IBM Research, Hawthorne NY

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

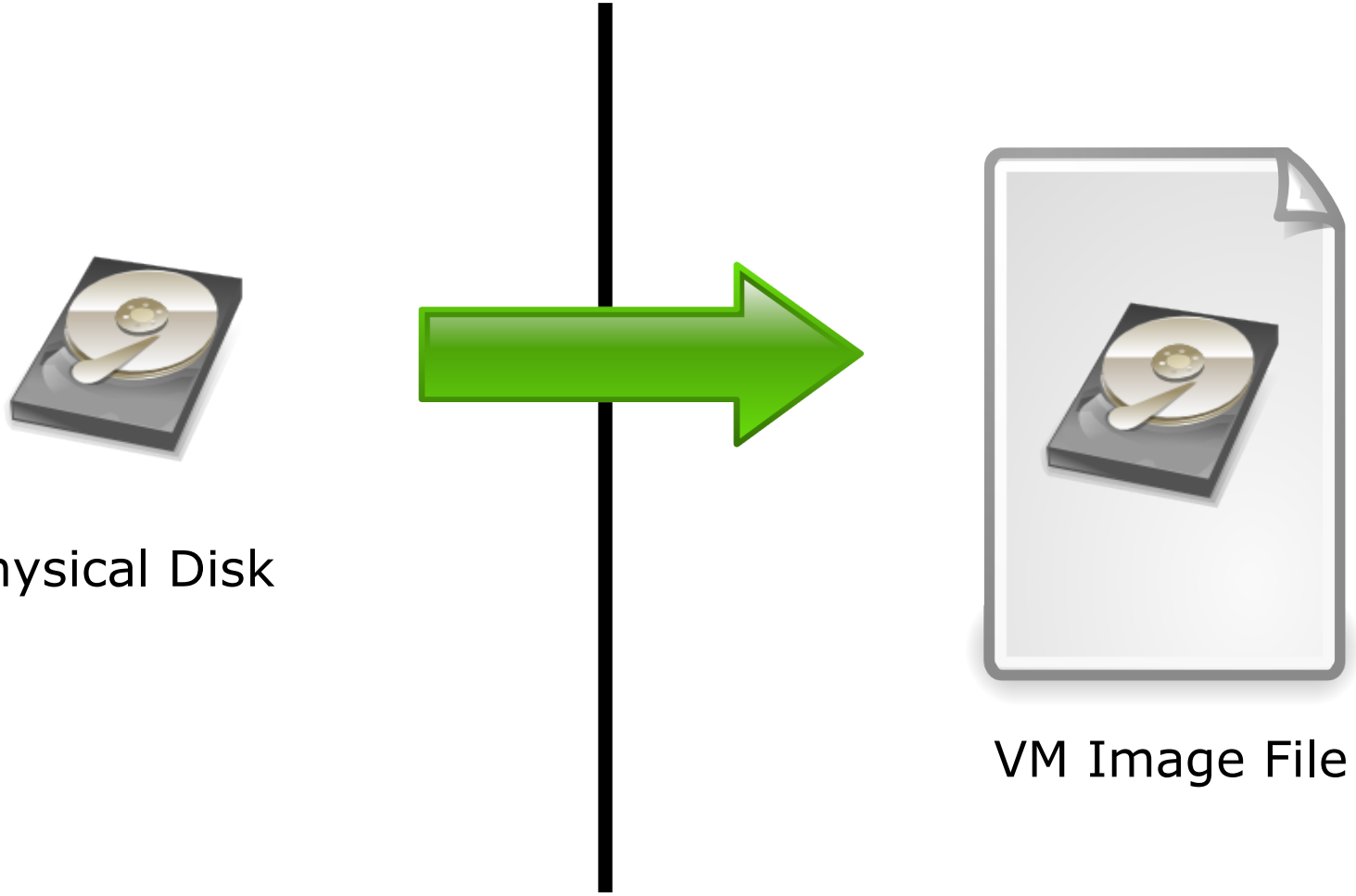
Darrell Reimer, **Arun Thomas***, Glenn Ammons, Todd Mummert, Bowen Alpern, Vasanth Bala

