Cornell University
Computer Systems Laboratory

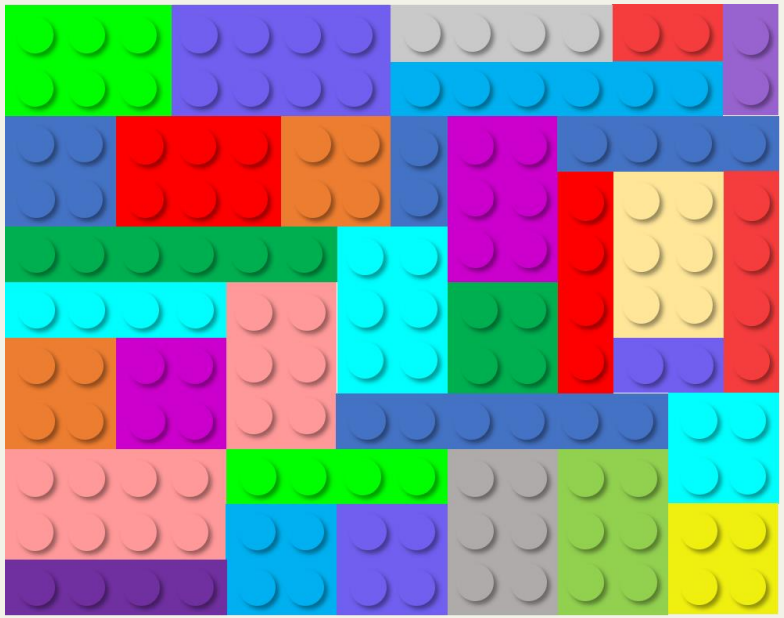
CSL

μQSim: Enabling Accurate and Scalable Simulation for Interactive Microservices

Yanqi Zhang, *Yu Gan, Christina Delimitrou*
Cornell University

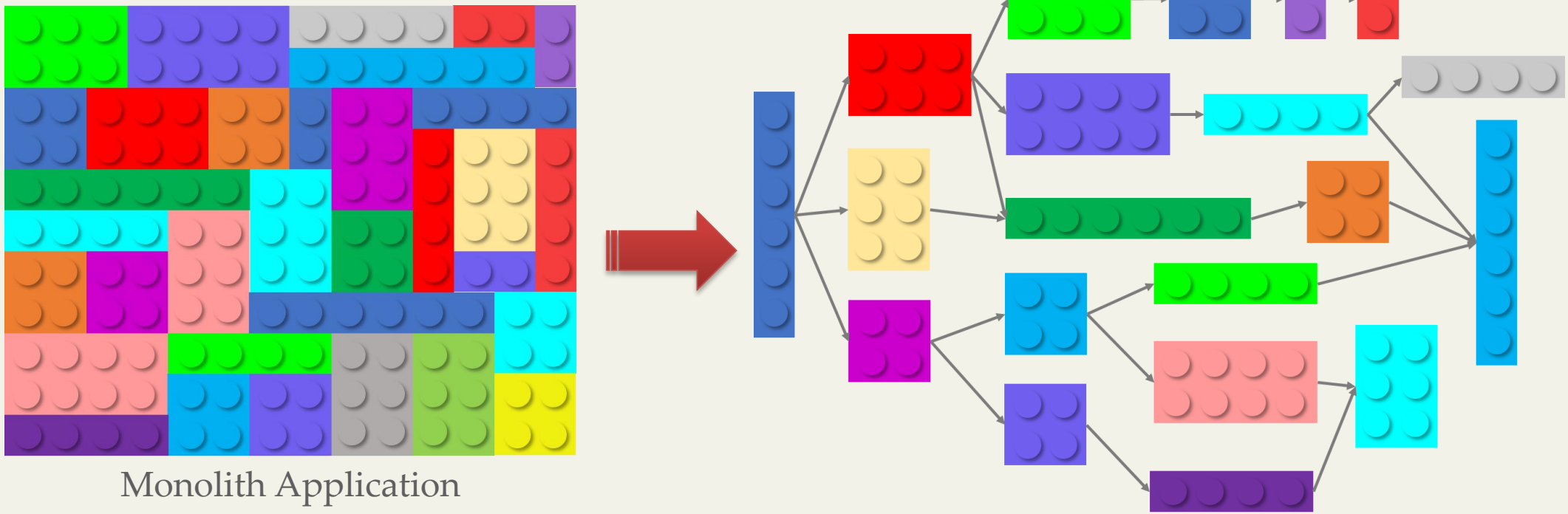
Session: Datacenters and Cloud Computing, March 26th

Marching to Microservices



Monolith Application

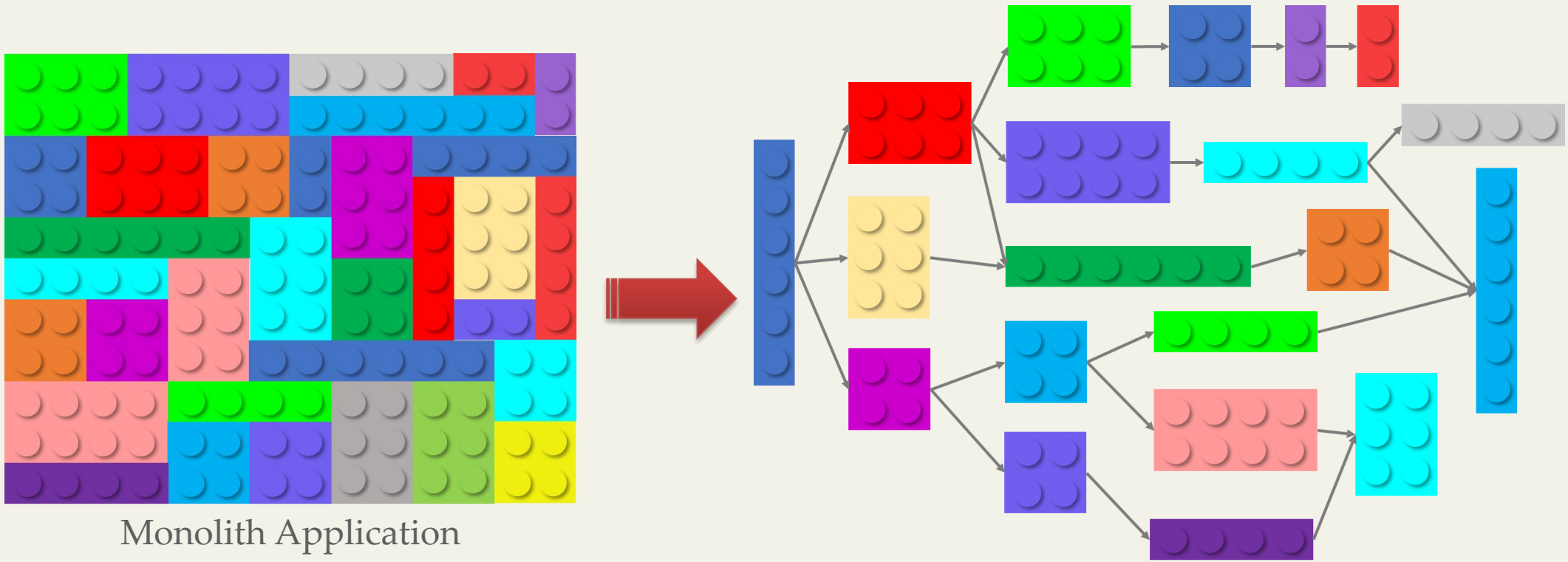
Marching to Microservices



Monolith Application

Microservices

Marching to Microservices

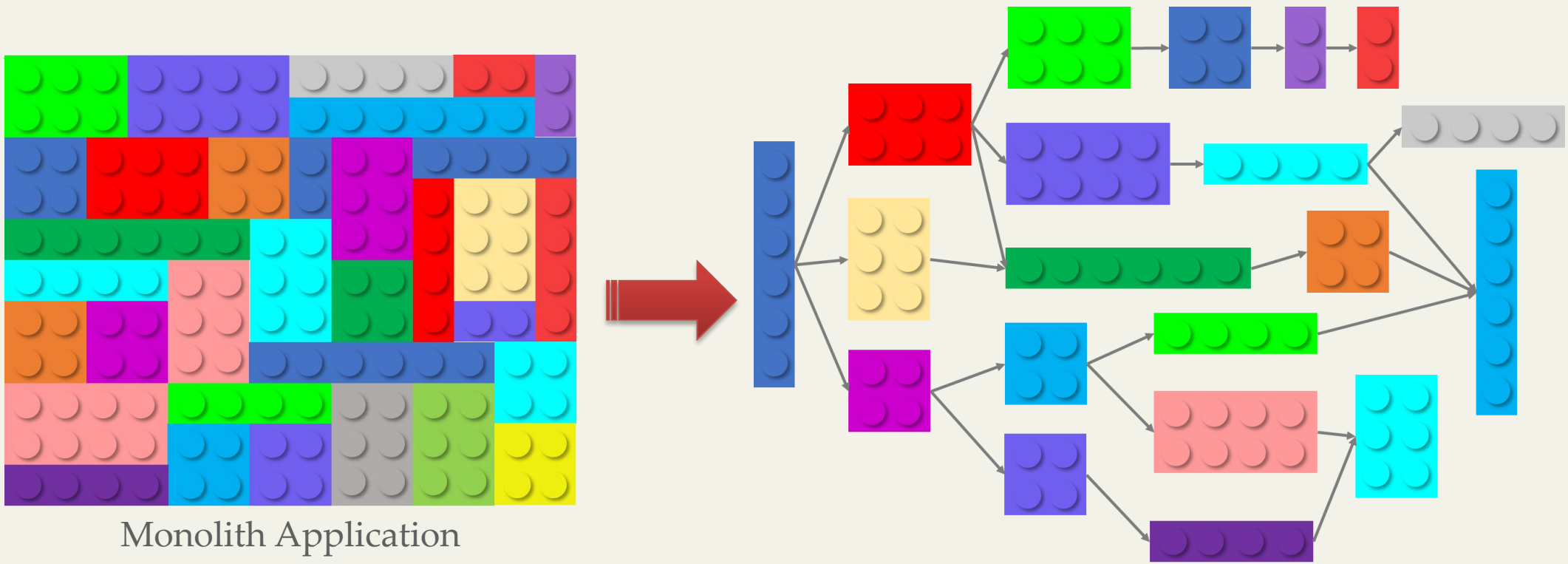


Monolith Application

Microservices

- Easier development & update

Marching to Microservices

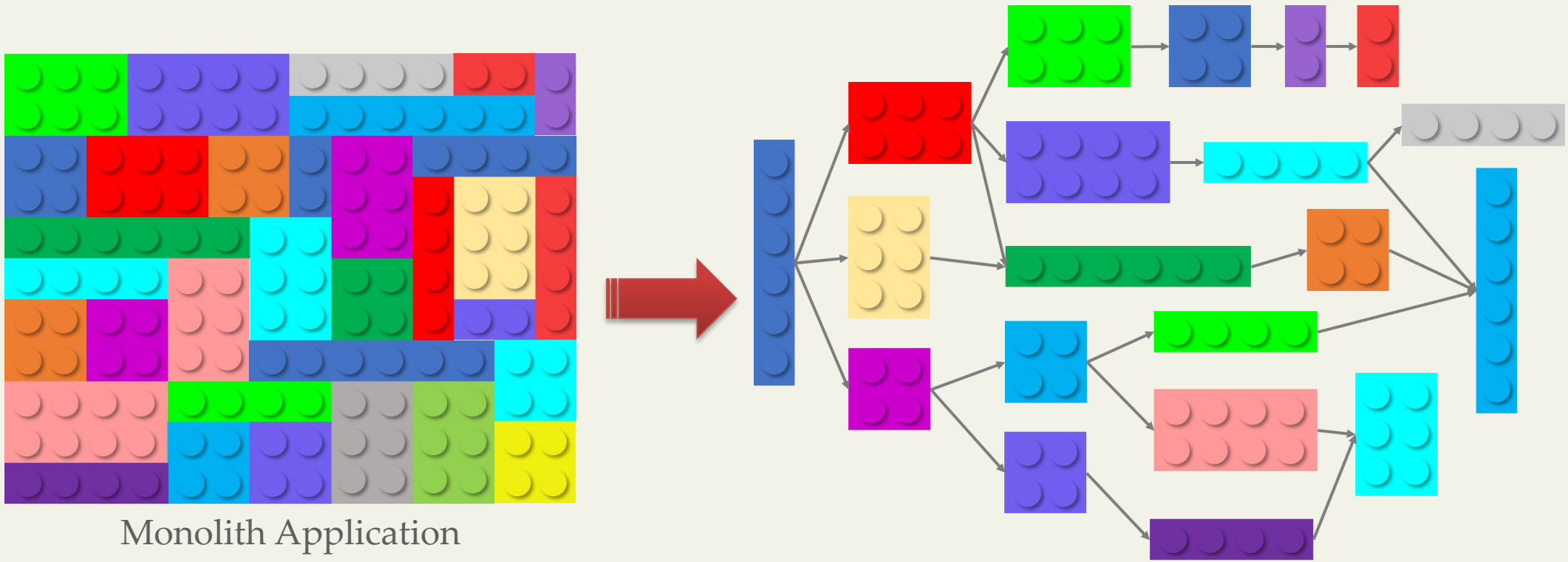


Monolith Application

Microservices

- Easier development & update
- Debugging & error isolation

Marching to Microservices

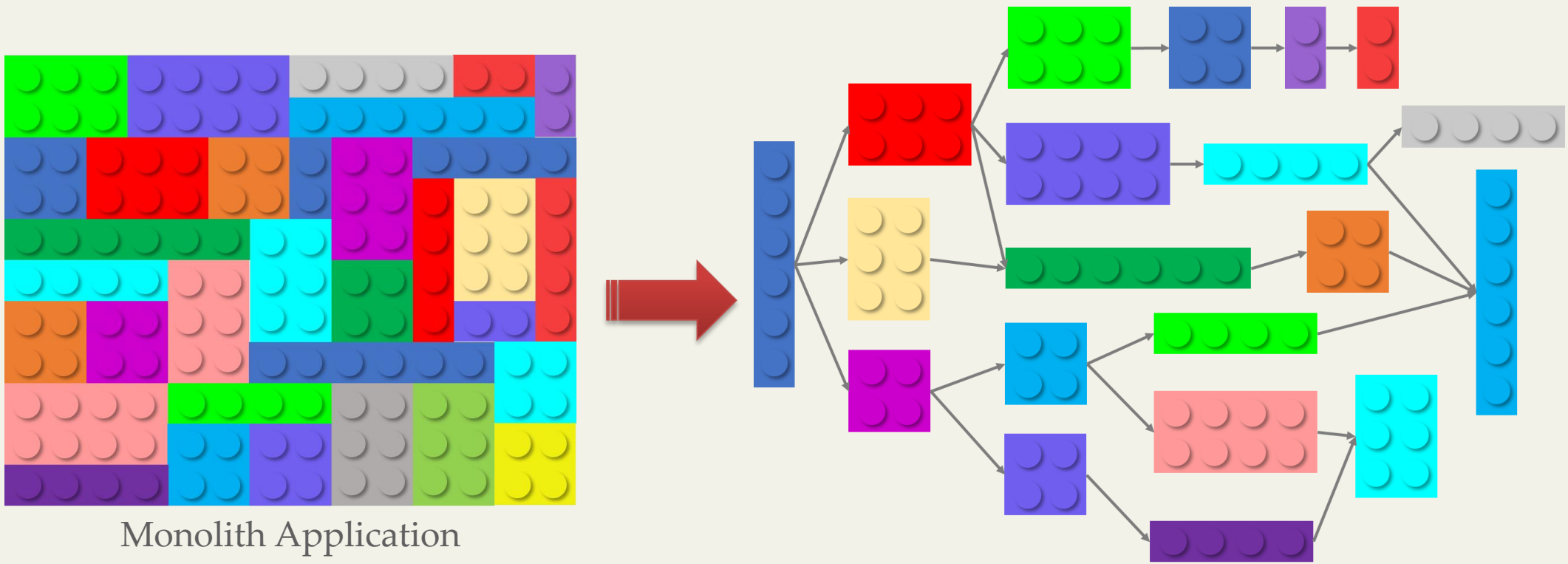


Monolith Application

Microservices

- Easier development & update
- Debugging & error isolation
- Elasticity

Marching to Microservices

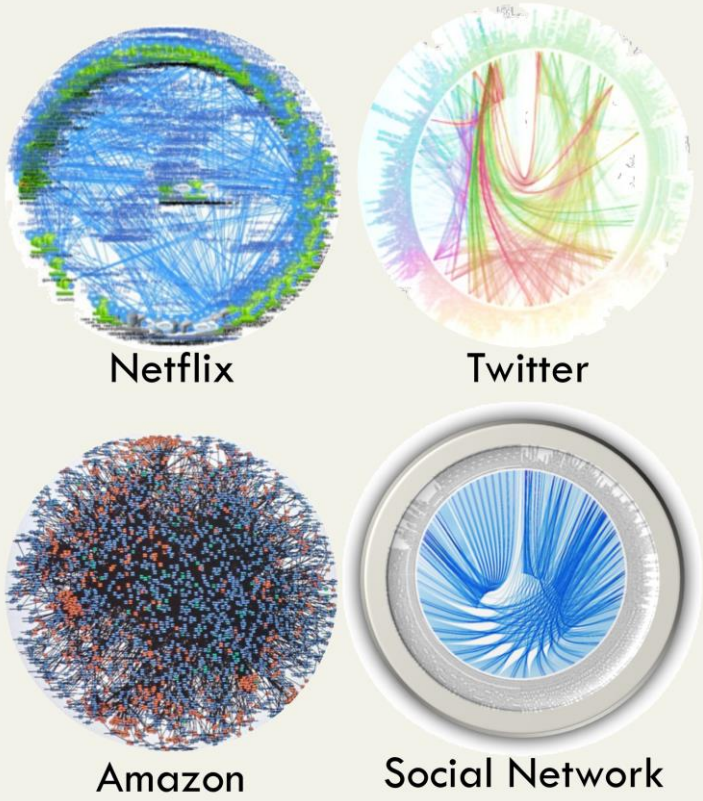


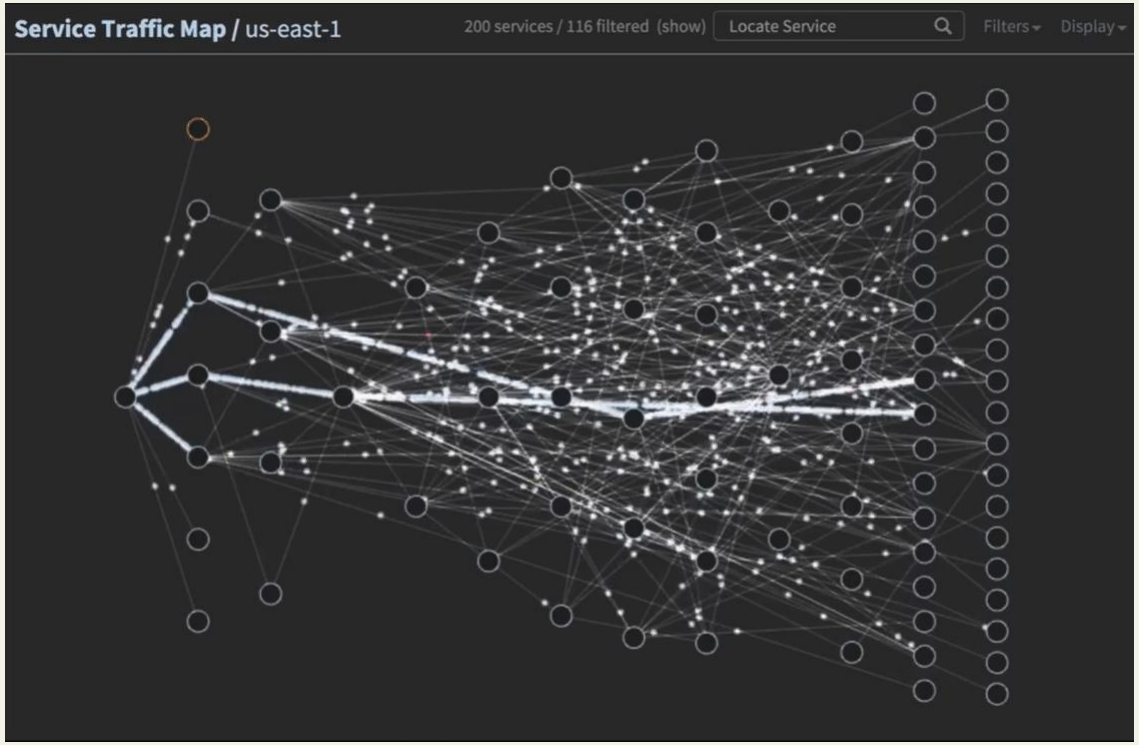
Monolith Application

Microservices

- Easier development & update
- Debugging & error isolation
- Elasticity
- PL/framework heterogeneity

Marching to Microservices

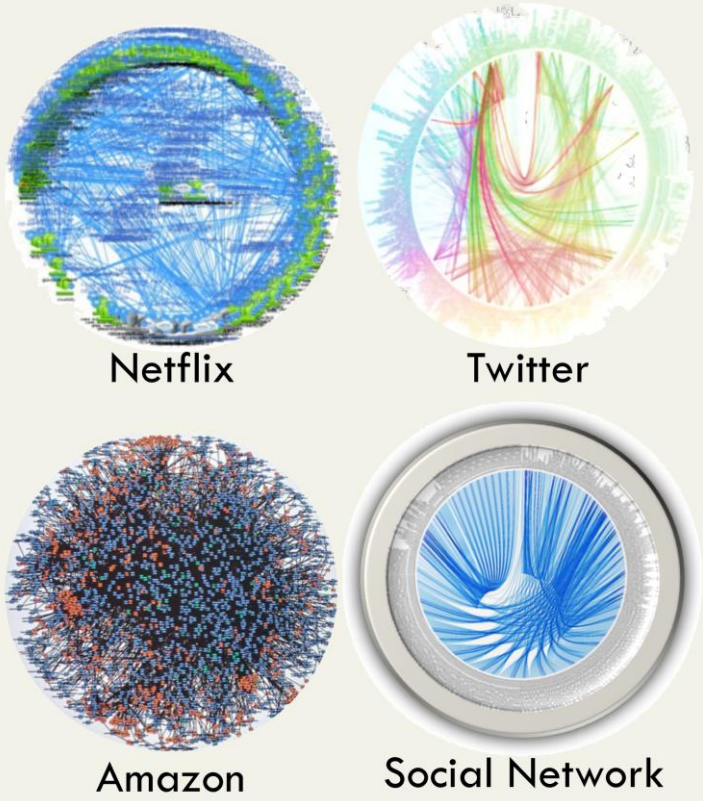


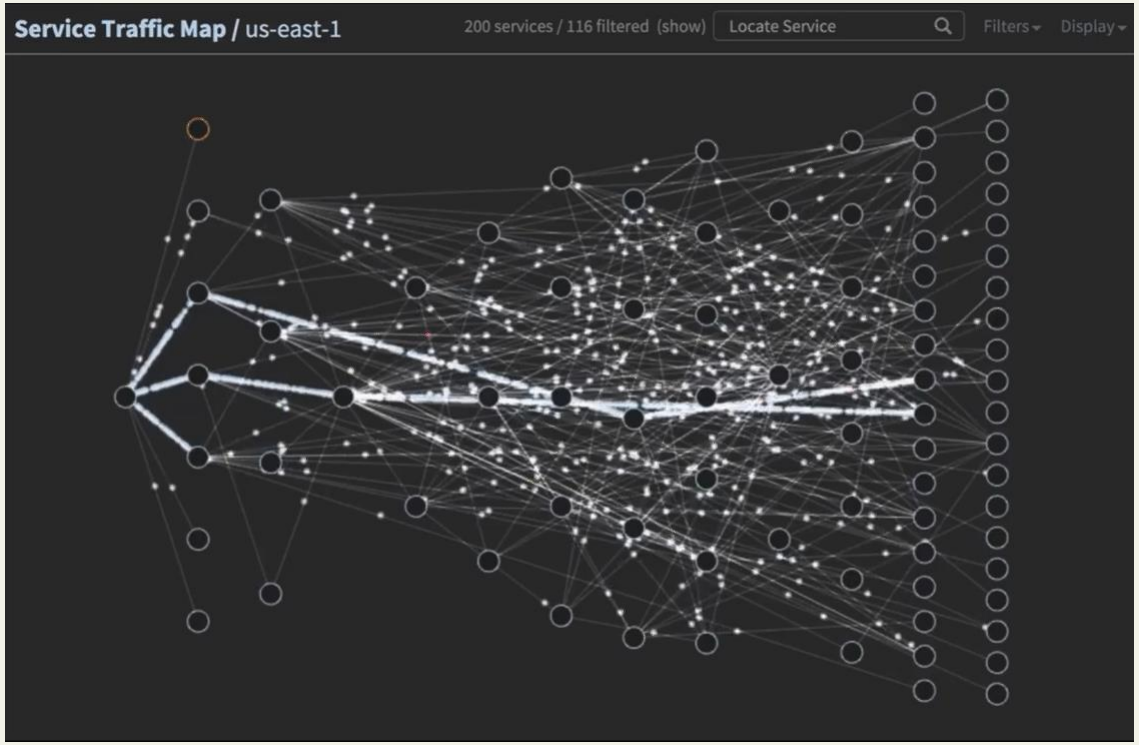


Real time traffic of Social Network Microservices

Visualization tool: https://github.com/netflix/vizceral.git

Marching to Microservices





Real time traffic of Social Network Microservices

- Complicate cluster management due to microservice dependencies

Visualization tool: https://github.com/netflix/vizceral.git

Infrastructure challenges for Microservices

Infrastructure challenges for Microservices

- **Performance unpredictability usually occurs at large scale**

Infrastructure challenges for Microservices

- Performance unpredictability usually occurs at large scale

- Leverage simulation

Infrastructure challenges for Microservices

- **Performance unpredictability usually occurs at large scale**

- **Leverage simulation**
  - Architecture simulator: Gem5, Zsim (ISCA'13)
  - Cluster level queueing simulator: BigHouse (ISPASS'12)

Infrastructure challenges for Microservices

- **Performance unpredictability usually occurs at large scale**
- **Leverage simulation**
  - Architecture simulator: Gem5, Zsim (ISCA'13)
  - Cluster level queueing simulator: BigHouse (ISPASS'12)
- **Microservices introduce new simulation requirements**

