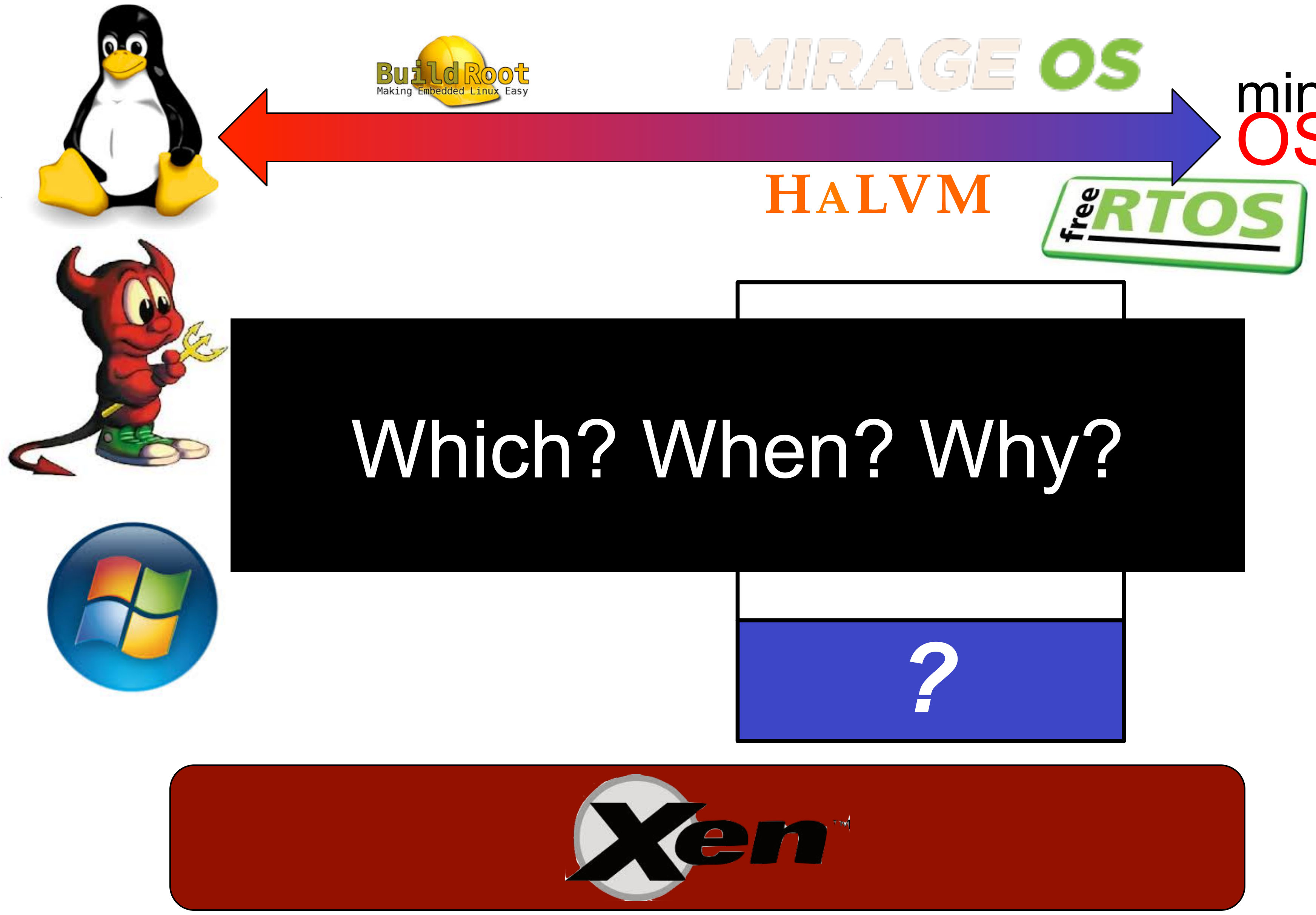
Unikernels: Who, What, Where, When, Why?

