HiDRA: Statistical Multi-dimensional Resource Discovery for Large-scale Systems

Michael Cardosa and Abhishek Chandra
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455
{cardosa, chandra}@cs.umn.edu

***Abstract*—Resource discovery enables applications deployed in heterogeneous large-scale distributed systems to find resources that meet QoS requirements. In particular, most applications need resource requirements to be satisfied *simultaneously* for multiple resources (such as CPU, memory and network bandwidth). Due to dynamism in many large-scale systems, providing statistical guarantees on such requirements is important to avoid application failures and overheads. However, existing techniques either provide guarantees only for individual resources, or take a static or memoryless approach along multiple dimensions. We present *HiDRA*, a scalable resource discovery technique providing statistical guarantees for resource requirements spanning multiple dimensions simultaneously. Through trace analysis and a 307-node PlanetLab implementation, we show that HiDRA, while using over 1,400 times less data, performs nearly as well as a fully-informed algorithm, showing better precision and having recall within 3%. We demonstrate that HiDRA is a feasible, low-overhead approach to statistical resource discovery in a distributed system.**

I. Introduction

Recent years have seen the increasing use of large-scale distributed systems such as open Grids [1], [2], distributed Clouds [3], and peer-to-peer systems [4], [5] for a wide range of applications such as scientific computing, file sharing and multimedia streaming. Most of these distributed platforms consist of large number of machines with heterogeneous resources, and many of them are also geographically distributed. In order to satisfy application QoS requirements such as throughput, latency, and jitter, *resource discovery* [6], [7], [5], [8] is often used in such systems to find suitable nodes for deploying application components, or to execute specific computational tasks.

Most distributed applications rely on multiple interacting resources, such as processing, memory, and network bandwidth, in order to meet their execution requirements. For instance, a data-intensive bioinformatics application [9] might be running multiple computational tasks on different machines, each analyzing gene sequences. In this scenario, each computational task needs sufficient CPU capacity and memory for each individual task to run efficiently. If each node has enough CPU capacity but insufficient memory, then the tasks may slow down considerably due to excessive swapping and disk accesses. In addition, these tasks may also need a minimum amount of network bandwidth for downloading the required genome data, and exchanging and communicating partial results. Similarly, a node participating in a peer-to-peer streaming application would require sufficient downstream bandwidth to download the required video frames along with enough buffer space and CPU capacity for storing and decoding the frames. At the same time, it should also have sufficient upstream bandwidth to share its frames with other peers in the system. In other words, most applications need to satisfy multiple resource requirements to avoid any single resource from becoming a bottleneck and affecting the performance of the application.

Many existing resource discovery mechanisms [6], [10], [7] have incorporated the need for multiple resources. However, many of these mechanisms either satisfy statically defined requirements, such as a node's physical CPU clock speed or total amount of RAM, or at best, provide information about the recent values affecting some of these attributes, such as CPU load. However, such static or limited resource capacity information is insufficient for most large-scale platforms. Many of these systems have been shown to exhibit a large degree of dynamism [11], [12] in terms of the effective resource capacity they can provide at any given time. This dynamism is caused by several factors, such as varying loads, network congestion, churn, or varying application demands. To take such dynamic resource capacities into account, a recent approach has focused on providing *statistical guarantees* on meeting resource requirements [8]. However, this approach focuses on individual resource requirements, and is insufficient for providing statistical guarantees for multiple resources *simultaneously.*

In this paper, we present a new technique for statistical multi-dimensional resource discovery called HiDRA (High-Dimensional Resource Allocation). This technique is designed to find nodes with desired resource capacities along multiple dimensions, and provide statistical guarantees on these capacities being satisfied over a time interval. An important aspect of this technique is that it can provide such guarantees on *any combination of a number of resources* (e.g., only CPU and memory, or CPU, memory and network bandwidth, etc.), and further, it provides guarantees *for all the resource requirements being satisfied simultaneously.* A key contribution of this work is the novel use of *multivariate normal distributions* for the probabilistic modeling of resource capacity over multiple dimensions. We provide a heuristic for converting general distributions (observed in real node data) to this representation, to

This work was supported in part by NSF Grant CNS-0643505.

provide high accuracy for common resource discovery queries. We conduct a data analysis of a month-long PlanetLab trace, and our results show that HiDRA performs nearly as well as a fully-informed algorithm, showing better precision and having recall within 3% of this algorithm. We have deployed HiDRA on a 307-machine PlanetLab testbed, and our live experiments on this testbed demonstrate that HiDRA is a feasible, low-overhead approach to statistical multi-dimensional resource discovery in a distributed system.

II. Background & System Model

Resource Discovery: Previous scalable approaches to resource discovery have taken place in a static-configuration context; that is, searching for nodes in a wide-area system such that hardware and software configurations meet specified requirements. Since these components are quite static, there is no need to update this information on any regular basis and therefore the use of content-addressable networks [6] has been used for such matchmaking of applications with nodes in a distributed and scalable fashion.

The focus of resource discovery techniques upon dynamic node-level characteristics [7] (e.g., CPU load, memory available, network bandwidth) shared many similarities with the static-configuration approaches. In SWORD [7], a multi-attribute range query system [13] implemented over a DHT was used to store and index load values for several node-level resource metrics. However, due to scalability concerns and the inability of DHTs to store or index distributions, only one load value per metric per node was maintained. Due to high variability in distributed systems such as PlanetLab [11], such a memoryless approach is unable to provide node-level resource capacity guarantees to the application. Recent work has extended SWORD to address data staleness issues [14], but the resource discovery method still lacks statistical properties.