Infrastructure challenges for Microservices

- **Performance unpredictability usually occurs at large scale**
- **Leverage simulation**
  - Architecture simulator: Gem5, Zsim (ISCA'13)
  - Cluster level queueing simulator: BigHouse (ISPASS'12)
- **Microservices introduce new simulation requirements**
  - Scalability to study cluster management

Infrastructure challenges for Microservices

- **Performance unpredictability usually occurs at large scale**
- **Leverage simulation**
  - Architecture simulator: Gem5, Zsim (ISCA'13)
  - Cluster level queueing simulator: BigHouse (ISPASS'12)
- **Microservices introduce new simulation requirements**
  - Scalability to study cluster management
  - Accurate multi-stage models for individual service/microservice

Infrastructure challenges for Microservices

- **Performance unpredictability usually occurs at large scale**
- **Leverage simulation**
  - Architecture simulator: Gem5, Zsim (ISCA'13)
  - Cluster level queueing simulator: BigHouse (ISPASS'12)
- **Microservices introduce new simulation requirements**
  - Scalability to study cluster management
  - Accurate multi-stage models for individual service/microservice
  - Modeling arbitrary dependency graphs and dataflow paths

Infrastructure challenges for Microservices

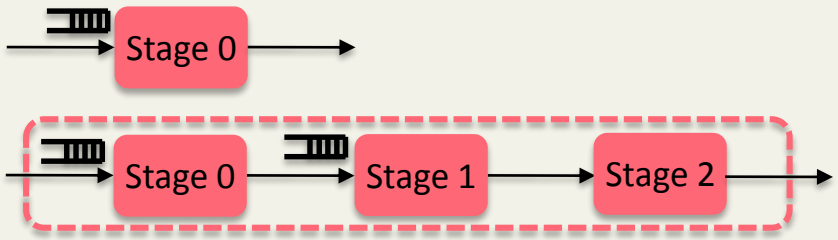
- **Performance unpredictability usually occurs at large scale**
- **Leverage simulation**
  - Architecture simulator: Gem5, Zsim (ISCA'13)
  - Cluster level queueing simulator: BigHouse (ISPASS'12)
- **Microservices introduce new simulation requirements**
  - Scalability to study cluster management
  - Accurate multi-stage models for individual service/microservice
  - Modeling arbitrary dependency graphs and dataflow paths
  - Modeling blocking and synchronization behavior

μQSim Design

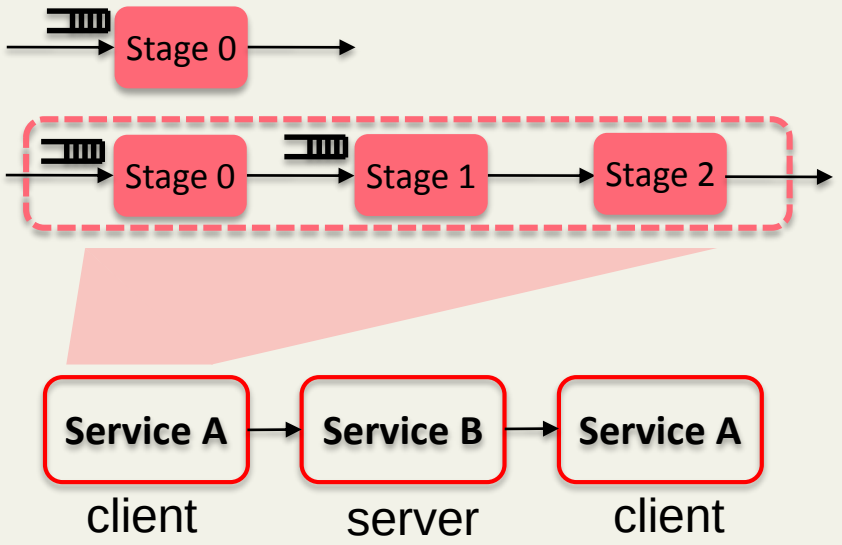
- **Accurate yet scalable service models**
  - Event-driven queueing simulator

- **Accurate yet scalable service models**
  - Event-driven queueing simulator
  - Simple microservice: single stage

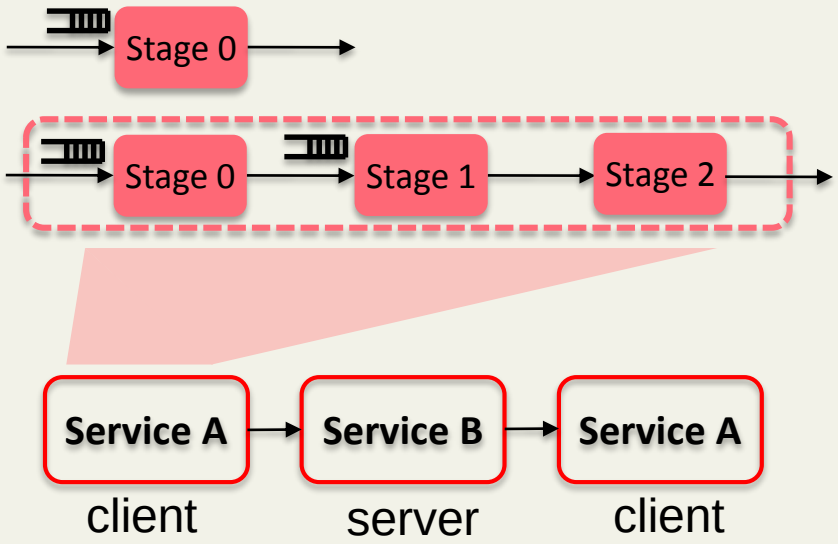
- **Accurate yet scalable service models**
  - Event-driven queueing simulator
  - Simple microservice: single stage
  - Complex service: decomposed to multiple stages



- **Accurate yet scalable service models**
  - Event-driven queueing simulator
  - Simple microservice: single stage
  - Complex service: decomposed to multiple stages
- **Modeling dependency graph & dataflow path**
  - Specify dependency graph & dataflow paths



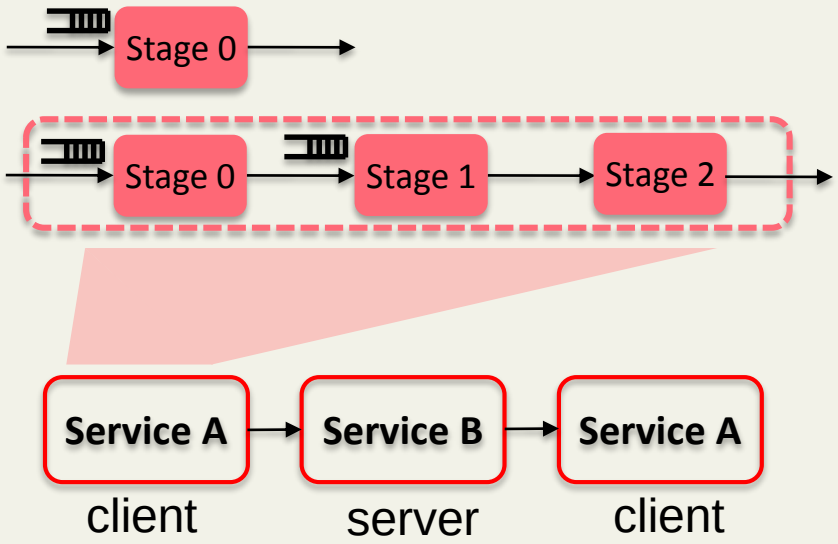
- **Accurate yet scalable service models**
  - Event-driven queueing simulator
  - Simple microservice: single stage
  - Complex service: decomposed to multiple stages



- **Modeling dependency graph & dataflow path**
  - Specify dependency graph & dataflow paths

- **Modeling blocking/synchronization**
  - Encode blocking/synchronization behavior in dataflow path

- **Accurate yet scalable service models**
  - Event-driven queueing simulator
  - Simple microservice: single stage
  - Complex service: decomposed to multiple stages



- **Modeling dependency graph & dataflow path**
  - Specify dependency graph & dataflow paths

- **Modeling blocking/synchronization**
  - Encode blocking/synchronization behavior in dataflow path
  - Model sources of blocking: network connections, threads blocked by I/O accesses

Modeling Individual Microservices

- **μqSim microservice model**

- **Memcached**

Modeling Individual Microservices

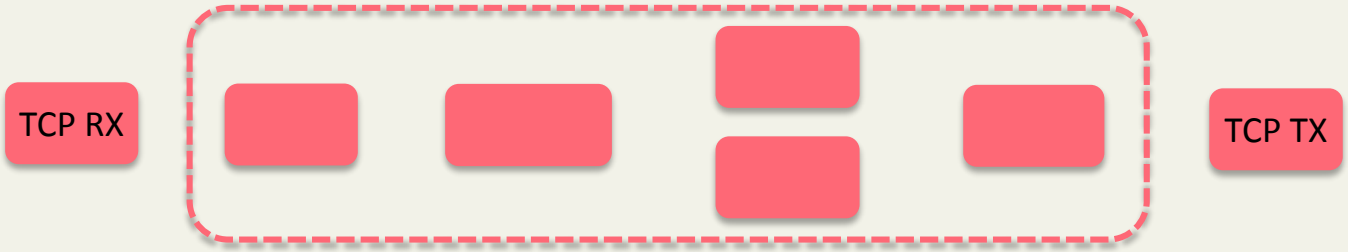
- **μqSim microservice model**
  - Multiple stages per microservice

- **Memcached**

Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice

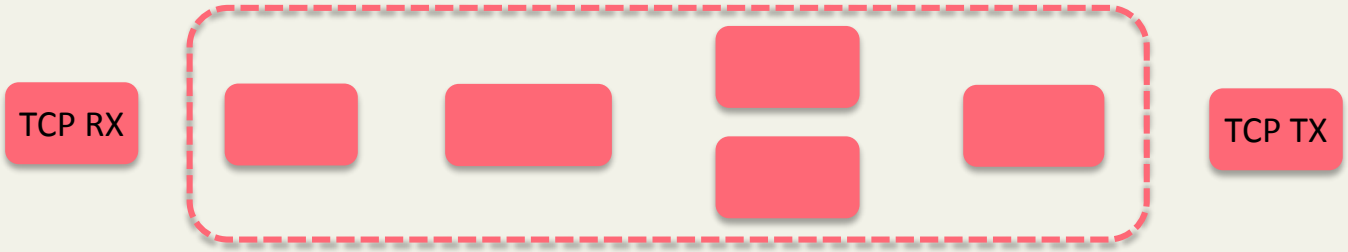
- **Memcached**



Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)

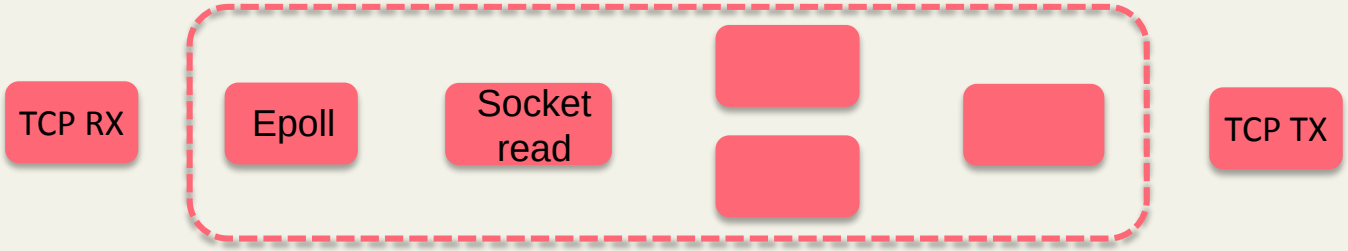
- **Memcached**



Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)

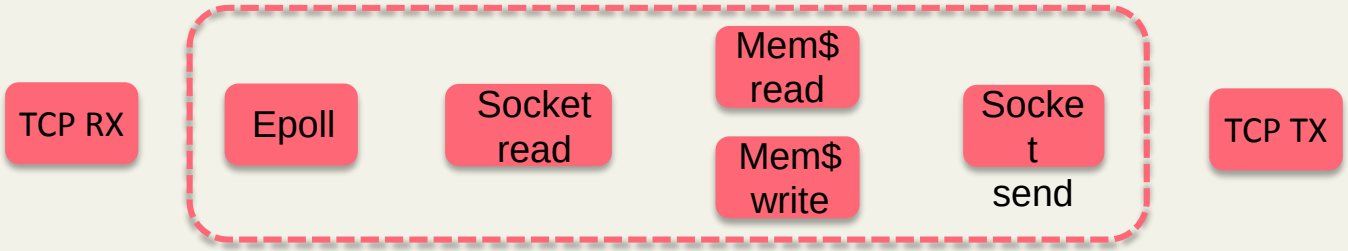
- **Memcached**



Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)

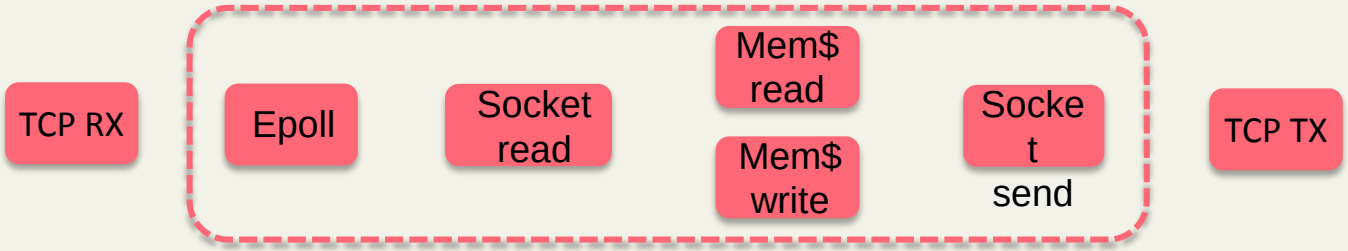
- **Memcached**



Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue

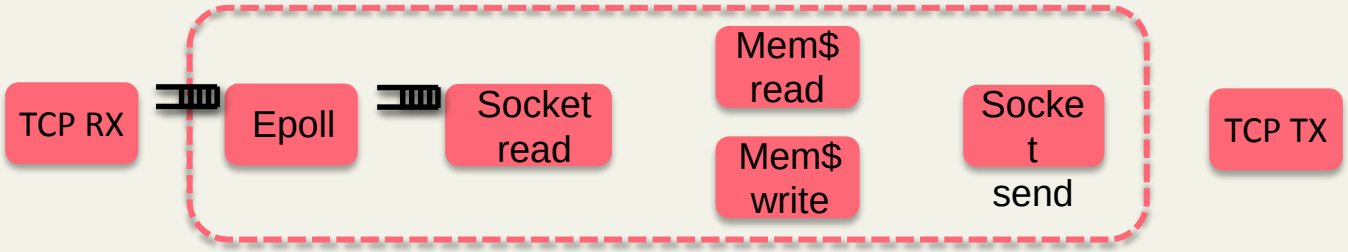
- **Memcached**



Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue

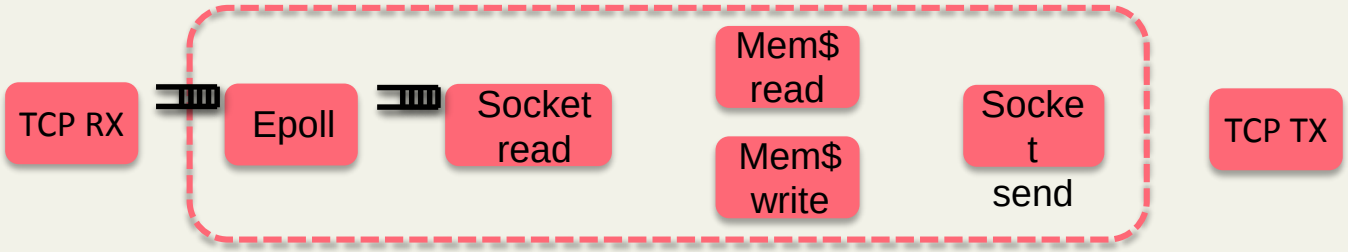
- **Memcached**



Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue
  - Multiple execution paths (e.g., Memcached read/write) per microservice

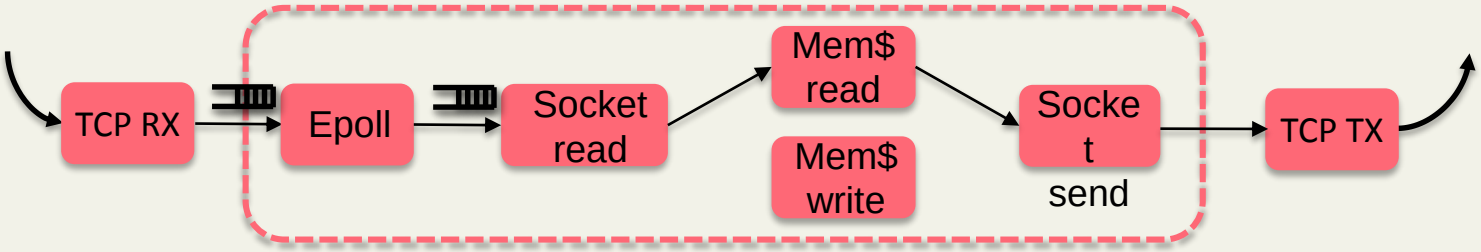
- **Memcached**



Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue
  - Multiple execution paths (e.g., Memcached read/write) per microservice

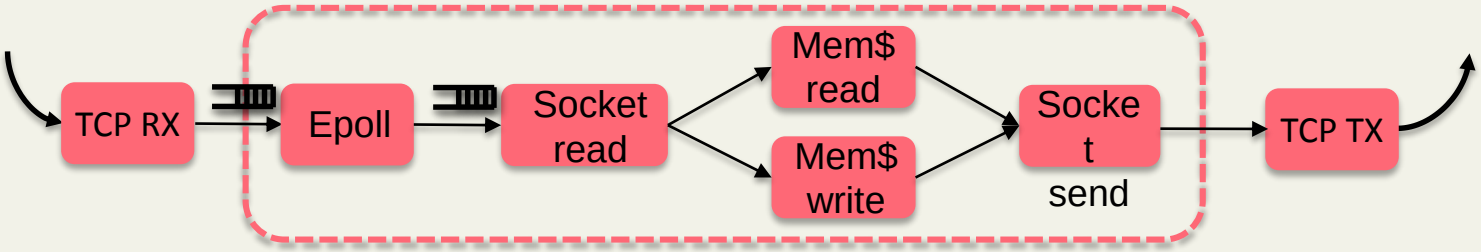
- **Memcached**



Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue
  - Multiple execution paths (e.g., Memcached read/write) per microservice

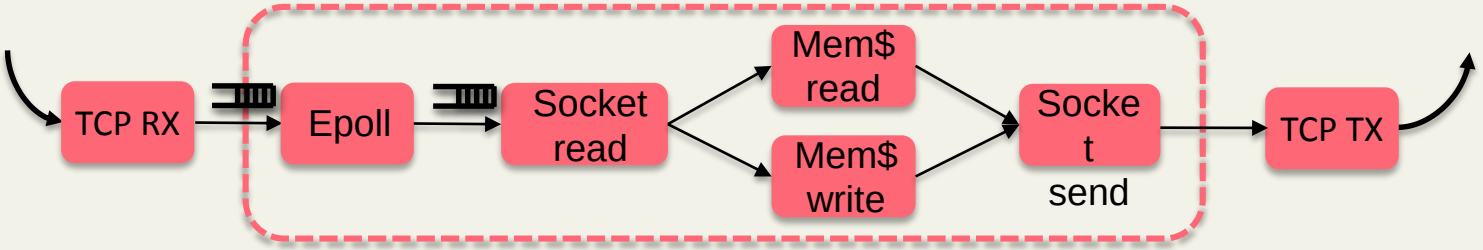
- **Memcached**



Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue
  - Multiple execution paths (e.g., Memcached read/write) per microservice

- **Memcached**



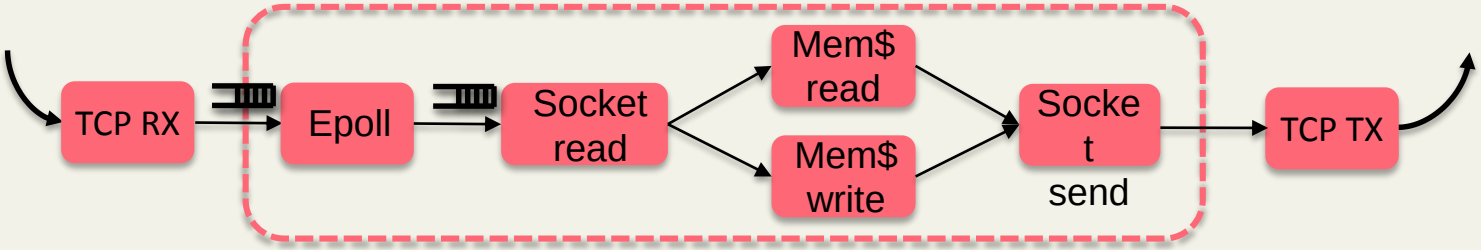
Connection A 1 2

Connection B 3

Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue
  - Multiple execution paths (e.g., Memcached read/write) per microservice

- **Memcached**



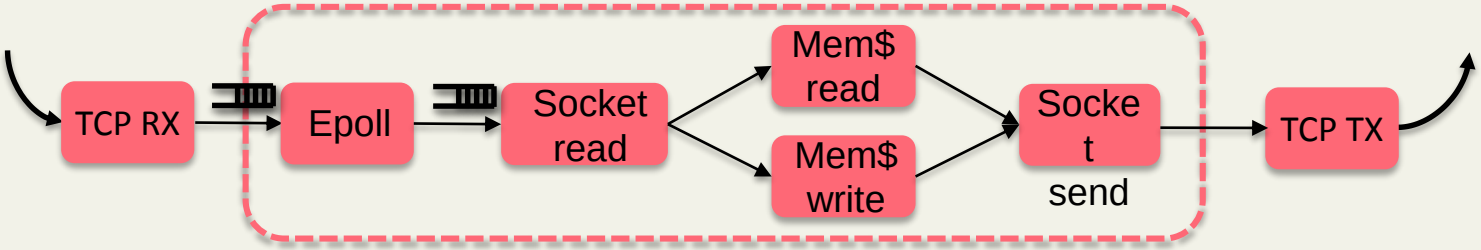
Connection A 1 2

Connection B 3

Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue
  - Multiple execution paths (e.g., Memcached read/write) per microservice

- **Memcached**



Connection A

Connection B

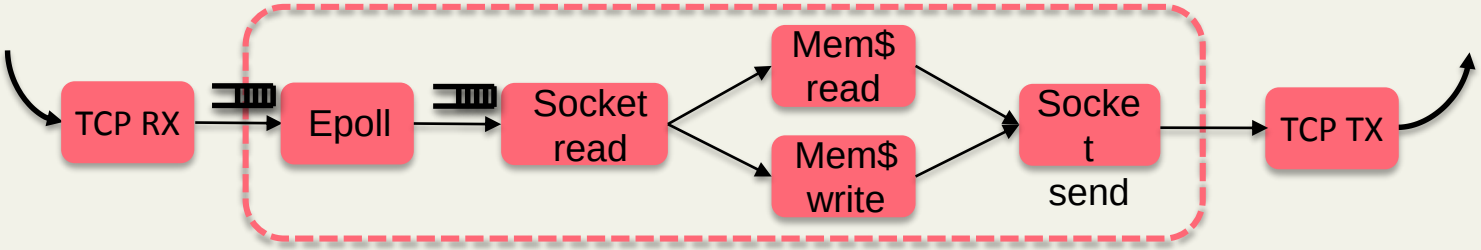
1 2

3

Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue
  - Multiple execution paths (e.g., Memcached read/write) per microservice

- **Memcached**



Connection A

Connection B

2

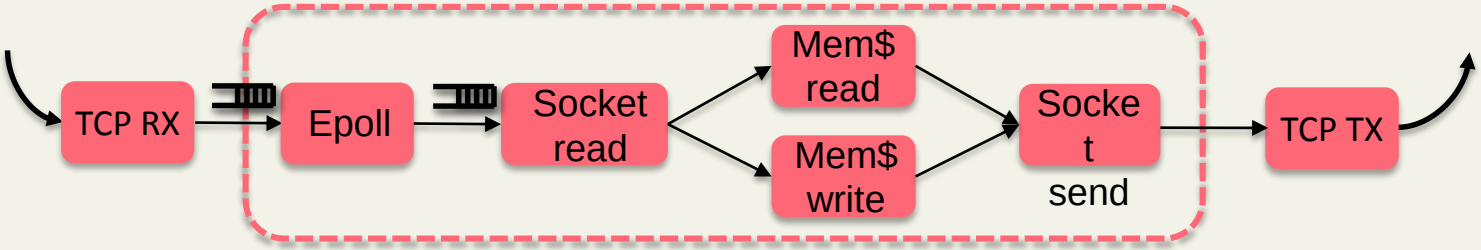
1

3

Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue
  - Multiple execution paths (e.g., Memcached read/write) per microservice

- **Memcached**



Connection A

Connection B

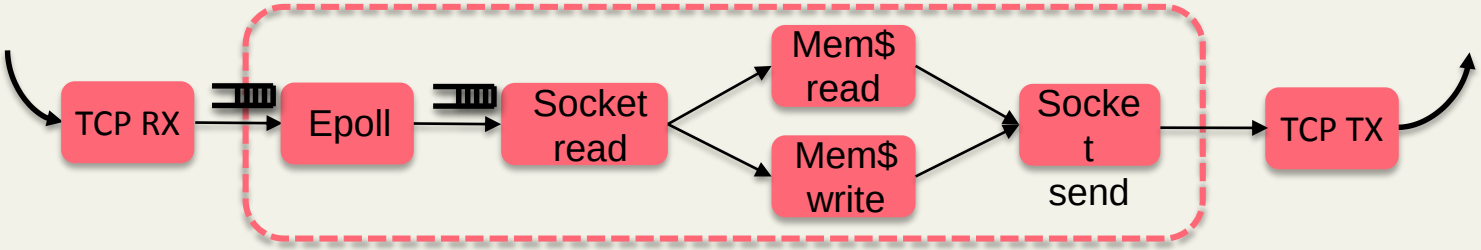
1 2

3

Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue
  - Multiple execution paths (e.g., Memcached read/write) per microservice

