Resource Bundles: Using Aggregation for Statistical Wide-Area Resource Discovery and Allocation*

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 is an important process for finding suitable nodes that satisfy application requirements in large loosely-coupled distributed systems. Besides inter-node heterogeneity, many of these systems also show a high degree of intra-node dynamism, so that selecting nodes based only on their recently observed resource capacities for scalability reasons can lead to poor deployment decisions resulting in application failures or migration overheads. In this paper, we propose the notion of a resource bundle*—a representative resource usage distribution for a group of nodes with similar resource usage patterns—that employs two complementary techniques to overcome the limitations of existing techniques:* resource usage histograms *to provide statistical guarantees for resource capacities, and* clustering-based resource aggregation *to achieve scalability. Using trace-driven simulations and data analysis of a month-long PlanetLab trace, we show that resource bundles are able to provide high accuracy for statistical resource discovery (up to 56% better precision than using only recent values), while achieving high scalability (up to 55% fewer messages than a non-aggregation algorithm). We also show that resource bundles are ideally suited for identifying group-level characteristics such as finding load hot spots and estimating total group capacity (within 8% of actual values).*

1. Introduction

Recent years have seen increasing use of loosely-coupled distributed platforms for scientific computation [1, 10], data sharing and dissemination [3, 6], and experimental testbeds [4]. While such platforms are highly attractive due to their low deployment cost and inherent scalability, they are also highly heterogeneous and dynamic [8]. The nodes participating in such platforms differ widely in their resource capabilities such as CPU speeds, bandwidth, and memory capacity. As a result, *resource discovery* is often used in such large-scale systems to find suitable nodes that satisfy application requirements.

Many existing resource discovery systems [13, 9, 8, 17] rely on the recently observed resource capacities of individual nodes to make their deployment decisions. However, resource allocation decisions based on current status of nodes have severe limitations in these systems, because of the presence of intra-node dynamism in addition to the inter-node heterogeneity. Individual nodes can have widely varying resource capabilities due to varying loads, network connectivity, churn, or user behavior. For instance, a resource usage study of PlanetLab [14] has shown that node resource capabilities fluctuate on the order of about 30 minutes. Such dynamism in node-level resource capacities makes it difficult to deploy long-running services and applications that need consistent resource availability to ensure desired performance and avoid disruptions or migration overheads.

To handle the inherent heterogeneity and dynamism in such systems, the resource discovery process employed in such systems must be able to provide *statistical guarantees* on application resource requirements. While incorporating long-term resource availability information is likely to improve the resource discovery decisions substantially [14], most existing resource discovery systems use only the recent node usage information for scalability and simplicity reasons. It helps in reducing the amount of monitoring data that needs to be exchanged between nodes in the system, and enables easy location of desirable nodes (e.g., by mapping resource requirements to node IDs in case of DHT-based resource discovery systems [13, 9]). We argue in this paper that for providing statistical resource guarantees in a scalable manner, the resource usage information from nodes can be approximated both in temporal (long-term usage pattern) and spatial (number of nodes with similar usage patterns) dimensions.

*This work was supported in part by an NSF CAREER Award CNS-0643505.

In this paper, we propose the notion of a *resource bundle*—a representative resource usage distribution for a group of nodes with similar resource usage patterns. A resource bundle employs two complementary techniques to capture the long-term resource usage behavior of a set of nodes: (i) *resource usage histograms* to provide statistical guarantees for resource capacities, and (ii) *clustering-based resource aggregation* to achieve compact representation of a set of similarly-behaving nodes for scalability.

Besides providing a scalable resource discovery mechanism to achieve stable application deployment, resource bundles can also be used for several other purposes in a large distributed system. Resource bundles can be used to easily find a *group of nodes* satisfying a common requirement. Resource bundles can also be used to find load *hot spots*: geographical regions in the distributed system with several nodes experiencing overloads due to reasons such as heavy demand for a popular resource in that region or locality-based application stresses. The identification of such hot spots can be used to inform decisions about application deployment or load balancing. Finally, resource bundles can also be used for *auditing and accounting* purposes, e.g., to determine the resource assignment of a distributed application running on multiple nodes, or to determine the spare capacity in an administrative domain.

We evaluate the performance of resource bundle-based resource discovery using trace-driven simulations and data analysis of a month-long PlanetLab trace. Our results show that resource bundles are able to provide high accuracy for resource discovery through the use of resource usage histograms (up to 56% better precision than an algorithm based on current usage values), while achieving high scalability through aggregation (up to 55% fewer messages than a non-aggregation algorithm). We also show that resource bundles are ideally suited for identifying group-level characteristics such as finding load hot spots and estimating total group capacity (within 8% of actual values).

