SpotProxy: Rediscovering the Cloud for Censorship Circumvention

Patrick Tser Jern Kon[1], Sina Kamali[2], Jinyu Pei[3],
Diogo Barradas[2], Ang Chen[1], Micah Sherr[4], and Moti Yung[5,6]

[1] University of Michigan [2] University of Waterloo [3] Rice University
[4] Georgetown University [5] Columbia University [6] Google

Contact info: Patrick Kon (patkon@umich.edu)

SpotProxy Elevator Pitch

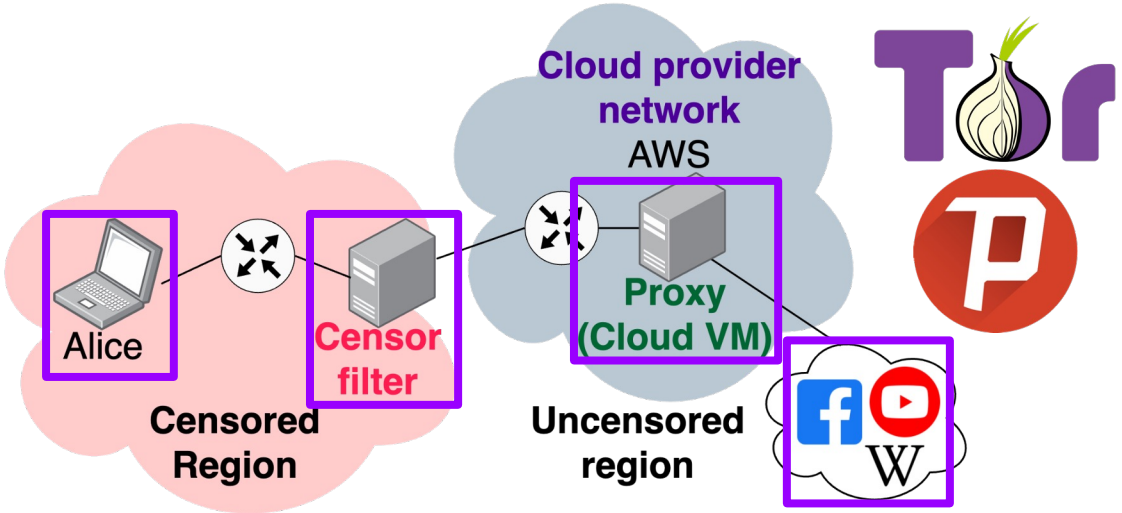
A circumvention proxy architecture inspired by Cloud Spot VMs

~90% VM cost reductions for hosting proxies

Massive proxy footprint coverage

Censors can't block our proxies without incurring significant costs

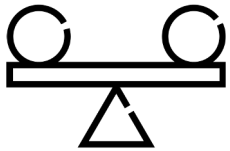
Censorship circumvention using Cloud VMs



- Censor blocks packets destined for **FB's IP**
- Alice's packets are destined for a **proxy's IP**
- **Power imbalance** between the censor and the censored

Benefits of the cloud

Pay-per-use model



Stable performance

Automated provisioning

Ease of configuration

High availability

Customizable performance

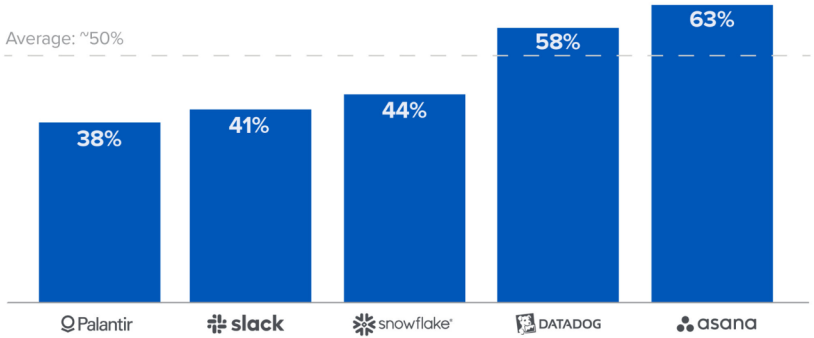
However…the cloud is expensive!