Resource Bundles [8] introduced statistical guarantees for resource discovery for single node-level resource metrics. Historical resource usage measurements were presented as profiles in the form of histograms. For scalability in large-scale systems, aggregation was used in a hierarchical overlay topology wherein nodes with similar resource usage profiles were grouped together to form higher-level representatives. This accurately provided statistical guarantees for resource discovery, but only for one resource metric at a time (e.g. *effective* CPU capacity available).

Other Related Work: Condor [15] used a centralized coordinator approach to finding idle workstations in grid environments among machines under the same administrative domain. Cluster computing on the fly [5] is a decentralized cycle-sharing approach to resource discovery in peer-to-peer environments. This work is single-metric, as cycles were the primary resource sought after; no notion of resource usage profiles were used. Their goal was to find idle machines near the edge of the network.

Network Weather Service [16] uses tournament predictors to accurately predict trends in resource usage levels. However, these predictions are limited to the next time instant, and our focus is on providing long-range statistical guarantees.

SDIMS [17] and Astrolabe [18] provide distributed "control planes" as a monitoring backbones for large-scale distributed services. Such a distributed overlay can be useful for disseminating resource usage information in a distributed resource discovery framework.

System Model: We assume our system is a large-scale, possibly wide-area or planetary-scale system. Participant nodes may be geographically distributed and could span multiple administrative domains. Nodes are able to monitor their own resource usages and capacities over time. We assume the nodes are connected via some interconnection overlay which they use to disseminate and share their resource usage information. The focus of our paper is on the scalability of data representation in the system, and we make no assumptions about the type or structure of the overlay or a specific dissemination technique. Examples of data dissemination techniques that could be employed include gossiping and epidemic protocols.

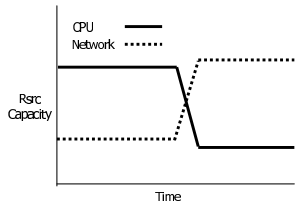
III. Statistical Multi-dimensional Resource Discovery

A. Statistical Resource Requirements

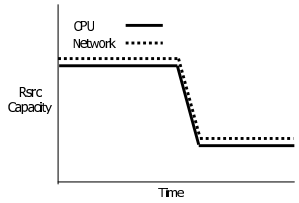
Existing multi-dimensional resource discovery techniques represent resource requirements as a tuple $\{[R_1, \ldots, R_m], [c_1, \ldots, c_m]\}$ for m resources, such that each resource type R_i satisfies a capacity value c_i. For instance, an application might need n nodes with $\{$CPU $\geq$ 1 GHz, RAM $\geq$ 1 GB, network b/w $\geq$ 1 Mbps$\}$. This requirement can then be specified as $\{$[CPU, RAM, network], [1GHz, 1GB, 1Mbps]$\}$. As discussed in the previous section, these capacity requirements can either be static values [6] or recent values [7]. However, due to dynamic variations in resource availability, for instance, due to load variations, failures, and competing applications, a statistical guarantee is desirable on these resource requirements.

Statistical resource requirements for a single resource have been defined [8] as a tuple $\{R, c, p, t\}$, where R is a resource type, c refers to a capacity level, p is a percentile value, and t is a time duration. This definition implies that a resource requirement can be specified as a resource type (e.g., CPU) satisfying an *effective capacity* c (e.g., 1 GHz) for at least p% (e.g., 95%) of a time duration t (e.g., 24 hrs). We extend this definition to incorporate multiple resource requirements, by allowing R and c to be vectors $[R_1, \ldots, R_m]$ and $[c_1, \ldots, c_m]$ for m resources, such that each resource type R_i satisfies a capacity requirement c_i, *and all these requirements are satisfied simultaneously* at least p% of the time[1]. Thus, for instance, the example requirement above can be specified as $\{$[CPU, RAM, network], [1GHz, 1GB, 1Mbps], 95%$\}$, where each capacity requirement corresponds to the *effective* capacity

[1]We omit the time duration t for clarity of discussion; one could easily extend our techniques for each time period t of interest.



(a) Negative correlation between CPU and Network usages



(b) Positive correlation between CPU and Network usages

Fig. 1. Two time series showing the same resource usage profiles for individual resources, but with substantially different correlations between the resources.

of the corresponding resource, and all these requirements should be satisfied *simultaneously*[2].

B. Resource Usage Representation

Based on the above definition of statistical resource requirements, the resource discovery process then involves finding nodes based on the following criterion: Given a requirement $\{R, c, p\}$, which nodes satisfy $\forall i(R_i \geq c_i)$ (vector comparison)[3] at least p% of time. A key issue here is how to represent the resource usage information of multiple resources to enable such a query to be resolved easily. Let us begin by considering a few possible approaches, based on existing techniques and their extensions.

One possible approach would be to use a key-based representation as used by several existing DHT-based resource discovery algorithms [6], [7]. These algorithms represent resource capacity/usage information of individual nodes as DHT keys, where, each key incorporates information about each resource type as well as its capacity value. However, while such a representation is effective for representing points in a multi-dimensional space (e.g., static values or recent values), it is not suitable for representing distributions or for range queries (as required for statistical inference).

Another possible approach could be to apply a statistical resource discovery technique such as Resource Bundles [8] to individual metrics and then to combine individual resource-level guarantees to derive multi-dimensional guarantees. For example, if a multi-dimensional requirement is {[CPU, network], [1 GHz, 10 Mbps], 90%}, then a node that can sustain 1 GHz effective CPU capacity for 90% of the time as well as 10 Mbps effective network bandwidth capacity for 90% of the time would be expected to satisfy the given requirement.