- **Memcached**



Connection A

Connection B

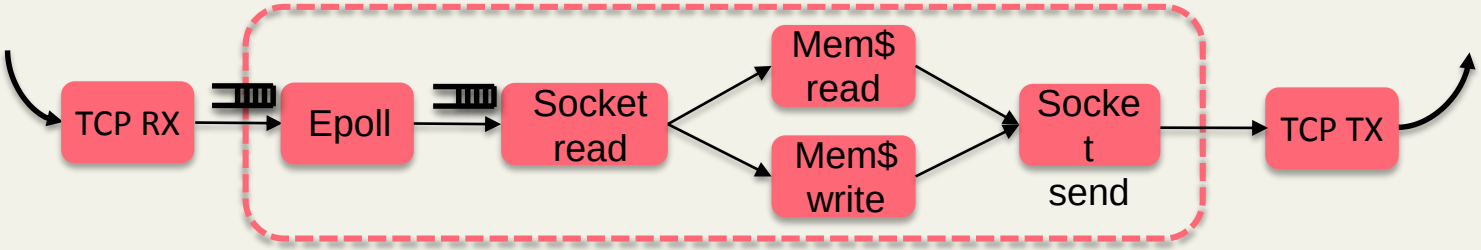
1 2

3

Modeling Individual Microservices

- **μqSim microservice model**
  - Multiple stages per microservice
  - Provided models for common OS queueing utilities (e.g., epoll & socket read)
  - Each stage optionally coupled with a queue
  - Multiple execution paths (e.g., Memcached read/write) per microservice

- **Memcached**



Connection A

Connection B

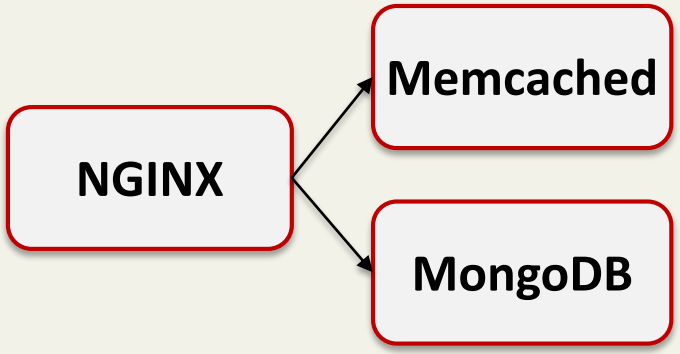
1 2

3

Modeling Dependency Graph & Dataflow

Modeling Dependency Graph & Dataflow

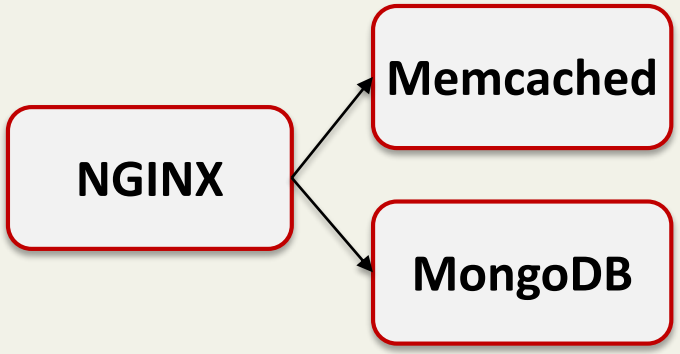
- **Microservice dependency graph**



Modeling Dependency Graph & Dataflow

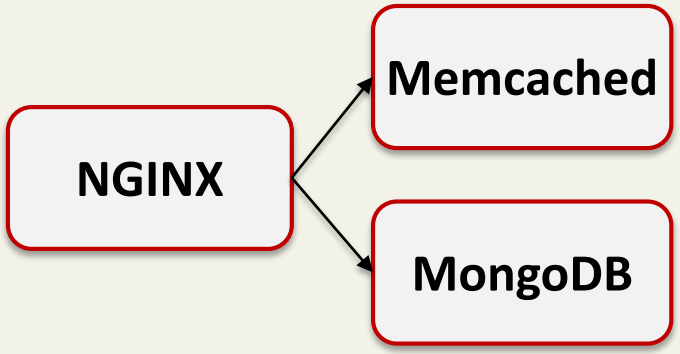
- **Microservice dependency graph**

- **Dataflow**
  - Sequence of microservices to execute
  - Differ across request types
  - Dataflow node encodes blocking/synchronization operation



Modeling Dependency Graph & Dataflow

- **Microservice dependency graph**



- **Dataflow**
  - Sequence of microservices to execute
  - Differ across request types
  - Dataflow node encodes blocking/synchronization operation

- **Deployment**
  - Available servers and hardware resources
  - Service to server mapping
  - Services on the same server share network stack & disk I/O

Modeling Dependency Graph & Dataflow

Modeling Dependency Graph & Dataflow

- Example: 2-tier application & http 1/1.1 protocol

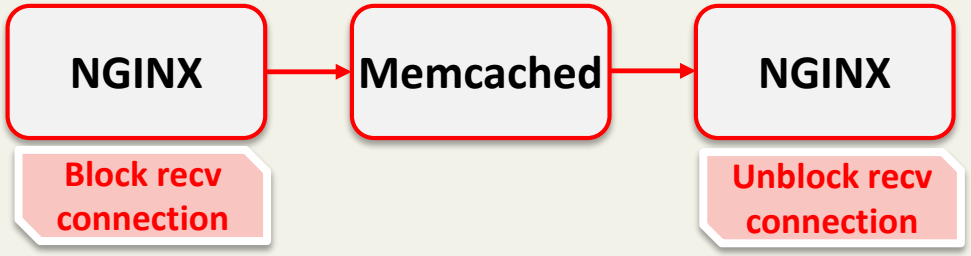
Modeling Dependency Graph & Dataflow

- **Example: 2-tier application & http 1/1.1 protocol**
  - Dependency graph

Modeling Dependency Graph & Dataflow

- **Example: 2-tier application & http 1/1.1 protocol**
  - Dependency graph

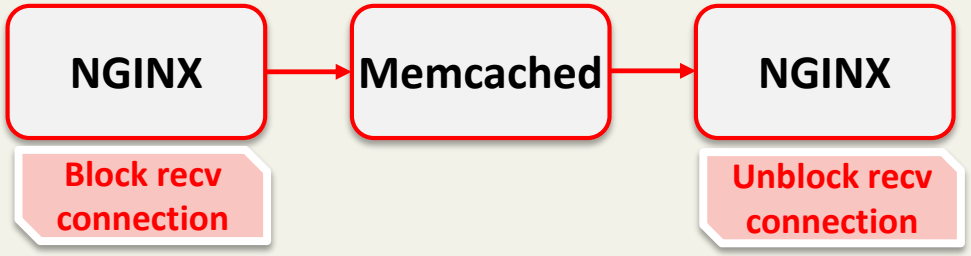
  - Dataflow



Modeling Dependency Graph & Dataflow

- **Example: 2-tier application & http 1/1.1 protocol**
  - Dependency graph

  - Dataflow



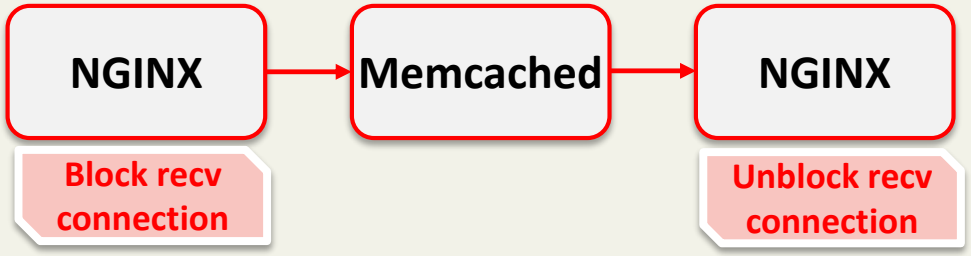
Connection A 1 2

Connection B 3

Modeling Dependency Graph & Dataflow

- **Example: 2-tier application & http 1/1.1 protocol**
  - Dependency graph

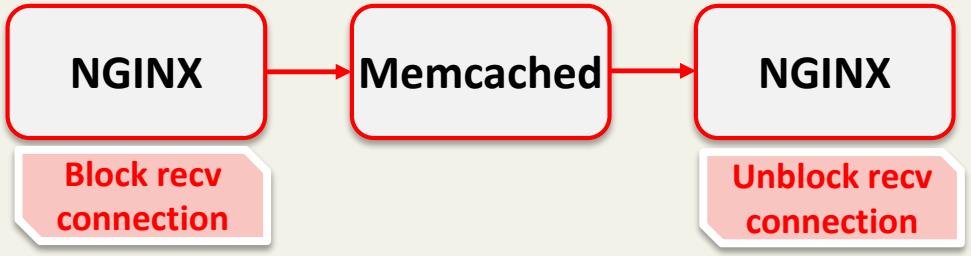
  - Dataflow



Modeling Dependency Graph & Dataflow

- **Example: 2-tier application & http 1/1.1 protocol**
  - Dependency graph

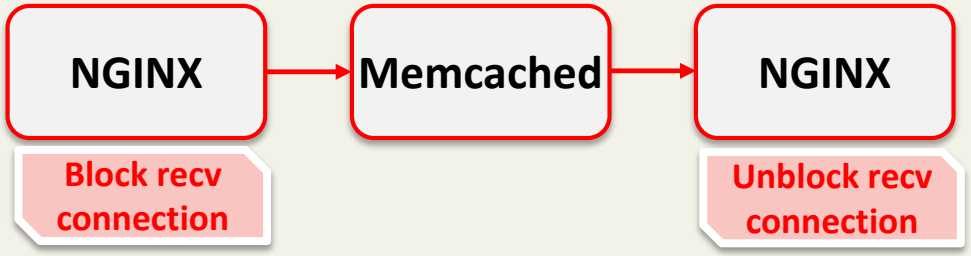
  - Dataflow



Modeling Dependency Graph & Dataflow

- **Example: 2-tier application & http 1/1.1 protocol**
  - Dependency graph

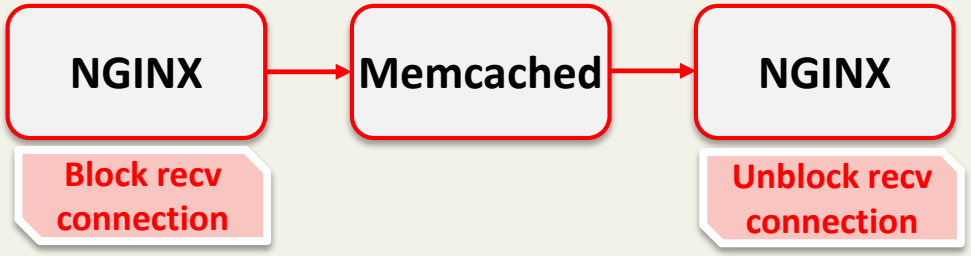
  - Dataflow



Modeling Dependency Graph & Dataflow

- **Example: 2-tier application & http 1/1.1 protocol**
  - Dependency graph

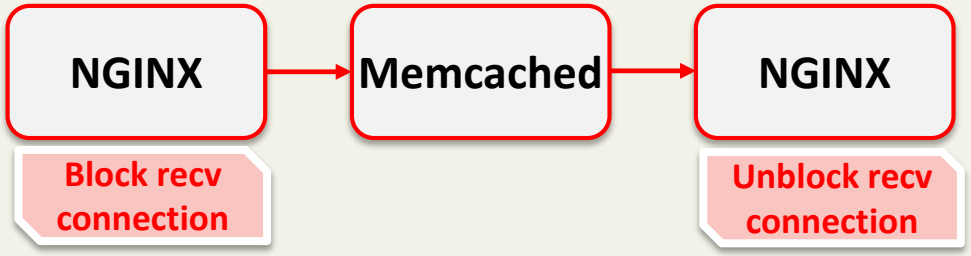
  - Dataflow



Modeling Dependency Graph & Dataflow

- **Example: 2-tier application & http 1/1.1 protocol**
  - Dependency graph

  - Dataflow



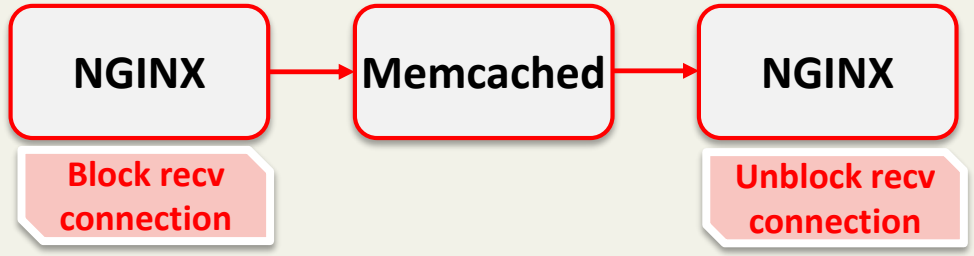
Connection A 1 2

Connection B 3

Modeling Dependency Graph & Dataflow

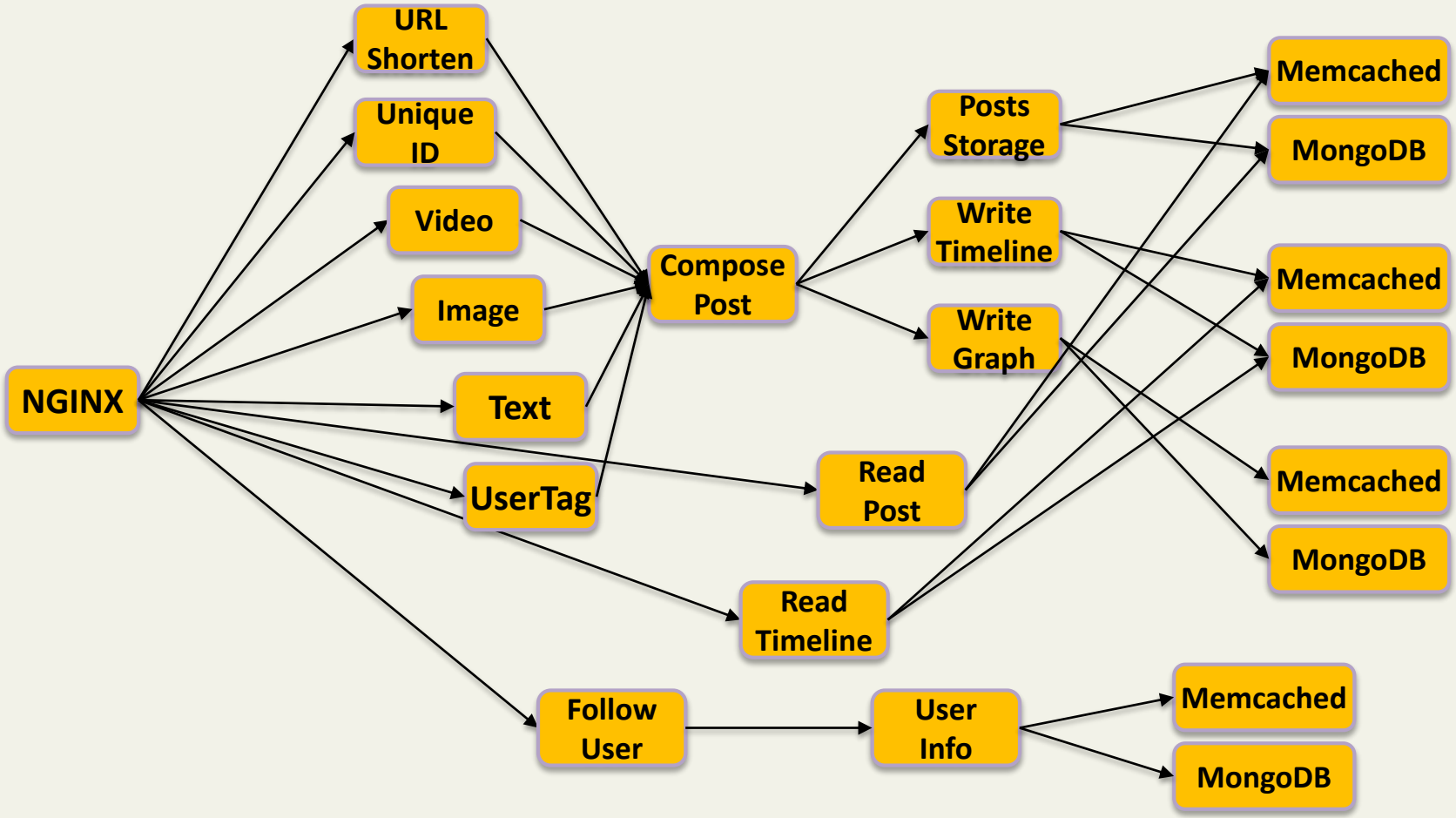
- **Example: 2-tier application & http 1/1.1 protocol**
  - Dependency graph

