Platform-of-Platforms: A Modular, Integrated Resource Framework for Large-Scale Services

Rahul Trivedi, Abhishek Chandra and Jon Weissman
Department of Computer Science and Enginnering
University of Minnesota, Minneapolis, MN
{trivedi, chandra, jon}@cs.umn.edu

1. INTRODUCTION

Over the past few years there has been a great deal of research activity in the development of diverse network service platforms ranging from large-scale Internet clusters to peer-to-peer networks and Grids. We believe that as the network service marketplace grows, the requirements and complexity of newly emerging services will require greater synergy between the existing disparate infrastructure alternatives as they each offer different advantages. For example, an Internet service for delivering weather updates could be augmented to provide weather prediction using a computation Grid for simulating weather patterns, while using a P2P network for deep archival of prior weather data, and a cluster to handle front-end requests. A multi-tier resource platform may be a natural fit for such multi-tier network services.

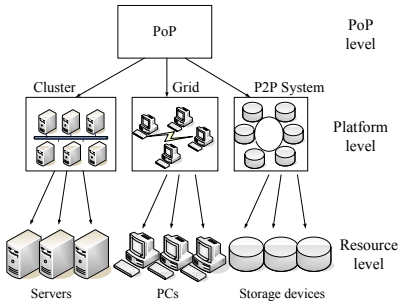

Figure 1: Hierarchical architecture of a PoP: each level corresponds to a different granularity of resource aggregation, ranging from the PoP level to the platform level to the individual resource level.

To this end, we introduce the concept of a *platform-of-platforms (PoP)* that integrates an arbitrary number of cluster, Grid, and P2P platforms into a single resource provider (see Figure 1). We have identified four potential benefits for the PoP: (1) component overflow – if a service component gets overloaded, it can "spill" onto another platform, (2) component matching – different service components naturally fit certain platforms better (see Table 1), (3) fault tolerance – if a platform fails, the component could be moved to another platform, and (4) differentiated QoS – redundant deployment of components to platforms could allow for request-specific servicing (e.g., shorter requests to clusters, longer to Grids, etc.).

2. ARCHITECTURE AND MECHANISMS

Figure 1 illustrates the hierarchical architecture of a PoP, where each level aggregates and manages the resources underneath it. Such a hierarchical architecture provides a natural and efficient way to map multi-tiered services to a PoP. To promote fault tolerance in the extreme, the PoPs are further arranged in an overlay network of resource providers to allow for fully dynamic service deployment that can handle massive failures and unpredictable demand spikes.

A PoP provides certain core mechanisms in this new architectural platform to achieve the above-stated benefits. These core mechanisms include: (1) *matching:* a mechanism by which the requirements of services and their components can be matched to different platform resources, (2) *routing:* routing of service requests based on their resource demands and characteristics to the appropriately matched platform, and (3) *metering:* monitoring and accounting of resource usage for a service distributed over multiple platforms within a PoP. Our goal is to show that a rich set of service- and PoP-specific policies can be implemented on top of these core mechanisms.

	Cluster	Grid
Front-end	657.9-669.6 ms	414.9-858.0 ms
Computation	166.39 s	133.79 s

Table 1: The results show the benefit of matching and platform affinity. Running the BLAST compute component on the Grid provides smaller computation times compared to that on the cluster, while the latency for the front-end component is less variable on the cluster. The results are shown for a 76 MB library and an input sequence of length 569. In both cases, the library is decomposed across the platform nodes.

As a proof-of-concept, we demonstrate the performance of a bioinformatics service deployed on a PoP prototype running on a local cluster and PlanetLab [2]. The service consists of a front-end component and a distributed computational component running the BLAST algorithm. Each of these components can be deployed either on a cluster or a Grid. While the cluster consists of 4 dedicated nodes with predictable performance and bandwidth, the Grid consists of 8 PlanetLab nodes with fluctuating performance and bandwidths. The Grid runs the BOINC middleware [1] to deliver work to the nodes. Table 1 shows the performance of these components for the different deployment options. The results depicted here illustrate that component matching (running the front-end component on the cluster and the compute component on the Grid) can exploit the heterogeneous nature of the PoP effectively to provide overall better service performance.

3. REFERENCES

[1] BOINC: Berkeley Open Infrastructure for Network Computing. `http://boinc.berkeley.edu`.

[2] A. Bavier et al. Operating System Support for Planetary-Scale Network Services. In *Proceedings of NSDI'04*, March 2004.