Perspectives on Virtualized Resource Management

Carl Waldspurger

June 26, 2013

10th International Conference on Autonomic Computing
USENIX Federated Conference Week, San Jose

Resource Management

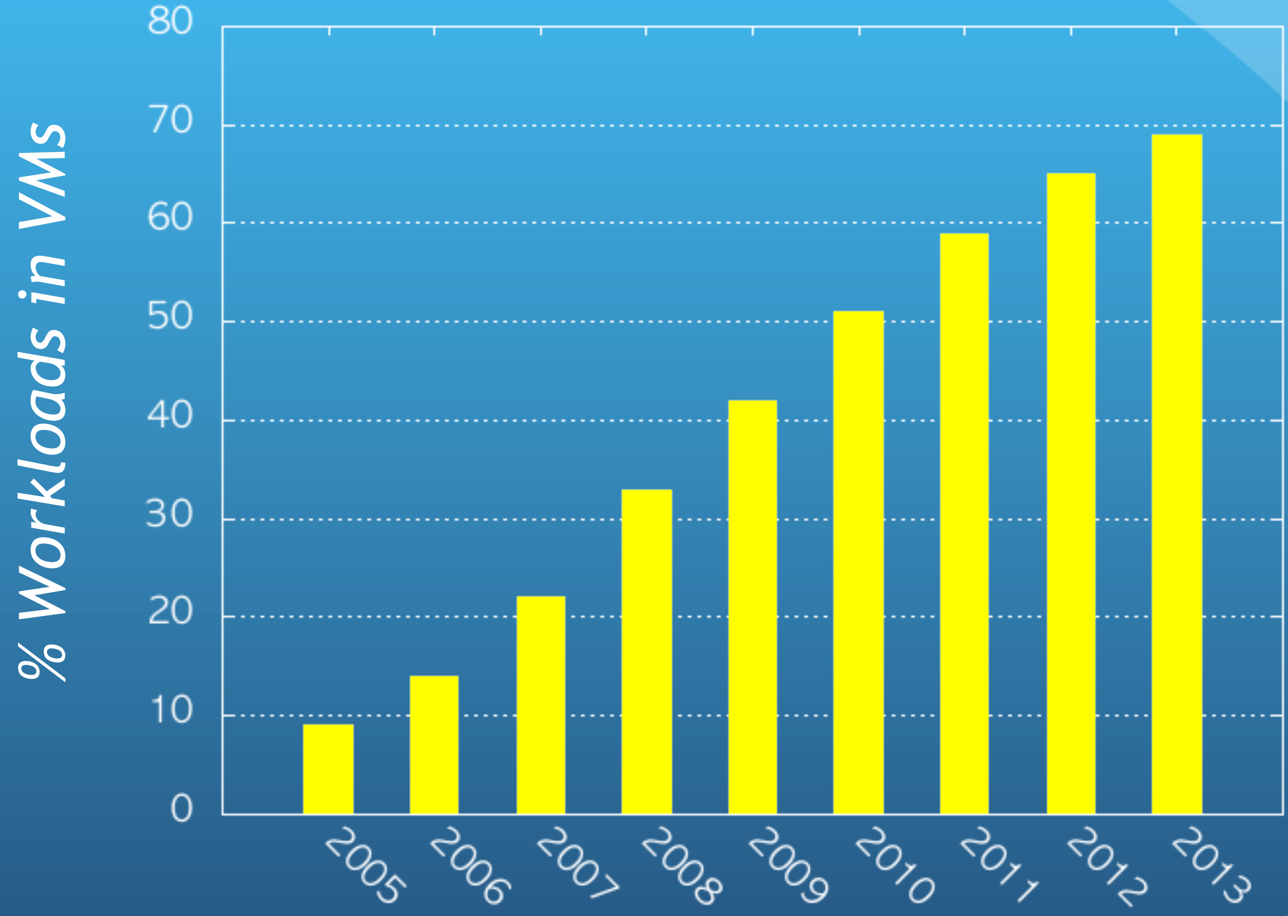
- Map workloads onto physical resources
- Varying importance
- Diverse resources, granularities
- Complex interactions

Virtualization

All problems in computer science can be solved by another level of indirection… *— David Wheeler*

- Hypervisor: extra level of indirection
- Powerful new capabilities

Virtualization: Wildly Successful



Source: IDC Server Virtualization Forecast

Indirection: Double-Edged Sword

… but that usually will create another problem.

— David Wheeler

- Performance isolation
- Semantic gap
- Complexity

My Vantage Point

- Research and product development
- Systems I’ve helped build

  Spawn (PARC), lottery/stride scheduling (MIT), DCPI and Itsy (DEC), ESX and DRS (VMware), …
- Challenges building autonomic systems

No Silver Bullet

Recurring Themes

- Randomization and sampling
- Indirection and interposition
- Semantic gap and transparency
- Hardware/software co-evolution

Path to Autonomic Systems

1. Measurement
2. Modeling
3. Mechanisms
4. Policies

If you can't measure something, you can't understand it. If you can't understand it, you can't control it. — *H. James Harrington*

1. Accurate Measurement

Profiling, accounting, virtualized timekeeping

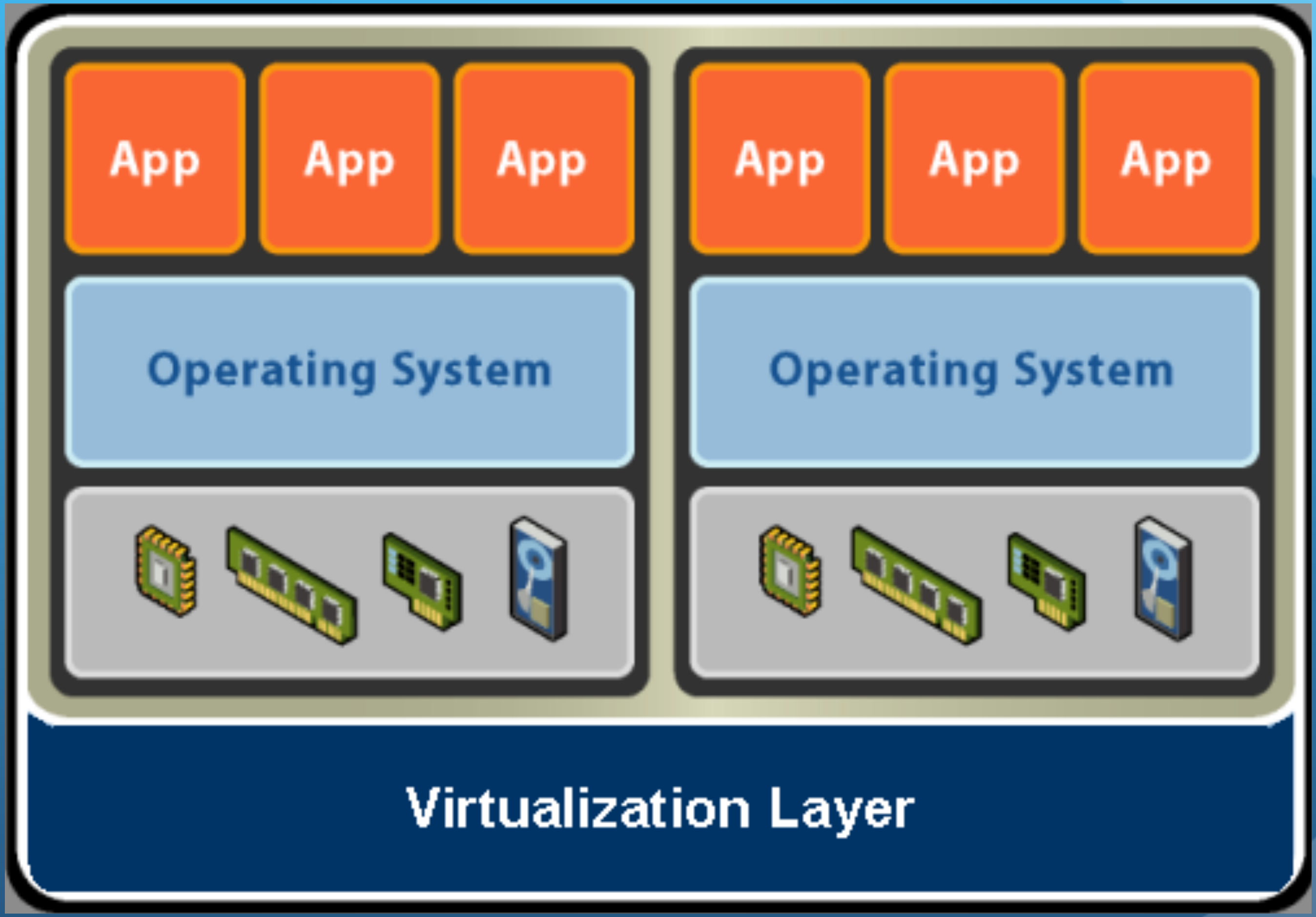
Measurements Gone Wrong

- Blind spots, distortions
- Statistical profiling
- CPU accounting
- Virtualized time-keeping

