Timothy Roscoe
Intel Research at Berkeley
Friday, May 13th, 2005

Overview

- Intro: history, goals, concepts
- Basics: accounts, slices, nodes
- Real work: developing & deploying

< break >

- Tools and services
- Advanced network programming
- Methodology issues
- Where to go for more information

What this won't cover…

- Emphasis is on using the PlanetLab PLC web interface
- This tutorial doesn't talk about:
  - How insitutions can join PlanetLab
  - Most of a PI's duties
  - PlanetLab's detailed architecture
  - PlanetLab's programmatic API
  - Future directions

Part I: History, goals, and concepts

PlanetLab is…

- Large collection of machines spread around the world for distributed systems research
- Focus/catalyst for systems and networking community
- Intel project ⇒ consortium of companies and universities

The value proposition

- Institutions join, provide nodes
  - IA32 architecture servers
  - Hosted outside the firewall
  - Provide power, cooling, & bandwidth
- In exchange, researchers get to use a small "slice" of many machines worldwide.

Origins: wide-area distributed systems research c.2002

- Researchers had no way to try out real systems
  - Architectures, simulations, emulation on large clusters, calling 17 friends before the next deadline…
- but *not* the surprises and frustrations of experience at scale to drive innovation
- How can research systems be validated?

Origins: large-scale networking research c.2002

- Strong feeling the Internet had ossified
  - Intellectually, infrastructure, etc.
  - NRC "looking over fence at networks"
- New ideas abandoned as undeployable
  - Overlays as a way out of the impasse
  - Next internet emerges as overlay (again)
- How can researchers deploy overlays?

Early timeline

- David Culler and Larry Peterson discuss initial idea early 2002
- "Underground" meeting March 2002
- Position paper (Anderson, Culler, Peterson, Roscoe) June 2002.
- Intel seeds project, core team, 100 nodes
- First node up July 2002
- By SOSP (deadline March 2003) ~25% of accepted papers refer to PlanetLab
- Large presence at SIGCOMM
- 11 out of 27 papers in NSDI 2004

PlanetLab is…

- A testbed for experiments
  - Experiment at scale
  - Real-world conditions
  - Potential for real workloads & users
- A deployment platform
  - Continuously running services
  - Design, deploy, measure cycle
  - Long-term studies
- A microcosm of the next Internet
  - Evolve network to support overlays and slices
  - Make the network architecture more *computational*
- A shared artifact!

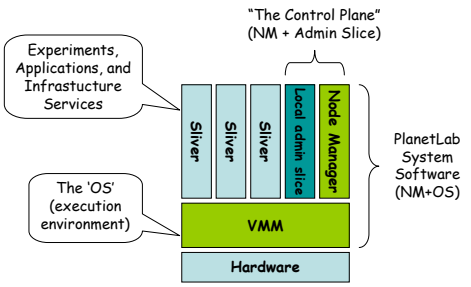
What is PlanetLab good for?

- *Planetary-Scale* applications:
  - **Low latency** to widely spread users
  - **Span boundaries:** jurisdictional and administrative
  - **Simultaneous viewpoints:** on the network or sensors
  - **Hardware deployment** is undesirable
- Deploy, Evaluate, Evolve the architecture

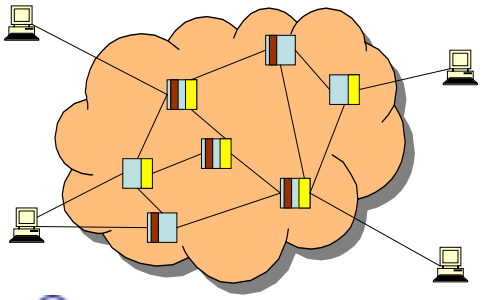
Distributed Virtualization

- *Slices*
  - Basic unit of isolation
  - Distributed set of virtual machines (slivers)
  - Services & applications run "in" slices
- *Nodes*
  - Physical machines, grouped into *Sites.*
  - One node hosts many slivers
- Infrastructure Services
  - Provide functionality to developers or other services rather than users