- Cloud VMs are expensive
- Many charges incurred even when idle

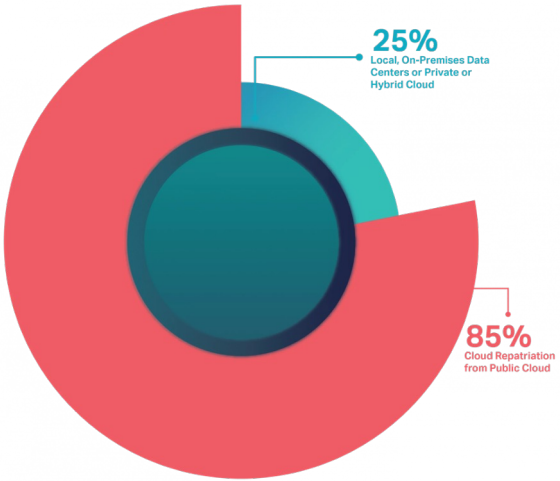
91%

are wasting money in the cloud

Estimated Annualized Committed Cloud Spend as % of Cost of Revenue

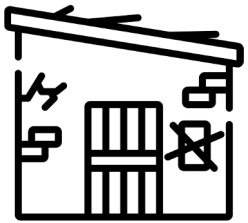


Source: Company S-1 and 10K filings



Source: https://www.pepperdata.com/blog/avoiding-cloud-repatriation/
Source: https://a16z.com/the-cost-of-cloud-a-trillion-dollar-paradox/
Source: https://www.hashicorp.com/state-of-the-cloud

Impact of cloud costs on circumvention



Financial strain on circumvention provider:

- Common especially during usage surges
- E.g., Lantern resorting to donations and using credit cards to sustain operations

Financial strain on censored clients:

- Clients could be financially constrained
- Clients may need government/NGO donations
- Payment channels for proxies could be restricted
- Paying for VPN services could be criminalized

Cloud features must be “rediscovered” for censorship. Simply “lifting-and-shifting” falls short

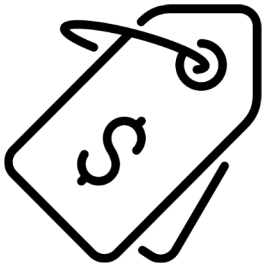
Inspiration: Spot VMs

Google Cloud aws ORACLE Cloud Infrastructure Azure

- Spawned out of cloud-provider excess resources to maximize profit
- Taken back when higher paying tenant requests arrive
- **Pro**: Cost a fraction of the price of regular VMs
- **Con**: Spot VMs can be reclaimed at a moment's notice
- Annoyance for regular applications, but a boon for circumvention

SpotProxy: Systematically leveraging cloud-native features to maximize the use of every dollar in circumvention

