Picocenter: Supporting long-lived, mostly-idle applications in cloud environments

Liang Zhang*

Theophilus Benson§

*Northeastern University

James Litton‡

Dave Levin‡

‡University of Maryland

Frank Cangialosi‡

Alan Mislove*

§Duke University

Open-source alternatives

Service oriented applications: **long-lived**

For single user or small group: **mostly-idle**

Long-lived, mostly-idle (LLMI) applications

How do users run LLMI applications in cloud?

Running LLMI applications in cloud

- Platform as a Service (PaaS)
  - Limited programming environment
  - Limited network protocol support

Azure Functions

heroku

- Infrastructure as a Service (IaaS)
  - User manages OS and software stack
  - Can be expensive to run

Microsoft Azure

Can we run LLMI applications in cloud efficiently?

This talk

- Goal: support LLMI applications in cloud environments
- Requirements:
  - Run wide variety of applications
  - Run efficiently so that we can dramatically lower cost
  - Be deployable in the cloud today
- Picocenter
  - Be able to run lots of LLMI applications in cloud
  - Swap idle application to cloud storage
  - Swap in application quickly when it is being requested

Related work

- Application running environment
  - Operating system containers   *BSD Jail*
  - Dedicated runtime  App Engine  *Embassies*
- Swapping
  - Pre-paging and migration *Virtual Machine migration* *Jettison*
  - Checkpoint and restore *BLCR* *DMTCP*
- Picocenter
  - First attempt to leverage them for LLMI apps running in the cloud

