VMTorrent: Virtual Machines On-Demand

Vishal Misra

CS @CU COMPUTER SCIENCE AT COLUMBIA UNIVERSITY

Virtual Appliances are virtual machines (VMs) created to run specific tasks, e.g. web application servers or office productivity tools. Virtual Appliances are becoming increasingly prevalent, and fit neatly in the Cloud paradigm.

Problem: How to deploy Virtual Appliances *immediately* and *scalably* (robust to flash crowds) in the datacenter, enterprise, or home?

Solution: VMTorrent efficiently deploys and executes VMs just-in-time, leveraging:

- *VM quick starting*
- *P2P delivery*

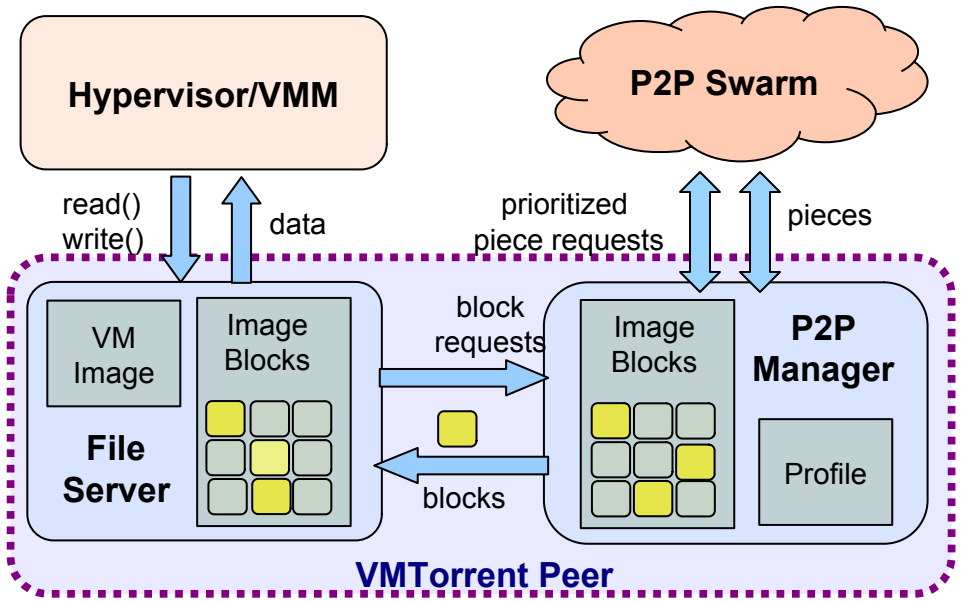
Quick starting of VMs achieved using file server that facilitates on-demand access to VM image, while blocks are fetched from P2P network.

VMTorrent is agnostic to particular VMMs. It presents a mutable copy of the VM image, abstracting missing blocks to allow VM execution with *only partial* download.

P2P delivery achieved using P2P manager that downloads (and uploads) VM image pieces in the background.

Intelligent piece selection strategies minimize VM stalls due to missing blocks by adjusting priorities:

- Reactively – in response to VMM demand
- Proactively – based on profiling of VM execution



VMTorrent Peer Operation

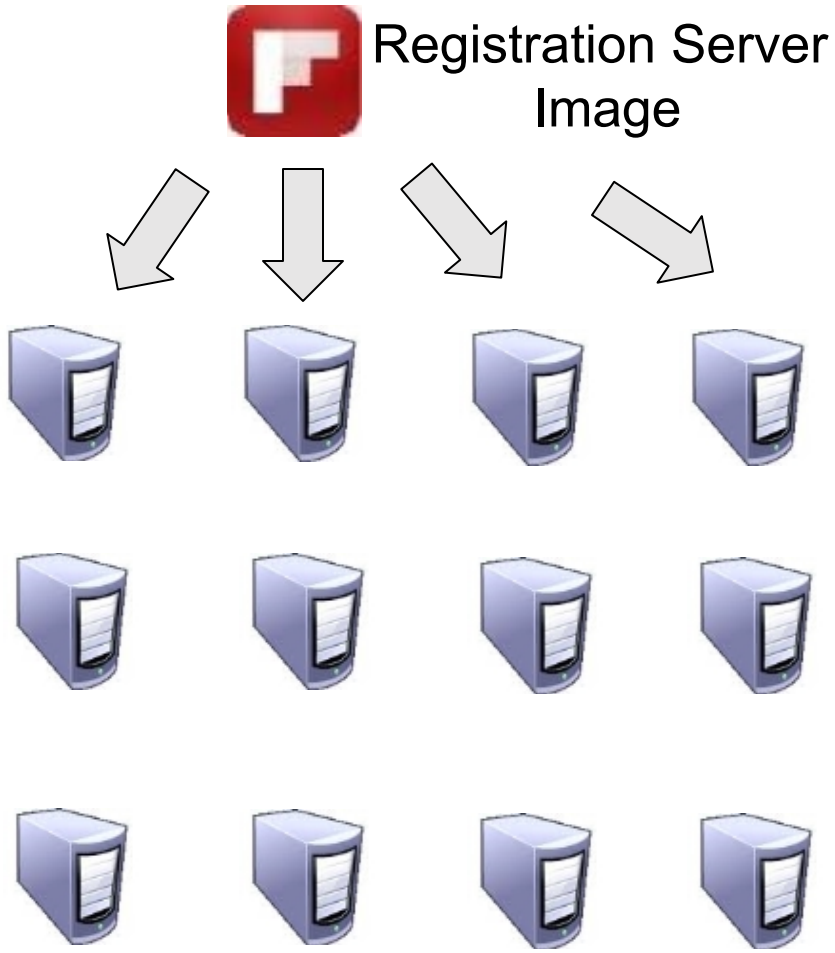
- Begin VM image download from P2P when VM spawned
- (Pre-) Fetch urgent blocks to minimize VM stall time
- Consult VM profile for optimal streaming ordering
- Keep pristine copy of VM image to serve other peers

