IBM Research, Hawthorne NY

Opening Black Boxes: Using Semantic Information to Combat Virtual Machine Image Sprawl

Darrell Reimer, Arun Thomas*,
Glenn Ammons, Todd Mummert,
Bowen Alpern, Vasanth Bala

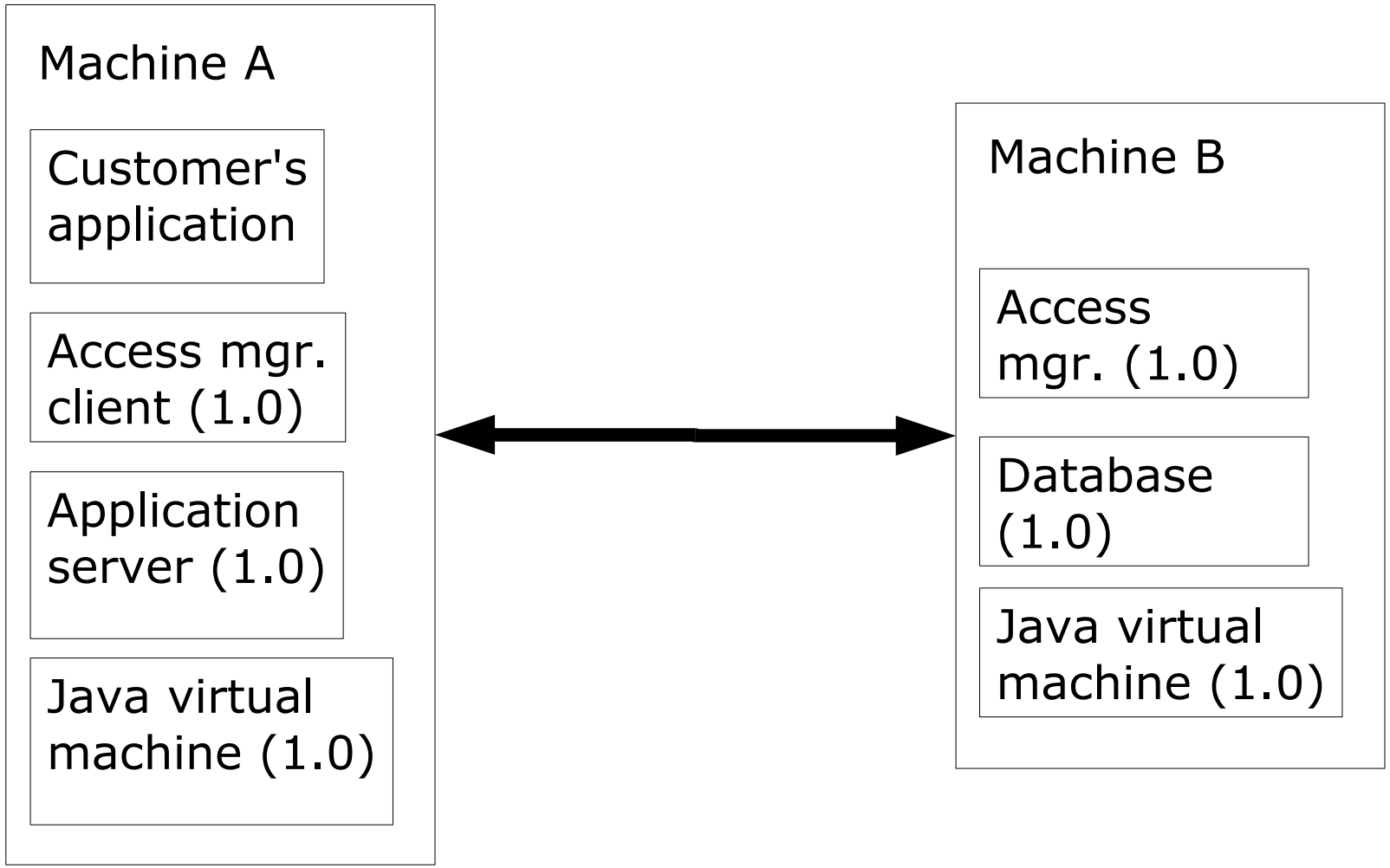
*University of Virginia

March 6, 2008

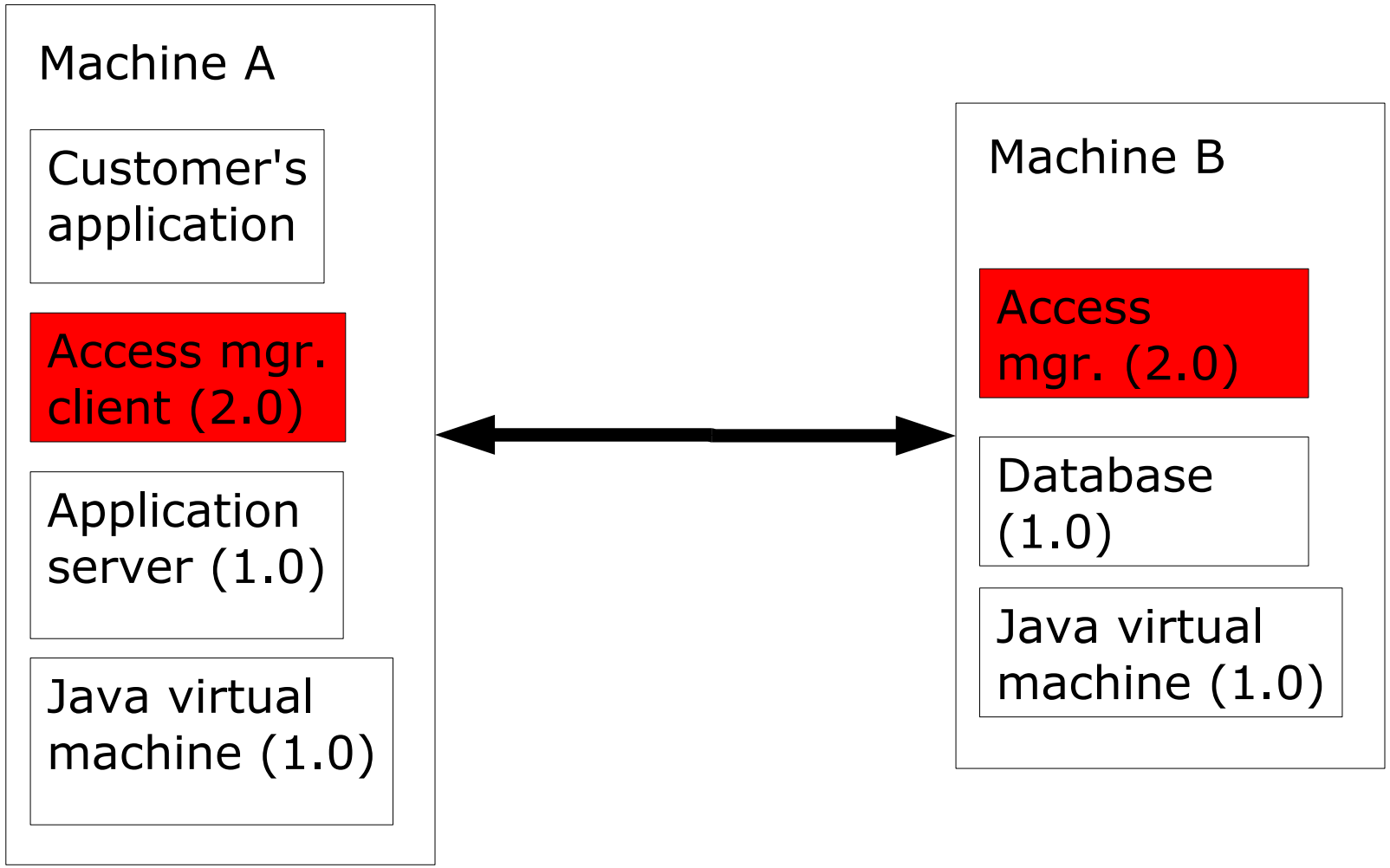
What is sprawl?