Node architecture



Slices



PlanetLab these days

About 550 nodes, 260 sites, 30 countries, 5 continents
Universities, Labs, POPs, CoLos, DSL lines
Huge presence in systems research conferences
> 400 projects so far

What do people use it for?
(a few we know about)

- Overlay Networks
  - RON++, Pluto, P2, etc.
- Network measurement
  - Scriptroute, *Probe, etc.
- Application-level multicast
  - ESM, Scribe, TACT, etc.
- Wide-area storage
  - Oceanstore, SFS, SFS-RO, CFS, Ivy, Palimpsest, IBP
- Resource allocation
  - SHARP, Bellagio, Automated contracts
- Distributed query processing
  - PIER, SDIMS, Sophia, IrisLog, etc.
- Network architecture
  - Evolve, Detour, I3
- Content Dist. Networks
  - CoDeeN, ESM, UltraPeer emulation, Gnutella mapping
- Management and Monitoring
  - Ganglia, InfoSpect, Sword, BGP Sensors, etc.
- Distributed Hash Tables
  - Chord, Tapestry, Pastry, Bamboo, Kademlia, etc.
- Virtualization and Isolation
  - Xen, VServers, SILK, Mgmt VMs, etc.
- Router Design implications
  - NetBind, Scout, Network capabilities, Icarus, etc.
- Testbed Federation
  - NetBed, RON, XenoServers
- Etc., etc., etc.

The PlanetLab Consortium

- Modelled on the W3C
- Run by Universities
  - U. Washington, U.C. Berkeley, U. Cambridge, Princeton U.
  - Based in Princeton, NJ, USA.
- Funded by Industry and Govts.
  - NSF, EU, Cernet, etc.
  - Intel, HP, Google, AT&T, FranceTelecom…

The PlanetLab Consortium

- Node resources provided by member institutions
- Small "support" team NOC in Princeton
  - Additional NOCs planned in Europe (Paris), China (Tsinghua)
- Steering Committee
  - University representatives
  - Top-level industrial sponsors

Staff is small

- ~5 people in Princeton.
- Q. Who develops the software then?
- A. You do!
  - System software, tools, services, libraries, measurement data, etc. contributed by participating institutions (often researchers)
  - PlanetLab is as much a community as an artifact
- Irregular meetings debate technical direction
  - Recommendations in PlanetLab Design Notes (PDNs)
  - Announced on mailing lists and web site; all welcome
  - Modelled on early IETF

Summary of Part I

- PlanetLab is a shared, communal infrastructure of machines ("nodes")
- Nodes are hosted by institutions and connected directly to the Internet
- Each node hosts many virtual machines ("slivers") on behalf of users
- Services, experiments, applications run in distributed collections of slivers called "slices".

Part II: Basics
Accounts, slices, nodes

First steps to using PlanetLab

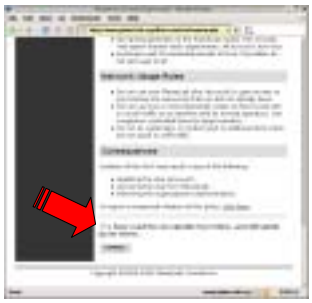
1. Register as a user
2. Create an ssh key
3. Create a slice
4. Add nodes to the slice
5. Describe the slice
6. Log into a sliver

Getting started: accounts

Go to http://www.planet-lab.org/:

Step 1: Register

The Acceptable Use Policy



What the AUP says…
(but read it carefully!)

- Research and educational use only
  - No commercial or illegal activities
- No suspicious network traffic
  - No portscans, DoS attacks, spoofing, repeated probing of routers, etc.
- Share resources responsibly
  - No tight loops, use congestion control, etc.
- Be nice and sensible!

