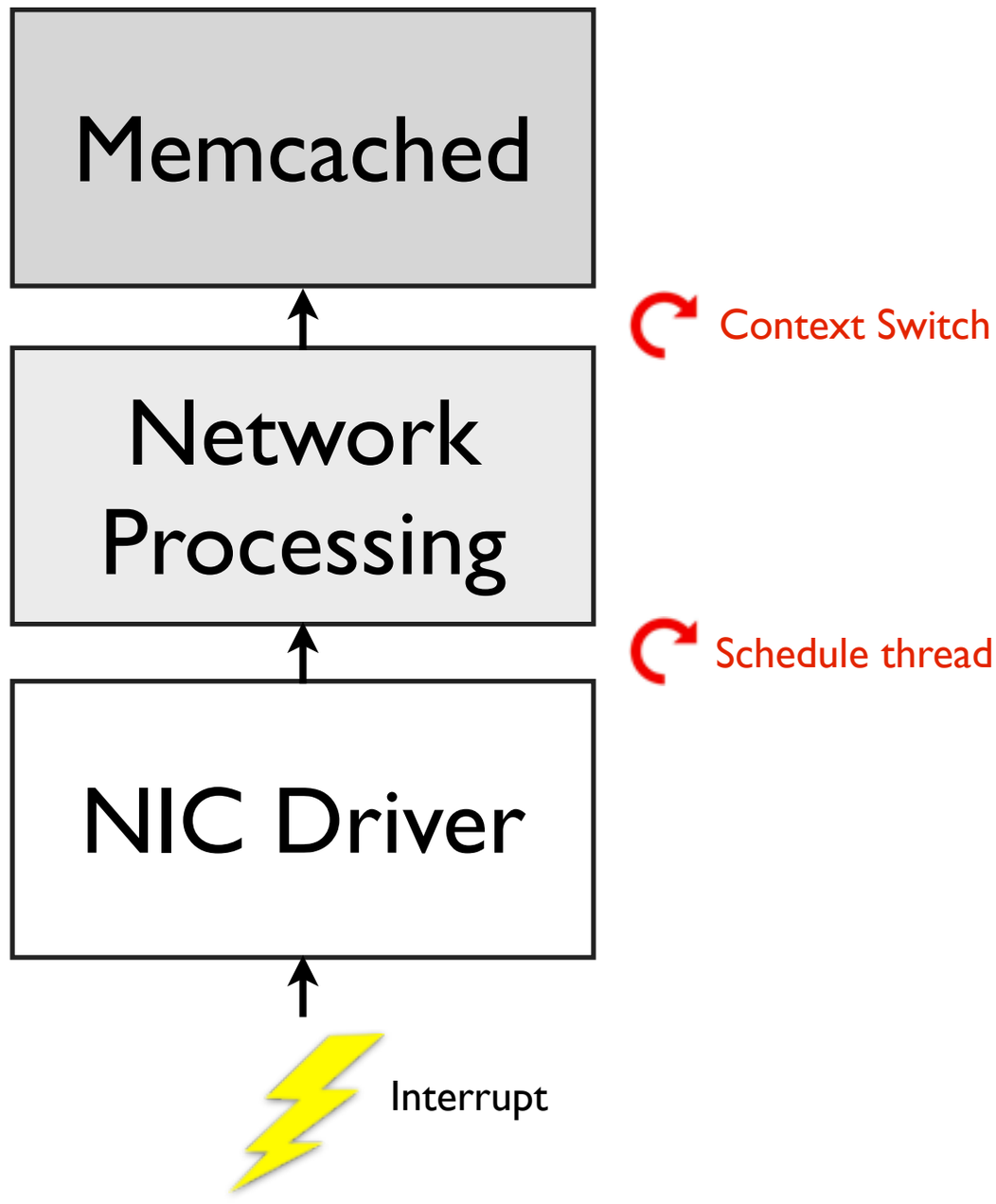
Typical Cloud Web Application



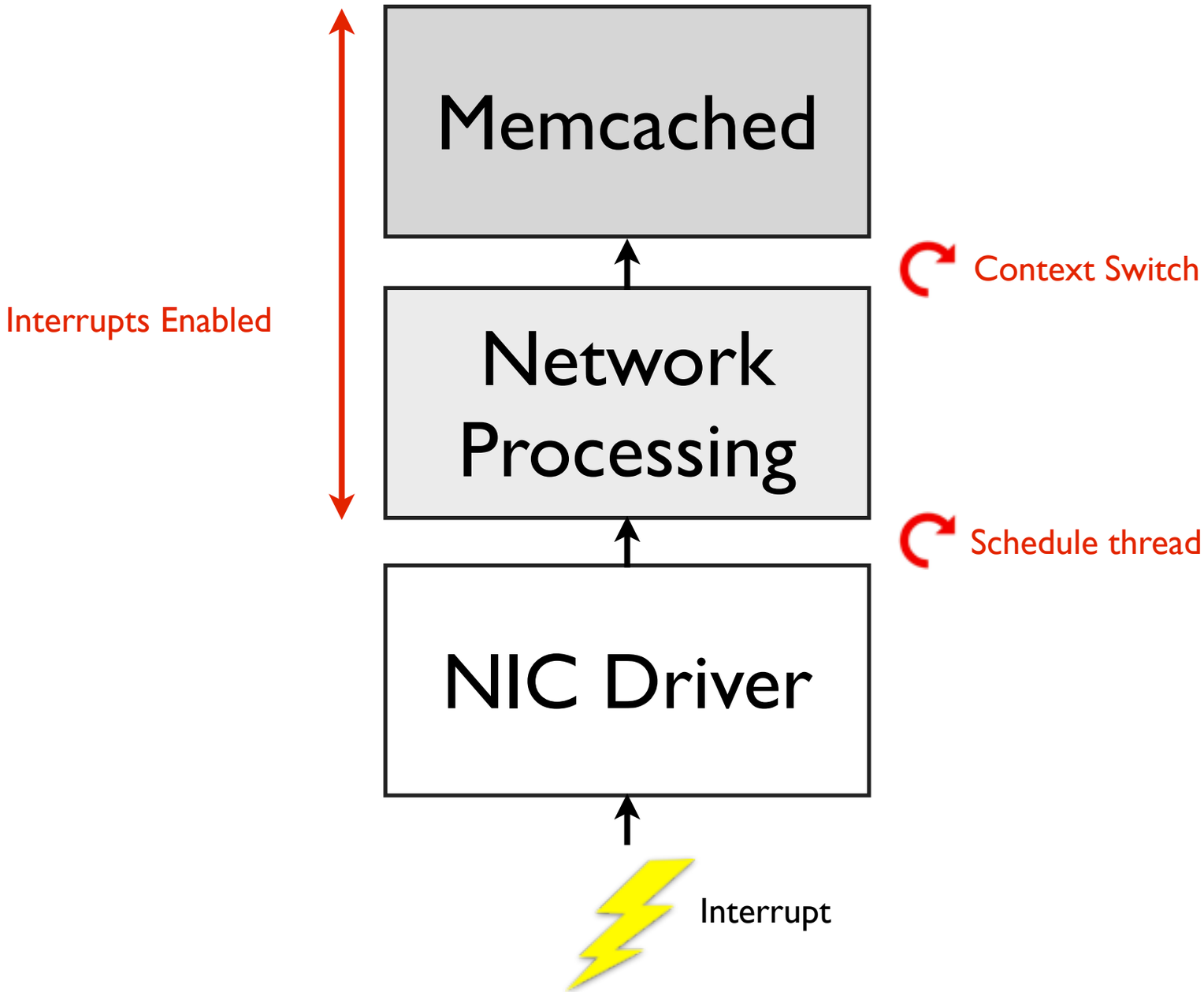
Linux Memcached



