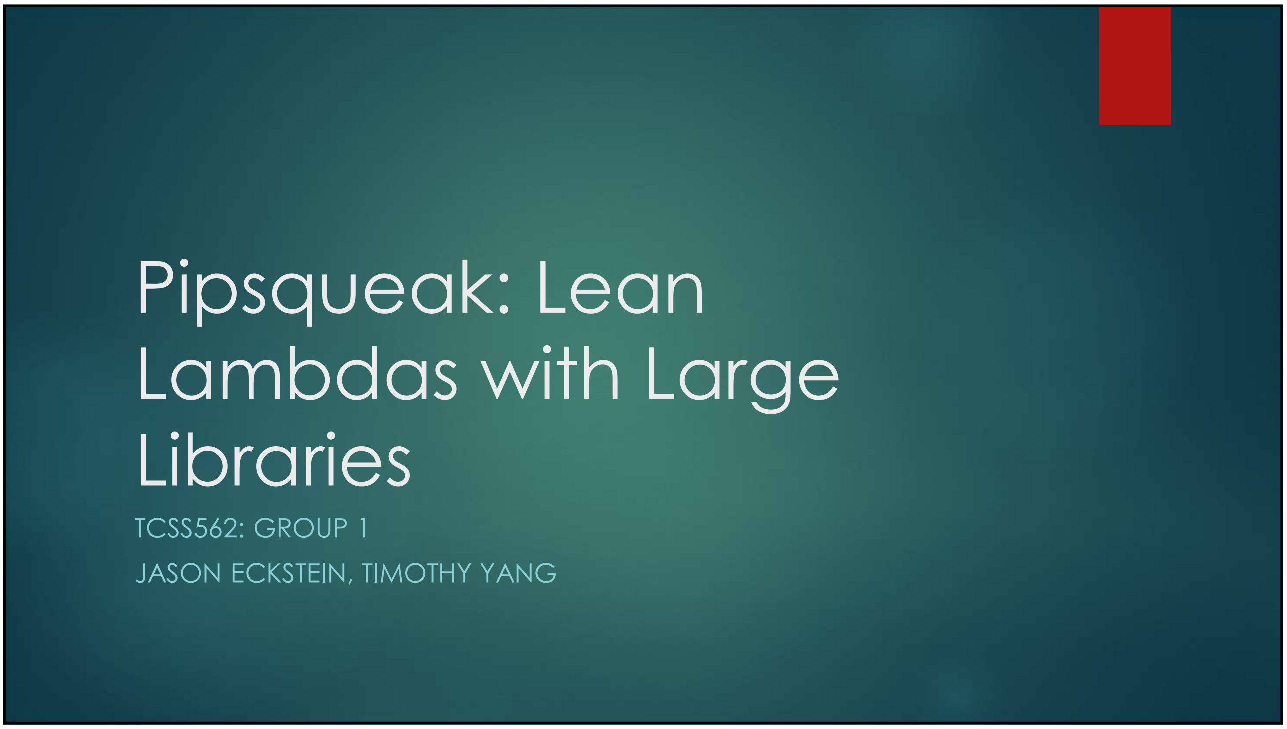
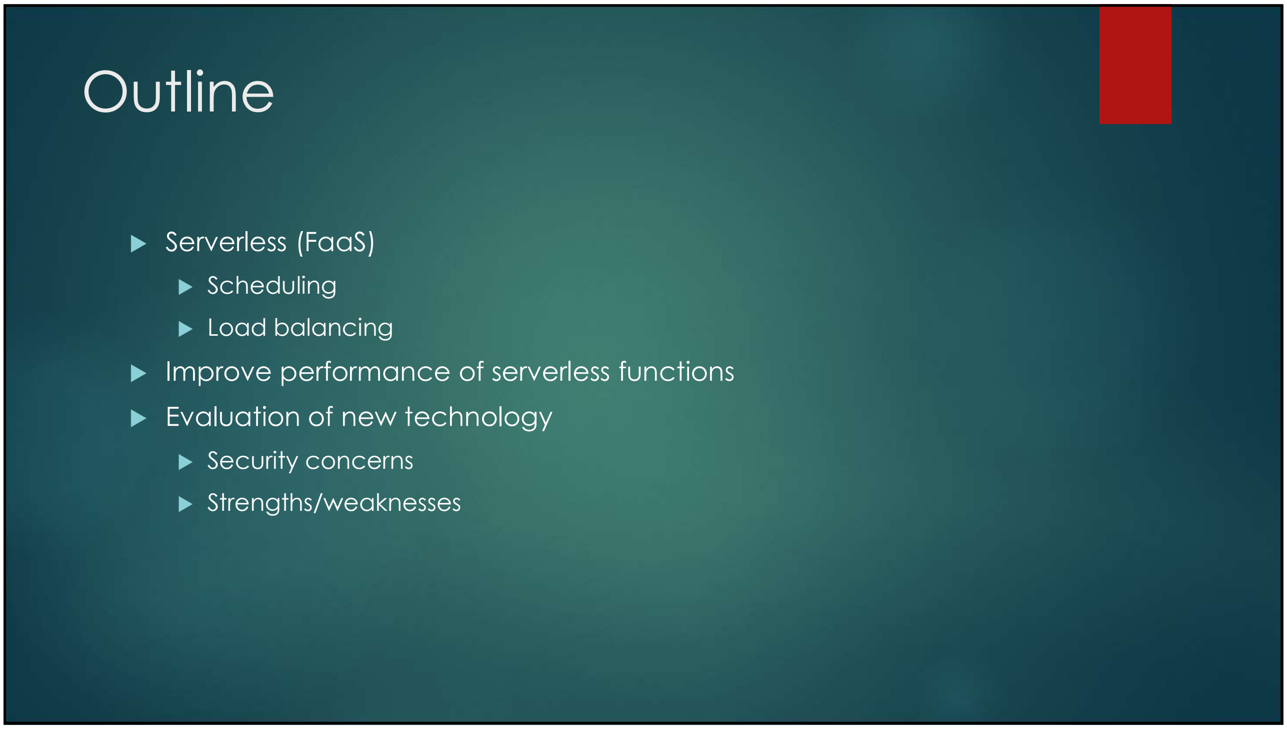


