Understanding and Characterizing PlanetLab Resource Usage for Federated Network Testbeds

Wonho Kim
Department of Computer Science
Princeton University
wonhokim@cs.princeton.edu

Ajay Roopakalu
Department of Computer Science
Princeton University
aroopaka@cs.princeton.edu

Katherine Y. Li
Department of Computer Science
Princeton University
klfive@cs.princeton.edu

Vivek S. Pai
Department of Computer Science
Princeton University
vivek@cs.princeton.edu

ABSTRACT

Global network testbeds are crucial for innovative network research. Built on the success of PlanetLab, the next generation of federated testbeds are under active development, but very little is known about resource usage in the shared infrastructures. In this paper, we conduct an extensive study of the usage profiles in PlanetLab that we collected for six years by running CoMon, a PlanetLab monitoring service. We examine various aspects of node-level behavior as well as experiment-centric behavior, and describe their implications for resource management in the federated testbeds. Our main contributions are threefold: (1) Contrary to common belief, our measurements show there is no tragedy of the commons in PlanetLab, since most PlanetLab experiments exploit the system's network reach more than just its hardware resources; (2) We examine resource allocation systems proposed for the federated testbeds, such as bartering and central banking schemes, and show that they would handle only a small percentage of the total usage in PlanetLab; and (3) Lastly, we identify factors that account for high resource contention or poor utilization in PlanetLab nodes. We analyze workload imbalance and problematic slices in PlanetLab, and describe the implications of our measurements for improving overall utility of the testbed.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design studies, Performance attributes

General Terms

Measurement, Design, Performance

IMC'11, November 2–4, 2011, Berlin, Germany.

Keywords

PlanetLab, Network Testbeds, Characterization

1. INTRODUCTION

Building on the unprecedented success of PlanetLab [23], the next generation of testbeds have been under active development recently. In its design phase, the GENI [14] project aims to federate multiple testbeds that are owned and operated by autonomous organizations. It plans to cover diverse networks including PlanetLab-like wide-area testbeds, fiber optics, and even sensor grids. To coordinate resource management across the organizational boundaries, the federated infrastructure suite needs an extensive policy framework and incentive system. However, very little is known about resource usage in the federated frameworks, which is required for designing future policy engines.

In this paper, we analyze resource usage in PlanetLab and discuss its design implications for the emerging federated testbeds. We note that PlanetLab itself is a federated platform. The nodes in PlanetLab are managed by a trusted intermediary named PlanetLab Central (PLC), but each site retains ultimate control over its own nodes. Since it was launched in 2002, PlanetLab has tried to balance fairness and the utility of the system without imposing strict resource controls [6, 22]. Thus, we believe that understanding resource usage in PlanetLab can help shape the policy decisions of future testbeds that have similar design requirements. Since planned testbeds such as GENI have architectures similar to PlanetLab, the lessons we have learned from our analysis can be generalized beyond PlanetLab to many federated systems that need to control shared resources donated by autonomous organizations.

Characterizing PlanetLab's resource usage is challenging because it is highly dynamic and evolves with changes in the underlying platform. For example, some experiments are active year-round and consume an almost constant amount of resources while many other experiments show heavy and bursty demand over short time periods. As a result, large-scale, long-term analysis is necessary to capture usage patterns and their evolution.

To address this challenge, we have collected detailed statistics on every online PlanetLab node and the active experiments running on the node since August 2004 through the PlanetLab monitoring system CoMon [20]. The collected

datasets have detailed information about both node-centric and experiment-centric data at five minutes granularity. In addition to passively recording OS-provided metrics, CoMon also actively gathers information about each node's status by periodically running a set of test programs. In this paper, we analyze six years of PlanetLab usage, from 2005 to 2010. Our three main observations follow:

No tragedy of the commons Conventional wisdom suggests that network testbeds should suffer from a tragedy of the commons, and this belief has led to much development on PlanetLab, including two deployed resource reservation schemes [16, 27], two deployed resource discovery systems [1, 2], and papers investigating resource allocation and migration [12, 19]. This belief has even shaped the requirements of testbeds like GENI, which are devoting much attention and software development cost to resource reservation systems [13].

However, we observe no indication of the tragedy of the commons on PlanetLab, and we find several measurements indicating that these kinds of network testbeds are unlikely to suffer such effects. Unlike compute clusters where users try to utilize every available resource, most PlanetLab users are not aggressive in using resources in the testbed. While PlanetLab hosts some long running services, most PlanetLab experiments have bursty resource consumption, and this resource consumption is tied to network activity. As a result, the resource consumption shows bimodal distributions along many axes. The primary reason for the non-aggressive behavior of PlanetLab users is that the main utility of PlanetLab comes from its wide network vantage points, not the aggregate amount of resources.

Limitations of market-based resource allocation Using data-driven analysis, we explore the effectiveness of two representative resource allocation schemes proposed for PlanetLab-like federated systems: pair-wise bartering and market-based banking. We find that the bartering and banking systems can account only for 3% and 14% of the total resource usage respectively, because most resource usage is from sites that do not donate the same amount of resources. Since the remaining 83% of the resources need to be allocated, market-based allocation approaches are not sufficient for network testbeds, and some mechanism must be employed to ensure that the bulk of the testbed's resources are used appropriately.

Improving utility of PlanetLab We examine factors that degrade the overall utility of PlanetLab, and discuss how to mitigate their impact. We find that workload is persistently unbalanced among PlanetLab nodes, resulting in high system lags in overloaded nodes as well as inefficient resource usage. Several factors are responsible for this imbalance, ranging from users staying with known-good nodes to node utility being degraded due to DNS failures, node unreliability, bandwidth limitations, and other reasons. We also find unstable experiments consume a disproportionately high share of the resource, typically dwarfing stable long-running services. We simulate pruning the problematic experiments to measure their impact on other well-behaved experiments in PlanetLab.

The rest of this paper is structured as follows. In Section 2, we describe some background on PlanetLab and CoMon datasets. We analyze per-slice characteristics in Section 3, and examine several resource allocation systems in Section 4. We examine the workload imbalance problem in Section 5, and discuss policing of slices in Section 6. We compare our observations with related work in Section 7. Section 8 concludes the paper.

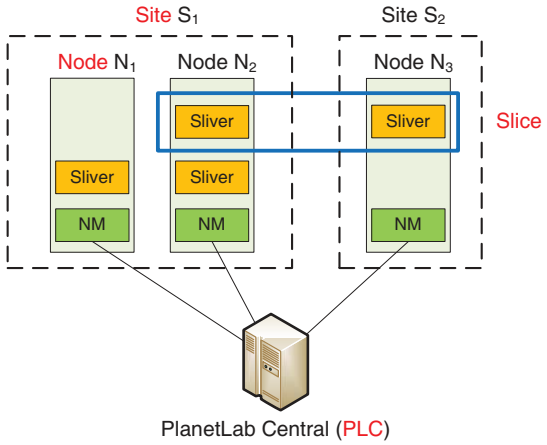


Figure 1: Overview of PlanetLab Architecture.

Year	Nodes	Slices	LiveSlices	Size
2005	354 (62)	215 (14)	106 (8)	164.8 GB
2006	433 (33)	278 (38)	136 (18)	232.9 GB
2007	438 (77)	371 (54)	133 (21)	260.7 GB
2008	474 (85)	254 (66)	139 (25)	291.2 GB
2009	613 (55)	349 (103)	145 (19)	430.1 GB
2010	683 (67)	421 (62)	158 (15)	503.9 GB
Total				1883.6 GB

Table 1: Summary statistics for CoMon datasets. Each row contains means and standard deviations of online nodes, in-memory slices, and live slices per day in each year.

2. BACKGROUND AND DATASETS

To better understand the analysis in this paper, some background on PlanetLab and its terminology is provided here. Figure 1 illustrates the architecture of PlanetLab and its components. When organizations join PlanetLab, they host physical servers at one or more locations. Each location is called a *site*, and the servers are called *nodes*. All account creation and node management is handled by a centralized database, called PlanetLab Central (PLC). Users create accounts on one or more PlanetLab nodes to perform their experiments. The nodes host one virtual machine per account, and users can run any number of processes within their own slivers. The virtual machines are called *slivers*, and the set of virtual machines assigned to one account is called a *slice*. PlanetLab is a shared testbed, so multiple slivers are running on the same node at any given time.

We classify a sliver as an *in-memory* sliver if it contains at least one instantiated process, regardless of whether the process is running or blocked. An in-memory sliver is called a *live* sliver if it uses more than 0.1% of the CPU per day[1]. A slice that has at least one live sliver is called a live slice. We say that a slice *uses* a node if the slice has in-memory slivers in the node. A node is considered to be live if it responds to CoMon requests.

[1]0.1% is the minimum CPU time that CoMon measures for a sliver's CPU usage at any given time.

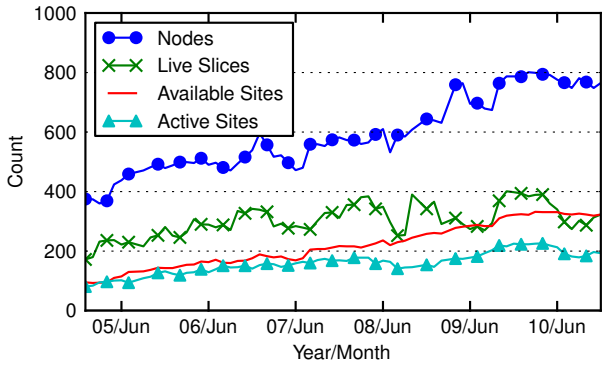


Figure 2: The scale of PlanetLab over time. PlanetLab node and live slice counts have increased as more researchers have joined the testbed in general.

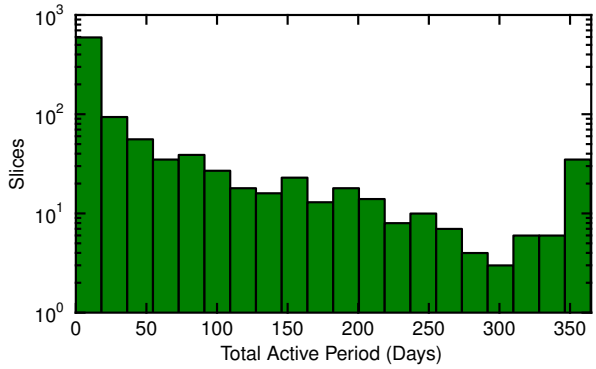


Figure 3: The distribution of slice total active period. While most slices are short-lived, a number of slices were active for an entire year.