VMTorrent Results on Virtual Appliance Workloads:

- Complete execution w/ only 10-20% image downloaded
- User-experience comparable to using a local VM image
- *Faster* and more *scalable* than traditional image streaming

The Problem: Distribution

- Context
  - Cloud
  - Enterprise
  - Home
- **Immediate availability**
  - SLAs in the Cloud
  - Impatient users at home
- **Scalable to demand**
  - Robust to flash crowds
  - e.g., Flipboard weeks to sign up all users attempting to register in one night!



The Solution: VMTorrent

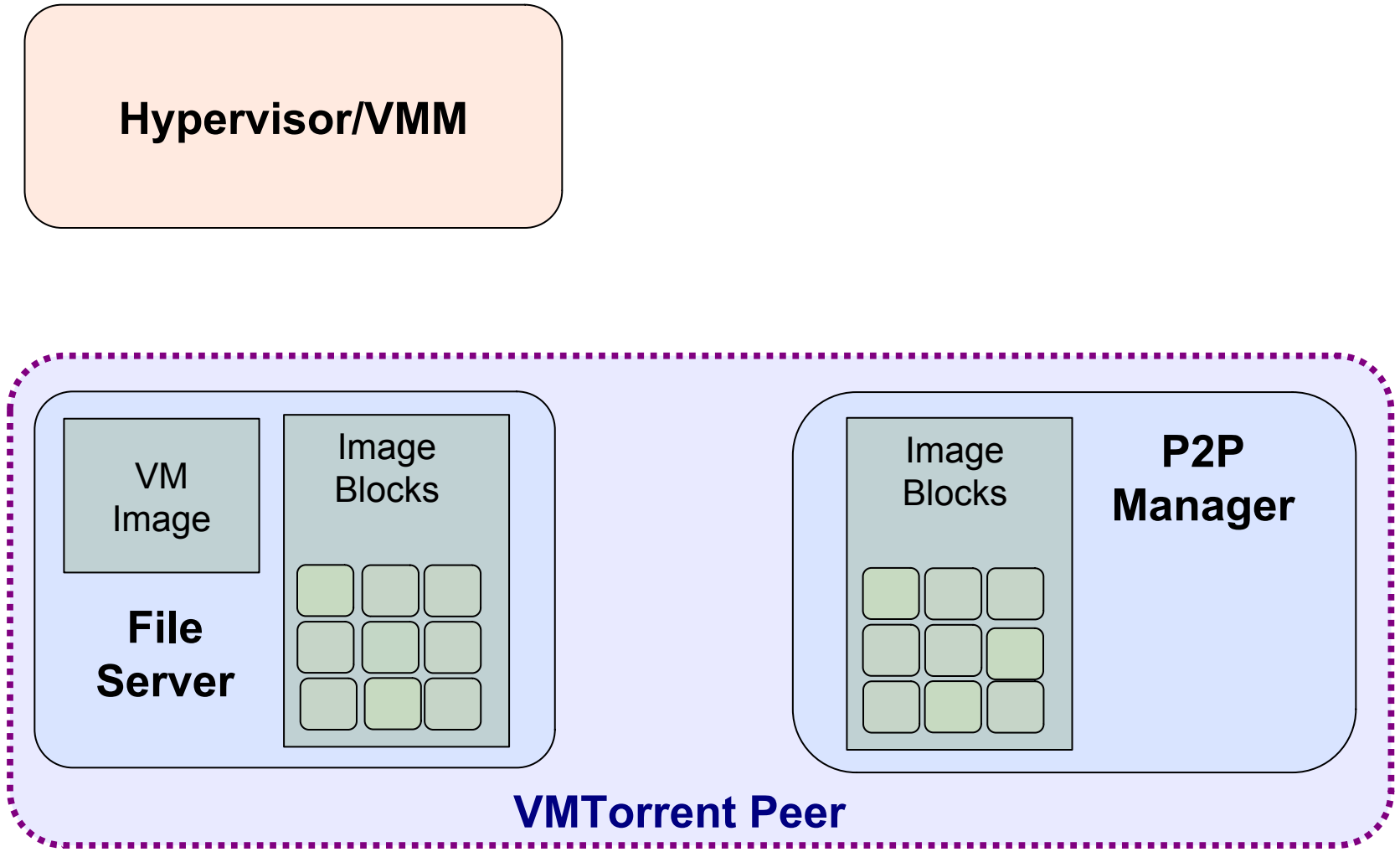
Efficiently deploys and executes VMs *just-in-time*.

