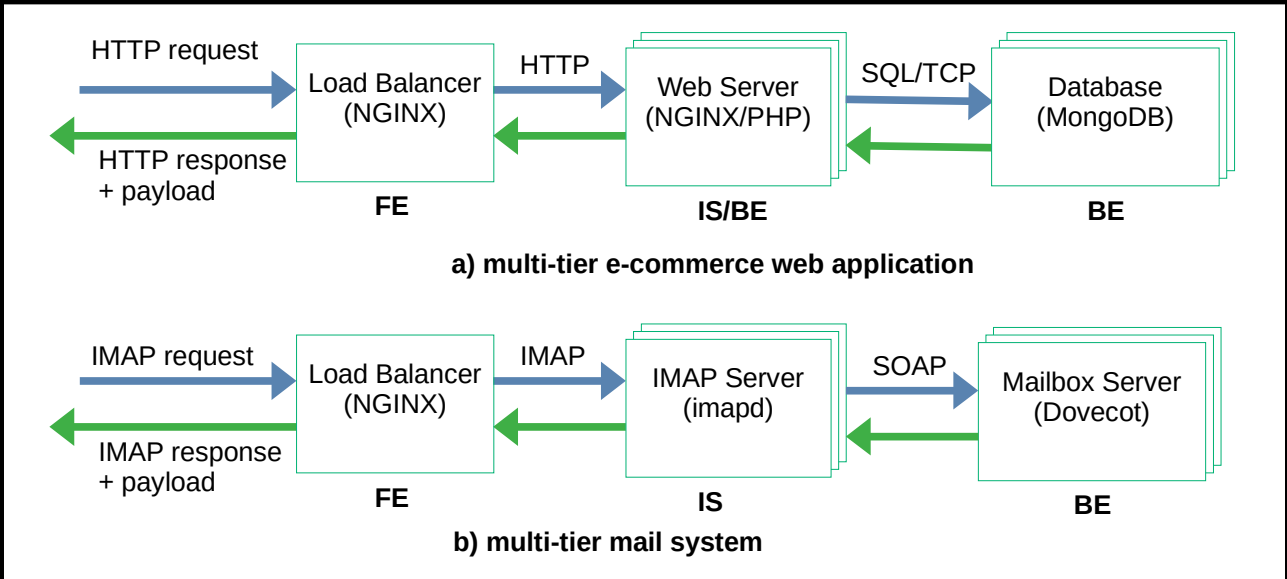
DiSC: Backpressure Mitigation In Multi-tier Applications With Distributed Shared Connection

Alain Tchana - KrakOS/Inria/Grenoble INP-UGA

NSDI 2025 - April 29 - Philadelphia (USA)

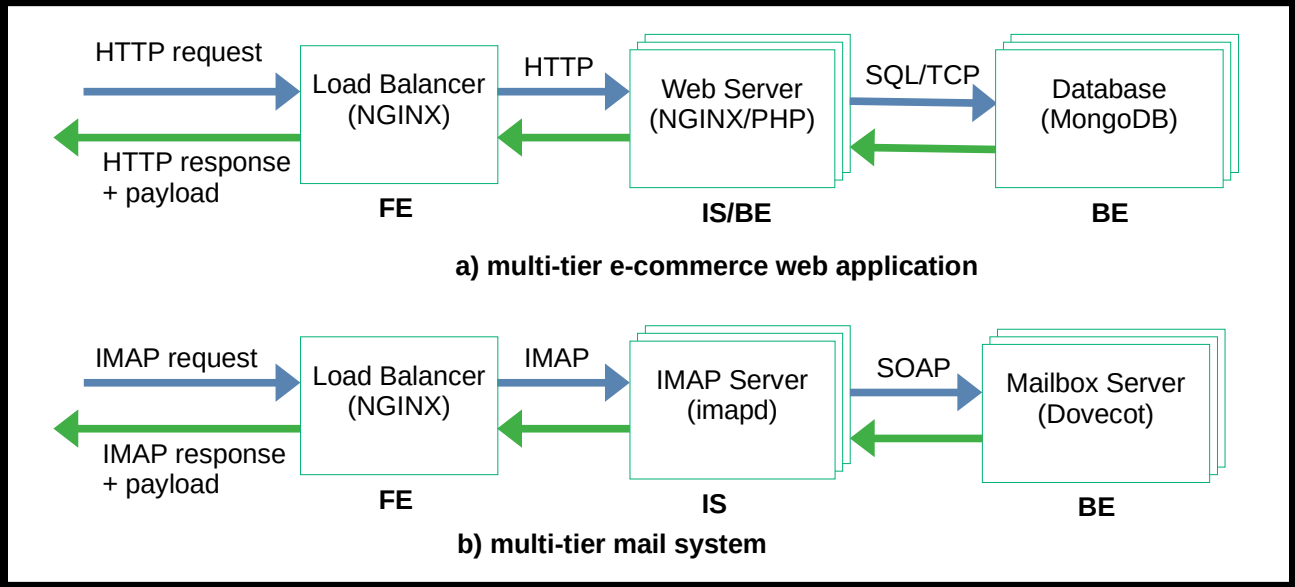
Joint work B. Ekane, D. Mvondo, R. Lachaize, D. Bromberg and D. Hagimont