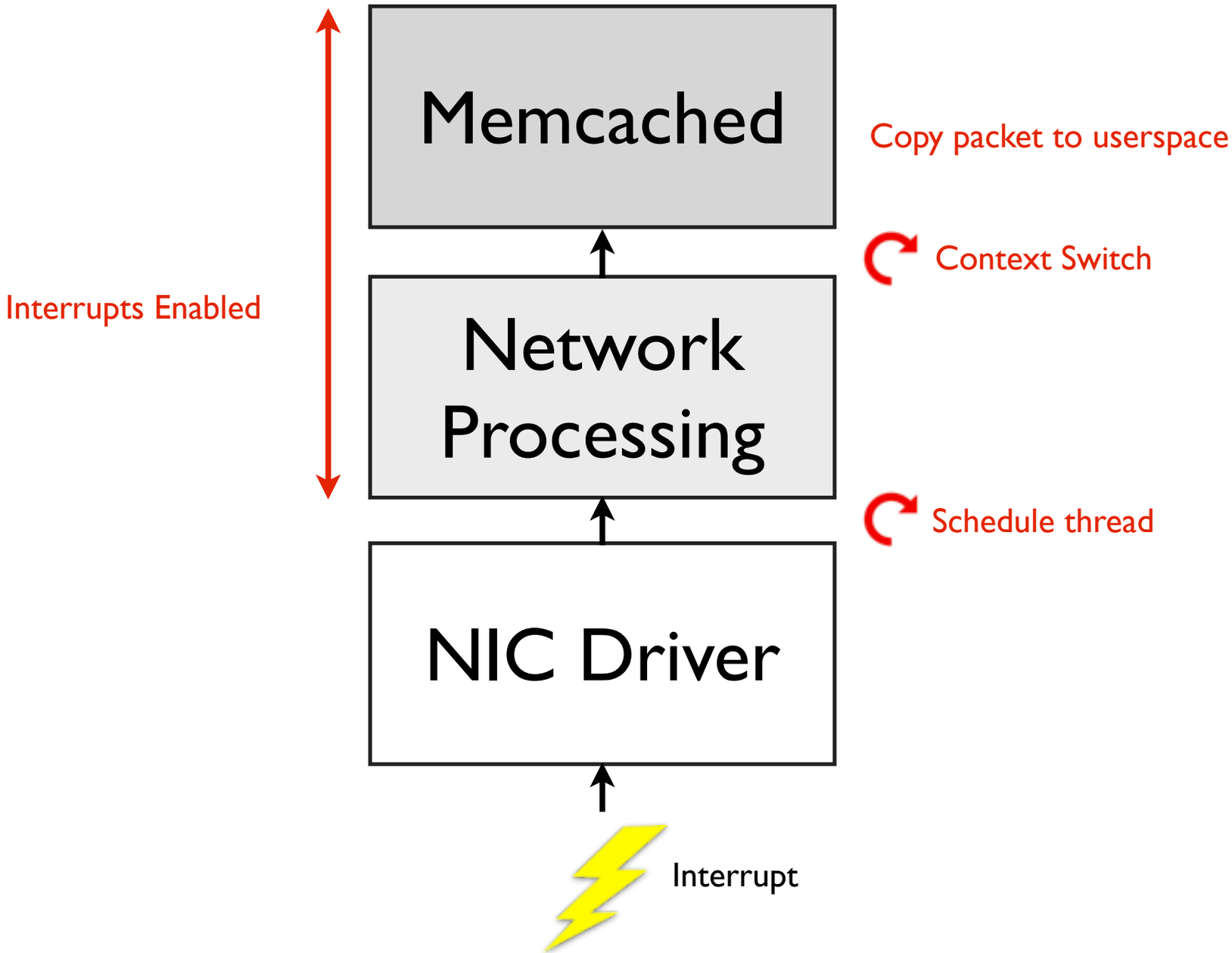
Linux Memcached



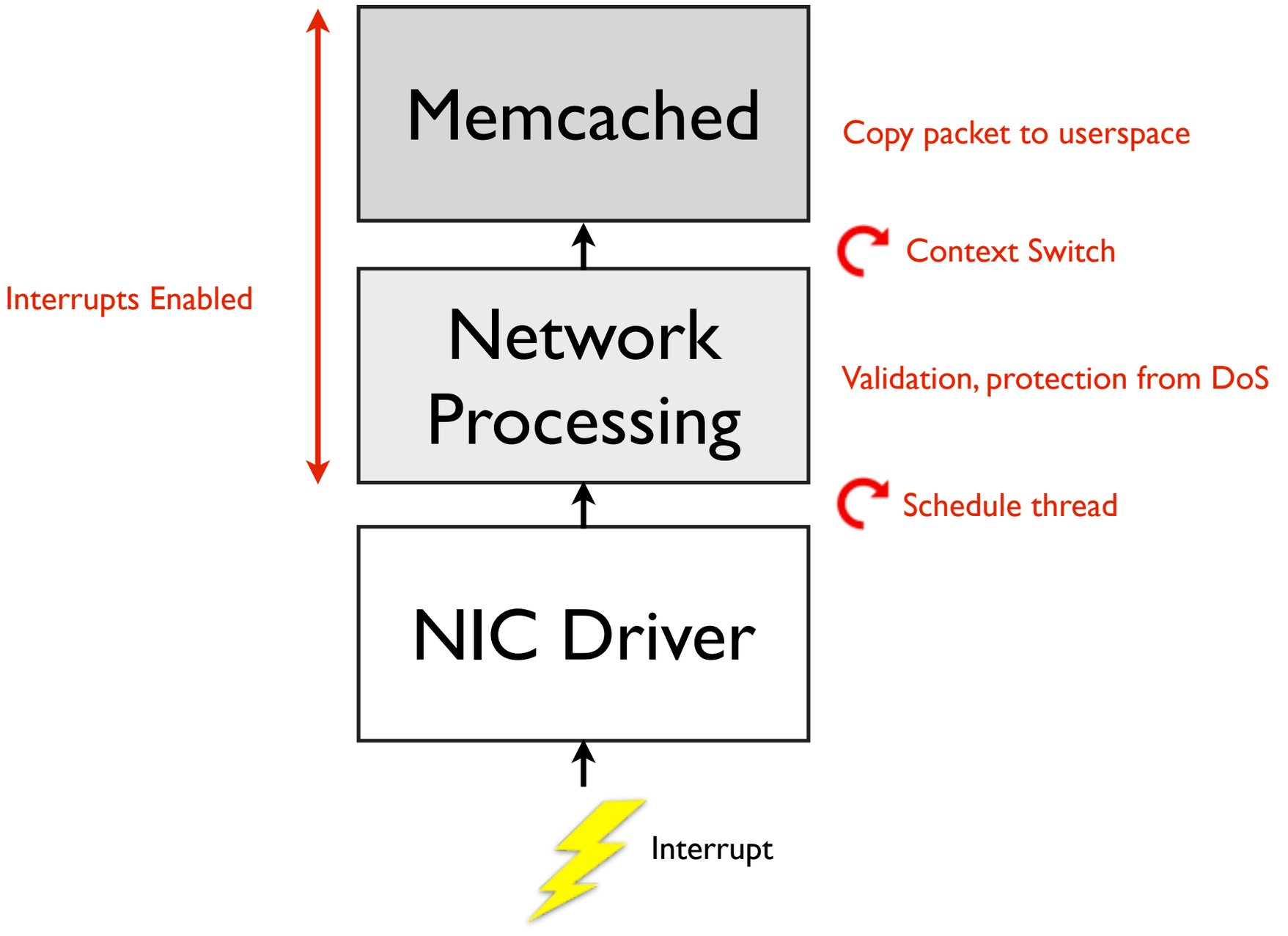
Linux Memcached



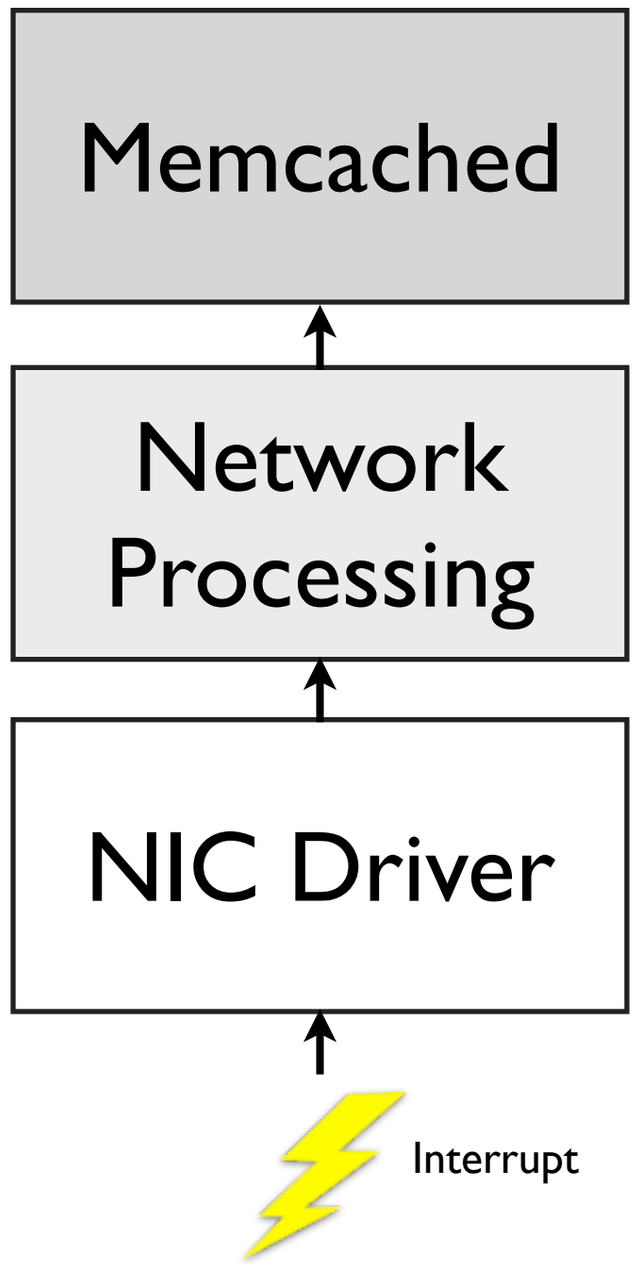