[2]To capture the notion of *meeting resource requirements for multiple metrics simultaneously*, we will use the terms *multi-dimensional*, *multi-metric*, and *multi-resource* interchangeably in the rest of our paper.

[3]We allow both $\geq$ and $\leq$ comparisons, but mention only one for clarity of discussion.

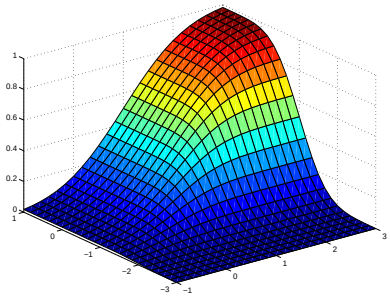


Fig. 2. MVN combines multiple normal distributions together with varying μ, σ parameters and correlations between dimensions in the Σ matrix. Shown is an example 2-dimensional requirement space.

However, this approach does not capture the *simultaneous* occurrence of the two requirements, which depends on the correlation between the two resource usages. Figure 1 illustrates this time-dependence of the resource usage behavior of multiple resources. Therefore in order to derive multi-dimensional resource requirement guarantees directly from single-metric guarantees, all involved dimensions would either need to be completely correlated or completely independent of each other; in practice, neither of these conditions occur. For instance, in our study of PlanetLab traces, correlations between CPU and network bandwidth metrics were observed to vary between $r = 0.13$ and $r = 0.96$.

C. Modeling Multiple Dimensions

Based on our discussion above, the question is how can we model individual resource profiles accurately while also capturing the inter-resource correlations. In order to achieve these goals, we have the following key requirements for modeling resource capacity information:

- A compact representation of the resource usage data over multiple resource types for each node.
- An efficient means for characterizing the inter-resource correlations at each node.
- A simple way to map the representation to the statistical requirement for resource discovery.

Histograms or probability distributions can provide compact representations of individual dimensions, and are also easy to map to individual statistical requirements (by finding the corresponding percentile values). However, as discussed above, they lose the time-dependent correlation among different dimensions. One possibility could be to maintain histograms for individual dimensions along with inter-dimension cross-correlation values. However, as discussed above, there is no straightforward way of combining individual histograms with the correlation values to determine the corresponding percentile values across multiple dimensions (See Figure 1).

A statistical distribution that satisfies the requirements mentioned above is the Multivariate Normal (MVN) Distribution. This distribution is a generalization of the normal distribution to n dimensions. It can be represented as a set of n normal distributions $N(\mu_i, \sigma_i)$, $i = 1, \ldots n$, each distribution

corresponding to one dimension, and an $n \times n$ covariance matrix Σ capturing the correlation among the different dimensions. The MVN distribution has the advantage of being a statistically "smooth" approach to finding the "volume" of an n-dimensional requirement plane. An example of a multi-dimensional requirement space generated by MVN is shown in Figure 2. Given desired ranges of values for each of the n normally distributed variables, and the covariances between each of the n variables, we can calculate the probability of these ranges co-occurring together. An MVN distribution meets each of the above-mentioned requirements as follows:

- It provides a compact representation of individual resource usage distributions: each distribution can essentially be represented by two numbers: (μ, σ), the mean and standard deviation of the normal distribution.
- The covariance matrix captures the time-dependent correlation between different dimensions.
- We can compute the probability of a multi-dimensional requirement being met by evaluating the parameterized MVN over the desired ranges along each of the resource dimensions.

MVN would be a suitable representation to use if the actually observed resource usage distributions could be accurately modeled as MVN distributions. However, we cannot assume resource capacities to be normally distributed[4]. The question is whether we can somehow exploit the nice properties of MVN for solving our problem, while accounting for the real resource usage behavior.

D. The Critical Region: Normal Approximation of Usage Profiles

To solve the above problem, we observe that given a resource capacity distribution, there is likely to be only a small region of interest for resource discovery purposes. For example, an application is unlikely to request an *effective CPU* rate of 2 GHz for only 50% of the time. Instead, any statistical guarantees are likely to lie in a high percentile range (such as 90-99 percentile). Therefore, since the *critical values* of the resource capacity distribution lie near the tail of the distribution, then a fair approximation would be to fit a normal distribution to more accurately capture that *critical region* of interest.

Therefore, under the assumption of such a critical region of interest, we approximate a resource usage profile to the normal distribution. As seen in Figure 3, only two points in the resource usage profile are necessary in order to provide a normal approximation. We define these points to be the left and right boundaries of the critical region of the resource usage profile (e.g. 99th and 90th percentiles, respectively). Note that this normal curve need not be accurate for the shape of the distribution outside the critical region, but should capture the critical region with high accuracy. In addition, to capture the correlations among the different dimensions, we simply use the covariance matrix for the original distributions. This set of normal distributions along each dimension coupled with the covariance matrix provides us with an MVN representation of the resource usage profiles.

[4]For instance, for PlanetLab data, we found that most resource usage distributions were *not* normally distributed.

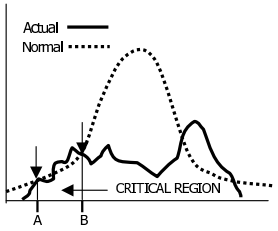


Fig. 3. Construction of an approximation of a resource capacity profile (Actual) by a normal distribution (Normal), by defining the *critical region* between points A and B.