Outline

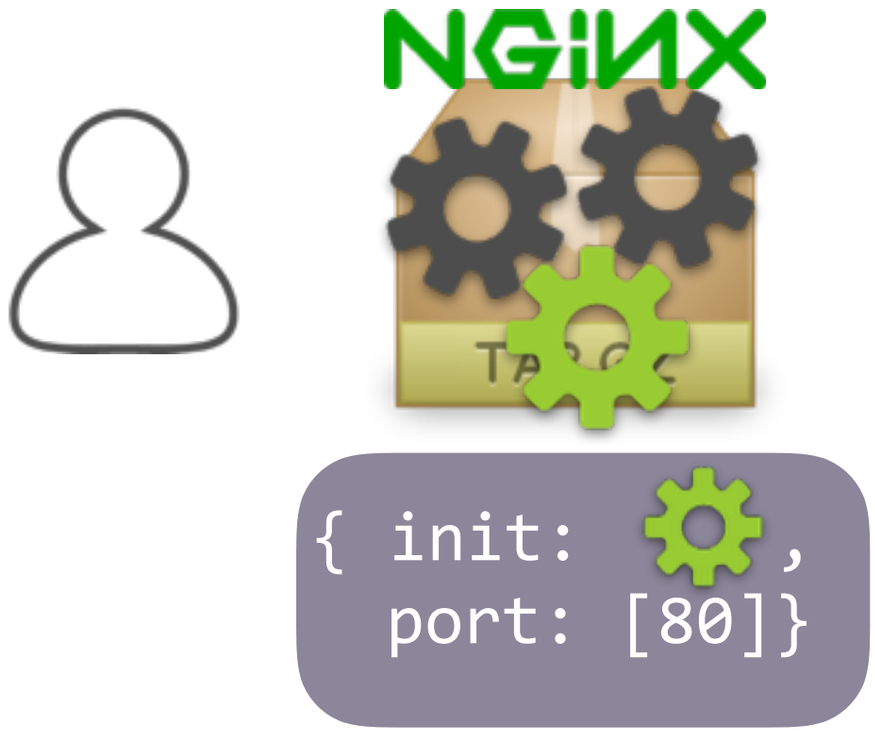
- ~~Introduction~~

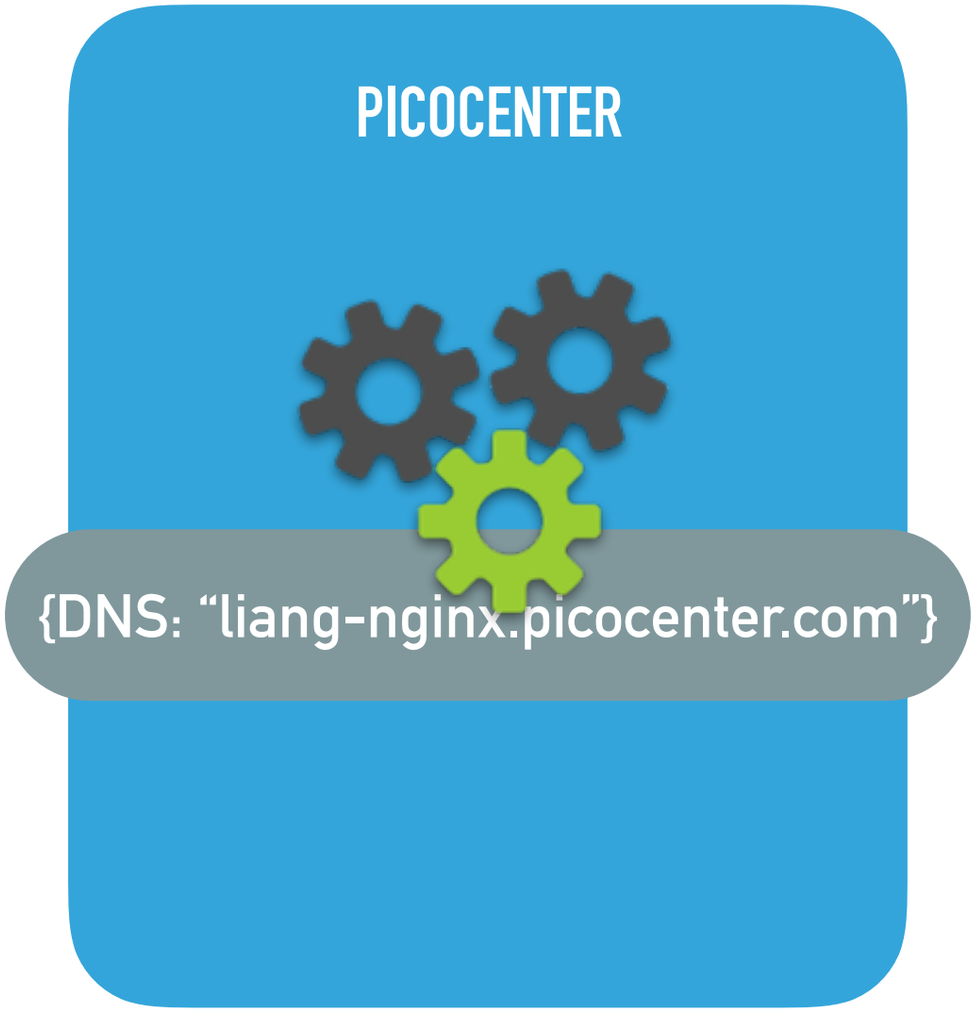
- Design

- Evaluation

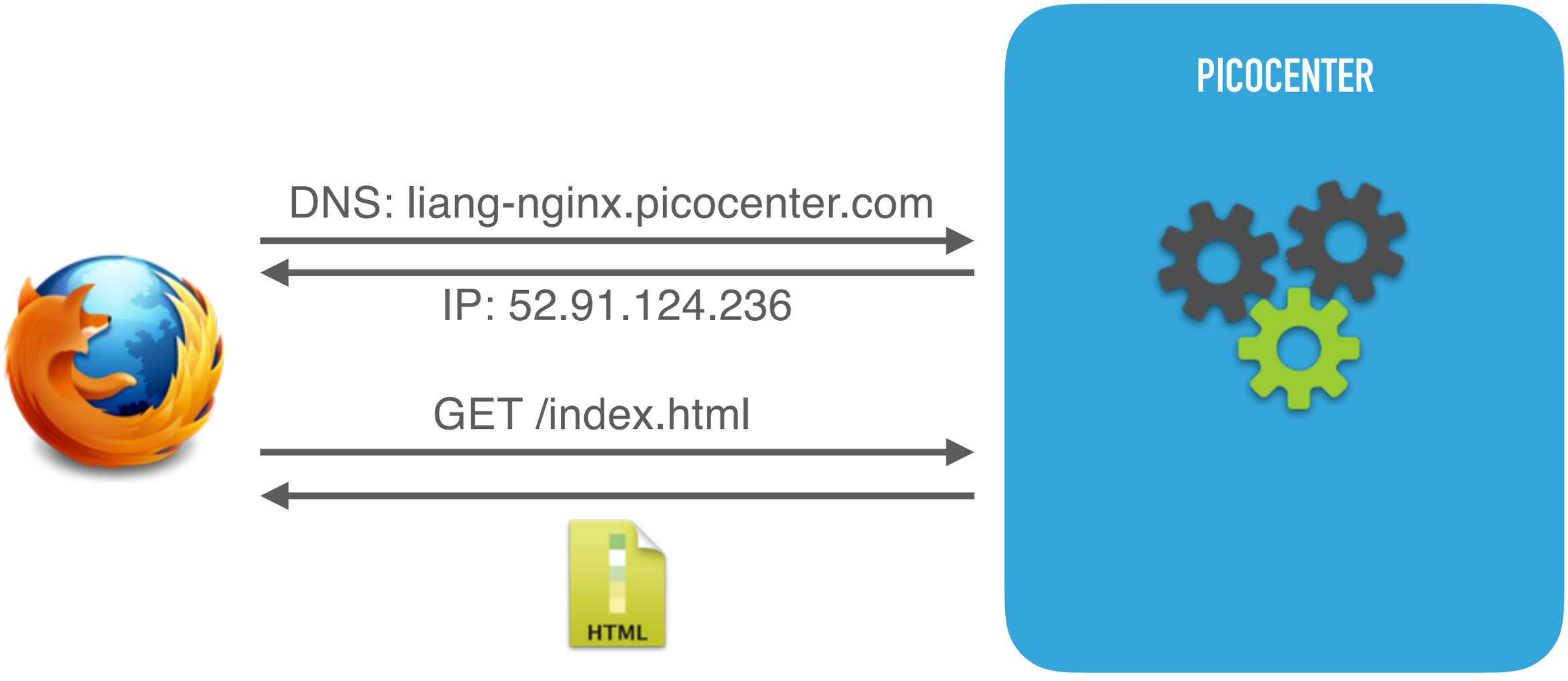
- Conclusion

Running LLMI applications in Picocenter

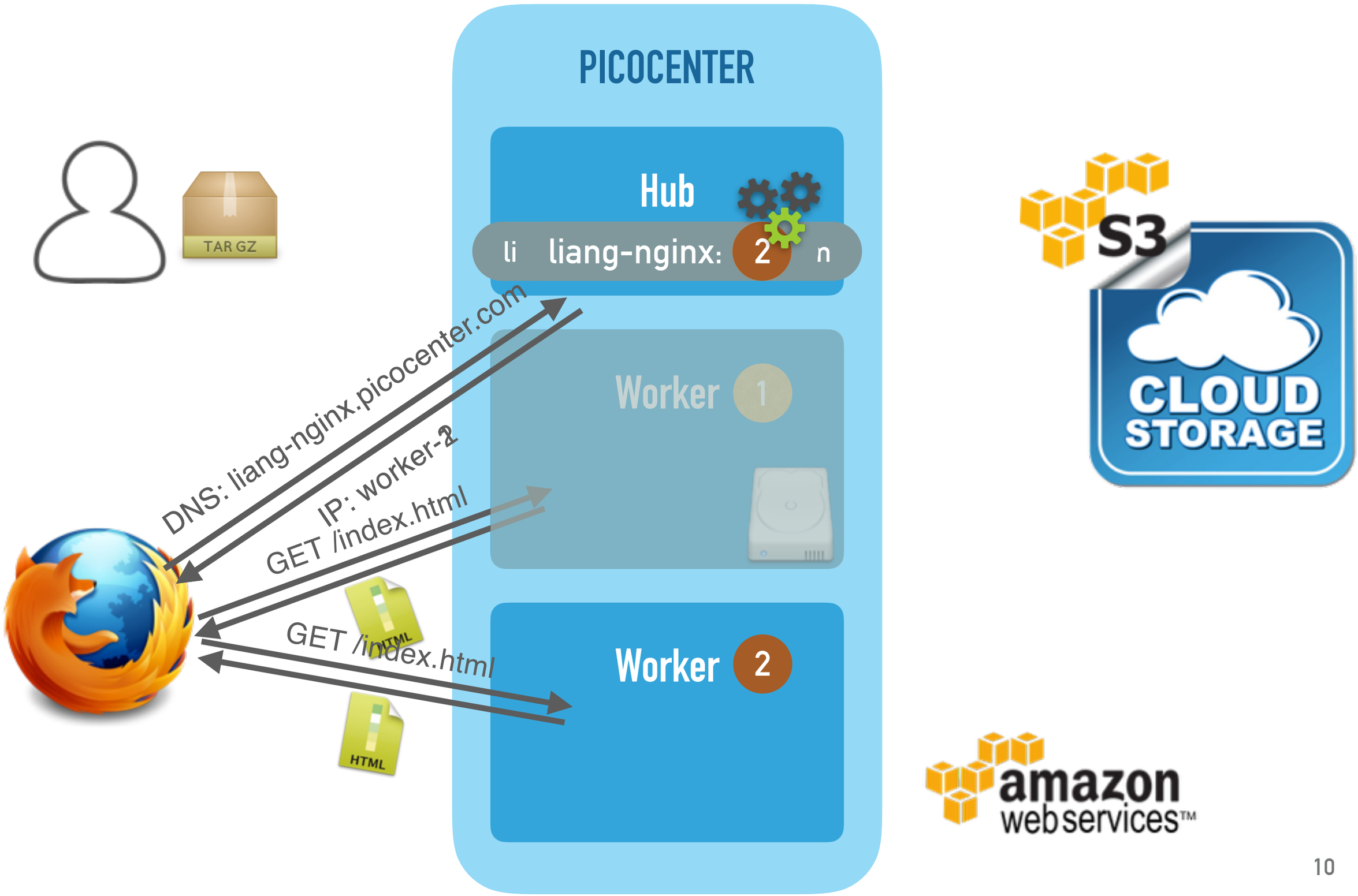




Running LLMI applications in Picocenter



Picocenter internals



Swapping strategies

Full checkpoint

Reactive page faulting

Slow start due to download of all pages	Only need page meta info to start
Fast processing time because all pages are fetched	Slow processing due to page fetching on page faults
Not all pages are necessary	Minimum pages for application processing

Can we combine the best of both strategies?

ActiveSet

- ActiveSet: Predict pages that are needed for the request
  - Reduce total download size
  - Minimize round trips of page faults

- Page prediction: most recently used
  - Future: prediction based on ports, ML on page faults

Implementation

- LLMI application runs in a process-like environment
  - Use Linux container (LXC)

- ActiveSet
  - Modified CRIU to map page to files
  - Catch page faults with FUSE