Linux Memcached



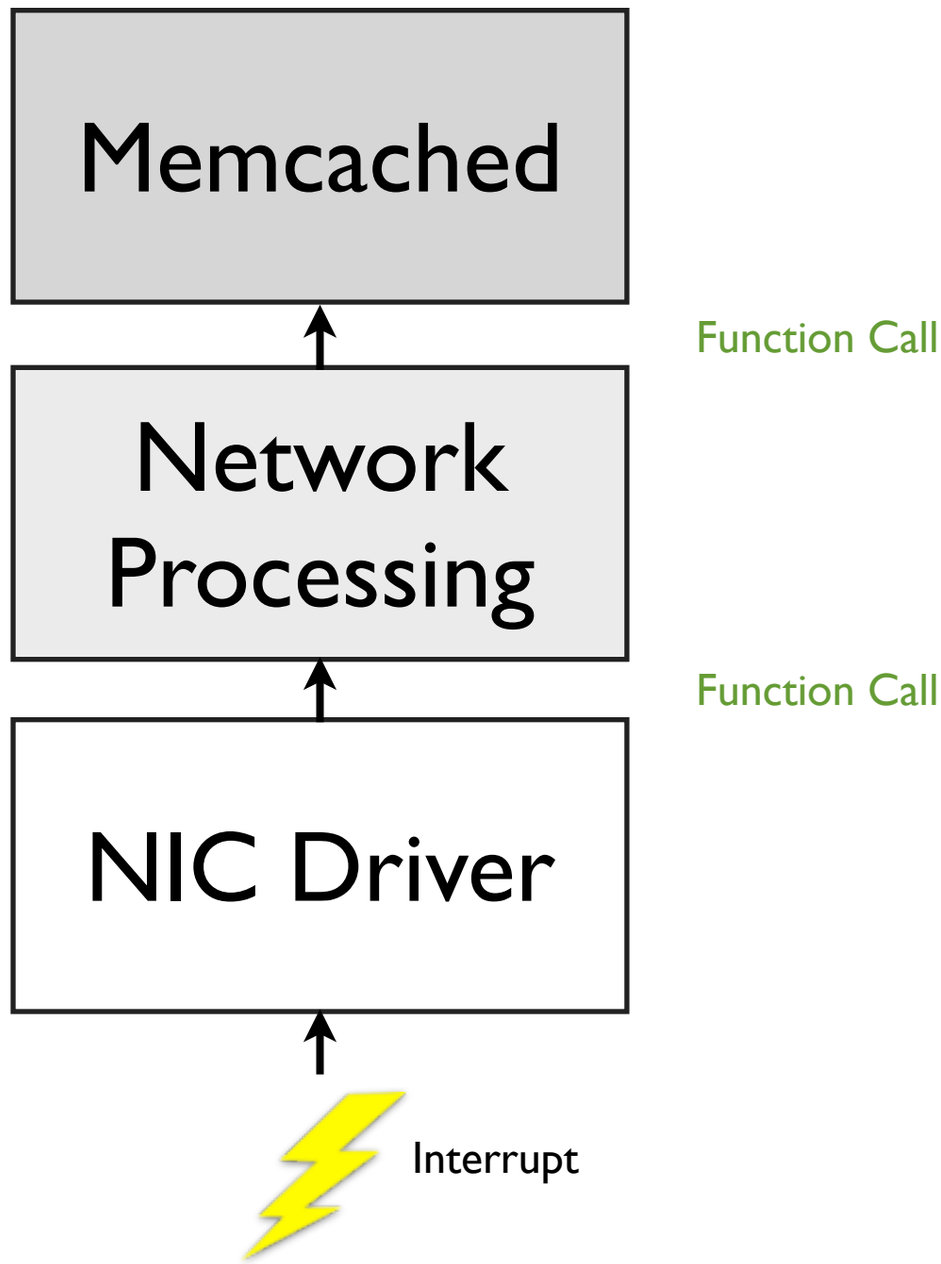
Linux Memcached



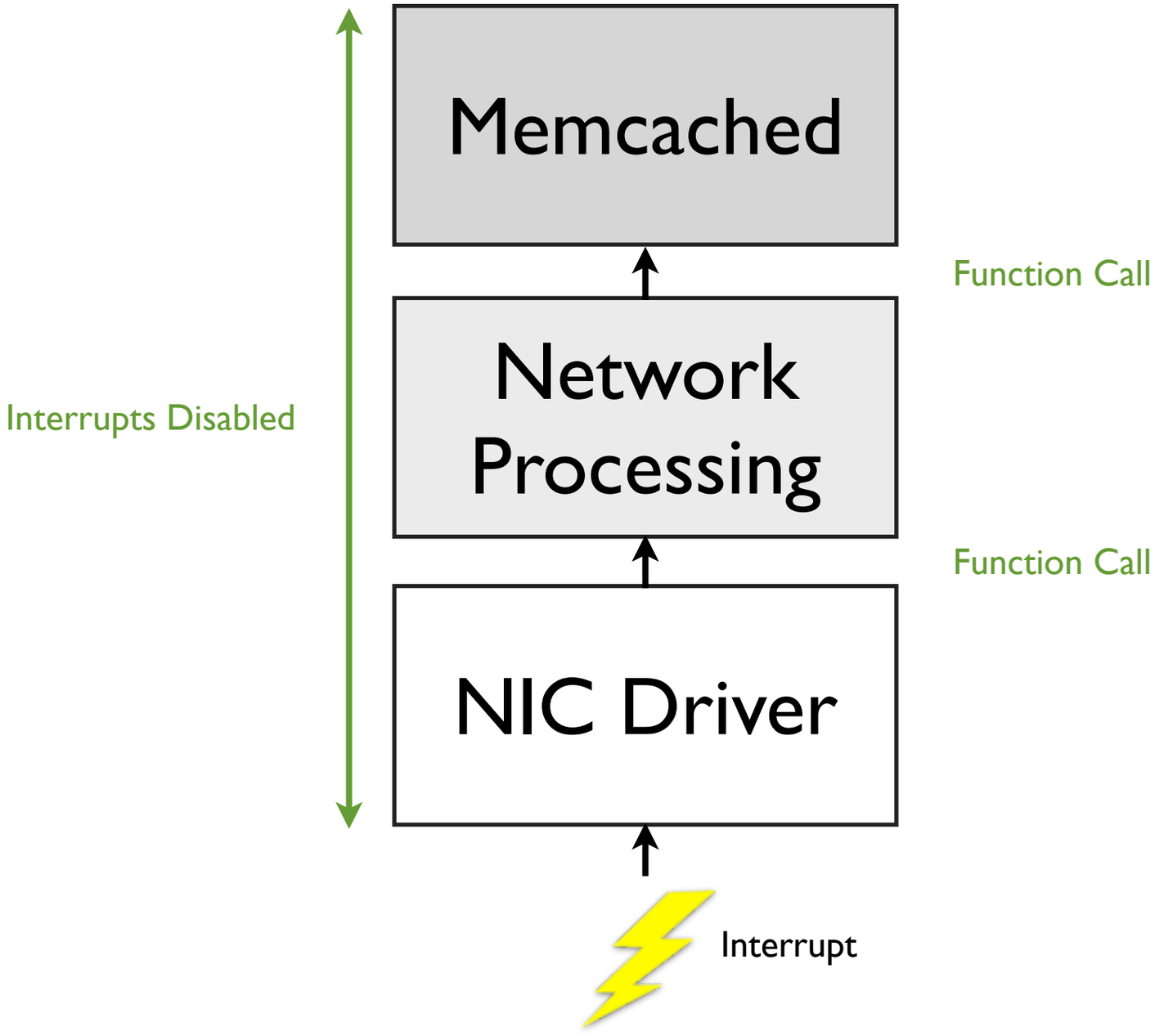