The question is whether these normal approximations are adequate representations of their respective resource usage profiles? Furthermore, are the combination of these approximations into the MVN distribution sufficient for accurate approximations of multi-dimensional resource requirements? We show empirically in Section IV-B that these approximations of single-metric resource usage profiles to normal distributions are, in fact, highly accurate approximations for their respective critical regions. Next, we show how the MVN model can be used for resource discovery.

E. Applying MVN to Multi-dimensional Resource Discovery

An MVN-based multi-dimensional resource discovery technique for n resource metrics, takes as its input an MVN model of n resource usage profiles in the form of normal distributions $\mu = (\mu_1, \mu_2, ..., \mu_n)$, $\sigma = (\sigma_1, \sigma_2, ..., \sigma_n)$, and an $n \times n$ covariance matrix Σ capturing the correlations between resources. Also, it accepts a range of desired values for each metric. For example: for effective CPU available, [1.5 GHz, ∞], and for observed (node-level) network transmission rates, [$-\infty$, 1 Mbps]. Notice we must use $-\infty$ instead of zero (even for nonnegative metrics) due to the use of normal approximations.

As its output, this resource discovery technique uses the MVN distribution CDF to compute the *probability that all of the resource usage variables will fall within their respective given ranges simultaneously*. Thus, if the resulting probability is greater than or equal to the desired guaranteed level of service for the given requirement, then that node will be deemed *suitable* for the application.

At a high level, each node in the system would analyze its own traces to determine the correlation values between all resource metrics, and perform approximations to the normal distribution for all such metrics. In order to represent its aggregate multi-dimensional resource requirement capacities, only the normal distribution parameters (μ_i, σ_i) and covariance matrix (Σ) need to be maintained, and propagated to other nodes in the system as required (e.g., a central manager, or neighbors in an overlay). Overlays appropriate for this such data dissemination for resource discovery are presented in

our previous work [8] which makes use of a hierarchical overlay [17].

IV. Evaluation

We next present an evaluation of *HiDRA*: our MVN distribution-based resource discovery technique. We carry out this evaluation using a data analysis of PlanetLab traces, as well as through a live deployment on PlanetLab. As part of the evaluation, we first validate the accuracy of approximating individual resource usage distributions using normal distributions. Then we evaluate the accuracy of HiDRA for multi-dimensional resource discovery. Finally, we evaluate the performance of this technique through a live deployment on PlanetLab. We begin by describing our data analysis methodology for evaluating resource discovery techniques.

A. Data Analysis Methodology

We used a month-long PlanetLab trace of 427 nodes obtained by CoMon [19] from February 2007 for our experiments. This trace provided resource usage values at 5 minute intervals for various resources for each node: CPU, memory, network bandwidth, etc. In particular, we considered resource usage metrics such as *effective CPU* (calculated from the *CPU Burp* statistic included in CoMon data), observed network transmission rate (NetTx), observed receive rate (NetRx)[5] and 5-minute load average (5LoadAvg). We did not consider memory usage, as PlanetLab has a stringent memory usage policy, which renders memory usage data useless for our purposes.

Emulating Resource Discovery: In order to evaluate resource discovery techniques through data analysis, we emulate the resource discovery process as follows. This process from the client-side perspective begins as a query (specifying a requirement), receives an answer from the resource discovery algorithm (consisting of a selection of nodes on which to deploy), and then ends by evaluating the goodness of the node selections, determining which nodes were *acceptable*, i.e., which nodes satisfied the requirement over the lifetime of the deployment.

We execute each resource discovery algorithm over a 24 hour trace of data for making its decisions (for node-level statistical guarantees over the next 24 hours), and then evaluate the goodness of the selection by observing the resource usage data for the selected nodes for the following 24 hours. We perform this evaluation for each successive day in the month-long trace. Each baseline algorithm (described next) may use all or part of the 24 hours of data to make its decisions. The MVN distribution-based algorithm (HiDRA) uses the 24 hours of data to construct single-metric normal approximations and cross-metric correlations which are then used for multi-metric approximations.

Evaluation Metrics: The goodness of choice between different resource discovery algorithms can be evaluated in several ways. Resource discovery algorithms return a set of nodes to the client application, indicating their best efforts at finding the most accurate set of acceptable nodes. To quantify the accuracy of resource discovery algorithms, we define the following evaluation metrics that take into account the proportion of *acceptable* nodes—nodes that actually meet the requirements—with respect to the selected nodes as well as the total number of nodes in the system.

- *Precision* is the proportion of nodes selected that were actually *acceptable*. A precision value of 1 indicates that every node selected by the resource discovery algorithm fulfilled its given requirements. However, precision by itself is not enough, since it could still mean that the algorithm may have missed many acceptable nodes in the system.
- *Recall* is the proportion of *acceptable* nodes that were chosen by the resource discovery algorithm of those that existed in the whole system. A low recall value means that the algorithm fails to find many acceptable nodes. However, one cannot consider recall alone, since a trivial algorithm that selects every node in the system will always have a recall of 1.

