Real Time-Sharing in Emulab through Preemption and Stateful Swapout

Anton Burtsev Prashanth Radhakrishnan Mike Hibler Jay Lepreau

School of Computing, University of Utah

Emulab is a highly successful space-shared network testbed that includes more than 350 PC nodes. Some 1500 users run "experiments" of size 1–300 nodes, on a first-come first-served basis. Demand far exceeds capacity, so Emulab provides a weak form of time-sharing. An Emulab experiment definition—its network topology and initial state—is analogous to a Unix program, a running experiment analogous to a process, and the persistent store (on an external file server) analogous to an on-disk filesystem. Upon swapout, an experiment's node local physical resources—memory and disk—are freed and lost, but the experiment definition and state persists.

When later "swapped in," the network topology and initial disk state are reinitialized, but experimenters must manually recover the approximate state of their live experiment. If true time-sharing were provided, the Emulab scheduler and operators could have free rein to optimize resource use according to any policy, at most causing only delay to experimenters, not lost work. The scheduler could preempt low-priority or idle experiments, save their full state, and replace with other experiments. The goal is that swapout/swapin must be transparent to the experiment, and except for scheduling delays, transparent to users.

In this work we are extending Emulab with the ability to swapout experiments without losing their node local state. Moreover, by virtualizing the network and experiment nodes, we ensure that the entire period of inactivity is transparent to the experiment. Effectively, we extend Emulab with a coarse grain scheduling mechanism similar in nature to the operating system process scheduling.

To achieve this goal we place experiment nodes under control of the Xen virtual machine monitor. This allows us to take a consistent snapshot of the entire network upon a swapout and resume upon a subsequent swapin.

There are several major issues and challenges:

Consistent checkpointing: To ensure that the experiment is checkpointed in a consistent state, we employ the well-known technique of tagging outgoing network packets with the checkpoint "epoch-id." To eliminate packet loss, we log packets that are in flight during a swapout and replay them upon a swapin.

Time virtualization: To conceal the fact that the experiment was swapped-out, we resume it in virtual time upon a swapin. By virtualizing time, synchronizing nodes and replaying network packets, we are able to eliminate retransmissions in TCP sessions between the nodes after swapin. Communication with the world outside the experiment, in particular on the "control network" through which the user and Emulab manage the experiment, may pose challenges with timestamps embedded in packets.

Network virtualization: Currently, Emulab relies on virtual network provided by Cisco switches to build experiment topologies. Unfortunately, setting up a hundred-node topology requires several minutes. Virtualization allows us to switch to software 802.11q VLANs and setup only one Cisco VLAN to isolate traffic between experiments. Software VLANs can be setup in several seconds.

Pipelined swapout: Node local state is saved to the Emulab repository upon a swapout and restored to potentially different experiment nodes at the time of a subsequent swapin. The time spent on this possibly time-consuming operation is pure experiment scheduling overhead. In order to improve node utilization, we plan to pipeline the swapout of an experiment with the swapin of the next experiment.

Redundancy elimination: The key insights behind reducing swapout time are the observations that most of the node local disk state does not get mutated and that a lot of identical changes are made across nodes. We exploit the local redundancy with copy-on-write (CoW) storage. For remote redundancy elimination, we employ content addressable storage (CAS). Upon a swapout, the content hashes of the memory state and CoW disk delta of all nodes in the experiment are sent to the server. The server requests for only those blocks that are not already in the CAS store (similar to LBFS and rsync).

Decoupling metadata and data: We believe that decoupling metadata and data is a key design point for scalability. Thus, the Emulab storage server decouples the CAS metadata—mappings of disk image deltas to content hashes—from the actual CAS store. During a swapout (or swapin) operation, the metadata server issues tokens to the experiment nodes for saving (or retrieving) data blocks in the CAS store.

Status: We have completed several of the components, have pieces of the others done, and are starting to knit things together. We have extended Xen with consistent checkpointing and time virtualization. We built a full-featured copy-on-write branching storage system by extending the Linux LVM snapshot mechanism. We have implemented a delta mechanism for disk "image" state storage and provisioning, and gathered statistics on disk state differences. We have implemented the virtual networking support. We are currently designing the metadata and full CAS services.

Students: Burtsev, Radhakrishnan. Corresponding author: aburtsev@cs.utah.edu.