2. Statistical Node Behavior

2.1. System Model

We assume our system is a large-scale wide-area distributed system. Participant nodes are geographically distributed and could span multiple administrative domains. We assume the nodes are interconnected by an interconnection overlay, using a DHT or a flooding-based approach, which allows nodes to communicate with other nearby nodes. Nodes monitor their own resource capacities over time and can exchange messages as required. Further, we assume a hierarchical structure can be constructed on top of the overlay, e.g., using methods provided in SDIMS [24].

2.2. Statistical Resource Requirements

During the resource discovery process, applications typically seek nodes meeting certain resource requirements. In the presense of intra-node dynamism, we must avoid application performance degradation, failures, or need for frequent migrations [14] resulting in large overhead. It would be desirable to provide statistical resource guarantees so that applications can be deployed on nodes that are likely to satisfy the minimum desired requirement for a certain period of time. We formalize this notion of statistical resource requirement as follows:

Definition 1 ***Statistical Requirement:*** *We define a statistical requirement r 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.*

Intuitively, an application can specify that it needs a resource R to meet a minimum capacity level *c* for at least *p* percent of time (corresponding to its tolerance to overload) over a time duration *t* (which could depend on its length of execution and overheads of disruption and migration, etc.). The goal is to avoid serious service disruptions or reallocation penalties (e.g. overloads, migration overheads) over time *t*. Thus, using this definition of statistical requirements, a compute-intensive application can specify a requirement {CPU, 1GHz, 95, 24hrs}, which means it requires a 95 percentile CPU capacity of 1 GHz over 24 hours.

2.3. Resource Usage Representation

Since different applications can specify different values of c, p, and t, our resource usage representation must be flexible towards a wide variety of application requirements. We may need to capture the resource usage behavior over different time durations (such as an hour, day, week, etc.) to incorporate requirements over different time granularities.

To provide a general way to handle different resource requirement specifications, we propose the use of resource usage histograms with an associated observation time period T, which represent the resource capacity distributions from observations over the past T time units. A statistical requirement can be mapped to a resource capacity histogram (with p% of the capacity observations to the right of a vertical line corresponding to capacity c). A separate histogram can be maintained for each resource type (e.g., CPU, memory) and for each time granularity (e.g., hour, day, week); intermediate time granularities can be interpolated from these histograms. Nodes can construct histograms from their own historical observations.

Using histograms to represent resource usage data has two primary advantages: (i) requirement percentiles (corresponding to p) for a particular resource capacity are now

straightforward computations from the given histograms, and (ii) histograms help us preserve all usage data, so that even if different applications specify different resource capacity requirements with different tolerances, these can be easily captured using the same histogram representation.

This representation technique is complementary to any prediction techniques that may be able to predict future resource usage behavior based on historical observations. Predictions could easily be converted into histograms.

3. Resource Bundles

While using resource usage histograms provides a means to capture an accurate representation of an individual node's dynamic resource usage pattern and enables the satisfaction of statistical resource requirements, it can potentially create a scalability problem in a large wide-area distributed system. The statistical information for each node would be represented by multiple histograms, corresponding to different resources and different time scales. Disseminating this amount of data over the network can create significant network traffic, making it infeasible for each node to have a global view of the system. Moreover, if the goal is to find *multiple* nodes meeting a certain requirement, it would be desirable to combine this discovery process rather than having to find individual suitable nodes separately.

This raises the following questions. Can we use these representations in a scalable manner to make better resource discovery decisions in a large system? Secondly, can we use these node behaviors to provide any collective information about group-level usage patterns, e.g. for nodes within an administrative domain or assigned to an application?

3.1. Resource Aggregation

Aggregation [24], particularly hierarchical aggregation [5], is a common technique employed in large distributed systems for the scalable dissemination of information. Aggregation essentially compresses the amount of transmitted data in the system while preserving the overall information content. In the context of resource discovery, this would correspond to a suitable "compression" of the node resource usage patterns to achieve a desirable tradeoff between the quality of resource discovery and the overhead of network data transmission in the system.

Our goal is to achieve the same quality of resource discovery as a global resource discovery system with full historical node-behavioral knowledge, but to significantly compress the amount of necessary node-behavioral representation data in the system in order to achieve scalability. Such an aggregation of node resource usage distributions for a group of nodes can be used to represent: (i) an accurate approximation of any individual node's resource usage for the accurate discovery of desirable nodes based on a resource requirement, (ii) collective resource usage behavior of a group of nodes to provide information about load patterns or resource usage behavior for a set of related nodes (e.g. geographically or by assignment), and (iii) overall group capacity for resource usage tracking (e.g. for audit or accounting purposes).