- **Many servers, each configured differently**
  - focus on enterprise software: different DBMS, app server, etc.
- **Impossible in practice to**
  - describe each configuration
  - duplicate any configuration
- **Difficult in practice to**
  - alter a configuration
  - test a configuration

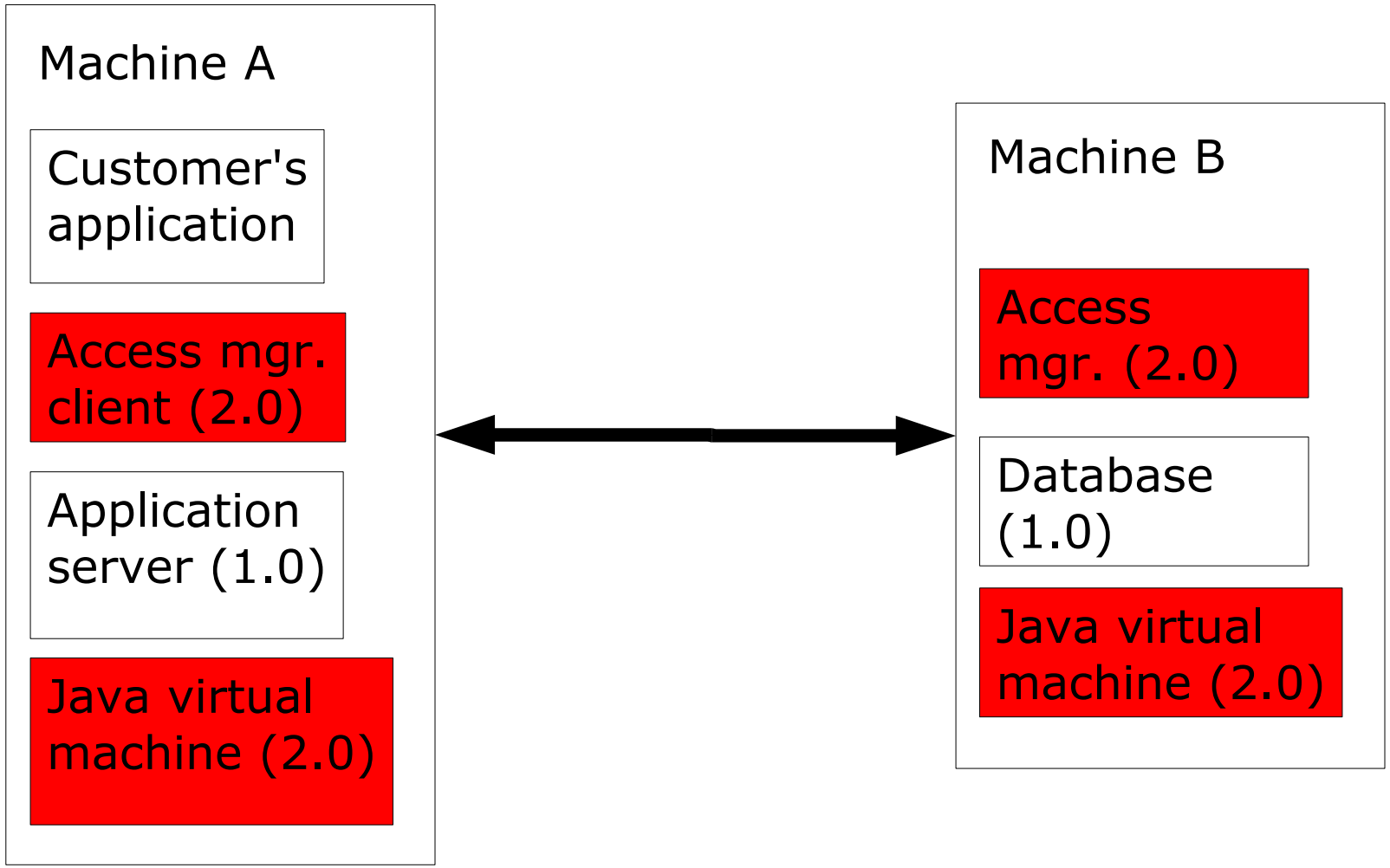
Anecdote: sprawl makes diagnosis harder



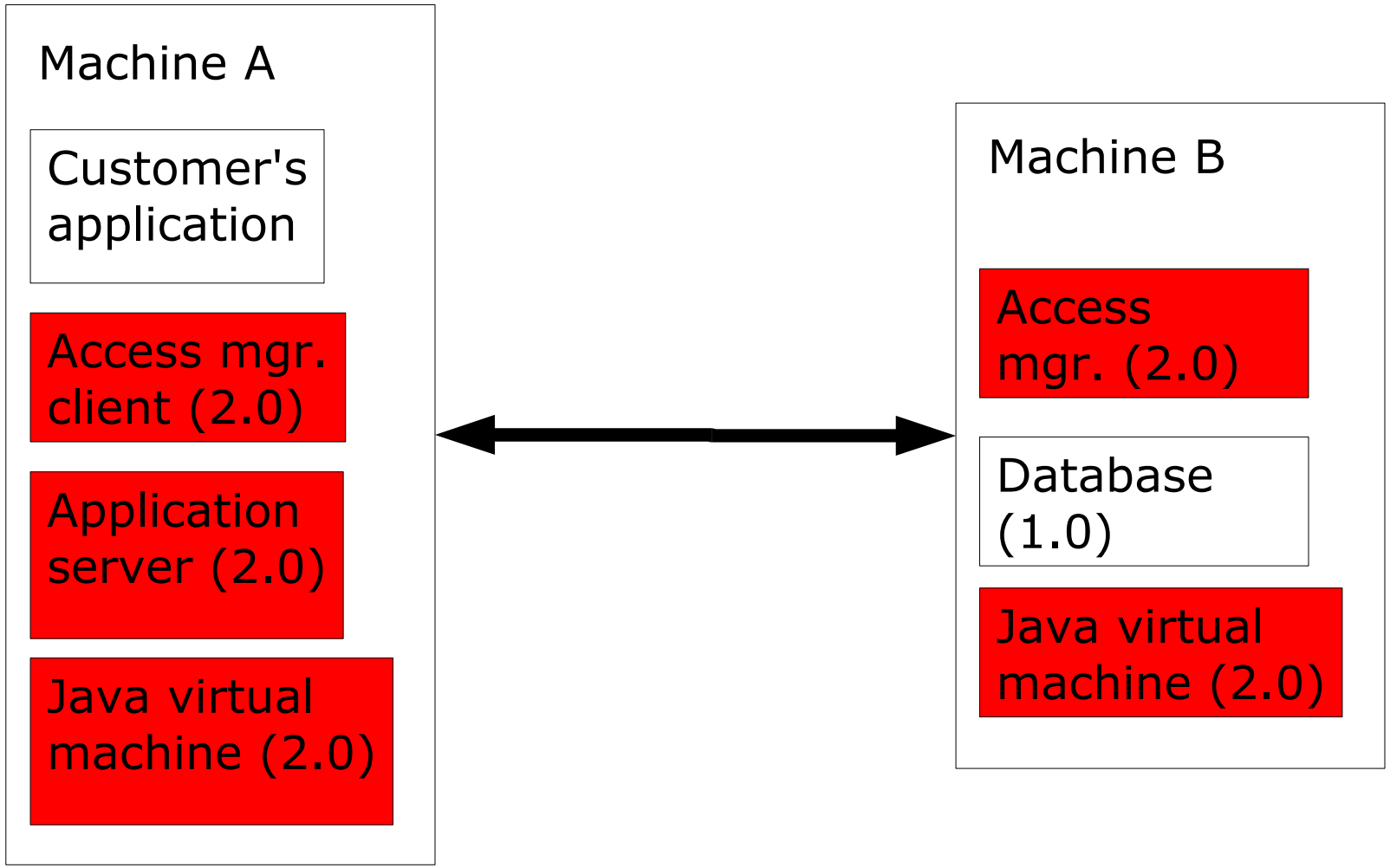
Upgrade made application slower!