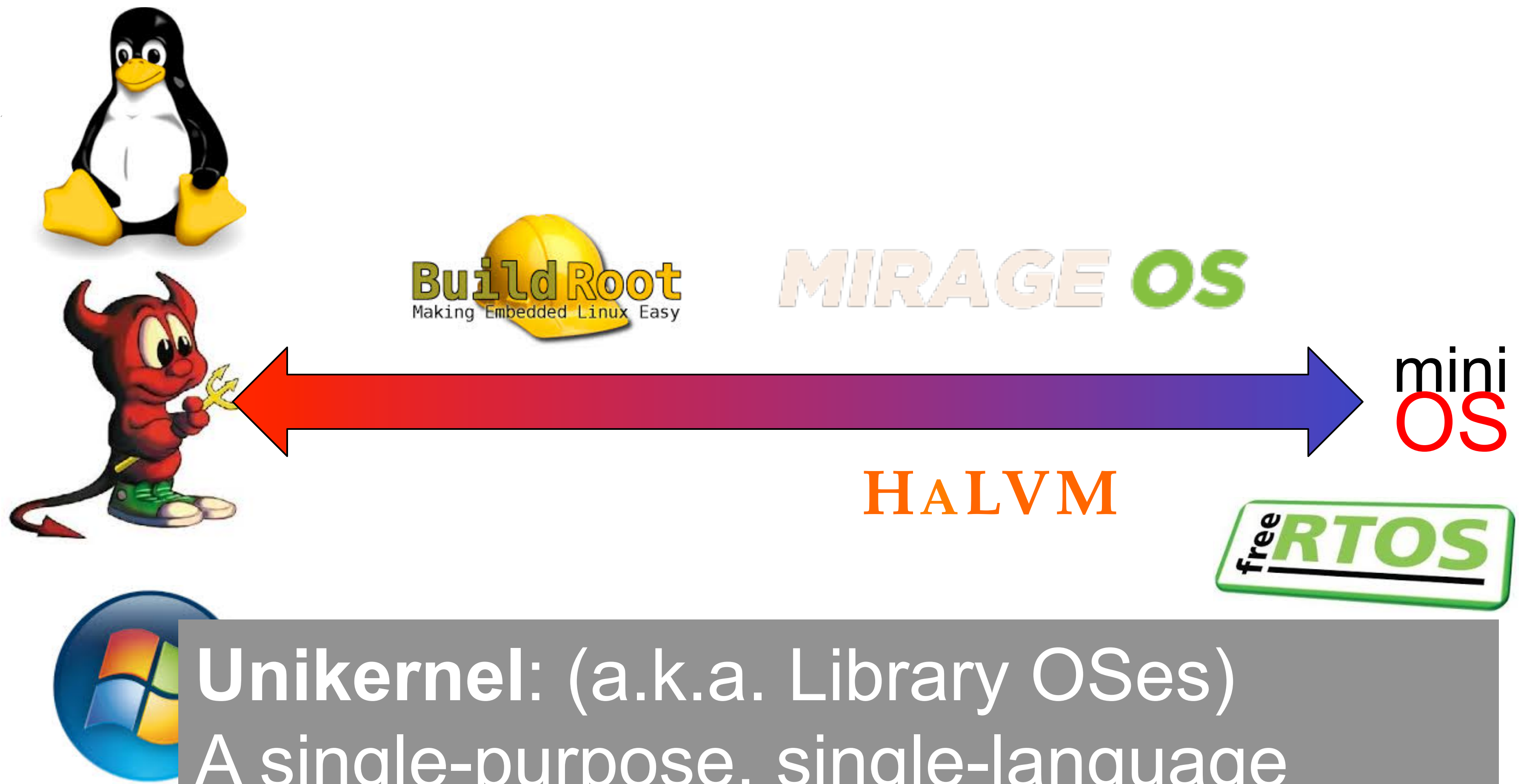
Adam Wick (awick@galois.com)

QCon SF | November 4th, 2014

|galois|



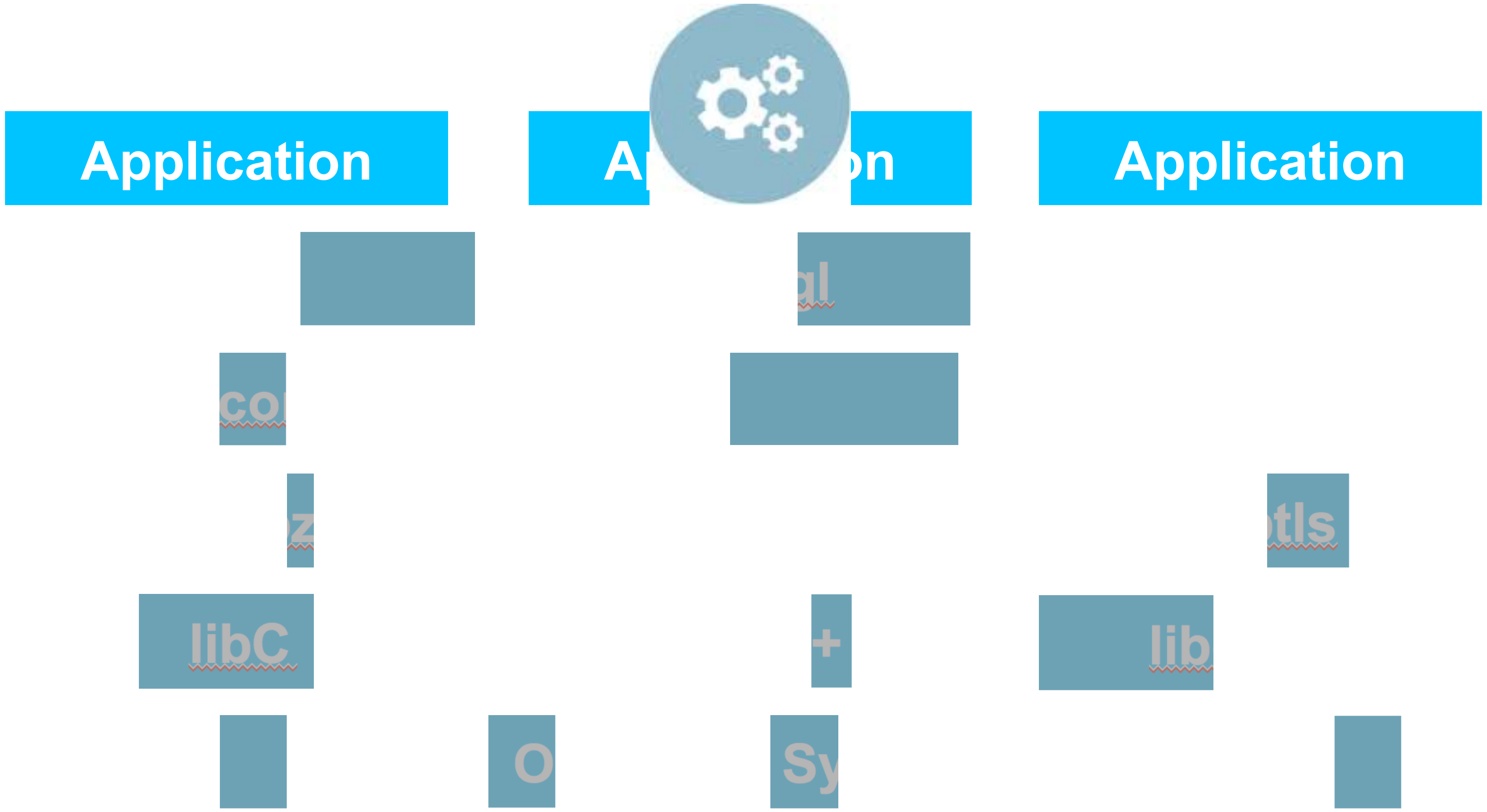
Which? When? Why?