Motivating Examples



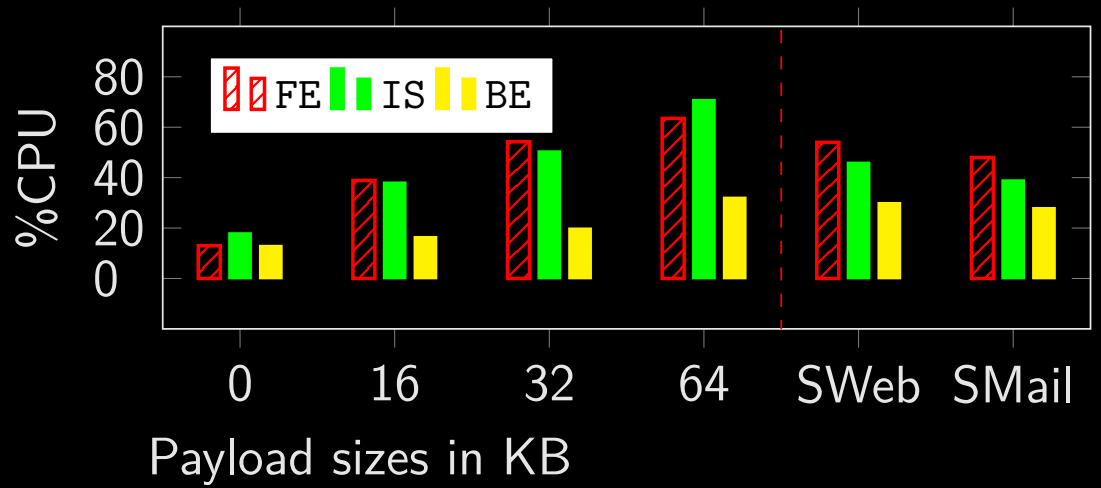
Final Data

- Final payload (e.g., mails, attached files) are never used by intermediate servers (IS) and the frontend (FE)
- Relaying final data
  $\Longrightarrow$ Backpressure on IS and FE
  $\Longrightarrow$ Scalability issues



Backpressure

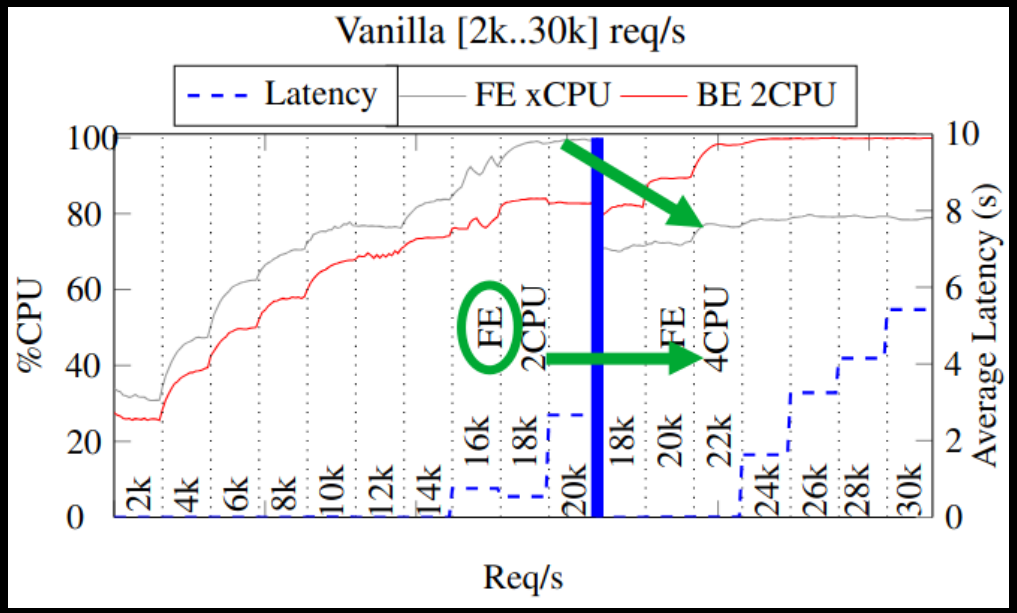
- Final payload (e.g., mails, attached files) are never used by intermediate servers (IS) and the frontend (FE)
- Relaying final data
  $\Longrightarrow$ Backpressure on IS and FE