Pipsqueak: Lean Lambdas with Large Libraries

TCSS562: GROUP 1

JASON ECKSTEIN, TIMOTHY YANG

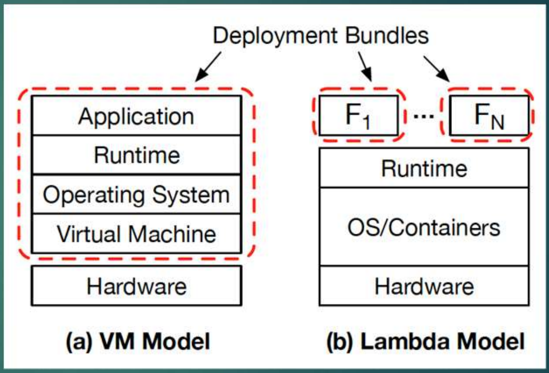


Outline

- Serverless (FaaS)
  - Scheduling
  - Load balancing
- Improve performance of serverless functions
- Evaluation of new technology
  - Security concerns
  - Strengths/weaknesses

Serverless

- Motivation
  - Scalability/elasticity
  - Performance
  - Reduce costs

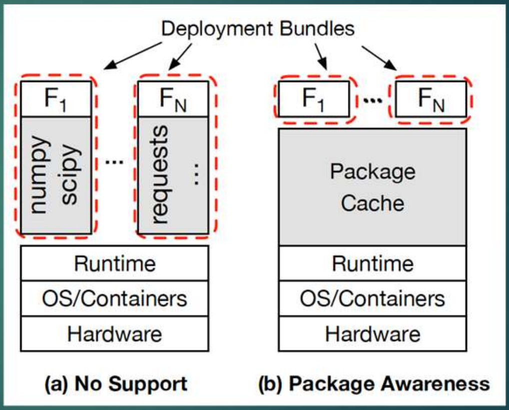


(a) VM Model

(b) Lambda Model

The Problem

- Lean Lambdas
  - Library dependencies
- Solution:
  - Rewrite old packages
  - split functionality of large dependencies
  - Caching