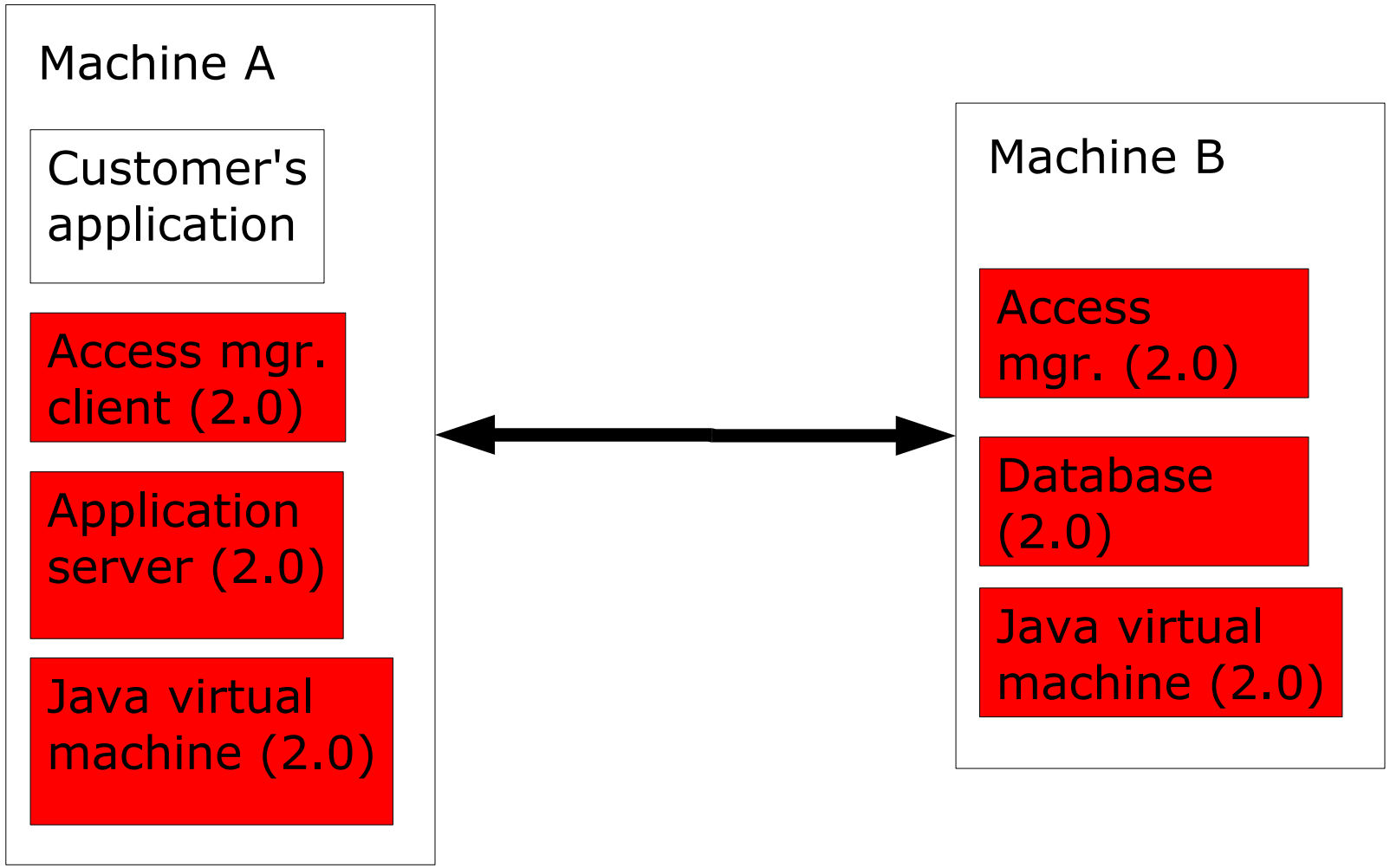
Good idea? "Fix" configuration



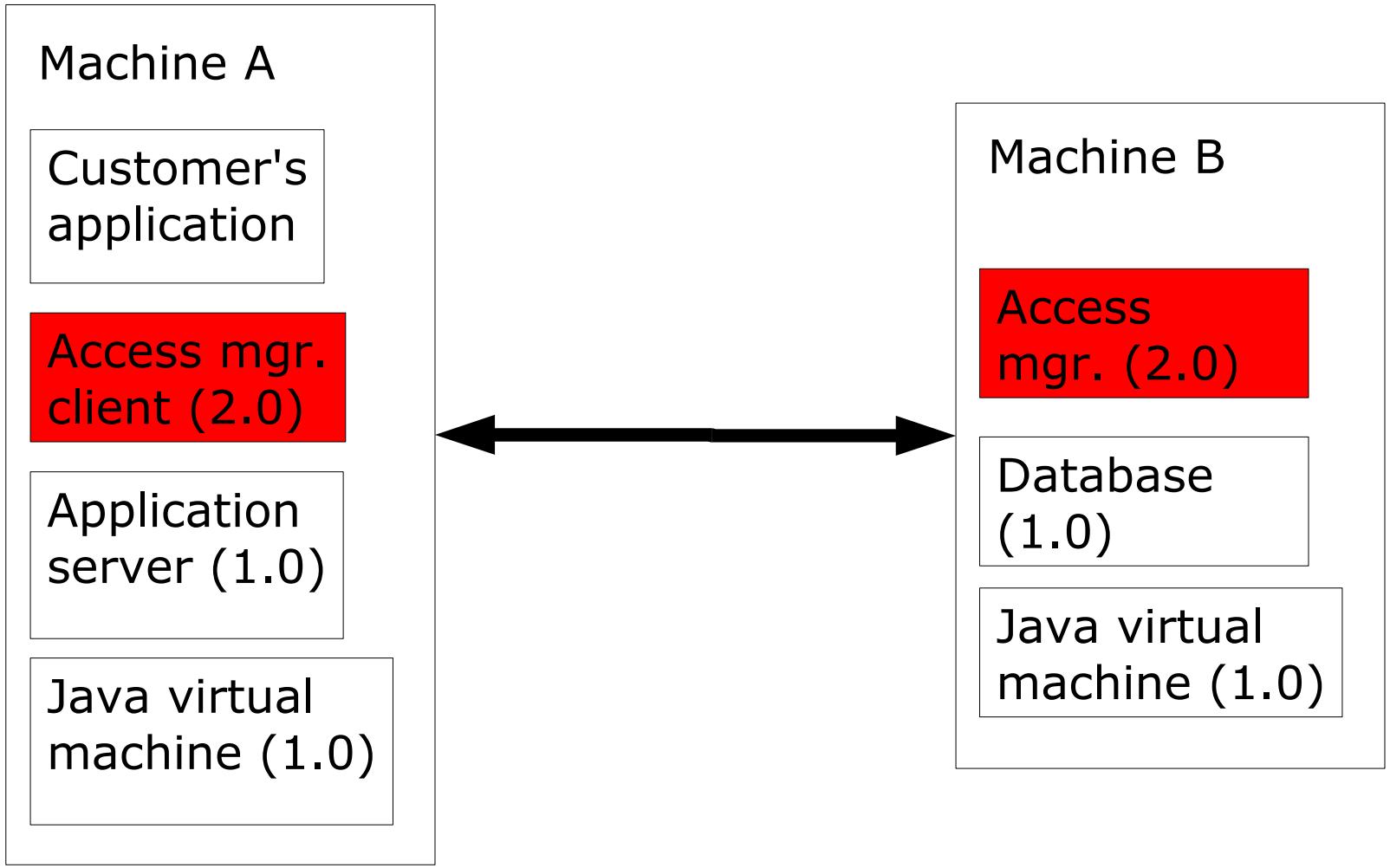
Good idea? "Fix" configuration



Good idea? "Fix" configuration



Good idea: reduce, experiment, measure



Anecdotal lessons

- **More dependences => harder problem**
- **Configurations are irreproducible**
  - limits application of scientific method
  - must experiment at customer's site
- **Would like to reduce configuration space**
  - but install problem makes that hard to do
    - if it doesn't work, you just lost a month
  - anyway, application code is always different

Is sprawl a problem elsewhere?

- Academia?
- Consumers?
- Government?
- Open-source projects?

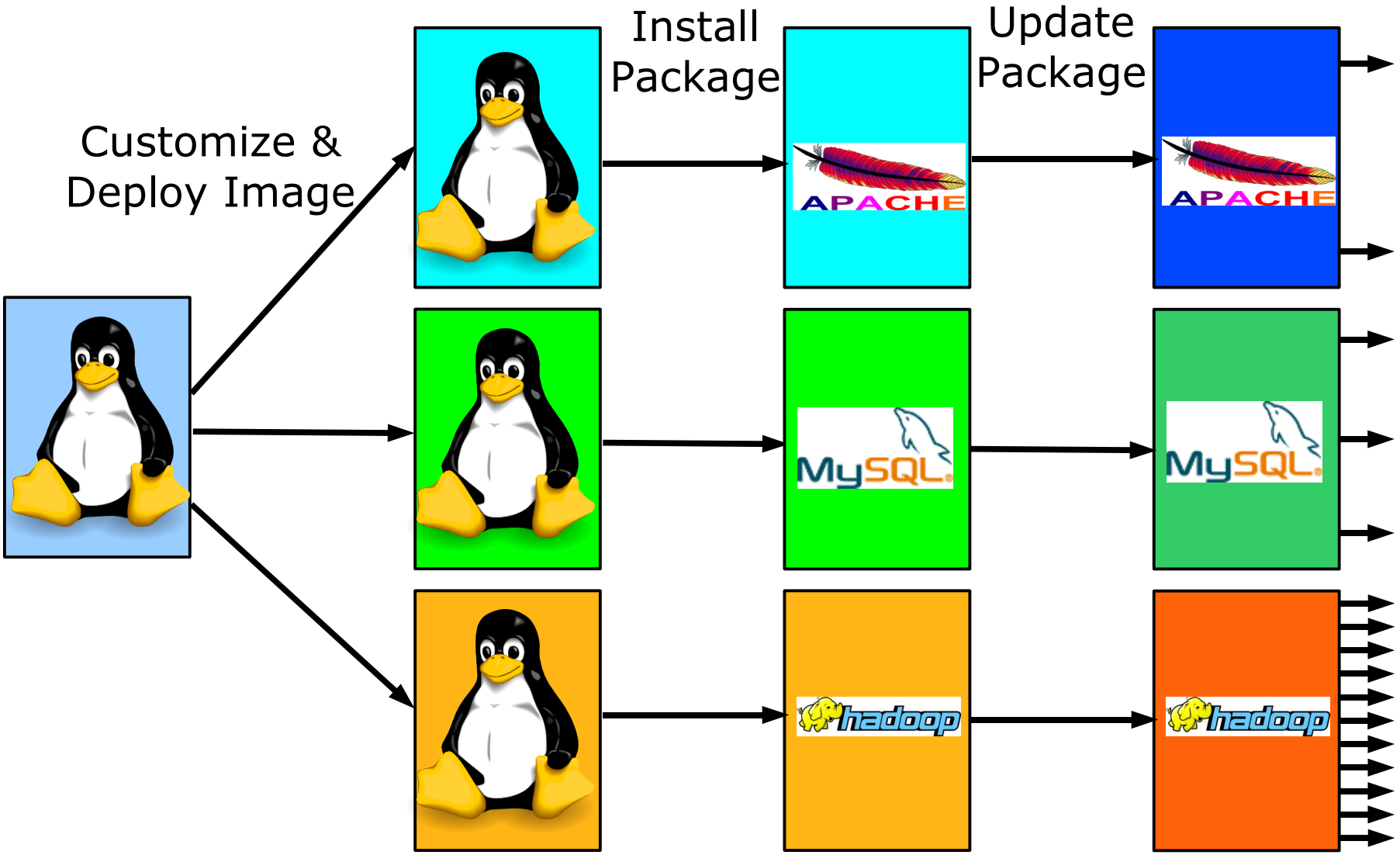
How does virtualization affect sprawl?