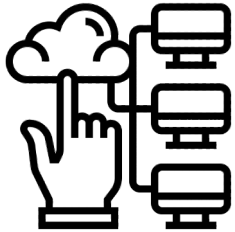
SpotProxy design goals



Minimal cost

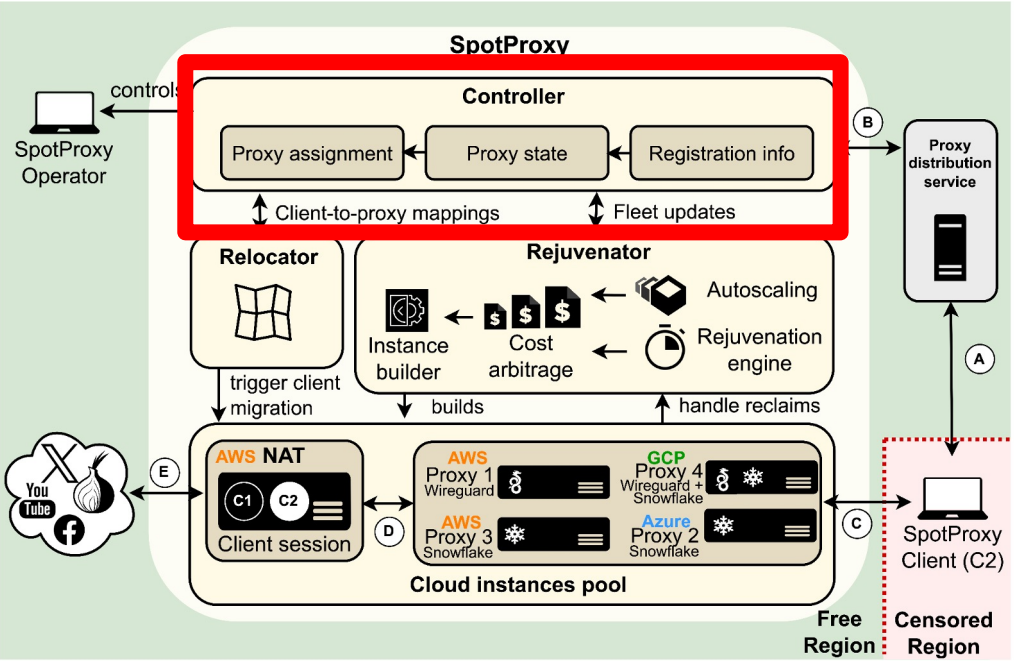
Cloud-native unblockability

Seamless connectivity



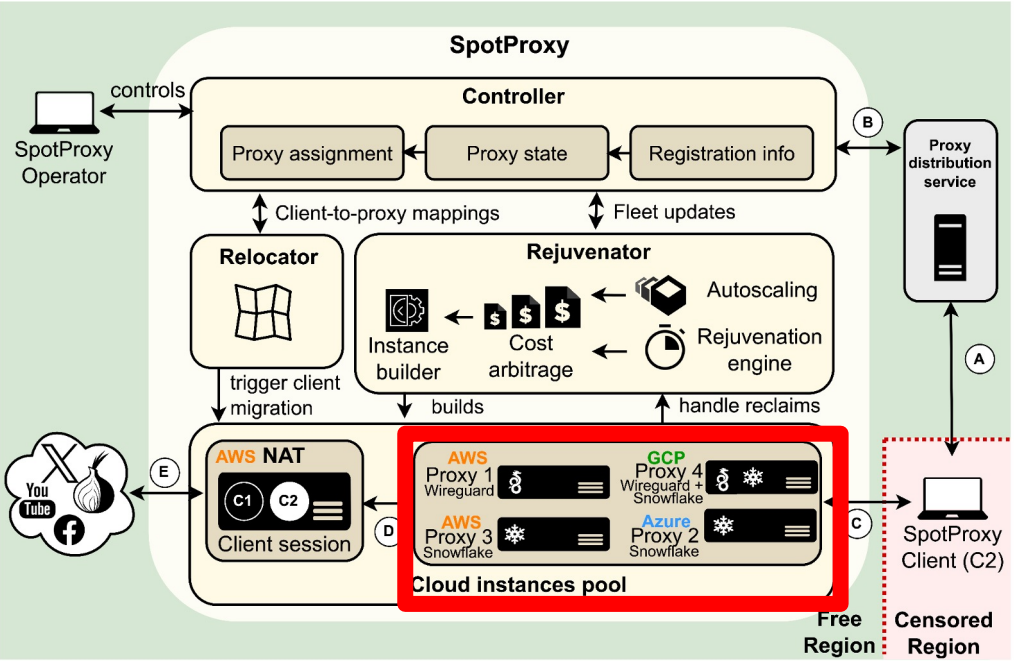
Cloud-agnostic features

SpotProxy architecture & workflow overview



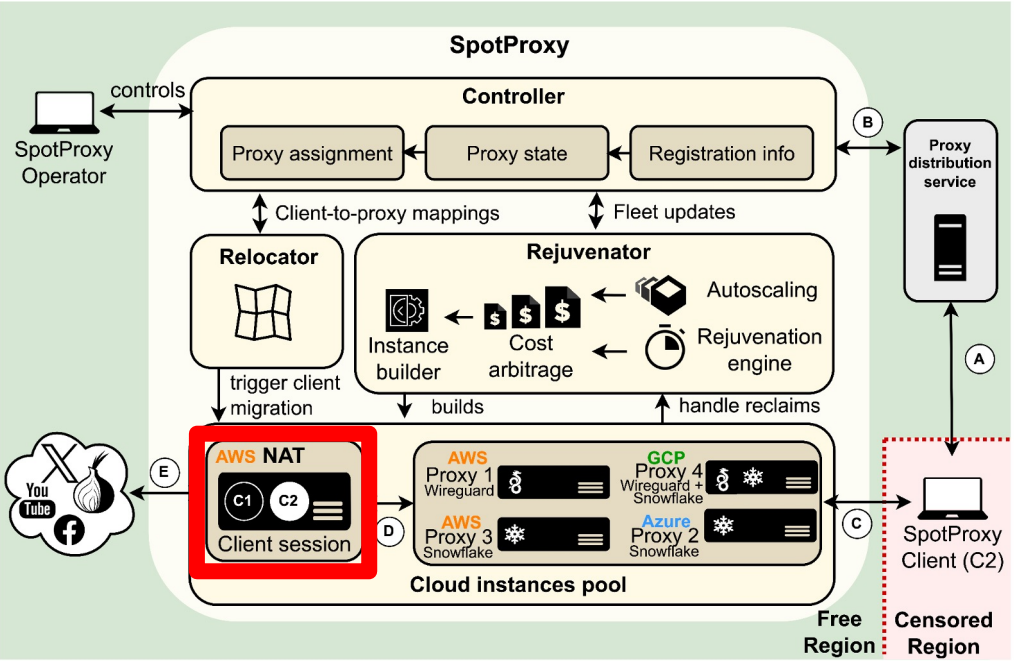
- The client first requests for a proxy from some arbitrary **proxy distribution service**