Virtualized Timekeeping

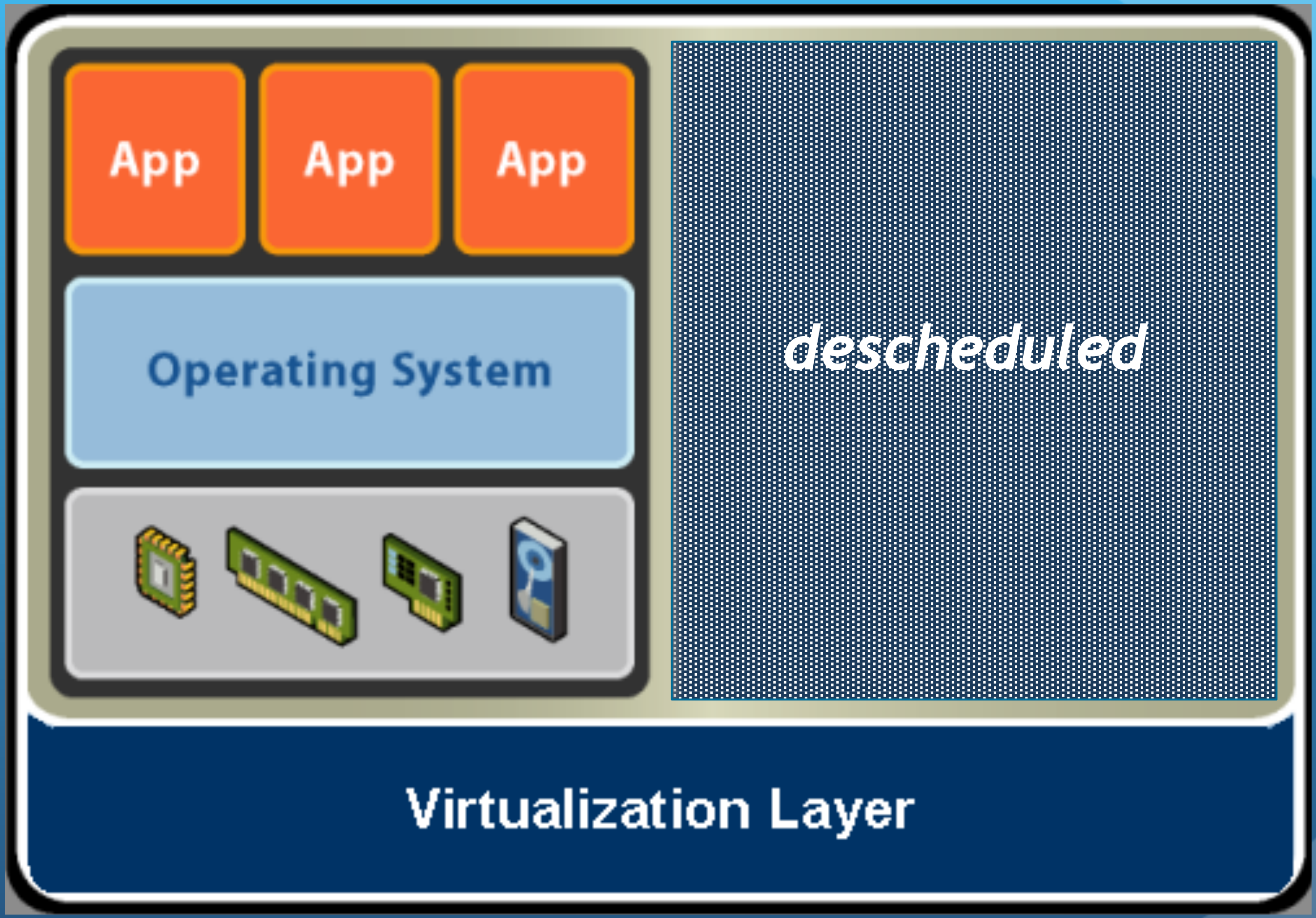
- Maintain illusion of dedicated system
- Periodic guest timer interrupts
  - Track passage of real time
  - Statistical process accounting
- What happens when VM descheduled?

Timer Interrupt Backlog



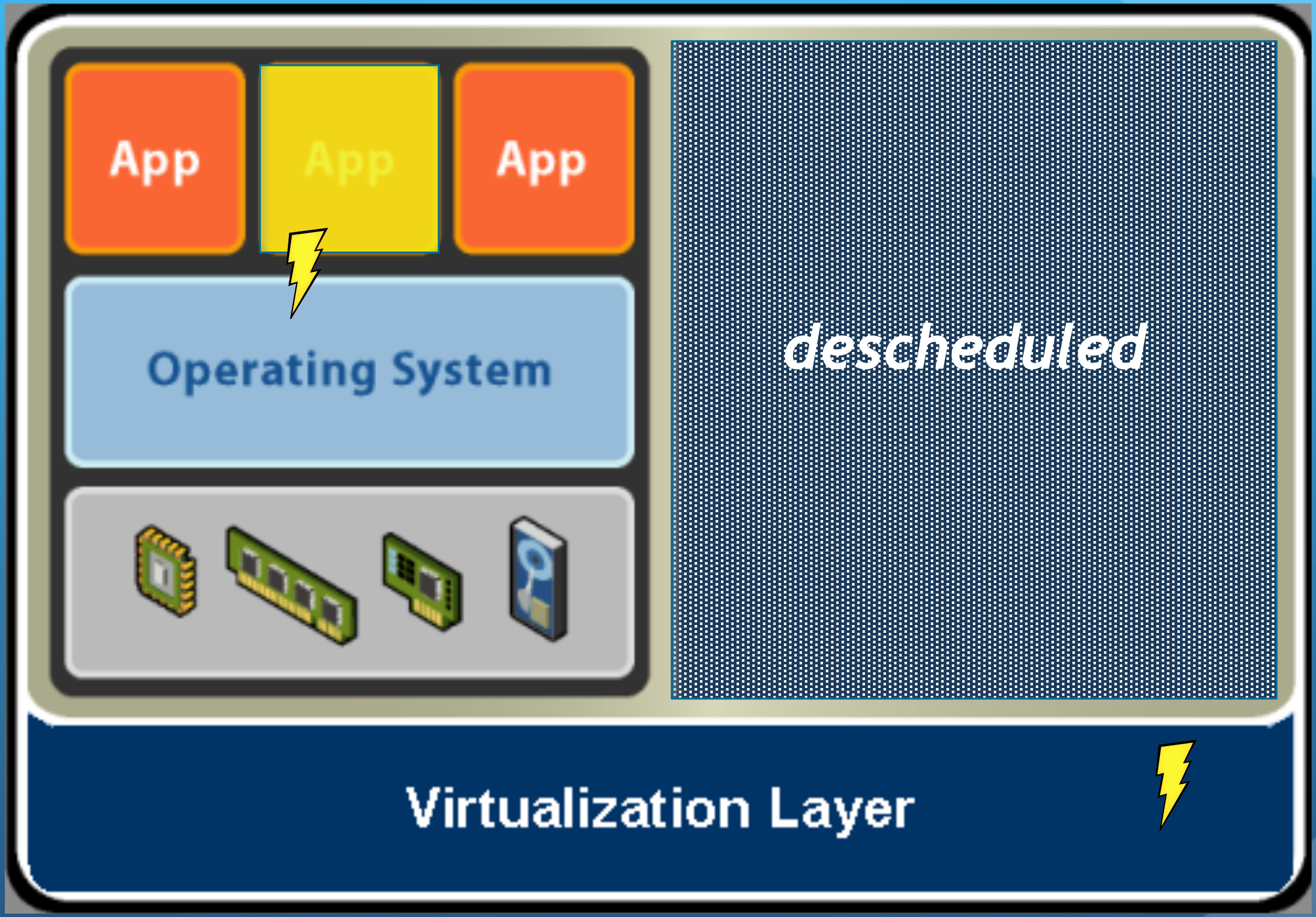
[Animation]

Timer Interrupt Backlog



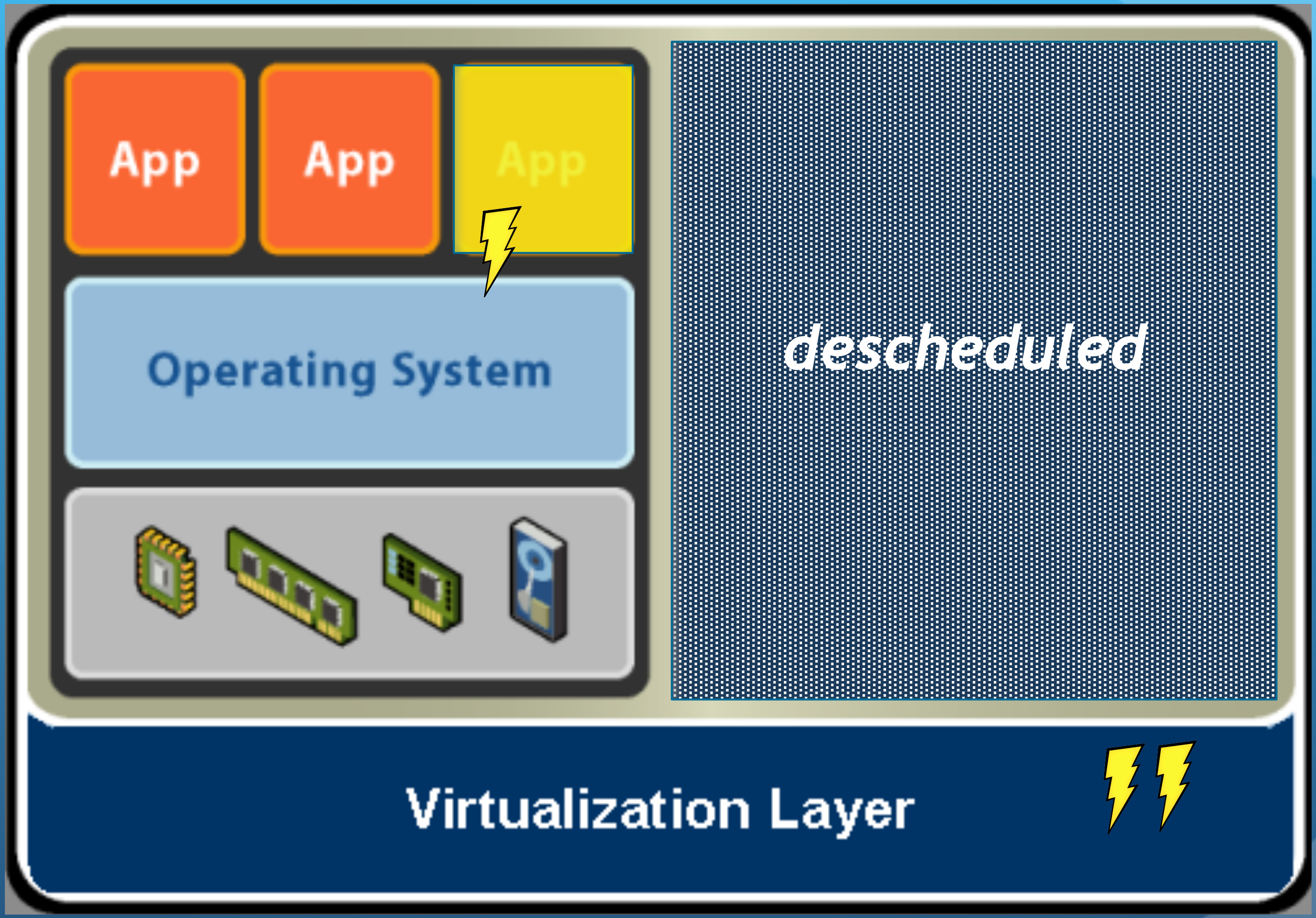
[Animation]