- **Nightmare scenario**
  - configurations multiply:
    - lots of clones
    - more virtual-machine images than physical machines
  - tools stay the same
    - install is as hard as before
    - image construction is ad hoc (as for physical machines)
    - images aren't portable
    - relationships between images aren't tracked

How does virtualization affect sprawl?

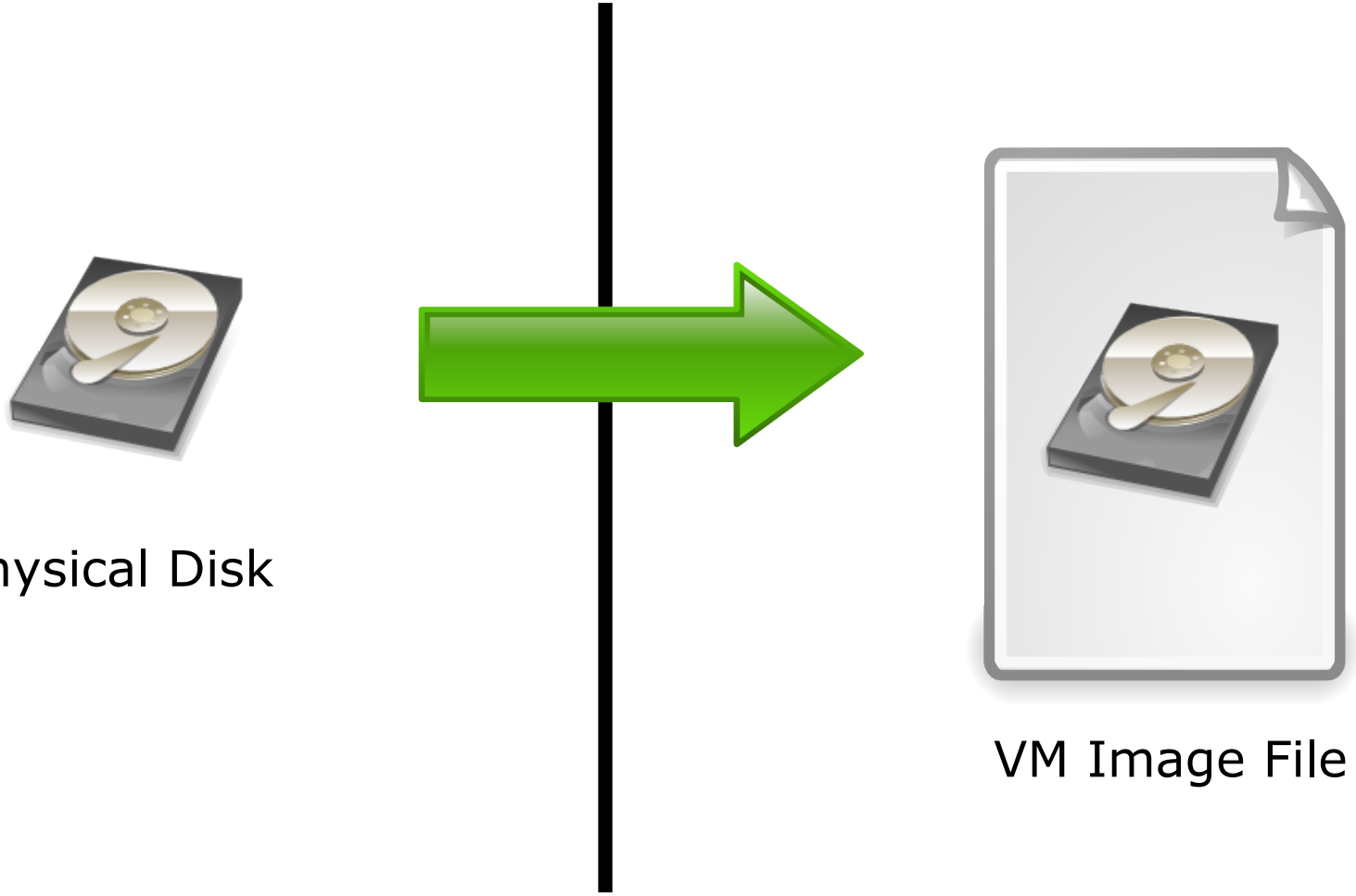
- **Mirage scenario**
  - configurations multiply:
    - lots of clones
    - more virtual-machine images than physical machines
  - tools improve
    - treat images as data, in a useful format
    - images are portable
    - images stored in repositories
      - searchable
      - provenance is tracked
      - construction is scripted

How VM Images Sprawl (our view)

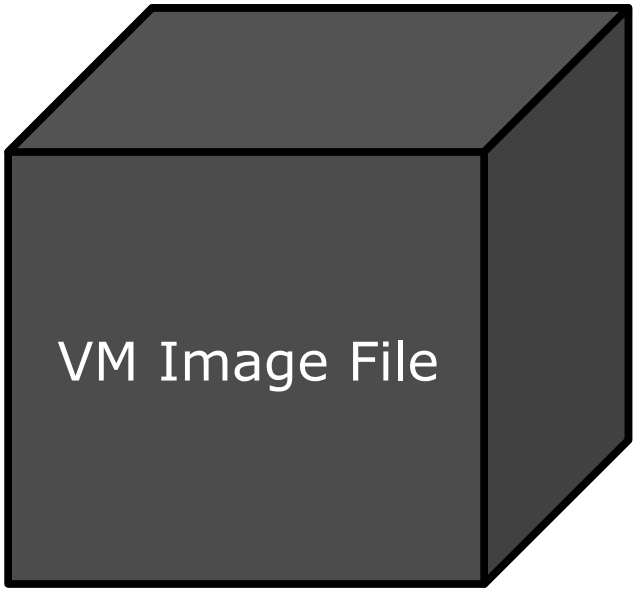


What is today's image format?