Unikernel: (a.k.a. Library OSes)
A single-purpose, single-language virtual machine hosted on a minimal environment.

Unikernels: Who, ~~What,~~ Where, When, Why?

- The advantages and disadvantages of unikernels.
- Where Galois has used them in the past that worked.
  - … and didn't work.
- What general rules we think apply.


Application
Ap
on
Application
gl
co
z
tls
libC
+
lib
O
Sy

Why?

Why?

- Reduced memory footprint.
- Greatly reduced need for disk space.
- Reduced computational burden.

Use less powerful VM classes for the same work, and save money.

Why?

- Reduced memory footprint.
- No extraneous processes taking up your CPU.
- Fewer schedulers interrupting things.

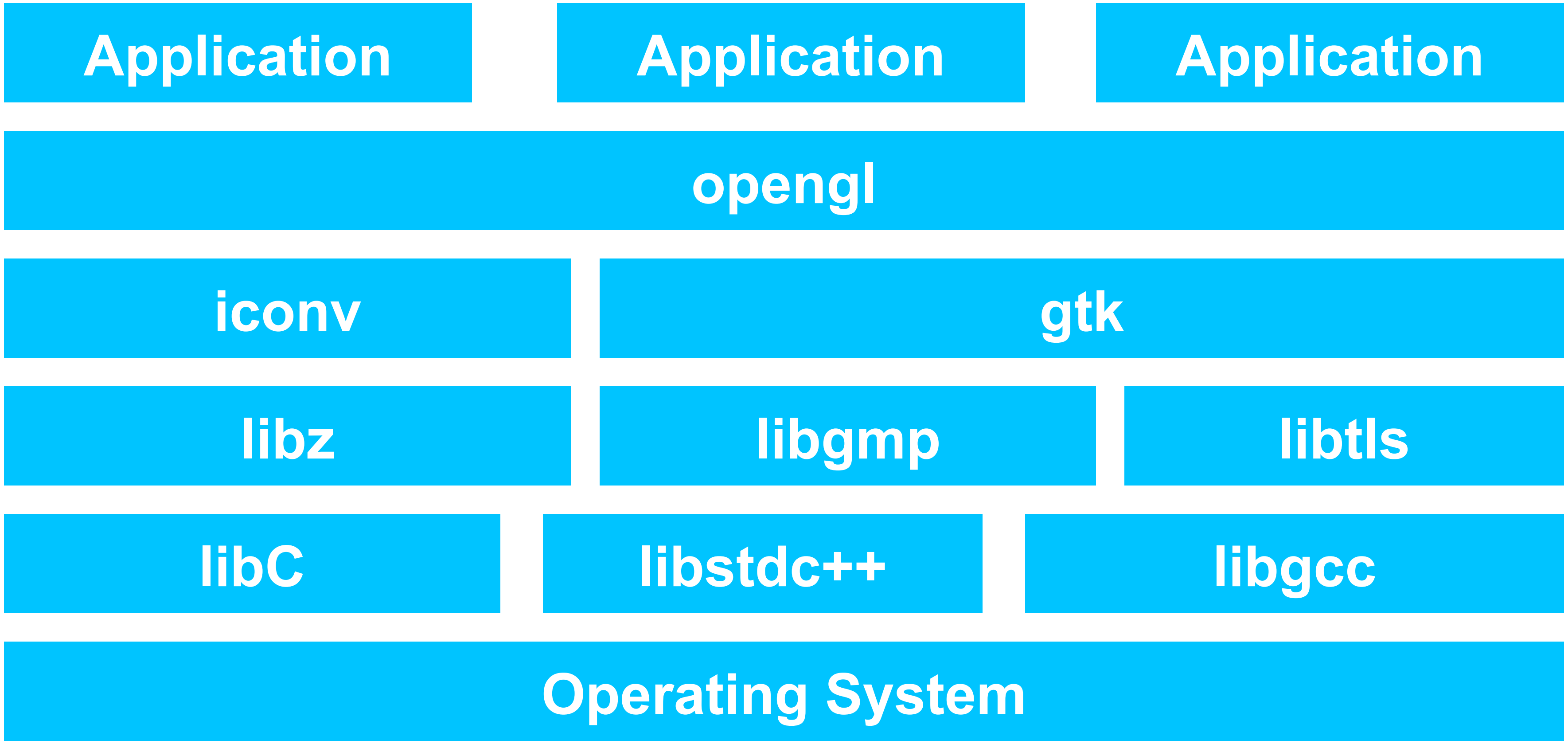
Faster load times, lower latencies.

Why?

- Reduced code size.
- Customized to application.
- (Potentially) Stronger walls between disparate components.

