A Study of Long-Tail Latency in n-Tier Systems: RPC vs. Asynchronous Invocations

Qingyang Wang, Louisiana State University

Long-Tail Latency Problem

Web-facing applications encounter large response time fluctuations at moderate utilization (e.g., 50%)

Causes:

- Strong inter-tier dependency between thread-based servers through RPC calls in the long invocation chain
- Millibottlenecks occur in all system layers at moderate system utilization
- Millibottlenecks plus inter-tier dependency leads to Cross-Tier-Queue-Overflow, which in turn cause dropped packets and TCP retransmissions.

Solution:

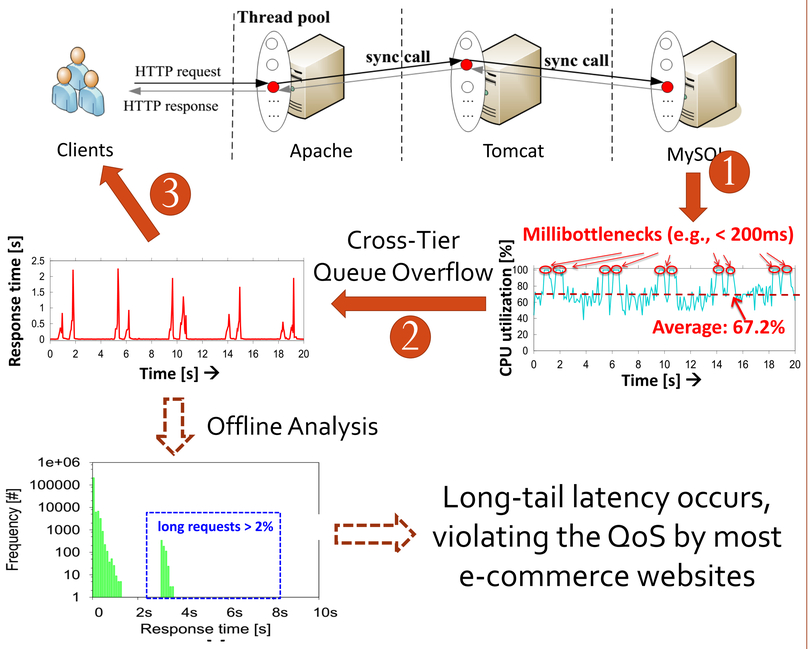
- Asynchronous invocation between consecutive tiers in the long invocation chain
- Break the strong inter-tier dependency and Cross-Tier-Queue-Overflow

Benefits:

- Achieve predicable performance of n-tier web applications at moderate to high utilization
- Increase resource efficiency and save power of cloud data centers.

Millibottlenecks ⇒ Long Tail Latency

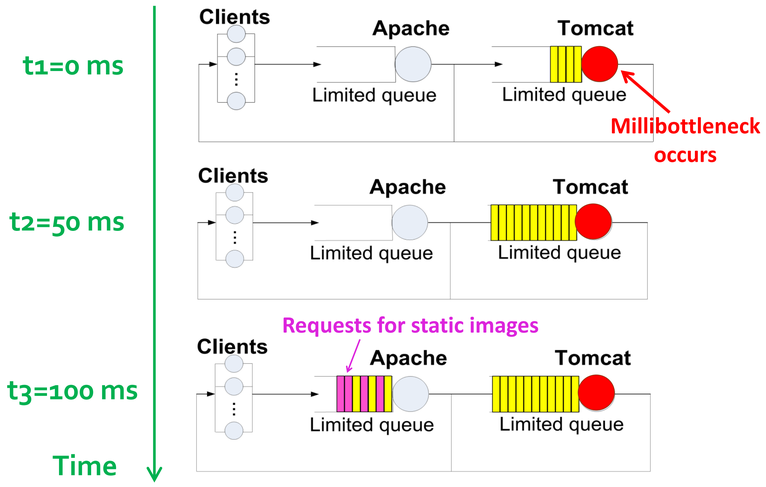
A 3-tier system with thread-based servers



Long-tail latency occurs, violating the QoS by most e-commerce websites

Push-back: Cross-tier-Queue-Overflow

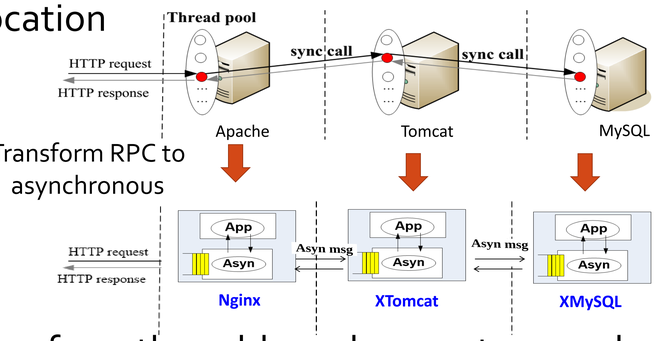
An illustration example:



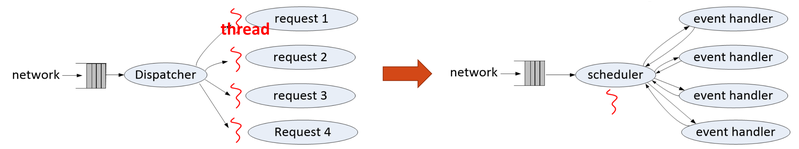
A millibottleneck in downstream Tomcat ⇒ long queue in upstream Apache

Solution: Asynchronous Invocation

Break push-back wave by asynchronous invocation



➢ Transform thread-based server to asynchronous



Thread-based Server design (e.g., Apache)

Asynchronous Server design (e.g., Apache)

➢ Transform sequential app to event driven app

```
[01] function doGet(request1) {
[02]     ... pre-processing request1 ...
[03]     ... form query1 ...
[04]     result1=SyncDBQuery1(query1);
[05]     ... think about result1 ...
[06]     ... form query2 ...
[07]     result2=SyncDBQuery2(query2);
[08]     ... post-processing result2 ...
[09]     ... form response ...
[10]     return response;
[11] }
```

(a) A simple synchronous Java Servlet

```
[01] function doGet(request1) {
[02]     ... pre-processing request1 ...
[03]     ... form query1 ...
[03]     AsynDBQuery1(query1, eventHandler1);
[04] }
[05] function eventHandler1(result1) {
[06]     ...think about result1 ...
[07]     ...form query2 ...
[08]     AsynDBQuery2(query2, eventHandler2);
[09] }
[10] function eventHandler2(result2) {
[11]     ...post-processing result2...
[12]     ...form response ...
[13]     return response;
[14] }
```

(b) An asynchronous event-driven Java servlet

Results & Future Work

Results:

- Long-tail latency remains absent at system utilization levels as high as 83%, despite the same millibottlenecks. --Wang et al. ICDCS '17

Future works:

- Design profiling tools for asynchronous n-tier systems.
- Develop tools to facilitate the transforming RPC code to asynchronous code
- Run large-scale cloud experiments for validation