  - Dataflow



Dependency Graph for Complex Microservices

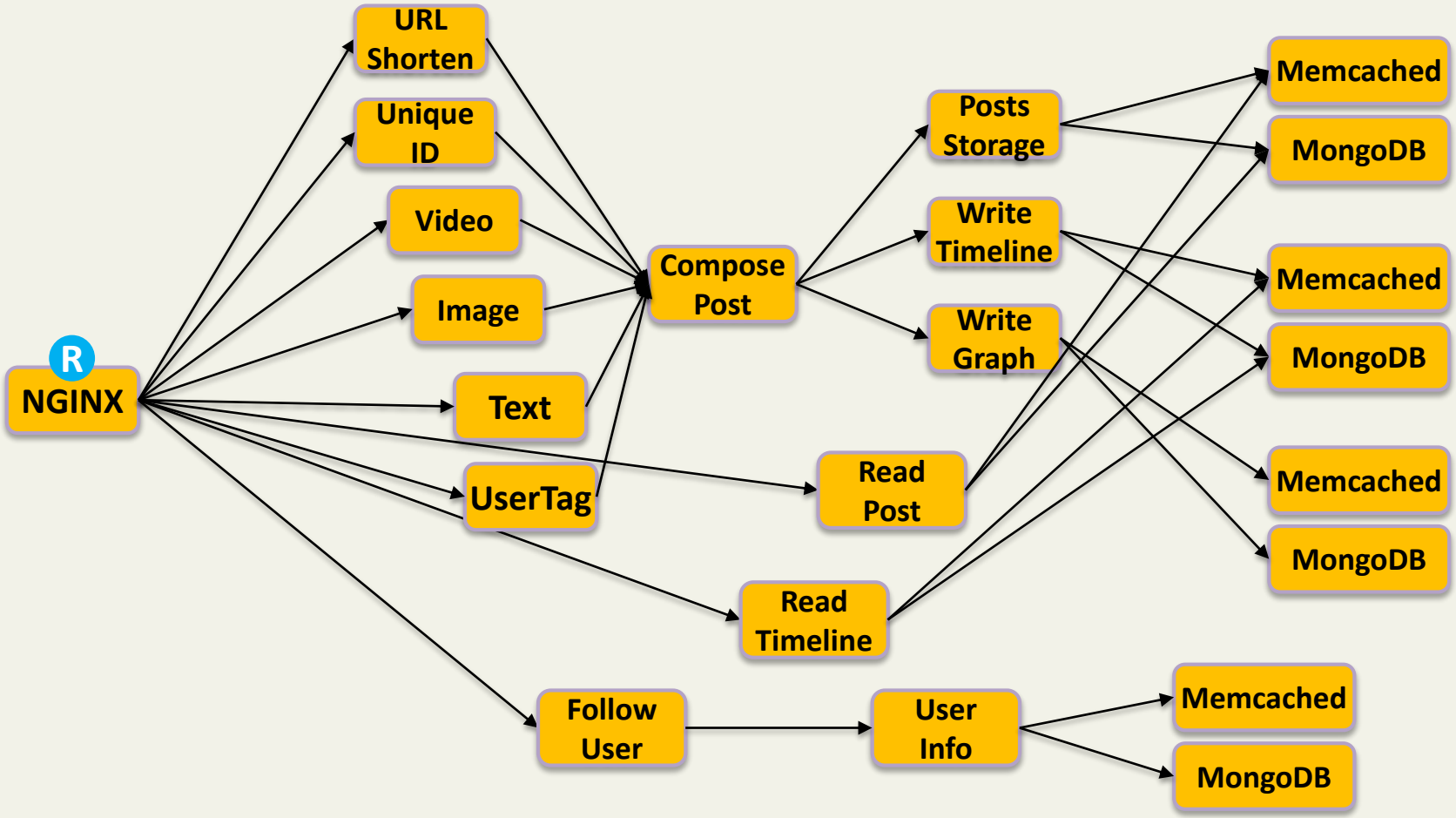
- Social network application



Source: http://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

Dependency Graph for Complex Microservices

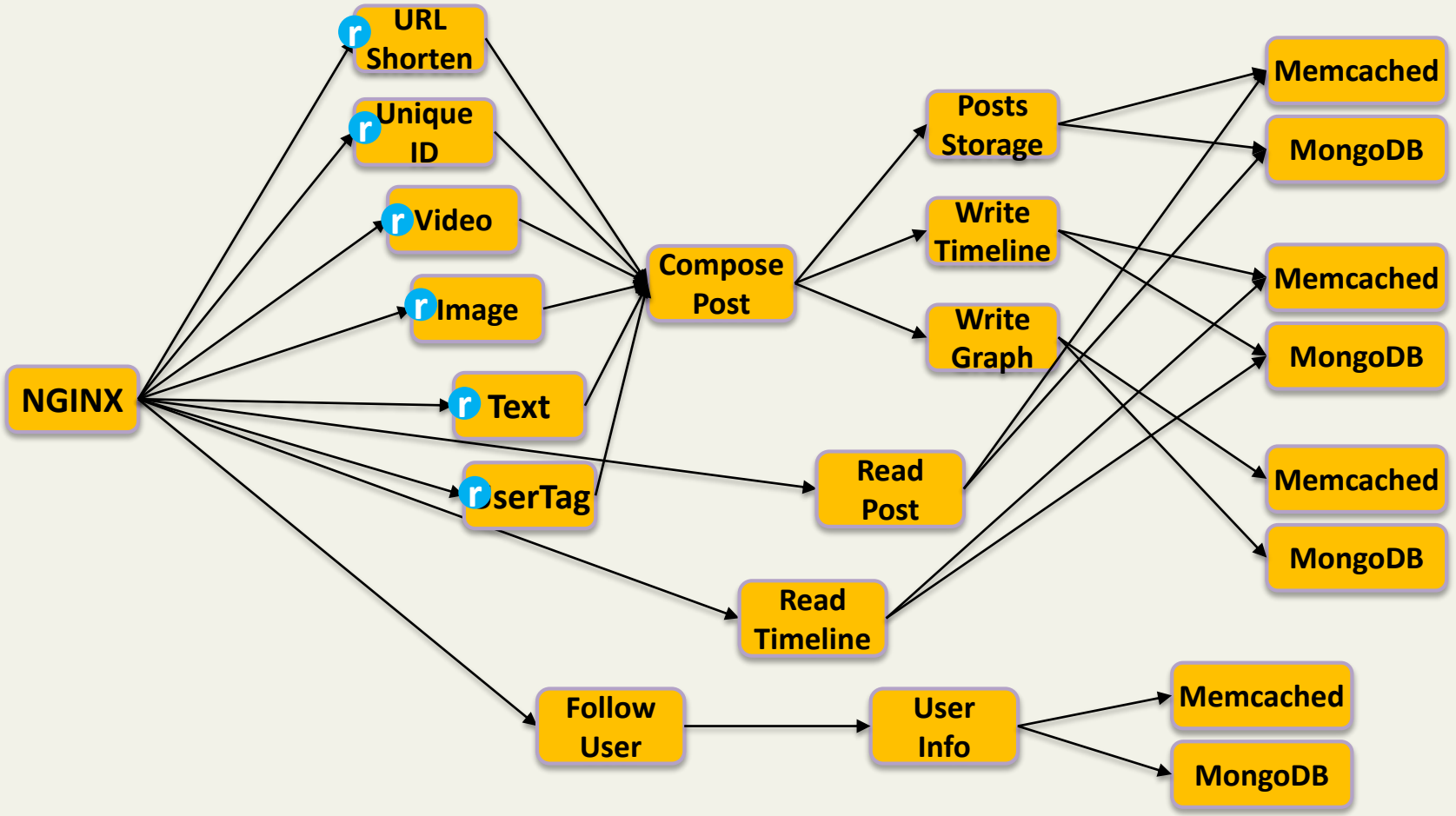
- Social network application



Source: http://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

Dependency Graph for Complex Microservices

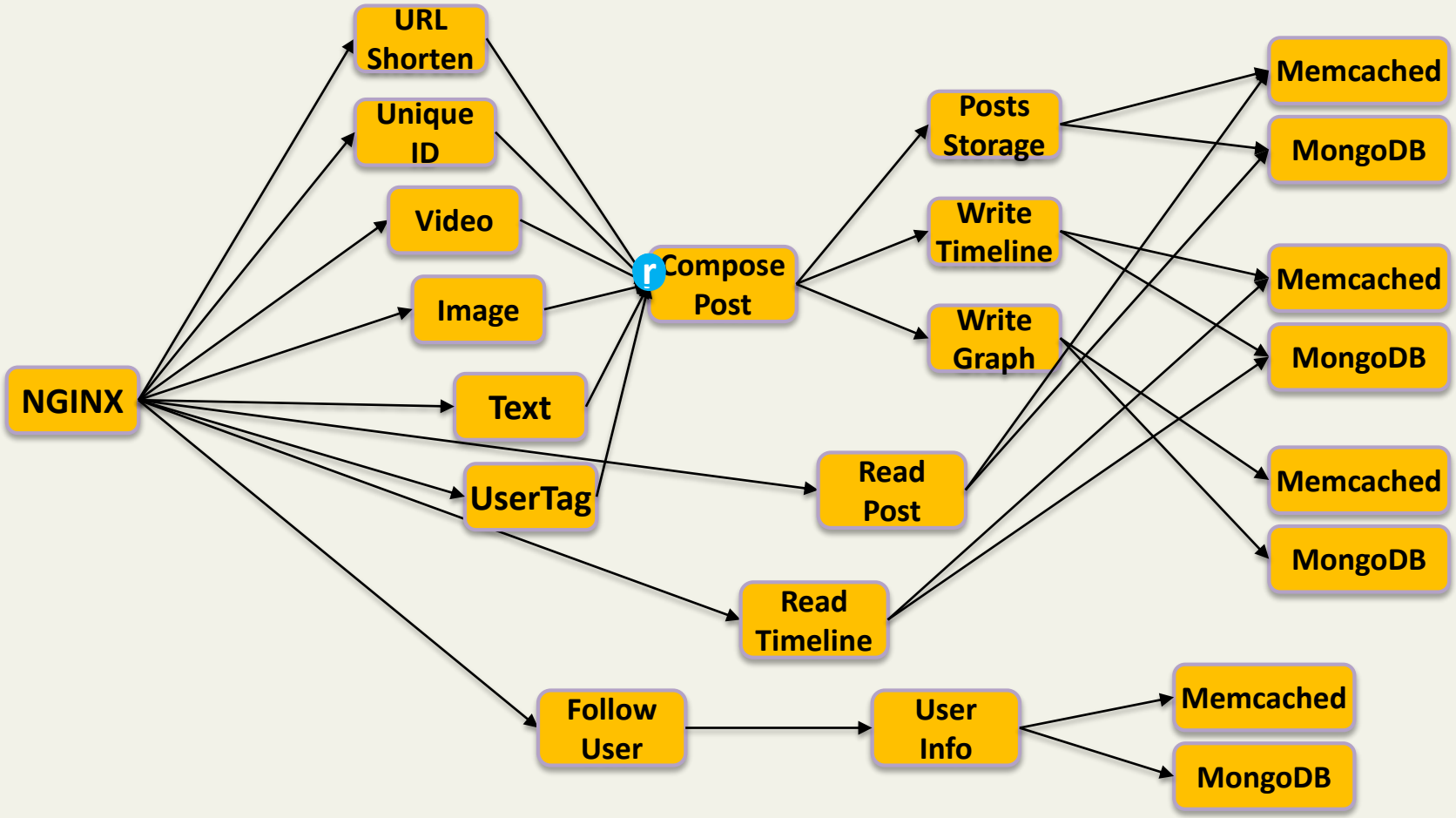
- Social network application



Source: http://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

Dependency Graph for Complex Microservices

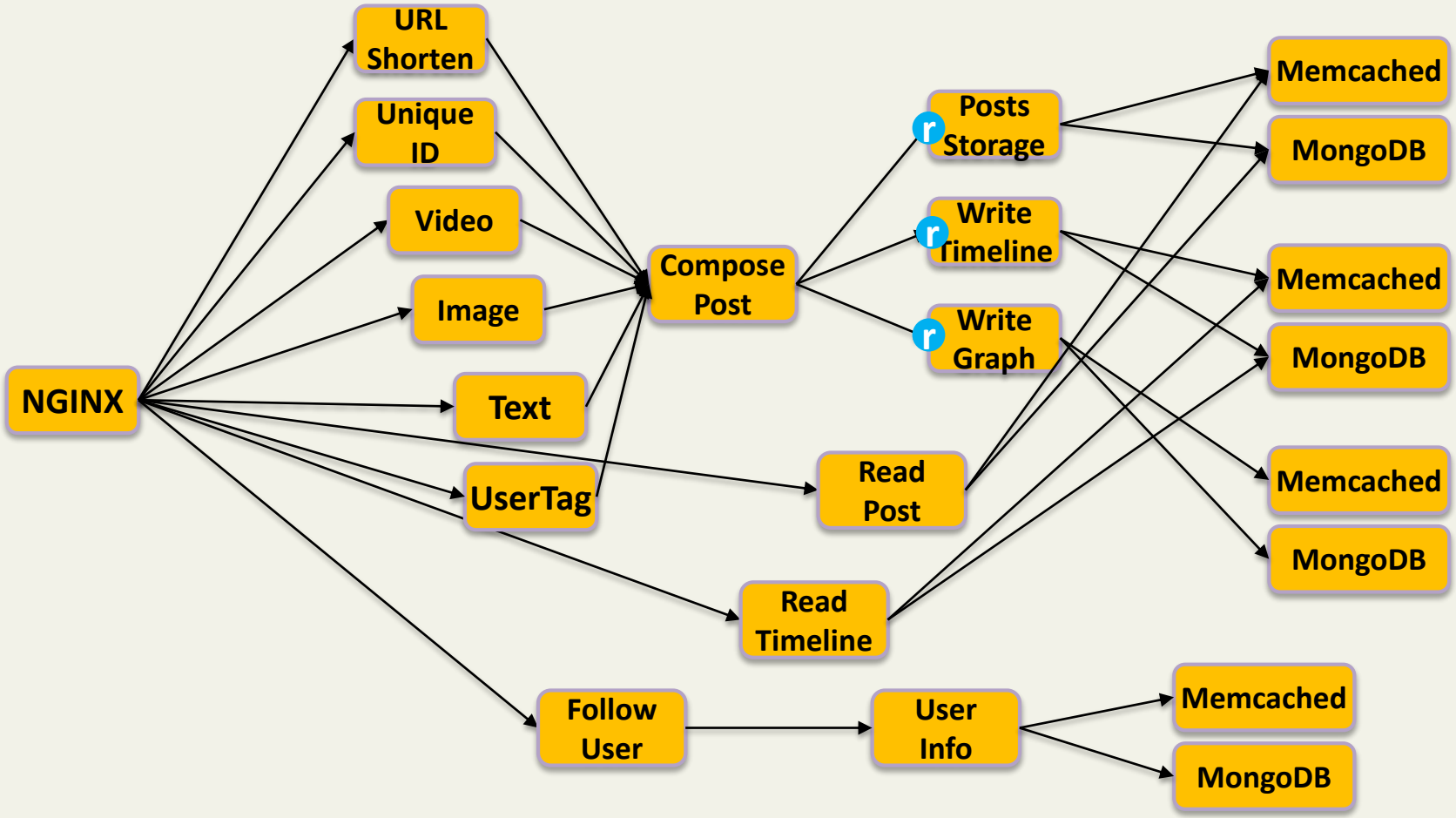
- Social network application



Source: http://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

Dependency Graph for Complex Microservices

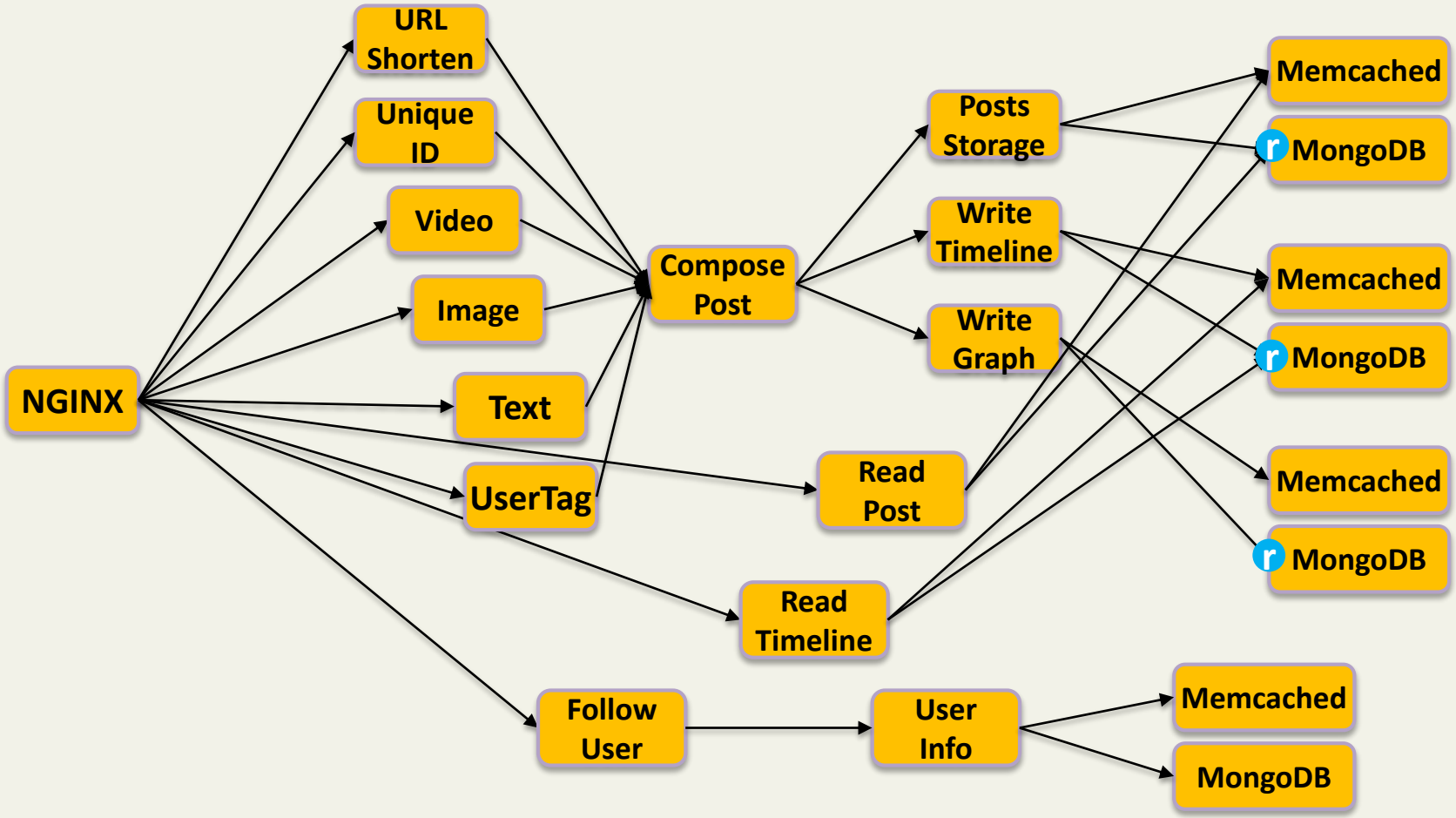
- Social network application



Source: http://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

Dependency Graph for Complex Microservices

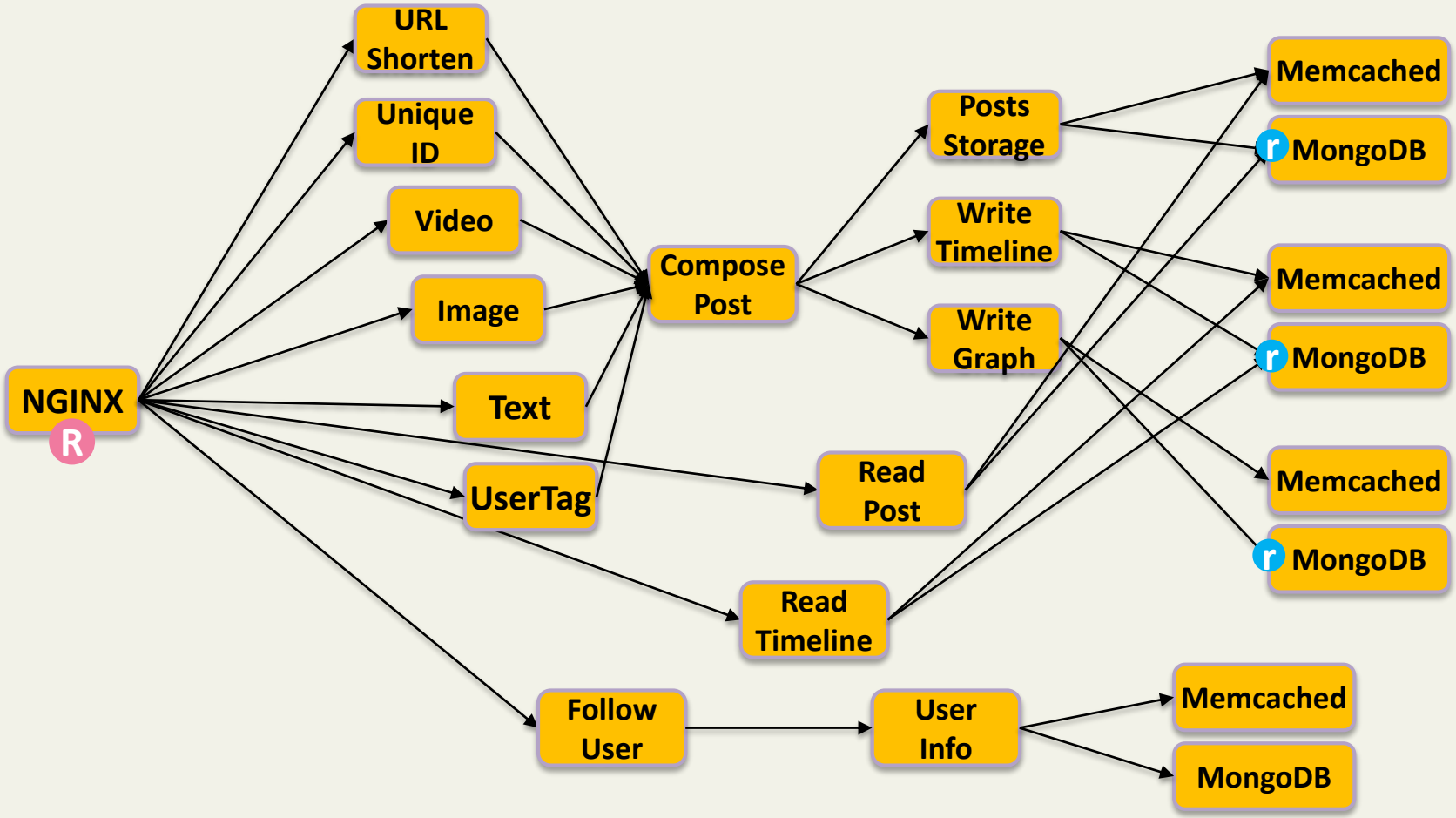
- Social network application



Source: http://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

Dependency Graph for Complex Microservices

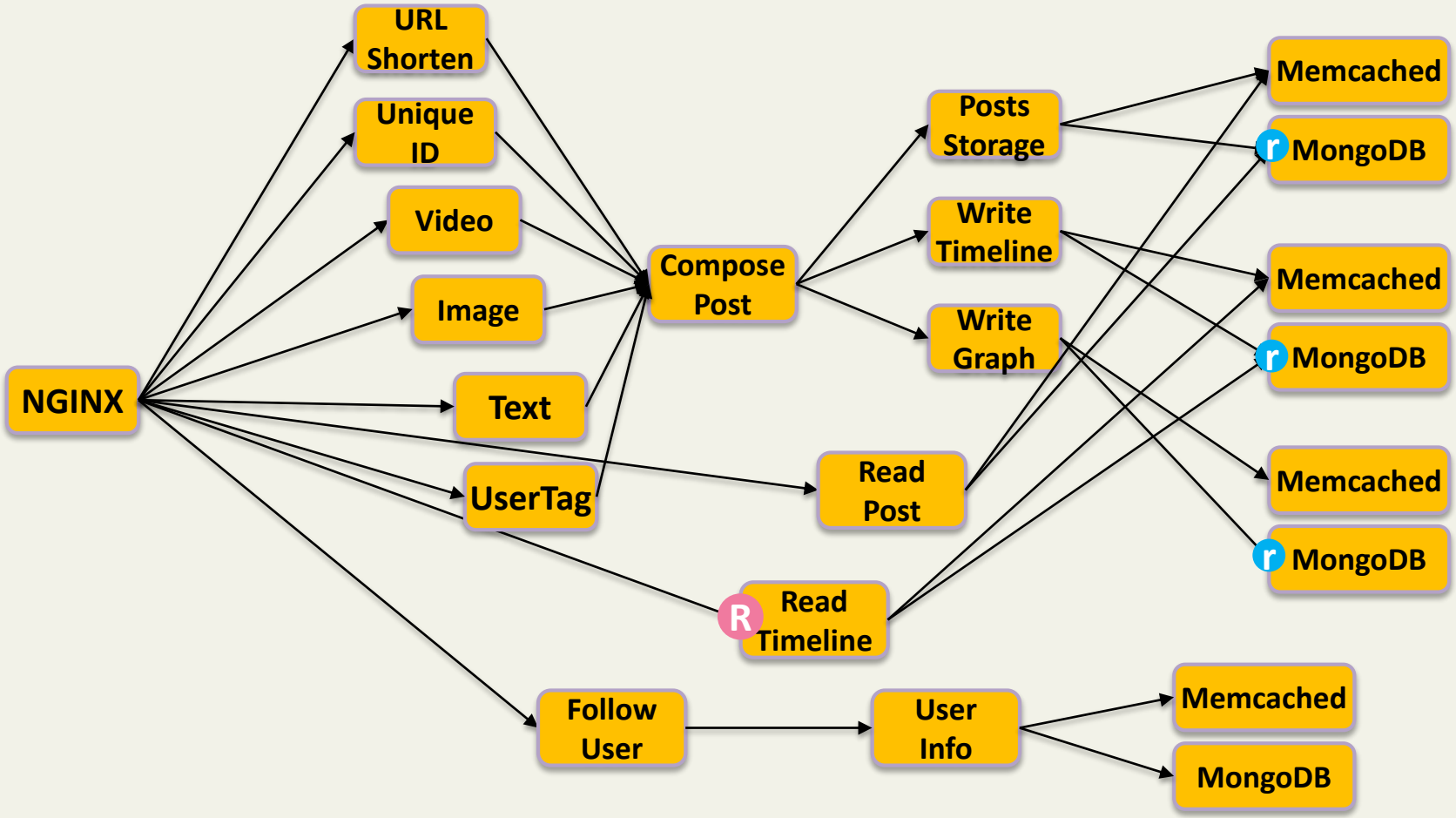
- Social network application



Source: http://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

Dependency Graph for Complex Microservices

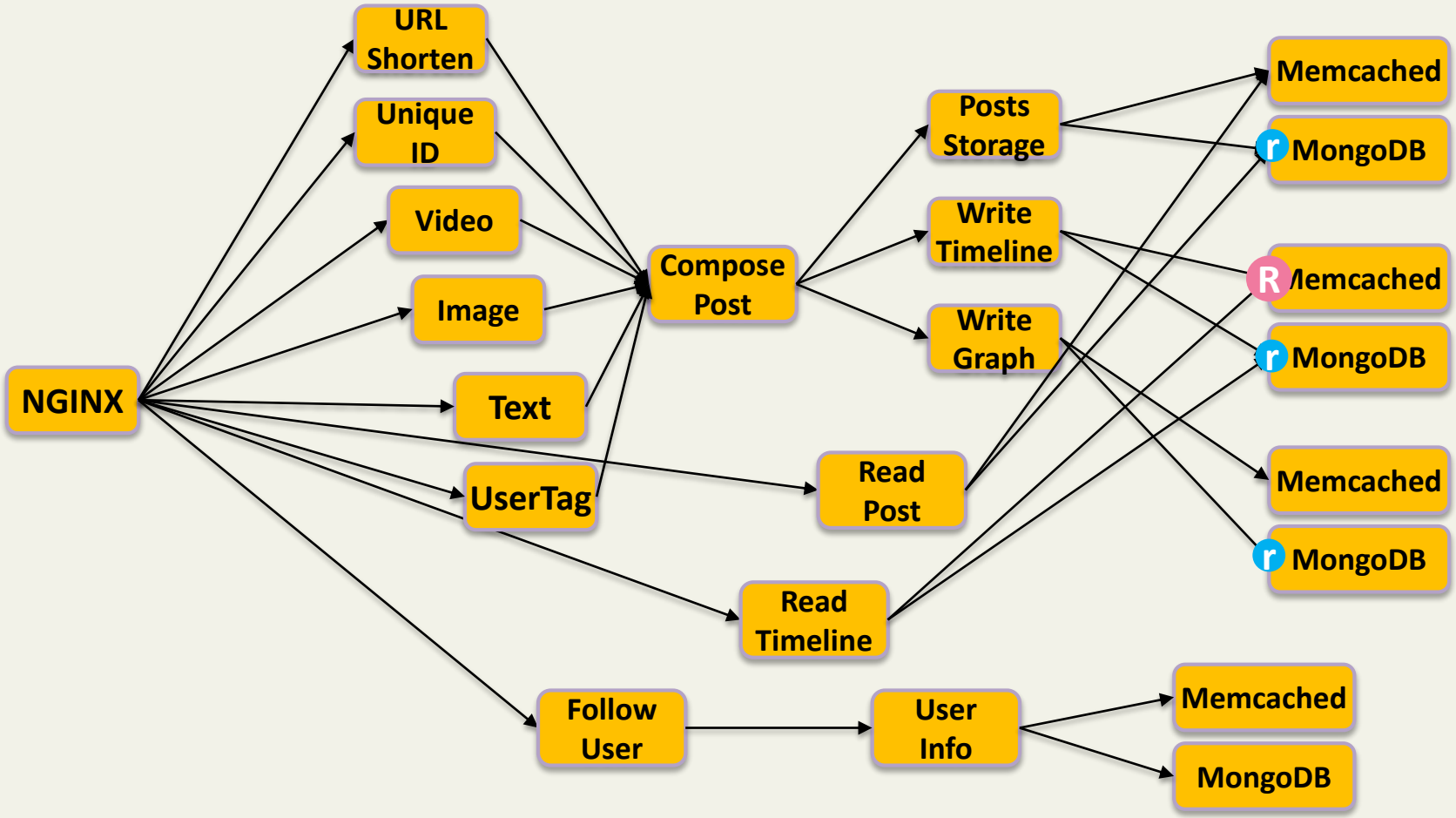
- Social network application



Source: http://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

Dependency Graph for Complex Microservices

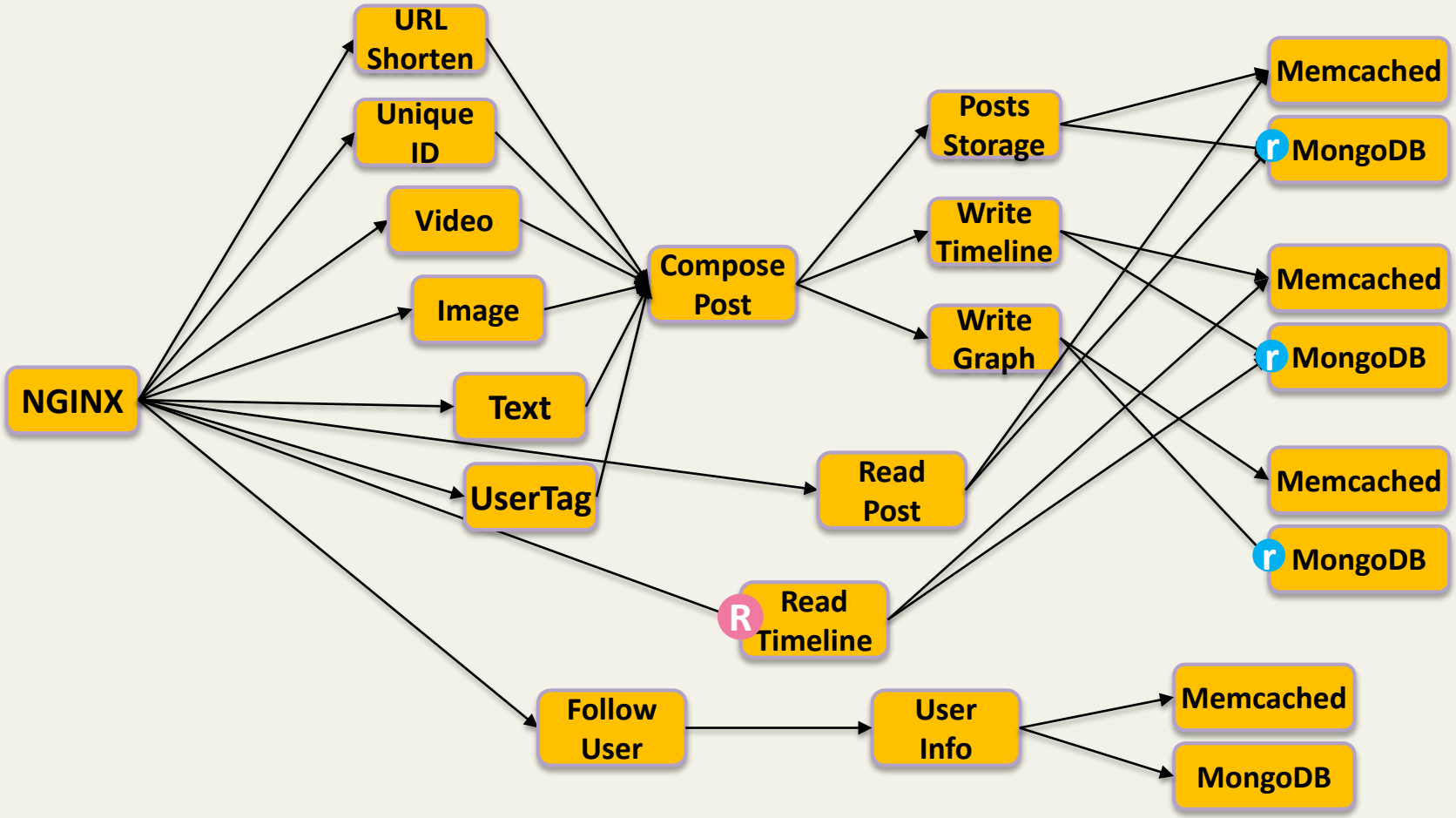
- Social network application



Source: http://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

Dependency Graph for Complex Microservices

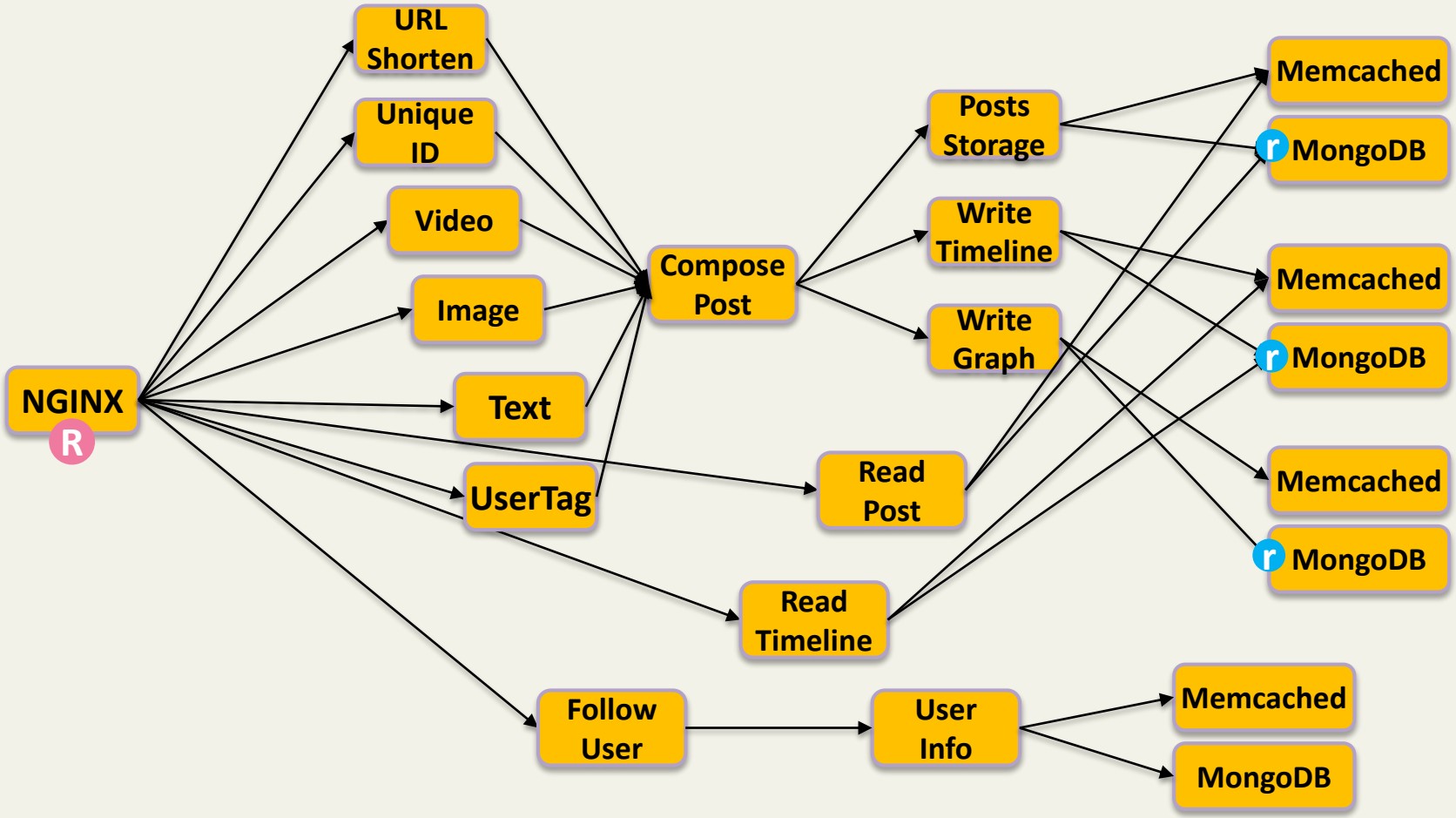
- Social network application



Source: http://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

Dependency Graph for Complex Microservices

- Social network application



Source: http://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

SIMULATOR INPUTS

Simulator inputs

- **Microservice queueing model**
  - Identifying sources of queueing
  - Typical queues: network, epoll/kqueue, socket read, disk I/O…
  - Microservice models reusable (open source community)