A naive approach to aggregation for a set of nodes would be to compute the average resource capacity distribution across all nodes. An example would be the averaging of left-skewed and right-skewed nodes, producing a bimodal representative. While averaging allows the estimation of overall capacity of the group of nodes, it is a poor representation of individual node-level behavior. This is because it does not account for the heterogeneity of the nodes in the system. Thus, this approach could result in a highly inaccurate view of individual node resource capacities.

3.2. Defining Resource Bundles

To account for the heterogeneity of nodes, we define the notion of a *resource bundle:* an aggregation of a group of nodes with *similar* resource capacity distributions. By combining only similar nodes together, such an aggregation process will preserve the individual node distributions more accurately. Figure 1 shows a high-level view of the notion of resource bundles and how they might be constructed. First, a group of nodes are bundled based on the similarity of their resource capacity histograms. Second, each bundle produces a representative distribution that can be used to characterize the whole bundle. The question is how can we identify such groups of similar nodes to construct a resource bundle, and compute its representative accurately?

3.2.1 Aggregation via Clustering

To identify nodes with similar distributions, we propose the use of *clustering* algorithms that have traditionally been used in data mining applications to group together statistically similar data elements. However, a clustering algorithm must meet several requirements in our context:

- The data to be clustered (i.e., the resource histograms) is not single-point, but multi-element (consisting of multiple histogram bins). The clustering algorithm must be able to *handle such multi-element data*.
- The clustering algorithm must be *distribution-free*. The node resource usage histograms could represent arbitrary distributions, and cannot be assumed to conform to standard distributions (e.g. Gaussian).
- It is desirable if it can produce a *compact representation of the collective resource usage* of these nodes in order to characterize the bundle (e.g. low-capacity).

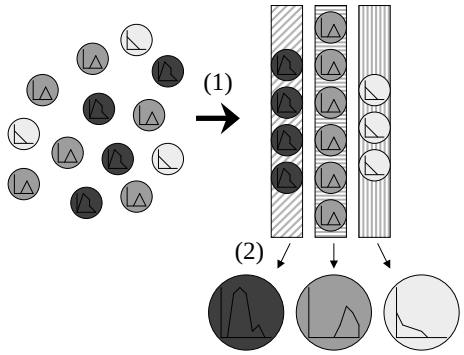


Figure 1. Constructing Resource Bundles.

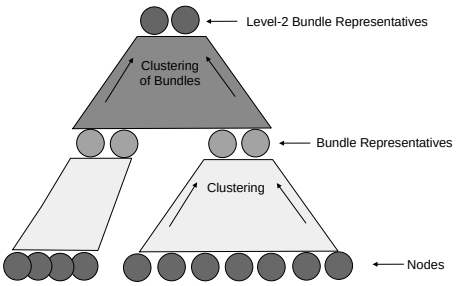


Figure 2. Hierarchical aggregation

A clustering algorithm that meets the above requirements is the *multinomial model-based expectation maximization (EM)* clustering algorithm [26]. This clustering algorithm has been used primarily for the purposes of clustering text documents with common words. We first describe this algorithm in a document clustering context for ease of exposition, and then describe how it maps to our context.

In a document clustering context, each document is considered as a "bag of words", and is represented as a vector of word frequencies. Then, the set of all documents is represented as a mixture of multinomial distributions on these word frequencies, with each document belonging to one such distribution. The probability that a document belongs to one of the clusters corresponding to a multinomial distribution is given by [26]: $P(d_i|\lambda_j) = \prod_l P_j(w_l)^{n_{il}}$, where λ_j is the set of parameters for model j, n_{il} is the frequency of occurrences of word w_l in document d_i, and $P_j(w_l)$ is the probability of word w_l occurring in cluster j. Further, $\sum_l P_j(w_l) = 1$ holds.

Mapping the document clustering scenario to our context, we can think of nodes corresponding to documents and histogram bin magnitudes for node-level resource usage distributions corresponding to document word frequencies. The clustering algorithm then maps nodes to clusters based on the similarity of their resource usage histograms. In other words, this algorithm will group those nodes together that have similar histogram bin-magnitudes, meaning it can cluster nodes having arbitrary (but similar) distributions. Such a cluster of closely related nodes returned by the clustering algorithm is considered a resource bundle.

In practice, the multinomial model-based EM clustering takes a set of vectors as its input and forms clusters based on the similarity of corresponding vector elements. It is a hill-climbing algorithm that adjusts its mapping of vectors to clusters iteratively in order to maximize the expected objective value achieved from its clustering.

This clustering algorithm meets the above requirements. It is able to handle multi-point (vector) data, it operates on arbitrary distributions, and characterizes the common statistical properties of clustered nodes in a compact manner.