In this paper, we analyze six years (2005 to 2010) of data from CoMon, a scalable monitoring system for PlanetLab. Since August 2004, CoMon has collected and reported statistics on PlanetLab nodes to help PlanetLab users monitor their services and spot problems. CoMon runs daemons in every PlanetLab node to gather values that are provided by operating systems, and values that are actively measured by means of test programs running on the nodes. A central CoMon server collects data from all PlanetLab nodes every 5 minutes.

CoMon monitors and collects *node-centric* and *slice-centric* data. The node-centric datasets consist of 51 fields that represent node health and aggregate resource consumption, including CPU utilization, memory usage, timing behavior, DNS resolver behavior, bandwidth consumption, etc. The slice-centric data contains information about each sliver's resource usage on its node, which is an aggregate resource used by all processes within the sliver in the node. The measured metrics include CPU usage, memory consumption, and transmit/receive bandwidths.

Table 1 shows the basic statistics about our datasets. The sizes of the datasets have increased over time because PlanetLab's node count has increased and more metrics have been added to CoMon over time. The slice-centric datasets contain resource usage of each sliver, and can be aggregated into slices as needed. Since CoMon fetches data from all PlanetLab nodes in parallel, the aggregated values estimate the total amount of resources that a slice uses across multiple nodes at a given time. We associate the two kinds of datasets to study the effect of an experiment's behavior on a node's status, and vice versa.

CoMon's task has grown over time as the testbed itself has expanded, since CoMon tries to monitor information about every sliver in the system. Figure 2 presents the numbers of online nodes, live slices, sites running online nodes (labeled as "Available Sites"), and sites having live slivers in other remote nodes ("Active Sites") per month. We find that the scale of the testbed has persistently increased over the time period (2005 to 2010) that we examine. The number of available sites and their nodes has increased by 179% and 82%. The active site count has increased at a slower rate (95%) because most PlanetLab users intermittently run their experiments in the testbed, and their usage of the testbed is spread over time. Similarly, the number of live slices has increased by 48%. In particular, it is notable that live slice count has been fluctuating over time, which implies that resource demands on PlanetLab are synchronized to some degree with external events such as conference submission deadlines. Sliver count has increased at a faster rate (123%) than slices because slivers counts grow as a result of slice growth and node growth. Most slices only create slivers on a fraction of the nodes, but some slices, particularly those related to infrastructure, are typically created on every node in the system.

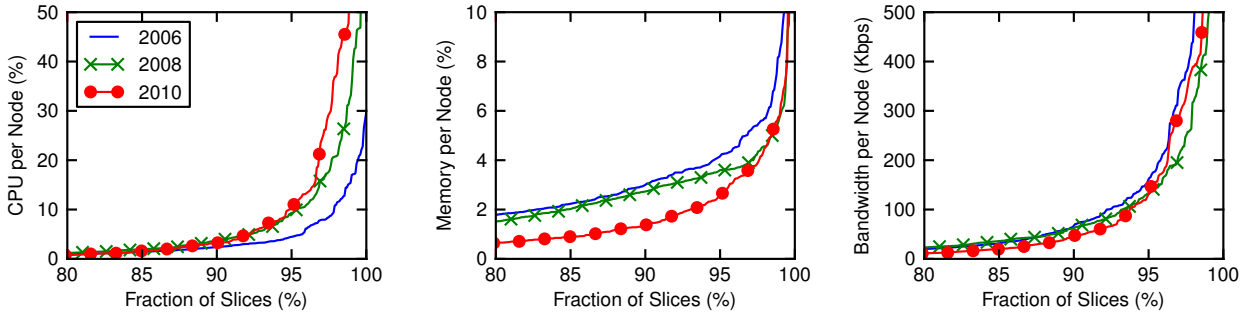
3. SLICE RESOURCE USAGE

Examining the slice resource usage in PlanetLab allows us to determine how experiments are using the system, and the patterns of resource consumption on the testbed. We find that PlanetLab experiments are typically bursty along several dimensions, and that most use relatively few resources at any given time. This likely stems from the network-centric experiments on PlanetLab – their resource consumption is tied to their network activity, rather than the total resource pool on PlanetLab.

3.1 Active Periods

Since PlanetLab slices share nodes, we begin our per-slice analysis by examining how long slices tend to run and actively use resources. We define a sliver's *active period* to be the number of hours during which the sliver is continuously live on its node. Experimenters typically leave slivers instantiated on nodes for long periods of time, and only use the slivers when actively performing experiments, resulting in multiple active periods separated by idle periods in CoMon datasets. We consider a slice active if any of its slivers are active, even if sliver count changes over time.

We find that slice activity is largely bimodal, with a great many short-lived slices and a number of very long-lived slices, as shown in Figure 3. The number of short-lived slices is not surprising, since many classes use PlanetLab for hands-on measurement projects and short assignments. It is also notable that there are 26 slices that were live longer than 360 days in 2010. These slices include 6 management slices (e.g., root and SliceStat [28]), and 20 long-running services that run on PlanetLab [7, 8, 11, 15, 21].



(a) CPU (b) Memory (c) Bandwidth

Figure 4: Per-sliver usage of CPU, memory, and bandwidth for slices in 2006, 2008, and 2010. Most slices have low resource consumption except for the heaviest 5% of slices. The CPU usage shows the heaviest slices gaining a larger share over time, while memory usage shows flatter curves. The heaviest bandwidth consumers typically provide services to large external user populations.

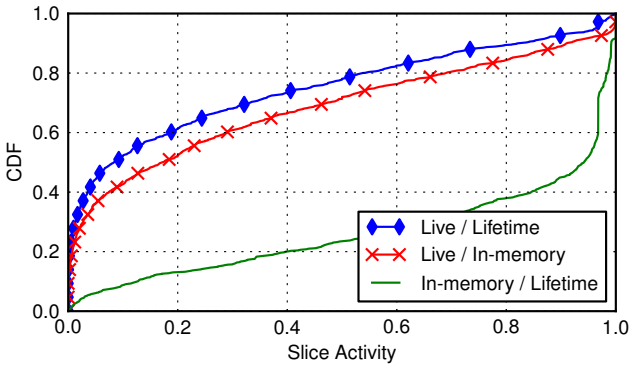


Figure 5: CDFs of the relative activity of slices over their lifetimes. Versus their lifetimes, most slices have relatively small periods of activity (Live/Lifetime). The ratio of activity is even low when compared to the time a slice is instantiated (Live/In-memory).

One possible observation from this data is that short-lived slices are an important aspect of PlanetLab usage, and that it may serve as a training facility for future developers. As such, all of the setup overhead on PlanetLab may be an issue for this class of user, who has to perform these tasks and then amortizes that effort over a relatively short usage. Testbed designers may be well advised to focus on simplicity as a way of gaining usage.

The other implication of this result is that most slices do not use a significant amount of resources, even when they are active. Shown in Figure 5 is that the total active periods of the slices is often spread over a much longer slice lifetime (Live/Lifetime), so many of those slices are idle for most of their lifetimes. The ratio of activity is even low when compared to the time a slice is instantiated (Live/In-memory). Slices often tend to also disappear and re-appear over time, with large gaps in time when they are not present at all on the testbed. Any attempt at introducing heavyweight resource allocation mechanisms would therefore have two side effects – it would burden the users of most slices, and it would often require resource overcommitment anyway in order to ensure that the resources are being used.

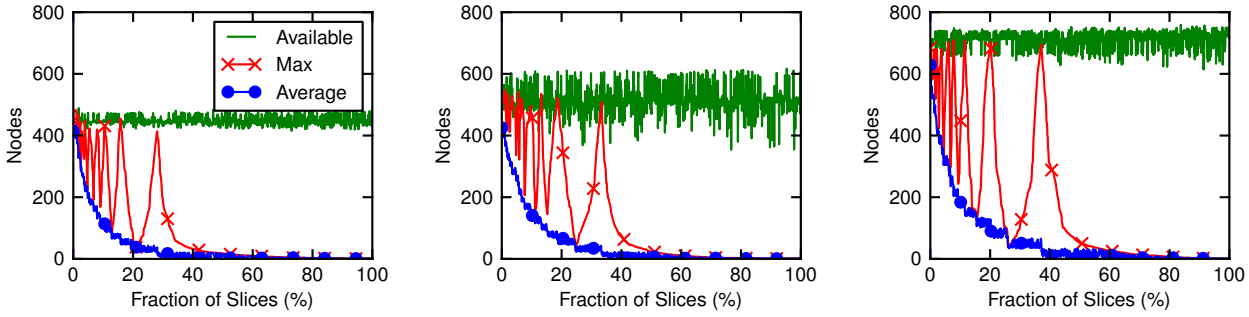
3.2 Local Resource Consumption

Understanding the distribution of slice resource usage and its change over time provides insight into the workload profiles of PlanetLab experiments. We focus on three resources – CPU, memory, and network bandwidth, and examine them on a per-sliver basis when the sliver is active. We do not include disk usage in our analysis because PlanetLab disk space is partitioned into per-sliver quotas (5 GB), and not shared by multiple users in a node.

Since most slivers have fairly low resource consumption, we focus only on the heaviest quintile of resource consumers, shown in Figure 4. The three graphs have similar characteristics, in that the aggregate resource consumption is a few percent at most, and increases sharply as we approach the heaviest 5% of slices. However, important differences are apparent when examined in closer detail. The CPU usage, for example, shows the heaviest slices gaining a larger share over time, while memory usage shows the exact opposite.

We believe these differences are a result of PlanetLab policy and the changes in hardware over time. PlanetLab uses a modified CPU scheduler [22], which allocates CPU evenly across slices (not threads/processes), and then allocates any unused CPU on demand. Over time, as more machines enter the PlanetLab testbed, and more powerful machines enter the system, the aggregate CPU in the system increases. As a result, more slices can have their CPU demands met, and the remaining CPU is used by the heaviest consumers. However, these heavy CPU consumers are not heavyweight long-running services, but instead are classified as spin-loop slivers that consume many CPUs but do not generate any network traffic, which we cover in more detail in Section 6. This result suggests that CPU contention in PlanetLab has been decreasing over time.