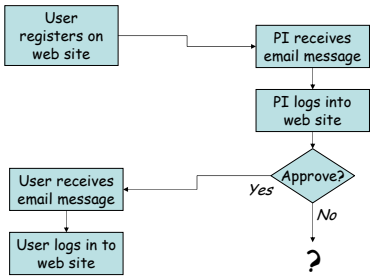
What the AUP says (contd)

- Assume no privacy
  - Assume other users, and site sysadmins, can see all your traffic.
- Assume no reliability
  - Nodes may reboot at any time, without warning. They may not come back.
- Assume no durability
  - Disks may be wiped at any time, without warning.
- In practice, this rarely happens.
  - But it *does* happen.

Filling out the form

- Fill in contact details
- Select your site
  - E.g. "`Universidade Federal de Minas Gerais`"
- Are you a PI?
  - PIs are responsible for PlanetLab site participation, approve users, create slices
  - Most users are not PIs
- Are you a tech contact?
  - Probably not!
- Submit the form…

User registration



Account approval (as PI)

Account approval (as PI)

You are now a user!

Step 2: Generating an SSH key

- PlanetLab uses 1024-bit RSA keys for authentication on nodes
- Upload your key to the website
- Using OpenSSH:
  - `ssh-keygen -q -f /.ssh/id_planetlab -t rsa`
  - *Do* use a secure passphrase

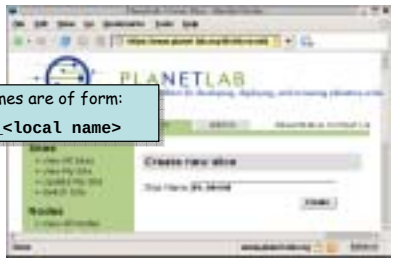
    See later for using ssh-agent to reduce typing.
  - Upload `id_planetlab.pub` to web site

Using PuTTY on Windows

- Use PuTTYgen to generate an ssh v2 1024-bit RSA key
- Copy text box to web form
  - Don't use more than 1-word comments
  - Should look a bit like this (except breaks):
  - `ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEArybtveURZwKjYY0j4ma9QQW cKD3iiSrVFhXrFs4YdSVRgnd6siYdGAJhLgsCentlQOOFhjqVtS 9AY/eIVx99ibmEDV7RrP6mk5NsPQseH/315oYbEkOmhrTuMZ2js WqiTzODQ+RCNF3iMxM+fPGmxVLRtDr2puIsXdlJqxTJYt8= troscoe@deleuze`
- Can also import OpenSSH key pairs

Step 3: Creating a slice

- Actually, ask your PI to create a slice..



Slice names are of form:
`<site>_<local name>`

Assign users to a slice (as PI)

By now…

- You have a user account
- Your PI has approved it
- Your PI has created a slice
- Your PI has assigned you as a user of the slice
- Next step: add nodes to the slice

Step 4: Adding nodes

Adding nodes

Adding nodes

- Page for adding/removing nodes
  - Select site to see nodes to add
  - Shown are all nodes currently in slice
  - Nodes can be removed
- For your first time, I suggest:
  - Add a couple of nodes, one nearby
- Note: Changes will take ~5 minutes to propagate!

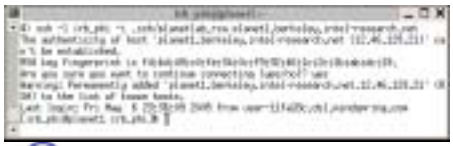
Step 5: Describe your slice

(while you're waiting for updates…)

Step 6: Log in to a node

(finally!)

- Your user name is your slice name
- E.g., with OpenSSH:
  - `ssh -l irb_phi \`
    `-i ~/.ssh/id_planetlab \`
    `planet1.berkeley.intel-research.net`



Logging in…

- Note that ssh asks you to accept the authenticity of the node
  - See later for how to avoid this
- If it works, congratulations!
- You should now have a Unix shell prompt
- Take a look around…

What do you get on a node?

- Basic Linux Fedora Core 2 machine.
- Not many packages installed
  - Compilers, etc. missing
- Nobody logged in?

- No root password?

-

What's going on?