(a) No Support

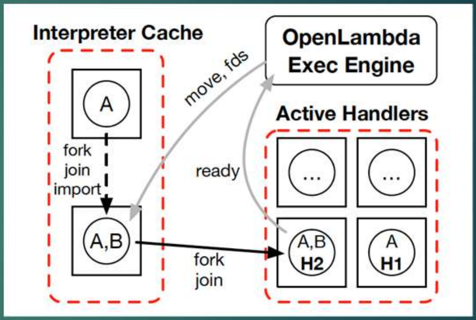
(b) Package Awareness

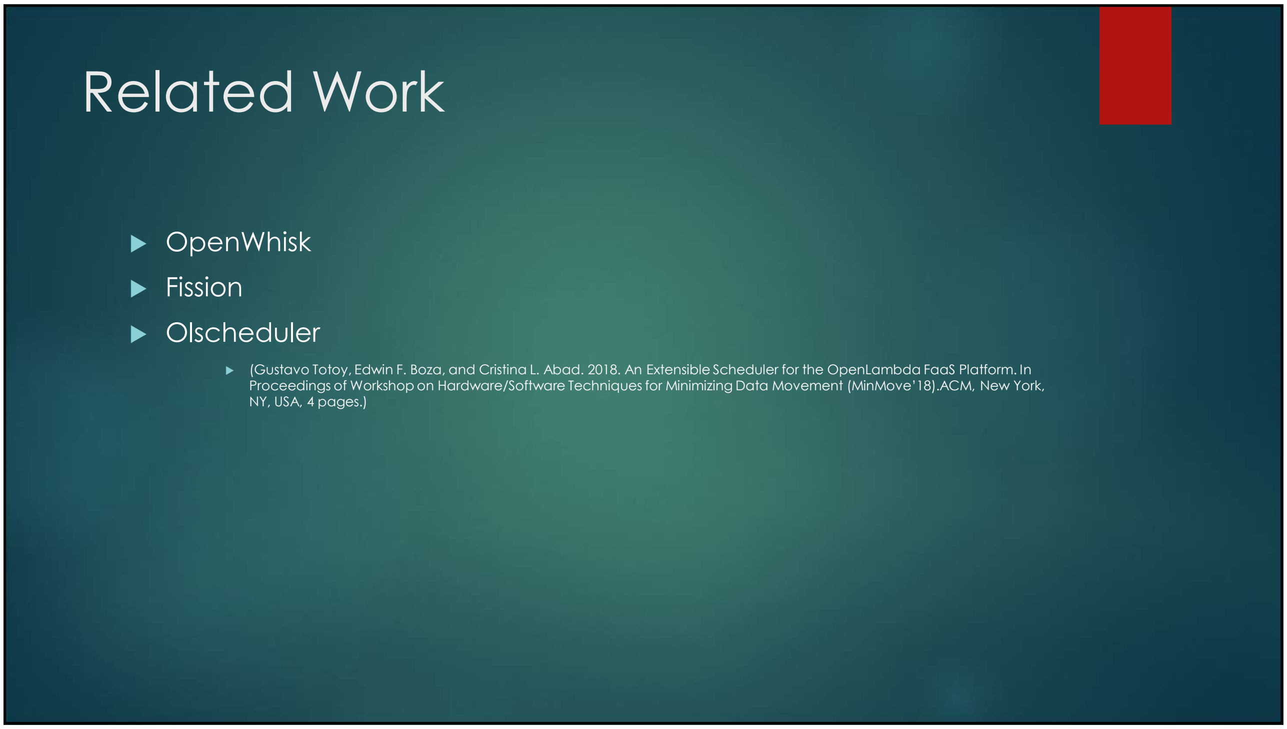
Background

- OpenLambda
  - Introduced in HotCloud '16
    - Co-located with USENIX Annual Technical Conference in Denver, Colorado
  - load balancing for function scheduling is performed by NGINX
    - Round Robin
    - Least-connected
    - IP-hash