The CDF of memory consumption differs in that the curves are much flatter, and that the opposite effect occurs over time, with the heaviest consumers using less of the testbed. One reason that partly explains both features is that PlanetLab's policy for memory allocation is that when a node runs out of swap space, the heaviest consumer of physical memory is killed on that node. As a result, slices have a tendency to police their own memory usage to avoid being the heaviest consumer, leading to a flatter memory consumption profile among slices. Over time, as the memory capacities of the



(a) 2006 (b) 2008 (c) 2010

Figure 6: The average and maximum daily sliver counts for slices in 2006, 2008, and 2010. Most slices have a low average number of slivers, but a large number of them have relatively high maximum sliver counts in their lifetimes.

nodes have increased, the self-policing behavior introduced by the memory-killing policy results in slices consuming a smaller fraction of memory over time.

Bandwidth consumption is related to the testbed itself – for those experiments without a large user population, bandwidth consumption is driven by the experiment itself and whatever bandwidth caps the nodes have been assigned. The heaviest bandwidth consumers typically involve large external (non-PlanetLab) user populations, such as content distribution networks or peer-to-peer systems, and the consumption of these systems is not captured on this graph.

3.3 Slice Sizes and Dynamics

As we have seen that PlanetLab's resource distribution and slice distribution has many bimodal properties, it is worth investigating whether PlanetLab is monopolized by only a handful of researchers, or whether it has a broader utility to the community. As we believe that PlanetLab's main differentiator versus other testbeds is its network reach, one measure of this utility would be to see how different slices use PlanetLab's scale.

To visualize the range of sliver usage within slices, we want to view the average and maximum daily sliver counts for the slices. However, viewing this data sorted by only one of these values results in garbled images since the average and maximum values are not necessarily correlated. To address this problem, we divide slices into 20 groups by their daily average sliver counts. We then sort slices within each group by their maximum sliver counts. The sort order alternates between ascending and descending for the different groups for aesthetic appeal. The results for 2006, 2008, and 2010 are shown in Figure 6.

These graphs show a number of interesting features – the first is that most slices have a relatively low average number of slivers, with 75% of the slices using less than 7% of the available nodes on average, and only 4% use more than half the available nodes on average. However, we also see that the maximum number of nodes used approaches the total size of PlanetLab for virtually every range of average node counts. This result suggests that while much of the development of experiments may happen at low sliver counts, many researchers are in fact expanding their experiment to a large fraction of the available nodes at some point in the slice's lifetime.

This usage pattern becomes more apparent when we aggregate the slices into the sites that created them, and then examine network reach, as shown in Figure 7. For each site, we combine the locations of all slivers created by researchers from that site, and examine the locations of those slivers, by site and continent. We find that 105 sites were purely donating resources in 2010, and did not run any slices of their own on the rest of the network. At the same time, 132 sites (35%) used more than 200 remote sites (53%). Taking the inactive sites into account, this result implies that 50% of all active sites used more than half of all available sites for their experiments, which would be impossible without PlanetLab-like global testbeds. Similarly, 73% of all active sites used nodes in every accessible continent (North America, South America, Europe, and Asia) in PlanetLab.

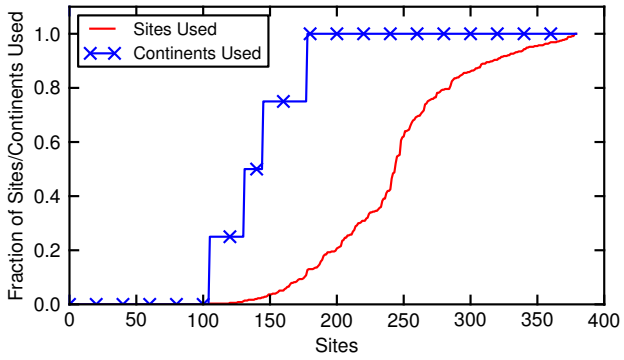
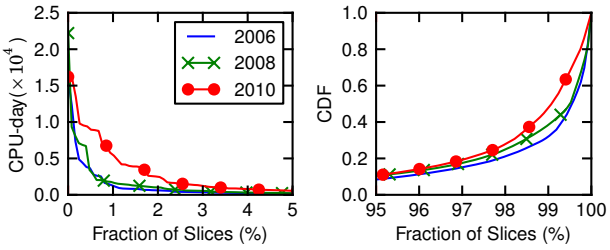


Figure 7: The distribution of network reach that PlanetLab sites used in 2010. More than 50% of all active sites used remote nodes in half of all available sites and every accessible continent in PlanetLab.

Combined, these two results demonstrate the main utility of PlanetLab – it allows researchers much larger network reach than they would have from just their own sites. Snapshots over small time periods are likely to understate this usage, since most experiments run on a small number of nodes for most of their lifetimes, but a large number of them expand to over half the testbed at some point in their lifetimes. This kind of utilization of the network is not likely to be captured by examining CPU or memory resources, as would be appropriate for computer clusters.



(a) Per-slice CPU usage (b) CDF of total CPU usage

Figure 8: Total CPU consumptions by slices in 2006, 2008, and 2010. Only 3% of all slices can account for 80% of all CPU usage in PlanetLab.

3.4 No Tragedy of the Commons

From the details presented earlier in this section, we find no measurement-based support for the idea that there is a tragedy of the commons on PlanetLab. Most experiments are relatively small most of their lifetimes, and use a large fraction of the testbed in a bursty manner. Likewise, resource consumption is relatively low for most slices, and the fact that the largest CPU consumers are runaway processes suggests that many more slices could get more CPU if needed.

The related question is that if PlanetLab appears to have excess capacity, why is that capacity not being (even surreptitiously) tapped by non-networking researchers? We believe that several PlanetLab policies make it unappealing for compute-intensive researchers. The first is that PlanetLab is organized with a small number of nodes (typically 2) per site, and a large number of sites. This model is different from compute grids, which have a large number of nodes per site connected with high-bandwidth local-area networks. In comparison, the external bandwidth capacity at many PlanetLab sites is in the range of 1-10 Mbps, so the ratio of CPU to bandwidth is much different than compute grids. Not only does PlanetLab have a worse bisection bandwidth than LANs, but the latency between nodes is much higher due to the physical distance. The other issue is the available memory – a large memory footprint increases the chances of a sliver being killed, so memory-intensive compute applications are not well-matched to PlanetLab. This combination of high CPU and low bandwidth is typical of certain bag-of-tasks parallel applications, such as SETI@home [26], but volunteers can provide far more CPUs than are available on PlanetLab.

4. RESOURCE ALLOCATION

Since the usage behavior of PlanetLab experiments is very different from compute-intensive testbeds, decisions on resource allocation policy are likely to also be impacted. In this section, we use the historical usage data to examine the resource allocation systems that have been proposed for federated network testbeds.

4.1 Total Resource Consumption

To understand the impact of resource allocation proposals, we must first understand resource consumption, which has many dimensions, such as node count, length of running time, resource consumption per node, etc. To capture the different usage patterns and to reduce the dimensions of the problem, we focus on the contended resources, CPU and memory, and aggregate per-slice usage across time and across the entire testbed.

We represent a slice's total resource usage in units of *CPU-day* and *MEM-day*. A CPU-day means the total CPU time that a single CPU core provides per day. Likewise, we define a MEM-day as the total memory space that a node provides per day. If a sliver uses 20% of CPU time and 10% of memory space in 10 nodes for 2 days, its total resource usage is 4 CPU-days and 2 MEM-days.

Figure 8a presents the distributions of per-slice total CPU usage in decreasing order. The top 1% of slices use more than 10^3 CPU-days while the medians are below 0.1 CPU-days in the years that we examined. PlanetLab slices consumed in total 0.9×10^5, 1.2×10^5, 2.2×10^5 CPU-days in 2006, 2008, and 2010 respectively. Figure 8b presents a detail from the CDFs of the total CPU usage. We find that only 3% of all slices can account for more than 80% of all CPU usage in PlanetLab. Memory usage has a similar pattern, with 4% of all slices account for more than 80% of all memory usage.

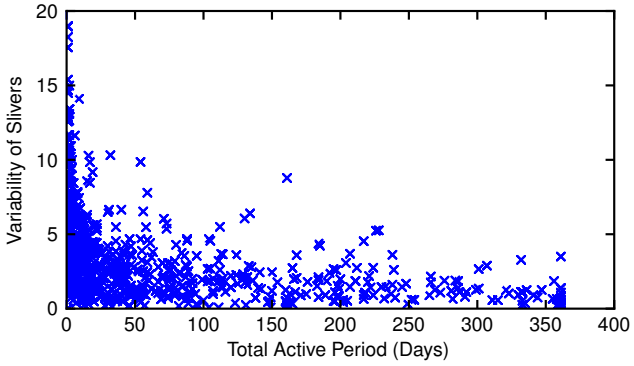


Figure 9: The distribution of each slice's total active period and coefficient of variation in its sliver count over time. Long-running slices show relatively lower variability than short-lived slices.

Slice group name	Slice count	Percentage
Short	718	56
Medium	370	29
Long-intermittent	171	13
Long-continuous	16	2
Total	1275	100

Table 2: The distribution of slice groups. The majority of slices are in the short or medium slice groups.

4.2 Resource Usage by Experiment Type

To understand what kinds of slices are creating resource demands and the extent of their demands, we categorize slices into several groups and compare their aggregate resource usages.

We first divide slices along two axes, the variability in the number of slivers and the lifetime of the slice. Using these groupings, we would expect an infrastructure service to have a long live and a stable sliver count, while a bursty experiment would have a short life and a variable sliver count. Figure 9 plots each slice's total active period and variability

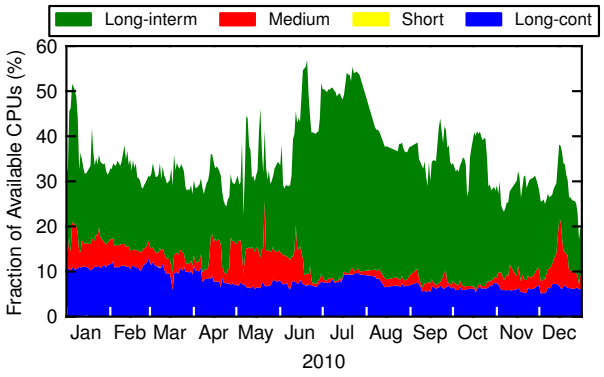


Figure 10: Time series of the distributions of CPU usage by each slice type in 2010. The y-axis represents the fraction of available CPUs consumed by slices per day. The Long-intermittent slices consume the largest amount of the resources with high variation.