- Which forwards the request to our controller
- The controller assigns some proxy to the client, using some arbitrary assignment algorithm (e.g., Trust-based)

SpotProxy architecture & workflow overview



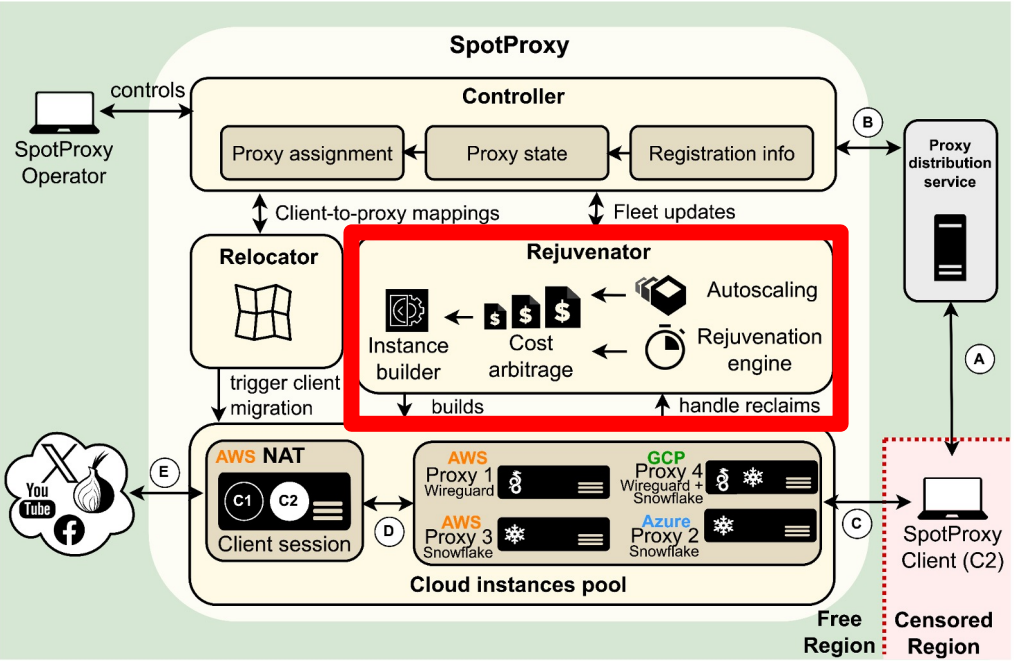
- The client connects to its assigned proxy device
- This proxy device lives on a VM within our Cloud instance pool
- This proxy pool is ephemeral
- Proxies can be of arbitrary implementations (e.g., Wireguard)