VM Images mirror the structure of physical disks



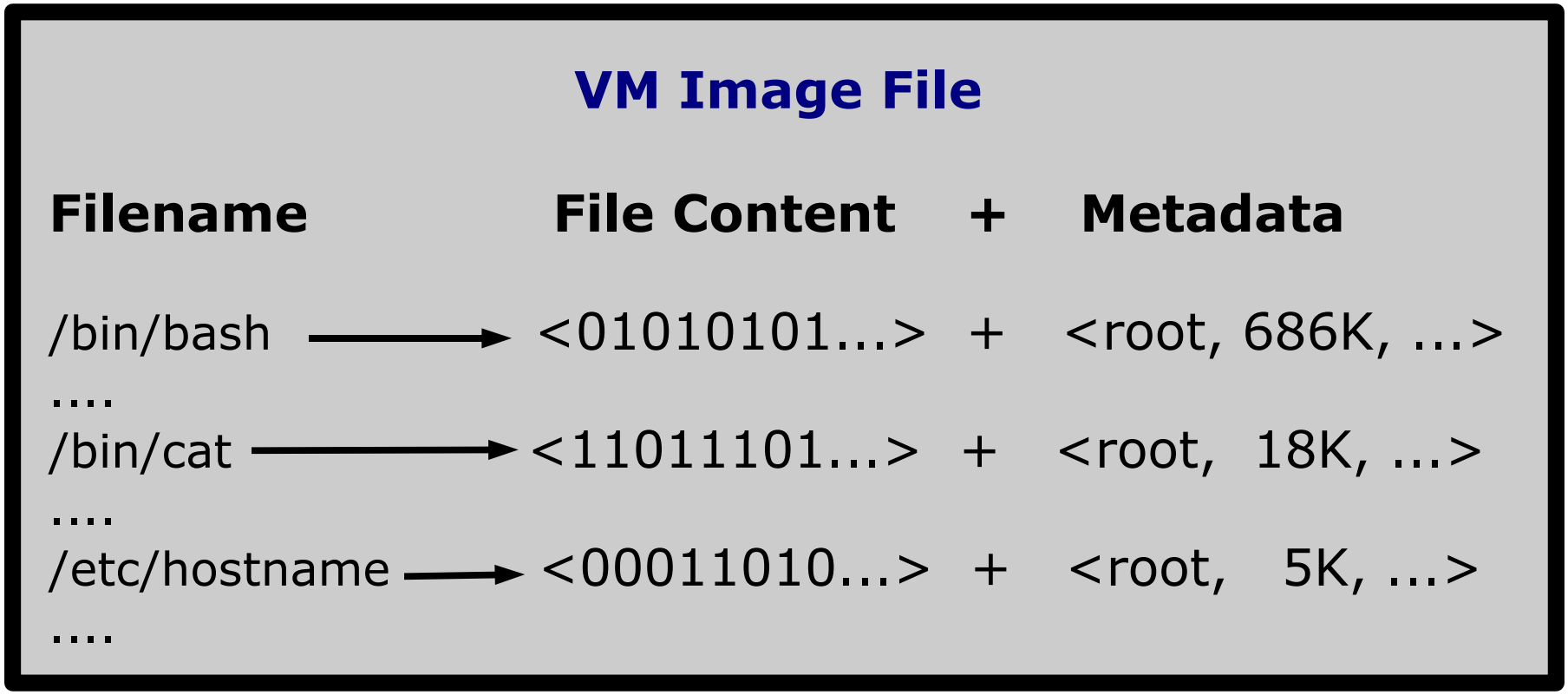
What's wrong with today's format



- **VM Images are black boxes**
- **Difficult to determine contents**
- **Designed for execution, not management of images**

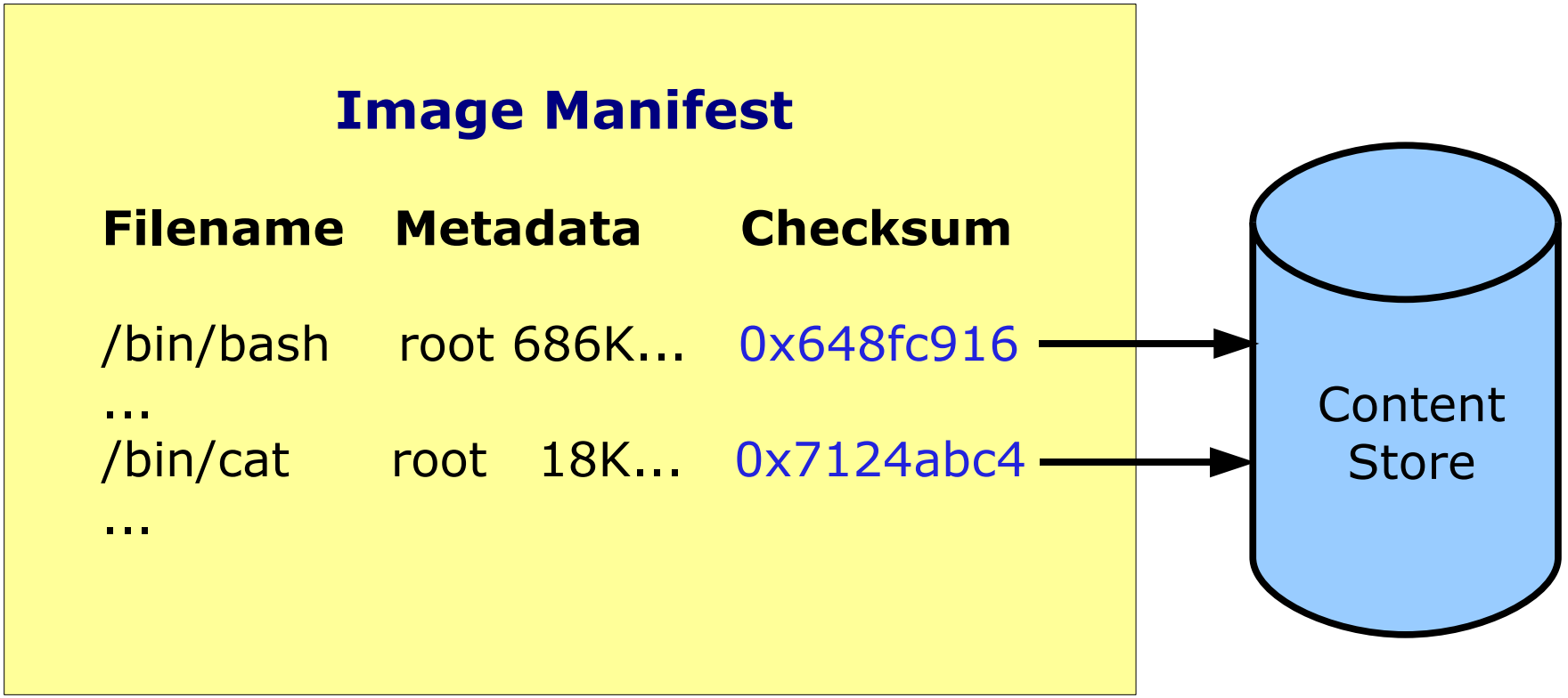
Semantic Information Buried in VM Image

Mapping from Filename to File Content/Metadata



Mirage Image Format exposes semantics

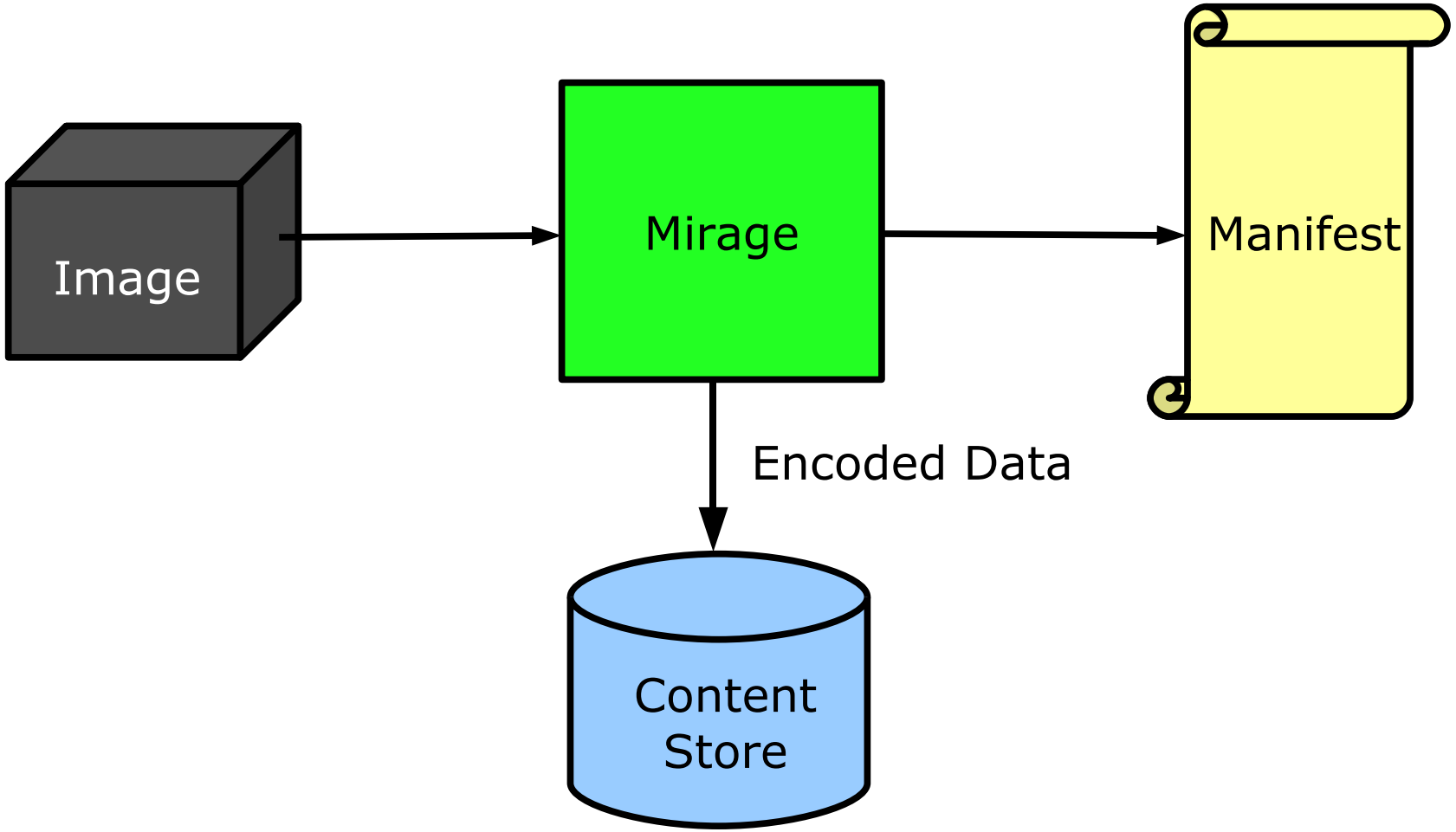
- **Manifest captures image metadata**
- **Content Store holds all data (file contents)**