Higher load on FE and IS, compared to BE

Scalability Issues

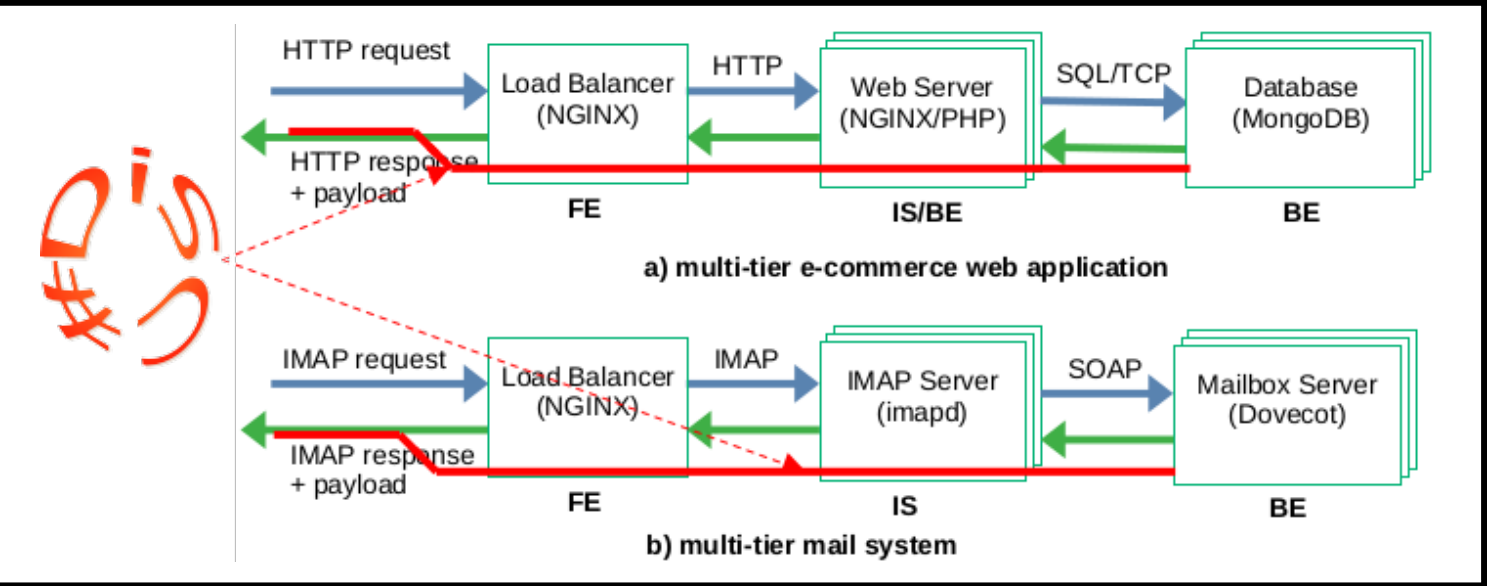
- Final payload (e.g., mails, attached files) are never used by intermediate servers (IS) and the frontend (FE)
- Relaying final data
  $\Longrightarrow$ Scalability