Timer Interrupt Backlog



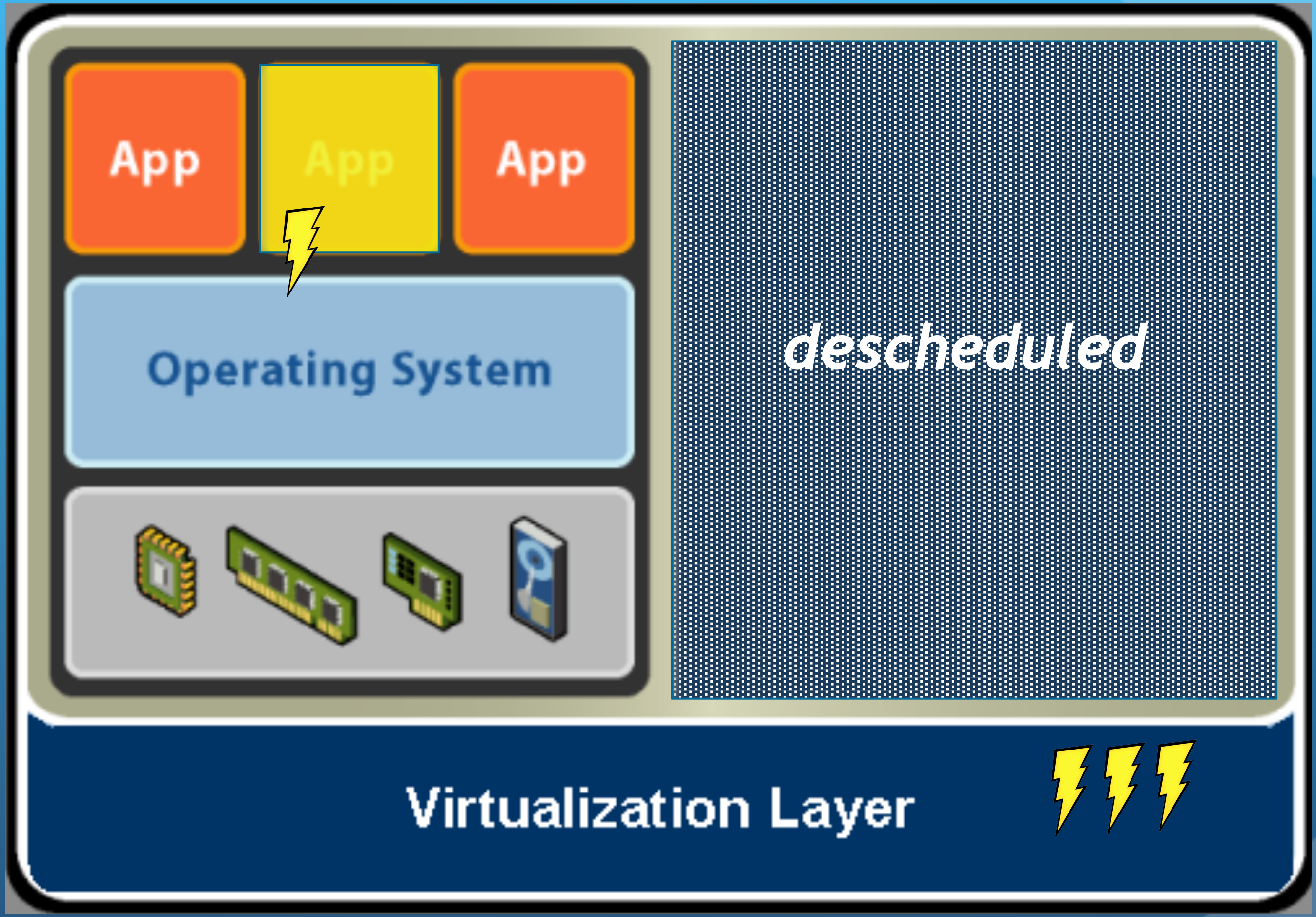
[Animation]

Timer Interrupt Backlog



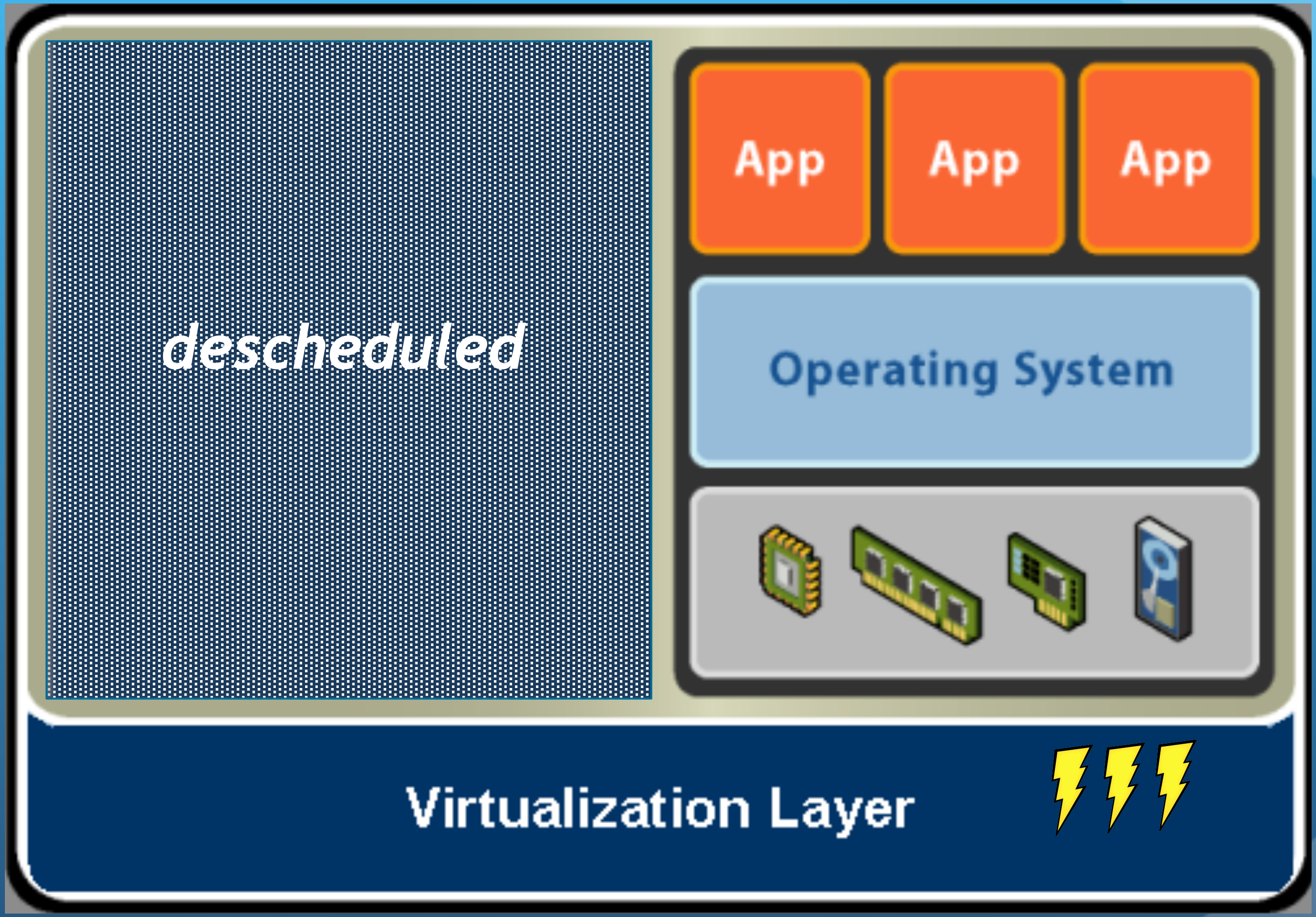
[Animation]

Timer Interrupt Backlog



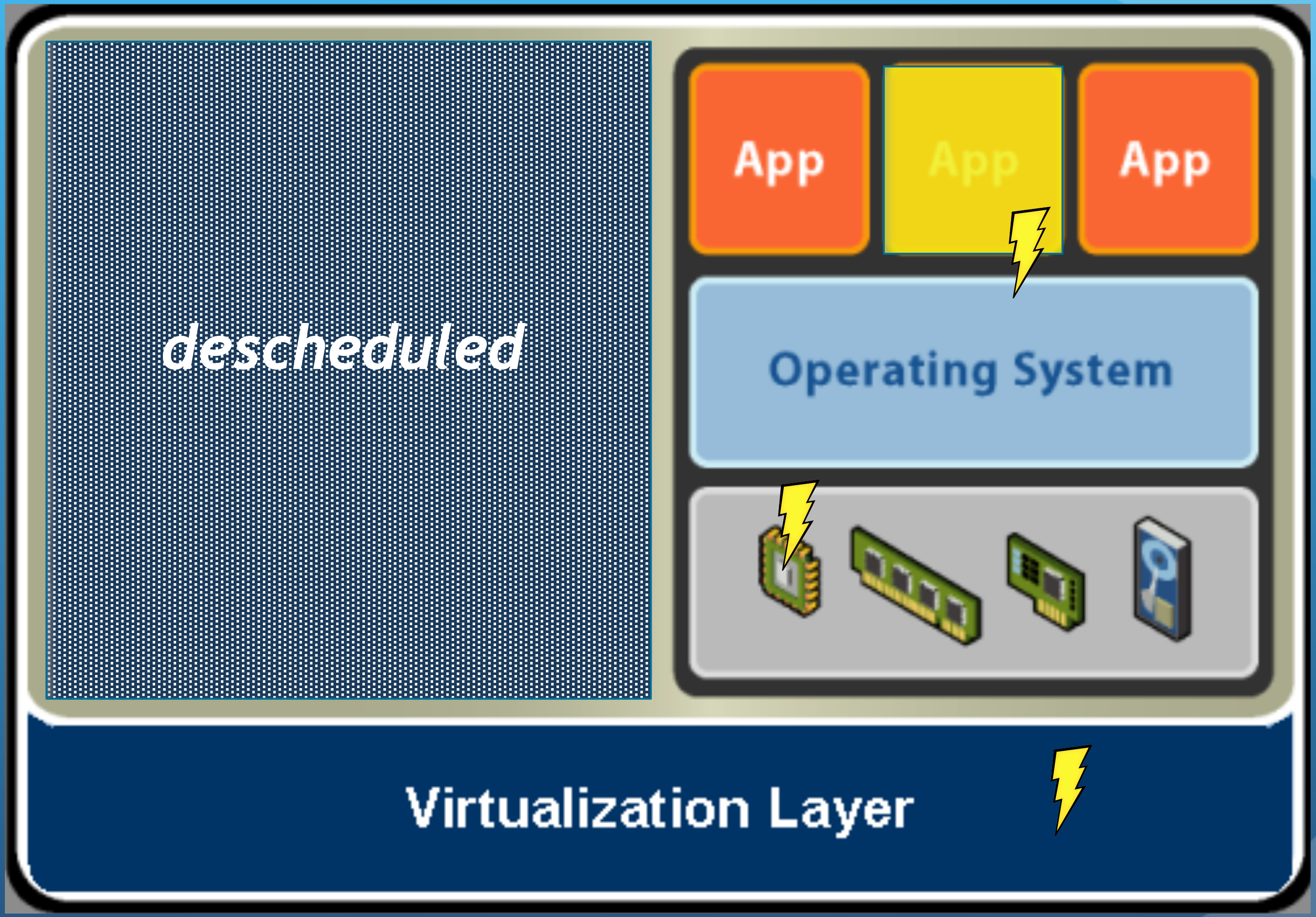
[Animation]