- ***VM quick starting*** **(Immediate availability)**
  - File server facilitates on-demand image access
  - Blocks are (pre-)fetched from network

- ***P2P delivery*** **(Scalable to demand)**
  - P2P manager downloads and uploads VM image pieces
  - Intelligent piece selection strategies minimize VM stalls due to missing blocks by adjusting priorities:
    - **Reactively** – in response to VMM demand
    - **Proactively** – based on profiling of VM execution

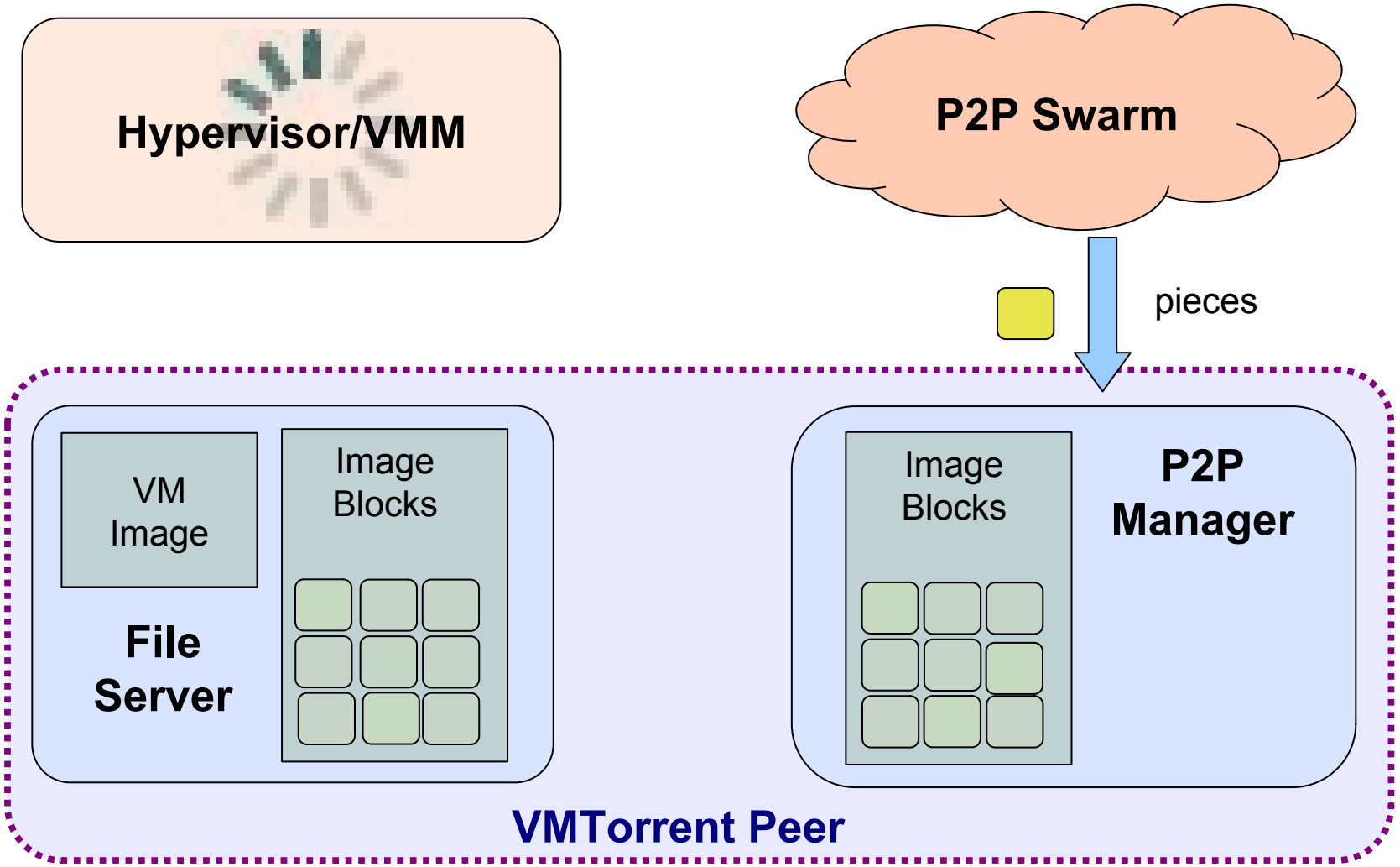
How VMTorrent Works

- VMM and VMTorrent run on same peer
  - Server, Workstation, Laptop



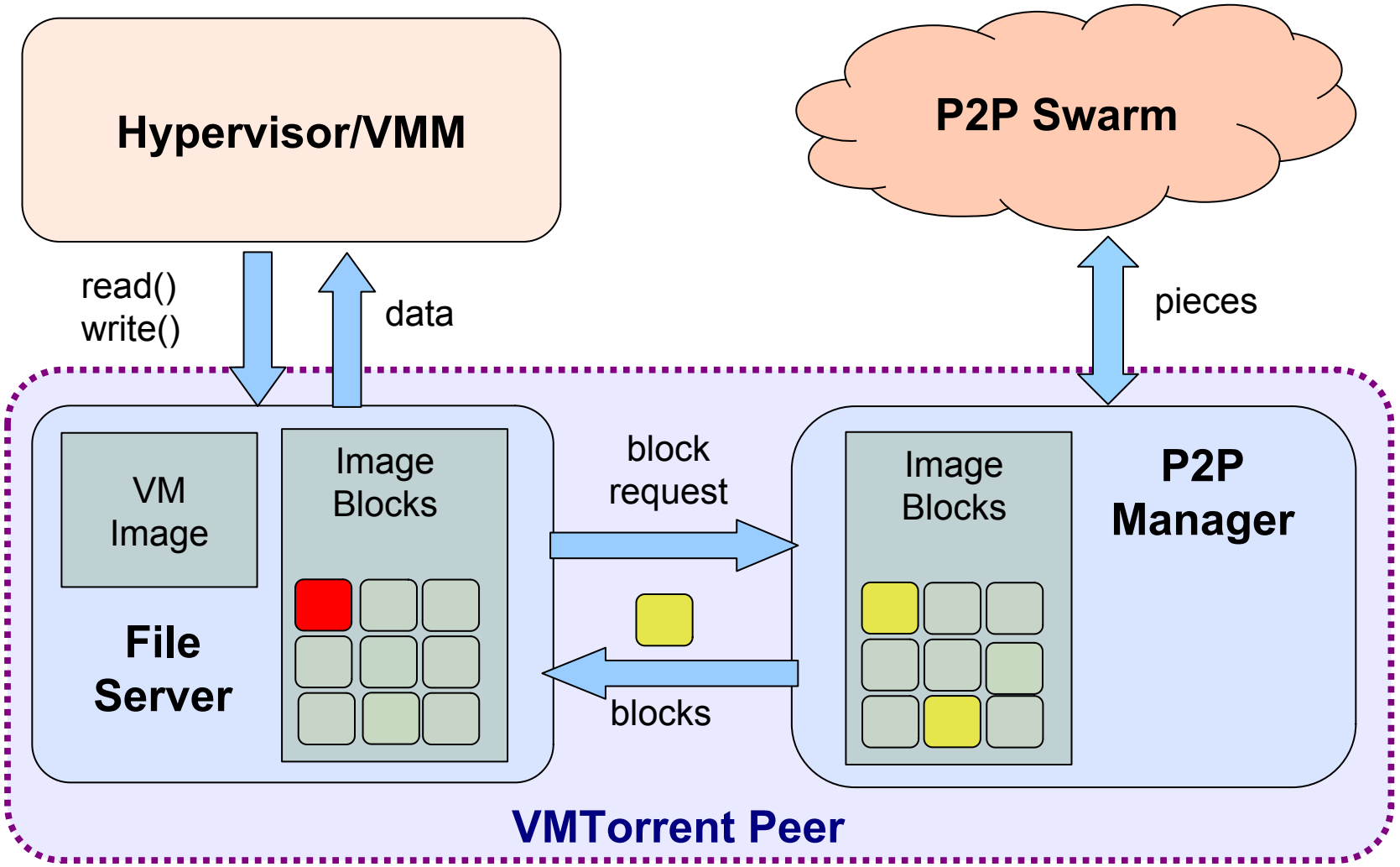
How VMTorrent Works

- Hypervisor begins starting up to run a given virtual appliance
- File Server presents apparently complete image though file system
- P2P Manager starts downloading blocks from swarm