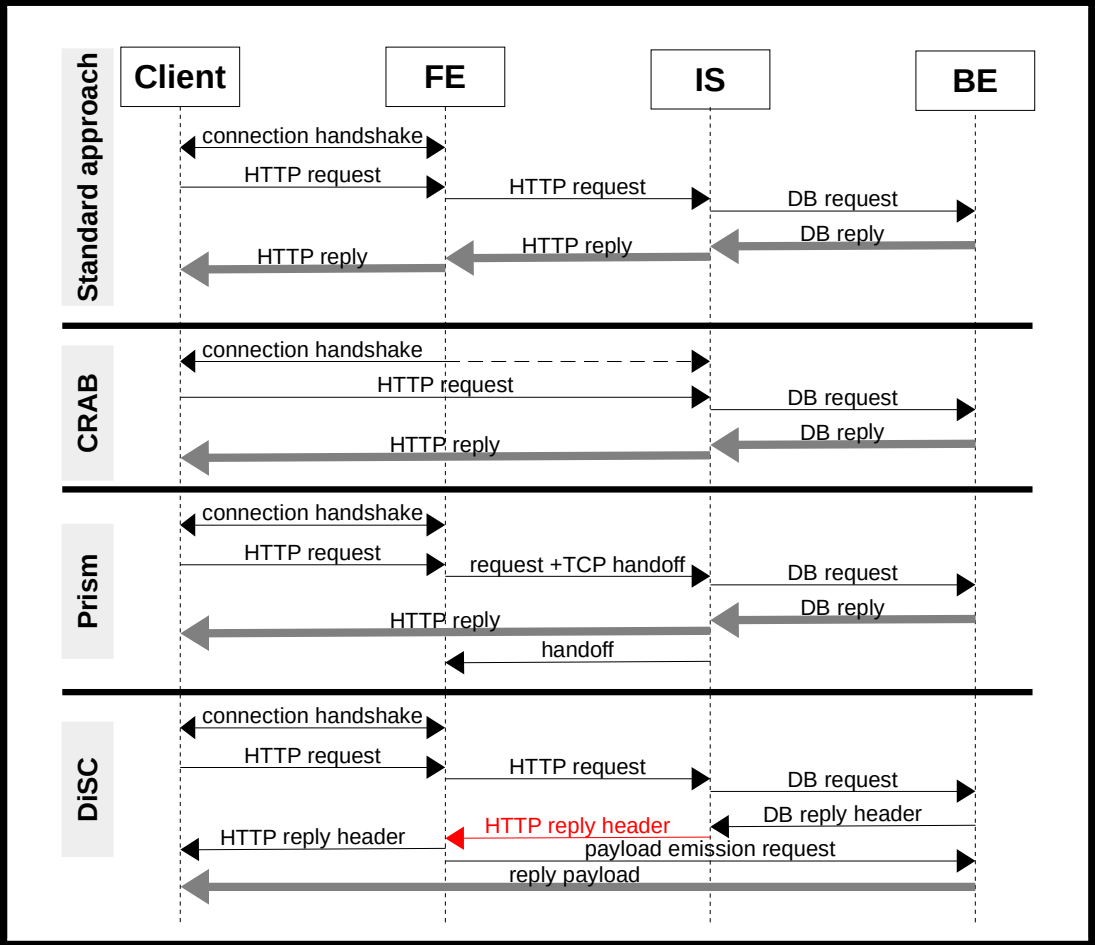
FE **sturates first**

DiSC: Bypass + Distributed TCP

- Final payload (e.g., mails, attached files) are never used by intermediate servers (IS) and the frontend (FE)
- Bypass IS and FE for final data



State-of-the-art



Connection handoff: DSR, CRAB [ATC'20] and Prism [NSDI'21], QDSR [ATC'24]

Main limitations

- Only bypass a single server (LB)
- All servers must implement the same protocol (e.g., HTTP)
- Fully bypass the LB
  - LB (API gateway) collect stats, provide authentication, manage billing, implement smart load balancing algo.

Behind DiSC

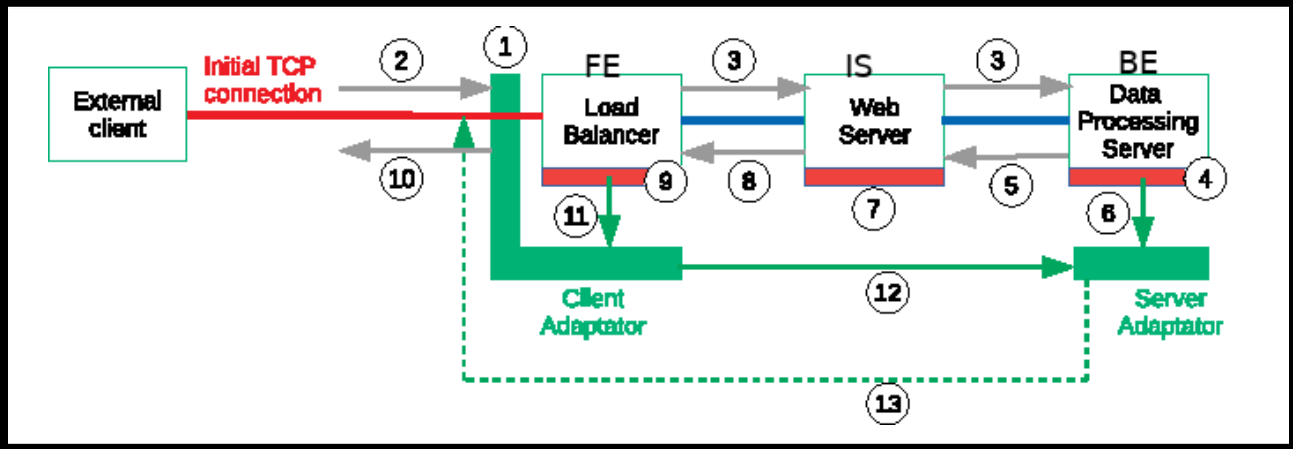
- A response is composed of
  - header+payload+footer
- Bypass only applied to the payload
- Header and footer follow the normal communication path