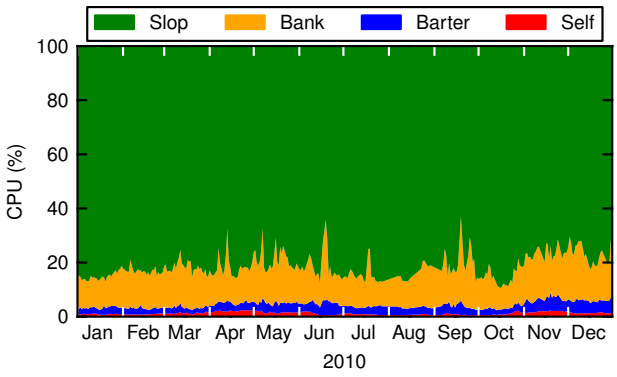


Figure 11: Time series of the distributions of CPU usage that could be addressed by several resource peering schemes. Barter and Bank can account for only 17% of the total CPU usage on PlanetLab because most CPU usage is from Slop.

of its slivers. The variability of slivers is measured as the coefficient of variation in a slice's sliver count over time. We find that long-running services show relatively lower variability than short-lived slices in general. For simplicity, we pick some break points, with Short slices have less than a week of total activity per year, Medium slices having between one week and 100 days of activity, and Long slices as have more than 100 days. We further divide the long slices into continuous or intermittent based on the normalized deviation of sliver count, with continuous slices having a deviation of less than 0.25. The number of slices in each of these groups is shown in Table 2.

Daily aggregate CPU usage across the testbed, divided by the slice types, is shown in Figure 10. We first calculate the total CPU usages by all slices per day. The Short slices, despite being over half of all slices, show virtually no CPU usage, while the Long-continuous slices, at 2% of all slices, consume roughly 5-12% of all available CPUs over time. The Medium and Long-intermittent slices are much more bursty in their CPU usage, with the Medium slices showing the least activity during the summer months and the start of the academic year. This behavior would correlate with PlanetLab being used for coursework and projects during the academic year.

Upon closer inspection of the slice groups, we find that the slices in the Long-continuous group run infrastructure services. The slices provide package management [7], monitoring [17, 20, 28], or scalable file distribution services [11, 21, 30]. These slices, while often heavy bandwidth consumers, are surprisingly not a huge impact on PlanetLab CPU, presumably because a production-quality service that has external users must take some care to run stably. On the other hand, the Long-intermittent slices are the primary source of the fluctuation in the workloads of the testbeds. For example, we find that many spin-loop slices, described in Section 6, belong to this group.

4.3 Resource Allocation Systems

We extend our analysis to explore the effectiveness of alternative resource allocation schemes in PlanetLab. Among the various resource allocation systems proposed for federated platforms, we focus on two representative approaches in this section: *pair-wise bartering* and *centralized banking*. Other schemes, such as chaining resources among a subset of the nodes [12], would fall between these two extremes. We examine how well the PlanetLab workload could be addressed by the alternative resource allocation systems if they were widely deployed in PlanetLab.

The two schemes we have selected represent the envelope of resource allocation schemes, since pair-wise barter is the most restrictive and banking is the most permissive. The two schemes assume that users trade their resources with each other, or bid their resources to reserve remote resources in other sites. In bartering, a site A grants a certain units of A's resources to site B in exchange for access to the same units of B's resources. This peering enables the sites to trade their resources without central agreement. In the central banking system, a site earns virtual currency budgets based on the amount of its donated resources, and obtains remote resources by spending its balance.

To calculate the amount of the PlanetLab workload that could be addressed by the resource peering schemes, we break each site's total resource usage into four categories: Self, Barter, Bank, and Slop. If a slice from site A runs on nodes owned by the same site A, we classify the slice's resource usage as *Self*. In bartering, each site keeps separate balances for the other sites. Site A's balance for site B, bal_{AB} increases when B uses resources at A's nodes, and decreases when A consumes B's nodes over time. If A uses r_{AB} resources at B, we classify $min(r_{AB}, bal_{AB})$ as *Barter* at a given date. In banking, each site has a balance across the testbed. A's balance, bal_A increases when other sites use resources at A's nodes. If A consumes r_A resources in total at other sites, then we classify $min(r_A, bal_A)$ as *Bank* at the given date. Lastly, the remaining resource usage is called *Slop*, which is the amount of resources that sites used beyond what they contributed.

For simplicity and generality, we make a number of assumptions about the resource allocation schemes, but these do not lead to a loss of generality in the results. For bartering schemes, we assume a uniform exchange rate, rather than a dynamic exchange rate proposed by some systems [4, 9, 16], since the dynamic rate would only restrict some of the exchanges we observe. Similarly, in central banking, we do

not impose any upper limit on resource balances, which allows us to capture all exchanges that could be performed in a central banking model. We consider the effect of resource limits later in this section.

Figure 11 presents the amount of testbed-wide CPU usage addressed by each category. As expected, the percentage of Self usage is low, since sites would have little reason to join PlanetLab and use solely their own machines. The Barter approach handles on average less than 3% of the CPU usage on PlanetLab, and the Bank approach handles an additional 14% beyond Barter. The vast majority of the CPU usage, however, cannot be handled by any of these approaches, and is allocated from Slop. This implies that, no matter the underlying exchange rate mechanism, there is not much demand for resource that the resource peering schemes are able to handle. It also means that there has to be a policy to handle allocation when bartering or banking fails since most resources will be allocated via Slop. For memory usage, the Bank and Barter schemes show slightly higher percentages (19% total) than for CPUs, but the results are consistent with CPU usage in that most memory usage also comes from Slop. The underlying cause of these results is that CPU demand is unbalanced, with most sites using a large network reach but relatively little CPU. Schemes that attempt to allocate resources must contend with the fact that, for the vast majority of users, CPU is in relatively low demand.

We extend our simulation of the pair-wise bartering scheme to examine the effectiveness of resource routing in PlanetLab, which enables sites to access resources in more remote sites through some transitive ticket redemption paths. If site A has a ticket to claim on B's resources and B has a ticket to claim on C's resources, A could request C's resources using the chained redemption of its ticket to B ($A \rightarrow B \rightarrow C$). For conservative evaluation, we assume that every site has the global knowledge of the distribution of tickets at a given time, and we set no limit on the length of the path that can be used to claim resources.

We find that such resource routing enables PlanetLab sites to find 3.2 times more CPUs than with the pair-wise bartering scheme, but that the total usage addressable by chained approaches is still less than 7% of total CPU usage. Even with unlimited chain lengths, the chained bartering schemes have a result that is much closer to pair-wise bartering than to central banking, which suggests that the chains are likely to be fairly short. Indeed, we find that the maximum length of the used path was 18 hops, but most of the CPU addressed (85% of the 7%) in bartering was found in sites within 4 hops.

Despite the Bank approach addressing only 14% beyond Barter, any feasible implementation would achieve even less since some limit has to be placed on the balances each site can accumulate. Using the data for January 2010, we examine the bank balance growth in Figure 12, and show the effect of various limits. With no limit on balances, the total bank balances grow quickly and exceed the daily capacity of PlanetLab ("Available CPUs") within three days. Beyond this limit, the virtual currency becomes inflated, as more currency is accumulated than can ever be spent. With balance limits enforced, the total balance converges at certain points. We find that five CPU-days limit (or lower) can prevent the total balances from growing beyond what PlanetLab can serve. Since most nodes in PlanetLab are dual-core or quad-core machines (82%), the five CPU-days correspond

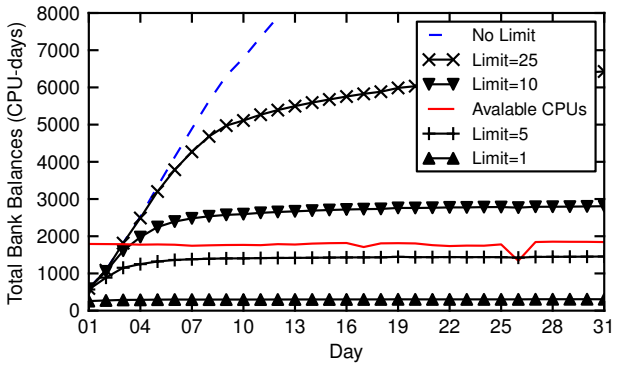
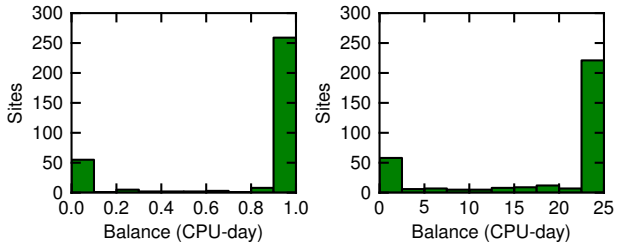


Figure 12: The total balances amassed at all sites. Without balance limits, the total balances will exceed the daily capacity of PlanetLab within 3 days, leading to inflation of virtual currency.



(a) 1 CPU-day Limit (b) 25 CPU-day Limit

Figure 13: The distribution of bank balances among sites. There are bimodal distributions with most sites being near the limits.

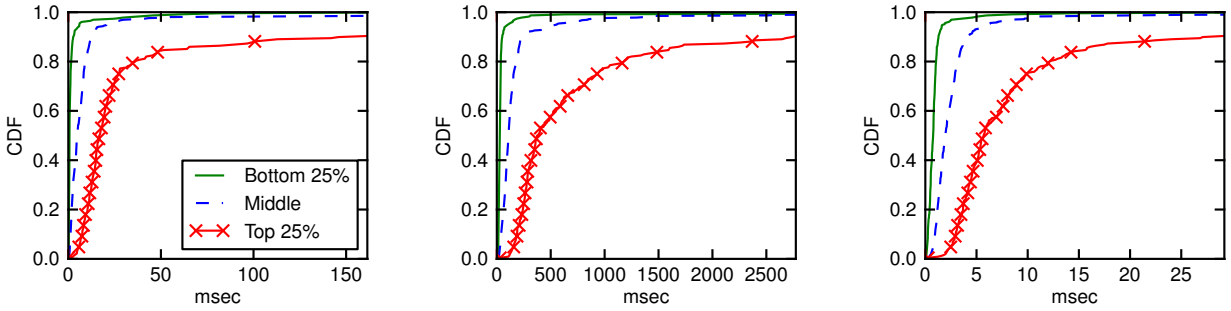
to two physical machines, which is what each site typically provides.

Even if the total balance is bounded, the utility of banking and resource auctions depends on users willing to outbid each other in order to get access to resources. However, we find that bank balances tend toward bimodal distributions, as shown in Figure 13. The majority of sites hit the balance limit, and a fraction of sites are constantly at a zero balance, with similar patterns at balance limits of 1 CPU-day and 25 CPU-days.

