Network Connection Optimization for Serverless Workloads

**Introduction**

**Serverless Background**
- Application developers produce an event-driven function to the cloud provider
- The cloud provider is responsible for invocation, scaling, billing, failure, and management

**Motivation**
The use of short-lived, independent units of computation can lead to avoidable inefficiencies:
- Traditional applications would maintain long-held TCP connections
- Each newly instantiated serverless function must create new sockets and perform TCP slow start
- Serverless functions block for writes, while distributed services use asynchronous writes
- Caching and advanced reads can reduce time spent on repetitive, predictable operations

We first develop an OS-level shim layer to provide socket reuse between identical short-lived functions.

**Architecture**

- A serverless function running in container $func_1$ opens a socket to an external database $DB_{base}$
- A shim layer intercepts socket system calls
- The pool returns pre-existing socket $S_1$
- $S_1$ is an unused warmed socket or a socket created by a previous invocation

**Preliminary Work - Reuse**

The effects of reuse are measured using two nodes (Figure 2):
- A node with a client program that runs twice (once to use the socket, once to reuse)
- A node running a server program representing an external resource

Figure 3 shows latency measured from connection establishment to completion for three client configurations:
- **Baseline**: No shim layer is installed, system is unmodified (median: 446 µs)
- **DummyShim**: A shim layer intercepts system calls, but does not change socket behavior (median: 459 µs)
- **Reuse**: A shim layer contains minimal logic needed to perform one instance of socket reuse (median: 322 µs)

Small flows ($<1$ MB) complete 2-24 ms faster (with congestion) or 0.7-1.2 ms faster (no congestion).

**Future Work**

Create a pool manager with features including:
- Communication between the pool manager and the function scheduler
- Intelligent connection pool garbage collection
- Network probing to approximate appropriate congestion windows for warmed sockets

The long-term goal is to explore:
- TLS integration
- Asynchronous writes
- Caching of commonly accessed data
- Proactive retrieval of network content