3.2.2 Bundle Representatives

As described above, the multinomial model-based EM algorithm associates each resource bundle with a multinomial distribution that captures the statistical properties of the nodes that are members of the bundle. This multinomial distribution can be thought of as the *bundle representative:* an aggregate distribution that is a compact representation of the individual node distributions in the bundle.

Bundle representatives are the resulting distributions that represent the mean node capacity distribution within each respective bundle. In the multinomial model clustering algorithm, cluster representatives are the result of probabilistic models; however the difference between these models and the mean node capacity distribution is negligible.

In Section 4, we will quantify the closeness of the representative to its members by its effectiveness in resource discovery, hot spot detection and capacity estimation.

3.3. Hierarchical Aggregation

While aggregation, as described above, enables the creation of resource bundles that closely approximate individual node behavior for a similar set of nodes, the question is whether bundles can be further aggregated to provide a meaningful view of the combined set of nodes corresponding to multiple bundles. The ability to combine resource bundles is particularly desirable in a large system where we may want to get concise estimates of resource capacities of nodes at different granularities; for instance, at a local site level, at an administrative domain level or at a global level.

Building on the assumed ability of the system-employed overlay to support a hierarchical information structure, we propose the use of *hierarchical aggregation* in combination with resource bundles through the use of recursive clustering, i.e., successive clustering of bundle representatives at different levels of the hierarchy. Figure 2 is a high-level illustration of this process. Groups at the bottom level are

individual sets of node distributions. These nodes are initially clustered, producing bundle representatives which are then propagated to the next level of the hierarchy to create level-2 representatives, thus beginning the recursive representative clustering process.

Note that this recursive aggregation must consider individual bundle cardinalities; this prevents undue influence from low-cardinality bundle representatives during the recursive bundling process. To remedy this issue, we use bundle cardinalities as *representative weights* in the process of hierarchical clustering. Formally, for a bundle histogram H_i having cardinality w_i, we present it to the clustering algorithm as $W_i = H_i w_i$.

If the underlying topology of the distributed system causes nodes to be grouped by locality, our hierarchical approach will be suitable for finding localized sets of resources for application deployment. Further, bundle representatives can be used to predict group-level capacity levels. Our ability to predict group-level capacities results in another high-level ability to detect hot spots within large distributed systems.

4. Evaluation

4.1. Data Analysis Methodology

We used a PlanetLab trace of 427 nodes obtained by CoMon [15] from February 2007 for our experiments. We used *free CPU capacity* (from *CPU Burp*) as the resource of interest. We used a time period of 24 hours, computing node histograms on a 24-hour period, one per day. Each histogram consisted of 100 bins, each representing 34 MHz.

We used MatLab as our main tool for data analysis incorporating an already implemented Matlab package for the multinomial model EM clustering algorithm [25]. We emulated single-level (flat) clusterings, as well as hierarchical clusterings of multiple levels. Due to space constraints, we present only a subset of our results and more details can be found in a technical report [2].

4.1.1 Emulating Resource Discovery

To evaluate the accuracy of a resource bundle-based resource discovery process in finding desirable nodes, we emulate a resource discovery process as follows. The resource discovery algorithm is run on an *observation time window* to determine the *choice* of acceptable nodes that meet the desired resource requirement. We then compare this choice to the actual set of nodes that satisfied the desired requirement over a *solution time window*: the time frame during which nodes would have been allocated to the application.

For the purposes of our experiments, we defined the statistical requirement r={CPU, c, p, 24hrs}, varying c and p. The goal was to *find all nodes* (of 427) meeting *r*. Notice we did not specify how many nodes an application needs for its deployment. Instead, we had the algorithms search for the complete set of acceptable nodes.

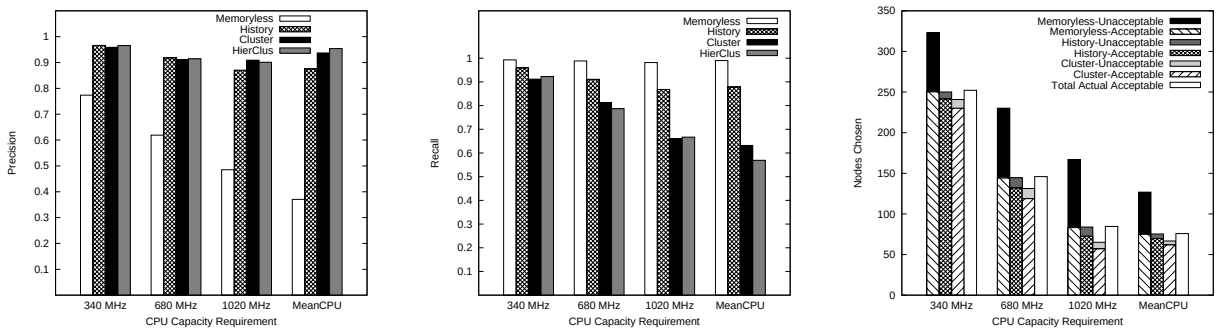
We used the entire trace by intially using Feb 1 as the observation window and Feb 2 as the solution window. The windows were then shifted by one day, thus giving us 27 samples of trace data to evaluate our algorithms.