Simulator inputs

- **Microservice queueing model**
  - Identifying sources of queueing
  - Typical queues: network, epoll/kqueue, socket read, disk I/O…
  - Microservice models reusable (open source community)

- **Processing time distribution**
  - Instrumenting applications and profiling on real servers
  - Instrumentations reusable

Simulator inputs

- **Microservice queueing model**
  - Identifying sources of queueing
  - Typical queues: network, epoll/kqueue, socket read, disk I/O…
  - Microservice models reusable (open source community)
- **Processing time distribution**
  - Instrumenting applications and profiling on real servers
  - Instrumentations reusable
- **Microservice dependencies & dataflow paths**
  - Obtained from app developers

Simulator inputs

- **Microservice queueing model**
  - Identifying sources of queueing
  - Typical queues: network, epoll/kqueue, socket read, disk I/O…
  - Microservice models reusable (open source community)
- **Processing time distribution**
  - Instrumenting applications and profiling on real servers
  - Instrumentations reusable
- **Microservice dependencies & dataflow paths**
  - Obtained from app developers
- **Server & system resources**

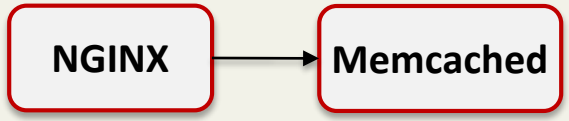
VALIDATION

- **Validation experiments**
  - Multi-tier microservices: 2/3-tier application
  - Load balancing & fanout effects
  - Microservices based on RPC
  - Comparison with BigHouse

- **Server platform for trace collection**
  - 10-server cluster
  - Intel(R) Xeon(R) CPU E5-2660 v3
  - 2 sockets, 10 cores/socket, 2 threads/core
  - Min/max DVFS frequency: 1.2GHz/2.6GHz
  - Network bandwidth: 1Gbps