Design goals

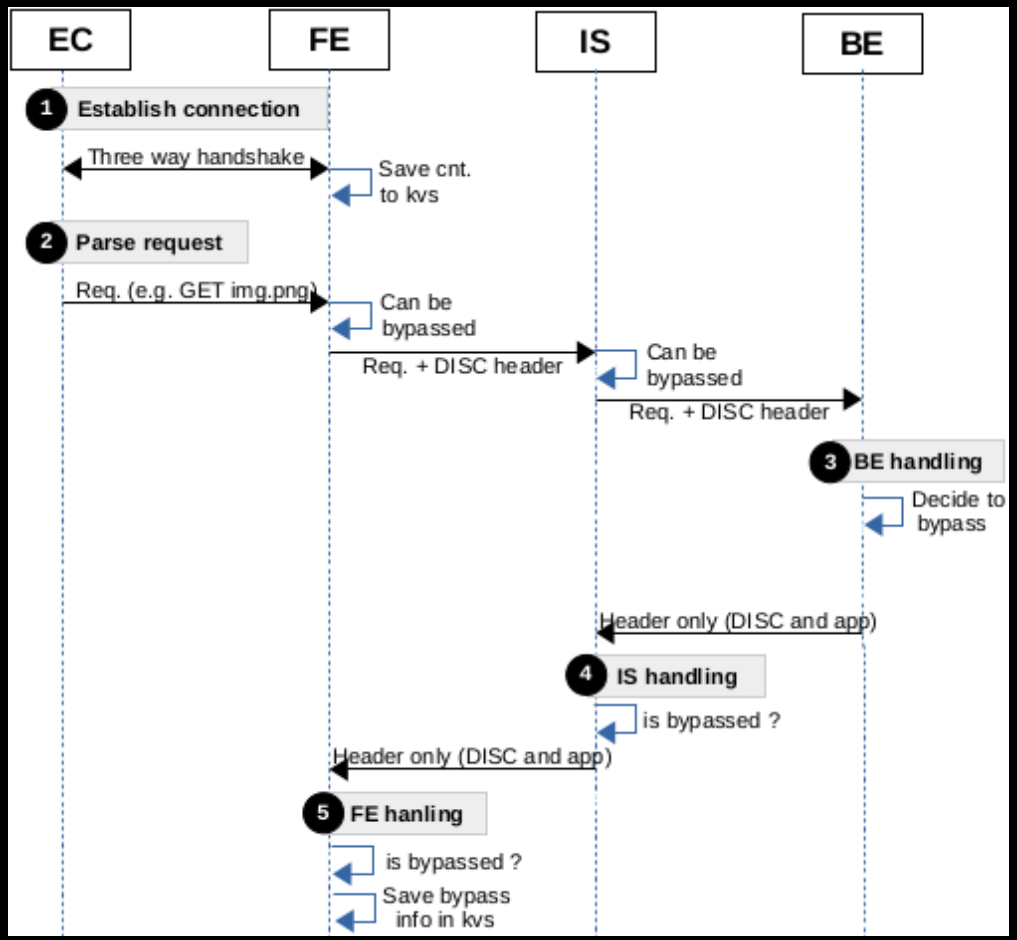
- Transparent to clients
- Fine-grained bypass
- Modest modification for the application
- Support all application-level protocols
- Keep the storage at the same security level
- Support TLS

DiSC

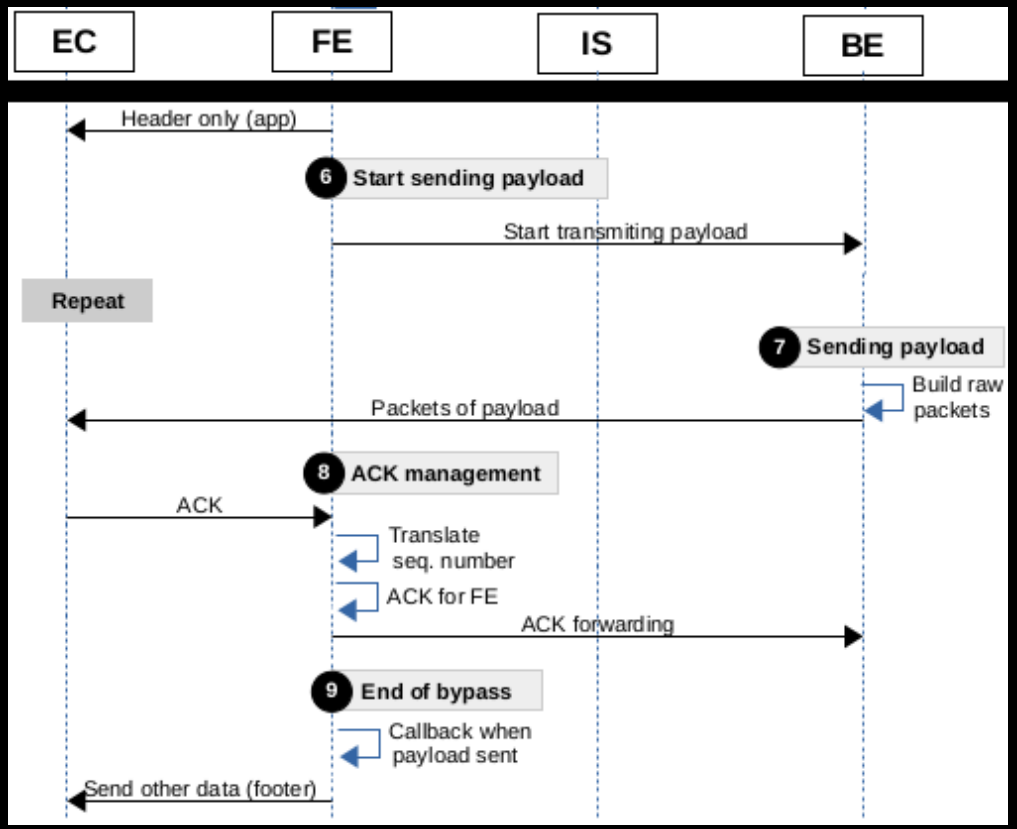
- DiSC-Prot, a new protocol
  - A node can indicate that it can be bypassed
  - Lies between TCP and the application-level protocol
- Main techniques
  - Transparent (BPF) Address spoofing, ACK redirect, Seq number rewriting, TLS session serialization-export