Timer Interrupt Backlog



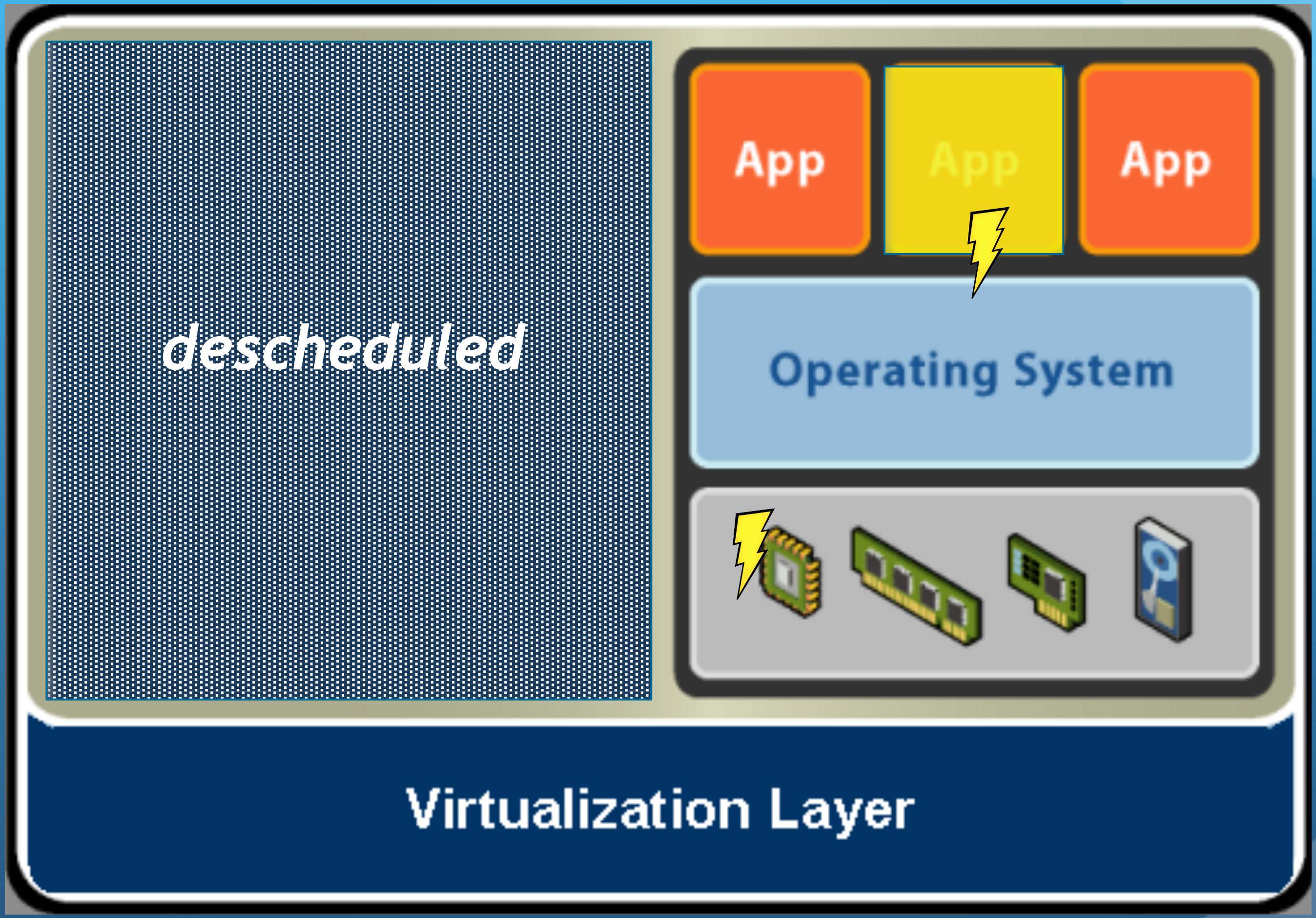
[Animation]

Timer Interrupt Backlog



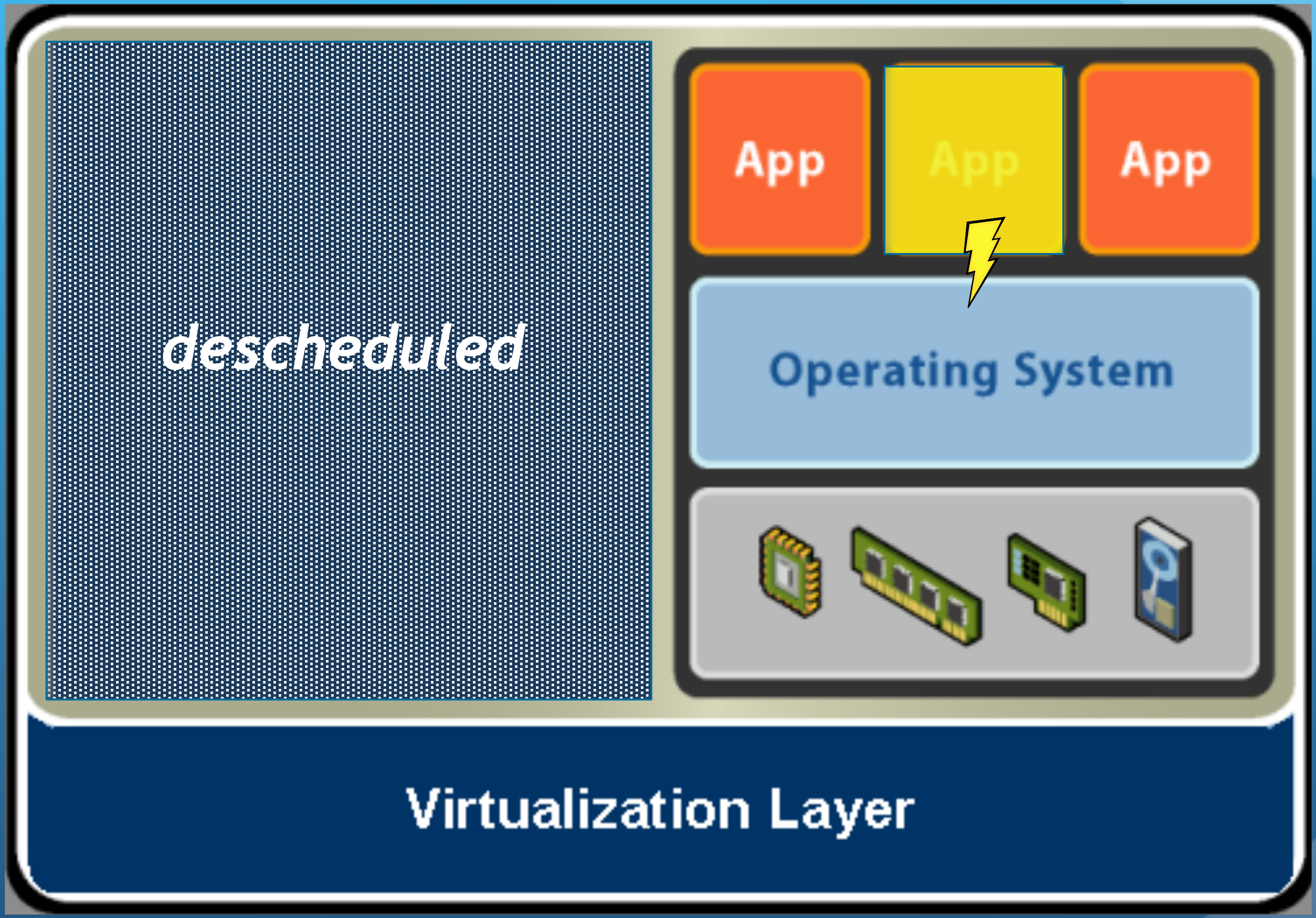
[Animation]

