RPC Performance

ECE 454 / 751: Distributed Computing

Instructor: Dr. Wojciech Golab

wgolab@uwaterloo.ca

Learning objectives

- understanding performance-limiting factors
- develop formulas for estimating throughput and latency

Assumptions and notation

Assumption 1: The workload is CPU-intensive.

Assumption 2: The software architecture has a layer of clients and a layer of servers.

Assumption 3: Clients use synchronous RPCs.

- Let *D* denote the time required to process one request in one thread at the server.
- Let *L* denote the one-way network latency between the client and server layer. (Round trip time is 2*L*.)
- Let *T* denote the total number of threads at the client layer.
- Let *C* denote the total number of cores allocated to the server layer.

Factors limiting throughput

Factor 1: server CPU resources

Each server thread can process at most $1/D$ requests per second if it is kept busy 100% of the time. Assuming C cores, the entire server layer is limited to at most C/D requests per second irrespective of how the client layer is implemented (e.g., synchronous vs. asynchronous RPCs). Processing overheads, such as marshalling and unmarshalling, reduce performance even further.

Thus, we obtain the following upper bound based on the server layer:

$$\text{Throughput} \leq C/D$$

Factors limiting throughput

Factor 2: client parallelism

The client threads spend most of their time waiting for RPCs to complete, and so we are more concerned with the number of client threads than with the number of cores allocated to the client layer. A single client thread must wait at least $(2L + D)$ time for one synchronous RPC, and so throughput per thread is limited by the reciprocal of this quantity. With T client threads, we expect at most T times higher throughput.

Thus, we obtain the following upper bound based on the client layer:

$$\text{Throughput} \le T/(2L + D)$$

Factors limiting throughput

Combining the formulas from the last two slides, we obtain the following upper bound:

$$\text{Throughput} \leq \min[\, C/D,\;\; T/(2L + D)\,]$$

We say that *performance is limited by the server* if $C/D < T/(2\text{L} + D)$, otherwise we say that *performance is limited by the client*.

It follows from the above formula that as long as $L > 0$, we need more client threads (T) than server cores (C) to keep the server CPU resources fully utilized.

Calculating latency

- Each request has a latency of at least $2L + D$ from the client's perspective because it has to traverse the network twice and perform computation for D time units at a server.
- Latency can be higher if there are queuing delays, such as when throughput is limited by the server layer and not the client layer.
- Assuming for simplicity that end-to-end latency is a constant rather than a random variable, and that client threads share the load equally, we obtain the following estimate:

$$\begin{aligned}\text{Latency} &= 1 / (\text{per-thread throughput}) \\ &= 1 / (\text{Throughput} / T) \\ &= T / \text{Throughput}\end{aligned}$$

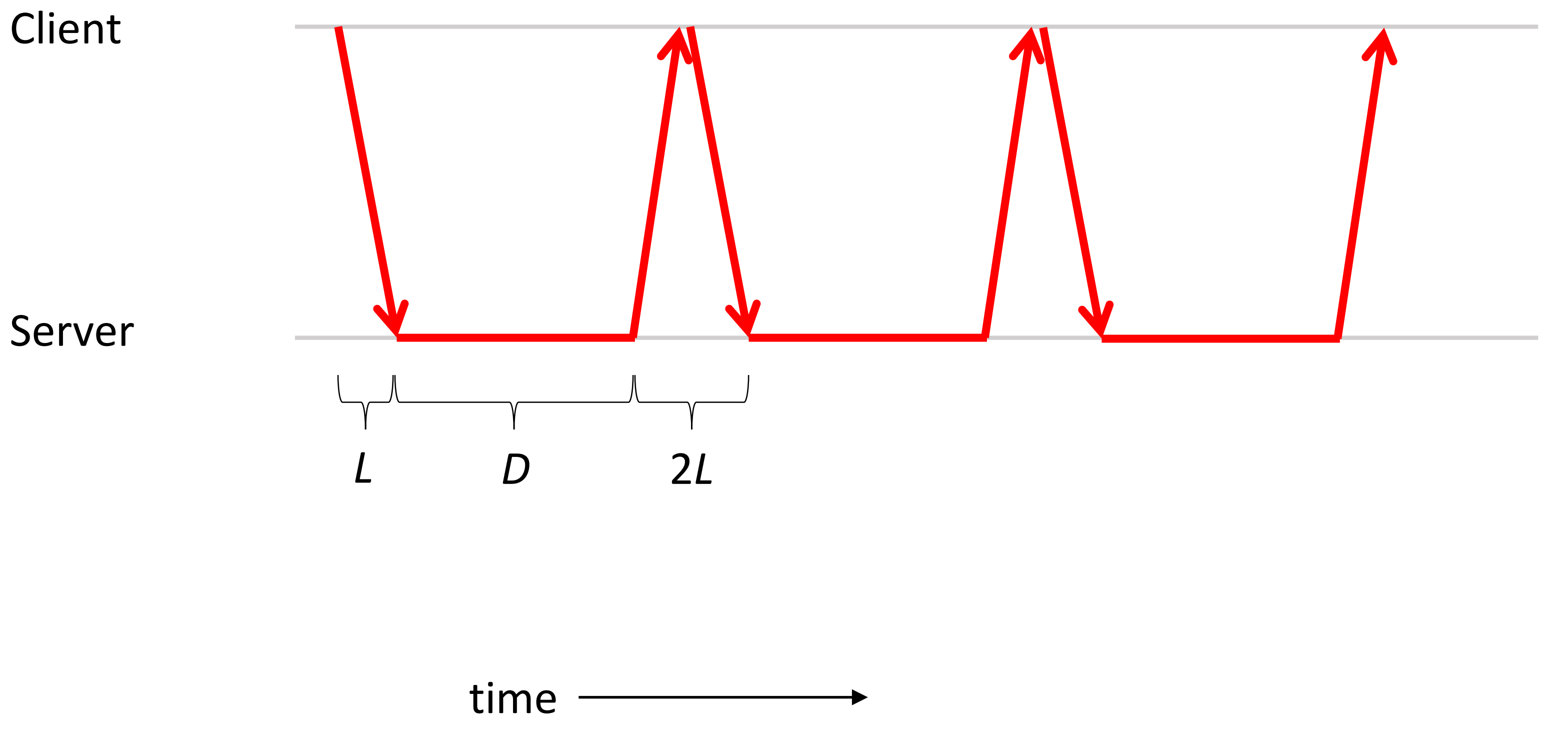
- Plugging in the upper bound on throughput from earlier formulas, the above yields a lower bound on end-to-end latency.