SpotProxy architecture & workflow overview



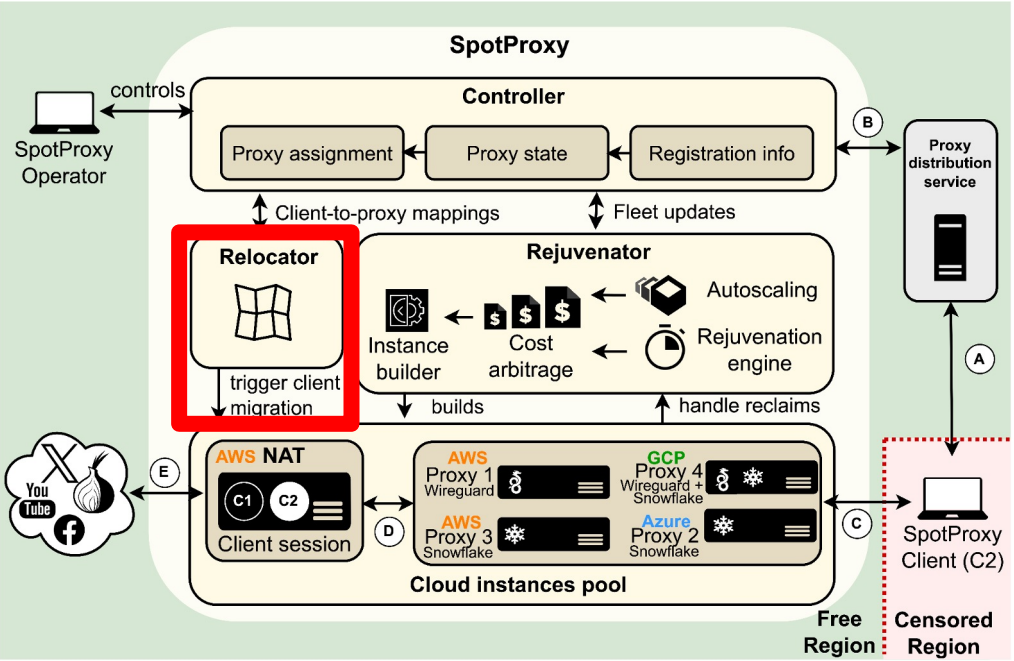
- All connections are forwarded to our NAT devices
- Which maintains a stable vantage point for the destination (e.g., Facebook)
- This is important for relocation
- NAT devices should be deployed on stable machines