End-to-end coordination with DiSC



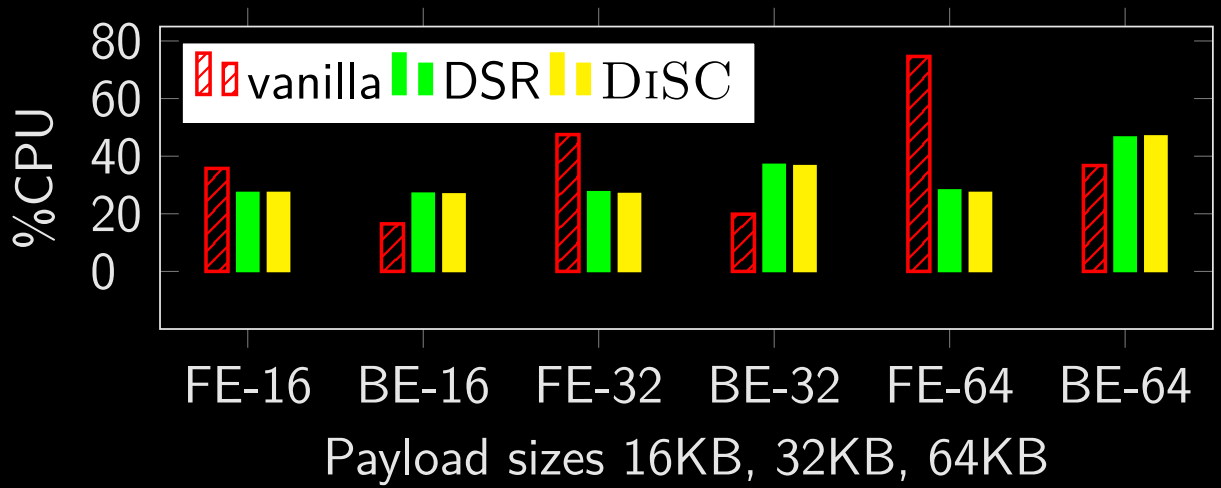
End-to-end coordination with DiSC



Evaluation

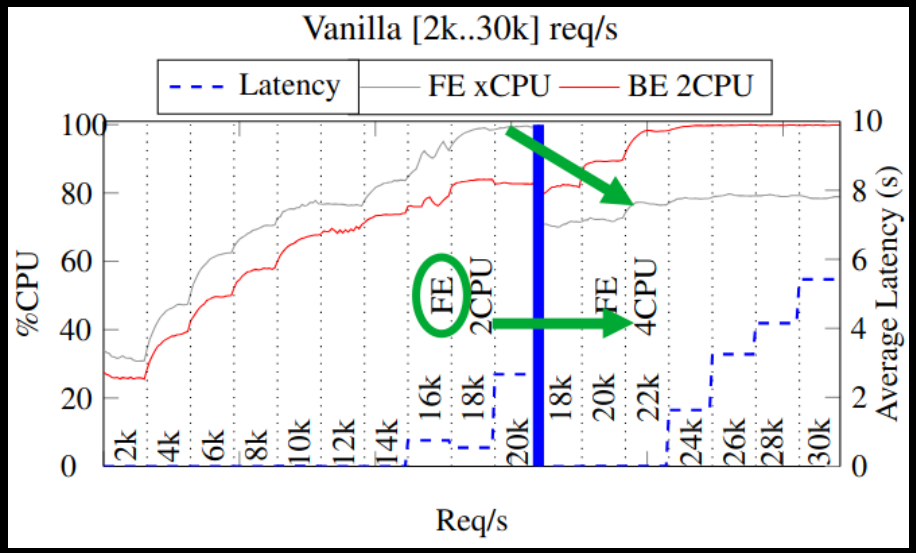
- Testbed: CloudLab
- Micro-benchmarks: Nginx-based n-tier web app that returns static contents
- Macro-benchmarks: SpecWeb, SpecMail, Train Ticket, and Social Media (DeathStarBench suite)