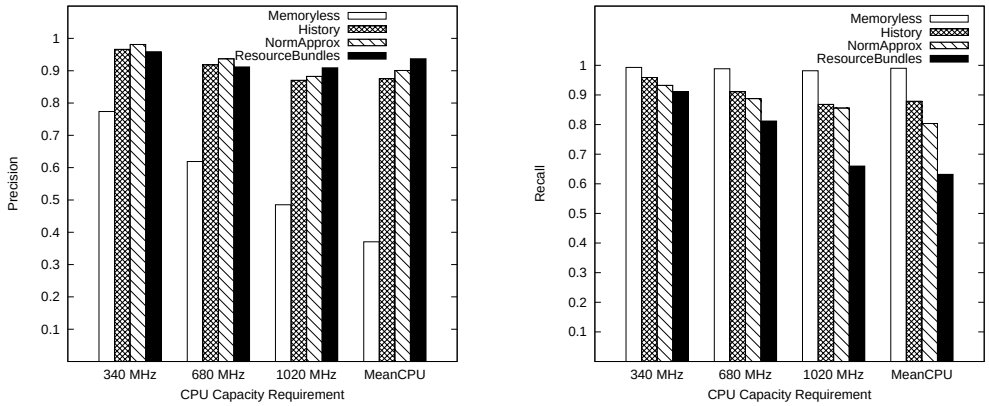
Resource Discovery Algorithms: We will compare the following resource discovery algorithms:

- ***Memoryless***: This algorithm uses the last data point for each metric on each node to estimate its expected capacity over the next day. This algorithm emulates resource discovery algorithms that use recent resource usage information to determine the suitability of a node to meet a minimum requirement, and does not incorporate statistical resource usage patterns into its decisions.
- ***History***: This is a centralized algorithm with global historical knowledge of the entire system. It maintains complete 24-hour traces for each node. This provides a baseline to determine the effect of data loss due to approximation/aggregation on the accuracy of resource discovery.
- ***Resource Bundles***: This algorithm is used for single-metric resource discovery results only. This uses Resource Bundles [8] to aggregate the resource usage histograms of groups of nodes into resource bundles. Note this technique is able to maintain the overall shape of the single-resource distributions, whereas the NormApprox method below sacrifices all but the critical region of the distributions. It must be noted, however, that the Resource Bundles algorithm performs aggregation at a more complex group-level granularity (instead of node-level granularity), supporting more functionality than merely resource discovery. In our experiments, nodes are bundled based on histogram similarity. For each bundle, if its representative meets the desired statistical requirement, all its members are considered acceptable.
- ***HiDRA***: This algorithm uses the MVN-based resource discovery described in the previous section. Our single-dimensional version of HiDRA is called **NormApprox**.

B. Validating Normal Approximations

In order to justify our modeling of resource usage profiles as normal distributions, we first show that these approximations are accurate models. Note that an important assumption we have is that the main area of interest for resource usage profiles

[5]Note that NetTx and NetRx should not be confused with network transmit or receive capacity.



(a) Precision (b) Recall

Fig. 4. Normal Approximations (NormApprox) compared against baseline resource discovery algorithms along with Resource Bundles. ($p = 95^{th}$ percentile for NormApprox, Cluster and History.) 420 PlanetLab node traces from Feb 2007 were used.

is the *critical region*. If we assume that applications will infrequently ask for percentiles less than the 90th percentile, then it is reasonable to choose the critical region endpoints to be the 90th and 99th percentiles.

But is this still accurate for resource discovery requirements between the 90th and 99th percentiles? In Figure 4 we show the accuracy in terms of precision and recall for 95th percentile CPU requirements. NormApprox shows more precision than History itself, similar to Resource Bundles. We explain this behavior as the algorithms being conservative about their choice of nodes; it finds fewer nodes than History, but is slightly more precise in its choice of nodes. This most likely is a result of the approximation and "smoothing" of the distributions so that fewer nodes are chosen. However, NormApprox has recall nearly as high as that of History; Resource Bundles lags behind in this regard since the accuracy of its aggregation technique depends on other "similar" node resource usage profiles, whereas our approximation technique is independent of other node distributions.

Thus, NormApprox is a highly accurate means for approximating node resource usage profiles using normal approximations, even though the distributions themselves may not entirely be normal. This is sufficient since we wish to model only the most important critical regions of the distribution. As we discussed in Section III-D, this normal approximation is crucial for the MVN approach to work, and we next show the benefit of this approach for multi-dimensional resource discovery.

C. Multi-dimensional resource discovery

We now evaluate HiDRA over multi-dimensional resource requirements. In our evaluation of HiDRA, we ask the following questions:

- *Is HiDRA accurate across a wide variety of resource metrics?* We would like to evaluate its performance on a set of resource requirements, varying of the number and combination of metrics selected.
- *Is HiDRA accurate across a wide variety of requirement percentiles inside the critical region?* Given the requirement metric values and critical region, if we vary the requirement percentile, we want to see how accurate HiDRA is inside the critical region, and investigate its accuracy for percentiles outside of the critical region.
- *How does the critical region size impact the accuracy of HiDRA?* As we hold constant the requirement metric values and the requirement percentile, we want to investigate the changes on accuracy as we vary the boundaries of the critical region.

1) Performance Across Different Requirements: In our analysis we used five multi-dimensional resource requirements for our evaluations, defined below.

Req	EffCPU ($\geq$ MHz)	NetTx ($\leq$ Kbps)	NetRx ($\leq$ Kbps)	5Load ($\leq$)	15Load ($\leq$)
1	500	1000	1000	5.00	
2	500	1000		5.00	
3		1000			8.00
4	1000	5000			
5	300	2000	2000		

These requirements were chosen to represent a wide variety of applications having a different multi-metric requirements. Also notice that there may be varying amounts of correlations between various metrics chosen above, e.g., while some of these metrics may be highly correlated, (e.g., effective CPU and the 5-minute Load Average), others may be largely independent of each other (e.g., Load Average and NetTx).