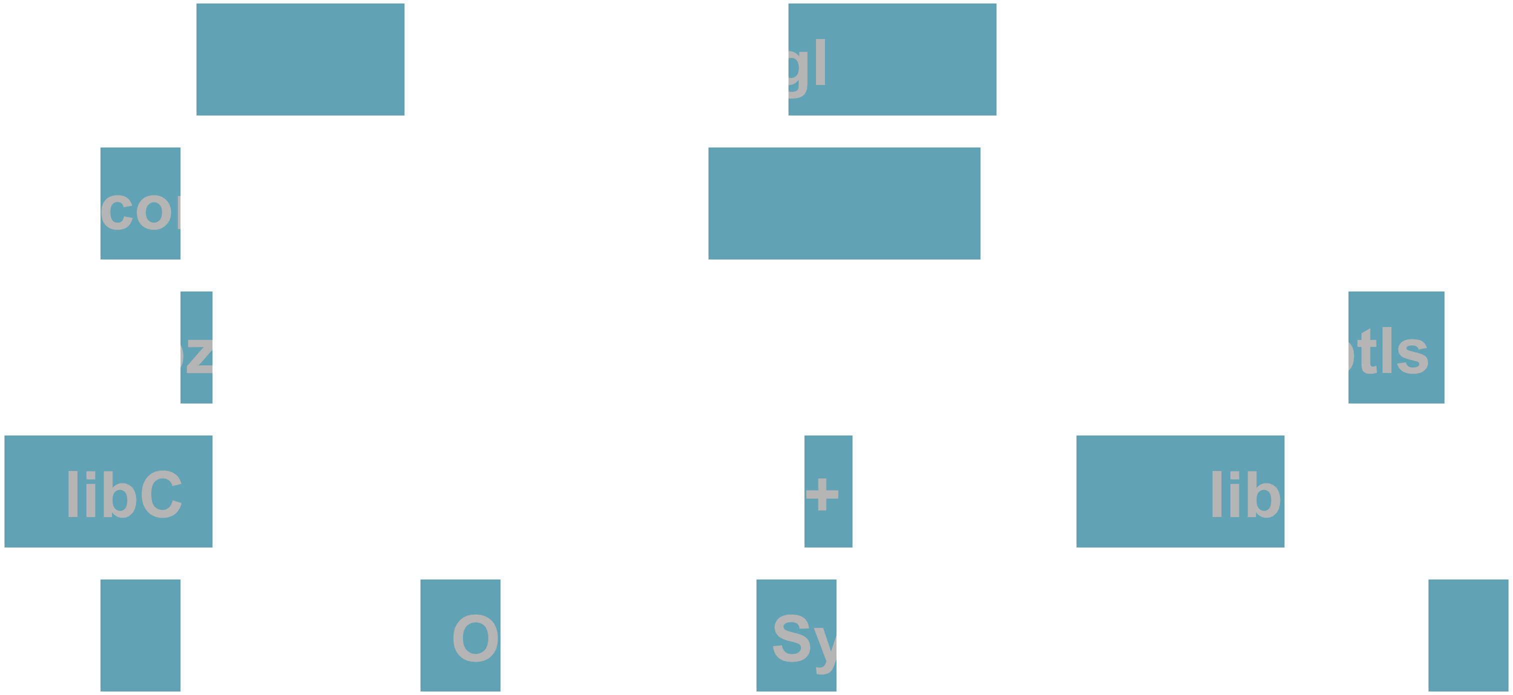
Less exposure to general attacks, reduced privileges, reduced attack surface.

Why not?



If this is what you want, don’t fix what isn’t broken.

Why not?



There is a lot of software for Linux; with a unikernel, you will end up writing these bits and pieces.

Why not?

* These savings come from avoiding some expenses: removing the need for disks, lowering processor costs.

If your application needs them anyways, you're not going to see any savings.

Unikernels:
Who, ~~What,~~ Where, When, Why?

- ~~The advantages and disadvantages of unikernels.~~
- Where Galois has used them in the past that worked.
    - … and didn't work.
- What general rules we think apply.

Mission:
To create trustworthiness in critical systems.

Research services and prototype development in computer science.

Security	Operating Systems
Safety	Networking Cryptography
Privacy	Scientific Computing
Reliability	Human/Computer Interaction
Predictability	Programming Lanuages
Integrity	Formal Methods

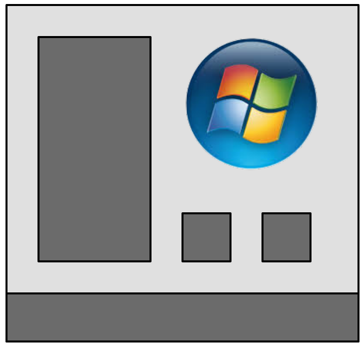
We built the HaLVM as part of one of our projects, and have successfully used it on many more since then. What have we learned?

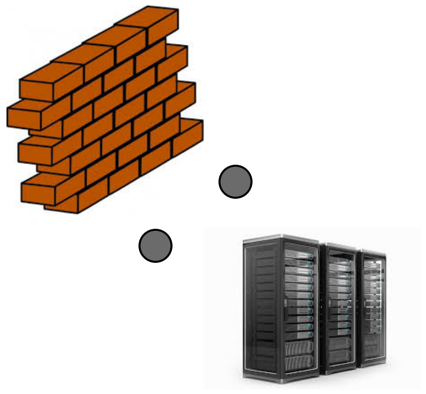
Three Main Use Cases

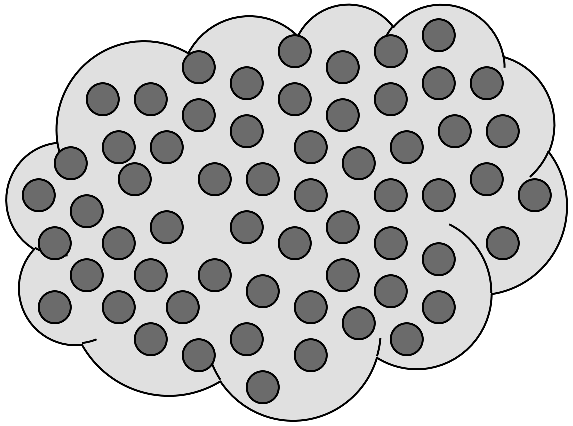
Use Case #1
Embedded,
Single-Host,
Deprivileged
Security Apparatus

