Xen Hypervisor Scheduler Optimization

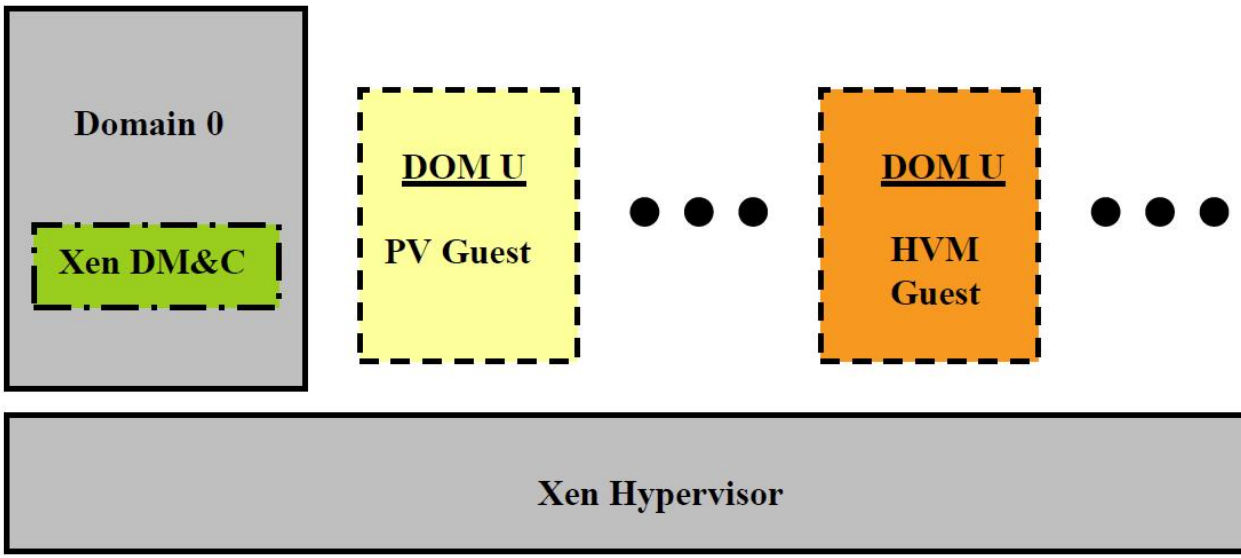
James Devine

April 16, 2009

- Xen is an Open Source Hypervisor
- First release in 2003 "Xen and the Art of Virtualization"
- At first only supported paravirtualization
- Now supports full virtualization
- In 2007 Xen became a part of Citrix

Components

- Hypervisor
- Domain 0
- Domain Management and Control
- Domain U PV Guest
- Domain U HVM Guest



Goals

- Want to make sure that VMs get "fair" share of CPU
- Want to keep the CPU busy
- High CPU utalization
- Low response times

Xen Scheduling Algorithms

- Borrowed Virtual Time
- Atropos- soft real time scheduler
- Round Robin
- Simple Earliest Deadline First (sEDF)

- Dynamic priority real-time scheduling policy
- Thread execution time is monitored in terms of virtual time, dispatching the runnable thread with the earliest effective virtual time
- A latency-sensitive thread is allowed to warp back in virtual time to make it appear earlier, gaining dispatch preference
- Per domain parameters

- Dynamic priority real-time scheduling policy
- Thread execution time is monitored in terms of virtual time, dispatching the runnable thread with the earliest effective virtual time
- A latency-sensitive thread is allowed to warp back in virtual time to make it appear earlier, gaining dispatch preference
- Per domain parameters
  - **mcuadv** - the MCU (Minimum Charging Unit) determines the proportional share of the CPU
  - **warp** - the amount of 'virtual time' the domain is allowed to warp backwards
  - **warpl (warp limit)** - the maximum time a domain can run warped for
  - **warpu (unwarp requirement)**- the minimum time a domain must run unwarped for before it can warp again

- Real time scheduler developed at Cambridge
- Provides guarantees about absolute shares of the CPU
- Facility for sharing slack CPU time on a best-effort basis
- Per domain parameters