SpotProxy architecture & workflow overview



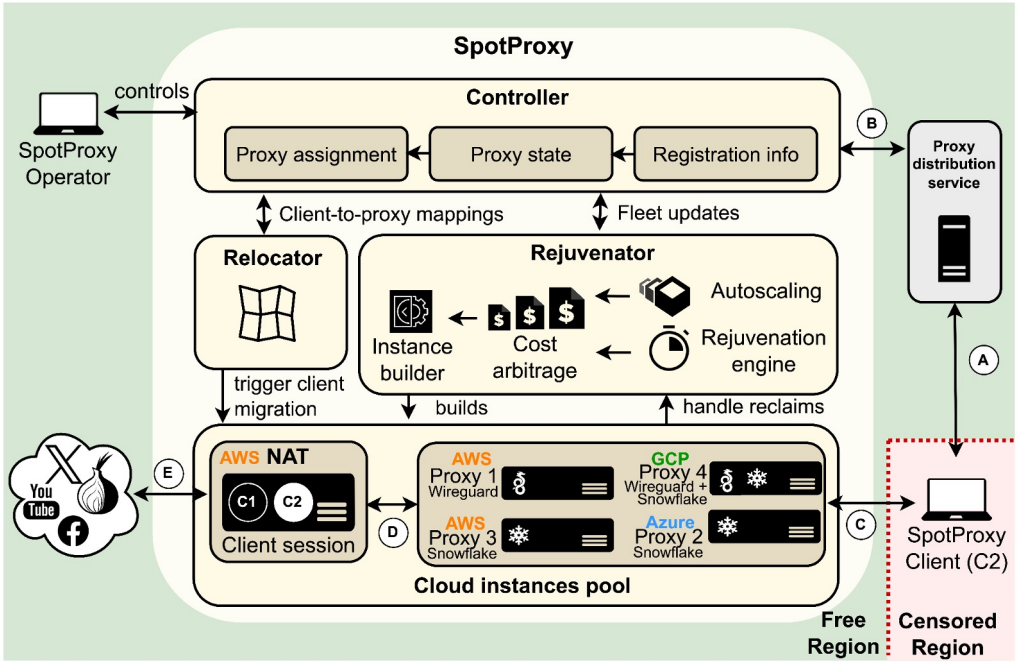
- Rejuvenator:
  - Continuous fleet refresh
  - Cheapest possible instances
  - Embracing cloud provider induced SpotVM reclamation
  - Counters IP-based blocking

SpotProxy architecture & workflow overview



- Rejuvenator:
  - Continuous fleet refresh
  - Cheapest possible instances
  - Embracing cloud provider induced SpotVM reclamation
  - Counters IP-based blocking
- Relocator:
  - Migrates clients to new proxies
  - E.g., fleet composition changes
  - No action required by client

SpotProxy architecture & workflow overview



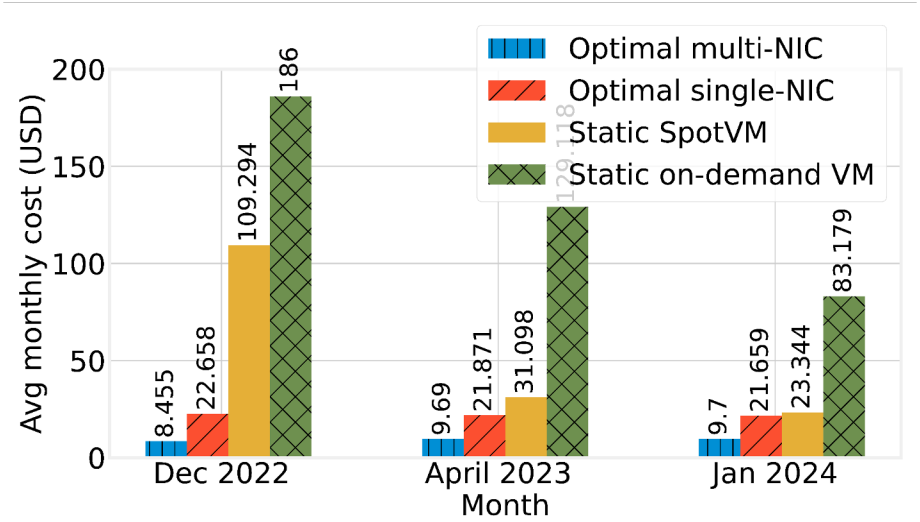
- Rejuvenator:
  - Continuous fleet refresh
  - Cheapest possible instances
  - Embracing cloud provider induced SpotVM reclamation
  - Counters IP-based blocking
- Relocator:
  - Migrates clients to new proxies
  - E.g., fleet composition changes
  - No action required by client

Refer to our paper for more details!