Example LibraryOS Memcached



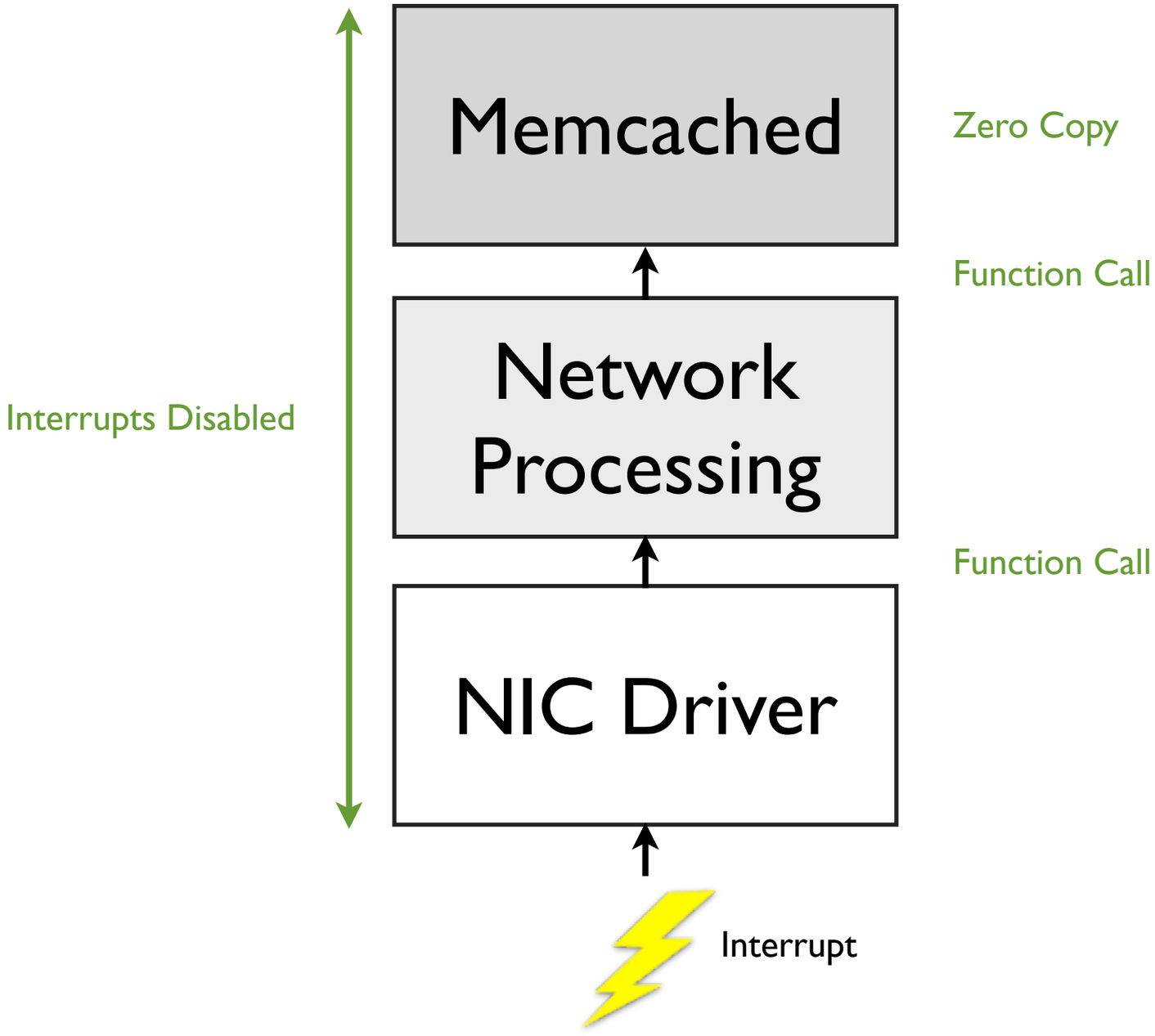
Example LibraryOS Memcached



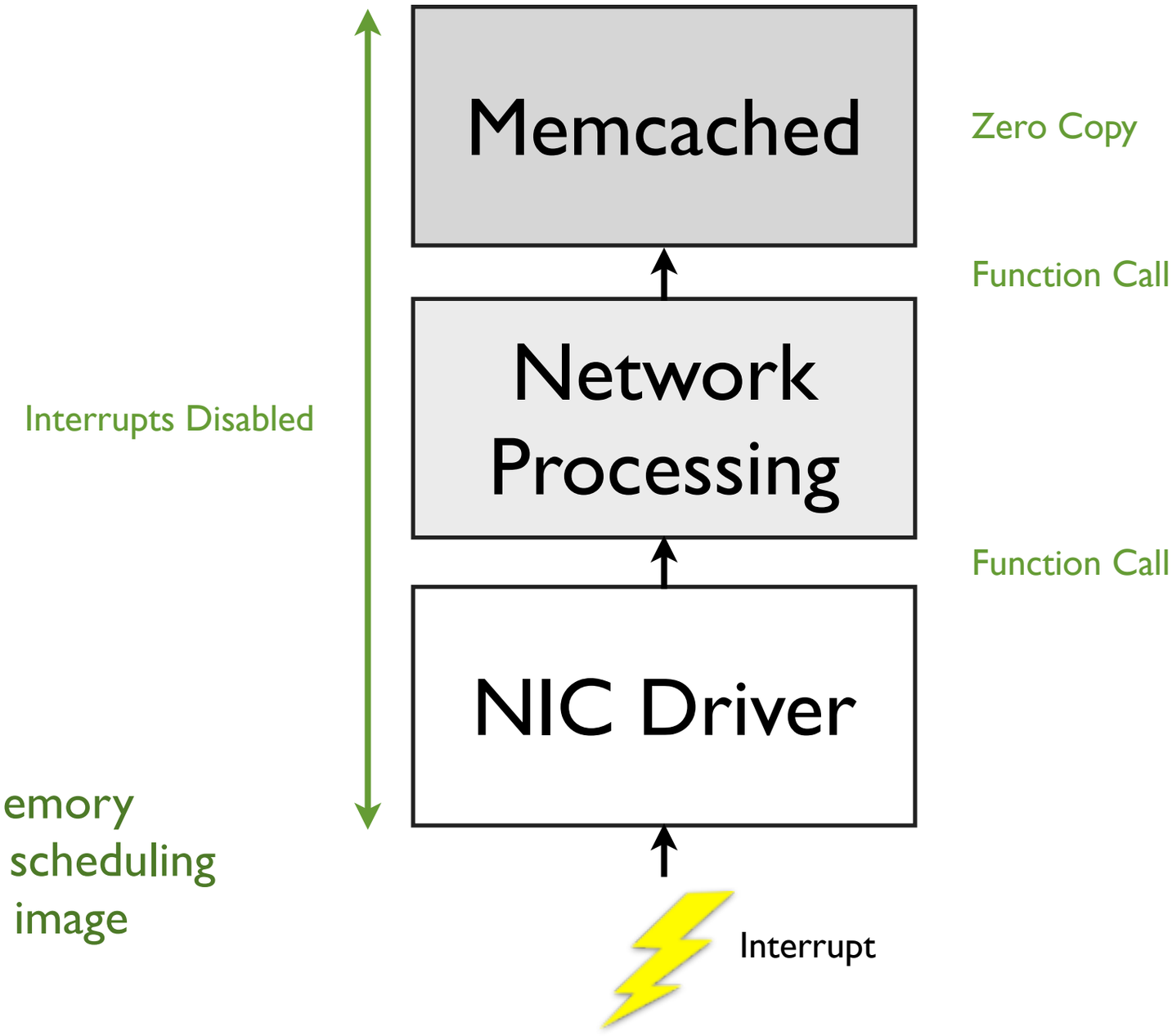
Example LibraryOS Memcached



Example LibraryOS Memcached



Example LibraryOS Memcached



- No virtual memory
- No complex scheduling
- Small system image

SageMath

- Open source mathematics environment (like Matlab, Mathematica, etc.)
- Python
- Incorporates many libraries