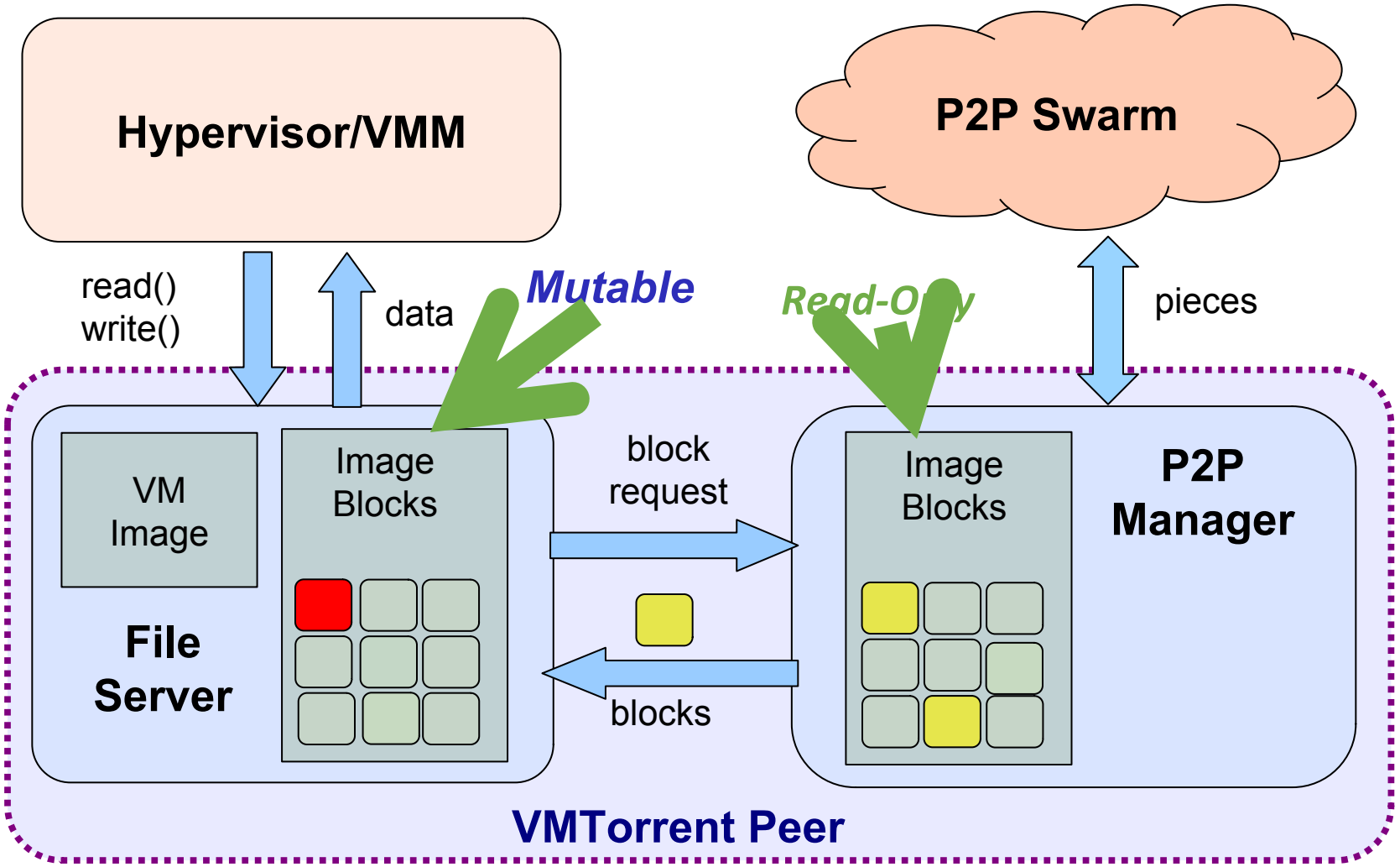
How VMTorrent Works

- Hypervisor begins accessing image
- File Server sends request to P2P Manager for missing blocks
- P2P Manager serves downloaded blocks to File Server



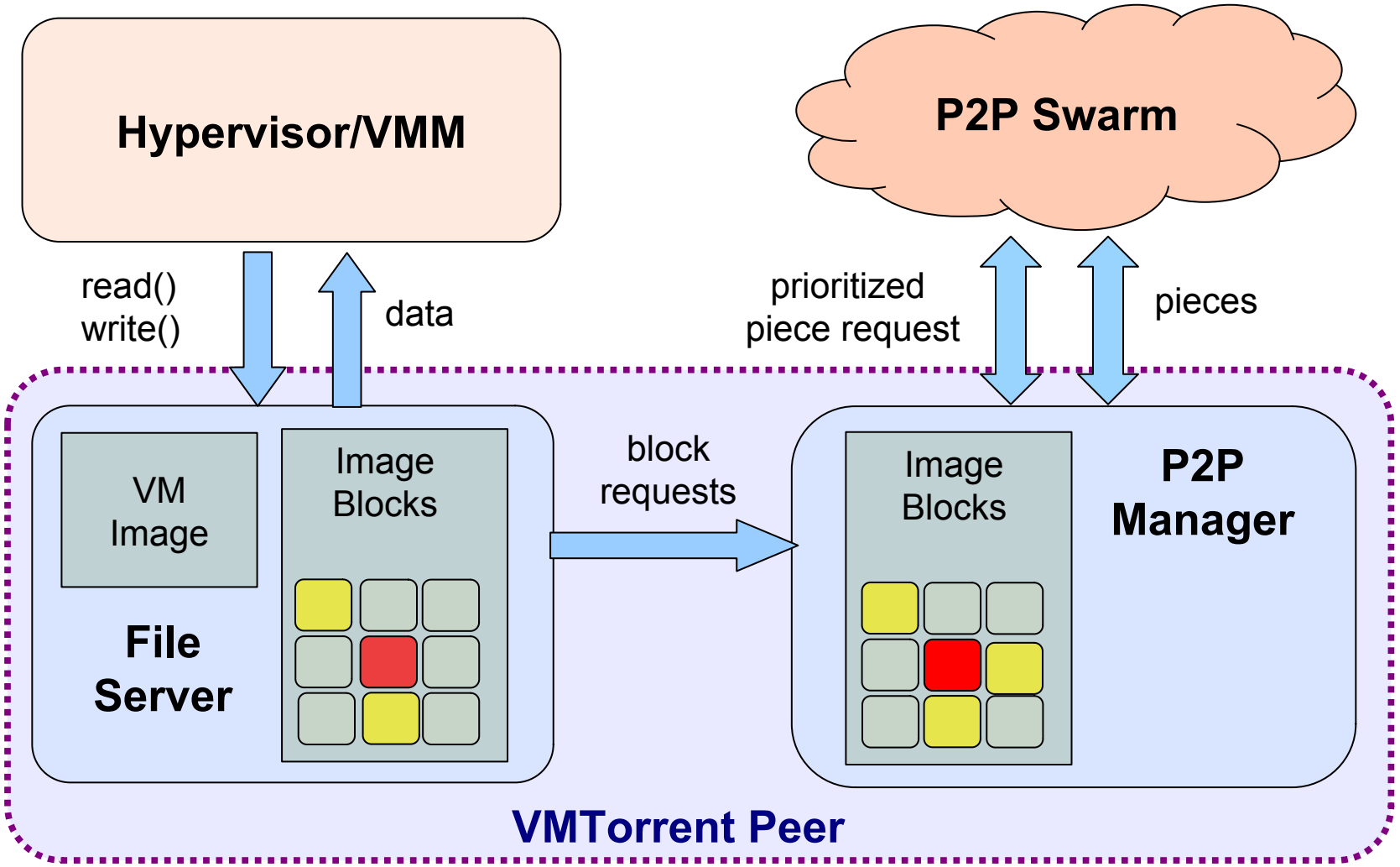
How VMTorrent Works

- P2P copy is read-only, FS copy is read-write
- Essentially a P2P-Filesystem that is *read-only from network* and *locally mutable*