Pipsqueak

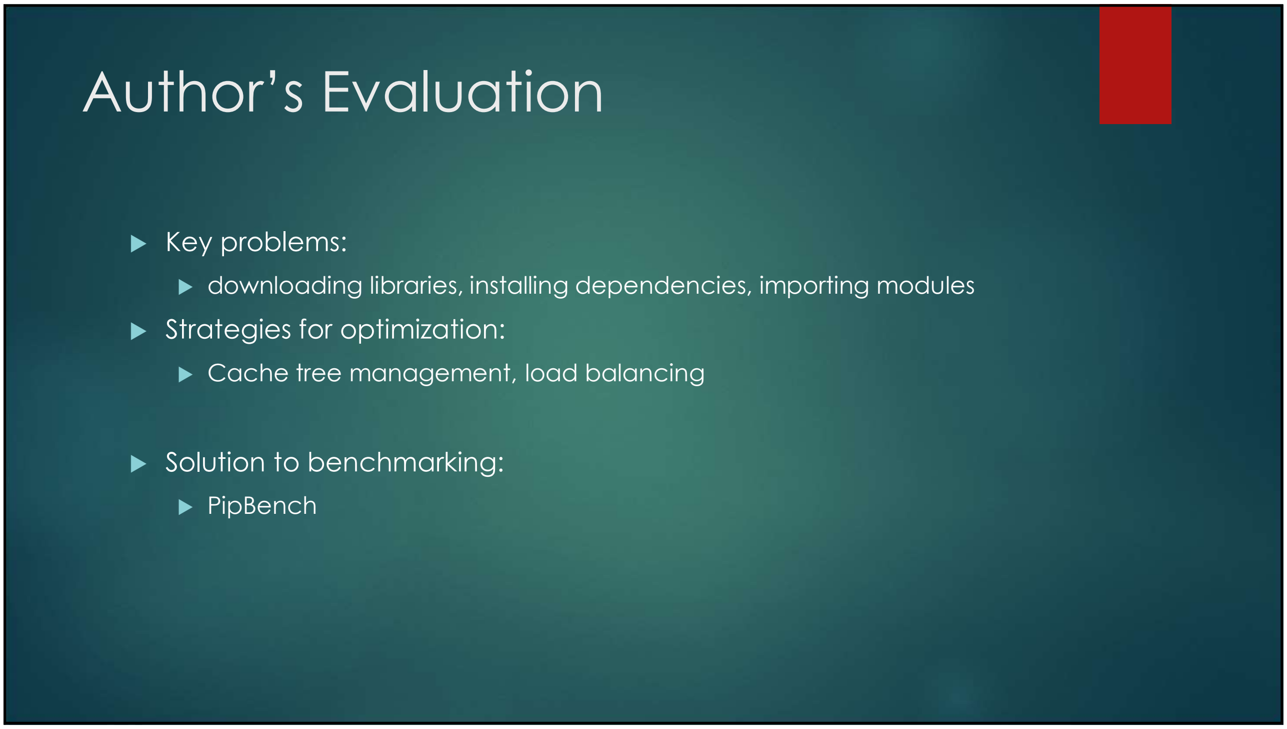
- Startup
  - Download, install, import
- Interpreter cache is a collection of paused processes
  - fork





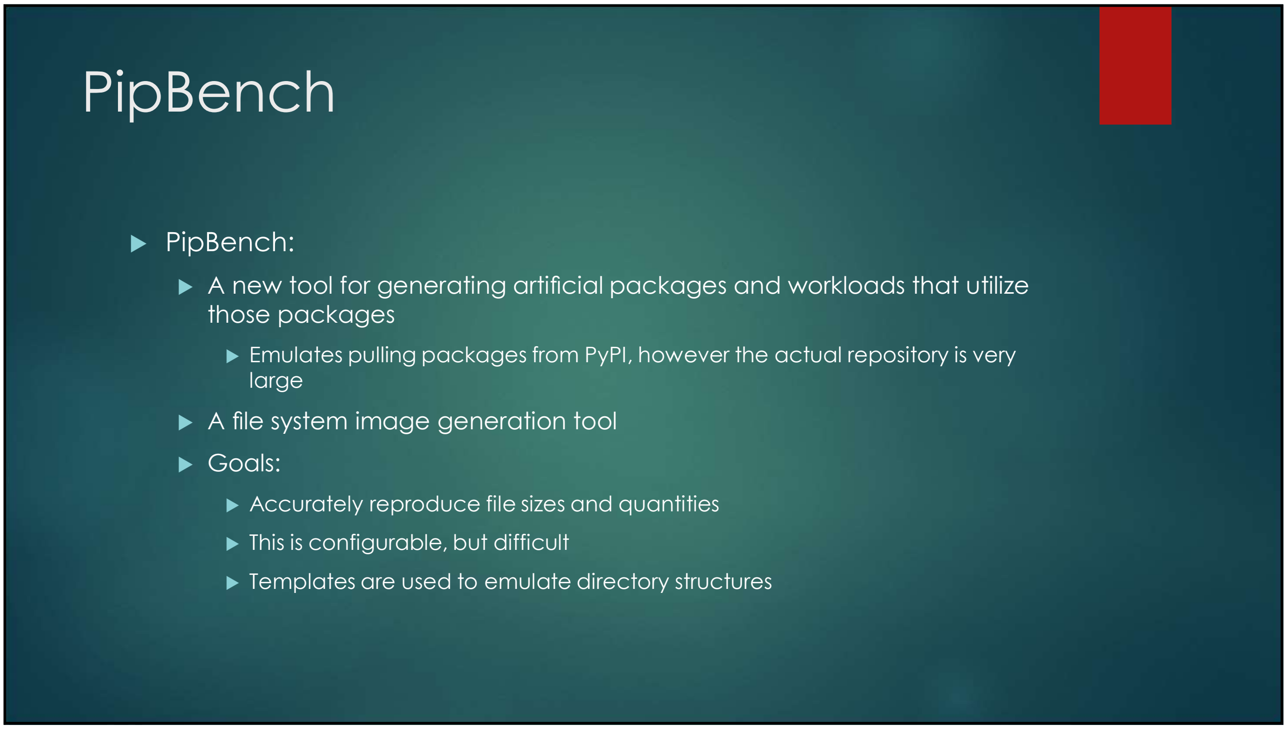
Related Work

- OpenWhisk
- Fission
- Olscheduler
  - (Gustavo Totoy, Edwin F. Boza, and Cristina L. Abad. 2018. An Extensible Scheduler for the OpenLambda FaaS Platform. In Proceedings of Workshop on Hardware/Software Techniques for Minimizing Data Movement (MinMove'18).ACM, New York, NY, USA, 4 pages.)