***University of Virginia**

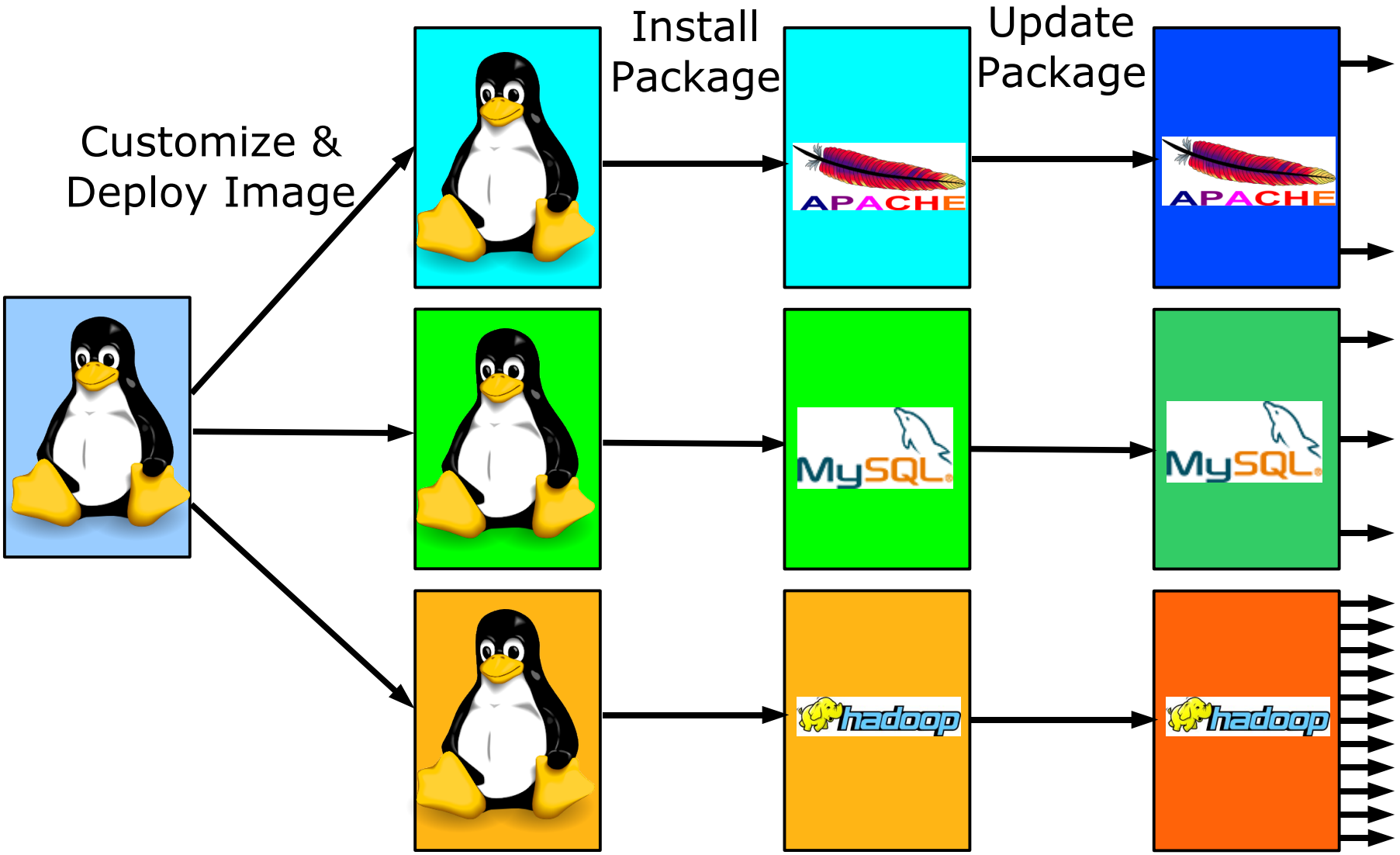
March 6, 2008

How are VM Images Distributed Today?

VM Images mirror the structure of physical disks



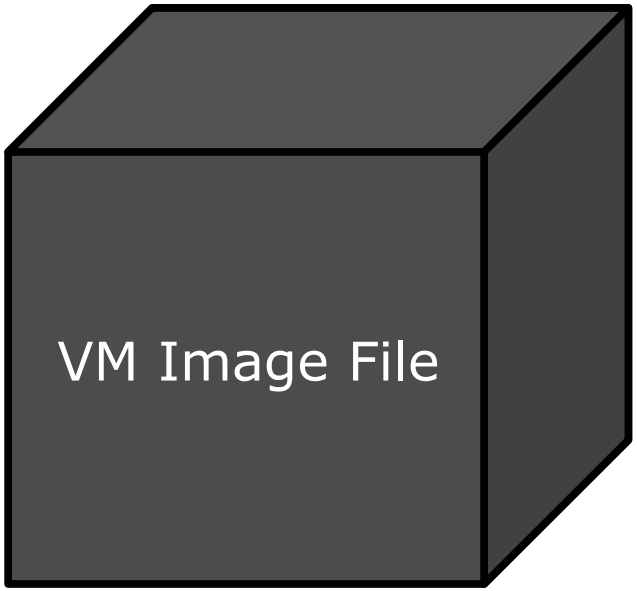
VM Images Sprawl



The Problem of VM Image Sprawl

- **VM images proliferate – Many images**
- **Images are large - Multiple GBs**
- **Sprawl makes it difficult to:**
  - Store images efficiently
  - Manage images efficiently – Inventory control, customized deployment, software updates
- **We address these problems**

Why is it Difficult to Manage Sprawl?