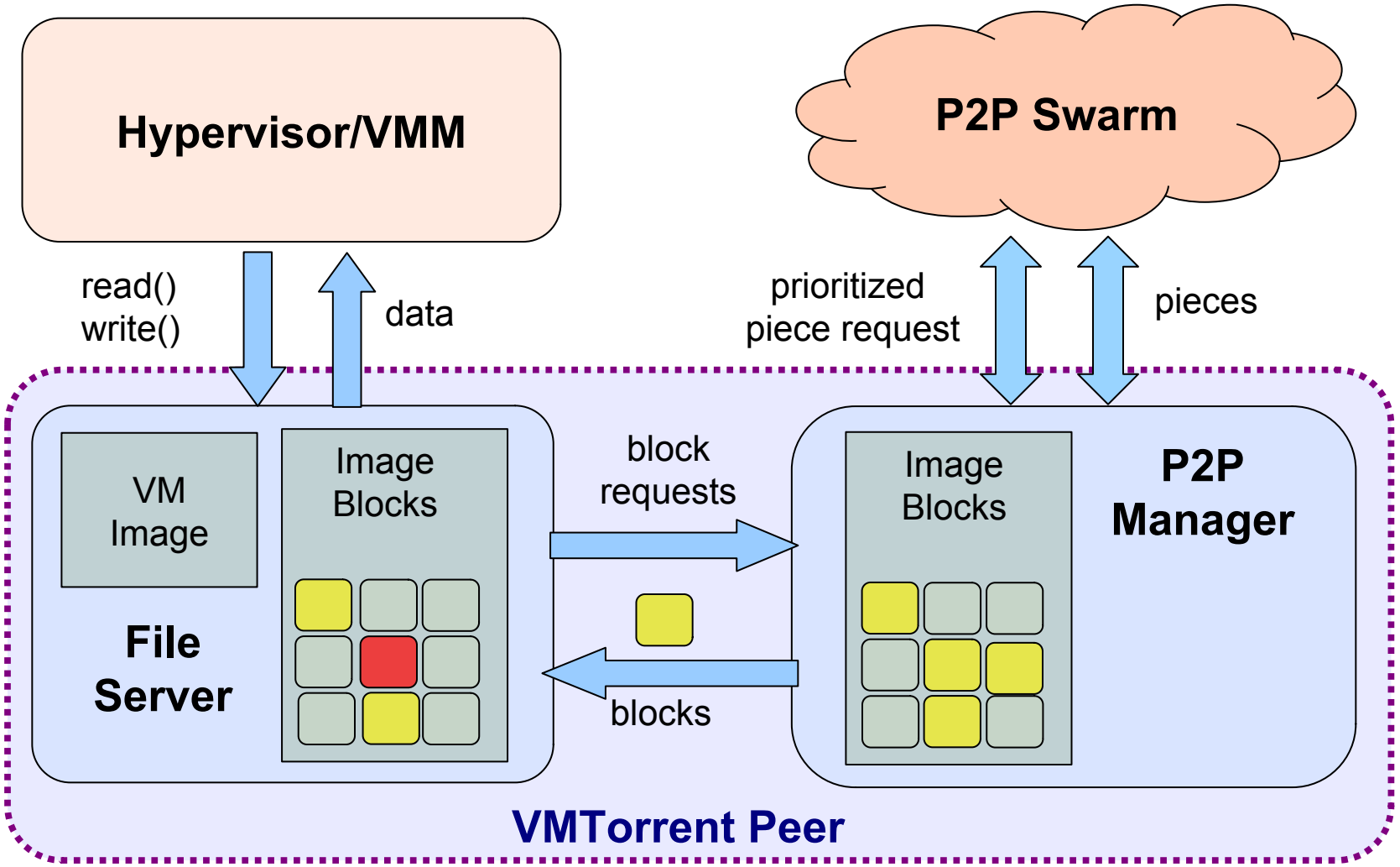
How VMTorrent Works

- When File Server requests block not yet downloaded
- P2P Manager issues highest priority demand request to swarm
- Read or write call waits to return, VMM stalls