Example 1: client-limited

throughout $= \frac{1}{\text{latency}} = \frac{1}{D + 2L}$

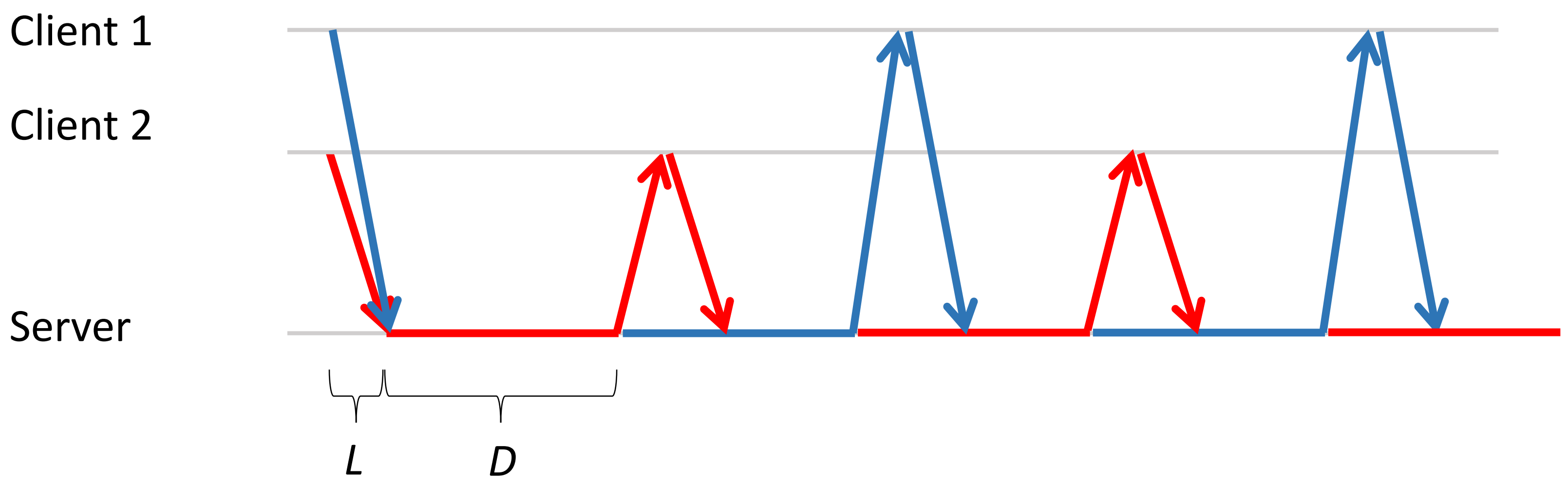
D: how long server process one request $\longleftrightarrow$ decrease D

C: one-way network delay



Example 2: server-limited

throughput $= \frac{1}{\text{latency}} = \frac{1}{D}$


Client 1
Client 2
Server
L
D