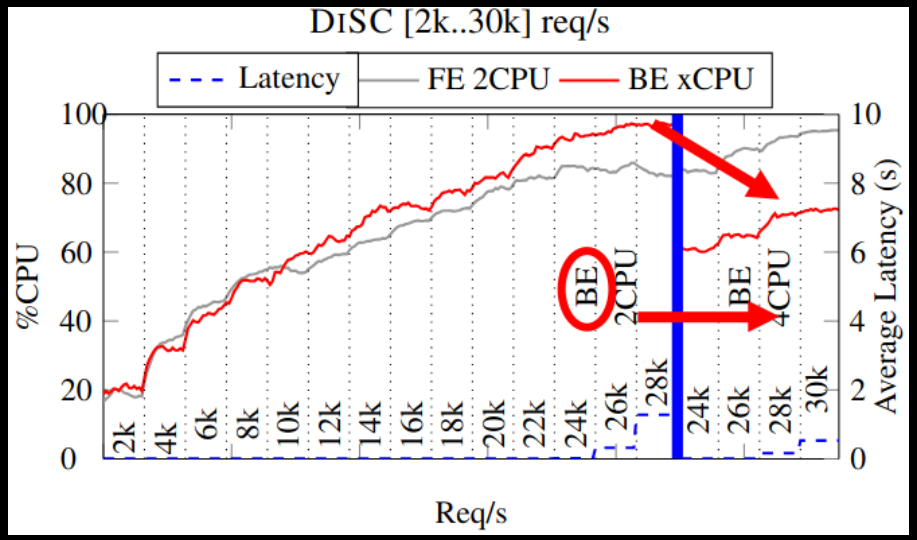
Load offloads from FE to BE



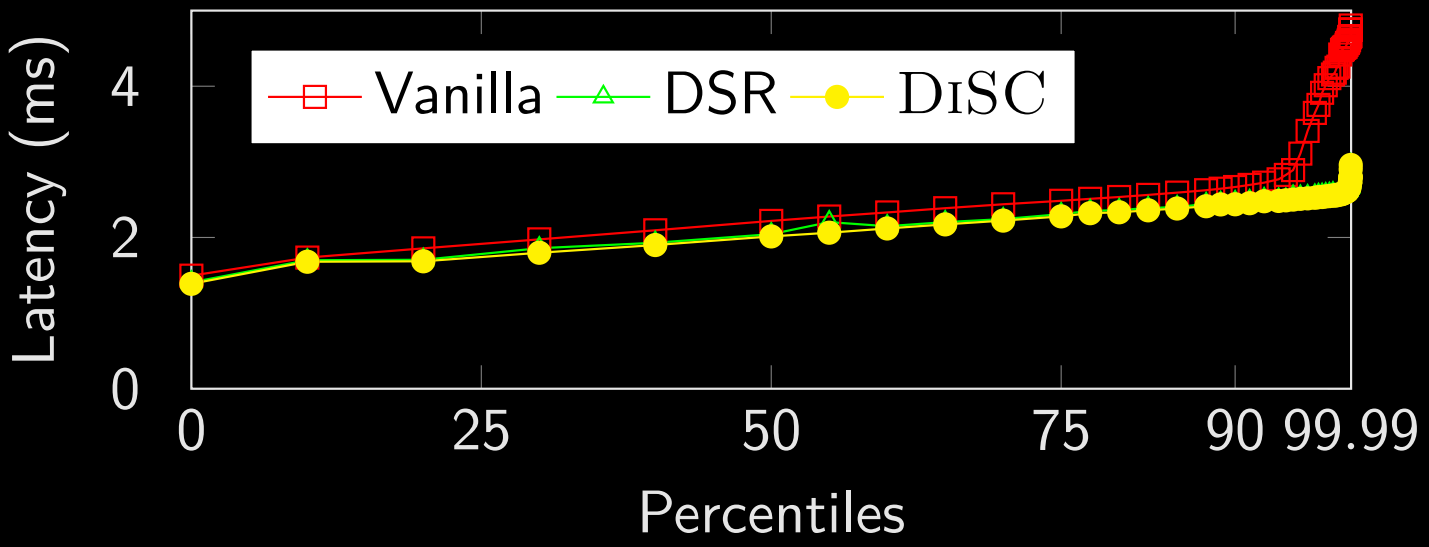
DiSC efficiently offloads load from FE to BE, almost as DSR