If banking is intended to solve the problem of demand before external events, such as conference submission deadlines, this bimodal wealth distribution suggests that auctions will fail, since all sites have the same amount of currency to bid on the same resource, and the utility of the resource presumably drops to zero after the paper submission deadline.

Conversely, when no external deadline exists, the bank balances provide little benefit, since PlanetLab already has most of its resources being allocated via Slop. So, a banking scheme would allow all of the current demand to be satisfied, without providing any additional benefit beyond what is currently present.

If banking were employed to reduce the resources being allocated via Slop, then a policy decision has to be made regarding the testbed. From our earlier examination of consumption, the Long-continuous slices were responsible for far less CPU consumption than the Long-intermittent slices.



(a) ServTest-max (b) Timer-max (c) SleepLoop

Figure 14: CDFs of system lags in nodes grouped by their average CPU load.

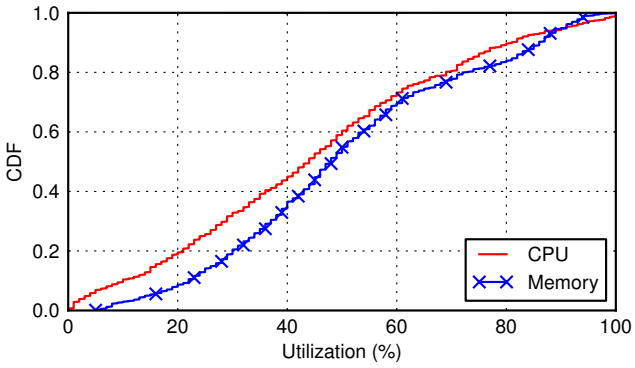


Figure 15: CDFs of average CPU/Memory utilization of all PlanetLab nodes in 2010.

One may decide that Slop should be allocated preferentially to continuously-running services in order to increase the visibility (internally and externally) of PlanetLab. In any case, the decision becomes how to allocate the Slop, not how to use banking, suggesting that banking by itself provides little utility.

5. WORKLOAD IMBALANCE

Despite the availability of resources on PlanetLab, some level of contention does occur from the testbed being shared, leading to workload imbalance. In this section, we examine the workload distribution in 2010, and analyze the degree of imbalance in PlanetLab and its effects on the system. We analyze some reasons for the distribution, and explore solutions based on these observations.

By measuring average CPU and memory utilization by node, we can see a persistent difference during the year, as shown in Figure 15. The effects of this imbalance are also quantifiable, using CoMon's metrics regarding system lag and timing. These metrics are related to the responsiveness of networked systems, so load-induced timing problems would degrade PlanetLab's overall utility. The metrics used are

- **ServTest** – Measure latency to make a loopback connection and receive a byte from it. Calculate maximum and average values over previous 60 runs. Used to measure connectivity responsiveness.

Field	CPU-Low	Med	High
ServTest-max	2.71	12.95	150.57
ServTest-avg	0.29	0.62	3.66
Timer-max	38.02	282.75	3247.94
Timer-avg	10.35	10.62	15.01
SleepLoop	1.28	4.15	27.72
SpinLoop	0.49	0.98	7.48

Table 3: The 90^{th} percentile values (in milliseconds) of the system lag metrics in nodes with low, medium, and high CPU load.

- **Timer** – Measure latency in wake-ups from 10 msec sleeps. Calculate maximum and average values over last 60 runs. Used to measure load on the scheduler.
- **SleepLoop** – Run 11 spin-loops with 10 msec sleeps in-between. Measure gaps between the loops and calculate (max - min) of the gaps. Used to simulate the behavior of low-rate measurement activity.
- **SpinLoop** – Run 11 spin-loops without sleeps. Calculate a diff value like SleepLoop. Used to simulate the behavior of high-rate measurement activity.

We divide PlanetLab nodes into three groups based on their CPU load shown in Figure 15: top 25%, bottom 25%, and the remaining 50% of all nodes. Then we compare the values of the lag-related metrics between the groups (Figure 14). It is noticeable that the nodes with high CPU load show two orders of magnitude of increase in the metrics compared to other lightly loaded nodes. This latency may degrade responsiveness of network services or add measurement noise to running experiments. Table 3 summarizes the 90^{th} percentile values of all the metrics in each group. We observe similar results in the nodes grouped by their memory load. Such timing increases complicate experiment design and add noise to measurements. On the positive side, long-running services will have to develop mechanisms to deal with these issues, which are also likely to occur in the real world, thereby making their services more robust outside of testbed environments.

5.1 Origins of Imbalance

While one may expect that a certain amount of workload imbalance is naturally to be expected in a large testbed, we believe that the imbalance on PlanetLab has other more

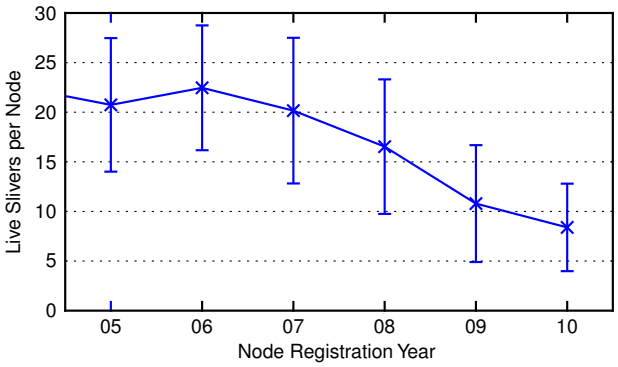


Figure 16: The number of live slivers per node in 2010. The recently registered nodes serve lower number of live slivers than older nodes. The error bars represent standard deviations.

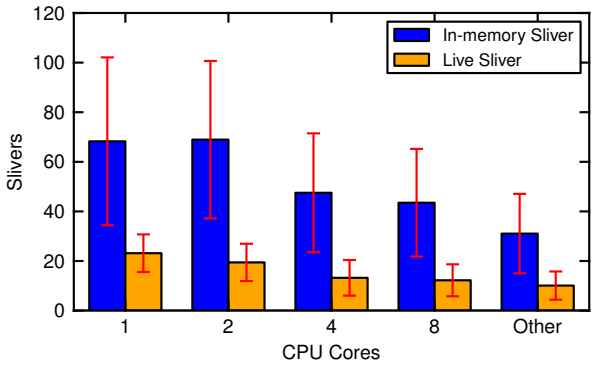


Figure 17: The distribution of slivers in nodes based on the number of CPU cores per node. The number of in-memory and live slivers shows a generally decreasing trend as the number of cores increases, which is responsible for some of the measured workload imbalance. The error bars represent standard deviations.

identifiable causes. Identifying these causes can help researchers in improving the resources their experiments use, and it can help future testbed operators determine policies that would help alleviate imbalance.

One of our observations regarding PlanetLab nodes is that newer, more powerful nodes are often very lightly used, and that older, less capable nodes are in heavy demand, which is the opposite behavior of what one would expect from researchers seeking out the most available CPU resources. We can quantify this behavior by examining the sliver counts on different nodes. Figure 16 shows the number of live slivers per node, broken down by the year the node entered PlanetLab. We see that the older nodes have more than twice as many slivers as the newer nodes, and this metric understates the difference, since these sliver counts include many of the long-running services.

The same behavior is evident if we examine the type of machine involved, since older nodes would tend to have fewer CPU cores, and newer machines would have more cores. Figure 17 plots the numbers of slivers in nodes, based on the number of CPU cores per node. The trend is clear – not only do fewer slices run on the more powerful machines, but even if we look at just the instantiated slices (the in-memory slivers), fewer of those exist on the more powerful machines. Even the most powerful nodes (labeled as "Other") are the least popular among the nodes. This breakdown is even more apparent when calculated the slivers per core, with the 8-core nodes serving only 1.53 slivers per core at a given time while single-core machines were busy with running 23.16 slivers.

We observe similar patterns in memory usage when compared to node memory size. Figure 18 shows the average memory usage of nodes, grouped by the memory size of the node. We see that nodes with more memory see relatively little extra usage of that memory. We believe that the reason that more memory is not used is because experiments that are deployed across multiple nodes have to plan for the lowest common denominator, and therefore restrict their memory usage to avoid being killed. At the same time, the reason we see any growth in memory usage on larger nodes is likely due to more memory-intensive experiments avoiding the smaller nodes, perhaps as a result of having found their

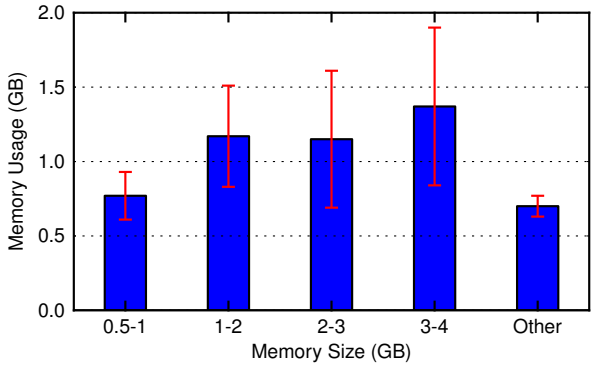


Figure 18: The memory usage by node memory size. The nodes with more memory see relatively little extra usage of that memory. The error bars represent standard deviations.

slivers killed on those nodes. Regarding live sliver counts, we see some of the same trends observed for the node's CPU cores. The nodes with the largest memory space ("Other") host only 6.64 live slivers on average while the most memory-constrained nodes ("0.5 - 1GB") are busy with running 20.14 live slivers.

Several possible explanations can explain this behavior, ranging from the history of PlanetLab to human nature. One may expect that users obtain a list of working nodes, and do not regularly update their lists, leading to a bias against newly-introduced nodes. Older nodes may also reflect more established hosting sites that joined PlanetLab earlier since they had more active network research groups. These sites may have a larger user population keeping their nodes well-maintained, and may be connected to the Internet using better-quality links. Users may also flock to more busy nodes precisely because other users have found them desirable – knowing nothing else about two nodes, the one with more active users may actually be the better node to use, because other users have already found the node to be more useful for their experiments.