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

Our Solution: Mirage

- **The Mirage project addresses large-scale virtual machine management**
- **Created a new format: Mirage Image Format (MIF)**
  - Designed for efficient image storage and management
  - Exposes semantic information buried in VM images
- **Created new tools that exploit MIF to make management tasks faster and simpler**

Semantic Information Buried in VM Image

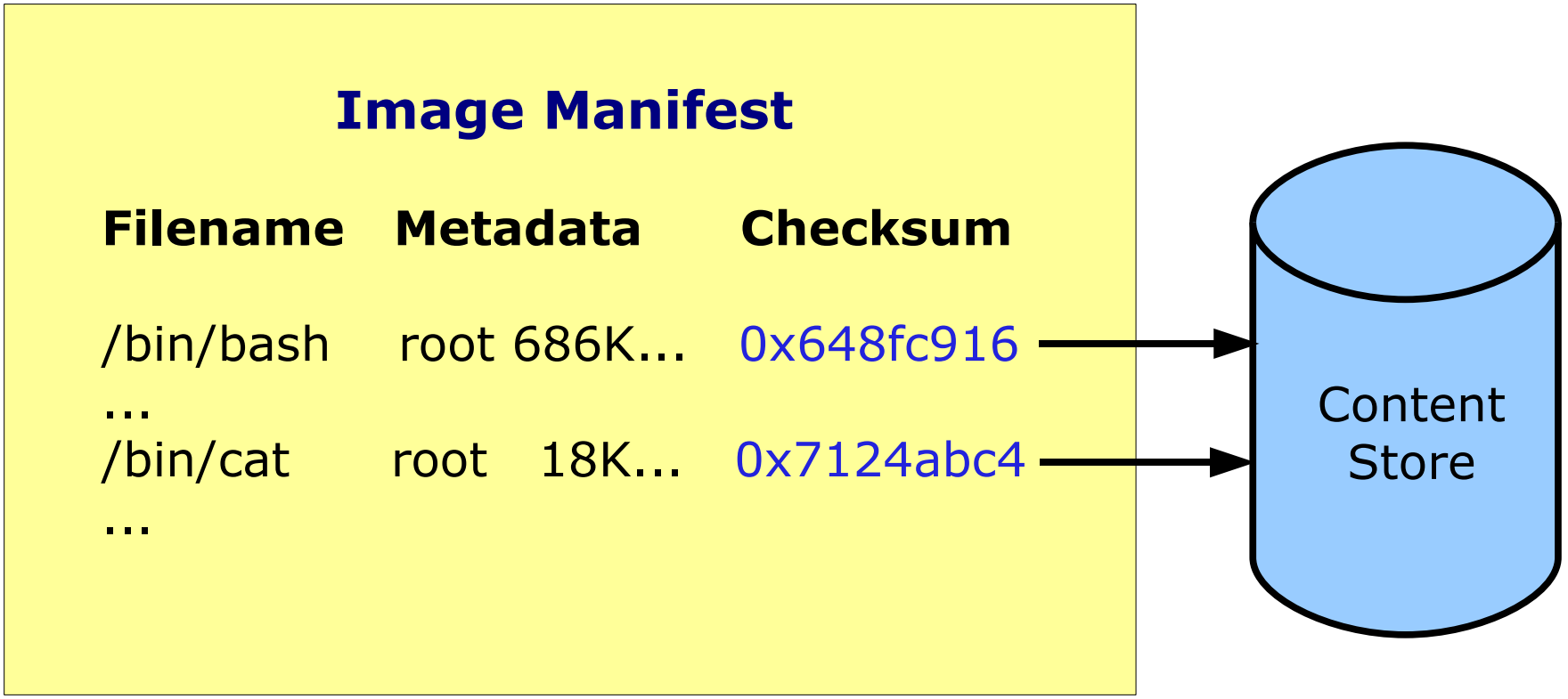
Mapping from Filename to File Content/Metadata

VM Image File

Filename		File Content	+	Metadata
/bin/bash	→	<01010101...>	+	<root, 686K, ...>
....				
/bin/cat	→	<11011101...>	+	<root, 18K, ...>
....				
/etc/hostname	→	<00011010...>	+	<root, 5K, ...>
....				