Use Case #2
Lightweight,
Scalable,
Local Network
Capabilities

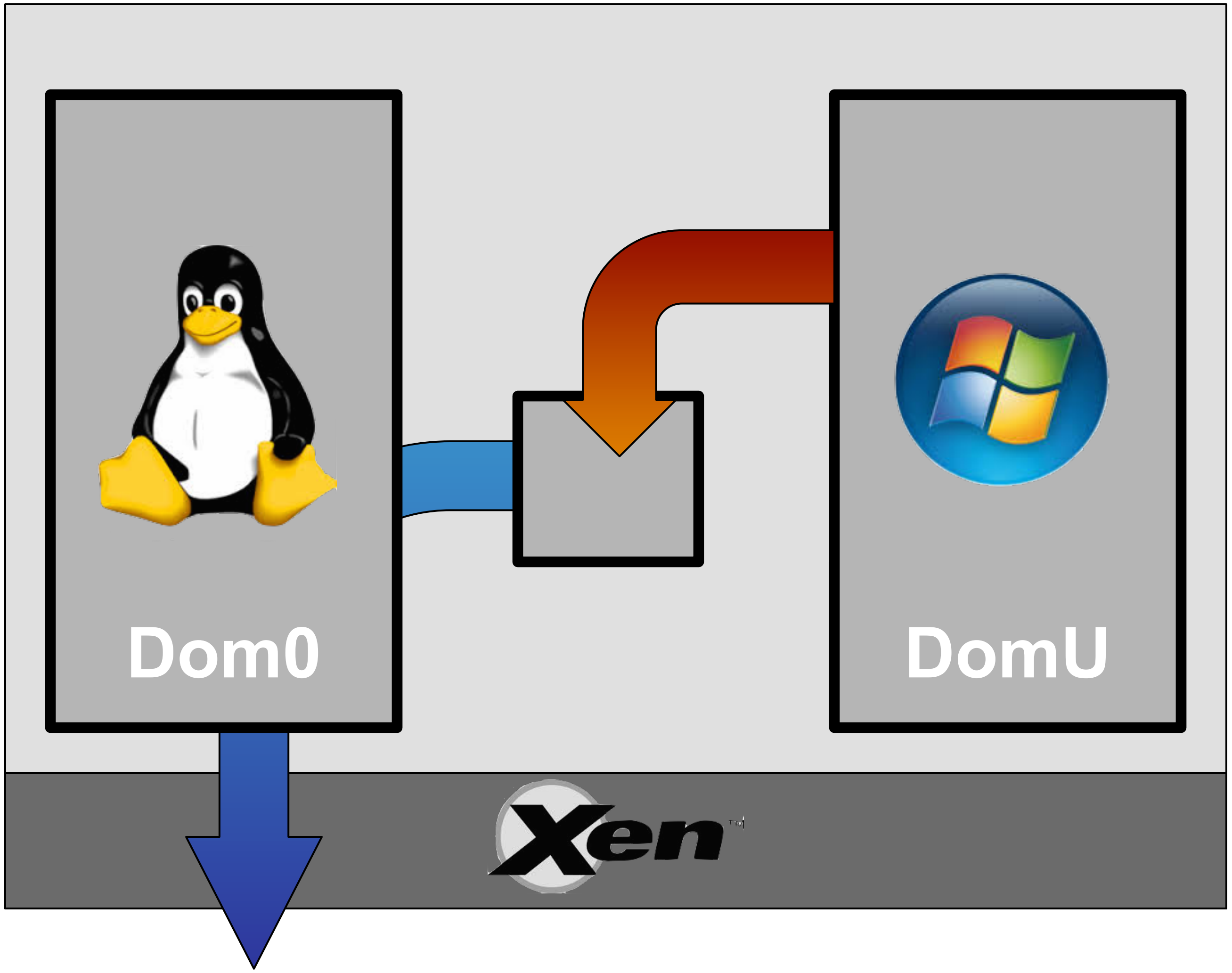
Use Case #3
Highly flexible,
Highly mobile,
On-demand
Network Nodes



