# Distributing Akaroa2 on PlanetLab

Farzana Yasmeen[*1] Greg Ewing[*2] Krzysztof Pawlikowski[*2] Shigeki Yamada[*3]

[*1] Department of Informatics, The Graduate University for Advanced Studies, Japan
[*2] Department of Computer Science and Software Engineering, University of Canterbury, New Zealand
[*3] National Institute of Informatics, Japan

## Abstract

PlanetLab is an overlay-based test-bed that serves as a research and deployment platform for academic institutions and industrial research labs around the world. Created in 2002 at Princeton University, PlanetLab has expanded to nearly 1000 machines worldwide. The University of Canterbury (UC) in Christchurch, New Zealand installed two such nodes at the end of 2002. As an initial phase to a broader deployment goal, we successfully distributed Akaroa2 on the machines at the Canterbury, NZ site of PlanetLab in October 2008. Akaroa2 is the latest version of a fully automated simulation controller, originally designed for running distributed stochastic simulations in a local area network environment. It was designed within the AKAROA project at UC by the Simulation Research Group. In this paper we describe our experience with PlanetLab, the methods we have practiced for the distribution of Akaroa2 over PlanetLab and associated issues.

## 1. Introduction

PlanetLab is a global research network infrastructure that supports the development of and experimentation with new concepts of distributed processing and services. It consists of interconnected groups of computers (nodes) around the globe that act as a worldwide distributed computer laboratory. Since the beginning of 2003, thousands of researchers located in over 40 countries have used PlanetLab to develop new technologies for distributed storage, network mapping, peer-to-peer systems, distributed hash tables, and query processing. PlanetLab currently consists of 936 nodes at 456 sites [1].

The Network Research Group from the Department of Computer Science and Software Engineering at UC was invited to join PlanetLab in late 2002, using donated hardware and PlanetLab node was made operational. But it was soon discovered that high charges for Internet connections made use of the node impractical. Recently, with support from a KAREN[1] research grant[2], two nodes at the UC site were re-activated in 2006 and again upgraded in mid-2008. This revived access provided motivation for the Simulation Research Group at UC to resume a PlanetLab-based project[3] on deployment of Akaroa2, to allow world-wide distributed quantitative discrete-event simulation.

[1] KAREN ( Kiwi Advanced Research and Education Network) is a New Zealand ultra-high speed network used for research and education; see http://www.karen.net.nz
[2] 2008-2009 KAREN Capability Build Project Fund: REANNZ CBF 626
[3] This project was partially funded by the SOKENDAI Short-stay Program Studying Abroad FY 2008 grant.

## 2. Akaroa2

Research in distributed and parallel simulation has been almost entirely focused on partitioning the model of the system being studied and simulating the parts on different processors. The main challenge of this technique concerns the logical synchronization of the interdependent simulated sub-processes as they autonomously evolve in time. An additional approach for speeding up stochastic simulation is known as Multiple Replications In Parallel, or MRIP [2].

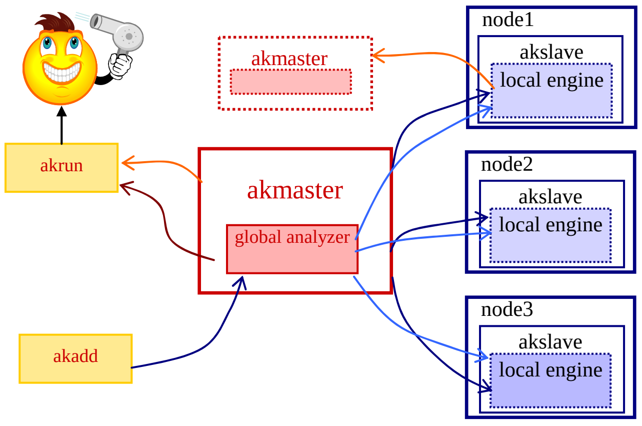

Figure1. Functional blocks of Akaroa2

In MRIP, the computers of the network run independent replications of the whole simulation process, generating statistically equivalent streams of simulation output data. These data streams are fed to a global data analyzer responsible for analysis of the final results and for stopping the simulation when the results reach a satisfactory statistical accuracy [3].

Akaroa2 is a fully automated simulation controller, designed for running distributed stochastic simulations in MRIP. Figure 1 illustrates the processes involved during running a simulation controlled by Akaroa2. In the simplest case, whenever a new observation/output data item is collected during simulation, it is passed to Akaroa2 by making the call:

```
AkObservation(value)
```

The akrun program is used to start a simulation in MRIP mode, for example by stating is:

```
% akrun -n 3 mm1 0.5  0.01
```

This command specifies that a simulation program called *mm1 0.5* (simulation of a M/M/1 queuing system with load =0.5) is to be run using 3 simulation engines and will be stopped when a relative statistical error of 5% is reached.

