Distributed Experiment Management for Large-scale Testbeds

Christian Hübsch
Institute of Telematics
Universität Karlsruhe (TH)
huebsch@tm.uka.de

Christoph P. Mayer
Institute of Telematics
Universität Karlsruhe (TH)
mayer@tm.uka.de

Large-scale testbeds like G-Lab or PlanetLab provide realistic environments for real-world evaluation of distributed applications. They offer means for realistic testing and validation that are hard to achieve using simulations. In contrast to simulated environments, real-world testbeds require a much more complex experiment management. Testbeds do provide management for the control of the testbed itself, nevertheless we see little support for the management of experiments: E. g. due to insufficient load-balancing of experiments many nodes suffer high load while others idle. Furthermore, use of centralized control machines does not scale as it results in high load through deployment, or status and result collection.

We identify four phases for what we call *experiment management*: (1) experiment preparation (e. g. selecting an appropriate subset of testbed machines), (2) experiment package deployment, (3) experiment execution (e. g. status feedback and handling of load changes), and (4) experiment data collection. Current solutions have been built with the idea of automating the four-step process of experiment management in a centralized way (e. g. [1]). In this paper we take a different approach towards the challenges in communication for experiment management using decentralized peer-to-peer mechanisms. Our approach has several advantages over a centralized solution: (i) implicit preselection of machines appropriate for the experiment, (ii) efficient distribution of experiment packages, (iii) load-balancing between competing experiments, and (iv) possibility of distributed preprocessing of experiment results during data collection. We employ a tree-like overlay structure $\mathbb{O}$ that includes all testbed machines $\mathbb{T}$. We define a simple fitness metric $m_{kt}(w_1 \cdot p_1, \ldots, w_n \cdot p_n) = \sum_{i=1}^{n} w_i \cdot p_i = x$ for experiment k and machine t with machine-dependent parameters p_i and experiment specific weights w_i. Parameters p_i can be local values like CPU load or free RAM, as well as inter-machine relations like network delay. The metric m is experiment-specific and defined by the user running the experiment. We use m to iteratively build up the overlay structure $\mathbb{O}$, resulting in a tree-like overlay with machines ordered with respect to their fitness m. Machines that have a high fitness for the specific experiment reside in the upper layers of $\mathbb{O}$. The machine in the root of $\mathbb{O}$ is further used as gateway towards the experiment machines $\mathbb{T}$.

We see the selection of an appropriate subset of testbed machines as the first step (1) of experiment management. Ideally, the number of available testbed machines $|\mathbb{T}|$ is larger than the set of machines $\mathbb{S}$ that are needed for the experiment. As $\mathbb{O}$ is ordered with respect to the specific metric m, the best-suited machines for experiment k reside in the upper part of the tree. Sending out of the experiment package in step (2) is performed as follows: The gateway machine is delivered with the package and the required number of machines $s = |\mathbb{S}|$. Distribution of the experiment package is now performed in a scalable manner employing Application-Layer Multicast [2] on the tree structure. Each node decreases s and forwards $\frac{s-1}{2}$ together with the package towards its children as long $s > 0$.

During the experiment (3), the metric-based tree is further maintained, ordering machines with respect to their 'fitness' higher or lower in the structure. As machines fall too low they are considered unusable for the experiment and excluded. They are replaced by new machines with a sufficient tree height in order to preserve the number of experiment participants. This implies that machines in the tree monitor their children and can deploy the experiment package to new children due to reordering of the tree. Also, foreign experiments imposing load on machines are reflected in the specific metric evaluations, leading to such nodes being located lower in the resulting tree. The metric-based tree structuring provides implicit load-balancing between multiple experiments at experiment preparation and during execution.

Status reports sent by the machines (3) to the central control system result in high load and are considered un-scalable. In our approach, status reports during the experiment are sent up the tree using concast. This way aggregation through processing of status information can be performed by machines in the tree and therewith overall status information is reduced. How strong status can be aggregated is highly experiment specific. The same mechanism is used for collecting experimental results from the machines (4) when the experiment has ended. Here, aggregation through preprocessing can be performed, too, again highly depending on the specific experiment. We see high reduction of the amount of traffic through precomputation—e. g. of average values—inside the tree in a distributed fashion.

The presented decentralized approach for experiment management in large-scale testbeds has several advantages over the conventional way. Besides implicit load-balancing it enables efficient distribution of experiment packages, maintenance of the machine set during simulation, and collection of simulation results using aggregation and preprocessing. Due to space limitations, we will show the feasibility of our approach in the presentation by simulative evaluation.

References

[1] "Stork," http://www.cs.arizona.edu/stork.
[2] M. Hosseini, D. Ahmed, S. Shirmohammadi, and N. Georganas, "A Survey of Application-Layer Multicast Protocols," *Communications Surveys & Tutorials, IEEE*, vol. 9, no. 3, pp. 58–74, Jul. 2007.