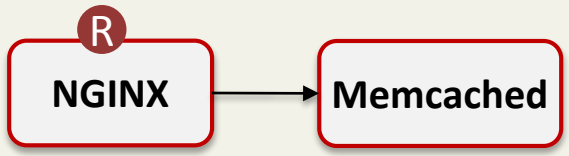
2/3-Tier Application

- 2-tier application



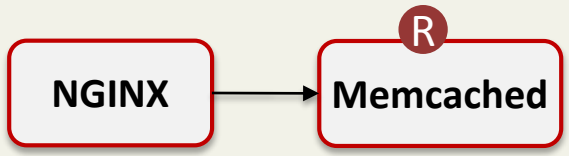
2/3-Tier Application

- 2-tier application



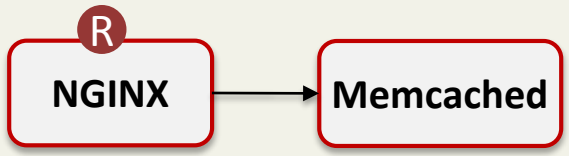
2/3-Tier Application

- 2-tier application



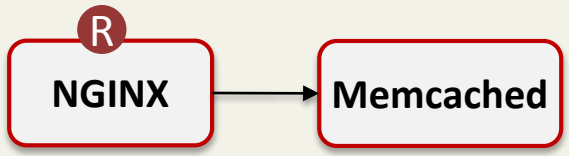
2/3-Tier Application

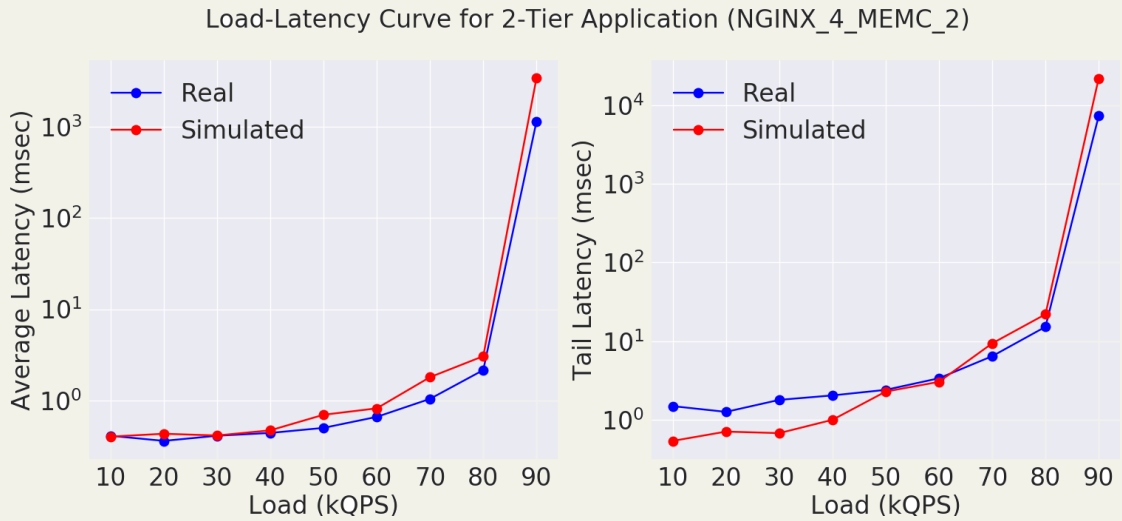
- 2-tier application



2/3-Tier Application

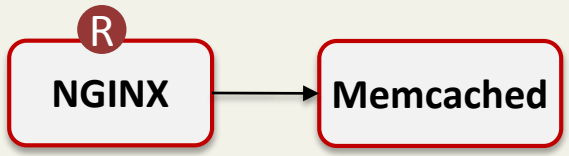
- 2-tier application



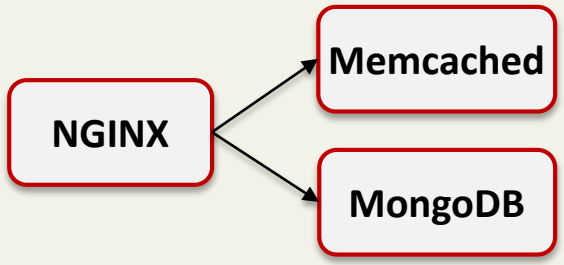


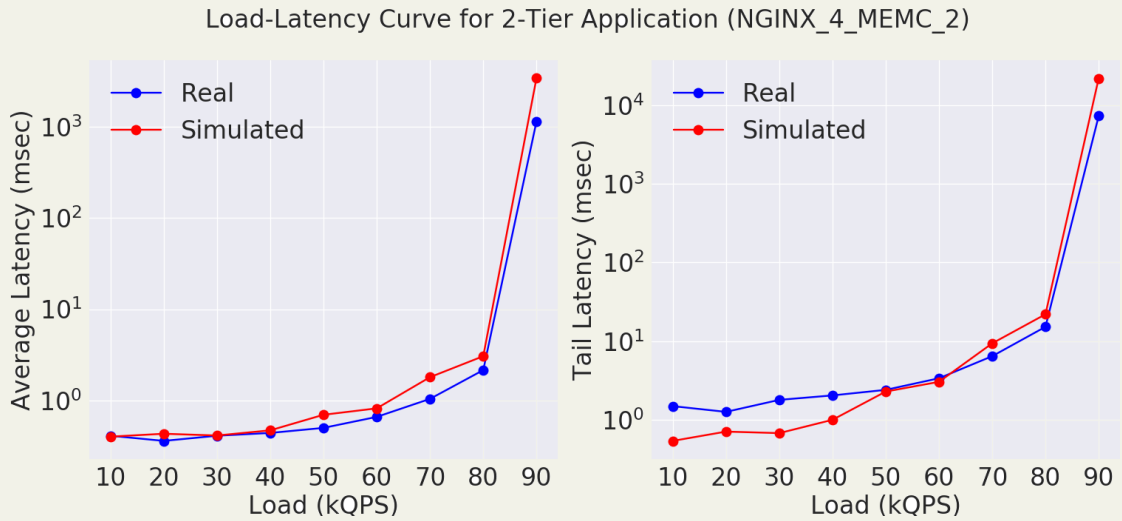
2/3-Tier Application

- 2-tier application



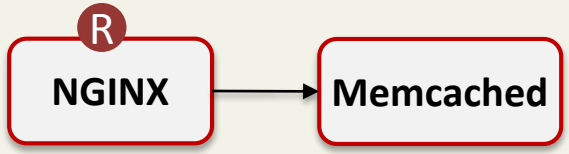
- 3-tier application



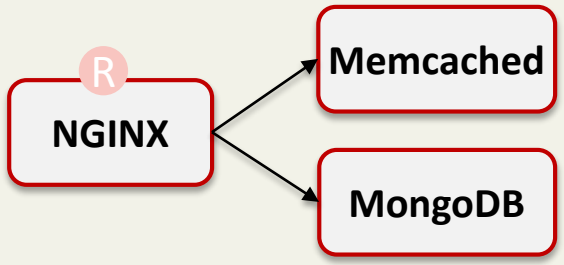


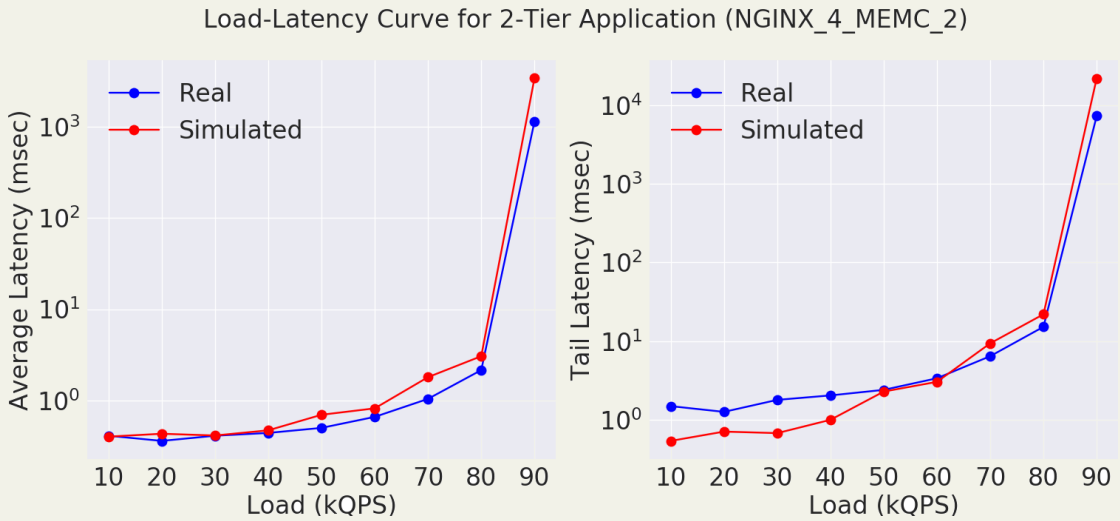
2/3-Tier Application

- 2-tier application



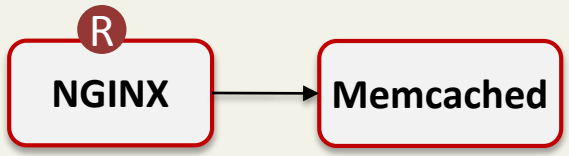
- 3-tier application



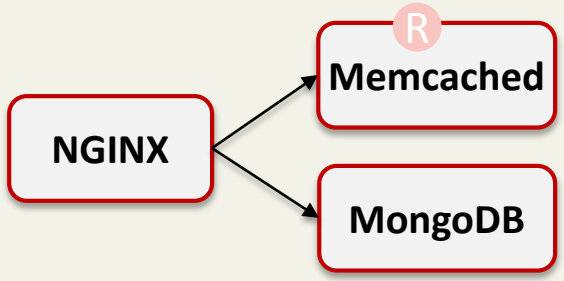


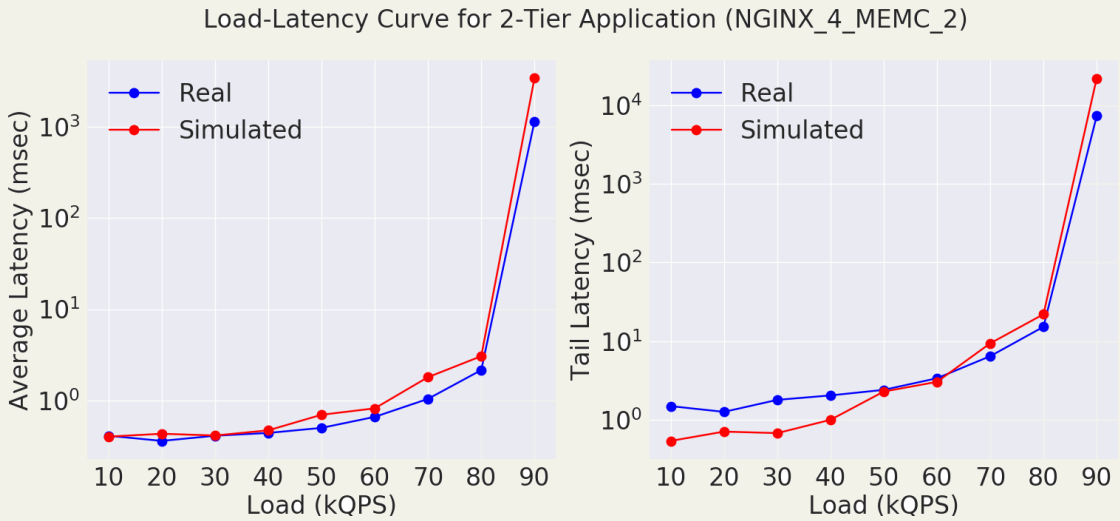
2/3-Tier Application

- 2-tier application



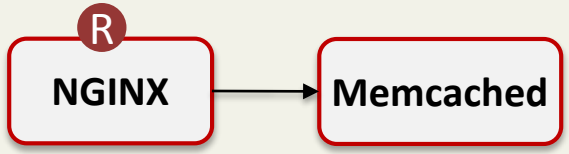
- 3-tier application



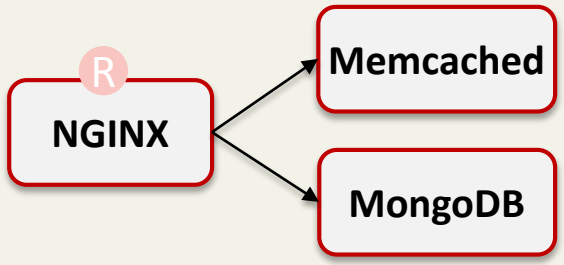


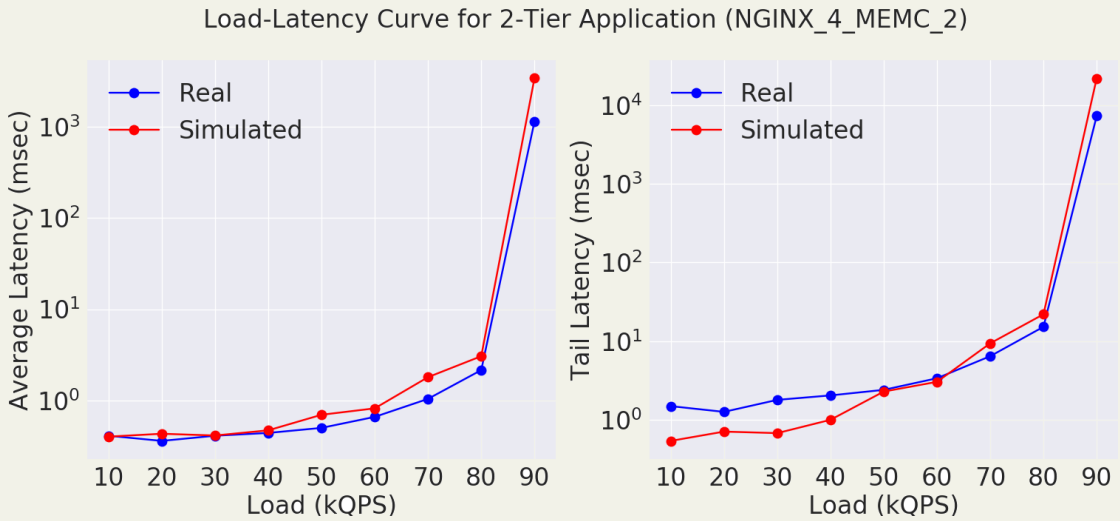
2/3-Tier Application

- 2-tier application



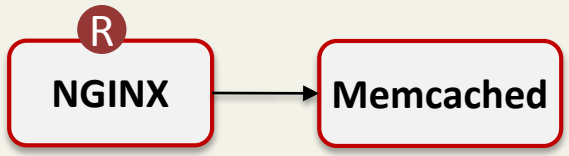
- 3-tier application



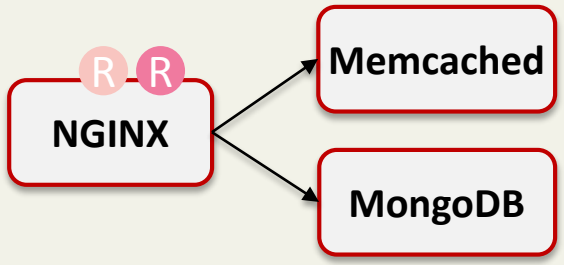


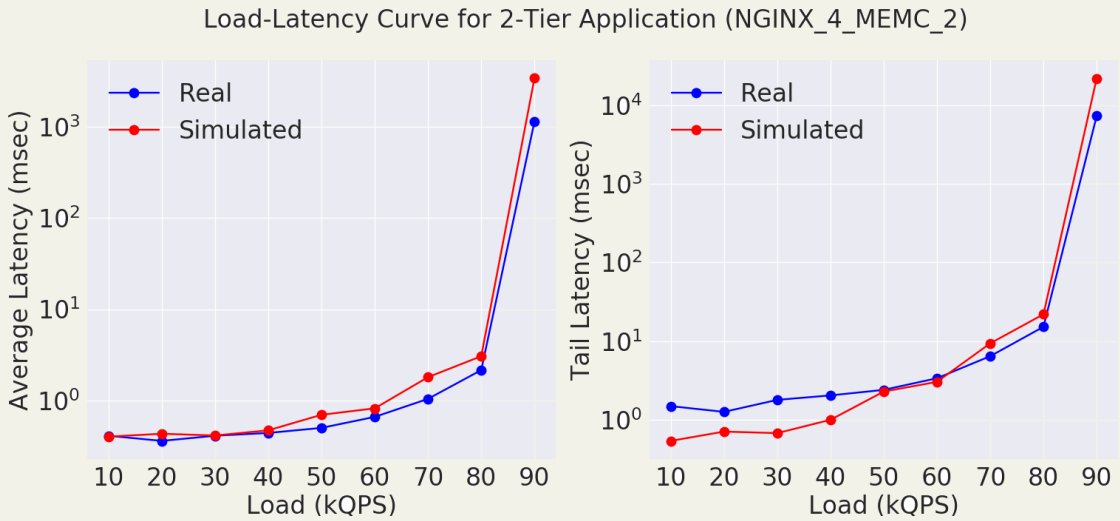
2/3-Tier Application

- 2-tier application



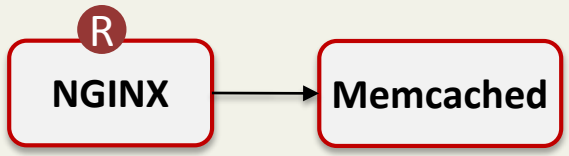
- 3-tier application



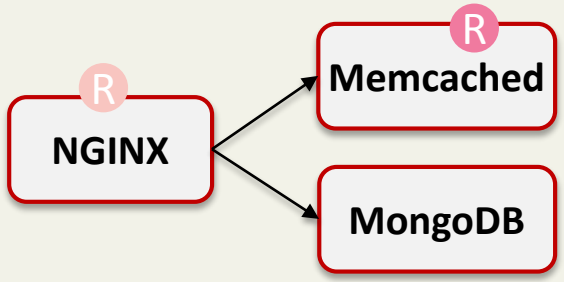


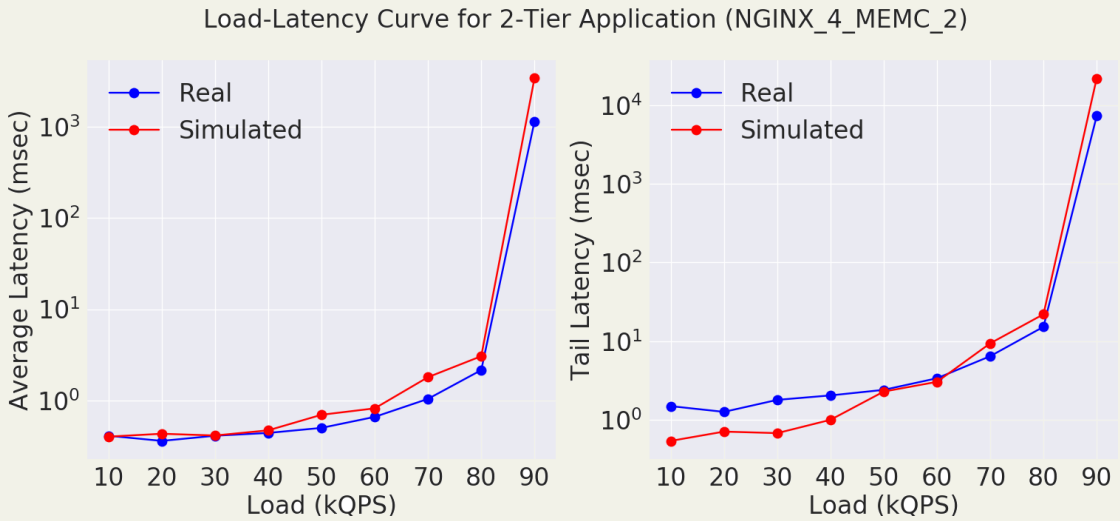
2/3-Tier Application

- 2-tier application



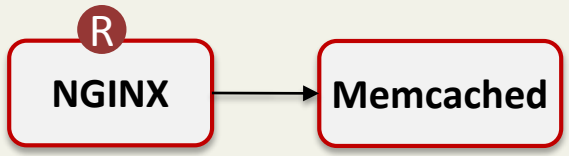
- 3-tier application



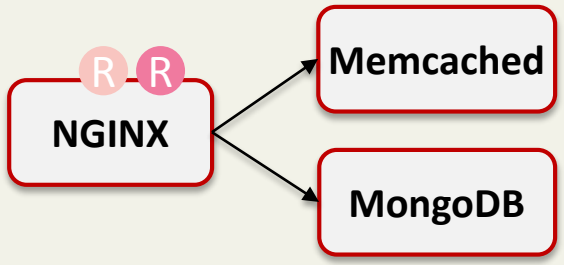


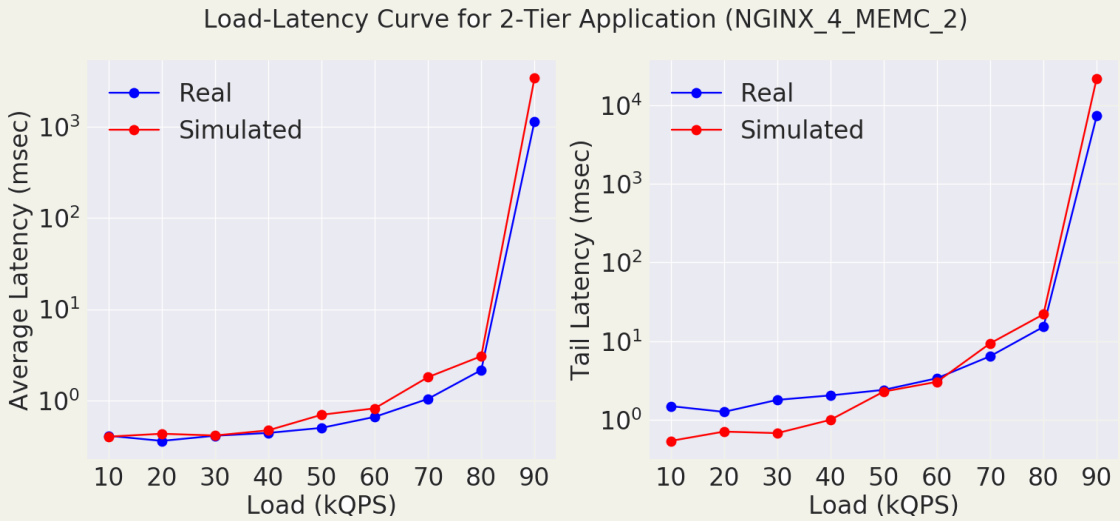
2/3-Tier Application

- 2-tier application



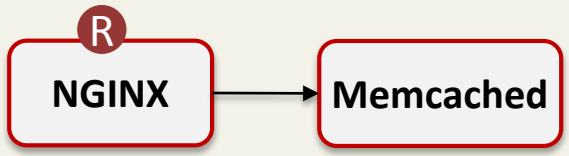
- 3-tier application



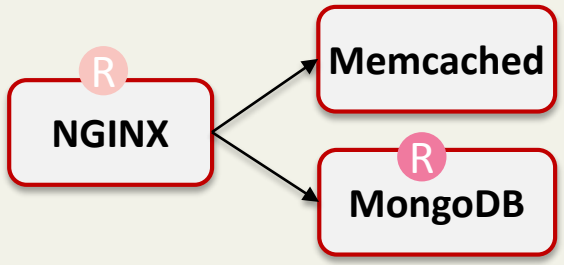


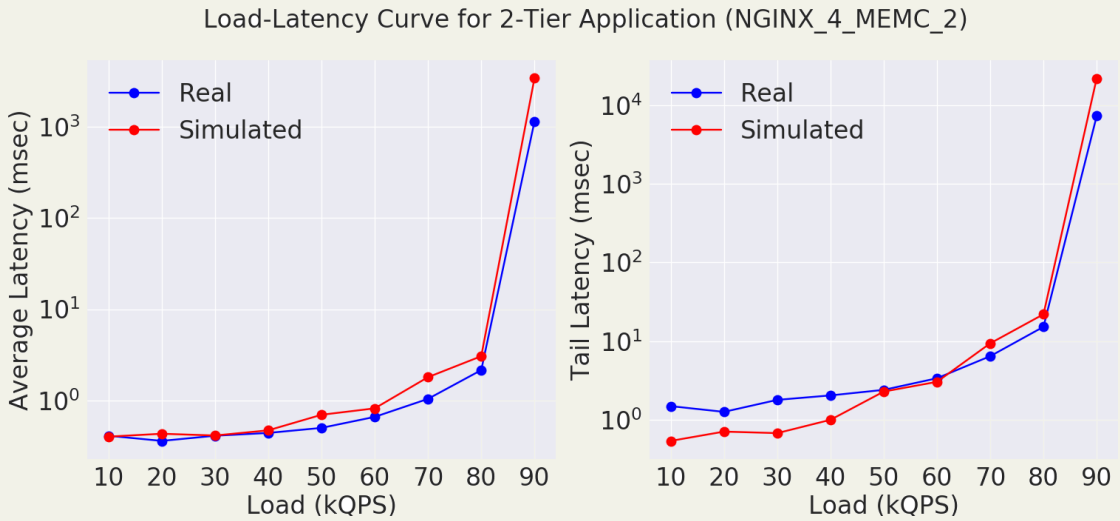
2/3-Tier Application

- 2-tier application



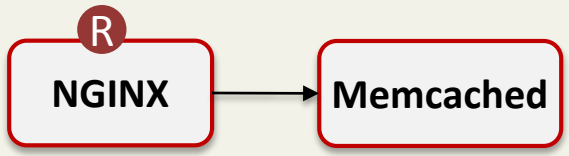
- 3-tier application



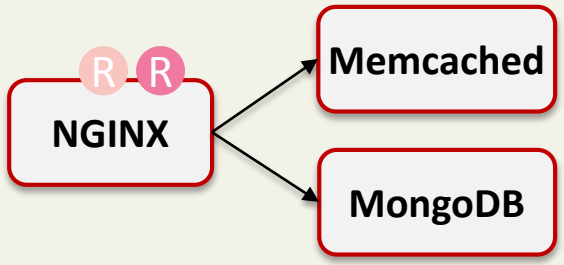


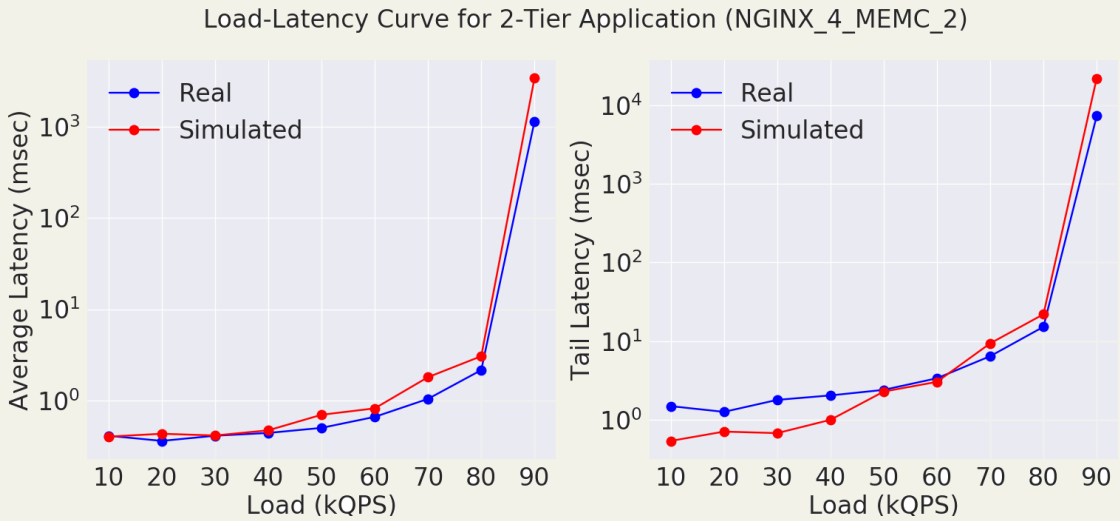
2/3-Tier Application

- 2-tier application



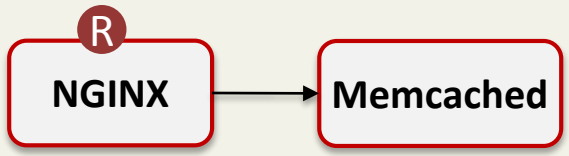
- 3-tier application



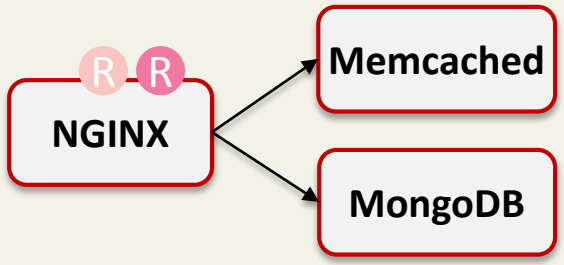


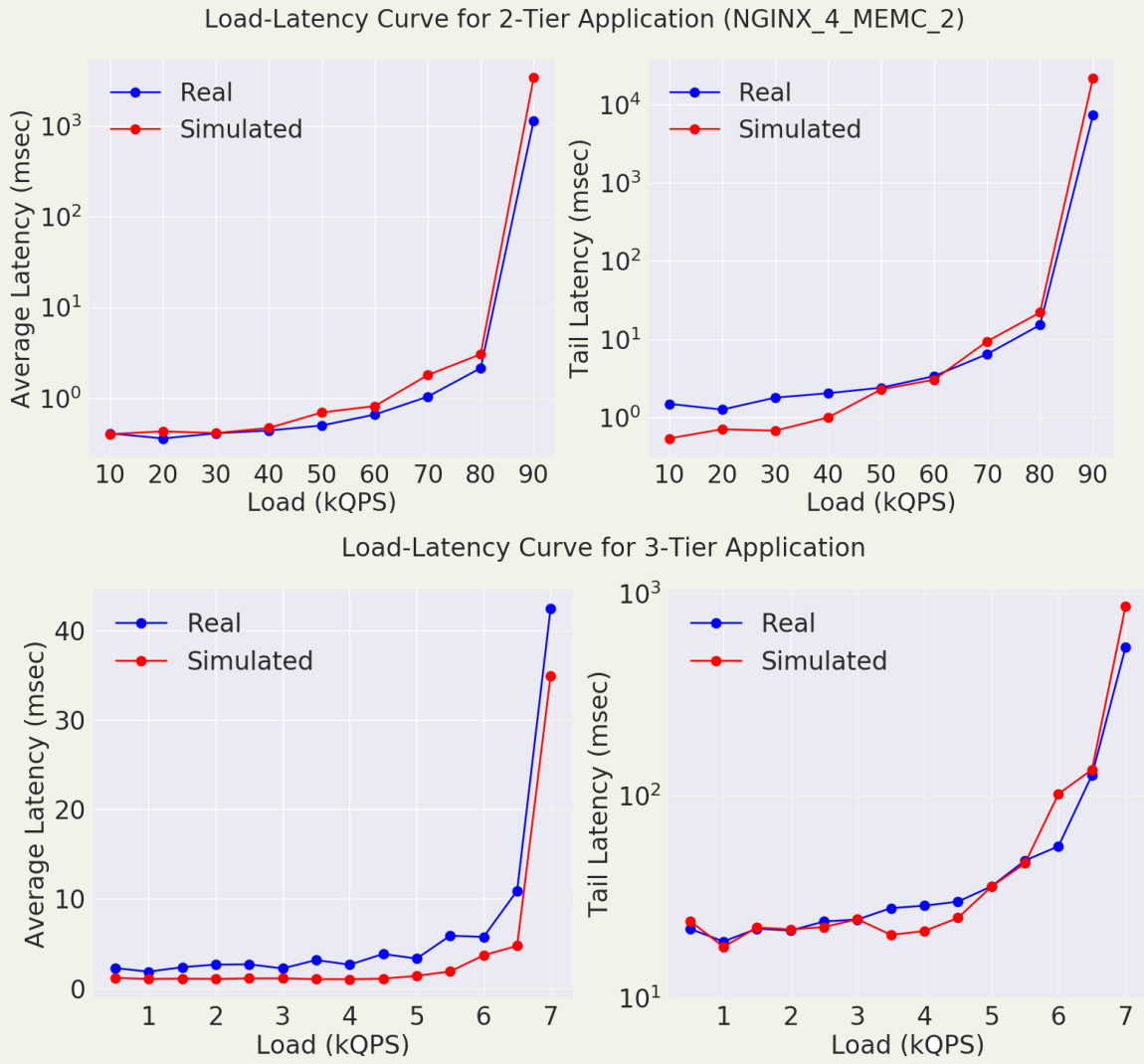
2/3-Tier Application

- 2-tier application

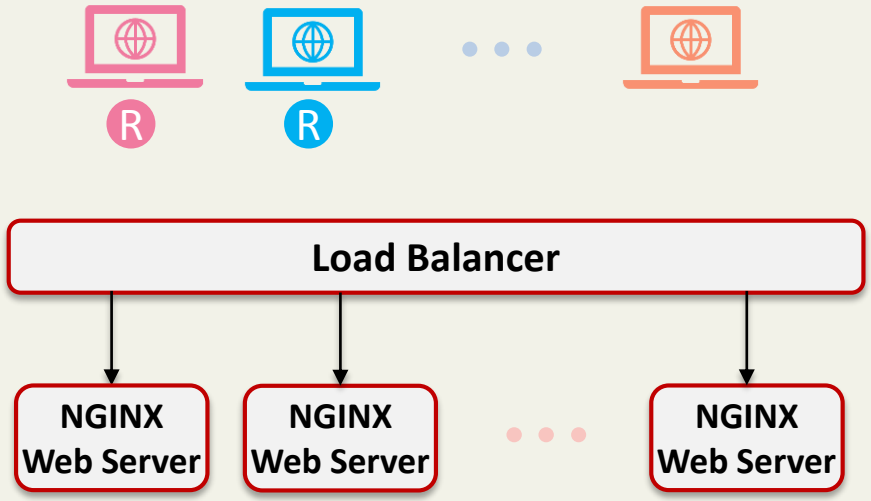


- 3-tier application

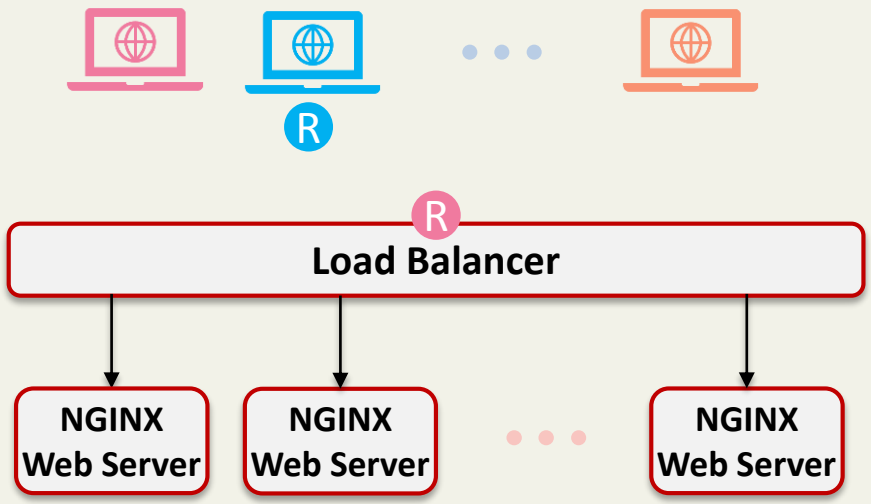




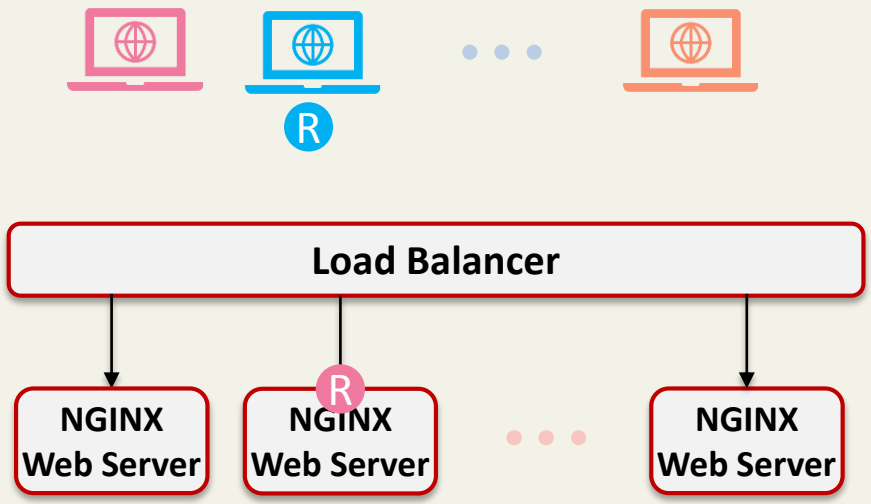
Load Balancing



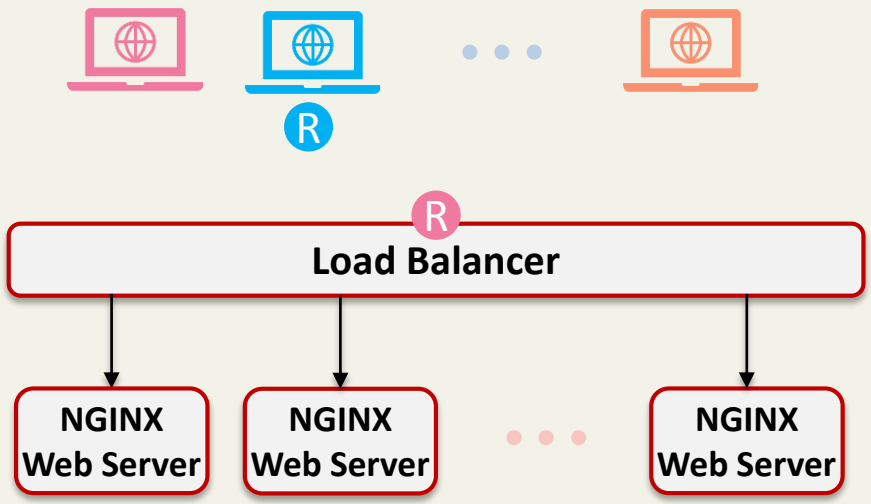
Load Balancing



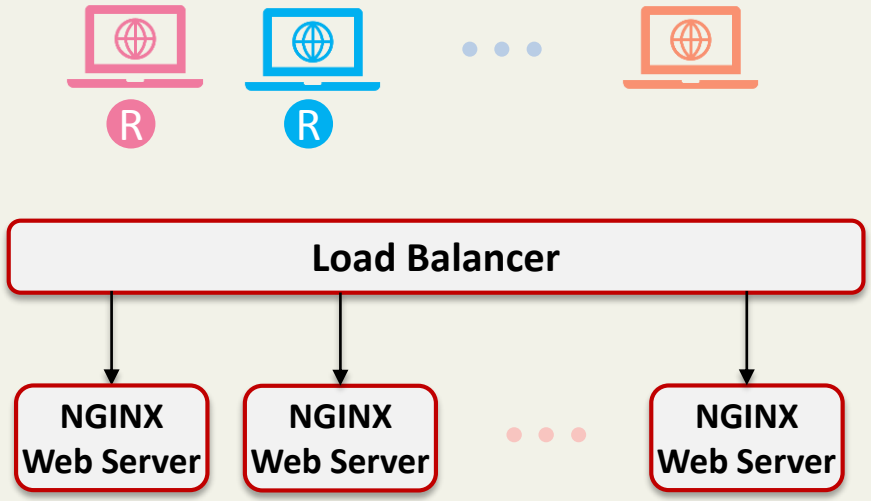
Load Balancing



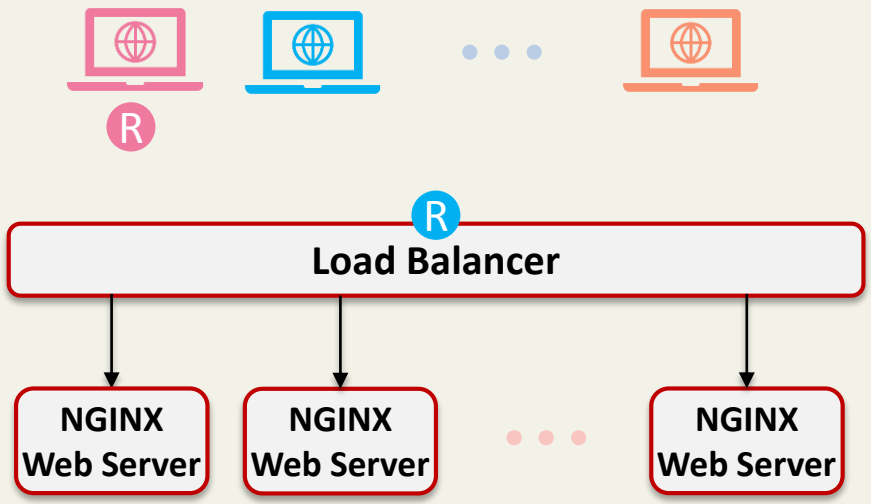
Load Balancing



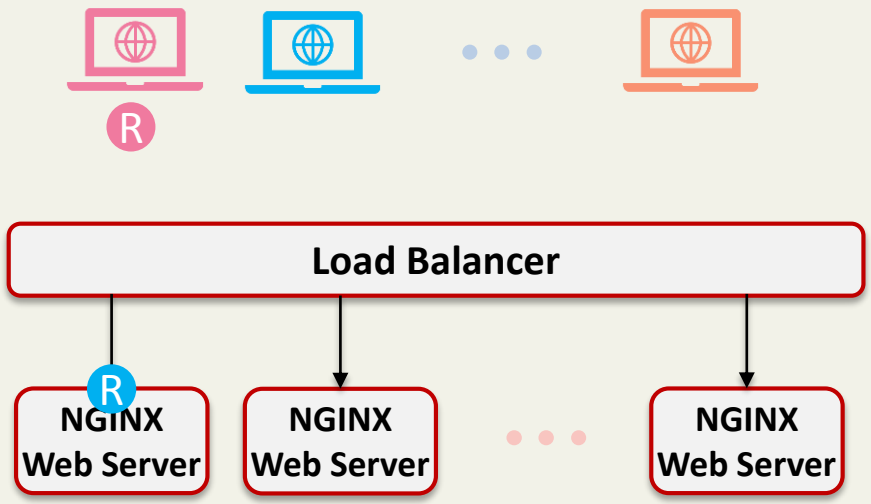
Load Balancing



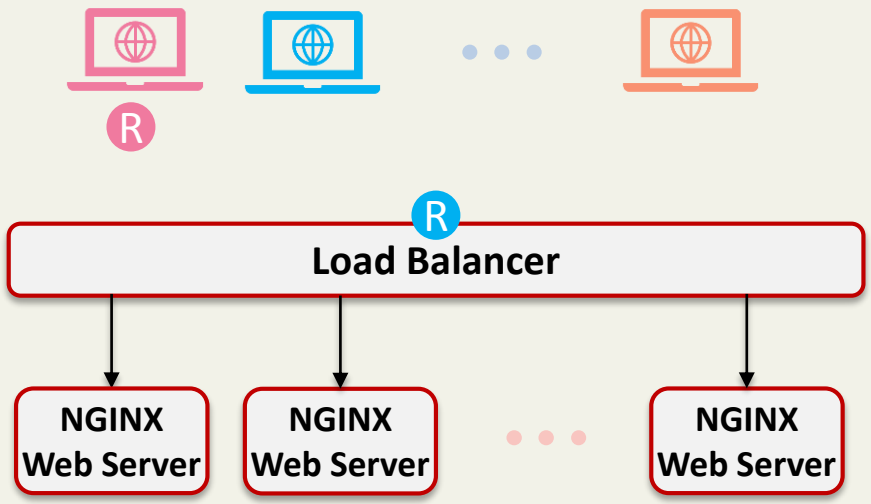
Load Balancing



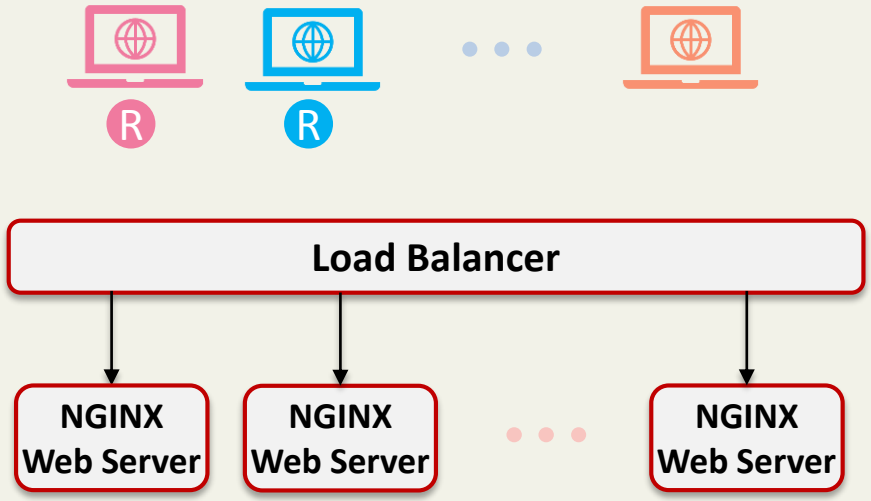
Load Balancing



Load Balancing

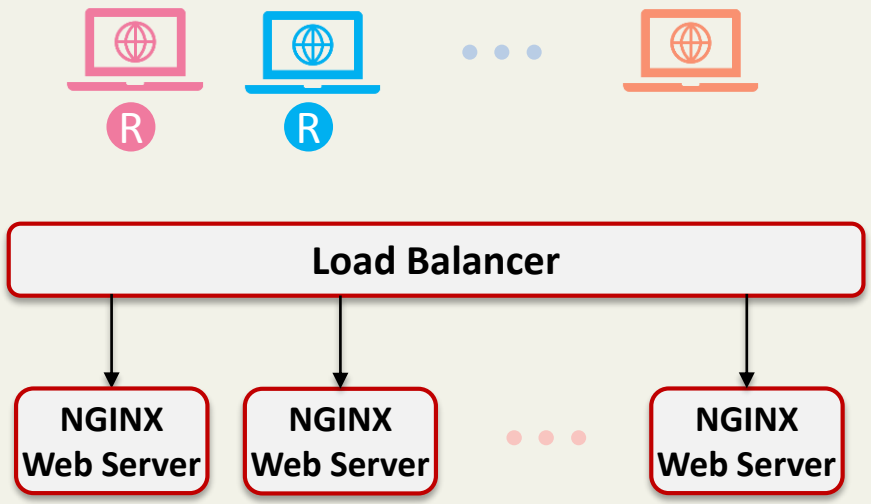


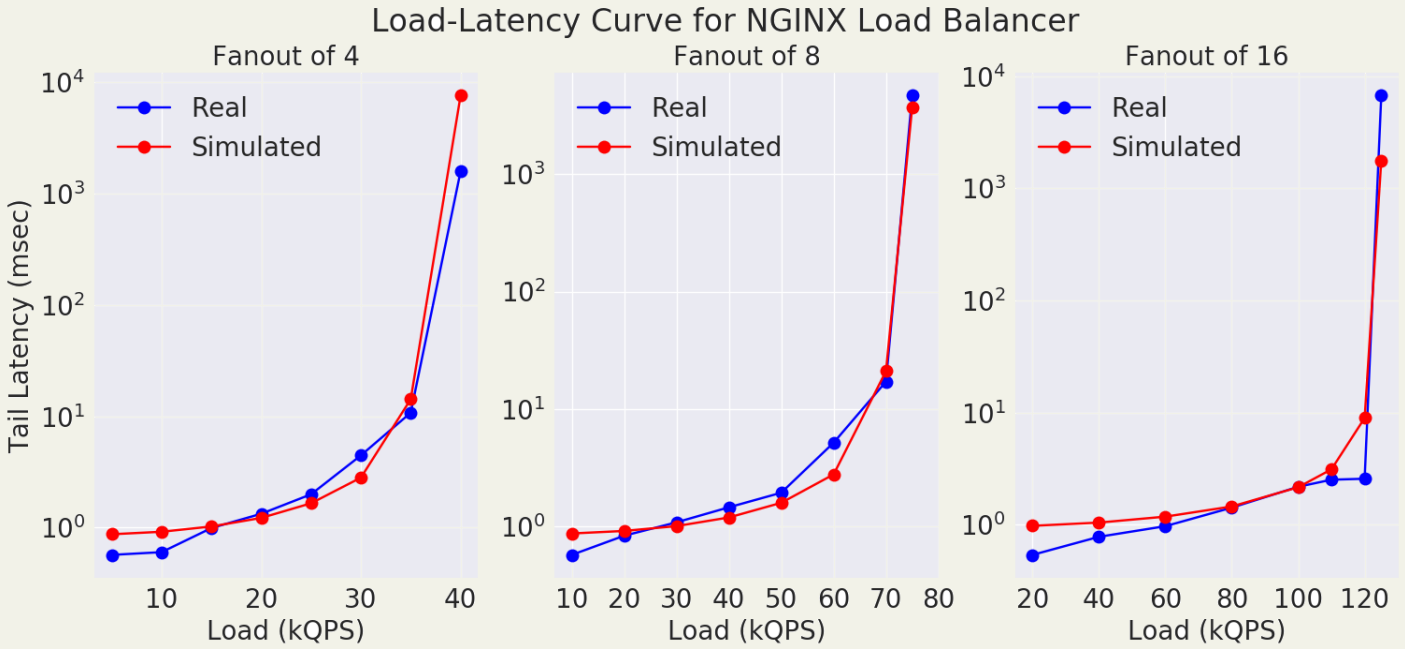
Load Balancing



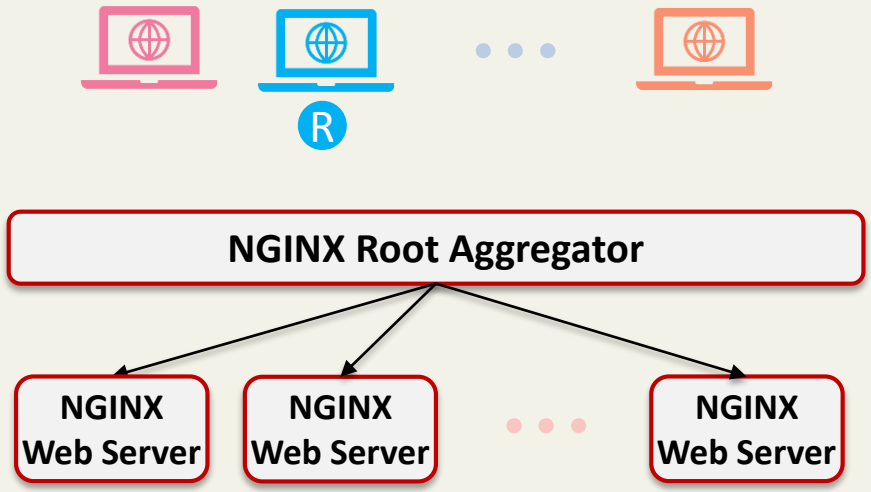
Load Balancing

- Linear throughput increase from cluster size of 4 to 8
- Sub-linear increase from cluster size of 8 to 16