4.1.2 Comparison Algorithms

We compared the following resource discovery algorithms:
• ***Memoryless***: This algorithm uses the last CPU capacity data point for 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 CPU capacity histograms for each node. This provides a baseline to determine the effect of data loss due to aggregation on the accuracy of resource discovery.
• ***Aggregation (Cluster)***: This algorithm uses resource bundles to aggregate the resource usage histograms of groups of nodes into resource bundles. Nodes are bundled into k bundles based on histogram similarity. For each bundle, if its representative meets the desired statistical requirement r, all of its members are selected as acceptable nodes.
• ***Hierarchical Aggregation (HierClus)***: This algorithm uses recursive bundling over a 2-level hierarchy consisting of 420 PlanetLab nodes divided into 6 random groups of size 70. The hierarchical aggregation algorithm first reduces each group of nodes to k first-level resource bundles. All the first-level resource bundles from across all groups are then further aggregated into k second-level resource bundles. For resource discovery purposes, the algorithm examines these second-level bundle representatives and if a bundle representative H_i meets requirement r, then all nodes represented by H_i (recursively) are selected as acceptable.

4.1.3 Evaluation Metrics

The accuracy of resource discovery was measured using two metrics: $Precision = \frac{n_{acc}}{n_{tot}}$ and $Recall = \frac{n_{acc}}{N_{acc}}$, where, n_{acc} = acceptable nodes chosen, n_{tot} = total nodes chosen, N_{acc} = total acceptable nodes in the system. Intuitively, high precision means that a high fraction of the nodes returned by a resource discovery algorithm are actually acceptable, thus reducing the chances of poor allocation decisions. On the other hand, recall measures what percentage of the total acceptable nodes in the system are



(a) Precision (b) Recall (c) Absolute number of nodes discovered

Figure 3. Aggregation compared against baseline resource discovery algorithms (95^{th} percentile).

discovered by the algorithm, indicating how well it can locate acceptable nodes in the system.

4.2. Accuracy in Resource Discovery

Figure 3 compares the precision and recall of the algorithms using 10 clusters[1]. The *mean requirement* is the average over all possible requirements (for 100 histogram bins).

As seen from Figure 3(a), both Cluster and History have significantly better precision (87-97%) than Memoryless (37-77%). This results from their use of historical information to make statistically accurate decisions.

This same behavior also explains the recall values in Figure 3(b), which shows Memoryless had the highest recall, while History came in second, and Cluster consistently had the worst recall among the three. Memoryless turns out to be an unusually optimistic predictor, finding nearly all of the acceptable nodes but also many other unacceptable (but temporarily well-performing) nodes, leading to poor precision and excellent recall.[2] Cluster suffers in recall because it is conservative similar to History. However it misses additional acceptable nodes due to its loss of information from aggregation; it might combine acceptable nodes along with unacceptable nodes into the same bundle. Thus it achieves higher precision than History but experiences worse recall.

However, when we look at the absolute number of nodes discovered by each algorithm in Figure 3(c), we find that the actual number of nodes missed by Cluster is typically between 10-20, even in the worst recall case (MeanCPU - 13 nodes missed), while the number of additional (unacceptable) nodes returned by Memoryless can be in the order of 50-85 nodes. This indicates that the impact of missing acceptable nodes by Cluster is comparatively smaller than that of finding additional poor nodes by Memoryless.

To consider the relative impact of high precision vs. low recall and vice versa, we note that the *goodness of choice* of nodes from the application's perspective is primarily affected by precision. Once the allocation decision has been made, precision ultimately reflects the confidence of the application in the selected nodes meeting the given requirement. On the other hand, recall's effects are dependent on the system context. In a system where very few nodes meet the given requirement, a high recall may be desired so that most of the acceptable nodes could be found. However, in a large system with several acceptable nodes, recall would primarily affect query latency: low recall implies that acceptable nodes will be missed, therefore taking the query longer to find the desired number of acceptable nodes.

Figure 3 also compares HierClus with Cluster for precision and recall statistics. Precision does not suffer from the recursive aggregation in HierClus, while the recall is slightly lower (about 6% for MeanCPU). This shows that the loss of accuracy in hierarchical clustering is small compared to a single-level clustering.