Implementation and experimental setup

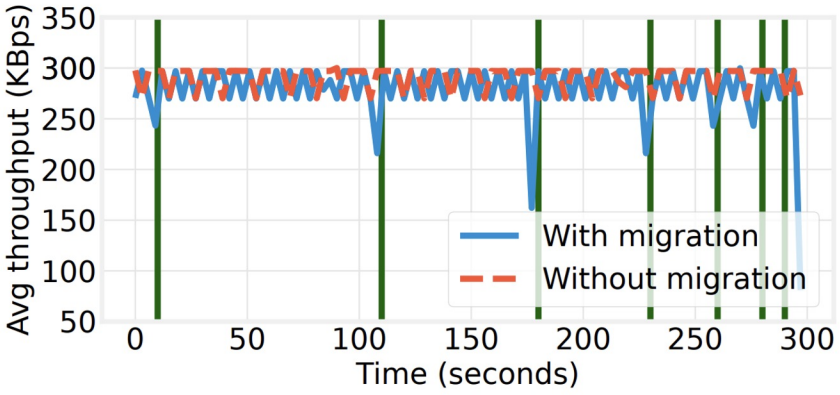
- SpotProxy relocator implementation on:
  - Wireguard: wrote a SpotProxy management layer around Wireguard
  - Snowflake: minimal targeted source code modifications
- SpotProxy rejuvenator currently supports AWS
  - Through its official SDK for Python (Boto3)
- Evaluation performed on a live network testbed with AWS EC2 VMs
- Circumvention efficacy evaluated using a SOTA censor simulation platform

Evaluation: instance cost savings

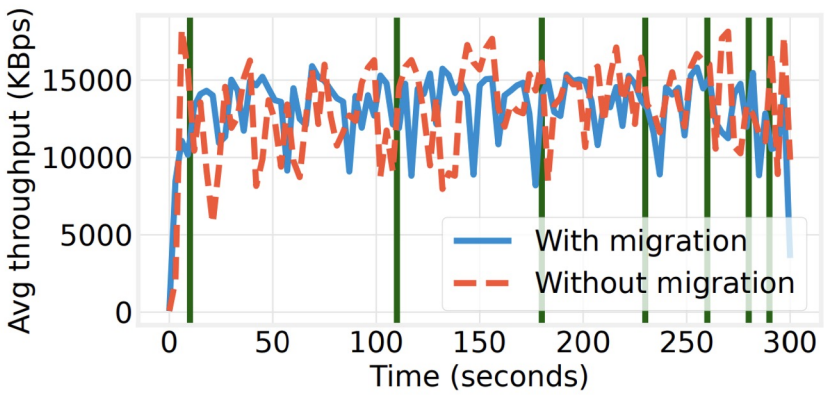


SpotProxy rejuvenation can provide significant cost savings

Evaluation: relocator performance



(a) Wireguard: HTTP web server loading.



(f) Snowflake: SCP bulk file download.

SpotProxy's relocator incurs negligible overheads

Extensive evaluations available in the paper!

SpotProxy: Cloud-hosted proxies with maximum circumvention utility

- SpotProxy currently provides:
  - Significant instance cost savings (**up to ~90% reduction**)
  - Cloud-native unblockability: **access to > ~60% of clients** even when 50% are sybils
  - **Seamless network connectivity** with SpotProxy's relocator
  - Support for **AWS** & integration with 2 proxy implementations: **Snowflake and Wireguard**
- **Working prototype** available on GitHub: https://github.com/spotproxy-project/spotproxy
  - Active expansions and improvements are in progress
  - All contributions are welcome!

- I'm actively searching for research internships in industry!

Thank you! Feel free to reach out for a chat!

Patrick Kon patkon@umich.edu