- You're in a *VServer:* a virtual Linux kernel
  - Each sliver is a VServer.
- Limited root capabilities, e.g.
  - Install software
  - Create new users
  - Open raw sockets
- Some resource sharing/scheduling

Summary of Part II

- Registering, uploading keys, etc.
  - Done *once.*
- Creating a slice, assigning nodes
  - Typically done infrequently (once per project)
- Logging in
  - Environment is a *virtual Linux server.*

Intel Research Berkeley

Part III: Real Work Developing and Deploying

Intel Research Berkeley

Developing code: the challenge

- Writing code to run on a machine which is > 1000km away
- While libraries, etc. are needed on the remote machine?
- Copying files to remote machine…
- …and for ~400 other machines
- Keeping programs up-to-date on remote machines
- Debugging programs at a distance

Intel Research Berkeley

The good news:

- The environment is (mostly) Fedora Core 2 Linux
- Many tools have been developed to help with this problem
- Other users have provided *services* to help you
- Many researchers find this an interesting problem!

Installing packages

- It's Fedora Core 2 so…
  - Yum, RPM, apt, etc. should work
  - As root – remember, you have root!
- But give Stork a try:
  - University of Arizona research project
  - Recently released (last week!)
  - Efficient shared package manager
  - Can also be used to install your own code!

Stork

- For details, see
  - http://www.cs.arizona.edu/stork/
- Uses CoBlitz CDN for high-speed multicast of large binaries
- Shares packages between slices
- Can automatically keep packages up-to-date

Developing code: "don't"s

- Don't compile on PlanetLab nodes
  - Requires you to install compilers, etc.
  - Debugging at a distance is harder
  - **Takes valuable CPU from others!**
- Don't run X11 clients on PlanetLab nodes
  - They're server machines
  - X11 doesn't work well over a WAN

Developing code: "don't"s

- Don't treat a sliver as a workstation
  - Although it's Unix, it's intended for hosting long-running *services*, not for general use

Developing code "do's"

- Do compile and test on your local desktop workstation
  - Using Fedora Core 2 is simplest
  - Can use other distros, but be careful with libraries
  - Static linking can often simplify things
  - Java usually portable, if you install the JRE on each node

Developing code "do's"

- Do then copy binaries to the node to run
  - scp or rsync works for small numbers of nodes
  - Ensure library dependencies are satisfied

OpenSSH hints and tips

- Authentication fails:
  - Try "`ssh -v -2 …`"
- Make sure local permissions are correct:

```
chmod go-w $HOME $HOME/.ssh
chmod 600 $HOME/.ssh/authorized_keys
```

- If ssh hangs on exit:
  - redirect stdin/stdout/stderr to /dev/null
  - `shopt -s huponexit` in bash
- For more information, see:
  `http://www.openssh.com/faq.html`

Use authentication agents

- Use `ssh-agent` to avoid typing passphrase for your key
  - Note: "agent forwarding" doesn't work on nodes (yet)
  - For PuTTY users: Pageant
- Host key checking:
  - Set "`StrictHostchecking no`" in `.ssh/config`
- Add all host keys from
  `http://www.planet-lab.org/xml/sites.xml`
  (see later!)

Summary of Part III

- PlanetLab's execution environment is (today) Linux
- Develop on your local workstations, deploy on PlanetLab nodes
- For one (or few) nodes, simply a matter of copying
- BUT: how to deal with many (>400) nodes? See next section…

Break!

Timothy Roscoe
Intel Research at Berkeley
Friday, May 13th, 2005

Part IV: Tools and Services

Dealing with many nodes

- Deploying a large networked system requires:
  - *Discovery:* finding the nodes
  - *Deployment:* pushing the system out to the nodes
  - *Monitoring:* are the nodes up? Is the system up?
  - And much, much more.

Management of Planetary-Scale Services

- Bad news:
  - This is a really hard problem
  - Perhaps the central problem in distributed systems research
- Good news:
  - Researchers are working on it
  - They are using PlanetLab
  - They make their tools available for you
- … and, of course, you can write tools as well.