Good scalability





Better tail latency



More results

DISC: Backpressure Mitigation in Multi-tier Applications with Distributed Shared Connection

Brice Ekane*, Djob Mvondo*, Renaud Lachaize†, Yérom-David Bromberg*, Alain Tchana†, Daniel Hagimont‡
*Univ. Rennes, Inria, CNRS, IRISA, France
†Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, 38000 Grenoble, France
‡IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3 Toulouse, France

Abstract

Most data-center applications are based on a multi-tier architecture, involving either coarse-grained software components (e.g., traditional 3-tier web applications) or fine-grained ones (e.g., microservices). Such applications are prone to the backpressure problem, which introduces a strong performance coupling between tiers, thus degrading scalability and resource consumption. This problem is due to the fact that, on the response path towards the initial client, a significant fraction of the payloads in the messages exchanged between tiers correspond to "final" data that are simply relayed (i.e., without further modifications) from a backend tier such as a database. This traffic results in additional pressure on the intermediate and frontend tiers.

To address this problem, we introduce DISC, a system allowing several tiers within a multi-tier chain to jointly act as endpoints of the same TCP connection. This enables the selective bypass of one or several tiers on the response path. Unlike existing solutions, DISC is (1) flexible — it accommodates arbitrary multi-tier topologies and heterogeneous application-level protocols, (2) fine-grained — it allows multiple tiers to be involved in the generation and emission of a given response message (e.g., to decouple the network path of the response headers and footers from the path of the response body), (3) and non-intrusive — it requires only minor and localized/modular modifications to the code base of legacy applications and is transparent for external clients. Evaluation results with several micro- and macro-benchmarks show that DISC can reduce the cumulative CPU load on servers by up to 41.5%, decrease the average and tail latencies respectively by up to 74.1% and 5.71×, and also improve the request rate by up to 45%.

1 Introduction

For reasons of modularity and, above all, scalability, Internet services are built in the form of an assembly of components in an architecture formerly referred to as *multi-tier* or more recently as *microservices*. In such an architecture, components are loosely coupled and communicate through messages using protocols such as HTTP, IMAP, gRPC, etc. In the remainder of this paper, we use the terms *"tier"* and *"service"* interchangeably, as our contribution is agnostic to the number and granularity of these components.

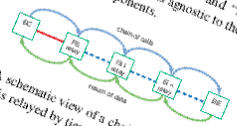


Figure 1: A schematic view of a chain of tiers. Tier BE returns data which is relayed by tiers T_{S_1} up to FE toward client EC.

Tier *independence* is a fundamental property that makes the popularity of multi-tier architectures. However, this independence is often compromised regarding performance, making it challenging to scale each tier independently. Our contribution aims at reducing redundant data transfers, which are a major impediment to *performance independence* of tiers. As illustrated in Fig. 1, there are redundant data transfers among different tiers involved: a request initiated by an External Client (EC) is received by a first-tier, known as the FrontEnd (FE). FE may require a call to another tier, thus forming a chain of invocations traversing Intermediate Servers (T_S) up to a final tier named the BackEnd (BE). Such a chain can be noted as the following: $EC \leftrightarrow FE \leftrightarrow T_{S_1} \leftrightarrow \ldots \leftrightarrow T_{S_n} \leftrightarrow BE$. In this chain, when BE produces a reply that must be returned to EC without further (read/write) processing by the T_S and FE tiers, it still must go up the chain in the reverse order of request calls, generating redundant communications. We refer to such replies as *final data*. An example of final data is the requested (and attached files) in a (multi-tier) webmail such as Zimbra [7]. Even if T_S and FE tiers only relay the final data, they undergo a load proportional to the activity of BE. Consequently, the performance of T_S and FE is strongly coupled with the activity of BE. This is known as the *backpressure* problem [15]. Performance dependency and ... challenging to manage the scalabi...

This work

- Introduced DiSC
- Intermediate server bypass to directly transfert final data
- Reduces CPU utilization on intermediate servers
- Reduces tail latency
- Better scalability
- Meta-data are not bypassed
- Support TLS, no client adaptation, modest server modification, based on BPF
- Experiments on several apps (C, Java-based, traditional n-tier and microservices-based)

Contact me!

- alain.tchana@gmail.com
- Postdoc in Systems at `lig-krakos.imag.fr`
- Grenoble (Alpes, Ski) - France (+250 types of cheese)