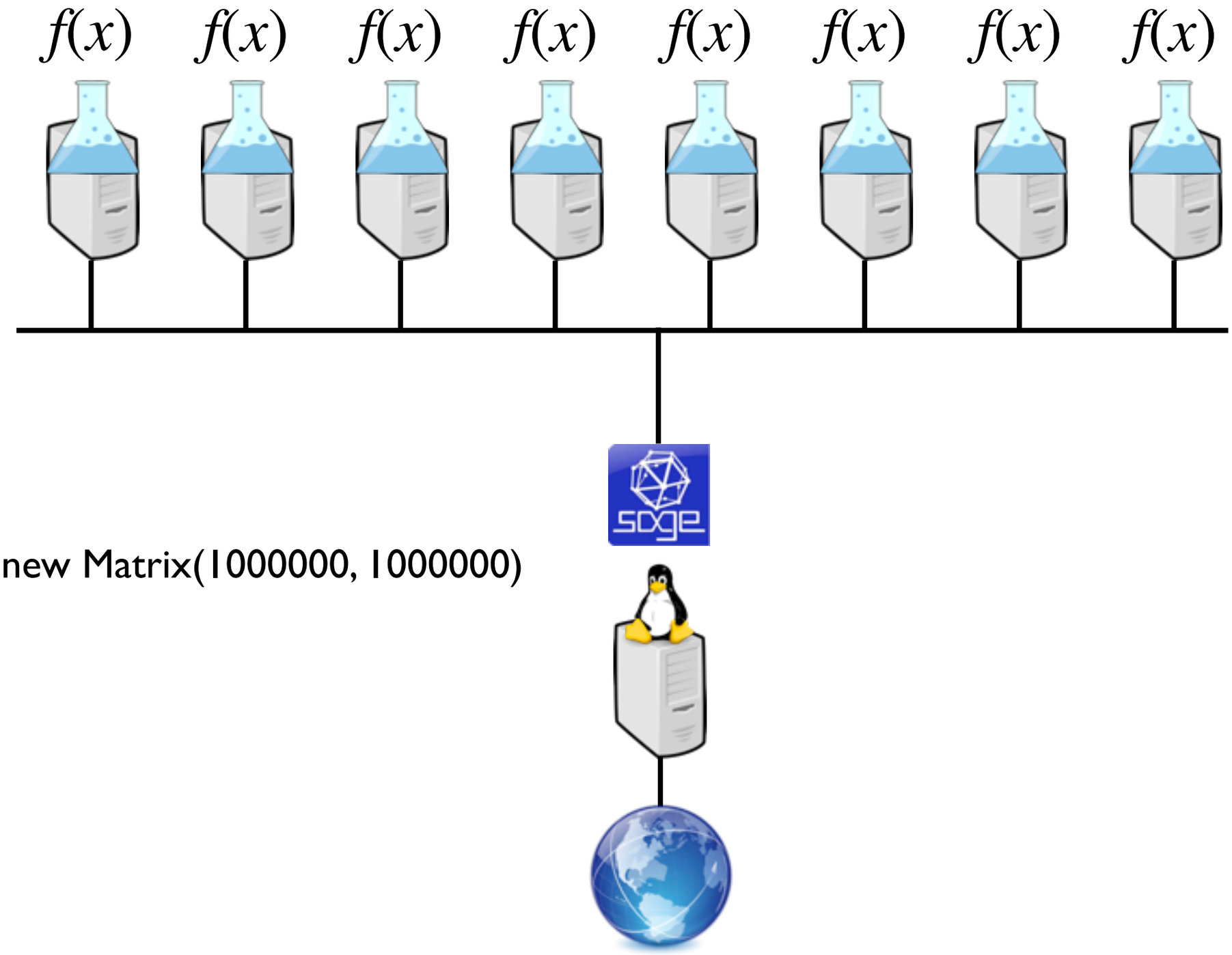
SageMath

- Open source mathematics environment (like Matlab, Mathematica, etc.)
- Python
- Incorporates many libraries

SageMath

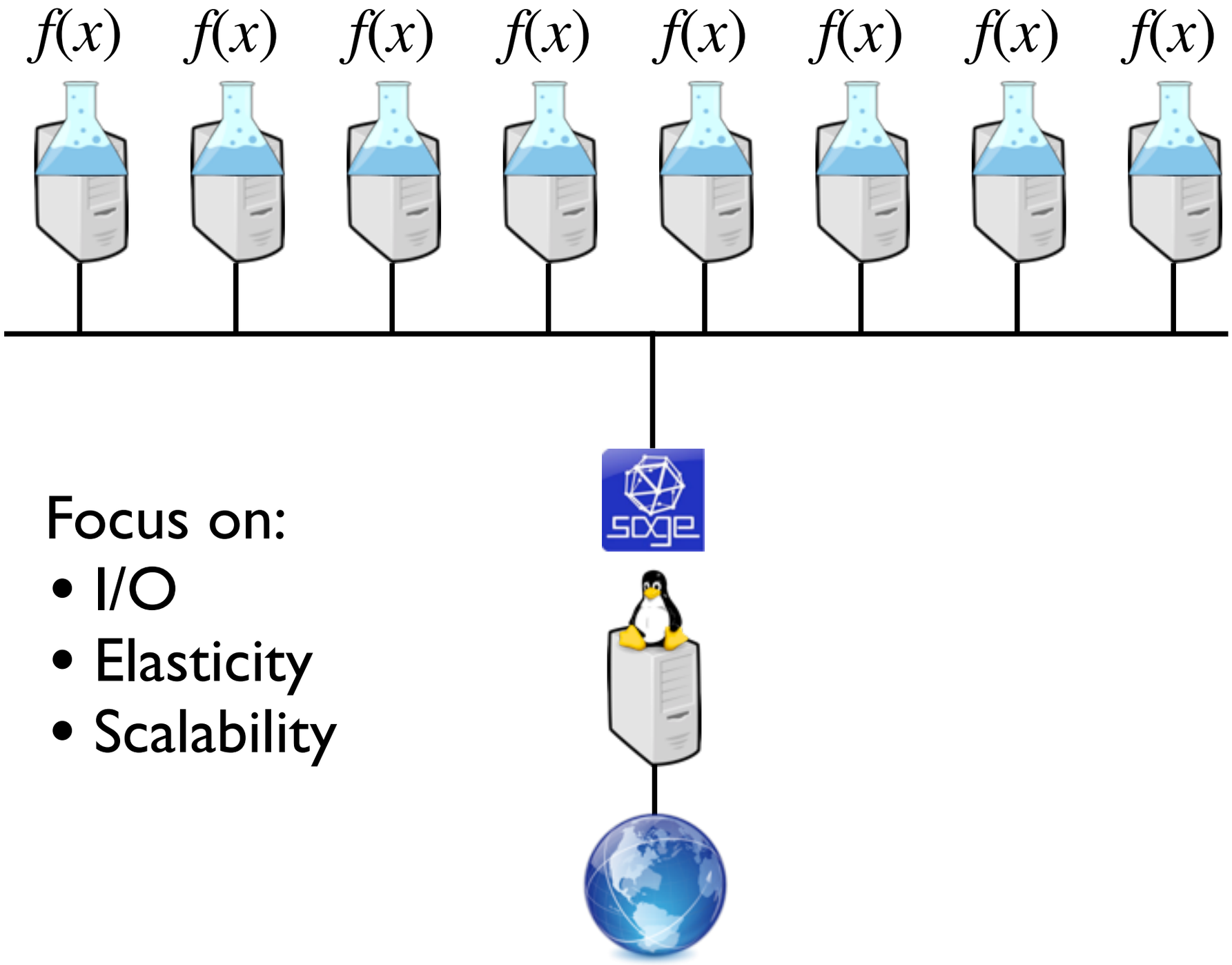
new Matrix(1000000, 1000000)