Simple stuff: pssh

- Brent Chun's pssh package
- Provides parallel versions of:
  - ssh
  - scp
  - rsync
  - nuke (parallel "kill" with regexps)
- Simple way to control lots of slivers
- http://www.theether.org/pssh/

Avoiding the web site

- Adding 300 nodes via web site is tedious (!)
- Solution: PlcApi
- XMLRPC interface
- All the web site's functionality
- Google for clients
- Or write your own!

Deploying software

- AppManager (Ryan Huebsh, UCB)
  - Centralized monitoring and update of software
  - Uses cron job on each node, & polling
  - Simple, but highly effective!
  - http://appmanager.berkeley.intel-research.net/
- CoDeploy / CoBlitz
  - Use the Princeton CoDeeN CDN for efficient distribution of software to PlanetLab nodes
  - http://codeen.cs.princeton.edu/codeploy/
- Stork (U. Arizona)
  - Package management, uses CoDeply for distribution

Finding nodes

- PLCAPI (again)
  - Site, machine, model
  - Disk space, RSA keys,
  - Longitude, latitude,
  - Other config information…
- http://www.planet-lab.org/xml/sites.xml
  - Collated version of the above
  - Updated periodically

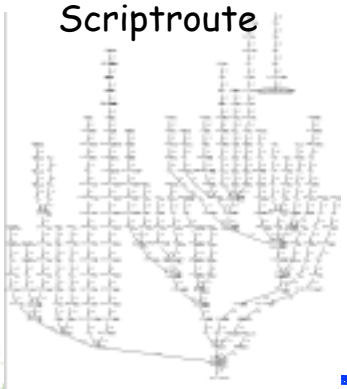
Finding nodes

- CoMon (Princeton)
  - Extensive monitoring of node liveness
  - Also "cotop" and "cotest" CLI tools
- SWORD (U.C.Berkeley)
  - XML-based queries over nodes
  - Realtime resource state
  - `http://www.cs.berkeley.edu/~davidopp/sword/`

Network measurement

- IPerf: network performance data
  `http://jabber.services.planet-lab.org/php/iperf/`
- All-pairs ping times between nodes
  - Run continously by MIT
  - `http://www.pdos.lcs.mit.edu/~strib/pl_app/`
- Scriptroute (Neil Spring, UW)
  - Highly scriptable network measurement tool
  - `http://www.scriptroute.org/`

Scriptroute



PlanetFlow

(point web browser at any node)

Resource Allocation

- PlanetLab has a rich resource allocation model
  - CPU, network, disk, etc.
  - Resources can be traded
- Several resource allocation services are emerging
- If you need more than the default, best-effort resources…

Resource Allocation

- Bellagio (U.C. San Diego):
  - Combinatorial auctions for resources
  - `https://bellagio.ucsd.edu/`
- Sirius (U. Georgia):
  - Calendaring service: reserve hard resources for a limited time
  - `http://snowball.cs.uga.edu/~dkl/pslogin.php`

Summary

- Brief tour of some tools and services for PlanetLab
- There are others: check the website, mailing lists, etc.
- More are on the way
- Please contribute your own!

Part V: Advanced Networking

Networking research on PlanetLab

- IP-level network research on PlanetLab is a bit different from on a dedicated machine
  - Each node is shared by many experiments
  - Sites limit bandwidth
  - Slices are restricted in what packets they can send

Raw Sockets

- PlanetLab uses VNET for network isolation
- Good news:
  - you can open raw sockets as normal
- Bad news:
  - you won't see everyone's packets, just your own
  - you won't be able to send arbitrary packets, just ones you could have sent anyway from a socket.
- Still: you can ping, traceroute, run a user-space TCP stack, etc…

Raw sockets part 2

- Slices with special privileges can open "true" raw sockets
  - Send / receive arbitrary packets
- "Proxy sockets" provide access to unused "dark" IP addresses
  - Used for network telescope experiments
- Capabilities handed out to "trusted" slices by the PlanetLab Consortium