## 3. Akaroa2 on PlanetLab: The Methods

The main motivation for this project was realized by the intent to popularize Akaroa2 in R&E environments along with utilizing the power and resources provided by PlanetLab. During our pre-deployment study of practices in PlanetLab, we found two evident ways to add nodes to a slice, in order to distribute our application on PlanetLab machines.

Before we describe the methods, here is some basic PlanetLab terminology:

**Site** means a physical location where PlanetLab nodes are located. The name of our site is: *University of Canterbury, New Zealand*. The abbreviate name is*: Canterbury*.

**Node** means a dedicated server that runs components of PlanetLab services. We currently have two nodes: *planetlab3.cosc.canterbury.ac.nz, palnetlab4.cosc.canterbury.ac.nz.*

**Slice** means a set of allocated resources distributed across PlanetLab assigned to its user. To most users, a slice means UNIX shell access to a number of PlanetLab nodes. A slice named *canterbury_slice1* was assigned for the purpose of the distribution.

The two deployment methods we practiced are:

## 3.1 PlanetLab Remote Access to Nodes

We found using the basic Unix shell commands being sufficient when distributing Akaroa2 on a handful of nodes. We added the nodes at the *Canterbury* site to our slice via the 'Manage Nodes' tab from our *slice1* GUI. We then created a SSH keypair to allow secure remote login to the added nodes. Then, on each node, we installed necessary files using the standard yum command. This allowed us to copy *akmaster, akauth, aksalve* files and create other files onto our target nodes:

```
$scp -r canterbury_slice1@planetlab3:/home/...
canterbury_slice1/.akmaster . .akmaster
```

By running akslave as a background process we were able to remotely start a simulation from our local machines onto the two nodes at our site:

```
$ akrun -n 2 mm1 0.1
Simulation ID = 1
Simulation engine started: host = planetlab3.cosc.canterbury.ac.nz, pid =
15281
Simulation engine started: host = planetlab4.cosc.canterbury.ac.nz, pid =
13392
```

This simulation, controlled by Akaroa2, and distributed over PlanetLab nodes, was stopped as required, when the estimated mean waiting time in the M/M/1 buffer was obtained with statistical error smaller than declared; see Table 1.

Table1: Simulation results from mm1 0.1 running on distributed PlanetLab nodes planetlab3 and planetlab4, Canterbury site

| Param | Estimate | Delta | Conf | Total Obs | Trans Obs |
|---|---|---|---|---|---|
| 1 | 0.109983 | 0.00464865 | 0.95 | 3840 | 256 |

## 3.2 PlanetLab User Tools and Services

Scientists engaged in PlanetLab-based research have developed a large number of free applications and services. In our second method, we used a service (along with a few scripts) called *CoMon* to populate our slice with active nodes. We then used the *CoDeploy* service to send all necessary Akaroa2 files to the nodes added on our slice. This is a particularly efficient approach if it is necessary to deploy an application onto a large number of nodes. However, since the scripts are mass deployed, is very difficult to monitor which of the nodes is responding if a failure occurs. Fortunately, a failure of a processor used as an Akaroa2 simulation engine is not fatal because of the fail-soft nature of MRIP.

## 4. Discussion

From our observations we recommend, especially for large-scale distributions, to try a small deployment on local nodes and then on a few external nodes via the remote access method before mass-deploying/replicating them on external sites using tools or services. This can lead to early detection of problems that may not be easily identified when running on several remote sites.

## 5. Future Work

We plan to do a continual performance analysis of Akaroa2 when it is employed as a controller of worldwide distributed stochastic simulations, to explore the limits of MRIP in such scenario. These limits could be specifically associated with Amdahl's law of speedup in MRIP [4], or/and with excessive propagation delays effecting exchange of data between simulation engines and akmaster. Our ultimate goal is to equip users of PlanetLab with a tool which would allow them to produce credible results from stochastic simulations of new networking solutions investigated over that powerful experimental global networking infrastructure; i.e. to overcome problems outlined in [5].

## References

[1] http://www.planet-lab.org/ last accessed on January 5, 2009.
[2] K. Pawlikowski, V. Yau and D. McNickle. "Distributed Stochastic Discrete-Event Simulation in Parallel Times Streams". Proc. Winter Simulation Conference, WSC'94, IEEE Press, 1994,pp. 723-730
[3] G. Ewing, K. Pawlikowski and D. McNickle. "Akaroa2: Exploiting Network Computing by Distributed Stochastic Simulation". Proc. European Simulation Multiconference ESM'99, (Warsaw, Poland, June). ESCS, pp. 160-164
[4] K. Pawlikowski and D. McNickle. "Speeding Up Stochatic Discrete-Event Simulation". Proc. European Simulation Symposium, ESS'01, Marseille, France, Oct. 18-20, 2001
[5] K. Pawlikowski, H.-D. J. Jeong and J.-S. R. Lee. "On Credibility of Simulation Studies of Telecommunication Networks". IEEE Communications Magazine, Jan. 2002, pp. 132-139