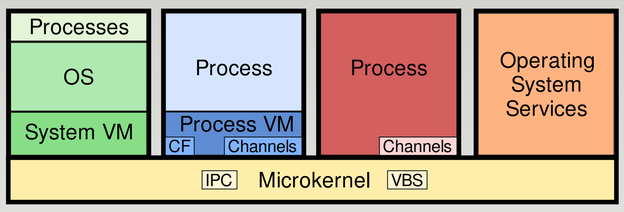
Everyone Virtualizes Everything But Time

Silviu Craciunas, Christoph Kirsch, Hannes Payer, Harald Röck, Ana Sokolova
Department of Computer Sciences
University of Salzburg, Austria
firstname.lastname@cs.uni-salzburg.at

Contribution

Imagine a virtualized execution environment (VEE) that virtualizes not only the host system it runs on, even not only other systems slower than the host system, but also maintains and adjusts the exact speed at which these systems operate, in strong temporal isolation from each other, when they execute code, process I/O, and manage memory.

Tiptoe



On the lowest level, there is a microkernel, which contains the VBS scheduler and an IPC mechanism. On top of the microkernel, processes using the channel subsystem, and operating system services, e.g., device drivers, may run along with operating system instances encapsulated in system VMs, and process VMs, which may take advantage of CF and the channel subsystem. Scheduling parameters for the VBS scheduler are set via system calls. [1]

Integration

The key problem is to design all system components such that there always exists at most a linear relationship between the amount of CPU time required by each component to process a workload and the actual amount of the workload.

Zero vs. non-zero overhead

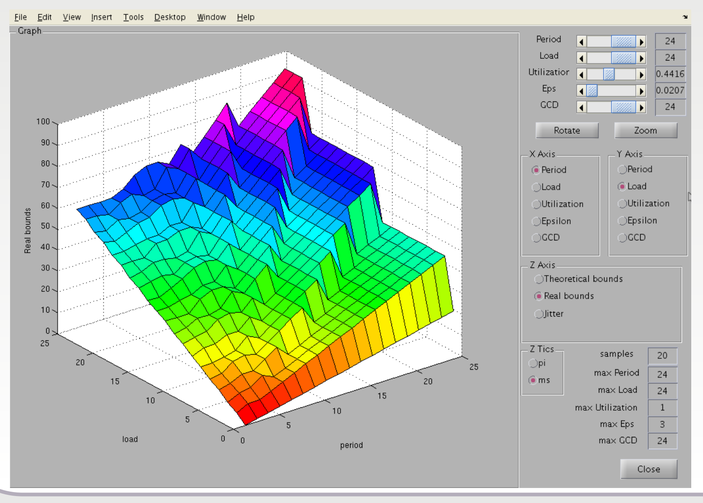
Zero overhead response bounds :

$$b_{i,j} = \pi_{i,j} - 1 + \left\lceil \frac{l_{i,j}}{\lambda_{i,j}} \right\rceil \cdot \pi_{i,j} \qquad (2)$$

Non-zero overhead response bounds:

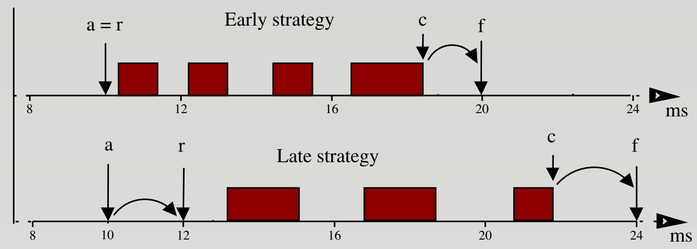
$$b^r_{i,j} = \pi_{i,j} - 1 + \left\lceil \frac{l_{i,j}}{\lambda_{i,j} - \delta_{S_{i,j}}} \right\rceil \cdot \pi_{i,j} \qquad (3)$$

$$\delta_{S_{i,j}} = \left(\left\lceil \frac{\pi_{i,j}}{gcd(\Pi)} \right\rceil + 1 \right) \cdot \sigma$$



VBS Scheduling

Tiptoe uses a real-time scheduler for scheduling all system activities. Tiptoe assigns each scheduling task, i.e., process or VM instance, in the system to a unique VBS, which essentially controls the execution speed of the assigned task and may even change the speed at any time upon request.