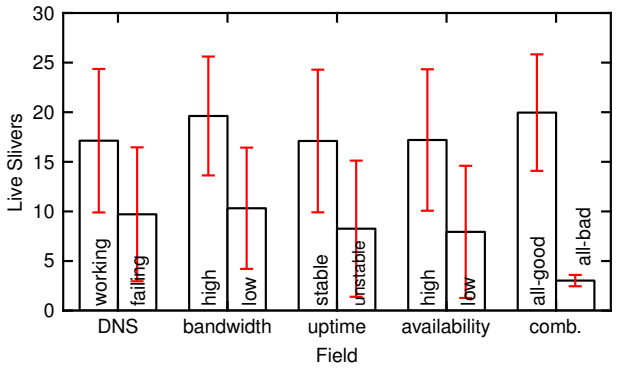


Figure 19: Popularity of nodes based on failure modes. PlanetLab users avoid nodes with high DNS failures, low bandwidth, and unstable operation. The error bars represent standard deviations.

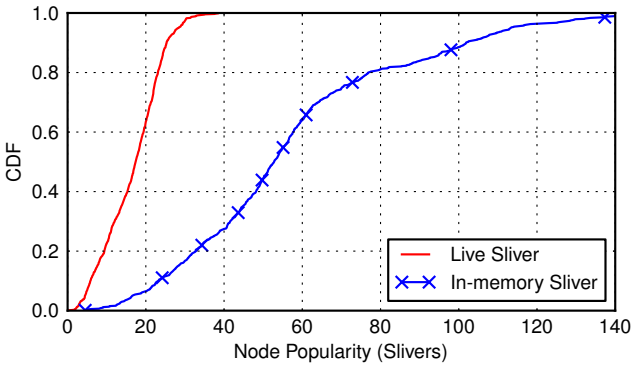


Figure 20: CDFs of node popularity. Each node's popularity is measured as its sliver count.

5.2 Nodes with Failures

While it may be argued that there is little harm letting users flock to known-good nodes, it can also be useful to explore why users avoid other nodes. From a policy perspective, the testbed itself has reasons to encourage a flatter load balance, because it can increase the capacity of the testbed, reduce experimental variance, and alleviate congestion. The testbed operators may also want to ensure that the participating sites are really contributing resources of value, instead of pro-forma resources that are useless to the rest of the testbed.

Since resource pressure on a node is not a main concern for users, we focus on the types of failures that could significantly limit network experiments on the node. CoMon records a set of metrics about node health. Among them, we selected four fields that we believe users are likely to correlate with the stability and the quality of network connections that a node provides. They are DNS failure rates, provisioned bandwidth, system uptime, and availability of each PlanetLab node.

We define several failure modes for nodes. A node is considered to have a non-working DNS system if its DNS failure rate is over 90% on average in the node. A node has low bandwidth if its 90^{th} percentile of the achieved bandwidth in the node is less than 1 Mbps. For stability, we define a node to be unstable if its average system uptime is less than a week. Lastly, a node's availability is classified as low if the node was online for less than a month in total during 2010. These choices are meant to be conservative, in that users may still prefer higher quality values than our thresholds.

We classify PlanetLab nodes based on each failure mode that we define, and compare their popularity using sliver counts as a proxy. Figure 19 presents the average number of live slivers in both failed nodes and healthy nodes. In contrast to resource availability, we find that PlanetLab users do react to the failures in the nodes. We find that healthy nodes have twice as many live slivers compared to failing nodes, and since many of these slivers are due to infrastructure services, the difference in experimental slivers is likely to be even higher. The "all-bad" nodes in the combination failures (labeled as "comb.") represent nodes that exhibit all the four failures. Those nodes had only 3 live slivers while other healthy nodes hosted 18.5 live slivers on average. Figure 20 plots the distribution of node popularity measured as live slivers per node and in-memory slivers per node. We find that the all-bad nodes are in the bottom 3% of the unpopular nodes. Roughly 65% of nodes were "all-good" nodes that do not have any failures. Those nodes serve 20 live slivers on average, which is in the 64^{th} percentile in node popularity.

5.3 Alternative Experiment Placement

In the previous sections, we showed that PlanetLab users seem to stay with the nodes they have been known to work well over time while avoiding non-working nodes for their experiments. However, this conservative placement strategy can collectively lead to inefficient resource allocation and undesirable system lags in the popular nodes. Also, the manual deployment prevents a new participant's nodes from being adopted by existing users, which can hamper the growth of the testbed in the long term.

We examine how the workload would change if researchers were to deploy their systems in a different manner. In PlanetLab, heavily loaded nodes are easily avoidable because there are available services to help users identify such nodes. CoMon provides a set of interfaces to allow users to pick lightly loaded nodes based on web-based queries. Similarly, CoMon provides interfaces to filter out failing nodes in using various metrics. Some available execution management systems [1, 2] locate resources based on high-level queries in XML given by users.

We consider a "what if" scenario that all PlanetLab users query a CoMon-like monitoring systems to find the set of lightly loaded *all-good* nodes when they deploy their experiments. We simulate the scenario using the CoMon data of 2010. We assume that a sliver is placed on the node selected by a monitoring service only when it is started, and that continuously live slivers do not migrate between nodes in our simulation.

Figure 21 plots the distribution of slivers with several alternative service-placement policies. *Original* plots the number of live slivers per core that we observed in the datasets. *BestFit* represents a policy that places a new sliver in the least loaded all-good node regardless of its location. However, it is not always desirable to deploy services in this way, because users may want to deploy their experiments at a certain range of network vantage points. *Continent* and

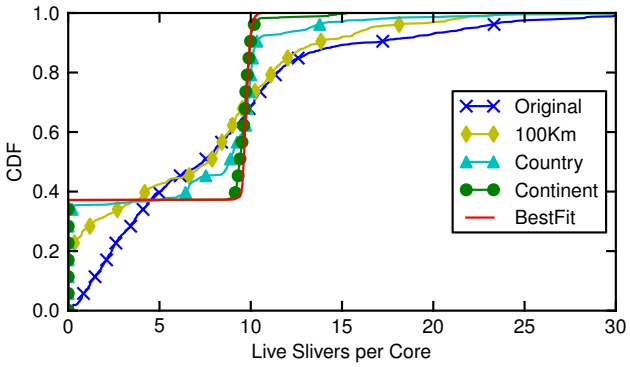


Figure 21: CDFs of live slivers per core in simulations of alternative node placement policies. Since lightly loaded and all-good nodes are selected, the workload is well balanced among nodes while any undesirable failures are avoided.

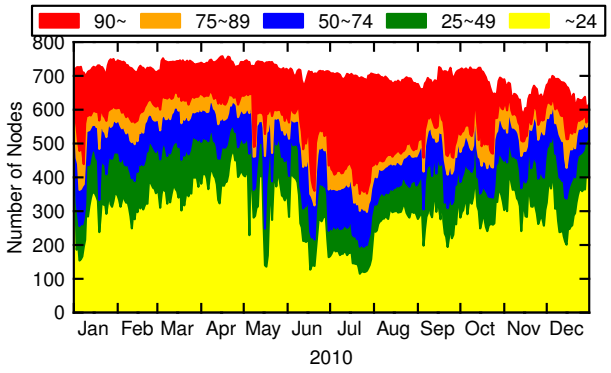


Figure 22: The distribution of per-day CPU loads on PlanetLab nodes in 2010. Each day, nodes are divided into five categories according to their per-day CPU usage. The nodes with more than 90% CPU usage account for up to 49% of all PlanetLab nodes (July 22).

Country represent policies that find nodes from the same continent or the same country as its original node in the dataset, respectively. Lastly, since some countries have a large number of nodes, we simulate a policy to find a node for a sliver within a configurable distance, 100 Kilometers, from its initial node. We compare against an ideal policy, BestFit, in which all good nodes host an equal number of slivers, and any nodes that exhibit failures (201 in total) are avoided entirely.

Our simulation result shows that the dynamic experiment placement could greatly improve the load balancing in PlanetLab while still largely avoiding problematic nodes. The 90^{th} percentile of per-node live slivers decreases from 16.1 to 13.7 (in 100 km) and 10.3 (in Country). As the placement restrictions are loosened, not only does it become possible to more evenly distribute load, but it also becomes more likely that the bad nodes are avoided entirely. The 100-km policy still allocates a substantial fraction of nodes from the bad set, whereas continent-level placement avoids them almost entirely. If a user wants to use a set of specific nodes, she should be able to deploy her service on the nodes, but many other users could benefit from the intelligent service placement. The primary reason for the improved load-balancing is that most experiments are short-lived (Section 3), so the dynamic experiment placement can help spread well the load over available nodes.

6. POLICING OF SLICES IN PLANETLAB

In this section, we show that PlanetLab's resources are affected by a few problematic slices. We examine the impact of those slices on PlanetLab and consider the implications for policing of resource usage in the testbed. We also explore how PlanetLab's workloads would change if stricter form of policing is introduced in PlanetLab.

6.1 Spin-loop Slices in PlanetLab

PlanetLab is a shared infrastructure, and while it provides some mechanisms to prevent experiments from interfering with each other, it also relies on the cooperation of researchers. For example, local tests and incremental rollouts are recommended before a new service is fully deployed in PlanetLab [29]. We observed that most slices follow this

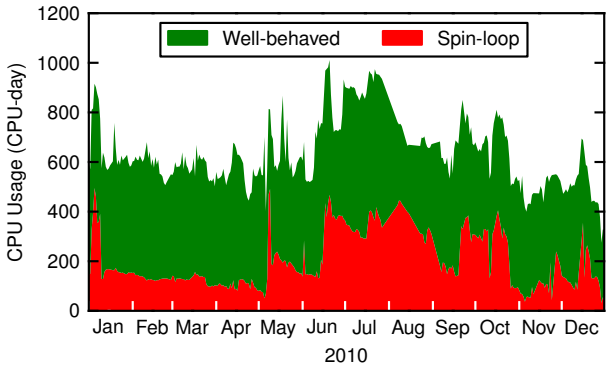


Figure 23: The CPU consumption of spin-loop slices in 2010. Although there are only a few spin-loop slices (4.8 slices among 152.7 live slices per day), the average CPU consumption of the spin-loop slices accounts for 31% of the total CPU usage of all slices.

practice well in Figure 6. To minimize centralized control, PlanetLab typically does not police resource usage of a slice unless there are significant risks of system crashes or security concerns that need to be addressed immediately.

However, some experiments may behave poorly because of design or implementation flaws, and negatively affect many other well-behaved experiments. We analyze the problem using CoMon data collected in 2010. We divide nodes into several categories according to their per-day CPU usage, and observe the distribution of the nodes over time. Figure 22 presents the distribution of per-day CPU loads on PlanetLab nodes in 2010. In this analysis, we define a node to be *overloaded* on a given date if its per-day CPU usage is over 90%, and up to 49% of all live PlanetLab nodes are overloaded. The number of overloaded nodes varies over time, but we did not find any noticeable correlation between the numbers of overloaded nodes and live slices a day (the graph is omitted for brevity). This implies that the overloaded nodes are not caused by the increase of active users.