The results of applying these requirements (with p = 95 percentile) can be seen in Figure 5. For each of the five requirements, HiDRA consistently performs very close to the fully-informed History technique (that uses complete node traces). The number of acceptable nodes chosen between the History and HiDRA algorithms is extremely close. As in the previous results, HiDRA selects slightly fewer nodes, showing a slightly better precision but slightly worse recall.

These results show that under a wide variety of resource requirements, also among varied configurations and metrics chosen under these requirements, HiDRA is a highly accurate

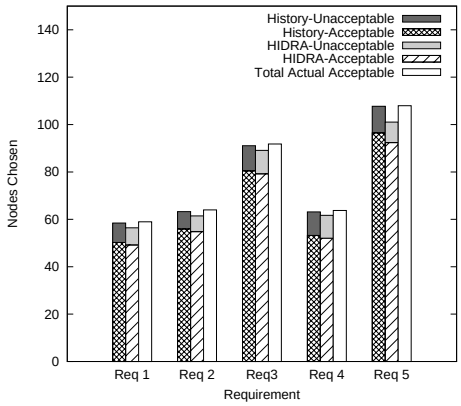


Fig. 5. Number of nodes chosen as acceptable nodes for five different requirements under the 95th percentile and critical region of 90-99.

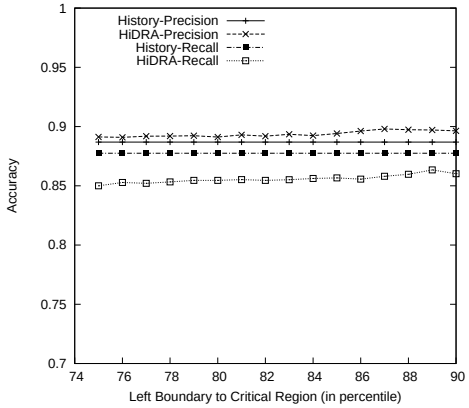


Fig. 7. 95th percentile for Req 2, Varying critical region left boundary from 75 to 90; right boundary is 99.

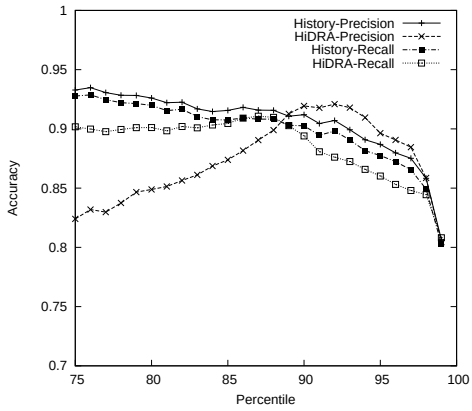


Fig. 6. Varying requirement percentiles from 75 to 99, for Req 2 and critical region of 90-99.

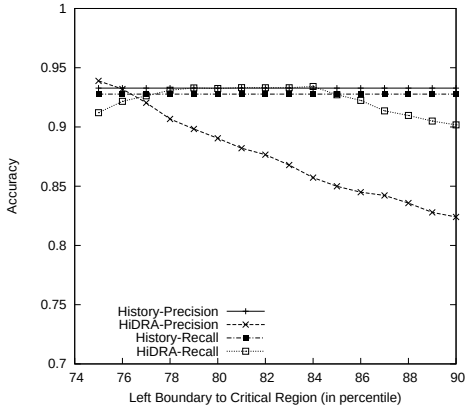


Fig. 8. 75th percentile for Req 2, Varying critical region left-boundary from 75 to 90; right boundary is 99.

algorithm for resource discovery, performing on par with a fully-informed algorithm.

2) Performance Across Requirement Percentiles: Next, we show HiDRA's performance as we vary the requirement percentile itself (for Requirement 2 above) with a critical region of 90-99. The results can be seen in Figure 6. First notice how both the precision and recall of History decline as the percentile value increases; this indicates that it is more difficult to accurately predict a selection of nodes for higher requirement percentiles. The precision and recall of HiDRA approaches the goodness of History as the percentile approaches the left boundary of the critical region, the 90th percentile. This is happening because the modeled normal distribution more accurately approximates the actual resource profile within the critical region, with an exact overlap at the endpoints of the critical region. Thus, we expect HiDRA's performance to follow History more closely within the critical region. In particular, it can be seen by how the precision and recall of History and HiDRA both match exactly at the 90th and 99th percentile points, the critical region boundaries. Inside the critical region, the precision of HiDRA is slightly better than that of History, while recall lags slightly behind. This indicates that, in the critical region, HiDRA tends to select slightly fewer nodes than History, but its selection is slightly more accurate than that of History. These results show that HiDRA is highly accurate when the requirement percentile falls inside the critical region.

3) Impact of Critical Region Boundaries: Next, we investigate the sensitivity of HiDRA's performance when we vary the critical region left boundary under two different percentile choices (for Requirement 2 above). Note that the accuracy measures for History will not change as it does not depend on the critical region of HiDRA.

We set the percentile of the requirement to 95 and vary the left boundary of the critical region, keeping its right boundary fixed at 99 percentile. The results can be seen in Figure 7. In this example, the requirement is always inside the critical region, and the measures of accuracy, both precision and recall, show signs of improvement as the critical region is bound tighter to the percentile, as expected.

Next we set the percentile to 75 and again vary the left boundary of the critical region in Figure 8. This time, the

requirement starts on the left boundary of the critical region, and moves outside of the region as we move the boundary of the critical region to the right. Not surprisingly, both precision and recall decline as the critical region moves away from the percentile of interest. However, the deviations from the History algorithm are not too severe (precision within 11%), showing that even with a misconfigured critical region, the results are still fairly accurate.