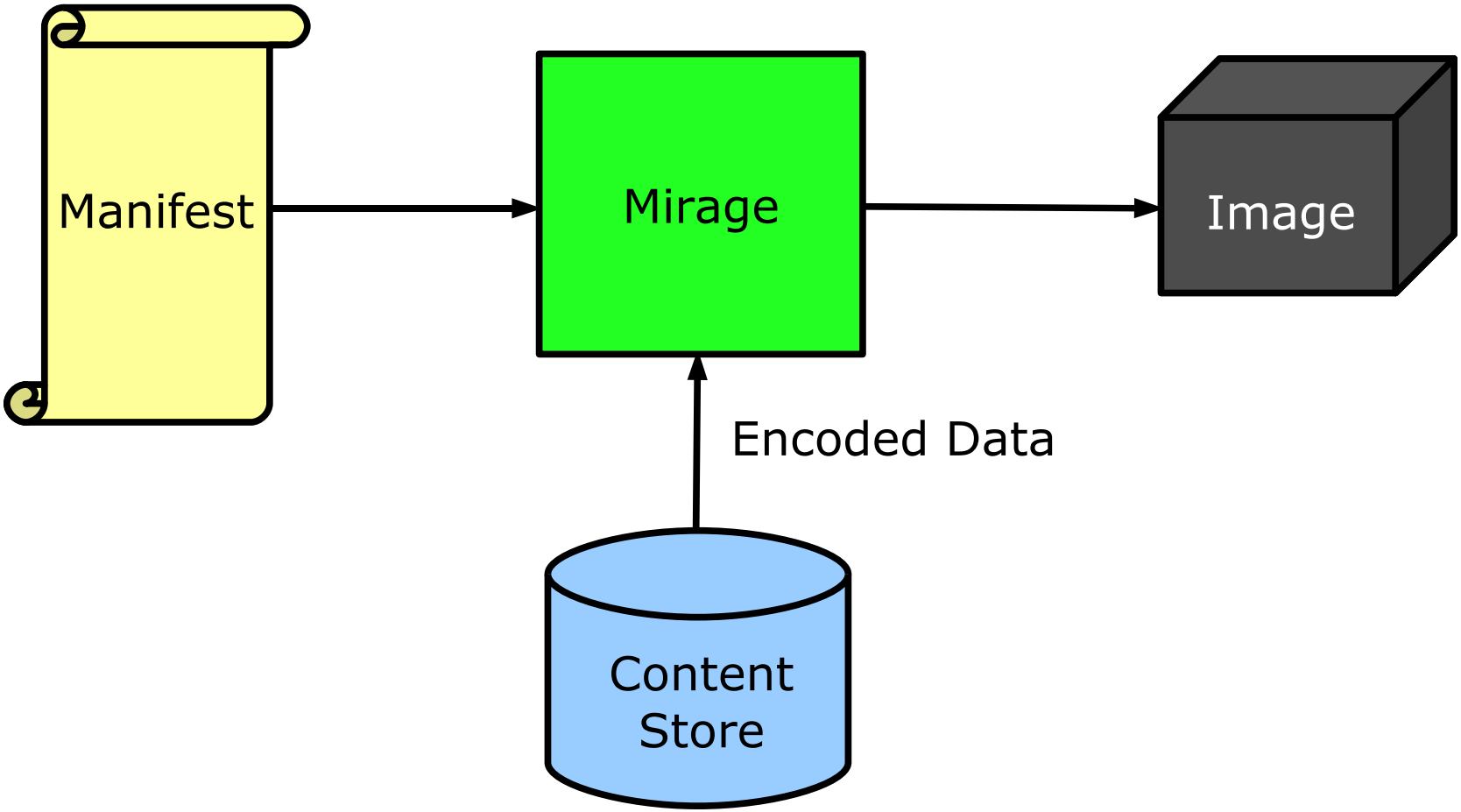
Manifests: Beneficial Characteristics

- **Manifests are much smaller than whole images**
- **Mirage management tools can operate on manifest only or manifest + partial image**
- **Mirage management tools can operate on dormant images**
- **File checksums allow for storage optimization**

Publishing Images in Mirage



Retrieving Images in Mirage



Mirage in Action

Images Used in Evaluation

- **Min** – Minimal Debian Linux image (280 MB)
- **Base** – Standard Debian Linux image (450 MB)
- **Wiki** – Base image + Mediawiki (840 MB)
- **GUI** – Base image + full Gnome desktop environment (1670 MB)
- **IDE** – Base image + commercial Eclipse-based IDE (2240 MB)

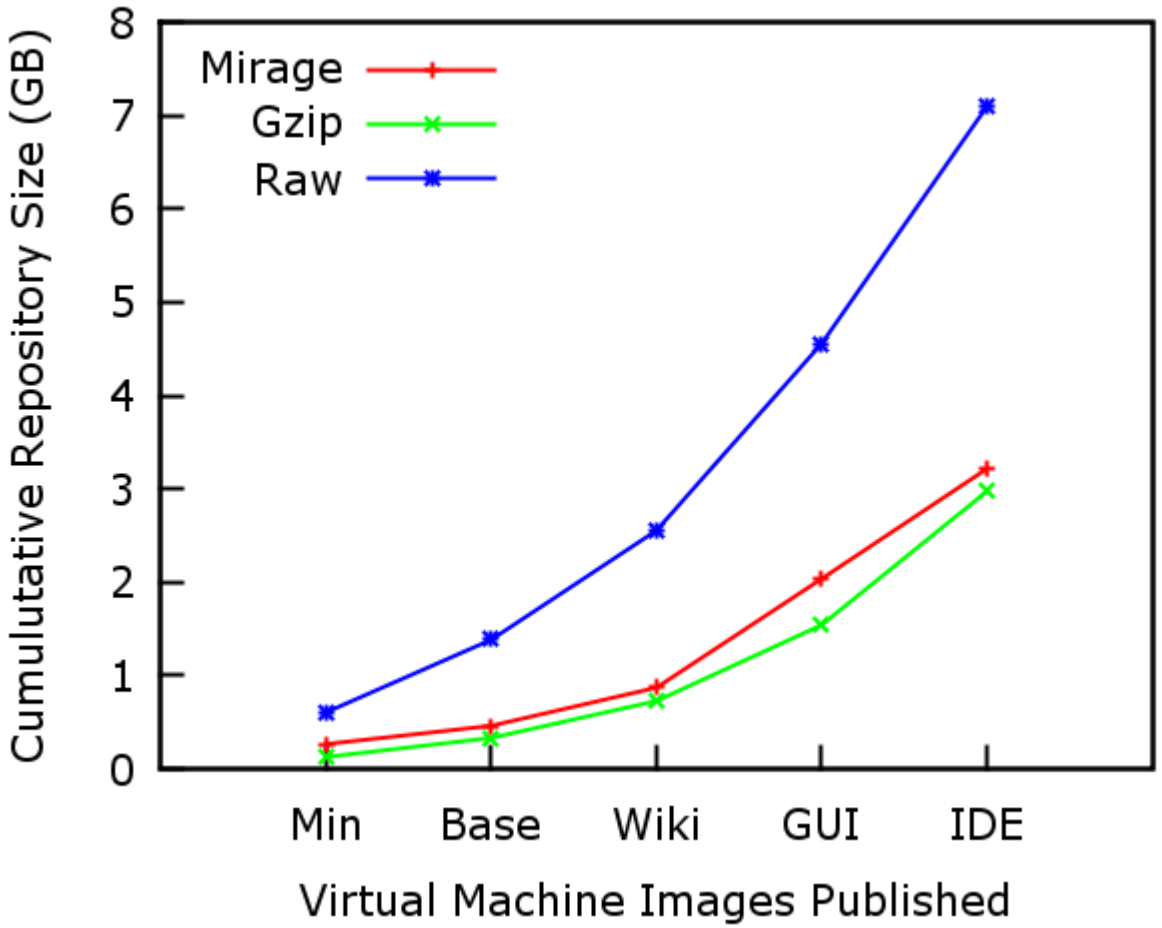
Publish/Retrieve are fast enough

Publish and Retrieve are I/O limited operations

Name	Image Size (MB)	Manifest Size (MB)	Time (sec) Publish	Retrieve
Min	280	3.5	34	21
Base	450	4.7	49	28
Wiki	840	7.3	137	102
GUI	1670	12.7	309	246
IDE	2240	15.5	451	353