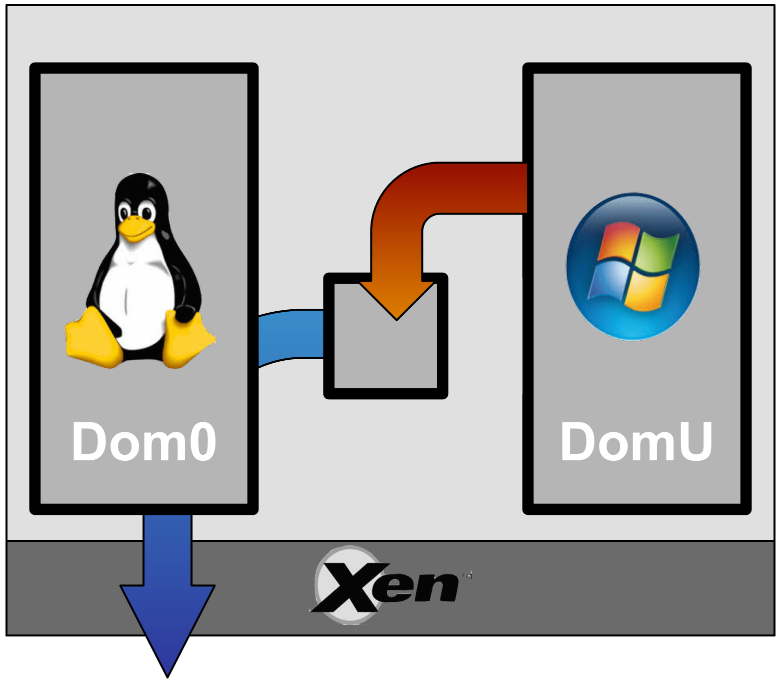




Use Case #1



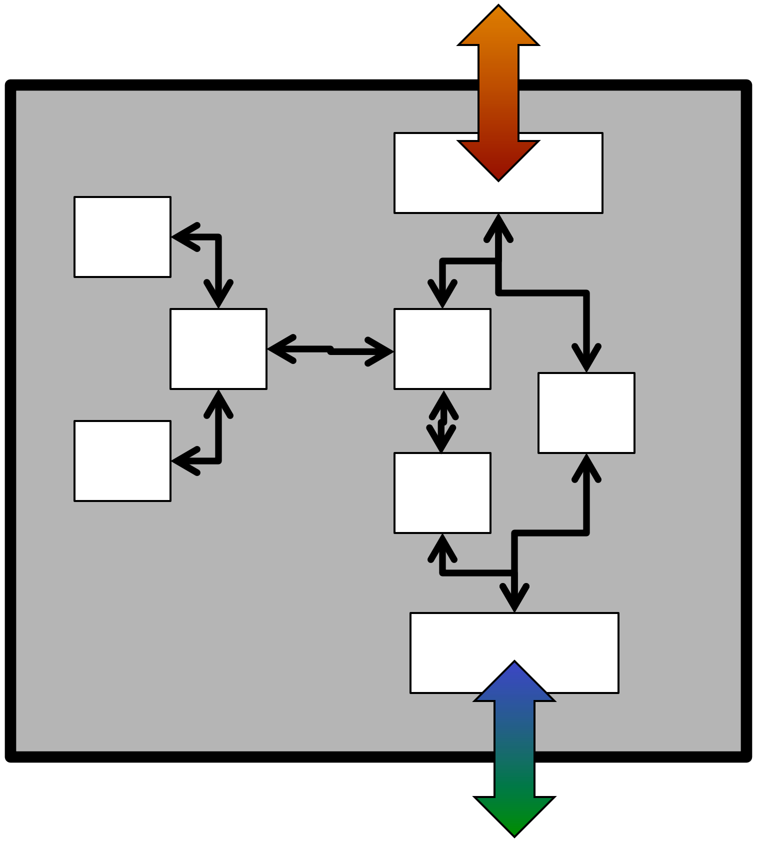
Use Case #1



Unavoidable …

… encryption.
… filtering.
… tunneling.

Use Case #1



Unikernels allow for fine-grained separation of concerns:

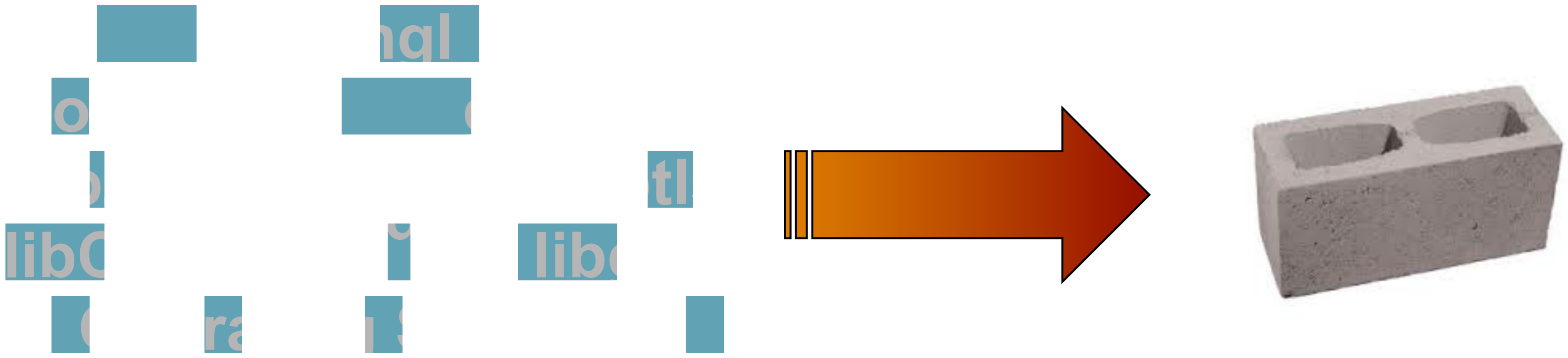
- Key management
- RNG
- Encrypt / Decrypt
- Networking

Use Case #1

We have been mostly successful using unikernels in this general area.

dangerous word

The one place it didn't work: a device driver.



Unikernels can make major, mid-stream changes more costly, because their specialization works against them.

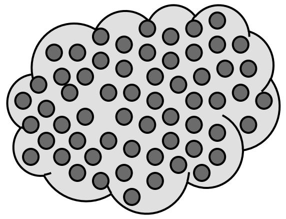
Awkward Transition Slide

Use Case #2 (Your Network)

Firewalls are nice, but they are going to be broken. (Or dodged.)

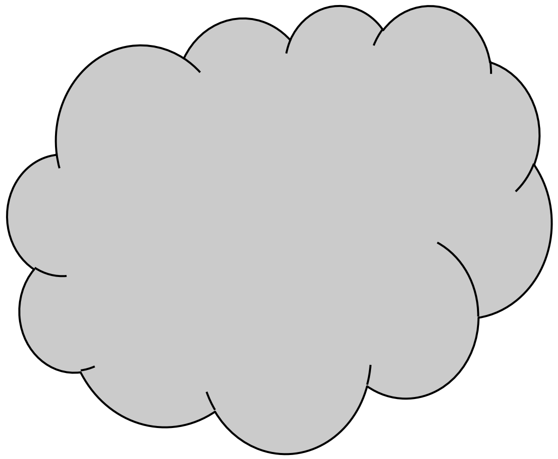
How quickly can you detect that someone has broken into your network, and how fast can you respond?