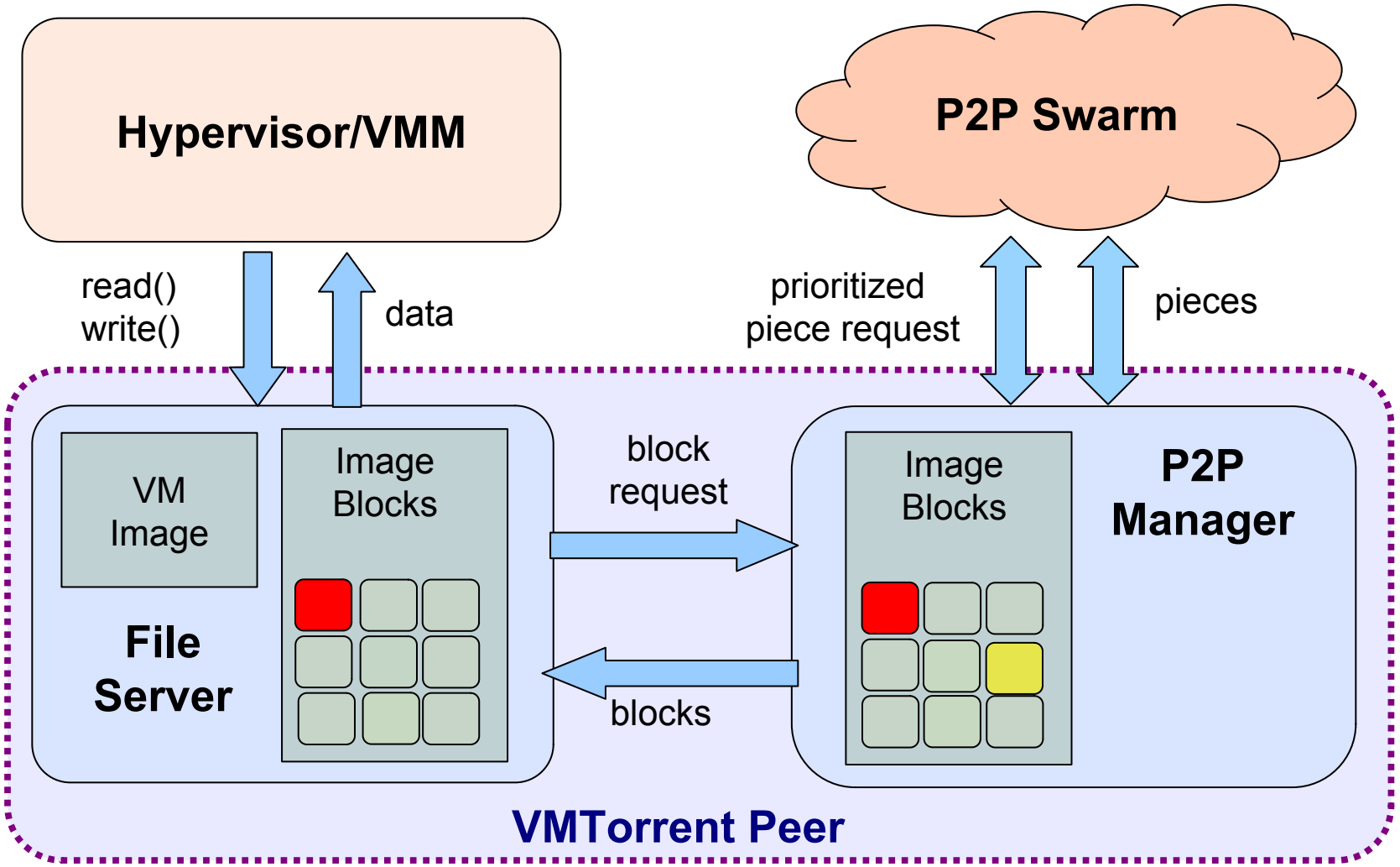
How VMTorrent Works

- As soon as piece received from swarm (<0.05s)
- P2P Manager sends block to File Server
- VMM resumes



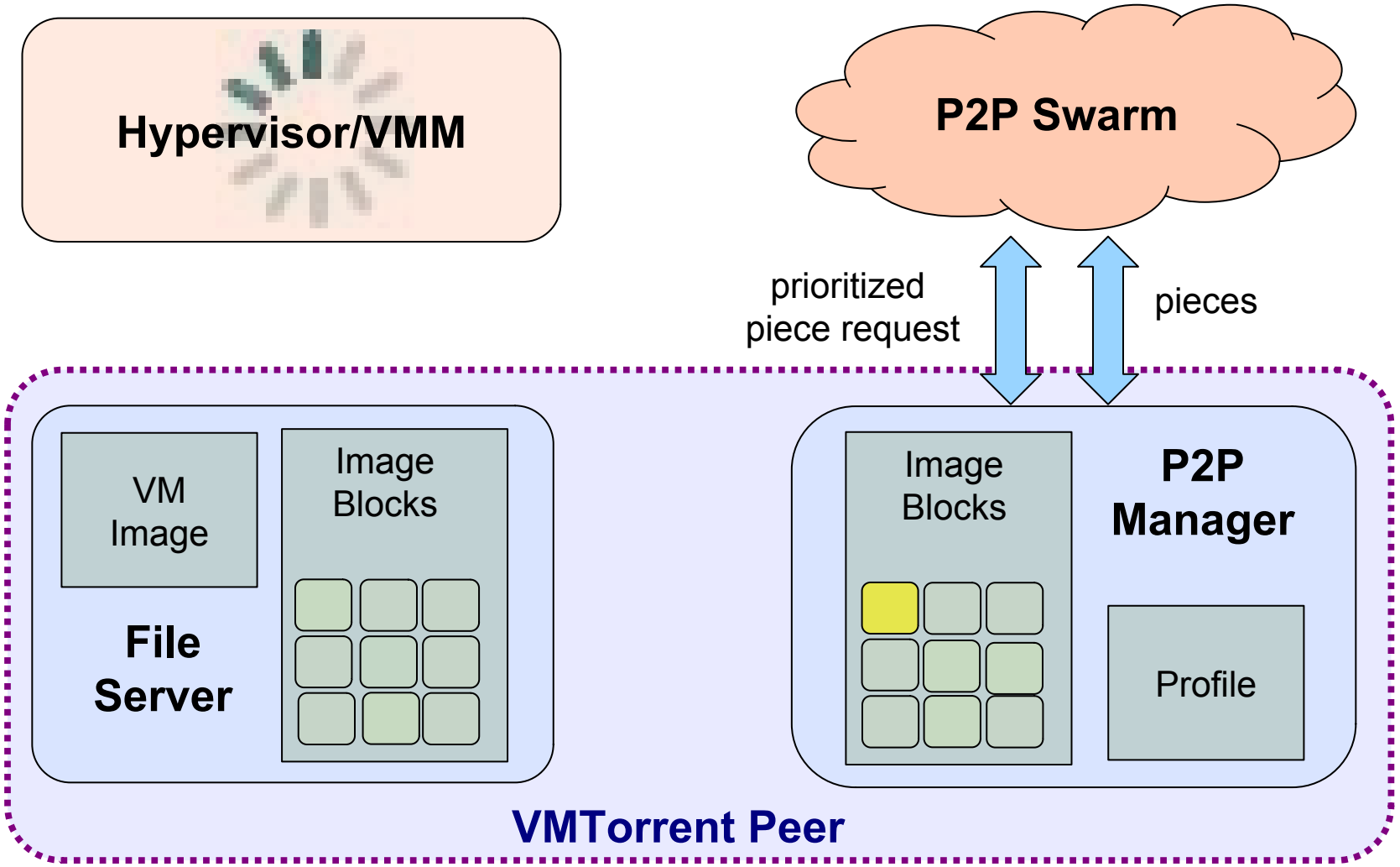
How VMTorrent Works

- But what if many peers start up simultaneously?
- We will bottleneck when all peers demand same block at boot!