Outline

- ~~Introduction~~

- ~~Design~~

- Evaluation

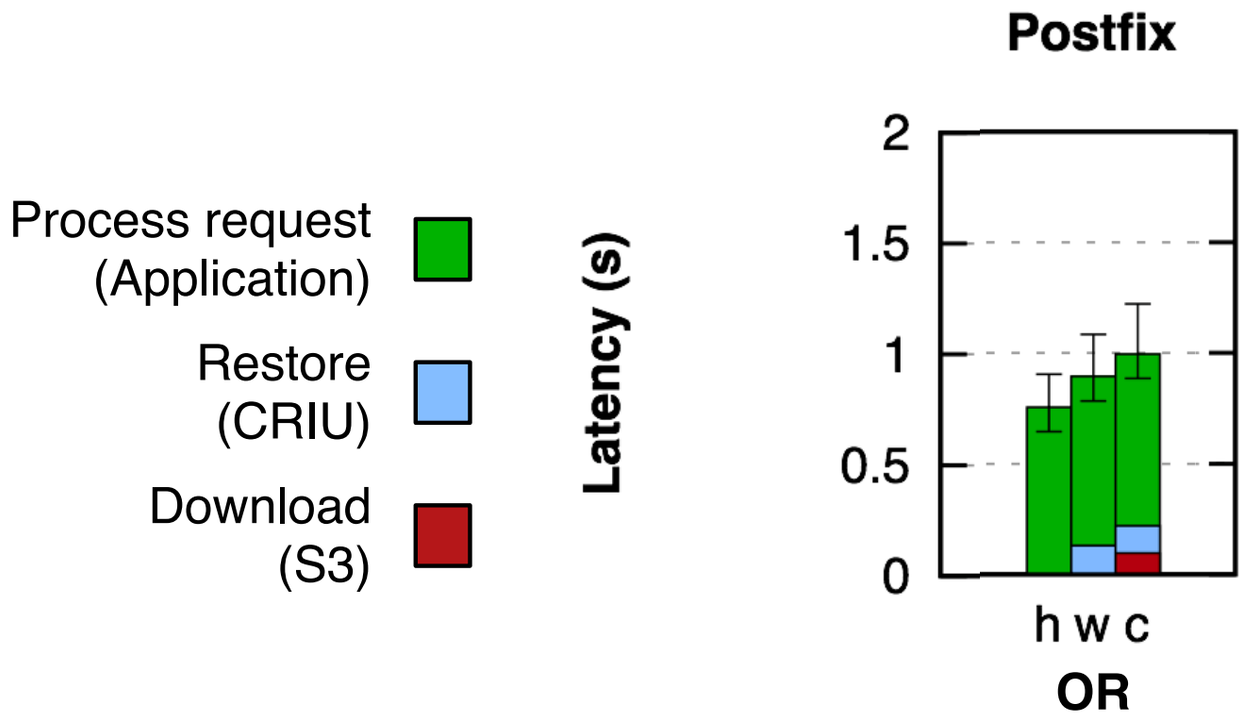
- Conclusion

Evaluation

- How quickly can Picocenter revive real-world processes from cloud storage?
- How does the ActiveSet technique help to reduce application reviving time?
- How does Picocenter perform with a challenging real-world application?
- What is the estimate cost of running applications in Picocenter

Please refer to the paper

How quickly can Picocenter revive real-world processes from cloud storage?

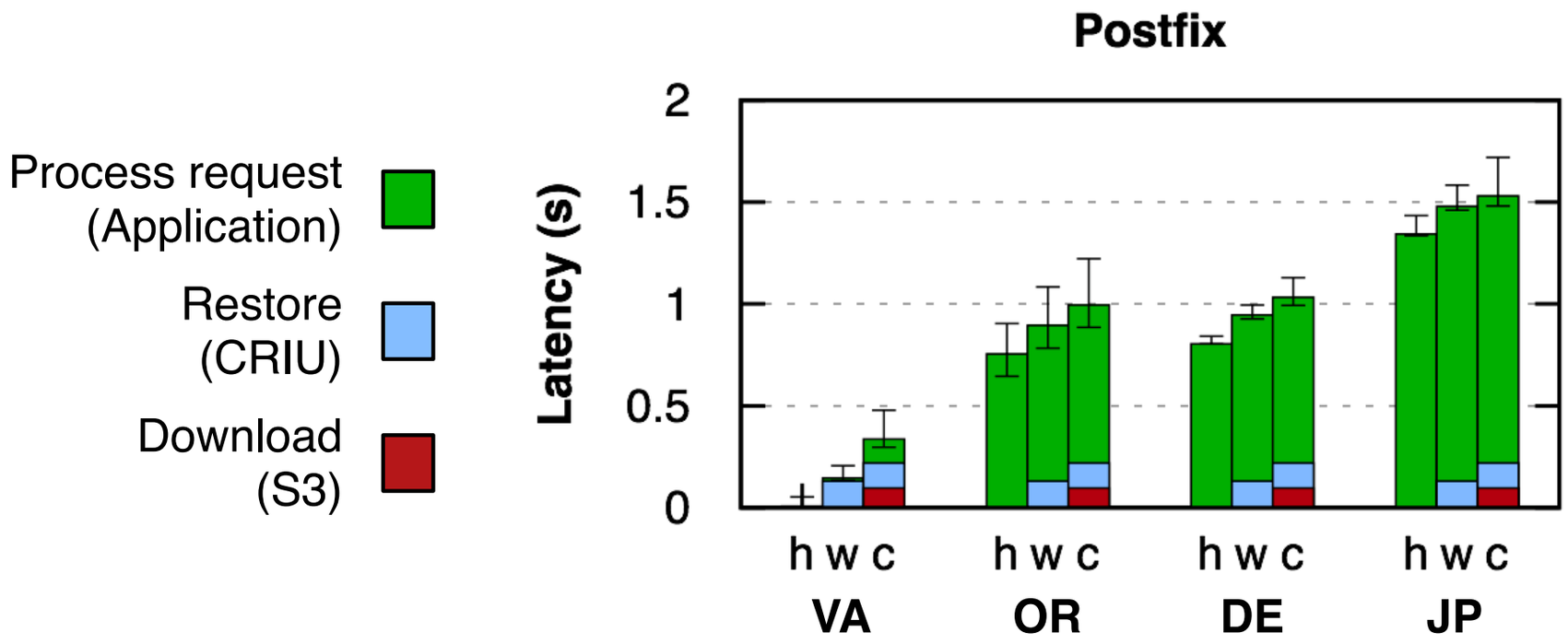


OR

- Host Picocenter with ActiveSet in Amazon Virginia (VA) datacenter
  - hot: application is alive; warm: swap in from disk; cold: swap in from cloud