Well-known port numbers

- Each PlanetLab node has a single globally routable IP address
- Hundreds of services run simultaneously
- Q. Who allocates TCP/UDP port numbers across all nodes?
- A. Right now – informally via a Wiki…

Reserved Port Numbers

sign up and claim a port number

Network measurement

- Like all comedy, often a question of *timing*
- PlanetLab nodes use NTP
  - But sometimes off – see CoMon
- CPU intensive tasks make exact timestamping difficult
  - Libpcap at least stamps packets in kernel
  - May need to busywait to send e.g. packet pairs
- Use Scriptroute where possible
  - This is what it is designed for.

Bandwidth is capped

- For many sites, bandwidth is by far the dominant cost of PlanetLab
- Per-node & per-slice b/w caps
- Note that you're sharing the link anyway
- Consider carefully what it means to measure the bandwidth!

What is your bandwidth cap?

- If you really need to know:

```
/sbin/tc -s -d class show dev eth0 \
   | grep 1:`id -u`
```

PlanetLab uses Linux Traffic Control (tc) for hierarchical fair queuing

What is your bandwidth cap?

- If you really need to know:

```
/sbin/tc -s -d class show dev eth0 \
   | grep 1:`id -u`
```

Look for the share assigned to your slice

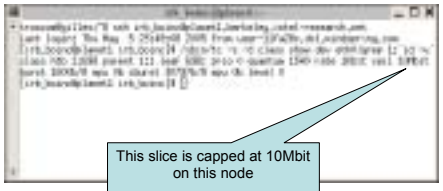
What is your bandwidth cap?

- If you really need to know:

```
/sbin/tc -s -d class show dev eth0 \
   | grep 1:`id -u`
```

PlanetLab nodes have just the one interface

What is your bandwidth cap?



This slice is capped at 10Mbit on this node

See 'man tc' for the other parameters

Firewalls and NATs

- No PlanetLab nodes are NATed
  - All have global IPv4 addresses
- Sites are *requested* not to filter any external access to ports
  - In practice, many filter ICMP
  - Few filter TCP/UDP ports
- Exception: access to the local institution
  - Often: software or library licensing based on IP addresses ☹

What about IPv6?

- Wasn't a priority early on
- Also, US-centric
  - ⇒ not supported, little demand
- All that changed with China!
- Chinese team at Tsinghua University working on IPv6 support for all PlanetLab infrastructure
- Watch this space…

Summary: Part V

- Socket programming is mostly conventional on PlanetLab
- But remember:
  - The node is shared with other researchers
  - Limits have been imposed on bandwidth

Part VI: Methodology Issues

PlanetLab as one of many research tools

- Network simulators
  - Ns2, SSF, OpNet,…
- Message-level simulators
  - Often ad-hoc, app-specific
- Cluster-based emulation
  - Emulab, Netbed
  - ModelNet
- Etc.

PlanetLab: the new NS2?

- PlanetLab is slowly changing the publishing culture
  - "real" systems must now be deployed
  - This is good…
- The danger is:
  - PlanetLab becomes *the* requirement
  - PlanetLab comes to *define* networking reality

PlanetLab Fallacies

- Symptom of its success and enthusiastic reception!
- No "methodology" consensus yet
  - Interesting area in itself
- Big difference between:
  - What PlanetLab can *teach*
  - What PlanetLab can *prove*
- Need a more critical approach to testbeds in research…

Fallacy #1: PlanetLab is representative of the Internet

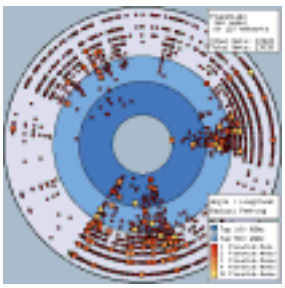
- PlanetLab is skewed towards:
  - The U.S.
  - The GREN
  - Well-connected commercial sites
- Measurement-related work on PlanetLab which extrapolates to the Internet must be **careful** in its claims!