4) Selecting The Critical Region: Our results provide some guiding principles for the choice of critical region boundaries. First, we observe that tighter critical regions surrounding the requirement percentile result in higher accuracy. Second, when the requirement percentile falls outside the critical region, there is a dropoff in accuracy.

These two observations highlight a tradeoff concerning critical region selection. If the critical region is too wide, several percentiles would likely fall inside the region, but accuracy would suffer from approximating such a wide region. On the other hand, if the critical region is too small, the accuracy would be high inside the region, but many percentiles are likely to fall outside of the critical region. This tradeoff suggests that the width of the critical region can be fine-tuned and dynamically adapted based on the frequently desired percentile values. For instance, even if the initial critical region was chosen as [90-99], if most queries are for 95 percentile requirements, then the critical region can be tightened to [95-99]. Similarly, if many percentiles appear to fall outside the critical region, then, it can be expanded or the range moved to include the desired values.

D. Implementation

We deployed our HiDRA algorithm on 307 nodes in PlanetLab. Our primary goal was to validate the results we saw in our analysis, through an online deployment of HiDRA in a real system and also to measure the overhead of HiDRA. We chose three simple multi-dimensional requirements to inject into our implementation:

Req	EffCPU ($\geq$ MHz)	NetTx ($\leq$ Mbps)	NetRx ($\leq$ Mbps)
1	500	10	10
2	1000	10	10
3	1500	8	8

Nodes monitored their own resource usage time series via their own access to their local CoMon daemon process. We limited our monitoring to the three resource metrics of interest: effective CPU, network transmit bandwidth observed and network receive bandwidth observed. From this time series, the nodes computed their own normal approximations to individual metrics and also the covariance (i.e., correlation) matrix. This functionality was implemented using a Perl script. Then these normal distribution and correlation data were sent to a centralized query manager node, which executed the HiDRA algorithm using a Fortran implementation of the MVN distribution function [20]. The *critical region* was defined between the 90th and 99th percentile values for each of the resource metrics. Also, the History and Memoryless algorithms were employed in this system by each node sending a historical trace of its resource usage. For clarity, we refer to the node-level resource profile MVN distribution parameters maintained by HiDRA as *resource descriptors*.

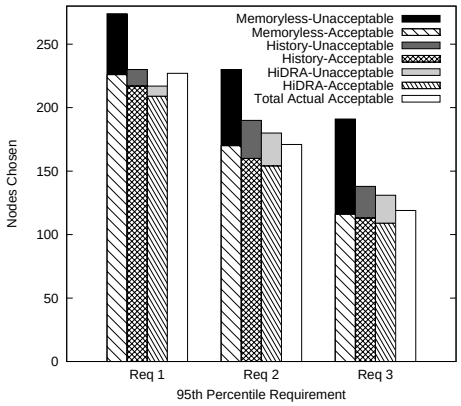


Fig. 9. Node selection for three different requirements under the 95th percentile in our PlanetLab implementation

In our resource discovery framework, we chose to utilize a centralized query manager because PlanetLab is a relatively small system. Here we place less focus on the actual means of data propagation in the system[6], and rather pay attention to the amount of data, and provide results on the data transfer overhead "per update" as a result.

We propagated updates every 10 minutes of the resource descriptors at each node to the centralized query manager. Note that if the centralized query manager goes offline, its complete data store of resource descriptors will be completely replenished within 10 minutes of coming back online by receiving the usual amount of data from each node every 10 minutes. Also note that the size of the resource descriptors is independent of the size of the trace from which it originated; it is of fixed size dependent only on how many resource dimensions are being measured.

We chose a time window of 24 hours for application deployment which is also used for resource descriptor construction. We submitted our query for the three multi-resource requirements to the central query manager and received responses from each of the algorithms. Then to evaluate this response of the resource discovery algorithms, we analyzed the future traces of the nodes chosen for deployment to measure the goodness of choice of nodes for a pseudo-application[7]. A node chosen by an algorithm that satisfied its requirements is hereby called "acceptable", and a node that does not satisfy its requirements is labeled "unacceptable" in the evaluation that follows.

Resource discovery accuracy: We evaluated our results over three different percentiles for each of the three multi-resource requirements. The results are shown in Figures 9 and 10.

[6]Forms of propagation (in structured or unstructured systems) include gossiping and flooding, as well as communication in systems that assume some super-node or hierarchical based overlay such as [17].

[7]PlanetLab has stringent rules for network bandwidth and memory consumption that are prohibitive to extensive multi-metric experimentation, which led us to use a pseudo-application, instead of a real application.

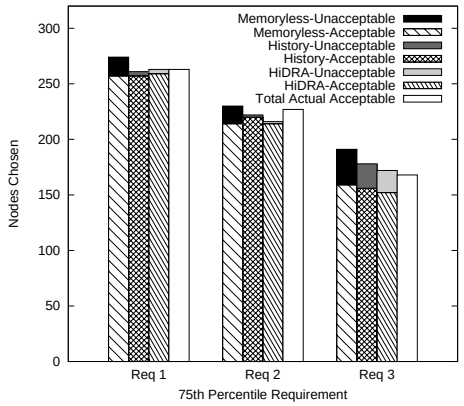


Fig. 10. Node selection for three different requirements under the 75th percentile in our PlanetLab implementation

As seen above in section IV-C, HiDRA's selection of nodes shows precision on-par with (or better than) History along each of the three requirements. Also, HiDRA has a recall slightly lower than that of History, which we also saw in the previous analysis. Even in the 75th percentile experiment that does not lie in the critical region, the precision and recall of HiDRA is remarkably close to History, showing again that HiDRA is robust to an improperly configured critical region. Our evaluation of this live implementation confirms our results in the data analysis section above that HiDRA is a highly accurate means for multi-dimensional resource discovery.