Timer Interrupt Backlog



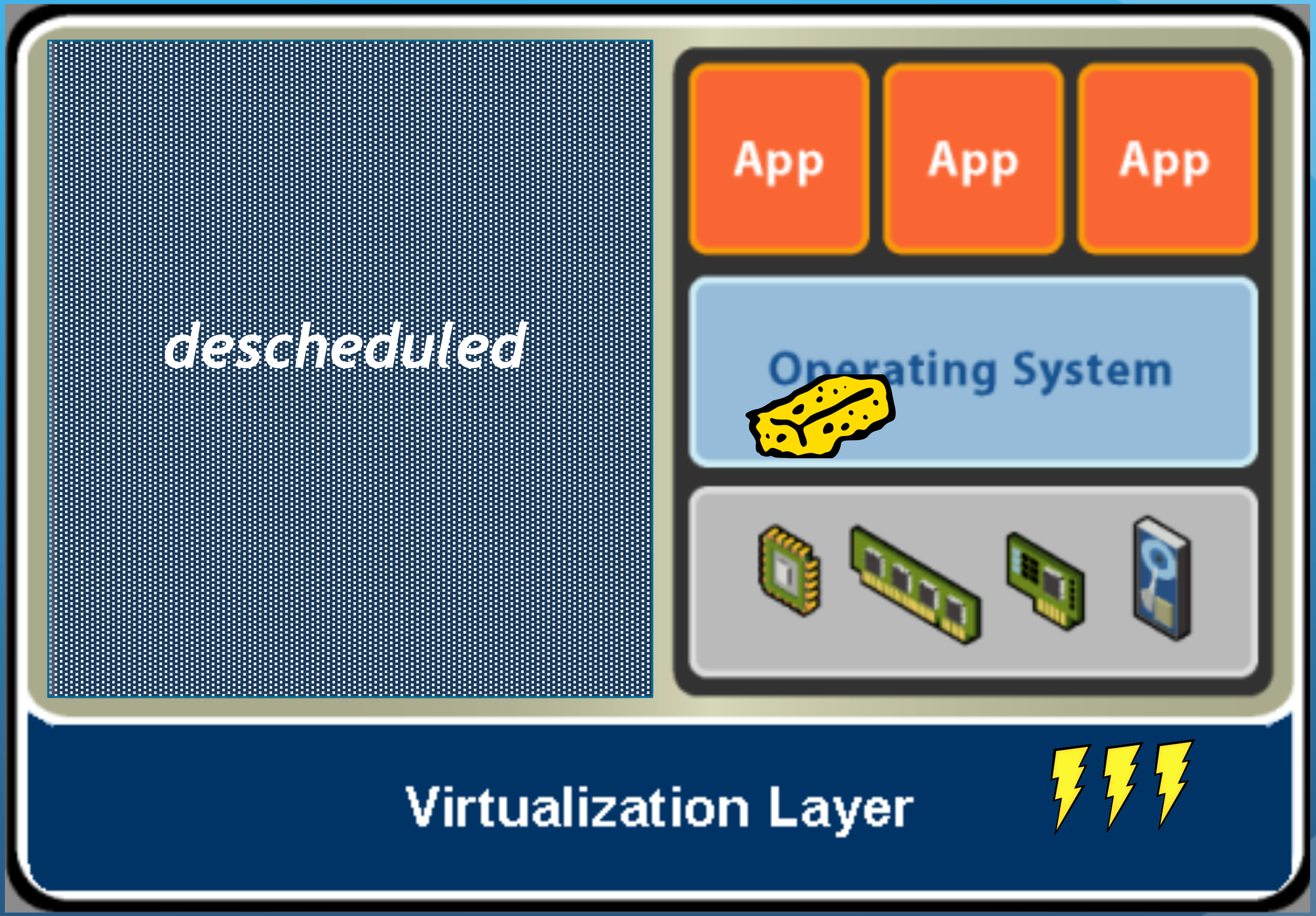
[Animation]

Timer Interrupt Backlog



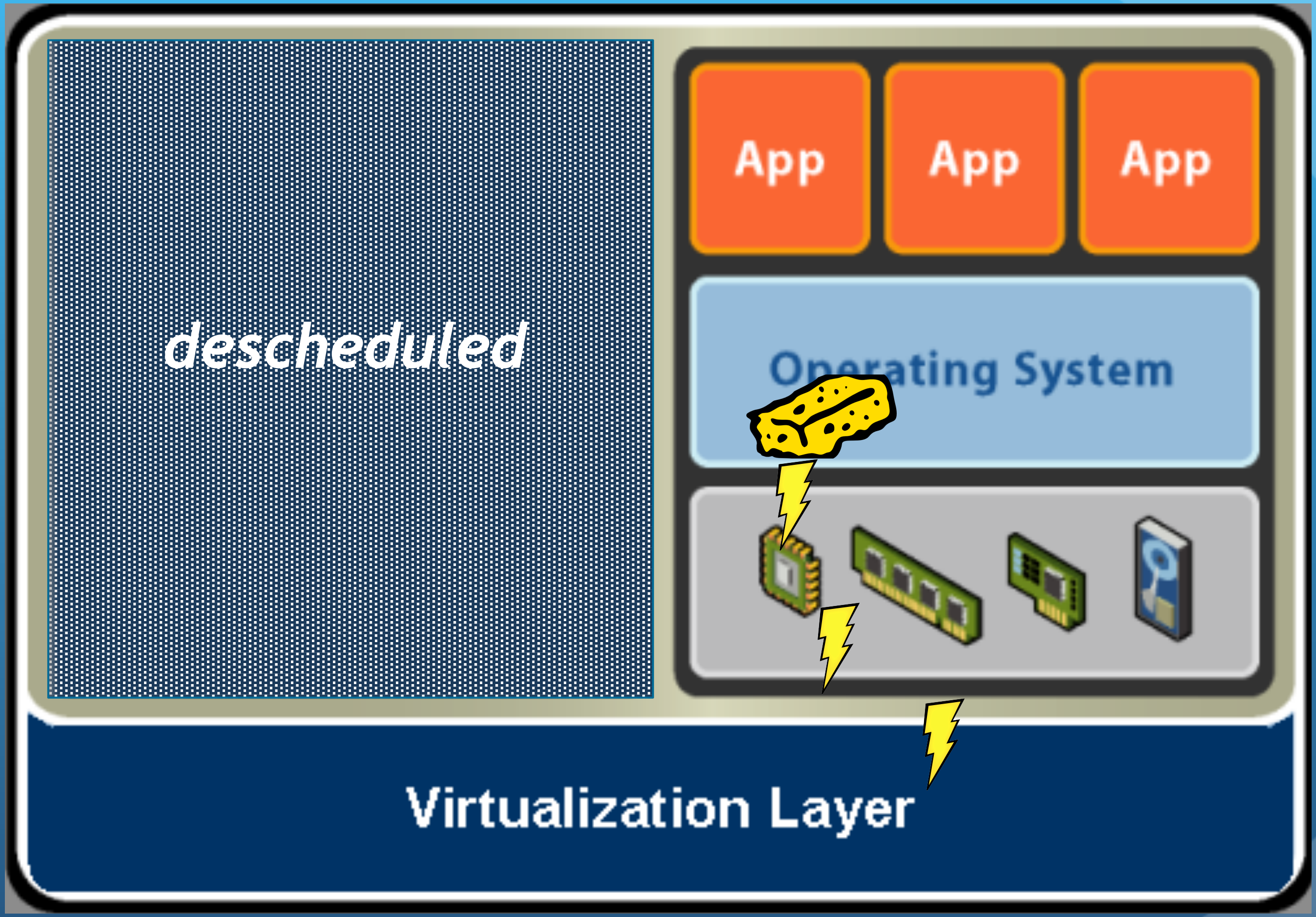
[Animation]

Less Distortion: Timer Sponge



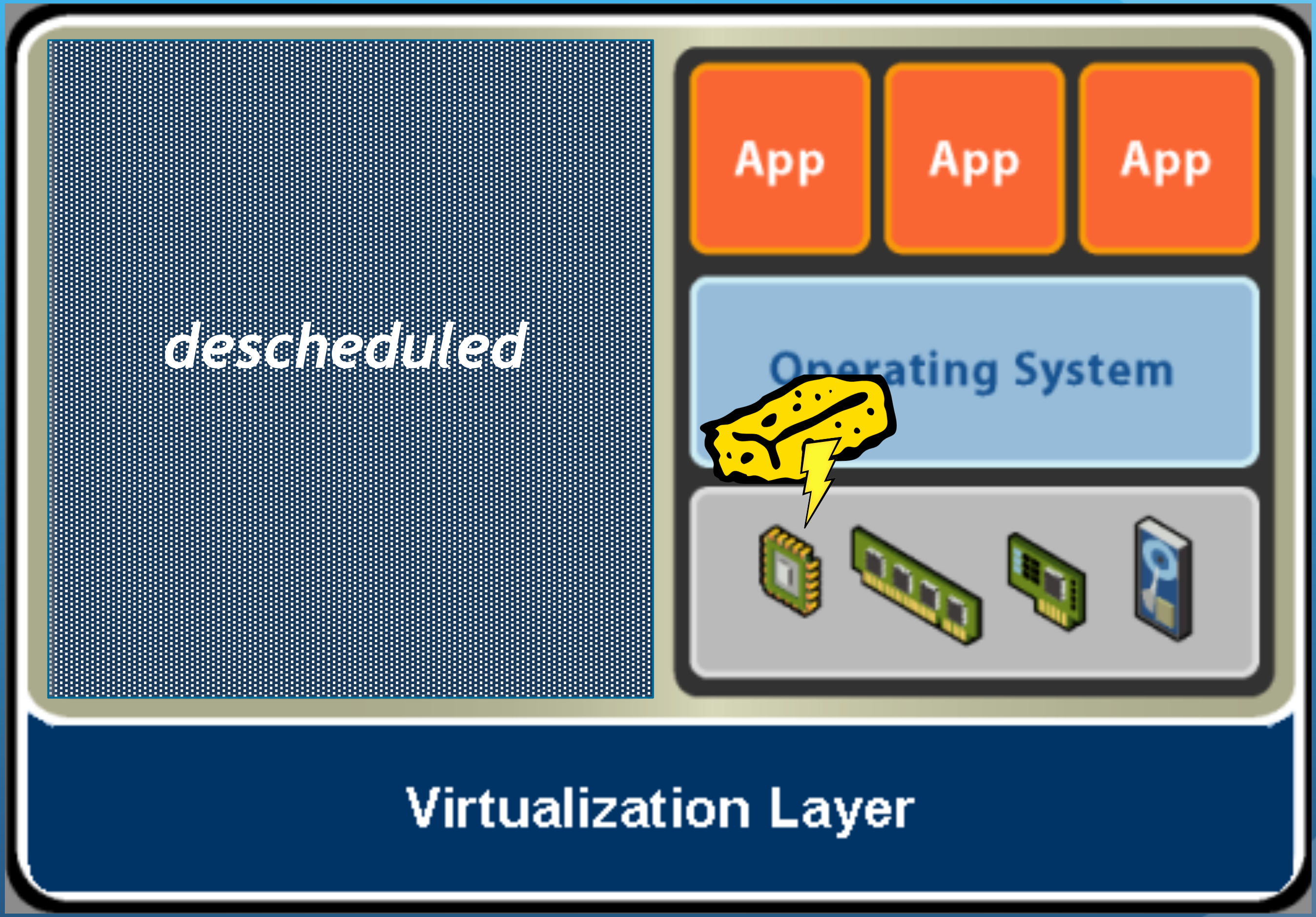
[Animation]