Manifests: Exposing Semantic Information

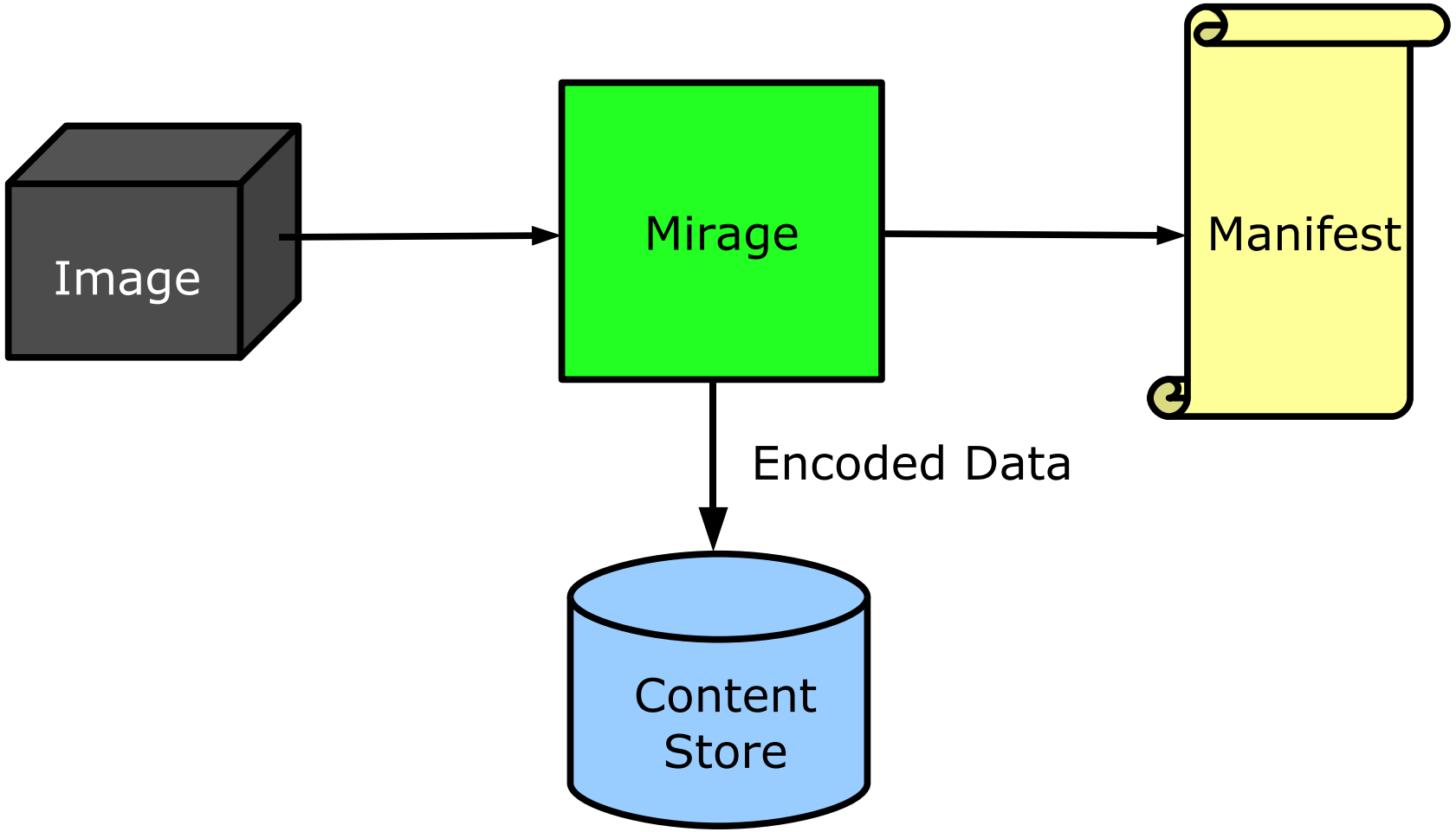
- **Manifest captures image metadata**
- **Content Store holds all data (file contents)**