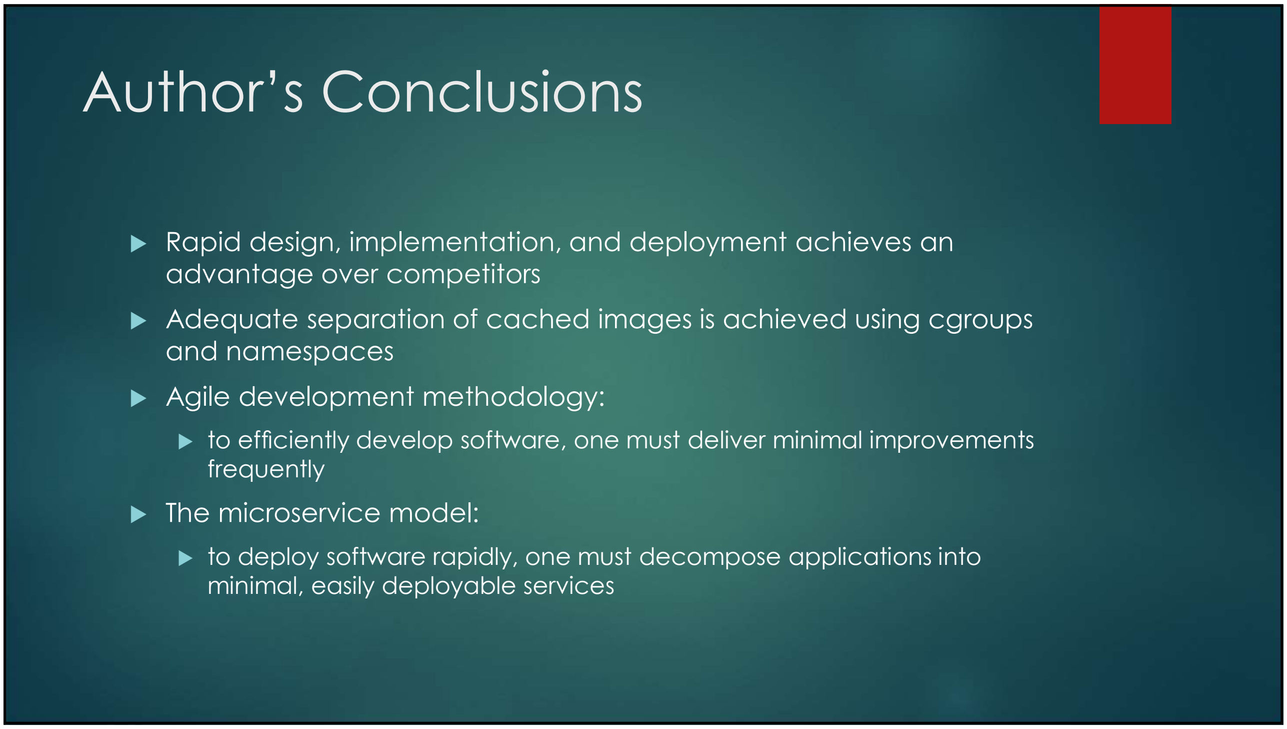
Author's Evaluation

- Key problems:
  - downloading libraries, installing dependencies, importing modules
- Strategies for optimization:
  - Cache tree management, load balancing
- Solution to benchmarking:
  - PipBench