Confuse, distract, delay, discover: Make them work harder, and make them easier to detect.



Unikernels make great hosts for these nodes, because they are **lightweight** and **responsive**.

Use Case #3 (The Cloud!)



Unikernels have two major advantages over more traditional systems in the cloud:

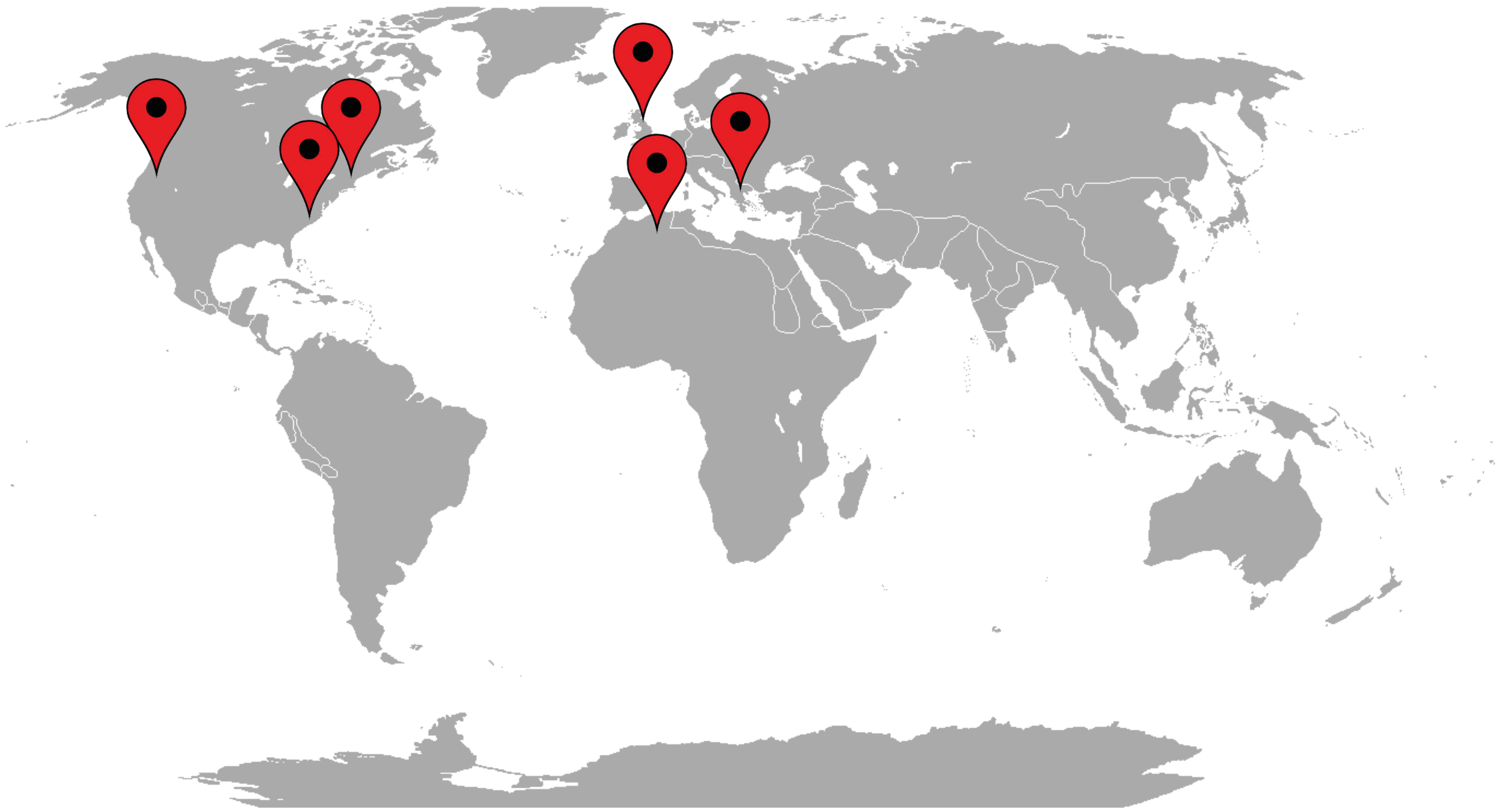
1. They're nimble.
2. They scale massively.

… as long as your service fits within a certain mold.