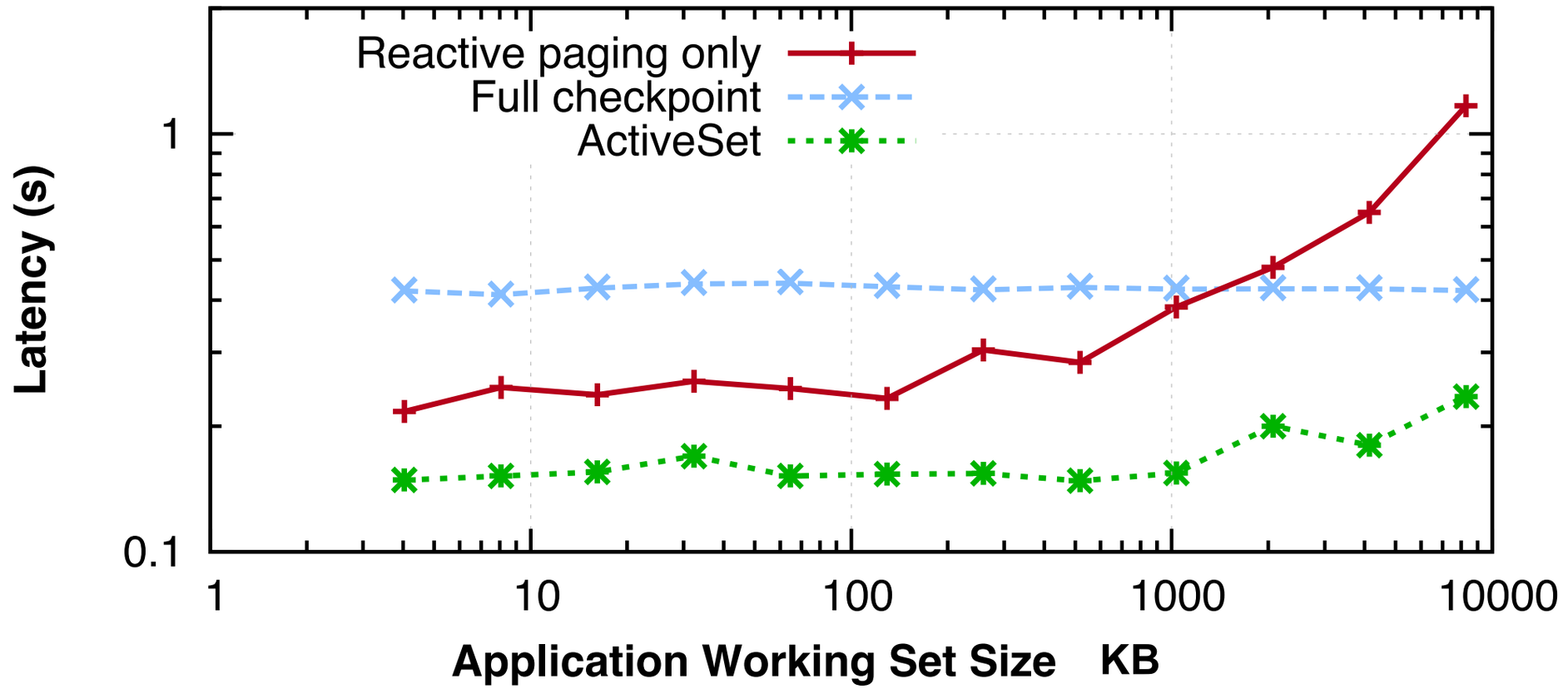
- Results
  - Restore overhead: ~120 ms for warm and ~220 ms for cold

How quickly can Picocenter revive real-world processes from cloud storage?



- Host Picocenter with ActiveSet in Amazon Virginia (VA) datacenter
  - hot: application is alive; warm: swap in from disk; cold: swap in from cloud
  - Client requests from Virginia (VA), Oregon (OR), Frankfurt (DE) or Tokyo (JP)
- Results
  - Restore overhead: ~120 ms for warm and ~220 ms for cold
  - Overhead can be dwarfed by the end-to-end performance of the protocol itself

How does the ActiveSet technique help to reduce application reviving time?



- Control experiment on ActiveSets
  - Total memory is configured to 64 MB
  - Vary the working set size between 4 KB and 8 MB
  - Download pages in blocks; each block has 32 pages
- ActiveSet technique significantly outperforms the baseline approaches

Conclusion

- Picocenter: a new approach for cloud computation
  - Support long-lived, mostly-idle (LLMI) applications
- Swap ilde application to cloud storage
- Provide process-like environment
- Swap in real world applications in under 250 ms
- Open source: https://github.com/leoliangzhang/Picocenter

Thank you!

Questions?

liang@ccs.neu.edu