Less Distortion: Timer Sponge



[Animation]

Less Distortion: Timer Sponge



[Animation]

Hazards of Warping Time

- Distorting guest time measurements
- Degrading network throughput
- Exposing guest bugs

Future Research Directions: Measurement

- Descheduled time distortion — still!
- Guest access to hardware counters
- Distributed measurements

Essentially, all models are wrong, but some are useful. — *George Box*

2. Practical Modeling

Cache locality, MRCs, big data

Modeling Goals

- Predict effect of change
  - Resource allocation
  - Reconfiguration
- Inform higher-level policies
  - Determine if satisfiable
  - Both reactive and proactive

Cache Modeling

- Inform cache sizing policy
  - Performance non-linear in allocation
  - Marginal utility
- Mattson stack algorithm (1970)
  - Computes misses for all possible sizes
  - Very powerful, single pass
  - Still expensive

Mattson Algorithm Example

references	…	C	B	A	D
distances	…	4	∞	3	7

- Reuse distance
  - Unique refs since last access
  - Distance from top of LRU-ordered stack
- Hit if distance < cache size, else miss

Mattson Algorithm Example

				✓		
references	…	C	B	A	D	A
distances	…	4	∞	3	7	1

[Animation]

- Reuse distance
  - Unique refs since last access
  - Distance from top of LRU-ordered stack
- Hit if distance < cache size, else miss

Mattson Algorithm Example

				✗	✓	✓	
references	…	C	B	A	D	A	B
distances	…	4	∞	3	7	1	2

[Animation]

- Reuse distance
  - Unique refs since last access
  - Distance from top of LRU-ordered stack
- Hit if distance < cache size, else miss

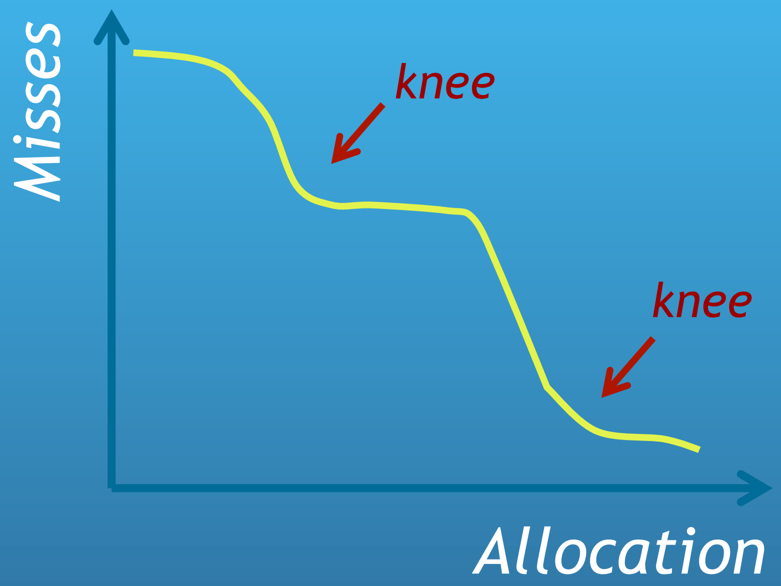
Mattson Algorithm Example

			✗	✗	✓	✓	✓	
references	…	C	B	A	D	A	B	C
distances	…	4	∞	3	7	1	2	3

[Animation]

- Reuse distance
  - Unique refs since last access
  - Distance from top of LRU-ordered stack
- Hit if distance < cache size, else miss

Cache Utility Curves



- How performance varies with size
- MRC
  - miss ratio curve
  - miss rate curve
- Working set "knees"
- Many applications

Mattson Implementations

- Naïve Stack
  - N = total refs, M = unique refs
  - $O(N \cdot M)$ time, $O(M)$ space
- Optimized
  - Balanced tree: compute reuse distance
  - Hash table: maps address to tree node
  - $O(N \log M)$ time, $O(M)$ space
- Parallel algorithms

MRC Approximations

- Hardware Support
  - Qureshi and Patt (MICRO '06)
- Temporal sampling
  - Bursty tracing, detect phase transitions
  - RapidMRC (ASPLOS '09), Zhao *et al.* (ATC '11)
- Spatial sampling
  - VMware memory MRCs (USPTO App '10)
  - CloudPhysics I/O MRCs

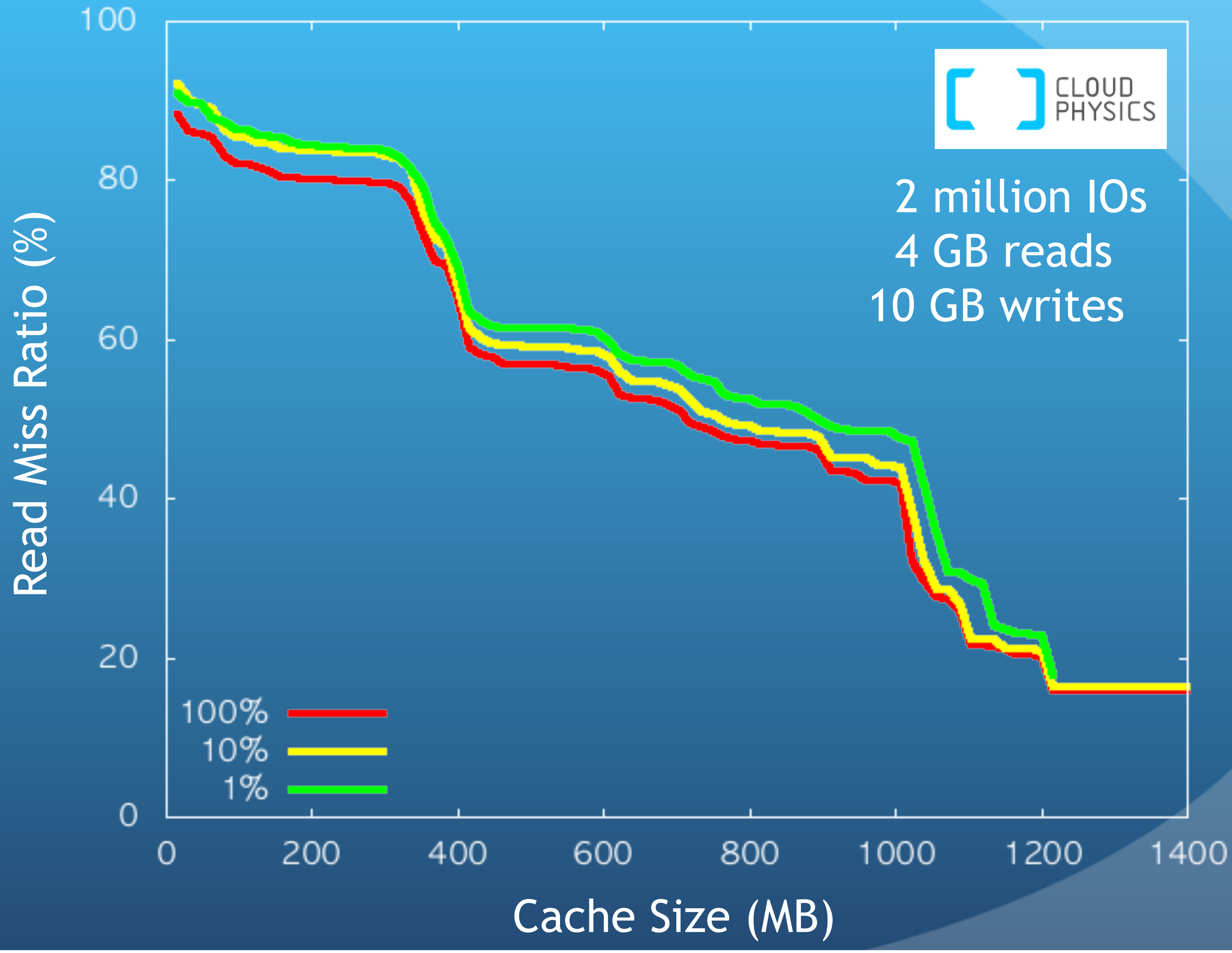
Sampled-Page MRCs

- Spatial sampling
  - Trace only small random subset of pages
  - Each sample represents many pages
  - Run full LRU-based Mattson on subset
- Rate-limit trace rearming for hot pages
- Extremely efficient
  - Excellent accuracy with < 1% overhead
  - Leave on continuously, online MRCs