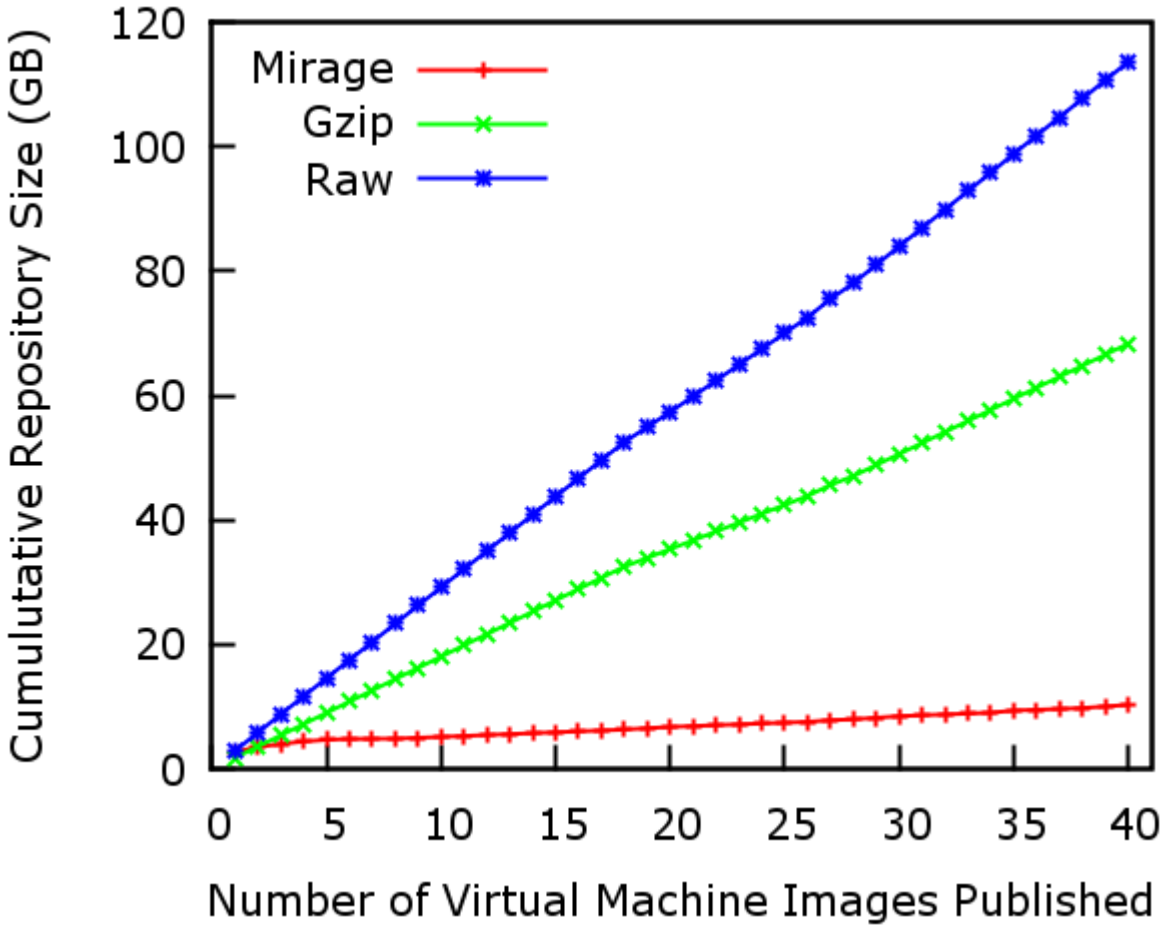
MIF is storage-efficient

On Debian images, reduces storage by 2.2x



MIF is storage-efficient

On IDE images, reduces storage by 10.9x



Mirage in Action – Software Management

Software Management Tasks

- **Inventory Control – Determine which software is installed in each image**
  - Scenario A: Query images for a program
- **Customized Deployment – Deploy customized clones of a master image**
  - Scenario B: Deploy a cluster of servers
- **Software Update – Update large numbers of similar images**
  - Scenario C: Install a package

Scenario A: Query images for a Program

- **Query repository for images that contain given file checksums**
- **Current Approach: Agent periodically scans images to create database of checksums**
- **Our approach:**
  - File manifest replaces checksum database
  - Only scan image once when publishing

Image Query Results

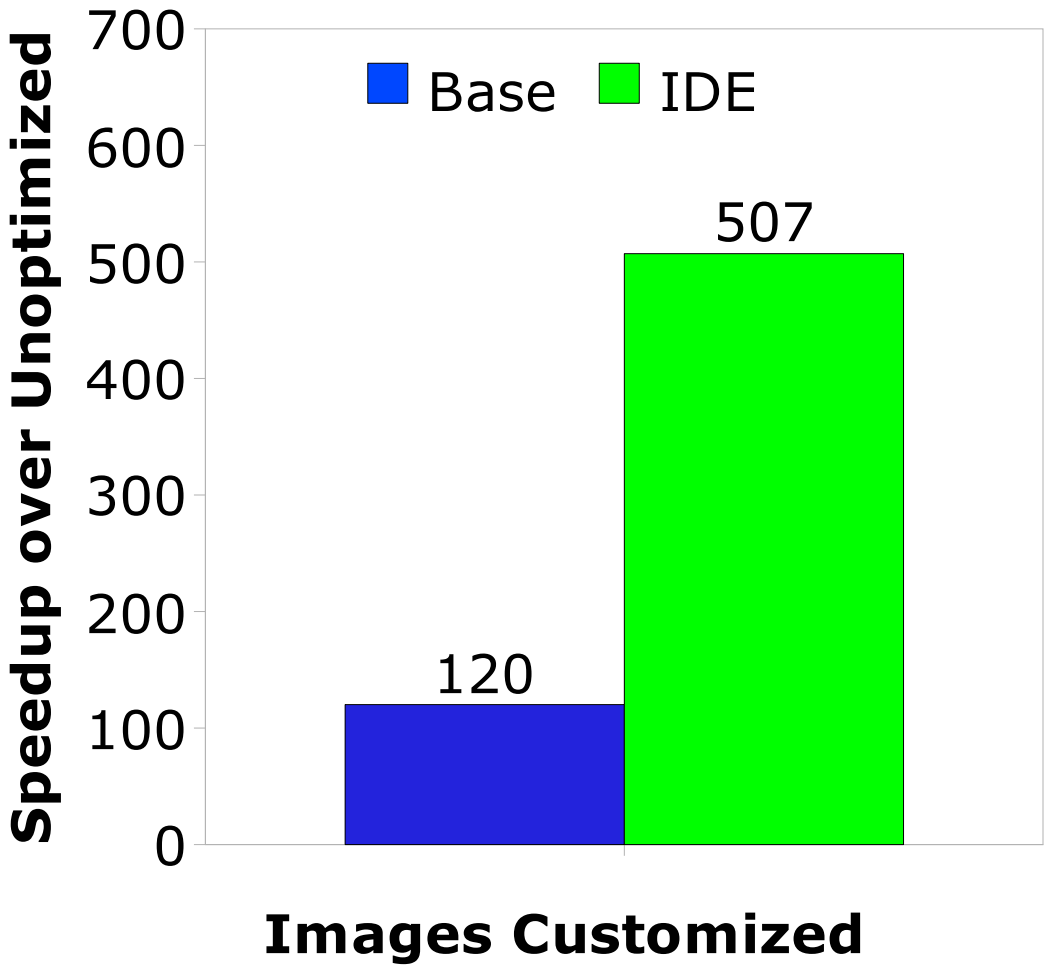
Name	Image Size (MB)	Lookup Time (sec) 1 File	1000 Files
Min	280	0.5	1.2
Base	450	1.1	1.3
Wiki	840	1.6	1.9
GUI	1670	2.2	3.0
IDE	2240	2.6	3.2

Scenario B: Deploy a Cluster of Servers

- **Baseline: Clone master image; Modify networking files; Push image**
- **MIF-based optimizations**
  - Selective retrieval – Retrieve only selected files from an image instead of entire image
  - Overlay manifests – Manifests that include only delta from base image
- **Can represent a customized image in KBs**
- **Optimizations significantly speed up customization process**

Cluster Deployment Results

Customization is up to 507x faster with MIF opts.