Of course, is the Internet itself representative?

- Enterprise networks are very different
- The (public) Internet is a very diverse environment anyway
- Hence:
  - What claims is the research making?
  - What do PlanetLab results do to substantiate those claims?

PlanetLab presence

(rather out of date)



Fallacy #2: Quantitative results from PlanetLab are valid

- Experiments on PlanetLab are never fully repeatable:
  - Network conditions, machine load…
  - Comparative system measurements on PlanetLab are rarely credible
- Instead, more rigorous approach:
  - Simulation, emulation, and deployment *cross-validation*
  - ModelNet, Emulab
  - Long-term studies

Fallacy #3. Quantitative results on PlanetLab are never valid

- Longitudinal studies of service behaviour over time
- Characterize the network conditions
- Run different approaches in parallel for long periods

Reasons to be sceptical when reviewing papers:

- "As well as simulation results, we have run our system on PlanetLab"
  - Were the results commensurable?
  - What was learnt from this?
  - What difficulties were encountered?
  - Does PlanetLab match the motivating scenario?
  - Did it *really* work?

Reasons to be sceptical when reviewing papers:

- "We present results from running on *n* nodes of PlanetLab"
  - Why is *n* << 100?
  - How were the nodes selected?
  - Were the nodes "cherry-picked"?
  - How is scalability beyond *n* being demonstrated?
  - Did it *really* work?

What PlanetLab can teach

- Discover new "issues" with system designs
  - E.g. DHT stability
  - Not easily found with simulations
- Highlight bad assumptions about the real network
- Derive well-grounded principles and abstractions for building real systems
- Attract real users, real workloads, and their challenges.

PlanetLab *is*...

- Almost the only platform so far to:
  - Capture the systems challenges in wide-area distributed systems
  - Test what works and what doesn't in the wide area
  - Enable researchers to deploy long-running broad-coverage services and attract real users

Part VII: Where to go for more information

Basic documentation

- Web site: http://www.planet-lab.org/
  - FAQ
  - Wiki (contribute!)
  - Guides
  - API documentation
- Other sites:
  - Danny Bickson's guide: http://www.cs.huji.ac.il/labs/danss/planetlab/PlanetLabProjectHowto.pdf
  - Ian Wakeman's tutorial: http://www.informatics.sussex.ac.uk/research/ngn/slides/planetlab05talks/tutorial/
- Google is your friend…

Not sure what you're doing?

- The users' list:
  - users@lists.planet-lab.org
- Moderated
- V. High signal to noise ratio
- Surprisingly helpful people!
- ALL users should subscribe.

Something seems to be broken?

- Support list:
  - support@planet-lab.org
- Front-end to trouble ticket system at Princeton
- Responses within 24 hours or so
- Sometimes responses go to the users list

Want to get more involved?

- The support community list:
  - support-community@planet-lab.org
- Copy of mail sent to support@planet-lab.org, plus intra-support traffic
- Chance to see how much work they really do!

Want to get much more involved?

- The architects' list:
  - arch@lists.planet-lab.org
- Discussion of architectural directions and technical decisions
- Come to PlanetLab meetings
  - Or host one!
- Write proposals up as PDNs

If you don't like it, fix it!

- Infrastructure services can be run by any PlanetLab user
- Ask PLC nicely if you need extra access privileges
  - E.g. ability to create slices
- Tell people about your service on users@lists.planet-lab.org
- Complain about why you can't build your service on arch@lists.planet-lab.org

If you don't like it, fix it!

- Download "PlanetLab-in-a-box"
- All node OS code available from https://cvs.planet-lab.org/
- Requirement for a node is only that it implements the node manager interface (mostly).

Above all…

- Have fun!
- Enjoy the experience of running a service for real
- Go and build useful things
- Do interesting research
- Remember: PlanetLab is a community
  - It's yours as much as anyone else's.

Thanks!

Timothy Roscoe (Mothy)
timothy.roscoe@intel.com