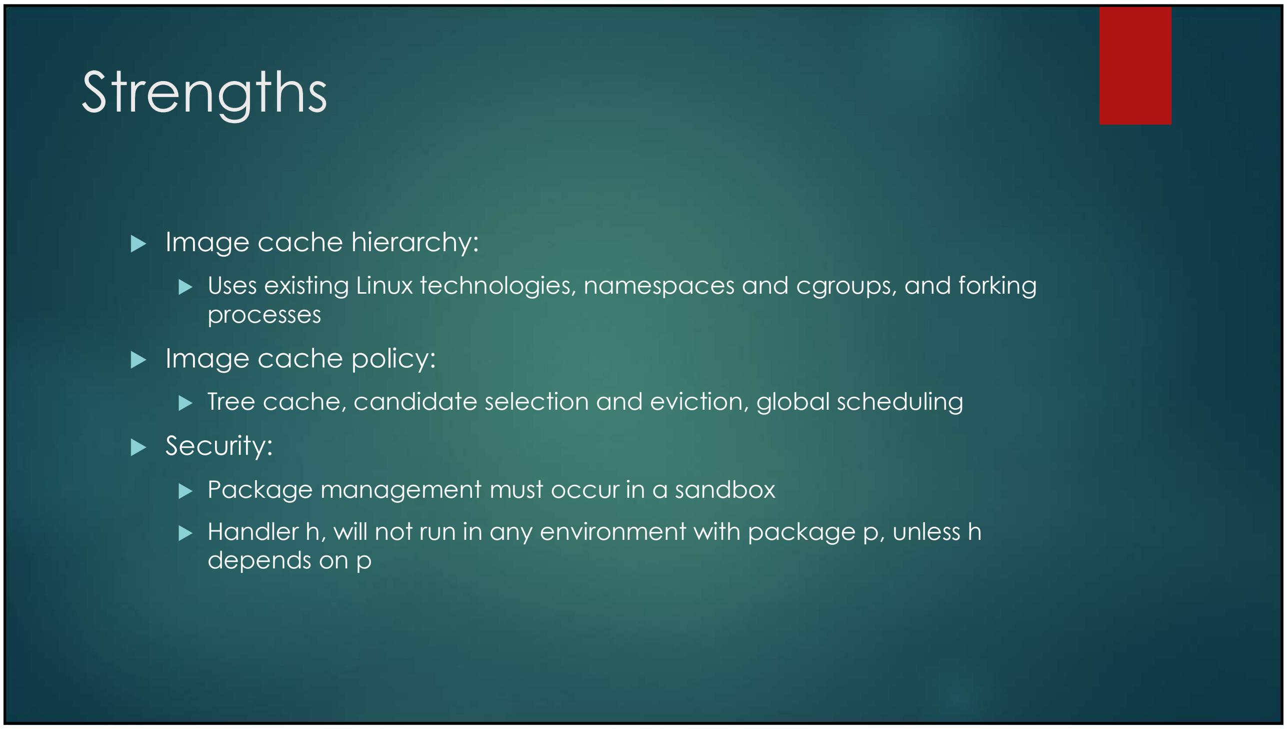
PipBench

- PipBench:
  - A new tool for generating artificial packages and workloads that utilize those packages
    - Emulates pulling packages from PyPI, however the actual repository is very large
  - A file system image generation tool
  - Goals:
    - Accurately reproduce file sizes and quantities
    - This is configurable, but difficult
    - Templates are used to emulate directory structures