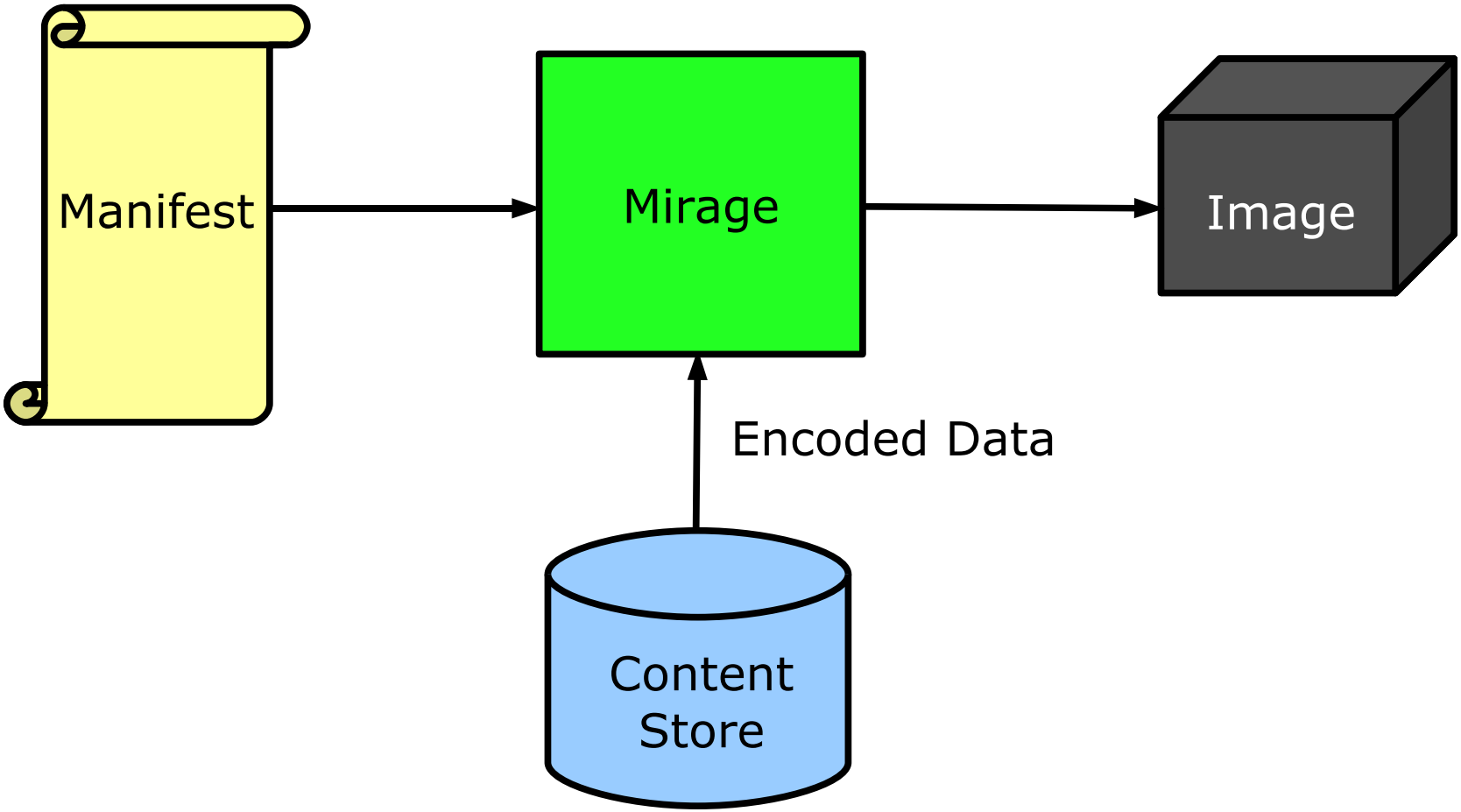
Manifests: Beneficial Characteristics

- **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)