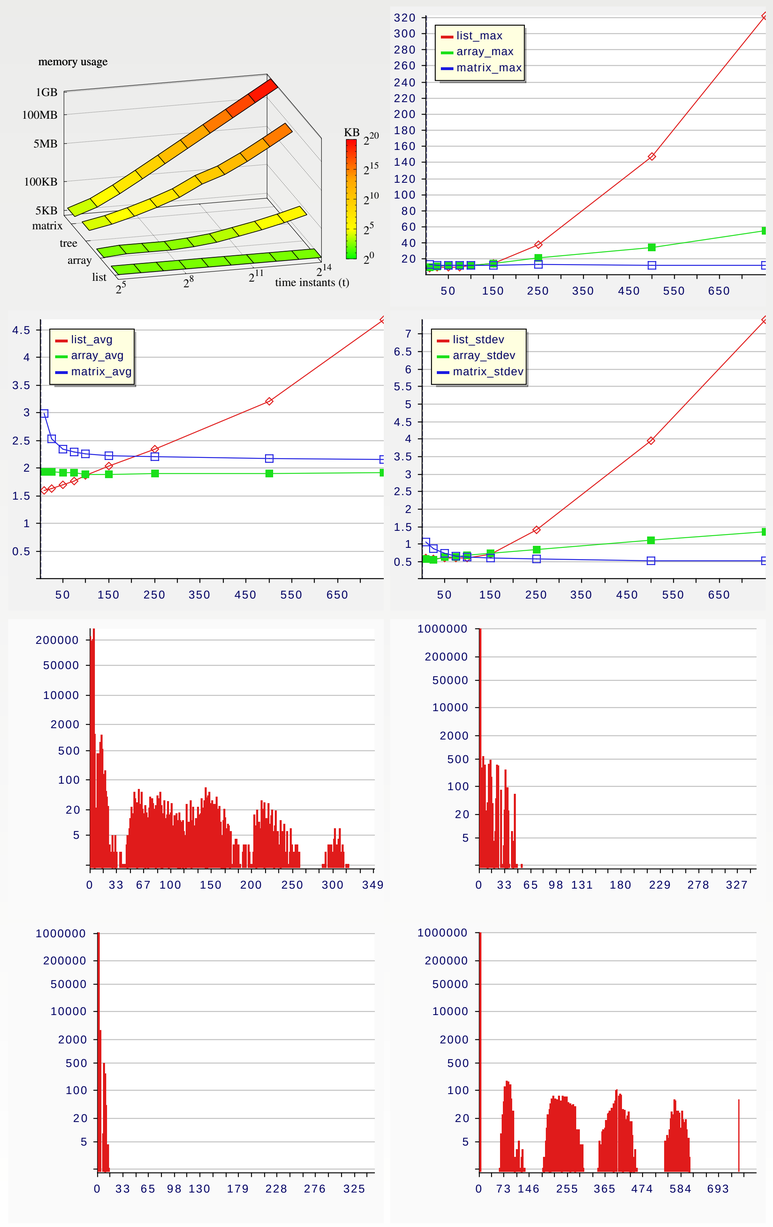
A VBS is configured by a single number u that determines a utilization bound (bandwidth cap). To configure their actual execution speed, each action of a process chooses a pair (λ, π) (virtual periodic resource) such that λ over π is less than or equal to the bandwidth cap u of the used VBS. Switching to different periods allows to trade off scheduling overhead and temporal isolation at runtime.
Let $\{P_i \mid i \in I\}$ be a set of processes each running on a VBS with utilization u_i. If

$$\sum_{i \in I} u_i \leq 1, \qquad (1)$$

then this set of processes is schedulable using the EDF strategy so that each action meets its response bounds.

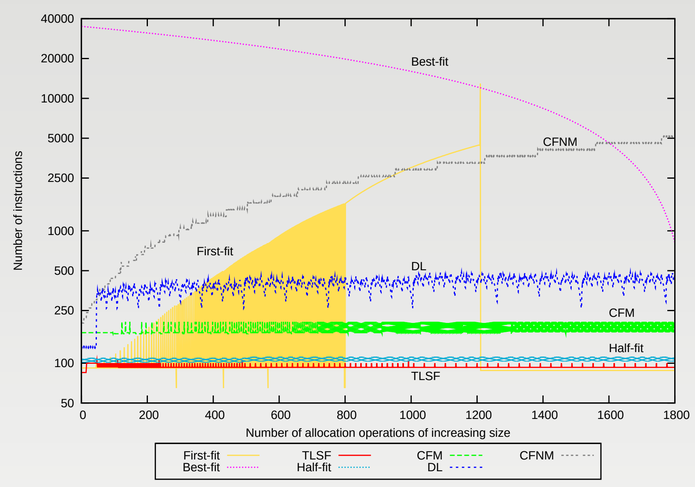
Three different implementations:

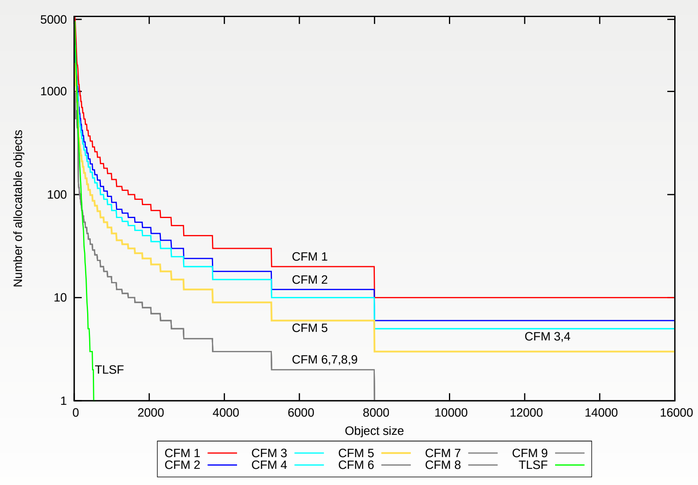
	list	array	matrix/tree
time	$O(n^2)$	$O(\log(t) + n \cdot \log(t))$	$\Theta(t)$
space	$\Theta(n)$	$\Theta(t + n)$	$O(t^2 + n)$



Compact-fit

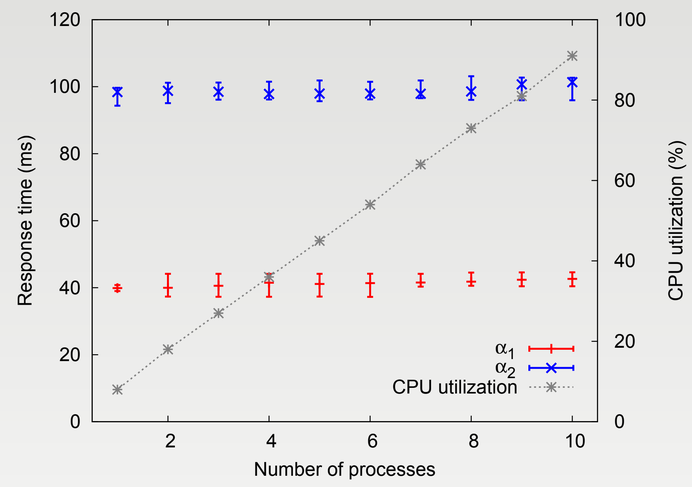
Process VMs that use an object-based memory model may use our Compact-fit explicit memory management system [2] to manage their internal heaps in real time. Compact-fit is a compacting memory management system for allocating, deallocating, and accessing memory in real time. The system provides predictable memory fragmentation and response times that are constant or linear in the size of the request, independently of the global memory state.





Results

Consider a process implementing a simple feedback controller. The first action (α_1) reads sensor values, computes a new control command, and writes the actuators. The second action (α_2) updates the state of the controller and has less stringent timing requirements. Action α_1 is associated with the virtual periodic resource $R_1 = (320\mu s, 3550\mu s)$ while action α_2 uses the resource $R_2 = (500\mu s, 5340\mu s)$.



We show the minimum, maximum, and average response times of action α_1 and α_2, respectively (left y-axis). The response time jitter varies within two periods of the virtual periodic resource independently of the overall system utilization (right y-axis). CPU utilization increases from 9% up to 92% when 9 additional processes run concurrently.

References

[1] S.S. Craciunas and C.M. Kirsch and H. Payer and H. Röck and A. Sokolova - Programmable Temporal Isolation in Real-Time and Embedded Execution Environments In *Proc. Workshop on Isolation and Integration in Embedded Systems (IIES) 2009*

[2] S.S. Craciunas and C.M. Kirsch and H. Payer and A. Sokolova and H. Stadler and R. Staudinger - A Compacting Real-Time Memory Management System In *Proc. USENIX Annual Technical Conference 2008*

Funding

This work is supported by a 2007 IBM Faculty Award, the EU ArtistDesign Network of Excellence on Embedded Systems Design, and the Austrian Science Fund No. P18913-N15.