Missive: Fast Application Launch from an Untrusted Buffer Cache

Jon Howell
Jeremy Elson
Bryan Parno
John R. Douceur

Microsoft Research
Installing things has gotten so fast and painless.

Why not skip it entirely, and make a phone that has every app "installed" already and just downloads and runs them on the fly?

I felt pretty clever until I realized I'd invented webpages.
Autonomy leads to painlessness
Client installations are not autonomous
Embassies:
autonomy on the client
Minimality leads to autonomy

- picoprocess:
  - memory allocation in and scheduling
- all communications
- crypto primitives (randomness)
- UI is pixel blitting
Consequences of minimality

• Apps are big.
  • browser
  • rendering libraries
  • OS, filesystem

• No shared, trusted buffer cache
WHAT IF I TOLD YOU
YOU COULD LAUNCH AN APP
IN 100ms?
Why will it work?

- Commonality is available
- We can exploit it
  - to reduce network costs
  - to minimize local startup latency
- Proof of concept: cost is around ~100ms
100MiB apps have commonality

• Servers *could* run anything...
  but a few programs serve each function
  • OpenSSL, PolarSSL, Windows SSL
  • postfix, qmail, exim

• Embassies clients *could* run anything...
  • 100 “best-of” interactive desktop apps
100MiB apps have commonality

100 Posix Packages—All Dependencies

Cum. frac. of zarrfiles

Cache Size 99
Cache Size 50
Cache Size 25
Cache Size 12
Cache Size 6
Cache Size 0

128 KiB 256 KiB 512 KiB 1 MiB 2 MiB 4 MiB 8 MiB 16 MiB 32 MiB 64 MiB 128 MiB 256 MiB 512 MiB 1 GiB
100MiB apps have commonality
Fast app launch

• Exploiting commonality to save network bandwidth
• Exploiting local commonality at low latency
App launch in Embassies

1. fetch boot block
2. instantiate
3. check signature
4. create process B
5. start process at boot block
6. request file Hash(ImageB)
7. acquire metadata
8. load common content
9. fetch novel content
10. assemble image
11. map image into App B
12. verify image
13. execute image
Zarfile structure

- Blocks vs. strings
- Block size
  - Small blocks: big recipes
  - Big blocks: extra padding
- Small-file packing
- Merkle tree degree
- File placement with consistent hashing
Small file packing
Small file packing

- Most files are less than 2-8 KB
- Most bytes are in files greater than 1-9 MB

- Blue line: Posix-complete
- Red line: Embassies-initial
- Green line: Posix-initial
Small file packing
Zarfile structure

• Blocks vs. strings
• Block size
  • Small blocks: big recipes
  • Big blocks: extra padding
• Small-file packing
• Merkle tree degree
• File placement with consistent hashing
Fast verification

• SHA-1: 390ms  (100MiB)
• VMAC:  29ms
  • depends on a secret
  • how can boot block have a secret?
Summary

• Client apps exhibit commonality
• Untrusted cache costs ~100ms
• We really can deliver 100MB apps on the fly