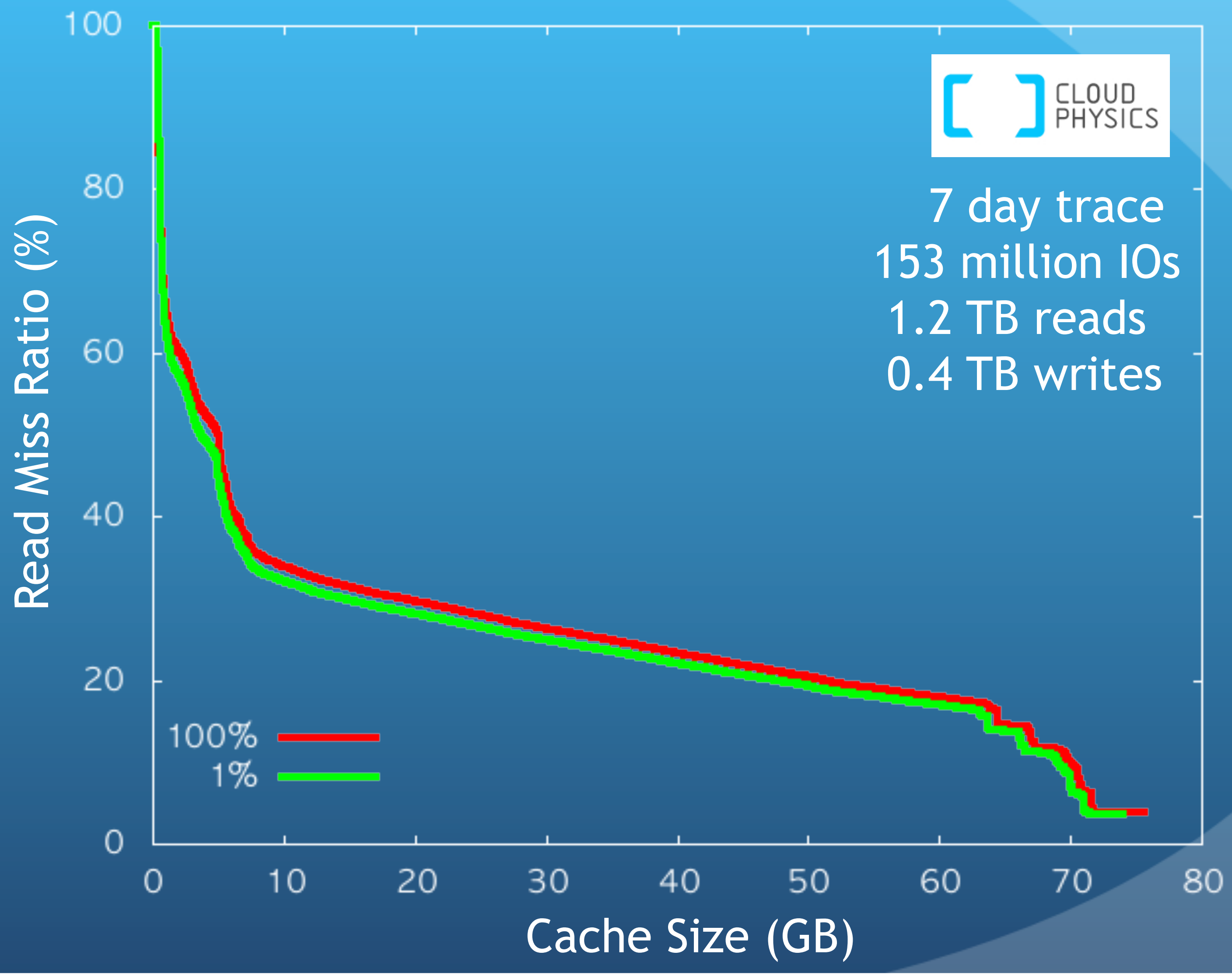
Sampled-IO MRCs

- New spatial sampling technique
  - CloudPhysics caching analytics
  - Detailed paper in preparation
- Huge performance wins
  - Orders of magnitude faster, smaller
  - Surprising accuracy with 1% sample
- Practical online construction

Sampled-IO MRC (Small Trace)



Sampled-IO MRC (Larger Trace)



Modeling Complex Systems

- Many interacting components
  - E.g. cache, bandwidth to backing store
  - Huge state space: cpu × mem × net × io × …
- Approaches
  - Analytical models
  - Simulation
  - Experimentation
  - Observation

Active Experimentation

- Run *many* experiments on real system
  - Load testing tools, *e.g.* HP LoadRunner
  - VMware SDRS load injector (SOCC '11)
- Experiment with cloned VMs
  - Fork using live migration, vary allocations
  - JustRunIt, Zheng *et al.* (ATC '09)
  - Nondeterminism, external dependencies

Passive Observation

- Observe *many* real systems
  - Diverse configurations, devices
  - Diverse workloads, demand patterns
- Reach critical mass of “big data”
  - Model-by-query: lookup similar scenarios
  - Interpolate to handle sparseness

Future Research Directions: Modeling

- MRC temporal dynamics
  - Behavior at different time scales
  - MRC “diffs” and “movies”
- General “microcosm” simulation?
- Multi-resource modeling
- Big data techniques

Rule of Separation: Separate policy from mechanism; separate interfaces from engines.

— Eric S. Raymond

3. Effective Mechanisms

Co-scheduling, ballooning

Co-scheduling vCPUs

- Semantic gap
  - What does 100% busy vCPU mean?
  - Useful work? *Or spinning on lock?*
- Co-scheduling
  - Maintain illusion of dedicated hardware
  - Limit skew between vCPUs within VM
- Alternatives
  - Para-virtualization, *e.g.* Hyper-V
  - Hardware assist, *e.g.* Intel PLE

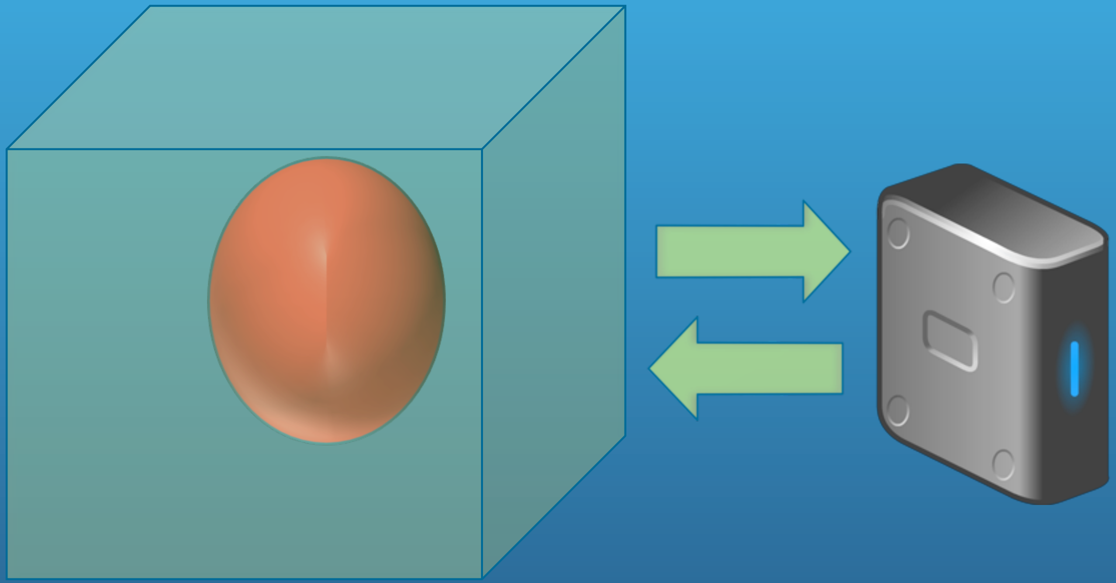
VM Memory Reclamation

- Transparent: demand paging
  - Hard meta-level page replacement decisions
  - Best data to guide decisions internal to guest
  - “Double paging” anomaly
- Alternative: implicit cooperation
  - Coax guest into doing page replacement
  - Avoid meta-level policy decisions

Ballooning

VM Physical Memory
Guest RAM

Virtual disk
Guest swap

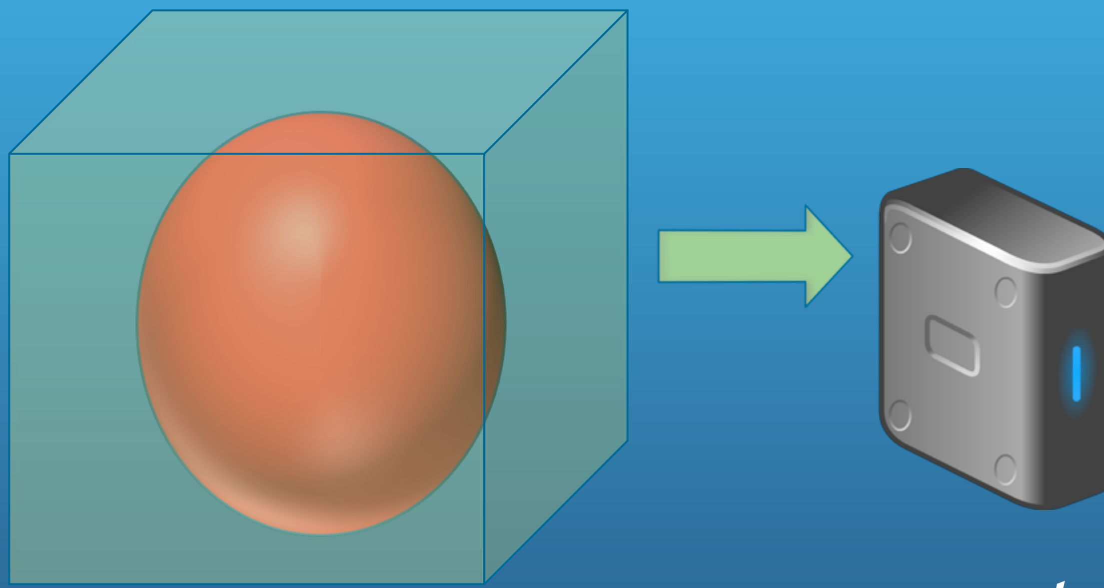


[Animation]

Ballooning

VM Physical Memory
Guest RAM

Virtual disk
Guest swap



may page out

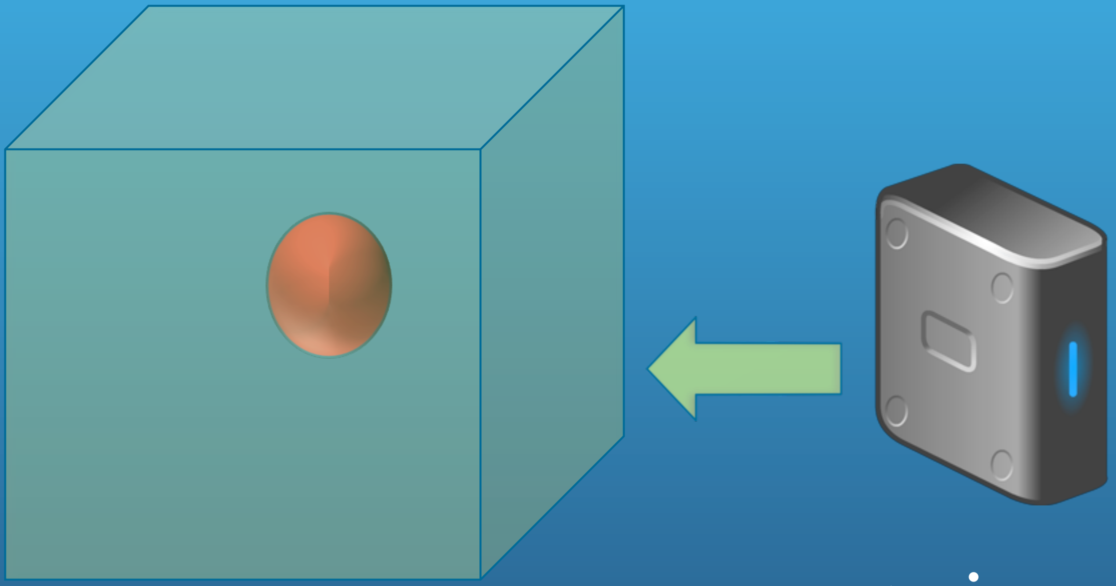
Inflate: more pressure

[Animation]