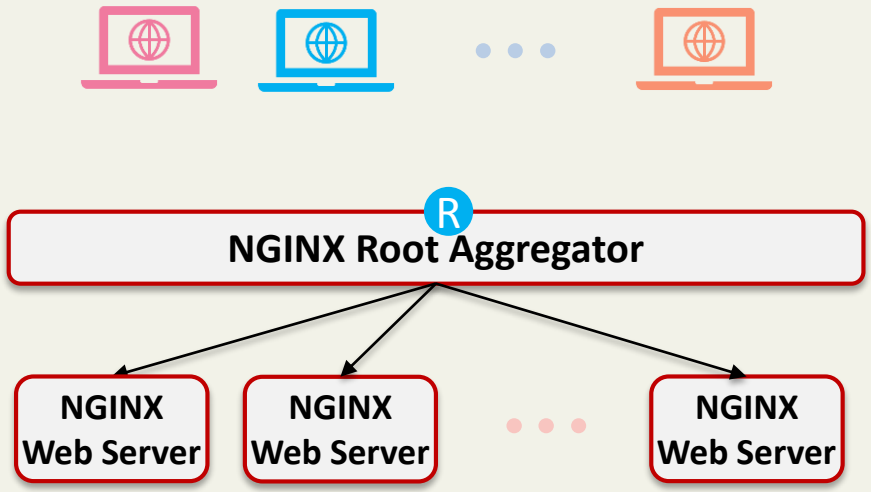




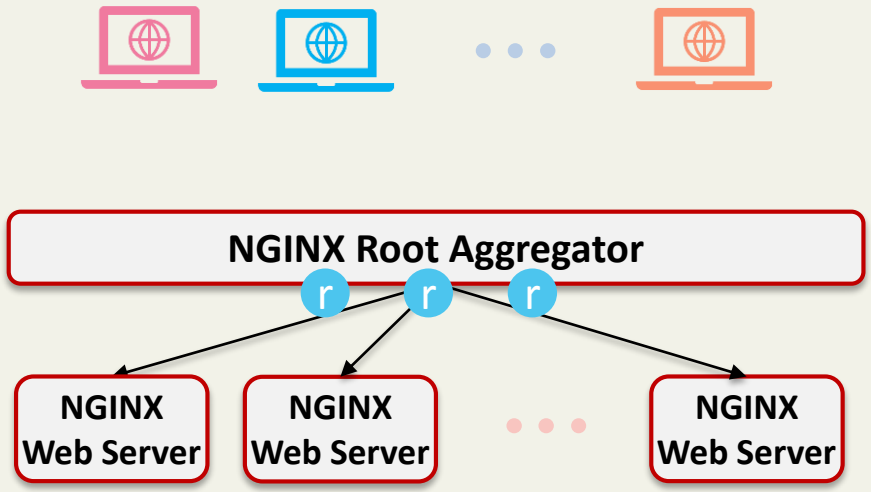
Request Fanout



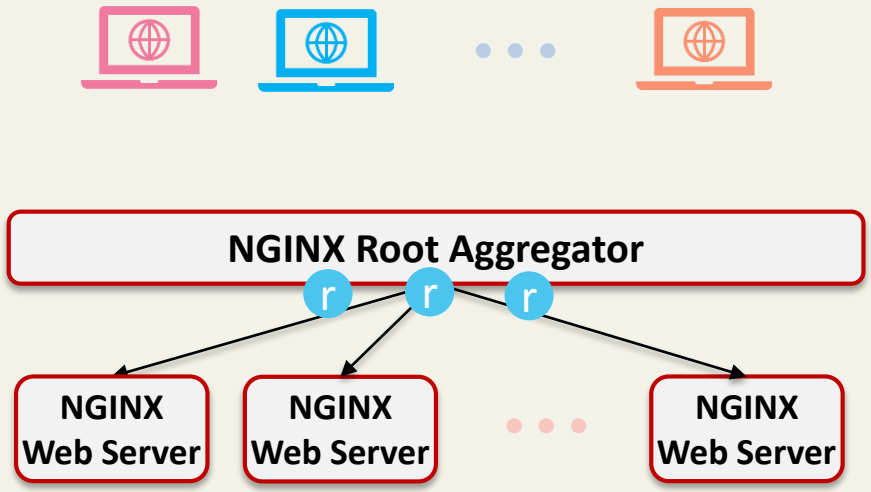
Request Fanout



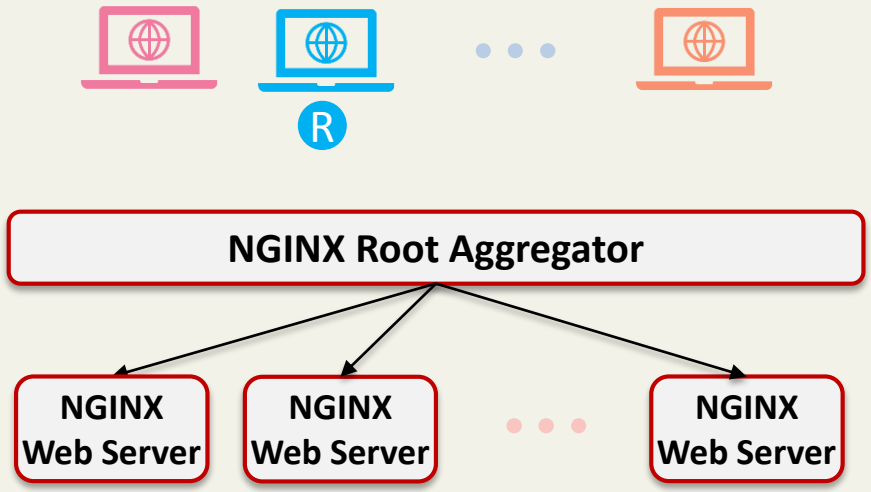
Request Fanout



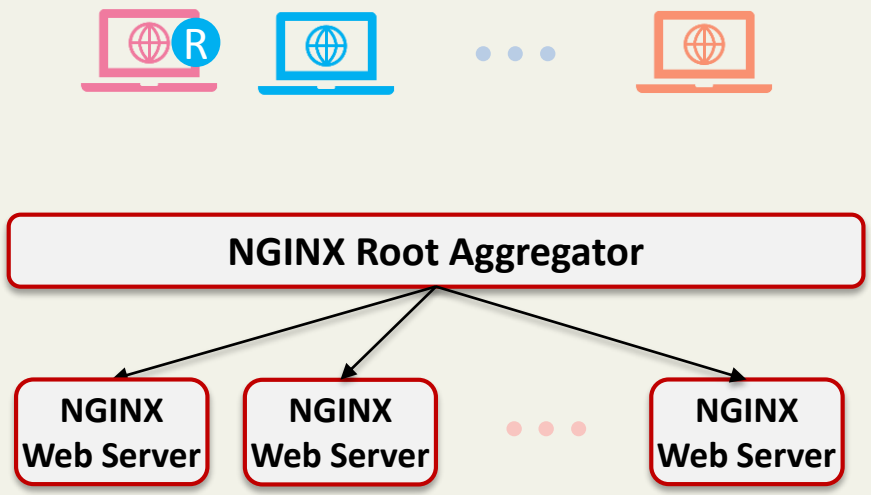
Request Fanout



Request Fanout

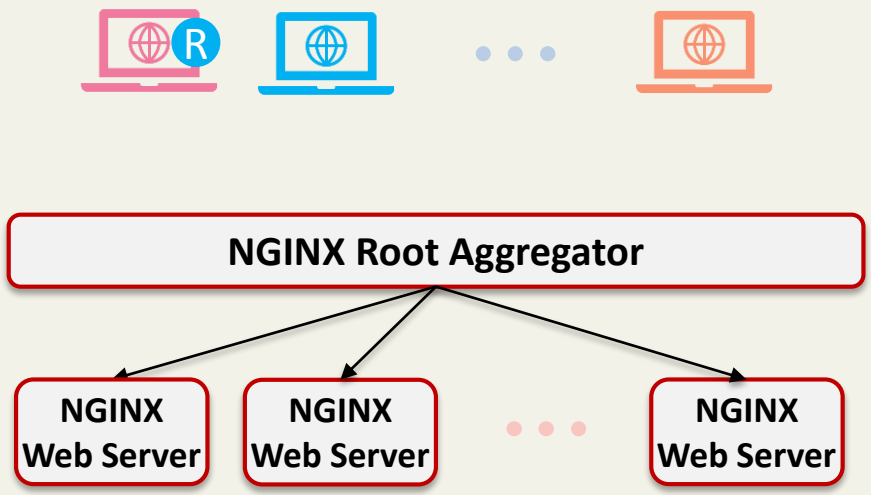


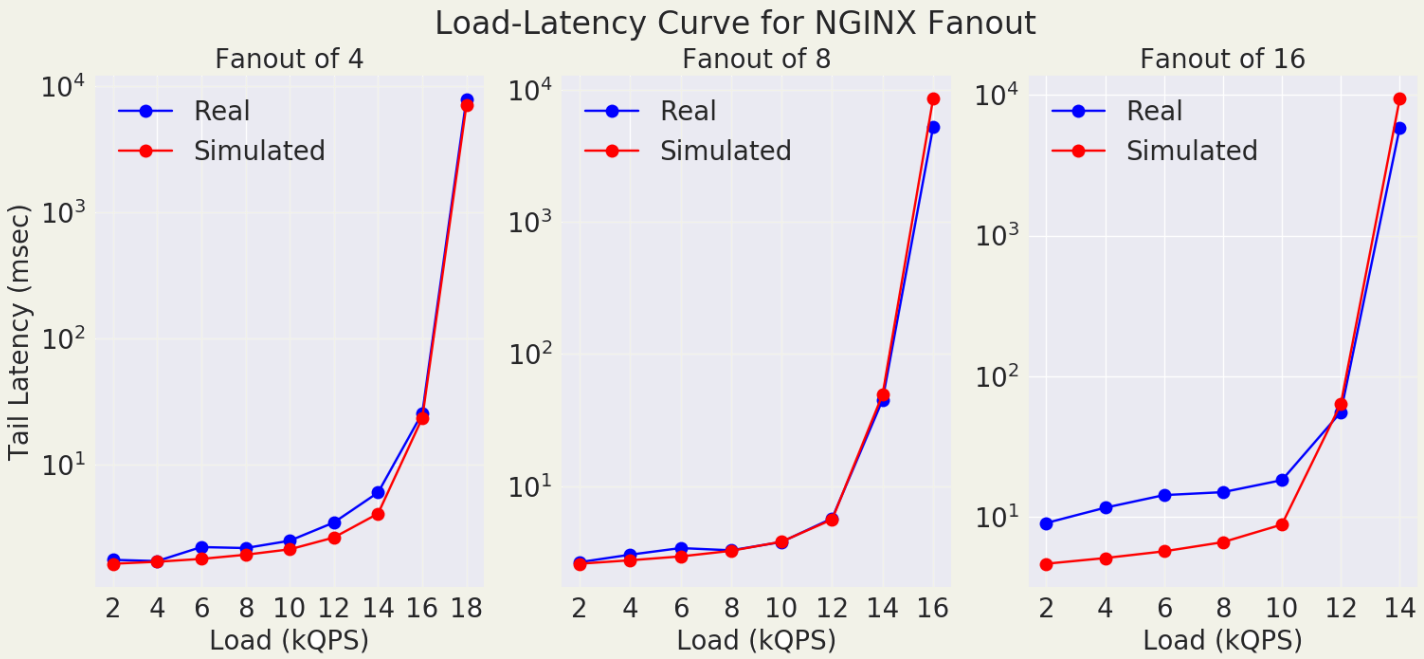
Request Fanout



Request Fanout

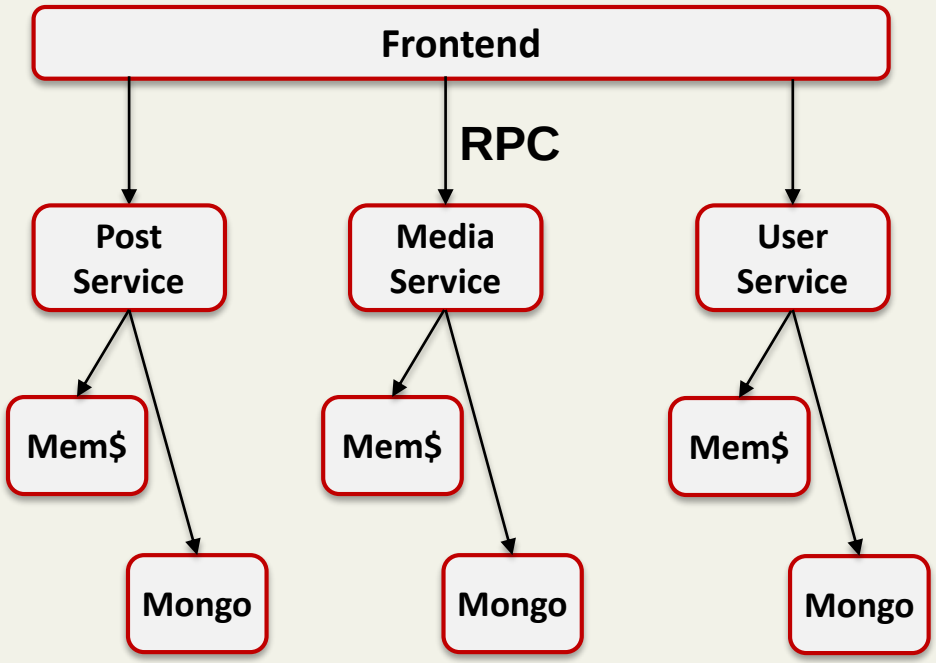
- **Throughput decreases with request fanout**