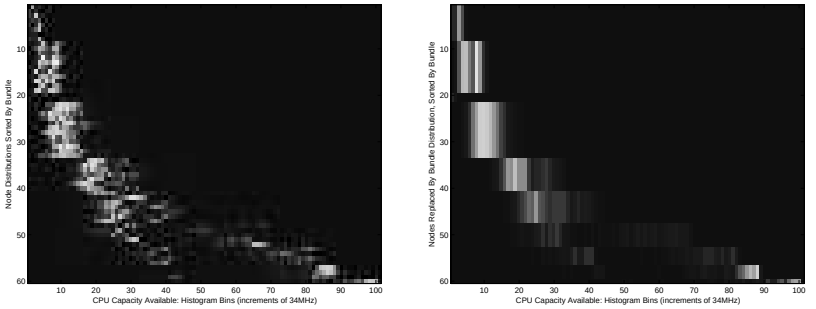
In summary, our precision and recall evaluations show that resource discovery using *our resource bundle approach is able to find a choice of nodes similar in quality to those discovered by a fully-informed history-based algorithm*, although its set of chosen nodes is smaller, thereby missing a few potentially acceptable nodes.

4.3. Identifying Group Characteristics

One potential advantage of using resource bundles is to concisely capture group characteristics of sets of nodes such as their overall load behavior and spare capacity. Such group-level characteristics could inform decisions of load balancing or capacity planning without having to rely on fine-grained node-level statistics. Next, we perform an ex-

[1]We investigate the impact of number of clusters on aggregation accuracy in the technical report [2].

[2]Memoryless finds, on average, $p = 95\%$ of the acceptable nodes in the system along with every other node that had a resource capacity $c_t \geq c$ at the moment of observation t.



(a) Nodes, sorted by bundle rep (b) Nodes, replaced and sorted by bundle rep

Figure 4. Bundle representatives can be used to detect hot spots in the network.

periment to evaluate this potential of resource bundles for geographically related nodes in PlanetLab.

To establish proximity-based groupings of nodes for our experiment, we hand-selected 5 geographically distributed group leaders in our PlanetLab trace. These leaders were respectively from UMASS Amherst, UFL, UT Austin, UWash, and UMN, representing different US regions. Pairwise pings between these leaders and the remaining PlanetLab nodes were taken, forming a total of 300 responsive PlanetLab nodes. We formed 5 groups of 60 nodes each based on their proximity from the selected group leaders.

4.3.1 Load Hot Spot Detection

Figure 4(a) shows a visual display of the resource usage histograms of 60 nodes from one of the geographical groups. The figure is a gray-scale image where the x-axis corresponds to CPU capacity, the y-axis corresponds to node identifiers (sorted based on the bundle identifier assigned by our clustering algorithm), and the brightness of a point corresponds to the magnitude of the histogram bin. Thus, a node with brighter values towards smaller CPU capacity values (e.g., node 10) has a smaller spare CPU capacity available. Figure 4(b) shows the corresponding bundle representatives we obtain by our clustering algorithm; low capacity nodes are clearly separated from high capacity nodes. Using such group-level views can allow administrators to identify potential load hot spots by using appropriate thresholds.

4.3.2 Capacity Estimation

We now investigate the capacity estimation abilities of our aggregation algorithm. To what level of accuracy can the aggregation-based algorithm estimate the resource capacity of a group of nodes for the next day based on observations of a 24-hour period? We compare the results to those of a history-based capacity estimation algorithm that has fine-grained knowledge of individual node resource usage. For the aggregation-based algorithm, group capacities are estimated by taking a weighted sum of the mean capacities of the bundle representatives; for the history-based estimation algorithm, the mean capacities of individual nodes are added.

We found the aggregation-based algorithm (using 10 clusters) is able to estimate group-level capacity within 8% accuracy overall and within 3% accuracy of a history-based estimation algorithm [2]. We also considered an average of our group-level predictions for all clusterings that used between 5 and 20 clusters. We found that the impact of the cluster size is minimal ($< 1\%$ accuracy).

4.4. Trace-Driven Aggregation Simulation

We now present results from a trace-driven simulation of a hierarchical aggregation algorithm. The purpose of these simulations was to quantify the overhead and query latency of our resource discovery algorithms in a large-scale system with a realistic topology. We have based our simulator on PeerSim [16], a widely used simulator for distributed protocol evaluation. We have used our month-long PlanetLab trace to drive these simulations. A hierarchical overlay was implemented using *WireInetTopology* in PeerSim. The depth of the hierarchy varied from 5 to 7 levels. We generated 10 random topologies for each network size and took an average over them. Single nodes may now be bundled together with multi-level bundles, so we define a *bundle* as representing 1 or more nodes.