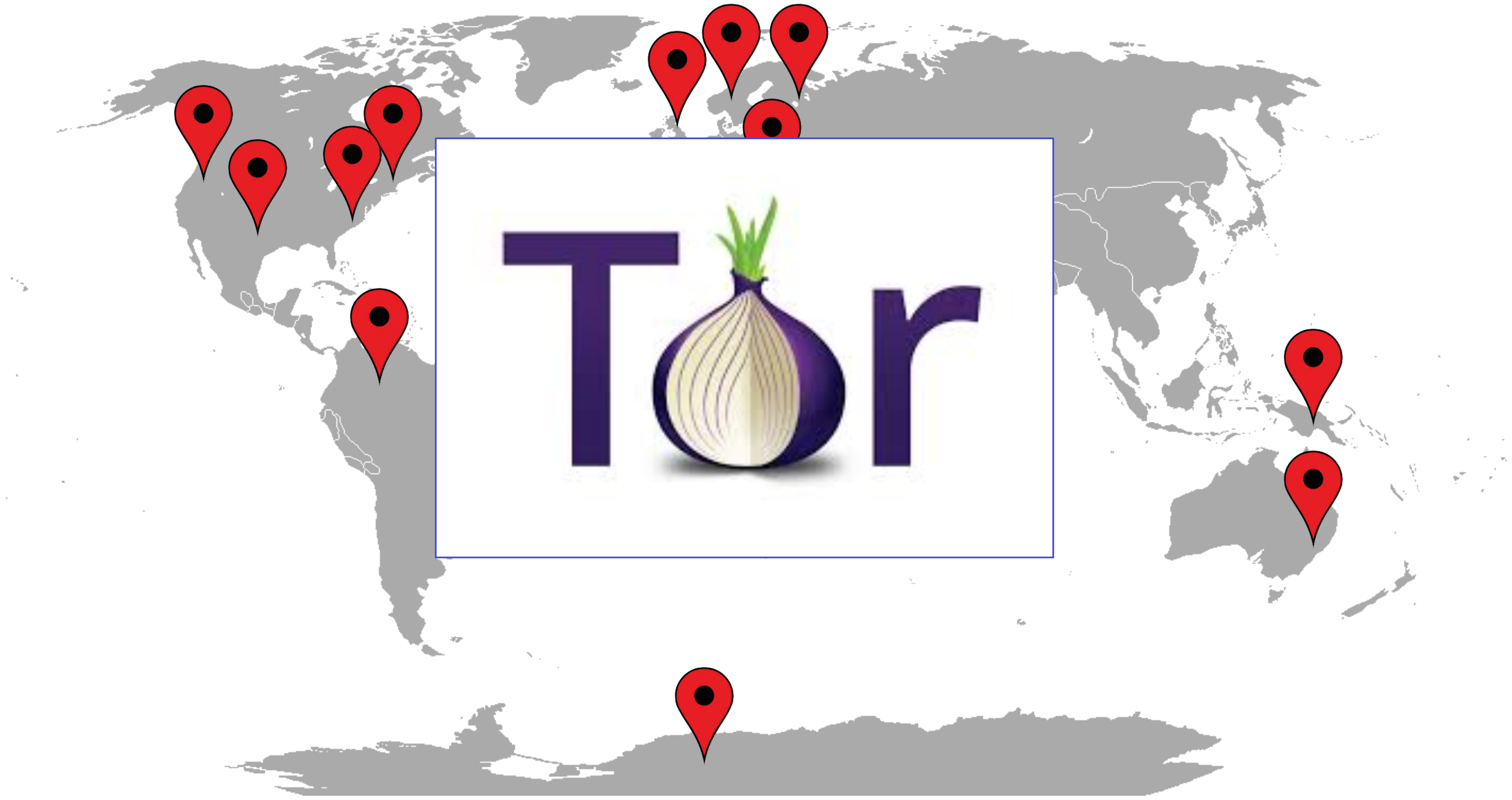
Do you need a local disk?

Use Case #3: Nimble

Use Case #3: Nimble



Use Case #3: Nimble + Massive

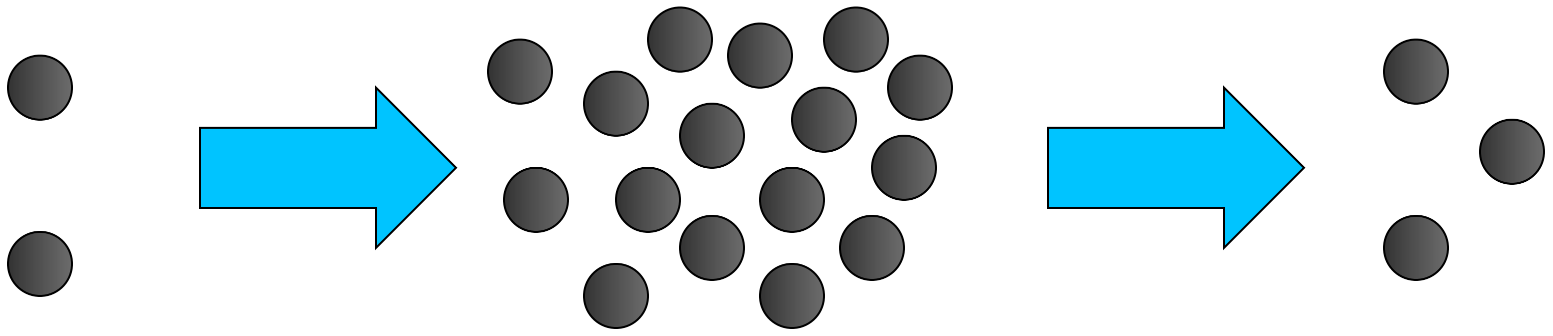


Use Case #3: Flexible + Responsive

One Erlang unikernel demo accepted a web request, booted a unikernel to handle it, and then shut down the unikernel once it was done, with no human-appreciable delay.

Cool demo, but a bit over the top.

But think about what this means in terms of dynamically scalability.



Unikernels: ~~Who, What, Where, When, Why?~~

- ~~The advantages and disadvantages of unikernels.~~
- ~~Where Galois has used them in the past that worked.~~
  - ~~… and didn't work.~~
- What general rules we think apply.

Unikernels

What?	A lightweight mechanism for implementing single-service components.
Who?	Designers of cloud, local network, or low-level security services.
Where?	Useful for securing or rapidly deploying lightweight or security-sensitive services.
When?	For situations, like those described, for which the increased cost of development for a unikernel is outweighed by a unikernel’s advantages.
Why?	To save money. To improve efficiency. To improve security.

Summary

Unikernels: Useful for us.

≥?

Where will you use them?

HaLVM:	http://halvm.org
Mirage:	http://openmirage.org