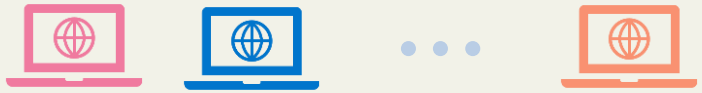
RPC & Large-Scale Microservices

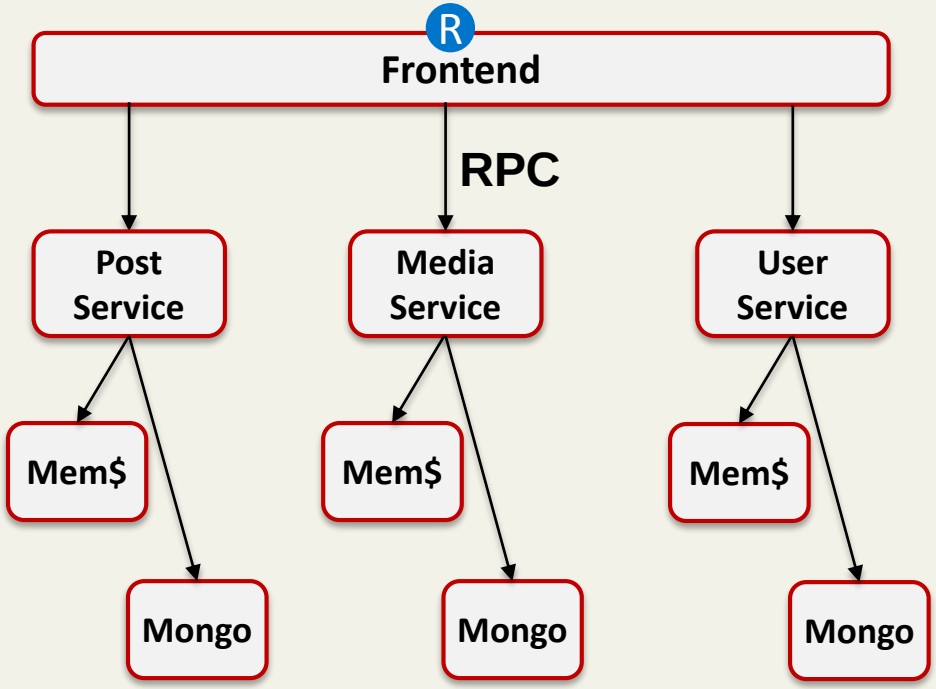
- Simplified social network



RPC & Large-Scale Microservices

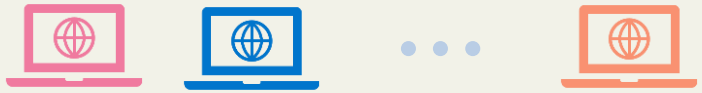
- Simplified social network

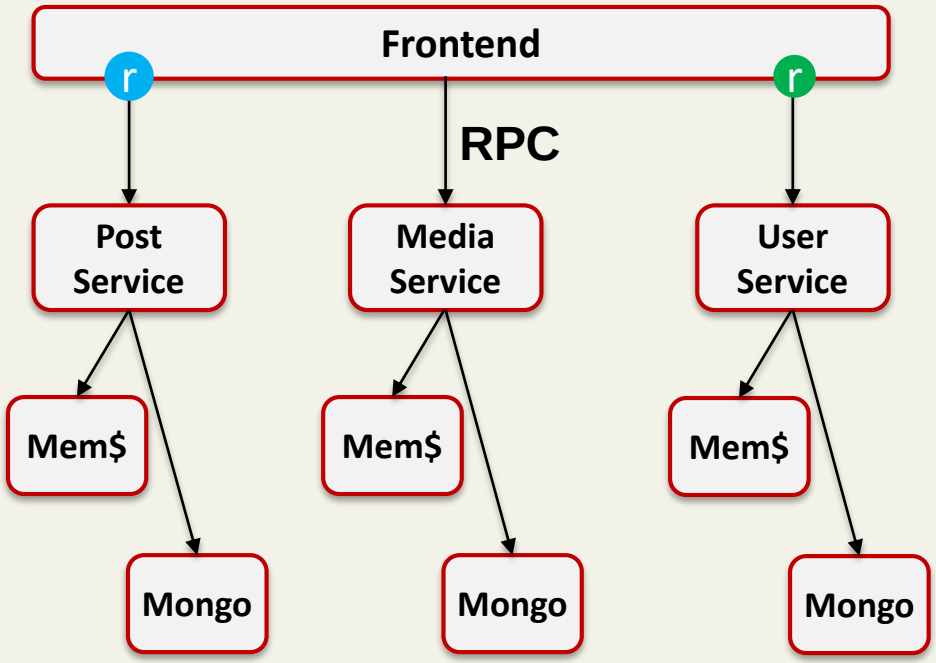




RPC & Large-Scale Microservices

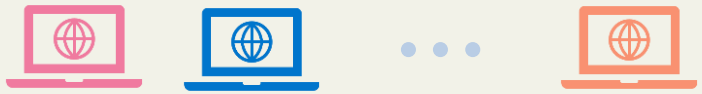
- Simplified social network

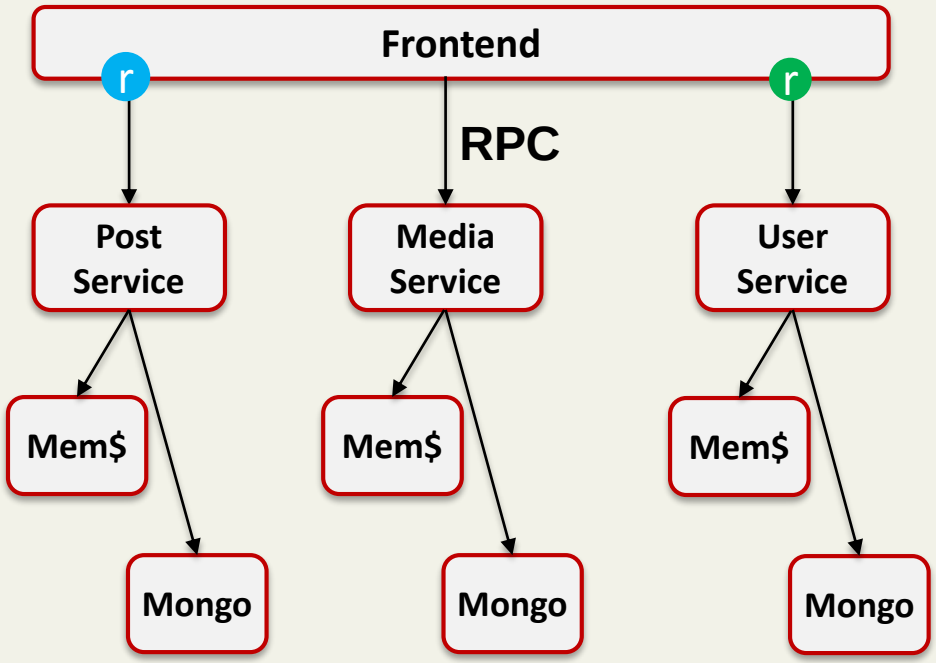




RPC & Large-Scale Microservices

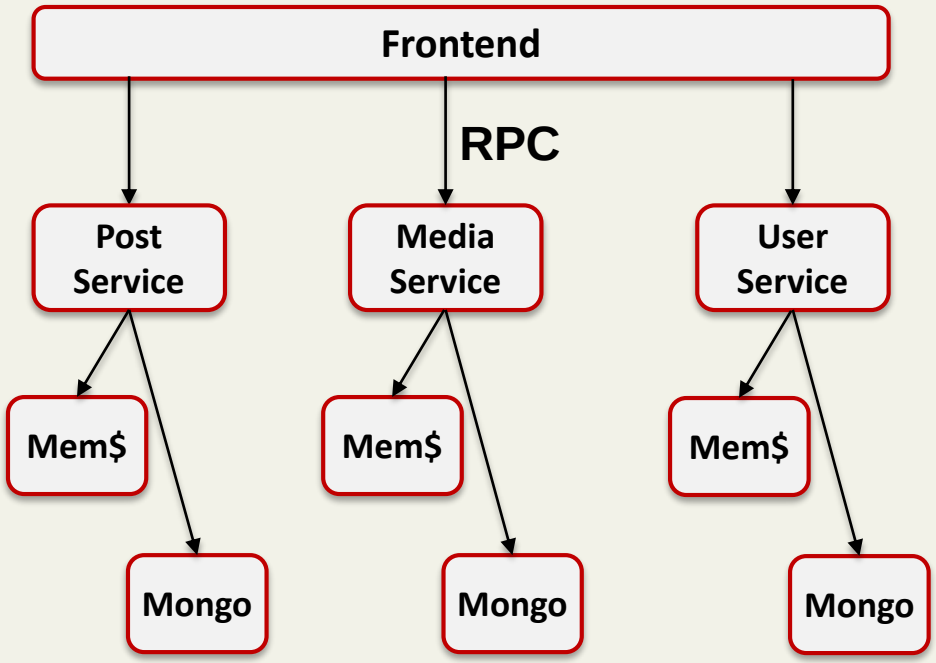
- Simplified social network





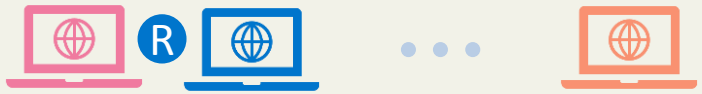
RPC & Large-Scale Microservices

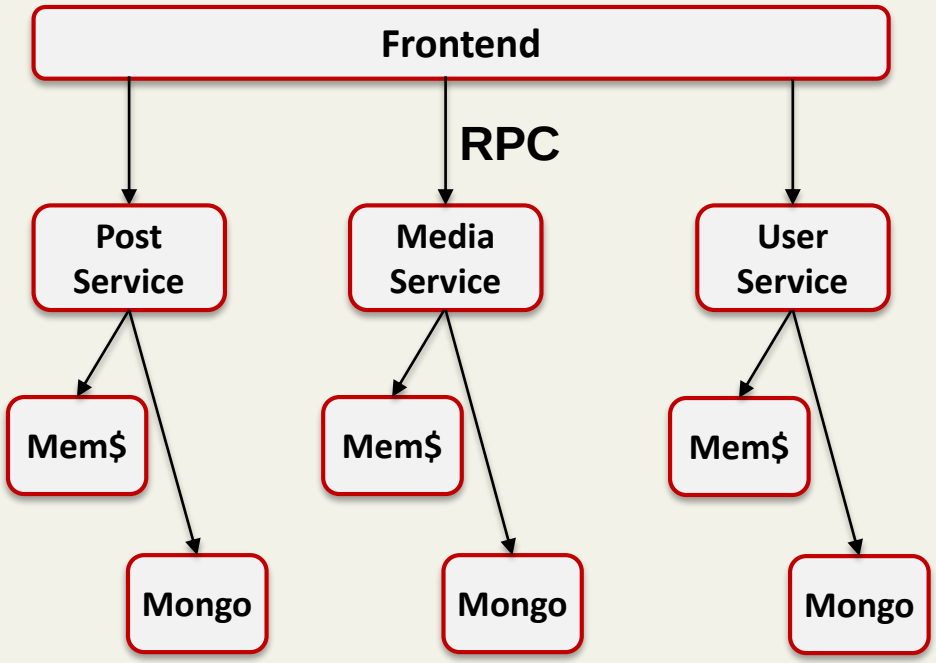
- Simplified social network



RPC & Large-Scale Microservices

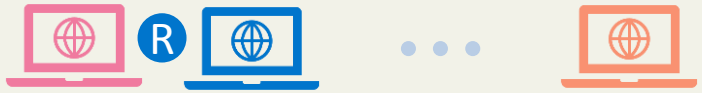
- Simplified social network

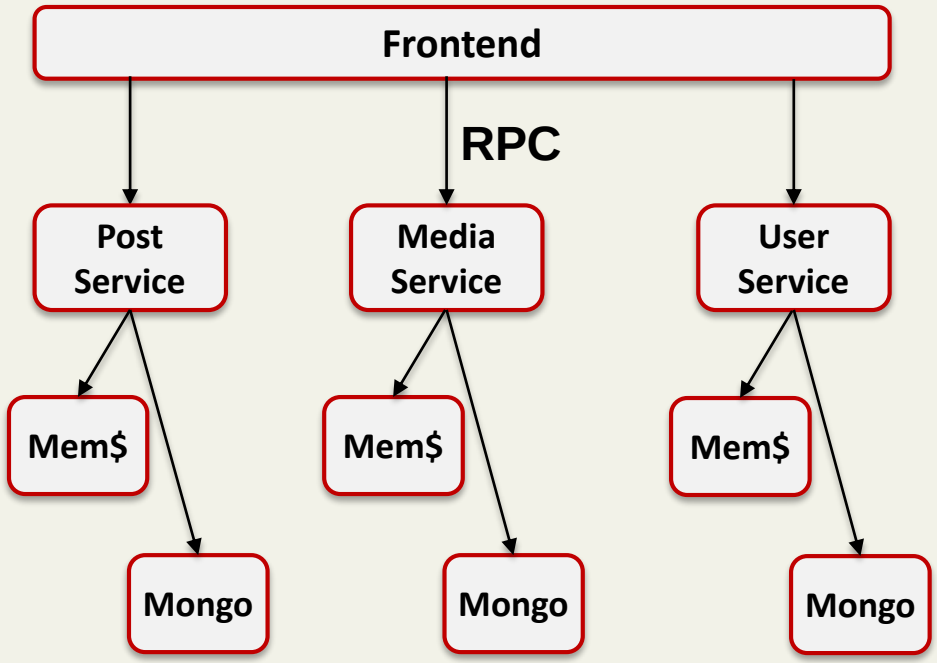


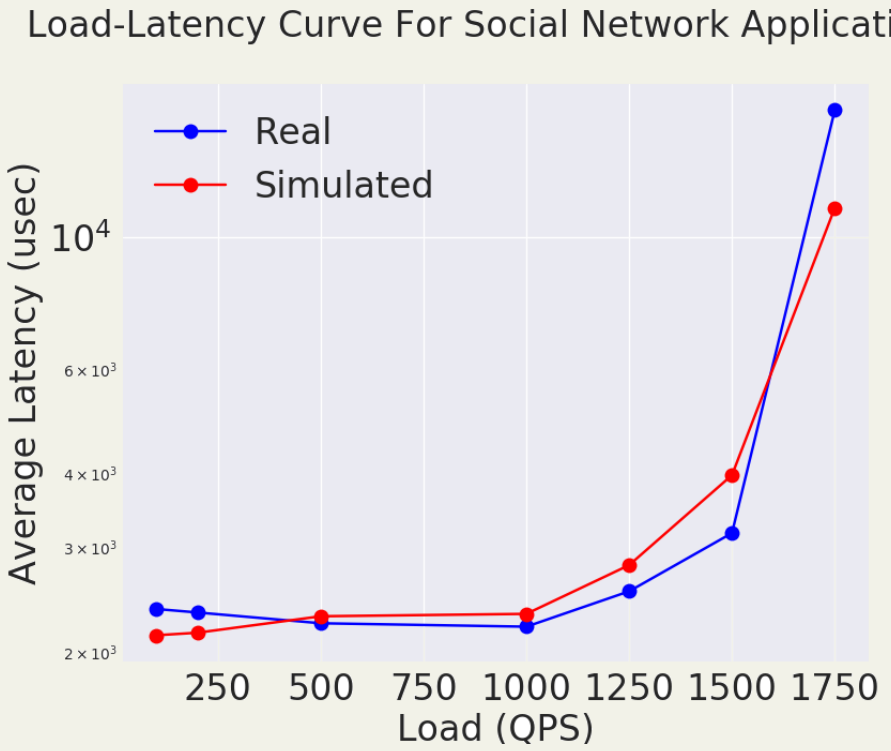


RPC & Large-Scale Microservices

- **Simplified social network**

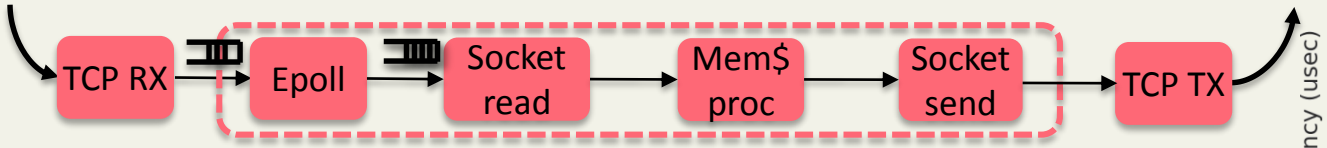


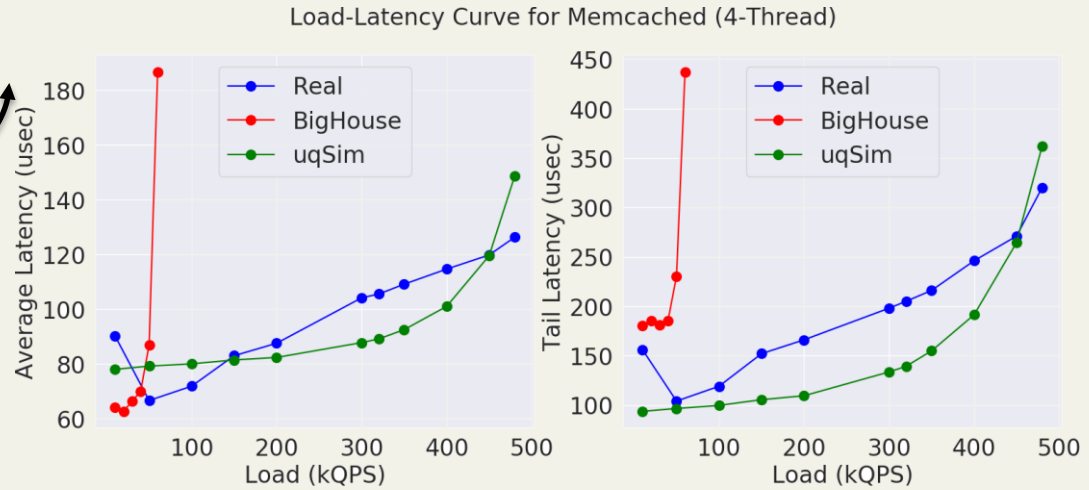




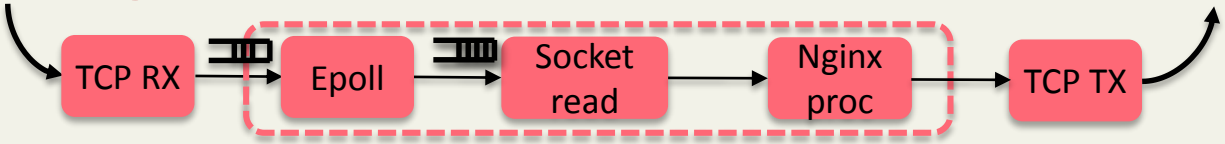
Comparison With BigHouse

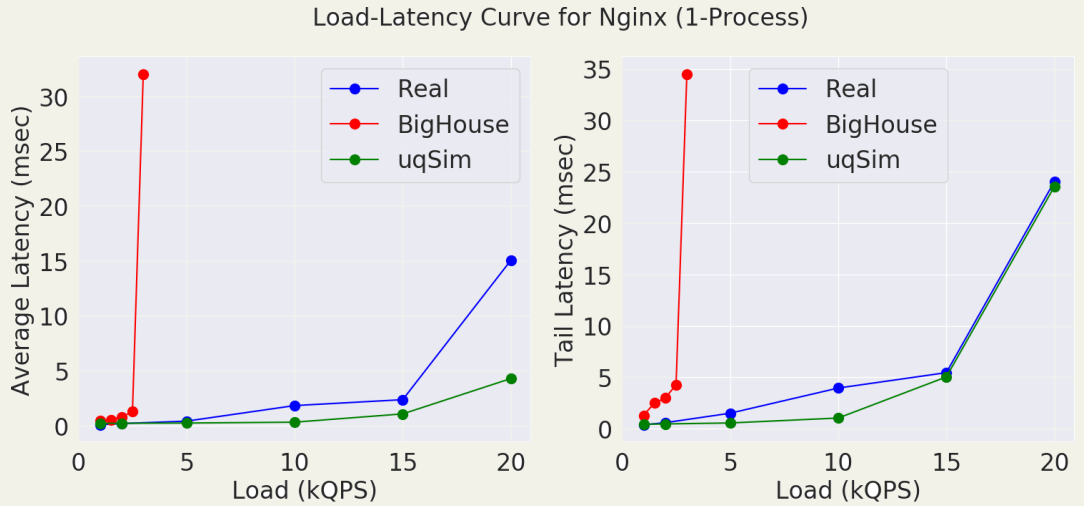
- **Memcached**





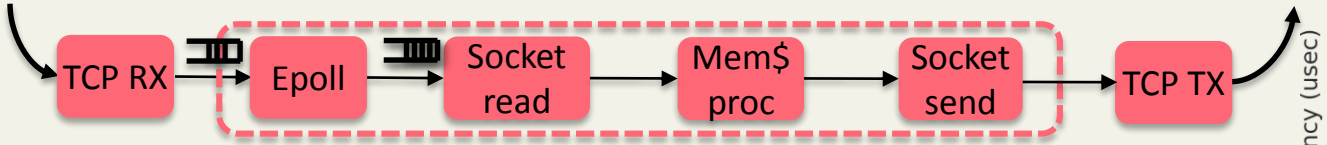
- **NGINX**



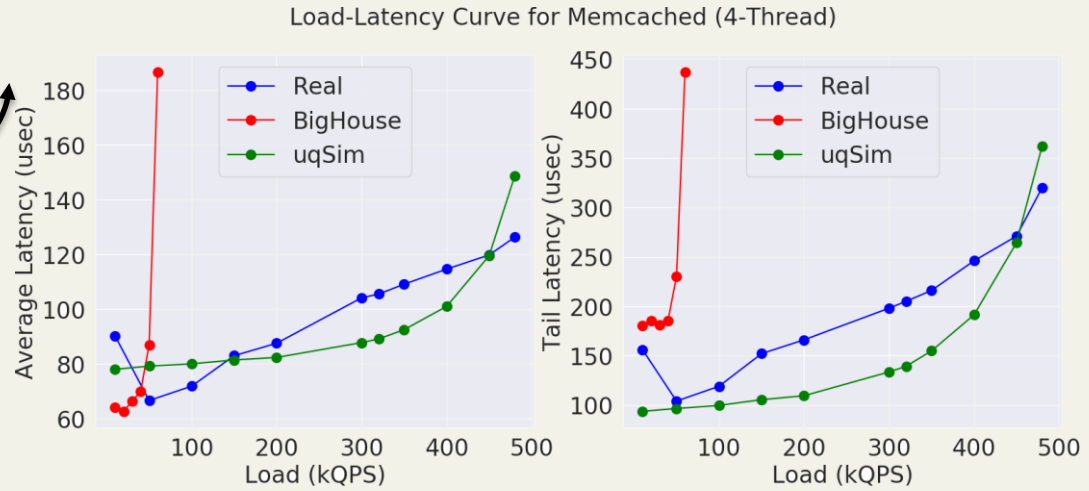


Comparison With BigHouse

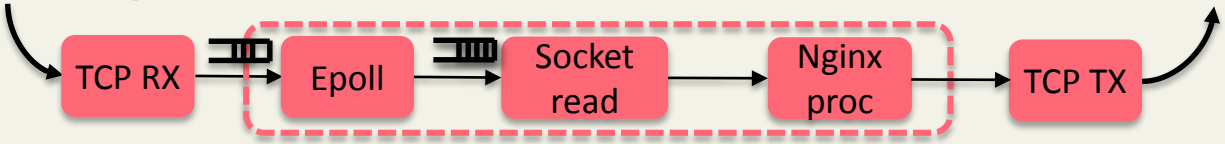
- **Memcached**

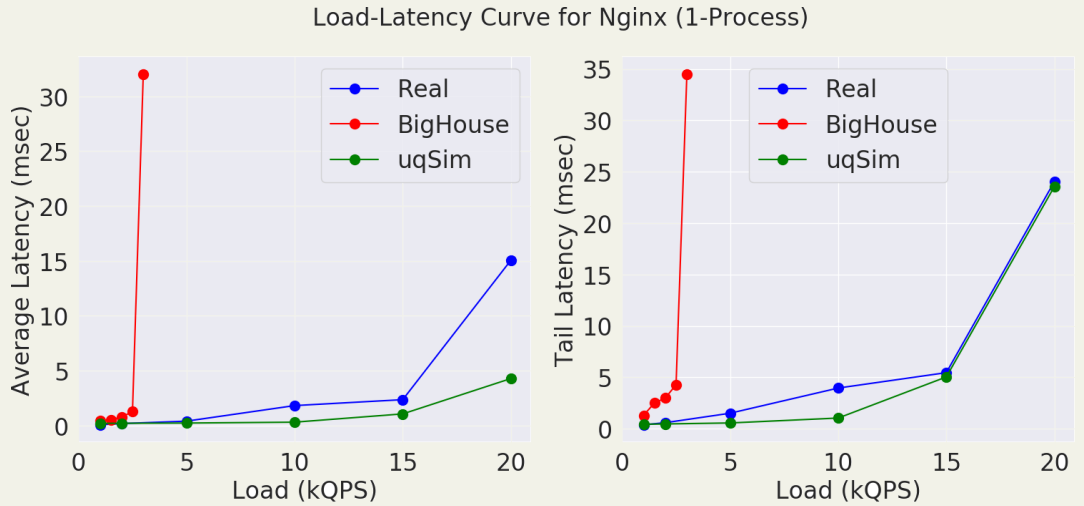


  - BigHouse saturates at 50kQPS
  - μqSim & real server saturates at 450kQPS



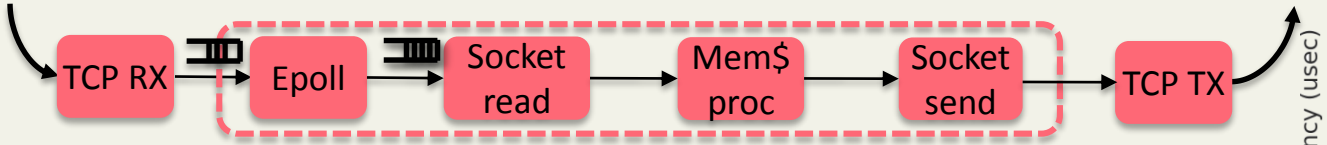
- **NGINX**



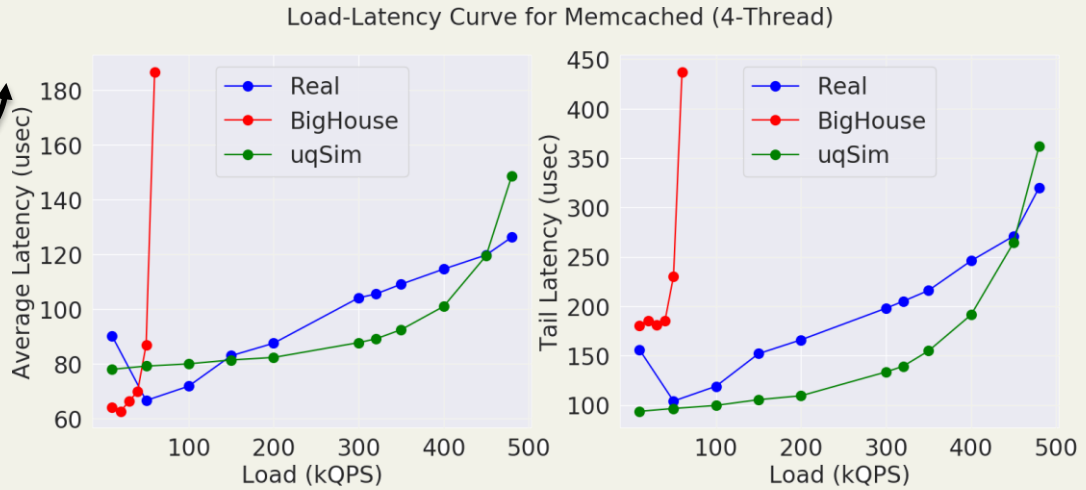


Comparison With BigHouse

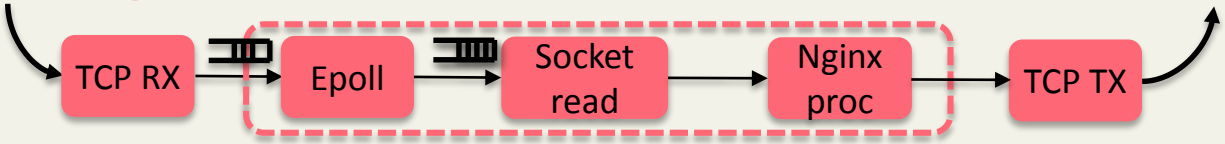
Memcached



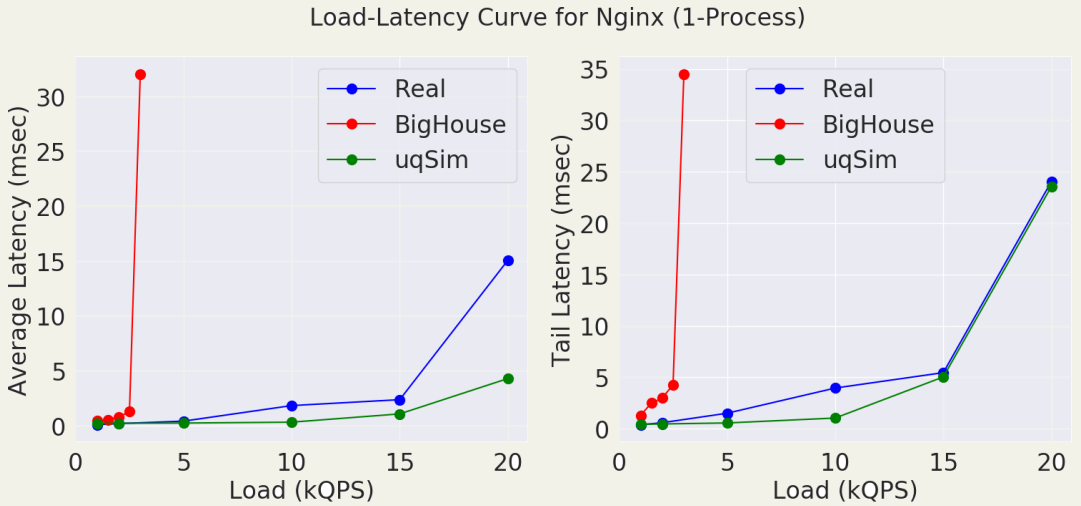
- BigHouse saturates at 50kQPS
- μqSim & real server saturates at 450kQPS



NGINX



- BigHouse saturates at 2.5kQPS
- μqSim & real server saturates at 15kQPS



Simulation Speed

- **Simulation speed relevant factors**
  - Simulated input load
  - Processing time distribution
  - Microservice model complexity & network complexity

Simulation Speed

- **Simulation speed relevant factors**
  - Simulated input load
  - Processing time distribution
  - Microservice model complexity & network complexity

- **Performance compared to real system**
  - Tail@Scale experiment: 26.5x slow down for cluster of 500 servers
  - 4-thread Memcached at 50kQPS: 2.5x speed up
  - 1-process Nginx at 10kQPS: 4x speed up

Simulation Speed

- **Simulation speed relevant factors**
  - Simulated input load
  - Processing time distribution
  - Microservice model complexity & network complexity
- **Performance compared to real system**
  - Tail@Scale experiment: 26.5x slow down for cluster of 500 servers
  - 4-thread Memcached at 50kQPS: 2.5x speed up
  - 1-process Nginx at 10kQPS: 4x speed up
- **Future work**
  - Parallelizing simulation
  - Each thread simulates a partition of the network

Conclusion

- **Microservices introduce new system challenges**
  - Need scalable simulation techniques to study large scale effects
- **μqSim: validated microservice simulator**
  - Modeling the internal queueing structure of individual microservices
  - Modeling dataflow behavior across the microservices
  - Validated against simple & complex microservices and accurately capturing throughput/latency
  - Planning to open source @ microservices.ece.cornell.edu

Thanks & Questions

- **Microservices introduce new system challenges**
  - Need scalable simulation techniques to study large scale effects

- **μqSim: validated microservice simulator**
  - Modeling the internal queueing structure of individual microservices
  - Modeling dataflow behavior across the microservices
  - Validated against simple & complex microservices and accurately capturing throughput/latency
  - Planning to open source @ microservices.ece.cornell.edu

Speed Comparison With BigHouse (Backup)

μqSim simulated real time	5s	20s	60s	
μqSim simulation time	12.8 s	51.2s	154.1 s	
BigHouse warmup samples	**5 x50k**	**20 x50k**	**60 x50k**	**5000**
BigHouse simulation time	8.2s	28.7s	154s	1.2s

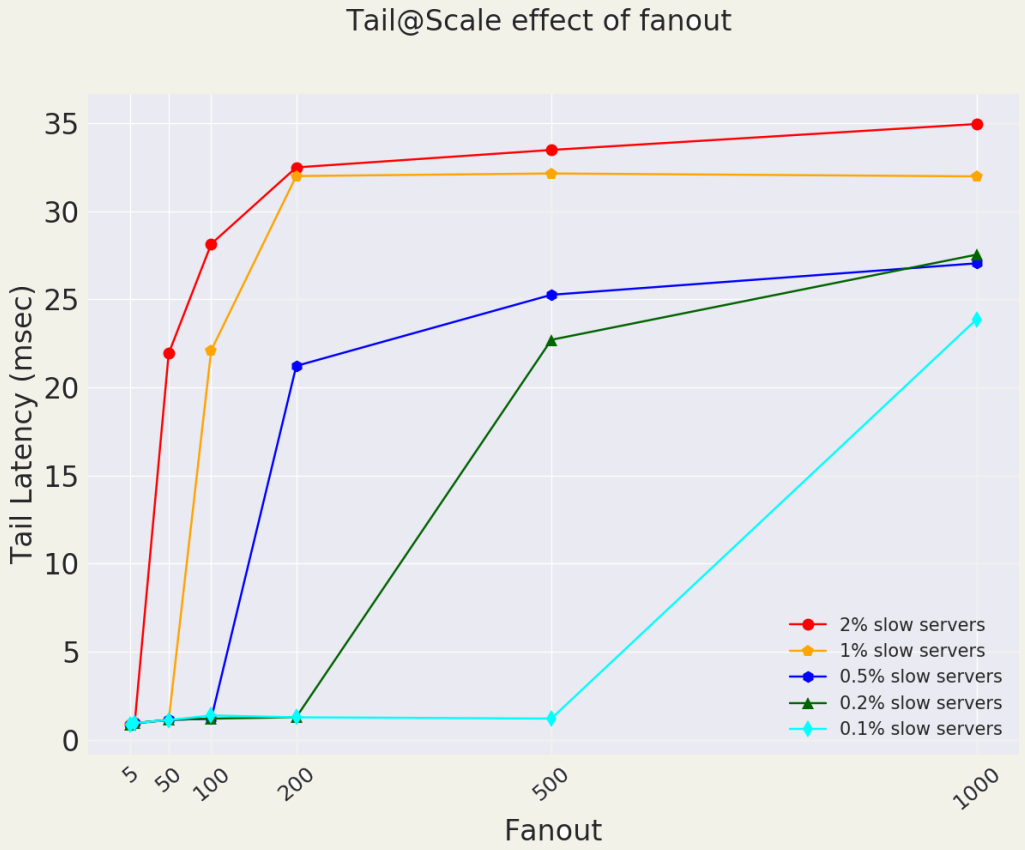
4-thread Memcached at 50k

μqSim simulated real time	5s	20s	60s	
μqSim simulation time	0.7s	2.7s	8.1s	
BigHouse warmup samples	**5 x2k**	**20 x2k**	**60 x2k**	**5000**
BigHouse simulation time	9s	23s	64s	7.1s

1-process NGINX at 2k

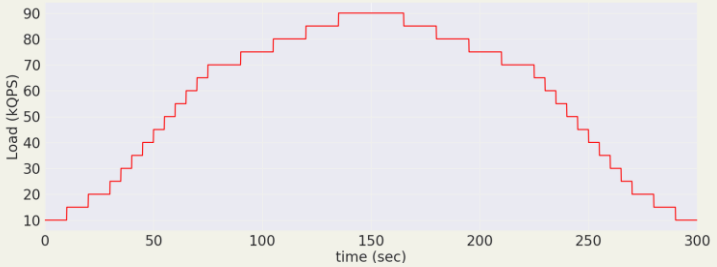
Use Cases – Tail@Scale (Backup)

- For simple single-tier single stage
- Different Cluster Sizes
- Different Fraction of Slow Servers



Use Cases – Power Management (Backup)

- **End-to-End QoS Target**
  - 5ms
- **Input Load**
  - Diurnal pattern



- **Results**