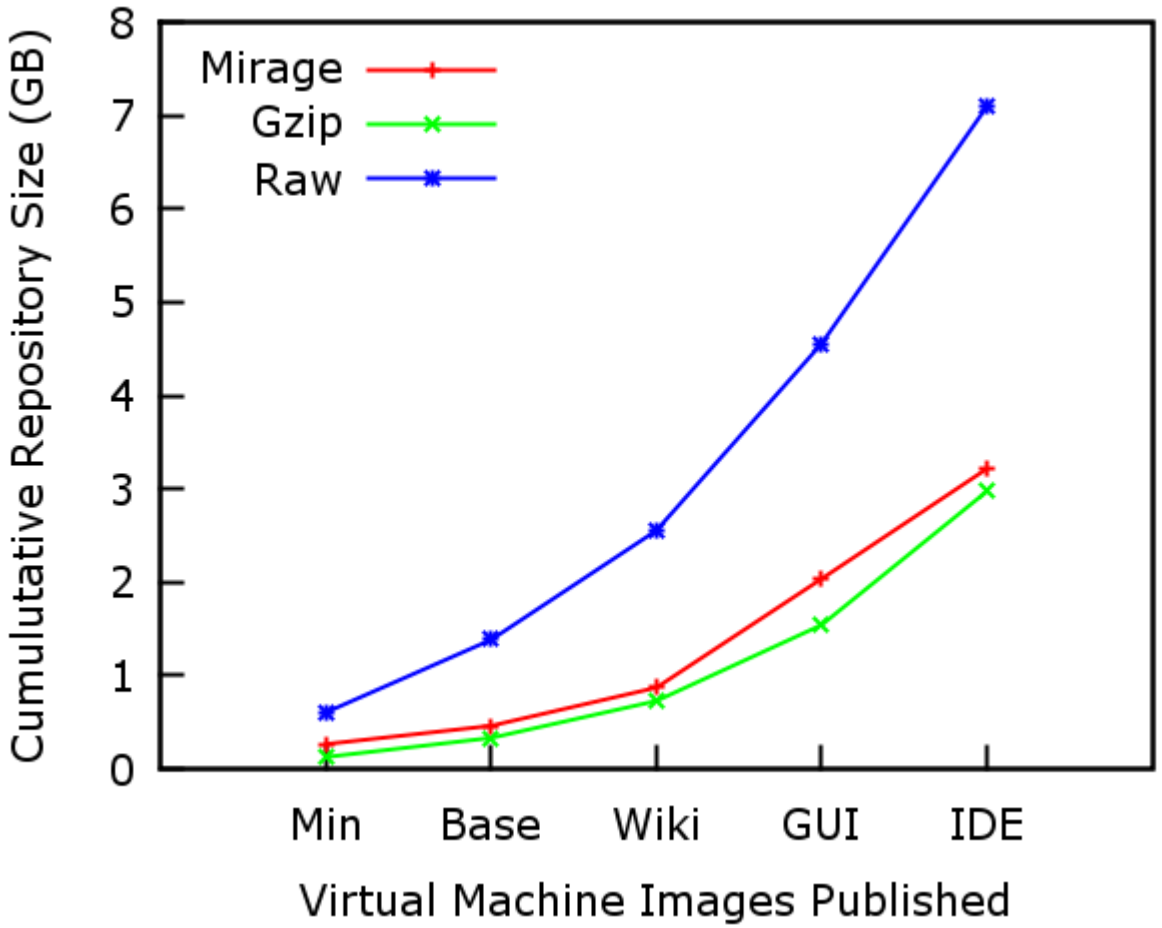
Mirage Publish/Retrieve Results

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

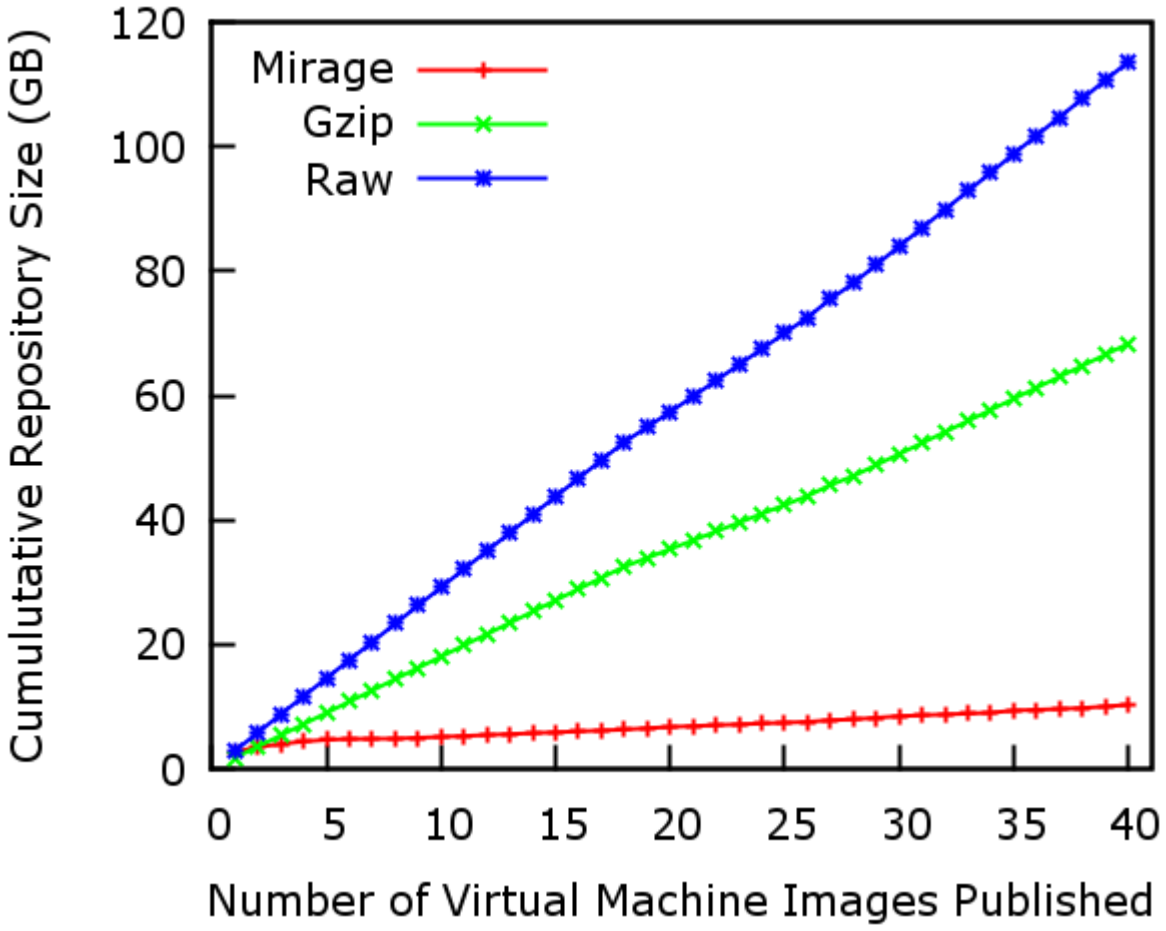
Mirage Storage: Debian Images

Mirage reduces storage requirements by 2.2x



Mirage Storage: Daily IDE Build Images

Mirage reduces storage requirements by 10.9x