At each simulator cycle, each node propagates its bundle list upwards to its parent. Each node upon receiving its messages, updates its data store and if the number of bundles it stores exceeds a *maximum bundle threshold* B_t, the aggregation algorithm is executed at that node, consolidating its current list of bundles into B_a aggregate bundles. We ran our resource bundle-based aggregation algorithm over the simulated topologies. We compared it to a baseline

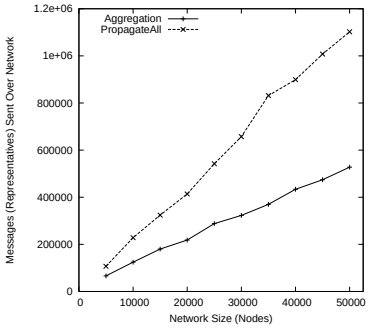


Figure 5. Data transfer overhead.

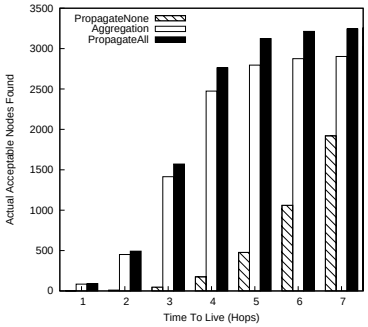


Figure 6. Nodes found by query latency.

PropagateAll algorithm, which uses a threshold $B_t = \infty$, so that no aggregation takes place and all node-level capacity distributions are propagated all the way to the root. Another baseline algorithm we used was ***PropagateNone***, where no resource discovery information is propagated between nodes beforehand; resource discovery must be performed through the probing of each node individually for its capacity distribution. All three algorithms use bundle histograms for resource discovery purposes.

Data Transfer Overhead. We investigated the overhead of Aggregation and PropagateAll by measuring the data transfer necessary for a complete system-wide propagation to the root node in the hierarchy. We used the average of 10 randomly generated topologies for each network size. Data transfer is measured in the number of bundles sent over the network. For Aggregation, we used $B_t = 70, B_a = 10$. Figure 5 measures overhead by the size of the network. The overhead of Aggregation increases at a slower pace than PropagateAll, with only about 48-61% messages required across the different network sizes.

Query Latency. We now analyze query latency from our trace of 11,529 PlanetLab node distributions (427 nodes $\times$ 27 days). We chose not to fix an application-desired number of acceptable nodes to be found; instead, we measured how many nodes were found given a time-to-live value measured in hops. We chose $r = \{$CPU, 680MHz, 95, 24hrs$\}$ to determine acceptable nodes. Our results are the average of queries injected at 500 random points in the network.

Figure 6 shows that Aggregation finds nearly as many nodes as PropagateAll (within 11%), but with about half the data transfer overhead in the same hierarchical structure. PropagateNone, which has no data overhead, performs much more poorly, returning only about 1-15% of nodes up to 5 hops. It also has significant query-time overhead, as it must query each individual acceptable node it finds.

Our results show that *using aggregation provides a good tradeoff between data dissemination overhead and query-time overhead, with fairly high accuracy.*

5. Related Work

Statistical resource guarantees: Previous work on resource overbooking [22] has used offline application profiling for determining application placement at the single-node level to meet statistical QoS levels. Our work targets online node-level as well as group-level capacity observations for statistical resource discovery. Recent work in Computational Markets [20] has used predictions of consumer market resource costs to maintain QoS levels.

Resource discovery: Condor matchmaking [17] relies on a centralized matchmaker for matching resources to applications. Our focus is on decentralized resource discovery. SWORD [13] is a scalable resource discovery service deployed on PlanetLab that uses a DHT underlay to store node statistics. While the focus of SWORD is on providing scalability in terms of querying, we have also addressed the issue of scalability in data collection and propagation. Several recent approaches have explored resource discovery in dynamic desktop Grid environments [8, 9, 7]. Most of these approaches use peer-to-peer query forwarding, and focus on finding single nodes for individual application tasks. Our work instead emphasizes hierarchical propagation of node resource usage information to enable the quick discovery of large groups of nodes for collective deployment. Also, while most existing resource discovery mechanisms focus on finding suitable nodes meeting short-term requirements, our approach is geared towards meeting long-term statistical requirements.

Aggregation: Recent work on Information Planes [5] and Astrolabe [18] provide frameworks for scalable deployments of information systems. They have shown hierarchical aggregation to be a useful model for scalability, but in a more general context. Similarly, SDIMS [24] mentions aggregation as a key model for scalability, and specifies language constructs to support aggregation. While these systems provide a general aggregation framework without providing specific aggregation functions, our focus is on solving the specific aggregation problem for resource usage distributions, and a key contribution of our work is to show the use of clustering-based resource bundles for this purpose.