SageMath



new Matrix(1000000, 1000000)

SageMath



Focus on:

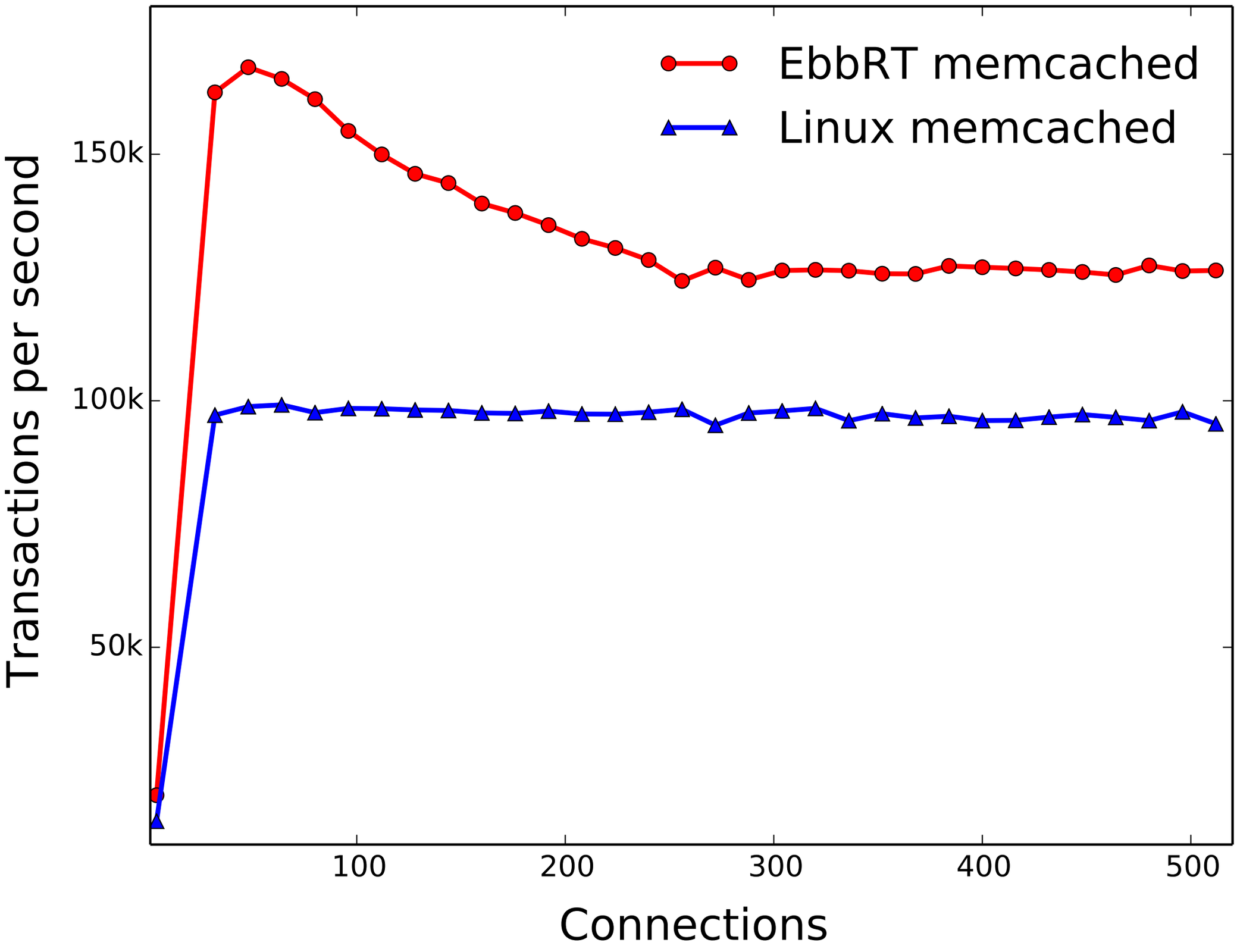
- I/O
- Elasticity
- Scalability

MultiLibOS Challenges

- Reuse vs. Specialization
- “Versionitis”
- Avoid building a new OS

Elastic Building Block Runtime (EbbRT)

- A MultiLibOS "toolkit" for elastic applications
- Components are
  - Efficient
  - Reusable
  - Customizable


EbbRT memcached
Linux memcached
Transactions per second
150k
100k
50k
100
200
300
400
500
Connections

Questions?

https://github.com/sesa/ebbrt

Dan Schatzberg, James Cadden, Orran Krieger, Jonathan Appavoo

dschatz@bu.edu jmcadden@bu.edu okrieg@bu.edu jappavoo@bu.edu