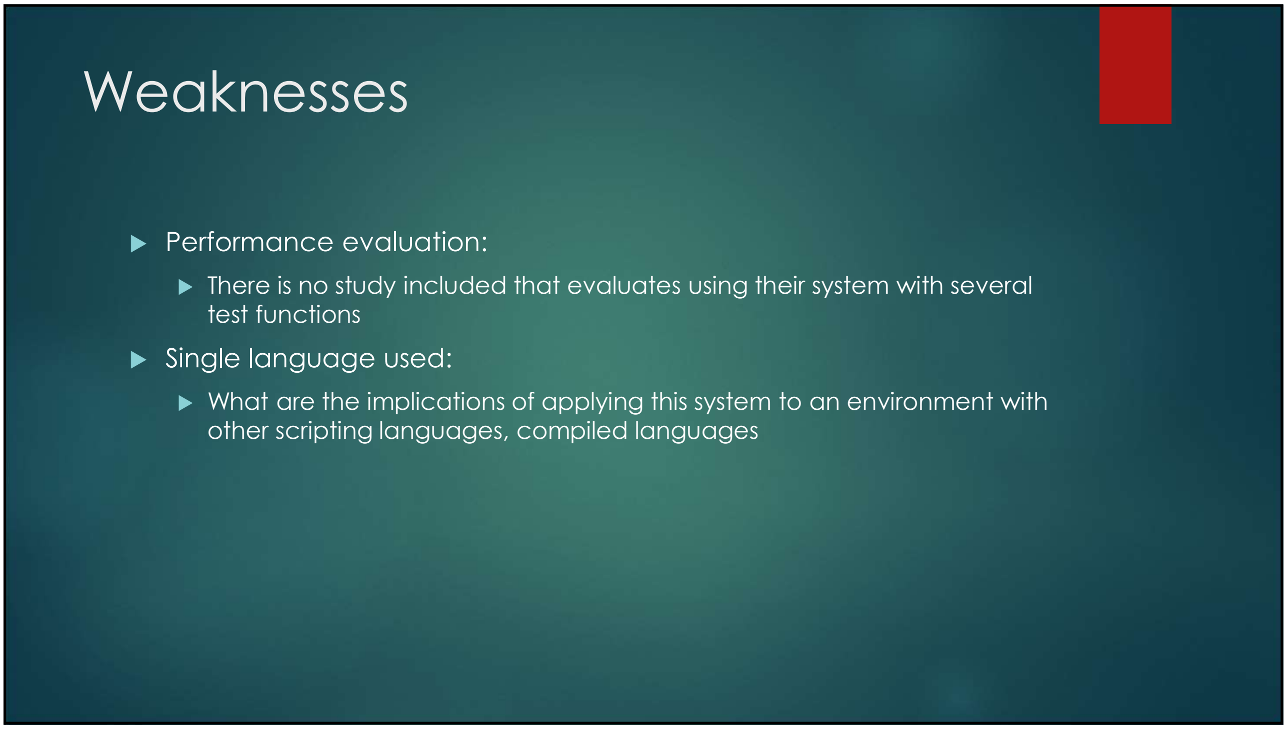
Author's Conclusions

- Rapid design, implementation, and deployment achieves an advantage over competitors
- Adequate separation of cached images is achieved using cgroups and namespaces
- Agile development methodology:
  - to efficiently develop software, one must deliver minimal improvements frequently
- The microservice model:
  - to deploy software rapidly, one must decompose applications into minimal, easily deployable services



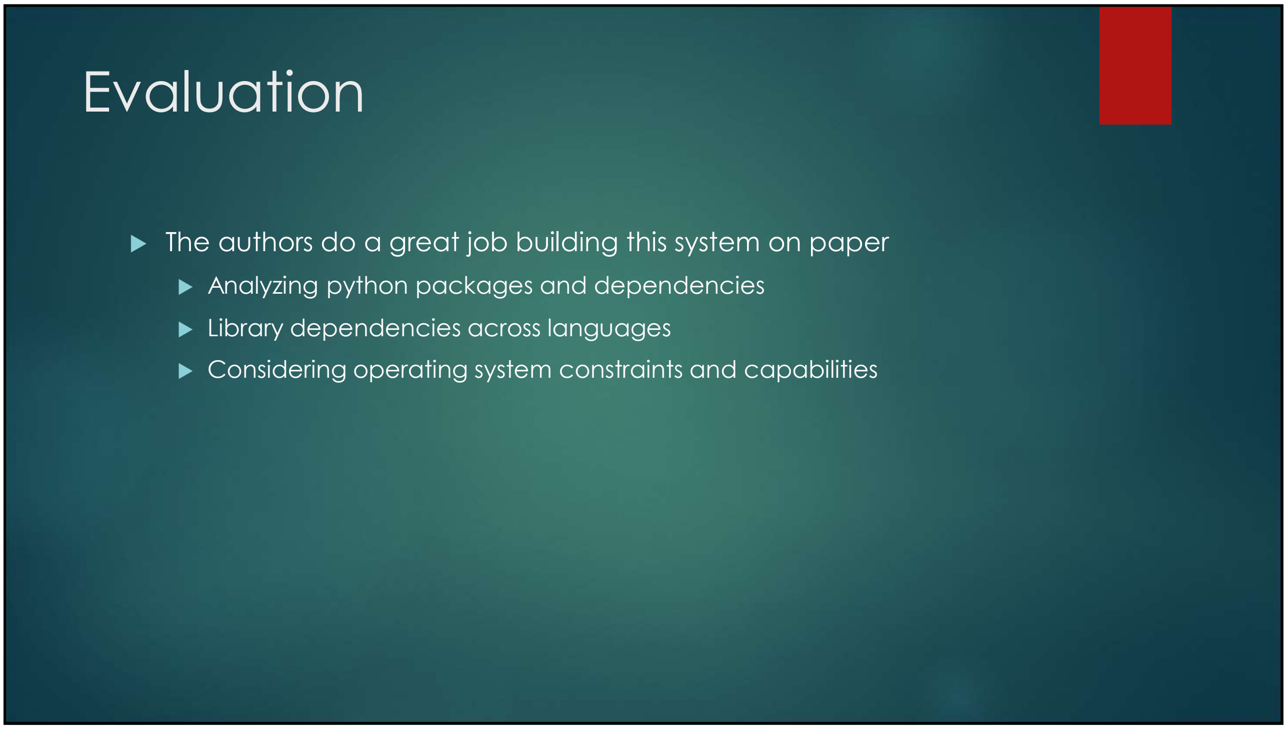
Strengths

- Image cache hierarchy:
  - Uses existing Linux technologies, namespaces and cgroups, and forking processes
- Image cache policy:
  - Tree cache, candidate selection and eviction, global scheduling
- Security:
  - Package management must occur in a sandbox
  - Handler h, will not run in any environment with package p, unless h depends on p