Mirage in Action – Software Management (The Good Stuff)

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

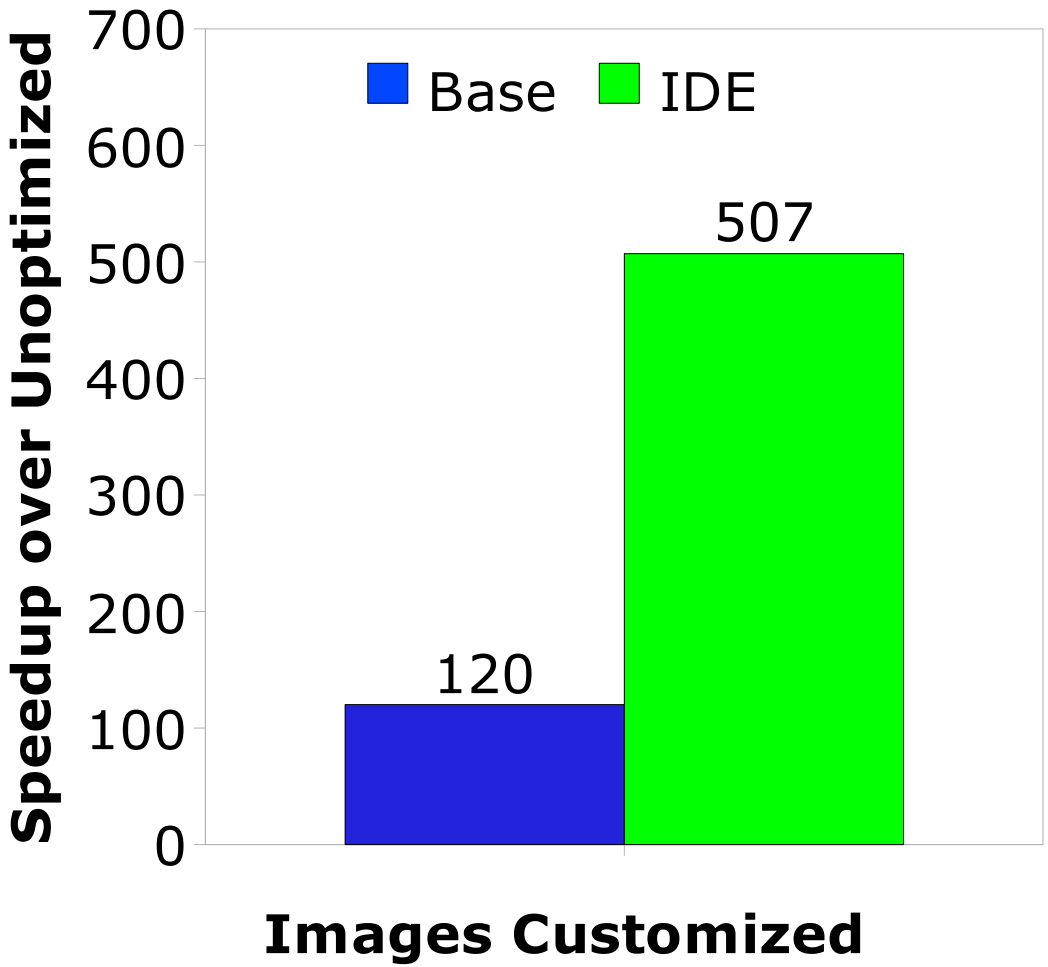
Name	Image Size (MB)	Lookup Time (sec) 1 File	1000 Files
Min	280	0.5	1.2
Base	450	1.1	1.3
Wiki	840	1.6	1.9
GUI	1670	2.2	3.0
IDE	2240	2.6	3.2

