Performance Interfaces for Systems Code

EPFL

Rishabh Iyer

Katerina Argyraki

George Candea

DSLAB

NAL

Problem: Semantic interfaces abstract functionality & are indispensable when building complex systems
There exists no equivalent, widely-used construct that abstracts performance behavior

Semantic interfaces summarize functionality simply & precisely

- Reveal only the information that a user of the system requires
  - All other implementation details are abstracted away
- Examples: Header files, code documentation, formal specifications
- Hard to imagine building systems without them

connect

```
public void connect(SocketAddress endpoint) throws IOException
```

Connects this socket to the server

Parameters:
`endpoint` - the `SocketAddress`

Throws:
`IOException` - if an error occurs during the connection
`IllegalBlockingModeException` - if this socket has an associated channel and the channel is in non-blocking mode

Can there exist a performance interface?

- Summarizing performance is hard!
  - Performance depends on low-level implementation details
  - Must take into account different hardware platforms
- Existing approaches to summarizing performance are insufficient
  - Big-Oh notation: Only talks about asymptotic scalability
  - Worst-Case-Execution Time: Only focused on absolute worst case
  - Service Level Objectives: Only focused on tail behavior

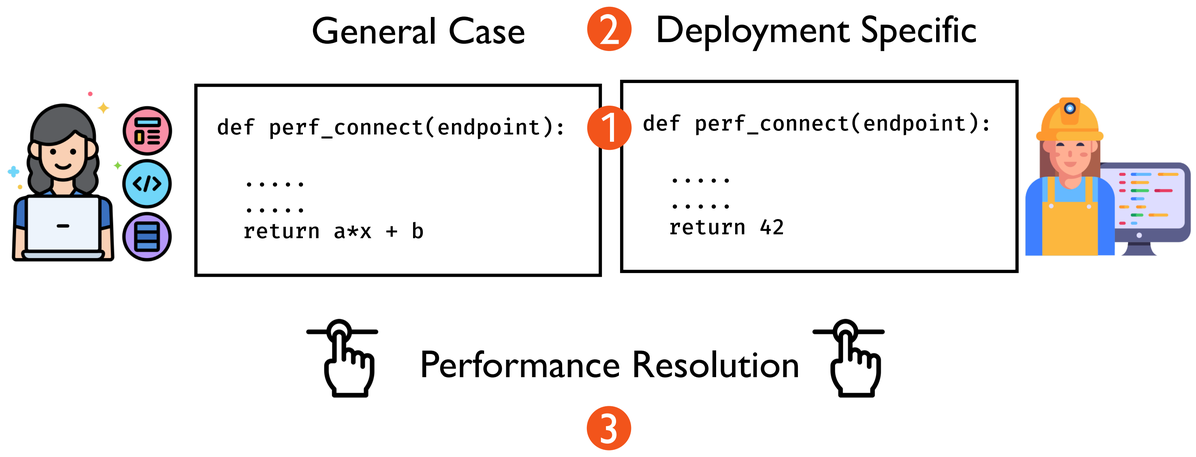
Existing performance summaries sacrifice precision for simplicity

Solution: Performance Interfaces as programs that summarize the latency of the system

Key insight: Reason about the system separately from the environment it runs in

Three key ideas;

1. Interface as a program with same inputs that returns latency
   - Systems engineers are intuitively familiar with programs
2. General-case (GC) vs Deployment-specific (DS) interface
   - GC interface represents the code, parameterizes environment
   - DS interface is an instantiation of the GC interface for an environment
3. Performance resolution: Granularity at which interface describes latency
   - Provides users with explicit control of the simplicity-precision tradeoff

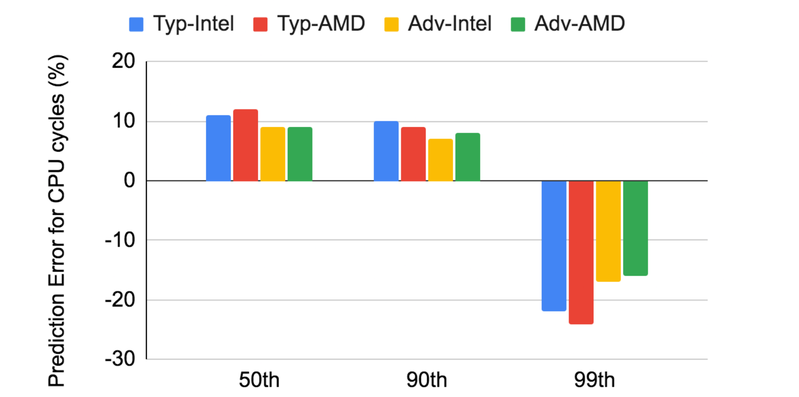


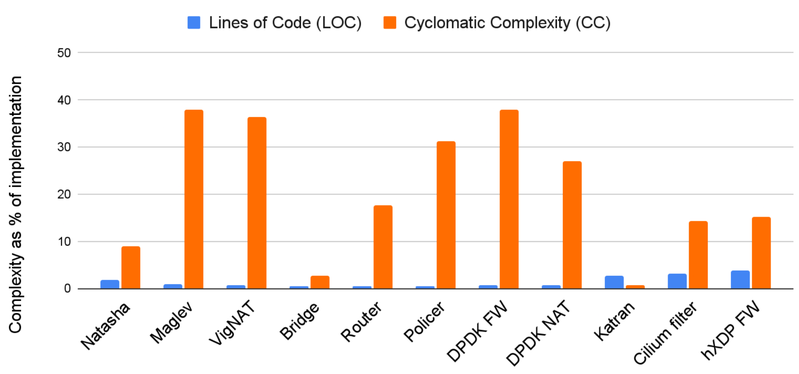
Two kinds of interfaces tailored to the requirements of developers (system) and operators (environment) respectively

PIX: Program Analysis tool that automatically extracts Performance Interfaces from source code

Evaluation Highlights:

- Extracted interfaces for 12 Network Functions, 12 programs from OpenSSL
  - 3 Network Functions used in production: Meta, Scaleway, Cilium
- Extracted interfaces are simple, precise and quick to extract
  - 100-1000x shorter than implementations
  - Average prediction error < 10% across multiple deployments
  - Take < 25 mins to extract even for production NFs
- PIX-extracted interfaces are useful to developers & operators!
  - Developers: automatically find performance bugs/regressions:
    - Found 15% latency in Meta's load balancer
    - Found constant-time violation in OpenSSL 3.0
  - Operators: Informed development decisions
    - Pick right Network Function for deployment hardware (NIC)
    - Diagnose root-cause of in-production performance anomalies





Performance interfaces summarize the latency of a system simply and precisely, just like semantic interfaces summarize functionality