The largest testbed-wide CPU consumers are often a few slices that use many aggregate CPU cycles without gen-

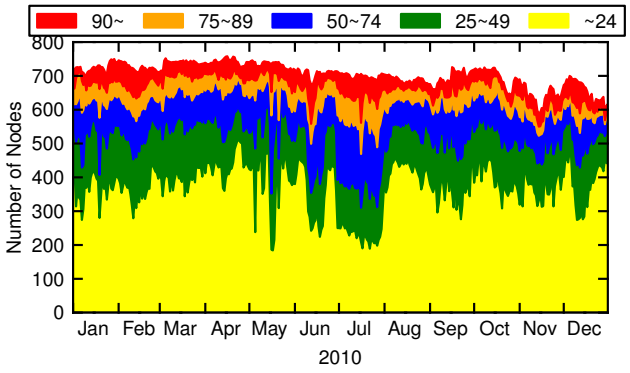


Figure 24: Time series of the updated distributions of per-day CPU loads after pruning spin-loop slices. The number of overloaded nodes is reduced by 71% (150 to 43) on average.

erating network traffic. We define a *spin-loop sliver* as a sliver that consumes more than 20% of a CPU and has an average bandwidth consumption (Tx + Rx) below a minimum value, 1 Kbps a day. We classify a slice as a *spin-loop slice* if the majority of its slivers are spin-loop slivers. Figure 23 presents the CPU consumption of the spin-loop slices in 2010. We find that the problematic slices accounted for on average 31% of the total CPU across all of PlanetLab. Since spin-loop slices average 4.8 per day among the 152.7 live slices per day, the 31% resource consumption is significantly large.

6.2 Pruning Spin-loop Slices

We simulate pruning the spin-loop slices to measure their impact on other slices. For a spin-loop slice, we subtract its sliver's CPU consumption from the node's CPU usage. If a spin-loop sliver had been pruned by PlanetLab, more CPUs might have been available to non-spinloop slivers in the node. Therefore, for each non-spinloop sliver, we select the larger of its CPU usage in a day or the median of its CPU usage throughout the year, to recompute the node's CPU usage after pruning.

Figure 24 plots the updated distribution of CPU load after spin-loop slices are pruned by PlanetLab. We find that the number of overloaded nodes is reduced from 150 to 43 nodes by policing only 4.8 spin-loop slices a day on average. It is notable that there are still some overloaded nodes even after pruning all identified spin-loop slices on PlanetLab nodes. We examine the nodes in order to understand what other factors made them remain overloaded. We find that 56% of the overloaded nodes are single core machines, which represent only 13% of all PlanetLab nodes.

7. RELATED WORK

Several resource management frameworks have been proposed for PlanetLab-like federated distributed computing infrastructures. In Sharp [12], multiple autonomous parties can exchange their resources using tickets. A ticket represents the holder's claim over a certain amount of resources in other peers, which can be issued, delegated, and redeemed in a cryptographically secure manner. Millennium [10], Mirage [9], Tycoon [16], and Bellagio [4] propose market-based mechanisms for trading resources in an economically efficient way. The systems focus on maximizing the values delivered to users by providing a means to express their valuation of resources. In Bellagio, participating users receive virtual currency budgets based on their resources that they contributed, and submit their preferences in the form of auction bids. The market-based systems are not widely deployed in PlanetLab because user valuation of resources is useful in the systems where resource demand exceeds resource supply (e.g., sensor network testbed). We expect that our analysis results will provide insights into user behavior in the federated infrastructures, which is required for designing similar economic resource allocation models.

Previous studies in traditional cluster resource management largely focused on resource utilization for compute-intensive applications in time-sharing or batch-queue systems. The job scheduling and resource allocation in the systems are designed to improve performance metrics such as throughput and mean response time. Distributed batch queue systems (Condor [18], Matchmaking [25]) provide resource sharing across loosely coupled pools of distributively owned machines. Load balancing systems (MOSIX [5], LSF [24]) balance CPU load across nodes in cluster by actively migrating processes across cluster machines. However, these cluster resource management systems do not address non-aggressive user behavior in wide-area network testbeds that are much different from compute clusters.

Our work relates to research projects that help PlanetLab users monitor and locate resources in the testbed. SWORD [2] is a resource discovery service deployed in PlanetLab. In SWORD, users describe desired resources such as per-node characteristics in XML and submit the queries, and then the service locates an appropriate set of resources for the user based on the given specification. CoMon [20] provides a comprehensive view of statistics about every node and slice in PlanetLab. It also provides a mechanism to select nodes based on queries provided by users. Also, several execution management systems [1, 3, 7] are available to provide GUI interfaces to help users deploy and monitor their systems across multiple remote nodes.

8. SUMMARY

In this work, we conduct an extensive and in-depth analysis of six years of PlanetLab measurements to understand and characterize PlanetLab's resource usage. Based on our analysis, we discuss the implications for designing resource management policies in new federated network testbeds. We find that the usage is much different from shared compute clusters, that conventional wisdom does not hold for PlanetLab, and that several properties of PlanetLab as a network testbed are largely responsible for this difference. We find that experiments typically utilize PlanetLab to expand their network reach, and that this metric of utility is a far better indicator of PlanetLab's effectiveness than compute-oriented metrics like CPU utilization.

We also find that approaches focusing on compute-oriented metrics are likely to be inapplicable to PlanetLab-like workloads. In particular, we find that resource usage is very bursty, and that the vast majority of experiments consume very little resources over much of their lifetimes. Based on the measurement of total resource usage, we explore the effectiveness of several resource allocation systems, and find that both pair-wise bartering and centralized banking systems can address only a small percentage of total resource

usage. This result implies that resource management systems in federated network testbeds still need policies to fairly allocate available resources for the vast majority of the workload. We examine some policies for better resource discovery and, node management, and pruning of runaway experiments.

9. ACKNOWLEDGMENTS

We would like to thank our shepherd, Olaf Maennel, as well as the anonymous IMC reviewers. We also thank KyoungSoo Park, Michael Golightly, and Sunghwan Ihm for several beneficial discussions and comments on earlier drafts of this paper. This research was partially supported by the NSF Awards CNS-0615237 and CNS-0916204.

10. REFERENCES

[1] J. R. Albrecht, R. Braud, D. Dao, N. Topilski, C. Tuttle, A. C. Snoeren, and A. Vahdat. Remote control: Distributed application configuration, management, and visualization with Plush. In *Proceedings of USENIX Large Installation System Administration Conference (LISA)*, 2007.

[2] J. R. Albrecht, D. L. Oppenheimer, A. Vahdat, and D. A. Patterson. Design and implementation trade-offs for wide-area resource discovery. *ACM Transactions on Internet Technology (TOIT)*, 8(4), Sept. 2008.

[3] AppManager. http://appmanager.berkeley.intel-research.net/.

[4] A. AuYoung, B. N. Chun, A. C. Snoeren, and A. Vahdat. Resource allocation in federated distributed computing infrastructures. In *Workshop on Operating System and Architectural Support for the Ondemand IT Infrastructure (OASIS)*, 2004.

[5] A. Barak and O. Laadan. The MOSIX multicomputer operating system for high performance cluster computing. *Future Generation Computer Systems*, 13(4-5), Mar. 1998.

[6] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating system support for planetary-scale network services. In *Proceedings of USENIX NSDI*, 2004.

[7] J. Cappos, S. Baker, J. Plichta, D. Nyugen, J. Hardies, M. Borgard, J. Johnston, and J. H. Hartman. Stork: Package management for distributed VM environments. In *Proceedings of USENIX Large Installation System Administration Conference (LISA)*, 2007.

[8] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson. Seattle: A platform for educational cloud computing. In *Proceedings of ACM Technical Symposium on Computer Science Education*, 2009.

[9] B. N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat. Mirage: A microeconomic resource allocation system for sensornet testbeds. In *Proceedings of IEEE Workshop on Embedded Networked Sensors*, 2005.

[10] B. N. Chun and D. E. Culler. User-centric performance analysis of market-based cluster batch schedulers. In *Proceedings of IEEE International Symposium on Cluster Computing and the Grid*, 2002.

[11] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *Proceedings of USENIX NSDI*, 2004.

[12] Y. Fu, J. S. Chase, B. N. Chun, S. Schwab, and A. Vahdat. SHARP: an architecture for secure resource peering. In *Proceedings of ACM SOSP*, 2003.

[13] GENI Aggregate Manager API. http://groups.geni.net/geni/wiki/GeniApi/.

[14] GENI: Global Environment for Network Innovations. http://www.geni.net/.

[15] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-preserving P2P data sharing with OneSwarm. In *Proceedings of ACM SIGCOMM*, 2010.

[16] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. A. Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent and Grid Systems*, 1(3), Aug. 2005.

[17] S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca. Measuring bandwidth between PlanetLab nodes. In *Proceedings of Passive and Active Measurement Conference (PAM)*, 2005.

[18] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor - a hunter of idle workstations. In *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, 1988.

[19] D. Oppenheimer, B. Chun, D. Patterson, A. C. Snoeren, and A. Vahdat. Service placement in a shared wide-area platform. In *Proceedings of USENIX Annual Technical Conference*, 2006.

[20] K. Park and V. S. Pai. CoMon: A mostly-scalable monitoring system for PlanetLab. *ACM SIGOPS Operating Systems Review*, 40(1), Jan. 2006.

[21] K. Park and V. S. Pai. Scale and performance in the CoBlitz large-file distribution service. In *Proceedings of USENIX NSDI*, 2006.

[22] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir. Experiences building PlanetLab. In *Proceedings of USENIX OSDI*, 2006.

[23] PlanetLab. http://www.planet-lab.org/.

[24] Platform Computing Load Sharing Facility (LSF). http://www.platform.com/.

[25] R. Raman, M. Livny, and M. H. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *Proceedings of IEEE International Symposium on High Performance Distributed Computing (HPDC)*, 1998.

[26] SETI@home. http://setiathome.berkeley.edu/.

[27] Sirius: A Calendar Service for PlanetLab. https://snowball.cs.uga.edu/~dkl/pslogin.php/.

[28] SliceStat: Slice monitoring sensor on PlanetLab. http://codeen.cs.princeton.edu/slicestat/.