Historical data and prediction: Prediction has been used

in several contexts such as availability prediction [11], Web workload prediction [19], and workload prediction for multi-tier Internet applications [21]. Network Weather Service [23, 12] uses tournament predictors to accurately predict trends in resource usage levels. However, their predictions are limited to the next measurement point, while our techniques are flexibile in the temporal and statistical descriptions of resource usage. Prediction is complimentary to resource usage profiles; we can incorporate prediction for greater accuracy in providing statistical guarantees.

6. Conclusion

In this paper, we addressed the problem of scalable resource discovery in large loosely-coupled distributed systems. A key problem in these systems is that the resource capacities of individual nodes can vary drastically over time. This dynamism means that selecting nodes based only on their recently observed resource capacities can lead to poor deployment decisions resulting in application failures or migration overheads. However, most existing resource discovery mechanisms rely only on recent observations to achieve scalability in large systems.

We proposed the notion of a resource bundle—a representative resource usage distribution for a group of nodes with similar resource usage patterns—that employs two complementary techniques to overcome the limitations of existing techniques: resource usage histograms to provide statistical guarantees for resource capacities, and clustering-based resource aggregation to achieve scalability. Using trace-driven simulations and data analysis of a month-long PlanetLab trace, we showed that resource bundles are able to provide high accuracy for statistical resource discovery, while achieving high scalability. We also showed that resource bundles are ideally suited for identifying group-level characteristics such as finding load hot spots and estimating total group capacity.

References

[1] D. Anderson. BOINC: A System for Public-Resource Computing and Storage. In *Proceedings of the 5th ACM/IEEE International Workshop on Grid Computing*, 2004.

[2] M. Cardosa and A. Chandra. Resource Bundles: Using Aggregation for Statistical Wide-Area Resource Discovery and Allocation. Technical Report 07-027, Dept. of CSE, Univ. of Minnesota, Nov. 2007.

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

[4] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, July 2003.

[5] B. Chun, J. M. Hellerstein, R. Huebsch, P. Maniatis, and T. Roscoe. Design Considerations for Information Planes. In *WORLDS'04*, Dec. 2004.

[6] B. Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems*, June 2003.

[7] A. Gupta, D. Agrawal, and A. E. Abbadi. Distributed resource discovery in large scale computing systems. In *SAINT'05*, 2005.

[8] A. Iamnitchi and I. Foster. On Fully Decentralized Resource Discovery in Grid Environments. In *International Workshop on Grid Computing*, Denver, Colorado, Nov. 2001. IEEE.

[9] 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*, June 2007.

[10] 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 *Proceedings of the IEEE Fourth International Conference on Peer-to-Peer Systems*, 2004.

[11] J. Mickens and B. Noble. Exploiting Availability Prediction in Distributed Systems. In *NSDI'06*, May 2006.

[12] Network Weather Service. `http://nws.cs.ucsb.edu/ewiki/`.

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

[14] 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*, June 2006.

[15] K. Park and V. S. Pai. Comon: a mostly-scalable monitoring system for planetlab. *SIGOPS Oper. Syst. Rev.*, 40(1), 2006.

[16] PeerSim. `http://peersim.sourceforge.net`.

[17] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed Resource Management for High Throughput Computing. In *HPDC'98*, July 1998.

[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 Trans. Comput. Syst.*, 21(2):164–206, 2003.

[19] J. Rolia, X. Zhu, M. Arlitt, and A. Andrzejak. Statistical Service Assurances for Applications in Utility Grid Environments. Technical Report HPL-2002-155, HP Labs, 2002.

[20] T. Sandholm and K. Lai. A statistical approach to risk mitigation in computational markets. In *HPDC'07*, 2007.

[21] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal. Dynamic Provisioning of Multi-tier Internet Applications. In *ICAC'05*, June 2005.

[22] B. Urgaonkar, P. Shenoy, and T. Roscoe. Resource Overbooking and Application Profiling in Shared Hosting Platforms. In *OSDI'02*, December 2002.

[23] R. Wolski. Experiences with predicting resource performance on-line in computational grid settings. *SIGMETRICS Perform. Eval. Rev.*, 30(4):41–49, 2003.

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

[25] S. Zhong. `http://www.cse.fau.edu/~zhong/software/index.htm`.

[26] S. Zhong and J. Ghosh. A comparative study of generative models for document clustering. In *SDM Workshop on Clustering High Dimensional Data and Its Applications*, 2003.