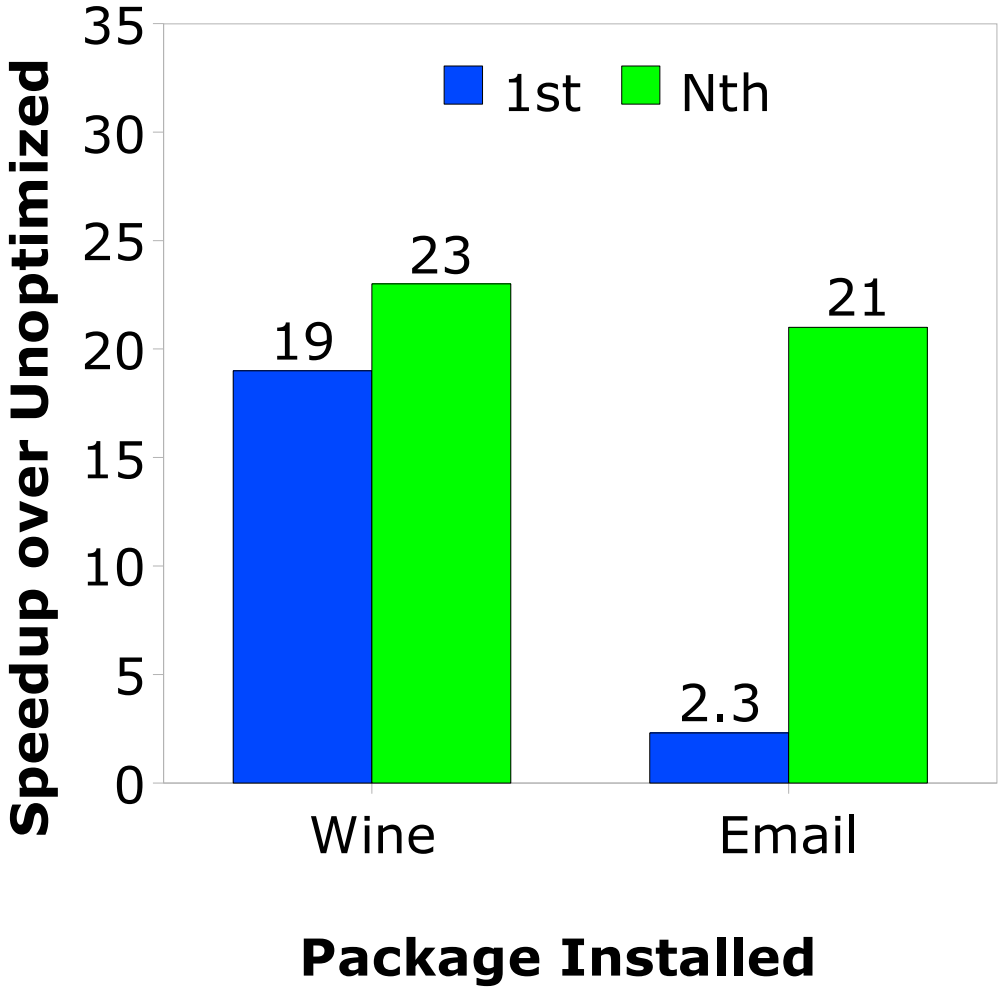


Scenario C: Install a Package

- **Current Approach: Pull image; Start/Mount Image; Install package; Push image**
- **Modify `dpkg` package manager to exploit MIF**
- **MIF-based optimizations**
  - Selective retrieval and overlay manifests
  - Memoization – Cache results of updates
- **Exploits the fact that many images are similar to each other**

Package Install Results

Installs are up to 23x faster with MIF opts.



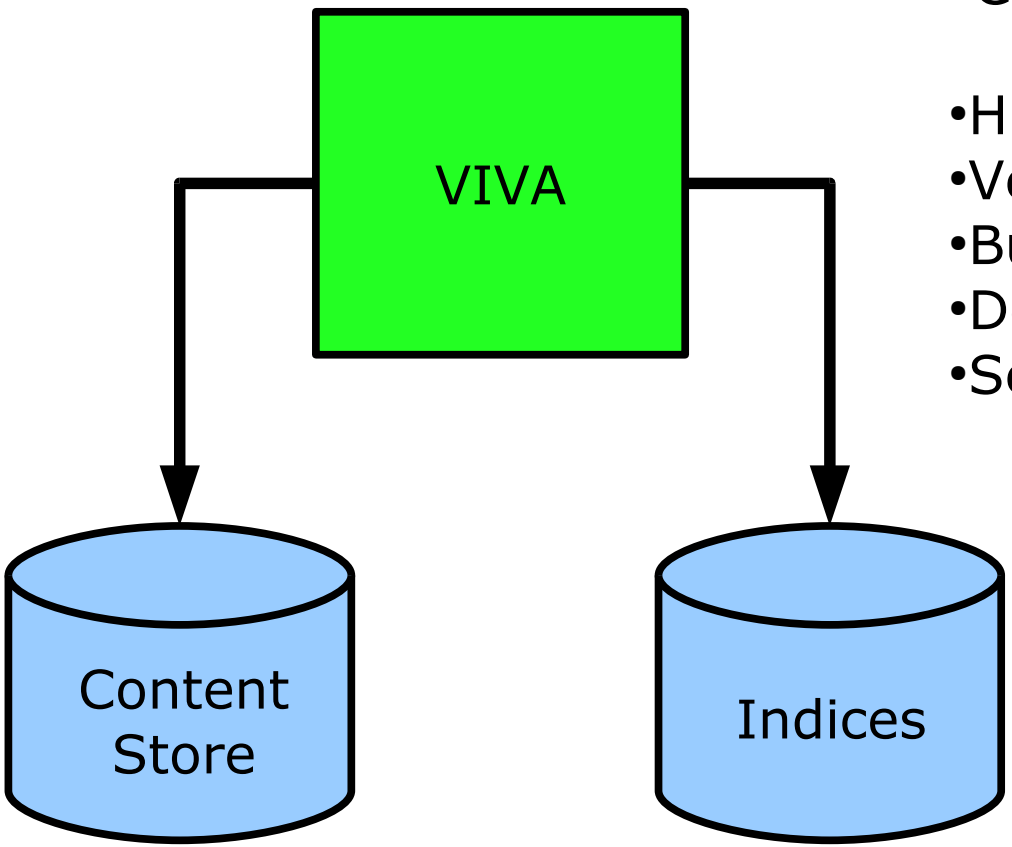
Related Work

- **Ventana (Stanford)**
  - virtualization-aware filesystem
- **Machine Bank (Microsoft)**
- **Moka5 Engine (Moka5)**
- **Lab Manager/Update Manager (VMWare)**
- **rBuilder (rPath)**
- **Nix OS**

Ongoing and Future Work

- **Goal: Build scalable repositories of VM images**
  - Efficient versioning support for VM images
  - Better query facilities for images and repositories of images
  - Further integration between package management and Mirage
  - Better hypervisor integration

VIVA: Virtual Image Versioning A(?)



"CVS + Make for VM images":

- Hierarchical namespace
- Versioned objects
- Build avoidance
- Derivations
- Search by
  - filename
  - file content
  - file attributes
  - keyword

Package-management for VM images

- **Now**
  - per-machine package database
  - central repository of packages
  - assumption: time-to-install dominated by download time
- **Want**
  - packages w/ large memoizable parts
  - central repository w/ memoization results
  - time-to-install dominated by non-memoizable part
  - compare Nix OS

Hypervisor integration

- **Now: create image, then run**
- **Want:**
  - demand-fetch
  - unstructured updates
  - preserve named content; communicate to hypervisor
- **Conflict:**
  - hypervisor implements “disk”
  - at high-level, need “file”
  - filesystem maps between two, but inflexibly

Summary

- **Sprawl is a huge problem**
- **Virtualization might make sprawl worse**
- **Solution: treat images as data**
  - New data format, MIF, exposes file-level information
  - tools that operate on MIF simplify and speed up software management tasks

For More Information

Project Website:

http://www.research.ibm.com/mirage

Email Contacts:

Bowen Alpern	alpernb@us.ibm.com
Glenn Ammons	ammons@us.ibm.com
Vasanth Bala	vbala@us.ibm.com
Todd Mummert	mummert@us.ibm.com
Darrell Reimer	dreimer@us.ibm.com
Arun Thomas	arun@cs.virginia.edu