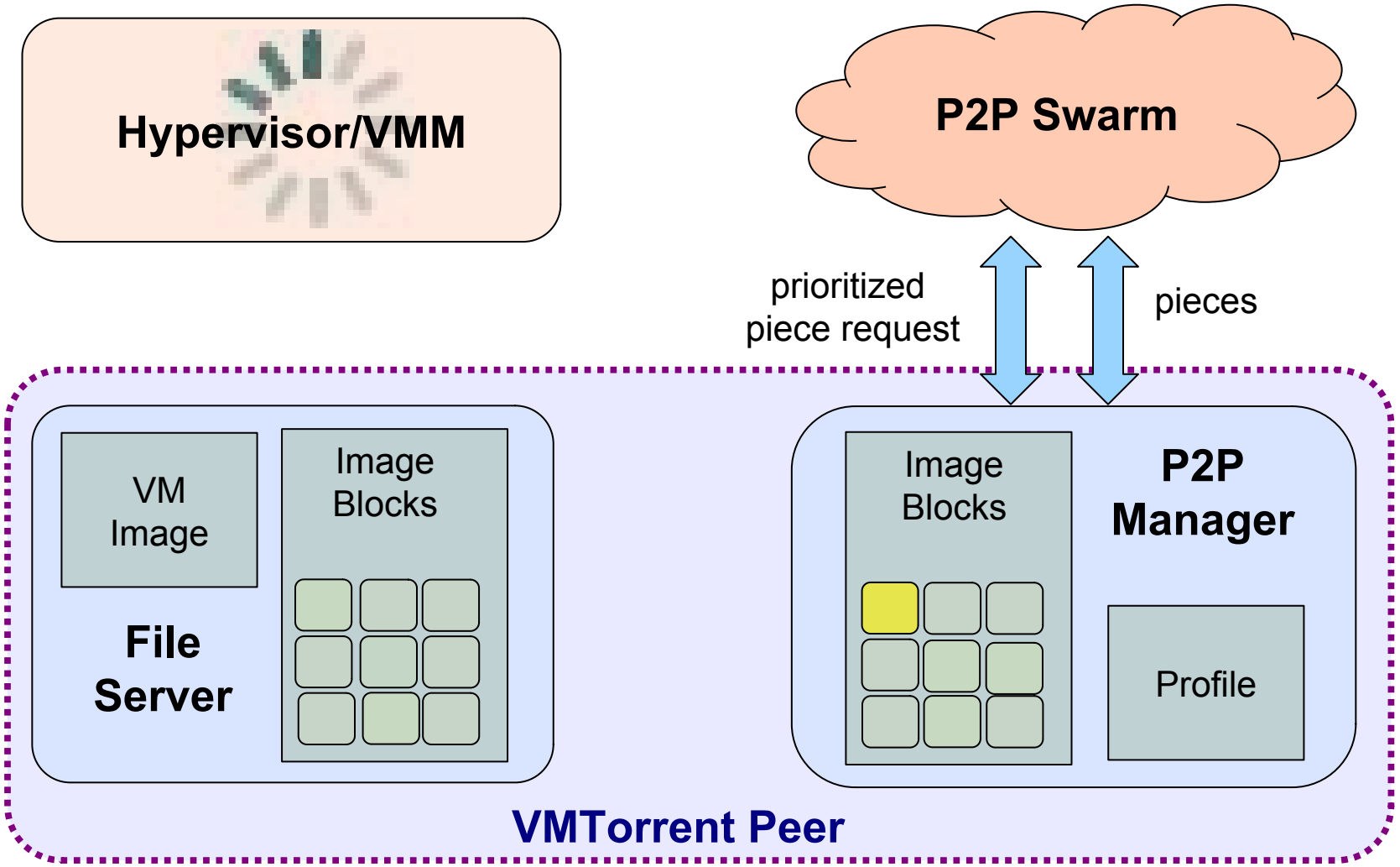
How VMTorrent Works

- To avoid this, we profile appliance behavior (block access sequences, delays, variance, etc.)
- P2P Manager pre-fetches (at 2nd highest priority) from profile



How VMTorrent Works

- Requests occur over window (e.g., first 20 pieces) of profile
- Ensures sufficient piece diversity in swarm to avoid bottleneck
- Window size may vary dependent on # of peers starting



The Results

- VMTorrent ***agnostic to VMM***
  - Level 1 VMMs would require vendor support
- ***Portable*** to all popular OSes (coded w/ FUSE)
- Able to completely run tasks
  - Downloading only *10-20%* of blocks
  - *Download amortized* over VMM/VM compute time
  - Even better on bloated VMs
- W/ high bandwidth
  - C-S/Small swarm user experience *comparable to local boot*
  - Currently increasing testbed size to determine scaling properties
- W/ any bandwidth
  - C-S *outperforms* SSHFS (network filesystem)

Demo