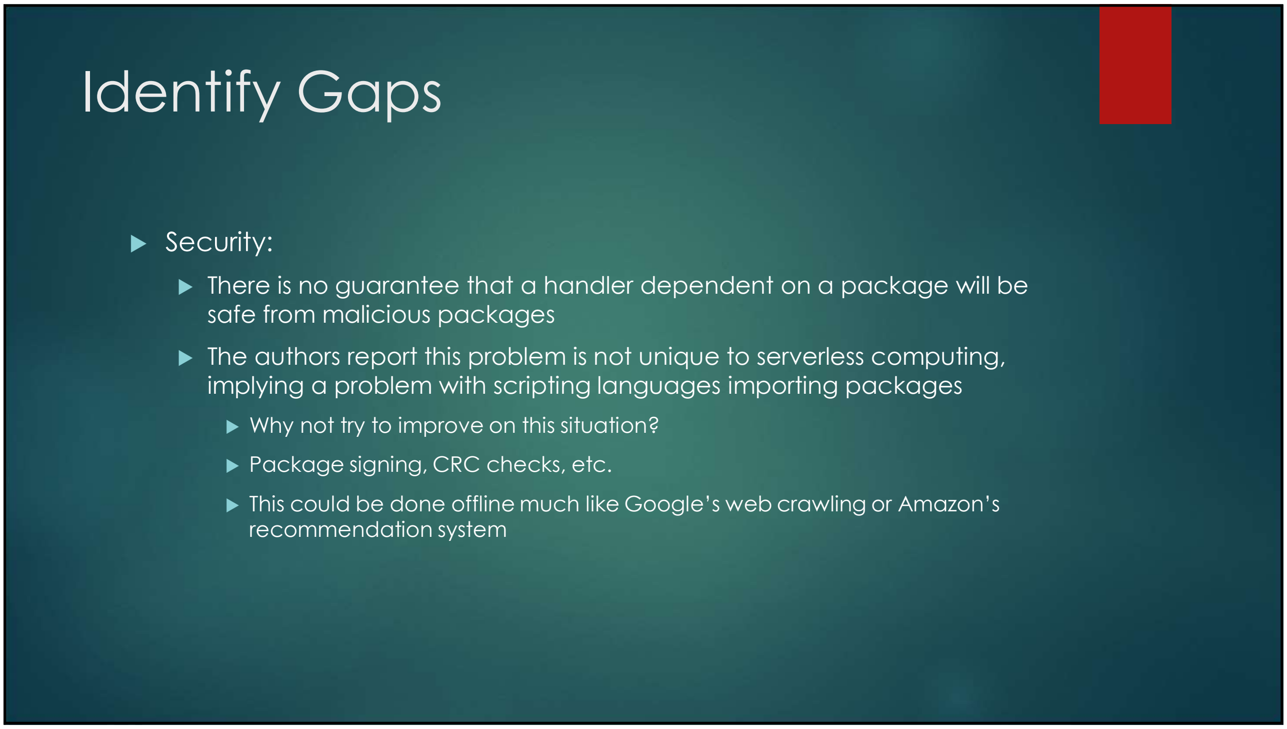
Weaknesses

- Performance evaluation:
  - There is no study included that evaluates using their system with several test functions
- Single language used:
  - What are the implications of applying this system to an environment with other scripting languages, compiled languages



Evaluation

- The authors do a great job building this system on paper
  - Analyzing python packages and dependencies
  - Library dependencies across languages
  - Considering operating system constraints and capabilities



Identify Gaps

- Security:
  - There is no guarantee that a handler dependent on a package will be safe from malicious packages
  - The authors report this problem is not unique to serverless computing, implying a problem with scripting languages importing packages
    - Why not try to improve on this situation?
    - Package signing, CRC checks, etc.
    - This could be done offline much like Google's web crawling or Amazon's recommendation system

Future Work

- Current technologies used: OpenLambda, Linux, Python
  - Integrate support for other scripting languages: Ruby, Node.js, etc.
  - Implications for running on other platforms such as AWS
    - Build using IaaS to explore running on Linux, as well as Windows
  - Add support for compiled languages such as C/C++

Questions

TCSS562: GROUP 1

JASON ECKSTEIN, TIMOTHY YANG