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.

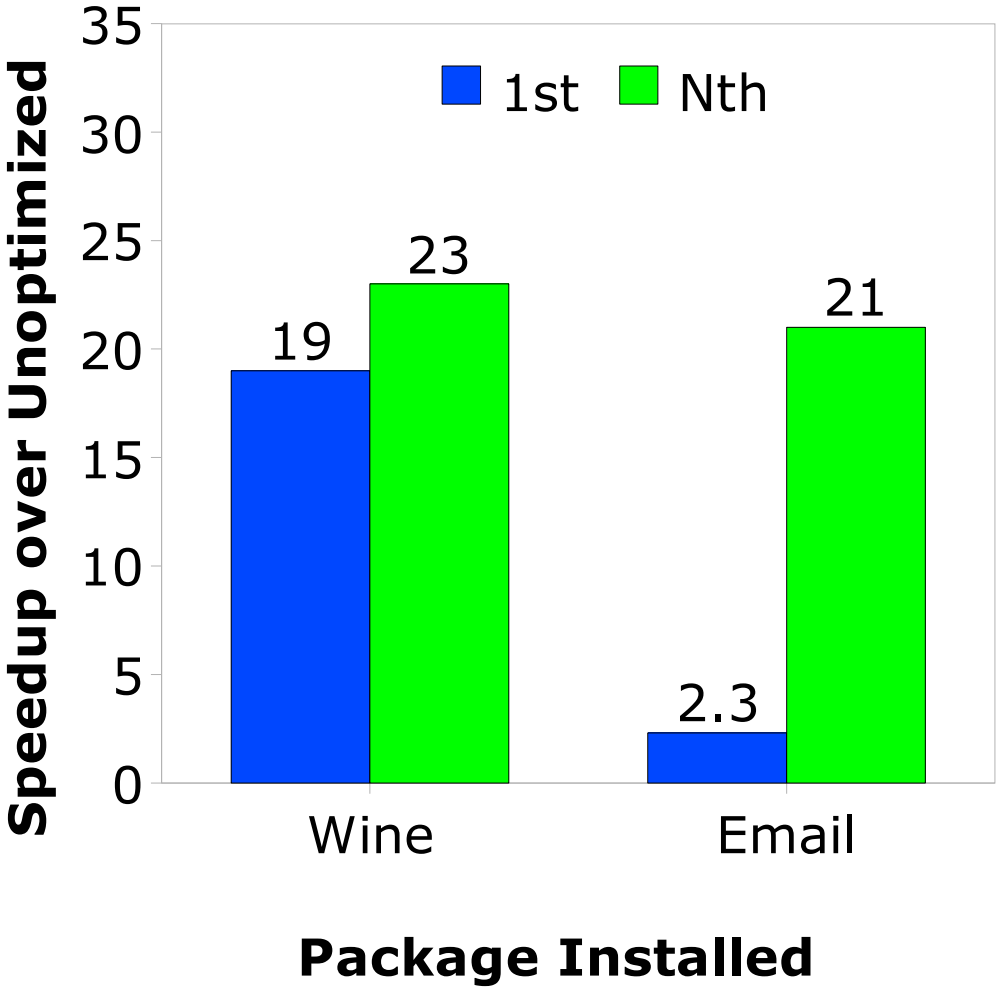


Scenario C: Install a Package

- **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)
- Machine Bank (Microsoft)
- Moka5 Engine (Moka5)
- Lab Manager/Update Manager (VMWare)
- rBuilder (rPath)

Ongoing and Future Work

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

Summary

- Created new image format (MIF) that is better suited to VM management
- MIF brings out semantic information buried in VM image
- MIF improves storage efficiency of large number of images
- MIF simplifies and speeds 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