Ballooning

VM Physical Memory
Guest RAM

Virtual disk
Guest swap



may page in

Deflate: less pressure

[Animation]

Ballooning Retrospective

- Exploits semantic gap
  - Complete transparency not always desirable
  - Coax guest into doing hard work
- Has worked well for a long time
  - Primary ESX memory reclamation mechanism
  - Now used by Hyper-V, Xen, KVM, EM4J, ...
- More recent issue: large pages

Large Pages

- Coarser mapping granularity
  - Single x86 large page covers 512 small pages
  - Reduces TLB misses, makes them cheaper
- Significant win for virtualization
  - x86 nested paging hardware: Intel EPT, AMD RVI
  - Two-dimensional page walk, quadratic cost
  - Large pages reduce number of levels

Ballooning and Large Pages

- ESX hypervisor large-page management
  - Start with large-page mappings
  - Fragment on overcommit, re-coalesce
- Primitive guest OS large-page support
  - Often pinned in memory, so can't balloon!
  - Windows can't swap, Linux swaps some

Future Research Directions: Mechanisms

- Coping with larger page granularity
  - Severe dedup impact, HICAMP (ASPLOS '12)
  - Coarsened visibility
- Extreme design points, PrivateCore vCage
- Meta-mechanisms
  - Cost-benefit, choose most appropriate
  - E.g. dedup, balloon, compress, swap
- End-to-end QoS controls

The limits of your language are the limits of your world. — *Ludwig Wittgenstein*

4. Intuitive Policies

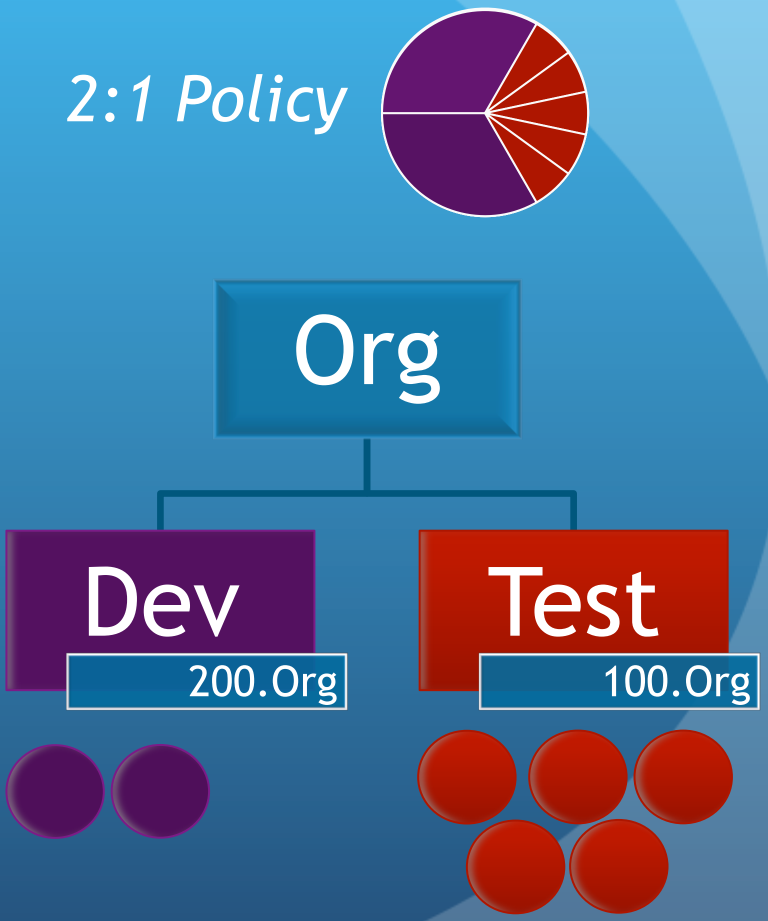
Specifications, microeconomics, automation

Expressing Policies

- Resource Level
  - Provided by modern virtualization systems
  - Physical resource allocation: GHz, GB, Gbps
- Application Level
  - Metrics more meaningful to user
  - Response times, transaction rates, …

Resource-Level Policies

- Basic VM controls
  - Reservations, Limits
  - Shares
- Resource pools
  - Manage sets of VMs
  - Hierarchical
  - Cloud service providers



Practical App-Level Policies

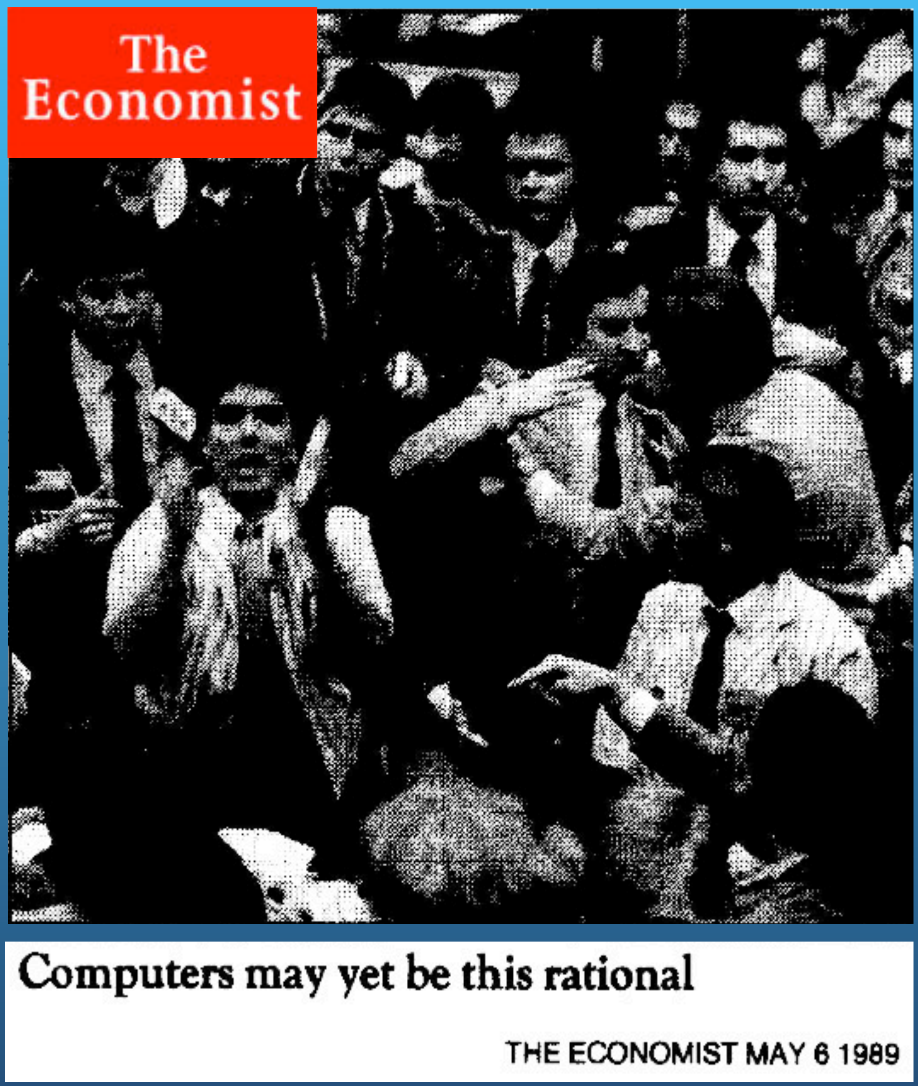
I never had a policy; I have just tried to do my very best each and every day. — *Abraham Lincoln*

- Real world?
  - Formal QoS/SLAs/SLOs surprisingly rare
  - Admins running virtualized datacenters
- Expressing utility functions even harder

Microeconomic Techniques

- Market-based resource allocation
  - Price equilibrates supply and demand
  - Distributed solution to conflicting goals
  - "Invisible hand" improves social welfare
- Much of real world works this way
  - Plenty of interesting analogies
  - Rent, taxes, arbitrage, …

Spawn: Early Computational Economy



- Xerox PARC, late 80s
- Distributed auction
  - Jobs bid for time slices
  - Hosts maximize profit
  - Sealed bid, second price
- Complex dynamics
  - Simple bidding strategy
  - Proportional control
  - Oscillations, chaos

Computational Economies Today

- Why not more common?
  - Better alternatives for simple policies
  - Auction overheads, stability concerns
- Public cloud pricing
  - VM resources rented for real money
  - Multi-tenancy requires sophisticated policies
  - Trends: finer-grain, market-based pricing

Bidding Strategies

- Determining what resources are worth
  - Utility as function of performance
  - Performance as function of allocation
- Getting a good price
  - Mechanical bid adjustment algorithm
  - Game theory
- Need to automate, build into apps
  - Apps aware of own performance tradeoffs
  - Dynamic stability, volatility

A More Direct Alternative?

- "Unhappy" button
  - Primitive, single-bit feedback
  - Squeaky wheel gets the grease
- Empathic Systems Project (Northwestern)
  - Incorporate direct user feedback
  - User-driven scheduling of interactive VMs

Future Research Directions: Policies

- Raising abstraction level
  - Single resource → multiple resources
  - Physical allocation → application goals
  - Many deep challenges
- Intuitive ways to specify
  - Application-level vocabulary?
  - Market-based prices?
  - Empathic systems?

We can only see a short distance ahead, but we can see plenty there that needs to be done.

— Alan Turing

Research Directions

Toward More Autonomic Systems

- Intuitive policies
  - KISS, app-level, empathic, market-based
- Effective mechanisms
  - End-to-end QoS, coarse control, meta
- Practical modeling
  - Multi-resource, big data, MRC dynamics
- Accurate measurement
  - Distortion, hardware access, distributed

Vision for Future: RMaaS

- Resource Management as a Service
- Offload decisions to "RM provider"
  - Remote monitoring and control
  - Leverage "big data" across customers
- Hybrid automation
  - Transparently escalate to human experts
  - Crowdsourcing possibilities

Questions?

carl@waldspurger.org