Data overhead: The total size of all 307 resource descriptors at the central node was 70 KB. A fully-informed history-based algorithm would need about 99 MB of full traces from all nodes, which is 1,458 times more overhead than using our resource descriptors. The memoryless approach would have a data transfer size of 6 KB per update, but we have shown it is highly inaccurate. Again, note that our resource descriptors describe the whole trace, so we feel this is a fair comparison, especially in systems that may use flooding or gossiping of the resource descriptors. An impressive property of HiDRA is its data size independence from the trace length. Additionally, we can also be flexible in how often we send data in HiDRA because resource usage distributions are unlikely to change over the short run, and hence HiDRA can send updates less frequently than a history-based or memoryless algorithm, reducing the network transmission overhead further.

V. Conclusion

Statistical resource discovery is critical for applications to find suitable resources in dynamic and heterogeneous large-scale distributed systems. A key problem is achieving such statistical resource discovery for multiple resources simultaneously. In this paper, we presented HiDRA, a scalable multi-dimensional resource discovery algorithm that employs *multivariate normal distribution* for the probabilistic modeling of resource capacity over multiple dimensions. Our PlanetLab trace-based analysis showed that HiDRA performs nearly as well as the fully-informed History technique (with better precision than History and recall within 3% of History). Since HiDRA has such a compact representation of node behavior on multiple metrics simultaneously, it becomes a very attractive solution for large-scale systems that need a scalable resource discovery mechanism. Also, HiDRA provides statistical guarantees to applications that allow deployments to be more stable and reliable, not subject to frequent failures or migration scenarios. Our live implementation in the PlanetLab testbed shows our system to be a feasible, low-overhead method in finding acceptable nodes for applications. In future work, we will investigate an integration with our previous work [8].

References

[1] I. Foster and C. Kesselman, Eds., *Grid2: Blueprint for a New Computing Infrastructure*. Morgan Kauffman, CA, USA, 2004.

[2] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: An Experiment in Public-Resource Computing," *Communications of the ACM*, vol. 45, no. 11, 2002.

[3] K. Church, A. Greenberg, and J. Hamilton, "On delivering embarrassingly distributed cloud services," in *Seventh ACM Workshop on Hot Topics in Networks (Hotnets'08)*, 2008.

[4] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable," in *Proceedings of ACM SIGCOMM*, Aug. 2003.

[5] V. Lo, D. Zappala, D. Zhou, Y. Liu, and S. Zhao, "Cluster Computing on the Fly: P2P Scheduling of Idle Cycles in the Internet," in *Proc. of the 4th IEEE Conference on Peer-to-Peer Systems*, 2004.

[6] J.-S. Kim, P. Keleher, M. Marsh, B. Bhattacharjee, and A. Sussman, "Using Content-Addressable Networks for Load Balancing in Desktop Grids," in *HPDC'07*, Jun. 2007.

[7] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat, "Distributed resource discovery on PlanetLab with SWORD," in *WORLDS'04*, Dec. 2004.

[8] M. Cardosa and A. Chandra, "Resource bundles: Using aggregation for statistical wide-area resource discovery and allocation," in *ICDCS'08*, Jun. 2008, pp. 760–768.

[9] Y. J. Kim, A. Boyd, B. D. Athey, and J. M. Patel, "miblast: scalable evaluation of a batch of nucleotide sequence queries with blast," *Nucleic Acids Res*, vol. 33, pp. 4335–4344, 2005.

[10] R. Raman, M. Livny, and M. Solomon, "Matchmaking: Distributed Resource Management for High Throughput Computing," in *HPDC'98*, Jul. 1998.

[11] D. Oppenheimer, B. Chun, D. Patterson, A. C. Snoeren, and A. Vahdat, "Service Placement in a Shared Wide-Area Platform," in *Usenix Annual Technical Conference*, Jun. 2006.

[12] J. W. Mickens and B. D. Noble, "Predicting node availability in peer-to-peer networks," in *Proceedings of the ACM SIGMETRICS Conference*, 2005, pp. 378–379.

[13] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: supporting scalable multi-attribute range queries," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 353–366, 2004.

[14] I. Chang-Yen, D. Smith, and N.-F. Tzeng, "Structured peer-to-peer resource discovery for computational grids," in *Proceedings of the 15th ACM Mardi Gras conference*, 2008, pp. 1–8.

[15] M. Litzkow, M. Livny, and M. Mutka, "Condor - a hunter of idle workstations," in *Proceedings of the 8th International Conference of Distributed Computing Systems*, June 1988.

[16] R. Wolski, "Experiences with predicting resource performance on-line in computational grid settings," *SIGMETRICS Perform. Eval. Rev.*, vol. 30, no. 4, pp. 41–49, 2003.

[17] P. Yalagandula and M. Dahlin, "A Scalable Distributed Information Management System," in *SIGCOMM'04*, 2004.

[18] R. V. Renesse, K. P. Birman, and W. Vogels, "Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining," *ACM Tr Comp Sys*, vol. 21, no. 2, pp. 164–206, 2003.

[19] K. Park and V. S. Pai, "Comon: a mostly-scalable monitoring system for planetlab," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, 2006.

[20] A. Genz, "http://www.math.wsu.edu/faculty/genz/software/software.html."