[29] N. Spring, L. Peterson, A. Bavier, and V. Pai. Using PlanetLab for network research: Myths, realities, and best practices. *ACM SIGOPS Operating Systems Review*, 40(1), Jan. 2006.

[30] L. Wang, K. Park, R. Pang, V. S. Pai, and L. Peterson. Reliability and security in the CoDeeN content distribution network. In *Proceedings of USENIX Annual Technical Conference*, 2004.

Summary Review Documentation for

"Understanding and Characterizing PlanetLab Resource Usage for Federated Network Testbeds"

Authors: W. Kim, A. Roopakalu, K. Li, V. Pai

Reviewer #1

Strengths: This is a very unique, extensive, and interesting dataset. PlanetLab as a platform has been a tremendous boost to networking/measurement research and a retrospective measurement study of usage there is very relevant. The problem of how a federated test bed will be used is important and timely given the interest in deploying more of these.

Weaknesses: I do not see any strong weaknesses in terms of techniques/results, other than the description of some of the results can be improved and I would like to see some more in-depth analysis of a few more interesting aspects of the problem instead of too many figures making the same, somewhat obvious observations.

Comments to Authors: In the introduction, I would tone down some of the claims about "conventional wisdom"/"tragedy of commons" etc, because it seems fairly obvious that the workloads for PlanetLab are very different from traditional compute clusters!

I found Section 3.2 very confusing. Maybe this is a terminology/writing problem – you have "slice" but really you are doing a "sliver"/node level analysis here. The writing goes back and forth and its confusing me a bit. Also, what is the "average" being computed over - across time for each sliver or across nodes?
Would it also be useful to just look at average over "active periods" instead of the overall average?

In Section 3.2, how you generate this visualization is very confusing.

I agree with the observation made in Section 3.4, but I feel that you are overselling the analysis by saying that this flies in the face of conventional wisdom etc.

When you calculate the timeseries of bank/barter/slop etc. – are these based on "instantaneous" measurements or over a sliding window or cumulative over history?

Fig 14 seems to suggest that most sites are incredibly altruistic – i.e. that they have a huge balance that they have not claimed. I wonder if you can correlate this with papers/projects that the sites have generated that use PlanetLab to verify.

In Section 5.2, I found your choice "memory" imbalance very puzzling. I imagine that most PlanetLab workloads are not really memory bound, so it's not terribly surprising that there is a memory imbalance. I would much rather see the corresponding result for network resources or CPU.

I agree with the overall notion of dynamic experiment placement; an obvious concern is w.r.t repeatability/correctness when you end up choosing different nodes over different runs of the same experiment/measurement. E.g., you could see network latency/congestion effects from different nodes being different. In fact, I am guessing this could be one motivation for manually choosing nodes. Could you comment on this, or try a dynamic-but-stable strategy, where the system also tries to find sites that overlap with the prior choice?

Also, I have heard and seen anecdotal evidence that people manually choose nodes because many nodes are inherently "unreliable". Can you comment on this?

You have an amazing dataset! I really hope that this dataset will be made public. I would try to cut some of the figures that are making more obvious points to try and fit in other aspect. Here are some potential directions to increase the depth of analysis that I could think of:

1. I wonder if you could characterize the slices into common buckets of "operating models". The goal is that if we have a few standard "configuration modes" and an interface for choosing these, then researchers could choose from these typical use cases for an easy to use dynamic resource allocation strategy.

2. There are two dimensions to your analysis space: Space (nodes) and Time. Would it help to try different ways of combining these — you have chosen specific points to e.g., average over time or max over time of sum over nodes etc. For example, it could be useful to look max over time max over nodes? I.e., when a slice is active does it hog a particular node?

3. Are there "runaway" experiments or slices that people have forgotten to turn off long after a deadline?

Reviewer #2

Strengths: PlanetLab is widely used. For every researcher working on PlanetLab there is a need to know what bias the system introduces. The paper looks at CPU and memory workloads.

Weaknesses: The study is more a sophisticated sysadmin report for the PlanetLab central administration, than a measurement paper for the research community.

Comments to Authors: In the current form the paper is written more from a system administration perspective and not for a user point of view. The authors study mainly CPU-days and Memory-days of users using the system. The authors warn several times of measurement errors that could be created by an overloaded system. For example, on page 10, the authors say: "This latency may degrade responsiveness of network services or add measurement noise to running experiments".
Admittedly, it is a very difficult task to estimate any boundaries on network performance errors that might be created due to an overloaded system, but it would have been nice to see some form of quantification.

It is certainly up to the experimenter to critically review the measurement results and understand the bias introduced by the system, but for a research paper it would have been nice to give at least an anecdotal flavor for what the upper bounds might be.

Reviewer #3

Strengths: Very cool data set, and an interesting topic. Clearly future testbeds can benefit from improved understanding of how PlanetLab has been put to use and working out over the past years.

Weaknesses: I find the authors discuss basic findings in too much depth, from too many angles, for too long, with too many plots, while I have a hard time following some of the more interesting analyses because of their resulting terseness.

Comments to Authors: My main problem with this paper is that I find some elementary findings eating too much space at the cost of interesting things that ended up overly terse. For example, one can sum up virtually all of Section 3 by the 80/20 rule - a small number of heavy users, along all dimensions, while everything else is intermittent. Do you really need 4 pages for that? Most of the paper is similarly one-dimensional.

When it matters, I found things brief and superficial. Your analysis of barter and banking systems for resource management, while interesting, feels quite simplistic. I feel that instead of adding a plot for virtually every paragraph of text, the paper would be a more productive experience for the reader if you had left out a good handful of plots and instead integrated analyses that go a bit beyond your dataset.

I disagree with your suggested expectation of a Tragedy of the Commons in PlanetLab. That model simply does not apply. General lack of resource-hogging is very much what I expected to see, simply from PlanetLab's intended purpose.

It would really help if instead of the overlong Sections 3 and 4 you put in a little more background on PlanetLab. If it is clear from the outset that experimenters pick nodes manually, you are eliminating a lot of the guesswork.

I find Figures 11 and 12 unsurprising. It seems unreasonable to expect a single site to contribute equal amounts of resources as it consumes in the entire remaining network, so clearly the slop fraction is highest. Surely you could come up with better banking schemes that factor this imbalance into the budget earning/spending calculation?

I do not understand your definition of bank. A provides resources in its nodes, and consumes resources elsewhere. Why is "bank" the minimum of the two, and not the difference? Any normal balance can be positive or negative. Did you just pick really unfortunate terminology?

I don't understand Figure 13. So PlanetLab's available CPU capacity was around 1,800 CPU-days per day, throughout. You also have the CPU consumptions accumulating at each node over the course of a month. I see no explanation on how the limits you mention would result in the curves you show. Is this simulation, as you are mentioning it in the next paragraph? I find the rest of that section too compressed to follow.

In Section 6.2 you refer back to Section 3, stating that you discussed how experiments follow the guideline of testing in-the-small and expanding gradually for d-day. You do not actually show that; you say it is true for one slice doing IPOP experiments (in Sec. 3.3).

Figure 23 is not surprising. If a program spins instead of blocking on I/O or otherwise behaving "nicely", then obviously its CPU contribution skyrockets.

It would have been nice to see some orthogonal analyses. For example, do people signing up actually run experiments? How often do they succeed in deploying something long-running? How often was that actually the goal? PlanetLab requires experimental descriptions for the studies it hosts. Does this fit with high CPU consumption at all? I.e., do they suggest obvious bugs, or are they openly declared CPU crunching?

Reviewer #4

Strengths: Fantastic dataset; thorough analysis of resource usage and slice classes; resource management in federated distributed testbeds is an important and persistent problem.

Weaknesses: Nothing obvious.

Comments to Authors: This is a great piece with many interesting findings on the use of PlanetLab and the effectiveness of policies and potential algorithms. There are a few issues with the presentation here and there, but all of them could be addressed in a camera-ready version.

It is surprising not to see references to all the work done in cluster-based and grid systems in you related work section (and only one in your overall reference list); particularly given your comments about how this is different from what you see in PlanetLab. A quick web search points to Hart's paper on measuring TeraGrid, which would seem appropriate even if to say how they were different. What is the length of the measurement period (less than 5', I know, but what). Is 0.1% CPU utilization a magic number? Seems alright to me but I tend to dislike these sort of constants.

There seem to be a neat shift in CPU usage over the three years and I wonder if there is a learning process there.

Fig 5 took me a while partially due to the short and unclear discussion and the heavy reliance on color. The reading of slices with more than 200 nodes helps a bit but it may be better to pick another point as well, perhaps the 200 slices that use over 200 nodes 50% of the time (or something like that).

"fly in the face of the common expectations" seems to be a bit too strong; if you are going to make this claim I think you should have references to work arguing this. It is either that or you soften your lines. And you probably want to speak 'of common expectations' and not 'of the common expectations'.

I agree with the reading in p. 8-9 about the effectiveness of different resource management systems. It is surprising and a bit disappointing, however, that a service aimed to the whole research community seems to be driven by a small set of synchronized deadlines.

I really like the analysis of Sec. 5.2, particularly the simulation-based study of alternative placement policies.

The paper seems to need a summary of findings as a way of closing a long list of interesting findings. For instance, I would like to see there a discussion of how your observations can be generalize to other federated systems and what would be the potential issues with those generalizations.

Reviewer #5

Strengths: Extensive characterization of resource usage based on long term and detailed measurement.

Weaknesses: Implications of some of the findings are not clear.

Comments to Authors: The paper examines a wide range of characteristics of resource usage. Overall, this paper is well organized and well written. The main issue for this reviewer is the limited discussion on the implications of some of the findings.

Response from the Authors

Most of the comments that we received from the reviewers were questions about our measurement results presented in the paper. We revised our descriptions of the results to answer those questions and make them clearer. To address the comment "…anecdotal evidence that people manually choose nodes because many nodes are inherently unreliable.", we added a new result in Section 5 to show that PlanetLab users seemingly avoid nodes in exhibiting various failure modes. At the same time, we also demonstrated other biases that may not be entirely appropriate. For a deeper analysis of bartering/banking systems, we extended the analysis by adding resource routing to bartering system. We discussed the effectiveness of the resource routing in PlanetLab by comparing the enhanced bartering with simple pair-wise resource exchange in Section 4. Besides resource consumption, we analyzed each site's usage of network reach in terms of remote sites and continents. The added result helps better explain the usage patterns seen and the reason why a tragedy of commons is not visible in PlanetLab.