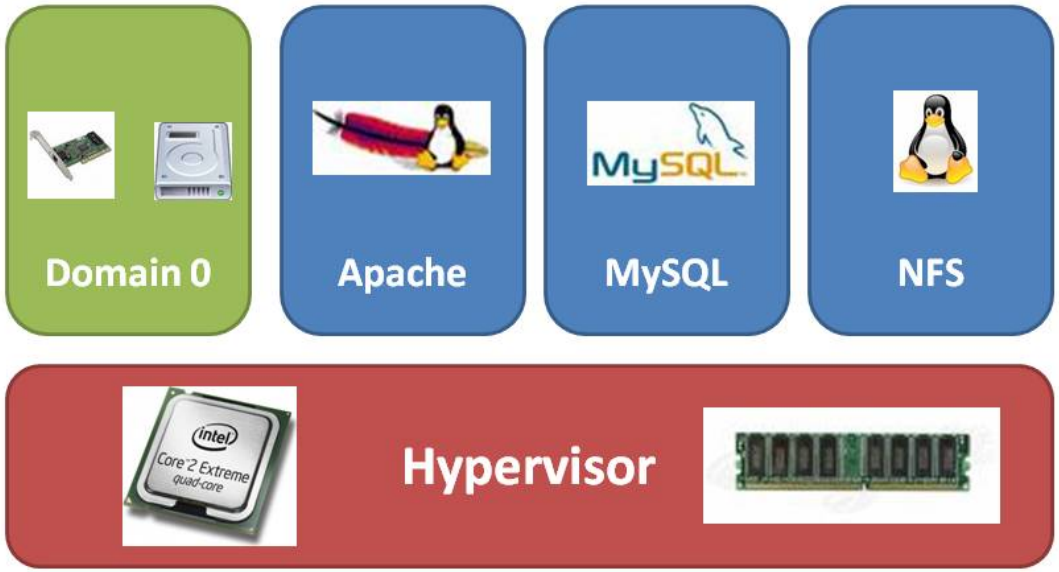
- Real time scheduler developed at Cambridge
- Provides guarantees about absolute shares of the CPU
- Facility for sharing slack CPU time on a best-effort basis
- Per domain parameters
  - **period** - time interval during which a domain is guaranteed to receive its allocation of CPU time
  - **slice** - time per period that a domain is guaranteed to run for
  - **latency** - hint that is used to control how soon after waking up a domain it should be scheduled
  - **xtratime** - boolean flag that specifies whether a domain should be allowed a share of the system slack time
- Every domain should receive s of every p

- Dynamic priority real-time scheduling policy
- Processes placed in a priority queue
- Prcesses closest to thier deadline execute first
- Per domain parameters

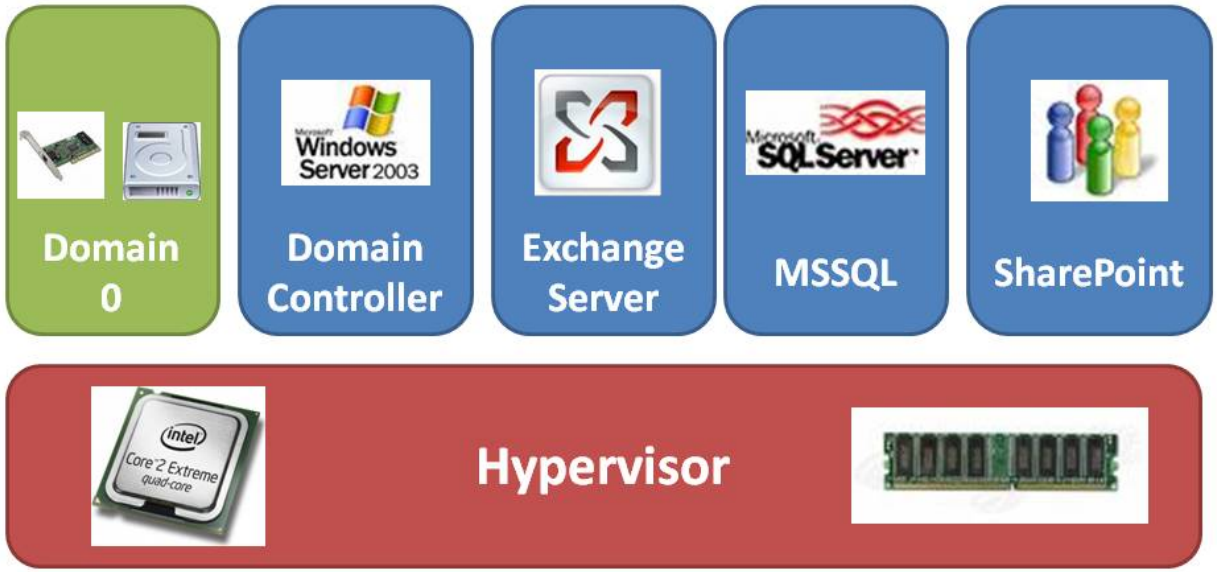
- Dynamic priority real-time scheduling policy
- Processes placed in a priority queue
- Prcesses closest to thier deadline execute first
- Per domain parameters
  - **period** - time interval during which a domain is guaranteed to receive its allocation of CPU time
  - **slice** - time per period that a domain is guaranteed to run for
  - **latency** - (unused by the currently compiled version)
  - **extra** - flag (0/1), which controls whether the domain can run in extra-time
  - **weight** - mutually exclusive with period/slice and specifies another way of setting a domains cpu slice

- Is the BVT schedular the most effiecient (highest thoughput, lowest reponse time, etc.)?
- Are some schedulers better suited to certain virtual workloads?
- Can a more effecient scheduler be implemented?

- Simulate a web application workload
- Create a simple web application (Web, SQL, and NFS Servers)
- Simulate a userload on the web application
- Introduce a VM running CPU burnin (100% CPU utalization)



- Simulate a Windows Domain Workload
- Domain Controller, Exchange Server, MSSQL, and Sharepoint
- Simulate a userload on domain (email, webpage edits)
- Introduce a VM running CPU burnin (100% CPU utalization)



- Run varying workloads with each case study
- Record data on thoughput and response time
- Repeat the procedure with each scheduler

- Two new case study applications
  - Open source case study can be released for future research
  - The methodology for the Domain case study can be relase so that it can be repeated
- Provide some notion of a cross-platform virtual system benchmark
- Adress the question as to which scheduler is "best" for each representative workload